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.

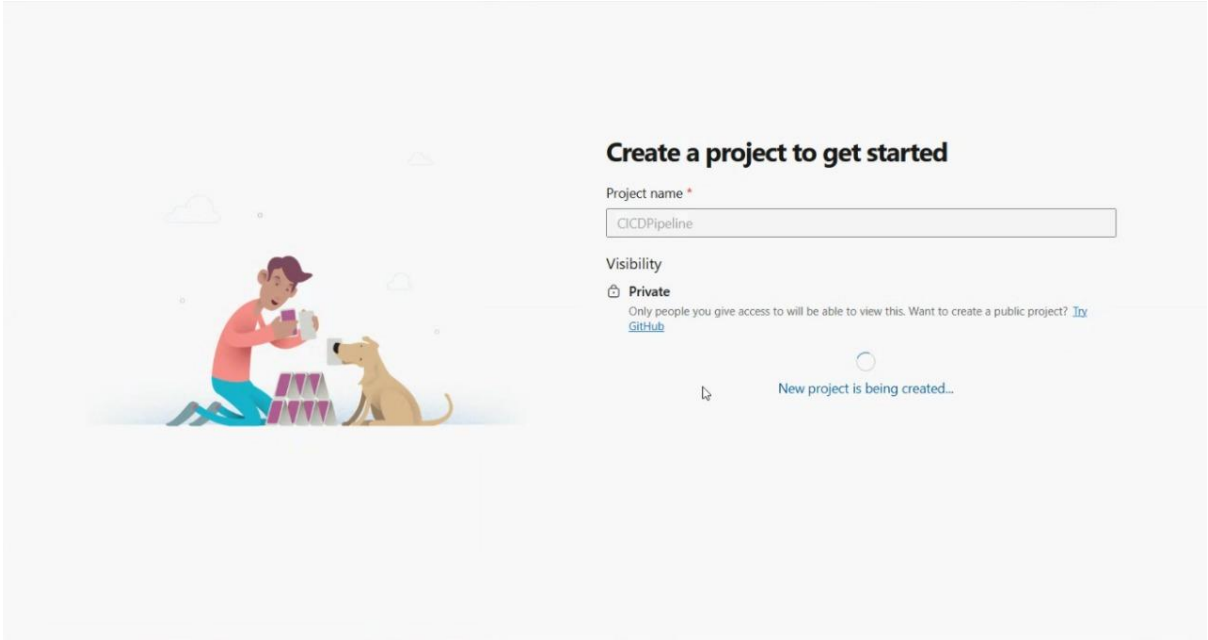
In this project, we:

