Devops – Final Assessment

Section 1: Multiple-Choice Questions (MCQs)

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

-Windows Subsystem for Linux

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

-Frequent integration of code changes

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

-Change the working directory

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

-Docker

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

-Packaging and deploying applications in containers

6. What is the primary purpose of Azure DevOps?

-Software development and delivery

7. Which components are part of Azure DevOps?

-Azure Boards and Azure Pipelines

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

-It tracks changes in source code and manages versions.

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

-Administrative tasks with superuser privileges

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

applications?

-Azure Pipelines

Lab 1: File and Directory Management

Objective: Practice basic file and directory management commands.

Tasks:

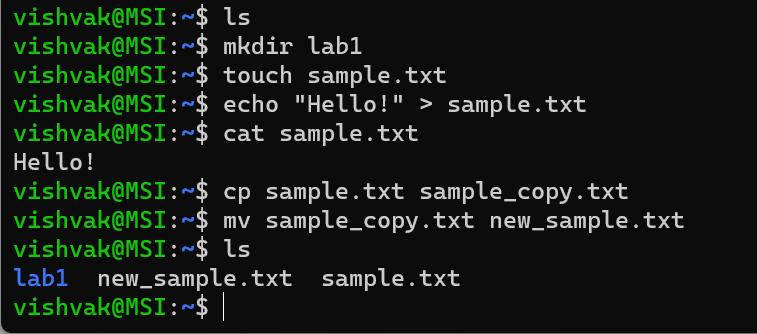
1. Create a directory called &quot;lab1&quot; in your home directory.

2. Inside &quot;lab1,&quot; create a text file named &quot;sample.txt&quot; with some content.

3. Make a copy of &quot;sample.txt&quot; and name it &quot;sample\_copy.txt.&quot;

4. Rename &quot;sample\_copy.txt&quot; to &quot;new\_sample.txt.&quot;

5. List the files in the &quot;lab1&quot; 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 &quot;secret.txt&quot; in the &quot;lab2&quot; directory.

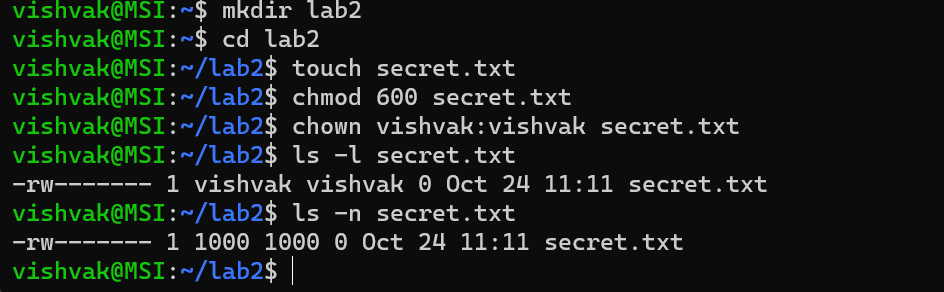
2. Set the file permissions to allow read and write access only to the

owner.

3. Change the owner of &quot;secret.txt&quot; to another user.

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

commands.



Lab 3: Text Processing with Command LineTools

Objective: Practice text processing using command-line tools.

Tasks:

1. Create a text file with some random text in the &quot;lab3&quot; directory.

2. Use the grep command to search for a specific word or pattern in the

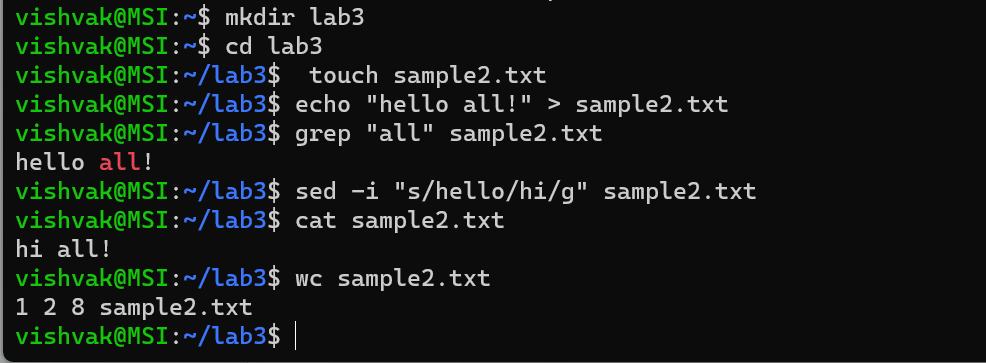
file.

