Ansible Corporate Project to Monitor VM and Send Mail

This Ansible project demonstrates automating VM monitoring and email reporting. The tutorial shows how to collect CPU, RAM, and disk usage metrics from multiple virtual machines. An automated email, formatted with HTML, then delivers a consolidated report.

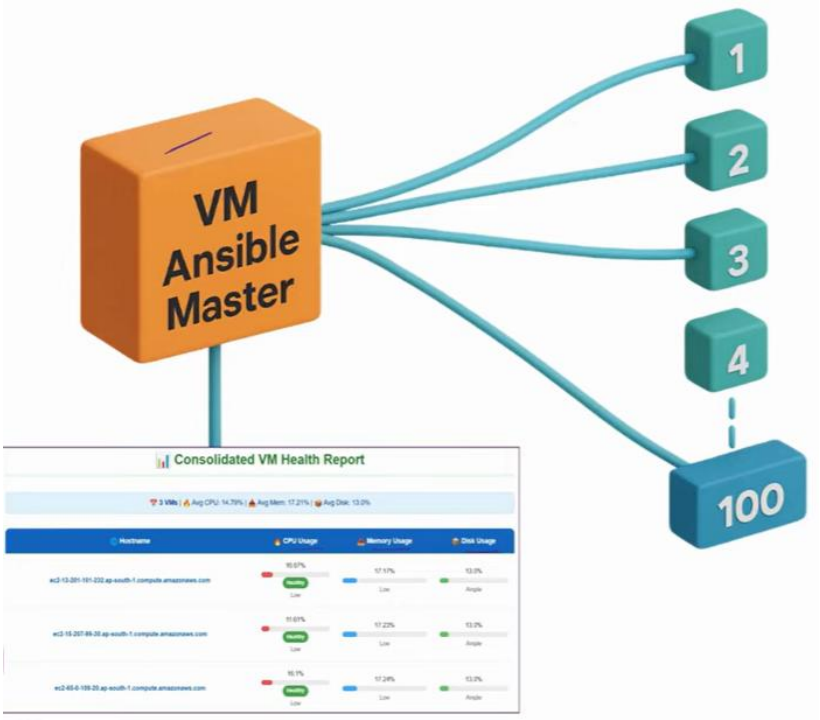
Let us assume a scenario that you are working in a company and you are supposed to manage 500 VMs. So, you are supposed to send the details of node metrics, like how much is CPU usage, how much is RAM usage and what is the space currently available on all those machines over email.

How can you achieve this? This can be achieved by using Ansible. With Ansible, we can just write our script which is going to automatically fetch the details of 500 VMs such as their node metrics like CPU usage, RAM usage and space available, consolidate everything inside an email and send that email to the manager or whoever is required.

All these things can be implemented very easily using Ansible. That is what we are going to do in this tutorial.

Project Description

Let us understand the project that we need to implement.

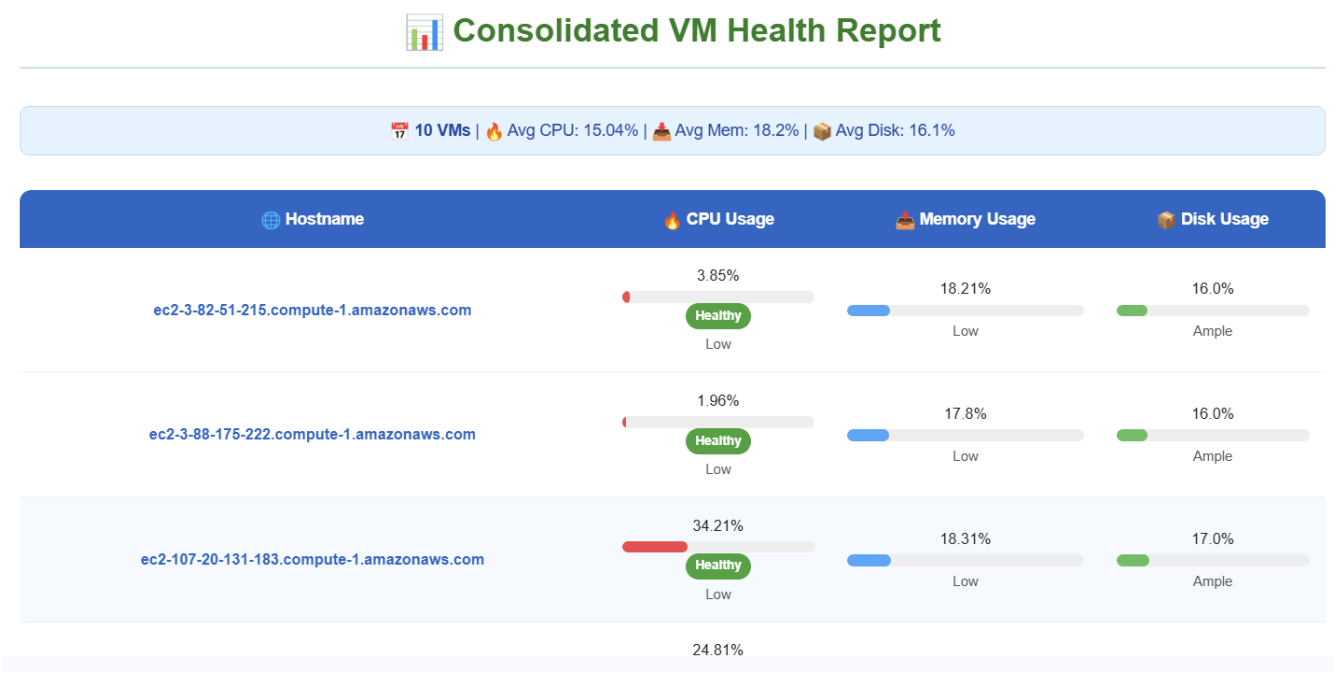


First, let us understand the use case. I have a use case that I have a total of 10 VMs. And I have been asked to share the information about node matric on the VM at any specific time.

Let us say currently time is 3:00pm, and my manager says at 3:00pm, I need to know information like 3:00pm what was the CPU usage, what was the RAM usage and how much disk or how much storage was available on the all those 10 VMs.

So, on this we need to have node metrics of CPU usage, RAM usage and disk availability. Once we have that information, my manager want that I should be sending that information to my manager in a proper format over email.

So, you can understand if I try to do this manually. It is going to be very hectic and not at all feasible. So, we need to use Ansible to set up an automation which is going to send this information collect this information on all those 10 VMs and send that information over email. So, the email format will be something like this



You can see DNS names of our machines, CPU usage, Memory Usage and Disk usage.

IMPLEMENTATION

We will now go ahead and implement this project

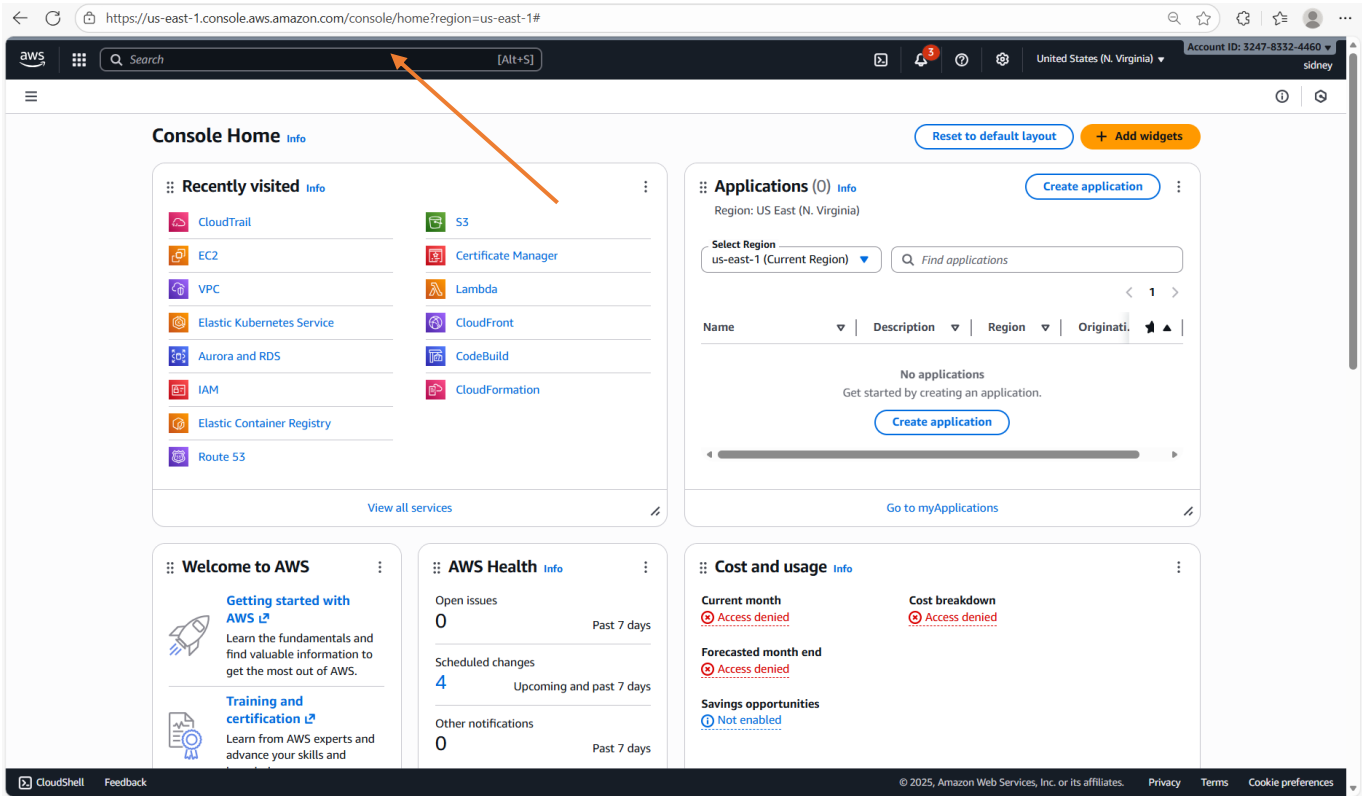
STEP 1: Create EC2 instances

We will create all the Ubuntu EC2 instances needed for this project,

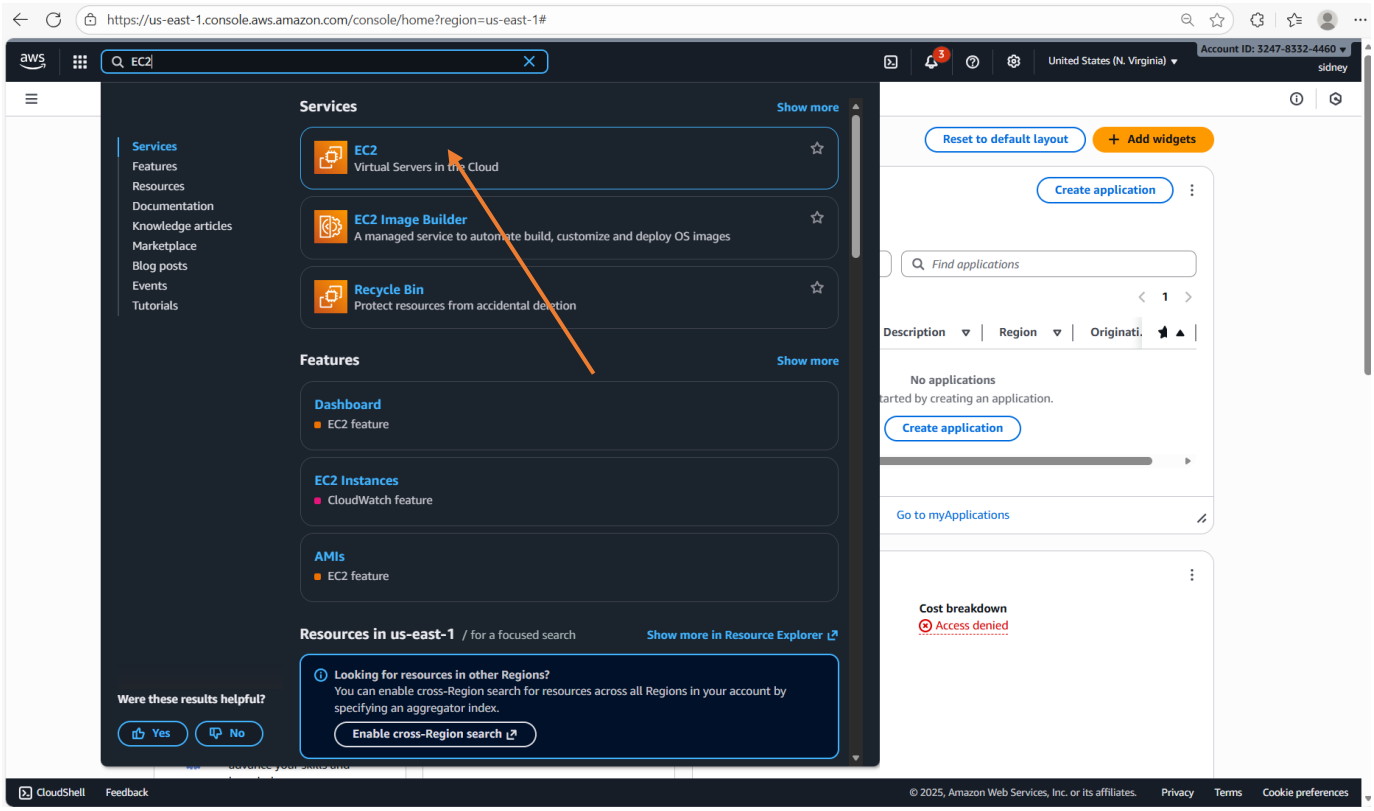
Part 1: Create EC2 instance for Ansible Master

We will first create an Ubuntu EC2 instance to server as our master server. We will call it "**Ansible-Master**", where we will execute all the commands. We will then SSH connect to it.

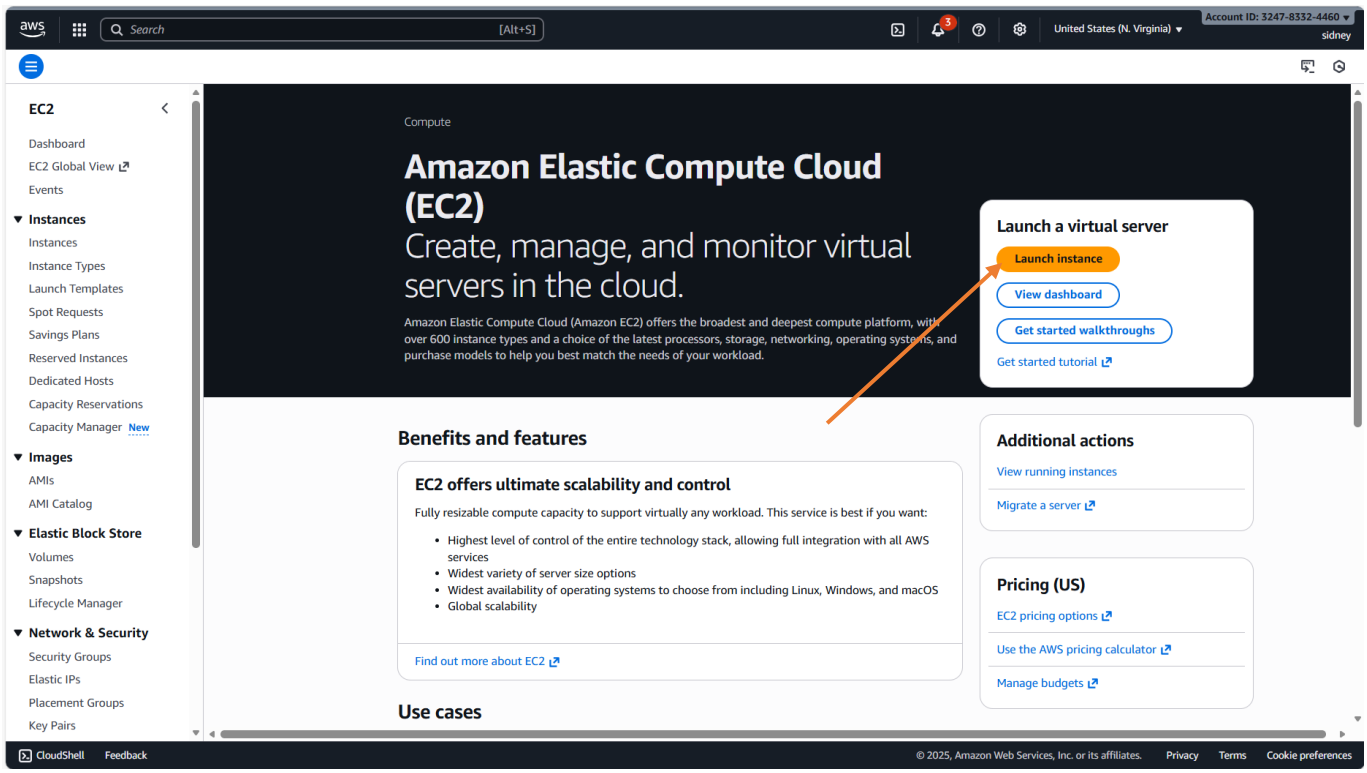
Go to AWS Management Console and login



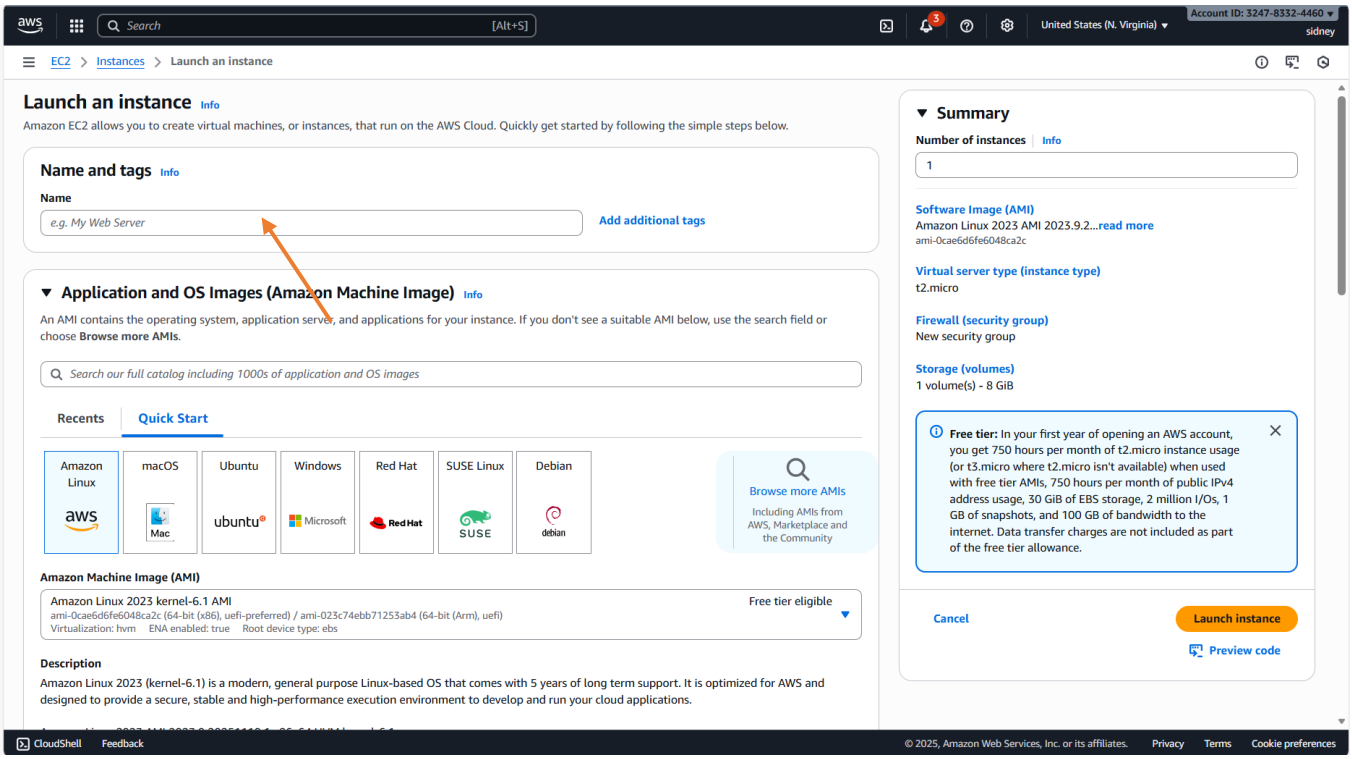
Search for "**EC2**"



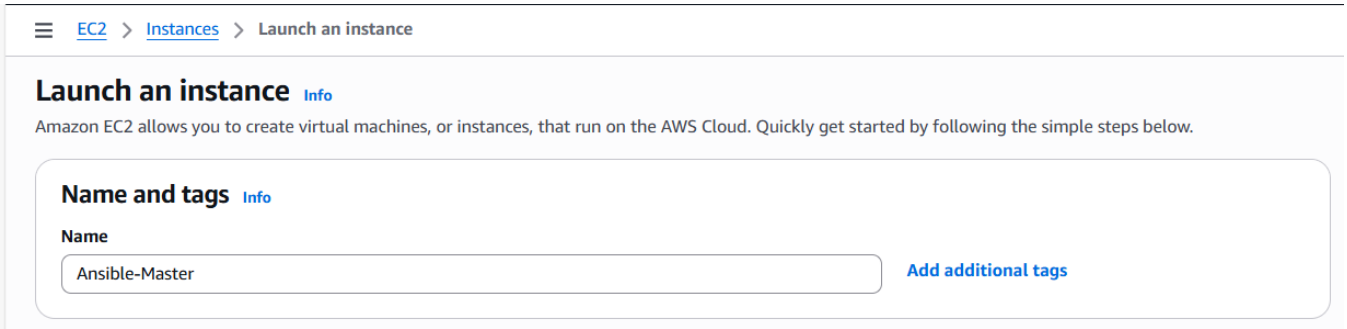
Click on "**EC2**"



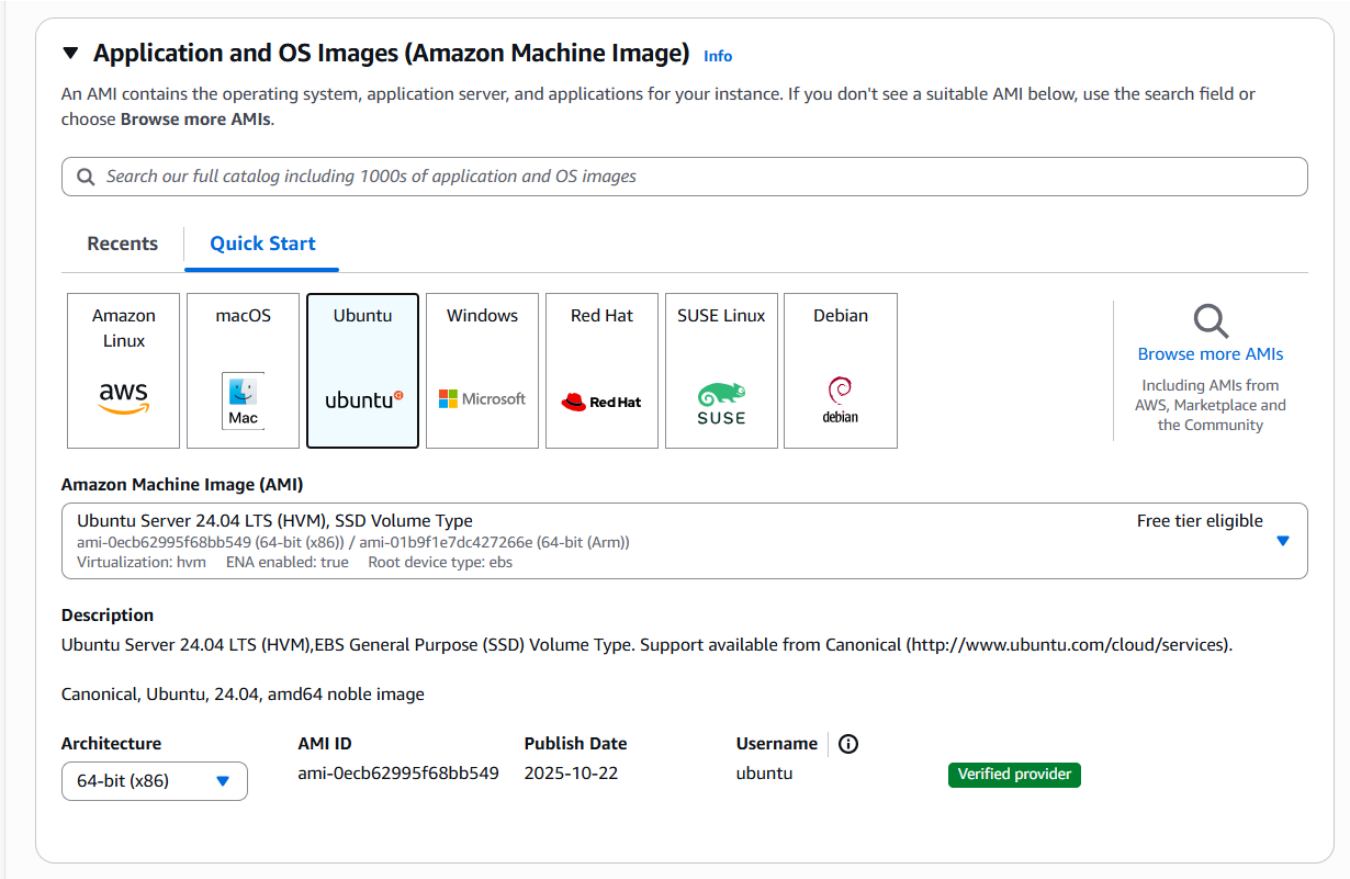
Click on "**Launch Instance**"



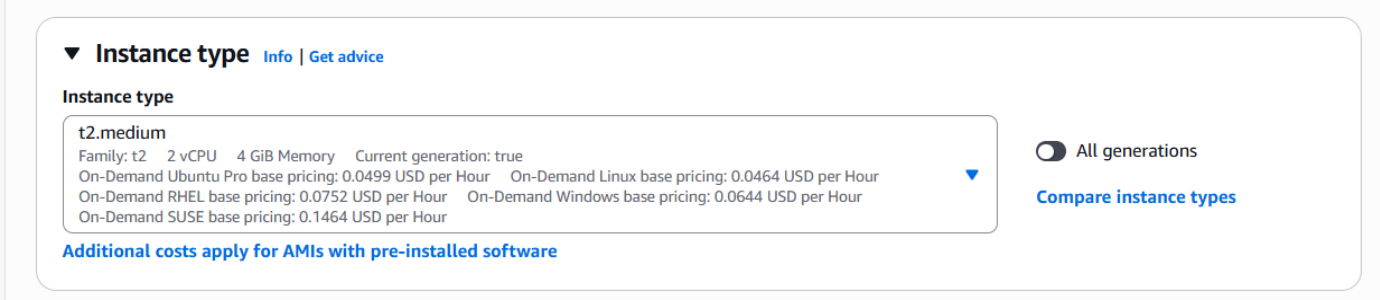
We will give the instance the name "**Ansible-Master**"



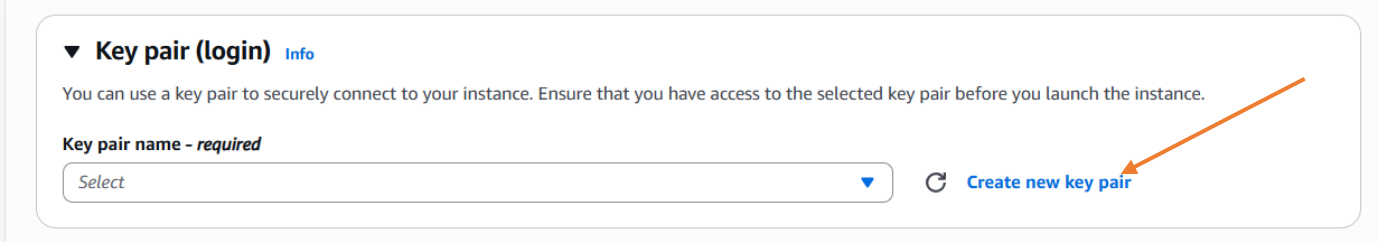
On "**AMI**", select "**Ubuntu**"



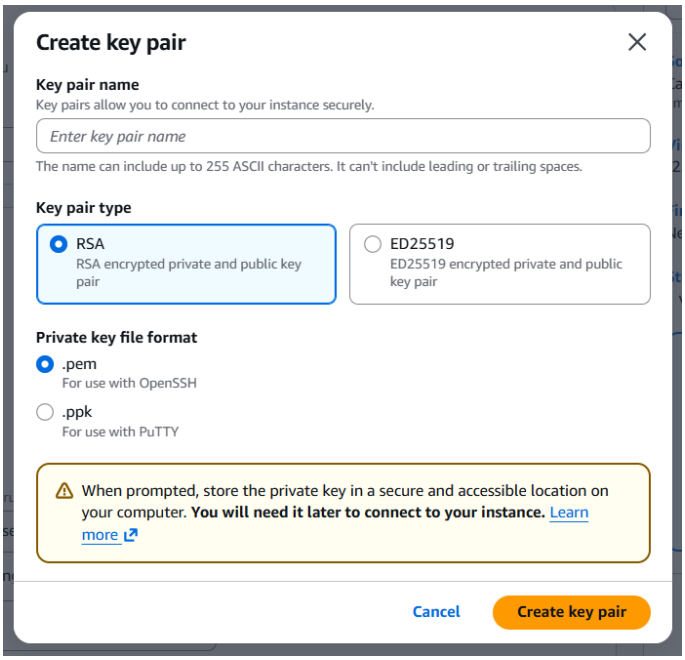
Scroll down to "**Instance Type**" and select "**t2.medium**"



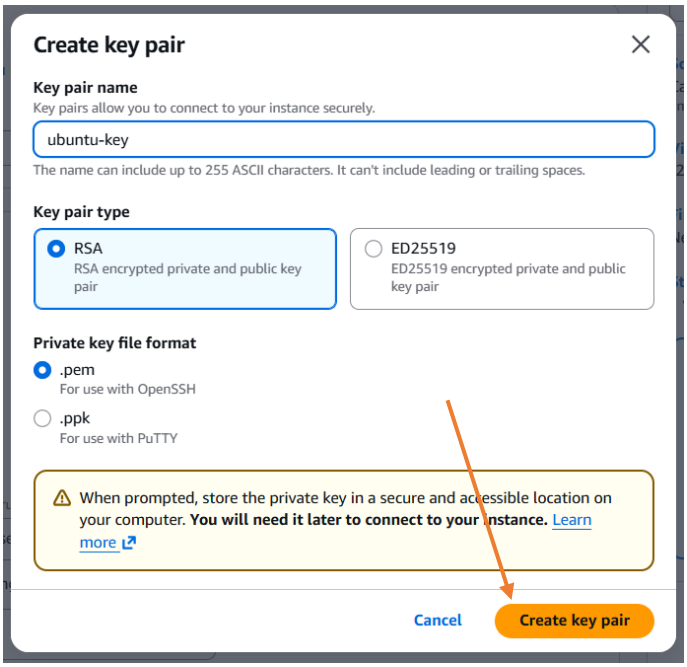
Scroll down to "**Key Pair**"



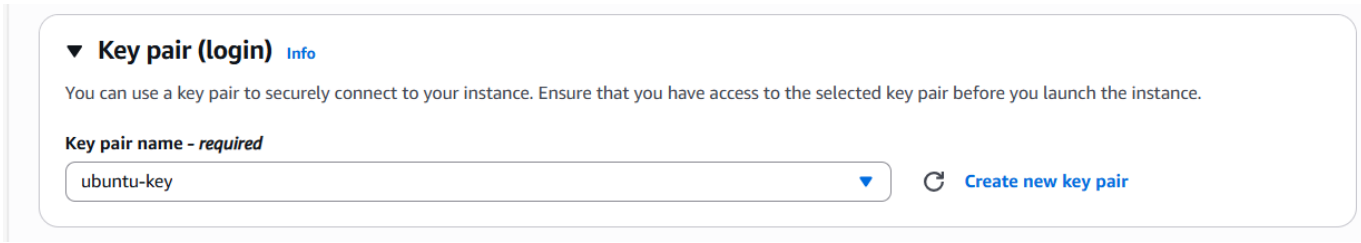
Click on "**Create new key pair**"



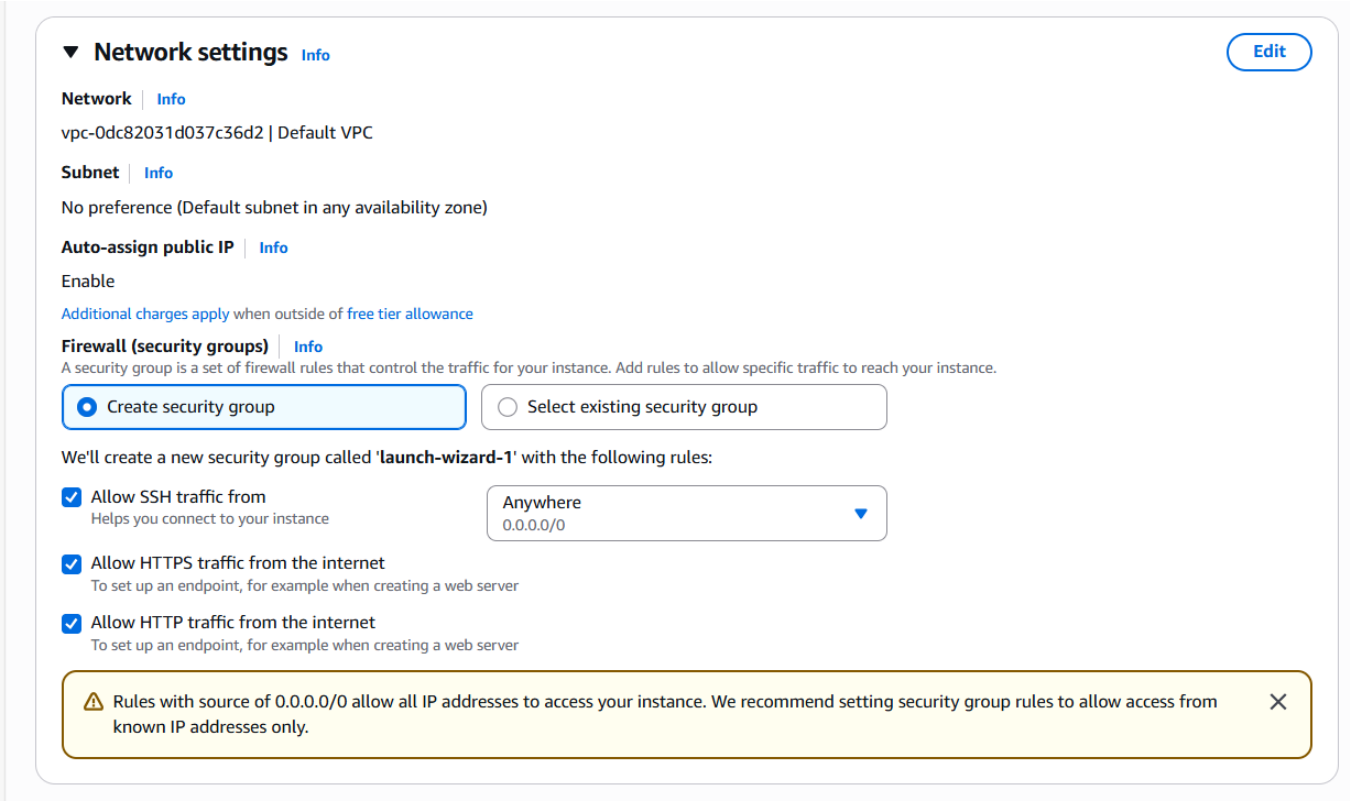
Let us give the key a name. We will call it "**ubuntu-key**"



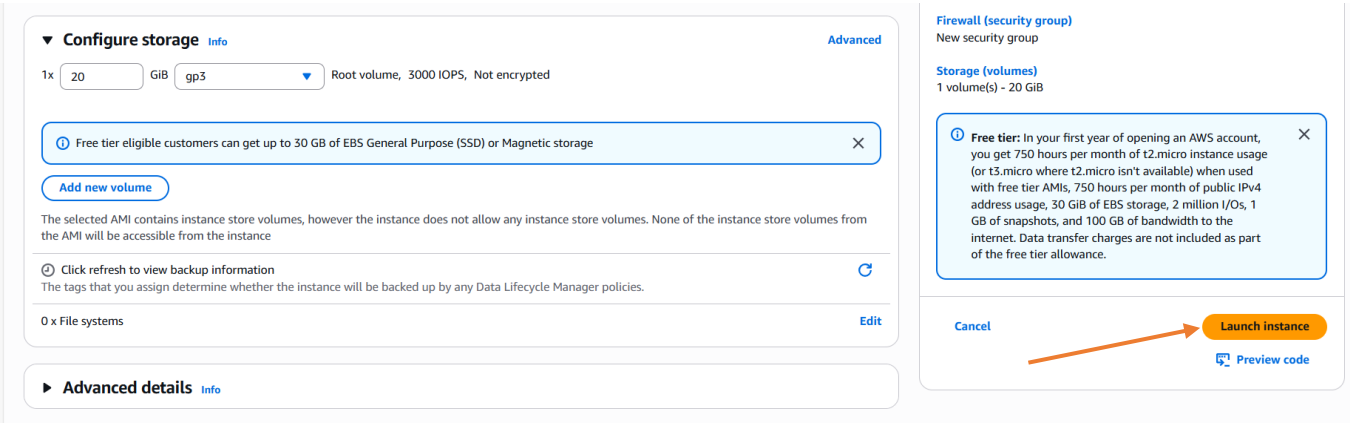
Click on "**Create key pair**"



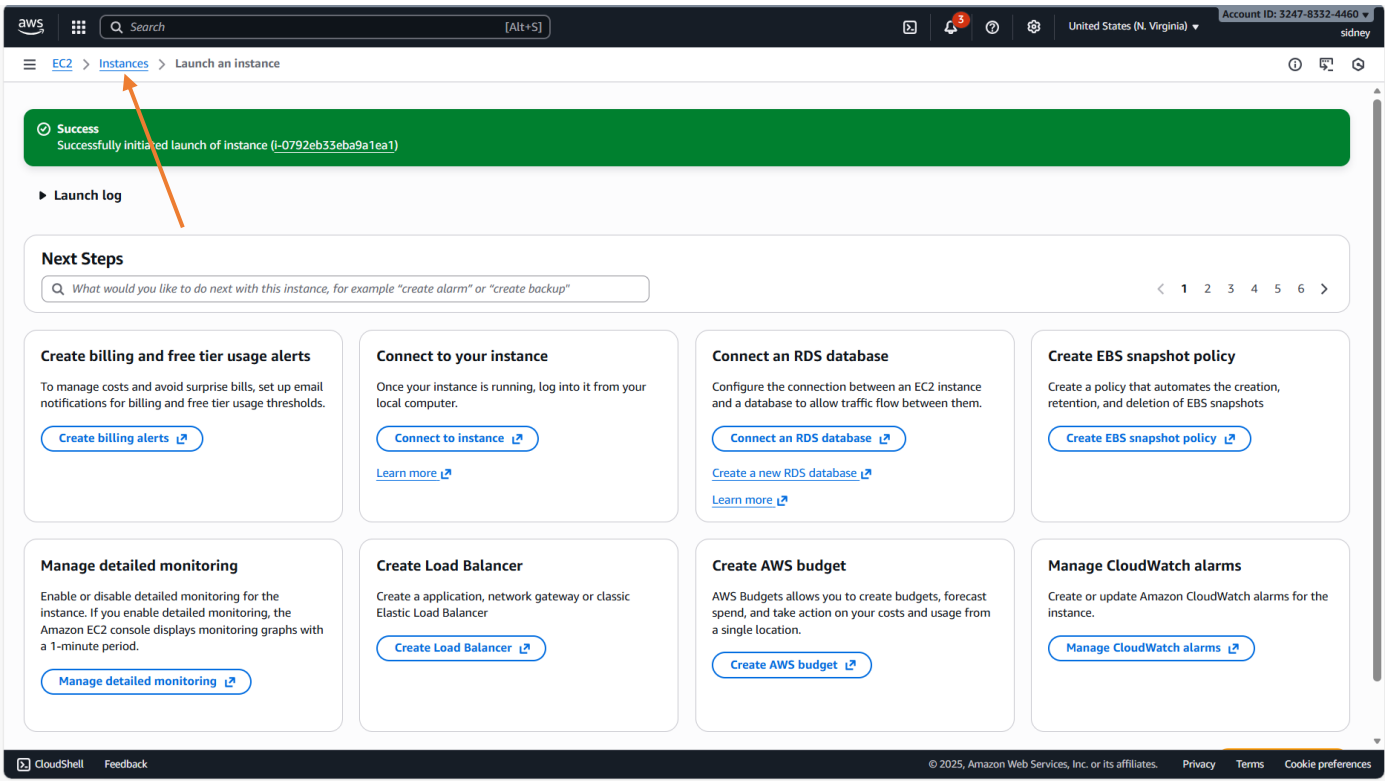
Scroll down to "**Network Settings**". Select "**create security group**". Then select "**Allow SSH traffic from**", "**Allow HTTPS traffic from the internet**" and "**Allow HTTP traffic from the internet**"



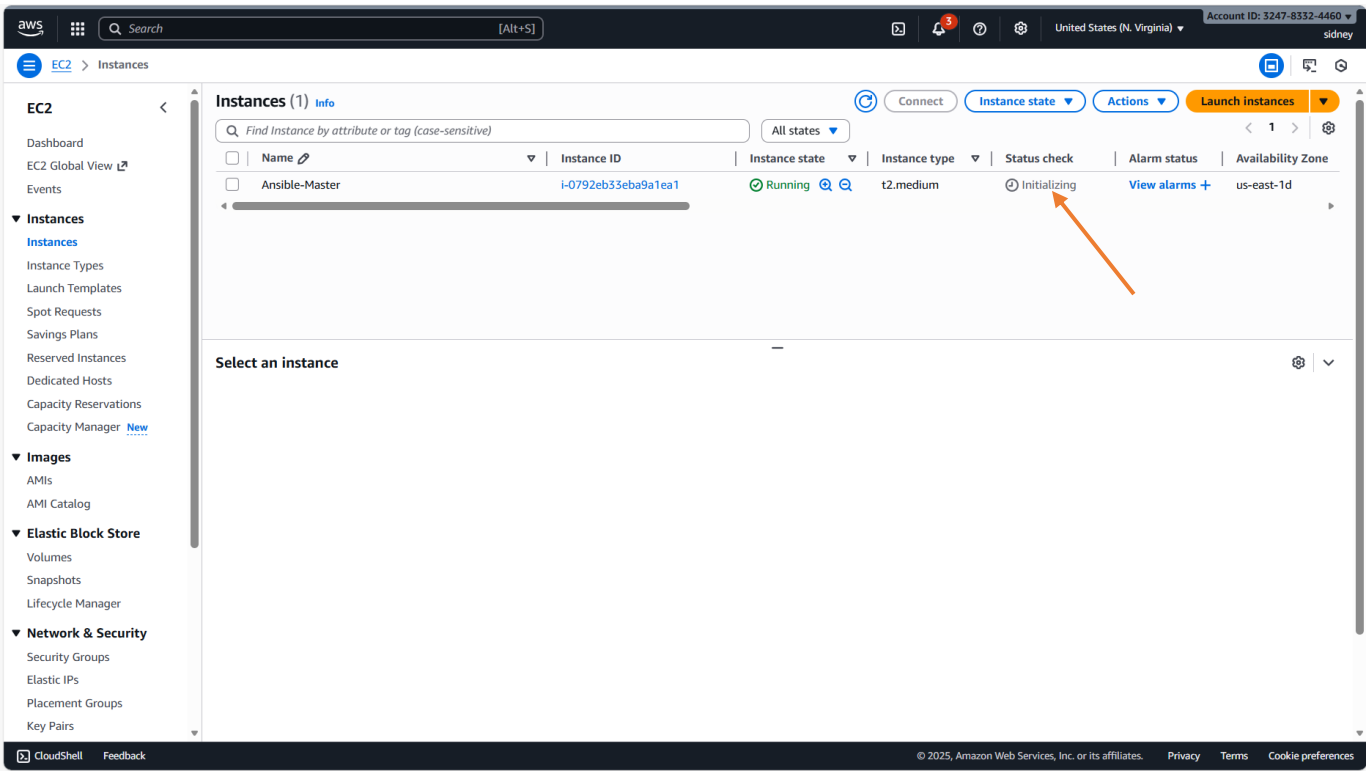
Then scroll down to "**Configure Storage**" and make the value "**20GiB**"



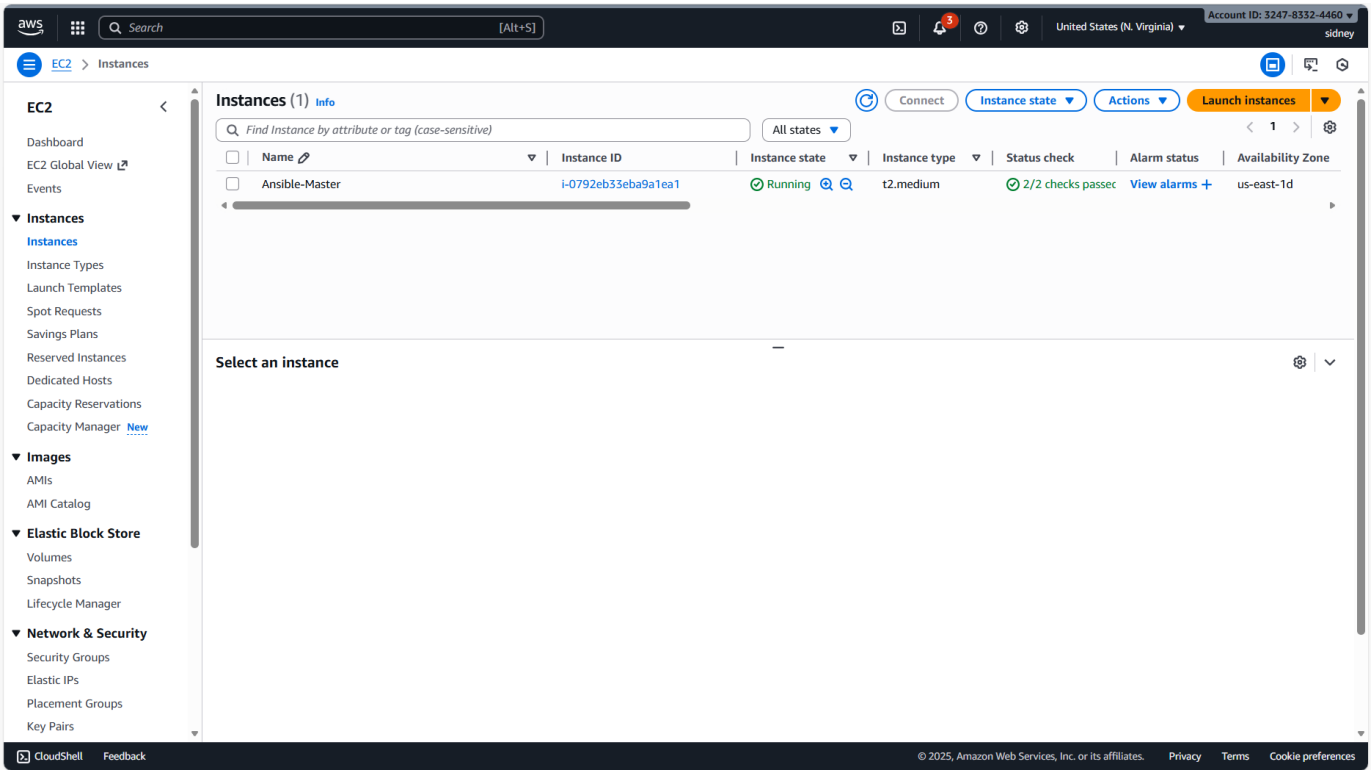
Then click on "**Launch Instance**"



Click on "**Instances**"



You can see that the instance has been created and it is initializing. Let us wait for it to pass the "**2/2 check**"

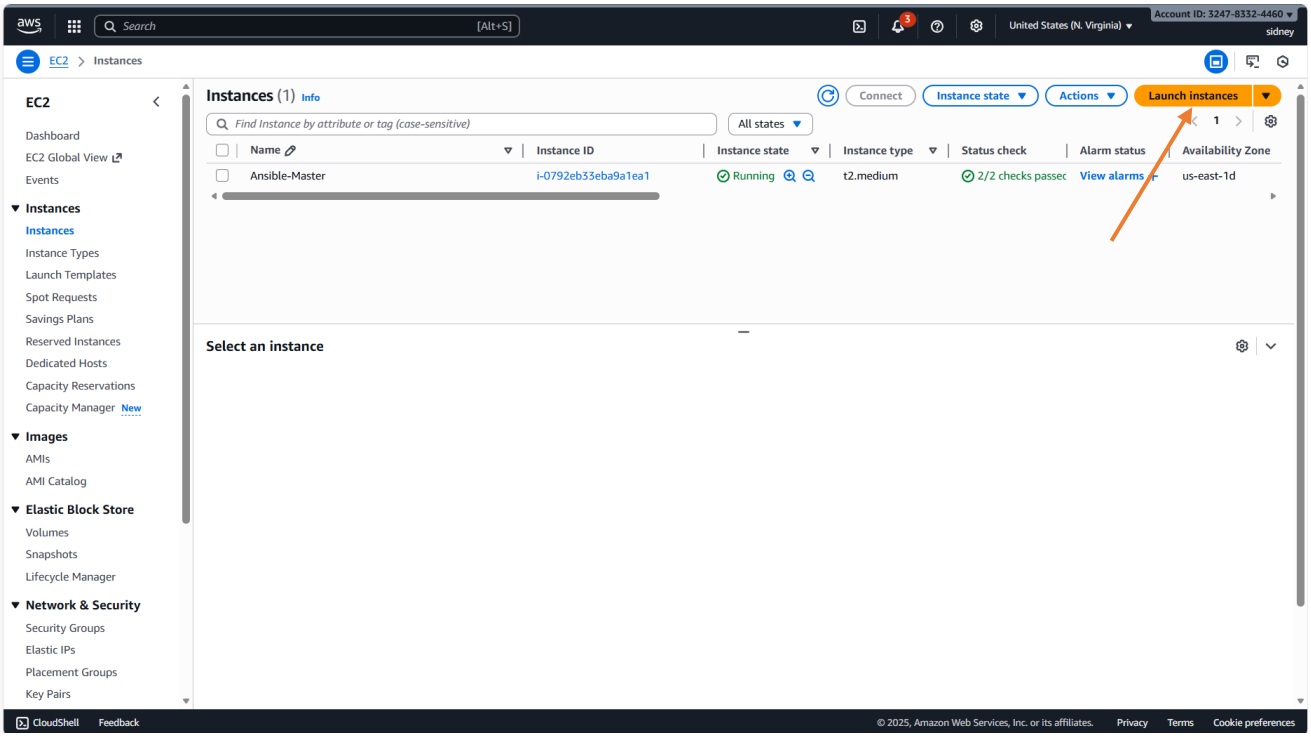


The EC2 instance has passed the "**2/2 check**"

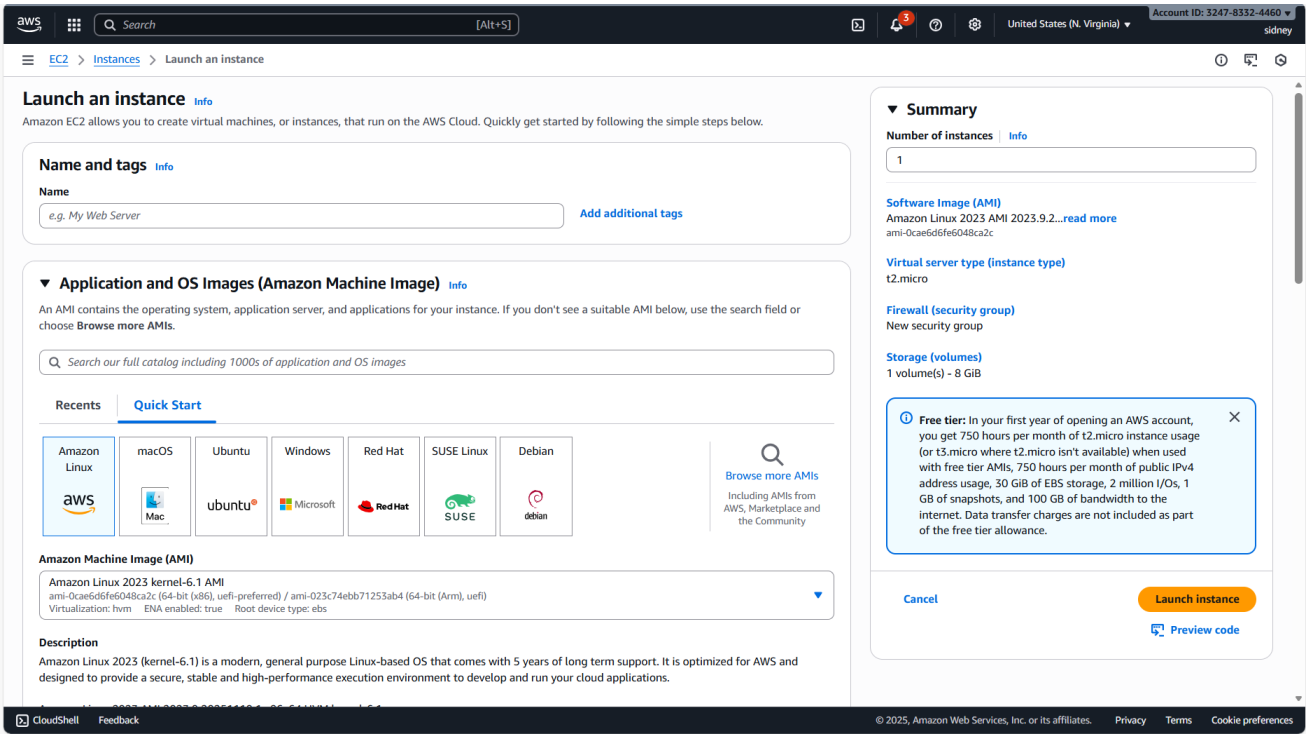
Part 2: Create Ten Target EC2 instances

We have to create the ten target servers. These are the servers whose metrics we want to get. We are going to be using dynamic inventory file. The file will automatically fetch all the EC2 machines and get their IP addresses. To achieve this, we are going to use "**tags**" on the EC2 instances.

