DEVOPS FINAL ASSESSMENT

SECTION : 1

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

   Answer: Windows Subsystem for Linux

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

   Answer: Frequent integration of code changes

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

   Answer: Change the working directory

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

   Answer: Docker

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

   Answer: Packaging and deploying applications in containers

6. What is the primary purpose of Azure DevOps?

   Answer: Software development and delivery

7. Which components are part of Azure DevOps?

   Answer: Azure Boards and Azure Pipelines

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

   Answer: It tracks changes in source code and manages versions

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

   Answer: Administrative tasks with superuser privileges

10. In Azure DevOps, which component is used to define, build, test, and deploy applications?

    Answer: Azure Pipelines

SECTION : 2

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.

```
kbrkannan@KBRKANNAN:~$ pwd
/home/kbrkannan
kbrkannan@KBRKANNAN:~$ mkdir lab1
kbrkannan@KBRKANNAN:~$ cd lab1
kbrkannan@KBRKANNAN:~/lab1$ touch sample.txt
kbrkannan@KBRKANNAN:~/lab1$ ls
sample.txt
kbrkannan@KBRKANNAN:~/lab1$ cp sample.txt sample_copy.txt
kbrkannan@KBRKANNAN:~/lab1$ mv sample_copy.txt new_sample.txt
kbrkannan@KBRKANNAN:~/lab1$ ls
new_sample.txt  sample.txt
kbrkannan@KBRKANNAN:~/lab1$
```

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.

```
kbrkannan@KBRKANNAN:~$ mkdir lab2
kbrkannan@KBRKANNAN:~$ cd lab2
kbrkannan@KBRKANNAN:~/lab2$ touch secret.txt
kbrkannan@KBRKANNAN:~/lab2$ chmod u+rw secret.txt
kbrkannan@KBRKANNAN:~/lab2$ cat > secret.txt
New File Addedkbrkannan@KBRKANNAN:~/lab2$ sudo chown new_user secret.txt
kbrkannan@KBRKANNAN:~/lab2$ ls -l
total 4
-rw-r--r-- 1 new_user kbrkannan 14 Oct 20 11:11 secret.txt
kbrkannan@KBRKANNAN:~/lab2$ ls -n
total 4
-rw-r--r-- 1 1001 1000 14 Oct 20 11:11 secret.txt
kbrkannan@KBRKANNAN:~/lab2$
```

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.

```
kbrkannan@KBRKANNAN:~$ mkdir lab3
kbrkannan@KBRKANNAN:~$ cd lab3
kbrkannan@KBRKANNAN:~/lab3$ touch newtextfile.txt
kbrkannan@KBRKANNAN:~/lab3$ cat > newtextfile.txt
This is the new file added in the new text file.kbrkannan@KBRKANNAN:~/lab3$ grep This newtextfile.txt
This is the new file added in the new text file.
kbrkannan@KBRKANNAN:~/lab3$ sed -i 's/added/replaced/g' newtextfile.txt
kbrkannan@KBRKANNAN:~/lab3$ cat newtextfile.txt
This is the new file replaced in the new text file.kbrkannan@KBRKANNAN:~/lab3$ wc newtextfile.txt
 0 11 51 newtextfile.txt
kbrkannan@KBRKANNAN:~/lab3$
```

Lab 4 : Creating a simple yaml file

 Objective: Create a basic YAML configuration file.

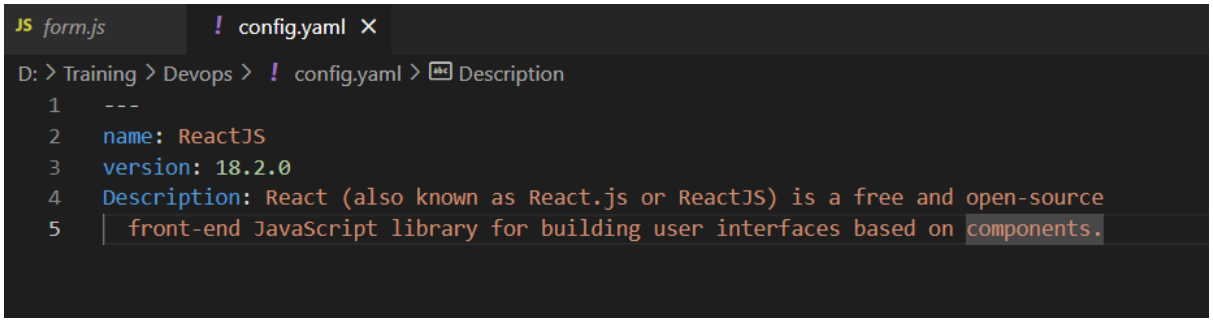
 Task:

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.