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:

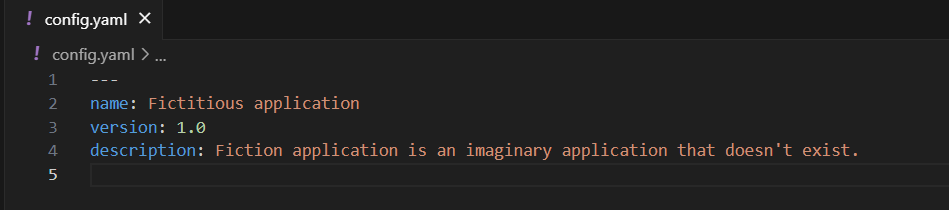
1. Create a YAML file named &quot;config.yaml.&quot;

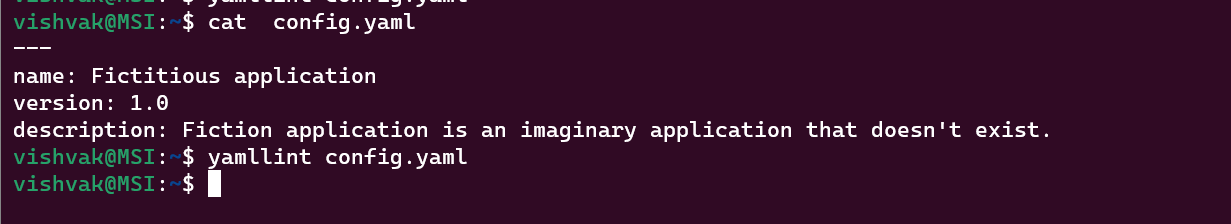
2. Define key-value pairs in YAML for a fictitious application, including

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.

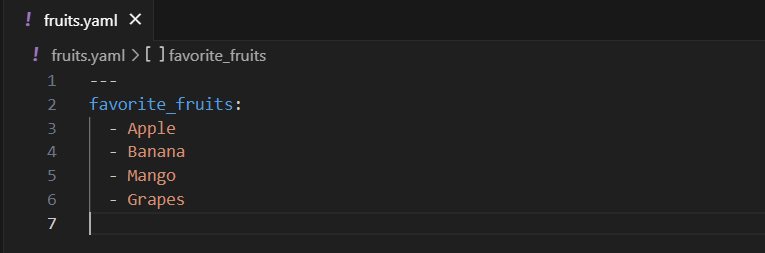
Task:

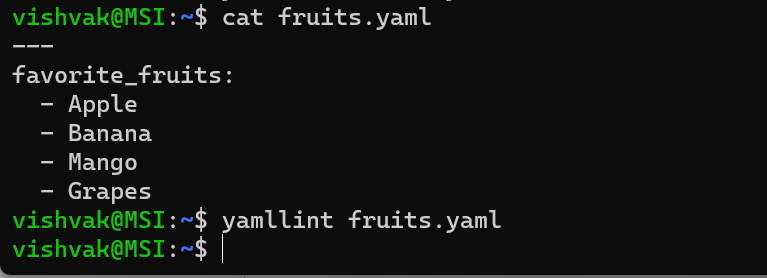
1. Create a YAML file named &quot;fruits.yaml.&quot;

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 &quot;data.yaml.&quot;

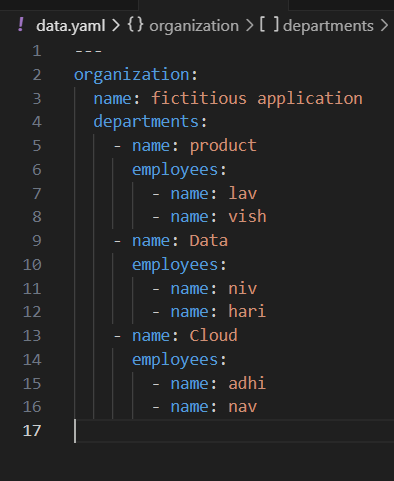
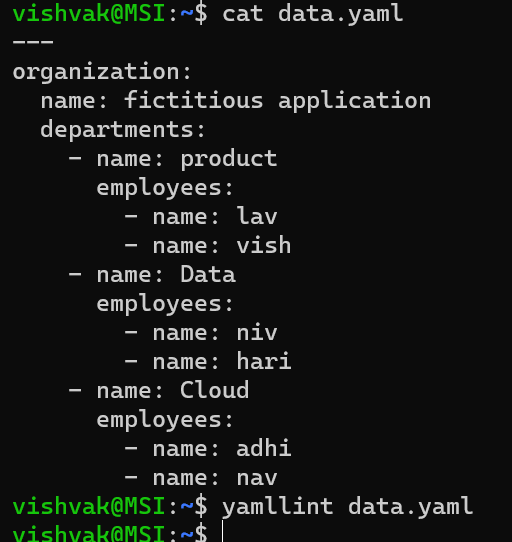
2. Define a nested structure representing a fictitious organization with

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.

