DevOps + docker

весна 2023

Вопросы к экзамену/зачёту/собеседованию

DevOps. Суть, назначение, этапы.

Docker. Суть, образ, управление.

CI/CD

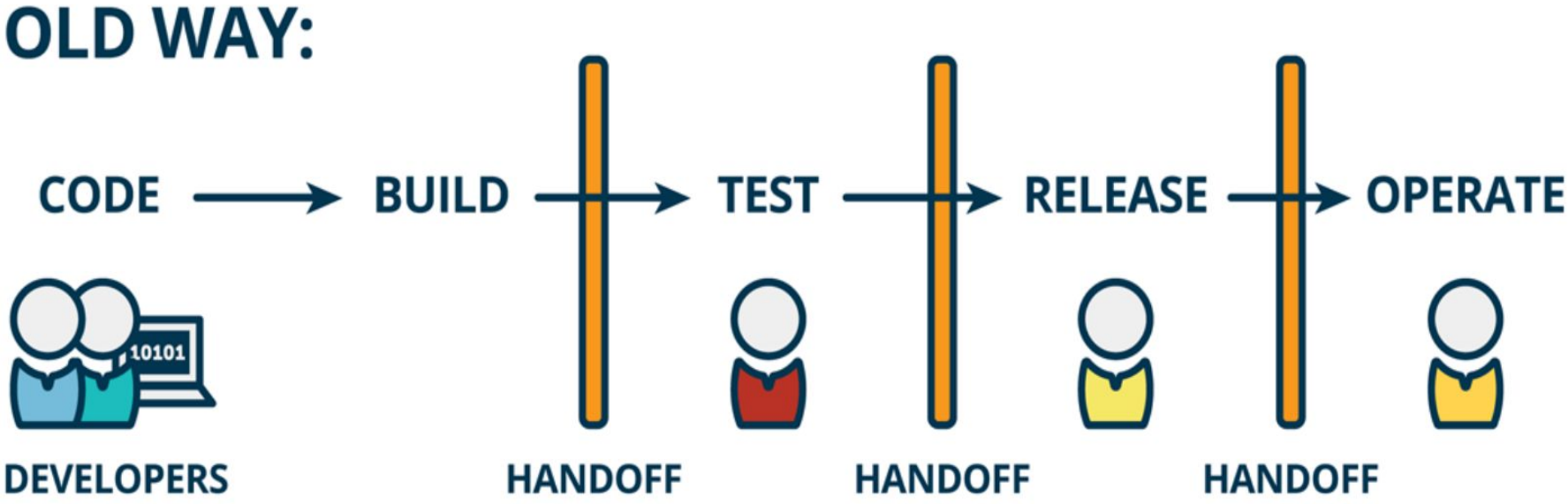
Сценарий рассказа

Рассмотрим параллельные ветки развития:

Управление разработкой

Техническая эксплуатация

Начало управления разработкой



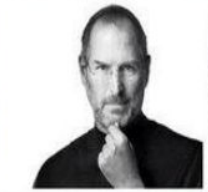
Проблемы в управлении разработкой

Конфликты между командами

Код выполняется редко

Поставка выполняется редко

При авариях/отказах требуется много времени на восстановление работоспособности

	DEVELOPERS	DESIGNERS	PROJECT MANAGERS	QA	SYSADMINS
SEEN BY DEVELOPERS					
SEEN BY DESIGNERS					
SEEN BY PROJECT MANAGERS					
SEEN BY QA					
SEEN BY SYSADMINS					

Начало технической эксплуатации

Эксплуатация информационных систем эволюционно развивалась по пути создания и установки приложений на какой-либо операционной системе.

Со временем количество установленного ПО и дополнительных настроек стало превышать все разумные пределы, а само ПО стало требовать разных настроек операционной системы, зависеть от разных версий системных библиотек и версий установленного на той же системе ПО.

Начало технической эксплуатации

Первоначальная настройка и подготовка к эксплуатации каждого отдельного сервера стала занимать дни и даже недели.

Любая ошибка в дальнейшей перенастройке сервера или обновлении софта могла приводить к неожиданным последствиям, вплоть до фатальных, требующих полной переустановки системы с нуля.

