**DevOps – Final Assessment**

**Section 1: Multiple-Choice Questions (MCQs)**

1. What does WSL stand for in the context of Windows?   
   **c. Windows Subsystem for Linux**
2. What is the primary goal of continuous integration (CI) in DevOps?   
   **b. Frequent integration of code changes**
3. In the Linux command line, what does the **cd** command do?   
   **b. Change the working directory**
4. Which of the following is not a Linux distribution?   
   **c. Docker**
5. What is Docker primarily used for in DevOps and containerization?   
   **c. Packaging and deploying applications in containers**
6. What is the primary purpose of Azure DevOps?   
   **b. Software development and delivery**
7. Which components are part of Azure DevOps?   
   **c. Azure Boards and Azure Pipelines**
8. How does Azure DevOps support version control in software development?   
   **b. It tracks changes in source code and manages versions.**
9. In Linux, what is the primary role of the root user?   
   **c. Administrative tasks with superuser privileges**
10. In Azure DevOps, which component is used to define, build, test, and deploy applications?   
    **c. Azure Pipelines**

**Section 2: Labs**

**Lab 1: File and Directory Management**

* Tasks:
  1. Create a directory called "lab1" in your home directory.
  2. Inside "lab1," create a text file named "sample.txt" with some content.
  3. Make a copy of "sample.txt" and name it "sample\_copy.txt."
  4. Rename "sample\_copy.txt" to "new\_sample.txt."
  5. List the files in the "lab1" directory to confirm their names.

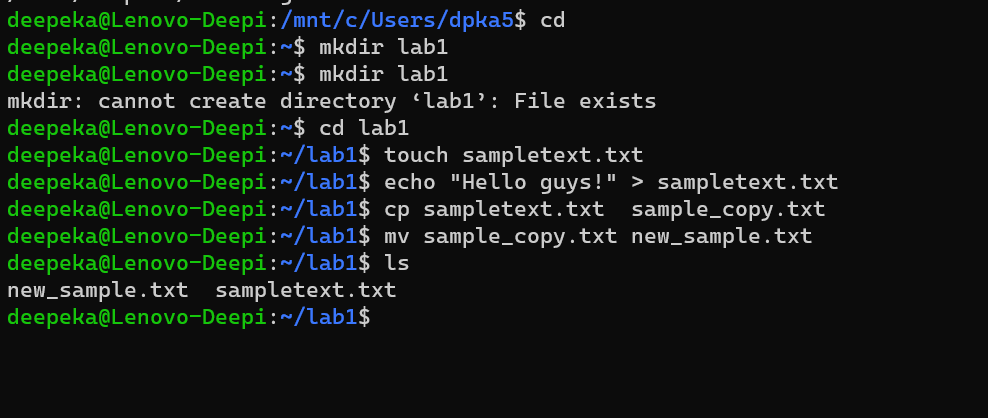
For creating new directory “mkdir” command is used.

For creating new file “touch ” command is used and “echo ” is used to add content to that file.

For creating a copy of the the file “cp” command is used.

For renaming the file “mv” command is used.

For listing the files in the current directory “ls” command is used.



**Lab 2: Permissions and Ownership**

* Tasks:
  1. Create a new file named "secret.txt" in the "lab2" directory.
  2. Set the file permissions to allow read and write access only to the owner.
  3. Change the owner of "secret.txt" to another user.
  4. Verify the new permissions and owner using the **ls -l** and **ls -n** commands.

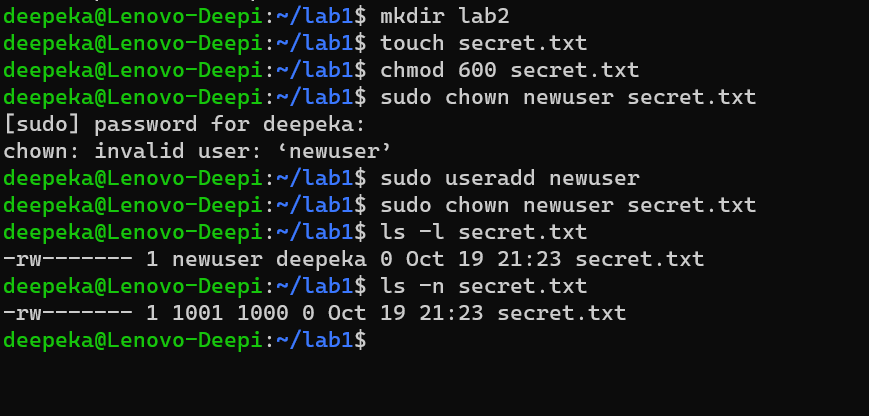
For creating a new directory “ls” command is used.

For creating a new file in that particular directory “touch” command is used.

Changing the permissions of the file includes “chmod” command and for giving permission to only the owner number 6 is used and 0 indicates that no other user has access to the file.

For altering the file permissions command should be wriiten by the super user ”sudo” command.

For changing the owner of the file “chown” command is used followed by the file name.



**Lab 3: Text Processing with Command Line Tools**

* Tasks:
  1. Create a text file with some random text in the "lab3" directory.
  2. Use the **grep** command to search for a specific word or pattern in the file.
  3. Use the **sed** command to replace a word or phrase with another in the file.
  4. Use the **wc** command to count the number of lines, words, and characters in the file.

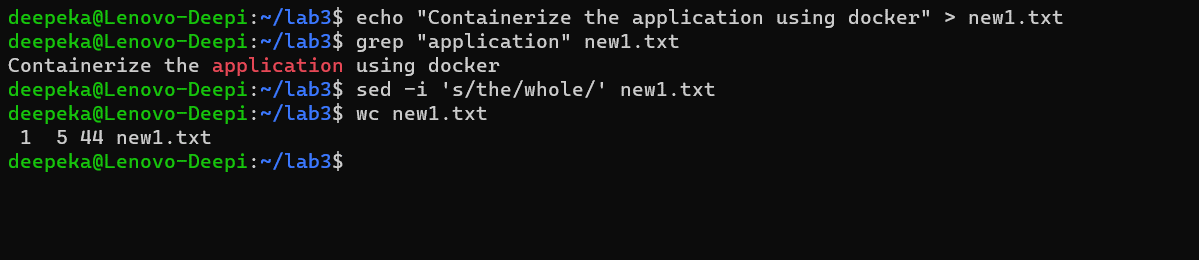
For creating a new directory “mkdir” command is used.

“grep” command is used to search a word in a particular file.

“sed” command is used to replace the word in the file with other word.

“wc” command displays the number of lines, words and characters in the file.





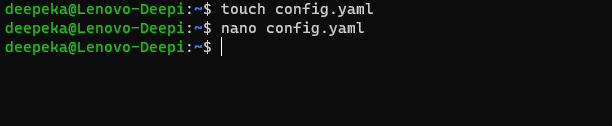
**Lab 4: Creating a Simple YAML File**

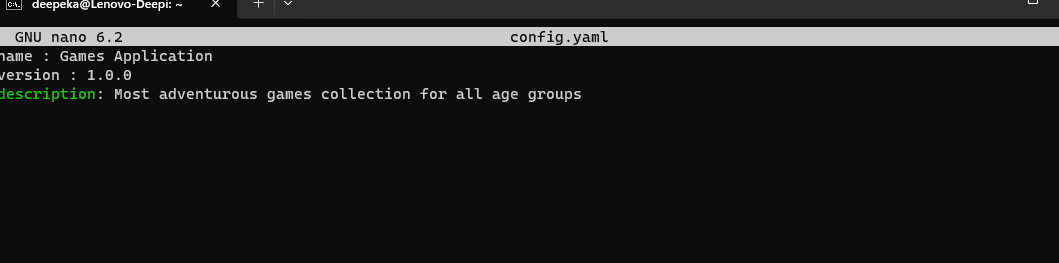
* Task:
  1. Create a YAML file named "config.yaml."
  2. Define key-value pairs in YAML for a fictitious application, including name, version, and description.
  3. Save the file.
  4. Validate that the YAML file is correctly formatted.

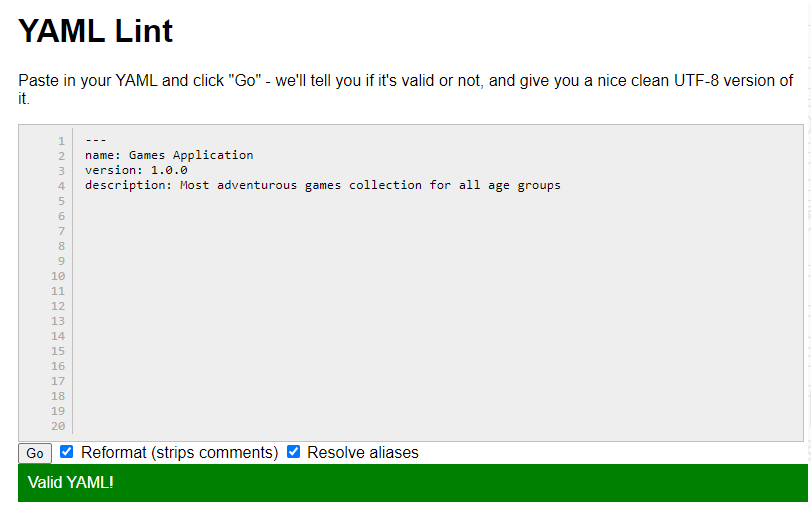
For creating a yaml file “touch” command is used.

For opening a Nano editor for Linux “nano” command is used followed by the file name.

For validating the yaml file yaml lint is used.







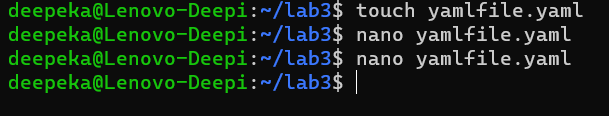
**Lab 5: Working with Lists in YAML**

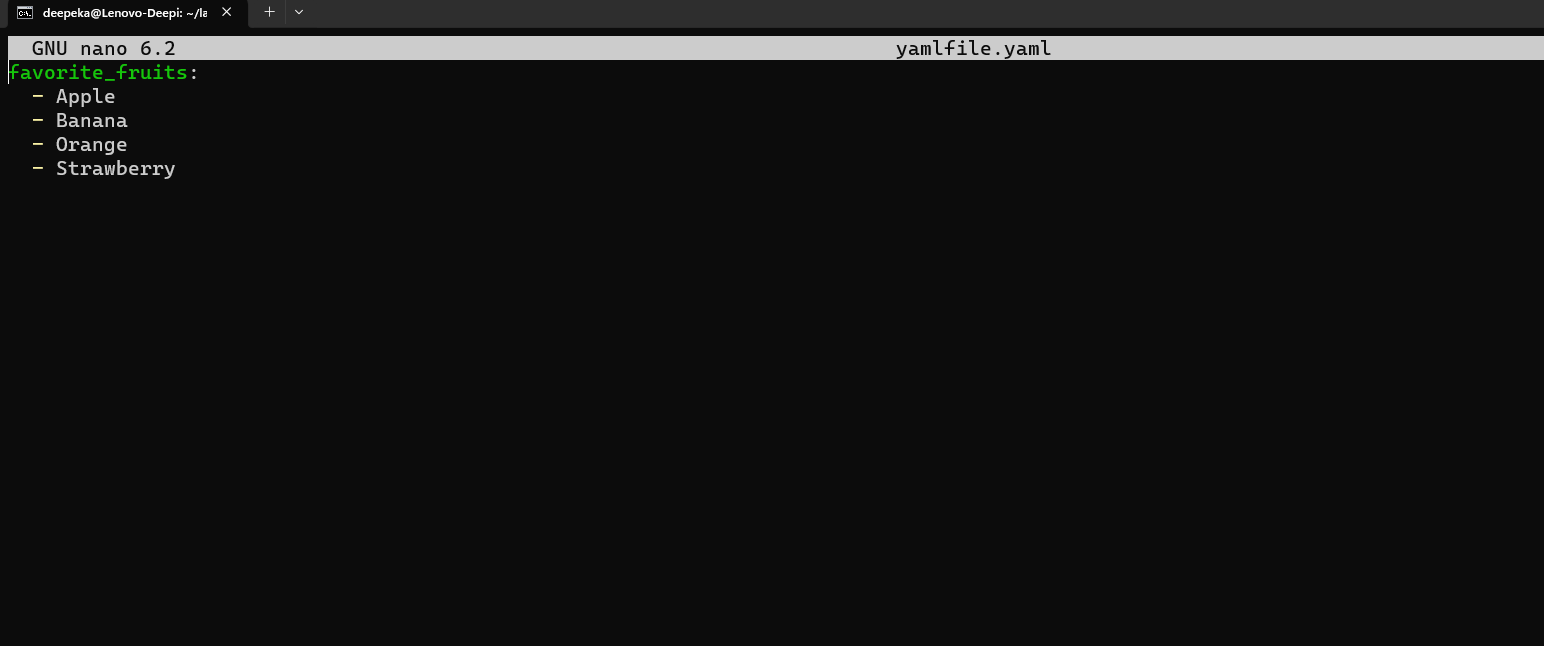
* Task:
  1. Create a YAML file named "fruits.yaml."
  2. Define a list of your favorite fruits using YAML syntax.
  3. Add items from the list.
  4. Save and validate the YAML file.