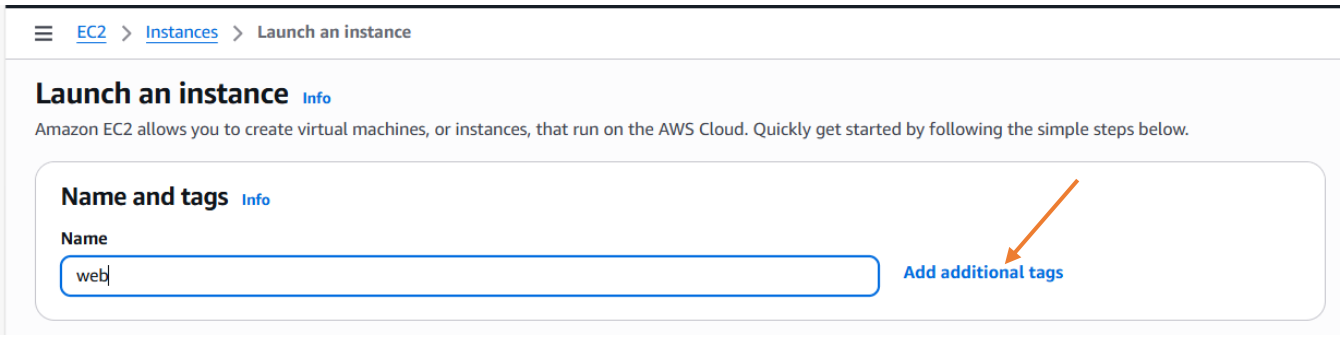
Part 1: Create EC2 instances.



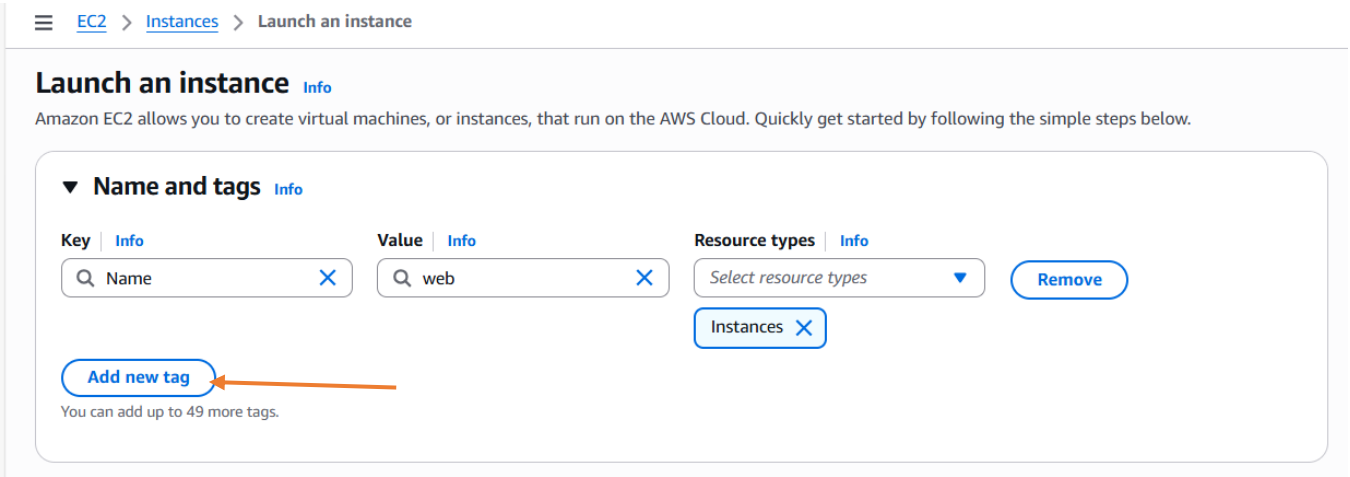
Click on "**Launch Instance**"



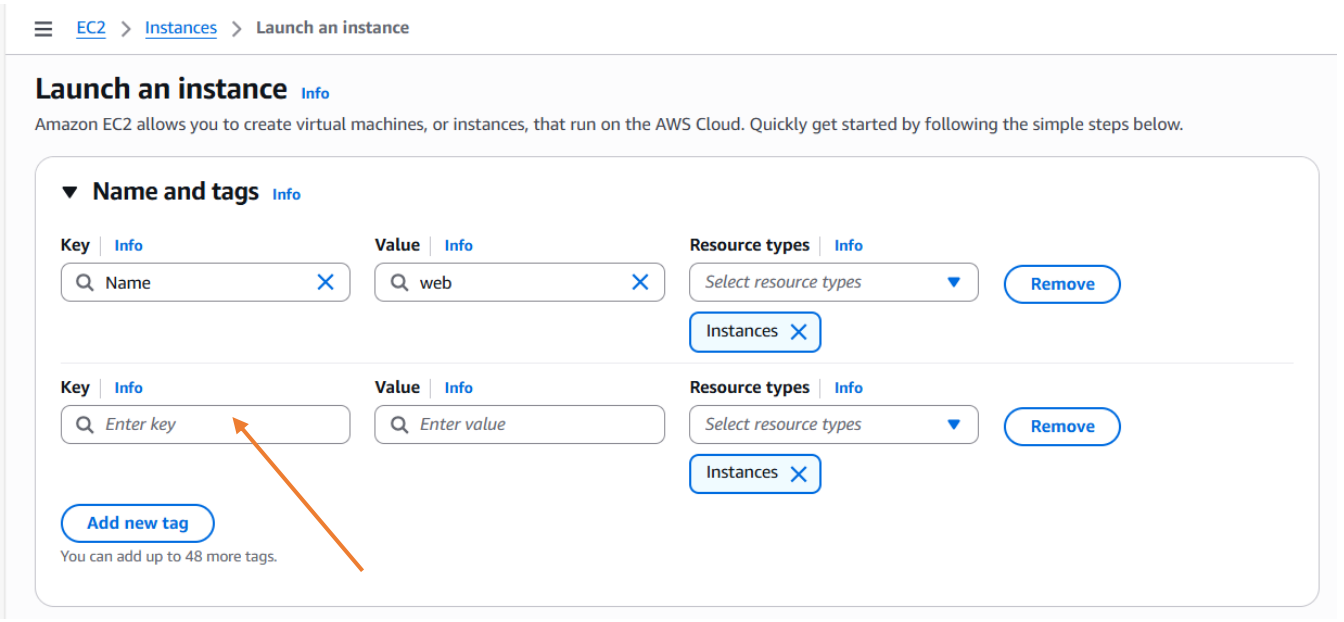
Give the instance a name, I will call it "**web**"



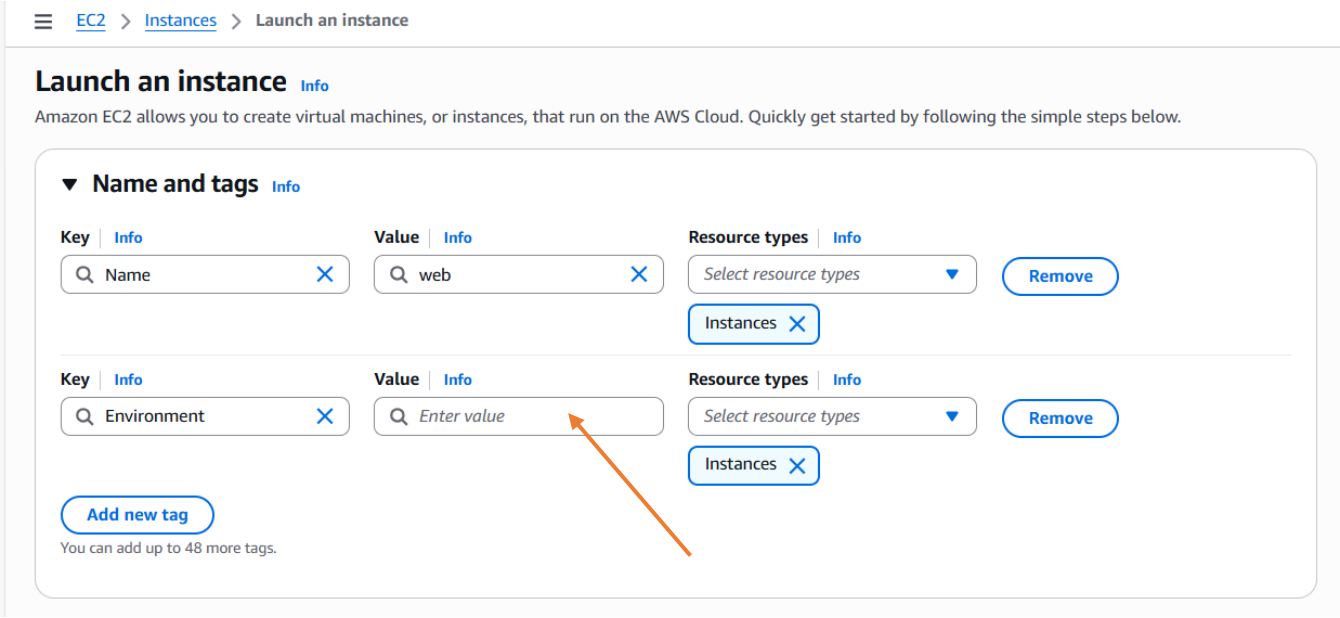
We have to add tags, click on "**Add additional tags**"



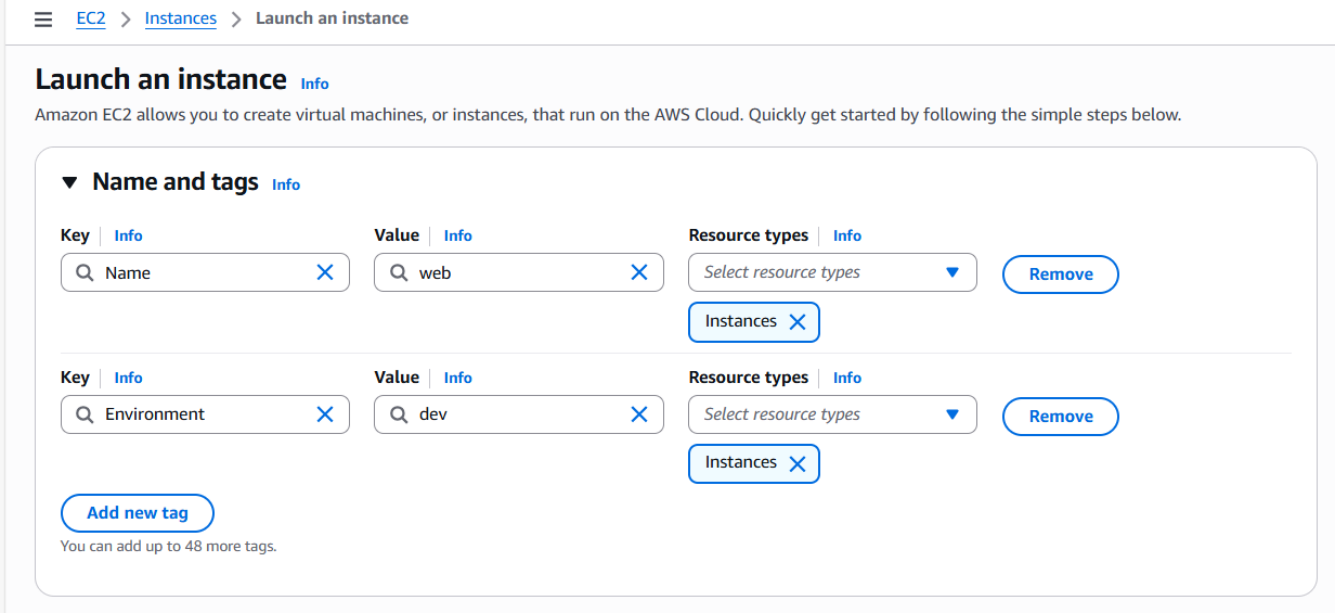
Click on "**Add new tag**"



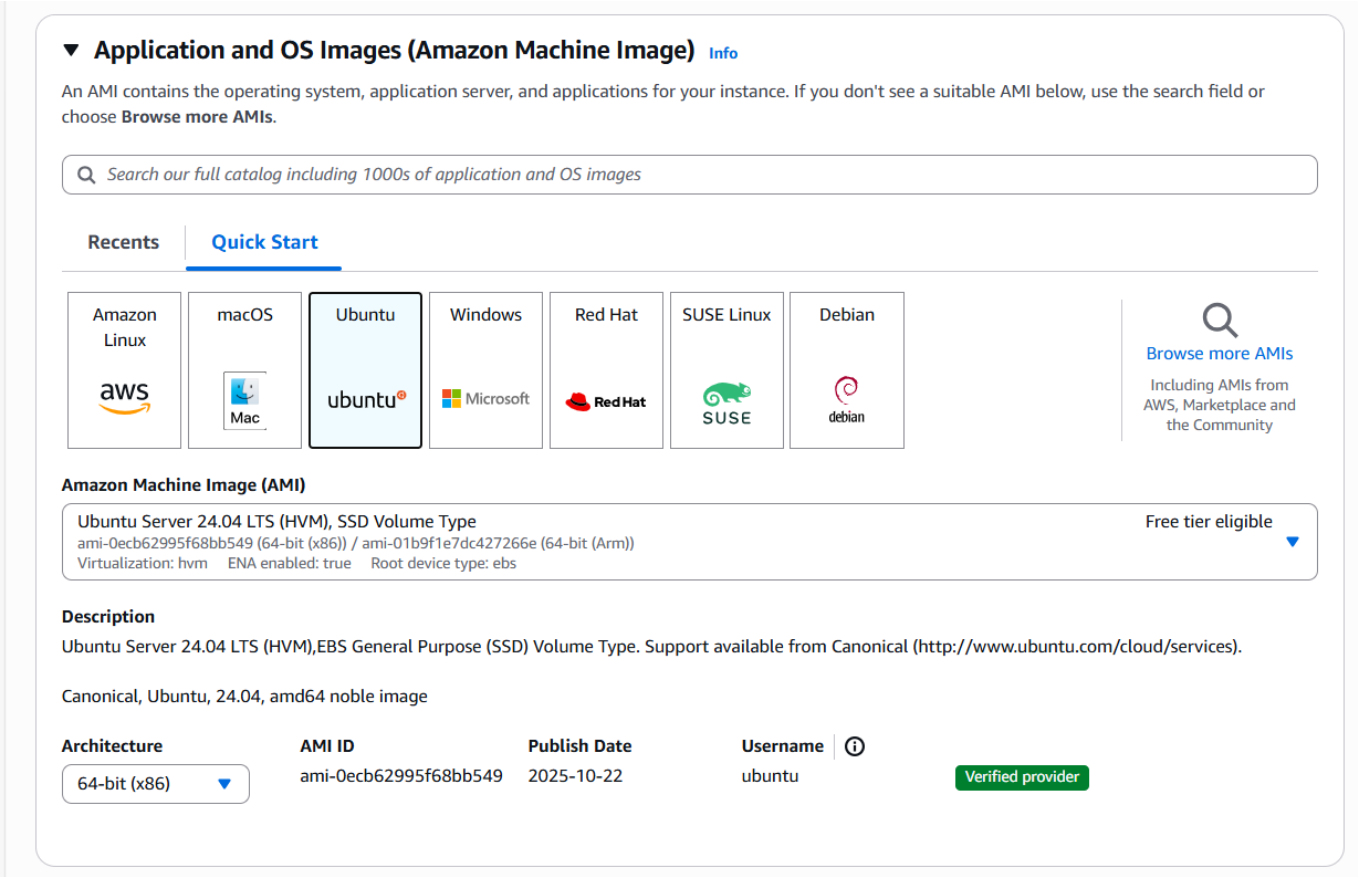
In the "**Key**", enter "**Environment**"



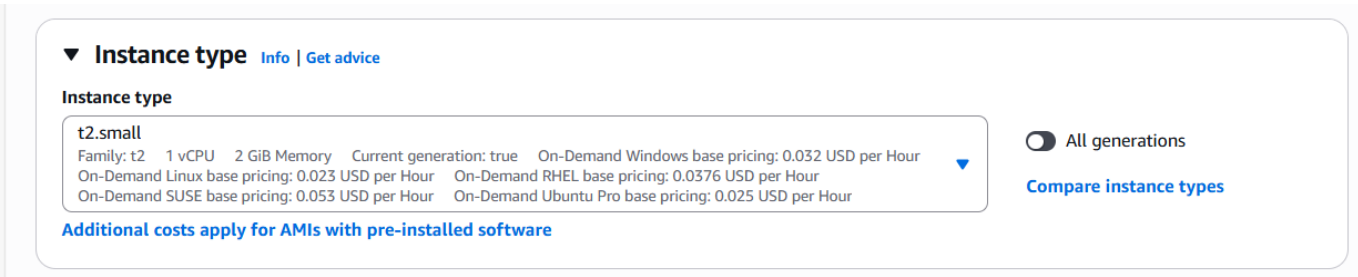
For "**Value**", enter "**dev**"



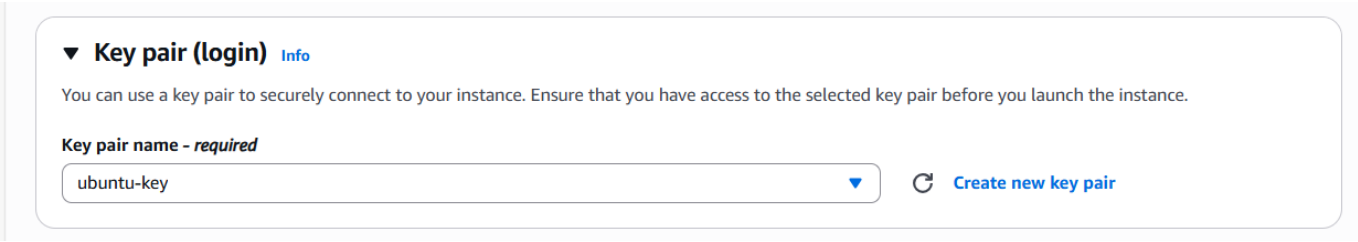
Then scroll down to "**AMI**" and select "**Ubuntu**"



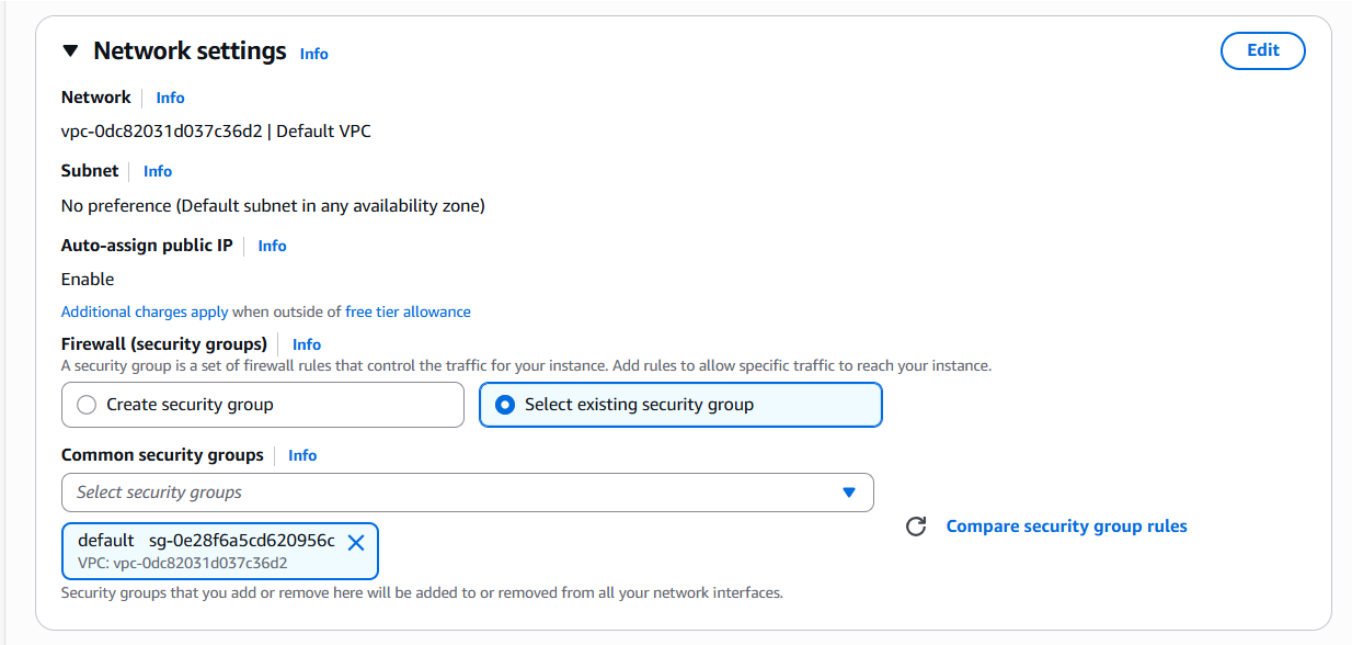
Then scroll down to "**Instance Type**" and select "**t2.small**"



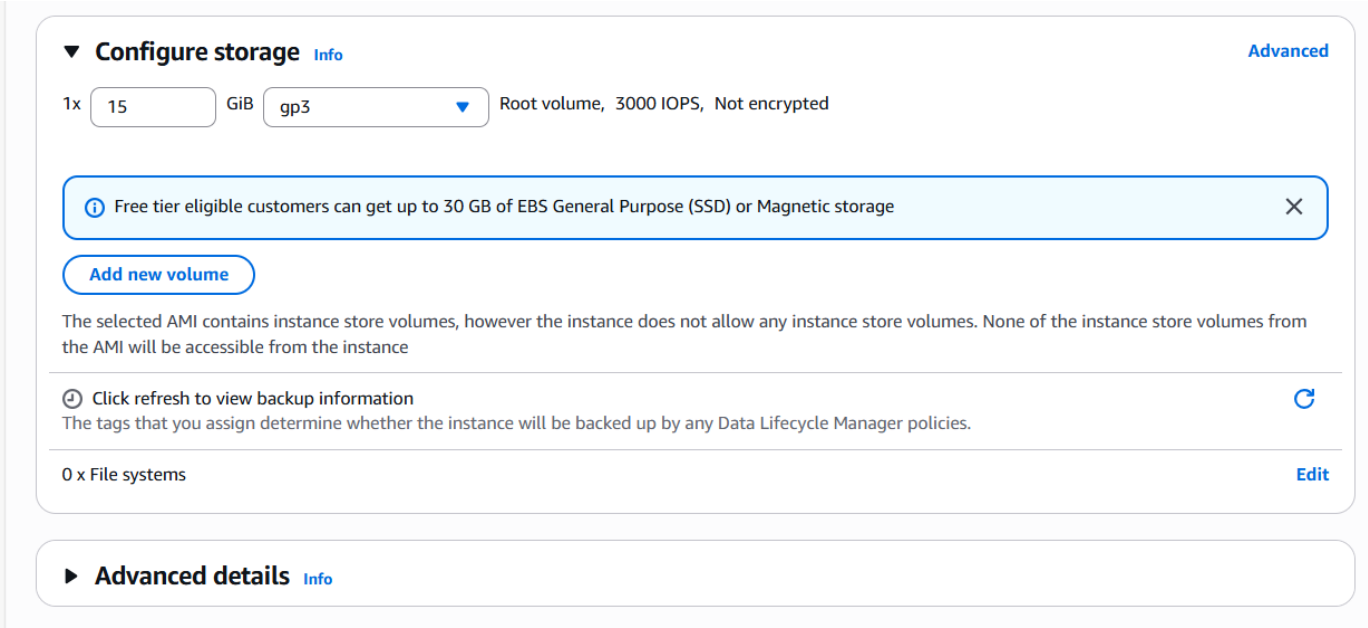
Scroll down to "**Key Pair**" and select the Key Pair we created before



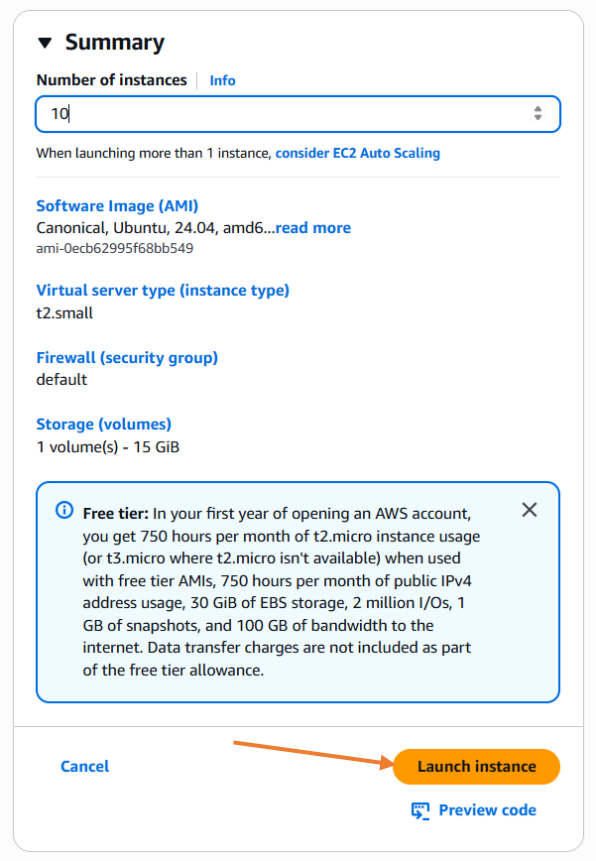
Scroll down to "**Network Settings**". We will use the default VPC and the default Security Group.



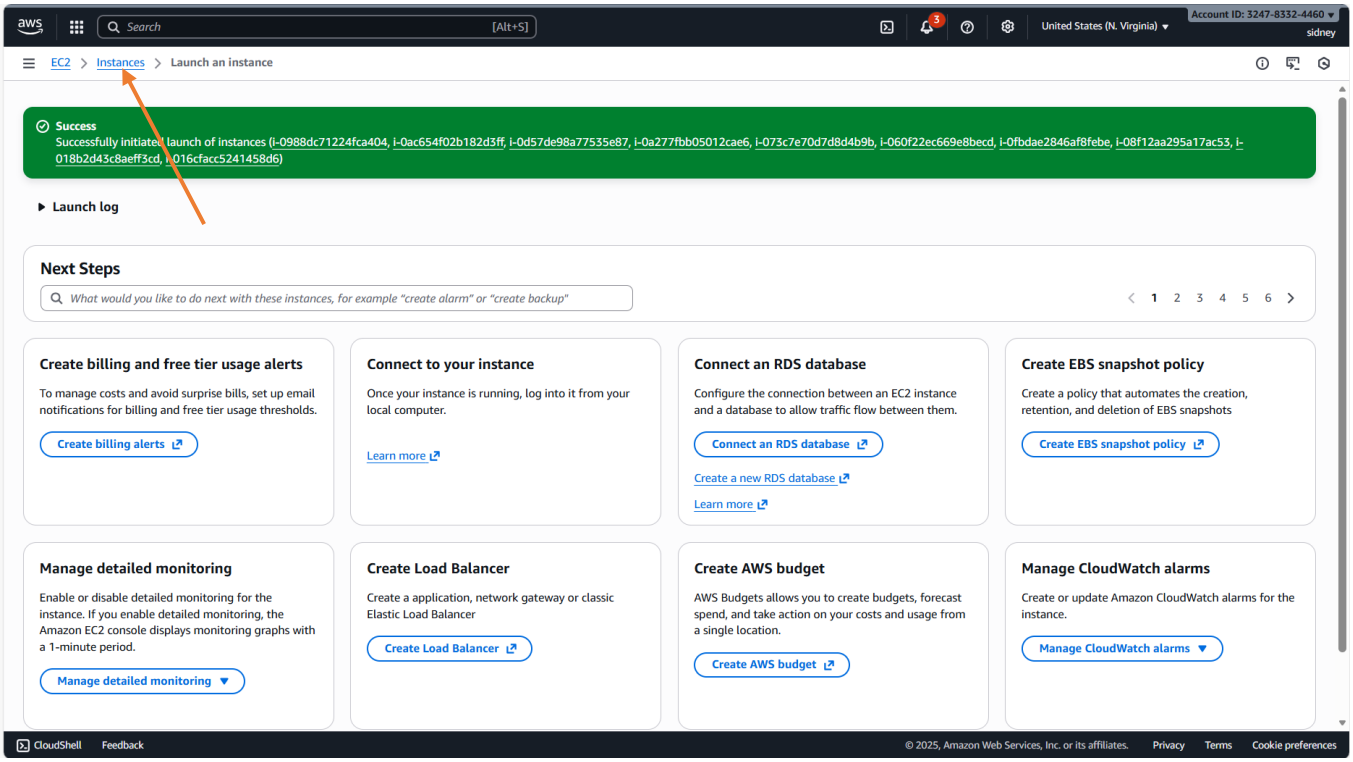
Then scroll down to "**Configure Storage**" and make it "**15GiB**"



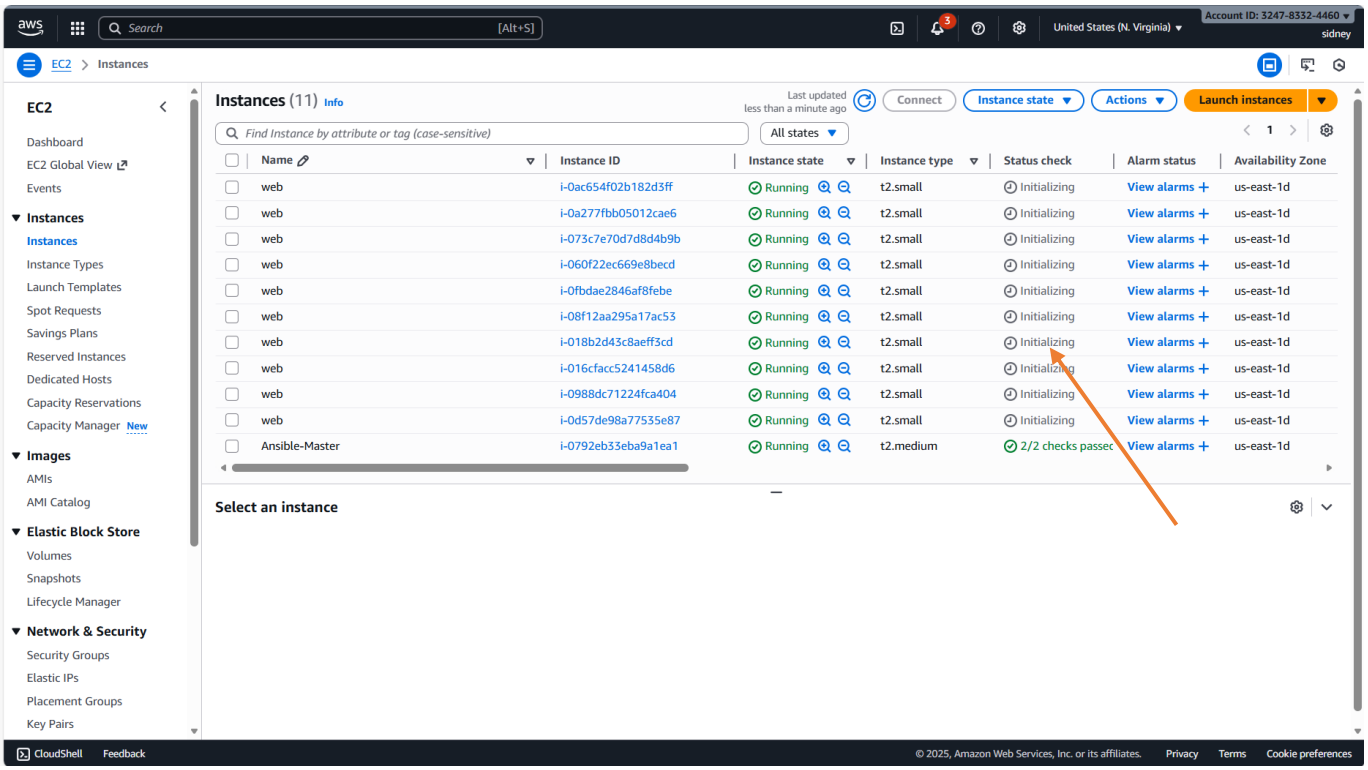
Then go to "**Summary**" and make the number of instances "**10**"



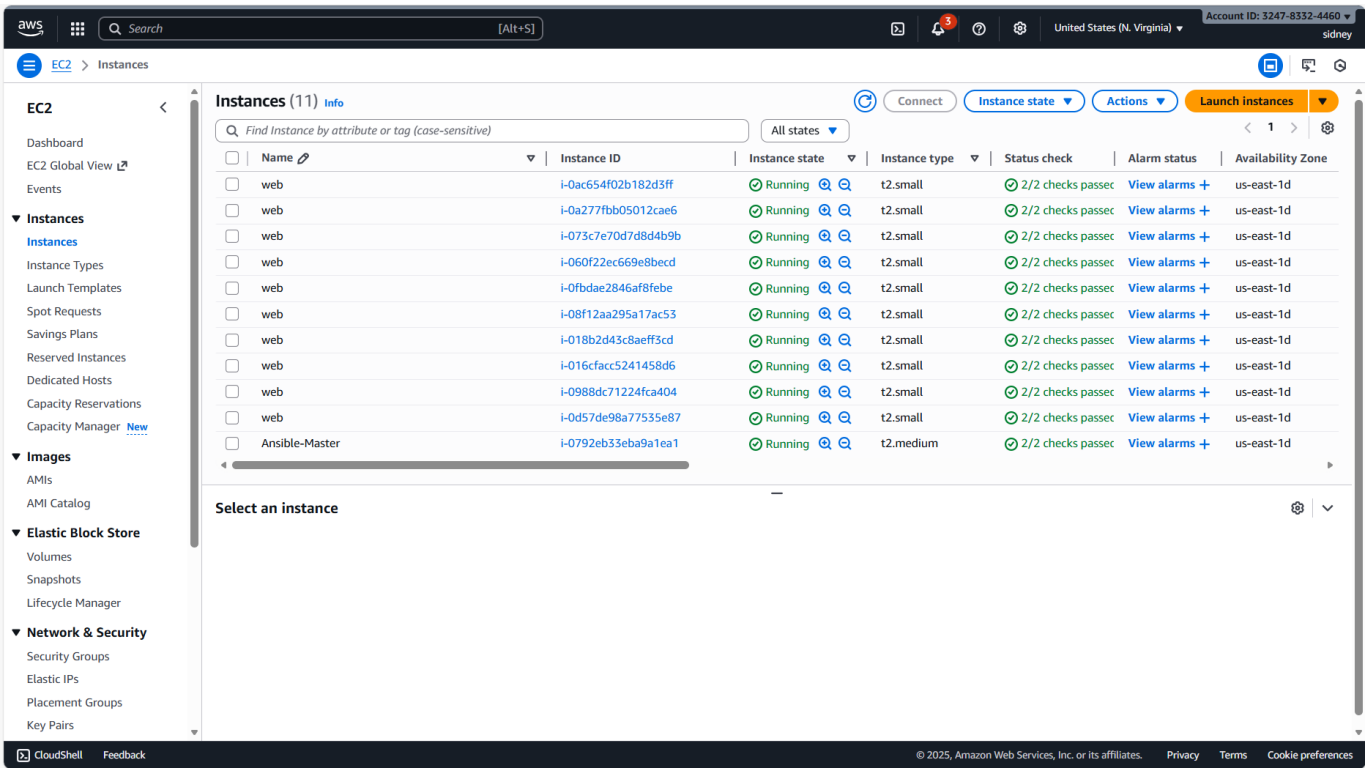
Then, click on "**Launch Instance**"



Click on "**Instances**"



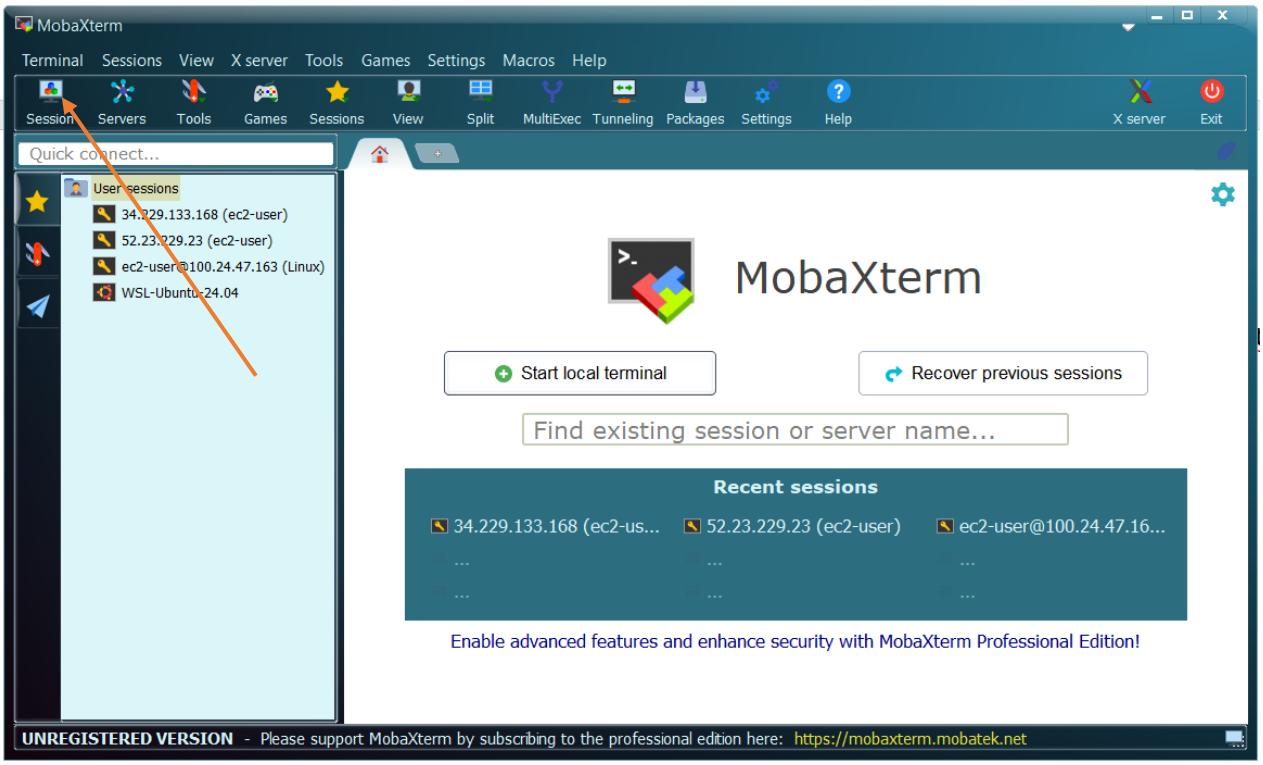
You can see the ten instances are initializing. We have to wait for them to pass the "**2/2 check**"



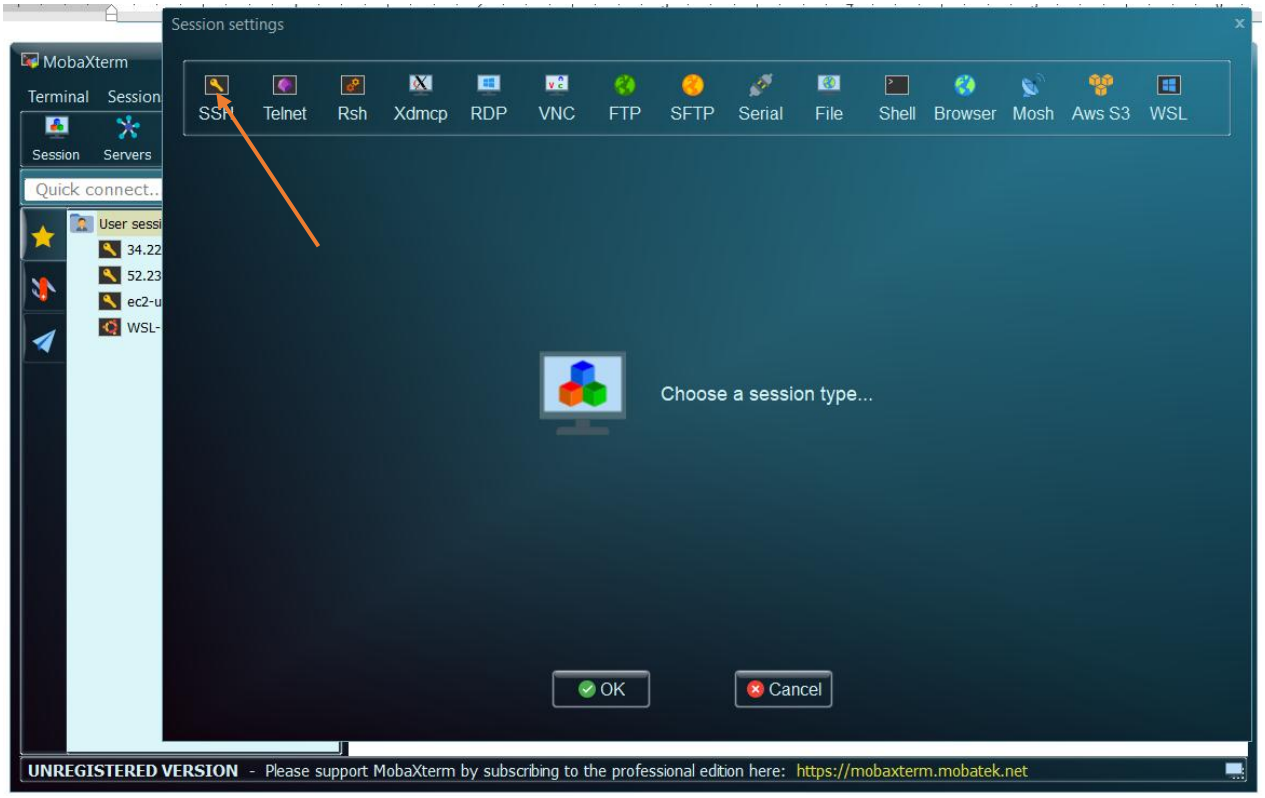
They have now passed the "**2/2 check**". Each one of the EC2 instance is named "**web**". This is not good. So, we are going to use a script to name them **web01, web03,……, web10**, so that we can differentiate between them.

Part 3: SSH Connect to the Master EC2 Instance

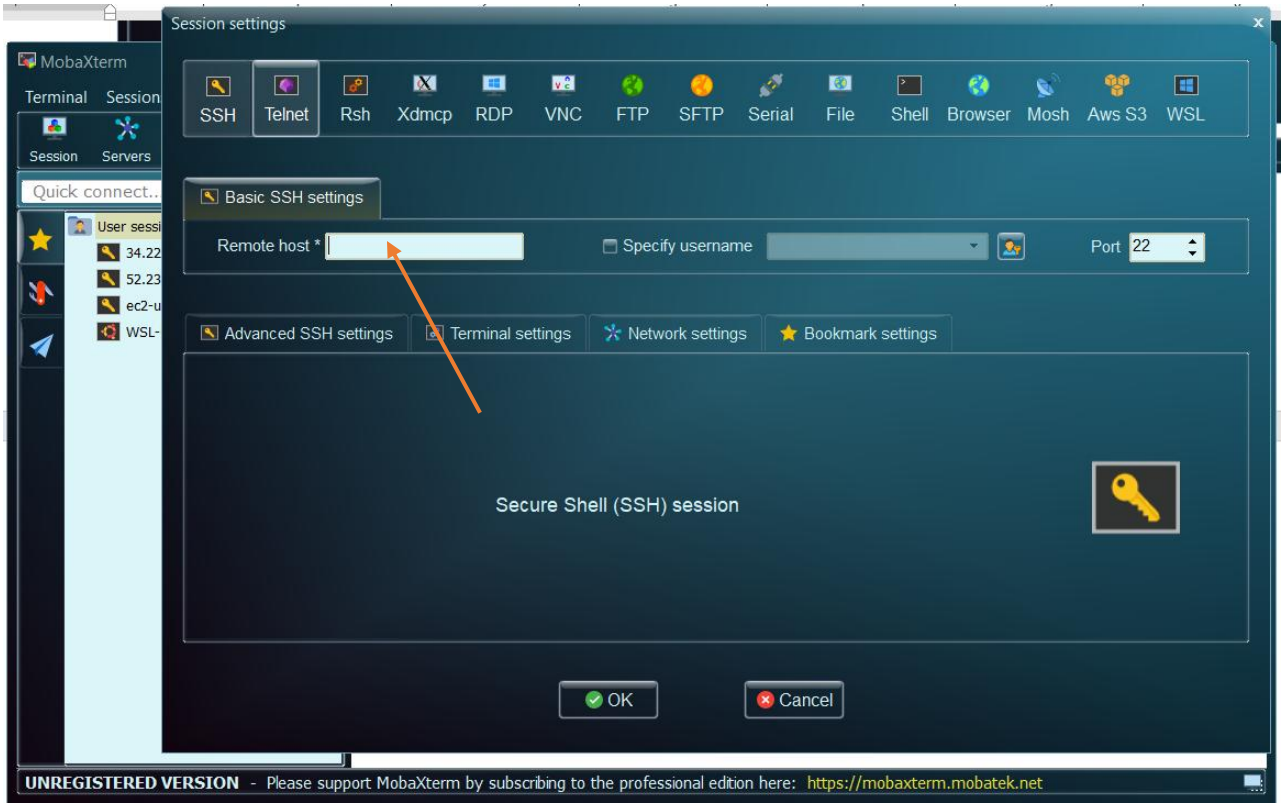
We will now SSH connect to the "**Ansible-Master**" instance using MobaXterm. Open MobaXterm.



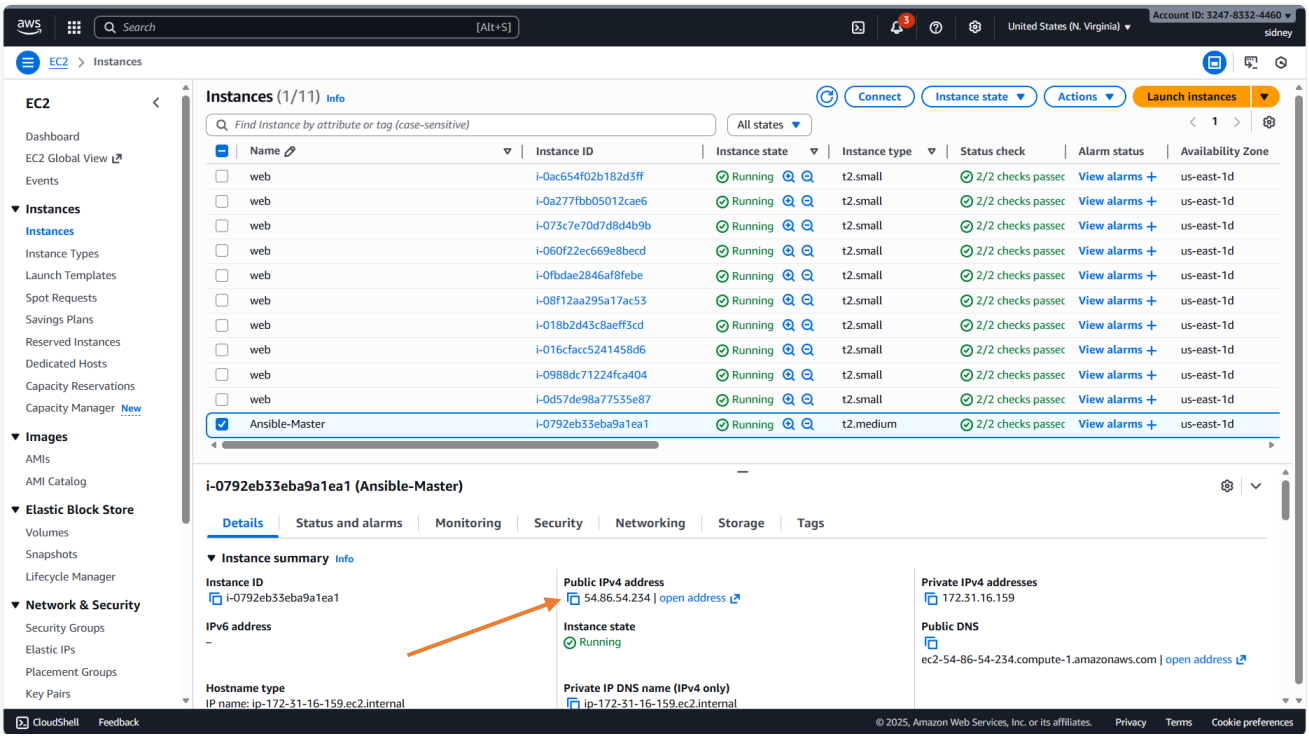
Click on "**Session**"



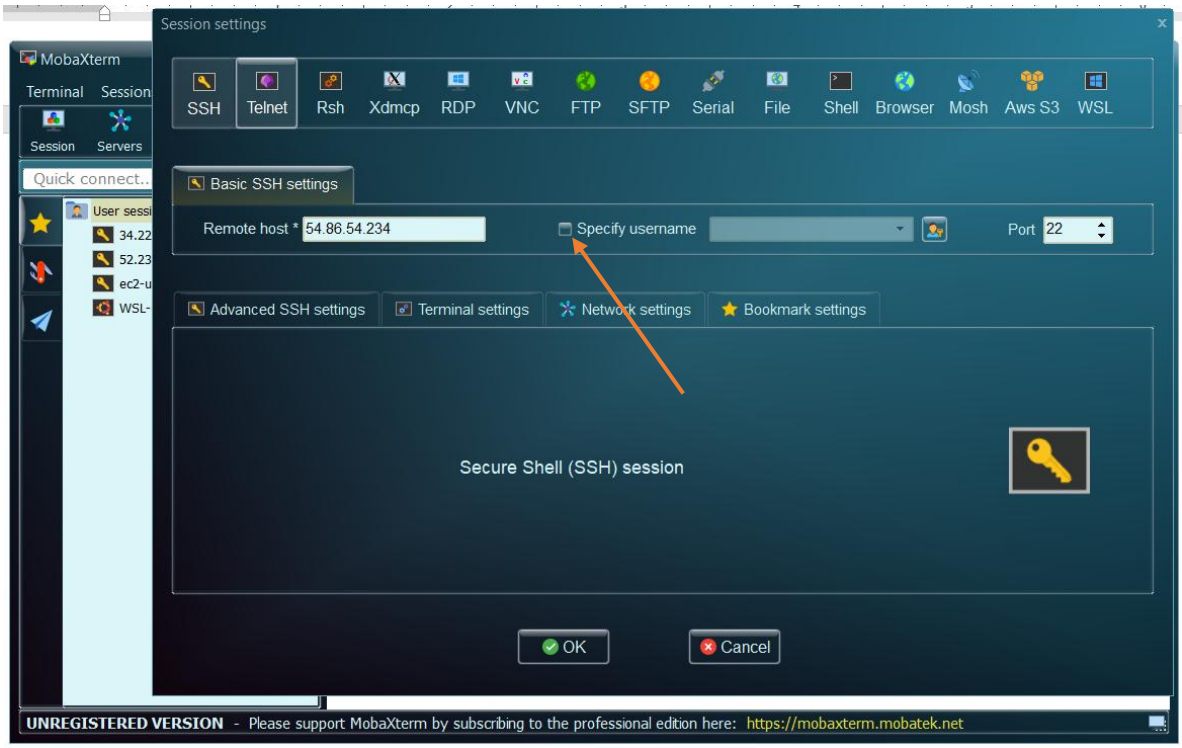
Click on "**SSH**"



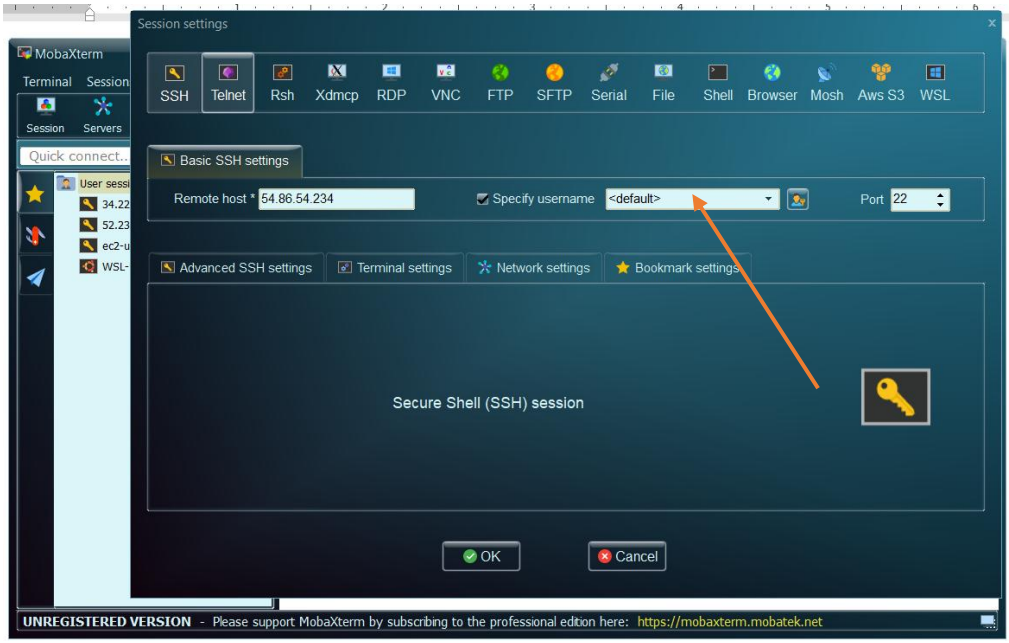
Go to our "**Ansible-Master**" instance



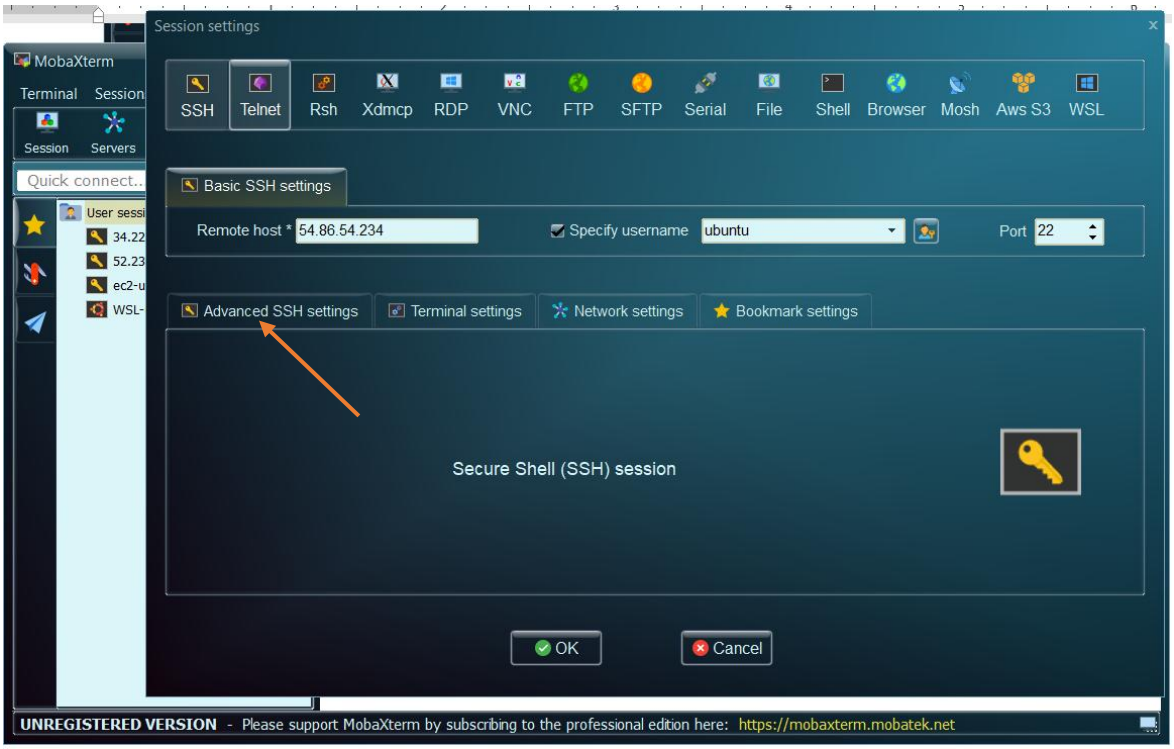
Copy the "**Public IPv4 address**" and paste on "**Remote Host**" on MobaXterm



