

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.

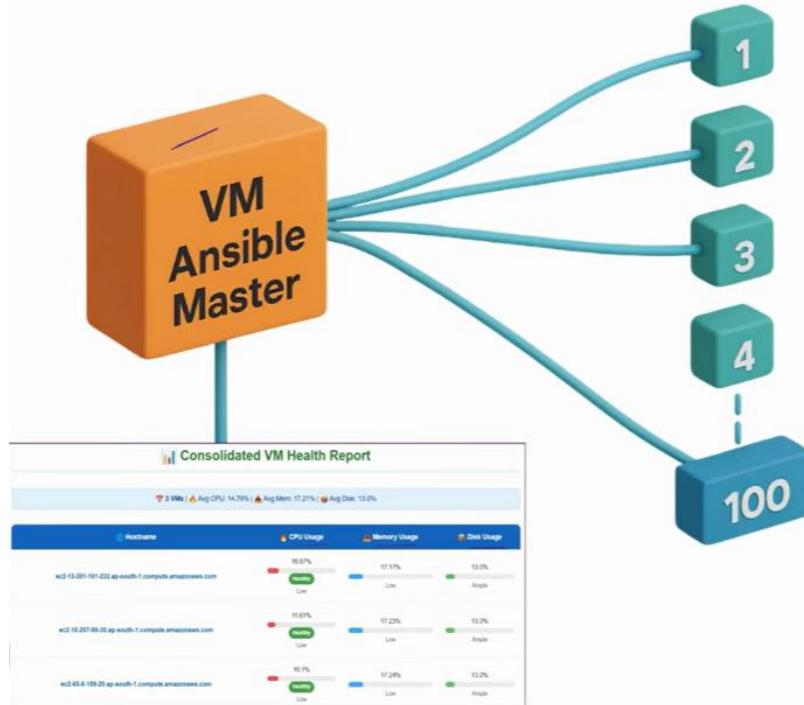
Let us assume a scenario that you are working in a company and you are supposed to manage 500 VMs. So, you are supposed to send the details of node metrics, like how much is CPU usage, how much is RAM usage and what is the space currently available on all those machines over email.

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.

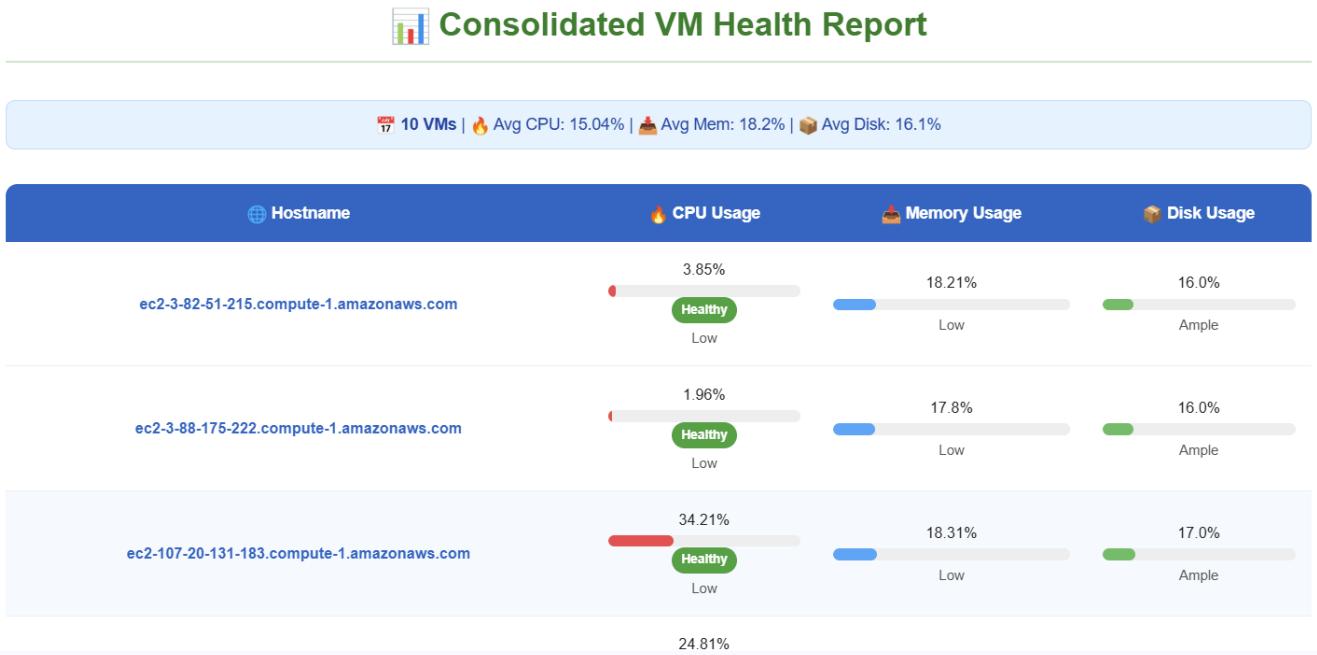


First, let us understand the use case. I have a use case that I have a total of 10 VMs. And I have been asked to share the information about node metric on the VM at any specific time.

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 serve 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

The screenshot shows the AWS Management Console Home page. At the top, there is a search bar with the URL <https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1#>. Below the search bar, the AWS logo is on the left, and the user account information "Account ID: 3247-8332-4460" and "sidney" are on the right. A red arrow points from the search bar down to the "EC2" icon in the "Recently visited" section. The "Recently visited" section also includes icons for CloudTrail, VPC, Elastic Kubernetes Service, Aurora and RDS, IAM, Elastic Container Registry, and Route 53. To the right of this section are sections for Applications (0), AWS Health, and Cost and usage. The Applications section shows a message: "Get started by creating an application." The AWS Health section shows "Open issues: 0" and "Past 7 days". The Cost and usage section shows "Current month: Access denied" and "Forecasted month end: Access denied". At the bottom of the page, there are links for CloudShell, Feedback, and various legal notices.

Search for “EC2”

The screenshot shows the AWS CloudWatch console home page. On the left, there's a sidebar with links to Services, Features, Resources, Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main content area has sections for Services (EC2, EC2 Image Builder, Recycle Bin), Features (Dashboard, EC2 Instances, AMIs), and Resources in us-east-1. A callout arrow points from the text "Click on ‘EC2’" to the "EC2" link in the Services section. The top right corner shows account information (Account ID: 3247-8332-4460) and navigation links.

Click on “EC2”

The screenshot shows the Amazon Elastic Compute Cloud (EC2) landing page. The left sidebar includes links for EC2 (Dashboard, EC2 Global View, Events), Instances (Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager), Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), and Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs). The main content area features the title "Amazon Elastic Compute Cloud (EC2)" and the subtitle "Create, manage, and monitor virtual servers in the cloud." A callout arrow points from the text "Click on ‘Launch Instance’" to the "Launch instance" button in the "Launch a virtual server" callout box. Other buttons in this box include "View dashboard", "Get started walkthrough", and "Get started tutorial". The bottom right corner shows account information (Account ID: 3247-8332-4460) and navigation links.

Click on “Launch Instance”

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags [Info](#)

Name
e.g. My Web Server

Add additional tags

Application and OS Images (Amazon Machine Image) [Info](#)

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Search our full catalog including 1000s of application and OS images

Recents Quick Start

Amazon Linux	macOS	Ubuntu	Windows	Red Hat	SUSE Linux	Debian

Amazon Machine Image (AMI)

Amazon Linux 2023 kernel-6.1 AMI
ami-0cae6d6fe6048ca2c (64-bit (x86), uefi-preferred) / ami-023c74ebb7125ab4 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Description
Amazon Linux 2023 (kernel-6.1) is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Summary

Number of instances [Info](#)
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.9.2... [read more](#)
ami-0cae6d6fe6048ca2c

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs. 750 hours per month of public IPv4 address usage, 30 GB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet. Data transfer charges are not included as part of the free tier allowance.

Cancel [Launch instance](#) [Preview code](#)

We will give the instance the name “Ansible-Master”

EC2 > Instances > Launch an instance

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags [Info](#)

Name
Ansible-Master

Add additional tags

On “AMI”, select “Ubuntu”

▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Search our full catalog including 1000s of application and OS images

Recents Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux Debian

ubuntu® Microsoft Red Hat SUSE debian

[Browse more AMIs](#)
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type Free tier eligible ▾

ami-0ecb62995f68bb549 (64-bit (x86)) / ami-01b9f1e7dc427266e (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description
Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Canonical, Ubuntu, 24.04, amd64 noble image

Architecture	AMI ID	Publish Date	Username	Verified provider
64-bit (x86) ▾	ami-0ecb62995f68bb549	2025-10-22	ubuntu	

Scroll down to “**Instance Type**” and select “**t2.medium**”

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.medium
Family: t2 2 vCPU 4 GiB Memory Current generation: true
On-Demand Ubuntu Pro base pricing: 0.0499 USD per Hour On-Demand Linux base pricing: 0.0464 USD per Hour
On-Demand RHEL base pricing: 0.0752 USD per Hour On-Demand Windows base pricing: 0.0644 USD per Hour
On-Demand SUSE base pricing: 0.1464 USD per Hour

All generations
[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

Scroll down to “**Key Pair**”

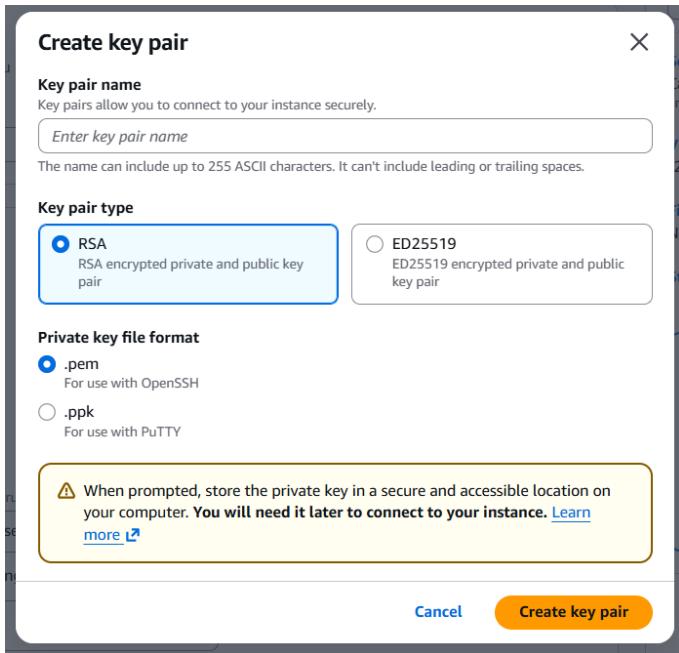
▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

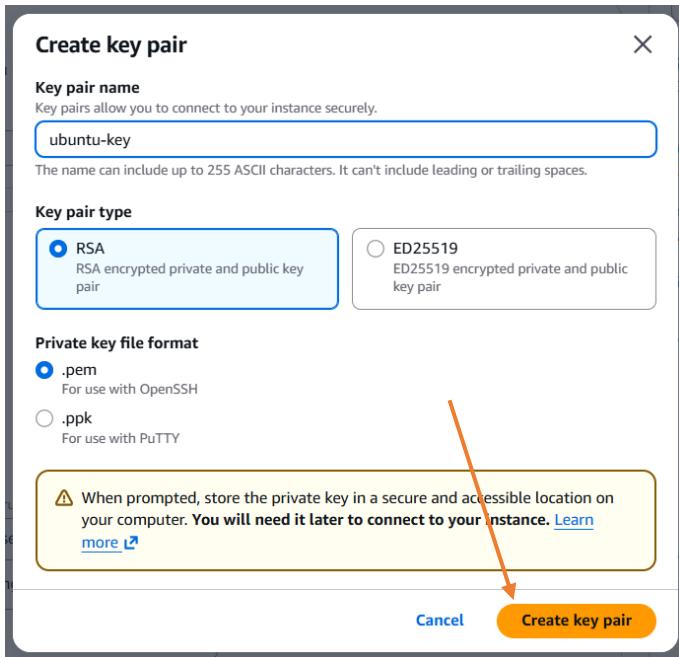
Key pair name - required

Select Create new 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”

The screenshot shows the AWS Lambda console with the 'Key pair (login)' dropdown set to 'ubuntu-key'. Below it, there is a note about using a key pair for secure connection and a 'Create new key pair' button.

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”

▼ Network settings [Info](#) [Edit](#)

Network | [Info](#)
vpc-0dc82031d037c36d2 | Default VPC

Subnet | [Info](#)
No preference (Default subnet in any availability zone)

Auto-assign public IP | [Info](#)
Enable
Additional charges apply when outside of free tier allowance

Firewall (security groups) | [Info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group Select existing security group

We'll create a new security group called 'launch-wizard-1' with the following rules:

Allow SSH traffic from Anywhere
Helps you connect to your instance

Allow HTTPS traffic from the internet ▾
To set up an endpoint, for example when creating a web server

Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

⚠️ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only. [X](#)

Then scroll down to “Configure Storage” and make the value “20GiB”

▼ Configure storage [Info](#) [Advanced](#)

1x 20 GiB gp3 ▾ Root volume, 3000 IOPS, Not encrypted

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage [X](#)

[Add new volume](#)

The selected AMI contains instance store volumes, however the instance does not allow any instance store volumes. None of the instance store volumes from the AMI will be accessible from the instance

[Click refresh to view backup information](#) [C](#)
The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

0 x File systems [Edit](#)

Advanced details [Info](#)

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 20 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GB of bandwidth to the internet. Data transfer charges are not included as part of the free tier allowance. [X](#)

[Cancel](#) [Launch instance](#) [Preview code](#)

Then click on “Launch Instance”

The screenshot shows the AWS EC2 Instances launch page. At the top, there is a green success banner stating "Successfully initiated launch of instance (i-0792eb33eba9a1ea1)". Below the banner, there is a "Launch log" link. The main area is titled "Next Steps" and contains several cards:

- Create billing and free tier usage alerts**: To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds. Includes a "Create billing alerts" button.
- Connect to your instance**: Once your instance is running, log into it from your local computer. Includes a "Connect to instance" button and a "Learn more" link.
- Connect an RDS database**: Configure the connection between an EC2 instance and a database to allow traffic flow between them. Includes a "Connect an RDS database" button and a "Create a new RDS database" link.
- Create EBS snapshot policy**: Create a policy that automates the creation, retention, and deletion of EBS snapshots. Includes a "Create EBS snapshot policy" button.
- Manage detailed monitoring**: Enable or disable detailed monitoring for the instance. If you enable detailed monitoring, the Amazon EC2 console displays monitoring graphs with a 1-minute period. Includes a "Manage detailed monitoring" button.
- Create Load Balancer**: Create a application, network gateway or classic Elastic Load Balancer. Includes a "Create Load Balancer" button.
- Create AWS budget**: AWS Budgets allows you to create budgets, forecast spend, and take action on your costs and usage from a single location. Includes a "Create AWS budget" button.
- Manage CloudWatch alarms**: Create or update Amazon CloudWatch alarms for the instance. Includes a "Manage CloudWatch alarms" button.

At the bottom of the page, there are links for "CloudShell", "Feedback", and copyright information: "© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

Click on “Instances”

The screenshot shows the AWS EC2 Instances list page. On the left, there is a navigation sidebar with categories like EC2, Dashboard, EC2 Global View, Events, Instances, Images, Elastic Block Store, Network & Security, and more. The "Instances" section is expanded, showing sub-options like Instances, Instance Types, Launch Templates, etc.

The main area displays a table titled "Instances (1) Info". The table has columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. One row is present, showing "Ansible-Master" with Instance ID "i-0792eb33eba9a1ea1", Instance state "Running", Instance type "t2.medium", Status check "Initializing", Alarm status "View alarms +", and Availability Zone "us-east-1d".

Below the table, there is a section titled "Select an instance" with a dropdown menu. At the bottom of the page, there are links for "CloudShell", "Feedback", and copyright information: "© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

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 screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed, and the main area displays a table titled "Instances (1) Info". The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. One instance, "Ansible-Master" (Instance ID: i-0792eb33eba9a1ea1), is listed. It is currently running on an t2.medium instance type and has passed 2/2 checks. The page also includes a search bar at the top and navigation buttons like "Connect", "Actions", and "Launch instances". A message "Select an instance" is displayed below the table.

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.

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with sections like Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, CloudShell, and Feedback. The main area displays a table for 'Instances (1) Info'. The single instance listed is 'Ansible-Master' with Instance ID i-0792eb33eba9a1ea1, running on a t2.medium type with 2/2 checks passed. A red arrow points to the 'Launch instances' button at the top right of the table header. The status bar at the bottom indicates '© 2025, Amazon Web Services, Inc. or its affiliates.' and includes links for Privacy, Terms, and Cookie preferences.

Click on “Launch Instance”

The screenshot shows the 'Launch an instance' wizard. It starts with the 'Name and tags' step, where you can enter a name like 'e.g. My Web Server' and add tags. Next is the 'Application and OS Images (Amazon Machine Image)' step, which allows you to search for AMIs or browse recent ones. The 'Amazon Machine Image (AMI)' section shows details for 'Amazon Linux 2023 kernel-6.1 AMI' (ami-0cae6d6fe048ca2c). The 'Description' section provides a brief overview of Amazon Linux 2023. Finally, the 'Summary' step shows a summary of the configuration: 1 instance, AMI 'Amazon Linux 2023 AMI 2023.9.2...', Virtual server type 't2.micro', and Storage '1 volume(s) - 8 GiB'. A callout box highlights the 'Free tier' information: 'In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GB of bandwidth to the internet. Data transfer charges are not included as part of the free tier allowance.' At the bottom, there are 'Cancel', 'Launch instance', and 'Preview code' buttons.

Give the instance a name, I will call it “web”

The screenshot shows the 'Launch an instance' wizard. In the 'Name and tags' section, the 'Name' field contains 'web'. An orange arrow points from the text 'We have to add tags, click on "Add additional tags"' to the 'Add additional tags' link located to the right of the name field.

We have to add tags, click on “Add additional tags”

The screenshot shows the 'Launch an instance' wizard. In the 'Name and tags' section, there is one tag: 'Name' with value 'web'. An orange arrow points from the text 'Click on "Add new tag"' to the 'Add new tag' button.

Click on “Add new tag”

The screenshot shows the 'Launch an instance' wizard. In the 'Name and tags' section, there is one tag: 'Name' with value 'web'. Below it, a new tag is being added with 'Key' set to 'Enter key' and 'Value' set to 'Enter value'. An orange arrow points from the text 'In the “Key”, enter “Environment”' to the 'Enter key' input field.

In the “Key”, enter “Environment”

☰ EC2 > Instances > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

▼ Name and tags Info

Key Info <input type="text" value="Name"/> X	Value Info <input type="text" value="web"/> X	Resource types Info <input type="text" value="Select resource types"/> ▼ Remove <input type="text" value="Instances"/> X
Key Info <input type="text" value="Environment"/> X	Value Info <input type="text" value="Enter value"/> X	Resource types Info <input type="text" value="Select resource types"/> ▼ Remove <input type="text" value="Instances"/> X

[Add new tag](#)

You can add up to 48 more tags.

For “Value”, enter “dev”

☰ EC2 > Instances > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

▼ Name and tags Info

Key Info <input type="text" value="Name"/> X	Value Info <input type="text" value="web"/> X	Resource types Info <input type="text" value="Select resource types"/> ▼ Remove <input type="text" value="Instances"/> X
Key Info <input type="text" value="Environment"/> X	Value Info <input type="text" value="dev"/> X	Resource types Info <input type="text" value="Select resource types"/> ▼ Remove <input type="text" value="Instances"/> X

[Add new tag](#)

You can add up to 48 more tags.

Then scroll down to “AMI” and select “Ubuntu”

▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Search our full catalog including 1000s of application and OS images

Recents Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux Debian

ubuntu® Microsoft Red Hat SUSE debian

Search Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type Free tier eligible

ami-0ecb62995f68bb549 (64-bit (x86)) / ami-01b9f1e7dc427266e (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description
Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Canonical, Ubuntu, 24.04, amd64 noble image

Architecture	AMI ID	Publish Date	Username	Verified provider
64-bit (x86) ▾	ami-0ecb62995f68bb549	2025-10-22	ubuntu	Verified provider

Then scroll down to “Instance Type” and select “t2.small”

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.small
Family: t2 1 vCPU 2 GiB Memory Current generation: true On-Demand Windows base pricing: 0.032 USD per Hour
On-Demand Linux base pricing: 0.023 USD per Hour On-Demand RHEL base pricing: 0.0376 USD per Hour
On-Demand SUSE base pricing: 0.053 USD per Hour On-Demand Ubuntu Pro base pricing: 0.025 USD per Hour

All generations [Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

Scroll down to “Key Pair” and select the Key Pair we created before

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

ubuntu-key [Create new key pair](#)

Scroll down to “Network Settings”. We will use the default VPC and the default Security Group.

▼ Network settings [Info](#) [Edit](#)

Network | [Info](#)
vpc-0dc82031d037c36d2 | Default VPC

Subnet | [Info](#)
No preference (Default subnet in any availability zone)

Auto-assign public IP | [Info](#)
Enable
Additional charges apply when outside of free tier allowance

Firewall (security groups) | [Info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group Select existing security group

Common security groups | [Info](#)
[Select security groups](#) ▾

default sg-0e28f6a5cd620956c [X](#)
VPC: vpc-0dc82031d037c36d2

Compare security group rules ↻

Security groups that you add or remove here will be added to or removed from all your network interfaces.

Then scroll down to “Configure Storage” and make it “15GiB”

▼ Configure storage [Info](#) [Advanced](#)

1x GiB Root volume, 3000 IOPS, Not encrypted

(i) Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage X

[Add new volume](#)

The selected AMI contains instance store volumes, however the instance does not allow any instance store volumes. None of the instance store volumes from the AMI will be accessible from the instance

⌚ Click refresh to view backup information ↻
The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

0 x File systems [Edit](#)

► Advanced details [Info](#)

Then go to “Summary” and make the number of instances “10”

The screenshot shows the 'Summary' step of the AWS EC2 'Launch instance' wizard. It includes the following configuration details:

- Number of instances:** 10
- Software Image (AMI):** Canonical, Ubuntu, 24.04, amd64... [read more](#)
- Virtual server type (instance type):** t2.small
- Firewall (security group):** default
- Storage (volumes):** 1 volume(s) - 15 GiB

A callout box highlights the **Free tier** information:

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet. Data transfer charges are not included as part of the free tier allowance.

At the bottom are three buttons: **Cancel**, **Launch instance** (highlighted with an orange arrow), and **Preview code**.

Then, click on “Launch Instance”

The screenshot shows the AWS EC2 Instances page. A success message at the top states: "Successfully initiated launch of instances i-0988dc71224fca404, i-0ac654f02b182d3ff, i-0d57de98a77535e87, i-0a277fb05012cae6, i-073c7e70d7d8d4b9b, i-060f22ec669e8becc, i-0fbdae2846af8febe, i-08f12aa295a17ac53, i-018bd2d43c8aef2cd, i-016fac5241458d6". An orange arrow points from the text "Then, click on ‘Launch Instance’" to the **Launch instance** button in the previous screenshot.

The page also displays several "Next Steps" options:

- Create billing and free tier usage alerts**: To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds. [Create billing alerts](#)
- Connect to your instance**: Once your instance is running, log into it from your local computer. [Learn more](#)
- Connect an RDS database**: Configure the connection between an EC2 instance and a database to allow traffic flow between them. [Connect an RDS database](#)
- Create EBS snapshot policy**: Create a policy that automates the creation, retention, and deletion of EBS snapshots. [Create EBS snapshot policy](#)
- Manage detailed monitoring**: Enable or disable detailed monitoring for the instance. If you enable detailed monitoring, the Amazon EC2 console displays monitoring graphs with a 1-minute period. [Manage detailed monitoring](#)
- Create Load Balancer**: Create a application, network gateway or classic Elastic Load Balancer. [Create Load Balancer](#)
- Create AWS budget**: AWS Budgets allows you to create budgets, forecast spend, and take action on your costs and usage from a single location. [Create AWS budget](#)
- Manage CloudWatch alarms**: Create or update Amazon CloudWatch alarms for the instance. [Manage CloudWatch alarms](#)

At the bottom, there are links for CloudShell, Feedback, and a footer with copyright information: © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

Click on “Instances”

The screenshot shows the AWS EC2 Instances page with 11 instances listed. All instances are in the 'initializing' state, indicated by a blue circle with a white question mark icon. The status check column for the last instance (i-0792eb33eba9a1ea1) shows '2/2 checks passed'. The 'Availability Zone' column shows 'us-east-1d' for all instances.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
web	i-0ac654f02b182d3ff	Running	t2.small	Initializing	View alarms +	us-east-1d
web	i-0a277fb05012cae6	Running	t2.small	Initializing	View alarms +	us-east-1d
web	i-073c7e70d78d4b9b	Running	t2.small	Initializing	View alarms +	us-east-1d
web	i-060f22ec669e8bedc	Running	t2.small	Initializing	View alarms +	us-east-1d
web	i-0fbdae2846af8febe	Running	t2.small	Initializing	View alarms +	us-east-1d
web	i-08f12aa295a17ac53	Running	t2.small	Initializing	View alarms +	us-east-1d
web	i-018b2d43c8aeff3cd	Running	t2.small	Initializing	View alarms +	us-east-1d
web	i-016cfacc5241458d6	Running	t2.small	Initializing	View alarms +	us-east-1d
web	i-0988dc71224fca404	Running	t2.small	Initializing	View alarms +	us-east-1d
web	i-0d57de98a77535e87	Running	t2.small	Initializing	View alarms +	us-east-1d
Ansible-Master	i-0792eb33eba9a1ea1	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1d

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

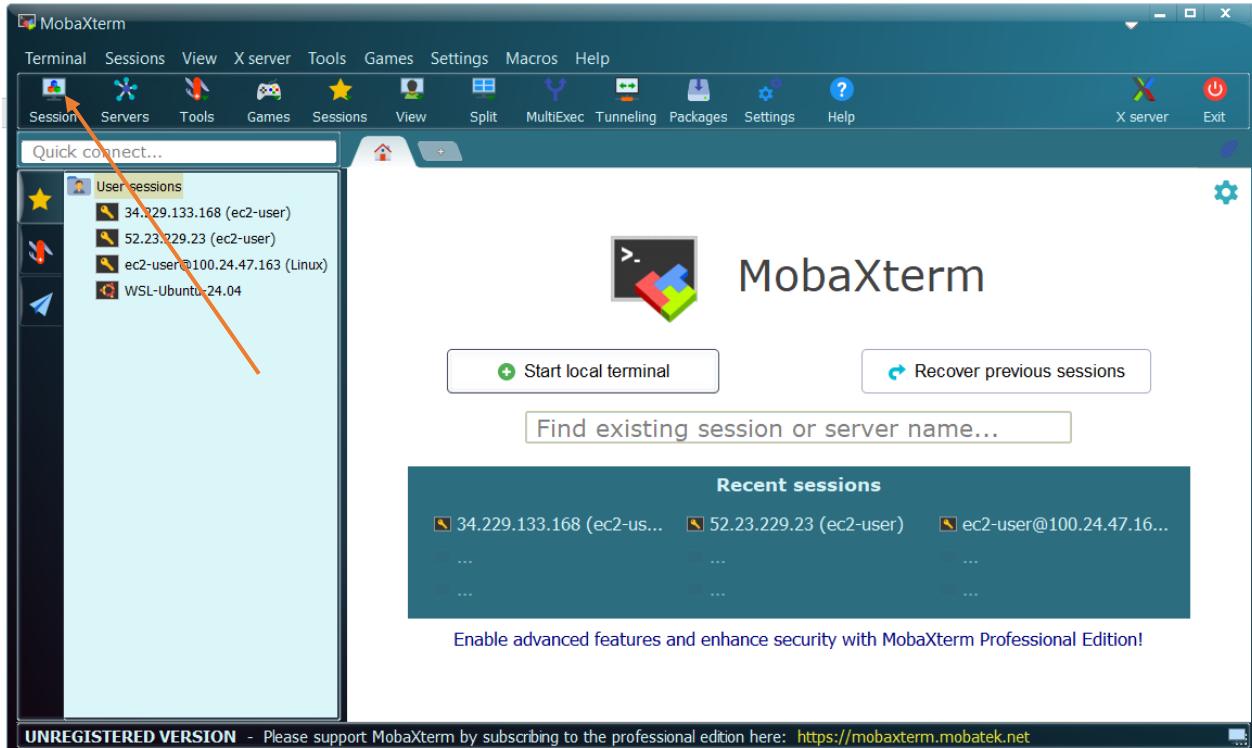
The screenshot shows the AWS EC2 Instances page with the same 11 instances. Now, all instances have passed their status checks, indicated by a green circle with a white checkmark icon. The 'Availability Zone' column shows 'us-east-1d' for all instances.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
web	i-0ac654f02b182d3ff	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web	i-0a277fb05012cae6	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web	i-073c7e70d78d4b9b	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web	i-060f22ec669e8bedc	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web	i-0fbdae2846af8febe	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web	i-08f12aa295a17ac53	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web	i-018b2d43c8aeff3cd	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web	i-016cfacc5241458d6	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web	i-0988dc71224fca404	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web	i-0d57de98a77535e87	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
Ansible-Master	i-0792eb33eba9a1ea1	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1d

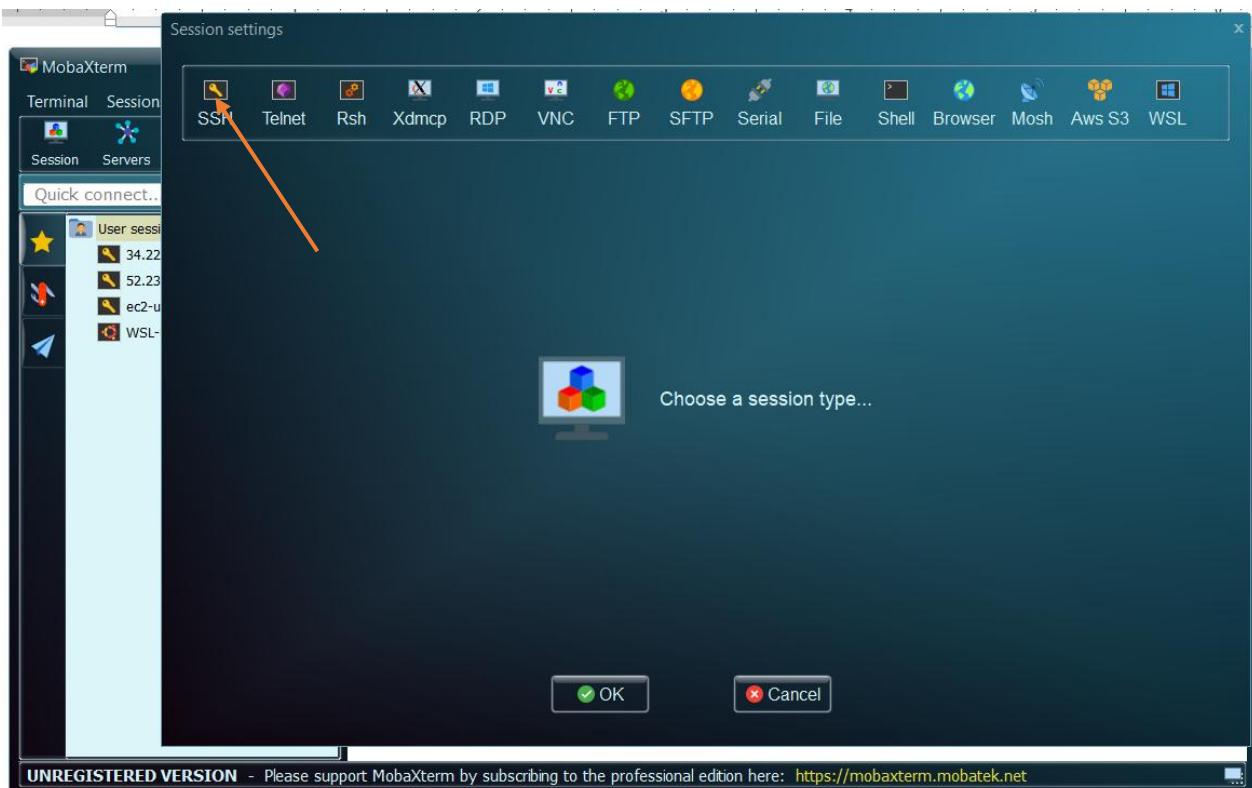
They have now passed the “2/2 check”. Each one of the EC2 instance is named “web”. This is not good. So, we are going to use a script to name them **web01, web03,....., web10**, so that we can differentiate between them.

Part 3: SSH Connect to the Master EC2 Instance

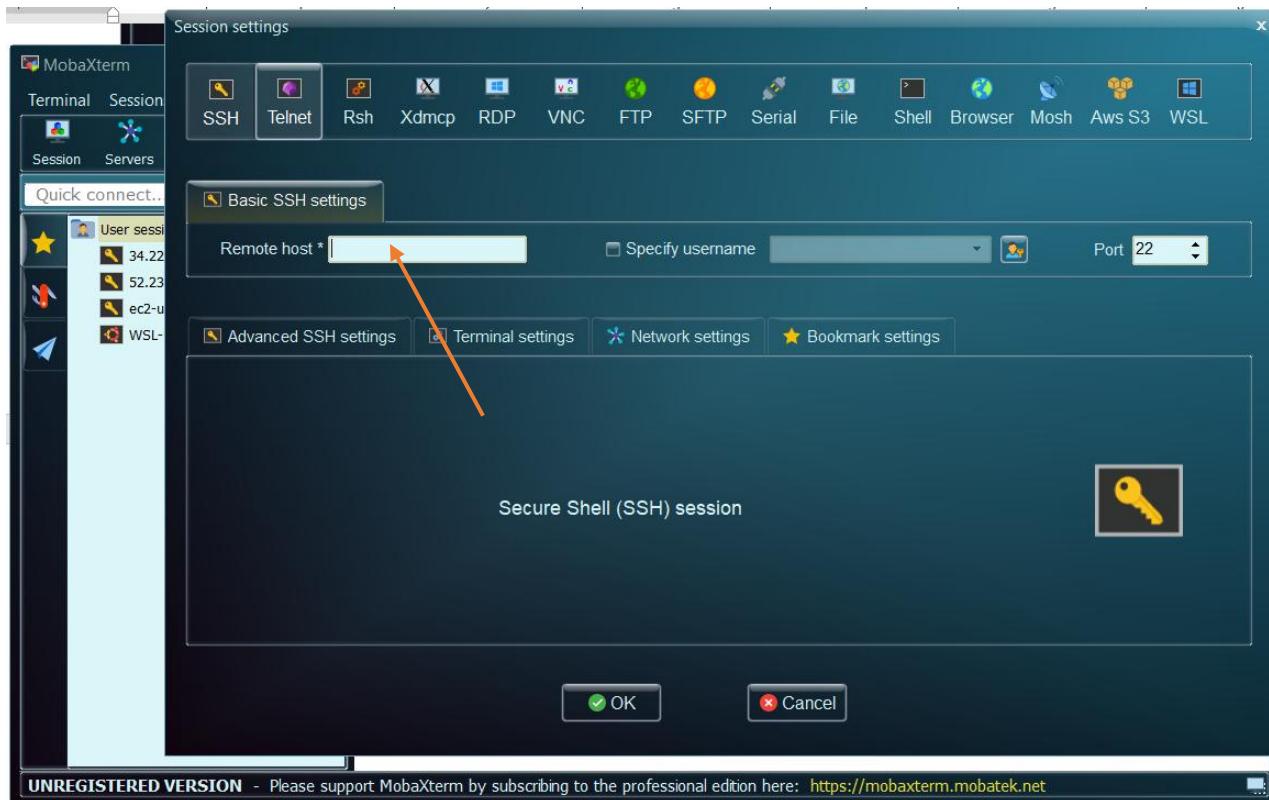
We will now SSH connect to the “Ansible-Master” instance using MobaXterm. Open MobaXterm.