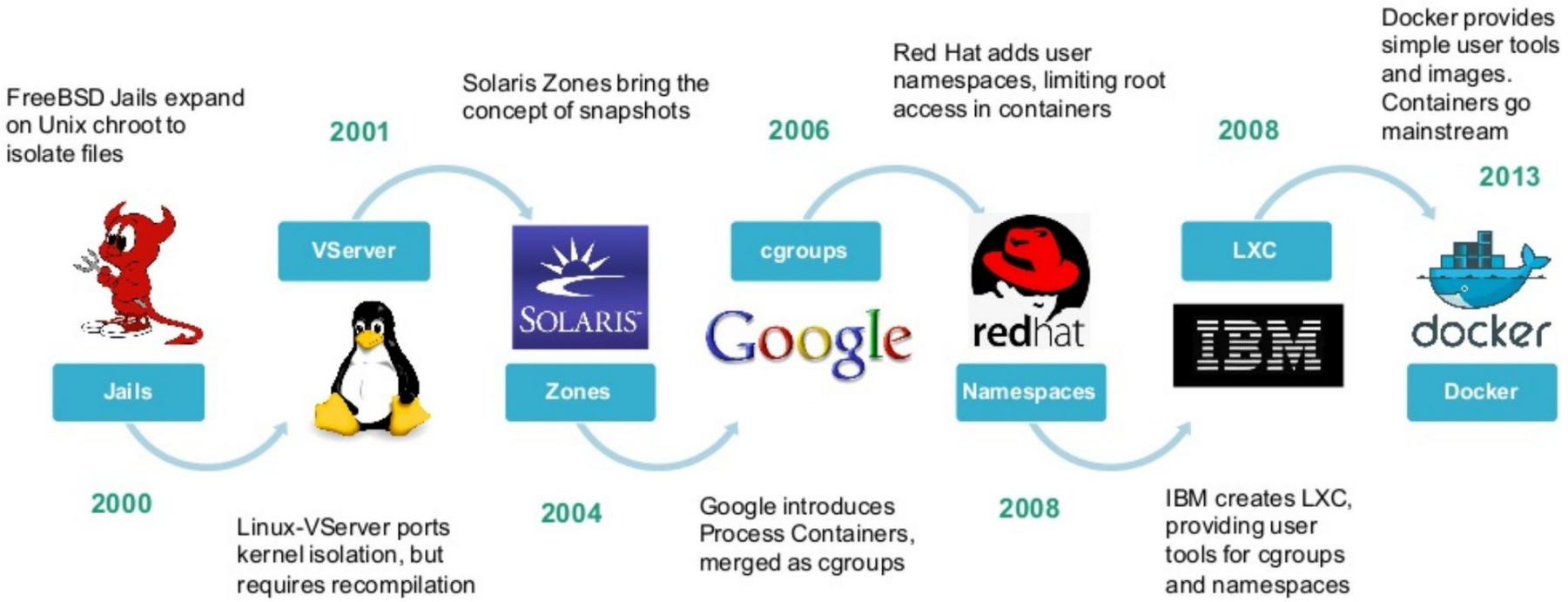
Проблемы технической эксплуатации

При запуске приложения на выделенном сервере:

- Зависимость от системных и пользовательских библиотек
- Сложные обновления ПО
- Долгая первоначальная настройка сервера к эксплуатации
- Долгая перенастройка сервера

Сообщество, работающее с unix-подобными системами, стало изобретать различные инструменты изоляции запускаемых приложений в своих собственных средах исполнения.

Эволюция технической эксплуатации



Docker

В 2008 на сцену вышла компания dotCloud (позже переименовалась в Docker) с технологией docker, объединившей достижения LXC с продвинутыми инструментами для разработчиков и еще больше облегчившей использование контейнеров.

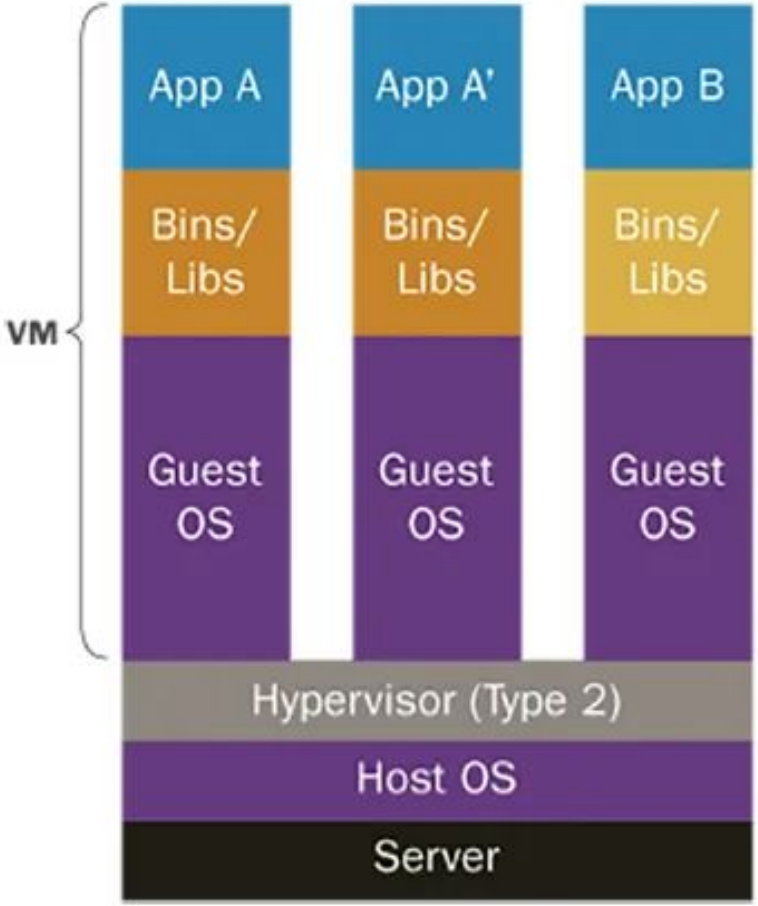
Сегодня технология с открытым кодом Docker – это наиболее известный и популярный инструментарий развертывания и управления Linux-контейнерами.

Контейнеры

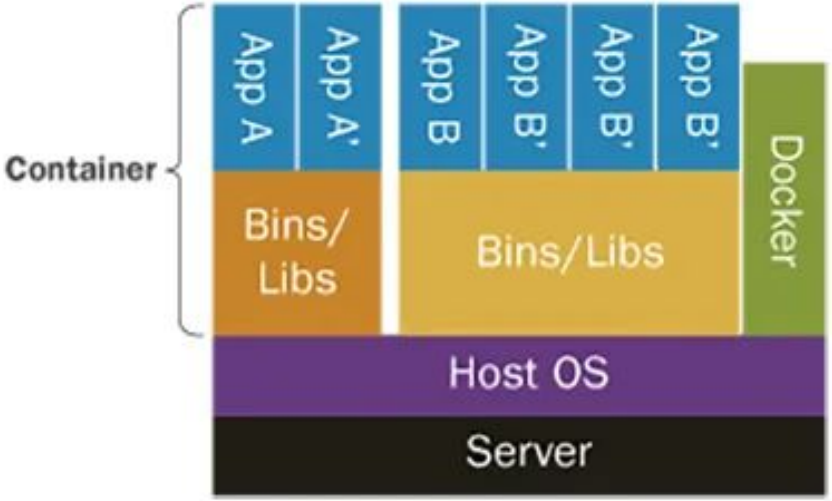
Контейнер – это набор процессов, изолированный от остальной операционной системы (хоста) и запускаемый на основе отдельного образа, который содержит все файлы, необходимые для их работы.

Образ содержит все зависимости приложения, и поэтому может легко переноситься из среды разработки в среду тестирования, а затем в промышленную среду.

Containers vs. VMs



Containers are isolated, but share OS and, where appropriate, bins/libraries



Контейнеры

Контейнеры используют одно и то же ядро операционной системы и изолируют процессы приложения от остальной системы.

Для одновременной работы нескольких операционных систем на одном гипервизоре, требуется существенно больше системных ресурсов, чем для аналогичной конфигурации на основе контейнеров.

