

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

Ans: c. Windows subsystem for linux

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

Ans: b. Frequent integration of code changes

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

Ans: b. Change the working directory

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

- a. Ubuntu
- b. CentOS
- c. Docker
- d. Debian

Ans: c. Docker

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

Ans: c. Packaging and deploying applications in containers

6.What is the primary purpose of Azure DevOps?

- a. Infrastructure management
- b. Software development and delivery
- c. Network security
- d. Virtualization

Ans: b. Software development and delivery

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

Ans: b. Azure Boards and Azure Pipelines

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.

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

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

Ans: c. Administrative tasks with superuser privileges

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

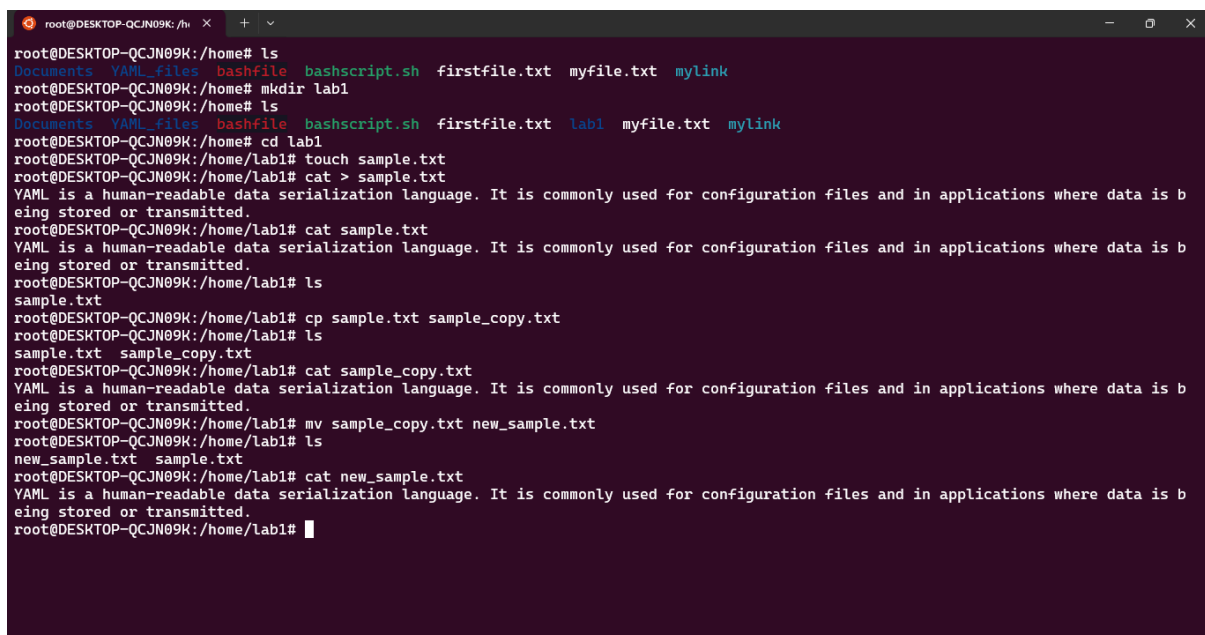
- a. Azure Boards
- b. Azure Repos
- c. Azure Pipelines
- d. Azure Artifacts

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.