- 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

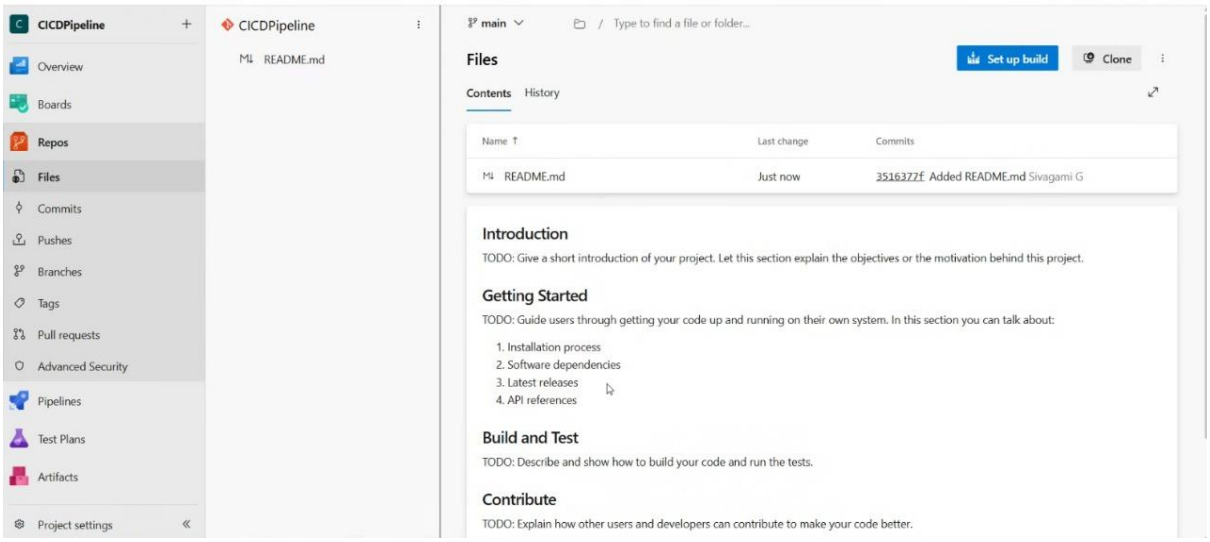
- **Azure DevOps Setup**

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

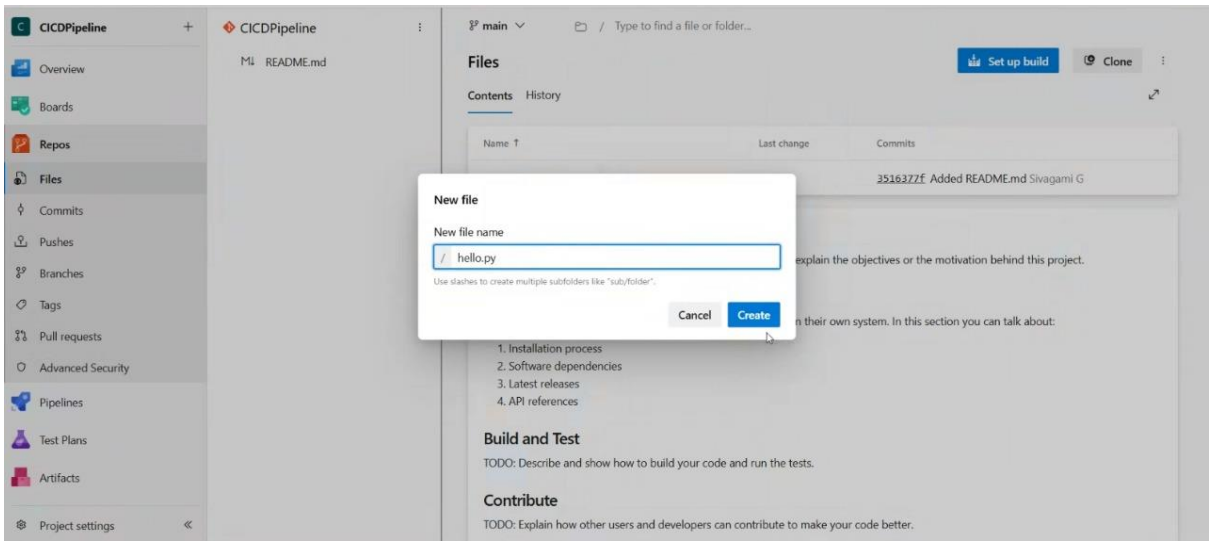


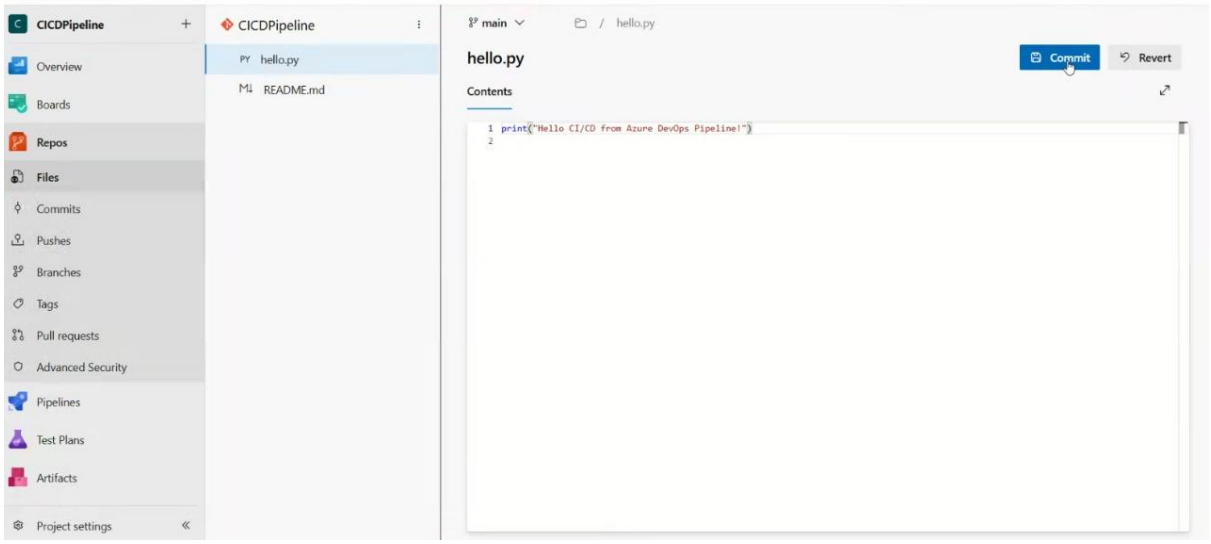
2. Created a Repository and initialized it with:

- README.md (project overview)



- A sample hello.py Python script.

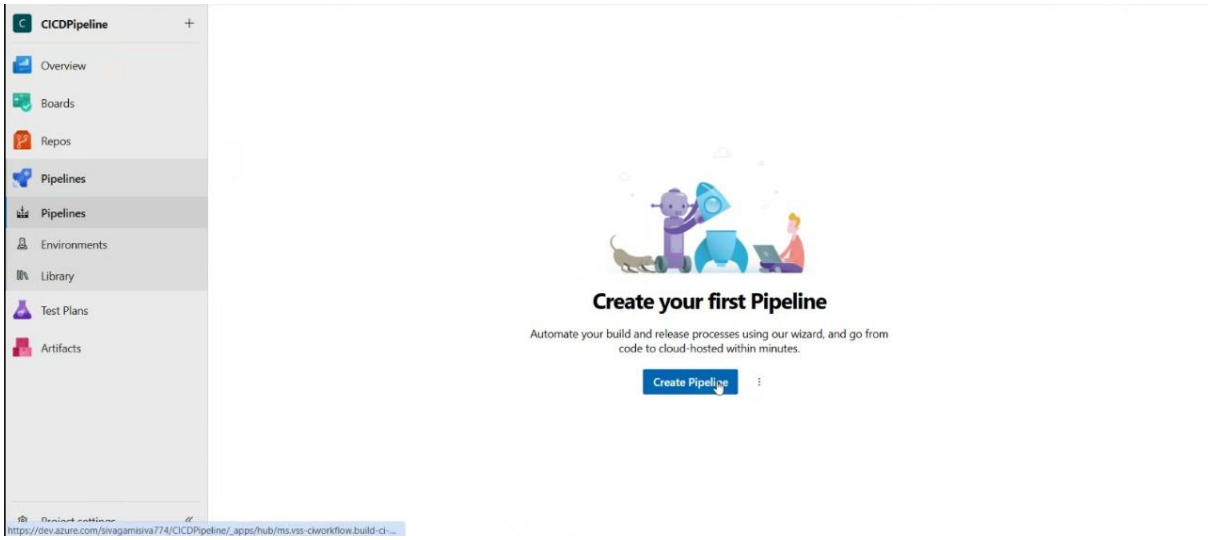




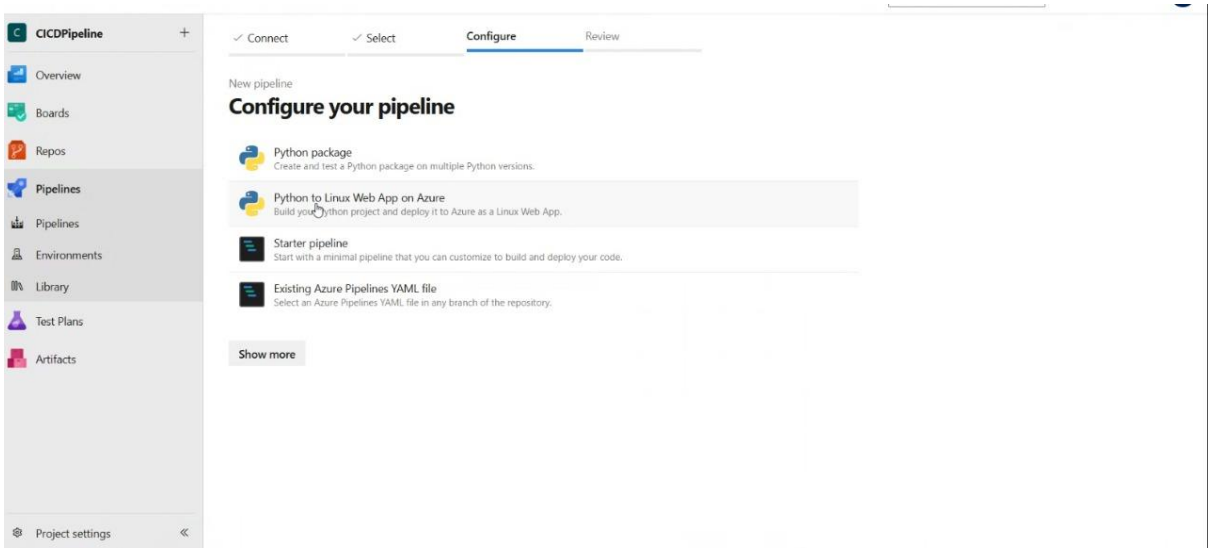
Pipeline Setup

- **Creating the Pipeline in Azure DevOps**

