Azure Pipelines

1. What is Azure Pipelines?

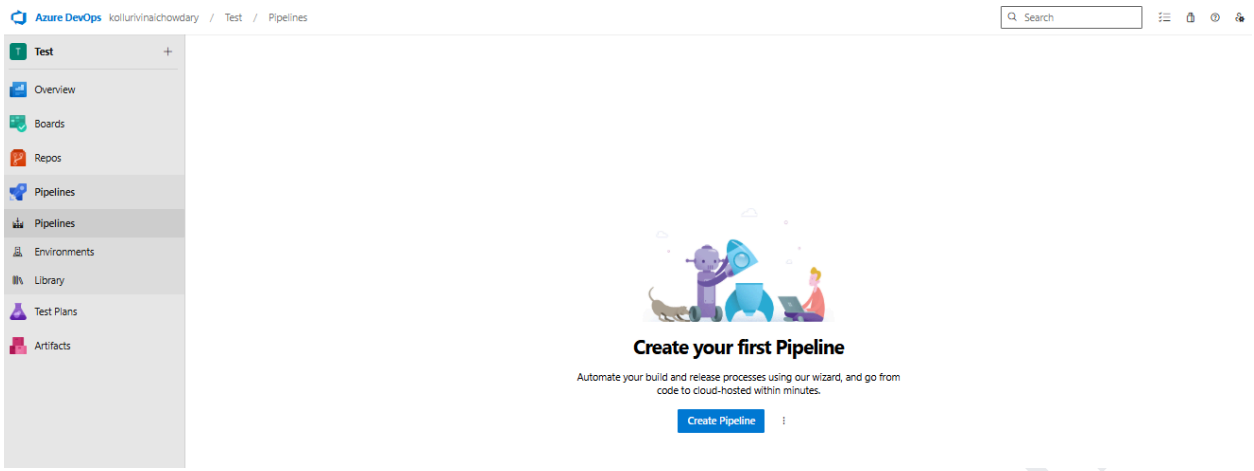
Definition

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



---

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:

- `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

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

Pipeline Trigger

↓

Agent Pool Selected

↓

Parallel Job Allocated

↓

Agent Executes Tasks

↓

Service Connection Authenticates

↓

Deployment Completed

4.6 Real Enterprise Design Example

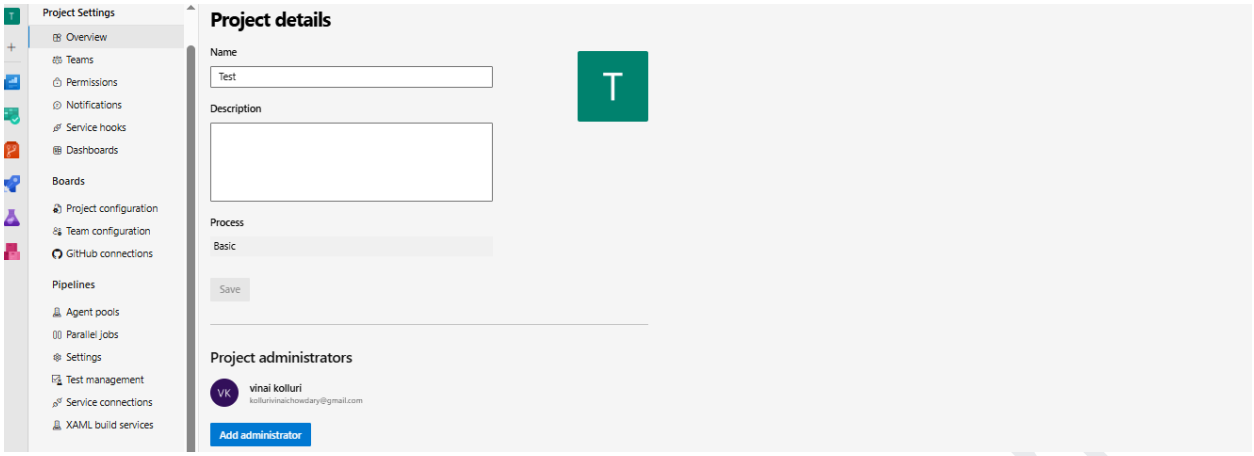
Scenario

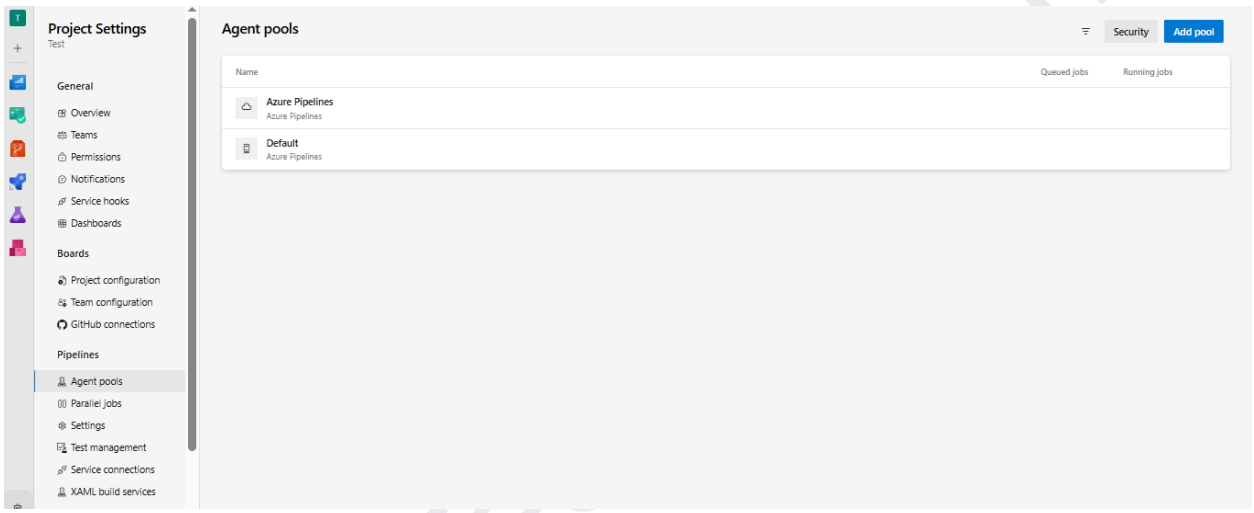
- 50 developers
- 10 pipelines
- Production compliance required