A terminal window with a dark background and light-colored text. The prompt is 'root@DESKTOP-QCJN09K: /home#'. The user enters 'ls', showing a list of files including 'Documents', 'YAML_files', 'bashfile', 'bashscript.sh', 'firstfile.txt', 'myfile.txt', and 'mylink'. Then they enter 'mkdir lab1'. Next, 'cd lab1'. Then 'touch sample.txt'. Then 'cat > sample.txt', followed by a multi-line text input: 'YAML is a human-readable data serialization language. It is commonly used for configuration files and in applications where data is being stored or transmitted.'. Then 'cat sample.txt' which outputs the same text. Then 'cp sample.txt sample_copy.txt'. Then 'ls' showing 'sample.txt' and 'sample_copy.txt'. Then 'cat sample_copy.txt' which outputs the same text. Then 'mv sample_copy.txt new_sample.txt'. Then 'ls' showing 'new_sample.txt' and 'sample.txt'. Finally, 'cat new_sample.txt' which outputs the same text. The prompt returns to 'root@DESKTOP-QCJN09K: /home/lab1#'.

```
root@DESKTOP-QCJN09K:/home# ls
Documents  YAML_files  bashfile    bashscript.sh  firstfile.txt  myfile.txt  mylink
root@DESKTOP-QCJN09K:/home# mkdir lab1
root@DESKTOP-QCJN09K:/home# ls
Documents  YAML_files  bashfile    bashscript.sh  firstfile.txt  lab1  myfile.txt  mylink
root@DESKTOP-QCJN09K:/home# cd lab1
root@DESKTOP-QCJN09K:/home/lab1# touch sample.txt
root@DESKTOP-QCJN09K:/home/lab1# cat > sample.txt
YAML is a human-readable data serialization language. It is commonly used for configuration files and in applications where data is b
eing stored or transmitted.
root@DESKTOP-QCJN09K:/home/lab1# cat sample.txt
YAML is a human-readable data serialization language. It is commonly used for configuration files and in applications where data is b
eing stored or transmitted.
root@DESKTOP-QCJN09K:/home/lab1# ls
sample.txt
root@DESKTOP-QCJN09K:/home/lab1# cp sample.txt sample_copy.txt
root@DESKTOP-QCJN09K:/home/lab1# ls
sample.txt  sample_copy.txt
root@DESKTOP-QCJN09K:/home/lab1# cat sample_copy.txt
YAML is a human-readable data serialization language. It is commonly used for configuration files and in applications where data is b
eing stored or transmitted.
root@DESKTOP-QCJN09K:/home/lab1# mv sample_copy.txt new_sample.txt
root@DESKTOP-QCJN09K:/home/lab1# ls
new_sample.txt  sample.txt
root@DESKTOP-QCJN09K:/home/lab1# cat new_sample.txt
YAML is a human-readable data serialization language. It is commonly used for configuration files and in applications where data is b
eing stored or transmitted.
root@DESKTOP-QCJN09K:/home/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.

```
balan@DESKTOP-QCJN09K:/# ls
Docker boot etc init lib32 libx32 media opt root sample3 snap sys usr
bin dev home lib lib64 lost+found mnt proc run sbin srv tmp var
root@DESKTOP-QCJN09K:/# cd home
root@DESKTOP-QCJN09K:/home# ls
Documents VAML_files balan bashfile bashscript.sh checks.txt firstfile.txt lab1 myfile.txt mylink
root@DESKTOP-QCJN09K:/home# mkdir lab2
root@DESKTOP-QCJN09K:/home# ls
Documents VAML_files balan bashfile bashscript.sh checks.txt firstfile.txt lab1 lab2 myfile.txt mylink
root@DESKTOP-QCJN09K:/home# cd lab2
root@DESKTOP-QCJN09K:/home/lab2# touch secret.txt
root@DESKTOP-QCJN09K:/home/lab2# cat > secret.txt
this is some sample text in secret file
root@DESKTOP-QCJN09K:/home/lab2# sudo chmod u+rw secret.txt
root@DESKTOP-QCJN09K:/home/lab2# chown owner2 secret.txt
root@DESKTOP-QCJN09K:/home/lab2# sudo login balan
Password:

balan@DESKTOP-QCJN09K:~$ ls
balan@DESKTOP-QCJN09K:~$ cd ..
balan@DESKTOP-QCJN09K:/home$ ls
Documents VAML_files balan bashfile bashscript.sh checks.txt firstfile.txt lab1 lab2 myfile.txt mylink
balan@DESKTOP-QCJN09K:/home$ cd lab2
balan@DESKTOP-QCJN09K:/home/lab2$ ls
secret.txt
balan@DESKTOP-QCJN09K:/home/lab2$ cat > secret.txt
-bash: secret.txt: Permission denied
balan@DESKTOP-QCJN09K:/home/lab2$ ls -l
total 4
-rw-r--r-- 1 owner2 root 40 Oct 20 12:54 secret.txt
balan@DESKTOP-QCJN09K:/home/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.

```
root@DESKTOP-QCJN09K:/# cd ..
root@DESKTOP-QCJN09K:/# ls
Docker boot etc init lib32 libx32 media opt root sample3 snap sys usr
bin dev home lib lib64 lost+found mnt proc run sbin srv tmp var
root@DESKTOP-QCJN09K:/# cd home
root@DESKTOP-QCJN09K:/home# ls
Documents VAML_files balan bashfile bashscript.sh checks.txt firstfile.txt lab1 lab2 myfile.txt mylink
root@DESKTOP-QCJN09K:/home# mkdir lab3
root@DESKTOP-QCJN09K:/home# ls
Documents VAML_files balan bashfile bashscript.sh checks.txt firstfile.txt lab1 lab2 lab3 myfile.txt mylink
root@DESKTOP-QCJN09K:/home# cd lab3
root@DESKTOP-QCJN09K:/home/lab3# touch textfile.txt
root@DESKTOP-QCJN09K:/home/lab3# cat > textfile.txt
YAML is a human-readable format for data serialization. This means it can be used for structured data, like what you can find in configuration files.
YAML: command not found
root@DESKTOP-QCJN09K:/home/lab3# cat textfile.txt
YAML is a human-readable format for data serialization. This means it can be used for structured data, like what you can find in configuration files.
root@DESKTOP-QCJN09K:/home/lab3# cat textfile.txt
YAML is a human-readable format for data serialization. This means it can be used for structured data, like what you can find in configuration files.
root@DESKTOP-QCJN09K:/home/lab3# grep structured textfile.txt
YAML is a human-readable format for data serialization. This means it can be used for structured data, like what you can find in configuration files.
root@DESKTOP-QCJN09K:/home/lab3# sed -i 's/YAML/Yet another markup language/g' textfile.txt
root@DESKTOP-QCJN09K:/home/lab3# cat textfile.txt
Yet another markup language is a human-readable format for data serialization. This means it can be used for structured data, like what you can find in configuration files.
root@DESKTOP-QCJN09K:/home/lab3# wc textfile.txt
  1 28 173 textfile.txt
root@DESKTOP-QCJN09K:/home/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.

```
root@DESKTOP-QCJN09K: ~# cd ..
root@DESKTOP-QCJN09K: /# ls
Docker boot etc init lib32 libx32 media opt root sample3 snap sys usr
bin dev home lib lib64 lost+found mnt proc run sbin srv tmp var
root@DESKTOP-QCJN09K: /# cd home
root@DESKTOP-QCJN09K: /home# ls
Documents YAML_files balan bashfile bashscript.sh checks.txt firstfile.txt lab1 lab2 lab3 myfile.txt mylink
root@DESKTOP-QCJN09K: /home# mkdir lab4
root@DESKTOP-QCJN09K: /home# cd lab4
root@DESKTOP-QCJN09K: /home/lab4# touch config.yaml
root@DESKTOP-QCJN09K: /home/lab4# ls
config.yaml
root@DESKTOP-QCJN09K: /home/lab4# code .
```

The screenshot shows the VS Code editor with the 'config.yaml' file open. The file content is:

```
1 ---
2 name: Fictitious application
3 version: 1.0.0
4 description: this is fictitious application
5
```

The terminal at the bottom shows the output of running 'yamllint config.yaml':

```
Unpacking yamllint (1.20.0-1) ...
Setting up python3-pathspect (0.7.0-1) ...
Setting up yamllint (1.20.0-1) ...
Processing triggers for man-db (2.9.1-1) ...
root@DESKTOP-QCJN09K: /home/lab4# yamllint config.yaml
config.yaml
1:1 warning missing document start "---" (document-start)
3:44 error no new line character at the end of file (new-line-at-end-of-file)
```

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.

