QA
מסע

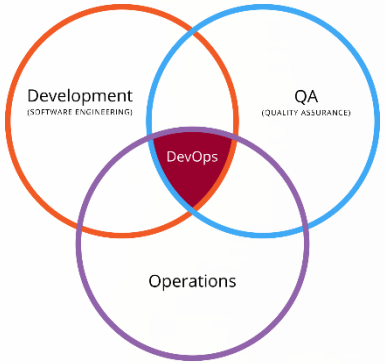
אלכס גורבצ'וב

АЛЕКСЕЙ ГОРБАЧЁВ

DEVOPS

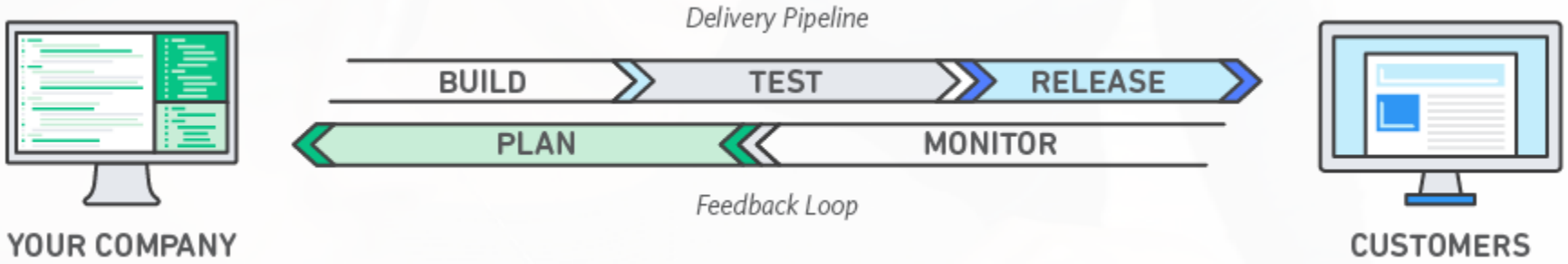
DEVOPS

DevOps

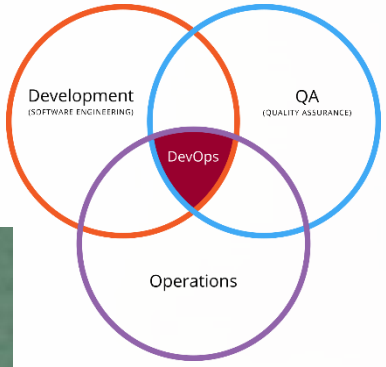


Что такое DevOps

DevOps — это комбинация слов «разработка» (development) и «эксплуатация» (operations), которая отражает процесс интеграции этих дисциплин в единый непрерывный процесс. Разработчики и тестировщики отвечают за Development, а администраторы - за Operations



DevOps

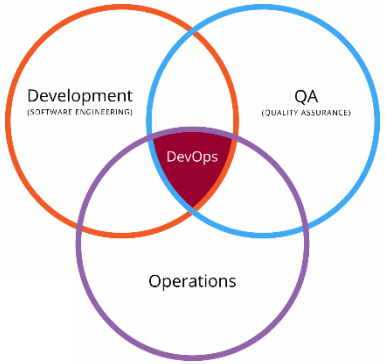


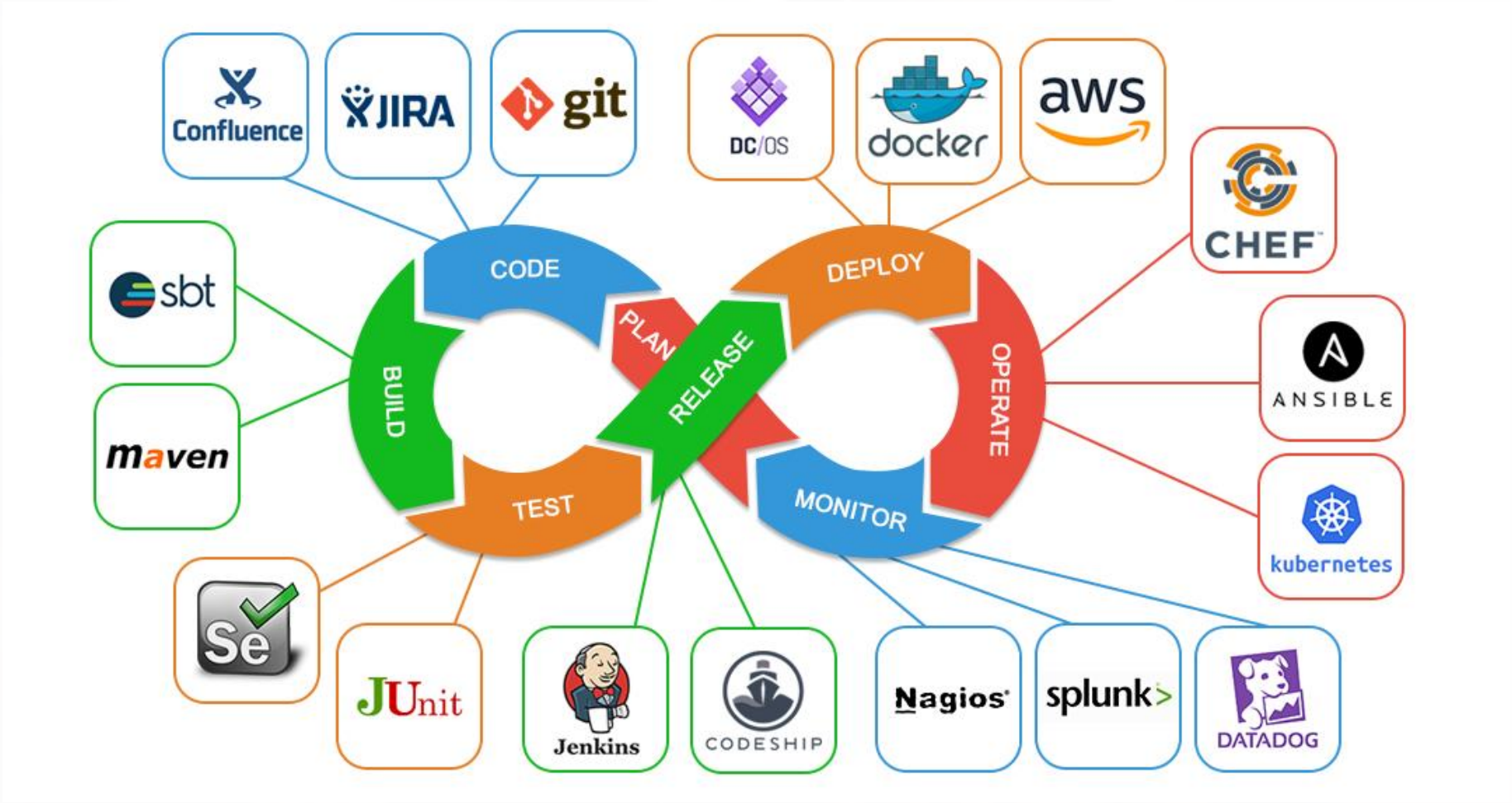
DevOps – это...

технологии + процессы + культура взаимодействия в команде

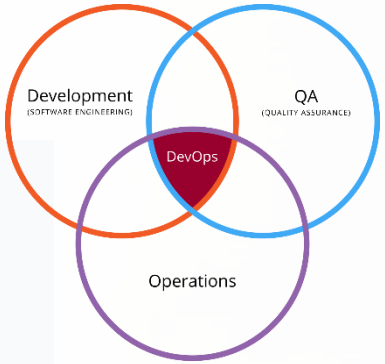
непрерывная доставка ценностей конечным пользователям

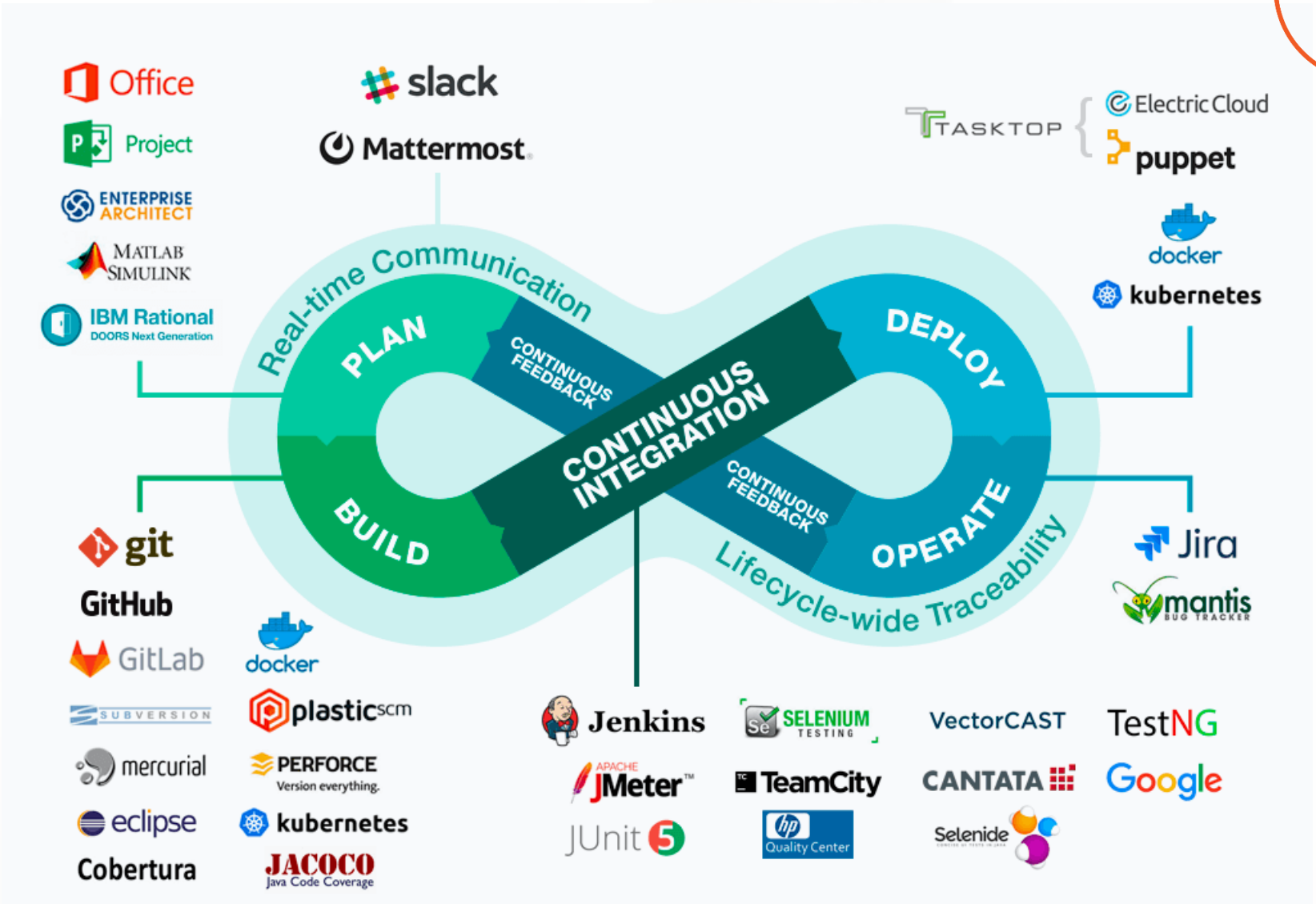
DevOps


Development
(SOFTWARE ENGINEERING)
QA
(QUALITY ASSURANCE)
DevOps
Operations

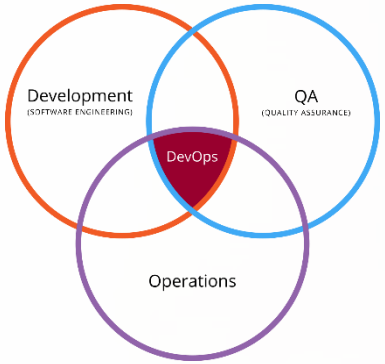

Confluence
JIRA
git
DC/OS
docker
aws
CHEF
sbt
maven
ANSIBLE
kubernetes
Se
JUnit
Jenkins
CODESHIP
Nagios
splunk>
DATADOG
CODE
PLAN
BUILD
RELEASE
DEPLOY
OPERATE
TEST
MONITOR

