**DevOps – Final Assessment**

Section 1: Multiple-Choice Questions (MCQs):

1. What does WSL stand for in the context of Windows?

Ans: c. Windows Subsystem for Linux

1. What is the primary goal of continuous integration (CI) in DevOps?

Ans: b. Frequent integration of code changes

1. In the Linux command line, what does the **cd** command do?

Ans: b. Change the working directory

1. Which of the following is not a Linux distribution?

Ans: c. Docker

1. What is Docker primarily used for in DevOps and containerization?

Ans: c. Packaging and deploying applications in containers

1. What is the primary purpose of Azure DevOps?

Ans: b. Software development and delivery

1. Which components are part of Azure DevOps?

Ans: c. Azure Boards and Azure Pipelines

1. How does Azure DevOps support version control in software development?

Ans: b. It tracks changes in source code and manages versions.

1. In Linux, what is the primary role of the root user?

Ans: c. Administrative tasks with superuser privileges

1. In Azure DevOps, which component is used to define, build, test, and deploy applications? Ans: c. Azure Pipelines.

Section 2: Labs:

Lab 1: File and Directory Management:

Objective: Practice basic file and directory management commands.

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

Ans:

1. To create a directory, use **mkdir** command.

Eg: mkdir lab1

1. Now, list the directories using **ls,** move to lab1 using **cd,** create a file using **touch.**

Eg: ls

Eg: cd lab1

Eg: touch sample.txt

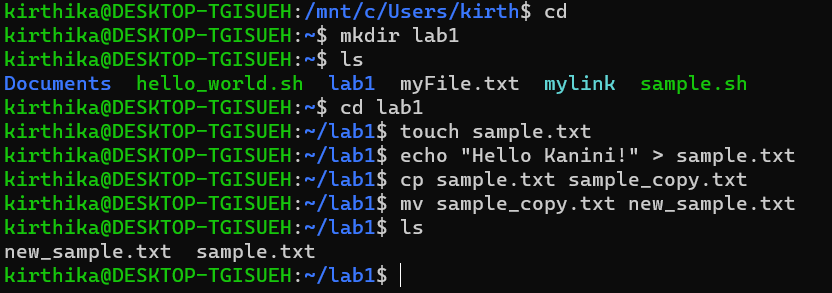
1. Make a copy of sample.txt and name it sample\_copy.txt using **cp**

Eg: cp sample.txt sample\_copy.txt

1. Rename it to new\_sample.txt using **mv**

Eg: mv sample\_copy.txt new\_sample.txt

1. List using **ls.**



**Lab 2: Permissions and Ownership**

Objective: Understand and manage file 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.

Ans:

1. Create new file using **touch**

Eg: touch secret.txt

1. Set file permissions to allow read, write access using **chmod 600**

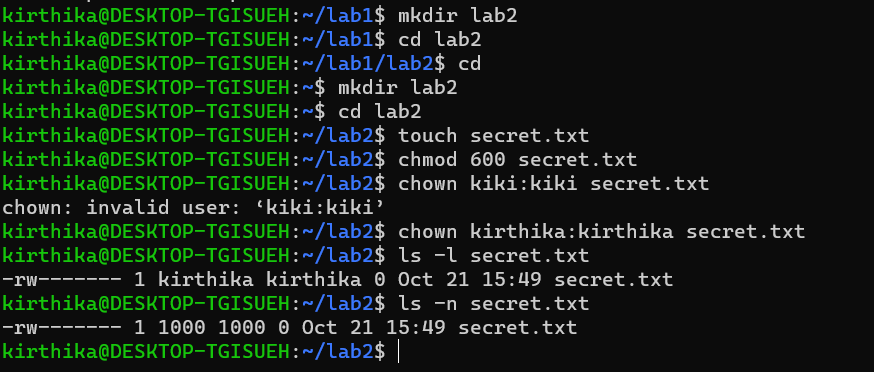
Eg: chmod 600 secret.txt

1. Change owner using **chown**

Eg: chmod mahasri:mahasri secret.txt

1. Verify new permission using **ls -l , ls -n**

Eg: ls-l secret.txt and ls-n secret.txt



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

Objective: Practice text processing using 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.

Ans:

1. Create text file using **touch**

Eg: touch sample.txt

1. Use **grep** to search for a word

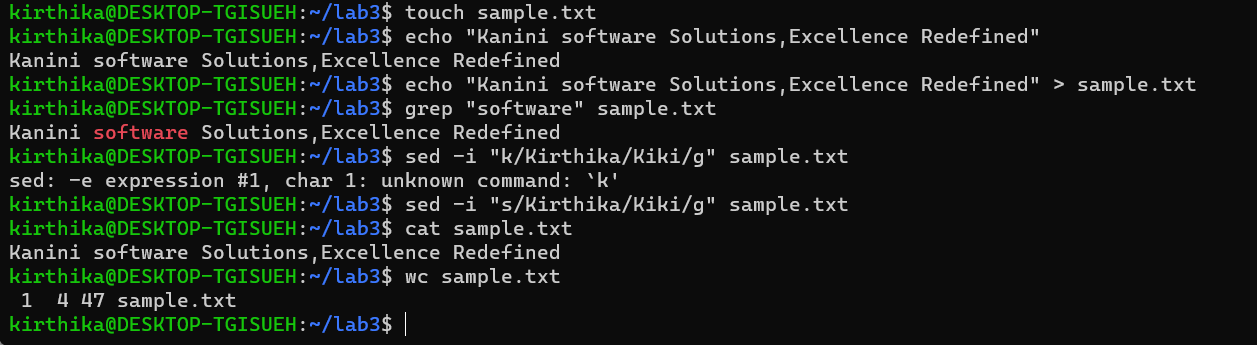
Eg: grep "kanini" sample.txt

1. Use **sed** to replace a word

Eg: sed -i "s/Mahasri/Maha/g" sample.txt

1. Use **wc** to count no.of lines

Eg: wc sample.txt



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

Objective: Create a basic YAML configuration file.

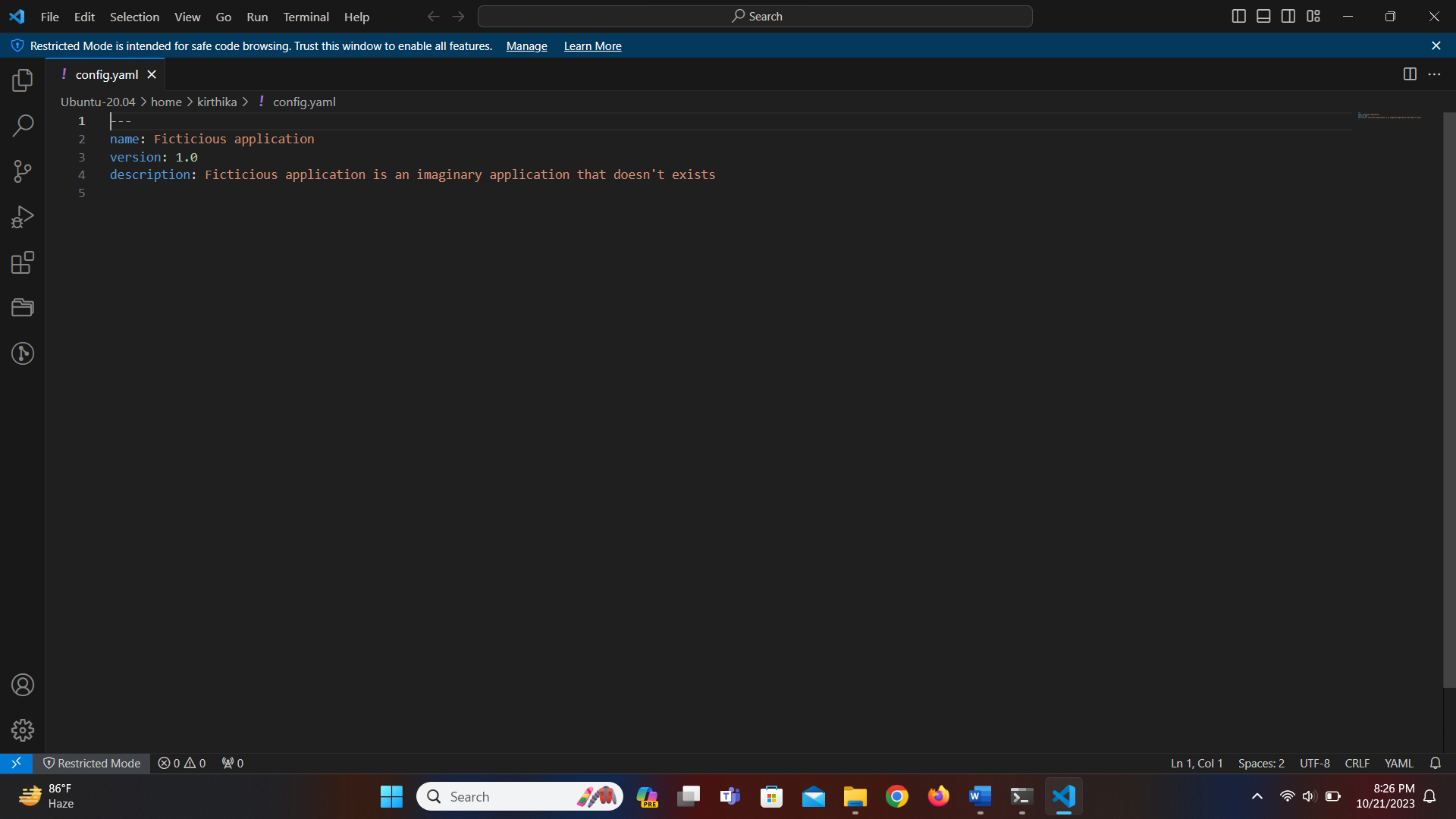
Task:

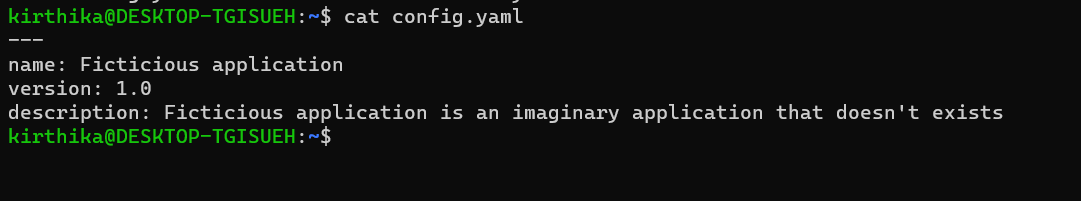
1.Create a YAML file named "config.yaml."

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.





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

Objective: Practice working with lists (arrays) in YAML.

Task:

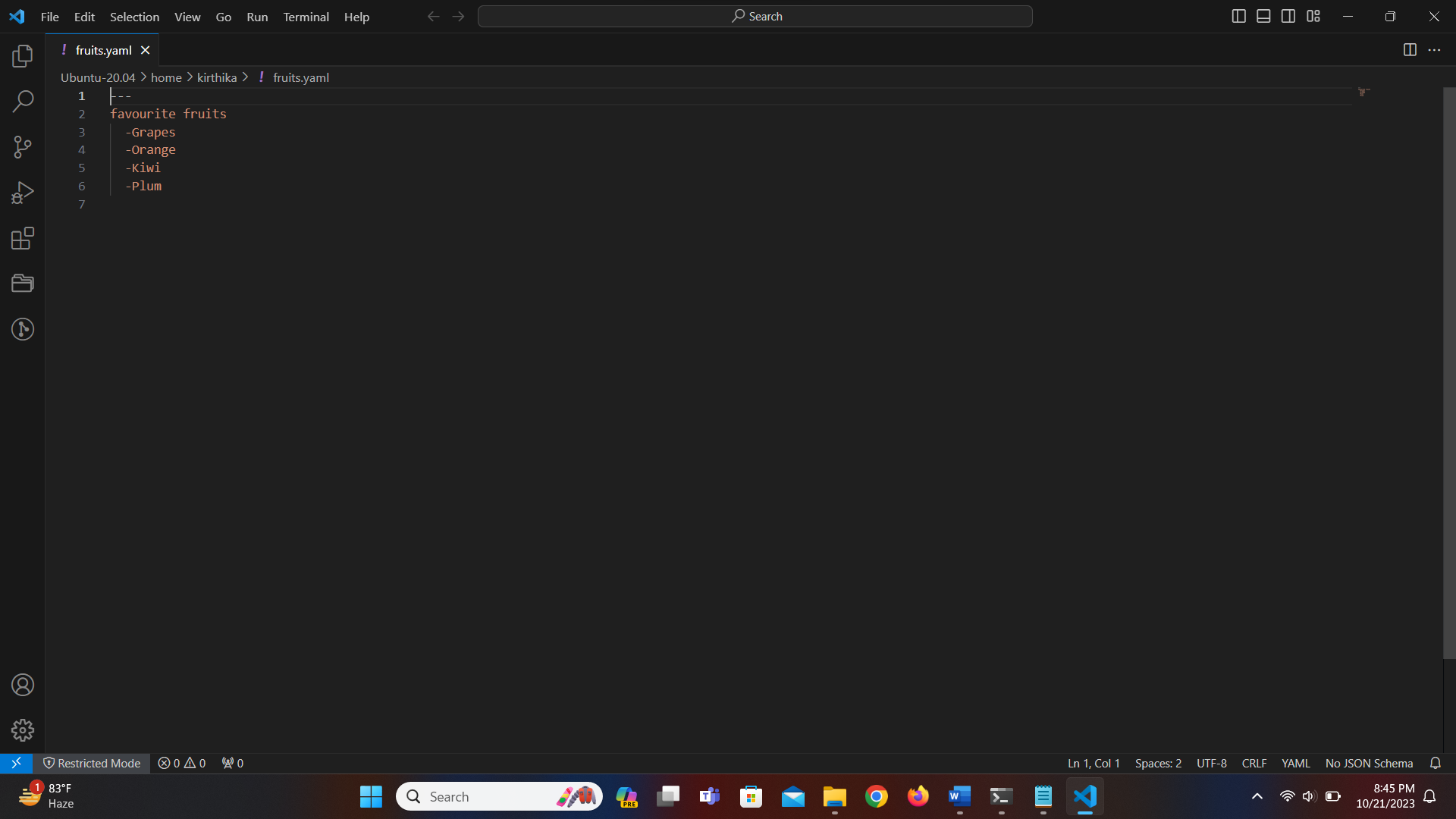
1.Create a YAML file named "fruits.yaml."

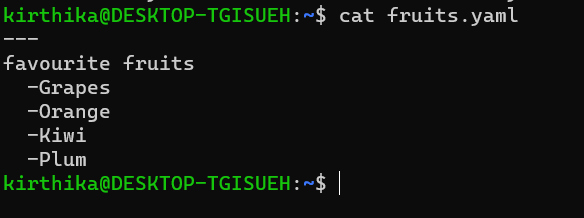
2.Define a list of your favorite fruits using YAML syntax.

3.Add items from the list.

4.Save and validate the YAML file.

Ans:





**Lab 6: Nested Structures in YAML**

Objective: Explore nested structures within YAML.

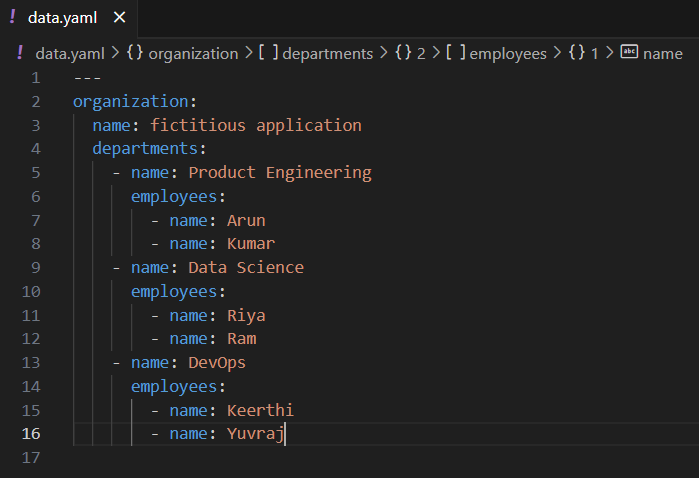
Task:

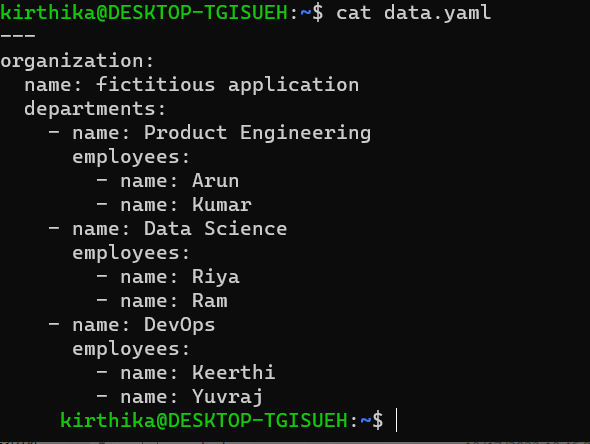
1.Create a YAML file named "data.yaml."

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.







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

Objective: Set up a classic Azure CI pipeline to build a simple Angular application with unit testing using Jasmine and Karma.

Tasks:

1.Create an Azure DevOps project.