Docker обеспечивает инструментарий создания и запуска контейнеров.

Образ контейнера

это файл, который используется для запуска экземпляра контейнера.

Образы контейнеров хранятся в реестрах. Самый популярный реестр - DockerHub (https://hub.docker.com/), с которого часто скачиваются все базовые контейнеры операционных систем.

Образы docker состоят из нескольких слоев, и каждый слой - это файловая система, доступная только для чтения.

Образ контейнера

Каждый слой - это отдельный файл, который хранится на машине, где запускаются контейнеры.

Одни и те же слои могут использоваться разными контейнерами.

Каждая инструкции docker-файла (Dockerfile) создаёт свой слой, который размещается поверх предыдущих слоев.

docker

Обеспечивая переносимость и управление версиями, образ контейнера гарантирует, что если приложение работает на компьютере разработчика, то оно будет работать и в промышленной среде.

Требуя меньше системных ресурсов по сравнению с виртуальной машиной, Linux-контейнер почти не уступает ей в возможностях изоляции и значительно облегчает сопровождение составных многоуровневых приложений.

docker daemon

docker daemon запускается на хост-машине. Обычно его запуском занимается операционная система хост-машины. Демон отвечает за создание и запуск контейнеров, за контроль того, как контейнеры работают, за создание и хранение образов.

docker daemon предоставляет HTTP API, с ним можно работать с помощью клиента, например, через командную строку или графический интерфейс.

docker

docker client = cli-интерфейс, который общается с docker daemon.

Вызывая команды запуска контейнера, создания образа или другие, мы управляем демоном с помощью клиента.

docker registry - это сервис, который используется для хранения и распространения образов. В целях безопасности компании размещают свой реестр на инфраструктуре своей компании.

docker compose

Когда мы разрабатываем достаточно сложное приложение, нам приходится иметь дело с зависимостями и подключаемыми ресурсами: базой данных, очередями, хранилищами и другими механизмами.

docker compose позволяет “скомпоновать” несколько контейнеров в один запуск; запускаемые контейнеры можно описать в конфигурационном файле и не нужно помнить все тонкие параметры, которые нужно задать.

Kubernetes (k8s)

оркестратор контейнеров для создания распределенных отказоустойчивых высокопроизводительных систем с избыточным дублированием мощностей.

https://kubernetes.io/

Kubernetes (k8s)

Мониторинг сервисов и распределение нагрузки

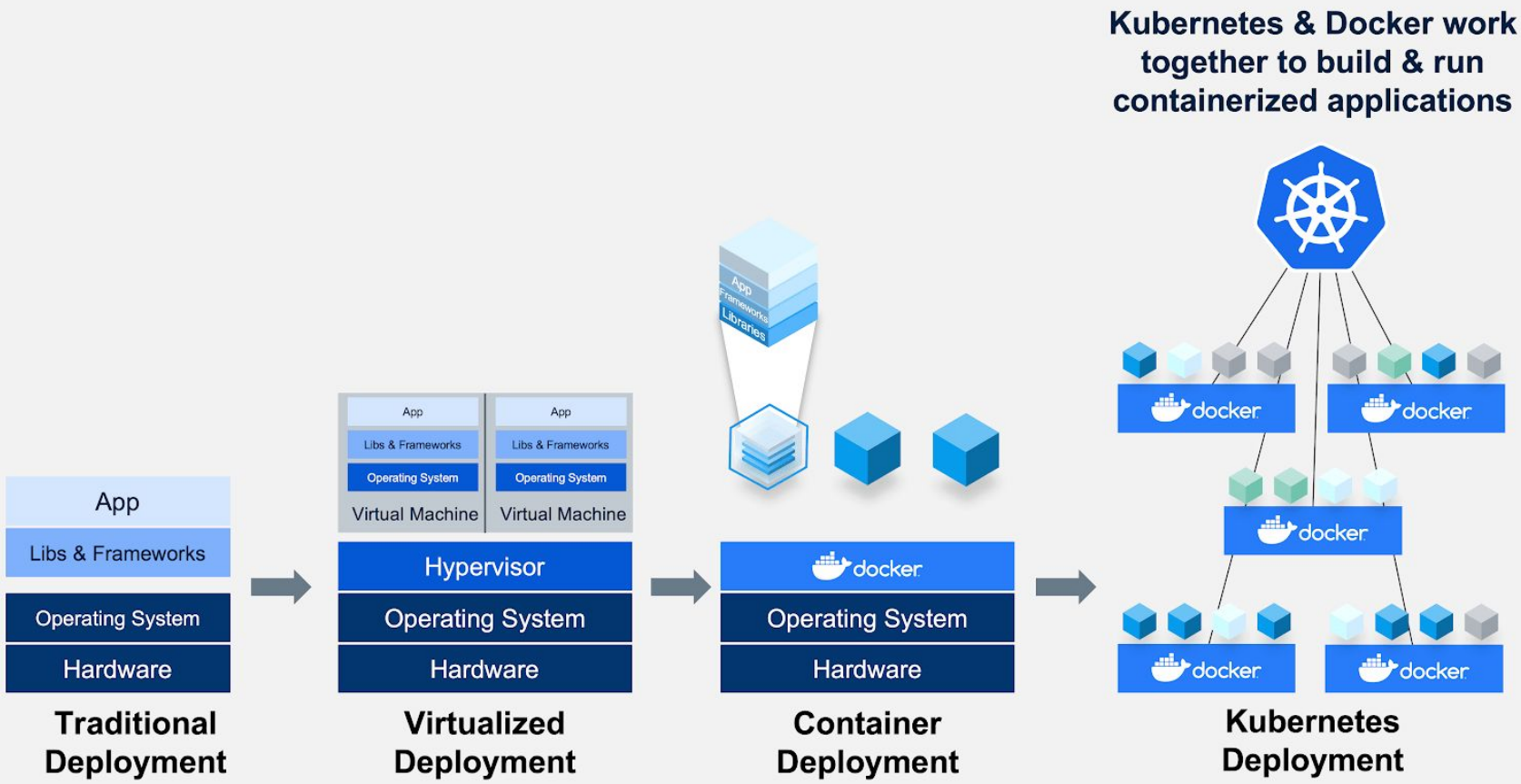
Оркестрацию хранилища

Автоматические развертывания и откаты

Автоматическое распределение нагрузки

Самоконтроль

Управление конфиденциальной информацией и конфигурацией


Kubernetes & Docker work together to build & run containerized applications
App
Libs & Frameworks
Operating System
Hardware
Traditional Deployment
App
Libs & Frameworks
Operating System
Virtual Machine
App
Libs & Frameworks
Operating System
Virtual Machine
Hypervisor
Operating System
Hardware
Virtualized Deployment
App
Frameworks
Libraries
docker
Operating System
Hardware
Container Deployment
docker
docker
docker
docker
docker
Kubernetes Deployment

Вопрос к аудитории

Вернёмся к теме управления разработкой.

