DevOps Final Assessment

Section-1: MCQ

1.C

2.B

3.B

4.C

5.C

6.B

7.C

8.B

9.C

10.C

Section-2: Labs

1. Objective: Practice basic file and directory management commands.

Tasks:

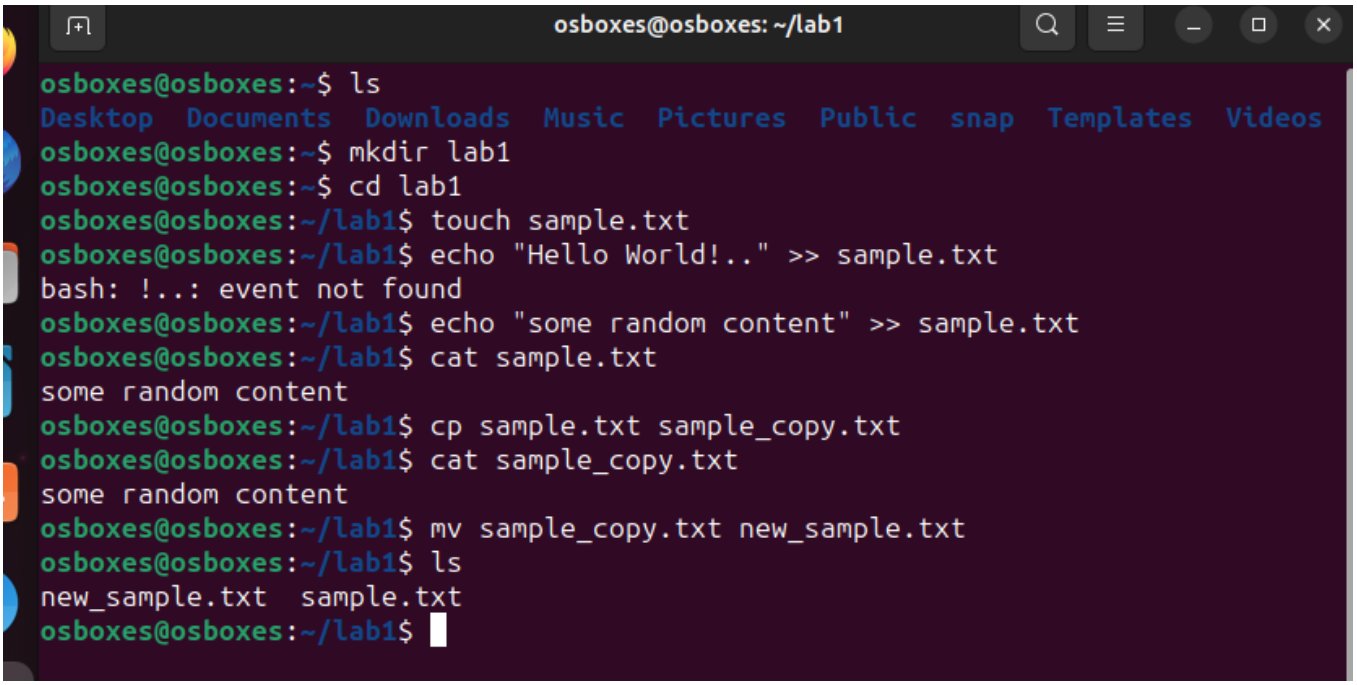
1. Create a directory called "lab1" in your home directory.

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.



2. 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.

```
osboxes@osboxes: ~/lab2
osboxes@osboxes:~/lab2$ touch secret.txt
osboxes@osboxes:~/lab2$ ls -l
total 0
-rw-rw-r-- 1 osboxes osboxes 0 Oct 20 01:36 secret.txt
osboxes@osboxes:~/lab2$ chmod g-r secret.txt
osboxes@osboxes:~/lab2$ chmod g-w secret.txt
osboxes@osboxes:~/lab2$ chmod o-r secret.txt
osboxes@osboxes:~/lab2$ ls -l
total 0
-rw------- 1 osboxes osboxes 0 Oct 20 01:36 secret.txt
osboxes@osboxes:~/lab2$ sudo chown rajnesh secret.txt
chown: invalid user: ‘rajnesh’
osboxes@osboxes:~/lab2$ sudo user add rajnesh
sudo: user: command not found
osboxes@osboxes:~/lab2$ sudo useradd rajnesh
osboxes@osboxes:~/lab2$ sudo chown rajnesh secret.txt
osboxes@osboxes:~/lab2$ ls -l secret.txt
-rw------- 1 rajnesh osboxes 0 Oct 20 01:36 secret.txt
osboxes@osboxes:~/lab2$ ls -n secret.txt
-rw------- 1 1001 1000 0 Oct 20 01:36 secret.txt
osboxes@osboxes:~/lab2$
```