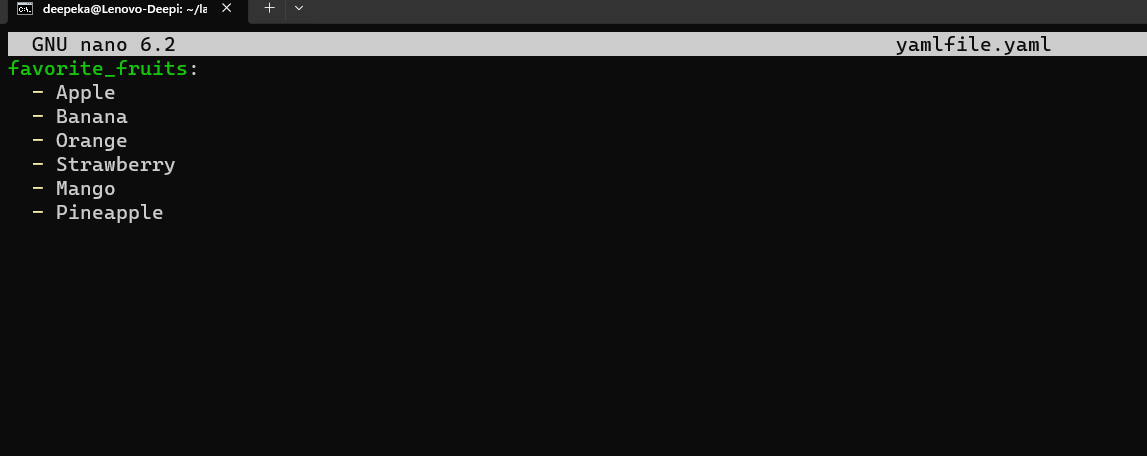
For creating a yaml file “touch” command is used.

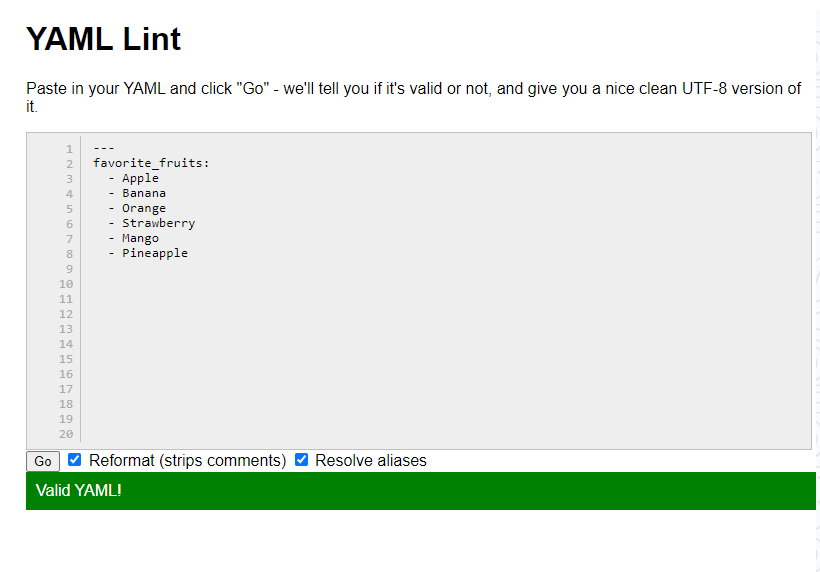
For opening a Nano editor for Linux “nano” command is used followed by the file name.

For validating the yaml file yaml lint is used.









**Lab 6: Nested Structures in YAML**

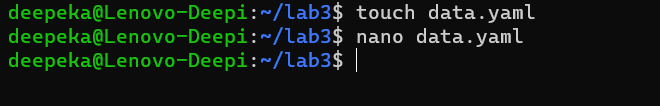
* Task:
  1. Create a YAML file named "data.yaml."
  2. Define a nested structure representing a fictitious organization with departments and employees.
  3. Use YAML syntax to add, update, or remove data within the nested structure.
  4. Save and validate the YAML file.

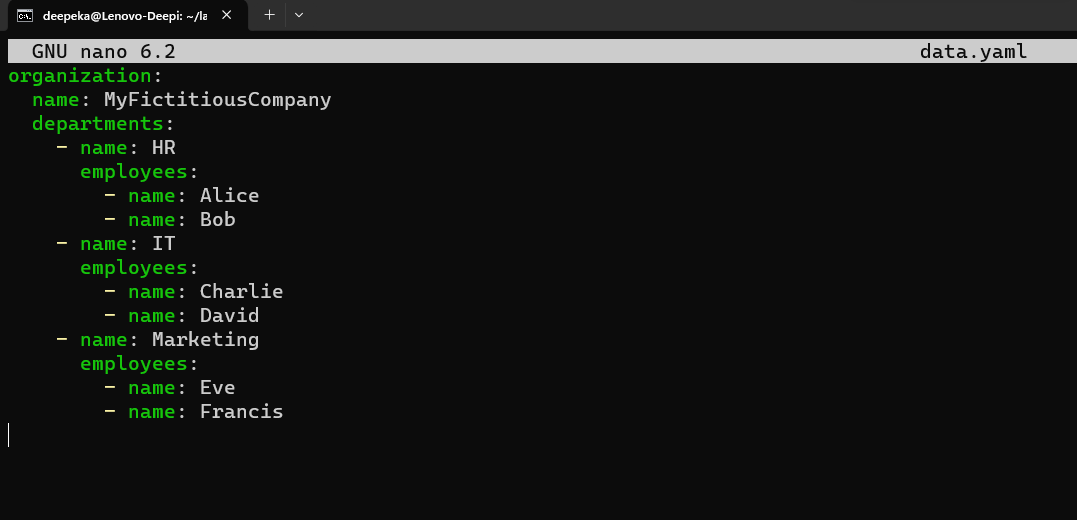
For creating a yaml file “touch” command is used.

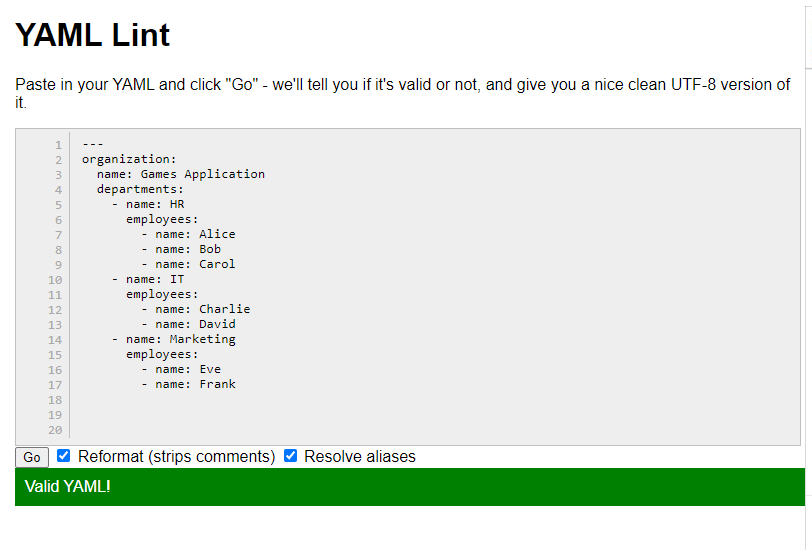
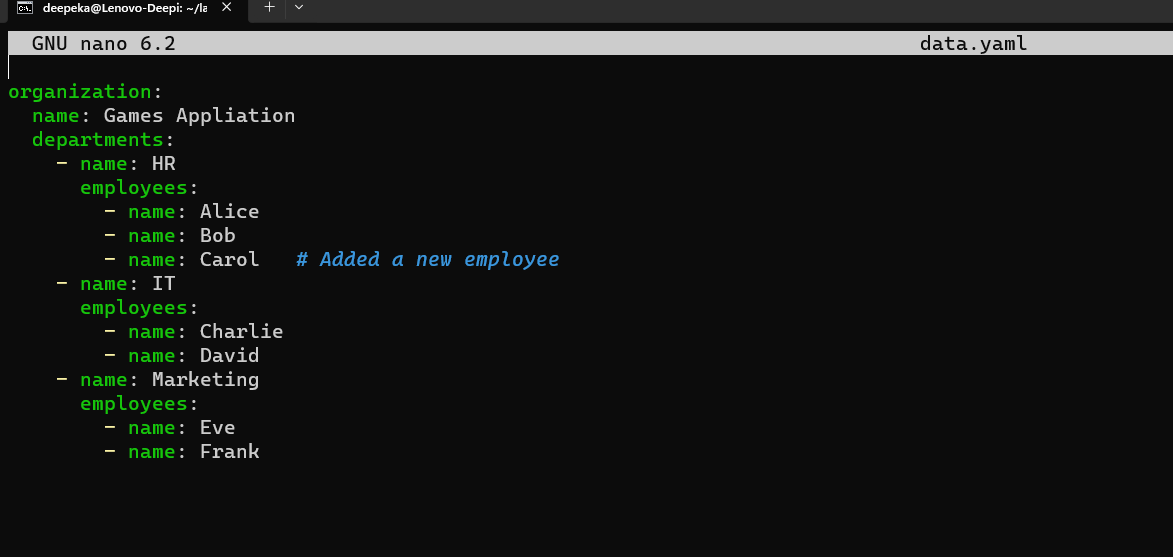
For opening a Nano editor for Linux “nano” command is used followed by the file name.

Even though, you saved the file after editing. You can open last edited file using the same command with file name.

For validating the yaml file yaml lint is used.



