**Implementing a CI/CD Pipeline in Azure DevOps**

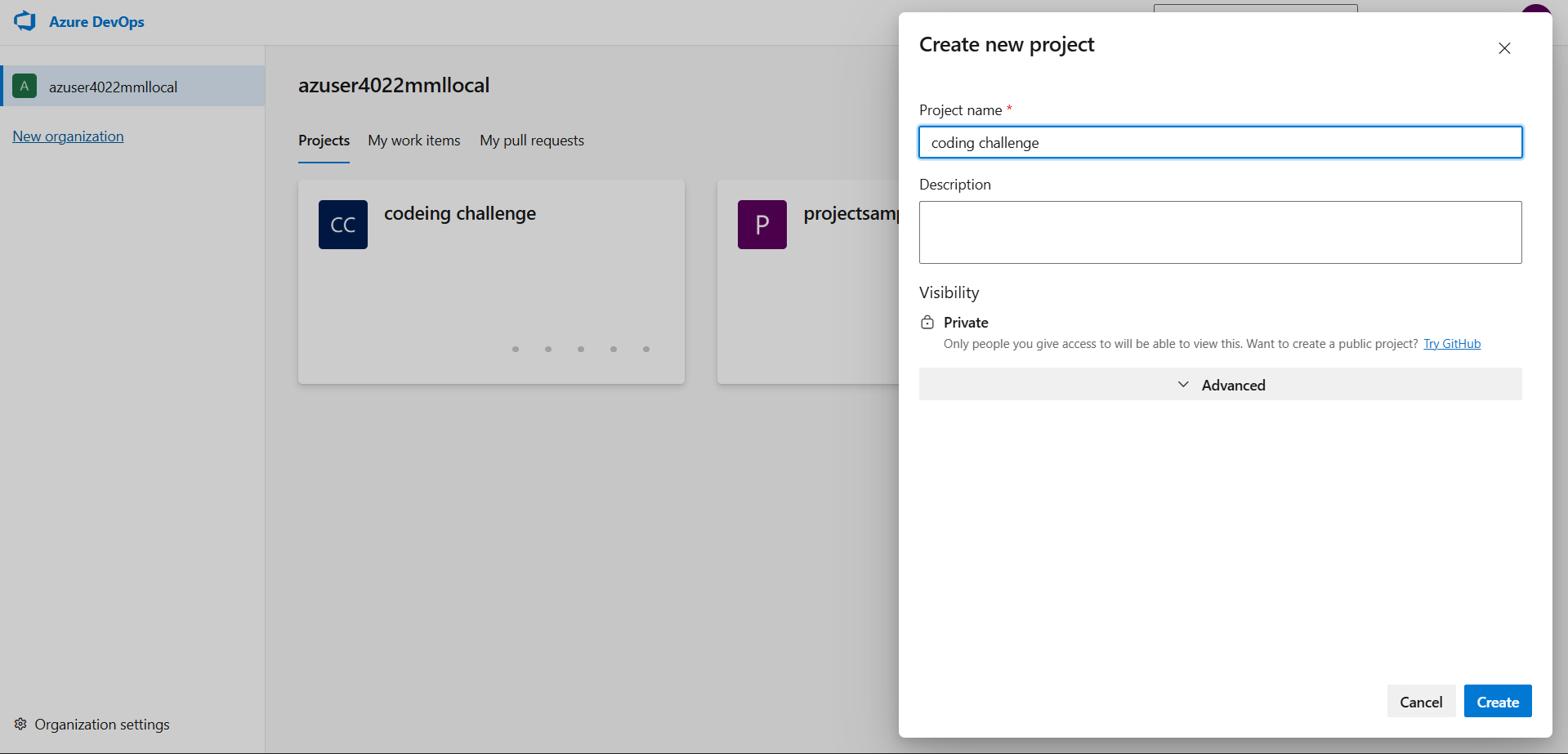
**Name: Priyeshwar**

**Mail:** [**priyesh2664@gmail.com**](mailto:priyesh2664@gmail.com)

A CI/CD pipeline in Azure DevOps automates the process of building, testing, and deploying applications. It enables teams to deliver software faster, with improved quality and reliability.

