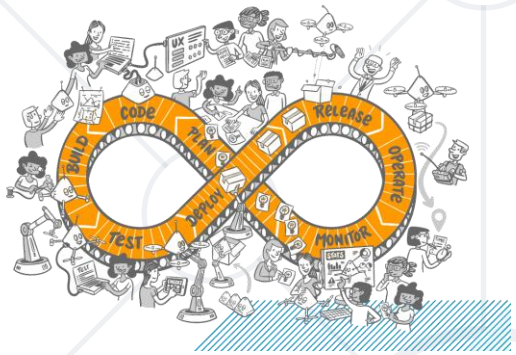
DevOps Overview

What Is DevOps, Practices, Tools, Trends



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Table of Content

What is DevOps?

Combining Software Development and IT Teams

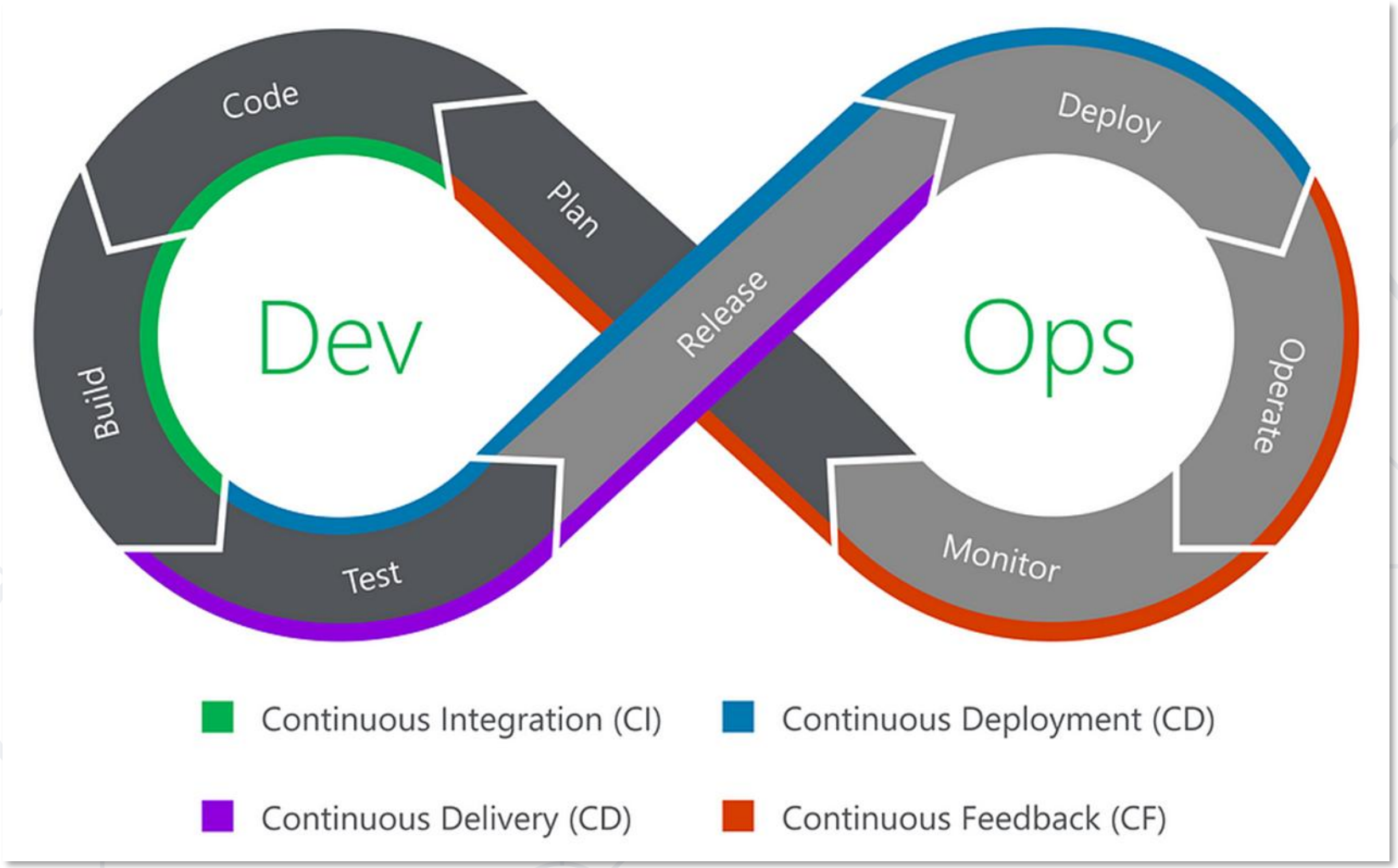
What is DevOps?