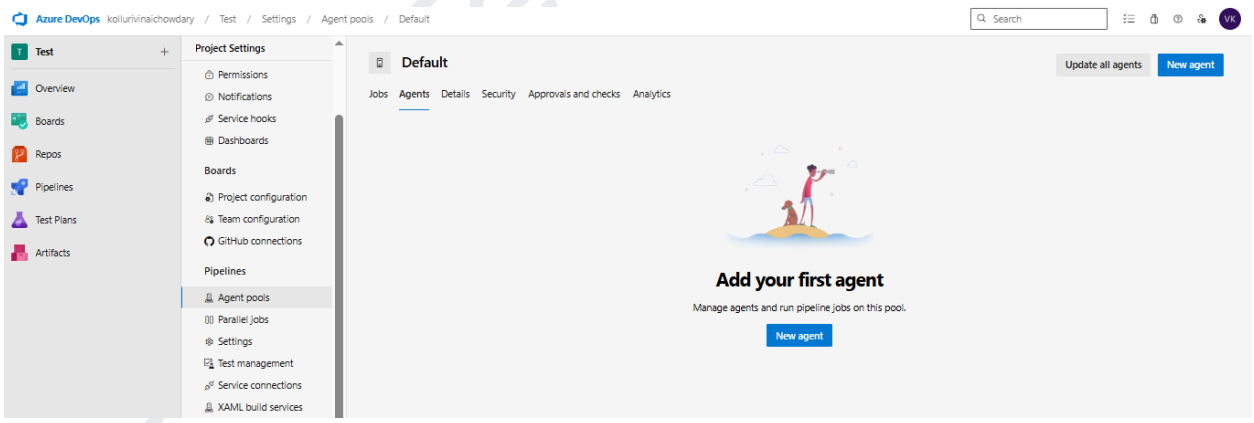
Setup

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

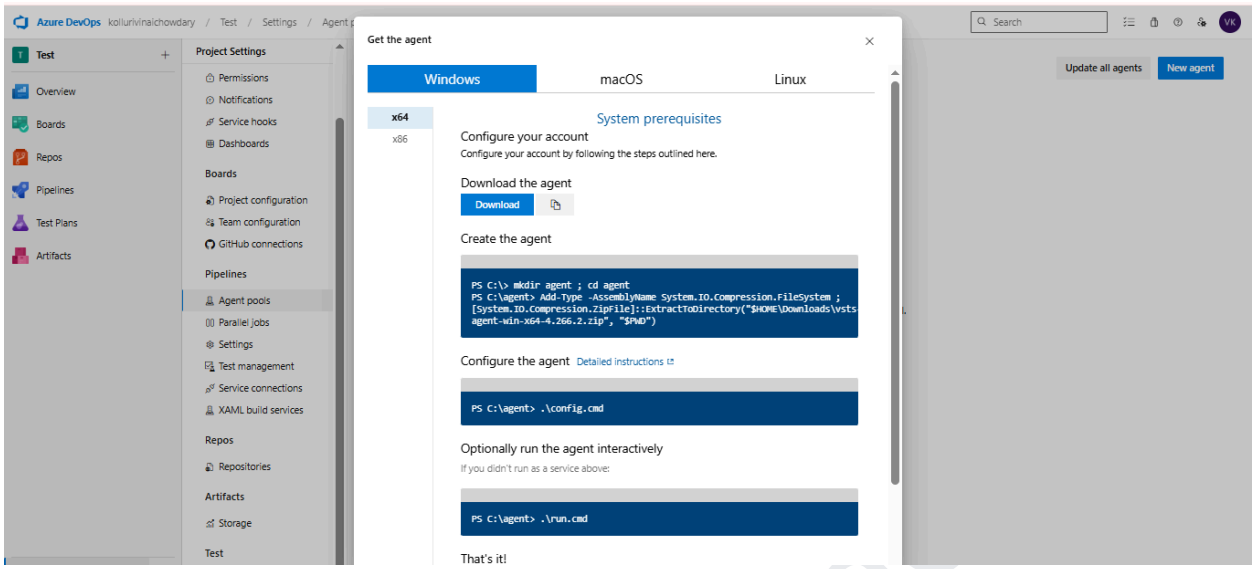
AZURE





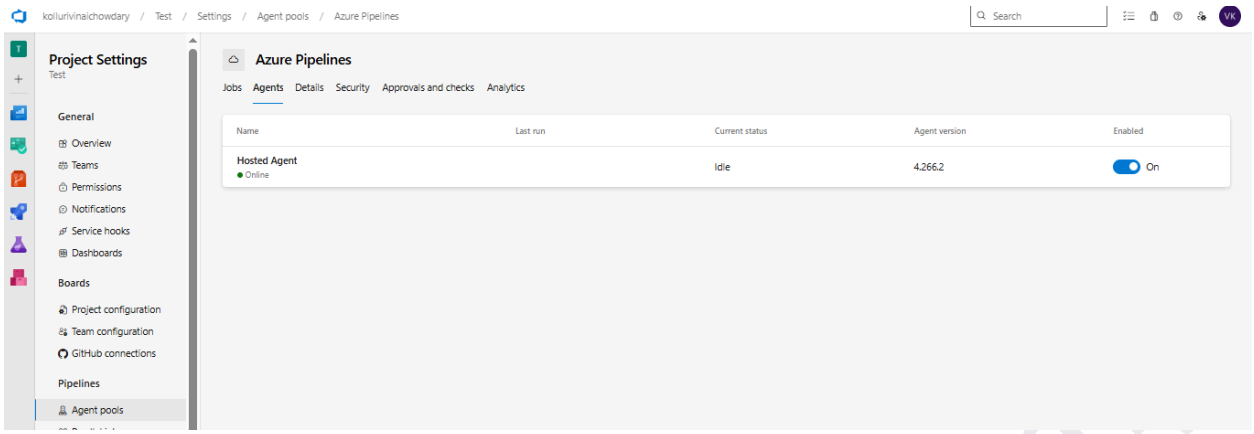


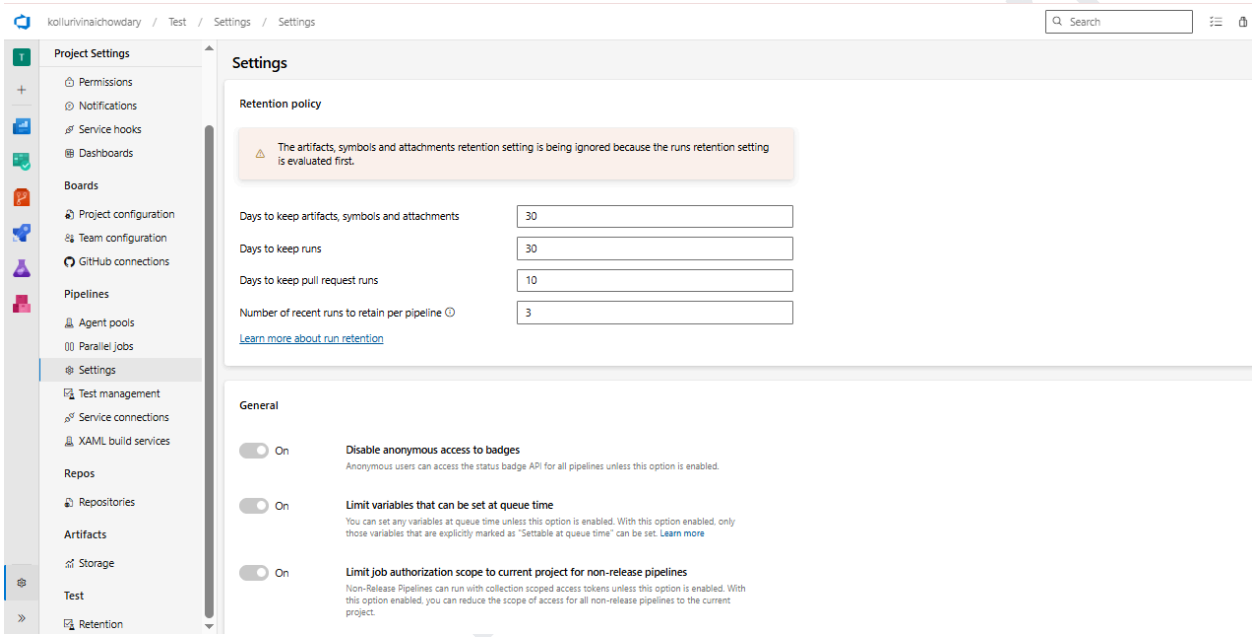
Contact : +91 9966107782
info@sdlctechacademy.com



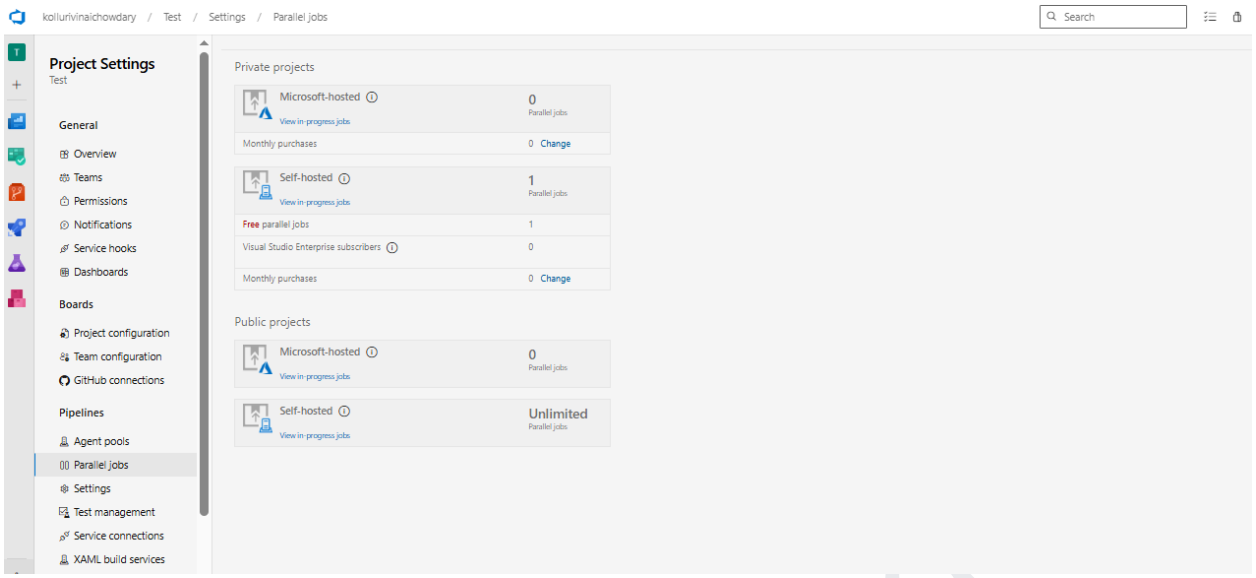
Contact : +91 9966107782
info@sdlctechacademy.com