1. Navigated to Pipelines → New Pipeline.

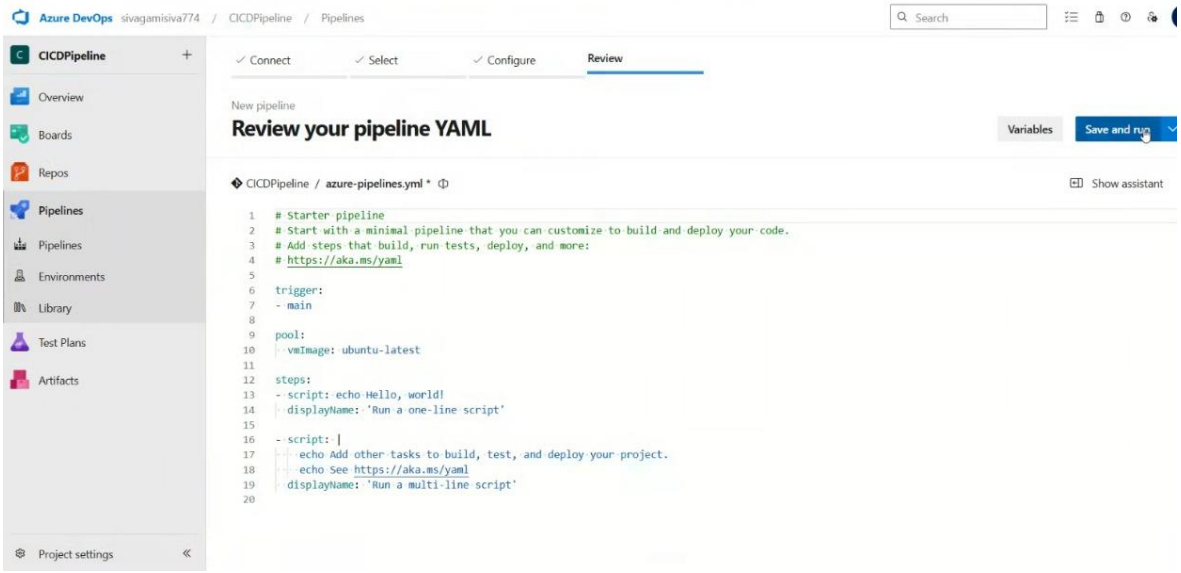


2. Selected Azure Repos Git as the source.
3. Chose Starter pipeline and modified the YAML file.



