LAB ACTIVITY 2

- Shaurya Srinet (RA2111032010006)

MULTIPLE HOST WITH SAME PEM KEYS

Procedure:

For task 1, where you have two EC2 instances using the same PEM key, and Ansible installed on both, you'll primarily use one of the instances as the Ansible control node to manage the other. Assuming you've decided which instance will be the control node (let's say EC2-1 with IP 44.211.78.252 is the control node, and EC2-2 with IP 44.206.249.140 is the node to be managed), here are the next steps:

Setting up the Ansible Control Node (EC2-1):

1. Transfer Your PEM Key to the Control Node: First, you need to securely copy the PEM key from your local machine to the control node. You can use SCP (Secure Copy Protocol) for this. From your local machine, run:

scp -i "C:\Users\Shaurya\Downloads\lab-assignment.pem" C:\Users\Shaurya\Downloads\lab-assignment.pem" ubuntu@44.211.78.252:/home/ubuntu/lab-assignment.pem

This command copies the lab-assignment.pem key from your local machine to the home directory of the ubuntu user on EC2-1.

2. SSH into Your Control Node: Now, SSH into your control node (EC2-1) using the PEM key.

ssh -i "C:\Users\Shaurya\Downloads\lab-assignment.pem" ubuntu@44.211.78.252

3. Set the Correct Permissions for Your PEM Key: Once logged in, ensure the PEM key has the correct permissions set.

chmod 400 /home/ubuntu/lab-assignment.pem

4. Create the Ansible Inventory File: Create an inventory file on your control node. You can nano inventory.ini

Add the following content, which includes both EC2 instances under the same group [ec2_instances]:

```
[ec2_instances]

ec2_1 ansible_host=44.211.78.252 ansible_user=ubuntu
ansible_ssh_private_key_file=/home/ubuntu/lab-assignment.pem

ec2_2 ansible_host=44.206.249.140 ansible_user=ubuntu
ansible_ssh_private_key_file=/home/ubuntu/lab-assignment.pem
```

Save and close the file (CTRL+O, Enter, and CTRL+X for nano).

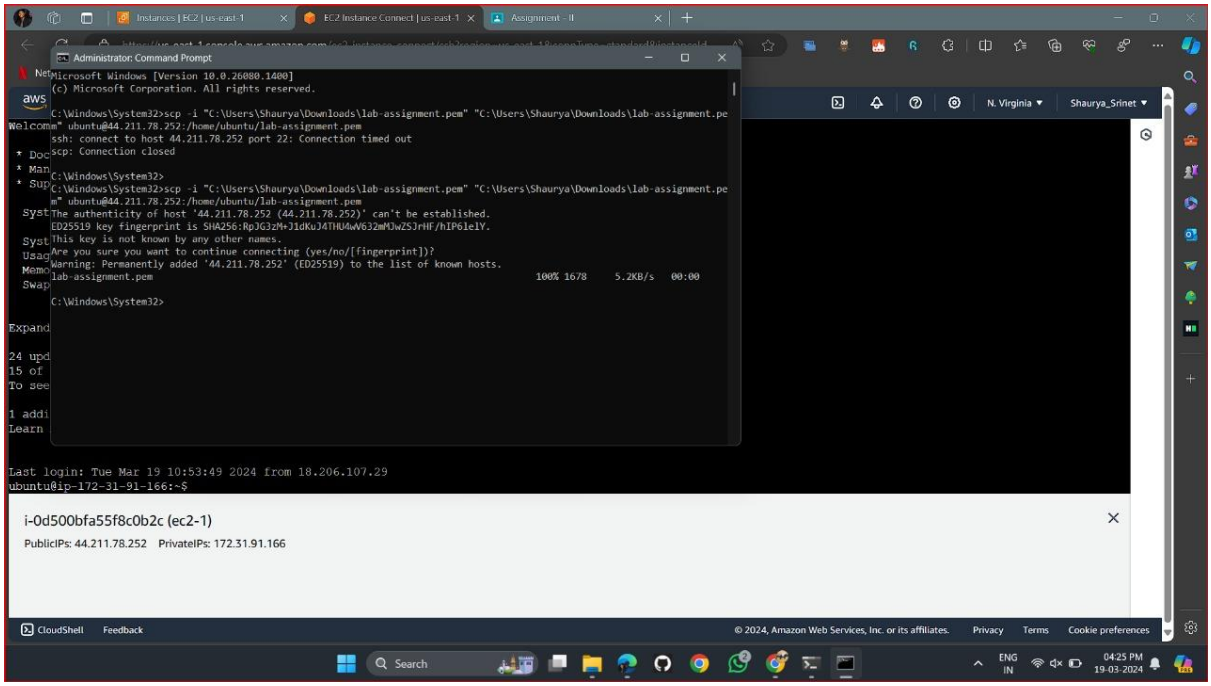
5. Test Ansible Connectivity: Ensure Ansible can connect to both instances by running the ping module. This step verifies SSH access.