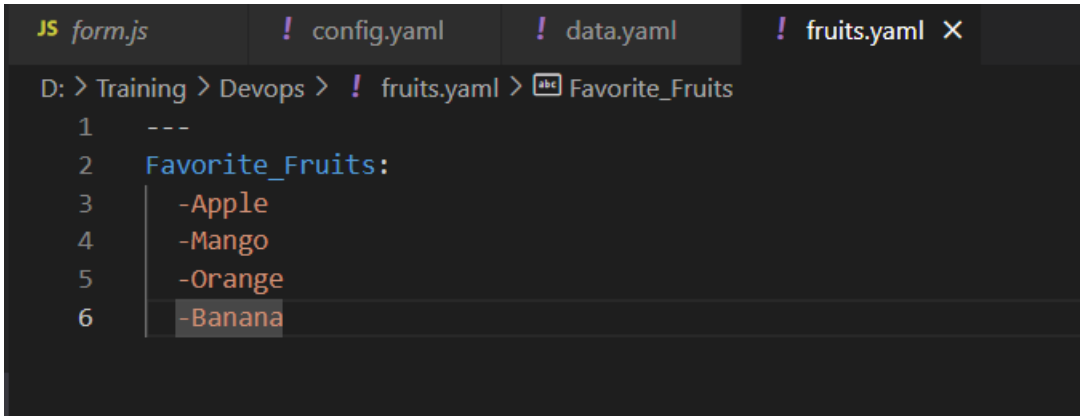


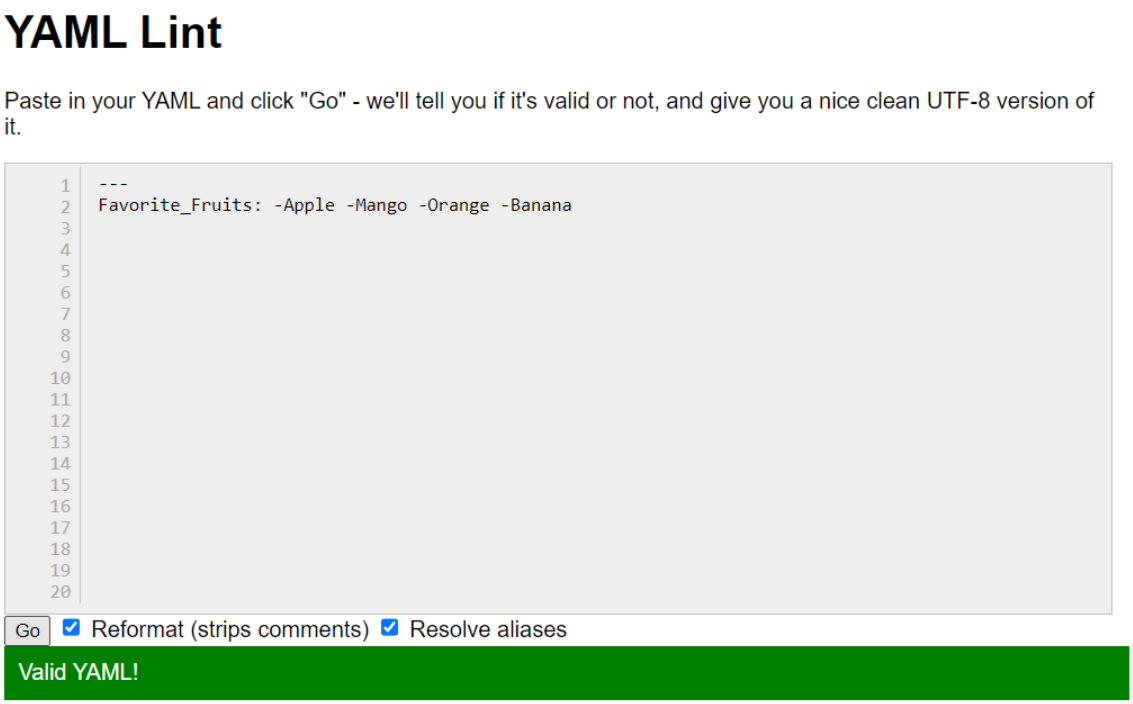
Lab 5 : Working with lists in yaml

 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.