3. 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.

```
osboxes@osboxes: ~/lab3
osboxes@osboxes:~/lab3$ touch cricket.txt
osboxes@osboxes:~/lab3$ echo "Sachin, Dhoni, Kholi, Bumrah" >> cricket.txt
osboxes@osboxes:~/lab3$ grep Dhoni cricket.txt
Sachin, Dhoni, Kholi, Bumrah
osboxes@osboxes:~/lab3$ sed -i 's/Sachin/Rohit/g' cricket.txt
osboxes@osboxes:~/lab3$ cat cricket.txt
Rohit, Dhoni, Kholi, Bumrah
osboxes@osboxes:~/lab3$ wc -c cricket.txt
28 cricket.txt
osboxes@osboxes:~/lab3$ wc -w cricket.txt
4 cricket.txt
osboxes@osboxes:~/lab3$ wc -l cricket.txt
1 cricket.txt
osboxes@osboxes:~/lab3$
```

4. 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.

```
osboxes@osboxes:~/lab4$ touch config.yaml
osboxes@osboxes:~/lab4$ sudo vim config.yaml
osboxes@osboxes:~/lab4$ yamllint config.yaml
osboxes@osboxes:~/lab4$
```

```
osboxes@osboxes: ~/lab4
---
name: SkillMetrics
version: 1.1
description: Describes Skill Levels of all the employees
~
~
~
~
~
~
```

5.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.

```
---
favourite_fruits:
  - Mango
  - Orange
  - Banana
~
~
~
~
~
~
```

```
osboxes@osboxes:~/lab5$ touch fruits.yaml
osboxes@osboxes:~/lab5$ sudo vim fruits.yaml
osboxes@osboxes:~/lab5$ yamllint fruits.yaml
osboxes@osboxes:~/lab5$
```

6. 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.

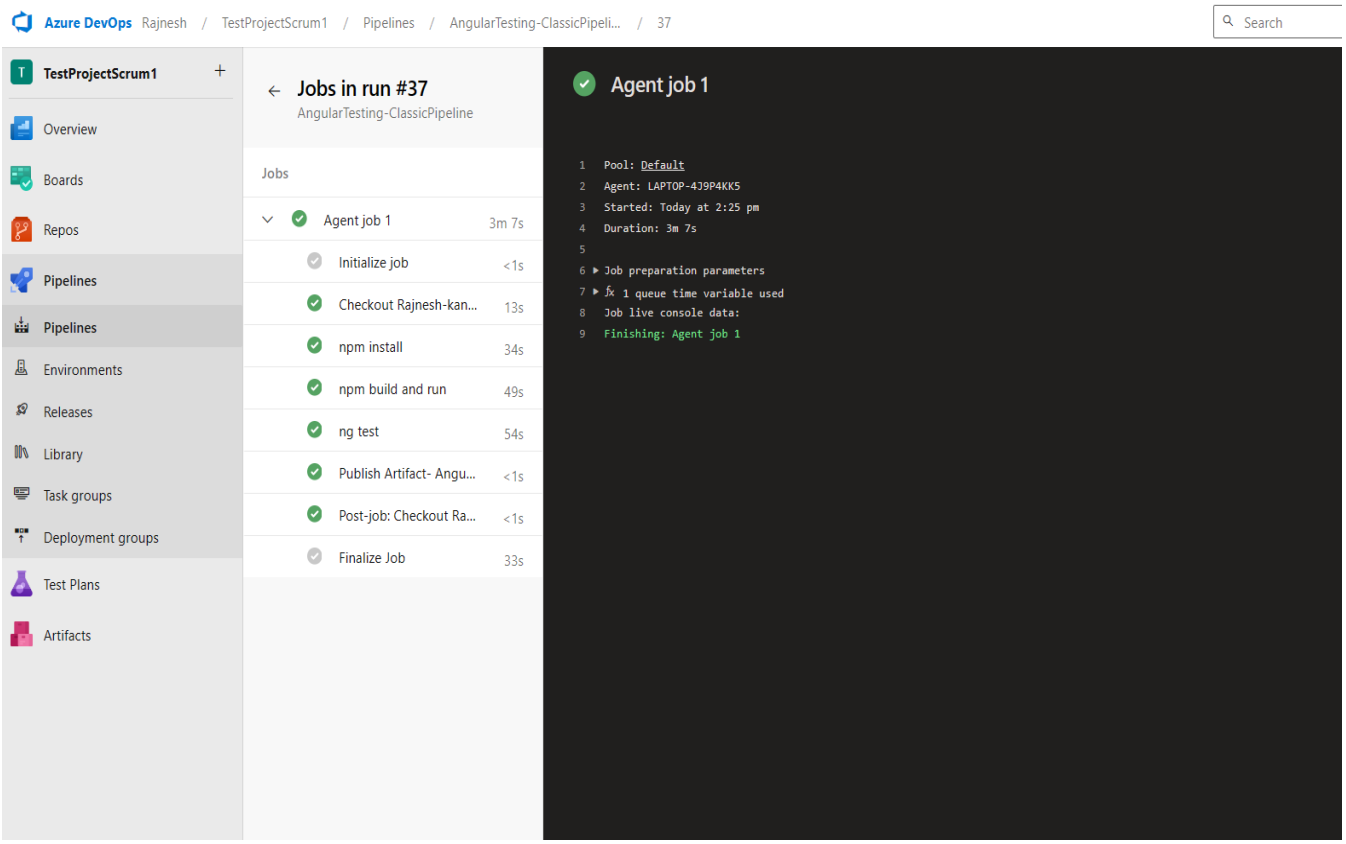
3. Use YAML syntax to add, update, or remove data within the nested structure.

