

Coding Challenge

CI/CD Pipeline Implementation in Azure DevOps

This project demonstrates how to implement a CI/CD pipeline in Azure DevOps, integrating with Azure Data Factory (ADF) and Azure Databricks. The goal is to automate code integration, testing, and deployment to ensure consistency, reliability, and faster delivery.

CI/CD (Continuous Integration / Continuous Deployment) is a DevOps practice that automates code integration, testing, and deployment.

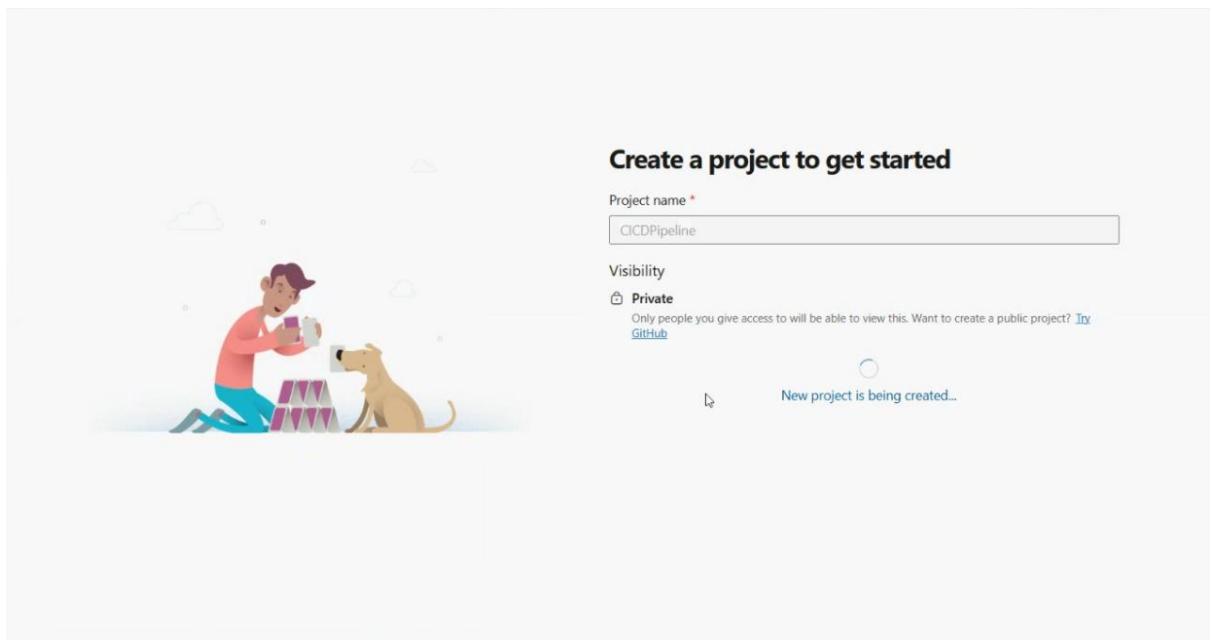
In this project, we:

- Created an Azure DevOps pipeline for Python scripts.
- Integrated with Azure Data Factory for orchestration.
- Used Azure Databricks for running transformations and advanced analytics.
- Configured a self-hosted agent to run the pipeline.

Project Setup

◆ Azure DevOps Setup

1. Created an Azure DevOps Organization and a new Project.



2. Created a Repository and initialized it with:

- o README.md (project overview)

The screenshot shows the Azure DevOps repository interface for a project named "CICDPipeline". The left sidebar has "Files" selected. In the main area, a file named "README.md" is shown with a preview of its content:

```

main / README.md
Type to find a file or folder...

Files
Contents History

Name Last change Commits
M README.md Just now 3516372f Added README.md Sivagami G

Introduction
TODO: Give a short introduction of your project. Let this section explain the objectives or the motivation behind this project.

Getting Started
TODO: Guide users through getting your code up and running on their own system. In this section you can talk about:
1. Installation process
2. Software dependencies
3. Latest releases
4. API references

Build and Test
TODO: Describe and show how to build your code and run the tests.

Contribute
TODO: Explain how other users and developers can contribute to make your code better.

```

- o A sample hello.py Python script.

The screenshot shows the Azure DevOps repository interface for the same project. A new file named "hello.py" is being created. A modal dialog is open for "New file", with the name "hello.py" entered in the input field. The background shows the repository structure with "README.md" and "hello.py" listed.

New file

New file name: / hello.py

Cancel Create

Explain the objectives or the motivation behind this project.

1. Installation process
2. Software dependencies
3. Latest releases
4. API references

Build and Test

TODO: Describe and show how to build your code and run the tests.

