**Devops – Final Assessment**

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

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

a. Windows Software Locator

b. Windows System Locator

**c. Windows Subsystem for Linux**

d. Windows Shell Language

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

a. Automating manual testing

**b. Frequent integration of code changes**

c. Managing cloud infrastructure

d. Monitoring server performance

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

a. Copy files and directories

**b. Change the working directory**

c. Create a new directory

d. Calculate directory size

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

a. Ubuntu

b. CentOS

**c. Docker**

d. Debian

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

a. Managing cloud infrastructure

b. Running virtual machines

**c. Packaging and deploying applications in containers**

d. Managing network security

6. What is the primary purpose of Azure DevOps?

a. Infrastructure management

**b. Software development and delivery**

c. Network security

d. Virtualization

7. Which components are part of Azure DevOps?

a. Azure App Service and Azure Functions

b. Azure Monitor and Azure Security Center

**c. Azure Boards and Azure Pipelines**

d. Azure Virtual Machines and Azure SQL Database

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

a. It provides automated database backups.

**b. It tracks changes in source code and manages versions.**

c. It monitors server performance.

d. It optimizes network configurations.

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

a. Managing user accounts

b. Running GUI applications

**c. Administrative tasks with superuser privileges**

d. Monitoring network traffic

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

a. Azure Boards

b. Azure Repos

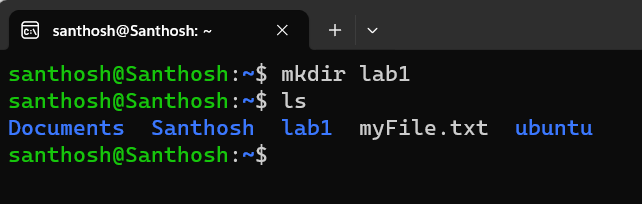
**c. Azure Pipelines**

d. Azure Artifacts

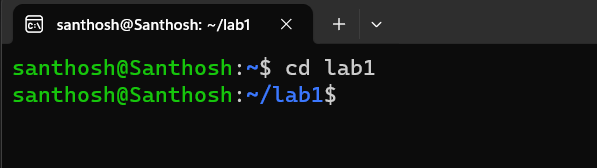
**Section 2: Labs**

**Lab 1:** File and Directory Management

1. Create **“lab1”** directory in home :



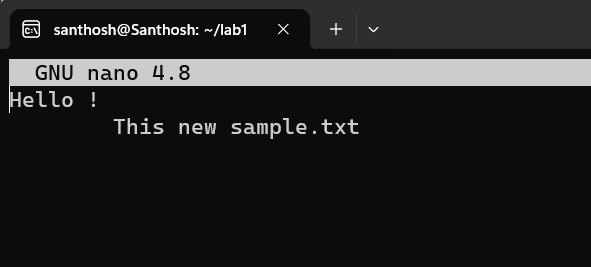
1. Inside the lab1 directory , ‘sample1.txt’ file created using the **“touch sample1.txt”** command



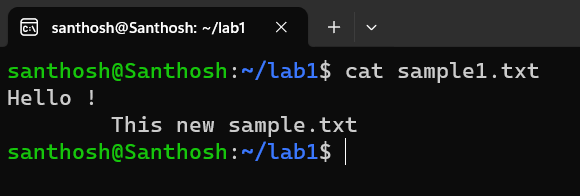
Screenshot 2023-10-22 222600

1. Using **“nano sample1.txt”** command , sample1.txt file in nano editor .

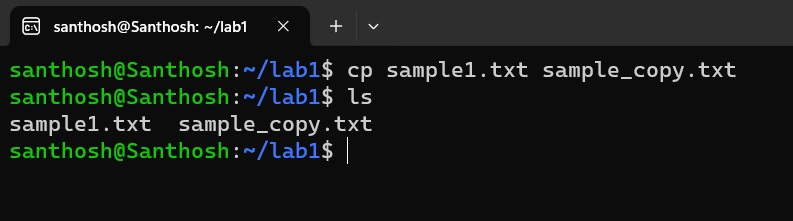
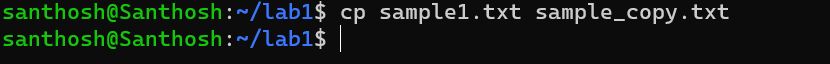
New text is added to that file.

Screenshot 2023-10-22 222715

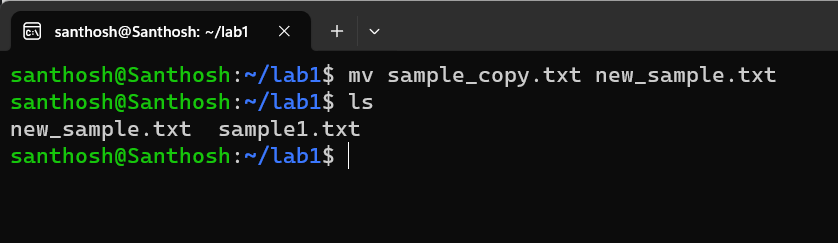
1. Using **“cat sample1.txt”** command , we can view the content ,



1. Make a copy of "sample1.txt" and name it "sample\_copy.txt" using the **“cp command”** :

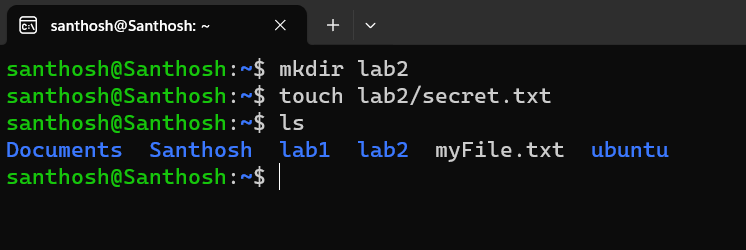


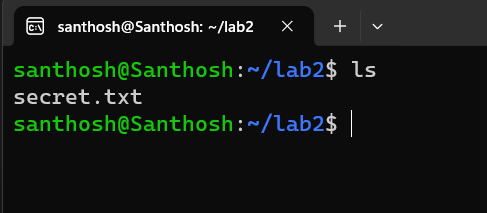
6 .Rename "sample\_copy.txt" to "new\_sample.txt" using the **“mv command”**:



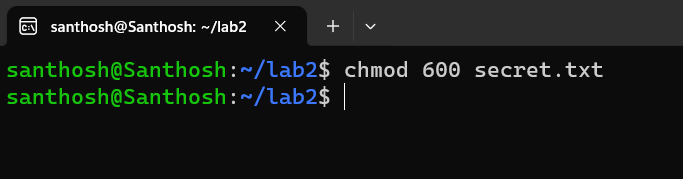
**Lab 2: Permissions and Ownership**

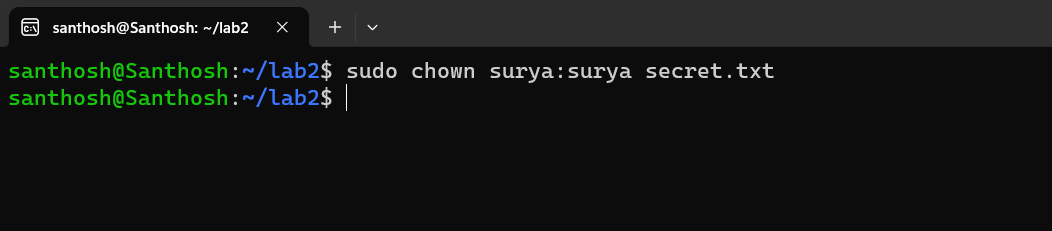
1. Create a new file named "secret.txt" in the "lab2" directory. and then create the "lab2" directory and the "secret.txt" file:



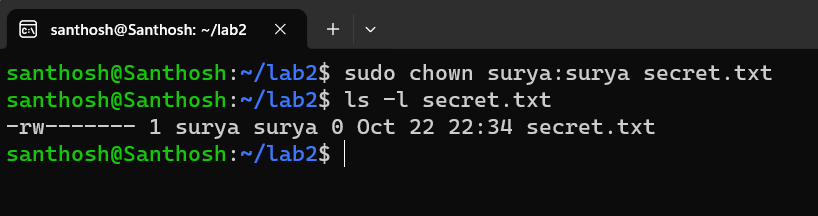


1. Set the file permissions to allow read and write access only to the owner. You can use the **“chmod”** command to do this.



