МЕТОДОЛОГИЯ DEVOPS: ПРИНЦИПЫ, МЕТОДЫ И СРЕДСТВА РЕАЛИЗИЦИИ

Кудашов Евгений Борисович

магистрант, кафедра информатики и информационных технологий, Калужский государственный университет им. К.Э. Циолковского, РФ, г. Калуга

DEVOPS METHODOLOGY: PRINCIPLES, METHODS AND MEANS OF IMPLEMENTATION

Evgeny Kudashov

Master's degree student, Department of Informatics and Information Technologies, Kaluga State University K.E. Tsiolkovsky, Russia, Kaluga

Аннотация. В данной статье рассматриваются актуальные вопросы методологии DevOps: принципы (модель CALMS), основные методы и их применение в процессах разработки программного обеспечения. Анализируются популярные средства реализации DevOps для автоматизации процессов разработки, тестирования и развертывания программного обеспечения.

Abstract. This article discusses topical issues of DevOps methodology: principles (CALMS model), basic methods and their application in software development processes. Popular DevOps implementation tools for automating the development, testing and deployment of software are analyzed.

Ключевые слова: методология, DevOps, Agile, методы, средства реализации, инструменты, автоматизация, интеграция, совместная работа.

Keywords: methodology, DevOps, Agile, methods, implementation tools, tools, automation, integration, collaboration.

Введение. DevOps (development and operations) – это методология автоматизации технологических процессов программного обеспечения. Методология DevOps фокусирует свое внимание на стандартизации окружений разработки с целью быстрого переноса программного обеспечения через стадии жизненного цикла и быстрому релизу программного продукта. Данная методология предполагает активное взаимодействие разработчиков, тестировщиков и других специалистов по информационно-техническому обслуживанию, а также взаимную интеграцию их технологических процессов.

Краткая историческая справка. До появления методологии DevOps ведущие инженеры-разработчики находились изолированно от тестировщиков и специалистов по информационно-техническому обслуживанию [1, с. 30]. Такая форма взаимодействия разработчиков и специалистов существовала с 1970 года до 2001 год, когда рынке доминировала каскадная модель процесса разработки программного обеспечения – «Waterfall» (Водопад). Методика

Waterfall имела очевидные преимущества и значительные недостатки. К преимуществам следует отнести понятную и простую структуру процесса разработки и удобную отчетность (ресурсы, затраченное время, риски и финансы). К недостаткам этой методологии относили отсутствие гибкости процесса и невозможность внесения изменений в предыдущие стадии разработки. Кроме того, как у разработчиков, так и у заказчиков постоянно присутствовал повышенный риск финансовых затрат в сторону увеличения и некоторому снижению качества программного продукта.

С 2001 году методологию Waterfall постепенно вытесняет довольно гибкая методология разработки – «Agile». Данная методология включает ряд подходов и практик, основанных на четырех базовых принципах и двенадцати принципах «Манифеста гибкой разработки программного обеспечения» [3, с. 50]. К этому обычно добавляют Scrum, Kanban, Lean SD, Feature driven development (FDD) и другие.

Agile получила широкое распространение в организации работы небольших команд, которые создавали программный продукт короткими итерациями (до четырех недель). Каждая итерация выглядела как полноценный программный продукт, который состоял из всех типовых последовательностей от планирования и проектирования до тестирования и реализации. В конце итерации клиент получал рабочий программный продукт. Успех Agile был очевиден, но не так прочен. Методологию критиковали за отсутствие управления требованиями. Заказчик мог постоянно выдвигать новые требования, изменения или усовершенствования программы в конце каждой итерации, что иногда приводило к противоречиям в архитектуре созданного программного продукта, к переписыванию кода или к изменению стоимости конечного продукта [7].

В 2009 году широкое распространение получает методология «DevOps». Основная идея этой методологии состояла в том, чтобы привить ведущим разработчикам и их командам новую культуру разработки программного продукта [3, с. 52]. Однако создатели DevOps столкнулись со множеством проблем как со стороны команд, так и со стороны инструментов разработки. Что в конечном итоге привело к появлению следующих поколений инструментов таких как Puppet и Chef (система управления конфигурацией), Vagrant (программное обеспечение для создания и конфигурирования виртуальной среды разработки), Jenkins и другие.