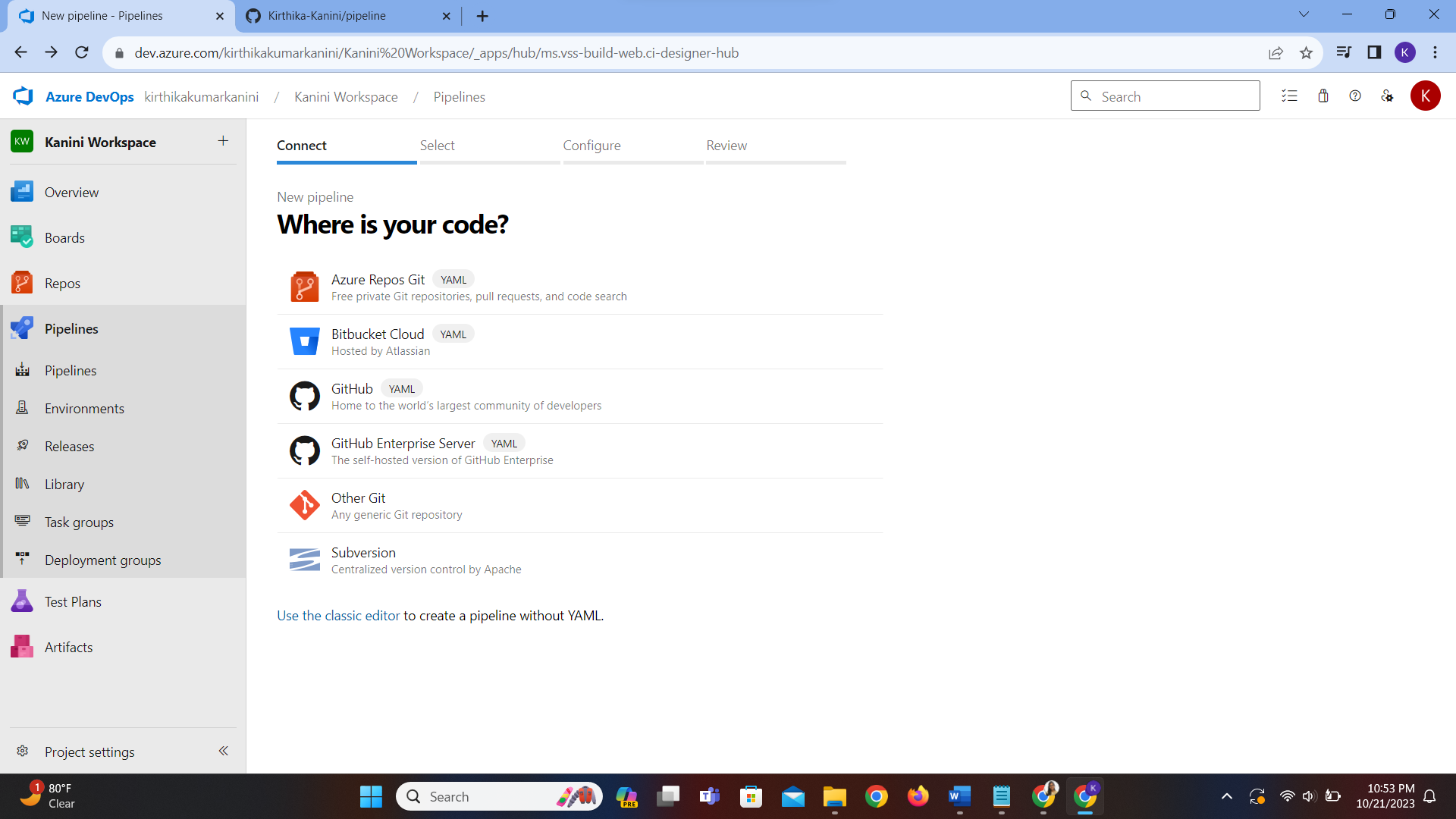
2.Set up a classic CI pipeline to build an Angular application.

3.Configure the pipeline to use Jasmine and Karma for unit testing.

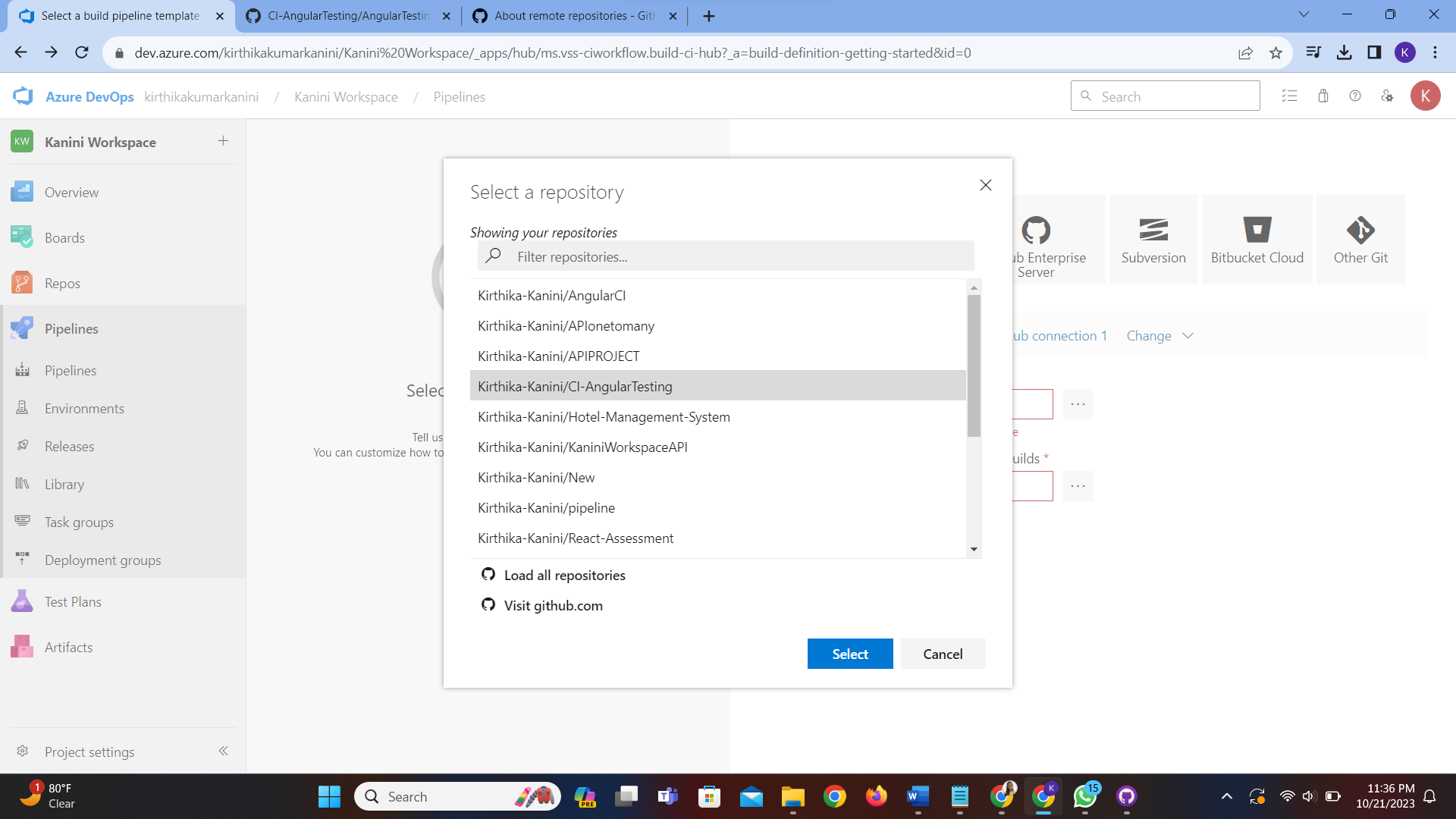
4.Run the pipeline and validate the test results.

Ans:

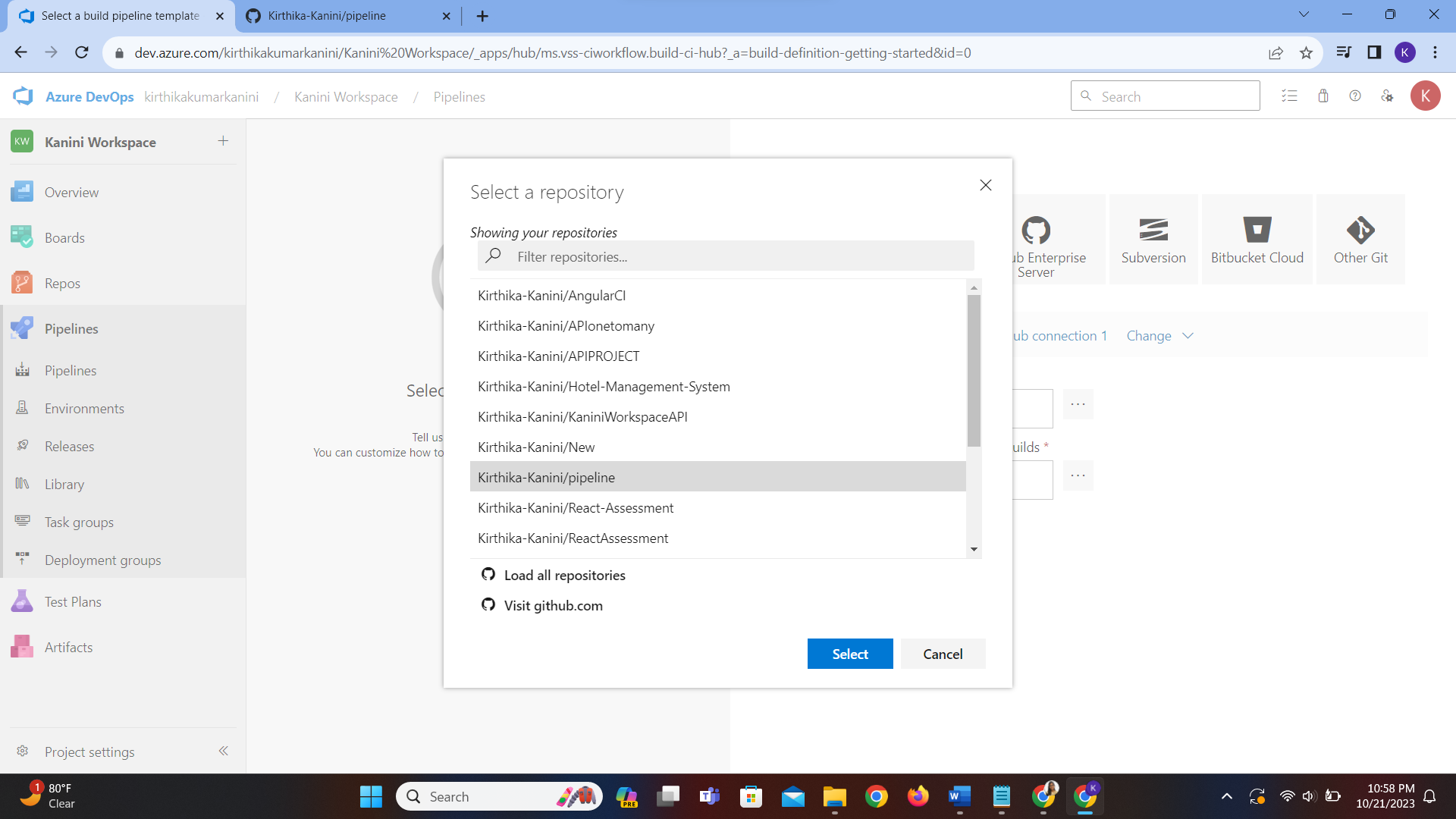
Create pipeline using classic editor:

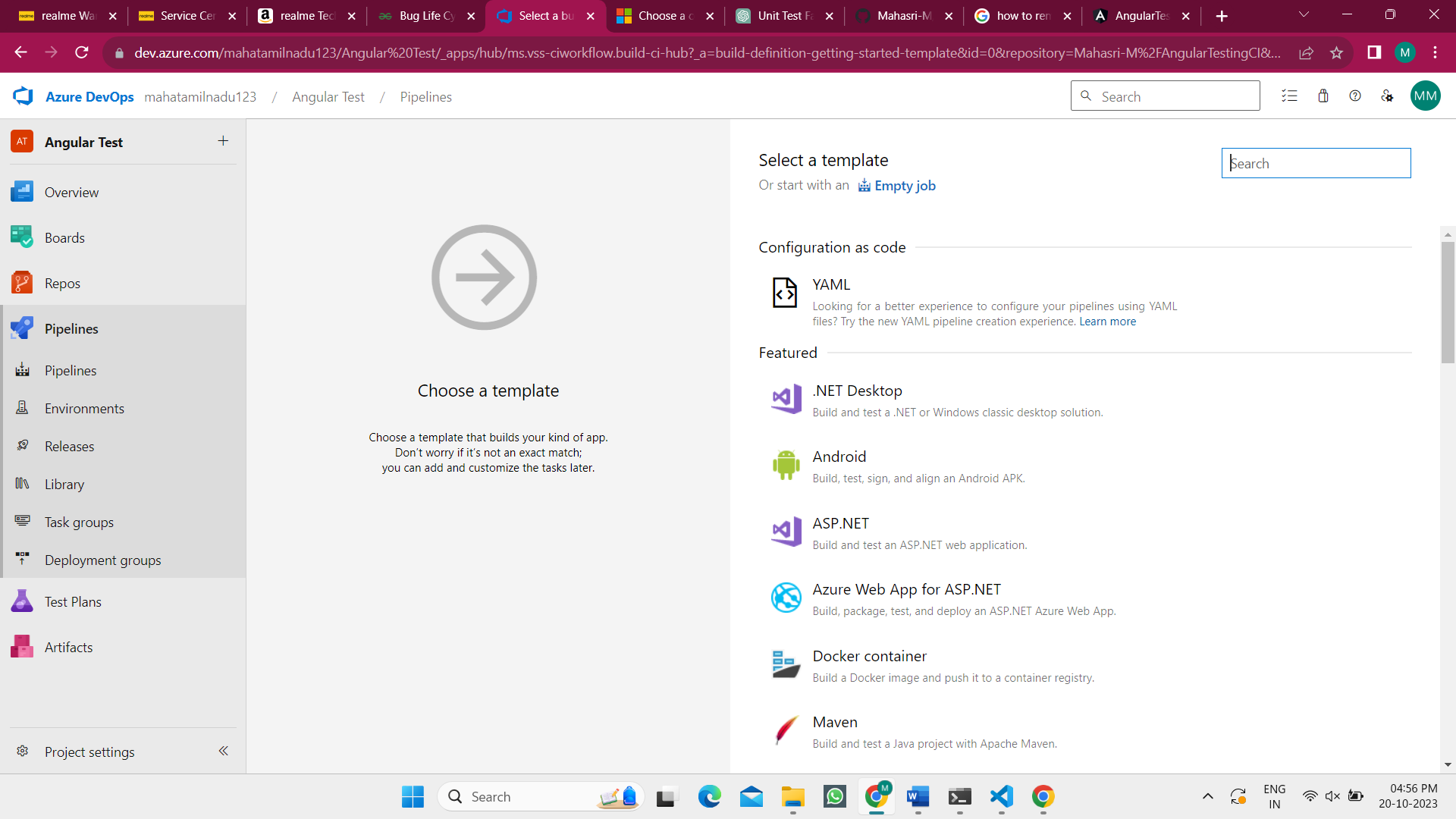


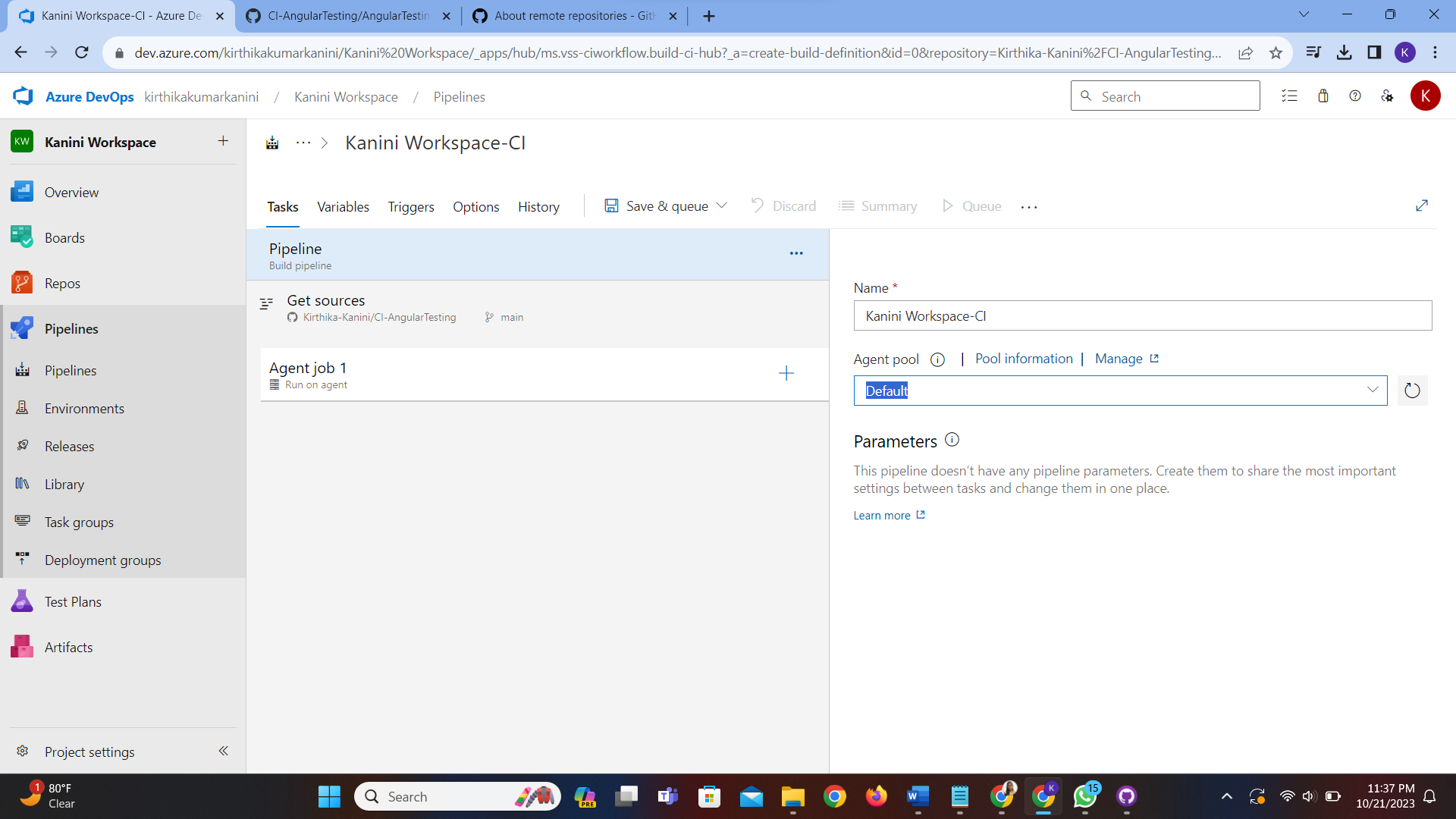
Select Github:

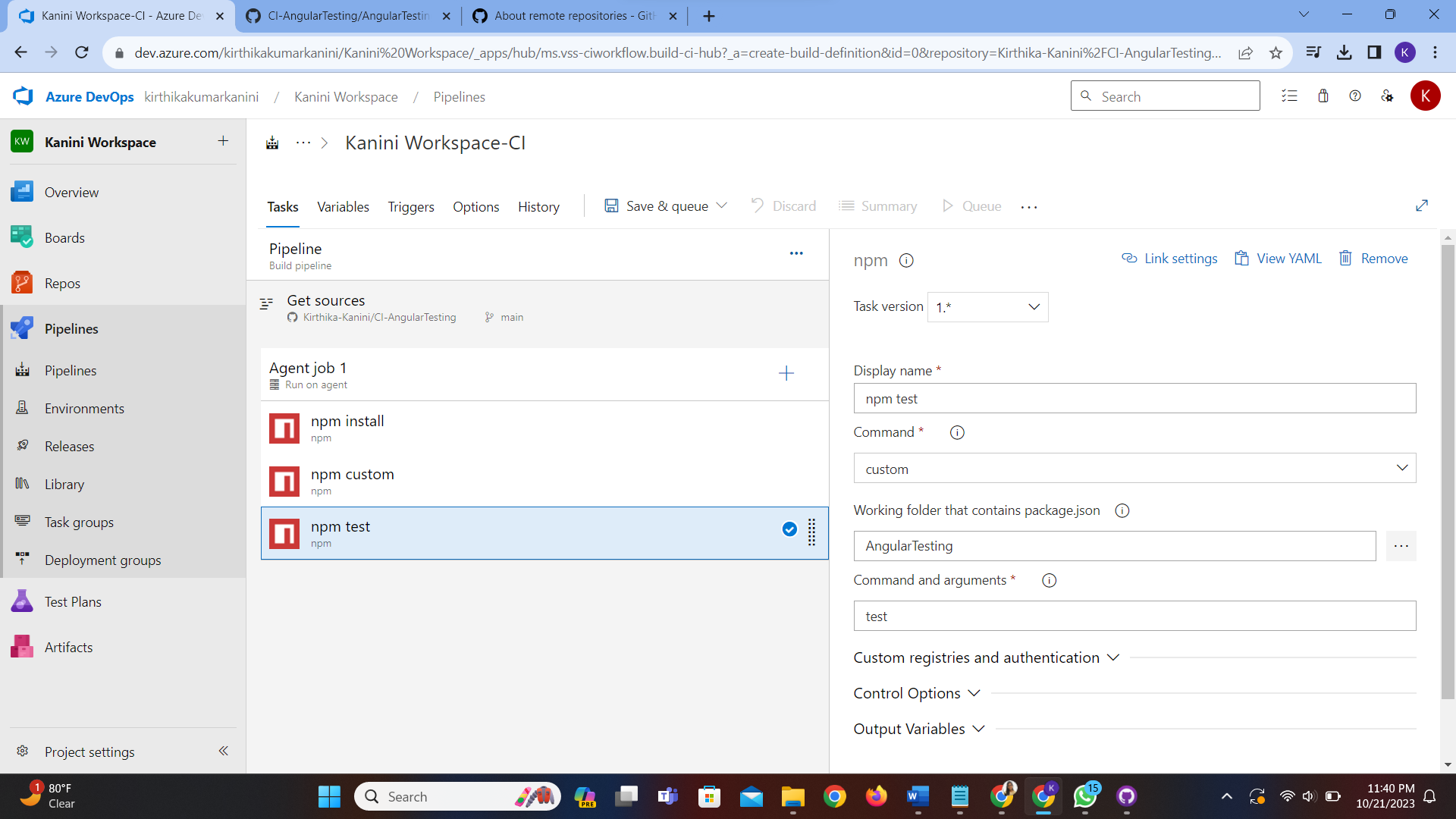
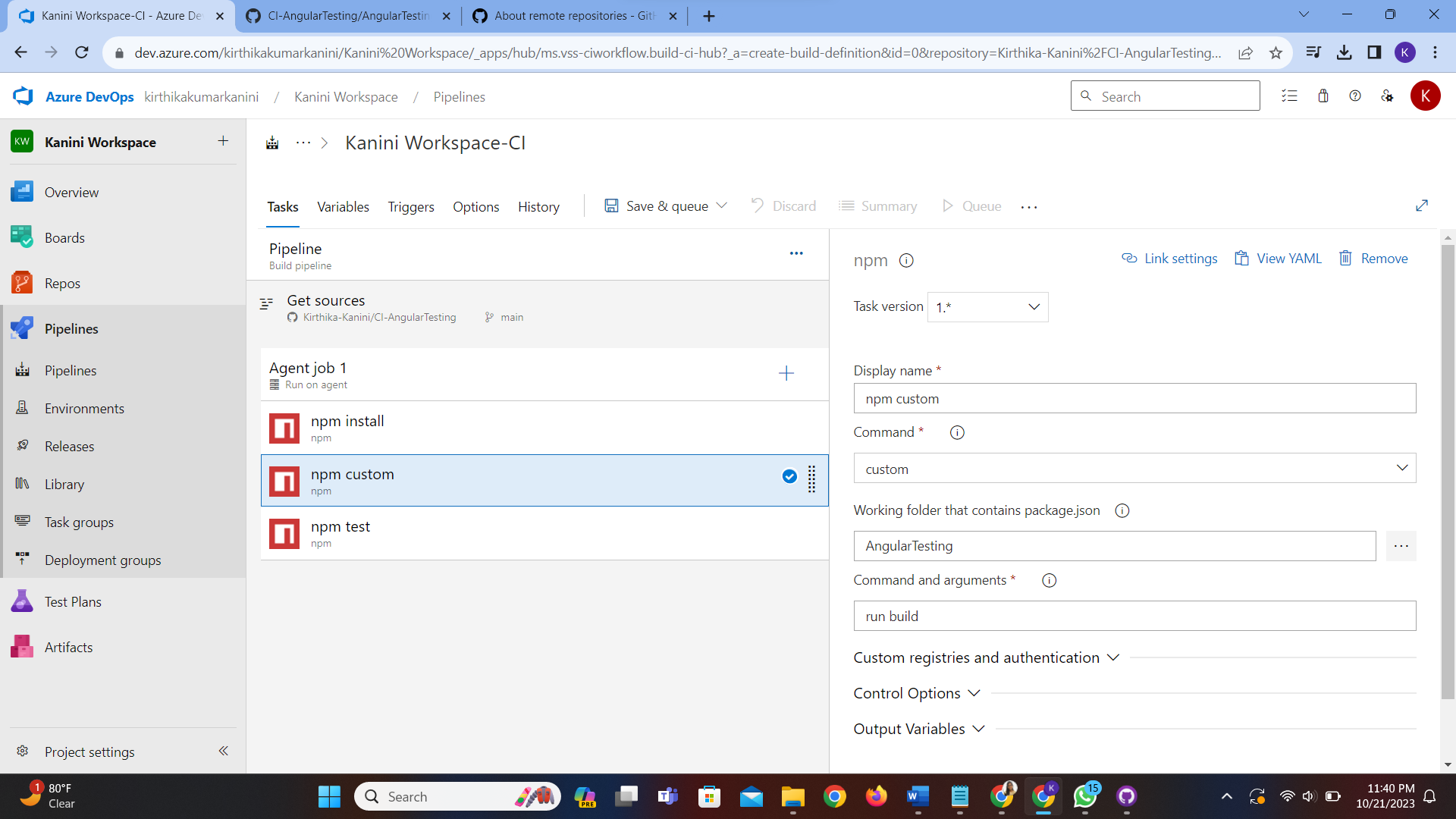
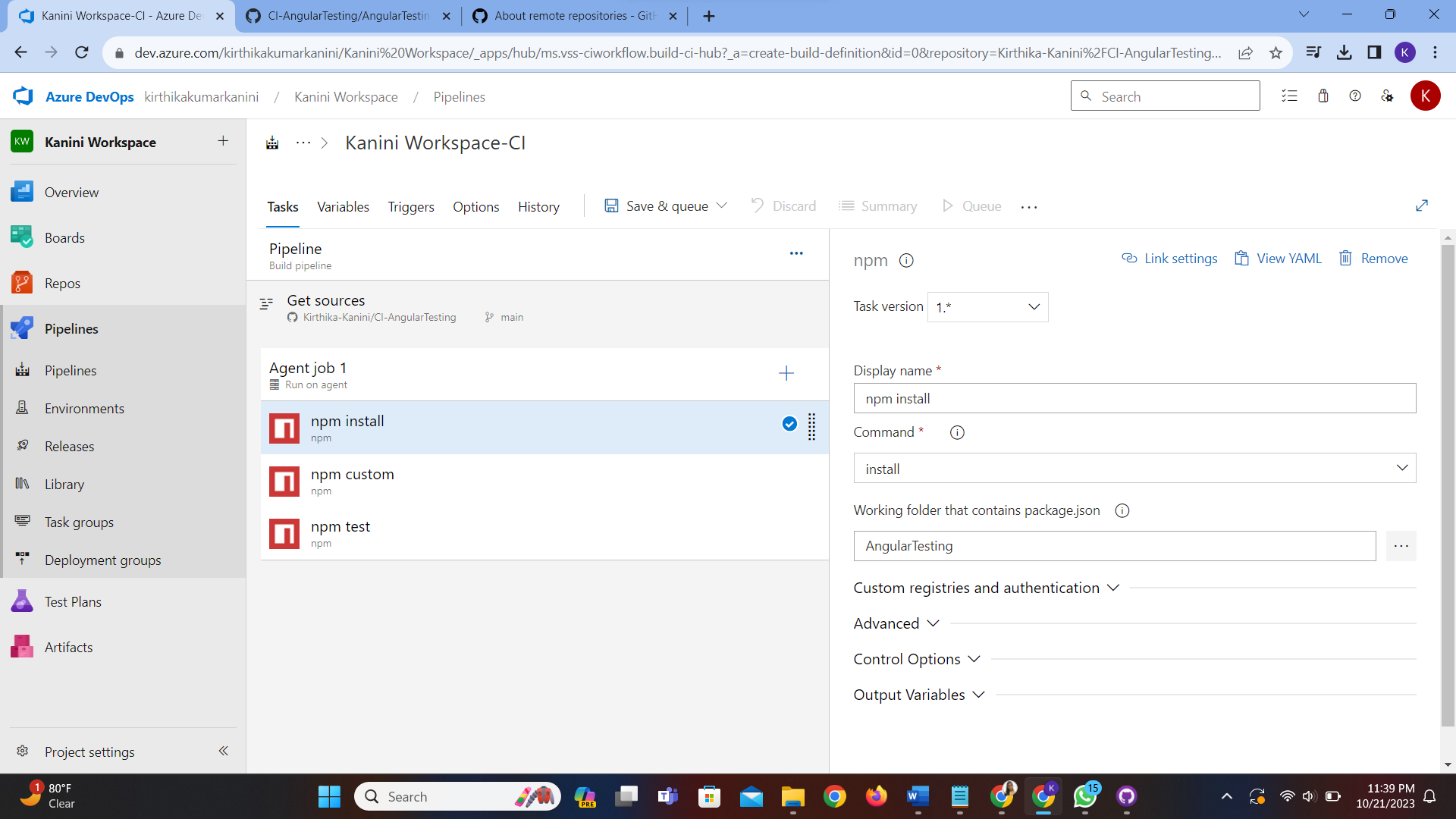


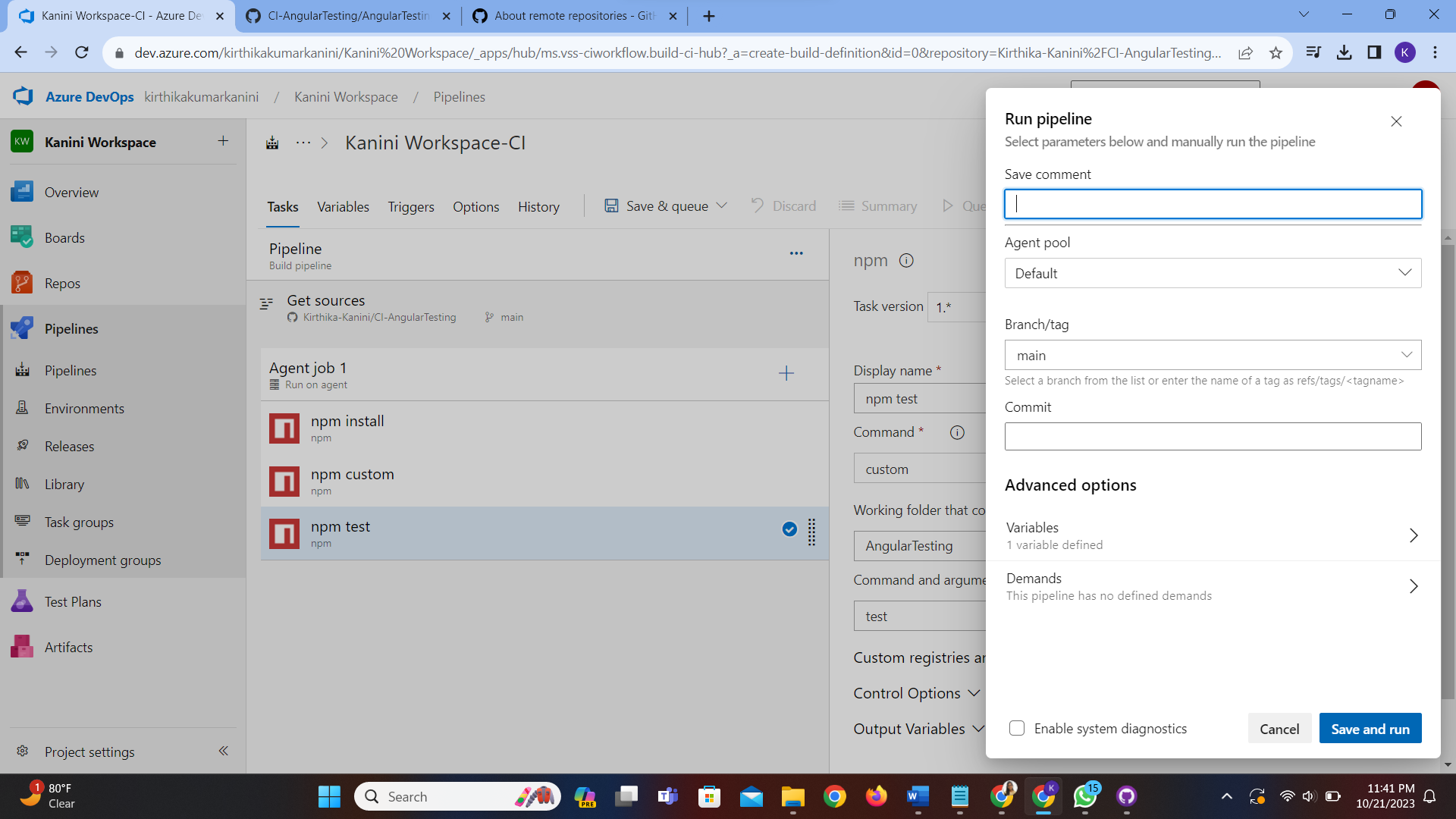
Choose a Repo:

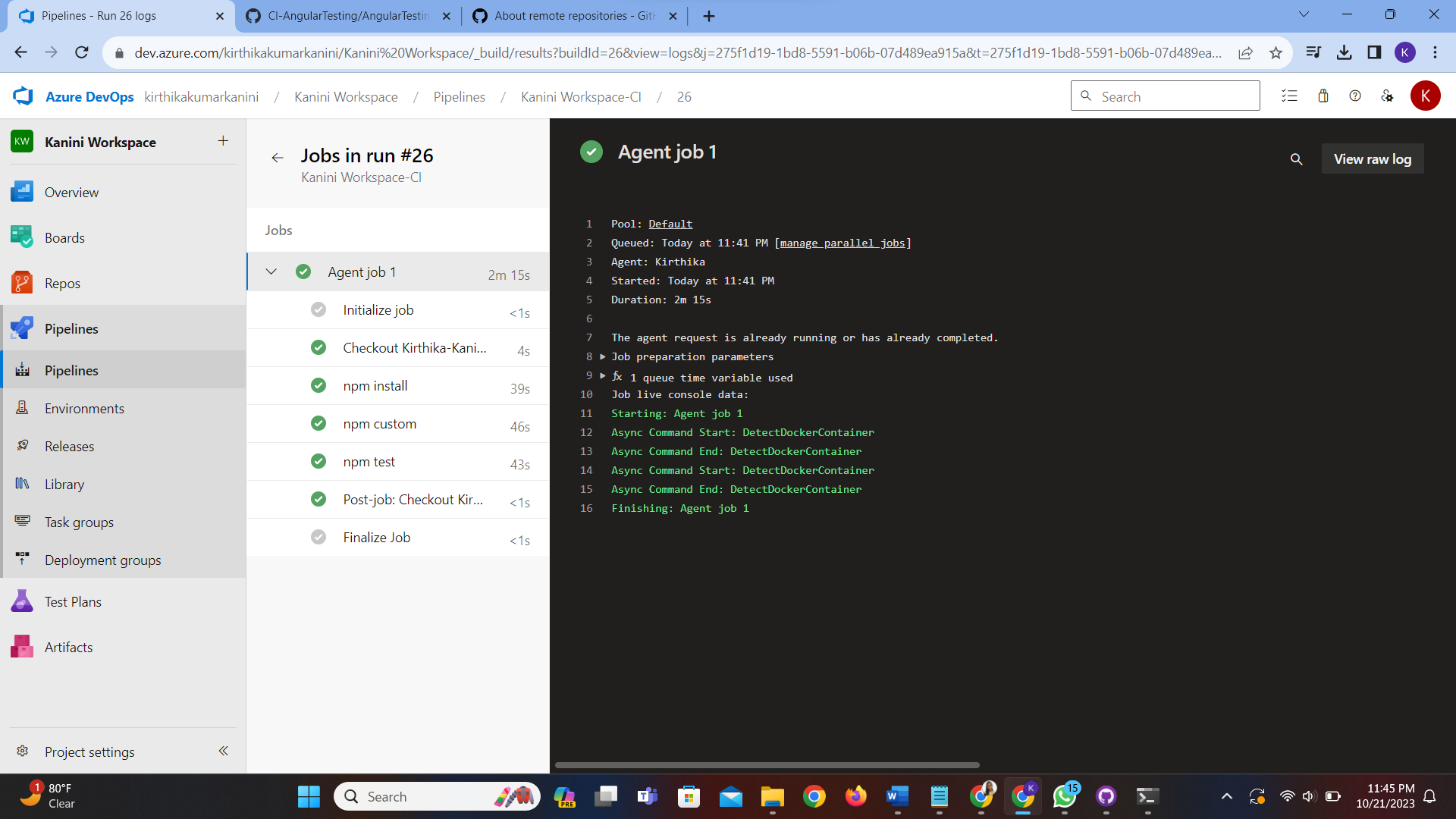


Click Empty Job:

Change to Default in agent job and pipeline







**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.

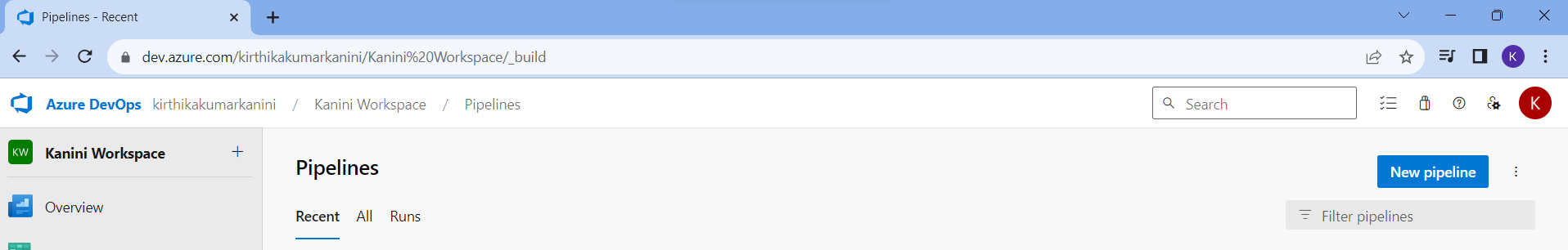
Tasks:

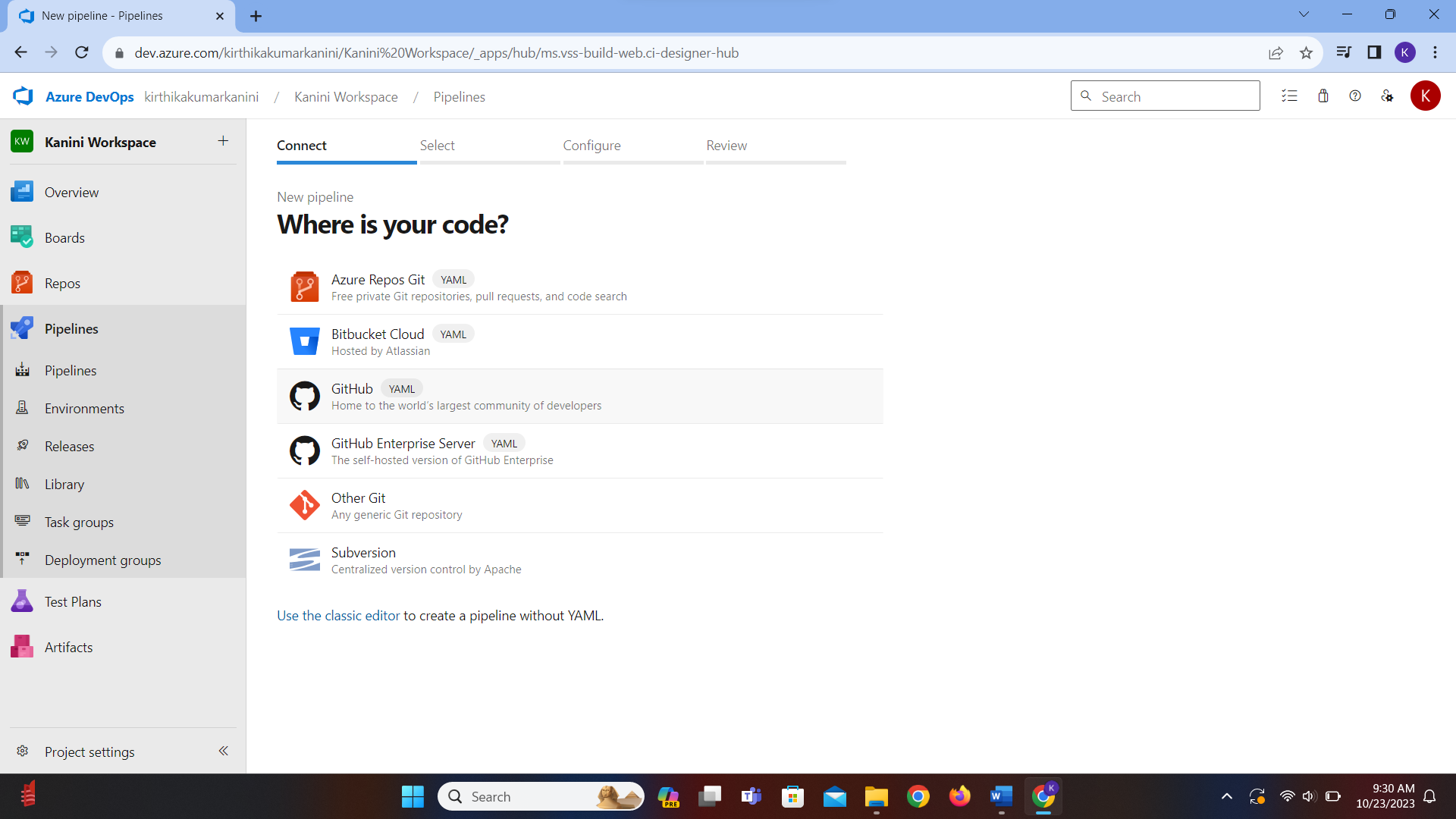
1.Create an Azure DevOps project.

2.Create a YAML-based CI pipeline to build a React application.

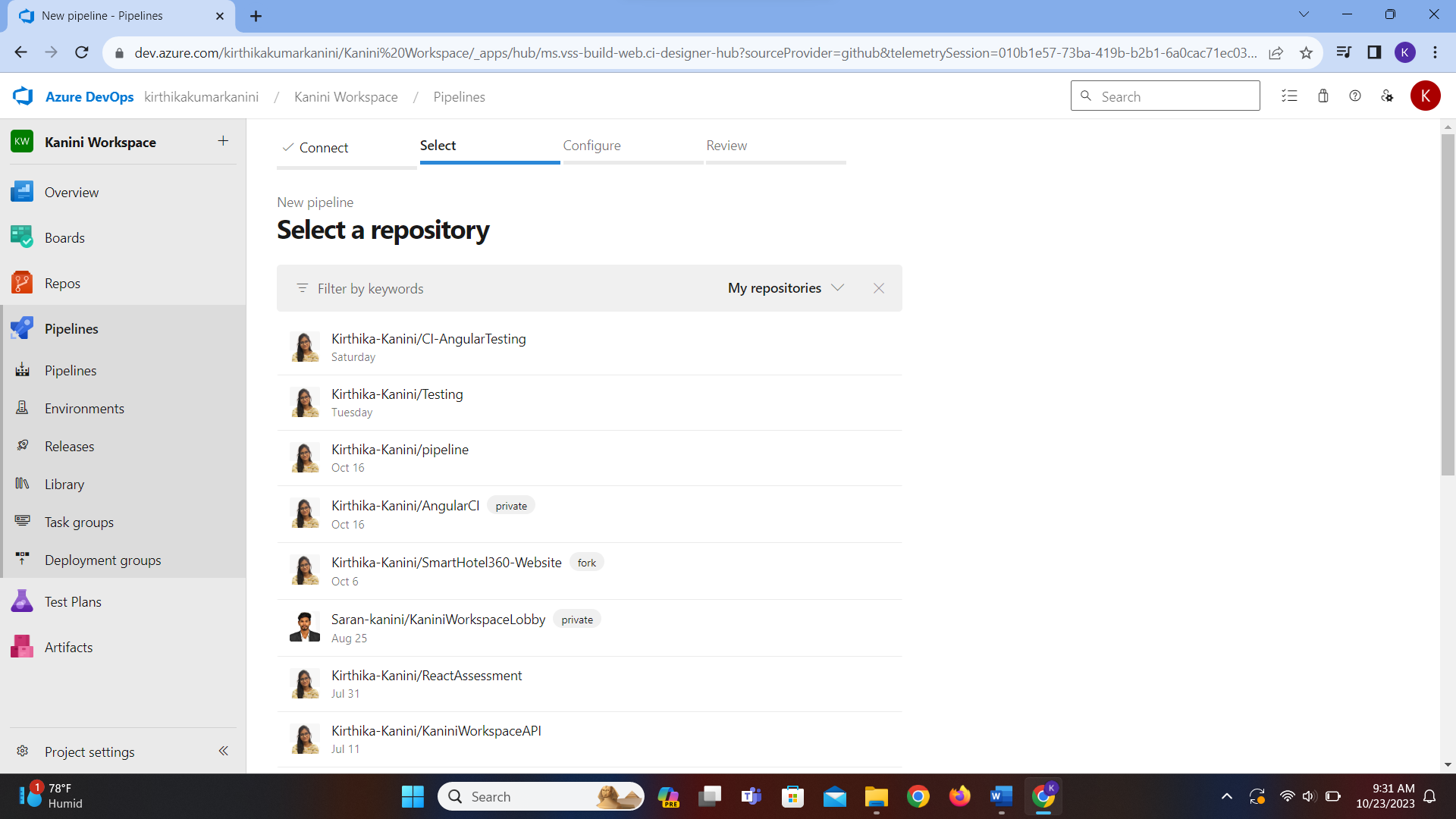
3.Configure the pipeline to use Enzyme and Jest for unit testing.

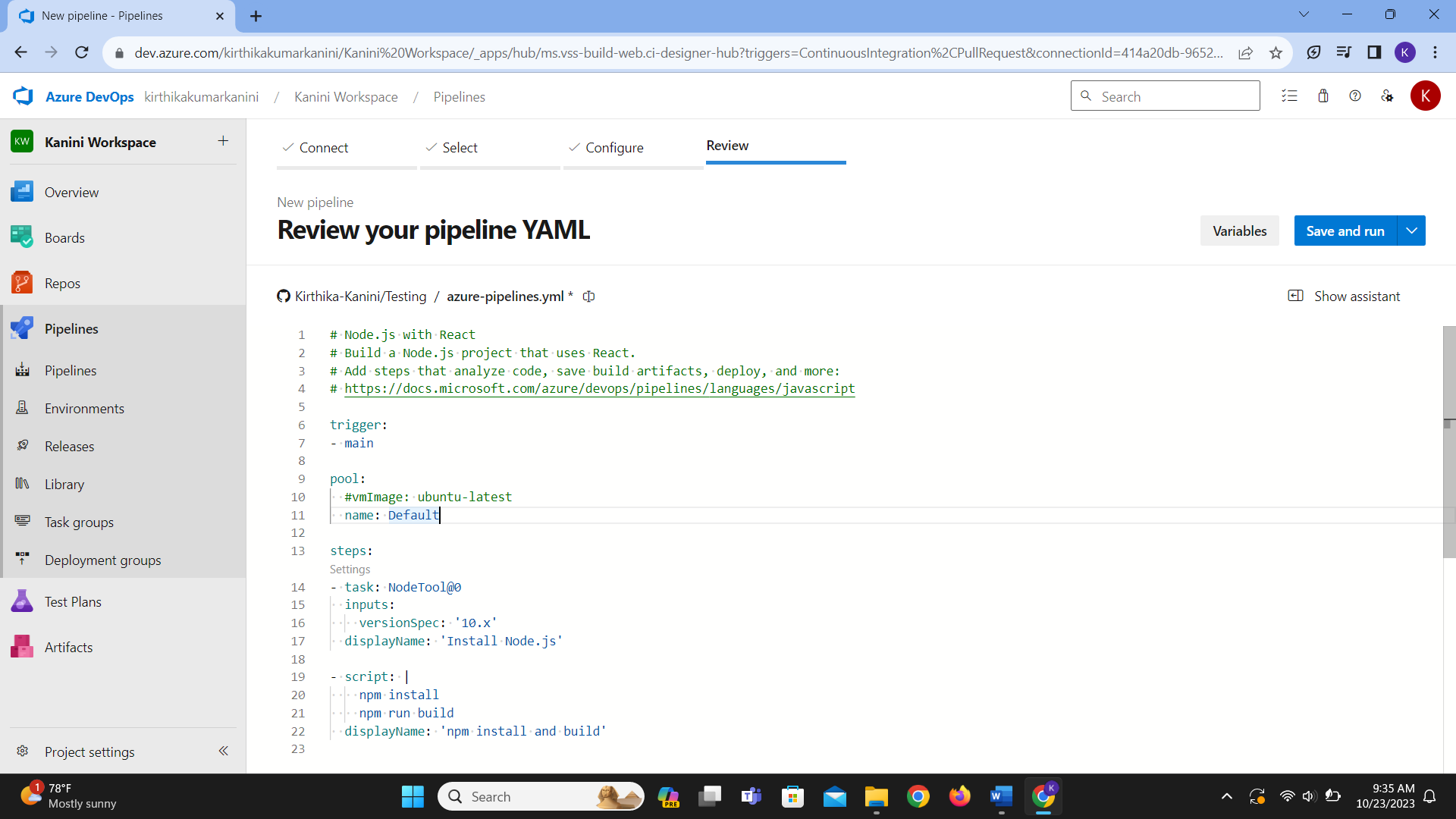
4.Trigger the pipeline and verify the test results.