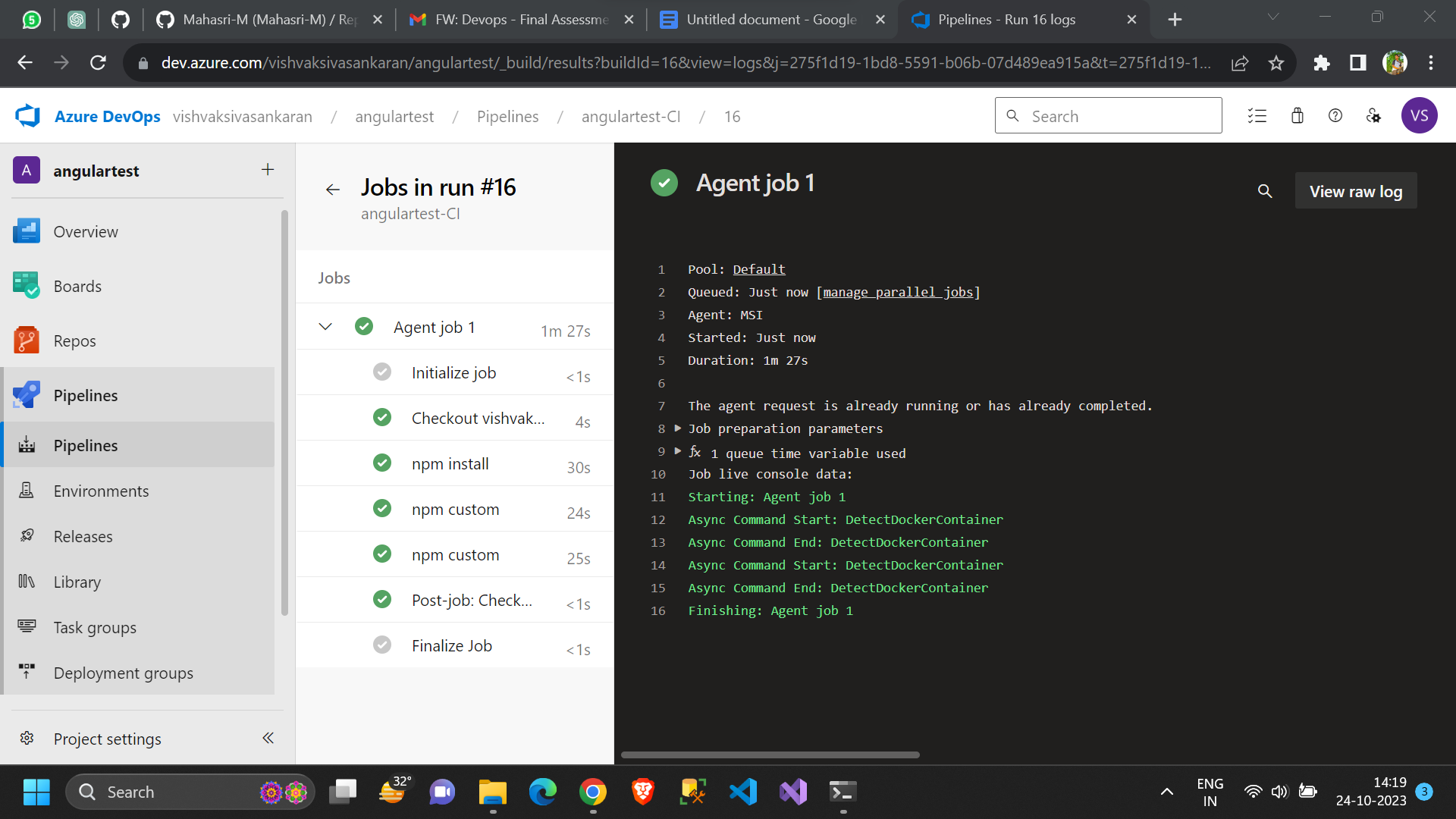
Tasks:

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.



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.

