BCSL657D Program 10

Creating Build Pipelines: Building a Maven/Gradle Project with Azure Pipelines, Integrating Code Repositories (e.g., GitHub, Azure Repos), Running Unit Tests and Generating Reports

This program focuses on implementing a Continuous Integration and Continuous Deployment (CI/CD) pipeline for Java-based projects using **Azure Pipelines**. It involves configuring build automation for **Maven or Gradle projects**, integrating with popular **source code repositories** such as GitHub or Azure Repos, executing **automated unit tests**, and generating **test reports**.

The goal is to demonstrate how modern DevOps practices can be applied to streamline the software delivery process—ensuring **code quality**, **build reliability**, and **fast feedback cycles**. This setup is a fundamental component of modern **DevOps workflows**, enabling teams to **build, test, and deliver** applications rapidly and consistently.

Key Focus Areas:

1. **Build Automation** using Azure Pipelines for Java projects.
2. **Source Code Integration** using GitHub or Azure Repos.
3. **Unit Testing** for quality assurance.
4. **Report Generation** for test outcomes and coverage insights.

Technologies and Tools Involved:

- **Azure DevOps** – For building and deploying CI/CD pipelines.
- **Maven/Gradle** – Build automation tools for Java projects.
- **GitHub / Azure Repos** – Source code management platforms.

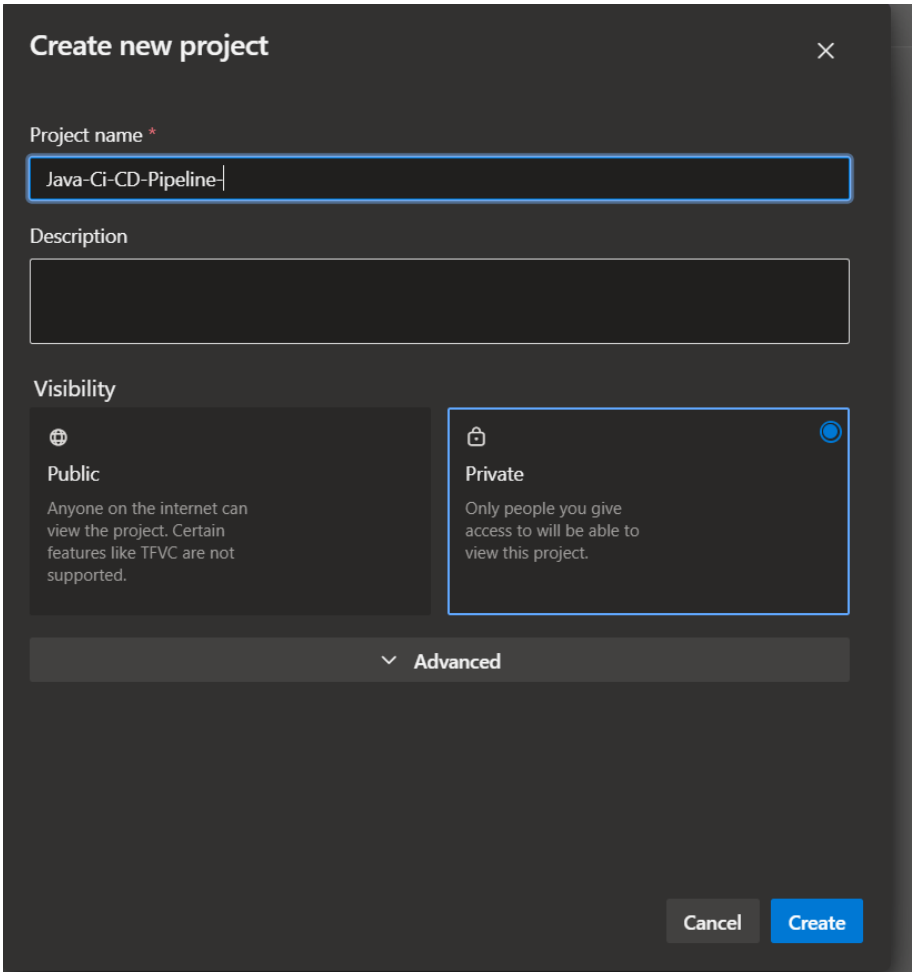
Why This is Important:

- Automates the entire software build and test process.
- Ensures code from version control is always in a deployable state.
- Detects issues early through testing in pipeline.
- Improves code quality through continuous feedback.
- Facilitates collaboration and faster iteration in agile teams.