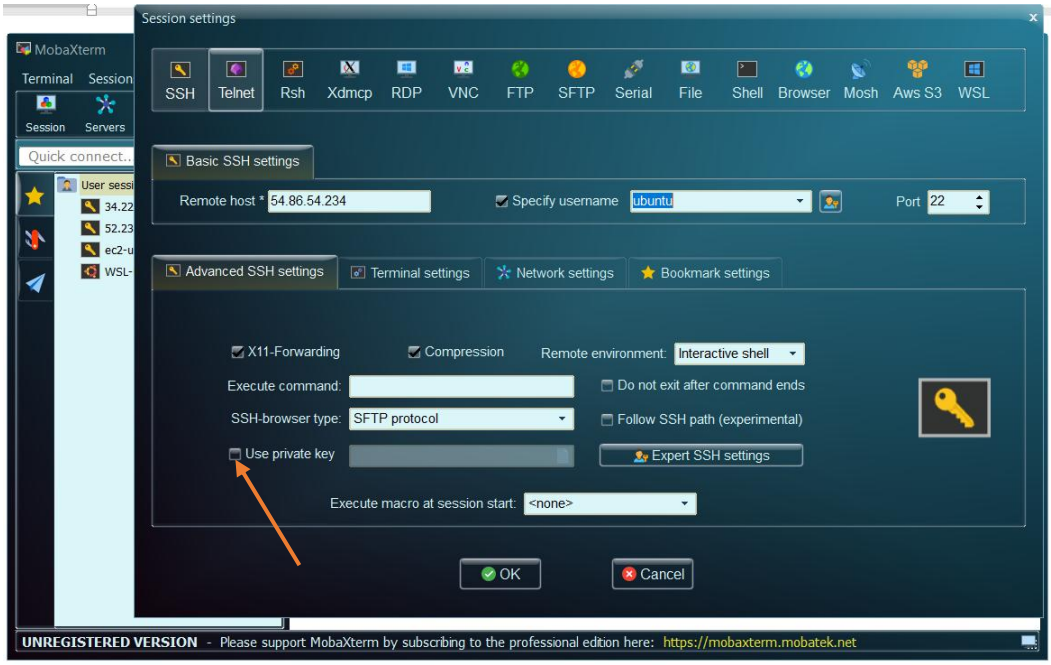
Select "**Specify username**"



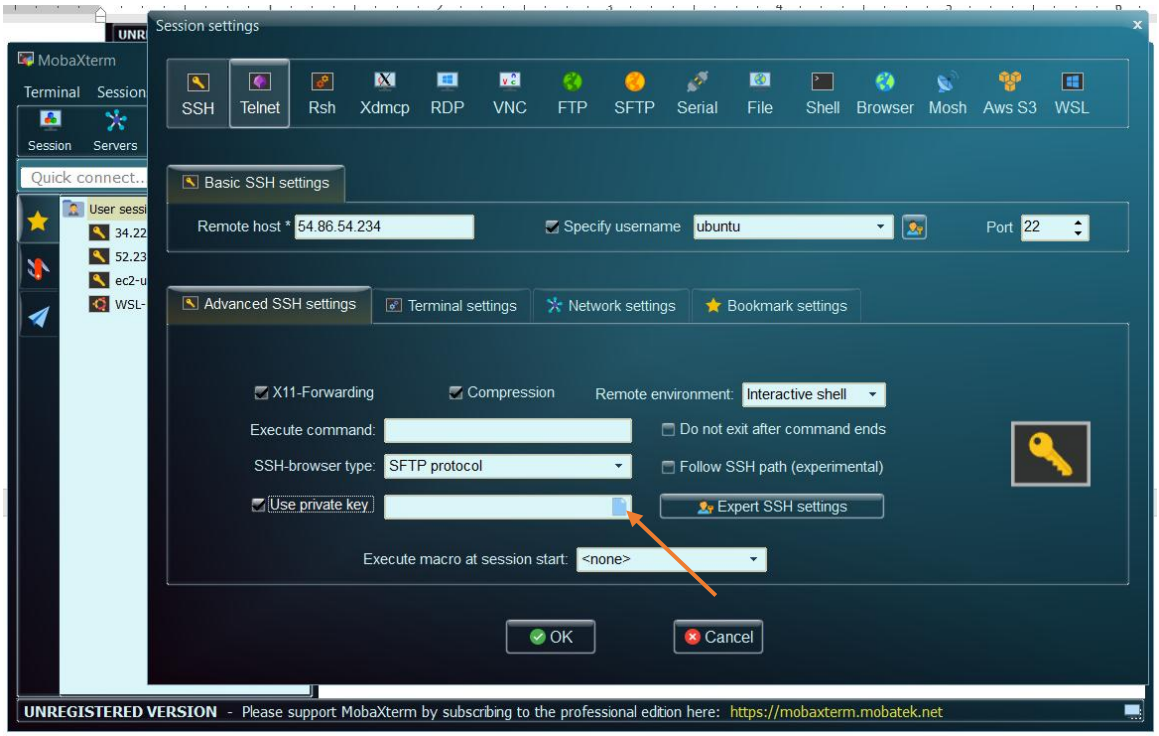
Then, enter "**ubuntu**" for username



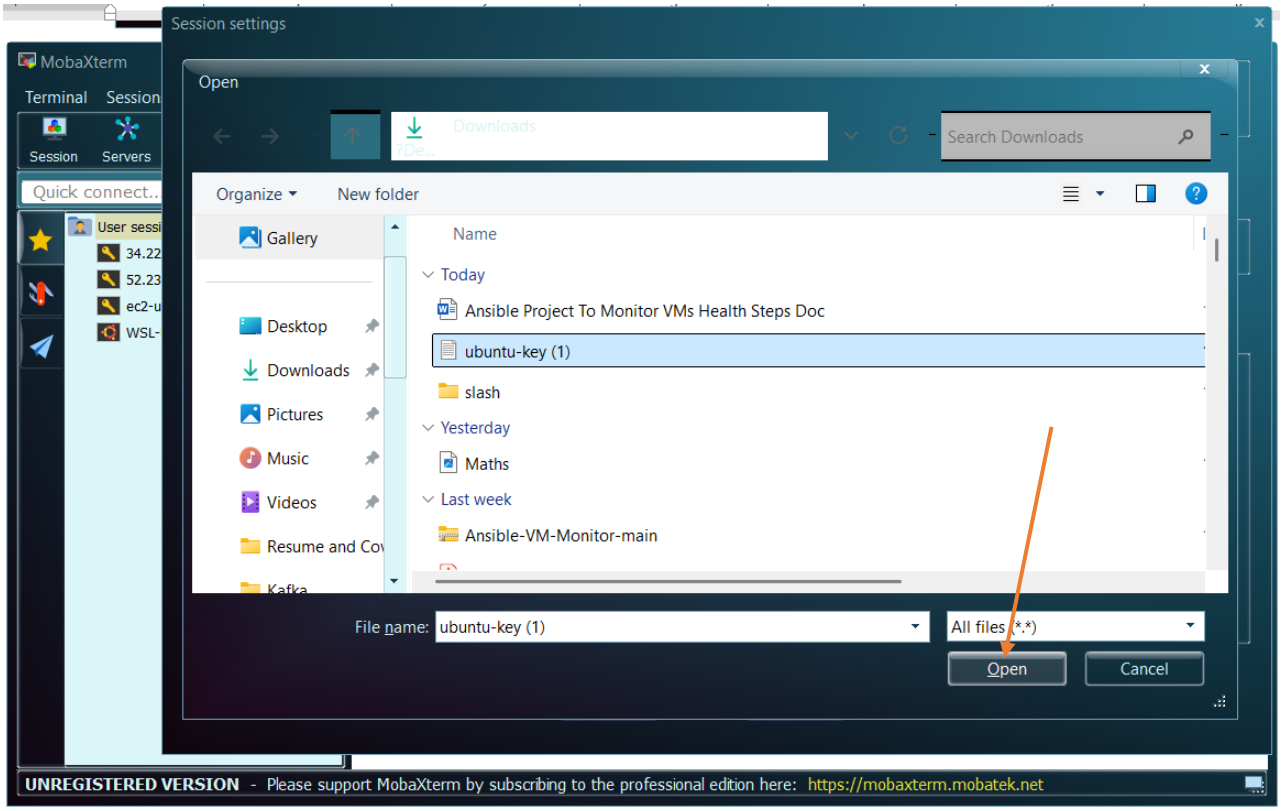
Then click on "**Advanced SSH Settings**"



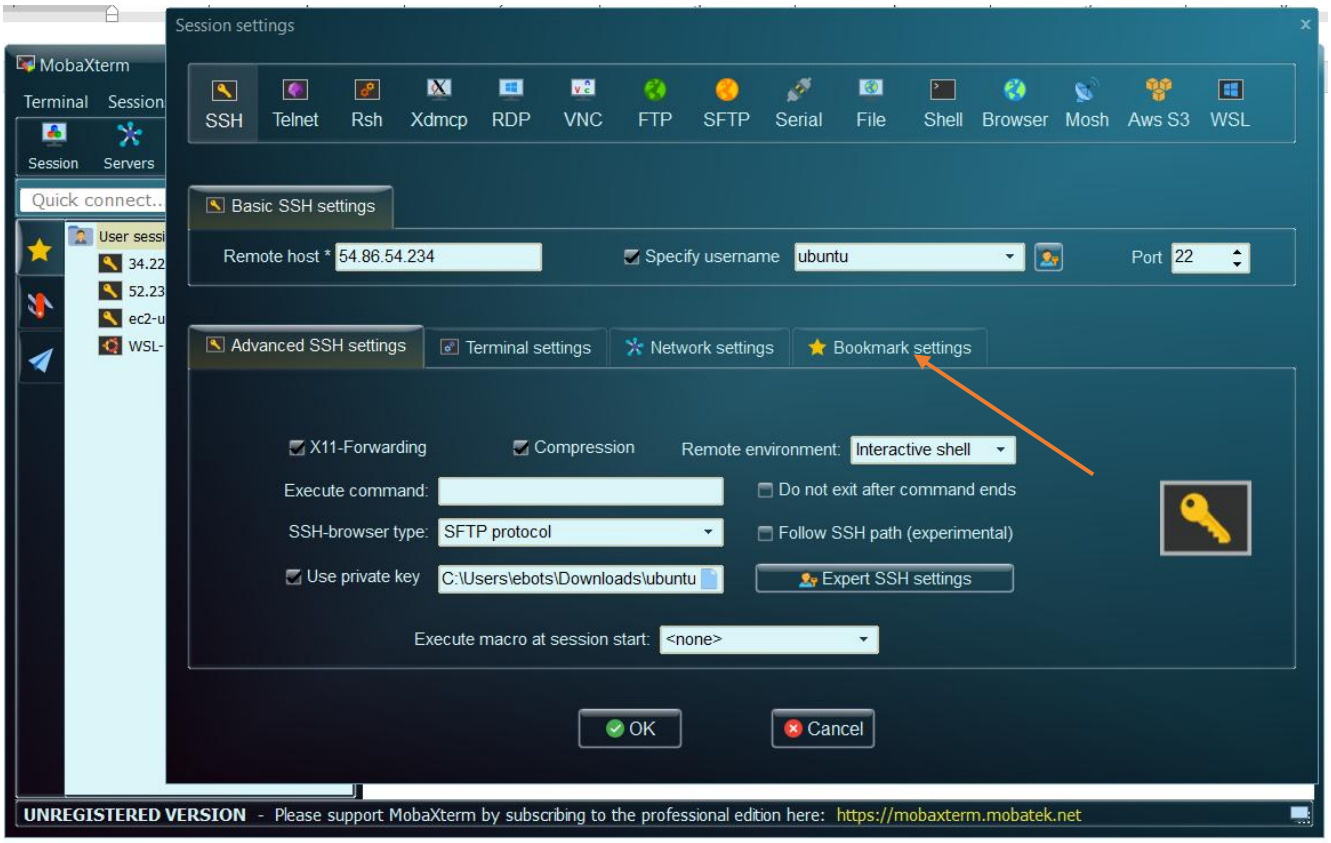
Check "**Use private key**"



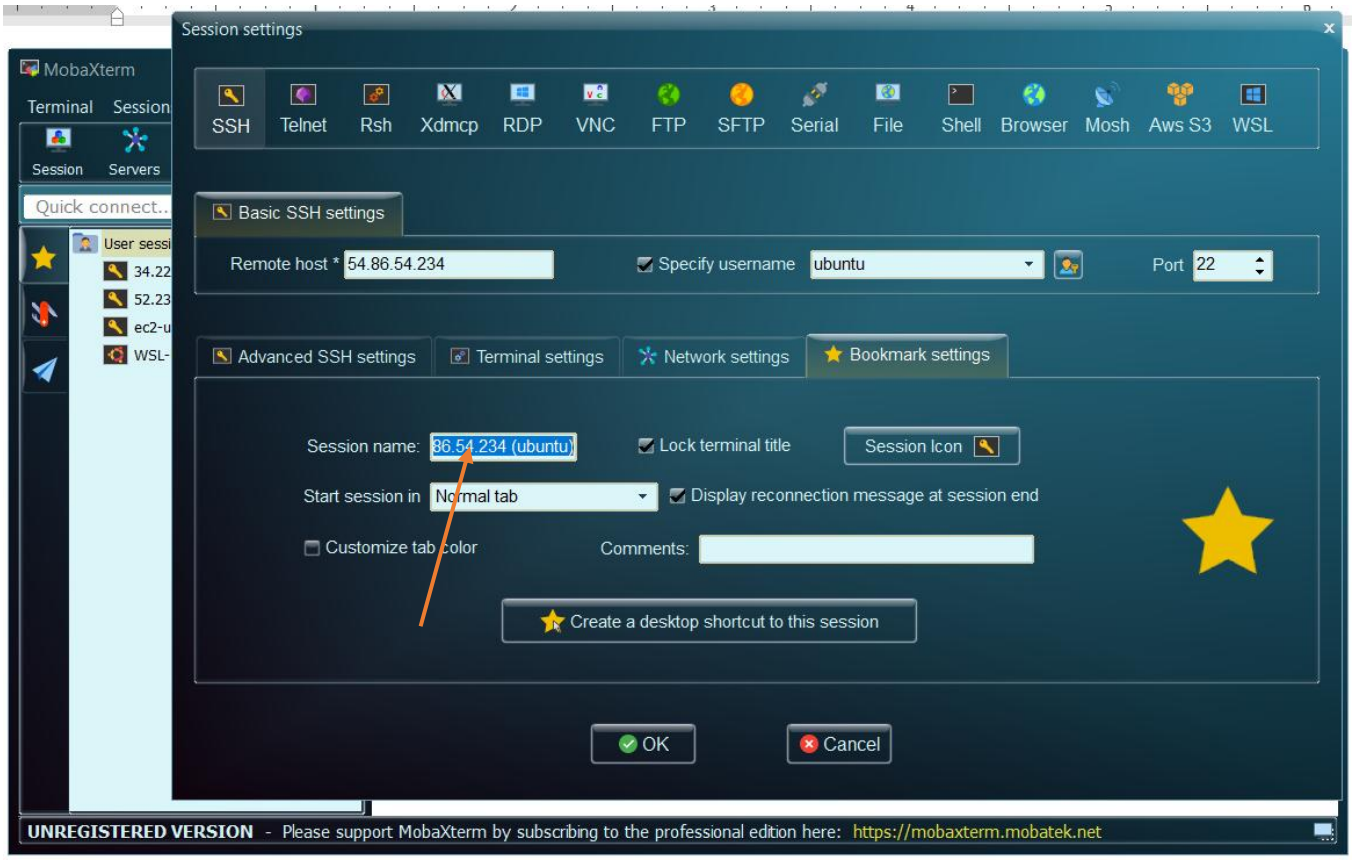
Then browse and select our **.pem** file for Key pair saved in Downloads folder



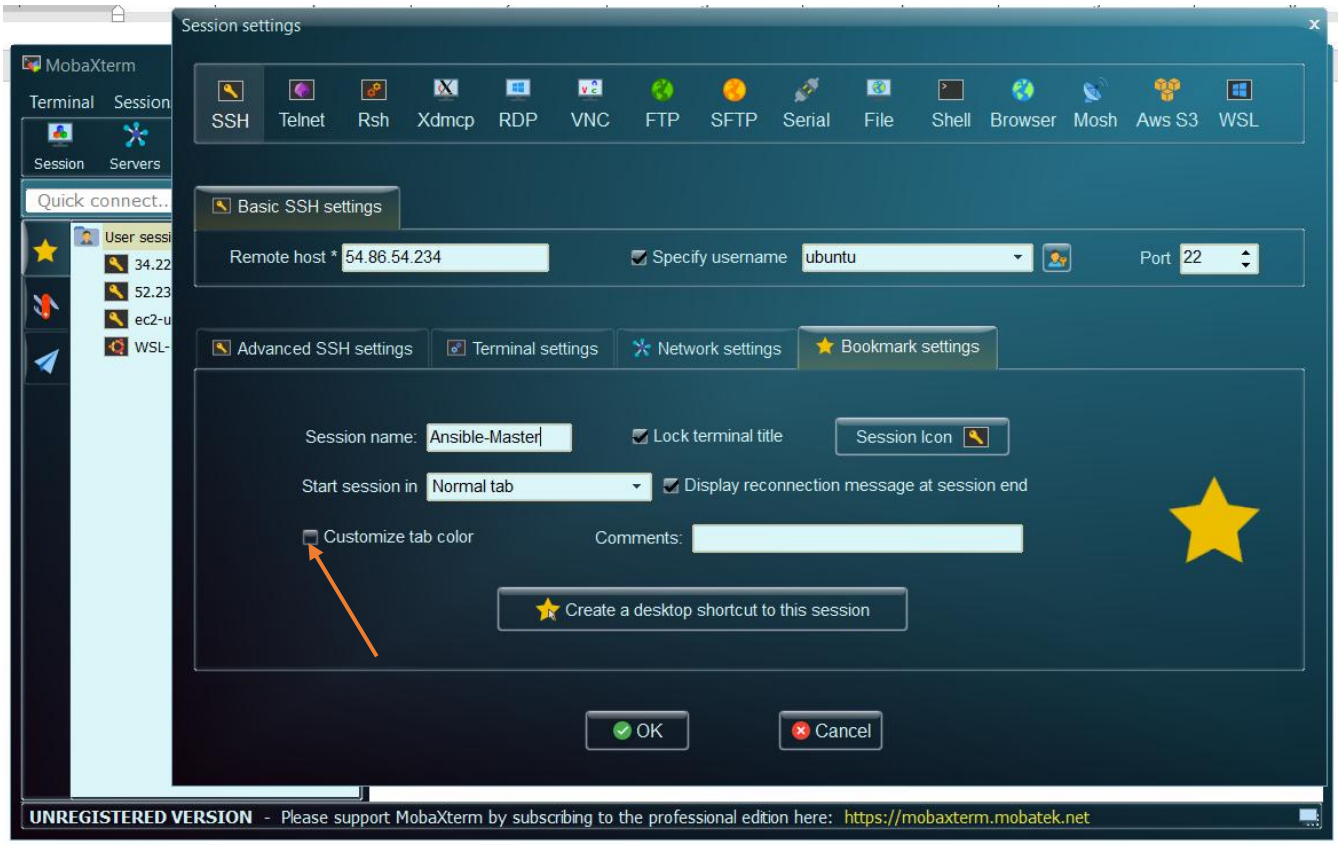
Click on "**Open**"



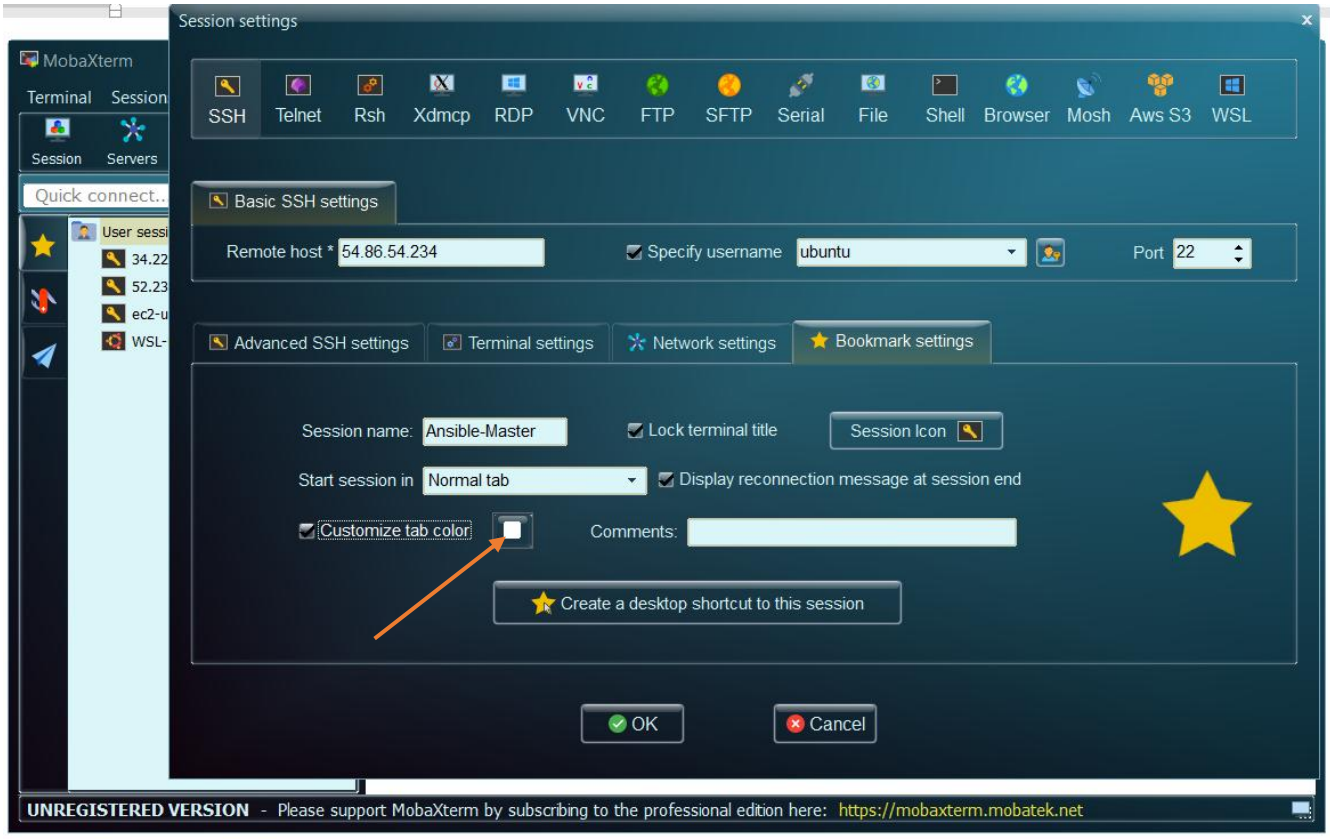
Then name the machine by clicking on "**Bookmark settings**"



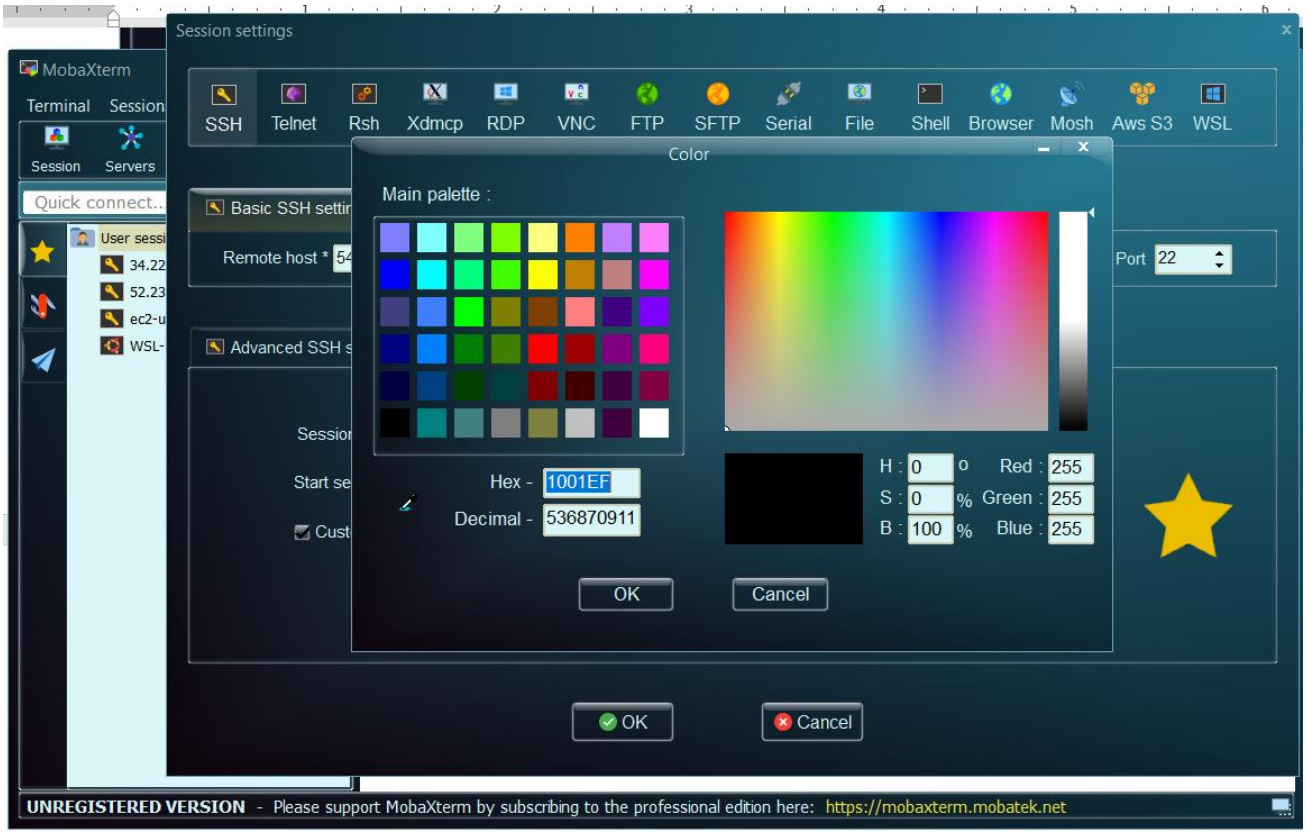
Then for "**Session name**", enter "**Ansible-Master**"



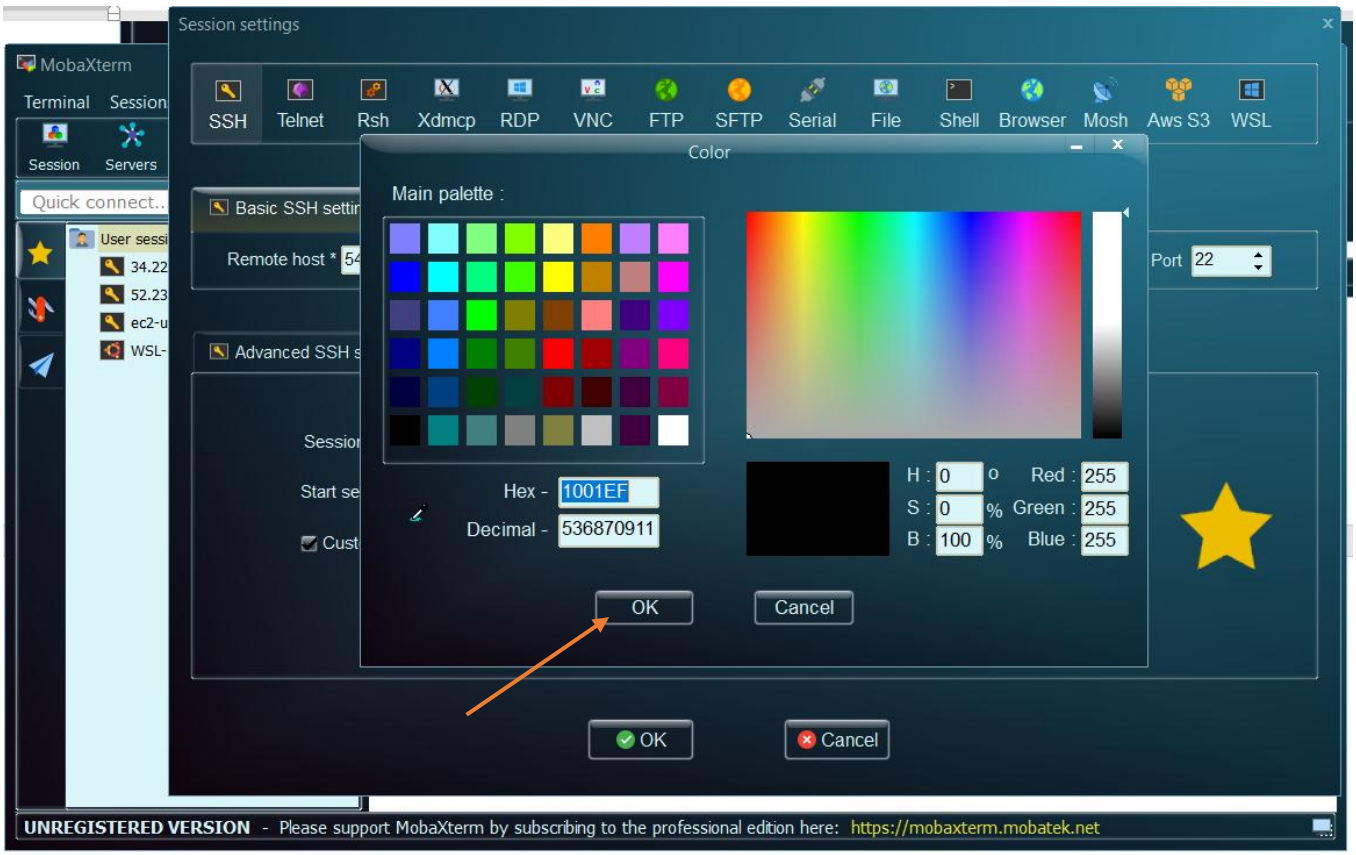
We can give the tab a colour by clicking on "**Customize tab colour**"



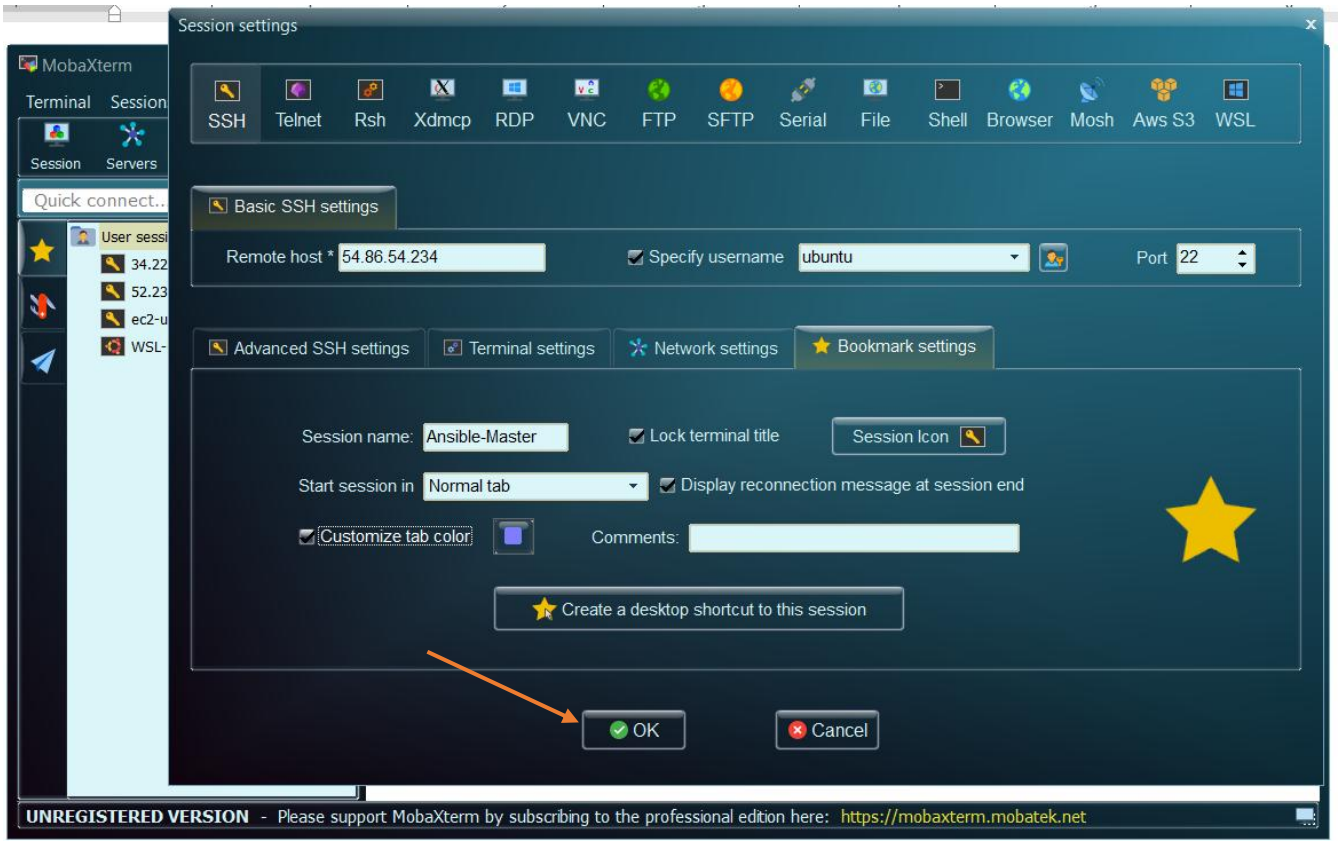
Then browse and select the colour



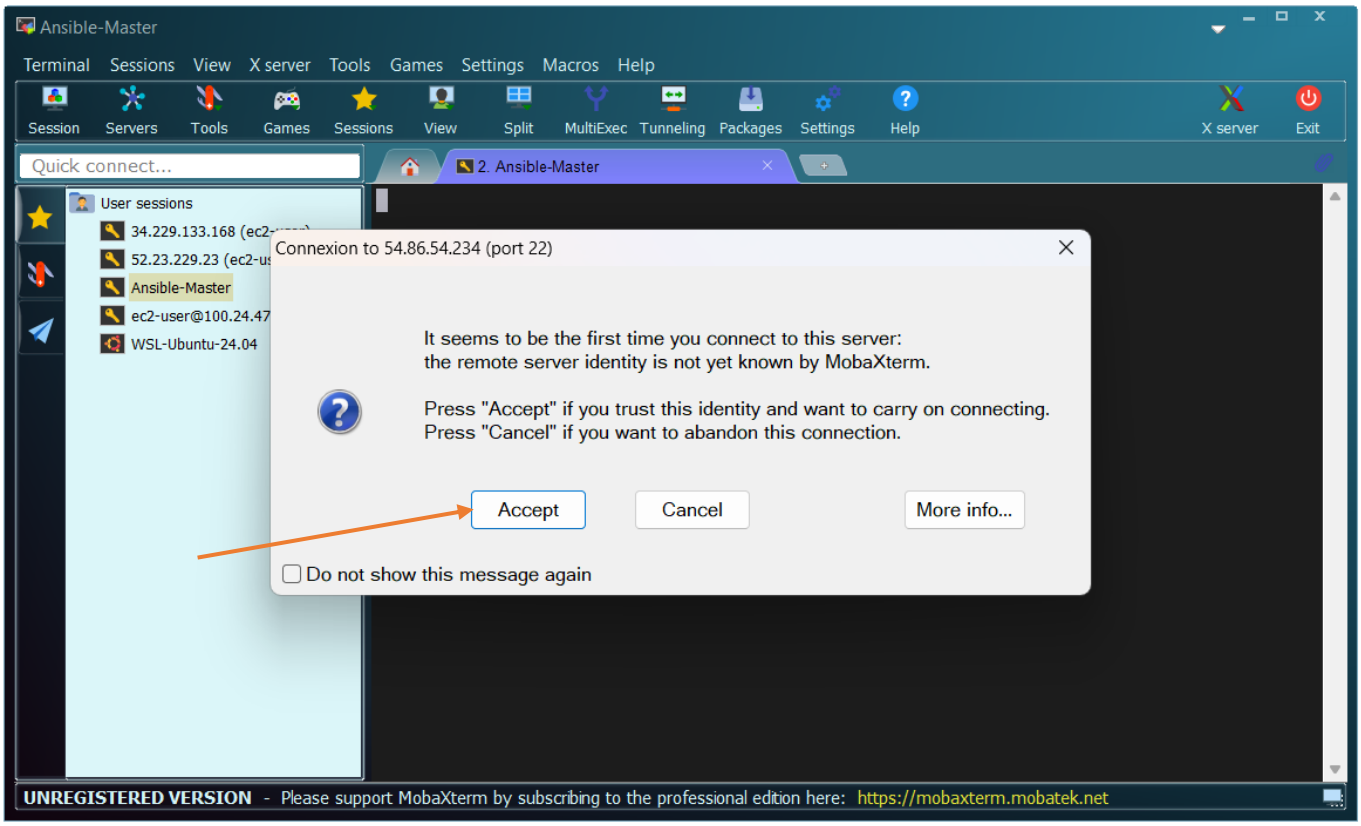
Select the colour you want to use.



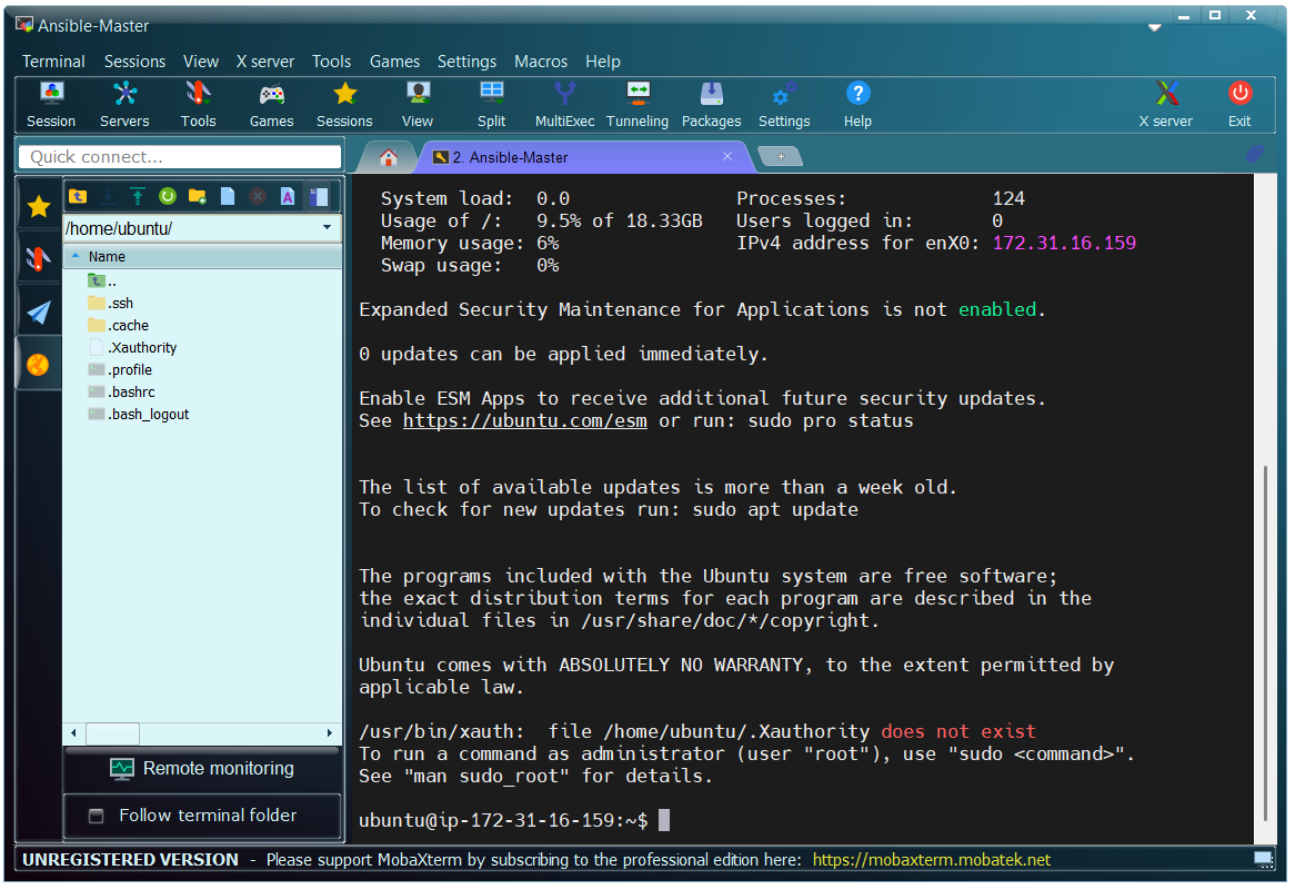
And press "**OK**"



Then click on "**OK**"



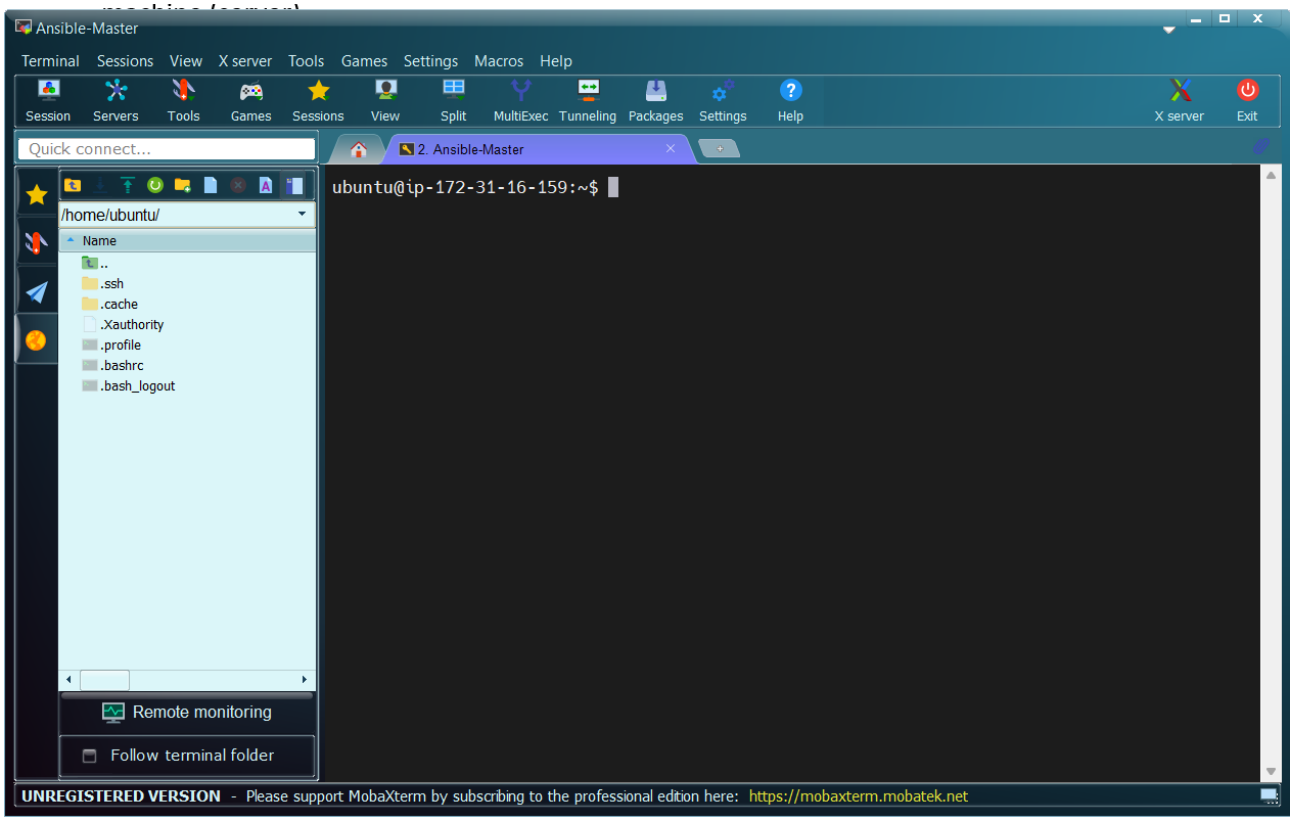
Then click on "**Accept**"



We are now connected to our "**Ansible-Master**".

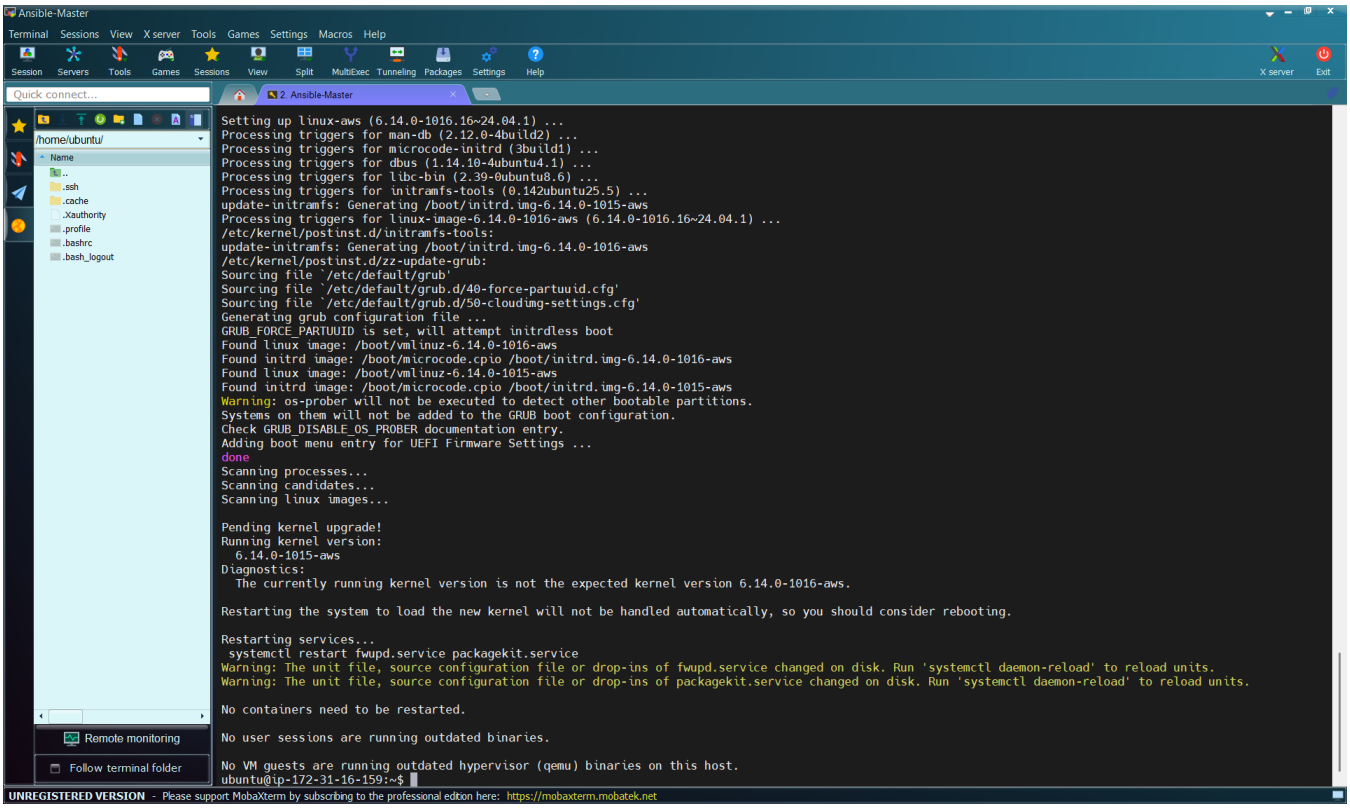
Part 4: Install Ansible on "Ansible-Master" Instance

The next thing we have to do is to install Ansible on this machine (server).



