**Devops – Final Assessment**

**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  
  
**Answer: (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  
  
**Answer: (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

**Answer: (b) Change the working directory**

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

a. Ubuntu

b. CentOS

c. Docker

d. Debian  
  
**Answer: (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

**Answer : (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  
**Answer: (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

**Answer: (c) 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.

**Answer: (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

**Answer: (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

**Answer: (c) Azure Pipelines  
  
  
  
Section 2: Labs**

**Lab 1: File and Directory Management**

Tasks:

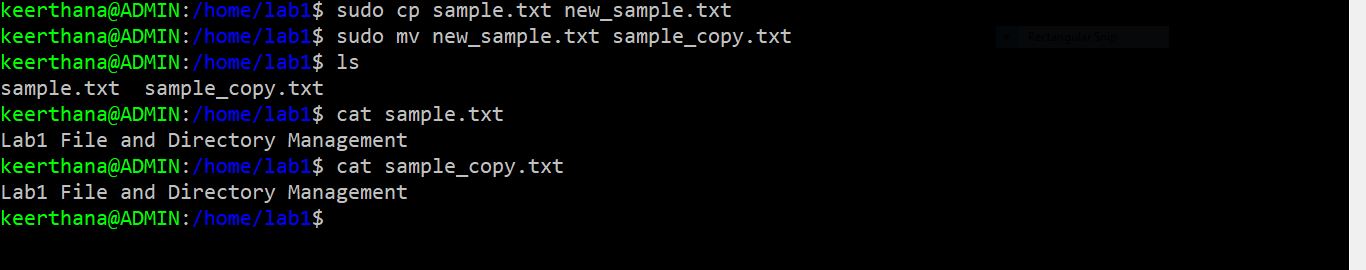
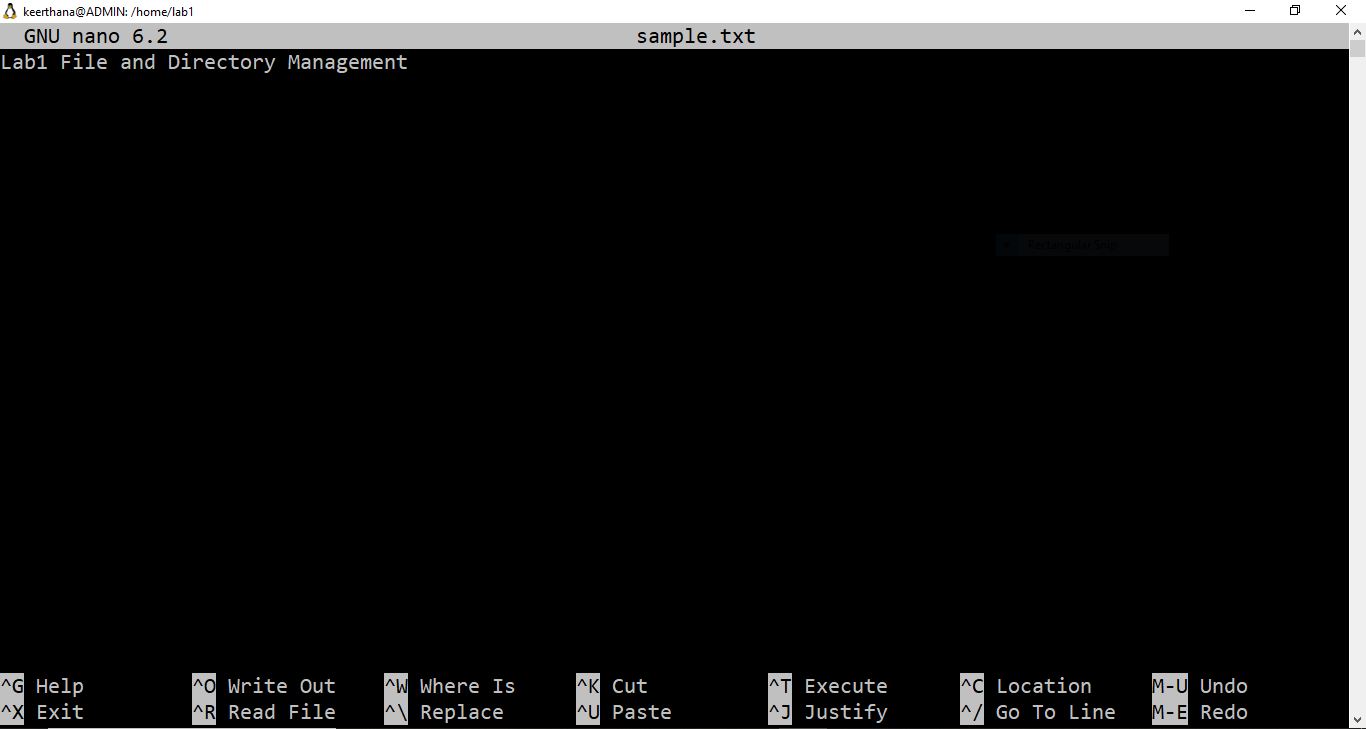
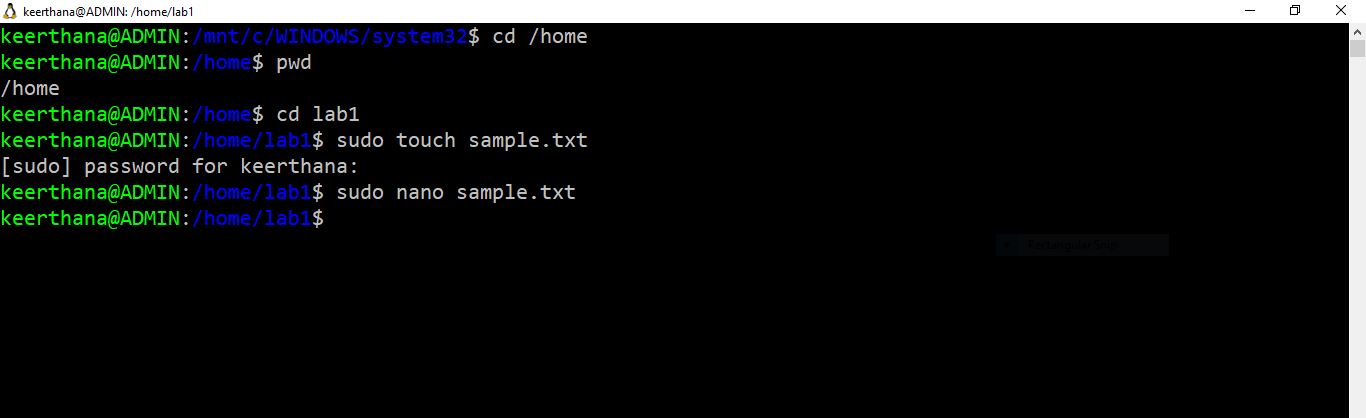
Step 1: Create a directory called "lab1" in your home directory.

Step 2: Inside "lab1," create a text file named "sample.txt" with some content.

Step 3: Make a copy of "sample.txt" and name it "sample\_copy.txt."

Step 4: Rename "sample\_copy.txt" to "new\_sample.txt."

