O'REILLY®

bhv®

Современное системное администрирование

Управление надежными и устойчивыми системами

Дженнифер Дэвис

Modern System Administration

Managing Reliable and Sustainable Systems

Jennifer Davis

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY

Дженнифер Дэвис

Современное системное администрирование

Санкт-Петербург
«БХВ-Петербург»
2024

УДК 004.4
ББК 32.973.26-018.2
Д94

Дэвис Д.

Д94 Современное системное администрирование: Пер. с англ. — СПб.: БХВ-Петербург, 2024. — 304 с.: ил.

ISBN 978-5-9775-1848-2

Книга посвящена современным практикам и технологиям системного администрирования. Приведены основные сведения о системах, архитектурах, вычислительных средах, хранилищах, сетях. Рассмотрены методы и наборы инструментов сисадмина, вопросы контроля версий, тестирования, документирования и представления информации. Описана сборка системы, разработка сценариев, управление инфраструктурой и обеспечение ее безопасности.

Рассмотрен мониторинг системы, программного обеспечения и работы сисадмина. Особое внимание уделено масштабированию системы, управлению мощностями, созданию надежной дежурной службы, управлению инцидентами и планированию системы предупреждений.

Для системных администраторов, инженеров службы поддержки и других ИТ-специалистов

УДК 004.4
ББК 32.973.26-018.2

Группа подготовки издания:

Руководитель проекта	*Евгений Рыбаков*
Зав. редакцией	*Людмила Гауль*
Перевод с английского	*Владимира Мелькина*
Редактор	*Зоя Корниенко*
Компьютерная верстка	*Ольги Сергиенко*
Оформление обложки	*Зои Канторович*

© 2024 BHV
Authorized Russian translation of the English edition of ***Modern System Administration*** ISBN 9781492055211
© 2023 Jennifer Davis.
This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

Авторизованный перевод с английского языка на русский издания ***Modern System Administration***
ISBN 9781492055211 © 2023 Jennifer Davis.
Перевод опубликован и продается с разрешения компании-правообладателя O'Reilly Media, Inc.

Подписано в печать 06.05.24.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 24,51.
Тираж 1200 экз. Заказ № 9435.
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Отпечатано с готового оригинал-макета
ООО "Принт-М", 142300, М.О., г. Чехов, ул. Полиграфистов, д. 1

ISBN 978-1-492-05521-1 (англ.)
ISBN 978-5-9775-1848-2 (рус.)

© Jennifer Davis, 2023
© Перевод на русский язык, оформление.
ООО "БХВ-Петербург", ООО "БХВ", 2024

Оглавление

Вступительное слово

В течение нескольких лет я неоднократно менял работу, пока, наконец, не устроился системным администратором в крупную финансовую организацию. Работники, которые обслуживали серверы и запускали сценарии командной оболочки, сидели на втором этаже, а разработчики, писавшие приложения, обитали на третьем. Я никогда не задумывался, почему нас разделяет лифт или почему мы общаемся в основном через запросы в службу поддержки, а не лично. Такова была иерархическая структура.

Я пришел в компанию во время зимних праздников, и догадайтесь, у кого не было ни дня выходных? Я обнаружил, что сижу один на втором этаже, обрабатывая запросы службы поддержки. Разработчики присылали запросы на развертывание программного обеспечения, а я запускал соответствующие скрипты на каждом сервере и закрывал заявки. В конце концов я устал от этой рутины и написал сценарий, который выполнял основную работу за меня.

Запуск сценариев на множестве машин раньше занимал у меня полчаса. Теперь же я управлялся с этой работой всего за минуту, а то и меньше. Я закрывал появляющиеся заявки почти сразу, как только они появлялись. Однажды кто-то сверху пришел и поинтересовался, как это у меня получается. Я рассказал, как автоматизировал рутинную работу, и в итоге стал получать более интересные задания. Передо мной постоянно ставили новые задачи, и в какой-то момент я решил, что эта беготня между этажами ни к чему, и перетащил свой стол на третий этаж, стерев границы между отделом разработки и отделом эксплуатации.

Однако всякое благое дело наказуемо. Руководство компании посчитало, что я нарушаю существующие процессы, разработанные на основе отраслевых правил, из-за чего у остальных членов команды появляются завышенные ожидания. Ведь в должностной инструкции не предусмотрено, чтобы системные администраторы учились писать программы, помогали специалистам по обеспечению качества автоматизировать тесты или определяли новые способы выполнения задач. Чем больше я выходил за границы дозволенного, тем больше проблем возникало у меня с теми, кто их обозначил.

Примерно через год, выполнив множество успешных проектов, я осознал, почему мой нешаблонный стиль работы был полезен и важен. Я понял, что работать в одиночку невозможно, и хорошие инструменты не заменят хорошую команду. Примерно в это же время мой директор (тот самый, которого я подвел однажды, самовольно подняв планку стандартов предприятия) вернулся с конференции и отозвал

меня в сторону. Он сказал: «Наконец-то я понял, как называется то, что вы делаете. Это DevOps». (*DevOps* — это объединение двух функций, которые обычно выполняются отдельно: разработки и эксплуатации, **https://ru.wikipedia.org/wiki/DevOps**.)

В последнее десятилетие понятие DevOps ошибочно использовалось для описания современного системного администрирования, но на самом деле это лишь один из множества новых подходов, которые мы должны внедрять, чтобы преуспевать в постоянно меняющейся среде. Современное системное администрирование — нечто большее, чем отдельная практика. Его нельзя охарактеризовать с помощью какого-то одного инструмента или учета индивидуальных вкладов участников.

Хотя кому-то могло показаться, что в нашей профессии наконец-то появилась путеводная звезда под названием DevOps. Многие отправились в путешествие, руководствуясь только ею, и заблудились. Эта книга представляет собой карту, на которой отмечены многочисленные отправные точки и пути, которые помогают овладеть современным системным администрированием, и автор лично проверил многие из них. Дженнифер не просто дает указания. Она предоставляет вам контекст, чтобы было понятно, для чего существует данная тропа. Не только, чтобы вы могли идти по следам тех, кто ее проложил, но и для того, чтобы помочь вам наметить свой собственный путь.

– Келси Хайтауэр (Kelsey Hightower)

Предисловие

Когда я начинала работать системным администратором, мои наставники посоветовали мне прочитать «Красную книгу» (второе издание книги «UNIX. Руководство системного администратора» Эви Немет и др. издательства Addison-Wesley) и посетить USENIX LISA — первую конференцию по администрированию больших систем. Большой системой тогда считалась та, где более ста пользователей. Они были правы. Я многому научилась, последовав этим двум советам. «Красная книга» заложила прочную основу для понимания работы определенного оборудования и служб Unix. Она была намного ценнее всех имеющихся руководств благодаря коллективной практической мудрости ее авторов.

А на своей первой конференции USENIX LISA я узнала о важности непрерывного обучения (из руководств Эви Немет (Evi Nemeth) «Актуальные вопросы системного администрирования» (*англ.* «Hot Topics in System Administration»)) и методах документирования (книга Майка Чавареллы (Mike Ciavarella) «Методы документирования для системных администраторов» («Documentation Techniques for SysAdmins»). Я познакомилась с очень многими сисадминами на неформальных собраниях и мероприятиях по обмену информацией типа Birds of a Feather (BOF), да и просто общаясь в кулуарах.

Помимо получения конкретных навыков и изучения технологий, я узнала следующее:

- работа по системному администрированию часто происходит на стыке различных дисциплин, что требует сотрудничества между отдельными командами;
- случайные знания могут оказаться неожиданно полезными;
- в обучении и преподавании существенную роль играют истории. Это своеобразный цемент, скрепляющий разрозненные крупицы знаний и облегчающий их применение.

Я по-прежнему ощущала, что существует некий разрыв, дистанция между тем, каким должно быть системное администрирование, и тем, что я видела на практике у себя на работе. И тогда я поняла: никогда не будет книги, в которой даются исчерпывающие ответы, что делать в каждой конкретной ситуации. Конечно, мы учимся, обмениваясь историями, но каждый из нас прокладывает собственные дорожки при поддержке конкретных систем в определенных средах.

Сегодня системным администраторам приходится изучать и использовать постоянно расширяющийся список технологий и сторонних служб при создании, разверты-

вании и эксплуатации систем, охватывающих тысячи, а иногда и миллионы пользователей.

С учетом этого я хочу поделиться в данной книге некоторыми своими историями, представив набор самых существенных понятий и практических методов, которые помогут вам при сборке, эксплуатации, масштабировании ваших систем и передаче их в дальнейшем другим специалистам.

Для кого предназначена эта книга

Я написала эту книгу для всех опытных системных администраторов, ИТ-специалистов, инженеров службы поддержки и других инженеров по эксплуатации, которые хотят познакомиться с современными практиками и технологиями эксплуатации.

Эта книга также может быть полезна разработчикам, тестировщикам и всем, кто хочет расширить свои навыки работы в данной области. Я понимаю, что иногда в команду входят специалисты, лишь изредка занимающиеся операционно-технологическим обслуживанием. Но им бывает нужно более четко представлять себе работу системы, чтобы эффективно выполнять свои функции.

Я постаралась сосредоточиться на принципах и практиках, которые помогают в работе, связанной с любыми видами современного информационно-технологического обслуживания. В то же время я допускаю, что мое ви́дение этих принципов сформировалось как результат администрирования многочисленных, преимущественно распределенных систем на базе Unix. Вся информация в данной книге актуальна для большинства системных администраторов, но у каждой организации свои потребности, которые и будут определять деятельность этих команд сисадминов. Предположим, например, что ваша работа в основном связана с управлением инфраструктурой на рабочем месте (т. е. точками доступа Wi-Fi, принтерами и телефонами). В этом случае материал в *части III* будет не столь востребован.

О том, чего в этой книге нет

Эта книга не является справочным руководством по инструментам, программным продуктам или определенным операционным системам. Информацию по этим конкретным темам следует искать в других источниках, которых немало. Однако там, где это уместно, я буду рекомендовать некоторые достойные материалы, которые помогут вам повысить свой профессиональный уровень.

Если вы ищете руководство по работе с конкретным инструментом, где пошагово объясняется администрирование некоторой системы, то эта книга не для вас. Существует масса литературы и ресурсов в сети, посвященных операционным системам и приложениям. Вот несколько вариантов, которые я рекомендую.

- По вопросам общего администрирования Unix обратитесь к последней версии «Красной книги» — «UNIX и Linux: руководство системного администратора»

(UNIX and Linux System Administration Handbook), 5-е издание, Трент Р. Хайн и др. (Addison-Wesley).

- Две книги Томаса Лимончелли (Thomas A. Limoncelli) и др., посвященные системному и сетевому администрированию и вобравшие в себя десятилетия коллективного опыта авторов:
  - «Практика системного и сетевого администрирования. Том 1». (*англ.* «The Practice of Systemand Network Administration: Volume 1: DevOps and Other Best Practices for Enterprise IT»), 3-е издание (издательство Addison-Wesley);
  - «The Practice of Cloud System Administration: DevOps and SRE Practices for Web Services: volume 2» (издательство Addison-Wesley).
- Глубоко изучить вопросы, связанные с системами обработки данных, позволит книга Высоконагруженные приложения. Программирование, масштабирование, поддержка» (*англ.* «Designing Data-Intensive Applications») Мартина Клеппмана (Martin Kleppmann), издательство O'Reilly.
- Если вы сосредоточены на управлении микросервисами, ознакомьтесь с книгой «Создание микросервисов», 2-е издание, Сэм Ньюмен (Sam Newman), издательство O'Reilly.

Круг вопросов, который охватывает книга

Как системные администраторы, мы уделяем время процессам на системном уровне и работе системы в целом (за какую бы часть системы мы ни отвечали). Ни у кого нет ответов на все вопросы, но я могу сориентировать вас, указав те методы работы и инструменты, которые помогут вам продвинуться в профессиональном плане, придадут уверенности в себе и позволят наладить контакты с другими людьми, которые движутся в том же направлении, что и вы.

Последнее, о чем хочется сказать

Системы по своей сути запутанны. Но нам хочется верить, что кто-то все же отыскал идеальный способ управления системами и такие процессы и инструменты, которые позволят поддерживать их в первозданном состоянии. Конечно, есть специалисты с опытом, которые могут поделиться полезными рекомендациями, но важно помнить следующее:

- их опыт может быть неприменим к вашей среде или для решения ваших проблем;
- они не знают то, чего не знают. Они могут не подозревать о дополнительных факторах, которые повлияли на их успешный конечный результат;
- их передовые подходы могут быть связаны с тем, что имеющиеся системы соответствуют определенным требованиям и работают определенным образом.

Вы больше не работаете в одиночку. Иногда интуитивный подход может оказаться ошибочным. Технологии развиваются, изменения происходят постоянно, и один человек не в силах учесть все факторы, оказывающие влияние на результат. Вы

можете быть специалистом широкого профиля или иметь глубокие знания в узкоспециализированной области, и все равно этого будет недостаточно. Внедрение практики сотрудничества позволяет вам планировать те или иные действия с учетом различных точек зрения и эффективно управлять системами. Совместная работа с другими людьми приводит к использованию новых подходов, отличающихся от традиционных. В процессе сотрудничества вам нужно будет также разъяснять свои цели, чтобы другие лучше понимали, какую проблему вы решаете, почему важно решить ее, и представляли себе суть вашего процесса.

И еще одно предупреждение

Когда что-то пойдет не так (а это обязательно случится) — вы не должны нести это бремя в одиночку. Ошибки неизбежны. Обращайтесь за помощью. На вас лежит большой груз ответственности за исправную работу систем, что может приводить к проблемам с физическим здоровьем и нервным срывам. Существует множество способов поддерживать работоспособность (и благополучное развитие) ваших систем, не жертвуя при этом собой. Следите за своим здоровьем, чтобы работать, строить карьеру и жить счастливо.

Условные обозначения, используемые в этой книге

В этой книге используются следующие обозначения.

Курсив

Указывает на новые термины, адреса электронной почты, имена и расширения файлов и расширения.

`Моноширинный шрифт`

Используется в листингах программ, а также внутри абзацев для обозначения таких элементов программ, как имена переменных или функций, базы данных, типы данных, переменные среды, операторы и ключевые слова.

`Моноширинный жирный шрифт`

Показывает команды или другой текст, который должен быть набран пользователем слово в слово.

Этот элемент означает совет или предложение.

Этот элемент обозначает общее примечание.

Этот элемент указывает на предупреждение или предостережение.

Благодарности

Писать книги непросто. А написать книгу во время пандемии, когда миллионы людей страдали, а системы здравоохранения по всему миру работали на пределе возможностей, ужасно трудно (особенно когда пишешь книгу об управлении системами).

Я бесконечно благодарна многим людям за то, что они помогли этой книге стать реальностью.

Я благодарна Эви Немет, которая создала культуру обмена опытом и непрерывного обучения в области системного и сетевого администрирования с помощью своих знаменитых книг и обучения на конференциях.

Спасибо тем, кто просмотрел черновики и предоставил отзывы: Крису Деверсу (Chris Devers), Ивонне Лэм (Yvonne Lam), Табите Сэйбл (Tabitha Sable), Бренне Флуд (Brenna Flood), Ами Тобей (Amy Tobey), Тому Лимончелли (Tom Limoncelli), Дэвиду Бланк-Эдельману (David Blank-Edelman), Брайану Смиту (Bryan Smith), Лучиано Сиквейре (Luciano Siqueira), Стивену Рагнарьеку (Steven Ragnarök), Эйлину Фришу (Æleen Frisch), Джесс Мэйлз (Jess Males), Мэту Берану (Matt Beran) и Дональду Эллису (Donald Ellis). Я беру на себя всю ответственность за любые ошибки в окончательном варианте.

Выражаю благодарность Крису Деверсу (Chris Devers). Вы были рядом с нами с момента появления первых черновых набросков глав, предлагая свои идеи, информацию и свое понимание, исходя из личного опыта.

Отдельное спасибо Томоми Имура (Tomomi Imura) за ее невероятно талантливые иллюстрации к этой книге.

Особая благодарность Вирджинии Уилсон (Virginia Wilson) — редактору-консультанту, которая всегда терпеливо помогала мне находить нужные слова. Благодаря ее поддержке эта книга и мои писательские способности значительно улучшились.

Я невероятно благодарна за любовь и терпение, которые моя семья проявляла ко мне на протяжении всего процесса написания книги. Без поддержки Брайана, который вел хозяйство, развлекал Фрэнки и был моим первым читателем, эта книга была бы невозможна. Спасибо тебе, Фрэнки, за то, что не даешь мне забыть о радостях творчества. Я очень люблю вас, Фрэнки (Frankie), Брайан (Brian) и Джордж (George).

Большое спасибо всем, кто принимал активное участие в сообществах USENIX LISA, SREcon, CoffeeOps и DevOps, кто делился своими историями и вносил вклад в развитие отраслевых технологий и практик. С большой признательностью ко всем вам!

Введение в современное системное администрирование

Система представляет собой группу компонентов и связей между ними, образующих единое целое. Чтобы добиться устойчивого управления системами, вы, по сути, пытаетесь выбирать правильный курс среди хаоса доступных маршрутов. Но единственно верной дороги нет, а есть только общие направления, которых вам следует придерживаться на пути к пониманию своих систем. Так вы сможете снизить физические и умственные нагрузки и долгие годы получать удовольствие от работы и решения интересных задач.

Я организовала эту книгу таким образом, чтобы предоставлять ресурсы, которые вам потребуются при переходе к современным технологиям, инструментам и методам системного администрирования. Во введении я покажу несколько целей, к которым следует стремиться и которые помогут вам проложить свой собственный путь к надежной и устойчивой работе ваших систем.

Составьте карту маршрута

Во многих отношениях системные администраторы похожи на туристов, отправляющихся в дикую местность. Как показано на рис. В.1, нам нравится идея, что существует некая карта, которая точно укажет, что и когда делать, и если следовать этим рекомендациям, можно добиться надежной работы системы. Мы воображаем, что путь хорошо освещен и что на карте, которую мы найдем, будут четко обозначены ключевые точки и цели.

Но современное системное администрирование больше напоминает ситуацию, показанную на рис. В.2. Вы можете подготовиться к путешествию с помощью некоего универсального инструментария — основополагающих и ключевых методов сборки, мониторинга и масштабирования любой системы. Невозможно предугадать, какие конкретно инструменты при этом понадобятся и как они будут использоваться, но вы будете готовы принять соответствующие решения и задействовать эти инструменты, когда придет время. И вам не придется делать это в одиночку!

Вы должны адаптировать алгоритмы, позволяющие добиться эффективного системного администрирования, для каждой организации и команды, с которой работаете. Так что и ключевые точки, и цели будут варьироваться.

Вы будете двигаться к ним, не зная заранее о каждом повороте. Даже на одном и том же пути возможно появление новых препятствий: размытой тропы или диких животных, которых вы не хотите потревожить. При системном администрировании вы обязательно столкнетесь с неожиданными проблемами (изгибами и поворо-

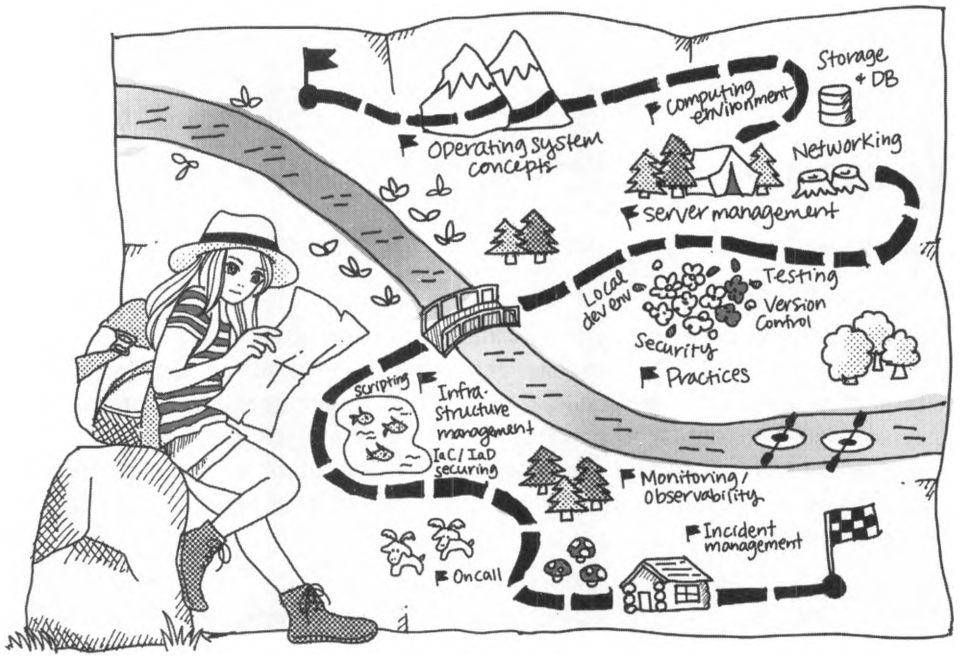

Рис. В.1. Этот рисунок показывает, как мыслит большинство из нас: четкая карта с ясными целями и путешествие в одиночку, стоит только найти нужные ресурсы и научиться нужным вещам. Это не соответствует действительности (иллюстрация Томоми Имура)

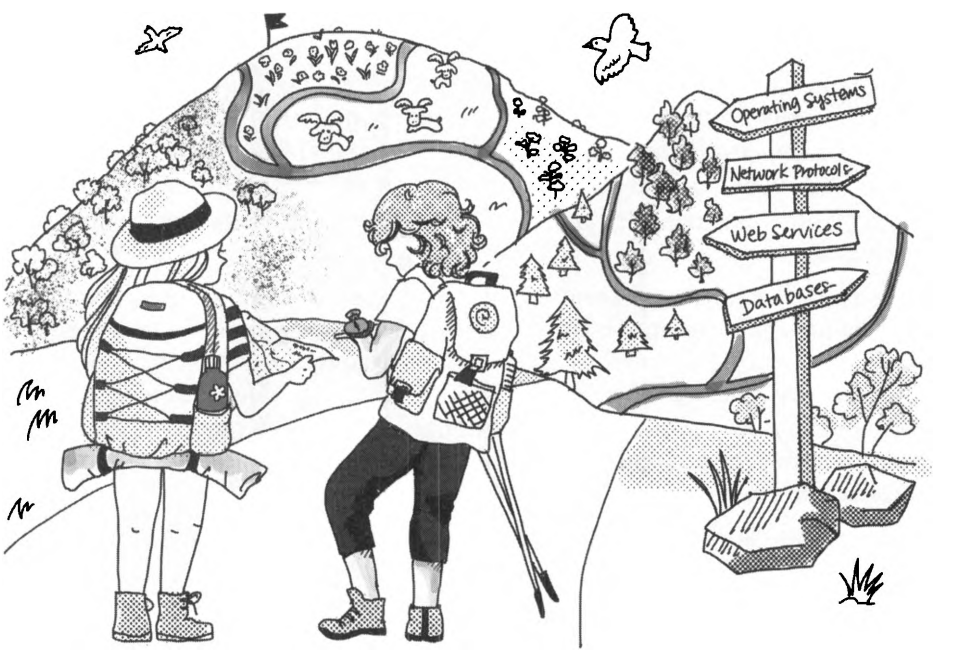

Рис. В.2. Не существует единого ресурса, где приводились бы исчерпывающие инструкции по управлению системами. Дорогу впереди застилает туман, и местность никогда не соответствует карте, но с надежными товарищами и правильно выбранным снаряжением можно отправляться в путь, точно зная, что вы справитесь со всеми перипетиями (иллюстрация Томоми Имура)

тами), которые будут влиять на исход ваших усилий. Поэтому учитесь на своих ошибках, пробуйте разные способы, просите помощи и не останавливайтесь, пока не достигнете цели.

Эта книга поможет вам выработать такой образ мышления и поведения, который позволит направлять ваше время и энергию в нужное русло, чтобы создавать качественные, надежные и устойчивые системы. Масштабы и характер ваших обязанностей могут быть различными. Вы можете отвечать за все и должны будете балансировать между поддержкой организации в целом и решением конкретных инженерных вопросов. Можете управлять информационно-технологической инфраструктурой и тем, как компания ведет бизнес. Можете поддерживать конкретную инфраструктуру для одного продукта.

Когда что-то идет не так, необходимо обеспечивать работоспособность систем, находящихся в вашем ведении, без ущерба для собственного физического и психического здоровья. При достижении поставленных целей перед вами будут открываться новые горизонты. Чтобы строить свою карьеру в долгосрочной перспективе, вам придется прокладывать другие тропы, приспосабливаясь к меняющемуся ландшафту по мере появления новых технологий и методов работы.

Поменяйте свое мышление

Прежде всего необходимо взять на вооружение «мышление роста», которое предполагает, что со временем человек способен развить свои способности и таланты. Вы можете продолжать совершенствовать свои навыки и знания и упорно противостоять трудностям и неудачам.

На протяжении всей книги я рассказываю о различных моделях, чтобы дать вам возможность поразмышлять о системах, которыми вы управляете. Модели облегчают взаимопонимание и представление информации, помогают объяснять концепции, доносить идеи и задают рамки общения. Ни одна из них не является безупречной. Да они и не подразумевают этого. Размышляя о системах, которые описываются при помощи моделей, помните слова Винсента ван Гога: «Модель не является вашей конечной целью»[1], и будьте осторожны, если определенная модель не предоставляет вам крепкую основу для поддержания ваших систем.

Используйте модели типа «инфраструктура как код» и пятиуровневой сетевой для обслуживания, визуализации и объяснения ваших систем. И на основе своего опыта разрабатывайте новые, совершенствуя методы и технологии системного администрирования.

В основе современного системного администрирования лежит тот факт, что сложность и размер ваших систем постоянно возрастают, поскольку «программное обеспечение поглощает весь мир». Для достижения эффективности вы должны

[1] Винсент ван Гог в письме к брату цитирует Диккенса: «Модель является не вашей конечной целью, а средством придать форму и силу вашей мысли и вдохновению» (**https://oreil.ly/5nkDi**).

принять неизбежность изменений и развивать понимание того, как эти изменения влияют на фактическую работу, будь то внедрение новых методов или технологий.

В чем заключается суть работы?

Вы отвечаете за создание надежных и устойчивых систем, их настройку и обслуживание. В роли систем могут выступать конкретные инструменты, приложения или сервисы. В то время как все в организации должны заботиться о безотказной работе, производительности и безопасности систем, вы рассматриваете эти характеристики в рамках ограничений бюджета организации или команды, учитывая конкретные потребности пользователей определенного инструмента, приложения или услуги.

Независимо от того, управляете ли вы сотнями или тысячами систем, вы являетесь сисадмином, если обладаете в системе более высокими привилегиями. К сожалению, многие рассматривают системное администрирование с точки зрения решаемых задач или характера выполняемой работы. Часто это происходит из-за того, что профессиональные обязанности четко не обозначены и на системного администратора взваливается вся работа, которую никто другой делать не хочет.

Многие описывают системного администратора как своего рода цифрового уборщика[2], ответственного за очистку систем, особенно когда они работают не так, как нужно. Хотя роль уборщика в организации очень важна, отождествление этих двух должностей — неуважение к ним обеим.

Системных администраторов можно сравнить скорее с электриками, сантехниками, или специалистами по системам отопления, вентиляции и кондиционированию воздуха. Люди считают само собой разумеющимся, что в современных домах и на предприятиях имеются водопровод, электричество и климат-контроль, но для создания, установки, обслуживания и ремонта этих систем, для их правильной и безопасной работы требуются квалифицированные специалисты.

Другие названия должности

Последние десять лет я испытывала замешательство, когда речь заходила о должностных функциях сисадмина. Существует большая путаница относительно характера его работы. Является ли он оператором? Или это лицо с правами администратора? Появляется огромное количество новых терминов и наименований этой должности, т. к. многих не устраивают связанные с ней стереотипы. Так что, когда кто-то сказал мне: «Я не сисадмин, а инженер по инфраструктуре», — я поняла, что не только я это ощущаю.

Знайте, что организации переименовывают должности, связанные с системным администрированием, чтобы соответствовать изменениям, происходящим в отрасли. Поэтому не лишайте себя возможностей из-за названия должности.

---

[2] Ознакомьтесь с многочисленными функциями системных администраторов в Приложении В книги Томаса Лимончелли и др. «Практика системного и сетевого администрирования» («The Practice of System and Network Administration», Volume 1: DevOps and Other Best Practices for Enterprise IT, 3rd edition (Addison-Wesley)) (**https://oreil.ly/JYWCK**).

Виды системного администрирования

Специалисты по управлению системами называются по-разному (например, сисадмин, SRE-инженер[3], инженер DevOps, инженер по платформам, инженер облачных технологий — это лишь некоторые из наименований). Название должности может указывать на то, какие навыки предпочтительны. Ожидается, например, что SRE-инженер является также инженером-программистом с навыками операционной деятельности.

В отношении инженеров DevOps часто предполагается, что они должны владеть как минимум одним современным языком программирования и обладать экспертными знаниями в области непрерывной интеграции и развертывания. Чаще всего это просто название, и не всегда оно трактуется единообразно. Иногда команда определяет данную роль совершенно иначе и ждет особых навыков, исходя из потребностей организации. Оценивая, подходит ли вам та или иная должность, обратитесь непосредственно к команде, чтобы избежать недоразумений. Например, в одних организациях под аббревиатурой SRE подразумевается инженер по доступности площадок, систем и сервисов, а в других — специалист, обеспечивающий устойчивость систем и их адаптивность к нестандартным и аварийным ситуациям.

Внедрение постоянного развития

Как инженерная дисциплина системное администрирование соединяет в себе искусство и науку. Это подход к работе (проектированию, созданию и мониторингу систем), который учитывает последствия для безопасности, человеческий фактор, государственное регулирование, целесообразность и стоимость мер. Могут существовать сотни различных способов выполнения какой-либо задачи. Ваши знания, навыки и опыт подскажут, какой из многочисленных путей следует выбрать. Вы будете использовать свои аналитические способности для мониторинга воздействия и достигнутых результатов, определять, когда следует расходовать (или экономить) деньги или время, и оценивать объем человеческих ресурсов, необходимых для поддержки системы.

Внедрение практических подходов

По мере развития технологий изменяются и методы управления ими. Будьте готовы осваивать новые подходы, чтобы быть в курсе изменений платформ, что позволит уменьшить воздействие на систему и сохранить удобство ее сопровождения.

Когда вы измеряете надежность вашей системы и организация меняет специалистов, отвечающих за повышение надежности, базовая динамика взаимодействия сисадминов и разработчиков также меняется. Сегодня повышение надежности про-

---

[3] Узнайте больше о профессии SRE-инженера из статьи в блоге Элис Гольдфус (Alice Goldfuss) «Как попасть в SRE» («How to Get into SRE», **https://oreil.ly/wALwU**) и статьи в блоге Молли Струве (Molly Struve) «Что значит быть SRE-инженером» («What It Means to Be a Site Reliability Engineer», **https://oreil.ly/35Es6**).

дукта чаще всего является общим делом, и задача по поддержке системы или сервиса не возлагается целиком и полностью на одну команду. Команды SRE-специалистов призваны оказывать помощь в уменьшении трудозатрат в системах[4].

Внедрение совместной работы

Темпы изменений, сложность наших сред и неизбежные риски нарушения их работы требуют следующих действий:

- объединение экспертных знаний из разных областей (например, разработка, эксплуатация, безопасность и тестирование);
- интеграция предложений вместо поиска компромисса, чтобы окончательное решение учитывало множество точек зрения.

Требуются серьезные усилия, чтобы создать доверительные отношения и психологически безопасную среду, побуждающие людей высказывать свое мнение и точки зрения. Когда члены команды преодолевают психологический барьер в отношении друг друга, они чувствуют себя достаточно спокойно, чтобы принимать риски и подвергать себя уязвимости. Например, человек в подобных условиях будет заблаговременно обращаться за помощью. Это может способствовать предотвращению сбоев благодаря налаженной системе взаимной поддержки.

Поощряйте культуру, которая содействует и оказывает поддержку людям, задающим зондирующие вопросы, чтобы все могли прийти к общему пониманию (мы работаем над одной целью), и поощряет интеллектуальную смелость (даже эксперты ошибаются). Некоторые вопросы, которые могут быть заданы:

- Почему? Почему мы это делаем? Почему это работает именно так?
- Не могли бы вы помочь мне понять вашу точку зрения?
- Какие еще способы решения этой проблемы вы придумали?

Узнайте больше о психологической безопасности — ключевом факторе для высокоэффективных команд, которую специалисты Google, занимающиеся наймом сотрудников, выявили в ходе исследовательской программы re:Work (**https://oreil.ly/uTpZU**).

Внедрение совместной работы ведет к хорошим отношениям с другими людьми, так что когда вам понадобится помощь, ваши коллеги обязательно окажут вам поддержку.

Курс на устойчивость

Устойчивость системы характеризует возможность людей в этой системе преуспевать и вести здоровый образ жизни во время работы. Независимо от масштаба и характера вашей работы, ее устойчивость определяют восемь параметров:

[4] Узнайте о сокращении трудозатрат и его влиянии на команды из статьи Стивена Торна (Stephen Thorne) на сайте Medium, посвященной принципам SRE (**https://oreil.ly/SpiwZ**).

Производительность

Характеризует способность системы выполнять полезную работу в течение определенного периода времени. Производительность системы определяется поразному в зависимости от того, какую услугу или продукт вы создаете.

Масштабируемость

Характеризует способность системы адаптироваться к добавлению и удалению отдельных компонентов.

Доступность

Показывает продолжительность времени, в течение которого система функционирует должным образом.

Надежность

Показывает, насколько стабильно система выполняет свою конкретную задачу в течение определенного периода времени.

Удобство сопровождения

Характеризует простоту развертывания, обновления и завершения использования системы.

Простота

Характеризует легкость понимания системы новым инженером.

Удобство использования

Показывает, насколько пользователи удовлетворены системой.

Наблюдаемость

Определяет, насколько легко понять, что не так с системой, находящейся под наблюдением. Отмечу, что не для всех систем требуется высокий уровень наблюдаемости.

В следующих главах я расскажу о различных технологиях и методах, которые позволяют улучшить данные целевые показатели и в конечном счете повысить устойчивость ваших систем.

Заключение

Ваша стратегия будет зависеть от имеющихся систем и людей, которые поддерживают эти системы. Никто не может предоставить образцовый контрольный список, который укажет, что и когда следует изучить или сделать. Тем не менее вы можете лучше подготовиться, имея соответствующий набор инструментов (понимание основ и ключевых методов работы, а также вопросов сборки, мониторинга и масштабирования систем).

Функции системного администратора постоянно меняются. Поэтому было бы полезно взять на вооружение «мышление роста» и развивать таланты и навыки, кото-

рые помогут строить карьеру в долгосрочной перспективе, задействуя новые технологии и методы работы.

Обращайтесь за помощью и опирайтесь на практики сотрудничества, которые позволят вам эффективно работать с командой, создавая психологическую безопасность. Используйте модели для обоснования своего понимания и опирайтесь на них для развития практики системного администрирования.

Держите курс на устойчивость. Вы заслуживаете того, чтобы добиться успеха и сделать блестящую карьеру, поддерживая системы, которыми управляете.

ЧАСТЬ I

Рассуждения о системах

В первых четырех главах приводятся основные сведения о системах и о том, какие варианты выбора существуют. Но не стоит рассуждать о решениях с точки зрения наилучшего выбора. Вместо этого нужно разобраться, какие из них оптимальны именно для вашей ситуации. Критерии оценки здесь опираются на изменчивый контекст: конфликтующие между собой цели, люди и различные компоненты, которые обеспечивают функционирование системы. «Системное мышление» побуждает вас думать о частях системы и об их взаимодействии в контексте ваших текущих проблем, что дает более полное понимание эволюции системы во времени.

Закономерности и взаимосвязи

Представьте, что вы с подругой готовите торт. Вы следовали рецепту, смешивая все ингредиенты (масло, муку, яйца и сахар), и торт выглядит отлично, но, попробовав его, вы понимаете, что вкус совсем не тот. Чтобы стать опытным кондитером, вы должны хорошо разбираться в ингредиентах (соотношении муки и жира и т. д.) и понимать, как они действуют вместе, оказывая влияние на качество готового продукта (например, вкус и текстуру). Так, наши кулинары на рис. 1.1 не знали, что кунжутное масло для торта не годится.

Теперь поменяйте пекаря на сисадмина, а идеальные пропорции ингредиентов — на взаимосвязанные компоненты вашей системы (например, смартфоны, встроенные устройства, мощные серверы и массивы хранения данных). Чтобы стать опытным системным администратором, вам необходимо понимать, как работают компоненты в сочетании друг с другом и как это влияет на качество вашей системы (например, надежность, масштабируемость и удобство сопровождения).

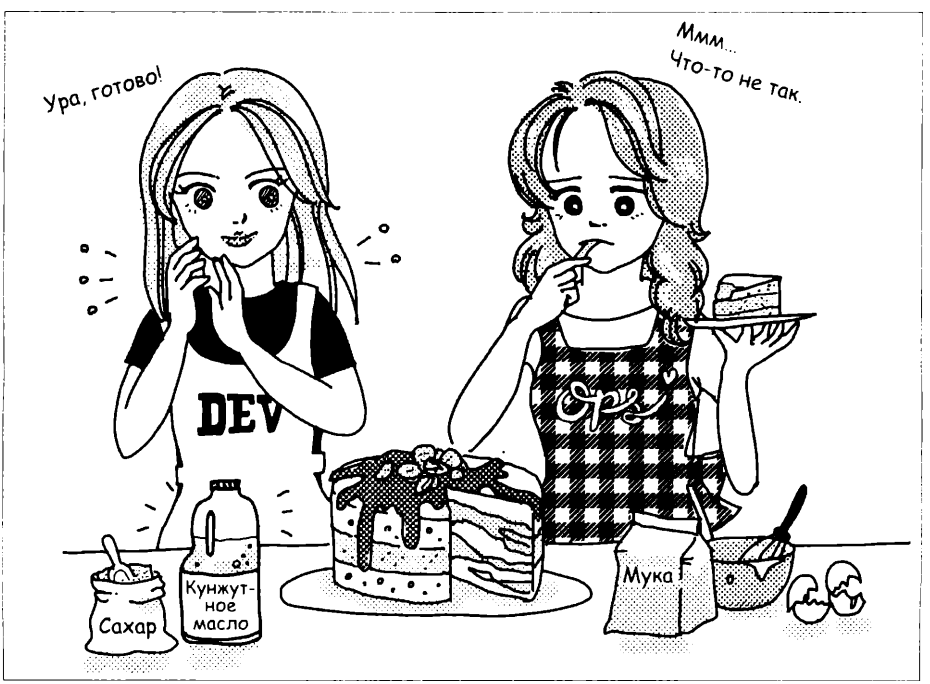

Рис. 1.1. Модель понимания системы (иллюстрация Томоми Имура)

В этой главе я помогу вам поразмышлять о ваших системах, чтобы вы увидели присущие им закономерности и взаимосвязи и уяснили, чем обоснован выбор компонентов при их проектировании.

Как соединять ингредиенты

Для работы с типичными рабочими нагрузками (например, пакетная обработка, веб-серверы и кеширование) инженеры выбирают *шаблоны архитектуры* — решения, используемые многократно. Это модели, которые отражают «коллективное понимание устройства системы»[1].

Многократно используемые решения развиваются повсеместно, в локальных и облачных вычислительных средах, позволяя поддерживать множество сервисов за счет их разбиения на микросервисы[2]. Такие шаблоны обеспечивают надежность, масштабируемость и удобство сопровождения систем, определяя компоненты системы и связи между ними.

Давайте рассмотрим три распространенных шаблона архитектуры, применяемых при проектировании систем, чтобы стало понятно, как они влияют на свойства системы (надежность, масштабируемость и ремонтопригодность), ее развитие и какие ограничения накладывают.

Многоуровневая архитектура

Наиболее распространенным и знакомым является шаблон многоуровневой или многослойной архитектуры общего назначения. Инженеры широко используют его для клиент-серверных приложений, таких как веб-серверы, электронная почта и др.

Компоненты сгруппированы в горизонтальные слои, каждый из которых выполняет свою функцию, что позволяет разграничить проблемные аспекты различных уровней. Слои обычно тесно связаны между собой за счет обработки запросов и откликов от смежных слоев. В результате компоненты в пределах каждого слоя можно обновлять и развертывать независимо от других.

Двухуровневая система включает клиент и сервер. Трехуровневая система состоит из клиента и еще двух уровней серверов; уровни представления, приложений и данных включают в себя различные компоненты. Каждый уровень также может быть разделен на отдельные логические уровни в многоуровневой системе. В зависимости от потребностей системы (например, отказоустойчивость, безопасность и масштабируемость) в ней может быть более трех уровней. По мере увеличения их числа масштабируемость и надежность возрастают за счет дополнительного разделения процессов, которые могут быть развернуты и обновлены независимо друг от друга.

---

[1] Мартин Фаулер (Martin Fowler), «Software Architecture Guide» (**www.martinfowler.com/architecture**)

[2] Узнайте больше о декомпозиции сервисов из главы 3 книги Сэма Ньюмана (Sam Newman) «Создание микросервисов» («Building Microservices», O'Reilly, **https://oreil.ly/SSx0B**)

Микросервисная архитектура

Микросервисная архитектура — это распределенная архитектура, которая состоит не из уровней, а из набора небольших независимых друг от друга единиц бизнес-кода. Микросервисы — это небольшие автономные сервисы. Поскольку сервисы являются обособленными, программный код для любого из них можно разрабатывать и внедрять независимо от остальных. Кроме того, каждый сервис может использовать тот язык или структуру, которые наилучшим образом подходят для заданного сценария использования.

Микросервисы повышают масштабируемость и надежность системы, поскольку их можно разворачивать независимо друг от друга по мере необходимости, и они изолированы от точек отказа в системе.

При этом декомпозиция сервиса на микросервисы снижает удобство сопровождения из-за увеличения когнитивной нагрузки на сисадминов. Чтобы понимать систему, необходимо знать детали каждого отдельного сервиса (т. е. языки, фреймворки, конвейеры сборки и развертывания, а также все соответствующие среды).

Управляемая событиями архитектура

Управляемая событиями архитектура — это шаблон распределенной асинхронной архитектуры, в которой приложения слабо связаны между собой и не имеют подробной информации друг о друге. Вместо этого они взаимодействуют опосредованно, на основе публикации событий и подписки на них.

События — это то, что происходит, факт, который можно отследить[3]. Системы генерируют события. В управляемых событиями системах источники событий создают события, брокеры принимают их, а потребители прослушивают и обрабатывают события.

Существуют две основные модели управляемых событиями систем: обмен сообщениями (pub/sub, оно же publisher-subscriber, оно же издатель-подписчик) и потоковая передача.

В системе обмена сообщениями источник событий или издатели помещают события в брокер сообщений. Брокер отправляет все эти события потребителям событий или подписчикам. Он получает публикуемые события от издателей, поддерживает их очередность, делает доступными для подписчиков и удаляет после потребления.

В системе потоковой передачи события публикуются в распределенном журнале — долговременном хранилище данных, в которое информация только добавляется. В результате потребители получают события из нужного им потока и могут воспроизводить их. Кроме того, распределенный журнал сохраняет события после их потребления, так что новые подписчики могут подписаться на события, произошедшие ранее.

---

[3] CloudEvents — проект, нацеленный на разработку спецификации для описания данных о событиях стандартным способом, который может быть реализован в различных сервисах и платформах (**https://cloudevents.io**).

Поскольку компоненты слабо связаны друг с другом, одним частям системы не нужно беспокоиться о здоровье других. Слабо связанные элементы повышают отказоустойчивость всей системы, поскольку их можно разворачивать и обновлять независимо от остальных. Хранение записей о событиях позволяет воспроизводить произошедшие события в случае сбоя.

* * *

В табл. 1.1 приведены сравнительные характеристики надежности, масштабируемости и удобства сопровождения трех распространенных архитектурных шаблонов, которые встречаются в системах.

Таблица 1.1. *Сравнительные характеристики надежности, масштабируемости и удобства сопровождения нескольких архитектурных шаблонов*

	Многоуровневая	**Микросервисы**	**Потоковая передача**
Надежность	Средняя	Высокая	Высокая
Масштабируемость	Средняя (ограничена в пределах слоев)	Высокая	Высокая
Удобство сопровождения	Высокая (системы с тесной связью)	Низкое (снижение тестируемости)	Среднее (уменьшение простоты)

Конечно, это не единственные шаблоны, которые встречаются при проектировании систем. Посетите веб-сайт Мартина Фаулера (Martin Fowler) the Software Architecture Guide, освещающий многие вопросы (**https://oreil.ly/Sf5lC**).

Как взаимодействуют компоненты

Компоненты системы существуют не изолированно, каждый из них взаимодействует с другими, и это взаимодействие может определяться архитектурным шаблоном (REST (**https://oreil.ly/CmRCT**) — для N-уровневой архитектуры, gRPC (**https://oreil.ly/MzO9n**) — для архитектуры, управляемой событиями). REST — акроним от английского REpresentational State Transfer, «передача состояния представления». gRPC — высокопроизводительный фреймворк, разработанный компанией Google для вызова удаленных процедур (RPC).

Существует несколько моделей для представления взаимодействия компонентов, например:

- сетевая модель;
- пятиуровневая модель сети;
- пятиуровневая эталонная модель TCP/IP;
- модель TCP/IP.

Хотя эти модели очень похожи, у них есть небольшие различия, которые могут показать, как люди воспринимают приложения и сервисы, которые разрабатывают для взаимодействия друг с другом.

Когда какой-то инженер или группа специалистов определяют область, требующую улучшения, они составляют рабочее предложение (RFC) и отправляют его на рецензирование. Являясь открытым международным сообществом, которое поддерживает работу сети Интернет, улучшает ее архитектуру, удобство использования и сопровождения, обеспечивает функциональную совместимость, рабочая группа по стандартам для сети Интернет (IETF, **https://oreil.ly/ydfJn**) принимает некоторые из поступающих предложений в качестве технических стандартов, определяющих официальные спецификации и протоколы. Спецификации протоколов определяют, как устройства взаимодействуют друг с другом, следуя при этом в общих чертах модели Интернета. И эти протоколы продолжают развиваться по мере расширения сети Интернет и изменения потребностей пользователей (пример показан в *приложении Б*).

Как показано в таблице 1.2, пятиуровневая модель Интернета включает в себя пять дискретных уровней. Каждый уровень взаимодействует со смежными посредством собственного объекта сообщения. Такая иерархия позволяет распределить задачи по уровням и независимо создавать или изменять отдельные части системы, что также способствует дифференциации каждого уровня.

Таблица 1.2. *Пятиуровневая модель сети Интернет и примеры протоколов*

Уровни	Примеры протоколов
Уровень приложений	HTTP, DNS, BGP
Транспортный уровень	TCP, UDP
Сетевой уровень	IP, ICMP
Канальный уровень	ARP
Физический уровень	Медный кабель, оптоволоконный кабель, Wi-Fi

Как и в торте на рис. 1.1, здесь нет четких границ между уровнями, которые позволяли бы точно локализовать какую-либо проблему. При реализации протоколов производители не обязаны строго следовать спецификациям, и уровни могут перекрываться. Например, протокол Border Gateway Protocol (BGP) определяет самый быстрый и эффективный маршрут для передачи данных. Он реализован таким образом, что можно относить его как к прикладному, так и к транспортному уровню.

Уровни в модели сети Интернет дают вам возможность очертить контекст и сосредоточиться на компонентах вашего приложения — исходном коде и зависимостях, или же на более низком физическом уровне. Такой подход упрощает сложную модель коммуникаций, разделяя ее на понятные фрагменты. Однако изредка вы будете сталкиваться с ситуациями, когда и сужение контекста не поможет вам понять, что происходит. Чтобы лучше разбираться в деталях, вы должны знать, как все это, вместе взятое, влияет на характеристики вашей системы.

Дополнительные сведения о протоколах

Если ваша работа связана с управлением сетями или реализацией протоколов, ознакомьтесь с книгами: Эндрю Таненбаум (Andrew Tanenbaum) «Компьютерные сети» (Pearson); Кевин Фол, Ричард Стивенс (Kevin Fall, Richard Stevens) «Иллюстрированный TCP/IP. Том 1: Протоколы (Вып. 2)» (Addison-Wesley), и посетите сайт the RFC Series (**https://oreil.ly/1DYQT**).

Давайте более подробно рассмотрим прикладной, транспортный, сетевой, канальный и физический уровни.

Прикладной уровень

Начнем с верхнего, прикладного уровня модели сети Интернет. Прикладной уровень описывает все высокоуровневые протоколы, с которыми приложения обычно взаимодействуют напрямую. Протоколы этого уровня определяют, как приложения взаимодействуют с нижележащим транспортным уровнем при отправке и получении данных.

Чтобы понять работу этого уровня, сосредоточьтесь на библиотеках или приложениях, которые реализуют протоколы в основе вашего приложения. Например, когда клиент посещает ваш сайт с помощью популярного браузера, выполняются следующие действия:

1. Браузер обращается к специальной библиотеке, чтобы получить IP-адрес вебсервера с помощью системы доменных имен (DNS).
2. Браузер инициирует HTTP-запрос.

Протоколы DNS и HTTP работают на уровне приложений модели сети Интернет.

Транспортный уровень

Следующий уровень модели сети Интернет — транспортный. Он обрабатывает поток информации между хостами. Здесь также имеются два основных протокола: Transmission Control Protocol (TCP) и User Datagram Protocol (UDP).

Исторически сложилось так, что UDP стал основой для более простых протоколов, таких как ping/ICMP, DHCP, ARP и DNS, в то время как на базе TCP создавались более продвинутые протоколы, такие как HTTP, SSH, FTP и SMB. Однако эта ситуация меняется, поскольку те качества, которые делали протокол TCP более надежным на уровне сеансов, приводят к уменьшению производительности в некоторых контекстах.

UDP — это протокол без сохранения информации о состоянии клиента на сервере, который прилагает все усилия, чтобы передавать сообщения, но не гарантирует их доставку получателям. TCP, с другой стороны, является протоколом, ориентированным на соединение, который использует трехэтапное рукопожатие для установления надежного сеанса связи с узлом сети.

Основные характеристики протокола UDP следующие.

Не ориентирован на установление соединения

UDP не является сеансовым протоколом. Сетевые узлы могут обмениваться пакетами без предварительного установления сеанса связи.

Отсутствие гарантии доставки данных

Функции обнаружения или исправления ошибок не поддерживаются. В приложениях должны предусматриваться собственные механизмы отказоустойчивости.

Отсутствие блокировки

Протокол TCP не способен справиться с блокировкой начала очереди, когда сеансы зависают из-за пропущенных пакетов или нарушения порядка получения данных. Это приводит к необходимости повторной передачи с того места, где возникла ошибка. При использовании UDP нарушение порядка доставки данных не является проблемой, и приложения могут выборочно запрашивать повторную передачу отсутствующих данных без повторной отправки тех пакетов, которые были успешно доставлены.

Основные характеристики протокола TCP:

Подтверждение приема данных

Получатель уведомляет отправителя о получении данных каждого пакета. Это не гарантия того, что приложение приняло или обработало данные, а просто подтверждение, что они прибыли по назначению.

Ориентация на установление соединения

Отправитель устанавливает сеанс перед отправкой данных.

Надежность

TCP отслеживает передаваемые и принимаемые данные. Сообщения получателя, подтверждающие прием данных, могут теряться. Но если очередность поступления сегментов нарушается, получатель не будет подтверждать их прием. В этом случае последний пакет, поступивший в порядке очереди, или ряд пакетов отправляются повторно. Более высокая надежность может приводить к задержкам при передаче данных.

Управление потоком

Получатель уведомляет отправителя о том, какой объем данных может быть принят.

Обратите внимание, что безопасность не была частью проекта или неотъемлемой характеристикой протоколов TCP и UDP. Напротив, недостаточная безопасность в первоначальных версиях этих протоколов привела к дополнительным сложностям в реализации приложений и систем и дальнейшим изменениям в протоколах.

Сетевой уровень

Расположенный посередине сетевой уровень связывает транспортный и канальный уровни, обеспечивая доставку пакетов на основе уникального иерархического IP-адреса.

Интернет-протокол (англ. Internet Protocol, IP) отвечает за правила адресации и разбиение данных на фрагменты при обмене информацией между двумя система-

ми. Он обеспечивает уникальную идентификацию сетевых интерфейсов для доставки пакетов данных с помощью IP-адресов. Протокол IP разбивает и вновь собирает пакеты по мере необходимости, передавая данные по каналам связи с меньшим максимально допустимым размером пакета (MTU). IPv4 — это наиболее распространенная версия IP-протокола, использующая 32-битное адресное пространство. Адрес в нем представлен в виде строки из четырех восьмизначных двоичных чисел (октетов) или четырех десятичных чисел, разделенных точками (точечная десятичная нотация или квадронотация). Стандарт IPv6 дает такие преимущества, как 128-битное адресное пространство, расширенные возможности маршрутизации и улучшенная поддержка групповой адресации. Однако внедрение IPv6 замедлилось — отчасти потому, что IPv4 и IPv6 несовместимы, и мириться с недостатками IPv4 было в целом проще, чем приводить все к новому стандарту.

Адреса в десятичной нотации, соответствующие двоичному представлению, для IPv4 лежат в диапазоне от 0 до 255. Кроме того, в документах RFC определены зарезервированные диапазоны IP-адресов для частных сетей (**https://oreil.ly/QkHVN**). Пакеты из этих сетей не маршрутизируются через общедоступный Интернет.

Основная функция протокола IP сетевого уровня — предоставление уникальных адресов узлам сети для их взаимодействия между собой. Но он не отвечает за передачу данных на канальном уровне и за управление сеансами, осуществляемое на транспортном уровне.

Канальный уровень

Следующий, канальный уровень использует для отправки и получения пакетов физический уровень.

Протокол разрешения адресов (*Address Resolution Protocol*, ARP) предназначен для определения аппаратного (физического) адреса устройства по известному IP-адресу. Аппаратные адреса известны также как MAC-адреса (*media access control* — управление доступом к среде). Каждому *контроллеру сетевого интерфейса* (NIC, *network interface controller*) присваивается уникальный MAC-адрес.

Предполагается, что MAC-адреса являются глобально уникальными. На этом основана работа устройств сетевого управления и программного обеспечения, отвечающих за проверку подлинности и безопасность устройств. Наличие одинаковых MAC-адресов в одной сети приводит к проблемам. Однако дублирующиеся MAC-адреса все же появляются из-за производственных ошибок при изготовлении оборудования (или намеренного изменения (рандомизации) MAC-адреса с помощью ПО)[4].

Они могут появляться также в виртуализированных системах — например, виртуальных машинах, клонированных из эталонного образа. Как бы то ни было, если

[4] Больше о проблемах при рандомизации MAC-адресов — в статье «MAC Address Randomization: Privacy at the Cost of Security and Convenience» (**https://oreil.ly/31Hsf**).

несколько сетевых узлов сообщают о наличии одного и того же MAC-адреса, это приводит к ошибкам и замедлению работы сети.

Настоящий MAC-адрес можно изменить с помощью программного обеспечения. Этот способ известен как *MAC-спуфинг*. Некоторые злоумышленники с помощью подмены MAC-адресов проводят атаку канального уровня, чтобы попытаться перехватить обмен данными между двумя системами для взлома одной из них.

Протокол обратного определения адресов (*Reverse Address Resolution Protocol*, RARP) проверяет соответствие IP-адресов аппаратным адресам и помогает выяснить, не отвечают ли несколько IP-адресов на один MAC-адрес. Если вы предполагаете, что два устройства в сети используют один и тот же IP-адрес — возможно, из-за того, что кто-то назначил статический IP, когда DHCP-сервер уже выделил тот же адрес другому хосту, RARP поможет определить виновника.

Физический уровень

Физический уровень преобразует поток двоичных данных верхних уровней в электрические, световые или радиосигналы, которые передаются через основное физическое оборудование. В каждой физической среде своя максимальная длина и скорость.

Даже при использовании облачных услуг вам придется заботиться о физическом уровне, несмотря на отсутствие стеллажей со множеством физических серверов. Увеличение задержки может быть связано с физической маршрутизацией между двумя точками. Например, если что-то случится с центром обработки данных (ударит молния или кабель повредят белки, **https://oreil.ly/miV4u**), ресурсы могут быть перенаправлены на альтернативные сервисы, расположенные дальше. Среди других вероятных причин — перезагрузка сетевого оборудования в центре обработки данных, износ кабеля или неисправная сетевая плата. В результате запросы, отправленные через эти физические компоненты, могут испытывать повышенную задержку.

Заключение

В своей среде вы обнаружите смешение различных шаблонов (многоуровневых, на основе микросервисов и управляемых событиями) и протоколов. Понимание архитектуры системы позволяет уяснить связь и принципы взаимодействия компонентов, а ваши требования влияют на надежность, удобство сопровождения и масштабируемость системы.

В следующей главе я расскажу, как следует рассматривать эти шаблоны и взаимосвязи и как они влияют на выбор основного программного обеспечения для вашей организации.

Вычислительные среды

Давайте углубимся в основы знаний о системах, начиная с вычислительной среды.

В этой главе рассматривается фундаментальный строительный блок системы — compute. Термин *«compute» (вычисление)* описывает некоторый объект в контексте связанных с ним ресурсов (таких как вычислительная мощность, память, хранилище и сеть). Современная инфраструктура не сводится только к технической реализации системы. Она включает также методы совместной работы при создании, настройке и развертывании вычислительных ресурсов, необходимых вашей организации. В этой главе мы рассмотрим способы, которые позволят распознать тип и среду инфраструктуры вычислительных ресурсов, чтобы вы могли адаптировать свой выбор к потребностям и технологиям своей организации или команды.

Распространенные рабочие нагрузки

Рабочие нагрузки характеризуются объемом и типом воздействия на ресурсы системы, которое оказывает приложение.

Системы, которыми вы управляете, будут иметь ряд приложений или сервисов, которые необходимо устанавливать, поддерживать и запускать в производственных средах. Все приложения или сервисы, которые вы поддерживаете, делятся на категории в зависимости от минимального и рекомендуемого набора требований к вычислительным ресурсам (процессор, память и хранилище): одним необходима высокая вычислительная мощность центрального процессора (CPU-bound), другим — большой объем памяти (memory-bound), а третьи интенсивно используют хранилища данных (storage-bound).

Мы поговорим более подробно о хранилищах и сетях в главах 3 и 4 соответственно.

Приложения, нуждающиеся в процессорной мощности, выигрывают от использования высокопроизводительных процессоров. В число таких рабочих нагрузок входят:

- пакетная обработка данных:
- игровые серверы:
- высокопроизводительные вычисления (high-performance computing, HPC);

- преобразование форматов медиафайлов;
- машинное обучение;
- научное моделирование;
- веб-серверы.

Приложения, активно использующие память, выигрывают от увеличения ее доступного объема. Большую часть времени они выполняют чтение и запись данных в память. В число таких рабочих нагрузок входят:

- кеширование данных;
- аналитическая обработка данных;
- базы данных.

Приложения, связанные с хранением данных, выигрывают от низких задержек при доступе к хранилищам и высокой скорости операций произвольного ввода-вывода. В число таких рабочих нагрузок входят:

- хранилища данных;
- озера данных;
- базы данных;
- распределенные файловые системы;
- Hadoop (база данных, предназначенная для работы с большими данными);
- приложения для ведения журнала или обработки данных.

Информация о типах рабочих нагрузок в вашей системе позволяет оценить, какие компоненты потребуются при создании системы. Варианты выбора для облачных архитектур часто представлены в виде систем, оптимизированных для этих рабочих нагрузок, т. е. наилучших с точки зрения использования ресурсов процессора, памяти и хранилища.

Нет необходимости скрупулезно подбирать вычислительные ресурсы при создании системы в облаке. В рамках ограничения бюджета вы можете остановиться пусть на неидеальных, но вполне подходящих вариантах. В главе 11 рассказывается о моделировании инфраструктуры с помощью конфигурационных файлов (инфракод).

Выбор места размещения рабочих нагрузок

Ваши вычислительные среды могут размещаться на серверах, расположенных в частных центрах обработки данных, которые обслуживаются и управляются силами самих владельцев. Они известны как локальные вычислительные среды (*on-prem*). Или же вы можете сделать выбор в пользу облачных вычислений (*cloud computing*), воспользовавшись предложениями поставщиков соответствующих услуг.

Новые способы ведения дел

Управление центром обработки данных, начиная с поиска оборудования и заканчивая его развертыванием и дальнейшим управлением, было интересной главой в моей карьере. Я руководила закупкой оборудования на миллионы долларов для поддержки среды разработки, которая требовала тестирования и оценки широкого спектра устройств хранения данных и сетевых устройств. Благодаря этому опыту я поняла, что у меня хорошо получается разбираться в сложных вопросах управления проектами и поставщиками и ориентироваться в графиках поставок и монтажа. Но мне совсем не нравится вся эта бумажная волокита и монотонная работа по прокладке кабелей и отслеживанию компонентов системы. Я очень рада появлению облачных вычислений и легкодоступных вычислительных ресурсов.

В главе 17 я расскажу о том, как вести наблюдение за своей работой и находить ту, которая нравится. Очень легко попасть в ловушку и делать то, что у вас хорошо получается, не задумываясь о том, что работа может не приносить удовлетворения.

Локальные вычислительные среды

В локальных вычислительных средах вы либо арендуете, либо приобретаете оборудование для размещения сервисов, отвечающих потребностям вашей организации.

Для каждого приложения можно развертывать различные ресурсы, в зависимости от его рабочей нагрузки. Стандартизированное оборудование упрощает развертывание и настройку, но для одних приложений его производительность может оказаться недостаточной, а для других — избыточной. Развертывание оборудования, соответствующего потребностям конкретного приложения, способствует более эффективному использованию потенциала этих ресурсов, но повышает сложность управления инфраструктурой.

Вы можете развертывать различное оборудование — в зависимости от того, какие ресурсы требуются тому или иному приложению: процессор, память или хранилище. Разнообразие оборудования повышает сложность развертывания сервера и конфигурации программного обеспечения системы.

Организации может быть выгодно содержать собственные центры обработки данных, когда управление такими центрами является профильным направлением ее деятельности или когда обычные поставщики облачных услуг не в состоянии решить ее задачи (например, в части соблюдения требований регуляторов).

При развертывании стандартизированного оборудования может оказаться полезным развернуть дополнительные приложения в той же системе. Например, имеет смысл разместить веб-сервер и сервер баз данных (двухуровневая архитектура системы) в одной системе, потому что это минимизирует сетевые задержки при передаче данных между этими сервисами. С другой стороны, совместная работа этих сервисов может привести к конкуренции за ресурсы процессора, памяти или хранилища, что, в свою очередь, может усложнить принятие решений о вертикальном или горизонтальном масштабировании. Обратите внимание на узкие места сервисов с ограниченными ресурсами и недогруженные сервисы и подумайте, как сбалансировать систему, чтобы она лучше соответствовала вашим ожиданиям.

Облачные вычисления

Поставщики услуг используют различные термины для описания своих предложений. Иногда это может привести к путанице, особенно когда у одного термина появляется множество значений[1].

Взгляните на рис. 2.1. Эта таблица с терминами показывает, как различные понятия перекрывают друг друга. В верхней части таблицы располагаются «функции как сервис» (FaaS, Function-as-a-Service). Здесь у вас будет наименьший контроль над работающей вычислительной инфраструктурой. В нижней части таблицы находятся виртуальные машины. Здесь вы получаете наибольший контроль и гибкость при использовании аппаратного обеспечения, но при этом наибольший объем работ по обслуживанию.

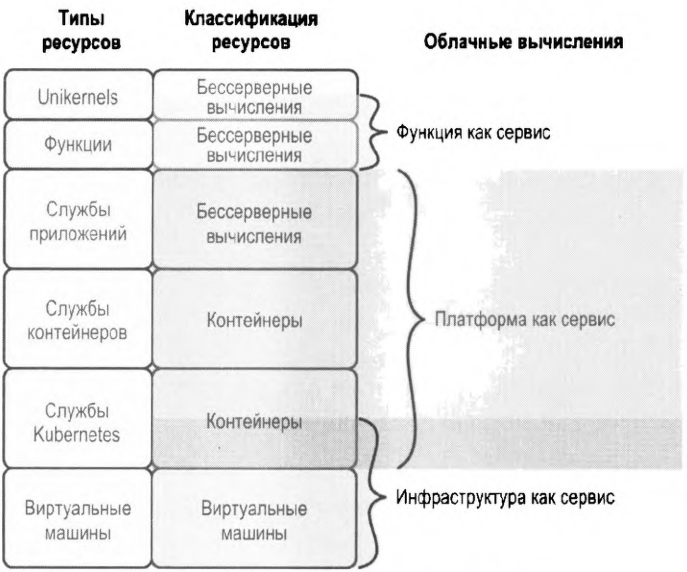

Рис. 2.1. Облачные вычислительные среды

Технологии облачных вычислений

Чтобы обозначить общие ориентиры, давайте рассмотрим такие технологии, как бессерверные вычисления, контейнеризация и виртуальные машины, доступные в этих вычислительных средах.

Бессерверные вычисления

Бессерверные архитектуры охватывают unikernels (образы виртуальных машин, содержащие единственное приложение), функции и службы приложений (а у некоторых поставщиков и контейнеры).

---

[1] В качестве примера можно привести запись в блоге Джулии Эванс (Julia Evans), где она замечательно объясняет разницу между предложениями IAM от Google Cloud и Amazon Web Services (**https://oreil.ly/bCBhT**).

Unikernels

Unikernels — это легкие, неизменяемые образы операционной системы (ОС), скомпилированные для запуска одного процесса. Я включила их сюда из-за специфических свойств и размера.

MirageOS — одна из самых долговечных операционных систем на основе библиотек, позволяющая создавать unikernels. Для практического изучения unikernels можете обратиться к руководству MirageOS tutorial (**https://oreil.ly/fwZex**).

Функции

Функции — это небольшие блоки кода, предпочтительно узкого назначения. Вы платите хостинговому сервису за обслуживание необходимой физической инфраструктуры «по запросу». Эта модель известна как FaaS (функция как сервис). Платформа предоставляет вам на определенное время ресурсы для запуска функции, или же вы сами можете выбирать это время. Предоставляемое время выполнения зависит от языков (например, Java или Go), определенных версий языка или конкретных встроенных библиотек. Ниже перечислены популярные поставщики FaaS:

- AWS Lambda;
- Azure Functions;
- Google Cloud Functions.

Вы также можете развернуть фреймворки для предоставления вашей организации функций как сервисов с помощью следующих платформ.

*Fn (**https://oreil.ly/nZxd5**)*

Легкая FaaS-платформа на базе Docker (Docker — программная платформа для разработки, доставки и запуска контейнерных приложений)

*OpenFaaS (**https://oreil.ly/UTsko**)*

Фреймворк для развертывания функций на Docker или Kubernetes (Kubernetes — программная платформа для автоматического управления контейнеризованными приложениями).

*OpenWhisk (**https://oreil.ly/KHVsw**)*

Фреймворк для выполнения функций в ответ на события. Поддерживает множество исполняющих сред для поддерживаемых языков и пользовательские исполняющие среды, поддерживаемые с помощью Docker.

Каждая платформа имеет сильные и слабые стороны в зависимости от сценария использования. В некоторых случаях выбор реализации может привязать ваше приложение к конкретной платформе FaaS. Это не означает, что вы не можете перенести свое приложение, но затраты на дополнительную работу по поддержанию возможностей приложения могут оказаться слишком большими.

Примерами дополнительных возможностей FaaS являются высокая доступность, балансировка нагрузки на конечные точки, время обработки запросов, параллель-

ные запросы и управление трафиком. Иногда можно выбирать количество процессоров и объем памяти. Ключевой особенностью платформы FaaS является то, что экземпляры функций являются активными лишь тогда, когда это необходимо. Организация среды для прототипирования нового приложения и для тестовых сред может стоить недорого в зависимости от конфигурации хранилища и сети.

Вместо определенных вычислительных ресурсов поставщики услуг взимают плату за вызов функции, использование сети и время выполнения функции. Триггеры определяют, как вызывается функция. Поддерживаются следующие распространенные триггеры:

- по расписанию;
- HTTP-запросы;
- триггеры на основе событий.

Службы приложений

Службы приложений обычно больше, чем функции. Платформы охватывают физическую инфраструктуру, включая масштабирование приложения с использованием дополнительного оборудования по мере необходимости, среды выполнения приложения и зависимости, необходимые для предлагаемых языков.

Наиболее известные сервисы хостинга приложений:

- DigitalOcean App Platform (**https://oreil.ly/uqDV3**);
- Google App Engine (**https://oreil.ly/tMR6F**).

Хотя у вас будет больше свободы при выборе конфигурации по сравнению с FaaS, вам будут предоставлены многие аналогичные возможности. После этого можете сосредоточиться на создании и развертывании приложения, а также мониторинге необходимых системных сред.

Контейнеры

Концептуально, *контейнер* — это изолированный процесс с переносимой средой выполнения. Говоря о контейнерах, часто полезнее рассматривать образ контейнера и среду запуска контейнера как отдельные понятия.

Образ контейнера — это неизменяемое упакованное приложение вместе со всеми зависимостями, необходимыми для правильной работы, включая системные библиотеки, утилиты и настройки конфигурации. Образы состоят из слоев, добавляемых поверх базового или родительского образа.

Среда запуска контейнера отвечает за настройку и запуск контейнера. Низкоуровневые среды запуска контейнеров имеют ограниченный набор возможностей (например, выделение ресурсов и настройка процессов) и выполняют, как правило, ключевые шаги по настройке. Высокоуровневые среды запуска контейнеров поддерживают функции более высокого уровня (например, управление образами и сетевое взаимодействие), и именно с ними вы обычно имеете дело.

Вот несколько примеров часто используемых сред запуска контейнеров:

containerd

Низкоуровневая среда выполнения контейнеров с открытым исходным кодом, поддерживаемая Linux и Windows

Docker

Высокоуровневая среда выполнения контейнеров

runC

Стандартная низкоуровневая среда выполнения контейнеров, написанная на языке Go Windows Containers

Оркестратор контейнера управляет кластерами контейнеров, заботясь о масштабировании, сетевом взаимодействии и безопасности.

Приведем несколько примеров оркестраторов:

- Kubernetes;
- Google Cloud Run;
- Amazon App Runner;
- Amazon Elastic Container Service (ECS);
- Amazon Elastic Kubernetes Service (EKS);
- Azure Container Instances (ACI);
- Azure Kubernetes Service (AKS);
- Google Kubernetes Engine (GKE);
- Red Hat OpenShift.

Прочитайте статью Джулии Эванс «What Even Is a Container: namespaces and cgroups» (**https://oreil.ly/1XDJQ**), которая представляет собой наглядное руководство по контейнерам.

Виртуальные машины

Независимо от того, размещаются ли виртуальные машины (virtual machines,VM) на сервере или ваша команда сама управляет ими, виртуализация сервера — это процесс создания среды, в которой несколько экземпляров операционной системы (ОС) могут работать на одном физическом сервере. С помощью технологии VM пользователи могут устанавливать и одновременно запускать совершенно разные ОС на одном компьютере.

С практической точки зрения, виртуализация дает вам возможность задействовать больше аппаратных ресурсов, которые простаивали бы на выделенных серверах.

Packer (**https://oreil.ly/lPzf8**) — это программное обеспечение с открытым исходным кодом от компании HashiCorp для создания идентичных образов машин из одного файла конфигурации для различных платформ, включая Amazon EC2, Microsoft Azure Virtual Machine, Docker и VirtualBox. Оно поможет вам создавать

одинаковые образы сходным образом у провайдеров, а также для локального применения.

Ключом к виртуализации серверов является гипервизор, который координирует низкоуровневое взаимодействие между виртуальными машинами и аппаратным обеспечением хоста.

Гипервизор — это специализированное оборудование, микропрограмма или программное обеспечение. На эту тему существует множество специализированных и всеобъемлющих ресурсов, в зависимости от вашей сферы деятельности.

Например, ознакомьтесь с записью в блоге Брендана Грегга (Brendan Gregg) «AWS EC2 Virtualization 2017: Introducing Nitro» (**https://oreil.ly/8Lisl**) для глубокого погружения в Nitro, технологию виртуализации, используемую в Amazon Elastic Compute Cloud (Amazon EC2).

В конфигурационном файле указываются виртуализированные аппаратные ресурсы, включая память, процессор и хранилище, которые выделяются виртуальной машине. Ниже приведены популярные приложения VM для настольных компьютеров:

- Microsoft Hyper-V (Windows) (**https://oreil.ly/BIFBK**);
- Oracle VM VirtualBox (Windows, Mac и Linux) (**https://oreil.ly/0LAc3**);
- Parallels Desktop (Mac) (**https://oreil.ly/TlcYV**);
- VMware Fusion and Workstation (Windows, Mac и Linux) (**https://oreil.ly/WfLYx**).

Благодаря виртуализации вам не придется покупать специализированное оборудование для каждого сервиса, который вы запускаете. Однако вам все равно придется управлять программным обеспечением на виртуальных машинах. Ниже приведены примеры виртуальных машин, расположенные на серверах:

- Amazon Elastic Compute Cloud (EC2);
- Azure Virtual Machines;
- DigitalOcean Droplets (**https://oreil.ly/r8mgZ**);
- Google Compute Engine.

Рекомендации по выбору вычислительных ресурсов

Чтобы оценить отдачу и выбрать подходящие вычислительные ресурсы для ваших нужд, обратите внимание на следующие аспекты.

Имеется ли у вас в наличии физическое оборудование, которое вы намереваетесь использовать?

Вы можете выбрать параметры этого специализированного оборудования в соответствии с потребностями ваших внутренних и внешних пользователей. Вы можете выполнить следующие действия:

- выделить сервер непосредственно для той системы, которой вам нужно управлять;
- управлять виртуальными машинами (и физическим оборудованием);
- управлять контейнерами (и физическим оборудованием).

Нужен ли вам определенный язык или фреймворк для работы вашего приложения (которыми вы не хотите управлять)?

Можете ли вы реализовать их с помощью небольших блоков бизнес-кода? Используйте «функции как сервисы». В противном случае используйте службу приложений, если ваш язык поддерживается, или службу контейнеров, если не поддерживается, и настройте свой образ сборки.

Вам нужна конкретная ОС?

Можете использовать платформу как услугу (PaaS) или инфраструктуру как услугу (IaaS).

Вам нужно быстро реагировать на растущие потребности?

Выбирайте FaaS или PaaS.

Насколько географически распределенным должно быть ваше приложение?

Изучите предложения конкретных облачных провайдеров, чтобы определить наилучший вариант распределения по регионам.

Этические соображения при выборе услуг

Вы также должны убедиться, что потенциальный поставщик услуг проявляет ответственность в социальном и этическом отношении (например, в отношении изменения климата, экономического неравенства или этичного выбора поставщиков). В современном мире социальных сетей негативные внешние факторы в вашей цепочке поставок оказываются на виду. Многим компаниям пришлось остановить производство и сменить поставщиков из-за того, что общественность узнала о проблемах этического характера или в трудовой сфере.

Ваша организация должна тщательно отбирать поставщиков, а работники, ответственные за их выбор, должны соблюдать профессиональную этику при принятии решений. Этические соображения являются теперь неотъемлемой частью финансовых вопросов организации.

Виртуализация серверов, контейнеризация и бессерверные вычисления позволяют запускать программное обеспечение в изолированной среде с согласованными зависимостями, библиотеками, двоичными файлами и конфигурациями, которые можно надежно и последовательно повторять. Существуют различия в уровне изоляции, времени запуска и межплатформенной переносимости.

Изоляция

В бессерверных вычислениях изоляция непрозрачна. Ваша функция или размещенное на сервере приложение может работать в контейнерах или на виртуальной машине. При использовании контейнеров на одном ядре Linux размещается несколько контейнерных приложений. При виртуализации на одном гипервизоре размещается несколько виртуализированных ОС, каждая из которых имеет свое независимое ядро.

В идеале ваша функция должна быть небольшой, иметь узкое назначение и использовать управляемую среду выполнения платформы. Контейнер инкапсулирует приложение объемом несколько мегабайт (Мбайт). Виртуальная машина будет больше, потому что она должна включать полный работоспособный образ ОС. Его размер может составлять от сотен мегабайт до нескольких гигабайт.