**Step 1: Create a Project in Azure DevOps**

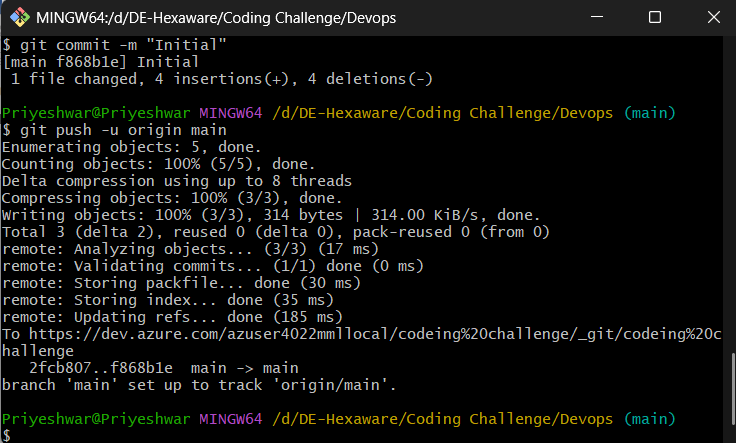
1. Go to [https://dev.azure.com](https://dev.azure.com/).
2. Click New Project and provide a name
3. Choose visibility (private or public).



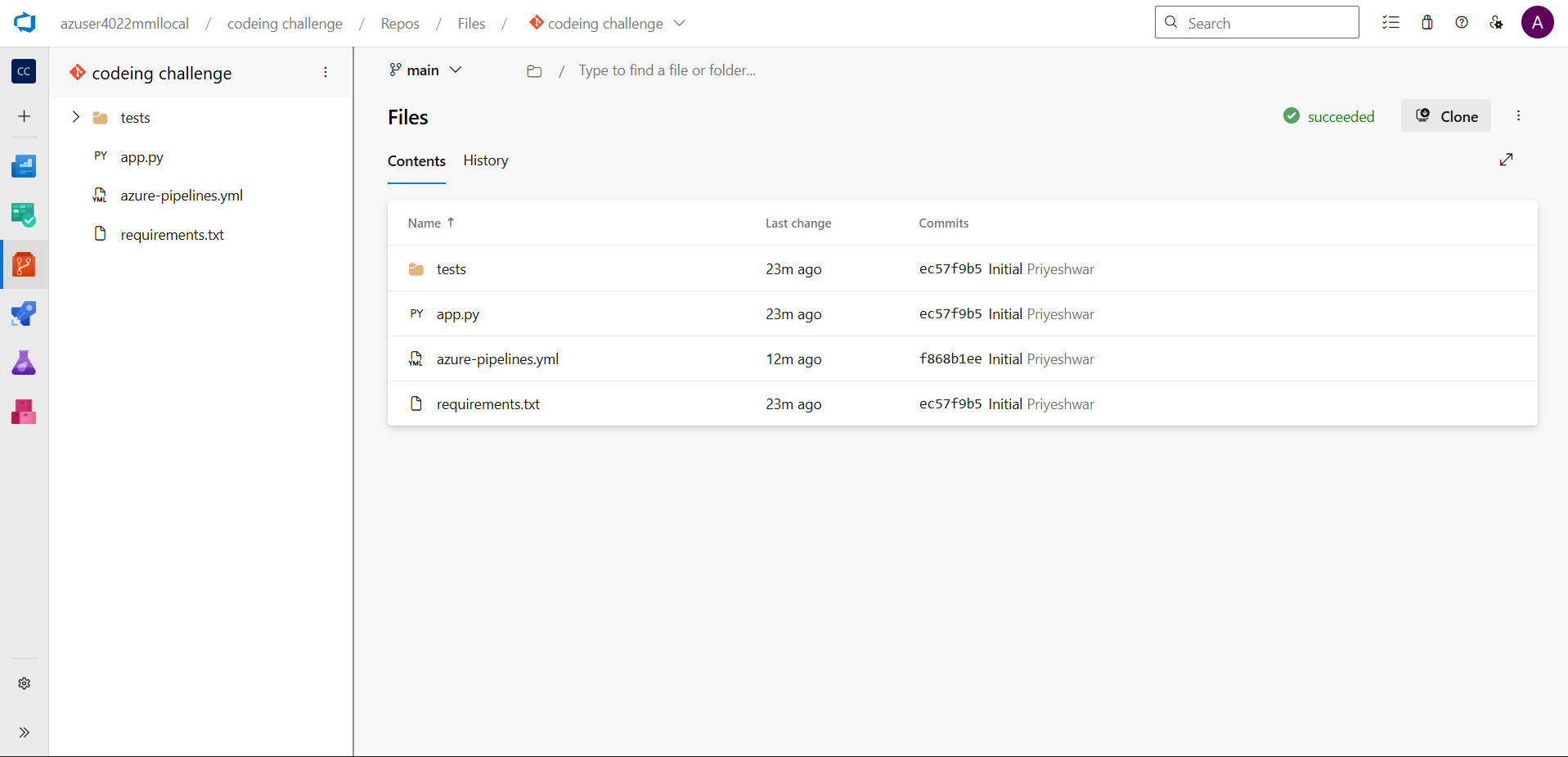
1. Select Git for version control.

