

DevOps

Ильсияр Гайнутдинов
Ilsiyar_gaynutdinov@ru.ibm.com
09.11.2020

Содержание

- [Что такое DevOps?](#)
- [Жизненный цикл DevOps](#)
- [Преимущества DevOps](#)
- [Методологии, принципы и стратегии DevOps](#)
- [Инструменты DevOps](#)
- [DevOps и IBM Cloud](#)
- [Рекомендации](#)

Что такое DevOps?

- распространенный подход к доставке программного обеспечения

Что такое DevOps?

- распространенный подход к доставке программного обеспечения
- Определенная роль

DevOps-инженер контролирует все этапы создания проекта: написание кода, тестирование и выпуск приложения. Он помогает отделам разработки и администрирования, синхронизирует их усилия и автоматизирует технические процессы.

DevOps

Developers

+

IT operations

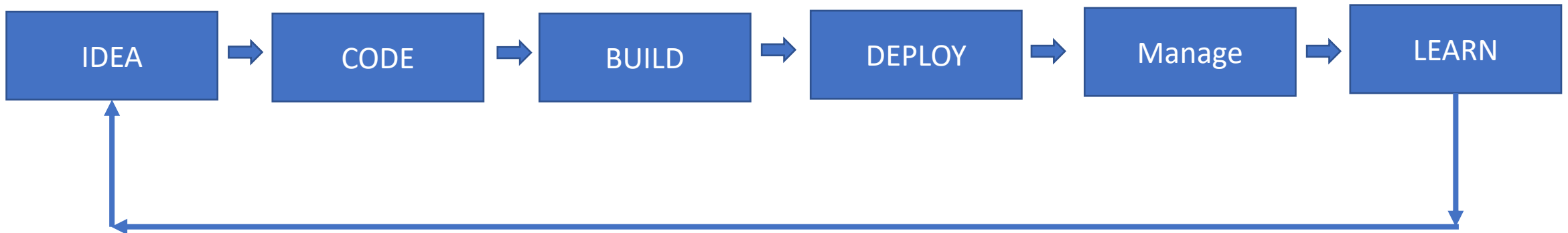
Features, changes

Defects

stability

availability

DevOps Lifecycle



Преимущества DevOps

- Быстрая доставка кода
- Более качественное программное обеспечение
- Более быстрое время выхода на рынок
- Улучшение сотрудничества между разработчиками и операторами
- Меньшее время потраченное на решения и исправления ошибок и уязвимостей

DevOps и Agile

- DevOps - это эволюция Agile. Модель Agile-разработки предписывает короткие концентрированные усилия на отдельных элементах продукта, которые обычно выполняются за две-четыре недели. В целом итеративный подход Agile преследует те же цели, что и DevOps: сотрудничество, обратная связь с клиентами и небольшие быстрые выпуски. В то время как Agile применяется к одной практике, DevOps применяется ко всему жизненному циклу проекта или приложения.
- Agile позволяет разработчикам доставлять свои функции каждые две недели в ответ на меняющиеся потребности бизнеса. DevOps фокусируется на операционной стороне жизненного цикла разработки программного обеспечения, сокращая передачу обслуживания от разработчика к операционным группам, сокращая время на тестирование и развертывание кода, а также уменьшая количество ошибок и время простоя операционных систем.

DevOps и SRE

- SRE - это подход к облачным операциям, который обеспечивает эффективную и надежную работу постоянно доставляемых облачных приложений за счет использования решений для разработки программного обеспечения и автоматизации.
- SRE будет включать в себя некоторые или все из следующих задач:
 - Устранение узких мест в производительности за счет рефакторинга сервисов в более масштабируемые единицы
 - Изоляция сбоев с помощью шаблонов проектирования для облачных вычислений
 - Создание модулей Runbook для быстрого восстановления
 - Автоматизация повседневных операционных процессов
- Самое интересное, что SRE - это практика [использования DevOps для защиты вашей инфраструктуры](#) . Это приводит к тому, что роль инженера по надежности сайта становится гибридной ролью DevOps - отчасти разработчика, отчасти системного администратора.

DevOps и SysOps (Сисадмин)

- DevOps сочетает в себе функции разработки и эксплуатации
- DevOps фокусируется на процессах более высокого уровня в компании и несет полную ответственность за продукты, в то время как

SysOps занимается настройкой и поддержкой отдельных компьютерных систем/техники, сети и программного обеспечения.

Типовые задачи сисадмина:

- подготовка и сохранение [резервных копий](#) данных, их периодическая проверка и уничтожение;
- установка и конфигурирование необходимых [обновлений](#) для [операционной системы](#) и используемых [программ](#);
- установка и конфигурирование нового [аппаратного](#) и [программного обеспечения](#);
- создание и поддержание в актуальном состоянии [пользовательских учётных записей](#);
- ответственность за информационную безопасность в компании;
- устранение неполадок в системе;
- планирование и проведение работ по расширению сетевой структуры предприятия;
- документирование всех произведенных действий.

Системных администраторов можно разделить на несколько категорий:

- *Администратор веб-сервера* — занимается установкой, настройкой и обслуживанием программного обеспечения веб-серверов. Как правило, работает в хостинговой компании.
- *Администратор баз данных* — специализируется на обслуживании баз данных.
- *Администратор сети* — занимается разработкой и обслуживанием сетей.
- *Системный администратор малой компании* (от 5 до 30 рабочих мест) — занимается поддержанием работоспособности небольшого парка компьютерной техники и обслуживанием сети. Выполняет все обязанности, связанные с компьютерами и коммуникациями, в том числе техническую поддержку пользователей.

DevSecOps

- комбинация DevOps и безопасности порождает безопасный DevOps или DevSecOps.

Чтобы реализовать безопасный DevOps , вы должны учитывать пять аспектов безопасности:

Пять аспектов безопасности:

- **Безопасная инженерия:** продукты разрабатываются с надежной защитой и соответствуют соответствующим стандартам безопасности.
- **Безопасное развертывание и операции:** облачная платформа и приложения безопасно настраиваются и развертываются, тестируются на уязвимости безопасности, тестируются исправленные ошибки(bugfix).
- **Разделение обязанностей:** пользователи получают доступ к тому, что требуется для выполнения их служебных обязанностей (т.е. Принцип наименьших привилегий).
- **Управление доступностью и непрерывностью бизнеса:** для 99,999-процентной доступности инфраструктуры, компонентов среды выполнения и компонентов управления.
- **Оценка и обучение** безопасности : вы поддерживаете функции и свойства безопасности в коде и сервисах по мере развития угроз и обнаружения новых уязвимостей.

Методологии, принципы и стратегии DevOps

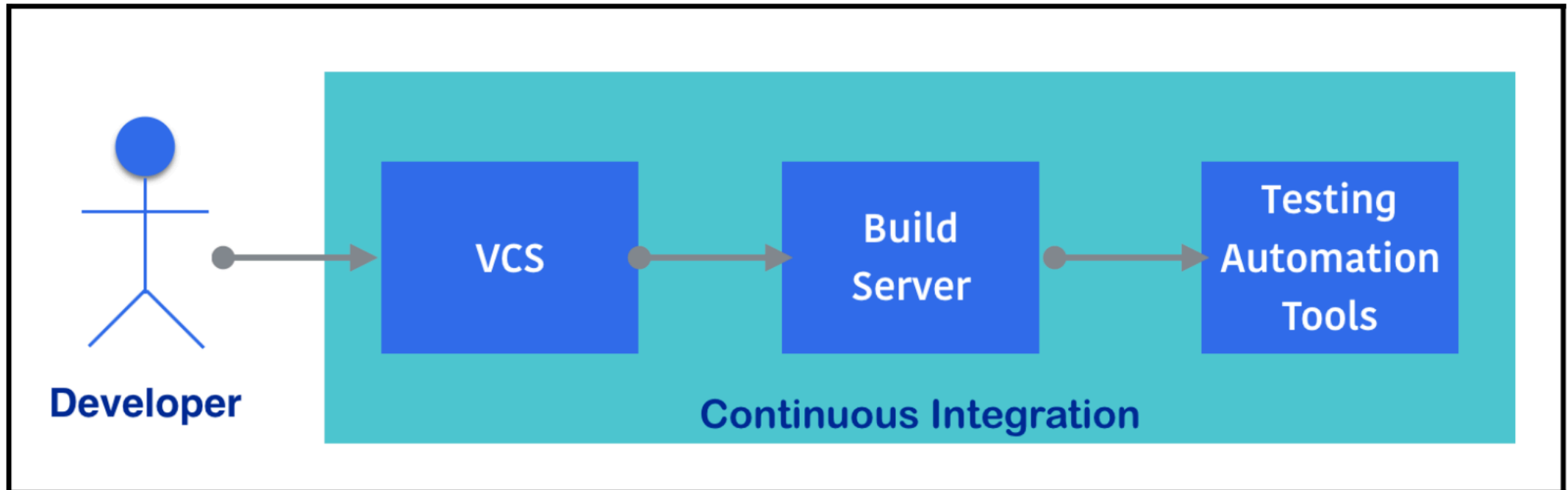
Методологии DevOps включают следующее:

- Непрерывная интеграция , при которой происходит кодирование, сборка, интеграция и тестирование.
- Непрерывная доставка , которая включает непрерывную интеграцию, но в основном ориентирована на выпуски продуктов.
- Непрерывное развертывание , направленное на автоматизацию выпусков проектов в кратчайшие сроки.