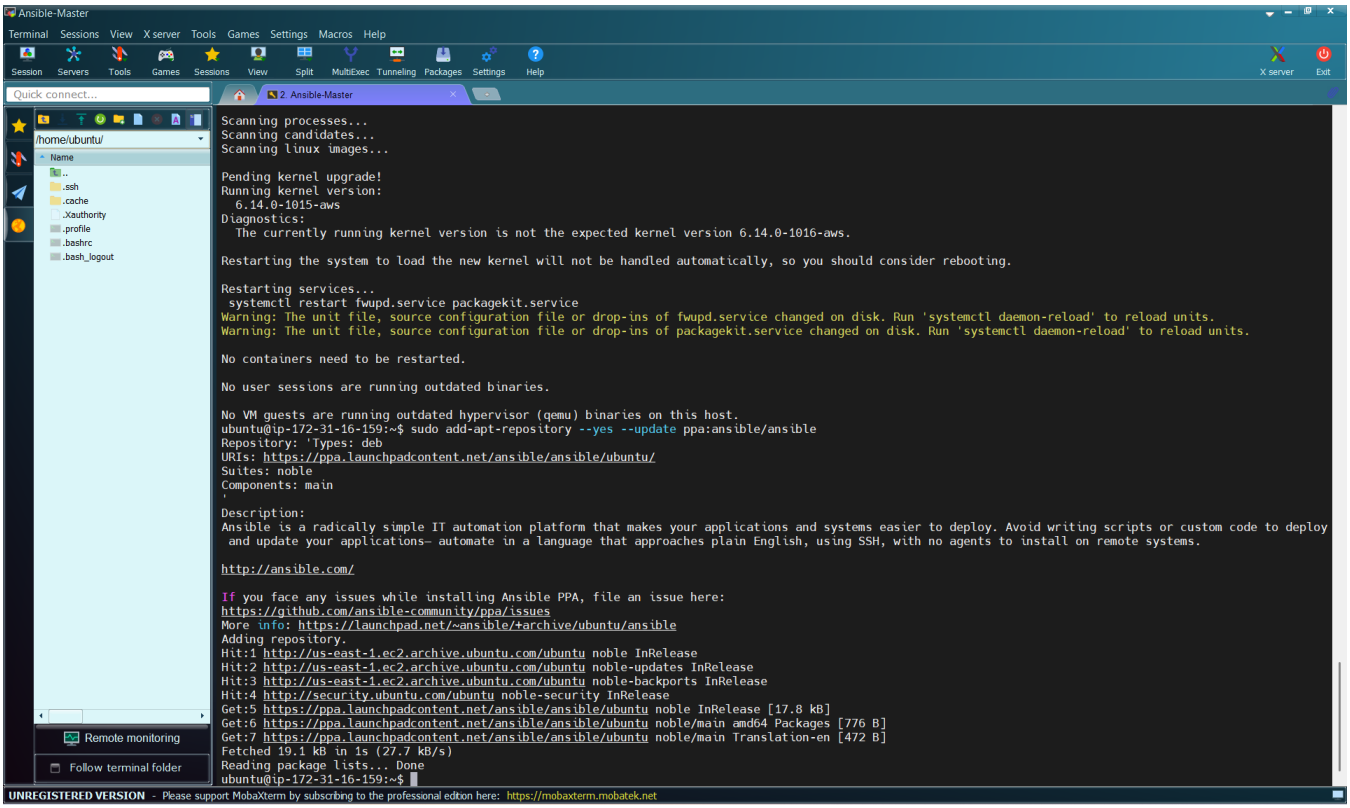
We have to first update and upgrade the system using the command:

```
sudo apt update && sudo apt upgrade -y
```



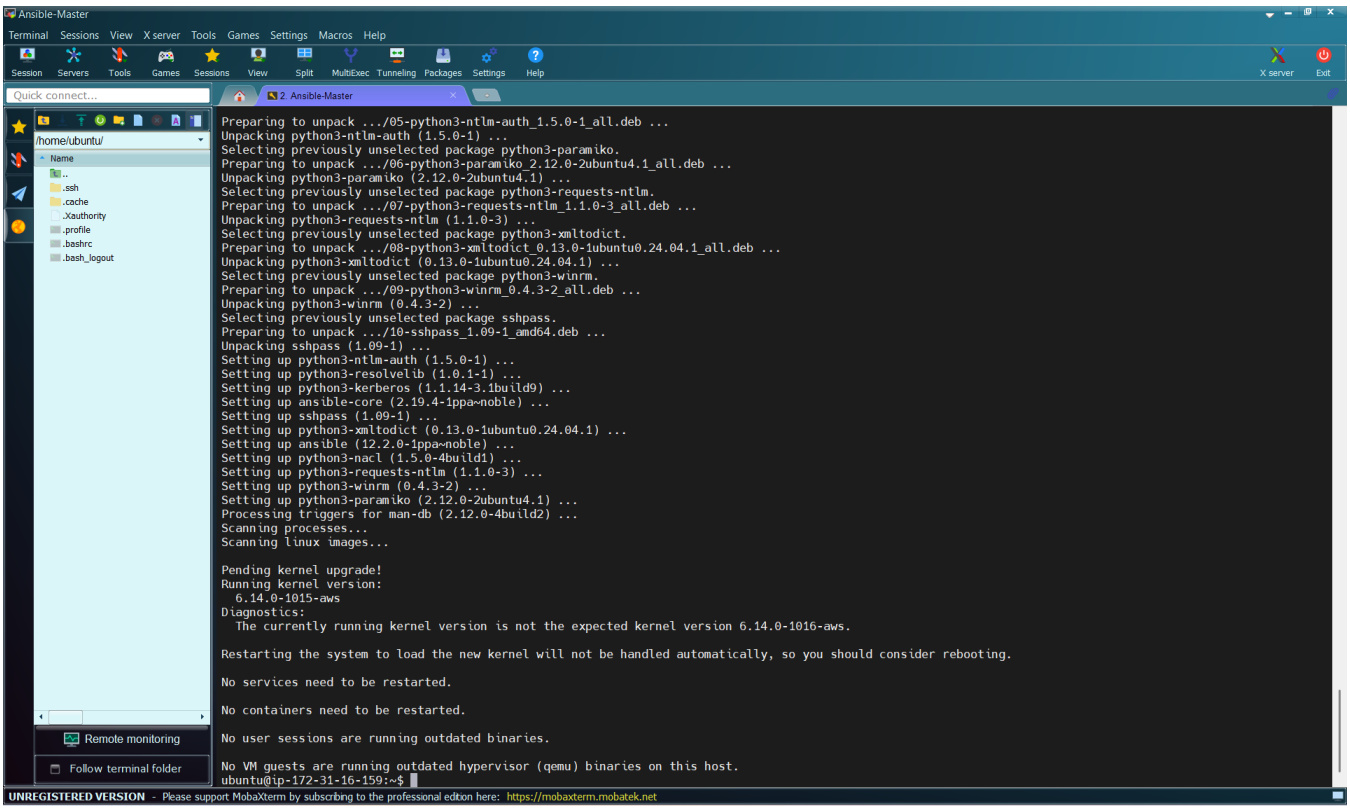
Next, add the Ansible PPA. Ansible provides an official maintained PPA (for latest versions):

```
sudo add-apt-repository --yes --update ppa:ansible/ansible
```



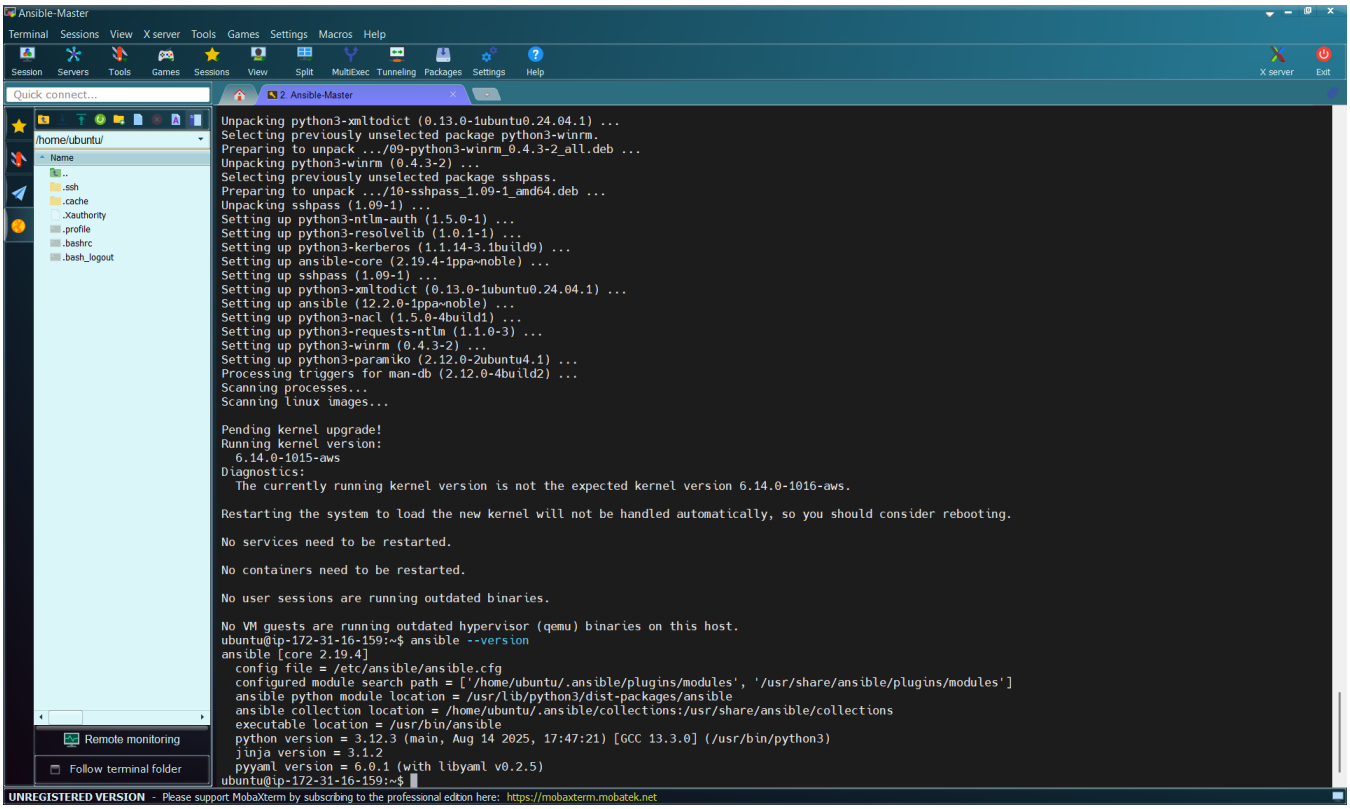
Then Install Ansible

```
sudo apt install ansible -y
```



Now, let us verify the installation using the command:

```
ansible --version
```

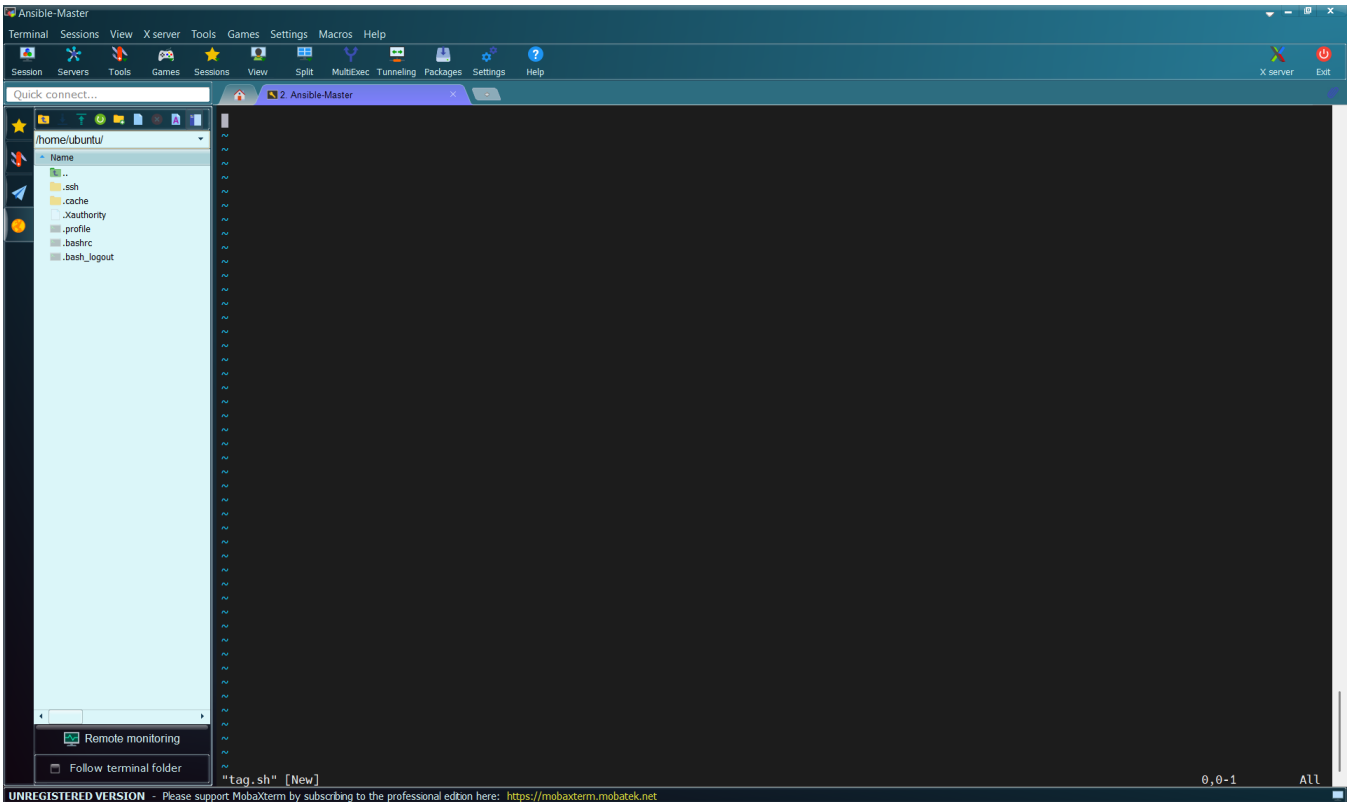


You can see that Ansible has been successfully installed

Part 5: Tagging “Target” instances

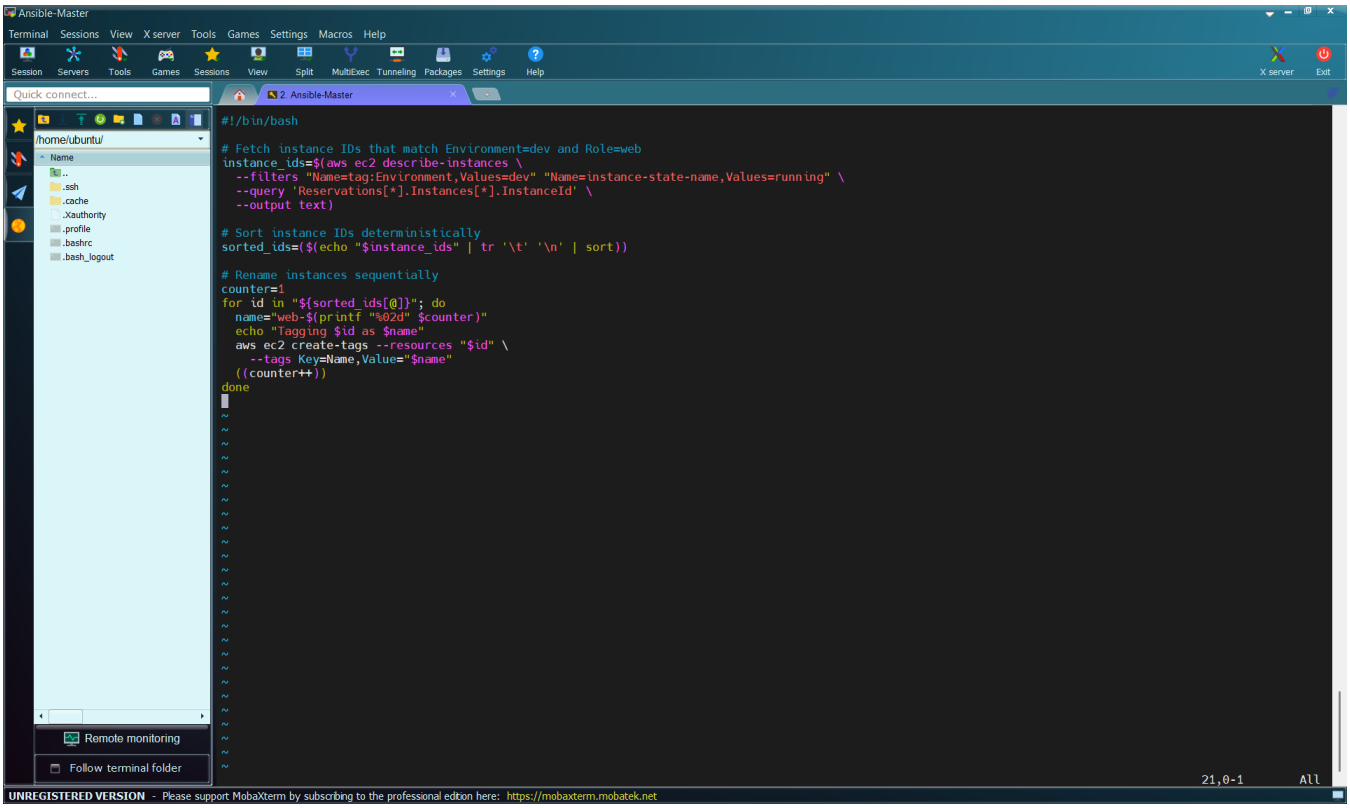
We now have to rename the target instances using a script. We will call the script “**tag.sh**”. We create the script using the command:

```
vi tag.sh
```

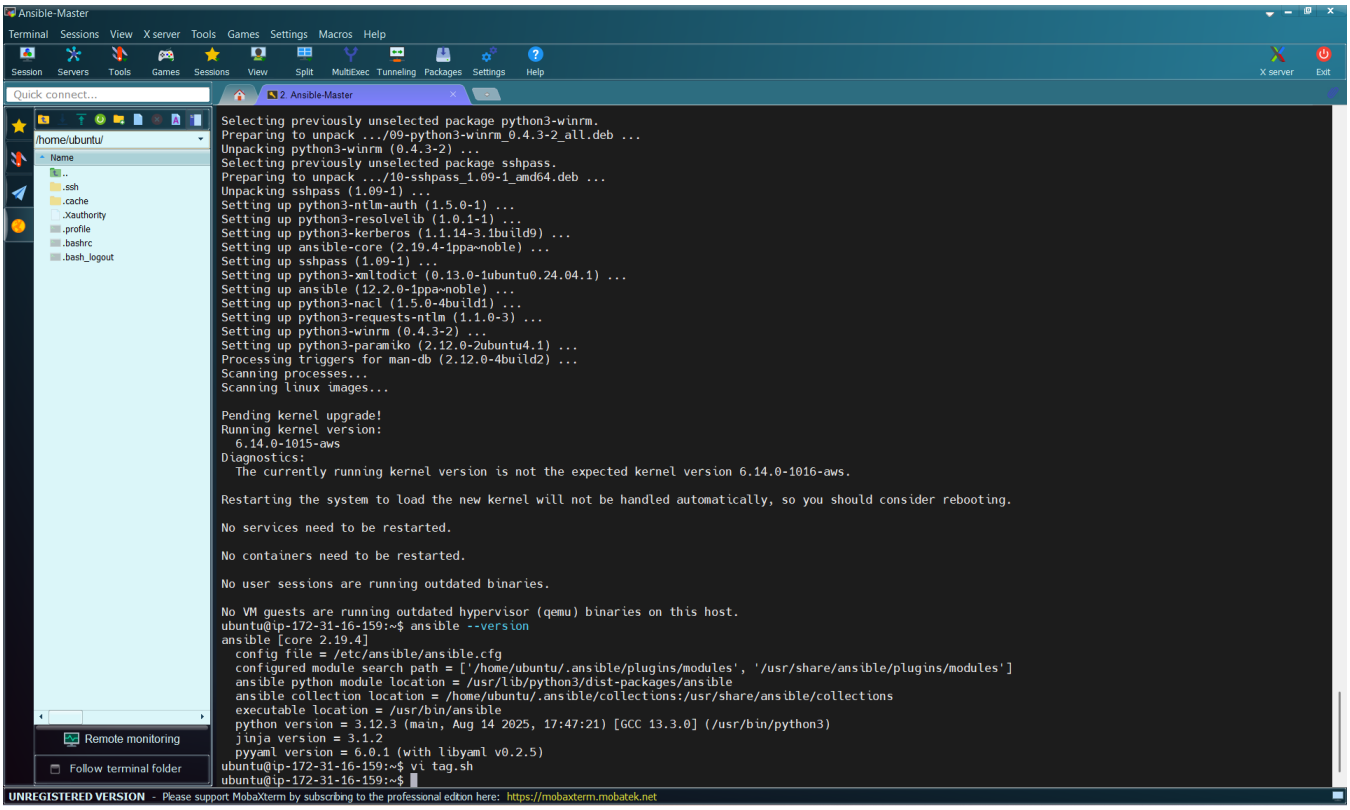


Then, we paste the code for the script

```
#!/bin/bash

# Fetch instance IDs that match Environment=dev and Role=web
instance_ids=$(aws ec2 describe-instances \
  --filters "Name=tag:Environment,Values=dev" "Name=instance-state-name,Values=running" \
  --query 'Reservations[*].Instances[*].InstanceId' \
  --output text)

# Sort instance IDs deterministically
sorted_ids=($(echo "$instance_ids" | tr '\t' '\n' | sort))

# Rename instances sequentially
counter=1
for id in "${sorted_ids[@]}"; do
  name="web-$(printf "%02d" $counter)"
  echo "Tagging $id as $name"
  aws ec2 create-tags --resources "$id" \
    --tags Key=Name,Value="$name"
  ((counter++))
done
```



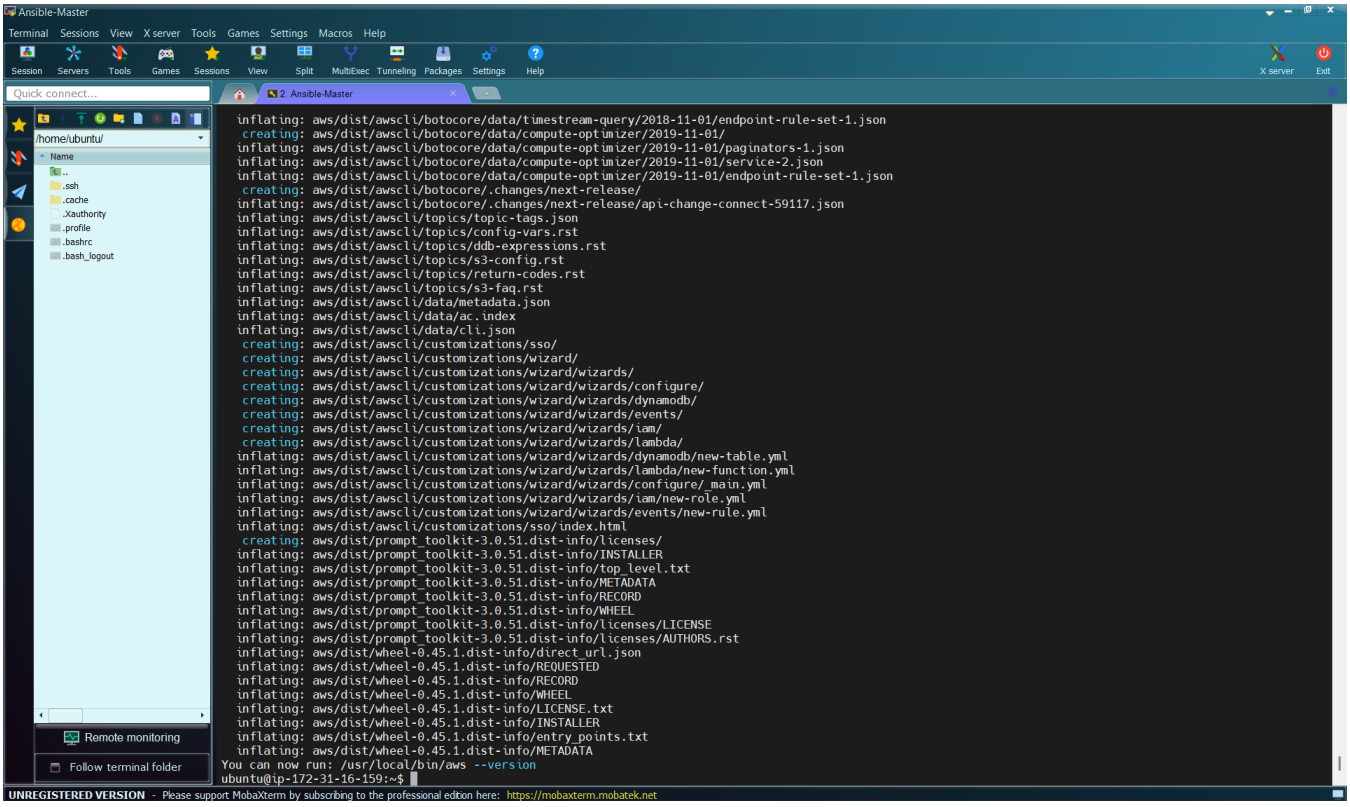
Save it by Press “**ESC**”, type :wq and Press “**enter**”



Part 6: Install AWS CLI

We have to install AWS CLI by running the commands below

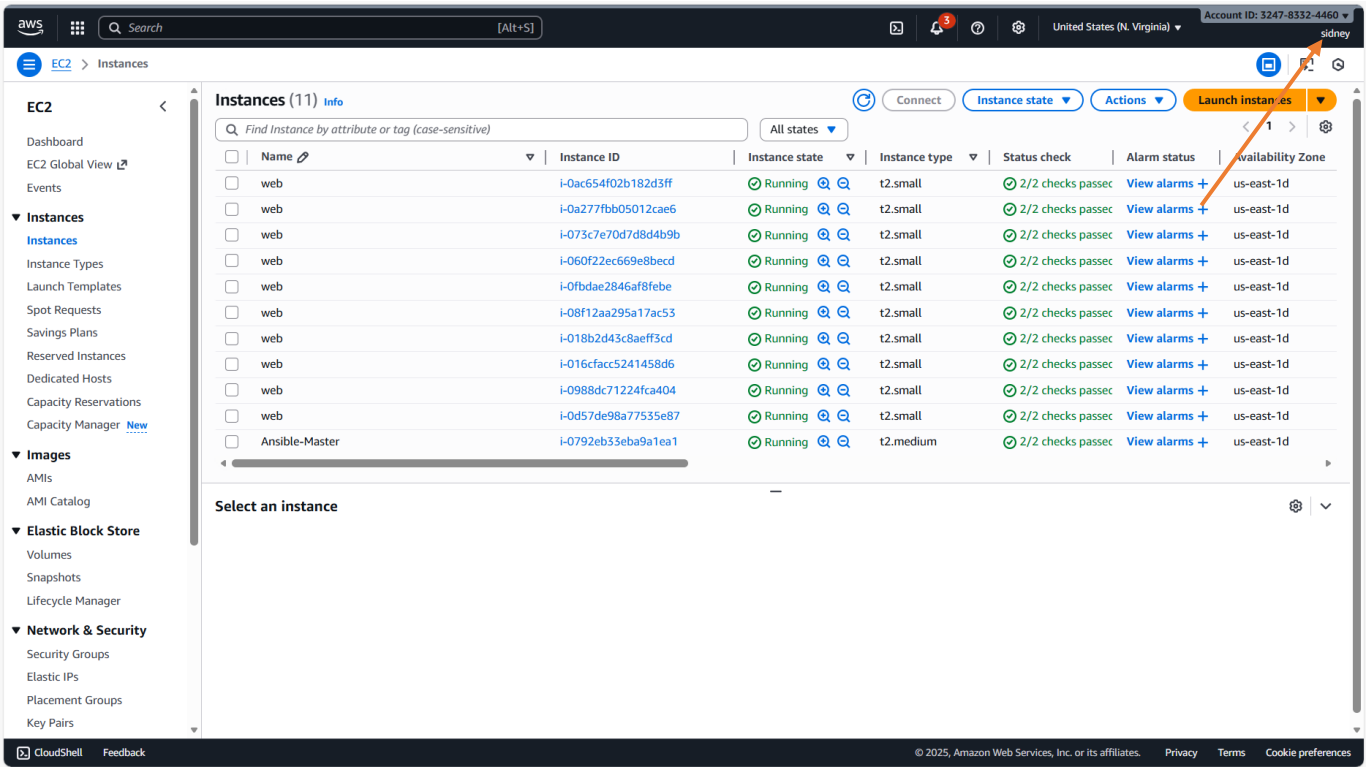
```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
sudo apt install unzip
unzip awscliv2.zip
sudo ./aws/install
```



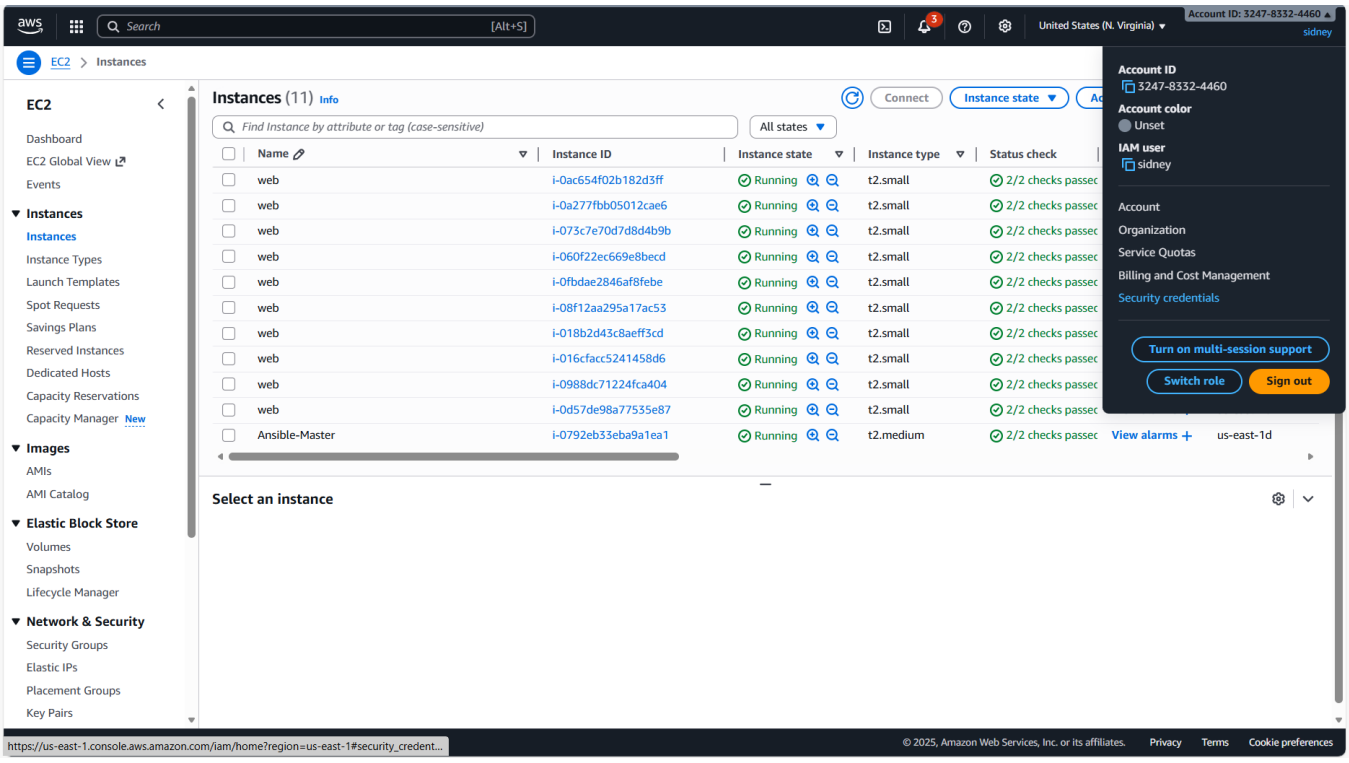
Part 7: Connect to AWS Account

The aws configure command is used to set up the AWS Command Line Interface (CLI) on your local machine, allowing you to interact with AWS services from your terminal.

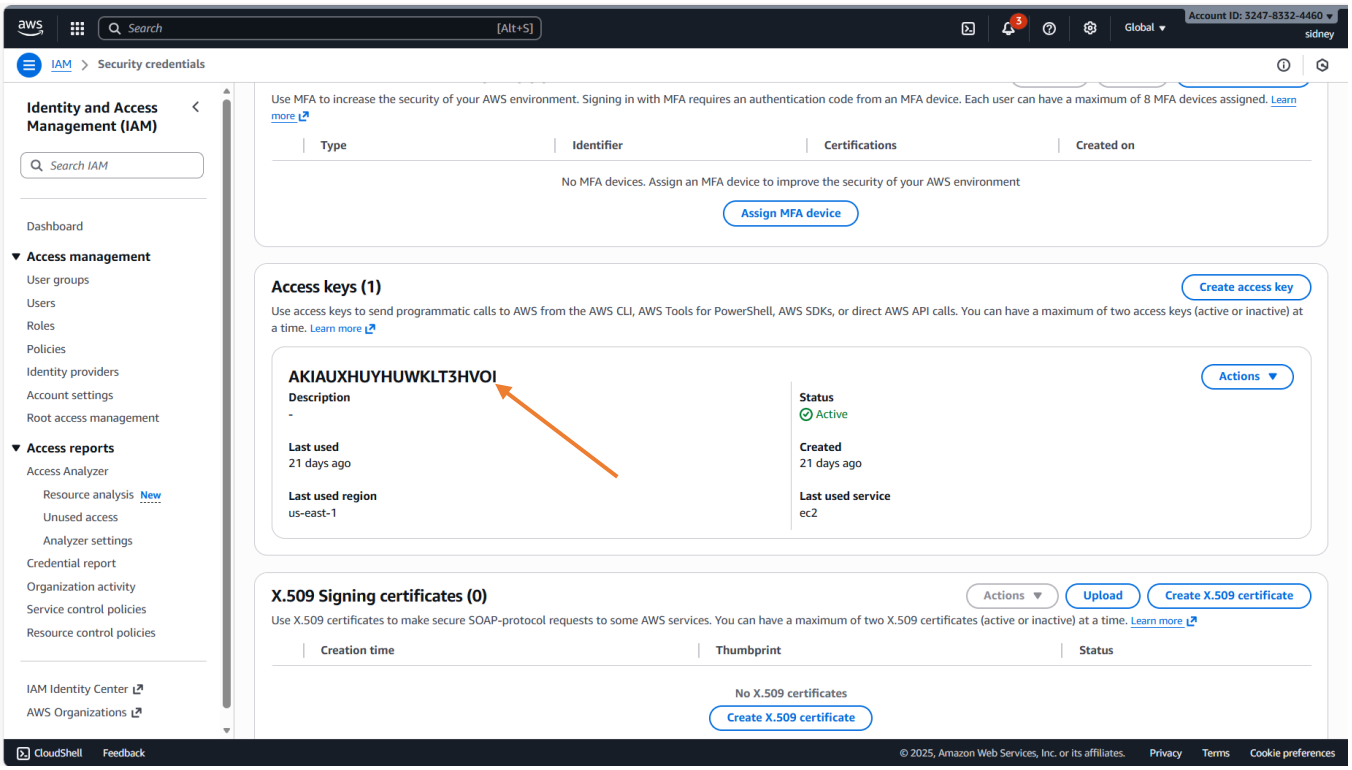
We have to go to our access key



Click on your account name

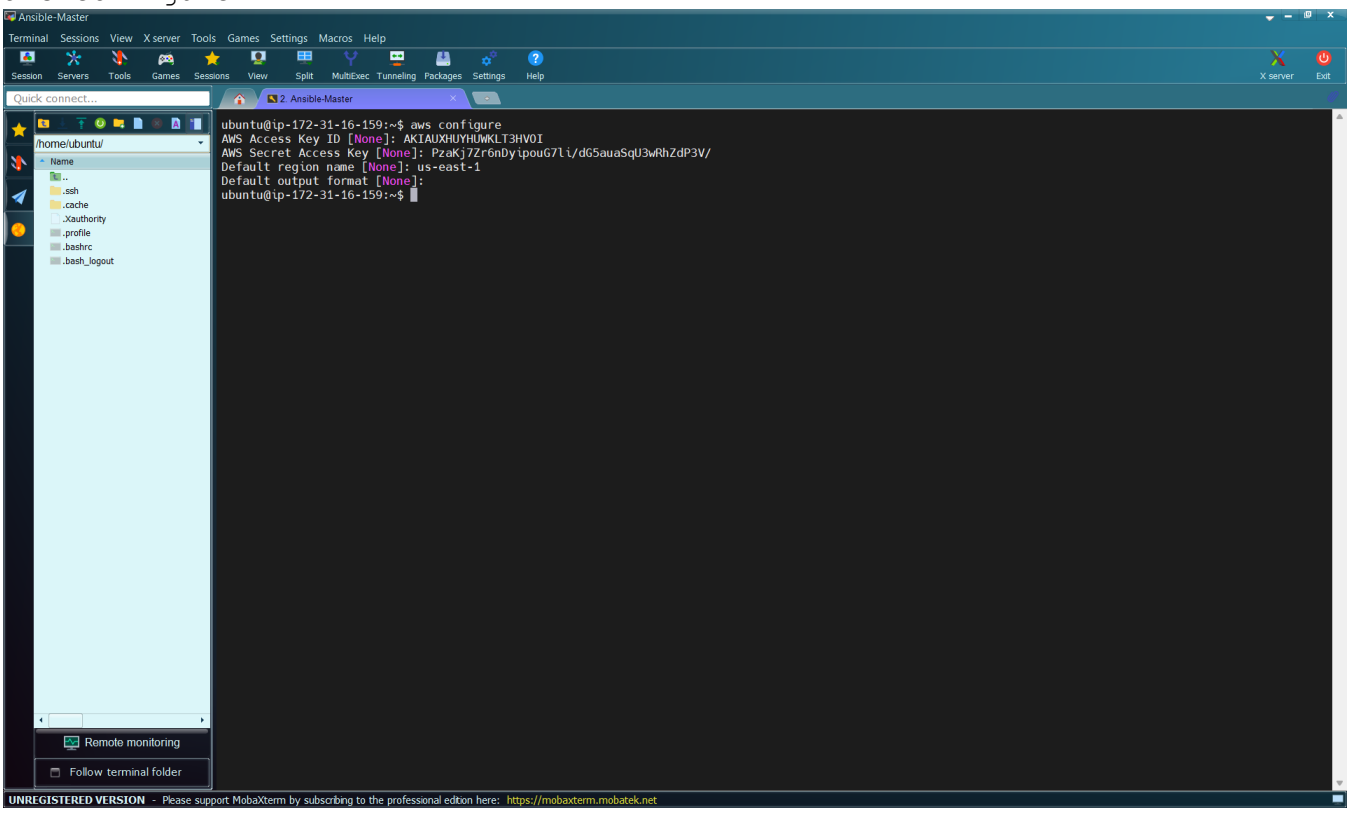


Click on “**Security Credentials**”



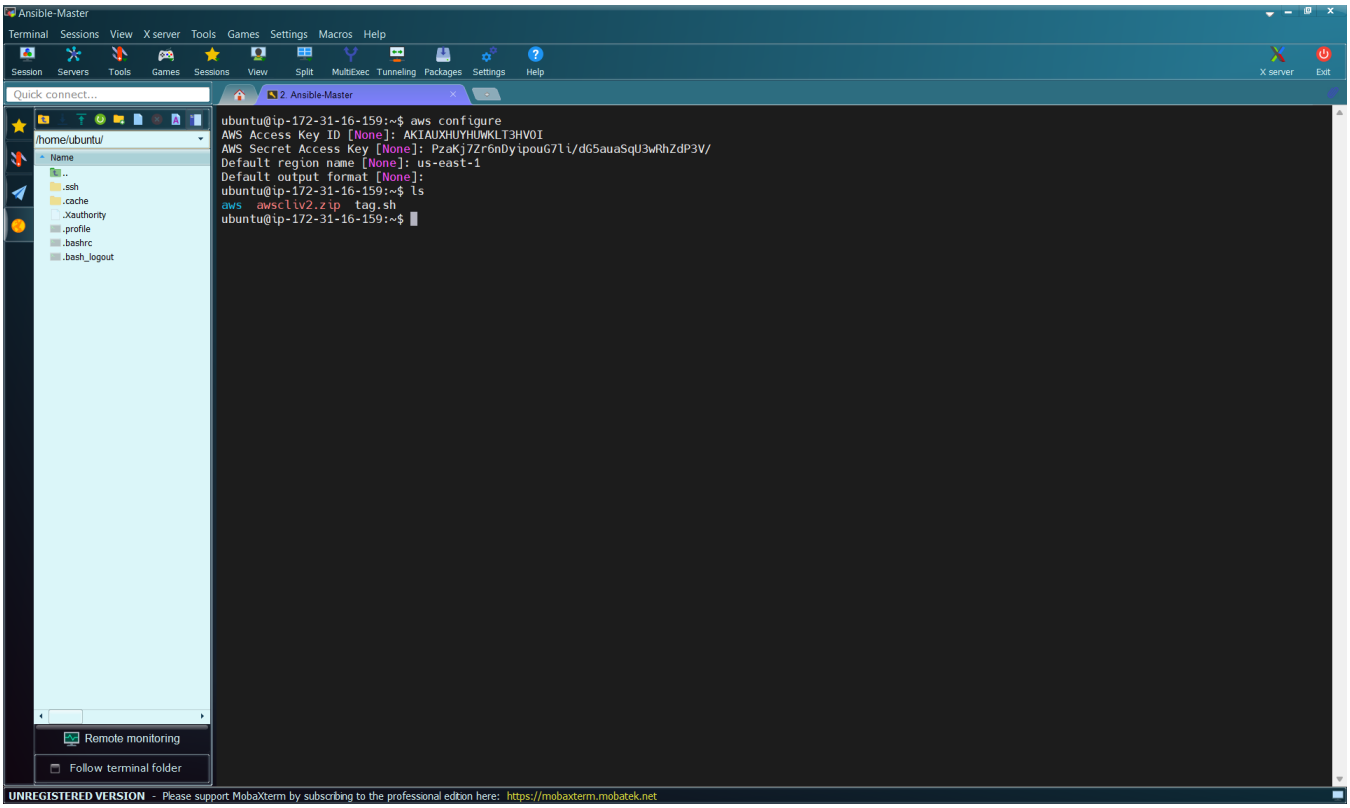
You can see our access key. The file containing these credentials has been save in my Downloads folder. Run the command and enter the credentials:

```
aws configure
```



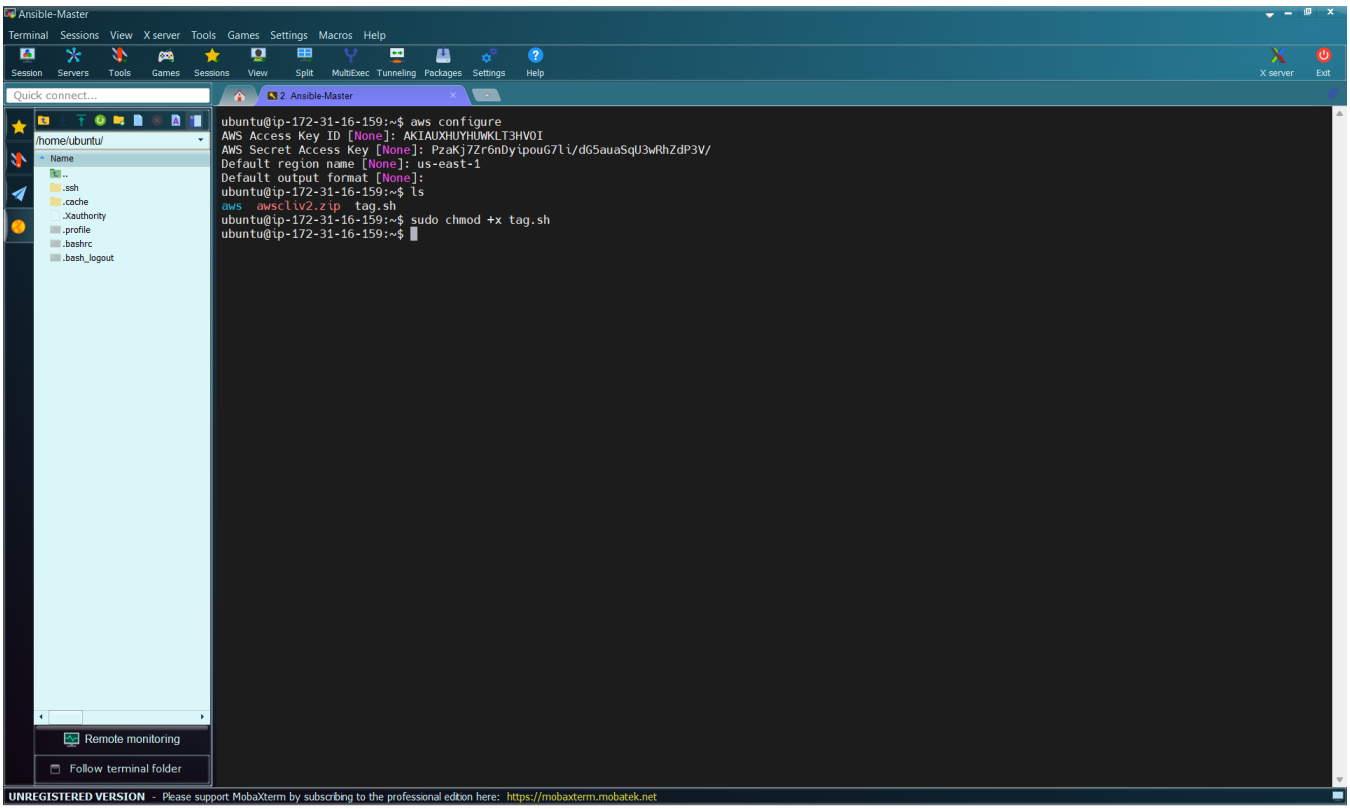
Let is check the content by using the command:

```
ls
```



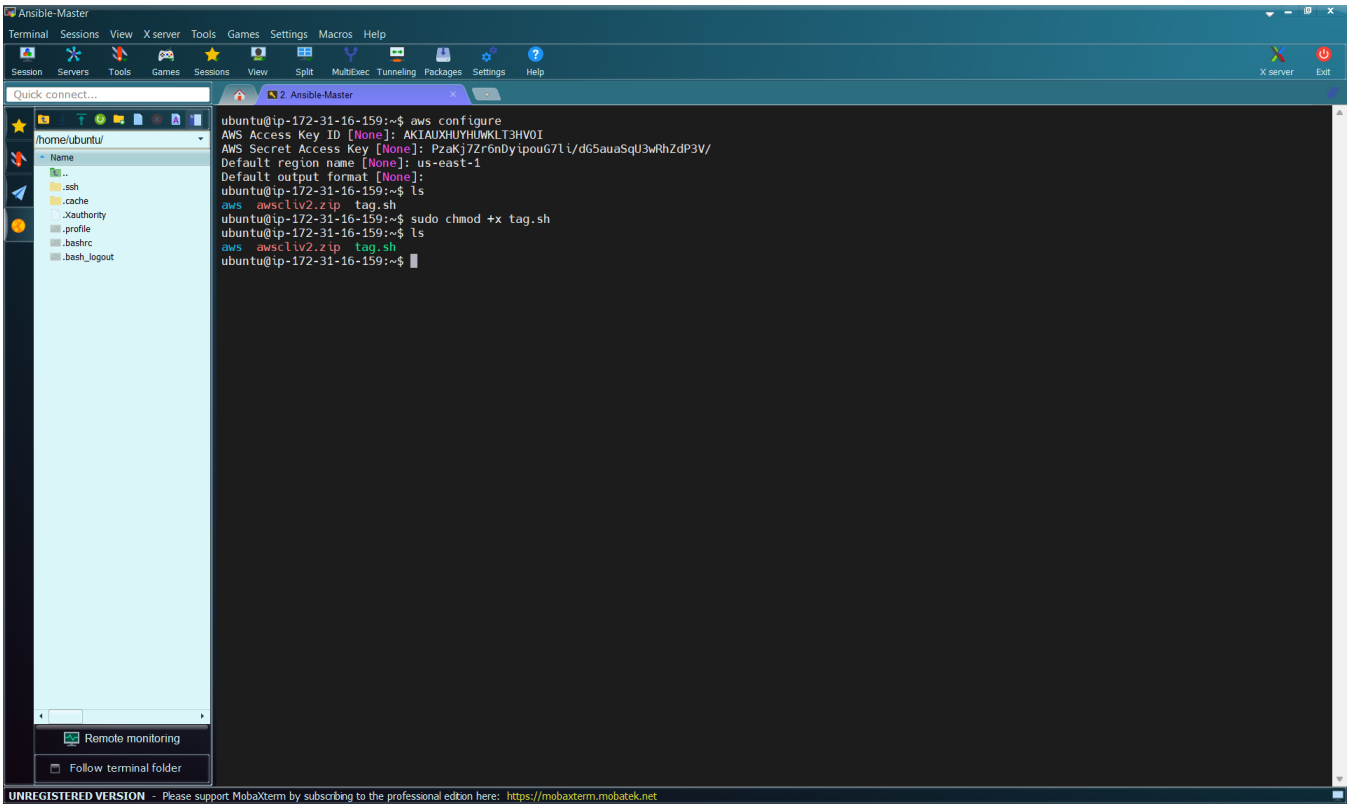
We have to now run the tag.sh script, but we do not have permission to do that. So, we grant permission by running:

```
sudo chmod +x tag.sh
```



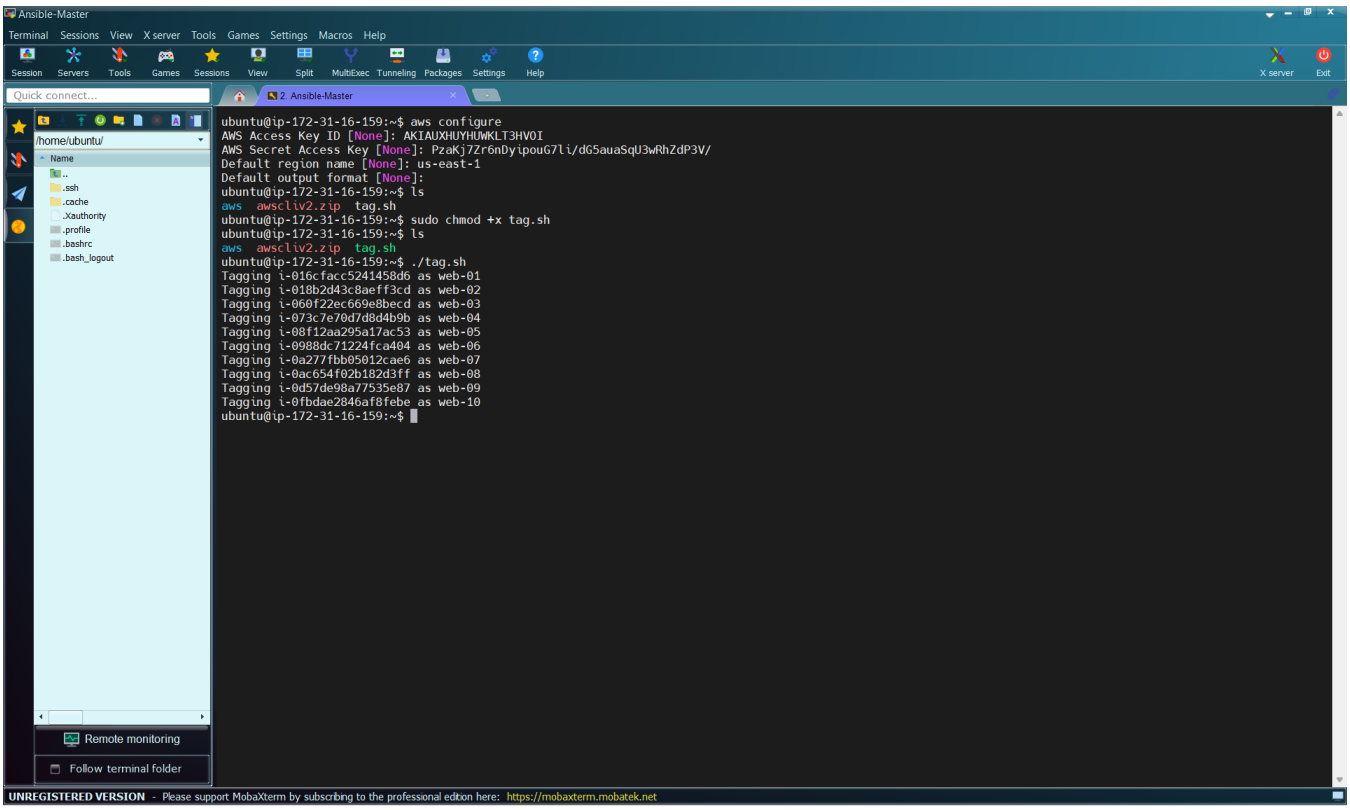
Run the command again

```
ls
```

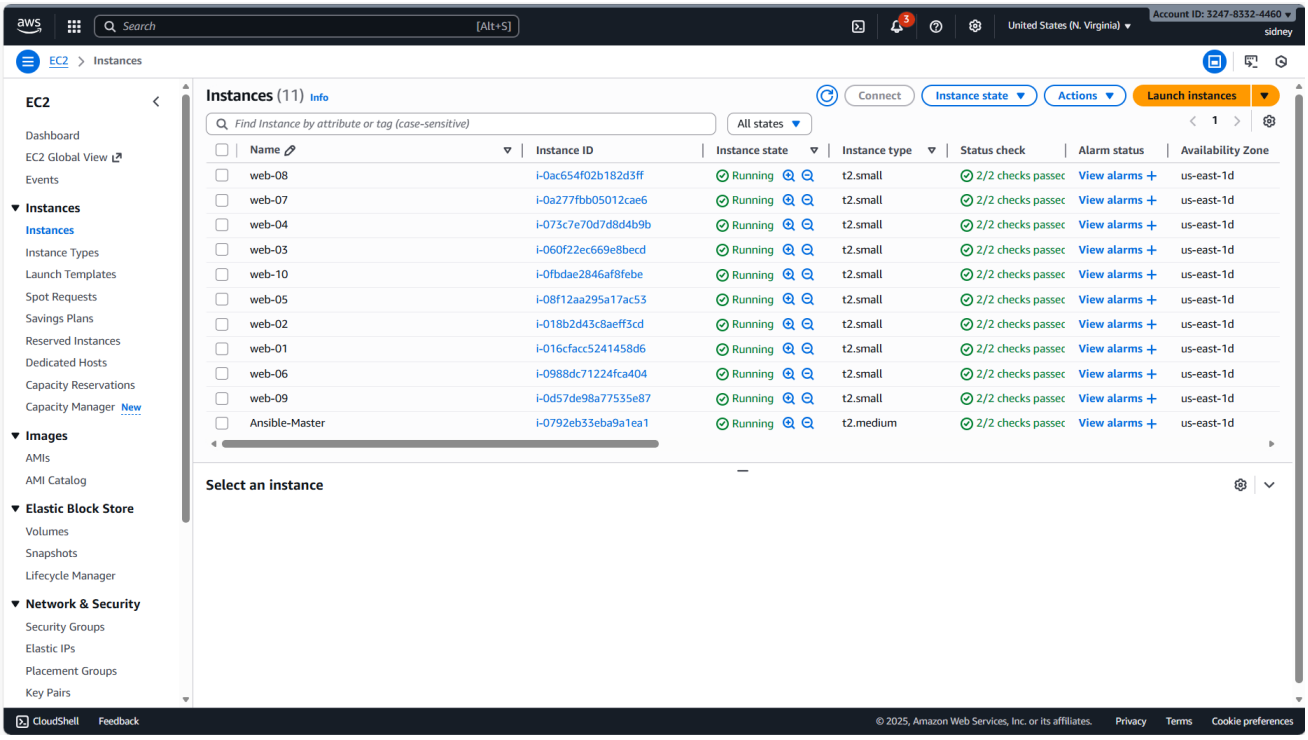


You can see that **tag.sh** is green now. We can now execute the script using the command:

```
./tag.sh
```



You can see that it has rename our target instances. Head back to AWS management console to see if the names have been changed.



You can see that the names have been changed as required.

STEP 2: Establishing communication between Ansible-Master and Target Instances

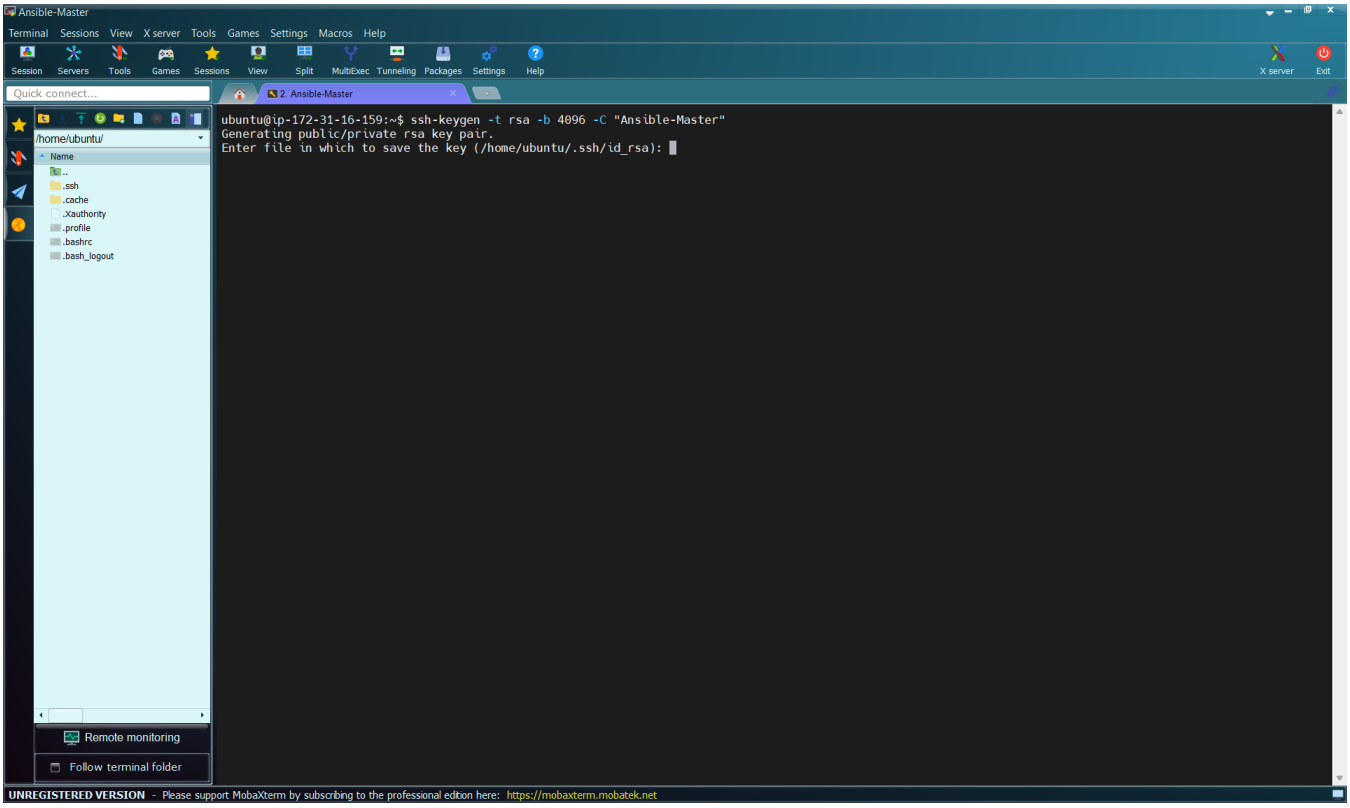
The next thing to do is to establish communication between the Master Instance (Ansible-Master) and the target instances.

For that, we need to make sure we are creating SSH key pair on the Ansible-master and copying the public key from **Ansible-Master** to the target instances.