Lab 6 : Nested Structures in yaml

 Objective: Explore nested structures within YAML.

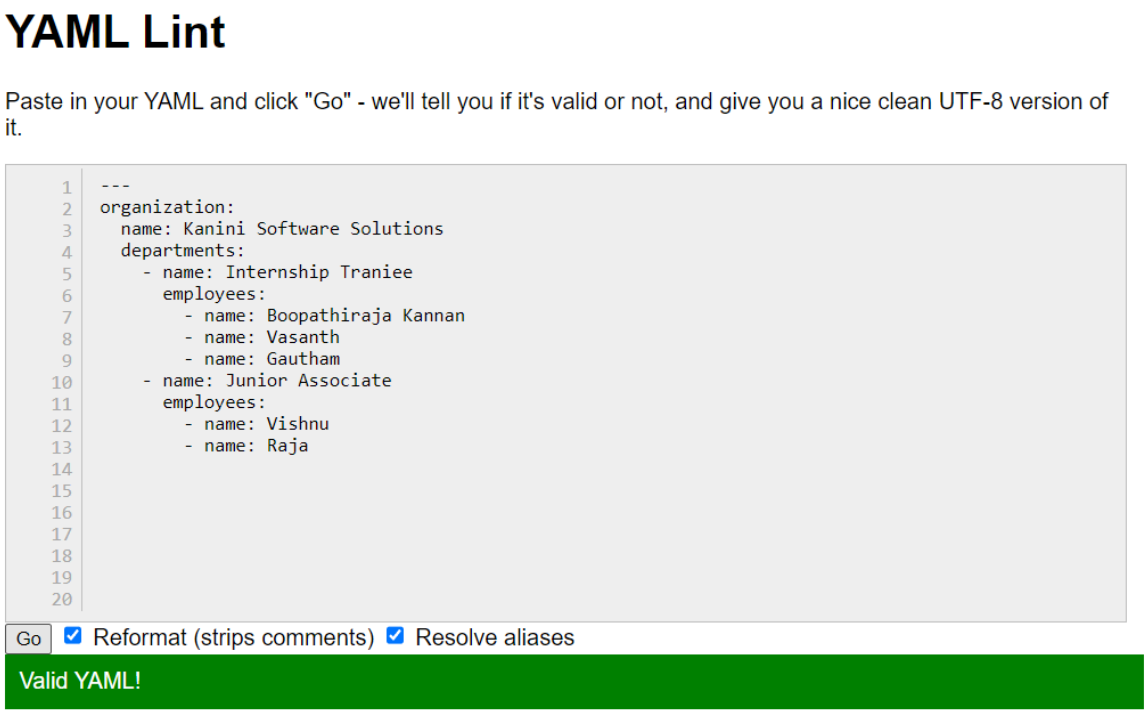
 Task:

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.