Что такое DevOps?

Определение DevOps

DevOps – набор практик в области разработки ПО, направленных на усиление взаимодействия между разработчиками и другими IT-специалистами с целью повышения эффективности разработки и доставки клиенту новых версий ПО

DevOps = сочетание разработки (Dev) и эксплуатации (Ops)


Разработка
Тестирование
DevOps
Эксплуатация

Определение DevOps

DevOps – методология автоматизации технологических процессов сборки, настройки и развёртывания программного обеспечения

Назначение DevOps

позволяет представителям ранее разрозненных подразделений — разработки, ИТ-операций, обеспечения качества и безопасности — координировать свои действия и совместно создавать более качественные и надежные продукты

Назначение DevOps

Внедрив культуру DevOps вместе с соответствующими методиками и инструментами, команды получают возможность лучше реагировать на потребности клиентов, повышать доверие к создаваемым приложениям и быстрее достигать бизнес-целей.

Назначение DevOps

Команды DevOps сохраняют гибкость, выпуская программное обеспечение короткими циклами.

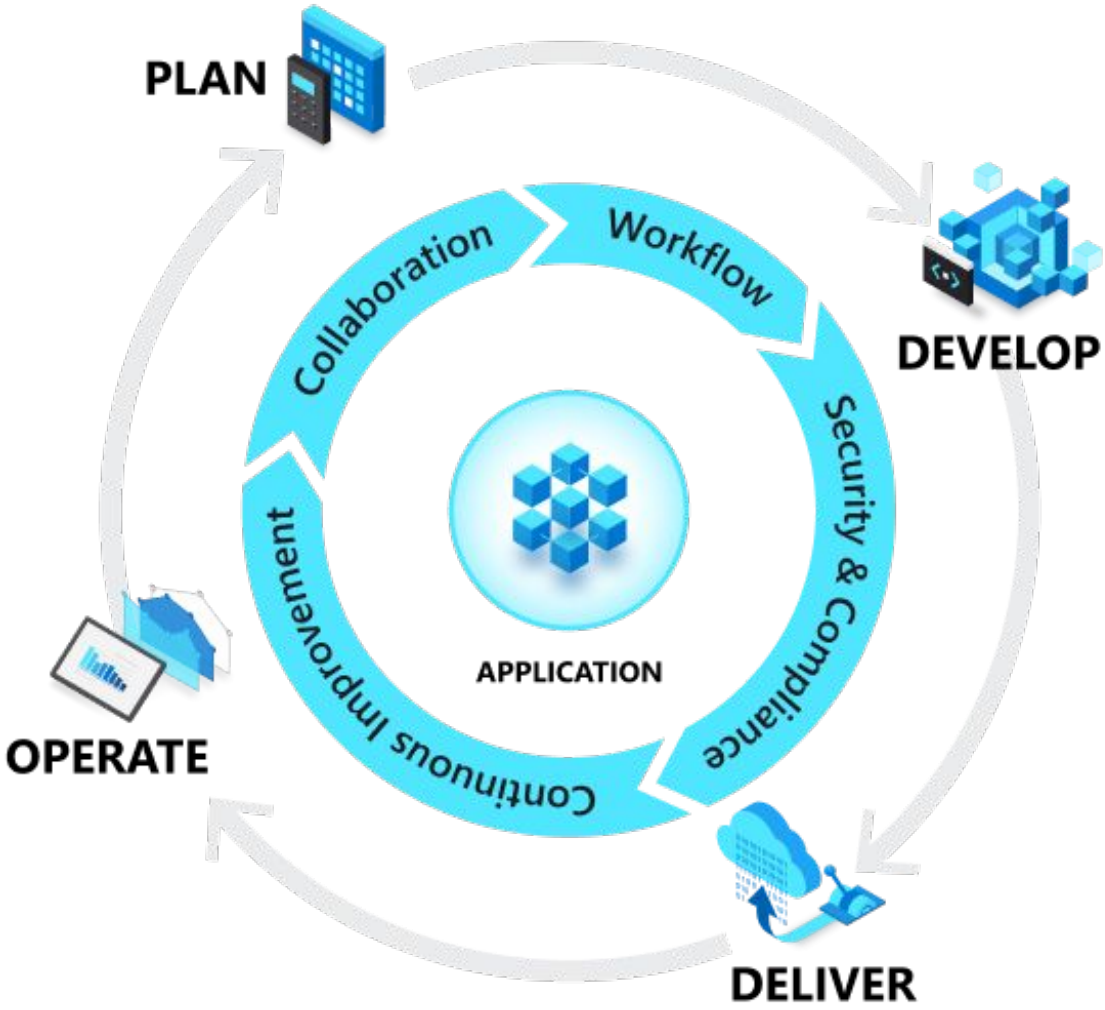
Более короткие циклы выпуска упрощают планирование и управление рисками.

Сокращение циклов выпуска также позволяет организациям адаптироваться к изменяющимся потребностям клиентов и давлению со стороны конкурентов и принимать соответствующие меры.

Назначение DevOps

Призваны обеспечить:

- Быстрый выход продуктов на рынок
- Адаптацию к условиям рынка в конкурентной среде
- Поддержание стабильности и надежности систем
- Сокращение среднего времени восстановления


PLAN
DEVELOP
DELIVER
OPERATE
Collaboration
Workflow
Security & Compliance
Continuous Improvement
APPLICATION

Этапы DevOps. Планирование

На этапе планирования команды представляют, определяют и описывают функции и возможности создаваемых приложений и систем.