ansible -i inventory.ini ec2_instances -m ping

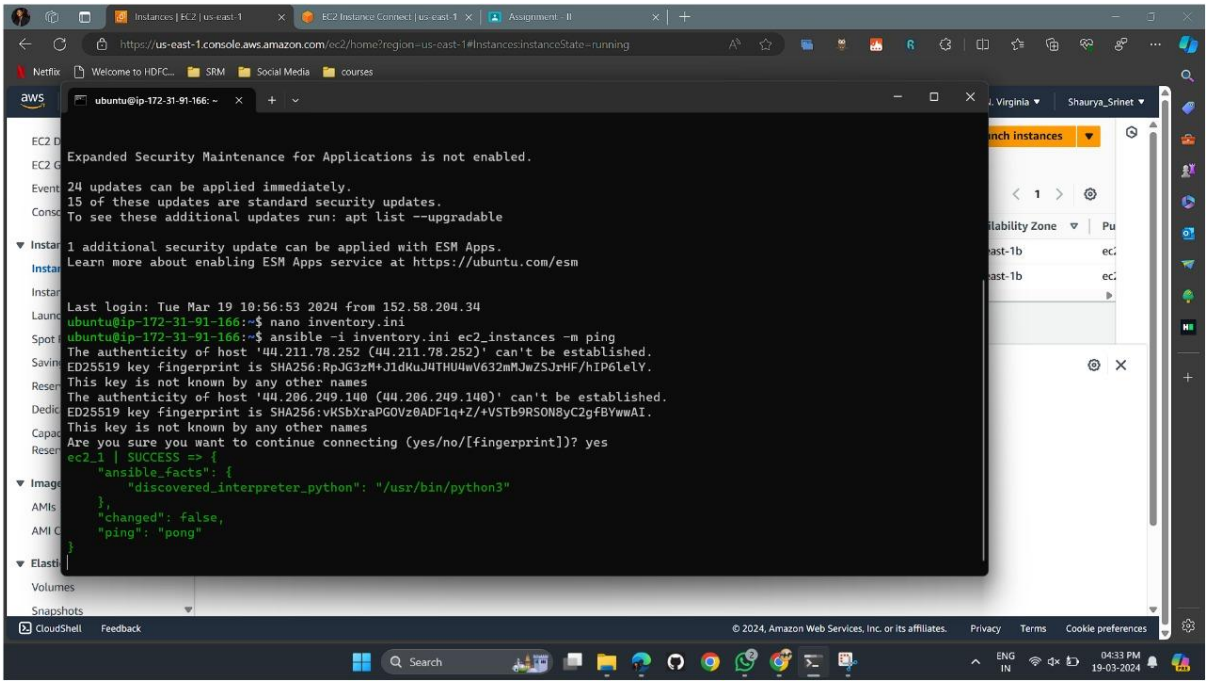
If the above command returns a successful ping response from both instances, your Ansible control node is correctly set up and can communicate with the other instance using the same PEM key.