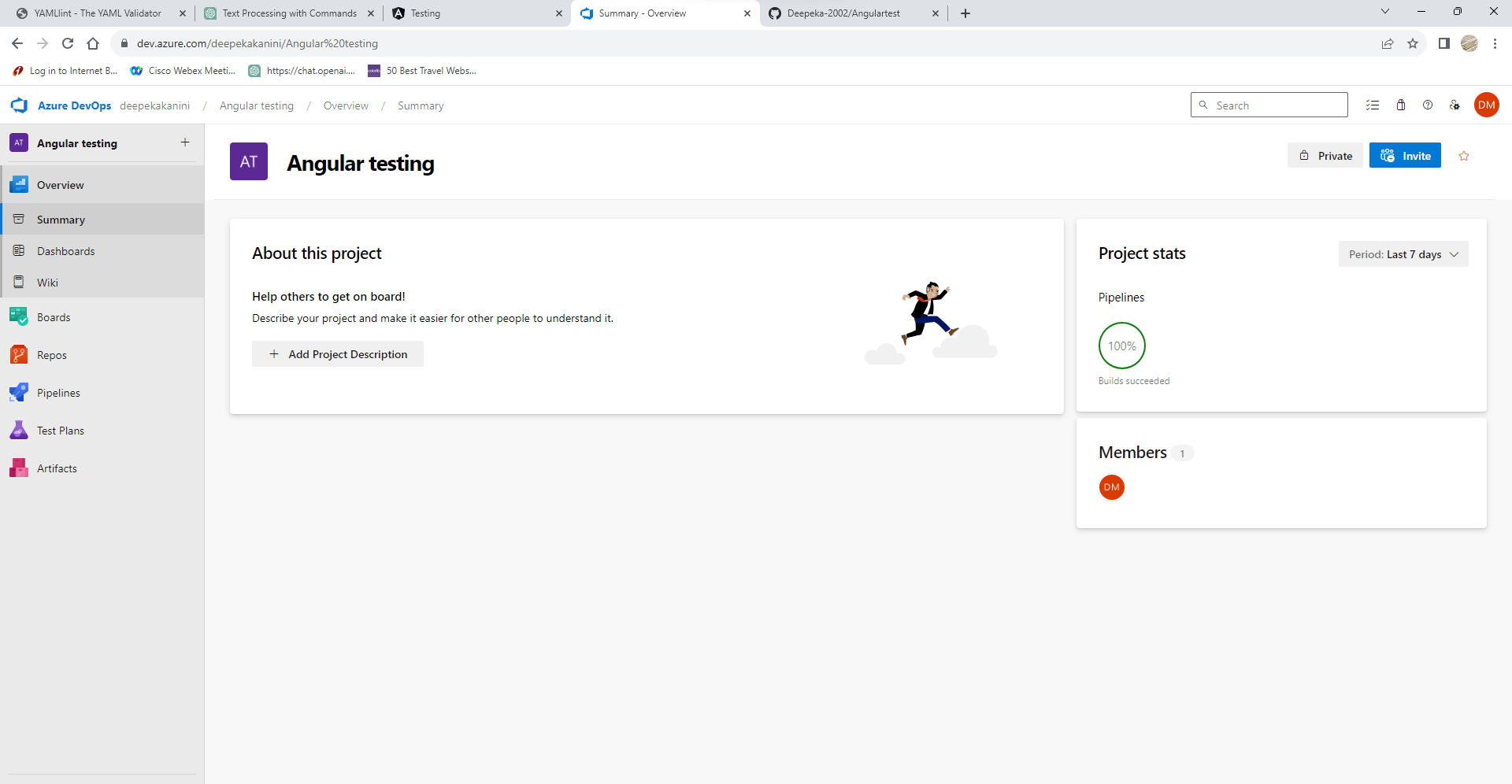




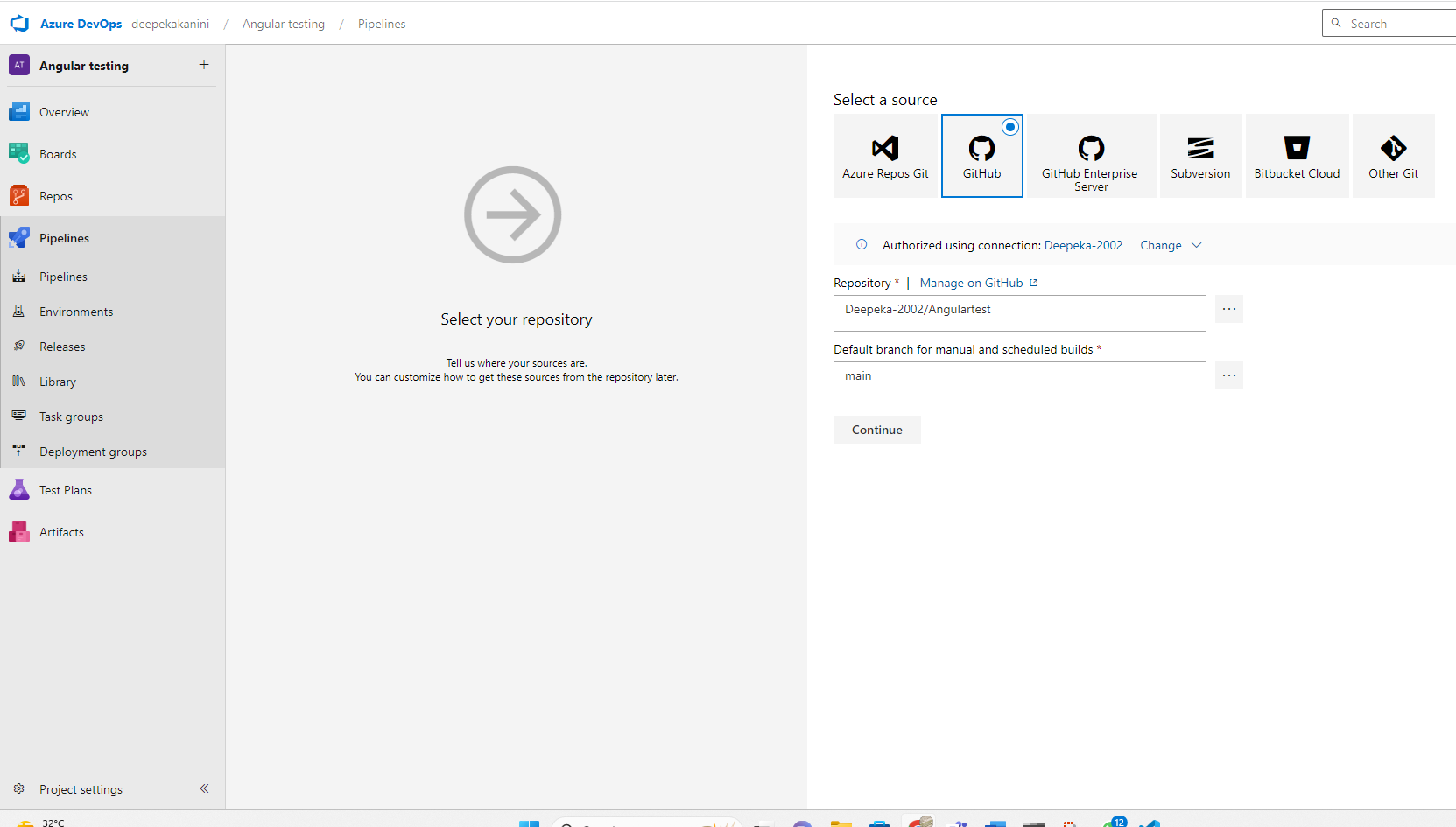
**Lab 7: Create Classic Azure CI Pipeline for Angular Application**

* Tasks:
  1. Create an Azure DevOps project.
  2. Set up a classic CI pipeline to build an Angular application.
  3. Configure the pipeline to use Jasmine and Karma for unit testing.
  4. Run the pipeline and validate the test results.

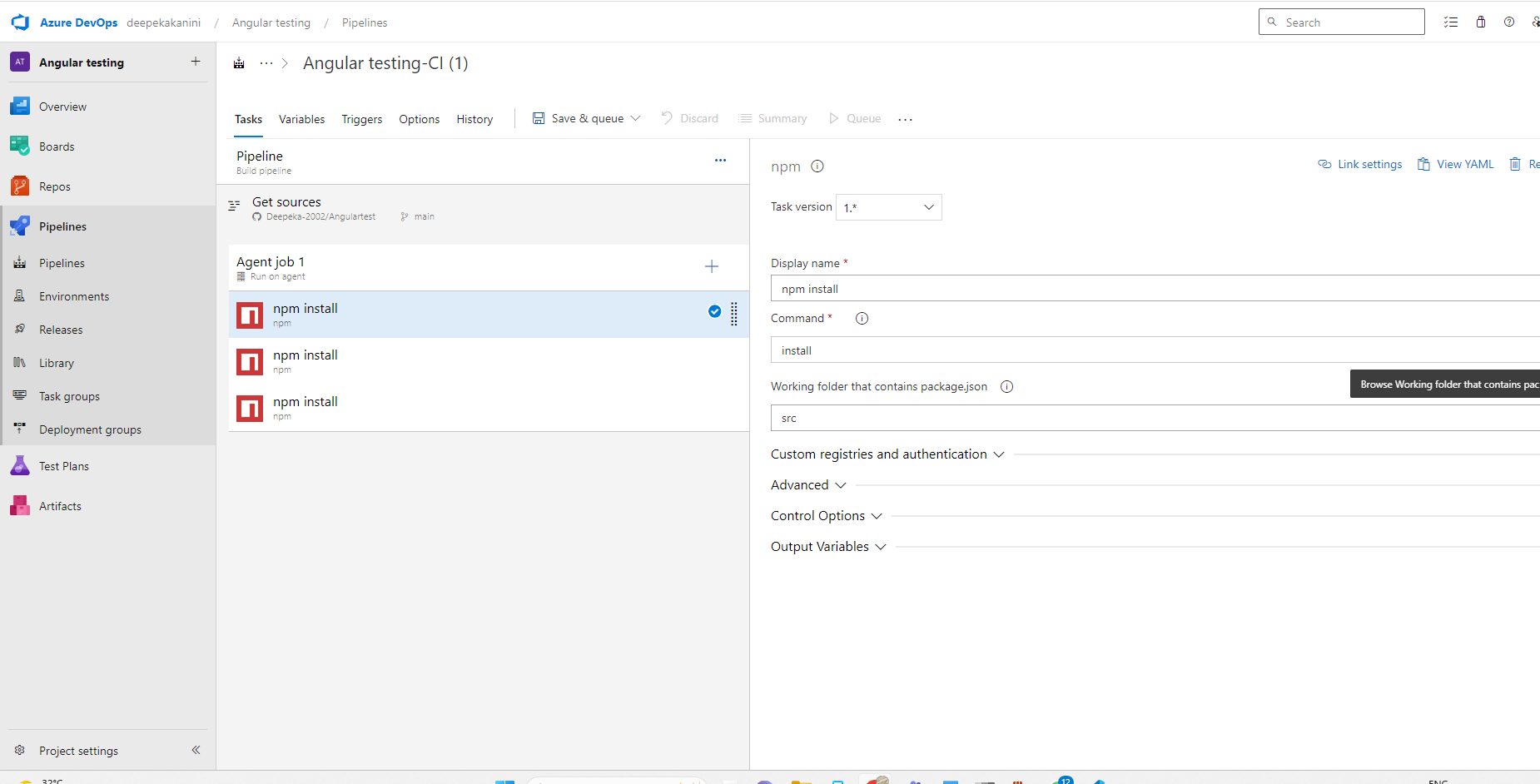
New project “Angular Testing” is created.