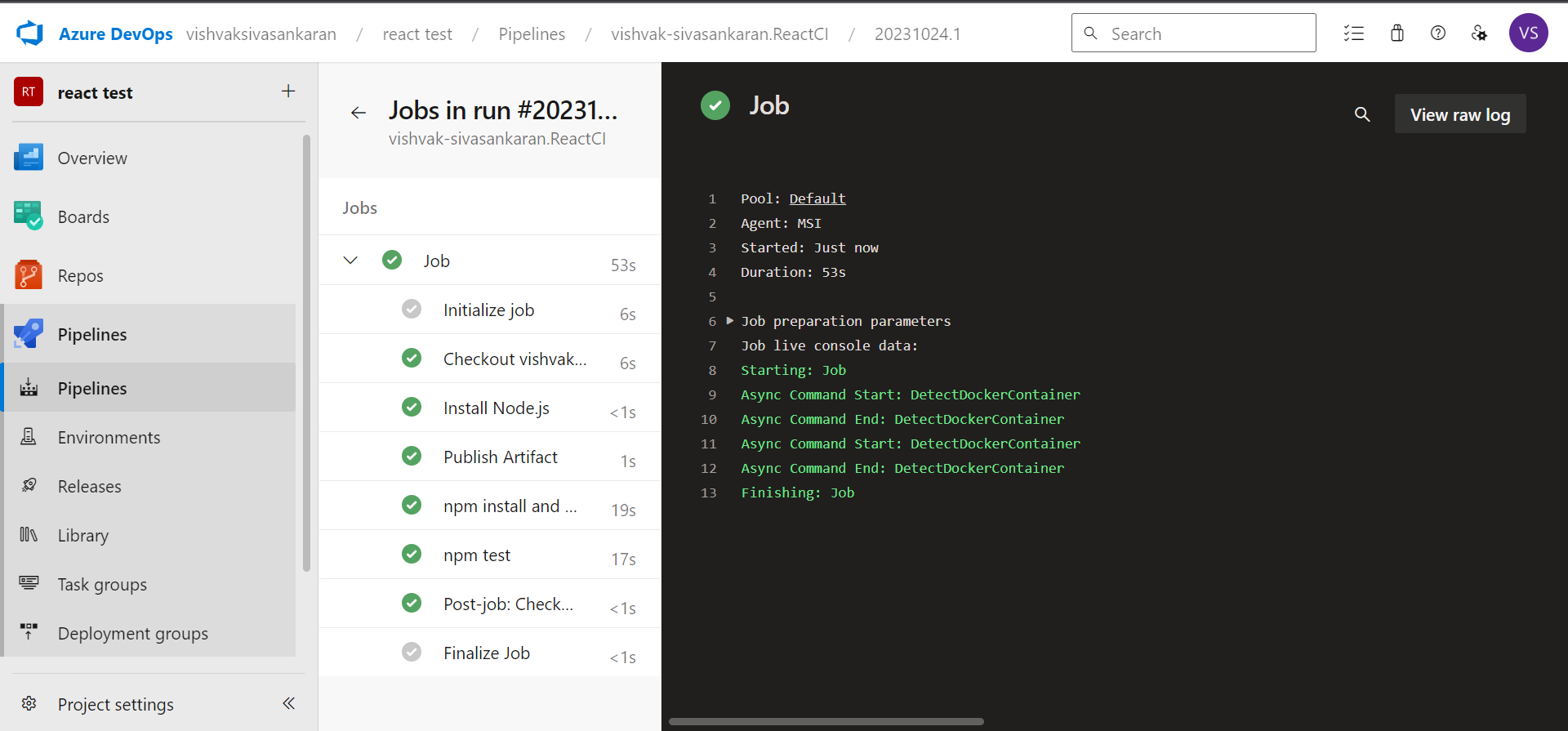
Tasks:

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.



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.

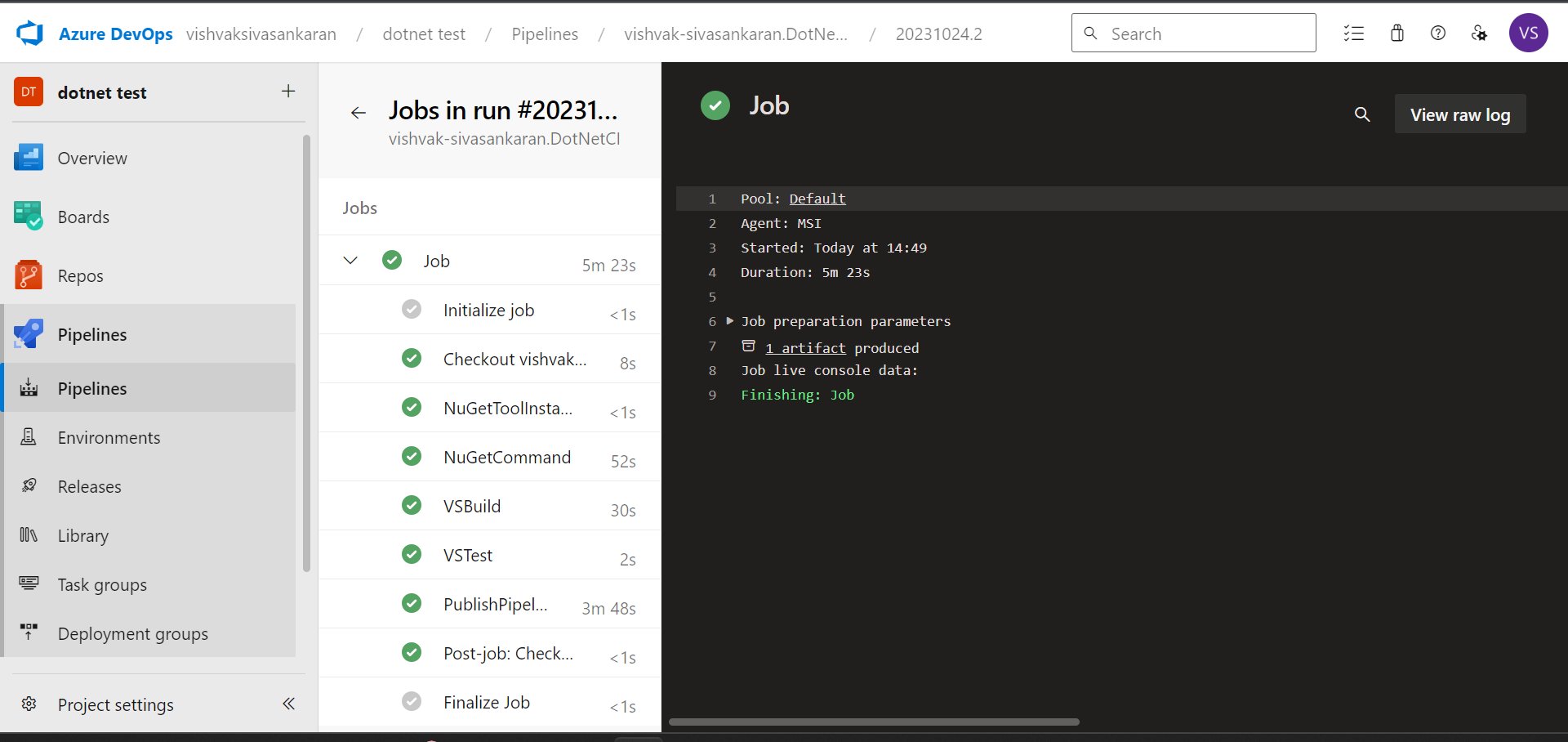
Tasks:

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

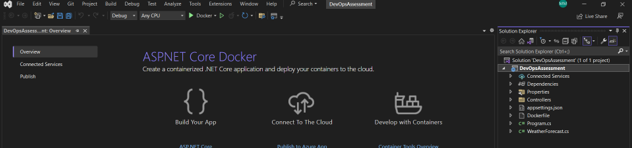
Step 3: Dockerize the .NET Core Web API

Step 4: Build the Docker Image

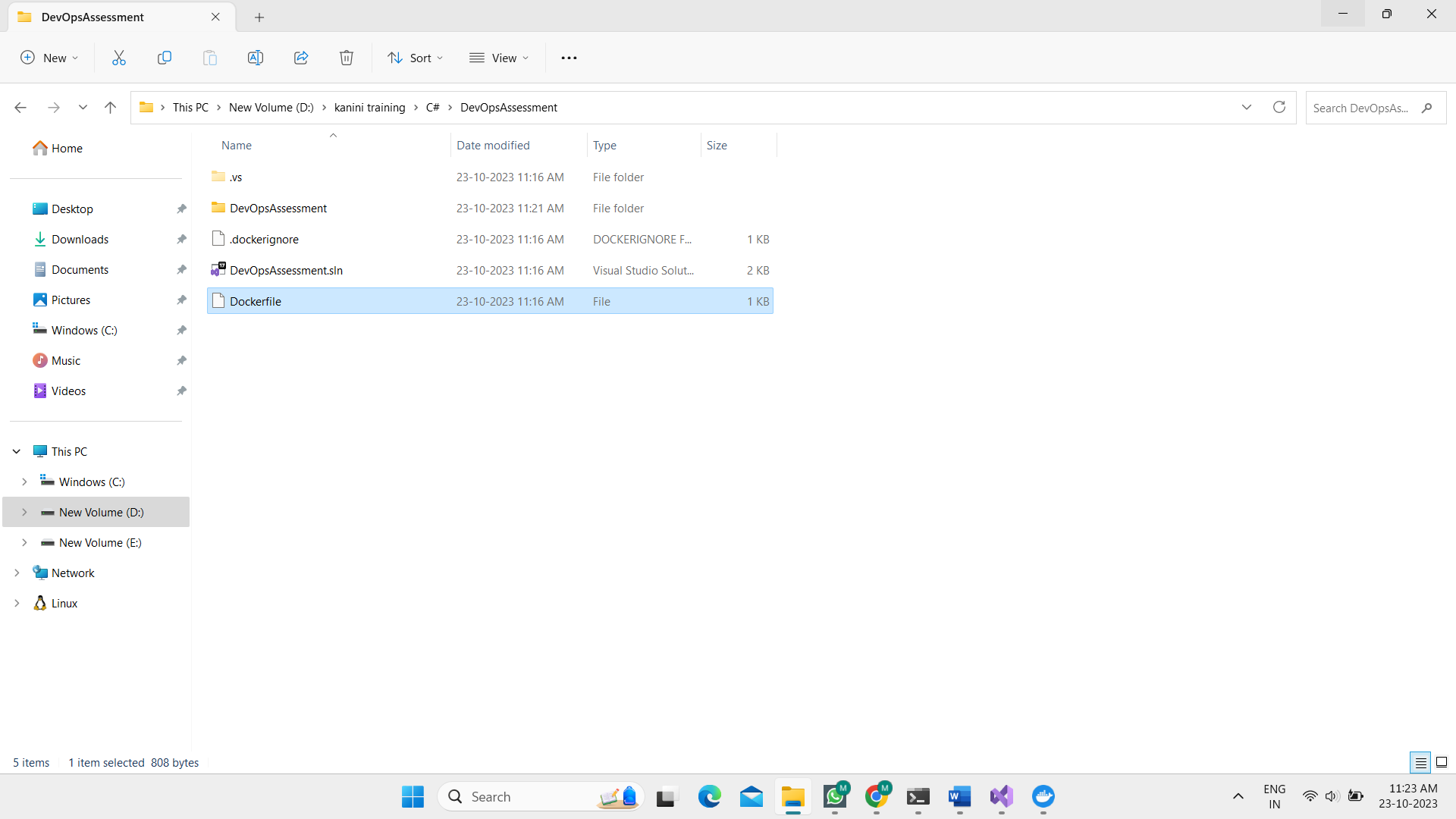
Step 5: Run the Docker Container in Rancher Desktop

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

Step 1: Create a .NET Core Web API

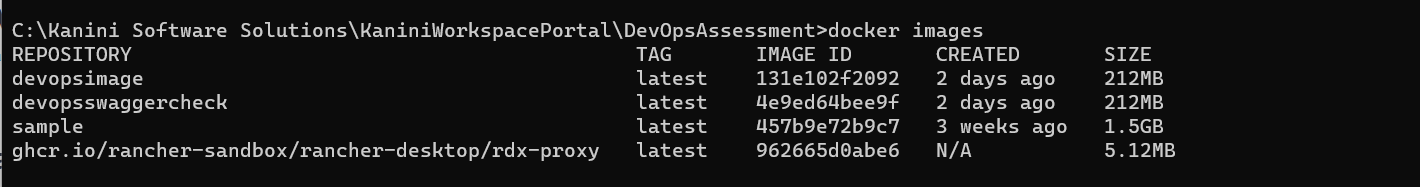


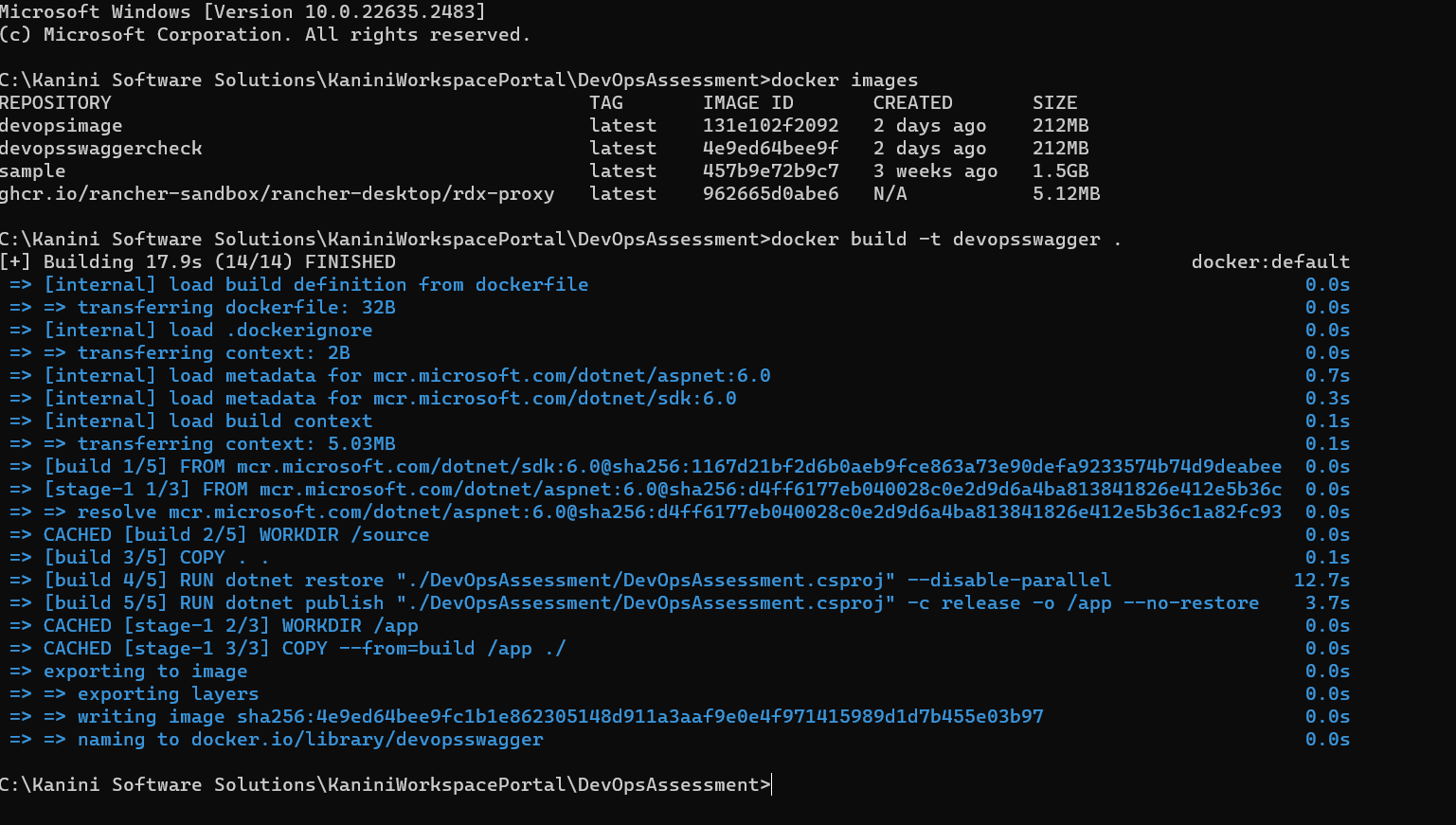
Step 2 & 3: Building and Dockerize .NET Core Web API



