Глава 19

Развертывание службы: основы

В этой главе рассказывается о запуске новой службы, такой как электронная почта, VPN-сервер, установщик операционной системы или веб-приложение. Способ, которым запускается новая служба, так же важен, как и способ ее разработки. Первые впечатления клиентов от службы будут влиять на их отношение к ней в будущем. Первое впечатление запоминается надолго, особенно плохое.

Запуск новой службы может быть самой захватывающей частью проекта. Это кульминация напряженной работы многих людей. После запуска служба является тем, на что вы ссылаетесь и чем гордитесь. Служба будет функционировать на протяжении многих месяцев, но ее запуск требует интенсивной работы и концентрации внимания.

Эта глава посвящена первому запуску службы на основе рекомендуемого процесса, состоящего из шести этапов. Запуск службы также рассматривается в главе 20, но там рассматриваются более сложные методы. В главе 21 рассматривается особый случай, когда запуск требует переноса пользователей со старой службы на новую. Связанные с этим темы, такие как методы обновления динамических служб, описываются в томе 2 этой серии книг.

19.1. Будьте готовы к проблемам

Несмотря на то что запуск службы — это интересный и насыщенный процесс, проблема с запуском новой службы заключается в том, что что-то всегда идет не так. Путь к запуску службы полон трюков и ловушек, которые подстерегают нас в самых неожиданных местах: то пользователи с определенным веб-браузером не могут получить к нему доступ, то служба не работает для удаленных пользователей, то в производственной среде недостаточно места, то документация сбивает с толку и т.д.

Мы должны верить в успех новой службы. В то же время неразумно ожидать безупречного запуска. Тем не менее мы склонны удивляться, когда возникают проблемы, и обычно не планируем контрмеры. Эта глава посвящена работе (и это *действительно* работа), которую мы должны выполнить, чтобы реальность оправдала наш оптимизм.

К счастью, чем больше мы запускаем служб, тем более квалифицированными становимся. Мы накапливаем знания, которые помогают нам предвидеть и преодолевать препятствия. Мы ищем потенциальные проблемы при разработке системы, а не после ее создания. Мы улучшаем нашу способность тестировать крайние случаи. Мы находим лучшие способы общения, делегирования и сотрудничества. Каждая проблема имеет решение. Чем больше мы планируем, тем лучше результат. Чем больше мы запускаем, тем легче все получается.

В массовом сознании закрепилось мнение, что у миссии Apollo 13 были проблемы. Обычно люди думают, что другие миссии обходились без проблем. На самом деле это не так. Каждая миссия Apollo была чревата проблемами. Миссия Apollo 7 столкнулась со множеством мелких проблем, связанных с оборудованием. Во время миссии Apollo 8 память компьютера была частично стерта, так что космический корабль на время потерял ориентацию в пространстве. Во время миссии Apollo 12 приборная панель погасла спустя 36 с после запуска, и астронавты были вынуждены вручную переключиться на резервное питание.

Агентство NASA предвидело возможность возникновения каждой из этих проблем и разработало пошаговые инструкции. Космонавты тренировались выполнять эти инструкции. Особенность проблемы Apollo 13 заключалась в том, что она была непредвиденной. У астронавтов и инженеров NASA не было соответствующей инструкции. Это была незапланированная проблема, и у них было не так много времени, чтобы изобрести решение. Разница между проблемой и кризисом заключается в подготовке к ним.

По мере накопления опыта мы лучше предвидим возможные проблемы. Мы составляем контрольный список потенциальных проблем, которые планируем. Типичный системный администратор накапливает этот контрольный список в своей голове и использует прошлый опыт для лучшей подготовки в будущем. Хороший системный администратор держит контрольный список где-то для справки. Отличный системный администратор поддерживает этот список таким образом, чтобы его могли использовать и обновлять все сотрудники организации. Каждая новая проблема запуска приводит к обновлению этого списка. В результате с каждым новым опытом вся организация становится умнее.

19.2. Шестиэтапный процесс запуска

В общем процессе запуска используются две основные концепции: **список готовности** (ready list) и итерации запусков и обучения.

Список готовности содержит информацию о том, что значит быть готовым к запуску. Мы работаем над этими вопросами либо напрямую, либо через делегирование. Обнаружив ошибки или регрессию, мы вносим в этом список новые пункты. Как только все пункты в списке готовности будут выполнены, мы можем попытаться запустить службу.

Сам запуск — это цикл, в котором мы делаем небольшой запуск, учимся на нем, а затем повторяем. Например, первоначальный запуск может быть бета-версией, а не фактическим запуском, видимым для клиентов. Каждая итерация может привести к добавлению большего количества пунктов в список готовности на основе того, что мы узнали.

После того как система запущена и полностью внедрена в производство, возникает еще больше возможностей для обучения. Мы останавливаемся, чтобы выполнить ретроспективный анализ, оглядываемся назад и размышляем над всем процессом, чтобы осознать полученную информацию и понять, как можно улучшить будущие запуски.

Весь процесс можно описать следующим образом.

- **Шаг 1. Определение списка готовности.** Определите список задач, которые должны быть выполнены до запуска.

- **Шаг 2. Работа со списком.** Создайте календарный план, выполните его и пометьте все пункты в списке как завершенные.
- **Шаг 3. Запуск бета-службы.** Запустите службу на одной из многих тестовых площадок, где проверяются собранные части.
- **Шаг 4. Запуск службы в производство**. Запустите службу в производство.
- **Шаг 5. Извлечение уроков.** Извлекайте уроки на будущее и просвещайте свою организацию.
- **Шаг 6. Повторение.** Начните процесс еще раз.