DevOps


Development
(SOFTWARE ENGINEERING)
QA
(QUALITY ASSURANCE)
DevOps
Operations


Office
Project
ENTERPRISE ARCHITECT
MATLAB SIMULINK
IBM Rational DOORS Next Generation
slack
Mattermost
TASKTOP
Electric Cloud
puppet
docker
kubernetes
Real-time Communication
PLAN
CONTINUOUS FEEDBACK
CONTINUOUS INTEGRATION
DEPLOY
BUILD
CONTINUOUS FEEDBACK
OPERATE
Lifecycle-wide Traceability
git
GitHub
GitLab
SUBVERSION
mercurial
eclipse
Cobertura
docker
plasticscm
PERFORCE Version everything.
kubernetes
JACOCO Java Code Coverage
Jenkins
SELENIUM TESTING
VectorCAST
TestNG
APACHE JMeter
TeamCity
CANTATA
Google
JUnit 5
hp Quality Center
Selenide
Jira
mantis BUG TRACKER

DevOps

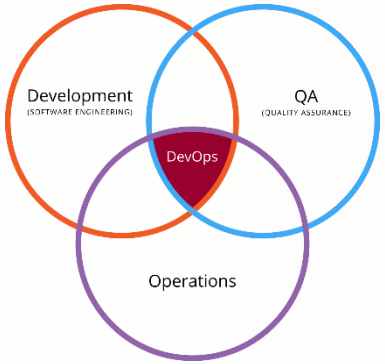


DevOps-практики

DevOps-практики покрывают все этапы жизненного цикла ПО:

- Инфраструктура как код
- CI/CD
- Logs
- Мониторинг

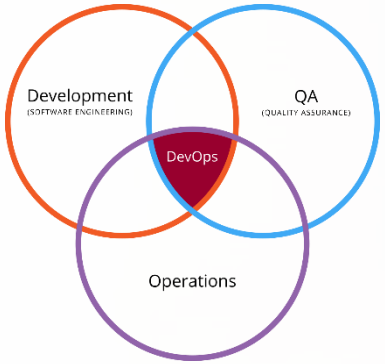
DevOps



Инфраструктура как код

- Так называют подход, при котором инфраструктура проекта создается и управляется при помощи кода, а не через прямое взаимодействие с серверами. Это про микросервесную архитектуру Docker and Kubernetes.
- Микросервисная архитектура - это подход к созданию приложения, подразумевающий отказ от единой, монолитной структуры. То есть вместо того чтобы исполнять все ограниченные контексты приложения на сервере с помощью внутрипроцессных взаимодействий, мы используем несколько небольших приложений, каждое из которых соответствует какому-то ограниченному контексту.
- Повышается отказоустойчивость, облегчается процесс доработки кода.

DevOps

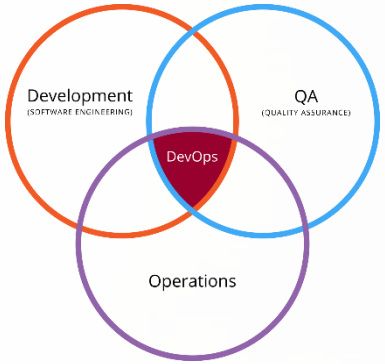


CI/CD – что это?

CI - Continuous Integration, CD - Continuous Delivery: непрерывная интеграция и непрерывная поставка.

Фактически под этим подразумевают цепочку действий, которая позволяет разработчикам увеличить скорость внедрения изменений, сохраняя или даже повышая качество кода, а с ним сервиса или приложения.

DevOps



Logs

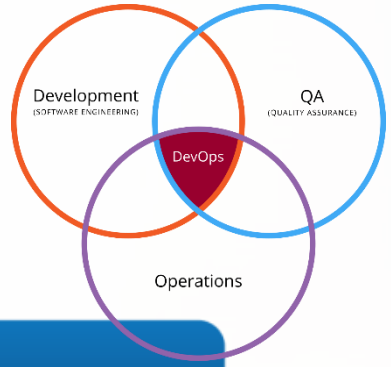
Улучшение процесса работы с логами:

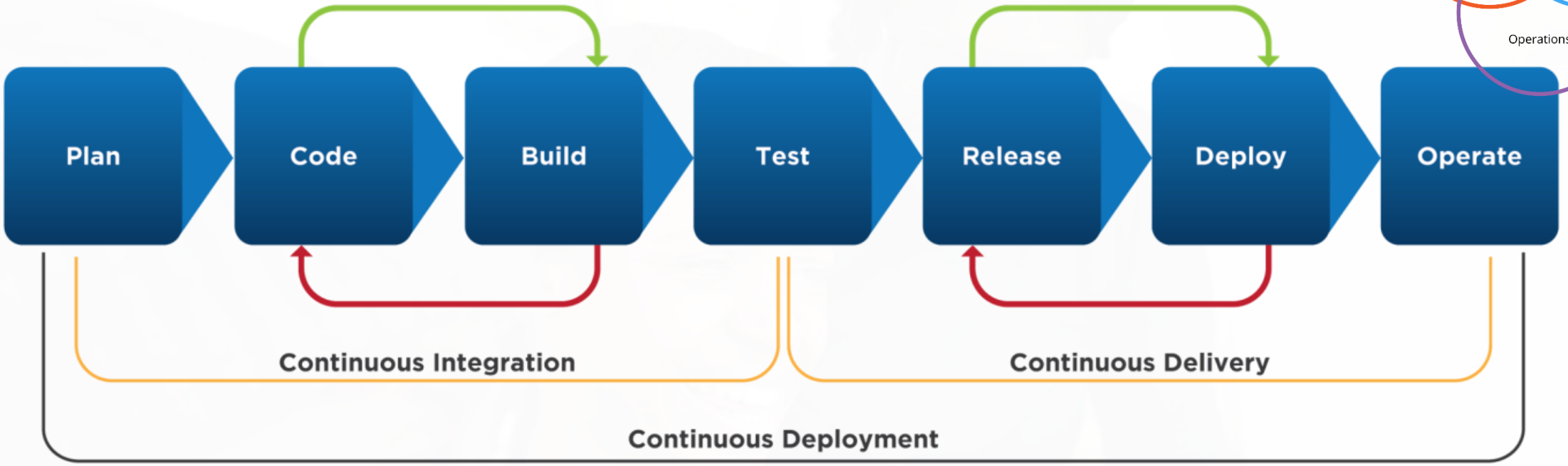
- Экономия места на СХД (Система хранения данных) во много раз
- Удобство работы с логами
- Централизованная система по работе с логами

DEVOPS PROJECT

DEVOPS PROJECT

DevOps Project





Planning

- Requirement finalization
- Updates & new changes
- Architecture & design
- Task assignment
- Timeline finalization

Code

- Development
- Configuration finalization
- Check-in source code
- Static-code analysis
- Automated review & peer review

Build

- Compile code
- Unit testing
- Code-metrics
- Build container images or package
- Preparation or update in deployment templated
- Create or update monitor dashboards

Test

- Integration test with other component
- Load & stress test
- UI testing
- Penetration testing
- Requirement testing

Release

- Preparing release notes
- Version tagging
- Code freeze
- Feature freeze

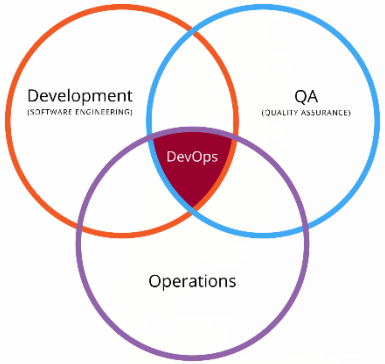
Deploy

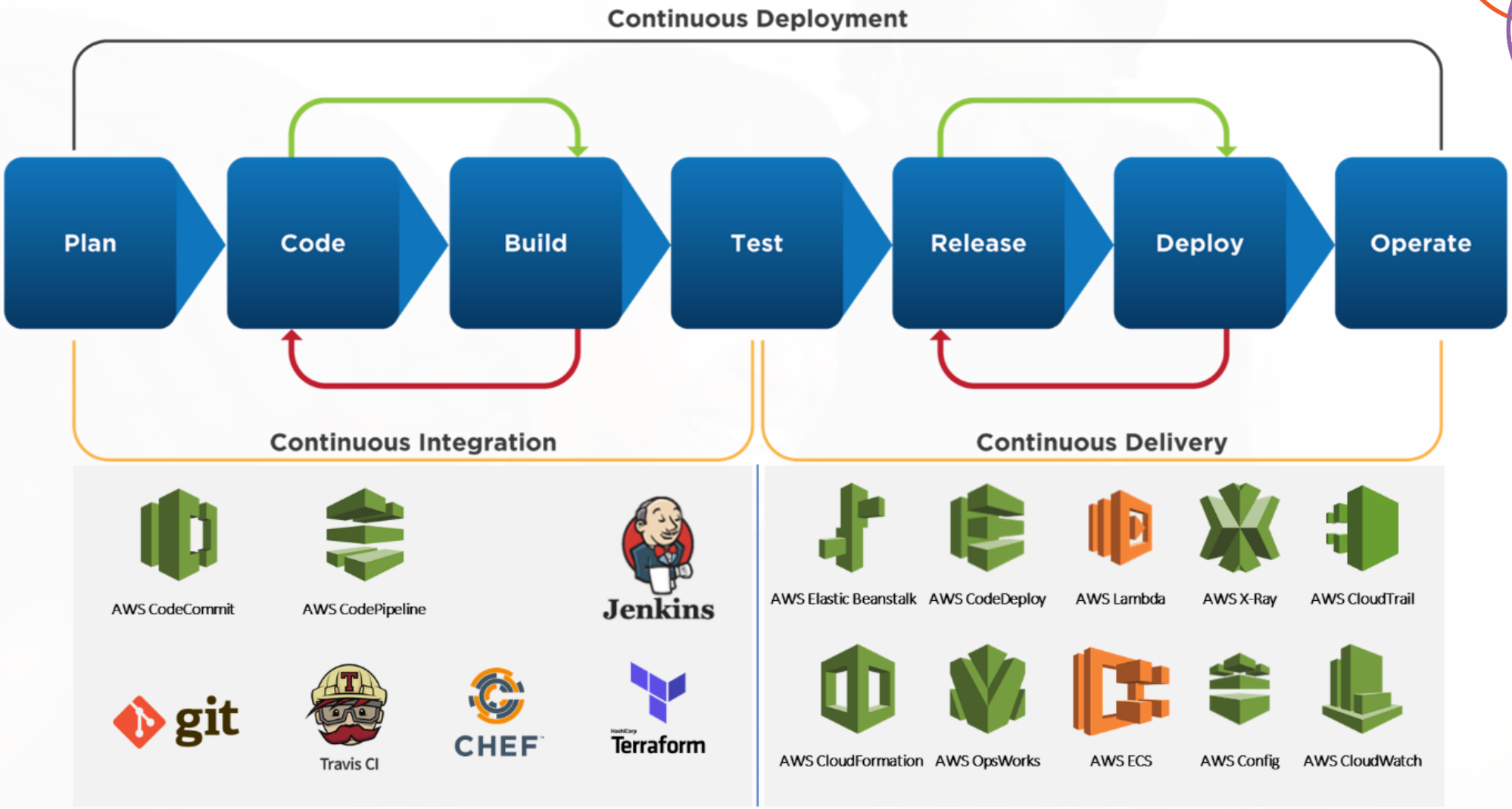
- Updating the infrastructure i.e staging, production
- Verification on deployment i.e smoke tests