Step 5: List the files in the "lab1" directory to confirm their names

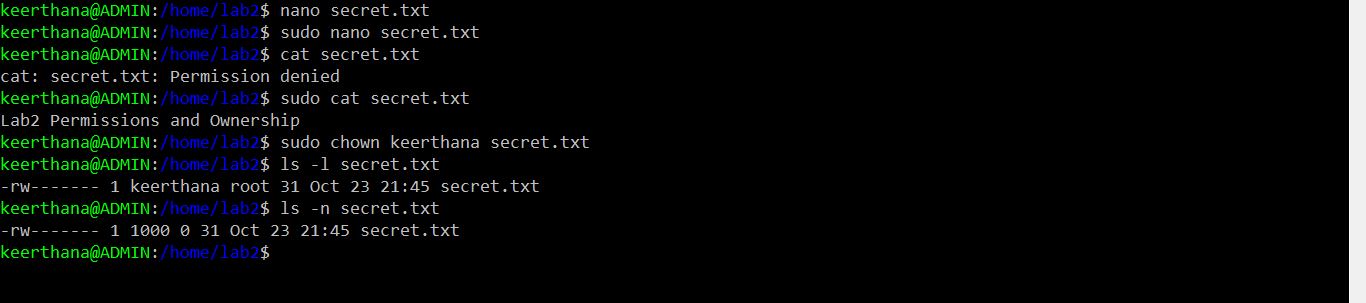
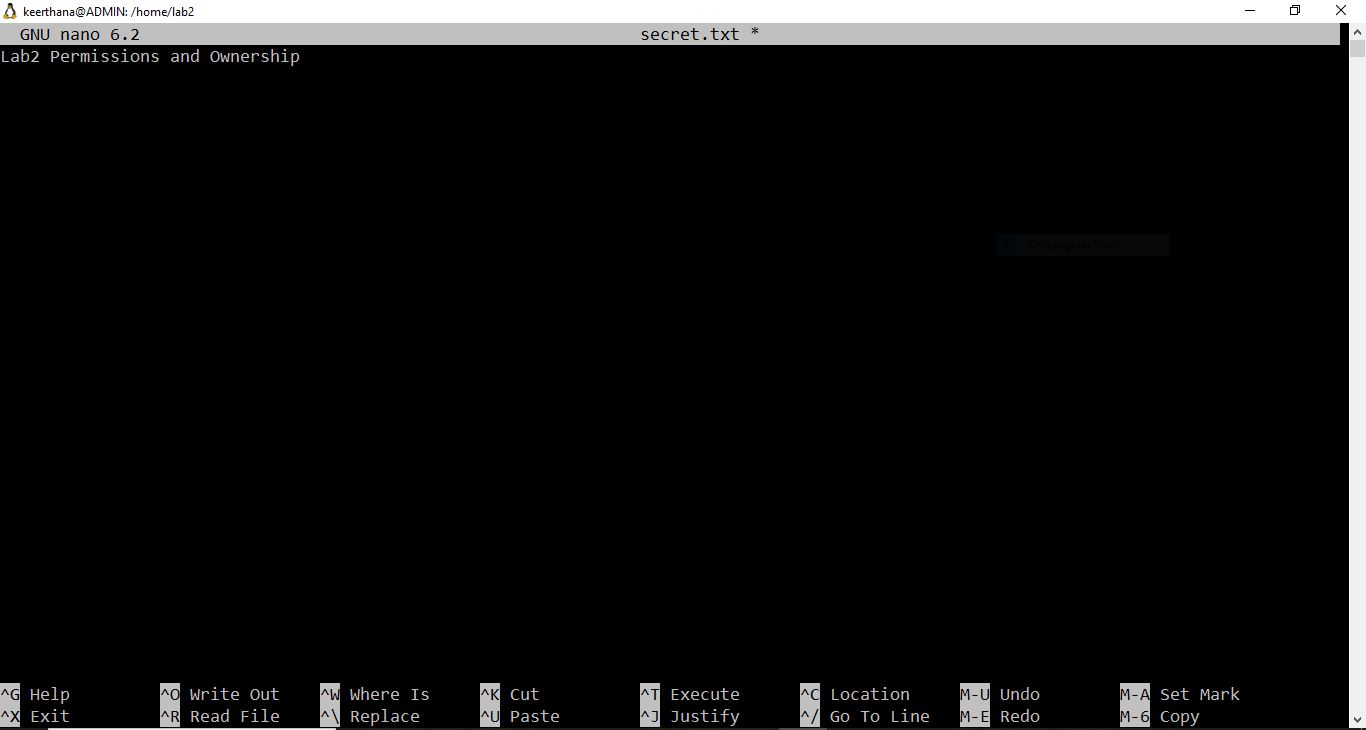
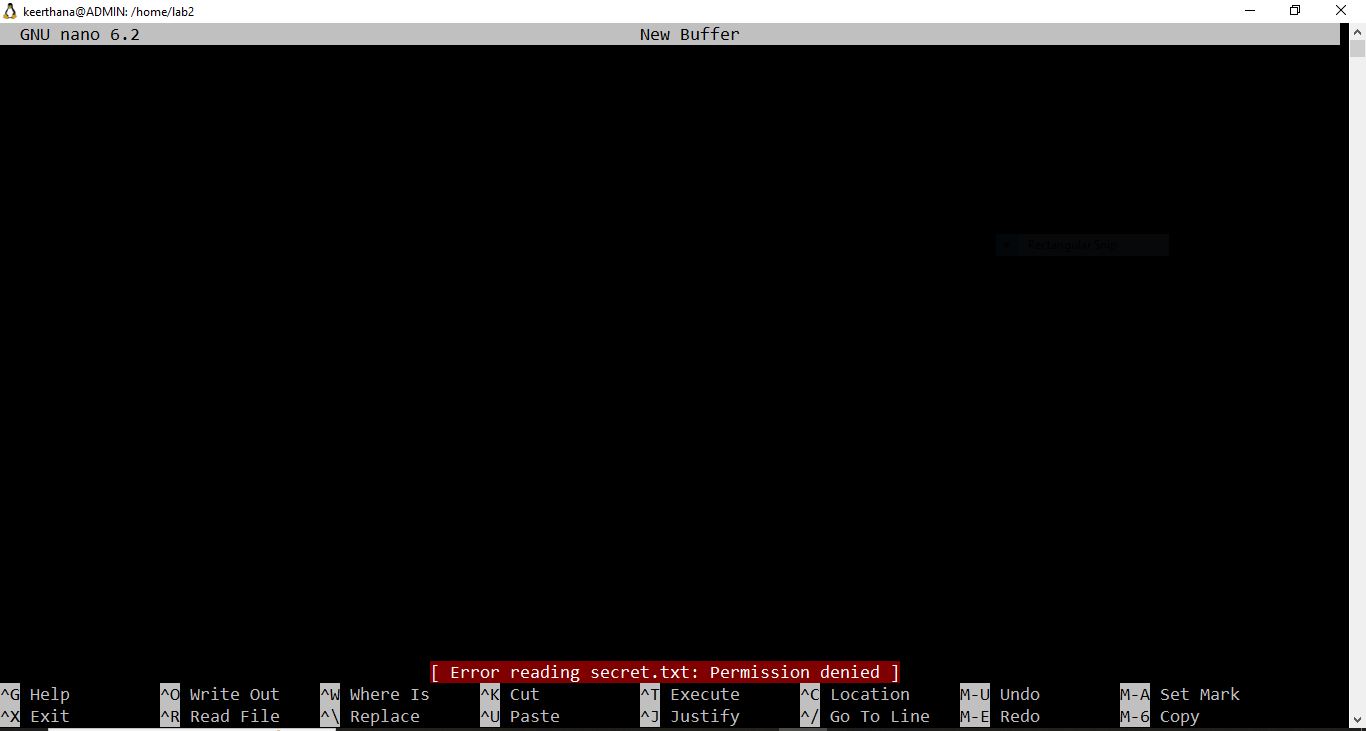
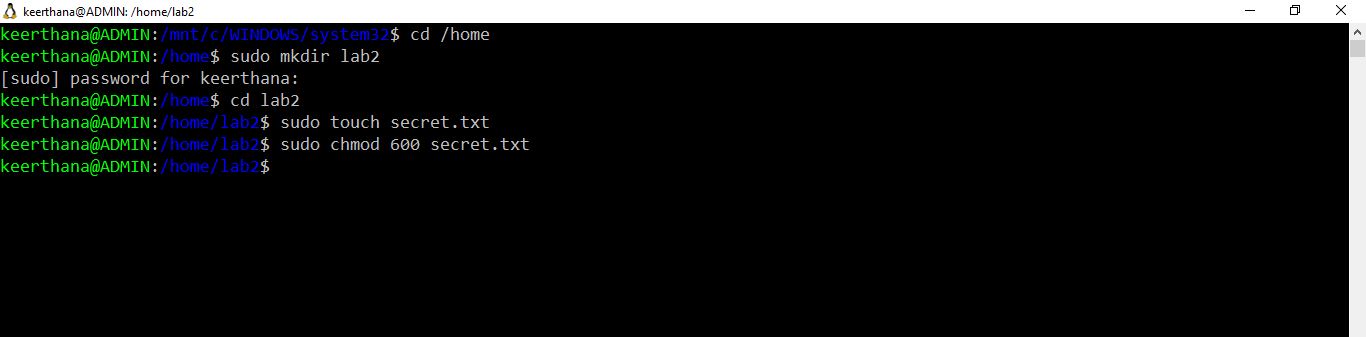
**  
  
  
  
Lab 2: Permissions and Ownership**

Tasks:

Step 1: Create a new file named "secret.txt" in the "lab2" directory.

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

Step 3: Change the owner of "secret.txt" to another user.

Step 4: Verify the new permissions and owner using the ls -l and ls -n commands.  
  
