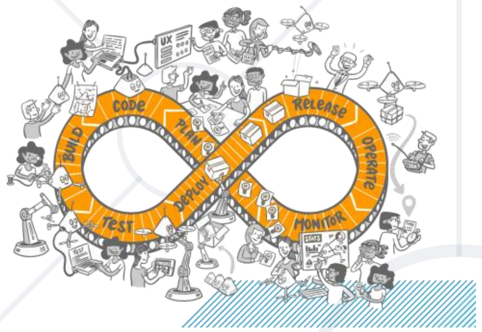
DevOps Overview

What Is It, Practices, Tools, Trends



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

sli.do

#QA-Auto-BackEnd

Table of Content

1. What is DevOps?
2. DevOps Practices
3. DevOps Trends

What is DevOps?

Combining Software Development and IT Teams

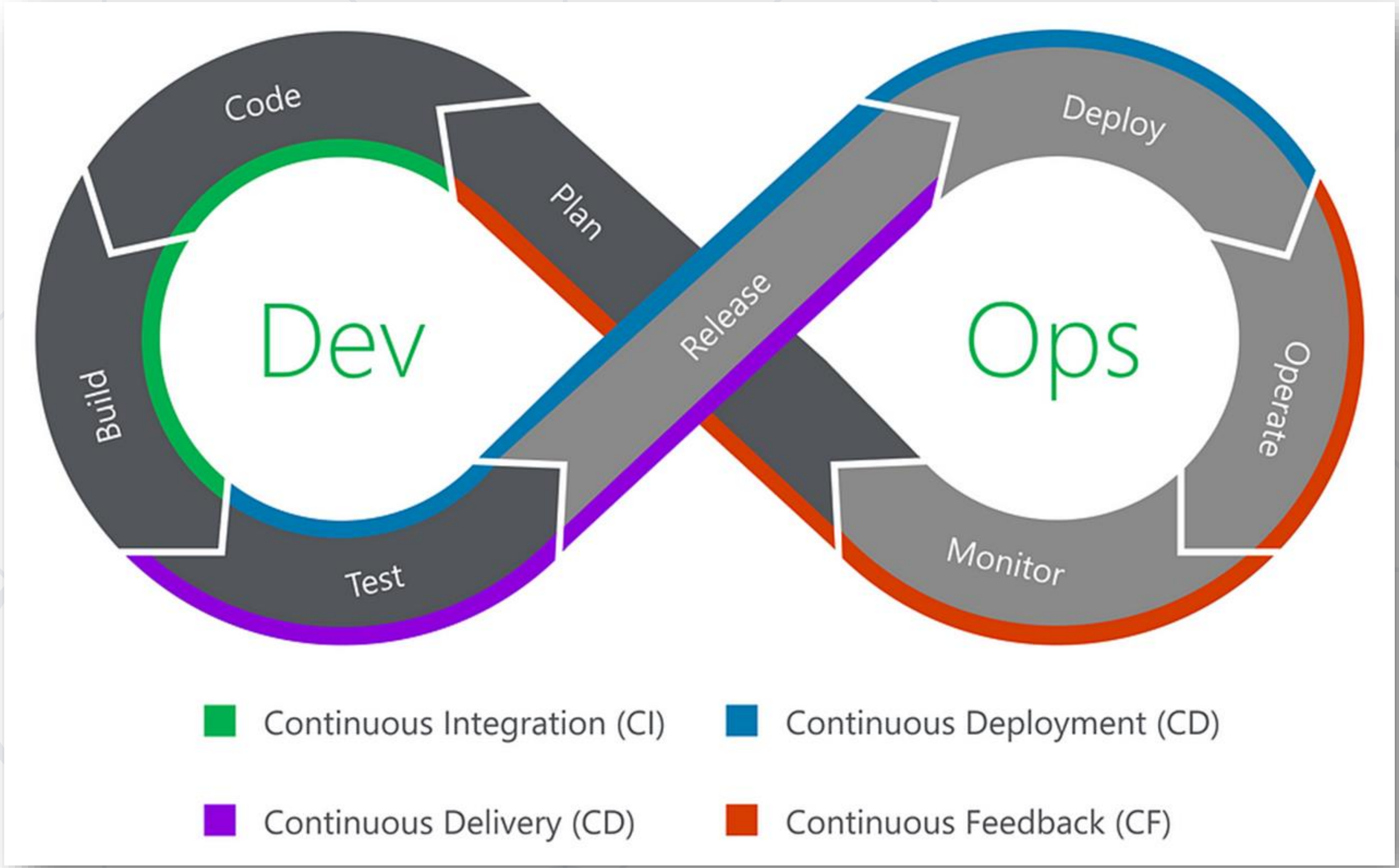
What is DevOps?

- **DevOps** == 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**) == 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



🟩 Continuous Integration (CI) 🟦 Continuous Deployment (CD)

🟪 Continuous Delivery (CD) 🟥 Continuous Feedback (CF)