Что такое CI?

- Непрерывная интеграция - это процесс разработки программного обеспечения, при котором разработчики интегрируют свой код чаще, по крайней мере, один раз в день, чтобы выявлять проблемы интеграции раньше, когда их легче исправить.
- В целом, непрерывная интеграция помогает сократить затраты на разработку кода, что приводит к более качественному программному обеспечению и более предсказуемым поставкам ПО.

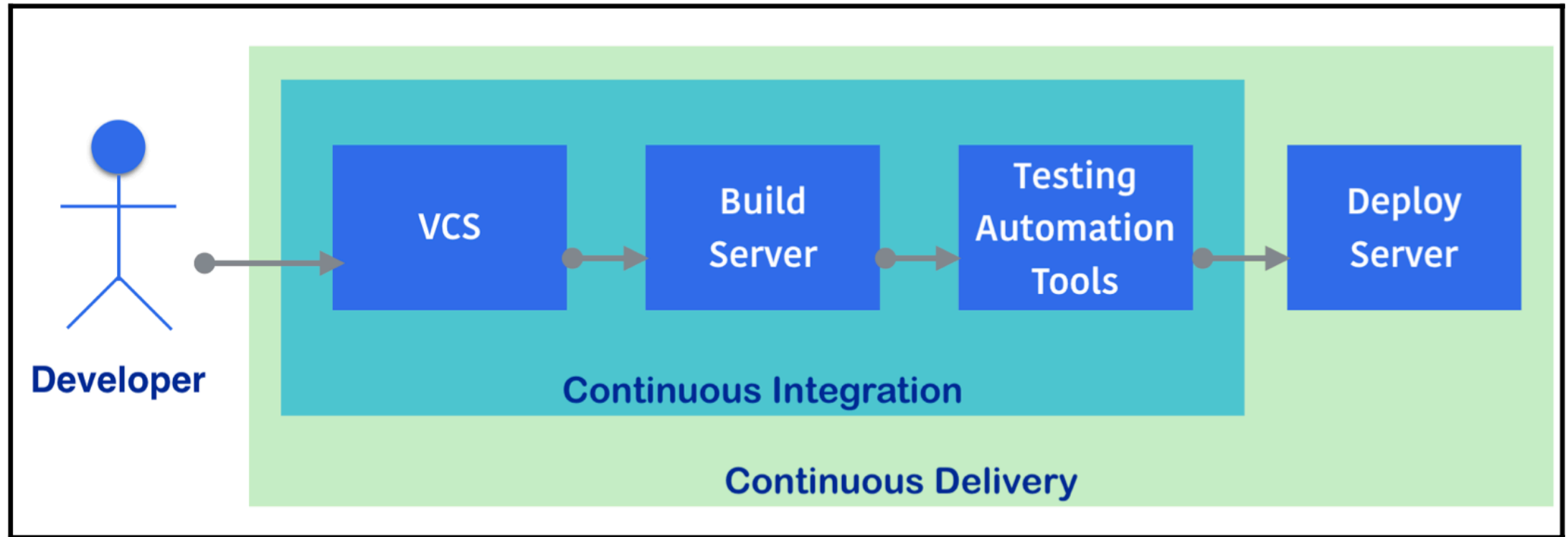
Continuous Integration



Что такое CD?

- Непрерывная доставка (CD) - это еще одна практика DevOps, которая фокусируется на доставке любых проверенных изменений кода - обновлений, исправлений ошибок и даже новых функций - пользователям как можно быстрее и безопаснее.
- Непрерывная доставка начинается там, где заканчивается непрерывная интеграция, автоматизируя доставку приложений в выбранные инфраструктурные среды. Он обеспечивает автоматическую отправку изменений кода в различные среды, такие как DEV, TEST и PROD(production).

Continuous Delivery



Best practices

- Поддерживайте единый исходный репозиторий: используйте VCS, чтобы отслеживать и контролировать все файлы для создания продукта. Это упрощает распространение и наглядность.
- Автоматизация сборки: это включает в себя компиляцию, компоновку и другие процессы, которые создают артефакты сборки. Само тестирование также следует автоматизировать.
- Используйте ежедневные фиксации основной ветки: заставляйте разработчиков фиксировать свои изменения в основном потоке разработки хотя бы раз в день. Каждый разработчик должен убедиться, что его рабочая копия соответствует основному потоку разработки.
- Тестируйте в клоне производственной среды: сделайте тестовую среду как можно более похожей на вашу конечную производственную среду.
- Автоматизация развертывания: внедрение нескольких сред (разработка, интеграция, производство) для запуска сборок и тестов.