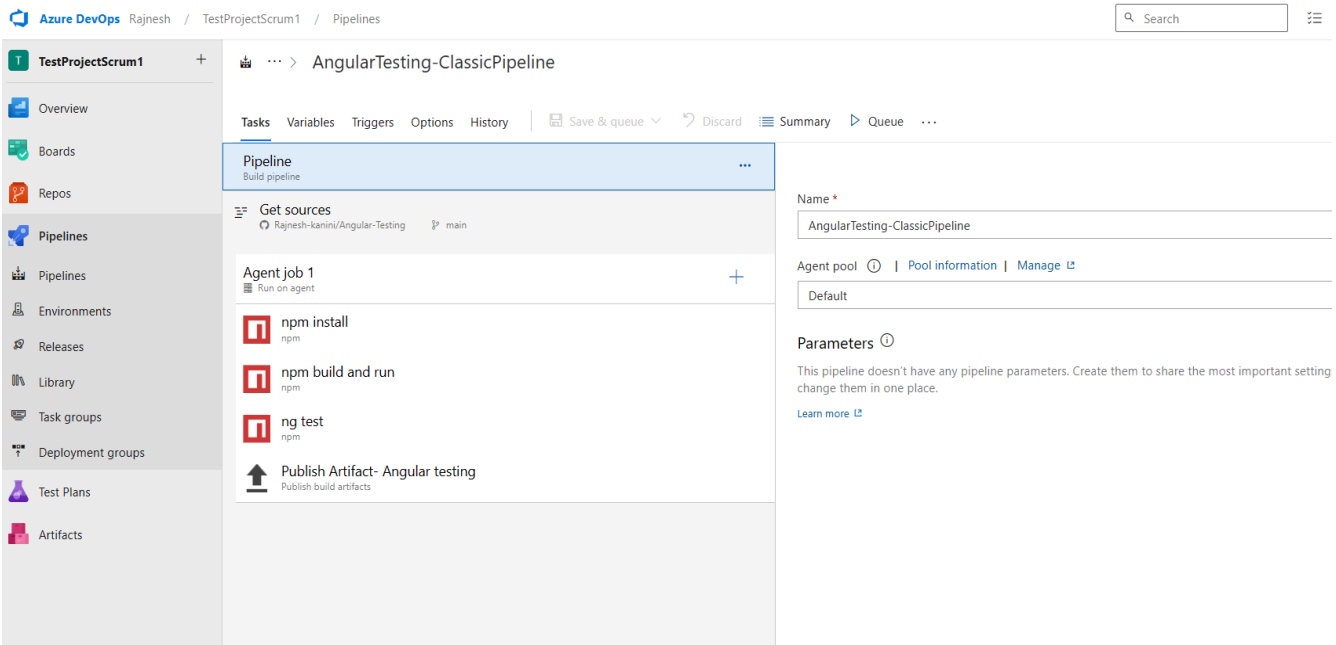
4. Save and validate the YAML file.