Step-by-Step Instructions

1. **Navigate to Azure DevOps Portal**:
   Open a browser and go to:
   https://dev.azure.com/{your-organization-name}
   (Replace {your-organization-name} with your actual Azure DevOps organization ID.)
2. **Sign In**:
   Use your Microsoft account credentials to log in. If you don't have one, create it here.

3. **Click on "New Project"**:
    - Located typically on the top right or in the project dashboard.
    - This opens a configuration form to set up your new project.
4. **Configure Project Details**:
    - **Project Name**: A unique name that represents the application or service you're building (e.g., Java-CI-CD-Demo).
    - **Description** *(Optional)*: Brief explanation of the project.
    - **Visibility**:
        - **Private** (default): Accessible only to added team members.
        - **Public**: Visible to everyone (not recommended unless open source).



5. Click on “Create”

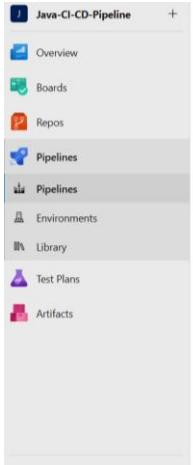
After configuring the project details such as name, visibility, and version control settings:

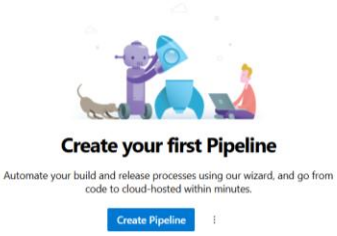
- Click the **“Create”** button at the bottom of the form.

- Azure DevOps will now:
    - Set up your project space.
    - Enable services like **Repos**, **Pipelines**, **Boards**, **Artifacts**, etc., depending on your permissions and organization policies.
- Once the project is created, you'll be redirected to the **project dashboard**.

6. Access the Pipelines Section

- On the **left sidebar navigation menu**, locate and click on **"Pipelines"**.
- This section is used to:
    - Create and manage **build pipelines** (CI).
    - Configure **release pipelines** (CD).
    - Monitor pipeline runs, view logs, and debug failures.
- The **Pipelines** menu may include:
    - **Pipelines** (actual CI pipeline definitions)
    - **Environments**
    - **Releases**
    - **Library**
    - **Task Groups**



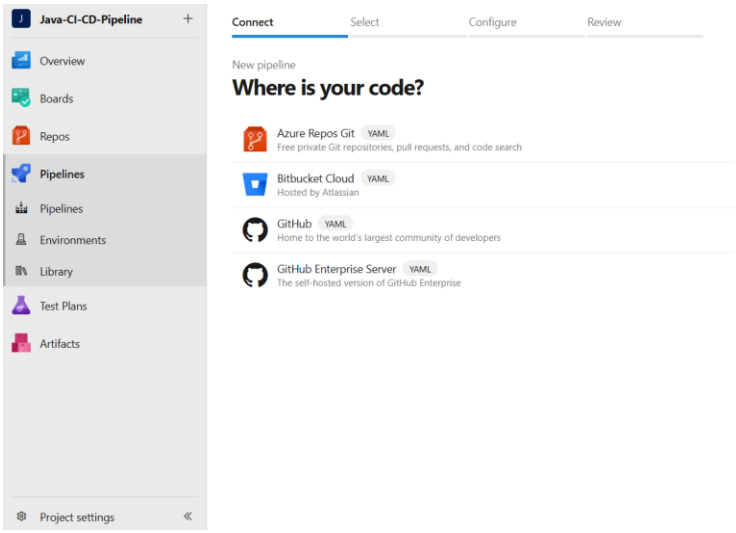


Step 7: Click on "Create Pipeline"

- Inside the **Pipelines** section, click the **"New Pipeline"** button to start creating your pipeline.

