# CI/CD: Build, Test, Deploy Training

BİLGİNÇ IT Academy
info@bilginc.com
bilginc.com

Duration: **3 Day**

**CI/CD: Build, Test, Deploy** is a hands-on DevOps training program designed to help professionals master modern software delivery automation. In today's fast-paced development environments, the ability to automate builds, testing, and deployments is essential for delivering reliable software at scale. This course provides practical knowledge of **Continuous Integration (CI)**, **Continuous Delivery (CD)**, and **Continuous Deployment (CDP)** methodologies.

Participants will learn how to build robust automation pipelines using industry-standard tools such as **GitHub Actions**, **Docker**, and **Ansible**. Through guided labs, practical demonstrations, and real-world scenarios, learners will gain hands-on experience designing scalable, secure, and efficient CI/CD workflows.

By the end of the course, attendees will be capable of implementing end-to-end software delivery pipelines that improve deployment speed, software quality, and operational reliability.

## Prerequisites

To get the most value from this course, participants should have:

- Experience with software development
- Comfort working in terminal environments
- Knowledge of **Git** and **GitHub**
- Basic scripting skills (Shell, Python, or similar)
- Familiarity with software development workflows

Prior experience with Docker or Ansible is helpful but not required.

## Who Should Attend

This course is ideal for:

- **Software Developers**
- **DevOps Engineers**
- **Site Reliability Engineers (SREs)**
- **QA Engineers**
- **IT Operations Specialists**
- Platform Engineers
- System Administrators
- Automation-focused IT professionals
- Teams building and maintaining CI/CD pipelines

# CI/CD: Build, Test, Deploy Training

BILGINC IT Academy
info@bilginc.com
bilginc.com

Duration: **3 Day**

## What You Will Learn

By the end of this course, learners will be able to:

- Explain the business value of **CI/CD**.
- Differentiate between Continuous Integration, Delivery, and Deployment.
- Build automated pipelines using **GitHub Actions**.
- Implement automated testing workflows.
- Containerize applications using Docker.
- Automate deployments with **Ansible**.
- Secure CI/CD workflows using modern best practices.
- Apply monitoring and observability techniques.
- Design fully automated software delivery pipelines.

## Training Outline

### Introduction to DevOps

#### DevOps Fundamentals

- What is DevOps?
- DevOps culture and principles
- Software development lifecycle
- Traditional development versus DevOps
- Collaboration and automation culture
- Real-world DevOps use cases

### Introduction to CI/CD

#### CI/CD Fundamentals

- **Continuous Integration (CI)**
- **Continuous Delivery (CD)**
- **Continuous Deployment (CDP)**
- Software delivery workflows
- Business value of automation
- Feedback loops
- CI/CD adoption challenges

#### CI/CD Tools and Architectures

- Pipeline architectures
- Workflow design
- Modern DevOps toolchains
- CI/CD ecosystems

### CI/CD Pipeline Stages

#### Source Control Management

- Source code management
- Git workflows
- Branching strategies

#### Build Stage

- Purpose of the build process
- Compilation workflows
- Packaging applications
- Build automation techniques

#### Test Stage

- Automated testing approaches
- Unit testing
- Integration testing
- Managing test results

#### Release and Deployment

- Release management
- Environment management
- Deployment strategies
- Rollback mechanisms

#### Operations and Monitoring

- Operational workflows
- Monitoring systems
- Performance tracking
- Continuous improvement

### GitHub Actions Automation

#### GitHub Actions Fundamentals

- **GitHub Actions**
- Workflow architecture
- Events and triggers
- YAML syntax
- Repository organization

#### Workflow Development

- Creating build workflows
- Pipeline design
- Logging and troubleshooting
- Workflow optimization

### Docker and Containerization

#### Docker Fundamentals

BİLGİNÇ IT Academy

info@bilginc.com

bilginc.com


- **Docker Architecture**
- Container technologies
- Container lifecycle management
- Docker ecosystem

#### Dockerfile Management

- Building Dockerfiles
- Image creation and tagging
- Registry management
- Container security best practices

#### Docker in CI/CD Pipelines

- Automated image creation
- Container deployments
- CI/CD integration strategies

### Test Automation

#### Testing Strategies

- Types of testing
- Quality assurance principles
- Test automation methodologies

#### Testing Tools

- **PyTest**
- **Jest**
- GitHub Actions integration
- Test reporting and analytics

### Deployment Automation

#### Deployment Concepts

- Deployment environments
- Release workflows
- Canary and Blue-Green deployments

#### GitHub Actions Deployments

- Automated deployments
- Docker image deployment
- Environment management

### Ansible Automation

#### Ansible Fundamentals

- **Ansible Architecture**

BİLGİNÇ IT Academy

info@bilginc.com

bilginc.com


- Inventory management
- SSH connectivity
- Playbook development

#### Deployment Scenarios

- Configuration management
- Application deployment
- Infrastructure automation

#### GitHub Actions and Ansible Integration

- Workflow integration
- **dawidd6/action-ansible-playbook**
- Secure secrets management
- Automated inventory handling

### CI/CD Security

#### Pipeline Security

- **DevSecOps** practices
- Secure software delivery
- Security controls and validation

#### Secret Management

- **GitHub Secrets**
- Environment variables
- Credential management
- Avoiding hardcoded configurations

#### Compliance and Approvals

- Compliance controls
- Approval workflows
- Security governance

### Monitoring and Best Practices

#### Monitoring Solutions

- **Prometheus Fundamentals**
- Pipeline monitoring
- Application performance monitoring
- Operational visibility

#### Performance Optimization

- Caching strategies
- Pipeline acceleration techniques
- Resource optimization

BILGINC IT Academy

info@bilginc.com

bilginc.com


- Scalability best practices

### CI/CD Best Practices

- Reliable deployment workflows
- Reducing deployment failures
- Automation standards
- Sustainable DevOps methodologies