Azure DevOps Tutorial for Beginners

Introduction

Overwhelmed by complicated development pipelines or frustrated when trying to orchestrate large fragmented teams? Progressive software development requires effective collaboration and automation.

Azure DevOps features a fully integrated set of tools to plan, develop, test, and deploy applications to any cloud-including Azure. This tutorial will break down concepts like Boards, Repos, and Pipelines into manageable steps, giving you the power to implement efficient Continuous Integration and Continuous Delivery.

Ready to simplify your development process? Download our complete **Azure DevOps Course Syllabus** now and begin your automation journey!

Why Students or Freshers Learn Azure DevOps?

Learning Azure DevOps imposes considerable career benefits on a fresher:

- **Integrated CI/CD Mastery:** It provides a broad, centralized platform for Continuous Integration/Continuous Delivery, CI/CD-a necessary skill for all modern software roles.
- **Industry Adoption:** Backed by Microsoft, it has wide enterprise adoption in project management on Boards and cloud deployments on Pipelines to Azure and other clouds.
- **Automation Focus:** You learn how to automate every stage of the lifecycle, from code commits to deployment and monitoring, drastically increasing efficiency and reducing manual errors.
- **Collaboration Tools:** Mastery of Azure Boards and Azure Repos attests to proficiency in agile planning and source control, showcasing Git functionalities necessary for a team environment.

Ready to land a DevOps or Cloud Engineering role? Here's a curated list of top **Azure DevOps interview questions and answers** to showcase your pipeline and automation experience. Download now!

Step-by-Step Azure DevOps Tutorial for Beginners

Welcome to Azure DevOps, a suite of tools provided by Microsoft encompassing the practices of DevOps from agile planning and source code management to automated build and release pipelines (CI/CD). Since Azure DevOps is a SaaS platform, there's no large installation, only setup and configuration. We're going to focus on setting up your organization and creating your first CI/CD pipeline.

Step 1: Account Setup and Organization Creation

You will need a Microsoft account in order to access and set up your Azure DevOps organization.

1.1 Creating a Microsoft Account

Create a free Microsoft Account if you don't already have one (e.g., Outlook, Hotmail).

1.2 Create the Azure DevOps Organization

- **Navigate:** To proceed, access the official Azure DevOps site and log in with your Microsoft account.
- **Start:** Click the button to Start free.
- **Name:** Provide a unique name for your organization, such as MyDevOpsProjectOrg. The URL will look like this: https://dev.azure.com/MyDevOpsProjectOrg
- **Confirm:** Finish the setup.

1.3 Creating the First Project

Once inside the organization:

1. Click **New Project.**
2. **Project Name:** Enter a name, for example, WebAppCI-CD
3. **Visibility:** Private – recommended if this is work-related
4. **Version Control:** Choose Git. It's the industry standard.
5. **Work Item Process:** Choose Scrum or Agile selects the structure of the planning board.
6. **Action:** Click Create.

Step 2: Preparation of the Application Code (Source Control)

Before you can create a pipeline, you'll need an application in a repository. We'll use the .NET Core Console App from the previous tutorial as our source code.

2.1 Create a local Git

Make sure that Git is installed on your local computer.

Action (Local PC): Create a sample .NET Console app, like in the previous tutorial, or just a simple Program.cs file in a local folder:

 # Create the app (if you don't have one)

dotnet new console -n SampleApp

cd SampleApp

2.2 Initialize and Commit Locally

Initialize the local folder as a Git repository and commit the code.

 # Initialize git

git init

 # Stage all files

git add .

 # Commit the initial files

git commit -m "Initial commit of the sample console application"

2.3 Connect to Azure Repos

- **Azure DevOps:** Within your project, select Repos within WebAppCI-CD. Azure DevOps automatically creates a blank repository with the same name as your project.

- **Copy URL:** From Azure DevOps, copy the remote URL provided-for example, *https://MyDevOpsProjectOrg@dev.azure.com/MyDevOpsProjectOrg/WebAppCI-CD/_git/WebAppCI-CD.*

2.4 Push Code to Azure Repos

Add the remote and push your code. You'll be prompted to sign in with your Microsoft account credentials.