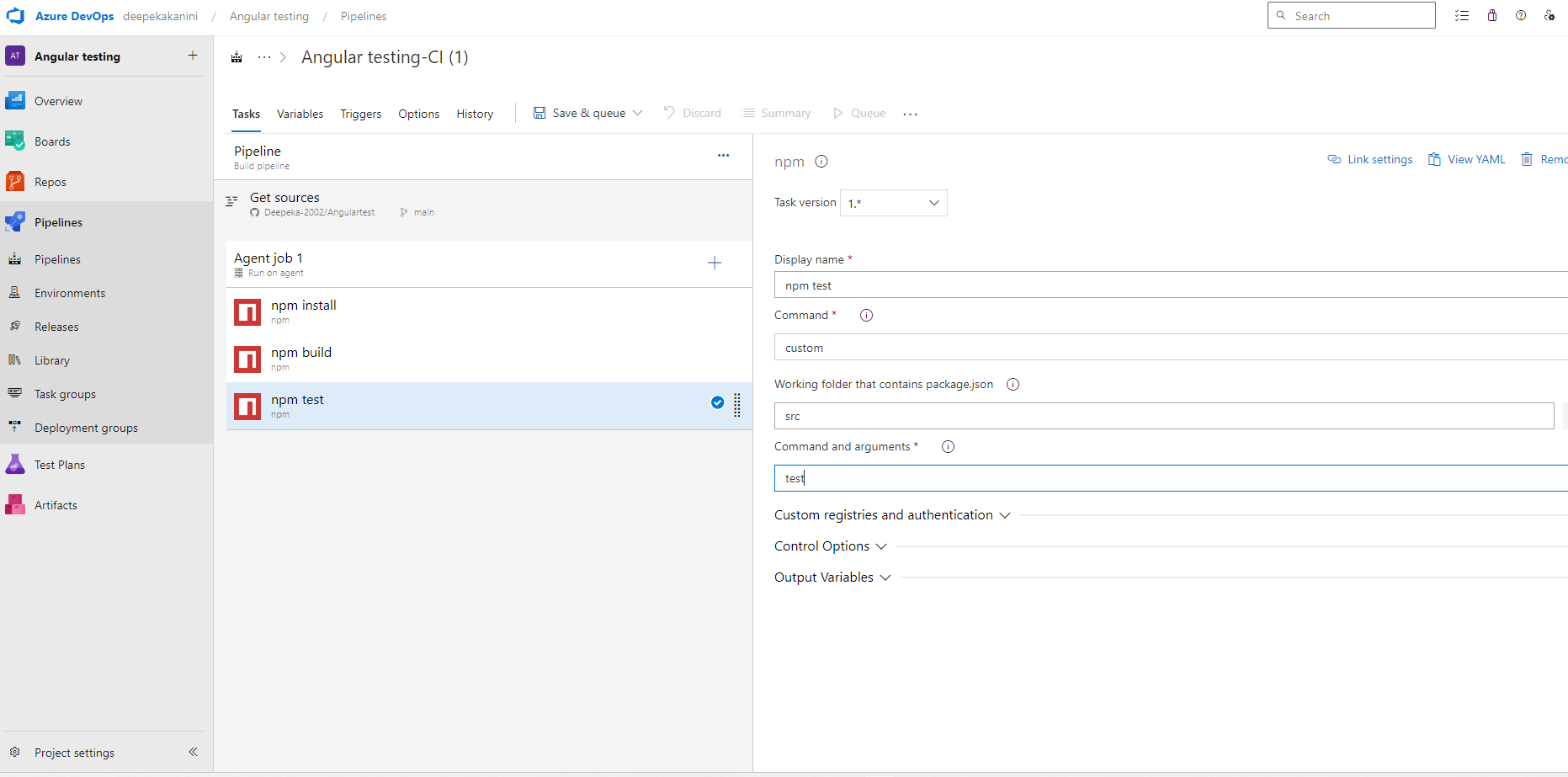


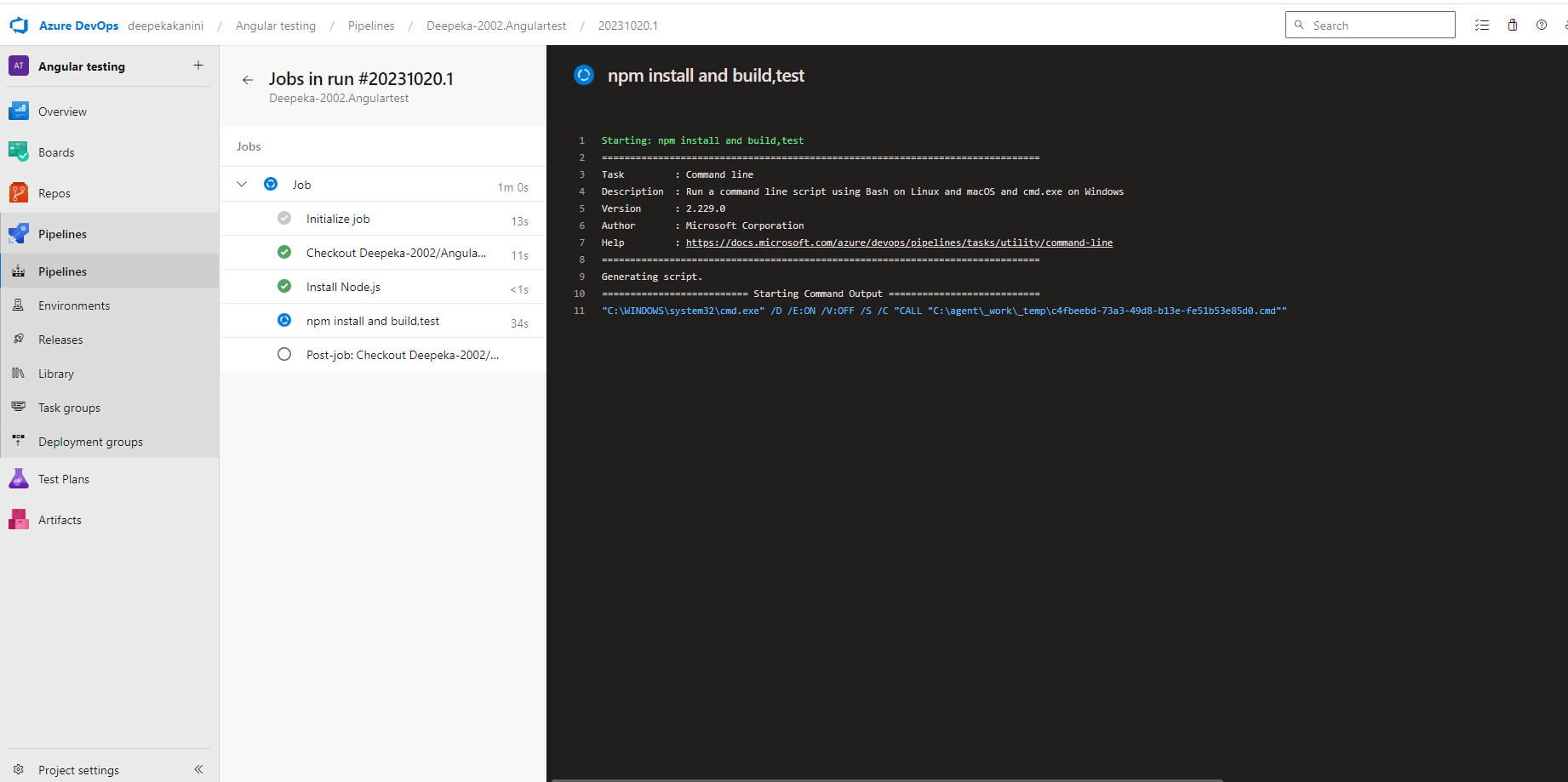
For source control Github repository is used .



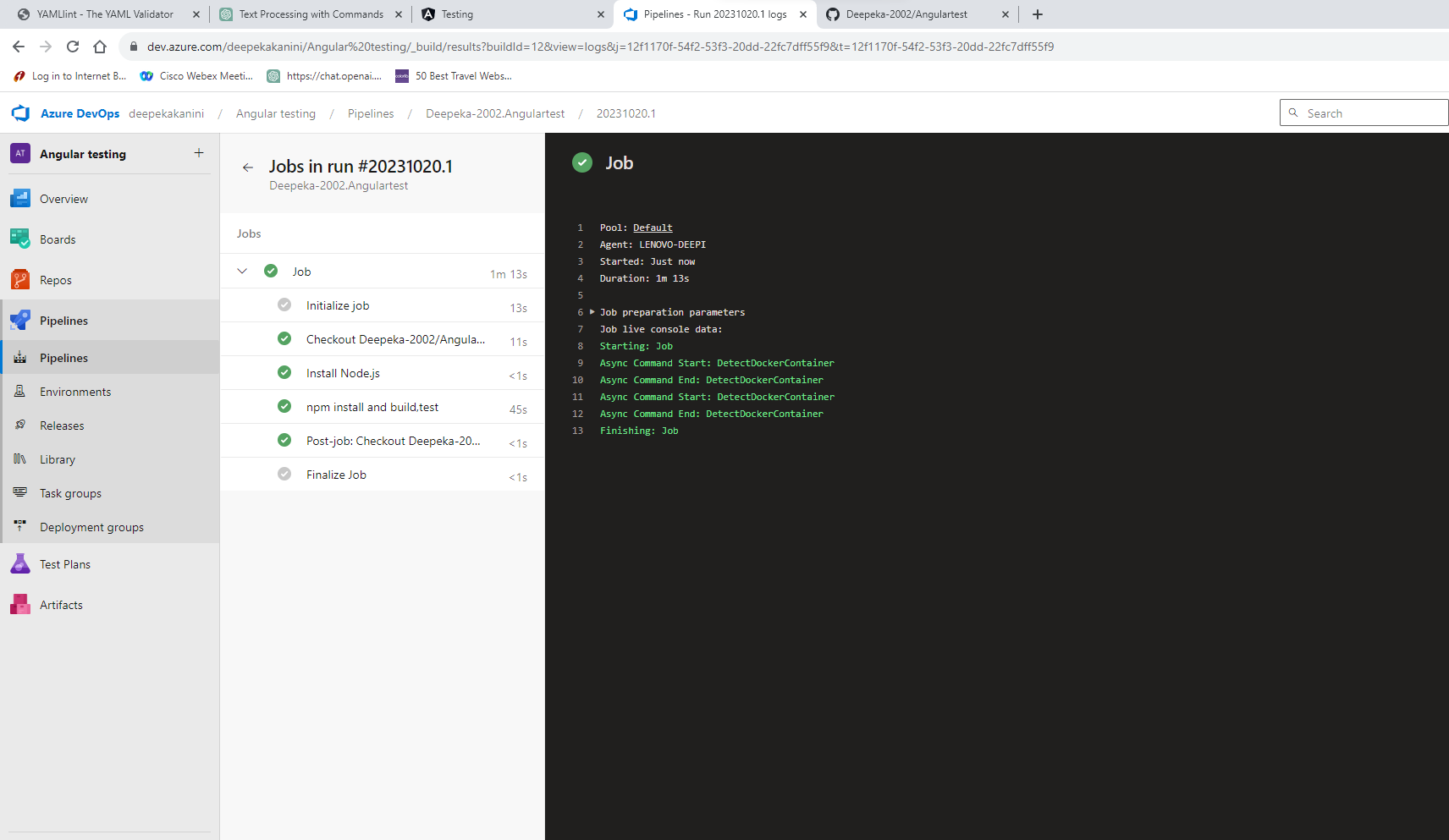
For creating a pipeline for Angular testing project npm install, npm build and npm test command is used.

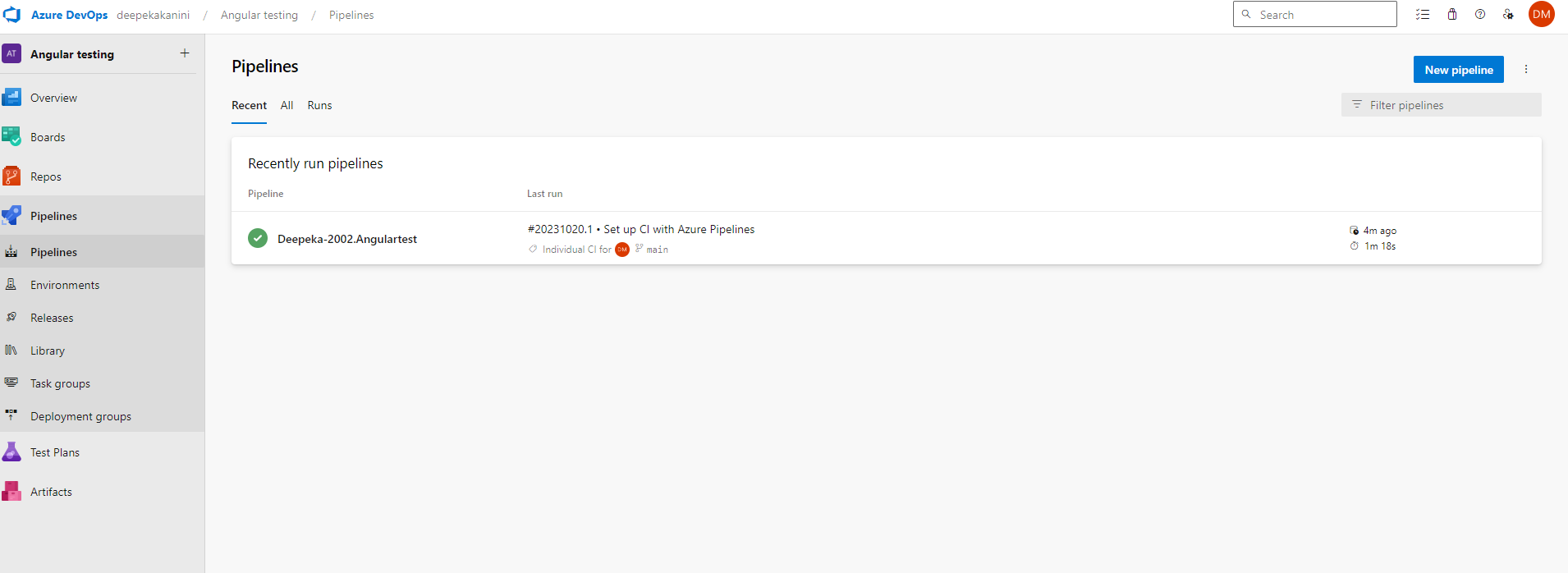






Jobs run successfully without errors and pipeline is created for the project.