Agile и DevOps, несмотря на очевидную схожесть с подходом к разработке программного обеспечения, имеют некоторые различия [3, с. 85]. Например, в методологии Agile разработка заканчивается сразу же после развертывания программного обеспечения. В то время как DevOps подключает операции, которые постоянно выполняются. Такими операциями могут быть мониторинг и модификация программного обеспечения.

В Agile ответственность за разработку, тестирование и развертывание несут разные специалисты. DevOps возлагает ответственность за указанные процессы на специально обученных специалистов - инженеров.

В Agile происходит поэтапное развертывание программного обеспечения после каждого спринта. DevOps выступает за «непрерывную доставку» (при любых изменениях в коде выполняется автоматическая сборка, тестирование и подготовка к окончательному выпуску). «Непрерывная доставка» является одним из основополагающих принципов разработки современных приложений.

Принципы DevOps. В 2010 году Дэймон Эдвардс и Джон Уиллис разработали модель CAMS, ключевые идеи которой стали основными принципами DevOps [7]. Согласно этой модели принципы DevOps выглядят следующим образом:

Culture, Collaboration (Культура). Предполагает устранение границ между функциональными подразделениями и их плодотворное взаимодействие на всех этапах разработки программного обеспечения.

Automation (Автоматизация). В DevOps автоматизируется практически все, что возможно. Автоматизация помогает убрать рутинную или избыточную работу, приводит к снижению количества сбоев или откатов системы. Благодаря автоматизации улучшается эффективность

рабочих процессов и производительность системы. Автоматизируется следующие разделы:

- тесты;
- сборка build (билда);
- развертывание инфраструктуры и приложений;
- мониторинг;
- создание бинарного пакета;
- уведомление об ошибках;
- развертывание баз данных.

Measurement (Измерения). В широком смысле DevOps говорит о том, что измерять необходимо практически все. Наблюдать же необходимо за действительно важными и значимыми метриками:

- производительность жизненного цикла разработки;
- сбор информации;
- анализ основных способов реагирования на обратную связь;
- ошибки (способы их исправления);
- помощь командам при работе над общей задачей.

Все указанные выше метрики должны регулярно собираться и своевременно анализироваться, чтобы в случае необходимости быстро разобраться в ситуации и устранить возникшие проблемы или сбои в работе системы [7].

Sharing (Обмен). В этой позиции говорится не только про обмен знаниями, но и о принятии решений. Ведущие специалисты DevOps и остальные участники процесса – маркетинг, логистика и так далее должны регулярно сотрудничать и предоставлять актуальную информацию друг другу. Сокрытие важной информации может привести к негативным и далеко идущим последствиям.

Через некоторое время Джез Хамбл, один из авторов книги «The DevOps Handbook» (Руководство по DevOps) добавил к четырем принципам DevOps еще одну позицию – «Lean» (Бережливость) и модель CAMS стала называться CALMS.

Lean (Бережливость). В этой позиции речь идет о выявлении и устранении потерь. Для этого процессы нужно визуализировать, сократить размер релизов и ограничить количество одновременно выполняемой работы. Это необходимо сделать, чтобы сократить объем незавершенной работы и снизить до минимума влияние многозадачности на метрики потока выполнения работы. Потому что многозадачность очень часто может скрывать действительно важные проблемы.

Методы DevOps. В DevOps на разных этапах разработки программного обеспечения используются определенные методы, которые помогают улучшить производственные процессы и делают их более программируемыми и гибкими. Существуют несколько основных методов:

- «Непрерывная интеграция» – это метод разработки программного обеспечения DevOps, при котором разработчики периодически объединяют изменения программного кода в центральном репозитории. После этого автоматически выполняется сборка, тестирование и запуск. Основная задача этого метода – быстро находить ошибки и исправлять их. Также этот метод призван улучшить качество программного обеспечения и сократить время на проверку и выпуск новых обновлений [2, с. 30].
- «Непрерывная доставка» – это практика разработки программного обеспечения, которая предполагает, что при любых изменениях в коде выполняется автоматическая сборка, тестирование и подготовка к выпуску [4, с. 40]. Этот метод является одним из основных принципов разработки современных приложений и расширяет метод непрерывной интеграции за счет того, что изменения в коде после сборки развертываются в тестовой или рабочей среде. Благодаря этому у разработчика всегда будет готовый к развертыванию экземпляр программного обеспечения, который