```
root@DESKTOP-QCJN09K:/home# ls
Documents  YAML_files  balan  bashfile  bashscript.sh  checks.txt  firstfile.txt  lab1  lab2  lab3  lab4  myfile.txt  mylink
root@DESKTOP-QCJN09K:/home# mkdir lab5
root@DESKTOP-QCJN09K:/home# ls
Documents  YAML_files  balan  bashfile  bashscript.sh  checks.txt  firstfile.txt  lab1  lab2  lab3  lab4  lab5  myfile.txt  mylink
root@DESKTOP-QCJN09K:/home# cd lab5
root@DESKTOP-QCJN09K:/home/lab5# touch fruits.yaml
root@DESKTOP-QCJN09K:/home/lab5# code .
root@DESKTOP-QCJN09K:/home/lab5# ls
fruits.yaml
root@DESKTOP-QCJN09K:/home/lab5# yamllint fruits.yaml
root@DESKTOP-QCJN09K:/home/lab5#
```

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.

```
root@DESKTOP-QCJN09K:/home# ls
Documents  YAML_files  balan  bashfile  bashscript.sh  checks.txt  firstfile.txt  lab1  lab2  lab3  lab4  lab5  myfile.txt  mylink
root@DESKTOP-QCJN09K:/home# mkdir lab6
root@DESKTOP-QCJN09K:/home# ls
Documents  balan  bashfile  bashscript.sh  firstfile.txt  lab2  lab4  lab6  mylink
YAML_files  bashfile  checks.txt  lab1  lab3  lab5  myfile.txt
root@DESKTOP-QCJN09K:/home# cd lab6
root@DESKTOP-QCJN09K:/home/lab6# touch data.yaml
root@DESKTOP-QCJN09K:/home/lab6# code .
root@DESKTOP-QCJN09K:/home/lab6# yamllint data.yaml
root@DESKTOP-QCJN09K:/home/lab6# cat data.yaml
---
organization:
  name: kanini software solutions
  location: Chennai

departments:
  - name: Software Developer
    employees:
      - name: employee1
        position: Junior SE
      - name: employee2
        position: Senior SE
  - name: Testing
    employees:
      - name: Tester1
        position: Design tester
      - name: Tester2
        position: Quality Assurance
root@DESKTOP-QCJN09K:/home/lab6# yamllint data.yaml
root@DESKTOP-QCJN09K:/home/lab6#
```

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.