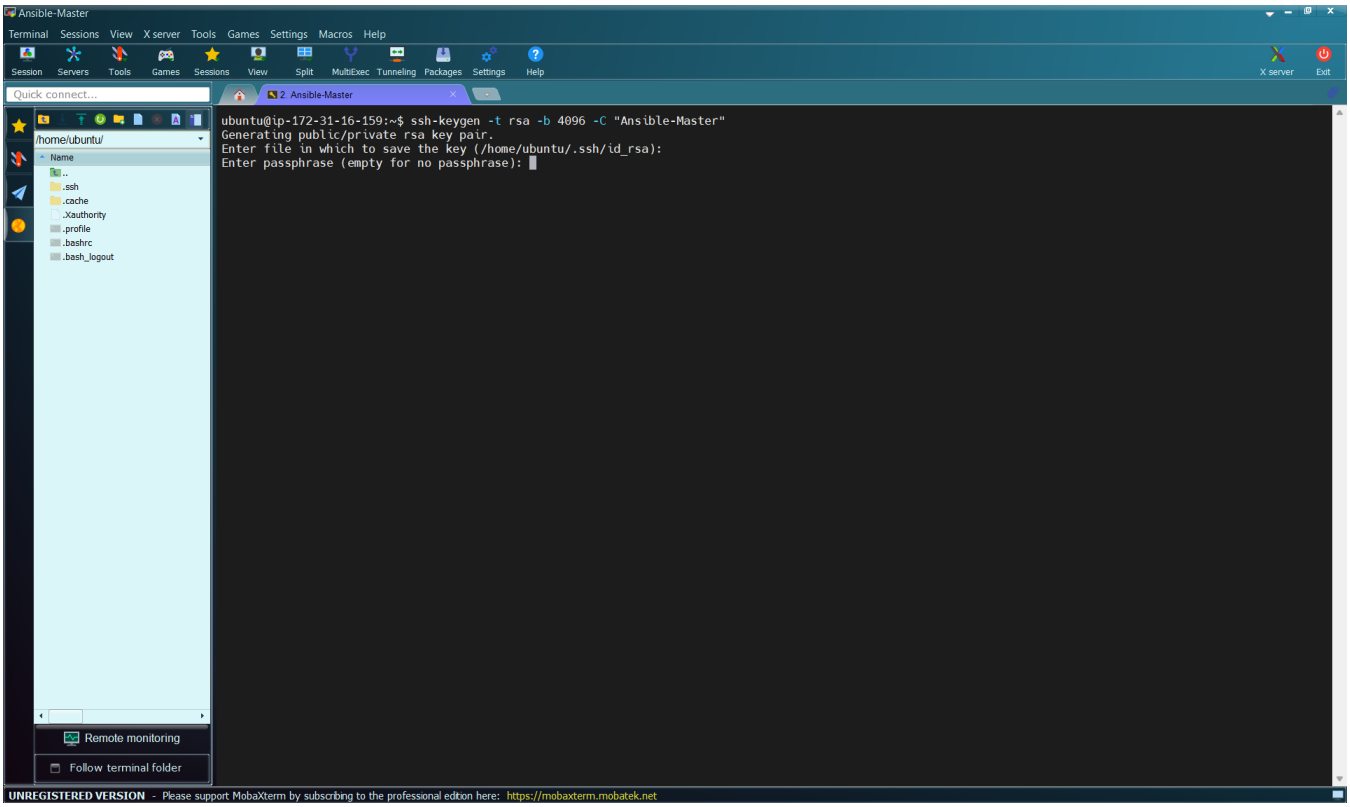
Part 1: Create or Generate Key Pair

To create the SSH Key Pair, we will use the command:

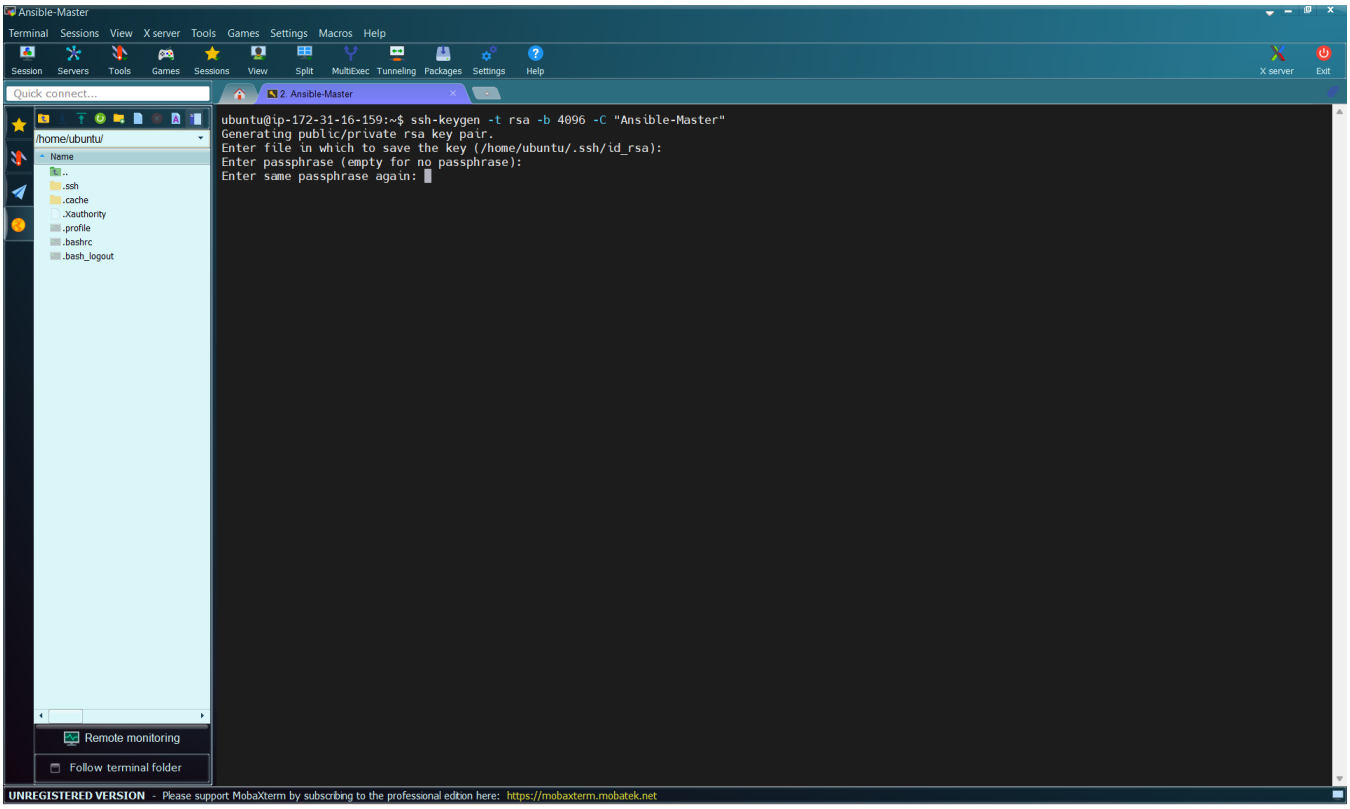
```
ssh-keygen -t rsa -b 4096 -C "Ansible-Master"
```



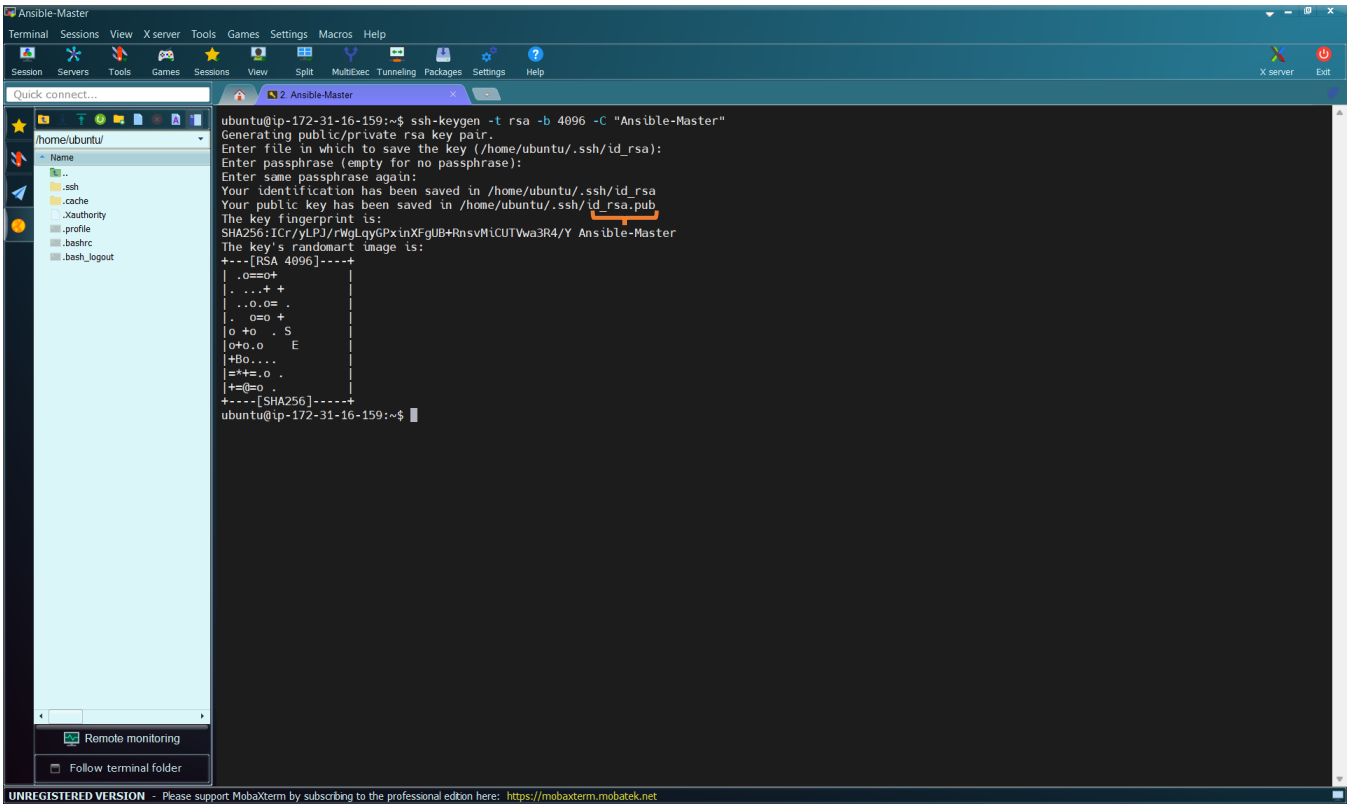
Press "**Enter**"



Press Enter again



And press Enter again



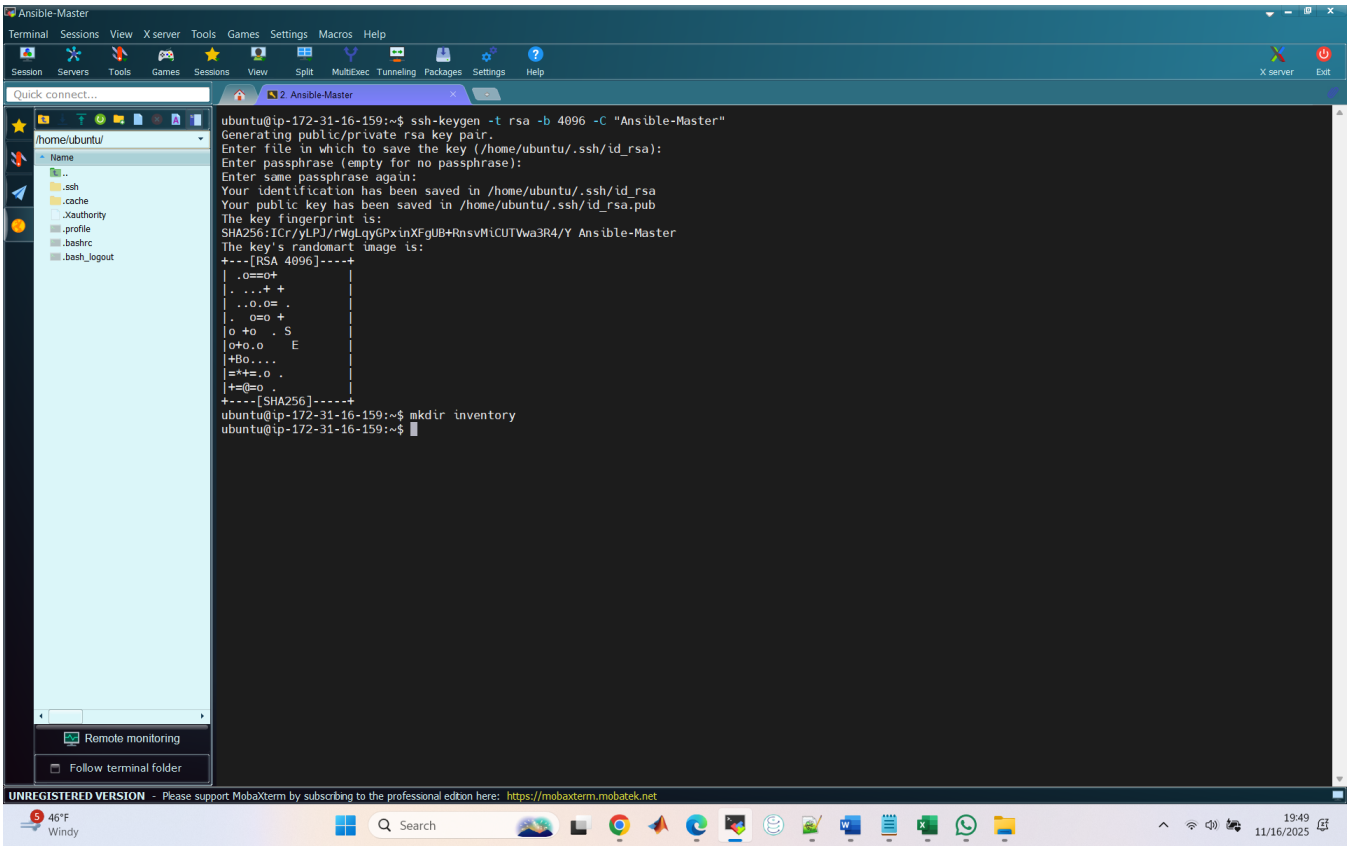
Part 2: Create Dynamic Inventory File

We have to copy the above public key file now using a script. The next thing we want to do how Ansible-Master knows on which machine I have to copy the public key. That means, we need to make sure Ansible-Master has the list of IP addresses of the target instances. How do we get them?

This is where we are going to use a very important concept of using "**dynamic inventory file**". In this concept, you create a yaml file that will fetch the IP addresses of the target instances.

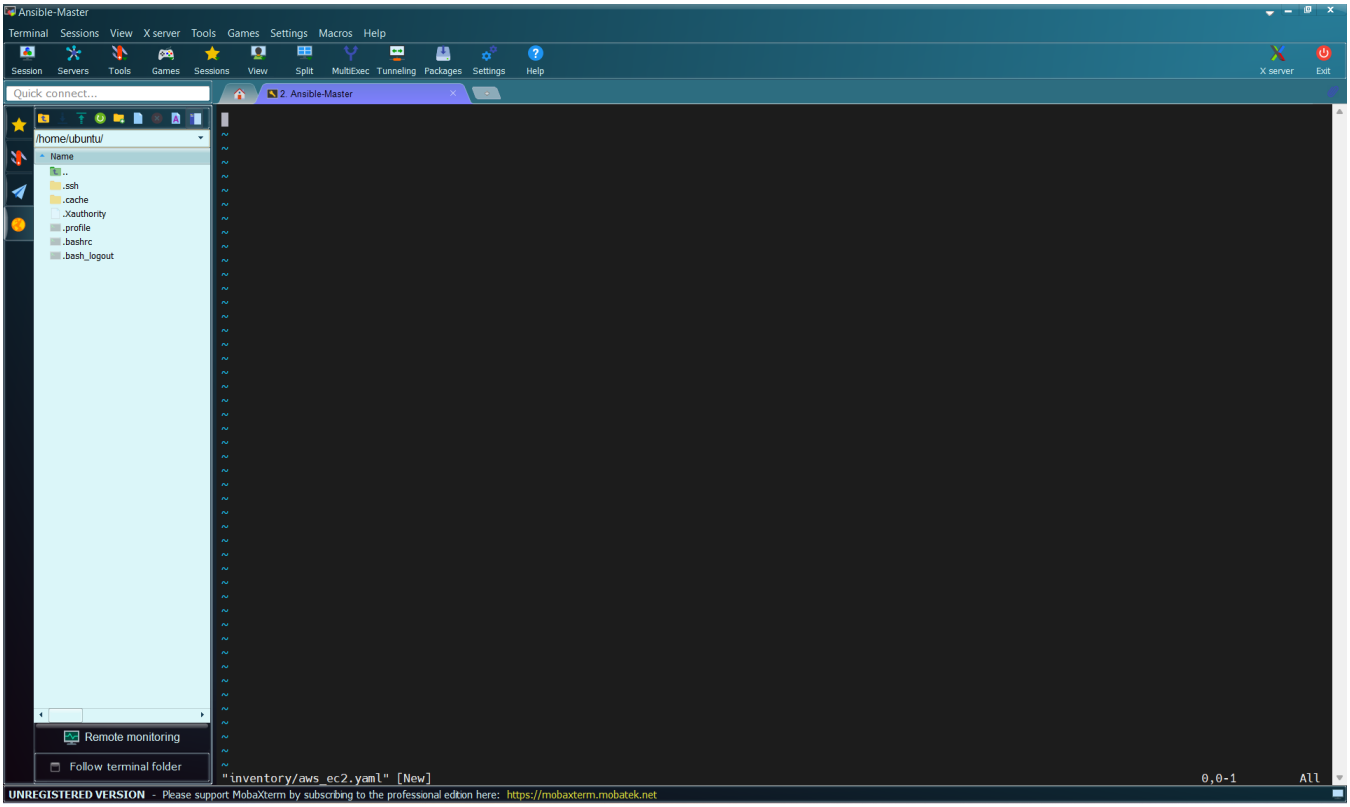
We will create the file in this folder: **inventory/aws_ec2.yaml**. To create the folder, we use the command:

```
mkdir inventory
```



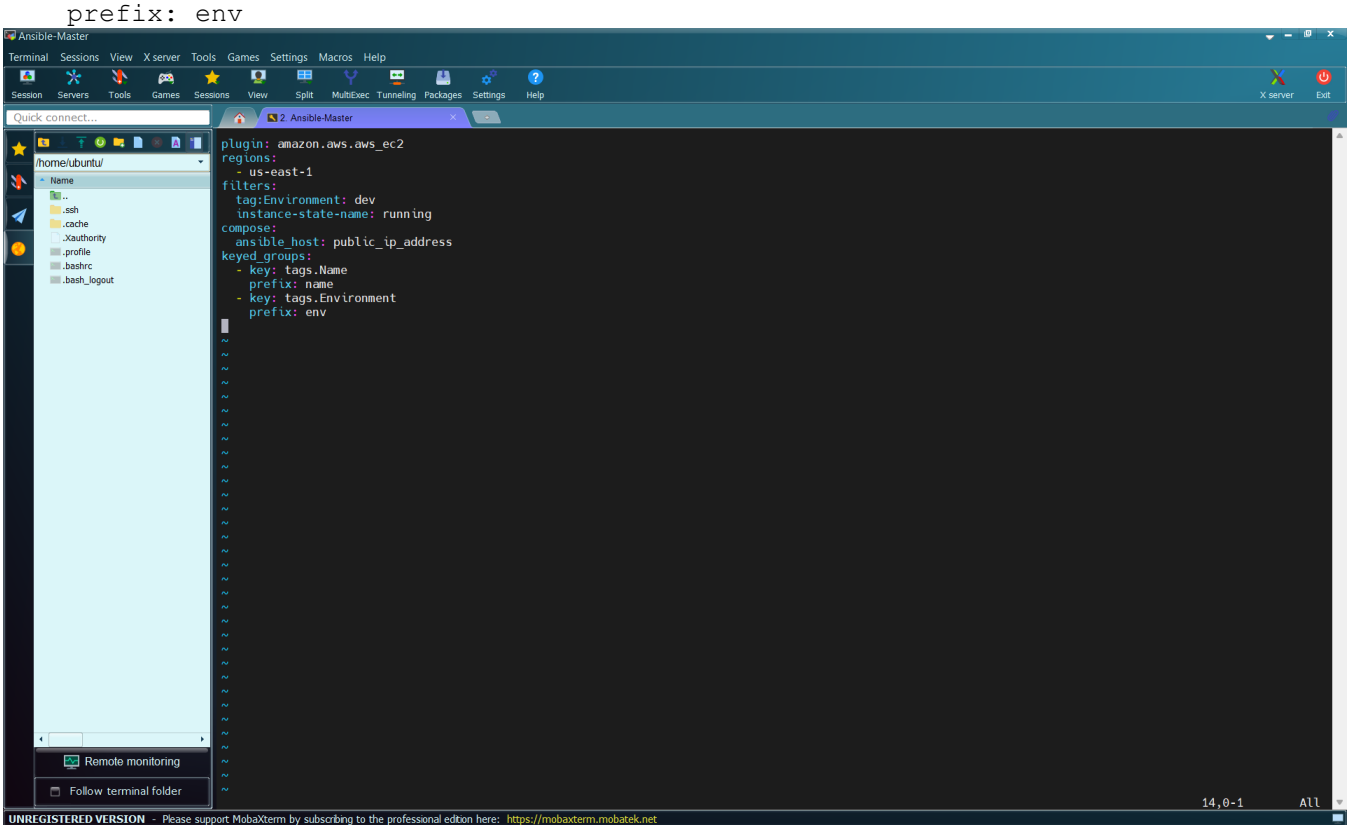
Then we create the yaml file in the folder using the command:

```
vi inventory/aws_ec2.yaml
```

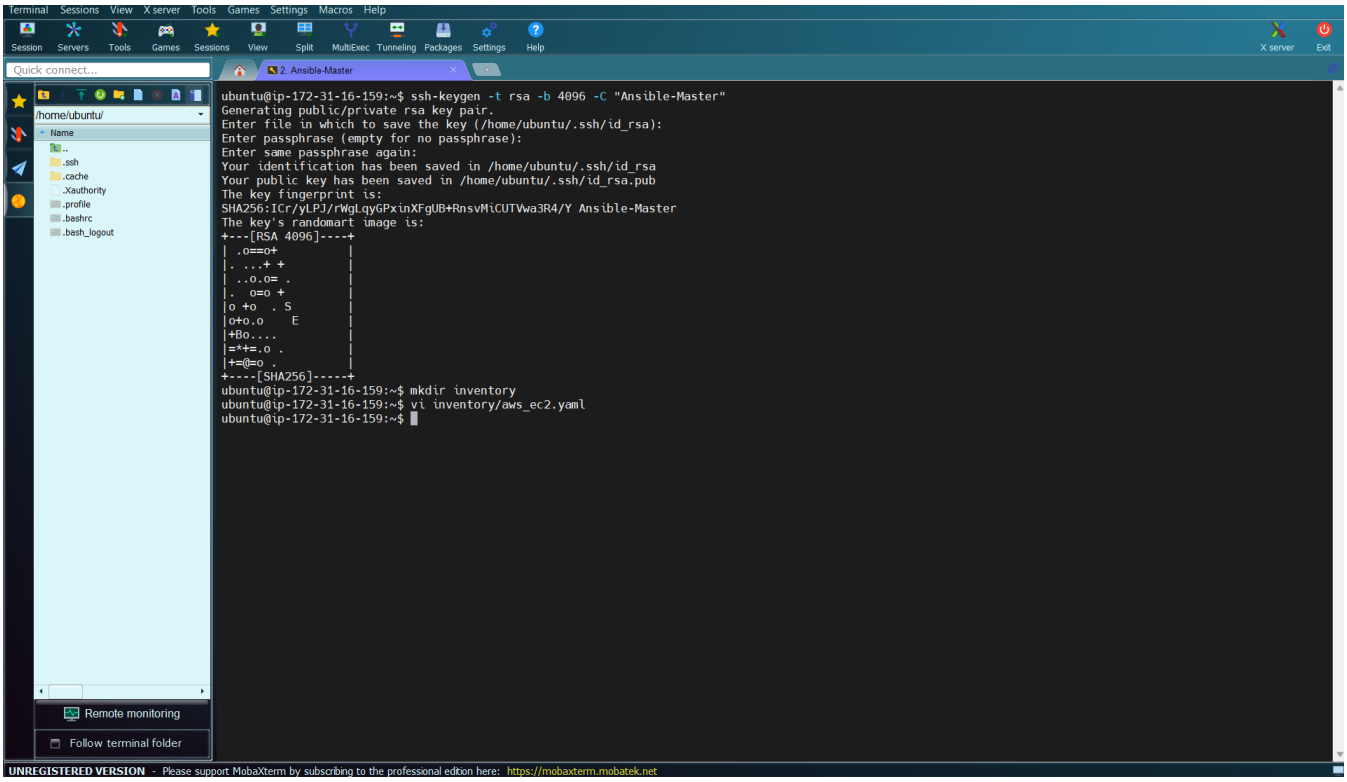


Then we paste the code:

```
plugin: amazon.aws.aws_ec2
regions:
  - us-east-1
filters:
  tag:Environment: dev
  instance-state-name: running
compose:
  ansible_host: public_ip_address
keyed_groups:
  - key: tags.Name
    prefix: name
  - key: tags.Environment
    prefix: env
```



Then save and exit by Pressing “ESC” followed by :wq and press “Enter”



We have created the yaml file

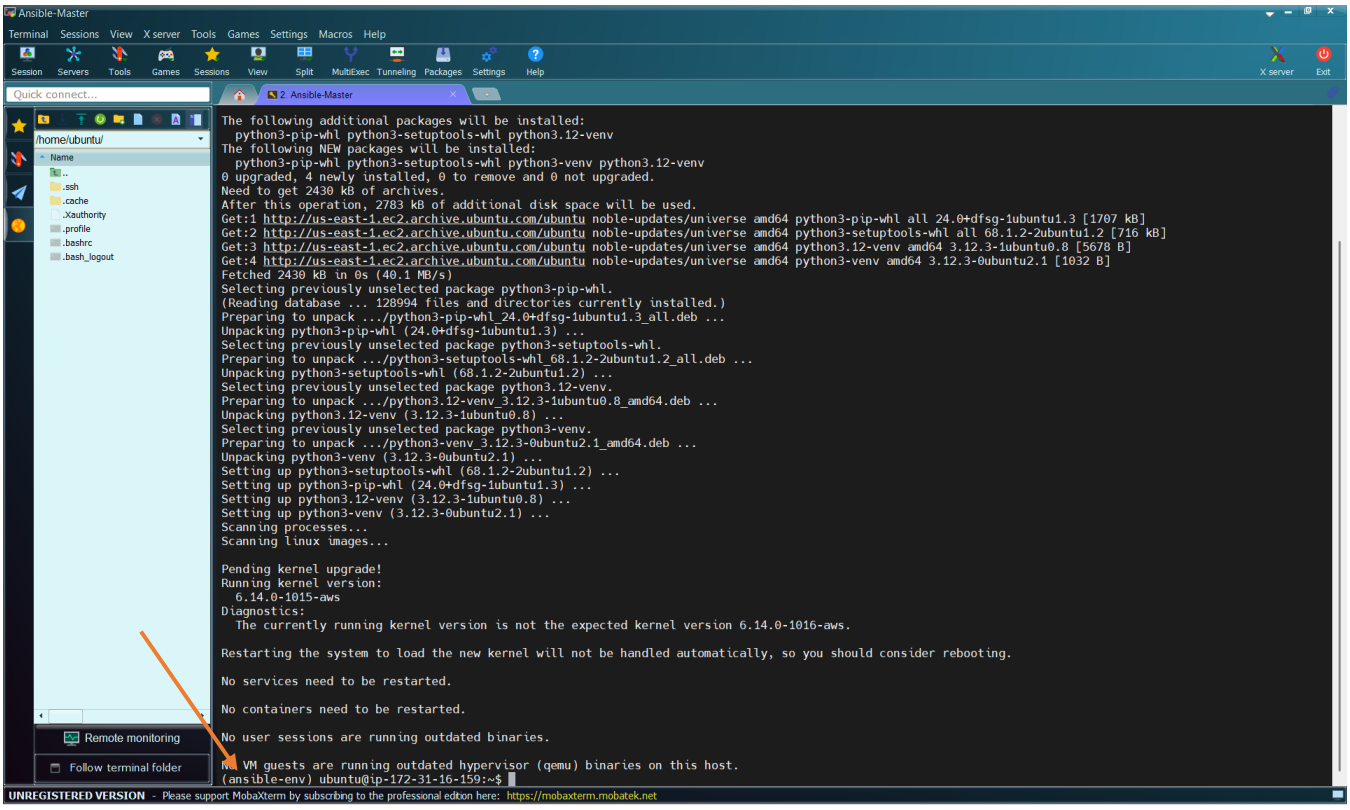
STEP 4: Install and Enable the Virtual Environment

The next thing we have to do is basically create a Virtual Environment for Ansible. We want to isolated the Ansible related dependencies by separating them from the environment.

Part 1: Create Virtual Environment

For that we will use a Python command to create our Virtual environment. For this, we will run the code which is going to install and enable the Virtual environment:

```
# Step 1: Install venv module if not already present
sudo apt install python3-venv -y

# Step 2: Create a virtual environment
python3 -m venv ansible-env

# Step 3: Activate it
source ansible-env/bin/activate
```

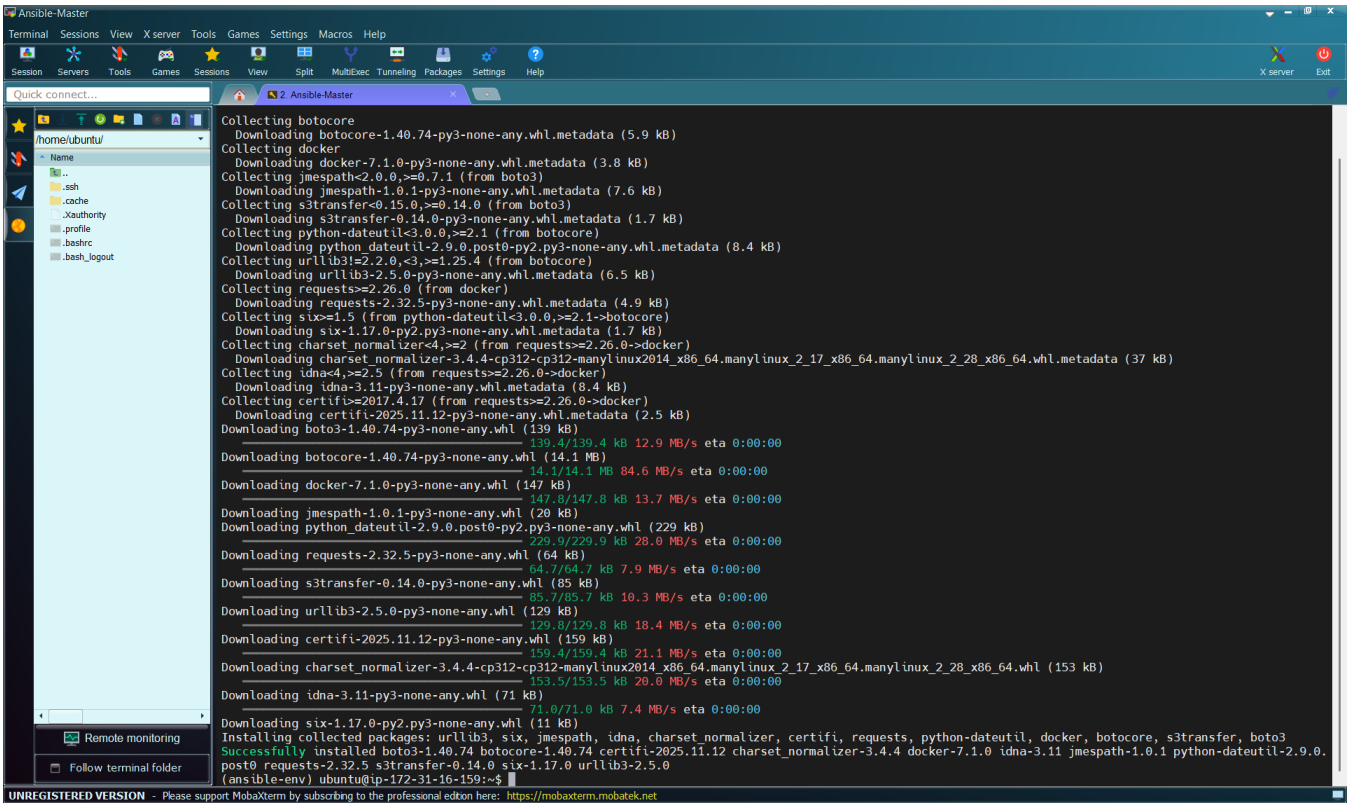


You can see that our virtual environment has been created.

Part 2: Install Python Dependencies

The next thing is to install Python dependencies using the command:

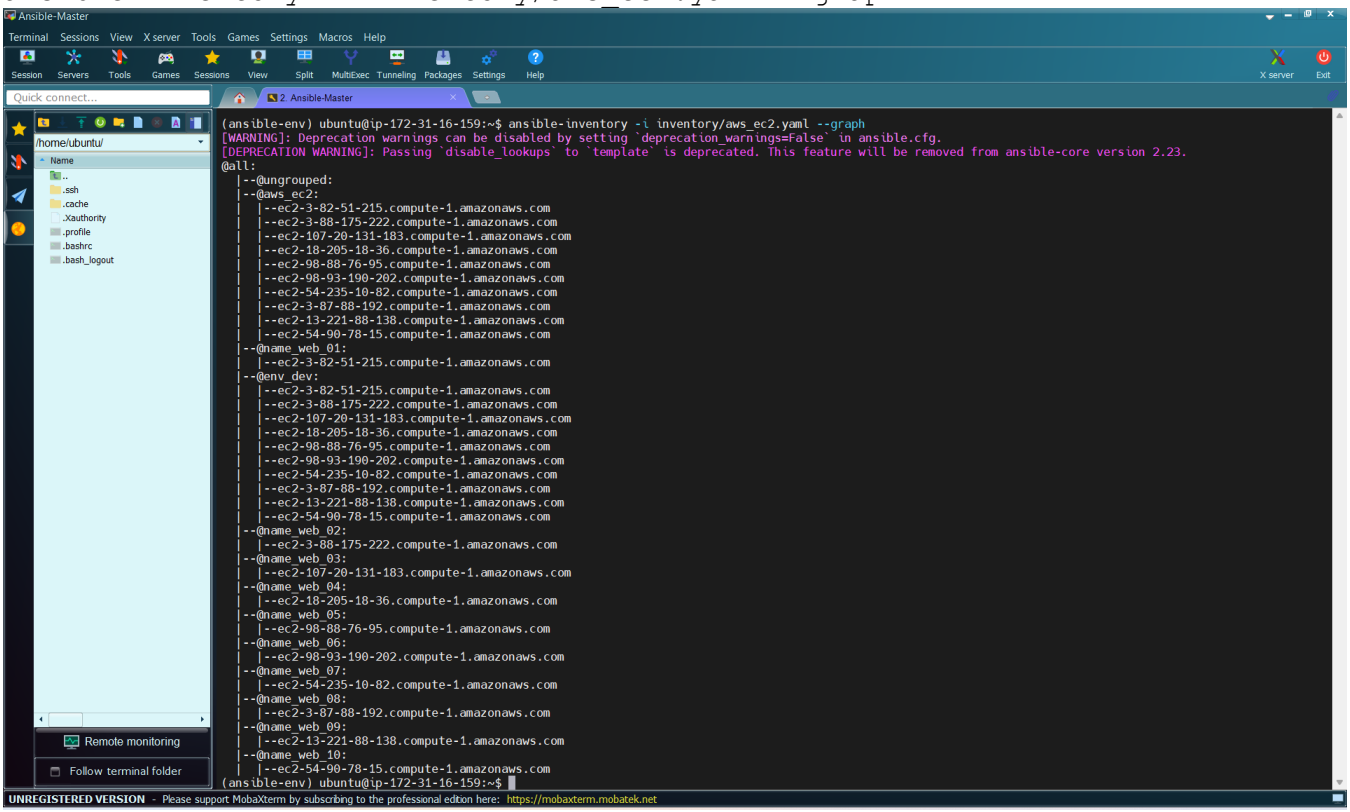
```
# Step 4: Install required Python packages
pip install boto3 botocore docker
```



Part 3: Get the IP address

We can now get the IP addresses of the Target instances by running the command:

```
ansible-inventory -i inventory/aws_ec2.yaml --graph
```

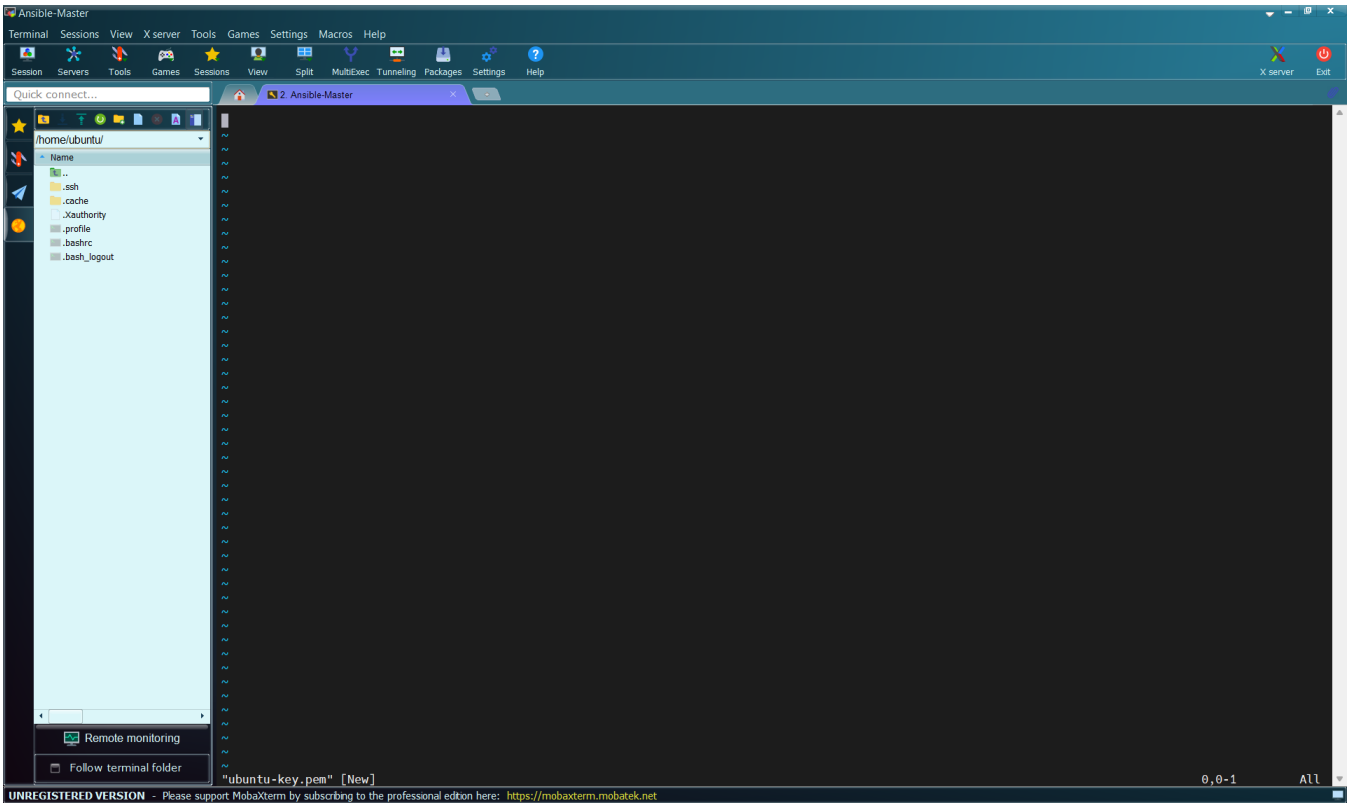


We now have all the IP addresses.

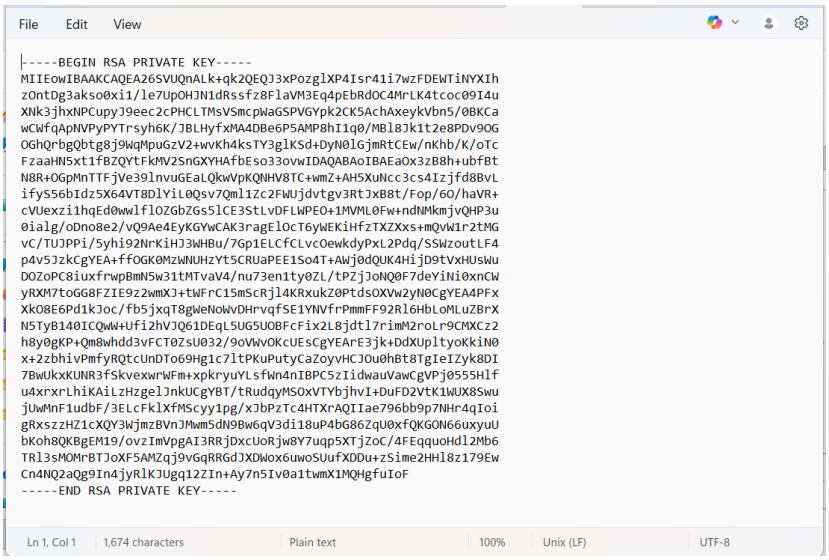
Part 4: Copy the Private Key

The next thing is to copy the private key. For this we will use a script. We will give the script the name of our **.pem** file. That is "**ubuntu-key.pem**". We create the file using the command:

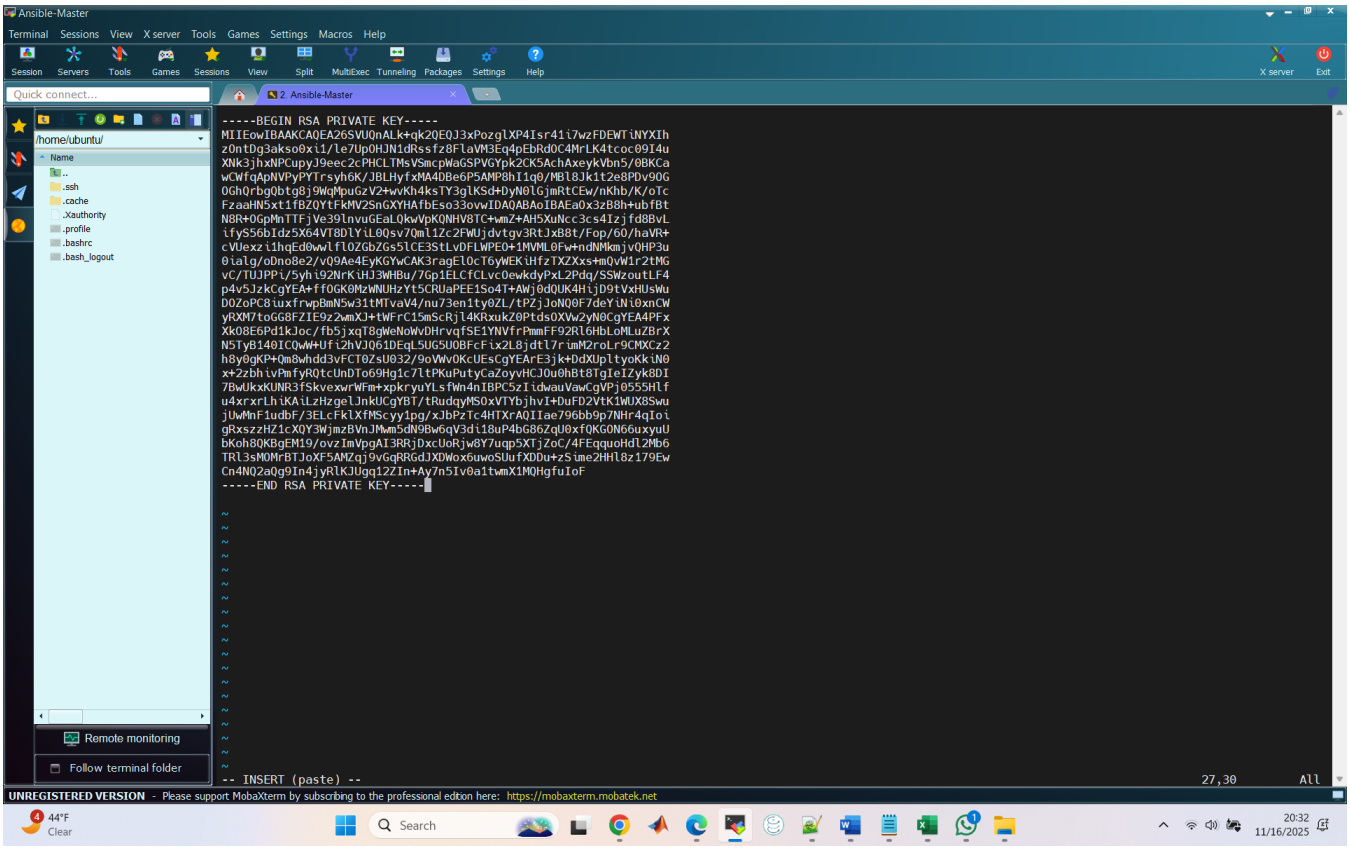
```
vi ubuntu-key.pem
```



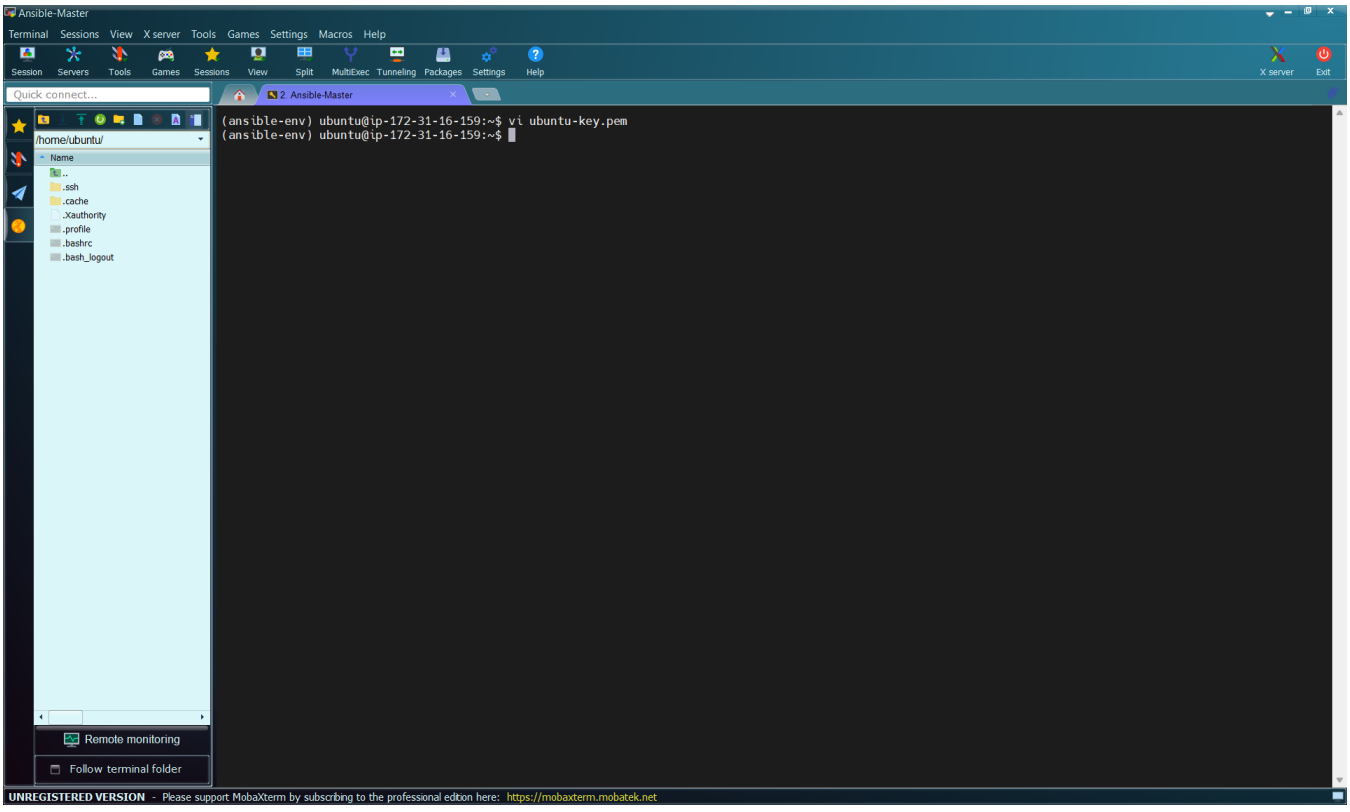
Go to our **.pem** file and open it in Notepad



Copy the code in this file and paste

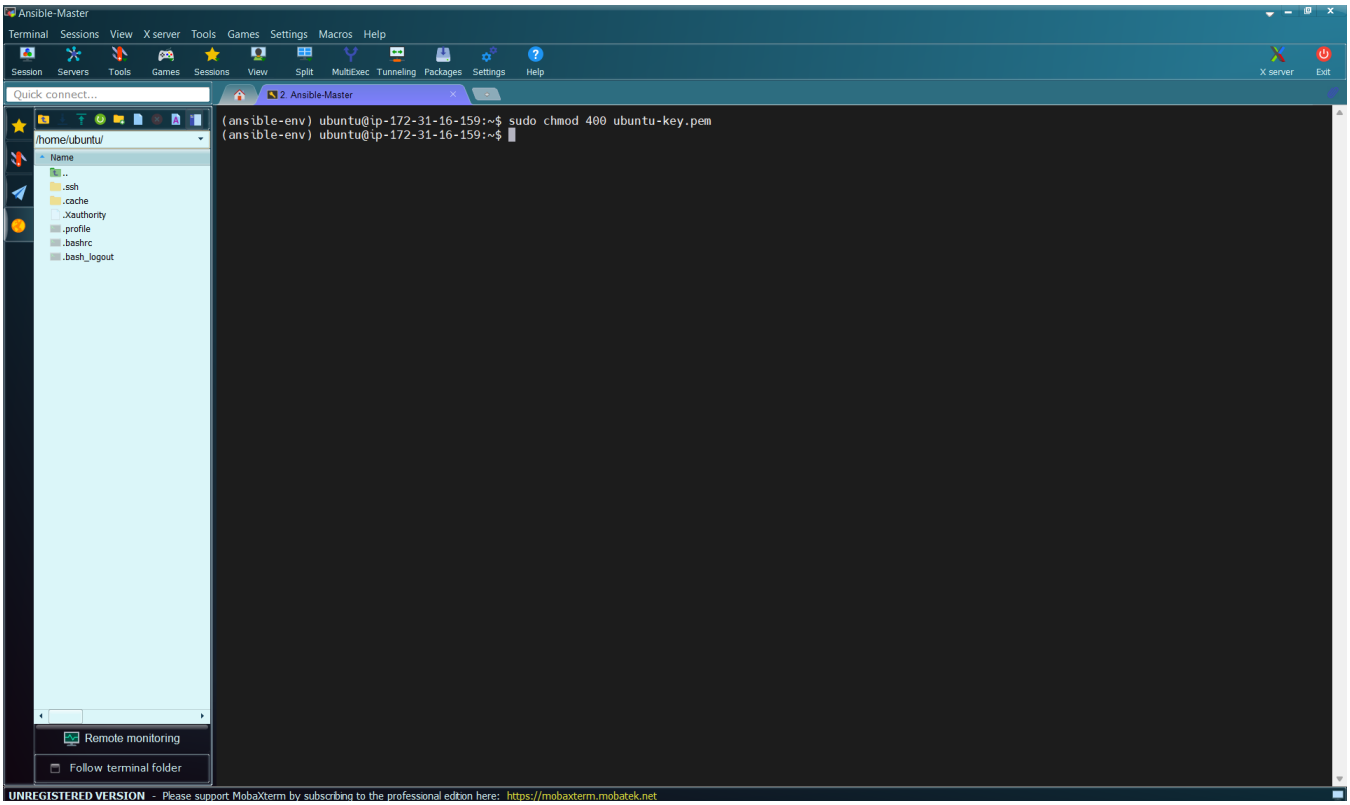


Save and Exit the code by pressing ESC followed by :wq and Press “Enter”



Linux is quite strict about permissions on the SSH key pair. So, we need to set up permission for this file as well. We will grant permission for read only using the command:

```
sudo chmod 400 ubuntu-key.pem
```

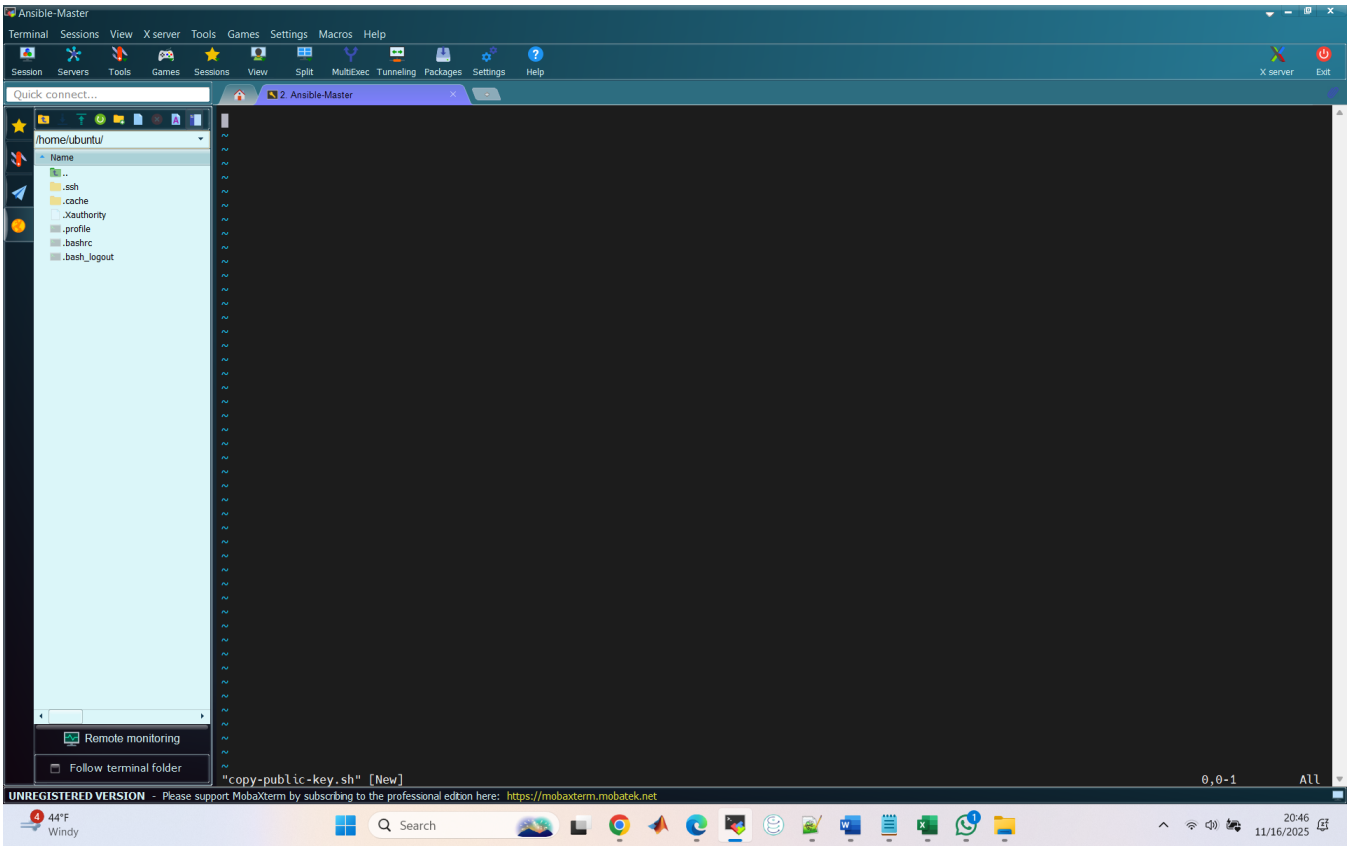


Part 5: Copy the Public Key

We have to create the script that is going to copy the public key of the Ansible-Master on the Target instances. We will call the script "**copy-public-key.sh**"