19.2.1. Шаг 1. Определение списка готовности

Список готовности — это список задач, которые должны быть выполнены, или условий, которые должны быть соблюдены как свидетельства готовности к запуску. Иначе говоря, это список, который определяет, когда работа заканчивается.

Поддержание этого списка помогает членам команды сосредоточиться и избегать лишней работы. Он демонстрирует ваш прогресс руководству, другим командам и клиентам.

Содержание списка готовности

Пункты в списке готовности обычно делятся на четыре категории.

- **Набор обязательных функций**. Существует множество запрашиваемых функций, но запуск не может быть выполнен без обязательных функций. Эта часть списка служит контрактом между клиентом и поставщиком. Такое формулирование ожиданий предотвращает проблему, когда после запуска заинтересованные стороны удивляются тому, что некоторые функции отсутствуют.
- **Набор желательных функций**. Все функции, которые не являются обязательными.
- **Ошибки, подлежащие исправлению**. Тестирование выявит ошибки, которые необходимо отслеживать. Если их не отслеживать, они не будут исправлены или возникнет дублирование усилий, поскольку несколько человек будут пытаться устранить одни и те же проблемы.
- **Условия и одобрения**. Это список условий, которые должны быть выполнены перед запуском. Например, юридический отдел одобрил товарный знак, аудит безопасности завершен, генеральный директор и руководитель отдела управления продуктами подписали соглашение и т.д. Этот контрольный список содержит технические элементы, такие как проверка емкости диска или подтверждение того, что соединение ISP установлено.

Атрибуты списка

Как минимум каждый элемент в списке должен иметь краткое название, приоритет, имя ответственного за выполнение задачи, ссылку на историю задачи и статус.

Приоритеты должны быть стандартизированы для всех проектов во избежание путаницы, чтобы облегчить людям переход от проекта к проекту и не допустить того, чтобы сотрудники тратили время на изобретение колеса в каждом проекте. Для простоты мы сосредоточимся на трех приоритетах функции: обязательные, необязательные и ненужные.

Если было принято решение не включать в службу какую-либо функцию, может быть проще отнести ее к категории "необязательные", чем удалять из списка. Если она будет удалена, то люди не будут знать о решении не включать ее в список.

Если ваша организация имеет стандартизованную схему приоритетов для ошибок, примите ее для использования в списке готовности, чтобы избежать путаницы. Например, в компании Google существует тщательно определенный набор приоритетов ошибок от P0 до P6. Из них определения P1, P2 и P5 хорошо согласуются с категориями обязательных, необязательных и ненужных функций. Поэтому мы бы приняли эти приоритеты, хотя они и не образуют непрерывную шкалу чисел.

Пункты должны быть конкретными и однозначными

Описание пункта должно быть измеримым. Формулировка "Улучшить работу" неоднозначна. Формулировки "Улучшить производительность сортировки на 20 процентов" и "Система включает RAID-контроллер" намного лучше. Один из способов убедиться в том, что цели хорошо сформулированы, — следовать советам Джорджа Т. Дорана (George T. Doran), изложенным в его статье 1981 года *There's a S.M.A.R.T. Way to Write Management's Goals and Objectives*. Цели должны быть конкретными, измеримыми, достижимыми, релевантными и ограниченными по времени.

- **Конкретность**. Цель способствует улучшению ситуации в определенной области.
- **Измеримость**. Успех (или прогресс) цели может быть измерен количественными показателями.
- **Достижимость**. Цель может быть достигнута при наличии доступных ресурсов.
- **Релевантность**. Цель является обоснованной и соответствует более широким целям.
- **Ограниченность во времени**. Цель имеет конечный срок ее достижения.

Доран отговаривает читателей от попыток достичь всех пяти качеств для каждого пункта. Например, указание крайнего срока является излишним для простых отчетов об ошибках.

Хранение списка готовности

Этот список должен быть доступен для просмотра всем заинтересованным сторонам — как членам команды, так и сторонним людям. Эта прозрачность позволяет каждому понять, что происходит за пределами его сферы влияния. Она также помогает руководству видеть прогресс и узкие места. Обновления списка

должны делать те, кто выполняет работу, а не центральный координатор. Это улучшает маневренность.

Ниже перечислены наиболее распространенные места для хранения и ведения списка. Выберите то, что подходит вашей команде и корпоративной культуре лучше всего.