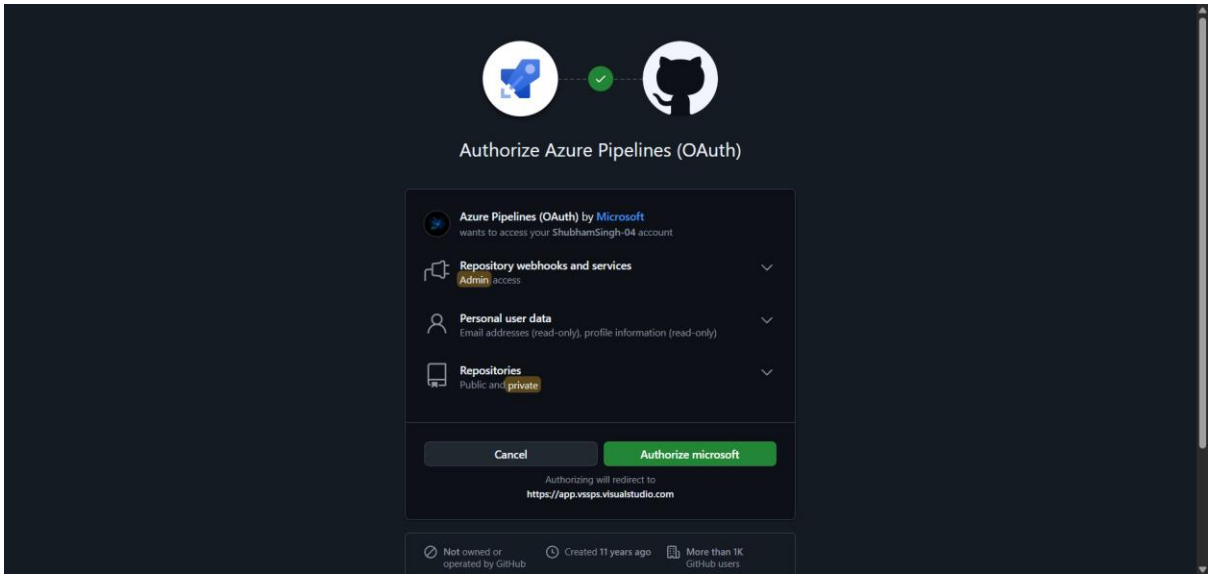
Step 8: Select "GitHub YAML"

- Choose **GitHub** as the source for the repository.
- Select **YAML** for the pipeline definition format.



Step 9: Authorize Microsoft to Access Your Git Repositories

- If prompted, authorize Azure DevOps to access your **GitHub** account by signing in and granting permissions.

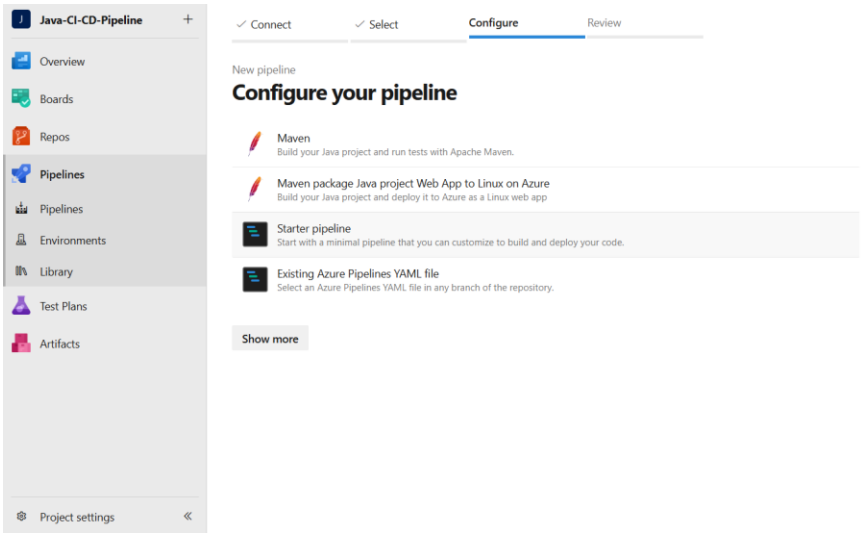


Step 10: Choose the Repository

- Select the **GitHub repository** that holds your Maven/Gradle project.
- Azure DevOps will automatically detect the repository and prompt you for YAML configuration.

Step 11: Configure Your Pipeline

- After selecting the repository, you'll be prompted to **configure your pipeline**.
- Select **Maven** as the build tool, which is used for Java project builds.