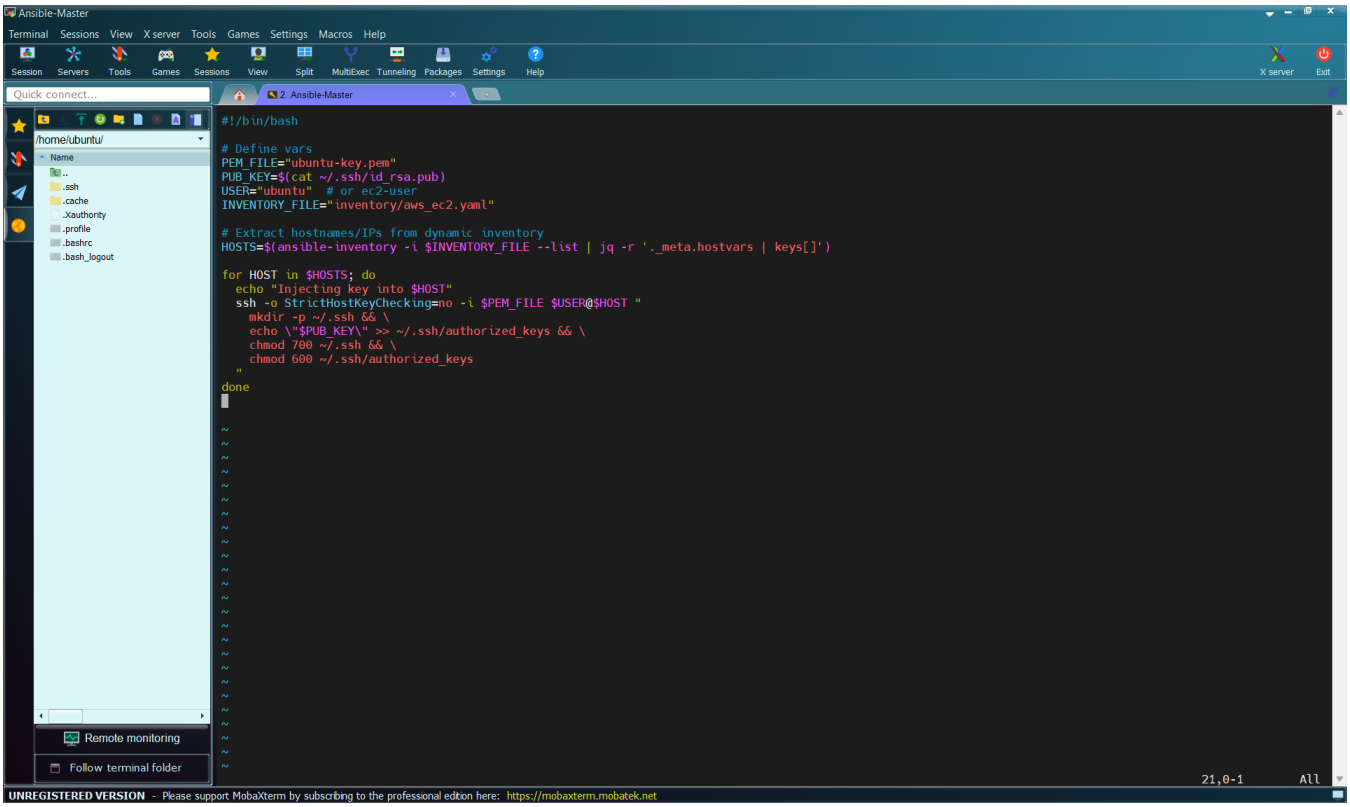
We create the script using the command:

```
vi copy-public-key.sh
```

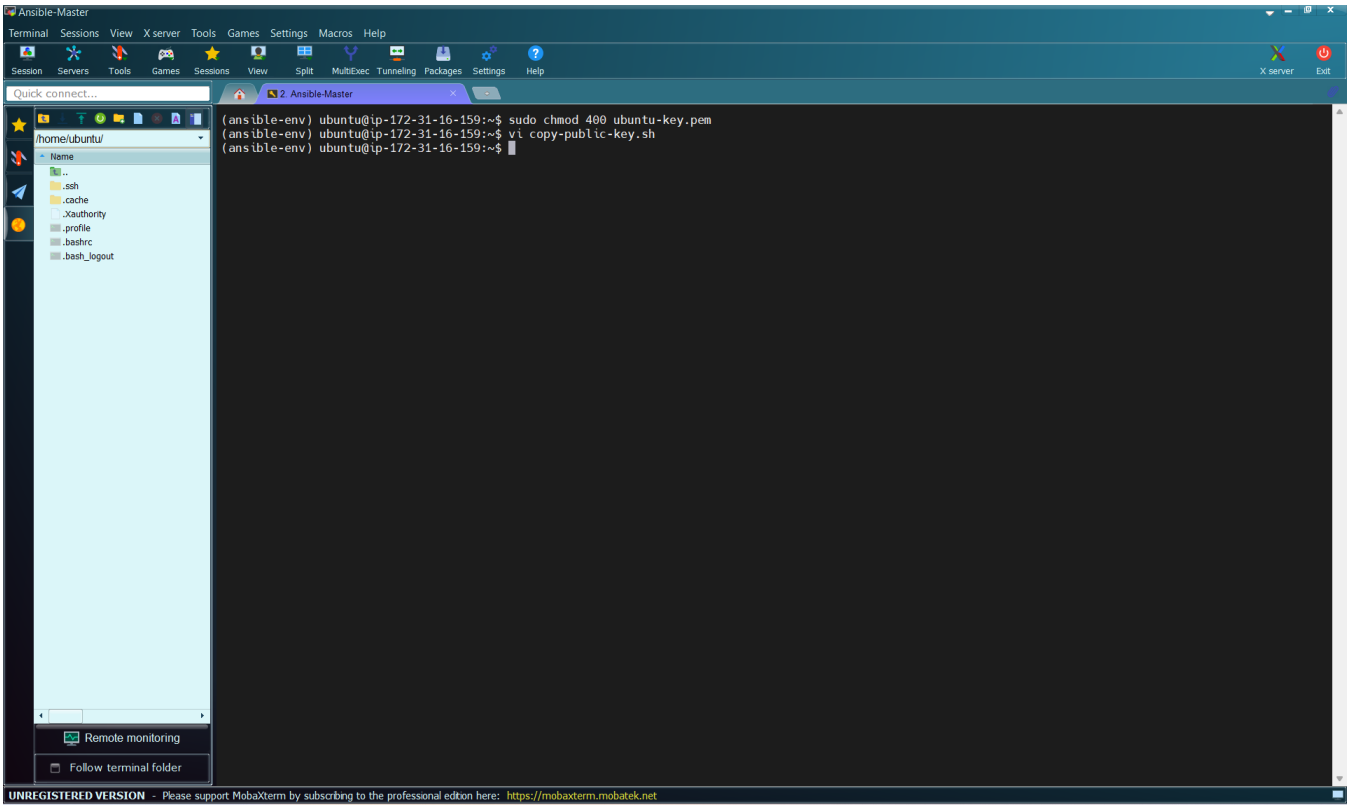


Then, paste the code

```
#!/bin/bash

# Define vars
PEM_FILE="ubuntu-key.pem"
PUB_KEY=$(cat ~/.ssh/id_rsa.pub)
USER="ubuntu"  # or ec2-user
INVENTORY_FILE="inventory/aws_ec2.yaml"

# Extract hostnames/IPs from dynamic inventory
HOSTS=$(ansible-inventory -i $INVENTORY_FILE --list | jq -r '._meta.hostvars | keys[]')

for HOST in $HOSTS; do
  echo "Injecting key into $HOST"
  ssh -o StrictHostKeyChecking=no -i $PEM_FILE $USER@$HOST "
    mkdir -p ~/.ssh && \
    echo \"$PUB_KEY\" >> ~/.ssh/authorized_keys && \
    chmod 700 ~/.ssh && \
    chmod 600 ~/.ssh/authorized_keys
  "
done
```



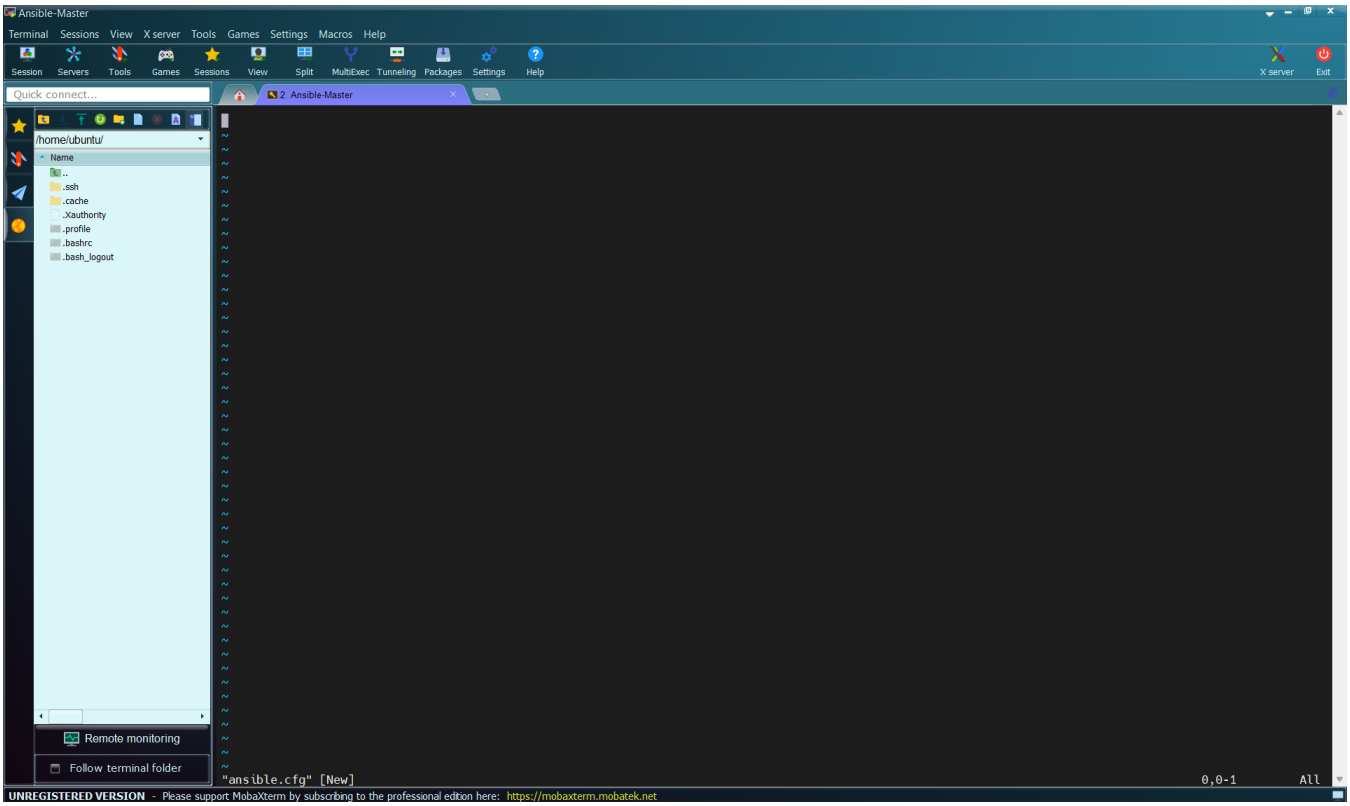
Then, save and exit by pressing ESC followed by :wq and Press “Enter”



NOTE: sometimes when you run a script, there is a prompt that as you to type yes or no. We do not need that in this case, so to avoid that, we will use a script. We will call the script “**ansible.cfg**”

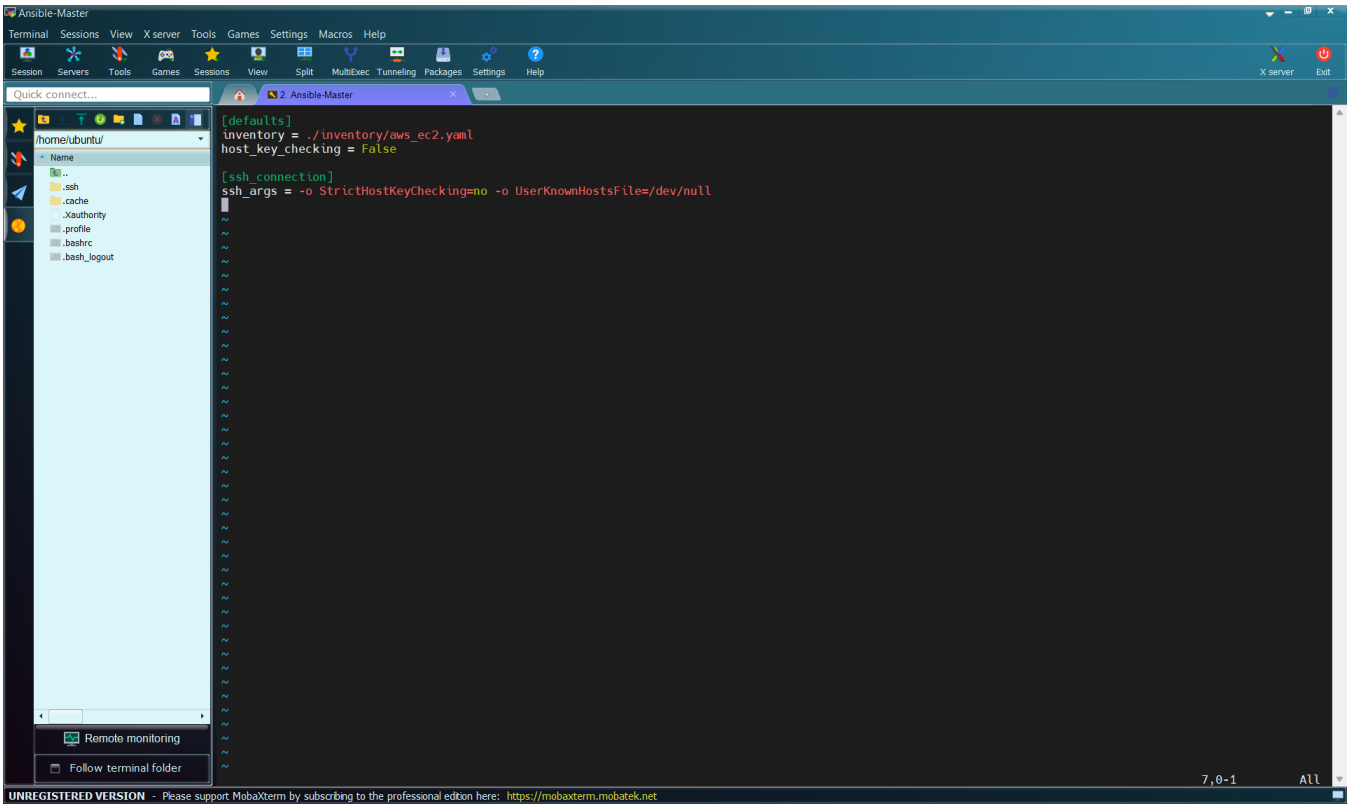
Let us create the file using the command:

```
vi ansible.cfg
```

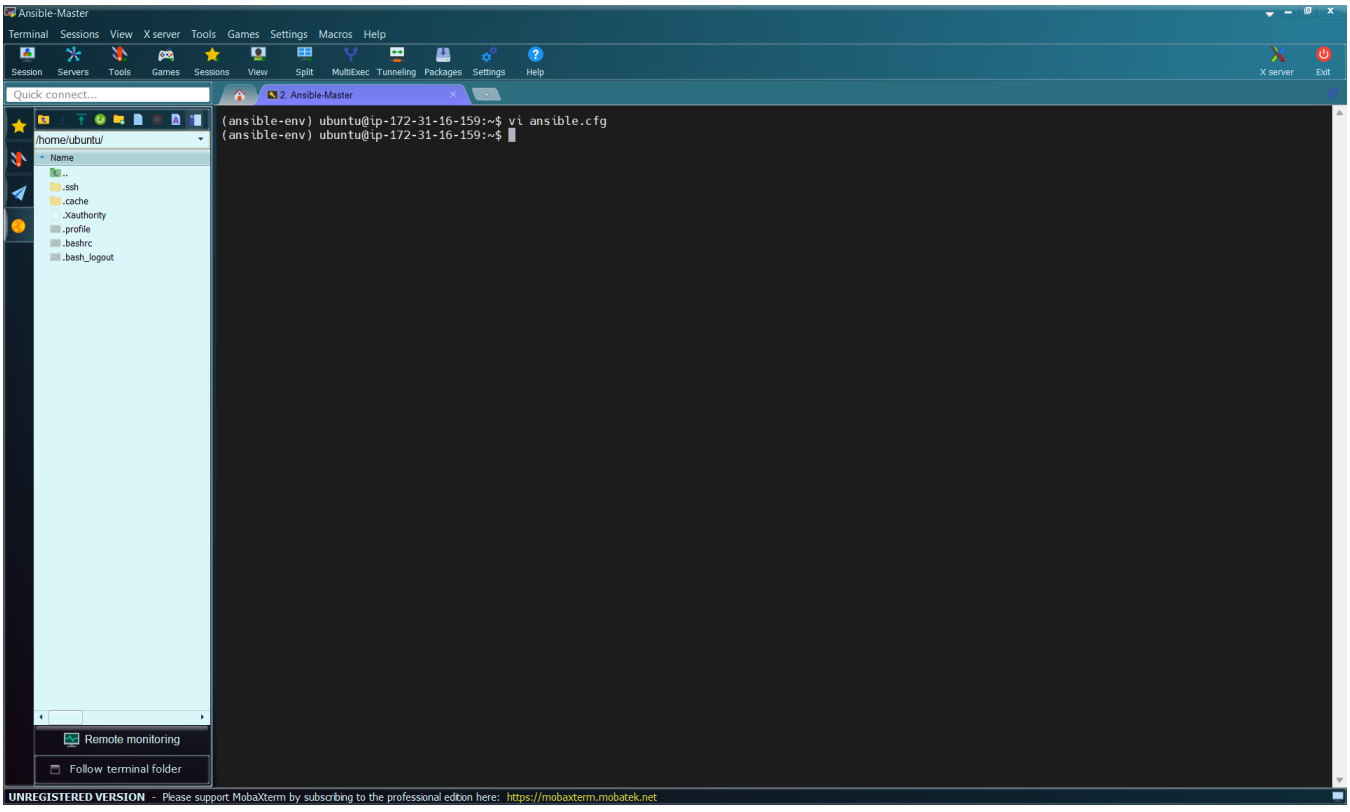


Then, paste the code:

```
[defaults]
inventory = ./inventory/aws_ec2.yaml
host_key_checking = False

[ssh_connection]
ssh_args = -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null
```

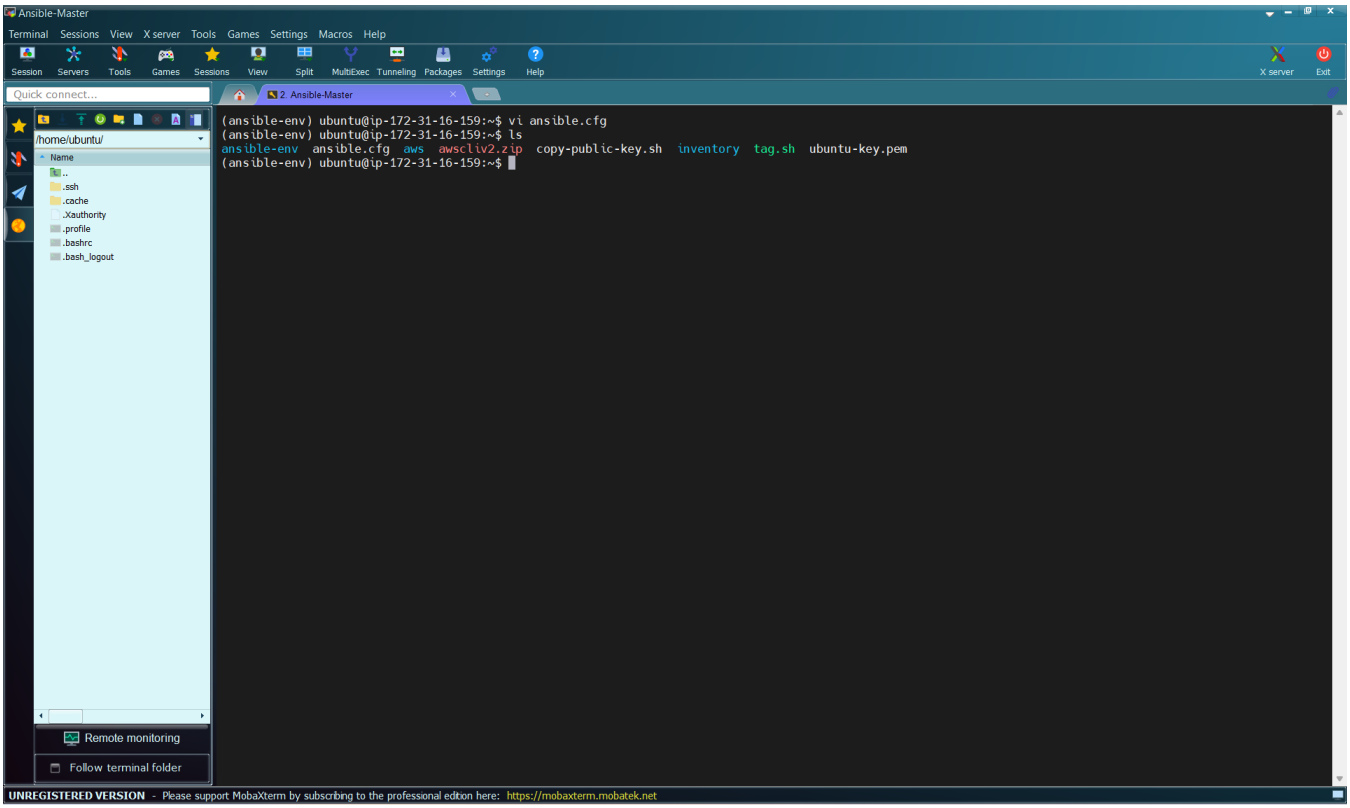


Then save and exit by pressing ESC followed by :wq and press “enter”



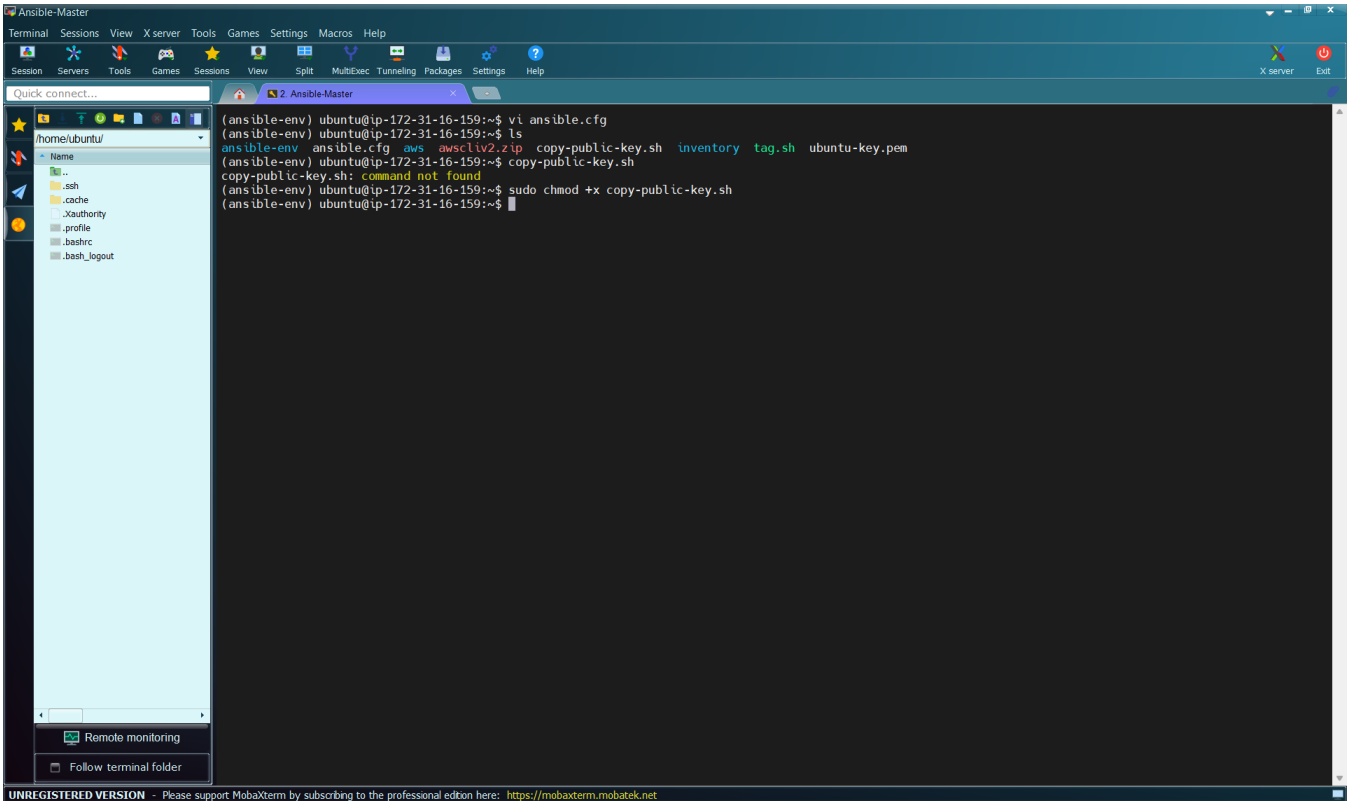
Check the content using the command:

```
ls
```



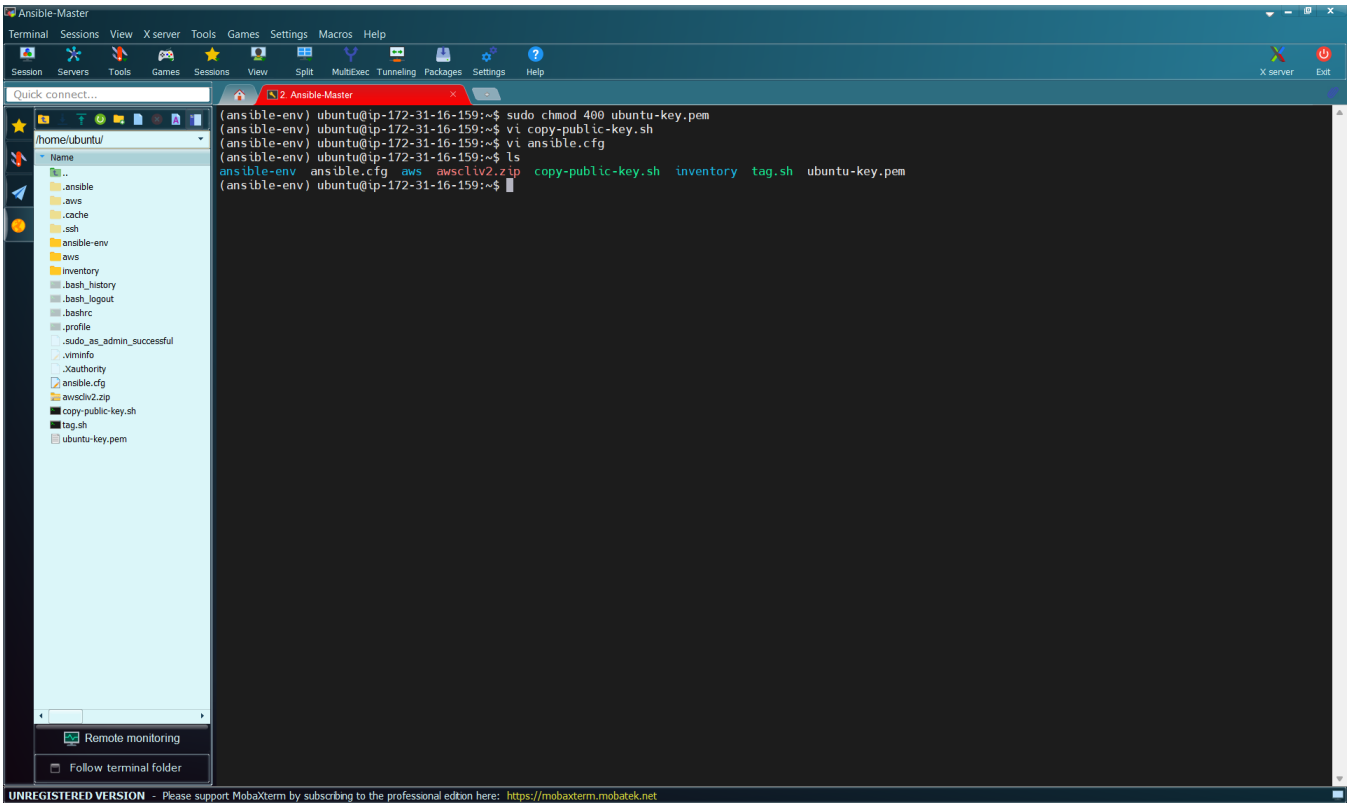
You can see that the file "**copy-public-key.sh**" is not executable. We have to make it executable by using the command:

```
sudo chmod +x copy-public-key.sh
```



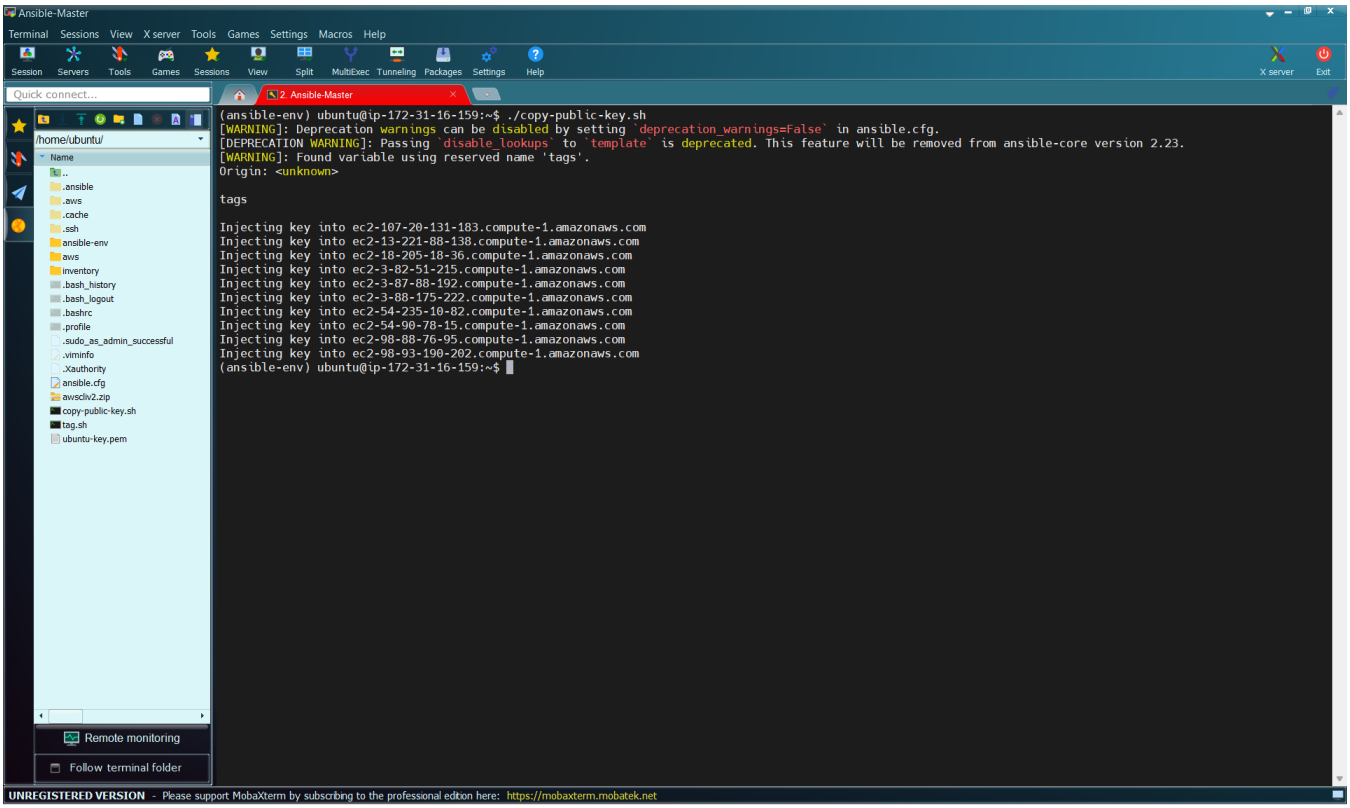
Then check again using the command:

```
ls
```



Then execute the file using the command:

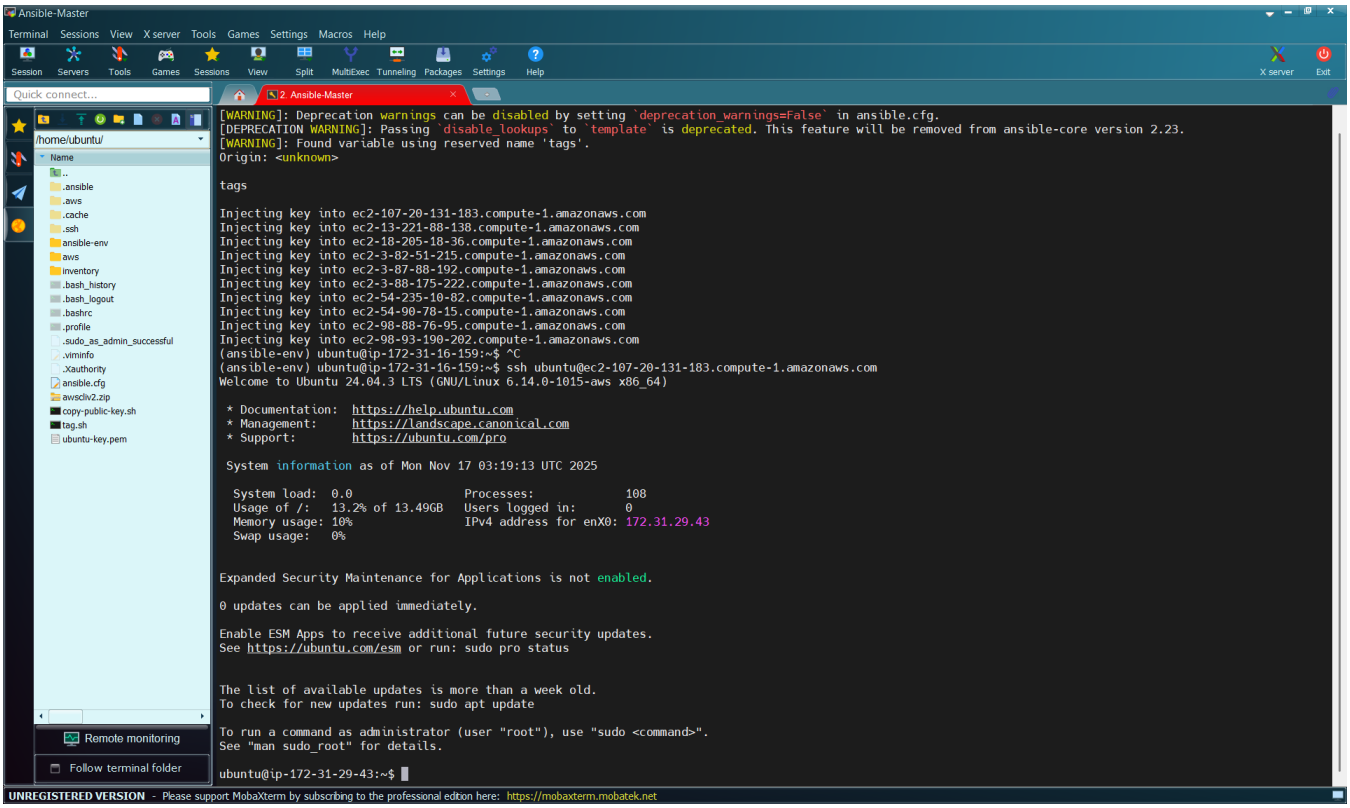
```
./copy-public-key.sh
```



It has copied the public key and set up the permission on the public key, set up the permission on the authorized keys on the target servers.

Once this is done then we will be able to access the machines on target server very easily. We can now connect to the first target server by running the command:

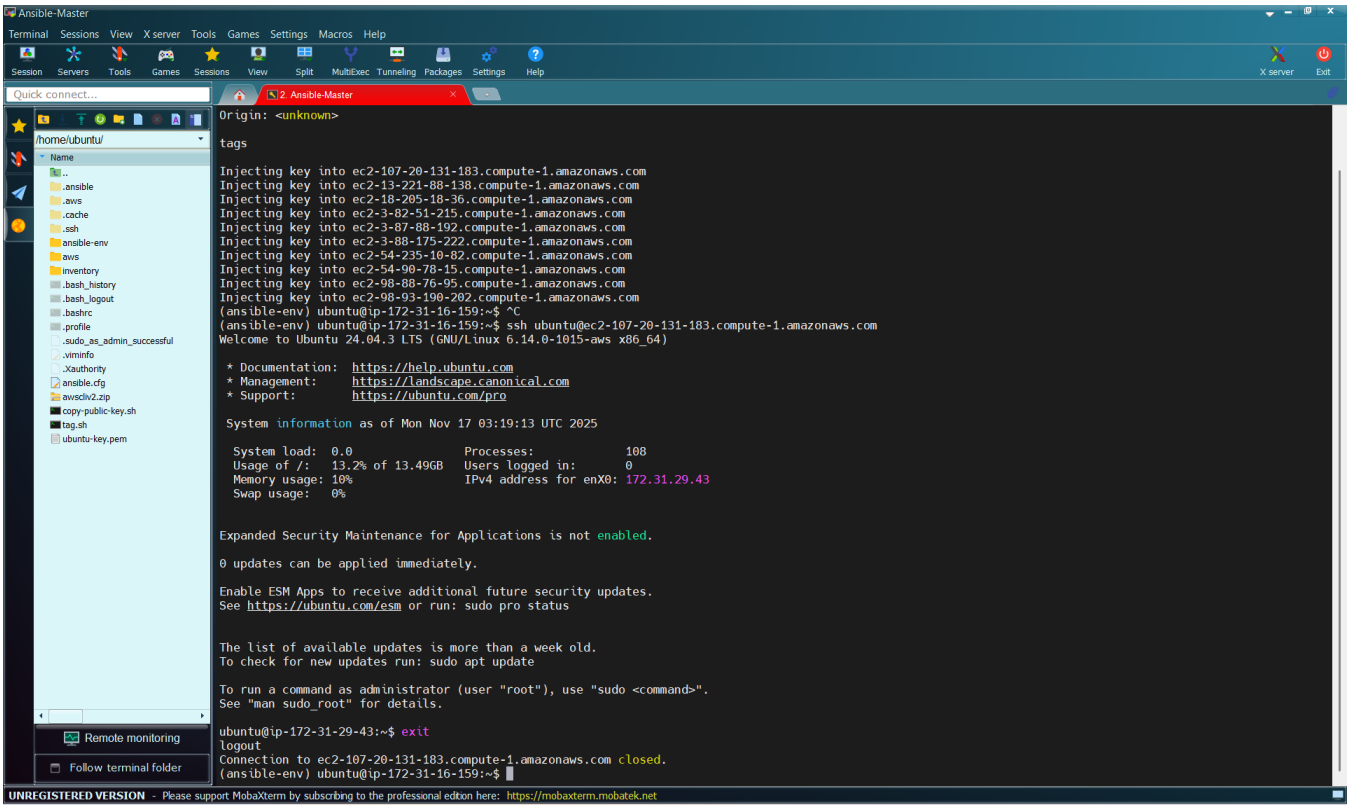
```
ssh ubuntu@ec2-107-20-131-183.compute-1.amazonaws.com
```



You can see that we are connected to the first target server. We are able to connect. That means from the Ansible side, "**Ansible-Master**" side now we are connected to the target servers and now we can go ahead and create our project code so that we can perform the task that is get the node metrics and send it over email.

Let us go back to our environment by using the command:

```
exit
```

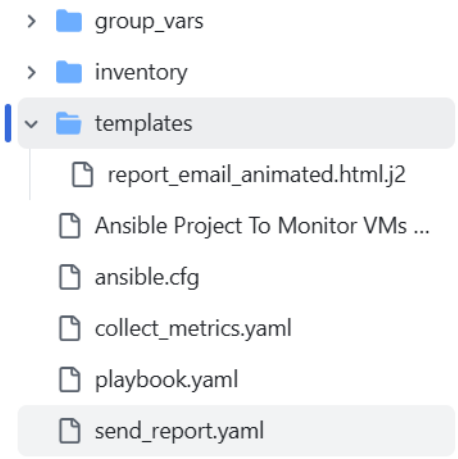

Ansible-Master
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
2 Ansible-Master
/home/ubuntu/
Name
.ansible
.aws
.cache
.ssh
ansible-env
aws
inventory
.bash_history
.bash_logout
.bashrc
.profile
.sudo_as_admin_successful
.viminfo
.Xauthority
ansible.cfg
awscliv2.zip
copy-public-key.sh
tag.sh
ubuntu-key.pem
Remote monitoring
Follow terminal folder
Origin: <unknown>
tags
Injecting key into ec2-107-20-131-183.compute-1.amazonaws.com
Injecting key into ec2-13-221-88-138.compute-1.amazonaws.com
Injecting key into ec2-18-205-18-36.compute-1.amazonaws.com
Injecting key into ec2-3-82-51-215.compute-1.amazonaws.com
Injecting key into ec2-3-87-88-192.compute-1.amazonaws.com
Injecting key into ec2-3-88-175-222.compute-1.amazonaws.com
Injecting key into ec2-54-235-10-82.compute-1.amazonaws.com
Injecting key into ec2-54-90-78-15.compute-1.amazonaws.com
Injecting key into ec2-98-88-76-95.compute-1.amazonaws.com
Injecting key into ec2-98-93-190-202.compute-1.amazonaws.com
(ansible-env) ubuntu@ip-172-31-16-159:~$ ^C
(ansible-env) ubuntu@ip-172-31-16-159:~$ ssh ubuntu@ec2-107-20-131-183.compute-1.amazonaws.com
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro
System information as of Mon Nov 17 03:19:13 UTC 2025
System load: 0.0
Usage of /: 13.2% of 13.49GB
Memory usage: 10%
Swap usage: 0%
Processes: 108
Users logged in: 0
IPv4 address for enX0: 172.31.29.43
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
ubuntu@ip-172-31-29-43:~$ exit
logout
Connection to ec2-107-20-131-183.compute-1.amazonaws.com closed.
(ansible-env) ubuntu@ip-172-31-16-159:~$
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

STEP 7: Setting up Ansible

Now, let us set up Ansible. Once we set up Ansible and make sure that Ansible-Master is able to connect to the target servers. Next thing that we have to do is to set up this project.

Let me explain to you the structure of the project and what files do we have to create. This is the GitHub repository of the project:

https://github.com/jaiswaladi246/Ansible-VM-Monitor/tree/main/templates



All these files and folders are in the "**Ansible-VM-Monitor**" folder. Now, let us see what each file is going to do and why do we have to create it.

First, we have "**group_vars**", when you have like multiple playbooks that you want to use specific variables, you know in Ansible there are multiple ways to create variables.

First way is that we call it as "**inline**" variables. In inline variables, you create or define variables inside the playbook itself. But this is not a good practice.

The second way is that we call "**External**" variable. In this, what happens is we create a separate folder like "**group_vars**" and inside that we create one file, let us call it "**all.yaml**". So, we will be adding variable and their values inside this. This is what external variable file means. This is very useful because this file can be utilized in multiple playbooks.

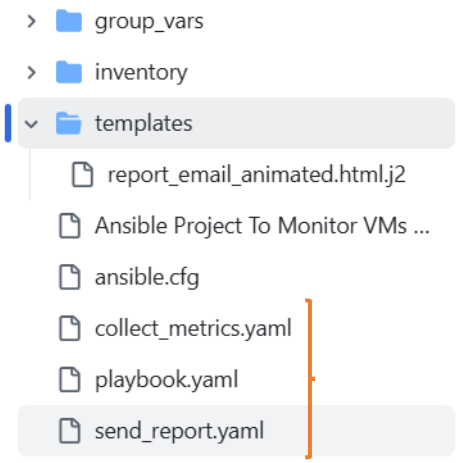
In this file, we are going to define our variables and also in the variables what things do we define. Basically, we are going to be defining information about the email, to where the email should be sent, email ID, of where we are going to send the report and a password of the email. The password is not the actual email password, it is a password that is known as "**app**" password.

The thing is when you want to use your email address inside like Pipelines, or Ansible playbooks, or anywhere else, you should be using app passwords. These are quite useful because these passwords cannot be used to access your email. Instead, they will be utilized with specific applications. Like here we are using this app password with Ansible. So, even if someone is able to get this app password, they won't be able to access your email.

We have already created the "**aws_ec2.yaml**" file for the "**inventory**" folder. Also, we have already created the file "**ansible.cfg**". But we have to still define them here because when you are going to run the Ansible playbook, generally you have to provide the path of Ansible with the inventory file. So, if I am defining it here, then it won't be required for us to separately define the path. Since they are in the same directory, we can simply run the Ansible playbook without defining the path of inventory files. For that reson, I am going to keep the two files in the project folder.

Next, we have template folder. Inside the template folder we have the file "**report_email_animated.html.j2**", which is a J2 extended file. This file will be containing the HTML code. The key is that, we want that when we receive the VM metrics report, we want them to be present in a very specific and beautiful format. So, we can create that format using HTML code. For that we are going to keep it inside "**templates**" folder.

Then we have the most important part of the playbooks shown below



First, we have "**collect_metrics.yaml**". Here we are going to be writing the code using different kind of modules which is going to basically fetch the metrics. That is the node metrics of CPU usage, RAM usage and Disk usage.

Then we have "**send_report.yaml**". This is going to write the code which we will be able to send the report to the defined email address.

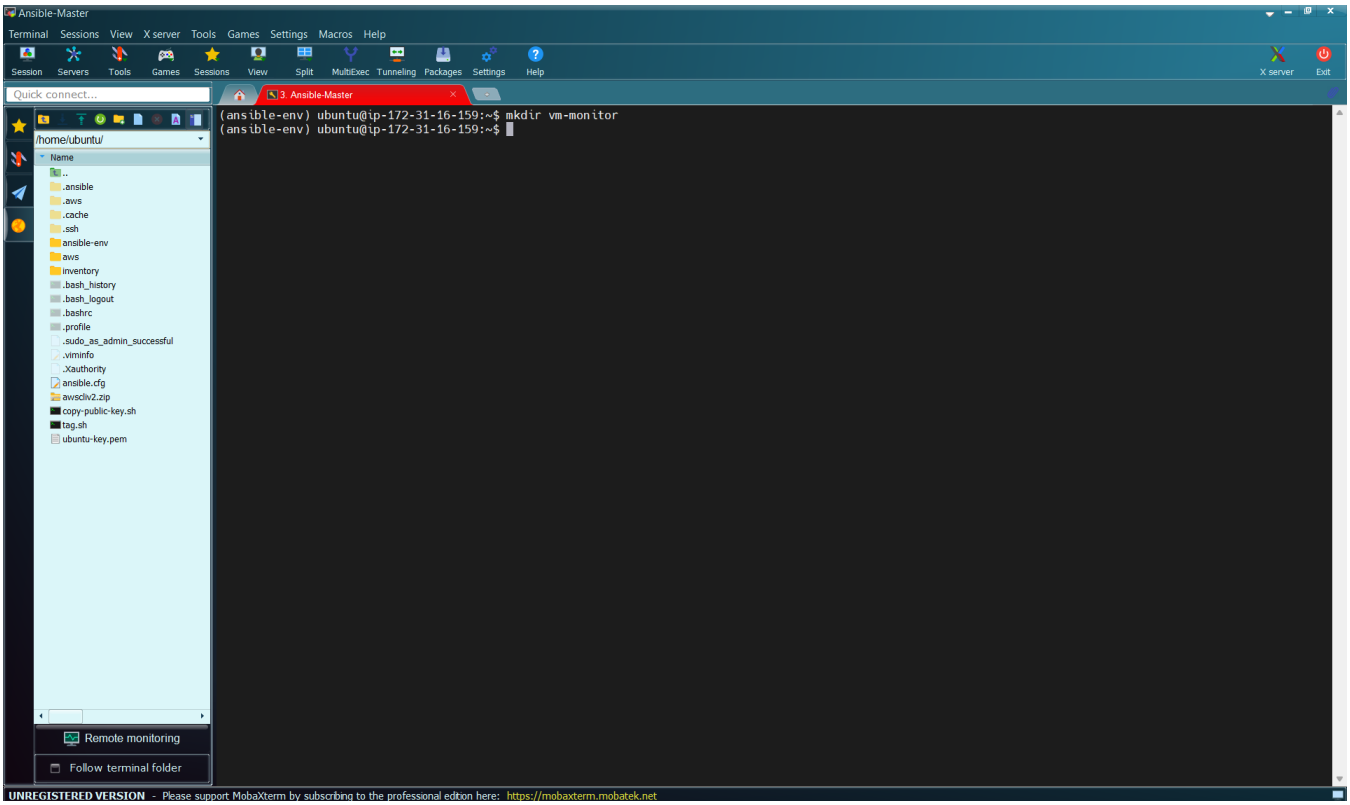
Then we will be executing the whole project using the "**playbook.yaml**" file. Here we will import the "**collect_metrics.yaml**" and the "**send_report.yaml**" files. Once we do that, then we will be able to execute our whole project by running the Command:

```
ansible-playbook
```

Then we can simply call "**playbook.yaml**" and this is going to execute the two imported files.