Click on “Session”



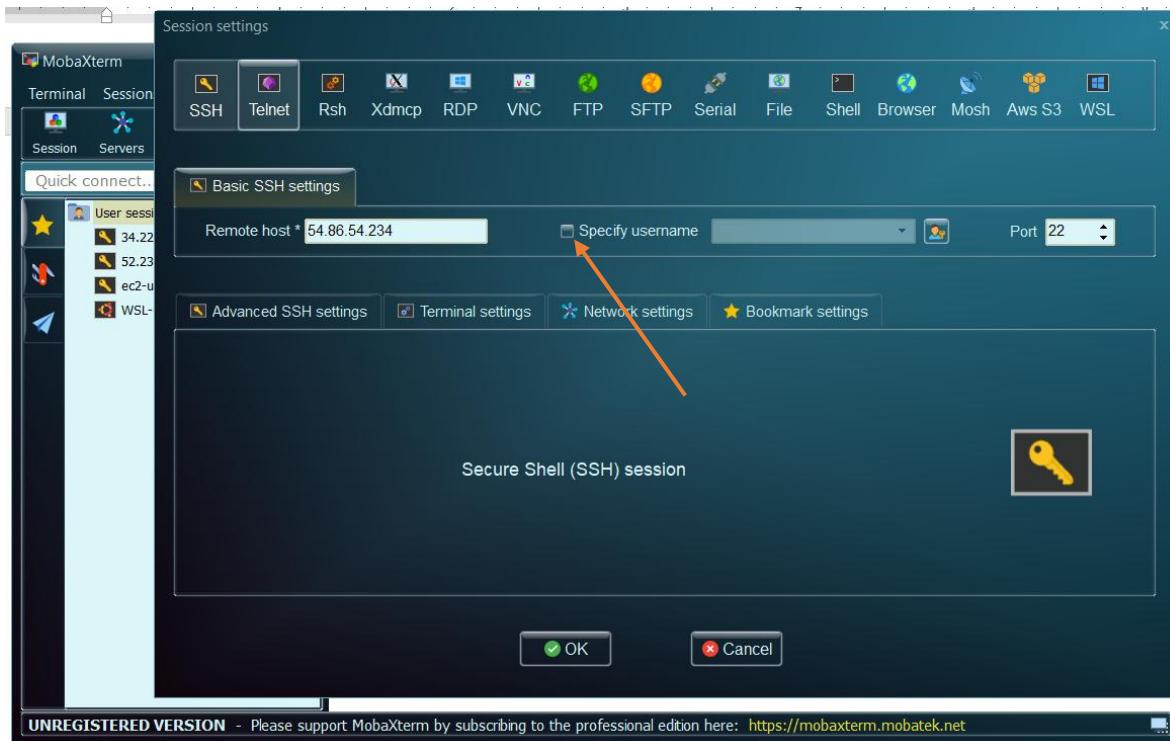
Click on “SSH”



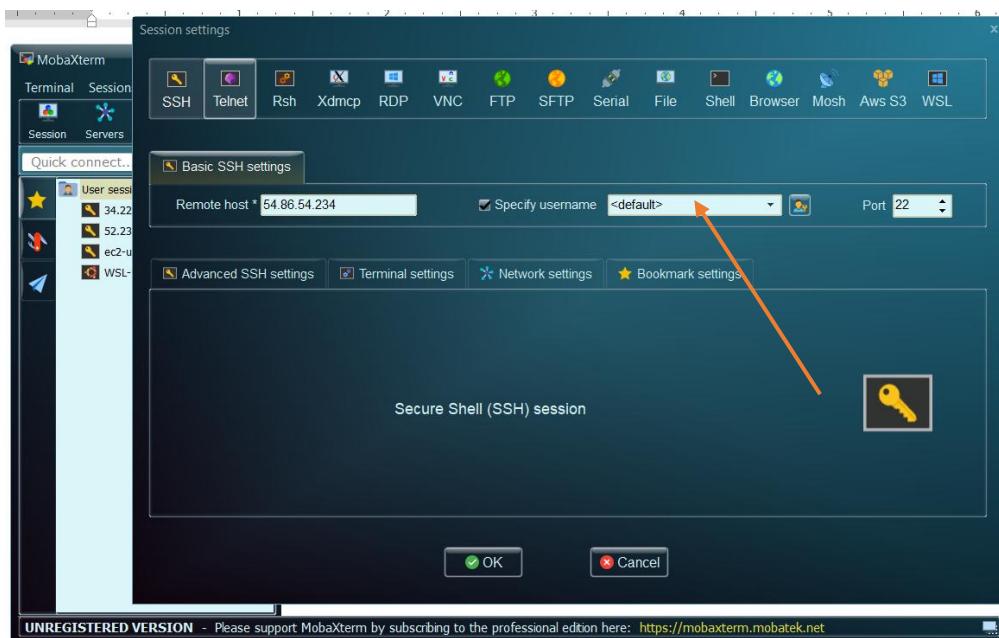
Go to our “Ansible-Master” instance

The screenshot shows the AWS EC2 Instances page. The left sidebar shows navigation options like Dashboard, EC2 Global View, Events, Instances, Images, Elastic Block Store, Network & Security, and CloudShell. The main area displays a table of instances with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. One instance, 'Ansible-Master' (Instance ID: i-0792eb33eba9a1ea1), is selected. An orange arrow points from the 'Public IPv4 address' field in the instance details to the 'Remote host' field in the MobaXterm dialog above. The instance details also show Public DNS: ec2-54-86-54-234.compute-1.amazonaws.com.

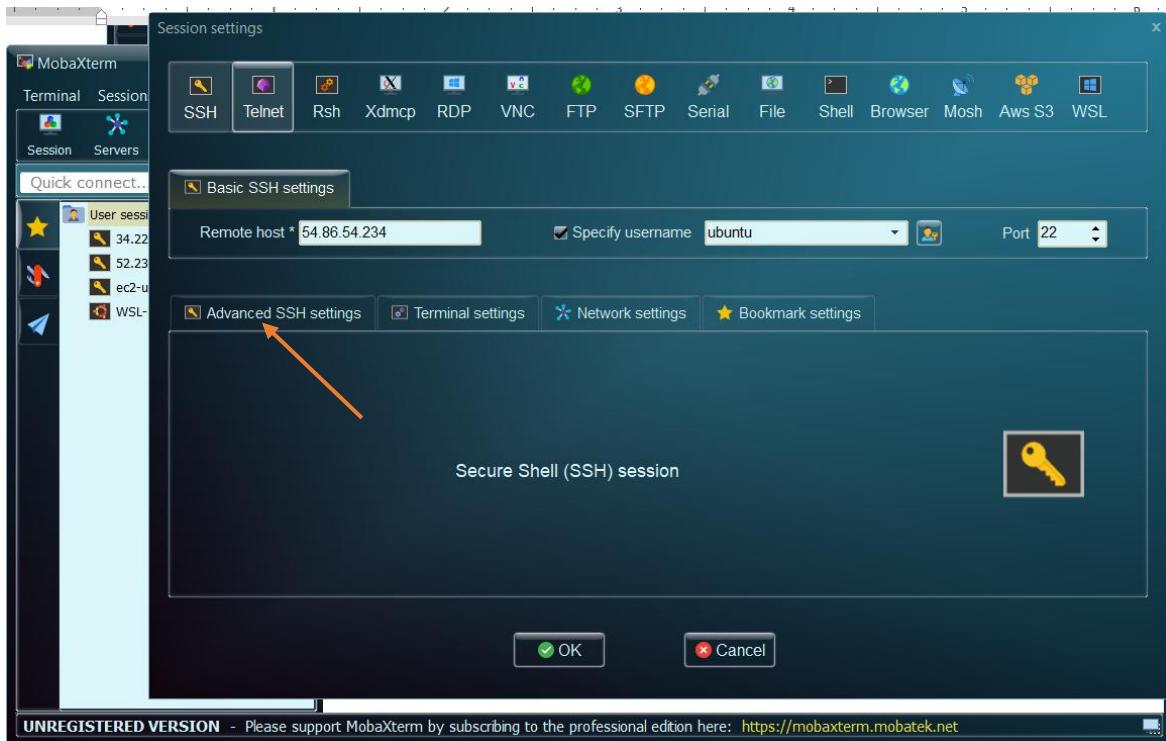
Copy the “Public IPv4 address” and paste on “Remote Host” on MobaXterm



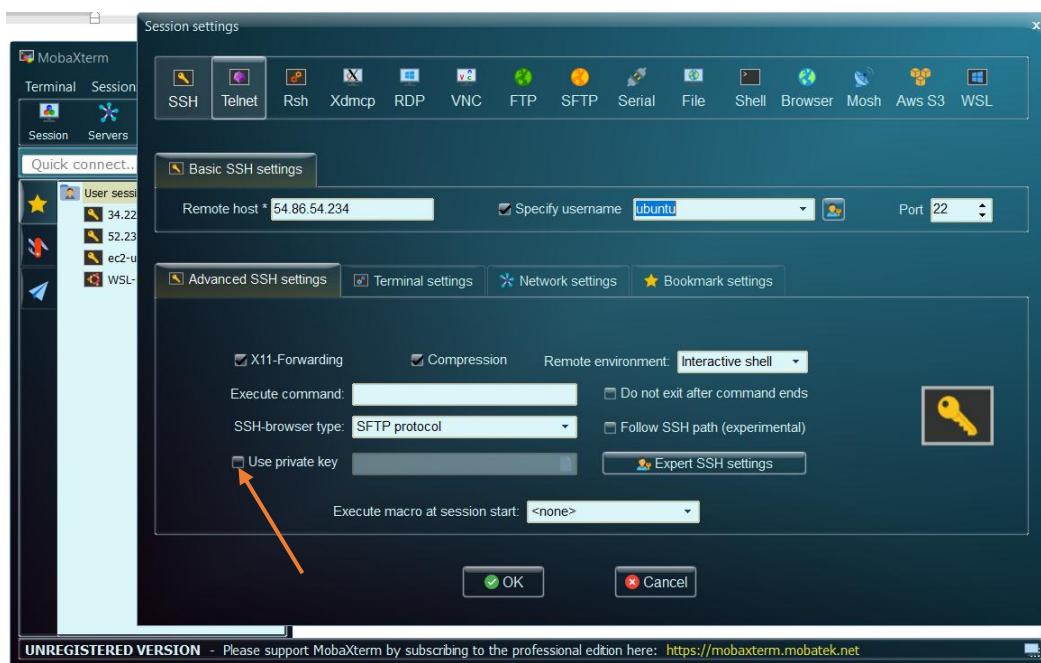
Select “Specify username”



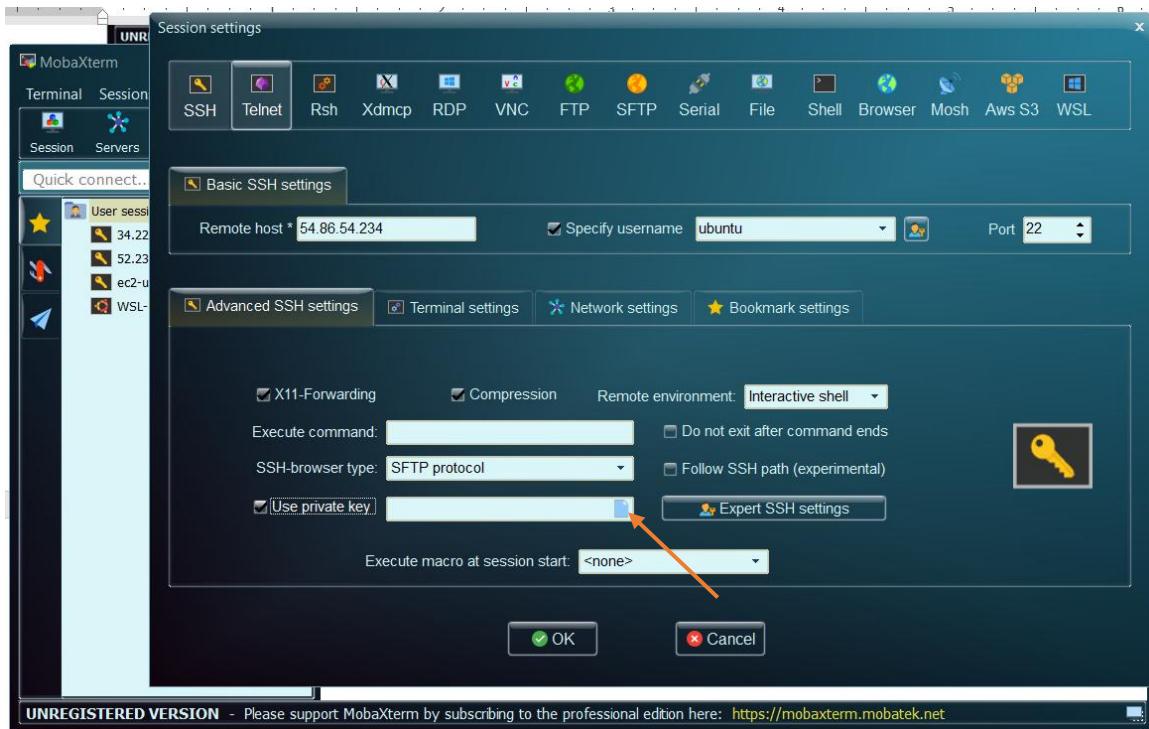
Then, enter “ubuntu” for username



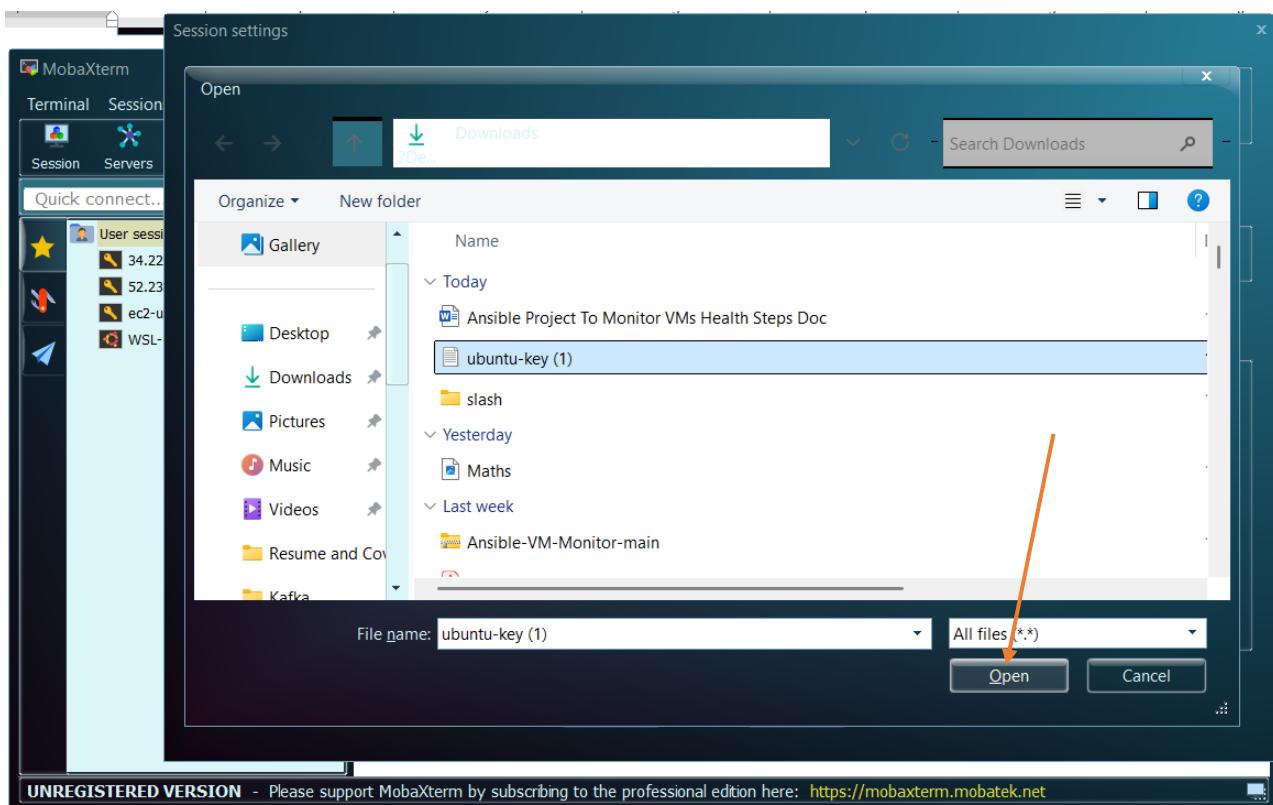
Then click on “Advanced SSH Settings”



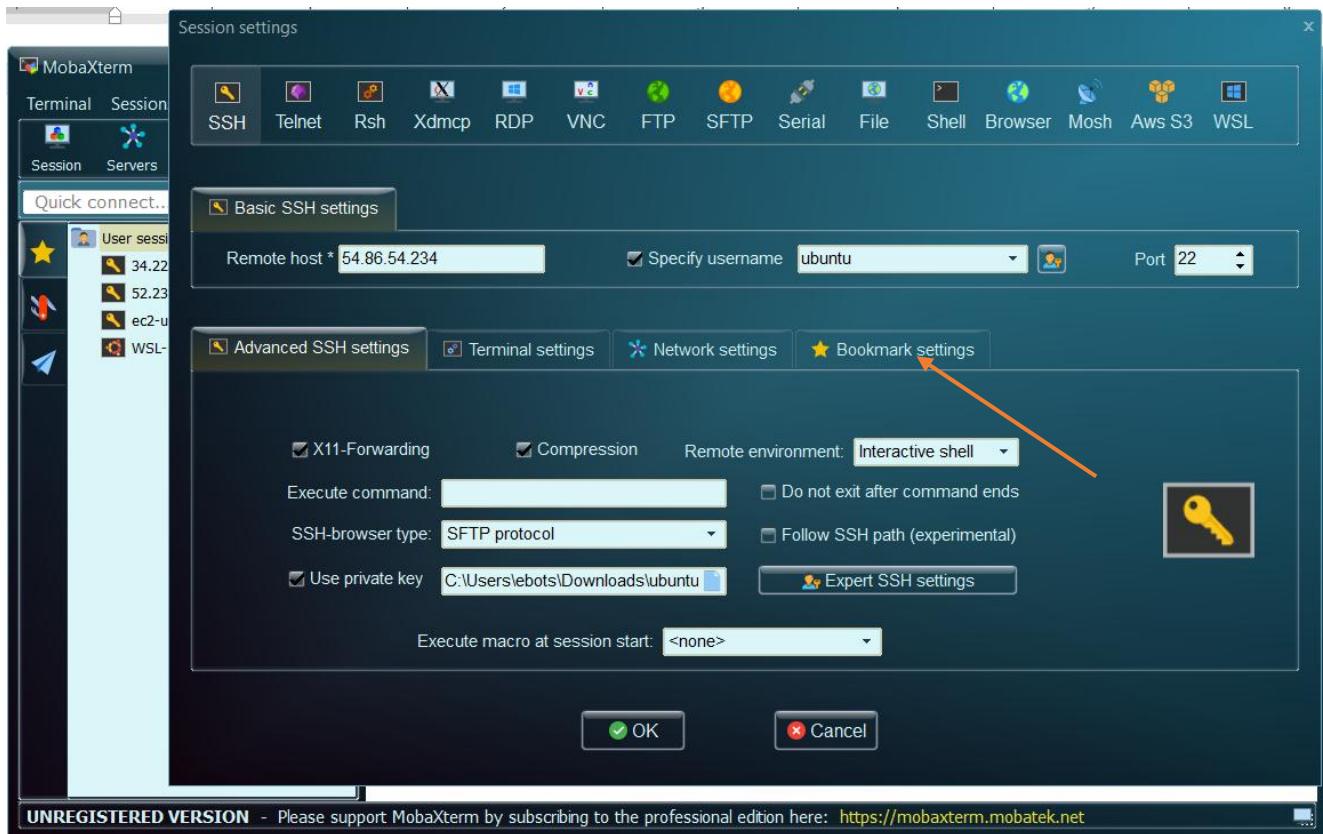
Check “Use private key”



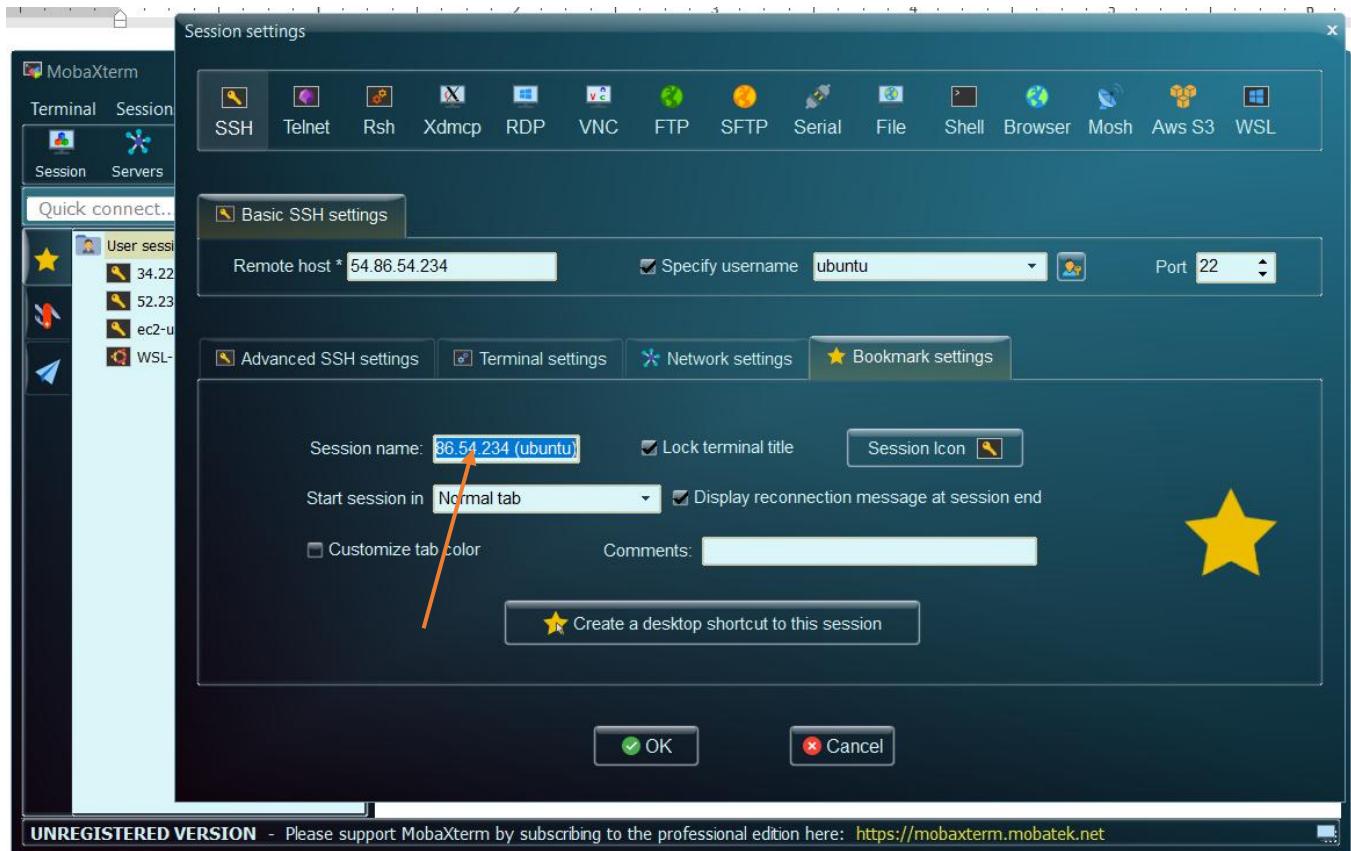
Then browse and select our .pem file for Key pair saved in Downloads folder



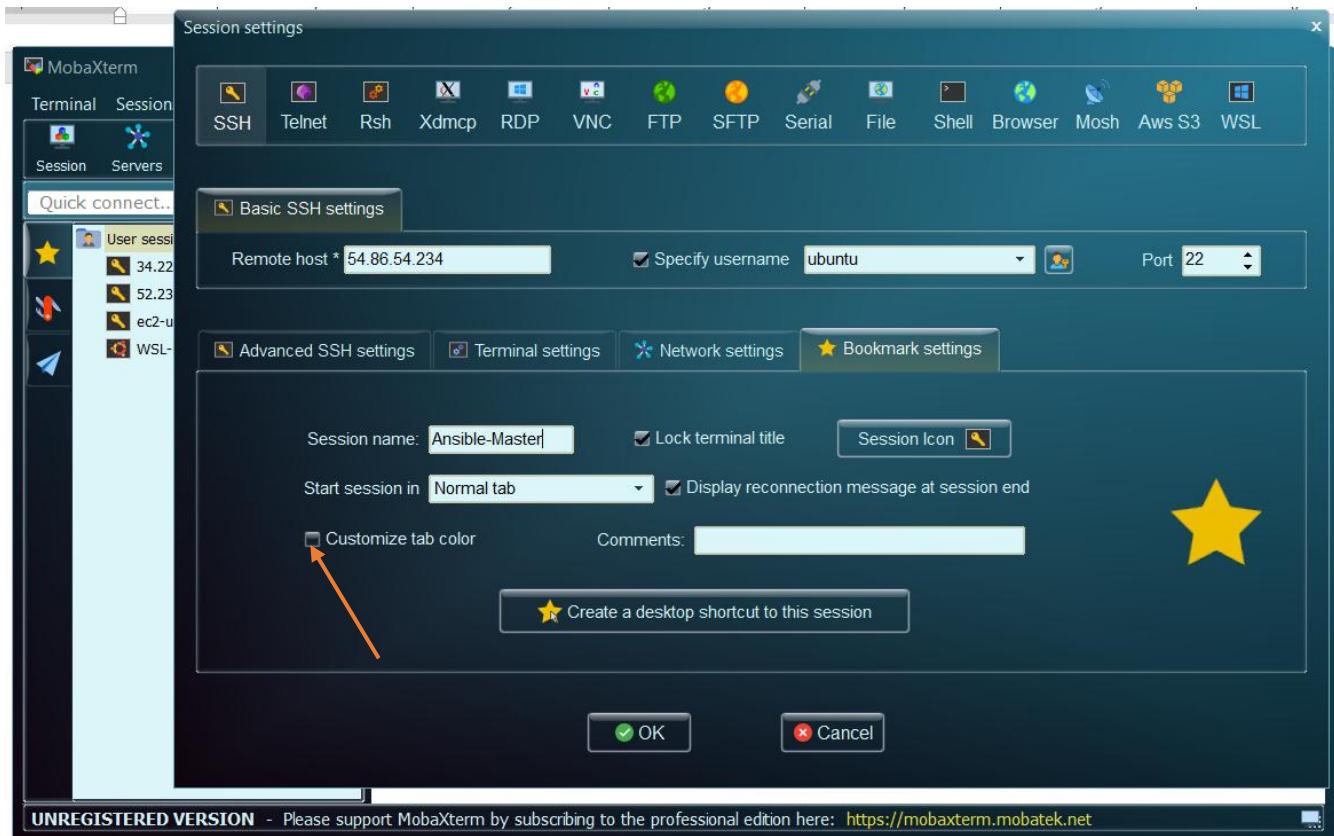
Click on "Open"



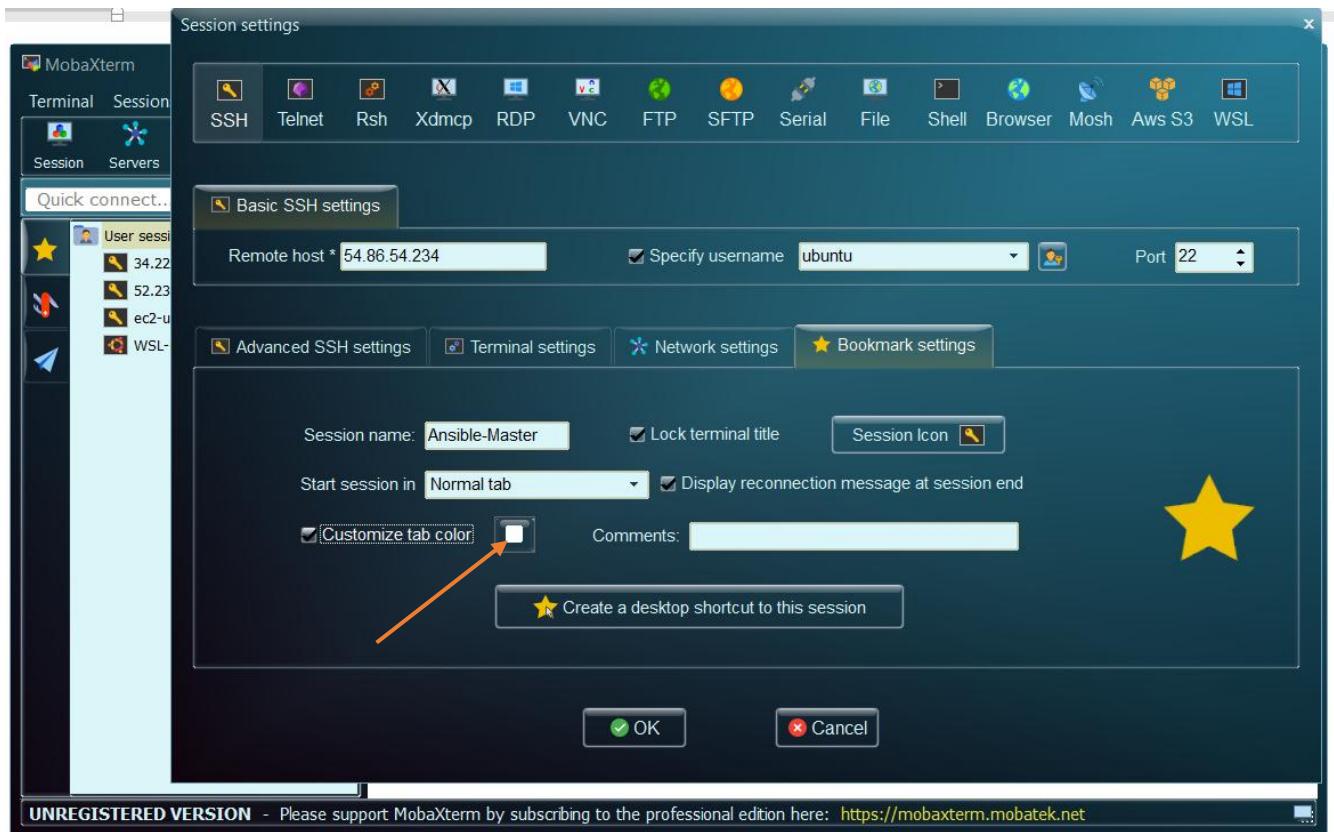
Then name the machine by clicking on “Bookmark settings”



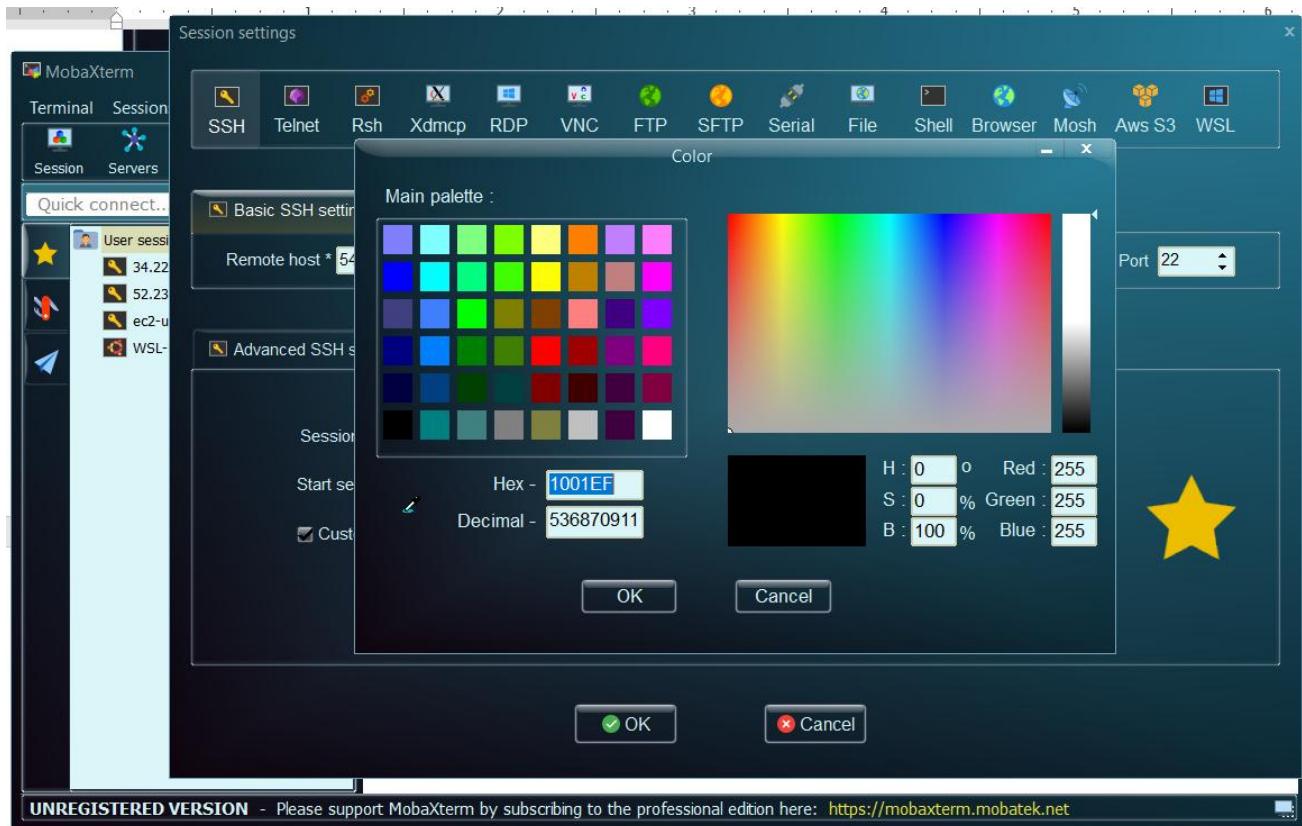
Then for “Session name”, enter “Ansible-Master”



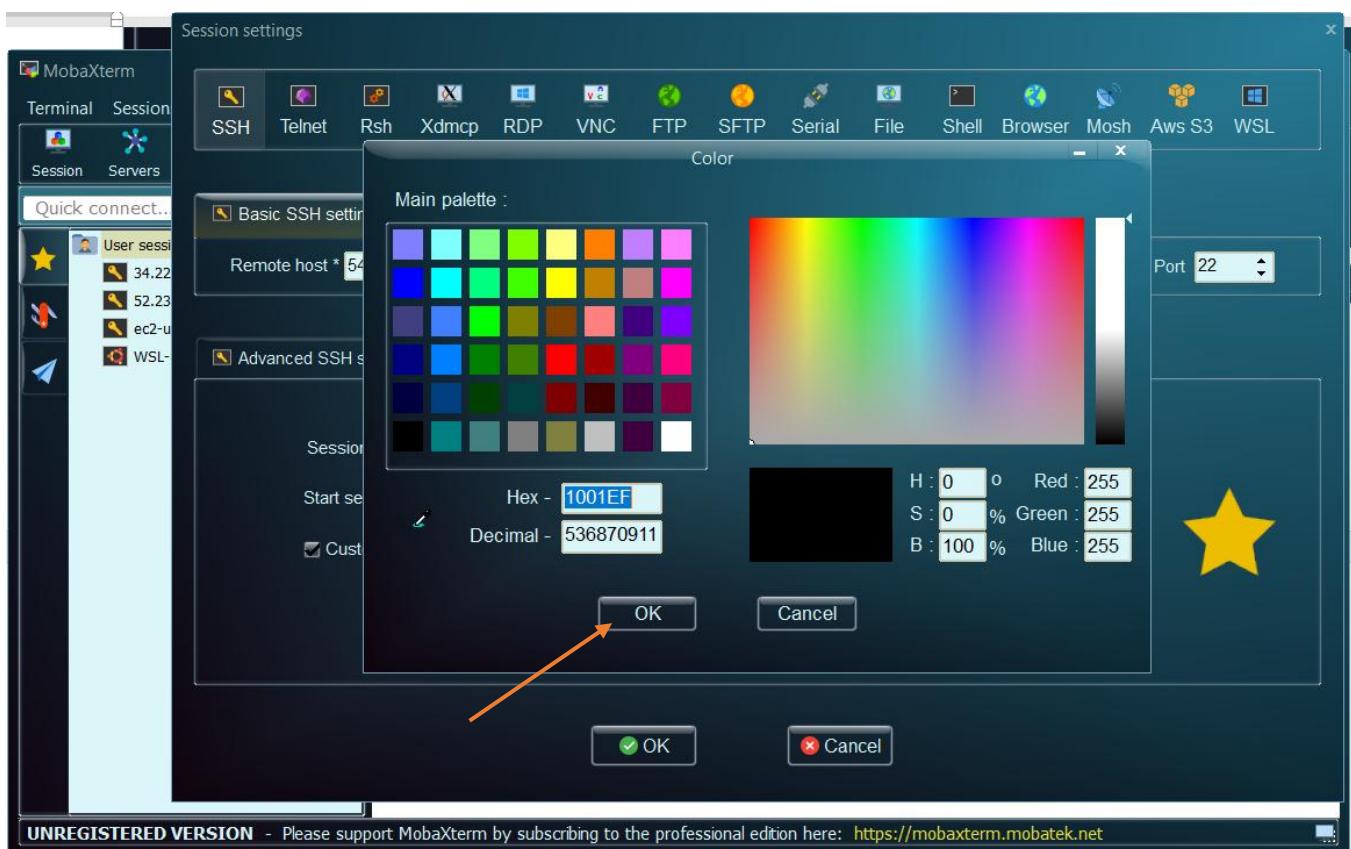
We can give the tab a colour by clicking on “Customize tab colour”