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

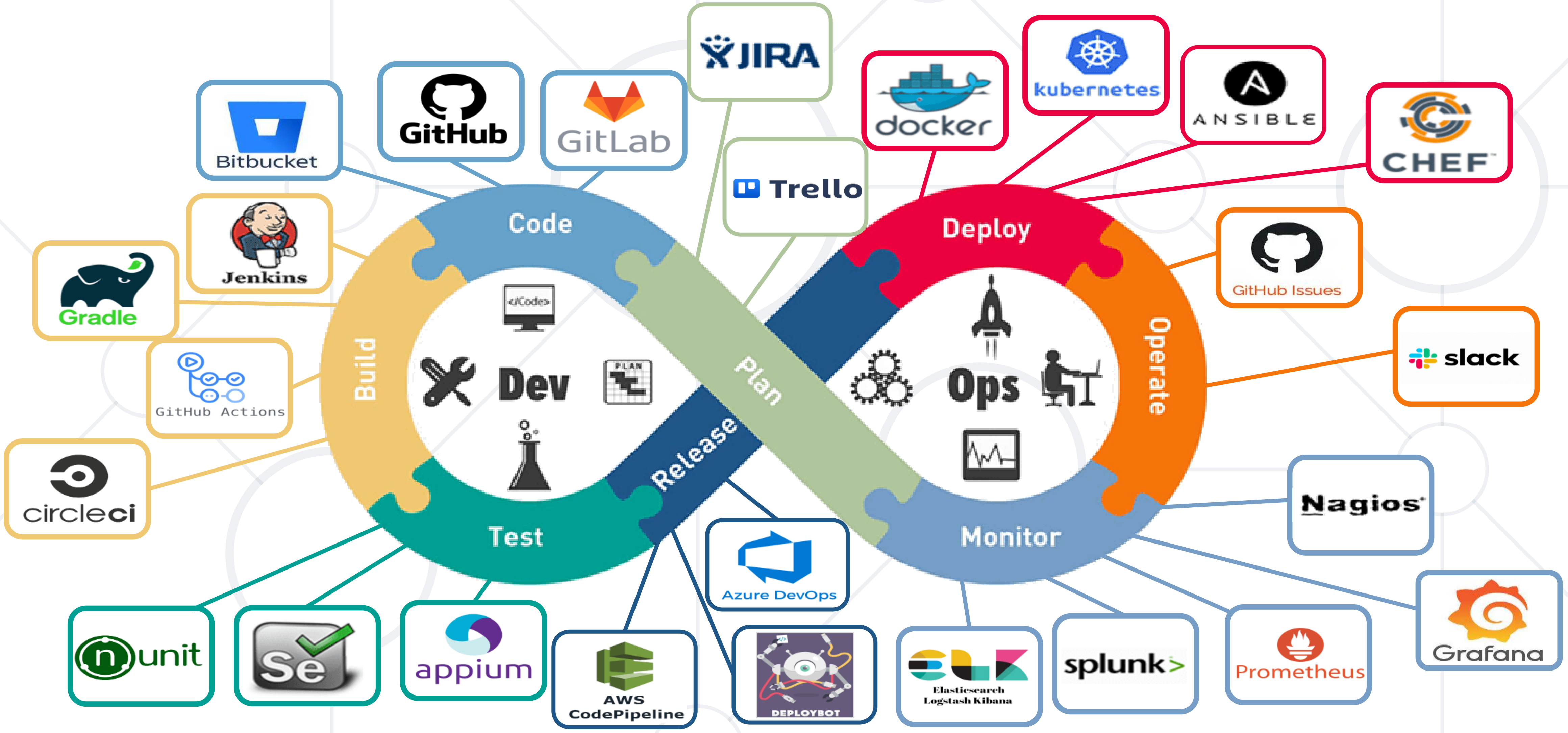
DevOps Lifecycle Stages

- **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** == 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