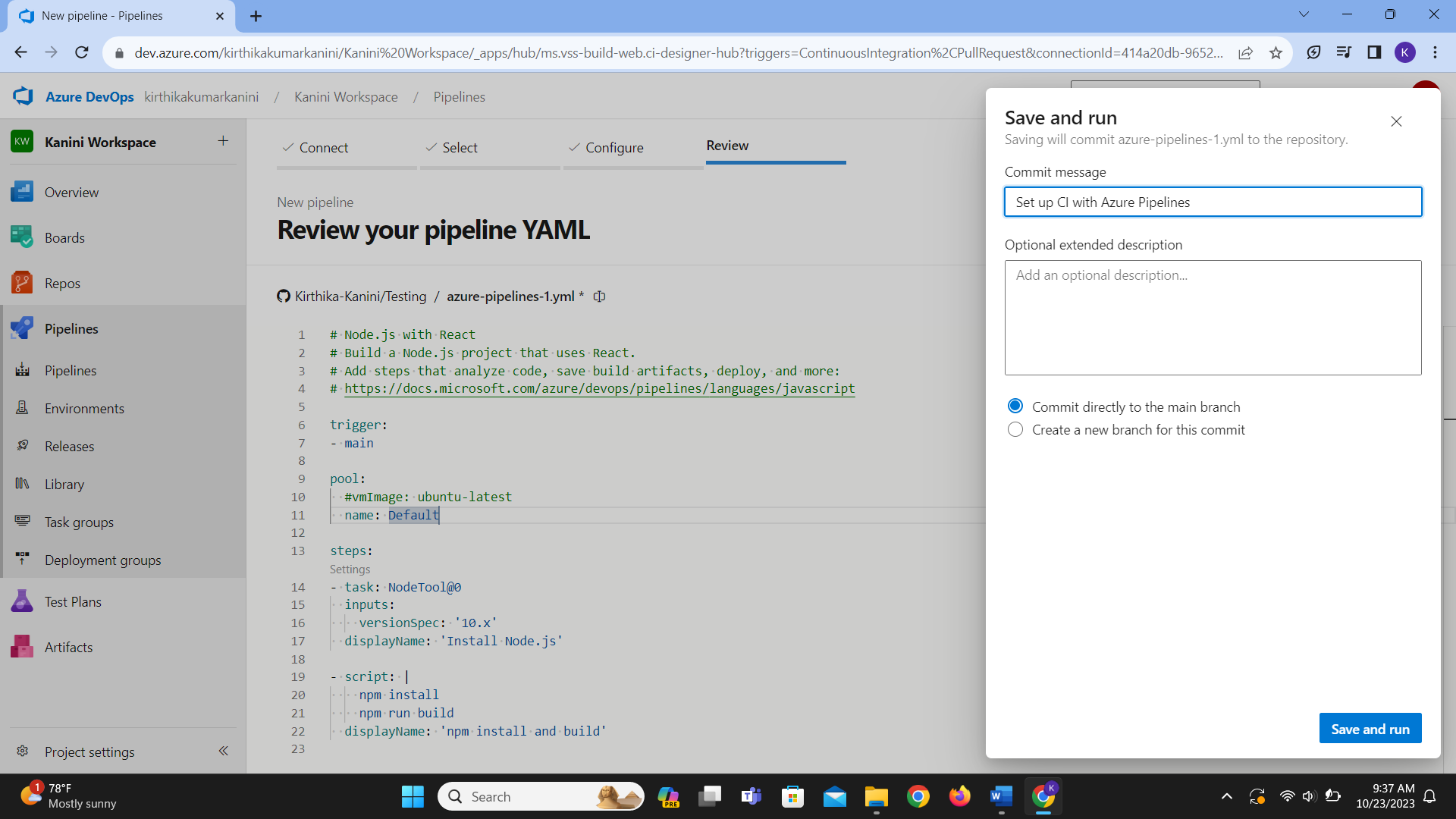
Create New Pipeline  


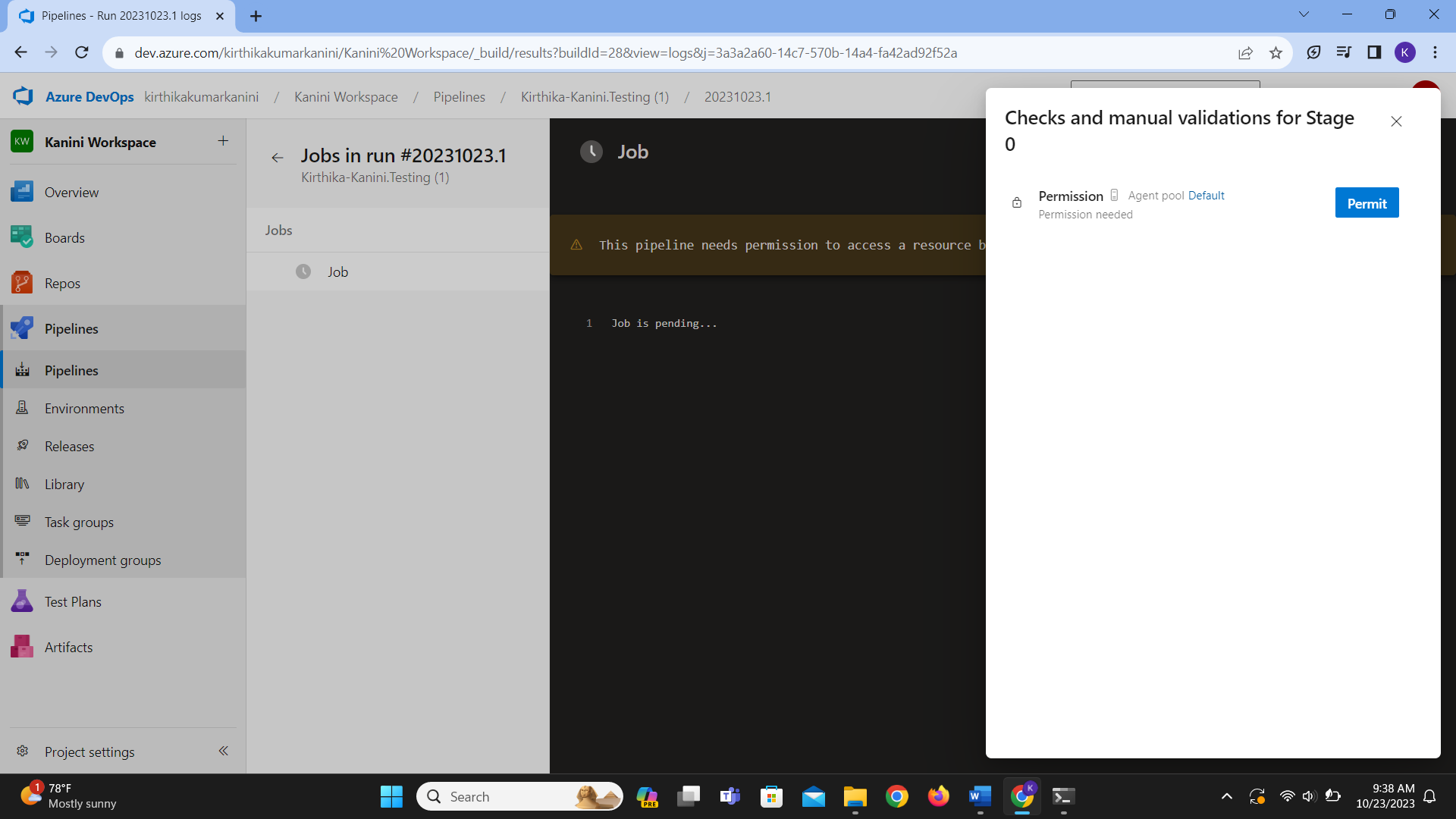
Choose Github  


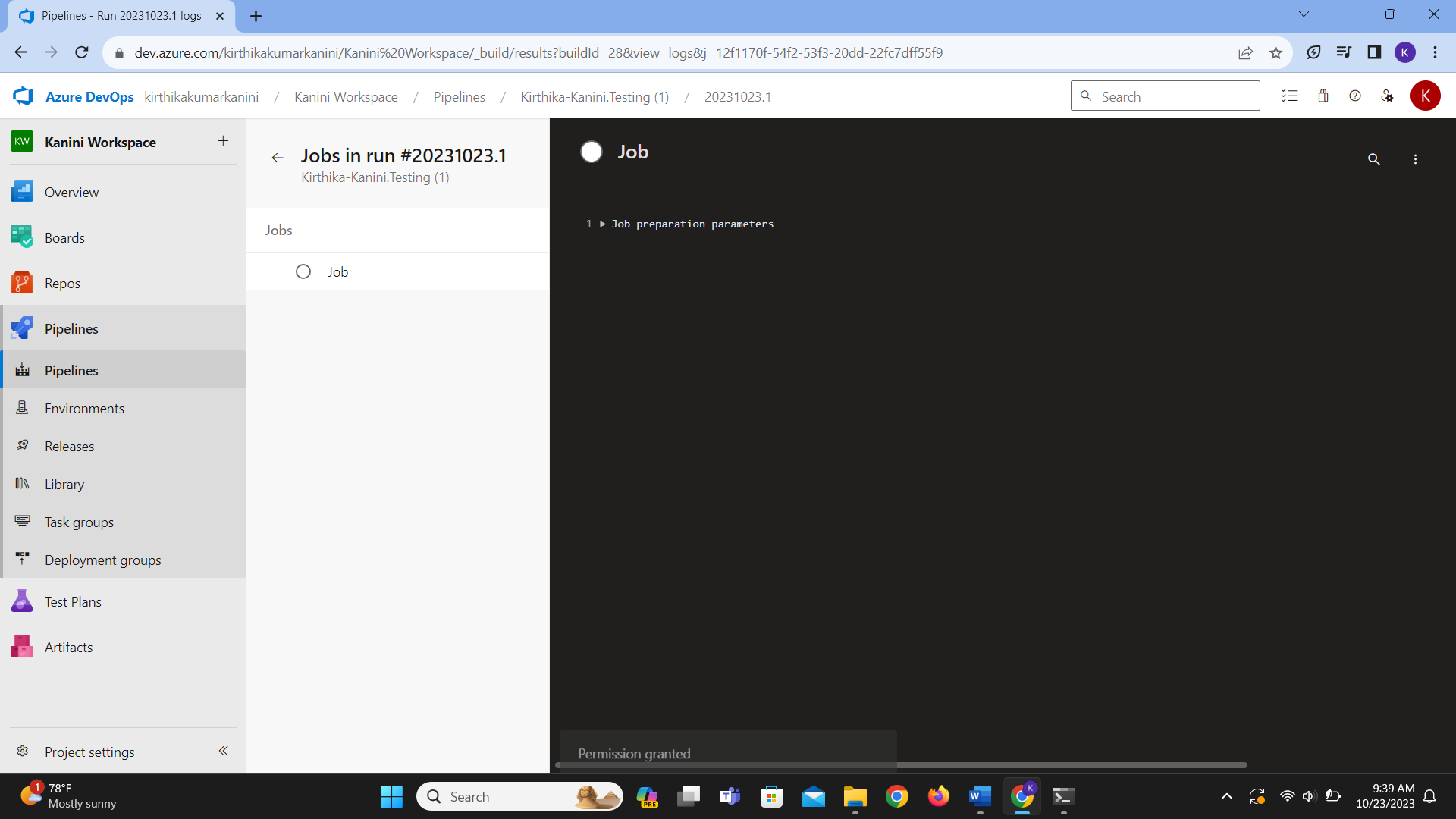
Choose a repo



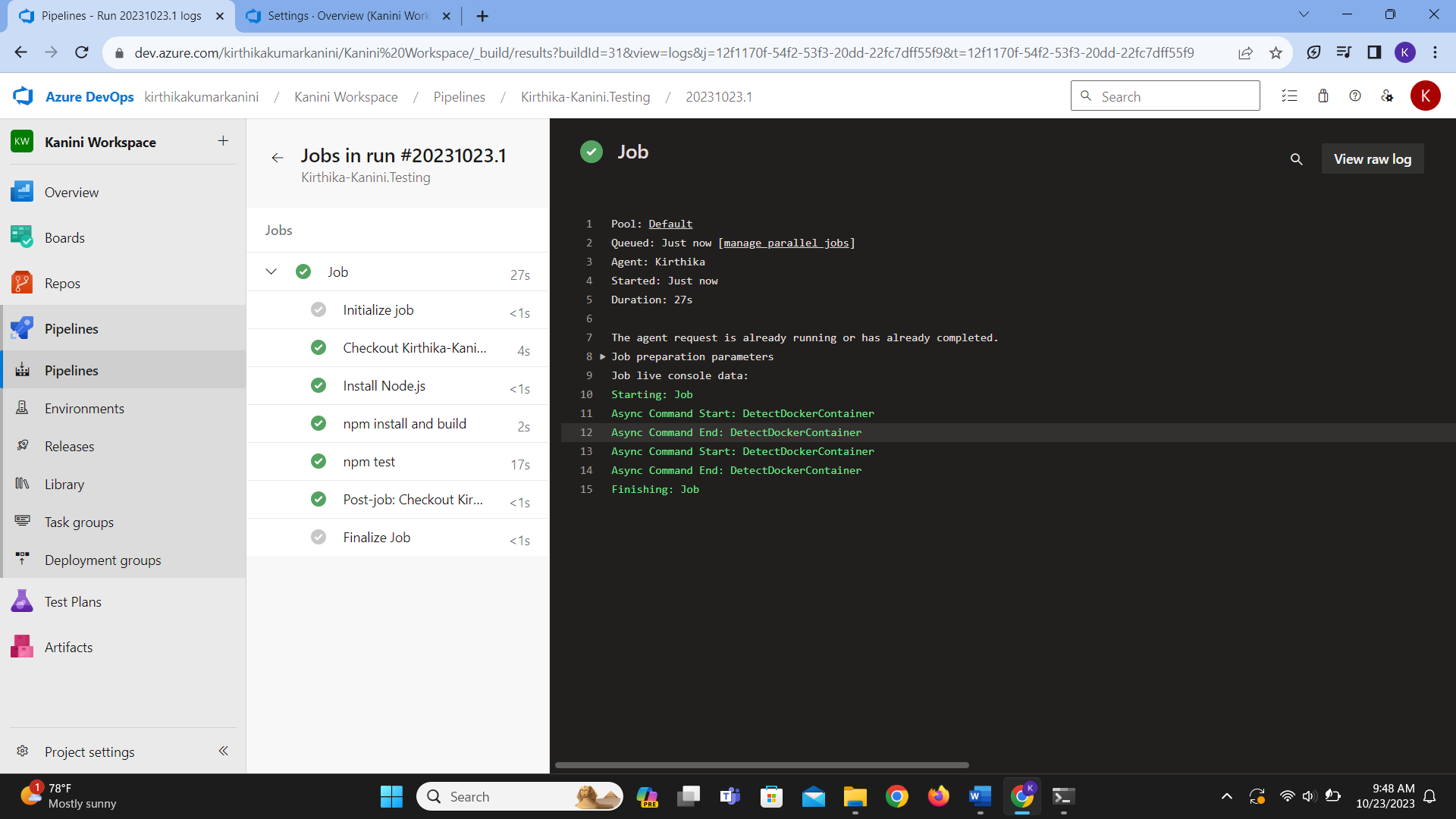
Choose Nodejs with React  


Click Save and Run  


Give permission  




Finishing Job



**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:

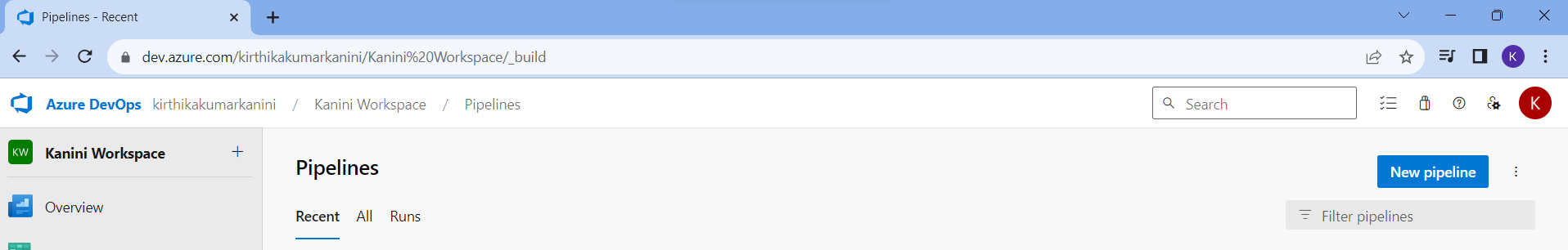
1.Set up a new Azure DevOps project.