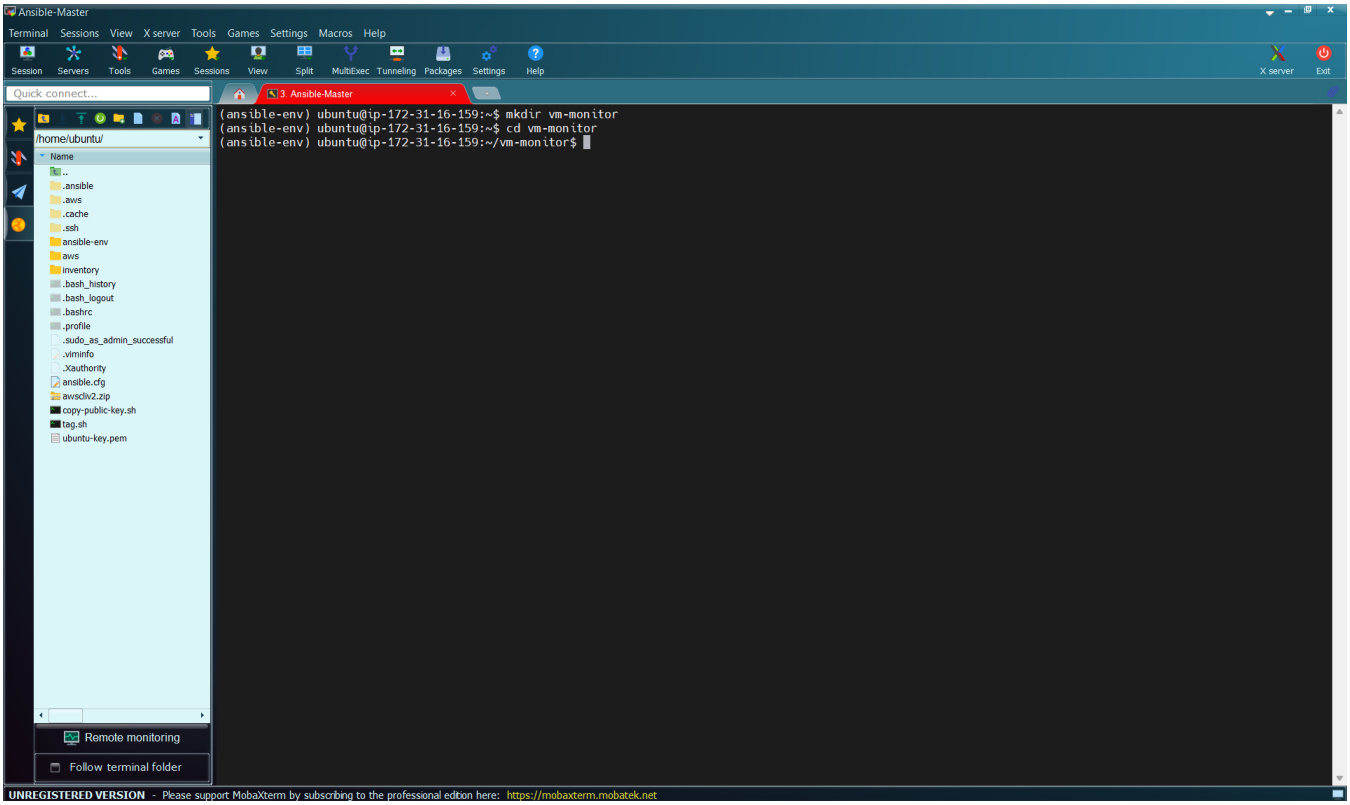
Let us start creating the folders and files for this project. We will start by creating the project folder called "**vm-monitor**", using the command:

```
mkdir vm-monitor
```



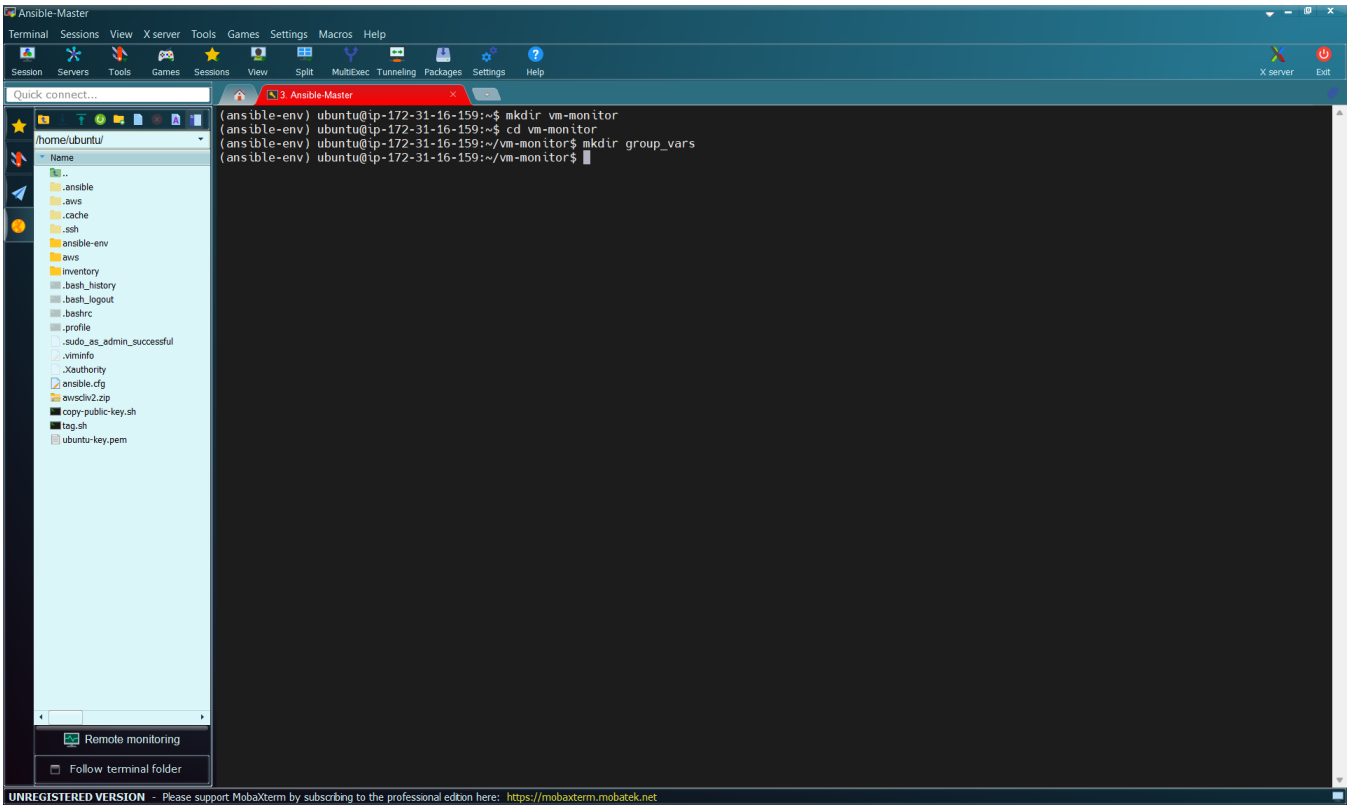
Then next we create the "**group_vars**" folder inside the project folder. Let us navigate to the project folder using the command:

```
cd vm-monitor
```



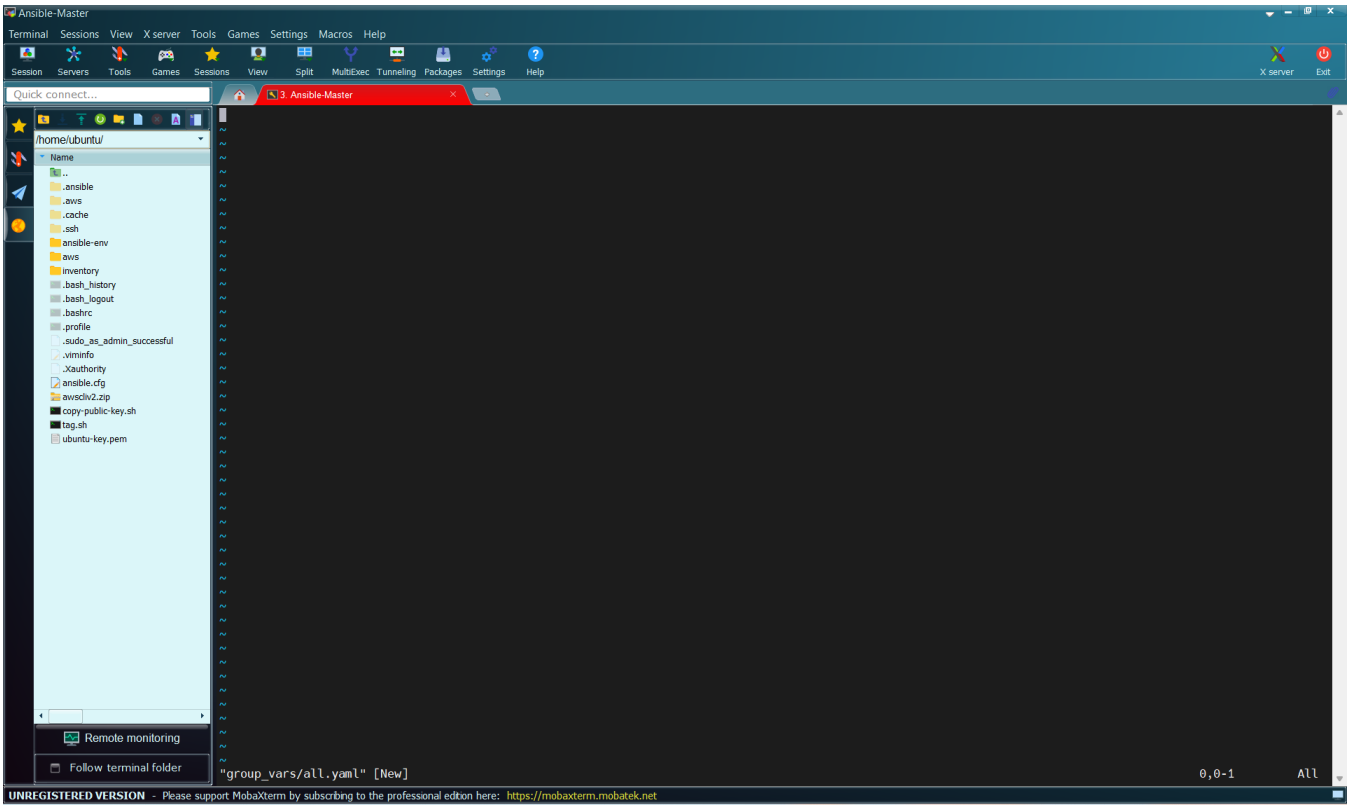
Then, we create the folder "**group_vars**" using the command:

```
mkdir group_vars
```



We have created the "**group_vars**" folder, now let is create the "**all.yaml**" file inside this folder by using the command:

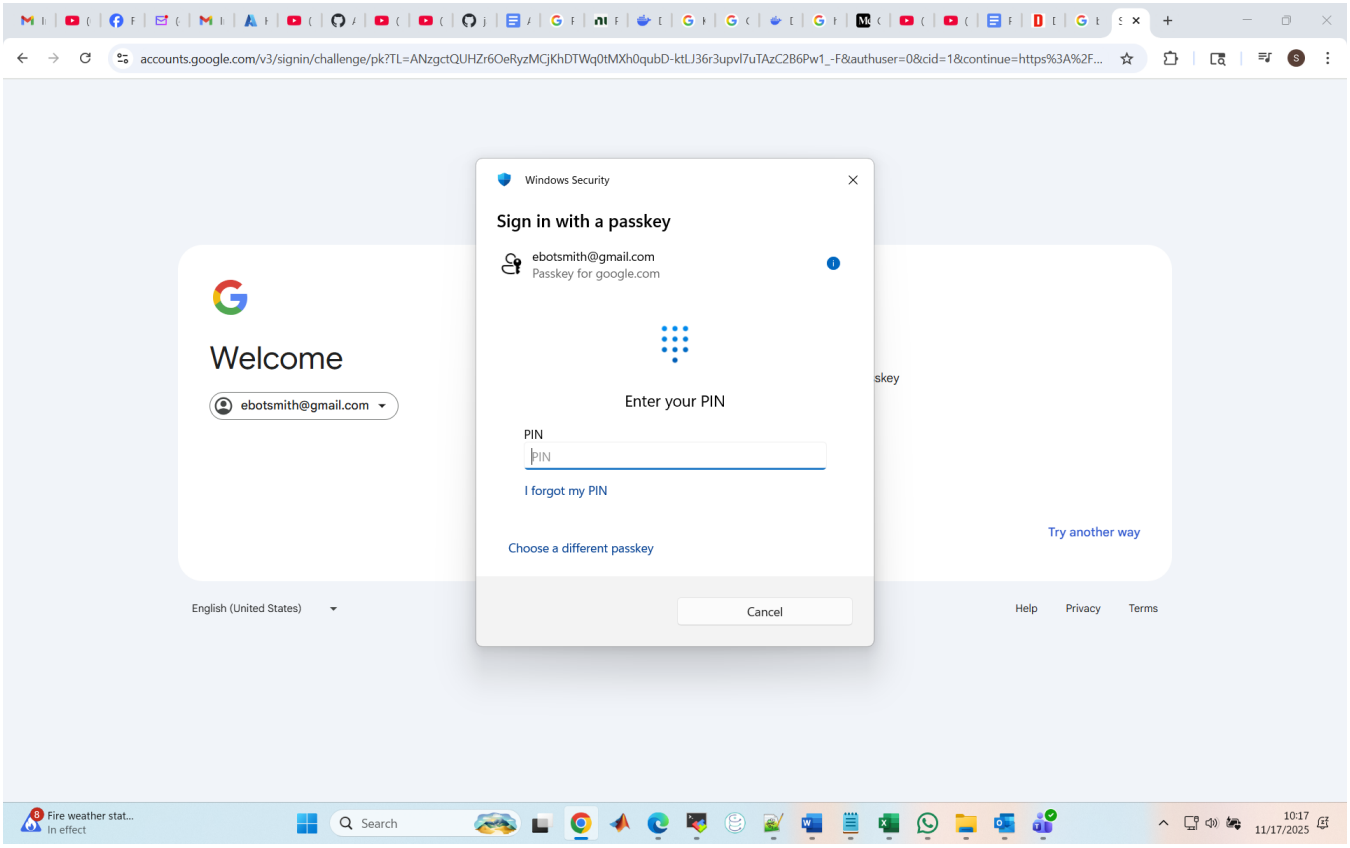
```
vi group_vars/all.yaml
```



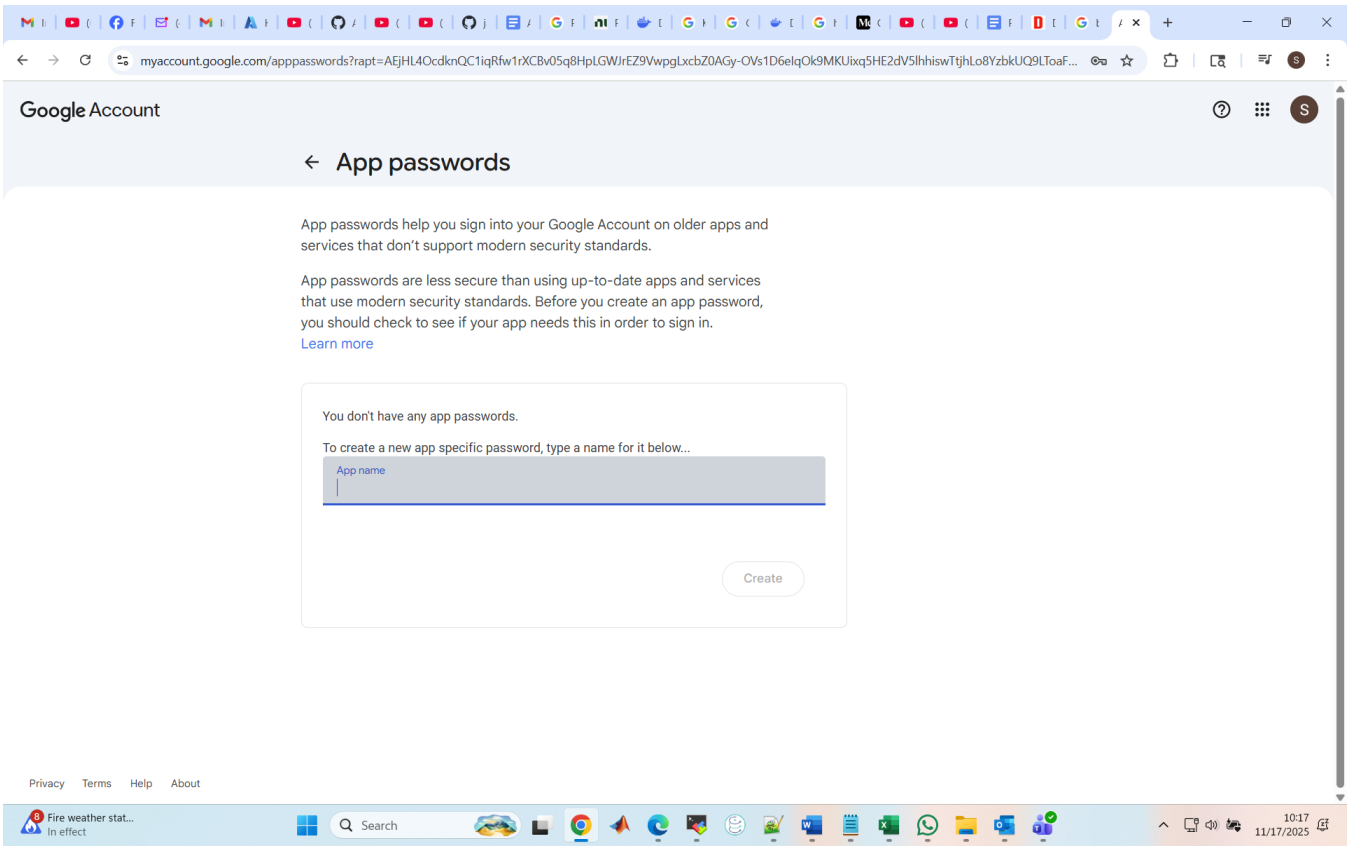
Then paste this code here

```
smtp_server: "smtp.gmail.com"
smtp_port: 587
email_user: "ebotsmith@gmail.com"
email_pass: "tqlj wtju jjgq fhay"
alert_recipient: "--"
```

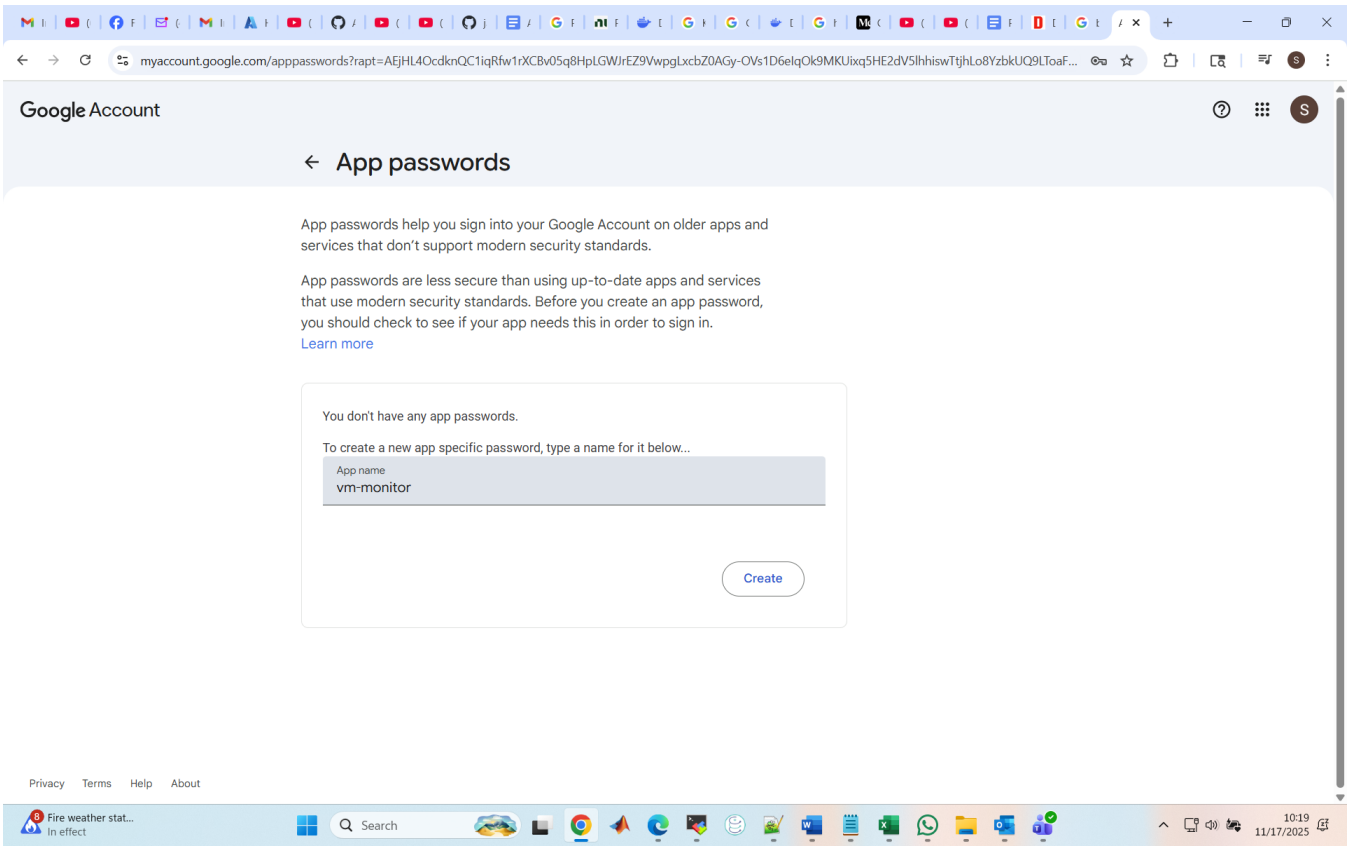
To get the email password, go to myaccount.google.com/apppasswords



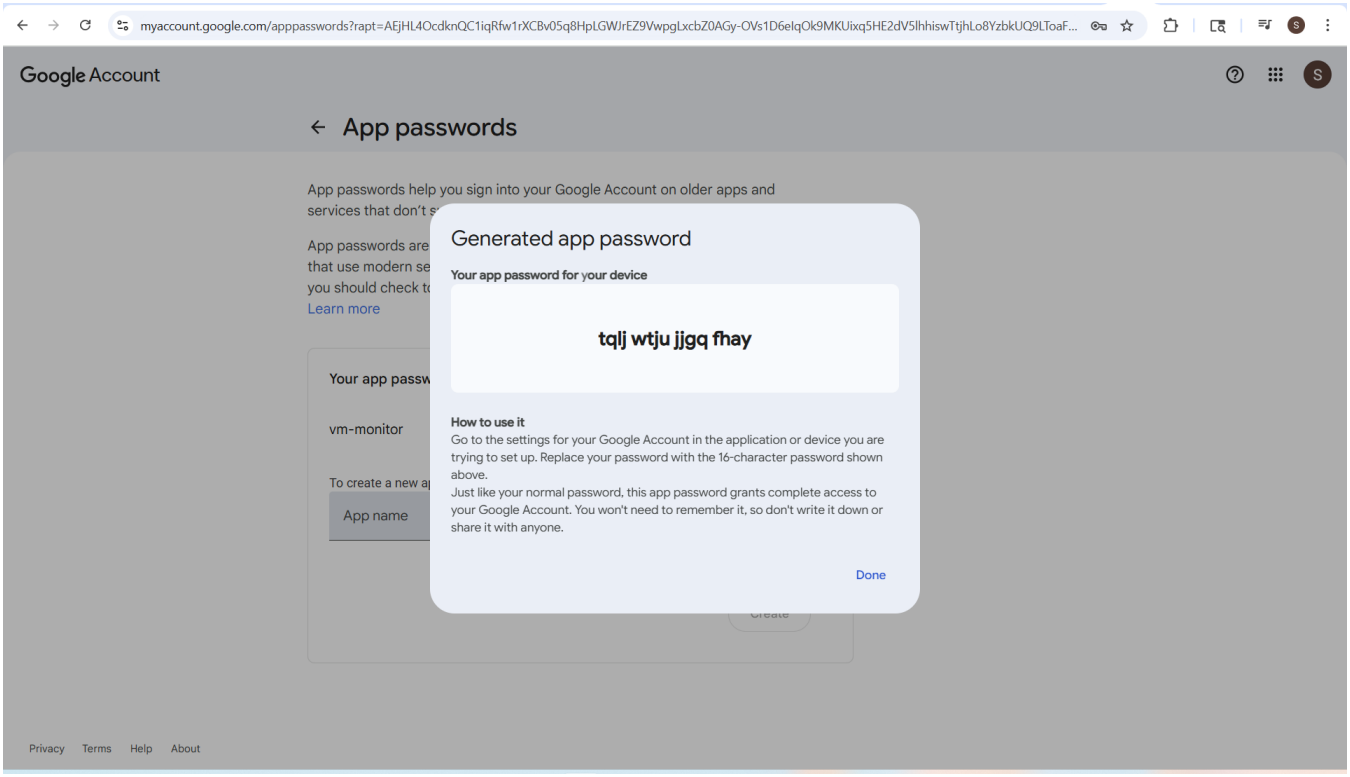
Enter your pin



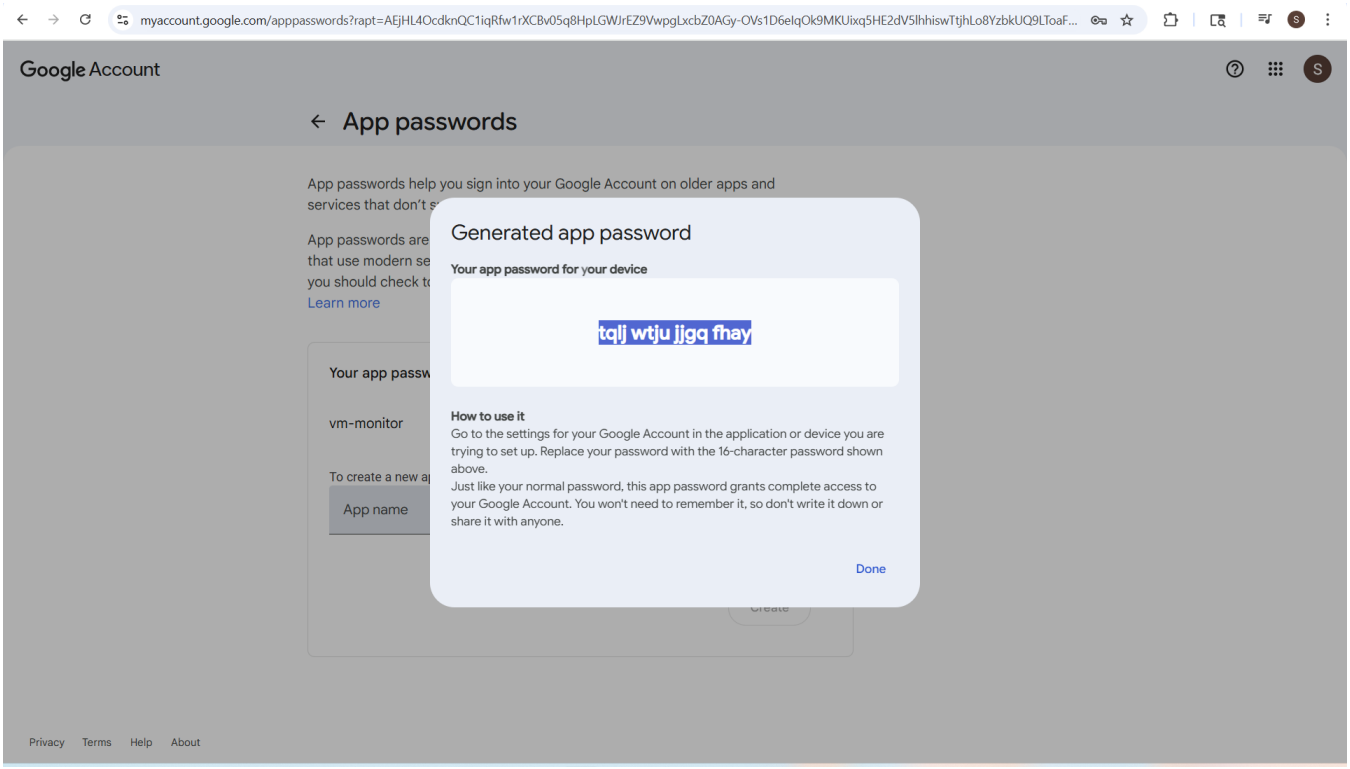
Enter the app name, I will call it "**vm-monitor**"



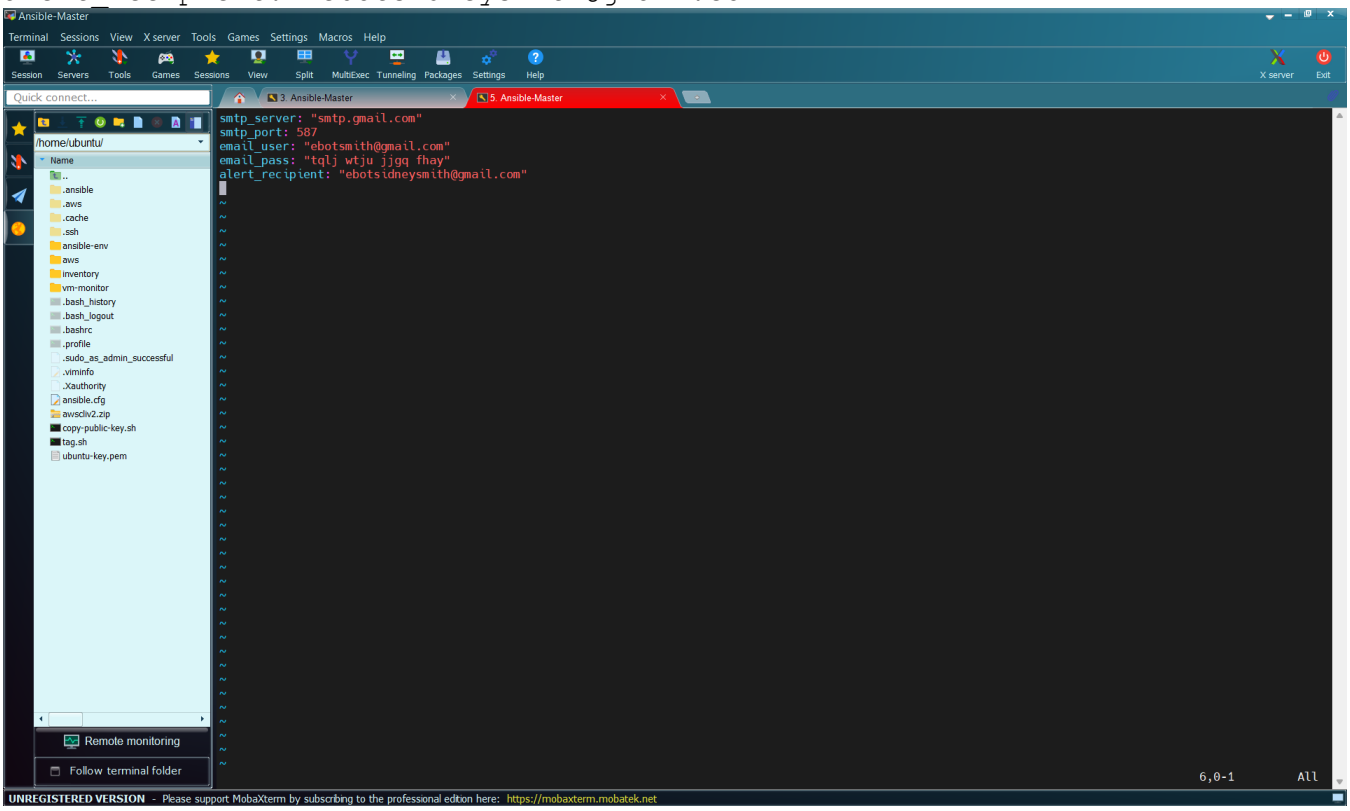
Click on "**Create**"



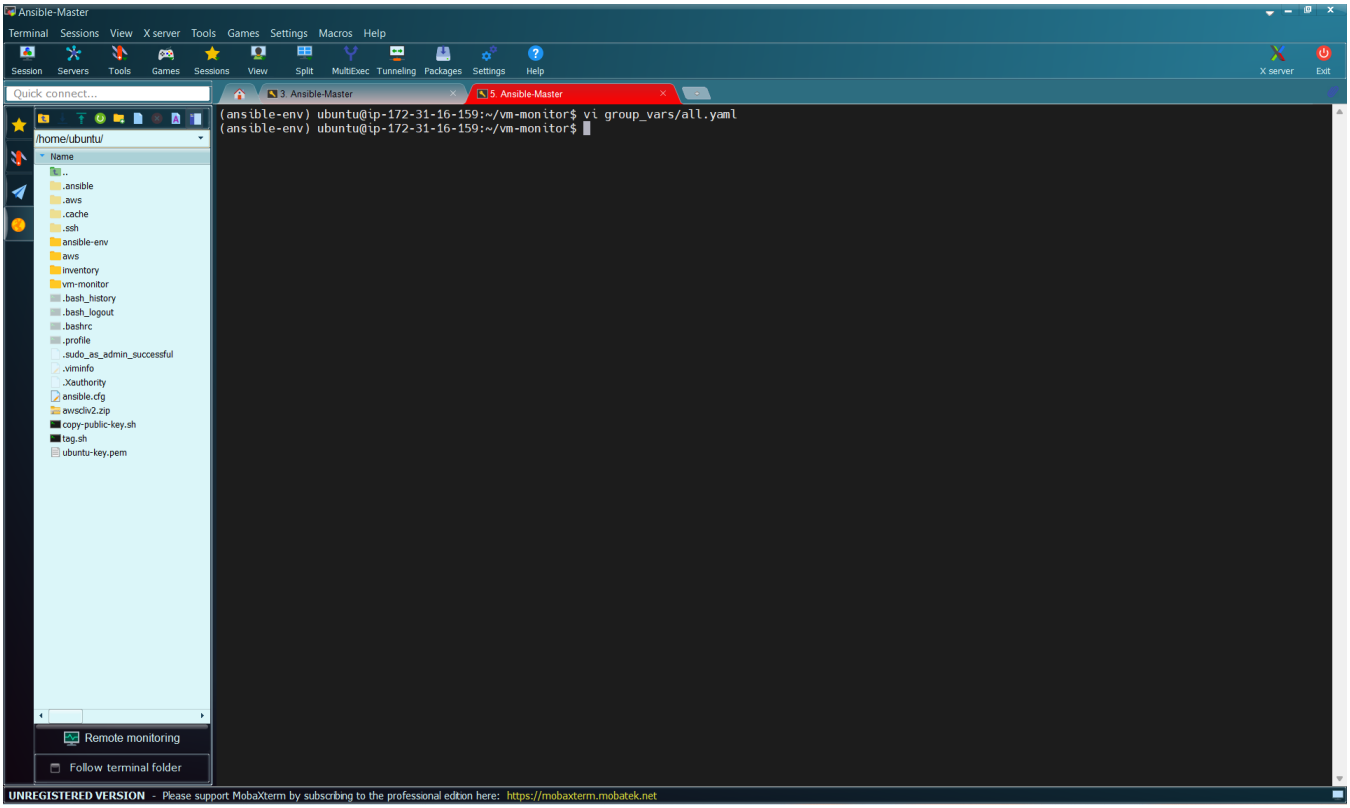
Copy the password and add on your code




```
smtp_server: "smtp.gmail.com"
smtp_port: 587
email_user: "ebotsmith@gmail.com"
email_pass: "tqlj wtju jjgq fhay"
alert_recipient: "ebotsidneysmith@gmail.com"
```

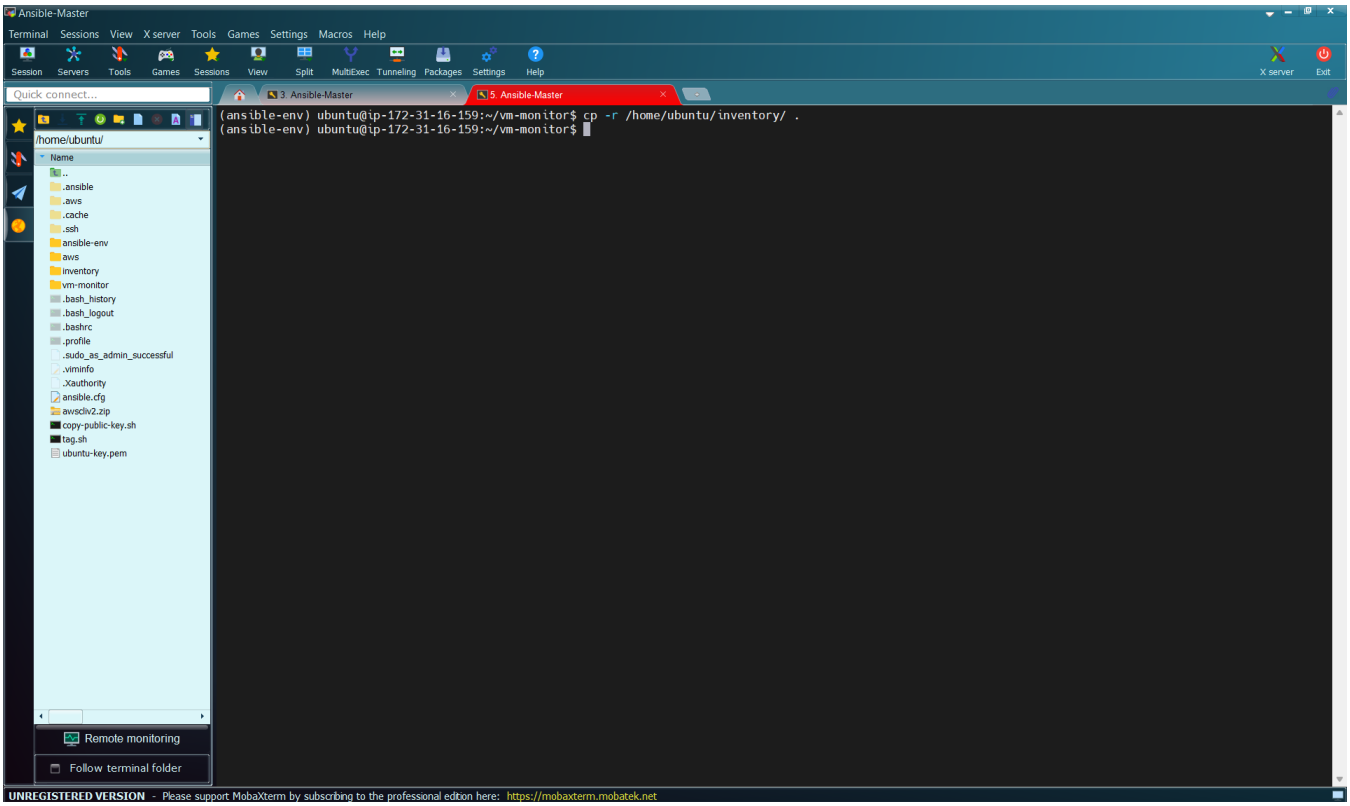


Then save and edit by pressing ESC followed by :wq and press "**enter**". Note make sure you add **port 587** in the inbound rules of Ansible-Master instance.



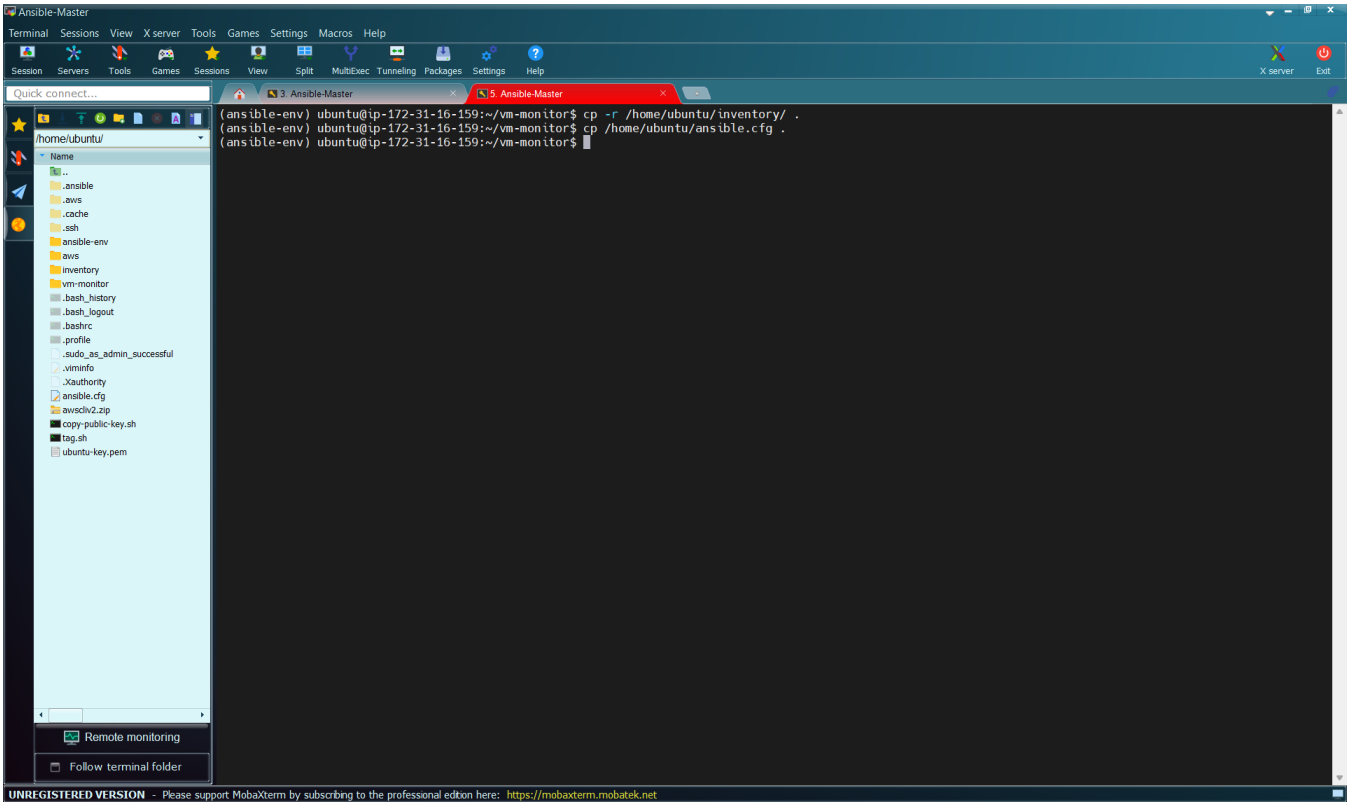
Now, let us copy the folder "**inventory**" and the file "**ansible.cfg**" that we created before into our project folder. We will copy the folder by using the command:

```
cp -r /home/ubuntu/inventory/ .
```



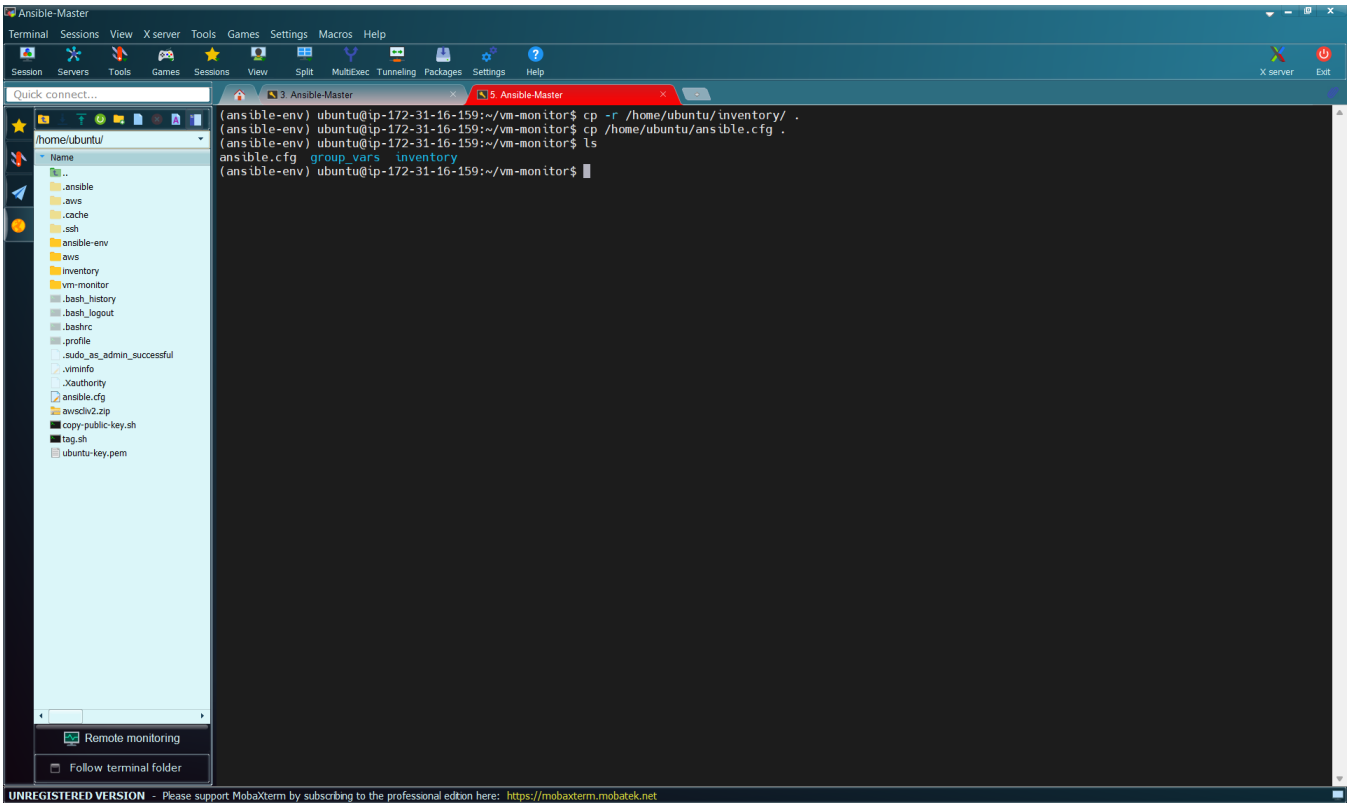
Then we will copy the file by using this command:

```
cp /home/ubuntu/ansible.cfg .
```



We can check the content of our project folder by using the command:

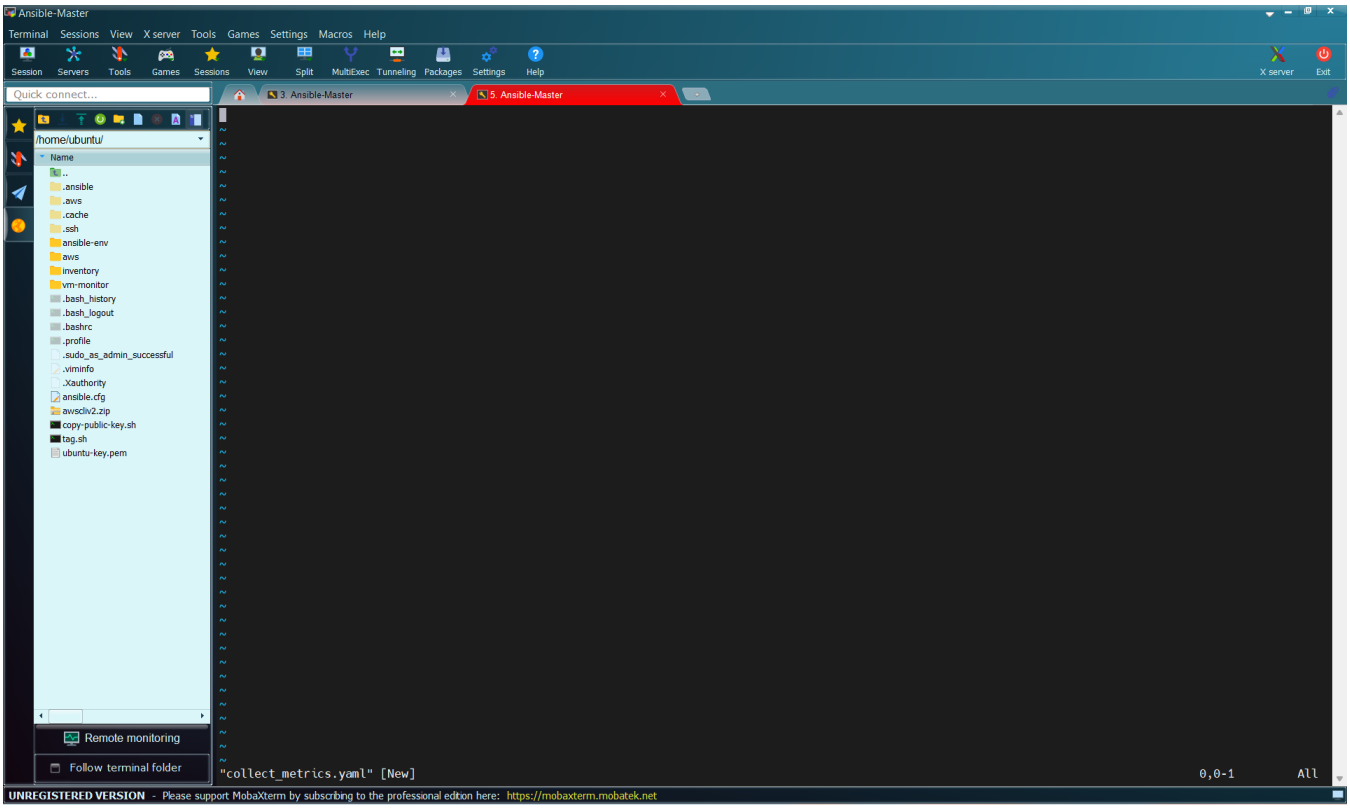
```
ls
```



You can see what we have added so far.

The next thing is to create the file "**collect_metrics.yaml**" using the command:

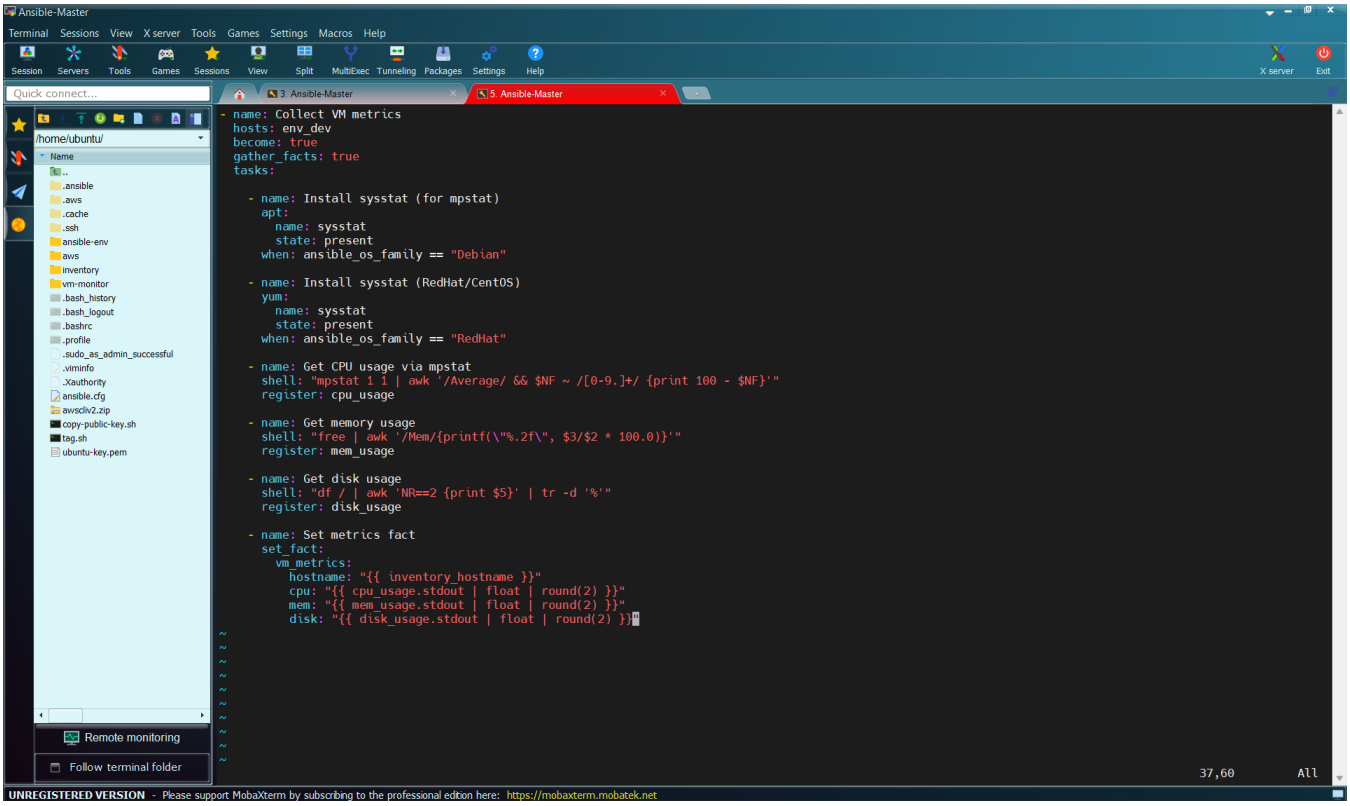
```
vi collect_metrics.yaml
```



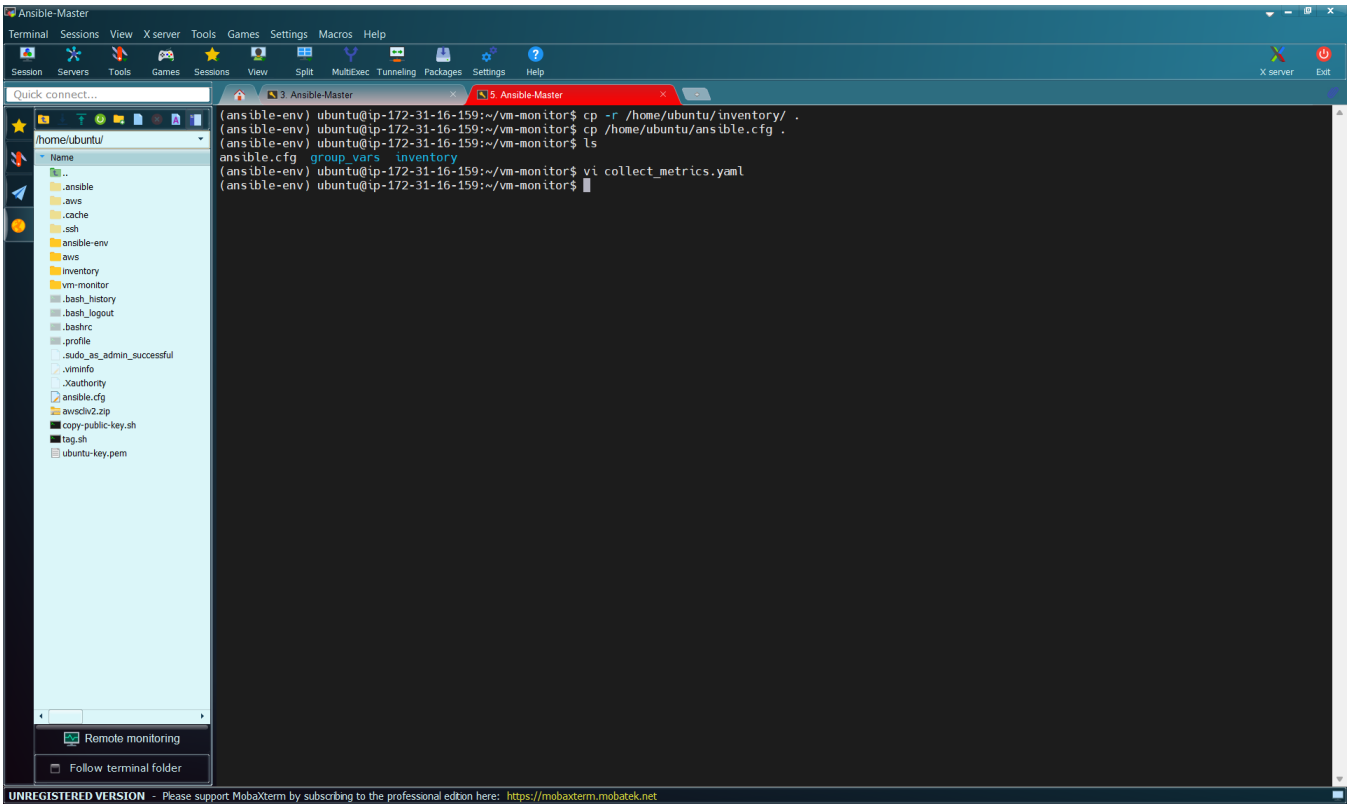
Then, paste the code

```
- name: Collect VM metrics
  hosts: env_dev
  become: true
  gather_facts: true
  tasks:

    - name: Install sysstat (for mpstat)
      apt:
        name: sysstat
        state: present
      when: ansible_os_family == "Debian"

    - name: Install sysstat (RedHat/CentOS)
      yum:
        name: sysstat
        state: present
      when: ansible_os_family == "RedHat"

    - name: Get CPU usage via mpstat
      shell: "mpstat 1 1 | awk '/Average/ && $NF ~ /[0-9.]+/ {print 100 - $NF}'"
      register: cpu_usage

    - name: Get memory usage
      shell: "free | awk '/Mem/{printf(\"%.2f\", $3/$2 * 100.0)}'"
      register: mem_usage

    - name: Get disk usage
      shell: "df / | awk 'NR==2 {print $5}' | tr -d '%'"
      register: disk_usage

    - name: Set metrics fact
```

```
    set_fact:
      vm_metrics:
        hostname: "{{ inventory_hostname }}"
        cpu: "{{ cpu_usage.stdout | float | round(2) }}"
        mem: "{{ mem_usage.stdout | float | round(2) }}"
        disk: "{{ disk_usage.stdout | float | round(2) }}"
```



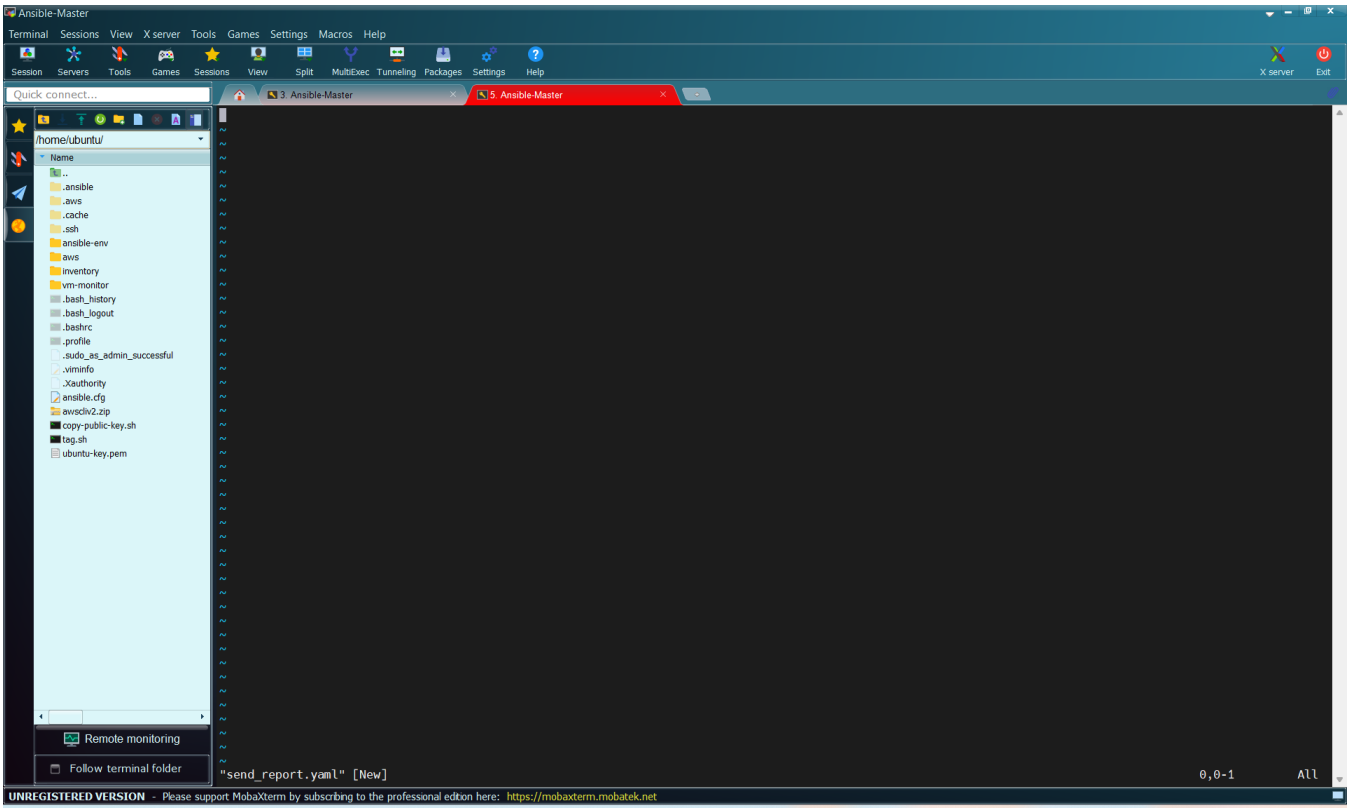
Now, let us save and exit the file by pressing “ESC” followed by :wq and press “Enter”



So, we have written the playbook to collect the metrics.