**Step 2: Connect Source Code Repository**

* Push your application code to Azure Repos Git, or connect to an external repository such as GitHub or Bitbucket.



* Ensure your main (or master) branch is set as the default branch for builds.



**Step 3: Configure Continuous Integration (CI) with Azure Pipelines**

1. In your project, go to Pipelines → Create Pipeline.
2. Select your source (Azure Repos, GitHub, Bitbucket, etc.).
3. Choose Starter Pipeline or point to an existing azure-pipelines.yml file in your repository.

Pipeline.YAML

trigger:

  branches:

    include:

      - main

pool:

  name: 'local-agent'   # use your self-hosted agent pool

steps:

- script: python --version

  displayName: 'Check Python version'

- script: pip install -r requirements.txt

  displayName: 'Install dependencies'

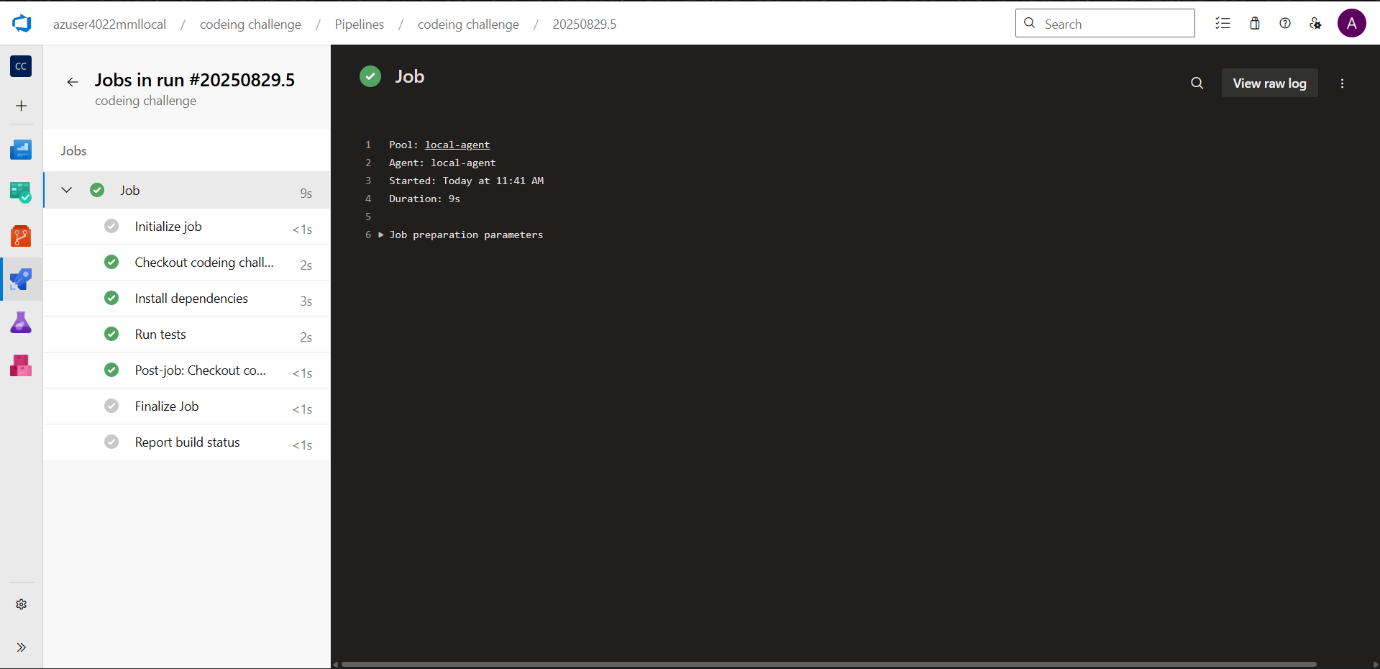
- script: pytest

  displayName: 'Run tests'

This pipeline triggers on every commit, automatically builds the application, and executes unit tests.

**Step 4: Add Continuous Testing**

* Use Azure Test Plans for manual or exploratory testing.
* Add automated test runs within your CI pipeline (e.g., dotnet test, pytest, or npm test).
* Configure approvals and release gates before production deployment



**Step 5: Monitor and Improve**

* Enable Azure Application Insights to monitor application performance and detect issues.
* Add Release Gates to block production deployments if monitoring detects failures.
* Use Azure DevOps Dashboards to track build and deployment success, as well as overall project health.