- **DevOps** is a set of practices, tools, and philosophy that combines **development** (**Dev**) and **operations** (**Ops**) into one, continuous process
- Unites **people**, **process**, and **technology** in application **planning**, **development**, **delivery**, and **operations**
  - Enables coordination and collaboration between isolated roles like development, IT operations, quality engineering, and security

DevOps Lifecycle

- **DevOps lifecycle** (or **pipeline**) is a series of automated development processes or workflows within an **iterative development lifecycle**
- Represents the processes, capabilities, and tools for **development** (left side) and **operations** (right side)
  - Merging both sides into one seamless process
- Follows a continuous approach

Continuous Everything



Source: https://medium.com/taptuit/the-eight-phases-of-a-devops-pipeline-fda53ec9bba

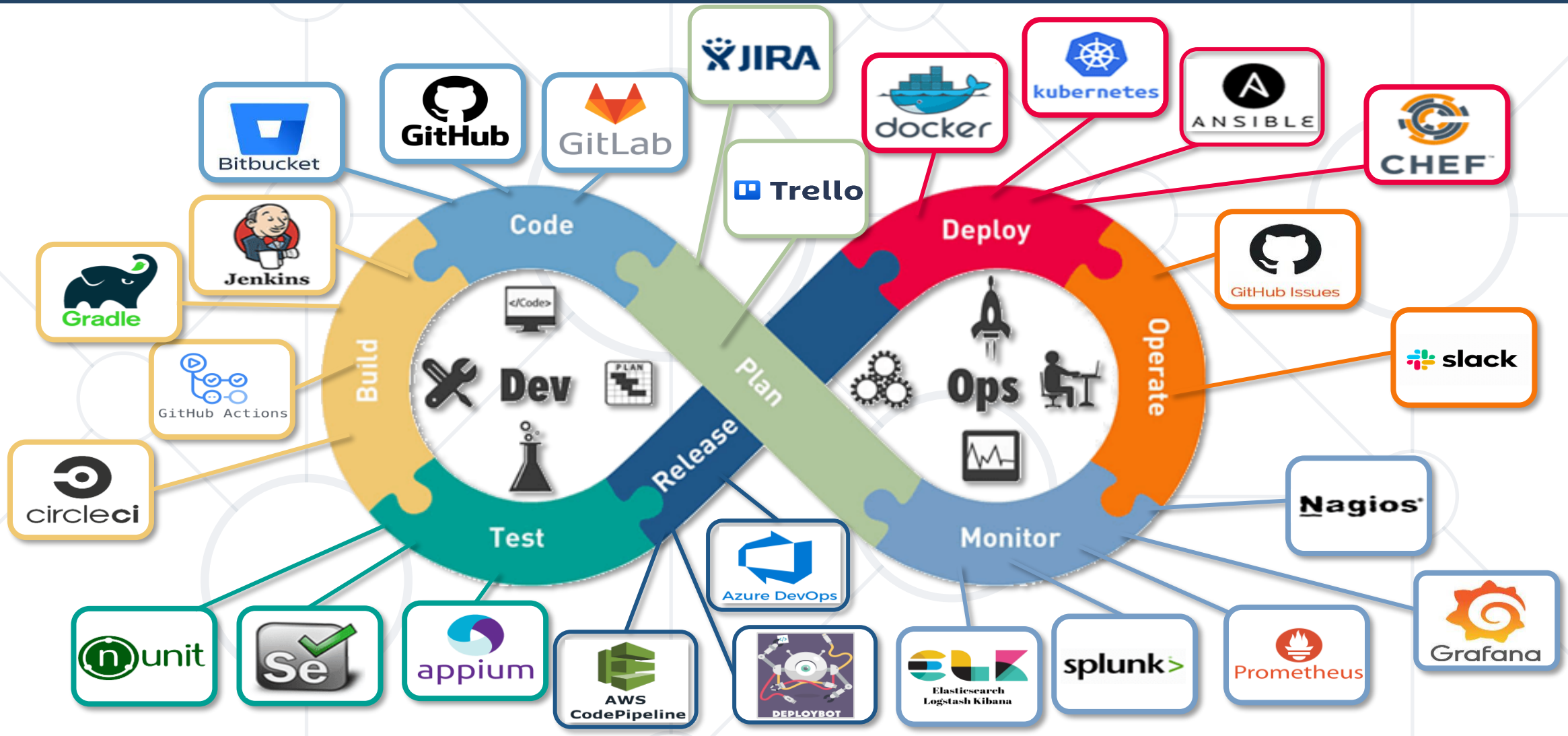
DevOps Lifecycle Stages

- **Plan**
  - Identify business requirements and collect end-user feedback
- **Code**
  - Code development
- **Build**
  - Finished code is committed to a shared repository
- **Test**
  - Build is deployed to a test environment and tests are performed
- **Release**
  - Operations team schedules the releases or deploys multiple releases to production
- **Deploy**
  - The production environment is built and the build is released
- **Operate**
  - Release is live now. Operations team takes care of server configuring and provisioning
- **Monitor**
  - DevOps pipeline is monitored to find problems / bottlenecks

DevOps Pipeline Phases

- Continuous **Development**
  - Plan and code
- Continuous **Integration**
  - Update code and add new features
- Continuous **Testing**
  - Run automated or manual tests
- Continuous **Deployment**
  - Code is automatically deployed on production servers
- Continuous **Feedback**
  - Evaluate user experience to improve future releases
- Continuous **Monitoring**
  - Monitor for system errors or performance issues
- Continuous **Operations**
  - Automate launching the app and its updates