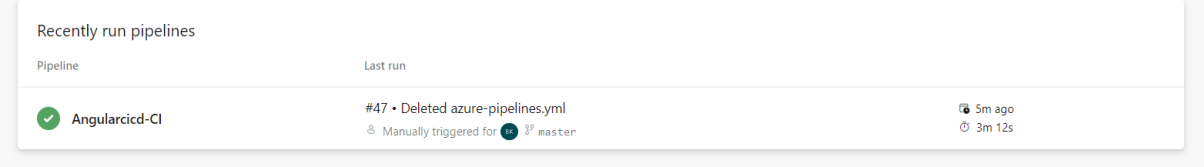


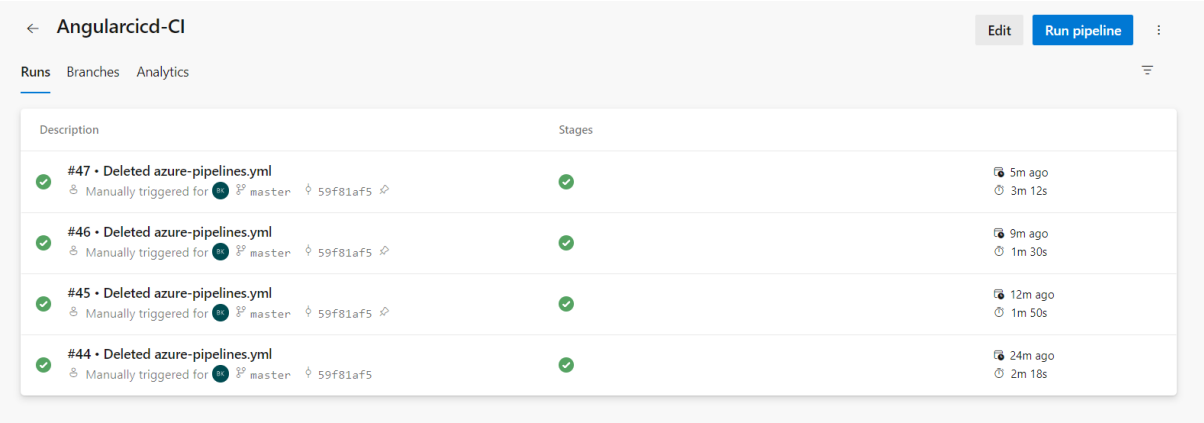
Lab 7: Create Classic Azure CI Pipeline for Angular Application

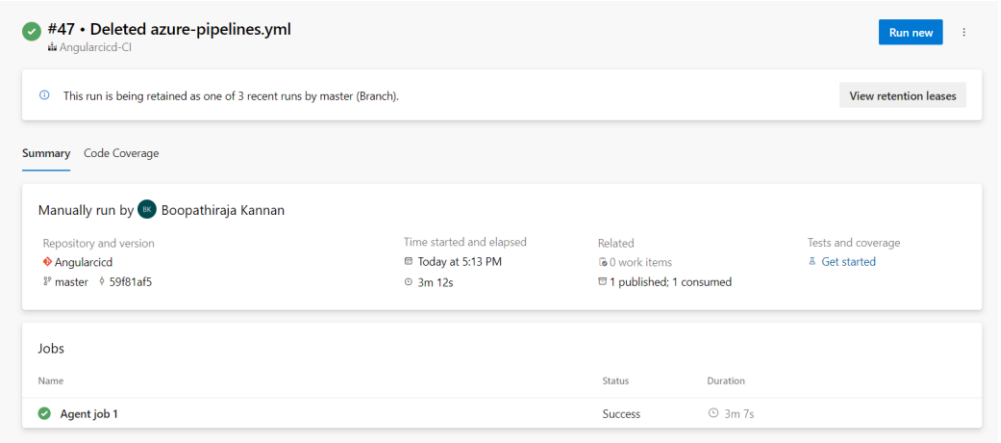
 Objective: Set up a classic Azure CI pipeline to build a simple Angular application with unit testing using Jasmine and Karma.

