



Azure Pipelines

1. What is Azure Pipelines?

Definition

Azure Pipelines is a **CI/CD service** in Azure DevOps used to:

- Build code
- Run tests
- Package applications
- Deploy to environments (VMs, App Service, AKS, etc.)

Why Enterprises Use Azure Pipelines

- Cloud & on-prem support
- YAML + Classic pipelines
- Deep integration with Azure Repos, GitHub, Artifacts
- Enterprise security, approvals, auditing



AZURE

The screenshot shows the Azure DevOps Pipelines interface. On the left, there's a sidebar with options like Test, Overview, Boards, Repos, Pipelines, Environments, Library, Test Plans, and Artifacts. The 'Pipelines' option is selected. The main area features a cartoon illustration of a person working at a laptop with a dog nearby. Below the illustration, the text 'Create your first Pipeline' is displayed, followed by a subtitle: 'Automate your build and release processes using our wizard, and go from code to cloud-hosted within minutes.' A prominent blue button labeled 'Create Pipeline' is at the bottom.

2. Azure Pipelines Architecture

Core Components

- **Pipeline**
- **Agent**
- **Stages**
- **Jobs**
- **Steps**
- **Tasks**

Flow

Code Commit → Pipeline Trigger → Agent → Tasks → Artifacts → Deployment

3. Types of Pipelines

3.1 Build Pipeline (CI)

- Compiles code
- Runs unit tests
- Publishes artifacts

3.2 Release Pipeline (CD – Classic)

- Deploys build artifacts
- Uses environments & approvals

3.3 YAML Pipeline (Modern – Recommended)

- CI + CD in one file
- Version controlled

4: Pipeline Infrastructure & Execution Control

(*Azure DevOps Enterprise Execution Layer*)

This explains how Azure Pipelines actually run, where jobs execute, how access is secured, and how enterprises control speed, scale, and permissions.

4.1 Azure Pipeline Agents – Execution Engines

What is an Agent?

An **Azure Pipeline Agent** is a **machine that executes pipeline jobs** such as:

- Building code
- Running tests
- Packaging artifacts
- Deploying applications

Pipelines do not run inside Azure DevOps.

They always run **on agents**.

Examples:



AZURE

- ubuntu-latest
- windows-latest

4.2 Agent Pools

What is an Agent Pool?

An **Agent Pool** is a **collection of agents** that pipelines can use to run jobs.

Agent pools help enterprises:

- Reuse agents
- Control access
- Isolate workloads
- Scale execution

Types of Agent Pools

1. Microsoft-Hosted Agent Pools

- Managed by Microsoft
- No maintenance required
- New VM for every run
- Recommended for learning and cloud-native workloads



Examples:

- `ubuntu-latest`
- `windows-latest`
- `macos-latest`

2. Self-Hosted Agent Pools (Enterprise Usage)

- Installed on customer-managed servers or VMs
- Used when:
 - Internal network access is required
 - Custom tools are needed
 - Compliance policies exist

Why Enterprises Use Multiple Agent Pools

- Separate **Build** and **Deployment** agents
- Restrict **Production deployments**
- Assign agents to specific teams

Example:

- `Build-Agent-Pool`
- `Deploy-Agent-Pool`
- `Prod-Restricted-Pool`



4.3 Agent Pool Settings & Management

Key Agent Pool Settings

Setting	Purpose
Pool Name	Logical identification
Agent Type	Hosted or self-hosted
Permissions	Who can use/manage
Auto-Provision	Managed by Microsoft
Capabilities	OS, tools, software

Agent Capabilities

Capabilities define **what an agent can do**:

- OS type
- Installed software
- Custom labels

Used for:

- Matching jobs to correct agents
- Avoiding failures due to missing tools



4.4 Parallel Jobs (Speed & Scale Control)

What are Parallel Jobs?

Parallel jobs define **how many pipeline jobs can run at the same time**.

Parallel jobs = **Pipeline execution capacity**

Types of Parallel Jobs

Microsoft-Hosted Parallel Jobs

- Limited by Azure DevOps license
- Shared across organization
- Additional capacity requires paid plans

Self-Hosted Parallel Jobs

- Limited only by:
 - Number of agents
 - Server resources



AZURE

Why Parallel Jobs Matter in Enterprises

- Faster CI/CD
- Multiple teams running pipelines
- Reduced waiting time
- High productivity

Example:

- 1 parallel job → pipelines queue
- 10 parallel jobs → faster feedback

Enterprise Best Practice

- Use hosted agents for CI
- Use self-hosted agents for CD
- Allocate parallel jobs per team

4.5 Service Connections (Secure Access Layer)

What is a Service Connection?