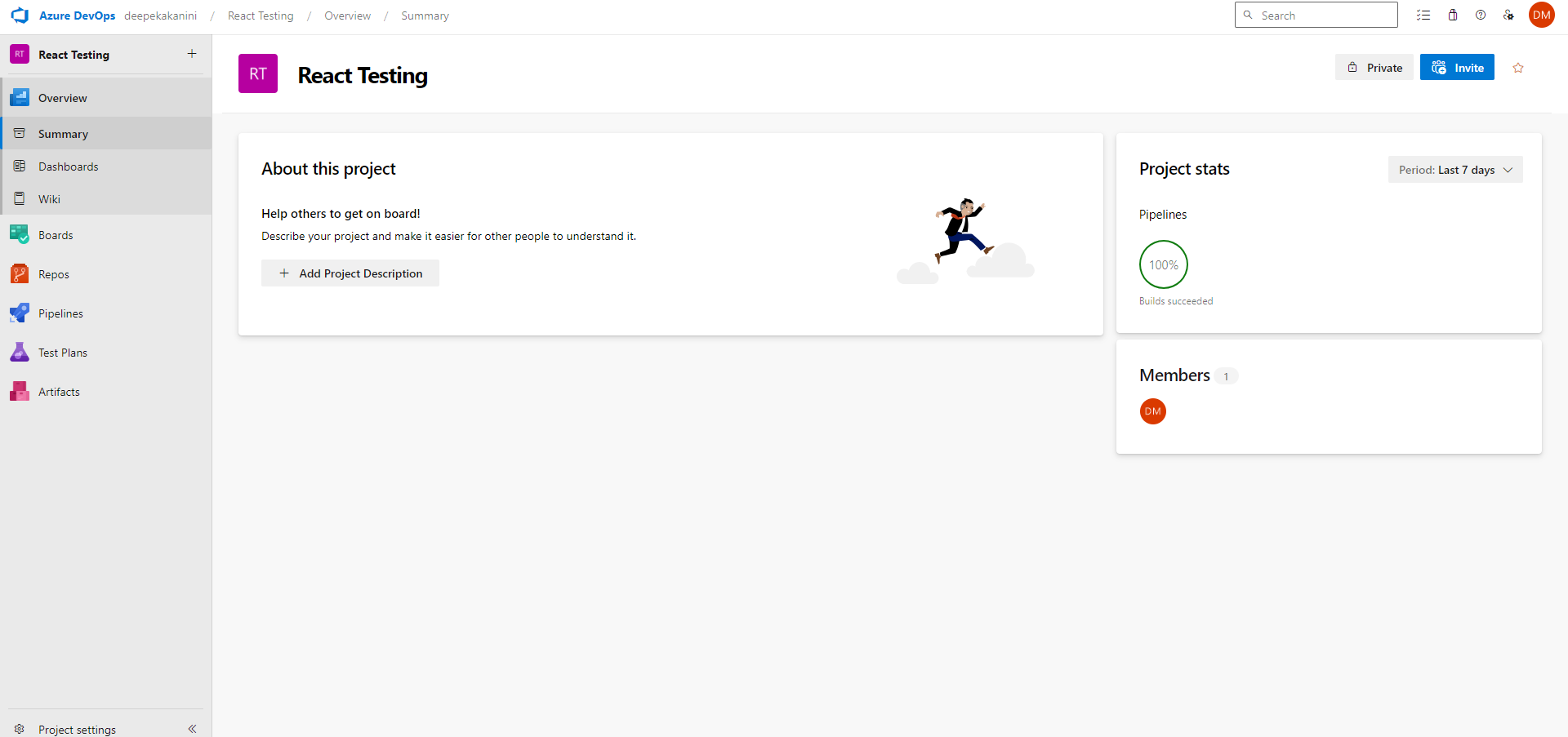




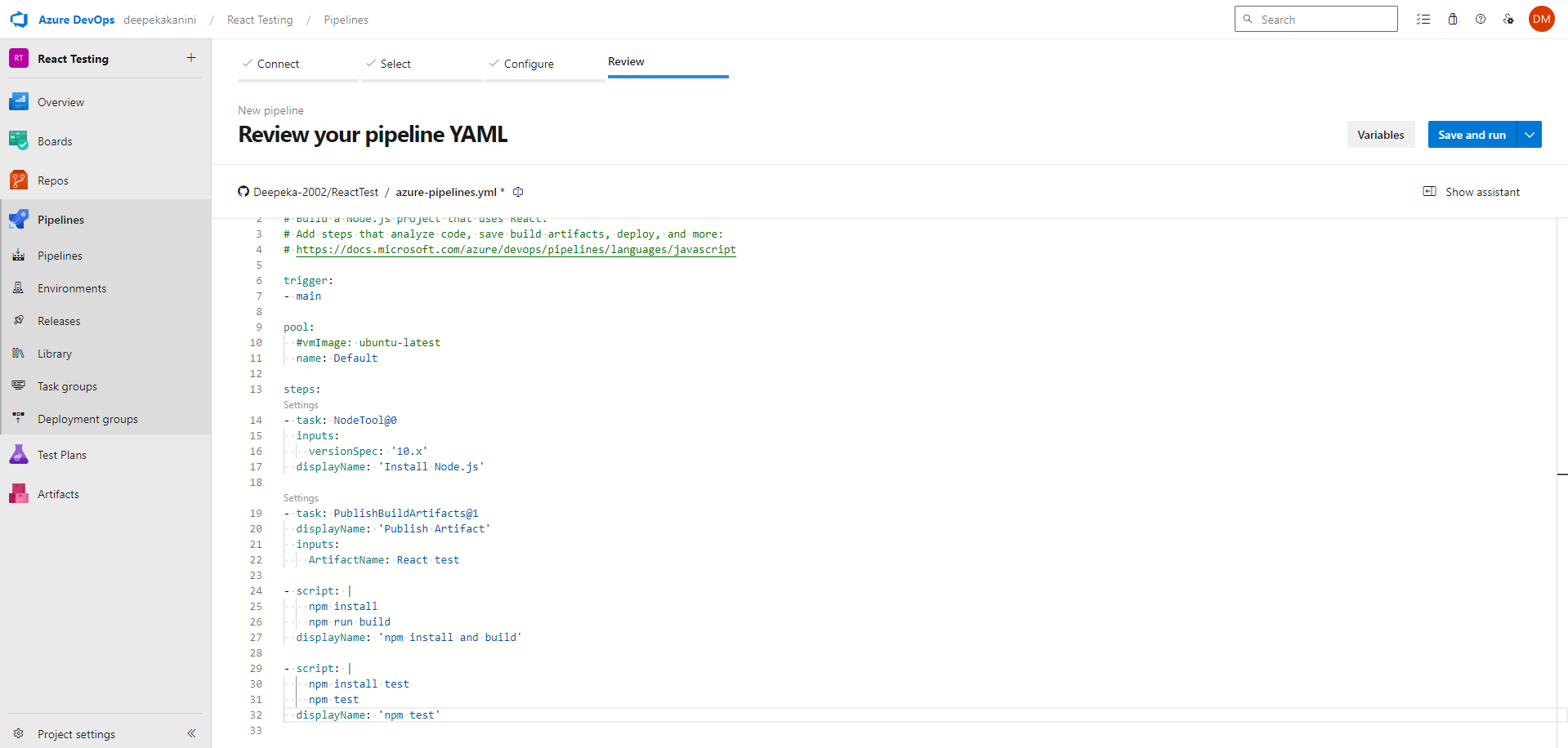
**Lab 8: Create YAML Azure CI Pipeline for React Application**

* Objective: Create a YAML-based Azure CI pipeline to build a simple React application with unit testing using Enzyme and Jest.
* Tasks:
  1. Create an Azure DevOps project.
  2. Create a YAML-based CI pipeline to build a React application.
  3. Configure the pipeline to use Enzyme and Jest for unit testing.
  4. Trigger the pipeline and verify the test results.

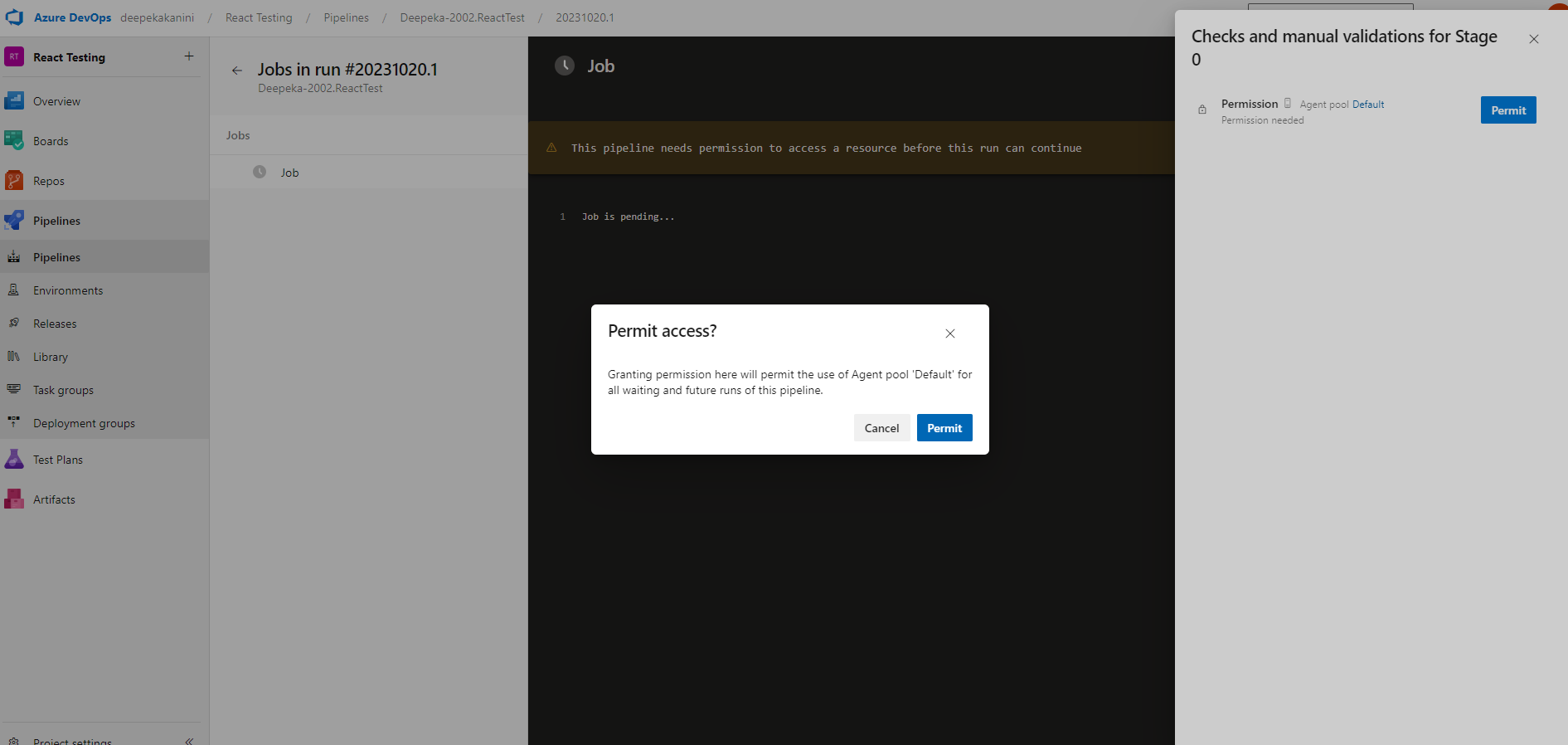
For creating a pipeline for a React Project , project is created.



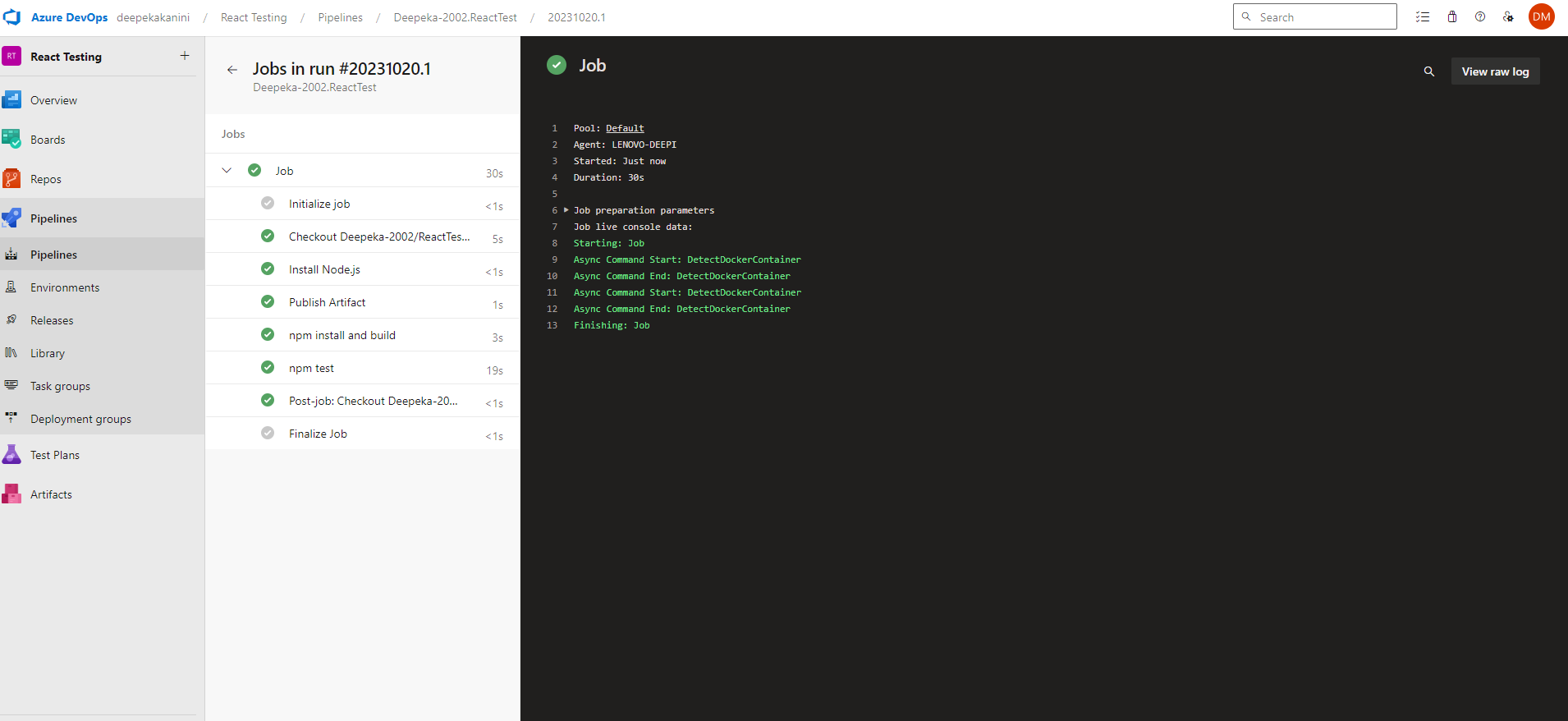
Yaml file is configured for the pipeline using Publish Artifacts.