DevOps Tools



DevOps Culture

- **DevOps culture** is a collaborative approach to software development and delivery that emphasizes **communication**, **automation**, and **improvement**
- **Collaboration** is crucial
  - All teams should communicate honestly and openly about DevOps processes, priorities, and concerns together
- As teams align, they take **ownership** and **become involved in other lifecycle phases**, not just the ones central to their roles
- DevOps teams remain agile by **releasing software in short cycles**
- Teams strive to **learn** and **continuously improve**

DevOps Engineers

- **DevOps engineers** are responsible for the **deployment**, and **maintenance** of software applications
  - **Collaborate** with development and operations teams
  - Balance a blend of soft skills with their tech knowledge
- They understand **development lifecycles**, **DevOps culture**, **practices** and **tools**

Role of DevOps Engineers

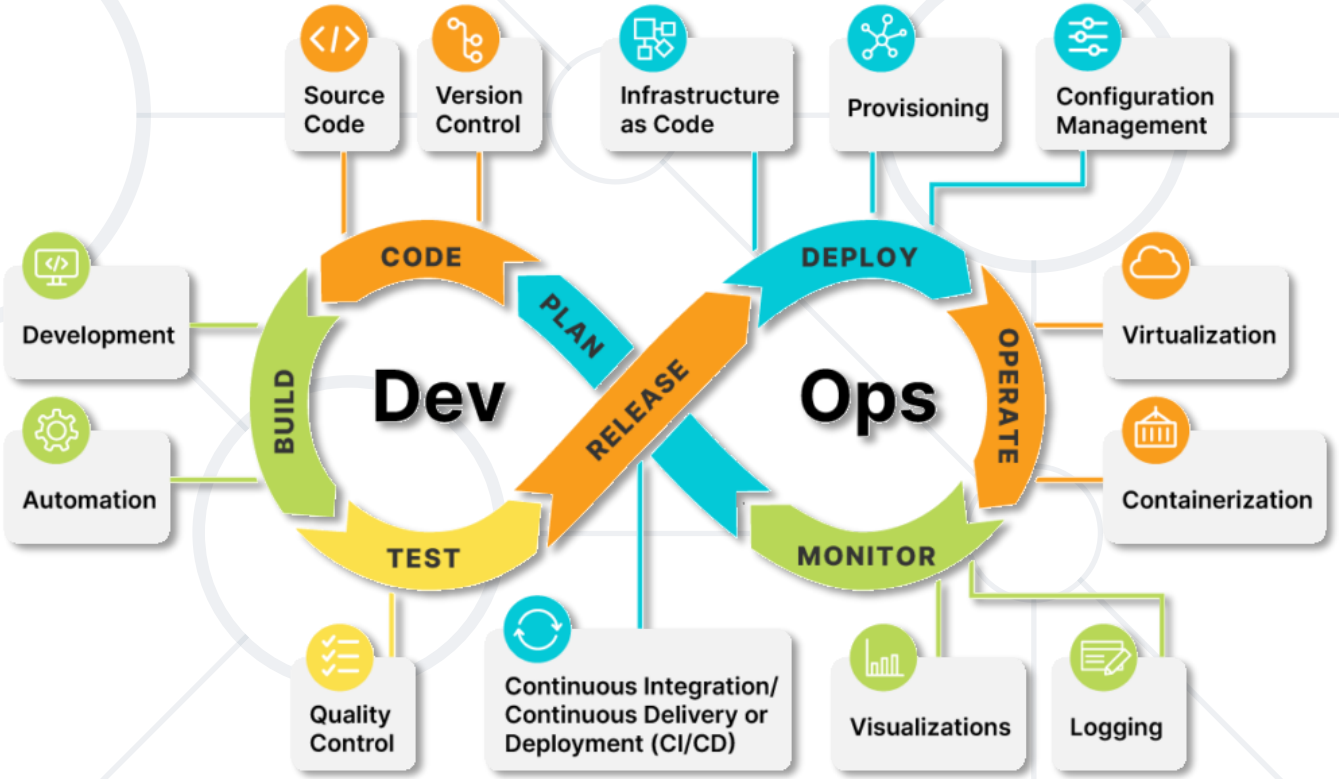
- Their job and responsibilities include
  - Automating processes
  - Managing and maintaining the infrastructure system
  - Monitoring performance
  - Ensuring the security of the software
  - Scale systems and ensure the availability of the services with developers

DevOps Practices

Helpful Throughout the Application Lifecycle

DevOps Practices

- Many **practices**, varying on the specific context and organization
- Some practices are
  - **CI/CD**
  - **Infrastructure as code (IaC)**
  - **Version control**
  - **Monitoring** and **logging**
  - **Automation**
  - **Agile software development**



CI/CD Pipeline

- **CI/CD Pipeline**
  - Cornerstone of DevOps describing the code journey **from a developer's machine to production**
- Consists of multiple **stages**
  - Development
  - Integration
  - Testing
  - Deployment
- **End goal**
  - Deliver features, updates and fixes to users quickly and reliably

CI/CD Pipeline

- **CI/CD** allows organizations to ship software quickly and efficiently
  - **Continuous integration**
    - Developers regularly **merge code changes** into a central repository, which are validated by **automated tests**
  - **Continuous delivery**
    - Code changes are automatically **prepared for a release** to production (and can be manually deployed)
  - **Continuous deployment**
    - Changes that pass all stages of production pipeline are **released automatically** (optional)
- Tools: **GitHub Actions**, Jenkins, CircleCI, etc.