- **YAML Configuration for Python Script Execution**

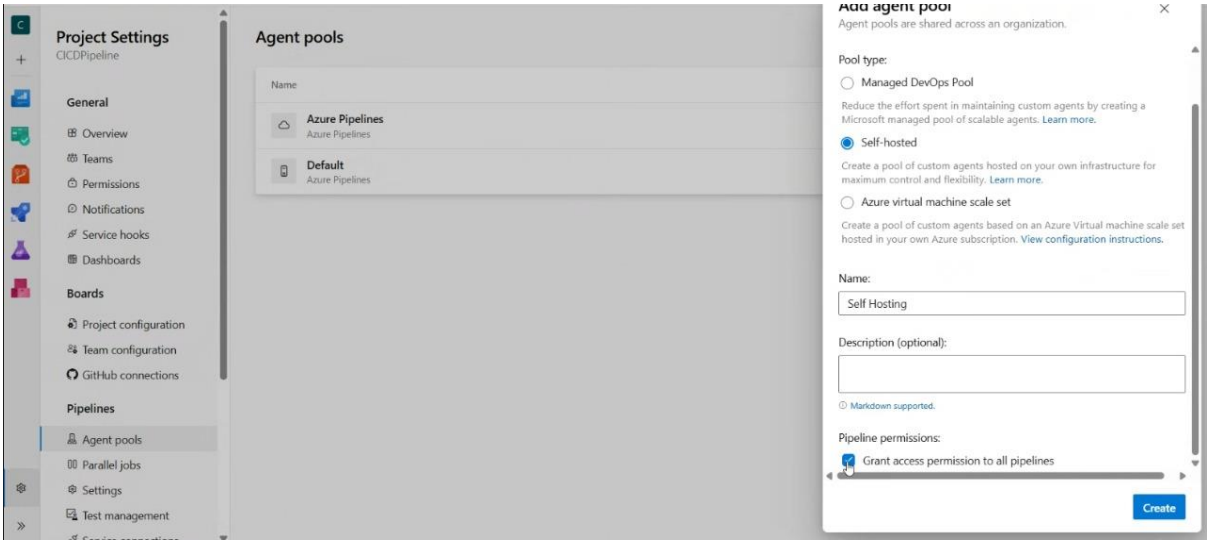
azure-pipelines.yml

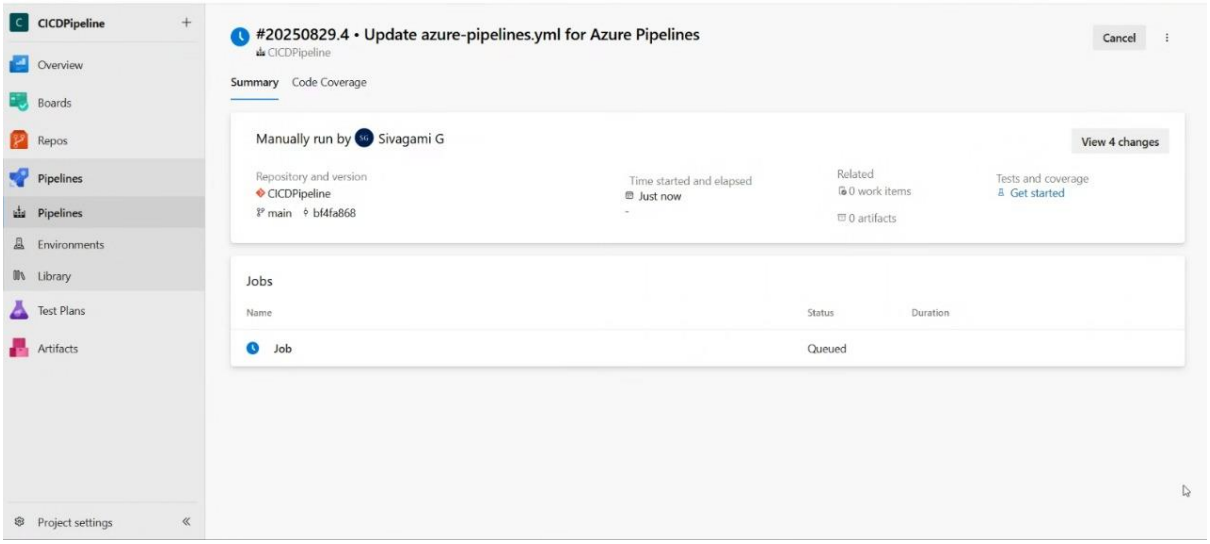


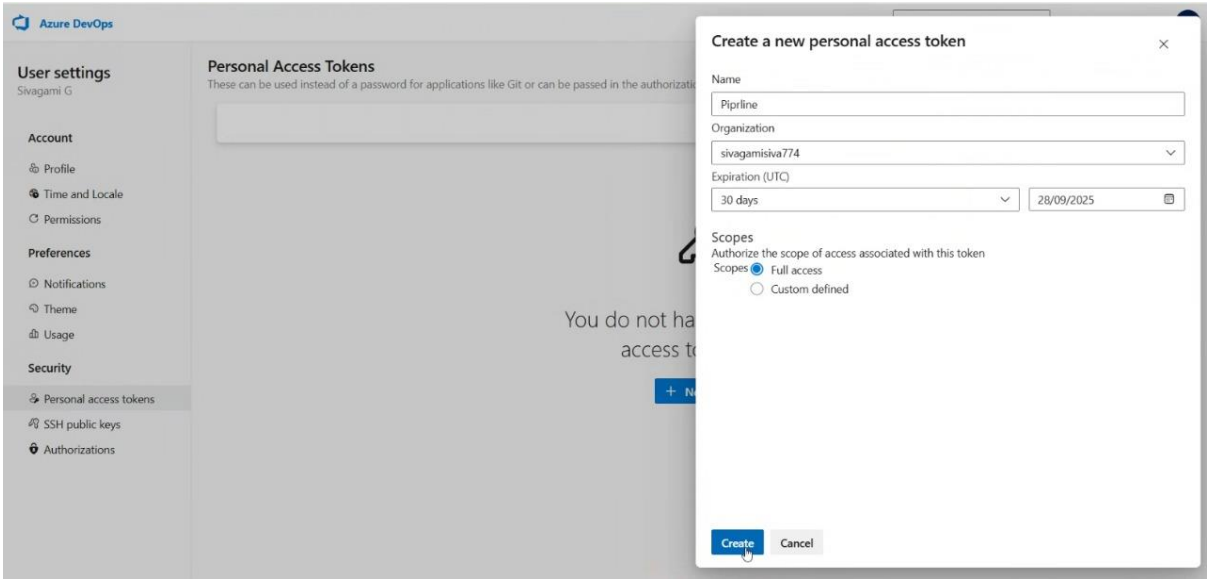