 **Lab 3: Text Processing with Command Line Tools**

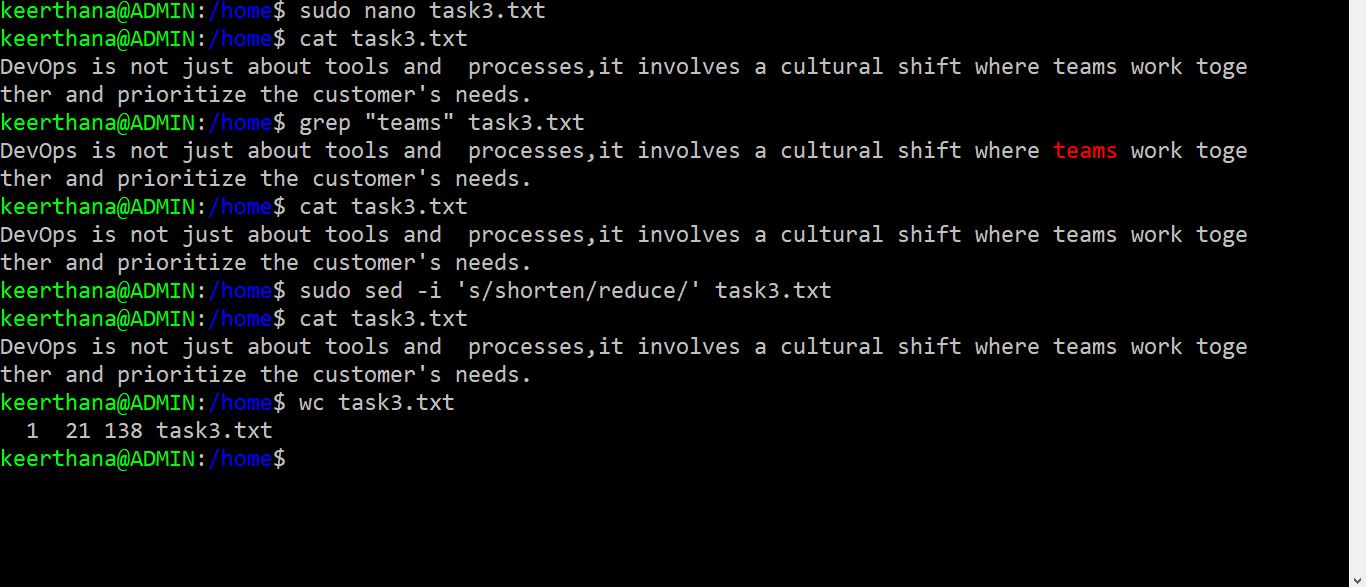
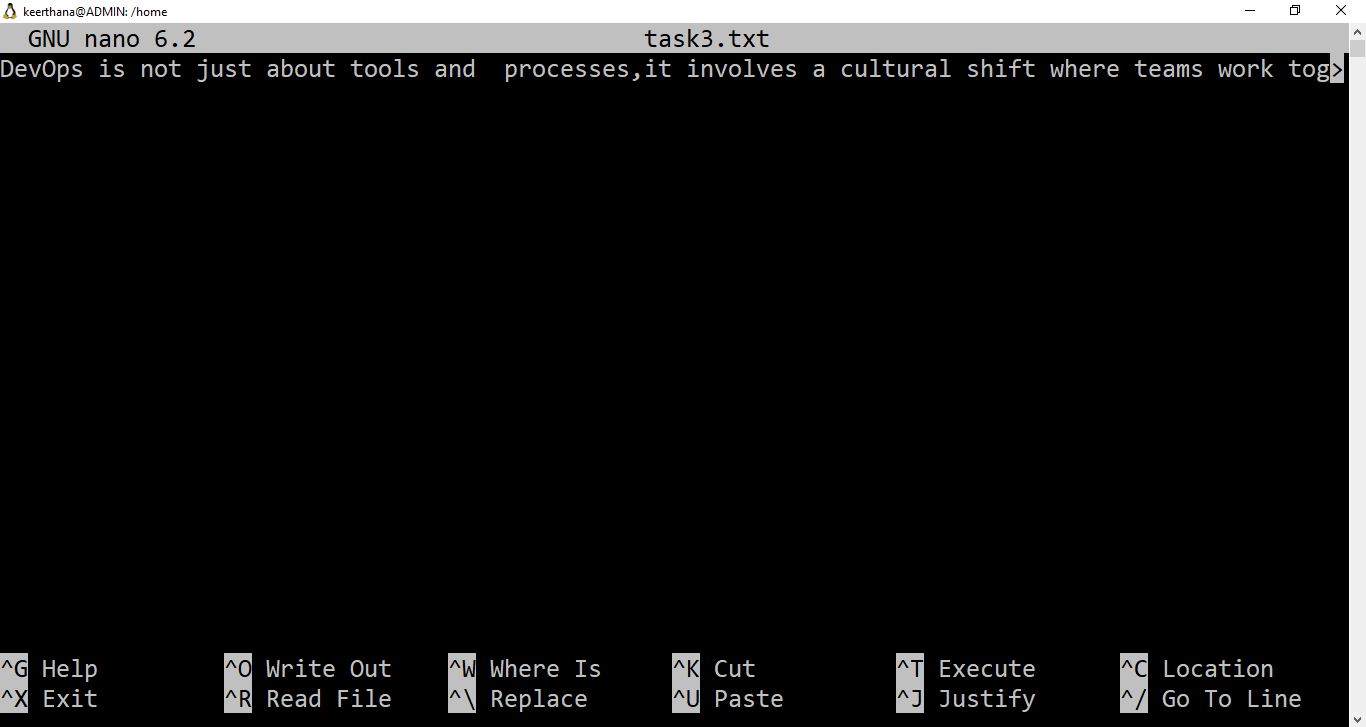
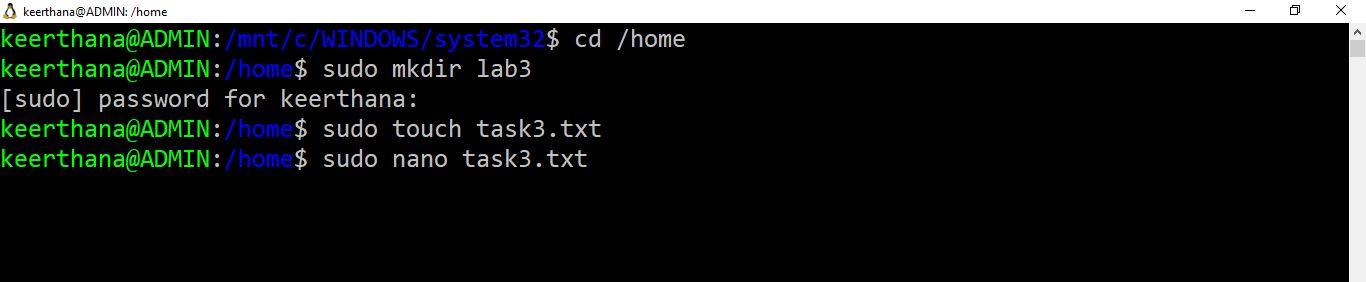
Tasks:

Step 1: Create a text file with some random text in the "lab3" directory.

Step 2: Use the ‘grep’ command to search for a specific word or pattern in the file.

Step 3: Use the ‘sed’ command to replace a word or phrase with another in the file.

Step 4: Use the ‘wc’ command to count the number of lines, words, and characters in the file.

****

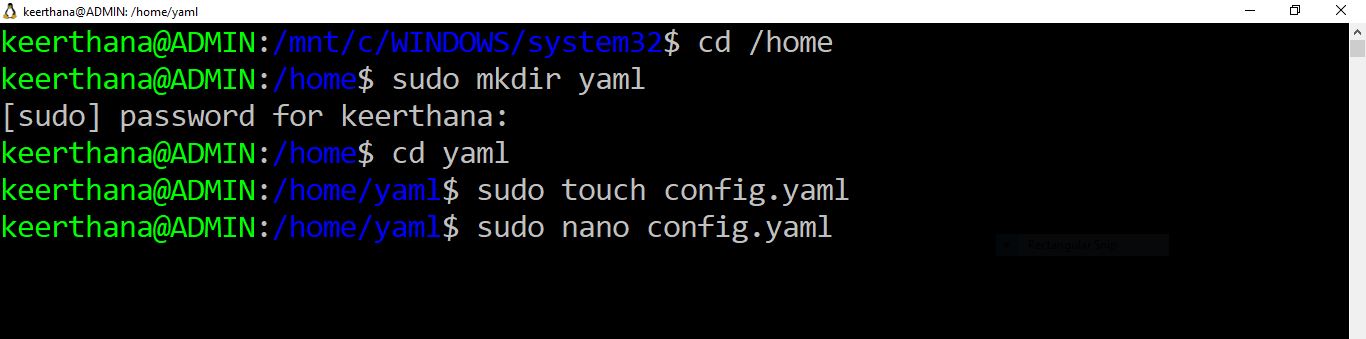
**Lab 4: Creating a Simple YAML File**

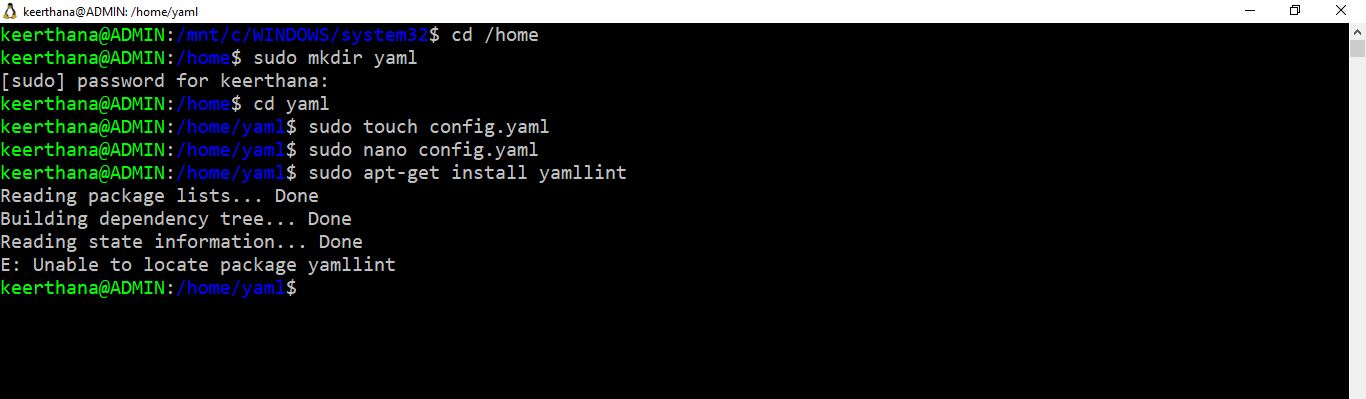
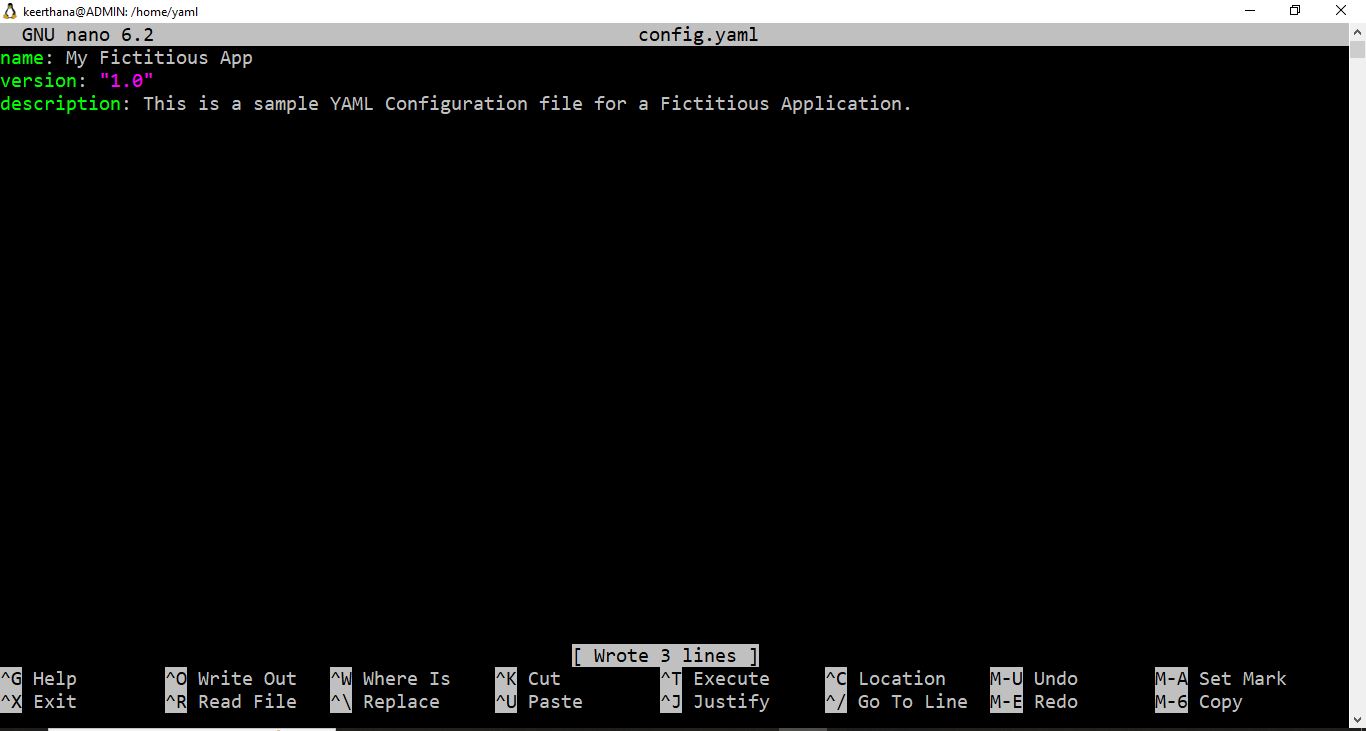
Task:

Step 1: Create a YAML file named "config.yaml."