Operate

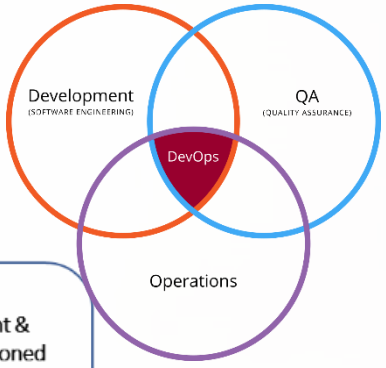
- Monitor designed dashboard
- Alarm triggers
- Automatic critical events handler
- Monitor error logs

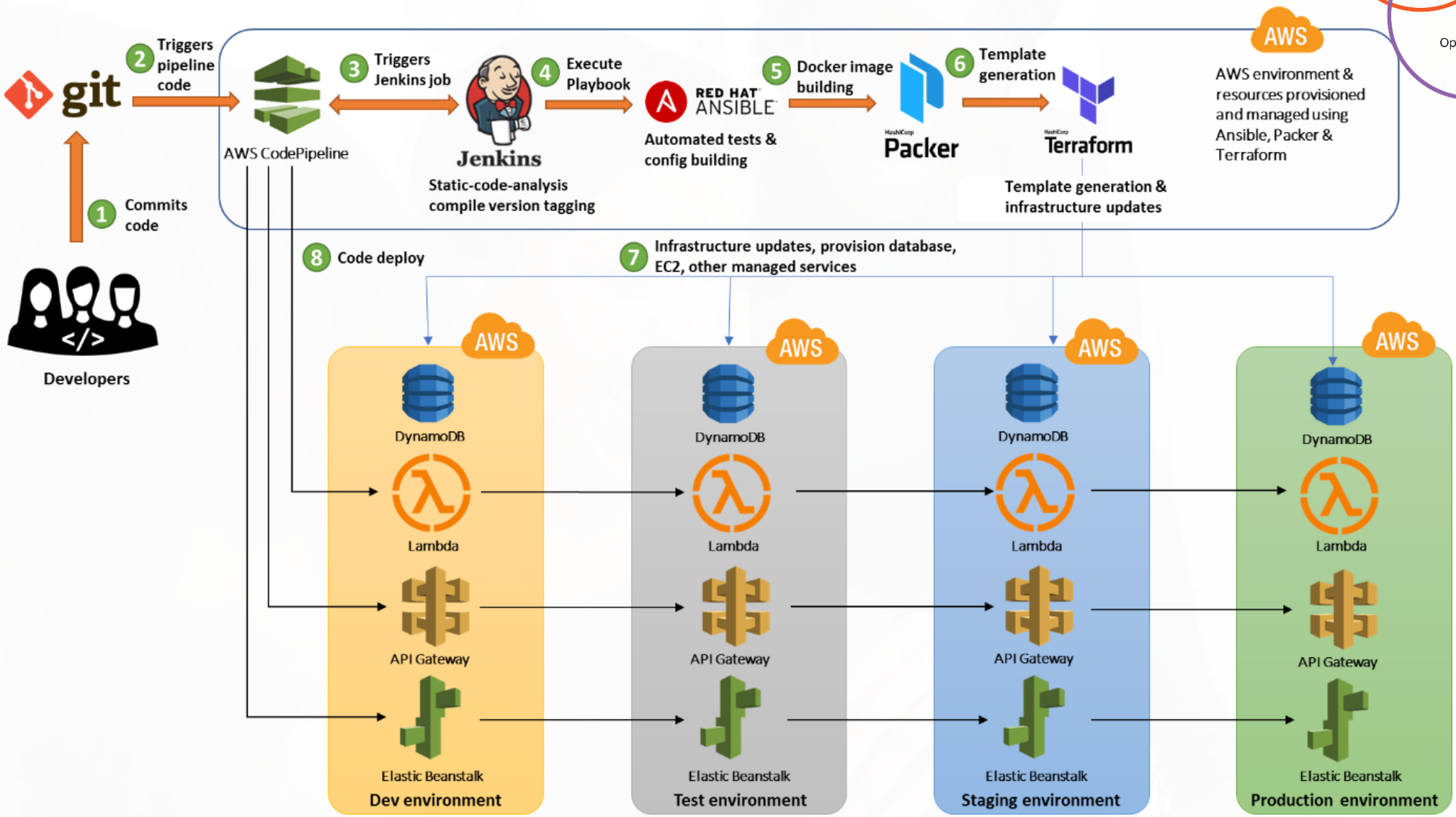
DevOps Project





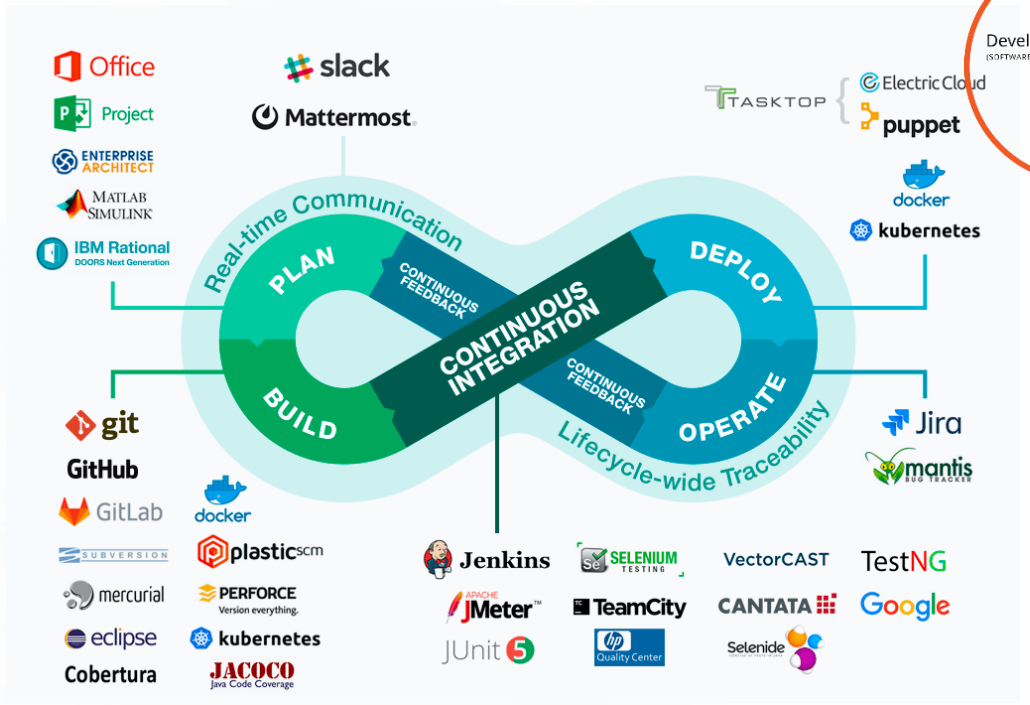
DevOps Project

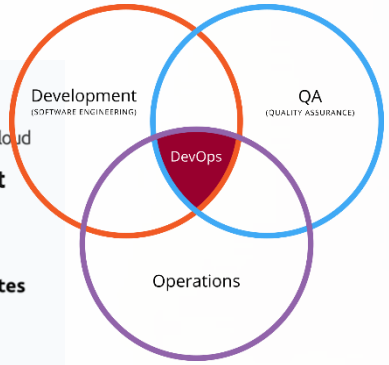


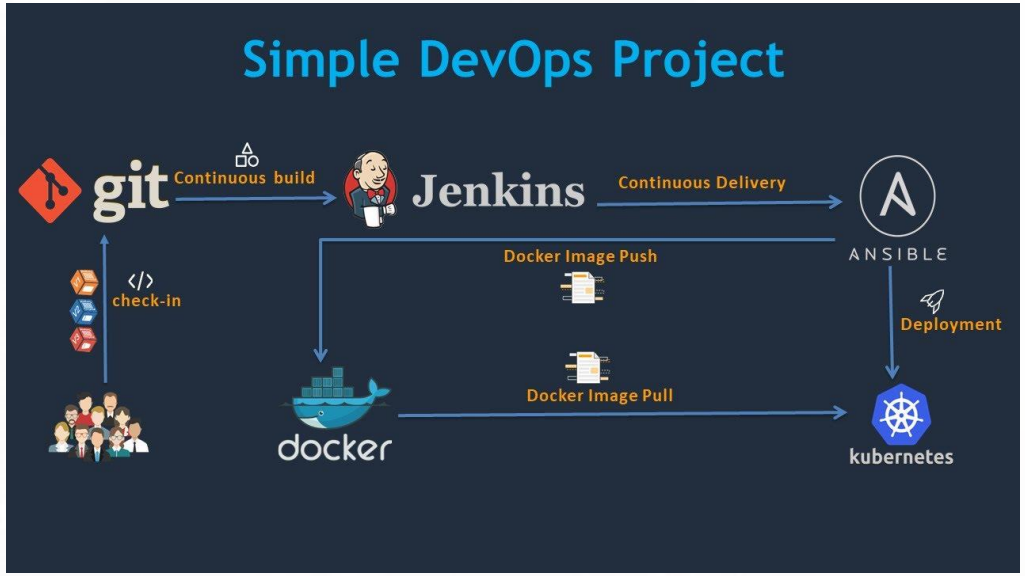


DevOps Project

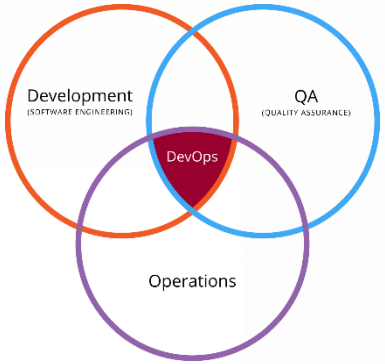
1. Plan (Capture & Tracking)
    a. Application Life Management -> Jira
    b. Knowledge Sharing -> Blueprint
2. Code Building
    a. Creating Repository (Version Control) -> GitHub/Flask/GitLab
    b. CI -> Jenkins
    c. Build -> Docker/Maven
3. Test -> JUnit
4. Release -> Docker
5. Deploy
    a. Configuration Management -> Ansible/Terraform
    b. Artefact Management - > Docker
6. Operate ->
    a. Cloud/Iaas/Paas -> AWS/AZURE
    b. Orchestration -> Kubernetes
    c. BI/Monitoring -> Kibana/Elasticsearch







DevOps Project



Product Owner | Dev Lead | Developer | Operations

Build

Author User Story → Review User Story → Get Code from GitHub → Develop Code → Unit Test on Desktop → Commit then Pull Request → Approve Pull Request