Infrastructure as Code (IaC)

- **Infrastructure as Code (IaC)**
  - Managing and provisioning of infrastructure through code instead of through manual processes
- Used to **automatically manage infrastructure resources**
  - Servers
  - Operating systems
  - Software platforms
  - Storage
  - Networking
  - Etc.

IaC Tools

- **IaC tools** define **infrastructure resources** using **code / config files**
  - Can be version controlled, tested, and deployed automatically
- Tools: Ansible, Puppet, Chef, Saltstack, Terraform, etc.
- **Approaches** to Iac
  - **Declarative**
    - Defines the desired state of the system, i.e. resources you need and their properties
  - **Imperative**
    - Defines the specific commands for the desired configuration

Version Control

- **Version control** (**source control**)
  - The practice of **managing code in versions** to make code easy to review and recover
  - Includes tracking revisions and change history
    - Saves each individual changes in a special database
  - Necessary for CI/CD and IaC
    - Helps enhance efficiency
    - Allows preserving agility when a team grows larger
- Tools: **Git**, SVN, Mercurial, etc.

Version Control

- **Essential** for software development
  - Serves as a **safety net** to protect code
  - Allows several people to work on a project **simultaneously**
    - Improves collaboration and enhances development speed
  - Manages changes in
    - Code
    - Configurations
    - Infrastructure definitions
    - Documentation

Monitoring and Logging

- **Monitoring** means having **full, real-time visibility** into the health and performance of the entire application stack
  - **App metrics**, **event data**, **logs**, traces, etc. are collected and analyzed
  - Actionable and meaningful **alerts** are set for failures in the entire deployment pipeline
    - Thus, DevOps team can mitigate issues in real time
- Tools: ELK Stack, Splunk, Prometheus, Grafana, Alertmanager, Nagios, etc.

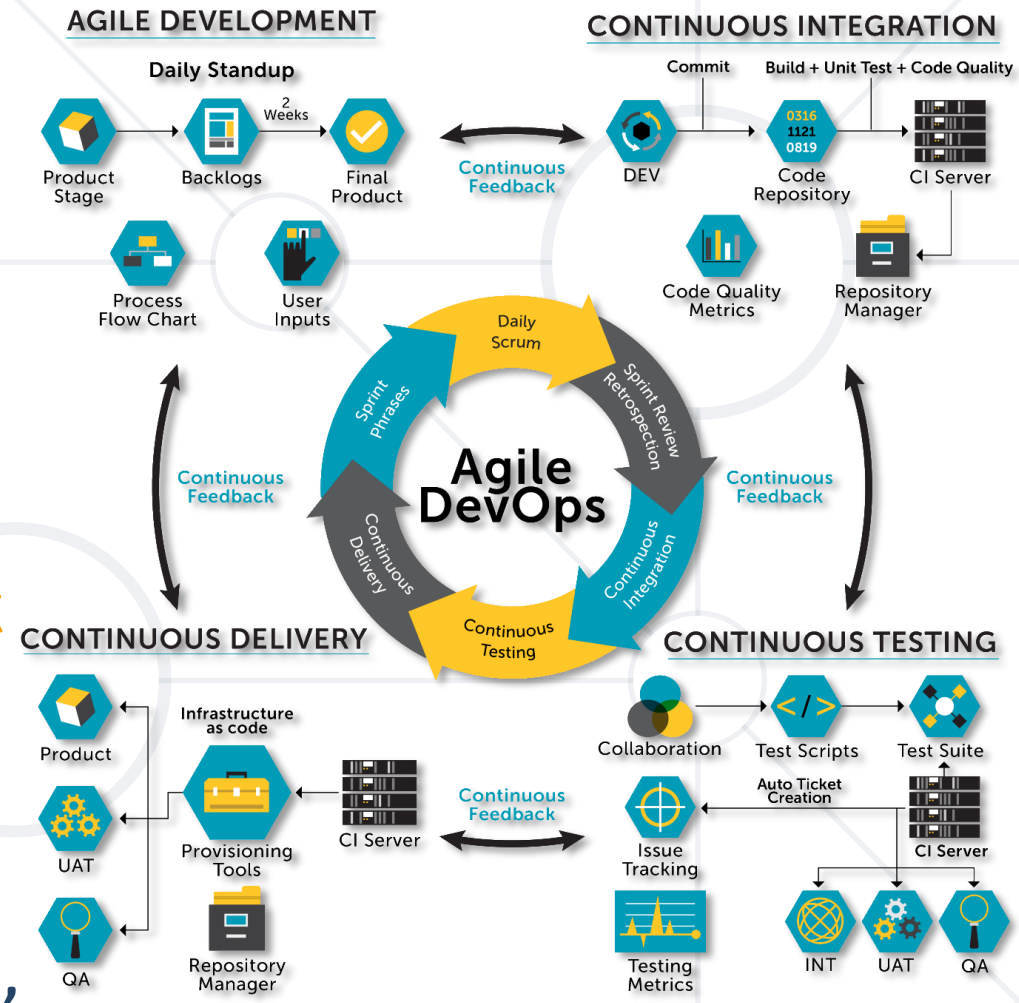
Automation

- DevOps teams aim to **automate** as much of the **software lifecycle** as possible to have more time for writing code and developing features
  - With **automation** the simple act of pushing code changes to a source code repository can trigger a build, test, and deployment process
  - Pros: **software delivery** is **faster**, **processes** are **consistent**, **predictable** and **scalable**, teams don't perform tedious manual tasks
- **Tools** are different for each step of the DevOps process