1. Change the owner of "secret.txt" to another user. Use the **chown** command to change the owner. 

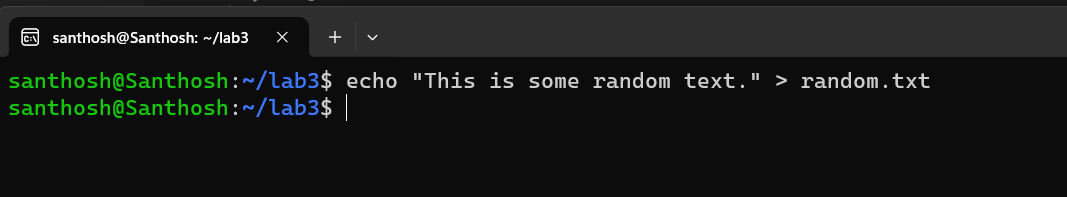
4 .Verify the new permissions and owner using the **ls -l** and **ls -n** commands:



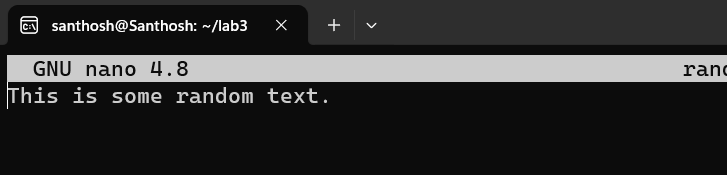
**Lab 3: Text Processing with Command Line Tools**

1. Create a text file with some random text in the "lab3" directory.

you can use a command to create the file and add text to it.

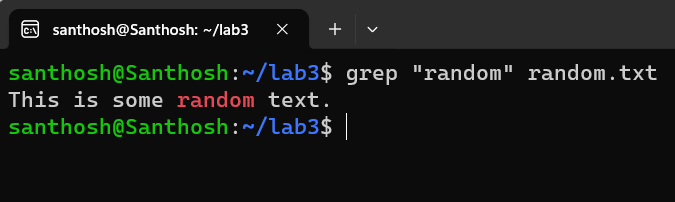


This command creates a file named "random.txt" in the "lab3" directory and adds the text "This is some random text." to it.



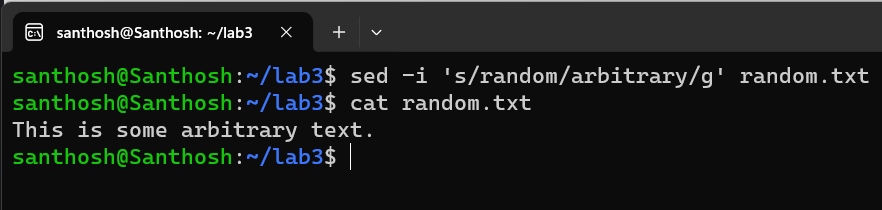
1. Use the **grep** command to search for a specific word or pattern in the file.

This command will search for the word "random" in the "random.txt" file and print any matching lines.

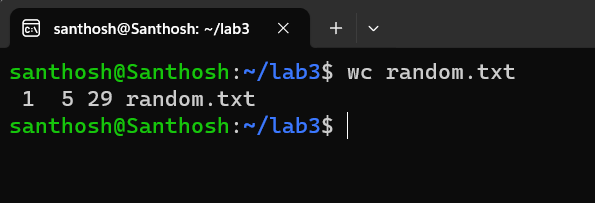


3. Use the **sed** command to replace a word or phrase with another in the file.

To replace a word, let's say you want to replace "random" with "arbitrary" in the "random.txt" file.

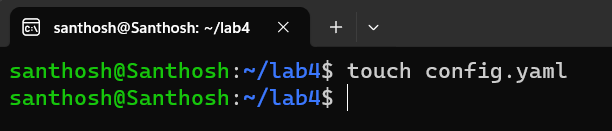


1. Use the **wc** command to count the number of lines, words, and characters in the file.



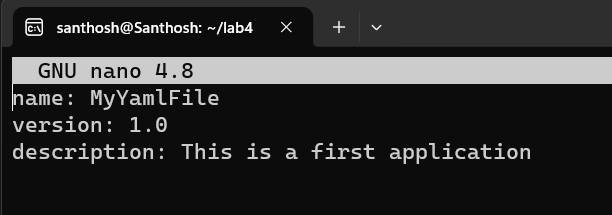
**Lab 4: Creating a Simple YAML File**

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



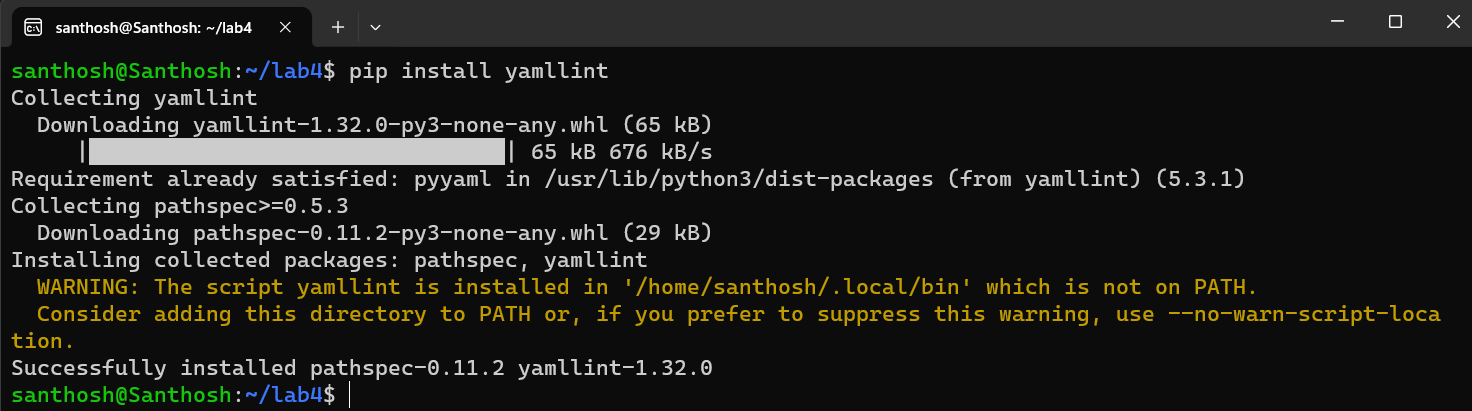
1. Define key-value pairs in YAML for a fictitious application.

Now, open the "config.yaml" file in a text editor and define key-value pairs for your fictitious application and Save the file.

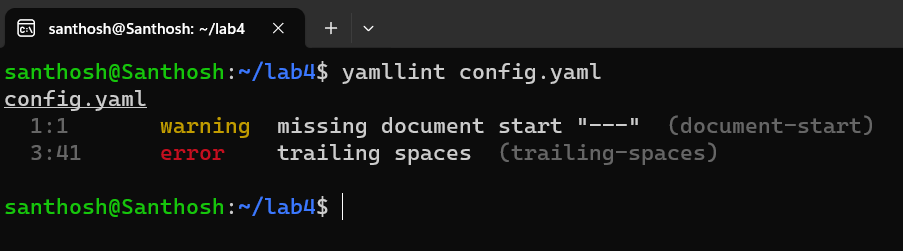


1. Validate that the YAML file is correctly formatted.

To validate that the YAML file is correctly formatted, you can use a YAML **linter** or parser. One popular tool for this is **yamllint**.