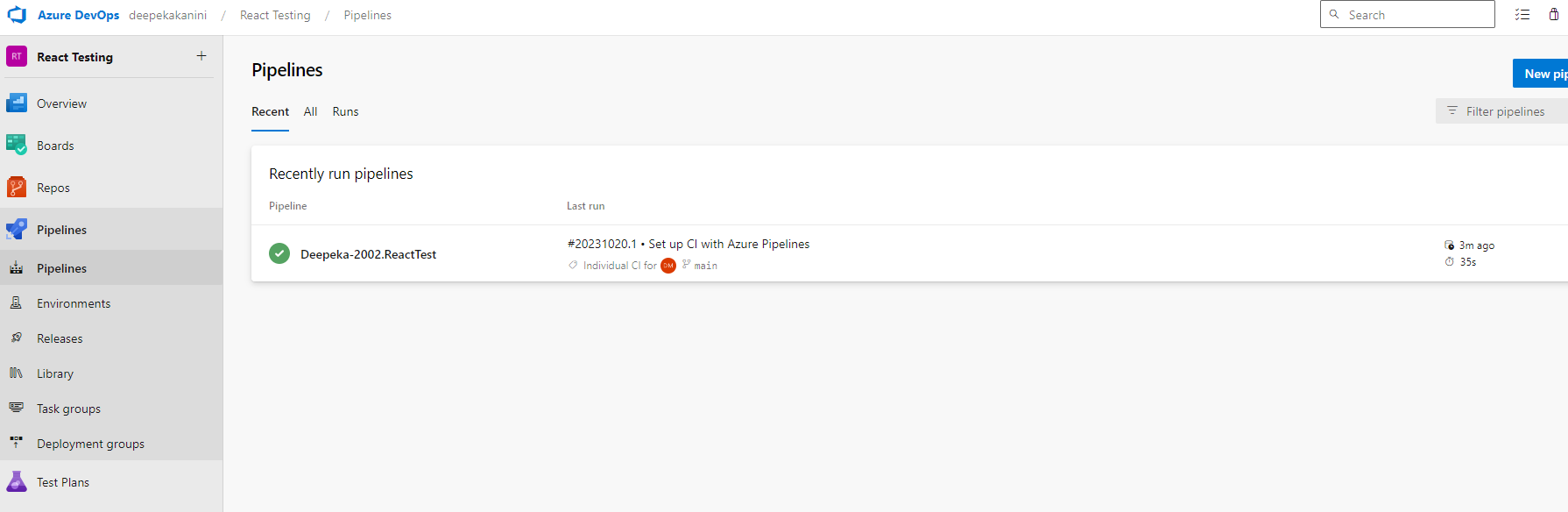


It wants access to run the pipeline. When permission is given job will run.



Jobs run successfully and the pipeline is created for the project.

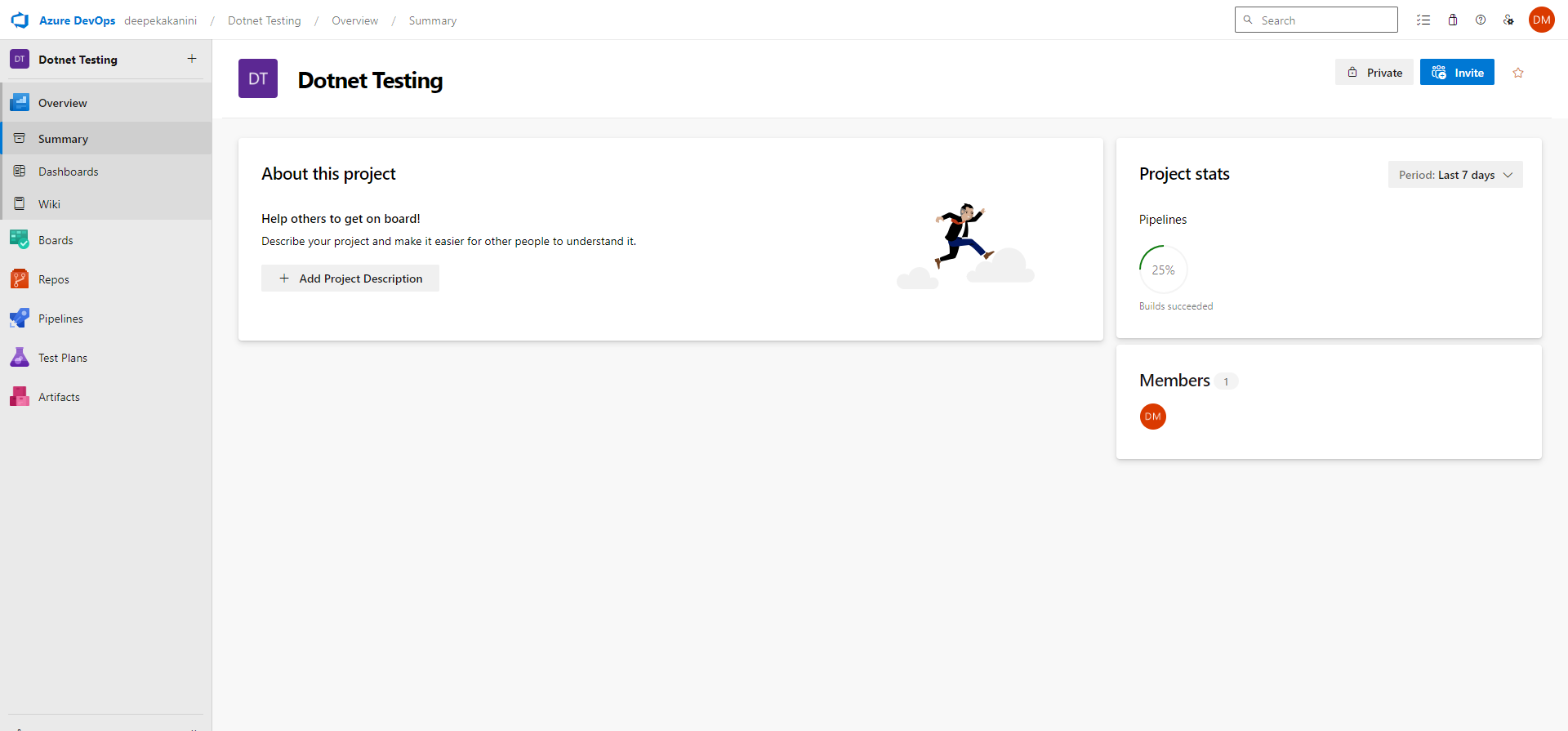




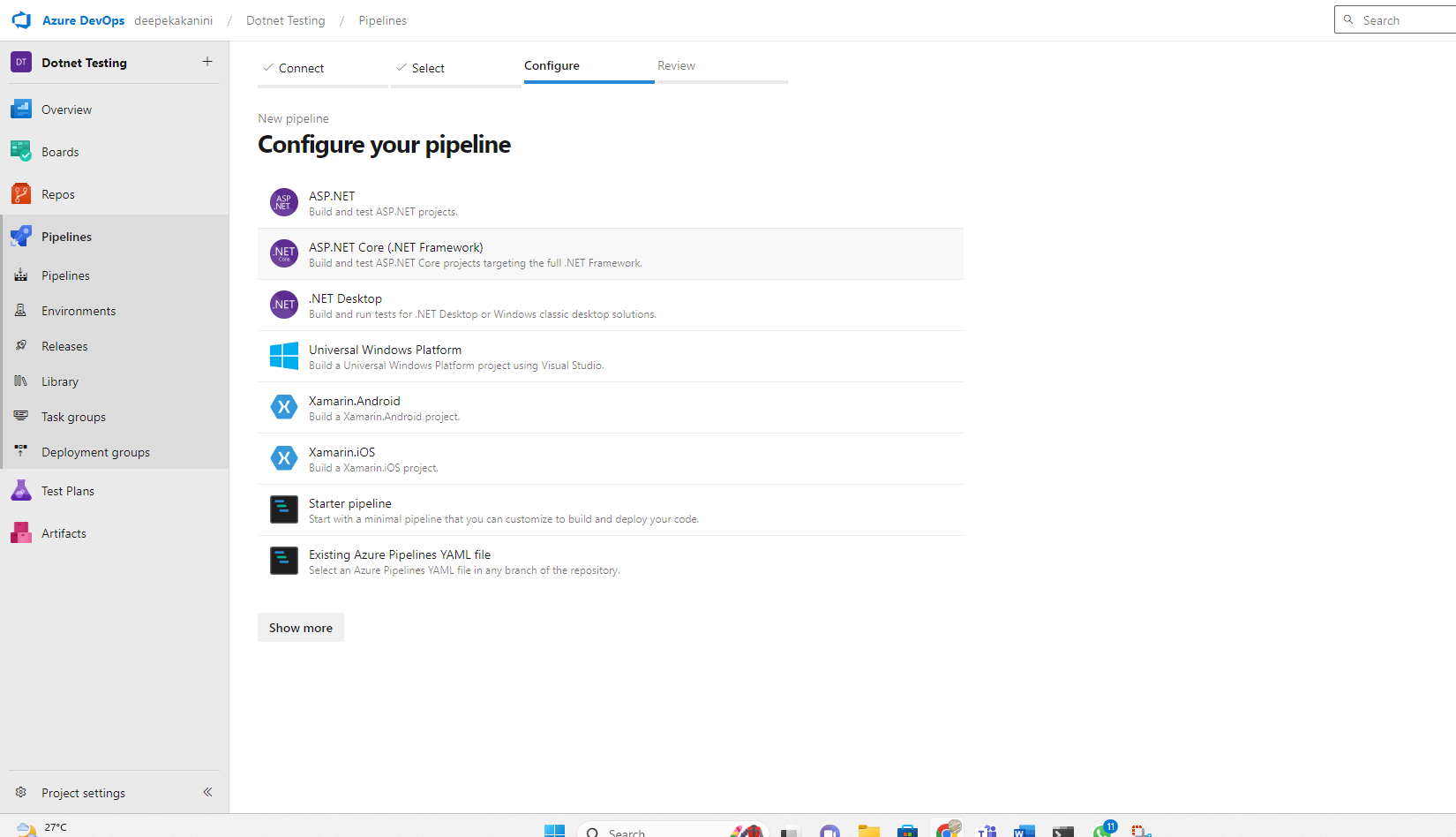
**Lab 9: Create CI Pipeline for .NET Core Application with MS Unit Test**

* Objective: Create a CI pipeline, either classic or YAML, to build a .NET Core application and run MS Unit tests.
* Tasks:
  1. Set up a new Azure DevOps project.
  2. Create a CI/CD pipeline for a .NET Core application.
  3. Configure the pipeline to use MS Unit tests.
  4. Trigger the pipeline and validate the test results