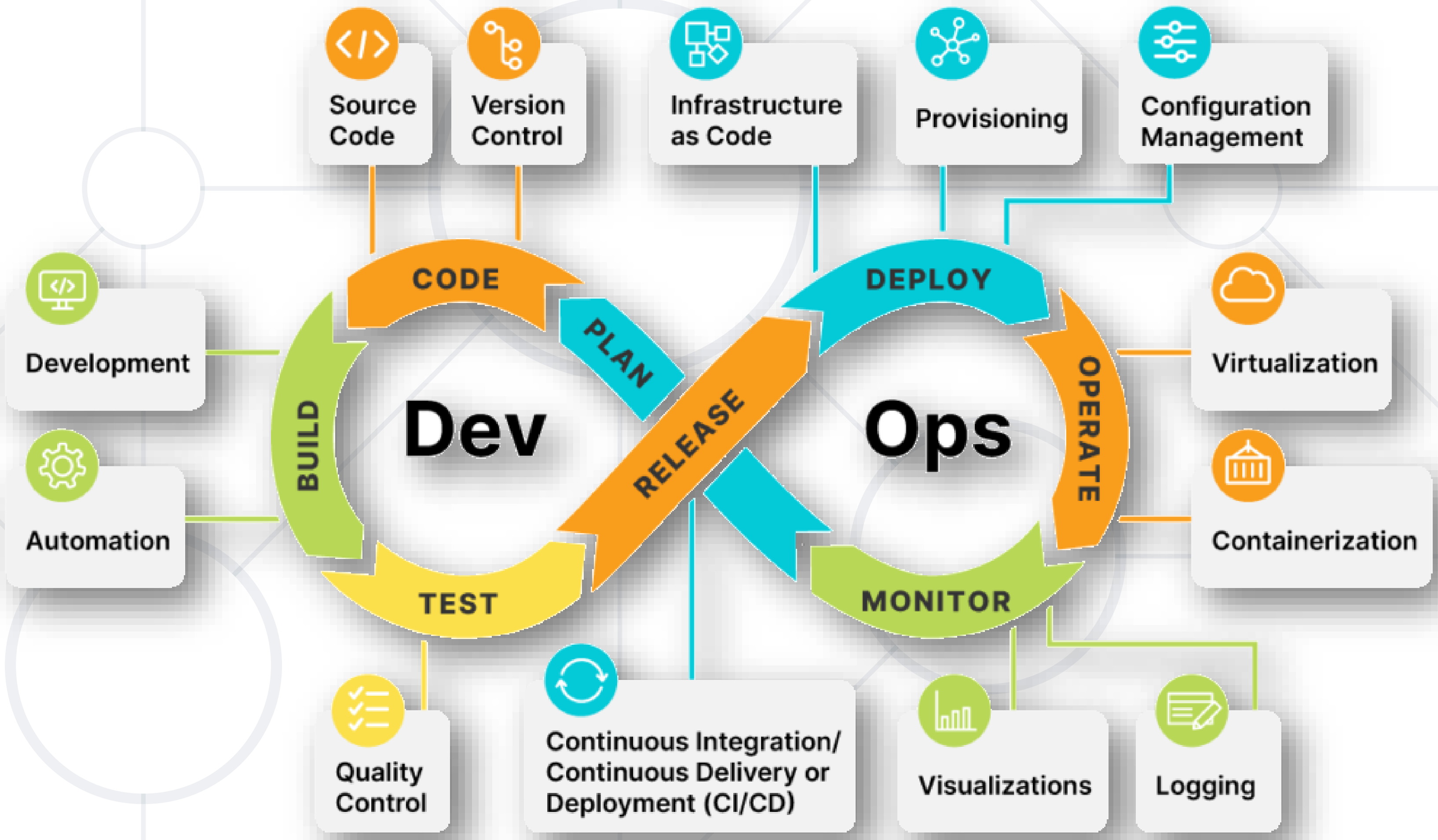
DevOps Practices

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**

DevOps Practices



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

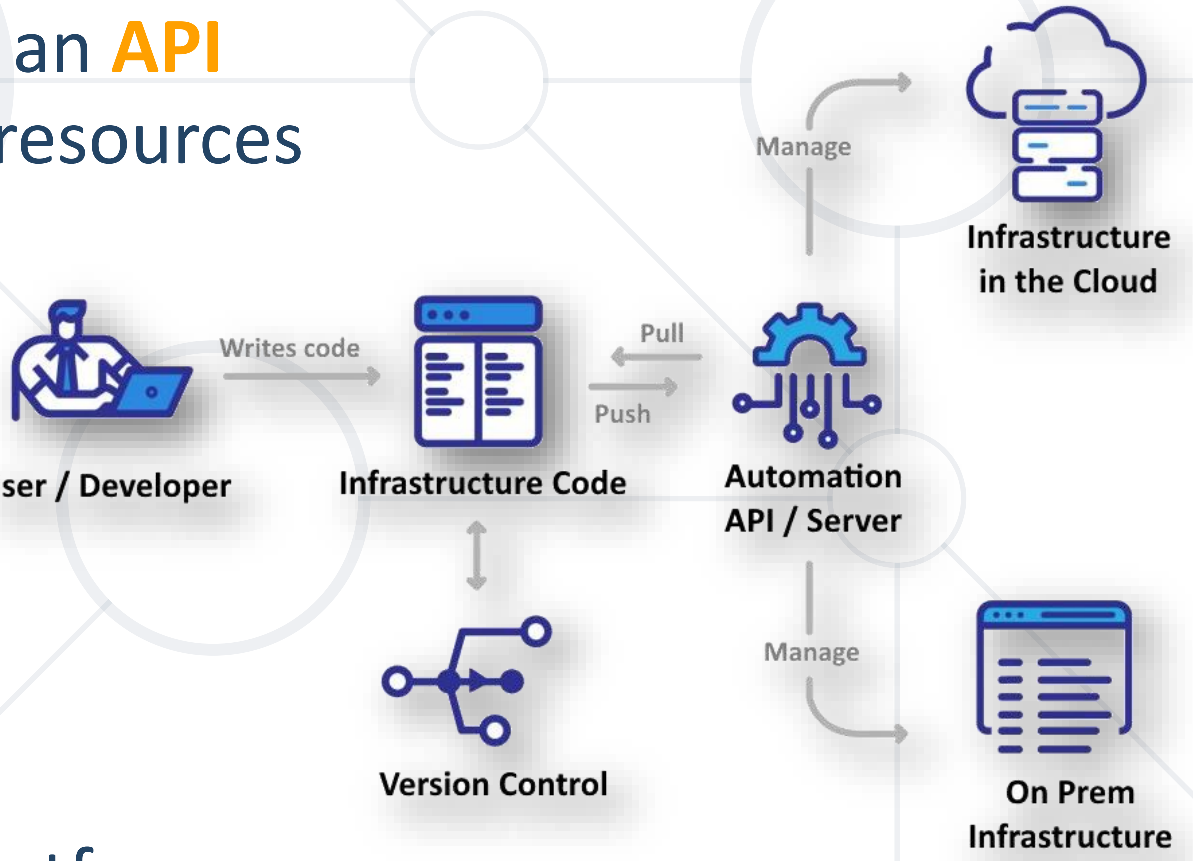
- **CI/CD** allows organizations to ship software quickly and efficiently
  - **Continuous integration**
    - Developers regularly **merge code changes** into a central repository, which are validated by **automated tests**
  - **Continuous delivery**
    - Code changes are automatically **prepared 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**) is the managing and provisioning of **infrastructure through code** instead of through manual processes
  - As VMs, networks, OS servers, storage, etc.
- **IaC** involves
  - Writing **code** to define the desired state of an **infrastructure environment**
  - Using **tools** to **automatically** deploy and configure the environment based on the code

What Do You Need for IaC?