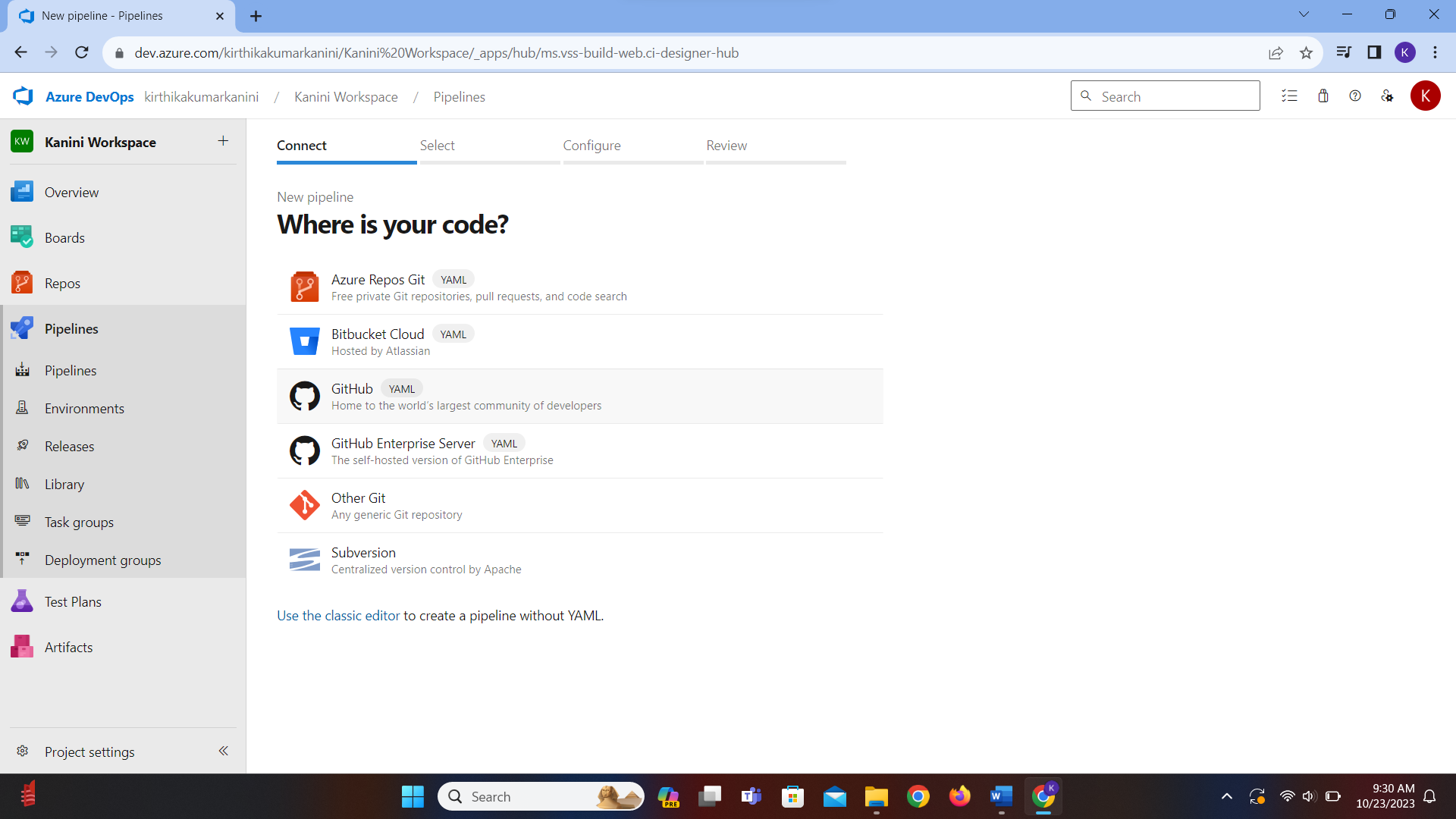
2.Create a CI/CD pipeline for a .NET Core application.

3.Configure the pipeline to use MS Unit tests.

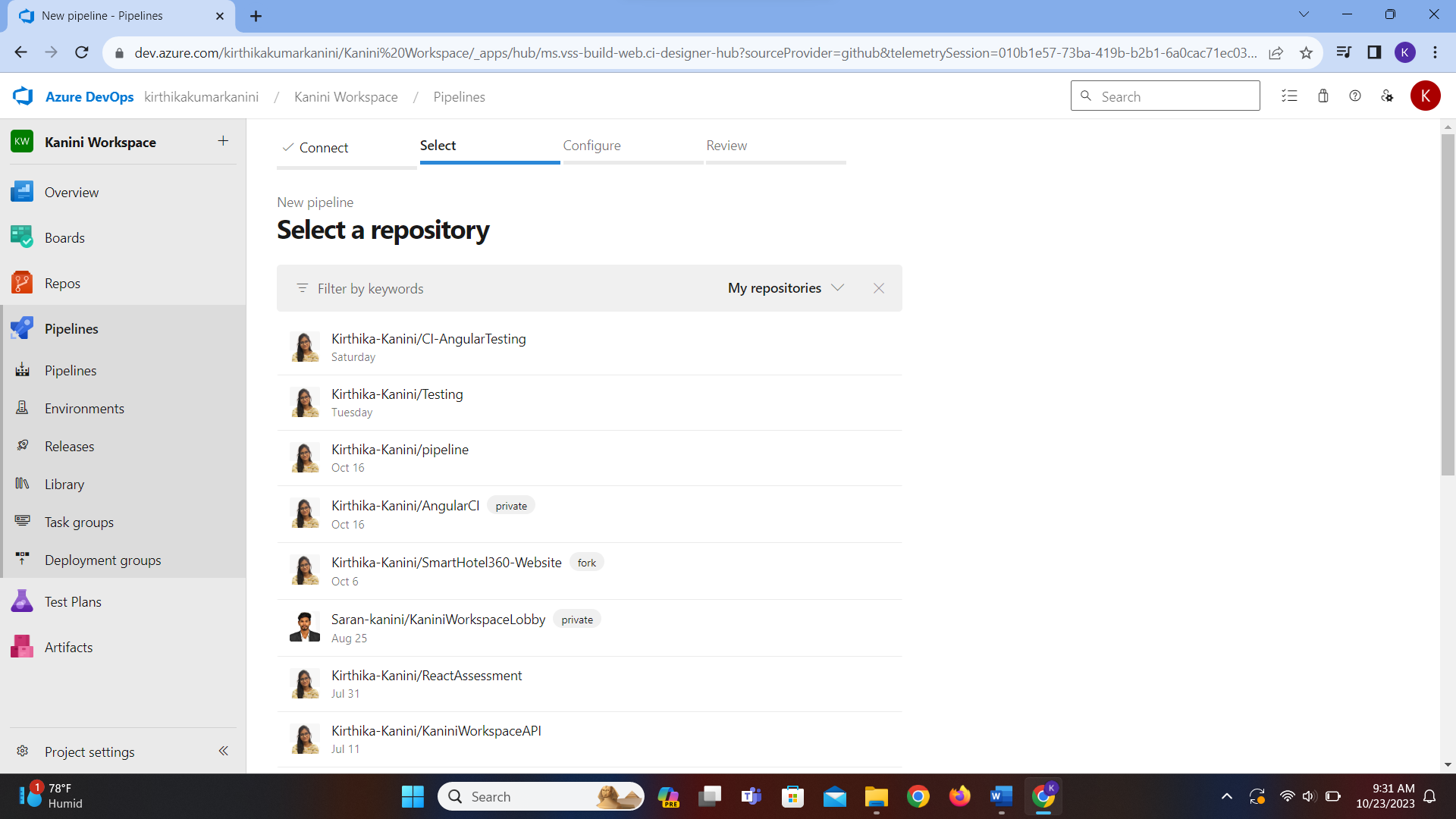
4.Trigger the pipeline and validate the test results.

Ans:

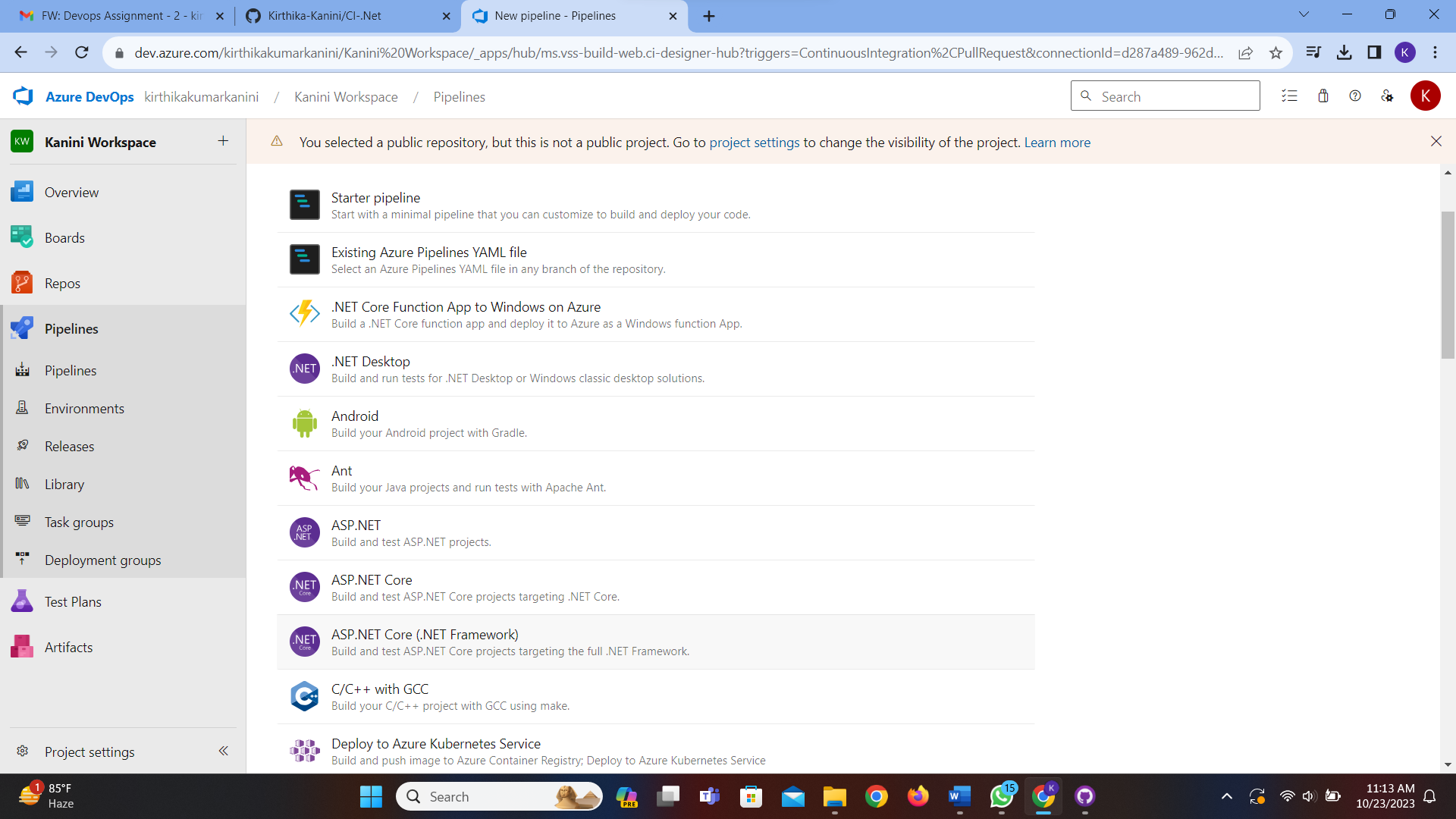
Create New Pipeline  


Choose Github  


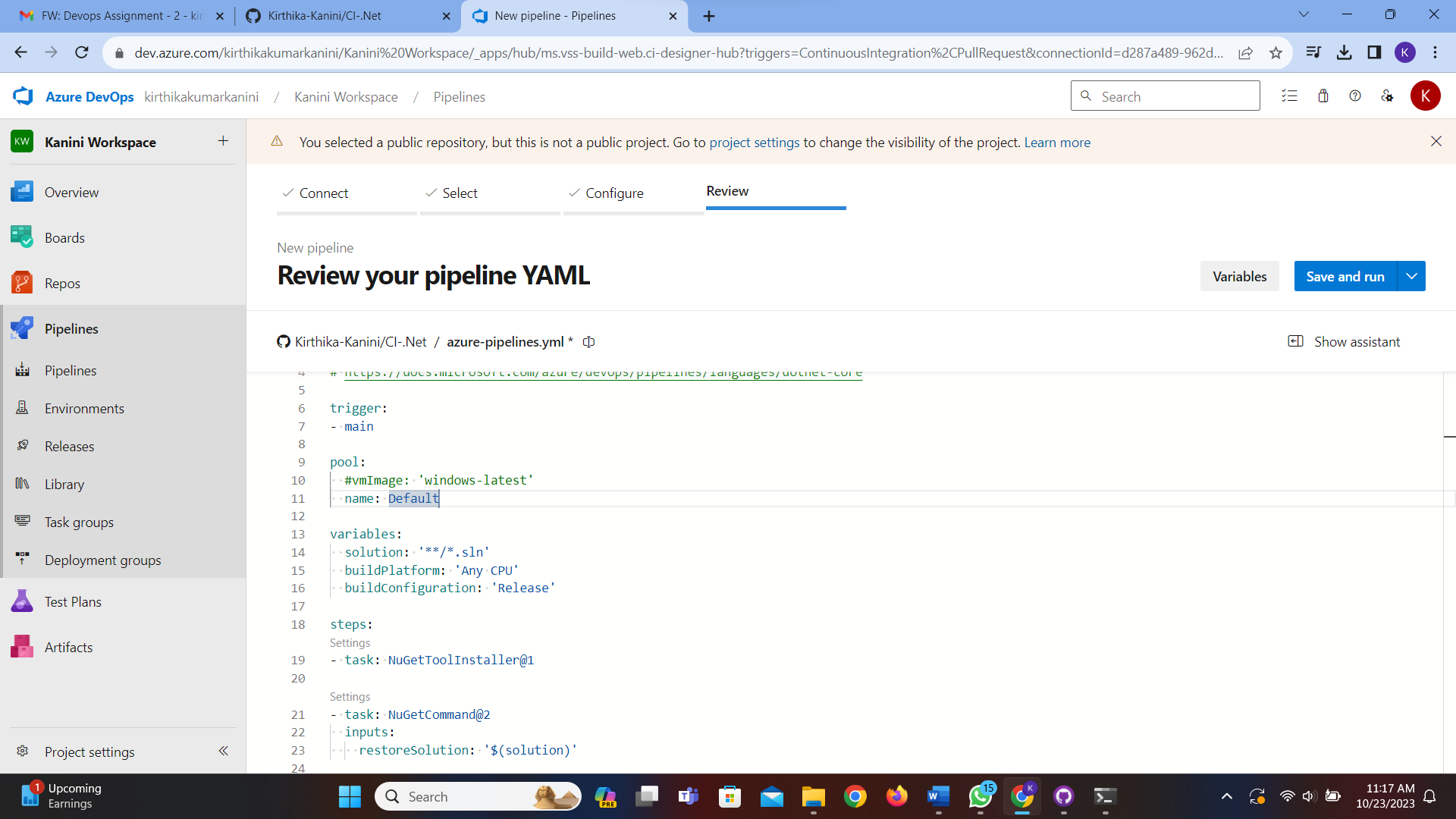
Choose a repo

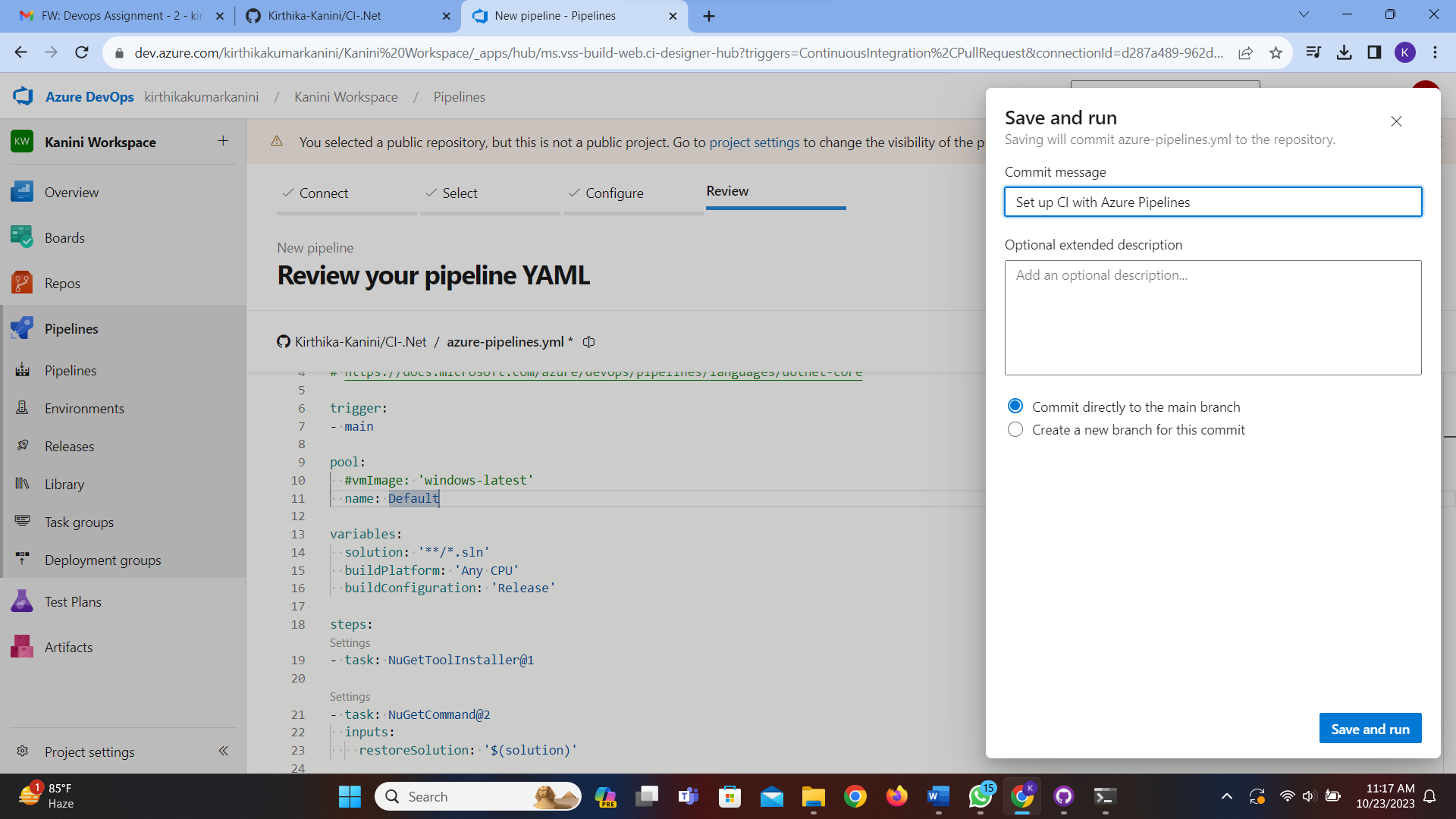


Choose ASP.NET Core (.NET framework):