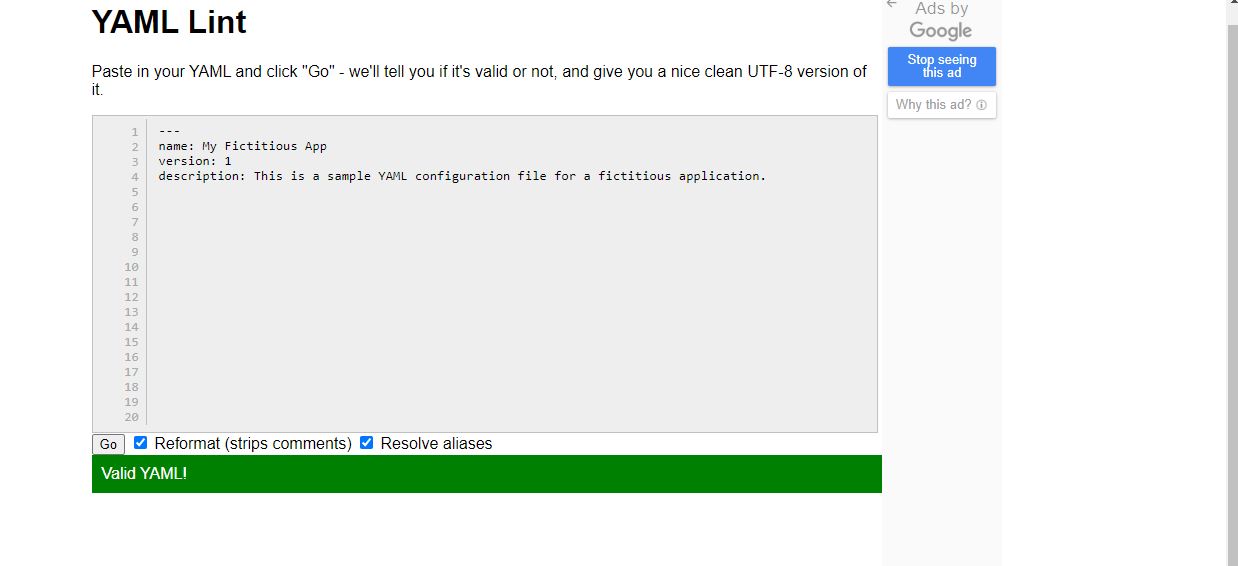
Step 2: Define key-value pairs in YAML for a fictitious application, including name, version, and description.

Step 3: Save the file.

Step 4: Validate that the YAML file is correctly formatted.



Online Editor:



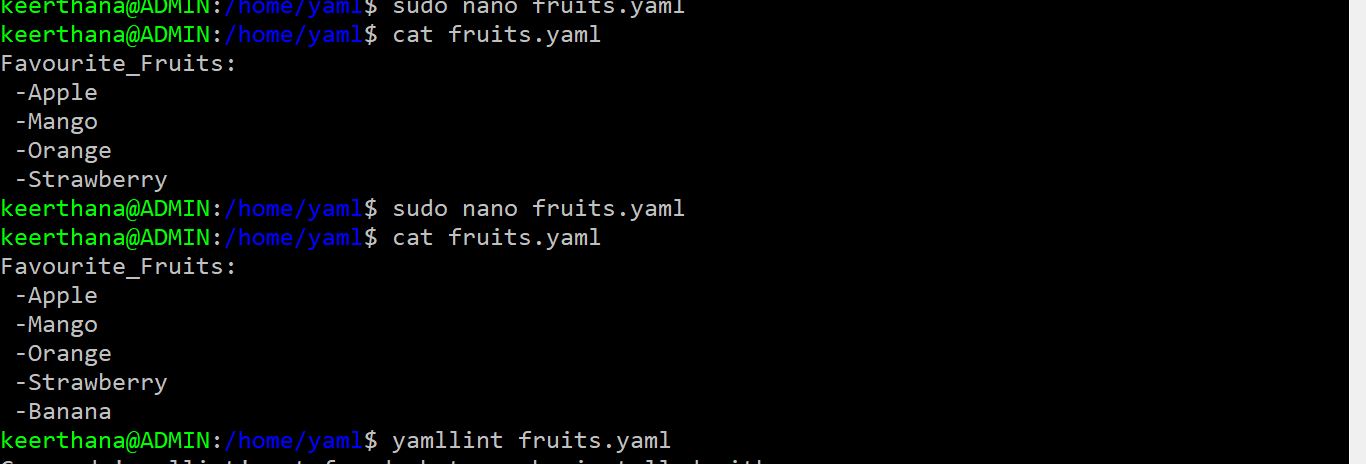
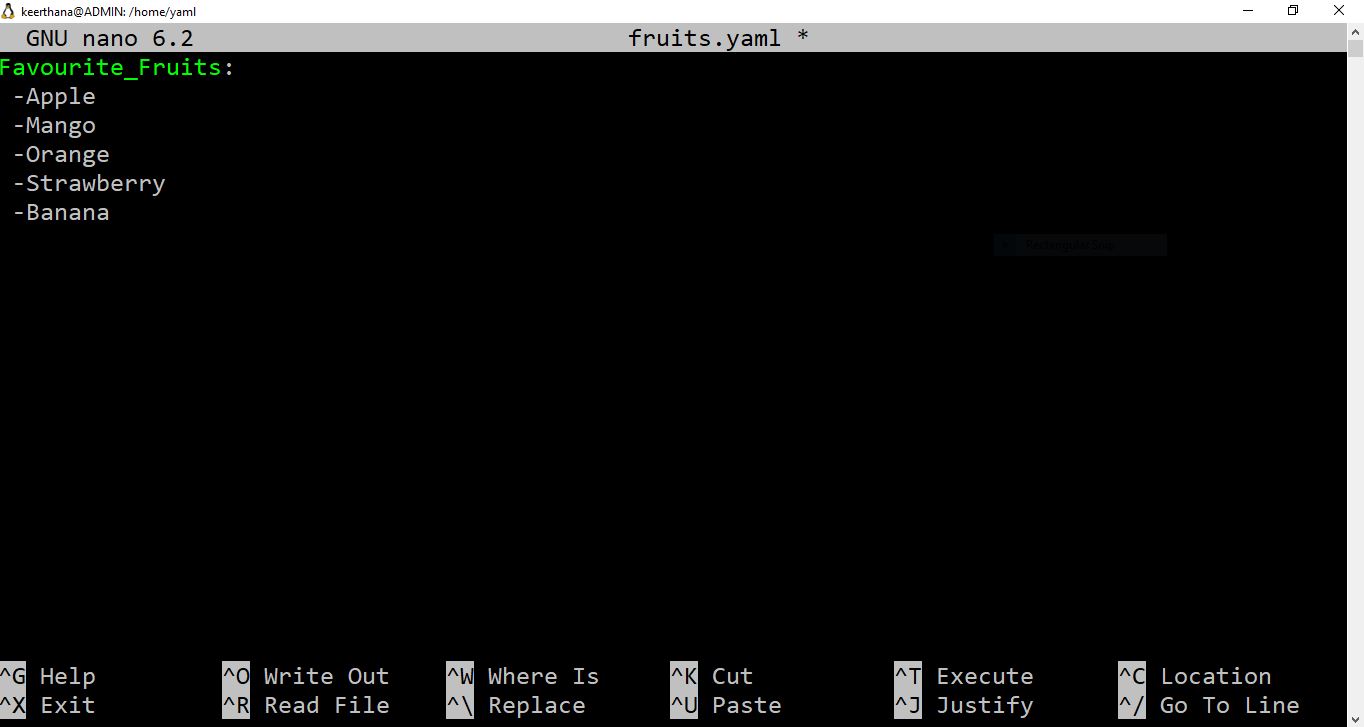
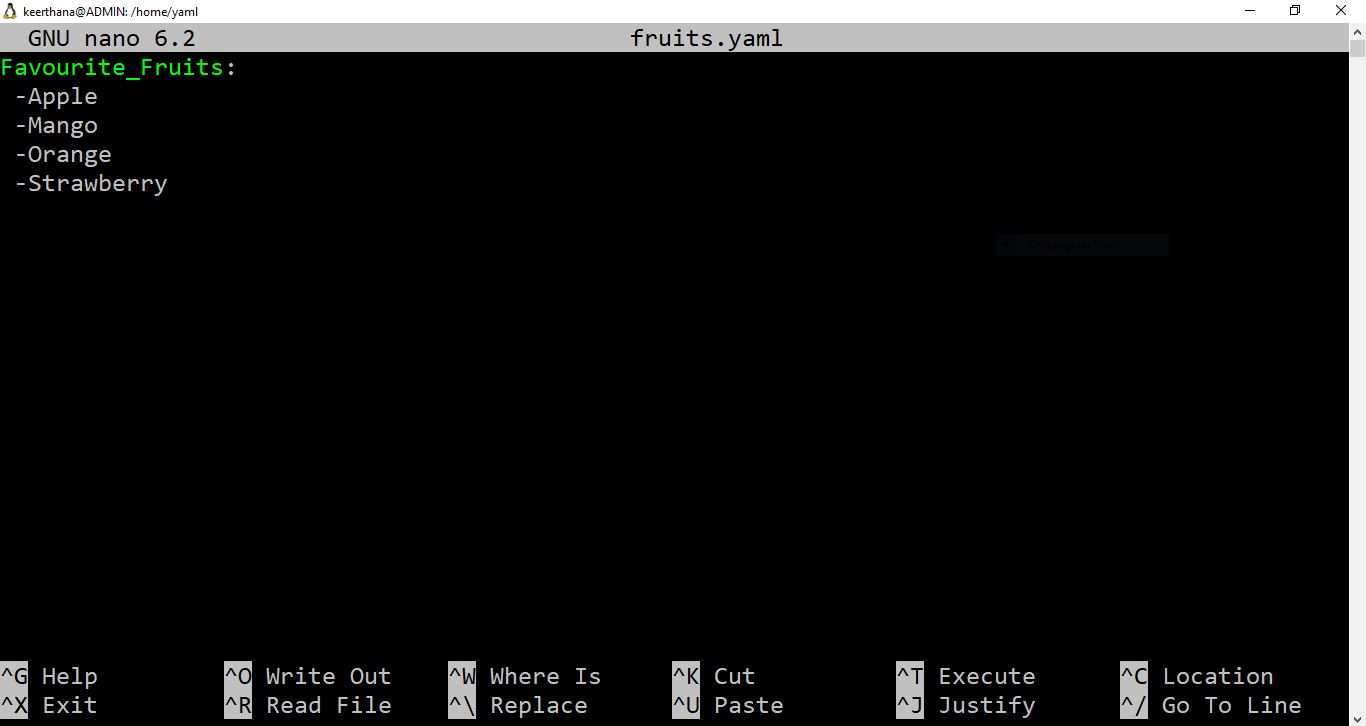
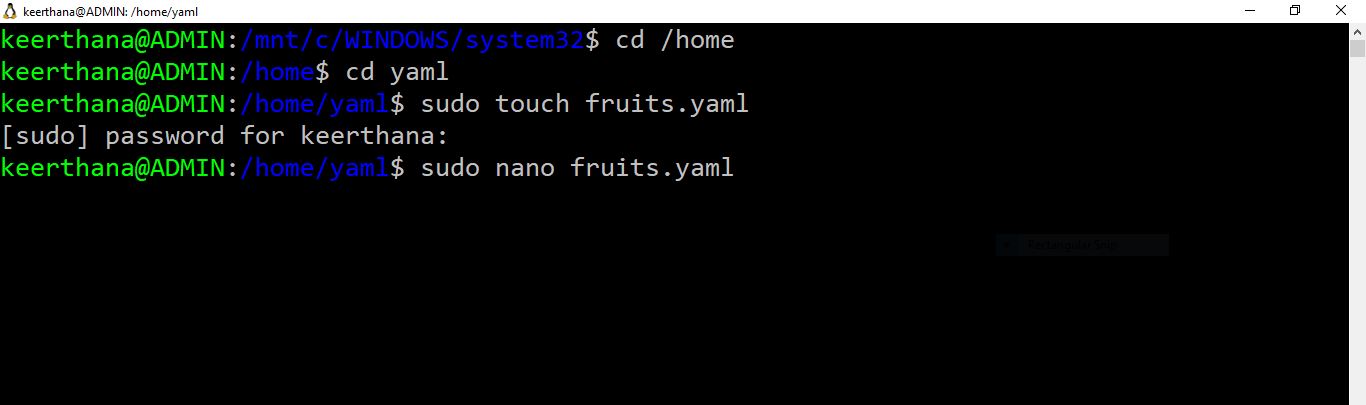
**Lab 5: Working with Lists in YAML**

