

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

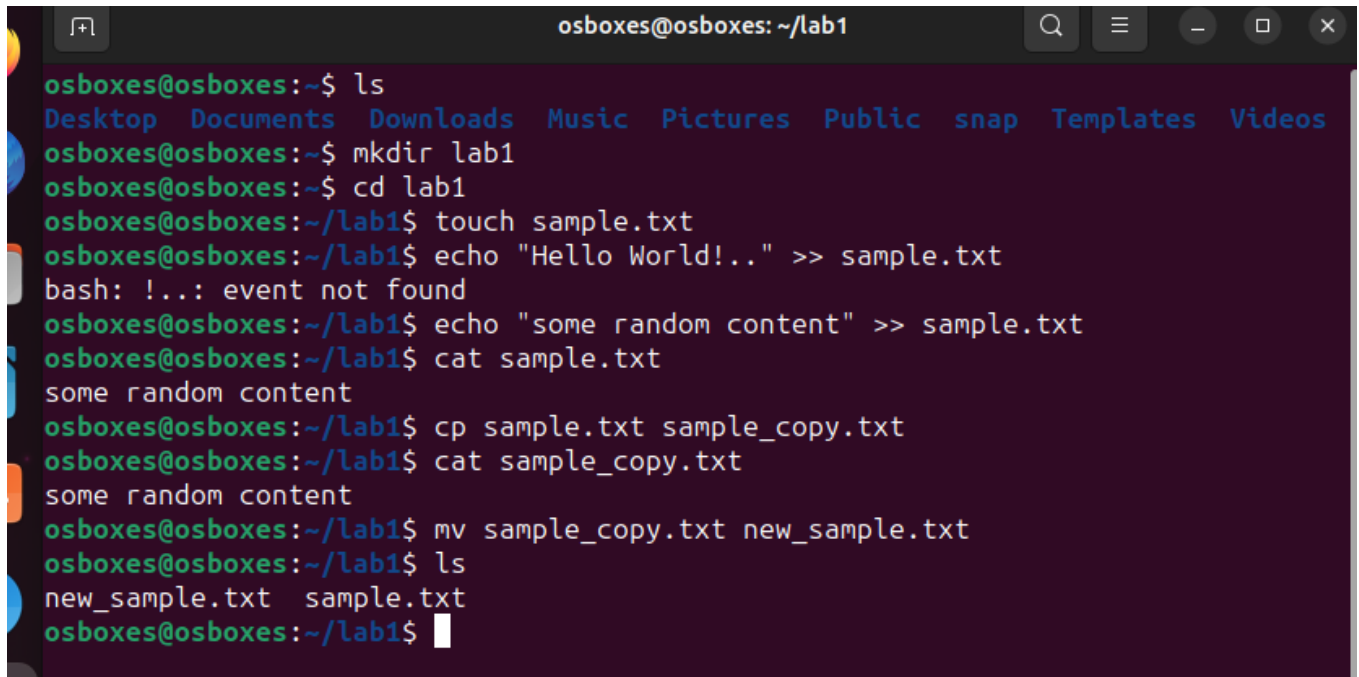
Section-2: Labs

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.

A terminal window titled 'osboxes@osboxes: ~/lab1' with standard window controls. The terminal shows a sequence of commands: 'ls' (listing desktop files), 'mkdir lab1' (creating the directory), 'cd lab1' (changing to the directory), 'touch sample.txt' (creating a file), 'echo "Hello World!.." >> sample.txt' (adding content), 'echo "some random content" >> sample.txt' (adding more content), 'cat sample.txt' (displaying content), 'cp sample.txt sample_copy.txt' (creating a copy), 'cat sample_copy.txt' (displaying the copy), 'mv sample_copy.txt new_sample.txt' (renaming the copy), and 'ls' (listing the directory). The final output shows 'new_sample.txt' and 'sample.txt' in the directory.

```
osboxes@osboxes:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  Videos
osboxes@osboxes:~$ mkdir lab1
osboxes@osboxes:~$ cd lab1
osboxes@osboxes:~/lab1$ touch sample.txt
osboxes@osboxes:~/lab1$ echo "Hello World!.." >> sample.txt
bash: !...: event not found
osboxes@osboxes:~/lab1$ echo "some random content" >> sample.txt
osboxes@osboxes:~/lab1$ cat sample.txt
some random content
osboxes@osboxes:~/lab1$ cp sample.txt sample_copy.txt
osboxes@osboxes:~/lab1$ cat sample_copy.txt
some random content
osboxes@osboxes:~/lab1$ mv sample_copy.txt new_sample.txt
osboxes@osboxes:~/lab1$ ls
new_sample.txt  sample.txt
osboxes@osboxes:~/lab1$
```