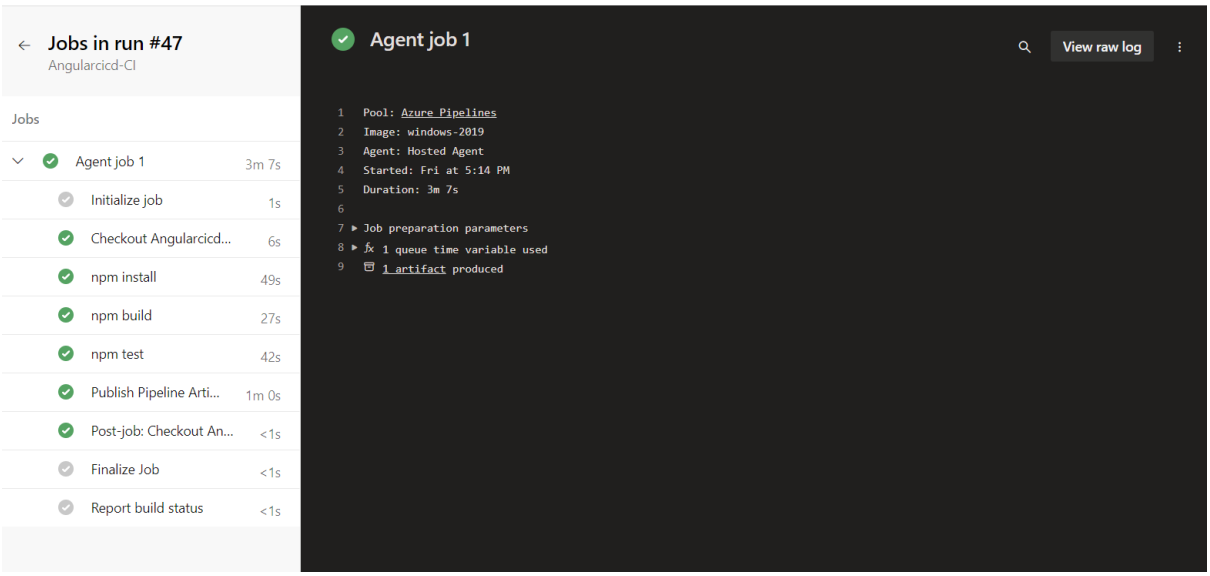
 Tasks:

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.







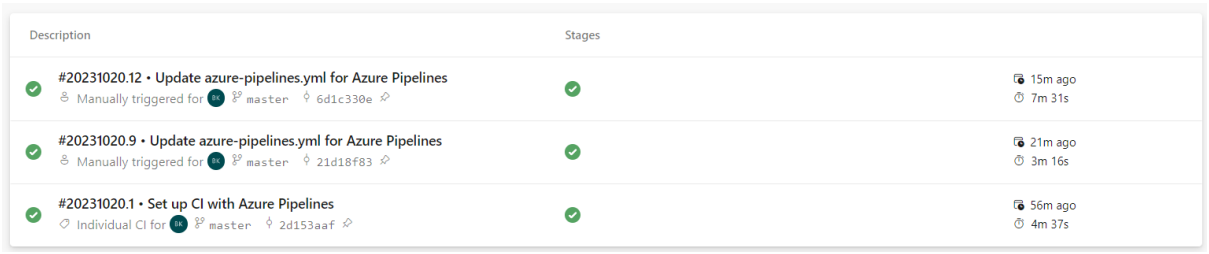


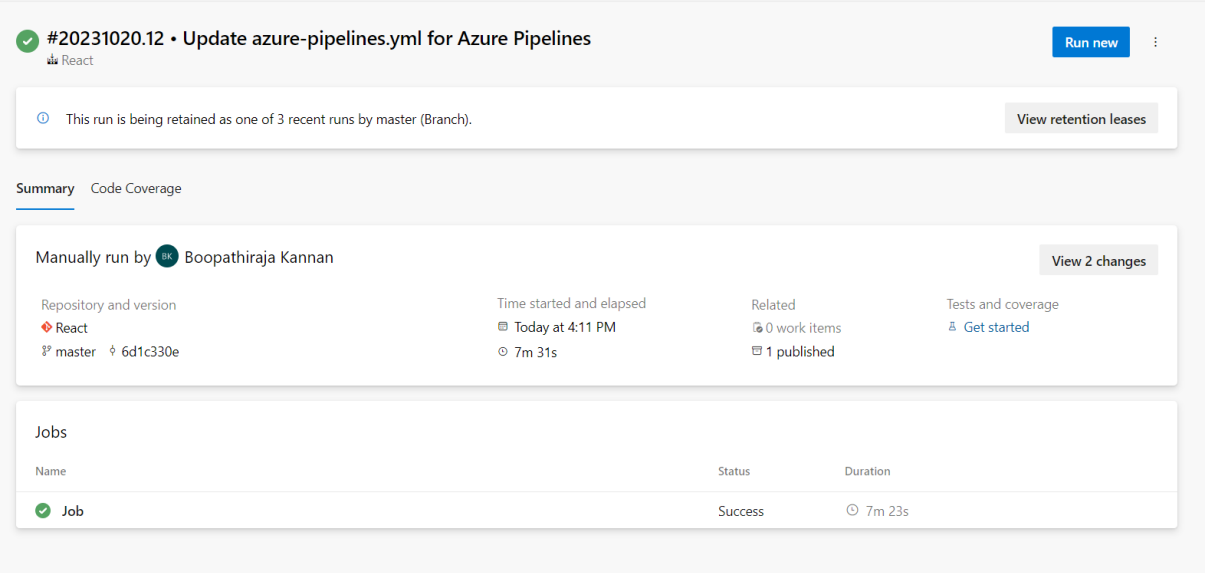
Lab 8 : Create YAML Azure CI Pipeline for React Application