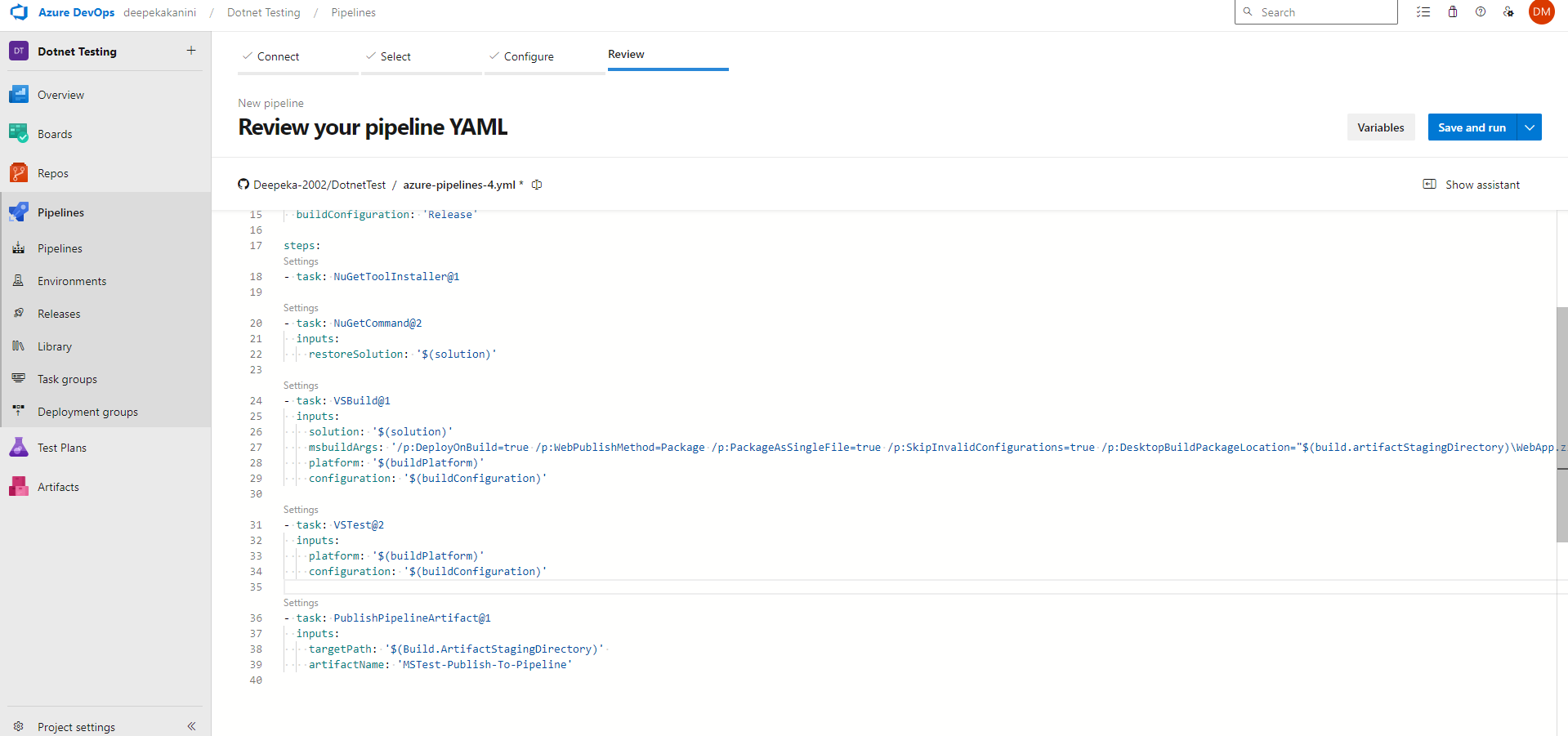
For creating a Pipeline for the DotNet with Ms Test , Azure Devops project is created.



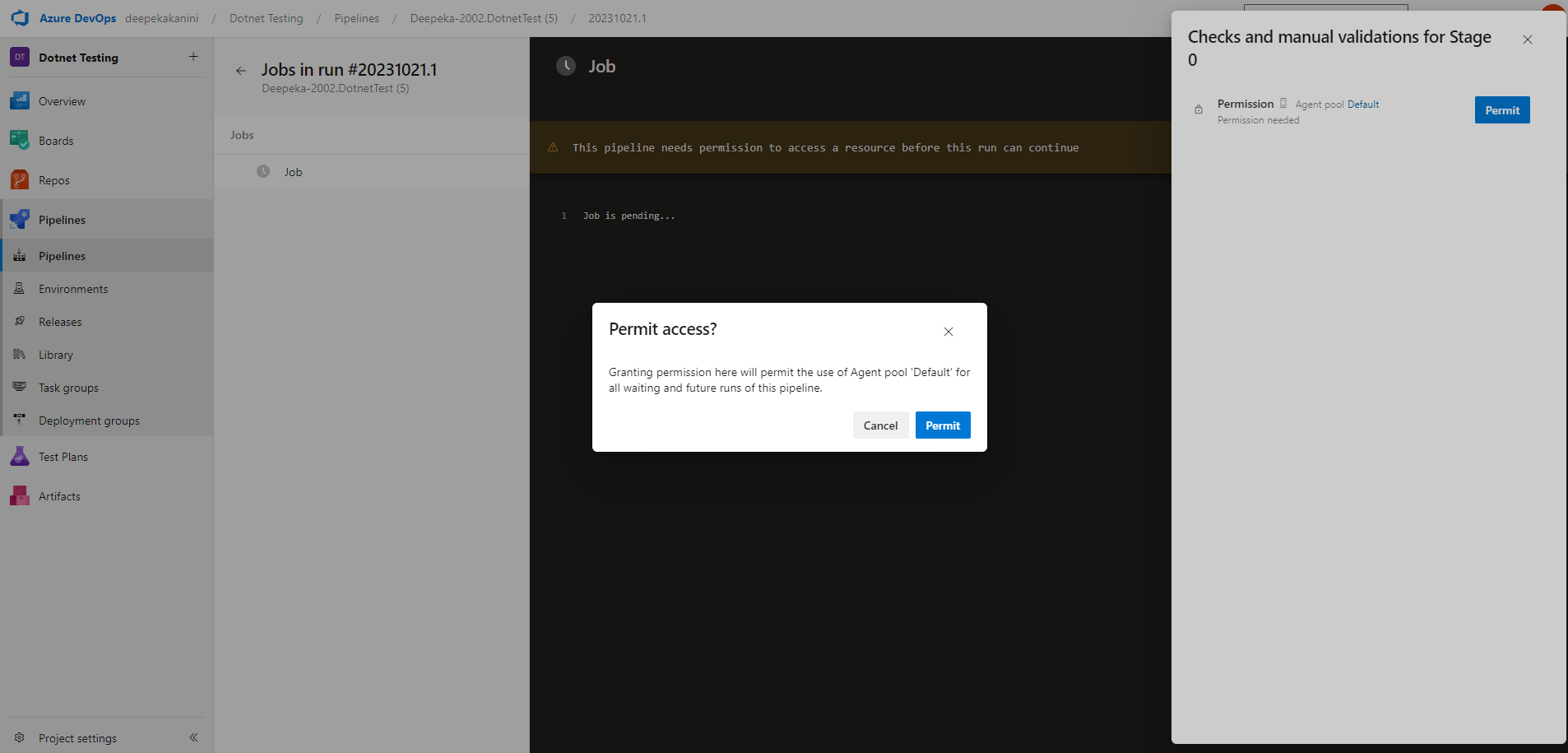
For creating the pipeline using Yaml File, it can be chose from Git Repository and from ASP.NET Core (.Net Framework) configuration.



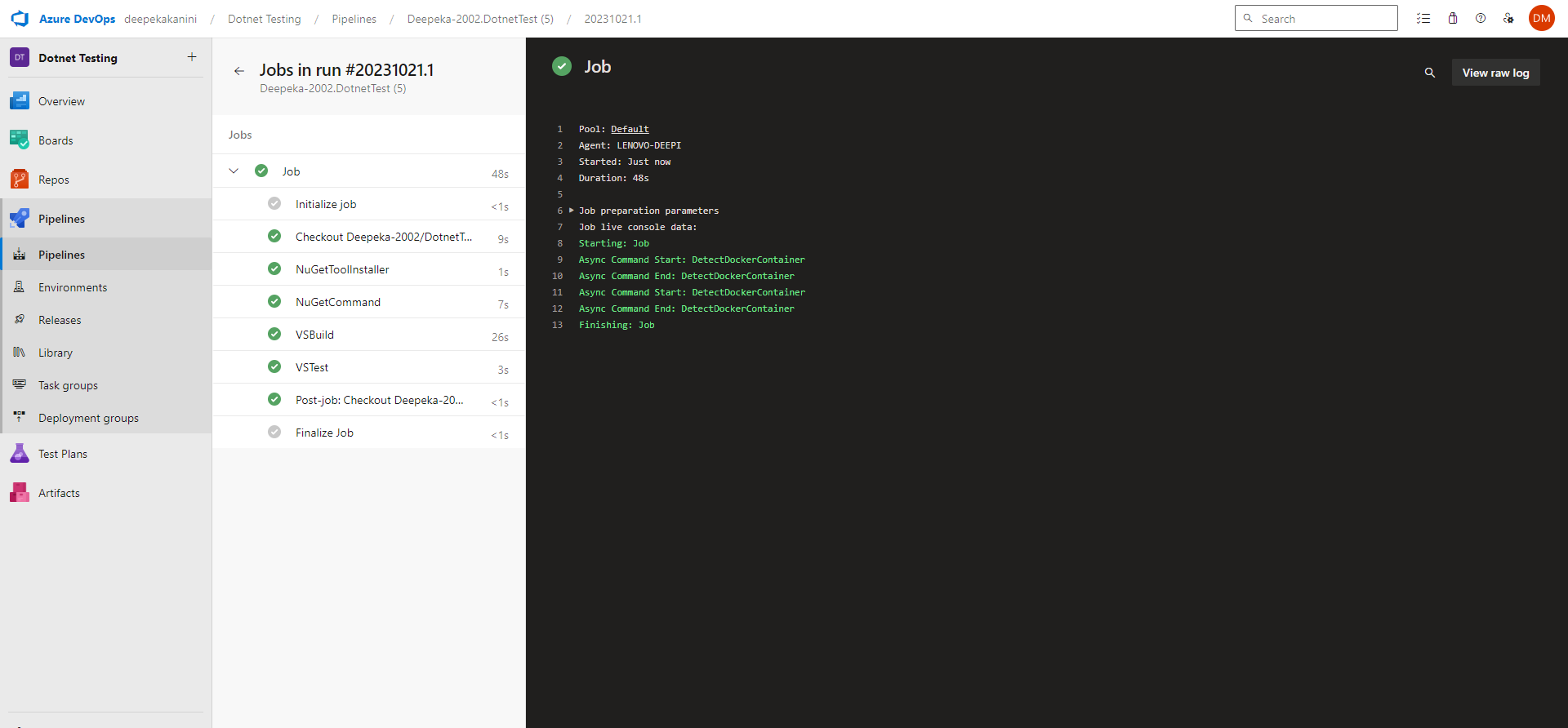
For creating a pipeline, yaml file is configured using artifacts.



For running the job, permission should be given.



Jobs run successfully and the pipeline was created for the project.



**Lab 10: Creating a Docker Image for a .NET Core Web API and Running it in Rancher Desktop**

**Objective**: In this lab, you will create a Docker image for a sample .NET Core Web API application and then run the Web API container in Rancher Desktop.

**Prerequisites:**

* Rancher Desktop installed and running.
* .NET Core SDK installed on your machine.

**Tasks**

Step 1: Create a .NET Core Web API Project

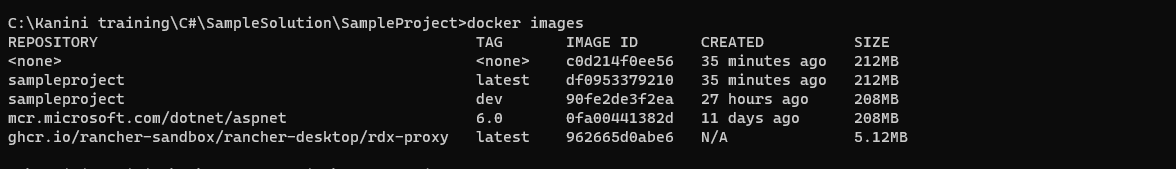
Step 2: Build the .NET Core Web API Project