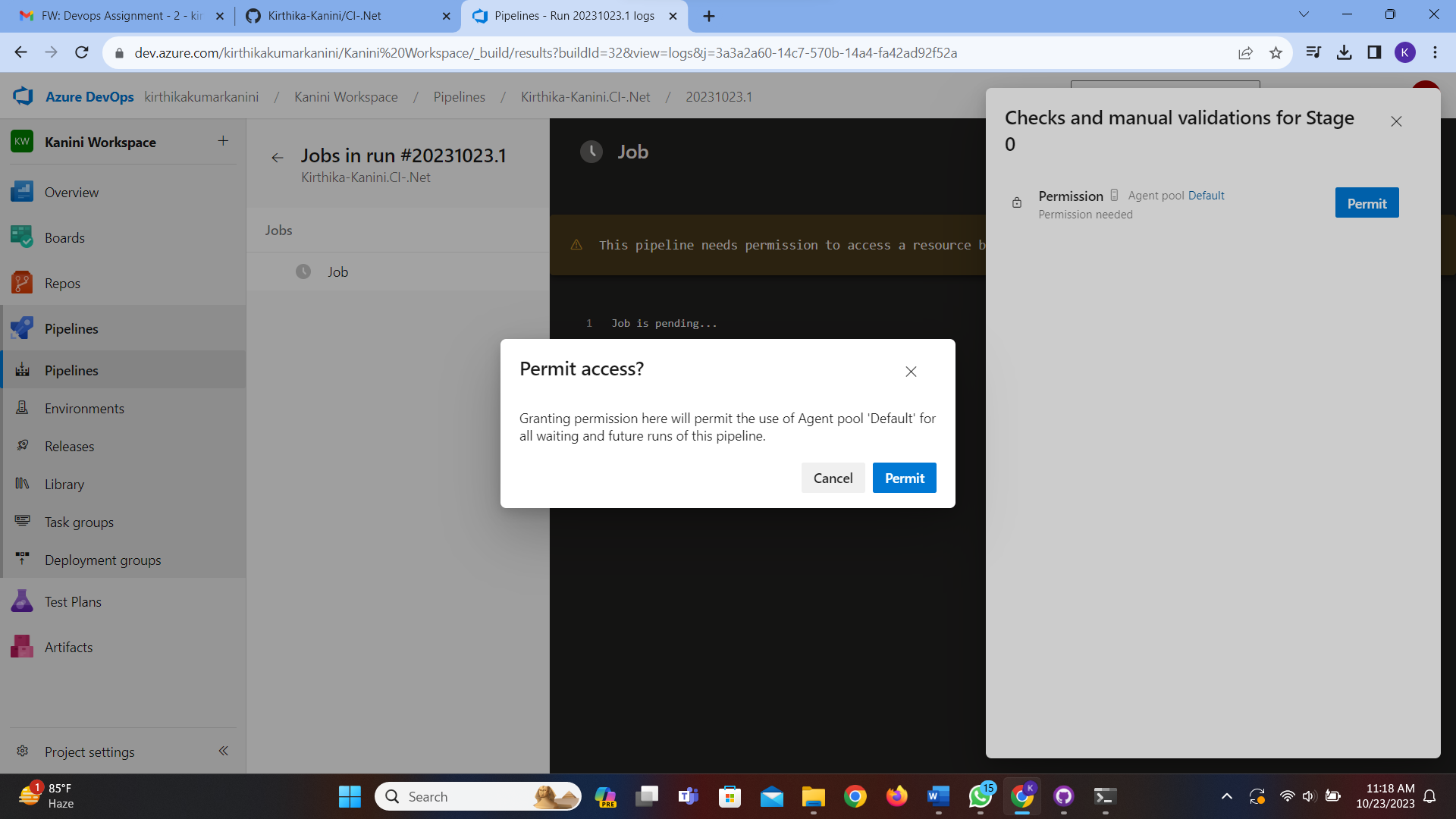


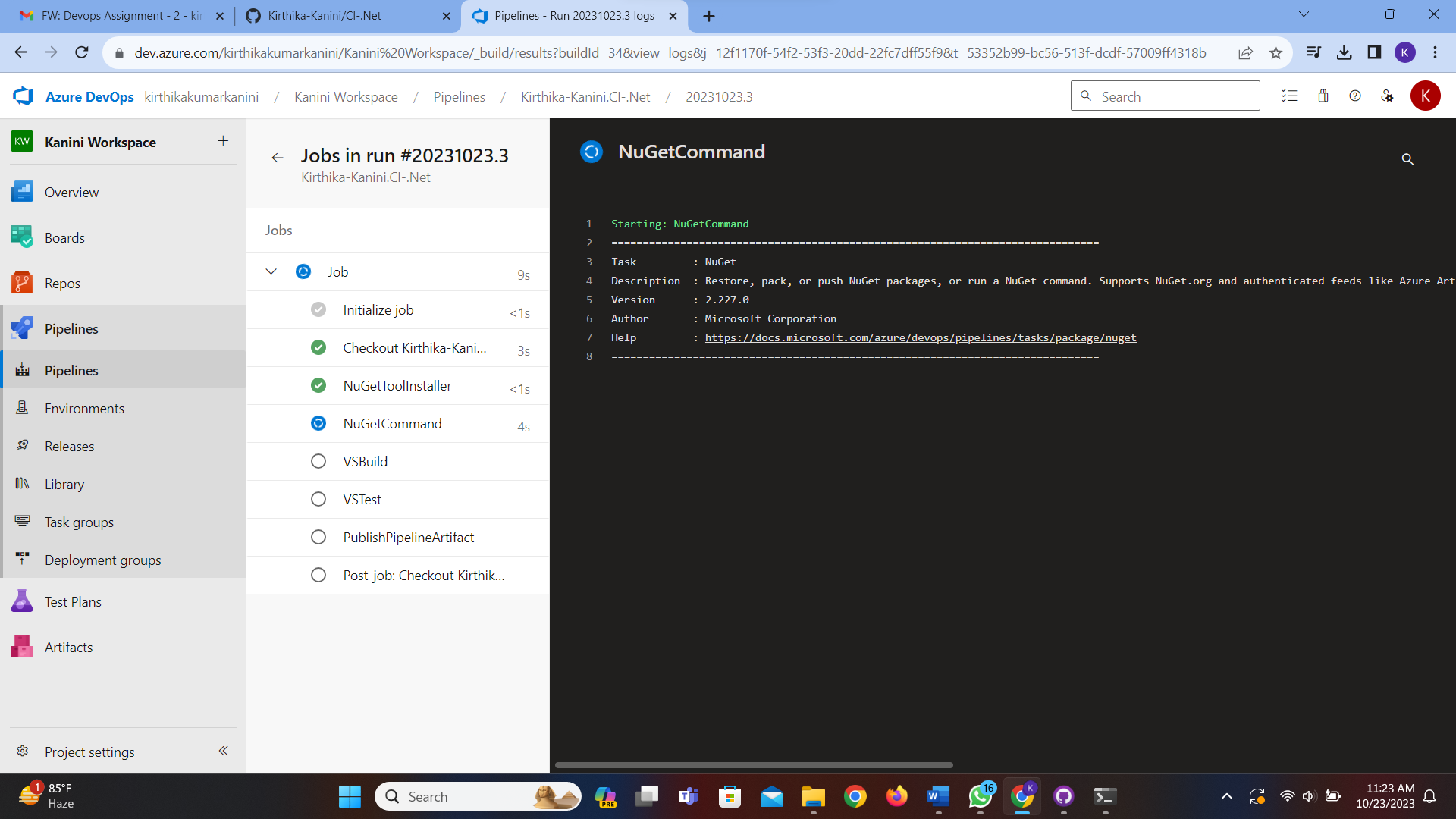
Edit Yaml:



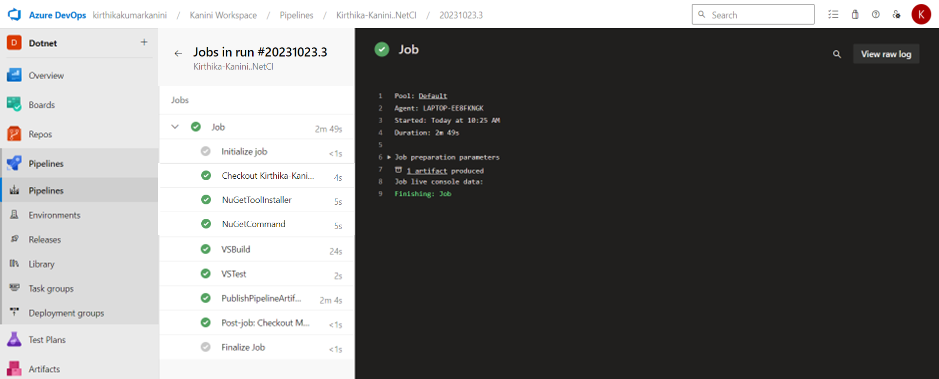
Save and Run:  


Give Permission:

****

Pipelines starts to run  


The Pipeline Ran Successfully:



**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

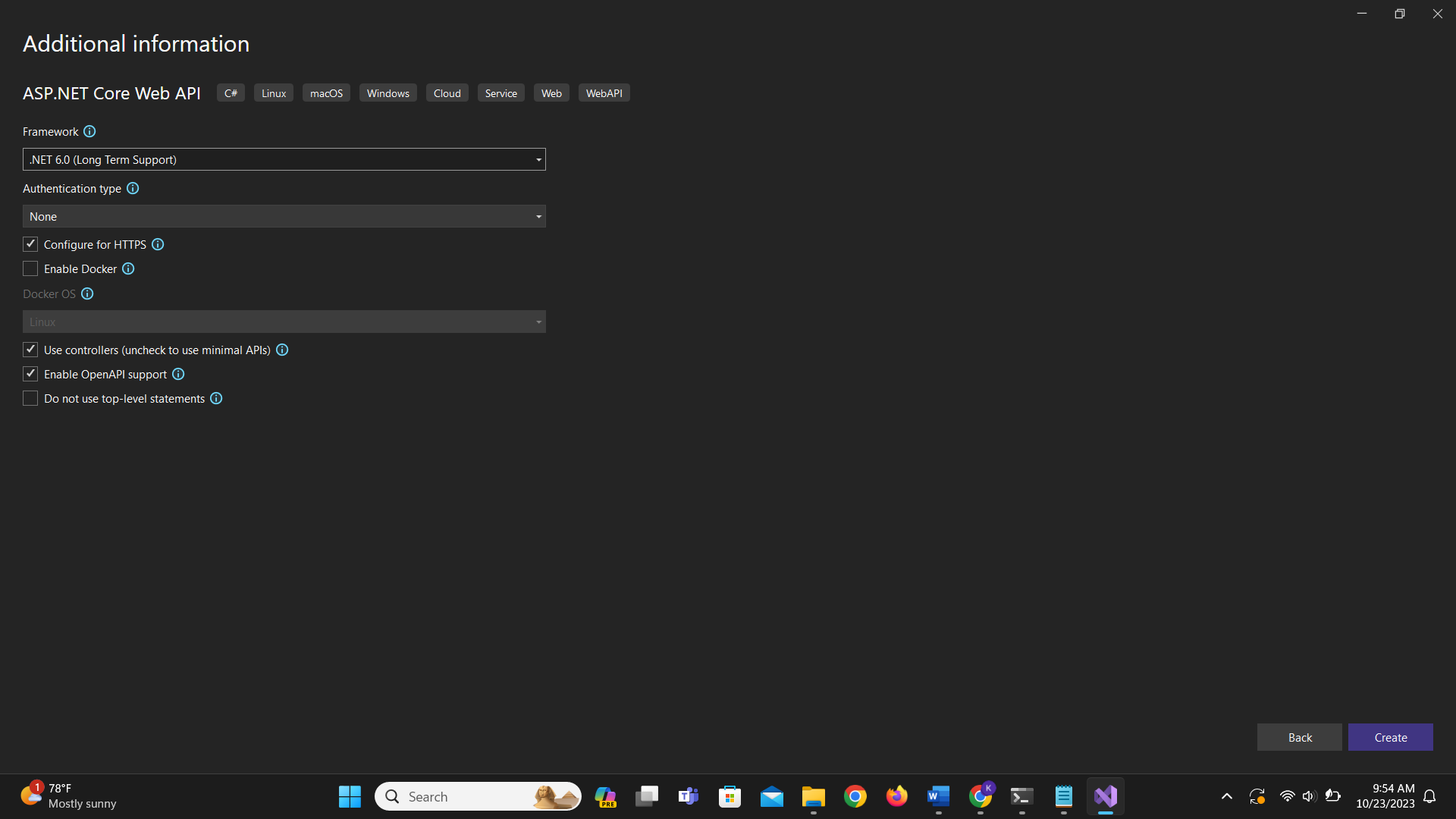
Step 3: Dockerize the .NET Core Web API

Step 4: Build the Docker Image

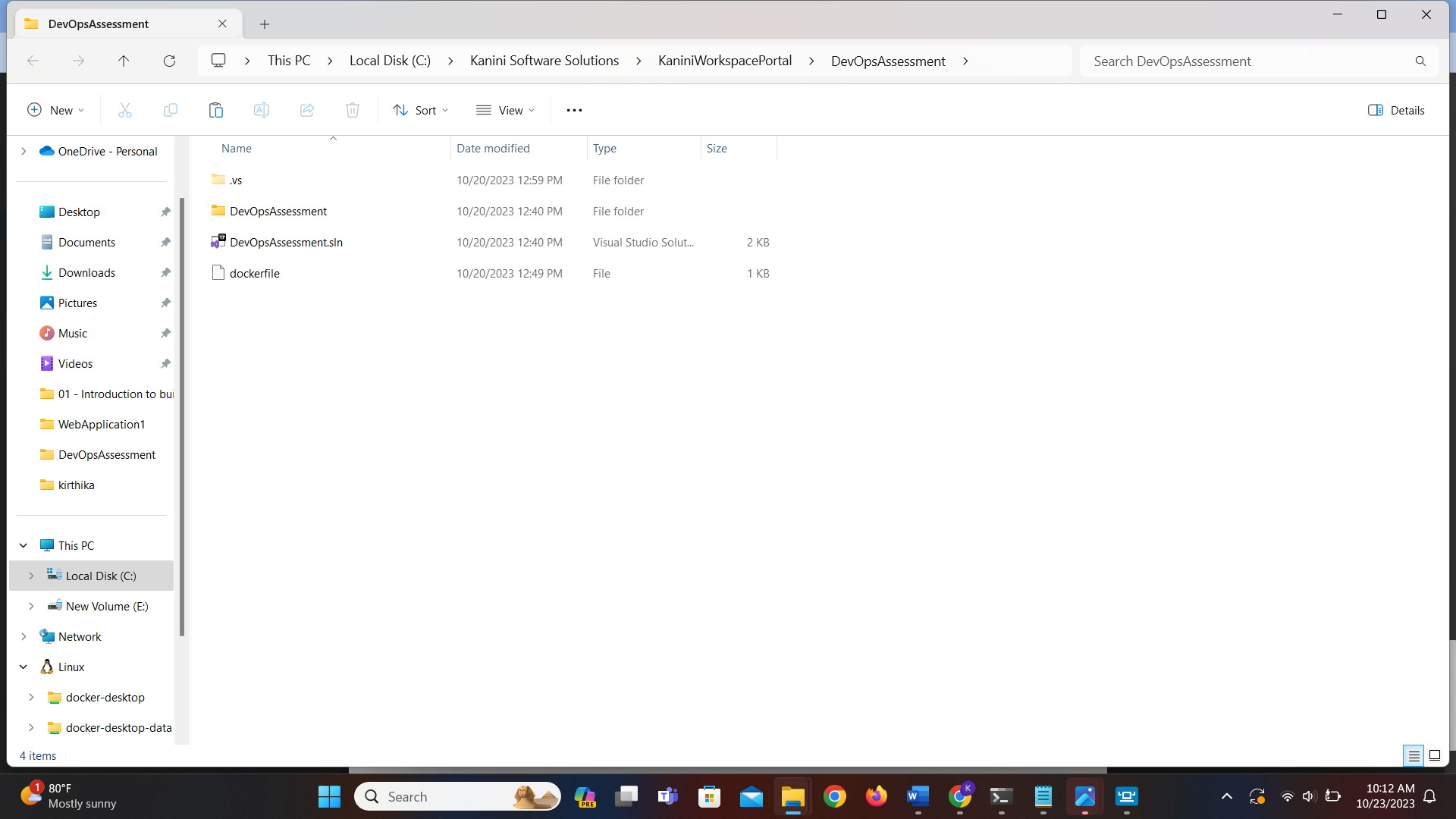
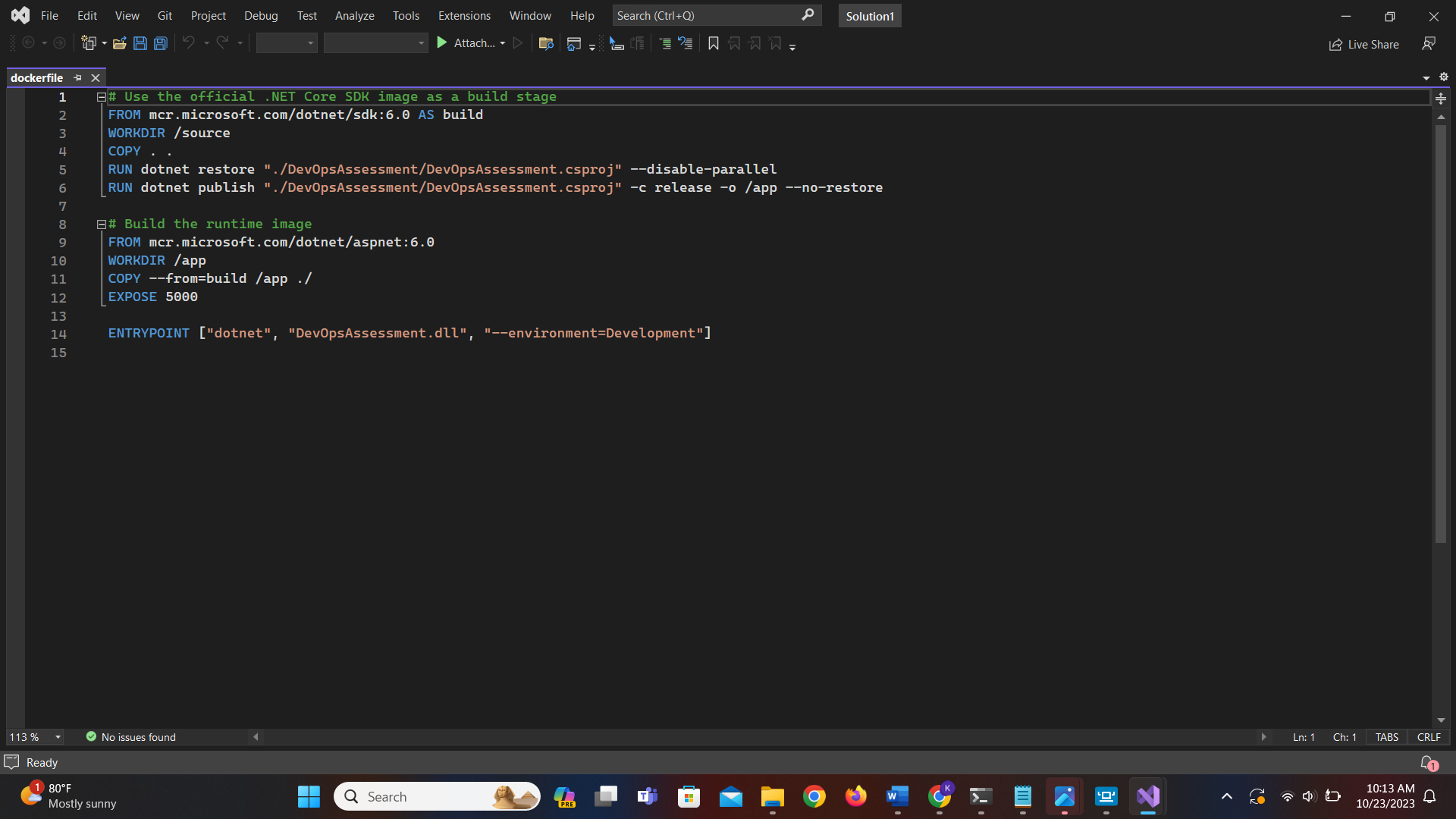
Step 5: Run the Docker Container in Rancher Desktop