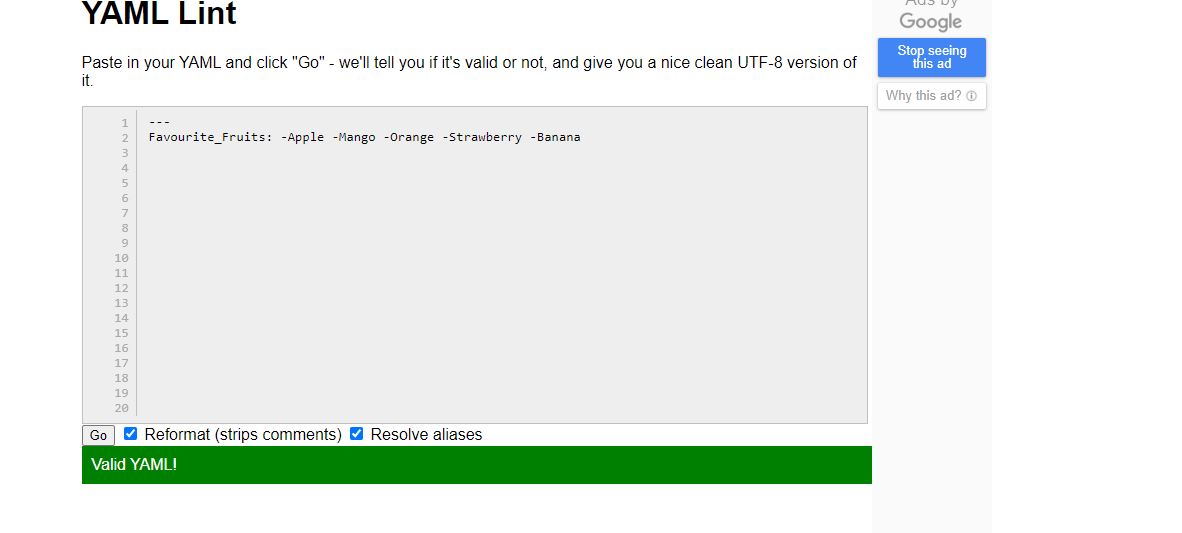
Task:

Step 1: Create a YAML file named "fruits.yaml."

Step 2: Define a list of your favorite fruits using YAML syntax.

Step 3: Add items to the list.

Step 4: Save and validate the YAML file.  
  


Online Editor:

**Lab 6: Nested Structures in YAML**

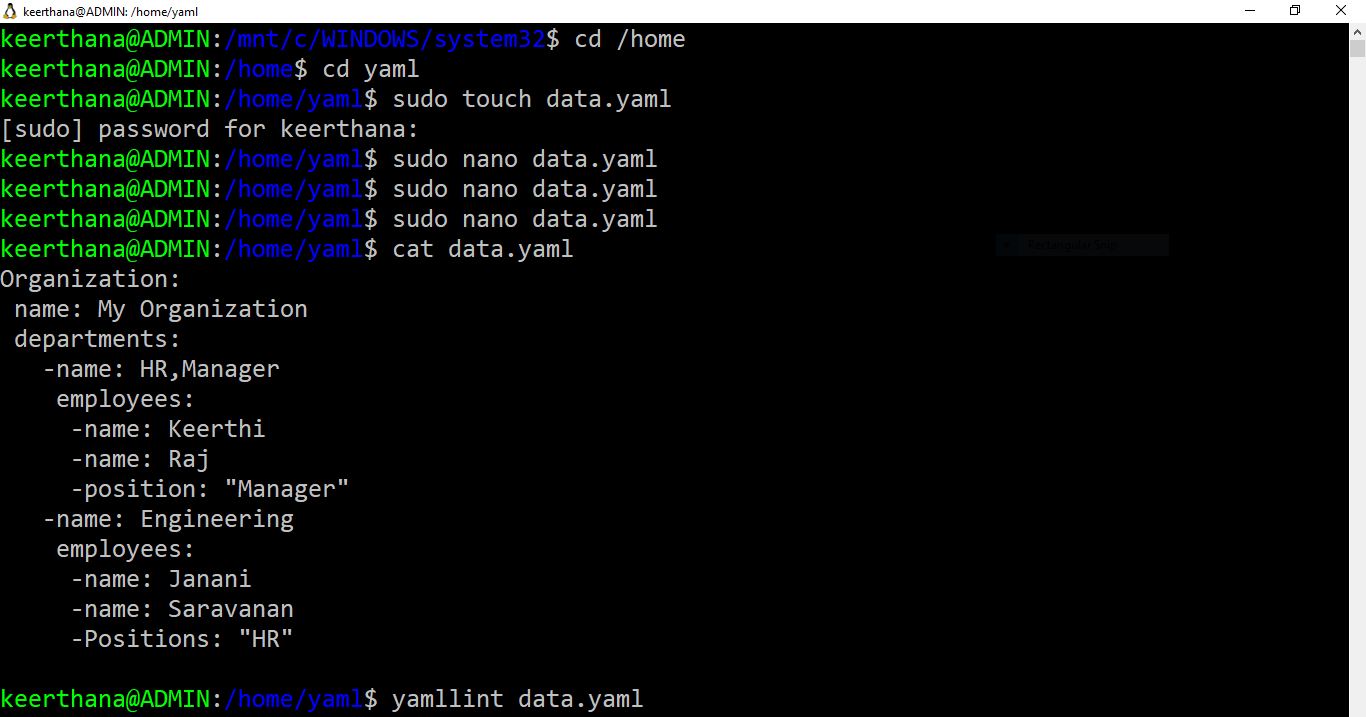
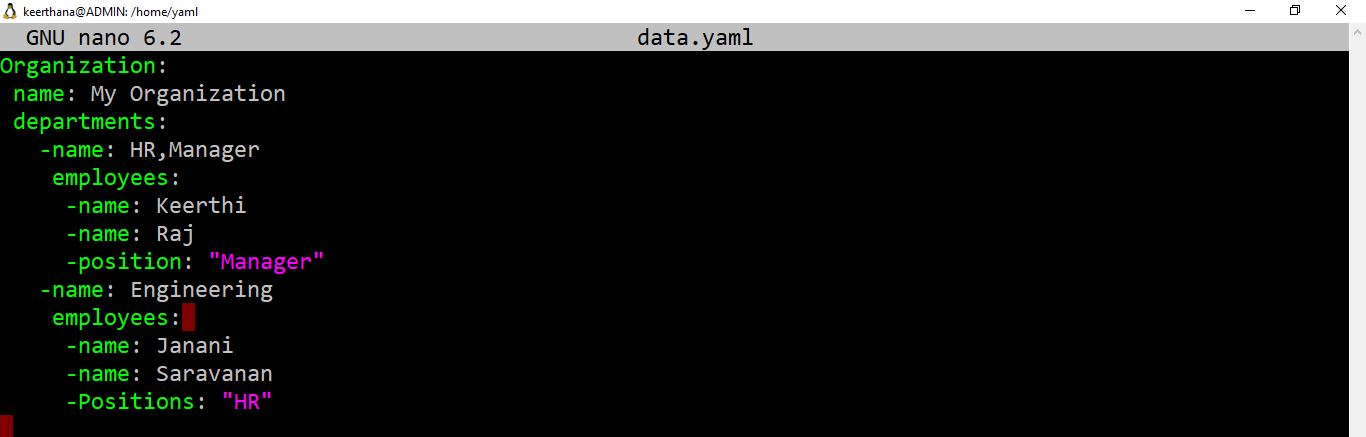
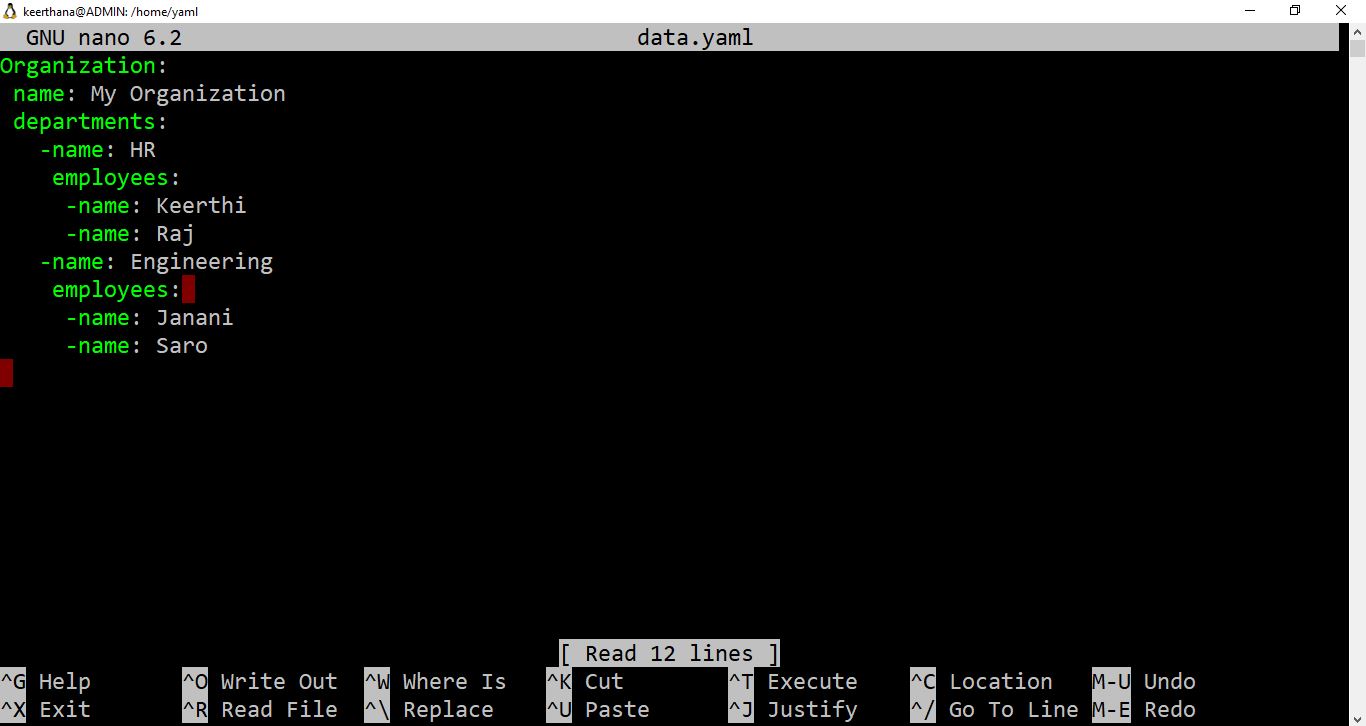
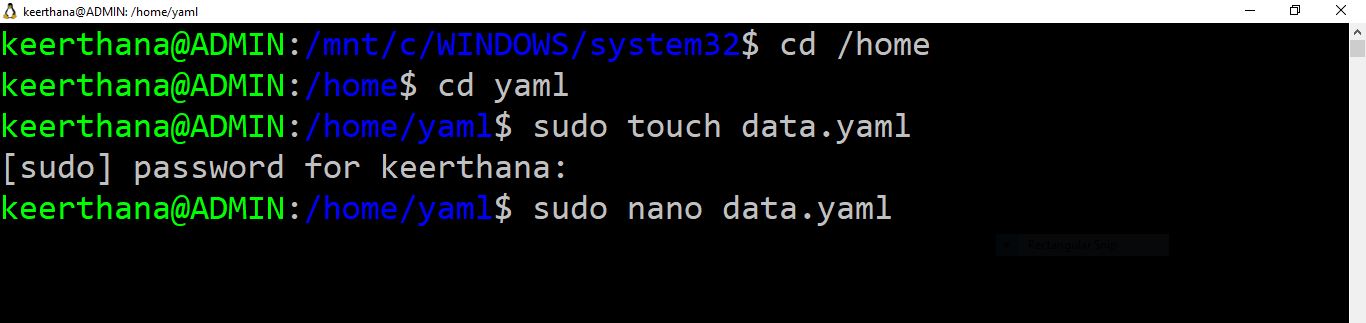
Task:

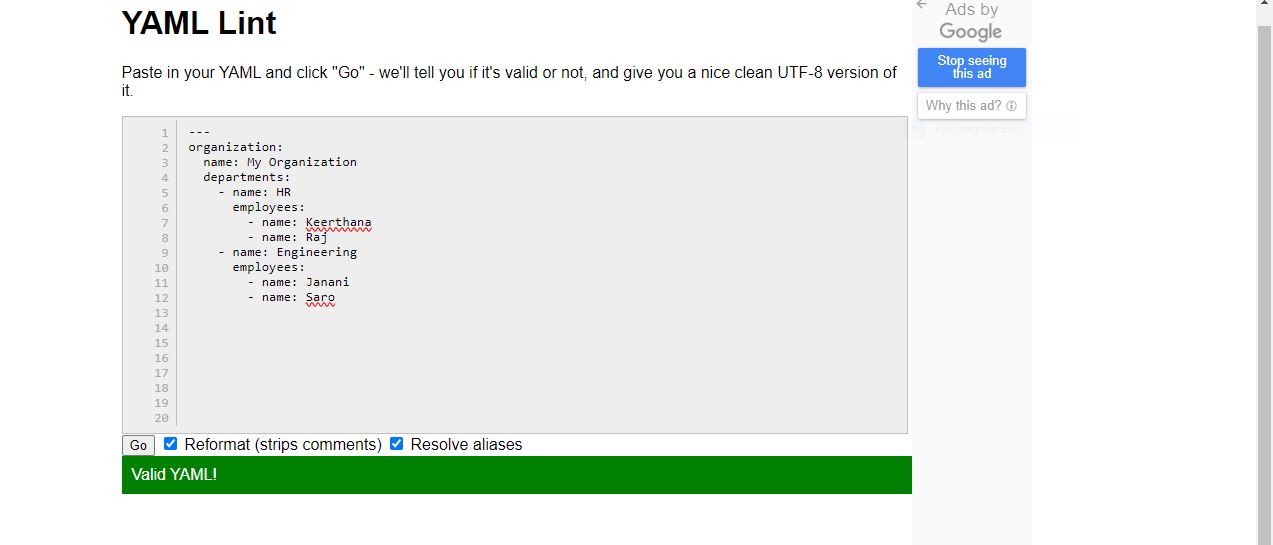
Step 1: Create a YAML file named "data.yaml."

Step 2: Define a nested structure representing a fictitious organization with departments and employees using YAML syntax.

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

Step 4: Save and validate the YAML file.