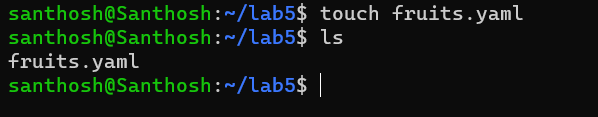


Once you have **yamllint** installed, you can check your YAML file for formatting issues using the following command:

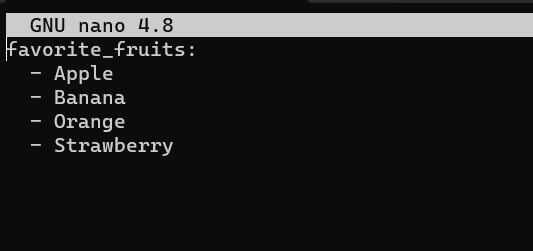


**Lab 5: Working with Lists in YAML**

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

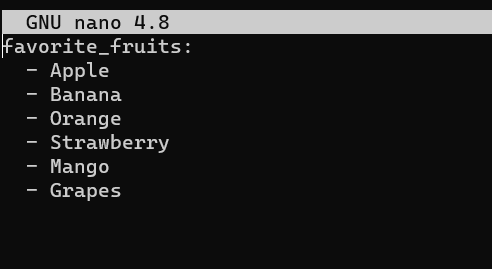


1. Define a list of your favorite fruits using YAML syntax.



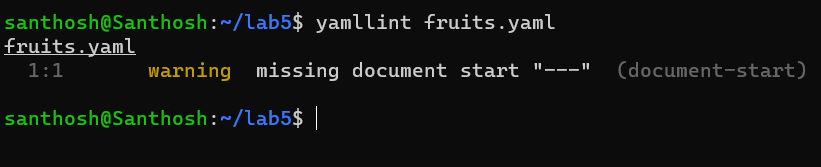
1. Add items to the list.

You can add more fruits to your list by simply extending the list with additional items. For instance, you can add "Mango" and "Grapes" like this:



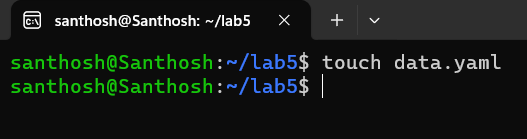
1. Save and validate the YAML file.

To validate that the YAML file is correctly formatted, you can use a YAML linter such as "yamllint" if you've installed it in a previous task.

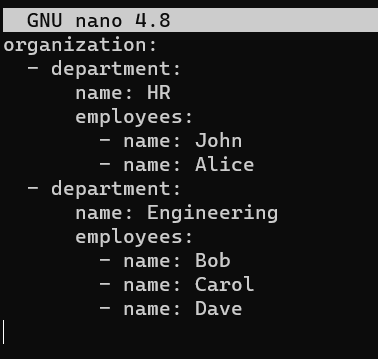


**Lab 6: Nested Structures in YAML**

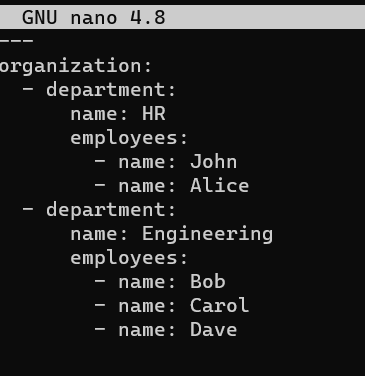
1. Create a YAML file named "data.yaml."



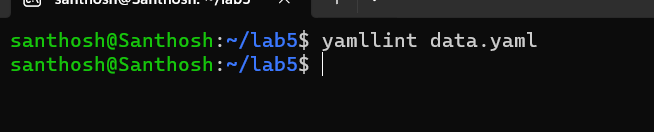
1. Define a nested structure representing a fictitious organization with departments and employees.