Отслеживают ход работы.

Создание журналов невыполненной работы (backlog и технический долг), отслеживание ошибок.

Scrum

Kanban

Этапы DevOps. Разработка

Написание, тестирование, проверка и интеграция кода.

Команды стремятся быстро внедрять инновации без ущерба для качества, стабильности и продуктивности. Для этого они используют высокопроизводительные инструменты, автоматизируют рутинные и выполняемые вручную действия, а также запускают итерации с небольшим шагом при помощи автоматического тестирования и непрерывной интеграции.

Непрерывная интеграция (CI = Continuous integration)

Выполнение частых автоматизированных сборок проекта для скорейшего выявления и решения проблем интеграции.

Обычно интеграция может непредсказуемо задержать окончание работ.

Переход к непрерывной интеграции позволяет снизить трудоемкость процесса интеграции, сделав его более предсказуемым за счет наиболее раннего обнаружения и устранения ошибок и противоречий.

Непрерывная интеграция

Линтер – утилита для статического анализа кода с целью обнаружения стилистических и программных ошибок.

Задачи линтеров:

- Форматирование кода
- Поиск потенциальных проблем
- Оптимизация производительности

Автоматическое тестирование

Модульные тесты (unit)

Интеграционные тесты

Сквозные тесты (end-to-end)

Этапы DevOps. Развёртывание (deploy)

Процесс последовательного и надежного развертывания приложений в рабочих средах, настройки полностью управляемой базовой инфраструктуры.

На этапе доставки команды определяют процесс управления выпусками, четко указывая этапы, требующие утверждения вручную; устанавливают автоматические шлюзы, с помощью которых приложения перемещаются между этапами, пока не станут доступны клиентам.

Этапы DevOps. Развёртывание (deploy)

Автоматизация этих процессов делает их масштабируемыми, воспроизводимыми и контролируемыми.

Управление конфигурацией на этом этапе позволяет составить формализованное описание состояния управляемого узла. Система сама определяет, что нужно сделать для достижения этого состояния, и осуществляет все необходимые действия. Это позволяет быстро и контролируемо осуществлять переконфигурирование системы путем изменения описания конфигурации.

Этапы DevOps. Эксплуатация (operations)

Обслуживание, мониторинг и устранение неполадок приложений.

Обеспечение надёжности, высокой доступности.

Снижение простоев.

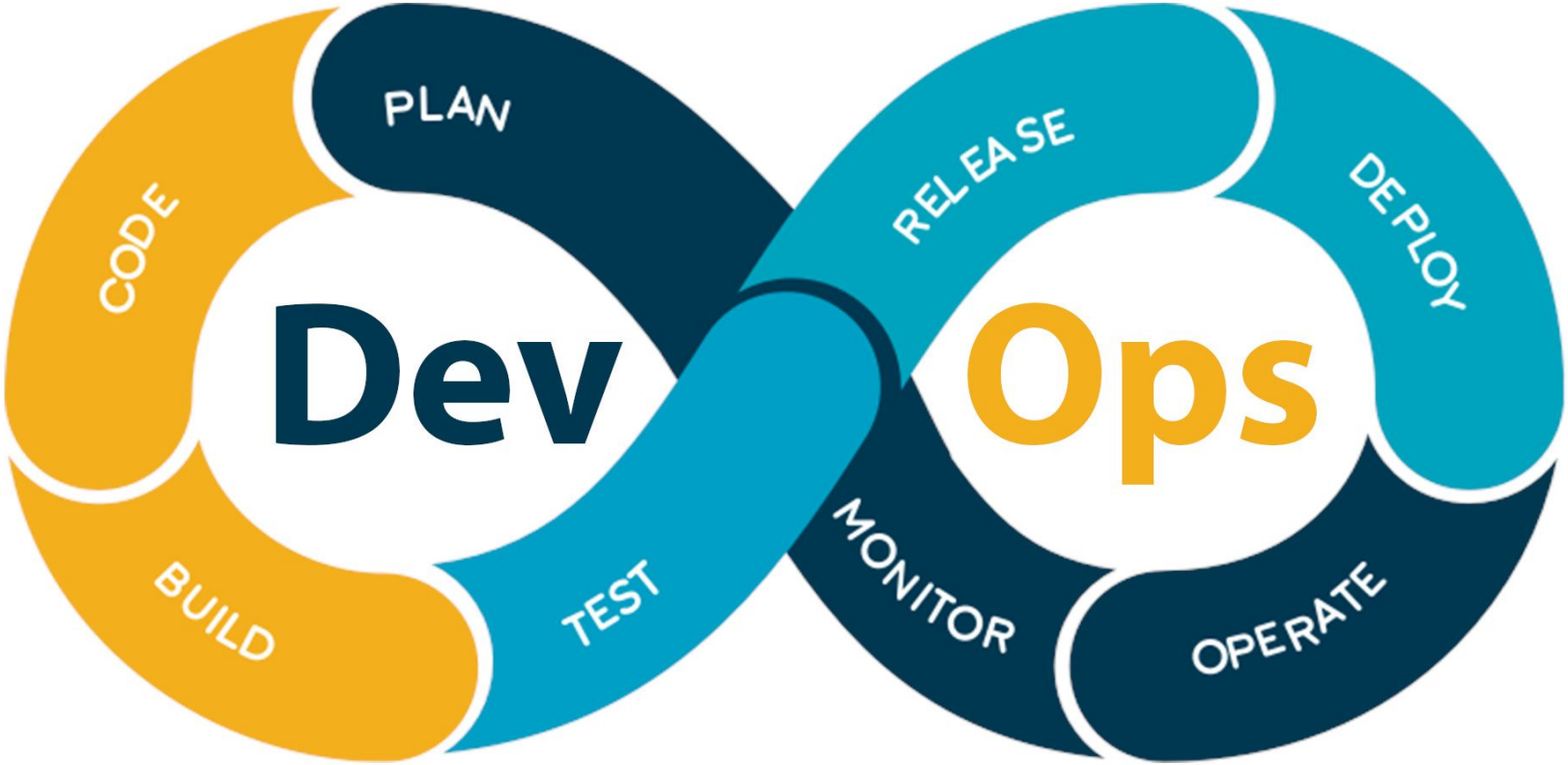
Повышение уровня безопасности.

