DOI 10.37882/2223–2966.2020.05.08

АНАЛИЗ СУЩЕСТВУЮЩИХ ПРОБЛЕМ В ОРГАНИЗАЦИИ НЕПРЕРЫВНОЙ ДОСТАВКИ И РАЗВЕРТЫВАНИИ ПРОГРАММНОГО ПРОДУКТА

ANALYSIS OF EXISTING PROBLEMS IN ORGANIZING CONTINUOUS DELIVERY AND DEPLOYING THE SOFTWARE PRODUCT

K. Bryukhanov
V. Petrov
E. Avksentieva

Summary. The article describes some of the problematic issues associated with delivering the deployment of software product code in IT companies, as well as some recommendations from some of the best international practices. The issues discussed touch upon the problems of the DevOps methodology, which has been actively developed recently. The main trends in support and support of the development process in modern realities are considered. There is an urgent need to include continuous delivery and deployment methods in software engineering courses, as well as teaching continuous integration and assembly methods, automated integration, system testing and code verification. In addition, it highlights the need for education about the realities of deploying code in large-scale work environments and issues related to data migration, deployment strategies and pipelines, and standards for developing telemetry collection and transmission mechanisms.

Keywords: devops, continuous delivery, continuous deployment.

Брюханов Константин Владимирович
Аспирант, Университет ИТМО
devops.spb@gmail.com
Петров Валерий Владимирович
Аспирант, Университет ИТМО
mu_valera@mail.ru
Авксентьева Елена Юрьевна
К.п.н., доцент, Университет ИТМО
avksentievaelena@rambler.ru

Аннотация. В статье описываются некоторые проблемные моменты, связанные с доставкой развертыванием кода программного продукта в IT-компаниях, а также указаны некоторые рекомендации из числа лучших международных практик. Рассмотренные вопросы затрагивают проблемы методологии DevOps, которая активно развивается в последнее время. Рассмотрены основные тенденции в поддержке и сопровождении процесса разработки в современных реалиях. Постулируется насущная необходимость включения методов непрерывной доставки и развертывания в курсы обучения программной инженерии, а также преподавания методов непрерывной интеграции и сборки, автоматизированной интеграции, системного тестирования и верификации кода. Кроме того, выделяется необходимость просвещения относительно реалий развертывания кода в крупномасштабных рабочих средах и проблем, касающихся миграции данных, стратегий и конвейеров развертывания, а также стандартов разработки механизмов сбора и передачи телеметрии.

Ключевые слова: devops, continuous delivery, continuous deployment.

В данный момент, одним из самых востребованных направлений в IT-эксплуатации является наладка и обеспечение непрерывной доставки и развертывания программного кода продукта на производственные серверы, с помощью разнообразных комбинаций технологий и инструментов. Новые требования к доставке и развертыванию, продиктованные развитием технологий и методологий разработки, включают в себя постоянную готовность и непрерывность тестирования изменений, подготовку и настройку тестовой среды для команды обеспечения качества. Кроме того, постоянное технологическое развитие сделало возможным и доступным большое количество сервисов, аппаратного обеспечения, которые ранее не рассматривались для использования в производстве. Таким образом данный технологический прорыв, а также успех свободно распространяемого ПО, привели к тому, что подход к организации процессов CI/CD значительно изменился. Переход на новые принципы привел к масштабным переменам в мире разработки ПО, оказав сильное влияние на корпоративную культуру, востребованные навыки сотрудников и сами принципы работы в организациях.

В связи с развитием технологии облачных вычислений облачная среда становится наиболее перспективным путем предоставления клиентам отраслевых решений. Для реализации непрерывной поставки программного обеспечения группы разработки, тестирования и эксплуатации должны эффективно взаимодействовать и работать совместно. Облачная среда хорошо подходит такого взаимодействия. Однако этап развертывания, выполняемый в сложной распределенной топологии, подвержен ошибкам и, как правило, требует ручного поиска и устранения неисправностей. Этап развертывания продуктов в рамках процесса непрерывной

поставки часто создает узкие места и отрицательно влияет на эффективность процесса DevOps.

Целью непрерывной поставки является автоматизации тестирования инкрементальных изменений программного обеспечения и в оперативном развертывании обновленных версий на рабочей платформе максимально эффективными и безопасными методами. Изменения любой части программной системы (на уровне инфраструктуры, приложения или данных настройки) непрерывно реализуются в производственной среде посредством конвейера поставки. Такой подход дает пользователям уверенность, что в производственной среде используется самая свежая версия кода, и изменения, вносимые программистами, могут попадать к клиентам за считанные часы или даже минуты.

Рассмотрим наиболее общий сценарий реализации CI/CDв проекте:

1. Команда разработки выпускает новую версию продукта (новый функционал или исправление ошибок предыдущего релиза)
2. Сервис непрерывной интеграции (CI) проверяет новый код рядом тестов, включающих несколько уровней тестирования, такие как синтаксические, модульные и регрессионные тесты
3. В случае успеха, сервис непрерывной интеграции готовит новую сборку программного продукта и совершает системный вызов к сервису, осуществляющему непрерывную доставку (CD)
4. Совместно с сервисом непрерывного развертывания (часто, этот функционал выполняет один и тот же сервис), автоматическим образом поднимается сущность, отвечающая за промежуточное развертывание (staging), где, в условиях близким к производственным, проводятся дополнительные интеграционные, дымовые и нагрузочные тесты
5. После успешного проведения stage-тестов, система CI/CD доставляет новую версию продукта на производственные мощности, где выполняется бесшовное развертывание параллельной инсталляции продукта
6. Производится переключения пользовательского траффика на новую версию ПО

Данный сценарий является наиболее общим и покрывает большинство нужд команд разработки и эксплуатации, но, всё же, имеет некоторые проблемы, например:

1. Замена файлов. Зачастую требуется обновление или замена конфигурационных файлов, или перегенерация какого-то статичного контента. В этом случае, пользователи могут получать ошибки, пока не произошло переключения траффика со старой версии ПО на новую. В случае же, когда развертывание закончилось неудачей, есть риск несовместимости файлов.
2. Обновление системных ресурсов. Предположим, ваша система конфигурируется под самообслуживание, настраиваются планировщики и системные демоны. Во время развертывания новой версии, у вас могут заменяться команды и префиксы для системных вызовов. Однако, существует вероятность, что какие-то из этих функций потребуются для работающей на текущий момент предыдущей версии приложения, из-за чего оно не будет должным образом обслужена и работа сервера может быть нарушена.
3. Конфигурирование баз данных. Изменения конфигурации базы данных порождают наиболее сложные проблемы. Обновление решения приводит к изменениям структуры таблиц, пользовательских полномочий, сохраненных данных и т.д. При выполнении развертывания, база данных может стать недоступна для работающего приложения, на время применения новых миграций. Также, существует вероятность ошибки развертывания, из-за чего база данных уже не будет соответствовать требуемой для нормальной работы приложения, и даже повторное развертывание прежней версии не решит проблему, т.к. потребуется разработка специальных обратных миграций.

Стоит заметить, что перечисленные выше проблемы — могут возникнуть даже в близкой к идеальной среде, но одна из основных сложностей внедрения методологии DevOps состоит в том, что нет единой картины, как должен выглядеть процесс непрерывной доставки и развертывания продукта. Многие IT-компании знают о DevOps слишком мало, иногда и вовсе не понимают этой методологии, в других же уже есть исторически сложившиеся решения, поверх которых приходится выстраивать новые процессы. С учетом высоких требований к квалификации devops-специалистов, и их нехваткой на рынке труда, работодатель зачастую вынужден использовать уже имеющиеся у него ресурсы, и отдавать devops-задачи на инженеров, которые еще не успели изучить тему в должной мере. Как итог, в системе получается больше слабых мест, чем в рассмотренной выше версии.

Из этого можно сделать вывод, что отсутствие правильного понимания методологии и аналитический подход к построению инфраструктуры и методов доставки кода — является одной из ключевых проблем при использовании CI/CD. Отсюда же вытекают и следующие возможные проблемы:

1. Человеческий фактор. Первый и самый существенный риск связан с человеческим фактором. Представим себе ситуацию, когда необходимо настроить ещё несколько серверов, аналогичных существующим. Если специалист, который производил предыдущие установки или настройки в данный момент недоступен по каким-либо причинам (болен, уволился и т.д.) и не подготовил подробных инструкций, то ситуация значительно осложняется. В этом случае каждый новый специалист должен изучить весь процесс настройки сервера полностью, при этом у него нет права на ошибку. В результате будет потрачено неопределенное количество времени на настройку, при этом результат не может быть гарантированно успешным. Кроме этого, всегда есть вероятность, что автор метода настройки — тоже допустил ошибку, забыл покрыть тестами процессы, отвечающие за доставку и развертывание, или просто не учел какой-то момент, который не вызывал проблем при прошлых развертываниях. Не стоит также забывать, что, нередко, в компаниях разрабатываются несколько проектов, а отдел IT-эксплуатации, обычно, один, и один инженер эксплуатации обслуживает несколько проектов. Если нет единой схемы и концепции, то процессы в разных командах будут выстроены по-разному, что значительно затруднит последующее развитие devops в компании, и сделает высокий порог вхождения инженера эксплуатации в другой проект, где уже используются процессы, отличные от тех, с которыми он работал ранее.
2. Неидемпотентные сценарии. Идемпотентность является важнейшим атрибутом сценариев непрерывной доставки и развертывания кода, особенно в условиях развертывания инфраструктуры. Инженер должен быть уверен, что каждый раз, при выполнении сценариев, результат будет однозначно гарантированным и ожидаемым, и неизменным, при повторном проигрывании того же сценарии. Зачастую, при внедрении devops в компании, инженеры стараются разработать бизнес-решение, и могут не учесть идемпотентность, или же просто не знать об этом требовании. Такой вариант представляет из себя бомбу замедленного действия, т.к. существует вероятность доставить на производственные мощности неожидаемый код (например, если кто-то обновил модуль системы управления конфигурациями для одного проекта, и тем самым повлиял на остальные, где этого не ожидают).
3. Хранение чувствительных данных и организация доступа. Один из самых важных моментов в devops — подход к хранению секретных данных, ограничению прав, организации сетевого и пользовательского доступа. До сих пор нет единых принятых практик и инструментов, позволяющих решить эту проблему, и инженерам приходится каждый раз проводить исследования, в зависимости от текущей организации инфраструктуры и принятых методов ограничения доступа. По этой причине, внедрение методологии devops на предприятии усложняется тем, что нельзя однозначно найти решение под свой частный случай, а применение чужих практик — не всегда гарантирует безопасность.
4. Определенная модель бюджетирования, более подходящая для Waterfall методологии.
5. Высокие требования к безопасности, следовательно, невозможность размещения инфраструктуры национальных IT проектов в зоне ответственности коммерческих, иностранных компаний, например, Amazon, Microsoft.
6. Высокий объем “legacycode”, “legacyinfrastructure”, которые необходимо поддерживать. Необходимость интеграции с большим количеством устаревших систем.

Из сказанного выше, можно сделать вывод, что процесс построения devops на предприятии может нести за собой ряд проблем, и не всегда решать задачи, для которых создан.

Первым важным шагом является отказ от отношения к серверам, как трудно настраиваемому, уникальному элементу инфраструктуры, переход от ручной настройки сервера к автоматизированному, централизованному управлению инфраструктурой. Таким образом процесс настройки каждого сервера должен быть описан в виде конфигурации, легко читаемой, изменяемой, и готовой к многократному безопасному переиспользованию, обеспечивающей однозначный гарантированный результат. Примерами промышленных систем-оркестраторов являются Chef или Ansible. Данные системы позволяют управлять большим количеством серверов с минимальными затратами.

Следующим важным шагом является применения автоматизированного тестирования для как можно большего покрытия функционала разрабатываемого кода (как программного, так и инфраструктурного). Иначе говоря, имея развернутую инфраструктуру, но без автоматизированного тестирования, узким местом процесса разработки будет своевременная проверка функционала. Автоматизирование процесса тестирования должно начинаться с собственно написания кода ПО (модульное тестирование), применение первичных тестов на сервере, отвечающим за сборку ПО, а также тест конфигурации серверов. Это позволит снизить нагрузку на команду обеспечения качества ПО и значительно снизит время

прохождения ПО на конвейере. Заключительным логичным шагом является централизованный сбор и анализ лог-файлов со всех серверов, для своевременного оповещения всех заинтересованных лиц и проактивном наблюдении за состоянием инфраструктуры в целом. Следование перечисленным выше рекомендациям позволит построить устойчивую, масштабируемую инфраструктуру, способную работать в интенсивном процессе разработки.

Внедрение DevOps требует вовлеченности в процесс каждого участника процесса, от отделов тестирования и разработки, до менеджеров и отдела эксплуатация. На каждом этапе, требуется постоянный ретроспективный анализа процесса — ведь в результате случайных ошибок при изменении конфигурации полностью останавливается работа систем. Необходимо улучшать телеметрию для обеспечения более успешного обнаружения ошибок и восстановления, защиты конвейера развертывания и достижения целей по изменению управления — чтобы получать максимальную поддержку со стороны руководства при внедрении инициатив DevOps, создать более оживленную и дружелюбную рабочую среду, чтобы любой участник смог учиться в течении всего времени — это не только поможет каждому исполнителю достичь целей, но и приведет организацию к успеху.

ЛИТЕРАТУРА

1. J. Paul Reed, DevOps in Practice — O'Reilly Media, Inc.— 2014, p25.
2. Liz Gallacher, Helen Morris. ITIL Foundation Exam Study Guide.— John Wiley & Sons.— 2012, p320.
3. Amirov A. Zh., Gerhardt E., Hon M. V. Features of the process of software deployment in conditions of intensive development // Young scientist.— 2016.— No. 9.— S. 44–47.— URL https://moluch.ru/archive/113/29456/ (accessed: 02.11.2019).
4. Z. Zhu. Ensuring Continuous Delivery When Deploying Industry-specific Cloud Platform Solutions — IBM — 2014 — URL https://www.ibm.com/developerworks/ru/library/deploy-industry-solutions-cloud-platform/
5. G. Kim, P. Debois, J. Willis, J. Humble, The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations — IT Revolution Press — 2016, p11–46
6. G. Kim, P. Debois, J. Willis, J. Humble, The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations — IT Revolution Press — 2016, p187–239
7. G. Kim, P. Debois, J. Willis, J. Humble, The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations — IT Revolution Press — 2016, p293–319
8. J. Davis, K. Daniels, Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale — O'Reilly Media, Inc.— 2016, p45–49.
9. J. Davis, K. Daniels, Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale — O'Reilly Media, Inc.— 2016, p71–79.
10. N.R. Murphy, B. Beyer, C. Jones, J. Petoff, Site Reliability Engineering: How Google Runs Production Systems — O'Reilly Media, Inc.— 2016, p40–46.
11. N.R. Murphy, B. Beyer, C. Jones, J. Petoff, Site Reliability Engineering: How Google Runs Production Systems — O'Reilly Media, Inc.— 2016, p447–463.

Журнал «Современная наука: актуальные проблемы теории и практики»