прошел стандартную процедуру тестирования.
- Архитектура микросервисов – это метод разработки приложений в виде набора небольших сервисов [5, с. 31]. Каждый такой сервис осуществляет работу в своем процессе и производит обмен данными с другими сервисами через программный интерфейс на базе HTTP (API). Микросервисы строятся вокруг практических требований бизнеса. За каждым из таких сервисов закрепляется одна конкретная задача.
- «Инфраструктура как код» – это практика, при которой выделение инфраструктуры и ее управление происходит с помощью кода и метода разработки программного обеспечения – «непрерывная интеграция». Облачная модель на основе API дает возможность разработчикам и системным администраторам работать с инфраструктурой на программном уровне вместо того, чтобы настраивать ресурсы вручную. Благодаря этому разработчики могут работать с инфраструктурой с помощью средств на основе кода также, как они работают с кодом приложений.
- «Мониторинг и ведение журналов» необходимы для того, чтобы инженер мог понять, как производительность приложения и его инфраструктура влияет на работу пользователя программного продукта. Также собранные данные помогают понять, как именно изменения и обновления действуют на пользователей продукта или найти источник сбоев или других проблем.
- «Обмен данными и совместная работа» является одним из ключевых и культурных аспектов DevOps. Взаимодействие между группами разработки, эксплуатации и другими подразделениями (маркетинг и логистика) помогает обмениваться важной информацией и позволяет всей компании сосредоточиться на конкретных целях и проектах.

Средства реализации DevOps. В DevOps для автоматизации процессов разработки, тестирования и развертывания применяются следующие технологии и популярные средства их реализации (инструменты):

- Планирование и совместная работа. Для этих задач подходит инструмент Jira Product Discovery. Основная цель этого инструмента – планирование и управление проектами. Идеально подходит для совместной работы в рамках проекта, несмотря на довольно сложный пользовательский интерфейс. Существуют другие инструменты с похожим функционалом – Mural и Miro.
- Управление версиями и совместная работа с кодом. Для этих задач подходят следующие инструменты:

Git – распределенная система управления версиями. Обеспечивает высокую скорость разработки, сохраняя при этом целостность данных, благодаря распределенным рабочим процессам [2, с. 446].

GitHub – крупнейший веб-сервис для хостинга проектов с возможностью контроля версий и совместной разработки, с понятным пользовательским интерфейсом и продвинутыми функциями. Идеально подходит для проектов с открытым исходным кодом.

Bitbucket – крупный веб-сервис для хостинга проектов и совместной разработки. Веб-сервис основан на системах контроля Mercurial и Git. Bitbucket известен своей интеграцией с Jira Product Discovery и Confluence.

GitLab – система управления репозиториями для Git с открытым исходным кодом. Присутствует интеграция с CI-системами. Идеально подходит для тех, кто хочет интегрировать CI/CD на собственном сервере.

- Контейнеризация. Для этих задач подходят следующие инструменты:

Docker – это программная платформа для быстрой разработки, тестирования и развертывания приложений. Инструмент упаковывает программное обеспечение в стандартные блоки – контейнеры. Каждый контейнер включает в себя все необходимое для работы приложений (библиотеки, системные инструменты, код) [4, с. 13].

Kubernetes – открытое программное обеспечение для автоматизации, развертывания, масштабирования и управления приложениями. Kubernetes – это IPI высокого уровня и он позволяет логически группировать контейнеры, регулировать нагрузку и определять для них размещение. Применяется в том случае, когда присутствует множество контейнеров и узлов, которыми нужно управлять [1, с. 40].

Nomad by HashiCorp – инструмент кластеризации контейнеров. Поддерживает интеграцию с основными инструментами контейнеризации (Docker и другие).

- Сборка и тестирование. Для этих задач подходят следующие инструменты:

Jenkins – инструмент автоматизации с открытым исходным кодом, используется для непрерывной доставки и интеграции кода с использованием контейнеров для сборки и развертывания. Довольно просто интегрируется с другими инструментами для тестирования и развертывания. Возможности Jenkins расширяются при использовании плагинов [8].