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 -ln 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, Kohli, Bumrah" >> cricket.txt
osboxes@osboxes:~/lab3$ grep Dhoni cricket.txt
Sachin, Dhoni, Kohli, Bumrah
osboxes@osboxes:~/lab3$ sed -i 's/Sachin/Rohit/g' cricket.txt
osboxes@osboxes:~/lab3$ cat cricket.txt
Rohit, Dhoni, Kohli, 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
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$
```

```
---
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

Azure DevOps Rajnesh / TestProjectScrum1 / Pipelines / AngularTesting-ClassicPipeli... / 37

Search

TestProjectScrum1

Overview

Boards

Repos

Pipelines

Pipelines

Environments

Releases

Library

Task groups

Deployment groups

Test Plans

Artifacts

← Jobs in run #37

AngularTesting-ClassicPipeline

Jobs		
Agent job 1	3m 7s	
Initialize job	<1s	
Checkout Rajnesh-kan...	13s	
npm install	34s	
npm build and run	49s	
ng test	54s	
Publish Artifact- Angu...	<1s	
Post-job: Checkout Ra...	<1s	
Finalize Job	33s	

✓ Agent job 1

```
1 Pool: Default
2 Agent: LAPTOP-439P4KK5
3 Started: Today at 2:25 pm
4 Duration: 3m 7s
5
6 Job preparation parameters
7 ▶ fx 1 queue time variable used
8 Job live console data:
9 Finishing: Agent job 1
```

Azure DevOps Rajnesh / TestProjectScrum1 / Pipelines

Search

TestProjectScrum1

Overview

Boards

Repos

Pipelines

Pipelines

Environments

Releases

Library

Task groups

Deployment groups

Test Plans

Artifacts

AngularTesting-ClassicPipeline

Tasks Variables Triggers Options History

Save & queue Discard Summary Queue

Pipeline

Build pipeline

Get sources

Rajnesh-kamini/Angular-Testing main

Agent job 1

Run on agent

npm install

npm

npm build and run

npm

ng test

npm

Publish Artifact- Angular testing

Publish build artifacts

Name *

AngularTesting-ClassicPipeline

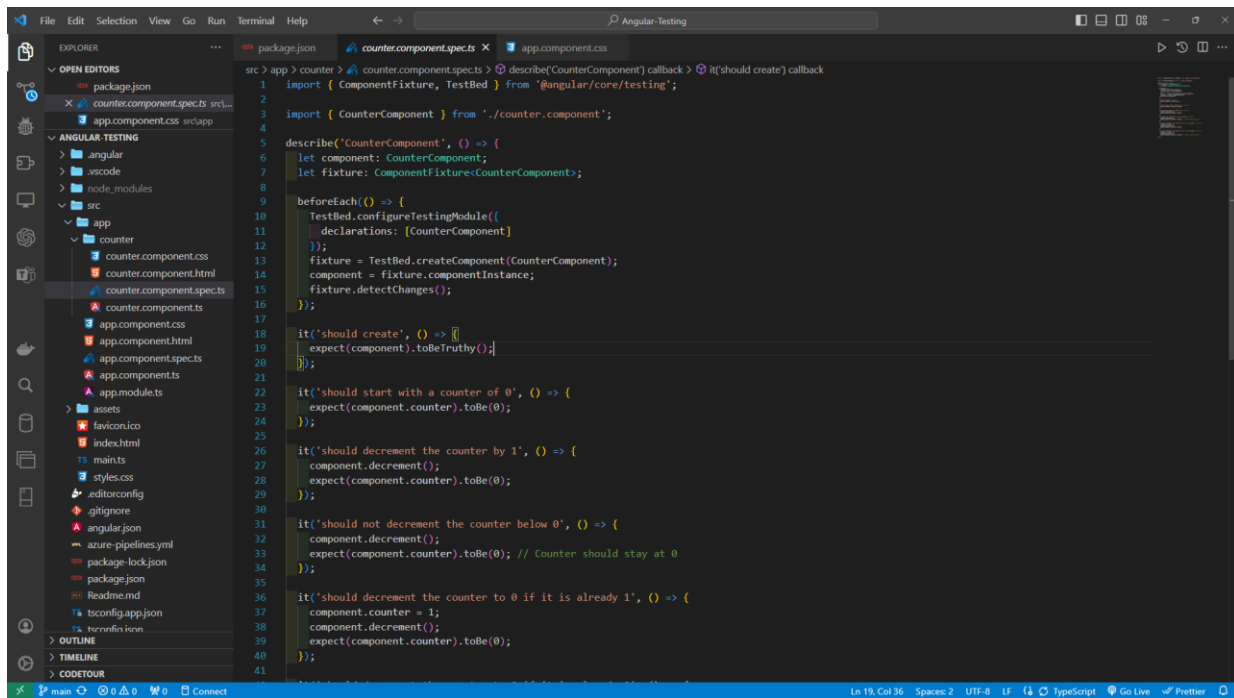
Agent pool ⓘ | Pool information | Manage