Это означает, что изоляция контейнеров не является полной, и если существует уязвимость в среде выполнения контейнера или ядре хоста, злоумышленник может выйти из контейнера и получить доступ к хост-машине.

Время запуска

VM необходимо загружать, как и любой другой компьютер, что может занять несколько минут. Контейнеры не инициализируют оборудование при запуске и запускаются за считанные секунды. Их можно так же быстро свернуть, когда они больше не нужны.

Межплатформенная переносимость

Контейнеры больше подходят для запуска множества экземпляров приложений на одной серверной ОС, в то время как виртуальные машины лучше подходят для запуска различных ОС на общем серверном оборудовании.

В конечном итоге, когда вы создаете архитектуру своих систем и рассматриваете возможность применения бессерверных решений, необходимо учитывать три основных аспекта.

Разработка новых приложений

Бессерверные вычисления способствуют быстрой разработке и масштабированию, обеспечивая очевидные преимущества для бизнеса. Вы можете экспериментировать и пробовать различные способы решения какой-либо задачи, не вкладывая большие средства в вычислительные ресурсы, прежде чем узнаете реальные потребности разрабатываемой системы.

Замена задач администратора

Вместо выделенного сервера для выполнения административных задач вы можете определить функции, которые будут иметь лишь необходимый уровень привилегий и заданное время выполнения. Вы экономите деньги и повышаете безопасность, устраняя административный сервер, на котором возможен доступ с привилегиями суперпользователя к эфемерным экземплярам. Теперь они будут запускаться только с теми разрешениями, которые требуются для выполнения их задач (например, почтовая рассылка, выполнение комплаенс-контроля инфраструктуры).

Изменения в вопросах эксплуатации и безопасности

Если ваша команда готова решать эксплуатационные проблемы и проблемы безопасности на ранних этапах процесса принятия решений, подход devops (предусматривающий сотрудничество и взаимодействие между различными группами, ответственными за систему) позволит принимать сложные решения, касающиеся шаблонов и реализации бессерверной архитектуры, и избежать

дорогостоящих ошибок (например, траты всего бюджета или раскрытия данных клиента вследствие неверных настроек)[2].

Может оказаться, что для начала стоит выбрать те компоненты, которые позволят вам быстрее приступить к работе и сосредоточиться на основных направлениях деятельности вашей организации. Со временем вы поймете, нужны ли доработки и улучшения при использовании ресурсов.

Если вы учтете эти соображения, то сможете выбрать те вычислительные ресурсы, которые адаптированы к потребностям и технологиям вашей организации или команды.

Заключение

Вычислительные ресурсы включают в себя все компоненты, связанные с вычислениями: вычислительную мощность, память, хранилище и сеть, а также операционную среду для использования этих ресурсов. Каждое приложение, которым вы управляете, должно быть развернуто с учетом предъявляемых к нему требований и ограничений. Например, приложения могут быть требовательными к мощности процессора, объему памяти или хранилища. Точно так же приложениям могут быть присущи иные особенности, такие как возможность масштабирования в соответствии с меняющимися потребностями, необходимость взаимодействия с определенным оборудованием или использования определенных языков, библиотек и операционных систем.

Ваша вычислительная среда будет состоять из множества вычислительных систем, входящих в различные архитектурные шаблоны. Делайте выбор, исходя из своих задач и учитывая требования к изоляции, времени запуска и межплатформенной переносимости.

[2] Вы можете заметить, что в этой книге встречаются написания «DevOps» и «devops». Это сделано намеренно. Когда речь идет о названиях продуктов, я употребляю термин DevOps. Применительно к практике использую первоначальный хештег devops. В основе практик devops лежат усилия по объединению людей и оказанию им помощи в совместной работе, переход от формата «мы или они» к диалогу, который дает возможность бизнесу внедрять подходы, нацеленные на устойчивое развитие. Организации могут продавать инструменты DevOps, но купить devops невозможно.

ГЛАВА 3

Хранилище

Подумайте о любом веб-сайте, на котором представлены современные товары. Это виртуальный парадный вход компании и единая общая управляемая система. Мне приходилось работать с множеством веб-сайтов, выполняя такие обязанности, как проектирование систем, поддержка веб-серверов и серверов баз данных, а также резервное копирование. Я сталкивалась со всеми типами сгенерированных файлов, включая фотографии продуктов и видеоотзывы, с операциями зарегистрированных пользователей, такими как поиск и покупки, а также обновлением различных внутренних систем инвентаризации.

При работе с хранилищем артефактов приходится принимать множество скрытых решений. Представьте, что вы управляете веб-сайтом компании. Когда кто-то ищет продукт на сайте и попадает на соответствующую страницу, это действие приводит к созданию множества записей журнала, которые должны куда-то стекаться. Каждый раз, когда кто-то покупает продукт, информация о заказе, детали отгрузки и данные о наличии продукта должны обновляться, потому что ваша организация должна продавать лишь то, что имеется в наличии. Создание системы для такого веб-сайта требует разработки проекта вычислительной среды и хранилища артефактов с подходящими размерами и возможностями масштабирования, что позволит принимать решения в бизнесе.

При управлении системами, будь то веб-сайты или что-то другое, вы создаете стратегии и принимаете решения, связанные с хранилищами, потому что вам нужно поразному оптимизировать массивы данных. Вы не будете тратить деньги на ненужное хранилище или держать редко используемые данные в более дорогом хранилище с низким уровнем задержек. Что же касается часто запрашиваемых данных — вы наверняка хотите, чтобы пользователи получали ответы как можно быстрее. Это может потребовать даже кеширования данных в памяти, что дороже, но интересы клиентов превыше всего.

Существует множество вариантов выбора, и при разработке системы необходимо учитывать особенности конкретного массива данных. В этой главе я сосредоточусь на описании стратегий и технологий хранения данных, а также соответствующих методов работы с ними.

Почему стоит заботиться о хранилище?

Хранилище данных является неотъемлемой частью систем, которыми вы управляете. Поэтому вам необходимо держать под контролем любые массивы данных, связанные с вашими системами, минимизируя возможные риски для бизнеса.

Решения, принимаемые вами в отношении хранилища данных, оказывают долгосрочное влияние на долговечность, межплатформенную переносимость, гибкость и согласованность ваших систем. К сожалению, бытует мнение, что хранение данных превратилось в товарный бизнес, в издержки, которые следует сводить к минимуму. Но на практике следует использовать более целостный подход и оптимизировать хранилище массивов данных с учетом их характеристик. Опять же, вспомните пример с веб-сайтом товара. Картинки, показываемые клиентам, должны загружаться быстро. История операций должна сохраняться в точном виде, но быстрый доступ к ней не требуется.

Наглядные примеры ценности данных

Часто данные оказываются гораздо более ценными, чем носитель, на котором они хранятся, ведь заменить диск гораздо проще, чем информацию. Есть много примеров, как организации теряют бесценные данные из-за небрежности, недальновидности или аварии.

- В 1980-х годах НАСА перезаписало оригинальные магнитные ленты с данными о лунных посадках «Аполлона» (**https://oreil.ly/3YvhP**).
- До 1978 года на Би-би-си видеокассеты зачастую использовались повторно, что привело к удалению тысяч часов вещательных программ, в частности были утеряны ранние эпизоды сериала «Доктор Кто» (**https://oreil.ly/pzefl**).
- В 2021 году пожар в дата-центре OVHcloud (**https://oreil.ly/dUdSk**) в Страсбурге, Франция, уничтожил один из четырех объектов, в результате чего пострадали 3,6 млн вебсайтов (**https://oreil.ly/TkuKG**).

Известно множество других случаев потери данных. Возможно, вы сами с ними сталкивались.

С другой стороны, бессрочное хранение данных связано с постоянными издержками. Например, многие люди обеспокоены тем, как организации хранят и агрегируют свои данные. В результате клиенты теряют доверие к организациям, которые небрежно относятся к хранилищам данных. Кроме того, законодательные рамки, например Европейский общий регламент по защите данных (GDPR), вынуждают компании предоставлять людям контроль над тем, как хранятся их данные. Таким образом, если вы допустите ошибку и будете хранить данные, нарушая принятые правила, то помимо расходов на хранение данных столкнетесь со штрафами.

Тщательно продумайте, какие данные вам необходимо хранить и как вы их храните. После этого регулярно проводите аудит массивов данных, убеждаясь, что ваши ожидания соответствуют действительности, и устраняйте все выявленные проблемы.

Некоторые собираемые данные могут включать идентифицирующую личность информацию (*англ.* personally identifiable information, PII), персональные данные, информацию о платежных картах и учетные данные. Хранение таких данных четко регламентируется законодательством. PII в основном используется в США, а персональные данные связаны с законом ЕС о конфиденциальности данных (Общий регламент ЕС по защите персональных данных, GDPR).

Ваши пользователи и их местоположение определяют требования, которым вы должны следовать, включая необходимость хранения данных в подходящих местах.

Национальный институт стандартов и технологий США (NIST) определяет PII (**https://oreil.ly/cWklx**) как информацию, позволяющую идентифицировать личность. Примером PII является индивидуальный номер карточки социального страхования.

Европейская комиссия определяет персональные данные (**https://oreil.ly/ly7lP**) как любую информацию, которая может прямо или косвенно идентифицировать человека. Примером персональных данных является домашний адрес.

Информация о платежных картах — это данные, содержащиеся на банковских картах человека, включая кредитные и дебетовые карты.

Учетные данные пользователя позволяют сайту проверить, что человек является тем, за кого себя выдает.

Проверьте, какие данные вы храните. Это поможет вам понять, какая ответственность возлагается на вас согласно законам и правилам, связанным с конфиденциальностью и хранением данных. При работе с PII, персональными данными, информацией о платежных картах и учетными данными убедитесь, что вы изучили требования соответствующих законов.

Давайте рассмотрим основные характеристики хранилища, чтобы вы знали, как оценить имеющиеся возможности при проектировании систем и возможности по усовершенствованию существующего хранилища артефактов.

Основные характеристики

В облачных и локальных вычислительных средах применяются одинаковые носители данных (жесткие диски, твердотельные накопители, флеш-память, магнитные ленты и оптические носители). У каждого из этих типов свои характеристики производительности, надежности и стоимости.

Если вы управляете физическим оборудованием, вы должны глубже понимать принципы работы хранилища, уметь разбивать его на разделы и подготавливать для работы с любой ОС, используя соответствующие драйверы устройств. С другой стороны, если вы перешли на облачные технологии, то функции низкого уровня находятся в ведении провайдера.

Одним из преимуществ облачных вычислений является возможность оптимизации расходов на хранение данных путем тестирования вариантов, которые предлагает поставщик облачных решений, поскольку именно он управляет физическими системами хранения данных.

Вы должны разбираться в принципах хранения данных независимо от того, в какой вычислительной среде вы работаете (локальной или облачной). Приведем важнейшие характеристики хранилищ данных.

Емкость

Общее дисковое пространство устройства хранения.

Операции ввода/вывода в секунду (Input/output operations per second, IOPS)

Определяет число возможных операций чтения и записи. Устройства хранения данных могут предназначаться для операций чтения или записи, для последовательного или произвольного доступа к данным.

Размер (блока) ввода/вывода

Объем данных, обрабатываемых для каждой транзакции ввода-вывода, может варьироваться в зависимости от ОС и приложения.

Пропускная способность.

Показывает скорость обмена данными между приложением и файловой системой за определенный промежуток времени.

Задержка

Характеризует время отклика, т. е. время, в течение которого приложение должно ждать завершения запроса.

Приложения используют разные модели при обращении к данным. Поэтому, приступая к созданию систем и выбирая ресурсы, вы будете обращать внимание на эти характеристики и сужать область поиска, исходя из потребностей. Отношение количества операций ввода/вывода (IOPS) к пропускной способности позволяет оценить быстродействие хранилища: IOPS = пропускная способность / размер ввода-вывода.

Предположим, вам нужно определить, какой том хранилища Amazon EBS подключить к экземпляру Amazon EC2. Вы знаете, сколько запросов в секунду требуется совершать вашему приложению, каков размер хранимых запросов и как это соотносится с пропускной способностью лежащей в основе файловой системы. Далее следует рассчитать минимальное количество операций ввода-вывода в секунду и оценить, какая комбинация предлагаемых решений для хранения данных будет наиболее подходящей.

В табл. 3.1 я привела некоторые характеристики твердотельных накопителей EBS, представленные в документации по Amazon EBS (**https://oreil.ly/6vVPR**) за август 2022 года. Первые два варианта похожи и имеют одинаковую стоимость, но третий вариант (EBS General Purpose SSD) существенно отличается.

Таблица 3.1. *Типы томов Amazon EBS в виде таблицы*[1]

Тип тома	EBS provisioned IOPS SSD (io2 Block Express)	EBS provisioned IOPS SSD (io2)	EBS general purpose SSD (gp3)
Надежность	99,999%	99,999%	99,8–99,9%
Размер тома	4 Гбайт — 64 Тбайт	4 Гбайт — 16 Тбайт	1 Гбайт — 16 Тбайт
Макс. IOPS/том	256 000	64 000	16 000
Макс. пропускная способность/том	4000 Мб/с	1000 Мб/с	1000 Мб/с
Задержка	Доли миллисекунды	Несколько миллисекунд	Несколько миллисекунд

[1] «Возможности Amazon EBS — Amazon Web Services» (**https://aws.amazon.com/ebs/features**), Amazon Web Services, Inc., по состоянию на 15 августа, 2022 г.

Предположим, вы определили, что вашему приложению требуется менее 16 000 IOPS, и не требуются задержки в доли секунды и гарантии высокой надежности. В этом случае выгодным решением станет сокращение затрат на хранение и ежемесячное получение бесплатных IOPS.

Освещение вопросов оптимизации производительности для конкретных рабочих нагрузок и приложений выходит за рамки этой книги. Однако, если это является для вас проблемной областью, изучите конкретные рекомендации для вашего приложения и используйте средства оценки производительности для определения и улучшения характеристик. Например, в Linux можете использовать утилиту `iostat`.

Разновидности систем хранения данных

Существует несколько разновидностей хранилищ — блочное, файловое, объектное или база данных. Давайте рассмотрим каждое из них, чтобы лучше понять основные уровни абстракции представления данных и их влияние на выбор системы.

Блочная система хранения данных

Для вычислительных сред блочное хранилище — это самый прямой способ взаимодействия с физическими устройствами хранения данных. Остальные варианты расположены на более высоком уровне абстракции. В блочных системах при записи на носитель данные разбиваются на сегменты одинакового размера. Система использует очереди чтения и записи для оптимизации доступа к носителю. Виртуализированное блочное хранилище использует ту же стратегию, но прозрачно добавляет уровень сети и хранит отдельные сегменты данных на разных дисках, серверах или даже в разных центрах обработки данных.

Технология, известная как избыточный массив независимых дисков (redundant array of independent disks, RAID), позволяет объединять несколько дисков в единое логическое блочное устройство, вертикально масштабируя емкость и производительность и добавляя уровень защиты данных. Сети хранения данных расширяют эту идею, распространяя горизонтальное масштабирование на несколько серверов.

Блочное хранилище идеально подходит, когда необходимо взаимодействовать с неотформатированными томами хранения данных, будь то загрузочный диск компьютера, логические тома виртуальных машин и контейнерные образы или диски с данными для баз данных или файловых хранилищ. Кроме того, у блочного хранилища обычно самая низкая задержка.

Файловая система хранения данных

Файловое хранилище представляет собой обычный интерфейс файловой системы, с иерархически вложенными папками, содержащими файлы, у каждого из которых есть такие атрибуты, как имя, владелец, разрешения и даты обращения.

Файловое хранилище может быть локальным, доступ к которому осуществляется через вашу ОС, или сетевым. Примерами сетевых файловых хранилищ являются

общие ресурсы Samba, смонтированные ресурсы NFS или облачные сервисы, такие как Dropbox, Google Drive, iCloud Drive или Microsoft OneDrive. Эти сетевые службы хранения данных предоставляют приложениям и ОС возможность взаимодействовать с сетевым хранилищем так же, как и с хранилищем на непосредственно подключенных дисках.

Хранилище, ориентированное на работу с файлами, — это обычный подход при работе с «настольными компьютерами», будь то совместный доступ к офисному файловому серверу или подписка на облачный сервис. Этот подход также актуален, когда вам нужно предоставить кластеру серверов общий доступ к конфигурационным файлам, файлам данных приложений или программному обеспечению. Примерами могут служить хранилища медиафайлов или домашние каталоги пользователей.

При работе с большим количеством файлов файловые хранилища могут столкнуться с проблемами масштабирования. Даже если внутренние сетевые службы масштабируются для поддержки теоретически неограниченной емкости, пользовательское программное обеспечение, взаимодействующее с хранилищем, может стать узким местом. Обычные файловые браузеры могут дать сбой при работе с многоуровневыми деревьями каталогов, содержащими тысячи или даже миллионы файлов и вложенных папок. Организация данных помогает, но у людей бывают разные представления о том, как лучше управлять содержимым больших иерархических файловых структур.

Объектная система хранения данных

В отличие от иерархического файлового хранилища, объектное хранилище работает с неструктурированными данными. Каждый фрагмент данных и связанные с ним метаданные хранятся с использованием уникального идентификатора, к которому пользователи могут быстро получить доступ. Объектное хранилище выгодно, если требуется хранить множество статичных элементов, которые не нужно группировать каким-либо определенным образом. Поэтому используйте объектные хранилища для обычных файлов, таких как текст, изображения, аудио- и видеофайлы и любые другие данные.

Обратите внимание, что в объектном хранилище объекты не сгруппированы в иерархическое дерево, как в традиционной файловой системе. Вместо этого имеется список объектов с непоследовательными идентификаторами, снабженных полями метаданных. Вы можете использовать фреймворк, который рассматривает объекты как файлы в упорядоченном дереве файлов. Тем не менее взаимодействие с метаданными объекта предоставляет множество других способов обращения с данными, например можно отбирать фотографии, сделанные определенной моделью фотоаппарата, в определенный день или в определенном месте, а также использовать соответствующие описательные теги. Объектные хранилища преимущественно используются как масштабируемые и гибкие хранилища для современных приложений, приложений, работающих с большими данными, резервного копирования и хранения мультимедийных данных.

Хранилище баз данных

Реляционные базы данных — это системы, которые часто используют диалект SQL для доступа к данным, хранящимся в связанных друг с другом таблицах, разбитых на строки и столбцы. К базам данных предъявляются такие требования, как атомарность, согласованность, изоляция и устойчивость (ACID, от *англ.* atomicity, consistency, isolation, durability). Они гарантируют целостность данных. Представьте себе банковские операции или операции с кредитными картами, когда необходимо перевести деньги с одного счета на другой. Если система не сможет завершить транзакцию, она должна отменить ее без изменения баланса средств на обоих счетах.

Система управления базами данных (СУБД), гарантирующая соответствие требованиям ACID, предоставляет надежное хранилище данных, которое может восстанавливаться после любых сбоев либо путем фиксации каждой операции, изменяющей состояние, либо путем отката к самому последнему состоянию перед попыткой неудачного обновления.

Масштабирование реляционной базы данных затрудняет выполнение гарантий в части требований ACID. Чтобы справиться с нагрузкой, можно выполнять вертикальное масштабирование, добавляя ресурсы хранения и вычислительные ресурсы. При этом можно установить столько оперативной памяти, процессоров и дисков, сколько допускает единичный сервер. В перспективе вам придется выполнять горизонтальное масштабирование и добавлять больше серверов, в результате чего ваше приложение станет распределенным. Когда у вас появится распределенное приложение, придется идти на неизбежные компромиссы.

Теорема CAP объясняет возможные компромиссные решения. Чтобы система продолжала функционировать в случае нарушения связности сети (partition tolerance, устойчивость к разделению), она может либо гарантировать, что любой запрос завершается корректным откликом, но не обязательно содержит самую последнюю запись (availability, доступность), либо гарантировать, что чтение получает самую последнюю запись или ошибку (consistency, согласованность). [Акроним CAP в наименовании теоремы сформирован из первых букв английских наименований этих трёх свойств.] Таким образом, система не может гарантировать доступность и согласованность одновременно.

Теорема PACELC расширяет теорему CAP для распределенных систем и гласит, что даже для систем, работающих оптимально (без разделения сети), необходимо выбирать между низкой задержкой и согласованностью:

- если ваше распределенное приложение чувствительно к задержкам (т. е. приоритетом является скорость ответов, а не точность), вы предпочтете, чтобы ответ на запрос возвращался до подтверждения того, что он является последней версией. Это моделирует конечную согласованность в системе;
- если ваше распределенное приложение обязательно должно быть согласованным (т. е. в приоритете точность данных), вы предпочтете, чтобы ответ возвращался после подтверждения того, что он является последней версией. Это при-

ведет к увеличению задержки запросов, которая будет зависеть от конкуренции за ресурсы среды, числа параллельных запросов или общего количества запросов на хранение, обрабатываемых в данный момент времени.

Другими словами, ваша система управления базой данных гарантирует, что информация всегда будет согласована (например, Apache HBase, **https://oreil.ly/v9yeR**), но, возможно, пользователям придется подождать, чтобы получить ответ. Либо ваша система управления базами данных обеспечивает быстроту реакции, жертвуя согласованностью (например, Apache Cassandra, **https://oreil.ly/6X6As**).

Для многих рабочих нагрузок быстрота реакции может быть не менее важна, чем согласованность. Представьте себе сайт социальной сети, где по умолчанию отображаются сообщения от ваших друзей, или поисковую систему, которая выдает список результатов поиска по запросу. Пользователю сайта не обязательно видеть каждое сообщение, когда он заходит на веб-страницу, но она не должна быть пустой. Если система пытается вывести релевантные результаты, такое поведение приемлемо.

Базы данных NoSQL (*англ.* not only SQL — не только SQL) — это распределенные системы, оптимизированные для обеспечения доступности с конечной согласованностью. Вместо того чтобы использовать SQL для следования схеме при записи в базу данных, базы данных NoSQL позволяют разработчику приложения обеспечить соблюдение схемы на уровне приложения. Эта задержка решает проблему непроизводительной потери времени, необходимого для обновления схемы базы данных, за счет усложнения уровня приложения.

Основные типы NoSQL включают хранилища типа «ключ-значение», документоориентированные хранилища, графовые базы данных и базы данных с широкими столбцами.

Хранилища «ключ-значение»

Подобно ассоциативным массивам, словарям или хеш-индексированию, которые поддерживаются многими языками программирования, хранилища «ключ-значение» связывают элемент данных с ключом-идентификатором. Примерами баз данных «ключ-значение» являются Redis и Amazon DynamoDB.

Базы данных этого типа предназначены для рабочих нагрузок с простыми требованиями к хранению, получению и удалению данных, например управления сеансами.

Документоориентированные хранилища

Документоориентированные базы данных связывают ключи со структурированным форматом (JSON, XML), известным как *документ*. Отдельные документы в хранилище не обязательно должны соответствовать единой схеме. Как и в случае с объектным хранилищем, документоориентированные базы данных являются частным примером хранилища «ключ-значение».

Документоориентированные базы данных предназначены для рабочих нагрузок, которым требуется гибкая схема, например профили пользователей, управление контентом и бизнес-аналитика в режиме реального времени.

Графовые базы данных

Графовые базы данных подчеркивают фундаментальную взаимозависимость всех сущностей. Граф включает в себя сущности (т. е. человека или место) и отношения между ними, что позволяет проводить анализ, который затруднен в более ранних реляционных системах. Примерами графовых баз данных являются Neo4j и AWS Neptune.

Графовые базы данных предназначены для любых систем, где нужно выявлять закономерности в наборах данных: социальные сети, рекомендательные сервисы, обнаружение мошенничества, оценка финансовых рисков и биоинформатика.

Базы данных с широкими столбцами

Хранилища с широкими столбцами структурируют данные по столбцам, а не по строкам, чтобы оптимизировать производительность при выполнении обычных запросов. Примерами приложений баз данных с широкими столбцами являются Apache Cassandra и Apache HBase.

Базы данных с широкими столбцами предназначены для распределенных систем с большими наборами данных.

Многие программные продукты для работы с базами данных и сервисы, размещенные на серверах, поддерживают сочетание этих вариантов. Создание комплексного решения, отвечающего потребностям вашей организации, предполагает определение типа данных и метаданных, которые необходимо хранить, и подбор вариантов, отвечающих этим потребностям.

Соображения по выбору стратегии хранения данных

Теперь, когда у вас есть общее представление о доступных вариантах хранения данных, нужно решить, какой из них использовать.

Стандартный вопрос на данном этапе заключается в выборе подхода к хранению данных. Можно использовать облачное, локальное или гибридное хранилище. Далее вам нужно будет определить решения, которые подойдут для каждой ниши системы, находящейся в вашем ведении, т. к. у них могут быть различные потребности (как объяснялось в начале этой главы).

В общем случае используйте любое подходящее экономически эффективное решение. Если говорить более конкретно, вам нужно изучить свои данные и их потоки, чтобы принять решение о хранении.

Вот несколько вопросов о ваших данных, которые следует рассмотреть.

- Какими данными вы управляете?
- Как они производятся и как потребляются?
- С каким объемом данных вы имеете дело и что вы с ними делаете?

- Кому нужен доступ к данным и как он предоставляется? Являются ли пользователи внутренними или внешними по отношению к вашей организации?
- Каковы ваши требования к хранению данных?
- Существуют ли вопросы правового характера, обстоятельства, обусловленные договором или соображениями конфиденциальности? Помните о любых ограничениях в отношении той информации, которую вы должны или, наоборот, не должны хранить.
- Как часто происходит обращение к данным?
- Работают ли пользователи в основном с последними данными или анализируют архивные данные?
- Обращаются ли отдельные пользователи, как правило, к большому количеству данных за один сеанс?
- Предъявляют ли пользовательские приложения требования к минимизации задержек?

Вот пара вопросов об устройствах, которыми вы управляете.

- Должны ли ваши вычислительные устройства загружаться автономно? Если это необходимо, каждому из них потребуется загрузочный диск. Конечно, вы можете централизовать администрирование, установив сервер `netboot`, обеспечивающий работу бездисковых систем.
- Нужен ли вашим компьютерам доступ к общему хранилищу? Возможно, имеет смысл разместить файлы конфигурации и домашние каталоги пользователей на центральном файловом сервере в локальной сети. Если у вас есть коллеги, которые работают удаленно и не всегда имеют свободный доступ к серверу в офисе, то, вероятно, стоит обратиться к облачным технологиям, чтобы синхронизировать локальные файлы с облаком.

Инвестиции в собственное локальное оборудование помогут обеспечить большую емкость и высокую пропускную способность с низкой задержкой для внутренних пользователей в одном географическом регионе.

Это потребует высоких первоначальных затрат и постоянного обслуживания, в частности резервного копирования данных и мониторинга состояния оборудования. Необходимость масштабирования также может быть ограничивающим фактором. Если сервер исчерпал свою емкость, вы можете удалить или выгрузить данные или добавить дополнительные серверы, что может потребовать значительного времени на подготовку.

С другой стороны, облачные решения предполагают минимальные первоначальные затраты и не требовательны к обслуживанию. По мере роста потребностей вы можете смешивать и сочетать блочные, файловые, объектные хранилища и базы данных, что открывает практически неограниченные возможности масштабирования. Кроме того, вы можете реплицировать данные в другие регионы, чтобы уменьшить задержку для глобальных пользователей.

Эти решения требуют оплаты подписки на постоянной основе, и затраты могут неожиданно вырасти, если вы не установите ограничения на объем используемого

хранилища. В дополнение к этим ограничениям, вы можете перенести данные, к которым не так часто обращаются, на «холодные» уровни хранения, чтобы сократить расходы. В итоге вы будете наблюдать за биллинг-панелью поставщика облачных решений, а не отчетами о состоянии жестких дисков.

В некоторых организациях может быть запрещено подключать критически важную инфраструктуру к сети Интернет. Но некоторые облачные провайдеры предлагают решения, соответствующие государственным стандартам хранения конфиденциальной информации. Тем не менее для учреждений, которые не могут даже рассматривать такую возможность, решения на базе локальных систем являются единственным приемлемым вариантом.

Заранее обдумайте требования к емкости и задержке

Рассмотрим сервис потокового вещания видео, который предоставляет своим подписчикам библиотеку из тысяч фильмов и телепередач. Каждый фрагмент контента кодируется с различным качеством, от стандартного формата (SD) до формата высокой четкости (HD) (720p, 1080i, 1080p, 4K и 8K). Требования к пропускной способности и объему хранилища данных быстро возрастают по мере развития технологий. Форматы кодирования, или кодеки, уменьшают объем данных, но при этом увеличивается стоимость за счет дополнительных вычислений и снижается качество изображения.

Пользователи сервисов потокового видео используют различные устройства, от мобильных телефонов до персональных компьютеров и телевизоров с большим экраном. Скорость их подключения также варьируется. Это может быть модем для коммутируемой линии или гигабитные или более быстрые широкополосные соединения. В результате сервисам приходится хранить десятки предварительно сгенерированных файлов в разных форматах для поддержки такого количества клиентов. Нетрудно подсчитать, что для хранения одного полнометражного фильма в различных форматах с высоким качеством может потребоваться терабайт емкости, а для библиотеки из тысячи фильмов — петабайт. И это только сами данные, не считая дополнительных расходов на хранение баз данных и программного обеспечения, управляющего доступом к видеотеке. А ведь нужно еще выполнять резервное копирование!

Чтобы минимизировать задержки для клиентов их разных регионов, потоковые сервисы используют сети доставки контента (content distribution network, CDN) для репликации данных на объекты в разных точках мира. Так что дубликаты этого петабайта данных для фильмотеки могут располагаться еще в сотне других мест. Для управления этими данными необходим дополнительный уровень инфраструктуры, поддерживающий реплицированные данные в актуальном состоянии и удаляющий всю устаревшую информацию.

Подумайте о требованиях к данным вашей собственной организации. Возможно, вы используете их не так активно, а возможно, даже с большим размахом. Какими бы ни были текущие показатели, велика вероятность, что в будущем они будут только возрастать.

Выбирайте разумную длительность хранения данных

У данных есть жизненный цикл. Некоторые из них недолговечны, другие необходимо хранить долгое время. Емкость хранилища стала недорогой, но все же не бесплатной, а хранение данных связано с постоянными расходами. Программное обеспечение, которому приходится обрабатывать больше данных, либо работает медленнее, либо потребляет больше вычислительных ресурсов для поддержания быстродействия.

Рассмотрим инфраструктуру, необходимую для поддержки фитнес-устройства пользователя, которое регистрирует показатели сердечного ритма и количество шагов. Датчики регистрируют события с высокой точностью, но не сохраняют исходные данные телеметрии из-за ограничений емкости. Вместо этого устройство рассчитывает заданные показатели и отбрасывает исходные данные. Наконец, пользователь синхронизирует свое устройство и загружает сводные данные в свой аккаунт.

Пользователям, как правило, хочется наблюдать тенденции во времени, и вряд ли им потребуется узнать количество шагов, сделанных в определенный день несколько лет назад. Поэтому вполне нормально, если приложение сохраняет обобщенные данные с детализацией в день, неделю или даже месяц. Обратите внимание, что на каждом этапе сбора данных система сокращает объем входящих данных до того уровня, который требуется для следующего этапа обработки.

Аналогичный подход применяется при работе с внутренними процессами организации. Например, работая над проектом, вы часто будете вести учет в тикетах, чатах, электронных письмах и общих документах. Нужно ли после завершения проекта иметь возможность обращаться к этим архивным данным или достаточно иметь краткий отчет, который пригодится в дальнейшей работе? Некоторые организации проводят политику, запрещающую хранение таких артефактов, в то время как другие поощряют сохранение этой служебной информации. Оба подхода имеют сильные и слабые стороны.

Удаление данных при утилизации оборудования

Ваша политика хранения данных также должна учитывать утилизацию оборудования, которое больше не требуется. Перед утилизацией серверов, персональных компьютеров, мобильных устройств, переносных дисков и т. д. всегда следует исходить из того, что люди могли хранить на этом оборудовании конфиденциальную информацию. Заинтересованные лица, имеющие доступ к нужным инструментам, в некоторых случаях могут восстановить «удаленные» данные с устройств хранения.

Если это вызывает опасения, возможно, стоит принять дополнительные меры по очистке данных — например, записать на диски нули или случайные данные. Скремблирование данных может стать эффективной стратегией для предотвращения их восстановления, но этот процесс требует много времени, и даже он не является полной гарантией уничтожения информации для некоторых организаций. С устройств, использующих шифрование «всего диска», «удалить» данные очень просто — достаточно уничтожить ключ шифрования. После этого восстановить информацию не сможет никто, разве что только самые злостные нарушители.

Если сомневаетесь, стирайте и физически уничтожайте диски. Тогда вы будете точно знать, что злоумышленники не смогут восстановить данные.

Уважайте озабоченность клиентов по поводу конфиденциальности

Я не могу не отметить, как важно уважать вопросы, связанные с соблюдением конфиденциальности ваших пользователей. Обратите особое внимание на то, как вы обращаетесь с персональными данными. Часто существует договорное или юридическое требование получать такую информацию только для определенной цели и удалять ее, как только она становится ненужной.

Защитники конфиденциальности обращают внимание на проблемы, возникающие при сборе, покупке и продаже пользовательских данных, часто без полностью информированного согласия самих владельцев этих данных. В юрисдикциях, где действует закон о «праве на забвение», люди могут попросить вас удалить их данные, и вы обязаны выполнять такие просьбы как с юридической, так и с моральной точки зрения.

Защитите свои данные

Утечки данных, т. е. нарушения конфиденциальности, при которых неавторизованная сторона копирует, крадет, передает, использует или просматривает частные данные, случаются часто. К таким данным относится финансовая информация, медицинские данные о состоянии здоровья, персональные данные, коммерческие секреты и другая интеллектуальная собственность. Подобные инциденты могут приводить к значительным прямым и косвенным затратам, связанным с мероприятиями по устранению последствий и с репутационным ущербом.

Утечки данных происходят по разным причинам, как по вине инсайдеров, так и из-за внешних участников. Причиной может стать небрежность — например, размещение оборудования в неподходящем месте, слишком простой пароль или отсутствие надежного шифрования, — а также преднамеренные действия, такие как взлом, саботаж или кража. Векторы атак включают вредоносное ПО, фишинг, программы-вымогатели, социальную инженерию и кражу физических носителей.

Одним из способов обезопасить данные является использование принципа минимальных привилегий: «Каждой программе и каждому привилегированному пользователю системы должны предоставляться только те привилегии, которые являются абсолютно необходимыми для выполнения поставленной перед ними задачи». Эта философия определяла правила разработки операционных систем на протяжении десятилетий и применима до сих пор.

Подумайте о том, кому потребуется доступ к вашим данным на разных этапах. Оцените конкретные услуги, которые предоставляет ваше программное обеспечение. Нужен ли разработчикам полный доступ ко всем данным или достаточно иметь только высокоуровневый доступ к метаданным для проверки работоспособности? Например, нужен ли администраторам доступ к учетной записи электронной почты пользователя и сообщениям в Slack или достаточно просто проверить, что учетная запись активна и пользователь имеет к ней доступ?

Ваша система должна шифровать данные при передаче и хранении, чтобы ужесточить политику наименьших привилегий. Сетевые протоколы должны шифровать данные по умолчанию. Времена, когда можно было без опасения использовать открытые стандартные протоколы (т. е. HTTP и SMTP), давно миновали.

Знаю, что повторяюсь, но хочу еще раз акцентировать ваше внимание на том, как важно ответственно подходить к обращению с финансовыми, медицинскими и персональными данными. Самый простой способ защитить эти данные — не собирать их.

Утечки данных могут дорого обходиться. Разглашение персональных данных может привести к краже личной информации и другим формам мошенничества. Жертвы таких преступлений вправе ожидать мер по исправлению ситуации — кредитного мониторинга, замены кредитных карт и других форм компенсации. Раскрытие конфиденциальной информации об интеллектуальной собственности, например исходного кода или других коммерческих секретов, может подорвать позиции компании на рынке и дать неожиданное преимущество конкурентам. Например, в результате утечки данных в компании Yahoo в 2013 и 2014 годах в свободном доступе оказалась информация о трех миллиардах учетных записей пользователей. В результате, когда Verizon приобрела Yahoo в 2016 году, стоимость компании была снижена на 350 миллионов долларов — почти 10% от цены приобретения в 4,8 миллиарда долларов.

Будьте готовы к необходимости восстановления после сбоев

Исходите из предположения, что потеря данных — жизненный факт. Люди случайно удаляют файлы или форматируют не тот диск. У поставщиков услуг бывают перебои в работе из-за аварий и отказов оборудования. Центры обработки данных не застрахованы от стихийных бедствий.

Учитывайте эти факторы при оценке доступности данных. Нужно ли резервное копирование всех данных? С какой регулярностью его выполнять? Насколько актуальными должны быть данные? Если пользователь создал файл десять минут назад, а через пять минут удалил его, сможете ли вы восстановить его? А если потребуется файл, который был удален пять месяцев или пять лет назад? Приемлемо ли восстановление в течение часа? Дня? Недели?

Резервное копирование обязательно нужно проверить на практике. Необходимо смоделировать различные ситуации потери данных (и периодически повторять эти действия) и убедиться, что у вас есть опробованная, документированная процедура восстановления данных в ожидаемые сроки. Хуже всего обнаружить, что резервное копирование не работает, когда у кого-то поджимают сроки сдачи работы.

Пример: «История игрушек 2» от компании Pixar

В 1998 году один из сотрудников Pixar случайно удалил весь киноархив, когда компьютерный анимационный фильм «История игрушек 2» находился в производстве почти два года. К сожалению, резервное копирование перестало работать из-за нехватки свободного места.

К счастью, у Галин Зусман (Galyn Susman), главного технического директора, которая некоторое время работала из дома, имелась копия данных.

Команда смогла использовать ее копию, резервную копию двухмесячной давности, а также множество собранных воедино файлов из неудачных рендеров и с локальных накопителей аниматоров. Кропотливо просмотрев десятки тысяч файлов, они собрали новое дерево исходников для мультфильма. Позже Pixar решила, что «История» не удалась, намеренно удалила проект и начала все сначала.

Из этого следует извлечь несколько уроков.

- Примите меры предосторожности на случай удаления данных.
- Следите, как выполняется резервное копирование.
- Разрешайте сотрудникам работать самостоятельно.
- Иногда полезно все бросить и начать сначала.

Заключение

Данные, как правило, являются самым ценным активом вашей организации, и в здоровой организации их объем постоянно увеличивается. Традиционные технологии хранения данных с использованием внутренних ресурсов отличаются высокой производительностью и большой емкостью, но связаны с высокими постоянными затратами и необходимостью текущего обслуживания. Облачные решения характеризуются неограниченной масштабируемостью, но требуют периодической и потенциально непредсказуемой платы за использование. Выберите портфель программно-аппаратных решений, отвечающих вашим потребностям.

Дополнительные ресурсы

Ознакомьтесь с книгой «*Modern Data Protection*», W. Curtis Preston (O'Reilly), чтобы получить более подробные сведения о защите данных с помощью резервного копирования. Более подробное объяснение концепций, инструментов и параметров настройки для операционных систем и приложений Linux вы найдете в книге «*Systems Performance*», Brendan Gregg (Prentice Hall).

Если вы ориентируетесь в первую очередь на управление базами данных, ознакомьтесь с указанными ресурсами:

- «*High Performance MySQL*», Silvia Boltros и Jeremy Tinley (O'Reilly);
- «*Database Reliability Engineering*», Laine Campbell и Charity Majors (O'Reilly).

ГЛАВА 4

Сеть

Давайте завершим изучение основ систем разговором о сети. Сети — это коммуникационная основа любой системы. Они соединяют все ваши ресурсы и услуги. Проблемы с сетью приводят к сбоям в работе системы. Критическая важность сетей быстро привела к появлению отдельной профессии сетевого администратора, который отвечает за работу сетевого оборудования. Микросервисы, виртуализация и контейнеризация вызвали тектонические сдвиги в построении современных сетей и управлении ими. Увеличение числа сетевых ресурсов, взаимодействующих между собой, программно-определяемые сети и приложения, чувствительные к задержкам, — все это изменило бытовавшие ранее представления о навыках, которыми должен обладать сетевой администратор, и некоторые из его задач снова перешли в сферу ответственности системной команды.

В этой главе я рассказываю о сетевых технологиях — виртуализации сетей, программно-определяемых сетях и сетях распространения контента, чтобы вы могли сотрудничать со своими командами специалистов по сетям и сетевой безопасности и наилучшим образом налаживать взаимодействие сетевых ресурсов вашей системы.

Посмотрим на сети внимательнее

Давайте вернемся к примеру из предыдущей главы о веб-сайте, посвященном современным товарам. Веб-сайт — это виртуальный парадный вход компании и пример системы, которой вы можете управлять.

Пользователь открывает браузер на телефоне, чтобы купить товар вашей компании. Поставщик услуг беспроводной связи направляет этот запрос в CDN (сеть распространения контента), которая размещается в центре обработки данных, расположенном физически рядом. Если у CDN нет данных для ответа на запрос, он направляется дальше. На следующем этапе балансировщик нагрузки направляет запрос на физический сервер, где гипервизор определяет, на какую виртуальную машину в вашей облачной инфраструктуре следует его переадресовать. Как только трафик достигнет ядра Linux виртуальной машины, ваше приложение обработает запрос, и ответ пройдет аналогичный путь обратно к клиенту.

Здесь происходит много событий. Сколько различных сетей вы насчитали? Каждая сеть выполняет некоторую обработку данных, т. к. маршрутизатор определяет наилучший путь к следующему пункту назначения. Большое число сетевых прыжков, различные типы сетей и разная скорость передачи данных в них приводят к неста-

бильности и длительному времени отклика. Сколько типов сетевых устройств было задействовано?

Ваших пользователей, как правило, не интересуют подробности работы сети, пока трафик передается надежно. Однако когда запросы не проходят, это большая беда, и лучше подготовиться к таким ситуациям заранее. Вместо того чтобы реагировать на проблему постфактум, полезно понимать, как взаимодействуют сети в вашей системе, чтобы создавать их и управлять ими на основе этих знаний. Понимание потребностей системы позволяет сделать обоснованный выбор, как показано в примерах с кешированием данных ближе к клиенту с помощью CDN и маршрутизацией запросов соответствующему адресату с помощью балансировщиков нагрузки.

Как и при любых решениях в отношении строительных блоков ваших систем, важен контекст того, что вы создаете. Эффективное применение доступных вам ресурсов и опций позволит снизить нагрузку на специалистов команды, занятых управлением системой, повысить удовлетворенность клиентов и общую прибыль предприятия.

Основные характеристики сетей

Как и в случае с хранилищами данных, есть две основных категории сетей — проводные и беспроводные, и в рамках каждой из этих широких категорий существуют различные среды передачи (например, медный провод, волоконно-оптические кабели) и коммуникационные протоколы.

Сети характеризуются топологией, расположением элементов и потоком данных. В зависимости от среды передачи данных топология сети будет определять расположение физических кабелей, расположение различных сетевых ресурсов и отказоустойчивость. Все эти факторы влияют на затраты, связанные с сетью.

Перечислим основные характеристики сетей.

Пропускная способность

Пропускная способность канала связи, обычно описываемая как объем данных, передаваемых за фиксированное время, т. е. мегабиты в секунду (Мбит/с) или гигабиты в секунду (Гбит/с).

Задержка

Время прохождения сигнала от определенной точки до места назначения, которое зависит от физического расстояния, которое должен пройти сигнал.

Чтобы определить задержку в сети, следует учитывать общее время передачи сообщения, время обработки запроса всеми сетевыми устройствами по маршруту следования (задержка обработки), а также время нахождения в очереди запросов, подлежащих обработке (задержка нахождения в очереди).

Джиттер (от англ. jitter — дрожание)

Отклонение от средней задержки. У любого запроса будет определенная задержка, которую можно увидеть. Для расчета ожидаемой задержки используется

среднее значение некоторого количества точек данных. Джиттер характеризует отклонения от этого показателя. Для рабочих нагрузок в сетях с низкой задержкой (например, передача аудиосигналов, потоковое вещание) джиттер может быть полезен при оценке качества сети с точки зрения согласованности ее компонентов.

Доступность

Вероятность того, что сеть доступна. Различные сети способны справиться с разным количеством отказов.

Создание сети

Представьте, что вы отвечаете за развертывание системы в центре обработки данных. В системе есть шлюз, который выполняет маршрутизацию трафика к приложению, состоящему из базы данных и банка веб-серверов. Центр обработки данных обеспечивает магистральное подключение, но за все остальное отвечаете вы. Итак, какие сетевые ресурсы вам понадобятся? Перечислим те, которые прежде всего приходят на ум.

- Межсетевой экран для фильтрации входящего и исходящего трафика
- Маршрутизатор-шлюз для приема входящего трафика из общедоступного Интернета, направления его на внутренние ресурсы для обработки и передачи исходящего трафика от внутренних узлов обратно удаленным клиентам.
- Балансировщик нагрузки для распределения трафика между веб-серверами
- Системы обнаружения вторжений для защиты сети от несанкционированного доступа извне и другой подозрительной сетевой активности.
- VPN-шлюз, предоставляющий авторизованным удаленным пользователям расширенный доступ к частной сети.

Учитывая потребности вашей сети, подумайте о характере, типе и объеме трафика.

Сети часто описываются на основе их пропускной способности. Однако даже если две вычислительные среды имеют высокую пропускную способность при подключении к более скоростному Интернету, их физическое разделение может ухудшать качество взаимодействия из-за джиттера и задержек.

Эталонная модель взаимодействия открытых систем (open systems interconnection reference model, OSI) — это семиуровневая архитектура, которая наглядно демонстрирует детали реализации протоколов и интерфейсов.

Например, традиционная балансировка нагрузки называется балансировкой нагрузки на четвертом уровне (L4), поскольку она и происходит на четвертом, транспортном уровне. Этот вариант балансировки нагрузки осуществляется за счет того, что сетевое устройство или приложение распределяет запросы на основе исходных и целевых IP-адресов и портов без более глубокого изучения содержимого пакетов. Балансировка нагрузки на седьмом уровне (L7) происходит на седьмом, прикладном уровне. Сеть или приложения, использующие балансировку нагрузки приложений, распределяют запросы на основе характеристик запросов.

Хотя такое деление не совсем точно, оно содержит достаточно контекста, чтобы им пользоваться. Например, балансировку нагрузки L4 было бы точнее называть L3/L4, поскольку балансировщик нагрузки при распределении запросов использует характеристики сетевого и транспортного уровня. А балансировку нагрузки L7 стоило бы называть L5–L7, поскольку балансировщик нагрузки использует характеристики протоколов уровня сеанса, представления данных и приложения для определения лучшего места назначения для запросов.

Ранее балансировка нагрузки L7 была очень дорогой из-за вычислений при обработке запросов. Теперь, с развитием технологий, затраты на реализацию балансировки нагрузки на уровнях L4 и L7 пренебрежимо малы, однако гибкость и эффективность L7 существенно выше[1].

Таблица 1.2. поможет вам вспомнить пять уровней модели сети Интернет.

Каждый уровень взаимодействует со смежными уровнями посредством собственного объекта сообщения. Разделение на уровни разграничивает функции и зоны ответственности, что позволяет людям создавать (и изменять) различные части коммуникационных протоколов. Это дало толчок значительному числу сетевых преобразований.

Виртуализация

Создание сети сводится к двум вещам — способности отправлять и получать данные и механизму принятия решений о том, как это делать.

Раньше для реализации каждой сетевой функции нужно было приобретать отдельное устройство. Теперь вы можете развертывать виртуализированные версии этих компонентов аналогично другим ресурсам инфраструктуры. Подобно тому как поставщики услуг виртуализировали традиционные функции серверов (например, базы данных и веб-серверы), они виртуализируют и сетевые услуги с помощью анонимного сетевого оборудования. А вы можете запускать программное обеспечение, управляющее тем, как оборудование передает и принимает данные.

Однако вы не можете виртуализировать все аспекты сетевой работы. Например, для связи с удаленными узлами обязательно используются физические каналы передачи данных, такие как кабели Ethernet, трансокеанские волоконно-оптические линии, спутники или адаптеры Wi-Fi.

Эти каналы отличаются друг от друга настолько, что для операций на канальном уровне требуется различное оборудование. Но в этом и заключается преимущество разделения реализации протокола и интерфейсов связи с Интернетом. Пока ресурсы физического уровня на месте и работают, у вас есть гибкость в настройке транспортных и сетевых ресурсов по своему усмотрению.

---

[1] Узнайте больше о балансировке нагрузки на седьмом уровне из документации NGINX (**https://oreil.ly/tFfiK**).

Возможность развертывания произвольного сетевого функционала на типовом оборудовании дает нам огромную гибкость. Вам не нужно приобретать специализированное оборудование, а затем устанавливать его в стойках центра обработки данных, ведь вызов API может выполнить ту же задачу. Вместе с тем сетевые ресурсы могут масштабироваться вертикально и горизонтально вместе с вашей остальной инфраструктурой.

Программно-определяемые сети

По мере увеличения количества развернутых сетевых ресурсов ваша задача будет заключаться в управлении этими ресурсами и их защите на основе комплексного, целостного подхода. Ранние подходы к межсетевому взаимодействию использовали децентрализованную философию, когда маршрутизаторы лишь смутно представляли, как передавать трафик в конечный пункт назначения. Децентрализованная философия сделала сеть Интернет достаточно устойчивой, чтобы восстанавливать работу после стихийных бедствий, но не гарантировала стабильность сети. Более того, такой подход не учитывал эволюционирующий характер систем безопасности. Хотя первые инженеры проектировали Интернет таким образом, чтобы он мог выдержать сегментацию, они не учли угрозы вредоносного ПО, такого как червь Морриса, и широкую интеграцию компьютеров в повседневную жизнь, что сделало всех гораздо более уязвимыми перед вредоносной активностью.

Рассмотрим задачи, стоящие перед администратором сети в университете. Институт предоставляет определенные вычислительные ресурсы (например, серверы, рабочие станции и принтеры) и позволяет студентам и преподавателям использовать собственные устройства (например, ноутбуки, планшеты и телефоны). ИТ-отдел устанавливает исправления и физически защищает оборудование университета, но обеспечить соблюдение соответствующих политик безопасности на чужом оборудовании существенно сложнее. В результате рано или поздно приходится сталкиваться с проблемами, связанными с вредоносными программами, программами-вымогателями или вирусами, распространяющимися с незащищенных персональных устройств.

Программно-определяемые сети (*англ.* software-defined networking, SDN) предоставляют инструменты, помогающие управлять ресурсами и защищать их. SDN — это подход к управлению сетью, в рамках которого вся сеть рассматривается как единый программируемый компьютер. Подобно тому как обычные компьютеры используют операционную систему для координации аппаратных ресурсов от имени приложений высокого уровня, SDN представляют централизованный фрейморк для координации работы распределенной сети, активируя ресурсы по мере необходимости, автоматически адаптируясь к меняющимся условиям и позволяя вам внедрять единые политики.

Таким образом, администратор сети может запустить приложение для анализа угроз безопасности и задействовать источники данных об общих угрозах (**https://oreil.ly/uetEd**), чтобы составить список вредоносных веб-сайтов. Теперь,

когда владельцы устройств попытаются посетить вредоносный веб-сайт, они будут направлены на страницу с предупреждением, что позволит им предпринять соответствующие действия.

Характерным признаком SDN является использование плоскости управления высокого уровня для регламентирования деятельности, осуществляемой на отдельных сетевых устройствах. В то время как провайдеры оптимизируют программное обеспечение на плоскости данных или плоскости передачи для обеспечения скорости, простоты и согласованности, плоскость управления предоставляет гибкий интерфейс для определения политик и обработки исключений.

В архитектуре SDN используется централизованный программируемый контроллер, который осуществляет надзор за работой сети. Этот контроллер использует API-интерфейсы «южного моста» (*англ.* southbound) для передачи информации на нисходящие устройства, такие как маршрутизаторы и межсетевые экраны, и API-интерфейсы «северного моста» (*англ.* northbound), которые передают информацию о состоянии контроллеру. Большинство реализаций SDN используют протокол OpenFlow для управления сетевыми устройствами независимо от поставщика. Если физическое или виртуальное оборудование поддерживает программный интерфейс для определения того, как маршрутизировать или выделять трафик, вы можете управлять им с помощью контроллера SDN.

Одновременно могут быть задействованы несколько приложений контроллера SDN. Например, некоторые приложения плоскости управления сосредоточены на операциях развертывания и предоставления ресурсов, другие могут вести учет трафика для выставления счетов, а третьи — отвечать за различные аспекты сетевой безопасности.

Сегментация — еще один способ защиты сетей. Сегментирование сети позволяет оптимизировать поток трафика для правомерного использования сети и классифицировать ущерб, нанесенный в случае атаки с использованием вредоносного ПО или утечки данных. С помощью машинного обучения современные программно-определяемые сети могут автоматически учиться определять шаблоны использования и применять эту информацию для управления работой микросегментов. Тем не менее, как и в случае со всеми системами машинного обучения, их конечные результаты зависят от данных, которые использовались для обучения.

Сети доставки контента

Ключевым элементом бесперебойной работы системы является оперативное реагирование сетевых сервисов. Пользователи привыкли ожидать практически мгновенного отклика и считают, что все сломалось, если присутствуют какие-то задержки. И все же никакая вычислительная мощность не может преодолеть скорость света. Чем дальше от вас находятся пользователи, тем заметнее задержки.

Рассмотрим сайт, расположенный в Сан-Франциско, как показано в табл. 4.1, и сделаем следующие допущения:

- все площадки подключены к Сан-Франциско по прямым волоконно-оптическим линиям и находятся на указанном расстоянии[2];
- в волоконно-оптической линии свет проходит приблизительно 1000 км за 5 мс.

Таблица 4.1. *Расстояние и средняя задержка при распространении сигнала от Сан-Франциско до других площадок*

	Нью-Йорк	Лондон	Токио	Сидней	Йоханнесбург
Расстояние от Сан-Франциско	4130 км	11 027 км	17 944 км	11 934 км	16 958 км
Задержка	21 мс	55 мс	90 мс	60 мс	85 мс
Время приема-передачи	42 мс	110 мс	180 мс	120 мс	170 мс

Теперь умножьте время приема-передачи (*англ.* round-trip time, RTT) на размер запроса. Разница между временем доступа на сайт из Нью-Йорка и Токио существенна. В реальном мире мы должны учитывать тот факт, что большинство площадок не соединены волоконно-оптическими линиями напрямую, среды передачи имеют разную задержку, и при каждом сетевом прыжке сетевые устройства добавляют задержку, связанную с обработкой маршрута. Кроме того, нет никаких гарантий относительно другого трафика в тех же сегментах сети.

Чтобы преодолеть ограничения, связанные с задержками в сети при связи между площадками, желательно иметь копию сайта, размещенную достаточно близко к вашим заказчикам, чтобы эти задержки оказались незначительными. Хотя вы можете сделать это, создав собственную глобальную сеть, гораздо проще передать эту работу на аутсорсинг CDN, которые берут на себя бремя управления глобальным массивом центров обработки данных, называемых точками присутствия (*англ.* points of presence, PoP). Распределив свой сайт на локальную PoP, вы можете уменьшить время отклика для пользователей, находящихся вблизи этих точек до значений менее 1 мс.

Выбирайте CDN на основе ряда параметров (таких как доступность, обслуживаемые регионы и варианты маршрутизации), чтобы оптимизировать расходы. Вот что можно сделать с помощью CDN:

- уменьшить время загрузки, разместив контент ближе к потребителям;
- снизить затраты на обеспечение пропускной способности. Вместо того чтобы совершать множество лишних перемещений из одного конца страны в другой, большинство запросов остаются в переферийной среде, а для ответов извлекается кешированное содержимое;

---

[2] В действительности сети не соединяются таким образом. Сложный комплекс партнерских отношений и географических точек задействует различные уровни сетевой инфраструктуры. Узнайте больше о точках обмена интернет-трафиком и о том, как подключаются интернет-провайдеры и CDN, из заметки Cloudflare Learning Center (**https://oreil.ly/Og7DC**).

- повысить доступность и резервирование за счет наличия многочисленных копий вашего контента в разных уголках мира;
- повысить безопасность за счет смягчения последствий распределенной атаки типа отказа в обслуживании (*англ.* distributed denial-of-service, DDoS). При DDoS-атаке злоумышленники пытаются наводнить сайт трафиком, чтобы исчерпать ресурсы системы. Некоторые поставщики CDN могут защитить ваши серверы от вредоносной активности, что позволит вашей системе избежать значительного простоя.

Использование CDN помогает решить некоторые из ваших проблем, но это добавляет целый ряд сложностей в управлении сервисами, специфическими конфигурациями, предоставляемыми CDN, и кешем вашего сайта.

Если в настоящее время вы используете CDN, ознакомьтесь с документацией поставщика услуг, чтобы узнать, когда следует очищать кешированные ресурсы. Рассмотрите следующие ситуации:

- проблемы возникают у некоторой части ваших пользователей. Например, кто-то внес изменение, которое повлекло за собой нежелательные последствия, в основе которых лежали существующие кешированные данные;
- проблемы возникают у всех ваших пользователей. Например, из-за плохо спроектированного сайта.

В целом избегайте полной очистки кеша, поскольку это приведет к каскаду запросов на его повторное заполнение.

Если вы используете кеш на веб-сервере, уделите время изучению такого вопроса, как *отравление кеша* — онлайн-атаки на ваши кешированные данные, когда злоумышленник использует уязвимость в вашем веб-сервере (без установленных исправлений), вызывающую изменения в вашем кеше, которые затем передаются другим пользователям. У Джеймса Кеттла (James Kettle) есть отличный ресурс о том, как работает кеш и как происходит отравление кеша (**https://oreil.ly/74vNx**).

Рекомендации по вашей сетевой стратегии

Разобравшись с сетевыми технологиями (виртуализация сетей, программно-определяемые сети и сети доставки контента), вы можете начать выстраивать сетевую стратегию. Необходимо учитывать следующие факторы.

- Разберитесь, какая задержка является приемлемой. Рассмотрите возможность перемещения необходимых систем ближе к конечным пользователям для сокращения задержки, будь то кеширование, зеркалирование систем или сегментированные данные. Для этого требуется хорошо понимать, как и где пользователи подключаются к вашей инфраструктуре, — например, через телефоны (ненадежная беспроводная связь), ноутбуки (в основном надежные беспроводные соединения), проводные соединения, а также знать физическое расстояние на основе мировых рынков.

- Используйте новые протоколы в своих системах:
  - задействуйте HTTP/2, чтобы обслуживание пользователей стало более быстрым и качественным;
  - используйте для передачи данных по сети протокол QUIC, чтобы поддерживать бесперебойное соединение даже при переходе мобильных пользователей от одного сетевого подключения к другому.
- Будьте в курсе угроз компьютерной безопасности и следите за рекомендациями, касающимися используемого вами программного обеспечения.

Заключение

Проводные, беспроводные или виртуализированные сети служат для обмена данными между ресурсами и службами, которыми вы управляете. Как и в случае с развитием devops, граница между системным администрированием и разработкой программного обеспечения стала размытой, стирается и различие между системными администраторами и администраторами сетей.

Современные программно-определяемые сети используют централизованный подход для эффективной маршрутизации сетевого трафика, предоставляя операторам сети инструменты для регулирования трафика, защиты от вредоносного ПО и несанкционированной активности, а также выставления счетов пользователям на лимитных подключениях. Аналогичным образом сети доставки контента повышают качество работы пользователей, находящихся в разных уголках земного шара, за счет кеширования данных веб-сайта на объектах, которые физически расположены рядом с ними.

Когда вы начинаете создавать свою сетевую инфраструктуру и управлять ею, нужно учитывать, как различные ресурсы в вашей сети взаимодействуют друг с другом, какой объем данных передают и насколько они устойчивы к задержкам. Благодаря современным подходам вы и ваши пользователи получите быструю, безопасную и отказоустойчивую сеть.

ЧАСТЬ II

Методы

В части II я делюсь методами, которые уменьшат влияние развивающихся технологий на ваши системы и повысят их надежность и устойчивость ко всеобщей радости пользователей этих систем.

Этот набор глав поможет вам поразмыслить о различных методах, которые подходят для ваших систем, позволяют повысить удобство сопровождения и использования и упрощают работу. Я хочу, чтобы вы могли брать на вооружение практики, которые улучшат поддержку ваших систем.

ГЛАВА 5

Набор инструментов сисадмина

В этом наборе собраны инструменты, ориентированные на решение определенного круга задач. В первые годы работы сисадмином у меня в сумке для ноутбука был настоящий набор инструментов, которые я применяла в работе. Со временем они менялись, но среди самых необходимых были ручка и маркер, стикеры (обернутые вокруг кабелей, они очень полезны при поиске неисправностей), набор небольших отверток для вскрытия корпусов и замены аппаратной начинки, загрузочные компакт-диски для множества операционных систем, а также куча всевозможных кабелей и донглов.

В комплекте инструментов современного сисадмина оказывается все больше нефизических инструментов. Ваша рабочая среда — это ваша первая управляемая система, она тоже входит в этот комплект. В этой главе рассказывается о том, как создать свой набор цифровых инструментов, используя кодовую среду разработки, чтобы автоматизировать повседневные задачи и улучшить взаимодействие с пользователями и коллегами — вы можете поделиться с ними своим набором или перенять их инструменты и методы.

Что представляет собой ваш цифровой инструментарий?

Как системный администратор, вы отвечаете за надежность систем. Какова бы ни была ваша конкретная роль, какой бы ни была система, вам нужен безопасный способ имитации реалистичной модели производственной среды, чтобы экспериментировать с ней и разбираться с рабочими процессами. В конечном итоге вам нужно определить устойчивые и рациональные подходы к работе.

Ваш цифровой инструментарий — это среда разработки, которая помогает вам минимизировать риск для любой системы, с которой взаимодействуют заказчики. Она предоставляет вам набор инструментов и технологий для разработки кода, изолированного от реальной среды. Ваша среда может находиться у вас на ноутбуке или на рабочей станции, а может быть конфиденциальной изолированной программной средой на удаленной системе поставщика облачных решений.

Ваш набор инструментов помогает решать следующие задачи:

- работа в автономном режиме;
- отладка кода/конфигурации;

- адаптация новых сотрудников или членов команды с использованием соответствующего контекста, необходимого для выполнения конкретной задачи;
- соответствие требованиям политики и рекомендуемым практикам с помощью соблюдения технических стандартов.

Как я перестал беспокоиться и полюбил свою среду разработки

Автор — Крис Деверс (Chris Devers)

Моя работа связана с поддержкой систем для вещания, и простои грозят серьезными последствиями, поэтому я должен чувствовать себя уверенно, прежде чем развертывать исправление при возникновении проблем. У меня могут быть идеи о том, как внедрить то или иное решение, но проведение экспериментов в этой сфере, как правило, не допускается. Поэтому мне нужен доступ к среде, которая в значительной степени повторяет производственную, чтобы опробовать разные варианты, никому не мешая и не подвергая опасности данные или сервисы.

Например, программное обеспечение, с которым я работаю, может создавать резервные копии материалов на магнитной ленте (LTO) для «холодного» архивного хранения. Но с ленточными библиотеками могут возникнуть всевозможные проблемы: кто-то удалит необходимую кассету, или она окажется нечитаемой. Иногда кассета физически в порядке, но не маркирована, поэтому библиотека не может отсканировать штрихкод. Или в роботизированный механизм для перемещения кассет попадает этикетка, и его заклинивает. Бывает, что оборудование работает нормально, но кабель передачи данных не полностью вставлен в разъем, и оно периодически отключается от сервера. Для сценариев «счастливого пути», когда оборудование работает, можно использовать программное обеспечение для виртуализированного ленточного накопителя. Но я не могу быть уверенным, развертывая исправление, которое не пробовал, что программное обеспечение не приведет к аппаратным сбоям.

Чтобы развернуть изменения в производственных системах, мне нужно точно знать, что эти изменения работают корректно в условиях, сравнимых с теми, где они будут запущены, и не вызывают нежелательных побочных эффектов. Локальные среды разработки, повторяющие производственные, дают мне возможность убедиться, что изменения, которые я вношу, будут работать так, как ожидается, не подвергая риску производственную систему.

Компоненты инструментария

Ваш инструментарий будет соответствовать набору задач и проектов, с которыми вы сталкиваетесь в работе. Он представляет собой комбинацию следующих элементов:

- редактор;
- языки программирования;
- фреймворки;
- библиотеки;
- приложения.

Конечно, каждый из этих компонентов допускает вариации в рамках своих функций. Рассмотрим эти компоненты более подробно.

Выбор редактора

Сисадмины пишут программный код, создают сценарии, инфраструктуру, документацию и тесты. Правильный текстовый редактор повышает эффективность работы, помогает выявить проблемы в программном коде на ранней стадии, предлагает варианты автозавершения кода на основе семантики языка, форматирует код в соответствии с ожиданиями команды и обеспечивает взаимодействие с другими инструментами.

Например, вы можете вручную создать Dockerfile — текстовый файл, содержащий инструкции по сборке контейнера Docker, отыскивая каждую инструкцию. Современный редактор предложит вам готовые фрагменты, соответствующие допустимым командам Dockerfile, что упростит и ускорит создание этого файла. Наведя курсор на команду в готовом файле Dockerfile, вы получите подробное описание того, что делает эта команда.

Какой функционал должен быть у редактора? Хотя вы, вероятно, уже знакомы с одним или более текстовыми редакторами, есть некоторые моменты, из-за которых стоит освоить еще один:

- встроенный статический анализ кода;
- автозавершение кода;
- слежение за отступами для соответствия принятым в команде правилам;
- распределенное парное программирование;
- рабочий процесс, интегрированный с git.

Относитесь с пониманием к тому, что другие пробуют и внедряют различные инструменты. Хотя vi или emacs могут обладать всеми нужными вам функциями, они могут не подойти другим пользователям. Освоение с нуля всех функций этих редакторов может показаться другим системным администраторам не лучшим вариантом использования времени, особенно когда им просто нужно эффективно выполнять свою работу.

Встроенный статический анализ кода

Вы можете ускорить разработку и уменьшить потенциальные проблемы с помощью инструментов статического анализа кода или расширений для редакторов (линтеров), выполняющих аналогичную функцию. Например, можете установить `shellcheck` и расширение `shellcheck` для написания сценариев `bash`. Тогда по мере написания кода оболочки (*англ.* shell code) редактор будет предупреждать вас о возможных проблемах. Следующий пример показывает, как найти все файлы с расширением *.png* в текущем каталоге. Я написал для этого такой код оболочки:

```
#!/bin/bash

for file_name in $(ls *.png) do
echo "$file_name" done
```

Текстовый редактор предупредил меня, что не рекомендуется перебирать файлы с помощью команды `ls` («iterating over `ls` output is fragile»). Поэтому я переделал код, удалил `ls` и использовал стандартную маску, как было рекомендовано:

```
for file_name in *.png do
echo "$file_name" done
```

Запуск линтера во время написания кода позволяет выявить и устранить проблемы. Существуют линтеры для многих типов файлов, от YAML до специфических языков. В редакторе можно задать параметры контроля качества программного кода, чтобы проверка выполнялась в реальном времени или, если это слишком отвлекает, после сохранения обновлений.

Автозавершение кода

Автозавершение кода повышает удобство при написании программного кода за счет предугадывания действий, которые вы собираетесь совершить. По мере ввода текста будут появляться опции автозаполнения. В некоторых языках автозавершение работает по умолчанию, в других требуется установка расширений.

Установление и утверждение правил, которые должны соблюдаться командой

Многие организации используют инструменты статического анализа кода для сохранения единого стиля написания программ, что облегчает командам поддержание общего репозитория программного кода. Например, команда может стандартизировать величину отступов в тексте и не спорить больше о том, что лучше — пробелы или табуляция. Каждый человек может настроить свой редактор для отображения предпочитаемого им варианта отступа. Кроме того, вы можете преобразовать значения интервалов, используемых в настоящее время в файле, чтобы они соответствовали новым требованиям.

Рабочий процесс, интегрированный с Git

В процессе работы над проектом полезно видеть свои изменения и видеть, что они учитываются. Имея перед глазами предполагаемые изменения, можно предотвратить неприятные сюрпризы — например, когда вы забываете загрузить файлы с исправлениями обратно в общий репозиторий исходного кода.

Выбор языков программирования

Даже если вы не разрабатываете приложения, овладение навыками разработки программ для оболочки или изучение еще одного языка программирования улучшит взаимодействие в команде и повысит производительность ее работы. Кроме того, автоматизация работы — от открытия заявок JIRA с предварительно заполненной метаинформацией до проверки вычислительных экземпляров на предмет несоответствия систем требуемым стандартам — высвобождает время команды, дает воз-

можность сосредоточиться на областях, требующих участия человека и творческого подхода.

Bash и PowerShell — подходящий выбор для большинства сред. Эти оболочки имеются в текущих версиях Linux и Windows, они всегда под рукой. Когда в сценарии оболочки накапливается более пятидесяти строк или он требует сложных структур данных, его становится труднее понять, что приводит к нестабильности механизма управления системой. Никто не хочет нарушать сценарий. В таких случаях может помочь преобразование скрипта в утилиту, написанную на языке программирования общего назначения. Такие языки, как Python, C#, Ruby и Go, обладают следующими преимуществами:

- улучшенная обработка ошибок;
- богатая коллекция библиотек;
- дополнительные инструменты для отладки и утилиты.

Как же выбрать конкретный язык? Учитывайте следующие аспекты.

Какие языки уже используются в вашей организации или команде? Сколько программного кода на определенном языке у вас уже есть?

Полезно научиться понимать любой язык (языки), который использует ваша команда разработчиков. Когда система работает не так, как ожидалось, бывает полезно проверить, в чем проблема — в программном коде или в тестах.

Удобно, когда несколько человек могут выполнять отладку или реализовывать функции.

Иногда стоит следовать принятым правилам, а иногда лучше быть не как все. Например, вполне допустимо выбирать языки и технологии, основываясь на тех навыках, которыми обладают специалисты команды.

Имеются ли инструменты и технологии, которые вы или ваша команда хотели бы внедрить, но они реализованы на языке, с которым никто в вашей команде не знаком?

Вы можете внедрить альтернативную технологию на базе того языка, который уже знает ваша команда, или использовать представившуюся возможность, чтобы развить свои компетенции и освоить новый язык программирования. Команды, которые не развиваются, начинают стагнировать. Они не смогут внедрять новое программное обеспечение, поскольку оно часто использует современные языки.

Включите в бюджет затраты на использование и поддержку тех инструментов, которые считаете нужными. Например, некоторые команды ценят возможности совместной работы, которые предоставляют технологии с открытым исходным кодом. Другим, наоборот, больше подходят коммерческие продукты с соответствующим обучением и поддержкой. Нельзя однозначно сказать, какой подход лучше. Правильным будет тот, который не противоречит сложившимся в команде традициям и который не создаст в будущем дополнительных сложностей при внедрении нового программного обеспечения.

Учитывайте, какое влияние оказывают различные виды изменений:

- новые версии языков могут нарушить совместимость с существующими базами исходного кода;
- новые библиотеки могут упростить сложную работу, но потребуют рефакторинга устаревшего программного кода;
- обновления для системы безопасности требуют прекращения использования уязвимых функций и их срочного рефакторинга. Любой достаточно популярный язык, активно поддерживаемый сообществом разработчиков, постоянно развивается. Поэтому при планировании задач, которыми ваша команда будет заниматься в будущем, следует рассматривать и альтернативные варианты.

Полезно документировать причины, по которым команда выбрала тот или иной язык. Это пригодится в будущем.

Какие языки чаще всего используют ваши коллеги по отрасли?

Широко распространенные в отрасли языки будут иметь больше ресурсов поддержки, более развернутую документацию и множество примеров программного кода на форумах сообщества.

С какими трудностями вы столкнулись, принимая предыдущие решения о внедрении языка?

Иногда, даже если определенный язык широко используется в вашей организации, сопутствующие проблемы могут мешать реализации нового проекта. Определение и документирование ваших размышлений при принятии нового языка является важным шагом.

Рефакторинг инструмента под новый язык требует времени и энергии и может потребовать от вас поддержки двух разных инструментов одновременно. Языки развиваются, так что даже если ваша команда придерживается одного основного языка, может потребоваться рефакторинг устаревшего кода, чтобы он продолжал работать с новыми версиями языка или библиотек. Например, организации, пытающиеся перейти с одного инструмента автоматизации инфраструктуры на другой, часто используют в итоге обе технологии. Кроме того, приложения с пересекающимися функциями вносят путаницу и усложняют инфраструктуру.

В конечном счете не существует одного правильного языка, который нужно изучать системному администратору. Вместо этого сопоставьте свой опыт с возможностями и навыками остальных членов команды.

Иногда операционная система включает в себя версию языка. Часто это устаревшая версия, и вам нужно будет обновить ее, чтобы использовать новейшие возможности языка. Не рекомендуется изменять язык, включенный в систему. Вместо этого установите нужную версию отдельно и настройте пути к исполняемым файлам соответствующим образом, чтобы работать с более поздней версией. Отдель-

ная установка поможет предотвратить нестабильность системы из-за изменения программного обеспечения. Это также помогает устранить неопределенные зависимости в средах.

Фреймворки и библиотеки

Дополнительные фреймворки или библиотеки, которые вам понадобятся, зависят от поставщиков услуг и языков, которые использует ваша организация. Приведем несколько примеров:

- AWS SDK для определенных языков;
- клиентские библиотеки PagerDuty API для управления конфигурациями PagerDuty;
- фреймворки для автоматизации ChatOps на определенном языке.

Это будут узкоспециальные решения для вашей среды и потребностей. Если функциональность будет меняться в разных версиях библиотек, возможны проблемы. Документирование версий этих фреймворков и библиотек и их индексация в средах позволяют избежать потери времени на отладку, когда выходные данные оказываются различными.

Другие полезные утилиты

Помимо редактора, языков, фреймворков и библиотек, в ваш инструментарий входят дополнительные приложения. Для вашей организации будут полезны различные инструменты для следующих задач:

- отслеживание заявок или ошибок;
- мониторинг инфраструктуры и приложений;
- оповещения;
- управление конфигурацией, оркестровка контейнеров и предоставление ресурсов для инфраструктуры;
- поточная обработка;
- репозитории артефактов;
- версии;
- исходный код;
- чат;
- управление знаниями.

Все эти инструменты можно помещать вместе с кодом инфраструктуры в готовые контейнеры, виртуальные машины или на удаленную систему, предоставляемую поставщиком облачных решений.

Кроме того, вы можете настроить командную строку с помощью настроек оболочки. Файлы с точкой — это файлы, которые обычно (но не всегда) начинаются

с точки и помогают с созданием резервных копий и настройкой наших систем. Вы можете делиться этими файлами с другими инженерами, и последние смогут повысить свою производительность за счет применения новых инструментов. Некоторые организации используют файлы с точкой для настройки отдельных функций новой системы и повышения ее производительности. Хотя файлы с точкой могут показаться более привычными в Unix-подобных системах, они доступны и в Windows.

Имейте в виду, что не стоит просто копировать чужие файлы с точкой в свою среду, не разобравшись полностью в коде. Кроме того, конфигурации, которые работают в одном случае, могут оказаться неоптимальными в другой ситуации.

Со временем вы также создадите наборы инструментов, которыми будете пользоваться независимо от своей среды. Я хочу поделиться некоторыми из моих любимых. Многие из перечисленных ниже рекомендуемых инструментов являются кросс-платформенными, хотя более привычно их видеть в UNIX.

Silver Searcher

Silver Searcher (**https://oreil.ly/Km2em**), или сокращенно Ag, позволяет выполнять поиск в репозиториях программного кода. Ag работает быстро и игнорирует шаблоны файлов из *.gitignore*. Он также может быть интегрирован с редакторами. При устранении ошибок или отыскании других дефектов кода, подобных иголке в стоге сена, возможность поиска определенной строки может очень пригодиться, чтобы понять, чем вызвана ошибка.

bash-completion

Современные оболочки автоматически завершают написание команд. Вы вводите начало команды, нажимаете клавишу табуляции и можете просмотреть возможные варианты завершения. `bash-completion` расширяет эту возможность и позволяет добавлять опции автозавершения. Расширения могут использоваться совместно всей командой.