Self-Hosted Agent

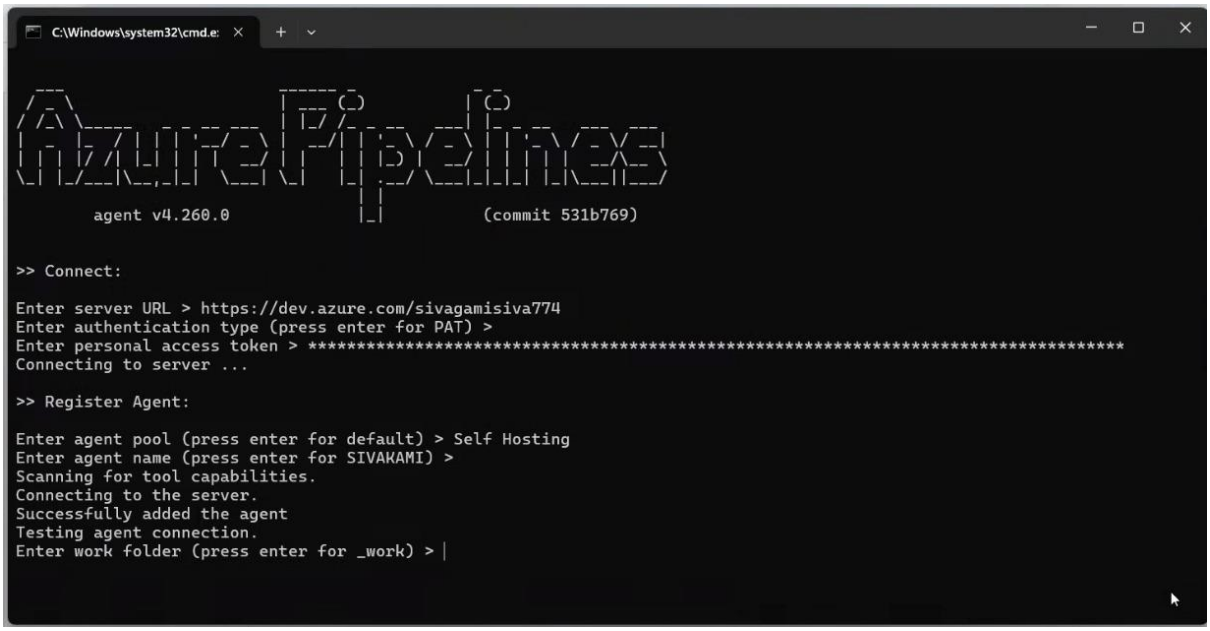
- **Setting Up the Agent**

1. Created an Agent Pool in Azure DevOps.









2. 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.