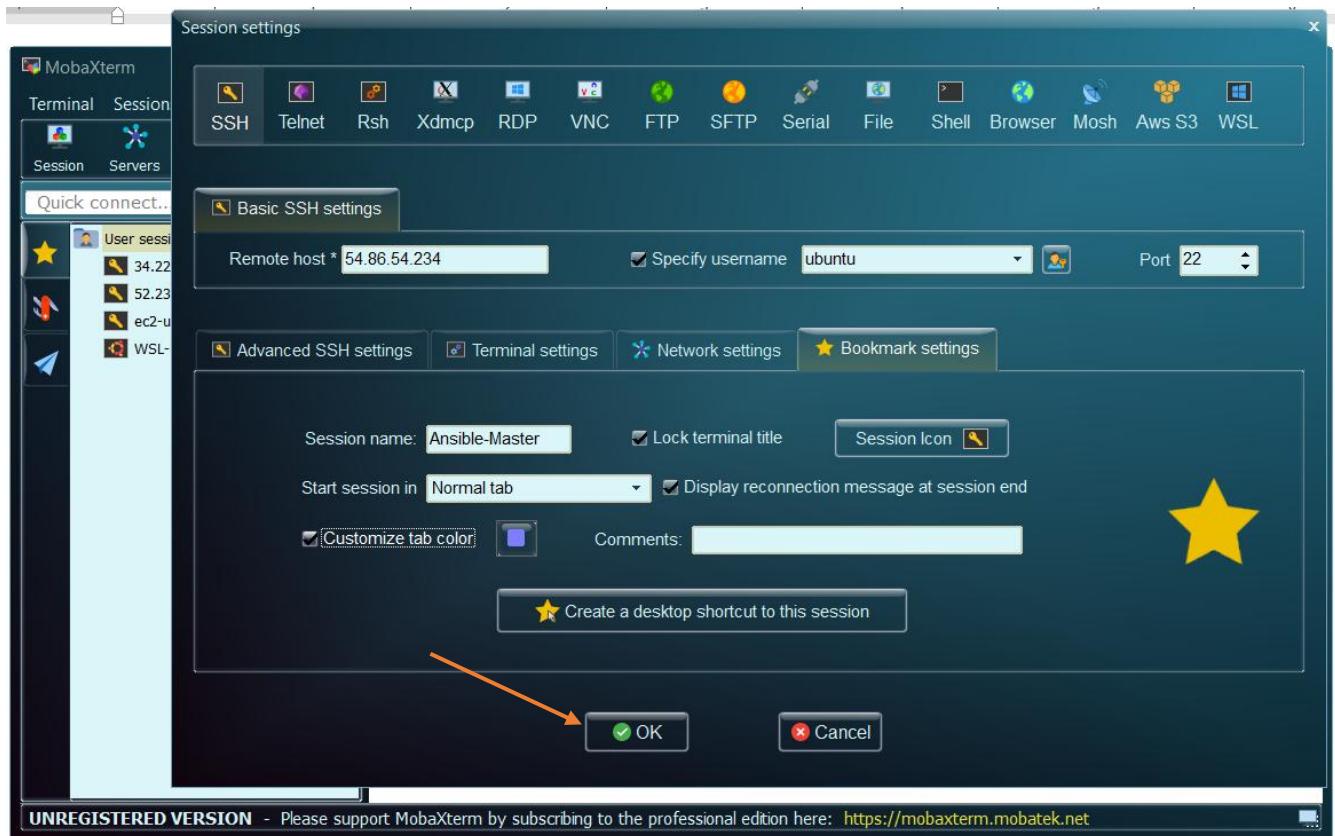
Then browse and select the colour



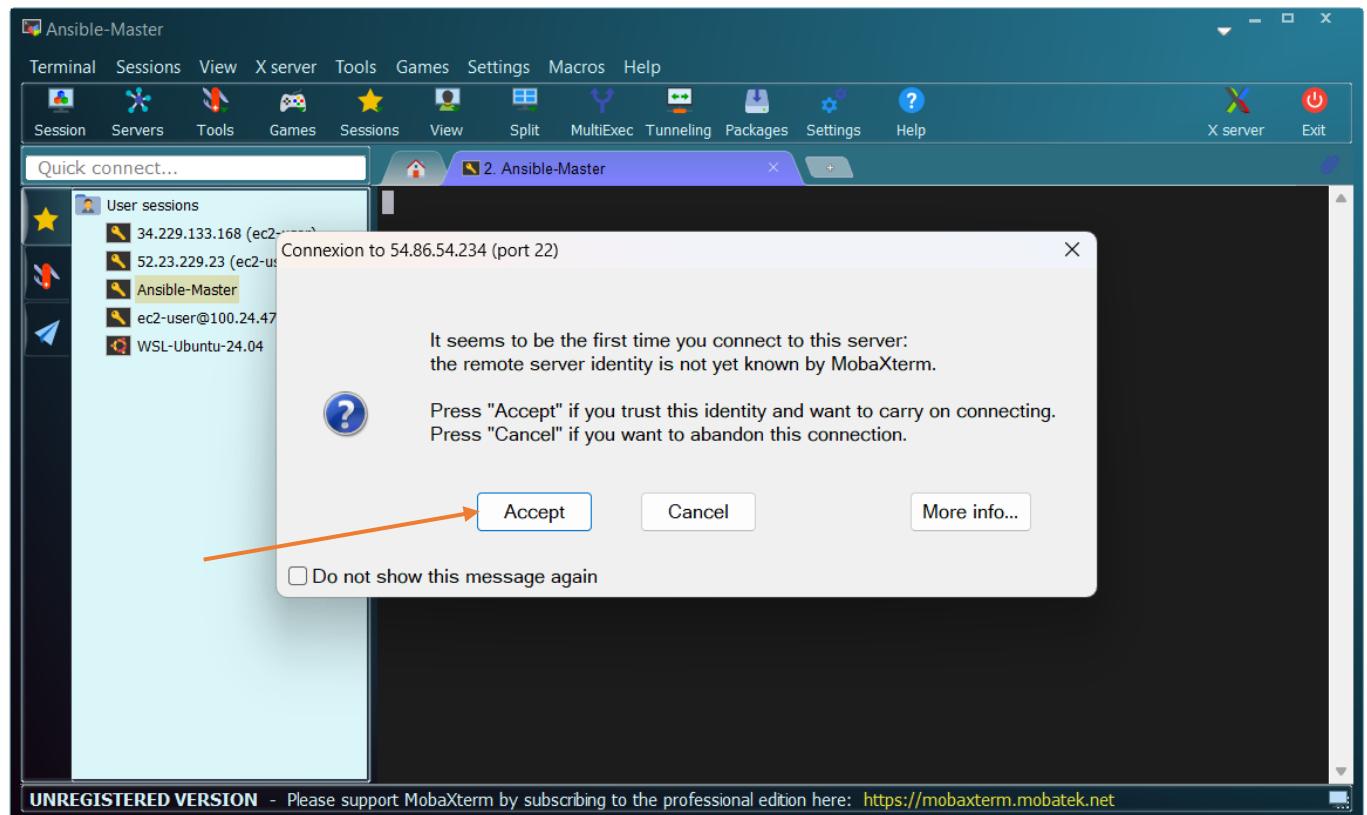
Select the colour you want to use.



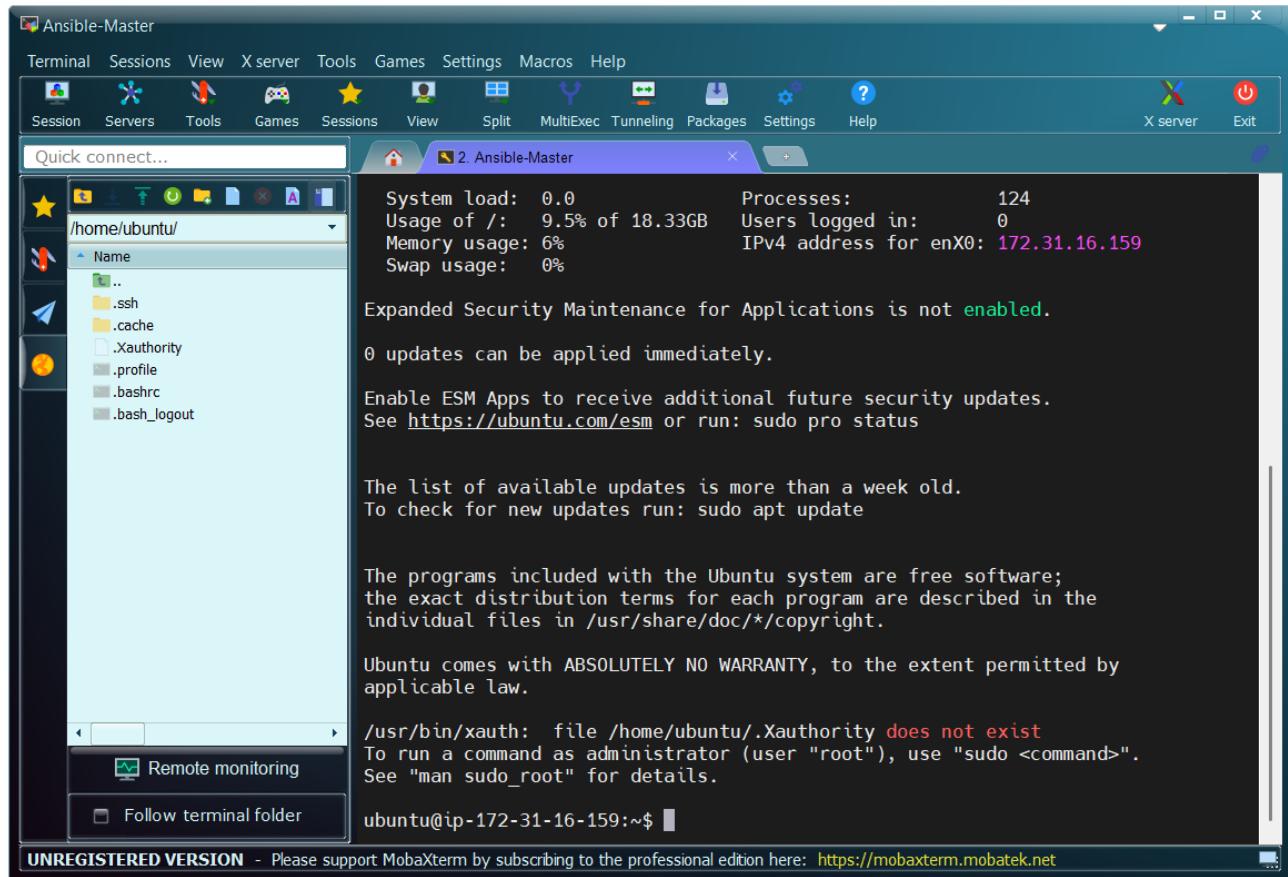
And press "OK"



Then click on “OK”



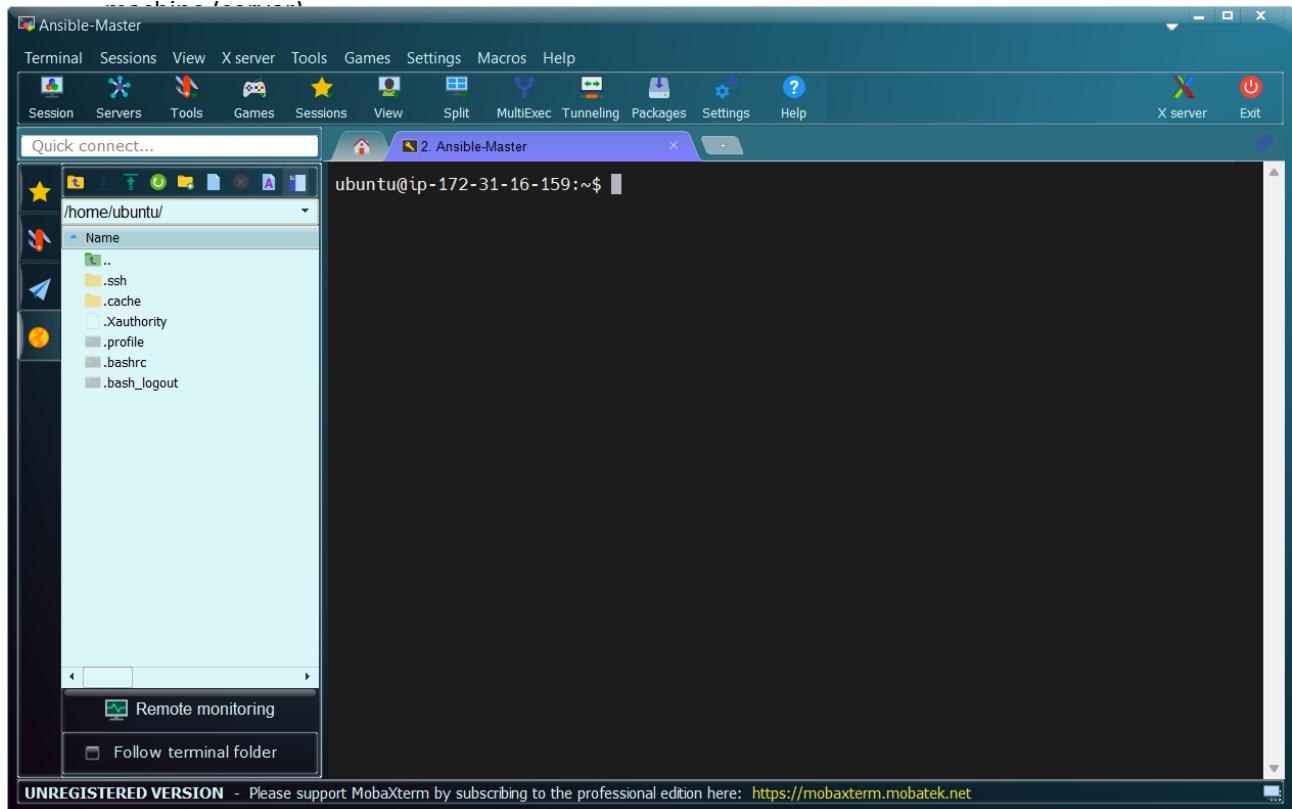
Then click on “Accept”



We are now connected to our “Ansible-Master”.

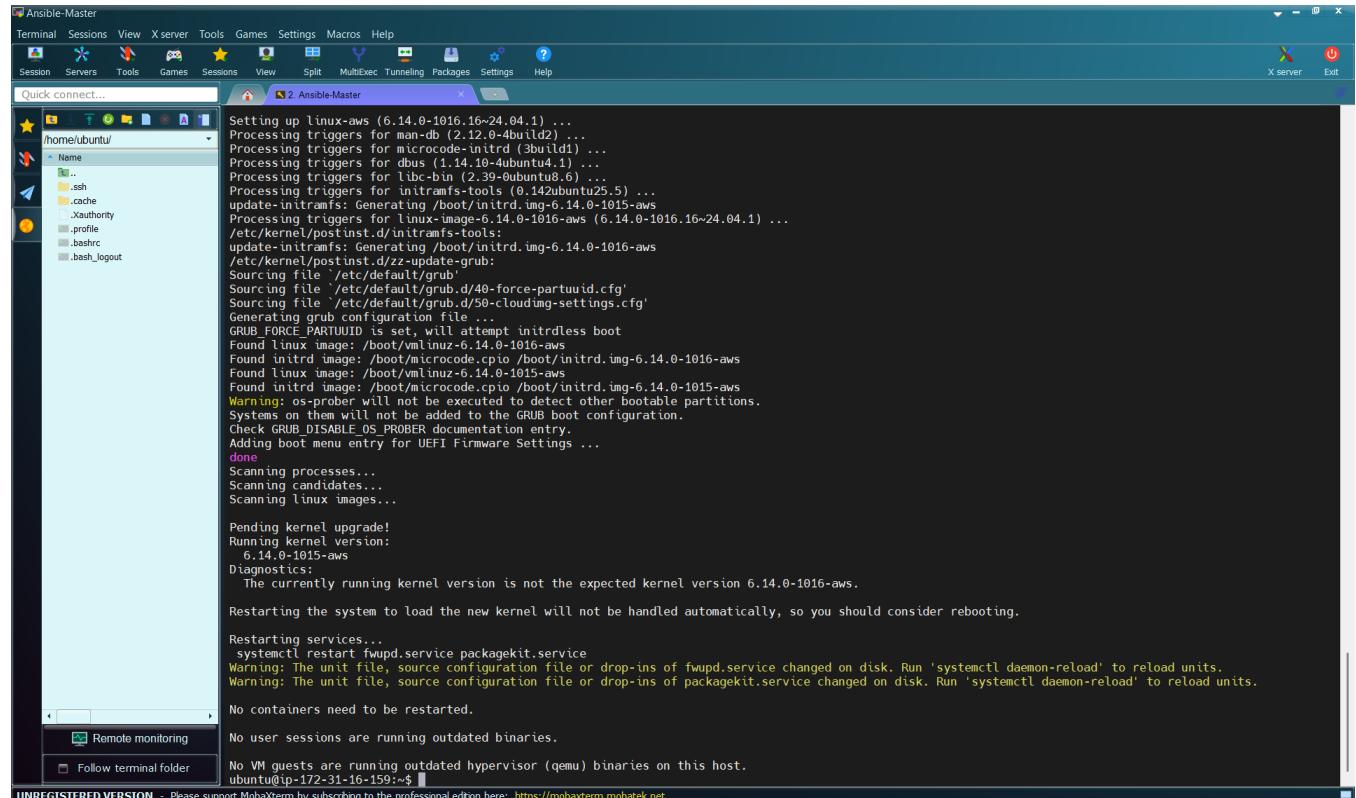
Part 4: Install Ansible on “Ansible-Master” Instance

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



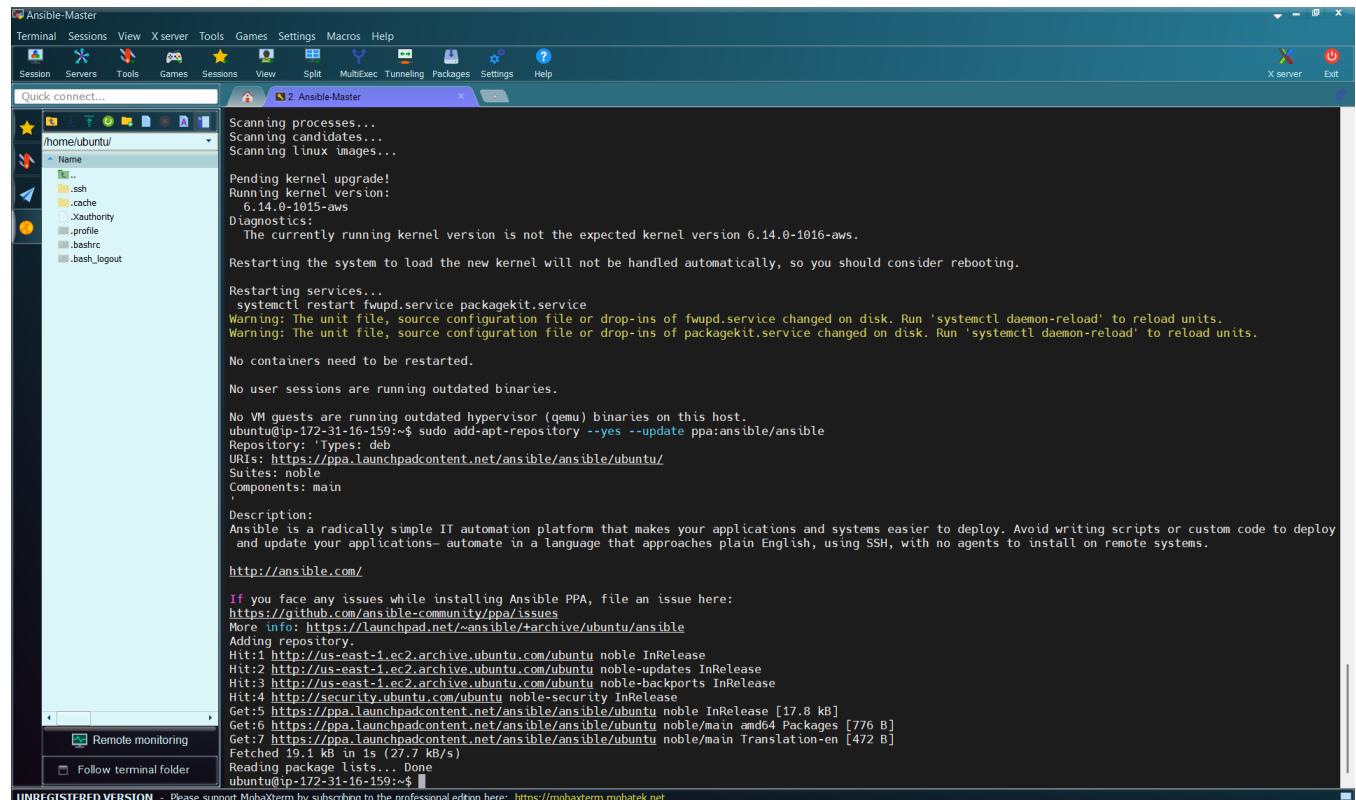
We have to first update and upgrade the system using the command:

```
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
```



The screenshot shows the MobaXterm interface with a terminal window titled 'Ansible-Master'. The terminal output displays the process of adding the Ansible PPA repository:

```

Scanning processes...
Scanning candidates...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
  6.14.0-1015-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.14.0-1016-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

Restarting services...
  systemctl restart fwupd.service packagekit.service
Warning: The unit file, source configuration file or drop-ins of fwupd.service changed on disk. Run 'systemctl daemon-reload' to reload units.
Warning: The unit file, source configuration file or drop-ins of packagekit.service changed on disk. Run 'systemctl daemon-reload' to reload units.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-16-199:~$ sudo add-apt-repository --yes --update ppa:ansible/ansible
Repository: 'Types: deb
URIs: https://ppa.launchpadcontent.net/ansible/ansible/ubuntu/
Suites: noble
Components: main

Description:
Ansible is a radically simple IT automation platform that makes your applications and systems easier to deploy. Avoid writing scripts or custom code to deploy and update your applications—automate in a language that approaches plain English, using SSH, with no agents to install on remote systems.

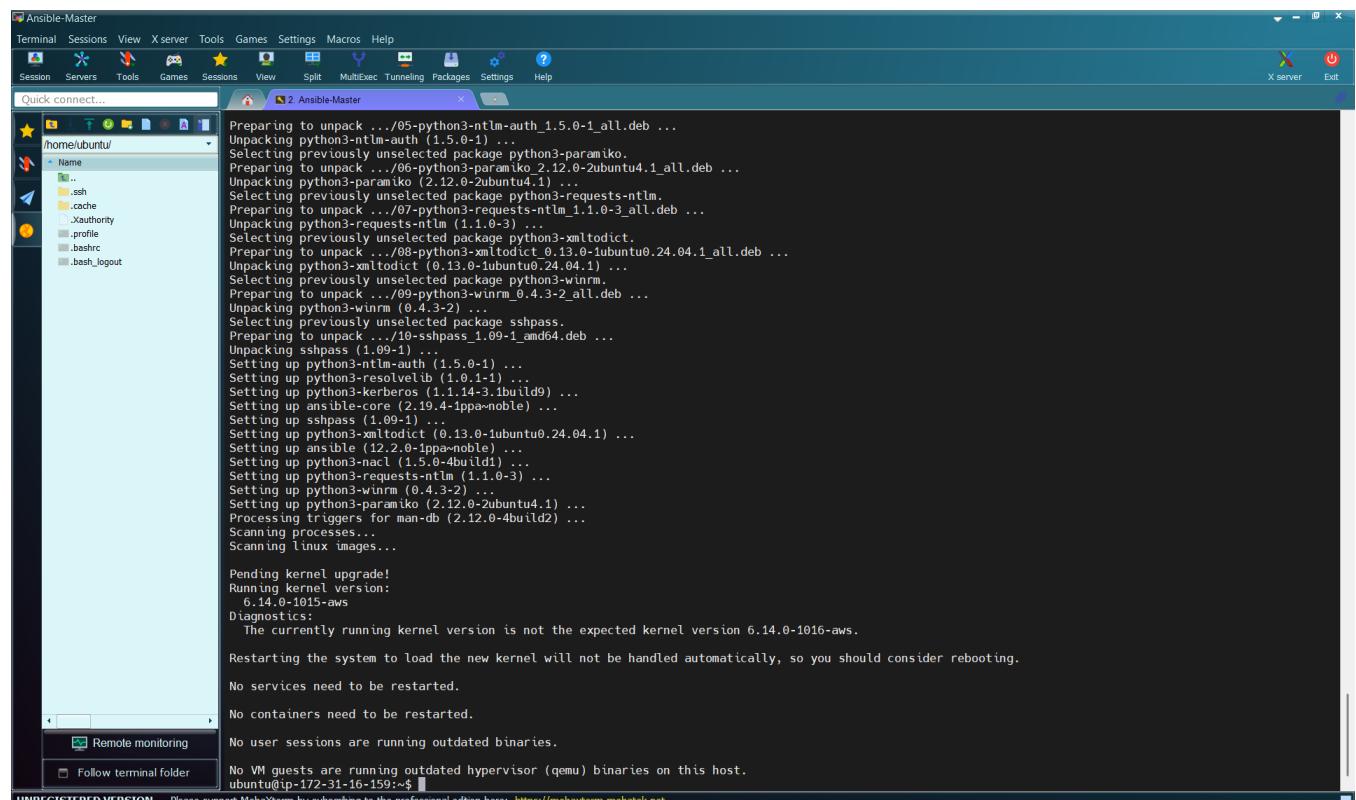
http://ansible.com

If you face any issues while installing Ansible PPA, file an issue here:
https://github.com/ansible-community/ppa/issues
More info: https://launchpad.net/~ansible/+archive/ubuntu/ansible
Adding repository.
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 https://ppa.launchpadcontent.net/ansible/ansible/ubuntu noble InRelease [17.8 kB]
Get:6 https://ppa.launchpadcontent.net/ansible/ansible/ubuntu/noble/main amd64 Packages [776 B]
Get:7 https://ppa.launchpadcontent.net/ansible/ansible/ubuntu/noble/main Translation-en [472 B]
Fetched 19.1 kB in 0s (27.7 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-16-199:~$ 
```

At the bottom of the terminal, it says 'UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>'.

Then Install Ansible

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



The screenshot shows the MobaXterm interface with a terminal window titled 'Ansible-Master'. The terminal output displays the process of installing Ansible:

```

Preparing to unpack .../05-python3-ntlm-auth_1.5.0-1_all.deb ...
Unpacking python3-ntlm-auth (1.5.0-1) ...
Selecting previously unselected package python3-paramiko.
Preparing to unpack .../06-python3-paramiko_2.12.0-2ubuntu4.1_all.deb ...
Unpacking python3-paramiko (2.12.0-2ubuntu4.1) ...
Selecting previously unselected package python3-requests_ntlm.
Preparing to unpack .../07-python3-requests_ntlm_1.1.0-3_all.deb ...
Unpacking python3-requests_ntlm (1.1.0-3) ...
Selecting previously unselected package python3-xmltodict.
Preparing to unpack .../08-python3-xmltodict_0.13.0-1ubuntu0.24.04.1_all.deb ...
Unpacking python3-xmltodict (0.13.0-1ubuntu0.24.04.1) ...
Selecting previously unselected package python3-winrm.
Preparing to unpack .../09-python3-winrm_0.4.3-2_all.deb ...
Unpacking python3-winrm (0.4.3-2) ...
Selecting previously unselected package sshpass.
Preparing to unpack .../10-sshpass_1.09-1_amd64.deb ...
Unpacking sshpass (1.09-1) ...
Setting up python3-ntlm-auth (1.5.0-1) ...
Setting up python3-resolvlib (1.0.1-1) ...
Setting up python3-kerberos (1.1.14-3.1build9) ...
Setting up ansible-core (2.19.4-1ppa-noble) ...
Setting up sshpass (1.09-1) ...
Setting up python3-xmltodict (0.13.0-1ubuntu0.24.04.1) ...
Setting up ansible (12.2.0-1ppa-noble) ...
Setting up python3-nacl (1.5.0-4build1) ...
Setting up python3-requests_ntlm (1.1.0-3) ...
Setting up python3-winrm (0.4.3-2) ...
Setting up python3-paramiko (2.12.0-2ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
  6.14.0-1015-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.14.0-1016-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-16-199:~$ 
```

At the bottom of the terminal, it says 'UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>'.

Now, let us verify the installation using the command:

```
ansible --version
```

The screenshot shows a terminal window titled "Ansible-Master" running on a Linux system. The terminal displays the output of the "ansible --version" command. The output includes details about the Python version (3.12.3), Jinja version (3.1.2), PyYAML version (6.0.1), and Ansible version (2.19.4). It also shows kernel information, including a pending upgrade from 6.14.0-1015-aws to 6.14.0-1016-aws. A warning message states that the currently running kernel version is not the expected kernel version. The terminal interface includes a file browser on the left and various MobaXterm-specific icons and buttons at the top.

```
Unpacking python3-xmldict (0.13.0-1ubuntu0.24.04.1) ...
Selecting previously unselected package python3-winrm ...
Preparing to unpack .../09-python3-winrm_0.4.3-2_all.deb ...
Unpacking python3-winrm (0.4.3-2) ...
Selecting previously unselected package sshpass.
Preparing to unpack .../10-sshpass_1.09-1_amd64.deb ...
Unpacking sshpass (1.09-1) ...
Setting up python3-ntlm-auth (1.5.0-1) ...
Setting up python3-resolvelib (1.0.1-1) ...
Setting up python3-kerberos (1.1.14-3.1build9) ...
Setting up ansible-core (2.19.4-1ppa-noble) ...
Setting up sshpass (1.09-1) ...
Setting up python3-xmldict (0.13.0-1ubuntu0.24.04.1) ...
Setting up ansible (12.2.0-1ppa-noble) ...
Setting up python3-nacl (1.5.8-4build1) ...
Setting up python3-requests-ntlm (1.1.0-3) ...
Setting up python3-winrm (0.4.3-2) ...
Setting up python3-paramiko (2.12.6-2ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
6.14.0-1015-aws
Diagnostics:
The currently running kernel version is not the expected kernel version 6.14.0-1016-aws.
Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-16-159:~$ ansible --version
ansible [core 2.19.4]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['~/home/ubuntu/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/ubuntu/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Aug 14 2025, 17:47:21) [GCC 13.3.0] (/usr/bin/python3)
  jinja version = 3.1.2
  pyyaml version = 6.0.1 (with libyaml v0.2.5)
ubuntu@ip-172-31-16-159:~$
```

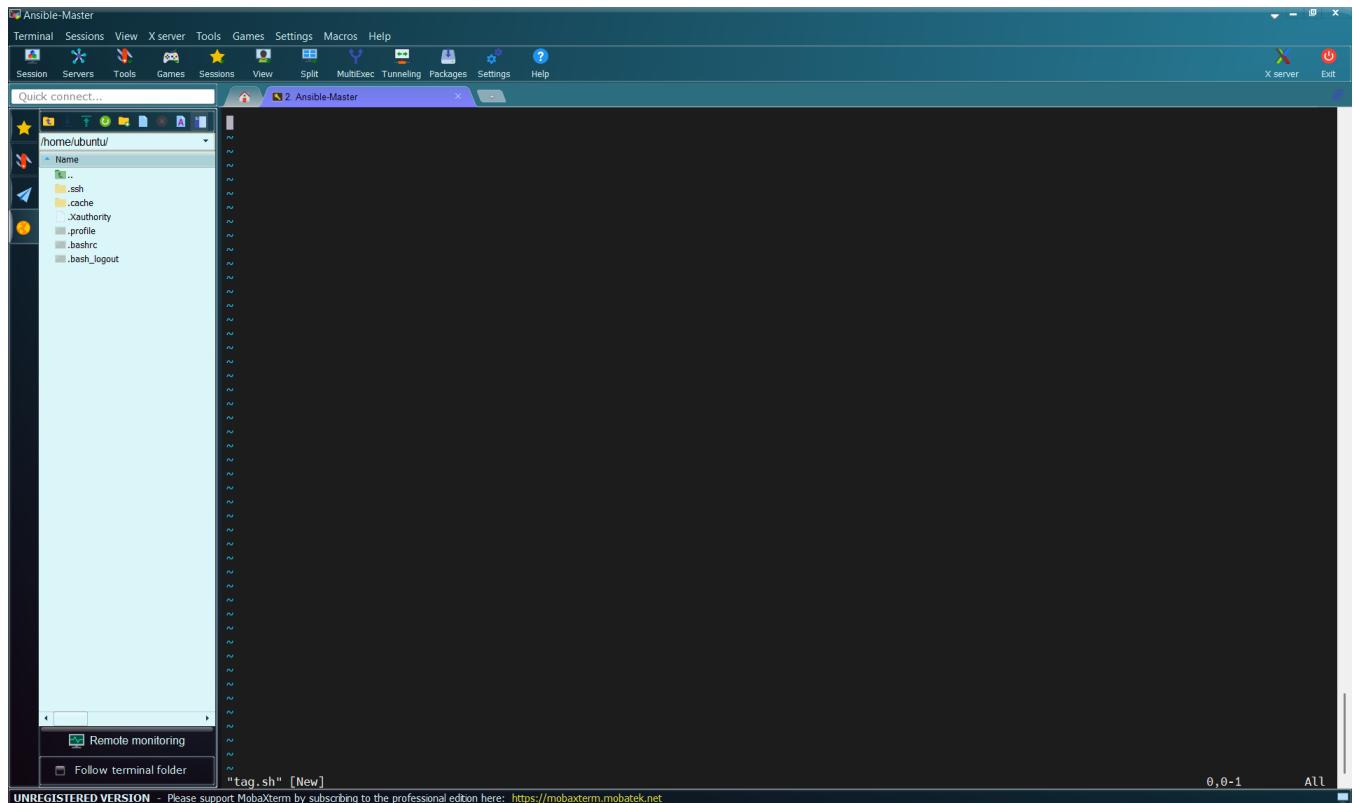
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

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
```

```

#!/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**”

```

Selecting previously unselected package python3-wi...
Preparing to unpack .../09-python3-wi_0.4.3-2_all.deb ...
Unpacking python3-wi (0.4.3-2) ...
Selecting previously unselected package sshpass.
Preparing to unpack .../10-sshpass_1.09-1_amd64.deb ...
Unpacking sshpass (1.09-1) ...
Setting up python3-ntlm-auth (1.5.0-1) ...
Setting up python3-resolvelib (1.0.1-1) ...
Setting up python3-kerberos (1.1.14-3.1build9) ...
Setting up python3-ansible-core (2.19.4-1ppa-noble) ...
Setting up sshpass (1.09-1) ...
Setting up python3-ntlm (0.13.0-1ubuntu0.24.04.1) ...
Setting up pnshtle (12.2.0-1ppa-noble) ...
Setting up python3-nacl (1.5.0-4build1) ...
Setting up python3-requests-ntlm (1.1.0-3) ...
Setting up python3-wi (0.4.3-2) ...
Setting up python3-paramiko (2.12.0-2ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
 6.14.0-1015-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.14.0-1016-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

No services need to be restarted.

No containers need to be restarted.

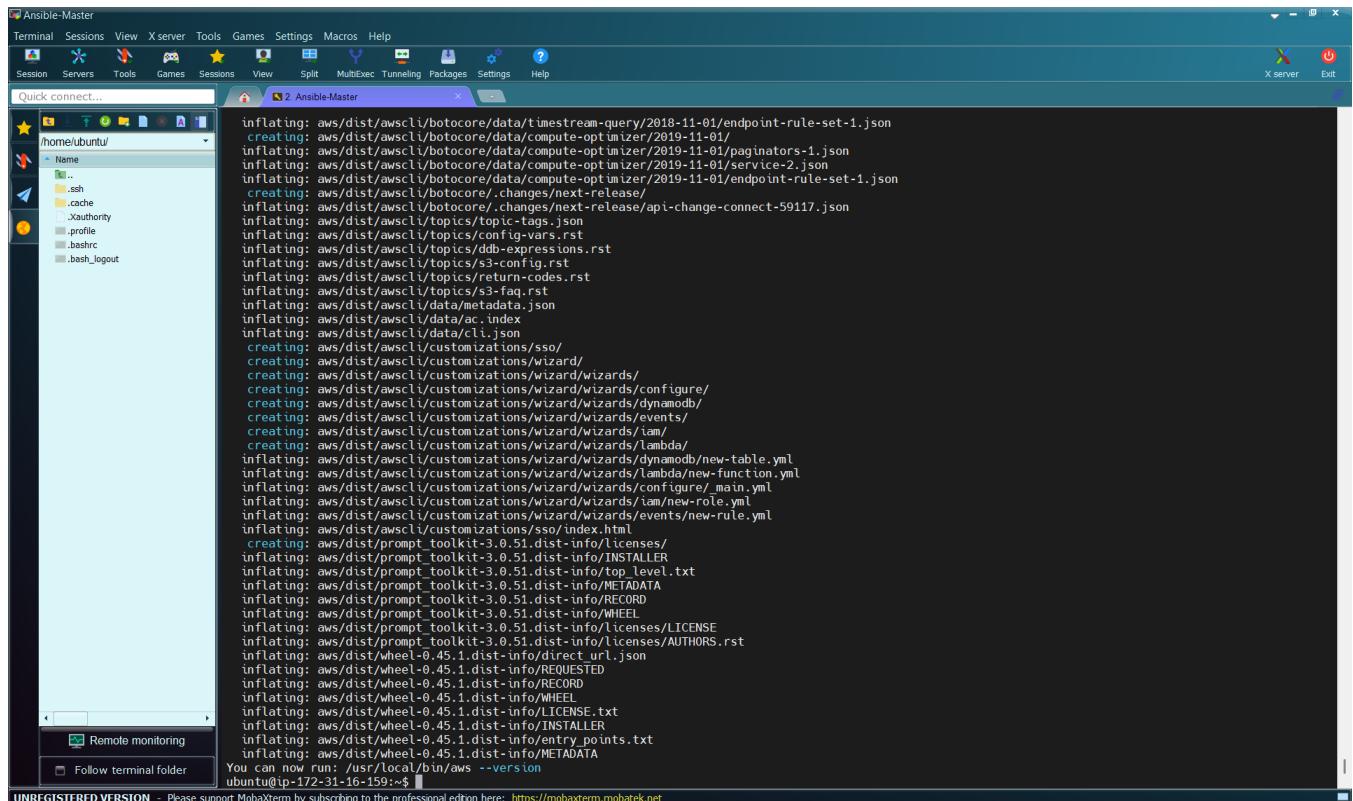
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-16-199:~$ ansible --version
ansible [core 2.19.4]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/ubuntu/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/ubuntu/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Aug 14 2025, 17:47:21) [GCC 13.3.0] (/usr/bin/python3)
  jinja version = 3.1.2
  pyyaml version = 6.0.1 (with libyaml v0.2.5)
ubuntu@ip-172-31-16-199:~$ vi tag.sh
ubuntu@ip-172-31-16-199:~$ 
```

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

Prepared by Sidney Smith Ebott

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with sections like EC2, Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, CloudShell, and Feedback. The main area displays a table titled 'Instances (11) Info' with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. All instances are listed as 'Running'. A search bar at the top says 'Find Instance by attribute or tag (case-sensitive)'. At the bottom, it says 'Select an instance'. In the top right corner, it shows 'Account ID: 3247-8332-4460' and the user name 'sidney'.