`cURL`

`cURL` — это инструмент командной строки и библиотека для передачи данных. Например, вы можете использовать его для определения возможности подключения к URL-адресу, что часто требуется при проверке веб-сервиса. Вы также можете использовать его для отправки или получения данных с URL-адреса или получения заголовков HTTP, чтобы просмотреть определенные коды ответа сервера.

Docker

Docker предоставляет механизм для создания изолированных сред, называемых *контейнерами*. В Dockerfile заключены ОС, файлы среды и требования приложения. Вы можете добавить Dockerfile в проект и подтвердить его в системе контроля версий.

При установленном Docker и доступе к Dockerfile для включения нового сотрудника в проект достаточно запустить команду `docker run` для создания рабо-

чей тестовой среды. Эта тестовая среда будет еще больше соответствовать производственной среде, если запустить инфраструктуру на контейнерах.

`gh`

Используя Git для контроля версий и GitHub в качестве репозитория проекта, `gh` (**https://oreil.ly/dSBI3**) расширяет функциональность Git, помогая выполнять задачи GitHub из командной строки.

Например, если я хочу протестировать запрос на включение внесенных изменений (PR), поданный в проект, я могу использовать команду `gh pr checkout <issue-number>`, чтобы проверить этот конкретный запрос и провести локальное тестирование в своей среде, прежде чем утвердить PR для слияния.

Git

Git — это распределенная система контроля версий. Более подробную информацию о контроле версий вы найдете в главе 6.

HTTPie

HTTPie — это HTTP-клиент командной строки для тестирования, отладки и взаимодействия с API с поддержкой JSON и подсветкой синтаксиса.

`jq`

`jq` — это легкий универсальный процессор JSON для командной строки. Используя его совместно с `cURL`, вы можете обрабатывать вывод JSON из командной строки.

`mkcert`

`mkcert` (**https://oreil.ly/T8R4i**) используется для генерации локально доверенных SSL-сертификатов для разработки.

ShellCheck

ShellCheck — это утилита, которая показывает проблемы в сценариях оболочки `bash` и `sh`. Она может выявлять распространенные ошибки и неправильно используемые команды. Вы можете игнорировать определенные проверки, если ваша команда сама производит проверку своего программного кода с помощью файла конфигурации.

`tmux`

`tmux` (**https://oreil.ly/Qrkny**) — это терминальный мультиплексор, который позволяет работать с несколькими программами в одном окне терминала (мультиплексор).

tree

tree — это утилита, входящая в состав большинства ОС, которая показывает содержимое каталога в виде дерева. Визуализация структуры файловой системы может понадобиться, например, при составлении документации, чтобы объяснить другим, что следует ожидать. Зачастую наглядная демонстрация ожидаемого результата вместо простой фразы «в текущем каталоге» позволяет избежать неправильного толкования.

Заключение

Инструментарий сисадмина — это набор инструментов и технологий, минимизирующий когнитивную нагрузку и повышающий эффективность. Он позволяет автоматизировать установку и настройку системы, а также управление ею, получив в результате согласованную и повторяемую основу.

Правильная локальная среда разработки предоставляет нужный текстовый редактор, языки программирования, фреймворки, библиотеки и другие приложения для экспериментов, изучения и оценки изменений в производственной среде.

Перенимайте то, что подходит вам, делитесь своими находками с коллегами, налаживайте совместную работу, общими усилиями снижая нагрузку на каждого человека при выполнении ежедневных задач.

Дополнительные ресурсы

В статье Томаса А. Лимончелли (Thomas A.Limoncelli) «Low-Context DevOps» подробно рассказывается о создании сред, которые поддерживают вашу производительность независимо от имеющихся у вас фундаментальных знаний.

А следующие ресурсы посвящены файлам с точкой:

- файлы с точкой на Github (**https://oreil.ly/KwzW2**);
- статья Ларса Капперта (Lars Kappert) на платформе Medium «Getting Started with Dotfiles» (**https://oreil.ly/l1zwu**).

ГЛАВА 6
Контроль версий

Представьте, что вы ведете небольшой бизнес вместе с партнером, который живет в другом месте. У вас есть совместный банковский счет для оплаты счетов. Хотя интернет-банкинг позволяет вам в любое время зайти в систему и посмотреть состояние ваших финансов, он не сообщает вам о планируемых изменениях и не дает представления об управлении финансами. В результате вам часто начисляют пени за несвоевременную оплату счетов, а иногда вы оплачиваете и комиссию за перерасход средств из-за дублирования платежей одному и тому же поставщику.

Вы внедряете систему, которая позволяет планировать оплату счетов и предоставляет отчетность по запланированным изменениям. Новая система улучшает взаимодействие и позволяет вам выполнять всю необходимую работу. По большому счету управление финансами — это не главное отличие вашего бизнеса от других.

Теперь замените «совместный банковский счет с бизнес-партнером» на систему, которой вам нужно управлять. Ваш бизнес-партнер — это остальные члены команды, с которыми вы работаете (в том числе и вы сами в будущем, когда в два часа ночи вам приходится разбираться с состоянием системы и допускать случайные ошибки). У каждого человека в команде могут быть свои предпочтения по управлению системой. Если вы не выработаете общий подход к этому вопросу, то будете каждый раз конфликтовать, пытаясь устранить неполадку. А теперь, предположим, вы внедряете контроль версий и используете инструменты, уже имеющиеся в вашей организации (например, Git, Artifactory, GitHub и GitLab). В этом случае вы получаете наглядность, отслеживаемость и успешно разрешаете все конфликты.

В этой главе я помогу вам понять, что такое контроль версий, чтобы вы могли выполнять свою работу по-новому — отслеживать изменения и управлять ими, обеспечивая воспроизводимость, отслеживаемость и разрешая конфликты.

Что такое контроль версий?

Контроль версий — запутанное понятие, отчасти потому, что мы используем одни и те же слова и аббревиатуры для обозначения разных вещей. Это слишком часто используемый термин со множеством значений.

Для развертывания системы требуется исходный код или бинарные пакеты, конфигурации, сценарии развертывания и сопутствующие процессы, чтобы все в итоге оказывалось на своем месте и поддавалось сквозному контролю. Контроль версий — это метод резервного копирования и способ минимизации риска для вашей системы.

Поскольку мы (ошибочно) употребляем термины «*контроль исходного кода*» и «*контроль версий*» как взаимозаменяемые, то полагаем, что контроль версий как метод входит в сферу деятельности одних лишь разработчиков программного обеспечения — для поддержания исходного кода. Однако практика контроля версий — отслеживание изменений в конфигурационных файлах, скриптах и образах сборки — имеет решающее значение и для системного администратора, позволяя ему создавать несколько сред с одинаковыми конфигурациями, реплицировать и восстанавливать системы до исходного состояния, а также применять опубликованные рекомендуемые практики для соответствия требованиям регуляторов.

Взгляните на рис. 6.1. В широком смысле слова, как показано в самом большом прямоугольнике (и именно на нем я сосредоточусь в этой главе), контроль версий — это *методы* отслеживания изменений данных и управления этими изменениями. Они могут применяться к текстовым файлам, таким как исходный код или файлы конфигурации, а также к выходным файлам сборки и образам.

Рис. 6.1. Разница между контролем версий как методом, программным обеспечением и услугой

Средний прямоугольник содержит *системы* контроля версий (СКВ). СКВ — такие как Git, распределенная система, и Subversion (SVN), централизованная система — являются конкретными реализациями *программного обеспечения* для контроля версий, которые по-разному обрабатывают функции репозиториев и ветвления.

Системы управления артефактами — это еще один тип систем контроля версий, не показанный на этом рисунке. Системы управления артефактами управляют хранилищами скомпилированных двоичных файлов вместо текстовых файлов.

Наконец, два внутренних прямоугольника представляют реализации СКВ — управляемые провайдером (например, GitHub и GitLab) и самоуправляемые.

Преимущества контроля версий

В самом начале моей первой официальной работы в качестве системного администратора, когда у меня уже были ключи от королевства (т. е. пароль суперпользователя), мой коллега объяснил мне, как обновить конфигурацию. Сначала нужно было создать резервную копию файла, чтобы в случае необходимости вернуть систему в заведомо исправное состояние. Затем следовало внести правки в файл. В некоторых системах это означало использование редактора `ed`, который не столь удобен, как современные редакторы. Можно было легко ошибиться, и я часто полагалась на резервный файл, внося изменения. Затем я перезапускала сервис и убеждалась, что он работает нормально. По мере того как я вникала в различные тонкости управления системами, я начала замечать множество старых резервных копий файлов, расположенных в служебных каталогах, со случайными именами от *.bak* до *.bak.date*, *.date.bak*, *.name.date.bak*. Было сложно понять, какие из них следует сохранить, а какие удалить.

Сегодня вам не нужно настраивать системы напрямую. Вы можете использовать СКВ, и вам не придется придумывать собственную систему наименования резервных копий. Но даже если вы не перешли на систему, которая автоматически следит за конфигурацией СКВ, вы все равно можете развернуть конфигурации с контролем версий для своих уникальных «серверов-снежинок», чтобы обеспечить их систематическое обслуживание, стабильную работу и документирование.

При использовании СКВ вы получаете возможность отслеживания изменений и управления ими, а также следующие функции:

- копия каждой версии;
- контроль доступа для создания, удаления и внесения изменений;
- история изменений, включая информацию о том, кто несет ответственность за то или иное изменение;
- процедура предотвращения или разрешения конфликта;
- способность документировать изменения.

Будучи единственным администратором некоторой системы, вы можете использовать контроль версий для отслеживания состояния системы во времени и документирования решений об изменениях. Кроме того, практика контроля версий является основой для сотрудничества с другими членами вашей команды и внутри организации. Она помогает им понять, как вы управляете своими системами, используя стандартные методы.

Использование контроля версий для кода инфраструктуры, конфигурационных файлов и системных инструментов в дополнение к исходным текстам дает важные преимущества.

Воспроизводимость

Вы можете развернуть определенную версию системы или среды с помощью сценариев, конфигурации и программных артефактов.

Инкультурация

Вы помогаете новым членам адаптироваться в команде, предоставляя им журналы изменений системы контроля версий, что способствует повышению производительности и эффективности работы.

Наглядность управления изменениями

Вы можете предоставить доступ на чтение репозиториев вашей команды людям, которые не входят в ее состав, но хотят просматривать или обсуждать ваши проекты. В результате они могут наблюдать, какие изменения вносятся, и видеть любую незавершенную работу. Кроме того, вы можете предоставить разрешения, позволяющие отдельным лицам утверждать изменения, что обеспечит взаимодействие без возможности изменения содержимого хранилищ.

Отслеживаемость

Вы используете СКВ для отслеживания изменений. История изменений системы обеспечивает аудиторский след, позволяя ответить на вопросы о том, кто создал каждую систему и каково ее предназначение. Кроме того, отслеживаемость может снизить затраты, поскольку вы можете проводить аудит систем, чтобы убедиться, что они по-прежнему необходимы.

Организация инфраструктурных проектов

Не существует единственно правильного способа организации проектов, когда нужно выбирать между вариантами с одним проектом на репозиторий (мультирепозиторий, *англ.* multirepo) или размещением всех проектов в одном репозитории (монорепозиторий, *англ.* monorepo). Каждый подход подразумевает ряд компромиссов, включая организацию программного кода, управление зависимостями и контроль конфигурации.

Организация программного кода

При использовании multirepo вы соглашаетесь с вариантом «один проект на репозиторий», но целостное определение проекта отсутствует. В результате некоторые проекты хорошо согласуются с определением проекта, а другие — не очень. Например, подумайте о таком случае: где будет находиться отдельный вспомогательный сценарий для настройки ноутбука? Вы можете поместить его в отдельный репозиторий. А можете сгруппировать с другими произвольными вспомогательными сценариями или со всем программным кодом, относящимся к рабочей станции.

А как вашей команде найти этот вспомогательный скрипт или определить, что он уже существует? В монорепозитории поиск ограничивается одним репозиторием. При использовании нескольких репозиториев команде необходимо знать обо всех возможных местах для поиска.

По мере увеличения размера проектов в монорепозитории вы можете столкнуться с проблемой снижения производительности при их проверке и сборке. При использовании нескольких репозиториев проекты не влияют друг на друга.

Множество репозиториев может приводить к дублированию кода или сложному переплетению зависимостей.

Какой бы модели вы ни придерживались, в идеале ваша организация должна выработать стандартный подход, снижающий когнитивную нагрузку на людей, чтобы они понимали порядок действий при организации нового проекта.

Поддержка версий

При использовании нескольких репозиториев поддержка версий может осуществляться отдельно для каждого проекта.

Управление зависимостями

В случае единственного репозитория вы можете привязать зависимости к определенным версиям, что может быть полезно, когда вашим проектам нужна одна и та же версия программного обеспечения. Однако если для них требуются разные версии одних и тех же пакетов программного обеспечения, вы можете столкнуться с проблемами, пытаясь соблюдать требования и заставляя один проект обновлять любой программный код, чтобы можно было использовать последнюю версию.

При использовании множества репозиториев каждый проект может иметь зависимости, без конфликтов привязанные к необходимой версии.

Управление конфигурацией

Когда отдельные функциональные команды должны взаимодействовать друг с другом в рамках различных проектов и по-разному работать с единственным репозиторием, предпочтения в работе могут стать причиной конфликтов между группами, вызывая проблемы при проверках и слиянии кода.

Также могут возникнуть вопросы о том, кто «владеет» содержимым репозитория, в который вносят вклад несколько команд, и, следовательно, отвечает за внесение изменений и проверяет, что эти изменения не нарушают работу других команд.

Размещение материала в более мелких репозиториях может уменьшить масштаб этой проблемы, но приводит к повышению крутизны кривой обучения, т. к. приходится разбираться, в каких репозиториях следует работать при конкретном требовании к изменению.

Этот список компромиссов не является исчерпывающим. Ваша команда должна будет решить, что лучше — один или несколько репозиториев. Лучше всего задокументировать свои предпочтения, чтобы, когда придет время работать с другими командами или привлекать новых участников, ваши коллеги понимали, почему был выбран тот или иной подход.

Заключение

Контроль версий — это метод управления данными и отслеживания изменений в них. Инфраструктура вашей системы и то, как вы ею управляете, — это критиче-

ски важные данные. Применяйте контроль версий для улучшения управления инфраструктурой системы. СКВ облегчают управление, благодаря следующим функциям:

- отслеживаемость изменений за счет аудиторского следа (информация о каждом изменении и версии проекта);
- контроль доступа для определения лиц, которые могут изменять данные;
- механизмы для разрешения конфликтов и обеспечения видимости.

Эти возможности предоставляют вам и вашей команде гибкий набор инструментов для совместного обслуживания инфраструктуры и ее расширения.

ГЛАВА 7

Тестирование

Из многочисленных разговоров с другими сисадминами у меня складывается впечатление, что они не считают себя тестировщиками. Но независимо от того, считаем мы себя тестировщиками или нет, мы используем тесты для получения информации о состоянии задач и для изучения и лучшего понимания сред. Мы хотим предотвратить эти ужасные подъемы в два часа ночи при возникновении аварийной ситуации или, по крайней мере, научиться быстро устранять проблемы. В этой главе вы узнаете, какие тесты следует подготавливать, чтобы использовать возможности автоматизированного тестирования, научитесь оценивать эффективность тестов и изменять их в соответствии со своими задачами. Эти базовые понятия понадобятся вам для применения тестирования по отношению к инфракоду (*глава 11*) и управлению инфраструктурой (*глава 12*).

Вы уже тестируете

Приходилось ли вам когда-нибудь устанавливать набор программного обеспечения на непроизводственную или неживую систему, наблюдая за тем, как она реагирует, и выяснять возможное влияние проблем на пользователей? Такой подход на основе тестирования вручную известен как *исследовательское тестирование*. Цель исследовательского тестирования — помочь обнаружить неизвестные проблемы, экспериментируя с системой и рассматривая области, которые могут больше нуждаться в субъективном анализе для определения их состояния. В статье «Exploratory Testing Explained» Джеймс Бах (James Bach, **https://oreil.ly/BZEa1**) определил исследовательское тестирование как «одновременное обучение, составление тестов и их выполнение». В отличие от скриптовых тестов, исследовательское тестирование проводится на основе вашего понимания и точки зрения, поэтому зависит от ваших личных предпочтений.

Вы можете повысить уровень ручного исследования и добавить объективности, применяя более строгий анализ — определяя цели тестирования с короткими циклами обратной связи для получения информации о дальнейших действиях. Затем, работая с инженерами по программному обеспечению и тестировщиками, вы можете помочь сформировать правила тестирования, чтобы исключить часть ручного тестирования для их скриптов и кода инфраструктуры (инфракод). Эти скриптовые тесты обеспечивают следующие преимущества.

Повышается доверие вашей команды к вашему программному коду

Тесты позволят вам меньше опасаться последствий при внесении изменений. Вместо того чтобы ожидать от каждого работника безупречного исполнения,

создайте системы безопасности, которые помогут им уверенно проводить изменения.

Ускоряется доставка рабочих инструментов и инфраструктуры

Благодаря тестам вы можете быстро создавать релизы, будучи уверенными в том, что если тесты прошли, то и релиз должен быть надежным.

Появляется возможность заниматься новыми проектами

Другие люди могут взять на себя ответственность за выполненную вами работу, если существуют автоматизированные тесты. Вы создаете безопасное пространство, где люди могут учиться и вносить изменения.

Появляется возможность документировать ожидания и контекст кода

Хорошие тесты позволяют качественно описать ожидаемую функциональность.

Тестирование поможет вам предоставить работающий продукт, включающий инфраструктуру и скрипты, которые помогут вашей системе работать эффективно, устранит скопление знаний в отдельных не связанных между собой точках и повысит уверенность в том, что проблемы не достигнут конечных пользователей.

Полезно, чтобы новые члены команды изучали продукты и процессы в ходе обучения. Их непредвзятое мнение будет способствовать повышению качества, поможет устранить проблемы с продуктом и процессами и выявить недоразумения и неувязки, которые могут уже присутствовать в команде.

Давайте рассмотрим другие распространенные типы тестов, которые вы можете разработать и использовать для создания автоматизированного тестирования: линтинг, модульные, интеграционные и сквозные тесты.

Общие виды тестирования

С помощью тестирования можно разрабатывать более эффективные инструменты и инфракод. Понимание различных типов тестирования, их преимуществ и недостатков, поможет вам разрабатывать удобные в сопровождении тесты с соответствующими уровнями. Поскольку точного определения этих типов тестов не существует, в разных командах эти тесты могут различаться. Например, некоторые команды Google создают наборы тестов, ориентируясь на размер, а не на тип (**https://oreil.ly/EgVgm**).

Линтинг

Линтинг — это базовый вариант статического анализа для решения проблем со стилевыми соглашениями. С помощью линтинга вы можете выявить проблемы в программном коде на ранней стадии, не придумывая специальных тестов. Кроме того, он поможет выявить логические ошибки, которые приводят к уязвимостям в системе безопасности. Линтинг отличается от инструментов форматирования кода тем, что он анализирует работу кода, а не только его внешний вид.

Вы можете получить несколько предупреждений, если запустите инструмент линтинга на существующем проекте. Стилистическое оформление программного кода выполняется по-разному в разных командах. Поэтому изменения функционального кода, вносимые одной командой кодеров, могут не понравиться другой. Вместо того чтобы сразу вносить изменения, проанализируйте результаты и задайте общие правила написания кода для линтера.

Есть три главные причины использования линтеров при разработке: для обнаружения ошибок, повышения читабельности кода и его стандартизации.

Обнаружение ошибок

Лучшее время для поиска ошибки — сразу же, как только она возникла, когда программный код еще свеж в вашей памяти и вы четко представляете себе, что хотели написать. Линтинг позволяет исправлять код в процессе написания на основе этого известного вам контекста. Хотя можно выполнять линтинг вручную, во многих редакторах есть соответствующие плагины, которые практически сразу же оповещают о потенциально проблемном коде. Вы устраняете проблемы по мере их возникновения, а не после того, как код уже готов и отправлен на проверку.

Повышение читабельности

Упорядоченный, легко читающийся код проще поддерживать, исправлять, легче расширять его функциональность. Когда вы обращаетесь к уже имеющемуся исходному коду, уже мало кто помнит контекст, но если разработчик строго придерживался правил, легче войти в курс дела. Линтеры способствуют повышению читабельности.

Стандартизация

Стандарты и методы, направленные на упорядочение кода, обеспечивают его связанность. Установка стилей исключает споры в командах кодеров, поэтому вы можете сосредоточиться на обсуждении действительно важных вещей — например, на проектировании архитектуры или исправлениях безопасности.

Для настройки линтера служит конфигурационный файл. Он позволяет реализовать принятые в командах стандарты и игнорировать или изменять правила по умолчанию. Например, длина строки по умолчанию для RuboCop, линтера языка Ruby, составляет 80 символов (**https://oreil.ly/9pvRX**). Разрешение современных дисплеев намного выше, чем у традиционных 72-символьных TTY-дисплеев, и ваша команда может посчитать удобным, когда в строке больше символов. Как показано в примере 7.1 *(см. ниже)*, вы можете задать длину строки в 100 символов, создав или обновив файл конфигурации RuboCop, *rubocop.yml*, в репозитории исходного кода проекта. Это значит, что для всех, кто обращается к проекту, запустится линтинг и предупреждение при длине строки менее 100 символов выводиться не будет.

Пример 7.1. Конфигурация RuboCop для обновления проверки количества символов в строке

```
Metrics/LineLength: Max: 100
```

Некоторые при написании кода вставляют два или четыре пробела, другие вместо пробелов используют символы табуляции. Теперь члены вашей команды могут проверять свой код и конфигурации на соответствие стандартам прямо в редакторе. И поэтому при проверке кода можно сосредоточиться на деталях реализации, а не на стилистических проблемах.

Вот несколько линтеров для распространенных языков, которые сисадмины используют в своей повседневной работе:

- ShellCheck для сценариев оболочки (**https://oreil.ly/saNSu**);
- jsonlint для JSON (**https://oreil.ly/U3XTU**);
- yamllint для YAML (**https://oreil.ly/Nvqgm**);
- Black для Python (**https://oreil.ly/QmFPJ**);
- Prettier для CSS, HTML, JavaScript, Markdown и других языков (**https://oreil.ly/tfMFe**).

Модульные тесты

Модульные тесты, или юнит-тесты (*англ.* unit test) — это небольшие, быстрые тесты, которые проверяют, что некоторая часть программного кода работает так, как ожидается. Они не запускаются на реальном экземпляре работающего кода. Любые ресурсы, которые зависят от вызовов внешних ресурсов, блокируются (например, запросы к базе данных или вызовы определенных сервисов). Такая блокировка делает модульные тесты исключительно полезными для быстрой оценки корректности кода, т. к. они быстро выполняются (обычно выполнение занимает менее секунды). При использовании модульных тестов вы не проверяете код на реальных экземплярах системы, поэтому не получаете представления о проблемах подключения или зависимостях между компонентами.

Модульные тесты обычно являются основой стратегии тестирования проекта, поскольку быстро выполняются, менее подвержены сбоям и искажениям и дают возможность локализовать ошибки. Они помогают ответить на вопросы о проектном решении, провести регресс (убедиться, что новый код не сломал старые части кода), понять, правильно ли работает программа и можно ли добавить в нее новые функции.

Когда вы пишете модульные тесты, убедитесь, что они тестируют ваш код. Например, когда вы пишете инфракод для конфигурирования файла или создания каталога, модульный тест должен подтвердить, что вы написали код для выполнения данных действий, а не то, что ваша платформа инфракода знает, как выполнить эти задачи. Напишите тесты, которые описывают желаемые конечные результаты и проверяют ваш код.

Примером юнита в инфракоде может быть управляемый файл, каталог или вычислительный экземпляр. В модульном тесте для проверки юнитов описаны требования к файлу, директории или вычислительному экземпляру, включая конкретные атрибуты. Юнит-тест описывает ожидаемое поведение.

Интеграционные тесты

Интеграционные тесты проверяют поведение нескольких объектов, работающих вместе. Конкретное поведение интеграционных тестов может варьироваться в зависимости от того, что ваша команда понимает под «множеством объектов». Интеграционные тесты могут быть как узконаправленными, например, для проверки совместной работы двух юнитов, так и расширенными, проверяющими совместную работу множества более значимых модулей. Интеграционные тесты проводятся в эфемерной среде и не проверяют каждый элемент проекта; они дают представление о поведении проекта в целом.

Поскольку интеграционные тесты не проверяют все на свете, они не позволяют точно определить проблему. Кроме того, выполнение интеграционных тестов занимает уже минуты из-за повышенной сложности настройки потенциальных зависимостей инфраструктуры, включая другие сервисы и программное обеспечение. Однако, например, они помогут вам понять, в каких условиях среды произошел сбой и что послужило причиной падения. Вы можете проверить, успешно ли устанавливается и конфигурируется база данных и запускается ли она должным образом, например можно ли к ней подключиться.

Сквозные тесты

Наконец, *сквозные тесты* (end-to-end, E2E) проверяют поток поведения функций проекта от начала до конца во временной среде с реалистичными тестовыми данными. Тесты E2E подтверждают, что приложения и сервисы, определенные инфракодом, работают надлежащим образом. Как вы догадываетесь, предоставление и настройка новых экземпляров и их прогон через набор тестов могут занять довольно много времени. Кроме того, падение сквозного теста не обусловлено одним компонентом и не позволяет локализовать неисправность. Сквозные тесты проверяют конкретные выходные данные функций и требуют более частых изменений в коде теста. Например, в тестовой среде в зоне доступности Amazon с сетевыми проблемами могут наблюдаться периодические сбои. Если надежность тестов невысока, вероятность того, что люди будут тратить усилия на их поддержание, снижается. В результате качество тестов в наборе ухудшается.

Но, несмотря на эти трудности, тесты E2E являются критически важными для стратегии тестирования. Они имитируют поведение настоящего пользователя, взаимодействующего с системой. Современное программное обеспечение может состоять из множества взаимосвязанных подсистем или сервисов, созданных различными командами внутри или вне организации. Организации полагаются на эти внешние системы вместо того, чтобы тратить собственные ресурсы на их создание (что, кстати, сопряжено с еще большим риском). Системные администраторы часто отвечают за эти граничные области, где системы должны взаимодействовать между собой.

Идентификация и чтение тестов в вашем продукте сервисов помогут вам определить инструменты или шаблоны, которые позволят устранить ручные процессы при тестировании инфраструктуры для этих услуг.

Четкая стратегия тестирования

Один из способов, позволяющих описать стратегию тестирования в отрасли, — это пирамида тестирования. В 2009 году Майк Кон (Mike Cohn) ввел термин «*пирамида тестирования*» (**https://oreil.ly/jmisr**) в своей книге «Succeeding with Agile» (Addison-Wesley Professional). Эта пирамида служит визуальным представлением того, как следует продумывать и планировать стратегию тестирования системы.

Как демонстрирует рис. 7.1, пирамида подчеркивает важность различных типов тестов, показывая при этом, что тесты имеют различное время выполнения и стоимость. По мере продвижения вниз по уровням тесты выполняются быстрее, поскольку уменьшается их объем и сложность.

Рис. 7.1. Концепция пирамиды тестирования

Хорошее правило для написания тестов — продвигать их как можно дальше вниз по стеку. Чем ниже тест в пирамиде, тем быстрее он будет выполняться и тем быстрее предоставит обратную связь, информируя о качестве и рисках программного обеспечения. Это связано с тем, что модульные тесты находятся ближе к коду, тестирующему конкретные функции. Сквозные тесты, наоборот, моделируют поведение конечного пользователя. Так что форма пирамиды зависит от того, сколько внимания и времени вы уделяете написанию конкретного теста.

Вы можете изучить тесты для проекта и определить стратегию на основе количества и типа тестов, чтобы получить представление об областях, где необходимо добавить или исключить тесты.

Представьте, что ваши тесты — это строительные блоки. Например, модульный тест — это блок размером 1×1. Интеграционный тест служит для проверки нескольких компонентов, его размер может меняться — например, от 1 × 2 (для тестирования двух компонентов) до более крупного. Сквозные тесты будут отличаться по размеру, но вам не обязательно вникать в тонкости каждого компонента, особенно если вы проверяли его раньше, в более быстром тесте.

На рис. 7.2 показано, как можно визуализировать стратегию тестирования вашего проекта в зависимости от количества и типа тестов. Слева показан здравый подход.

Вы запускаете преимущественно модульные тесты, немного интеграционных тестов, связывающих компоненты, и совсем немного сквозных тестов. Справа — неоптимальная ситуация, когда имеется много слишком специфических сквозных тестов, на выполнение которых потребуется больше времени.

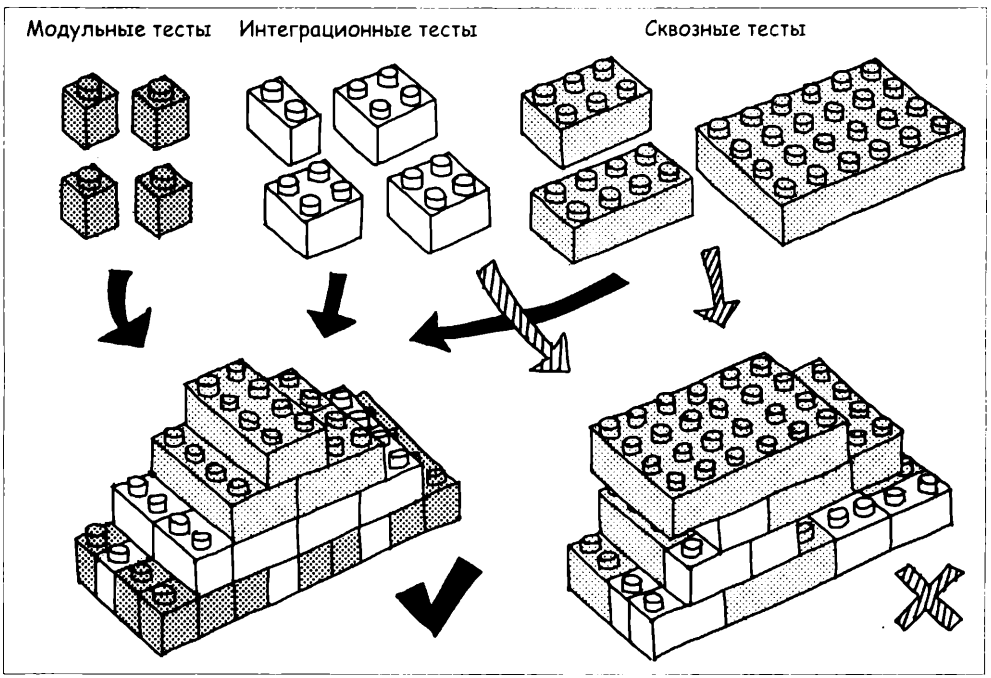

Рис. 7.2. Оценка автоматизированного тестирования

Давайте смоделируем несколько различных реализаций тестирования. Будем добавлять и удалять тесты, разберемся, что может пойти не так и какие шаги нужно предпринять для решения проблемы. Изучая реализацию тестирования, вы поймете, сколько невидимой работы перекладывается на вашу команду.

На рис. 7.3 мы видим примерно одинаковое количество тестов на каждом уровне. Это говорит о том, что в тестировании присутствует дублирование, — другими словами, вы тестируете один и тот же компонент на разных уровнях. В результате увеличивается время тестирования и задерживается сдача в эксплуатацию. Определите, где происходит дублирование, и сократите соответствующие области тестирования в рамках цикла сквозного тестирования.

На рис. 7.4 мы видим больше сквозных тестов и меньше модульных тестов, что указывает на недостаточный объем тестирования на нижних уровнях, где находятся модульные и интеграционные тесты. Это означает увеличение времени тестирования и задержку интеграции кода, поскольку для проверки надлежащей работы кода потребуется больше времени. Однако увеличение объема, протестированного при помощи юнит-тестов кода, повысит уверенность в успешности внесенных в код изменений и сократит время, необходимое для слияния кода, что приведет к уменьшению количества конфликтов!

Рис. 7.3. Структура тестов в виде квадрата

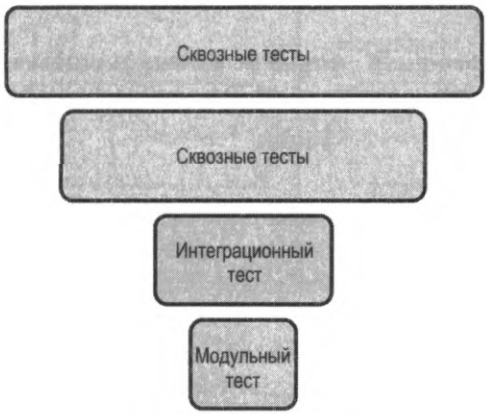

Рис. 7.4. Структура тестов в виде перевернутой пирамиды

На рис. 7.5 мы видим полное покрытие кода тестами, но сквозных тестов больше, чем интеграционных, а значит, последние не охватывают код в достаточной мере. Кроме того, сквозных тестов может оказаться больше, чем нужно. А они являются более нестабильными, требующими большего внимания и обслуживания при изменениях.

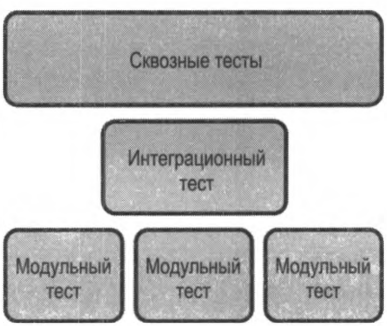

Рис. 7.5. Структура тестов в виде песочных часов

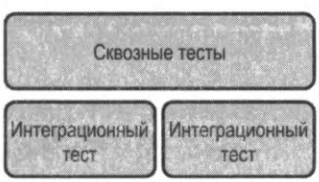

Рис. 7.6. Модифицированная пирамида тестирования для инфракода

Эти разные стратегии могут также указывать на то, что команда тратит больше времени на поддержание тестов, чем на разработку новых функций. Однако тестирование кода инфраструктуры не всегда выполняется по этим шаблонам. В частности, тестирование кода конфигурации инфраструктуры не выигрывает от модульных тестов, за исключением случаев, когда имеются различные пути конфигурации. Например, модульный тест полезен, когда существуют различия в требованиях к платформе из-за поддержки разных операционных систем. Он также может быть полезен, когда имеются различия в средах разработки, тестирования и производства или когда вам нужно убедиться, что API-ключи для рабочей среды не будут развернуты в средах разработки и тестирования.

Как видно из рис. 7.6, из-за природы инфраструктурного кода интеграционные тесты можно продвигать вниз по стеку на такую глубину, где это имеет смысл.

Совершенствуем тесты, извлекая уроки из их падения

Создавать тесты просто. Сложно создавать правильные тесты. Но это умение можно развить, как и любой другой навык. Чтобы повысить качество своих тестов, размышляйте о падениях тестов и учитесь на этих ошибках. Успешное прохождение тестов говорит о том, что проблема не обнаружена — во всяком случае, пока не обнаружена. Давайте же обсудим, как использовать обратную связь, полученную в результате тестирования.

Чтобы оценивать и внедрять автоматизированные тесты, необходимо понять причины падения тестов. Неудачные тесты расскажут вам больше, чем сообщение об обнаружении проблемы в коде. Изучение этих причин и получение разнообразной обратной связи позволяет составить дорожную карту и по возможности автоматизировать ответы.

Об этом полезно думать в процессе создания и обновления автоматических тестов. Автоматизация без использования обратной связи при тестировании добавляет работы, препятствует достижению целей, выгодных для ваших заказчиков и команды. Вы можете планировать оценку различных результатов тестов и создавать механизмы контроля за тем, что можно автоматизировать, а что требует вмешательства человека.

Подготовьтесь к возможным падениям тестов. Возможны четыре основных неблагоприятных исхода.

Проблемы, связанные с инфраструктурой

Это наиболее вероятные и быстро решаемые проблемы. Проблемы с инфраструктурой могут быть связаны с разрешением на доступ к файлам, подключением к сети, аппаратным обеспечением или различиями между тестовой и реальной средой.

Ошибочная логика теста

Эта проблема возникает, когда тест проверяет код некорректно, будь то из-за изменения спецификаций или изначального недопонимания назначения кода.

Изменение предположений

Может быть, проблемы при реализации теста возникли из-за того, что вы неверно представляли себе принцип работы определенного узла? Например, вы меняете время выполнения тестов, и это вдруг приводит к их падению, хотя в коде не произошло никаких изменений.

Дефекты кода

Это, как правило, наименее распространенная причина падения тестов, но обнаружить и устранить ее оказывается сложнее всего. Если вы считаете, что исключили другие вероятные причины, возможно, пора запускать отладчик и искать проблемы в коде.

Случай из практики: анализ падения тестов

Автор — Крис Деверс (Chris Devers)

Моя команда обнаружила, что наша тестовая среда сообщала об отключенных веб-сервисах при запуске инструмента развертывания. Однако было непонятно, почему вдруг появлялась эта ошибка — мы не меняли код для настройки веб-сервисов.

В результате расследования мы обнаружили, что скрипт настройки более широкой системы без проверки каких-либо условий запускал сценарий настройки веб-сервиса, за которым следовал запуск второго инструмента для настройки других сервисов. Второй инструмент повторно включил веб-сервис, маскируя ошибку в сценарии настройки самого веб-сервиса. К сожалению, когда мы переделали сценарий настройки системы и изменили порядок шагов, веб-сервис был выключен своим сценарием настройки и никогда не перезапускался.

Это демонстрирует различные варианты падений тестов. Исправление сценария настройки более широкой системы создало проблему инфраструктуры, которой раньше не существовало. Обоснование устранения зависимости выглядело логично, но мы предположили, что отдельные шаги были идемпотентными. В конечном итоге мы отследили дефект до сценария настройки веб-сервиса, хотя могли бы обнаружить его раньше с помощью модульных тестов. С другой стороны, даже без модульных тестов среда интеграционного тестирования обнаружила проблему до того, как ее заметил клиент, поэтому такой подход к многоуровневому тестированию все равно оказался выигрышным.

Легко сказать, что дефекты кода являются причиной падения тестов в уже существующем проекте, но их обнаружение может оказаться самой дорогостоящей задачей. Вместо того чтобы тратить время на изменение кода, исключите проблемы с инфраструктурой, реализацией тестов или изменениями в предположениях.

Дальнейшие шаги

Я не смогла охватить в этой главе все варианты тестирования, например стресс-тестирование, тестирование на производительность, комплаенс, долговечность, безопасность, проникновение и емкость. И вы можете постоянно совершенствовать код. Однако вам может понадобиться принять другую стратегию тестирования. Например, долговечные монолитные базы данных могут быть склонны к трудно выявляемым ошибкам выделения ресурсов из-за утечек памяти. К сожалению, эти проблемы трудно увидеть в ходе коротких тестов, из-за чего ваша реальная инфраструктура может быть подвержена перебоям в обслуживании. В этом случае имеет смысл запустить моделирование рабочей нагрузки в тестовой среде в течение нескольких дней или даже недель с тем, чтобы обнаружить проблемы до того, как они возникнут в инфраструктуре.

Если ваша производственная среда является эфемерной, т. е. ее ресурсы недолговечны и регулярно перезапускаются, вам не нужно тратить деньги на тестирование в долговременной среде. По мере развития навыков тестирования вы научитесь понимать качество ваших систем и видеть потенциальные уязвимости. Существует множество ресурсов, посвященных отдельным видам тестирования. Обращайтесь к ним по мере необходимости для решения вопросов, обусловленных конкретными обстоятельствами.

Наконец, вам не нужно начинать с выстраивания идеальной стратегии тестирования. Вместо этого совершенствуйте подходы по мере выявления конкретных проблем в ваших средах.

Заключение

Тестирование — это один из способов узнать о своих системах, поработать с чем-то новым и убедиться, что изменения будут работать так, как ожидается. При тестировании систем используйте модель пирамиды тестирования, чтобы организовать свои усилия по тестированию на основе модульных, интеграционных и сквозных тестов.

Среды тестирования, в которых много высокоуровневых сквозных тестов, трудоемки и затрудняют выявление конкретных причин сбоев. С другой стороны, набор тестов, в котором преимущественное внимание уделяется модульным тестам, хорошо поддается автоматизации, обеспечивает четкую и быструю обратную связь и легко расширяется.

При написании тестов для проверки вашего кода определите, как вы будете оценивать их, чтобы понимать, какие из них являются подходящими. Прохождение теста может говорить о том, что изменение кода не привело к появлению ошибок, а может означать, что тест был недостаточно тщательным. Неудачные тесты могут иметь множество корневых причин. Это и внешние факторы среды, порождающие ошибки, и недостатки самих тестов, и устаревшие предположения, вызывающие ошибки в коде, который раньше работал. Следует оценить и устранить эти причины, прежде чем делать выводы о наличии дефекта в коде.

Безопасность инфраструктуры

В начале своей карьеры, будучи системным администратором Unix, я испытывала настоящий ужас при виде множества неудачных попыток входа в систему с внешних IP-адресов за пределами США, потому что у нас было всего два человека, отвечающих за безопасность. Они занимались всем на свете — от физической сети до предотвращения вторжений в наши Unix-системы. Увидев эти сбои, я задумалась о других видах вредоносной активности, которые мы не обнаруживали. Обсуждая эти проблемы с сотрудниками, отвечающими за безопасность, я стала лучше понимать возможные риски, мотивы злоумышленников, узнала о моделях их поведения и имеющихся ресурсах, а также поняла, как выстраивать отношения между группами.

Вы не в силах гарантировать идеальную защиту, но, сотрудничая с другими отделами в организации, можете обеспечить приемлемый уровень безопасности. Всю работу по обеспечению безопасности, которую необходимо выполнить каждой организации для достижения «приемлемого уровня защиты», невозможно поручить одной команде, особенно с учетом того, что атаки становятся всё более массовыми, а их обнаружение и блокировка — всё более дорогостоящими. В этой главе я сосредоточусь на общих принципах безопасности, чтобы вы понимали ее сущность, объясню, что такое моделирование угроз и рассмотрю несколько методов обеспечения безопасности при планировании архитектуры.

Что такое безопасность инфраструктуры?

Безопасность инфраструктуры предусматривает защиту оборудования, программного обеспечения, сетей и данных от повреждения, кражи или несанкционированного доступа. К сожалению, многие полагают, что меры по обеспечению безопасности ухудшают функционал и снижают удобства для пользователя, что может провоцировать нежелание их внедрения, даже если конечная цель этих мер — снизить риски для людей.

В число уязвимостей могут входить риски для ваших сетей, физических и виртуальных машин, приложений или хранимых и обрабатываемых данных. Когда срабатывает оповещение, меньше всего вы хотите обнаружить неработающие системы, поврежденные данные или взломанные веб-сайты. Как же повысить безопасность ваших систем и сервисов?

Решайте проблемы безопасности так же, как и другие сложные проблемы. Разбейте эту огромную задачу на небольшие выполнимые задачи, над которыми команда

постоянно работает. Используйте обратную связь и обучение на ее основе, чтобы информировать команду и изменять методы ее работы в сотрудничестве с программистами и экспертами по безопасности.

Инциденты, связанные с безопасностью, — это лишь вопрос времени. Они неизбежно случаются, наносят финансовый ущерб компаниям и снижают доверие пользователей. Принципиально невозможно создать абсолютно безопасное приложение или сервис или управлять ими, когда могут иметь проблемы с безопасностью базовые библиотеки, операционные системы и сетевые протоколы. Независимо от того, какое программное обеспечение вы создаете или развертываете — с открытым исходным кодом или коммерческое, планируйте комплексную стратегию, чтобы минимизировать количество уязвимостей и уменьшить возможности злоумышленников по их использованию.

Распределяйте обязанности по обеспечению безопасности

Выбирая размещаемые вычисления, вы также разделяете ответственность за безопасность. Чем больше операционной нагрузки вы передаете облачному провайдеру, тем больше вопросов безопасности вы с себя снимаете. Например, облачный провайдер, предлагающий выделенные серверы, приобретает оборудование, подключает его к сети, предоставляя вам доступ к нему, и управляет физическим доступом к серверу.

Давайте рассмотрим вычислительные среды из *главы 2* с точки зрения безопасности.

Если вы управляете выделенным физическим оборудованием (нижняя часть рис. 8.1), то отвечаете за все, что перечислено в колонке «Обязанности по обеспечению безопасности». По мере движения к верхней части рисунка видно, что поставщик услуг берет на себя долю этих обязанностей.

Команда, отвечающая за безопасность, выполняет различные функции. Она не обязательно несет все обязанности по обеспечению безопасности организации. Вы не должны выполнять работу по обеспечению безопасности, не изучив этот вопрос, особенно если являетесь единственным системным администратором, отвечающим за обслуживание систем. Это путь к выгоранию. Вместо этого объясните членам команды и руководству, какие мероприятия следует выполнить, чтобы они могли оценить их и расставить приоритеты.

Независимо от того, используете ли вы предоставляемую поставщиком инфраструктуру как услугу (IaaS), платформу как услугу (PaaS) или функцию как услугу (FaaS), вы все равно несете ответственность за часть мер по обеспечению своей безопасности. Вы можете предположить, что ваш поставщик услуг сам позаботится о безопасности всей инфраструктуры. Но как минимум вы должны настроить управление учетными записями и доступом, указать и настроить конечные точки, а также управлять данными. Например, не имеет значения, шифрует ли ваш поставщик облачных решений все данные на диске, если вы предоставляете глобальный общий доступ к нему.

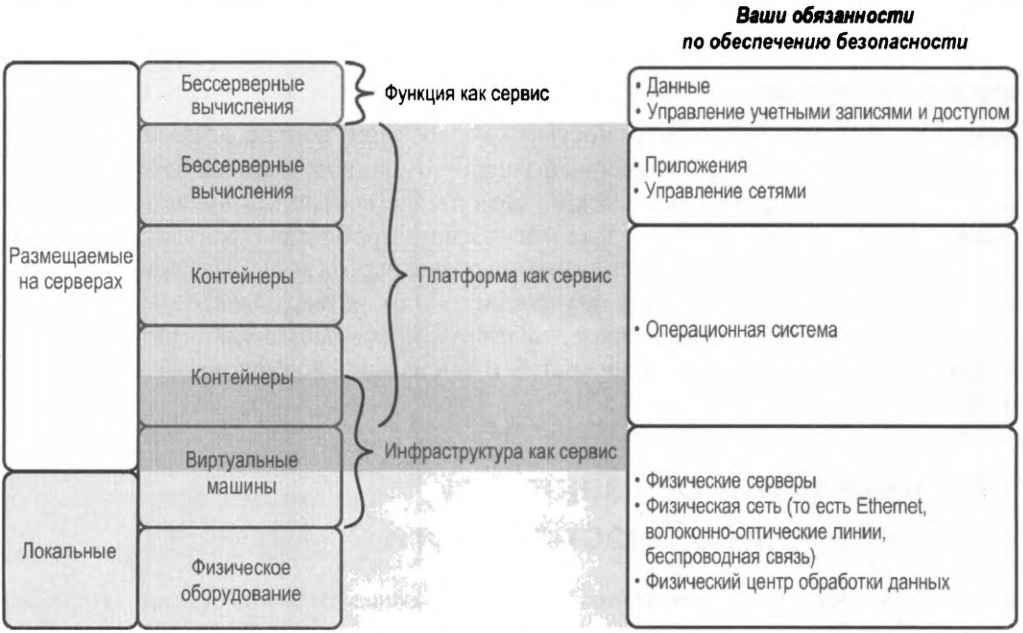

Рис. 8.1. Обязанности по обеспечению безопасности для различных вычислительных сред

Выясняйте у всех поставщиков услуг, как они обеспечивают безопасность, чтобы понимать свою зону ответственности. Если кто-то воспользуется уязвимостью в системе безопасности вашего поставщика и вы скажете своим клиентам, что это его вина, то потеряете их доверие и понесете финансовый ущерб от компрометации данных. Выясните по меньшей мере, как поставщик обрабатывает уведомления и каков соответствующий маршрут эскалации для обнаружения событий безопасности.

Оценивайте безопасность с точки зрения злоумышленника

Взгляд на системы, которыми вы управляете, с точки зрения злоумышленника, поможет повысить их безопасность.

Моделирование угроз — это процесс, с помощью которого вы определяете, анализируете и документируете потенциальные угрозы, представляющие опасность для ресурсов вашей организации, таких как физическое оборудование, программное обеспечение и данные. Моделирование способствует созданию более защищенных систем. К сожалению, ресурсы не всегда оказываются изучены в достаточной мере, особенно если вы сами не разрабатывали и не внедряли определенную систему или сервис. Иногда в процессе моделирования угроз обнаруживаются подробные данные, которые увеличивают риски для вашей организации, хотя и не представляют собой большой ценности. Поэтому они являются главными кандидатами на удаление, что позволяет снизить затраты в целом.

Далее рассмотрим различные векторы атак или направления атак. *Направления атак* — это все потенциальные точки входа при вторжении в определенные ресурсы вашей организации. Например, проверьте уязвимости всех конечных точек, подключений к базам данных и сетевого транспорта.

Инструменты моделирования угроз

Существуют различные инструменты моделирования угроз, которые помогут выявить и исследовать проблемы в ваших системах:

- NIST Common Vulnerability Scoring System Calculator (**https://oreil.ly/MWlE6**);
- Threat Modeling Tool от Microsoft (**https://oreil.ly/lT8Ml**);
- Процесс эмуляции атаки анализа угроз (Process for Attack Simulation and Threat Analysis, PASTA)[1];
- OWASP Threat Modeling Control Cheat Sheet (**https://oreil.ly/QS72x**).

Используйте инструмент моделирования угроз, чтобы лучше понимать уязвимости и области, требующие улучшения. Не существует единственно верного способа или инструмента при выполнении этой работы. Учитесь, инициируя необходимые дискуссии на соответствующих интернет-ресурсах.

Задайте себе такие вопросы.

Кем являются ваши злоумышленники?

Злоумышленником может оказаться кто угодно. Это могут быть люди внутри или вне вашей организации. Разбор тысяч инцидентов, связанных с безопасностью, анализируемых в ежегодном отчете Verizon Data Breach Investigations Report (DBIR) (**https://oreil.ly/pWfI7**), показывает, что большинство атак осуществляются извне. Изредка в роли злоумышленников выступают сами системные администраторы, но чаще всего проблемы внутренней безопасности возникают из-за ошибок в конфигурации системы или когда конфиденциальная информация попадает в открытый доступ. В *главе 11* я расскажу о некоторых инструментах и технологиях, помогающих снизить количество ошибок, приводящих к инцидентам в области внутренней безопасности.

Каковы их мотивы и цели?

Мотивы и цели злоумышленников разнообразны. Основные мотивы можно разделить на несколько групп:

- Развлечение. Некоторые атаки совершаются просто так, в качестве развлечения.
- Личные убеждения. Нарушители из числа штатных сотрудников могут преследовать личные цели, противоречащие ценностям и интересам организации.

[1] Ознакомьтесь с презентацией (**https://oreil.ly/GgZfR**) сообщества OWASP (Open Web Application Security Project) по моделированию угроз при атаках на банки с использованием вредоносного программного обеспечения, в которой был представлен процесс PASTA.

- Идеология. Работники, имеющие социальные или политические идеологические разногласия с организацией, могут стремиться навредить ее репутации или нарушить обслуживание клиентов.
- Месть. Вред может причинить инсайдер, затаивший злобу на вашу организацию.
- Финансовая выгода. В некоторых атаках персональные данные продаются кампаниям, рассылающим спам, используются для подачи заявок на кредитные карты, мошенничества с платежными картами или получения доступа к ресурсам и услугам человека.
- Шпионаж. Растущей угрозой становятся атаки на национальном уровне. Известны многочисленные случаи взломов (**https://oreil.ly/BVku1**) с целью получения разведданных о государственных секретах, интеллектуальной собственности и влияния на политику.

Какими ресурсами для атак они располагают?

Ресурсы злоумышленника включают время, деньги, инфраструктуру и навыки. Появляются инструменты, которые уменьшают объем знаний, необходимых отдельному злоумышленнику для доступа к целевым активам. Не все атаки можно предотвратить, но можно сделать каждую из них более дорогостоящей для атакующего.

Какие возможности для атак у них имеются?

Возможности — это окна, через которые можно получить доступ к определенному ресурсу. Например, когда обнаруживаются уязвимости или ошибки в программном обеспечении и выпускается соответствующее обновление, появляется окно времени для использования этих уязвимостей в системах и сервисах, где обновление еще не установлено. Чтобы избежать возможных атак, необходимо знать о том, какие обновления выпускаются, и иметь достаточно времени и полномочий для их установки.

В некоторых случаях могут существовать активы, не входящие в сферу вашей ответственности, которые злоумышленники используют для проникновения в производственные системы. Минимизируйте эти возможности путем отслеживания всех активов и своевременной установки обновлений для ОС и программного обеспечения.

Посмотрите доклад Яна Колдуотера (Ian Coldwater) на конференции KubeCon + CloudNativeCon 2019 «Hello from the Other Side: Dispatches from a Kubernetes Attacker» (**https://oreil.ly/KsOSO**), чтобы узнать больше о том, чему можно научиться, если оценивать риски с точки зрения злоумышленника.

Еще одним хорошим ресурсом является ежегодный отчет Verizon Data Breach Investigations Report (DBIR) (**https://oreil.ly/g77MX**), в котором анализируются тысячи инцидентов и нарушений и дается представление о тенденциях в области обеспечения безопасности.

Проектирование с учетом обеспечения безопасности

Выстраивайте свои стратегии с учетом снижения риска для сервисов и приложений, ограничивая возможности злоумышленника и масштабы ущерба от возможного взлома. Этот подход известен как глубокая защита. Многоуровневая защита означает, что даже если один уровень защиты в вашей системе будет нарушен, вы все же сможете ограничить радиус поражения. Например, создайте защиту на границах сетей с помощью межсетевых экранов и настройте подсети, разрешив трафик только из доверенных сетей. В локальных системах заблокируйте учетные записи с повышенными привилегиями. Кроме того, помните, что не существует программного обеспечения, защищенного на 100%, и возьмите на вооружение модель «нулевого доверия». Эта модель подразумевает, что нельзя полностью доверять никаким сервисам, системам или сетям, даже если вы используете облачные сервисы.

Участие в проектировании архитектуры и дизайна на ранних стадиях с учетом требований по безопасности позволит вам сразу же получать данные, касающиеся архитектуры системы и снизит вероятность последующего рефакторинга для обеспечения безопасности. Расскажу один случай. Однажды я присоединилась к относительно новой команде, создающей многопользовательский сервис для внутренней аудитории. Я просмотрела архитектуру и поняла, что программный код не предполагает использования пароля пользователя `root` в MySQL. Поскольку для этого сервиса планировалось использовать сотни внутренних серверов MySQL, меня беспокоило наличие множества незащищенных сервисов.

Вот некоторые из возможных векторов атак:

- неправильно настроенная подсеть может сделать эти серверы доступными напрямую из Интернета;
- злоумышленники, проникшие в системы внутренней сети, могут легко взломать незащищенные системы.

Общаясь с командой инженеров по безопасности, я смогла доказать важность работ по устранению этого проектного дефекта. Выявление проблемы до этапа производственного развертывания было очень полезным. Однако если бы мы более тесно сотрудничали с самого начала работы над проектом, то могли бы избежать затрат на устранение уязвимостей, причиной которых стала полностью открытая база данных.

Часто лица, принимающие решения по продуктам, забывают пригласить системных администраторов на совещания по проектированию. Поддерживая и налаживая отношения с людьми, разрабатывающими и создающими программное обеспечение, вы можете повысить вероятность получения таких приглашений. Доступ к информации на ранних стадиях проектирования означает, что вы можете влиять на принятие решений о работе системы еще до ее создания. А когда вам понадобится внедрить изменения, у вас уже будут налажены необходимые связи, что даст вам возможность обозначить приоритетные задачи.

Одним из способов совместного решения задач в части соответствия требованиям безопасности и определения приоритетов в работе является «триада КЦД» — конфиденциальность, целостность, доступность (от *англ.* CIA — confidentiality, integrity, availability). Эта модель позволяет установить общий контекст и выработать единый подход для работы над функциями.

Конфиденциальность

Набор правил, гарантирующий попадание конфиденциальной информации только тем людям, которые должны иметь к ней доступ.

Целостность

Гарантия того, что информация является достоверной и точной и что изменять ее может только лицо, обладающее соответствующими правами доступа.

Доступность

Надежный и беспрепятственный доступ к информации и ресурсам для пользователей с соответствующими правами доступа.

В случае с паролем пользователя `root` для MySQL, о котором я рассказывала ранее, любой человек, имеющий доступ, смог бы войти в систему управления базами данных (нарушена конфиденциальность) и просмотреть и отредактировать любые хранящиеся данные (нарушена целостность). Сисадмины могут рассматривать соответствие правилам CIA как часть условий приемки программного обеспечения и руководствоваться ими при соответствующей расстановке приоритетов. Целенаправленное обсуждение аспектов проектирования и периодическое возвращение к этим обсуждениям помогает командам разработчиков и командам по продуктам в принятии решений.

Открытый проект обеспечения безопасности веб-приложений (Open Web Application Security Project, OWASP) (**https://oreil.ly/CuLHL**) предоставляет набор требований и элементов управления для проектирования, разработки и тестирования веб-приложений и веб-сервисов под названием «Стандарт подтверждения безопасности приложений» (Application Security Verification Standard, ASVS) (**https://oreil.ly/rZ7gf**).

Если вам сложно получить поддержку руководства в ваших усилиях по разработке и внедрению качественных механизмов непрерывной интеграции и развертывания, можете обосновать необходимость этих мер уменьшением количества уязвимостей системы безопасности.

Классификация обнаруженных проблем

Независимо от того, сколько усилий прилагает команда, чтобы изучить программное обеспечение и сервисы с точки зрения злоумышленника и разработать системы с учетом принципов безопасности, все равно будут возникать проблемы. Могут обнаружиться проблемы в программном обеспечении вашей компании или программном обеспечении, которое вы используете прямо или косвенно. Обычно организации отслеживают уязвимости в официально опубликованных программ-

ных пакетах с помощью идентификаторов общеизвестных уязвимостей информационной безопасности (Common Vulnerabilities and Exposures, CVE, **https://oreil.ly/UrrSp).**

При количественной оценке финансовых затрат и возможных последствий, связанных с обнаруженной проблемой, классифицируйте проблемы с помощью маркировки (например, ошибка или дефект программы), чтобы передать дополнительный контекст.

Ошибки реализации — это проблемы в реализации, которые заставляют систему работать не так, как предусмотрено. Ошибки в реализации иногда могут стать причиной серьезных уязвимостей в системе безопасности, как, например, Heartbleed[2]. Heartbleed — это уязвимость в OpenSSL, которая позволяла злоумышленникам перехватывать сведения в процессе предположительно безопасного обмена информацией, красть данные непосредственно у сервисов и пользователей, а также выдавать себя за этих пользователей и сервисы.

Дефекты при проектировании — это проблемы, которые мешают системе работать так, как задумано. Проблема возникает из-за неправильного проектирования или спецификации. Устранение этих дефектов может оказаться очень дорогостоящим, особенно если другие инструменты основаны на системе в ее нынешнем виде или зависят от ее реализации. Таким образом, иногда дефекты не устраняют, т. к. это очень дорого, а выпускают специальные предупреждения об использовании.

Хотя вам, как системному администратору, не нужны показатели обнаружения дефектов и ошибок, следует оповещать о найденных уязвимостях, особенно если их устранение поможет предотвратить утечку данных или инциденты в области безопасности. Показатели предотвращения инцидентов четко показывают вашу заинтересованность и результаты вашей работы, даже если нельзя подсчитать ущерб от события, которое не случилось.

Определяя категории работ, вы начнете лучше понимать, какие направления деятельности являются наиболее подходящими и полезными. Вы укрепляете отношения и доверие, поскольку другие сотрудники получают больше информации о проделанной вами работе и узнают, какие выгоды она приносит бизнесу.

Ознакомьтесь с примерами ошибок реализации:

- MS17-010/EternalBlue;
- CVE-2016-5195/Dirty COW.

И взгляните на примеры дефектов проектирования:

- Meltdown (**https://meltdownattack.com**);
- Атака реинсталляции ключей (WPA2 key reinstallation attack, KRACK).

[2] «The Heartbeat Bug», Synopsis, Inc., последнее изменение от 3 июня 2020 г., **https://heartbleed.com**.

Заключение

Безопасность инфраструктуры — это действия по защите ваших систем от угроз. Конечно, идеальной системы безопасности не существует, но, проявляя бдительность и используя многоуровневый подход, вы можете снизить риски для своих систем.

Безопасность инфраструктуры подразумевает распределение обязанностей. Если организация еще не применяет соответствующие методы, вы можете возглавить работу по их внедрению. Хорошей стратегией в данном случае будет оценка ваших систем с точки зрения злоумышленника, чтобы понять, какие сведения они надеются получить в результате атаки и какие ресурсы могут задействовать для достижения цели. На основе этих оценок можно выстраивать линии защиты. Далее при создании и развертывании систем используйте «триаду КЦД» — модель, учитывающую такие показатели, как конфиденциальность, целостность и доступность. Это поможет определить требования по обеспечению безопасности.

Классификация вопросов безопасности инфраструктуры предоставляет дополнительный контекст для определения стоимости и возможных последствий. Например, можно использовать такие ярлыки, как «ошибки» и «дефекты проектирования». Ошибки реализации являются следствием ошибок программирования, которые обычно поддаются исправлению. Дефекты проектирования — это архитектурные проблемы, которые сложнее устранить без фундаментального перепроектирования частей системы.

ГЛАВА 9

Документация

Это были первые дни, когда я только начала работать сисадмином, и все казалось таким новым. Я чувствовала, что каждое мое действие жизненно важно для обеспечения доступности и производительности системы. Я быстро выучила все задачи системного администратора, прочитав книгу, но не получила информации о том, когда использовать некоторые команды, требующие полномочий пользователя root. К счастью, опытные сисадмины установили культуру обмена знаниями. Наш сайт с документами представлял собой wiki-страницу с темой, взятой из карточной игры Cheapass Games «Дай мне мозги» (Give Me the Brain, GMTB) (**https://oreil.ly/u2f04**).

Основная сюжетная линия игры GMTB заключалась в том, что вы были зомби, работающим в ресторане быстрого питания, и у вас был один мозг, который можно было передавать друг другу. Только один человек мог «обладать мозгом» в каждый отдельно взятый момент времени. Ассоциирование документации с этой игрой закрепило поведение, ожидаемое от команды, — особенно ту мысль, что именно документация определяет успех вашей работы, когда в два часа ночи вам придется выступать в роли зомби, устраняя какой-либо инцидент.

В этой главе я хочу помочь вам поразмышлять о документации. Я хочу, чтобы вы чувствовали себя способными внедрять методики, поддерживающие качественную документацию — точную, имеющуюся в наличии, доступную, упорядоченную и удобную в сопровождении, потому что документация — это часть системы.

Узнайте свою аудиторию

Людям нужна информация, относящаяся к их обязанностям, начиная от деталей работы конкретной системы и заканчивая обзором деятельности «с высоты птичьего полета» для понимания общего контекста.

Предоставление людям актуальной, точной и своевременной информации помогает им эффективно выполнять свои обязанности. Если повседневные действия не выполняются, это может быть связано с тем, что работники располагают устаревшими, расплывчатыми данными, которые невозможно использовать. Если члены команды сосредоточены преимущественно на решении сиюминутных проблем в ущерб долгосрочной стратегии, это может свидетельствовать о нарушении цикла обратной связи.

На рис. 9.1 автор документации пытается помочь всем, что приводит к появлению не относящейся к делу информации, которая может оказаться бесполезной для кон-

кретного пользователя. На рис. 9.2 автор документации ориентируется на пользователя и своевременно предоставляет только ту информацию, которая необходима ему для выполнения задач.

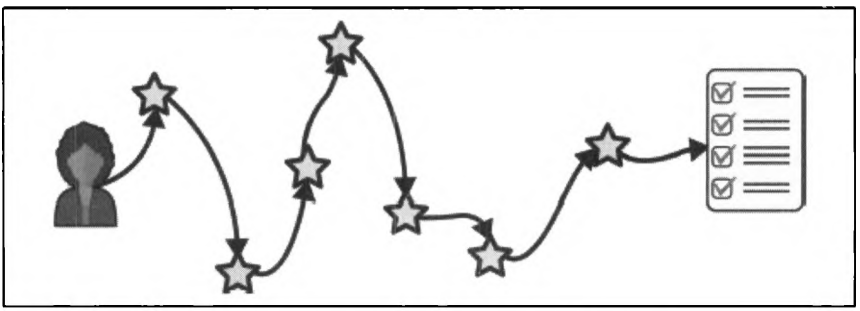

Рис. 9.1. Документация с излишней информацией

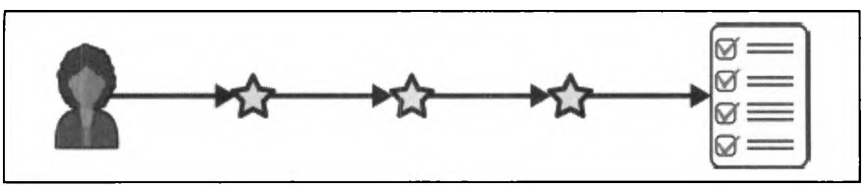

Рис. 9.2. Документация с конкретной информацией, ориентированной на пользователя

Обратите внимание, что путь с ориентацией на пользователя короче и ведет прямо к цели. Но один путь не лучше другого. Опять же, изучите свою аудиторию и ее потребности, когда разрабатываете документацию. Например, рассмотрим потребности человека, устраняющего конкретную проблему. Ему нужна документация, нацеленная на решение конкретной задачи, где представлена узкоспециализированная информация. Излишняя информация только обескураживает. Вот несколько подсказок, которые помогут вам поразмышлять и выяснить потребности аудитории.

- Кто является вашей аудиторией?
- Что важно для этих людей?
- Какими знаниями они должны обладать и какие действия вы от них ожидаете?
- Что они уже знают?
- Какой способ потребления информации они предпочитают?
- Как ваши данные подтверждают вашу точку зрения?
- Существует ли уже какая-либо документация?
- Знаете ли вы, что нужно вашей аудитории?

Аудитории нужны разные уровни детализации. Потребности людей могут меняться в зависимости от того, чем они заняты в конкретный момент. А документация,

с которой они столкнутся, выведет их на следующий этап работы и будет определять выбор следующего комплекта документации. Если это опытные работники, им может потребоваться информация непосредственно по теме. Если опыта или знаний не хватает, может понадобиться более длинное, подробное, концептуальное объяснение с предоставлением соответствующего контента. Если вы не знаете, что нужно вашей аудитории, опросите ее, чтобы определить, что именно для нее полезно.

Варианты представления документации

Вы можете разрабатывать документацию для себя, своей команды, организации в целом или более широких групп людей за пределами организации. Вы можете читать документацию онлайн (например, на мониторе или мобильном устройстве) или в распечатанном виде. Вы будете создавать различные документы, в том числе записи, документацию общего назначения, документацию по задачам, справочные руководства и проектную документацию.

Записи

Ссылка на решения, действия или обсуждения. Записи включают в себя записи встреч и протоколы решений. Это исторические материалы, которые помогают понять, почему был сделан тот или иной выбор, а также информируют о будущих решениях.

Документация общего назначения

Содержит общие сведения. Документация общего назначения представляет собой введение в тему. Используйте этот тип документации, когда читателю необходимо разобраться с терминологией, идеями и абстрактными понятиями.

Документация по задачам

Это пошаговое руководство, помогающее читателю решить конкретную задачу. Примеры документации по задачам — руководства и учебники. Используйте этот тип документации, когда читателю нужно разобраться, как что-то сделать, или понять, что происходит. Иногда вы можете использовать эту документацию для автоматизации задач.

Справочные руководства

Подробная документация, часто в виде перечня или таблицы. Примерами справочных руководств являются руководства по эксплуатации и устранению неисправностей. Используйте справочные руководства, когда вам нужно упорядочить и сгруппировать большой объем информации.

Проектная документация

Раскрывает сущность более крупных проектов. Этот тип документации часто включает в себя содержание или цель проекта и предоставляет необходимую справочную информацию, сам проект, а также ориентировочные сроки выполнения и любые необходимые ресурсы для поддержки проекта. Рассмотрение

плана проектных мероприятий с остальными членами команды помогает выявить потенциальные проблемы и области, нуждающиеся в доработке.

Наконец, эффективная документация обобщает опыт и знания и является результатом совместной работы множества людей. Для совместной работы необходимо общее понимание цели, нужны руководства по стилю, чтобы сохранялось единообразие при изложении материала, и набор процедур, которым все следуют.

Разработка и совместное использование шаблонов дают возможность быстро подготовить документацию необходимого вида с соблюдением принятого стиля.

Методы организации информации

Информационная архитектура — это структурная организация вашей информации. Качественная организация информации отвечает следующим требованиям.

Повторное использование

Вы можете повторно использовать документацию различными способами, чтобы поддерживать вовлеченность сотрудников, их уровень знаний, облегчать обучение.

Управление изменениями

Вы можете добавлять и обновлять документацию, выпускать различные версии и объявлять документацию устаревшей.

Организация управления

Документация имеет четко определенные роли и обязанности, что позволяет поддерживать ее и назначать ответственных.

Связность

Документация показывает связь между темами.

Организация тем

При структурировании темы дайте ей четкое и конкретное название. Однако заголовки не должны быть слишком общими. Вы распределяете информацию по разделам с множеством заголовков и подзаголовков. После того как вы напишете текст, просмотрите заголовки, чтобы проверить логичность изложения информации. Хорошо, когда содержание, созданное на основе заголовков, понятно вашему читателю. Введение и заключение создавайте в последнюю очередь, потому что содержание может измениться в процессе написания документации.

Темы следуют (документированному) и определенному жизненному циклу, что позволяет управлять изменениями и создает основу для организации управления. На рис. 9.3 показан возможный жизненный цикл документа, в котором вы исследуете и анализируете тему, создаете (или обновляете существующий) документ,

проверяете правильность информации, присваиваете теме номер версии и выпускаете документ. Отзывы и предложения по теме или изменения в системах продлевают жизненный цикл до тех пор, пока анализ не покажет, что пора объявить документ устаревшим, а впоследствии отправить в архив.

Рис. 9.3. Жизненный цикл документа

Организация информации на сайте

При структурировании всех тематических разделов сайта организуйте информацию так, чтобы она была доступна для обнаружения и располагалась в логической последовательности. Сайт должен предоставлять информацию о принятой тематической структуре. Упорядоченное расположение тематических разделов помогает читателю понять, как найти необходимую информацию, и снижает когнитивную нагрузку на него.

В зависимости от сложившейся в команде культуры, можно отдать предпочтение одному длинному документу или множеству небольших страниц для организации базы знаний команды. Не существует единственно верного пути. Загрузка одного объемного документа с удаленной системы может занимать много времени, и у пользователей некоторых браузеров могут возникнуть проблемы. Но когда документация представлена на одной странице, читатель может пробежать глазами текст и понять, что если информация не найдена, то ее нет.

Небольшие страницы загружаются быстро, но в этом случае требуется система поиска и индексирования, чтобы находить нужные сведения. Не обнаруживая нужную информацию, читатели либо тратят больше времени на поиск, либо ошибочно заключают, что она отсутствует.

Какой бы вариант ни выбрала ваша команда, будьте последовательны, чтобы пользователи знали, что ожидать от документации и когда обратиться за помощью или попросить внести дополнения в документацию.

Рекомендации по созданию качественной документации

Итак, вы понимаете важность качественной документации, но у вас складывается впечатление, что люди не читают ее и не хотят вносить свой вклад в ее разработку. Вместо того чтобы сокрушаться по этому поводу, ответьте на следующие вопросы.

- Предоставляете ли вы им время на чтение документации?
- Не используете ли вы неизвестные термины?
- Информация слишком расплывчата?
- Информация слишком разрозненна?

Вместо того чтобы сосредоточиваться на том, что люди не создают или не обновляют документацию, задумайтесь над такими вопросами.

- Предоставляется ли им достаточно времени на написание документации?
- Нужно ли создавать документацию?
- Предоставляется ли им достаточно времени на организацию документации?
- Известна ли им аудитория и ее потребности?
- Позволяют ли инструменты вносить вклад в существующий рабочий процесс?

Затем оцените свою документацию по следующим критериям, чтобы выявить области, где возможны улучшения.

Точность

Точная документация должна быть актуальной, полной и правильной. Она содержит сведения, необходимые читателю для оперативного выполнения своей задачи. Она повышает вероятность успешного выполнения работы и снижает риск ошибок.

Наличие

Документация считается имеющейся в наличии, если к ней можно обратиться, когда возникает необходимость. Поэтому обязательно распечатайте все соответствующие инструкции по действиям в аварийных ситуациях, которые помогут вам восстановить работоспособность систем — на тот случай, если из-за сбоя в локальной сети или отключения электроэнергии доступ к электронным руководствам будет ограничен.

Доступность

Доступная документация инклюзивна, т. е. каждый работник должен быть в состоянии воспользоваться документацией на своем рабочем месте. Вам может понадобиться представить ее несколькими способами, в зависимости от уровня читателя

Упорядоченность

Упорядоченная документация позволяет читателям найти нужный им документ.

Удобство сопровождения

Удобная для сопровождения документация позволяет разработчику успешно добавлять, обновлять и удалять документы. Хранение документации с применением системы контроля версий обеспечивает управление изменениями и их отслеживаемость, а также позволяет использовать рабочие процессы, которые уже используются в вашей организации.

Регулярно просматривайте документацию, чтобы оценить ее качество в соответствии с указанными критериями. Официально закрепите правила обращения с документацией, чтобы они стали регулярной частью процессов команды.

Заключение

Эффективная документация требует знания своей аудитории, понимания того, в каком виде должна быть представлена документация (например, онлайн- или печатные документы, а также записи, документация общего назначения, документация по задачам, справочные руководства). Необходимо упорядочить документацию для повторного использования, управления изменениями, организации управления и связности.

В следующей главе я более подробно раскрою некоторые понятия из этой главы для лучшего понимания и представления информации в различных форматах.

Дополнительные ресурсы

Джаред Бхатти (Jared Bhatti) и др. в книге «Docs for Developers:An Engineer's Field Guide to Technical Writing» (Apress) подробно рассматривают методы для всех этапов жизненного цикла документации, применяемые при разработке программного обеспечения, но они подходят и для сисадминов, управляющих системами.

В книге «Living Documentation: Continuous Knowledge Sharing by Design» (Addison-Wesley Professional) Сирил Мартрейр (Cyrille Martraire) предлагает концепцию Documentation 2.0 и объясняет, как использовать качественно подготовленные компоненты и автоматизацию для улучшения документации.

Презентации

Истории — это основной способ организации и осмысления человеком информации. Истории упорядочивают данные и показывают, для чего они нужны. Квалифицированные системные администраторы признают силу хорошего нарратива и используют различные способы, чтобы эффективно делиться информацией. Они упорядочивают имеющиеся сведения и общаются со своей аудиторией не только посредством текста, но и рассказывая истории — с помощью изображений, фотографий, графиков, диаграмм, аудио- и даже видеоматериалов. Поэтому часто, обучая других сисадминов, которые пытаются что-то изменить в своей организации, я делюсь некоторыми ключевыми понятиями об упорядочении и представлении данных.

Покажите пяти умным людям одни и те же данные, и они представят десять вариантов их интерпретации. Вы не можете знать наверняка, что другие придут к тем же выводам, что и вы, если не приложите усилий для создания повествования, которое будет направлять людей и оказывать на них влияние. В этой главе я хочу научить вас извлекать информацию и убедительно представлять ее, рассказывая подходящие истории, чтобы воздействовать на людей независимо от того, какие должности они занимают.

Узнайте свою аудиторию

Как и при написании документации, при подготовке и представлении информации необходимо оценить свою аудиторию, чтобы создать конкретные, предназначенные именно для нее визуальные материалы.

В фильмах главный герой часто может решить, что делать дальше, выполнив один запрос или взглянув на приборную панель, объединяющую все нужные данные. Одного взгляда на цифровой дисплей с информацией оказывается достаточно, чтобы получить нужный контекст для быстрого принятия решения.