```
PS C:\agent> .\run.cmd
Error reported in diagnostic logs. Please examine the log for more details.
- C:\agent\_diag\Agent_20231020-120616-utc.log
Scanning for tool capabilities.
Connecting to the server.
2023-10-20 12:06:36Z: Agent connect error: No such host is known. (dev.azure.com:443). Retrying until reconnected.
2023-10-20 12:07:43Z: Agent reconnected.
2023-10-20 12:07:43Z: Listening for Jobs
2023-10-20 12:07:46Z: Running job: Agent job 1
2023-10-20 12:09:36Z: Job Agent job 1 completed with result: Failed
2023-10-20 12:14:14Z: Running job: Agent job 1
2023-10-20 12:15:02Z: Job Agent job 1 completed with result: Failed
2023-10-20 12:17:47Z: Running job: angularci
2023-10-20 12:24:53Z: Job angularci completed with result: Succeeded
```

Azure DevOps balan16 / Classic Azure CI Pipeline for... / Pipelines

Classic Azure CI Pipeline for Angular Application-CI

Tasks Variables Triggers Options History Save & queue Discard Summary Queue

Pipeline Build pipeline

Get sources

angularci

npm install

npm builds

npm test

Publish Artifact: drop

Name *

Classic Azure CI Pipeline for Angular Application-CI

Agent pool Default

Parameters

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

Azure DevOps balan16 / Classic Azure CI Pipeline for... / Pipelines / Classic Azure CI Pipeline for... / 28

Classic Azure CI Pipeline for Angular Application-CI

Jobs in run #28

Jobs

Job	Duration
angularci	7m 0s
Initialize job	<1s
Checkout vasanthabala...	7s
npm install	2m 21s
npm builds	1m 54s
npm test	2m 3s
Publish Artifact: drop	9s
Post-job: Checkout vas...	2s
Finalize Job	<1s

angularci

Pool: Default

Agent: DESKTOP-QC3M99K

Started: Today at 5:47 PM

Duration: 7m 0s

Job preparation parameters

2 queue time variables used

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.

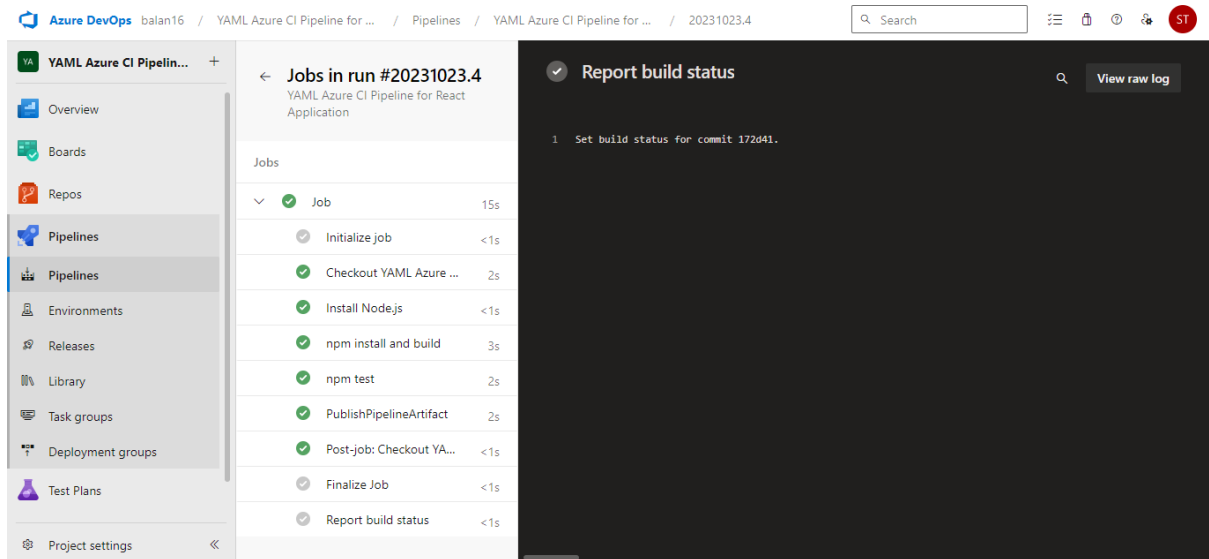
- **Tasks:**

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.