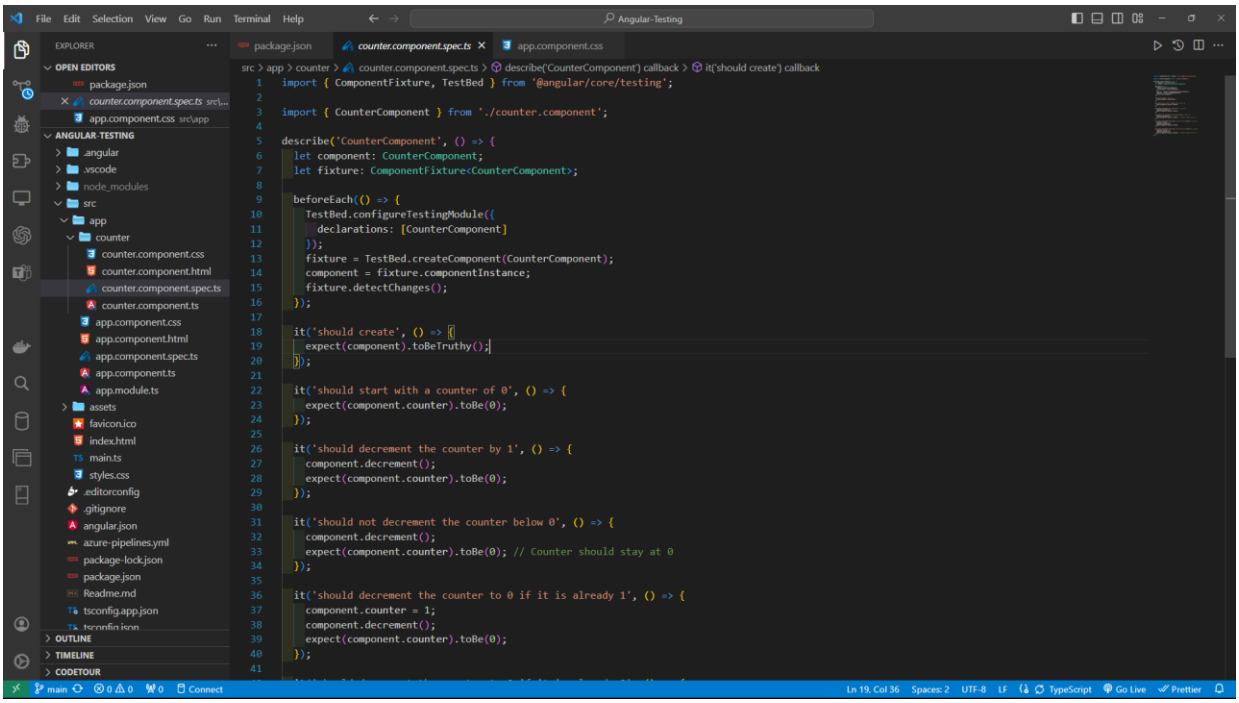
```
osboxes@osboxes:~/lab6$ touch data.yaml
osboxes@osboxes:~/lab6$ sudo vim data.yaml
osboxes@osboxes:~/lab6$ yamllint data.yaml
osboxes@osboxes:~/lab6$
```

```yaml
---
organization:
  - department:
      department_name: Developer
      employees:
        - name: Rajnesh
          role: Associate
        - name: Rajeesh
          role: Trainee
  - department:
      department_name: Data Analyst
      employees:
        - name: Jeeva
          role: Associate
        - name: Nandha
          role: Trainee
~
~
~
~
```