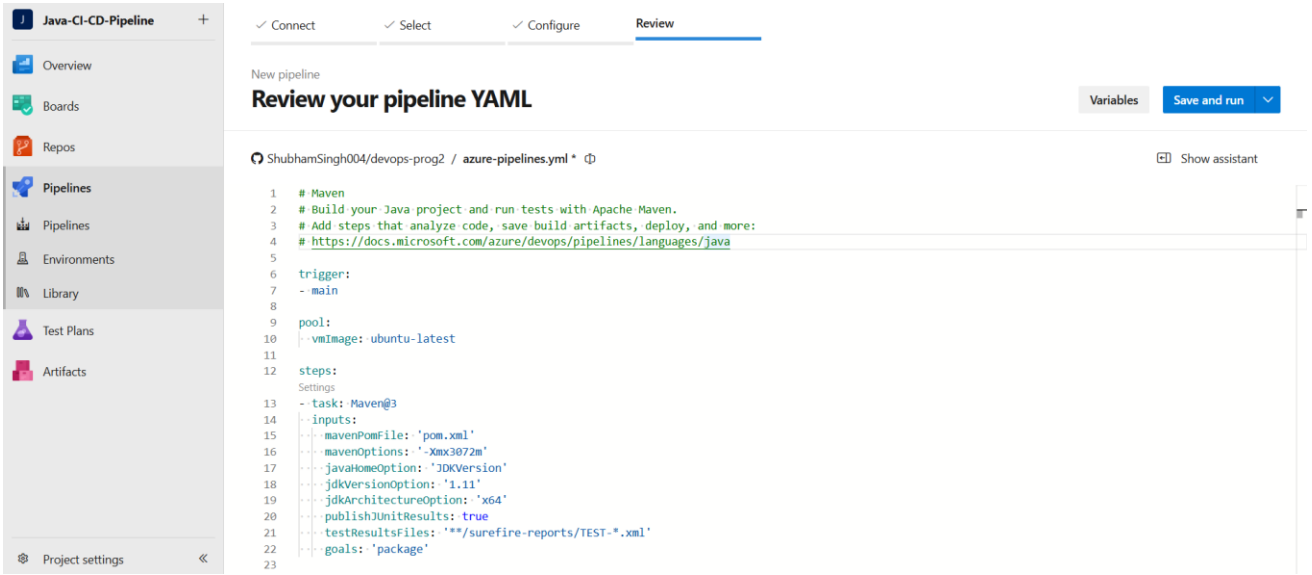


Step 12: Choose Maven for the Build

- In the **"Configure your pipeline"** section, Azure DevOps will auto-detect the Maven project.
- If not detected, manually choose **Maven** from the list of options (this will create a maven.yml file).

Step 13: Review Your Pipeline YAML

- The **YAML file** will be automatically generated for the pipeline.
- Review the YAML configuration to ensure it aligns with your build process, such as:



Change the YAML for self hosted Agent

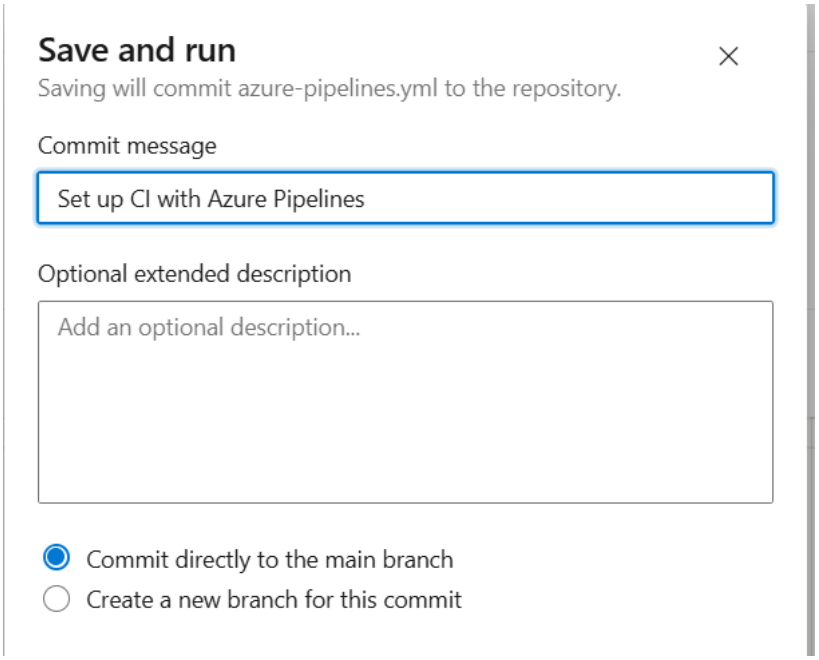
```
trigger:
- main


pool:
  name: Default  # Matches your self-hosted agent pool


steps:
- task: Maven@3
  inputs:
    mavenPomFile: 'pom.xml'
    mavenOptions: '-Xmx3072m'
    javaHomeOption: 'Path'
    jdkDirectory: 'C:\\Program Files\\Microsoft\\jdk-17.0.14.7-
hotspot'  # PATH in your system
    jdkArchitectureOption: 'x64'
    publishJUnitResults: true
    testResultsFiles: '**/surefire-reports/TEST-*.xml'
    goals: 'package'
```