В реальном мире, хотя руководство может потребовать от вас создания единой информационной панели, куда будут поступать данные со сложных систем, и которая предоставит весь контекст, необходимый для принятия решений, единое представление всех данных невозможно. Конечно, можно выгрузить все возможные параметры в одну консоль, но когнитивная перегрузка не позволит эффективно обращаться к той нужной и актуальной информации, которая требуется вам в данный момент времени.

Если ваш руководитель заводит речь о единой информационной панели, выясните, на какой вопрос он пытается ответить или какую проблему решает, и предоставьте ему подходящий набор визуализаций. Можно предоставить каждому человеку индивидуальную информационную панель и иметь свою собственную, которая отражает необходимый вам контекст.

Вы боретесь за внимание аудитории, за то, чтобы она доверяла тем данным, на основании которых вы делаете свои выводы, поэтому обязательно ознакомьтесь с рекомендациями, касающимися аудитории, представленными в *главе 9*. Ни один график и ни одна информационная панель не смогут обобщить информацию таким образом, чтобы она была в равной степени полезна для каждого. Поэтому лучше адаптируйте их к потребностям конкретной аудитории.

Я сидела на очередном совещании, когда коллега пытался донести до меня важность нового проекта. Он зачитывал фразы прямо со слайдов, описывая утомительные работы по техническому обслуживанию, которые могли бы сэкономить уже потраченные деньги. К сожалению, большие цифры, полученные в результате его исследований, не развеяли скуку и не заставили меня принять участие в дополнительной работе по достижению целей его проекта. Было непонятно, почему данная работа должна быть самой приоритетной для команды, и есть ли у нас нужные инструменты для выполнения этих рутинных мероприятий.

Не тяните резину

Вместо этого, когда вам нужна помощь, объясните людям, о чем вы просите, чтобы они понимали контекст, слушая вашу презентацию. Например, сисадмин и автор книг Томас Лимончелли (Thomas Limoncelli) (**https://oreil.ly/UcJF8**) предлагает следующие варианты некоторых своих вводных предложений:

- Я здесь, чтобы попросить о финансировании [или ресурсах, или деньгах].
- Я здесь, чтобы попросить о принятии важного решения.
- Я здесь, чтобы спросить совета [о том, как что-то сделать или с кем поговорить].
- Я здесь, чтобы сообщить о состоянии дел.

Руководители должны отчитываться перед заинтересованными сторонами, и у них напряженный график работы. Они также могут использовать ограниченное количество рычагов для контроля результатов: могут предоставить ресурсы, разъяснить политику и направить вас к различным ресурсам.

Этот случай напоминает мне цитату Марка Твена: «Часто самый верный способ ввести человека в заблуждение — сказать ему чистую правду». Недостаточно представить людям сухие факты и таким образом побудить их к действию. Вы должны показать, почему нужно обращать внимание на эти факты и как они связаны с более важными целями, а затем создать эмоциональную связь, чтобы люди захотели помочь вам в вашем деле.

Как показано на рис. 10.1 слева, если у вас на экране много текста, ваша аудитория будет читать его и, возможно, отключится. Если вы представите данные в виде диаграмм, как показано на рисунке справа, ваши коллеги с большей вероятностью будут вдохновлены вашими идеями и мотивированы на скорейшее принятие тех действий, которые вам требуются.

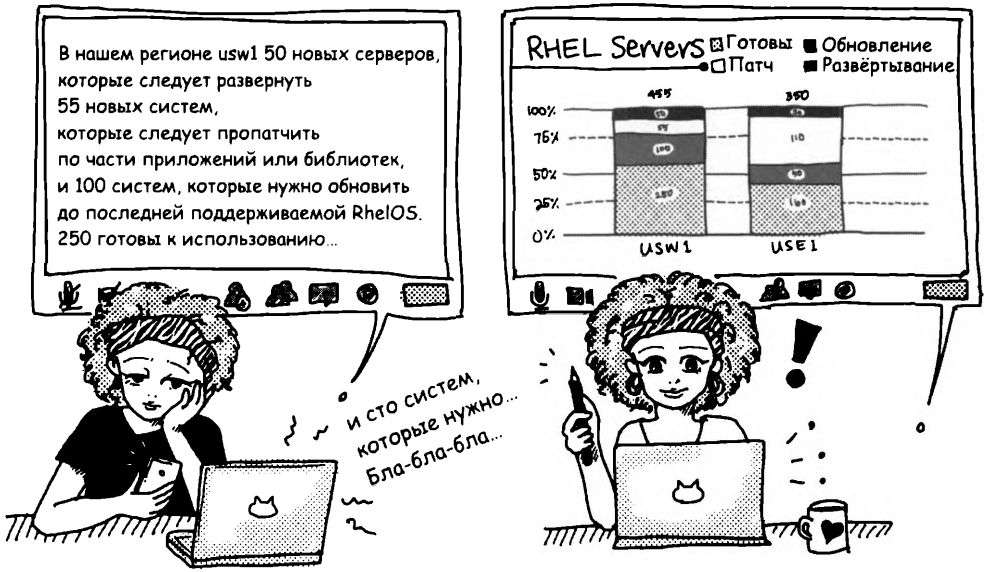

Рис. 10.1. Стили выступлений

И если вы подготовите самые важные показатели для своего технического директора, он с большей вероятностью профинансирует вашу инициативу. Не загромождайте презентационные материалы всякими деталями и мелочами.

Иногда, как бы вы ни видоизменяли свое сообщение, не получается добиться изменений, к которым вы стремитесь. Отсутствие действий сигнализирует о том, что люди не уделяют время размышлениям о том, как их работа согласуется с целями организации. Это может свидетельствовать также о системной проблеме в среде из-за нарушения циклов обратной связи. В этом случае отпустите проблему. Постоянная погоня за недостижимым результатом ведет к выгоранию. И может случиться так, что если вы подождете, то система изменится и вы сможете достичь своих изначальных целей.

Выберите канал общения

Поразмыслив над вопросами о своей аудитории, подумайте, чего вы хотите от нее. Затем решите, что лучше — устное или письменное общение. Выбор будет зависеть от вашей цели и типа сообщения.

Вербальное общение происходит в основном в режиме реального времени и позволяет передать чувства наряду с фактами. Это наиболее ценно, когда нужно пробудить эмоции или получить немедленную обратную связь.

Советы для тех, кто выступает с речью

Чем чаще вы будете выступать публично, тем лучше у вас это будет получаться. Практика — великое дело, но я хотела бы дать также несколько рекомендаций из своего многолетнего опыта, которые помогут вам развить навыки публичных выступлений.

Дыхание

Вы, может быть, замечали, что, когда нервничаете, начинаете дышать учащенно или задерживаете дыхание. Изменение ритма дыхания может положительно повлиять на вашу речь и темп изложения, благодаря чему люди будут лучше понимать вас. Вы можете добиться этого, добавив в свои заметки подсказки, напоминающие о необходимости дышать соответствующим образом, а также специально использовать паузы, чтобы привлекать внимание, или когда требуется разрядить обстановку с помощью шутки.

Терминология

Ваша речь должна быть похожа на обычный разговор. Используйте привычные и понятные слова, особенно когда рассказываете о технических вещах. Обстановка в помещении, опыт и компетенции слушателей повлияют на то, как будут восприняты ваши идеи. Избегайте жаргона и сокращений и убедитесь, что аудитория понимает все технические термины, которые вам предстоит использовать. Уделите время объяснению незнакомых концепций.

Голос

Голосовая модуляция сделает речь ярче и поможет пробудить интерес к вашему выступлению. Потренируйтесь на разных словах, чтобы увидеть, как меняется сообщение. Когда подберете подходящий вариант, сделайте пометки в презентации.

Темп речи

Темп вашего выступления может варьироваться в зависимости от аудитории. В целом для простых, понятных тем можно ускорить темп. Объясняя более сложные темы, говорите медленнее. Если аудитория включает новичков и экспертов, легко угодить в ловушку, пытаясь подобрать некую золотую середину для темпа своего выступления. Новичкам может показаться, что вы объясняли материал слишком быстро, а экспертам не понравится слишком медленная подача информации. Продуманное и логично построенное выступление позволит удовлетворить как минимум половину аудитории.

Искренность

Выражение вашего лица должно соответствовать словам. Язык вашего тела и мимика также передают информацию. Например, улыбка может вызвать интерес и привлечь внимание к вашей теме. Однако если смысловое содержание и манера поведения не совпадают, возникает диссонанс, который аудитория обычно воспринимает как фальшь. Например, когда выступающий говорит: «Я так рад поделиться с вами...», а сам выглядит скучающим и равнодушным, верите ли вы ему?

Обстановка

Наконец, очные выступления сильно отличаются от виртуальных. При живом выступлении возникает определенная энергетика между вами и аудиторией, которая позволяет понять, как люди реагируют на ваши слова. Работая с камерой, вы этой возможности лишены, что может вызывать дискомфорт. Чтобы сделать выступление перед камерой более естественным, можете создать себе виртуальную аудиторию, разместив где-нибудь сбоку видео с живыми слушателями, на которых вы можете направить свое внимание, а не просто смотреть в камеру.

Часто письменная форма общения не подразумевает немедленного ответа, будь то предложения, проектная документация, программный код или отзывы. Однако некоторые виды коммуникации, например общение в чатах и мессенджерах, могут происходить как в реальном времени, так и с задержкой по времени. Письменное общение — лучший выбор, если вы хотите сосредоточиться на фактах, или когда нужно время на обдумывание ответа, а также если срочный ответ не требуется. Однако для донесения более сложной информации может потребоваться *совместить*

письменное общение с устным, что придаст дополнительную значимость предмету обсуждения.

Любой из этих методов коммуникации выиграет, если подкреплять слова наглядными материалами. Конкретные визуализации выбираются в зависимости от информации, которой вы делитесь, и тех историй, которыми подкрепляете свои идеи. Чтобы грамотно использовать оба метода, требуется время и усилия.

Выберите подходящий тип истории

Вы можете использовать истории, чтобы осмыслить прошлое, объяснить произошедшее и определить направление будущей деятельности. Истории разных типов позволяют взглянуть на события немного иначе, и выбор подходящей истории для представления информации поможет вам склонить читателя на свою сторону. Приведем некоторые типы историй.

Фактоид

Фактоиды преподносят данные в виде интересных фактов, подчеркивая наиболее распространенные тенденции или значимые особенности. Захватывающая история может побудить аудиторию к поиску дополнительной информации. Например, вы можете продемонстрировать общее число членов сообщества, использующих определенную технологию, или общее число уникальных посетителей веб-сайта. В маркетинге такие интересные факты демонстрируют на информационных панелях сайтов, — например, для отображения статистики посещений, или указывают в информационных бюллетенях по продуктам.

Взаимодействие

Взаимодействия показывают взаимосвязи между различными наборами данных. Положительные корреляции между наборами данных означают, что когда один из показателей увеличивается или уменьшается, значения другого изменяются в том же направлении. При отрицательной корреляции зависимость между значениями наборов данных обратная — когда один показатель уменьшается, другой растет.

Выявление положительной или отрицательной связи полезно, но оно не объясняет, почему наборы данных изменяются совместно. Имейте ввиду, что корреляции могут быть мнимыми, и связь величин является простым совпадением. Убедительная история показывает корреляцию и представляет доказательства, что данные явно связаны друг с другом[1].

Например, вы можете показать на графике время запроса к MySQL и задержку запроса между конечными пунктами, чтобы более четко объяснить, связана ли производительность с рабочей нагрузкой или задержка связана с неправильной конфигурацией базы данных, что стало узким местом.

---

[1] Ознакомьтесь с публикацией «Beware Spurious Correlations» (**https://oreil.ly/qU688**) в журнале Harvard Business Review, в которой рассказывается о том, почему следует быть осторожным с корреляциями.

Изменение

Истории об изменениях — это способ описать, как что-то развивается во времени. Например, истории об изменениях можно использовать при решении проблем, связанных с управлением емкостью. Вы можете графически показать рост текущей используемой емкости с течением времени по мере ее приближения к общей сконфигурированной емкости. Кроме того, можно отобразить скорость (изменение уровня использования в разные моменты времени) и ускорение (наклоны линий), чтобы проиллюстрировать, насколько срочно требуется увеличивать емкость.

Сравнение

Истории, в которых проводится сравнение, позволяют продемонстрировать влияние данных из разных областей. Например, вы можете показать различие в характеристиках производительности при развертывании управляемой реляционной базы данных у поставщика услуг и самостоятельно управляемого экземпляра MySQL, демонстрируя сильные и слабые стороны этих подходов. Соберите воедино такие показатели, как стоимость (включая стоимость самостоятельной поддержки), производительность, масштабируемость и надежность.

Персонализация

Персональные истории связаны с реальным опытом. Например, вы вкратце описываете какой-либо инцидент, рассматривая технические вопросы в контексте опыта отдельных людей и выбора, который они сделали на основе своего ви́дения ситуации.

Сторителлинг на практике

Я хочу поделиться несколькими сценариями из своего опыта, когда представление данных командам приносило пользу. В первом случае я использовала визуальные материалы, чтобы донести информацию и изменить представления о работе своей команды. Во втором поделилась данными, которые были нужны разным аудиториям.

Случай № 1. Одна диаграмма вместо тысячи слов

Это было ужасное совещание по квартальному планированию, когда команда оценивает достижения предыдущего квартала и берет на себя обязательства по проектам на следующий квартал. Я была новичком в команде и не совсем ориентировалась в ситуации. Мои коллеги выражали недовольство, потому что «у них совсем не было времени работать над командными проектами по решению технических проблем, поскольку приходилось отвлекаться на клиентов».

Моим скрытым мотивом присоединения к команде было то, что я слышала о проблемах с просмотром рабочей очереди и о том, что возникали задержки с выполнением запросов, или же они до конца не выполнялись без каких-либо уведомлений. Менеджер специально обратился ко мне, чтобы усовершенствовать технические процессы и наладить работу команды.

После совещания по планированию я определила, какие данные потребуются для достижения целей. Я работала с командой, чтобы распределить работу на категории, связанные с обработкой входящих запросов и решением технических проблем. Я написала программный код на Perl для обращения к внутреннему API, содержащему ошибку, и, основываясь на классификации запросов, создала несколько различных информационных панелей для визуализации работы. На следующей встрече я представила изображение, показанное на рис. 10.2, показывающее, что, вопреки предположениям, мы сосредоточились в первую очередь на выбранной нами работе, а не обращениях или запросах клиентов.

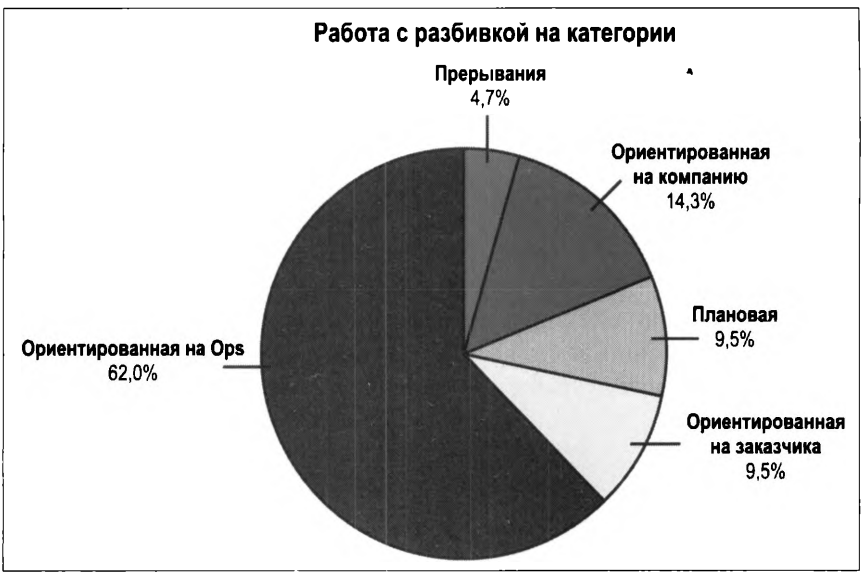

Рис. 10.2. Категории работ, выполненных за квартал, на основе сопоставления с метками

Я могла бы написать отчет, но эта простая графика была абсолютно понятна и в сочетании с основными данными обусловила изменения в расстановке приоритетов в работе нашей команды, благодаря чему клиенты стали лучше осведомлены о ходе работы.

Случай № 2. Одна и та же история для разной аудитории

Прежде чем я перейду к этому примеру, я должна объяснить важный контекст. В проектах, где вам необходимо сосредоточиться на анализе и представлении данных, подумайте о тех, кому вы будете преподносить информацию, и о том, как лучше представить данные, особенно по части технических терминов, которые допустимо использовать. Рис. 10.3 лучше проиллюстрирует то, что я имею в виду, но помните, что ваша команда и организация могут отличаться.

Рис. 10.3. Рисунок, демонстрирующий возможность говорить на одном языке с разной аудиторией. Чем больше сегмент, тем яснее общий контекст

Давайте рассмотрим уровни на рис. 10.3.

- Команда — это самый широкий уровень. Членам команды легче всего понимать друг друга и контекст. Они тесно сотрудничают друг с другом, используют общие инструменты и процессы. У них даже может быть свой сленг или специфическое использование терминологии, означающей что-то особенное в контексте их систем. В результате они выигрывают от того, что имеют всю доступную информацию, когда управляют своими системами. Это помогает в рабочем процессе любому человеку, который управляет on-call системами или производственными системами.
- Команды коллег могут использовать некоторую общую терминологию, но полезно будет установить общий контекст и понять их ожидания. В некоторых случаях при использовании сленга можно выявить неверные предположения о вопросах, требующих решения.
- Руководители могут понимать некоторую терминологию, если имеют соответствующее образование. Тем не менее чем шире сфера их деятельности и ответственности, тем больше слов может потребоваться перевести на более понятный для них язык, чтобы определить соответствующий контекст и уровень риска.
- Наконец, заказчики могут понимать вашу терминологию, но объяснить что-либо всем заказчикам, используя одни и те же термины, намного труднее. Общение с клиентами требует наибольшего внимания и аккуратности, чтобы донести верную информацию и правильно определить их ожидания.

Поэтому, представляя информацию для различных групп, продумывайте, какими сведениями следует делиться, чтобы обеспечивать необходимый уровень информированности и предоставлять данные в нужное время. Единая информационная панель или набор визуализаций могут оказаться слишком объемными.

Теперь позвольте мне поделиться историей из моей практики. Как-то раз высшее руководство объявило о закрытии нескольких объектов совместного размещения (объекты колокейшен, *англ.* colocation) в целях сокращения расходов. Нам требовалось перенести данные в ближайшие регионы, минимизировать задержки и обеспечить доступность сервисов для клиентов. Я должна была продумать действия нашей команды, которые позволили бы минимизировать влияние всех этих мероприятий на повседневную работу (например, обновление программного обеспечения и обслуживание оборудования) и работу по обслуживанию заказчиков (например, привлечение новых клиентов и увеличение емкости). Часть данных, которые необходимо было перенести, относились к данным клиентов.

Мы предоставили многопользовательскую базу данных NoSQL, состоящую из ряда таблиц. Мы управляли таблицами базы данных для клиентов, чтобы они могли сосредоточиться на своих приложениях. Поэтому мне также нужно было подумать о том, как минимизировать влияние на данные наших клиентов.

Сроки закрытия каждого объекта совместного размещения варьировались, и на основании этого я могла определить, где хранятся данные каждого клиента, и подобрать наилучшую конфигурацию для минимизации влияния задержек, с учетом новых проектов и ограничений пропускной способности системы в целом. Я разработала план мероприятий, оптимизировав такие параметры, как скорость, производительность и емкость. Затем написала несколько скриптов на Perl для запросов к API различных сервисов и на JavaScript с использованием библиотеки *D3.js* для создания диаграмм.

Для таблицы каждого клиента в каждом регионе, которую предстояло переместить, нужно было сделать следующее:

1. Выпустить копию таблицы.
2. Проследить за ходом копирования таблицы.
3. Убедиться, что копия создана.

Одновременное создание нескольких копий таблицы и выполнение других действий по администрированию этой таблицы не допускалось.

Для каждого региона нам нужно было сделать следующее:

1. Подождать, пока все клиенты переведут свои услуги на новую конечную точку (минимизировать проблемы задержки).
2. Обновить конфигурации таблиц, чтобы перевести таблицы в определенный регион.
3. Выключить все серверы в регионе.
4. Уведомить SiteOps об отключении серверов и отмене выделения ресурсов.

Информационная панель для команды

Информационная панель для команды выглядела так, как показано на рис. 10.4. Таблица содержала упорядоченный список задач в порядке приоритета, включающий регионы, текущие работы (Р), незатронутые регионы и завершенные рабо-

ты (С). Эта информация позволяла службе эксплуатации быстро решать, что делать при новых запросах клиентов на изменение определенной таблицы — останавливать задачу или дожидаться завершения работы.

	Регион 1	Регион 2	Регион 3	Регион 4	Регион 5	Регион 6
Задача 1	С	С	С	С	С	С
Задача 2	-	С	-	Р	С	С
Задача 3	С	С	С	С	Р	С
Задача 4	С	-	-	С	С	Р

Рис. 10.4. Общий для команды контекст, включающий запланированные работы, места проведения работ и ход их выполнения

Глядя на строки таблицы, любой SRE-специалист мог быстро узнать, в каких регионах задачи полностью выполнены, а в каких — в процессе выполнения. Мы знали, что в тех регионах, где работы еще не завершены, следовало особенно осторожно действовать при установке обновлений: либо приостанавливая миграцию данных, либо перенаправляя трафик клиентов в следующий объект колокейшен, чтобы минимизировать прерывания.

Таким образом, при минимальной координации действий каждый мог завершить модернизацию, вычисления и развертывание таблиц для региона, где работы уже были завершены. И наконец, отменить выделение ресурсов и отправить запрос в SiteOps на отключение оборудования, когда все клиенты завершат работу по миграции.

Информационная панель менеджера

Информационная панель менеджера выглядела так, как показано на рис. 10.5. Им не нужно было знать все детали каждого отдельного задания. Требовались только данные о том, какие работы ведутся, нет ли у нас проблем и вовремя ли мы закончим.

На этой информационной панели отображается показатель для каждого региона и реплики, что позволяет видеть весь объем работ на объектах колокейшен. Диаграммы обновляются ежедневно на основе данных о выполненной работе. Например, взглянув на эту информационную панель, руководители могли видеть, что три региона укладывались в запланированный срок при выполнении работы, а один — нет. Тогда они задавали ключевые вопросы команде и получали обновленную информацию для предоставления заинтересованным сторонам.

Поскольку это был долговременный проект, руководству предоставлялась информация, необходимая для изменения приоритетов в работе или прерывания работы, поскольку они могли сразу же видеть, как это влияет на сроки осуществления проекта.

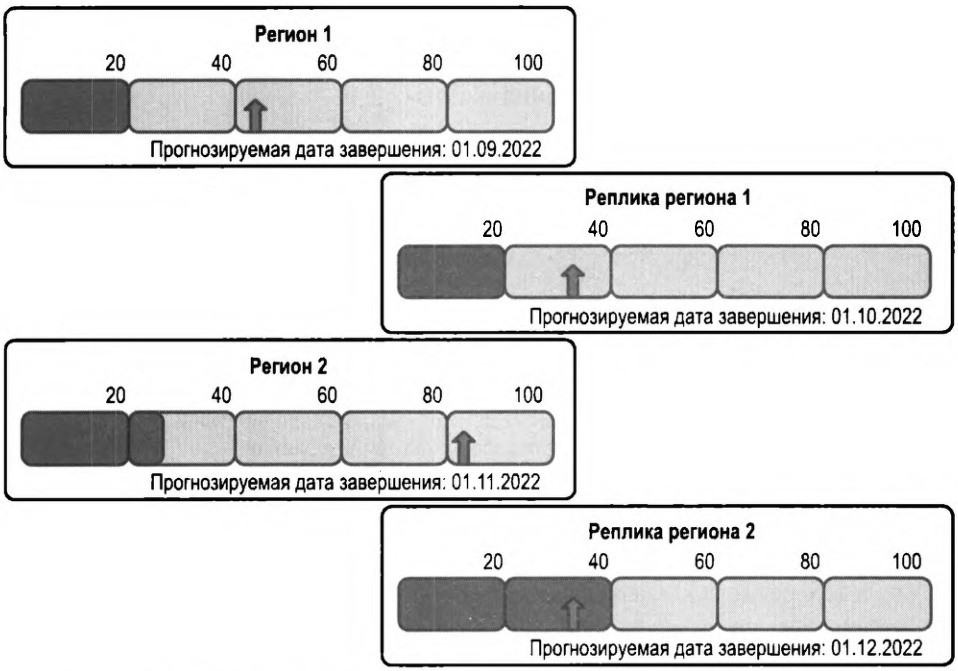

Рис. 10.5. Контекст для менеджеров включает планируемое состояние работы, текущее состояние работы и прогнозное состояние работы

Информационная панель клиента

Информационные панели клиентов, показанные на рис. 10.6, предоставляют клиентам информацию о том, где таблицы их базы данных доступны, что мы запланировали для их таблиц, каковы предполагаемые даты завершения и в каких регионах их таблицы недоступны.

	Регион 1	Регион 2	Регион 3	Регион 4	Регион 5
Таблица 1	Готово к использованию	Готово к использованию	1 мая	15 мая	Готово к использованию
Таблица 2			Готово к использованию	Готово к использованию	
Таблица 3	Готово к использованию	Готово к использованию	15 мая	Готово к использованию	
Таблица 4	Готово к использованию	Готово к использованию	Готово к использованию	Готово к использованию	

Рис. 10.6. Упрощенный вид панели клиента

Обратите внимание, что клиентам не нужно было знать о репликах регионов. Реплики были созданы для резервного копирования с низкой задержкой в пределах одного объекта колокейшен. Однако для руководства эта информация была важна, поскольку она показывала, своевременно ли выполняются мероприятия по отключению сервисов в рамках объекта колокейшен.

Ни одному клиенту не нужно было обращаться к нам за обновлением своих таблиц. Вместо этого они могли заранее выполнить миграцию, когда были готовы, и могли быть уверены, что не добавят задержку к своим запросам, развертывая сервисы в неправильных регионах.

Но я все же отправляла по электронной почте сводки, когда мы завершали все действия с их таблицами. Благодаря обновляемой графической информации они могли устанавливать приоритеты в работе по миграции на объекте колокейшен.

Основные выводы

Эти различные визуализации сократили количество запросов на поддержку и получение данных о текущем состоянии дел, позволив членам команды сосредоточиться на работе. Адаптируйте свое сообщение в соответствии с нуждами вашей аудитории. Не требуется предоставлять все собираемые данные. Лучше сосредоточиться на той информации, которая важна для отдельных людей. Расскажите своей аудитории, какие данные отсутствуют и что они могут узнать из собранной информации.

Варианты графического представления информации

> Самая большая ценность изображения — это когда оно заставляет нас обратить внимание на то, что мы никак не ожидали увидеть.
>
> — *Джон В. Тьюки (John W. Tukey)*

Я показала несколько способов визуализации данных в предыдущих двух случаях. Тем не менее существует множество других вариантов графического представления информации, которые помогут превратить ваши данные в убедительные истории. Вы также можете использовать принципы дизайна, чтобы помочь вашей аудитории увидеть то, что вы хотите.

Визуальные подсказки

Визуальные подсказки делают информацию интуитивно понятной. Четыре основных визуальных сигнала — это цвет, форма, движение и пространственное положение.

Цвет

Варьируя оттенки, вы можете обозначить взаимосвязь между двумя показателями или моментами времени. Чтобы показать количество или силу, измените насыщенность. Изменение цветовой температуры заставляет нас воспринимать цвет как теплый или холодный, что можно использовать для фокусировки внимания. Более теплые цвета выдвигаются на передний план, а более холодные становятся фоном. Используйте цвет, чтобы показать важные точки данных, но не полагайтесь на него как на единственное средство выражения этих данных.

Форма

Вы можете изменять длину, ширину, ориентацию, размер и форму. Например, увеличьте размер какого-либо объекта или используйте свободное пространство, чтобы подчеркнуть его важность.

Движение

Мерцание и движение привлекают внимание к отдельным важным областям, однако могут оказаться отвлекающими или раздражающими. Вы можете обозначить движение с помощью других визуальных сигналов, а не демонстрировать его напрямую.

Положение

Вы можете использовать двумерное изображение и пространственную группировку.

Подсказки неуместны, если они вводят в заблуждение или мешают аудитории понять тот зрительный образ, который вы демонстрируете. Например, не используйте круги разного размера, чтобы показать категориальные данные, если разница в величине категорий не имеет никакого значения.

Узнайте больше о принципах дизайна из книги Робина Уильямса (Robin Williams) «Дизайн. Книга для недизайнеров» («The Non-Designer's Design Book» (Peachpit Press)).

Типы диаграмм

Вместо того чтобы просто обмениваться строками данных или использовать привычную круговую диаграмму, выберите диаграмму, подходящую для ваших данных.

Таблицы данных

Таблицы данных организуют данные в строки и столбцы. Таблицы могут быть ценным инструментом при выполнении следующих действий.

Планирование

Так, вы можете перечислить по пунктам список требований к предложению, провести мозговой штурм при квартальном планировании и подробно остановиться на деталях, относящихся к каждому выявленному элементу, например автору предложения или периоду времени.

Документирование

Например, можно составить список вариантов или провести сравнение между различными инструментами и услугами.

Определение

Списки, к которым можно быстро периодически обращаться для получения тактических указаний. В качестве примера можно привести начальные страницы или источники трафика для веб-сайтов.

Исследование

Большие наборы для фильтрации, отображения данных и детализации отдельных запросов.

Таблицы в подавляющем большинстве случаев используются для представления большого объема данных, поэтому дополните их другими визуальными образами, чтобы обратить внимание на тенденции, экстремальные значения и различные закономерности в исходных табличных данных.

В табл. 10.1 табличное представление используется для сравнения пропускной способности сервиса Amazon DynamoDB в режимах «по требованию» и «с распределением». Такой формат работает, потому что данных не так много, и понятно, что именно отличается.

***Таблица. 10.1**. Пропускная способность сервиса Amazon DynamoDB, представленная в виде таблицы*[2]

	По требованию	**С распределением**
На одну таблицу	40К единиц запросов на чтение и 40К единиц запросов на запись	40К единиц запросов на чтение и 40К единиц запросов на запись
На одну учетную запись	Неприменимо	80К единиц емкости чтения и 80К единиц емкости записи
Минимальная пропускная способность для любой таблицы или глобального вторичного индекса	Неприменимо	Одна единица емкости чтения и одна единица емкости записи

Timestamp UTC	cache_status	client_ip_hash	content_type	geo_c
2021-01-18 21:15:25	HIT	0a8b1069f217aed3f6f3307136f2e0a59d351ded15f339428af10cd011fb0a52	binary/octet-stream	cambr
2021-01-18 21:15:25	HIT	dbdb4a3a86785feaa5e29dd1bf2466bb187ff3f967ddfe0e6027a5e5c92bf78e	application/octet-stream	ashbu
2021-01-18 21:15:25	HIT	4dbc8a020963bf0553e087fea64e85c875b4cdf1156e38141c27d67379629a40	application/octet-stream; charset=utf-8	louis
2021-01-18 21:15:25	HIT	769a0fe4eb80fa5e2685ea60ac6a861ef8717eaa543fe17e3029113ba7c8dd12	text/plain; charset=utf-8	ashbu
2021-01-18 21:15:25	HIT	cbf0ab785c36706715ff57c25996a39327f49bf2add13d78a99030070f45ebef	application/octet-stream; charset=utf-8	dubli
2021-01-18 21:15:25	HIT	afca926e07a90d729928645dbfb566dda787599478a6b248c4c1ea506750c1c1	binary/octet-stream	londo
2021-01-18 21:15:25	HIT	61cba6bf0089bcd8273eb5ca78245f636171ba83948cb30795b17130ed2aa962	application/octet-stream; charset=utf-8	san j
2021-01-18 21:15:25	HIT	794e7164d2f423d2a10173106e2b9c87601c2bea9b91fb78a1209c81881fdca5	text/html	pavas

Рис. 10.7. Необработанные данные в формате таблицы для Rubygems.org[3]

[2] «Service, Account, and Table Quotas in Amazon DynamoDB» (**https://oreil.ly/M2cjV**), Amazon, последнее изменение от 15 декабря, 2020 г.

[3] «Honeycomb's Play with Live Rubygems.org», Honeycomb (**https://honeycomb.io**).

На рис. 10.7 показана часть таблицы с данными с сайта Honeycomb Play with Live Rubygems.org data playground (**https://oreil.ly/hmGNE**). В таблице данных используется табличный формат наряду с визуальными подсказками к журналам сырых событий в таблице данных. Фон строк чередуется, чтобы было легче читать таблицу.

Столбчатые диаграммы

Столбчатые диаграммы полезны при сравнении количественных данных, представленных в двух-трех категориях. Составные столбчатые диаграммы расширяют идею визуализации. Каждый столбец показывает, как соотносятся отдельные элементы в рамках категории, и их вклад в общий итог. Столбчатые диаграммы чаще отображаются вертикально, в основном при представлении данных временного ряда. Горизонтальная ориентация может оказаться удобнее, если категории имеют длинные названия.

Например, я использовала столбчатые диаграммы для визуализации аудитов системы на нескольких объектах колокейшен, чтобы увидеть количество узлов с устаревшими операционными системами. Столбчатые диаграммы применяются также, чтобы показать потребление дискового пространства рядом каталогов, разделов или серверов, что помогает понять, как используется емкость хранилища.

Линейные диаграммы (графики)

Линейные диаграммы показывают, как меняется значение, например с течением времени, и позволяют понять взаимосвязь двух переменных. Приведите несколько графиков на диаграмме, чтобы отобразить тенденции между рядами данных. Они часто используются для демонстрации временны́х тенденций и различий между рядами данных.

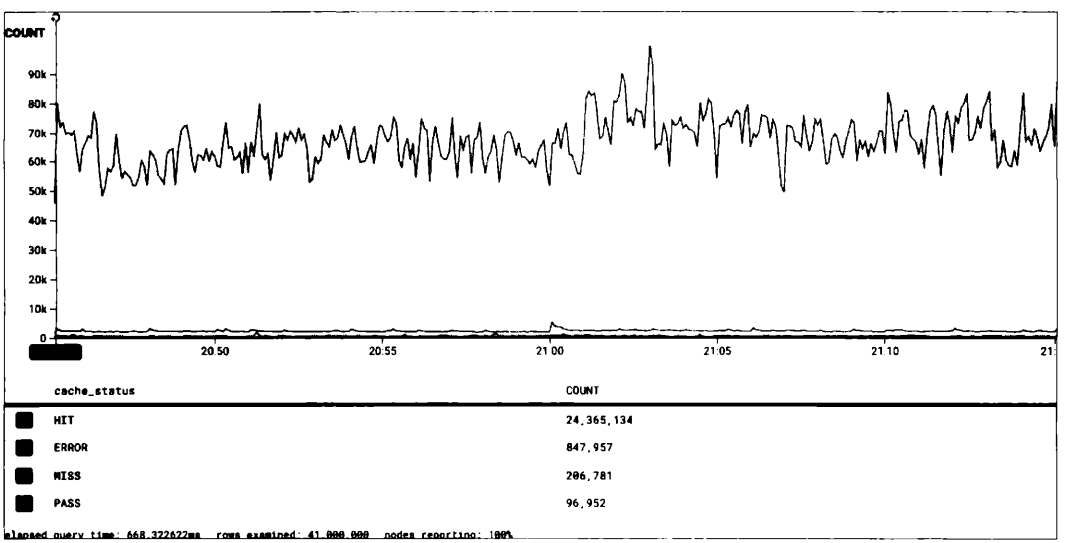

Рис. 10.8. Линейная диаграмма, отображающая результаты для Rubygems.org

Часто по вертикальной оси располагаются статистические данные — например, количество, сумма или среднее значение измеряемого атрибута по всему набору данных. Используйте непрерывный интервал на горизонтальной оси, например время.

На рис. 10.8 приведен еще один пример с сайта Honeycomb live Rubygems.org data playground (**https://oreil.ly/AzouI**). График показывает число очков, а в таблице приведена легенда — общее количество удачных обращений к кеш-памяти, неудавшиеся попытки, ошибки и проходы за определенное время.

Диаграмма с областями

Диаграммы с областями основаны на линейных диаграммах и показывают изменение количественных данных с течением времени. Комплексные диаграммы с областями позволяют показать часть целого или кумулятивные значения.

Тепловые карты

Тепловые карты показывают образцы данных с помощью затенения или цвета. Одна из проблем графической информации такого типа в том, что нужно контролировать доступность используемых цветовых схем и отсутствие искусственных градиентов. Тепловые карты также могут создавать проблемы и затруднять понимание, когда нет четко выраженных образцов.

Flame-графики

Flame-графики (от *англ.* flame — пламя) — это способ визуализации профилирования программного обеспечения. Они помогают при решении проблем, связанных с истощением ресурсов.

Древовидные карты

Древовидные карты используют плитки разного размера, иллюстрирующие количественные соотношения. По сути, это двумерные составные линейчатые диаграммы. Древовидные карты помогают увидеть вклад множества небольших элементов в суммарную величину. Чтобы передать дополнительную информацию, плитки могут обозначаться цветом.

Например, используйте древовидную карту для отображения пространства на жестком диске: большие прямоугольники — для файлов, занимающих много места на диске, и группы прямоугольников — для каталогов. Цветовая маркировка отдельных плиток позволяет указать такие атрибуты, как типы файлов, их возраст или право собственности.

Дополнительные ресурсы по визуализации диаграмм

Чтобы узнать больше о визуализации информации, ознакомьтесь с книгами Эдварда Тафти (Edward Tufte): «The Visual Display of Quantitative Information», «Envisioning Information», «Visual Explanations» и «Beautiful Evidence» (**https://oreil.ly/YgGks**,Graphics Press).

С подробной информацией о других диаграммах можете ознакомиться на сайте AnyChart's «Chart Type: Chartopedia» (**https://oreil.ly/S7dVS**).

Узнайте больше о flame-графиках (**https://oreil.ly/tHfqS**) у их изобретателя и эксперта по производительности систем Брендана Грегга.

Рекомендуемые методы визуализации

Представляя информацию, вы ведете повествование так, как считаете нужным и задаете способ интерпретации данных. Современные инструменты позволяют нам исследовать данные и взаимодействовать, проверять гипотезы или предоставлять альтернативные версии для объяснения происходящего.

Представьте, что вы управляете кластером веб-серверов с балансировкой нагрузки. Вы можете использовать графики разных цветов, чтобы показать общее количество ошибок каждого сервера. Множество линий могут выглядеть смазанно, но экстремальные значения, отражающие типы ошибок, хорошо заметны.

Вы можете подготовить также отдельные графики для каждого сервера, по характерным формам которых можно понять тип ошибки. По форме графиков можно с первого взгляда понять, когда определенный сервер сталкивается с большим количеством ошибок и какого типа эти ошибки.

При использовании графических материалов возьмите на вооружение следующие методы.

- Выделите ключевые моменты. Не полагайтесь только на текст. Вместо этого подберите подходящие визуализации, облегчающие понимание ключевых моментов.
- Используйте подходящие цвета для информационной панели с несколькими диаграммами, а также в пределах одной диаграммы. Цвет позволяет акцентировать внимание на определенных деталях. Уменьшите насыщенность для вспомогательных или второстепенных данных. Ограничьте количество используемых цветов. Хотя цвет может быть информативен, диаграммы должны быть понятны, даже если отображаются в оттенках серого.
- Графики всегда должны иметь размеченные оси и сопровождаться легендой. Однако следует избегать дублирования информации в рамках диаграммы. Например, нет смысла приводить легенду, если вы используете столбчатые диаграммы с маркированными категориями.
- Приводите ссылки на источники данных. Тогда, если график покажется непонятным, люди смогут уточнить данные и при необходимости разобраться с возникающими вопросами.
- Дизайн определяется форматом выступления. Не стоит сочинять пространные тексты для презентаций. В потоке слов самые важные сообщения могут остаться незамеченными. Вы поймете ценность подробно изложенной информации, с четкими и конкретными шагами, которая будет представлена на вашей информационной панели, когда вам придется решать неотложную проблему в два часа ночи.

- Указывайте ключевые выводы с помощью аннотаций и выделения при визуализации конкретного набора данных.
- Проектируйте информационную панель таким образом, чтобы диаграммы могли объяснить все нюансы при поиске неисправности, особенно если вы создаете ее на тот случай, что вас вызовут на работу в неурочное время.

Взгляните на различные визуализации, относящиеся к одному набору данных, и оцените, как они меняют суть сообщения: Nathan Yau (Натан Яу), «One Dataset, Visualized 25 Ways» на сайте FlowingData (**https://oreil.ly/DmUGO**).

Заключение

Эффективная презентация предоставляет аудитории интерпретированные данные и соответствующий контекст для понимания и быстрого принятия решений. Расскажите убедительную историю, отвечающую задачам и интересам вашей аудитории. Учитывайте природу данных, основную мысль выступления и способ интерпретации данных. Помните, что истории составляют основу эффективной коммуникации. Поэтому, когда готовитесь представить информацию коллегам-сисадминам, руководству, клиентам, задайте себе следующие вопросы.

- Кто является вашей аудиторией? Что волнует этих людей и в чем они нуждаются?
- Какова природа данных? Историю какого типа вы рассказываете?
- Какой формат будет наиболее эффективным для этой конкретной аудитории? В каком виде лучше представить информацию — письменном, устном, графическом, или использовать средства мультимедиа?
- Какую идею вы хотите донести до аудитории? Какой контекст им нужен, чтобы прийти к тем выводам, которые вы имеете в виду?
- Какая информация нужна аудитории, чтобы понять вашу историю? Какую информацию следует опустить, потому что она отвлекает от истории? Какую информацию следует включить, даже если она не связана напрямую с вашим рассказом, чтобы аудитория пришла к выводам, которые вы, возможно, не рассматривали?
- Визуализации помогают лаконично представить информацию. Какой тип визуального контента будет эффективен для вашей истории?

Вы поймете, что презентация прошла успешно, когда аудитория поймет ваши мысли и сможет быстро принимать решение на основе вашей информации.

ЧАСТЬ III

Сборка системы

В части II вы узнали о методах, которые помогают обеспечивать надежность и устойчивость систем. В части III я сосредоточусь на вопросах сборки систем, сводя воедино методы из части II, а также фундаментальные стандартные блоки (вычислительные среды, хранилища и сети) из части I. Инфраструктура обширна и разнообразна. Общепринятой практикой является избавление от «серверов-снежинок» (snowflake servers — серверы, которые уникальны и неповторимы, подобно снежинкам, **https://oreil.ly/zWuZ4**) с помощью подхода «инфраструктура как код». Тем не менее каждая организация практикует свои уникальные методы, что затрудняет решение вопросов управления инфраструктурой и приводит к ненужным спорам о единственно правильном подходе.

За годы работы в отрасли я видела множество рекламируемых инструментов, методов и практик управления инфраструктурой. Некоторые из них выдержали испытание временем, другие — нет. В конечном счете вам нужно создавать многократно используемые, версированные артефакты из исходных кодов. Это подразумевает создание и настройку конвейера непрерывной интеграции и непрерывной доставки. Автоматизация инфраструктуры снижает затраты на создание и поддержку сред, уменьшает риск концентрации критически важных знаний у небольшого числа специалистов, а также упрощает тестирование и обновление сред.

Разработка сценариев инфраструктуры

В *главе 1*, рассуждая о ваших системах, я приводила в пример процесс приготовления торта. Кулинария, действительно, помогает объяснить работу систем, поэтому давайте воспользуемся еще одной аналогией из этой области. Печенье — восхитительное сладкое лакомство. В состав этих небольших кондитерских изделий обычно входят сахар, жир и мука в определенном соотношении. Вы можете купить готовое печенье, испечь его из готового песочного теста или самостоятельно замесить тесто для печенья из ингредиентов, имеющихся на кухне.

То же самое и с инфраструктурой. Вы можете использовать сервисы, покупать ресурсы в предварительно упакованном виде или создавать свои собственные из того, что есть в наличии. Все проблемы, которые могут возникнуть с вашей инфраструктурой (процесс, состояние ресурсов или условия среды), можно устранить с помощью сценариев инфракода и создания нужных рецептов для инфраструктуры. В этой главе я объясню, почему вам необходимо создавать сценарии инфраструктуры независимо от ее типа, а также расскажу о различных подходах к моделированию инфраструктуры, которые помогают планировать инфраструктурный проект.

Эта глава будет посвящена инфракоду — на языках Ruby, YAML или других, используемых для описания вашей инфраструктуры. В *главе 12* я расскажу о модели «Инфраструктура как код» и методиках, применимых к вашему инфракоду.

Зачем создавать сценарии инфраструктуры?

Я повидала организации, в которых темпы изменений замедлялись, потому что вечно имелось так много срочной и отвлекающей работы, что на сценарии времени уже не оставалось. Иногда это происходило из-за страха людей потерять работу в результате автоматизации. Для автоматического управления инфраструктурой можно создать инфракод на языке, который понятен как человеку, так и компьютеру, чтобы описать оборудование, программное обеспечение и сетевые ресурсы — для автоматизации согласованного, повторяемого и прозрачного управления ресурсами.

Независимо от того, какие инструменты автоматизации управления инфраструктурой внедряются в организации, вы можете получить следующие преимущества.

- Повысить скорость развертывания одной и той же инфраструктуры.
- Снизить риски для инфраструктуры за счет устранения ошибок, возникающих при ручной настройке и развертывании.
- Повысить прозрачность систем управления, безопасности и контроля соответствия требованиям в рамках организации.
- Стандартизировать инструменты конфигурирования, предоставления ресурсов и развертывания.

Эти конечные результаты не всегда тесно связаны с конкретными преимуществами для бизнеса, поэтому иногда трудно добиться финансирования или поддержки проекта, связанного с инфракодом. В некотором смысле это логично: автоматизация ручной работы требует времени, а сложные вещи не всегда поддаются автоматизации.

Чтобы мотивировать свою команду и добиться согласия заинтересованных сторон, особенно когда у команды имеется множество высокоприоритетных задач одновременно, попробуйте предпринять следующие шаги.

- Продумайте и задокументируйте процесс ручного предоставления ресурсов, конфигурирования и развертывания (что и как делается, как решаются проблемные ситуации).
- Подберите небольшие проекты, которые могут быть успешно завершены и поддерживают вашу точку зрения.

Давайте рассмотрим несколько способов, с помощью которых вы можете описать концепцию так, чтобы она соответствовала потребностям бизнеса.

Согласованность

Вы единообразно развертываете и настраиваете системы, которые были протестированы и документированы. Это соответствует потребностям бизнеса, поскольку согласованность повышает производительность и эффективность команды.

Масштабируемость

Инфракод упрощает процесс предоставления и блокирования доступа к ресурсам, позволяя вам по мере необходимости активировать и деактивировать парки систем с минимальными усилиями. Эти усилия могут представлять собой простое изменение масштаба вручную — в большую или меньшую сторону, полностью автоматизированное управление облаком или любую другую комбинацию действий, которые позволяют системе динамически реагировать на пики и спады спроса и одновременно предоставляют людям полномочия управлять работой автоматизированной системы. Это соответствует потребностям бизнеса, поскольку масштабируемость способствует увеличению дохода, дифференциации продукта, снижению затрат на инфраструктуру постоянной готовности и повышает степень удовлетворенности пользователей.

Делегирование полномочий

Вы определяете уровни ответственности, чтобы предоставить различным командам автономию в управлении ресурсами. Вы решаете, как распределить

ответственность между командами, связанными с инфраструктурой, безопасностью и приложениями, предоставляя возможность самообслуживания в рамках установленных границ и сохраняя общий обзор.

Это соответствует потребностям бизнеса, поскольку делегирование полномочий облегчает внедрение новых продуктов и позволяет удерживать расходы в приемлемых границах. Операционные отделы могут проверять, как другие отделы используют ресурсы, чтобы убедиться в эффективном расходовании бюджетов. Такая автономия ведет к увеличению доходов и дифференциации при разработке продукции.

Отслеживаемость

Отслеживание изменений инфракода с помощью управления версиями дает вам историю изменений системы и аудиторский след, так что все смогут ответить на вопросы о созданных системах. Это соответствует потребностям бизнеса, поскольку отслеживаемость способствует снижению затрат: вы можете вывести из эксплуатации системы, которые больше не должны использоваться, и пересмотреть решения, если после того, как предположения вдруг изменятся, им можно будет найти лучшие альтернативы.

Инкультурация

Журналы изменений в системе управления версиями облегчают процесс адаптации новых членов команды. Они видят, как вы выполняете работу, и могут действовать аналогичным образом. Это соответствует потребностям бизнеса, поскольку инкультурация повышает производительность и эффективность.

Экспериментирование

Инфракод облегчает создание тестовых сред, дает возможность опробовать новые технологии и быстро переводить их в производственную среду, если эксперименты оказываются успешными. Это соответствует потребностям бизнеса, поскольку эксперименты могут увеличить доход и помочь команде сосредоточиться на дифференциации рынка.

Вы лучше всех знаете свою организацию и ее руководителей. Исходя из ориентиров компании и ее головной организации, определите область проекта и соответствующие цели, которые помогут достичь этих ориентиров. Как только вы определили область и цели проекта, вы можете моделировать инфраструктуру, исходя из конкретной перспективы, чтобы успешно реализовать проект и цели.

Три подхода к моделированию инфраструктуры

Подумайте об инфраструктуре, которой вы управляете. У вас может быть физическое оборудование или вычислительные экземпляры с различными зависимыми службами. Каждая вычислительная сущность будет иметь ОС и может содержать несколько контейнеров или виртуальных машин. Сеть соединяет различные сущности с помощью списков контроля доступа или политик, которые разрешают или

ограничивают коммуникации. Теперь подумайте о том, как вы описываете свою инфраструктуру.

На высоком уровне вы можете предоставлять ресурсы в облаке. Каждый ресурс имеет жизненный цикл, от подготовки к работе до изъятия из обслуживания, как показано на рис. 11.1 слева. Или вы можете сосредоточиться на низком уровне, как показано справа, и настроить одиночный вычислительный экземпляр на соответствие определенному набору политик, чтобы создавать образы машин с помощью повторяемого, последовательного и многократно применяемого способа.

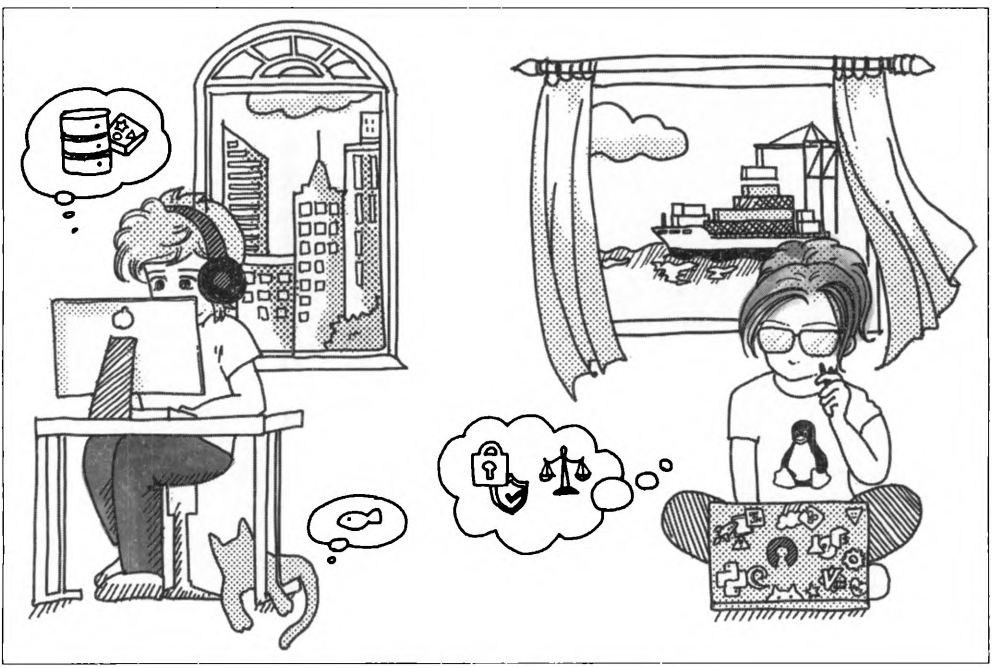

Рис. 11.1. Принятие решения о способе реализации инфракода. Все варианты допустимы

Технологический прогресс подобен биологической экосистеме с различными средами обитания, экологическими нишами и биологическими видами. Технология предоставляет какой-то новый инструмент, который закрывает некую потребность. Сообщество принимает на вооружение если не технологию, то определенные методы, что приводит к сотрудничеству и изменению модели взаимодействия. Другие технологические платформы меняются в соответствии с новыми потребностями общества.

Я надеюсь показать вам в этой книге общие закономерности, потому что книги отражают определенный момент времени, и когда вы будете ее читать, появятся уже более новые инструменты, технологии и методы. Ознакомьтесь с документацией к конкретной версии выбранного вами инструмента, чтобы получить актуальные рекомендации.

С учетом всего этого при выборе инструмента подумайте, какой из подходов соответствует вашим насущным потребностям в управлении инфраструктурой, программируя ее на выполнение следующих действий:

- создание образов машин;
- предоставление ресурсов инфраструктуры;
- настройка ресурсов инфраструктуры.

Код для создания образов машин

В начале своей карьеры я развертывала и обслуживала множество физических систем. К счастью, на одной работе у меня был дубликатор жестких дисков, который позволял мне клонировать один жесткий диск на несколько дисков одновременно, чтобы ускорить процесс развертывания. Конечно, мне все равно приходилось обновлять конфигурации для каждой системы после установки нового клонированного диска, но я экономила массу времени, т. к. не нужно было устанавливать и обновлять ОС. Клонирование выполнялось вручную, но это было быстрее, чем подготовить физическую машину, установить ОС с компакт-дисков, а затем думать, как обновить систему, пока она еще потенциально уязвима.

Такой подход известен как создание золотого образа: идеального, заведомо хорошего шаблона, на основе которого вы создаете образы для новых систем. Современные рабочие процессы концептуально восходят к этому методу, где *образ машины* (например, образы машин Amazon или шаблоны VMware) служит во многом той же цели, что и золотые образы. С помощью образов машин вы автоматизируете сборку системы, усиливаете защиту ОС для снижения уязвимости, предварительно устанавливаете все необходимые и стандартные инструменты и в конечном итоге предоставляете вычислительные ресурсы на более безопасной и надежной основе.

Важнейшей задачей системных администраторов было развертывание физических компьютеров, но теперь этот процесс претерпел изменения. Вычислительная инфраструктура, которой вы управляете, может включать физические машины, виртуальные машины и контейнеры.

Поскольку технологии повторно используют многие схожие концепции, разработчики систем автоматизации инфраструктуры склонны задействовать сложившуюся терминологию, но это может приводить к путанице, когда вам нужно конкретизировать уровень абстракции. В этом объяснении я буду ссылаться на термины *машины* и *образы машин*, понимая, что на практике слово «машина» имеет множество значений — от физических систем и виртуальных машин до контейнеров.

Посмотрите на рис. 11.2, где показан образ машины для сервера, на котором будет работать определенная ОС и ряд контейнерных приложений.

Это примеры инструментов, создающих образы машин:

- Packer — мультиплатформенный инструмент для создания образов машин;
- EC2 Image Builder — для образов машин Amazon;
- Buildah — для создания образов контейнеров Open Container Initiative (OCI).

Вероятно, вы заходите написать программный код для создания образов машин, если для вас важны следующие аспекты:

Рис. 11.2. Построение образов машин

- вам нужно точно знать, что все системы созданы на основе стандартного обновленного базового образа;
- нужно установить набор общих инструментов или утилит на все системы;
- вы хотите использовать образы, созданные внутри системы, с информацией о происхождении каждого программного пакета в системе.

Код для предоставления инфраструктуры

Когда поставщики внедряли облачные архитектуры, я радовалась возможности быстрого доступа к сложной инфраструктуре с помощью простых API. Никаких тебе стоек и полок, не нужно прокладывать кабели и настраивать сетевые порты в дополнение к установке приложения. Установка комплекта средств разработки программного обеспечения (*англ.* software development kit, SDK) и инструментария, предоставляемых провайдером, позволяла мне быстро подготавливать и настраивать необходимую инфраструктуру. Предоставление облачных ресурсов с помощью инфракода позволяет вам решать следующие задачи:

- выбирать виртуальные машины, контейнеры, сети и другую необходимую инфраструктуру, доступную через API, исходя из ваших архитектурных решений;
- объединять отдельные компоненты инфраструктуры в стеки;
- устанавливать и настраивать компоненты;
- развертывать стек как единое устройство.

Взгляните на рис. 11.3, на котором показаны отдельные предоставляемые ресурсы (например, серверы и базы данных).

Ниже приведены некоторые инструменты, предоставляющие ресурсы инфраструктуры.

- HashiCorp Terraform;
- Pulumi;

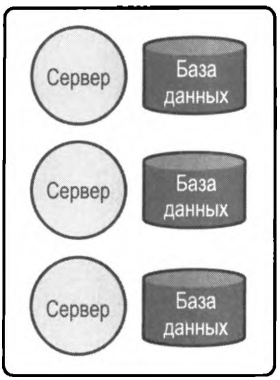

Рис. 11.3. Предоставление ресурсов

- AWS CloudFormation;
- Azure Resource Manager;
- Диспетчер развертывания Google Cloud.

Чтобы создавать корректный инфракод для успешного предоставления и конфигурирования ресурсов инфраструктуры, нужны большие знания. Более того, хотя поставщики облачных решений часто предлагают принципиально схожие услуги, существуют тонкие различия в возможностях. Сравнение функционала провайдеров один в один, особенно с помощью инфракода, может оказаться затруднительным, поскольку синтаксис и средства сильно различаются. При наличии мультиоблачной архитектуры вы, скорее всего, выберете такие фреймворки, как Pulumi и Terraform, которые могут быть развернуты на множестве платформ.

Инфракод уводит на второй план вопрос о том, как это работает. Люди взаимодействуют с этими системами и должны знать не только об автоматизации развертывания. Когда возникают проблемы (а это неизбежно), вам нужно знать, где искать проблему.

Например, предположим, что вы написали инфракод для управления записями DNS для почты и забыли записи SPF и DKIM. В этом случае неправильная конфигурация может нарушить доставку почты из вашего домена большинству провайдеров. К сожалению, проверка правильности синтаксиса не предотвращает ошибок, влияющих на работоспособность кода. Кроме того, повторное развертывание инфракода не поможет обнаружить неверные настройки конфигурации.

Вам может понадобиться написать код для предоставления инфраструктуры, если у вас уже есть или вам требуется следующее:

- системы, которые уже частично используют предоставление ресурсов;
- поддержка мультиоблачных технологий;
- многоуровневые приложения;
- воспроизводимые среды — например, тестовая среда, которая является уменьшенным клоном рабочей среды.

Код для настройки инфраструктуры

Настройка ресурсов инфраструктуры с помощью инфракода позволяет сконфигурировать программное обеспечение и сервисы, как только будет доступна аппаратная инфраструктура. Посмотрите на рис. 11.4, где показана конфигурация ОС и приложений, которая должна быть последовательной, повторяемой и надежной.

Рис. 11.4. Настройка инфраструктуры

Вот несколько примеров инструментов для настройки ресурсов инфраструктуры:

- CFEngine (**https://oreil.ly/8v4HK**);
- Puppet (**https://oreil.ly/dTIk8**);
- Chef Infra (**https://oreil.ly/n62da**);
- Salt (**https://oreil.ly/BfTSD**);
- Red Hat Ansible (**https://oreil.ly/olg8H**).

Каждый из них реализует управление конфигурацией немного по-разному, используя различную терминологию для описания стандартных блоков, которые представляют собой средства конфигурации инфраструктуры.

Вы можете написать код для настройки инфраструктуры, чтобы выполнять следующие действия:

- Управлять установкой и настройкой программного обеспечения на системах.
- Настраивать параметры ОС.
- Повторно устанавливать и настраивать системы.
- Автоматизировать исправление изменений, непосредственно внесенных в системы вручную.

Начало работы

Я рассказала о трех подходах, которые помогут вам сузить круг вопросов, связанных с инфракодом. У различных инструментов есть вспомогательные функции. Они определяются той инфраструктурой, которой вы собираетесь управлять, и могут заставить вас выбрать и другую базовую технологию!

Если вы не используете инфракод, подумайте, как тот или иной инструмент вписывается в контекст вашей среды. Например, примените схему принятия решения о выборе языка программирования из набора инструментов сисадмина (*глава 5*) для определения средств управления инфраструктурой.

Выбор и внедрение платформ инфракода оказывают долговременное воздействие на команду, если не на всю организацию. Трудно вывести из эксплуатации технологию, которая все еще используется, — трудно, но возможно. Эта область быстро развивается, и вы рискуете оказаться привязанными к набору инструментов конкретного поставщика, что может оказаться не лучшим вариантом.

Развертывание выбранного вами инструмента зависит от того, принимаете ли вы его для новой среды (новых проектов) или он нужен для решения проблем в существующей среде (текущих проектов). В новых проектах постарайтесь использовать выбранный инструментарий во всех рабочих процессах, для которых он подходит, чтобы стимулировать внедрение инфракода и выявить любые проблемы рабочего процесса. Вы можете решать проблемы, изменяя процессы и инструменты, или прийти к выводу, что нужно пересмотреть область проекта.

При развертывании для текущих проектов определите приоритеты рабочих процессов и постепенно внедряйте новый инструмент. Каждый раз сосредоточивайтесь на одной области, которую можно улучшить. Например, вы можете управлять всей конфигурацией SSH с помощью таких инструментов, как Puppet или Chef, а затем перейти к другим деталям конфигурации веб-сервера, выполняя все это на каком-либо одном сервере. Если автоматизированный процесс устанавливает разумные значения по умолчанию, то команда будет рассматривать это как улучшение, снижающее трудозатраты, и, скорее всего, захочет использовать инструмент для дальнейшей автоматизации. С другой стороны, если люди настраивают системы вручную, они будут считать автоматизацию контрпродуктивной и искать способы обойти и подорвать усилия по автоматизации процессов.

Кроме того, будьте осторожны, принимаясь за слишком сложные проекты, и не пытайтесь решить все проблемы с помощью одного инструмента. Например, если у вас множество платформ, но в основном это Linux, сосредоточьтесь на платформе Linux, прежде чем адаптировать свой инфракод для множества платформ, в том числе Windows. Вы можете обнаружить, что вам требуются совершенно разные рабочие процессы и инструменты, и не будете стараться распространить один инструмент на все платформы.

Часто решение по инфракоду представляет собой многосторонний подход, учитывающий сложность имеющейся инфраструктуры. Многосторонний подход вполне допустим. Вполне разумно использовать Packer для создания образов машин,

Terraform — для неизменяемых эфемерных контейнеров в облаке, а Terraform и Chef — для более долговечных экземпляров. Вы можете разработать целостный подход, который объединит эти инструменты в устойчивое решение.

Заключение

Цель инфракода — дать вам возможность управлять инфраструктурой совместно с командой, используя последовательный, надежный и масштабируемый подход. Широко используемые в настоящее время инструменты инфракода обычно ориентированы на три основных варианта использования: создание образов машин, предоставление ресурсов инфраструктуры и настройку существующей инфраструктуры. Используя эти рекомендации, вы сможете начать использовать инфракод, адаптированный к потребностям вашей организации или команды, технологиям, учитывая все сильные и слабые стороны.

Возможно, у вас есть текущие проекты с проблемными вопросами, которые вы пытаетесь решить, или же вы приступаете к новому проекту и имеете широкие возможности для выбора новейших и лучших инструментов. В обоих случаях необходимо подумать о соответствующих рабочих процессах, в которые вовлечены люди, и найти инструменты, которые наилучшим образом удовлетворят потребности этих рабочих процессов. Используя эти рекомендации, вы сможете подобрать инструментарий для создания инфракода с учетом потребностей, технологий, сильных и слабых сторон вашей команды. В *главе 12* я поделюсь идеями устойчивого управления в большом масштабе, рассмотрев, как используется «инфраструктура как код» и «инфраструктура как данные».

Управление инфраструктурой

Современные вычислительные среды варьируются от управляемых вычислений до оркестраторов контейнеров типа Kubernetes с виртуализированными хранилищами и сетями. Как обсуждалось в *главе 11*, вы можете выбрать различные инструменты для определения этих критически важных ресурсов с помощью инфракода.

Однажды я обнаружила 11 различных активных способов управления отдельными частями конфигурации и развертывания для одного сервиса. Я завершила теневое обновление и попыталась выполнить следующее обновление самостоятельно. Однако этот процесс не был автоматизирован и зависел от созданного разработчиком пакета, который отсутствовал. И хотя я следовала обширному контрольному списку и выполняла различные сценарии оболочки, используя 11 различных систем конфигурации, я обновила только часть системы из тысячи узлов, что привело всю ее в неустойчивое состояние.

Управление системами даже с помощью тщательной сверки с контрольным списком и применением инфракода не является рациональным. В этой главе представлены модели инфраструктуры для улучшения и модернизации управления инфраструктурой и даются рекомендации по началу работы. Вы поймете, как ориентироваться в сложных сценариях инфраструктуры, и сможете постепенно внедрять более современные (и устойчивые) методы.

Инфраструктура как код

Начнем с более известной модели: инфраструктура как код (*англ.* infrastructure as code, IaC). IaC берет проверенные временем методики из области разработки программного обеспечения и применяет их для повышения качества и наглядности управления инфраструктурой.

IaC — это все методики разработки программного обеспечения, применяемые к инфраструктурному коду, а инфраструктурный код — это Ruby, YAML или язык, используемый для описания вашей инфраструктуры.

Отрасль будет внедрять их по мере совершенствования методов разработки программного обеспечения. Существующие подходы включают хранение кода инфраструктуры (инфракода, *англ.* infracode) в системе контроля версий, проверку кода, автоматизированное тестирование и автоматизацию развертывания.

Инфракод

В *главе 11* я представила вашему вниманию инфракод — язык, который понятен как человеку, так и компьютеру, предназначенный для описания аппаратных, программных и сетевых ресурсов и делающий автоматизированное управление ресурсами последовательным, повторяемым и прозрачным.

Управление версиями

В *главе 6* вы познакомились с базовой практикой управления версиями. В системе контроля версий можно сохранять инфракод для обеспечения воспроизводимости, видимости (как изменились ресурсы и когда) и подотчетности (кто внес то или иное изменение).

Проверка кода

Хранение инфракода в системе контроля версий очень удобно, т. к. позволяет получить представление о внесенных изменениях. Как вы внедряете изменения и решаете, хотите ли вы включить их в систему на регулярной основе?

Проверка кода — это процесс просмотра кода коллегами (иногда перед слиянием в главную ветвь репозитория контроля версий). Цели проверки кода следующие:

- проверка реализованного решения;
- проверка того, что проблема была понята и решена;
- обмен информацией о требуемом изменении;
- предоставление возможностей для наставничества в качестве создателя кода или рецензента;
- содействие соблюдению стандартов кодирования и обнаружение ошибок на ранних этапах.

В конечном счете проверка кода — это один из способов сделать ваш код общим кодом команды, а также способ преодоления разногласий. Поэтому подходы к проверке кода в вашей команде будут совершенствоваться по мере того, как вы будете учиться друг у друга.

Автоматизированное тестирование

Как обсуждалось в *главе 7*, тесты повышают уверенность и уменьшают опасения при внесении изменений. Цели тестирования инфракода — помочь вам оценить риск, быстро реагировать на проблемы и восстанавливаться после них, а также улучшить процессы доставки.

Автоматизированное тестирование — единственный способ справиться с потребностями современных систем поддержки работы, обеспечив быстрые темпы развертывания, уведомления об уязвимостях зависимостей и развитие инфраструктуры.

Непрерывная интеграция (англ. continuous integration, CI)

Непрерывная интеграция — это практика автоматического объединения результатов работы нескольких авторов в одну ветвь общего репозитория кода. Плат-

формы CI позволяют командам автоматизировать тестирование и получать быструю обратную связь о качестве потенциальных изменений до слияния кода.

Непрерывное развертывание (англ. continuous deployment, CD)

Непрерывное развертывание — это практика автоматического развертывания проверенных выпусков программного обеспечения. CD-платформы позволяют командам автоматизировать развертывание, что позволяет быстрее получать отзывы и предложения клиентов по поводу новых функций и улучшений.

Автоматизация развертывания

С помощью соответствующих проверочных тестов можно настроить конвейер сборки или непрерывной интеграции/доставки или развертывания (CI/CD), описав методику автоматизированной интеграции кода и автоматизированных сборок. Шаги или этапы в конвейере будут представлять собой отдельные подмножества задач, сгруппированных в соответствии с различными этапами.

На рис. 12.1 показан пример автоматизации развертывания с конвейером сборки, который имеет отдельные этапы и сгруппированные задачи.

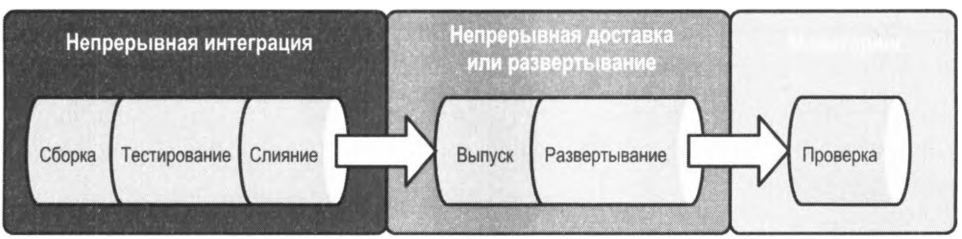

Рис. 12.1. Различные этапы конвейера сборки

В рамках этапа CI существует три шага:

1. Сборка.

   (Проект скомпилирован с учетом предлагаемых изменений).

2. Тестирование.

   (Запуск сценарных тестов для проекта).

3. Слияние.

   (Изменения сливаются в главную ветвь).

В рамках этапа CD существуют два шага:

1. Выпуск.

   Версия проекта публикуется в хранилище артефактов.

2. Развертывание.

   Конкретная версия проекта развертывается в реальной среде, при этом либо обновляется существующая система, либо выполняется развертывание на новых ресурсах.

В рамках этапа мониторинга выполняется последний шаг:

1. Валидация.

   Рабочая среда проверяется на соответствие ожиданиям. На этом последнем этапе осуществляется мониторинг артефактов, развернутых на производстве. Он проводится также для оценки каждой задачи на всех остальных этапах.

У приложений, созданных для работы в системах с бессерверными вычислениями, значительно отличаются требования к архитектуре в части локальной разработки, тестирования и мониторинга. Вы можете использовать любую инфраструктуру CI для внедрения изменений в приложение на различных этапах жизненного цикла разработки.

Кроме того, поскольку системные администраторы не управляют оборудованием, продвижение приложения между средами происходит быстрее. Наконец, бессерверные вычисления опираются на базовые сервисы, предоставляемые облаком, поэтому ваше приложение может работать по-другому, когда эти сервисы меняются.

В реальности ваш конвейер должен моделировать ваши процессы сборки, что может означать отсутствие точно таких же фаз. У вас могут быть разные конвейеры для каждого проекта или специальные конфигурации, которые направляют поток в зависимости от части одного проекта.

Давайте проследим путь запроса на вытягивание после того, как вы напишете код, выполните его линтинг и протестируете в локальной среде разработки (рис. 12.2).

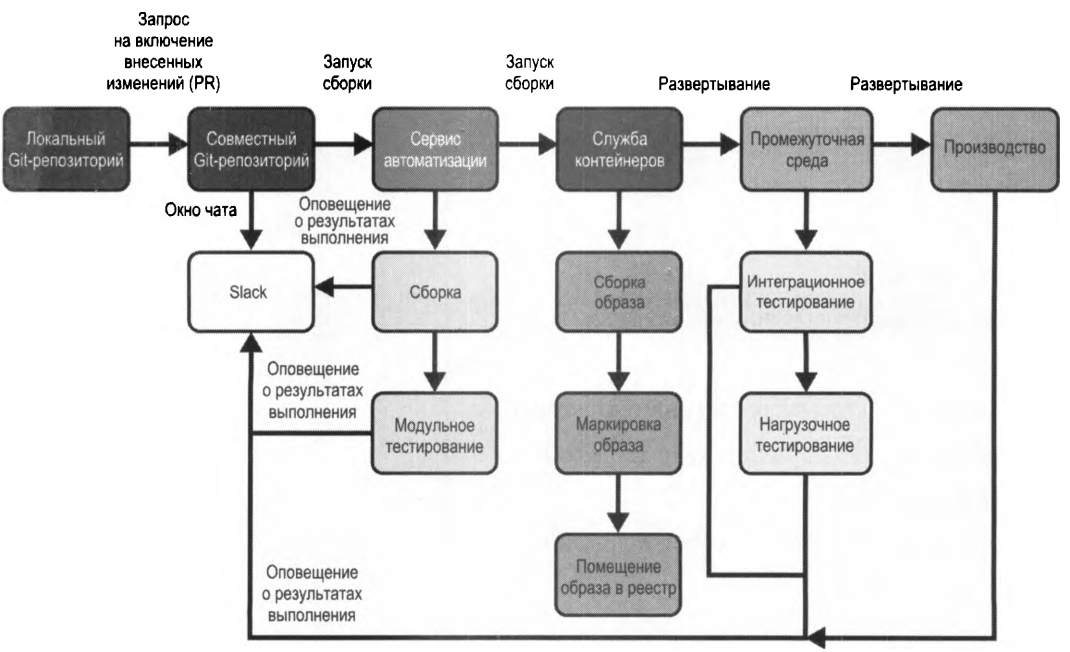

Рис. 12.2. Более сложный процесс сборки, включающий интеграцию изменений кода и развертывание в инфраструктуре

После отправки запроса на вытягивание срабатывает ПО автоматизации развертывания, которое отправляет уведомление в чат группы, канал Slack и запускает сборку программного обеспечения с предложенным вами изменением. На этом этапе или позже, после тестирования, ваша команда может просмотреть код и одобрить или отклонить запрос на вытягивание (*англ.* pull request).

Служба автоматизации запускает модульные тесты и отправляет результаты в Slack. После успешной сборки кода, прошедшего модульные тесты, служба автоматизации запускает службу сборки контейнера, которая собирает, маркирует и размещает образ контейнера в реестре артефактов.

После успешной доставки образа контейнера в реестр запускается создание промежуточной среды, эфемерной среды, эмулирующей инфраструктуру, для начала второго раунда тестирования, который может включать интеграционные, нагрузочные и другие тесты. Наконец, после успешного тестирования образ развертывается в инфраструктуре, где может проводиться дальнейшее оценочное испытание.

В любой момент в случае сбоя подается сигнал в чат группы.

Ниже приведены примеры инструментов, поддерживающих автоматическое тестирование и автоматизацию развертывания:

- GitHub Actions;
- CircleCI;
- Jenkins;
- Azure DevOps;
- Google Cloud Build;
- AWS CodePipeline.

Часто IaC путают с IaaS, но это два разных понятия. Вы можете использовать IaC с локальным оборудованием и облачными вычислительными средами. IaaS — это сервис, который предлагает поставщик облачных решений, и модель предоставления услуг.

Декларативный и императивный инфракод

Инструменты управления инфраструктурой используют два основных подхода к работе с инфракодом: декларативный и императивный. С помощью декларативного инфракода вы описываете желаемое конечное состояние, а за его реализацию отвечает определенный инструмент. С помощью императивного инфракода, напротив, вы определяете процедуру выполнения задачи.

На практике с инструментами, которые поддерживают лишь один из этих подходов, трудно работать, и причиной этого являются ваши ресурсы. Например, декларативный фреймворк может работать для наиболее распространенных развертываний, но очень сильно ограничивает возможности, когда вам нужно выразить то, что должно происходить в других сценариях.

Императивный фреймворк может обеспечить лучшие выразительные возможности для этих пограничных случаев. Тем не менее он оказывается слишком громоздким, когда вам нужно просто развернуть стандартный образ, внеся пару небольших изменений с помощью пользовательских переменных.

Инструменты инфракода, получившие широкое распространение, как правило, балансируют между декларативностью и императивностью, предоставляя простые и гибкие способы реализации многих конвейеров развертывания.