Contribute

TODO: Explain how other users and developers can contribute to make your code better.

hello.py

```

1 print("Hello CI/CD from Azure DevOps Pipeline!")
2

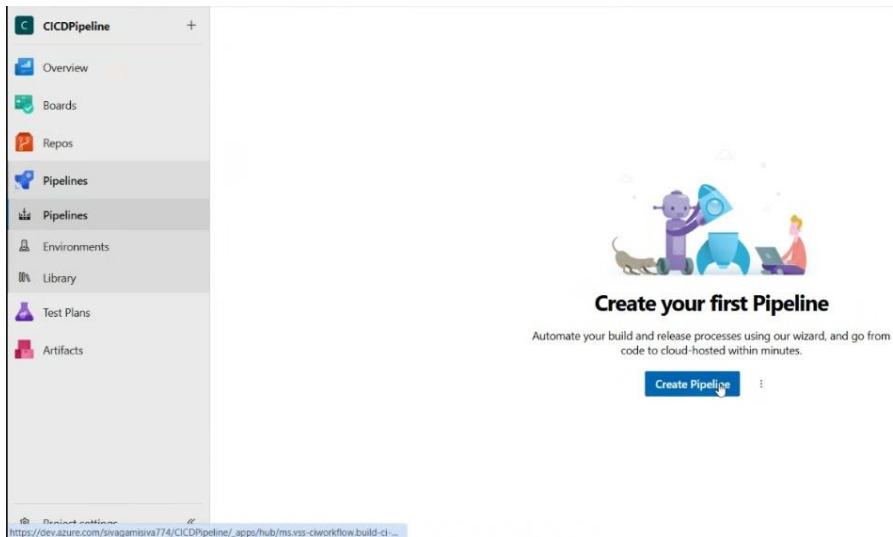
```

Commit Revert

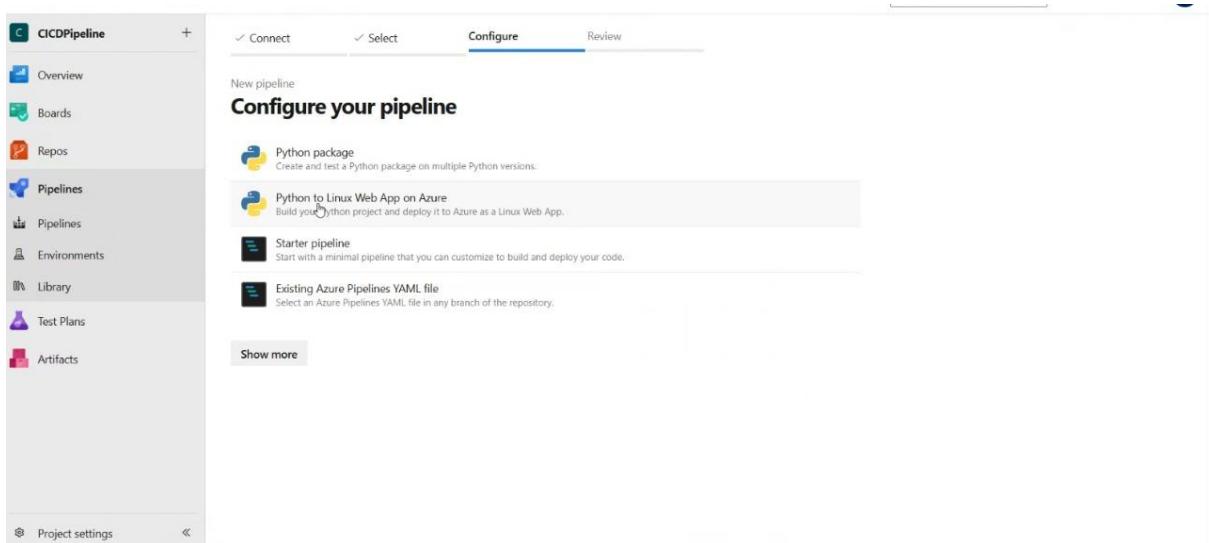
Pipeline Setup

◆ Creating the Pipeline in Azure DevOps

1. Navigated to Pipelines → New Pipeline.



2. Selected Azure Repos Git as the source.
3. Chose Starter pipeline and modified the YAML file.



◆ YAML Configuration for Python Script Execution

azure-pipelines.yml

The screenshot shows the Azure DevOps Pipelines interface. On the left, there's a sidebar with project navigation: Overview, Boards, Repos, Pipelines (selected), Pipelines, Environments, Library, Test Plans, and Artifacts. The main area is titled "Review your pipeline YAML". It shows a code editor with the file "CICDPipeline / azure-pipelines.yml". The code is a YAML configuration for a pipeline:

```

1 # Starter pipeline
2 # Start with a minimal pipeline that you can customize to build and deploy your code.
3 # Add steps that build, run tests, deploy, and more:
4 # https://aka.ms/yaml
5
6 trigger:
7 - main
8
9 pool:
10  - vmImage: ubuntu-latest
11
12 steps:
13 - script: echo Hello, world!
14   displayName: 'Run a one-line script'
15
16 - script: |
17   echo Add other tasks to build, test, and deploy your project.
18   echo See https://aka.ms/yaml
19   displayName: 'Run a multi-line script'
20