Agile Software Development

Software University

- **Agile** == modern software development approach
- It emphasizes on
  - **High adaptability to change** through **short release cycles**
  - **Customer** and **user feedback**
  - **Team collaboration**
- In **DevOps**, **Agile practices** include increased automation, improved collaboration, etc.

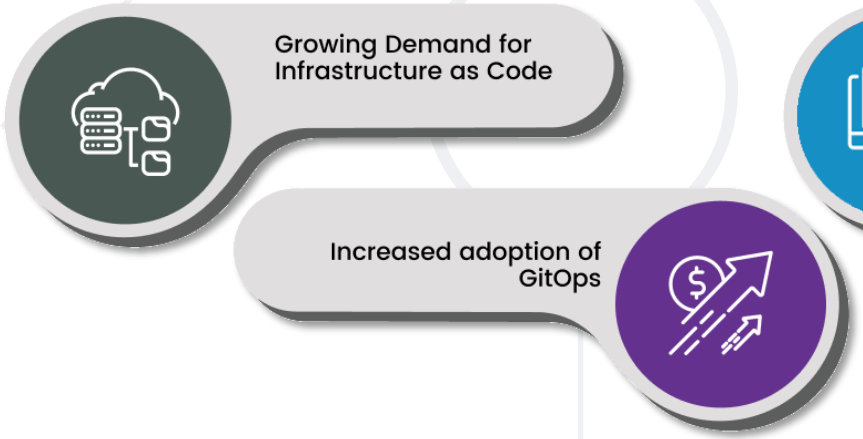


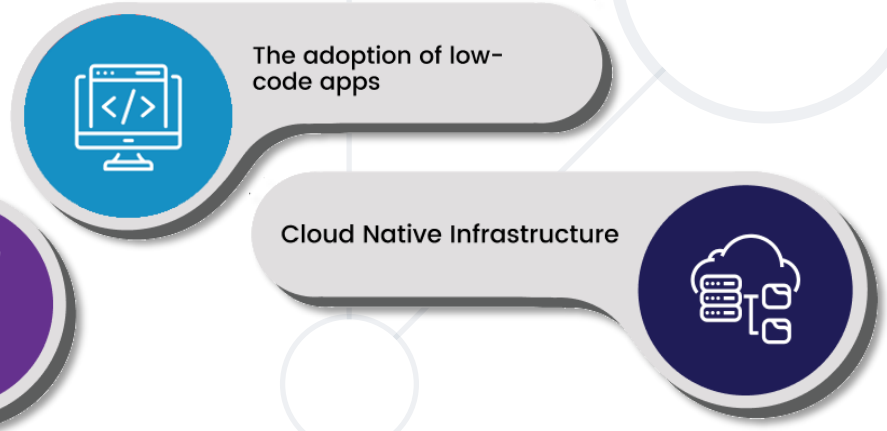
DevOps Trends

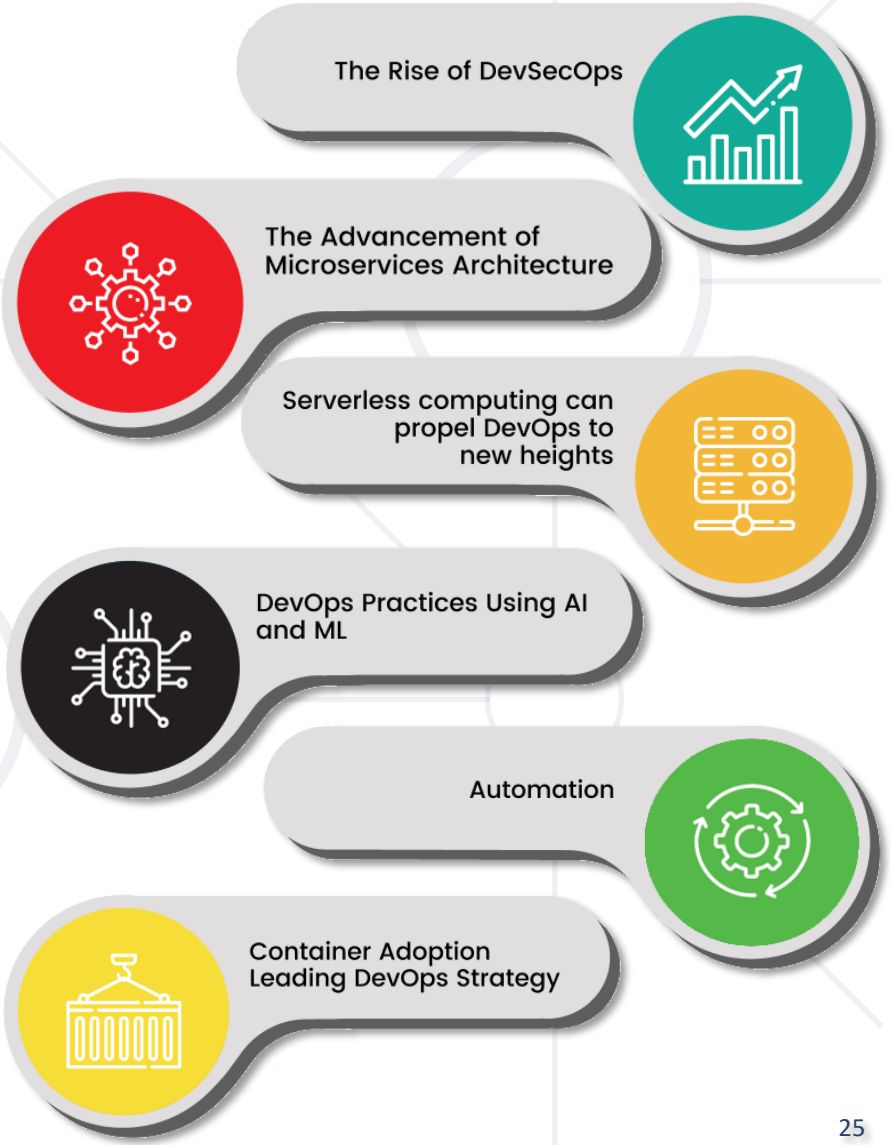
Additional DevOps Practices for Improved Lifecycle

DevOps Trends

- **DevOps movement trends** include
  - Increased focus on **security** and **compliance**
  - Adoption of **microservices architecture**
  - Evolution of **automation** and **AI**
  - And many more…
- They improve overall job productivity







DevSecOps

- **DevSecOps** = **development** + **security** + **operations**