7.Angular CI with Classic Pipeline. Testing- Jasmine

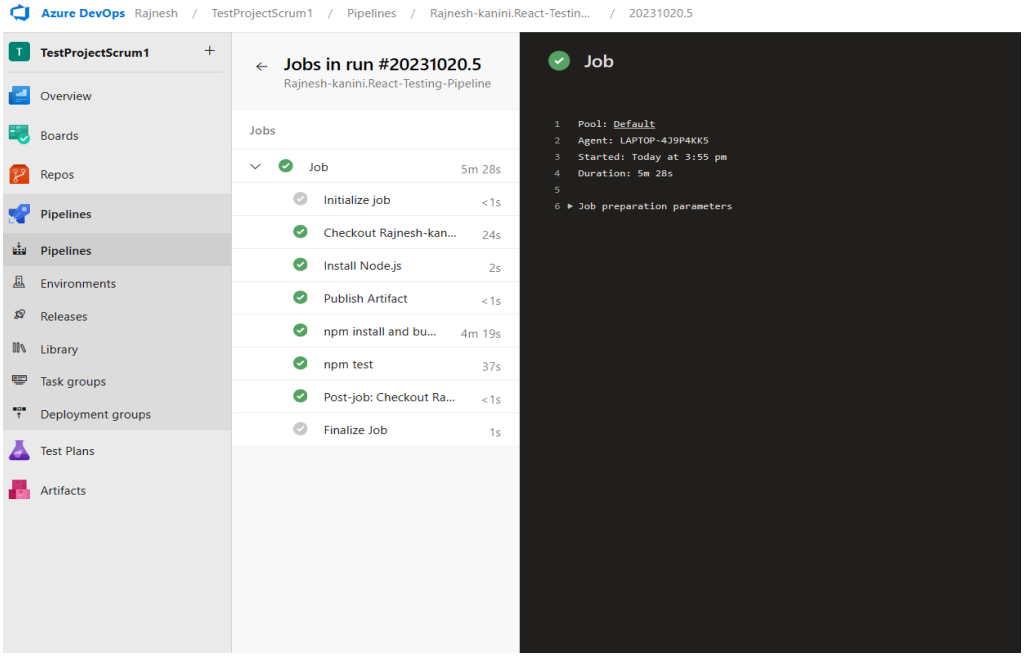




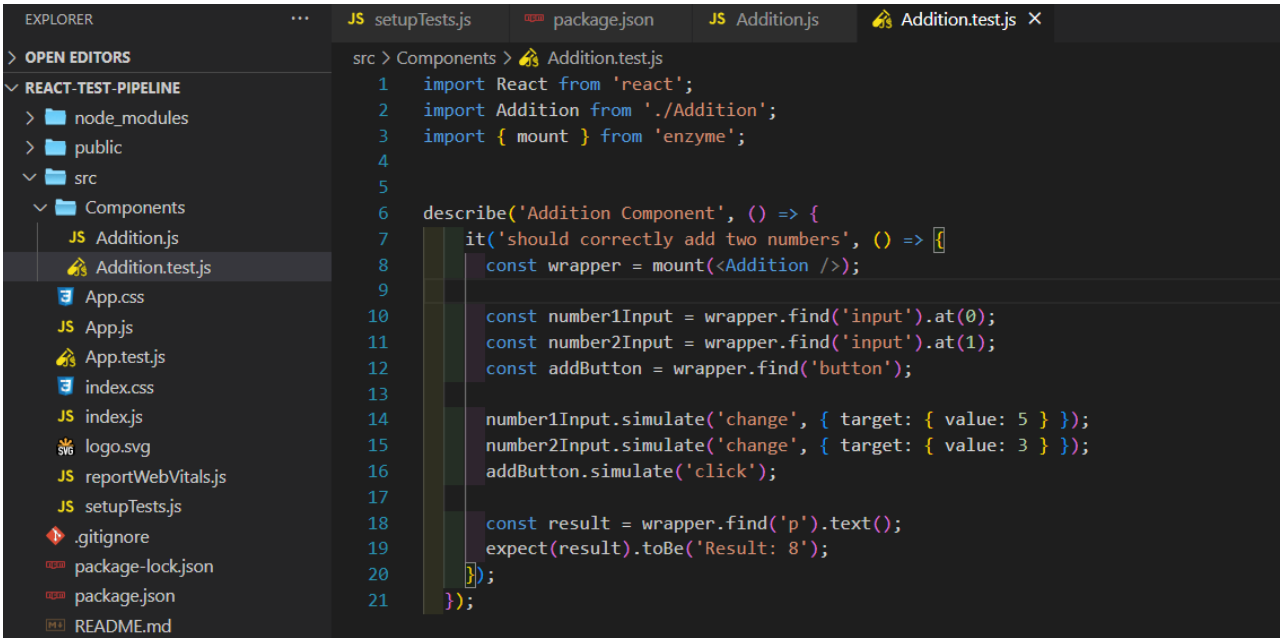


Above is the sample test in Counter Component

8.React CI with Pipeline. Testing-Enzyme

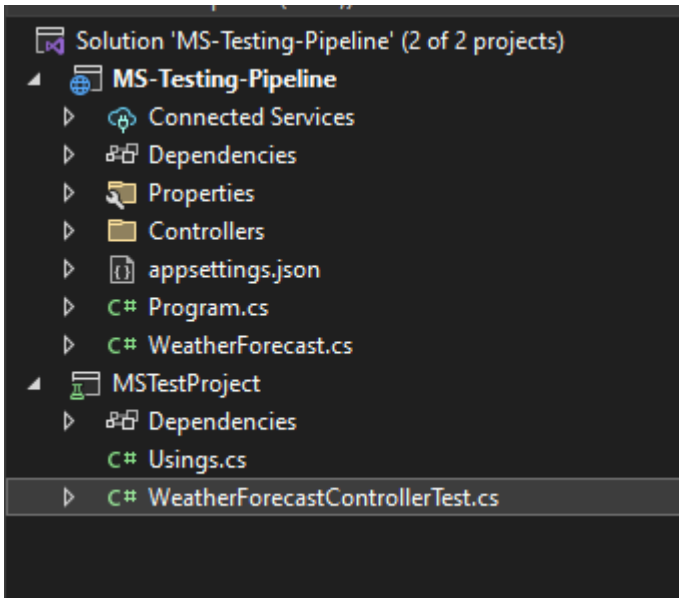


```yaml
# Node.js with React
# Build a Node.js project that uses React.
# Add steps that analyze code, save build artifacts, deploy, and more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/javascri

trigger:
- main

pool:
  #vmImage: ubuntu-latest
  name: Default

steps:
Settings
- task: NodeTool@0
  inputs:
    versionSpec: '10.x'
  displayName: 'Install Node.js'

Settings
- task: PublishBuildArtifacts@1
  displayName: 'Publish Artifact'
  inputs:
    ArtifactName: ReactTestProject

- script: |
    npm install
    npm run build
  displayName: 'npm install and build'

- script: |
    npm install test
    npm test
  displayName: 'npm test'
```




```python
import React from 'react';
import Addition from './Addition';
import { mount } from 'enzyme';


describe('Addition Component', () => {
    it('should correctly add two numbers', () => {
      const wrapper = mount(<Addition />);

      const number1Input = wrapper.find('input').at(0);
      const number2Input = wrapper.find('input').at(1);
      const addButton = wrapper.find('button');

      number1Input.simulate('change', { target: { value: 5 } });
      number2Input.simulate('change', { target: { value: 3 } });
      addButton.simulate('click');

      const result = wrapper.find('p').text();
      expect(result).toBe('Result: 8');
    });
  });
```

9.DOT-NET CORE WITH MS TEST




```
using Microsoft.Extensions.Logging;
using Moq;
using MS_Testing_Pipeline;
using MS_Testing_Pipeline.Controllers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MSTestProject
{
    [TestClass]
    0 references
    public class WeatherForecastControllerTest
    {
        [TestMethod]
        0 references
        public void Get_ReturnsWeatherForecastData()
        {
            // Arrange
            var logger = Mock.Of<ILogger<WeatherForecastController>>();
            var controller = new WeatherForecastController(logger);

            // Act
            var result = controller.Get();

            // Assert
            Assert.IsNotNull(result);
            Assert.IsInstanceOfType(result, typeof(IEnumerable<WeatherForecast>));
            Assert.AreEqual(5, result.Count()); // We expect 5 items as defined in the controller
        }
    }
}
```