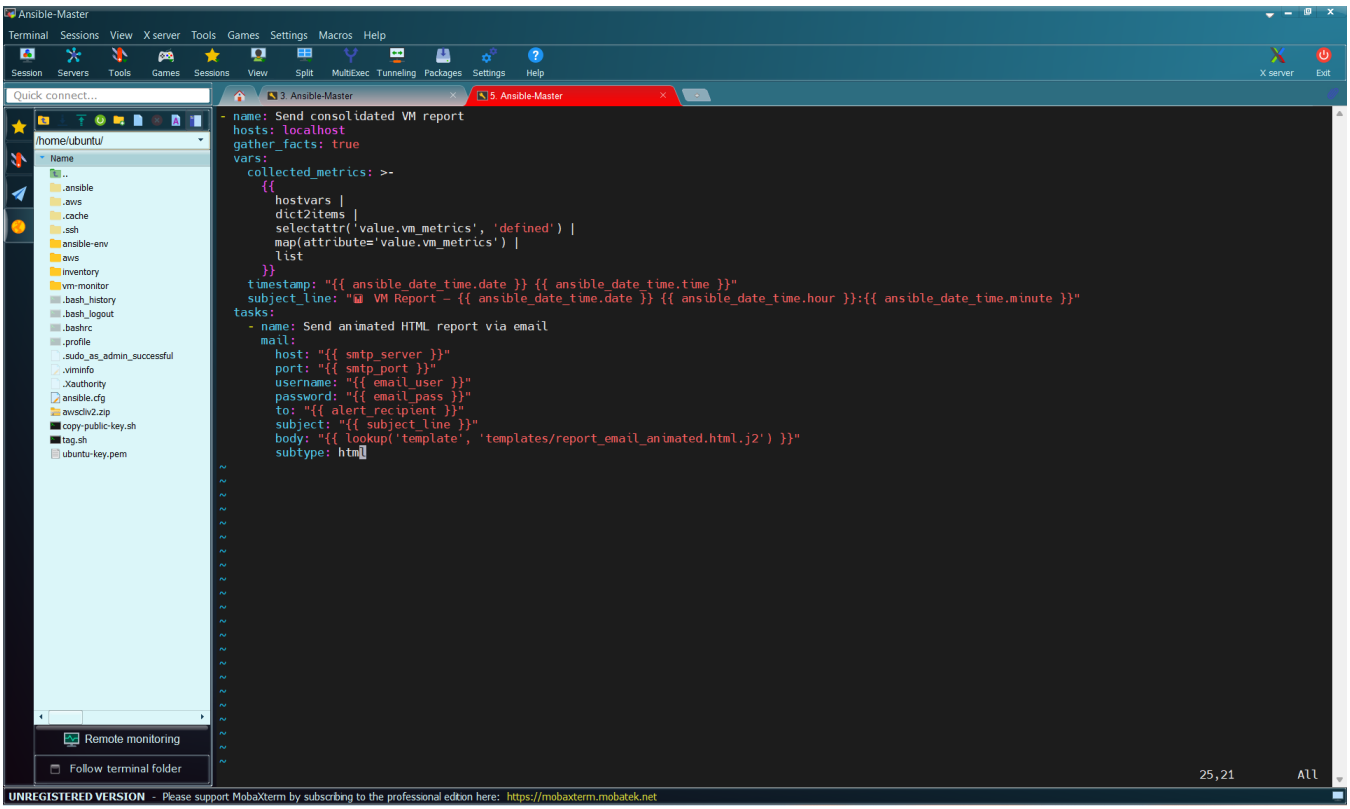
The next thing we have to do is to write the playbook to send the collected metrics to the email. The file will be called "**send_report.yaml**". Let us create the file using the command:

```
vi send_report.yaml
```

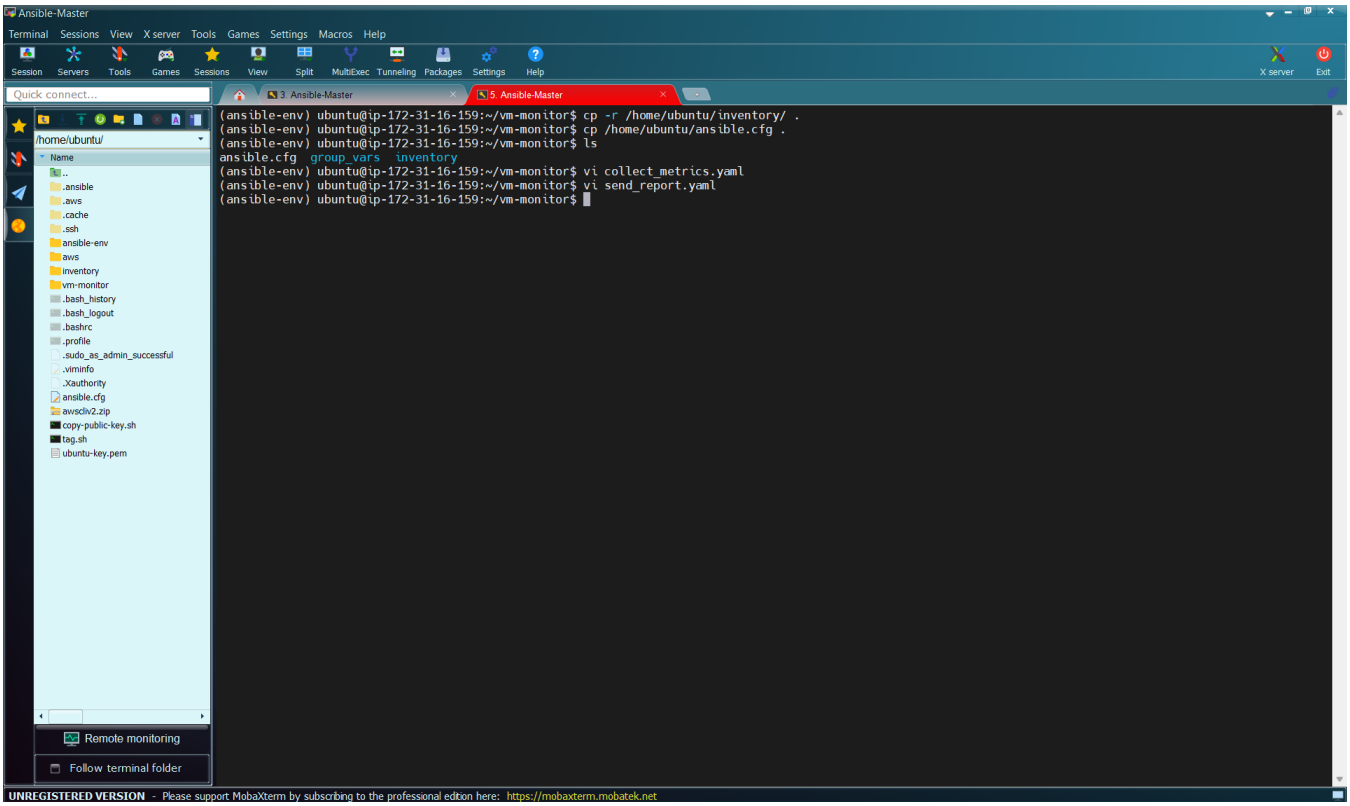


And we paste the code:

```
- name: Send consolidated VM report
  hosts: localhost
  gather_facts: true
  vars:
    collected_metrics: >-
      {{
        hostvars |
        dict2items |
        selectattr('value.vm_metrics', 'defined') |
        map(attribute='value.vm_metrics') |
        list
      }}
    timestamp: "{{ ansible_date_time.date }} {{ ansible_date_time.time }}"
    subject_line: "📊 VM Report – {{ ansible_date_time.date }} {{
ansible_date_time.hour }}:{{ ansible_date_time.minute }}"
  tasks:
    - name: Send animated HTML report via email
      mail:
        host: "{{ smtp_server }}"
        port: "{{ smtp_port }}"
        username: "{{ email_user }}"
        password: "{{ email_pass }}"
        to: "{{ alert_recipient }}"
        subject: "{{ subject_line }}"
        body: "{{ lookup('template',
'templates/report_email_animated.html.j2') }}"
        subtype: html
```



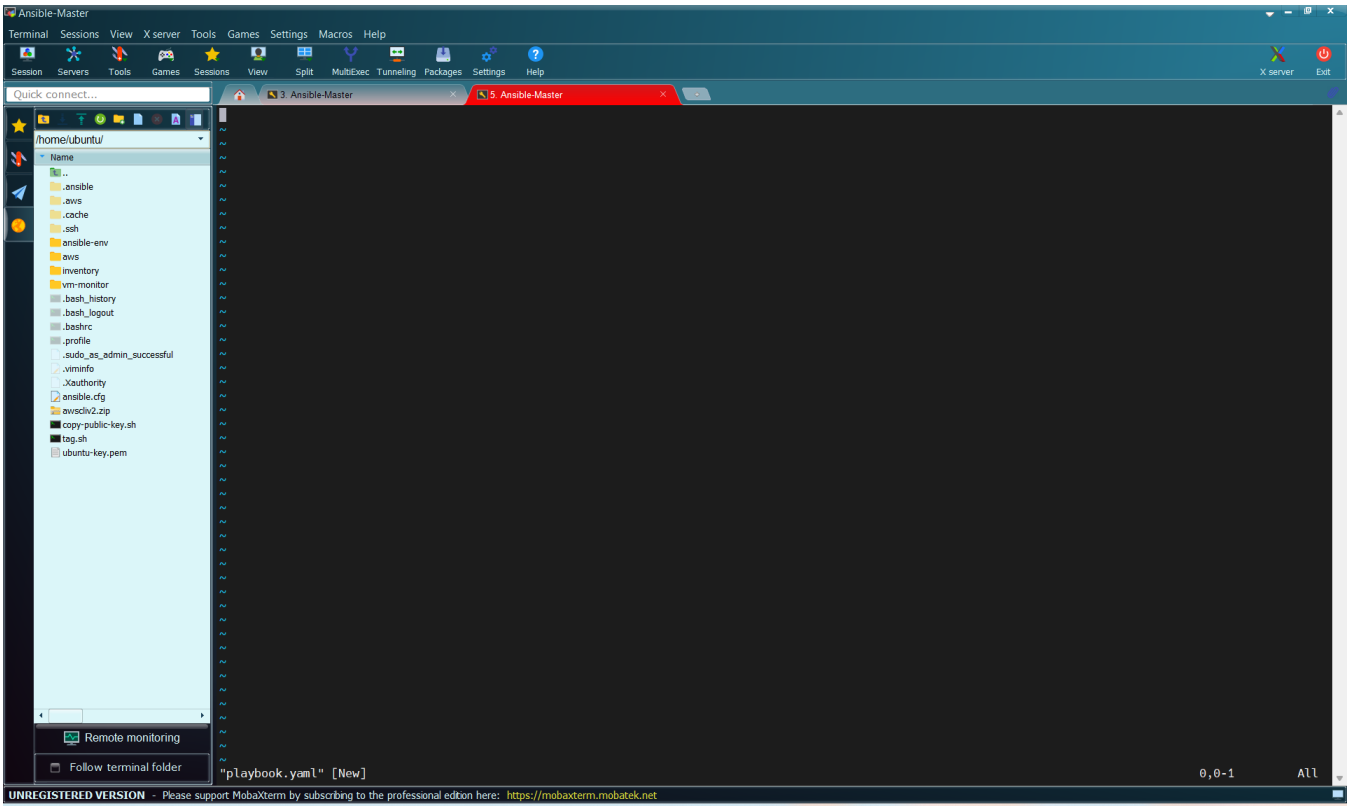
Save and exit by pressing “**ESC**” followed by :wq and press “ENTER”



Now, let us create the file which we are going to import the two playbooks, that is “**collect_metrics.yaml**” and “**send_report.yaml**”.

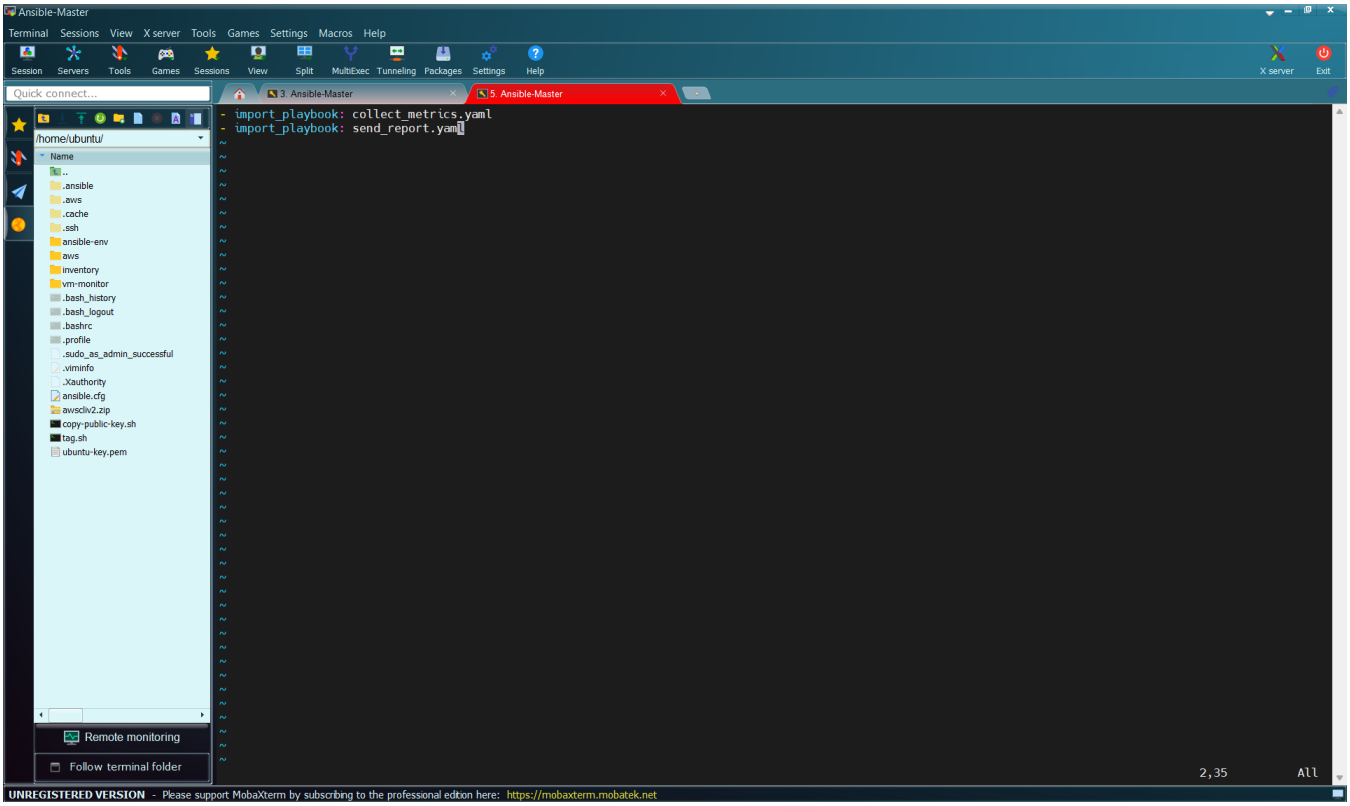
We will call the file “**playbook.yaml**”, that will be created by using the command:

```
vi playbook.yaml
```

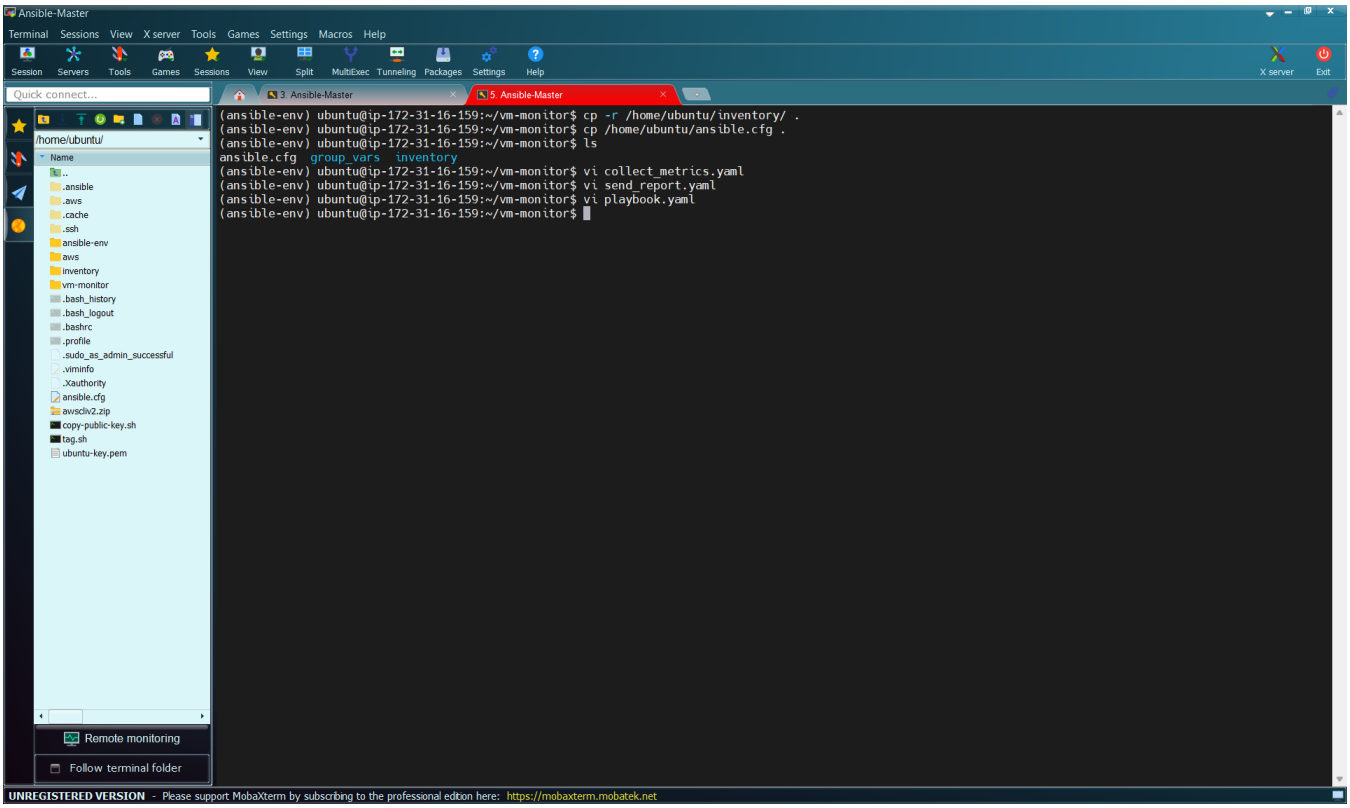


Then add the code:

```
- import_playbook: collect_metrics.yaml
- import_playbook: send_report.yaml
```

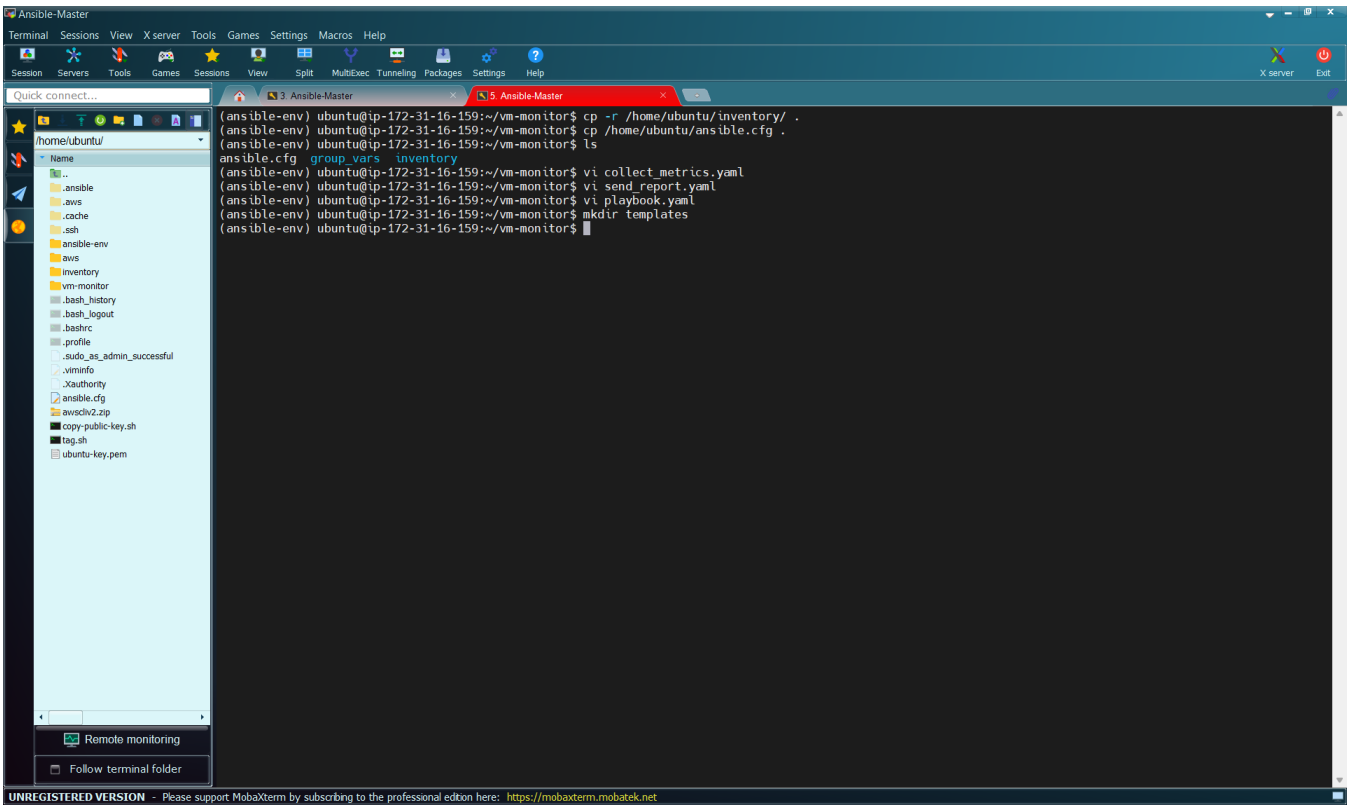


Save and exit by pressing “ESC” followed by :wq and press “ENTER”



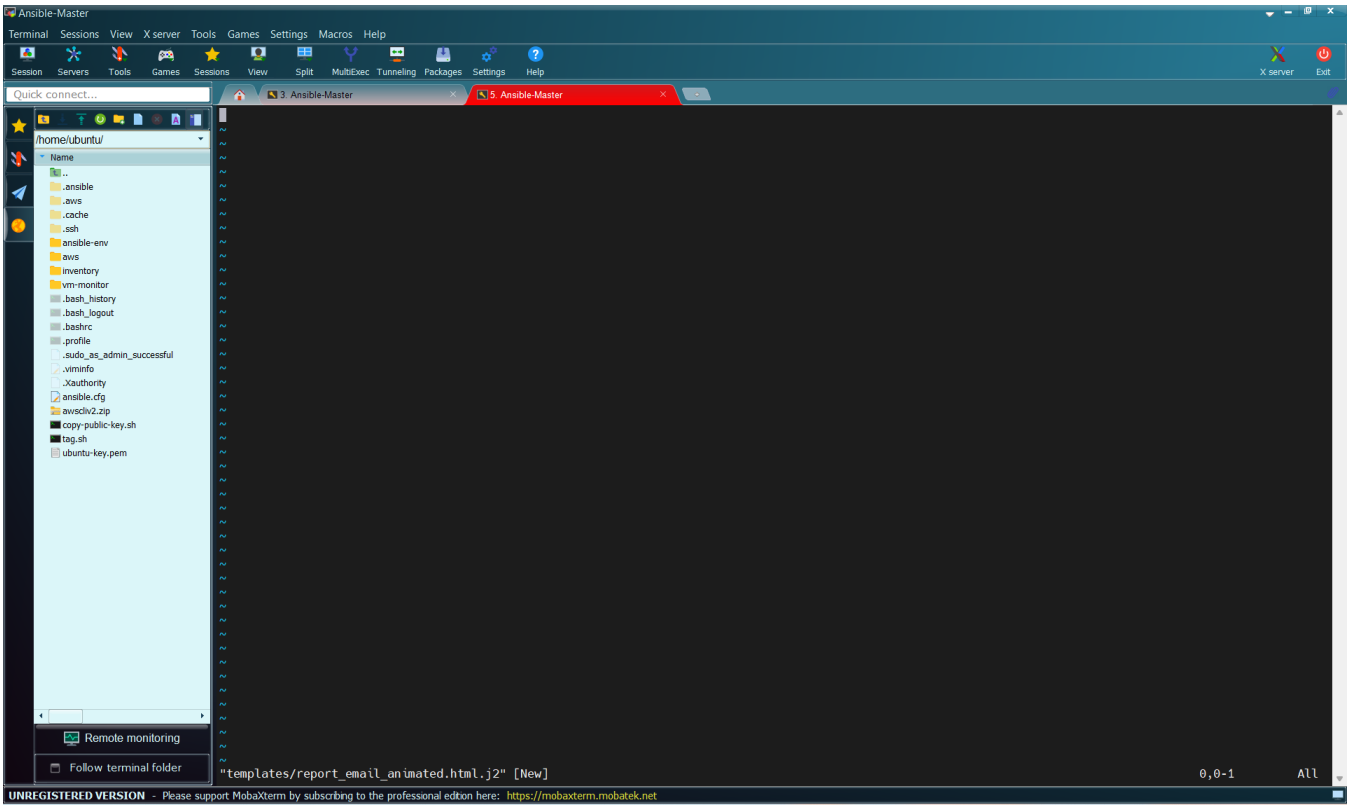
Finally, we have to create the email format. You can use the basic email format, but we want to use a user-friendly format with HTML. So, we are going to create a template for it. We will first create a folder for it called "**templates**" by using the command:

```
mkdir templates
```



Then let us create the file called "**report_email_animated.html.j2**" in this template folder by using the command:

```
vi templates/report_email_animated.html.j2
```



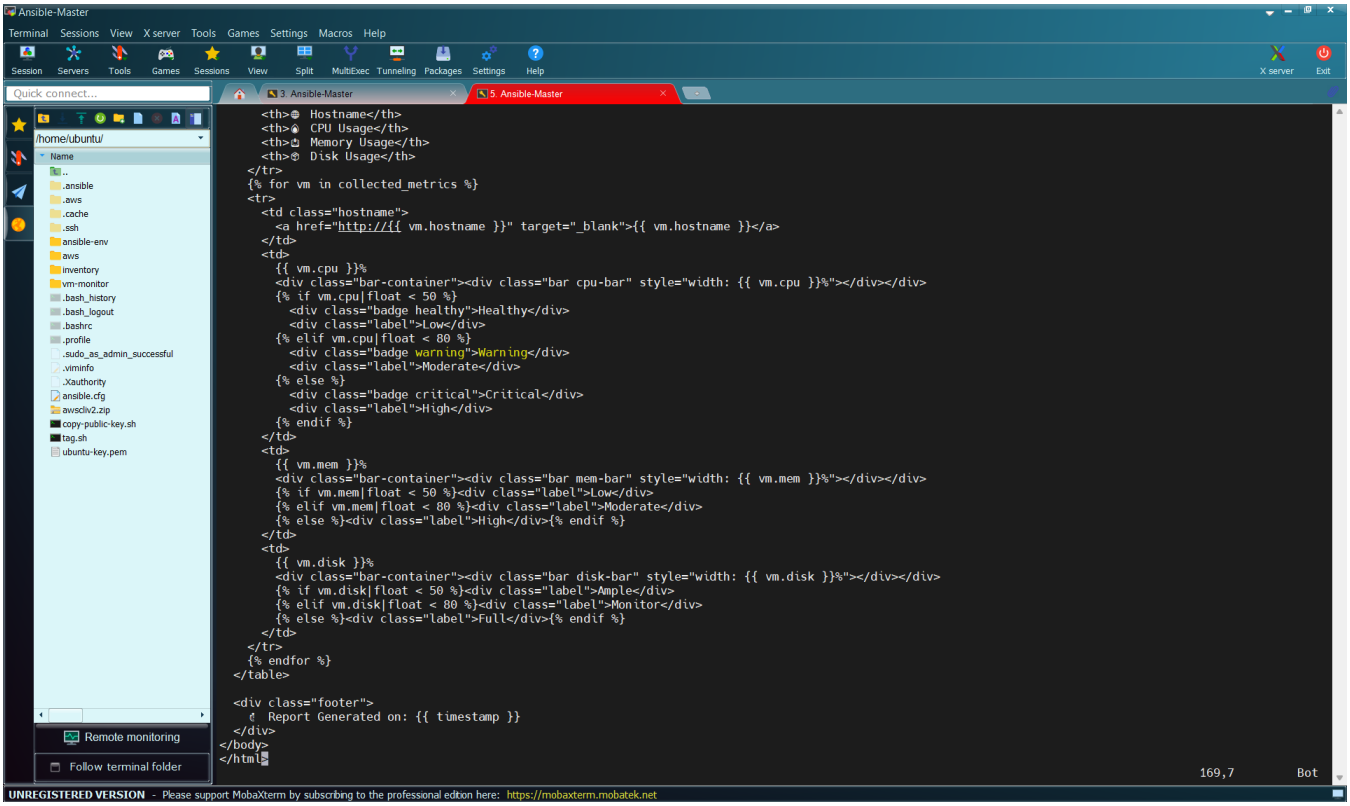
Then, let us paste the code:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <style>
    body {
      font-family: 'Segoe UI', sans-serif;
      background-color: #f9fbfc;
      padding: 32px;
      color: #333;
      max-width: 960px;
      margin: auto;
    }

    h2 {
      text-align: center;
      color: #2e7d32;
      font-size: 28px;
      margin-bottom: 32px;
      border-bottom: 2px solid #c8e6c9;
      padding-bottom: 10px;
    }

    .summary {
      text-align: center;
      margin-bottom: 30px;
      background-color: #e3f2fd;
      border: 1px solid #bbdefb;
      padding: 10px 18px;
```

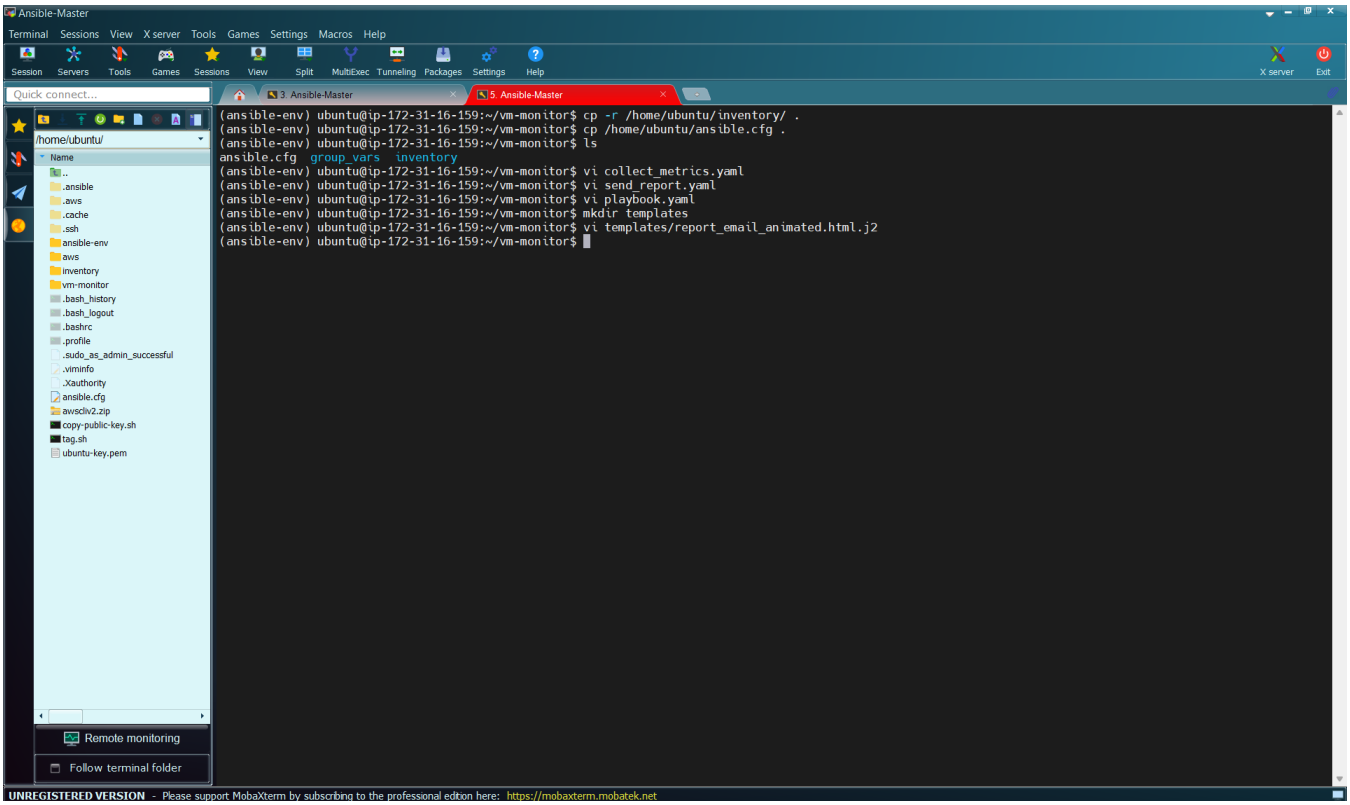
```
  border-radius: 8px;
  font-size: 14px;
  color: #0d47a1;
}

table {
  width: 100%;
  border-collapse: collapse;
  box-shadow: 0 4px 12px rgba(0,0,0,0.08);
  background-color: #ffffff;
  border-radius: 10px;
  overflow: hidden;
}

th {
  background-color: #1565c0;
  color: white;
  padding: 14px;
  font-size: 14px;
  text-align: center;
}

td {
  padding: 14px;
  font-size: 13px;
  text-align: center;
  border-bottom: 1px solid #f0f0f0;
}

tr:hover td {
  background-color: #f5faff;
}

.hostname a {
  color: #1565c0;
  text-decoration: none;
  font-weight: bold;
}

.bar-container {
  width: 100%;
  background-color: #eee;
  border-radius: 6px;
  height: 10px;
  overflow: hidden;
  margin-top: 4px;
}

.bar {
  height: 100%;
  border-radius: 6px;
}

.cpu-bar { background-color: #ef5350; }
.mem-bar { background-color: #42a5f5; }
.disk-bar { background-color: #66bb6a; }

.label {
  margin-top: 4px;
  font-size: 12px;
  color: #555;
}

.badge {
  font-size: 11px;
  padding: 3px 8px;
  border-radius: 12px;
  display: inline-block;
```

```
      font-weight: 600;
      color: #fff;
    }

    .healthy { background-color: #43a047; }
    .warning { background-color: #fbc02d; color: #000; }
    .critical { background-color: #d32f2f; }

    .footer {
      text-align: center;
      font-size: 12px;
      margin-top: 24px;
      color: #666;
    }
  </style>
</head>
<body>
  <h2>📊 Consolidated VM Health Report</h2>

  <div class="summary">
    🗓️ <b>{{ collected_metrics | length }} VMs</b> | 🔥 Avg CPU: {{ ((collected_metrics |
map(attribute='cpu') | map('float') | sum | float) / (collected_metrics | length)) | round(2)
}}% | 🕹️ Avg Mem: {{ ((collected_metrics | map(attribute='mem') | map('float') | sum | float)
/ (collected_metrics | length)) | round(2) }}% | 📦 Avg Disk: {{ ((collected_metrics |
map(attribute='disk') | map('float') | sum | float) / (collected_metrics | length)) |
round(2) }}%
  </div>

  <table>
    <tr>
      <th>🌐 Hostname</th>
      <th>🔥 CPU Usage</th>
      <th>🕹️ Memory Usage</th>
      <th>📦 Disk Usage</th>
    </tr>
    {% for vm in collected_metrics %}
    <tr>
      <td class="hostname">
        <a href="http://{{ vm.hostname }}" target="_blank">{{ vm.hostname }}</a>
      </td>
      <td>
        {{ vm.cpu }}%
        <div class="bar-container"><div class="bar cpu-bar" style="width: {{ vm.cpu
}}%"></div></div>
        {% if vm.cpu|float < 50 %}
          <div class="badge healthy">Healthy</div>
          <div class="label">Low</div>
        {% elif vm.cpu|float < 80 %}
          <div class="badge warning">Warning</div>
          <div class="label">Moderate</div>
        {% else %}
          <div class="badge critical">Critical</div>
          <div class="label">High</div>
        {% endif %}
      </td>
      <td>
        {{ vm.mem }}%
        <div class="bar-container"><div class="bar mem-bar" style="width: {{ vm.mem
}}%"></div></div>
        {% if vm.mem|float < 50 %}<div class="label">Low</div>
        {% elif vm.mem|float < 80 %}<div class="label">Moderate</div>
        {% else %}<div class="label">High</div>{% endif %}
      </td>
      <td>
        {{ vm.disk }}%
```

```
        <div class="bar-container"><div class="bar disk-bar" style="width: {{ vm.disk
}}%"></div></div>
        {% if vm.disk|float < 50 %}<div class="label">Ample</div>
        {% elif vm.disk|float < 80 %}<div class="label">Monitor</div>
        {% else %}<div class="label">Full</div>{% endif %}
      </td>
    </tr>
    {% endfor %}
  </table>

  <div class="footer">
    Report Generated on: {{ timestamp }}
  </div>
</body>
</html>
```

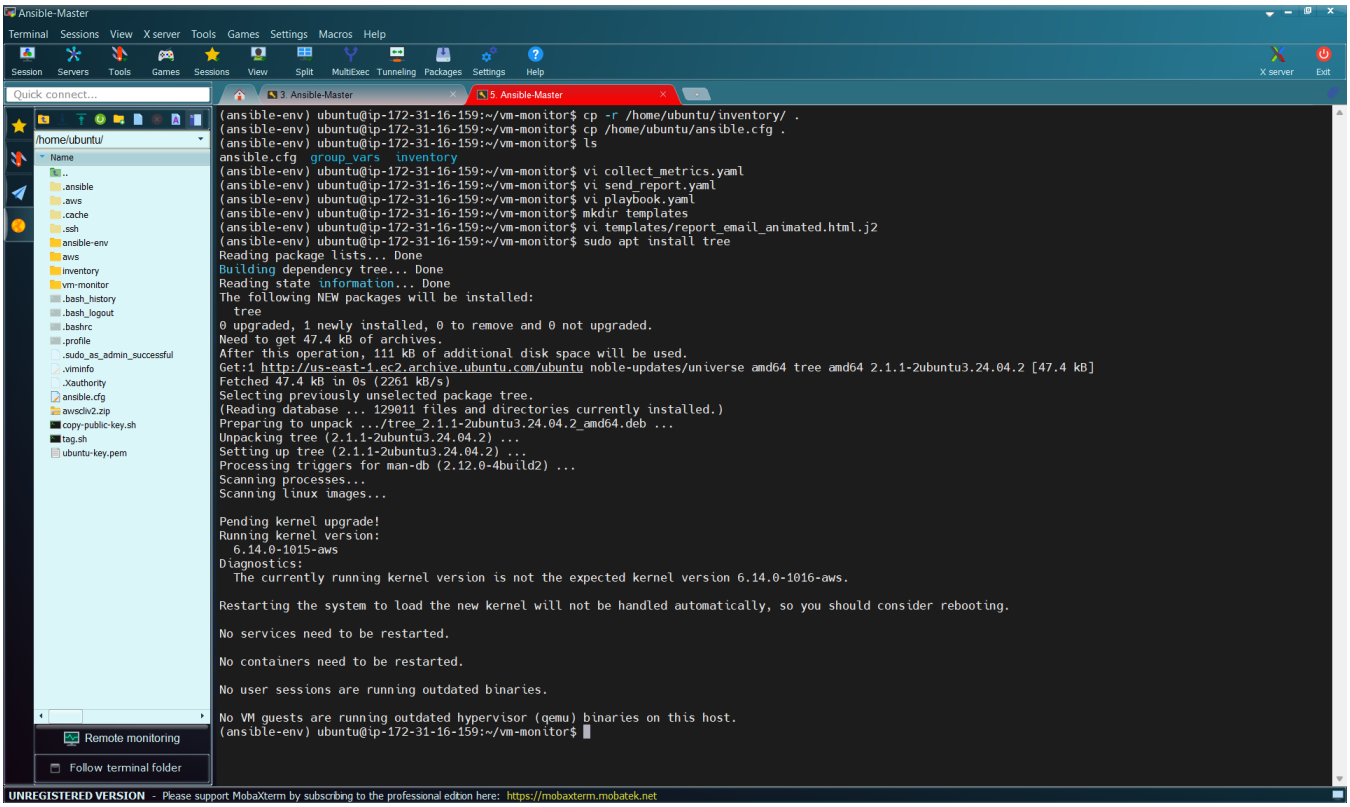


Save and exit by pressing “ESC” followed by :wq and press “ENTER”

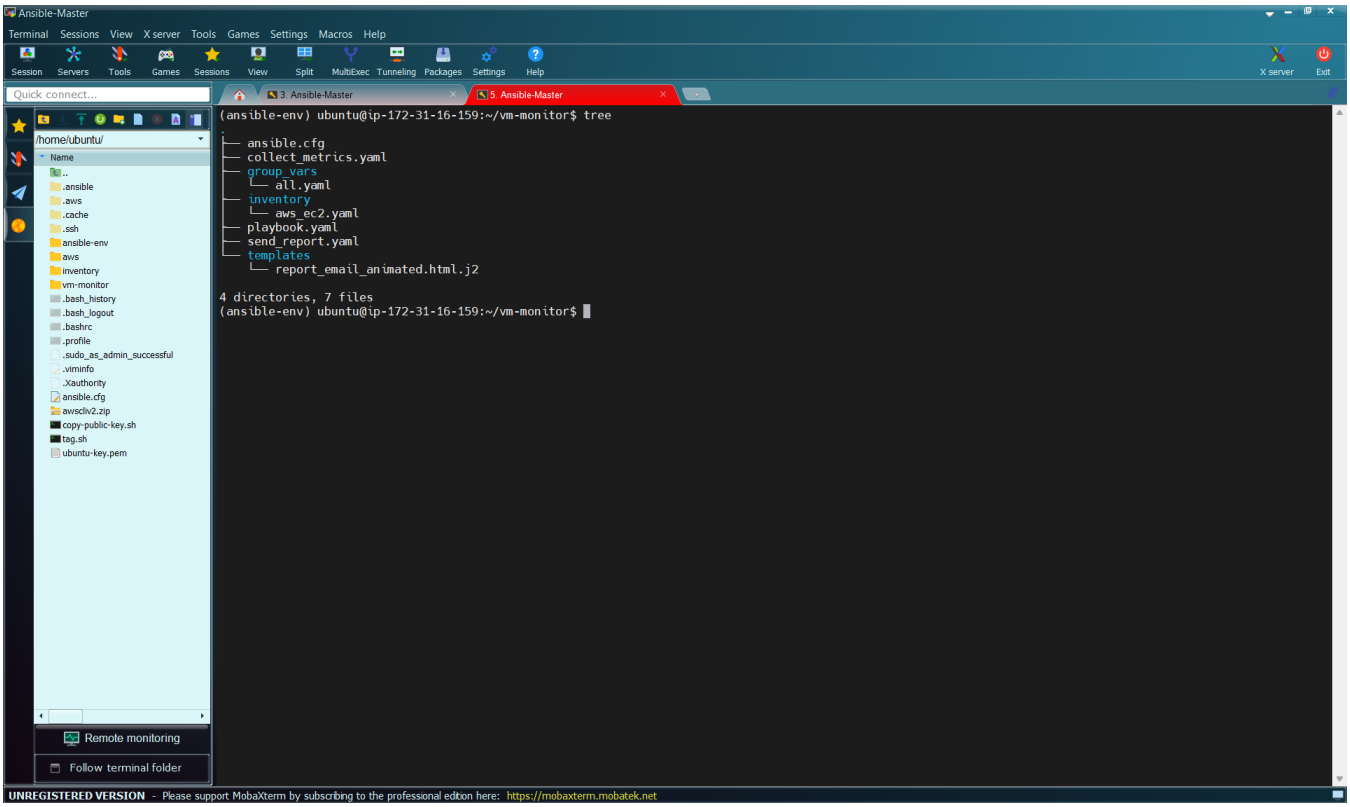


Now, let us check the structure of our project using the "**tree**" command. But before we do this, we have to first install tree using the command:

```
sudo apt install tree
```

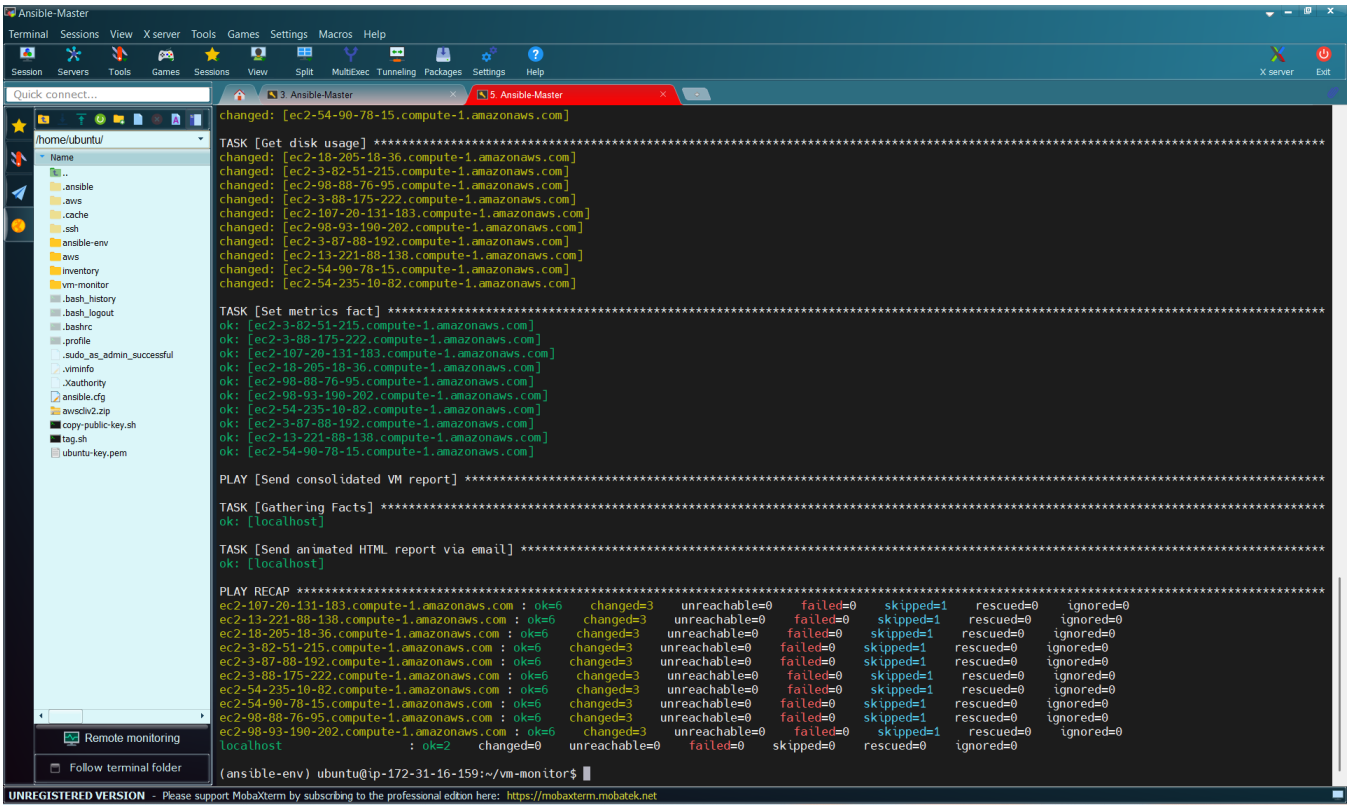


Then let us check the structure of the project folder using the command:

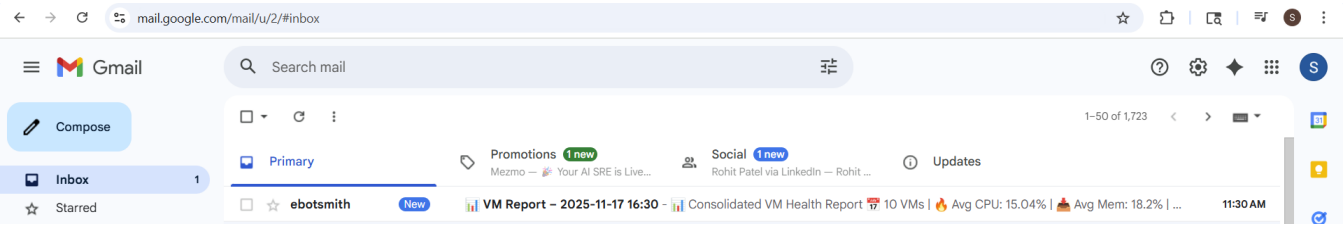


You can see that the structure is as expected. Now, we will run ansible by using the command:

`ansible-playbook playbook.yaml`

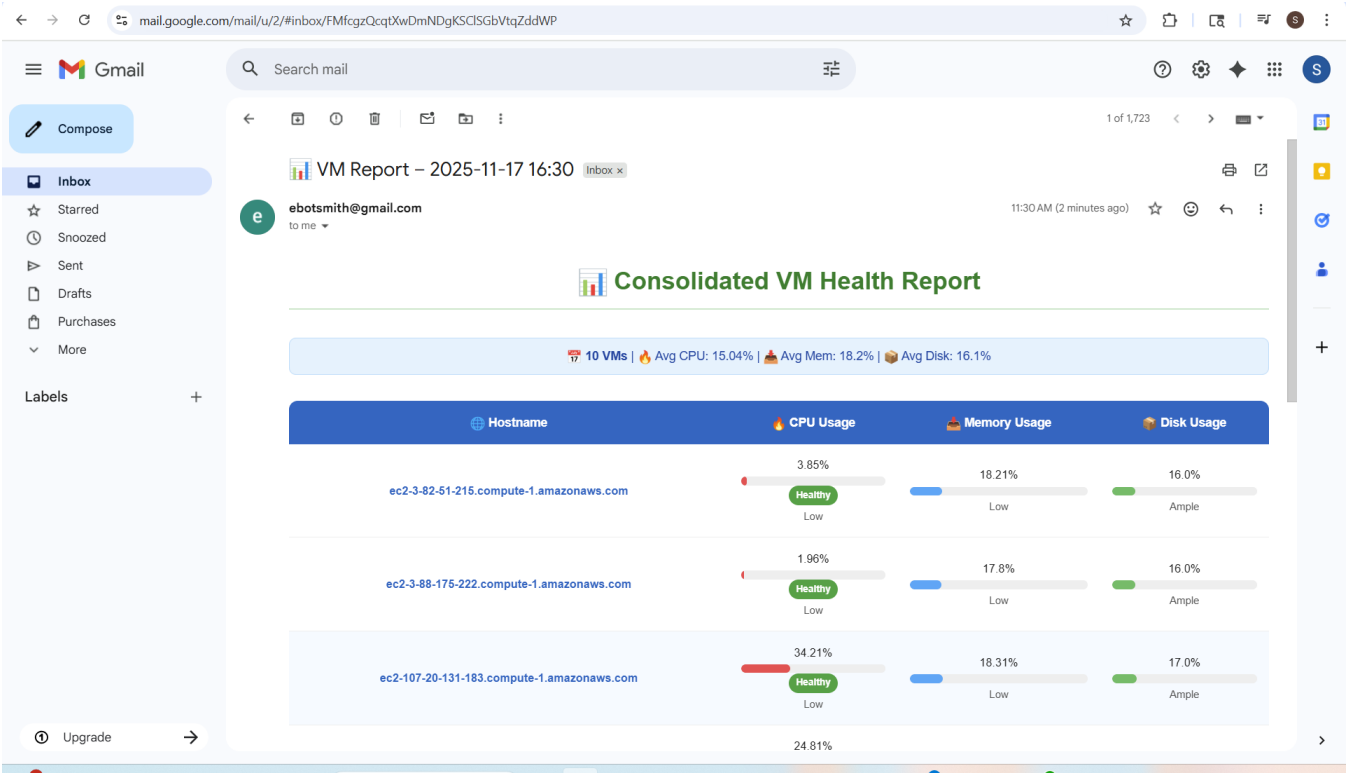


Now, let us go to the email "**ebotsidneysmith@gmail.com**" and see of the report has been sent.



You can see that the email has been sent from "**ebotsmith@gmail.com**"

Open the email



You can see the report showing the CPU usage, Memory usage and Disk usage.