Mastering Ansible for DevOps Engineering

A Free Guide

English Book

Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

Table of Contents

History of Automation

Use Cases of Ansible

Introduction to YAML Format

Introduction to API Capability

Introduction to Ansible Connection

Introduction to Ansible Architecture

How to Set Up Ansible and Infrastructure

How to Install Ansible on the Control Machine

Introduction to Creating an Inventory

How to Group Targets in the Inventory

How to Use Variables in the Inventory File

Introduction to Ad Hoc Command

Introduction to Playbook and Module

How to Create a Playbook

How to Find and Use Modules

Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

History of Automation

Before DevOps became popular and adopted by companies, automation was already in use. This automation was performed on Windows operating systems through Batch Scripting and on Linux operating systems through Bash Scripting.

After OS automation, there was a need to automate databases, virtualization, and cloud environments as well. This required a different kind of automation that was not limited to operating systems. At this point, programming languages like Python, Ruby, Perl, and others entered the market. Among these languages, Python gained more popularity due to its vast library support and is now widely used in the field of automation.

In addition to these programming languages, Microsoft introduced PowerShell, which is used for automating Windows and other network-related tasks such as virtualization and more.

Then, Puppet was introduced as a Configuration Management Tool. Puppet is not actually a tool for automation, but rather for managing configurations. For example, when you have a large number of servers or a cluster of servers, you can use Puppet to centrally configure them. In other words, an agent is installed on each server that continuously communicates with Puppet, and Puppet executes its instructions in the form of configurations through its agents on the servers. However, when DevOps was introduced, Puppet started being used as a tool for automation as well.

In addition to Puppet, other tools like Salt Stack were introduced, which is a simple tool for executing commands on remote machines. Combining Puppet and Salt provides us with many capabilities. We can run the Puppet agent through Salt Stack, requiring an initial setup via Salt Stack followed by configuration through Puppet.

After tools like Puppet and Salt Stack, Chef was introduced. Puppet is generally a configuration tool and not a programming language.

Puppet is built using the Ruby language, and Salt Stack is built using Python, but there is no need to write Python or Ruby code, as they each have their own specific language.

Puppet uses a Domain Specific Language (DSL).

Chef was introduced with greater manageability and includes many templates and features. It has a graphical user interface and a good reporting system. In addition to these capabilities, Chef can operate on various parts of the network, such as servers and clients.

Then, Ansible was introduced as a Configuration Management Tool and Scripting Language, created by Michael DeHaan.

Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

Ansible is written in Python and is one of the simplest automation languages. It was acquired by Red Hat, and now there are many features available in Ansible.

Ansible was introduced with the goal of simplification and can manage a large infrastructure with simple code. Initially, Ansible focused on the Linux operating system, and later it was extended to support Windows automation, cloud automation, networking automation, and database automation.

Then, another tool called Terraform was introduced by HashiCorp, which is used for cloud automation.

Use Cases of Ansible

Ansible is used in the following areas:

-Any System Automation

-Change Management

-Provisioning

-Orchestration

Ansible is a simple and clean tool compared to other automation tools and does not require any agents like Chef, Puppet, or Salt Stack. Ansible uses SSH for managing Linux operating systems and winRM or API modules for Windows operating systems. Therefore, based on the type of target, it uses a specific communication module.

Ansible does not use any database. It uses YAML or text format, and its output is in JSON format.

Setting up Ansible is simple, as it comes in the form of a Python library, and you can install it using the following command.

```
pip install ansible
```

You can create Ansible code using a Playbook. A Playbook can execute a series of Python scripts on the target and then provide you with the output.

Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

Introduction to YAML Format

You don't need to write code in a programming language for automation; instead, you can use YAML. Ansible scripts or Ansible playbooks are written in the form of a YAML file, which is easy to read and write.

The YAML format has less structure and complexity compared to other programming language formats, which is why Ansible is very easy to read and write.

Introduction to API Capability

Ansible has many modules for performing specific tasks. These modules are API-based, meaning that if you want to create an AWS EC2 instance using Ansible, it can use an API-based tool to communicate with it.

You can use a RESTful URL that can be called by Ansible.

You can also execute Shell commands or PowerShell commands through Ansible.

Introduction to Ansible Connection

Ansible can connect to targets using various methods. A target can be Windows, Linux, a cloud account, a database, or any network-connected device.

Ansible uses SSH to connect to any Linux system, so SSH must be running on the Linux operating systems.

To connect to Windows, Ansible uses winRM, which is a service available on Windows. You need to enable remote connection on Windows. In Windows PowerShell, you can enable remote connection so that you can connect to Windows through winRM.

To connect to clouds like AWS, Azure, Google, or any other type of cloud, you can use APIs.

Ansible acts as a control machine. In other words, Ansible is not a server because no service runs on it. However, tools like Puppet, Salt, or Chef are considered servers because each of them runs as a service on a server.

Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

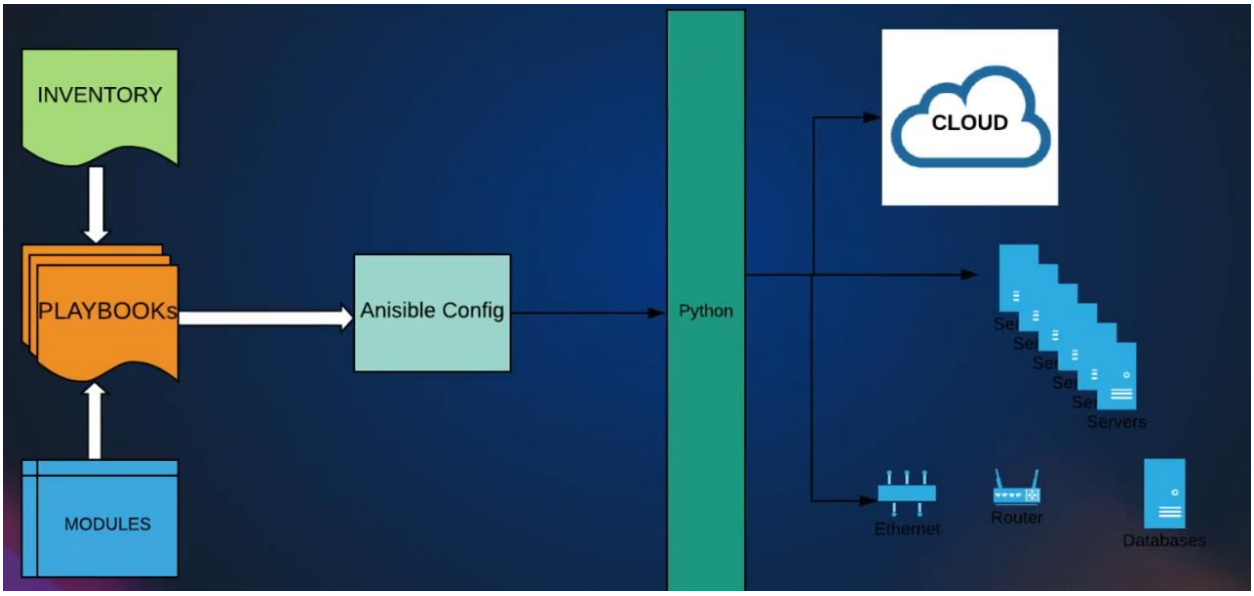
Ansible uses Python modules and scripts to connect to the target machine.

For connections made through SSH or winRM, Ansible sends Python scripts to the target, executes them, and returns the output.

For API tasks, the commands are executed on the control machine.

Introduction to Ansible Architecture

In this architecture, you need to create an inventory file. An inventory file contains information about the target machines, such as IP address, username, and password.



Ansible has many built-in modules, each used for specific tasks, such as installing a package, restarting a service, or taking a snapshot of an AWS EBS volume.

You can write a playbook. A playbook contains a task that uses module and inventory information to execute the task.

You can use Ansible configuration to create a Python script. When you execute it, Ansible runs a Python package on your destination.

Instructor: Fariborz Fallahzadeh

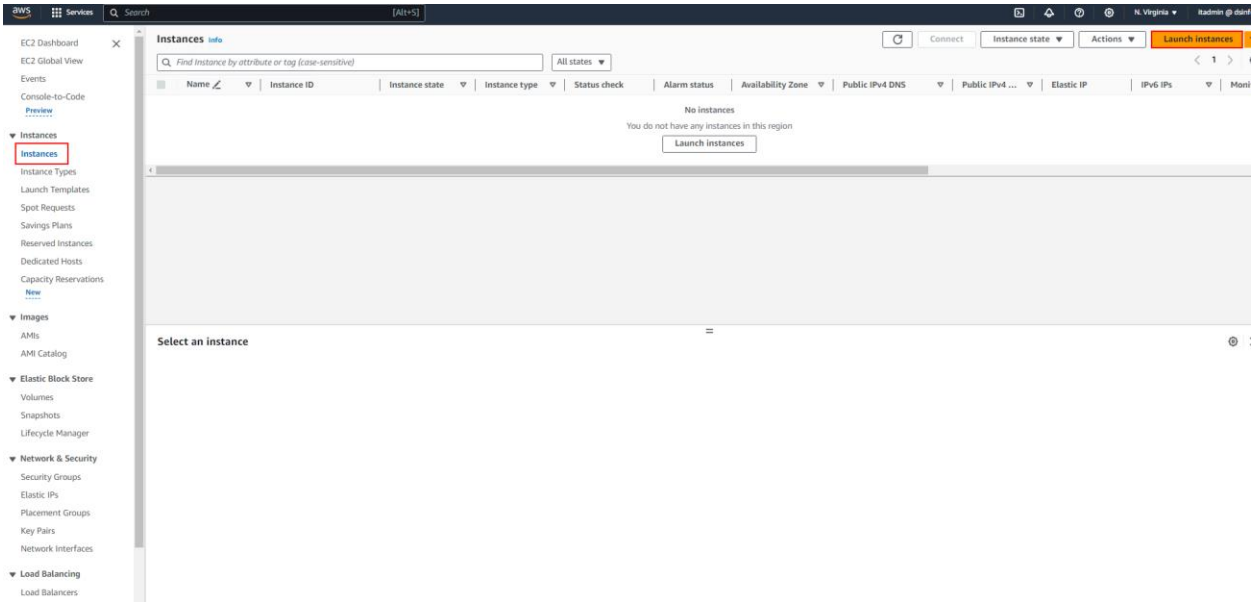
Email Address: fariborz.fallahzadeh@gmail.com