Package

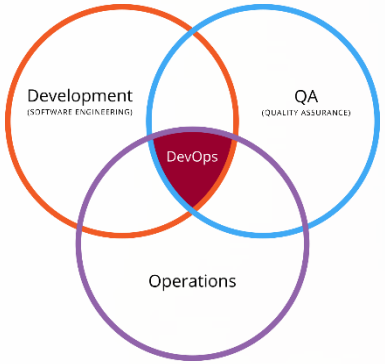
Github web hook to Jenkins → Jenkins run Jobs → Jenkins push MCM version → Post to Slack

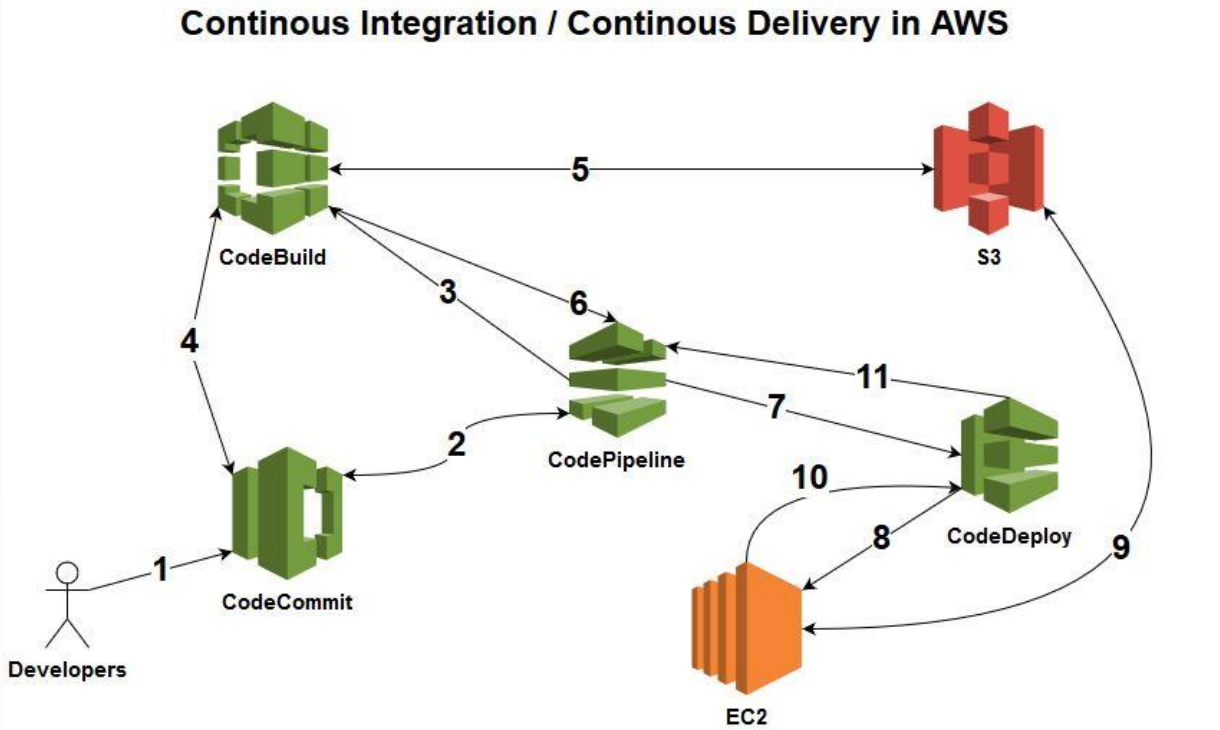
Verify

Deploy/Smoke Tests → Deploy/ Functional → Deploy/ Performance → Deploy/ QA → Deploy/ UAT → Deploy/ Production → Post to Slack

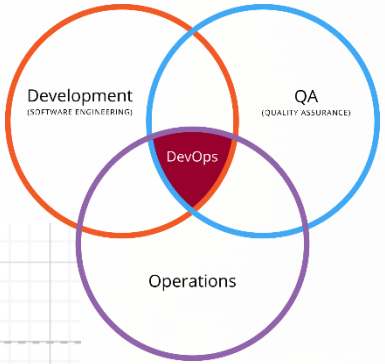
DevOps Project

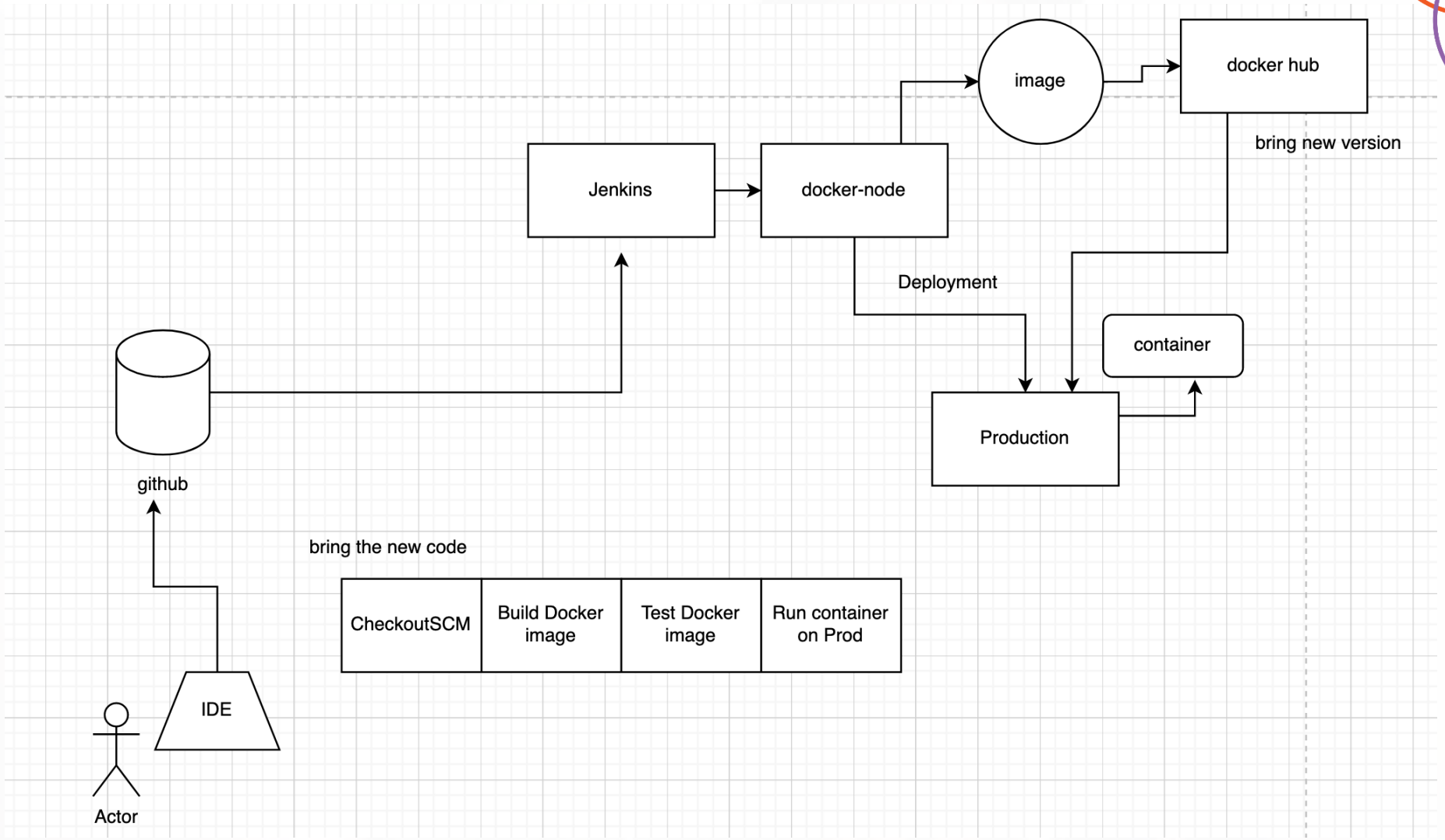
1) The Developer writes a code and commits it to CodeCommit GIT repository
2) CodePipeline detects a change in repository and starts the CI/CD process
3) CodePipeline triggers CodeBuild to start building our application
4) CodeBuild copies the code from CodeCommit and builds an application
5) When the application is compiled, artifacts are copied to S3 bucket
6) Upon completion of the build process, CodeBuild informs CodePipeline about success.
7) CodePipeline triggers CodeDeploy to start its job
8) CodeDeploy contact the agents and sends the instructions
9) Agents copy installation bundle from S3 and start the installation process
10) Upon completion, agents inform CodeDeploy about the deployment status
11) CodeDeploy inform CodePipeline about the completion status





DevOps Project





VIRTUALIZATION

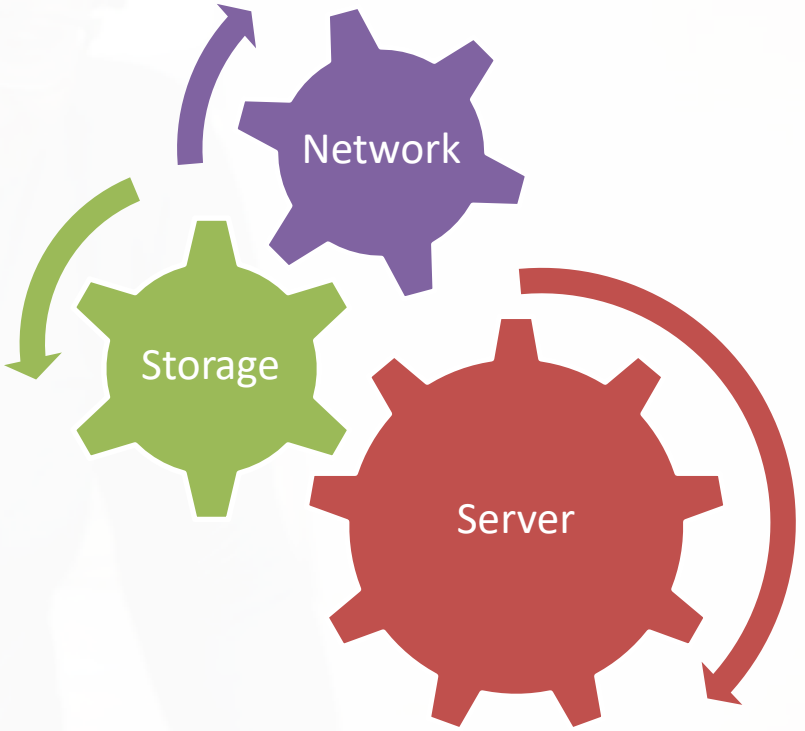
ВИРТУАЛИЗАЦИЯ

Virtualization

Виртуализация лежит в основе облачных технологий, а они - повсюду.

Благодаря им мы не скачиваем программы на компьютер, а работаем онлайн; не закупаем оборудование для вычислений, а просто арендуем мощности. Рядовые исследователи могут работать с большими данными, перенося их обработку в облака. Беспилотный транспорт и автономные роботы хорошо ориентируются в пространстве, потому что «думают» в бесконечно мощном дата-центре.

Виртуализация помогает экономить на оборудовании. Владельцы инфраструктуры делят один физический сервер на несколько виртуальных и так снижают затраты. Использовать все преимущества технологии помогает программное обеспечение для виртуализации.