 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.



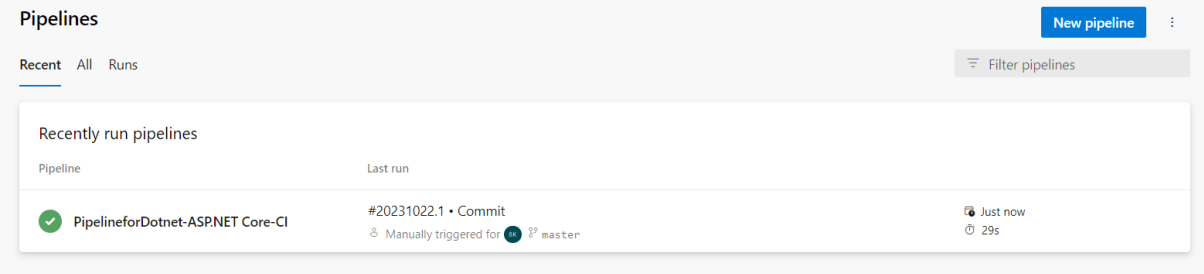


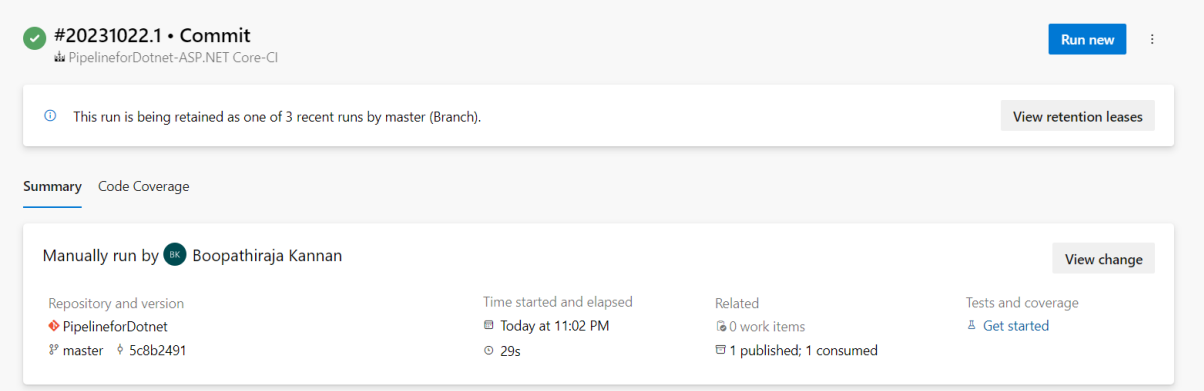
Lab 9: Create CI Pipeline for .NET Core Application with MS Unit Test