Click on your account name

This screenshot is similar to the previous one, showing the EC2 Instances page. However, a context menu is open over the account name 'sidney' in the top right. The menu items are: Account ID (3247-8332-4460), Account color (Unset), IAM user (sidney), Account, Organization, Service Quotas, Billing and Cost Management, Security credentials, Turn on multi-session support, Switch role, and Sign out. The URL at the bottom of the page is https://us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#security_credential...

Click on “Security Credentials”

The screenshot shows the AWS IAM Access Keys page. On the left, there's a sidebar with navigation links like Dashboard, Access management, and Access reports. The main area displays a table for access keys. One key is listed:

Type	Identifier	Certifications	Created on
No MFA devices.	Assign MFA device		

Access keys (1)

AKIAUXHUYHUWKL3HVOI

Description: -

Last used: 21 days ago

Last used region: us-east-1

Status: Active

Created: 21 days ago

Last used service: ec2

[Actions](#)

X.509 Signing certificates (0)

No X.509 certificates

[Create X.509 certificate](#)

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:

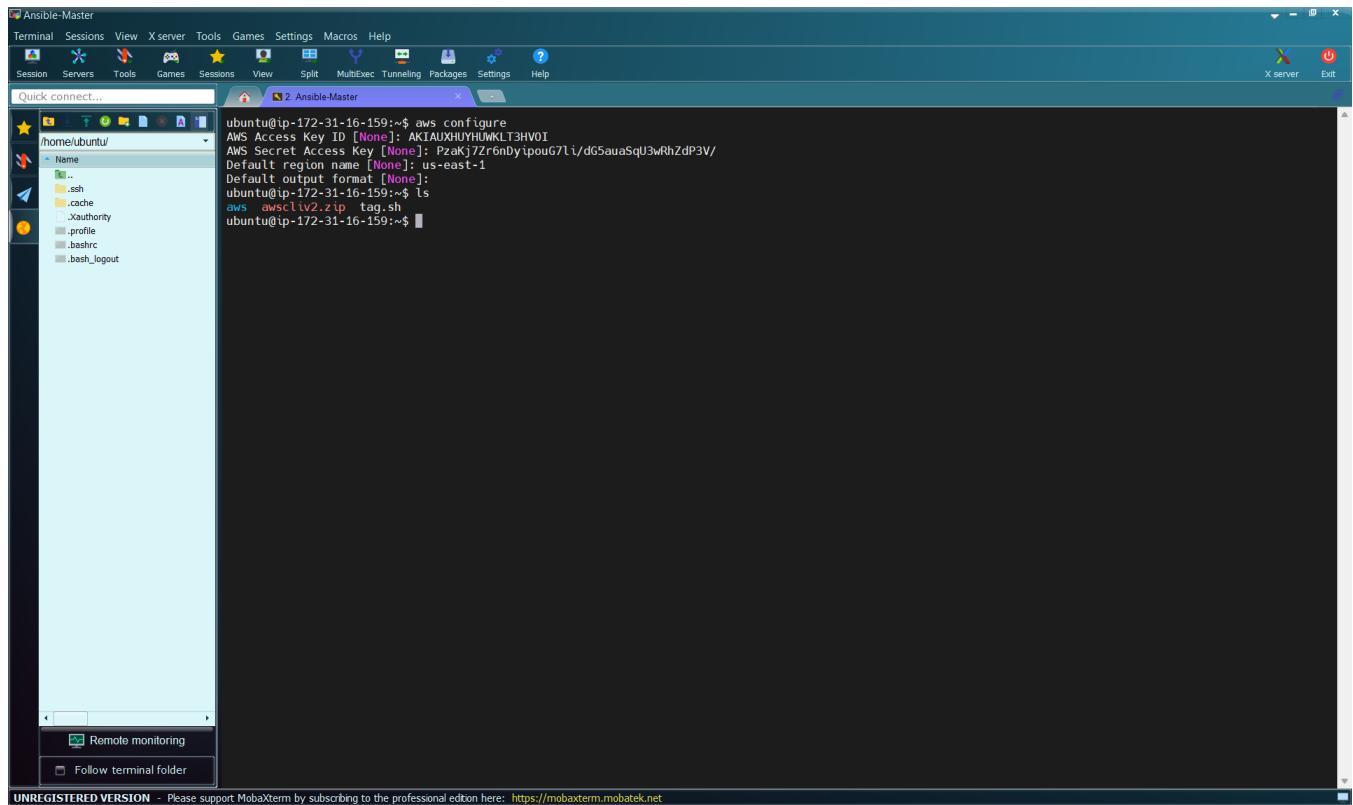
aws configure

The terminal window shows the command being run:

```
ubuntu@ip-172-31-16-159:~$ aws configure
AWS Access Key ID [None]: AKIAUXHUYHUWKL3HVOI
AWS Secret Access Key [None]: PzAkj7Zr6nDyIpouG7lVd65auaSqJ3wRhZdp3V/
Default region name [None]: us-east-1
Default output format [None]:
ubuntu@ip-172-31-16-159:~$
```

Let's check the content by using the command:

ls



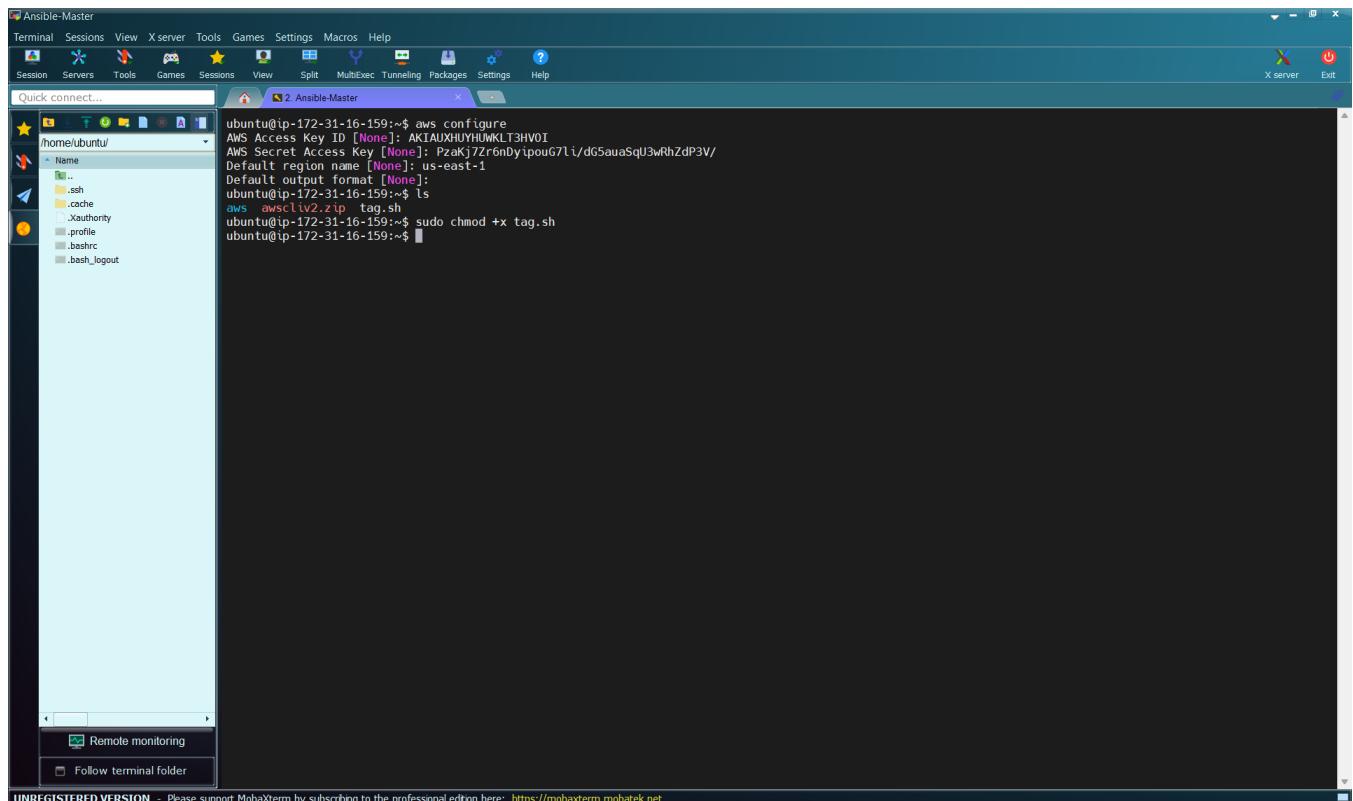
The screenshot shows a MobaXterm session titled "Ansible-Master". The terminal window displays the following command output:

```
ubuntu@ip-172-31-16-159:~$ aws configure
AWS Access Key ID [None]: AKIAUXHUYHUMKLT3HVOI
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:
ubuntu@ip-172-31-16-159:~$ ls
aws awscli2.zip tag.sh
ubuntu@ip-172-31-16-159:~$
```

The left pane shows a file tree for the current directory (~). The right pane contains a "Remote monitoring" and "Follow terminal folder" option.

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
```



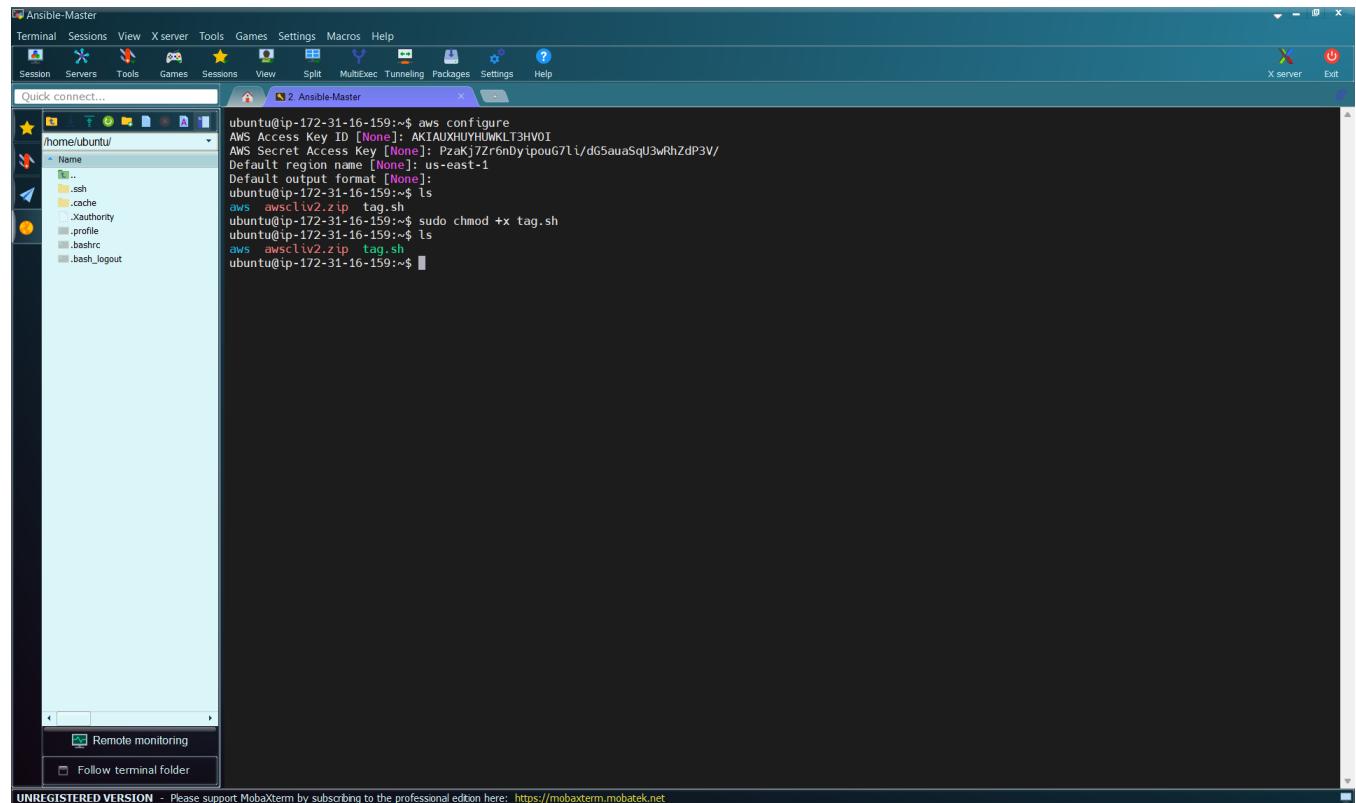
The screenshot shows a MobaXterm session titled "Ansible-Master". The terminal window displays the following command output:

```
ubuntu@ip-172-31-16-159:~$ aws configure
AWS Access Key ID [None]: AKIAUXHUYHUMKLT3HVOI
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:
ubuntu@ip-172-31-16-159:~$ ls
aws awscli2.zip tag.sh
ubuntu@ip-172-31-16-159:~$ sudo chmod +x tag.sh
ubuntu@ip-172-31-16-159:~$
```

The left pane shows a file tree for the current directory (~). The right pane contains a "Remote monitoring" and "Follow terminal folder" option.

Run the command again

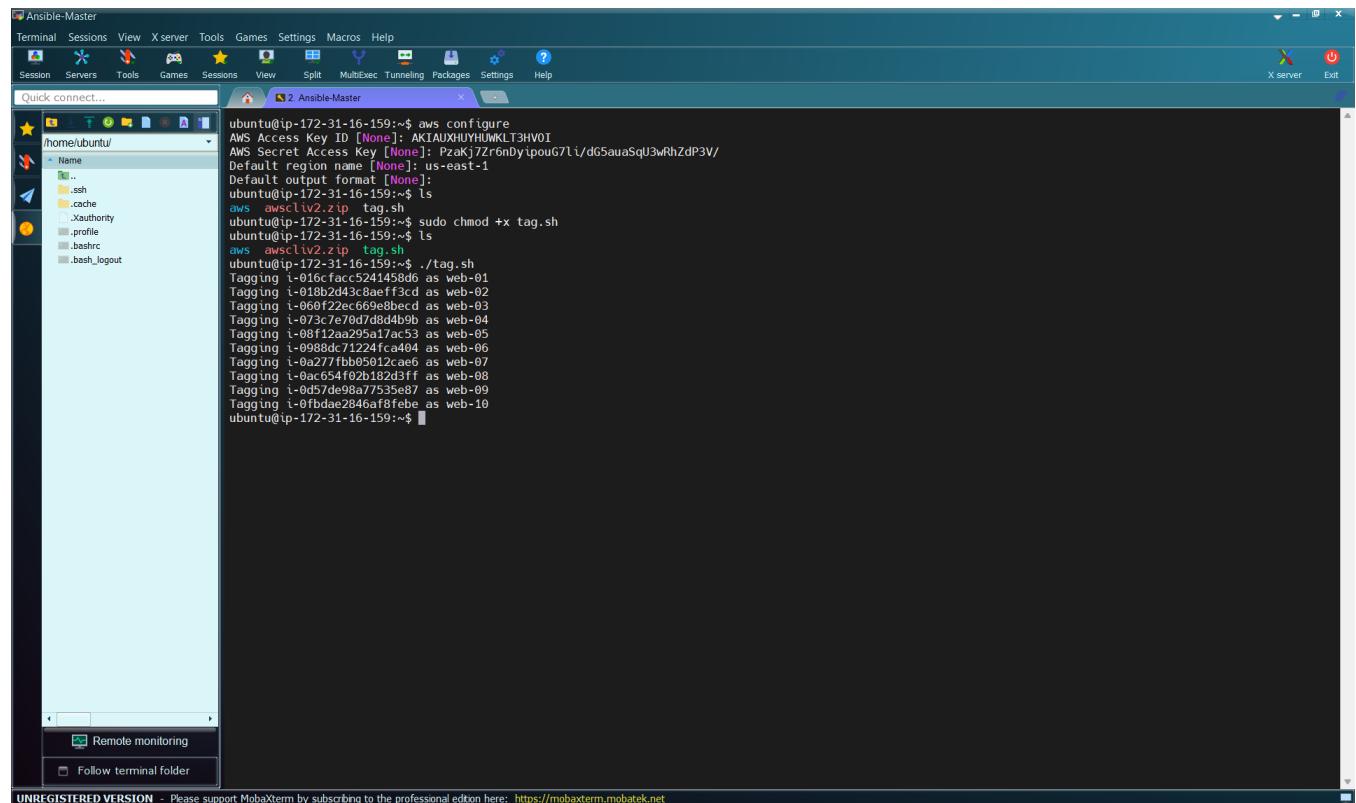
ls



A screenshot of the MobaXterm application window titled "Ansible-Master". The terminal tab is active, showing a Linux command-line session. The user has run the "aws configure" command, which prompts for AWS Access Key ID and Secret Access Key. The user has provided these credentials. The terminal then lists several files in the current directory: .., .ssh, .cache, .xauthority, .profile, .bashrc, and .bash_logout. At the bottom of the terminal window, there is a message: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

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

./tag.sh



A screenshot of the MobaXterm application window titled "Ansible-Master". The terminal tab is active, showing a Linux command-line session. The user has run the command "./tag.sh", which triggers the execution of the "tag.sh" script. The output of the script shows numerous lines starting with "Tagging" followed by a long string of characters, indicating the tagging of multiple AWS resources. The terminal window also displays the same "UNREGISTERED VERSION" message at the bottom.

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

The screenshot shows the AWS EC2 Instances page with 11 instances listed:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
web-08	i-Oac654f02b182d3ff	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web-07	i-0a277fb05012cae6	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web-04	i-073ce7e70d78d4b9b	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web-05	i-060f22ec669e8becd	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web-10	i-0fbdae2846af8febe	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web-09	i-08f12aa295a17ac53	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web-02	i-018bd43c8aef73cd	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web-01	i-016cfacc5241458d6	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
web-06	i-0988dc71224fca404	Running	t2.small	2/2 checks passed	View alarms +	us-east-1d
Ansible-Master	i-0792eb33eba9a1ea1	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1d

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

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

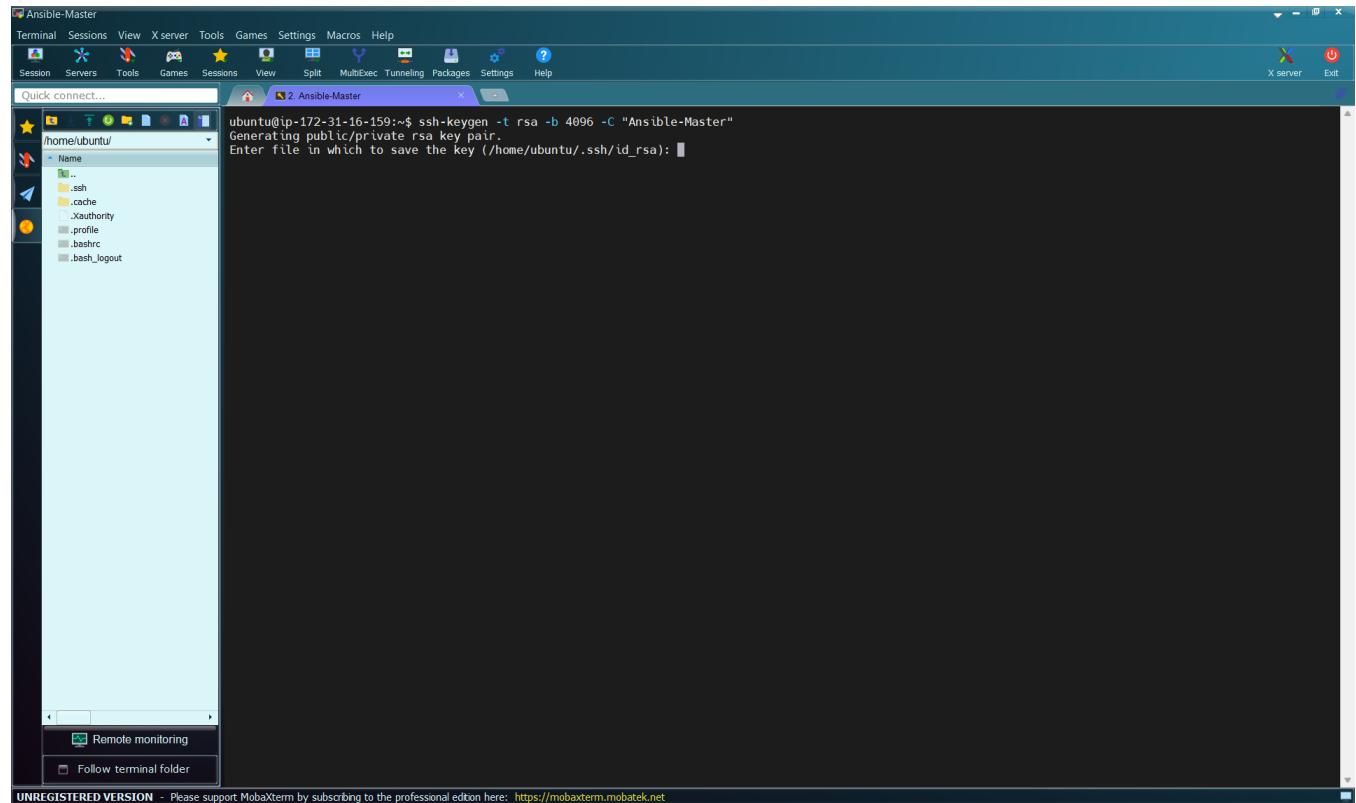
The next thing to do is to establish communication between the Master Instance (Ansible-Master) and the target instances.

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

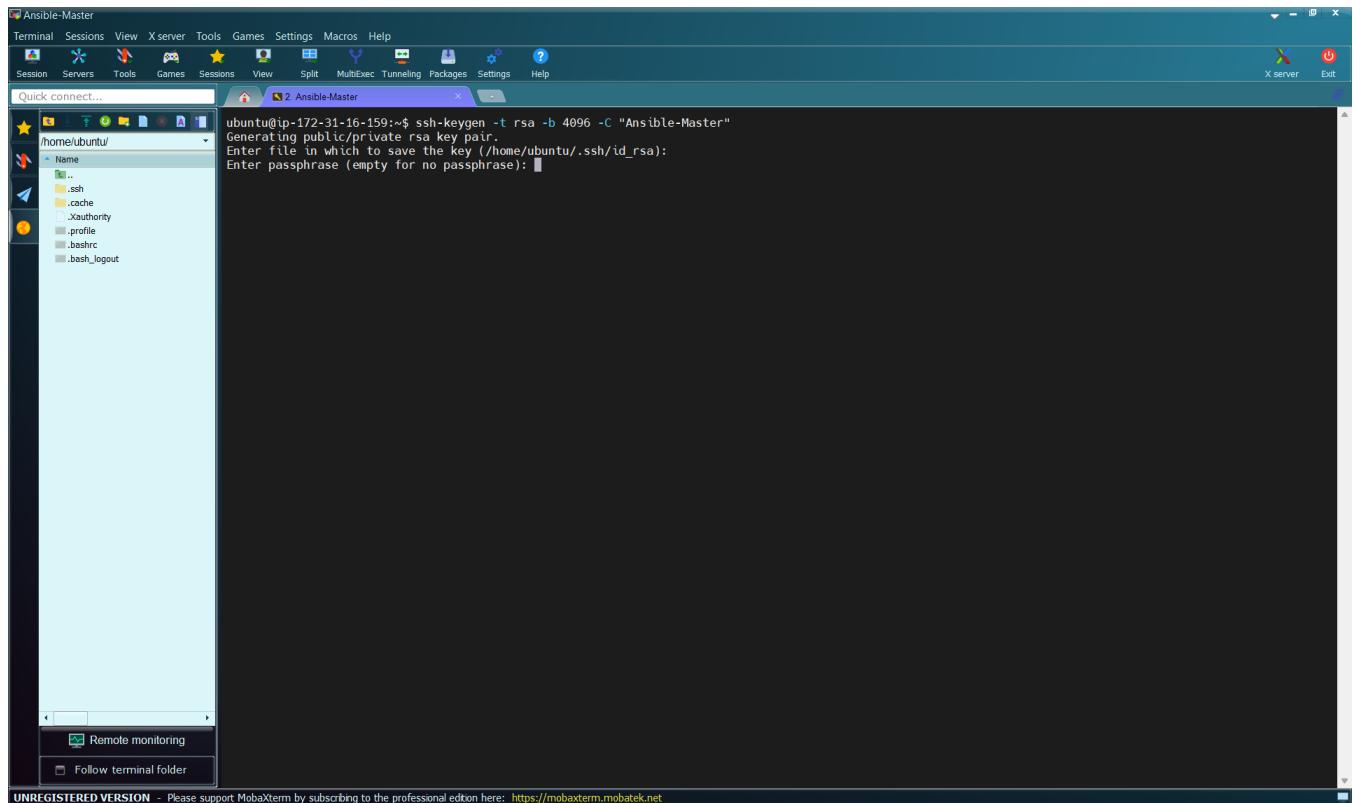
To create the SSH Key Pair, we will use the command:

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



Press “Enter”

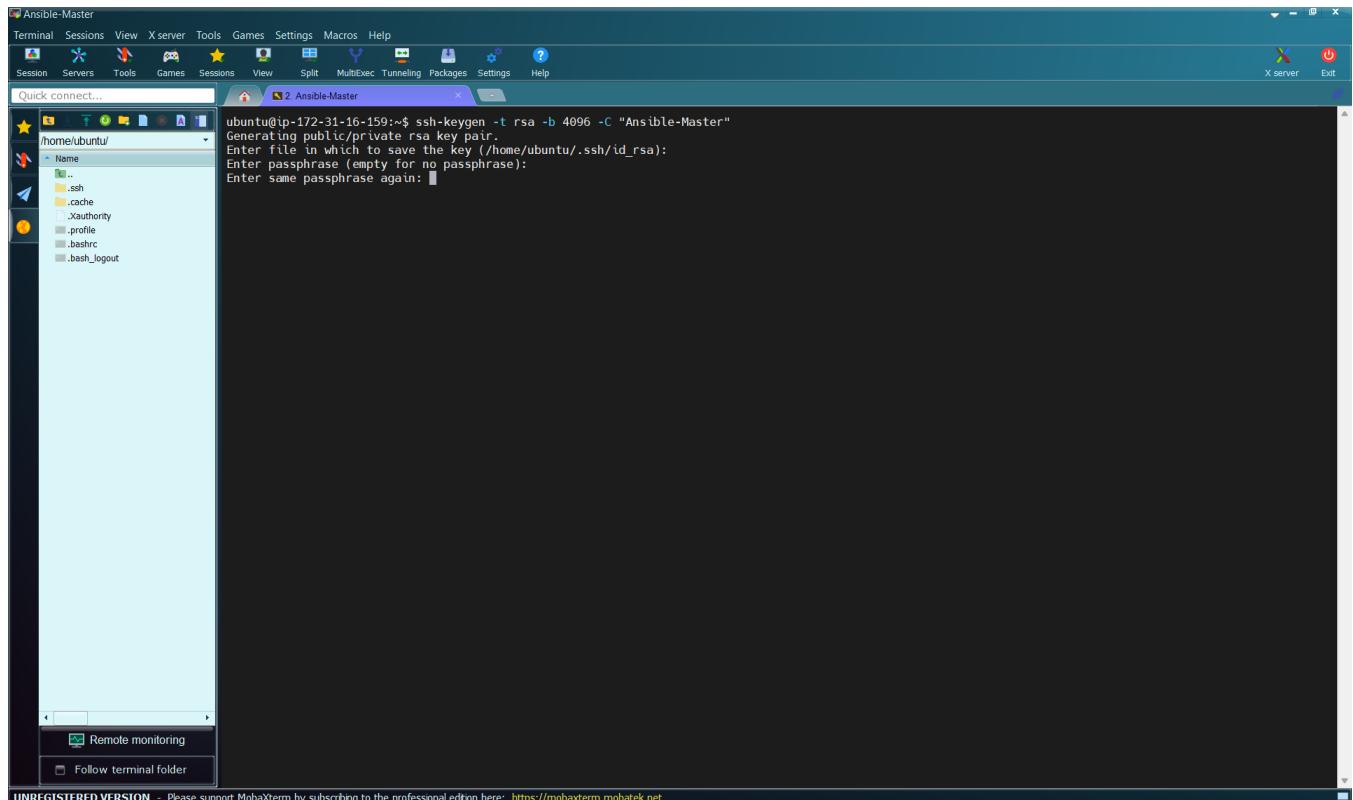
Prepared by Sidney Smith Ebot



The screenshot shows a MobaXterm window titled "Ansible-Master". The terminal session is running on an Ubuntu system. The user has just entered the command `ssh-keygen -t rsa -b 4096 -C "Ansible-Master"`. The terminal is prompting for the location to save the key and a passphrase.

```
ubuntu@ip-172-31-16-159:~$ ssh-keygen -t rsa -b 4096 -C "Ansible-Master"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
```

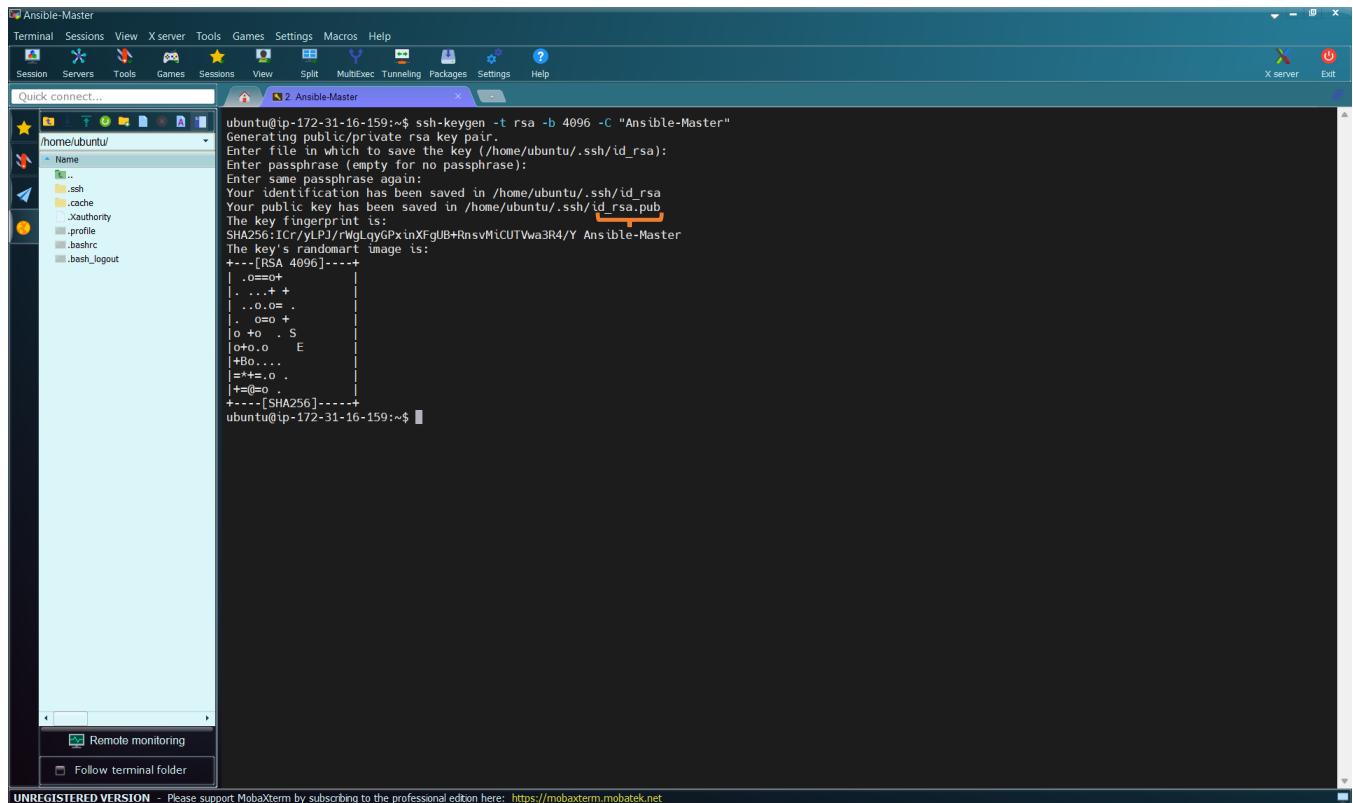
Press Enter again



The screenshot shows a MobaXterm window titled "Ansible-Master". The terminal session is running on an Ubuntu system. The user has just entered the command `ssh-keygen -t rsa -b 4096 -C "Ansible-Master"`. The terminal is prompting for the location to save the key and a passphrase. In this step, the user has left the passphrase field empty.