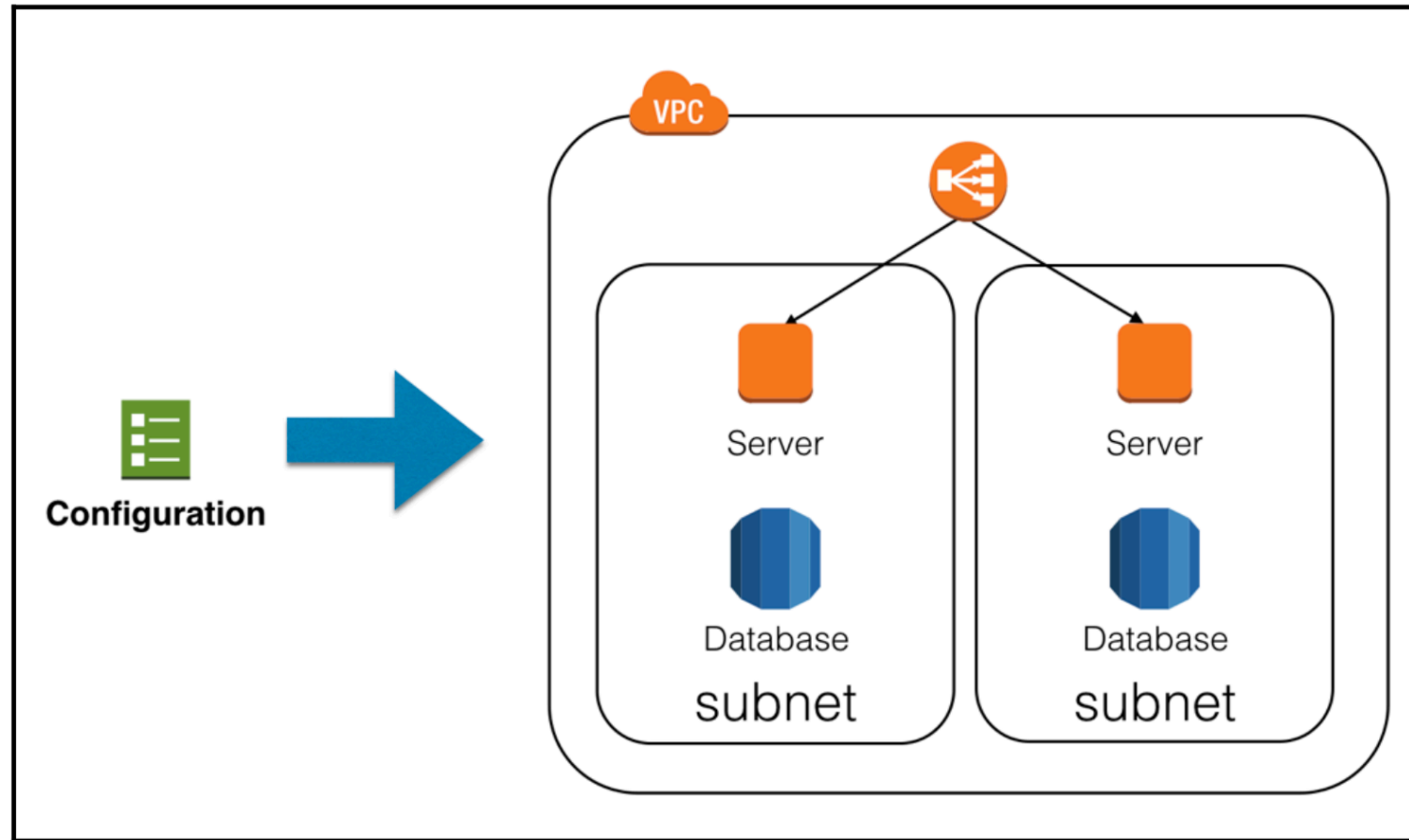
Best practices

- Сделайте каждое изменение выпускаемым: если вы используете непрерывную доставку, чтобы сделать каждое изменение выпускаемым, вы должны включить пользовательскую документацию, модули выполнения операций и информацию о том, что изменилось, для целей аудита.
- Используйте разработку на основе PIPELINEs: непрерывная доставка строится на непрерывной интеграции. Избегайте ветвей, которые задерживают интеграцию, насколько это возможно, чтобы каждое изменение создавалось, тестировалось и развертывалось вместе для максимально быстрой обратной связи.
- Доставка через автоматизированный конвейер: для успешной реализации непрерывной доставки вам необходим хорошо сконструированный автоматизированный конвейер доставки, чтобы гарантировать, что все выпуски вашего кода единообразно перемещаются в тестовую и производственную среды.
- Максимально автоматизируйте: при непрерывной доставке вы должны автоматизировать как можно больше процессов в жизненном цикле разработки программного обеспечения, чтобы создать хороший и надежный конвейер доставки не только для сборки и развертывания кода, но и для создания новых сред разработки.
- Стремитесь к отсутствию простоев: чтобы обеспечить доступность приложения при частой непрерывной доставке обновлений, когда вы запускаете новую функцию в производственную среду, вы должны сначала проверить ее перед развертыванием в общедоступном экземпляре запущенного приложения.

Что такое Continuous Deployment?

- Непрерывное развертывание - это стратегия разработки программного обеспечения, при которой изменения кода приложения автоматически переносятся в производственную среду. Эта автоматизация управляется серией заранее определенных тестов. После того, как новые обновления проходят эти тесты, система отправляет обновления непосредственно пользователям программного обеспечения.
- Непрерывное развертывание предлагает несколько преимуществ для организаций, которые хотят масштабировать свои приложения и ИТ-портфель. Во-первых, он ускоряет вывод продукта на рынок, устраняя разрыв между кодированием и ценностью для потребителя - обычно дни, недели или даже месяцы.
- Для этого необходимо автоматизировать регрессионные тесты, что исключает дорогостоящее ручное регрессионное тестирование.

Infrastructure as code



Infrastructure as code

← → ↻ terraform.io ☆ 📧 ⚙️ 👤 🔴

Deliver infrastructure as code with Terraform

Write declarative configuration files

- ✓ Collaborate and share configurations
- ✓ Evolve and version your infrastructure
- ✓ Automate provisioning

Define infrastructure as code to manage the full lifecycle — create new resources, manage existing ones, and destroy those no longer needed.

```
Terraform will perform the following actions:

# kubernetes_pod.example will be updated in-place
~ resource "kubernetes_pod" "test" {
  id = "default/terraform-test"

  metadata {
    generation = 0
    labels = {
      "app" = "MyApp"
    }
  }
  name = "terraform-test"
  namespace = "default"
  resource_version = "650"
  self_link = "/api/v1/namespaces/default/pods/terraform-test"
```

```
variable "base_network_cidr" {
  default = "10.0.0.0/8"
}

resource "google_compute_network" "example" {
  name = "test-network"
  auto_create_subnetworks = false
}

resource "google_compute_subnetwork" "example" {
  count = 4

  name = "test-subnetwork"
  ip_cidr_range = cidrsubnet(var.base_network_cidr, 4, count.index)
  region = "us-central1"
  network = google_compute_network.custom-test.self_link
}
```

Plan and predict changes

- ✓ Clearly mapped resource dependencies
- ✓ Separation of plan and apply
- ✓ Consistent, repeatable workflow

Terraform provides an elegant user experience for

Инструменты DevOps

Инструменты DevOps охватывают ряд процессов в рамках жизненного цикла разработки программного обеспечения:

- **Определение и планирование**, в котором основное внимание уделяется планированию рабочих процессов DevOps для итераций, управлению выпусками и отслеживанию проблем. Известные инструменты или поставщики инструментов в этой области включают Atlassian, CA Technologies, IBM, iRise и Jama Software.
- **Код, сборка и настройка**, в котором основное внимание уделяется разработке и обзору кода, управлению исходным кодом и слиянию кода. Известные инструменты / поставщики инструментов включают BitBucket, Electric Cloud, GitLab, GitHub и IBM.
- **Тестирование**, которое проверяет, что качество выпуска программного обеспечения и кода поддерживается на протяжении всего процесса разработки и что самое высокое качество развертывается в производственной среде. Известные инструменты / поставщики инструментов включают Delphix, FlawCheck, HP, IBM, Microsoft, Parasoft, SonarSource, Skytap и ThoughtWorks.
- **Упаковка и подготовка к выпуску**, которые относятся к действиям, выполняемым после того, как релиз будет готов к развертыванию; это также называется постановкой или подготовкой. Известные инструменты / поставщики инструментов включают IBM, ProGet Inedo, Artifactory Jfrog, репозиторий Sonatype Nexus.
- **Выпуск, развертывание и оркестровка**, который представляет собой процесс фактического выпуска программного обеспечения и обычно включает в себя управление изменениями, утверждения выпуска, автоматизацию выпуска, оркестровку расписания, подготовку и развертывание в производственной среде. Инструменты / поставщики инструментов в этой области включают Automatic, Clarive, BMC, IBM, Flexagon, VMware и XebiaLabs.
- **Непрерывное управление и конфигурация** включает непрерывную автоматизацию конфигурации, управление конфигурацией и инфраструктуру как код. Известные инструменты / поставщики инструментов включают Ansible, Chef, IBM, Puppet Labs, Otter и Salt.
- **Мониторинг** сообщает о производительности приложений и помогает выявлять проблемы, влияющие на работу пользователей. Инструменты / поставщики инструментов включают Big Panda, IBM, New Relic, PlumbR и Wireshark.

Набор инструментов DevOps

- [IBM Architecture Room Live](https://www.ibm.com/products/architecture-room-live) используется для проектирования и передачи новых архитектур. Это инструмент (доступный в браузере) для проектирования и создания архитектуры ПО, позволяющий в реальном времени многопользовательскую и многосайтовую совместную работу через виртуальную «доску».

<https://www.ibm.com/products/architecture-room-live>

Набор инструментов DevOps

- [IBM Rational Test Workbench](https://www.ibm.com/products/rational-test-workbench) используется для виртуализации зависимостей тестовой среды, когда команде необходимо начать тестирование. Этот тул позволяет создавать, выполнять и составлять отчеты о качестве интеграции, функциональных возможностях и производительности, а также тестировать сквозные бизнес-сценарии и технические сценарии.

<https://www.ibm.com/products/rational-test-workbench>

Набор инструментов DevOps

- [IBM UrbanCode Deploy](https://www.ibm.com/cloud/urbancode/deploy) используется для автоматизации развертывания программного обеспечения в различных средах. Это снижает риск управления несколькими конфигурациями, интеграциями и версиями приложений в нескольких экосистемах.

<https://www.ibm.com/cloud/urbancode/deploy>

Набор инструментов DevOps

- [IBM UrbanCode Velocity](https://www.ibm.com/cloud/urbancode/velocity) используется для организации общей поставки всего программного решения. Он дает представление о том, как ценность проходит через каждый конвейер доставки, и показывает, где узкие места нарушают поток.