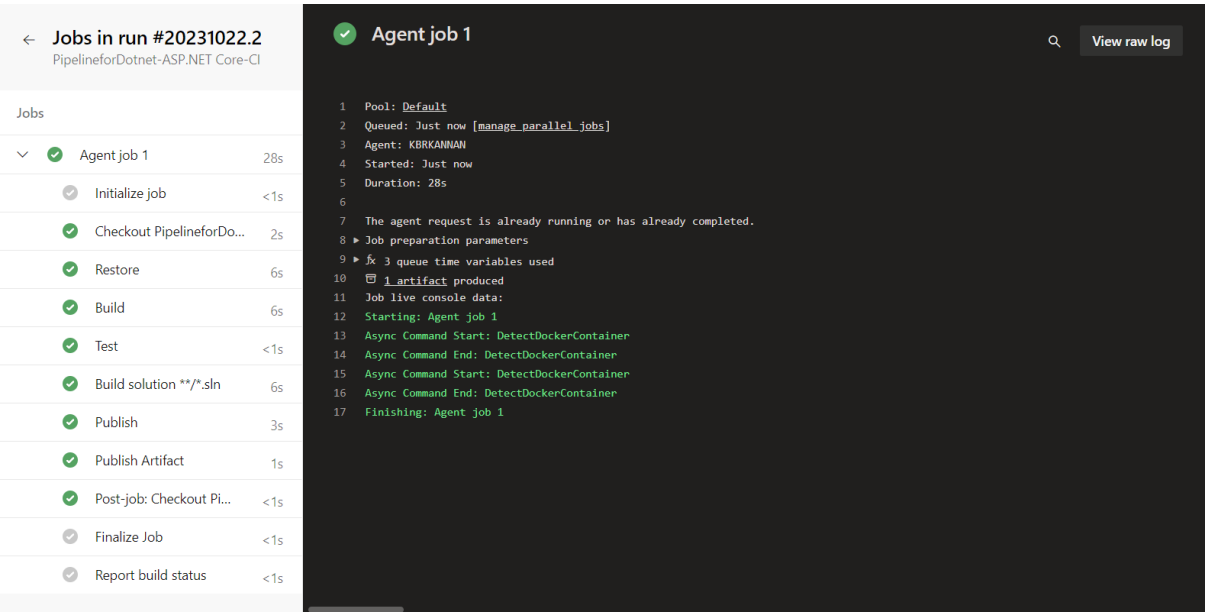
 Objective: Create a CI pipeline, either classic or YAML, to build a .NET Core application and run MS Unit tests.

 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.







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

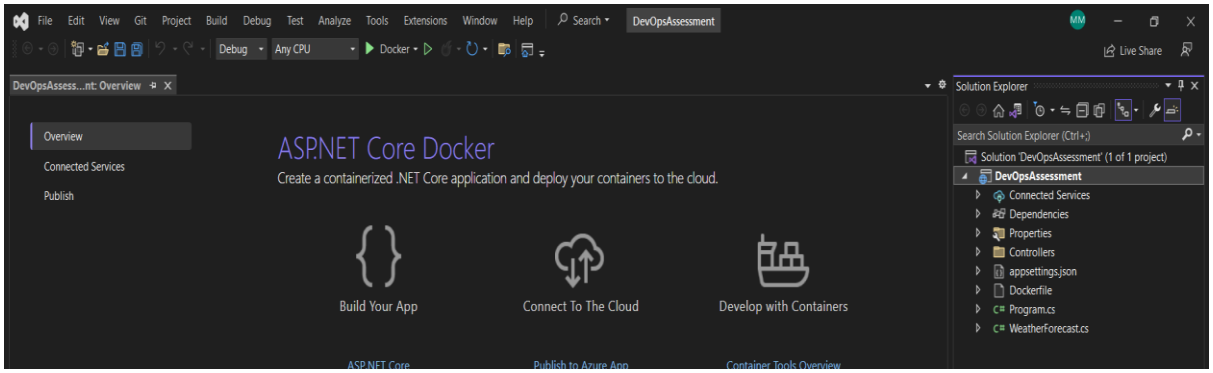
Step 2: Build the .NET Core Web API Project