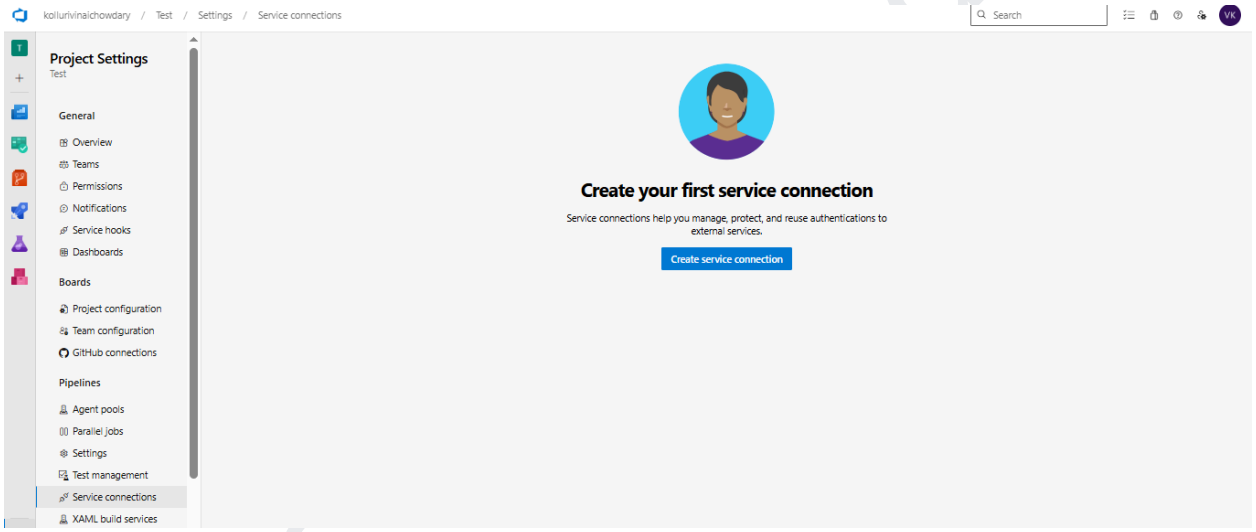
AZURE





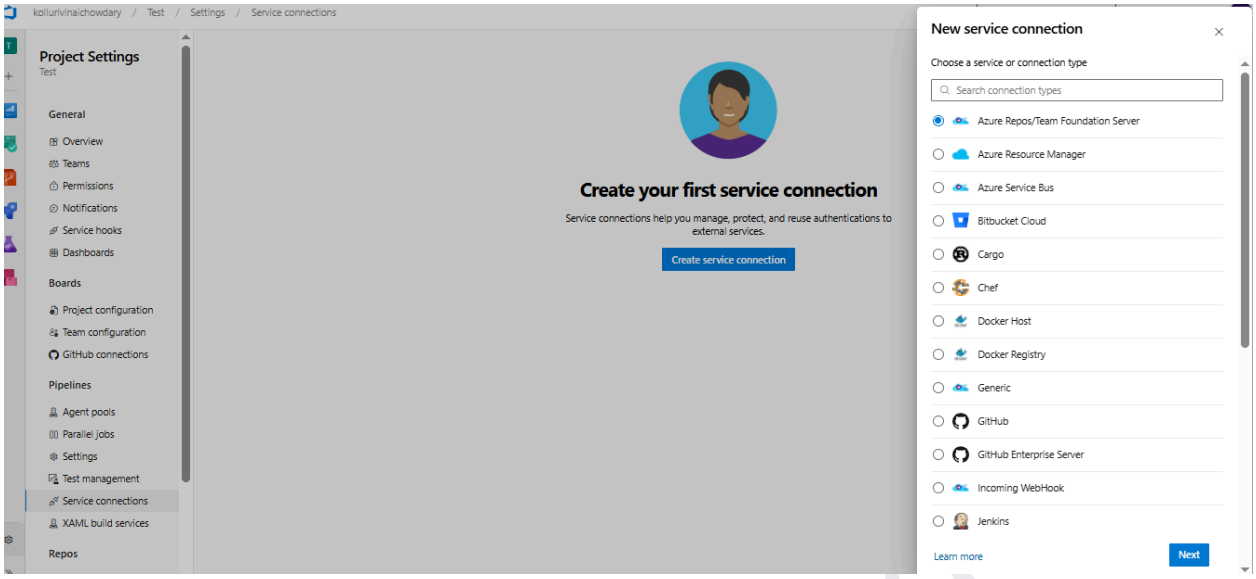
Contact : +91 9966107782
info@sdlctechacademy.com