Инфраструктура как данные

В 2013 году Михаэль Дехан (Michael DeHaan), создатель Ansible, написал в блоге O'Reilly Radar «The Rise of Infrastructure as Data» (**https://oreil.ly/n2xUH**): «Инфраструктуру лучше всего моделировать не как код и не в графическом интерфейсе, а используя промежуточный подход, основанный на текстах и определяемый данными». Он ввел термин *«инфраструктура как данные»* (IaD), чтобы расширить концепцию декларативного инфракода.

Итак, следует ли думать об инфраструктуре как о данных или как о коде? Есть что-то привлекательное в том, чтобы моделировать инфраструктуру как данные с помощью модели данных и как код с помощью инфракода и всех соответствующих методов. Это очень похоже на моделирование света, когда мы описываем его как волну или частицу. Вместо того чтобы привязываться к одному или другому варианту, используйте оба.

Вспомните о том, что вы узнали в *главе 3* о важности данных и их ценности для компании. Ваша задача — обеспечить безопасность и доступность данных, а также возможность управления ими. Когда вы создаете модель данных инфраструктуры, вы осознаете стратегическую ценность всех основных ресурсов, необходимых для функционирования вашей системы. Примите во внимание следующие факторы.

- Где хранится ваша модель данных?
- Где содержатся метаданные о вашей модели данных?
- Как вы отслеживаете изменения в модели данных?

Что насчет GitOps?

GitOps — это более новая модель управления инфраструктурой (по сравнению с IaC и IaD), которая возникла в результате управления кластерами Kubernetes в 2017 году. OpenGitOps (**https://opengitops.dev**) — это управляемый сообществом набор стандартов, который описывает рекомендуемые методы и принципы GitOps. Принципы версии 1.0.0 следующие:

- декларативный инфракод;
- версионированное и неизменяемое состояние;
- ресурсы извлекают утвержденную конфигурацию из центрального хранилища;
- ресурсы постоянно сверяют фактическое состояние с необходимым состоянием.

Принципы GitOps не новы. GitOps — это заново упакованные компоненты существующих моделей (инфракод, управление версиями, автоматизация развертывания из модели IaC и неизменяемое состояние из модели IaD). Тем не менее GitOps окажется действенным подходом, если поможет вам внедрить улучшенные методы управления инфраструктурой в вашей организации.

Приступаем к управлению инфраструктурой

Памятуя о моделях IaC и IaD, вы можете подумать о постепенном внедрении практик, которые помогут улучшить и модернизировать способ управления инфраструктурой. В организации, где в данный момент мало или совсем нет соответствующих практик, внедрение и модернизация инфраструктуры — это значительные технические преобразования, которые помимо практик и технологий требуют изменения процессов и обновления навыков. Например, если вы в настоящее время не создаете инфраструктуру и не управляете ею с помощью кода, вам будет очень трудно найти отправную точку. В организации, где соответствующая практика имеется, может быть сложно разобраться со всеми имеющимися системами, не говоря уже о возможностях вносить улучшения.

Чтобы добиться успеха в этом деле, нужно, чтобы ваша команда и заинтересованные стороны имели единое мнение о тех сторонах управления инфраструктурой, которые вы хотите улучшить. Если ваши взгляды не совпадают, вам будет трудно завершить проект.

Кроме того, убедитесь, что ваш проект не направлен на автоматизацию всего подряд. Наоборот, он должен быть привязан к конкретной цели. Автоматизация всего — это многоквартальный и, вероятно, даже многолетний проект, что означает удлинение сроков, в которые можно добиться желаемых результатов. Ниже показаны примерные проекты по управлению инфраструктурой с правильно выбранной областью охвата:

- повышение скорости развертывания для сред разработки;
- помощь членам команды, работающей с географически распределенной инфраструктурой, наладить сотрудничество с учетом разницы в часовых поясах. Для этого задачи по конфигурации системы, выполняемые в реальном времени, переводятся в запланированные изменения кода, выполняемые в удобное время;
- оптимизация процесса приема новых сотрудников, упрощающая их переход на почасовую работу с некоторыми проектами и содействующая сотрудничеству между различными командами;
- рассмотрение проблемных вопросов, возникающих во время дежурств, и того, как автоматизация работы может сделать эти дежурства менее напряженными.

Определение проблемы — ключевой момент. Инфракод не является самоцелью, даже если ваше руководство так считает и пытается подойти к работе с этой меркой.

Как только вы поставили цель, разбейте ее на более мелкие задачи, поддающиеся измерению, или ключевые точки: управление версиями, проверка кода, автоматизированное тестирование и автоматизация развертывания вашего инфракода, используя при необходимости все имеющиеся средства.

Начните с управления версиями

В команде, которая уже внедрила программное обеспечение для управления версиями, решите, где будет располагаться инфракод (в проекте или в выделен-

ном для него репозитории кода). Если ваша команда еще не использует управление версиями, обратитесь к *главе 6* и узнайте больше о том, как начать использовать управление версиями.

Внедрите процессы проверки кода

В команде, которая уже выполняет проверку кода, оцените и задокументируйте текущий процесс.

Определите или выберите инструмент(ы) управления инфраструктурой

В *главе 11* я представила три модели инфракода, чтобы помочь вам определить и оценить используемые инструменты и выбрать дополнительные средства управления инфраструктурой.

Внедрение единых точек управления элементами инфраструктуры

Устраните области, где несколько инструментов обновляют одни и те же ресурсы. Конфликты при обновлении вызовут досаду, разочарование и лишние оповещения.

Не пренебрегайте сотрудничеством

Для долгосрочного успеха проекта по управлению инфраструктурой необходимо учитывать рабочие процессы, которые будет стимулироваться инструментом, и то, как они изменят динамику вашей команды. Как только ваша команда внедряет инструмент автоматизации инфраструктуры, он вносит в систему соответствующие изменения, которые будут проявляться в будущем. Поэтому каждый член команды должен понимать принцип работы инструмента, чтобы ежедневно использовать его.

Если в проект по управлению инфраструктурой будет вовлечена лишь часть команды, это приведет к появлению узкого места: оставшиеся работники будут обращаться к ним, чтобы внести изменения, которые раньше они выполняли сами. Это никому не нравится. При внедрении решения обязательно соберите всю команду для обсуждения, проведите соответствующие презентации, чтобы людям было проще работать в дальнейшем.

Убедитесь, что нет единых точек отказа

Помните, что инфраструктура включает в себя программное обеспечение, которое будет развернуто. Поэтому убедитесь, что каждый специалист понимает определение приложения, а также когда и для чего оно запускается. Например, облачные приложения, использующие бессерверную инфраструктуру, должны быть определены в системе управления исходными версиями и развернуты автоматически у поставщика облачных решений. Создание стандартного каркаса проекта покажет инженерам, что начинать нужно с инкапсуляции определенных моделей безопасности, и поможет предотвратить непродуманное и бесконтрольное создание ресурсов.

Определите, какие навыки нужны для достижения успеха

Даже самым опытным из нас требуется определенное обучение, чтобы успешно внедрить какую-либо технологию. Если ранее вы не использовали систему

управления версиями, прежде всего нужно получить навыки работы с ней, уметь создавать учетные записи в той системе управления версиями, которая используется в вашей организации. Например, для Terraform вам может потребоваться обучение языку HashiCorp Configuration Language (HCL) (**https://oreil.ly/6sgXp**) наряду с обучением внедрению Terraform в вашей организации.

Добивайтесь качества с помощью автоматизированных тестов

Подумайте о том, как вы используете или собираетесь использовать инфракод. Вы можете включить тестирование в свои первоначальные планы или добавить его после для существующего инфракода. Во многих случаях процесс автоматизации тестов инфракода (инфратестов) сам по себе уже является масштабным мероприятием.

Советы по проверке кода

Проверка чужой работы — задача не из легких. Когда вы достигаете особого состояния потока при общении с другим человеком, то передаете друг другу больше информации, чем несут в себе слова языка, как таковые. Но вне этого состояния слова могут восприниматься поразному, и не обязательно их обилие поможет решить проблему.

Вот несколько рекомендаций, основанных на моем многолетнем опыте.

- Программный код не обязательно должен быть совершенным. Прежде чем оценивать чей-то код, необходимо получить некоторое представление о процессе проверки. Возможно, вы будете использовать какой-то вариант с общепринятыми комментариями (**https://oreil.ly/P3gM1**), что позволит установить единый и привычный для всех формат. При проверке вы можете ставить метки, показывающие, что вы против внесения предлагаемых изменений или, наоборот, одобряете их. Вы сможете поделиться своими мыслями, не замедляя прогресса. Вполне нормально интегрировать код, который улучшает состояние системы, даже если он не является на 100% совершенным.
- Не забывайте просматривать комментарии. Комментарии в программном коде должны объяснять, для чего нужен код, а не что он делает (за исключением сложных вещей, таких как регулярные выражения).
- Не забывайте хвалить за хорошую работу. При проверках часто уделяется внимание только проблемным моментам. Выражайте признательность за использование рекомендуемых методов.
- Не будьте слишком категоричны. Не стоит использовать такие слова, как «всегда» и «никогда», потому что обстоятельства меняются и ваше суждение может оказаться неверным в новом контексте.
- Самое главное — будьте доброжелательны. Я не говорю, что вы не должны указывать на проблемный код, доброта состоит не в том, чтобы замалчивать правду. Они должны знать, как обстоят дела. Просто при возникновении проблем потратьте лишнюю минуту, чтобы корректно сформулировать свою мысль.

Если вы серьезно обеспокоены по поводу кода, может быть, стоит поговорить с человеком с глазу на глаз в неофициальной обстановке или отправить личное сообщение в Slack, чтобы исправить положение дел. Иногда люди воспринимают письменные проверки как более жесткие из-за того, что нет контекстных подсказок. При более индивидуальном подходе — например, когда вы совместно работаете над проверенным проектом — члены вашей команды могут учиться у вас и не будут трактовать ваши намерения неправильно.

Как было сказано в *главе 7*, сложность написания тестов для инфракода заключается в том, что зачастую тестировать работающую инфраструктурную платформу проще, чем код. Подумайте, проверяете ли вы сам код или тестируете работу программного обеспечения для управления инфраструктурой. Если только это не система собственной разработки, считайте, что программное обеспечение делает именно то, что и должно. Даже если вы работаете с собственной системой конфигурации, тестируйте эту платформу в ее Git-проекте отдельно от вашего проекта инфракода.

Давайте вернемся к четырем типам тестов, которые мы рассматривали в *главе 7*: линтинг, модульные, интеграционные и сквозные тесты, в частности с помощью инфракода.

Линтинг

Из-за природы линтеров и развития рекомендуемых методик версии линтеров особенно чувствительны к изменениям. Если у одного человека в системе установлена одна версия линтера, а у его соседа — другая, то линтер будет влиять на их код, вызывая ненужные конфликты при работе над одним и тем же проектом. Как и в случае с другими инструментами в среде, убедитесь, что все используют одну и ту же версию программного обеспечения для линтинга.

Когда ваш линтер возвращает ошибку, не всегда нужно вносить изменения в код. Вместо этого изучите проблемные вопросы и определите, реальна ли проблема или требуется просто настроить правила линтинга для проекта.

Написание модульных тестов

При использовании модульных тестов для инфракода, как правило, существует определенный пакет, который предназначен для тестирования используемой вами платформы. Например, у Chef есть Chefspec, а у Puppet — rspec-puppet. Инфракод может усложниться, если у вас есть специфические настройки — например, различные операционные системы, вычислительные экземпляры, тестовая или производственная среда, в которой существует система. Полезные модульные тесты будут проверять те входные данные, которые изменяют работу кода, чтобы вы могли иметь детерминированные выходные данные. Они помогают будущим сисадминам изменять ваш код и замечать проблемы на ранней стадии.

Как правило, для очень простого инфракода не требуются модульные тесты, потому что мы можем предположить, что система управления инфраструктурой управляет отдельными блоками ресурсов так, как задумано. Я знаю, этот совет обескураживает, т. к. противоречит принципу пирамиды тестирования, но дело в том, что в отношении инфракода только сложные паттерны требуют тестирования на объектном уровне. Полезно регулярно оценивать пользу тестов, поскольку вы должны их поддерживать. Неработоспособные тесты могут помешать совместной работе людей!

Написание интеграционных тестов

Как было сказано в *главе 7*, организации по-разному трактуют интеграционные тесты. Интеграционные тесты могут быть узкими (тестирование двух компонентов) или широкими (тестирование множества компонентов). Прежде чем внедрять тесты для инфракода, согласуйте цель их проведения. Например, при наличии инфракода для настройки стороннего сервиса вы будете тестировать активную конфигурацию или имитировать подключение к службе, предполагая, что ей предстоит работать в различных средах?

Подумайте о сценариях, содержащих системные команды, отклик на которые различается в зависимости от внешних факторов. Например, при интеграционном тестировании вы, возможно, захотите контролировать результат выполнения системной команды, потому что вы тестируете не ее, а написанный вами сценарий, включающий эту команду. Имитация (мокинг) — это важная техника, которую вы можете использовать для обеспечения воспроизводимости и целенаправленного тестирования.

Написание сквозных тестов

Сквозные тесты позволяют проверить, что проект функционирует так, как ожидалось от начала и до конца в эфемерной среде, подобной производственной. Это означает, что их проведение связано с довольно большими затратами. Поэтому существует опасность, что если система CI не наведет порядок в инфраструктуре тестирования, вы будете тратить деньги на ненужные ресурсы.

Управление инфраструктурой тестирования: укрепляем площадку для тестирования

Автор — Крис Деверс (Chris Devers)

Что касается поддержания и расширения линейки продуктов в течение многих лет, я обнаружил, что тестовая среда организации (тесты, фреймворк и инфраструктура тестирования) не менее важна, чем системное обслуживание продукта. Что хорошо, в среде тестирования моей команды большинство проблем обнаруживается раньше, чем их замечают клиенты, поэтому мы можем раньше выпускать исправления и опробовать новые идеи. Так, благодаря вспомогательным инструментам для проведения интеграционных тестов моя команда обратила внимание на новые способы развертывания и управления производственными системами. В результате наши продукты стали лучше работать, принося выгоды нынешним и будущим клиентам.

Проблема возникает, когда становится трудно поддерживать саму среду тестирования. Та же самая среда тестирования, которая помогает выявлять проблемы на ранней стадии и дает нам возможность экспериментировать с новыми идеями, также страдает от этих проблем:

Проблема: «Трагедия ресурсов общего пользования»

Это преимущество для многих разработчиков, однако ни люди в отдельности, ни команды не несут на самом деле ответственности за поддержку и качество самой среды. Поэтому она страдает и становится трудноподдерживаемой, т. к. все больше людей начинают активно ее использовать.

Проблема: «Кто наблюдает за наблюдателями?»

По сообщению о падении теста бывает трудно определить корень проблемы: это может быть ошибка самого теста при попытке реализовать сценарий, который не может про-

изойти в производственной среде, или дефект программного обеспечения для производства, и тогда указанный сценарий вполне оправдан.

Проблема: «Мальчик, который кричал о волках»

Для проблем, которые уже решены, активируются оповещения, потому что результаты тестирования интерпретируются неправильно и разработчикам приходится перепроверять результаты, чтобы понять, какие проблемы реальны, а какие являются ложными или относятся к отголоскам более ранних дефектов.

Избежать этой ловушки можно, потратив время и средства регулярного бюджета на поддержание в актуальном состоянии документации по используемым системам, включая ваши тестовые системы. В приложении В приведена информация о том, как распределить время и ресурсы при устранении четырех типов сбоев, которые могут возникнуть в вашей тестовой среде, чтобы поддерживать ее в исправном состоянии.

Следуйте этим рекомендациям, и вы сможете внедрить практики IaC и IaD, отвечающие потребностям, технологиям, сильным и слабым сторонам вашей организации или команды, более эффективно управлять инфраструктурой и более эффективно сотрудничать.

Заключение

В эпоху, когда ресурсами управляют с помощью программного обеспечения, практики IaC и IaD — это инструментарий для создания, тестирования, развертывания и управления инфраструктурой ваших систем. Такие практики, как управление версиями, проверка кода и автоматизированное тестирование, являются стандартом для команд разработчиков программного обеспечения на протяжении многих лет. И эти практики полезно внедрить для управления вашей инфраструктурой.

В организациях, где данные практики отсутствуют, внедрять модели IaC и IaD следует постепенно. Ищите конкретные области, где передовые методы помогут вашей команде стать более эффективной — быстрее достигать прогресса и улучшений в этих направлениях.

Дополнительные ресурсы

Узнайте больше о практиках IaC из обновленной версии книги Кифа Морриса (Kief Morris) «Infrastructure as Code» («Инфраструктура как код», O'Reilly, **https://oreil.ly/xxTX8**).

Обеспечение безопасности инфраструктуры

Возможно, в вашей организации есть эксперты по безопасности, которые занимаются защитой инфраструктуры. А может быть, нет ни одной должности, в названии которой фигурирует слово «безопасность». Случается и так, что опыт специалиста минимален либо вовсе отсутствует. Независимо от того, есть ли у вас возможность сотрудничать с другими или нужно самостоятельно разобраться, что делать, вы можете повысить защиту своей инфраструктуры, приняв философию безопасности.

В идеале обеспечение безопасности инфраструктуры начинается с момента планирования и создания необходимых для системы ресурсов. Но как понять, с чего начать, если речь идет об уже существующей инфраструктуре? Глубокая защита предписывает применять методы обеспечения безопасности на разных уровнях для предотвращения ущерба вашей инфраструктуре, но это не значит, что можно делать все одновременно. Тем не менее, приняв философию безопасности, вы можете улучшить надежность, стабильность и общую работоспособность конкретных систем, которыми управляете, включая приложения, инструменты и сервисы (т. е. обеспечить желательные качества ваших «кондитерских изделий»).

В этой главе я моделирую подход к обеспечению безопасности инфраструктуры. Прежде всего, оцените векторы атак для общего конвейера сборки, чтобы найти уязвимости, и применяйте различные подходы для концентрации усилий по снижению рисков в соответствующих направлениях (например, управление доступом к идентификационным данным и секретам, защита вычислительных систем и сетей), чтобы устранить наиболее частые атаки. В заключение я предлагаю ряд рекомендаций по управлению инфраструктурой. Эта глава ни в коем случае не является исчерпывающим руководством по методам защиты инфраструктуры. В ней затрагиваются лишь самые основы. Однако она предлагает вам задуматься над тем, как разбить критически важную работу по этим вопросам на небольшие достижимые цели.

Изучите *главу 8*, «Безопасность инфраструктуры» для получения более подробной информации о базовых методах обеспечения безопасности.

Оценка векторов атак

Хотя информация о том, как полностью оценить векторы атак в среде, выходит далеко за рамки этой книги, повышение безопасности инфраструктуры начинается с размышлений о потенциальных точках входа в ресурсы вашей организации.

Рис. 13.1. Проверка процесса сборки на наличие уязвимых мест

Давайте проверим общий конвейер сборки из *главы 12* на наличие уязвимых мест, как показано на рис. 13.1.

В каждой точке существуют различные векторы атак. Вот лишь некоторые из них:

- система управления версиями;
- ресурсы в среде сборки;
- платформа сборки;
- программные пакеты и образы контейнеров в репозиториях артефактов;
- ресурсы в вашей производственной и других средах;
- данные обо всей вашей инфраструктуре и о том, что происходит.

Риски для ресурсов на ранних стадиях процесса могут влиять на все ресурсы, созданные позже. Ниже перечислены некоторые распространенные атаки на эти ресурсы:

1. Компрометация учетных данных (например, взлом сайта и раскрытие имен пользователей и паролей, а также пользователей, которые используют одно и то же имя и пароль на нескольких сайтах, включая взломанный).
2. Слабозащищенные учетные данные (например, пароль 123456, который легко угадать).
3. Неправильная конфигурация (например, конфигурация сервиса, которая не требует аутентификации и авторизации).
4. Уязвимости в программных пакетах и образах контейнеров (например, образы *docker* с еще не установленными исправлениями).

5. Операционная система и программное обеспечение без установленных исправлений (например, ОС была установлена, но не обновлена до последней исправленной версии).

Как улучшить средства обеспечения безопасности, чтобы решить эти проблемы? С помощью инфракода можно свести к минимуму ошибки и неправильную конфигурацию, отделить архитектуру с моделью «нулевого доверия» и уменьшить подверженность риску определенных ресурсов за счет ограничения доступа к ним извне. Как и в случае с IaC и IaD, постепенно внедряйте практики, повышающие безопасность управления инфраструктурой. Давайте рассмотрим различные подходы, помогающие сосредоточить ваше внимание на определенных аспектах:

- управление идентификацией и доступом (направлено на решение проблем 1, 2);
- управление секретами (направлено на решение проблем 1, 2);
- обеспечение безопасности вашей вычислительной среды (направлено на решение проблем 3–5);
- обеспечение безопасности вашей сети (направлено на решение проблемы 4).

Управление идентификацией и доступом

В зависимости от того, сколько времени вы администрируете системы, какие ОС имеются в вашей среде и какие размещенные сервисы используются, вы могли управлять пользователями и доступом самыми разными способами, включая следующие:

- синхронизация `/etc/passwd` и предотвращение дублирования идентификаторов пользователей;
- управление сервером LDAP, службами Kerberos или Active Directory;
- управление идентификационными данными в файле `htpasswd`;
- выполнение сценариев SQL для добавления пользователей и назначения ролей для баз данных MySQL.

Некоторые из этих способов по-прежнему допустимы для управления доступом пользователей. Однако новые методы и технологии помогают обеспечить автоматизацию, прозрачность и соответствие требованиям.

Как следует управлять доступом к системе?

Управление идентификацией и доступом (*англ.* identity and access management, IAM) — это способ настройки ролей и привилегий для пользователей, групп и сервисов, а также базовая технология и процессы, поддерживающие выделение и отзыв привилегий. IAM включает в себя три базовых компонента:

Аутентификация

Пользователи действительно являются теми, за кого себя выдают.

Авторизация

Пользователь имеет право на выполнение определенных действий.

Регистрация действий

Действия пользователя записываются в журнал.

Помимо ваших собственных решений, которыми вы управляете, внешние службы также внедряют IAM — с использованием различной терминологии и концепций. IAM может быть по-разному реализовано в разных сервисах, предлагаемых тем или иным поставщиком (например, аутентификация и авторизация с использованием вычислительного экземпляра или базы данных). Вам нужно будет ознакомиться с документацией по сервису, который вы планируете использовать, чтобы понять, как выполнять аутентификацию, авторизацию и регистрировать действия пользователей. Если вы начинаете работать на новой должности, перехóдите на другое облако или используете новый веб-сервис, не ждите, что идентификация будет везде реализована одинаково.

Примерами поставщиков услуг и их служб идентификации являются Amazon AWS Identity and Access Management, Google GCP Cloud Identity and Identity and Access Management и Microsoft Azure Active Directory. Из-за различий между сервисами и провайдерами вы можете ненамеренно ослабить свою систему, выбрав неправильные настройки.

Далее перечислены некоторые примеры изменений в методах идентификации, применяемые в современной инфраструктуре:

- вместо единственного фактора аутентификации, такого как пароль для входа в систему, используйте многофакторную аутентификацию (multifactor authentication, MFA), которая требует предоставления двух или более доказательств. В случае с MFA люди подтверждают, что они именно те, кем представились, с помощью двух «ключей»: один они знают (например, пароль или PIN-код), а вторым владеют (например, маркер безопасности или карта);
- вместо синхронизации и централизации `/etc/passwd` для множества систем Unix или привязки их к LDAP-каталогу вы можете положиться на конфигурационный инфракод, который будет гарантировать, что пользователи имеют учетные записи только на тех системах, которые им нужны.

IAM может быть сложным. Например, в гибридном сценарии, когда вы управляете идентификационными данными с помощью корпоративного каталога пользователей отдельно от поставщика услуг, вам, возможно, придется управлять отношениями доверия и внедрять комплексное управление идентификацией для доступа в различные сервисы. Это позволяет совместно использовать методы аутентификации в разных сервисах, чтобы пользователи могли применять имеющиеся учетные данные.

Сложность также увеличивается из-за необходимости внедрения IAM в различных областях: корпоративные идентификационные данные в рамках организации, удостоверения служб для взаимодействия приложений и идентификационные данные потребителей для доступа к услугам, ориентированным на клиентов.

Инструменты для IAM, которые вы используете для различных служб, могут стать сложнее. Использование инфракода позволяет создавать последовательные, повторяемые и тестируемые конфигурации. Вам также понадобятся прозрачные процессы, особенно в отношении приема и увольнения сотрудников, для настройки всего, что не интегрировано с автоматизацией.

Создание более удобных для разработчиков способов управления предоставлением ресурсов потребует дополнительных защитных механизмов и проверок. Например, одной из наиболее распространенных ошибок в конфигурации доступа в таких службах объектного хранения, как AWS S3, является настройка полного анонимного доступа к контейнеру или разрешение любому желающему выполнять чтение и запись в контейнер. Почему это происходит? Многие руководства иллюстрируют концепции, лежащие в основе сервисов, заставляя разработчиков сразу же открывать доступ, чтобы было легче сосредоточиться на изучении сервиса, и не объясняют, что делают эти конфигурации. К сожалению, эти шаблоны затем копируются в живую среду, из-за чего возникают уязвимости. Предоставление тестовых фрагментов инфракода, отражающих лучшие практики, облегчит работу другим и поможет сохранить единообразие настроек в вашей организации.

Возможно, вам придется провести анализ проблем, связанных с инфраструктурой, и научить инженеров в вашей организации пользоваться определенной технологией. Предположим, вы захотите убедиться, что для всех учетных записей пользователей включена MFA. Вы можете настроить автоматическую проверку учетных записей на отсутствие MFA и напоминать пользователям, чтобы они либо настроили многофакторную авторизацию, либо деактивировали свою учетную запись.

Вы можете использовать свои инструменты инфракода для отслеживания, аудита и изменения корпоративных и служебных идентификаторов в ваших системах в рамках процесса предоставления доступа. Использование инструментов инфракода гарантирует, что настройки, которые вы задаете, применяются единообразно, а обновления рассылаются по мере необходимости.

Кто должен иметь доступ к вашей системе?

Как только вы разберетесь с управлением доступом к различным системам, необходимо установить, кто должен иметь доступ к вашей системе. При изучении документации по приложениям или услугам часто можно найти рекомендации, связанные с работой систем, — например, какие учетные записи и разрешения необходимы. Вы также должны задать себе следующие вопросы.

- Требуется ли более высокий уровень привилегий для индивидуальных учетных записей или учетных записей служб?
- Требуется ли ограничение доступа по времени?
- Требуются ли зарегистрированным пользователям, которые вошли в систему, иные функции, нежели случайным, анонимным пользователям?

Вы можете минимизировать возможный вред для вашей системы, применяя принципы минимальных привилегий и разделения обязанностей при предоставлении

доступа к вашим системам. Эти принципы гарантируют, что пользователю или компоненту предоставляется минимально необходимый доступ и полномочия и они не имеют возможности использовать учетные записи `root` или администратора. Другими словами, при взломе учетной записи вред ограничивается компонентами системы, к которым эта учетная запись имеет доступ.

Вы можете проверить, к каким программным интерфейсам (*англ.* application programming interface, API) открыт доступ. API часто считаются сферой деятельности разработчиков, но они оказываются критически важным вектором атаки. Большинство современных веб-приложений в той или иной форме предоставляют API пользователям. Например, в сервисах, размещенных на серверах, вы осуществляете настройку и управляете доступом ко всем своим системам и данным через API-шлюз поставщика. Проверьте, к каким ресурсам ваш сервис по умолчанию предоставляет открытый доступ.

IAM и ведение журнала аналогичны дверным замкам, камерам видеонаблюдения и другим физическим средствам для управления доступом в локальном центре обработки данных или серверном шкафу. Инфракод является насущной необходимостью, т. к. гарантирует, что все «двери» остаются запертыми, а доступ контролируется.

Управление секретными данными

Инженеры хотят выполнить работу как можно быстрее и с минимальным количеством препятствий, доверяя порой таким приложениям, которые не гарантируют соблюдение конфиденциальности, или случайно добавляя их в в систему управления версиями. Однако часто вы не получаете полного представления обо всех рисках, связанных с раскрытием секретных данных, поскольку они могут быть встроены в код, а различные сервисы требуют различных процессов. Секретные данные — это пароли, сертификаты mTLS, токены носителя и ключи API.

Эти данные подвержены проблеме начальной загрузки: как получить доступ к определенному ресурсу? Если нужен пароль, то как его получить? Я помню, как в начале карьеры мне вручили записку с подробными инструкциями и сообщили, что крайне важно запомнить пароль, а затем уничтожить записку. Было проблематично сбрасывать пароли `root` и администратора, когда кто-то покидал команду, и при этом гарантировать, что у всех остальных работников есть новый пароль.

В современных средах вам нужно хранить в тайне не только пароли хостов. Использование инфракода позволит шире внедрять передовые практики управления секретными данными и отслеживать успехи. Однако инфракод создает и новые проблемы при управлении секретами, поскольку инструментам инфракода требуется доступ к этим данным. Давайте рассмотрим эти проблемы, чтобы вы могли лучше управлять секретами.

Менеджеры паролей и программное обеспечение для управления секретными данными

Использовать секретные данные или иметь доступ к ним должны то люди, то автоматизированные процессы, то и те и другие. Эти модели доступа диктуют, какой тип интерфейса лучше выбрать, поэтому программное обеспечение для управления секретами обычно адаптировано под одну из задач.

Когда основной задачей является интерактивное использование людьми, программное обеспечение для управления секретными данными обычно называется *менеджером паролей* или *приложением для управления привилегированным доступом*. С помощью менеджера паролей вы можете генерировать надежные, уникальные пароли и хранить их. Менеджеры паролей снижают риск повторного использования паролей и позволяют делиться секретными данными в рамках команды, не прибегая к таким небезопасным методам, как запись, отправка через сервисы совместной работы или электронную почту. Некоторые известные менеджеры паролей приведены ниже.

- 1Password (**https://1password.com**);
- LastPass (**https://www.lastpass.com**);
- KeePass (**https://keepass.info**);
- Bitwarden (**https://bitwarden.com**);
- pass (**https://www.passwordstore.org**).

Программное обеспечение для управления секретами для других задач представляет собой базу данных «ключ-значение» с функциями аутентификации и аудита. Поставщики добавляют ценность своим решениям по управлению секретными данными, интегрируя их с различными программными экологическими системами или поддерживая определенные модели использования. Основное назначение платформы управления секретами — отделить их хранение от программного кода или конфигурации, которые являются потребителями этих данных. Помимо возможности поддерживать такое разделение, при оценке программного обеспечения для управления секретными данными следует принимать в расчет следующие аспекты.

Централизация

Все секретные данные хранятся в одном месте, что снижает риск их утечки по сравнению с тем, когда они хранятся в коде или когда об их существовании забывают.

Отзыв

Маркировка секретных данных как недействительных и более не заслуживающих доверия.

Ротация

Обновление учетных данных для удостоверения. Оно может включать версионирование секретных данных, позволяющее постепенно развертывать новые секретные идентификационные данные, чтобы не создавать хрупких взаимозависимостей между секретными данными и приложениями.

Изоляция

Возможность присваивать секретные данные отдельным лицам или ролям, чтобы предоставить наименьшее количество привилегий. Одному приложению не нужен полный доступ ко всей секретной информации проекта.

Инвентаризация

Визуальный контроль хранимых секретных данных (без доступа к ним) для устранения бесконтрольного роста этих данных.

Хранение

Визуальный контроль над тем, как и где хранятся и реплицируются секреты, и возможность настройки данных правил.

Контроль

Информация о взаимодействии с секретными данными регистрируется и отслеживается.

Шифрование

Секретные данные шифруются при хранении и во время передачи. Эта информация не должна записываться на диск или передаваться по сети в незашифрованном виде.

Генерация

Создание новых секретных идентификационных данных.

Поддержка интеграции

Удобство использования с другими сервисами и возможность интеграции с вашим программным обеспечением.

Надежность

Доступ к секретным данным должен быть надежным. Как будут работать сервисы и системы, если хранилище секретов не работает?

Защита секретов и наблюдение за их использованием

Наблюдение за доступом к учетным данным и другим секретам, а также их использованием является неотъемлемой частью стратегии глубокой защиты. Существует много каналов утечки секретной информации, поэтому важно иметь механизмы для обнаружения подобных событий и реагирования на них. Доступ к секретным данным можно получить, например, проанализировав историю команд, журналы отладки и использование переменных среды. Последние заслуживают особого внимания, поскольку они доступны процессам, и секреты могут быть раскрыты через список процессов при отсутствии журналов аудита для отслеживания воздействия.

В 2020 году инженеры компании Ubiquiti обнаружили незаконную деятельность (**https://oreil.ly/g42C5**) в сети, связанную с неправомерным использованием учетных данных ИТ-администратора, которые хранились в LastPass. Отсутствие протоколирования не позволило отследить, что было сделано злоумышленниками, пока

они имели доступ к системам. Даже если вы полагаете, что любой, кто имеет доступ к вашей системе, должен в любое время иметь доступ ко всем секретным данным, подумайте о риске, исходящем от сторонних сервисов, получающих журналы, где могут присутствовать секретные данные в незашифрованном виде.

Рассмотрим перемещение секретных данных, внесенных в журнал при возникновении проблемы. Например, они могут быть приняты платформой Splunk, включены в оповещение PagerDuty и отправлены по электронной почте и через службы обмена текстовыми сообщениями.

Вы должны знать, какие системы доступны (и должны быть доступны!), и уметь обнаруживать подозрительное использование учетных данных (с нетипичных IP-адресов или в неподходящее время). Многие приложения и сервисы используют машинное обучение, чтобы обнаруживать аномалии учетных записей, обращая ваше внимание на неожиданное поведение.

Чтобы понять, насколько обширна и серьезна утечка данных, необходима комплексная и четкая стратегия управления данными в журналах аудита. Используйте разделение привилегий, чтобы мероприятия по администрированию системы были отделены от действий, связанных с журналами аудита.

В прежних средах вам приходилось беспокоиться об управлении доступом пользователей. Теперь следует учитывать и доступ к сервисам. Инструменты и методы развиваются, но при управлении секретами по-прежнему возникают проблемы, особенно это характерно для межкомпьютерной коммуникации. Часто у вас есть неполное понимание рисков, связанных с раскрытием секретов, поскольку секретные данные могут содержаться в программном коде, а различные сервисы требуют различных процессов. Журналы регистрации доступа из программного обеспечения для управления секретами могут помочь в этом вопросе: сервисы, которые получают доступ к секретным данным, будут иметь определенный шаблон, что поможет сделать доступ, не соответствующий нормам, более заметным. Кроме того, можно проанализировать, какие службы или приложения не используют выбранное программное обеспечение для управления секретами и несут в себе потенциальный риск. Инфракод поможет закрыть эти пробелы.

Обеспечение безопасности вычислительной среды

Защита вычислительной среды минимизирует векторы атак, направленные на ваши системы, обеспечивая конфиденциальность, целостность и доступность ОС, сервисов и инструментов ваших систем. Усилия по обеспечению безопасности вычислительной инфраструктуры зависят от типа сервисов, с которыми вы работаете. Например, при использовании управляемых услуг вы принимаете на себя часть расходов, связанных с ответственностью поставщика услуг за обеспечение безопасности инфраструктуры.

Что касается виртуальных машин и контейнеров, которые вы решили создать и запустить, сервис-провайдер обеспечивает только их физическую безопасность и

предоставляет операционную среду (гипервизор или хост контейнера) для рабочей нагрузки. Управление конфигурацией инфраструктуры с помощью инфракода — это более ответственный подход к обеспечению безопасности ваших систем.

Операционные системы и приложения часто по умолчанию используют открытые конфигурации, отдавая предпочтение простоте использования, а не безопасности. Так что вы уменьшите число возможных направлений атак, если защитите конфигурацию сервисов, требующих управления ОС и приложениями. Это обычное требование регуляторов, входящее во многие регламенты и стандарты, включая стандарт защиты информации в индустрии платёжных карт (Payment Card Industry Data Security Standard, PCI-DSS), ISO 27001, закон Сарбейнса–Оксли в США (Sarbanes–Oxley Act, SOX) и федеральный закон об управлении информационной безопасностью (Federal Information Security Management Act, FISMA).

Дополнительные указания вы можете найти на следующих ресурсах:

- Справочное руководство (**https://oreil.ly/4Ises**) Центра интернет-безопасности (Center for Internet Security, CIS);
- Руководства по технической реализации мер обеспечения безопасности (Security Technical Implementation Guides, STIG) (**https://oreil.ly/4aLd3**).

Эти рецензируемые стандарты доступны для многих ОС, популярных приложений и сетевых устройств. В них содержатся подробные инструкции по настройке всевозможных параметров, связанных с безопасностью, некоторые из которых могут оказаться неподходящими для вашей ситуации. Изучайте стандарты и внедряйте рекомендации, которые имеют смысл для вашей отрасли и среды.

Документируйте решения, связанные с отклонением от стандартов

Автор — Крис Деверс (Chris Devers)

В моей команде появился новый руководитель, который хотел, чтобы мы обновили предложения нашей компании в соответствии с рекомендациями CIS. Одна из рекомендаций касалась того, чтобы для Linux-хостов выполнялись определенные требования по размещению стандартных каталогов верхнего уровня на разных разделах. Новый менеджер не знал, что мы обнаружили гораздо более распространенный источник нестабильности: когда один раздел заполнялся, журналы переставали обновляться, а базы данных переставали добавлять новые записи. К тому времени мы уже взвесили все плюсы и минусы этой схемы размещения разделов по сравнению с объединением их в менее замысловатую схему и решили, что объединение большинства разделов будет чистым выигрышем.

Для нас было важно понять, что стоит за стандартными отраслевыми рекомендациями, оценить наши системы и соответствие стандартам, а также решить, стоит ли принимать рекомендации, которые могут принести больше вреда, чем пользы.

Если вы решили отойти от рекомендаций, подумайте, решает ли альтернативный подход проблему, которая может быть для вас более критичной, чем те, для решения которых предназначен стандарт, и задокументируйте это решение в политике и итоговых результатах ее выполнения.

Еще одной важной частью управления безопасностью вычислительной инфраструктуры является обновление ОС, установленных пакетов и приложений. К сожалению, установка обновлений может оказаться сложной по многим причинам, включая зависимость приложений от конкретных версий ОС или других пакетов,

нерациональную практику развертывания или опасения по поводу совместимости и стабильной работы.

Инфракод помогает справиться и с этими проблемами. Зависимости приложений могут быть документированы и учтены в инфракоде. Инфракод, по своей природе содействующий автоматизации и повторяемости, стимулирует частые развертывания и позволяет тестировать исправления для критически важных систем. Вы можете внедрить автоматические тесты различных вариантов зависимостей, чтобы выявить риски, которые несет установка исправления, и без лишнего волнения устанавливать исправления по мере необходимости.

Вы будете обновлять контейнерное приложение так же, как если бы оно работало непосредственно на сервере. Большинство образов контейнеров включает значительное количество пакетов ОС, которые требуют периодического обновления. Вы можете использовать инфракод для создания новых, исправленных образов контейнеров, а затем тестировать и разворачивать их.

Популярная методология проектирования программного обеспечения, приложение двенадцати факторов (The Twelve-Factor App)[1], рекомендует *явно объявлять и изолировать зависимости* (**https://oreil.ly/45S4A**), что устраняет неявную зависимость от общесистемных пакетов. Включая манифест с определенными версиями приложений, вы можете надежно воспроизводить сборки без влияния на базовую ОС. Кроме того, это дает возможность тестировать сборки с новыми версиями, обновляя манифест, а не полагаясь на доступные обновления от поставщика ОС. Если вы изолируете зависимости, помните, что в дополнение к установке обновлений для ОС вам необходимо планировать поддержание актуальности манифеста зависимостей, что включает повторные сборку, тестирование и развертывание вашего приложения.

Обеспечение безопасности вашей сети

Средства управления сетью обеспечивают глубокую защиту сетевых служб. Если злоумышленник не может взаимодействовать со службой, он не может атаковать ее напрямую, независимо от имеющихся в ней уязвимостей или неправильной конфигурации. Это базовое понимание привело к разработке классической стандартной сетевой топологии. Системные администраторы создавали доверенную основную сеть, в которую входило большинство систем организации, и настраивали межсетевые экраны для ограничения доступа к этой основной сети из внешних, менее доверенных сетевых зон. В этой топологии общедоступные системы, такие как вебсерверы, находятся в самой внешней, так называемой *демилитаризованной зоне* (demilitarized zone, DMZ). Все объекты, находящиеся за пределами основной сети и DMZ, не являются доверенными и должны быть проверены, прежде чем начнут передавать данные в основную сеть.

---

[1] Узнайте больше о методологии двенадцати факторов (**https://12factor.net**), разработанной инженерами компании Heroku, которые «были свидетелями разработки, эксплуатации и масштабирования сотен тысяч приложений».

Система безопасности в этом случае подобна карамели (*англ.* candy bar network security) — жесткая снаружи и мягкая внутри. Идея заключается в том, что внимание атакующего сосредоточено на периметре, и предполагается, что пользователи, имеющие доступ к внутренним ресурсам, делают все необходимое, не испытывая каких-либо неудобств из-за ограничений системы безопасности.

Недостатки такой сети, основанной на доверии, становятся очевидными, когда сетевая изоляция начинает использоваться в качестве основной защиты от небезопасных систем или протоколов. Тогда, например, злоумышленник, получающий доступ к одной системе в доверенной сети, получает доступ к незащищенным системам.

Первым шагом к улучшению защиты данных в сети может стать внедрение виртуальных локальных сетей (VLAN) для сегментирования сети. Но хотя это добавляет несколько дополнительных уровней защиты, сеть остается фактически однородной, напоминая ту самую мягкую карамельную начинку.

Более передовой подход заключается в применении программно-определяемого межсетевого экрана. Межсетевые экраны устанавливаются на каждом из вычислительных узлов и узлов хранения данных. Теперь система не только должна быть внутренней по отношению к сети, но и должна быть сконфигурирована соответствующим образом для получения доступа. Продукты, поддерживающие программно-конфигурируемую сеть, позволяют быстро изменять конфигурацию при добавлении и удалении серверов и сервисов. При добавлении новых систем проверяйте, к каким сервисам им требуется доступ, и ограничивайте сетевое взаимодействие только этими сервисами. Первоначальные усилия по установлению этих сетевых зависимостей окупаются в дальнейшем, т. к. вы получаете более понятную архитектуру с потоками данных, которые явно задокументированы в инфракоде.

В целом в отрасли набирает популярность подход, использующий модель архитектуры нулевого доверия. Ключевые принципы модели «нулевого доверия» следующие:

- отсутствие неявного доверия к сущностям, основанного на их местоположении;
- модель требует, чтобы доступ к ресурсам предоставлялся после аутентификации и авторизации;
- защита ориентирована на ресурсы, а не на сегменты сети.

Другими словами, речь идет не столько о защите сети, сколько о предоставлении возможности каждой авторизованной и аутентифицированной сущности в сети (например, серверу или чьей-то рабочей станции) взаимодействовать только с теми сервисами, которые разрешены на основе установленных политик.

Динамический характер контейнерных и бессерверных рабочих нагрузок создает дополнительные проблемы и возможности для сегментации сети. Большинство продуктов и сервисов имеют встроенные или дополнительные функции для обеспечения сетевого взаимодействия в стиле нулевого доверия, интегрированного с оркестрацией рабочей нагрузки. Например, сетевые политики в Kubernetes могут

быть нацелены на определенные поды в соответствии с привычными селекторами, которые администраторы и разработчики используют для всего остального. Если вы хотите использовать сетевые политики в Kubernetes, важно убедиться, что выбранный вами сетевой подключаемый модуль Kubernetes поддерживает функции, необходимые для достижения целей по защите данных в сети.

Рекомендации по безопасности для управления инфраструктурой

Если в вашей организации в настоящее время практики IaC/IaD задействованы в минимальном объеме или вовсе отсутствуют, первым делом разберитесь, что используется или запланировано. Включите вопросы безопасности в свои первоначальные планы или добавьте их в общую стратегию.

Вот мои общие рекомендации (независимо от уровня управления вашей инфраструктурой):

1. Проверьте, у кого есть доступ на запуск автоматизации и инфракода. Убедитесь, что данные привилегии позволяют выполнять только необходимые задачи и не дают возможности изменять правила ведения журнала для этих задач.
2. Используйте безопасные способы создания и хранения учетных записей.
3. Не используйте повторно учетные данные пользователей или сервисов. С помощью IAM можно генерировать и отменять учетные данные, задействуя их по мере необходимости.
4. Убедитесь, что инфракод предоставляет только необходимые привилегии пользователям и ресурсам (например, виртуальным машинам).
5. Ищите конфигурации, которые помогут укрепить целостность используемых ресурсов.
6. В средах облачных вычислений ограничьте радиус возможной компрометации, привязав по одной учетной записи к каждой рабочей нагрузке.
7. Автоматизируйте выполнение требований политик в вычислительных средах. Например, если вы используете Google Cloud Storage в качестве сетевого хранилища файлов, можете рассмотреть возможность решения только проблем, связанных с доступом, как было показано ранее в этой главе. Для этого конкретного ресурса можно зашифровывать объекты в контейнере, что добавляет еще один уровень безопасности на тот случай, если кому-то удастся получить доступ непосредственно к контейнеру. С помощью этого фрагмента кода Terraform можно включить унифицированный доступ на уровне контейнера (**https://oreil.ly/imioB**) и предоставить ключ, который использовался для шифрования объектов в контейнере Google Cloud Storage:

```
resource "google_storage_bucket" "static-assets" {
  name = "static.example.com"
  uniform_bucket_level_access = true
```

```
  encryption {
    default_kms_key_name = "static-assets-key"
  }
}
```

8. Добавьте статический анализ кода для проверки вашего инфракода, чтобы исключить ошибки в конфигурации системы безопасности или неоптимальные решения. Эту функцию можно добавить непосредственно в инструмент автоматизации развертывания в качестве одного из условий, определяющих автоматическое развертывание. Для проверки инфракода можете использовать, например, `checkov` (**https://oreil.ly/mzuBD**) — инструмент с открытым исходным кодом. Проверка, выполненная для предыдущего примера с контейнером Cloud Storage и Terraform, возвращает следующие результаты:

```
terraform scan results:

Passed checks: 2, Failed checks: 0, Skipped checks: 0

Check: CKV_GCP_5: "Ensure Google storage bucket have encryption enabled"
  PASSED for resource: google_storage_bucket.static-assets
  File: /gcp_bucket.tf:1-7
  Guide: https://docs.bridgecrew.io/docs/bc_gcp_gcs_1

Check: CKV_GCP_29: "Ensure that Cloud Storage buckets have uniform
  bucket-level access enabled"
  PASSED for resource: google_storage_bucket.static-assets File: /gcp_bucket.tf:1-7
  Guide: https://docs.bridgecrew.io/docs/bc_gcp_gcs_2
```

9. Проверьте свои репозитории контроля версий на наличие секретов. Для этого можно использовать, например, gitleaks (**https://oreil.ly/6tnYR**) — инструмент с открытым исходным кодом, который обнаруживает жестко закодированные секреты в репозиториях Git. Хостинговые сервисы для управления версиями, такие как GitHub (**https://oreil.ly/6wck4**), начали предоставлять услуги проверки на наличие секретов, что позволяет предупреждать администраторов хранилищ и владельцев бизнеса о потенциальных утечках.
10. Наконец, вы можете использовать инфракод для обеспечения соблюдения политики безопасности в вашей организации. Рассмотрите способы усиления уже используемых инструментов, применяйте автоматизацию для проверки соблюдения политик, используйте системы управления изменениями, чтобы упростить подготовку документации одновременно с обновлением политик, и обеспечьте видимость, предоставив возможность аудита политик безопасности и соблюдения требований регуляторов.

Заключение

Независимо от того, фигурирует ли слово «безопасность» в названии вашей должности, поддержание безопасности систем, находящихся в вашем ведении, является важным аспектом вашей работы в качестве системного администратора. Подумайте

о векторах атак, которые могут нанести вред вашим системам. Они могут возникать из-за неправильного обращения с учетными данными, неправильной настройки программного обеспечения и неустановленных исправлений. Чтобы противостоять этим рискам, необходим многовекторный подход к защите учетных записей, секретов и ресурсов инфраструктуры.

Традиционные меры по обеспечению сетевой безопасности направлены на создание межсетевых экранов, но это приводит к эффекту «карамельки», когда крепким является только периметр сети, а внутри она не защищена. В модели с нулевым доверием периметру не уделяется основное внимание. На вооружение берется подход «никогда не доверяй, всегда проверяй». В нем акцент сделан на проверке того, что каждая попытка доступа к любому ресурсу должна исходить от проверенной учетной записи на надежном устройстве, независимо от источника трафика. Это также позволяет отказаться от учетных записей `root`, которые имеют максимально широкие права и полный контроль над всем. Теперь учетным записям предоставляются только те права, которые достаточны для выполнения определенных задач.

Подход к безопасности на основе нулевого доверия задействует систему управления доступом к идентификационным данным, которая позволяет вам проверять пользователей и сервисы, авторизованные для доступа к вашим ресурсам, и, в свою очередь, зависит от подхода к управлению секретными данными для защиты паролей и других токенов доступа, используемых для получения доступа к ресурсам.

При внедрении философии безопасности в вашу работу, так же, как и в случае с подходами «инфраструктура как данные» и «инфраструктура как код», действуйте постепенно, ищите конкретные области, где вы можете разработать последовательные, удобные в сопровождении и масштабируемые стандарты безопасности.

Дополнительные ресурсы

Вы можете узнать больше о нулевом доверии из следующих источников:

- статья Джона Киндервага (John Kindervag) «No More Chewy Centers: Introducing the Zero Trust Model of Information Security» (**https://oreil.ly/7Nuaa**), опубликованная агентством Forrester Research;
- реализация модели нулевого доверия компанией Google: BeyondCorp (**https://oreil.ly/CLGm7**);
- если вам нужна официальная ссылка, ознакомьтесь с NIST SP 800-207 (**https://oreil.ly/GiQXj**).

Узнать больше о защите инфраструктуры можно на следующих ресурсах:

- SLSA (**https://oreil.ly/AmWmy**), попытка создать набор отраслевых стандартов по улучшению ресурсов инфраструктуры;
- Лиз Райс (Liz Rice) «Безопасность контейнеров» («Container Security»,(O'Reilly);
- Разнообразные материалы по безопасности от экспертов отрасли представлены в книге «97 Things Every Information Security Professional Should Know» под редакцией Тобиаса Мейси (Tobias Macey, O'Reilly).

ЧАСТЬ IV

Наблюдение за системой

Вы можете работать с произвольным числом различных систем. В следующих четырех главах представлена основа для определения эффективных стратегий мониторинга, оценки существующих инструментов и платформ для мониторинга. Показывается, как управлять данными мониторинга и планировать работу в процессе профессиональной деятельности.

Мониторинг сложных систем включает сбор телеметрии приложений и предоставляет возможность более глубокого наблюдения за компонентами вашей системы. В прошлом системное администрирование больше фокусировалось на метриках системы. Но по мере перехода к более крупным и сложным средам метрики системы становятся менее полезными, а в некоторых случаях и вовсе недоступны. Кроме того, влияние отдельных систем становится менее критичным, когда вы сосредоточиваетесь на качестве приложения и влиянии на ваших пользователей.

ГЛАВА 14

Теоретические аспекты мониторинга

Мониторинг — это измерение, сбор, хранение, изучение и визуализация данных инфраструктуры (включая оборудование, программное обеспечение и процессы, связанные с человеческой деятельностью). Мониторинг помогает вам ответить на вопросы «когда» и «почему» в вашей работе, на его основе принимаются бизнес-решения, поддерживающие стабильность работы сотрудников (например, прием в штат дополнительных специалистов, чтобы ваши сисадмины не работали постоянно на пределе возможностей).

В этой главе я помогу вам поразмышлять о мониторинге, предоставив основу для определения его эффективных стратегий. Я покажу различие между мониторингом и наблюдаемостью, поясню, что такое элементы и этапы процесса мониторинга, и продемонстрирую, как они работают вместе. Понимание в общих чертах этих механизмов поможет вам обозначить приоритеты для других желаемых результатов, которые позволяет получить мониторинг, решить, как и что вы будете контролировать, и повысить прозрачность рабочего процесса, систем и команд, независимо от выбранных вами инструментов.

Зачем нужен мониторинг?

Существует множество причин для проведения мониторинга и повышения прозрачности системы. Он привлекает внимание к слабым сторонам, неустойчивости или риску и помогает вам принимать более правильные решения. Прозрачность необходима по следующим причинам.

Обнаружение проблем

Вы выявляете проблемы и понимаете, как можно решить их. Например, вы можете обнаружить проблемы, отслеживая задержки веб-запросов и определяя, когда медленные запросы к MySQL влияют на клиентов.

Совершенствование процессов

Вы постоянно совершенствуете процессы в команде, чтобы повысить точность и скорость решения задач, автоматизировать трудоемкую работу и повысить общую эффективность, не перегружая при этом сотрудников. Например, вы можете улучшить процессы, отслеживая рабочие очереди, чтобы определить их влияние на команду.

Управление рисками

Вы выявляете и оцениваете потенциальные проблемы, устанавливаете очередность их решения. Например, вы можете управлять рисками, наблюдая за раз-

вертыванием программного обеспечения и корректируя параметры автоматизации или процессов для снижения частоты и серьезности поломок.

Базовые модели поведения

Вы индексируете типичное поведение системы при стандартной нагрузке. Например, вы можете установить базовую модель поведения, отслеживая данные в течение длительного периода времени, чтобы увидеть тенденции в работе сервиса и проанализировать влияние отдельных периодов, таких как праздники и выходные, и ожидаемых мероприятий — например, выборов и спортивных соревнований.

Формирование бюджета

Вы определяете и оцениваете возможные инвестиции в инфраструктуру, определяете их приоритеты и обеспечиваете подконтрольность расходов. Одним из способов формирования бюджета является мониторинг расходов на инфраструктуру для выявления областей, где экономическая эффективность тех или иных решений может оказаться более высокой, или установление ограничений, чтобы инженеры могли тестировать новые решения, не опасаясь превысить расходы.

Управление мощностями

Вы создаете устойчивый потенциал на основе потребностей бизнеса. Например, можно управлять мощностями с помощью мониторинга инфраструктуры, чтобы определить, когда зарезервированные экземпляры позволят сэкономить деньги по сравнению с разовыми экземплярами.

Мониторинг — это нечто гораздо большее, чем внедрение одного инструмента. Это выявление того, какие сведения вам нужны, какие результаты вы хотели бы получить, оценка имеющихся инструментов и внедрение методов, которые лучше всего помогают достичь данных целей. Кроме того, размышления о том, для чего нужен мониторинг, и установка конкретных его целей стимулируют критическое мышление в контексте вашего бизнеса, чтобы вы не копировали методы мониторинга поставщика услуг, если они не подходят для ваших целей.

Будьте сами себе авторитетом

Многие специалисты говорят нам, что и зачем нужно контролировать, но я хочу, чтобы вы сами стали лучшим авторитетом в отношении систем в своей среде. Представьте, например, что вы запускаете веб-сервис для своей компании. Хотя это может быть то же самое программное обеспечение, которое используется в других организациях, специфика вашего веб-сервиса будет отличаться от других организаций. Вы знаете, каковы риски сбоев в различных частях сервиса и кто отвечает за его функционирование в реальном времени, от разработчиков до специалистов службы поддержки. Программное обеспечение, конфигурации, процессы и люди — все это влияет на стратегию мониторинга, позволяя извлечь максимальную пользу для бизнеса и одновременно помочь людям, которые работают с программным обеспечением.

Чем отличаются мониторинг и наблюдаемость?

Рудольф Е. Калман (Rudolf E. Kálmán) в 70-х годах ввел понятие наблюдаемости для линейных динамических систем. Наблюдаемость показывает, насколько хорошо вы можете судить о процессах внутри системы, наблюдая только за ее выходными данными. Система в данном случае — это совокупность взаимосвязанных объектов, которые рассматриваются как единое целое для моделирования поведения. Например, вы можете изучать отдельный узел, контейнер или распределенный сервис в целом.

Наблюдаемость — это не мониторинг, а мониторинг — не наблюдаемость. Наблюдаемость — это свойство системы, мониторинг — многоэтапный процесс наблюдения за системой. Часто люди думают о мониторинге как об информационных панелях и оповещениях в производственной среде. Такое определение мониторинга приводит к тому, что его считают подмножеством наблюдаемости. Далее возникает проблема с определением: как следует называть другие виды деятельности, которые необходимо контролировать? В итоге вы используете дублирующиеся термины, чтобы охватить все возможные варианты использования, что повышает вероятность недопонимания. Методы мониторинга в разных организациях всегда различались.

В некотором смысле гораздо проще думать о «ненаблюдаемости» системы. Например, представьте, что ваши клиенты столкнулись с проблемой, которую ваши информационные панели и оповещения не выявили или не объяснили. Если ваши исходные данные не помогают вам объяснить, почему и как возникла проблема, это указывает на отсутствие наблюдаемости.

Вы можете контролировать наблюдаемость ваших систем, оценивая разнообразие возникающих проблем, то, как часто вы можете ответить на вопросы, используя имеющиеся данные, и как часто о причинах возникновения проблемы можно сказать лишь: «Я не знаю».

Наличие наблюдаемости помогло бы вам найти проблемы, о которых вы не знаете, и лучше настроить отклик системы. Наблюдаемость заключается в деталях.

Вам не нужна наблюдаемость в каждой системе. Так, например, если вас интересует только то, работает ли конкретная система, и вы не пытаетесь настраивать ее ресурсы, вам не нужно заботиться о наблюдаемости этой системы. Внедрение расширенной трассировки, даже с выборкой, равно как и настройка многочисленных метрик на случай, если они вам понадобятся, — это неверный подход.

Давайте рассмотрим реальный случай. Когда я работаю на своем MacBook Pro, а система начинает тормозить, как выяснить основную причину? По умолчанию все события записываются в системный журнал. Я установила iStat Menus для сбора данных с физических компонентов, чтобы сразу можно было увидеть исчерпание ресурсов процессора, памяти и сети. И я не приобретала никаких дополнительных инструментов мониторинга, поэтому, когда что-то идет не так, мне приходится копаться в системных инструментах, чтобы наблюдать за системой.

Насколько наблюдаема моя система? Это зависит от обстоятельств. Если бы я собирала все метрики по своей системе, она стала бы непригодной для использования. Поэтому я использую прикладные и системные инструменты, необходимые для отслеживания проблемных областей. У меня нет возможностей глубокого анализа или автоматизированного выявления и решения проблем в системе, зато есть инструменты для решения большинства проблем с программным обеспечением на ноутбуке.

Термины, которые используются в командах, организациях и отрасли, постоянно меняются. В результате в сообществе, занимающемся вопросами мониторинга, возникают конфликты, сигнализирующие об отсутствии общего контекста в отношении того, как используются такие термины, как мониторинг и наблюдаемость, и является ли наблюдаемость подмножеством или расширенным вариантом мониторинга. Когда поставщики выводят свои решения на рынок, они могут использовать одну и ту же терминологию с небольшими различиями в значении слов, что приводит к недопониманию.

Потратьте время на создание общего контекста в команде, касающегося использования терминов, связанных с мониторингом. Тогда вы будете лучше разбираться в предложениях поставщиков услуг в области мониторинга и сможете выбрать наиболее подходящие для вашей команды.

Основные элементы мониторинга

Давайте познакомимся с основными элементами мониторинга: событиями, мониторами и собираемыми данными.

События

Событие — это то, что происходит, факт, который можно отследить. Событием может быть система, приложение или отдельный сервис. События происходят независимо от того, отслеживаете вы их или нет. Ниже приведены несколько примеров событий:

- загрузка процессора в определенный момент времени;
- выполнение определенного программного кода;
- завершение работы приложения сисадмином.

Мониторы

Монитор — это инструмент, который определяет и фиксирует события, представляющие интерес. Мониторы бывают фиксированными (заранее определенные явления, о которых вы знаете) или гибкими (эпизодические явления, о которых вы еще не знаете). Фиксированные мониторы — это специальные функциональные тесты на известную проблему, которые вы не настраиваете во время выполнения. Вы можете использовать фиксированные мониторы для работы с журналами событий, датчиками процессора или памяти. Гибкие мониторы — это проверки, которые вы можете менять в зависимости от ситуации. Трассировка — это пример гибкого монитора, который собирает и записывает события. Например, в системе Linux вы

можете запустить утилиту `strace` для процесса, чтобы собрать все сделанные системные вызовы. Гибкие мониторы используются при диагностике проблем, оценке производительности или изучении работы системы.

Кроме того, мониторы могут быть узкими или широкими. Узкие мониторы могут определять событие как одну инструкцию, например вызов журнала. Широкие мониторы могут определять событие как совокупность инструкций — например, один веб-запрос, который приводит к множеству действий системы.

Мониторы могут быть событийно управляемыми или с периодической выборкой. Событийно управляемые мониторы выполняются при наступлении события, предоставляя сводные показатели за отчетный период. Мониторы с периодической выборкой запускаются через определенные интервалы времени, собирая статистически значимое количество событий.

Данные: метрики, журналы и трассировка

Мониторы собирают данные о сконфигурированных событиях трех основных типов: метрики, журналы и трассировка. Вы автоматически собираете данные с систем, устройств, приложений и сетей. Возможно, вы сможете применить фильтры для ограничения сбора данных или собирать их выборочно для оценки, а не для точного анализа всех данных. Более подробно я остановлюсь на тонкостях мониторинга данных в главе 16.

Мониторинг первого уровня

Процесс мониторинга включает в себя ряд последовательных шагов: обнаружение события, сбор данных, сокращение объема данных, анализ данных и их представление (*см. рис. 14.1*).

Давайте рассмотрим эти пять шагов, показанных на рис. 14.1, по отдельности.

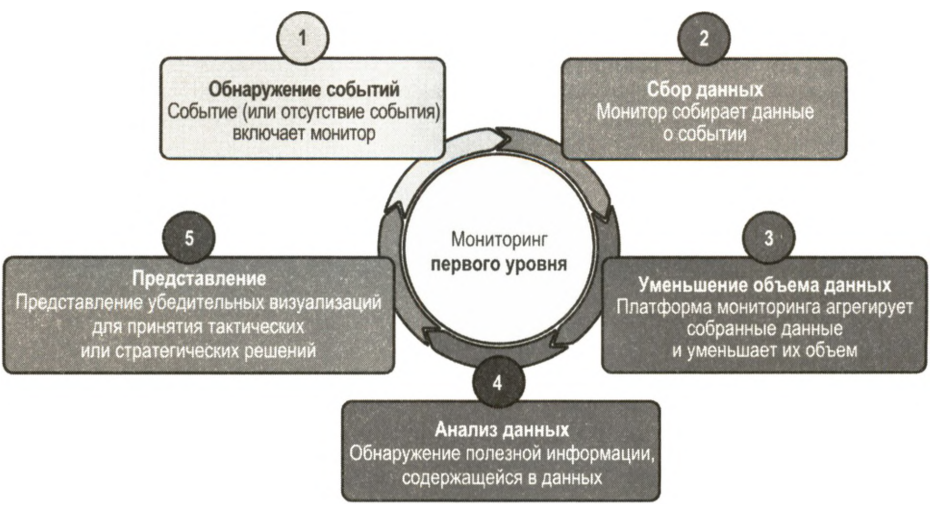

Рис. 14.1. Пять шагов в мониторинге первого уровня

Обнаружение события

Первым шагом в процессе мониторинга является *обнаружение события*. События вызывают запуск мониторов. Кроме того, некоторые мониторы отслеживают отсутствие ожидаемых событий.

Сбор данных

Второй шаг в процессе мониторинга — *сбор данных*, когда мониторы собирают данные о произошедших событиях. Данные мониторинга могут быть собраны отслеживаемой системой следующими способами:

- отправка данных на центральный сервер мониторинга по расписанию или на основе события;
- подача сигнала серверу для отправки данных;
- получение данных посредством диагностики системы.

В зависимости от размеров вашей среды и параметров, которые вы измеряете, в централизованной системе мониторинга, отвечающей за сбор данных, может возникнуть проблема масштабирования: она должна успевать обрабатывать заданное количество событий, чтобы не возникала очередь, влияющая на производительность сервиса, и не было отброшенных событий.

Метод сбора может вызывать эффект наблюдателя: представьте себе последствия стратегии сбора данных по времени, когда каждый монитор проверяется в полночь. Такой мониторинг может вызвать истощение ресурсов процессора или диска, что приведет к увеличению задержки и появлению ненужных предупреждений.

Метод сбора может влиять на то, какие данные и как вы отслеживаете. Например, метрики обычно определяются по событиям и объединяются в сводные показатели для уменьшения объема данных.

Если у вас есть метрики, относящиеся к людям, убедитесь, что конфиденциальные сведения находятся под защитой и вы получаете согласие на сбор таких данных. В случае с персональными данными вам, возможно, придется учитывать дополнительные правила и нормы, поэтому в первую очередь обращайте на это внимание, чтобы избежать нарушения конфиденциальности пользователей.

Кроме того, не предполагайте, что пользователи будут всегда соглашаться на ваши условия, особенно если вы меняете контекст или метод сбора данных. Примером, когда вам может понадобиться учитывать эти аспекты, являются телеметрические данные, собранные и зарегистрированные в процессе использования приложения человеком.

Сокращение объема данных

На третьем этапе платформа мониторинга определяет сводные показатели и сокращает объем данных. Хотя агрегация данных и сокращение их объема могут происходить во время сбора, вы можете захотеть, чтобы эти действия выполнялись по отдельности, особенно при работе с распределенными данными.

Агенты мониторинга собирают данные из множества различных источников. Платформа мониторинга может агрегировать, редактировать, сортировать или сжимать данные, предоставляя вам наиболее значимую информацию.

Что касается метрик, более старые данные иногда агрегируются в целях хранения, что позволяет проводить сравнение с базовыми показателями в разные моменты времени. «Более старые» — контекстно-зависимое понятие и может означать недели, месяцы или годы. Например, для мониторинга количества запросов не требуется точно знать все значения за шесть месяцев с пятиминутным интервалом. Вместо этого вычислите усредненное значение за некоторый интервал времени, учитывая, что получить данные за меньший отрезок времени будет уже невозможно.

Для некоторых метрик, однако, такой подход может оказаться менее полезным. Но, с другой стороны, хранение данных стоит денег, поэтому агрегирование — это баланс затрат и практической пользы.

Анализ данных

На четвертом этапе вы анализируете данные, чтобы получить полезную информацию о бизнес-операциях и прямых действиях. В ходе этого анализа вы определяете набор показателей уровня обслуживания (*англ.* service-level indicators, SLI), которые помогут вам измерить надежность вашей системы. Надежность может характеризоваться с помощью различных показателей в зависимости от вашего сервиса:

- доступность — это продолжительность времени, в течение которого система функционировала так, как ожидалось;
- задержка — это общее время обслуживания запроса при его движении от источника к месту назначения. Задержка должна измеряться отдельно для успешных и неудачных запросов. Часто неудачные запросы могут быть очень быстрыми;
- пропускная способность показывает количество запросов, проходящих через систему;
- устойчивость характеризует долговременную защиту данных. Сохраненные данные не ухудшаются и не повреждаются.

Когда у вас есть SLI, вы можете определить достижимые и приемлемые уровни надежности, установив цели уровня обслуживания (*англ.* service-level objectives, SLO), которые измеряют ожидаемое поведение системы. Поскольку сложно (и дорого) обеспечить более высокую надежность, чем у внешних поставщиков услуг, вы должны учитывать эти зависимости при определении целей. Не забудьте учесть сеть и DNS.

Вы можете узнать больше о SLO в главе 4 (**https://oreil.ly/KCy2H**) книги «Site Reliability Engineering» от Google, а об их внедрении — в главе 2 (**https://oreil.ly/acmoK**) книги «The Site Reliability Workbook».

Узнайте больше о практической реализации SLI, SLO и бюджетов на ошибки из книги Алекса Идальго (Alex Hidalgo) «Implementing Service Level Objectives» (O'Reilly).

Представление данных

Пятым шагом в процессе мониторинга является представление информации. Чтобы превратить данные в информацию, вы создаете визуализации. Сначала вы собираете диаграммы на информационных панелях, охватывая известные узкие места и зоны повышенного риска. Далее создаете другие узкоспециализированные визуализации для изучения имеющихся данных.

Вы можете создавать графики на основе автономных данных, полученных в режиме реального времени. Например, оповещения должны быть максимально приближены к данным в реальном времени, чтобы ограничить влияние проблем. При квартальном планировании мощностей для кластера Hadoop можно агрегировать различные источники данных и обрабатывать их в автономном режиме.

Информационные панели объединяют набор визуализаций для передачи информации и зависят от ваших потребностей. Например, вы можете принять единовременное стратегическое решение, определить повседневное оперативное направление, или пересматривать систему еженедельно либо ежемесячно, чтобы выработать тактический подход. Эти информационные панели являются продуктами, которые побуждают к действиям. Конечные результаты и информация должны возвращаться в различные командные и организационные процессы.

Мониторинг второго уровня

Ваши системы существуют для того, чтобы выполнять какую-то функцию, предоставлять определенную услугу. Мониторинг не обеспечивает безопасность ваших систем. Он помогает вам предоставлять конкретные услуги вашей организации, а ещё получить ответы на вопросы «когда» и «почему» в процессе вашей работы. Сидни Деккер (Sidney Dekker) ввел понятие «уход в сторону неудачи», согласно которому системы постепенно и по нарастающей приходят в упадок из-за увеличения удельного веса факторов риска.

Уход в сторону неудачи связан не столько с поломками или неправильным функционированием компонентов, сколько с тем, что организация не может эффективно адаптироваться, чтобы справиться со сложностью своей структуры и среды[1].

Йенс Расмуссен (Jens Rasmussen) разработал основанную на состоянии модель социотехнической системы, окруженную тремя границами (экономическая неэффективность, неприемлемая нагрузка и приемлемая производительность), демонстрирующую риски для системы при работе вблизи этих границ[2].

Проведение более критического анализа — двойной петли обучения — добавляет второй уровень мониторинга, поскольку вы включаете данные, полученные в ре-

---

[1] Сидни Деккер (Sidney Dekker) «Уход в сторону неудачи: от поиска проблемных компонентов до понимания сложных систем» («Drift into Failure: From Hunting Broken Components to Understanding Complex Systems», Boca Raton, FL: CRC Press, 2011).

[2] Подробнее о моделировании управления рисками читайте в статье Йенса Расмуссена (Jens Rasmussen) «Risk Management in a Dynamic Society» (**https://oreil.ly/ewYMX**).

зультате мониторинга первого уровня, обратно в систему, чтобы лучше понимать, куда движется система и какие свойства проявляются в процессе эксплуатации. Двойная петля обучения основана на том, что руководство дает возможность людям учиться постепенно, путем проб и ошибок, анализа и рефлексии, и вносить соответствующие изменения.

Узнайте больше о модели состояния из выступления доктора Ричарда Кука (Richard Cook) на конференции Velocity в Нью-Йорке в 2013 году «Resilience in Complex Adaptive Systems» (**https://oreil.ly/l1fw7**).

Заключение

Вам необходим подход, основанный на данных, для управления вашей инфраструктурой и внедрения системы мониторинга, чтобы ответить на вопросы «когда» и «почему» о ваших системах. Информация, полученная в процессе мониторинга, поможет вам обнаружить проблемы, улучшить процессы, снизить риски, подтвердить правильность выбора при выделении ресурсов и принять обоснованные решения по планированию мощностей.

Мониторинг и наблюдаемость — это разные понятия. Разные специалисты трактуют их по-своему. Наблюдаемость — это неотъемлемое свойство системы, которое существует независимо от того, наблюдаете ли вы за этой системой или нет. Мониторинг — это многоступенчатый процесс наблюдения за системой. Мониторинг включает в себя обнаружение событий, сбор, фильтрацию, извлечение, анализ данных и их представление. Этот процесс определяет ваши решения по управлению системами.

Применяя двойную петлю обучения и опираясь на обратную связь от системы при внесении изменений в нее, вы сможете избежать ухода в сторону неудачи.

Мониторинг вычислительной инфраструктуры и программного обеспечения на практике

Длительная поддержка сервиса приучает вас обращать внимание на оперативные сигналы, предупреждающие о проблемах в системе. Из журналов событий можно быстро получить полезную информацию. Но у новичка в команде нет таких преимуществ, как время и опыт работы с системами. Даже просматривая те же журналы событий и метрики, что и вы, он не сможет получить полезную информацию. Кроме того, если работа подразумевает получение информации о системе только из журналов и метрик, значит, мониторинг и документирование не настроены должным образом.

Если вы управляете широким спектром систем, вы должны ответить на следующие вопросы: за какими показателями можно наблюдать и какие из них дают преимущество для бизнеса? Ваша среда и бизнес-цели уникальны, поэтому ответы на данные вопросы могут не походить на ответы других людей. По этой причине я не буду предлагать в этой главе конкретную стратегию мониторинга или рассказывать вам о необходимости отслеживать четыре метрики для завершения настройки мониторинга.

Вместо этого я объясню, какие мониторы важны для вас, и предложу методы оценки различных инструментов и фреймворков, чтобы вы понимали, как их использовать. Результаты мониторинга должны представлять ценность для вашего бизнеса и способствовать устойчивости команды.

Определите желаемые результаты

При планировании стратегии мониторинга многие начинают с вопроса: «За какими параметрами мне следует наблюдать?» Я же считаю, что вопрос должен звучать так: «Что мне нужно сейчас?» или «Что вызывает проблемы в работе моей команды?»

В верхней части рис. 15.1 типичные метрики показывают, что все в порядке, но клиент недоволен, т. к. его ожидания относительно вкуса кексов не оправдались. Конечно, системный администратор мог бы добавить шоколад. Тем не менее для эффективного администрирования необходимо продумать, какие данные и как собирать, какие индикаторы уровня обслуживания в целом следует использовать для изменения результатов работы системы.

В нижней части рис. 15.1 системный администратор определяет количественный показатель «шоколадности». Стандартные показатели для оценки конечного продукта по-прежнему имеются. Кроме того, появился монитор, измеряющий уровень

шоколада в экземпляре, чтобы можно было добавить больше шоколада в зависимости от конкретных ожиданий клиента. Теперь можно напрямую реагировать на потребности клиента и повышать ценность предоставляемых услуг.

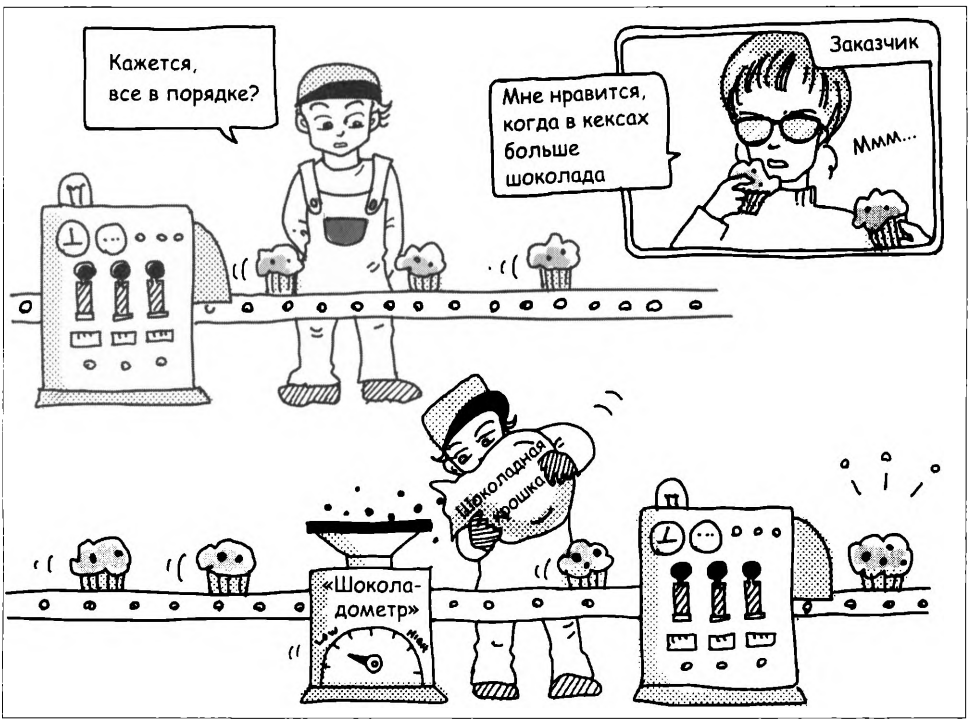

Рис. 15.1. Системный администратор следит за выходом своей системы, добавляет новый монитор, определяет, что шоколада недостаточно, и реагирует соответствующим образом

Давайте вспомним этапы процесса мониторинга из *главы 14*. Они показаны в табл. 15.1.