Rajnesh-kanini..Net-MSTest-Pipeline

main

Rajnesh-kanini/.Net-MSTest-Pipeline / **azure-pipelines.yml**

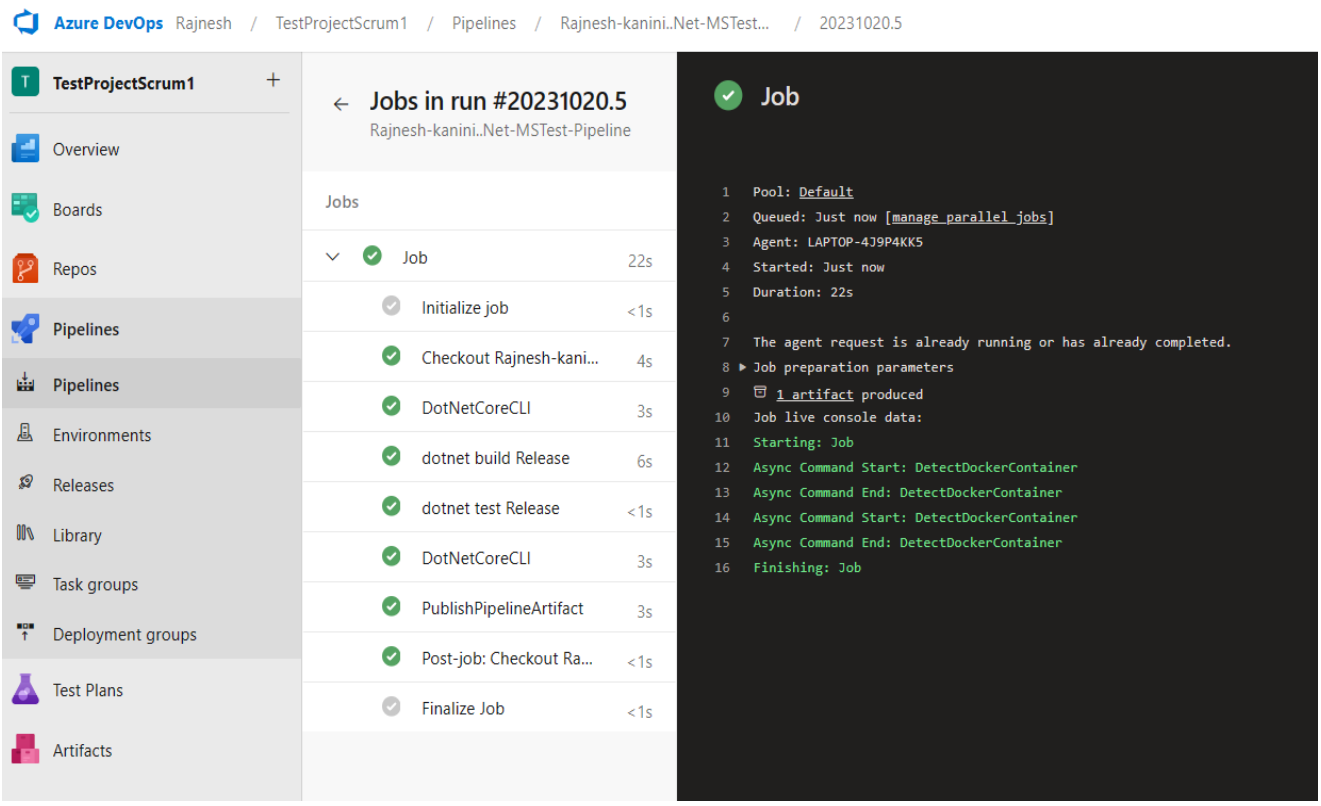
```yaml
# ASP.NET Core (.NET Framework)
# Build and test ASP.NET Core projects targeting the full .NET Framework.
# Add steps that publish symbols, save build artifacts, and more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/dotnet-core

trigger:
- main

pool:
  #vmImage: 'windows-latest'
  name: Default

variables:
  buildConfiguration: 'Release'

steps:
Settings
- task: DotNetCoreCLI@2
  inputs:
    command: 'restore'
    feedsToUse: 'select'
    vstsFeed: 'my-vsts-feed' # A series of numbers and letters

Settings
- task: DotNetCoreCLI@2
  inputs:
    command: 'build'
    arguments: '--configuration $(buildConfiguration)'
  displayName: 'dotnet build $(buildConfiguration)'
Settings
- task: DotNetCoreCLI@2
  inputs:
    command: test
    projects: '**/WeatherForecastControllerTest.csproj'
    arguments: '--configuration $(buildConfiguration)'
  displayName: 'dotnet test $(buildConfiguration)'
```