TeamCity – инструмент для непрерывной интеграции от JetBrains на основе сервера, который создает инструмент управления. Известен своим простым и понятным интерфейсом и удобными настройками. Позволяет делать несколько сборок одновременно. Тестирование может проводиться на разных платформах и разном программном окружении. Имеет встроенную интеграцию с GitHub.

Bamboo - инструмент непрерывной интеграции, который не требует дополнительной загрузки плагинов. Имеется возможность простой интеграции с Jira и Bitbucket. Инструмент Bamboo можно установить на сервер или использовать в облаке.

GitLab CI – инструмент для непрерывной интеграции при помощи образа Docker и Docker Hub.

- Развертывание и управление конфигурацией. Для этих задач подходят следующие инструменты:

Puppet – это инструмент с открытым исходным кодом для управления конфигурацией. Позволяет настраивать и управлять парком компьютеров и серверов. Присутствует возможность интеграции с Git.

Ansible – это инструмент с открытым исходным кодом для управления инфраструктурой и развертывания приложений. Позволяет автоматизировать как простые, так и многоуровневые приложения [6, с. 142].

Chef – это инструмент с открытым исходным кодом для управления инфраструктурой. Может применяться для сложных инфраструктур.

Saltstack – это инструмент с открытым исходным кодом. Написан на языке программирования Python и использует push-модель для выполнения команд по SHH.

- Облачные ресурсы и управление. Для этих задач подходит следующий инструмент:

Terraform – позволяет настраивать, разворачивать, изменять и управлять версиями инфраструктуры. Что позволяет достаточно просто вносить или отслеживать инфраструктуру. На рынке облачных ресурсов присутствуют похожие инструменты со сходным функционалом, например, VMware, DRS [8].

- Ведение журнала и мониторинг. Для этих задач подходят следующие инструменты:

Prometheus – осуществляет мониторинг любых систем – от серверов и баз данных до отдельных виртуальных машин. Подходит для решений с использованием контейнеров и распределенной архитектуры. На рынке мониторинга присутствуют похожие инструменты со сходным функционалом, например, Grafana, New Relic, Datadog, AlertManager, PagerDuty и другие.

Представленный список средств реализации (инструментов) является неполным, но достаточным для полноценной автоматизации процессов разработки программного обеспечения.

Заключение. Таким образом, в результате проведенного исследования мы пришли к следующим выводам: DevOps поддерживает основные принципы методологии Agile и принцип непрерывной доставки (Continuous delivery). В DevOps на разных этапах разработки программного обеспечения используются различные методы, которые призваны улучшить производственные процессы и сделать их более программируемыми и динамическими. Для автоматизации процессов разработки, тестирования и развертывания в DevOps применяются современные технологии и популярные средства их реализации, многие из которых имеют важное значение при настройке инструментальных средств DevOps и их применения в различных организациях.

Список литературы:

1. Арундел Д., Домингус Д. Kubernetes для DevOps: развертывание, запуск и масштабирование в облаке. – СПб.: Питер, 2020. – 384 с.

2. Вехен Д. Безопасный DevOps. Эффективная эксплуатация систем. – СПб.: Питер, 2020. – 432 с.

3. Ким Д., Дебуа П., Уиллис Д., Хамбл Д. Руководство по DevOps. – М.: Манн, 2018. – 512 с.

4. Маркелов А.А. Введение в технологии контейнеров и Kubernetes. – М.: ДМК Пресс, 2019. – 194 с.

5. Ньюмен С. Создание микросервисов. – СПб.: Питер, 2023. – 624 с.

6. Чоу Э. Python для сетевых инженеров. Автоматизация сети, программирование и DevOps. – СПб.: Питер, 2023. – 528 с.

7. Основы методологии DevOps. [Электронный ресурс]. – Режим доступа: https://proglib.io/p/osnovy-metodologii-devops-2021-02-20 (Дата обращения: 15.08.2023).

8. Инструменты DevOps. [Электронный ресурс]. – Режим доступа: https://efsol.ru/promo/devops-tools.html (Дата обращения: 16.08.2023).