Step 15: Save and Run the Pipeline

- After reviewing and confirming the pipeline configuration, click on the **Save and Run** button.
- This will trigger the pipeline execution immediately.

Step 16: Provide a Commit Message and Description

- In the pop-up dialog, you'll be prompted to provide a **commit message**.
    - **Commit Message**: Write a meaningful message.
    - **Description (Optional)**: You can add a brief description explaining the changes made.
- Ensure the **branch** is set correctly (usually **main** or **master**).



Step 18: Navigate to Agent Pools

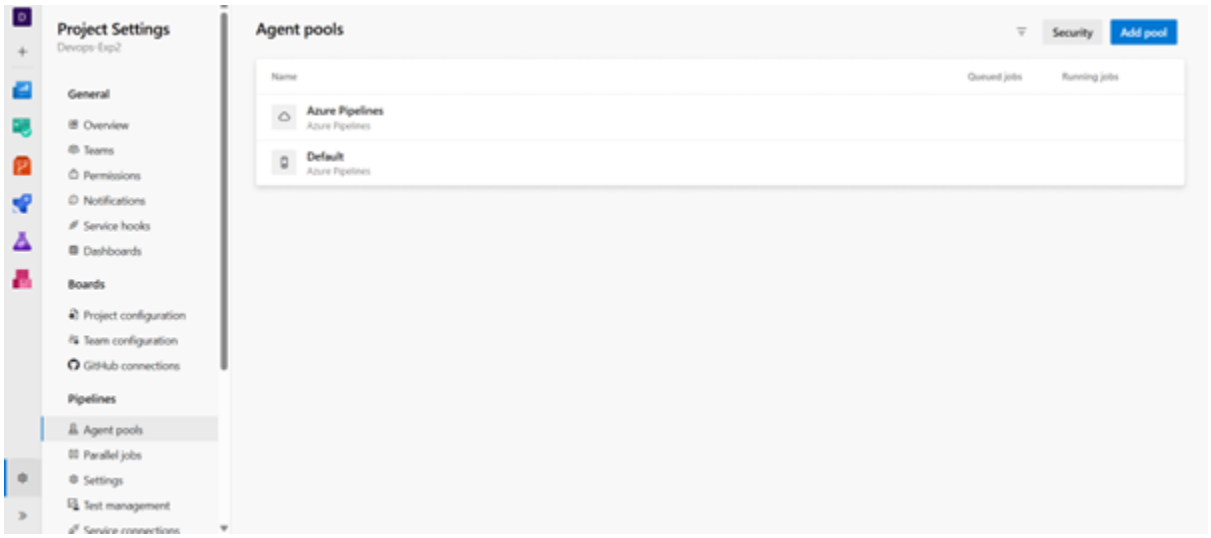
- In your Azure DevOps project dashboard, scroll down and click on **"Project Settings"** (bottom-left corner).

Step 19: Access Agent Pools

- Under the **Pipelines** section in the settings menu, click on **"Agent Pools"**.
- Agent Pools allow you to manage agents that run your pipeline jobs.