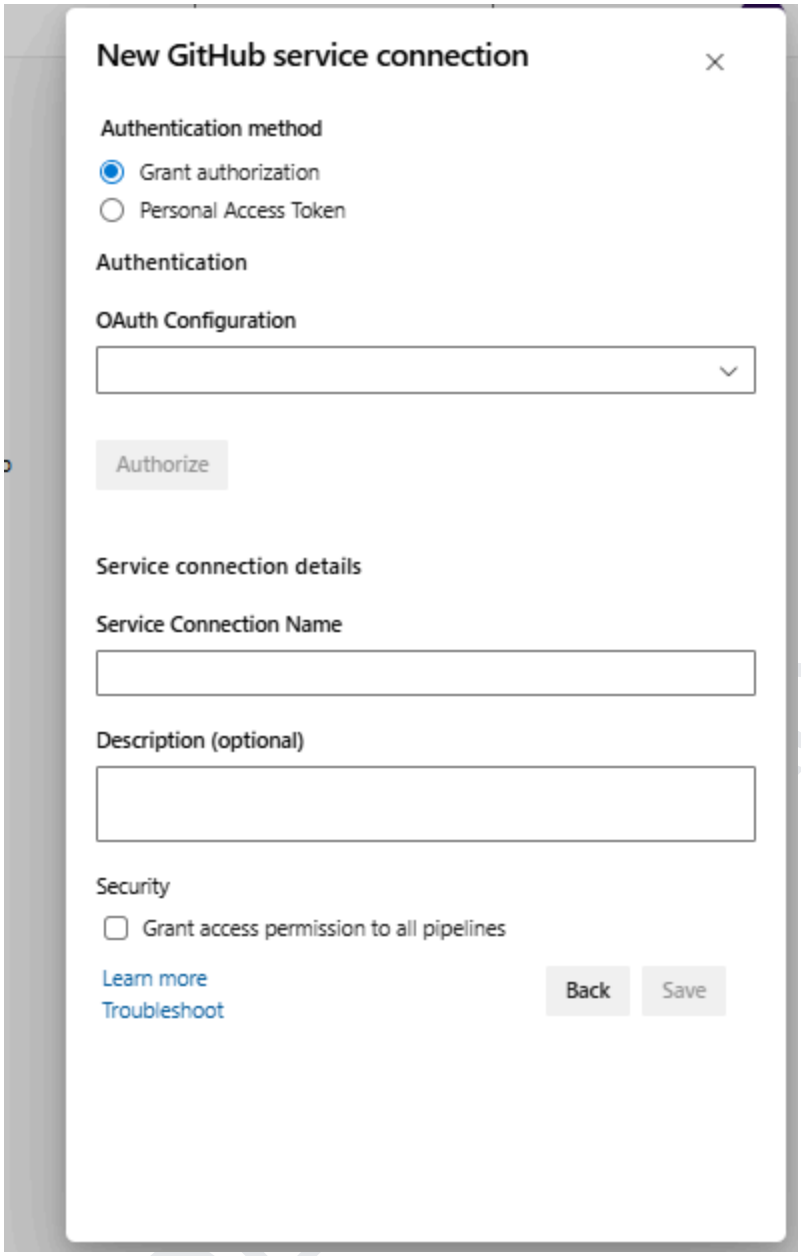


Contact : +91 9966107782
info@sdlctechacademy.com

AZURE



Contact : +91 9966107782
info@sdlctechacademy.com



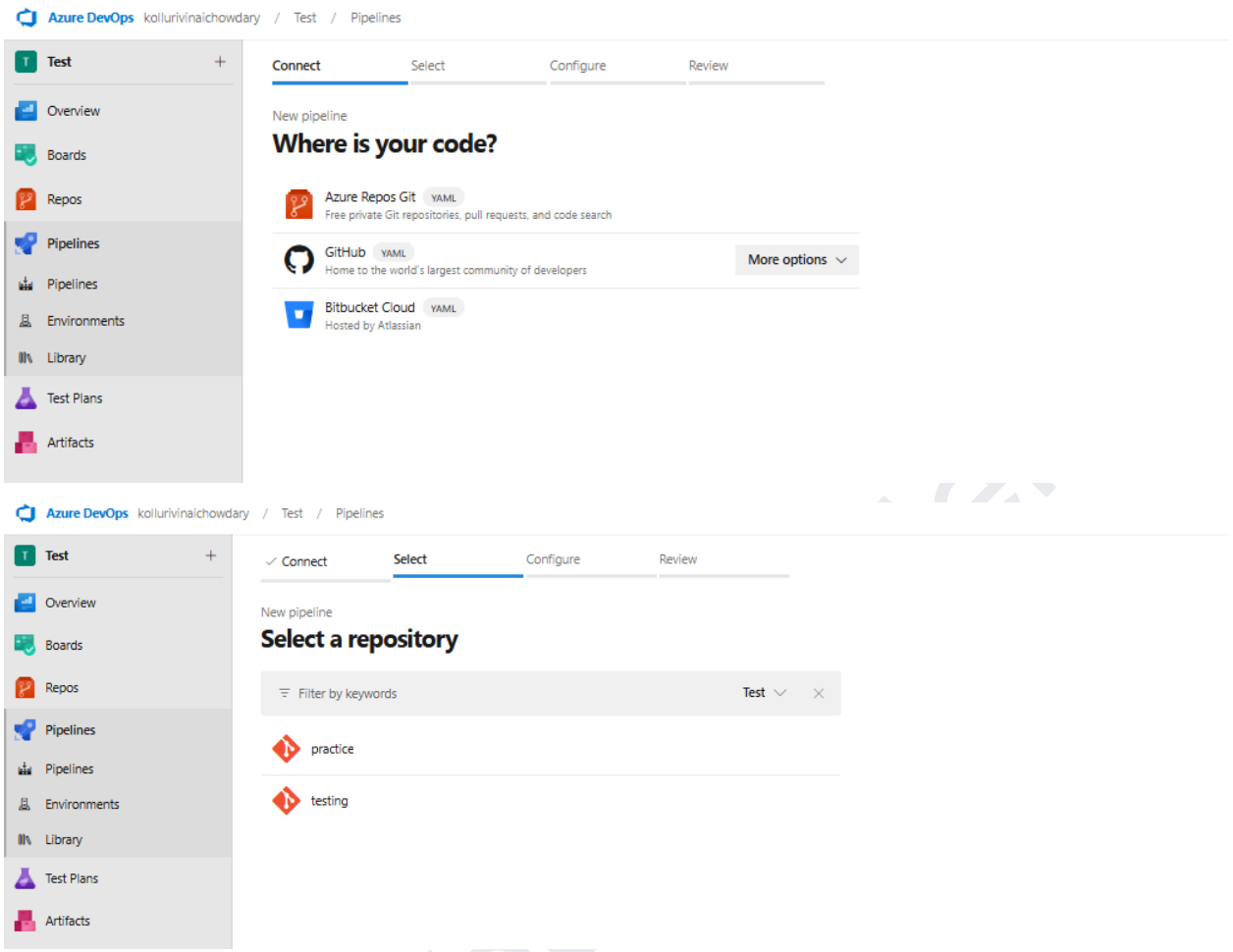
5. Pipeline Creation – Step by Step (YAML)