A **Service Connection** is a **secure authentication method** that allows Azure Pipelines to connect to external systems.

Pipelines never use user credentials directly.

Why Service Connections Are Mandatory

- Secure authentication
- Centralized access control
- Audit and compliance
- No hard-coded secrets

Common Service Connection Types

Service	Purpose
Azure Resource Manager	Deploy to Azure
GitHub	Access GitHub repos
Docker Registry	Push/pull images
Kubernetes	Deploy to AKS
Generic	External APIs

Service Connection Security (Enterprise Level)

- RBAC controlled
- Scoped to:
 - Subscription
 - Resource group
- Approval-based usage
- Auditable access



Execution Flow





4.6 Real Enterprise Design Example

Scenario

- 50 developers
- 10 pipelines
- Production compliance required

Setup

- Microsoft-hosted agents → Build
- Self-hosted agents → Deployment
- Separate prod agent pool
- Limited service connection access
- Parallel jobs allocated per team



AZURE

A

The image displays three screenshots of the Azure DevOps interface, illustrating the configuration of a project and its pipelines.

Project Settings (Top Screenshot):

- Project details:** Shows a project named "Test".
 - Name:** Test
 - Description:** (Empty)
 - Process:** Basic
- Project administrators:** Lists "vinal kolluri" (vk) as the administrator, with the email "kollurivinalchowdary@gmail.com". A "Add administrator" button is present.

Agent pools (Middle Screenshot):

- Project Settings:** Shows the "Agent pools" section selected in the sidebar.
- Agent pools list:** Displays two agent pools:
 - Azure Pipelines:** Queued jobs: 0, Running jobs: 0
 - Default:** Queued jobs: 0, Running jobs: 0

Agent pool Details (Bottom Screenshot):

- Project Settings:** Shows the "Agent pools" section selected in the sidebar.
- Agent pool details:** Shows the "Default" pool.
 - Jobs:** (Empty)
 - Agents:** (Empty)
 - Details:** (Empty)
 - Security:** (Empty)
 - Approvals and checks:** (Empty)
 - Analytics:** (Empty)
- Add your first agent:** A call-to-action button: "New agent".

Contact : +91 9966107782
info@sdlctechacademy.com



AZURE

The screenshot shows the Azure DevOps interface for managing pipelines. On the left, the sidebar has 'Test' selected. Under 'Pipelines', 'Agent pools' is also selected. A modal window titled 'Get the agent' is open, specifically for the 'Windows' tab (x64). The window contains instructions for configuring the account, downloading the agent (with a 'Download' button), creating the agent (showing PowerShell commands to extract and run the agent setup), and optional interactive runs (showing another PowerShell command). At the top right of the modal are 'Update all agents' and 'New agent' buttons.

AZURE

The screenshot displays two Azure Project Settings pages. The top page shows the 'Agent pools' section under 'Pipelines'. It lists a single 'Hosted Agent' named 'Hosted Agent' which is currently 'Idle'. The status bar indicates it is 'Online'. The bottom page shows the 'Settings' section under 'General'. It includes a 'Retention policy' section with fields for 'Days to keep artifacts, symbols and attachments' (30), 'Days to keep runs' (30), 'Days to keep pull request runs' (10), and 'Number of recent runs to retain per pipeline' (3). It also includes sections for 'General' settings like 'Disable anonymous access to badges' (On), 'Limit variables that can be set at queue time' (On), and 'Limit job authorization scope to current project for non-release pipelines' (On).

Contact : +91 9966107782
info@sdlctechacademy.com

AZURE



kollurvinichowdary / Test / Settings / Parallel jobs

Project Settings

Test

- General
- Overview
- Teams
- Permissions
- Notifications
- Service hooks
- Dashboards
- Boards
- Project configuration
- Team configuration
- Github connections
- Pipelines
- Agent pools
- Parallel jobs
- Settings
- Test management
- Service connections
- XAML build services

Private projects

Hosted	Parallel jobs
Microsoft-hosted	0
Monthly purchases	0 Change
Self-hosted	1
Free parallel jobs	1
Visual Studio Enterprise subscribers	0
Monthly purchases	0 Change

Public projects

Hosted	Parallel jobs
Microsoft-hosted	0
Self-hosted	Unlimited
View in progress jobs	

kollurvinichowdary / Test / Settings / Service connections

Project Settings

Test

- General
- Overview
- Teams
- Permissions
- Notifications
- Service hooks
- Dashboards
- Boards
- Project configuration
- Team configuration
- Github connections
- Pipelines
- Agent pools
- Parallel jobs
- Settings
- Test management
- Service connections
- XAML build services

Create your first service connection

Service connections help you manage, protect, and reuse authentications to external services.