- **Remote accessible hosting** or **IaaS cloud hosting platform**
  - IaC tools connect and modify remote host
  - IaaS cloud hosting platforms have an **API** for modification of infrastructure resources
- **Provisioning tool**
  - Automates the infrastructure deploy and management
- **Configuration management tool**
  - Manages infrastructure state
- **Version control system**
  - Stores text files used by the CM platform



Approaches to IaC

- **Imperative approach**
  - Tell the system **how** to do something every step of the way
  - Defines the **specific commands** to be executed in a **specific order** for the **desired configuration**
- **Declarative approach**
  - Tell the system **what** you want and let it figure out how to do it
  - Defines the **desired state** of the **system** – resources, their properties and an IaC tool for configuration

IaC Configuration Files

- **IaC** is a form of configuration management that **codifies infrastructure resources** into text files
- **Configuration files** are created with your infrastructure specifications
  - Should be **version controlled** and **tested** (unit, integration, ... tests)
  - Ensure that you provision the **same environment every time**
  - Allow you to divide your infrastructure into **modular components** and combine them through automation
  - Should contain always **up-to-date infrastructure documentation**

Infrastructure Provisioning Tools

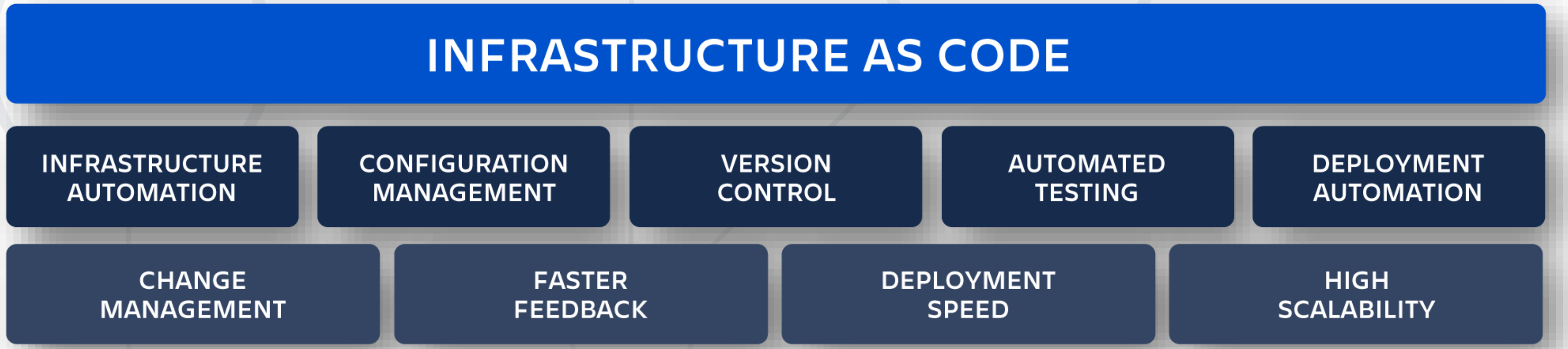
- **Infrastructure provisioning**
  - Create infrastructure resources like virtual servers, storage, networking, cloud managed services, etc.
- Primary goal
  - **Keep the infrastructure in its desired state** and reproduce or update it
- Tools
  - Terraform, AWS Cloudformation, Azure Resource Manager (ARM) Templates, Pulumi
- They can also **trigger CM tools**

Configuration Management Tools

- **Configuration management**
  - Configuring infrastructure resources
    - e.g., configuring a server with required applications or configuring a firewall device
- Primary goal
  - **Configure the server**
- Tools
  - Ansible, Chef, Puppet, SaltStack, etc.
- In **cloud environments**, tools use an **API-based dynamic inventory** to get the server details

IaC Benefits for DevOps

- IaC is an important part of implementing DevOps practices
  - Version control, test and deploy of infrastructure code changes
  - Improved collaboration – Ops team can participate in writing IaC templates together with Dev team, as IaC uses simple, text-based files
  - Automation of creation and management of infrastructure resources
  - Consistency and reliability across environments is achieved as IaC generates the same environment every time



Software Configuration Management (SCM)

- **Version Control System ≈ Source Control System**
  - Tool for **managing** the changes during the **development**
  - A **repository** keeps the **source code** and **other project assets**
    - Keeps a **full history** of **all changes** during the time
      - **Change log** shows who, when and why changed what
  - Solves **conflicts** on **concurrent changes**
    - Allows **reverting** of old versions
- Popular **source control systems**
  - **Git** – distributed source control (hierarchical)
  - Subversion (SVN) – central repository (centralized)

What is Git?

- **Git**
  - Distributed **source-control** system
  - The most popular source control in the world
  - Free open-source software
  - Works with local and remote repositories
  - Runs on Linux, Mac OS and Windows
- **GitHub**
  - **Social network** for **developers**
  - **Free project hosting** site with **Git repository**

Vocabulary

- **Repo** (repository)
  - Holds the project in a remote server
- **Branch**
  - Parallel development path (separate version of the project)
- **Merge branches**
  - Merge two versions of the same projects
- **Clone**
  - Download a local copy of the remote project
- **Commit**
  - Saves a set of changes locally
- **Pull**
  - Take and merge the changes from the Remote
- **Push**
  - Send local changes to the Remote

What is GitHub?

- **GitHub**
  - Platform and cloud-based service, based on **Git**
  - World's **most** used source code host
  - Used for **software development** and **version control**
    - Free for **open-source projects** and small private projects
    - Paid plans for private repositories with advanced features
- **GitHub Desktop**
  - Enables interacting with **GitHub** using a **GUI instead** of the **command line** or a **web browser**

Basic Git Commands

- **Clone** an existing Git repository

```
git clone [remote url]
```

- **Fetch** and **merge** the latest changes from the remote repository

```
git pull
```

- **Prepare** (add / select) files for a commit

```
git add [filename] ("git add ." adds everything)
```

- **Commit** to the local repository

```
git commit -m "[your message here]"
```

Basic Git Commands

- Check the **status** of your local repository (see the local changes)

```
git status
```

- Create a **new** local repository (in the current directory)

```
git init
```

- Create a **remote** (assign a short name for remote Git URL)

```
git remote add [remote name] [remote url]
```

- **Push** to a remote (send changes to the remote repository)

```
git push [remote name] [local name]
```

Git Conflict

- **Conflicts** generally arise when two or more people **change** the **same file simultaneously**
  - Or if a developer **deletes** a **file** while another developer is **modifying** it
- In these cases, **Git cannot automatically determine** what is **correct**
- **Conflicts** only **affect** the **developer conducting** the **merge**
- The rest of the team is **unaware** of the **conflict**

What is Branching?

- **Branches** allow you to work on **different parts** of a project without **impacting** the **main / master branch**
  - Serve as an abstraction for the **edit / stage / commit process**
- Represent a way to **request** a **brand new working directory**, staging area, and **project history**
  - Any new **commits** are **recorded** in the **history** for the **current branch**
    - Without impacting the main branch until it's decided to integrate the changes
- You can **switch between branches** and work on different projects without them interfering with each other

Pull Requests in GitHub

- **Pull requests**
  - A **mechanism** for developers to **notify** their team members that they have **completed** a **feature**
- The **pull request** is more than just a **notification**—it's a **code review process**, including **discussions** on the **proposed feature**
- If there are any **problems** with the **changes**, **teammates** can **post feedback** in the **pull request**
- This **activity** is **tracked** directly **inside** of the **pull request**

Monitoring and Logging

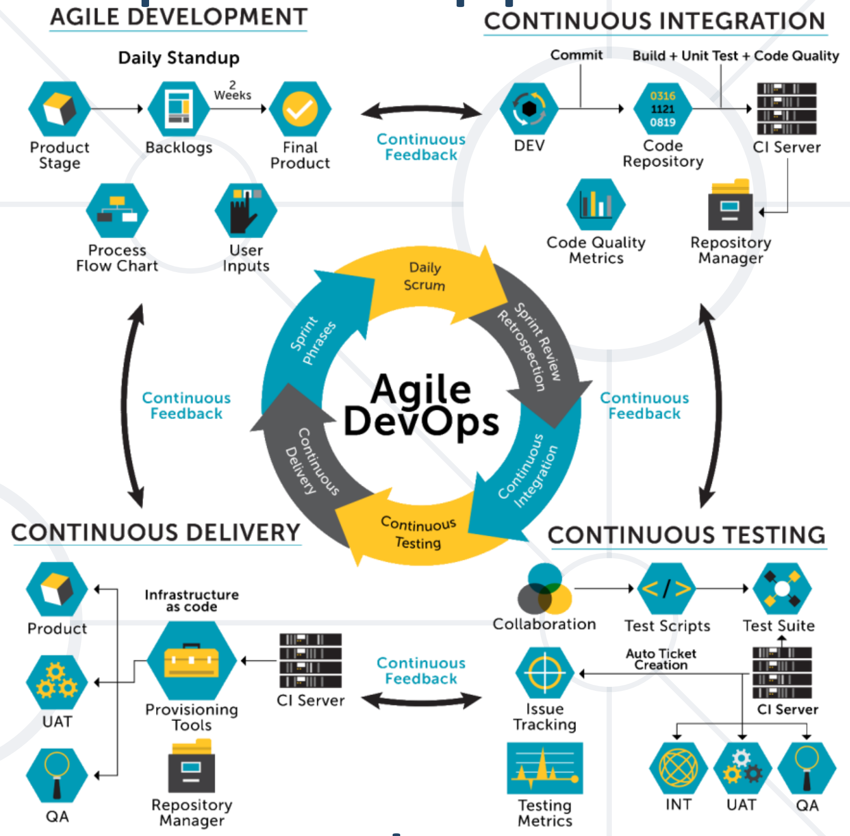
- **Monitoring** means having **full, real-time visibility** into the health and performance of the entire application stack
  - **App metrics**, **event data**, **logs**, traces, etc. are collected and analyzed
  - Actionable and meaningful **alerts** are set for failures in the entire
    deployment 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

- **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.