Default

Parameters ⓘ

This pipeline doesn't have any pipeline parameters. Create them to share the most important setting change them in one place.

Learn more

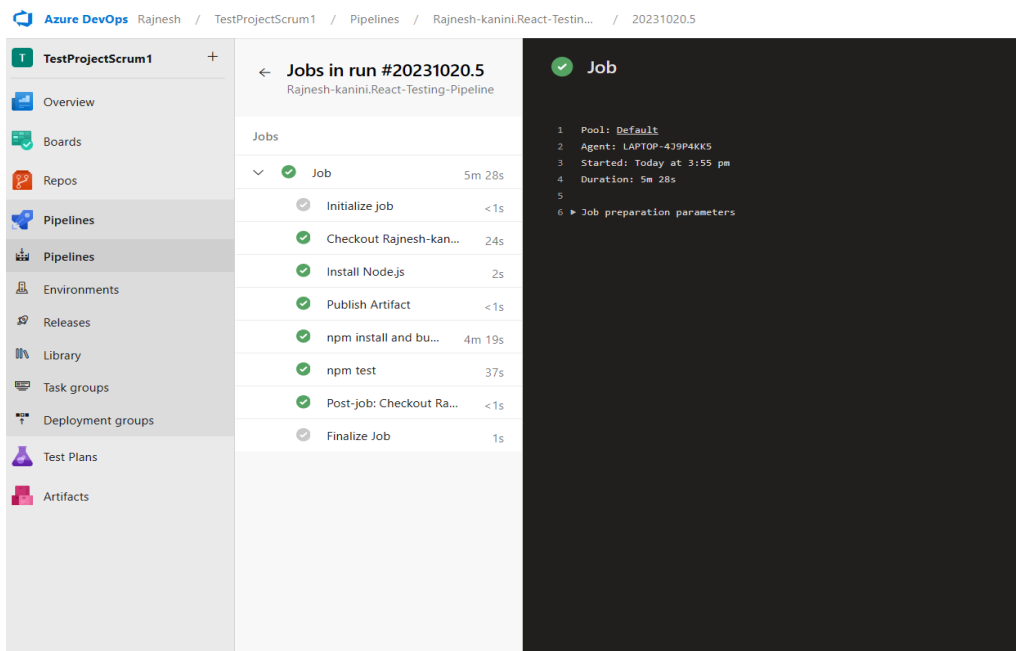


The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'counter' directory containing 'counter.component.spec.ts', 'counter.component.ts', 'counter.component.html', and 'counter.component.css'. The code editor displays the content of 'counter.component.spec.ts', which is a Jest test file for the 'CounterComponent'. The test file includes imports for 'ComponentFixture', 'TestBed', and 'CounterComponent'. It defines a 'describe' block for 'CounterComponent' with several 'it' blocks for testing: 'should create', 'should start with a counter of 0', 'should decrement the counter by 1', 'should not decrement the counter below 0', and 'should decrement the counter to 0 if it is already 1'. The test file also includes a 'beforeEach' block for configuring the testing module.

```
src > app > counter > counter.component.spec.ts > describe('CounterComponent') callback > it('should create') callback
1 import { ComponentFixture, TestBed } from '@angular/core/testing';
2
3 import { CounterComponent } from './counter.component';
4
5 describe('CounterComponent', () => {
6   let component: CounterComponent;
7   let fixture: ComponentFixture<CounterComponent>;
8
9   beforeEach(() => {
10     TestBed.configureTestingModule({
11       declarations: [CounterComponent]
12     });
13     fixture = TestBed.createComponent(CounterComponent);
14     component = fixture.componentInstance;
15     fixture.detectChanges();
16   });
17
18   it('should create', () => {
19     expect(component).toBeTruthy();
20   });
21
22   it('should start with a counter of 0', () => {
23     expect(component.counter).toBe(0);
24   });
25
26   it('should decrement the counter by 1', () => {
27     component.decrement();
28     expect(component.counter).toBe(0);
29   });
30
31   it('should not decrement the counter below 0', () => {
32     component.decrement();
33     expect(component.counter).toBe(0); // Counter should stay at 0
34   });
35
36   it('should decrement the counter to 0 if it is already 1', () => {
37     component.counter = 1;
38     component.decrement();
39     expect(component.counter).toBe(0);
40   });
41
42 })
```

Above is the sample test in Counter Component

8.React CI with Pipeline. Testing-Enzyme



The screenshot shows the Azure DevOps interface for a pipeline named 'Rajesh-kanini.React-Testing-Pipeline'. The pipeline is in a 'Completed' state, and the 'Jobs in run #20231020.5' are listed. The jobs are: 'Initialize job' (5m 28s), 'Checkout Rajesh-kanini...' (24s), 'Install Node.js' (2s), 'Publish Artifact' (<1s), 'npm install and bu...' (4m 19s), 'npm test' (37s), 'Post-job: Checkout Ra...' (<1s), and 'Finalize Job' (1s). The 'Job' is marked as 'Succeeded'.

Jobs in run #20231020.5
Rajesh-kanini.React-Testing-Pipeline

Job	Duration
Initialize job	5m 28s
Checkout Rajesh-kanini...	24s
Install Node.js	2s
Publish Artifact	<1s
npm install and bu...	4m 19s
npm test	37s
Post-job: Checkout Ra...	<1s
Finalize Job	1s