- **Система регистрации ошибок или билетов с маркерами**. Один из методов заключается в присвоении каждому пункту списка идентификатора ошибки или билета и в маркировке пунктов так, чтобы их можно было легко проверить. Например, некоторые системы допускают маркировку. Одна из них присваивает маркер всем элементам в списке готовности. Например, если проект называется "email2017", то каждый пункт помечается маркером `tag:email2017launch01`. Опубликуйте указатель URL страницы состояния запуска, которая запрашивает пункты с этим маркером, отсортированные сначала по приоритету, а затем по статусу.
- **Система ошибок или билетов с контрольными списками**. Некоторые системы отслеживания проблем позволяют создать контрольный (watchlist), или рабочий (hotlist), список, который обеспечивает аналогичную функциональность маркера. В контрольный список добавляются проблемы, требующие решения. Пользователи, подписавшиеся на этот список, могут рассматривать его как единое целое, сортируя и фильтруя его элементы.
- **Доска Kanban**. Доски Kanban могут быть либо карточками-указателями на стене, либо электронными эквивалентами, такими как LeanKit или Trello. Карточки помещаются в столбцы: в столбец "Запланировано" по вопросам, которые пока не обрабатываются, в столбец "В работе" для текущих вопросов и в столбец "Завершено" для выполненных пунктов. История элемента хранится на каждой карточке и может содержать URL-адрес для ошибки или билета, если это необходимо.
- **Интерактивная или общедоступная электронная таблица**. Можно отображать каждый элемент в строке таблицы, используя столбцы для заголовка, приоритета и т.д. Веб-таблицы могут быть просмотрены любым пользователем в любое время, а с помощью системы прав доступа может быть обновлена любым членом команды. Для простых элементов можно хранить историю в электронной таблице. Для более сложных элементов можно указать URL-адрес соответствующей ошибки или билета.
- **Автономный координатор электронной таблицы статуса**. С помощью этой опции назначенный координатор ведет таблицу, в которой перечислены элементы и их статусы. Все обновления проходят через этого человека и он периодически отправляет обновленную электронную таблицу всем членам команды. Например, если вы завершили элемент, то отправите хорошие новости координатору, который затем обновит электронную таблицу. Один раз в неделю координатор отправляет электронное сообщение с последним файлом электронной таблицы в качестве приложения для всех участников. Мы не рекомендуем использовать этот метод по нескольким причинам. Люди всегда будут работать с устаревшей информацией. Наличие стороннего обновления электронной таблицы может стать

источником ошибок. Координатор становится узким местом информации. Прозрачность снижается. Этот метод мог иметь смысл до появления Интернета, но теперь обмен информацией стал легким и удобным. Мы приводим этот метод здесь только для того, чтобы вы могли его распознать, когда увидите.

Преимущество маркировки или использования контрольного списка состоит в том, что он сохраняет и элементы, и их историю в одном месте. Система Kanban пользуется все большей популярностью, поскольку новые онлайн-инструменты упрощают ее использование. Они обеспечивают прозрачность и более современный способ установки темпа выполнения процесса.

Подумайте о том, чтобы использовать системы, которые дают визуальное чувство выполненного долга. Завершенные элементы должны оставаться в списке, но выделяться другим цветом или шрифтом или перемещаться в список завершенных элементов. Если система, которую вы используете, удаляет любые завершенные элементы, значит, вы буквально скрываете свои достижения.

Демонстрация видимых достижений мотивирует команду. Команда увидит, что завершенная часть растет с течением времени, что дает им чувство удовлетворения. Если ваши достижения скрываются и команда не видит прогресса, то она эмоционально истощается.

Визуальная индикация прогресса также помогает в общении с руководством. Желательно точно сообщать о прогрессе руководству, особенно исполнительному менеджменту. Если они заглядывают в список еженедельно, то должны видеть, что все больше и больше пунктов помечаются как завершенные. Если они видят только список того, что осталось сделать, то подсознательно у них возникает убеждение, что вы еще даже не начинали. Руководители высшего уровня получают слишком много информации, чтобы помнить, как выглядел список на прошлой неделе. Неразумно ожидать, что кто-то вспомнит, что на прошлой неделе в списке было 20 пунктов, а на этой неделе — только 15.

19.2.2. Шаг 2. Работа со списком

Теперь перейдем к работе с пунктами списка. Одни мы выполняем сами, другие делегируем.

Если раньше вы никогда не управляли проектом, то вас может удивить, как много работы требуется, чтобы призвать сотрудников к ответственности. Если они не чувствуют, что несут ответственность за работу, которую согласились делать, то они не будут ее выполнять. Это не потому, что они злые или ленивые, просто они обычно перегружены. Срочные предметы привлекают внимание, а другие ускользают. Тем не менее, если мы переусердствуем, то наши замечания будут восприниматься как придирки или мелочная опека. Список готовности помогает обеспечивать подотчетность людей и выполнение календарного плана, не вызывая ворчания и мелочной опеки.

Во-первых, убедитесь, что все знают, что от них ожидается и в какие сроки. Не думайте, что они это "просто знают". Вы можете сделать это, убедившись, что список готовности содержит конкретный, а не расплывчатый результат. Например, "Установщик работает на оборудовании Dell 740DX" лучше, чем "Повышенная надежность установки".

Люди должны знать о сроке выполнения задачи. Поэтому сроки должны быть согласованы всеми и записаны в списке готовности, чтобы не было двусмысленности.

Люди чаще придерживаются крайних сроков, когда сами их устанавливают. По этой причине лучше не устанавливать крайний срок, а попросить человека, выполняющего задание, самому предложить крайний срок. В девяти случаях из десяти этот человек предложит предельный срок, который будет таким же или даже более ранним, чем вы ожидали. Вы можете принять это предложение, и человек будет чувствовать себя хорошо, потому что сам контролировал ситуацию. Вам придется согласовывать более раннюю дату только в том случае, если человек предлагает что-то, что было бы несовместимо с общим графиком. Даже если дата была согласована путем переговоров, человек почувствует некоторое чувство контроля.

Во-вторых, держите людей под контролем. Теперь, когда люди знают, что они должны делать и в какие сроки, вы должны быть уверены, что с них можно спросить.

