Визитка

СЕРГЕЙ ГОЛОВАШОВ, ведущий инженер DevOps, компания Bell Integrator, sgolovashov@bellintegrator.com

Программные решения
для обеспечения цикла непрерывной разработки

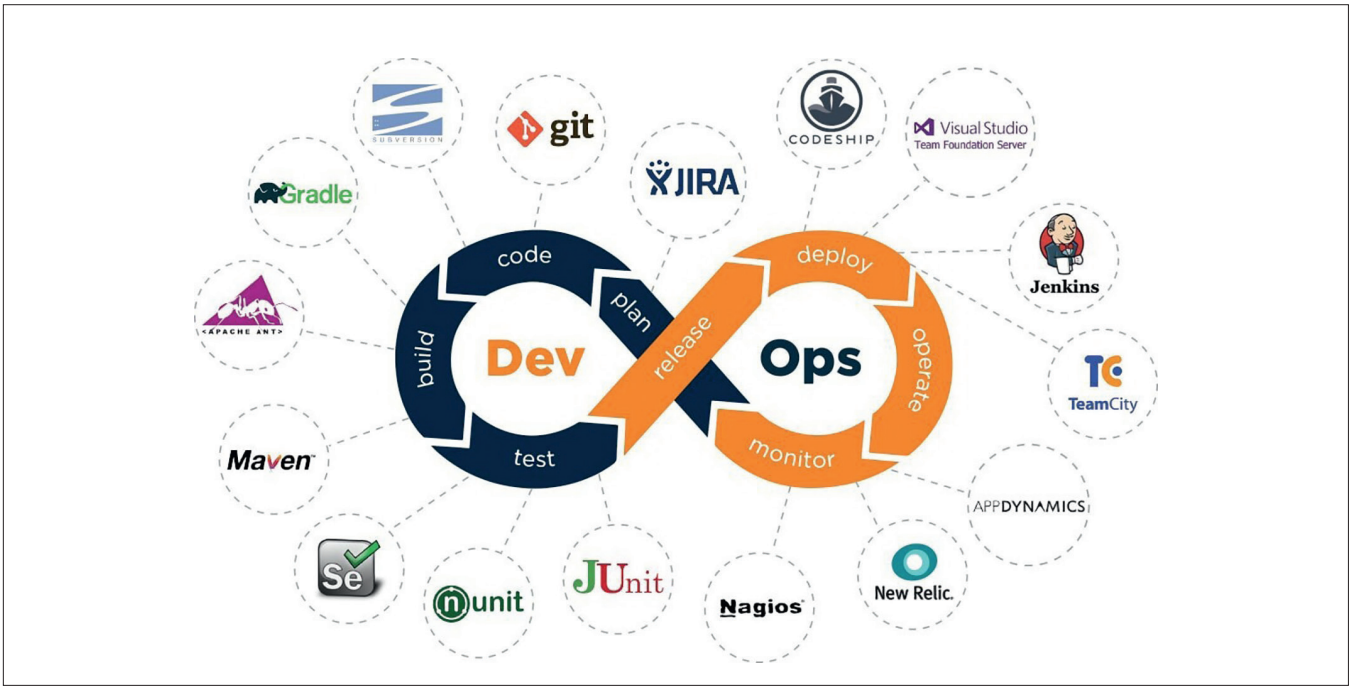
В данной статье расскажем о программном обеспечении, которое, при правильной настройке, должно обеспечить цикл непрерывной разработки

В цикле наших статей [1, 2] мы поставили себе задачу изучить более глобальную проблему, чем просто работу в консоли с целью установки какого-нибудь приложения. Поэтому приступаем к освещению инструментов DevOps и DevSecOps.

У DevOps есть разные определения. Он, как и Agile, включает в себя различные дисциплины. Но большинство согласятся со следующим определением: DevOps – это метод или жизненный цикл разработки ПО, главный принцип которого – создание культуры, где разработчики и другие сотрудники находятся «на одной волне», где ручной труд автоматизирован, каждый занимаются тем, что лучше всего умеет, поэтому растет частота поставок, повышается продуктивность работы, увеличивается гибкость.

И хотя одних только инструментов недостаточно для создания среды DevOps, без них не обойтись. Самым важным из них является непрерывная интеграция и непрерывная поставка (CI/CD).