Step 6: Test the .NET Core Web API via swagger

Ans:

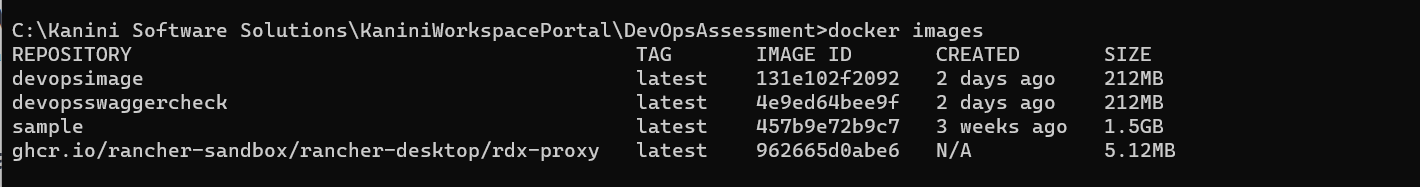
Step 1: Create a .NET Core Web API Project  


Step 2 & 3: Build the .NET Core Web API Project and Dockerize it

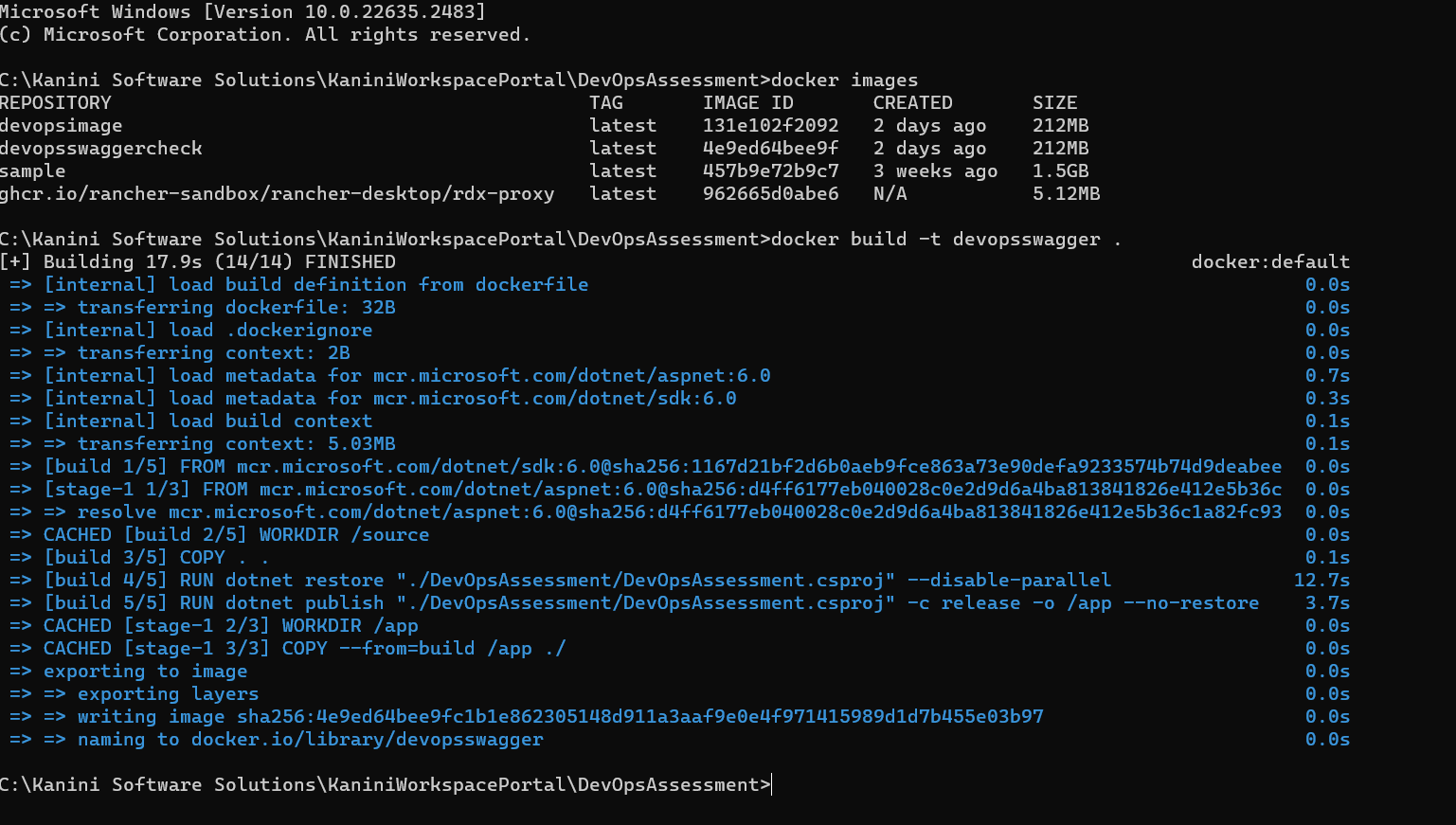
  


Step 4: Build the Docker Image

Listing all images :

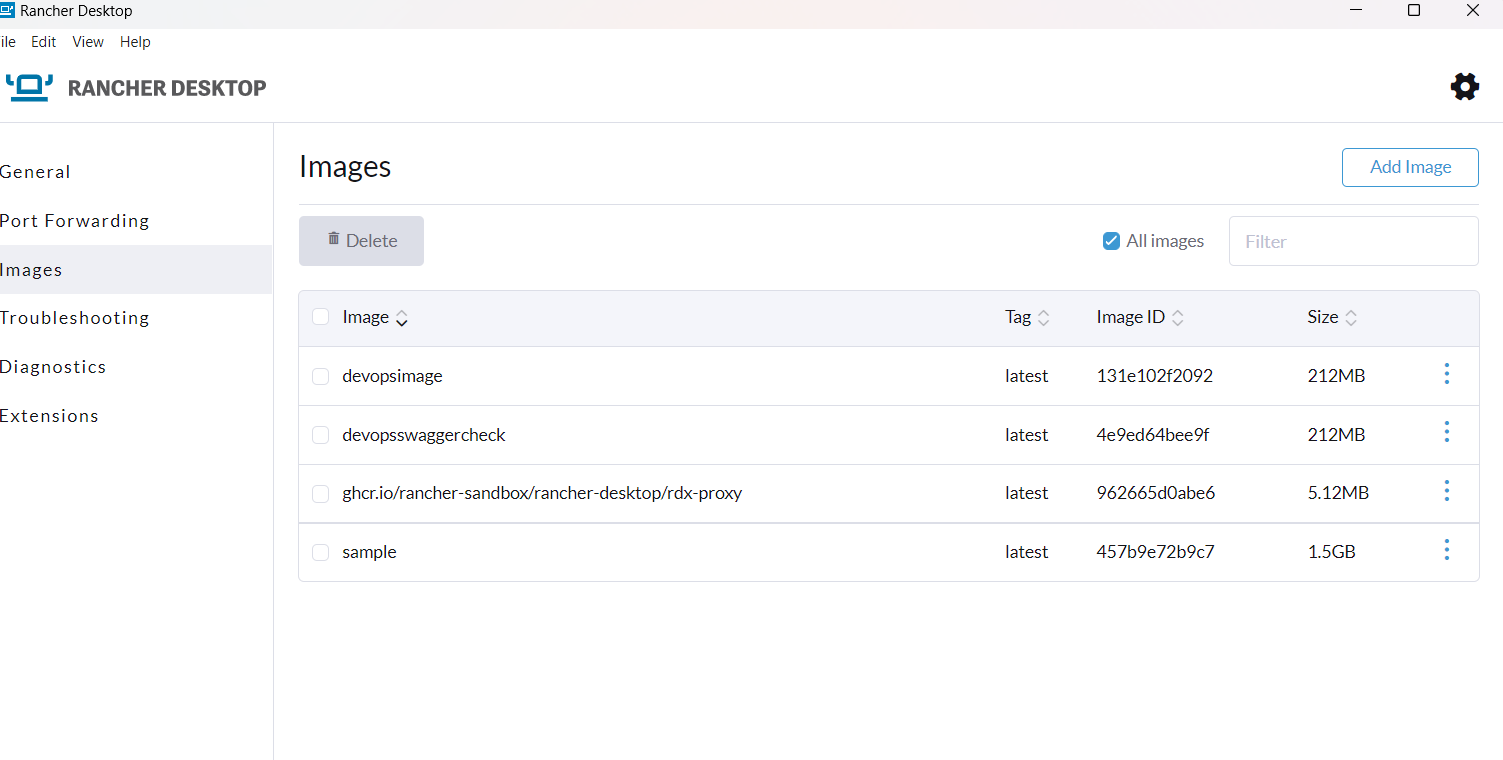


Building the docker file



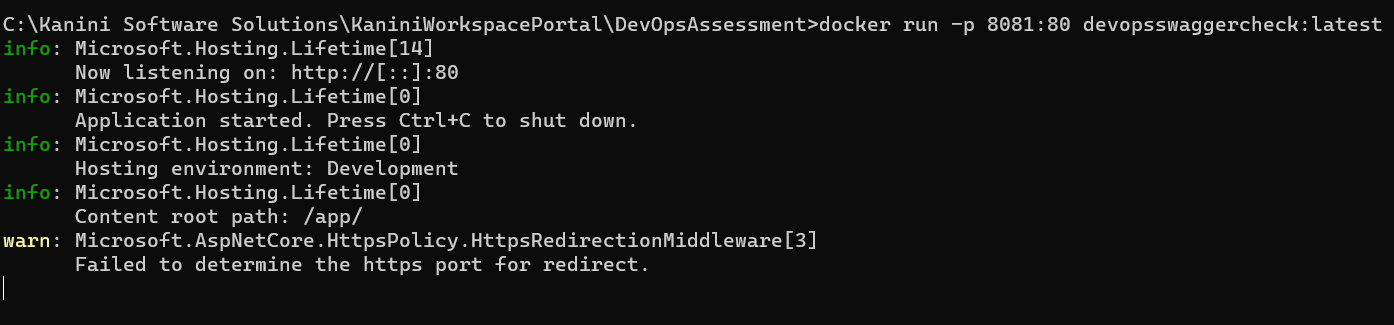
Step 5: Run the Docker Container in Rancher Desktop

You can find the docker image “devopsswaggercheck” in Rancher

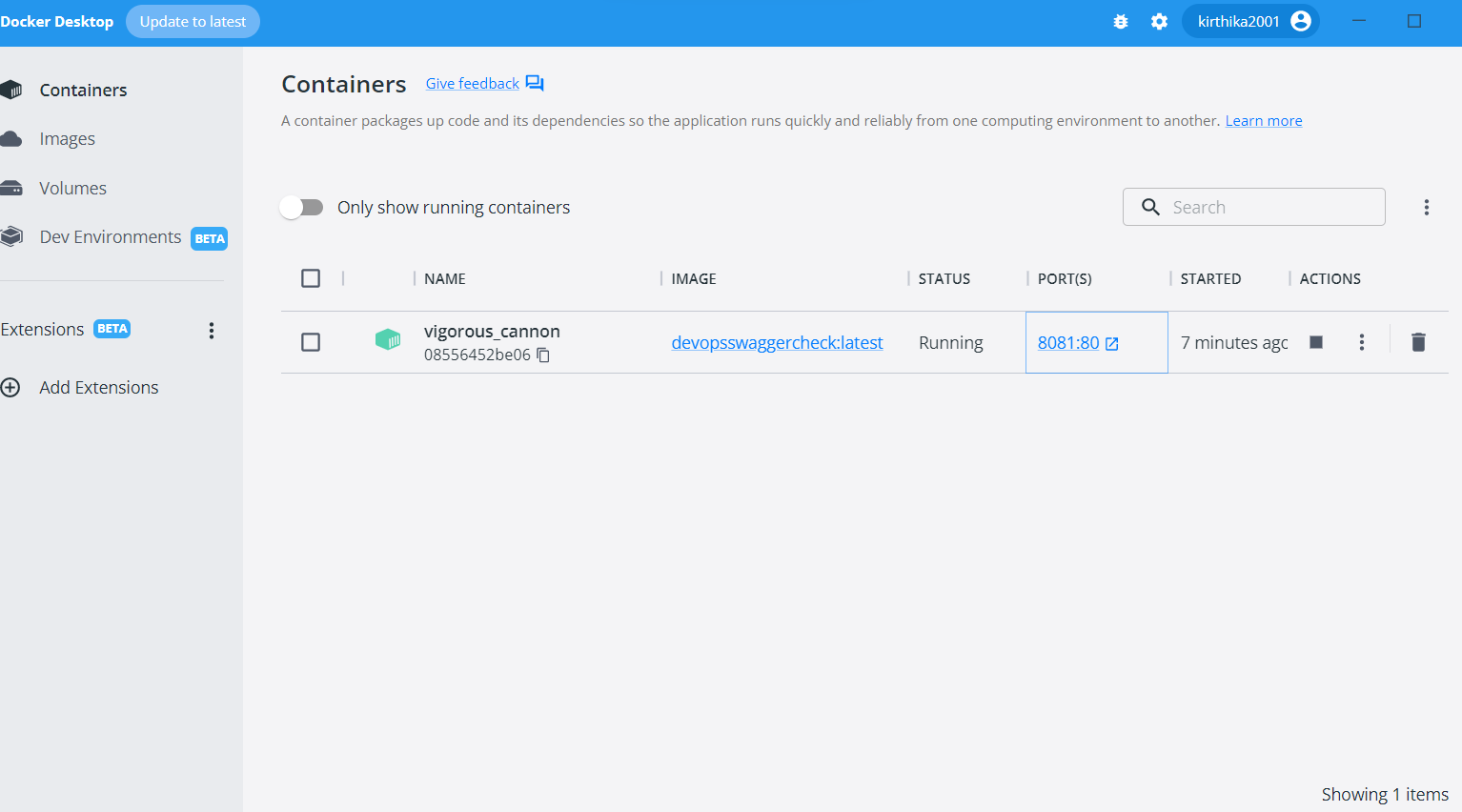


Step 6: Test the .NET Core Web API via swagger

Now I am running the docker image :



Its runs in swagger via docker



Finally the .net application runs on swagger via docker and rancher

