Agile Devops Tools – Azure YAML – Pipeline as Code

[1. Purpose 1](#_Toc206156930)

[2. Theory 1](#_Toc206156931)

[3. Prerequisites 2](#_Toc206156932)

[4. Code Example (Step-by-Step) 2](#_Toc206156933)

[✅ Step 1: Create azure-pipelines.yml in your repo root 2](#_Toc206156934)

[✅ Step 2: Push to Git repo 3](#_Toc206156935)

[✅ Step 3: View pipeline in Azure DevOps 3](#_Toc206156936)

[5. Project Structure Snapshot 3](#_Toc206156937)

[6. Summary 3](#_Toc206156938)

✅ Task 7: Azure YAML – Pipeline as Code

## 1. Purpose

To define Azure Pipelines using YAML (Yet Another Markup Language), allowing your CI/CD processes to be version-controlled, automated, and managed directly in source code.

## 2. Theory

**🔹 What are YAML Pipelines?**

YAML pipelines are a way to define Azure DevOps build and release pipelines using code (azure-pipelines.yml), replacing classic UI-based pipelines.

**🔹 Why use YAML pipelines?**

* **Version-controlled** with code (in Git).
* **Reusable** templates and steps.
* **Scalable** across environments and projects.
* Supports **multi-stage pipelines** (build, test, deploy).

**🔹 Basic Structure:**

trigger:

- main # Branch name

pool:

vmImage: 'windows-latest'

variables:

buildConfiguration: 'Release'

steps:

- task: DotNetCoreCLI@2

inputs:

command: 'build'

projects: '\*\*/\*.csproj'

**🔹 Key Concepts:**

* **Trigger**: When to run the pipeline (e.g., on commit to a branch).
* **Pool**: VM or agent used for the job.
* **Variables**: Reusable values.
* **Steps**: Actions like build, test, publish.
* **Stages & Jobs**: Define separation of environments or phases.
* **Templates**: Reusable YAML files.

## 3. Prerequisites

* Basic understanding of Azure DevOps.
* Git repository in Azure Repos.
* Azure DevOps Project with Pipelines enabled.
* Codebase with .NET, Node.js, or other tech (optional for hands-on).

## 4. Code Example (Step-by-Step)

### ✅ Step 1: Create azure-pipelines.yml in your repo root

trigger:

- main

pool:

vmImage: 'windows-latest'

variables:

buildConfiguration: 'Release'

steps:

- task: UseDotNet@2

inputs:

packageType: 'sdk'

version: '7.x'

installationPath: $(Agent.ToolsDirectory)/dotnet

- task: DotNetCoreCLI@2

inputs:

command: 'restore'

projects: '\*\*/\*.csproj'

- task: DotNetCoreCLI@2

inputs:

command: 'build'

projects: '\*\*/\*.csproj'

arguments: '--configuration $(buildConfiguration)'

- task: DotNetCoreCLI@2

inputs:

command: 'test'

projects: '\*\*/\*Tests/\*.csproj'

### ✅ Step 2: Push to Git repo

* Azure DevOps detects the pipeline and will prompt to create/run it.

### ✅ Step 3: View pipeline in Azure DevOps

* Go to **Pipelines > Runs** and check logs/output.

## 5. Project Structure Snapshot

student-course-api/

│

├── Controllers/

├── Models/

├── Services/

├── azure-pipelines.yml ✅

└── student-course-api.csproj

## 6. Summary

| **Feature** | **YAML Pipelines** |
| --- | --- |
| Config Type | Code (YAML format) |
| Trigger | Commit-based, PRs, schedules |
| Benefits | Version control, reusability |
| Example Tools | DotNetCLI, npm, bash, PowerShell |
| Best Practice | Use templates, split stages/jobs |