Step 1: Go to Pipelines

- Azure DevOps → Pipelines → New Pipeline

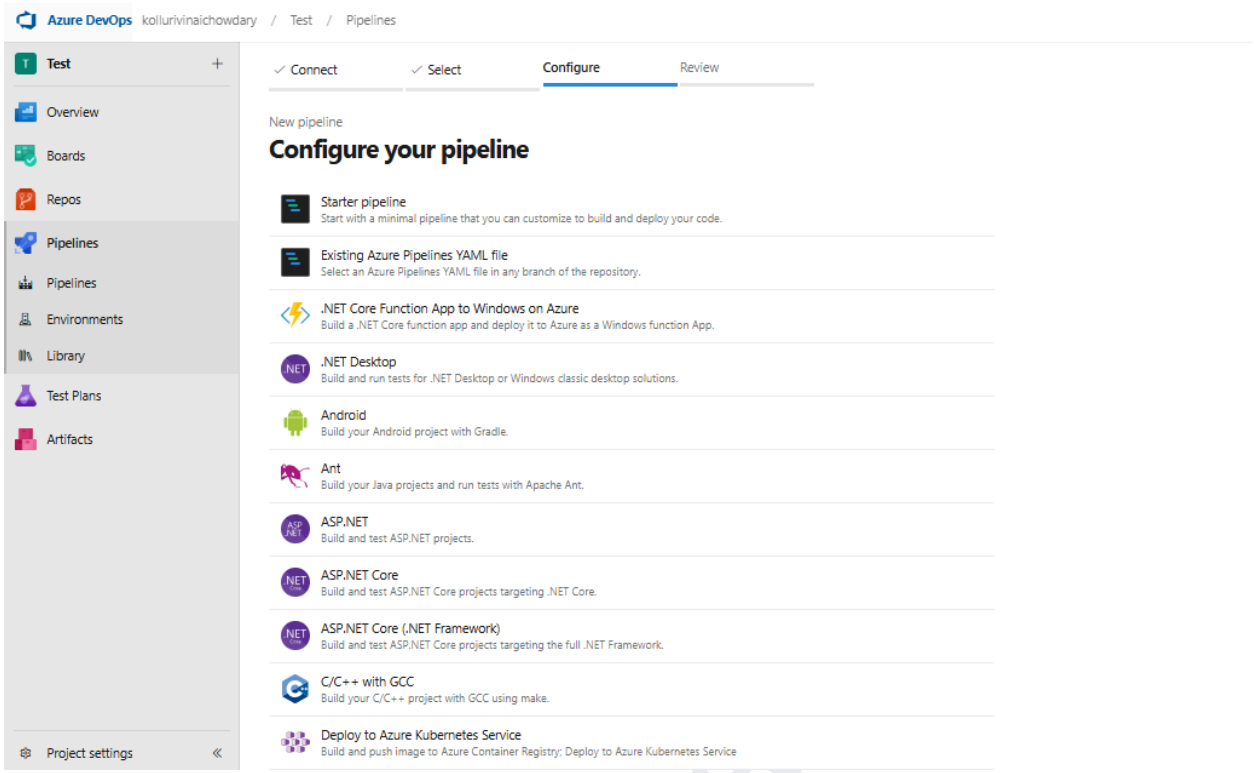
Step 2: Select Code Source

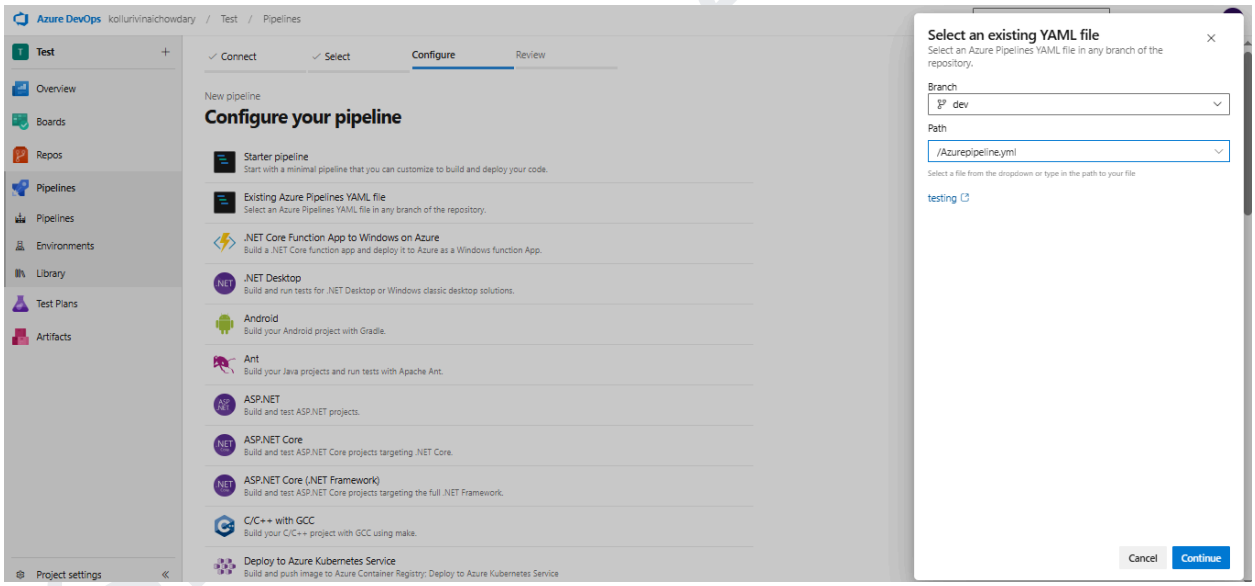
- Azure Repos Git
- GitHub
- Bitbucket



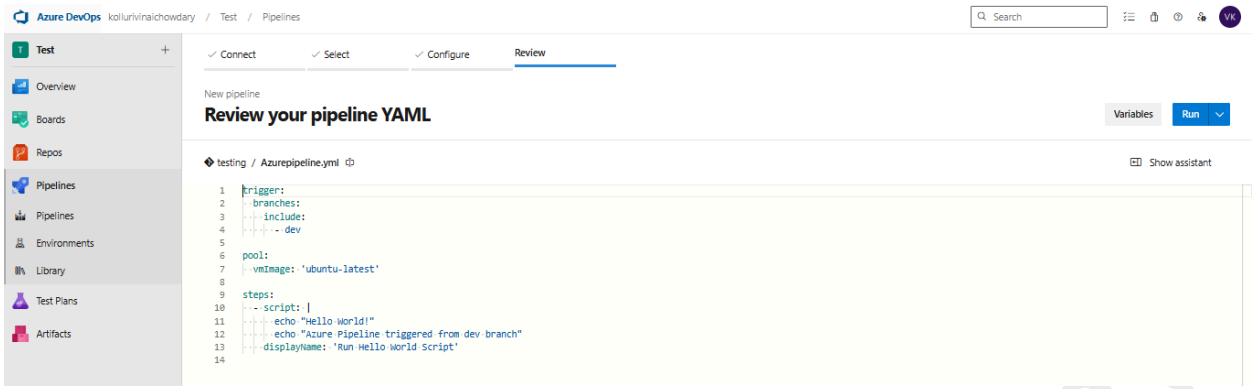
Step 3: Select Pipeline Template

