


# DEVOPS FUNDAMENTALS

TechRacine Academy

https://techracine.com

https://academy.techracine.com

## Table of Contents


1. Introduction to DevOps ..... 3
2. Why DevOps Matters ..... 3
3. DevOps Culture and Collaboration ..... 3
4. Software Development Life Cycle and DevOps ..... 3
5. Continuous Integration (CI) ..... 4
6. Continuous Delivery and Continuous Development ..... 4
7. Version Control with Git ..... 4
8. Build and Package management ..... 4
9. Automated Testing ..... 5
10. Infrastructure as Code (IaC) ..... 5
11. Containerization and Docker ..... 5
12. Kubernetes and Container Orchestration ..... 5
13. Cloud and DevOps ..... 6
14. Monitoring and Logging ..... 6
15. DevSecOps and Security ..... 6
16. DevOPs Tools and Toolchains ..... 6
17. CI/CD Pipeline ..... 7
18. DevOps and Agile ..... 7
19. DevOps Best Practices ..... 7
20. Common DevOps Mistakes ..... 7
21. Business and Operational Importance ..... 8
22. Real World Usage ..... 8
23. Summary ..... 8
24. Exercises ..... 9

## 1. Introduction to DevOps

DevOps is a set of practices that brings software development and IT operations closer together.

It focuses on collaboration, automation, continuous delivery, and reliable software releases.

DevOps helps teams deliver software faster while maintaining quality and stability.

It connects development, testing, deployment, monitoring, and operations as a continuous process.

## 2. Why DevOps Matters

DevOps reduces delays between development and deployment.

Automation helps reduce repetitive manual work and human error.

Continuous feedback allows teams to identify and fix problems earlier.

DevOps can improve release speed, reliability, scalability, and collaboration.

## 3. DevOps Culture and Collaboration

DevOps is not only a collection of tools; it also involves a collaborative culture.

Developers, testers, operations teams, security teams, and other stakeholders work toward shared goals.

Teams communicate regularly and take shared responsibility for software delivery and reliability.

Automation and continuous feedback support this culture.

## 4. Software Development Life Cycle and DevOps

DevOps connects different stages of the Software Development Life Cycle (SDLC).

Typical activities include planning, coding, building, testing, releasing, deploying, operating, and monitoring.

Instead of treating these stages as isolated activities, DevOps encourages continuous flow and feedback.

## 5. Continuous Integration (CI)

Continuous Integration means frequently integrating code changes into a shared repository.

Automated builds and tests can run whenever changes are committed.

CI helps detect integration problems early and keeps the codebase in a working state.

## 6. Continuous Delivery and Continuous Deployment

Continuous Delivery keeps software ready for release through an automated and repeatable process.

Continuous Deployment goes one step further by automatically releasing validated changes to production.

Both approaches reduce manual release effort and make deployments more consistent.

## 7. Version Control with Git

Version control records changes to source code and helps developers work together.

Git supports commits, branches, merges, and collaboration through repositories.

A typical workflow is to create a branch, make changes, commit them, review or merge them, and push the changes to a remote repository.

## 8. Build and Package management

Build tools convert source code into software that can be tested or deployed.

Package managers help teams install and manage project dependencies.

Automated builds make the process repeatable and help maintain consistent environments.

## 9. Automated Testing

Automated testing checks software without requiring every test to be performed manually.

Unit, integration, system, and end-to-end tests can be included in a DevOps pipeline.

Automated tests provide fast feedback and help prevent defective changes from reaching later stages.

## 10. Infrastructure as Code (IaC)

Infrastructure as Code manages infrastructure using configuration files or code.

Instead of manually creating servers and environments, teams can define infrastructure in a repeatable way.

IaC improves consistency, version control, automation, and environment reproducibility.

## 11. Containerization and Docker

Containers package an application together with its required dependencies.

Docker is a widely used platform for building and running containers.

Containers provide consistent application environments and make applications easier to move between development, testing, and deployment environments.

## 12. Kubernetes and Container Orchestration

Container orchestration manages groups of containers across computing resources.

Kubernetes is a popular platform for deploying, scaling, and managing containerized applications.

It supports concepts such as pods, services, deployments, scaling, and self-healing.

## 13. Cloud and DevOps

Cloud platforms provide on-demand computing, storage, networking, and other services.

DevOps practices work well with cloud environments because infrastructure and deployments can be automated.

Cloud-based DevOps can support scalable applications, automated environments, and faster provisioning.

## 14. Monitoring and Logging

Monitoring collects information about application and infrastructure health.

Logging records events and messages that help teams understand application behavior.

Metrics, logs, alerts, and dashboards help teams detect issues and investigate failures.

## 15. DevSecOps and Security

DevSecOps integrates security practices into the DevOps process.

Security checks can be included in source control, code analysis, dependency scanning, testing, and deployment.

Early security testing helps identify vulnerabilities before software reaches production.

## 16. DevOPs Tools and Toolchains

DevOps commonly uses multiple tools for different stages of delivery.

Examples include Git for version control, Jenkins or GitHub Actions for CI/CD, Docker for containers, Kubernetes for orchestration, and Terraform for infrastructure as code.

The exact toolchain depends on the project's technology, cloud platform, team skills, and requirements.

## 17. CI/CD Pipeline

A CI/CD pipeline automates the movement of code from development toward deployment.

A simple pipeline can contain these stages:

Code → Build → Test → Security Checks → Package → Deploy → Monitor

Automation makes the process repeatable and provides feedback at each stage.

## 18. DevOps and Agile

Agile focuses on iterative development, customer feedback, and frequent delivery.

DevOps extends this continuous approach into testing, deployment, operations, and monitoring.

Together, Agile and DevOps can support shorter feedback cycles and more frequent software releases.

## 19. DevOps Best Practices

Use version control for application and infrastructure code.

Automate builds, tests, and deployments.

Keep environments consistent and reproducible.

Monitor applications and infrastructure continuously.

Use small, frequent, and reviewable changes.

Include security throughout the delivery pipeline.

Document important processes and maintain clear team ownership.

## 20. Common DevOps Mistakes

Avoid treating DevOps as only a tool installation exercise.

Do not depend heavily on manual deployments when automation is practical.

Avoid skipping automated tests and security checks.

Do not ignore monitoring after deployment.

Avoid unclear responsibilities and poor communication between teams.

Do not create unnecessarily complex pipelines for simple projects.

## 21. Business and Operational Importance

DevOps can reduce the time required to deliver software changes.

Automation can reduce repetitive work and improve consistency.

Continuous monitoring and feedback can help teams respond to incidents more quickly.

For organizations, DevOps supports collaboration, release management, reliability, and operational efficiency.

## 22. Real World Usage

DevOps is used in web applications, mobile applications, SaaS products, cloud platforms, banking systems, e-commerce, enterprise software, and many other technology environments.

Development teams use CI/CD to automate software delivery, containers to package applications, cloud infrastructure to scale systems, and monitoring tools to maintain reliability.

## 23. Summary

DevOps combines development and operations practices to improve software delivery and reliability.

Important concepts include collaboration, version control, CI/CD, automated testing, infrastructure as code, containers, cloud computing, monitoring, and security.

A successful DevOps approach combines suitable tools with automation, communication, and continuous improvement.

## 24. Exercises

1. Create a Git repository and practice commits and branches.

2. Build a simple CI pipeline for a small application.

3. Add automated tests to a sample project.

4. Create a basic Dockerfile and build a container image.

5. Run a containerized application locally.

6. Write a simple infrastructure configuration using an IaC tool.

7. Design a basic CI/CD pipeline diagram.

8. Add a security check to a software pipeline.

9. Create a monitoring dashboard for a sample application.

10. Explain how DevOps can improve a real-world web application delivery process.