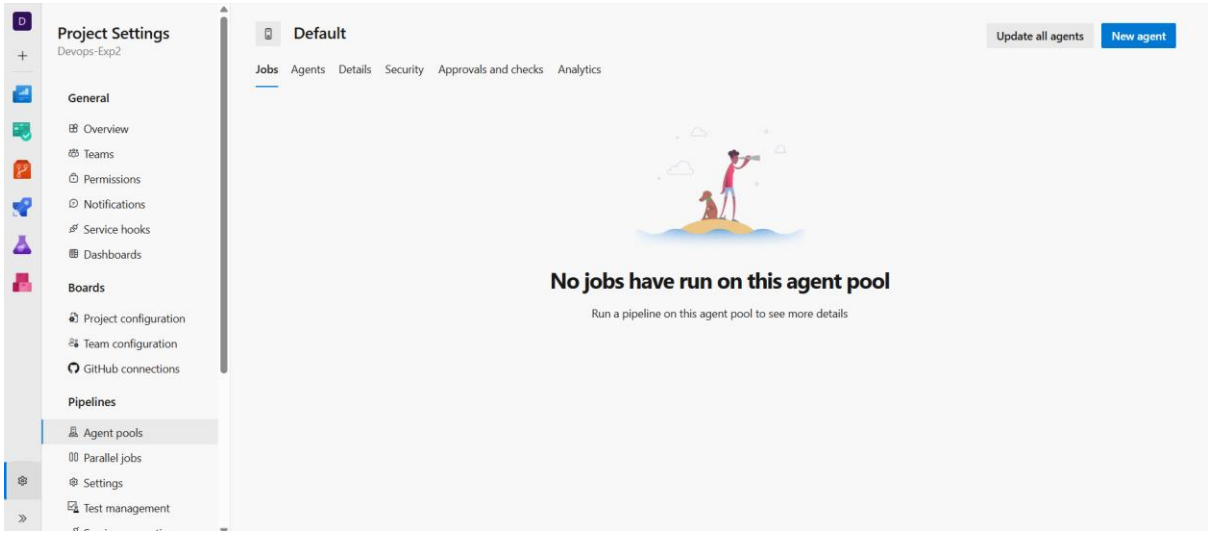
Step 20: Select the Default Azure Pipeline Pool

- Click on the **"Default"** agent pool (this is the built-in Microsoft-hosted pool).
- This pool already includes common build environments like Ubuntu, Windows, and macOS.



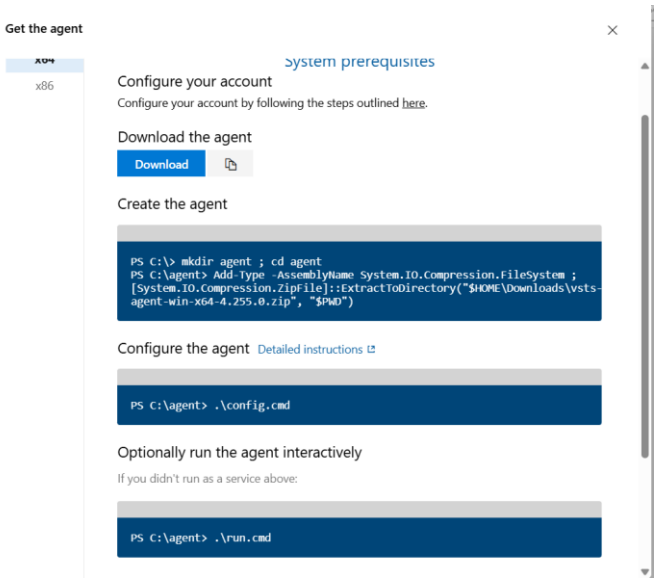
Step 21: Click on "New Agent"

- Inside the selected pool, click on the **"New agent"** button (usually top-right).
- A dialog or new page will appear showing instructions to download and configure a self-hosted agent.



Step 22: Follow On-Screen Instructions to Create a New Agent

- Choose the operating system for your self-hosted agent (Windows, macOS, or Linux).
- Download the agent package as directed.