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





Contact : +91 9966107782
info@sdlctechacademy.com



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

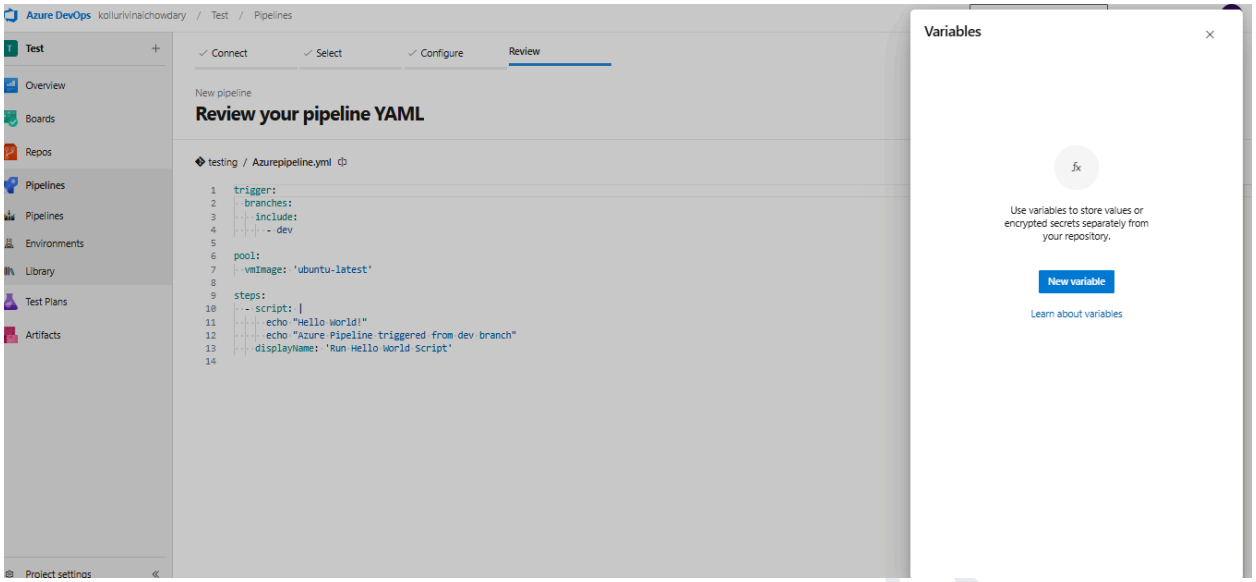
8.2 Variable Groups

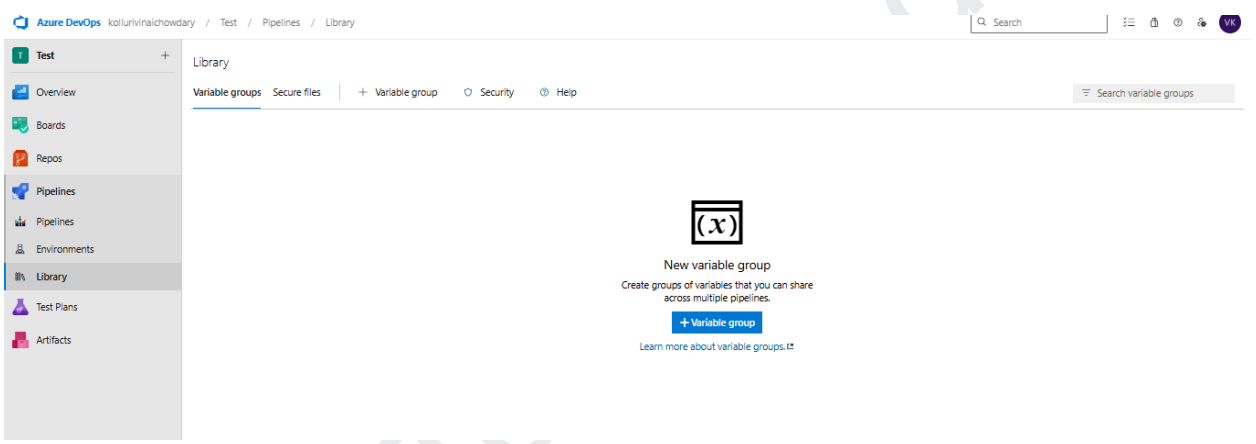
- Centralized variables
- Shared across pipelines