В цепочке для каждого окружения есть разные этапы (например, DEV (разработка), INT (интеграция), TST



(тестирование), QA (контроль качества), UAT (приемочное тестирование пользователями), STG (подготовка), PROD (использование), ручные задачи автоматизированы, разработчики могут делать качественный код, делают его поставку и могут легко перестраиваться.

Поговорим об этих инструментах.

Шаг 1. Платформа CI/CD (pipeline)

Первым делом вам нужен инструмент CI/CD. Jenkins – это открытый инструмент CI/CD, написанный на Java, с лицензией MIT, с него началась популяризация движения DevOps, который де-факто стал стандартом для CI/CD.

А что такое Jenkins? Представьте, что у вас есть волшебный пульт управления для самых разных сервисов и инструментов. Сам по себе инструмент CI/CD, типа Jenkins, бесполезен, но с разными инструментами и сервисами он становится всемогущим.

Кроме Jenkins, есть множество других открытых инструментов. Их мы перечислим ниже в таблице. Выбирайте любой.

Название	Лицензия
Jenkins	CC и MIT
Travis CI	MIT
CruiseControl	BSD
Buildbot	GPL
Apache Gump	Apache 2.0
Cabie	GNU

Шаг 2. Управление версиями

Лучший (и, пожалуй, самый простой) способ проверить магию инструмента CI/CD – интегрировать его с инструментом контроля версий (source control management, SCM).

Зачем нужен контроль версий? Допустим, вы делаете приложение. Вы пишете его на Java, Python, C++, Go, Ruby, JavaScript или на любом другом языке, коих вагон и маленькая тележка. То, что вы пишете, называется исходным кодом.

> Сам по себе инструмент CI/CD, типа Jenkins, бесполезен, но с разными инструментами и сервисами он становится всемогущим

Поначалу, особенно если вы работаете в одиночку, можно сохранять всё в локальный каталог. Но когда проект разрастается и к нему присоединяется больше людей, вам нужен способ, чтобы делиться изменениями в коде, но при этом избегать конфликтов при слияниях изменений.

А еще вам нужно как-то восстанавливать предыдущие версии без использования резервных копий и применения метода copy-paste для файлов с кодом. И тут без SCM никуда. SCM сохраняет код в репозиториях, управляет его версиями и координирует его среди разработчиков.

Инструментов SCM немало, но стандартом де-факто заслуженно стал Git. Я советую использовать именно его, хотя есть и другие варианты, и лично мне был всегда очень симпатичен SVN.

Название	Лицензия
Git	GPLv2
Subversion	Apache 2.0
CVS	GNU
Vesta	LGPL
Mercurial	GPLv2

Шаг 3. Инструмент автоматизации сборки

Все идет как надо. Вы можете выгружать код и фиксировать изменения в системе контроля версий, а также пригласить друзей поработать с вами. Но приложения у вас пока нет. Чтобы это было веб-приложение, его нужно скомпилировать и поместить в пакет для поставки или запустить как исполняемый файл (интерпретируемый язык программирования вроде JavaScript или PHP компилировать не надо).

Применяйте инструмент автоматизации сборки. Какой бы инструмент вы ни выбрали, он будет собирать код в нужном формате и автоматизировать очистку, компиляцию, тестирование и поставку. Инструменты сборки бывают разные в зависимости от языка, но обычно используются следующие варианты с открытым кодом.

Название	Лицензия	Язык программирования
Maven	Apache 2.0	Java
Ant	Apache 2.0	Java
Gragle	Apache 2.0	Java
Bazel	Apache 2.0	Java
Make	GNU	-
Grunt	MIT	JavaScript
Gulp	MIT	JavaScript
Buildrm	Apache	Ruby
Rake	MIT	Ruby
A-A-P	GNU	Python
SCons	MIT	Python
BitBake	GPLv2	Python
Cake	MIT	C#
ASDF	MIT	LISP
Cabal	BSD	Haskell

Шаг 4. Сервер веб-приложений

Итак, у вас есть запакованный файл, который можно исполнять или выкатывать. Чтобы приложение действительно приносило пользу, у него должен быть какой-то сервис или интерфейс, но вам нужно где-то это всё разместить.

Веб-приложение можно разместить на сервере веб-приложений. Сервер приложений обеспечивает среду, где можно исполнить программную логику из пакета, выполнить отрисовку интерфейса и открыть веб-сервисы через сокет. Вам нужен HTTP-сервер и несколько других окружений (виртуальная машина, например), чтобы установить сервер приложений. Пока давайте представим, что вы разбираетесь со всем этим в процессе (хотя о контейнерах я расскажу ниже).