```
ubuntu@ip-172-31-16-159:~$ ssh-keygen -t rsa -b 4096 -C "Ansible-Master"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
```

And press Enter again



The screenshot shows a terminal window in MobaXterm titled "Ansible-Master". The command "ssh-keygen -t rsa -b 4096 -C \"Ansible-Master\"" is being run. The terminal output shows the key generation process, including prompts for saving the key and entering a passphrase. The public key is saved to /home/ubuntu/.ssh/id_rsa.pub, and its SHA256 fingerprint is displayed. The private key's randomart image is also shown.

```
ubuntu@ip-172-31-16-159:~$ ssh-keygen -t rsa -b 4096 -C "Ansible-Master"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:1Cr/yLPJ/rwGQyGPxinXfgUB+RnsvMiCUTWa3R4/Y Ansible-Master
The key's randomart image is:
+---[RSA 4096]----+
| .o==o+
| . .+++
| ..0.o .
| . o=o+
| o +o . S
| o+o.o E
| +Bo...+
| =*=+.o .
| +=@=o
+---[SHA256]----+
ubuntu@ip-172-31-16-159:~$
```

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 is 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
```

The screenshot shows a MobaXterm terminal window titled "Ansible-Master". The terminal session is running on an Ubuntu host. The user has run the command `ssh-keygen -t rsa -b 4096 -C "Ansible-Master"`, which generates an RSA key pair. The public key is saved to `/home/ubuntu/.ssh/id_rsa`. The key's fingerprint is displayed as `SHA256:1Cr/yLPJ/rwGQyGPxinXfgUB+RnsvMiCUTwa3R4/Y Ansible-Master`. The user then creates a new directory named "inventory" in their home folder.

```
ubuntu@ip-172-31-16-159:~$ ssh-keygen -t rsa -b 4096 -C "Ansible-Master"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:1Cr/yLPJ/rwGQyGPxinXfgUB+RnsvMiCUTwa3R4/Y Ansible-Master
The key's randomart image is:
+---[RSA 4096]----+
| .o==o+
| . .+++
| ..o.o .
| . o=o+
| o +o . S
| o+o+E
| +Bo...+
| =*=+.o .
| +o=o
+---[SHA256]----+
ubuntu@ip-172-31-16-159:~$ mkdir inventory
ubuntu@ip-172-31-16-159:~$
```

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

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

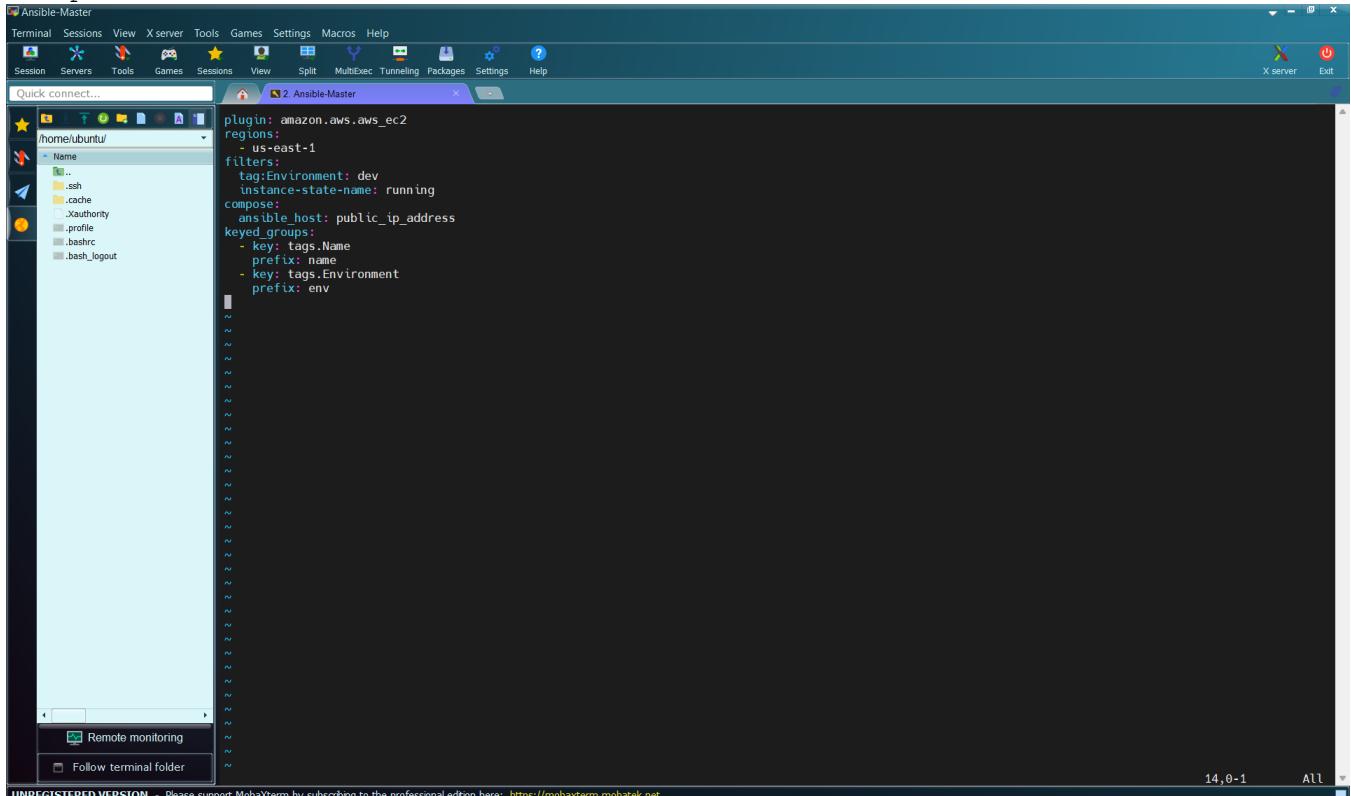
The screenshot shows a MobaXterm terminal window titled "Ansible-Master". The terminal session is running on an Ubuntu host. The user has opened a new file named "inventory/aws_ec2.yaml" in the "inventory" directory using the `vi` editor. The file is currently empty.

```
"inventory/aws_ec2.yaml" [New]
```

Then we paste the code:

Prepared by Sidney Smith Ebot

```
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”

The screenshot shows a MobaXterm interface with two sessions open. Session 1 is titled 'Ansible-Master' and contains the following terminal output:

```
ubuntu@ip-172-31-16-159:~$ ssh-keygen -t rsa -b 4096 -C "Ansible-Master"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:ICr/YLPJ/rMqlqyGPxInXFqUB+RnsvMiCUTWa3R4/Y Ansible-Master
The key's randomart image is:
+---[RSA 4096]---+
| .o=+ |
| ..+ + |
| ..o= . |
| .. o=+ |
| o+o . S |
| o+o.o E |
| +B.o... |
| =*=+o . |
| |=o=o . |
+----[SHA256]----+
ubuntu@ip-172-31-16-159:~$ mkdir inventory
ubuntu@ip-172-31-16-159:~$ vi inventory/aws_ec2.yaml
```

Session 2 is titled 'Ansible-Master' and is currently active, showing the command `vi inventory/aws_ec2.yaml`. The file editor interface is visible.

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 isolate the Ansible related dependencies by separating them from the environment.

Part 1: Create Virtual Environment

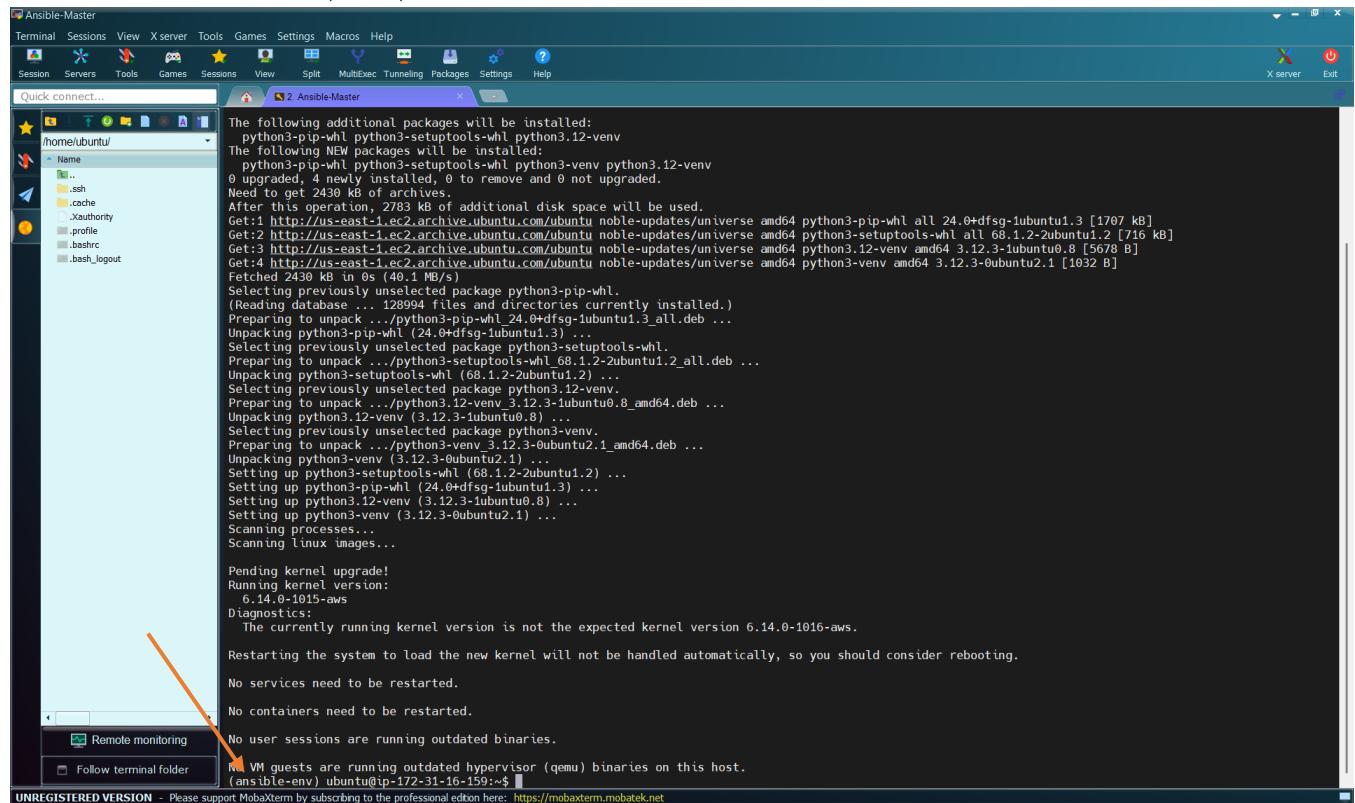
For that we will use a Python command to create our Virtual environment. For this, we will run the code which is going to install and enable the Virtual environment:

```
# 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
```



The screenshot shows a MobaXterm window titled "Ansible-Master". The terminal session is running on a remote host. The user has run the command "source ansible-env/bin/activate" and is now in the "ansible-env" virtual environment. The terminal output shows the installation of the "python3-venv" module and the creation of the virtual environment directory. A red arrow points from the text "You can see that our virtual environment has been created." to the line "Setting up python3-venv (3.12.3-0ubuntu2.1) ...". Another red arrow points to the message "VM guests are running outdated hypervisor (qemu) binaries on this host." at the bottom of the terminal window.

```
The following additional packages will be installed:
python3-pip-whl python3-setuptools-whl python3.12-venv
The following NEW packages will be installed:
python3-pip-whl python3-setuptools-whl python3-venv python3.12-venv
0 upgraded, 4 newly installed, 0 to remove and 0 not upgraded.
Need to get 2436 kB of archives.
After this operation, 2783 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 python3-pip-whl all 24.0+dfsg-1ubuntu1.3 [1707 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 python3-setuptools-whl all 68.1.2-2ubuntu1.2 [716 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 python3.12-venv amd64 3.12.3-1ubuntu0.8 [5678 B]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 python3-venv amd64 3.12.3-0ubuntu2.1 [1032 B]
Fetched 2436 kB in 0s (40.1 MB/s)
Selecting previously unselected package python3-pip-whl.
(Reading databases... 128994 files and directories currently installed.)
Preparing to unpack .../python3-pip-whl_24.0+dfsg-1ubuntu1.3_all.deb ...
Unpacking python3-pip-whl (24.0+dfsg-1ubuntu1.3) ...
Selecting previously unselected package python3-setuptools-whl.
Preparing to unpack .../python3-setuptools-whl_68.1.2-2ubuntu1.2_all.deb ...
Unpacking python3-setuptools-whl (68.1.2-2ubuntu1.2) ...
Selecting previously unselected package python3.12-venv.
Preparing to unpack .../python3.12-venv_3.12.3-1ubuntu0.8_amd64.deb ...
Unpacking python3.12-venv (3.12.3-1ubuntu0.8) ...
Selecting previously unselected package python3-venv.
Preparing to unpack .../python3-venv_3.12.3-0ubuntu2.1_amd64.deb ...
Unpacking python3-venv (3.12.3-0ubuntu2.1) ...
Setting up python3-setuptools-whl (68.1.2-2ubuntu1.2) ...
Setting up python3-pip-whl (24.0+dfsg-1ubuntu1.3) ...
Setting up python3.12-venv (3.12.3-1ubuntu0.8) ...
Setting up python3-venv (3.12.3-0ubuntu2.1) ...
Scanning processes...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
6.14.0-1015-aws
Diagnostics:
The currently running kernel version is not the expected kernel version 6.14.0-1016-aws.
Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.

VM guests are running outdated hypervisor (qemu) binaries on this host.
(ansible-env) ubuntu@ip-172-31-16-159:~$
```

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
```

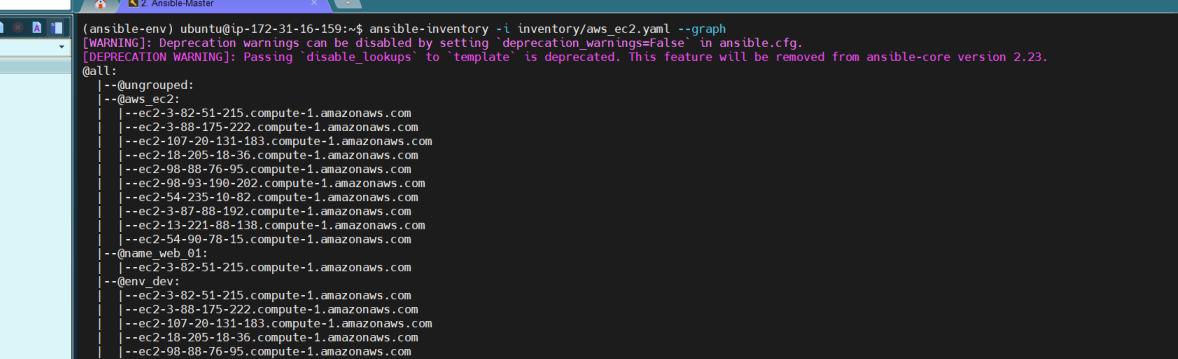
Prepared by Sidney Smith Ebot

```
Collecting botocore
  Downloading botocore-1.40.74-py3-none-any.whl.metadata (5.9 kB)
Collecting docker
  Downloading docker-7.1.0-py3-none-any.whl.metadata (3.8 kB)
Collecting jmespath<2.0.0,>=0.7.1 (from boto3)
  Downloading jmespath-1.0.1-py3-none-any.whl.metadata (7.6 kB)
Collecting s3transfer<0.15.0,>=0.14.0 (from boto3)
  Downloading s3transfer-0.14.0-py3-none-any.whl.metadata (1.7 kB)
Collecting python-dateutil<3.0.0,>=2.1 (from boto3)
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting urllib3!=2.2.0,<2.2.1,>=1.25.4 (from boto3)
  Downloading urllib3-2.5.0-py3-none-any.whl.metadata (6.5 kB)
Collecting requests<2.32.5-py3-none-any.whl> (from docker)
  Downloading requests-2.32.5-py3-none-any.whl.metadata (4.9 kB)
Collecting six<1.5 (from python-dateutil<3.0.0,>=2.1->botocore)
  Downloading six-1.17.0-py2.py3-none-any.whl.metadata (1.7 kB)
Collecting charset_normalizer<4,>=2 (from requests<2.26.0->docker)
  Downloading charset_normalizer-3.4.4-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl.metadata (37 kB)
Collecting idna<4,>=2.5 (from requests<2.26.0->docker)
  Downloading idna-3.11-py3-none-any.whl.metadata (8.4 kB)
Collecting certifi<=2017.4.17 (from requests<2.26.0->docker)
  Downloading certifi-2025.11.12-py3-none-any.whl.metadata (2.5 kB)
  Downloading boto3-1.40.74-py3-none-any.whl (139 kB)
    159.4/139.4 kB 12.9 MB/s eta 0:00:00
  Downloading botocore-1.40.74-py3-none-any.whl (141 kB)
    14.1/14.1 kB 84.6 MB/s eta 0:00:00
  Downloading docker-7.1.0-py3-none-any.whl (147 kB)
    147.8/147.8 kB 13.7 MB/s eta 0:00:00
  Downloading jmespath-1.0.1-py3-none-any.whl (28 kB)
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
    229.9/229.9 kB 28.0 MB/s eta 0:00:00
  Downloading requests-2.32.5-py3-none-any.whl (64 kB)
    64.7/64.7 kB 7.9 MB/s eta 0:00:00
  Downloading s3transfer-0.14.0-py3-none-any.whl (85 kB)
    85.7/85.7 kB 10.3 MB/s eta 0:00:00
  Downloading urllib3-2.5.0-py3-none-any.whl (129 kB)
    129.8/129.8 kB 18.4 MB/s eta 0:00:00
  Downloading certifi-2025.11.12-py3-none-any.whl (159 kB)
    159.4/159.4 kB 21.1 MB/s eta 0:00:00
  Downloading charset_normalizer-3.4.4-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl (153 kB)
    153.5/153.5 kB 20.0 MB/s eta 0:00:00
  Downloading idna-3.11-py3-none-any.whl (71 kB)
    71.0/71.0 kB 7.4 MB/s eta 0:00:00
  Downloading six-1.17.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: urllib3, six, jmespath, idna, charset_normalizer, certifi, requests, python-dateutil, docker, botocore, s3transfer, boto3, post0-requests>2.32.5 s3transfer<0.14.0 six-1.17.0 urllib3<2.5.0
Successfully installed boto3-1.40.74 botocore-1.40.74 certifi-2025.11.12 charset_normalizer-3.4.4 docker-7.1.0 idna-3.11 jmespath-1.0.1 python-dateutil-2.9.0, post0-requests>2.32.5 s3transfer<0.14.0 six-1.17.0 urllib3<2.5.0
(ansible-env) ubuntu@ip-172-31-16-159:~$
```

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
```



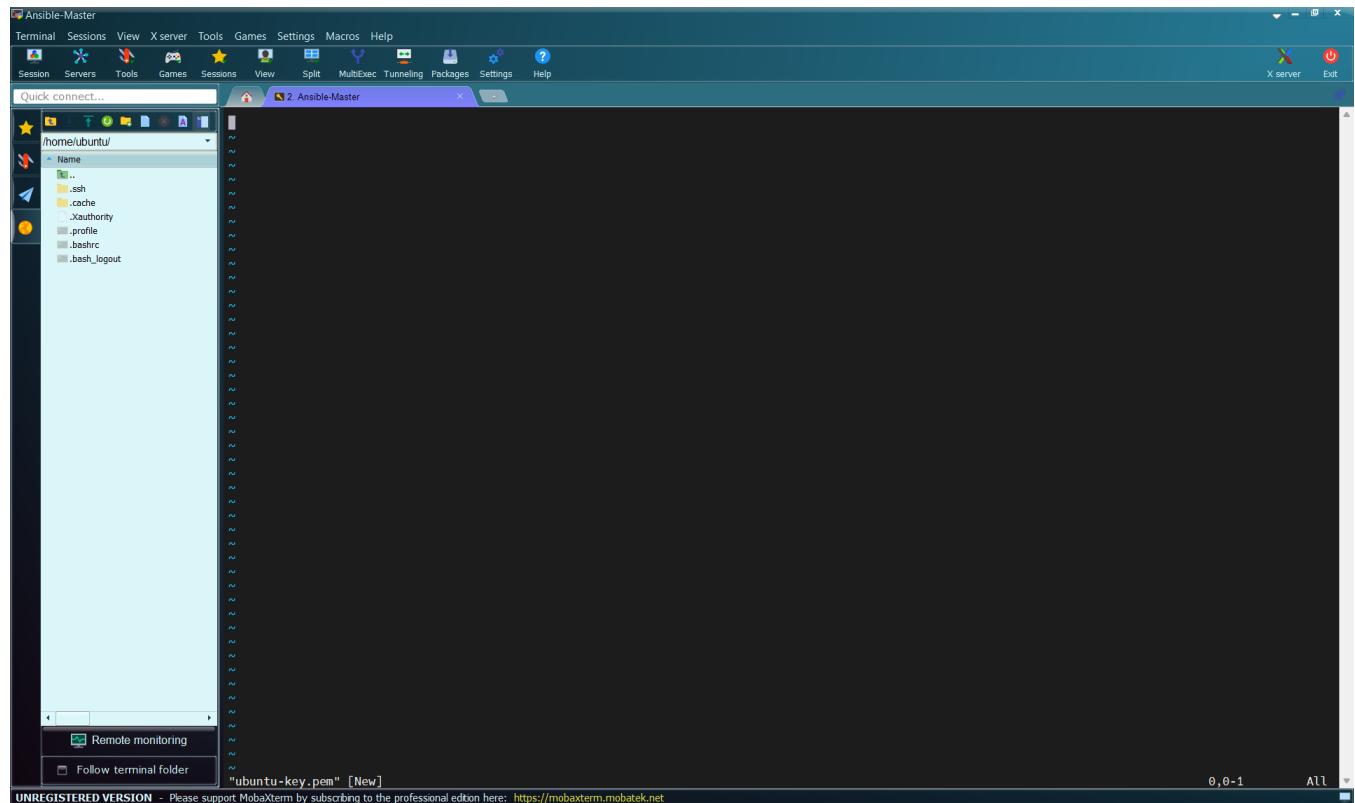
```
(ansible-env) ubuntu@ip-172-31-16-159:~$ ansible-inventory -i inventory/aws_ec2.yaml --graph
[WARNING]: Deprecation warnings can be disabled by setting 'deprecation_warnings=False' in ansible.cfg.
[DEPRECATION WARNING]: Passing 'disable_lookups' to 'template' is deprecated. This feature will be removed from ansible-core version 2.23.
@all:
  |--@ungrouped:
  |   |--@aws_ec2:
  |       |--ec2-3-82-51-215.compute-1.amazonaws.com
  |       |--ec2-3-88-175-222.compute-1.amazonaws.com
  |       |--ec2-107-20-131-183.compute-1.amazonaws.com
  |       |--ec2-18-205-18-36.compute-1.amazonaws.com
  |       |--ec2-98-88-76-95.compute-1.amazonaws.com
  |       |--ec2-98-93-190-202.compute-1.amazonaws.com
  |       |--ec2-54-235-10-82.compute-1.amazonaws.com
  |       |--ec2-3-87-88-192.compute-1.amazonaws.com
  |       |--ec2-13-221-88-138.compute-1.amazonaws.com
  |       |--ec2-54-99-78-15.compute-1.amazonaws.com
  |       @name_web_01:
  |           |--ec2-3-82-51-215.compute-1.amazonaws.com
  |       @env_dev:
  |           |--ec2-3-82-51-215.compute-1.amazonaws.com
  |           |--ec2-3-88-175-222.compute-1.amazonaws.com
  |           |--ec2-107-20-131-183.compute-1.amazonaws.com
  |           |--ec2-18-205-18-36.compute-1.amazonaws.com
  |           |--ec2-98-88-76-95.compute-1.amazonaws.com
  |           |--ec2-98-93-190-202.compute-1.amazonaws.com
  |           |--ec2-54-235-10-82.compute-1.amazonaws.com
  |           |--ec2-3-87-88-192.compute-1.amazonaws.com
  |           |--ec2-13-221-88-138.compute-1.amazonaws.com
  |           |--ec2-54-99-78-15.compute-1.amazonaws.com
  |       @name_web_02:
  |           |--ec2-3-88-175-222.compute-1.amazonaws.com
  |       @name_web_03:
  |           |--ec2-107-20-131-183.compute-1.amazonaws.com
  |       @name_web_04:
  |           |--ec2-18-205-18-36.compute-1.amazonaws.com
  |       @name_web_05:
  |           |--ec2-98-88-76-95.compute-1.amazonaws.com
  |       @name_web_06:
  |           |--ec2-98-93-190-202.compute-1.amazonaws.com
  |       @name_web_07:
  |           |--ec2-54-235-10-82.compute-1.amazonaws.com
  |       @name_web_08:
  |           |--ec2-3-87-88-192.compute-1.amazonaws.com
  |       @name_web_09:
  |           |--ec2-13-221-88-138.compute-1.amazonaws.com
  |       @name_web_10:
  |           |--ec2-54-99-78-15.compute-1.amazonaws.com
(ansible-env) ubuntu@ip-172-31-16-159:~$
```

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
```



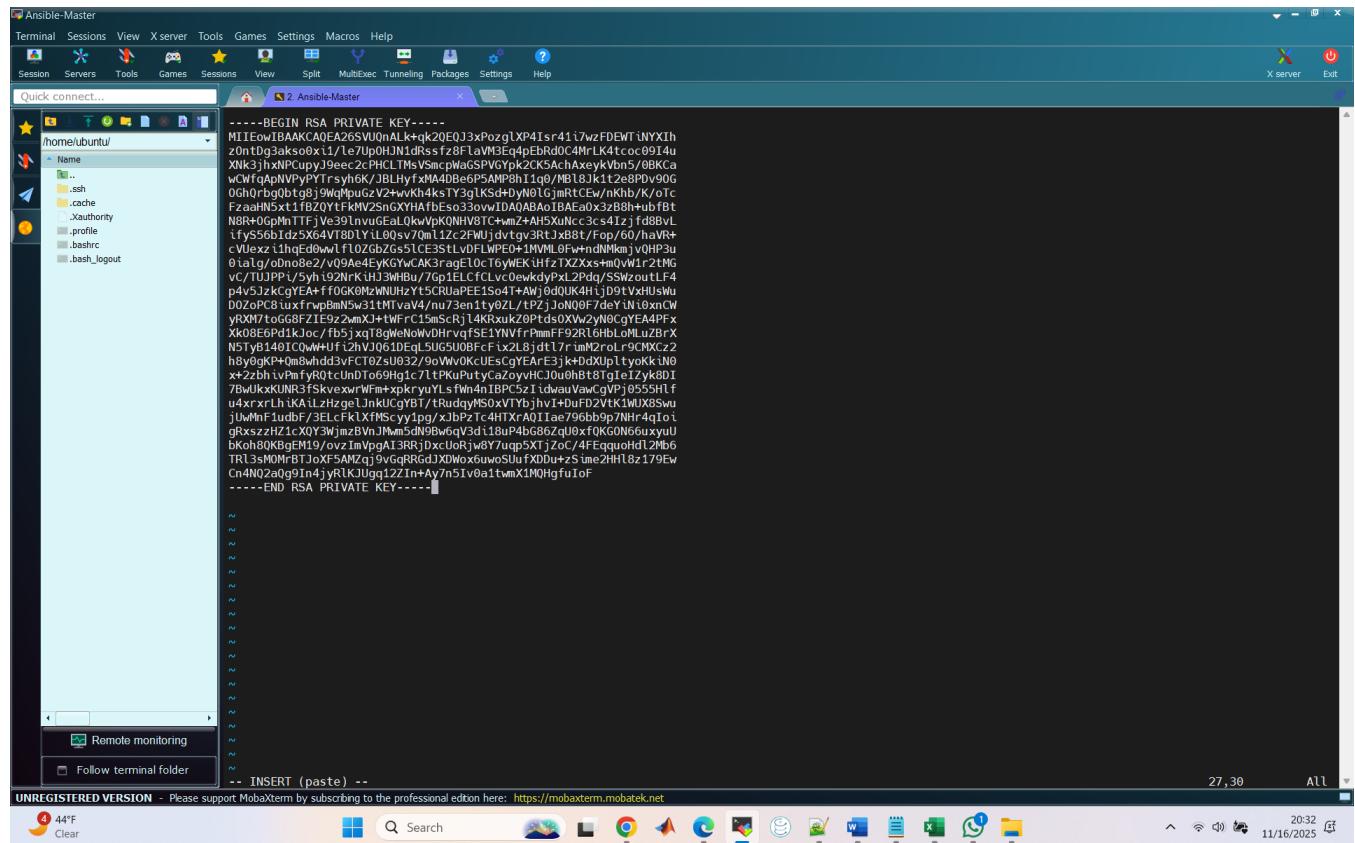
Go to our .pem file and open it in Notepad

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA26SVUQnALK+qk2QEJ3xPozglXP4Isr4ii7wzFDEWTiNYXi
z0ntDg3aksod0xi1i/le7UpOHJN1dRssfz8flavM3Eq4pbRdc04MrLk4tcc09i4u
XNk3jhjxhPCUpy39eeC2cPHCLTMsVSmcPwgASPVGYpk2c53AchAxeVkbvn5/08KCa
wCwfqaphIVPyPTTrsyh6K/JBLHytfxMA4DBe6p5AMP8h1i0q/0MB18jK1t2e8PDv90G
OGhqrbg0btg8j9WQmpuGzV2+wvhkh4ksTy3g1KSD+DyN0lGjmrtCEw/nKhB/K/oTc
FzaaHN5xt1fbZQYfKMV2SnGXyHafBES030vwIDAQABAOIBAEa0x32BBh+ubft
N8R+0gpmtTTfjve391nuvGEalQkwvpQNHV8TC+w+AHSXUNCC3cs4Izjfd8BvL
1fy56bibd25X64VT8D1Y1L0qsV7Qm112c2FWUjdvtgv3RTJXB8t/Fop/60/havR+
cVexzijihgEd0wwlf1oZgbzG51CE35tLvdFLPEO+1VML0Fw+ndNNkmkjQHP3u
0ialg/obno8e2/v094e4EyKGwCAK3ragflocT6ywEkiHf2TXZXxs+mQwlrl2tMG
vc/TUJPPi/Syhi92NrKiH3WhBu/7Gp1ELCfClvcOewkdypxL2Pdq+SSWzoutLF4
p4V5j2kcgYEAtffOGK0M2NUHizytsCRUapEE1so4t-Awjed0quK4HiJd97VxHuswU
D0ZoPC81uxfrwpBw531tTVa/4/nu73en1ty0zL/tbZjDN00F7deyNi0xncw
yRMX7t0gGBFZIE922wmX3+tWFrC15mScRj14KRxuzk0ptdsOXv2y2NOGcYEAPFx
Xk0BE6Pd1kJoc/fb5jxq78gwewNoDhrVqfSe1YmVfrPmmFF92R16hbLoMLU2BrX
N5TyB140ICQwHufi2hV3Q61DqELsUG5u0BFcfi2L8jdt1r7im2rolrL9CMKCZ2
haygkP+0m8whdd3vFCt0ZsU03/9oWwv0KcUEScgYArE3jk/DdxUp1tyKkIn0
x+22bhivPmfvRqtCudnTo69Hg1c7ltpKwPutycaoyvHJ0u0hb7TgeIzky80I
7BWlkKUNR3fskvexwrWfm+xpkrlyUlsfw4n1BPC5zidwauvawCvgPj0555Hif
u4xxrLhikAilHzge1JnkudByBT/trudyMSoxTYbjh1/Dd2VtK1WUx8Swu
juWnf1udbF/3ELCfkLXfMScy1pg/xjbpZt4HTXrAOIIaet796bb9p77Hr4q4Io1
gRxszzH21cXQY3WjmzbVnJlw5dnBw6qv3di18uPAbG86ZgQu0xFKG0N66uxyu
bk0h8QKBgEM19/ovzImVpgA13RRj0xcuRjw8Y7uqp5XTjZoc/4FEquuoHd12m6
TRJ3sMOHbTJ0XF5MAzqj9vGgRRGdjxDwoxuwosUufxDdu+zSime2HH182179EW
Cn4NQ2aqg9In4jyR1KJUgq122In+Ay7n5iv0a1twmX1QHgfuiOf
-----END RSA PRIVATE KEY-----
```

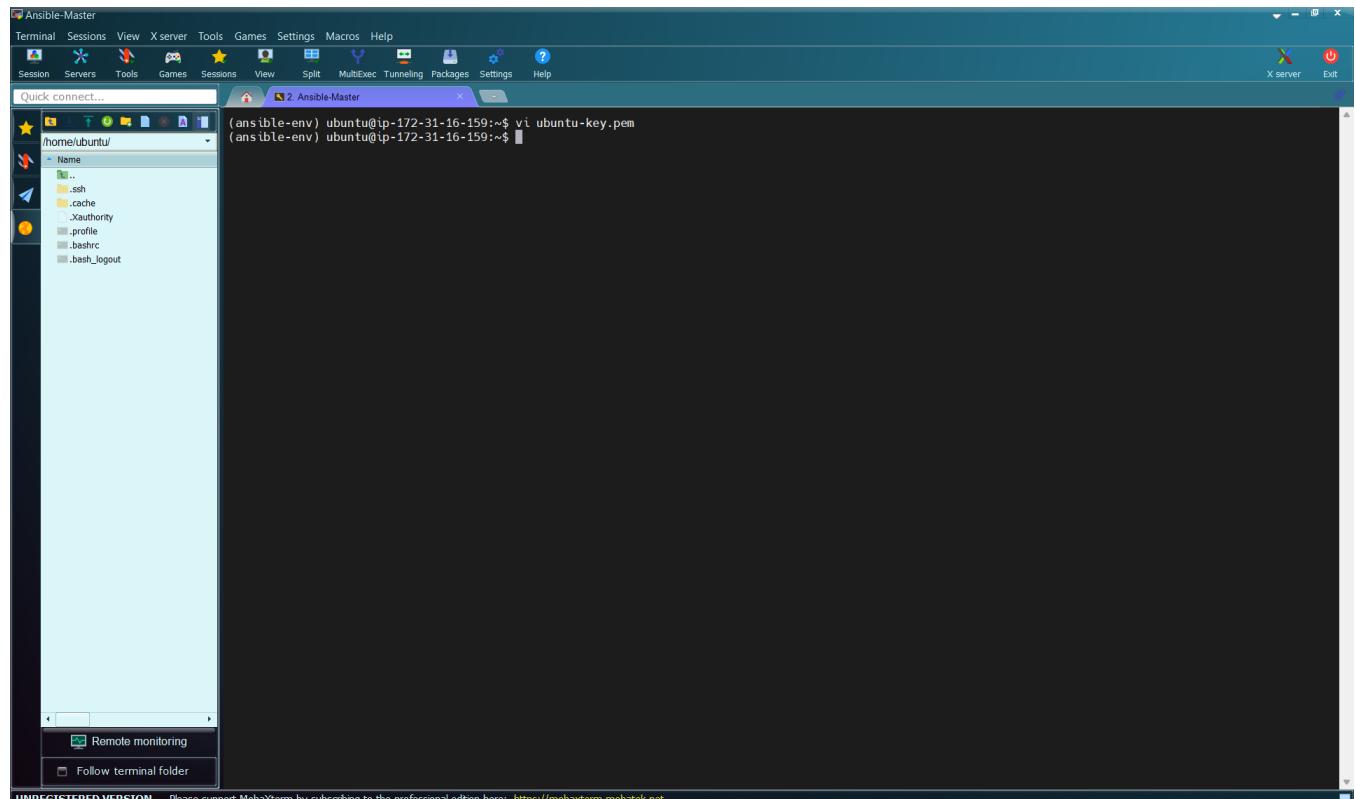
Ln 1, Col 1 1,674 characters Plain text 100% Unix (LF) UTF-8

Copy the code in this file and paste

Prepared by Sidney Smith Ebot

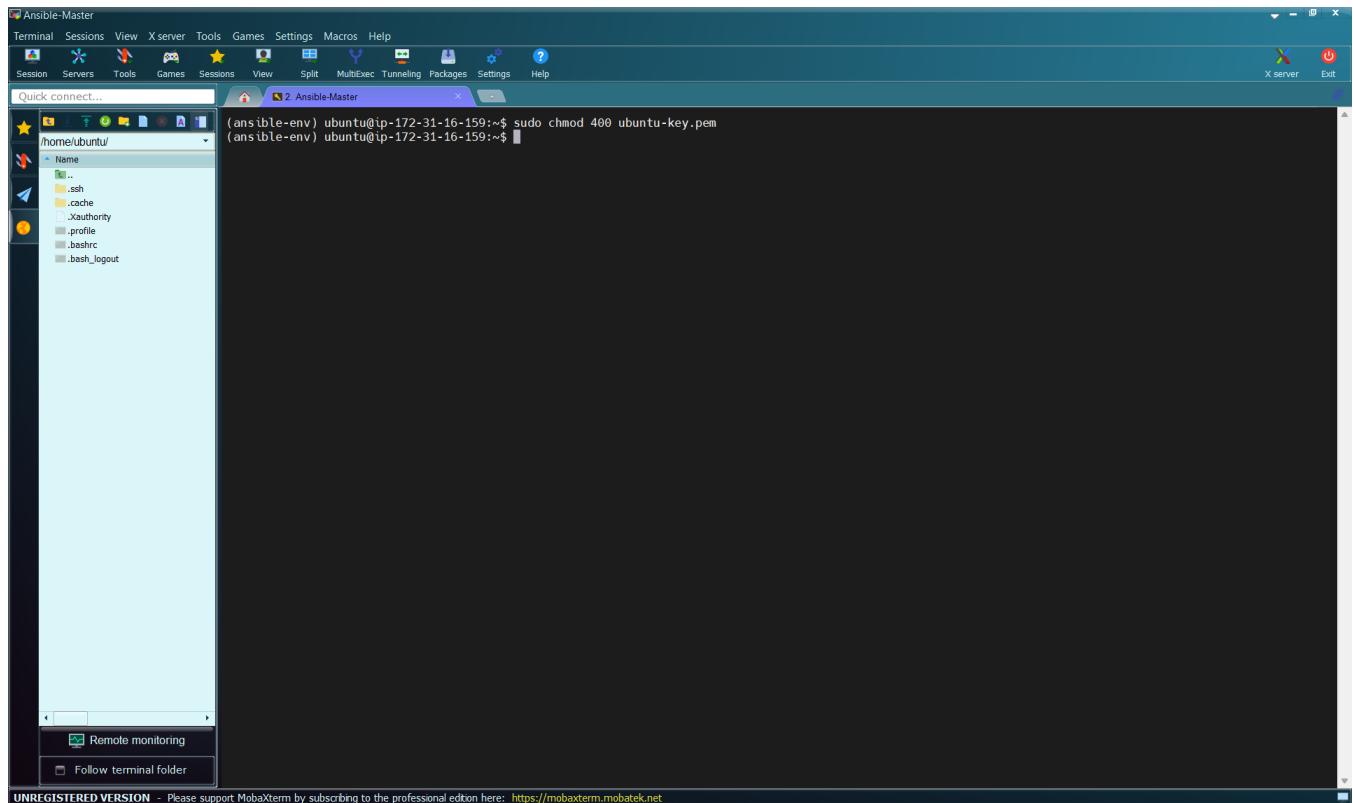


Save and Exit the code by pressing ESC followed by :wq and Press “Enter”



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
```