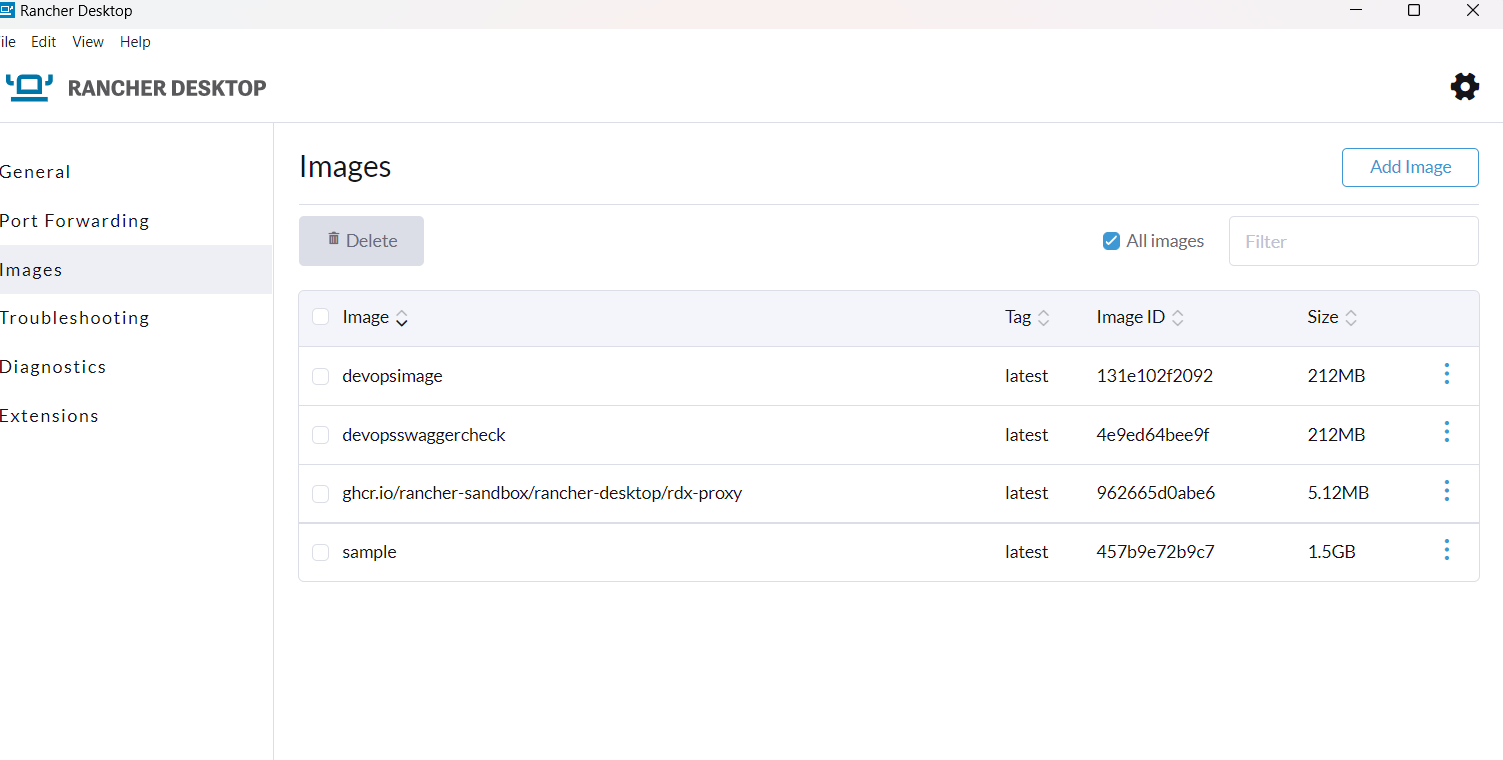
Step 4: Build Docker Image

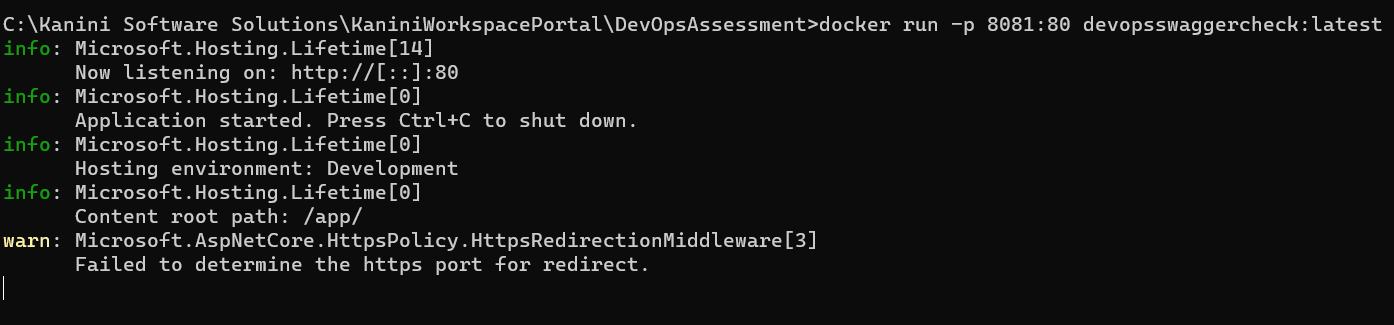
Listing all images:



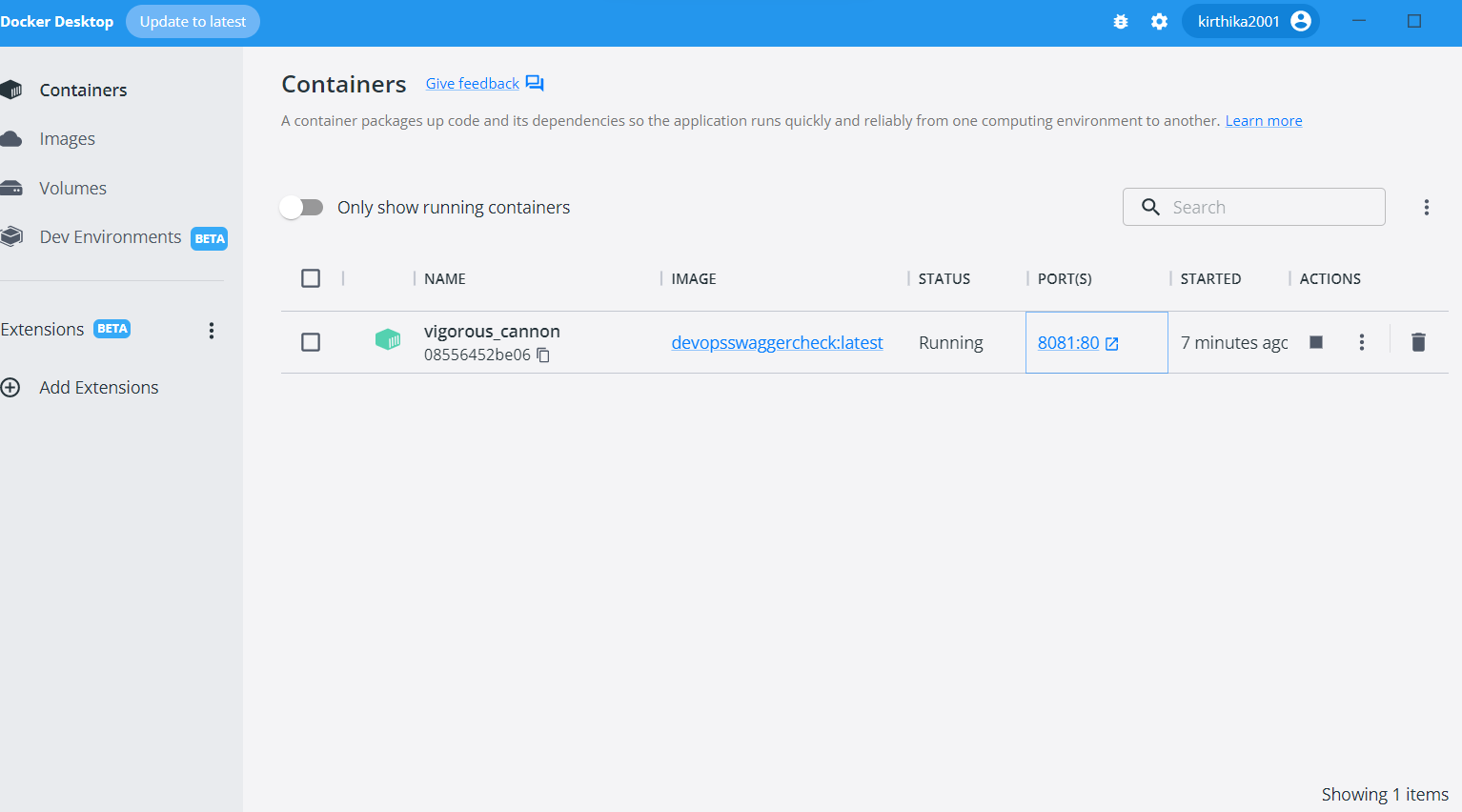
Building the docker file

Step 5: Run the Docker Container in Rancher Desktop

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



Runs in swagger via docker



Net application runs in swagger via docker and rancher