<https://www.ibm.com/cloud/urbancode/velocity>

Набор инструментов DevOps

- Система контроля версий
Git/GitHub/GitLab

<https://git-scm.com>

GitHub (<https://github.com>)

The screenshot shows the GitHub interface for the repository **betamaniac / tjbot**, which is forked from **ibmtjbot/tjbot**. The repository has 1 pull request, 0 stars, and 282 forks. The **Code** tab is selected, showing the **master** branch with 2 branches and 6 tags. A status bar indicates the current branch is 5 commits ahead and 14 commits behind the upstream **ibmtjbot:master**. The commit history table lists recent changes, including translations to Russian and updates to the README and wiring diagram. The right sidebar provides information about the project, including a link to **ibmtjbot.github.io**, the Apache-2.0 license, and a link to create a new release.

Search or jump to... Pull requests Issues Marketplace Explore

betamaniac / tjbot
forked from **ibmtjbot/tjbot**

Unwatch 1 Star 0 Fork 282

Code Pull requests Actions Projects Wiki Security Insights Settings

master 2 branches 6 tags

Go to file Add file Code

This branch is 5 commits ahead, 14 commits behind **ibmtjbot:master**. Pull request Compare

betamaniac translated to russian		6c64653 on 13 May 2019 292 commits
bootstrap	fixed path for runTests script	2 years ago
featured	Update README.md	2 years ago
images	updated the wiring diagram	4 years ago
recipes	translated to russian	2 years ago
CONDUCT.md	cleaning up permission bits	4 years ago
CONTRIBUTING.md	Broken Links	3 years ago
DCO1.1.txt	first push .. go love	4 years ago
LICENSE	Update LICENSE	2 years ago

About

IBM TJBot

ibmtjbot.github.io

Readme

Apache-2.0 License

Releases

6 tags

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Bitbucket (<https://bitbucket.org>)

The screenshot shows the Bitbucket web interface for a repository named "Tutorial". The left sidebar contains navigation links: Source (selected), Commits, Branches, Pull requests, Pipelines, Jira issues, Downloads, and Repository settings. The main content area shows the repository path "betamaniac betamaniac / Untitled project" and the repository name "Tutorial". Below this, there is a message: "Here's where you'll find this repository's source files. To give your users an idea of what they'll find here, [add a description to your repository](#)." A dropdown menu shows the current branch as "master". Below the branch selector, there is a table of files and commits.

Name	Size	Last commit	Message
README.md	16.42 KB	2015-02-12	README.md edited online with Bitbucket
README.text	16.8 KB	2014-08-29	adding all the content files for the first time
sample.html	212 B	2014-08-29	adding all the content files for the first time
sandbucket.jpg	1.27 KB	2014-08-29	adding all the content files for the first time

Below the table, there is a section titled "README.md" with the following content:

Welcome to your tutorial repository!

Learn how to use Git and Bitbucket with either SourceTree, one of the best Git clients available, or using Git from the command line. Whichever you choose you will learn how set up Git, clone this repository locally. Then learn how to make and commit a change locally and push that change back to Bitbucket.

Start here

Choose either SourceTree, Atlassian's Git client, or the command line to learn source control using Bitbucket and Git.

Use [SourceTree Atlassian's Git client](#) for Windows and Mac

Gitlab (<https://about.gitlab.com>)

The screenshot displays the GitLab web interface for a project named 'frontend-v2'. The top navigation bar includes links for Projects, Groups, Activity, Milestones, Snippets, and a search bar. The left sidebar contains a list of project features: Project, Details, Activity, Releases, Cycle Analytics, Repository, Issues (0), Jira, Merge Requests (0), CI / CD, Operations, Wiki, Snippets, and Settings. The main content area shows the project details for 'frontend-v2' (Project ID: 492). It includes a 'Clone' button, a 'Fork' button, and a 'Star' button. Below this, it shows the project's license (No license), commit count (1,325), branch count (12), tag count (0), and file count (10.1 MB). A merge request summary is displayed, showing a merge of branch 'bugfix/SC-3016' into 'master' by Sergey Gubernskiy, with a commit hash of 4ff170e9. Below the merge request, there are buttons for 'README', 'CI/CD configuration', and 'Add Kubernetes cluster'. A table lists the project's files and their last commit details.

Name	Last commit	Last update
e2e	initial commit	1 year ago
k8s	MR changes 2	8 months ago
src	SC-3016 change initial status for isIndustrial...	7 months ago
.editorconfig	initial commit	1 year ago
.gitignore	initial commit	1 year ago
.gitlab-ci.yml	restore property	9 months ago
.graphqlconfig	add Haifa environment	8 months ago
.npmrc	SC-2866 install npm package from nexus	8 months ago
.prettiignore	SC-1091 Add field validation	1 year ago

SonarQube (<https://www.sonarqube.org>)

The screenshot displays the SonarQube web interface for a project named 'Your Project' (branch 'master'). The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. A search bar and a user profile icon are also present. The main content area is divided into two sections: 'QUALITY GATE STATUS' and 'MEASURES'.

QUALITY GATE STATUS

Passed
All conditions passed.

MEASURES

The 'MEASURES' section is divided into two tabs: 'New Code' and 'Overall Code'. The 'Overall Code' tab is active, showing the following metrics:

- Bugs:** 1 (Reliability: C)
- Vulnerabilities:** 0 (Security: A)
- Security Hotspots:** 6 (Security Review: E, 0.0% Reviewed)
- Debt:** 4d 4h
- Code Smells:** 259 (Maintainability: A)

At the bottom, there are two summary cards:

- Coverage:** 0.0% (Coverage on 12k Lines to cover)
- Unit Tests:** 659
- Duplications:** 1.4% (Duplications on 30k Lines)
- Duplicated Blocks:** 32

Jenkins (<https://jenkins.io>)

The screenshot shows the Jenkins web interface for a pipeline named 'simple-java-maven-app 2'. The top navigation bar includes links for Pipeline, Changes, Tests, Artifacts, and a Logout button. Below the navigation bar, the pipeline status is shown as '25s' and 'a few seconds ago'. The main area displays a pipeline graph with four stages: Start, Build, Test, and End. The Build and Test stages are marked with green checkmarks, indicating successful completion. Below the graph, the 'Steps Test' section shows the execution of a 'mvn test' shell script. The output log includes various Maven build messages, such as 'Scanning for projects...', 'Building my-app 1.0-SNAPSHOT', and 'maven-resources-plugin:2.6:resources (default-resources) @ my-app ---'. The log also shows warnings about platform encoding and skip non-existing resourceDirectory.

✓ simple-java-maven-app 2

Pipeline Changes Tests Artifacts Logout

Branch: — 25s Changes by gilesgas
Commit: — a few seconds ago Started by user Alex

Start Build Test End

Steps Test

✓ mvn test — Shell Script 9s

```
1 [simple-java-maven-app] Running shell script
2 + mvn test
3 [INFO] Scanning for projects...
4 [INFO]
5 [INFO] -----
6 [INFO] Building my-app 1.0-SNAPSHOT
7 [INFO] -----
8 [INFO]
9 [INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ my-app ---
10 [WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
11 [INFO] skip non existing resourceDirectory /var/jenkins_home/workspace/simple-java-maven-app/src/main/resources
12 [INFO]
13 [INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ my-app ---
14 [INFO] Nothing to compile - all classes are up to date
15 [INFO]
16 [INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ my-app ---
17 [WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
18 [INFO] skip non existing resourceDirectory /var/jenkins_home/workspace/simple-java-maven-app/src/test/resources
```

Jenkins (<https://jenkins.io>)

```
pipeline {
  agent {
    docker {
      image 'maven:3-alpine'
      args '-v /root/.m2:/root/.m2'
    }
  }
  stages {
    stage('Build') {
      steps {
        sh 'mvn -B -DskipTests clean package'
      }
    }
    stage('Test') { ❶
      steps {
        sh 'mvn test' ❷
      }
      post {
        always {
          junit 'target/surefire-reports/*.xml' ❸
        }
      }
    }
  }
}
```


Travis CI (<https://travis-ci.org>)

Travis CI Docs

Search all repositories

My Repositories

Running (0/0)