Некоторые команды используют еженедельные планерки для мониторинга прогресса. Целью планерки является проверка соблюдения сроков и устранение любых препятствий, которые могут помешать соблюдению сроков. Некоторые команды начинают каждый день с очень короткого совещания, которое называется "stand-up". Этот метод заимствован из методологии Agile. Люди стоят в кругу и сменяют друг друга, объявляя о трех вещах: что они завершили вчера, что они планируют делать сегодня и с какими препятствиями они столкнулись. Для того чтобы совещание не затягивалось, обсуждения не проводятся, а просто делаются объявления. Поскольку люди стоят, у них есть физическая причина сделать совещание коротким. Помощь коллегам с их проблемами оказывается только после завершения совещания. После завершения встречи люди немного общаются, как это часто бывает. Однако теперь это общение будет продуктивным, потому что люди инициируют специальные обсуждения по поводу поднятых проблем. Эффективнее обсуждать проблему вдвоем после совещания, чем приостанавливать заседание и ждать окончания обсуждения.

Одному человеку тоже нужен список готовности

Возможно, вы не думаете, что вам нужно поддерживать список готовности, если вы являетесь единственным системным администратором или единственным человеком в проекте. На самом деле все наоборот. В этой ситуации уведомлять других людей о ходе работы может быть еще более важным. Это делает выполняемую вами работу видимой для руководства и держит клиентов в курсе дела, чтобы они задавали вопросы и тем самым не отвлекали вас от вашей работы.

19.2.3. Шаг 3. Запуск бета-службы

Перед запуском для реальных клиентов служба должна быть запущена в изолированной среде для тестирования. Эта изолированная среда называется **промежуточной областью** (staging area). Найденные ошибки и проблемы должны быть добавлены в список готовности.

Концепция промежуточных областей обычно применяется к выпускам программного обеспечения, но вы можете применить ее к любому запуску службы. В промежуточной области вы можете внедрить новую версию операционной системы в свою автоматизированную систему установки. Здесь можно выполнить бета-тестирование с дружественными пользователями, а также перестроить и заново выполнить процесс запуска службы для вновь нанятых сотрудников.

Существует много разных типов промежуточных областей. У каждой из них свое название и предназначение.

- **Dev**. Среда Dev — это Дикий Запад промежуточных областей. Именно там разработчики могут развертывать службы в любое время для любых целей тестирования без каких-либо ограничений.
- **Обеспечение качества**. Разработчики передают новую версию продукта в область обеспечения качества (QA — quality assurance). В результате проверки качества версия либо принимается, либо отклоняется, а также создается список обнаруженных ошибок. Новые версии поступают в эту область в соответствии с графиком процесса обеспечения качества.
- **Пользовательское приемочное тестирование**. Область пользовательского приемочного тестирования (UAT — user acceptance testing) — это песочница для внешних клиентов, выполняющих тестирование новой версии. Они вводят в систему реальные данные, проверяют свои рабочие процедуры или выясняют, как их адаптировать к новой версии.
- **Бета-тестирование или ранний доступ**. Бета-версия — это место, где уполномоченные клиенты получают ранний доступ к новой версии, чтобы создать обратную связь и сообщать об ошибках.
- **Производство**. Здесь пользователи в реальном времени используют эту службу.

Каждый этап имеет критерии входа и выхода. Критерии входа определяют время начала тестирования. Например, в область QA от разработчиков поступает новая версия программного обеспечения, которая, по мнению разработчиков, готова к использованию. Критерии выхода определяют **условия** (gate) успеха или неудачи версии. Например, условием выхода из области QA может быть успешное прохождение автоматических и ручных тестов. Условием выхода из области UAT могут быть отсутствие ошибок P0 и наличие не более трех ошибок P1. Чем масштабнее проект, тем важнее, чтобы эти критерии входа и выхода были ясно сформулированными и документированными.

19.2.4. Шаг 4. Запуск службы в производство

Наконец, мы готовы к запуску. Все необходимое сделано. Кроме того, реализованы многие из обязательных и желательных функций.

Запуск — это и техническая, и коммуникационная задача. Техническое задание включает в себя передачу кода в производственную среду и выполнение любого другого шага, необходимого для предоставления новой версии пользователям.

Запуск может требовать какой-то формы окончательного утверждения. Это утверждение часто принимает форму одобрения исполнительного руководства, менеджера продукта, который подписывает документ, или согласия старшего

члена команды. Позже мы обсудим возможность замены таких механизмов комплексным автоматизированным тестированием.

Общение не менее важно. Нам нужно сообщить, что происходит запуск, чем новая версия службы отличается от предыдущей и, самое главное, какие элементы службы еще не завершены (их необходимо четко перечислить).

Вы никогда не должны запускать службу для всех пользователей сразу. Это слишком рискованно. Запускайте ее для нескольких пользователей, а затем увеличивайте их количество. Выходите за рамки простого бета-пользовательского процесса. Выбирая волны пользователей, такие как отделы или команды, начинайте с наиболее толерантных к риску и заканчивайте самыми консервативными. Выполнение постепенного развертывания — это тема, которую вы еще увидите в этой книге.

19.2.5. Шаг 5. Извлечение уроков

Поздравляем с запуском! Следующим шагом будет изучение этого опыта. Ретроспективное совещание — это совещание, на котором рассматривается процесс запуска. Его называют ретроспективным, потому что оно подразумевает взгляд назад на то, что случилось. Мы определяем, что прошло хорошо, что не получилось и что нужно улучшить. Ретроспективное совещание преследует четыре цели.

- Достижение согласия относительно необходимых улучшений среди участников проекта.
- Разъяснения для всей команды о проблемах, видимых лишь немногим.
- Уведомление руководства о существующих проблемах, особенно о тех, для решения которых потребуются дополнительные ресурсы.
- Распространение полученной информации во всей организации, чтобы все могли извлечь выгоду из новой службы.