```
PS C:\> cd agent
PS C:\agent> .\run.cmd
Scanning for tool capabilities.
Connecting to the server.
2023-10-23 04:29:57Z: Listening for Jobs
2023-10-23 04:30:59Z: Running job: Job
2023-10-23 04:31:49Z: Job Job completed with result: Failed
2023-10-23 04:31:51Z: Running job: Job
2023-10-23 04:32:10Z: Job Job completed with result: Failed
2023-10-23 04:33:39Z: Running job: Job
2023-10-23 04:33:55Z: Job Job completed with result: Succeeded
2023-10-23 04:33:58Z: Running job: Job
2023-10-23 04:34:16Z: Job Job completed with result: Succeeded
```

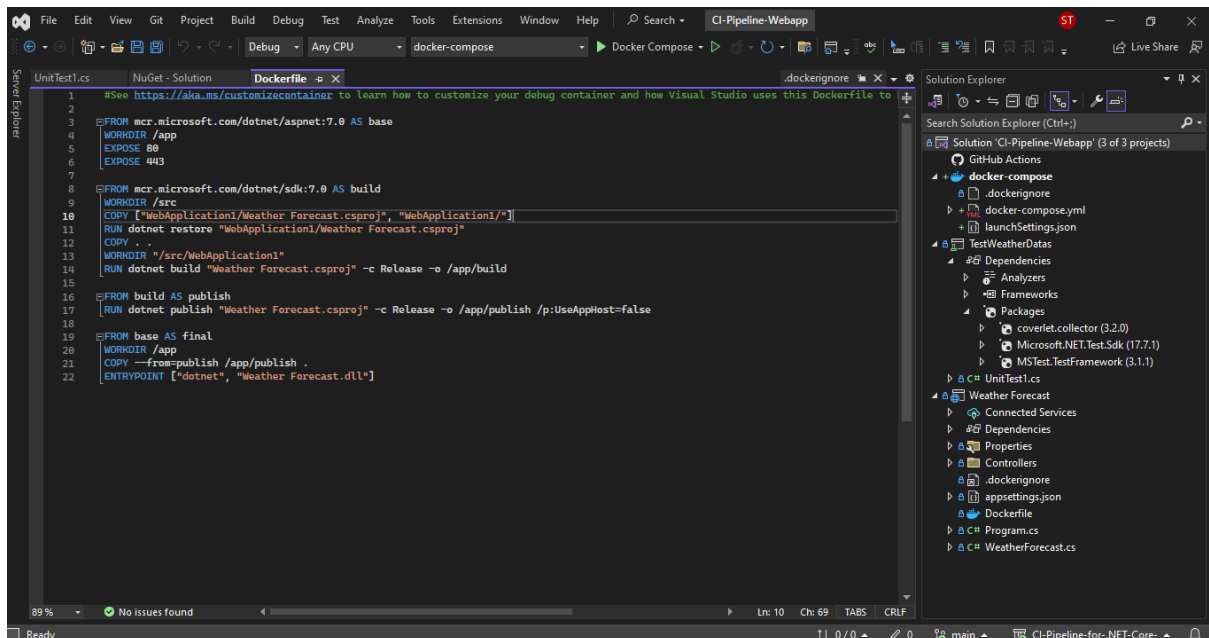
The screenshot displays the Azure DevOps web interface for a project named 'YAML Azure CI Pipeline for React Application'. The left sidebar shows the navigation menu with 'Pipelines' selected. The main area shows the pipeline definition in 'azure-pipelines.yml' format. The pipeline is triggered by a push to the 'master' branch and consists of three main steps: 'NodeTool@0' (installing Node.js), 'script' (installing dependencies and running tests), and 'PublishPipelineArtifact@1' (publishing the build artifacts). The right sidebar shows a list of tasks available in the pipeline library, including '.NET Core', 'Android signing', 'Ant', 'App Center distribute', 'App Center test', and 'Archive files'.

```
4 # https://docs.microsoft.com/azure/devops/pipelines/languages/javascript
5
6 trigger:
7   - master
8
9 pool:
10  | name: Default
11
12 steps:
13   - task: NodeTool@0
14     inputs:
15       | versionSpec: '18.x'
16       | displayName: 'Install Node.js'
17
18   - script: |
19       | npm install
20       | npm run build
21       | displayName: 'npm install and build'
22
23   - script: |
24       | npm install
25       | npm test
26       | displayName: 'npm test'
27
28
29 settings:
30   | task: PublishPipelineArtifact@1
31   | inputs:
32     | targetPath: '$(Pipeline.Workspace)'
33     | artifact: 'drop'
34     | publishLocation: 'pipeline'
```



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.



Azure DevOps CI Pipeline for .NET Core Application with MS Unit Test-ASP.NET Core-CI

Jobs in run #2023102...

Agent job 1

Initialize job 2s

Checkout vasantha... 29s

Restore 10s

Build 37s

Test 1s

Build solution *... 1m 46s

Publish 38s

Publish Artifact 16s

Post-job: Checkout v... 2s

Finalize Job 3s

```

##[debug]Agent running environment resource - Disk:C:\ available:67545.00MB out of 203886.00MB, Memory: us
54 ##[debug]Agent running environment resource - Disk:C:\ available:67545.00MB out of 203886.00MB, Memory: us
55 ##[debug]Agent running environment resource - Disk:C:\ available:67545.00MB out of 203886.00MB, Memory: us
56 ##[debug]Agent running environment resource - Disk:C:\ available:67545.00MB out of 203886.00MB, Memory: us
57 ##[debug]Agent running environment resource - Disk:C:\ available:67543.00MB out of 203886.00MB, Memory: us
58 ##[debug]Agent running environment resource - Disk:C:\ available:67543.00MB out of 203886.00MB, Memory: us
59 ##[debug]Agent running environment resource - Disk:C:\ available:67543.00MB out of 203886.00MB, Memory: us
60 ##[debug]Agent running environment resource - Disk:C:\ available:67543.00MB out of 203886.00MB, Memory: us
61 ##[debug]Agent running environment resource - Disk:C:\ available:67543.00MB out of 203886.00MB, Memory: us
62 ##[debug]Agent running environment resource - Disk:C:\ available:67543.00MB out of 203886.00MB, Memory: us
63 ##[debug]Agent running environment resource - Disk:C:\ available:67540.00MB out of 203886.00MB, Memory: us
64 ##[debug]Agent running environment resource - Disk:C:\ available:67539.00MB out of 203886.00MB, Memory: us
65 ##[debug]Agent running environment resource - Disk:C:\ available:67545.00MB out of 203886.00MB, Memory: us
66 ##[debug]Agent running environment resource - Disk:C:\ available:67544.00MB out of 203886.00MB, Memory: us
67 ##[debug]Agent running environment resource - Disk:C:\ available:67544.00MB out of 203886.00MB, Memory: us
68 ##[debug]Agent running environment resource - Disk:C:\ available:67544.00MB out of 203886.00MB, Memory: us
69 ##[debug]Starting diagnostic file upload.
70 ##[debug]Setting up diagnostic log folders.
71 ##[debug]Creating diagnostic log files folder.
72 ##[debug]Creating diagnostic log environment file.
73 ##[debug]Agent running environment resource - Disk:C:\ available:67544.00MB out of 203886.00MB, Memory: us
74 ##[debug]Agent running environment resource - Disk:C:\ available:67544.00MB out of 203886.00MB, Memory: us
75 ##[debug]Agent running environment resource - Disk:C:\ available:67544.00MB out of 203886.00MB, Memory: us
76 ##[debug]Agent running environment resource - Disk:C:\ available:67544.00MB out of 203886.00MB, Memory: us

```

Azure DevOps CI Pipeline for .NET Core Application with MS Unit Test-ASP.NET Core-CI

Runs

Description	Stages	Time
#20231024.2 - test data checed Manually triggered for main e45331a2	✓	17m ago 5m 7s
#20231024.1 - test data checed Manually triggered for main e45331a2	✓	20m ago 13m 22s

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

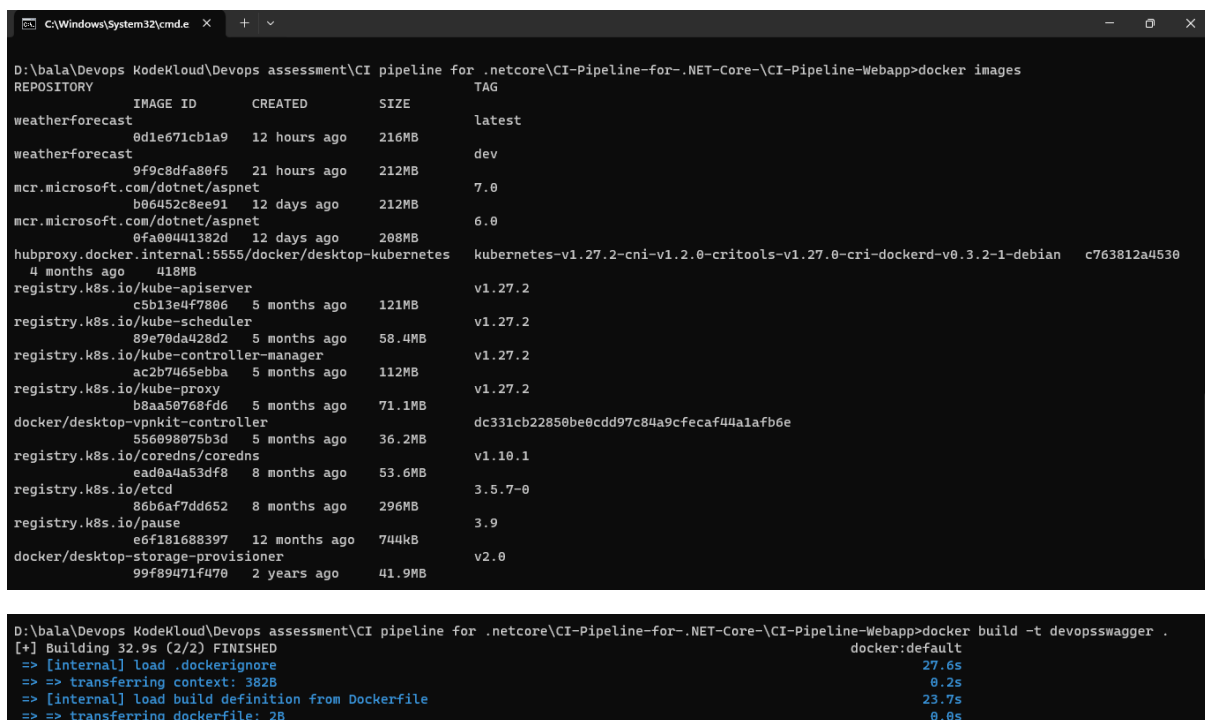
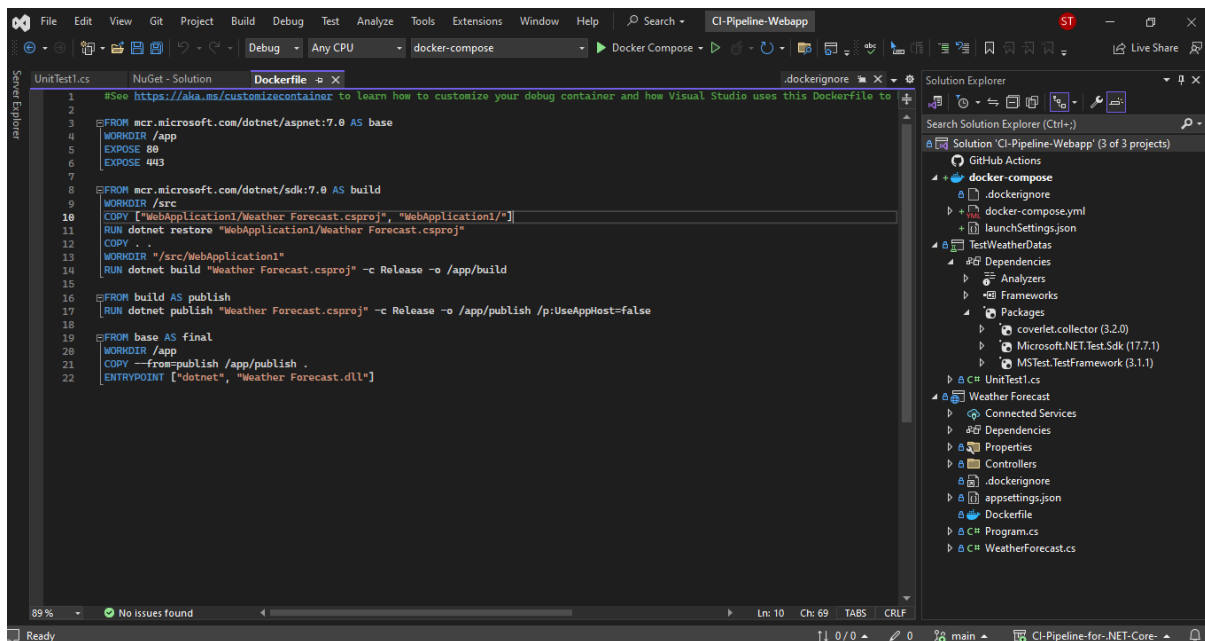
Step 2: Build the .NET Core Web API Project

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



Docker Desktop

Upgrade plan

Search for images, containers, volumes, extensions and more...

Ctrl+K

balan1...

Containers

Images

Volumes

Dev Environments BETA

Docker Scout

Learning center

Extensions

Add Extensions

Containers [Give feedback](#)

Container CPU usage

0.00% / 800% (8 cores allocated)

Container memory usage

21.04MB / 3.68GB

Show charts

Search

Only show running containers

	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
	kind_joliot	weatherforecast1ai	Running	0%		6 minutes ago	
	dockercom		Exited	0%		14 hours ago	

Showing 2 items

Walkthroughs

What is a container?

5 mins

How do I run a container?

6 mins

[View more in the Learning center](#)

Engine running

RAM 3.66 GB CPU 3.89% Signed in

v4.24.2