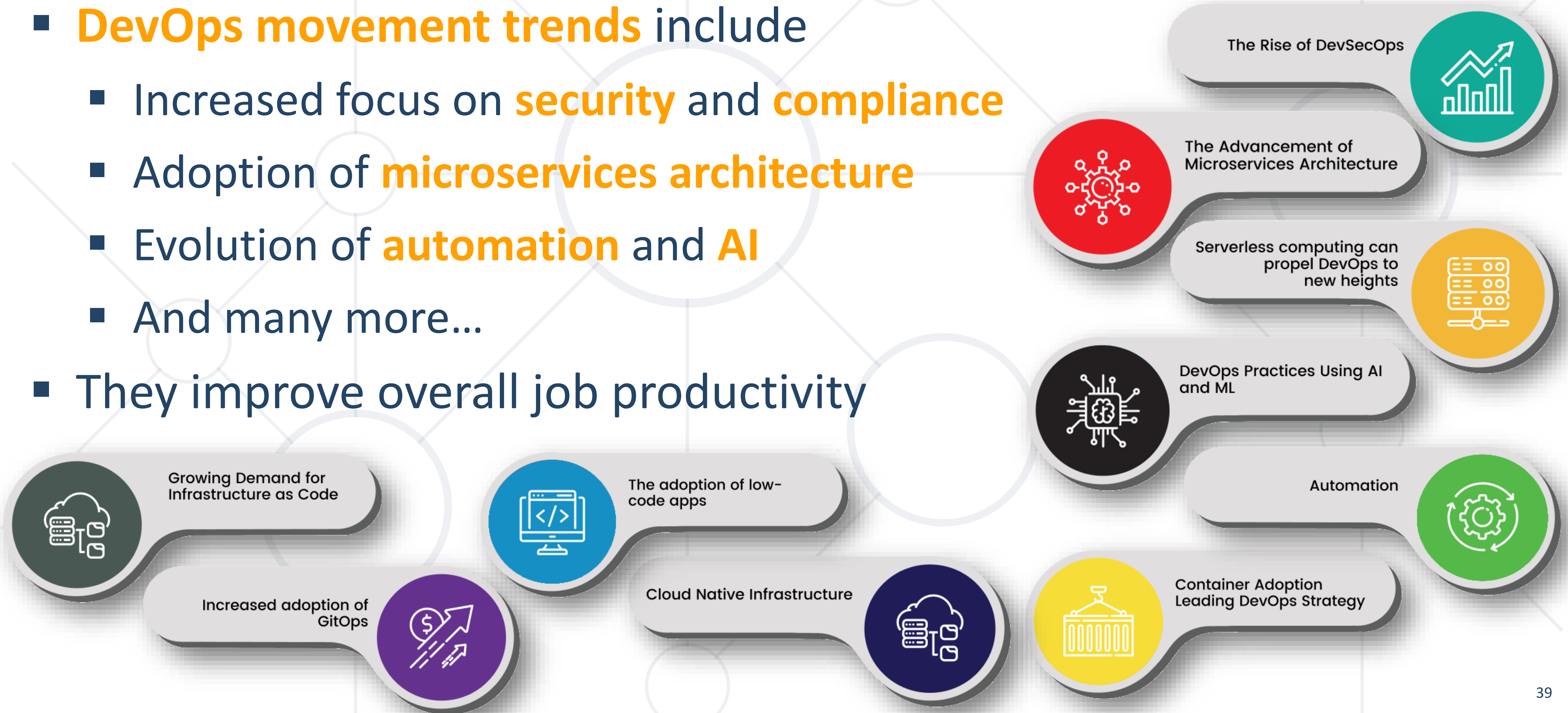


DevOps Trends

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



The Rise of DevSecOps
The Advancement of Microservices Architecture
Serverless computing can propel DevOps to new heights
DevOps Practices Using AI and ML
Automation
Container Adoption Leading DevOps Strategy
Growing Demand for Infrastructure as Code
The adoption of low-code apps
Increased adoption of GitOps
Cloud Native Infrastructure

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

	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

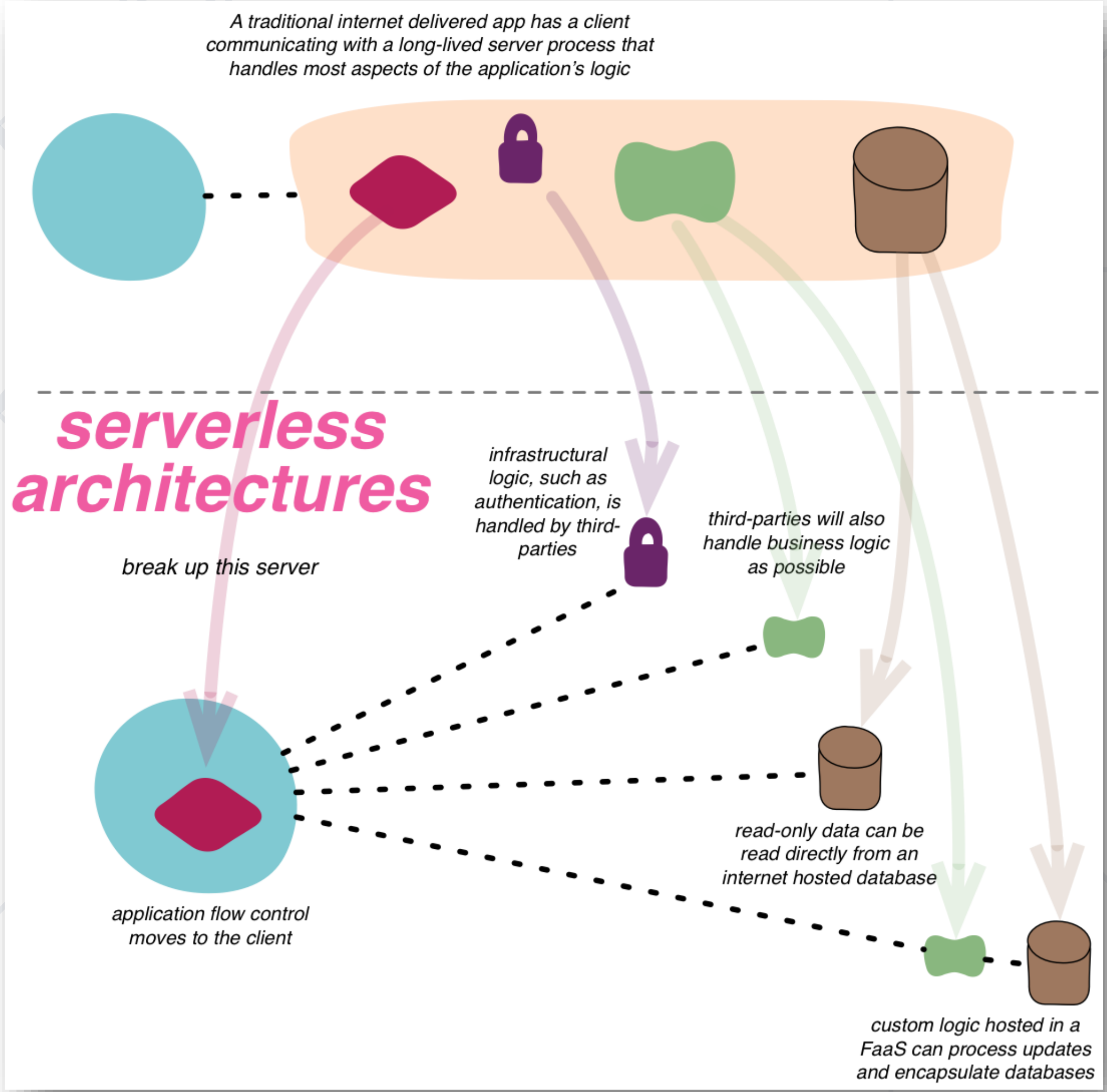
Static vs Dynamic Analysis in DevSecOps

- 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


A traditional internet delivered app has a client communicating with a long-lived server process that handles most aspects of the application's logic
serverless architectures
break up this server
infrastructural logic, such as authentication, is handled by third-parties
third-parties will also handle business logic as possible
application flow control moves to the client
read-only data can be read directly from an internet hosted database
custom logic hosted in a FaaS can process updates and encapsulate databases