Лучше проводить ретроспективное совещание со всеми заинтересованными сторонами и членами команды в одной комнате или в виде видеоконференции. Используйте это собрание для совместного создания документа. Затем этот документ должен быть распространен среди всех участников и помещен в вики-систему для просмотра. Ретроспективный документ должен содержать разделы, включающие резюме, сроки, успехи и неудачи, а также краткосрочные и долгосрочные изменения, которые должны быть сделаны.

- **Резюме**. Что произошло, когда и, вкратце, что прошло хорошо и что следует улучшить в будущем. Этот раздел должен быть написан на нетехническом, деловом языке, который может понять любой руководитель.
- **Хронология**. Краткое описание того, когда проект начинался, когда он закончился, и основных этапов.
- **Успехи**. Что получилось. Отметьте то, что вы сделали правильно.
- **Неудачи**. Что не получилось. Будьте честными. Не обвиняйте отдельных лиц, а описывайте произошедшие факты и их влияние на проект.
- **Краткосрочные изменения**. Что следует сделать по-другому в ближайшее время и без усилий.

- **Долгосрочные изменения**. Что следует делать по-другому в долгосрочной перспективе. Эти изменения достаточно велики и могут потребовать создания нового проекта, распределения ресурсов, утверждения бюджета и т.д.

Как только ретроспективный документ будет завершен, каждое из краткосрочных и долгосрочных изменений должно быть введено в систему отслеживания ошибок, чтобы их можно было сортировать вместе с другими видами запросов. Вот как мы убеждаемся, что проблемы эксплуатации не забыты.

Создание ретроспективного документа и его публикация для всей компании — на самом деле достаточно радикальный шаг. Традиционно предприятия объявляют о своих достижениях и скрывают или искажают информацию о своих проблемах. Люди склонны скрывать свои ошибки, молча бороться с их последствиями и информировать только небольшую группу людей о недостатках. На ретроспективном совещании мы не только обсуждаем результаты, но и раскрываем их для всеобщего обозрения.

Мы делаем это, потому что это наилучший способ учиться. Если мы сохраняем наши проблемы в тайне, то ограничиваемся только теми решениями, о которых думаем. Мы лишаем себя знаний, вытекающих из чужих предложений и переживаний. Если команда не делится своими проблемами и не обучается на примере других команд, то она упускает возможность обучить всю организацию.

Большинство компаний имеют много IT-групп. Представьте себе компанию с десятью IT-группами. Если они не делятся своими проблемами и не учатся друг у друга, то можно представить десять запусков подряд, в которых каждая группа испытывает одни и те же проблемы. Внешние клиенты не понимают, что существует много разных групп. Они просто видят большую компанию, которая, похоже, не может сделать ничего правильного, так как постоянно повторяет свои ошибки.

В традиционной модели наши проблемы определяют наши слабости. Наилучшая модель — это модель, в которой проблемы и решения обсуждаются открыто. Проблемы — это средство, с помощью которого мы узнаем что-то ценное. Каждая проблема — это возможность как можно шире разделить эту новую мудрость. Недостатки — это не слабость, а обнаружение и обсуждение проблемы, что делает организацию более сильной.

19.2.6. Шаг 6. Повторение

Нет такого понятия, как окончательный запуск. Будут выпущены новые версии с новыми функциями, исправлениями ошибок и т.д. Каждая из них требует другого запуска.

Каждый запуск должен иметь собственный список готовности и использовать аналогичный процесс, хотя будущие запуски, как правило, имеют меньший масштаб и не требуют такой бюрократии.

Как описано в разделе 16.6.4, программное обеспечение требует обновления. Даже если вы не добавляете возможности или не исправляете ошибки, базовые библиотеки и каркасы необходимо будет обновить, поскольку в них обнаружатся слабые места с точки зрения безопасности. Они начинают работу свежими, но потом портятся. Джошуа Корман (Joshua Corman (2015)) объяснил это так: “Программное обеспечение скисает, как молоко”.

19.3. Анализ готовности к запуску

Анализ готовности к запуску (LRR — launch readiness review) — это процесс, предусмотренный для обеспечения успешных запусков путем сравнения состояния службы с критериями готовности к запуску (LRC — launch readiness criteria).

Анализ готовности к производству в компании Google

Термин "LRR" происходит от принципов разработки надежных сайтов в компании Google, которая описана в книге *Site Reliability Engineering: How Google Runs Production Systems* (Beyer, Jones, Petoff & Murphy 2016).

19.3.1. Критерии готовности к запуску

К критериям LRC относятся такие основные задачи, как подтверждение того, что заинтересованные лица одобрили запуск, имеется план аварийного восстановления, резервные копии данных работают и прошли проверку и были завершены различные аудиты безопасности. В LRC также входят данные, полученные на основе уроков, извлеченных во время предыдущих запусков.

Одна из основных поисковых систем однажды запустила новый инструмент аналитики. В первый же день он неожиданно стал популярным и привлек в десять раз больше пользователей, чем планировалось. Служба быстро перегрузилась. У компании не было никакого плана в отношении того, как справиться с этой ситуацией, поэтому служба работала недопустимо медленно, так как оперативная группа пыталась внести коррективы и масштабировать систему. Запуск стал для компании пятном на репутации.

