

DevOps

DevOps - это культура и методология, способствующая автоматизации процессов разработки, тестирования и развертывания программного обеспечения. Она обеспечивает более эффективное взаимодействие между разработчиками и специалистами по обслуживанию для улучшения скорости и стабильности поставки продуктов.



by Илья Исаев

План курса:

- 1) Введение в DevOps
- 2) Configuration Management
- 3) GitOps
- 4) CI/CD
- 5) Docker
- 6) Observability
- 7) Kubernetes

4 групповых лабораторных работы

Литература по DevOps

- 1) The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations, 2nd Edition - Авторы: Gene Kim, Jez Humble, Patrick Debois, John Allspaw, and John Willis.
- 2) The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win - Автор: Gene Kim.
- 3) The Unicorn Project: A Novel about Developers, Digital Disruption, and Thriving in the Age of Data - Автор: Gene Kim.
- 4) The DevOps Adoption Playbook: A Guide to Adopting DevOps in a Multi-Speed IT Enterprise - Автор: Sanjeev Sharma.
- 5) Accelerate: The Science of Lean Software and Devops: Building and Scaling High Performing Technology Organizations - Авторы: Nicole Forsgren, Jez Humble, and Gene Kim.

DevOps-инженер

DevOps-инженер - это специалист, который объединяет разработку и эксплуатацию программного обеспечения. Он отвечает за создание и поддержку инфраструктуры, автоматизацию процессов разработки и внедрения, а также за обеспечение высокой доступности и надёжности системы.

Роль DevOps-инженера становится все более востребованной в современных организациях, так как он помогает сократить время разработки и доставки продуктов на рынок, повысить качество разработки и улучшить коммуникацию и сотрудничество между различными командами.

Основные задачи DevOps-инженера включают:

1. Настройка и поддержка инфраструктуры: установка и конфигурация серверов, сетей и других средств разработки.
2. Автоматизация процессов: создание скриптов и инструментов для автоматической сборки, тестирования и развертывания приложений.
3. Обеспечение надежности и безопасности: мониторинг и анализ системы, резервное копирование данных, обеспечение защиты от внешних угроз.
4. Сотрудничество с различными командами: работа с разработчиками, тестировщиками, администраторами систем и другими участниками проекта для достижения общих целей.

DevOps-инженер зарплаты

Зарплата DevOps-инженера в России может варьироваться

На Хабр Карьера указана средняя зарплата DevOps-инженера в размере 225 000 рублей

DreamJob.ru утверждает, что средняя зарплата на должности DevOps-инженера в России составляет около 160,000 рублей

Вилки примерные 200 000 - 500 000 рублей

Точная зарплата может зависеть от местоположения, опыта и квалификации и грейда в должности.

История возникновения DevOps

1

Автоматизация и Agile

DevOps возник как ответ на необходимость автоматизации процессов разработки под влиянием принципов Agile (Time to Market)

2

Сближение Dev и Ops

Тенденция к укрупнению команд и сближению разработчиков и администраторов Обеспечения.

3

Рост облачных технологий

Рост популярности облачных технологий потребовал новых подходов к совместной работе команд.