- Includes **DevOps framework** with **security** as a shared responsibility
- Its mindset is to **integrate security practices** into applications and infrastructure from the start
- Identifying security vulnerabilities via analysis
- Tools
  - Static analysis
    - SonarCube, Fortify , Veracode, Chekmarx
  - Dynamic analysis
    - OWASP Zed Attack Proxy, Burp Suite, Acunetix, WebInspect

DevOps vs DevSecOps

Software University

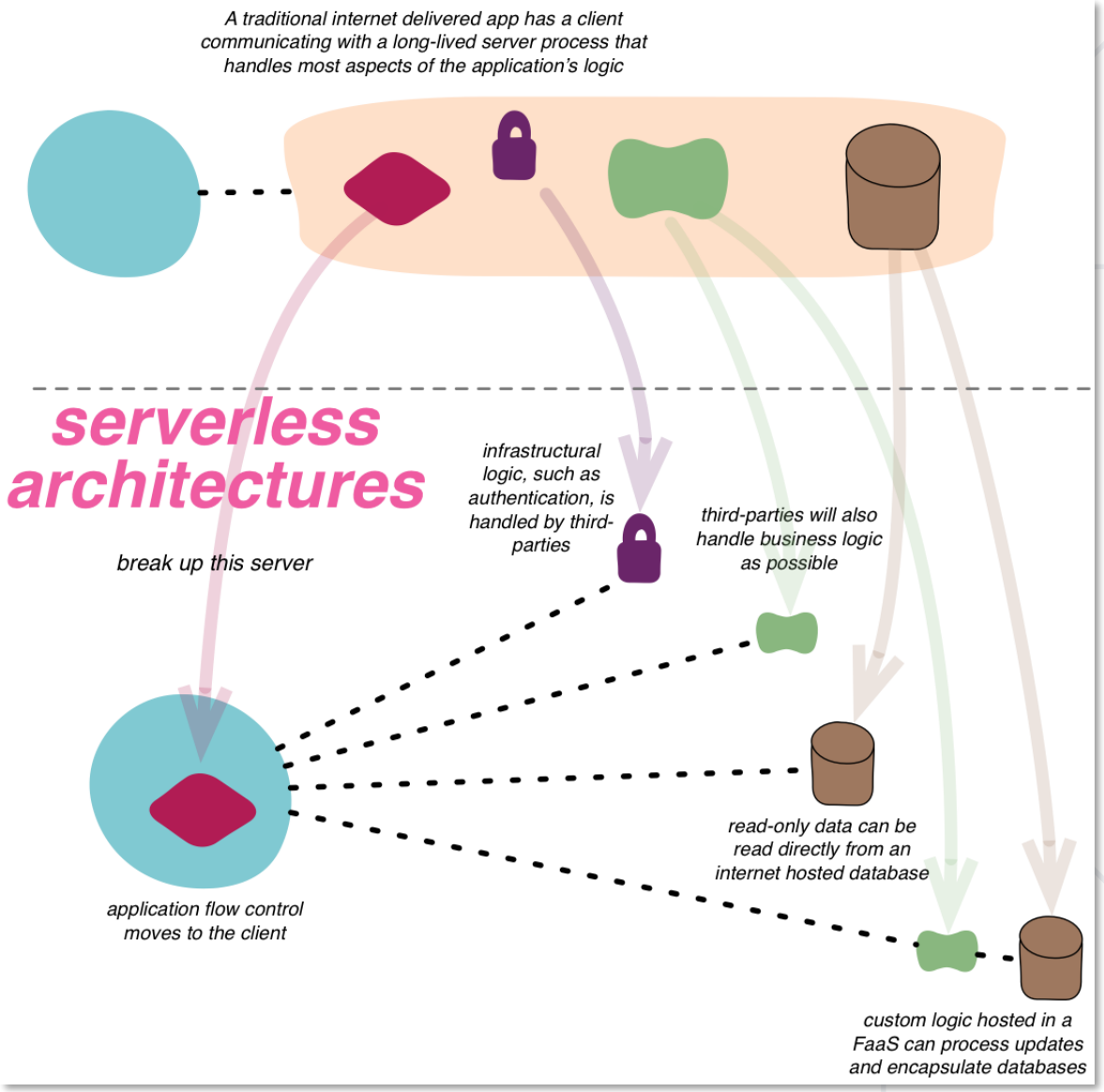
	DevOps	DevSecOps
Focus	Increasing quality and speed of software development and delivery	Secure software development processes by integrating security
Process	CI/CD	CI/CD + additional security-related processes
Activities	Continuous testing, development and monitoring QA tasks	Pre-commit, commit-time, build-time, test-time, deploy time checks of code