У этой организации существовали критерии запуска, в соответствии с которыми менеджеры, руководившие производством продукта, должны были оценить возможную степень его использования и выполнить нагрузочное тестирование, чтобы убедиться, что система справится с такой нагрузкой. Однако из-за неожиданной популярности этой службы организация добавила новый критерий: у команды должен быть план действий в том маловероятном случае, когда служба оказывается в десять раз более популярной, чем ожидалось. Означает ли это, что служба будет отключена (и кто может одобрить это решение) или ограничена небольшим подмножеством пользователей? В любом случае эти решения должны приниматься заблаговременно, чтобы было достигнуто предварительное соглашение о том, что делать, и можно было загодя запланировать необходимые операции.

Обратите внимание на то, что список критериев запуска отличается от процедуры запуска. Последний может сам быть контрольным списком технических шагов, необходимых для фактического запуска, таких как обновления DNS, обновления брандмауэра, изменения балансировки нагрузки и т.д.

Когда все критерии будут выполнены, мы готовы к запуску. Руководство должно быть строгим в применении критериев. Если допускаются исключения, бизнес-риск должен быть определен количественно, чтобы менеджеры понимали последствия того, с чем соглашаются.

19.3.2. Критерии запуска образца

Вот несколько примеров, которые могут появиться в списках критериев готовности к запуску.

- **Существует мониторинг службы**. Служба отслеживается как по времени работы, так и по производительности. Для внутренних систем измеряются степень использования ресурсов, а также количество обращений и ошибок.
- **Предупреждения мониторинга сконфигурированы**. В случае нарушения соглашения SLA, а также возникновения ситуаций, которые могут привести к нарушениям этого соглашения, система мониторинга выдает предупреждение.
- **Резервное копирование и механизм восстановления проверены и работоспособны**. В хранилище данных предусмотрен механизм резервного копирования, а процесс восстановления прошел тестирование.
- **Механизмы стандартной аутентификации, авторизации и контроль доступа протестированы и работают**. Служба использует стандартную службу AAA (authentication, authorization, and access control), и доступ успешно прошел тестирование.
- **Соглашение SLA определено**. Соглашение SLA для службы сформулировано в письменной форме.
- **Создана модель запроса на обслуживание**. Клиенты смогут отправлять запросы на обслуживание с использованием стандартных инструментов, и эти запросы будут обработаны должным образом.
- **Пользовательская документация завершена**. Необходимая для пользователей документация завершена и проверена.
- **Документация по эксплуатации завершена**. Документация, требуемая для операций по запуску службы, завершена, и существует письменная процедура для того, что делать в каждой исключительной ситуации (предупреждения).
- **Обучение пользователей завершено**. Пользователи прошли обучение. Если обучение не может начаться до запуска системы, как люди будут обучаться до того, как их потребность в службе станет критической?
- **Нагрузочное тестирование завершено**. Система была подвергнута нагрузочной проверке с использованием моделирования ожидаемого и оптимистичного уровней трафика.
- **Разработан и протестирован план реагирования на десятикратную нагрузку**. Какие шаги будут предприняты, если в первый день служба получит в десять раз больше пользователей, чем ожидалось? Этот план должен быть написан и проверен.

19.3.3. Организационное обучение

Иногда проблемы, выявленные во время ретроспективного анализа, становятся кандидатами для добавления в контрольный список готовности к запуску. Критерии LRC определяют то, как мы используем обучение в организационном плане.

Без критериев LRC каждая команда, вероятно, будет делать одинаковые ошибки при запуске. Все они находятся в одной среде, с теми же препятствиями и ловушками, и все, вероятно, одинаково наивны, когда делают свой первый запуск, или, возможно, последний запуск произошел так давно, что текущий запуск может считаться первым.

Критерии LRC — это средство улучшить коллективные знания организации. Они передают обучение от одной группы к другой, чтобы люди не повторяли ошибок друг друга.

19.3.4. Техническое обслуживание LRC

По мере роста список LCR может стать бюрократическим бременем. Мы можем предотвратить эту проблему, опасаясь добавлять новые элементы и удаляя устаревшие элементы.

Каждый элемент списка несет расходы для компании. Независимо от того, определяется ли эта стоимость просто ответом на вопрос в форме LRC или реальной компенсацией, это работа, которую кто-то должен делать при каждом запуске.

По мере того как этот список становится длиннее, возрастают опасения. Если это произойдет, то критерии LRC могут стать предлогом для отмены запуска, и компания станет негибкой, неспособной адаптироваться к изменениям и отвечать новым бизнес-потребностям. Кроме того, люди могут найти способы обойти LRC, что означает неофициальные запуски, которые не приносят пользу для обучения других. Эти изменения не происходят моментально. Это долгий и медленный процесс. Просто однажды мы просыпаемся и обнаруживаем, что запуски идут не так, потому что критерии LRC больше не используются.

Список должен расти медленно, на основе опыта. Такой подход к увеличению списка сохраняет реалистичность критериев и препятствует благим намерениям людей добавлять гипотетические ситуации, которые вряд ли произойдут. Это позволяет избежать соблазна добавлять любую, казалось бы, хорошую идею в список.

Элементы должны обновляться и переписываться по мере необходимости. Устаревшие элементы должны удаляться. Это трудно, потому что сокращение списка связано с личным риском, а за добавление элементов никто никакой ответственности не несет. Люди боятся, что их лично обвинят в том, что они предложили удалить элемент, если впоследствии это вызовет проблему или сбой. Больший риск состоит в том, что список станет настолько большим, что запуски станут менее частыми. Исчезнут преимущества принципа малых шагов: каждый запуск будет включать в себя большую партию изменений, люди будут отвлекаться от работы при выполнении запусков и т.д. Это будет препятствовать запуску, что еще больше задержит или заблокирует новые функции. В результате компания станет застывшей. По этой причине обязанность каждого и ответственность руководства — обеспечивать дисциплину при ведении списка.