Create service connection

SDLC

AZURE

The screenshot shows the Azure Project Settings interface for a project named 'Test'. The left sidebar lists various settings sections: General, Boards, Pipelines, and Repos. Under Pipelines, 'Service connections' is selected, which is highlighted with a blue border. The main area displays a placeholder for a user profile picture and the text 'Create your first service connection'. Below this, a subtitle reads 'Service connections help you manage, protect, and reuse authentications to external services.' A prominent blue button labeled 'Create service connection' is centered. To the right, a modal window titled 'New service connection' is open, listing various service connection types with their corresponding icons. The listed services include:

- Azure Repos/Team Foundation Server
- Azure Resource Manager
- Azure Service Bus
- Bitbucket Cloud
- Cargo
- Chef
- Docker Host
- Docker Registry
- Generic
- GitHub
- GitHub Enterprise Server
- Incoming WebHook
- Jenkins

At the bottom right of the modal is a blue 'Next' button.

AZURE

New GitHub service connection

Authentication method

Grant authorization
 Personal Access Token

Authentication

OAuth Configuration

Authorise

Service connection details

Service Connection Name

Description (optional)

Security

Grant access permission to all pipelines

[Learn more](#) [Troubleshoot](#)

[Back](#) [Save](#)



5. Pipeline Creation – Step by Step (YAML)

Step 1: Go to Pipelines

- Azure DevOps → Pipelines → New Pipeline

Step 2: Select Code Source

- Azure Repos Git
- GitHub
- Bitbucket



AZURE

The screenshot shows the 'Where is your code?' step in the Azure DevOps Pipelines setup. The left sidebar is visible with 'Pipelines' selected. The top navigation bar shows 'Azure DevOps kollurivinaichowdary / Test / Pipelines'. The main area has tabs 'Connect', 'Select', 'Configure', and 'Review', with 'Select' being active. It displays options for connecting to code repositories: 'Azure Repos Git' (YAML), 'GitHub' (YAML), and 'Bitbucket Cloud' (YAML). A 'More options' dropdown is also present.

The screenshot shows the 'Select a repository' step. The left sidebar is visible with 'Pipelines' selected. The top navigation bar shows 'Azure DevOps kollurivinaichowdary / Test / Pipelines'. The main area has tabs 'Connect', 'Select', 'Configure', and 'Review', with 'Select' being active. It displays a search bar 'Filter by keywords' set to 'Test' and a list of repositories: 'practice' and 'testing'.

Step 3: Select Pipeline Template

- Starter pipeline
- ASP.NET
- Java
- Node.js



AZURE

Azure DevOps kollurvinainchowdary / Test / Pipelines

Test Overview Boards Repos Pipelines Pipelines Environments Library Test Plans Artifacts Project settings

✓ Connect ✓ Select Configure Review

New pipeline

Configure your pipeline

- Starter pipeline**
Start with a minimal pipeline that you can customize to build and deploy your code.
- Existing Azure Pipelines YAML file**
Select an Azure Pipelines YAML file in any branch of the repository.
- .NET Core Function App to Windows on Azure**
Build a .NET Core function app and deploy it to Azure as a Windows function App.
- .NET Desktop**
Build and run tests for .NET Desktop or Windows classic desktop solutions.
- Android**
Build your Android project with Gradle.
- Ant**
Build your Java projects and run tests with Apache Ant.
- ASP.NET**
Build and test ASP.NET projects.
- ASP.NET Core**
Build and test ASP.NET Core projects targeting .NET Core.
- ASP.NET Core (.NET Framework)**
Build and test ASP.NET Core projects targeting the full .NET Framework.
- C/C++ with GCC**
Build your C/C++ project with GCC using make.
- Deploy to Azure Kubernetes Service**
Build and push image to Azure Container Registry; Deploy to Azure Kubernetes Service

Azure DevOps kollurvinainchowdary / Test / Pipelines

Test Overview Boards Repos Pipelines Pipelines Environments Library Test Plans Artifacts Project settings