Output Screenshots:

1. Transfer Your PEM Key to the Control Node:



2. Test Ansible Connectivity



MULTIPLE HOST WITH DIFFERENT PEM KEYS

Procedure:

For Task 2, where you need to manage multiple hosts with different PEM keys, the setup is slightly different. Since you've indicated both EC2 instances are using the same lab-assignment.pem key, for the sake of this task, let's assume you've added a second key to EC2-3 (let's call it lab-assignment2.pem) to demonstrate how you would handle different keys.

Since you've already transferred one key to your control node (EC2-1) and set up Ansible there, you'll need to transfer the second key (assuming it exists for demonstration purposes) and adjust your inventory to reflect the different keys for each instance.

Setting Up for Different PEM Keys

1. Transfer the Second PEM Key to the Control Node: Like the first key, you need to securely copy the second PEM key to your control node (EC2-1). From your local machine, execute:

scp -i "C:\Users\Shaurya\Downloads\lab-assignment.pem" “C:\Users\Shaurya\Downloads\lab-assignment2.pem" ubuntu@44.211.78.252:/home/ubuntu/lab-assignment2.pem

This assumes you have a second key named lab-assignment2.pem for EC2-3.

2. Set Correct Permissions for the Second Key: After logging back into EC2-1, set the correct permissions for the new key.

chmod 400 /home/ubuntu/lab-assignment2.pem

3. Adjust the Ansible Inventory File: You need to modify your inventory.ini file to specify a different PEM key for EC2-3. Open the inventory file:

nano inventory.ini

Adjust it to something like this:

```
[ec2_instances]

ec2_1 ansible_host=44.211.78.252 ansible_user=ubuntu
ansible_ssh_private_key_file=/home/ubuntu/lab-assignment.pem

[ec2_instances_diff_key]

ec2_3 ansible_host=34.201.107.190 ansible_user=ubuntu
ansible_ssh_private_key_file=/home/ubuntu/lab-assignment2.pem
```

This configuration assumes EC2-1 is still using lab-assignment.pem, and EC2-3 is now using a different key, lab-assignment2.pem. Save and close the file.

4. Test Ansible Connectivity for Both Groups: Finally, verify that Ansible can connect to both instances using their respective keys. Run the ping module separately for each group.

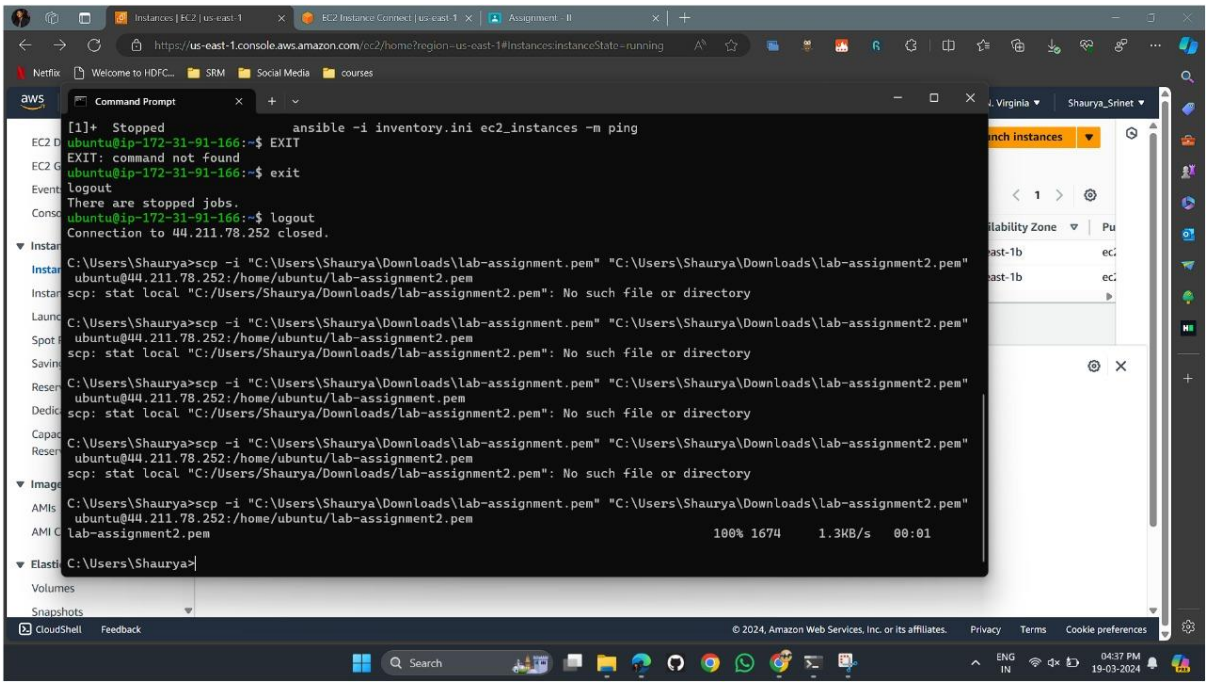
```
ansible -i inventory.ini ec2_instances -m ping

ansible -i inventory.ini ec2_instances_diff_key -m ping
```