+

✓ hrl-eco/ECO_WebBackend # 3858

⌚ Duration: 3 min 57 sec

📅 Finished: about 2 hours ago

✓ hrl-elal/ELAL_Daemon # 1828

⌚ Duration: 2 min 45 sec

📅 Finished: about 2 hours ago

✓ hrl-eco/ECO_Visual # 1763

⌚ Duration: 1 min 55 sec

📅 Finished: about 2 hours ago

✓ hrl-aeroflot/aeroflot # 1508

⌚ Duration: 4 min 10 sec

📅 Finished: about 2 hours ago

✓ hrl-elal/ELAL_DataETL # 1227


⌚ Duration: 1 min 46 sec

📅 Finished: about 2 hours ago

✓ hrl-eco/CPRO_Optimization # 1065

⌚ Duration: 1 min 30 sec

📅 Finished: about 2 hours ago

hrl-eco / ECO_WebBackend  build passing

Current

Branches

Build History

Pull Requests

More options

⋮

✓ master triggered by hrl-eco/ECO_Visual update

🔗 Commit b096d54

🔗 Branch master

🔄 HAIFA\lipets authored and committed

🟢 #3858 passed

⌚ Ran for 3 min 57 sec

📅 about 2 hours ago

🔄 Restart build

Job log

View config

Remove log

Raw log

1 Worker information

6 Build system information

156

157 Warning: apt-key output should not be parsed (stdout is not a terminal)

158

159 Installing openjdk8

161 Installing SSH key from: repository settings

163 \$ git clone --depth=50 --branch=master git@github.ibm.com:hrl-eco/ECO_WebBackend.git hrl-eco/ECO_WebBackend

174

175 Setting environment variables from repository settings

176 \$ export TRAVIS_ACCESS_TOKEN=[secure]

177 \$ export GITHUB_ACCESS_TOKEN=[secure]

178 \$ export TRAVIS_SIGNED_JWT=[secure]

179

180 Setting environment variables from .travis.yml

181 \$ export DB2_PATH="Tools/jars/db2jcc4.jar"

182 \$ export DB2_VERSION="10.1"

183 \$ export REPO_NAME="ECO_WebBackend"


184 \$ export META_OWNER="hrl-aeroflot"

185 \$ export META_REPO="aeroflot"




Travis CI (<https://travis-ci.org>)

[Code](#) [Issues 0](#) [Pull requests 0](#) [Wiki](#) [Insights](#) [Settings](#)

Branch: ECOD-24 [eco-ui / .travis.yml](#) [Find file](#) [Copy path](#)

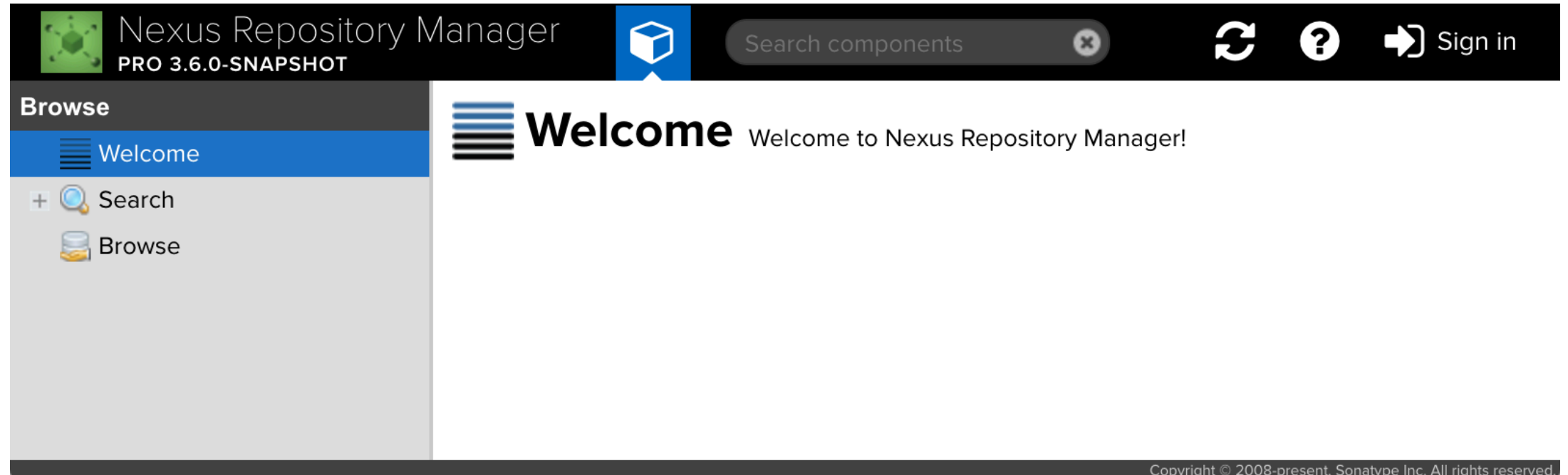
 **ilsiyar-gaynutdinov** update to login 0e34752 on 8 Sep

1 contributor

12 lines (12 sloc) | 533 Bytes [Raw](#) [Blame](#) [History](#)   

```
1 language: java
2 services:
3   - docker
4 before_install:
5   - cp .travis.settings.xml $HOME/.m2/settings.xml
6   - echo "$MVN_PASSWORD" | docker login sys-eco-team-docker-local.artifactory.swg-devops.com -u "$MVN_USER" --password-stdin
7 script:
8   - mvn clean package
9 after_success:
10  - docker build -t eco-ui:version .
11  - docker tag eco-ui:version sys-eco-team-docker-local.artifactory.swg-devops.com/sys-eco-team-docker-local/eco-ui:version
12  - docker push sys-eco-team-docker-local.artifactory.swg-devops.com/sys-eco-team-docker-local/eco-ui:version
```

Nexus (<https://my.sonatype.com/>)

















Nexus



Browse

Browse assets and components

Filter 

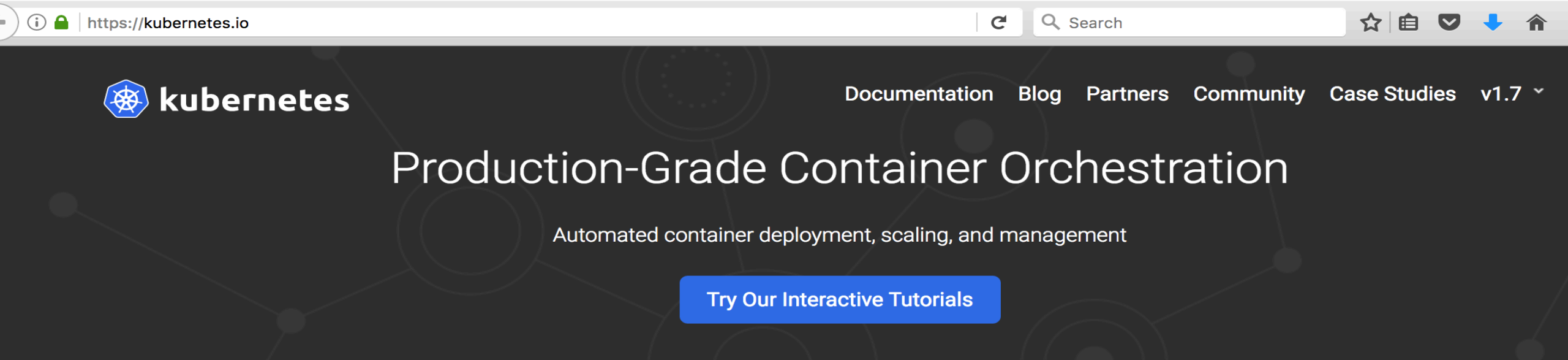
	Name ↑	Type	Format	Status	URL	Health check	
	maven-central	proxy	maven2	Online - Remote...	 copy	Analyze	>
	maven-public	group	maven2	Online	 copy	⊘	>
	maven-releases	hosted	maven2	Online	 copy	⊘	>
	maven-snapshots	hosted	maven2	Online	 copy	⊘	>
	nuget-group	group	nuget	Online	 copy	⊘	>
	nuget-hosted	hosted	nuget	Online	 copy	⊘	>
	nuget.org-proxy	proxy	nuget	Online - Ready t...	 copy	Analyze	>

Configuration management

- Puppet (<https://puppet.com>)
- Chef (<https://www.chef.io>)
- Ansible (<https://www.ansible.com>)



<https://Kubernetes.io>



Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon [15 years of experience of running production workloads at Google](#), combined with best-of-breed ideas and practices from the community.




Что такое Kubernetes?

- ✓ Это open-source система для автоматического деплоя, масштабирования и управления контейнеризированными приложениями
- ✓ k8s группирует контейнеры, которые составляют приложение, в логические блоки для легкого управления и развертывания
- ✓ Kubernetes опирается на 15-летний опыт работы с рабочими нагрузками в Google в сочетании с лучшими в своем классе идеями и практикой сообщества.

Что такое Kubernetes?