```

At the top right, there are "Variables" and "Save and run" buttons. Below the code editor, there's a "Project settings" link.

Self-Hosted Agent

◆ Setting Up the Agent

1. Created an Agent Pool in Azure DevOps.

The screenshot shows the "Project Settings" page for the "CICDPipeline" project. In the left sidebar under "Pipelines", "Agent pools" is selected. The main area shows the "Agent pools" list with two entries: "Azure Pipelines" and "Default". To the right, a modal window titled "Add agent pool" is open. The modal has the following fields:

- Name:** Self Hosting
- Pool type:** Self-hosted
- Description (optional):** (empty)
- Pipeline permissions:** Grant access permission to all pipelines

At the bottom right of the modal is a "Create" button.

The screenshot shows the Azure DevOps Pipelines interface for a project named 'CICDPipeline'. A specific pipeline run (#20250829.4) is selected. The summary card displays details such as the repository ('CICDPipeline'), branch ('main'), commit hash ('bf4fa868'), and the fact that it was 'Manually run by Sivagami G'. It also shows the duration ('Just now'), related work items (0), artifacts (0), and tests/coverage information. Below the summary is a table of jobs, with one job listed as 'Queued'.

The screenshot shows the 'User settings' page for the user 'Sivagami G'. Under the 'Personal Access Tokens' section, a modal dialog titled 'Create a new personal access token' is open. The form fields include 'Name' (set to 'Pipeline'), 'Organization' (set to 'sivagamisiva774'), 'Expiration (UTC)' (set to '30 days' or '28/09/2025'), and 'Scopes' (set to 'Full access'). The 'Custom defined' option is also visible. At the bottom of the dialog are 'Create' and 'Cancel' buttons.

The screenshot shows a Windows Command Prompt window with the path 'C:\Windows\system32\cmd.exe'. The output of the command 'agent v4.260.0' is displayed, showing the version and a commit hash. Subsequent commands show the connection to the Azure DevOps server, registration of the agent, and configuration of the agent pool and work folder.

```

C:\Windows\system32\cmd.exe: ~ + ~
agent v4.260.0
               (commit 531b769)

>> Connect:
Enter server URL > https://dev.azure.com/sivagamisiva774
Enter authentication type (press enter for PAT) >
Enter personal access token > *****
Connecting to server ...

>> Register Agent:
Enter agent pool (press enter for default) > Self Hosting
Enter agent name (press enter for SIVAKAMI) >
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) > |

```

- Downloaded and configured the Self-Hosted Agent on a VM/Local machine.

- Registered with the DevOps project.
- Verified agent is online and active



◆ Linking Pipeline to Self-Hosted Agent

- In azure-pipelines.yml, we referenced the custom pool name.
- Ensured the agent had required Python runtime and permissions.

Pipeline Execution

1. Committed code changes to the main branch.
2. Pipeline triggered automatically (CI).
3. Self-Hosted Agent picked up the job.
4. Python script executed successfully.

A screenshot of the Azure DevOps Pipelines page for the 'CICDPipeline'. The left sidebar shows navigation links for Overview, Boards, Repos, Pipelines (selected), Pipelines, Environments, Library, Test Plans, and Artifacts. The main area displays a table of recent pipeline runs under the 'Runs' tab. The table has columns for 'Description', 'Stages', and 'Last Run'. The first two runs are successful (green checkmark), while the last five are failed (red X). Each row includes a 'Run pipeline' button and a 'More options' menu icon.

Description	Stages	Last Run
#20250829.6 - Update azure-pipelines.yml for Azure Pipelines	✓	Just now 22s
#20250829.5 - Update azure-pipelines.yml for Azure Pipelines	✓	4m ago 26s
#20250829.4 - Update azure-pipelines.yml for Azure Pipelines	✗	16m ago <1s
#20250829.3 - Update azure-pipelines.yml for Azure Pipelines	✗	16m ago <1s
#20250829.2 - Set up CI with Azure Pipelines	✗	22m ago <1s
#20250829.1 - Set up CI with Azure Pipelines	✗	25m ago <1s