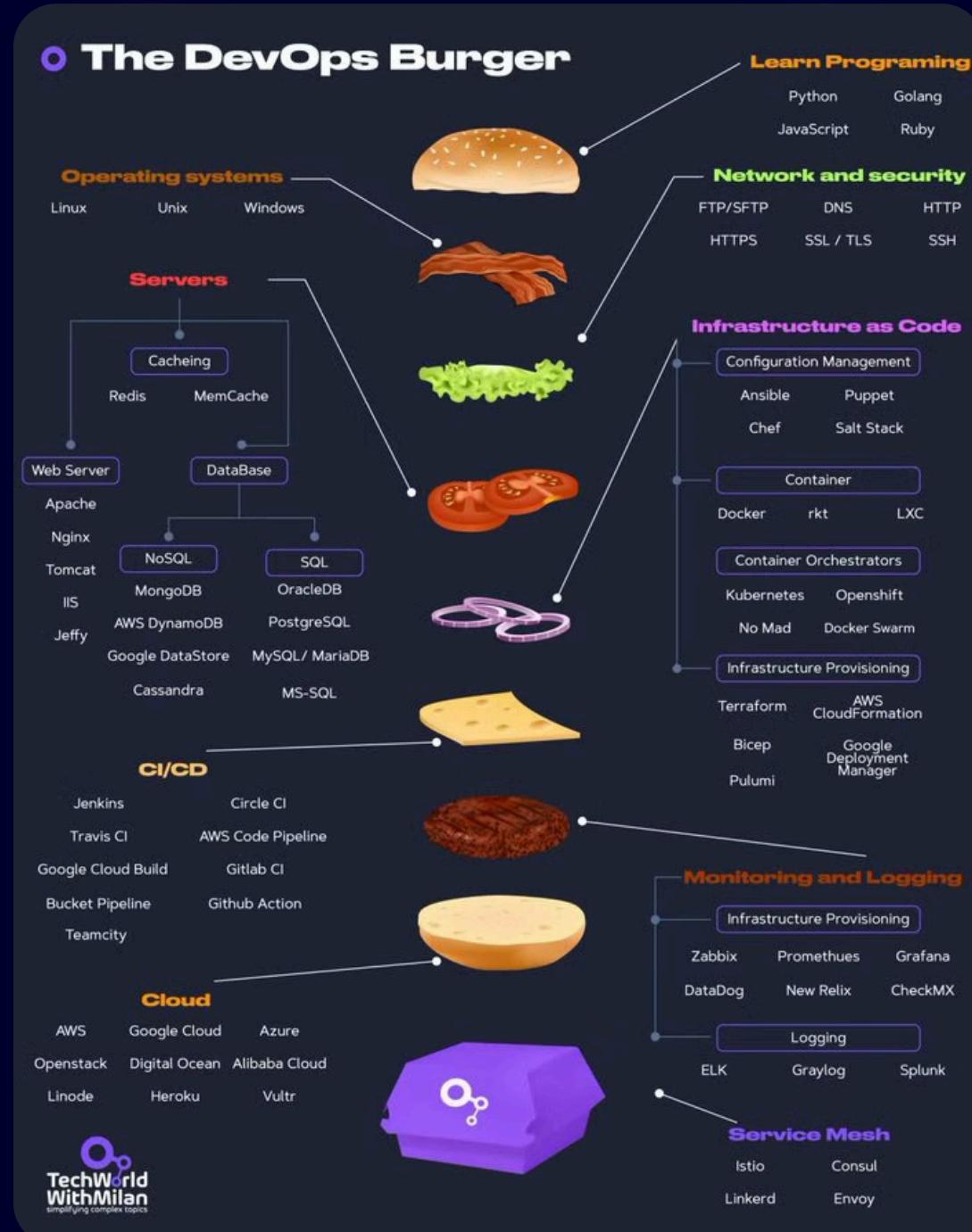


Made with Gamma

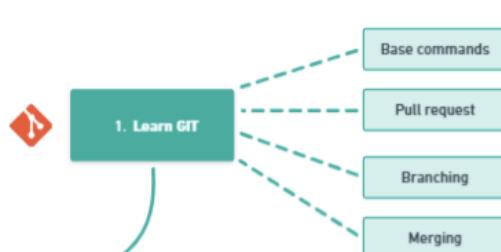
Кем работать?

- Разработчик бэк/фулстек
- Сетевой инженер
- Инженер по безопасности
- Тестировщик
- Аналитик/ML
- Архитектор сервисов
- Менеджер проектов

Roadmap



DevOps Roadmap 2025.



[GitHub - milanm/DevOps-Roadmap: DevOps Roadmap for 2...](#)

DevOps Roadmap for 2025. with learning resources. Contribute to milanm/DevOps-Roadmap development by creating an account on...



Основные концепции DevOps

- GitOps
- Infrastructure as Code
- ChatOps
- Continues Integration/Continuous Delivery (автоматически)/Continuous Deployment (ручное) (CI/CD) (Unleash)
- Monitoring (Zabbix)
- Tracing (Jaeger)
- QA + Security (SecOps)

DevOps vs SRE инженер

DevOps и SRE (Site Reliability Engineering) инженеры имеют различные фокусы и задачи.

DevOps сосредотачивается на непрерывности и скорости разработки продукта, в то время как основное внимание SRE уделяется надёжности системы, её масштабируемости и доступности.