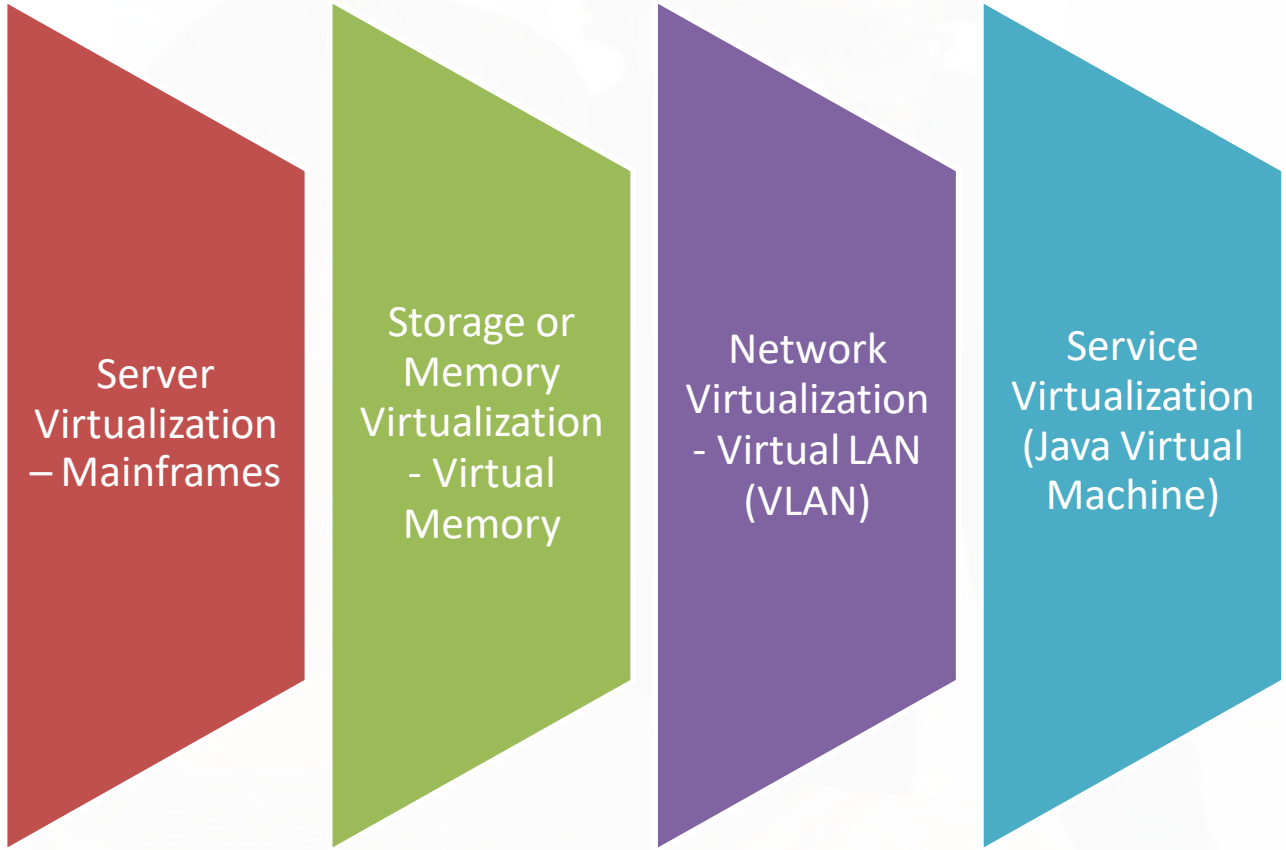
Virtualization

Что такое виртуализация

Виртуализация - это выделение вычислительных ресурсов, а также изолирование процессов, которые выполняются на одном оборудовании.
Виртуальным может быть сервер, хранилище или сеть. У виртуального сервера, как и у настоящего, есть место на диске, оперативная память, процессор. На него можно установить операционную систему.

Virtualization

Что такое виртуализация

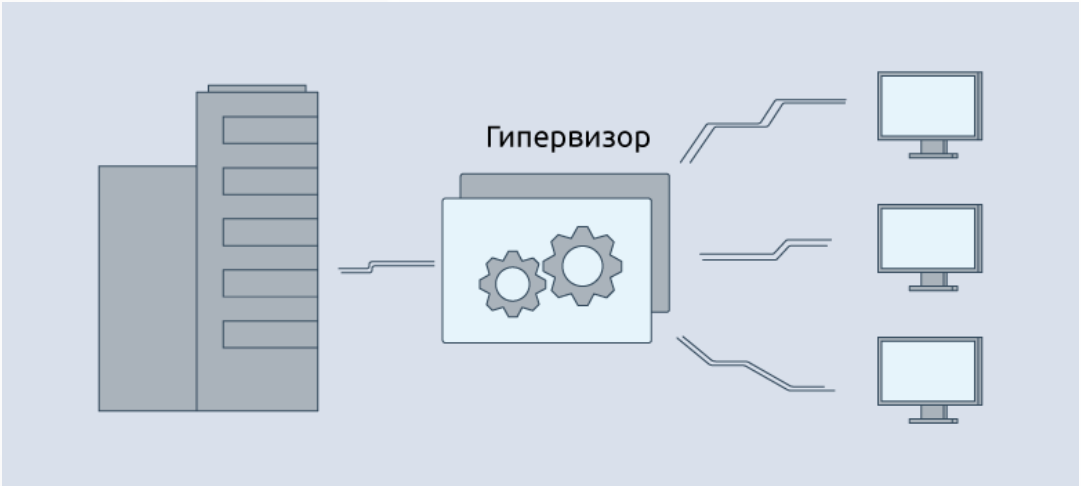


Virtualization

Гипервизоры

Гипервизор - это программа или устройство, которое создаёт и запускает виртуальные машины. Гипервизор делает так, что на каждой ВМ можно запустить операционную систему. Разделяет ресурсы между ВМ, обеспечивает их независимое включение и выключение, изолирует друг от друга.

Гипервизоры бывают программные и аппаратные. Аппаратные считаются более производительными, чем программные.

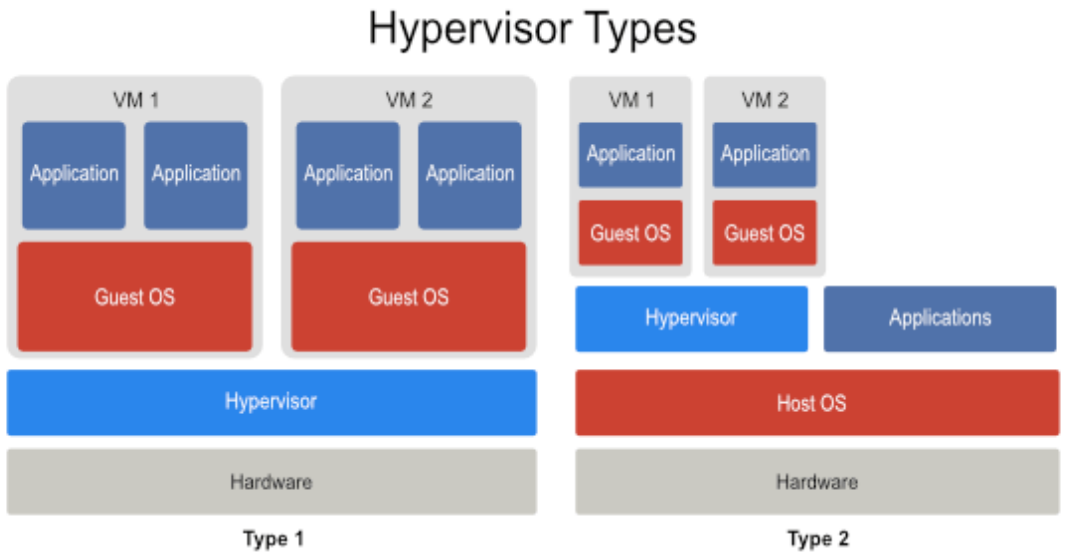


Virtualization

Гипервизоры

Гипервизор первого типа

Работает непосредственно на физическом аппаратном обеспечении хост-машины и называется "bare-metal гипервизор". Гипервизор типа 1 не должен загружать базовую операционную систему. Он использует прямой доступ к исходному оборудованию и никакому другому программному обеспечению (ОС и драйверы), и считается самым эффективным и наиболее производительным.



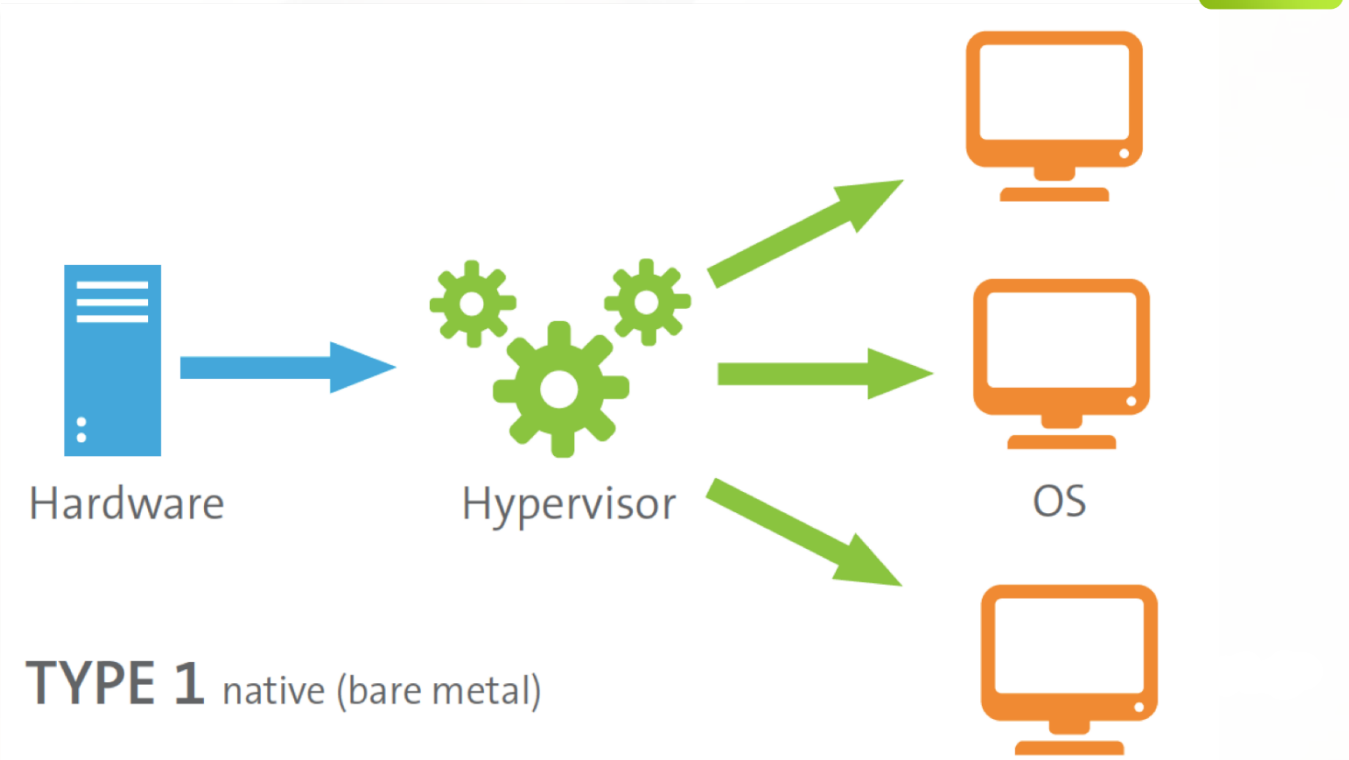
Гипервизор второго типа

Устанавливается поверх существующей ОС. Иногда его называют хостируемым гипервизором, потому что он зависит от существующей ОС хост-машины для управления вызовами к процессору, памяти, хранилищу и сетевым ресурсам.