Step 3: Dockerize the .NET Core Web API

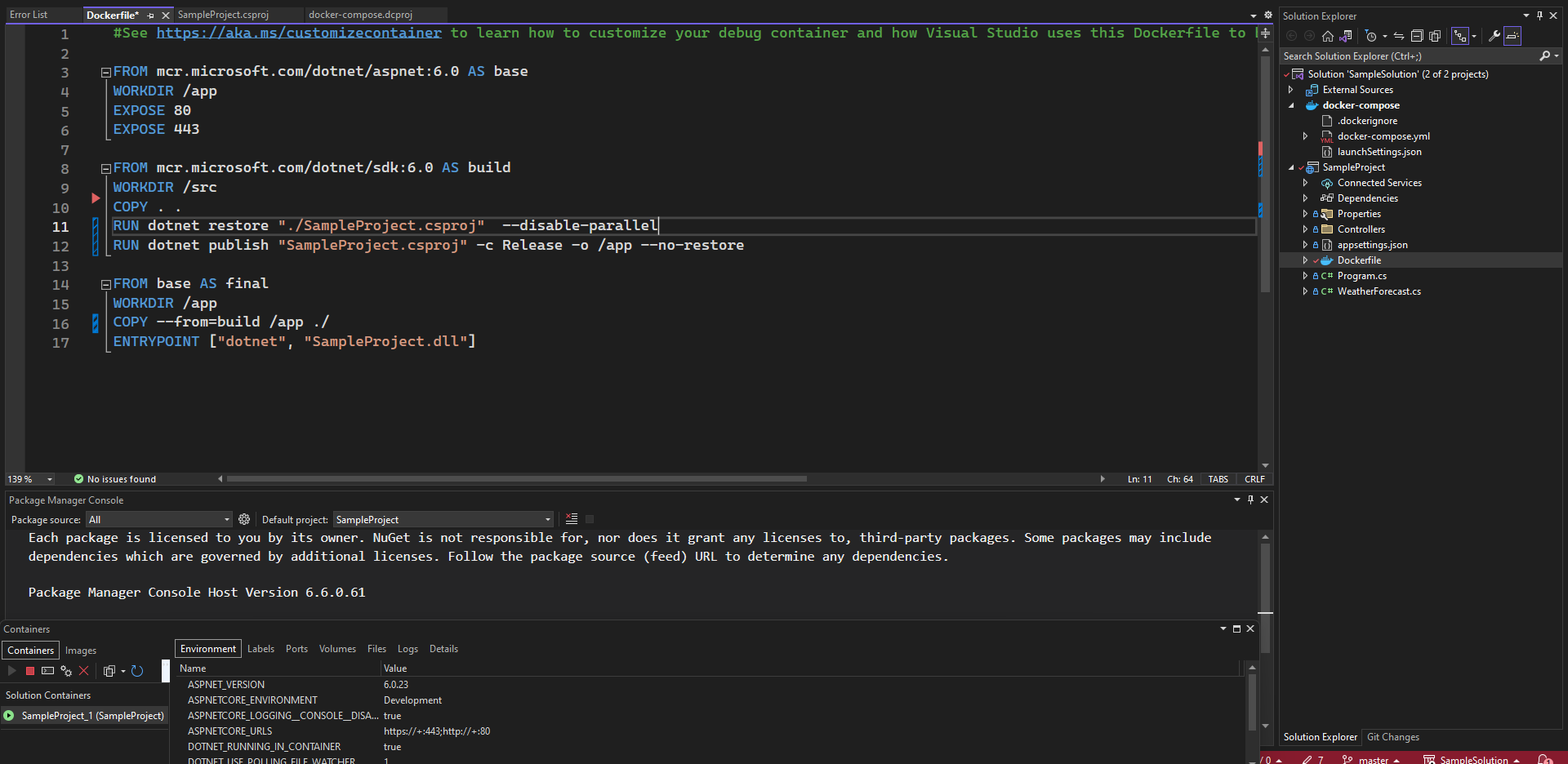
Step 4: Build the Docker Image

Step 5: Run the Docker Container in Rancher Desktop  
Step 6: Test the .NET Core Web API via swagger

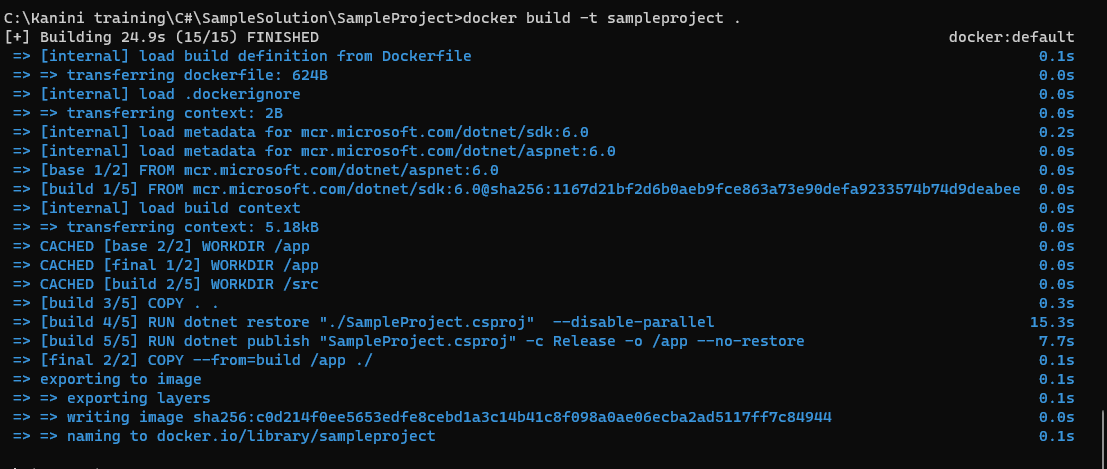
Docker Images :



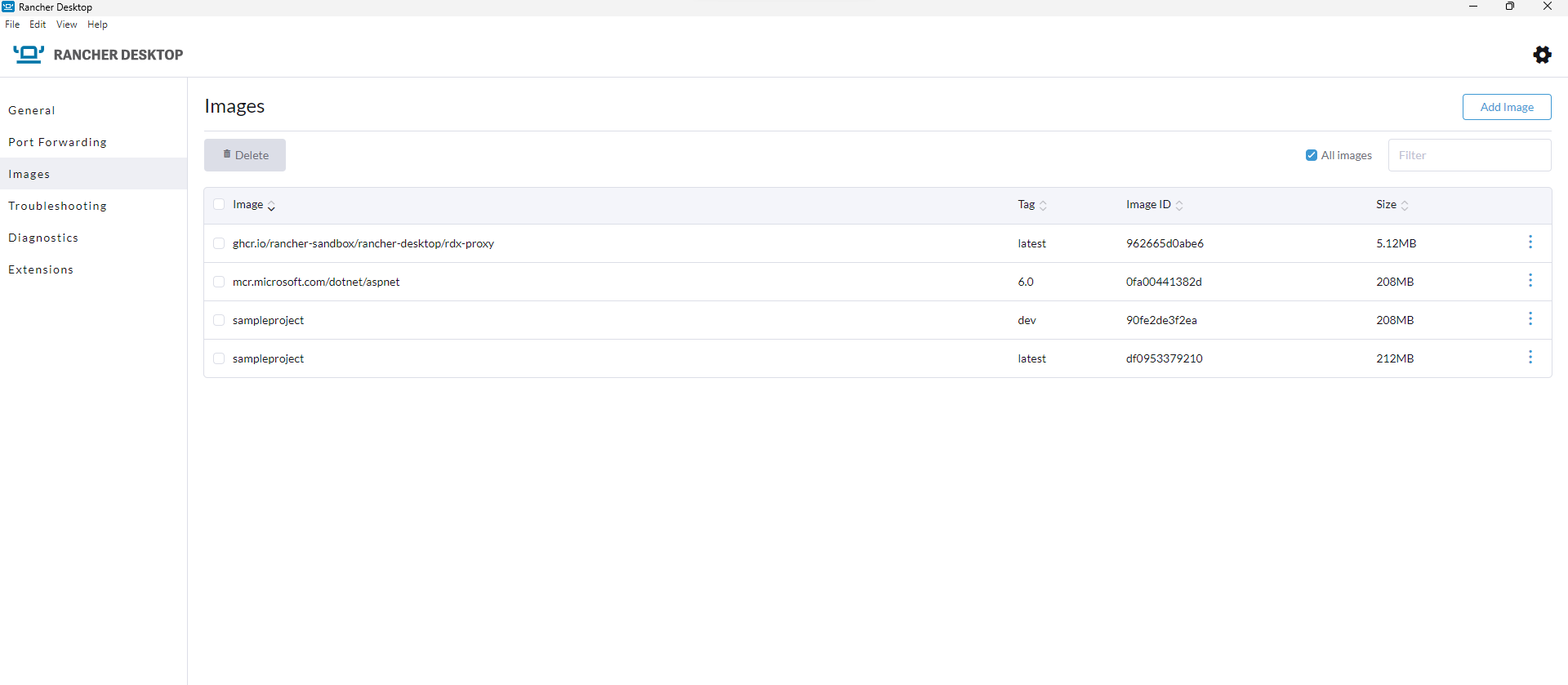
Docker File for the project in development



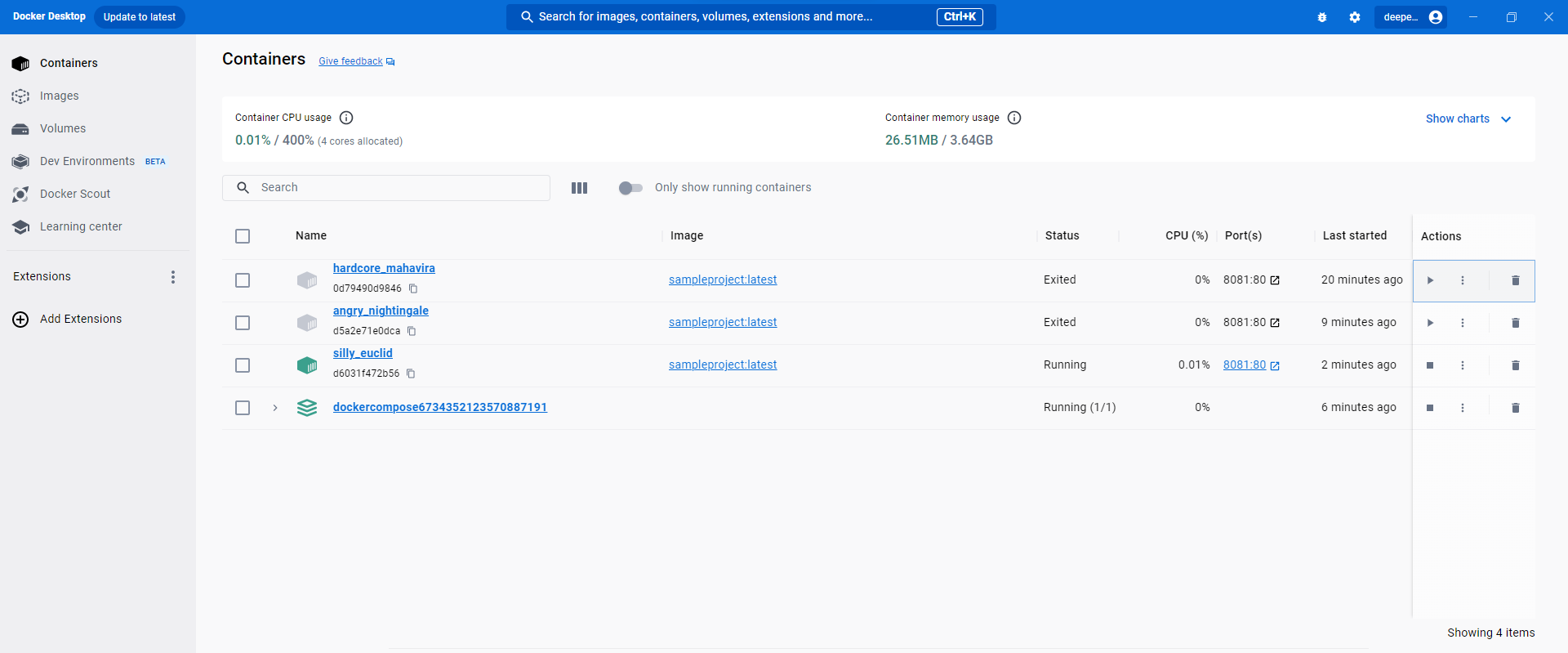
Building the Docker image

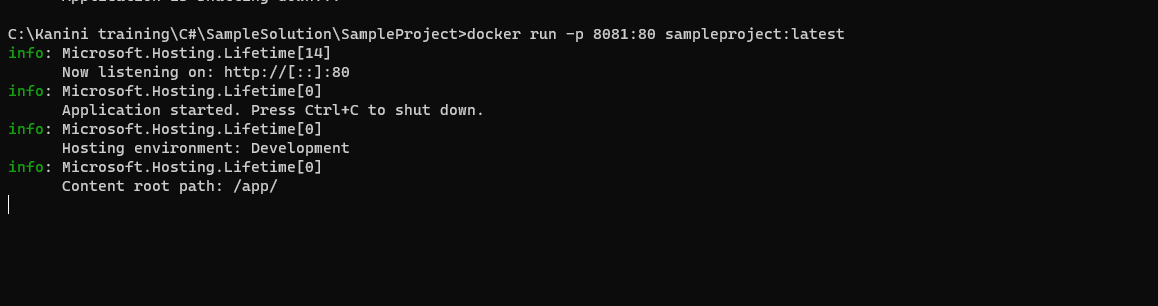


Docker images in Rancher Desktop



Docker image in Docker Desktop





Swagger check for the project