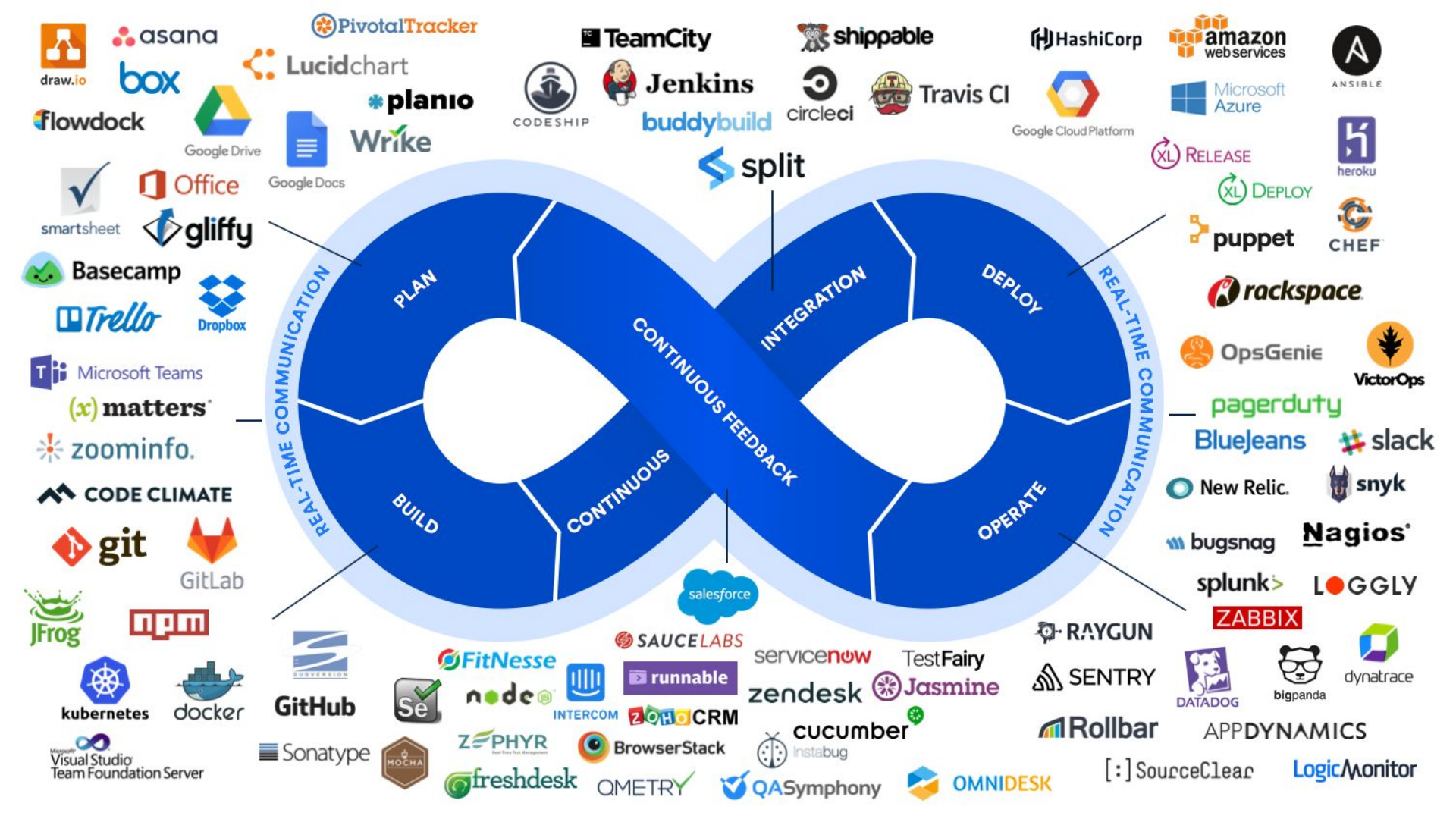
Этапы DevOps. Эксплуатация (operations)

Выявление проблем прежде, чем они повлияют на качество обслуживания клиентов.

Оперативное устранение проблем при возникновении.

Необходимы широкие возможности телеметрии, функциональных оповещений и полная прозрачность приложений и базовой системы.


PLAN
CODE
BUILD
TEST
RELEASE
DEPLOY
MONITOR
OPERATE
Dev
Ops


PLAN
CONTINUOUS
BUILD
INTEGRATION
DEPLOY
OPERATE
CONTINUOUS FEEDBACK
REAL-TIME COMMUNICATION
REAL-TIME COMMUNICATION
draw.io
asana
PivotalTracker
box
Lucidchart
planio
flowdock
Google Drive
Google Docs
Wrike
Office
smartsheet
gliffy
Basecamp
Trello
Dropbox
Microsoft Teams
(x) matters
zoominfo.
CODE CLIMATE
git
GitLab
JFrog
npm
kubernetes
docker
Visual Studio Team Foundation Server
GitHub
Sonatype
CODESHIP
TeamCity
Jenkins
buddybuild
shippable
circleci
Travis CI
split
salesforce
SAUCELABS
FitNesse
node
INTERCOM
runnable
ZOHO CRM
ZEPHYR
BrowserStack
freshdesk
QMETRY
QASymphony
cucumber
Instabug
OMNIDESK
servicenow
TestFairy
zendesk
Jasmine
HashiCorp
amazon web services
Microsoft Azure
Google Cloud Platform
ANSIBLE
RELEASE
DEPLOY
heroku
puppet
CHEF
rackspace
OpsGenie
VictorOps
pagerduty
BlueJeans
slack
New Relic.
snyk
bugsnag
Nagios
splunk>
LOGGLY
ZABBIX
RAYGUN
SENTRY
DATADOG
bigpanda
dynatrace
Rollbar
APPDYNAMICS
[:] SourceClear
LogicMonitor

Build an Open-Source DevOps Toolchain

Repository:
- JFrog Artifactory
- Sonatype Nexus OSS
- Git (on-premises)
- GitLab CE
- Bit Component Repo