DevOps работает над устранением организационных барьеров, которые мешают сотрудничеству между функциями разработки и операций, в то время как SRE стремится создать масштабируемые и надёжные системы, обеспечивающие максимальную надёжность.

Важно отметить, что у DevOps и SRE есть перекрывающиеся области ответственности, но существует широкое разделение функций между ними.

DevOps инструменты



Docker

Основное средство контейнеризации для упаковки, доставки и запуска приложений.



Jenkins

Инструмент для автоматизации различных этапов разработки, тестирования и развертывания ПО.



Ansible

Инструмент для автоматизации управления конфигурациями и развертывания инфраструктуры.



Git как инструмент для DevOps

Процесс разработки

Git упрощает совместную работу разработчиков и управление версиями кода.

Контроль доступа

Git позволяет управлять доступом к репозиториям и контролировать права пользователей.

Непрерывная интеграция

Git интегрируется с инструментами CI/CD для автоматической сборки и тестирования кода.



Использование Docker

Простота в использовании

Docker позволяет легко создавать, развертывать и управлять контейнерами.

Портабельность

Контейнеры Docker могут быть запущены на различных платформах без изменений.

Масштабируемость

Докер обеспечивает горизонтальное масштабирование с помощью оркестрации контейнеров.



Мониторинг, профилирование, логирование (Grafana, Chronograf)

Мониторинг, профилирование и логирование являются важными инструментами в области DevOps. Они позволяют отслеживать и анализировать работу приложений, выявлять проблемы и оптимизировать их производительность и надежность.

Мониторинг представляет собой процесс наблюдения за работой системы и сбора различных метрик, таких как загрузка сервера, использование ресурсов, время отклика и другие. Это помогает оперативно реагировать на проблемы и предотвращать возможные сбои в работе приложений.

Профилирование позволяет анализировать производительность приложения и идентифицировать узкие места или проблемные участки кода. С помощью профилирования можно оптимизировать работу приложения, улучшить его отклик и снизить потребление ресурсов.

Логирование представляет собой запись и анализ журналов работы приложения. Логи позволяют отслеживать действия и события в системе, выявлять ошибки и проблемы, а также проводить аудит и анализ производительности.

Infrastructure as Code (Terraform, Ansible, Puppet)

Infrastructure as Code (IaC) - это методология, которая позволяет создавать, управлять и автоматизировать инфраструктуру с использованием кода

Этот подход позволяет развертывать и масштабировать инфраструктуру более эффективно и надежно, а также обеспечивает возможность контролировать весь процесс и отменять изменения при необходимости

Использование IaC позволяет ускорить процесс развертывания и управления инфраструктурой, а также облегчает сопровождение и обновление системы

Основные принципы IaC включают автоматизацию, версионирование, документирование и тестирование инфраструктурного кода

Важно отметить, что IaC является ключевым элементом для эффективных инженерных сетапов и способствует улучшению процессов разработки и обслуживания инфраструктуры

RESTfull приложение

Для создания простого проекта Spring Boot приложения с одним REST контроллером, вам следует выполнить следующие шаги:

- 1) Создайте новый проект Spring Boot с помощью Spring Initializr (<https://start.spring.io/>), выбрав необходимые зависимости (например, Spring Web).
- 2) Создайте класс контроллера в вашем проекте. Ниже приведен пример простого REST контроллера:

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class BestController {

    @GetMapping("/student")
    public String getStudent() {
        return "Привет, я всё сдам!";
    }
}
```

- 3) Запустите ваше Spring Boot приложение. После запуска вы сможете обратиться к эндпоинту /student и получить ответ "Привет, я всё сдам!".

Это простой пример Spring Boot приложения с одним REST контроллером. Вы можете дополнить его другими функциями и настройками в зависимости от ваших потребностей.



Упаковка Spring Boot приложения в Docker

Упаковка приложения в контейнер Docker - это один из ключевых шагов в DevOps. Мы можем использовать скрипты упаковки для автоматизации процесса создания образов Docker и развертывания приложений в среде production. Spring Boot приложения могут быть легко упакованы и запущены в Docker-контейнерах.

Пример скрипта

Для упаковки простого Spring Boot приложения в Docker с использованием файла docker-compose.yml, вы можете создать следующий пример:

```
version: '3.8'

services:
  spring-boot-app:
    image: your-spring-boot-image
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "8080:8080"
```

Пример скрипта

В этом примере:

- `version: '3.8'` указывает на версию `docker-compose.yml`.
- `services` определяет сервисы, в данном случае, ваше Spring Boot приложение.
- `spring-boot-app` - название сервиса.
- `image` - имя образа, которое вы хотите использовать.
- `build` - указывает на необходимость сборки образа.
- `context: .` - путь к контексту сборки (где находится Dockerfile).
- `dockerfile: Dockerfile` - указывает на использование Dockerfile для сборки образа.
- `ports` - пробрасывает порт 8080 из контейнера на хост.

Этот файл позволит вам собрать и запустить ваше Spring Boot приложение в контейнере Docker с помощью команды `docker-compose up`.

Использование Jenkins

1 Простота в использовании

Jenkins предоставляет простой и интуитивно понятный интерфейс для настройки и запуска автоматических сборок.

2 Гибкость настройки

С помощью плагинов и конфигурационных файлов можно настроить практически любой тип сборки и развертывания.

3 Масштабируемость

Jenkins позволяет масштабировать сборочные агенты горизонтально для обработки большого количества сборок.



Запуск Spring Boot приложения с помощью Jenkins из docker образа

Используя Jenkins и Docker, мы можем автоматизировать процесс сборки и развертывания Spring Boot приложений. Jenkins позволяет настроить pipeline, который собирает и тестирует приложение, а затем запускает его в контейнере Docker.

Пример

Для запуска Spring Boot приложения с помощью Jenkins из Docker образа с использованием Gradle, вам потребуется настроить Jenkins Pipeline скрипт в файле Jenkinsfile. Ниже приведен пример такого скрипта:

```
pipeline {  
    agent any  
  
    stages {  
        stage('Build') {  
            steps {  
                script {  
                    docker.image('your-spring-boot-image').inside {  
                        sh './gradlew build'  
                    }  
                }  
            }  
        }  
    }  
  
    stage('Run') {  
        steps {  
            script {  
                docker.image('your-spring-boot-image').inside('-p 8080:8080') {  
                    sh 'java -jar build/libs/your-spring-boot-app.jar'  
                }  
            }  
        }  
    }  
}
```

Пример

В этом примере:

- `agent any` указывает на то, что пайплайн может запускаться на любом агенте.
- `Build` этап собирает приложение с помощью Gradle.
- `Run` этап запускает Spring Boot приложение из Docker образа, пробрасывая порт 8080.
- `docker.image('your-spring-boot-image').inside` используется для выполнения команд внутри контейнера Docker с вашим образом.
- `sh './gradlew build'` выполняет сборку проекта с помощью Gradle.
- `sh 'java -jar build/libs/your-spring-boot-app.jar'` запускает Spring Boot приложение.

Этот Jenkins Pipeline скрипт позволит вам автоматизировать процесс сборки и запуска Spring Boot приложения из Docker образа с использованием Gradle.

Использование Ansible

1 Простота в использовании

Ansible позволяет легко автоматизировать различные задачи конфигурации и развертывания.

2 Гибкость и масштабируемость

Ansible поддерживает широкий спектр устройств и операционных систем, а также может масштабироваться для управления большим количеством хостов.

3 Открытый и расширяемый

Ansible основан на открытых стандартах, поддерживает язык разметки YAML и может быть расширен с помощью модулей и плагинов.



Пример использования Ansible для управления Jenkins

Для управления Jenkins с помощью Ansible, вам необходимо настроить соответствующие плагины и скрипты. Вот пример использования Ansible для управления Jenkins:

1) Установите плагин Ansible в Jenkins:

- Перейдите в раздел "Manage Jenkins" -> "Manage Plugins".
- Найдите и установите плагин Ansible.
- После установки перезагрузите Jenkins.

2) Создайте Jenkins Pipeline скрипт с использованием Ansible:

```
groovy
pipeline {
    agent any
    stages {
        stage('Deploy with Ansible') {
            steps {
                script {
                    ansiblePlaybook(
                        playbook: 'path/to/your/playbook.yml',
                        inventory: 'path/to/your/inventory',
                        credentialsId: 'your-ansible-credentials'
                    )
                }
            }
        }
    }
}
```

Пример использования Ansible для управления Jenkins

- 3) В вашем Ansible playbook (`playbook.yml`) опишите задачи для управления Jenkins.
- 4) Укажите путь к инвентарному файлу (`inventory`) и учетным данным Ansible (`credentialsId`) в вашем Jenkins Pipeline скрипте.

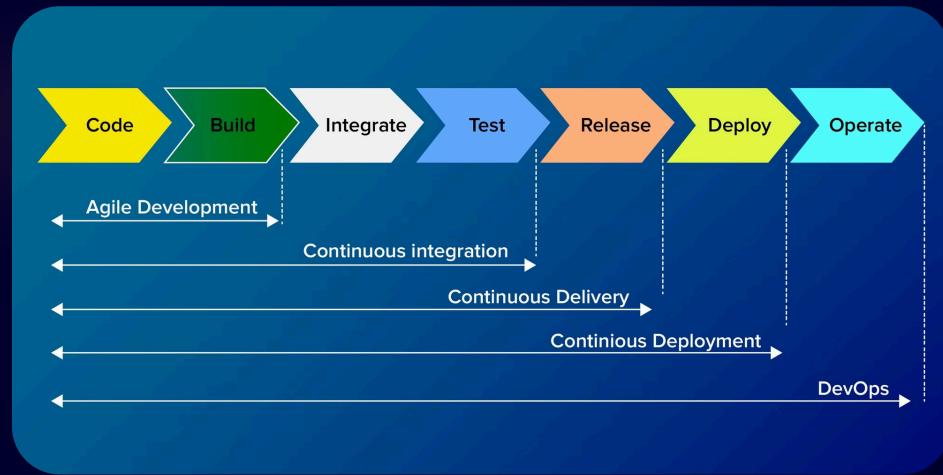
Этот пример демонстрирует базовый способ использования Ansible для управления Jenkins через Jenkins Pipeline. Для более сложных сценариев и настроек, обратитесь к документации Ansible и Jenkins для более подробной информации.

Какие задачи решает DevOps инженер

DevOps инженер ответственен за автоматизацию процессов разработки, тестирования и развертывания ПО.

Он также занимается созданием и поддержкой инфраструктуры для разработки и обеспечения непрерывной поставки ПО.

DevOps инженеры тесно сотрудничают с разработчиками и администраторами для оптимизации процессов разработки.



Различия между DevOps и традиционными методами разработки

Традиционные методы разработки

Частые выпуски софта

Низкая автоматизация

Очень медленная разработка и отгрузка ПО

Узкоспециализированные роли разработчиков и операционных инженеров

DevOps

Частые поставки

Высокая степень автоматизации

Быстрая разработка и отгрузка ПО

Команда DevOps - полная автоматизация и управление кодированием

Ключевые преимущества DevOps

- **Улучшенная скорость доставки:** DevOps позволяет ускорить процесс развертывания приложений и обновлений.
- **Повышенная надежность:** Автоматизация и стандартизация процессов позволяют снизить количество ошибок.
- **Лучшее взаимодействие команд:** DevOps способствует сотрудничеству между разработчиками и операционными специалистами.

Основные вызовы при внедрении DevOps

1

Отделение и автоматизация процессов

Адаптация к изменениям и внедрение новых инструментов.

2

Изменение корпоративной культуры

Преодоление сопротивления изменениям внутри организации.

3

Коммуникация и сотрудничество

Объединение различных функциональных групп для достижения общих целей.



Лучшие практики внедрения DevOps

Автоматизация

Используйте инструменты для автоматизации сборки, тестирования и развертывания приложений.

Мониторинг и Логирование

Внедряйте системы мониторинга и логирования для отслеживания производительности и выявления проблем.

Инфраструктура как Код

Определяйте и управляйте инфраструктурой через код для повышения надежности и гибкости.

Непрерывная Доставка

Разрабатывайте практики для доставки обновлений приложений в любое время без проблем.



Заключение

DevOps - это ключевое направление в IT-индустрии, обеспечивающее эффективное взаимодействие между разработчиками и операционной командой.

Используя автоматизацию, коллаборацию и мониторинг, DevOps позволяет компаниям улучшить качество своего программного обеспечения и ускорить его выход на рынок.