**Lab 7: Create Classic Azure CI Pipeline for Angular Application**

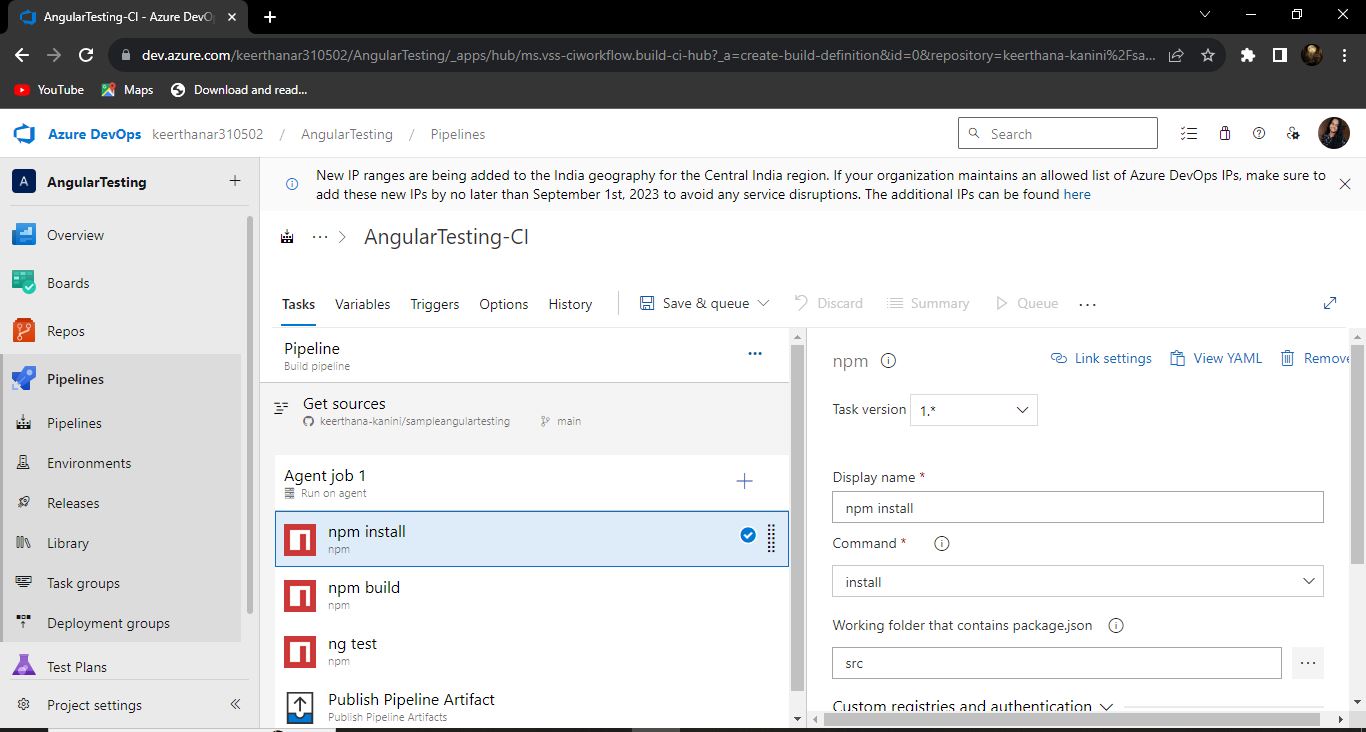
Tasks:

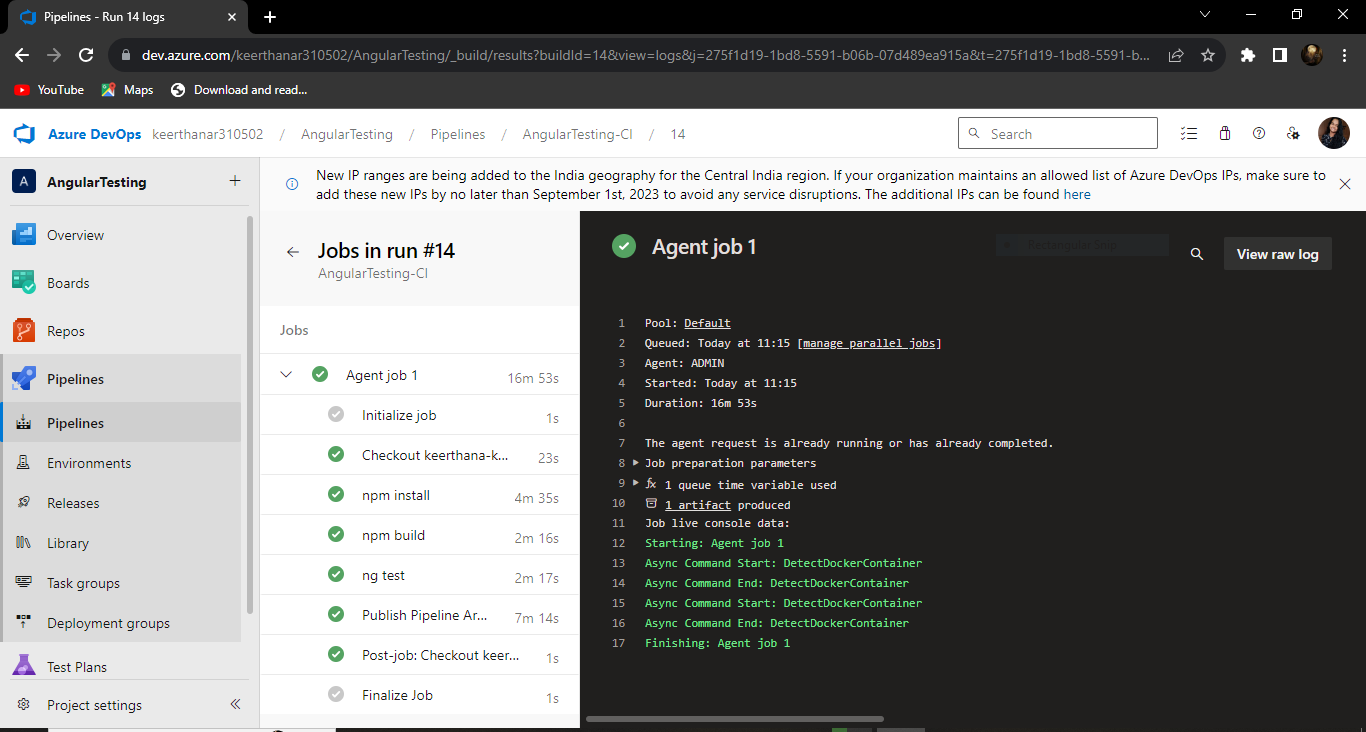
Step1. Create an Azure DevOps project.

Step 2. Set up a classic CI pipeline to build an Angular application.

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

Step 4. Run the pipeline and validate the test results.





**Lab 8: Create YAML Azure CI Pipeline for React Application**

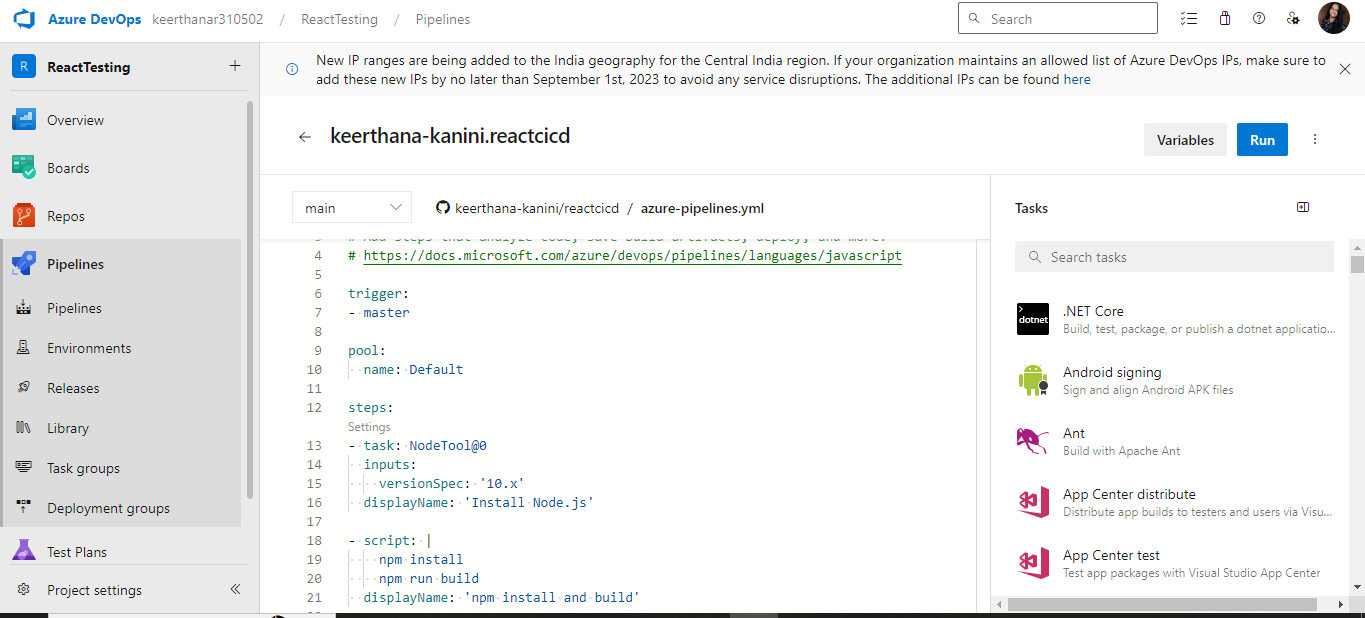
Tasks:

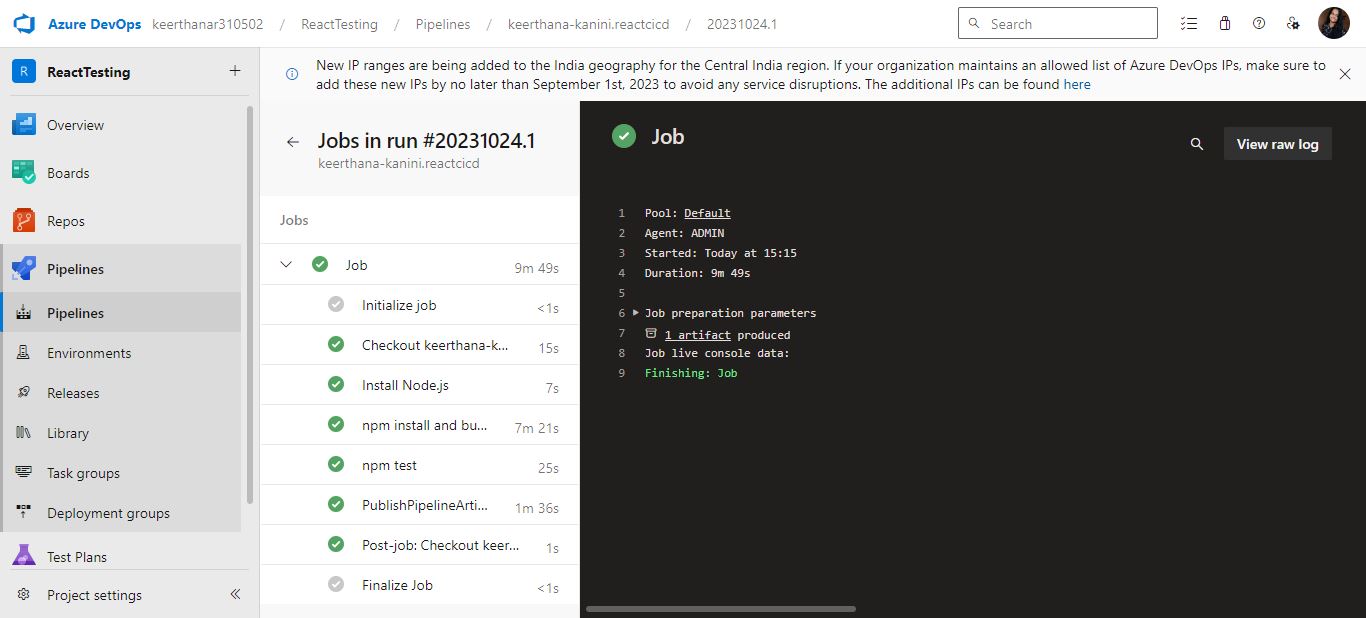
Step 1. Create an Azure DevOps project.