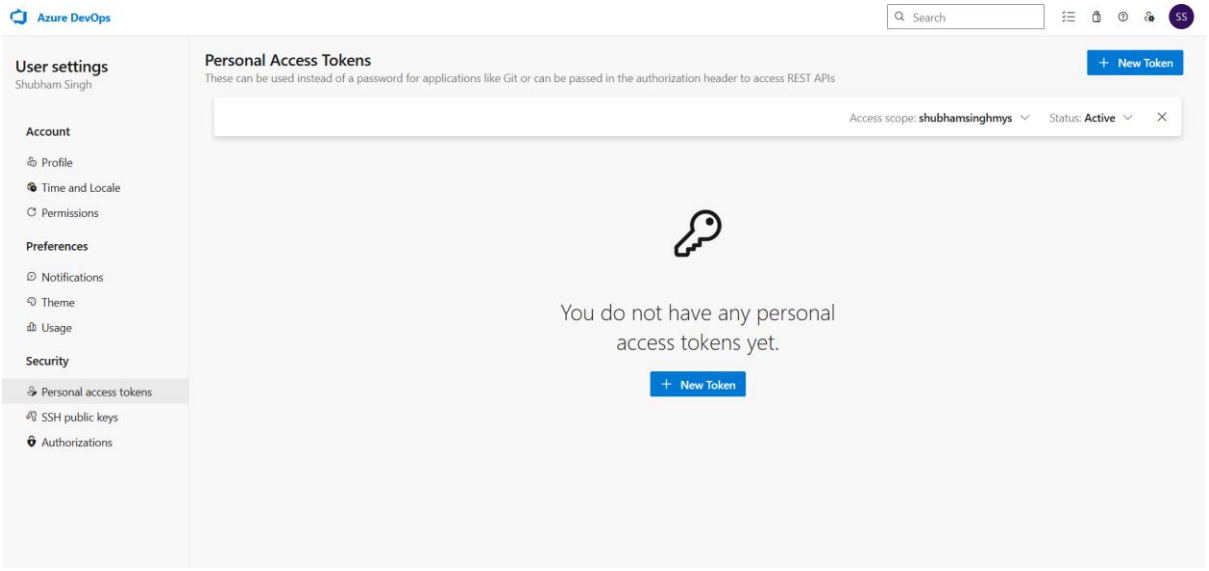


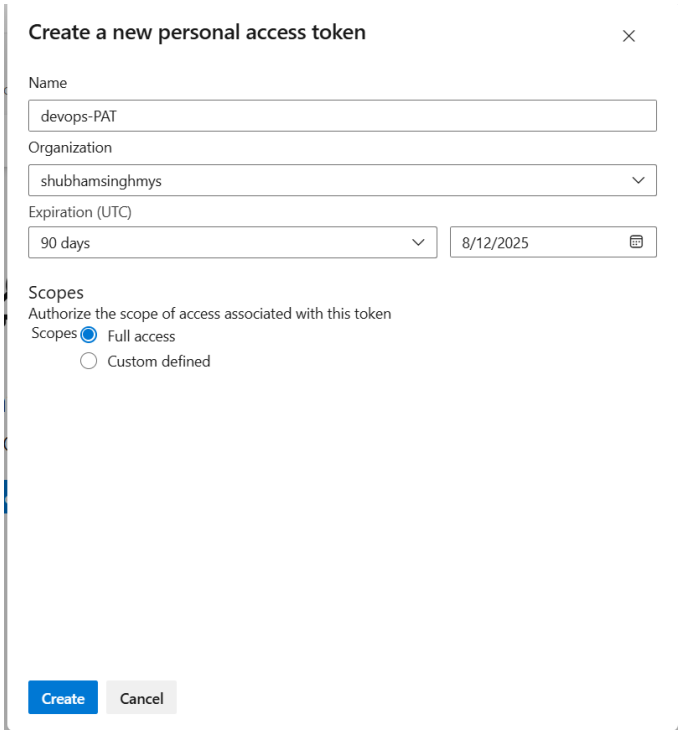
- Run the configuration commands in your terminal or command prompt:
  - Example for Windows:

.\config.cmd --url https://dev.azure.com/{organization} --auth pat

  - You'll be prompted to enter:
    - Your **organization URL**

- **Authentication token (PAT)** – generate it from Azure DevOps under User Settings → Personal Access Tokens.
    - Go to User Settings → Personal Access Tokens.
    - Click + New Token, name it, confirm org and set expiration.
    - Choose appropriate scopes (e.g., Agent Pools, Code).
    - Click Create, then copy the token immediately for use.





- Once configured, **run the agent** using the appropriate script (run.cmd or ./run.sh).

*Note: This step is only necessary if you want to use a **self-hosted agent**. For most cases, Azure's default **Microsoft-hosted agents** are sufficient.*

Step 23: Open Terminal and Navigate to Agent Directory

PS C:\WINDOWS\system32> cd "D:\6th sem\Devops\Progs\prog10\agent"

PS D:\6th sem\Devops\Progs\prog10\agent>

Step 24: Run the Agent Configuration Script

PS D:\6th sem\Devops\Progs\prog10\agent> ./config.cmd

Step 25: Connect to Azure DevOps Server

- When prompted, enter the **Azure DevOps organization URL**:

mathematica

CopyEdit

Enter server URL > https://dev.azure.com/shubhamsinghmys

- Select authentication type (press **Enter** to choose **PAT - Personal Access Token**).
- Enter your **Personal Access Token (PAT)** twice (input is masked):

Enter personal access token > **********

Enter personal access token > **********

- The agent will attempt to connect to the server.