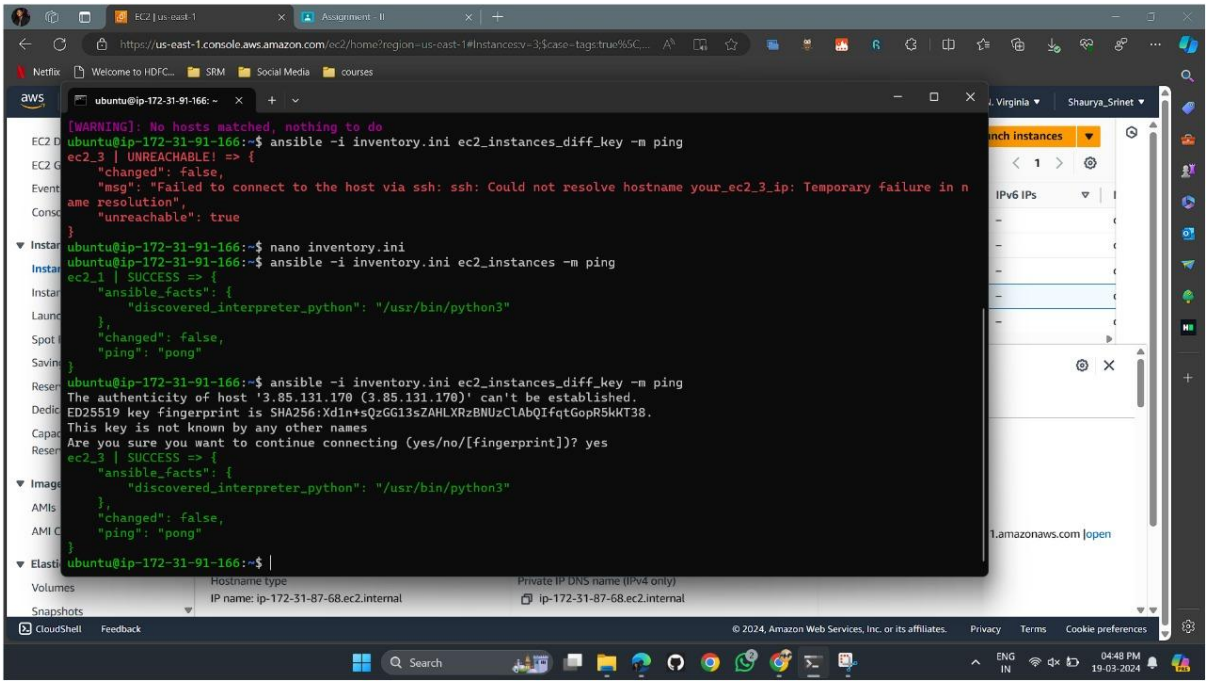
If these commands return successful responses, it means Ansible can communicate with both EC2 instances using different PEM keys, completing Task 2 of your lab assignment. Remember, in a real-world scenario, each instance would genuinely have a different key; this example assumes such a setup for demonstration purposes.