Наилучший способ удалить элемент из списка — объявить его устаревшим, потому что он стал автоматическим. Например, предположим, что один элемент LRC объясняет, как служба переживает сбои с одним хостом. Каждый запуск проектирует и реализует механизм переключения при отказе, который затем документируется, тестируется, анализируется и оценивается как часть LRC. Если компания создает централизованную службу балансировки нагрузки, которая

может использоваться на основе подписки (“балансировка нагрузки как служба”), труд, связанный с удовлетворением этого элемента LRC, становится значительно проще. Критерии можно удовлетворить, просто показав, что система использует платформенный балансировщик нагрузки по умолчанию.

19.4. Календарь запусков

Крупным организациям, которые участвуют во многих одновременных запусках, требуется некоторая координация этих действий. Обычно эта координация принимает форму календаря запусков. Календарь запусков может быть простым хронологическим списком будущих запусков на вики-системе или чем-то более сложным, например календарем групповой работы, созданным с помощью системы Microsoft Exchange или Google Calendar.

По мере того как организации становятся все больше, становится более вероятным, что некоторые окажутся в неведении, а это может привести к ошибкам. Календарь обеспечивает осведомленность о будущих запусках. В небольшой организации всем легко узнать о будущих запусках, поскольку все, вероятно, в нем участвуют. Наличие общего календаря запуска предотвращает конфликты. Это также облегчает планирование изолированных запусков. Некоторые запуски лучше делать изолированно по техническим причинам или просто для того, чтобы избежать путаницы.

Календарь также помогает командам планировать запуски одна другой. Например, запуск чего-то внешне видимого может потребовать маркетинговых объявлений и другой работы, которая должна быть сделана загодя. Календарь позволяет планировать и координировать эту деятельность.

Одни элементы в календаре будут назначены на обязательную запланированную дату, другие даты запуска являются приблизительными или желательными. Элементы, отложенные на далекое будущем, скорее всего, будут оценками, в то время как элементы, которые должны быть реализованы в ближайшее время, должны описываться более точно. Полезно указывать элементы, дата которых меняется, с помощью маркера, значка или другого цвета.

Каждая команда должна нести ответственность за то, чтобы ее записи в календаре запуска обновлялись.

19.5. Общие проблемы запуска

В этом разделе описаны наиболее распространенные проблемы запуска. Наблюдательный читатель может заметить, что большинства этих проблем можно избежать, используя описанную ранее методологию DevOps. Культура DevOps также поощряет раннее вовлечение операций, которые предотвратили бы другие проблемы.

19.5.1. Нарушение процессов на этапе производства

Процессы развертывания, которые работали на других этапах, могут не сработать на этапе производства. Часто процесс развертывания по-разному реализуется в средах разработки, тестирования, бета-тестирования и производства. Это часто происходит, если за каждый этап отвечает другая команда, и они не

обмениваются информацией или не делятся кодом. Члены каждой команды разрабатывают собственный процесс инсталляции. Часто производственная среда совершенно отличается от остальных, и производственная команда обнаруживает, что сама разрабатывает процедуру "с нуля", не используя опыт развертывания на других этапах. Или, возможно, среда бета-версии и производственная среда достаточно схожи, так что один процесс должен работать для обоих. Тем не менее производственная среда может измениться за многие месяцы, прошедшие с момента последней передачи кода, и процесс будет нарушен. В любом случае урок здесь заключается в том, что лучший опыт запуска достигается благодаря наличию единого механизма развертывания на всех этапах. Он должен быть разработан как часть кода, а не как что-то, что осталось сделать в тот день, когда служба будет передана на производство. Одна из целей контейнеров, таких как Docker, — обеспечить единый формат доставки, который работает на всех этапах.

19.5.2. Неожиданные методы доступа

То, как пользователи получают доступ к службе, различается на всех этапах, предшествующих производству. Многие запуски служб выходят из строя, поскольку тесты выполнялись с использованием одного метода доступа, а клиенты фактически обращаются к нему иначе. Например, в одной компании предполагали, что протокол SSL сети VPN будет обеспечивать доступ к службе для 80% пользователей, но разработчики и тестеры использовали эту службу без VPN. В другом случае система была протестирована только локальными пользователями с быстрыми сетями. Никто не думал, что реальные пользователи будут получать доступ к системе по каналам с высокой задержкой, что приведет к сбою некоторых протоколов. Урок здесь заключается в том, чтобы убедиться, что тестирование учитывает метод доступа, который будут использовать реальные пользователи.

19.5.3 Недоступность производственных ресурсов

Обычно делаются предположения о ресурсах, доступных в производстве. Предположим, незадолго до запуска разработчики объявили, что для новой службы потребуется большой объем дискового пространства, ресурсов центрального процессора или пропускной способности. Разработчики просто предположили, что эти ресурсы будут доступны, не указав заранее их требования. Реальная логистика установки дополнительных серверов или получения большей пропускной способности является сложной и часто занимает недели или месяцы предварительной работы. Разработчики часто бывают шокированы, узнав, что нет петабайта свободного места на диске, которое они могут использовать, или что добавление пропускной способности ISP может занять 12 или более недель. Урок здесь состоит в том, чтобы вовлекать операции с планированием ресурсов и мощностей на раннем этапе разработки, а не в конце.

19.5.4. Новые технологические сбои

При первом использовании новой технологии запуск отнимает больше времени. При первом использовании новой технологии потребуется значительно больше времени, чтобы заставить ее работать на производстве. Это относится к новому языку, структуре, базе данных или другой службе. Например, если ваша организация использует язык C#, то в первый раз, когда вы пытаетесь использовать

NodeJS в производстве, вы обнаружите десятки неожиданных проблем, каждая из которых требует недели экспериментов, проектирования и тестирования. Если у вас есть большой опыт работы с одной базой данных и разработчик решает, что часть новых данных должна храниться в другой базе данных, разработчик может быть удивлен, обнаружив, что добавление этой базы данных требует новых знаний и разработки новых процедур. К сожалению, приверженцы новой технологии не могут представить, что любая из этих задач займет какое-то время, потому что новая технология решает все проблемы (или они так думают). В результате об использовании этих новых технологий могут даже не предупредить до последней минуты, и запуск будет отложен. Урок здесь в том, что новые технологии требуют недель и, возможно, месяцев работы, прежде чем они смогут быть использованы в производстве. Этот фактор следует взвесить как часть решения о внедрении новой технологии. Описания таких неудач описаны в статьях, таких как *Why MongoDB Never Worked Out at Etsy* (McKinley 2012) и *Providence: Failure Is Always an Option* (Punyon 2015).

19.5.5. Отсутствие обучения пользователей

Отсутствие обучения пользователей приводит к катастрофе. Для большинства новых служб пользователям требуется обучение или по крайней мере простые снимки экранов, которые демонстрируют все этапы доступа к системе. Во многих случаях причиной задержки обучения является следующий парадокс: обучение требует запуска системы, но запуск не может быть выполнен до завершения обучения. Поэтому организация решает начать работу без обучения и попадает в сложную ситуацию, когда люди не могут вернуться к старой системе (она отключена) и не знают, как использовать новую. Одним из решений является создание промежуточной области, используемой исключительно для учебных целей. Это обеспечивает платформу для обучения, не нарушая работу унаследованной службы производства. Урок здесь заключается в том, что процесс обучения должен быть запланирован так же, как любая другая функция.

19.5.6. Отсутствие резервных копий

Предположим, что незадолго до запуска выясняется, что новая система не имеет плана аварийного восстановления. Система построена таким образом, что сделать резервное копирование и восстановление очень сложно. Что еще более важно, для проверки процесса восстановления может потребоваться несколько недель, а может и месяцев, пока он не станет достаточно надежным. Урок здесь заключается в том, что функции аварийного восстановления должны быть встроены в службу и не могут разрабатываться вдогонку.

19.6. Резюме

Запуск службы требует большой координации. Мы должны быть готовы к неожиданным сюрпризам, особенно в первый раз, когда используем новую технологию или находимся в новой среде.

Существует шесть шагов для запуска новой службы. Мы определяем конечную цель с точки зрения списка готовности или задач, которые должны быть выполнены. Мы работаем над этим списком. Мы запускаем бета-тестирование,

а затем запускаем его в производство. Мы фиксируем извлеченные уроки и повторяем этот процесс.

Список готовности следует хранить там, где каждый имеет к нему доступ, чтобы повысить прозрачность и коммуникацию. Подумайте об использовании системы отслеживания ошибок или запросов, доски Kanban или общей электронной таблицы. Такая координация важна даже для команды, состоящей из одного человека.

По мере увеличения количества запущенных служб мы начинаем накапливать лучшие практики и рекомендации, которые помогут обеспечить успех будущих запусков. Мы можем поделиться этими уроками, собрав их в контрольный список готовности к запуску, используемый для всех запусков.

Некоторые общие проблемы запуска, которые следует учитывать, включают различия между тестовыми и производственными средами, эксплуатационные трудности при первом использовании новой технологии на производстве и отсутствие планирования ресурсов и мощности.

Чем больше запусков вы выполните, тем лучше вы будете запускать новые службы.

Упражнения

1. Опишите шестиэтапный процесс запуска.
2. Этапы 5 и 6 процесса запуска не являются частью фактического запуска. Почему они важны?
3. Что такое список готовности?
4. Как поддерживался список готовности в предыдущем проекте, над которым вы работали? Каковы были плюсы и минусы в его сохранении? Что бы вы сделали по-другому?
5. Зачем команде, состоящей из одного человека, нужен список готовности?
6. Если система достаточно протестирована, для чего ей нужна бета-стадия?
7. Что такое ретроспективный документ? Каковы его основные компоненты?
8. Какие критерии готовности к запуску (LRC) вы применили бы к запуску нового почтового сервера?
9. В разделе 19.5 описаны некоторые общие проблемы запуска. Что вы испытали? Что можно было сделать, чтобы их предотвратить?
10. Проект. Пять глав этой книги посвящены управлению сложными процессами, которые приносят что-то новое в организацию: запуск новой службы (эта глава); запуск через быстрые итерации (глава 20); перемещение пользователей со старой службы на новую (глава 21); модернизация одной сложной машины (глава 33) и начало организационных изменений (глава 37). Составьте диаграмму со строкой для каждой основной рекомендации и столбец для каждой главы. Поставьте символ в поле, чтобы указать, что рекомендация была сделана в этой главе. Какие рекомендации являются наиболее распространенными? Какие из них относятся только к конкретным ситуациям?