Таблица 15.1. *Примеры различных результатов мониторинга*

Этап процесса мониторинга	Результаты
Обнаружение событий	Мониторы
Сбор данных	Метрики, журналы, следы
Анализ данных	Индикаторы уровня обслуживания, запросы платформы регистрации, предупреждения
Представление данных	Цели уровня обслуживания, диаграммы, информационные панели

Каждый этап процесса мониторинга имеет определенные артефакты, которые возникают как выходные данные. Поэтому вместо того, чтобы думать о предмете мониторинга (и сфокусироваться на обнаружении событий), размышляйте о конкретных выходных данных, которые улучшат ваши процессы или общие результа-

ты (например, приборные панели, показатели уровня обслуживания, подробные метрики или мониторы).

Конечно, выходные данные, которые появляются позже в процессе мониторинга, зависят от предыдущих этапов, поэтому при планировании проекта учитывайте эти ограничения в зависимостях, чтобы избежать угрозы срыва или смещения сроков.

Теперь, когда вы рассмотрели различные этапы процесса мониторинга, подумайте о том, за какими параметрами нужно наблюдать.

За какими параметрами следует наблюдать?

> Начните оттуда, где вы сейчас находитесь. Используйте то, что у вас есть, и делайте все, что можете.
> – *Артур Эш (Arthur Ashe)*

Выяснение того, что именно вы должны отслеживать в вашем конкретном случае, требует многостороннего подхода: сузьте сферу охвата любого проекта, чтобы увеличить шансы на успех (делайте все, что можете сейчас), задайте себе правильные вопросы (отслеживайте то, что важно), а затем постепенно вносите небольшие изменения, непрерывно обучаясь.

Делайте то, что можете сейчас

Мониторинг может оказаться непосильной задачей, особенно когда вы знаете, что необходимо изменить, но все эти изменения требуют финансирования и поддержки со стороны руководства, на что уйдут месяцы. Секрет устойчивого реального мониторинга заключается в том, чтобы сосредоточиться на том, что вы можете сделать сейчас — постепенно вносить изменения, которые возможны на данный момент, и постоянно работать над достижением более масштабных целей. Мониторинг никогда не бывает «завершенным». Поэтому лучше всего относиться к нему как к постоянному процессу.

Я участвовала во многих проектах внедрения мониторинга, где руководство хотело получить быструю победу и покончить с ним. Если вы оказались в такой ситуации, устраните недопонимание, связанное с вашим проектом. Сузьте сферу охвата вашего проекта, связанного с мониторингом, чтобы можно было успешно отслеживать некоторое количество параметров и иметь возможность для дальнейшего расширения.

Вспомните о шести областях мониторинга, рассмотренных в *главе 14*: выявление проблем, совершенствование процессов, управление рисками, базовые модели поведения, определение бюджета и управление мощностями. Первым шагом при определении сферы охвата проекта является определение основных областей. Чем больше областей вы включаете в проект, тем больше и продолжительнее он будет. Чем больше времени требуется для выполнения проекта, тем сложнее определить точную дату его завершения. Как и при разработке программного обеспечения, вы должны отдавать предпочтение небольшим постепенным изменениям вашей стратегии мониторинга, поскольку это позволяет вам откатываться назад или модифицировать изменения, которые не приносят пользы (или приносят вред). Четкое

изложение целей проекта поможет убедить высшее руководство в необходимости предлагаемых вами изменений.

Четко определите, что включает в себя ваш проект, и не пытайтесь сделать все сразу. Сосредоточьтесь на том и начните с того, что у вас есть в данный момент. Рассмотрите преимущества и проблемы вашего текущего решения. Подумайте, какие выходные данные, полученные в процессе мониторинга, улучшатся или изменятся. При общении с руководством и коллегами четко формулируйте цели и не злоупотребляйте терминологией (например, говорите «повысить качество обнаружения ошибок для выявления проблем с длинными веб-запросами», а не «наладить мониторинг»). Не вдавайтесь в то, как могут измениться ваши методы по мере выполнения проекта, но укажите желаемые результаты.

Оцените, что у вас имеется в настоящее время. Обратите внимание, как текущая реализация помогает или мешает процессу мониторинга, включая способы анализа и представления информации. Например, вы можете собрать данные о всевозможных приложениях, операционных системах и вычислительных ресурсах. Но выводится ли вся эта информация на единую информационную панель? Упростите информационную панель, чтобы клиенты получали самую важную информацию, поскольку невозможно уделять внимание сразу всему. Не всегда получается увидеть ключевые проблемы клиентов. Анализ поможет выявить наиболее значимые области.

Предположим, вы внедряете исследовательский мониторинг, чтобы можно было анализировать данные для обнаружения имеющихся тенденций или проблем с производительностью. В этом случае вам, вероятно, потребуется минимальная проверка концепции для тех областей, которые представляют ценность для бизнеса.

Документируйте проблемы, отражая их в своей рабочей системе контроля, поскольку эта информация позволяет выполнить более глубокую и всеобъемлющую оценку мониторинга. Например, каждое обращение к дежурному инженеру, которое не завершилось решением проблемы, открывает возможности для улучшения процесса. Без вспомогательной документации вы можете не обратить внимания на области, требующие улучшения.

Мониторы, которые имеют значение

Возможно, вы слышали о четырех «золотых сигналах» мониторинга (задержка, ошибки, трафик и насыщенность), о которых рассказывает Роб Иващук (Rob Ewaschuk) в главе 6 (**https://oreil.ly/HqYFQ**) своей книги «Site Reliability Engineering» (O'Reilly). Помимо четырех золотых сигналов, существует еще несколько общепринятых методов мониторинга:

- метод RED — шаблон, ориентированный на микросервисы для инструментирования и мониторинга, представленный Томом Уилки (Tom Wilkie), который предлагает вам мониторить следующее (для каждого ресурса):
  - количество запросов в единицу времени (количество запросов в секунду);
  - количество ошибок (количество запросов, которые не увенчались успехом);
  - длительность (время обработки одного запроса).

- метод USE (**https://oreil.ly/H4N9E**) — методология производительности системы, представленная Бренданом Греггом (Brendan Gregg), которая предлагает вам мониторить следующее (для каждого ресурса):
  - использование (процент использования ресурса, занятого «полезной работой»);
  - насыщенность (объем работы, которую должен выполнить ресурс, часто длина очереди);
  - ошибки (количество ошибок).

Если вы рассматриваете свою среду сверху вниз, с точки зрения пользователя, сосредоточьтесь на методе RED. В противном случае используйте метод USE, если движетесь снизу вверх и фокусируетесь на ресурсах, оказывающих влияние на пользователя.

Возможно, вы заметили, что этих сигналов недостаточно для решения имеющихся проблем в вашей среде. Все в порядке! Золотые сигналы являются отправной точкой и применимы не к каждой среде. Моя цель здесь — поощрить вас, чтобы вы сами решили, какие параметры отслеживать важнее всего.

План проекта по мониторингу

Начните с изучения архитектуры конкретной системы, связанной с проблемой, которую вам нужно решить. Это может быть одна система с различными компонентами или сложный сервис с несколькими приложениями. В процессе выполнения вы можете обновлять схемы архитектуры. Задайте себе такие вопросы.

- Какая ОС используется (включая конкретный дистрибутив, версию, исправления системы и установленные пакеты)?
- Есть ли в сети списки управления доступом (ACL): конфигурации подсети?
- Как выглядит мой трафик (например, количество запросов в секунду, типы запросов и записанные/прочитанные данные)?
- Какие вычислительные среды используются (т. е. сколько и какого типа)? Каковы конкретные используемые конфигурации?
- Как построена и сконфигурирована моя вычислительная инфраструктура, как она обновляется?
- Существует ли уровень сервисов приложений, который обслуживает запросы?
- Какие сервисы работают в системе?
- Как хранятся данные? Может существовать несколько хранилищ данных.
- Есть ли внутренняя база данных? Если да, то какое программное обеспечение базы данных, какая версия программного обеспечения и какие схемы баз данных используются?
- Реплицируются ли данные в другое место?

- Выполняется ли резервное копирование данных?
- Существуют ли конвейеры обработки данных? Являются ли они потоковыми или пакетными?
- Есть ли балансировщик нагрузки? Какого типа?
- Существуют ли специальные конечные точки пользовательского API? Существуют ли конечные точки API на уровне системы, которые пользователи не должны использовать?
- Где включено кеширование? На уровне приложения, в базе данных, в памяти, снаружи в CDN или в браузере пользователя?
- Есть ли очередь сообщений?

Инструменты, легкодоступные в самоуправляемой инфраструктуре, не всегда имеются на бессерверных платформах. Так что вы можете наблюдать за общим состоянием системы, но разобраться в конкретных проблемах будет уже сложнее, а то и вовсе невозможно.

Мониторинг бессерверных сервисов может потребовать дополнительного сотрудничества с командой разработчиков программного обеспечения, создающих функции, приложения или контейнеры. Для функций весь код, необходимый во время вызова функции, должен быть связан с развертыванием функции. Вам нужно будет написать и зафиксировать изменения кода мониторинга непосредственно в репозитории проекта, чтобы развернуть его вместе с кодом функции.

Изучая и оценивая свою инфраструктуру, подумайте над такими вопросами:

- Каких данных о событиях в моей инфраструктуре мне не хватает?
- Какие данные я собираю?
- Сбор каких данных следует прекратить?

Ваши мониторы могут перекрывать друг друга, хотя иногда они используются для разных целей с разным уровнем детализации. Намеренное использование перекрывающихся мониторов допустимо, если они не предназначены для одной и той же цели. Например, ситуация, когда одно событие приводит к отправке нескольких сообщений дежурному инженеру, представляет проблему, поскольку человек устает от оповещений (теряет чувствительность к громким оповещениям) и дублирование данных должно быть устранено.

Оцените и задокументируйте, какую информацию предоставляют различные мониторы ОС, системного уровня, сети и приложений, чтобы не удалить случайно критически важные. Иногда по мере развития системы мониторинга кажется, что ее упрощение и снижение затрат на хранение путем удаления мониторов — это легкая победа. Однако если вы рассматриваете только одно направление (например, обнаружение проблем), то можете упустить из виду причину существования этих мониторов (например, планирование бюджета или управление мощностями).

Помните, что есть разница между тем, что вы отслеживаете, и тем, о чем вы предупреждаете. Рассмотрите возможность отказа от оповещения о некоторых событиях. Продолжайте совершенствовать то, над чем работаете, и не пытайтесь сделать все на свете. Начните с того, что у вас есть, и работайте над проблемами, связанными

с вашим текущим решением. Если у вас нет мониторинга, это первая проблема, которую нужно решить!

Подумайте о базовых ограничениях TCP/IP и о том, не ограничивает ли пропускная способность сети общее количество метрик и журналов, которые вы можете получить за определенный период времени. Обратите внимание на все инструменты и скрипты, используемые для мониторинга. Документируйте перекрывающиеся мониторы и их назначение. Обратите внимание на области, где рефакторинг мониторов может уменьшить затраты на хранение и сетевые операции. Кроме того, документируйте области, где визуализация сбивает с толку или отвлекает.

Могут существовать дополнительные области, которые необходимо оценить с точки зрения необходимости добавления мониторов. Не распыляйте внимание, пытаясь определить все отсутствующие мониторы, т. к. это масштабное мероприятие. Вместо этого уделите внимание основной области вашей системы.

Вспомните, во введении мы говорили, что надежность показывает, насколько стабильно система выполняет свою конкретную задачу. Чтобы измерить надежность, необходимо понять назначение системы и лежащие в ее основе ожидания. Надежность каждого компонента инфраструктуры будет оцениваться по-разному.

Случай из практики: изучение очереди сообщений

Давайте рассмотрим пример, где оценивается надежность одного компонента системы — очереди сообщений. Как было сказано в *главе 1*, в очередь сообщений входят источник событий, очередь, брокер и потребители событий. Исходя из реализации, учитывайте следующие области при сборе информации для измерения надежности:

Хранение сообщений

Размер хранимых сообщений.

Задержка сообщения

За какое время сообщение определенного размера попадает от производителя к потребителю? В зависимости от архитектуры вашей системы вам может понадобиться несколько метрик, охватывающих один регион и несколько регионов.

Пропускная способность при передаче сообщений

Размер сообщений, скорость их передачи и приема.

Задержка при доставке потребителю

Количество сообщений, ожидающих потребления потребителем.

Загрузка канала

Количество производителей и потребителей сообщений и количество одновременных соединений, поддерживаемых системой.

Если у вас нет мониторинга, эти рекомендации станут хорошей отправной точкой, но вы и так лучше других разберетесь, какие данные обеспечат преимущества для бизнеса.

Какие оповещения следует рассылать?

В прошлом сисадмины выбирали оповещения, основанные на метриках системы (таких как недостаток ресурсов процессора и памяти или задержка запросов), а не прямом воздействии на пользователя. К сожалению, концентрация на метриках системы может привести к перегрузке и нарушению повседневной работы людей.

В крупномасштабных средах, которыми я управляла в прошлом, это были оповещения из-за отказа диска, высокой загрузки процессора или остановки одной виртуальной машины. А когда возникали серьезные проблемы с сервисом, рассылались многочисленные оповещения. Мне приходилось принимать сообщения, когда я уже выясняла причину неполадки. Все эти многочисленные оповещения мешали мне нормально работать в течение дня и не давали нормально отдохнуть ночью. В то время это считалось нормой. Мне это совсем не нравилось. Во время циклов планирования этим прерываниям не придавали значения, поэтому никто не спешил устранять соответствующие проблемы

Как выбраться из такой ловушки? Во-первых, какие аспекты являются важными для системы или систем? Какие неисправности нужно срочно устранять? Для каждого оповещения должно быть ясно, каково влияние сбоя, даже если основная причина не понятна. Представьте данные, которые привели к созданию оповещения, чтобы дежурный инженер мог предпринять соответствующие шаги.

Ваша система на основе оценки имеющейся инфраструктуры должна выдавать данные, которые вы можете использовать для измерения состояния системы в соответствующих единицах. Изучите их и определите подходящие компоненты для SLI или оценки качества обслуживания с учетом воздействия на пользователей. Качественные показатели SLI вычисляются с учетом точки зрения пользователей. Например, оценка веб-сервиса, с точки зрения пользователя, может заключаться в том, насколько успешно и быстро открывается веб-страница[1].

Применение процентилей при измерениях

Отслеживать каждое событие в системе непомерно дорого. Вместо этого делайте выборки. Выборка — это часть общей совокупности измерений.

Процентили — это обычная статистическая величина, которая разбивает выборку на сто одинаковых по размеру интервалов. Процентили являются более точными, чем средние значения, которые предполагают, что результаты измерений распределяются в соответствии с колоколообразной кривой. С помощью процентилей можно точнее передать распределение результатов измерений.

Чтобы узнать больше о выборках, обратитесь к главе 17 «Cheap and Accurate Enough: Sampling» книги «*Observability Engineering*» Чарити Мэджорс (Charity Majors) и др. (O'Reilly).

Как и в случае с другими аспектами вашей стратегии мониторинга, вы должны постоянно совершенствовать настройку оповещений по мере того, как начинаете лучше понимать работу системы.

---

[1] Если вы работаете с веб-сайтами, то должны понимать, почему время загрузки страницы имеет значение. **Pingdom.com** предлагает анализ показателей отказов на основе времени загрузки страницы (**https://oreil.ly/m82hX**).

Поговорите об оповещениях. Чем раньше, тем лучше. Когда люди устают от оповещений, они начинают игнорировать или отключать их, забывая потом за работой о выполнении соответствующих действий, что приводит к более серьезным системным сбоям.

Иногда можно избежать оповещения, предусмотрев обработку сбоев в системе. Например, ваша система может автоматически обслуживать сервис, качество которого ухудшилось, а не выдавать ошибку. Вы по-прежнему следите за метриками системы, но уже не получаете оповещения в два часа ночи, т. к. ваша система в это время исправно обслуживает клиентов.

На основе SLI можно определить приемлемую надежность и сформулировать цели уровня обслуживания (*англ.* service-level objectives, SLO). Для начала необходимо, чтобы ваши SLO соответствовали вашей текущей инфраструктуре. Затем, по мере совершенствования системы, вы можете обновлять эти SLO. SLO — это либо конкретные целевые значения, либо диапазон значений для системы, с которой вы работаете. Например, у веб-сервиса из предыдущего примера может быть такой SLO: «99% запросов веб-сервисов должны успешно завершаться менее чем за одну секунду».

Вы можете изменять показатели в большую или меньшую сторону в зависимости от текущего состояния команды. Например, для команд, которые тратят слишком много времени на трудоемкую работу по поддержанию работоспособности сайта, критически важно скорректировать SLO в сторону уменьшения, чтобы у них было больше времени на улучшение используемых систем.

Как уже упоминалось в *главе 14*, книга Алекса Идальго (Alex Hidalgo) «*Implementing Service Level Objectives*» — это отличный ресурс для изучения практической реализации SLI, SLO и бюджетов на ошибки.

Изучение платформ мониторинга

Прежние решения в части мониторинга оказали воздействие на многие современные платформы и подходы к наблюдению. Nagios (**https://oreil.ly/oDQsk**) — одна из платформ мониторинга первого поколения. Это программное обеспечение с открытым исходным кодом, которое поддерживает функции мониторинга и оповещений, а также подключаемые модули, разработанные сообществом. Многие системные администраторы разворачивали Nagios для мониторинга хостов и оповещения. Однако тогда еще не было ни готовых пакетов и конфигураций, ни инфраструктуры в виде кода, ни GitHub. Когда Марк Бюргес (Mark Burgess) представил CFEngine, этот продукт не получил широкого понимания и распространения.

Вам нужно было загрузить исходный код Nagios, настроить программное обеспечение и выполнить его сборку перед установкой. Конфигурация работающей системы была сложной, и если была допущена ошибка при настройке, мониторинг мог быть нарушен. Если не велось наблюдение за сервером мониторинга, то можно было не заметить отсутствие активного мониторинга остальной части сайта или сервисов.

Nagios нужно было настраивать для каждого узла с помощью соответствующих сервисов и определенных проверок. Такую проверку выполнял монитор событий, развернутый в системе. Со временем по мере усложнения систем ограничения Nagios вызывали недовольство пользователей. Ниже приведены некоторые из этих ограничений:

- дублирование оповещений, приводящее к избыточному числу уведомлений;
- статическая конфигурация, которую нелегко обновить в динамической среде (например, каждый раз, когда IP-адрес менялся, приходилось перезапускать Nagios);
- легко забыть о необходимости включения отключенного оповещения;
- люди забывают отключать оповещения о плановых приостановках работы;
- сложности с организацией проверок;
- отсутствие комплексных интегрированных проверок сервисов.

Даже несмотря на все эти недочеты, Nagios давал возможность сисадминам обнаруживать проблемы до того, как пользователи начинали обращаться в службу поддержки, что помогало сократить расходы на поддержку клиентов и приносило выгоды бизнесу.

Имейте в виду, что до сих пор есть много сред, использующих Nagios. Тем не менее можно улучшить систему оповещения с помощью современных интеграций с такими сервисами разрешения инцидентов, как PagerDuty (**https://oreil.ly/u35Bz**)[2].

Платформы мониторинга продолжают развиваться по мере того, как сообщество делится рекомендуемыми практиками. Это приводит к усовершенствованию платформ, появлению новых практик и специальных инструментов, помогающих сосредоточить внимание на определенных аспектах процесса мониторинга. Платформы становятся все более сложными и быстро меняются, поэтому для получения актуальной информации и руководств по эксплуатации обращайтесь непосредственно к ресурсам, связанным с этими платформами.

Не существует единого решения для всех этапов процесса мониторинга, поэтому убедитесь, что решения, которые вы рассматриваете, нацелены на нужный этап, с учетом области вашего проекта. Вы можете установить программное обеспечение и управлять им самостоятельно или воспользоваться размещенными решениями, например, такими:

- сбор метрик (например, Prometheus, Graphite, InfluxDB, Datadog, Azure Monitor, AWS CloudWatch и Google Cloud Operations);
- визуализация (например, Grafana и Kibana);
- управление оповещениями (например, PagerDuty, Opsgenie, VictorOps и xMatters);
- управление журналами (например, Splunk и Humio);
- анализ данных (например, Google Data Studio).

---

[2] Одна из интеграций — Perl Wrapper Nagios Integration for PagerDuty (**https://oreil.ly/v9fBp**).

Выбор инструмента или платформы для мониторинга

Выбор инструмента или платформы для мониторинга может быть сложной и эмоционально непростой задачей. Решения для мониторинга, размещенные на сервере, могут казаться более дорогими по сравнению с собственными. Частные лица могут опасаться нарушения требований безопасности, поскольку в журналах могут присутствовать личные данные, не отфильтрованные должным образом. Когда ИТ-директор или технический директор, прослушав убедительную маркетинговую презентацию, посвященную наблюдаемости, решает, что организация будет внедрять мониторинг на основе конкретного инструмента, они могут быть не в курсе того, что уже имеется. Получив какой-либо запрос сверху, ваша главная задача — выяснить, что есть в наличии, и работает ли этот инструмент.

Предположим, вы управляете собственной внутренней системой мониторинга. В таком случае можно обратиться к руководству с просьбой выделить бюджет (ресурсы и время) для оценки и устранения узких мест, мешающих вашей команде разработать современную стратегию мониторинга с использованием доступных решений. Работа по обслуживанию собственной системы мешает вам сосредоточиться на системах, которые приносят выгоды вашей организации. Кроме того, этот подход требует специальных знаний об имеющемся решении для мониторинга, которые трудно использовать в отрасли (что затрудняет поиск новых возможностей применения таких навыков).

Для внедрения платформ мониторинга (как собственных, так и размещенных на сервере) вам, скорее всего, придется комбинировать разные инструменты, а не использовать один-единственный, чтобы добиться всех желаемых результатов. Существуют две абсолютные истины:

1. Не выбирайте инструмент только потому, что его используют все остальные.
2. Прежде чем выбрать инструмент, определите конечный результат и желаемое поведение.

Если вы собираетесь отслеживать, отображать и анализировать данные с фиксированных мониторов (т. е. когда определенные события соответствуют известным пороговым значениям), задайте себе эти вопросы при оценке платформ мониторинга:

- Как собираются метрики?
- Какая модель данных используется для сбора метрик и как они хранятся? (Не делайте предположений о том, как хранятся данные. Когда происходит определенное событие, которое вы отслеживаете, это не означает, что данные о нем сохраняются в таком виде, что можно точно описать его в последующем, запросив данные метрик.)
- Можете ли вы запросить необработанные данные? Нужно ли вам учить другой язык и похож ли он на языки, с которыми уже знакома ваша команда?
- Как происходит устаревание данных?

- Какие интеграции вам нужны? Есть ли у вас сторонние сервисы, с которыми должен работать инструмент? Существуют ли готовые интеграции, например со Slack или PagerDuty?
- Можно ли расширить функционал инструмента с помощью подключаемых модулей или примесей?
- Для мониторинга приложений: поддерживают ли языки, используемые в ваших приложениях, инструментирование? (Даже если разработчики отвечают за инструментирование написанного ими кода, вам все равно нужно понимать, какие процессы происходят и как наблюдать за ними. Возможно, вам потребуется руководство по тому, что инструментировать и как использовать конкретное средство.)
- Как быстро события обнаруживаются, собираются, сокращаются в количественном отношении и представляются?

Одна из проблем, о которой следует помнить, рассматривая коллекцию данных платформы мониторинга, — это разрешение данных. Вам необходим мониторинг с разрешением менее секунды при создании платформ и сервисов, от которых клиенты ожидают реакции в течение секунд. Если ваша система мониторинга может получать данные только с интервалом в одну минуту, выборка будет необъективной и не отразит пользовательский опыт.

Другая проблема заключается в том, что для выявления и устранения проблемы может потребоваться подключение и контекстуализация данных из различных приложений, систем, регионов или центров обработки данных. Время относительно, особенно если не работает NTP (сетевой протокол службы времени), и временные метки для подключенных событий сильно различаются.

Если вы рассматриваете размещенные услуги, обратите внимание на следующее:

- интеграция с системами управления конфигурацией;
- эфемерные экземпляры могут быть дорогостоящими, когда оплата взимается за каждый экземпляр или период времени, в течение которого производится подсчет экземпляров;
- возможно ли тестирование интеграций;
- разделение производственных и непроизводственных систем.

Что касается визуализации, не ограничивайтесь тем, что предоставляет ваша платформа мониторинга. Например, с помощью R и D3 можно создавать визуализации, имея доступ к необработанным данным. У каждого инструмента есть сильные и слабые стороны в зависимости от того, что вы отслеживаете.

Возможно, вы захотите перечитать *главу 10*, чтобы применить навык обработки информации для передачи смысла и достижения желаемых результатов с помощью ваших систем.

Заключение

В этой главе была представлена основа для оценки и планирования проектов мониторинга в соответствии с потребностями вашей организации. Обдумайте, какие задачи предстоит решить: выявление проблем, совершенствование процессов, управление рисками, базовые модели поведения, определение бюджета и управление мощностями. Достигнуть всех этих целей в рамках одного проекта нереально. Уменьшите сферу охвата проекта, сообщите об особенностях заинтересованным сторонам и включите оценку тех инструментов, которые уже используются.

Надежная система должна работать устойчиво и не беспокоить вас без необходимости. Показатели уровня обслуживания (SLI) помогут вам определить необходимые эталонные метрики и результаты с учетом воздействия на клиентов. При наличии правильных SLI вы можете определить цели уровня обслуживания (SLO), которые устанавливают ожидания в отношении того, насколько хорошо работает система. SLO могут предоставить ценную информацию, позволяющую вам точно знать, что высокие результаты, необходимые клиентам, и технические сведения, которые получает ваша команда, соответствующим образом согласованы.

Существует множество продуктов для мониторинга систем. Если ваша организация использует собственные инструменты, возможно, пришло время рассмотреть современный подход. Кроме того, подумайте, как можно объединить различные инструменты, чтобы и вы, и заинтересованные стороны получили полную информацию о надежности и устойчивости вашей системы.

Управление данными мониторинга

Пятьсот лет назад моряки узнали, что с помощью выбрасываемой за борт веревки можно определить скорость движения судна. На веревке завязывали узелки на одинаковом расстоянии друг от друга, а к ее концу привязывали бревно. Его выбрасывали в море и считали, сколько узелков окажется за бортом корабля за 28,8 секунды, что позволяло рассчитать его скорость. Показания (количество узлов) записывали в вахтенный или судовой журнал.

Такие журналы становились важнейшим справочным источником, в них заносилась ежедневная информация и значимые события, включая скорость, курс, астрономические наблюдения, погодные явления, сведения об экипаже, посещенных портах и записи о техническом обслуживании, что крайне важно для безопасного плавания в открытом море. Навигаторы использовали эти журналы и в дальнейших путешествиях. Наряду с данными о текущих погодных условиях они помогали им держаться правильного курса и благополучно достигать желаемого пункта назначения. Наконец, судовые журналы служили в качестве официального доказательства, если происходило какое-либо несчастье.

В современных вычислительных системах не используются веревки с узлами, но «бортжурналы» — неизменное дополнение к метрикам и трассировке. Так же как моряки использовали морские журналы для записи наблюдений за скоростью и определения местоположения корабля при путешествии через океаны, вы следите за состоянием систем с помощью метрик, журналов и трассировки, чтобы, как «навигатор» этих систем, отслеживать их состояние и делать прогнозы. Эта глава поможет вам управлять данными мониторинга (метриками, журналами и трассировкой), что является важной частью вашей работы. Вы сможете ориентироваться в настоящем и будущем состоянии системы и получите исторический контекст при отладке события.

Что такое данные мониторинга?

Раздается телефонный звонок, время за полночь. Происходит критический сбой, и команда уже несколько часов пытается разобраться с проблемой, но ничего не получается, и вас зовут на помощь. С чего начать?

С данных мониторинга. *Данные мониторинга* — это все события, относящиеся к вашим системам, которые, по вашему мнению, необходимо собирать. Они могут существовать в виде метрик, журналов и трассировки. Могут быть временными или

сохраняться на диске, могут поступать в постоянном режиме или изредка. Как обсуждалось в *главе 14*, существует множество причин для сбора этих данных.

Эффективное управление данными мониторинга требует, чтобы данные были неизменяемыми, т. е. чтобы их нельзя было бы модифицировать после создания; чтобы была четко определена политика в отношении сроков их хранения и чтобы нужные данные были доступны в случае необходимости.

При внедрении метрик, журналов или трассировок приходится идти на компромиссы.

Метрики

Метрики — это количественное определение свойств событий, представляющих интерес. Большинство метрик системы — это числовые значения с меткой времени, представленные в виде счетчика или индикатора. Например, вы можете собирать данные о количестве запросов в секунду для веб-сервиса, чтобы измерить популярность сайта.

Индикатор (*англ.* gauge) отображает значение на данный момент и ничего не говорит вам о данных предшествующих измерений.

Счетчик (counter) показывает суммарное значение, характеризующее события, происходящие с какого-то момента в прошлом. Когда он достигает своего верхнего или нижнего предела, происходит сброс показаний. Например, вы можете использовать счетчики для измерения за промежуток времени и сброса через определенный временно́й интервал. Можете также сбрасывать счетчики при наступлении определенных системных событий (например, перезагрузки) или по запросу.

Давайте рассмотрим различия между индикатором и счетчиком. Спидометр автомобиля — это индикатор, который показывает, медленно или быстро вы едете. Вы руководствуетесь этой информацией для незамедлительных действий, чтобы соблюдать скоростные ограничения. С другой стороны, одометр автомобиля — это счетчик, который показывает, какое расстояние вы проехали. Вы используете эту информацию для проведения профилактических работ, таких как перестановка шин и замена масла.

Внимательно изучите, какие типы метрик предоставляют те или иные платформы мониторинга, поскольку от этого зависит, как будут собираться и храниться данные об интересующих вас событиях. Например, если платформа сокращает объем данных или агрегирует их слишком рано, она может предоставлять недостаточно информации для отладки. С другой стороны, если сокращение объема и агрегирование данных происходят слишком поздно, трафик от систем мониторинга может захлестнуть вашу сеть, что повлияет на ее производительность и качество обслуживания.

Журналы

Журналы — это неизменяемые записи событий с временны́ми метками, информация в которые может лишь добавляться. Ведение журнала позволяет сохранить историю действий в системе. Такие события, как запуск и завершение работы сис-

темы, запуск и остановка сервиса и сетевая активность, являются примерами действий, регистрируемых в журналах. Когда вам нужно узнать, что произошло на компьютере, вы полагаетесь на журналы, чтобы получить эту информацию. Вот несколько примеров.

- Во время загрузки ваша ОС согласно правилам проверяет наличие устаревшего и неприменимого оборудования — флоппи-дисководов, модемов, принтеров, факсов — и выводит «предупреждение» или «ошибку», если такое оборудование не обнаружено.
- Задача планировщика запускается каждые 10 минут и регистрирует в системном журнале статус выполнения.
- Фоновый процесс добросовестно регистрирует ошибку в конфигурационном файле тысячи раз в день.

Как правило, журналы неструктурированы. Формат файла не предоставляет контекст или значения для полей. У всех разные представления о том, какие действия следует записывать в файл журнала и какую структуру он должен иметь. Хотя для каждого языка или приложения существуют определенные соглашения, люди не всегда следуют им.

Рассмотрим, например, метки времени. Приложения ведут журнал, используя различные форматы даты (например, ГГ/ММ/ДД, ММ/ДД/ГГ, ДД/ММ). Кроме того, существуют часовые пояса и поправки, связанные с переходом на летнее время в отдельных странах. Стандартная работа по восстановлению последовательности событий из различных файлов журналов напоминает поиски сокровищ, только вместо лопаты на помощь приходит синтаксис регулярных выражений вашего любимого языка сценариев. Более того, метки времени событий точны лишь настолько, насколько позволяет формат даты: трудно восстановить быструю последовательность событий из множества источников, если дата представлена с точностью более секунды.

Вы можете узнать о различных форматах журналов из статьи на сайте Graylog «Log Formats — A (Mostly) Complete Guide» (**https://oreil.ly/Xim6V**).

Структурированные журналы

Структурированные журналы имеют формат «ключ-значение». Их проще обрабатывать компьютеру, но сложнее понять людям. Изменения в конфигурации приложения могут повлиять на то, какие поля будут отображаться, но не изменят существующие сценарии для анализа журналов.

Структурированные журналы позволяют приложениям использовать произвольный текст для описания событий, но требуют последовательного использования определенного списка полей с едиными типами данных. Даты, например, кодируются в формате UTC с точностью до микросекунды и выше. Программное обеспечение для управления журналами обрабатывает временны́е метки, представляя их в удобном

для пользователя формате. Это способствует также дополнительной экономии пространства при хранении данных: текстовая временна́я метка типа «Thursday, May 4, 2017 6:09:42 AM GMT-04:00» имеет длину 42 символа, а 1 493 908 962 000 микросекунд с начала эпохи UNIX можно представить как 4-байтовое десятичное целое число.

Первые сисадмины осваивали беглое чтение текстовых файлов журналов, но это не подходит для сложных систем, которые вы обслуживаете сегодня. Во-первых, отдельные файлы журналов слишком несогласованны. Поддержание инструментов для анализа таких текстовых потоков становится задачей, которой нет конца. Хуже того, журналов просто слишком много, чтобы кто-то мог прочитать их все.

Современные операционные системы предоставляют платформу ведения журналов, состоящую из индексной базы данных со структурированными полями: `journald` в дистрибутивах Linux с поддержкой инструмента `systemd`, Apple System Log (ASL) в macOS и Windows Event Log в Windows.

Трассировка

Трассировка — это специализированная форма ведения журнала, которая позволяет заглянуть в работающую систему. След (*англ.* trace) — это богатый набор данных, который рассказывает (упорядоченную) историю события в системе. Примерами инструментов, обеспечивающих трассировку, являются `strace` и `tcpdump`.

Распределенная трассировка

Распределенная трассировка — это специализированная форма трассировки, которая позволяет приложению предоставлять богатые журналы и метрики, охватывающие различные системы для объединения контекстных данных по всем системам.

Рассмотрим (упрощенную) последовательность событий, связанных с ответом на запрос пользователя, отправленный на современный веб-сайт какого-нибудь продукта:

1. Браузер преобразует URL-адрес сайта в IP-адрес.
2. Браузер отправляет веб-запрос (например, HTTP, HTTP/2 или QUIC).
3. Сервер отвечает (например, код 200 в случае успеха, 400 при ошибке на стороне клиента или 500 при ошибке внутреннего сервера), используя статические ресурсы, такие как изображения, файлы CSS и JavaScript.
4. Браузер начинает формировать страницу для просмотра.
5. Браузер инициирует дополнительные запросы.

Технически подкованный пользователь может использовать трассировку на основе браузера, чтобы пройти по всем элементам такой транзакции, получив информацию о том, сколько времени заняло выполнение каждого этапа запроса, и о любых зарегистрированных ошибках.

Для вашей системы вам нужен инструментированный развернутый код, который расскажет вам, что происходит во время обработки запроса. Нет гарантии, что вы сможете воспроизвести в собственном браузере то, что видит пользователь, и вам может не встретиться технически подкованный пользователь, который сможет предоставить вам данные подробной трассировки на основе браузера.

OpenTelemetry (**https://opentelemetry.io**) — это коллекция инструментов, API и SDK с открытым исходным кодом для обеспечения телеметрии в программном обеспечении. В OpenTelemetry спан (*англ.* span) — это одна именованная и привязанная ко времени операция. Несколько спанов образуют след.

Выберите типы данных

Сбор, хранение и изучение данных мониторинга позволят вам понять, что делают ваши системы и что вы хотите от них получить в дальнейшем. Поскольку большинство метрик представляют собой числовые значения, они оптимизированы для хранения, обработки и анализа, в том числе при больших объемах данных. В результате метрики отлично подходят для отображения на информационных панелях, демонстрации исторических тенденций и общего состояния системы, но предоставляют лишь ограниченное количество контекста, что минимизирует общий объем ресурсов, необходимых для их хранения. Кроме того, они имеют единый формат, что позволяет легко понять закономерность их изменения во времени: собрав метрики за неделю, вы можете оценить, какой объем хранилища понадобится в дальнейшем.

Журналы дают больше контекста для собираемых данных, но занимают больше места в хранилище и требуют больше времени на обработку. Данные трассировки содержат наибольшее количество контекста на событие и требуют наибольших ресурсов для хранения. Как при ведении журнала, так и при трассировке данные могут быть потеряны. Библиотеки журналирования, использующие протокол Reliable Event Logging Protocol (RELP), могут повысить надежность доставки сообщений, но это связано с увеличением затрат на инфраструктуру и может привести к дублированию сообщений.

Повышение надежности за счет использования протоколов типа RELP может быть оправдано только для финансовых данных, а также информации, связанной с выставлением счетов и оплатой, или для соответствия нормативным требованиям.

Выбирая наиболее подходящий тип данных, подумайте о следующих аспектах.

- Измерьте ежедневный объем данных, учитывая пиковые показатели.
- В течение какого времени вам необходимо собирать данные?
- Как вы собираетесь использовать полученные данные?
- Нужен ли мониторинг в реальном времени? (Требуется низкая задержка.)

Сохранение данных журналов

Данные журналов, если их не ограничивать, будут увеличиваться в объеме, пока не заполнят диск, на котором хранятся. Для решения этой проблемы традиционно использовались следующие подходы к управлению журналом:

- регулярная ротация файлов журнала;
- сжатие старых журналов;
- удаление журналов, когда их размер или возраст достигают определенных пороговых значений.

К сожалению, некоторые сервисы не очень хорошо работают с ротацией журналов и продолжат записывать данные в «старый» журнал, если это разрешено. Поэтому ваши инструменты ротации журналов должны предусматривать возможность перезагрузки программного обеспечения при ротации журналов, если это необходимо. Ротация журналов обычно выполняется раз в день, и если работа приложения нарушается, диск может оказаться заполненным до того, как будет запущена ежедневная очистка, что потребует ручного вмешательства для решения проблемы.

Современные системы управления журналами справляются с ротацией журналов автоматически, позволяя администраторам устанавливать простые правила, такие как «использовать не более 10 Гбайт» или «оставить не менее 5 Гбайт свободного места». Более того, эти политики соблюдаются постоянно, поэтому если сервис внезапно генерирует поток событий журнала, система принимает необходимые меры, гарантируя постоянное выполнение указанных правил.

Анализ данных журналов

В традиционных журналах поиск конкретного слова означал открытие журнала в текстовом редакторе или использование инструмента командной строки, такого как `grep`, для поиска событий в файле. Конечно, процесс замедлялся по мере разрастания файлов журнала. Фильтрация событий по временны́м интервалам, хостам или другим критериям поддерживалась, но нужно было уметь использовать регулярные выражения для построения сложных фильтров поиска.

Если платформа ведения журналов представляет собой базу данных, вы можете выполнять запросы в режиме, аналогичном использованию операторов SQL `SELECT`, используя любое из индексированных полей в качестве фильтров. В индексной базе данных данные организованы таким образом, чтобы обеспечить эффективный произвольный доступ к информации, разбросанной по большому объему данных о событиях, например: «Показать все события со степенью серьезности WARN или выше со вчерашнего дня» или «Показать события, предшествовавшие предыдущей перезагрузке». Вы даже можете выполнить отладку задним числом, запросив детальный просмотр записанных событий.

Мониторинг данных в масштабе

Теперь, когда вы собрали все эти данные, вам нужна основа для понимания того, что делают ваши системы.

Очень важно записывать и предоставлять правильную информацию. Как было сказано в *главе 14*, наблюдаемость — это свойство системы: насколько глубокое представление о работе системы вы можете получить исходя из имеющихся данных мониторинга. Улучшение наблюдаемости ваших систем — это итеративный процесс. Если это ваш первый проект по мониторингу в данной системе, в качестве руководства у вас будет только опыт участников команды и интуиция. Это нормально. По мере накопления опыта работы с системой обязательно включайте эти знания в свой конвейер данных.

Анализ ситуации после инцидента — отличная возможность для оценки данных мониторинга. Учитывайте следующие факторы.

- Сколько проблем было устранено в результате просмотра данных мониторинга? Это ваши победы: собранные данные сэкономили вам время и силы.
- Исходя из того, что вы уже собираете, насколько больше можно было бы сделать? Это возможности представления данных. Когда это имеет смысл, добавьте дополнительные информационные панели, оповещения или другие визуализации, чтобы извлечь больше пользы из данных, которые у вас уже есть.
- Какие шаги по устранению неполадок пришлось предпринимать ситуативно из-за отсутствия необходимых данных? Можете ли вы каким-то образом получить доступ к этим данным? Возможно, события собираются, но не сохраняются, или имеются дополнительные настройки, чтобы повысить уровень детализации журналов. Если ваша организация разрабатывает программное обеспечение, о котором идет речь, может быть, следует добавить дополнительные функции журналирования для более эффективного обнаружения или предотвращения такого рода инцидентов?

Хранение и анализ данных мониторинга требует значительных затрат, особенно в случае использования журналов и трассировок, поэтому регулярно проверяйте, что данные все еще представляют ценность. Возможно, вы сможете сэкономить деньги, собирая меньшее количество данных или делая больше выборок.

Контроль доступа и управление данными — еще одна ключевая проблема для зрелого проекта по управлению журналами. Возможно, вы начнете с агрегирования журналов для своей группы сисадминов: каждый из них будет иметь полный доступ ко всем данным. В этом случае у вас не будет жестких требований в части контроля доступа, но если ваша система успешно работает, экономя ваши усилия, о ней узнают все. Вскоре и другие захотят воспользоваться данными преимуществами.

Журналы часто содержат конфиденциальные данные, и предоставлять их на всеобщее обозрение нецелесообразно. Требования конфиденциальности будут определять, кто может получить доступ к журналам, содержащим личную информацию. В зависимости от характера данных и вашей среды, вероятно, будут применяться и

политики, касающиеся срока службы данных. Вам могут разрешить хранить одни журналы только в течение некоторого времени или потребовать, чтобы другие журналы хранились как минимум какое-то время.

Заключение

Данные мониторинга — это чрезвычайно важная информация о ваших рабочих системах, предоставляющая историческую запись событий, а также успешных действий и сбоев, которые произошли во время эксплуатации. Эффективный мониторинг системы предусматривает сбор нужных данных, их хранение в подходящем виде, автоматическое удаление данных, утративших актуальность, использование систематизированных методов для анализа и представления полученной информации о работающей системе.

Метрики — это данные, получаемые с индикаторов и счетчиков, которые дают представление о работе системы в определенный момент времени. Они могут быть представлены в виде графиков, чтобы отразить тенденции, закономерности и аномалии, проявляющиеся с течением времени. Журналы — это история событий, которая ведется с помощью программного обеспечения. Данные трассировки — это особая форма журналов, предоставляющих более детальную информацию о конкретных аспектах работы системы. И журналы, и метрики полезны для анализа поведения ваших систем, понимания первопричин инцидентов и прогнозирования дальнейшего развития систем.

Данные мониторинга, как и любые другие данные, нуждаются в продуманном управлении. Поскольку архив данных мониторинга со временем увеличивается, необходимо обращать внимание на то, какой объем памяти он занимает. Используйте эти данные для изучения инцидентов, происходящих в ваших системах. Анализируйте, как данные помогли решить проблемы, и выявляйте недостатки, которые необходимо устранить. Правильная интерпретация инцидентов, охватывающих множество систем, может стать настоящим вызовом, но хорошо отлаженный подход к мониторингу данных облегчит эту работу.

ГЛАВА 17

Мониторинг вашей работы

> Не путайте трудные вещи с ценными. Многие вещи в жизни трудны. Но даже если вы прилагаете большие усилия, это еще не значит, что вы работаете над достижением большого результата.
>
> – *Джеймс Клир*

Работая в должности инженера по эксплуатации, я сталкивалась с проблемой, которая ставит в тупик многих в нашей отрасли: недостаточная видимость. Обидно, когда на твою работу не обращают внимания или тебя неправильно понимают. Чтобы донести свои мысли до других людей, нужно выстраивать аргументацию, используя правильные метрики. Я никогда не рассматривала строки кода или количество решенных проблем как показатели качества своей работы.

В нашей отрасли начинают осознавать важность устойчивых систем. Здоровье системы влияет на здоровье человека, а от здоровья работников, в свою очередь, зависит качество управления системами. В этой главе я показываю, как важно вести мониторинг своей работы, чтобы вы могли повышать ее эффективность, — так же, как и систем, находящихся в вашем ведении. В результате вы станете действовать более последовательно, будете выбирать более надежные решения и получать устойчивые результаты — как в отношении системы, так и себя.

Зачем вести мониторинг своей работы?

Мониторинг вашей работы способствует четкой координации и помогает вам совместно с другими специалистами выявлять области, требующие приложения усилий. Кроме того, визуализация результатов вашей работы позволяет представить ее в контексте, а не просто в виде списка выполненных задач и проектов.

К сожалению, внешние обстоятельства зачастую вынуждают вас заниматься второстепенными делами, в то время как нужную работу приходится выполнять в неподходящее время. Иногда мотивируют грядущие перспективы. Ваша личность и самооценка часто оказываются неразрывно связаны с конкретной работой, однако новые возможности для роста не просматриваются.

Благодаря мониторингу вы понимаете, когда работа превращается в скучную рутину. Выполнение одного и того же задания снова и снова может привести к стагнации, особенно если эта работа вам не нравится. Это может запросто произойти, если вы хорошо справляетесь со своими задачами, а никто другой не берет на себя ответственность за них.

Стагнация — это когда вы в течение десяти лет выполняете какую-то работу, но при этом не можете назвать себя опытным профессионалом. Когда вы проходите собеседование по поводу новой должности, потенциальные работодатели хотят видеть, что у вас есть десять лет опыта в различных областях, демонстрирующего ваш рост, а не опыт выполнения рутинных повторяющихся задач.

Как показано на рис. 17 1, визуализация вашей работы помогает лучше увидеть, как ваши навыки и таланты соотносятся с той работой, которой вам нравится заниматься. Вы увидите, в каких областях следует повышать квалификацию и оттачивать свое мастерство. Поймете, почему вы недовольны своей нынешней работой, когда нет возможности заниматься интересным делом.

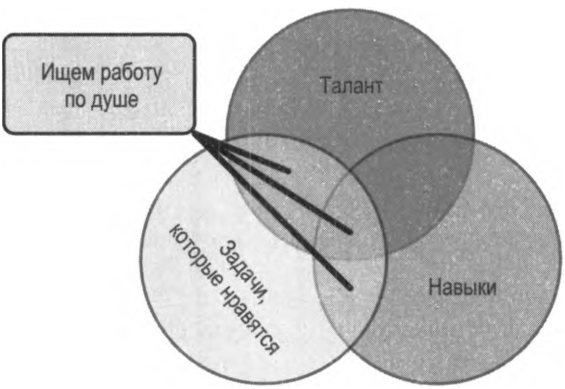

Рис. 17.1. Поиск работы, которую вам нравится выполнять

Работая системным администратором, вы сталкиваетесь с множеством видов работ. Выбирайте свой путь осознанно и не наступайте на горло собственной песне в угоду обстоятельствам и срочным делам.

Когда вы отслеживаете свою работу, то лучше видите свои успехи и эффективно используете время[1]. Это также дает вам ощущение свободы:

- чувство контроля и самостоятельности;
- выбор той работы, которая имеет значение;
- ответственность;
- определение компетенции и опыта.

На что влияет мониторинг работы

Мониторинг вашей работы может оказывать влияние не только на вас. Обратите внимание на следующие аспекты:

- в процессе мониторинга своей работы вы укрепляете отношения и доверие на уровне команды, наблюдая за прогрессом своих коллег в выполнении важнейших задач и проектов.

[1] Узнайте, как лучше управлять своим временем из книги Томаса Лимончелли (Thomas Limoncelli) «Тайм-менеджмент для системных администраторов: хватит работать сверхурочно, пора работать рационально» (Time Management for System Administrators: Stop Working Late and Start Working Smart, O'Reilly).

- Кроме того, визуализация работы оберегает команду от необходимости проявлять героизм при авралах — благодаря измерению важных параметров и поддержке устойчивых практик;
- что же касается уровня компании, мониторинг вашей работы помогает давать оценки на уровне команды, смещает фокус внимания с сисадминов на преимущественно совместную работу и обеспечивает визуальную обратную связь.

Управляйте своей работой по методу канбан

Существует множество различных способов отслеживать ход работы и делиться информацией об этом. Канбан — один из них. Это японское слово, означающее «визуальный сигнал». Такое название получила система управления рабочими процессами, разработанная Тайити Оно (Taiichi Ohno), инженером-технологом, работавшим в компании Toyota в 1940-х годах.

Применение принципов метода канбан для организации собственной работы отличается от того, как этот метод используется в рамках команды. Ни один человек не должен диктовать команде, как следует вводить этот подход в практику. Напротив, командные процессы должны учитывать потребности каждого человека, а сотрудники должны быть вовлечены в разработку и внедрение специальной доски, которая будет показывать, как движется их работа.

Канбан дает понимание, на каком этапе проекта вы находитесь в данный момент и каково текущее состояние вашей работы, а также предоставляет гибкие механизмы для внедрения изменений. При персональном использовании данного метода достичь указанной цели вам помогут следующие правила:

- начните с того, что вы делаете сейчас;
- будьте готовы осуществлять постепенные изменения.

Основные принципы метода канбан служат руководством для организации и управления работой:

- визуализируйте свой рабочий процесс, отмечая свою работу на доске;
- ограничьте количество незавершенных работ (work in progress, WIP), чтобы сосредоточиться на завершении начатых дел;
- управляйте потоком своей работы, чтобы отслеживать ее ход и понимать, насколько быстро вы справляетесь с задачами;
- постоянно совершенствуйтесь, оценивая свою работу, чтобы определить области для улучшения и смягчить ограничения, которые снижают скорость работы — так называемые узкие места.

Разбейте работу на этапы, которые вы сможете выполнить за определенное время. Помните, что это всего лишь приблизительные оценки. Никто не ожидает, что вы сможете точно предсказать будущее. Вместо того чтобы указывать конкретные цифры, используйте оценочные показатели, наподобие размеров одежды. Это поможет более точно оценивать время выполнения задач. В табл. 17.1 показан пример разбивки работ по объему.

Таблица 17. 1. Приблизительные размеры для задач

Размер	Срок выполнения
XS	< 1 часа
S	< 4 часов (максимум 1 день)
M	< 8 часов (максимум 2 дня)
L	< 20 часов (максимум неделя)
XL > L	Это уже не задача, а проект

Например, добавление пользователя в систему будет задачей XS. Добавление нового пользователя во все системы и сервисы в среде может быть задачей категории S, в зависимости от сложности вашей среды. Создание нового сервиса потребовало бы разбивки на несколько заданий, включая создание учетных записей для входа в систему — работа весьма объемная или даже выходящая за рамки простой задачи.

У проектов более широкая сфера охвата, и у них будет уже другой набор размеров. В табл. 17.2 показано, как я определяю размер проектов.

Таблица 17.2. Приблизительные размеры для проектов

Размер	Срок выполнения
S	> 1 недели
M	> 1 месяца
L	> 1 квартала
XL	> L

Представьте себе, что к 1 января должно быть выполнено критическое обновление. По вашим оценкам, это проект размера L. Исходя из этого, вы не выполните требования, если будете ждать до 10 декабря, чтобы начать работу. Благодаря визуализации этого требования вы можете перенести на более позднее время несрочные и не очень важные запросы, которые могут привести к срыву сроков.

Когда появляется дополнительная работа и команда ориентируется на размеры задач и проектов, это порождает обсуждения с внешними заинтересованными сторонами для определения приоритетности тех или иных мероприятий — в зависимости от объема текущей работы.

Как только вы сможете определить приблизительный объем предстоящей работы, ориентируясь на ее размер, то будете готовы перенести задачи на карточки. Создайте карточку для каждой задачи. Приведите на ней информацию о задаче (например, название, предполагаемый объем, категория работы). Для разных типов задач можете использовать карточки разных цветов. Можете также добавить дополнительную информацию, например о ценности работы (преимущества для бизнеса, удовлетворение запросов клиентов, ориентированность на сотрудников).

Кроме того, поговорите с заинтересованными лицами на стороне бизнеса, которые помогут вам понять, какие критически важные параметры они отслеживают. А затем определите, как классифицировать свою работу, чтобы измерить это влияние.

Потребности бизнеса и потребности клиентов часто отождествляют. Но бывает, что бизнес интересуют вещи, которые не представляют непосредственного интереса для вашего клиента. Кроме того, вы можете обнаружить, что имеется критически важная работа, которой не придается значения в вашей компании. Если вы будете игнорировать все вещи, которые считаете важными, то не будете счастливы. Очень важно найти такую работу, которая принесет наибольшую пользу компании и будет оправдывать прилагаемые вами физические и психологические усилия.

Вы можете начать с таблицы с тремя колонками («Запланировано», «В работе» и «Готово»), которые представляют основной рабочий процесс:

Запланировано

Работа, которую вам предстоит выполнить. Эта колонка отражает ваш список дел.

В работе

Вся работа, которую вы начали (но еще не завершили).

Готово

Завершенная работа

Со временем вы, возможно, захотите изменить колонки своей доски, чтобы точно измерять свой рабочий процесс и выявлять области, требующие улучшения. Начните с размышлений о том, как вы выполняли задачи и проекты. Этапы работы должны быть отражены в колонках таблицы. Может оказаться, что задачи и проекты находятся в разных фазах выполнения. Например, вы регулярно сталкиваетесь с тем, что ваша работа тормозится из-за кого-то еще, и хотите отслеживать это узкое место (например, сколько вам приходится ждать, пока другой человек выполнит следующие этапы задачи, или единовременное число заблокированных задач).

Теперь, когда у вас есть доска и карточки, наклеивайте заметки в соответствующие колонки в зависимости от фазы задачи. Вы можете просматривать свои накопившиеся дела, а также активные и завершенные работы.

Ну, вперед! Когда кто-то просит вас выполнить какое-либо дело, вы начинаете работать над задачей или завершаете ее выполнение, а карточка, представляющая эту задачу, продвигается от одного этапа к другому.

Не забывайте отслеживать метрики, относящиеся к работе. Это могут быть метрики, которые вам приходится вручную обновлять в электронной таблице, или те, которые предоставляет выбранный вами инструмент. Через пару недель просмотрите сделанную работу и оцените, чего вы достигли. Затем подумайте о том, какие изменения можно было бы внести, чтобы повысить вашу эффективность

Выбор платформы

Существует множество инструментов для отслеживания и визуализации работы, с различными уровнями настройки: например, Trello, Atlassian Jira, GitHub Projects, Kanbanize и Microsoft Azure Boards. Они различаются по стоимости и собираемым метрикам, у них разные информационные панели и возможности API-интеграции. Если платформа не имеет встроенных метрик, измеряйте свою работу на начальном этапе вручную, чтобы получить информацию о влиянии производимых изменений.

Принимая решения о выборе инструмента для командной работы, помните, что ни одна платформа или технология не станет идеальной, особенно если ее должны использовать несколько команд. Иногда приходится идти на компромиссы в рабочем процессе исходя из допустимых затрат и прозрачности в масштабах организации. Руководство должно осознавать эти аспекты, по возможности помогая людям ориентироваться в программных продуктах, и независимо от выбранных инструментов понимать важность работы, которую выполняют сотрудники.

Например, Scrum, как правило, не очень подходит для сисадминов, но многие пытаются заставить команду работать по этой модели, потому что Scrum уже используется в организации. Инструменты, навязывающие Scrum-стиль управления проектами и задачами, будут раздражать, если команда еще не использует этот подход.

Вы не обязаны следовать стандартным рекомендациям, поскольку у каждого сисадмина есть свой круг обязанностей и собственные подходы к работе. Однако примите во внимание следующие факторы.

Бюджет

Если платить за инструмент приходится из своего кармана, рассмотрите бесплатные варианты (например, GitHub Projects или Google Sheets).

Существующие инструменты

Если ваша команда уже внедряет инструменты для отслеживания работы, используйте те, что уже есть. Дополнительно можете задействовать другие инструменты, упрощающие визуальное представление данных о вашей работе.

Функции

Все инструменты различаются по функционалу. Одни лучше подходят для сбора метрик и визуализации. Другие ориентированы на такую важную функцию, как наличие API для интеграции.

В прошлом я использовала API, предназначенные для интеграции, чтобы извлекать данные из Bugzilla и направлять их в Leankit. Хотя базовые концепции в Leankit (Board, Cards и User) и Bugzilla (Product, Bug и User) различаются, я написала скрипты для преобразования данных в Task, Project и Goal. Без API у меня не было бы возможности преобразовать одно в другое, и пришлось бы ждать, когда Leankit предоставит такую функциональность.

Вот ряд дополнительных вопросов, которые помогут вам принять решение.

- Можете ли вы запросить необработанные данные? Нужно ли вам учить другой язык, чтобы получить доступ к необработанным данным? Нет смысла загружать

систему большим количеством данных, если вы не сможете выводить их потом в удобном виде.

- Данные не устарели?
- Какие интеграции вам нужны? Приходится ли вам работать со сторонними сервисами? Существуют ли готовые интеграции?
- Можно ли расширить функционал инструмента с помощью подключаемых модулей или примесей?
- Какие отчеты и информационные панели вам доступны?
- Можете ли вы отнести задачу к нескольким категориям (например, с помощью какого-либо тега)?

Даже если дополнительные отчеты или информационные панели не доступны непосредственно через платформу, иногда достаточно возможности извлекать необработанные данные. А для создания нужных отчетов или информационных панелей можно применять другие инструменты.

Поиск интересующей информации

Как только вы начнете вести мониторинг своей работы с помощью соответствующей системы, можете приступать к анализу собранных данных. И снова, в зависимости от инструмента и решения для управления проектами, которые вы используете для мониторинга, вам будут доступны различные метрики и визуализации.

Давайте посмотрим на интересные данные с канбан-доски: «Запланировано», «В работе», «Выполнено» (табл. 17.3).

Таблица. 17.3. *Типы метрик для канбан: «Запланировано», «В работе», «Выполнено»*

Тип	Определение
Скорость	Время, которое требуется для перевода задания из состояний «Запланировано», «В работе» и «Выполнено», также известное как «Время выполнения»
Нагрузка	Количество задач в разделе «В работе», также известное как WIP (количество незавершенных работ)
Пропускная способность	Общее количество заданий, выполненных за единицу времени
Эффективность процесса	Работы в процессе выполнения/скорость (закон Литтла)

Собирая метрики (такие как скорость, пропускная способность, нагрузка и эффективность процесса), вы можете обнаружить интересные факты.

Изменчивость

Отслеживание скорости выполнения задач каждого типа поможет выявить области, в которых наблюдаются отклонения. Для задач с небольшими отклонениями вы можете лучше оценить время, которое потребуется для выполнения запроса.

Отслеживание типов выполняемой работы поможет контролировать застои в работе. В этом случае уменьшение разнообразия работ с течением времени является сигналом к тому, чтобы рассмотреть переход на другую работу в целях карьерного роста.

Слишком много работы, которая находится в процессе выполнения

Отслеживание нагрузки покажет, насколько часто вам приходится переключать внимание из-за меняющегося контекста, что влияет на скорость выполнения работы и общую производительность при выполнении задач.

Баланс тщательности и эффективности

Всегда приходится искать компромисс между ресурсами (время и усилия), которые вы затрачиваете на исследование проблемы и подготовку к работе (тщательность), и ресурсами (время, усилия и материалы), которые вы тратите на выполнение работы (эффективность). Это известный принцип компромисса между эффективностью и тщательностью (*англ.* efficiency-thoroughness trade-off, ETTO)[2]. Когда вы отдаете приоритет пропускной способности, эффективность важнее тщательности. Когда вы нацелены на качество результата, на первый план выходит тщательность.

Если вы будете тратить слишком много времени на размышления о том, как решить проблему, у вас может не хватить времени на выполнение работы, и вы пропустите входящие запросы. С другой стороны, если вы действуете слишком быстро и без достаточного обдумывания, то у вас может не хватить информации для выполнения нужной работы или вы окажетесь плохо подготовлены к ней.

Продумайте и проанализируйте, какие корректировки необходимы для каждой задачи, чтобы подойти к ее решению системно. Обратите внимание на то, как задачи распределяются по категориям, учитывая следующие моменты:

- особые типы задач, требующих больше времени;
- задачи, выполнение которых часто тормозится. Ищите узкие места;
- задания, при выполнении которых работа обычно сворачивается. При получении подобных запросов понижайте их приоритет и срочность;
- задания, которые вам понравились (или оказались проблематичными). Если вы можете по-разному классифицировать задачи или указывать свое отношение к ним, то сможете определить психологическое напряжение для определенной рабочей нагрузки и влияние этих задач на общий объем выполненных работ;
- время, потраченное на обучение или отработку определенных навыков. Вы можете более целенаправленно совершенствовать свое мастерство в той или иной области.

---

[2] Эрик Холльнагель (Erik Hollnagel) «The ETTO Principle: Efficiency-Thoroughness Trade-Off: Why Things That Go Right Sometimes Go Wrong» (Boca Raton, FL: CRC Press, 2009).

Заключение

Наладив мониторинг своей работы, вы, возможно, захотите внедрить подобный подход и в рамках своей команды. Поработайте с командой над тем, чтобы представить работу каждого сотрудника в наглядном виде.

В *главе 21* я рассказываю немного больше о том, как вести мониторинг работы всей команды. Чтобы выполнить работу, вам необходимо объединить множество людей и учитывать разные точки зрения. Если все смогут делиться своим ви́дением, это поможет визуализировать зависимости между командами для лучшего понимания проблем, предотвращения задержек в работе или проведения ненужных мероприятий.

Дополнительные ресурсы

Узнайте, как наладить работу и повысить свою производительность, из книги Доминики Деграндис (Dominica Degrandis) «Сделать работу видимой: как выявить расхитителей времени и оптимизировать процессы» («Making Work Visible:Exposing Time Theft to Optimize Work and Flow», IT Revolution Press, **https://oreil.ly/va8cC**).

ЧАСТЬ V

Масштабирование системы

В заключительной части этой книги мы рассмотрим, как подготовить систему к масштабированию (будь то расширение или уменьшение системы). Нелегко понять, что и когда нужно менять в системе. Конечно, опыт помогает наметить правильный подход в различных средах, но если полагаться только на него, ваши решения могут оказаться необъективными. В итоге это приведет к ошибочным действиям. Не пытайтесь достичь совершенства, всякий раз отыскивая стопроцентно верное решение. Лучше создайте в своей системе защитные механизмы, которые выручат вас, когда вы сделаете неправильный прогноз. Благодаря им ваша система сможет уверенно адаптироваться к потребностям людей, которые являются ее частью.

Ландшафт ожиданий пользователей изменился. Четко видно те области, где достигается удовлетворение или неудовлетворение потребностей клиентов. Чтобы сохранить доверие пользователей (в том числе потенциальных) и получить таким образом возможность развивать свои системы, вам необходимо следующее:

- планирование емкости;
- гибкая практика дежурств;
- хорошо отлаженный механизм реагирования на инциденты, когда вы обнаруживаете проблемы (или когда их обнаруживают ваши пользователи, что хуже);
- руководство, которое приветствует и расширяет возможности обучения.

Управление мощностями

Управление мощностями — это процесс увеличения производительности систем в зависимости от потребительского спроса и преимуществ для бизнеса при минимизации затрат для людей, поддерживающих эти системы. Исторически сложилось так, что при настройке систем доступа реального времени системные администраторы стремились обеспечить минимальную задержку для них или сокращение времени выполнения заданий для систем пакетной обработки. В современных средах системные администраторы могут сосредоточиться на масштабировании пулов ресурсов в самоуправляемых центрах обработки данных, приложений в облачных сервисах или того и другого в гибридных средах.

В этой главе я дам определение таких терминов, как «мощности» и «управление мощностями», и представлю схему, которая поможет вам разобраться с процессом планирования мощностей. Вы сможете определять приоритетность различных инженерных задач, связанных с управлением мощностями.

Что такое мощности?

Прежде чем дать определение управлению мощностями, необходимо поговорить о том, что такое мощности. Это понятие выходит за рамки абсолютных значений, характеризующих производительность процессора, объем дисков или памяти. В него входит также объем выхода продукции, поддающийся количественной оценке, который можно получить при соблюдении стандартов качества и производительности.

Мощность — это не точно измеренное значение в системах, а скорее приблизительное, основанное на имеющейся у вас информации. Со временем вы начнете понимать, как ваши клиенты используют системы. Этот опыт позволит вам точно настраивать показатели, которые позволяют судить о мощности вашей системы.

Существуют различные варианты измерения мощности. Они зависят от определенных метрик, которые важны для системы, находящейся в вашем ведении. Существует несколько способов определения мощности при описании систем:

Проектная мощность

При проектировании архитектуры системы или ее оценке вы определяете максимально возможный объем выхода продукции, используя имеющиеся у вас инструменты или предшествующий опыт. Эта оценка характеризует проектную мощность данной системы. Например, вы можете протестировать производи-

тельность веб-сайта и установить, что он поддерживает тысячу одновременных входов пользователей.

Производственная мощность

Когда ваша система будет работать в реальных условиях, вы сможете измерить фактический максимально возможный объем выхода продукции (с учетом всех эксплуатационных ограничений) — это и будет производственная мощность. Когда система работает, вы можете производить наблюдения на основе данных об использовании сайта в обычных рабочих условиях, чтобы лучше оценить возможности системы. Например, пользователи начинают сталкиваться со значительной задержкой в развернутой вами системе, функционирующей в промышленной среде, еще до того, как число одновременных входов пользователей приблизится к тысяче: производственная мощность сайта составляет восемьсот одновременных входов пользователей.

Эффективная мощность

Когда ваша система находится в обычных рабочих условиях и на нее накладываются реальные ограничения (воздействие сезонных факторов или экономических событий), то максимально возможный объем выхода продукции покажет эффективную мощность. Например, в период рождественских распродаж вы замечаете каскадное ухудшение характеристик системы, в результате чего ее эффективная мощность составляет триста одновременных входов пользователей.

При описании мощности уточните, какой из показателей — проектную, производственную или эффективную мощность — вы измеряете или анализируете. Ограничения по мощности — это ресурсы, которые ограничивают объем выхода продукции системы. Они позволяют вам продумать вероятные сценарии отказов. Их иногда называют узкими местами, и, как правило, именно там в первую очередь возникают неполадки в системе. Ограничения по мощности в вашей системе могут быть связаны с зависимой службой, конкретными аппаратными ресурсами или наличием людей для выполнения работы. Исходя из риска наступления события, вы можете считать ограничение приемлемым или принять меры для смягчения последствий.

Модель управления мощностями

Управление мощностями — это одна из областей инженерии, на которой сисадмины имеют возможность сосредоточиться, когда они не перегружены тяжелой работой. Некоторые работы, связанные с управлением мощностями, являются повседневными, а другие относятся к средне- и долгосрочным проектам по проектированию и планированию.

Снижение эксплуатационных расходов не является целью управления мощностями, хотя оно может стать результатом применения качественных методов. Цель управления мощностями — найти баланс между затратами на ресурсы и удовлетворением запросов потребителей с помощью определенных действий:

- накопление знаний с течением времени, чтобы направлять рост и спад;
- оценка наличия людей и ресурсов для поддержки новых проектов и изменений в текущих проектах;
- выявление периодических циклов, связанных с праздниками, специальными мероприятиями, налоговыми сезонами и выборами.

При управлении мощностями очень важно понимать, какие преимущества для бизнеса предоставляет система, которой вы управляете. Отсутствие практики управления мощностями приводит к нарушению сроков, упущенным возможностям и оттоку клиентов.

Посмотрите на рис. 18.1, где показаны четыре ресурсные составляющие процесса управления мощностями.

Рис. 18.1. Модель управления мощностями

Давайте рассмотрим эти различные компоненты более подробно, начиная с закупок.

Закупка ресурсов

Процессы закупок в компаниях различного масштаба варьируются в зависимости от режима их работы и структуры. Небольшие компании могут платить больше за оборудование или ресурсы из-за небольших объемов заказов, но у них меньше бюрократических проволочек при утверждении решений, в то время как в крупных компаниях в этот процесс могут быть вовлечены нескольких групп, которые должны дать одобрение, прежде чем заявка фактически начнет выполняться.

При планировании центра обработки данных учитывайте накладные расходы, долгосрочные затраты на оборудование и ограничения логистической цепочки. Облачные решения обеспечивают повышенную надежность, но вы можете столкнуться с неограниченными комплексными затратами.

Сложность настройки в центре обработки данных и в облаке сильно различается. Например, сравните время, которое требуется на поставку и настройку оборудова-

ния в центре обработки данных с практически мгновенной доставкой у поставщика облачных решений.

Независимо от вашей инфраструктуры задайте себе следующие направляющие вопросы.

- Какие уровни производительности и доступности вам требуются? Изменяются ли они?
- Будут ли затраты на статические экземпляры или серверы выше затрат на автоматическое масштабирование от месяца к месяцу? От года к году?
- Нужно ли предусматривать мощности, чтобы справиться с всплесками активности или скачками обычной нагрузки?

Обоснование

Понимание процесса закупок, через который вам нужно пройти, дает информацию для обоснования закупок ресурсов. При длительных сроках поставки работа по обоснованию закупок ресурсов может проводиться задолго до того, как они вам понадобятся. Если ресурсы предоставляются в кратчайшие сроки, к обоснованию можно приступать, когда вы будете готовы к проведению соответствующих мероприятий по развертыванию ресурсов. Как и в случае с закупками, процессы, проводимые в рамках организации для обоснования закупки ресурсов, варьируются. Обоснование может быть как детальным, так и очень формальным. Актуальность предложения определяет комиссия.

Даже если среда не требует формального проведения проверки, важно иметь эту информацию под рукой, чтобы лучше понимать, почему было принято определенное решение, какие варианты рассматривались и в итоге не были одобрены. Обстоятельства меняются, и решение, которое когда-то посчитали неподходящим, будет представлять интерес для очередного проекта или даст новое направление для текущего проекта. Кроме того, будет нелишним отметить, почему то или иное решение было отвергнуто — возможно, оно имеет какой-то принципиальный недостаток, и другим тоже стоит воздержаться от его использования.

Задокументируйте следующие аспекты.

- Опишите свою проблему, учитывая, что у читателя не будет представления о сопутствующих обстоятельствах.
- Опишите возможные решения.
- Объясните свой выбор, например: «Используя данный ресурс, я ожидаю, что определенный показатель (поддающийся измерению) улучшится на столько-то, что позволит увеличить доход (приведите ориентировочную оценку)».
- Предоставьте вспомогательные данные для ответа на вопросы «почему» и «сколько».
- Укажите другие возможные ограничения и риски, препятствующие достижению успеха.

Управление

Управление ресурсами охватывает весь их жизненный цикл, от развертывания до вывода из эксплуатации, и зависит от типа ресурсов и степени автоматизации. Жизненный цикл ресурса поможет вам наметить комплекс действий в соответствии с бизнес-целями.

Как показано на рис. 18.2, при использовании оборудования управляемой физической инфраструктуры вы планируете выделение, конфигурацию, развертывание и последующий вывод из эксплуатации. Эти вопросы нужно обдумать еще до приобретения оборудования.

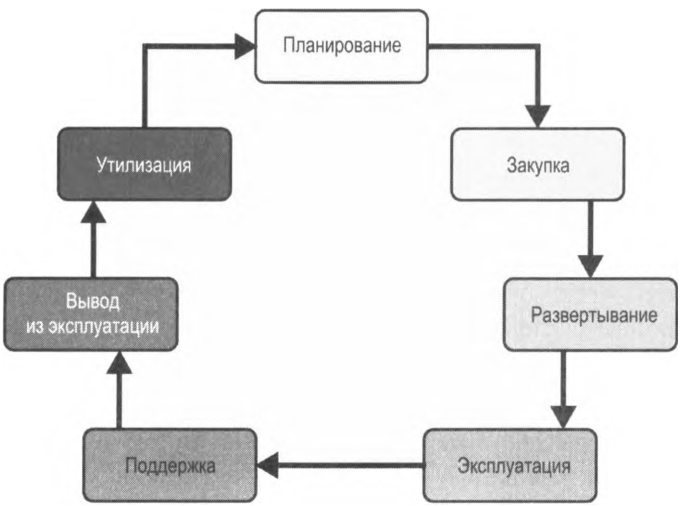

Рис. 18.2. Жизненный цикл единицы оборудования в физической инфраструктуре

Ниже приведены этапы жизненного цикла единицы оборудования:

Планирование

Вы планируете закупку оборудования, учитывая, какое пространство оно занимает, рассчитываете мощность систем охлаждения и питания с учетом имеющегося оборудования.

Закупка

Выбрав оборудование, вы решаете, покупать или арендовать его, исходя из наличия оборудования и цен поставщика, в соответствии с вашим планом. Построение прочных отношений с поставщиками серверов, систем хранения данных и сетевого оборудования поможет вам получать продукцию по выгодным ценам и необходимую поддержку.

Развертывание

После получения оборудования необходимо убедиться, что системы поставляются в соответствии со спецификацией. За физическое развертывание в стойках может отвечать другая команда, или же эта работа может входить в ваши должностные обязанности.

Вы устанавливаете нужную ОС и обновления. Интенсивность отказов оборудования характеризуется обычно U-образной кривой: дефектные компоненты выходят из строя на ранних этапах своего жизненного цикла. Можете провести ряд стресс-тестов и убедиться, что система не выйдет из строя раньше времени, что она ведет себя, как положено, и нет никаких различий в производительности компонентов.

Наконец, вы устанавливаете и развертываете необходимое программное обеспечение и сервисы, чтобы запустить систему в работу.

Эксплуатация

Вы обновляете ОС, модернизируете при необходимости любое оборудование, чтобы предоставлять требуемые сервисы.

Поддержка

Вы осуществляете мониторинг оборудования, следите, чтобы не было проблем, и выполняете ремонт, обеспечивая функционирование сервисов. Это может означать координацию поддержки, а при необходимости физическую замену неисправного оборудования на новое.

Вывод из эксплуатации

Вы определяете, когда оборудование больше не нужно, и выводите из эксплуатации работающие системы. Может потребоваться длительное время, чтобы выявить любой доступ к системе.

Иногда новое оборудование вводится в эксплуатацию для замены старого. Если масштаб архитектуры системы можно увеличить путем добавления нового оборудования, а затем уменьшить за счет удаления старого, процессы вывода из эксплуатации и развертывания упрощаются, а воздействие на конечного потребителя оказывается минимальным. Если вам придется выключить систему, чтобы полностью удалить оборудование, это окажет определенное влияние на конечных пользователей.

Утилизация

После прекращения поддержки программного обеспечения в системе и вывода его из эксплуатации (и если оно больше не подходит для использования в вашей организации в каком-то ином качестве) необходимо утилизировать оборудование. Вы должны позаботиться о том, чтобы в системе не оставалось конфиденциальных данных. Может потребоваться изучить специальные законы и правила, касающиеся утилизации.

При определении потребностей в оборудовании считается, что срок службы неспециализированного оборудования составляет от трех до пяти лет. Отчасти это связано с развитием технологий, удешевляющих стоимость эксплуатации нового серверного оборудования. На это влияет также усовершенствование системного программного обеспечения: старое оборудование может не поддерживать текущие операционные системы.

При использовании специализированного оборудования, такого как устройства хранения данных, жизненный цикл немного меняется, поскольку затраты могут

составлять от десятков тысяч до почти миллиона долларов. Кроме того, обслуживание и поддержка — это отдельные статьи расходов и более долгосрочные инвестиции.

Нередко ИТ-отделы организационно входят в состав финансовых отделов, и бухгалтерские графики амортизации используются при формировании ИТ-политики.