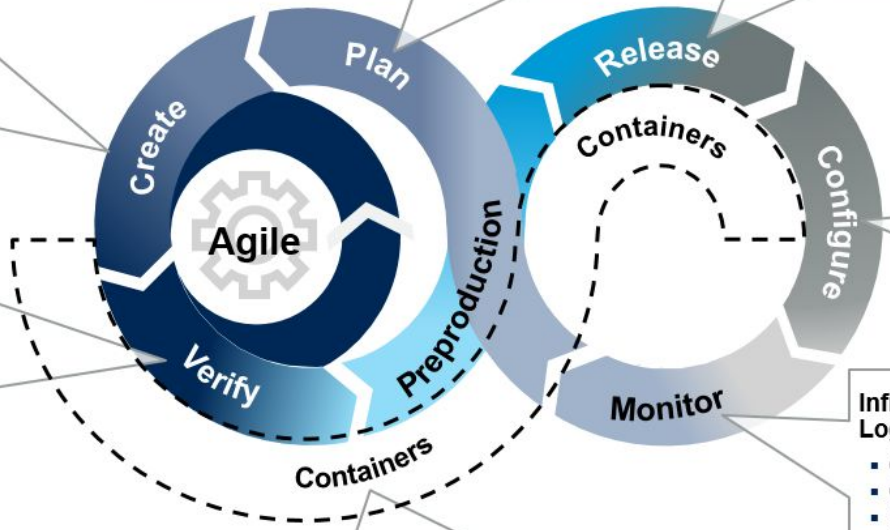
Build/Integration:
- Jenkins
- CloudBees (Jenkins)
- Apache Maven
- Gradle
- CruiseControl
- Travis CI
- XebiaLabs (Hudson)

IDE:
- Eclipse
- CircleCI
- Electric Cloud
- Visual Studio Code
- JetBrains IntelliJ

Task Management:
- Taiga
- TaskBoard
- Pencil
- Inkscape
- OpenProject
- XWiki

BDD:
- Behat
- Concordian
- Cucumber
- Jbehave
- Jasmine
- SpecFlow

Release, Deploy and Coordination:
- JFrog Bintray
- OpenMake DeployHub
- Hashicorp TerraForm
- ElectricFlow
- Capistrano
- Spinnaker
- GoCD



Continuous Configuration Automation
- CFEngine
- Chef
- Inedo Otter
- Puppet Labs
- Red Hat (Ansible Tower)
- SaltStack
- StackStorm

Testing:
- Appium
- Jmeter
- Protractor
- Selenium
- Visual Studio Test Platform
- Watir

Security:
- ChaoSlingr
- Hashicorp Vault
- OWASP ZAP
- OSSEC

Bug Tracking:
- Mantis
- Bugzilla
- Redmine
- Trac

Code Review:
- Gerrit
- Review Board
- Sonarqube

Compliance Verification:
- Inspec
- Compliance Masonry

- Docker
- Canonical LXD
- CoreOS
- CentOS
- Kubernetes
- Vagrant
- Mesos
- Nomad

Infrastructure, APM and Analytics, and Log Management:
- Cacti
- Caliper
- ELK Stack
- Ganglia
- Graphite
- Hyperic
- Hawtio
- Heapster InfluxData
- Nagios
- Prometheus
- Wireshark
- Zabbix
- Zenoss Core
- Fluentd
- Collectd
- Sematext Logagent
- Apache ELK

Docker и DevOps

Контейнеры – это отличный инструмент для непрерывной интеграции и доставки приложений заказчикам

CI/CD

Continuous integration – частый push изменений и автоматическая проверка работоспособности приложения

Continuous delivery – непрерывная доставка ресурсов продукта, готовых к развертыванию

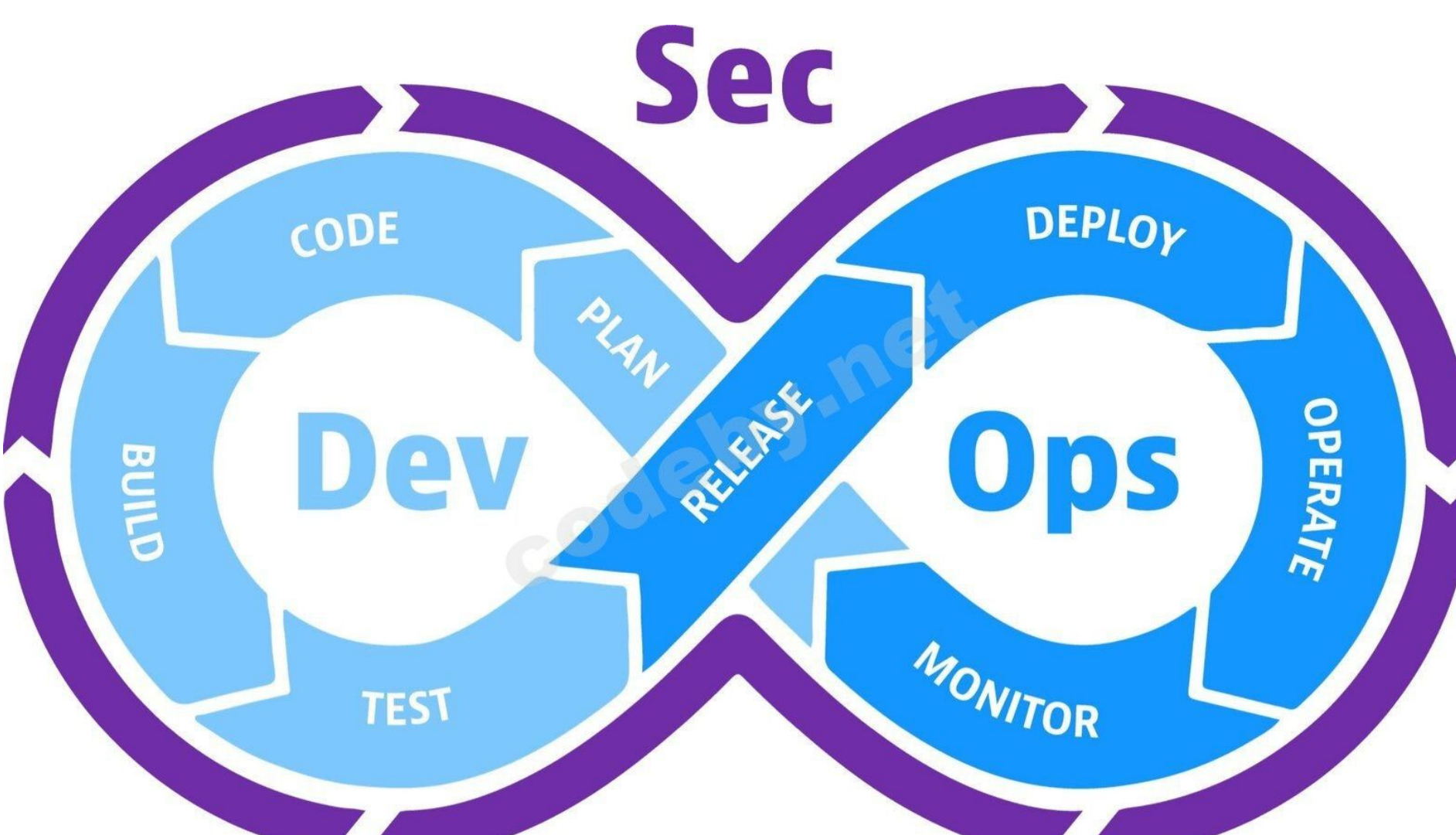
Continuous deployment – непрерывное развертывание, автоматическое развертывание новой версии продукта (не требует ручного управления)