1. Use YAML syntax to add, update, or remove data within the nested structure.

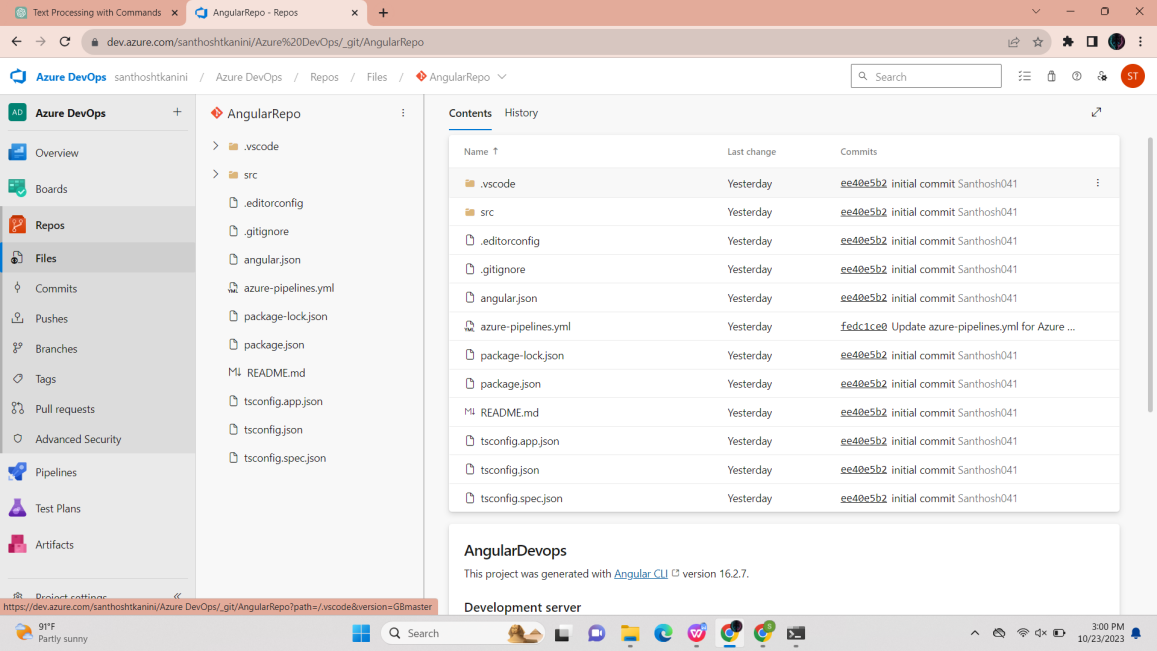


1. Save and validate the YAML file.

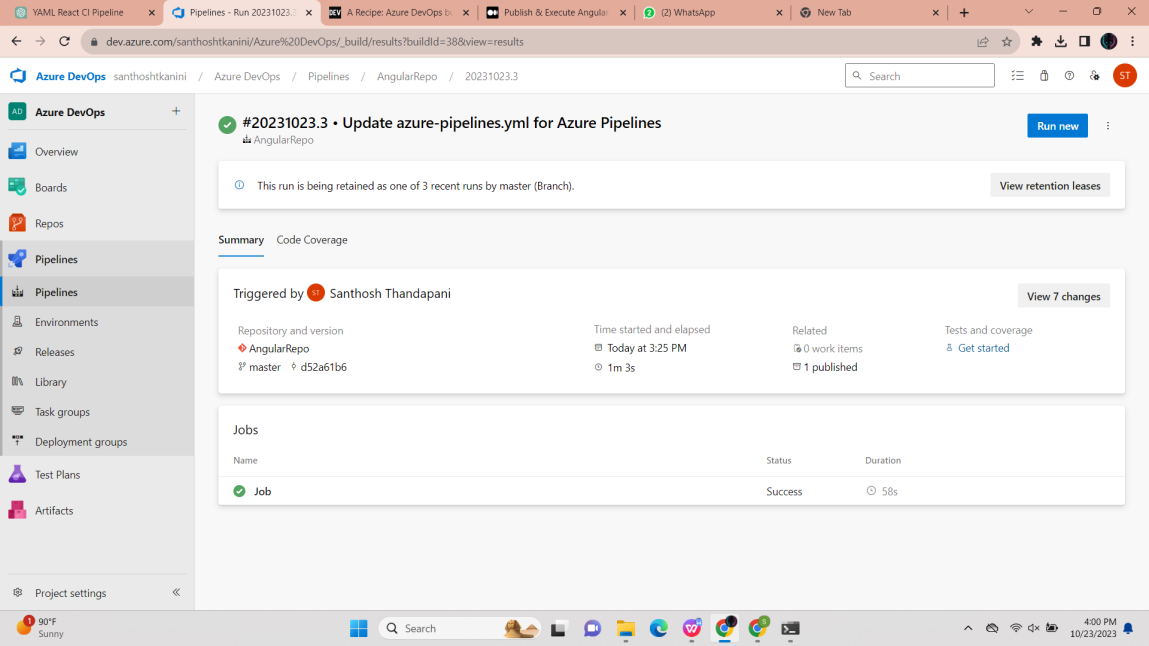


**Lab 7: Create Classic Azure CI Pipeline for Angular Application**

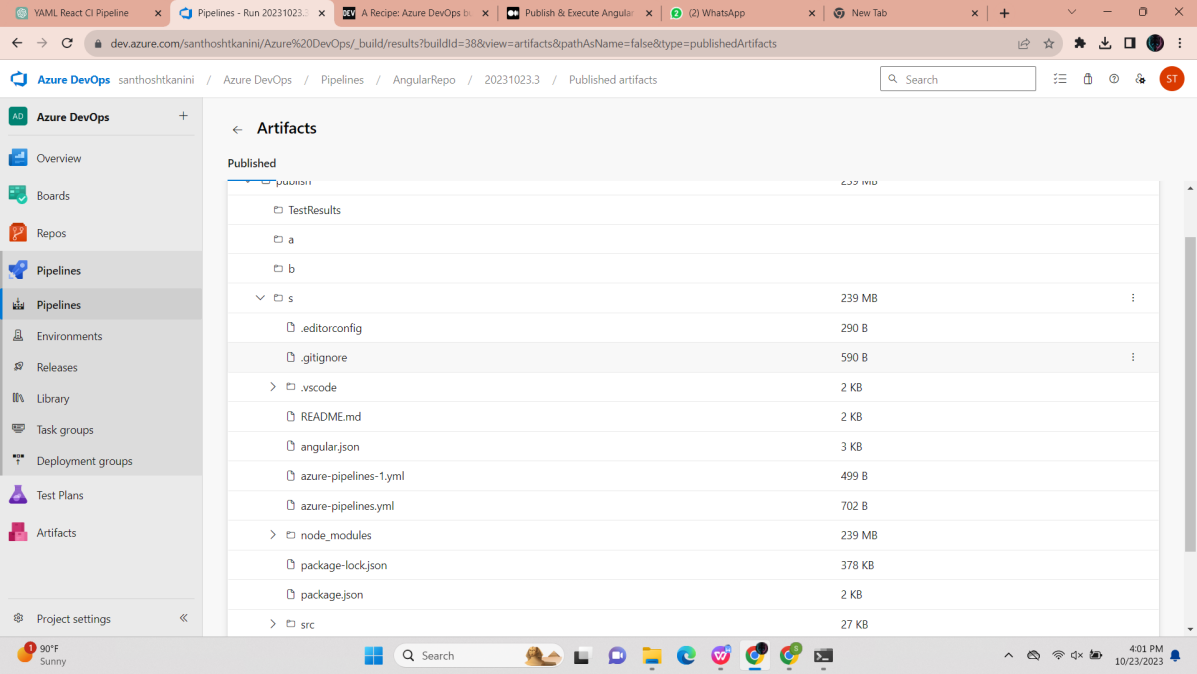
Angular repo

****

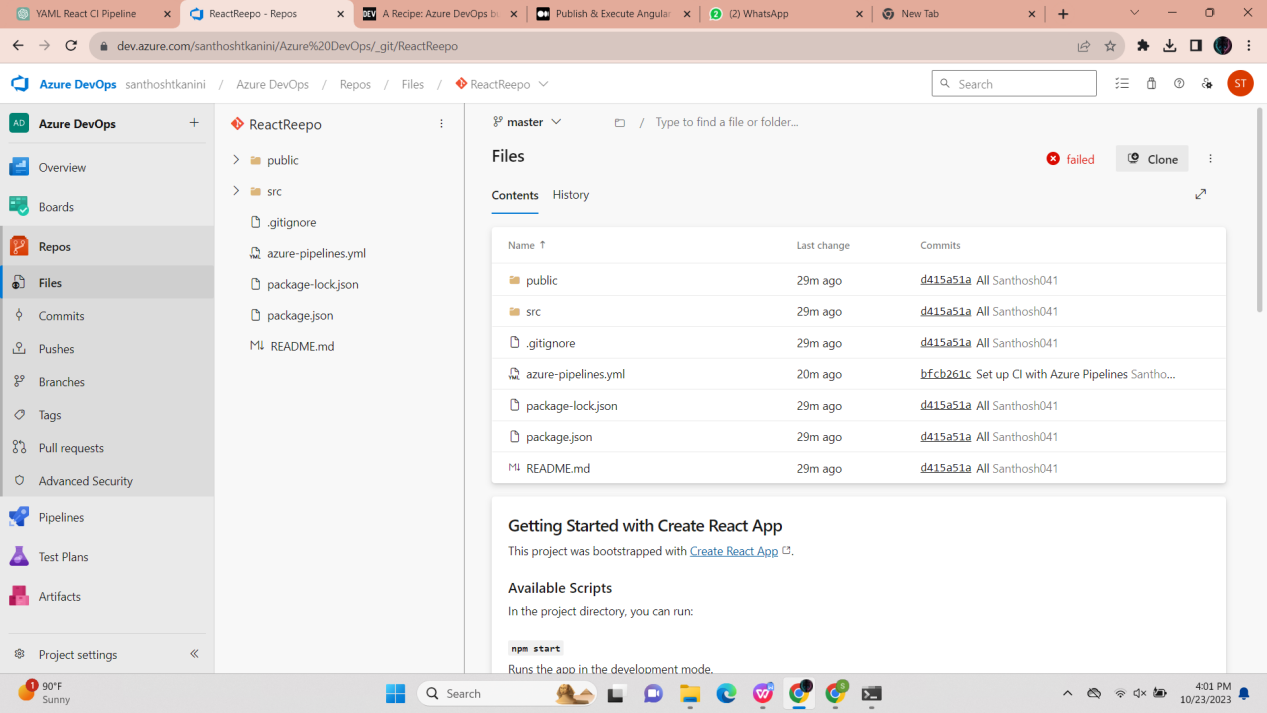
Azure pipeline for Angular:

****

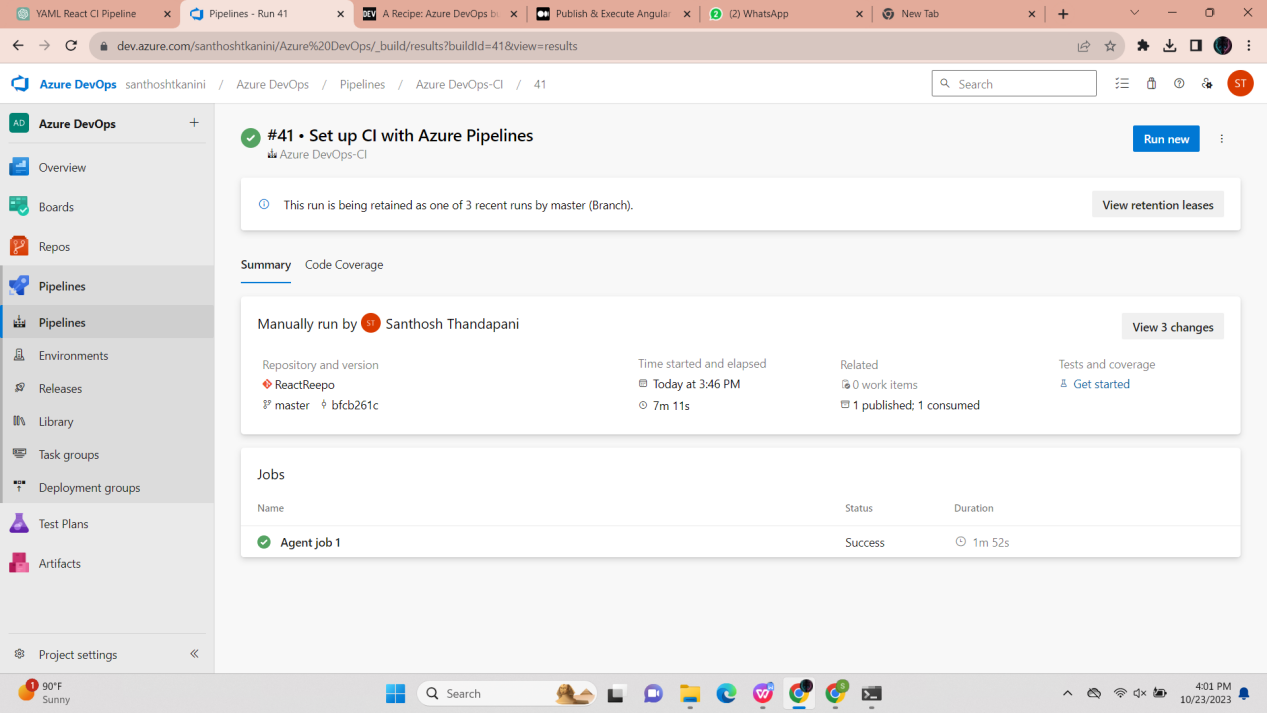
Published artifacts:

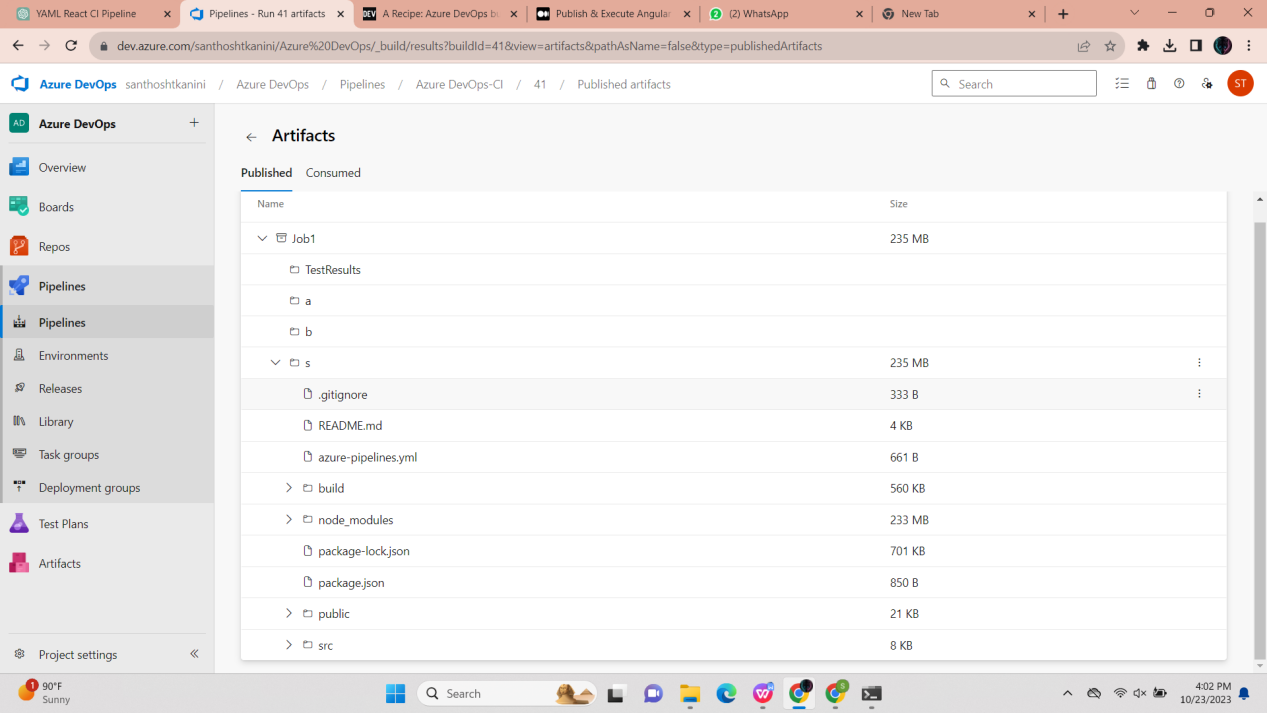
****

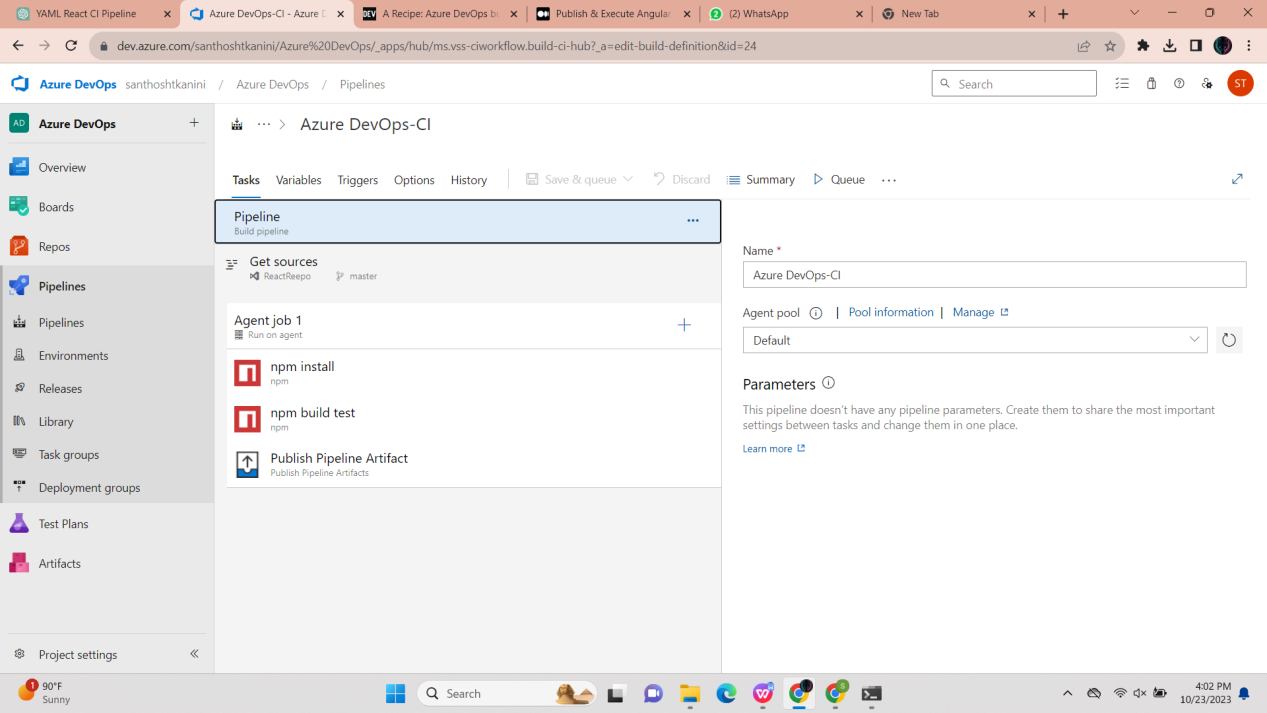
**Lab 8: Create YAML Azure CI Pipeline for React Application**