Организации могут использовать другую стратегию начисления амортизации, существуют также особые юридические/налоговые правила, которым необходимо следовать, но интервалы от трех до пяти лет соответствуют общим принципам расчета амортизации дорогого оборудования.

Планирование сбоев как составляющая часть управления мощностями

Рассмотрим U-образную кривую с крутыми краями и ровной серединой, показанную на рис. 18.3.

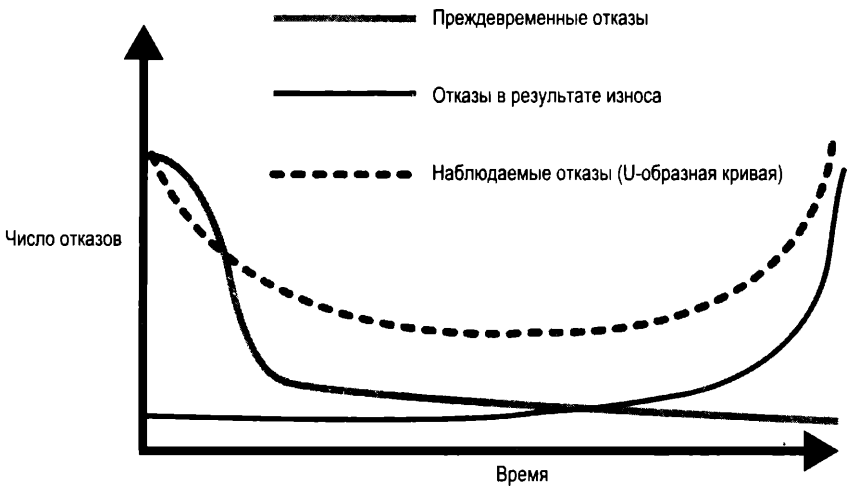

Рис. 18.3. График зависимости числа сбоев оборудования от времени с преждевременными, постоянными и износовыми отказами

Эта модель показывает распределение сбоев оборудования во времени. Отказ физических ресурсов случается обычно на одном из трех временных интервалов: преждевременный отказ из-за брака, случайный отказ при регулярном использовании или отказ в конце срока службы из-за износа. Частота сбоев будет повторять форму этой кривой. Большое число проблем возникает на ранних этапах (поэтому проводите для оборудования стресс-тесты, выявляющие такие проблемы вне производственной среды) и в конце срока службы (следите за возрастом оборудования и заранее планируйте его списание).

Даже если система все еще выполняет полезную работу, как старый автомобиль, необходимо оценить, не выгоднее ли вместо текущего ремонта заменить ее, полностью исключив сбои.

При внедрении качественного управления оборудованием в стратегию развития инфраструктуры возникают проблемы, связанные с персоналом, доступностью инструментов и сложностью гибридных сред. В командах инженеров по эксплуатации часто недостаточно работников, из-за чего не хватает времени на разработку качественных методов эффективного управления оборудованием. Это приводит к за-

держкам с поступлением и развертыванием оборудования и несвоевременному выводу из эксплуатации устаревших систем.

Еще одна проблема связана с отсутствием инвестиций и недоступностью качественных инструментов. Проектирование центров обработки данных (включая системы охлаждения и питания), регулирование взаимодействия с поставщиками и инвентаризация (начиная с физического оборудования и заканчивая кабельными системами) нередко ведутся с помощью электронных таблиц. Это затрудняет совместную работу, коммуникацию и передачу знаний в рамках организации.

Гибридная среда, в которой часть инфраструктуры является локальной, а часть обслуживается поставщиком облачных услуг, создает дополнительные сложности. Все это допустимо, если собственных знаний для управления необходимыми сервисами недостаточно.

Взгляните на рис. 18.4. При использовании облачных услуг учитывайте изменения в жизненном цикле активов. За физическое размещение оборудования в стативах и стойках, безопасность оборудования, обслуживание и утилизацию систем отвечает поставщик услуг. На ваших плечах остаются следующие задачи.

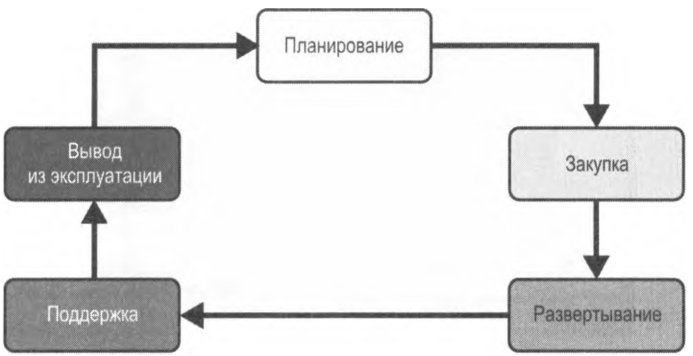

Рис. 18.4. Жизненный цикл облачных активов

Планирование

Ваша основная задача — выбрать конкретные облачные услуги (например, определенные типы компьютеров, зарезервированные мощности или мощности по требованию) и спрогнозировать бюджет.

Закупка

Теперь вам не нужно планировать все расходы одномоментно. Вместо этого вы устанавливаете бюджеты для каждого работника или команды, чтобы увязывать расходы и покупательную способность в рамках организации. Вы налаживаете отношения с различными поставщиками облачных услуг и определяете совместимые услуги, соответствующие бизнес-требованиям.

Развертывание

Вместо физического развертывания серверов вы пишете код инфраструктуры для программного предоставления, проверки и развертывания необходимых облачных ресурсов.

Поддержка

Благодаря тщательному мониторингу используемых систем вы выявляете области, где возможна экономия средств. Вы оцениваете, отслеживаете и устраняете уязвимости в программном обеспечении и на нижележащих уровнях — в зависимости от используемого сервиса. Вы также можете быть основным контактным лицом при взаимодействии с поставщиком услуг и координации поддержки.

Вывод из эксплуатации

Вместо того чтобы пытаться получить максимум от физических узлов, поддерживая их работу в течение трех-пяти лет, настройте экземпляры так, чтобы они работали только до тех пор, пока это необходимо. Исключите облачные ресурсы, которые работают, но не приносят выгоды. Вы можете настроить политику отключения и вывода из эксплуатации ресурсов, которые больше не используются.

Миграция в облако может снять часть нагрузки с инженерно-эксплуатационных служб, позволяя им сосредоточиться на методах управления инфраструктурой. Однако, когда быстрое предоставление ресурсов не вызывает затруднений, очень важно визуализировать используемые ресурсы, чтобы предотвратить дорогостоящие ошибки.

И наконец, взгляните на рис. 18.5. Бессерверные среды — особый тип облачных вычислительных ресурсов, систем хранения и сетей. При использовании бессерверных технологий жизненный цикл активов упрощается, многие этапы контролируются поставщиком услуг.

Рис. 18.5. Жизненный цикл активов в бессерверной среде

Планирование

Вам нужно провести исследование и разработать архитектуру и сервисы, необходимые для достижения желаемого результата.

Развертывание

Вам необходимо разворачивать разные конфигурации, приложения, подключенные службы и инструментарий для мониторинга.

Поддержка

Вы несете ответственность за то, чтобы ваши пользователи получали выгоду от использования ваших систем. Когда что-то идет не так, необходимо обратиться к журналам и данным трассировок для понимания и устранения проблем.

Мониторинг

Мониторинг ресурсов — это наблюдение за конкретными используемыми ресурсами для того, чтобы достичь баланса между затратами на ресурсы, требованиями клиентов и преимуществами для бизнеса. Этот аспект управления мощностями подробно рассматривается в части IV.

Схема планирования мощностей

Вам следует рассмотреть возможность документирования компонентов управления мощностями для каждой среды, поскольку основные процессы, которым необходимо следовать, будут отличаться в разных командах и организациях. Я не знаю, как они будут выглядеть в вашем случае, но могу предоставить схему, которая подскажет вам, что делать дальше, когда вы поймете процессы и политики, действующие в вашей среде (рис. 18.6).

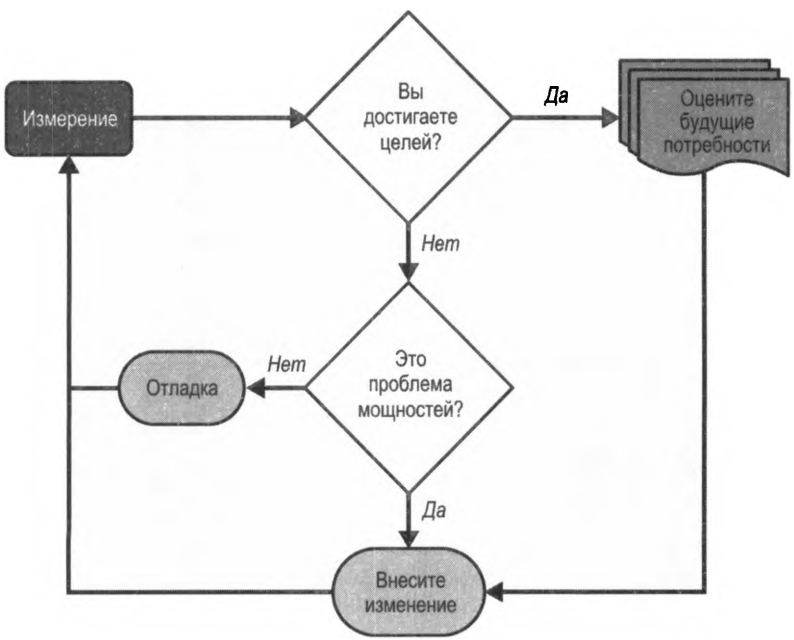

Рис. 18.6. Схема процесса планирования мощностей

Давайте рассмотрим шаги на рис. 18.6.

1. Измерьте текущую рабочую нагрузку для всех компонентов объекта в оцениваемой системе.
2. Оцените на основе запросов, достигаете ли вы SLO.
3. Если вы достигаете целей, уделите время оценке будущих потребностей (например, есть ли новая вычислительная технология, которая лучше отвечает текущим требованиям?).

4. Если вы не достигаете поставленных целей, оцените, связана ли эта проблема с мощностями. Иногда необходимо решить другие вопросы, прежде чем вносить изменения в мощности. Повышения производительности можно добиться за счет оптимизации конфигурации.
5. Если проблема связана с мощностями, определите, какие изменения можно произвести, и примените одно из этих изменений, чтобы увидеть его влияние. Убедитесь, что информация об изменении понятна соответствующей команде или командам, что поможет в принятии будущих решений. Если у вас недостаточно информации из-за отсутствия нужных измерений, внесите изменения в измерения.

Как стратегии организации влияют на планирование

Организации используют три основные стратегии для оценки будущих потребностей: опережение, запаздывание и соответствие. Любая из этих стратегий помогает получить информацию, определить приоритет действий и уменьшить трения.

При стратегии опережения вы добавляете мощности, как только получаете сигналы о том, что потребности системы будут расти. Эта стратегия часто используется при управлении локальными ресурсами. Она компенсирует невозможность быстрых изменений в случае, если спрос превышает возможности имеющихся ресурсов, а заказ и доставка оборудования занимают длительное время. Сопутствующие расходы увеличиваются, если наращивание мощностей выполнено, а спрос со стороны пользователей так и не вырос.

При стратегии запаздывания вы удовлетворяете спрос после того, как он появляется. Если вы не можете своевременно сделать это, стратегия запаздывания увеличивает вероятность потери клиентов, негативно влияет на доверие и отношение к компании. Стратегию запаздывания нереально использовать для управления локальными ресурсами в небольших компаниях из-за того, что для заказа и получения оборудования требуется длительное время. В крупных компаниях ресурсы для отдельных команд могут выделяться за счет других подразделений в рамках организации. Конфликт из-за ресурсов возникает, когда их оттягивают популярные проекты, лишенные надлежащего управления мощностями. Из-за этого некоторые команды требуют существенно больше ресурсов, чем им необходимо, предполагая, что часть из них будет потеряна. В результате могут уменьшиться финансовые инвестиции для других проектов.

Стратегия соответствия представляет собой компромисс между стратегиями опережения и запаздывания. Он достигается за счет постепенного наращивания мощности при повышении спроса. Например, можно заранее несколько нарастить мощность, чтобы частично компенсировать ожидаемое повышение спроса, а остальное оборудование вводить в эксплуатацию тогда, когда действительно возникнет такая необходимость.

Другой термин для обозначения стратегии запаздывания — *just in time* (JIT) или «точно в срок». Вместо того, чтобы поддерживать запасы компонентов на производстве, детали приобретаются по мере необходимости, в зависимости от спроса. При этом затраты снижаются, а нежелательные излишки сводятся к минимуму, что приводит к повышению рентабельности. Но для достижения такой эффективности требуется точно прогнозировать будущий спрос. Неверные прогнозы приведут к нарушению работы конвейера.

Рассмотрим экономические последствия пандемии COVID-19 для глобальных цепочек поставок. Нехватка таких товаров, как туалетная бумага, возникла не потому, что люди стали больше их использовать. Офисы и школы не нуждались в коммерческой однослойной туалетной бумаге, а потребителям нужно было больше бытовой туалетной бумаги. Производителям бумаги потребовалось время, чтобы перевести производство с коммерческих продуктов на бытовые. Тем временем полки магазинов опустели, а запасы коммерческой туалетной бумаги на складах росли.

Оценивая свои потребности при планировании мощностей, подумайте о переменчивых факторах, на основе которых делаются прогнозы. Разработайте планы на непредвиденные случаи, чтобы справиться с неожиданным изменением спроса.

Требуется ли планирование мощностей при использовании облачных вычислений?

Даже при использовании облачных сервисов необходимо разработать четкую стратегию управления мощностями. Да и для сервисов, поддерживающих динамическое масштабирование, следует предусмотреть по меньшей мере возможности управления ресурсами и мониторинга. Учитывайте следующие ограничения:

- время, необходимое на ввод новых ресурсов;
- верхние пределы выделяемых ресурсов, установленные провайдером в зависимости от выбранных типов экземпляров. Производительность процессора, сети и хранилища ограничена выбранными параметрами начальной конфигурации. В некоторых случаях данные характеристики можно изменить, но при этом вероятны простои — это зависит от поставщика облачных решений. Другие ограничения связаны с тем, что приходится обращаться к поставщику облачных служб для внесения корректировок, что может занять разное время. Вопреки идее о том, что в облаке все основано на API и делается мгновенно, поставщики услуг вводят определенные ограничения, чтобы повысить качество обслуживания в среднем;
- ограничения конфигурации управляемого хранилища данных. Облачные провайдеры создают многоуровневые предложения, которые отчасти упрощают управление базами данных, но вам может понадобиться большая универсальность. Обычно чем дороже предложение, тем больше предоставляется тонких настроек при управлении ресурсами. Но все равно придется выбирать конкретную функциональность, будь то сегментирование, репликация или балансировка нагрузки, и эти варианты могут быть очень дорогими. Планирование мощностей позволяет определить правильную конфигурацию ресурсов;
- ограниченные мощности самого облачного провайдера. На определенных масштабных уровнях невозможно добавить больше ресурсов по требованию из-за ограниченного объема доступного оборудования у поставщика услуг;
- иногда внешние зависимости имеют дополнительные ограничения и не поддерживают динамическое масштабирование. В качестве примера можно привести шлюзы и прокси-серверы.

Облачные вычисления облегчают динамическую адаптацию к реальному спросу. Технические требования могут быть более тонко настроены, чтобы лучше отражать переменчивый характер спроса и информировать персонал о последствиях внесения изменений в основную инфраструктуру.

Поставщики услуг устанавливают различные ограничения на услуги. Хотя поставщик услуг отвечает за масштабирование, люди все равно должны быть осведомле-

ны о влиянии зависимых служб и связанных с ними ограничениях. Без контроля можно легко нарушить эти ограничения (например, максимальный лимит 75 Гбайт для хранилища функций и уровней в AWS (**https://oreil.ly/jmOPe**).

Заключение

Будущее непредсказуемо. Развертывание новых ресурсов требует времени, а избыточное выделение ресурсов сопряжено с затратами. Планирование мощностей сочетает в себе искусство и науку, позволяя согласовать имеющиеся ресурсы с ожидаемыми потребностями вашей организации без ограничения возможностей вашей системы или чрезмерных затрат.

Следуйте указанным шагам, оценивая мощности своих систем:

1. Определите потребность и объясните, как конкретный ресурс удовлетворит ее.
2. Закупите необходимый ресурс с учетом сопутствующих расходов на его обслуживание.
3. Наблюдайте за ресурсом.
4. Управляйте ресурсом на протяжении всего его жизненного цикла.

Планирование мощностей важно для всех ресурсов, которые вы контролируете, включая физические и облачные системы, но специфика закупок отличается. При использовании аппаратных систем требуется время на приобретение и развертывание нового оборудования, а возможности масштабирования в большую или меньшую сторону по мере изменения спроса ограничены. Масштабирование систем, построенных на базе облачных сервисов, может выполняться автоматически, но при этом легко допустить перерасход средств, если не следить за ситуацией. Эффективное планирование мощностей требует постоянной оценки и корректировки ваших процессов.

Дополнительные ресурсы

Ознакомьтесь со следующими дополнительными ресурсами по управлению мощностями:

- Узнайте больше о планировании мощностей для веб-сайтов из книги «*The Art of Capacity Plan ning: Scaling Web Resources in the Cloud*», авторы Арун Кеджаривал (Arun Kejariwal) и Джон Оллспоу (John Allspaw), (Pragmatic Bookshelf).
- Ознакомьтесь с примером управления мощностями в компании Capital One, который привел Кевин Маклафлин (Kevin McLaughlin) на конференции Velocity в Нью-Йорке в 2016 году: «Is Capacity Management Still Needed in Public Cloud?» (**https://oreil.ly/MHR8v**).

Создание надежной дежурной службы

Самой заметной обязанностью по поддержке сервисов или систем является дежурная служба, которая реагирует на значимые события. Если ваша система оповещения постоянно выдает вам сообщения, у вас может не хватить времени или сил на эффективное улучшение инфраструктуры системы. В крайнем случае, вы можете не думать о дежурствах, когда сами не дежурите, потому что вам комфортнее выполнять плановую работу. В этой главе я предлагаю план, который позволит вам сформировать стрессоустойчивость, наметить заблаговременные и регулярные мероприятия для подготовки к дежурствам. Он позволит вам справиться с проблемами и стрессом, которые возникают в связи с необходимостью решать любые возникающие вопросы.

Что представляют собой дежурства

Дежурство — это временное назначение на определенную работу, которую вы выполняете поочередно с другими сотрудниками. Например, вы должны находиться на связи в нерабочее время (по вечерам, выходным и праздникам), чтобы отвечать на запросы службы поддержки и принимать соответствующие меры при получении оповещений. Когда вы находитесь на дежурстве, то являетесь одним из лиц, отвечающих за эту работу в течение определенного периода времени. В зависимости от размера команды и распределения обязанностей, дежурства обычно представляют собой 8- или 24-часовые смены один раз в неделю или в две недели.

Обязанности дежурного в различных организациях могут варьироваться в широких пределах. Это может быть работа, связанная как со сбоями служб приложений, так и с перебоями в подаче электроэнергии. Вы можете оказаться тем человеком, который принимает меры при отключении сервисов или эскалирует проблемы в соответствующие подразделения. Вам может потребоваться выяснить посреди ночи, почему не работает веб-сайт, или восстановить данные с резервных копий при сбое файлового сервера. Некоторые дежурные назначаются для решения эпизодических проблем — так, на всякий случай. В других случаях оповещения приходят настолько часто, что их обработка занимает все время. Часто работа дежурного и другая незапланированная работа сливаются в одну рабочую очередь.

Существует множество факторов, вносящих сумятицу в практику дежурств, из-за чего сисадмин вынужден заниматься реактивной работой и лишается возможностей для профессионального роста. Два основных фактора — неверная оценка важности и приоритета проблем.

Когда люди считают проблему очень серьезной, они могут потребовать ее устранения, даже если существует приемлемое временное решение. Если проблему считают не очень важной, ее приоритет может быть понижен, что затронет множество людей. У эксплуатационной группы могут возникнуть трудности с определением приоритета. Ниже приведены некоторые неудачные практики:

- автоматическая установка высокого приоритета для всех возникающих проблем;
- невозможность ранжирования поступающих проблем;
- дублирование отчетов, относящихся к одной и той же проблеме;
- неспособность устранить известные проблемы, чтобы исключить возможность дублирования отчетов.

В идеале срочность запроса и влияние проблемы признаются и разделяются, включая следующие аспекты:

- Сколько людей затронуто?
- Есть ли подходящее временное решение?
- Подвергаются ли данные риску?
- Каково влияние на деятельность вашей организации?
- Каковы последствия для вашего клиента (клиентов)?

Давайте обсудим, какие инструменты и методы помогут вам повысить стрессоустойчивость, рассмотрев особенности работы дежурного специалиста.

Человеческие факторы процессов при дежурстве

Мне приходилось заниматься этой работой. Оповещения, приходящие поздно ночью, недосыпы. Годами я просыпалась в панике, думая, что пропустила оповещение. Я не ходила в отпуск, не ела вовремя или на бегу съедала холодный кусок пиццы после командного совещания, где мы решали сложную проблему, влиявшую на доходы компании. Я пропускала семейные торжества и встречи с друзьями, а родственники уже знали, что меня могут внезапно вызвать на работу. У меня остались неприятные воспоминания о дежурствах. Эта работа оказала влияние на мои отношения, психическое и физическое здоровье. Я отказывалась от активностей, которые могли бы помочь, потому что не видела возможностей для приобретения более устойчивого опыта.

Но эта работа не обязательно должна быть такой. Хотя у вас есть обязательства перед своей компанией, вы также несете ответственность за себя и свое здоровье. Вы можете быть ответственным и внимательным работником, занятым на дежурствах, но при этом отстаивать свои интересы и поддерживать отношения с друзьями и семьей.

В следующих разделах книги я поделюсь своими рекомендациями по организации рациональной работы дежурных на сменах, начиная с подготовительных шагов,

которые вы можете предпринять еще до начала дежурства, и заканчивая дежурством и сдачей смены. Затем сравните свои процессы с теми, что я описала здесь, и возьмите на вооружение те методы, которые вам помогут.

Проверьте свою политику дежурств

В идеале политика дежурств должна быть четко документирована. Если этого не сделано, то нужно понять, какая работа вас ожидает. Для этого я задаю такие вопросы.

- Получаю ли я компенсацию за время, которое провожу на дежурстве, или когда активно работаю сверхурочно? Это включает компенсацию за дежурства и вызовы, будь то дополнительные отгулы или дополнительная оплата за необходимость быть на связи и готовность трудиться в нерабочее время.
- Как команда определяет приоритетность поступающих запросов? Как узнать, какие проблемы должны быть решены в нерабочее время? Приоритизируйте запросы, чтобы четко понимать, какие из них имеют высокую степень важности и срочности, — чтобы наладить эффективное и планомерное сотрудничество.

Используйте рис. 19.1 для классификации типов запросов. Важные и неотложные запросы имеют более высокий приоритет, чем остальные.

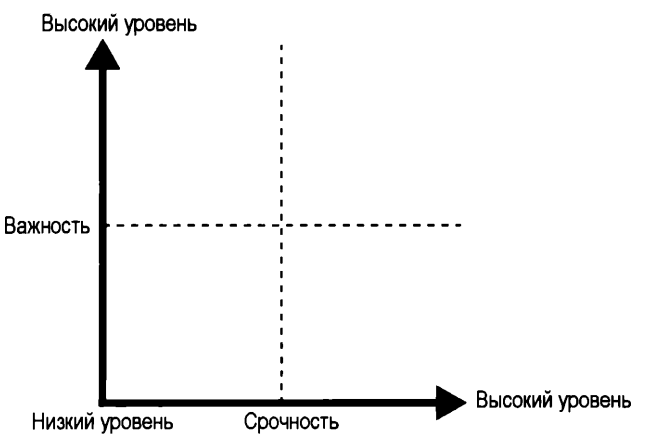

Рис. 19.1. Матрица важности и срочности

- Сколько времени у меня есть, прежде чем нужно будет дать ответ? Разные типы запросов имеют разный уровень реагирования?
- К кому еще я могу обратиться, если потребуется помощь по вопросам сетевого взаимодействия или безопасности? Что ожидают от службы дежурств профильные эксперты, особенно в тех областях, где есть только один человек с опытом?
- Кого и когда я должен уведомлять об инцидентах? Различается ли процесс эскалации в рабочее и нерабочее время?
- Сколько времени я буду дежурить?

- Сколько времени я должен быть на связи? Можно ли поменяться сменами с другим дежурным, если у меня будет важное дело?

Подготовка к дежурству

В течение нескольких недель, предшествующих вашей первой дежурной смене, убедитесь, что вы знаете обо всех системах, за которые отвечаете, и о пути эскалации, т. е. о том, к кому и когда следует обращаться за помощью. Вы должны знать об уровне доступности систем. Иногда перебои в работе в течение нескольких минут или даже секунд являются критической проблемой, в то время как в других случаях они могут не оказывать заметного влияния на клиентов, и достаточно оставить записку, чтобы кто-то занялся этим вопросом на следующий день.

Независимо от того, существует ли формальный процесс обучения на рабочем месте, когда подопечный становится «тенью» дежурного инженера в реальной рабочей обстановке (также известный как shadowing on-call), поинтересуйтесь, можете ли вы наблюдать за действиями других сотрудников команды, чтобы изучить особенности их работы. Такой способ позволяет понять, какие инструменты и процессы используются, как реагировать и взаимодействовать с командой, а также присмотреться к работе дежурного.

Вы получите представление о частоте оповещений и о том, как реагировать на них.

- Каким образом приходят сообщения о новых инцидентах?
- Имеется ли электронное или SMS-сообщение, уведомление в службе обмена сообщениями, например через канал Slack, или отчет о состоянии на информационной панели?
- Создается ли запрос в службу поддержки?
- Если да, происходит ли это автоматически или кому-то нужно создавать файл вручную?
- Насколько оперативно необходимо подтверждать запросы?
- Как быстро требуется решение?
- Если решение требует специальных знаний, какова процедура эскалации?
- Когда эскалация считается уместной?
- Какие дополнительные шаги вы предпринимаете после решения проблемы для того, чтобы она не повторилась?

Убедитесь, что ваш ноутбук и телефон заряжены, а программное обеспечение обновлено, и что вы можете получить доступ к необходимым вам сервисам из дома и тех мест, где вы находитесь во время дежурства, будь то любимое кафе, футбольное поле или велосипедная дорожка. Вне зависимости от характера ваших дежурств, у вас должна быть некоторая свобода действий. Главное, чтобы вы могли оперативно получать и подтверждать запросы и были готовы помогать в решении проблем.

Добавьте в закладки нужные вам сервисы и проверьте, что вы можете войти в систему и получить к ним доступ. Тогда при оповещении вам не придется долго

копаться, чтобы узнать, куда перейти и как получить больше данных, касающихся проблемы.

Настройте телефон и другие устройства для работы со службой оповещения. Службы имеют различные настройки политики эскалации, поэтому указывайте не только электронную почту. Например, я уделяю особое внимание оповещениям, когда нахожусь на дежурстве, поэтому стараюсь минимизировать дублирующие оповещения о той же проблеме, получая их по другим каналам. Если ожидаемое время ответа составляет 15 минут, я предпочитаю электронную почту и SMS, а через 10 минут мне поступает телефонный звонок с напоминанием, если я еще не отправила ответ. Такая конфигурация дает мне 10 минут для ответа на SMS, прежде чем я получу еще одно оповещение, что уменьшает вероятность их дублирования и позволяет дать ответ в течение 15 минут.

Хотя для команды назначается определенное ожидаемое время ответа, вы можете настроить также свои предпочтения. Необходимо учитывать требования к ротации дежурных и время ответа, а также ваш стиль работы. Найдите баланс, чтобы вы могли отвечать достаточно быстро, но при этом вас не беспокоили громкие уведомления.

Сверьтесь с политикой расходов вашей компании и поговорите с руководством о возможности приобретения дополнительных зарядных кабелей для всех ваших устройств, чтобы не беспокоиться о том, что забыли необходимый кабель. Например, я ношу в сумке дополнительные шнуры питания для ноутбука и телефона, чтобы не приходилось отключать какие-то устройства или беспокоиться о том, что забыла какой-либо кабель.

Аккумуляторы или пауэрбанки дадут вам дополнительное время для решения проблем на телефоне и ноутбуке.

Хотя не каждый раз приходится совершать или принимать телефонные звонки, будьте готовы к проведению голосовых или видеоконференций во время дежурств с использованием гарнитуры, чтобы можно было продолжать печатать на клавиатуре, не жертвуя качеством звука.

Мобильная точка доступа или Wi-Fi-модем повысят качество дежурства, позволив вам работать из любого места. Вы не будете испытывать ограничений из-за привязки к офису и всегда найдете возможность, чтобы подключиться, когда вас вызовут, и решить проблему в допустимые сроки. Благодаря мобильной точке доступа я могла наслаждаться семейными пикниками и входить в компьютерную систему прямо из парка, чтобы решить вопросы, которые занимали зачастую всего несколько минут.

Отдельное устройство позволяет использовать телефон для получения дополнительных оповещений по другим вопросам или подключения к конференциям по мере необходимости. Кроме того, это расширяет возможности подключения — если ваш телефон подключен к одному оператору, а устройство — к другому, у вас с большей вероятностью будет устойчивый сигнал.

За неделю до дежурства

За неделю до дежурства вы можете уведомить все заинтересованные команды, в зависимости от вашей работы, и обновить связанные с проектом тикеты, чтобы поделиться информацией о состоянии задач. Информация о вашем дежурстве в системе отслеживания проектов позволит свести к минимуму неожиданный стресс, который может возникнуть у сотрудников по каким-то задачам. Кроме того, заблаговременно обновив информацию, вы меньше отвлекаетесь при выполнении запланированного. Если у вас есть критически важные задачи, которые необходимо выполнить к определенному сроку, сообщите об этом своему руководителю и попросите делегировать их. Актуальное задокументированное состояние текущих дел означает, что другие могут подключиться и продолжить их, если вам придется разбираться с затянувшимся инцидентом.

При возможности отправьте тестовые оповещения, чтобы убедиться, что они вам приходят. Даже если вы выполняли такую проверку ранее, убедитесь, что изменения конфигурации не повлияли на отправку уведомлений. Я вовремя обнаружила, что службы оповещения блокировали моего оператора телефонной связи, что избавило меня от необходимости разбираться с этими проблемами и искать причины блокировки во время дежурства.

Заранее планируйте приемы пищи. На дежурствах нужно особенно заботиться о себе. Неизвестно, когда и как часто вас будут вызывать. Хотя вы можете приблизительно оценить обстановку на основе своего прошлого опыта, это не гарантия, что все будет происходить так же. Запланируйте все, что можете. Это исключит дополнительный стресс при каскадных авариях. Энергетические батончики придут на помощь, когда приходится начинать рабочий день раньше обычного и нужно быстро заставить мозг работать.

Построение взаимоотношений: есть ли у вас семья или друзья, на которых вы можете положиться, чтобы поддержать вас во время дежурств? Обращайтесь за помощью. Делитесь с людьми своими переживаниями. Вам не обязательно находиться в изоляции. Предоставьте людям возможность помочь вам. Тем самым вы наладите связи и сами сможете прийти на помощь другому человеку, когда не будете заняты на дежурстве.

Поговорите с работниками, которые смогут подменить вас. Вам приходится долго добираться на работу или регулярно посещать врача? Нужно отвезти ребенка в детский сад или попасть на футбольный матч? Требуется показать домашнего питомца ветеринару? Поговорите с младшим инженером, а еще лучше — с тем инженером, который сможет подстраховать вас. Не забывайте отвечать взаимностью, когда в поддержке нуждаются они.

Заранее договаривайтесь о том, чтобы вас заменили на рабочем месте. При организации посменной работы следует учитывать жизненные обстоятельства людей, а подход должен быть гибким. Команда, которая уже внедрила эту практику, лучше справится с кратковременными трудностями, возникающими при перебоях в работе.

Хотя вы не обязаны разговаривать с каждым дежурным специалистом, такие беседы помогут создать и поддерживать хорошие отношения и успешно организовать посменную работу без излишнего драматизма. В процессе общения я обнаруживала проблемы — например, некоторые люди не заботились о том, кто сможет заменить их во время отпуска. Мне удалось решить вопрос с нехваткой специалистов в смене и повысить качество нашей работы.

Общение также помогает, когда работники на дежурстве объединены в виртуальную группу. Специалисты из разных отделов могут не знать о навыках своих коллег, с которыми находятся в смене.

Установите контакты с остальными членами дежурной команды. В идеале в ней должны быть ведущий и младший инженеры, контактные лица, занимающиеся эскалацией проблем, и ответственный за урегулирование инцидентов. Встреча с остальными членами дежурной команды помогает убедиться, что все готовы к дежурству. Если вы отвечаете за дежурство, то получаете дополнительную уверенность в организации надлежащей поддержки.

Свяжитесь со специализированными инженерами. Хотя они могут и не находиться на дежурстве официально, у вас должны быть контакты инженеров по безопасности, сетям или базам данных, если в организации нет единого центра ответственности. Поскольку один человек не может полноценно поддерживать все дежурства, задокументируйте, о каких проблемах вы должны оповещать его.

Поговорите с членами своей семьи или соседями по комнате о предстоящем дежурстве. Объясните им, что такое дежурство, чтобы избежать недопонимания. Установите рамки приемлемого поведения (например, не устраивать вечеринки в выходные дни, когда вы находитесь на дежурстве).

Подготовьтесь к работе. Убедитесь, текущие дела не мешают вашей подготовке.

Ночь накануне дежурства

Убедитесь, что устройство, на которое приходят уведомления, заряжено, звук не отключен, и оно не находится в режиме «не беспокоить». Хорошенько выспитесь, чтобы чувствовать себя отдохнувшими — это важнейшее условие, позволяющее сохранять концентрацию внимания в меняющейся обстановке. Усталость перед началом смены снизит эффективность работы, особенно когда необходима сосредоточенность.

Люди часто забывают о создании комфортных условий работы для самих себя. Вот несколько советов от других опытных сисадминов:

- «Держите рядом с кроватью теплую толстовку или халат, чтобы не думать, где их найти посреди ночи, когда поступит оповещение». *(Сэра (Sera) (@tsdubz), 19 сентября 2021 г.)*
- «Все необходимое для быстрого приготовления чая или кофе, если вас вызовут на работу ночью». *(Ивонна Лэм (Yvonne Lam) (@yvonnezlam), 19 сентября 2021 г.)*

Подумайте о тех вещах (напитки, еда и/или одежда), которые обеспечат комфортную работу, когда вам нужно будет быстро реагировать на аварию и какое-то время заниматься ремонтом.

Ваша дежурная смена

Процесс дежурства может варьироваться в зависимости от принятых в команде правил, но общий подход включает следующие шаги.

- Получить одно или несколько оповещений:
  - подтвердить, что оповещение или оповещения получены;
  - произвести триаж;
  - исправить;
- Повышение качества дежурной службы;
  - документирование
  - мониторинг;
  - оценить нормальную работу системы.

Когда вы получаете оповещение, первое, что нужно сделать, — подтвердить получение. Подтверждение говорит людям о том, что вы знаете об аварии, и вас не будут отвлекать, чтобы еще раз сообщить те же сведения. Далее необходимо провести триаж, т. е. оценить серьезность и неотложность проблемы, и на основании этих факторов предпринять дальнейшие действия. Наконец, устраните проблему, о которой вас оповестили. При этом нужно отключить также громкие звуковые оповещения, которые продолжают рассылаться.

Оцените свою готовность к дежурствам. Инциденты с высоким уровнем влияния, для устранения которых требуется длительное время, и многочисленные часто поступающие предупреждения вызывают тревожное состояние. Возможно, для вас и для команды будет лучше, если вы передадите кому-то основное дежурство, и сделаете перерыв.

Обычно во время дежурства, когда оповещения не приходят, основное внимание уделяется повышению качества работы дежурного (а не текущим делам). В число задач может входить улучшение документации или мониторинга, а также наблюдение за системами и понимание того, как происходит их нормальная работа. Иногда в процессе проверки работающей системы вы обнаружите какие-то огрехи, требующие исправления. Убедитесь, что найденные неполадки задокументированы (в очереди работ, а также в журнале дежурств) и оповещения настроены.

Формализация процессов дежурства

Оценка готовности к дежурствам должна быть формализована в рамках процессов команды. Это помогает установить ожидания и изменить рабочие нормы. В противном случае такая работа может сказаться на здоровье человека и его отношении к профессиональной этике. Вот несколько примеров:

- если члену команды приходится работать после того, как рабочий день завершен, а решение проблемы занимает более часа, он может изменить свой график работы на следующий день;
- если на устранение проблемы члену команды требуется более восьми часов в течение рабочего дня или четырех часов в нерабочее время, то он автоматически получает выходной на следующий день.

Благодаря такой четкой политике люди знают, чего ожидать, и охотнее соглашаются на дежурства, подстраховывают других работников, когда кому-то нужен перерыв.

Сдача смены

Итак, пришел заветный час, и ваше дежурство окончено. Вы хотите завершить работу. Но еще не все сделано. Еще нужно передать смену следующему дежурному инженеру. Официальное оформление сдачи дежурства позволяет решить две задачи. Во-первых, это поможет заступающим на дежурство инженерам понять, какие проблемы и нерешенные вопросы остались с прошлой недели, чтобы настроить их на успешное выполнение задач. Во-вторых, это даст вам столь необходимую психологическую разрядку — своего рода рубеж, по достижении которого можно «выдохнуть», завершая напряженное дежурство на производстве. Это ритуал, знаменующий окончание смены, который говорит вашему телу, что все в порядке, можно расслабиться, и это великолепно.

Но это также и ритуал начала следующей смены, отправная точка для другого человека, который принимает эстафету и должен теперь быть в боевой готовности. Когда приходит ваш черед заступать на дежурство, ваши коллеги должны передавать вам дела таким же образом. В противном случае вы можете больше напрягаться из-за нечеткого понимания, какие проблемы решены, а какие еще нет — в зависимости от состояния систем, которыми вы управляете.

Вы можете подумать: «В моей среде нет таких проблем, моя среда не такая сложная, нам не часто приходят оповещения и т. д.». Но наша задача — не оптимизировать среду так, чтобы в ней все было спокойно и проблемы вообще не возникали. Мы пытаемся создать командные процессы, которые окажутся устойчивыми, несмотря на неизбежные инциденты: повреждение данных, перебои в работе центров обработки данных или поставщиков облачных решений, или инциденты, связанные с безопасностью. Четкий процесс сдачи смены настраивает вас и вашу команду на успех, если что-либо все же случается, потому что команда уже хорошо отработала передачу ответственности при решении текущих вопросов, а сотрудники хорошо отдохнули и находятся во всеоружии, готовые справляться со сложными вызовами.

Частью сдачи смены является журнал с обзором событий за неделю. Вот примерная информация, которую можно включить в журнал:

- период;
- работники, которые входили в дежурную бригаду в данный период времени;
- инциденты и соответствующие ссылки на дополнительные сведения о них;
- открытые инциденты;
- устраненные инциденты;
- инциденты, которые не были зафиксированы системой и о которых не было оповещений;
- работа, выполняемая вручную;

- возможности для автоматизации и совершенствования;
- открытые вопросы. Хотя потребителей это уже, вероятно, не затрагивает, все еще могут оставаться вопросы, на которые нет ответов;
- обращения по специфическим вопросам, которые были успешно решены и которые нуждаются в улучшении.

Журнал, в который заносятся события, произошедшие за неделю, имеет очень важное значение. Он дает мне уверенность в том, что человек, который дежурил передо мной, справился с делами и предоставил мне все необходимые сведения, а следующий дежурный будет знать, что я справлюсь с делами и предоставлю необходимую информацию.

Процедуры сдачи смены жизненно важны для эффективного сотрудничества между регионами. Хорошей практикой является проведение коротких видеоконференций при сдаче смены, когда люди, заканчивающие рабочий день, могут ввести следующую группу в курс дел. На постоянной основе можно делиться информацией о делах в системе с поддержкой тикетов, такой как Zendesk, или с поддержкой чатов, такой как Slack. Такой подход позволит региональным командам быстрее вникать в те дела, на которых остановились их коллеги. Кроме того, сведения о проблемах с возможностью поиска закладывают основу для внутренней документации, документации для клиентов, а также отчетов об ошибках для команды разработчиков программного обеспечения.

День после дежурства

Когда вы закончили дежурство, это не означает, что вам не нужно работать над улучшением дежурной службы. Пока события еще свежи в памяти (либо в день сдачи смены, либо на следующий день), еще раз взгляните на вопросы, которые вы отметили. Это лучшее время для творческих идей, которые позволяют улучшить те аспекты работы, с которыми вы недавно имели дело. Обновите необходимую документацию, устраните излишние звуковые оповещения (для некоторых имеет смысл понизить степень серьезности) и укажите, какую плановую работу можно вести для улучшения системы в долгосрочной перспективе. При любых инцидентах записывайте соответствующие сведения в отчет об инциденте.

Одним из способов, позволяющих постоянно совершенствовать систему оповещений, является регулярное рассмотрение оповещений вместе с командой, чтобы обсудить их влияние и значение.

Многообразие опыта, полученного во время дежурств

Автор — Крис Деверс (Chris Devers)

Читая эту главу, я был поражен тем, насколько подход к дежурству в моей организации отличается от того, который описан в этой главе. Работа дежурного, с которой я сталкивался за время своей карьеры, предполагала взаимодействие как с людьми, так и с системами. Например, поступала такая информация: «Только что произошел сбой в системе новостей, а мы выходим в эфир через 17 минут, помогите!» Да, при решении подобных проблем наряду с техническими аспектами присутствует и человеческий фактор, приходится много общаться с расстроенными людьми, импровизировать, чтобы урегулировать вопросы.

Мой работодатель создает решения для индустрии медиа и развлечений, где развертывание локальных серверов продолжает играть важную роль. Люди, работающие в этой области, имеют дело с такими вещами, как камеры, магнитофоны, спутниковые каналы связи, системы вещания и огромные медиаархивы, а серверы, которые мы создаем, помогают связать все это воедино, чтобы шоу, как говорится, продолжалось.

Многие из продаваемых нами систем представляют собой физические серверы, которые наши клиенты устанавливают в своих студиях, офисах и центрах обработки данных по всему миру и управляют ими самостоятельно. Администрирование этих систем является обязанностью клиента, но если у него возникают проблемы, он может обратиться к нам. Персонал службы технической поддержки у нас — это, по сути, команда сисадминов-консультантов, которые помогают администраторам на местах.

Наша компания невелика. Но у нас есть офисы по всему миру, и это стало ключевым фактором создания надежных дежурных служб. Если вещательная компания в Индии сообщает ночью о проблеме, дежурная команда в Европе готова прийти на помощь. Если решение проблемы в Европе затягивается до конца рабочего дня, дело передается американской команде, у которой рабочий день только начинается. Аналогичным образом, глобальное штатное расписание позволяет продолжать работу и в праздничные дни. Регионы корректируют время дежурства, чтобы минимизировать влияние на клиентов, когда наши региональные офисы закрываются на праздники, будь то китайский Новый год в Восточной Азии или День благодарения в Северной Америке. И когда в результате COVID-19 начали повсеместно переходить к удаленной работе, мы восприняли это спокойно, потому что уже привыкли сотрудничать с коллегами и клиентами в дистанционном режиме.

Чтобы охватить выходные дни, мы внедряем скользящий график сменной работы, которая напоминает дежурства, описанные в других разделах этой главы: люди должны следить за уведомлениями электронной почты и Slack на своих телефонах и быть готовыми в любой момент сесть за ноутбук. Или, возможно, клиент запланировал техническое обслуживание на выходные, и дежурный инженер заранее знает, как сложится его работа в субботний день. Но устранять проблемы поздно ночью приходится редко, потому что они передаются по регионам, так же как и в течение рабочей недели.

Мы также поощряем тесные рабочие отношения между нашими командами, занятыми поддержкой и разработкой, что дает целый ряд преимуществ. Служба поддержки, конечно, прекрасно осведомлена о болевых точках клиентов, но она также получает отличную обратную связь о том, какие улучшения и функционал желательно добавить в продукт. В то же время разработчики будут рады, что их работа значима не только для клиентов, но и помогает сотрудникам службы поддержки. Такое сотрудничество способствует также распространению знаний: если определенный сотрудник является признанным экспертом по отдельным аспектам продукта и делится своим набором приемов, это облегчает работу всем. Очевидно, что если постоянно отвлекаться, то невозможно сосредоточиться и довести дело до конца, и все стараются помнить об этом. Но когда преимущества такого сотрудничества оказываются налицо, привлечь больше людей к решению проблемы — гораздо проще, и это создает положительную обратную связь: квалификация команды поддержки повышается, эскалации происходят реже, а к разработчикам обращаются уже не так часто.

Каждая организация должна разработать свой подход к дежурной службе сообразно задачам, которые вам нужно решать, и ресурсам, с которыми приходится работать. В моем случае работа с глобальной командой привела к формированию рационального подхода к организации дежурств, при котором влияние проблем на клиентов минимально. Подумайте, какие творческие решения можно применить в вашей организации, чтобы наладить устойчивую работу дежурной службы.

Мониторинг процесса дежурства

Еще раз отмечу, что мониторинг нужен не только для производственных систем. Не менее важно наблюдать и за социальными системами. Необходимо вести мониторинг самого процесса дежурства, чтобы узнавать о недочетах, и предупреждать нежелательные ситуации. Тем самым вы обеспечиваете собственную защиту. Для того чтобы понять, насколько эффективны дежурства, и предоставить эту информацию руководству, которое может внести изменения, необходим мониторинг. Он позволяет собрать нужные показатели и представить эту информацию в убедительной форме. Обратитесь к *главе 11* и применяйте эти улучшения, обмениваясь данными о дежурствах.

Первое, что нужно измерить, — это объем незавершенной работы, даже если вы единственный дежурный инженер и вам не нужно отчитываться о своей работе кому-либо. В идеале работа, связанная с оповещениями, должна поступать в общую рабочую очередь. У вас должна быть возможность делиться визуализациями, отражающими вашу работу за определенное время, а когда вы вносите изменения, вы должны видеть их результаты. При первом измерении вы можете установить определенный базовый уровень, а затем наблюдать за тем, как изменения (например, увеличение количества людей на дежурстве, улучшение программного кода или инфраструктуры) влияют на работу, мониторинг которой вы ведете.

Вот несколько вопросов, над которыми стоит задуматься, приступая к мониторингу.

- Сколько часов дежурства приходится на один временно́й интервал?
- Сколько часов активного дежурства приходится на один временно́й интервал?
- Как часто поступают оповещения?
- Как часто требуется принимать меры? Бывает ли, что проблема разрешается сама собой?
- Когда в последний раз приходило оповещение?
- Когда в последний раз обновлялась документация?
- Каковы последствия отказа системы? Требуется ли отправка оповещений в нерабочее время?
- Какое число работников может быть охвачено? Если специалист получил оповещение и занят решением проблемы, кому отправлять следующее оповещение?
- Как часто работник на дежурстве отвлекается от обычных дел, включая сон, прием пищи и душ?
- Как часто приходится прерывать семейные встречи и запланированные дела? Существует множество мероприятий, которые нельзя перенести на другой день и которые имеют решающее значение для поддержания здоровых отношений.

Вместо того чтобы фокусироваться только на времени восстановления системы и времени обнаружения неполадок, используйте эти метрики, чтобы совершенствовать условия труда дежурных. Во время производственных совещаний полезно об-

суждать данные показатели, чтобы команда отмечала те мероприятия, которые будут способствовать дальнейшему улучшению наблюдаемых тенденций.

Если ваша команда периодически проводит ретроспективные собрания, поговорите о прогрессе в отношении дежурной службы. В числе возможных мер по улучшению положения дел, которые вы можете рекомендовать, — обновление расписания отправки оповещений и политики эскалации. (Если ретроспективные собрания в вашей команде не приняты, я призываю вас предложить, чтобы они проводились.)

Давайте рассмотрим некоторые другие стратегии смягчения последствий, к которым можно обратиться, когда вы обнаружили конкретные проблемы вашей дежурной службы.

Вам трудно расслабиться после дежурства

Проанализируйте глубинные причины. Связано ли это с общим количеством времени, необходимым для выполнения работы? Может быть, из-за того, что вы готовы выполнять работу до конца, вам не удается полностью восстановиться?

Смягчение отрицательных последствий в данном случае заключается в следующем. Убедитесь, что вы отслеживаете, сколько свободного времени у вас в действительности имеется, насколько вы ограничены в использовании этого свободного времени и какими делами вы можете заниматься. Если ничем не ограниченного свободного времени недостаточно, проблема в системе. Поработайте с руководством и командой, чтобы определить пути исправления ситуации. В долгосрочной перспективе это вредно для здоровья. Если оставить все как есть, это приведет к усталости, эмоциональному истощению и в конечном итоге к выгоранию. Установите для себя разумные границы.

Если это не связано с работой, обратите внимание на другие области вашей деятельности. Это нормально, если вы попросите предоставить вам условия, чтобы вы могли перестроиться.

Люди, занятые на дежурстве, не имеют достаточных знаний о системе

Здесь смягчение отрицательных последствий заключается в том, чтобы помочь персоналу развить свои навыки и больше узнать о системе. Используйте эту возможность, чтобы расширить базу знаний о системе и найти области для улучшения. Если вы недостаточно хорошо разбираетесь в системе, а ваше окружение поощряет психологическую безопасность и приветствует, когда люди делятся своими проблемами, расскажите о том, что вам нужно. Вы можете поговорить об обучении на рабочем месте, или чтобы вам дали возможность понаблюдать за другим специалистом на дежурстве, что позволит лучше понять работу системы.

Не хватает людей, чтобы организовать дежурные смены нужной продолжительности

Если вы не являетесь менеджером, это действительно сложная проблема. Документируйте издержки, которые несете вы и ваша команда. Апеллируя к этим данным, попросите выделить дополнительных сотрудников для работы, напри-

мер расширить штат или предложить другим сотрудникам компании также взять на себя обязанности дежурного. Рассмотрите также возможность понизить приоритет реагирования для дежурной службы, чтобы заниматься этой работой только в обычное рабочее время. Если вы не можете изменить принятые в системе правила, вам нужно менять работу. Когда вы измотаны, вам становится сложнее проходить собеседования в других компаниях, и вы будете по-прежнему подрывать здоровье.

Непоследовательный подход к дежурствам

Во-первых, непоследовательность сама по себе не является проблемой. Хотя вы, возможно, хотите иметь хорошую, отточенную, идеальную систему, люди будут оставаться несобранными и непоследовательными. Никто не может предугадать, как отреагирует человек в тот или иной день. Это одна из причин того, что решение проблем становится таким запутанным, когда речь идет о более крупном масштабе.

Если непоследовательный подход является результатом недопонимания или недостаточной осведомленности, внесите исправления в документацию. Если он является преднамеренным и вредит команде, есть два варианта решения проблемы. Команда с высоким уровнем психологической безопасности может договориться о том, чтобы каждый отвечал за свою работу и выполнял те функции, которые от него ожидают другие члены команды. В противном случае руководство должно взять на себя эту роль, четко сформулировать предъявляемые требования и систематически контролировать их исполнение.

Неизвестно заранее, когда придется быть на дежурстве

Время выхода на дежурство становится невозможно предугадать. Чем лучше вы и ваша команда справляетесь с обслуживанием системы в реальных условиях, тем более непредсказуемой может стать работа, особенно по мере роста использования вашей системы. Внедрение мониторинга на начальных этапах для выявления закономерностей и справедливое распределение работы позволят членам команды чувствовать взаимную поддержку, чтобы противостоять любым неожиданностям.

Недостаточная компенсация за работу, которую, как вам кажется, вы выполняете

Это выходит за рамки вашей компетенции как исполнителя. Ознакомьтесь с *главой 21*, чтобы найти идеи по смягчению данной ситуации, и поговорите со своим руководителем, если вам нужна помощь в этом вопросе. Я включила эту проблему в список, потому что когда у людей появляется такое ощущение, это может повлиять на их отношение к дежурствам и работе в целом. Вместо того чтобы перекладывать свое недовольство друг на друга, признайте, что это обоснованное беспокойство, которое может быть у каждого человека.

Заключение

Поддержка системы посредством участия в дежурствах — неотъемлемая часть управления системами, но дежурные смены можно организовать должным образом, чтобы эта работа позволяла вести здоровый образ жизни, а у человека оставалось время на друзей, семью и хобби. Выделите время и попробуйте отойти от обычной рутины, сосредоточившись на том, как можно сделать управление системами более удобным и устойчивым.

Дополнительные ресурсы

Ознакомьтесь с этими ресурсами, чтобы узнать больше о работе на дежурстве.

- «Crafting Sustainable On-Call Rotations», Рин Дэниелс (Ryn Daniels, **https://oreil.ly/eaCvN**).
- «The On-Call Handbook», Элис Гольдфус (Alice Goldfuss) и др. (GitHub, **https://oreil.ly/zJKwv**))
- Глава 10 («Notifications») в книге «The Art of Monitoring», Джеймс Тернбулл (James Turnbull, Turnbull Press, **https://oreil.ly/p6bMj**)

Управление инцидентами

Как было показано в *главе 19*, цель дежурства — быть в курсе состояния систем, чтобы поддерживать их нормальную работу. Но как бы вы ни старались снизить риск, сбои все равно возникают — происходят инциденты. Управление инцидентами начинается, когда вы обнаруживаете проблему во время дежурства, но зачастую эта деятельность выходит за рамки дежурства, когда для решения проблемы необходимо обращаться к другим профильным экспертам и командам. Целью управления инцидентами является минимизация последствий инцидента.

Вам, как исполнителю, необходимы такие инструменты, техники и методы, которые не только помогут преодолеть последствия инцидента с минимальными потерями, но и придадут уверенность в том, что вы готовы эффективно реагировать на инциденты, когда бы они ни происходили. Вам необходима хорошая, четкая коммуникация между командами, чтобы соответствующие профильные эксперты могли поделиться своими знаниями и минимизировать время решения проблемы. Вам также нужно уметь собирать и применять сведения, полученные в результате инцидента, чтобы улучшить общий уровень обслуживания, уменьшить последствия для клиентов в будущем и снизить трудозатраты для команды.

В этой главе я поделюсь методикой совместного и устойчивого управления инцидентами, начиная с их выявления и заканчивая разбором инцидентов и определением плана действий для улучшения работы системы в реальных условиях.

Я предполагаю, что ваша команда знакома с процессом управления инцидентами и что у вас уже будет некая основа, к которой вы сможете применить то, чем я делюсь, чтобы улучшить этот процесс. Если ваша команда в настоящее время не занимается управлением инцидентами, поделитесь этой книгой или *главой 21* со своим руководством.

Что такое инцидент?

Понятие «инцидент» в каждой организации понимается по-своему. Инцидентом может быть любое событие, из-за которого дежурному инженеру отправляется оповещение, или же только то, которое связано с нарушением безопасности. В этой книге я определяю инцидент как нештатную ситуацию (*исключение*) в работе *действующего* сайта, сервиса или программного приложения, которая оказывает *влияние*.

Давайте разложим это определение на составляющие, начиная с исключений. Исключения возникают, когда система ведет себя не так, как ожидалось. Это могут

быть ошибки в программном коде, сбои в базовых системах (например, DNS или сеть) или недопонимание при планировании проекта, которое привело к другой реализации.

Действующий сайт, сервис или приложение — то, что используется клиентами или заказчиками. Во многих случаях это рабочая среда для сайта или сервиса, но сюда входят также приложения, установленные на устройствах.

Влияние — это качественный эффект, который исключение оказывает на клиентов или заказчиков. Иногда это влияние может быть заметно внешне, а в другой раз остается незамеченным, и необходимо принять решение о том, разглашать или нет этот инцидент.

Ниже приведено несколько примеров инцидентов.

- В октябре 2021 года потеря IP-маршрутов к DNS-серверам крупной социальной сети привела к глобальному сбою доступа к этой сети и вспомогательным сайтам более чем на шесть часов. Когда эта система вышла из строя, она потянула за собой и другую систему, отвечающую за контроль доступа по карточкам в здания и серверные помещения, поэтому никто не мог получить удаленный доступ к серверам, а сисадмины на местах не могли попасть в здания и серверные комнаты, чтобы непосредственно приступить к устранению проблем.
- В июле 2020 года из-за истечения срока действия сертификата сервера и сбоя в передаче данных система California Reportable Disease Information Exchange не могла принимать результаты лабораторных исследований COVID-19 от партнеров, что привело к расхождениям и занижению данных о случаях заболевания.
- В октябре 2019 года компания Docker столкнулась с инцидентом, когда реестр Docker Hub не работал. Любая организация, которая полагалась на получение образов напрямую из реестра, столкнулась бы с проблемами. Те организации, которые кешировали образы Docker или имели собственный реестр, минимизировали влияние этого инцидента.
- В мае 2019 года компания Slack (**https://oreil.ly/Z18FC**) начала развертывание функции, из-за которой некоторые клиенты не могли подключиться к Slack и пользоваться мессенджером. Организации, которые были затронуты, полностью простаивали.

Как видно из этих примеров, инциденты могут различаться по степени внешнего воздействия. Кроме того, инциденты могут быть теми промахами, которые ваши клиенты (пока) не заметили.

Что такое управление инцидентами?

Управление инцидентами — нечто большее, чем просто реагирование на событие и восстановление работоспособности системы. Управление инцидентами — это процесс планирования, подготовки, реагирования, расследования и извлечения уроков из инцидента.

Взгляните на рис. 20.1, на котором показан такой цикл непрерывного обучения.

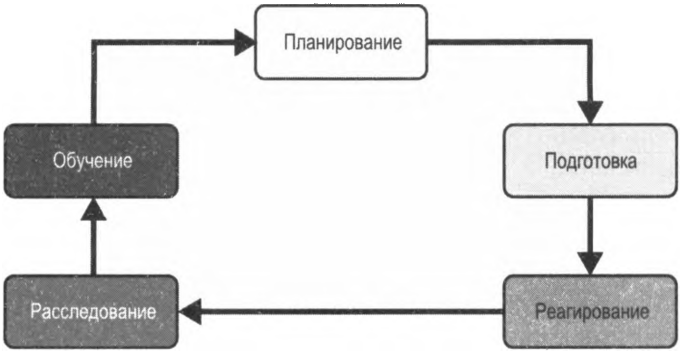

Рис. 20.1. Цикл управления инцидентами

Конечные результаты всех этих шагов по управлению инцидентами таковы:

- сокращение ущерба, затрат и времени на восстановление;
- выявление проблем с программным кодом или процессом;
- устранение проблем, чтобы предотвратить повторные инциденты;
- документирование инцидентов;
- извлечение уроков из инцидентов.

Все этапы цикла управления инцидентами основаны на общих базовых принципах, в числе которых четко определенные роли и обязанности, а также возможности для постоянного совместного обучения. Эффективное управление инцидентами позволяет получать данные, чтобы аргументировать необходимую численность персонала, усовершенствовать подготовку специалистов и наметить маркетинговые мероприятия.

Как понять, что проблема действительно в системе, а не в вас

В *главе 19* я рассказывала о мерах по обеспечению собственной стрессоустойчивости при работе на дежурстве. Тем не менее иногда некоторые аспекты работы находятся вне вашего контроля, и никакие личные качества не помогут улучшить рабочие условия.

Когда речь идет об управлении инцидентами, есть несколько предупреждающих признаков того, что у вас не будет достаточно возможностей для развития на данной должности, в том числе:

- отсутствие прозрачности в отношении сбоев;
- культура вины и страха, когда люди боятся говорить об ошибках;
- повторяющиеся инциденты, без принятия мер по улучшению ситуации или исправлению в долгосрочной перспективе.

Существуют и другие проблемные области, но эти особенно болезненны, поскольку мешают обучению, подрывают доверие и установление отношений и провоцируют выгорание, что усугубляет последствия инцидентов. Если вы замечаете эти сигналы и не можете добиться более приемлемых условий у себя на работе, ищите новые возможности, пока не наступило выгорание, потому что проходить собеседование при устройстве в новую компанию особенно сложно, когда вы уже измотаны.

Подготовка к инцидентам и план действий

Вы можете надеяться, что в вашу смену ничего страшного не произойдет, но инциденты неизбежно случаются, поэтому нужно иметь план реагирования и систематически подготавливаться к таким ситуациям. При работе с современными системами следует наладить сотрудничество и координацию, как в рамках команды, так и между командами, чтобы донести целостную и достоверную информацию до различных заинтересованных сторон. В некоторых организациях для координации и совместной работы по разрешению инцидентов создается специальная временная команда, например группа реагирования на инциденты (*англ.* incident response team, IRT) или группа управления инцидентами (*англ.* incident management team, IMT).

В следующих подразделах описаны шаги, связанные с планированием и подготовкой.

Настройка и документирование коммуникационных каналов

Во время инцидента команда не должна разбираться, как наладить взаимодействие, особенно когда люди могут находиться в разных местах или даже часовых поясах.

Не существует единственно правильного способа организовать обсуждение при возникновении инцидентов. Один из подходов заключается в создании единого канала связи для обсуждений `#oncall`, посвященного дежурствам. Когда в канале для обсуждений выявляется значительный инцидент, создается новый канал `#incident_NUMBER`, в результате чего основной канал `#oncall` не заполняется информацией, связанной с данным инцидентом, чтобы другие потенциальные проблемы не терялись из виду. Проблема такого подхода заключается в управлении и отслеживании большого числа недолговечных каналов.

Другой подход заключается в создании единого канала `#oncall` и обсуждении инцидентов в тредах (ветвях, соответствующих отдельным темам). Это помогает упорядочить и сделать заметными инциденты, но канал может оказаться перегружен, особенно когда в тредах, связанных с инцидентами, набираются сотни сообщений.

Третий подход — это компромисс: начните с тредов в основном канале `#oncall` и помните о сфере охвата расследования. Разбивайте основной канал на несколько отдельных, когда это становится необходимо.

Выберите стандарт и меняйте подход в зависимости от того, насколько он подходит для вашей команды.

Обучение эффективному общению

Четкое определение критериев, определяющих как должно происходить общение во время инцидента, сокращает количество ошибок и время разрешения. Обратитесь к «Случаю #2: одна и та же история для разной аудитории» *(см. гл. 10)*. Учитывайте уровень детализации, который подходит для разных аудиторий. Внутренние команды, работающие над проблемой, должны обмениваться нефильтрованной

информацией в режиме реального времени. Руководителям, контролирующим ход дела, могут понадобиться только периодические отчеты о текущей ситуации, а клиентов и другие внешние заинтересованные стороны устроит просто краткое резюме.

Люди, ответственные за процедуры общения, стимулируют дискуссии и обмен знаниями во время расследования.

Внедрение управления ресурсами экипажа при обучении сотрудников авиакомпаний

28 декабря 1978 года самолет авиакомпании United Airlines выполнял рейс UA173, когда во время захода на посадку в международном аэропорту Портленда экипаж обнаружил сбой и не получил четкого сигнала об успешном опускании шасси. Экипаж запросил режим ожидания, чтобы подготовить пассажиров к возможной аварийной посадке. Капитан был занят решением проблемы с шасси и не следил за показаниями датчиков топлива. Бортинженеры выразили обеспокоенность по поводу уровня топлива, но не смогли успешно донести свои опасения до капитана. Когда произошел срыв пламени в двигателях из-за нехватки топлива, экипажу пришлось совершать вынужденную посадку, и самолет потерпел крушение.

После расследования, проведенного Национальным советом по безопасности на транспорте (National Transportation Safety Board, NTSB), было установлено, что причиной других авиационных происшествий были аналогичные проблемы, связанные с недостаточным взаимодействием между членами экипажа и невозможностью совместного участия в принятии решений. Некоторые члены экипажа обладали критически важной информацией, но не делились ею и не ставили под сомнение принятые решения. NTSB выявил необходимость в новой методике обучения, которая позволила бы улучшить процесс принятия решений и устранения проблем и делала акцент на эффективном взаимодействии между членами экипажа. Исследовательский центр Эймса НАСА представил оригинальную методику управления ресурсами экипажа (Crew resource management, CRM) на семинаре в 1979 году[1].

Сотрудничать сложно. Вы должны понимать, как это делать, и регулярно практиковать. Правила могут варьироваться в зависимости от ситуации и людей, входящих в команду. Не стоит учиться сотрудничать, когда инцидент уже произойдет.

Кроме того, вам необходимо обрести уверенность, которая позволит вам задавать вопросы лидерам или экспертам в данной области. Все совершают ошибки. Если вы не решаетесь высказаться, когда замечаете ошибки, это может привести к более негативным последствиям.

Создание шаблонов

Шаблоны помогают добиться единообразия в управлении инцидентами и повысить эффективность работы, поскольку люди действуют в соответствии со схемой и планом. Шаблоны устанавливают цели и стандарты.

Работников могут раздражать шаблоны со слишком большим количеством строк или полей. Убедитесь, что шаблоны можно использовать, обладая минимально необходимой информацией.

[1] Джерри Муленбург (Jerry Mulenburg), «Crew Resource Management Improves Decision Making» (**https://oreil.ly/Z5cbE**), APPEL Knowledge Services, последнее изменение от 11 мая 2011 г.

Ведение документации

Необходимо регулярно пересматривать и обновлять документацию, связанную с дежурствами и разрешением инцидентов. Устаревшая документация, не отражающая текущие процессы, мешает организационному обучению и затрудняет работу инженеров. Обязательно изучите процедуры обработки оповещений, восстановления после сбоев и другие процессы, которые на первый взгляд могут не относиться к документации.

Документирование рисков

Каким рискам вы подвержены, какова их вероятность и последствия? Цель управления инцидентами заключается не в том, чтобы не допускать их, а в том, чтобы снизить риски и дать возможность вашей организации продолжать вносить изменения.

Подумайте о возможных сбоях и об их причинах. Это поможет вам подготовить планы резервного копирования данных и выделить факторы, способствующие успешному решению проблемы.

Подробнее о рисках читайте в главе 3 книги Марка Альвидреса (Marc Alvidrez) «Site Reliability Engineering from Google» (**https://oreil.ly/es3Qs**).

Моделирование аварийных ситуаций

Выполните и проверьте имеющиеся процедуры обработки инцидентов, чтобы довести выполнение всех шагов до автоматизма. Подобно различиям между тестированием и реальной эксплуатацией системы, отработка реакции на смоделированную аварийную ситуацию сильно отличается от действий в случае настоящих инцидентов. Но даже при моделировании сбоев вы можете обнаружить и устранить пробелы в документации и процессах, благодаря чему будете гораздо более подготовлены к внештатным ситуациям, когда приходится спешно разбираться с проблемой посреди ночи.

Изучение имеющихся инструментов

У вашей команды будет набор инструментов, методов и процессов для управления инцидентами. В табл. 20.1 приведены некоторые инструменты, на которые следует обратить внимание.

Убедитесь, что у вас есть учетные записи для доступа к каждому из этих инструментов и способы доступа, будь то специальное приложение, которое вы устанавливаете на свой телефон, или URL-адрес.

Таблица 20.1. *Категории инструментов*

Категория	Цель
Мониторинг	Измерение, сбор, хранение, изучение и визуализация данных из инфраструктуры
Оповещение	Управление дежурными сменами и эскалацией вызовов и уведомление назначенных дежурных
Служба чата	Обеспечивает общение в режиме реального времени для обмена наблюдениями, ссылками и скриншотами
Видеочат	Обеспечение связи в режиме реального времени для обсуждения и согласования подходов к реагированию на инциденты
Отслеживание инцидентов	Обработка, устранение неполадок и отслеживание общего хода инцидентов
Документация	Классификация и обобщение сведений (отчеты об управлении инцидентами, исследования инцидентов)
Отслеживание проблем	Обработка, устранение неполадок и отслеживание общего хода решения проблем с вашими системами и программным обеспечением. Это не обязательно будут те же самые инструменты, которые используется для отслеживания инцидентов

Четкое определение должностей и обязанностей

Группы реагирования на инциденты различаются по организациям. Если в вашей организации есть IRT, то должности ее участников могут называться по-другому или имеются еще какие-то отличия. Вот несколько важных ролей (независимо от того, как они называются в вашей организации): руководитель работ по ликвидации последствий аварийного происшествия, специалист в соответствующей области, официальный представитель руководства и лицо, ответственное за ведение записей.

Руководитель работ по ликвидации последствий аварийного происшествия (Incident commander, IC)

Отвечает за урегулирование инцидента. Во время инцидента всегда есть один действующий руководитель, который координирует различные мероприятия. Ответственность может переходить от одного лица к другому на протяжении всего времени урегулирования инцидента.

Специалист в соответствующей области (Subject matter expert, SME)

Дежурный инженер или уполномоченный руководитель определенной части сервиса. Для разрешения определенных инцидентов может потребоваться несколько таких специалистов.

Лицо, ответственное за обмен информацией

Отвечает за обмен информацией о состоянии текущего инцидента внутри организации и за ее пределами. В зависимости от масштаба инцидента таких специалистов может быть несколько.

Лицо, ответственное за ведение записей

Ведет записи, подробно описывая важные действия и последующие события, которые происходят во время инцидента. Этот специалист может использовать программное обеспечение, реагирующее на специальные команды, или чат-бот. Урегулирование инцидентов с помощью инструмента чата, такого как Slack, или записи видеоконференции также возможно, поскольку в обоих случаях можно получить стенограммы того, что обсуждалось. Эти записи имеют решающее значение для предоставления контекста при описании событий. Впоследствии он позволит лучше понять ситуацию и извлечь соответствующие уроки.

Если в вашей организации нет официального принятого порядка реагирования на инциденты, вероятно, следует скорректировать это, чтобы работа дежурной службы и процесс управления инцидентами стали более рациональными. Для этого потребуется заручиться поддержкой руководства.

Представление о степени серьезности и протоколах эскалации

Когда вы получаете оповещение во время дежурства, вам нужен надежный способ определения его приоритета и идентификации проблемы как инцидента. Понимание того, как ваша команда определяет степень серьезности, поможет вам решить, что делать и кого ставить в известность.

Более низкое значение степени серьезности обычно свидетельствует о более серьезном инциденте. Это может выглядеть следующим образом.

Степень серьезности 1

Критический инцидент с высокой степенью влияния. Например, это может быть полностью скомпрометированная система, что затрагивает всех клиентов, нарушение конфиденциальности из-за взлома системы или потеря данных клиента.

Степень серьезности 2

Крупный инцидент со значительной степенью влияния. Например, система, производительность которой уменьшилась, что затрагивает некоторых клиентов.

Степень серьезности 3

Незначительный инцидент. Например, система реагирует медленнее, но не полностью вышла из строя.

Когда у команды имеется единое понимание, что означает степени серьезности, они могут сообщить о серьезности инцидента и оперативно инициировать соответствующие протоколы эскалации, которые обеспечивают нужный уровень реагирования. Чем серьезнее инцидент, тем важнее, чтобы разные люди выполняли различные роли при управлении инцидентом.

Реагирование на инциденты

В каждой команде есть некий процесс (документированный или нет) урегулирования инцидентов. Обдумывание и четкое документирование каждого аспекта этого процесса поможет лучше скоординировать действия, когда придется действительно урегулировать инцидент. Пример на рис. 20.2 показывает процесс управления реагированием на инциденты. Вы видите четко обозначенные роли и обязанности различных частей команды по управлению инцидентами.

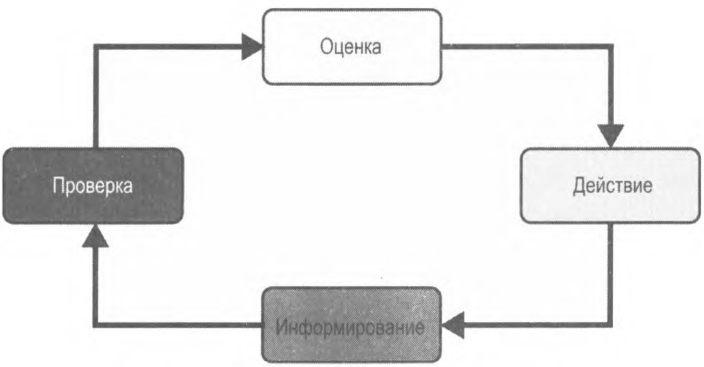

Рис. 20.2. Цикл реагирования на инциденты

IC *оценивает* степень серьезности инцидента по его признакам, масштаб проблемы и потенциальные риски.

Когда наступает время *действовать*, выполняются следующие шаги:

1. IC определяет возможные действия и связанные с ними риски.
2. IC принимает решение и произносит его вслух по телефону или в канале чат-платформы.
3. IC добивается консенсуса относительно данного решения, выясняя, есть ли у кого-то серьезные возражения. IC корректирует действия на основе обратной связи, но в конечном итоге окончательное решение остается за ним.
4. IC делегирует проведение мероприятий по стабилизации ситуации. Задания должны быть ясными и конкретными, в них следует четко указать время, когда человек будет информировать команду о ходе работы.

   Иногда у работника может не хватать навыков для выполнения определенных действий. Это хорошая возможность, чтобы поучиться у более опытных коллег. Если времени недостаточно или задач слишком много, проведение стабилизационных мероприятий следует поручить более опытному специалисту.

   Задания должны корректироваться на основе обратной связи и требуемых сроков. В случае серьезных инцидентов можно привлечь больше людей в группу реагирования на инциденты для своевременного выполнения необходимых задач.

Следующая фаза цикла — *информирование*. В зависимости от численности группы, занимающейся инцидентом, IC может назначить отдельного человека, ответственного за обновление информации. Этот работник не обязан активно исследовать и ремонтировать систему. В противном случае ему придется переключаться с отладки на коммуникацию и выполнять критически важные команды, что усугубляет стресс и увеличивает количество ошибок.

Когда действующий сайт находится в нерабочем состоянии, четкое и своевременное общение с клиентами требует мастерства. Плохо сформулированные объяснения могут вызвать больше проблем, чем само отключение.

Лицо, ответственное за распространение информации, регулярно отправляет обновленные сведения команде, клиентам и руководителям. Частота и содержание сообщений будут варьироваться в зависимости от аудитории. Обновления должны включать информацию о происходящем и предпринятых шагах. Когда узкие специалисты больше не требуются, IC уменьшает масштаб инцидента. Он информирует об этом группу реагирования на инциденты, которая все еще необходима для решения проблемы, и предлагает сделать перерыв тем работникам, в которых уже нет острой необходимости.

Последним шагом в цикле реагирования на инцидент является *проверка*. Требуется подтвердить, что стабилизационные мероприятия завершены, и проверить их результаты. Если инцидент по-прежнему влияет на функционирование системы, руководитель работ по ликвидации последствий аварийного происшествия повторяет указанные шаги, начиная с оценки инцидента.

Извлечение уроков из инцидента

После устранения инцидента сопоставьте информацию от всех, кто участвовал в мероприятиях. Это нужно не для поиска виноватого, а для выяснения обстоятельств происшествия и для того, чтобы побудить людей к разговору. Один из способов предотвратить ложные обвинения — убедиться, что в центре внимания находится само происшествие и соответствующие действия работников, а не то, как должна была бы работать система, или что можно было бы сделать.

Как глубоко следует расследовать инцидент?

Организации различаются размерами и структурой. Они могут постоянно сталкиваться с инцидентами различного рода. Приходилось ли вам когда-нибудь участвовать в совещании по разбору инцидента, где цель, казалось, состоит в том, чтобы просто пройтись по контрольному списку, а не сосредоточиться на последствиях инцидента и мерах, которые принимались для его разрешения? Я присутствовала на многих таких совещаниях, размышляя, как избежать такого положения дел в будущем. Чтобы извлечь уроки из инцидента, будьте готовы к получению и изучению новой информации, а не к сверке с каким-либо перечнем, не допускающим отклонений.

Каждый инцидент неповторим, как снежинка, даже если он выглядит знакомым. Каждая система имеет собственную комбинацию вычислительной инфраструктуры, систем хранения данных или сетей. Может использоваться другое программное обеспечение (либо другая версия или конфигурация программного обеспечения).

Системами управляют разные люди. В зависимости от уровня зрелости вашей организации, набора инструментов, которыми вы располагаете, и количества людей в штате время на анализ всех инцидентов может варьироваться, и его может не хватать. Как же понять, что и насколько подробно нужно исследовать? Фактически это зависит от того, чему ваша команда хочет или должна научиться. В числе инцидентов, представляющих интерес, могут быть те, которые требуют участия множества команд или оказывающие большое влияние; инциденты, связанные с новыми системами или функциями, или события, когда инцидента удалось избежать.