Есть несколько открытых серверов веб-приложений.

Название	Лицензия	Язык программирования
Tomcat	Apache 2.0	Java
Jetty	Apache 2.0	Java
WildFly	GNU Lasser Public	Java
GlassFish	GNU Less Public	Java
Django	BSD	Python
Tornado	Apache 2.0	Python
Gunicorn	MIT	Python
Python Paste	MIT	Python
Rails	MIT	Ruby
Node.js	MIT	JavaScript

У нас уже получилась почти рабочая цепочка DevOps.

Здесь можно остановиться, дальше вы справитесь сами, но стоит еще рассказать о качестве кода.

Шаг 5. Тестовое покрытие

Тестирование отнимает много времени и сил, но лучше сразу найти ошибки и улучшить код, чтобы порадовать конечных пользователей. Для этой цели есть много открытых инструментов, которые не только протестируют код, но еще и посоветуют, как его улучшить. Большинство инструментов CI/CD могут подключаться к этим инструментам и автоматизировать процесс.

> Тестирование отнимает много времени и сил, но лучше сразу найти ошибки и улучшить код, чтобы порадовать конечных пользователей

Тестирование разделено на две части: фреймворки тестирования, чтобы писать и выполнять тесты, и инструменты с подсказками по повышению качества кода.

Ниже представлены фреймворки тестирования.

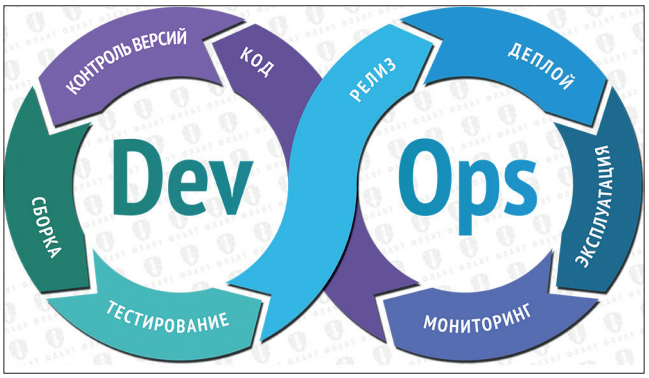
Название	Лицензия	Язык
JUnit	Eclipse Public License	Java
EasyMock	Apache	Java
Mockito	MIT	Java
PowerMock	Apache 2.0	Java
Pytest	MIT	Python
Hypothesis	Mozilla	Python
Tox	MIT	Python

И инструменты с подсказками по качеству.

Название	Лицензия	Язык
Cobertura	GNU	Java
CodeCover	Eclipse Public (EPL)	Java
Coverage.py	Apache 2.0	Python
Emma	Common Public License	Java
JaCoCo	Eclipse Public License	Java
Hypothesis	Mozilla	Python
Tox	MIT	Python
Jasmine	MIT	JavaScript
Karma	MIT	JavaScript
Mocha	MIT	JavaScript
Jest	MIT	JavaScript

Большинство этих инструментов и фреймворков написаны для Java, Python и JavaScript, потому что C++ и C# – проприетарные (хотя у GCC открытый исходный код).

Инструменты тестового покрытия мы применили, и теперь пайплайн DevOps приобрел те черты, которые необходимы для его успешной и безукоризненной работы.



• • •

Как итог вы получаете тот самый цикл замкнутой разработки, про который мы столько времени говорим. В следующей статье поговорим про мониторинг, метрики и графики, а также про нагрузочное тестирование. И в скором времени перейдем к обеспечению безопасности в разработке. EOF

[1] Головашов С. DevOps: Цифровая трансформация // Системный администратор. 2020. № 7–8. С. 79–81. – http://samag.ru/archive/article/4202

[2] Головашов С. Контрольные точки управления и понятие труда. Инструменты постановки и контроля выполнения задач. // Системный администратор. 2020. № 9. С. 45–49. – http://samag.ru/archive/article/4248

Ключевые слова: менеджмент организации, стандарты управления, стандарты информационной безопасности, техники SRE, DevOps, DevSecOps, инструменты защиты, компенсационные меры защиты, соблюдение требований регуляторов, 382-П, ГОСТ 57580, ISO 9000, ISO 27000