Job
Succeeded

```
1 Pool: Default
2 Agent: LAPTOP-439P4KK5
3 Started: Today at 3:55 pm
4 Duration: 5m 28s
5
6 Job preparation parameters
```



```

1  # Node.js with React
2  # Build a Node.js project that uses React.
3  # Add steps that analyze code, save build artifacts, deploy, and more:
4  # https://docs.microsoft.com/azure/devops/pipelines/languages/javascript
5
6  trigger:
7    - main
8
9  pool:
10   - vmImage: ubuntu-latest
11   - name: Default
12
13  steps:
14    Settings
15    - task: NodeTool@0
16      inputs:
17        - versionSpec: '10.x'
18        - displayName: 'Install Node.js'
19
20    Settings
21    - task: PublishBuildArtifacts@1
22      displayName: 'Publish Artifact'
23      inputs:
24        - ArtifactName: ReactTestProject
25
26    - script: |
27      - npm install
28      - npm run build
29      - displayName: 'npm install and build'
30
31    - script: |
32      - npm install test
33      - npm test
34      - displayName: 'npm test'

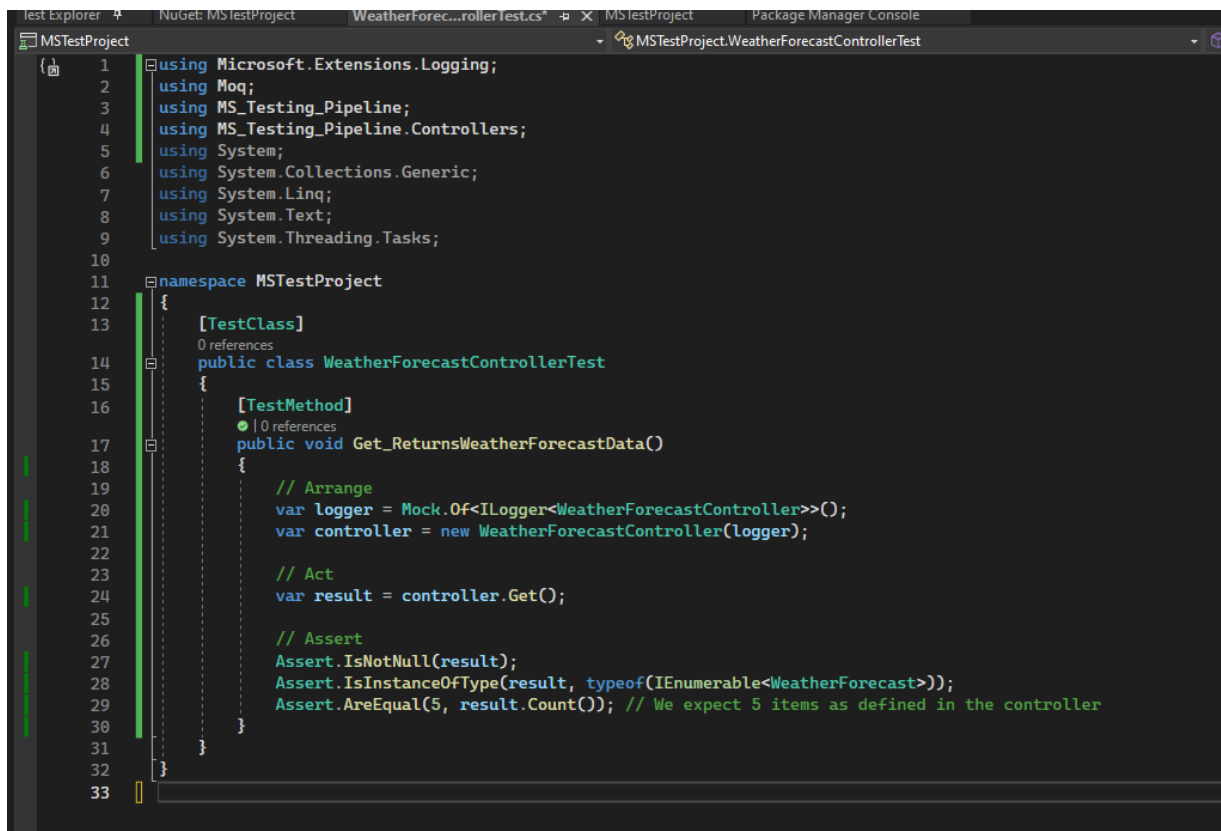
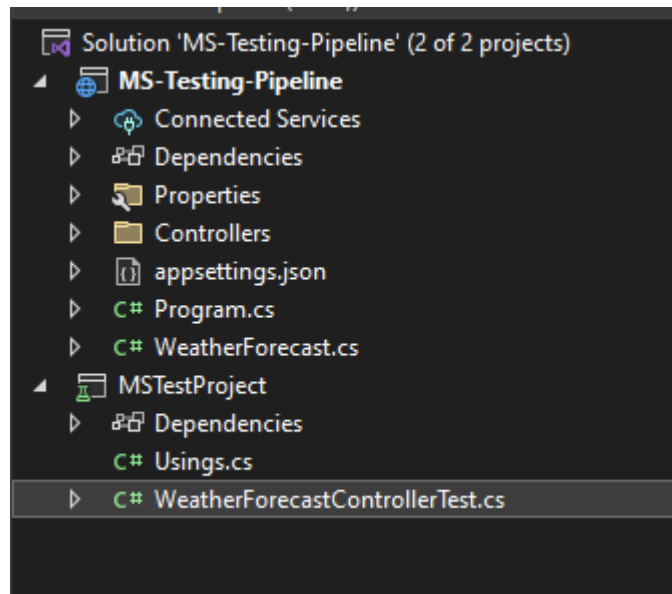
```

```

src > Components > Addition.test.js
1  import React from 'react';
2  import Addition from './Addition';
3  import { mount } from 'enzyme';
4
5
6  describe('Addition Component', () => {
7    it('should correctly add two numbers', () => {
8      const wrapper = mount(<Addition />);
9
10     const number1Input = wrapper.find('input').at(0);
11     const number2Input = wrapper.find('input').at(1);
12     const addButton = wrapper.find('button');
13
14     number1Input.simulate('change', { target: { value: 5 } });
15     number2Input.simulate('change', { target: { value: 3 } });
16     addButton.simulate('click');
17
18     const result = wrapper.find('p').text();
19     expect(result).toBe('Result: 8');
20   });
21 });