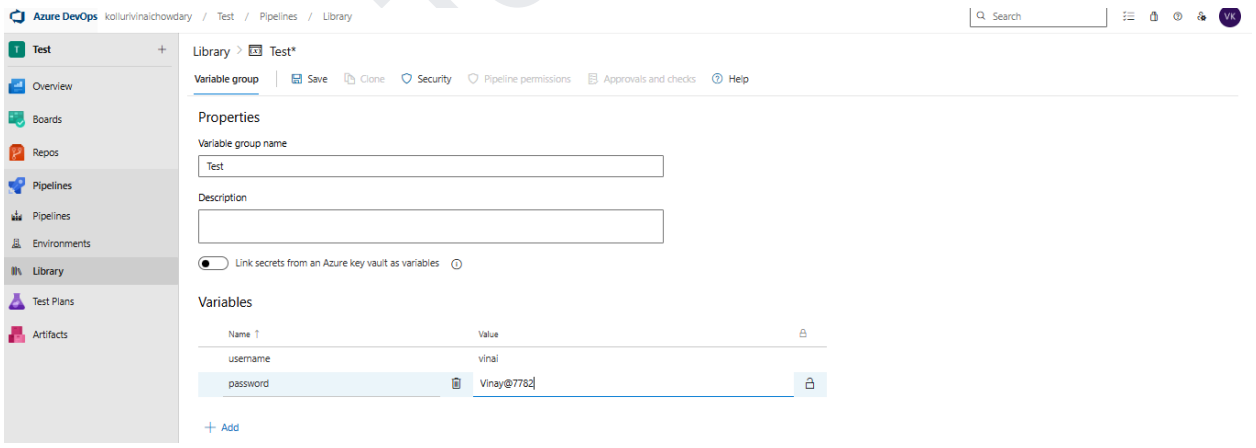
8.3 Secret Variables

- Passwords
- Tokens
- Hidden in logs
- Secure Files

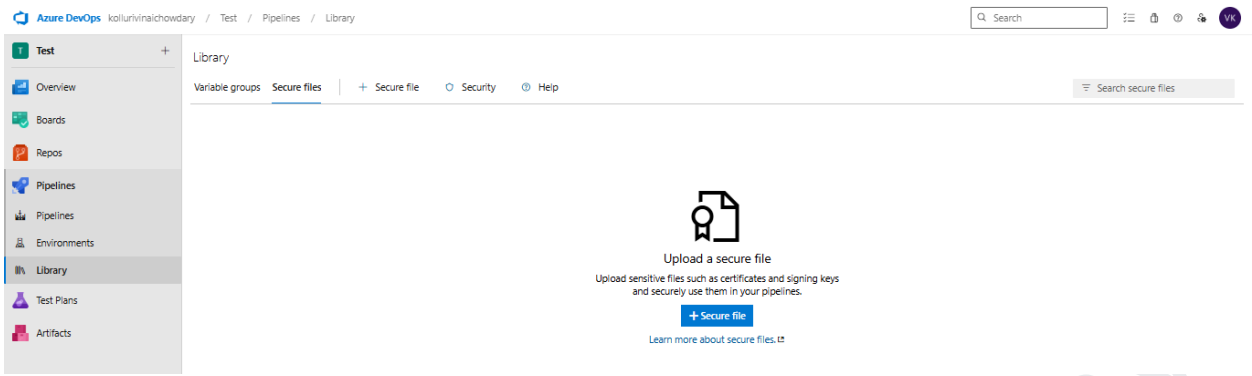
AZURE







Contact : +91 9966107782
info@sdlctechacademy.com



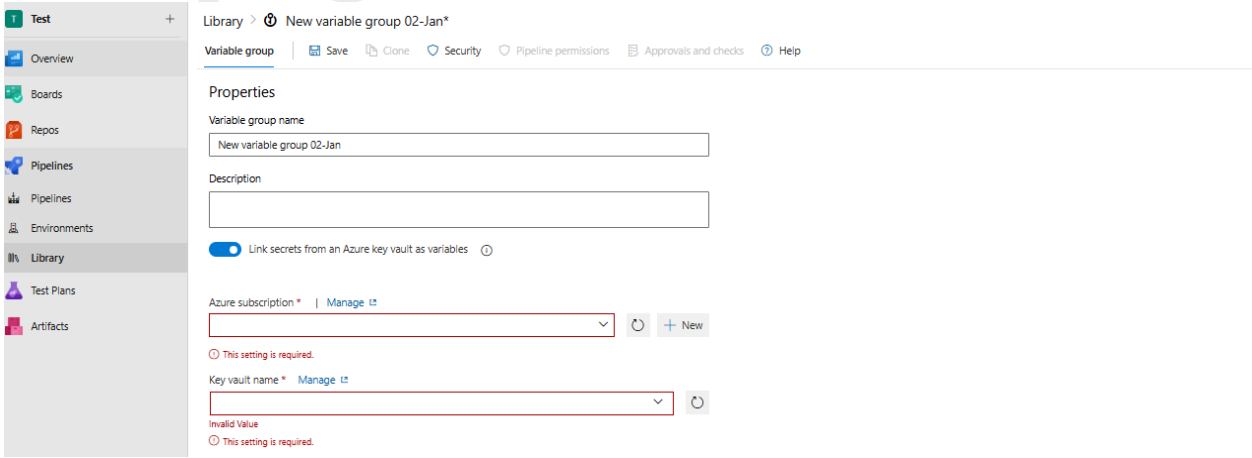
9. Secure Secrets with Azure Key Vault

Why?