- ✓ Предоставляет все необходимое: orchestration, service discovery, load balancing и многое другое
- ✓ Изначально был сделан для GCE (google cloud engine)
- ✓ Предлагает свою концепцию: Pods, services, replication controller

Today ~71 500 stars





Pull requests


Issues


Marketplace

Explore







 **kubernetes / kubernetes**

Watch

3.3k


Star


71.5k


Fork


25.9k


<> Code


 Issues 2k

 Pull requests 817

 Actions

 Projects 6

 Security

 Insights

master


42 branches

717 tags

Go to file











Add file

Code

 **k8s-ci-robot** Merge pull request #92312 from Sh4d1/kep_1860


ef16faf 5 hours ago

95,550 commits

 .github	Update triage/support label references to kind/support	last month
 CHANGELOG	Merge pull request #96207 from qingsenLi/201104-changelog	3 days ago
 LICENSES	Update cobra dependency to v1.1.1	7 days ago
 api	Update generated	2 days ago
 build	pause image: Disable DiagTrack service on Windows image	5 days ago
 cluster	Merge pull request #95863 from RaunakShah/snap_e2e	4 days ago
 cmd	Merge pull request #96306 from SataQiu/small-fix-20201106	2 days ago
 docs	Add sigs for root folders	3 months ago
 hack	Merge pull request #95872 from 22dm/kube-proxy-comment-fix	9 hours ago
 logs	Merge pull request #92975 from sur1254/master	2 months ago

About

Production-Grade Container Scheduling and Management

 **kubernetes.io**

kubernetes

go


cncf

containers

Readme

Apache-2.0 License

Releases 717

 **v1.18.10** Latest

25 days ago

+ 716 releases

Master Components

Основные компоненты обеспечивают управление кластера. Основные компоненты принимают глобальные решения о кластере (например, планирование), а также обнаруживают и реагируют на события кластера (запуск нового модуля, когда поле «replicas» контроллера репликации неудовлетворено).

Основные компоненты могут быть запущены на любом узле кластера. Но как правило запускаются на одной машине.

Master Components

kube-apiserver

предоставляет API Kubernetes. Это интерфейс для панели управления Kubernetes. Он предназначен для масштабирования по горизонтали, т.е. масштабируется путем развертывания большего количества экземпляров.

Etcd

используется как резервное хранилище Kubernetes. Здесь хранятся все данные кластера.

kube-scheduler

наблюдает за недавно созданными pods, которые не имеют назначенного узла, и выбирает узел для их запуска.

kube-controller-manager

запускает контроллеры, которые являются фоновыми потоками, которые обрабатывают обычные задачи в кластере. Логически каждый контроллер является отдельным процессом, но для уменьшения сложности все они скомпилированы в один бинарный файл и выполняются в одном процессе.

Эти контроллеры включают:

- Node Controller
- Route Controller
- Service Controller
- Volume Controller

Master Components

DNS

Кластер DNS - это DNS-сервер, в дополнение к другим DNS-серверам в вашей среде, который обслуживает DNS-записи для служб Kubernetes.

Web UI (Dashboard)

web-based UI for Kubernetes clusters. Он позволяет пользователям управлять и устранять неполадки приложений, запущенных в кластере, а также самого кластера.

Container Resource Monitoring

записывает общие метрики временных рядов о контейнерах в центральной базе данных и предоставляет пользовательский UI

Kubernetes Components

Node components

Компоненты узла запускаются на каждом узле, поддерживая запуск модулей и обеспечивая среду выполнения Kubernetes

Kubelet

является основным агентом узла. Он следит за контейнерами, которые были назначены его узлу (либо `apiserver`, либо через локальный файл конфигурации), и:

- Монтирует требуемые volumes
- Загружает секреты Pods
- Запускает контейнеры Pods используя docker
- Периодически выполняет любые запросы liveness probes
- Сообщает о статусе Pods обратно в остальную часть системы, создавая при необходимости зеркальный Pod.
- Сообщает о статусе узла Node обратно в остальную часть системы.

Node components

kube-proxy

позволяет использовать абстракцию сервиса Kubernetes, поддерживая сетевые правила на хосте и выполняя переадресацию соединений.

Docker

используется для запуска контейнеров.

Supervisord

это легкий монитор процесса и система управления, которые можно использовать для поддержания работы kubelet и докеров.

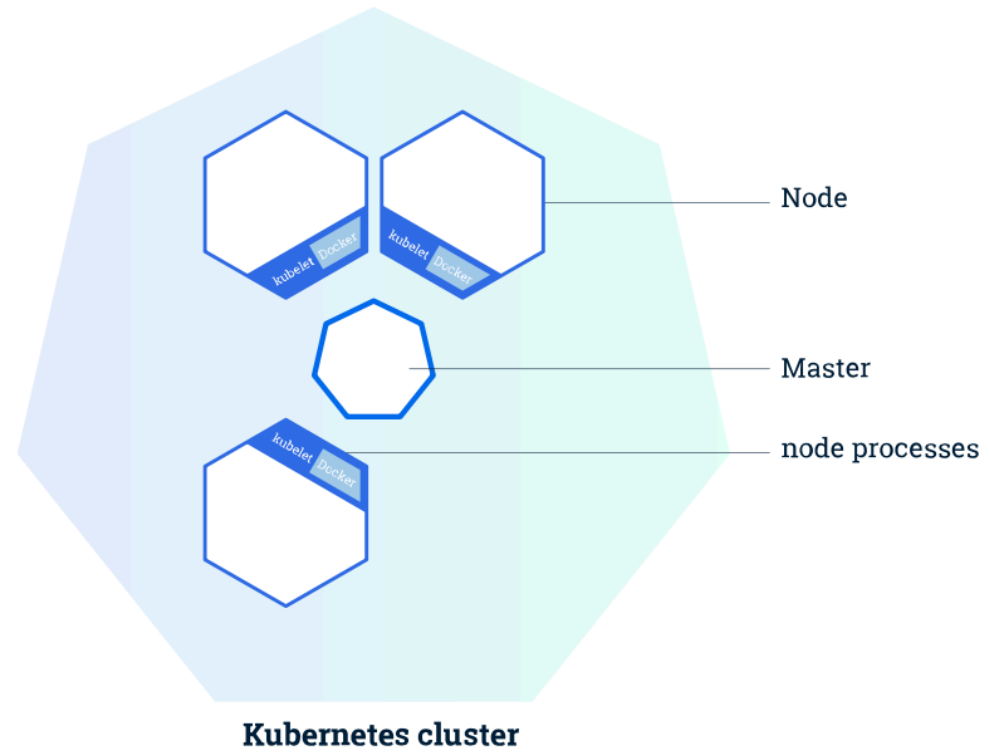
fluentd

это демон, который помогает обеспечить протоколирование на уровне кластера.

Kubernetes Master and Minions

- ✓ Мастер контролирует и управляет вашими k8s нодами
- ✓ K8s Minions исполняют задачи от мастера
- ✓ Общение через etcd между всеми компонентами

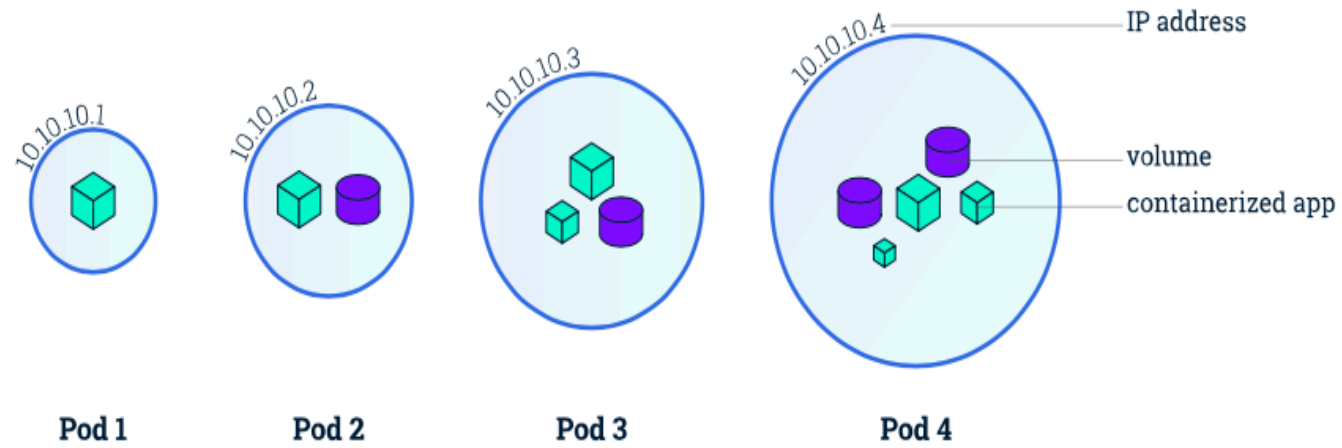
Cluster Diagram



Pod

- ✓ Состоит из сгруппированных вместе Docker контейнеров
- ✓ Наиболее базовый юнит в k8s - вам более не требуется думать на уровне контейнера в k8s deployment
 - ✓ Все контейнеры которые которые включаются в один pod гарантировано будут запланированы и развернуты вместе на одном minion
 - ✓ Значительное изменение в философии дизайна. Некоторые считают что абстракция на уровне контейнеров уже не достаточно для распределенных приложений.