Output Screenshots:

1. Transfer the Second PEM Key to the Control Node:



2. Test Ansible Connectivity for Both Groups:



ENVIRONMENT SETUP TO WORK ON LOCAL HOST AND REMOTE HOST

Procedure:

Given your setup with three EC2 instances where EC2-1 and EC2-2 use `lab-assignment.pem` and EC2-3 uses `lab-assignment2.pem`, you can configure your Ansible to manage all three, with your local machine serving as the Ansible control node.

1: Ansible Installation on Your Local Machine

Ensure Ansible is installed on your local machine. If it's a Linux or MacOS system, you can typically install Ansible with a package manager:

sudo apt update && sudo apt install ansible -y

2: Preparing Your Inventory File

Create or update an Ansible inventory file (e.g., `inventory.ini`) on your local machine. This file should include all three EC2 instances, with EC2-1 and EC2-2 in one group and EC2-3 in another due to the different PEM key.

[ec2_instances_same_key]

ec2_1 ansible_host=184.72.149.171 ansible_user=ubuntu ansible_ssh_private_key_file=C:/Users/Shaurya/Downloads/lab-assignment.pem

ec2_2 ansible_host=34.203.217.122 ansible_user=ubuntu ansible_ssh_private_key_file=C:/Users/Shaurya/Downloads/lab-assignment.pem

[ec2_instance_diff_key]

ec2_3 ansible_host=34.201.107.190 ansible_user=ubuntu ansible_ssh_private_key_file=C:/Users/Shaurya/Downloads/lab-assignment2.pem

Replace `184.72.149.171`, `34.203.217.122`, and `34.201.107.190` with the actual public IP addresses of your instances. The paths to the PEM files should be adjusted according to where they are stored on your local machine. Ensure the PEM files have the correct permissions (read-only for the owner is typically sufficient).

Step 3: Testing Ansible Connectivity

To verify that Ansible can communicate with all your EC2 instances, run the ping module:

ansible -i inventory.ini all -m ping

This command should return a success message for each instance if everything is configured correctly.

4: Running Ansible Commands or Playbooks

With your inventory set up, you can now run ad-hoc commands or playbooks across your EC2 instances. For example, to check the disk space on all instances, you could use:

ansible -i inventory.ini all -a "df -h"

Or to only target the instances with the same PEM key:

ansible -i inventory.ini ec2_instances_same_key -a "df -h"