 # Add the Azure Repo as the remote origin

git remote add origin https://MyDevOpsProjectOrg@dev.azure.com/MyDevOpsProjectOrg/WebAppCI-CD/_git/WebAppCI-CD

 # Push the local 'main' branch to the remote 'origin'

git push -u origin main

Verification: To verify, check the Repos section in Azure DevOps. You should now see your Program.cs and project files there.

Step 3: Setting up the Build Pipeline (Continuous Integration – CI)

Every push to the repository automatically compiles the code, runs tests, and validates it via the Build Pipeline.

3.1 Creating a New Pipeline

1. **Azure DevOps:** In the navigation menu, select Pipelines > Pipelines.
2. Click **New Pipeline**.
3. **Connect:** Select Azure Repos Git.
4. **Repository:** Select your WebAppCI-CD repository.

5. **Configure:** Select .NET Desktop (we are building a console app, but this template is fitting).

3.2 Define the Pipeline YAML

Azure DevOps employs YAML, YAML Ain't Markup Language, to define pipelines as code. These are stored in your repository. The template provided will yield a simple CI flow.

We'll simplify and focus the YAML file to just restore, build, and publish the artifact.

```
# azure-pipelines.yml

# Triggers the pipeline on every commit to the main branch

trigger:

– main

pool:

  vmImage: 'windows-latest' # Use a hosted Microsoft agent

steps:

– task: UseDotNet@2 # Ensure the correct .NET version is used

  displayName: 'Use .NET SDK'

  inputs:

    version: '8.x' # Use the latest .NET version

– task: DotNetCoreCLI@2
```

```
  displayName: ‘Restore’
  inputs:
    command: ‘restore’
    projects: ‘**/*.csproj’
– task: DotNetCoreCLI@2
  displayName: ‘Build’
  inputs:
    command: ‘build’
    projects: ‘**/*.csproj’
    arguments: ‘–configuration Release’
# 1. Publish the built application binaries as an Artifact
– task: DotNetCoreCLI@2
  displayName: ‘Publish’
  inputs:
    command: ‘publish’
    publishWebProjects: false
    arguments: ‘–configuration Release –output $(Build.ArtifactStagingDirectory)’
```

zipAfterPublish: true # Zips the output folder for easier transfer

 # 2. Upload the artifact (the zipped application) so the Release pipeline can use it

– task: PublishBuildArtifacts@1

displayName: 'Upload Artifact'

inputs:

PathtoPublish: '$(Build.ArtifactStagingDirectory)'

ArtifactName: 'drop'

- **Action:** Save the YAML file – it will commit directly to your main branch.
- **Verification:** The pipeline should immediately queue and start running. In the Azure DevOps interface, monitor the progress. Once complete, under the Summary tab there should be an available artifact named drop.

Step 4: Setting up the Release Pipeline (Continuous Delivery – CD)

The Release Pipeline, or CD, takes the compiled application, the Artifact, from the result of the CI step and deploys it to a target environment – for example, a server, Azure App Service, etc. For this console app, we will simulate a deployment by showing you the classic release process.

4.1 Creating a New Classic Release Pipeline

- **Azure DevOps:** In the navigation menu, select Pipelines > Releases.
- Click **New Pipeline**.
- **Template:** Select Empty job (to start simple).
- **Stage Name:** DevEnvironment.

4.2 Link the Build Artifact

- In the Release Pipeline editor, click the **+ Add an artifact box**.
- **Source Type:** Select Build.
- **Source (Build pipeline):** *MyWebApp* or whatever you named the created CI pipeline.
- **Alias:** Use the *default _MyWebApp*.
- **Enable Continuous Deployment Trigger:** On the artifact, click the **lightning bolt icon** to open the **Continuous deployment trigger** and set it to **Enabled**. This way, every time your CI build succeeds, a new deployment automatically begins.

4.3 Define Deployment Steps

Now define what happens at the DevEnvironment stage.

- Click the link under **Stage 1** that says **1 job, 0 task**.
- **Agent's Job:** The Agent Specification needs to be set to windows-latest.
- **Add a Task:** Click the + next to the Agent job. Search for **PowerShell** and add the PowerShell task.

4.4 Simulate Deployment using PowerShell

We will use PowerShell to simulate the deployment just by listing the files downloaded from the artifact.

PowerShell Task Configuration

Type: Select **Inline**.

Enter the following script:

Write-Host "— Deployment Started for DevEnvironment —"

 # Define the path where the artifact was downloaded

$artifactPath = “$(System.DefaultWorkingDirectory)/_MyWebApp/drop”

Write-Host “Checking contents of artifact folder: $artifactPath”

 # List the files to prove the application was successfully deployed/transferred

Get-ChildItem -Path $artifactPath -Recurse

Write-Host “— Deployment Finished Successfully —“

Save: Saves the Release Pipeline.

Step 5: Triggering Full CI/CD Pipeline

The whole process is automated now, whenever code is being pushed – building and release, when successful, run.

5.1 Manual Code Change

Local PC: Open your *Program.cs* file and make a slight modification.

// Program.cs

// Change the output message

Console.WriteLine(“Hello, Azure DevOps CI/CD is working!”);

Commit and Push the Change:

git add .

git commit -m “Updated message to test CI/CD trigger”

git push origin main

5.2 Monitor the CI/CD Flow

- **Azure DevOps Pipelines > Pipelines:** The Build Pipeline should automatically start to run. Follow it until it succeeds (compiles the code and creates the artifact).
- **Azure DevOps Pipelines > Releases:** Following completion of build, the Release Pipeline should pick up this new artifact and trigger a deployment automatically to the DevEnvironment stage.
- **Verification:** Check the logs of the Release pipeline's PowerShell task. The output should include the list of the compiled files of your application, which means simulation of automated deployment is done.

That's it! You've just successfully created your first CI/CD pipeline with Azure DevOps. You have completely automated the path from committing code to deploying it.

Step 6: Introduction to Azure Boards

Azure Boards is utilized for Agile project management, including tracking work items and planning sprints.

6.1 Create Work Items

1. **Azure DevOps:** *Boards → Work Items.*
2. Click *New Work Item* and select *User Story* or ***Product Backlog Item***, depending on your template.
3. **Title:** Provide a title, such as "Implement user login feature".
4. **Assign:** Assign the item to yourself.
5. **State:** New

6.2 Plan a Sprint

1. **Azure DevOps:** Go to *Boards → Sprints*
2. Define the dates of your first Sprint, for example, 2 weeks.
3. **Backlog:** Drag the *"Implement user login feature"* User Story from the Backlog into your new Sprint.
4. **Action:** When you begin working on the work item, use the Boards view to drag the work item from New to Active.

Step 7: Next Steps in Azure DevOps

You have mastered core CI/CD and planning tools. Further learning should focus on:

- **Real Deployment:** The deployment of the application to an actual hosting service, like Azure App Service or to a VM by using specific deployment tasks.
- **Testing:** Adding activities that run automated Unit Tests or Integration Tests during the Build Pipeline.
- **Infrastructure as Code:** Integrating Terraform or ARM Templates to automatically provision the cloud resources required for deployment.

Congratulations! You just automated your first DevOps pipeline! Download our complete **Azure DevOps Challenges and Solutions** Guide! Work through exercises involving creating multistage pipelines, integrating automated testing, setting up secure variables, and deploying to various cloud targets with professional YAML solutions.

Real Time Examples for Azure DevOps Tutorial for Beginners

Azure DevOps is the integrated platform enterprises use to manage their entire software development lifecycle, from planning to production.

Cloud Deployment Automation (CI/CD Pipelines)

- **Scenario:** A development team commits a code change to the main branch of an application hosted on an Azure App Service.

- **Azure DevOps Applied:** A CI Pipeline automatically detects the commit, runs unit tests, compiles the application, and then creates a deployment artifact. The successful CI triggers the CD Pipeline, which uses the artifact to automatically deploy the new version to either the staging or production environment, without any manual intervention.
- **Goal:** achieve fast, reliable, and repeatable releases – Continuous Delivery

Agile Project Management and Tracking (Azure Boards)

- **Scenario:** A Product Owner needs to track the progress of features, bugs, and tasks across a two-week sprint.
- **Azure DevOps Applied:** The team leverages Azure Boards (most often set to the Scrum process) to identify User Stories, assign these to a developer, estimate effort, and track state (New, Active, Resolved, Closed). Automatic Sprint Burndown Chart reveals remaining work to keep the project on schedule.
- **Objective:** Be transparent, keep the product backlog under control, and change priorities fast.