Virtualization

Гипервизоры

Гипервизор первого типа



CITRIX XenServer

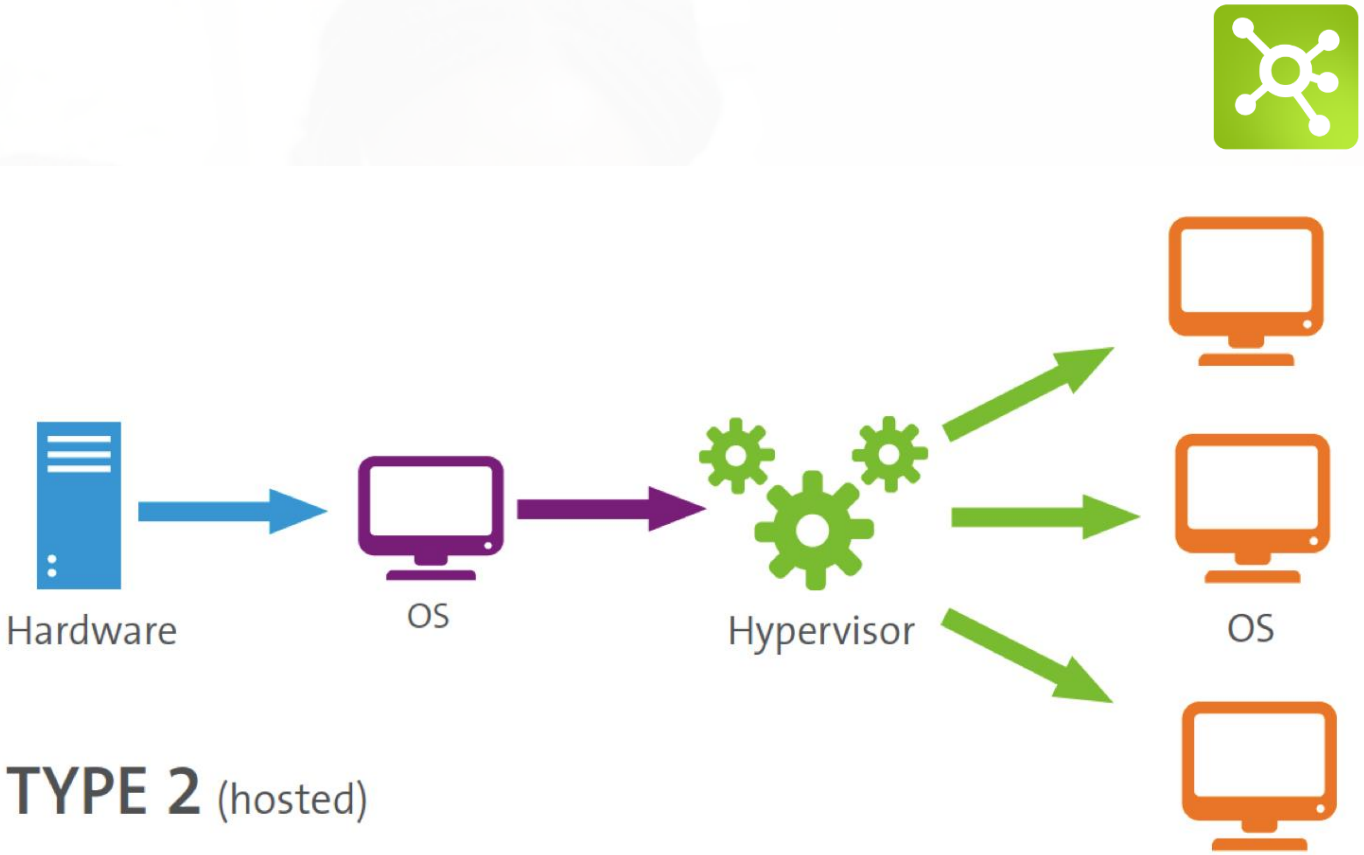
vmware ESXi

Microsoft Hyper-V

Virtualization

Гипервизоры

Гипервизор второго типа



ORACLE VM VirtualBox

Linux KVM

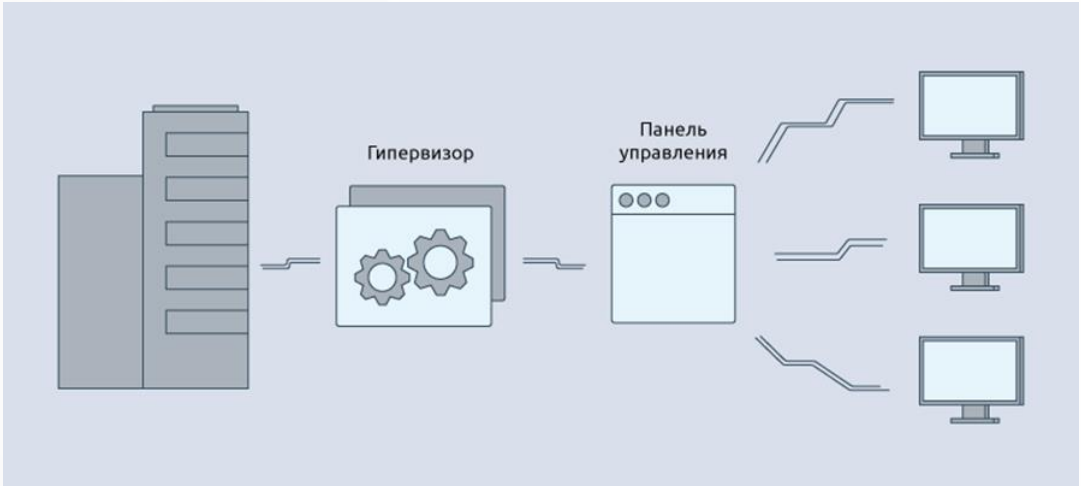
vmware

Virtualization

Панели управления

Панель управления виртуализацией - сервис, в котором можно работать с гипервизором через графический интерфейс. Помогает создавать виртуальные машины нужных конфигураций, устанавливать на них ПО, настраивать и делать резервные копии.

Панели виртуализации различаются по тому, с каким гипервизором работают. Мы рассмотрим те, что работают с гипервизором KVM, так как у коммерческих гипервизоров Oracle VM Server, Microsoft Hyper-V и VMware ESX свои панели управления.



Virtualization

Виртуальные машины

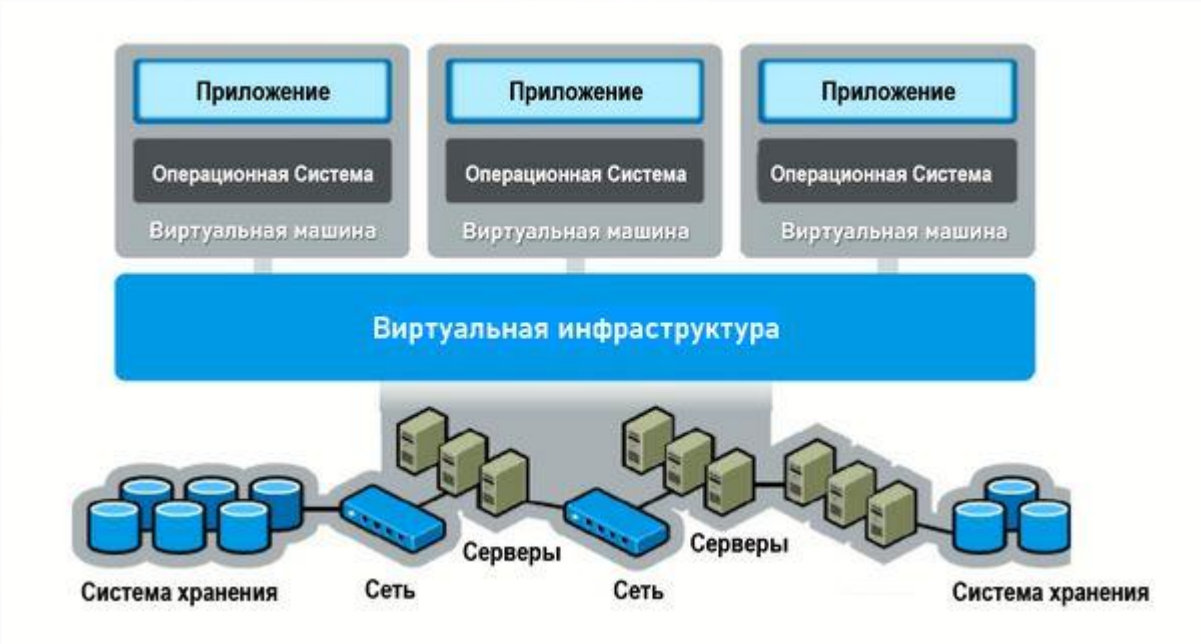
Виртуальная компьютерная система, также называемая виртуальной машиной (ВМ), - это строго изолированный контейнер ПО, содержащий операционную систему и приложения. Каждая автономная ВМ полностью независима. Размещение нескольких ВМ на одном компьютере обеспечивает работу нескольких операционных систем и приложений на одном физическом сервере, так называемом «узле».

Тонкий слой ПО, называемый гипервизором, отделяет виртуальные машины от узла и по мере необходимости динамически выделяет вычислительные ресурсы каждой виртуальной машине.

Virtualization

Типы виртуализации

- Виртуализация серверов
- Виртуализация сети
- Виртуализация настольных компьютеров

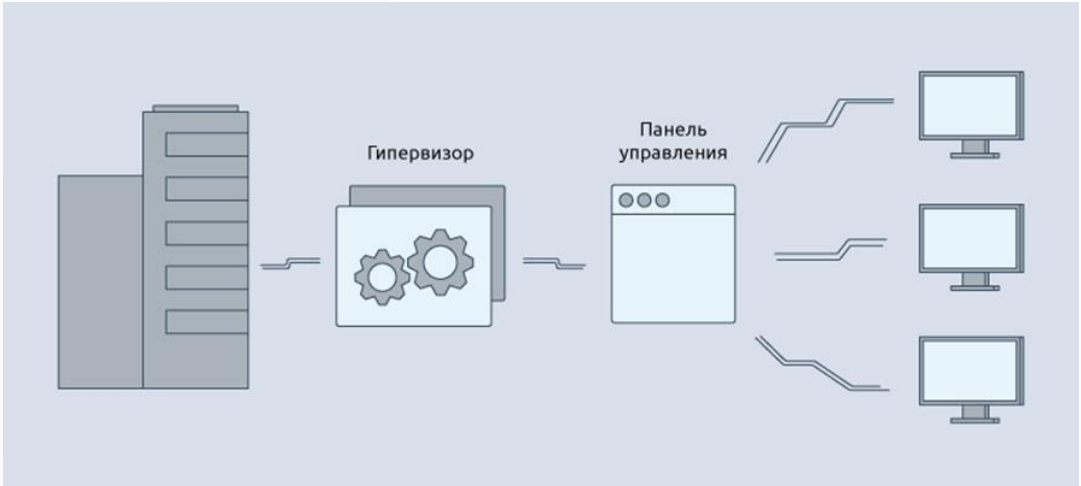


Virtualization

Создание виртуальных машин

Укажите количество vCPU, объем хранилища и RAM, выберите ОС и приложения для установки.

Остальное панель сделает сама: за 2 минуты создаст ВМ и установит софт. Если нужно много однотипных машин, используйте возможность клонирования или создайте пользовательский шаблон и разворачивайте новые виртуальные машины из него. Кроме того, на форме создания можно указать, сколько ВМ нужно запустить.