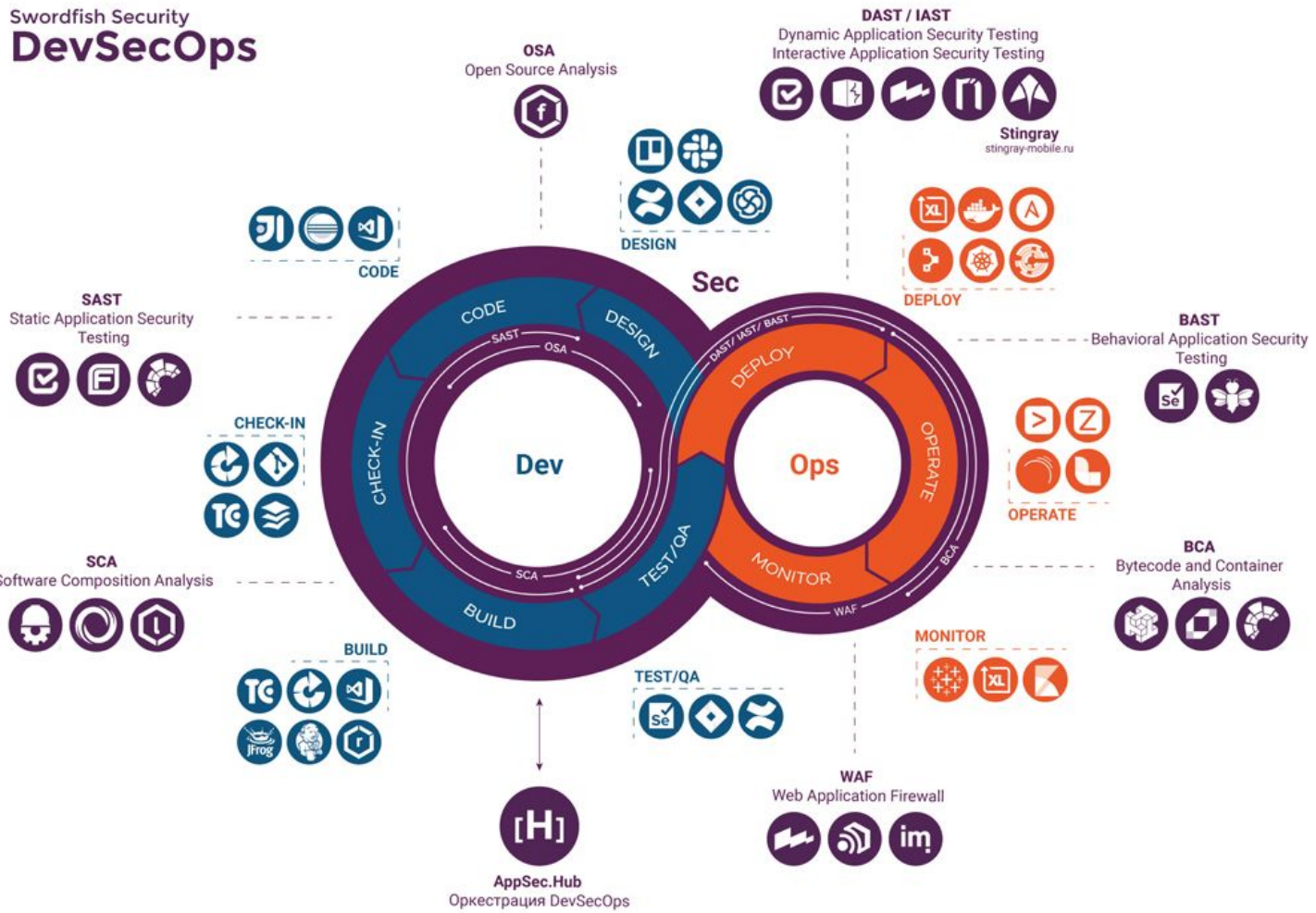
Вопросы к экзамену/зачёту/собеседованию

DevOps. Суть, назначение, этапы.

Docker. Суть, образ, управление.

CI/CD


Sec
CODE
PLAN
BUILD
Dev
TEST
RELEASE
DEPLOY
OPERATE
Ops
MONITOR


Swordfish Security
DevSecOps
OSA
Open Source Analysis
DAST / IAST
Dynamic Application Security Testing
Interactive Application Security Testing
Stingray
stingray-mobile.ru
DESIGN
CODE
DEPLOY
SAST
Static Application Security Testing
Sec
BAST
Behavioral Application Security Testing
CHECK-IN
Dev
Ops
OPERATE
SCA
Software Composition Analysis
BCA
Bytecode and Container Analysis
BUILD
MONITOR
TEST/QA
WAF
Web Application Firewall
AppSec.Hub
Оркестрация DevSecOps