Опасность когнитивных искажений

Ряд когнитивных искажений может помешать вам выявить системные причины инцидентов.

Эффект привязки

Когда вы принимаете в расчет лишь отдельные фрагменты данных или полагаетесь на один источник информации при принятии решения вместо того, чтобы рассматривать все в комплексе. Контрольные списки уменьшают этот эффект, позволяя вам не упускать из виду важные детали[2]. Сверка с контрольным списком не является признаком некомпетентности. Это говорит о том, что даже профессионалы совершают ошибки и стремятся свести их к минимуму.

Эффект доступности

Возникает, когда вы находитесь под влиянием запоминающихся событий или можете легко получить доступ к сведениям о них. Один из способов минимизировать такую предвзятость — вести архив предшествующих инцидентов с возможностью поиска. Тогда вы сможете сравнить текущий инцидент с теми, которые ваша команда успешно разрешила.

Необъективность восприятия

В этом случае вы опираетесь на данные, которые согласуются с ранее существовавшими у вас убеждениями, отвергая всю остальную информацию. Чтобы справиться с этой проблемой, ищите свидетельства, противоречащие вашему мнению, и обращайте внимание на различные точки зрения.

Ретроспективное искажение

Предположение о том, что можно было предсказать наступление определенного события.

Предвзятость в отношении существующего положения вещей

В этом случае вы предпочитаете, чтобы все оставалось по-прежнему, что приводит к сопротивлению переменам.

Чтобы противостоять предвзятому отношению:

1. Сначала соберите факты, не выясняя непосредственную причину.
2. Отметьте возможные причины.

---

[2] В качестве примера отметим, что даже высококвалифицированным хирургам полезны контрольные списки. Контрольные перечни мер (**https://oreil.ly/DSJIq**) по обеспечению хирургической безопасности зарекомендовали себя как эффективный способ улучшения медицинских результатов для пациентов, перенесших операцию.

3. Поищите и оцените противоречивые свидетельства.
4. Пересмотрите данные.

Помощь в обнаружении

Рассуждая в сослагательном наклонении о событиях, вы рискуете так и не получить истинную картину случившегося. Это не значит, что вы не должны признавать ошибки. Их нужно обсуждать и использовать как возможность для обучения. Но если в команде отсутствует психологическая безопасность, признать собственную неправоту или ошибку бывает очень трудно. Если люди не высказывают свое мнение из опасения, что сделали что-то неправильно, вы упускаете возможность решить системные проблемы или выявить ошибочные предположения о рекомендуемых действиях.

В зависимости от своей роли в процессе обнаружения аварий и расследования их причин, особенно если вы не входили в группу реагирования на инциденты, можете задать следующие вопросы.

- Каким образом вы были уведомлены о событии?
- Случались ли подобные инциденты раньше?
- Если инцидент происходил ранее, какое влияние он оказал?
- Что вас удивило в этом инциденте?
- Может ли этот инцидент повториться?

Вы можете обнаружить, что имеются различные точки зрения на систему и сценарии развития событий, а также скрытые различия в том, как люди принимают решения по управлению системами.

Эффективное документирование инцидентов

Отчеты об инцидентах — это ценные артефакты для команды, помогающие распространять знания и предотвращать стагнацию. Артефакты команды должны храниться централизованно. В зависимости от организации они могут быть полезны и другим командам.

Каждый артефакт может иметь несколько иное содержание в зависимости от характера инцидента. Целевая аудитория отчета об инциденте, который готовит команда, — это отдельные члены команды, поэтому такие отчеты могут быть более длинными и включать больше подробностей, чем брифинги для внешней аудитории и руководства. Вот примерный шаблон отчета об инциденте.

- Заголовок.
- Дата.
- Автор(ы).
- Краткое описание инцидента.
- Участники инцидента и их роль(и).

- Влияние.
- Хронология событий.
- Графики и журналы, которые помогают подтвердить факты, приведенные в хронологии событий.
- Извлеченные уроки в отношении того, что прошло хорошо, а что требует улучшения.
- Мероприятия — сюда следует включить ответы на вопросы: кто, что, какого типа мероприятие и когда проводилось. Другие лица, не входящие в группу реагирования на инцидент, после ознакомления с описанием мероприятий могут внести свои предложения о дополнительных действиях.

Все, кто принимал участие в разрешении инцидента, должны просмотреть запись инцидента и добавить отсутствующую информацию, включая те области, где они могли бы испытывать затруднения или не знали бы, как действовать дальше.

Отчеты об инцидентах, подготовленные командой, — не единственные артефакты, представляющие интерес. На основании своего опыта могу сказать, что когда внимание сосредоточено на создании одного артефакта, создается ощущение, что это попытка возложить на кого-либо вину за инцидент. Это порождает страх и мешает сотрудничеству при изучении происшествия. Генерируется множество данных, изучаются разнообразные графики, и многие люди, возможно, принимали участие в том, чтобы вернуть сервисы в работоспособное состояние. Анализируйте всю эту информацию и составляйте необходимые артефакты. Это может быть общий доклад для руководителя организации или переписка с клиентом, которые послужат дополнением к отчету об инциденте, подготовленному командой.

Распространение информации

После документирования инцидента поделитесь с организацией теми знаниями, которые приобрели. Можете отправить сообщение по электронной почте, добавить новость на веб-сайт или представить информацию на собрании. Собрание, посвященное разбору инцидента, является важной частью процесса непрерывного обучения в организации.

Все, кто собирается на эту встречу, должны иметь общие цели, чтобы можно было скоординировать усилия по их достижению. Встреча после разрешения инцидента при отсутствии общих целей часто хуже, чем отсутствие встречи вообще. Когда интересы конкурируют или люди не получают признания за тот полезный вклад, который они делают, это может привести к стремлению демонстрировать героизм или дистанцированию от всего процесса.

Цели не должны отражать идеалистический совершенный мир. Например, невозможно полностью предотвратить появление инцидентов, поэтому бессмысленно ставить такую цель и пытаться достичь ее.

Лучше постарайтесь, чтобы следующий аналогичный инцидент не проходил по тому же сценарию. Другие полезные цели могут включать выявление таких сфер

деятельности, где у людей нет четкого понимания, почему случились те или иные события, или таких участков, где отдельные работники обладали определенной информацией, но она не была известна всей команде. Другими словами, результат этого собрания призван расширить знания и определить области, на которых необходимо сосредоточиться. После того как информация будет распространена среди большей группы людей, вероятно, потребуется обновить некоторую документацию.

Дальнейшие шаги

Достижение успеха в управлении инцидентами часто связывают с улучшением таких метрик, как среднее время работы до отказа (*англ.* mean time between failures, MTBF), среднее время работы на отказ (*англ.* mean time to failure, MTTF), среднее время обнаружения неисправности (*англ.* mean time to detection, MTTD) и среднее время восстановления сервиса (*англ.* mean time to recovery, MTTR). Эти метрики были полезны при изучении характеристик оборудования, т. к. позволяли заблаговременно производить его замену, не допуская перебоев в работе. Их ценность становится существенно ниже, когда речь идет о современных системах, ориентированных на облачные вычисления. Акцент сместился с физических серверов на виртуализированные вычисления, и прогнозирование отказов оборудования стало неактуально. Кроме того, усреднение времени отклика для разных периодов сбоев не дает полезной и действенной информации. Более точные показатели успеха могут быть получены при постоянном совместном изучении отчетов об инцидентах.

Успешный процесс управления инцидентами приносит следующие выгоды:

- меньше людей вовлекается в процесс реагирования на инциденты (работники чувствуют себя более уверенно);
- увеличивается число людей, посещающих разборы инцидентов (они чувствуют, что используют свое время с пользой);
- больше времени отводится на расследование событий.

Заключение

Инциденты — это нештатные ситуации в работе производственной системы, которые оказывают влияние на пользователей этой системы. Нереалистично считать, что можно полностью устранить их. Вместо этого сосредоточьтесь на совершенствовании процесса реагирования на инциденты, осуществляя целенаправленные и взвешенные изменения. Подумайте, насколько хорошо ваша команда реагирует на инциденты и извлекает из них уроки.

Вы со своей командой можете подготовиться к инцидентам, наладив процессы коммуникации, обучения и документирования. Когда происходит инцидент, четко сообщите о нем внутренним и внешним заинтересованным сторонам, клиентам и команде, привлекайте необходимых профильных экспертов и извлекайте уроки для улучшения систем.

Разрешение инцидента должно включать в себя обмен информацией, определение слабых сторон и планирование изменений для снижения рисков в будущем в тех случаях, когда можно обнаружить закономерности в развитии событий, влияющих на ваши системы, или в возникновении инцидентов.

Дополнительные ресурсы

Узнайте больше об управлении инцидентами из следующих материалов:

- Ванесса Хуэрта Гранда (Vanessa Huerta Granda), запись в блоге — «Making Sense Out of Incident Metrics» (**https://oreil.ly/mLpOt**);
- Джон Оллспоу (John Allspaw), запись в блоге — «Moving Past Shallow Incident Data» (**https://oreil.ly/SLnMR**);
- Ричард Кук (Richard Cook), исследование природы сбоев — «How Complex Systems Fail» (**https://oreil.ly/uxHqA**);
- курируемая сообществом коллекция проектной документации, посвященной отказоустойчивости (**https://oreil.ly/xjkf1**).

Руководство устойчивыми командами

Давайте вернемся к некоторым понятиям из предыдущих глав и рассмотрим их с точки зрения людей, занимающих руководящие должности. На протяжении всей своей карьеры я наблюдала, как комплексные системы влияют на гибкость и возможности команды. У руководителя есть дополнительные возможности, чтобы поощрять и поддерживать команду, изменяя работу комплексной системы.

Итак, эта глава предназначена непосредственно для людей, занимающих руководящие должности. Хотя я твердо верю, что каждый может быть руководителем, я знаю, что в различных организациях функциональное руководство запрещается. Если вы сейчас не являетесь руководителем, я призываю вас прочитать эту главу и поделиться информацией со своим руководством, если вам понравятся представленные здесь идеи.

В этой главе я расскажу о том, как руководить командами, применяя подход, ориентированный на команду и направленный на непрерывное обучение на всех этапах жизненного цикла системы, не замыкаясь на какой-либо конкретной области деятельности (например, на разработке, обеспечении качества или эксплуатации).

Коллективное руководство

> Лидер — не тот, кто хочет думать за людей, а тот, кто учит их думать самостоятельно.
> – *Мэри Паркер Фоллетт*

Лидер нужен, когда проблема настолько масштабна, что один человек не может решить ее или как-то повлиять на ситуацию. Лидерство — это не только управление и раздача указаний подчиненным. Это масса усилий, направленных на то, чтобы убедить других, что проблема существует, и, действуя сообща, вы сможете решить ее.

Кроме того, я признаю, что в организации существует формальная власть. Лица, уполномоченные принимать решения, связанные с финансами (например, прием на работу, увольнение, удержание сотрудников, реорганизация), имеют больше возможностей для создания и поддержания устойчивости.

Лидерство связано с большой ответственностью и привилегией. То, что происходит в вашей команде и у вас на работе, влияет на жизнь людей, их здоровье, личные отношения, личные цели и мечты. Вы влияете на то, как они проводят свое время — самый ценный ресурс, который есть у каждого из нас. У вас есть обязательства перед компанией, но не меньшая ответственность и по отношению к людям.

Вместо модели жизненного цикла, в котором разработка и эксплуатация разделены, я представляю на рис. 21.1 модель жизненного цикла devops с непрерывным обучением. Она устраняет это разделение и фокусируется на непрерывном обучении на каждом этапе, а мониторинг дает возможность вести наблюдение и экспериментировать всем сотрудникам, независимо от занимаемой должности или этапа работы.

Рис. 21.1. Современный жизненный цикл devops требует непрерывного обучения

Организации часто основывают решения о финансировании (численность персонала, проекты, инструменты и технологии) на первоначальных требованиях и игнорируют изменения на остальных этапах жизненного цикла системы. Например, при создании программного обеспечения выделение ресурсов сосредоточено на потребностях разработчика программного обеспечения, а не на поддержке системы после ее запуска в эксплуатацию. Кроме того, люди считают само собой разумеющимся, что «кто-то» будет в курсе, если что-то пойдет не так, и немедленно отреагирует.

Понаблюдайте за подобными явлениями в вашей организации, поскольку они свидетельствуют о том, что ответственные за работу системы будут нести основную нагрузку по управлению ей.

Внедрение командного подхода

Командный поход предполагает, что в команду входят все специалисты (например, системные администраторы, разработчики, администраторы безопасности, сетевые администраторы, менеджеры по продукции), причастные к системе, и каждый несет ответственность за ее успешную работу.

Этот подход требует, чтобы каждый человек ценил сильные стороны и навыки других членов команды. Невозможно знать все, и работа в отрыве от других людей ограничивает возможности отдельного человека при решении проблем.

Командный подход позволяет полнее осмыслить сложные среды: понимание всеми членами команды компонентов системы помогает снизить риски, которые могут

негативно повлиять на опыт использования системы или управления ею. Не стремитесь получить «идеальную» систему с нулевым временем простоя, где все риски исключены. Погоня за нулевым временем простоя дорого обходится бизнесу и потенциально вредна для людей, поддерживающих системы.

Сочетание командного подхода и такой корпоративной культуры, где сотрудники не боятся задавать вопросы, побуждает людей обращаться за уточняющими сведениями. В результате улучшается общее понимание, поскольку людям не приходится строить предположения в отрыве от других.

Важное замечание, особенно для руководителей: хотя ретроспективные собрания и корректировка протоколов реагирования на инциденты играют важную роль, не нужно критиковать действия сотрудников, которые занимались разрешением инцидента, когда вы стараетесь оказать им социальную поддержку. Согласно исследованию, основанному на модели рабочих требований/ресурсов (Jobs Demands-Resources model), предоставление свободы действий и социальная поддержка уменьшают эмоциональное истощение и другие последствия большой рабочей нагрузки[1], но даже благонамеренная и содержательная критика усиливает эти негативные факторы, когда человек лишен нормального сна и испытывает раздражение.

Давайте рассмотрим, как внедрить командный подход при организации дежурств, и создать устойчивую команду дежурных специалистов.

Создание устойчивых команд дежурных специалистов

Дежурная команда является устойчивой, если способна справиться со стрессом, вызванным необходимостью решать неизвестные проблемы, возникающие в системе. Это виртуальная команда людей, которые заботятся об определенном сервисе или системе и рассматривают действующую систему с точки зрения совместного владения для распределения ответственности.

Дежурство дает непосредственное понимание ценности используемого программного обеспечения и обязательств перед клиентами. Люди, которые создают продукт, лучше всего представляют, как он должен работать, и понимают соответствующий контекст. Вместе с тем системные администраторы часто изучают, как работает продукт в производственной среде. Когда на оповещения первого уровня реагируют разработчики, это гарантирует, что наиболее ответственными участками системы занимаются люди, создающие продукт.

Кроме того, если они участвуют в дежурствах, то могут более четко определить порядок приоритетности при работе над функциями, будь то рефакторинг существующих систем, внедрение новых функций или удаление критически важного для системы функционала. Наконец, когда разработчики обращаются за инфраструктурной поддержкой, они могут лучше оценить серьезность проблемы, т. к. хорошо представляют тонкости работы системы.

---

[1] А. В. Баккер (A. B. Bakker) и др., «Job Resources Buffer the Impact of Job Demands on Burnout», J. Journal of Occupational Health Psychology 10 (2005): 170–80.

Некоторые организации стали добавлять разработчиков в дежурные смены и начали выступать за ликвидацию эксплуатационных групп. Команда, в которой нет специалистов по эксплуатации (no-ops), может представлять проблему, потому что многие разработчики не обладают навыками в этой области. Отсутствие этих навыков не отражается на отдельных работниках. При разделении труда всегда существуют компромиссы. Разработчики часто фокусируются на программном обеспечении для решения конкретной бизнес-задачи, в то время как сисадмины специализируются на действующих системах. Когда у вас нет людей, которые разбираются в действующих системах, требующих постоянного ухода и обслуживания, это создает проблему. Один из способов определить, имеется ли подобная проблема в вашей организации, — изучить, есть ли трения при описании работы команды.

В то время как разработчики могут отвечать за конкретный компонент, сисадмины обычно обладают информацией о взаимодействии различных систем. Эксплуатационные группы играют довольно важную роль в дежурствах, будь то в рамках первичной ротации или поддержки второго или третьего уровня. Помимо поддержки первого уровня, системные администраторы помогают интегрировать продукт с другими зависимостями или стандартной функциональностью (например, резервное копирование и восстановление).

Сисадмины часто анализируют системные ресурсы, обращаясь к поставщикам облачных услуг и сторонних сервисов. В результате они хорошо представляют, чем отличаются производственные системы от экспериментальных и тестовых экземпляров других инженеров.

Бывает трудно изменить динамику коллектива, где люди считают, что за дежурства отвечает один человек, который должен нести ответственность. Поощрения и вознаграждения должны соответствовать важности среды, влияющей на клиента. Поддерживать модель совместного владения и следить за надежностью и прочностью действующей системы так же важно, как и за работой ее функций.

Ознакомьтесь с данным примером изменения практики дежурств в LinkedIn (**https://oreil.ly/DYZjR**).

Не устанавливайте жесткое расписание дежурных смен. Это может быть план или запись, которые позволяют учитывать меняющиеся обстоятельства. Сформируйте устойчивую сплоченную команду, в которой люди готовы подменять друг друга в сменах или подменять дежурного инженера, когда ему требуется передышка.

Обновление процесса дежурства

Стрессовые ситуации на дежурствах без надлежащего отдыха вредят психическому здоровью и могут вызывать тревогу, депрессию, выгорание и другие проблемы. Кроме того, люди могут размышлять о производстве непродуктивным образом, отказываясь от творческого подхода, так необходимого для работы над сложными проектами и для разработки планов, помогающих эффективно решать коренные проблемы.

В командах, которые применяют непрерывное обучение по отношению к дежурствам, снижается риск появления этих проблем и повышается устойчивость команды при реагировании на сбои в работе. Предположим, что дежурство организовано таким образом, что уровень стресса у инженеров снижается. В этом случае все выигрывают от того, что у них есть возможность обдумывать и генерировать идеи, потому что людям больше не приходится волноваться или опасаться более значительных неблагоприятных воздействий или сбоев.

Обновите процесс дежурства, выполнив следующие действия.

Мониторинг условий труда дежурных

Большое число оповещений и затягивающиеся события вызывают усталость дежурных инженеров. Когда люди устают, то чаще совершают ошибки. Мониторинг условий труда дежурных улучшает общее состояние здоровья отдельных сотрудников и команды в целом, поскольку позволяет уставшим работникам не задерживаться на дежурстве, а также способствует формированию культуры взаимной поддержки.

Чтобы добиться успеха в этом начинании, вы должны заслужить доверие команды. У людей не должно складываться ощущение, что кто-то заглядывает им через плечо лишь для того, чтобы оптимизировать ключевые показатели эффективности. Напротив, нужно, чтобы все члены команды поддерживали эту инициативу, что позволит оценить, изучить и облегчить давление, которое они испытывают на работе для осуществления устойчивых и долгосрочных изменений.

Поддержка идеи командного подхода

Настоятельно рекомендуйте работникам обращаться за помощью, чтобы повысить общую устойчивость команды. Запланированные эскалации уменьшают страх перед неизвестным.

Мониторинг и установка параметров оповещений

Громкие оповещения раздражают и снижают способность к длительной концентрации внимания (усталость от оповещений), увеличивая вероятность ошибок. Если вы используете справочник по SLO, убедитесь, что документация обновлена и отражает изменения в SLI, — в том числе указаны причины, по которым вносились изменения.

Разработка протоколов работы с инцидентами

Не каждое событие требует количественной оценки. Не каждое событие, которому была дана оценка, считается сигналом тревоги и требует отправки оповещений сотрудникам. Не каждое оповещение является инцидентом. Разработка четких, понятных протоколов помогает снизить утомляемость и эффективнее отслеживать тревожные сигналы. Люди будут понимать, как действовать, если вдруг посреди ночи получат критически важное оповещение.

Наблюдение за влиянием графика работы

Составленные графики работы могут казаться сотрудникам несправедливыми или неоптимальными. Заблаговременное планирование и учет индивидуальных

интересов позволяют создавать более продуманные, гибкие графики которые устраивают людей.

В конечном счете компании, которые создают устойчивые системы оперативного реагирования, будут иметь конкурентное преимущество перед теми, кто этого не делает, поскольку системы становятся все более сложными.

Необходимо запланировать снабжение закусками и питанием команд, занятых разрешением инцидентов. Что касается распределенных команд, важно иметь контактную информацию о дежурных инженерах и вариантах доставки или предусмотреть оплату питания.

Люди забывают о еде, когда поглощены работой по ремонту и восстановлению работоспособности системы. Это усугубляет усталость, связанную с тем, что приходится долгое время концентрировать внимание на определенной проблеме. Руководство команды должно следить за состоянием людей, которые обслуживают систему.

Мониторинг работы команды

Как уже упоминалось в главе 14, мониторинг необходим для повышения прозрачности систем и оценки возможных рисков. Это подразумевает также получение большего количества данных о людях и процессах. Когда вы думаете о работе команды, что вы пытаетесь оценить и каковы желаемые результаты? Какие сигналы помогут вам определить цели и достичь их? Как вы определяете сложные вопросы, которые необходимо решить для повышения эффективности работы команды?

Зачем нужен мониторинг команды?

Мониторинг на уровне команды помогает построить более прочные отношения и доверие, поскольку люди в команде получают лучшее представление о том вкладе, который каждый из них вносит в общее дело, и соответствующем контексте. Прозрачность всех работ, их количественная оценка и поддержка правильных действий отучают команду от стремления к героизму.

Предоставление оценок работы на уровне команды помогает перераспределить задачи, улучшить взаимодействие и получить визуальную обратную связь. Сисадмин теперь не остается с проблемами один на один. Можно устанавливать приоритеты в работе сотрудников, чтобы они уделяли первоочередное внимание самым важным и срочным делам, а по мере возможности переключались на другую, менее срочную работу.

Процесс мониторинга является итеративным. Мониторинг предоставляет информацию, которая помогает анализировать происходящее, дает идеи для обучения команды и внесения изменений. Иногда эти изменения касаются вычислительной инфраструктуры, в других случаях они относятся к человеческой деятельности. Люди являются частью систем, которыми вы управляете, начиная с разработки и заканчивая эксплуатацией.

Например, мне приходилось бывать в средах, где средняя нагрузка на эксплуатационную группу была такова, что каждый работал на пределе возможностей. Если

кто-то брал отгул, запланированный или незапланированный, это создавало дополнительную нагрузку на систему, что приводило к увеличению количества ошибок при разрешении инцидентов и вызывало недовольство членов команды. Мониторинг помог нам определить, что нужны дополнительные люди в команду, чтобы справляться с такой рабочей нагрузкой. Это давало нам дополнительные возможности, когда все были на рабочих местах, но уменьшало взаимные трения, когда кому-либо требовалось отлучиться.

Расширение штата сотрудников

Если вы понимаете, что в вашей команде постоянно не хватает сотрудников, какие действия вы будете предпринимать для расширения штата? К этой проблеме можно подходить поразному, особенно если «команда» — это только вы и есть, но вот мой совет.

Во-первых, не начинайте разговор на эту тему с того, что у вас не хватает сотрудников. Кроме того, при расширении команды учитывайте, что существуют ограничения на то, сколько новых сотрудников могут одновременно приступить к работе. Исследуйте тему, составьте план, заручитесь поддержкой и приступайте к достижению цели, как и при реализации любого другого проекта.

Исследования

Определите, кто обладает полномочиями по принятию решений. Не обращайтесь к вышестоящему руководителю через голову своего непосредственного начальника. Найдите критически важные и значимые проекты, в которых участвовала ваша команда и которые имеют ценность для руководства. Здесь ключевым моментом является краткость изложения. Не нужно рассказывать обо всем, что вы сделали.

Планирование

Сформулируйте свое предложение, указав конкретную работу, которую будут выполнять новые сотрудники, и то, какой непосредственный эффект будет достигнут. Говорите на языке бизнеса и тех, кто принимает решение. Расскажите о преимуществах, которые могут быть незаметны на первый взгляд: возможность проходить стажировки, оперативно подменять других членов команды и прочее.

Получение поддержки

Предоставьте возможность руководству поддержать ваши усилия, предложив рассмотреть ваш план, дать обратную связь и одобрить его.

Выполнение

Если вы получили одобрение, составьте заявку на подбор персонала на основе вашего плана. Затем выполните свои обязательства, чтобы все, обещанное вами, свершилось, и люди захотели большего.

Если вы не получили одобрение, выслушайте причины отказа и действуйте соответствующим образом. Возможно, команда выходит за рамки своих обязанностей и должна перестать выполнять некоторую часть работы. Иногда при этом часть системы начнет давать сбои. Как было сказано в предыдущей главе, не нужно пытаться создать идеальную систему, сбои могут быть приемлемы для бизнеса.

За какими параметрами следует наблюдать?

Чтобы определить, за какими событиями наблюдать, поговорите о том, как вы выполняете работу, о тех сопутствующих делах, которые ее сопровождают. Эта дискуссия не должна быть однократным мероприятием. Со временем процессы

команды изменяются, и необходимо регулярно проводить ретроспективные собрания для документирования новых подходов к работе.

Ретроспективное собрание проводится при завершении определенного периода, чтобы проанализировать работу команды и определить, что можно улучшить[2]. Когда проделанную работу видно, люди могут изучать имеющиеся данные и принимать решения на основе этих данных, а не тех событий, которые остались у них в памяти.

Обратитесь к «Случаю #1: одна диаграмма вместо тысячи слов» *(см. гл. 10)*, где я рассказывала о мониторинге работы. В дополнение к расстановке приоритетов в работе команды и повышению видимости для заинтересованных сторон мы стали лучше понимать, что происходит. Благодаря регулярным ретроспективным собраниям мы выявили и устранили основную проблему — слишком большой объем незавершенной работы.

Когда мы внедрили общую доску для отслеживания нашей работы и стали проводить регулярные ретроспективные собрания, мы стали ставить менее масштабные цели, которых можно достичь за квартал. Это уменьшило объем незавершенной работы. В результате мы постепенно внесли несколько изменений в наши процессы: мы приостанавливали большие проекты, чтобы у каждого из нас был только один большой текущий проект, и стали проводить ретроспективные собрания не один раз в квартал, а каждые две недели.

Несмотря на то что у нас было больше встреч, эти изменения помогли нам выполнить больше работы за квартал, потому что каждый из нас, отдельных работников, концентрировал свои усилия.

Когда на ретроспективных собраниях вы размышляете о том, что было сделано правильно, включите информацию о том, кто и какие факторы способствовали успешным результатам. Признание вклада — это способ поддержать доверие в коллективе, даже если предполагается, что люди «просто делают свою работу». Когда вы размышляете о том, что пошло не так, рассмотрите подробно, что помешало достижению целей.

Со временем благодаря качественным ретроспективным собраниям заурядные команды превратятся в великолепные, потому что будут непрерывно совершенствоваться. Без таких собраний работа может казаться хаотичной и непродуктивной, жизненно важные задачи и проекты откладываются, а ненужные выполняются.

Первоначальные темы для обсуждения в команде включают цели команды, определение задач и проектов, описание работы и этапов ее выполнения.

Помните, что в общем и целом (независимо от типа работы) людям необходимы пять вещей[3]:

---

[2] Узнайте больше о ретроспективных собраниях в книге «Agile ретроспектива: как превратить хорошую команду в великую» («Agile Retrospectives: Making Good Teams Great»), авторы Эстер Дерби (Esther Derby) и Диана Ларсен (Diana Larsen), Pragmatic Bookshelf.

[3] Шейла Хендерсон (Sheila Henderson), «Follow Your Bliss: A Process for Career Happiness», Journal of Counseling and Development 78 (2000): 305–3, **https://doi.org/10.1002/j.1556-6676.2000.tb01912.x (https://doi.org/10.1002/j.1556-6676.2000.tb01912.x)**.

Свобода

Чувство контроля и самостоятельности в работе.

Стимул

Ощущение, что их разум стимулируется. Они размышляют над тем, как решить проблему, и полностью используют свой потенциал, чтобы исследовать, придумывать или внедрять новые подходы.

Личностно значимый вклад

Ощущение, что их работа имеет значение.

Позитивная атмосфера

Восприятие связи с коллегами по работе в положительном ключе.

Образование

Повышение компетентности и квалификации.

Не забывайте об этих вещах, когда даете задания сотрудникам. Например, вместо того, чтобы говорить кому-то: «Сделайте так-то и так-то», спросите: «Как думаете, что нам нужно сделать, чтобы решить эту проблему?»

Каковы задачи команды?

Задачи — это всеобъемлющие цели команды, которые помогают согласовать поступающую работу и производительность команды. Например, предположим, что команда завершает поступившую работу, которая не отвечает задачам команды. В этом случае необходимо, чтобы руководство либо обновило задачи для команды, — чтобы люди, выполняющие эту работу, получали благодарность за свой вклад в решение этих задач, — либо помогло команде отказаться от их выполнения. Если руководитель команды, являющейся частью инфраструктуры, не включает работу сисадмина в общие задачи команды, эта работа может не оцениваться по достоинству и не вознаграждаться!

Отдельные люди не определяют задачи команды. Когда это происходит, человеку, который формулирует задачи, часто приходится добиваться их выполнения. В идеале люди выполняют работу, потому что они мотивированы и заинтересованы в выполнении задач, а не потому, что их заставляют делать ее.

Как команда определяет для себя задачу?

Задача должна быть конкретной, чтобы можно было отследить этапы ее выполнения и завершить в определенные временные рамки. Продемонстрируйте на примерах реальных задач, где могут возникать различия в определениях. Конкретный временно́й интервал может составлять час, день или неделю. Например, настройка нового сервиса в одних командах может считаться задачей, а в других — проектом.

Будущие обсуждения могут также включать характеристики задач — являются ли они более-менее привычными (и могут быть задокументированы с помощью инструкции или контрольного списка) или новыми, требующими дополнительного времени на планирование?

Как команда определяет для себя проект?

Я обнаружила, что довольно трудно выделить больше недели времени для работы над чем-то одним. Чтобы решить масштабную задачу, я разбиваю весь процесс на этапы. Поэтому всю работу, на выполнение которой мне нужно больше недели, я обычно считаю проектом. Это означает, что я должна предпринять некоторые дополнительные шаги для его выполнения.

Проект состоит из ряда задач. Чтобы сделать работу видимой, люди должны разделить проект на отдельные задачи и разбить большие задачи на более мелкие.

Чтобы прийти к общему пониманию, может потребоваться дополнительная терминология, которую вы определяете вместе с командой. Например, я использую слово *программа* для описания работы, которая представляет собой проект с привлечением участников из внешних команд. Их могут интересовать разные конечные результаты, но коллективные усилия служат достижению общей цели.

Что представляет собой каталог услуг, которые предлагает ваша команда?

Каталог услуг — это структурированный перечень услуг, предлагаемых вашей командой. Обсудите все виды работ и уровни квалификации сотрудников, чтобы вы могли сформировать (и курировать) это предложение. Первый шаг — просто составить полный список работ, а затем устранить избыточные описания. Когда достигнута психологическая безопасность в команде, полезно задать вопрос: «Для чего мы делаем эту работу?» Команда может обнаружить, что есть работы, которые следует исключить из числа приоритетных, чтобы добиться реального конкурентного преимущества.

Изучите работу

Когда у вас будет список работ, изучите их более пристально, обращая внимание на отдельные этапы или стадии. Одинаковы ли они или значительно различаются? Вот пример того, как может выглядеть такое распределение по категориям в зависимости от типа работы:

- исправление ошибок;
- реагирование на инциденты;
- административные задачи;
- запросы, управляемые прерываниями;
- регулярные встречи, в том числе один на один;
- разговоры в курилке;
- специфическая проектная работа.

Некоторые из этих видов работ (такие как разговоры в курилке и один на один) не поддаются автоматическому мониторингу или качественной визуализации. Работы, выполняемые схожим образом, можно сгруппировать и использовать одну и ту же

визуализацию при мониторинге. Если это сделать не получается, не стоит пытаться объединить их в один рабочий поток. Каждый этап должен иметь четкие границы с определенными условиями выхода и входа.

В процессе размышления вы можете обнаружить различия во взглядах на то, что означают эти этапы. Например, что означает этап «Принято»? Приняла ли команда обязательство выполнить работу? Означает ли это, что кто-то ответствен за выполнение этой работы на этой неделе? Дополнительные определения помогают прояснить намерения и установить общее понимание.

Возьмите за правило на каждом ретроспективном собрании обновлять документацию по приему новых сотрудников, включая систематизацию понятий, информацию о процессах и политиках.

Измерение влияния на команду

Как было сказано в *главе 20*, такие метрики, как MTBF, MTTF, MTTD и MTTR, не подходят для современного системного администрирования. Они способствуют внедрению неверных изменений (например, выполнению не той работы и выбору неподходящего времени) или обесценивают работу человека и демотивируют его, заставляя предоставлять метрики, которые не имеют смысла (вспомните о том, что людям важна самостоятельность при работе и они должны ощущать ее значимость).

Вместо сбора этих метрик определите, основываясь на бизнес-целях, как эти цели достигаются (или не достигаются), и внедрите информационные панели, которые помогут достичь желаемого с помощью непрерывного совместного обучения. Если в настоящее время мониторинг отсутствует, используйте те данные, которые доступны. Однако ключевым моментом здесь, и это стоит повторить, является способность оценивать ситуацию.

Когда вы будете анализировать систему, то иногда будете замечать закономерности появления проблем в ней. Например, на прошлой работе тестирование могло спровоцировать сбой сетевых устройств. Ожидалось, что в выходные я буду готова в любой момент приехать на работу и перезагрузить сбойное сетевое устройство, чтобы автоматизированное тестирование продолжалось. В то время возможность инвестиций в более современное сетевое оборудование (десятки тысяч долларов) или коммутируемый блок распределения питания (около тысячи долларов на устройство) никогда не рассматривалась руководством, потому что мое время не ценили и не оплачивали, даже несмотря на то, что мне приходилось дежурить и выходить на работу в выходные дни.

Навязывание людям ложных понятий об ограниченных возможностях приводит к тому, что работа кажется им бесперспективной. Вот пара примеров:

- не приобретаются подходящие инструменты;
- не инвестируются средства или не набирается достаточный штат сотрудников, чтобы поддерживать инфраструктуру, необходимую для предоставления услуги.

Ограниченные возможности создают дополнительную нагрузку на людей, которая часто ошибочно рассматривается как бесплатная, но имеет высокие издержки, такие как время в пути, рабочее время и поиски решения для эффективной работы.

Иногда вы (или ваше руководство) можете принять намеренное решение не инвестировать в автоматизацию или инструменты, потому что сложность системы вызывает высокий риск неудачи при внедрении. Если сложность препятствует внедрению автоматизации, возможно, есть области, которые можно оптимизировать для облегчения поддержки в ручном режиме.

Для проактивного управления мощностями необходимо понимать, сколько труда требуется для поддержки текущих систем. Я имею в виду не прогнозирование конкретных трудозатрат (навыки у команд разные, чрезвычайные ситуации и отпуска вносят свои коррективы), а понимание того, сколько времени команда тратит на каждую систему.

Каким образом можно измерить такое влияние? Вы сможете лучше оценить время и трудозатраты, если члены команды будут регулярно сообщать о своей работе (что требует психологической безопасности в коллективе) и делиться тем, сколько времени необходимо для поддержки конкретных проектов. Конечно, все будут затрачивать разное время на различные виды работ, но если члены команды предоставят эти данные, вы сможете создать общую визуализацию работы, чтобы выявить некоторые ее особенности. Анализ этих данных может выявить области для улучшения.

Вы можете использовать регулярные опросы для определения других метрик, например таких:

- удовлетворены ли сотрудники компенсацией;
- насколько сложной была конкретная задача или проект;
- как влияет работа на деятельность в нерабочее время;
- как люди могли отдыхать в нерабочее время.

Поощряйте swarming — совместное решение (*англ.* swarm — роиться, копошиться) больших проблем, которые можно разбить на отдельные части. При этом для решения задачи или проблемы привлекается несколько человек из команды. Человек чувствует поддержку, а команда получает более глубокое понимание работы.

Поддержка инфраструктуры команды с помощью документации

> Очень редко бывает так, что человек предпочитает просто не делать работу и перекладывает ее на других. Это происходит, когда что-то в системе, которую мы создали в команде или компании, не оставляет ему другого выбора. Чтобы исправить такое положение дел, нужно не рассказывать о том, что следует лучше выполнять свою работу, а изменить саму систему.
>
> – *Кэролин Ван Слик (Carolyn Van Slyck)* (@carolynvs), 15 октября 2021 г.

Команда постоянно развивается. Когда приходит новый работник или кто-то уходит из коллектива, документация формирует те шаблоны, которые позволяют понять, как работает команда в настоящее время и в процессе изменений. В допол-

нение к документированию определений, выявленных в ходе обсуждения метаработы, имеются и другие области для поддержки команды:

Включите проектную документацию в определение того, что сделано

Это могут быть все связанные тикеты, соответствующее взаимодействие через Chat-Ops, а также релизы или развертывания.

Относитесь к документации как к своему программному обеспечению

Присваивайте номер версии, тестируйте и выпускайте.

Четко прописывайте правила в документации

Иногда бывает трудно понять, что именно нужно документировать и сколько времени потратить на составление документации, особенно когда неясно, для какой аудитории она делается, и вы раздумываете, сидя перед пустым экраном. Руководители могут помочь определить приоритет этой работы, снабдить соответствующими инструментами и устранить все возможные препятствия. Лидеры могут подсказать, как действовать, поделившись своей собственной документацией, создав шаблоны и упростив процесс для других.

Если нормы задокументированы, это еще не значит, что кто-то будет следовать им. Если человек не соблюдает стандарты, не думайте сразу, что он виноват. Важно найти время и разобраться, почему это происходит.

Четко документируйте политики. Не предполагайте, что у всех одинаковые ожидания или понимание. Например, как упоминалось в *главе 19*, следует четко документировать политику дежурств.

Поддерживайте культуру обучения

Как было отмечено во введении и показано на протяжении всей книги, количество инструментов, технологий и сторонних сервисов множится. Однако если вы сосредоточитесь на том, как обстоят дела в настоящее время и не будете использовать новую информацию, данные, инструменты и процессы, вы можете застрять в прошлом.

Финансирование культуры обучения требует изменения процесса, выделения времени на обучение и адаптацию к изменениям, а также денег. Чтобы оценить имеющиеся возможности поддержки культуры обучения и запланировать соответствующие мероприятия, обратите внимание на следующие четыре пункта и регулярно пересматривайте свои оценки.

Установите бюджет на обучение

Помимо установления бюджета на обучение, предусмотрите время для того, чтобы люди могли использовать выделенные средства.

Поощряйте обмен знаниями

Делитесь знаниями активно, проводя учебные занятия (например, короткие презентации, встречи для обсуждения прочитанного), или пассивно, составляя от-

четы (например, «Чему я научился на этой конференции или узнал из беседы, исследования»).

Моделируйте аварийные ситуации

Инциденты будут происходить. Команды должны отработать процесс их урегулирования и протестировать вероятные сценарии развития событий. Как и тестирование при разработке программ, моделирование аварийных ситуаций сильно отличается от разрешения настоящего инцидента, но тем не менее оно имеет большое значение. Это помогает выявить пробелы в документации и лучше понять процессы, не дожидаясь аврала в ночное время или обнаружения недостаточной компетенции в тех или иных вопросах.

Выделяйте время для анализа работы

Дайте людям время для совершенствования системы. Для работы, связанной с дежурствами, это выявление и устранение слабых мест в системе или документирование неоправданных ожиданий. Для трудоемкой работы это может быть выявление повторяющихся задач и выделение времени для их автоматизации.

Адаптация к вызовам

Менеджмент старой школы рассматривал людей как ресурсы и управлял этими ресурсами как стандартными винтиками. Современный менеджмент использует неиерархическую модель для представления современных организаций, больше похожую на матрицу[4]. Люди поддерживают отношения друг с другом, обладают эмоциями и не являются взаимозаменяемыми.

Старая модель системного администрирования отделяет людей от работы. Такой подход не срабатывает, потому что создает две параллельные системы — людей и собственно управляемую систему.

Современная модель системного администрирования признает, что люди и работа неразделимы. Люди являются частью системы. У каждого человека есть свой набор навыков, талантов, интересов, мотиваций, стилей работы, триггеров, взглядов на мир и подходов к работе. Воспринимайте системы в целом в более человеческом смысле, чтобы сделать их устойчивыми по отношению к людям, являющимся их частью. У людей ограниченные возможности, и они склонны к ошибкам. И я не могу относиться к людям как к машинам лишь потому, что они являются частью системы. Когда система заставляет людей действовать во вред своему здоровью, например нести дежурства в режиме 24/7, а затем не признает своего влияния, это негуманно.

Вы не можете контролировать все аспекты систем, и люди являются частью этой нестабильности, что усложняет их жизнь. Поэтому будьте искренни, цените людей, которые работают с вами, их эмоции и сотрудничайте с ними.

---

[4] «DevOps Culture: Westrum Organizational Culture» (**https://oreil.ly/Ar8OL**), Google Cloud, по состоянию на 13 июня 2022 г.

В Yahoo я никогда не воспринимала себя как часть системы, пока не ушла в свой первый отпуск, — и все пошло наперекосяк, хотя я оставила несколько контрольных списков и инструкций. В реалиях отрасли мы говорим о единичных точках отказа при проектировании систем. Но, оценивая систему, мы не обращаем внимания на людей, обладающих знаниями, пониманием, контекстом или занятых ее поддержкой. Все это хрупкие элементы системы. Вы не можете автоматизировать систему так, чтобы исключить эту хрупкость. Создание самовосстанавливающейся системы только усложняет ее понимание для человека, которому в конечном итоге придется ее отлаживать или поддерживать.

Эти системы строятся по нашему образу и подобию, моделируя наши отношения. Обратитесь к человеческим отношениям, чтобы понять, как мы относимся к системе. Это позволит создавать устойчивые системы, которые поддерживают людей, являющихся их частью. В качестве примера можно привести взаимную поддержку, возможность признаться, что ты чего-то не знаешь, и понимание того, что другие поддержат тебя.

> «Дежурство не должно быть обязанностью лишь самого работника. Команда, окружение, ситуация и система — все должны поддерживать человека».
>
> *– Райан Китченс (Ryan Kitchens) (@this_hits_home), 15 октября 2021 г.*

Все то, что поддерживает людей — общение, возможность учиться и чувство, что их ценят (вспомните эти три из пяти общих потребностей, которые есть у каждого человека, приведенных выше в *разделе «За какими параметрами следует наблюдать?»*), — должно быть частью оценки системы. Без такой оценки вы подвергаете людей риску выгорания. И это может создать дополнительное напряжение между командами.

И наша работа, и нагрузки на систему становятся все более сложными. Вы должны планировать непредвиденные события, которые находятся вне вашего контроля, потому что они происходят постоянно (например, наплыв запросов и отключение питания). Вспомните об эпидемии COVID-19 в 2020 году и посмотрите, что она сделала с системой: снижение производительности команды (формулы для расчета производительности отдельного работника оставались прежними) наряду с возросшими требованиями увеличило скорость выгорания.

Возможности человека меняются со временем и непредсказуемы. Чтобы добиться успеха, ваша команда должна адаптироваться к сложному ряду беспорядочных и хаотичных вызовов, встающих перед системой. Люди должны чувствовать, что им доверяют, что они наделены нужными полномочиями, и укреплять доверие друг к другу, чтобы в случае необходимости получить поддержку. Менеджеры должны распознавать проблемы и привлекать людей для помощи команде или сокращать объем работы и облегчать нагрузку, исходя из реальных возможностей.

Заключение

Применяйте командный подход, основанный на непрерывном обучении, на каждом этапе жизненного цикла системы, придерживаясь следующих моделей поведения:

- проводите много встреч один на один;
- управляйте объемом работы и межорганизационными связями;
- создайте комфортное пространство для взаимодействия членов команды друг с другом;
- поощряйте совместную работу и ответственность друг перед другом;
- минимизируйте сопутствующие расходы, которые не способствуют устранению или уменьшению технического долга;
- поощряйте командный подход при решении общих проблем;
- обеспечьте четкую и регулярную обратную связь для понимания приоритетов и ограничений команды;
- предоставьте людям возможность делиться своими идеями;
- приветствуйте создание открытой и прозрачной среды.

Заключение

Вы дошли до конца книги и, надеюсь, чувствуете себя лучше подготовленными к многообразию открывающихся перед вами дорог. Теперь вы сможете выбрать правильный курс, чтобы уверенно перемещаться через хаос ваших систем и внедрять современные технологии, инструменты и методы системного администрирования.

На протяжении всей этой книги я показываю путь, который поможет вам понять существующие системы и методы, проектировать системы, используя эти методы и код инфраструктуры, а также управлять системами и масштабировать их. Возможно, вы следовали по этому маршруту, от одной главы к другой, или же «срезали дорогу», переходя к самым насущным проблемам, с которыми вы сталкиваетесь в данный момент. Независимо от того, на каком этапе пути вы находитесь — уже опытный системный администратор или только начинаете карьеру инженера и знакомитесь с эксплуатацией, — этот ресурс даст вам возможность разобраться в своих системах и понять, как решать следующую задачу, шаг за шагом.

Вспомните рис. B.2 из введения и сравнение системного администрирования с походом (здесь это рис. C.1).

В походе, достигнув вершины, нужно снова спуститься, чтобы подняться на следующую гору. В системном администрировании то же самое: вы должны покинуть систему, прежде чем приступить к решению следующей задачи. Уход может означать миграцию существующих клиентов на новую систему, постепенный вывод системы из эксплуатации, передачу ее на обслуживание новым инженерам или ваш уход из компании и работу на новом поприще. Как и в любом путешествии, у вас есть возможность расти, учиться и адаптировать свой подход по мере того, как вы осваиваете новые горизонты.

Я поделилась с вами некоторыми своими историями и проблемами и призываю вас тоже поделиться своими историями и опытом — участвуйте в конференциях (неофициальные встречи, Birds of a Feather или презентации), заведите блог (например, на **dev.to**, **sysadvent** или собственной платформе) или доносите информацию в более сжатой форме (например, сообщения в LinkedIn).

Не стесняйтесь поделиться со мной на LinkedIn **(https://www.linkedin.com/in/sigje)** или dev.to **(https://dev.to/sigje)**.

– Дженнифер (Jennifer)

Рис. С.1. Будущее выглядит светлым, но к нему не ведет четкий маршрут. И все же с помощью знаний, опыта, «мышления роста» и поддержки коллег вы можете уверенно двигаться вперед, зная, что справитесь со всем, что вас ждет

Протоколы на практике

Основываясь на *главе 1*, давайте рассмотрим примеры веб-протоколов, использующихся на практике: HTTP, QUIC, and DNS. Понимание протоколов поможет объяснить сущность взаимодействия систем. И эти протоколы развиваются по мере того, как развиваются потребности Интернета.

Протокол HTTP

HTTP (Hypertext Transfer Protocol, протокол передачи гипертекста) охватывает набор веб-стандартов, описывающих, как системы обмениваются данными в Интернете. Различные типы HTTP-серверов оптимизированы для определенных вариантов использования. Это могут быть серверы приложений, выполняющие код веб-приложений, например Apache Tomcat (**https://tomcat.apache.org/**), серверы кеширования, например Squid (**http://www.squid-cache.org**), или веб-серверы, такие как Apache HTTP Server Project (**https://httpd.apache.org**). Современный веб-стек может включать в себя несколько HTTP-серверов для предоставления сервисов.

Изначально HTTP был разработан как протокол клиент/сервер, с одним запросом / одним ответом. С помощью таких утилит, как Wireshark и tcpdump, можно было просматривать перехваченный трафик и восстанавливать веб-беседы, поскольку коммуникации осуществлялись посредством обычного текста.

Со временем HTTP эволюционировал. Одна из редакций включает использование HTTP-заголовков (**https://oreil.ly/z87B9**) для отправки дополнительной информации с HTTP-запросом или ответом. Пользовательские собственные заголовки исторически использовались с префиксом X-, но это соглашение было объявлено устаревшим (**https://oreil.ly/baSAc**). HTTPS, помимо HTTP/1 и HTTP/2, использует возможности протоколов TLS, TCP и IP (см. рис. А.1).

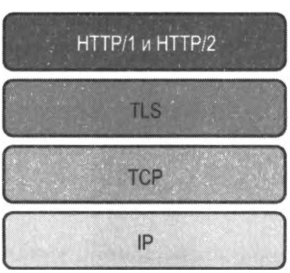

Рис. А.1. Стек протокола HTTPS, использующий модель TCP/IP

Прежде чем две системы смогут взаимодействовать по протоколу HTTP/1, они должны установить TCP-подключение. HTTP/1.0 открывает отдельное TCP-подключение для каждого HTTP-запроса/ответа. Установление ряда отдельных TCP-сеансов для каждого HTTP-запроса приводит к значительному объему служебных данных.

В HTTP/1.1 появился заголовок Connection, чтобы последующие запросы к одному и тому же серверу могли повторно использовать имеющееся TCP-подключение. При этом сокращается общая задержка запроса, особенно если клиент не является локальным по отношению к серверу, поскольку меньше служебных данных требуется на установление трехстороннего рукопожатия при установлении TCP-подключения.

Благодаря новому уровню двоичного фрейма HTTP/2 оптимизирует транспортировку HTTP-запросов и уменьшает задержку при загрузке веб-страниц:

- мультиплексирование запросов и ответов через одно TCP-подключение;
- сжатие данных в заголовках HTTP;
- приоритизация запросов;
- отправка данных по инициативе сервера.

HTTP/2 сохраняет основные концепции HTTP/1.0 и HTTP/1.1, включая методы, коды состояния, поля заголовков и URI. Поэтому вам не придется менять существующие веб-приложения, чтобы воспользоваться преимуществами повышения производительности. Вместо этого HTTP/2 изменяет способ форматирования и передачи данных между клиентом и сервером.

Мультиплексирование соединений позволяет обрабатывать несколько потоков данных независимо друг от друга, и потеря пакетов в одном потоке не повлияет на другие потоки. Но при работе HTTP/2 over TCP все еще имеются проблемы, связанные с блокировкой головных узлов, когда поток данных прерывается, если какой-либо из пакетов задерживается или отбрасывается.

Стандарт HTTP/3 (**https://oreil.ly/raR0k**) переходит от отдельных сеансов TCP к соединениям QUIC, о чем пойдет речь в следующем разделе.

Хотя HTTP на протяжении многих лет определял развитие Интернета, его архитектура была основана на доверии между равноправными узлами. Разработчики не ставили целью обеспечить сохранность или конфиденциальность данных, потому что главная задача была — открыть доступ к исследованиям и общению, а не защищаться от злоумышленников, изменяющих или считывающих трафик. В настоящее время большинство браузеров по умолчанию требуют HTTP/2 over TLS и выдают предупреждение при попытке доступа к старому, незащищенному веб-серверу.

Вы можете настроить 301 редирект с **http://** на **https://**, но злоумышленник все равно сможет перехватывать файлы cookie или идентификаторы сеанса либо использо-

вать принудительное перенаправление на фишинговый сайт. Заголовок HTTP Strict Transport Security (HSTS) сообщает браузеру, что он никогда не должен использовать HTTP и должен автоматически преобразовывать запросы в HTTPS *(см. пример А.1)*.

Пример А.1. Основной синтаксис для заголовка HSTS

```
Strict-Transport-Security: max-age=<EXPIRY TIME>
```

Процесс выглядит следующим образом:

1. Браузер впервые получает доступ к вашему сайту с использованием HTTPS.
2. Ваш сайт возвращает заголовок HSTS.
3. Браузер сохраняет эту информацию, и все последующие подключения будут использовать HTTPS (в течение *max-age* секунд, пока не истек срок действия).

Пока пользователь посещает сайт в течение *max-age* секунд, указанных в заголовке, браузер будет автоматически устанавливать защищенное подключение, даже не пробуя использовать HTTP. Добавление в заголовок флажка `includeSubDomains` позволит применять эту политику и ко всем поддоменам.

Будьте осторожны при использовании флажка includeSubDomain. Если ваша организация использует устаревшие сервисы, которые нельзя обновить, чтобы они поддерживали стандарт SSL/TLS, это может вызвать трудноустранимые проблемы.

Если HSTS-совместимый браузер не может убедиться в безопасности сертификата, он прерывает соединение с HSTS-совместимым сервером.

Узнать больше о системе HSTS можно на следующих ресурсах:

- OWASP HTTP Strict Transport Security Cheat Sheet (**https://oreil.ly/09Wim**);
- «HTTP Strict Transport Security (HSTS)», RFC 6797 (**https://oreil.ly/eLrbS**).

Клиент проверяет подлинность сервера с помощью сертификата сервера. Во время начального рукопожатия TLS клиент и сервер обмениваются секретом, который используется для шифрования сеанса. Этот секрет необходим для расшифровки и анализа захваченных пакетов HTTP/2. Без него вы сможете захватывать зашифрованные пакеты, но у вас не будет возможности расшифровать и понять их. Установите переменную среды `SSLKEYLOGFIL` перед началом захвата, чтобы захватить сеансовые ключи. Браузеры будут добавлять ключи к файлу, который вы определите.

Если вы хотите отладить сессию, установите на клиенте переменную среды `SSLKEYLOGFILE` (**https://oreil.ly/9Yili**) перед выполнением захвата, чтобы захватить сеансовые ключи Ваш браузер будет добавлять ключи к файлу, который вы определите.

Чтобы узнать больше о протоколе TLS, смотрите документ «The Transport Layer Security (TLS) Protocol Version 1.32», RFC 8446 (**https://oreil.ly/lRdTS**).

QUIC

Первоначально представленный компанией Google в 2012 году, протокол QUIC (*англ.* Quick UDP Internet Connections, произносится «quick») устраняет недостатки при использовании прикладного уровня HTTP и транспортного уровня TCP, он стал предлагаемым стандартом (**https://oreil.ly/6Mg5J**) в мае 2021 года.

Появившийся стандарт HTTP/3, официально ставший предлагаемым стандартом (**https://oreil.ly/uSfvB**) в июне 2022 года, включает традиционную семантику HTTP (методы запроса, коды состояния и поля сообщений), которая использует протокол QUIC. По состоянию на ноябрь 2022 года три четверти веб-браузеров (**https://oreil.ly/n3qVI**) и четверть веб-сайтов (**https://oreil.ly/bViGD**) уже поддерживают HTTP/3, и эти показатели, вероятно, будут быстро расти по мере того, как протокол HTTP/3 начнет проходить процесс утверждения в IETF, чтобы стать официальным стандартом, что уже сделано для QUIC.

Как видно из рис. А.2, HTTP/2 имеет четкие отдельные уровни, но в срезе HTTP/3 имеется перекрытие, поскольку протокол QUIC является частью и прикладного, и транспортного уровня.

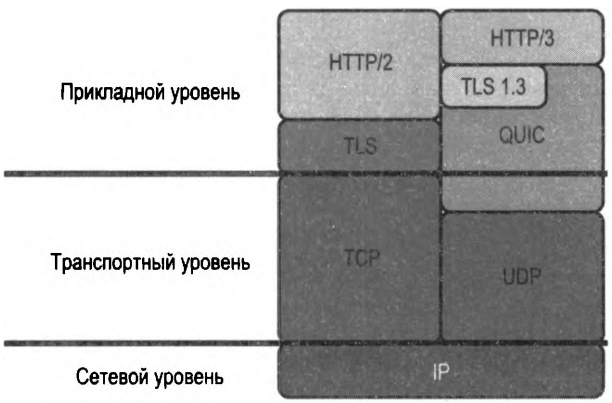

Рис. А.2. Сравнение HTTP/2 с HTTP/3 — похоже на многослойный торт. Вы можете прорезать каждый слой насквозь

QUIC преодолевает ограничения HTTP/2, переключаясь с TCP на UDP для передачи трафика HTTP/3 и более поздних версий. Использование UDP исключает необходимость передачи служебных данных для управления сеансами TCP, и хотя это повышает риск появления ошибок, на практике, в относительно стабильной сети, уровень потери пакетов низок. Даже если периодически возникает необходимость повторной передачи данных, это занимает меньше времени, чем управление сеансами TCP.

Другие преимущества, которые дает использование протокола QUIC:

- прямая коррекция ошибок реализована на прикладном уровне, поскольку теперь, когда управление сеансами TCP отсутствует, исправление ошибок на нижних уровнях не производится;

- шифрование TLS является обязательным, благодаря чему обеспечивается эффективный подход к созданию защищенных сеансовых ключей в рамках первоначального рукопожатия. Сеансовые ключи затем используются для отдельных циклов запросов/ответов, и каждый пакет отдельно шифруется с помощью ключа, в отличие от TCP, который обычно должен ждать, чтобы зашифровать весь поток байтов. Это делает QUIC более безопасным независимо от сервиса;
- улучшена поддержка таких событий, как переключение с одной сети на другую с миграцией соединения, — например, когда мобильное устройство переключается с Wi-Fi на сотовое соединение или обратно, что приводит к изменению IP-адреса устройства. При использовании TCP это приведет к тому, что все активные соединения разрываются по тайм-ауту, а затем устанавливаются заново, по новому каналу. QUIC избавлен от этой проблемы за счет того, что каждому соединению присваивается уникальный идентификатор для ответа клиента. Когда происходит переключение сети, сервер распознает, что у того же самого устройства теперь другой IP-адрес, и можно продолжить существующий мультиплексированный сеанс.

Программное обеспечение, пытающееся использовать HTTP/3, должно при необходимости прибегать к более традиционным протоколам. Например, некоторые сетевые администраторы могут ограничить UDP-трафик за пределами своего периметра.

Система доменных имен

Организация отслеживает все хосты в своих доменах и делится этими данными с другими сайтами через DNS. DNS — это распределенная, иерархическая и реплицируемая база данных, которая сопоставляет понятные человеку имена хостов с IP-адресами.

DNS отделена от регистрации доменных имен. Реестры управляют доменами верхнего уровня (*англ.* top-level domains, TLD), последним сегментом доменного имени, включая *.com*, *.net* и *.org*. Реестры делегируют полномочия по регистрации доменов регистраторам, которые занимаются резервированием доменных имен.

Некоторые системные администраторы управляют DNS-серверами. С появлением облачных сервисов системные администраторы все чаще используют управляемые службы DNS, поскольку это еще одна служба, неправильная настройка которой может привести к сбою в работе всего сайта. У глобальных поставщиков услуг проблемы с DNS все же возникают, но вероятность их появления значительно ниже. При сравнительно невысокой цене за запись этот подход позволяет системным администраторам сосредоточиться на других оперативных задачах.

Протокол DNS состоит из двух частей: обмен сообщениями между клиентом и сервером по схеме «запрос-ответ» для разрешения DNS-запросов и обмена записями базы данных с DNS-серверами.

DNS-запрос состоит из полного доменного имени (*англ.* fully qualified domain name, FQDN) — имени, которое ссылается на конкретный ресурс в иерархии DNS. Кли-

енты и DNS-серверы могут предоставлять ответы на основе данных, кешированных в памяти DNS в течение времени, указываемого в секундах и известного как «время жизни» (*англ.* time to live). В противном случае полномочные серверы предоставляют DNS-ответы.

Помимо разрешения имен DNS также используется для выполнения следующих функций:

Распределение нагрузки

Round-robin DNS можно использовать для распределения нагрузки без дополнительного оборудования или программного обеспечения. Чтобы реализовать Round-robin DNS для службы без отслеживания состояния, включите IP-адреса всех реплик службы в запись DNS. При каждом запросе IP-адреса DNS-сервер будет переупорядочивать ответы с IP-адресами, распределяя запросы хостов. Если одна реплика дает сбой, DNS не проводит никакой проверки, поэтому некоторый процент запросов не достигнет успеха, пока кеш для ее IP-адреса не будет очищен на основе TTL и любого кеширования на стороне клиента.

Управление функциями

Часто клиенты используют полное доменное имя для доступа к сервису. Настроив CNAMES (канонические имена), указывающие на определенные версии развернутого программного обеспечения, вы можете обновлять CNAME при выпуске новых версий ПО.

Обнаружение сервисов

Владельцы доменов определяют конечные точки служб путем добавления SRV-записи с указанием хоста, номера доступного порта, приоритета и веса для указанных служб.

Авторизация и аутентификация электронной почты

С увеличением числа фишинговых и спуффинговых атак были добавлены механизмы, использующие DNS для проверки подлинности электронной почты, включая DKIM и SPF.

Как и в случае с HTTP, первоначальный протокол DNS не имел никаких средств защиты, поэтому он уязвим для проблем, возникающих из-за неправильной конфигурации или вредоносных атак. Например, фальсифицированные или подделанные данные могут быть использованы для атак отравления кеша DNS для перенаправления трафика на вредоносные сайты. Спецификации DNS Security Extensions (DNSSEC) обеспечивают обратно совместимый способ защиты трафика DNS с помощью криптографической аутентификации, обеспечения целостности данных и других усовершенствований. Однако DNSSEC не обеспечивает конфиденциальности путем шифрования самого трафика DNS, поэтому для защиты трафика DNS появились такие протоколы, как DNSCrypt, DNS over TLS (DoT) и DNS over HTTPS (DoH).

Дополнительные ресурсы

Чтобы узнать больше о DNS, ознакомьтесь с этими ресурсами:

- DNS and BIND Cookbook (**https://oreil.ly/OhGCx**), Cricket Liu (O'Reilly);
- «Domain Names: Implementation and Specification RFC 1035» (**https://oreil.ly/26quq**).

Если вы отвечаете за управление электронной почтой или проведение почтовых рассылок в вашей организации, обязательно изучите DKIM, SPF и Domain-based Message Authentication, Reporting, and Conformance (DMARC):

- DKIM (http://www.dkim.org);
- DMARC Overview (**https://dmarc.org/overview**);
- Email Authentication for Internationalized Mail RFC 8616 (**https://www.rfc-editor.org/info/rfc8616**).

Определение причин сбоя тестов

Основываясь на *главе 7*, давайте узнаем больше о различных вариантах сбоя тестов (проблемы с инфраструктурой, ошибочная логика тестирования, неверные предположения, ненадежные тесты и дефекты кода).

Сбой теста, вариант № 1: проблемы, связанные с инфраструктурой

Проблемы с инфраструктурой могут быть очень неприятными, потому что многое может пойти не так, особенно в больших сквозных тестах, которые задействуют различные сервисы. Убедитесь, что проводится достаточное количество модульных тестов, т. к. они не подвержены влиянию проблем, связанных с инфраструктурой. Существует много проблем, причиной которых является инфраструктура:

- тестовая среда не соответствует производственной среде по масштабу или функциям;
- функциональные элементы могут играть негативную роль, например агенты мониторинга влияют на тесты, хотя и не должны;
- отсутствует локальная среда тестирования из-за непонимания того, что можно иметь такую среду;
- зависимости не заданы жестко и варьируются при изменении инфраструктуры;
- в работе сторонних служб непрерывной интеграции и развертывания происходят сбои.

Это лишь несколько примеров инфраструктурных проблем, которые могут стать причиной падения тестов.

Сталкиваясь с проблемами в общих тестовых средах, люди могут настаивать на том, что тестовая среда не нужна и что лучше выполнять тестирование непосредственно на производстве с помощью фича-флагов (*англ.* feature flag) и канареечного тестирования. Фича-флаги делают функции доступными для определенного числа пользователей, в то время как инженеры, ответственные за процесс разработки, проверяют, все ли работает, как задумано, или отключают функцию. Канареечное тестирование предоставляет определенному количеству пользователей доступ к функции или продукту, чтобы определить качество релиза, и если все в порядке, то развертывание продолжается. Если пользователи сообщают о проблемах, вы можете вернуть стандартные условия.

Не существует способа воспроизвести тестовую среду, в точности соответствующую производственной. Поэтому фича-флаги и канареечное тестирование в производственных условиях являются важнейшими способами улучшения обратной связи и снижения риска массовых изменений в производстве.

Они не заменяют необходимость в постоянной и быстрой обратной связи с разработчиками из тестовой среды на ранних этапах. Когда время выполнения растянуто (т. е. приходится долго ждать внедрения в производство), вы теряете контекст своей работы. В данной ситуации речь идет о разнице от минут до дней, которая со временем возрастает.

Кроме того, общие тестовые среды создают безопасное пространство для экспериментов и исследовательского тестирования. Однако руководство может рассматривать общие тестовые среды как необходимость дополнительных расходов, не оценивая должным образом возврат инвестиций.

Один из способов убедить людей, что это не пустая трата ресурсов, — мониторинг и управление созданием и выводом из эксплуатации тестовых сред с помощью автоматизации инфраструктуры. Тестовые среды становятся доступны при необходимости и являются последовательными и повторяемыми, чтобы инженеры, нуждающиеся в тестировании, могли получить доступ, когда им это требуется. Вы сможете исключить простаивающие системы и не тратить время на ожидание в очереди, чтобы воспользоваться одной тестовой средой.

Иногда условия среды находятся вне вашего контроля, например, когда сторонние CI/CD сервисы дают сбои: скажем, GitHub, Travis или CircleCI не работают. Передача этих сервисов на аутсорсинг приносит выгоды в краткосрочной и среднесрочной перспективе, поскольку не требуются специалисты, обеспечивающие их работоспособность на локальном уровне. У сторонних сервисов будут возникать сбои. Это гарантировано. Обратите внимание на следующие вопросы, чтобы спланировать меры по смягчению последствий:

- Как вы будете работать над программным кодом, пока сторонние сервисы не работают?
- Как вы будете проводить тестирование?
- Как вы будете приносить пользу своим клиентам?
- Что делать в случае потери данных?

Если, например, тесты не предупреждают о сбоях, это еще не значит, что все в порядке. Возможно, сторонняя система дала сбой и больше не считает, что управляет проектом.

Сбой теста, вариант № 2: ошибочная логика теста

Иногда проблема, которую вы обнаруживаете, связана с тем, как вы интерпретировали требования или как клиент выразил свои потребности. Программный код работает верно, но тесты дают сбой. Эти неудачные тесты выявляют проблемы

с совместной работой или взаимопониманием. Например, сбой может происходить из-за того, что информация отсутствует, является неясной или противоречивой. Хуже того, вы можете не обнаружить проблему, если не тестируете то, что нужно. При обнаружении сбоев, вызванных ошибочной логикой теста, модифицируйте тест, проанализируйте процессы, которые привели к отсутствию контекста, и устраните их.

Продукты эволюционируют. Иногда их характеристики меняются, начиная с первоначальных совещаний по проектированию до этапа реализации разработчиком, и тесты, которые в какой-то момент были допустимыми, теперь могут давать сбои.

Поэтому, если падение теста вызвано ошибочной логикой тестирования, исправьте тест, а также оцените, где возникла проблема:

- В первоначальных обсуждениях не участвовали необходимые специалисты?
- Согласовывались ли критерии приемки тестирования с требованиями заказчика в процессе сбора требований?
- Когда изменилась реализация, данные обратной связи не попали в обсуждения и не дошли до разработчиков?

В зависимости от вашего конвейера разработки и каналов продвижения программного обеспечения, существуют области, в которых коммуникация и сотрудничество могут быть недостаточно развиты.

Сбой теста, вариант № 3: изменение предположений

Иногда вы делаете предположения о том, что произойдет, и оказываетесь правы, а в другой раз — ошибаетесь. Эти предположения неочевидны до тех пор, пока не изменятся обстоятельства, лежащие в их основе. Например, вы изменили время запуска теста, и теперь какие-то тесты завершаются сбоем, не соответствуя проверенному программному коду. Предположения также могут стать видимыми при изменении порядка действий в определенной задаче и при изменении базы данных.

Неудачи, вызванные изменением предположений, выявляют ранее скрытые недостатки в программном коде или тестах.

Автоматические тесты должны быть детерминированными, поэтому выявление скрытых предположений и изложение их в явном виде поможет устранить неработающие тесты. Иногда вместо сквозного тестирования имеет смысл провести тестирование самих компонентов, чтобы изменения интерфейса не привели к сбоям.

Сбой теста, вариант № 4: ненадежные тесты

Ненадежный тест — это недетерминированный тест, который при одной и той же конфигурации может то выполняться, то заканчиваться сбоем. Обычно они встречаются в более комплексных тестах — интеграционных и сквозных. При обнару-

жении недетерминированных тестов очень важно выполнить их рефакторинг или исключить, чтобы они перестали создавать проблемы.

Вот некоторые распространенные причины, по которым тест может оказаться ненадежным.

Кеширование данных

Полагается ли приложение на кешированные данные? Существует множество типов кешей и способов, с помощью которых они могут вызвать проблемы при тестировании. Например, при использовании веб-кеша файлы, критически важные для рендеринга веб-страницы, могут кешироваться на веб-серверах, на пограничных сервисах или локально в браузере. Если данные в кеше на любом из этих уровней оказываются устаревшими, тесты могут стать недетерминированными.

Жизненный цикл экземпляра теста

Какова политика в отношении установки и переналадки экземпляра теста? Тесты могут быть неверными или возвращать противоречивые результаты, если среда используется повторно или является многопользовательской. Соблюдение чистоты тестовых сред и упрощение настройки и очистки всех экземпляров тестов снижает риск появления ненадежных тестов.

Несоответствие конфигурации

В непоследовательных средах или когда условия тестирования не совпадают с реальными производственными условиями, это может привести к проблемам. Например, предположим, что какой-то параметр зависит от времени, и одна среда синхронизируется с NTP-сервером, а другая — нет. В этом случае могут возникнуть противоречия в реагировании теста, особенно в специфических условиях (например, при переходе на летнее время). Чтобы сохранить согласованность конфигурации, используйте код инфраструктуры для тестовых сред.

Сбои вычислительной среды

Иногда сам тест надежен, но он выявляет скрытые проблемы в вычислительной среде. Необходимо проводить мониторинг, чтобы обнаруживать эти проблемы по мере их появления, а не тратить время на отладку несуществующих проблем.

Сторонние сервисы

Ваша организация будет все больше и больше полагаться на сторонние сервисы, чтобы сосредоточиться на областях, обеспечивающих преимущества для бизнеса. Когда возникают проблемы с этими сервисами, могут возникнуть трудности с интеграцией и сквозными тестами. Поэтому важно изолировать данные проблемы и убедиться, что вы можете точно определить место их возникновения, как и в случае со сбоями в инфраструктуре.

Сбой теста, вариант № 5: дефекты кода

Дефекты кода занимают последнее место в моем списке, потому что именно для поиска дефектов кода вы и создали свои тесты. Таким образом, при поиске причин падения теста проще сосредоточиться, прежде всего, на дефектах кода, а не на других аспектах. Не спешите! Прежде чем приступить к устранению дефектов кода, изучите условия среды, вспомните об ошибочной логике тестирования и проверьте, не изменились ли какие-либо предположения.

Когда вы обнаружили проблему в коде и выяснили, повторяется ли она и как часто, выполните следующие действия:

1. Четко опишите проблему, включив информацию о том, что и как произошло, шаги по ее воспроизведению и версию инфраструктуры (программное обеспечение, приложение, вычислительные среды). Если вы обнаружили проблему вручную, добавьте тест, который в следующий раз обнаружит ее автоматически.
2. Как только вы составите отчет о дефекте, вам необходимо отследить его и убедиться, что кто-то несет ответственность за решение проблемы.
3. Совместно с командой определите приоритетность отчета на основе того, что находится в рабочей очереди. Как только команда устранит проблему, проверьте, что все в порядке, пересмотрите все граничные условия, которые, возможно, необходимо изменить, и закройте отчет о дефекте.
4. Если приоритет ошибки недостаточно высок, чтобы над ней работали или назначили ответственного, подумайте, стоит ли оставлять отчет открытым. Отчеты об ошибках, которые долго не закрываются, создают когнитивную нагрузку на команду.

Я не советую вам сразу же назначать или закрывать все отчеты. Я говорю, что нужно оценивать проблемы и активно выступать за решение тех, которые являются наиболее значимыми.

Предметный указатель

А

Б

В

Г

Д

Ж

З

И

К

Л

М

Ц

Ш

Э

Ю

Об авторе

Дженнифер Дэвис — опытный руководитель инженерно-технической группы, инженер по эксплуатации, международный спикер и автор. В числе других ее книг — «Effective DevOps» и «Collaborating in DevOps Culture». Она страстно поддерживает сообщество, организовывала и участвовала в работе ряда конференций, а также основала CoffeeOps — международное сообщество, способствующее коммуникации и сотрудничеству между компаниями. Дженнифер работала в самых разных компаниях, от стартапов до крупных предприятий, совершенствуя методы обеспечения работоспособности и поощряя улучшение условий труда.

Об обложке

Птица на обложке книги «Современное системное администрирование» — это райский зимородок (*Tanysiptera galatea*). Существует девять видов зимородков, только два из которых обитают за пределами Папуа — Новой Гвинеи (в Австралии и Индонезии). Райский зимородок обитает в тропических лесах острова.

У райских зимородков невероятно яркие синие перья от макушки головы до хвоста и белая грудка, яркий красновато-оранжевый клюв, длинный и заостренный. Два рулевых пера выходят за пределы хвостовых перьев, увеличивая размер птицы до 13–17 дюймов. Зимородки весят около двух унций. Они питаются насекомыми — червяками и кузнечиками, которых находят в лесах, где обитают.

Райский зимородок имеет охранный статус «Виды, вызывающие наименьшее опасение» (Least Concern). Многие животные, изображенные на обложках O'Reilly, находятся под угрозой исчезновения. Все они важны для нашего мира.

Цветная иллюстрация выполнена Карен Монтгомери на основе черно-белой гравюры из «English Cyclopedia Natural History».

O'REILLY®

Современное системное администрирование

Как поддерживать надежную и устойчивую работу систем по мере развития технологий, появления новых инструментов и сервисов? В этом практическом руководстве Дженнифер Дэвис показывает современную инфраструктуру, вычислительные среды, методы и технологии, давая путеводную нить системным администраторам и разработчикам, отвечающим за надежное функционирование вычислительных систем.

Книга посвящена современным практикам и технологиям системного администрирования. Приведены основные сведения о системах, архитектурах, вычислительных средах, хранилищах, сетях. Рассмотрены методы и наборы инструментов сисадмина, вопросы контроля версий, тестирования, документирования и представления информации. Описана сборка системы, разработка сценариев, управление инфраструктурой и обеспечение ее безопасности.

Рассмотрен мониторинг системы, программного обеспечения и работы сисадмина. Особое внимание уделено масштабированию системы, управлению мощностями и созданию надежной дежурной службы.

Вы изучите:

- **основные принципы работы систем и взаимодействия друг с другом отдельных компонентов;**
- **практические методы повышения надежности и устойчивости систем и снижения нагрузки на системного администратора при использовании новых технологий;**
- **приемы сборки систем, автоматизации работы и управления системной инфраструктурой, снижения затрат на денежные и людские ресурсы при техническом обслуживании;**
- **стратегии мониторинга, современные инструменты мониторинга и фреймворки, управление данными наблюдений;**
- **масштабирование системы и повышение ее устойчивости, планирование системы предупреждений и реагирования на инциденты.**

«Дженнифер — ваша путеводная звезда в сфере технологий, она добавляет позитива и экспансивности традиционной работе. В этой книге она подробно раскрывает вопросы профессионального системного администрирования, от основ сетевых технологий до социотехнических факторов, таких как пропускная способность и управление инцидентами. Независимо от того, начинаете ли вы свой путь или уже работаете десятилетия в данной сфере, здесь есть чему поучиться».

— Эми Тобей,
главный инженер
компании Equinix

Дженнифер Дэвис — опытный инженер по эксплуатации, докладчик на международных конференциях и автор ряда книг по DevOps. Она работала с различными компаниями, от стартапов до крупных предприятий, повышая надежность систем и обеспечивая их устойчивую работу.

ISBN 978-5-9775-1848-2

191036, Санкт-Петербург,
Гончарная ул., 20
Тел.: (812) 717-10-50,
339-54-17, 339-54-28
E-mail: mail@bhv.ru
Internet: www.bhv.ru

bhv®