Step 2. Create a YAML-based CI pipeline to build a React application.

Step 3. Configure the pipeline to use Enzyme and Jest for unit testing.

Step 4. Trigger the pipeline and verify the test results.





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

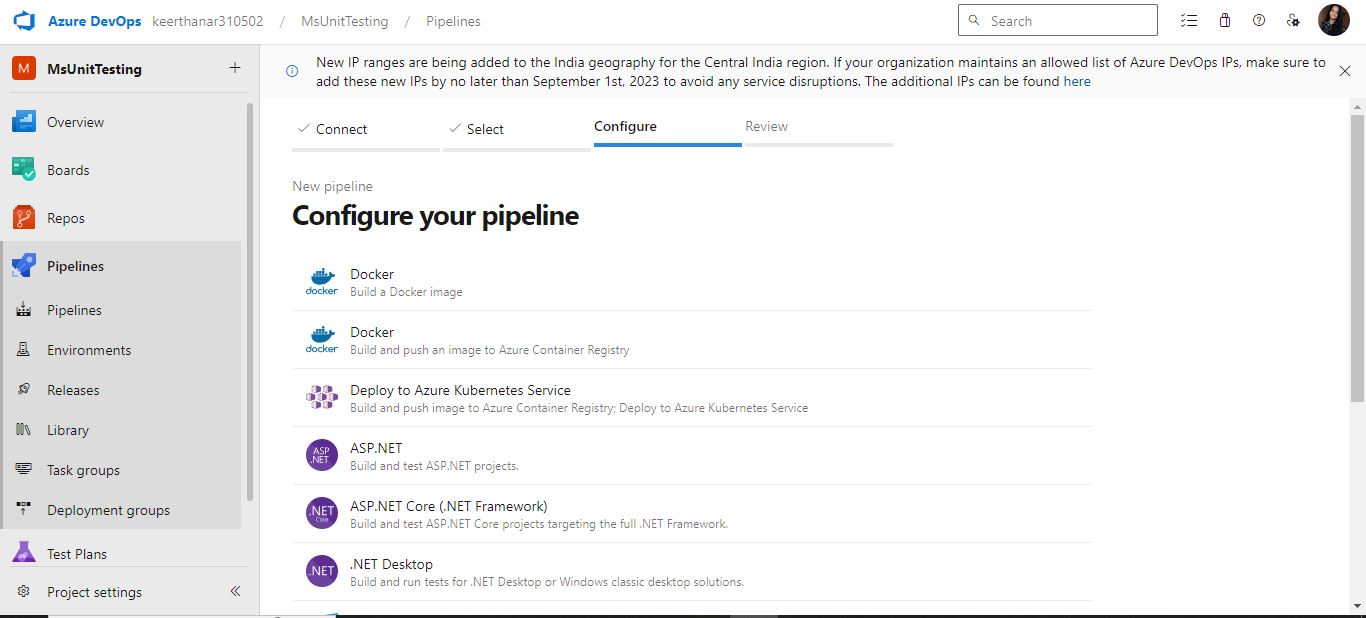
Tasks:

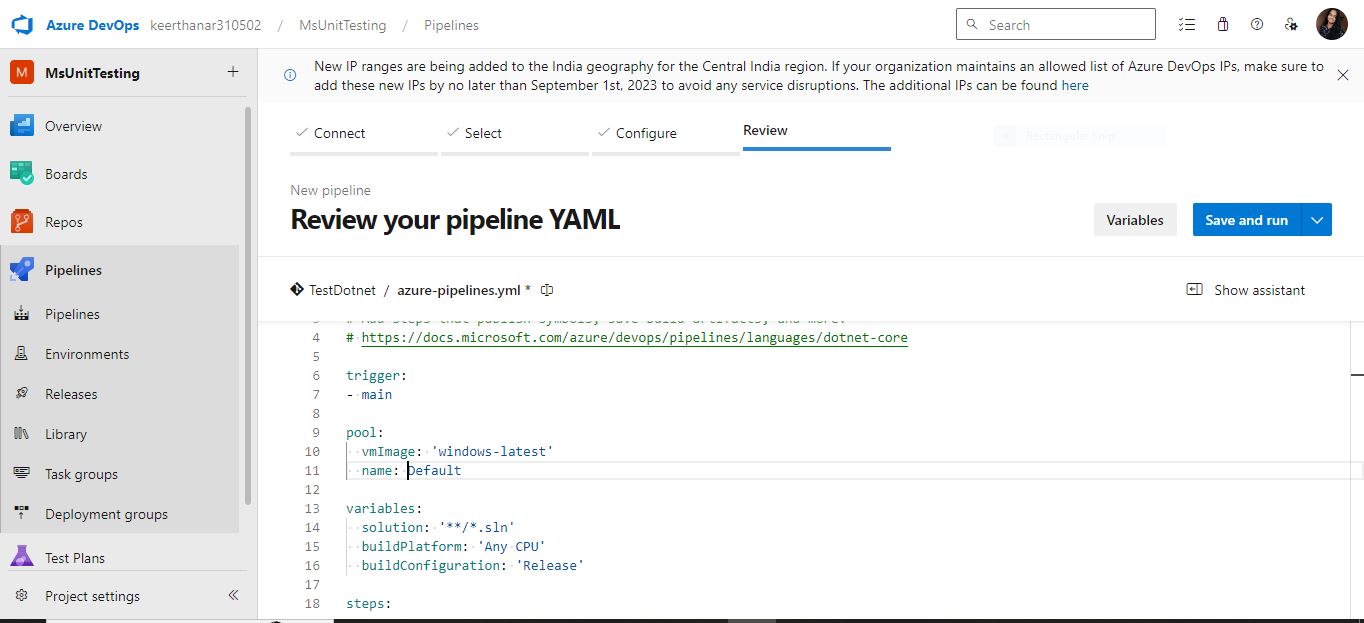
Step 1. Set up a new Azure DevOps project.

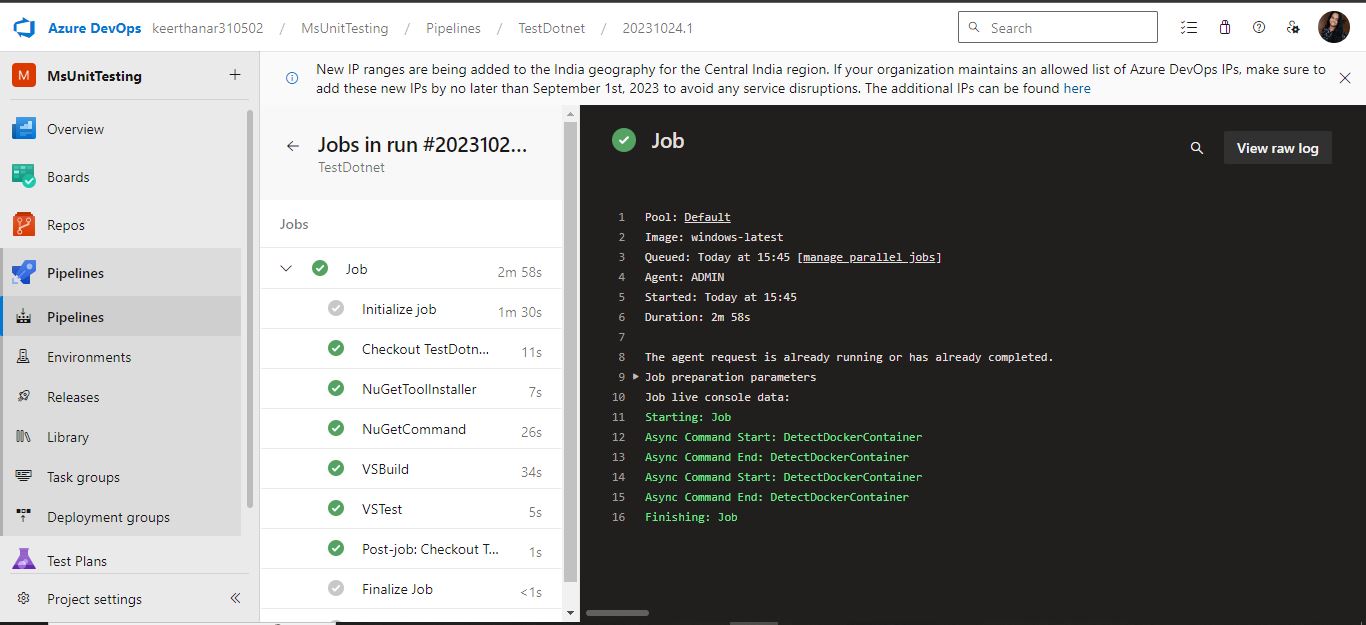
Step 2. Create a CI/CD pipeline for a .NET Core application.

Step 3. Configure the pipeline to use MS Unit tests.

Step 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**

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