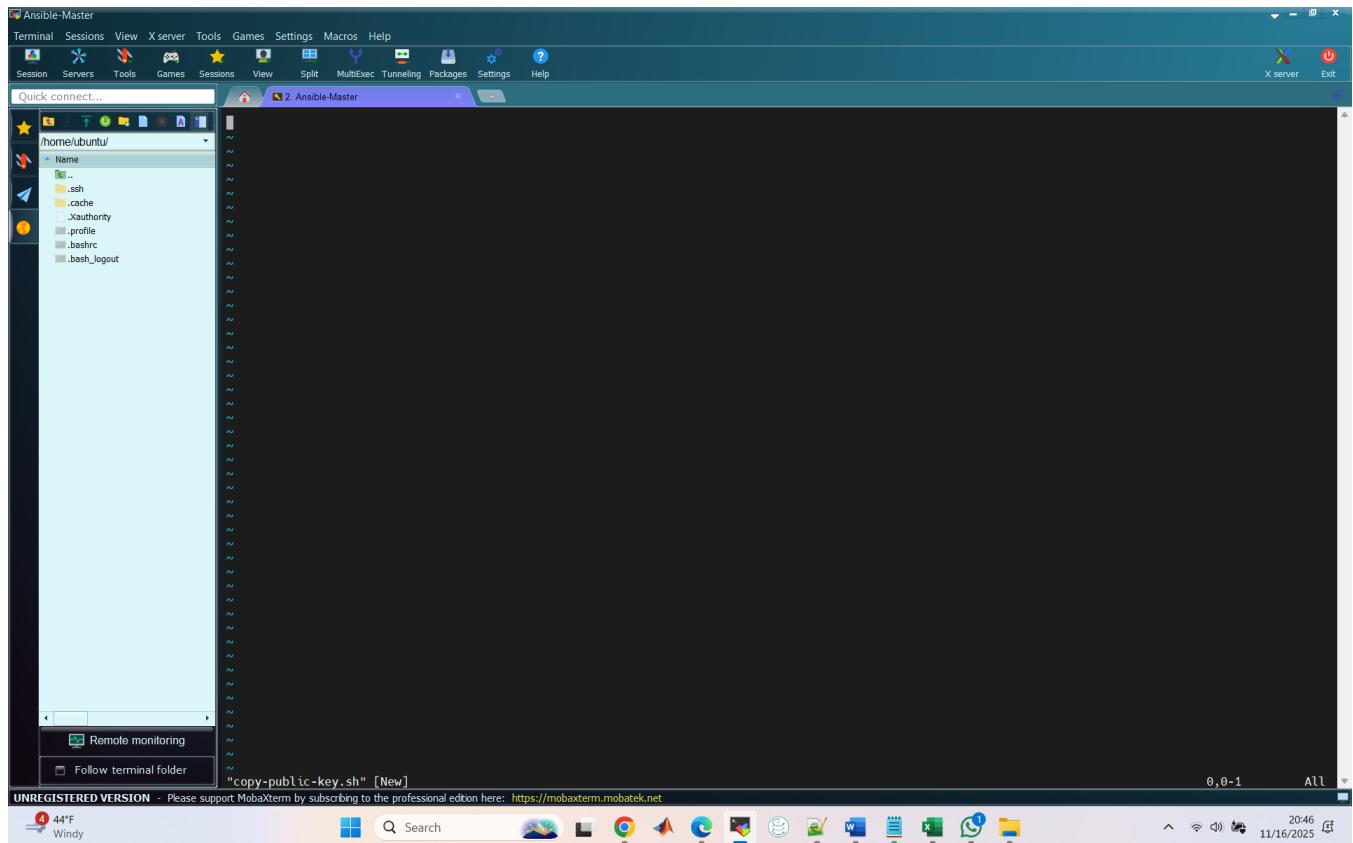


Part 5: Copy the Public Key

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

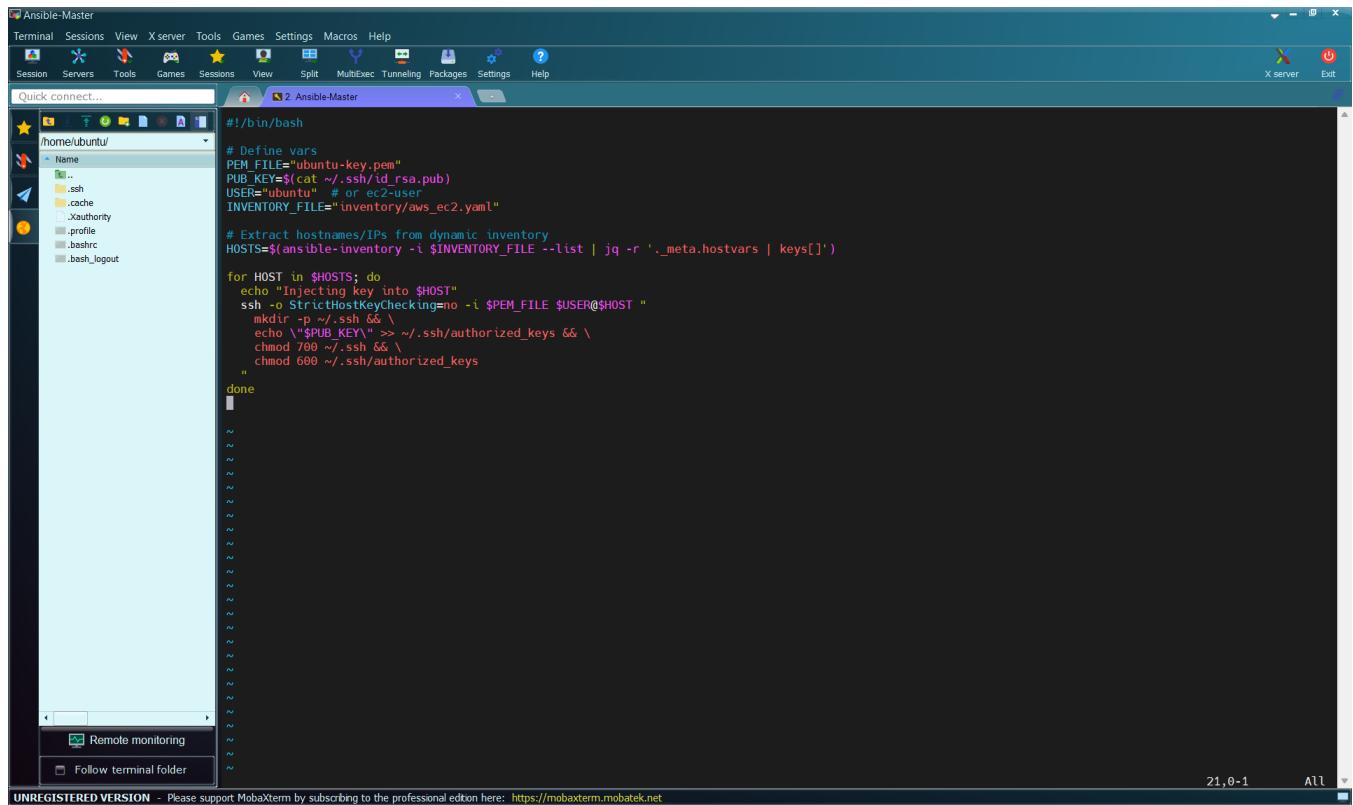
```
#!/bin/bash

# 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/authorized_keys && \
        chmod 600 ~/.ssh/authorized_keys
    "
done
```

Prepared by Sidney Smith Ebot



The screenshot shows a MobaXterm window titled "Ansible-Master". The terminal session is titled "Ansible-Master". The script being run is as follows:

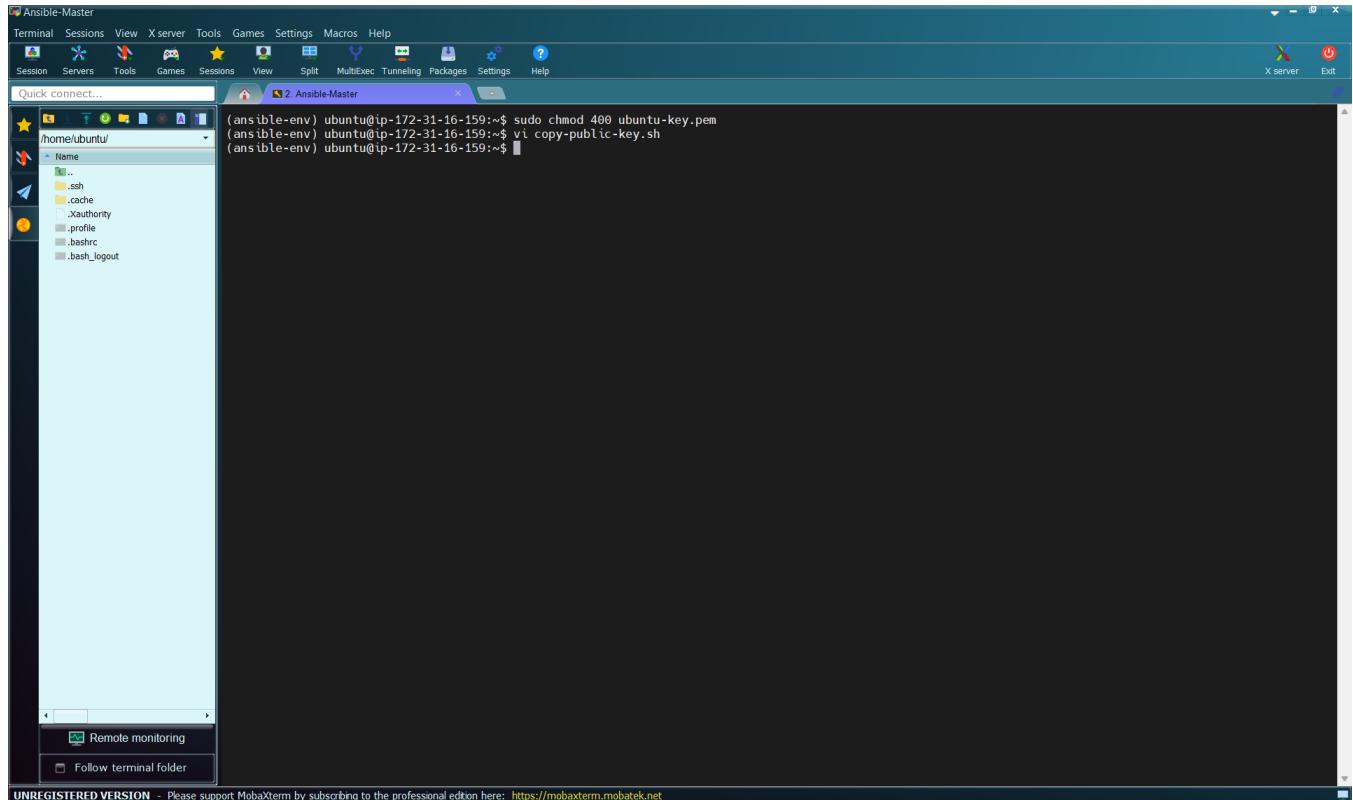
```
#!/bin/bash
# Define vars
PEM_FILE="ubuntu-key.pem"
PUB_KEY=$(cat ~/.ssh/id_rsa.pub)
USER="ubuntu" # or ec2-user
INVENTORY_FILE="inventory/aws_ec2.yaml"

# Extract hostnames/ IPs from dynamic inventory
HOSTS=$(ansible-inventory -t $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
```

The terminal window also displays a file tree on the left showing the user's home directory (~) containing .ssh, .cache, .Xauthority, .profile, .bashrc, and .bash_logout.

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



The screenshot shows a MobaXterm window titled "Ansible-Master". The terminal session is titled "Ansible-Master". The command entered was:

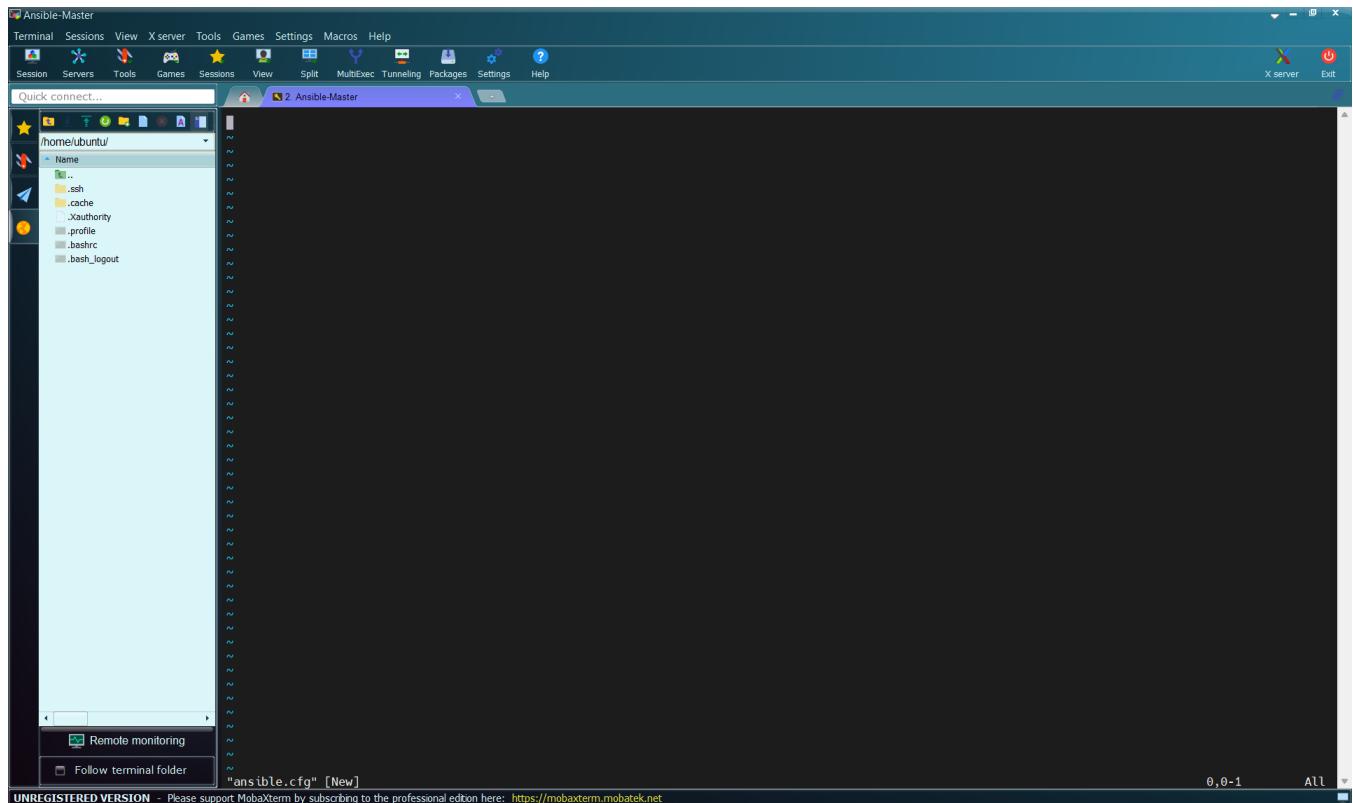
```
(ansible-env) ubuntu@ip-172-31-16-159:~$ sudo chmod 400 ubuntu-key.pem
(ansible-env) ubuntu@ip-172-31-16-159:~$ vi copy-public-key.sh
(ansible-env) ubuntu@ip-172-31-16-159:~$
```

The terminal window also displays a file tree on the left showing the user's home directory (~) containing .ssh, .cache, .Xauthority, .profile, .bashrc, and .bash_logout.

NOTE: sometimes when you run a script, there is a prompt that asks 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
```

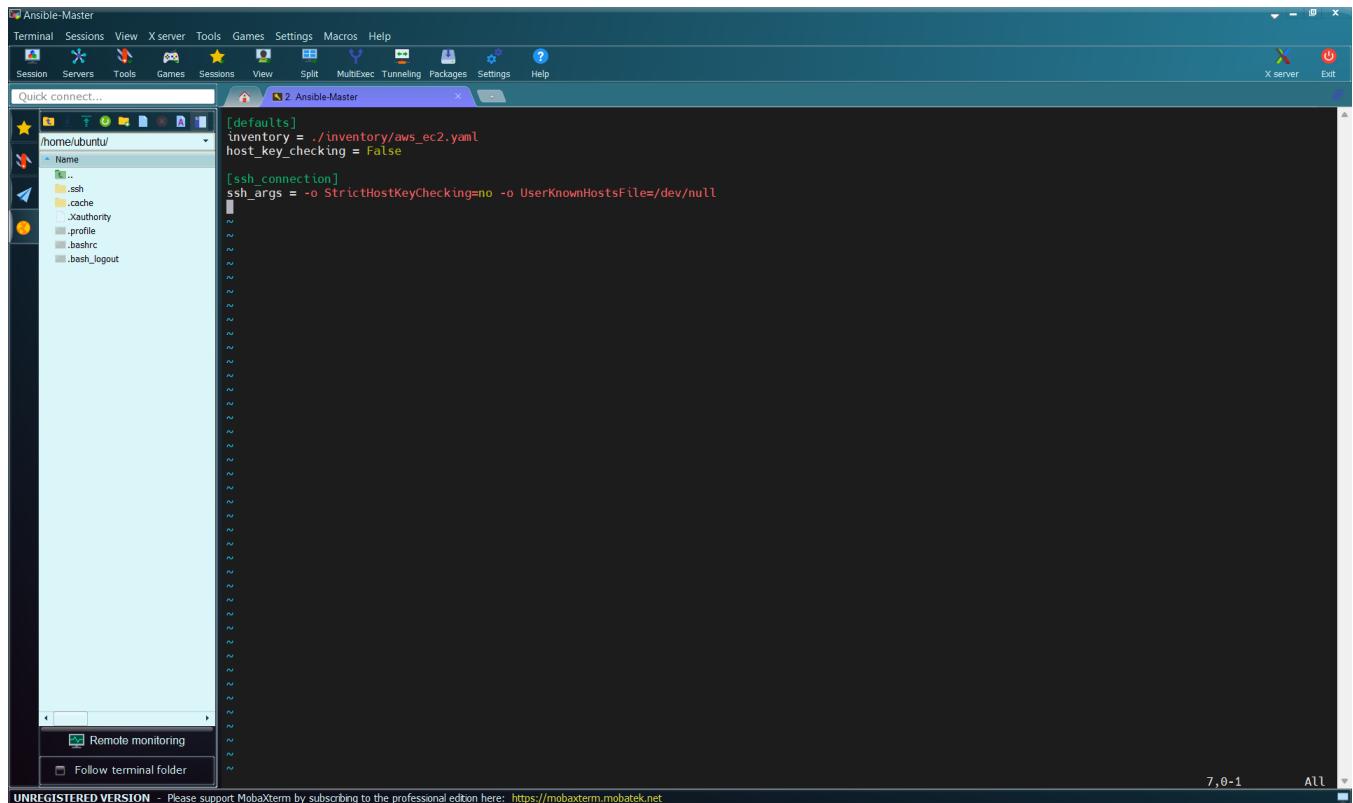


Then, paste the code:

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

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

Prepared by Sidney Smith Ebot



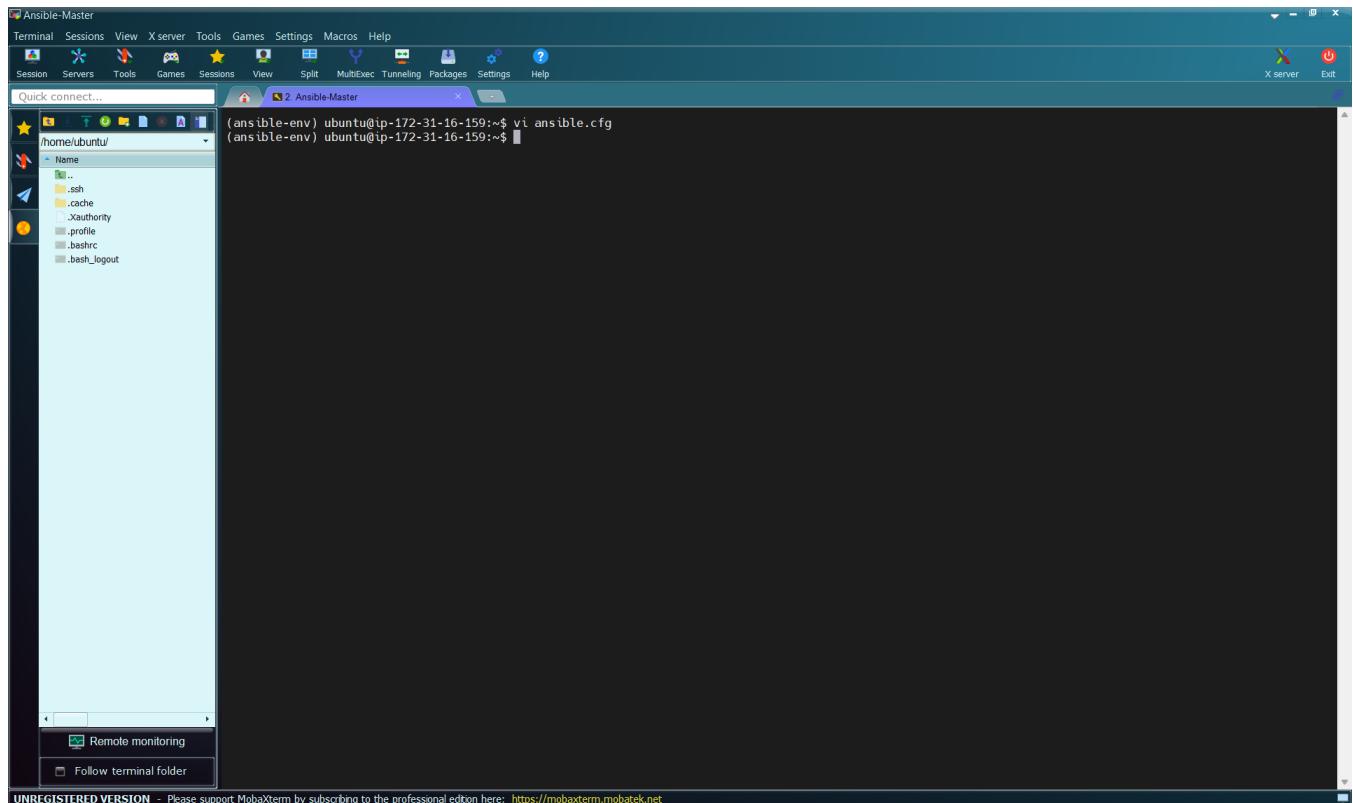
The screenshot shows the MobaXterm interface with a terminal window titled "Ansible-Master". The terminal displays the following Ansible configuration:

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

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

The terminal window has a dark background and light-colored text. The left sidebar shows a file tree under "/home/ubuntu/" with files like ".ssh", ".cache", ".Xauthority", ".profile", ".bashrc", and ".bash_logout". A context menu is open at the bottom of the terminal window, showing options like "Remote monitoring" and "Follow terminal folder". The status bar at the bottom indicates "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

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



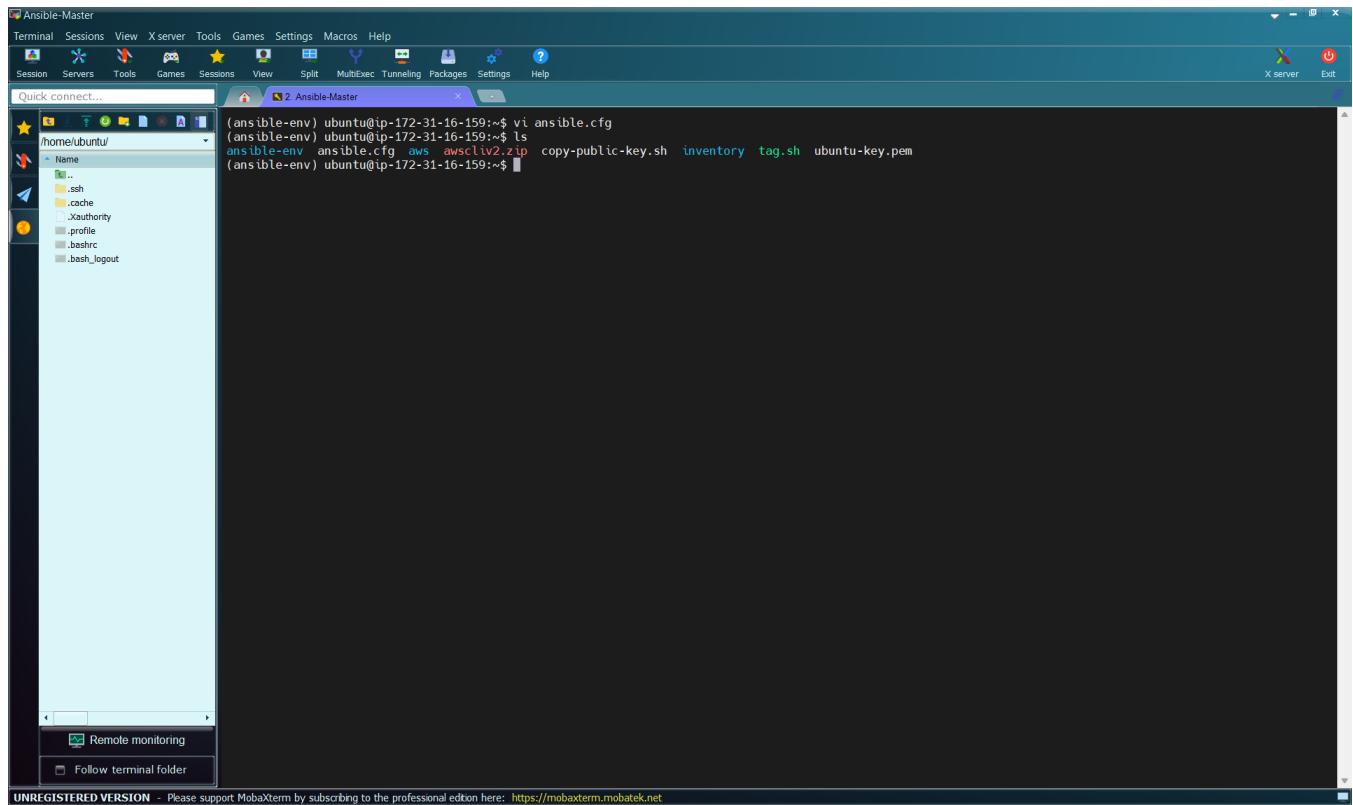
The screenshot shows the MobaXterm interface with a terminal window titled "Ansible-Master". The terminal displays the command:

```
(ansible-env) ubuntu@ip-172-31-16-159:~$ vi ansible.cfg
(ansible-env) ubuntu@ip-172-31-16-159:~$
```

The terminal window has a dark background and light-colored text. The left sidebar shows a file tree under "/home/ubuntu/" with files like ".ssh", ".cache", ".Xauthority", ".profile", ".bashrc", and ".bash_logout". A context menu is open at the bottom of the terminal window, showing options like "Remote monitoring" and "Follow terminal folder". The status bar at the bottom indicates "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

Check the content using the command:

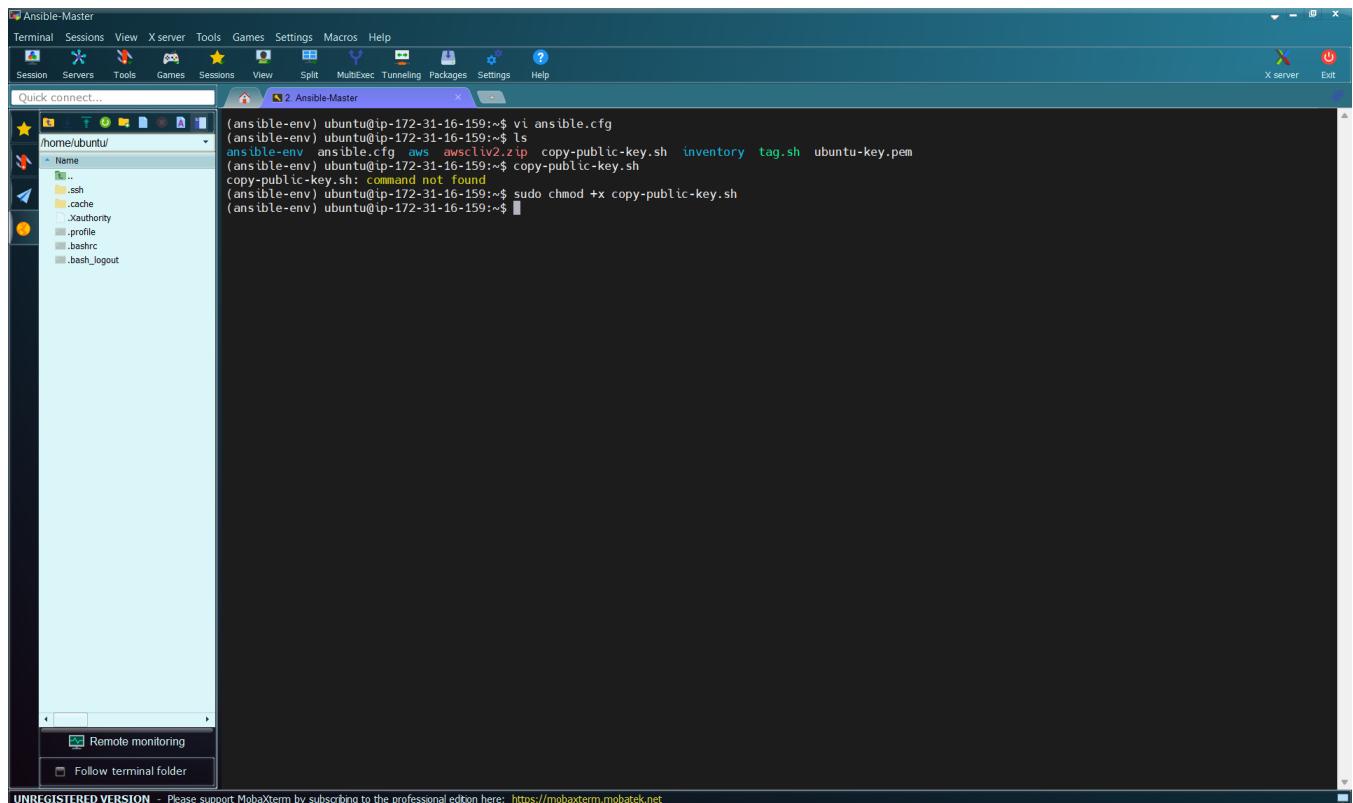
ls



```
(ansible-env) ubuntu@ip-172-31-16-159:~$ vi ansible.cfg
(ansible-env) ubuntu@ip-172-31-16-159:~$ ls
ansible.cfg aws awscli2.zip copy-public-key.sh inventory tag.sh ubuntu-key.pem
(ansible-env) ubuntu@ip-172-31-16-159:~$
```

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

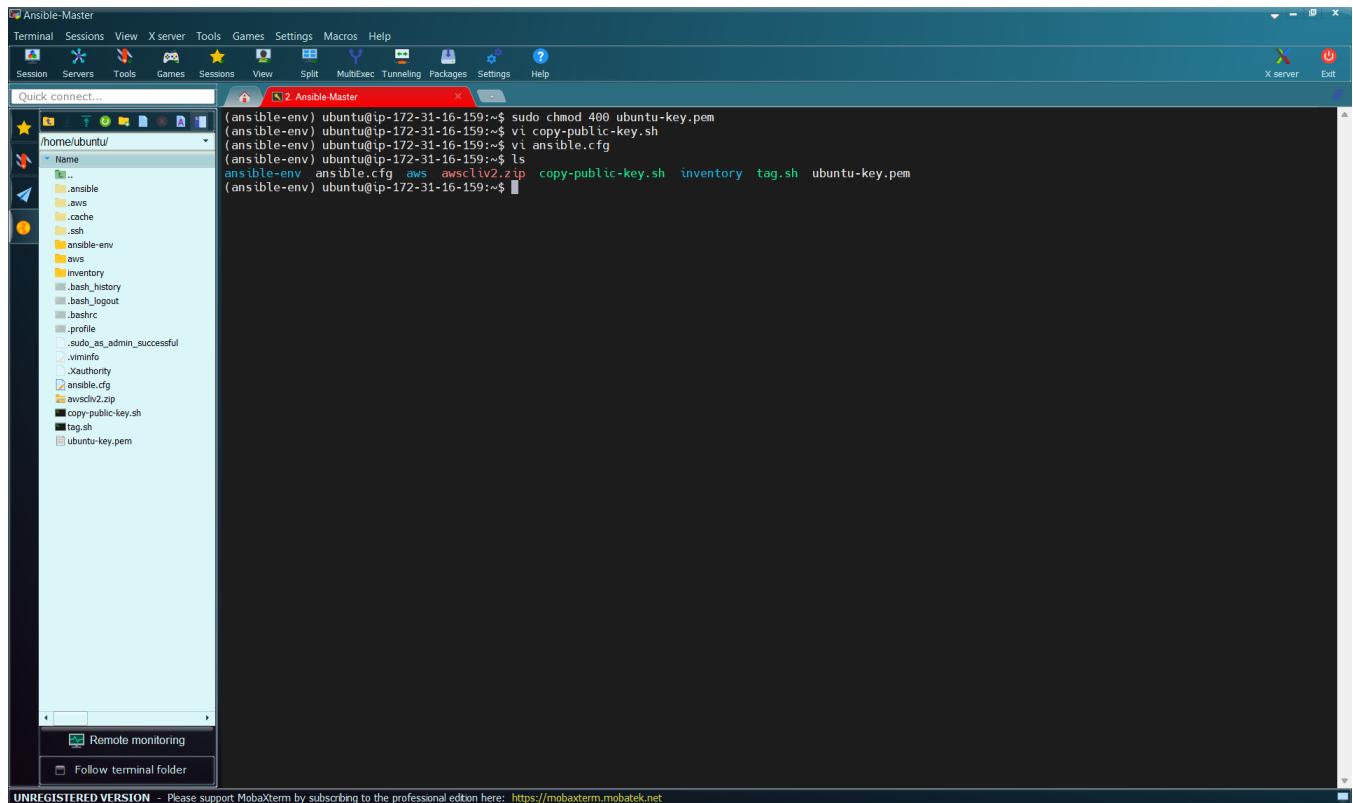
```
sudo chmod +x copy-public-key.sh
```