Rajnesh-kanini..Net-MSTest-Pipeline

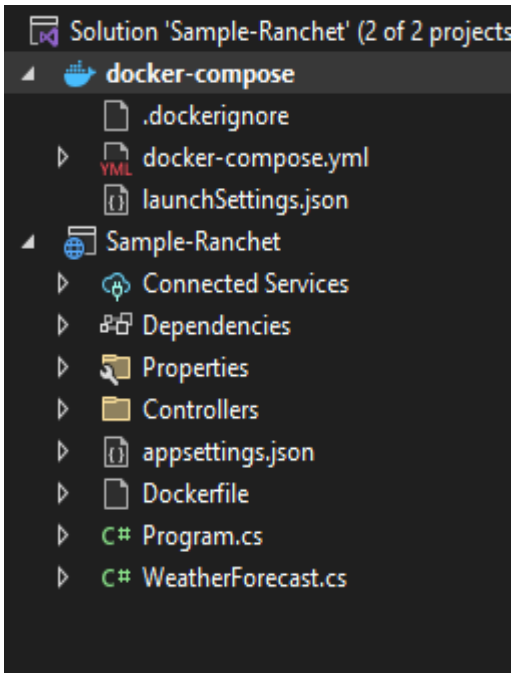
main

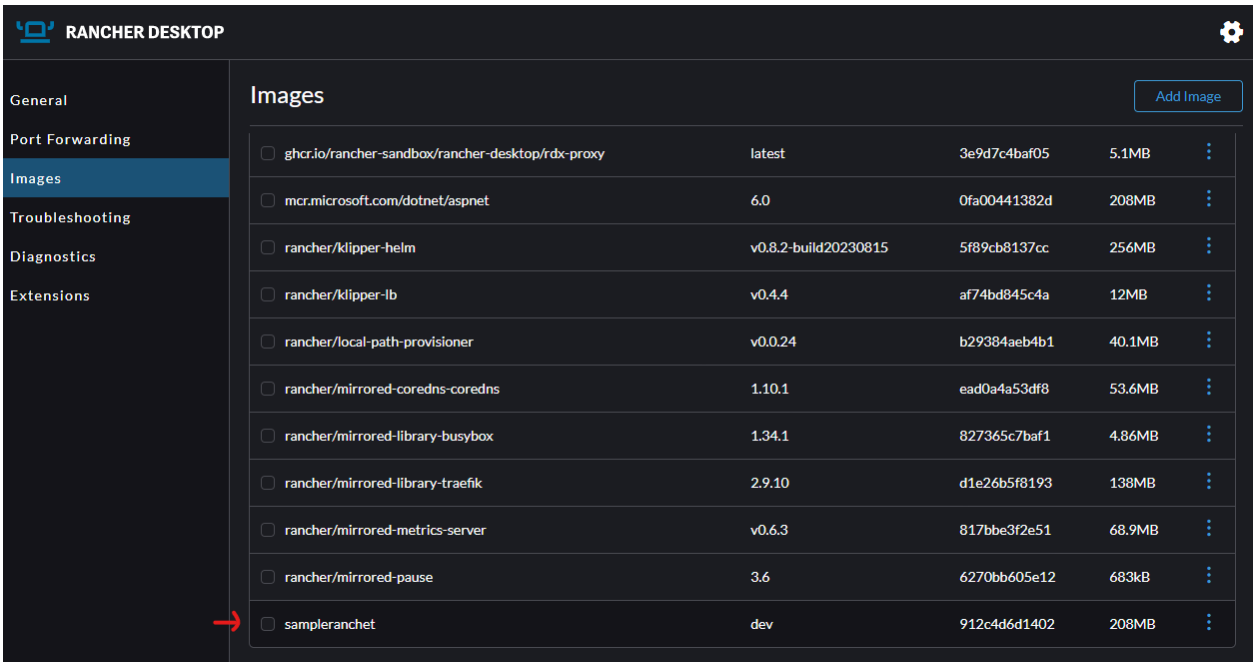
Rajnesh-kanini/.Net-MSTest-Pipeline / **azure-pipelines.yml**