VIRTUALBOX

VIRTUALBOX

VirtualBox

VirtualBox

VirtualBox - это программное обеспечение, которое позволяет создавать виртуальные компьютеры на вашем компьютере. Для установки VirtualBox на компьютер с Windows следуйте этим инструкциям:

1. Загрузите установочный файл VirtualBox с официального сайта VirtualBox: https://www.virtualbox.org/.
2. Запустите установочный файл и следуйте инструкциям на экране. Нажмите "Далее", чтобы продолжить.
3. Выберите папку для установки VirtualBox. По умолчанию программа будет установлена в "C:\Program Files\Oracle\VirtualBox". Если вы хотите изменить путь, нажмите кнопку "Изменить" и выберите другую папку.

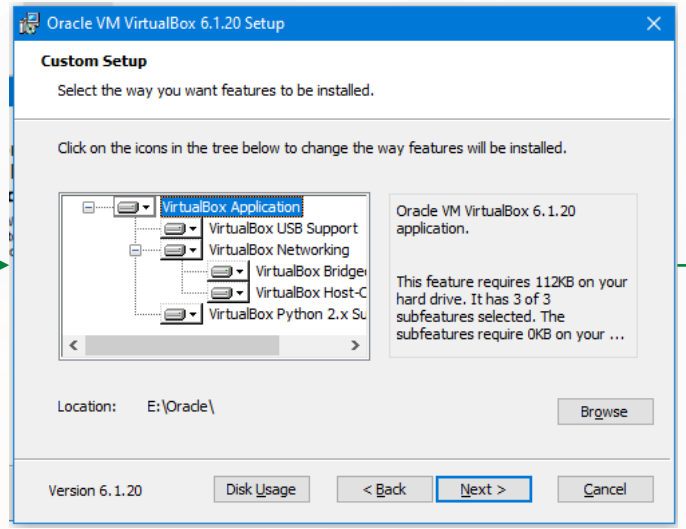
VirtualBox

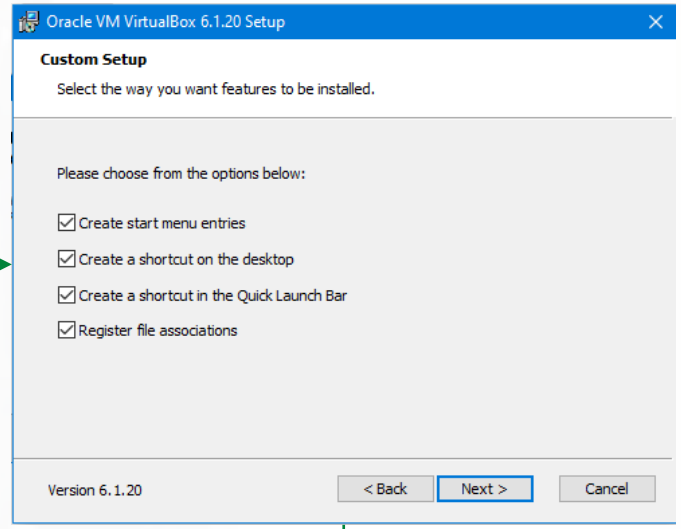
VirtualBox

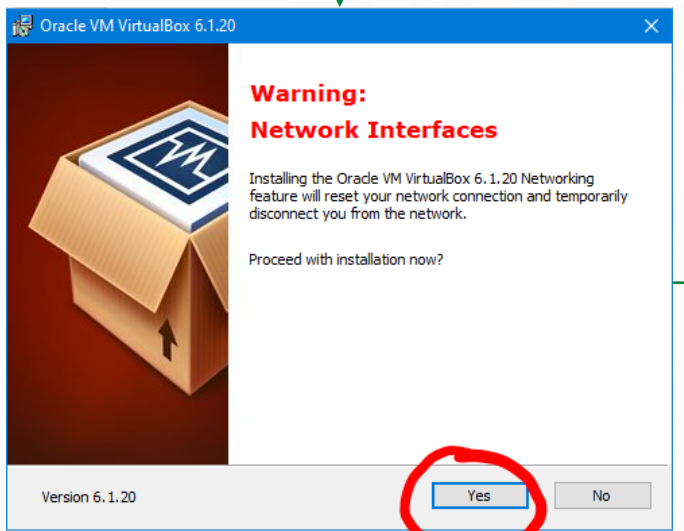
4. Установите дополнительные компоненты, если это необходимо. Вы можете выбрать, какие компоненты установить. Если вы не уверены, оставьте выбранными все компоненты по умолчанию.
5. Нажмите "Далее" и установите VirtualBox.
6. После завершения установки запустите VirtualBox, нажав на значок на рабочем столе или в меню "Пуск".
7. Чтобы создать новую виртуальную машину, нажмите на кнопку "Новая". Далее следуйте инструкциям мастера создания виртуальной машины.

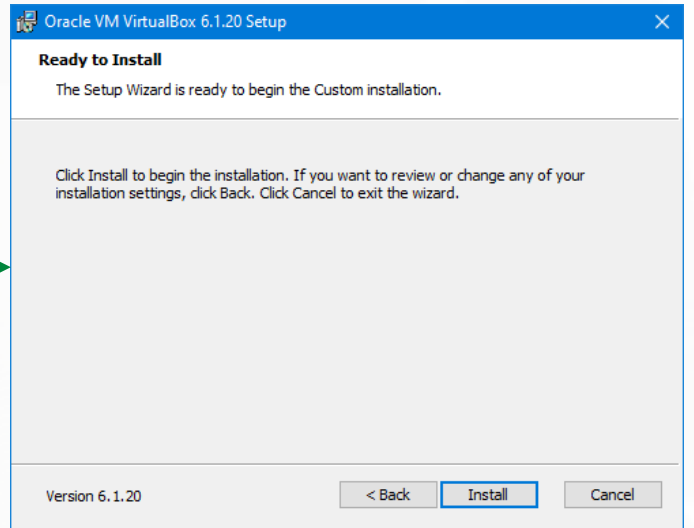
VirtualBox

VirtualBox









VirtualBox

VirtualBox

Чтобы установить ОС Ubuntu в VirtualBox, выполним следующие действия:

1. Скачайте образ диска Ubuntu с официального сайта https://ubuntu.com/download.
2. Запустите VirtualBox и создайте новую виртуальную машину, нажав на кнопку "Новая". В появившемся окне укажите имя машины, выберите тип операционной системы "Linux" и версию "Ubuntu". Нажмите "Далее".
3. Выберите объем оперативной памяти, выделенной для виртуальной машины. Рекомендуется выбрать значение не менее 2 ГБ. Нажмите "Далее".
4. Создайте виртуальный жесткий диск, выбрав опцию "Создать виртуальный жесткий диск сейчас". Выберите тип жесткого диска "VDI" и нажмите "Далее".

VirtualBox

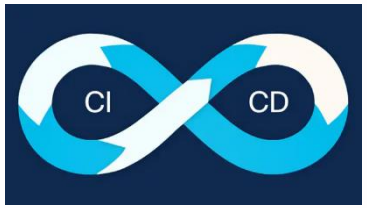
VirtualBox

5. Выберите тип хранения жесткого диска. Рекомендуется выбрать опцию "Динамический размер", чтобы жесткий диск расширялся по мере необходимости. Нажмите "Далее".
6. Укажите размер жесткого диска. Рекомендуется выбрать значение не менее 20 ГБ. Нажмите "Создать".
7. Выберите созданную виртуальную машину и нажмите на кнопку "Настроить". В появившемся окне перейдите на вкладку "Накопители".
8. Нажмите на кнопку "Добавить оптический диск" и выберите скачанный образ диска Ubuntu. Нажмите "ОК".
9. Запустите виртуальную машину, нажав на кнопку "Старт". Запустится установщик Ubuntu. Следуйте инструкциям установщика, чтобы установить ОС Ubuntu на виртуальный жесткий диск.

CI/CD

CI/CD

CI/CD

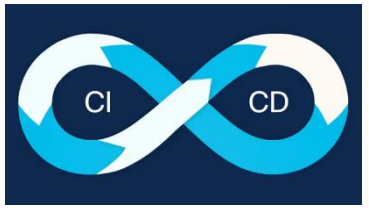


CI/CD и его среда обитания

Continuous Integration (CI) и Continuous Delivery (CD) - это две практики разработки программного обеспечения, которые взаимодействуют друг с другом и позволяют более эффективно и быстро создавать и доставлять качественное программное обеспечение.

Continuous Integration - это практика автоматической сборки и тестирования приложения после каждого изменения кода в репозитории. Она предполагает интеграцию нового кода в основную ветку проекта, сборку и запуск автоматических тестов. Это позволяет быстро обнаруживать и устранять ошибки, а также обеспечивает единый и актуальный кодовую базу.

CI/CD

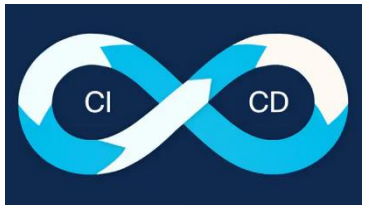


CI/CD и его среда обитания

Continuous Delivery - это практика автоматической доставки приложения в боевую среду после каждого успешного цикла **Continuous Integration**. Она предполагает автоматизированную сборку, тестирование и доставку приложения в тестовую среду, а затем автоматическую доставку в боевую среду. Это позволяет быстро доставлять качественное и актуальное программное обеспечение, уменьшить время доставки и риски ошибок.

Использование **Continuous Integration** и **Continuous Delivery** позволяет улучшить качество программного обеспечения, ускорить его разработку и доставку, а также уменьшить количество ошибок и риски для бизнеса. Они являются важными инструментами в современной разработке программного обеспечения и широко применяются в индустрии.

CI/CD



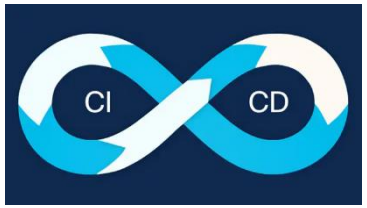
CI/CD и его среда обитания

Continuous Integration (CI) и **Continuous Delivery (CD)** играют важную роль в QA процессе, позволяя улучшить качество программного обеспечения и оптимизировать тестирование.

CI обеспечивает быструю обратную связь и позволяет быстро обнаруживать и исправлять ошибки в коде, что уменьшает количество ошибок и дефектов в приложении. Непрерывная интеграция позволяет автоматически собирать и запускать тесты на каждый новый коммит в репозитории. Это позволяет обнаруживать проблемы в коде на ранней стадии разработки и своевременно их исправлять.

CD автоматизирует процесс доставки приложения, что позволяет быстрее и чаще выпускать новые версии приложения в продакшн среду. Это также снижает количество ошибок, связанных с доставкой и установкой приложения.

CI/CD

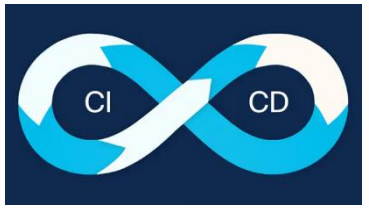


CI/CD и его среда обитания

CI и CD также позволяют повысить эффективность тестирования. Автоматическое тестирование при каждом коммите позволяет получать непрерывную обратную связь о качестве приложения, а автоматическая доставка в тестовую среду позволяет тестировщикам быстро тестировать новые функции и исправления.

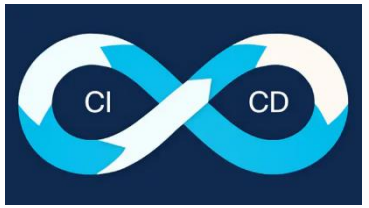
В итоге, использование CI и CD в QA процессе позволяет оптимизировать тестирование, улучшить качество программного обеспечения и снизить время доставки новых версий в продакшн среду.