Serverless Computing

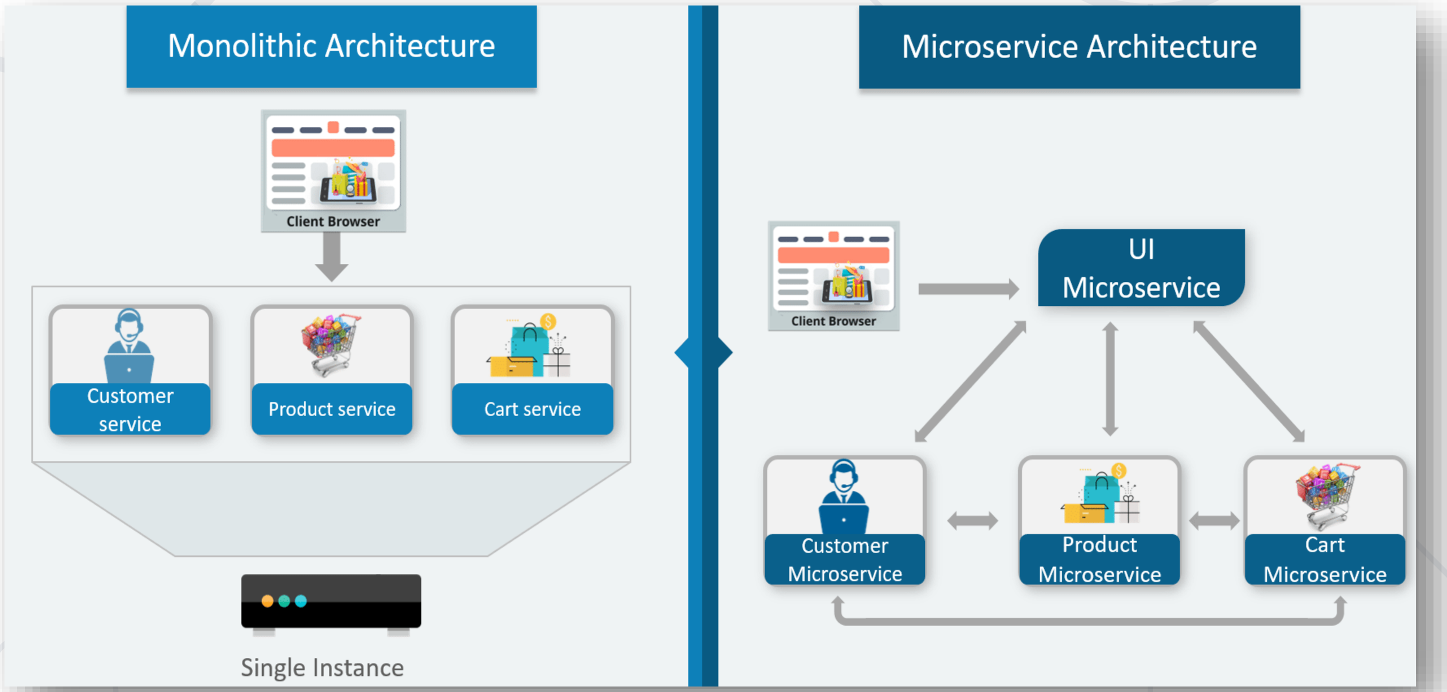
- **Serverless computing** == **Function-as-a-Service** (**Faas**)
- Based on **event-driven execution**
  - Allows functions to be triggered in response to specific events (changes in data or user requests, etc.)
- **Stateless nature**
  - Serverless functions are designed to be stateless
- Wide range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