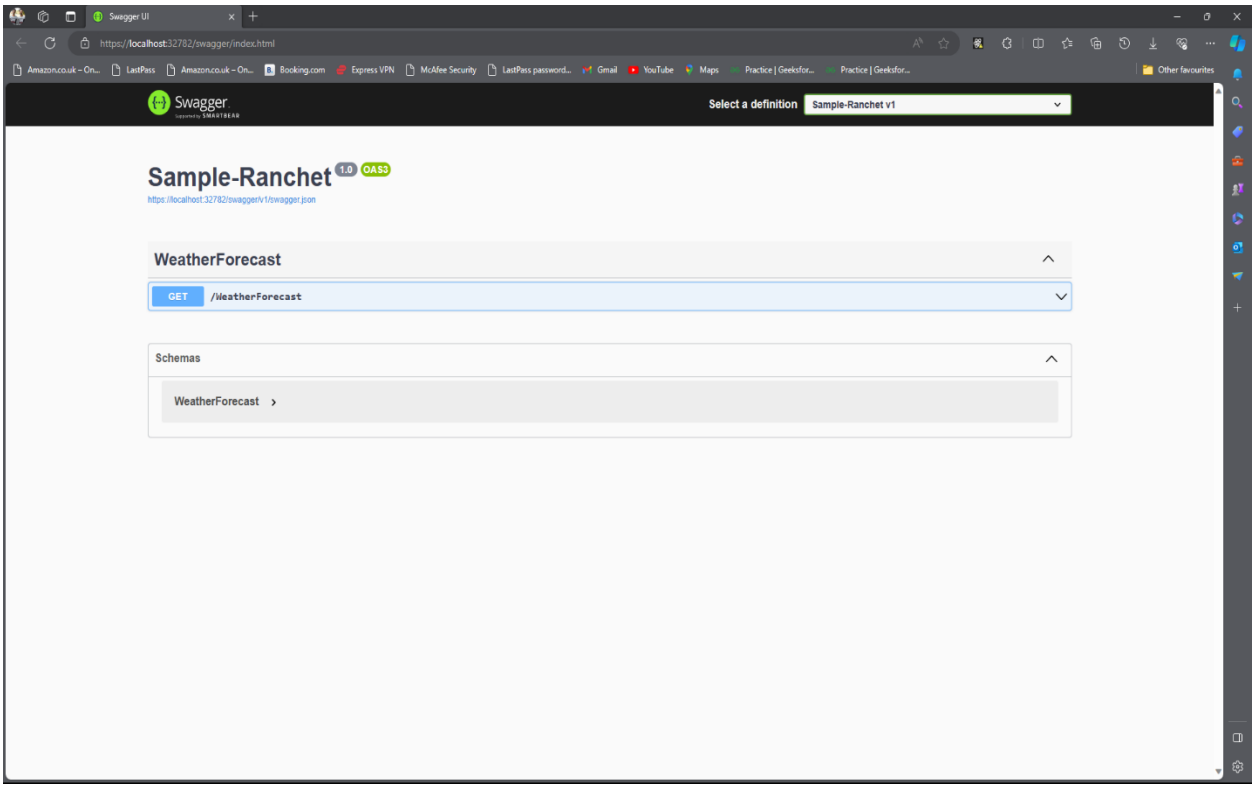
```yaml
    feedsToUse: 'select'
    vstsFeed: 'my-vsts-feed' # A series of numbers and letters

Settings
- task: DotNetCoreCLI@2
  inputs:
    command: 'build'
    arguments: '--configuration $(buildConfiguration)'
  displayName: 'dotnet build $(buildConfiguration)'
Settings
- task: DotNetCoreCLI@2
  inputs:
    command: test
    projects: '**/WeatherForecastControllerTest.csproj'
    arguments: '--configuration $(buildConfiguration)'
  displayName: 'dotnet test $(buildConfiguration)'

Settings
- task: DotNetCoreCLI@2
  inputs:
    command: publish
    publishWebProjects: True
    arguments: '--configuration $(BuildConfiguration) --output $(Build.ArtifactStagingDirectory)'
    zipAfterPublish: True

# this code takes all the files in $(Build.ArtifactStagingDirectory) and uploads them as an artifact of your build.
Settings
- task: PublishPipelineArtifact@1
  inputs:
    targetPath: '$(Build.ArtifactStagingDirectory)'
    artifactName: 'MSTest-Publish-To-Pipline'
```



10. Sample .NET project is containerized and that image is running on Rancher Desktop




RANCHER DESKTOP
General
Port Forwarding
Images
Troubleshooting
Diagnostics
Extensions
Images
Add Image
ghcr.io/rancher-sandbox/rancher-desktop/rdx-proxy latest 3e9d7c4baf05 5.1MB
mcr.microsoft.com/dotnet/aspnet 6.0 0fa00441382d 208MB
rancher/klipper-helm v0.8.2-build20230815 5f89cb8137cc 256MB
rancher/klipper-lb v0.4.4 af74bd845c4a 12MB
rancher/local-path-provisioner v0.0.24 b29384aeb4b1 40.1MB
rancher/mirrored-coredns-coredns 1.10.1 ead0a4a53df8 53.6MB
rancher/mirrored-library-busybox 1.34.1 827365c7baf1 4.86MB
rancher/mirrored-library-traefik 2.9.10 d1e26b5f8193 138MB
rancher/mirrored-metrics-server v0.6.3 817bbe3f2e51 68.9MB
rancher/mirrored-pause 3.6 6270bb605e12 683kB
sampleranchet dev 912c4d6d1402 208MB


Swagger UI
https://localhost:32782/swagger/index.html
Swagger
Select a definition
Sample-Ranchet v1
Sample-Ranchet
1.0
OAS3
https://localhost:32782/swagger/v1/swagger.json
WeatherForecast
GET
/WeatherForecast
Schemas
WeatherForecast