DevOps Fundamentals - Continuous Integration (CI)

[**🔍 1. Purpose** 1](#_Toc204936474)

[**📘 2. Theory** 1](#_Toc204936475)

[**🧰 3. Prerequisites** 2](#_Toc204936476)

[**🔧 4. Step-by-Step: Example with Azure Pipelines** 2](#_Toc204936477)

[**📸 5. Snapshot: CI Pipeline Flow** 3](#_Toc204936478)

[**✅ 6. Summary** 3](#_Toc204936479)

**🔍 1. Purpose**

To understand **Continuous Integration (CI)** — a key DevOps practice where code changes are automatically built, tested, and integrated frequently into a shared repository.

**📘 2. Theory**

**🔹 What is Continuous Integration?**

**Continuous Integration (CI)** is the practice of:

* Regularly **merging** code changes into a central repository (e.g., main or dev branch),
* Automatically **building** and **testing** the code after each commit.

This allows teams to detect issues early, reduce integration problems, and deliver software faster.

**🔹 Why CI is Important:**

| **🔧 Benefit** | **💡 Explanation** |
| --- | --- |
| 🧪 Early Bug Detection | Automated tests catch issues with every commit. |
| 🔄 Faster Feedback | Developers know immediately if their code broke the build. |
| 📦 Frequent Builds | Teams can deliver working software at any point. |
| ⚙️ Automation | Reduces manual steps and human errors. |

**🔹 Typical CI Workflow:**

Developer pushes code →

CI tool detects changes →

Build is triggered →

Automated tests run →

Feedback is sent (Pass/Fail)

**🔹 Common CI Tools:**

| **Tool** | **Platform** |
| --- | --- |
| Azure Pipelines | Azure DevOps |
| Jenkins | Open Source |
| GitHub Actions | GitHub |
| GitLab CI/CD | GitLab |
| CircleCI | Cloud-native |

**🧰 3. Prerequisites**

* Git repository with branching strategy
* A CI tool configured (e.g., Azure Pipelines or GitHub Actions)
* Basic unit tests in the codebase
* YAML or GUI pipeline setup

**🔧 4. Step-by-Step: Example with Azure Pipelines**

**Scenario**: Automatically build a .NET project when code is pushed to GitHub.

**✅ Step 1: Connect Azure DevOps to GitHub**

* Go to [Azure DevOps](https://dev.azure.com/)
* Create a new project
* Create a new pipeline and select **GitHub** as source

**✅ Step 2: Add a azure-pipelines.yml file in root:**

trigger:

- main

pool:

vmImage: 'windows-latest'

steps:

- task: UseDotNet@2

inputs:

packageType: 'sdk'

version: '7.0.x'

- script: dotnet build --configuration Release

displayName: 'Build Project'

- script: dotnet test

displayName: 'Run Tests'

**✅ Step 3: Commit and Push**

* The pipeline will be triggered automatically on commit to main branch.

**✅ Step 4: Monitor Pipeline**

* Check build status, test pass/fail, and logs from Azure DevOps dashboard.

**📸 5. Snapshot: CI Pipeline Flow**

GitHub Repo (push) → Azure DevOps Pipeline → Build → Run Tests → Notify Developer

**✅ 6. Summary**

| **Key Point** |
| --- |
| **Continuous Integration (CI)** automates the process of building and testing code with every change, enabling teams to detect issues early, speed up delivery, and maintain high code quality. |