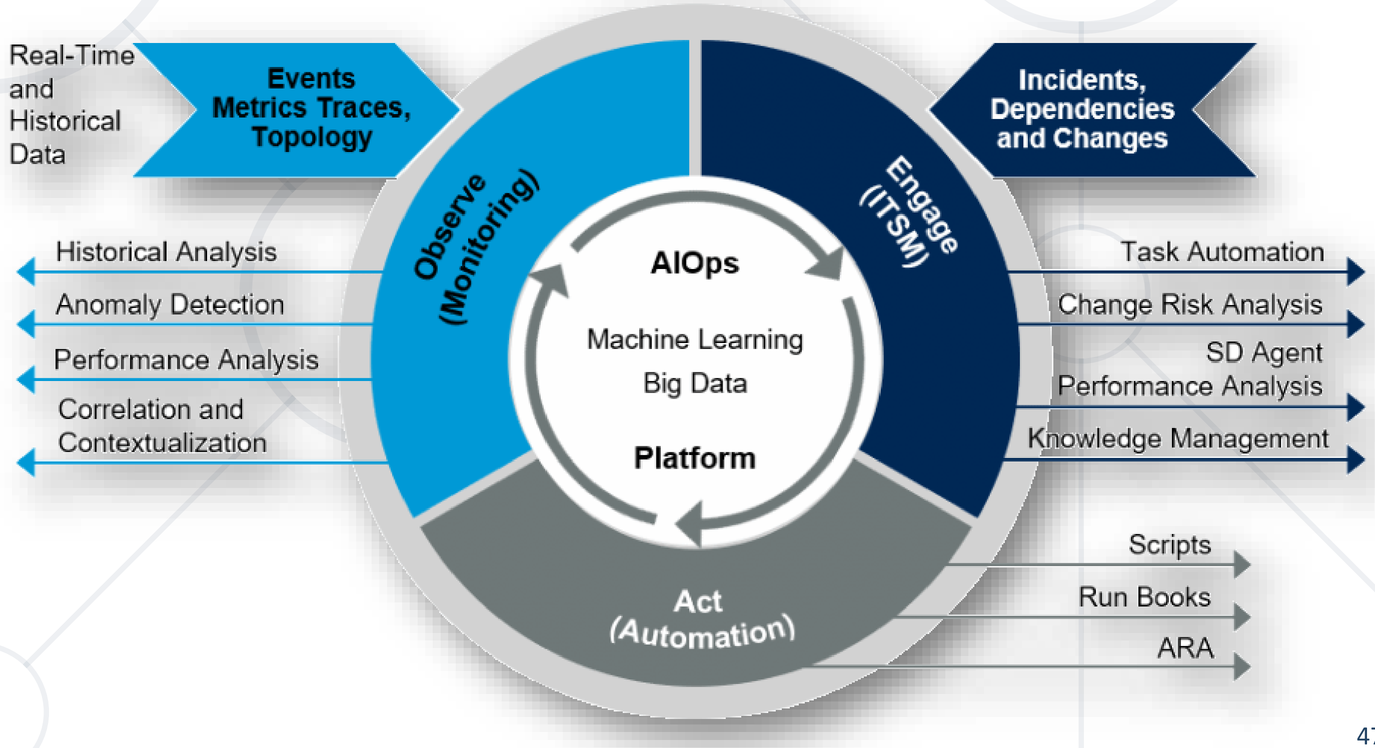
Microservices Architecture

- **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

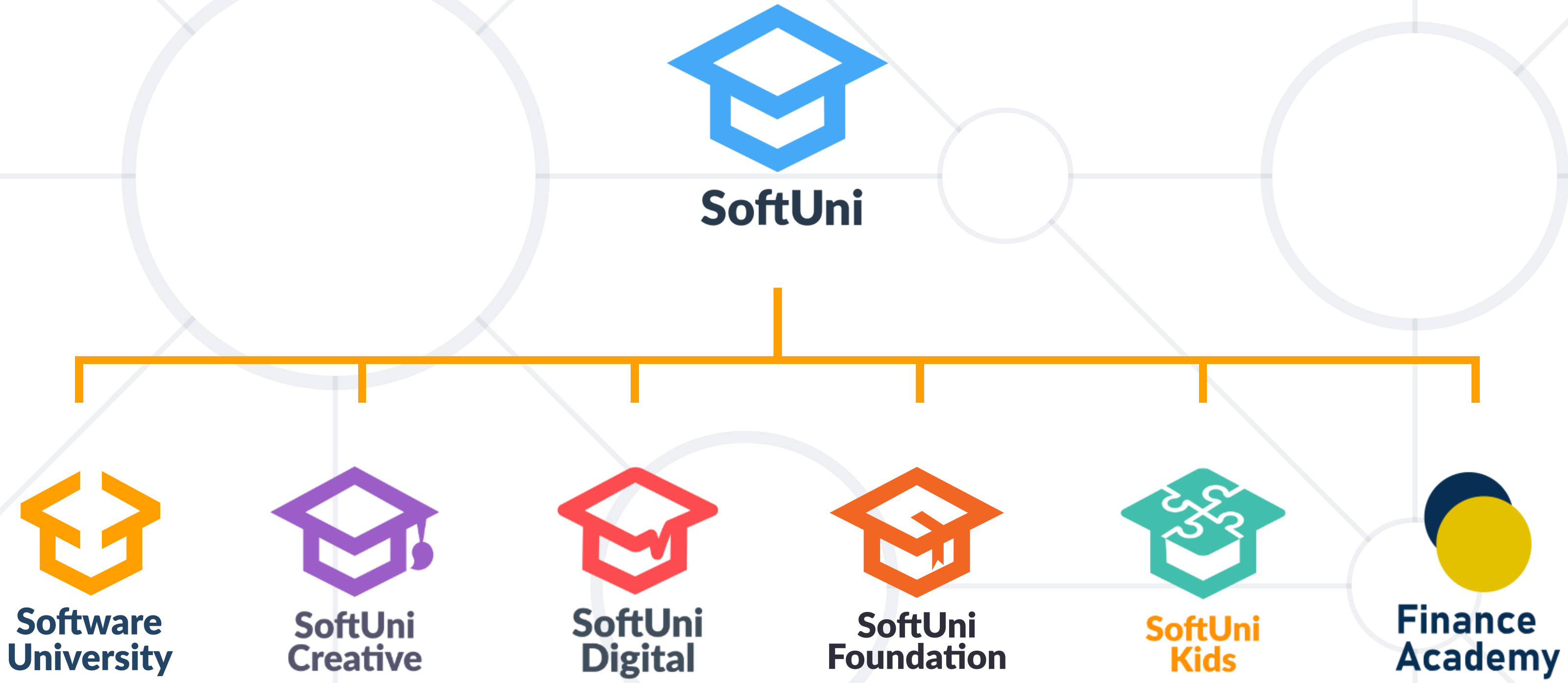
- **AIOps** (Artificial Intelligence for IT Operations) refers to the use of artificial intelligence (**AI**) and machine learning (**ML**) technologies to automate and enhance various IT operations and processes
- **AIOps** helps with identifying the main cause of the problems that hamper operational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

Questions?


SoftUni
Software University
SoftUni Creative
SoftUni Digital
SoftUni Foundation
SoftUni Kids
Finance Academy

SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SOFTWARE GROUP

DXC TECHNOLOGY

createX

Postbank

Решения за твоето утре

SUPER HOSTING .BG

DRAFT KINGS

VIVACOM

INDEAVR

Serving the high achievers

AMBITIONED

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License