```

9.DOT-NET CORE WITH MS TEST



← Rajnesh-kanini..Net-MSTest-Pipeline

main



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

```
1  # ASP.NET Core (.NET Framework)
2  # Build and test ASP.NET Core projects targeting the full .NET Framework.
3  # Add steps that publish symbols, save build artifacts, and more:
4  # https://docs.microsoft.com/azure/devops/pipelines/languages/dotnet-core
5
6  trigger:
7    - main
8
9  pool:
10   - vmImage: 'windows-latest'
11   - name: Default
12
13  variables:
14   - buildConfiguration: 'Release'
15
16  steps:
17    Settings
18    - task: DotNetCoreCLI@2
19      inputs:
20        command: 'restore'
21        feedsToUse: 'select'
22        vstsFeed: 'my-vsts-feed' # A series of numbers and letters
23
24    Settings
25    - task: DotNetCoreCLI@2
26      inputs:
27        command: 'build'
28        arguments: '--configuration $(buildConfiguration)'
29        displayName: 'dotnet build $(buildConfiguration)'
30
31    Settings
32    - task: DotNetCoreCLI@2
33      inputs:
34        command: 'test'
35        projects: '**/WeatherForecastControllerTest.csproj'
36        arguments: '--configuration $(buildConfiguration)'
37        displayName: 'dotnet test $(buildConfiguration)'
```

← Rajnesh-kanini..Net-MSTest-Pipeline

main

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

```
20 | | feedsToUse: select
21 | | vstsFeed: 'my-vsts-feed' # A series of numbers and letters
22
23 | Settings
24 | - task: DotNetCoreCLI@2
25 |   inputs:
26 |     command: 'build'
27 |     arguments: '--configuration $(buildConfiguration)'
28 |     displayName: 'dotnet build $(buildConfiguration)'
29 | Settings
30 | - task: DotNetCoreCLI@2
31 |   inputs:
32 |     command: test
33 |     projects: '**/WeatherForecastControllerTest.csproj'
34 |     arguments: '--configuration $(buildConfiguration)'
35 |     displayName: 'dotnet test $(buildConfiguration)'
36 | Settings
37 | - task: DotNetCoreCLI@2
38 |   inputs:
39 |     command: publish
40 |     publishWebProjects: True
41 |     arguments: '--configuration $(BuildConfiguration) --output $(Build.ArtifactStagingDirectory)'
42 |     zipAfterPublish: True
43 | # this code takes all the files in $(Build.ArtifactStagingDirectory) and uploads them as an artifact of your build.
44 | Settings
45 | - task: PublishPipelineArtifact@1
46 |   inputs:
47 |     targetPath: '$(Build.ArtifactStagingDirectory)'
48 |     artifactName: 'MSTest-Publish-To-Pipeline'
```

TestProjectScrum1

Overview

Boards

Repos

Pipelines

Pipelines

Environments

Releases

Library

Task groups

Deployment groups

Test Plans

Artifacts

Jobs in run #20231020.5

Rajnesh-kanini..Net-MSTest-Pipeline

Jobs

Job

Initialize job

Checkout Rajnesh-kani...

DotNetCoreCLI

dotnet build Release

dotnet test Release

DotNetCoreCLI

PublishPipelineArtifact

Post-job: Checkout Ra...

Finalize Job

Job

1 Pool: Default

2 Queued: Just now [manage_parallel_jobs]

3 Agent: LAPTOP-4J9P4KK5

4 Started: Just now

5 Duration: 22s

6

7 The agent request is already running or has already completed.

8 Job preparation parameters

9 1 artifact produced

10 Job live console data:

11 Starting: Job

12 Async Command Start: DetectDockerContainer

13 Async Command End: DetectDockerContainer

14 Async Command Start: DetectDockerContainer

15 Async Command End: DetectDockerContainer

16 Finishing: Job

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