Static vs Dynamic Analysis in DevSecOps

- Static Analysis
  - Used for identifying **security vulnerabilities**
  - Analysis of the code **without executing** it
  - Catch potential security issues **early** in the development stage
- Dynamic Analysis
  - Used for identifying **security weaknesses**
  - Analysis of the code by **executing the app** in real or simulated environment
  - Detect security issues **at runtime**

Serverless Computing

- **Serverless computing** refers to **outsourcing back-end cloud** infrastructure and operations tasks to a **cloud provider**
  - Developers **focus on writing code**
  - Cloud provider manages the **infrastructure**, ensuring agility and scalability



Serverless Computing

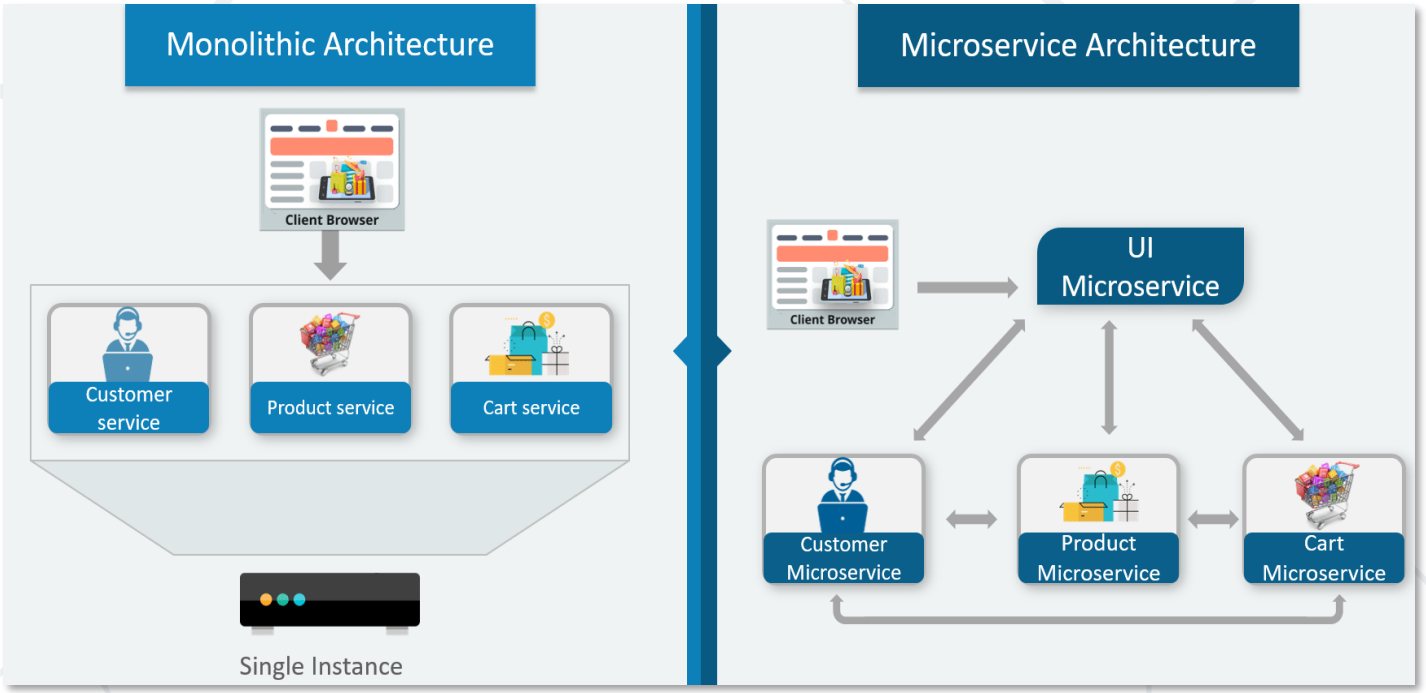
- **Serverless computing** == **Function-as-a-Service** (**Faas**)
- Based on **event-driven execution**
  - Allows functions to be triggered in response to specific events (changes in data or user requests, etc.)
- **Stateless nature**
  - Serverless functions are designed to be stateless
- Wide range of tools
  - Frameworks, SDKs, CLIs