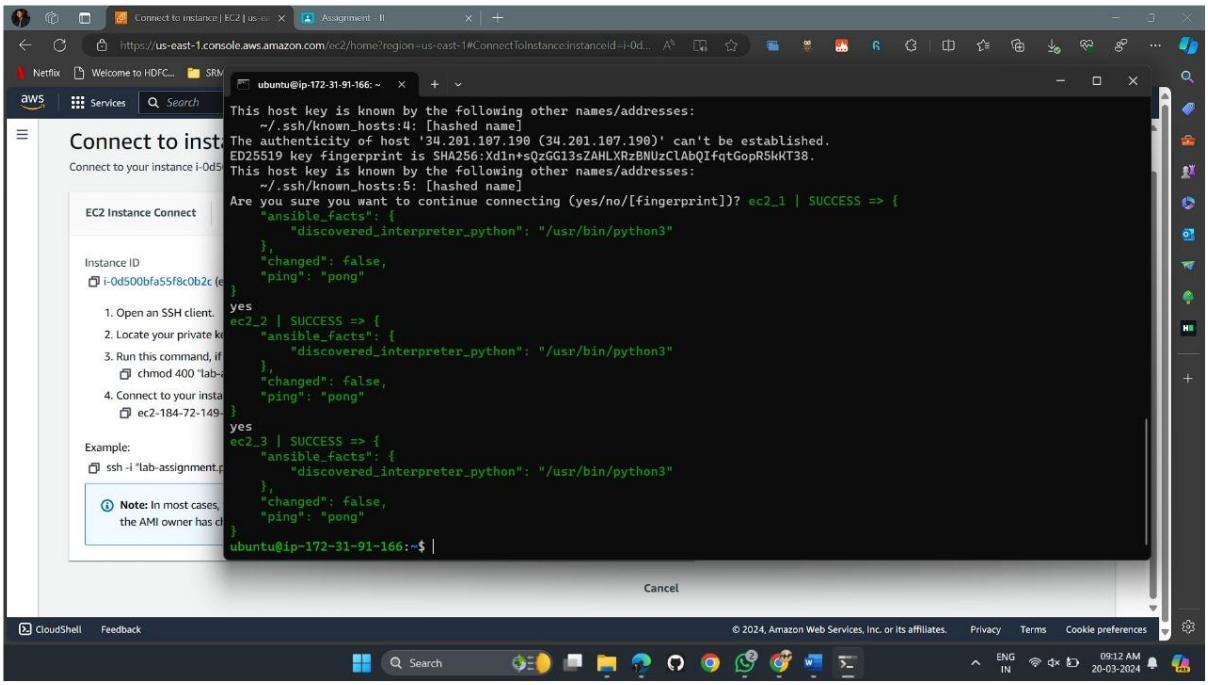
And for the instance with a different PEM key:

ansible -i inventory.ini ec2_instance_diff_key -a "df -h"

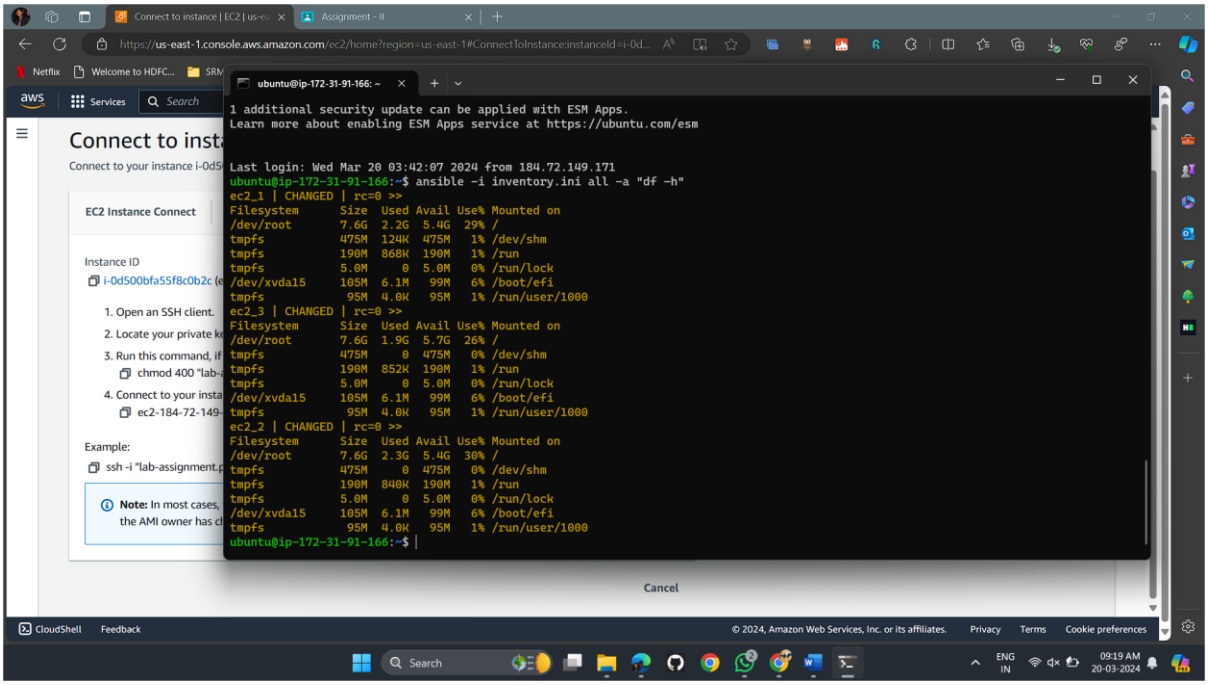
This setup allows you to manage all your instances from your local machine, regardless of which PEM key they use. Ensure that your security groups and network ACLs allow SSH access from your local IP to these instances.