```
(ansible-env) ubuntu@ip-172-31-16-159:~$ vi ansible.cfg
(ansible-env) ubuntu@ip-172-31-16-159:~$ ls
ansible.cfg aws awscli2.zip copy-public-key.sh inventory tag.sh ubuntu-key.pem
(copy-public-key.sh: command not found)
(ansible-env) ubuntu@ip-172-31-16-159:~$ sudo chmod +x copy-public-key.sh
(ansible-env) ubuntu@ip-172-31-16-159:~$
```

Then check again using the command:

ls



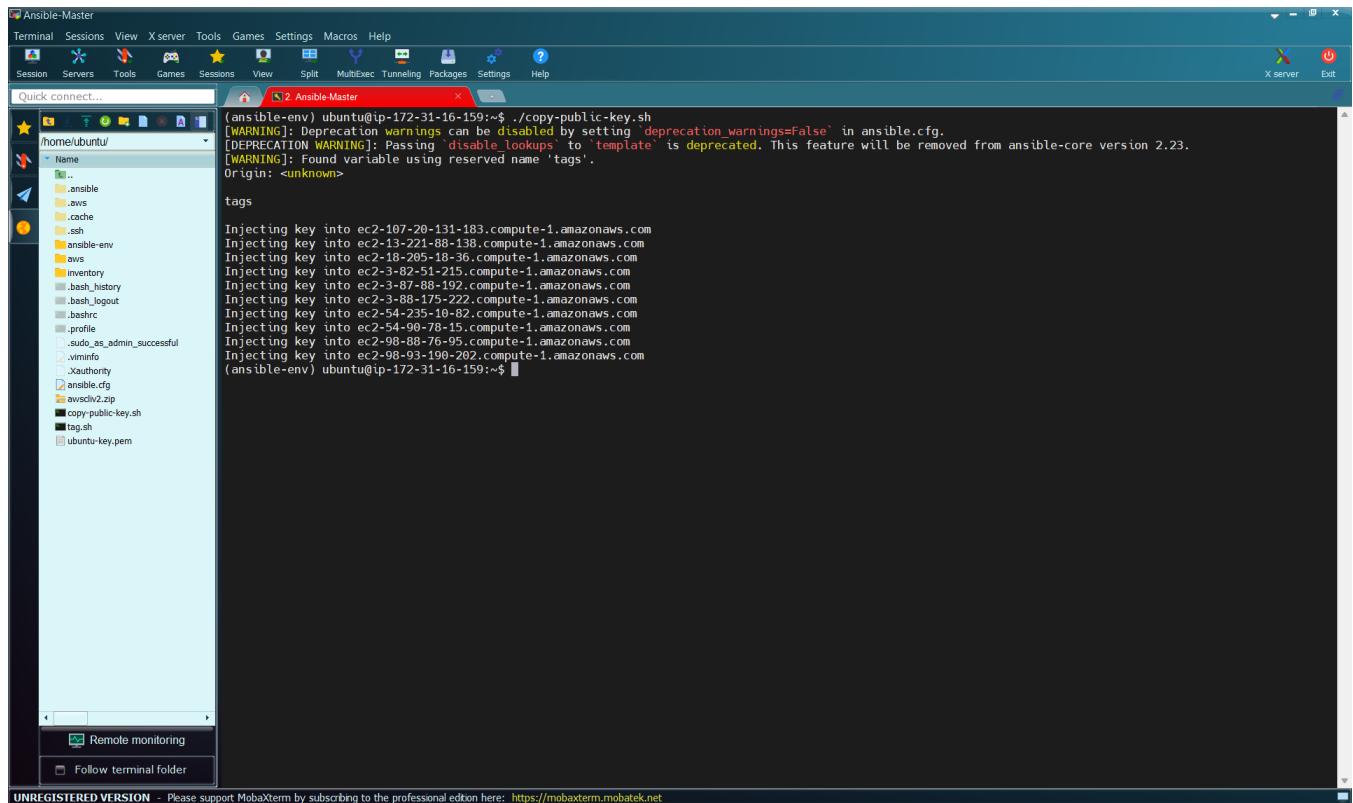
The screenshot shows a terminal window titled "Ansible-Master" in MobaXterm. The terminal is displaying the command "ls" and its output. The output shows several files and directories in the current directory:

```
(ansible-env) ubuntu@ip-172-31-16-159:~$ ls
ansible  aws  cache  .ssh  ansible.cfg
aws      awscli2.zip  copy-public-key.sh  inventory  tag.sh  ubuntu-key.pem
ansible-env
aws      awscli2.zip  copy-public-key.sh  inventory  tag.sh  ubuntu-key.pem
ansible  aws  cache  .ssh  ansible.cfg
aws      awscli2.zip  copy-public-key.sh  inventory  tag.sh  ubuntu-key.pem
```

The terminal window also includes a file browser sidebar on the left and a status bar at the bottom.

Then execute the file using the command:

./copy-public-key.sh



The screenshot shows a terminal window titled "Ansible-Master" in MobaXterm. The terminal is displaying the command "./copy-public-key.sh" and its output. The output shows the script injecting keys into various AWS instances:

```
(ansible-env) ubuntu@ip-172-31-16-159:~$ ./copy-public-key.sh
[WARNING]: Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
[DEPRECATION WARNING]: Passing `disable_lookups` to `template` is deprecated. This feature will be removed from ansible-core version 2.23.
[WARNING]: Found variable using reserved name 'tags'.
Original: <unknown>

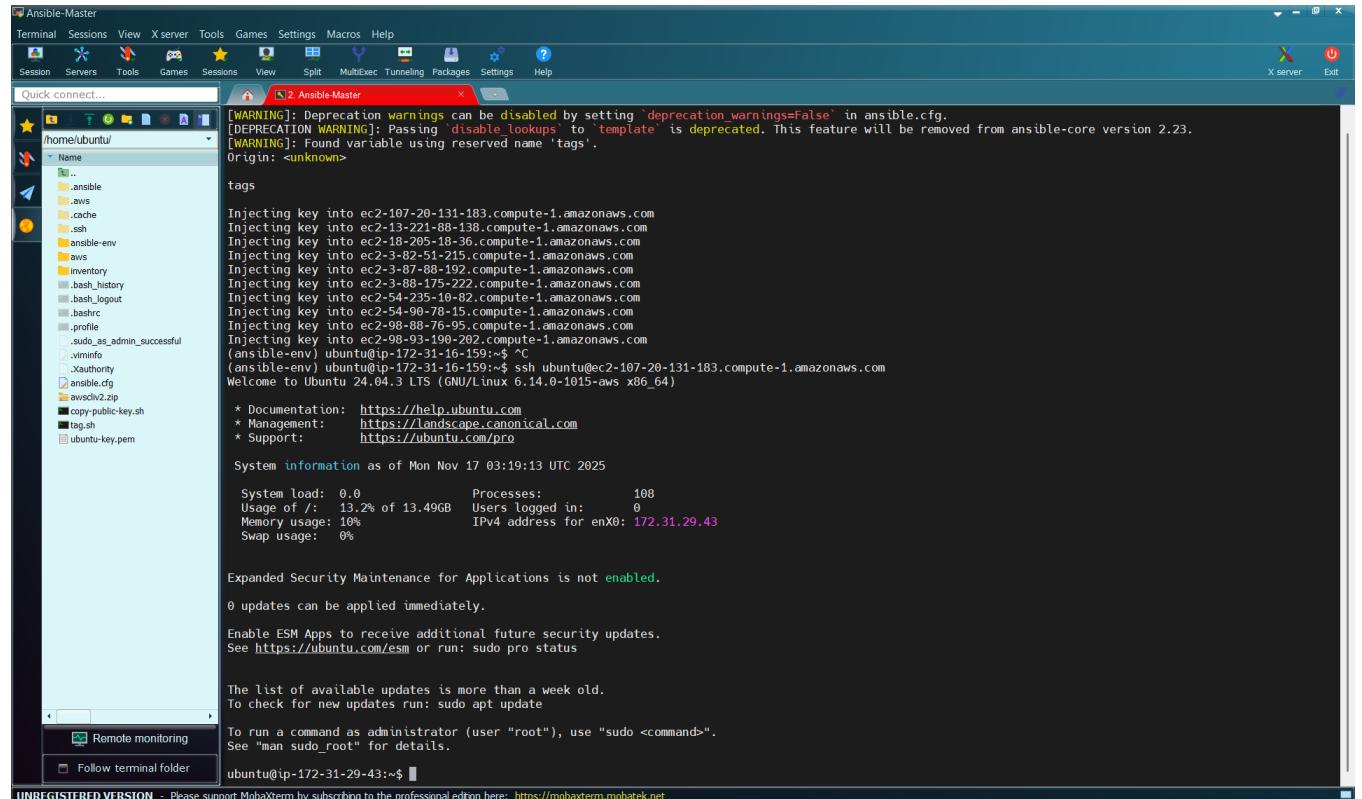
tags
Injecting key into ec2-107-20-131-183.compute-1.amazonaws.com
Injecting key into ec2-13-221-88-138.compute-1.amazonaws.com
Injecting key into ec2-18-205-18-36.compute-1.amazonaws.com
Injecting key into ec2-3-82-51-215.compute-1.amazonaws.com
Injecting key into ec2-3-87-88-192.compute-1.amazonaws.com
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:~$
```

The terminal window also includes a file browser sidebar on the left and a status bar at the bottom.

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
```



```
[WARNING]: Deprecation warnings can be disabled by setting `deprecation_warnings=False` in ansible.cfg.
[DEPRECATION WARNING]: Passing `disable_lookups` to `template` is deprecated. This feature will be removed from ansible-core version 2.23.
[WARNING]: Found variable using reserved name 'tags'.
  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:~$ 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          Processes:          108
Usage of /: 13.2% of 13.49GB Users logged in: 0
Memory usage: 10%          IPv4 address for enx0: 172.31.29.43
Swap usage: 0%             

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:~$
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

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
```

Prepared by Sidney Smith Ebot

The screenshot shows a terminal window titled "Ansible-Master" in MobaXterm. The session path is "/home/ubuntu/". The terminal displays the output of an Ansible command, specifically "ansible -i inventory all -m ping". The output shows that the key is being injected into multiple AWS instances (ec2-107-20-131-183.compute-1.amazonaws.com, ec2-13-221-88-138.compute-1.amazonaws.com, etc.) and then connects to the first instance via SSH.

```
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      Processes:          108
Usage of /: 13.2% of 13.49GB   Users logged in:    0
Memory usage: 10%           IPv4 address for enx0: 172.31.29.43
Swap usage:  0%

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:~$
```

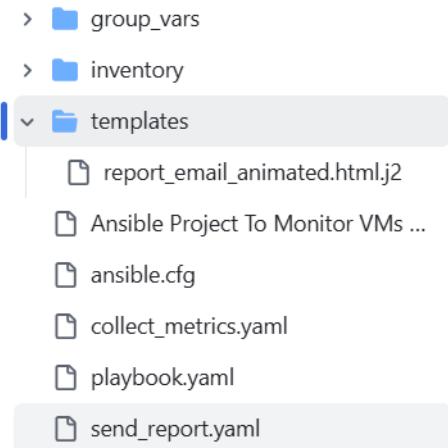
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

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.

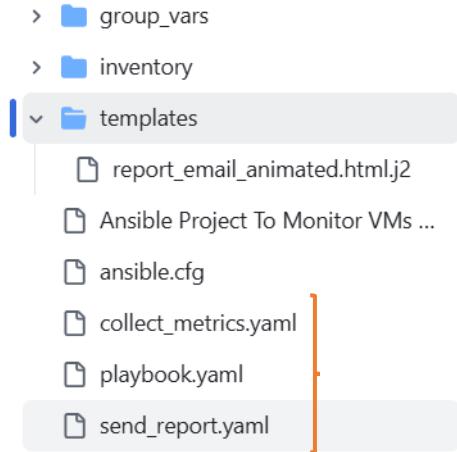
In this file, we are going to define our variables and also in the variables what things do we define. Basically, we are going to be defining information about the email, to where the email should be sent, email ID, of where we are going to send the report and a password of the email. The password is 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 reason, 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
```

```
(ansible-env) ubuntu@ip-172-31-16-159:~$ mkdir vm-monitor
(ansible-env) ubuntu@ip-172-31-16-159:~$
```

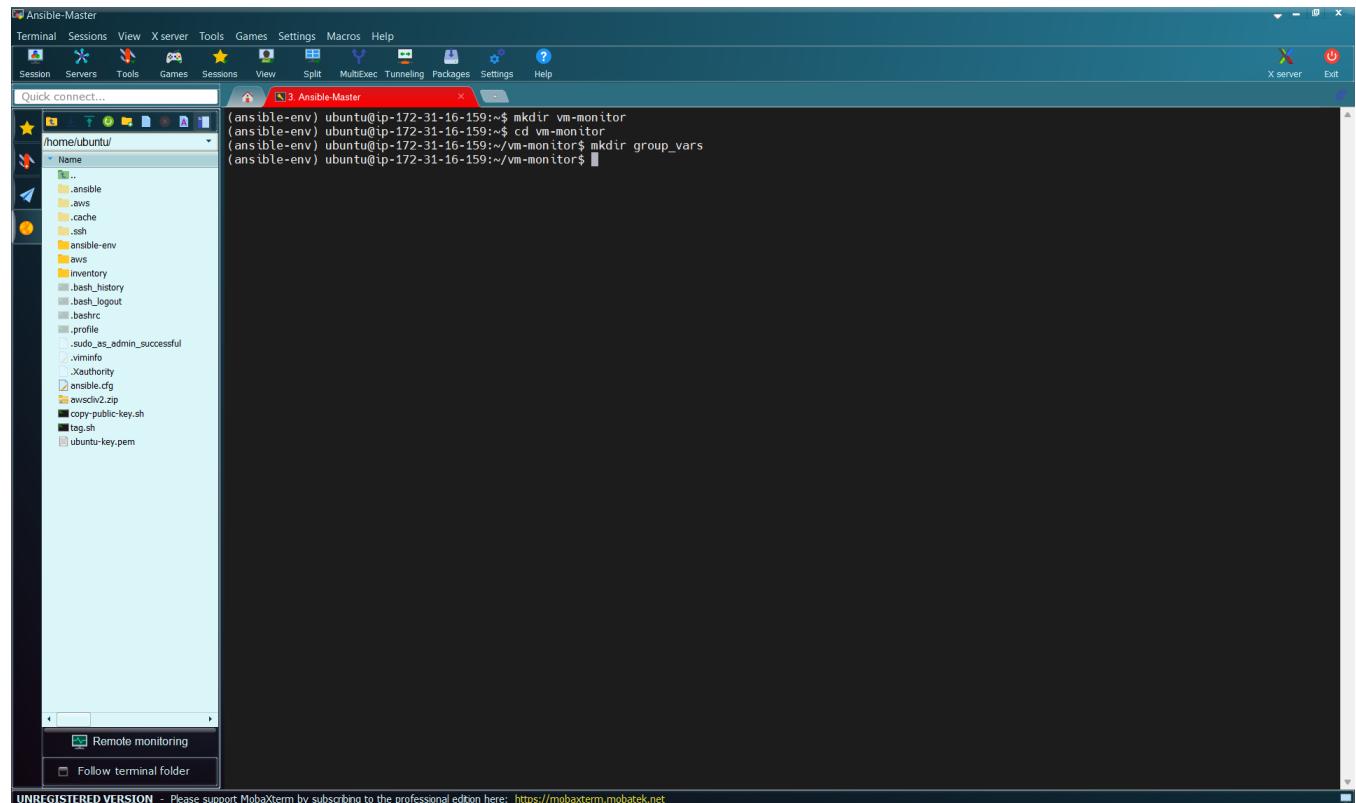
Then next we create the “**group_vars**” folder inside the project folder. Let us navigate to the project folder using the command:

```
cd vm-monitor
```

```
(ansible-env) ubuntu@ip-172-31-16-159:~$ mkdir vm-monitor
(ansible-env) ubuntu@ip-172-31-16-159:~$ cd vm-monitor
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$
```

Then, we create the folder “**group_vars**” using the command:

```
mkdir group_vars
```



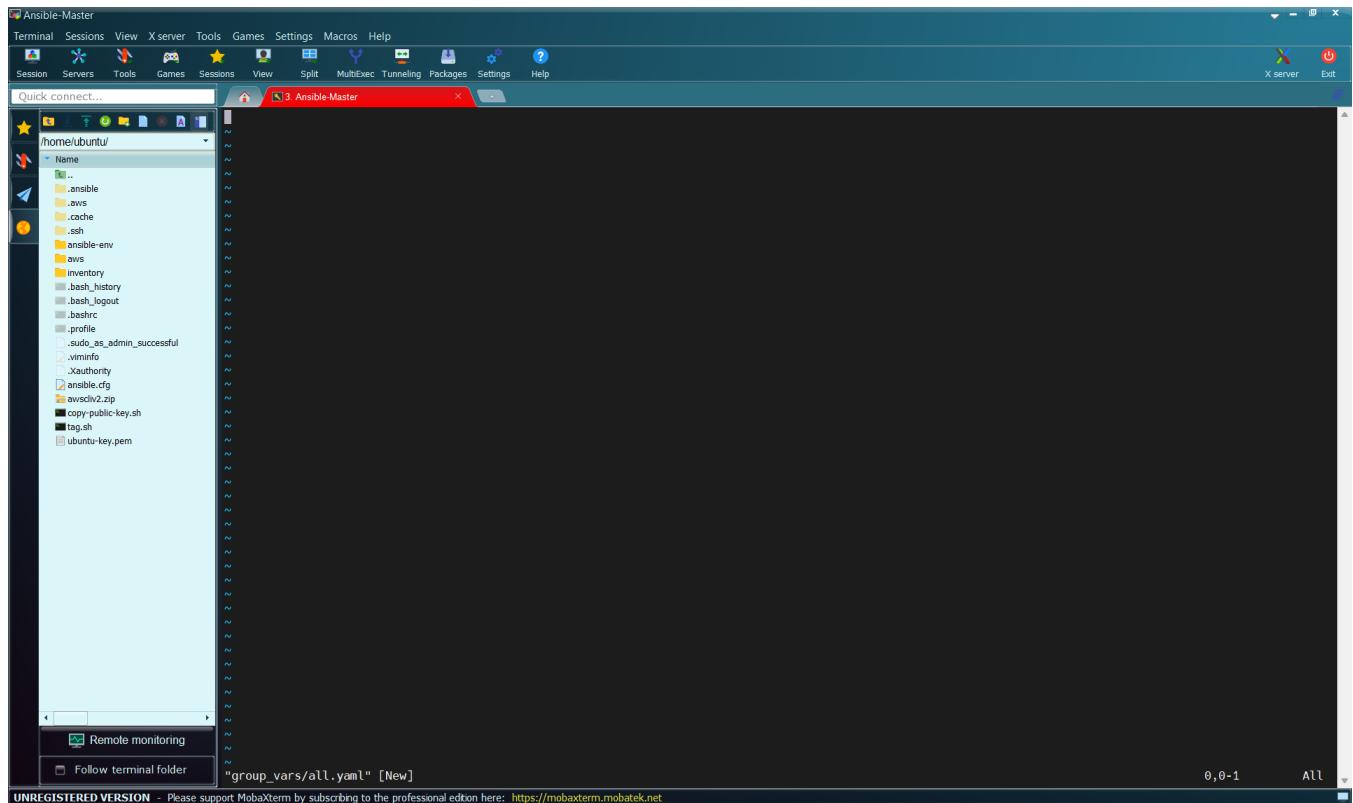
The screenshot shows a terminal window titled "Ansible-Master" running on a Linux system. The terminal session is as follows:

```
(ansible-env) ubuntu@ip-172-31-16-159:~$ mkdir vm-monitor
(ansible-env) ubuntu@ip-172-31-16-159:~$ cd vm-monitor
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ mkdir group_vars
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$
```

To the left of the terminal is a file explorer window showing the directory structure under "/home/ubuntu/". The "group_vars" folder has been created in the "vm-monitor" directory.

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

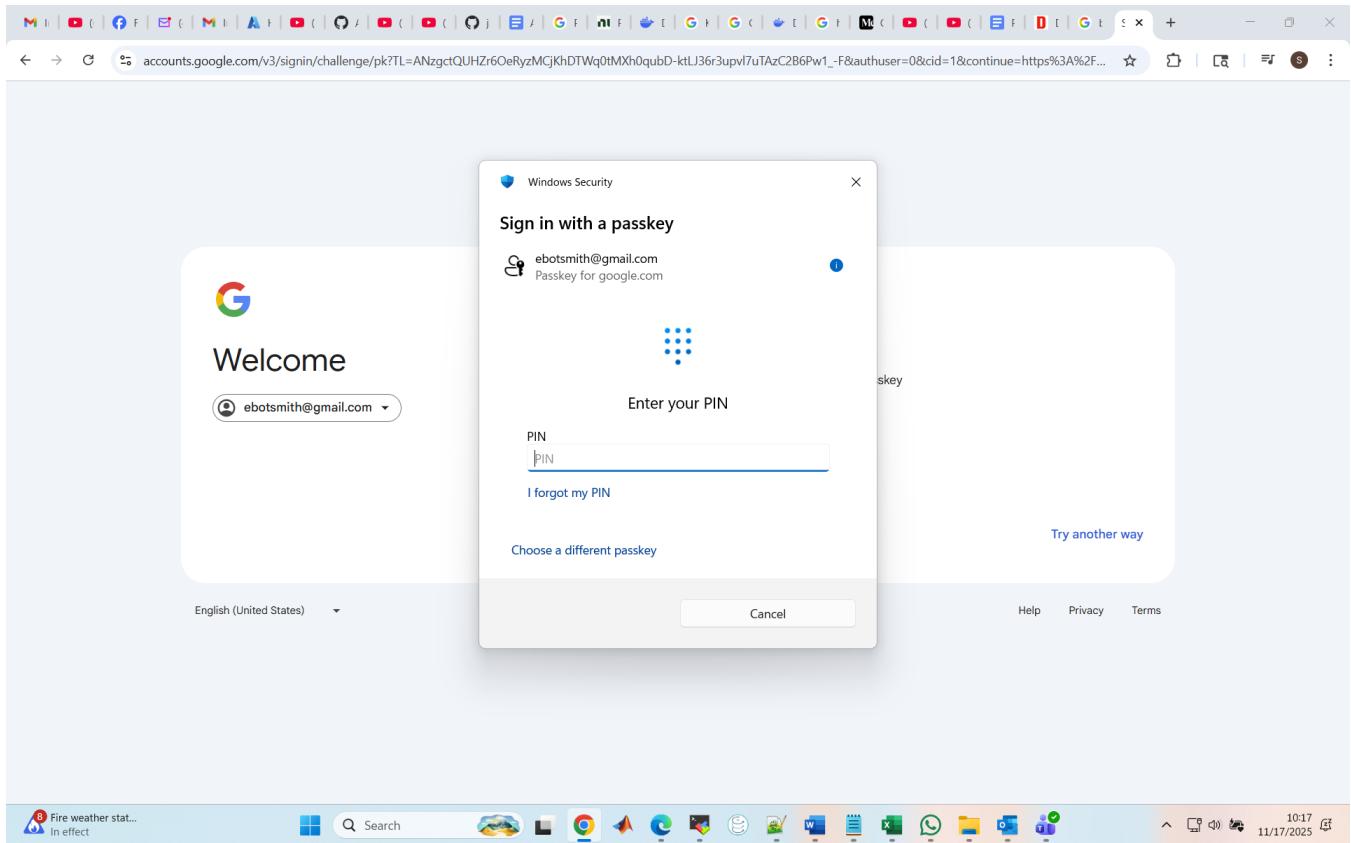
```
vi group_vars/all.yaml
```



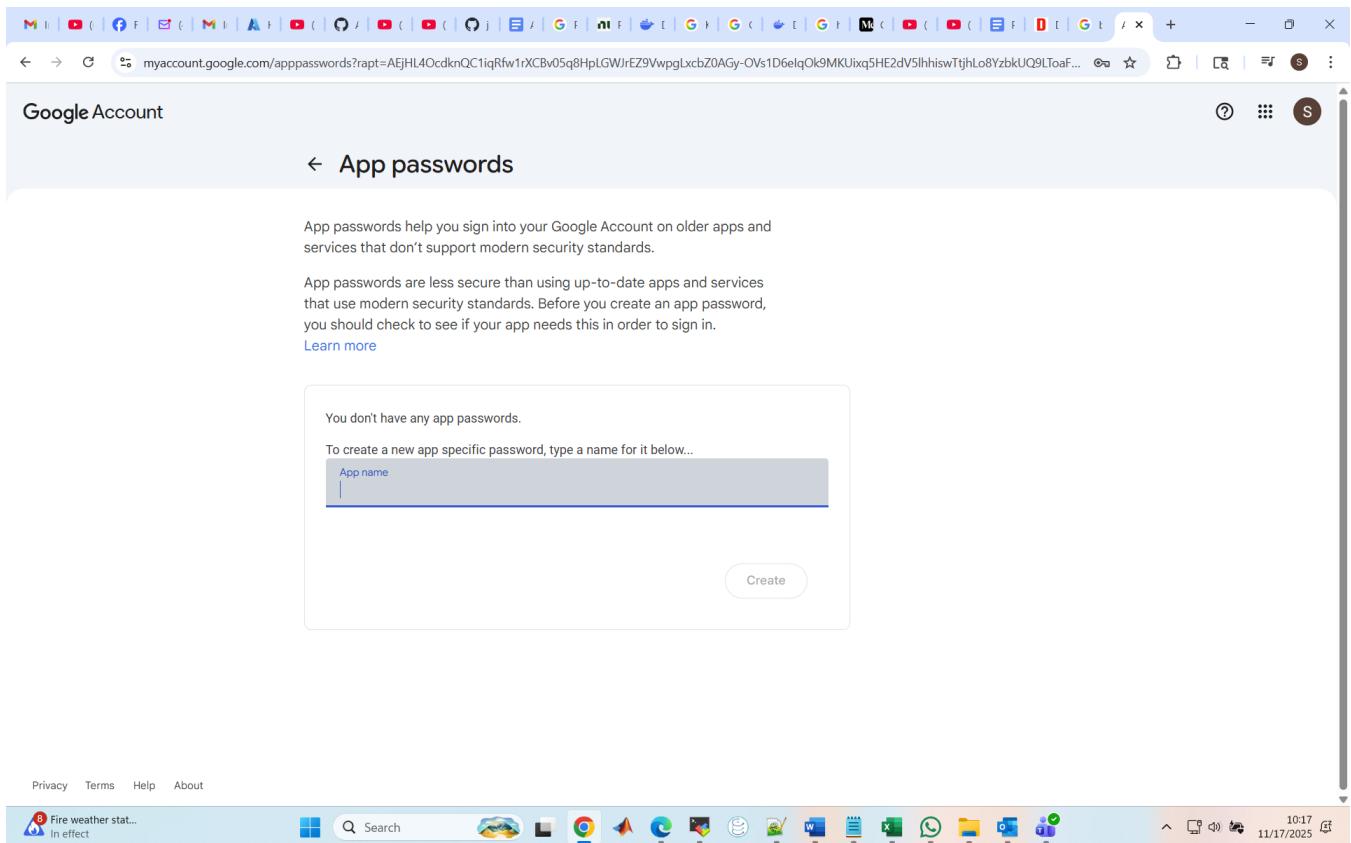
Then paste this code here

```
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**"

The screenshot shows a web browser window for Google Account's App passwords section. A modal dialog box is open, prompting the user to enter an app name. The input field contains "vm-monitor". Below the input field is a "Create" button. The background of the page displays information about app passwords and a note that the user doesn't have any app passwords.

Click on “Create”

The screenshot shows the same Google Account App passwords page after the "Create" button was clicked. A modal dialog box is now displayed, titled "Generated app password", containing the generated password "tqlj wtju jjgq fhay". Below the password, there is a "How to use it" section with instructions and a "Done" button. The background page shows the previous step where the app name "vm-monitor" was entered.

Copy the password and add on your code

The screenshot shows the 'App passwords' section of the Google Account settings. A modal window titled 'Generated app password' displays the password 'tqlj wtju jjgq fhay'. Below it, instructions explain how to use the password for an application named 'vm-monitor'. The main interface shows a sidebar with 'Your app password' and a list of existing app names: 'vm-monitor', 'To create a new app', and 'App name'. At the bottom right of the modal is a 'Done' button.

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

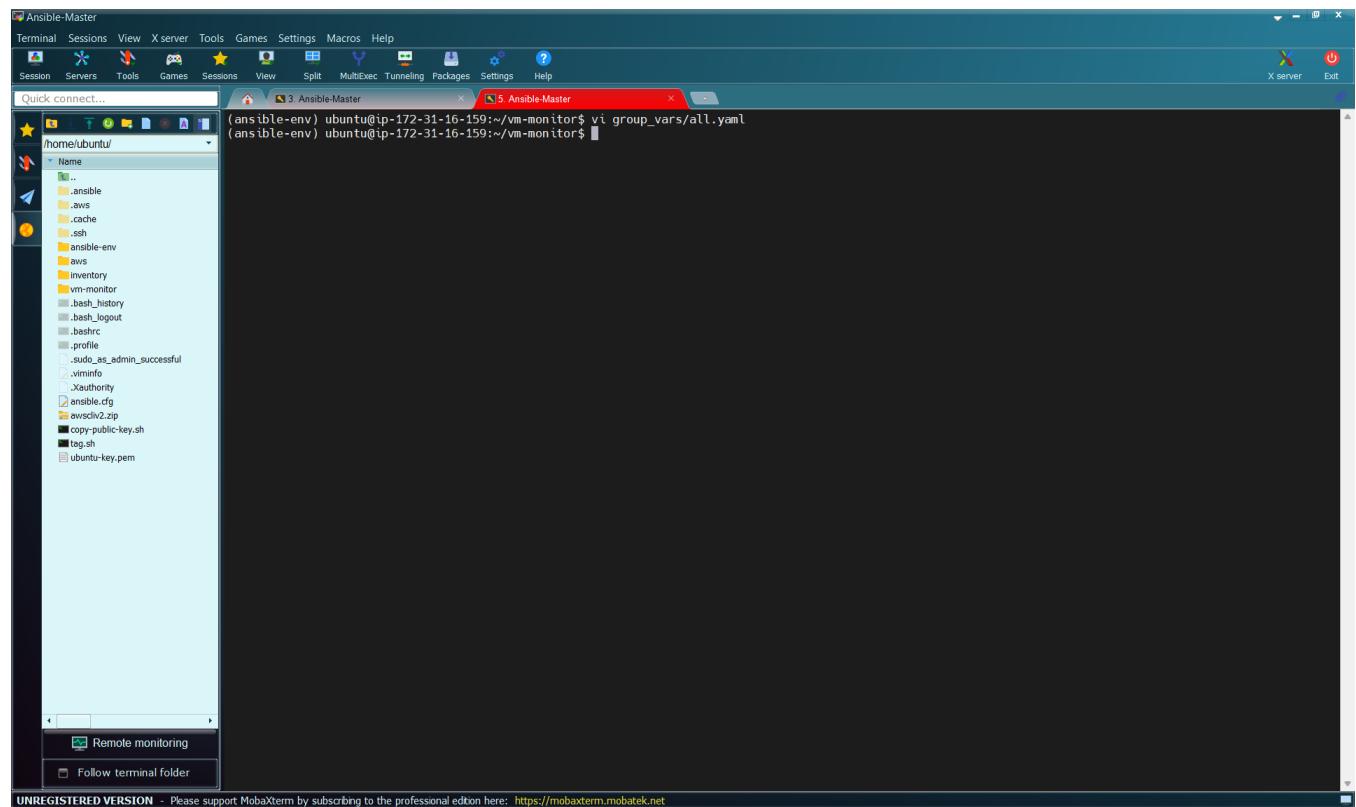
The screenshot shows a MobaXterm terminal window with two tabs. The left tab shows the file structure of a local directory, including files like 'ansible.cfg', 'awscli2.zip', 'copy-public-key.sh', 'tag.sh', and 'ubuntu-key.pem'. The right tab contains the configuration variables defined in the previous code block:

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

The terminal window has a dark theme and includes standard navigation and status bars at the top and bottom.

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

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/ .
```

The screenshot shows a terminal window titled "Ansible-Master" running on a Linux system. The terminal displays the command `cp -r /home/ubuntu/inventory/ .`. To the left of the terminal is a file explorer sidebar showing a folder structure under "Name". A context menu is open at the bottom left of the terminal window, with options "Remote monitoring" and "Follow terminal folder".

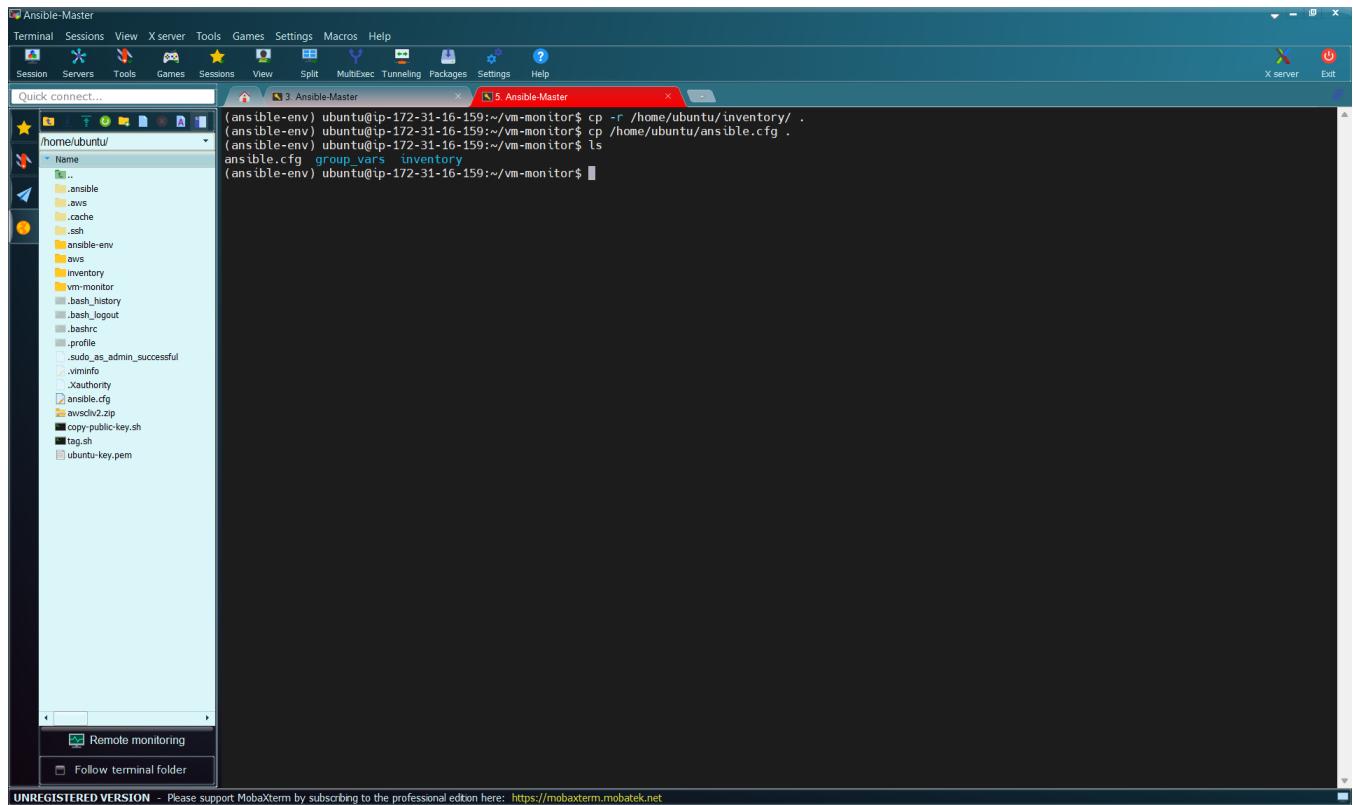
Then we will copy the file by using this command:

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

The screenshot shows a terminal window titled "Ansible-Master" running on a Linux system. The terminal displays the command `cp /home/ubuntu/ansible.cfg .`. To the left of the terminal is a file explorer sidebar showing a folder structure under "Name". A context menu is open at the bottom left of the terminal window, with options "Remote monitoring" and "Follow terminal folder".

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

ls



The screenshot shows a MobaXterm window titled "Ansible-Master". On the left, there is a file explorer sidebar showing the directory structure of "/home/ubuntu/". The main terminal window displays the following command-line session:

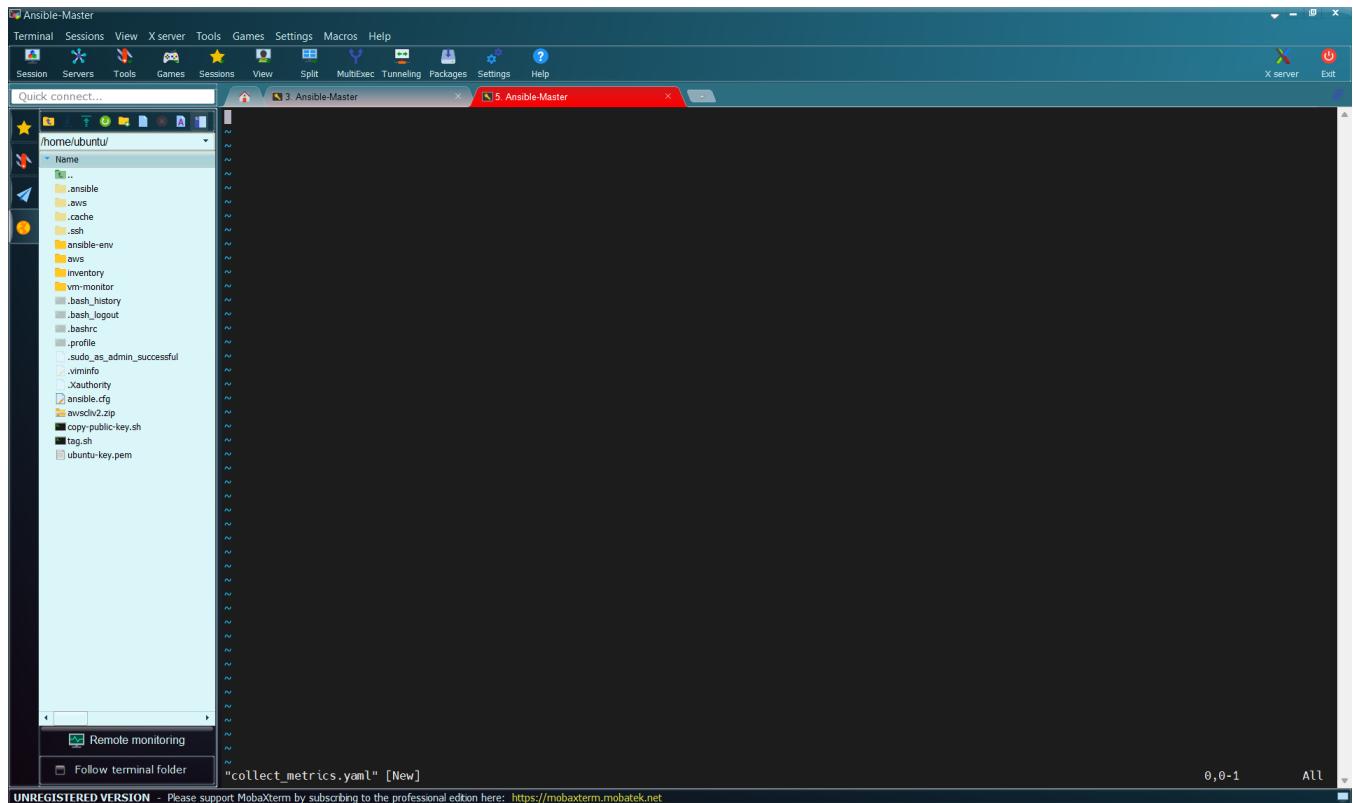
```
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ cp -r /home/ubuntu/inventory/ .
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ cp ./home/ubuntu/ansible.cfg .
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ ls
ansible.cfg  group_vars  inventory
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$
```

At the bottom of the terminal window, there is a watermark: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

You can see what we have added so far.

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

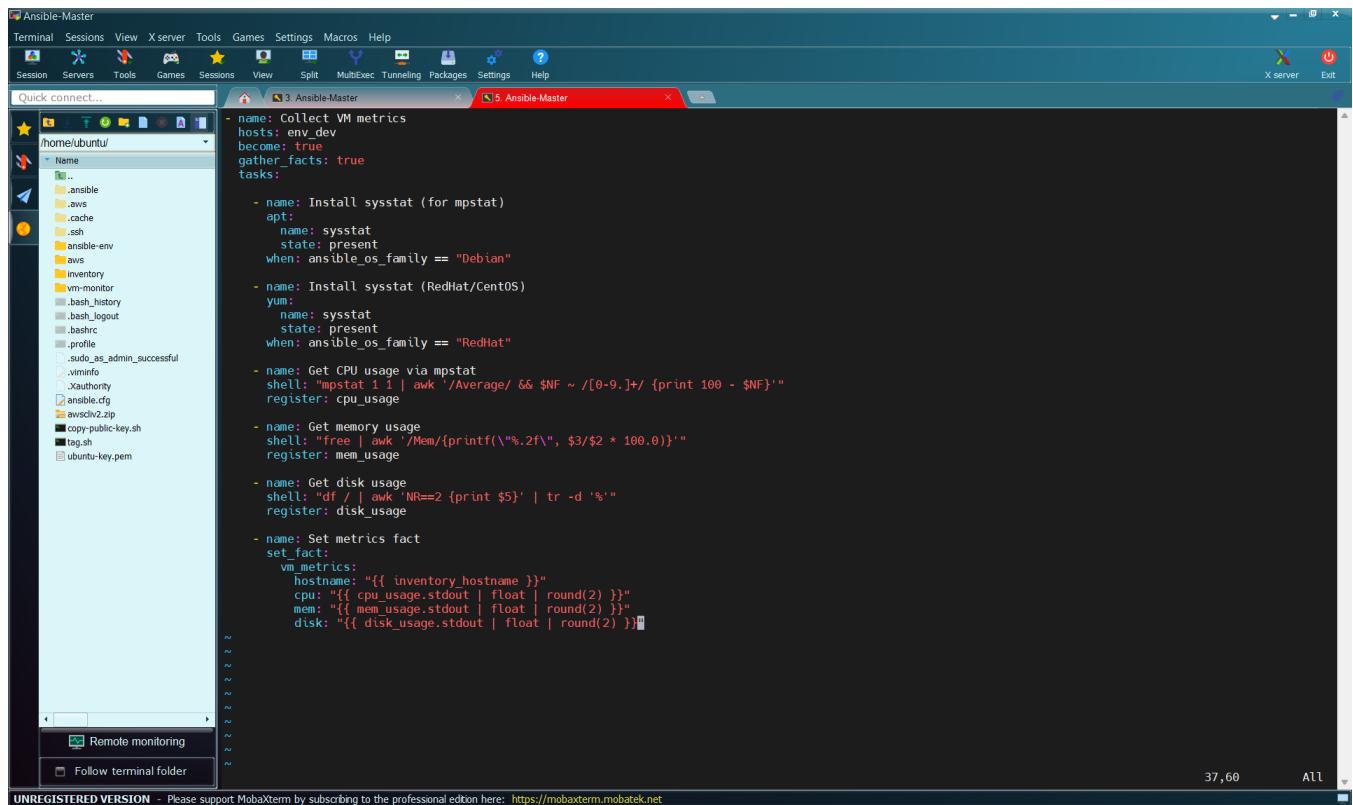
```
vi collect_metrics.yaml
```



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) }}"
```



The screenshot shows the MobaXterm interface with two terminal windows open. The left window displays the Ansible playbooks for collecting VM metrics. The right window shows the file structure of the Ansible project, which includes files like .aws, .cache, .ssh, ansible-env, aws, inventory, vm-monitor, bash_history, bash_logout, bashrc, profile, .sudo_as_admin_successful, .viminfo, .xauthority, ansible.cfg, awscli.v2.zip, copy-public-key.sh, tag.sh, and ubuntu-key.pem.

```
- 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”

```
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ cp -r /home/ubuntu/inventory/ .
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ cp /home/ubuntu/ansible.cfg .
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ ls
ansible.cfg  group_vars  inventory
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi collect_metrics.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

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:

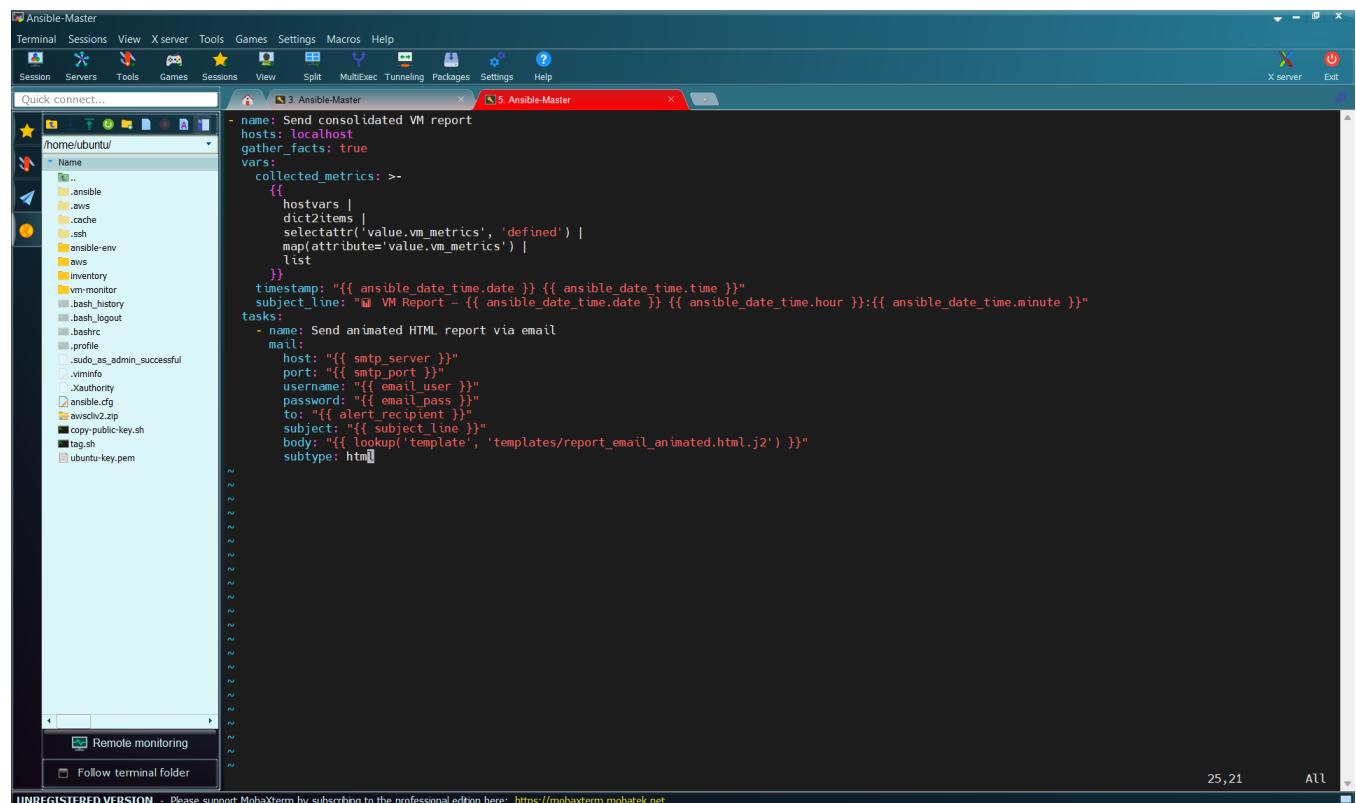
```
vi send_report.yaml
```

"send_report.yaml" [New]

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

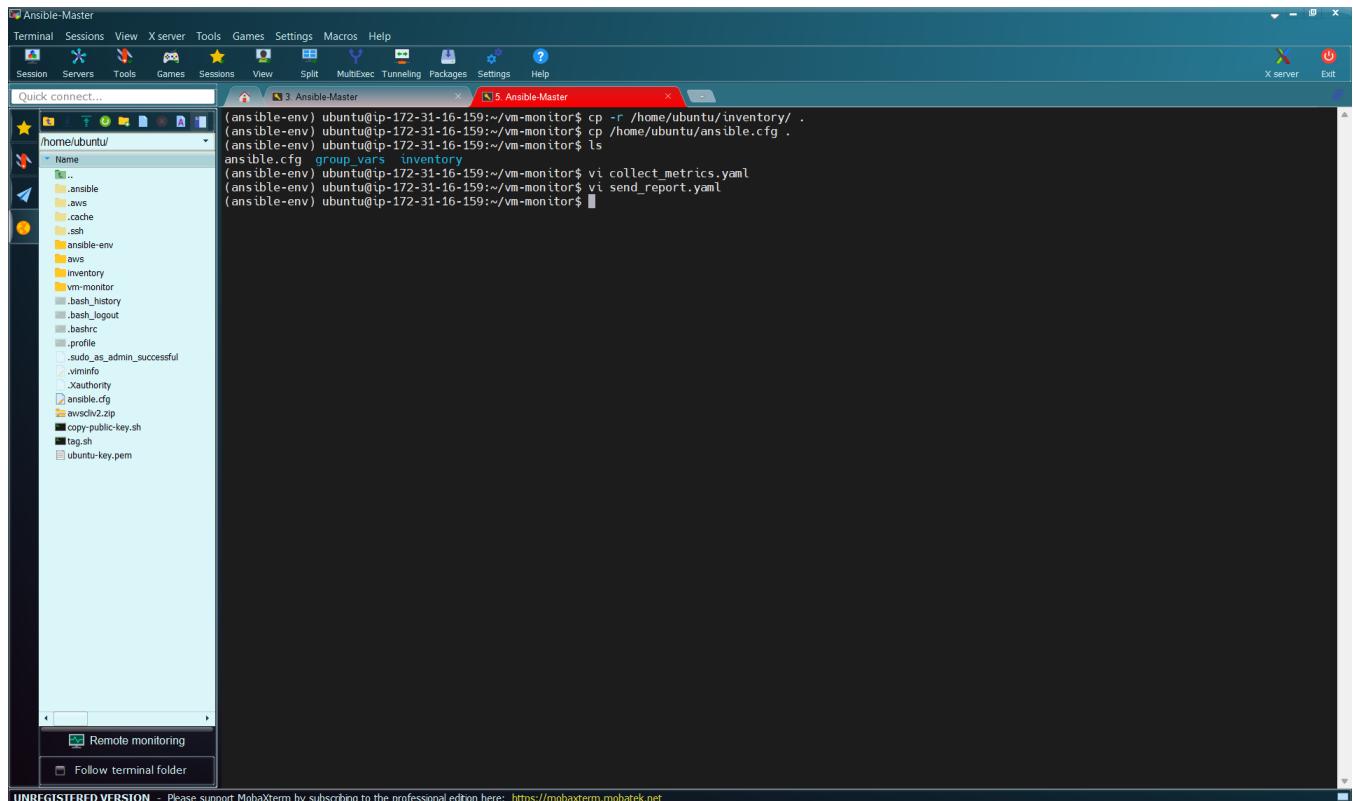
And we paste the code:

```
- 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
```



The screenshot shows the MobaXterm interface with two terminal windows open. The left window is titled 'Ansible-Master' and displays the Ansible playbook code. The right window is also titled 'Ansible-Master' and shows a blank terminal session. The MobaXterm toolbar at the top includes icons for Terminal, Sessions, View, X server, Tools, Games, Settings, Macros, Help, Session, Servers, Tools, Games, Sessions, View, Split, MultiExec, Tunneling, Packages, Settings, Help, X server, and Exit. The bottom status bar indicates 'UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>'.

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



The screenshot shows a MobaXterm window titled "Ansible-Master". The terminal session contains the following command history:

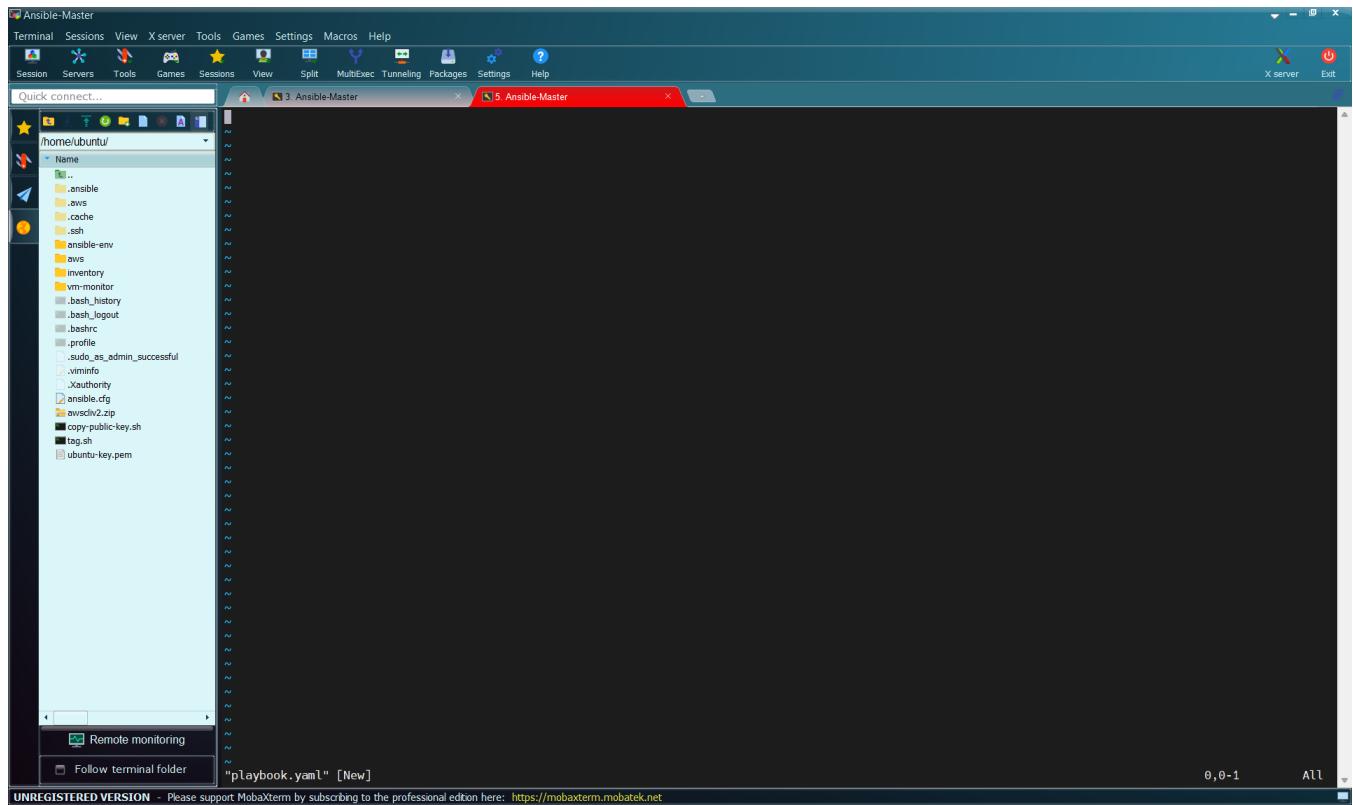
```
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ cp -r /home/ubuntu/inventory/ .
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ cp /home/ubuntu/ansible.cfg .
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ ls
ansible.cfg group_vars inventory
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi collect_metrics.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi send_report.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$
```

The left sidebar shows a file tree under "/home/ubuntu/". The terminal window has a status bar at the bottom with the message "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

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

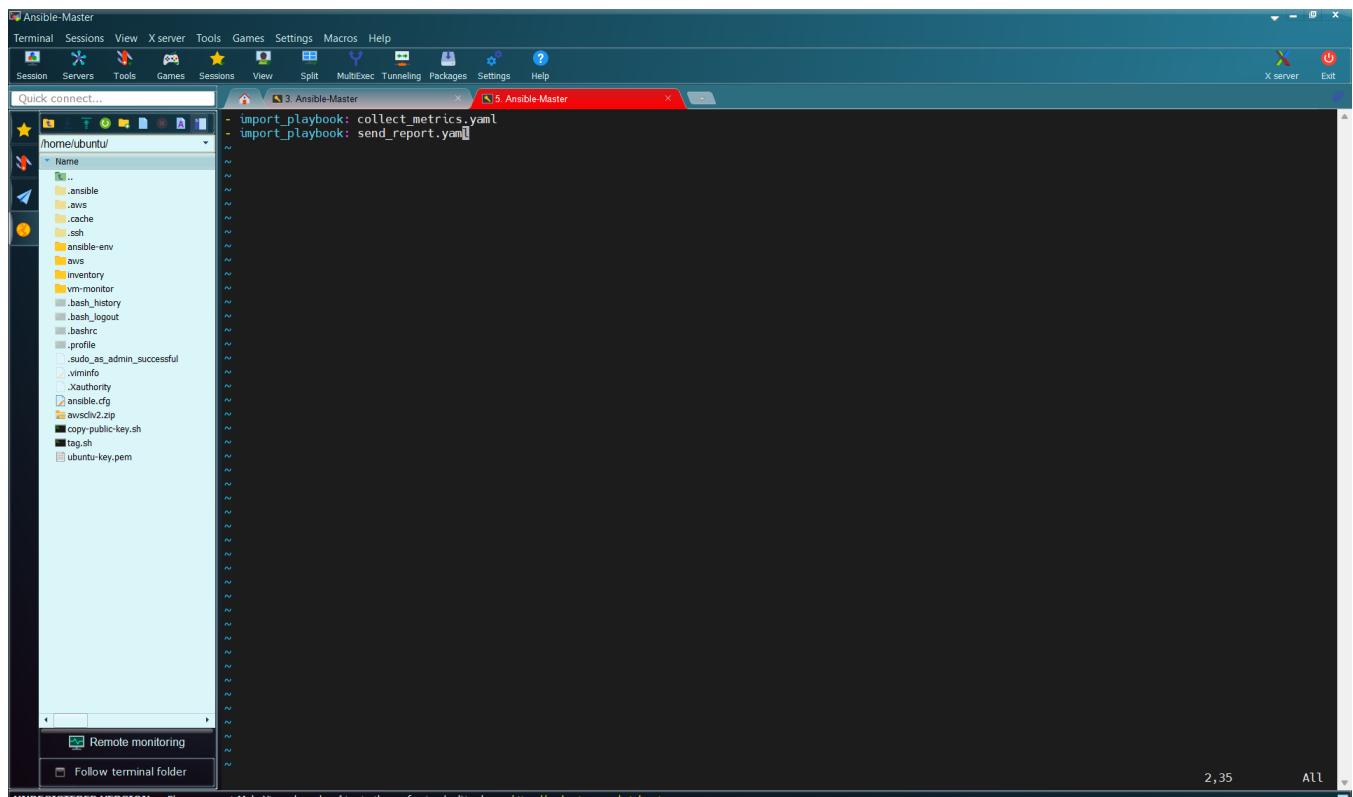
We will call the file **“playbook.yaml”**, that will be created by using the command:

```
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”

```
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ cp -r /home/ubuntu/inventory/ .
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ cp /home/ubuntu/ansible.cfg .
ansible.cfg  group_vars inventory
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi collect_metrics.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi send_report.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi playbook.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ ls
ansibile.cfg  group_vars  inventory
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ ls
ansibile.cfg  group_vars  inventory
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ cp /home/ubuntu/inventory/
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ cp /home/ubuntu/ansible.cfg .
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ ls
ansible.cfg  group_vars  inventory
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi collect_metrics.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi send_report.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi playbook.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ mkdir templates
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

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:

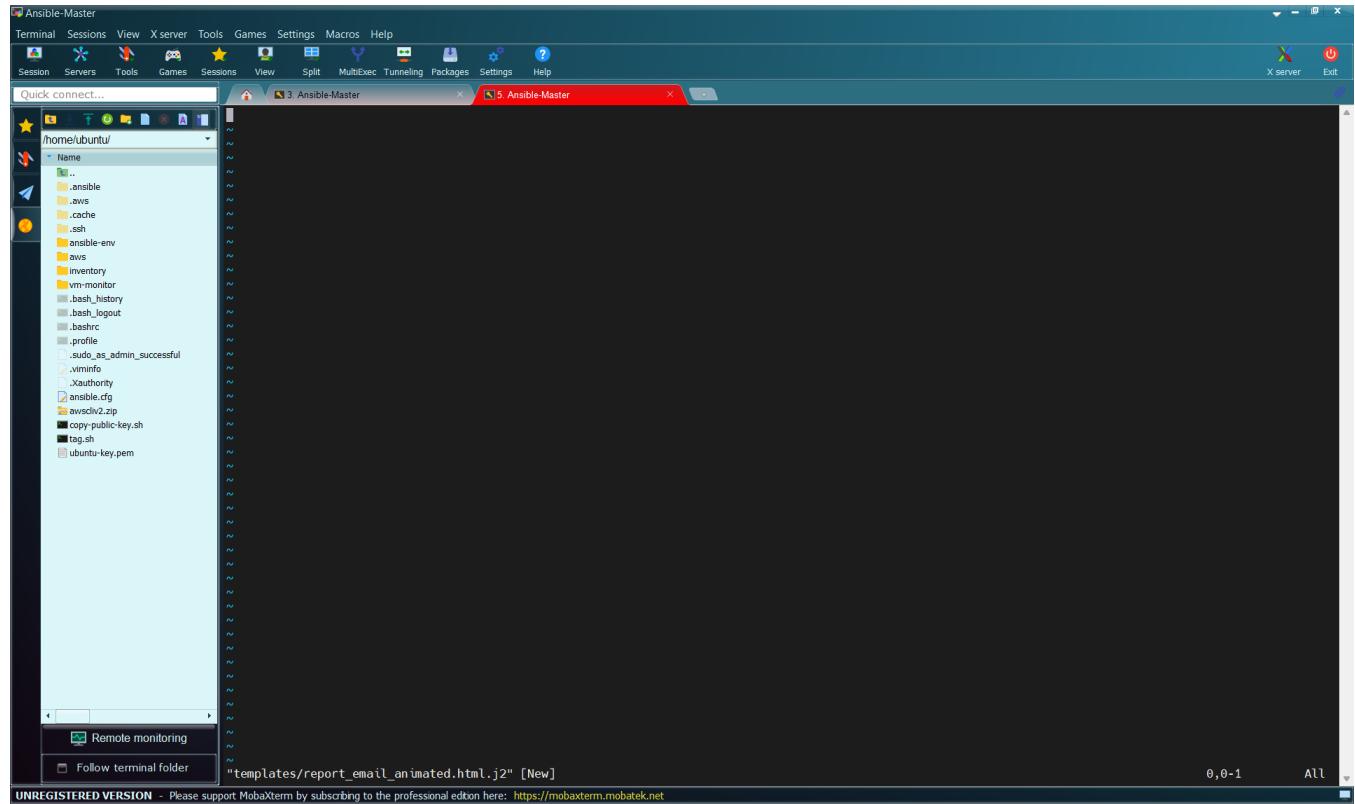
`mkdir templates`

```
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ cp -r /home/ubuntu/inventory/ .
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ cp /home/ubuntu/ansible.cfg .
ansible.cfg  group_vars inventory
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi collect_metrics.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi send_report.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi playbook.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ ls
ansibile.cfg  group_vars  inventory
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ ls
ansibile.cfg  group_vars  inventory
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ cp /home/ubuntu/inventory/
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ cp /home/ubuntu/ansible.cfg .
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ ls
ansible.cfg  group_vars  inventory
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi collect_metrics.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi send_report.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi playbook.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ mkdir templates
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

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;
        }
    </style>
</head>
<body>
    <h2>Report Email</h2>
    <div class="summary">
        <p>This is a summary of the report email. It contains key information about the system configuration and status.</p>
    </div>
</body>
</html>
```

```
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>

📊 Consolidated VM Health Report

<div class="summary">

💻 {{ collected_metrics | length }} VMs | 🔥 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>

🌐 Hostname </th>

🔥 CPU Usage </th>

💡 Memory Usage </th>

📁 Disk Usage </th>

</tr>

{% for vm in collected_metrics %}

<tr>

<td class="hostname">

{{ vm.hostname }}

</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>

```

The screenshot shows a MobaXterm session titled "Ansible-Master". The left pane shows a file tree with various Ansible-related files like `ansible.cfg`, `awsenv2.zip`, and `ubuntu-key.pem`. The right pane displays the generated HTML report. The report includes a header with the title "Ansible-Master" and a footer indicating the report was generated on a specific timestamp. The main content is a table showing system metrics for multiple VMs. The table has columns for Hostname, CPU Usage, Memory Usage, and Disk Usage. Each row contains a link to the VM's hostname. The CPU and Memory columns use bar charts to show usage levels (green for Ample, yellow for Monitor, red for Full). The Disk column uses a similar bar chart. The report also includes a legend for the badge colors: healthy (green), low (yellow), warning (orange), moderate (light blue), and critical (red).

```

<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="label">Healthy</div>
        {% elif vm.cpu|float < 80 %}<div class="label">Low</div>
        {% else %}<div class="label">Warning</div>
        {% elif vm.cpu|float < 80 %}<div class="label">Moderate</div>
        {% else %}<div class="label">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”

```
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ cp -r /home/ubuntu/inventory/ .
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ cp /home/ubuntu/ansible.cfg .
(ansible.cfg group_vars inventory)
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi collect_metrics.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi send_report.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi playbook.yaml
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ mkdir templates
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ vi templates/report_email_animated.html.j2
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$
```

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
```

```
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ sudo apt install tree
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  tree
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 47.4 kB of archives.
After this operation, 111 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 tree amd64 2.1.1-2ubuntu3.24.04.2 [47.4 kB]
Fetched 47.4 kB in 0s (2261 kB/s)
Selecting previously unselected package tree.
(Reading database
 129011 files and directories currently installed.)
Preparing to unpack .../tree_2.1.1-2ubuntu3.24.04.2_amd64.deb ...
Unpacking tree (2.1.1-2ubuntu3.24.04.2)...
Setting up tree (2.1.1-2ubuntu3.24.04.2)...
Processing triggers for man-db (2.12.0-4build2)...
Scanning processes...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
 6.14.0-1015-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.14.0-1016-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$
```

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

The screenshot shows the MobaXterm interface with two terminal windows. The left terminal window shows the directory structure of `ansible-env` on an Ubuntu system:

```
(ansible-env) ubuntu@ip-172-31-16-159:~/vm-monitor$ tree
.
├── ansible.cfg
├── collect_metrics.yaml
├── group_vars
│   └── all.yaml
├── inventory
│   └── aws_ec2.yaml
├── playbook.yaml
└── send_report.yaml

4 directories, 7 files
```

The right terminal window is titled "Ansible-Master" and shows the command `tree` being run again, resulting in the same output.

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

```
ansible-playbook playbook.yaml
```

The screenshot shows the MobaXterm interface with two terminal windows. The left terminal window shows the directory structure of `ansible-env`. The right terminal window shows the execution of the playbook:

```
changed: [ec2-54-90-78-15.compute-1.amazonaws.com]

TASK [Get disk usage] *****
changed: [ec2-18-205-18-36.compute-1.amazonaws.com]
changed: [ec2-3-82-51-215.compute-1.amazonaws.com]
changed: [ec2-98-88-76-93.compute-1.amazonaws.com]
changed: [ec2-3-88-175-222.compute-1.amazonaws.com]
changed: [ec2-107-20-131-183.compute-1.amazonaws.com]
changed: [ec2-98-93-190-202.compute-1.amazonaws.com]
changed: [ec2-3-87-88-192.compute-1.amazonaws.com]
changed: [ec2-13-221-88-138.compute-1.amazonaws.com]
changed: [ec2-54-90-78-15.compute-1.amazonaws.com]
changed: [ec2-54-235-10-82.compute-1.amazonaws.com]

TASK [Set metrics fact] *****
ok: [ec2-3-82-51-215.compute-1.amazonaws.com]
ok: [ec2-3-88-175-222.compute-1.amazonaws.com]
ok: [ec2-107-20-131-183.compute-1.amazonaws.com]
ok: [ec2-18-205-18-36.compute-1.amazonaws.com]
ok: [ec2-98-88-76-95.compute-1.amazonaws.com]
ok: [ec2-98-93-190-202.compute-1.amazonaws.com]
ok: [ec2-54-235-10-82.compute-1.amazonaws.com]
ok: [ec2-3-87-88-192.compute-1.amazonaws.com]
ok: [ec2-13-221-88-138.compute-1.amazonaws.com]
ok: [ec2-54-90-78-15.compute-1.amazonaws.com]

PLAY [Send consolidated VM report] *****
TASK [Gathering Facts] *****
ok: [localhost]

TASK [Send animated HTML report via email] *****
ok: [localhost]

PLAY RECAP *****
ec2-107-20-131-183.compute-1.amazonaws.com : ok=6    changed=3    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
ec2-18-205-18-36.compute-1.amazonaws.com : ok=6    changed=3    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
ec2-3-82-51-215.compute-1.amazonaws.com : ok=6    changed=3    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
ec2-3-87-88-192.compute-1.amazonaws.com : ok=6    changed=3    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
ec2-3-88-175-222.compute-1.amazonaws.com : ok=6    changed=3    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
ec2-13-221-88-138.compute-1.amazonaws.com : ok=6    changed=3    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
ec2-54-90-78-15.compute-1.amazonaws.com : ok=6    changed=3    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
ec2-98-88-76-95.compute-1.amazonaws.com : ok=6    changed=3    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
ec2-98-93-190-202.compute-1.amazonaws.com : ok=6    changed=3    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
localhost : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

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

A screenshot of a Gmail inbox. The top navigation bar shows 'mail.google.com/mail/u/2/#inbox'. The inbox list includes an email from 'ebotsmith' with the subject 'VM Report - 2025-11-17 16:30'. Other visible emails are 'Promotions 1 new Mezmo — Your AI SRE is Live...' and 'Social 1 new Rohit Patel via LinkedIn — Rohit ...'. The bottom status bar shows '11:30 AM'.

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

Open the email

A screenshot of an email message titled 'VM Report - 2025-11-17 16:30' from 'ebotsmith@gmail.com' to the user. The email subject is 'Consolidated VM Health Report'. The message body contains a summary bar with the text '10 VMs | Avg CPU: 15.04% | Avg Mem: 18.2% | Avg Disk: 16.1%'. Below this is a detailed table for three hosts:

Hostname	CPU Usage	Memory Usage	Disk Usage
ec2-3-82-51-215.compute-1.amazonaws.com	3.85% (Healthy, Low)	18.21% (Low)	16.0% (Ample)
ec2-3-88-175-222.compute-1.amazonaws.com	1.96% (Healthy, Low)	17.8% (Low)	16.0% (Ample)
ec2-107-20-131-183.compute-1.amazonaws.com	34.21% (Low)	18.31% (Low)	17.0% (Ample)

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