электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

 # 07, июль 2017

УДК 004.45

Методология DevOps

***Яковенко М. А.**, студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Информационные системы и телекоммуникации»

Научный руководитель: Павлов Ю.Н., д.т.н., профессор
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Информационные системы и телекоммуникации»
iu3@bmstu.ru

Введение

Существует огромное множество определений термина DevOps. Первоначальное знакомство с этим термином может вызвать путаницу, так как в разных источниках он может трактоваться по-разному. Далее будет приведено определение, которые наиболее точно раскрывают суть методологии.

DevOps – (акроним от англ. development (разработка) и operations (ИТ-подразделение)) — набор практик, нацеленных на активное взаимодействие и интеграцию специалистов по разработке и специалистов по информационно-технологическому обслуживанию. Базируется на идее о тесной взаимозависимости разработки и эксплуатации программного обеспечения, и нацелен на то, чтобы помогать организациям быстрее создавать и обновлять программные продукты и сервисы [1].

Принципы DevOps

Существует несколько основных принципов DevOps. Детально рассмотрим каждый из них:

Культура.

DevOps предполагает под собой новую культуру общения, в которой нет барьера между командами. Должна быть организована рабочая среда с развитой коммуникацией и доверием, где можно быстро получить ответ на свой вопрос, а не ожидать его днями и неделями.

Автоматизация.

Автоматизация — это наиболее видимый аспект DevOps. Она позволяет не только сэкономить время, а также способствует устранению многих дефектов. Также автоматизация делает доступным самообслуживание, т.е. каждый участник команды может самостоятельно произвести развертывание, тестирование и многие другие сложные технологические процессы. В то же время нет необходимости автоматизировать всё, сначала нужно проанализировать процессы, которые нуждаются в автоматизации, так как автоматизация сложного процесса порой бывает неоправданной и отнимет в разы больше времени, нежели выполнять этот процесс вручную [2].

Измерения.

Как можно добиться постоянных улучшений без возможности измерить эти

улучшения? Как узнать, стоит ли задача автоматизации? Поэтому очень важно снимать метрики, чтобы понять эффективность процессов, которые мы привносим.

Обмен знаниями.

Обмен передовым опытом уже давно имеет решающее значение для постоянного совершенствования. Находя людей с аналогичными потребностями в организации, можно найти новые возможности для сотрудничества, а также устранить дублируемую работу.

Преимущества от использования DevOps

Ниже перечислены основные преимущества от использования DevOps:

- Быстрая поставка
- Снижение частоты отказов
- Более стабильная инфраструктура
- Улучшенная масштабируемость
- Сокращение расходов
- Больше свободного времени для совершенствования продукта, написания нового функционала и т.п. (вместо исправлений / поддержки)

Инструменты DevOps

Для разработчиков наибольший интерес представляет инструментарий DevOps. Инструментов очень много и все они делятся на несколько категорий. Нужно сказать, что в каждой категории есть инструмент с открыты исходным кодом, использование которого совершенно бесплатно.

Категории инструментов DevOps:

- Системы контроля версий
- Инструменты для сборки с развертывания приложений
- Инструменты для функционального и нефункционального тестирования
- Инструменты для контроля и логирования изменений конфигурации

Процесс разработки с использованием практик DevOps

Ниже будет пошагово описан процесс разработки программного продукта с использованием инструментов DevOps.

1 Шаг. Написание исходного кода.

При разработке продукта, первое, что заботит разработчика, это обеспечение целостности и сохранности кода. В этих целях используются системы контроля версий, которые регистрируют изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов, позволяют определить кто и когда сделал то или иное изменение, и многое другое. На данный момент наиболее популярная система контроля версий это Git, менее популярны Subversion и Mercurial.

2 Шаг. Сборка.

Простые проекты можно собрать в командной строке. Если собирать большие проекты с кучей зависимостей с командной строки, то команда для сборки будет очень длинной. Для того, чтобы упростить этот шаг разработки, используют специальные системы для сборки проекта. Наиболее популярные из них это: Maven, Ant, Gradle, CMake.

3 Шаг. Тестирование.

Тестирование — это неотъемлемый этап разработки ПО. Тестирование является очень трудоемким процессом, а зачастую может отнять больше времени, чем сама разработка. Автоматизация тестов очень важна для непрерывной поставки. После разработки авто-тестов их можно быстро и многократно использовать, и быть уверенным, что при добавлении новой функциональности, старый функционал не сломан и исправно работает. Таким образом регрессионное тестирование доступно «по нажатию одной кнопки». В этой категории очень много инструментов. Перечислим наиболее популярные из них: JUnit, FitNesse, Selenium, Karma, JMeter.

4 Шаг. Хранение артефактов.

Сборка может проходить несколько раз в день (при хорошо поставленном процессе) и необходимо место, чтобы хранить все артефакты. Хранилище артефактов используется для того, чтобы оптимизировать загрузку бинарных файлов и их метаданных. Таким образом хранилище централизует использования бинарных файлов, когда-либо созданных разработчиками.

Ниже перечислен основной функционал, который предоставляют хранилища артефактов:

- Стабильный и надежный доступ
- Безопасность и управление доступом
- Отслеживание изменений и контроль версий
- Быстрая передача бинарных файлов на удаленные машины

Наиболее популярны в этой категории следующие инструменты: Nexus, Artifactory, Docker Hub и npm.

5 Шаг. Непрерывная интеграция.

Инструменты непрерывной интеграции направлены на то, чтобы непрерывно проверять статус программного обеспечения. Как только программный продукт были внесены изменения, запускается автоматическая сборка, тем самым облегчает для всех просмотр текущего состояния продукта.

Как правило подобные инструменты поддерживают следующие функции:

- История сборок
- Сложное управление сборкой (создание интеграционных цепочек)
- Отчет о ходе работ
- Отправка уведомлений

Инструментов непрерывной интеграции очень много, перечислим несколько из них: Jenkins, TeamCity, Visual Studio, Bamboo.

6 Шаг. Развертывание.

Инструменты развертывания упрощают автоматизацию развертывания. Они обычно поддерживают следующие функции:

- Развертывание без доступа к серверам
- Конфигурации, специфичные для окружающей среды
- Интеграция с системами непрерывной интеграции
- Интеграция с облачными платформами
- Автоматический откат
- История, отчеты и аналитика

Наиболее популярны следующие продукты: Ansible, Chef, Puppet, Vagrant.

7 Шаг. Контейнеризация.

Контейнер приложений - это метод виртуализации уровня операционной системы для развертывания и работы распределенных приложений без запуска всей виртуальной машины для каждого приложения. Вместо этого несколько изолированных систем запускаются на одном управляющем хосте и получают доступ к одному ядру.

Контейнеры приложений содержат компоненты, такие как файлы, переменные среды и библиотеки, необходимые для запуска требуемого программного обеспечения.

Поскольку ресурсы распределяются таким образом, можно создавать контейнеры приложений, которые меньше нагружают общие ресурсы [3].

Самым популярным продуктом контейнеризации является Docker. Но есть и другие продукты: Mesos, Swarm, rkt.

8 Шаг. Инструменты управления релизами.

Системы управления релизами предназначены для менеджеров релизов, в отличие от систем автоматизации развертывания, предназначенных для системных администраторов и администраторов приложений.

Основными задачами менеджмента релизов является:

- Планирование релиза
- Координация проектирования, создания и настройки релизов
- Согласие на прием в эксплуатацию
- Планирование развертывания
- Координация распространения и установки релизов

Заключение

DevOps – это культура, которая подразумевает близкое сотрудничество разработчиков и специалистов по ИТ обслуживанию. С технической точки зрения, автоматизация - это ключевой фактор для развития DevOps. Автоматизация не всегда оправдывает затраченное на неё время, поэтому автоматизируемые процессы сперва требуют аналитики. Для автоматизации и упрощения этапов разработки существует множество инструментов (в том числе с открытым исходным кодом). Финальной целью этой методологии является создание качественного, протестированного продукта в более короткие сроки.

Список литературы

[1]. DevOps. Available at: https://ru.wikipedia.org/wiki/DevOps, accessed 15.05.17.

[2]. 11 важных вещей, которые нужно знать про DevOps. Режим доступа https://habrahabr.ru/company/scrumtrek/blog/166039/ (дата обращения 15.05.17).

[3]. Виртуализация на уровне операционной системы. Режим доступа: https://ru.wikipedia.org/wiki/Виртуализация_на_уровне_операционной_системы (дата обращения 16.05.17).