CI/CD



CI/CD и его среда обитания

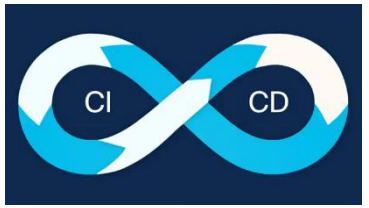
CI/CD



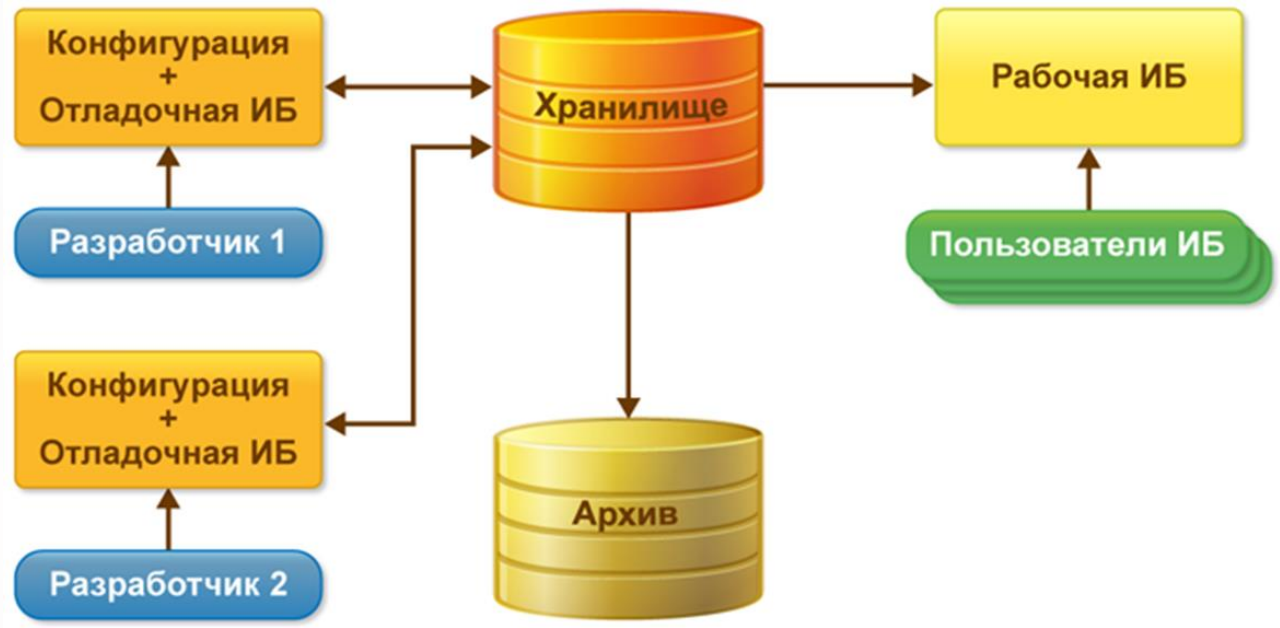
CI/CD и его среда обитания

Continuous Integration	Continuous Delivery	Continuous Deployment
Необходимы автотесты на каждую новую фичу или баг-фикс	Тестами должно быть покрыто достаточный объём кода	Качество тестов влияет на качество готового продукта
Из-за ранней регрессии меньше багов попадает на продакшен	Сборка должна быть автоматизирована, обновления должны распространяться вручную	Развёртывание автоматизировано и триггеры настроены на каждое изменение
Разработчики должны отправлять изменения как можно чаще (минимум раз в день)		

CI/CD



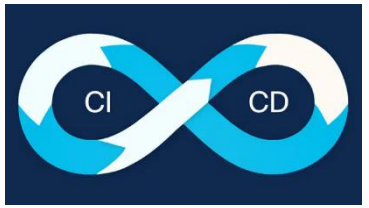
CI/CD и его среда обитания



CI/CD

GIT

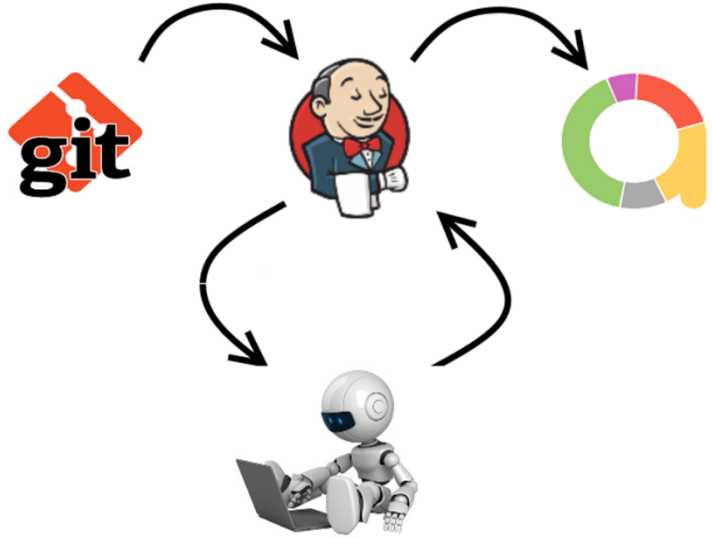
CI/CD



Jenkins

Jenkins - это популярный инструмент с открытым исходным кодом для реализации **Continuous Integration (CI) и Continuous Delivery (CD)** процессов. Он предоставляет автоматизированный пайплайн для разработки, тестирования и доставки программного обеспечения. **Jenkins** может интегрироваться с различными системами контроля версий, средами разработки и тестирования, и обеспечивает множество плагинов для расширения функциональности. Он позволяет быстро обнаруживать и исправлять ошибки в коде, автоматизировать сборку и тестирование приложений, а также управлять процессом доставки.

Jenkins имеет широкую базу пользователей и сообщества, которое активно разрабатывает новые плагины и расширения.



CI/CD

Jenkins

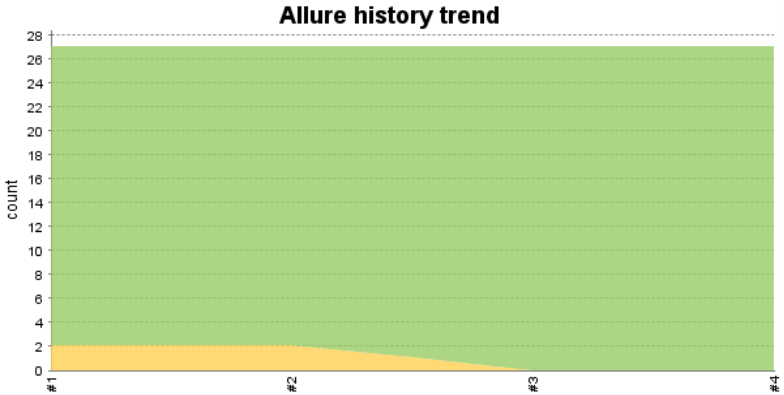
Last Successful Artifacts
allure-report.zip 1,00 MB посмотреть

Recent Changes

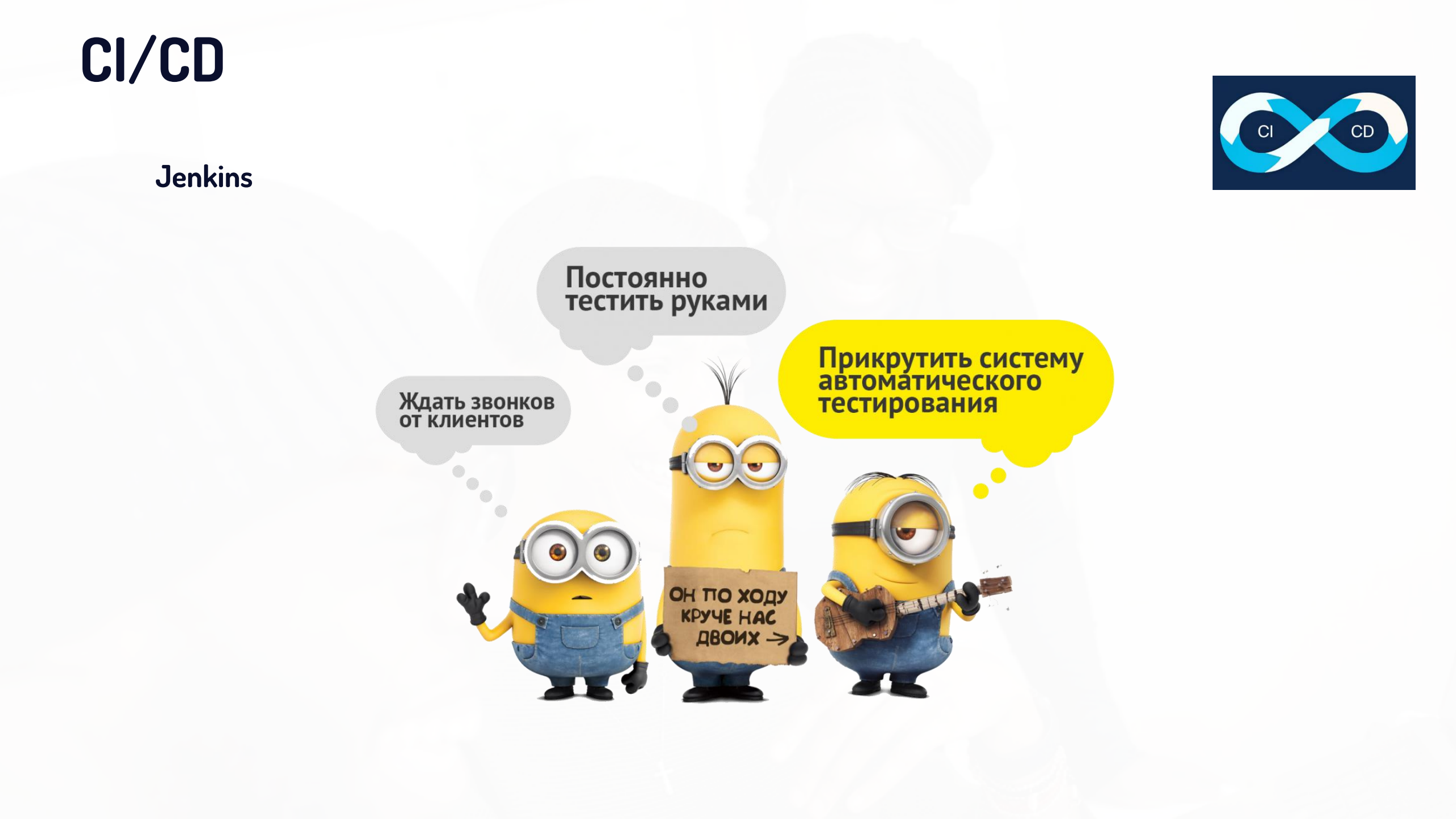
Stage View

Average stage times:
(Average full run time: ~4min 21s)

	Подготовка стенда	Автотестирование	Генерация документации	Анализ технического долга	Сборка	Деплой	Пост-деплой
	37s	3min 22s	63ms	54ms	52ms	52ms	51ms
#4 Oct 05 16:55 1 commit	35s	3min 18s	57ms	46ms	48ms	45ms	47ms
#3 Oct 05 12:44 8 commits	45s	3min 30s	52ms	47ms	49ms	47ms	53ms
#2 Oct 05 12:00 2 commits	34s	3min 18s	61ms	50ms	50ms		
#1 Oct 04 16:33 No Changes	32s	3min 24s	83ms	75ms	64ms		



CI/CD



Jenkins

Thanks for your time ☺