Azure Repo for React Project ****

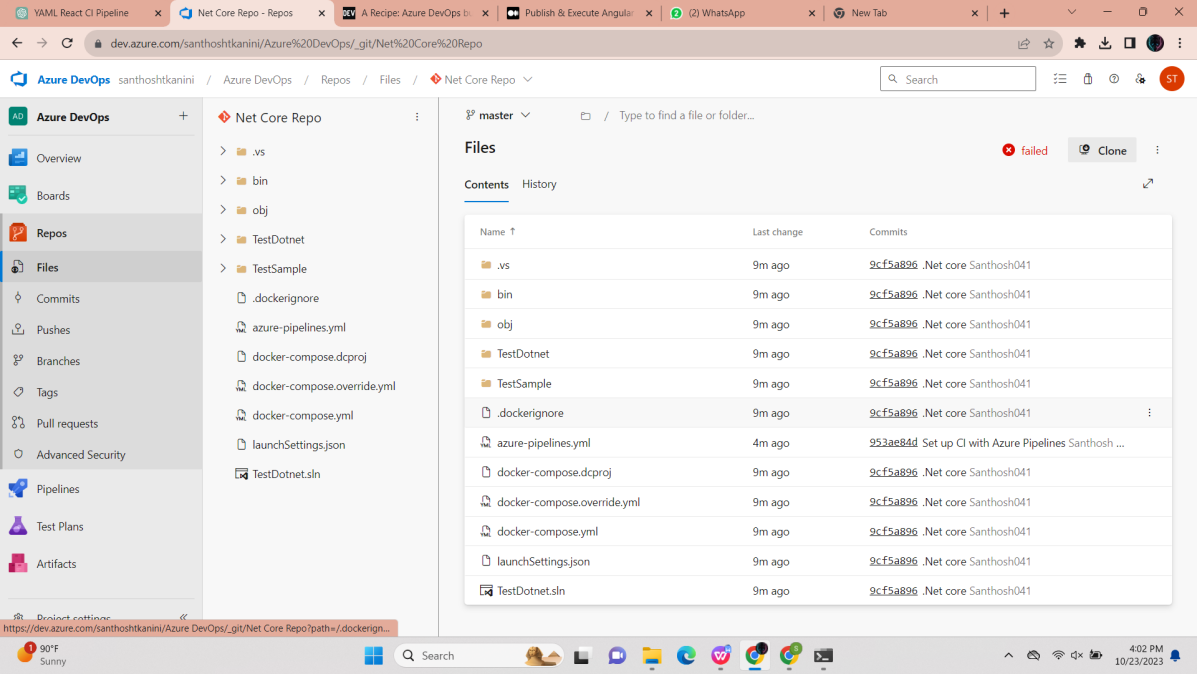
Azure pipeline for React :