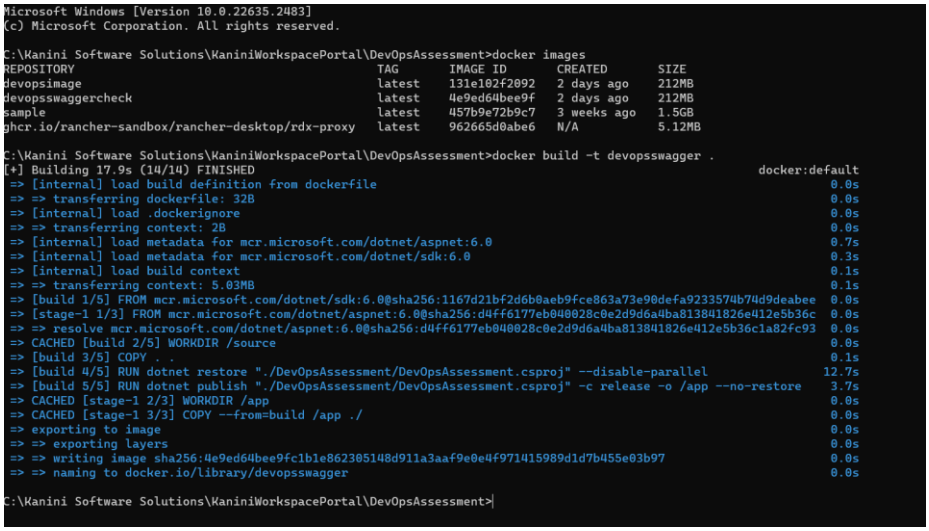
Step 3: Dockerize the .NET Core Web API

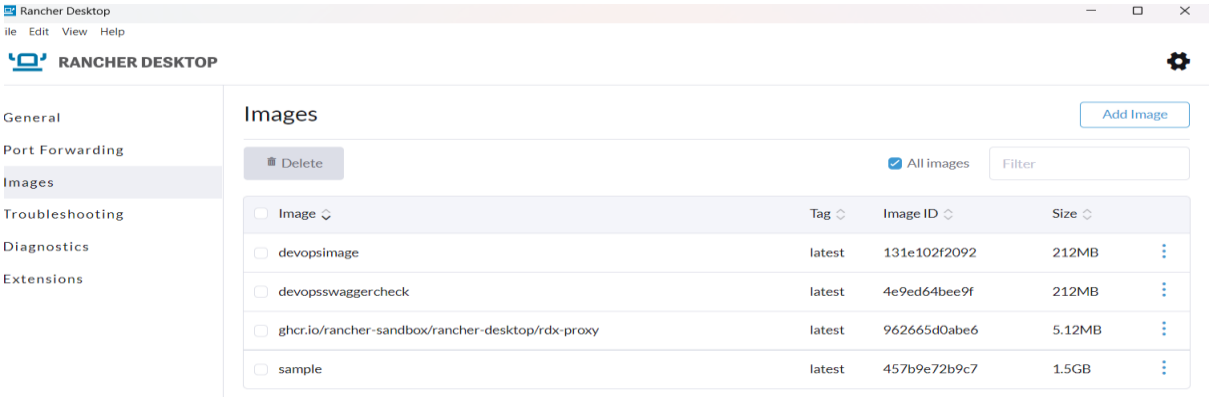
Step 4: Build the Docker Image

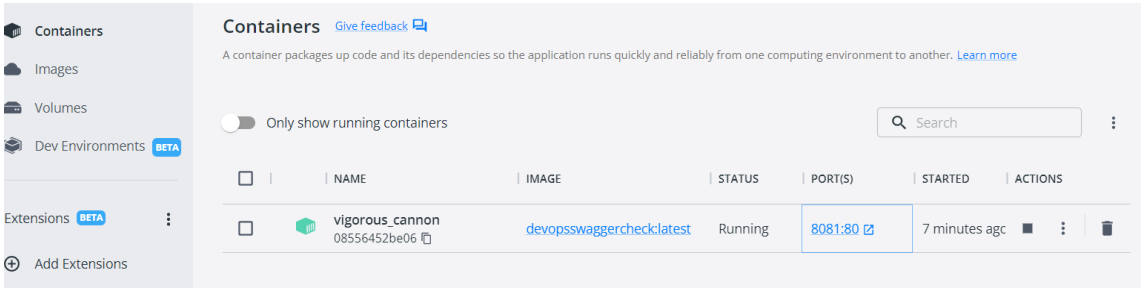
Step 5: Run the Docker Container in Rancher Desktop

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










```
C:\Kanini Software Solutions\KaniniWorkspacePortal\DevOpsAssessment>docker run -p 8081:80 devopsswaggercheck:latest
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:80
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app/
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
      Failed to determine the https port for redirect.
```