✓ Connect ✓ Select Configure Review

New pipeline

Configure your pipeline

- Starter pipeline**
Start with a minimal pipeline that you can customize to build and deploy your code.
- Existing Azure Pipelines YAML file**
Select an Azure Pipelines YAML file in any branch of the repository.
- .NET Core Function App to Windows on Azure**
Build a .NET Core function app and deploy it to Azure as a Windows function App.
- .NET Desktop**
Build and run tests for .NET Desktop or Windows classic desktop solutions.
- Android**
Build your Android project with Gradle.
- Ant**
Build your Java projects and run tests with Apache Ant.
- ASP.NET**
Build and test ASP.NET projects.
- ASP.NET Core**
Build and test ASP.NET Core projects targeting .NET Core.
- ASP.NET Core (.NET Framework)**
Build and test ASP.NET Core projects targeting the full .NET Framework.
- C/C++ with GCC**
Build your C/C++ project with GCC using make.
- Deploy to Azure Kubernetes Service**
Build and push image to Azure Container Registry; Deploy to Azure Kubernetes Service

Select an existing YAML file
Select an Azure Pipelines YAML file in any branch of the repository.
Branch: dev
Path: /Azurepipeline.yml
testing

Cancel Continue

Contact : +91 9966107782
info@sdlctechacademy.com



AZURE

The screenshot shows the Azure DevOps Pipelines interface. On the left, there's a sidebar with options like Test, Overview, Boards, Repos, Pipelines (which is selected), Pipelines, Environments, Library, Test Plans, and Artifacts. The main area is titled 'Review your pipeline YAML' and shows a code editor with the following YAML configuration:

```
trigger:
  branches:
    include:
      - dev
pool:
  vmImage: 'ubuntu-latest'
steps:
- script:
  - echo "Hello World"
  - echo "Azure Pipeline triggered from dev branch"
  displayName: 'Run Hello World Script'
```

At the top right, there are buttons for 'Variables', 'Run', and a dropdown. A 'Show assistant' button is also visible.

Step 4: YAML File Created

- `azure-pipelines.yml`
- Stored inside repo



6. YAML Pipeline Structure (Core Learning)

```
trigger:  
  - main  
  
pool:  
  vmImage: ubuntu-latest  
  
stages:  
  - stage: Build  
    jobs:  
      - job: BuildJob  
        steps:  
          - task: ExampleTask
```

Key Sections Explained

Key Sections Explained

Section	Purpose
trigger	When pipeline runs
pool	Agent selection
stages	High-level phases
jobs	Logical execution units
steps	Actual tasks

7. Triggers in Azure Pipelines

7.1 CI Trigger

- Runs on code commit

7.2 Branch Filters

- main
- develop
- feature/*

7.3 Path Filters

- Trigger only if specific folders change



7.4 Scheduled Triggers

- Nightly builds
- Weekly security scans

8. Variables & Variable Groups (Enterprise Usage)

8.1 Pipeline Variables

- Key-value pairs
- Used across pipeline

8.2 Variable Groups

- Centralized variables
- Shared across pipelines

8.3 Secret Variables

- Passwords
- Tokens
- Hidden in logs
- Secure Files

AZURE



Review your pipeline YAML

```

trigger:
- branches:
  - dev
pool:
  vmImage: 'ubuntu-latest'
steps:
- script: |
  echo "Hello World!"
  echo "Azure Pipeline triggered from dev branch"
  displayName: 'Run Hello world Script'

```

Variables

Use variables to store values or encrypted secrets separately from your repository.

New variable

Learn about variables

Library

Variable groups Secure files + Variable group Security Help

New variable group

Create groups of variables that you can share across multiple pipelines.

+ Variable group

Learn more about variable groups.

Test

Overview Boards Repos Pipelines Environments Library Test Plans Artifacts

Test

Overview Boards Repos Pipelines Environments Library Test Plans Artifacts

Test

Variable group Save Clone Pipeline permissions Approvals and checks Help

Properties

Variable group name: Test

Description:

Link secrets from an Azure key vault as variables

Variables

Name	Value
username	vinal
password	Vinay@7782

+ Add

AZURE

The screenshot shows the Azure DevOps interface for the 'Test' project. The left sidebar has 'Library' selected. The main area is titled 'Library' and shows the 'Secure files' tab is active. It features a large icon of a document with a lock, a button to 'Upload a secure file', and a note about uploading sensitive files like certificates and keys. A 'Secure file' button is highlighted in blue.

9. Secure Secrets with Azure Key Vault

Why?