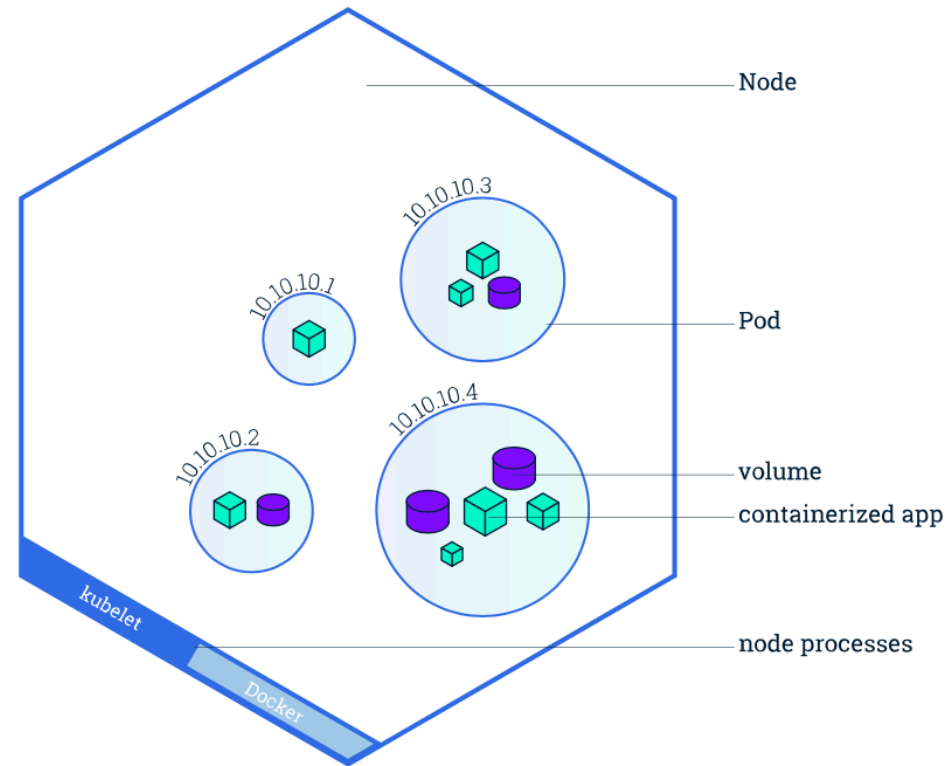
Pods overview



Replication controller

- ✓ Позволяет быть вам уверенным что определенное количество подов запущено
- ✓ Replication controller не только проверяет что нужное количество подов выполняется но и создаст заново или реплицирует под если что-то пойдет не так.

Node Overview



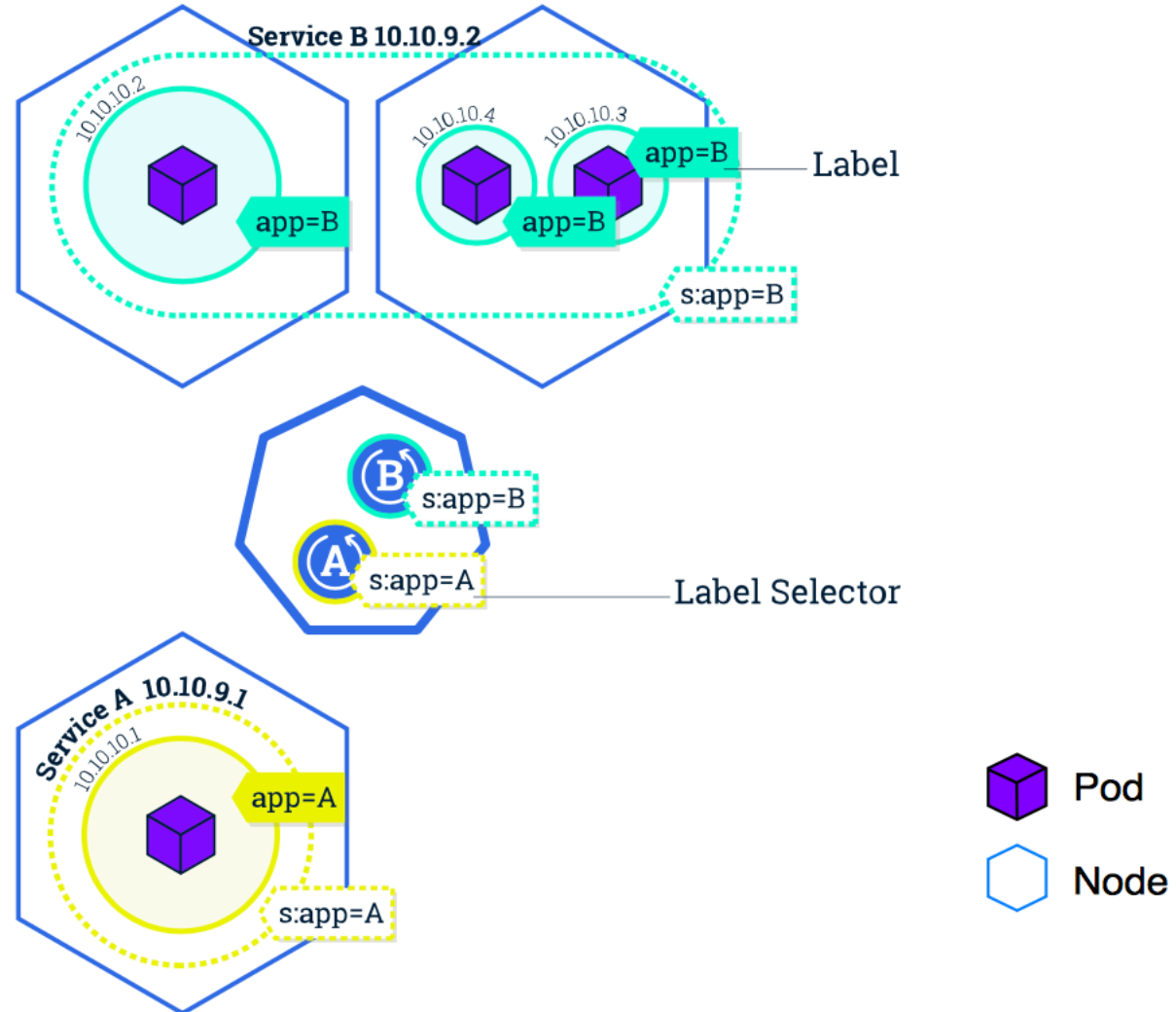
Service

- ✓ K8s service выполняет роль endpoint и load balance
- ✓ Позволяет пользователю заботиться об одной endpoint
- ✓ Особенно service будет “фронтом” для группы из replicated pods и распределять трафик для них
- ✓ При правильной конфигурации поды, созданные replicated controllers будут автоматически зарегистрированы как соответствующий сервис