Output Screenshots:

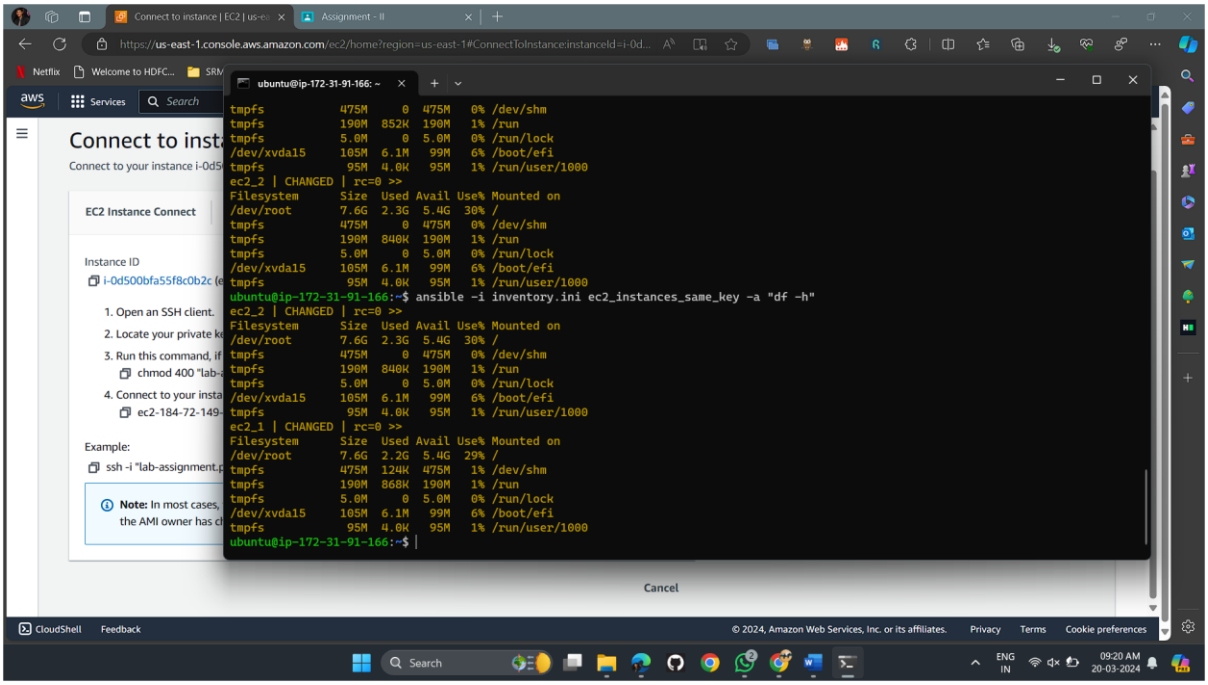
1. Testing Ansible Connectivity:



2. The disk space on all instances:



3. Targeting the instances with the same PEM key:



4. Targeting instance with a different PEM key: