Devops – Final Assessment **VASANTHABALAN M –KB23028**

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

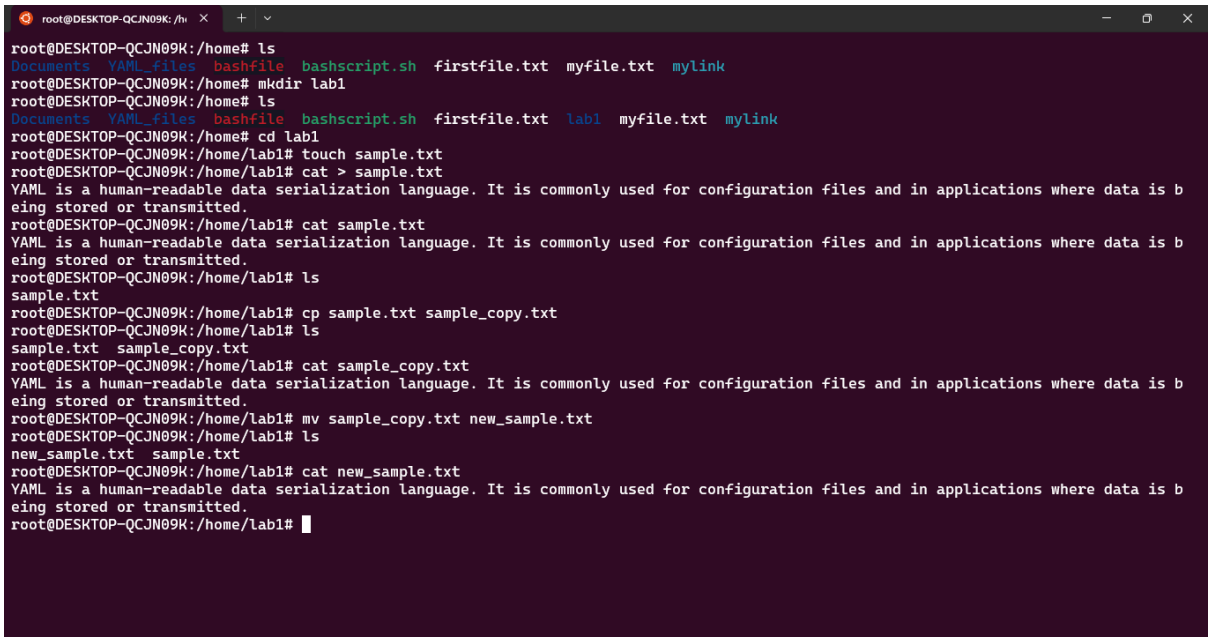
Ans: c. Azure Pipelines

Section 2: Labs

Lab 1: File and Directory Management

- **Objective: Practice basic file and directory management commands.**
- **Tasks:**

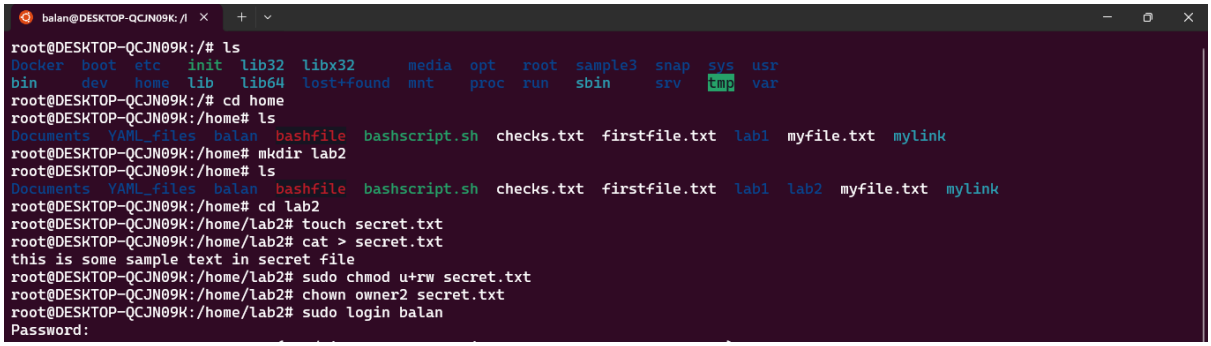
1. Create a directory called "lab1" in your home directory.
2. Inside "lab1," create a text file named "sample.txt" with some content.
3. Make a copy of "sample.txt" and name it "sample_copy.txt."
4. Rename "sample_copy.txt" to "new_sample.txt."
5. List the files in the "lab1" directory to confirm their names.



Lab 2: Permissions and Ownership

- **Objective: Understand and manage file permissions and ownership.**
- **Tasks:**

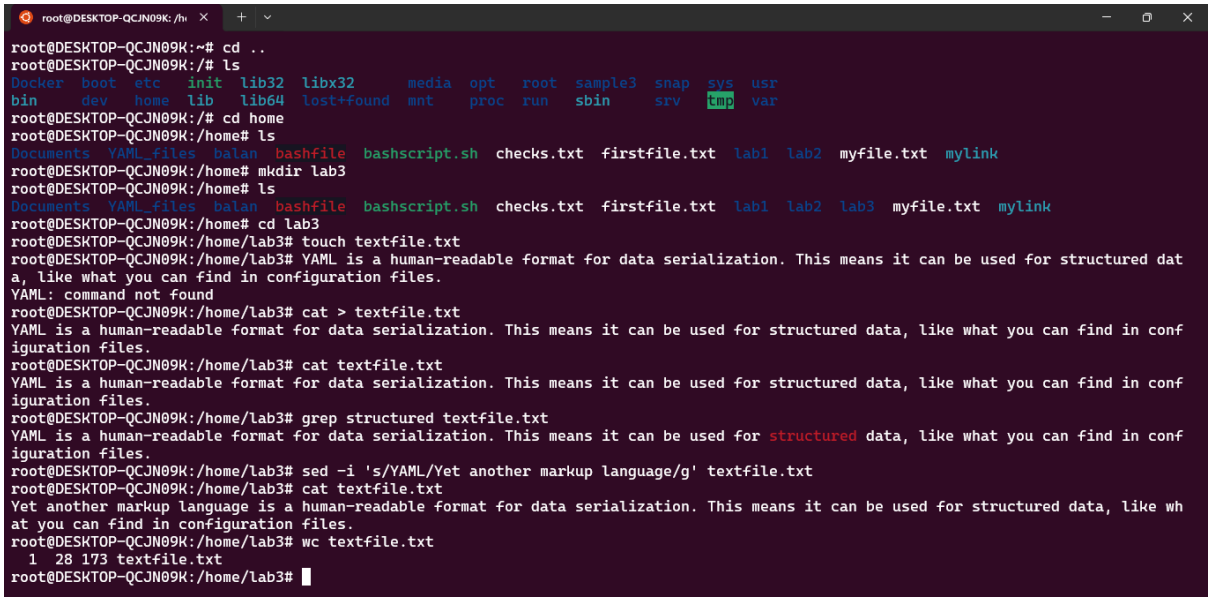
1. Create a new file named "secret.txt" in the "lab2" directory.
2. Set the file permissions to allow read and write access only to the owner.
3. Change the owner of "secret.txt" to another user.
4. Verify the new permissions and owner using the ls -l and ls -n commands.



Lab 3: Text Processing with Command Line Tools

- **Objective: Practice text processing using command-line tools.**
- **Tasks:**

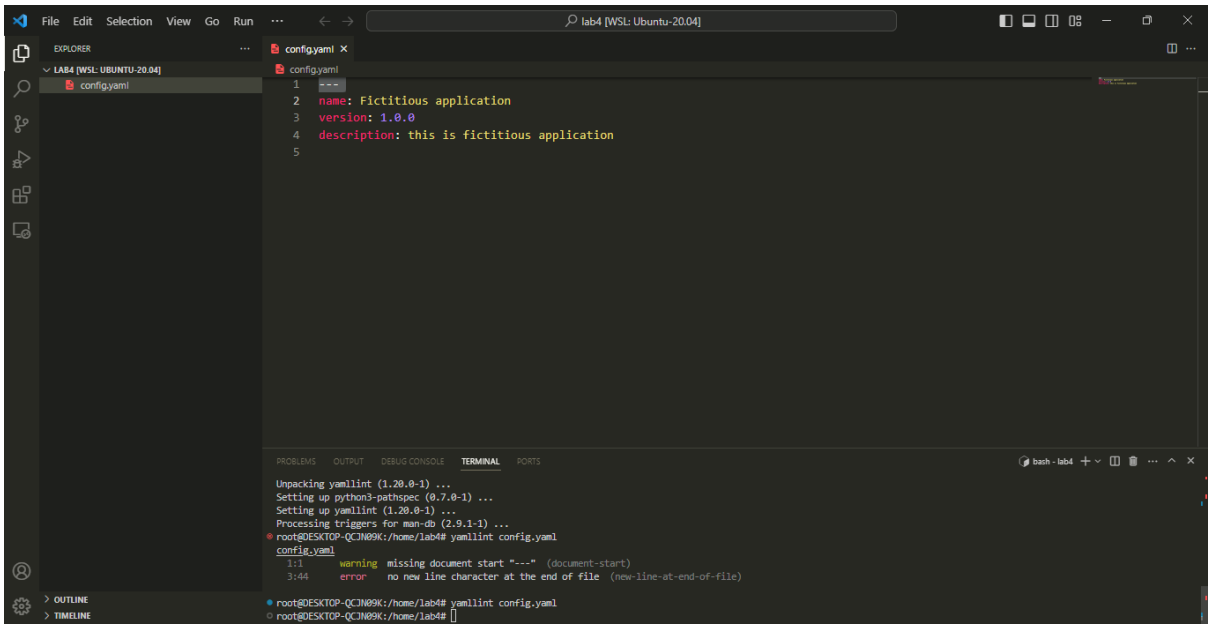
1. Create a text file with some random text in the "lab3" directory.
2. Use the grep command to search for a specific word or pattern in the file.
3. Use the sed command to replace a word or phrase with another in the file.
4. Use the wc command to count the number of lines, words, and characters in the file.



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

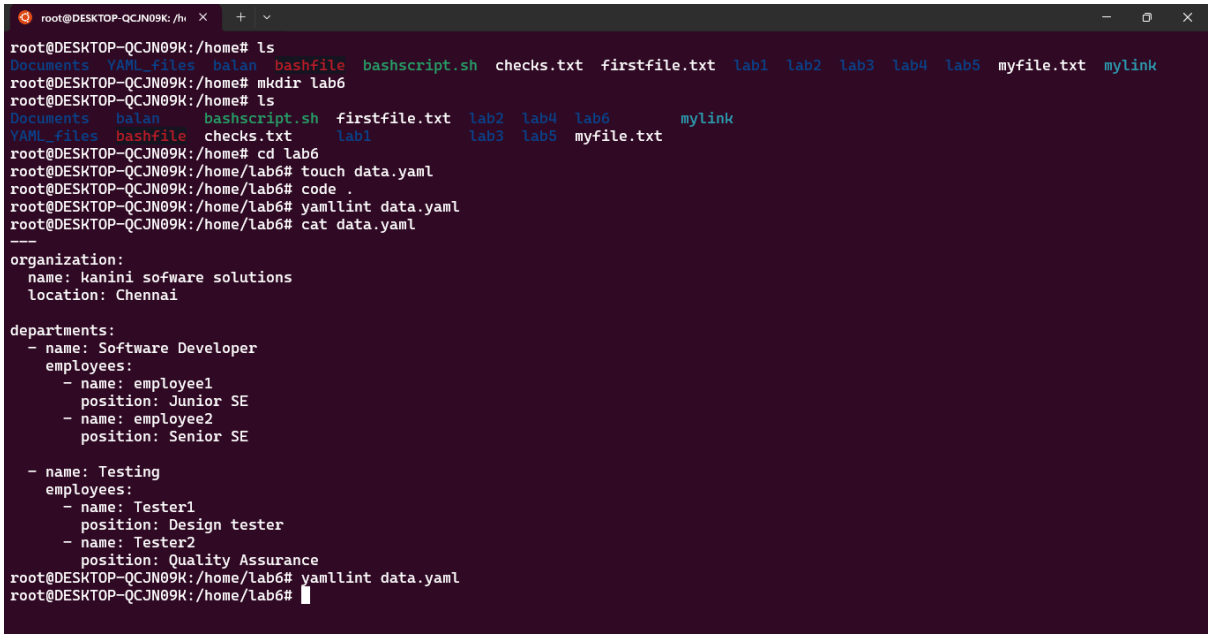
- **Objective: Practice working with lists (arrays) in YAML.**
- **Task:**

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

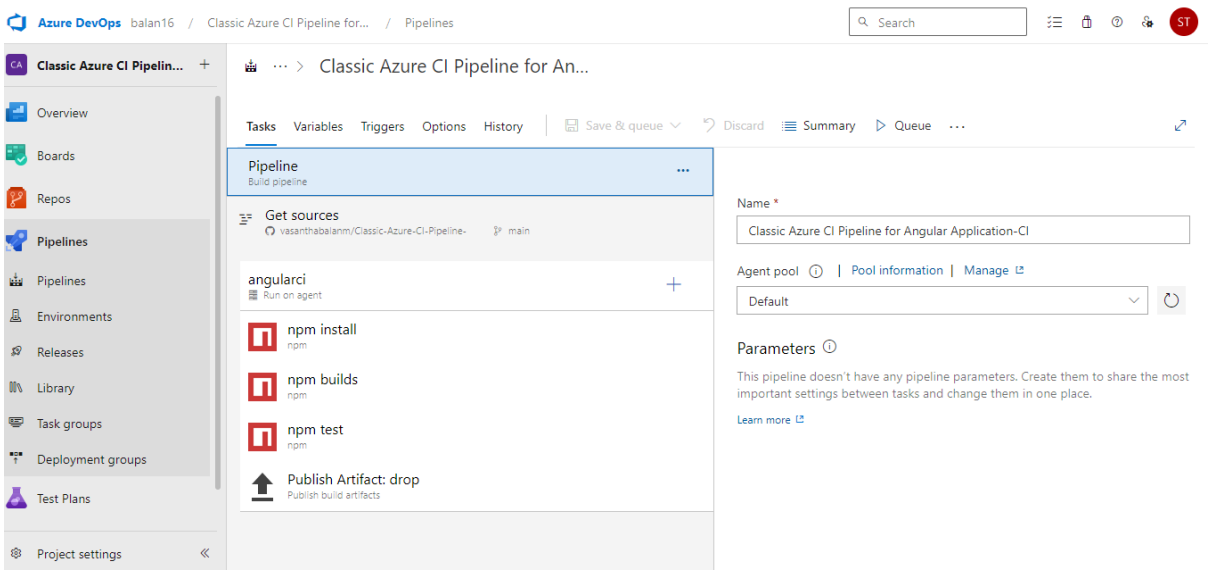
- **Objective: Set up a classic Azure CI pipeline to build a simple Angular application with unit testing using Jasmine and Karma.**
- **Tasks:**

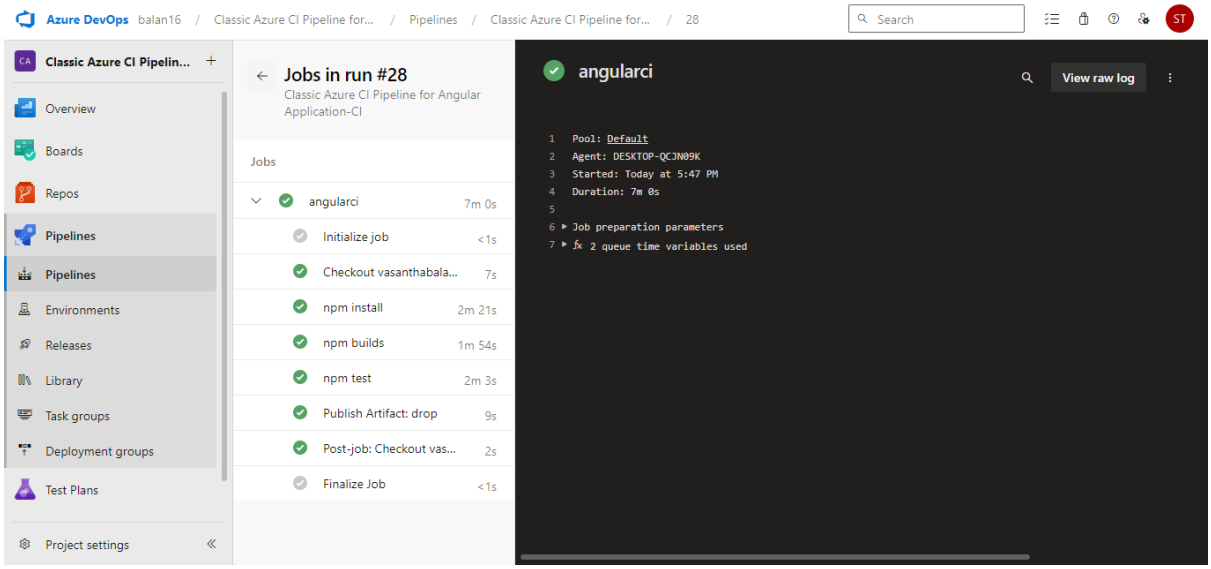
1. Create an Azure DevOps project.
2. Set up a classic CI pipeline to build an Angular application.

3. Configure the pipeline to use Jasmine and Karma for unit testing.

4. Run the pipeline and validate the test results.

```
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
```



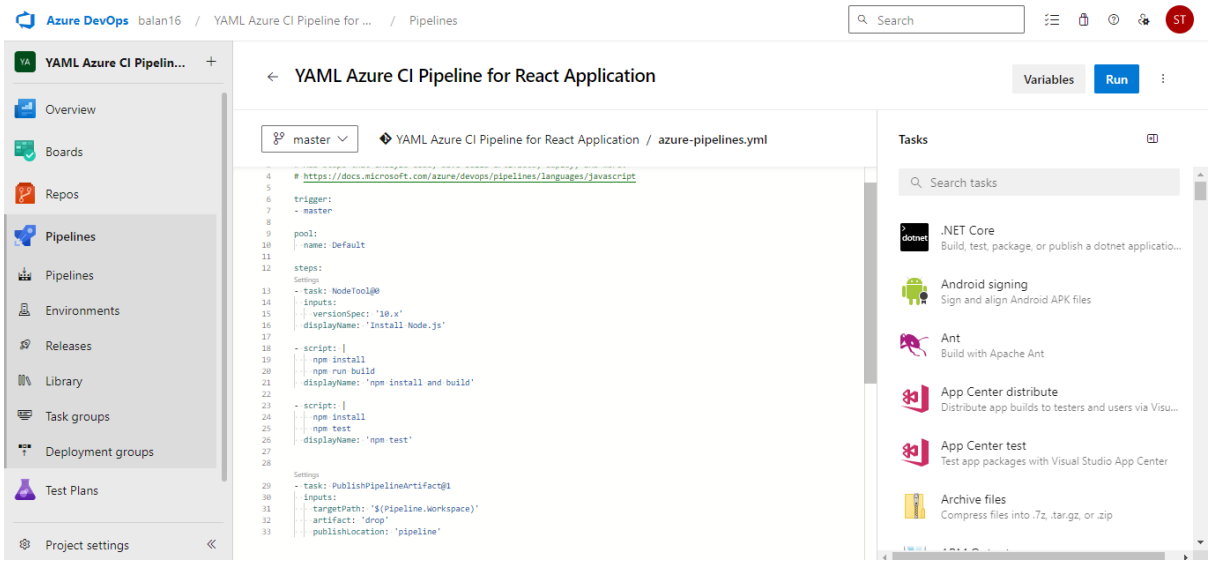


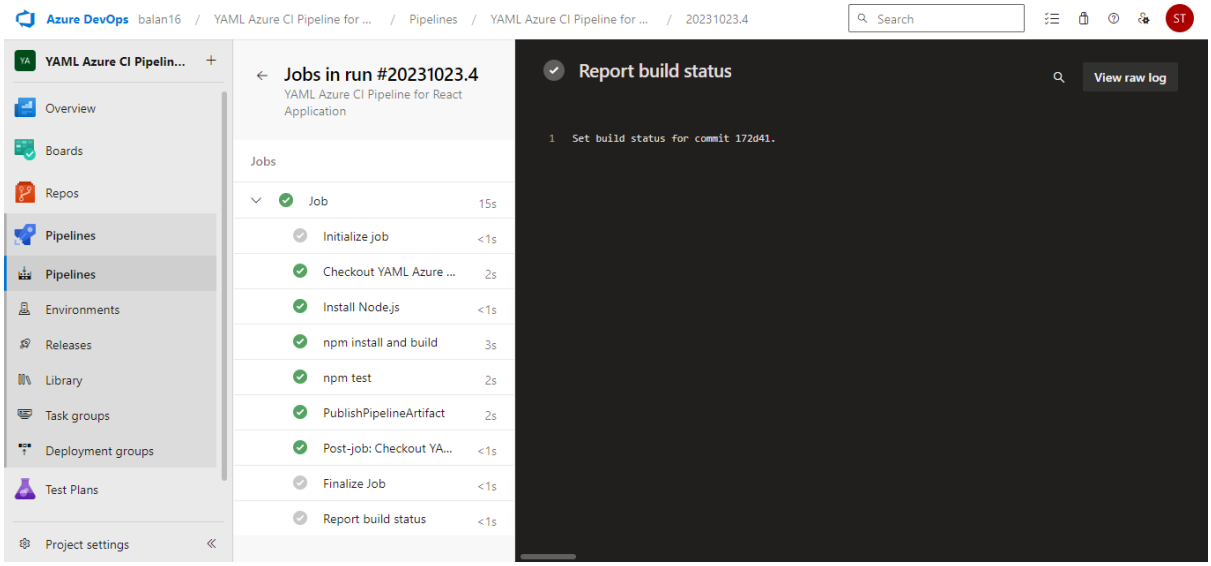
Lab 8: Create YAML Azure CI Pipeline for React Application

- **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
```



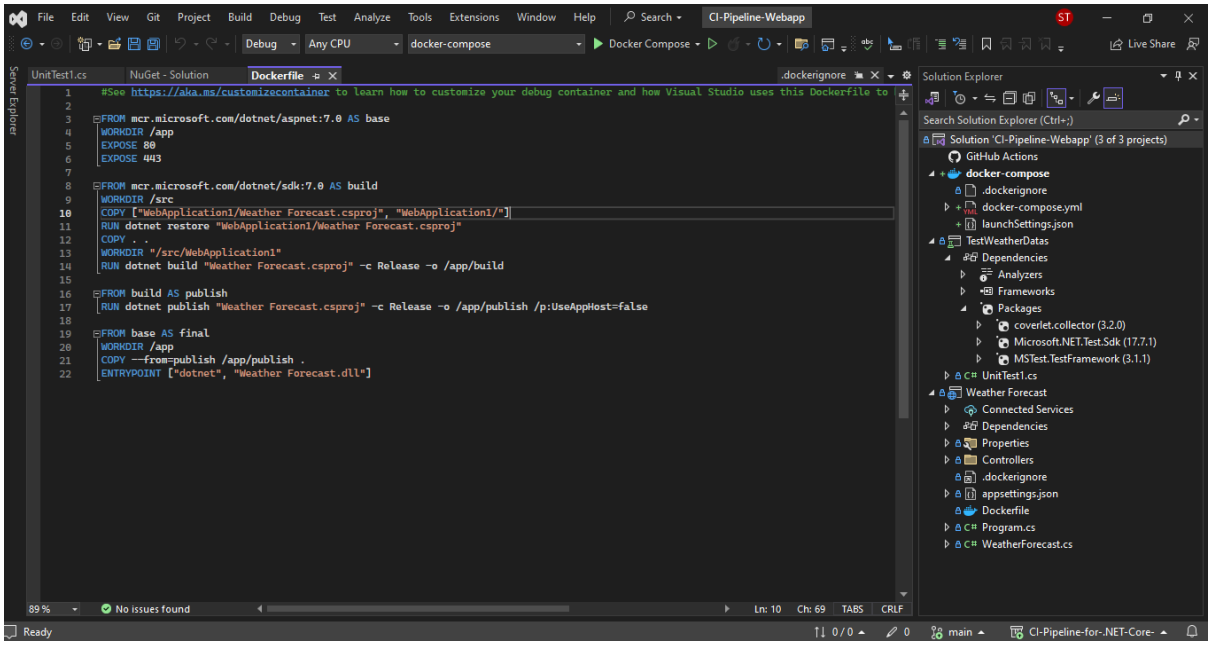


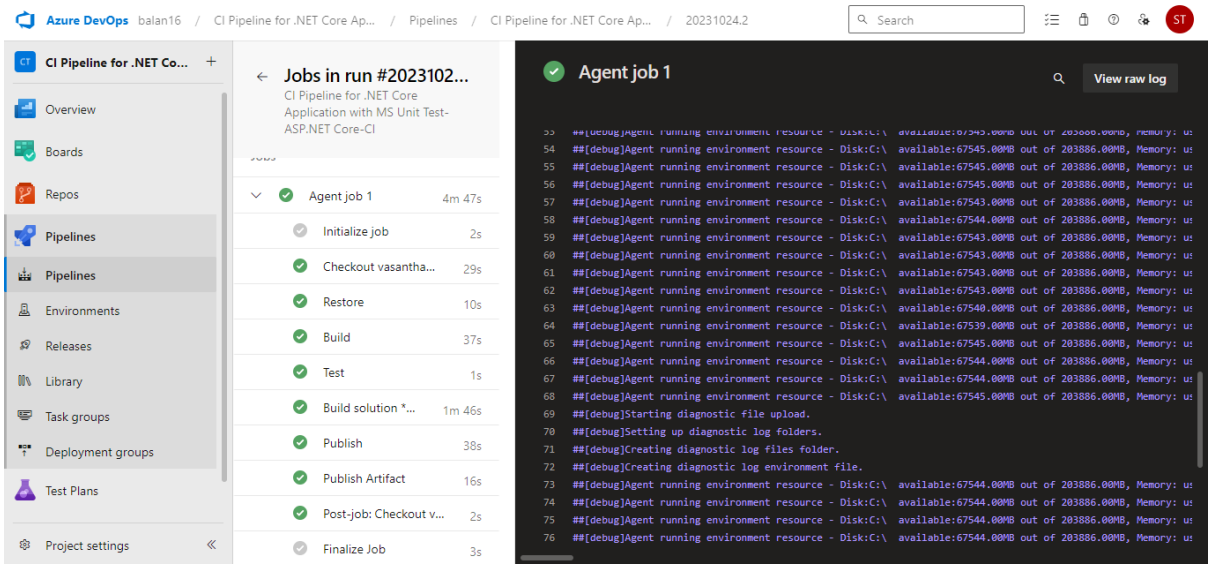
Lab 9: Create CI Pipeline for .NET Core Application with MS Unit Test

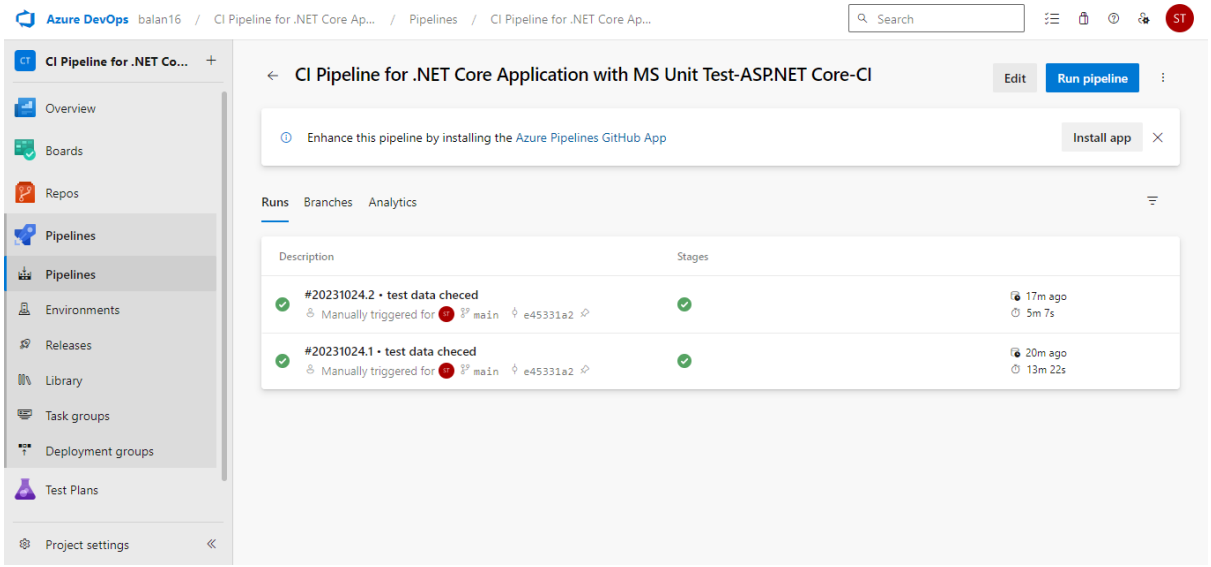
- **Objective: Create a CI pipeline, either classic or YAML, to build a .NET Core application and run MS Unit tests.**

- **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

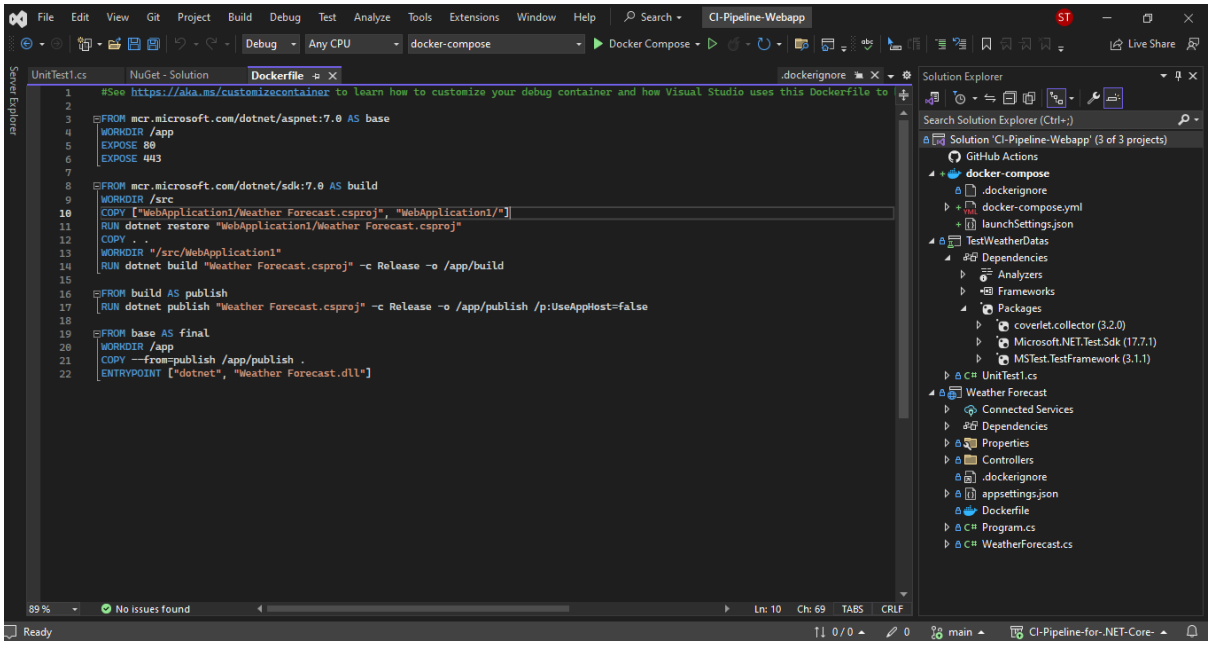
Step 2: Build the .NET Core Web API Project

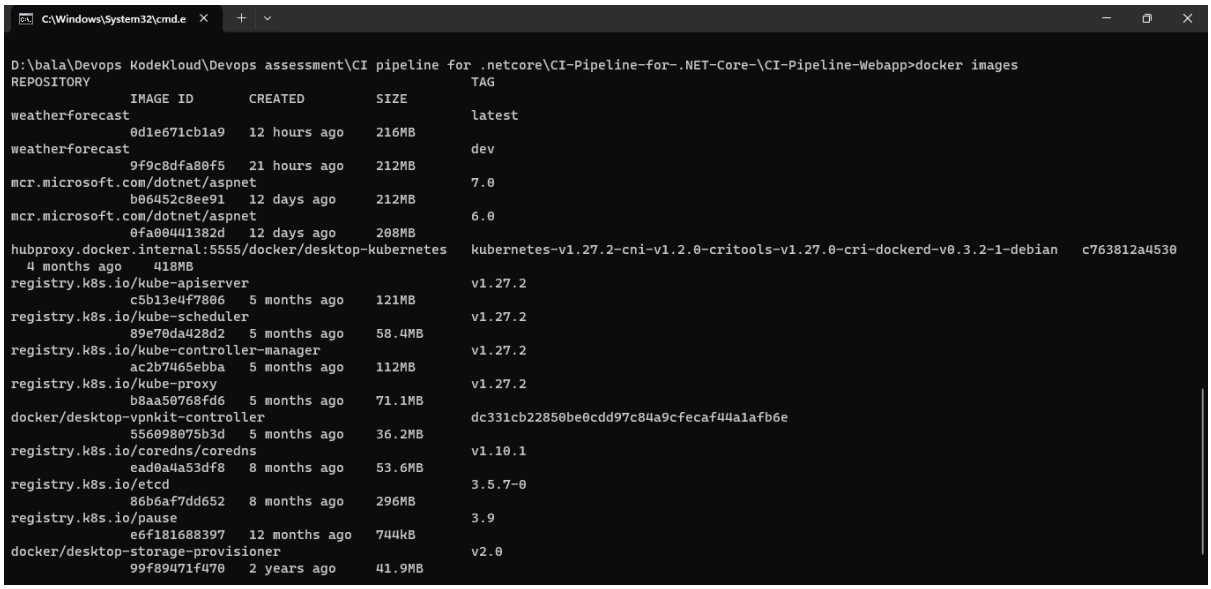
Step 3: Dockerize the .NET Core Web API

Step 4: Build the Docker Image

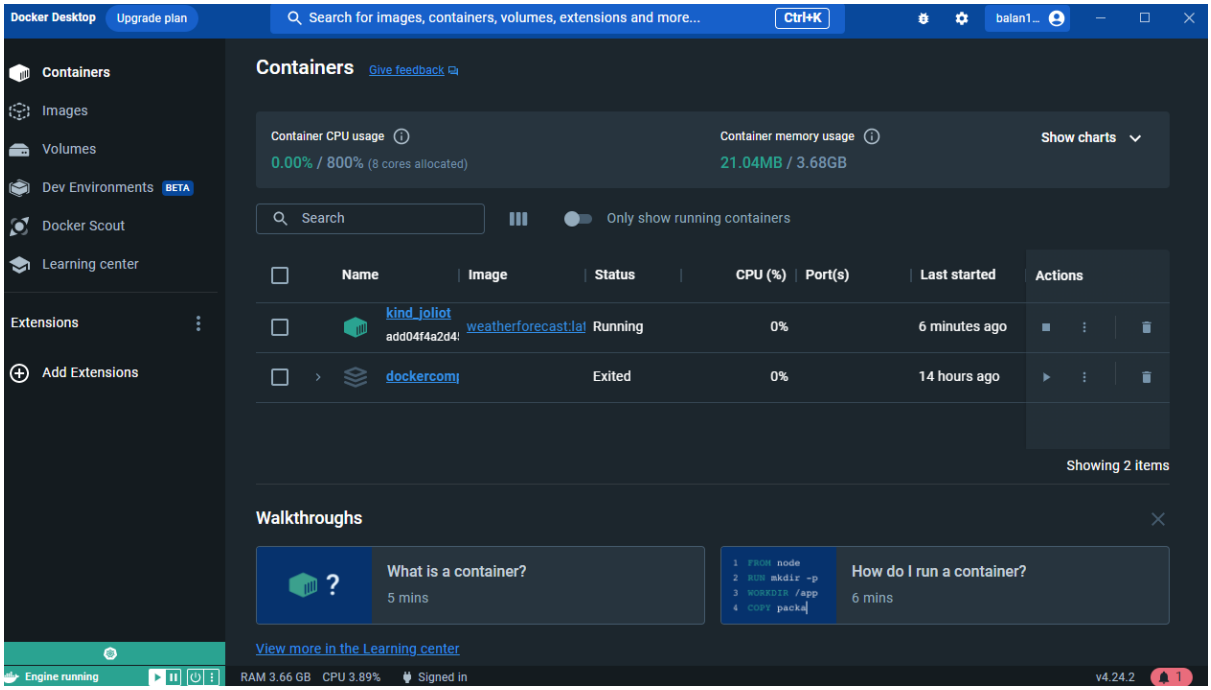
Step 5: Run the Docker Container in Rancher Desktop

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






```
D:\bala\Devops KodeKloud\Devops assessment\CI pipeline for .netcore\CI-Pipeline-for-.NET-Core-\CI-Pipeline-Webapp>docker build -t devopsswagger .
[+] Building 32.9s (2/2) FINISHED                                                                                docker:default
 => [internal] load .dockerignore                                                                                         27.6s
 => => transferring context: 382B                                                                                          0.2s
 => [internal] load build definition from Dockerfile                                                                      23.7s
 => => transferring dockerfile: 2B                                                                                         0.0s
```


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
add04f4a2d4
weatherforecast:lat
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