Infrastructure as Code Integration

- **Scenario:** A new testing environment, including a Virtual Machine and database, should be provisioned before the application is deployed on it.
- **Azure DevOps Applied:** One of the configuration tasks for the CD Pipeline is to run Terraform or ARM Templates stored in Azure Repos. The IaC task, prior to deployment, sets up the requisite cloud resources in Azure. A subsequent deployment task will then deploy to those newly created resources.
- **Objective:** Have consistent, documented environments provisioned automatically on demand.

Ready to apply these concepts and build your own pipelines? Download our curated list of **Azure DevOps Project Ideas** to practice CI/CD, Boards, and IaC integration!

FAQs About Azure DevOps Tutorial for Beginners

1. What are the 7C's of DevOps?

These are core DevOps principles focused on automation and improvement across the lifecycle. The 7Cs are: Culture, Collaboration, Continuous Integration (CI), Continuous Delivery (CD), Continuous Testing, Continuous Monitoring, and Continuous Learning.

2. What are the 5 pillars of Azure DevOps?

The five components, or services, of Azure DevOps include Azure Boards for planning, Azure Repos for source control, Azure Pipelines for CI/CD, Azure Test Plans for manual/exploratory testing, and Azure Artifacts for package management.

3. Can I learn DevOps in 3 months?

Yes, that's ambitious. And, yes, you can learn the basics in 3 months with studying 30-40 hours a week: Git, basic Linux, Docker, and at least one of the CI/CD tools like Azure Pipelines. You will only reach complete proficiency by working hands-on with cloud (AWS/Azure) and IaC (Terraform).

4. What are the 9 pillars of DevOps?

While common models use 5 or 7, a nine-pillar model is often expanded upon for security and infrastructure. These usually include Leadership, Culture, Design for DevOps, Continuous Integration, Continuous Testing, Elastic Infrastructure, Continuous Monitoring, Continuous Security, and Continuous Delivery.

5. Is DevOps IT or CS?

DevOps is both an IT and a CS hybrid. It requires CS knowledge for scripting and understanding application architecture, and it also requires IT operations skills for managing infrastructure, networking, and system reliability.

6. Will AI replace DevOps?

It will not replace the DevOps engineers; it will change their role. AI tools automate monitoring, anomaly detection, and basic troubleshooting that were previously done by a person. In its place, the role of DevOps will evolve to architecting and integrating AI services and managing complex systems.

7. What is DevOps' job salary?

Generally, DevOps Engineers command a very good salary because of the specialized skill set. In India, an experienced professional often gets an average of ₹12.5 Lakhs per annum or more, which may rise considerably with expertise in specific clouds like Azure and AWS, and leadership roles. Explore **Azure DevOps salary for freshers and experienced** here.

8. Is DevOps harder than coding?

It is seen that DevOps has a higher learning curve than basic coding. If coding requires deep logic, DevOps requires extensive multi-domain skills in areas such as coding, networking, cloud platforms, automation tools, security, and constant troubleshooting.

9. Is Azure DevOps a CI tool?

Azure DevOps is a complete platform, and Azure Pipelines is its main CI/CD tool. It automates the whole software process that comprises Continuous Integration for building and testing, and Continuous Delivery/Deployment for releasing the application.

10. Can I use Azure for free?

Yes, Microsoft provides an Azure Free Account, granting $200 credit for the first 30 days for new users, plus 65+ Always Free services like Azure DevOps for up to 5 users and certain compute/database tiers with limited usage caps.

Conclusion

You've walked through the very center of Azure DevOps: setting up your organization, managing your code with Azure Repos, and mastering the automation of the CI/CD pipeline. This set of integrated skills, from planning on Boards to continuous deployment, is what really enables modern development. You are now ready to tackle real-world complexity, security, and scalability challenges in cloud deployments. Ready to be recognized as an expert in cloud automation and release management? Enroll in our advanced **Azure DevOps Course in Chennai** to master multi-stage environments, testing integration, and Infrastructure as Code!