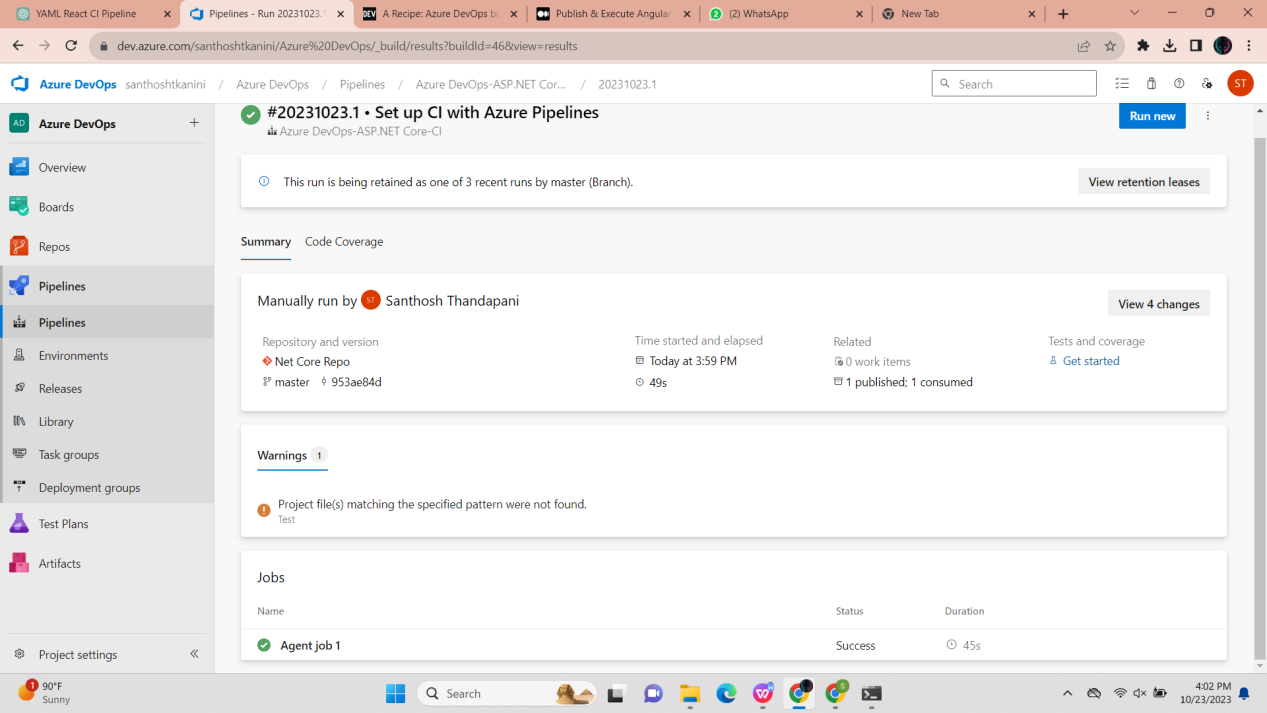
****

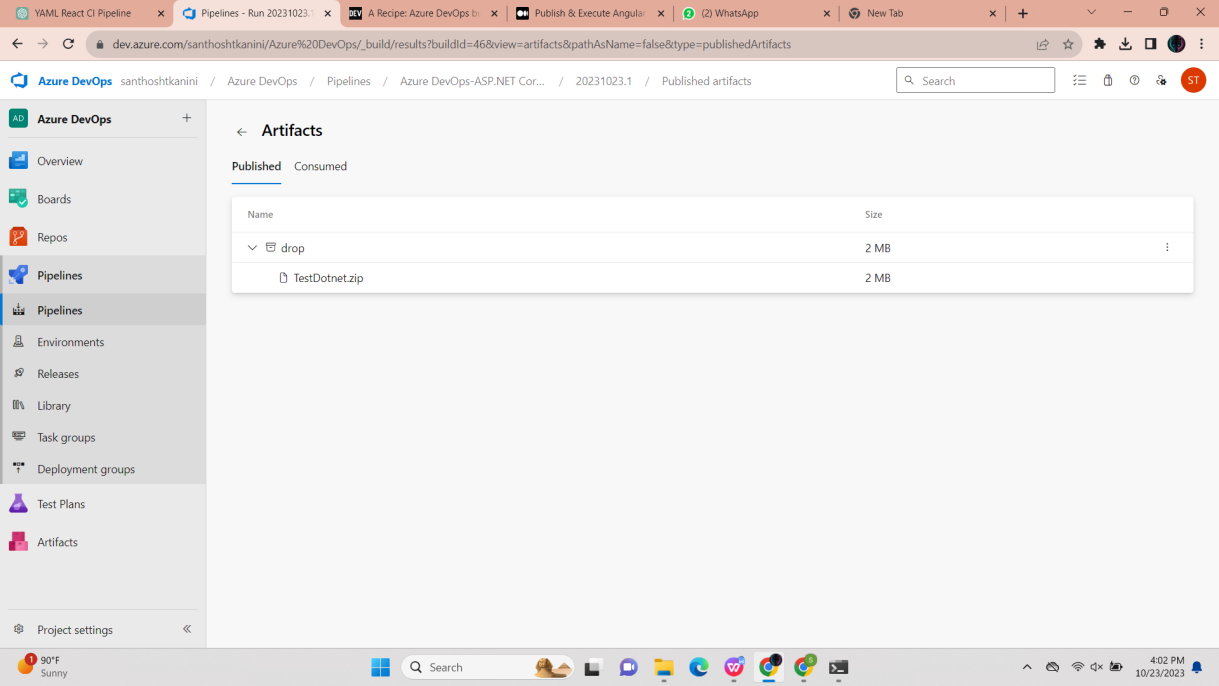
Published Artifacts : ****

Agents : ****

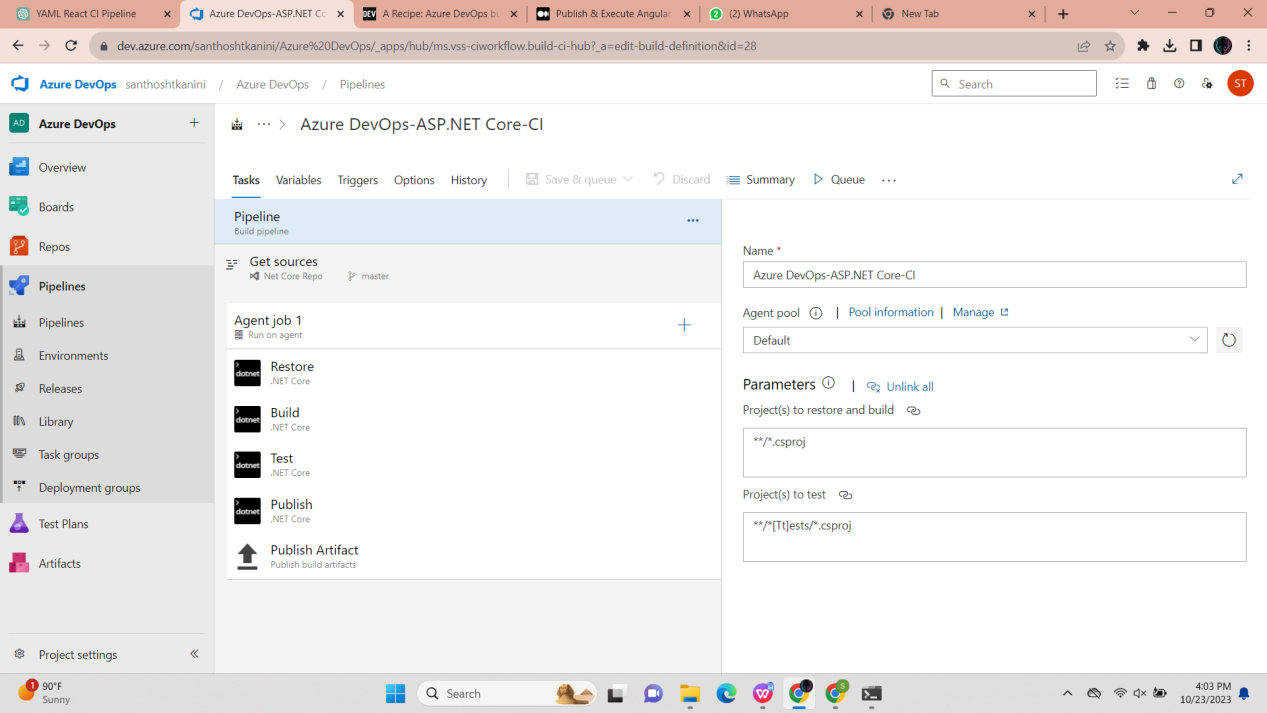
**Lab 9: Create CI Pipeline for .NET Core Application with MS Unit Test**