- Enterprise security
- No hard-coded secrets

Flow

```
Pipeline → Variable Group → Key Vault → Secret
```



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

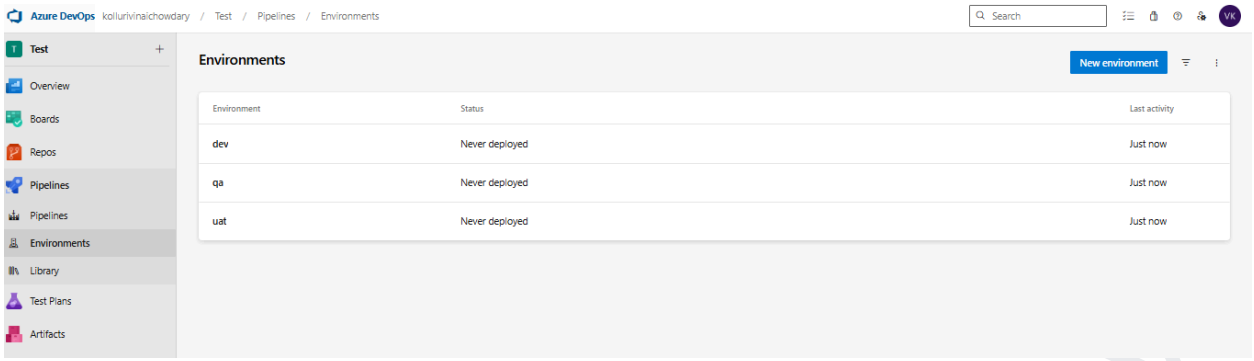
13. Environments in Azure Pipelines

What is an Environment?

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

Features

- Deployment history
- Approvals
- Resource tracking



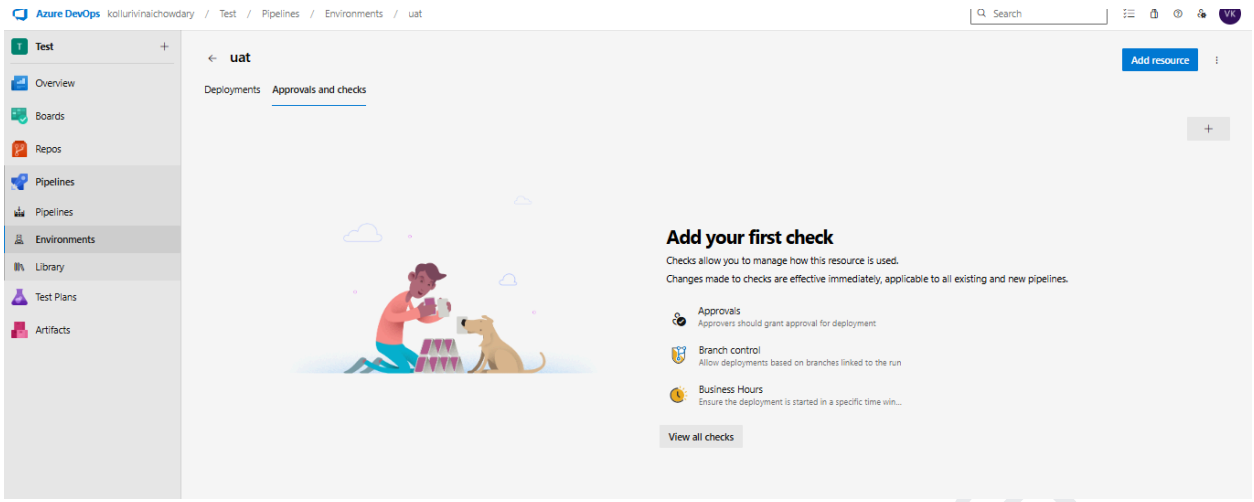
14. Approvals & Gates

Manual Approvals

- Required before deployment

Automated Gates

- Health checks
- Monitoring validation



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

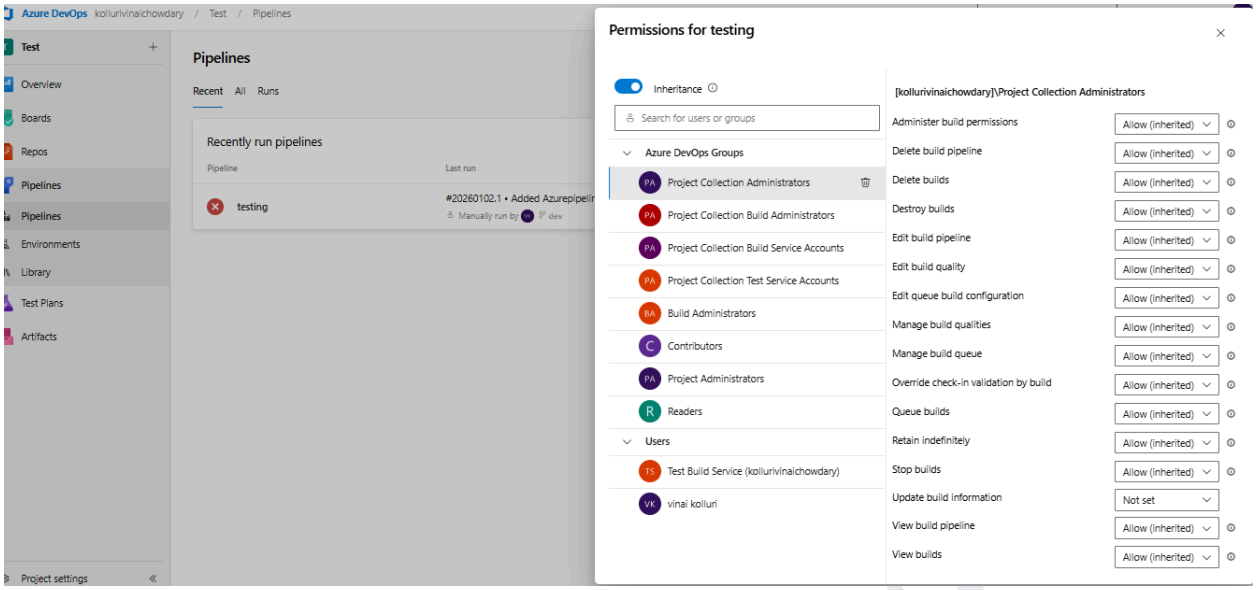
17. Permissions & Security

Pipeline Security

- Who can edit pipeline
- Who can run pipeline

Environment Security

- Deployment approvals
- Role-based access



18. Monitoring & Troubleshooting

Features

- Pipeline logs
- Task-level logs
- Timeline view

Common Failures

- Agent issues

- 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

Developer Commit

↓

CI Build

↓

Unit Tests

↓

Artifact Publish

↓

Deploy Dev (Auto)

↓

Deploy QA (Approval)

↓

Deploy Prod (Manager Approval)