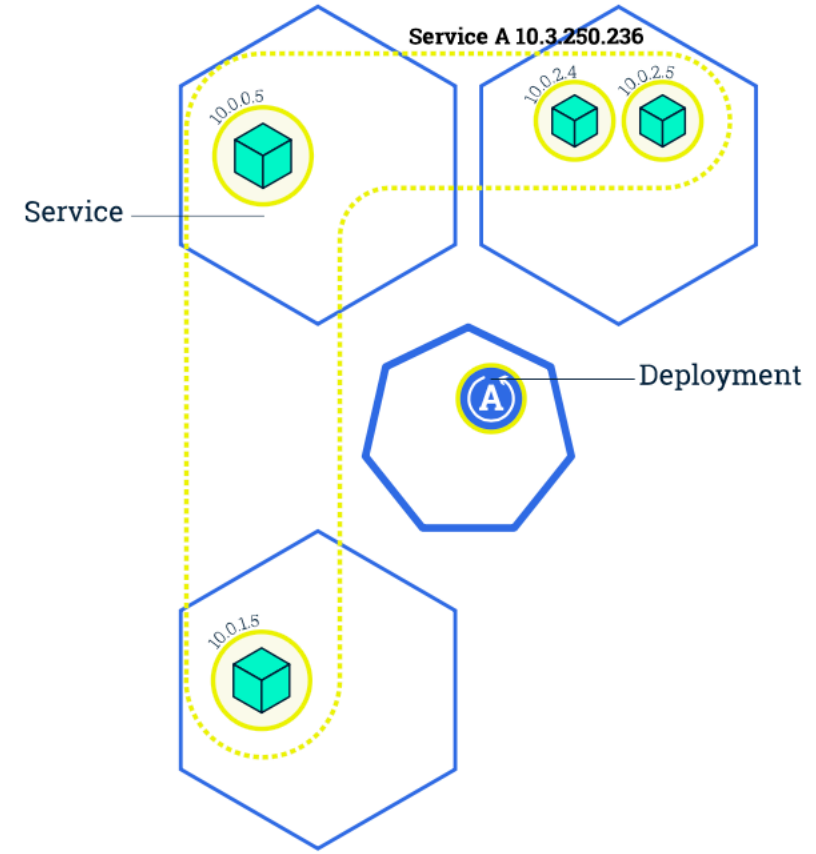
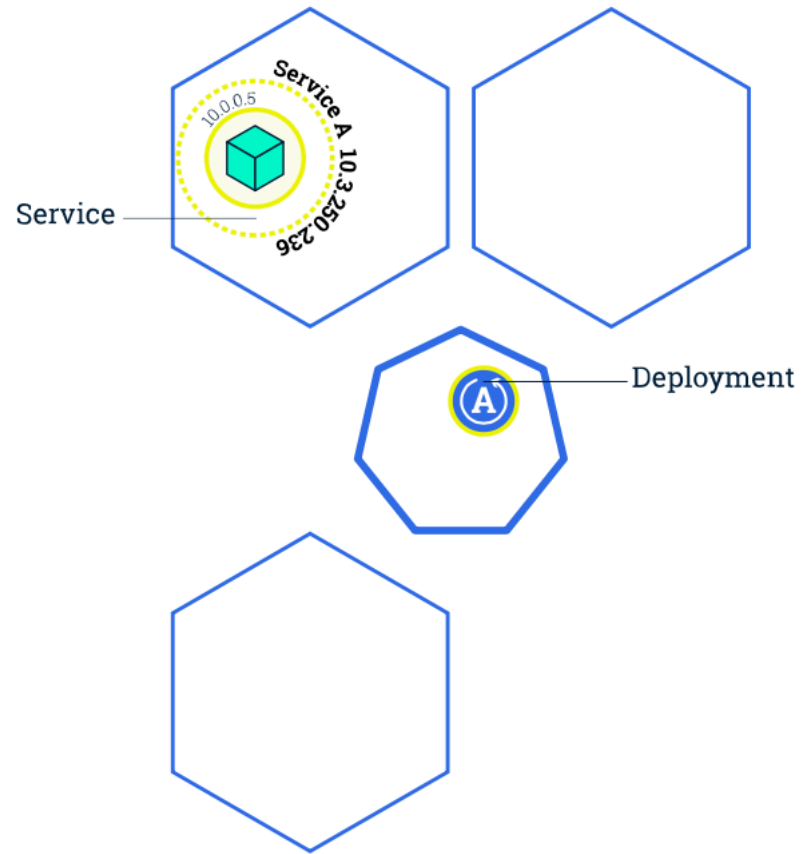
Label

- ✓ Ярлыки которые могут быть использованы для других юнитов
- ✓ Очень полезны для контроля и управления вашими юнитами
- ✓ Юниты могут иметь более одного Label

Services and Labels



Scaling overview



Плюсы Kubernetes

- ✓ Встроенный механизм деплоя
- ✓ Встроенный автоскейлинг
- ✓ Полностью готовая внутренняя инфраструктура
- ✓ Опенсорс
- ✓ Комьюнити

Devops и Cloud

✓ AWS

✓ Azure

✓ IBM Cloud

Мои рекомендации

- Средства разработки

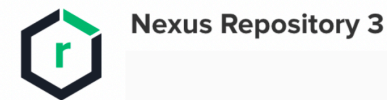
- Хранилище исходного кода (GitLab)
- Хранилище артефактов и образов (Nexus)
- CI/CD (Jenkins)

- Средства Автоматизации

- Развертывание инфраструктуры решения (Ansible)
- Сборка/тестирование/развертывание компонентов решения

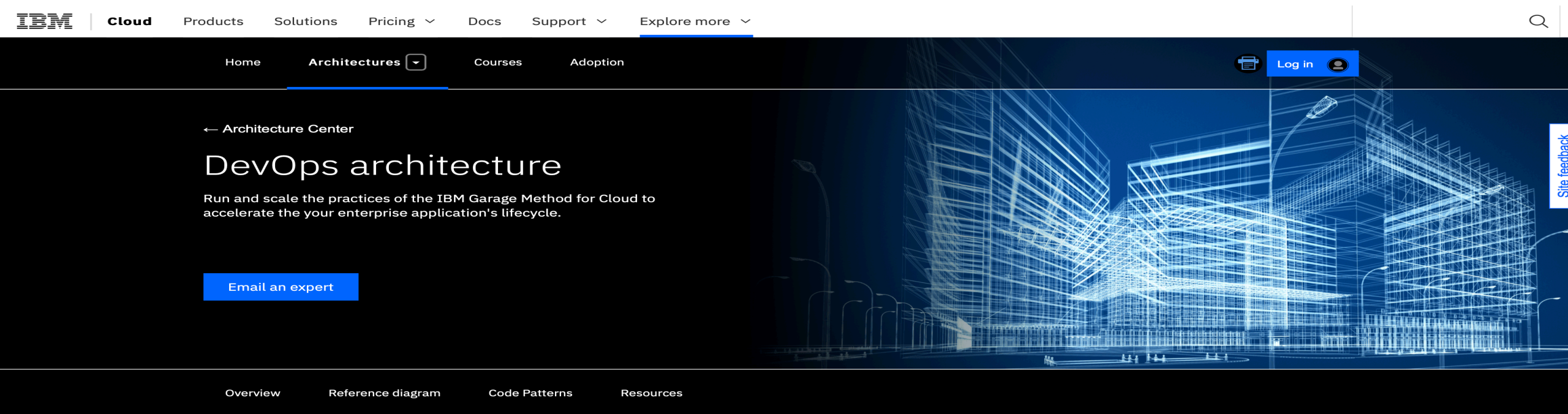
- Средства Сопровождения

- Сбор и управления логами (ELK)



Полезные ресурсы

- <https://www.ibm.com/cloud/architecture/architectures/devOpsArchitecture/overview>



The screenshot shows the IBM Cloud Architecture Center page for DevOps architecture. The top navigation bar includes the IBM logo, 'Cloud', and links for Products, Solutions, Pricing, Docs, Support, and Explore more. A secondary navigation bar has links for Home, Architectures (selected), Courses, and Adoption, along with a 'Log in' button. The main content area features a blue wireframe background image of a modern building. On the left, there is a breadcrumb '← Architecture Center', the title 'DevOps architecture', a descriptive paragraph, and a blue 'Email an expert' button. On the right, there is a vertical 'Site feedback' button. A bottom navigation bar contains links for Overview, Reference diagram, Code Patterns, and Resources.

IBM | Cloud | Products | Solutions | Pricing | Docs | Support | Explore more

Home | Architectures | Courses | Adoption | Log in

← Architecture Center

DevOps architecture

Run and scale the practices of the IBM Garage Method for Cloud to accelerate the your enterprise application's lifecycle.

Email an expert

Overview | Reference diagram | Code Patterns | Resources

DevOps is the modern way of delivering digital products

The IBM Garage Method for Cloud includes DevOps practices that are combined with Enterprise Design Thinking, agile methods, The Lean Startup, and modern architectures to achieve enterprise agility and responsiveness.

DevOps is an approach where the traditionally siloed teams of development and operations come together with a product-focused mindset to deliver faster and with higher quality. Over time, other stakeholders, such as quality assurance (testing), security, and even users can come together to collaborate to continuously deliver software. By following this approach, your

[Read the field guide](#)