Azure repo for .Net Core Wep API ****

Pipeline for .Net Core: ****

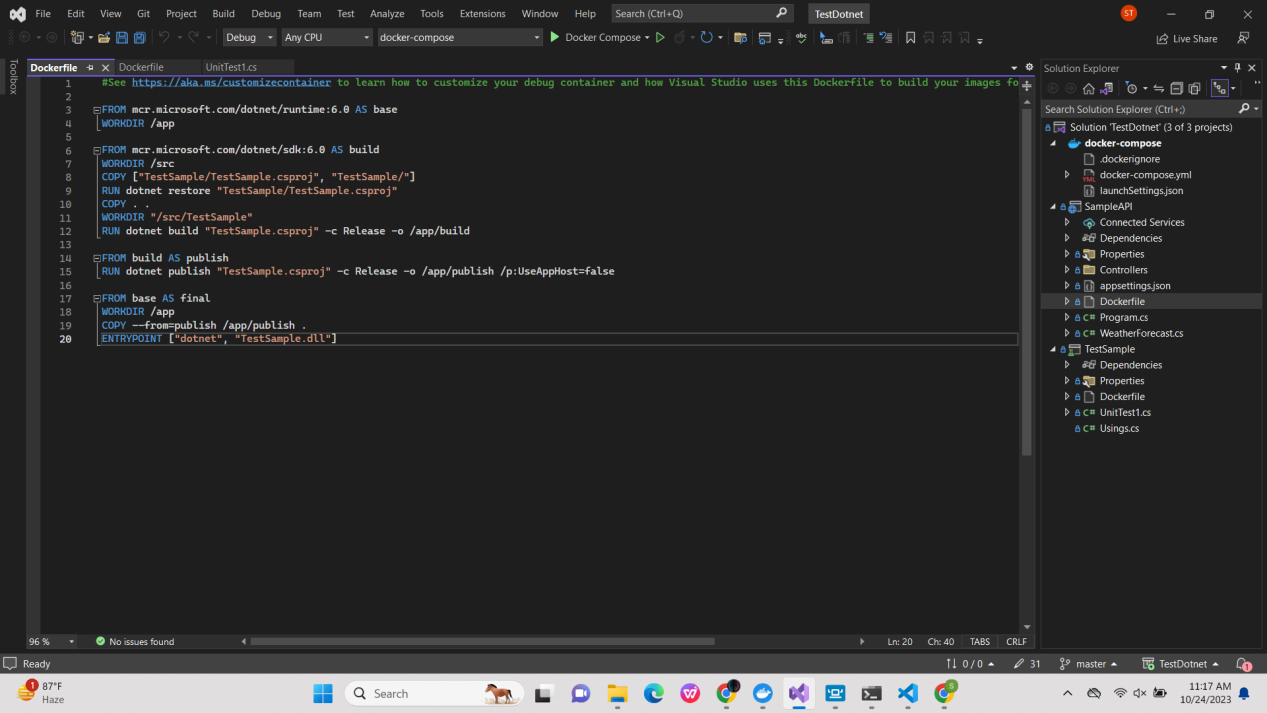
Published Artifacts :  ****

**Agents :**

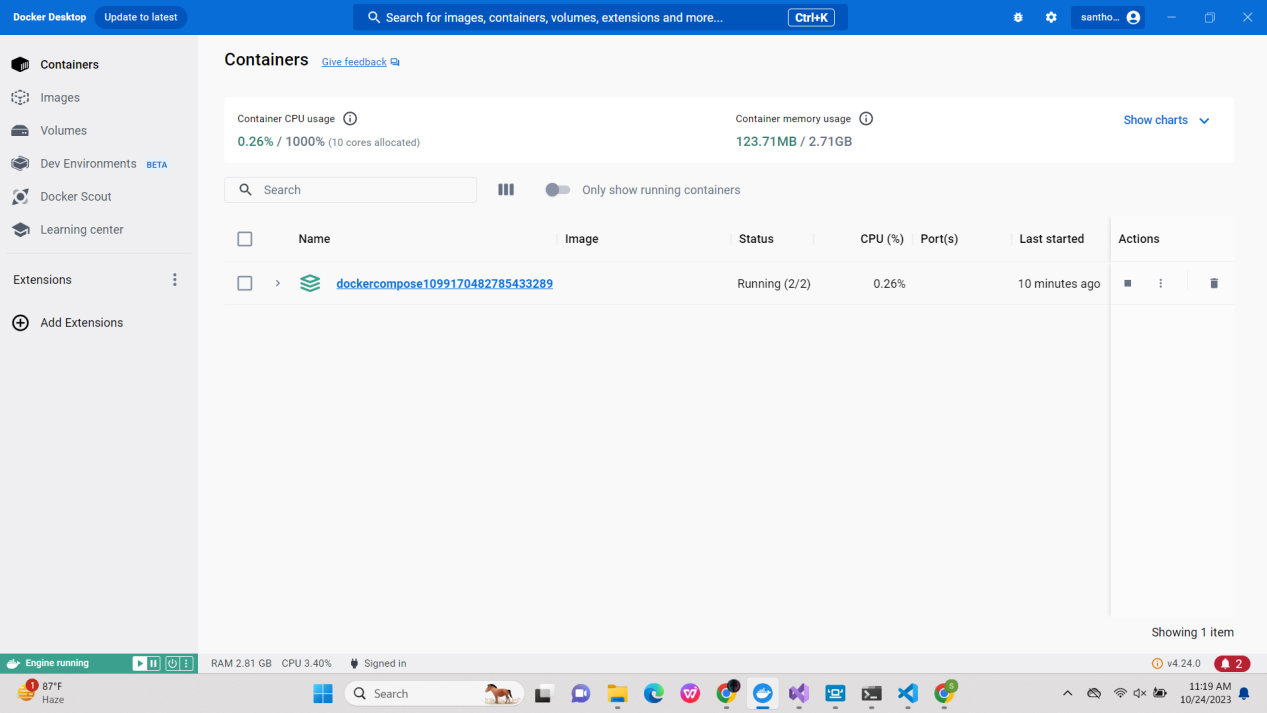
****

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

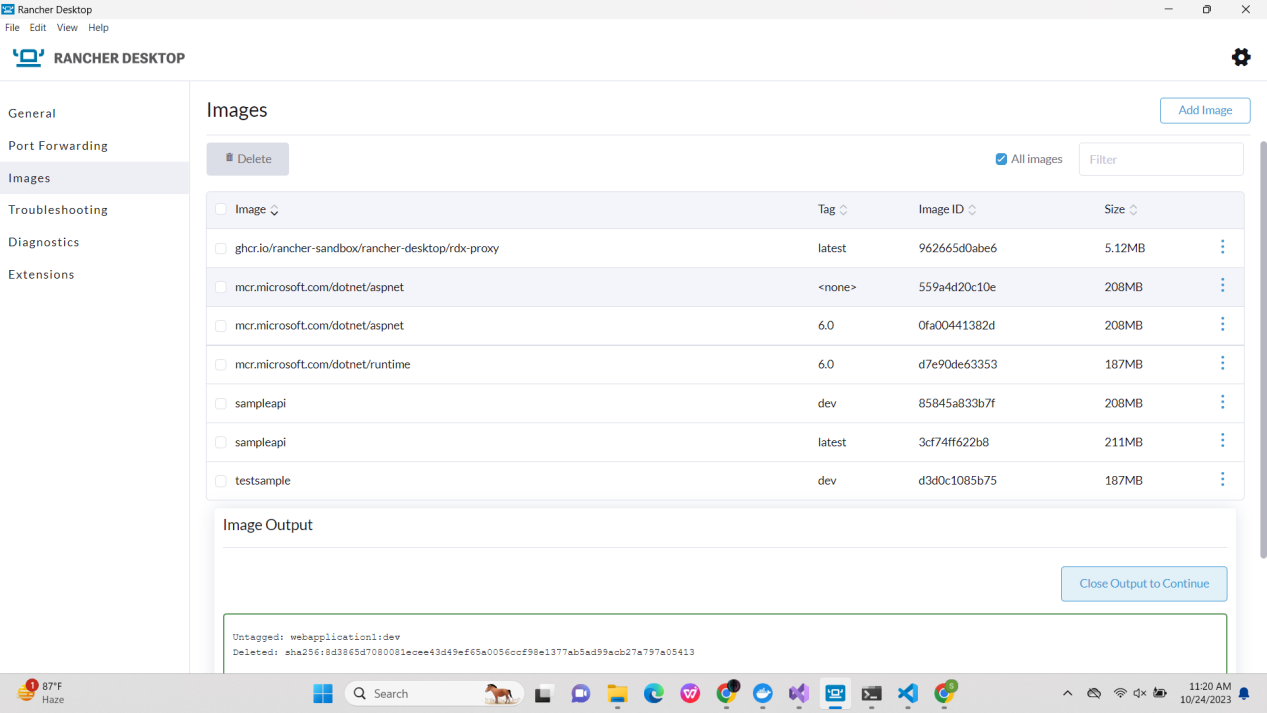
Docker file for .Net Core Web API :

****

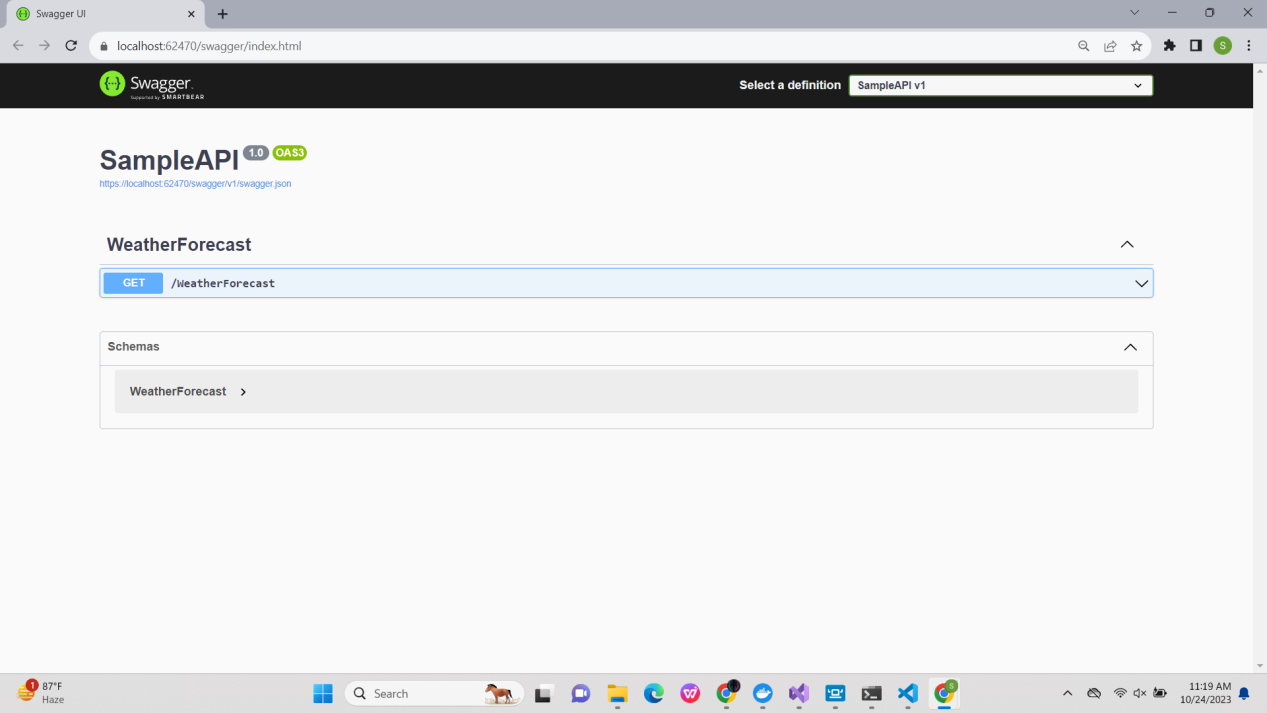
Container is running on the Docker Desktop :

****

Rancher Desktop:

****

Project is running on swagger:

****