- Enterprise security
- No hard-coded secrets

Flow

Pipeline → Variable Group → Key Vault → Secret

The screenshot shows the 'Variable group' configuration page for a group named 'New variable group 02-Jan'. The left sidebar has 'Library' selected. The main area shows fields for 'Variable group name' (set to 'New variable group 02-Jan'), 'Description' (empty), and a toggle switch for 'Link secrets from an Azure key vault as variables' which is turned on. Below these are dropdowns for 'Azure subscription' (set to 'Default') and 'Key vault name' (set to 'Invalid Value'). Error messages indicate both fields are required.



10. Tasks in Azure Pipelines

What is a Task?

- Prebuilt action
- Example:
 - Build
 - Copy files
 - Publish artifacts
 - Deploy

Types

- Built-in tasks
- Marketplace extensions
- Custom scripts

11. Artifacts (Build Outputs)

Why Artifacts?

- Store build output
- Used by deployment pipelines



Artifact Types

- Pipeline artifact
- Universal package

12. Multi-Stage Pipelines

Example Stages

- Build
- Test
- Deploy-Dev
- Deploy-QA
- Deploy-Prod

Benefits

- Single pipeline
- Full visibility
- Controlled promotion



13. Environments in Azure Pipelines

What is an Environment?

- Logical deployment target
- Example:
 - Dev
 - QA
 - Prod

Features

- Deployment history
- Approvals
- Resource tracking



AZURE

The screenshot shows the Azure DevOps interface for the 'Environments' section. The left sidebar has 'Test' selected. The main area is titled 'Environments' and lists three environments: 'dev', 'qa', and 'uat'. Each environment row includes a 'Status' column showing 'Never deployed' and a 'Last activity' column showing 'Just now'. A 'New environment' button is located at the top right of the table.

Environment	Status	Last activity
dev	Never deployed	Just now
qa	Never deployed	Just now
uat	Never deployed	Just now

14. Approvals & Gates

Manual Approvals

- Required before deployment

Automated Gates

- Health checks
- Monitoring validation

AZURE

Azure DevOps kollurvinai/chowdary / Test / Pipelines / Environments / uat

Test +

← uat

Deployments Approvals and checks

Add resource

Add your first check

Checks allow you to manage how this resource is used. Changes made to checks are effective immediately, applicable to all existing and new pipelines.

Approvals
Approvers should grant approval for deployment

Branch control
Allow deployments based on branches linked to the run

Business Hours
Ensure the deployment is started in a specific time window

View all checks

15. Deployment Strategies

Types

- RunOnce
- Rolling
- Canary
- Blue-Green (via YAML logic)



16. Service Connections

What is Service Connection?

- Secure connection to external systems

Examples:

- Azure subscription
- Docker registry
- Kubernetes cluster

17. Permissions & Security

Pipeline Security

- Who can edit pipeline
- Who can run pipeline

Environment Security

- Deployment approvals
- Role-based access



AZURE

The screenshot shows the Azure DevOps interface. On the left, the navigation bar includes Test, Overview, Boards, Repos, Pipelines (selected), Pipelines, Environments, Library, Test Plans, and Artifacts. The Pipelines section shows a pipeline named 'testing' with a status of 'Manually run by dev'. The main content area is titled 'Permissions for testing' under 'Pipelines'. It displays a list of users and groups with their permissions. The 'Inheritance' toggle is turned on. The list includes:

User/Group	Permission	Action
[kollurvinainachowdary]\Project Collection Administrators	Administer build permissions	Allow (inherited)
	Delete build pipeline	Allow (inherited)
	Delete builds	Allow (inherited)
	Destroy builds	Allow (inherited)
	Edit build pipeline	Allow (inherited)
	Edit build quality	Allow (inherited)
	Edit queue build configuration	Allow (inherited)
	Manage build qualities	Allow (inherited)
	Manage build queue	Allow (inherited)
	Override check-in validation by build	Allow (inherited)
	Queue builds	Allow (inherited)
	Retain indefinitely	Allow (inherited)
	Stop builds	Allow (inherited)
	Update build information	Not set
	View build pipeline	Allow (inherited)
	View builds	Allow (inherited)

18. Monitoring & Troubleshooting

Features

- Pipeline logs
- Task-level logs
- Timeline view

Common Failures

- Agent issues



AZURE

- Permission issues
- Variable not found

19. Pipeline Best Practices

- Use YAML pipelines
- Separate build & deploy stages
- Use Key Vault for secrets
- Enable approvals for prod
- Use reusable templates
- Enforce branch policies

20. Azure Pipelines vs Jenkins

Azure Pipelines	Jenkins
Fully managed	Self-managed
Azure native	Plugin dependent
Secure by default	Needs hardening

21. Real Enterprise Pipeline Flow