Microservices Architecture

- **Microservices** == architectural approach to development that **breaks the application** into different **loosely coupled services**
  - Each service focuses on a specific business capability
    - Can be independently developed, deployed and scaled
- As everything is broken down into separate services, **development teams** can also be **divided** to **tackle each service**
  - Makes the development process more flexible

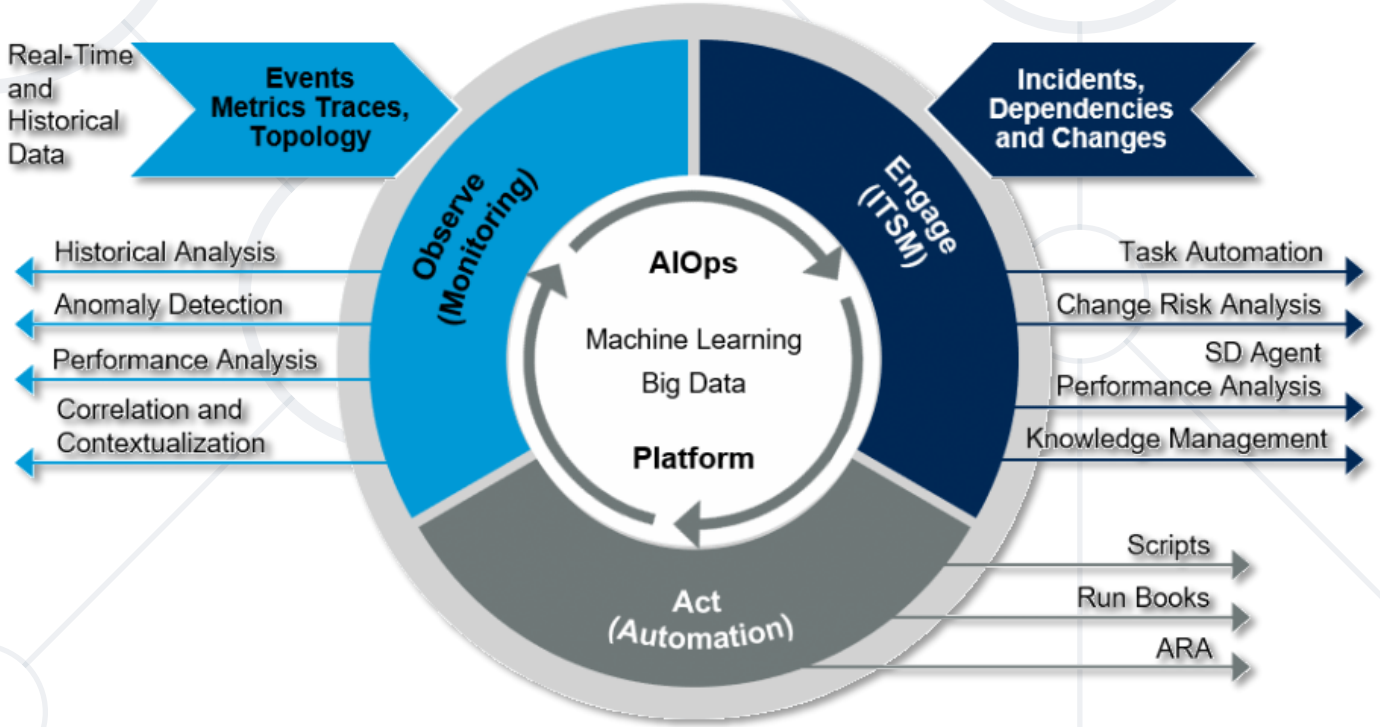
Microservices Architecture

- **Communication** between services is typically achieved through **lightweight protocols**, e.g., HTTP/REST
- Each microservice can have its own technology stack, programming language and database
  - These depend on the specific business requirements



AIOps and MLOps

- **AIOps** (Artificial Intelligence for IT Operations) refers to the use of artificial intelligence (**AI**) and machine learning (**ML**) technologies to automate and enhance various IT operations and processes
- **AIOps** helps with identifying the main cause of the problems that hamper operational productivity
- **MLOps** helps with optimizing operations and enhancing productivity



Summary

- **DevOps** == a set of **practices**, **tools** and a **cultural philosophy** that automate and integrate the processes between **software development** and **IT operations teams**
- 8 **DevOps lifecycle stages** and 7 **pipeline phases**
- **DevOps practices** include **CI/CD**, **Infrastructure as Code**, **Version Control**, **Monitoring** and **Logging**, **Automation**, **Agile Software Development**, etc.
- DevOps trends include **DevSecOps**, **Microservices**, **Serverless Computing** and **AIOps**

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg