DevOps Fundamentals - Continuous Delivery - Deployment (CD)

[**🔍 1. Purpose** 1](#_Toc204937435)

[**📘 2. Theory** 1](#_Toc204937436)

[**🧰 3. Prerequisites** 2](#_Toc204937437)

[**🔧 4. Step-by-Step: Example Using Azure Pipelines** 2](#_Toc204937438)

[**📸 5. Snapshot: CI/CD Flow** 3](#_Toc204937439)

[**✅ 6. Summary** 3](#_Toc204937440)

**🔍 1. Purpose**

To understand **Continuous Delivery (CD)** and **Continuous Deployment**—two key DevOps practices that ensure code is always in a **deployable state** and can be released to production at **any time**, either **manually** or **automatically**.

**📘 2. Theory**

**🔹 What is Continuous Delivery?**

Continuous Delivery means that every code change is **automatically built, tested, and prepared for release** to production.

Manual approval is required before deploying to production.

**🔹 What is Continuous Deployment?**

Continuous Deployment takes Continuous Delivery one step further—**every successful change is automatically deployed** to production **without human intervention**.

| **Feature** | **Continuous Delivery** | **Continuous Deployment** |
| --- | --- | --- |
| Deploys after CI | ✅ | ✅ |
| Manual approval required | ✅ (before production) | ❌ (fully automated) |
| Automation level | High | Very High |
| Risk of failure | Lower (manual checks) | Higher (if untested) |

**🔹 Key Benefits:**

| **🚀 Benefit** | **💡 Explanation** |
| --- | --- |
| Faster Release Cycles | Ship features to users quicker |
| Reduced Deployment Risks | Small, frequent changes reduce risk |
| Improved Developer Confidence | Teams know the state of production at all times |
| Less Manual Work | Automation removes repetitive deployment steps |

**🧰 3. Prerequisites**

* CI already configured
* Deployment environments defined (e.g., staging, production)
* Infrastructure ready (App Services, Containers, etc.)
* Optional: Deployment approval gates (for CD, not Deployment)

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

**✅ Step 1: Add CD to existing pipeline**

Extend your azure-pipelines.yml:

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'

- task: AzureWebApp@1

inputs:

azureSubscription: '<your-subscription-service-connection>'

appType: 'webApp'

appName: '<your-app-name>'

package: '$(System.DefaultWorkingDirectory)/\*\*/\*.zip'

This step **automatically deploys** to Azure App Service if the build and test succeed.

**✅ Step 2: Optional Manual Approval (for Delivery)**

Use **Environments** in Azure DevOps → Add **approval before deploy** to production.

**✅ Step 3: Monitor Release**

Track deployment status in Azure Pipelines → Environments tab.

**📸 5. Snapshot: CI/CD Flow**

Developer Commit →

CI Build/Test →

CD (Prepare Artifact) →

✅ Manual Approval (Delivery)

OR

🚀 Auto Deploy (Deployment)

**✅ 6. Summary**

| **Key Takeaway** |
| --- |
| **Continuous Delivery** ensures your code is always **ready to deploy** with manual controls, while **Continuous Deployment** goes further by **automatically releasing** changes to production. Both increase speed, reduce risk, and improve agility. |