Step 26: Register the Agent

- Press **Enter** to accept the default **agent pool** (Default):

Enter agent pool (press enter for default) >

- Press **Enter** to accept the suggested **agent name** or provide a custom name:

Enter agent name (press enter for SS-DESKTOP-8JBF) >

- If an agent with the same name exists, it will ask whether to replace it:

Pool Default already contains an agent with name SS-DESKTOP-8JBF.

Enter replace? (Y/N) (press enter for N) > Y

- The agent connection will be tested and confirmed.

Step 27: Configure Work Folder

- Press **Enter** to accept the default work folder (_work):

Enter work folder (press enter for _work) >

- The settings will be saved.

Step 28: Install and Configure Agent as a Service

- When asked to run the agent as a service, type **Y** and press Enter:

Enter run agent as service? (Y/N) (press enter for N) > Y

- Enable unrestricted service SID type, type **Y** and press Enter:

Enter enable SERVICE_SID_TYPE_UNRESTRICTED for agent service (Y/N) (press enter for N) > Y

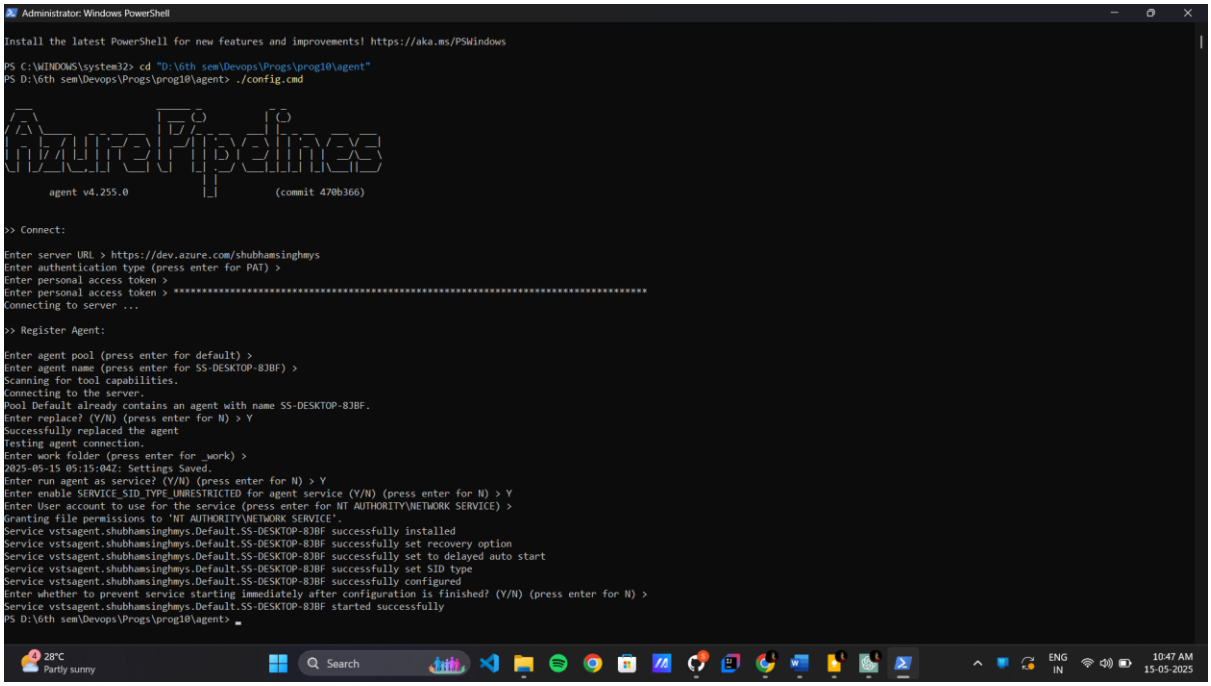
- Press Enter to accept the default user account (NT AUTHORITY\NETWORK SERVICE):

Enter User account to use for the service (press enter for NT AUTHORITY\NETWORK SERVICE) >

- The agent service will be installed, configured, and started successfully with status messages:

Service vstsagent.shubhamsinghmys.Default.SS-DESKTOP-8JBF successfully installed

Service vstsagent.shubhamsinghmys.Default.SS-DESKTOP-8JBF started successfully



The Job Will be finished when you hit run & save