How to Set Up Ansible and Infrastructure

In this section, we are going to set up an infrastructure for Ansible. We will use AWS Cloud infrastructure and create an Ubuntu EC2 instance to act as the control machine, on which Ansible will be installed. Additionally, we will create three EC2 instances with CentOS as the operating system—two of them will be used as web servers and the other one for the database. The setup of these three EC2 instances will be done using Ansible.

How to Create an Ansible Control EC2 Instance

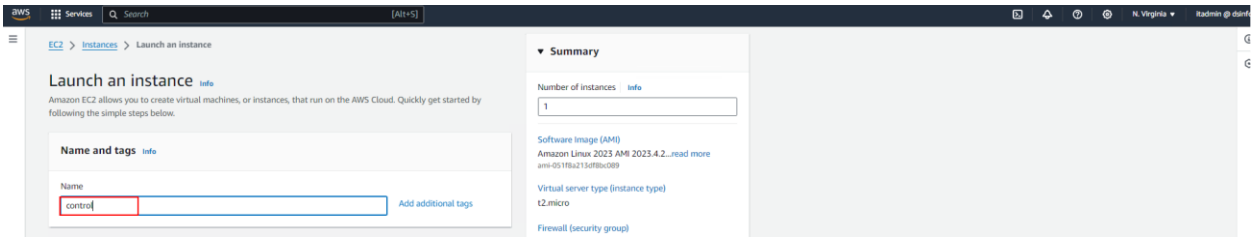
First, to create this instance, go to the Instances section and click on the **Launch Instance** button.



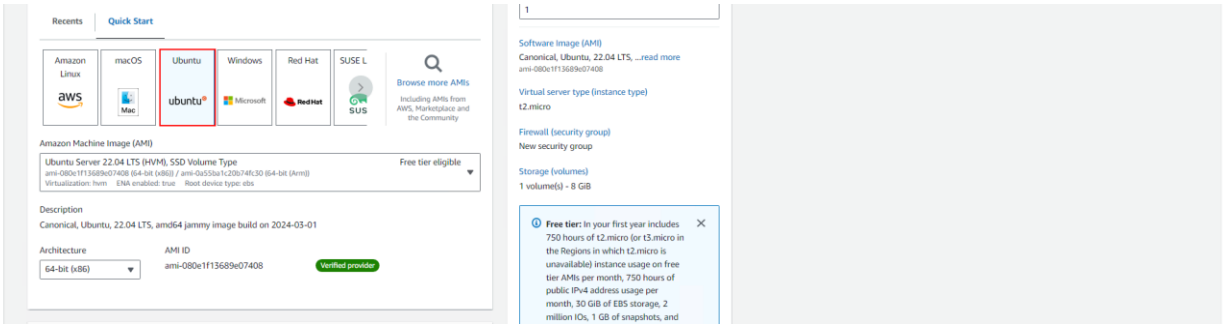
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

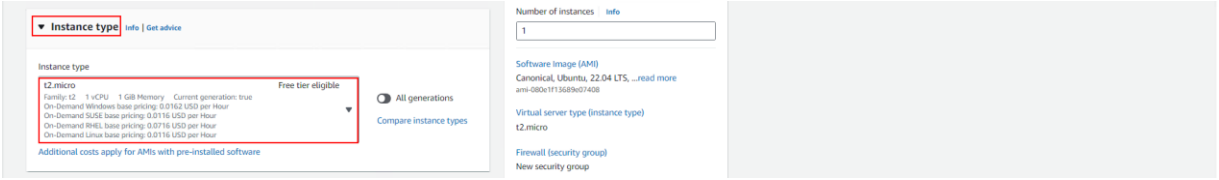
Specify a name for the instance.



Set the AMI type to Ubuntu.



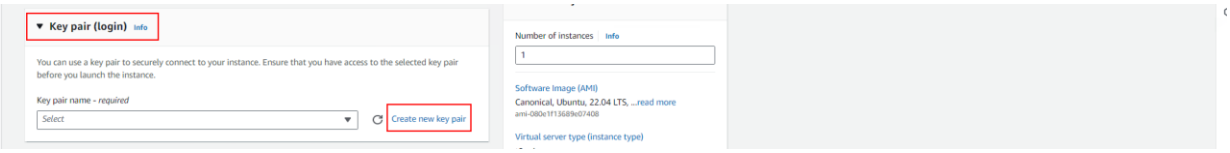
In the Instance Type section, set it to **t2.micro**.



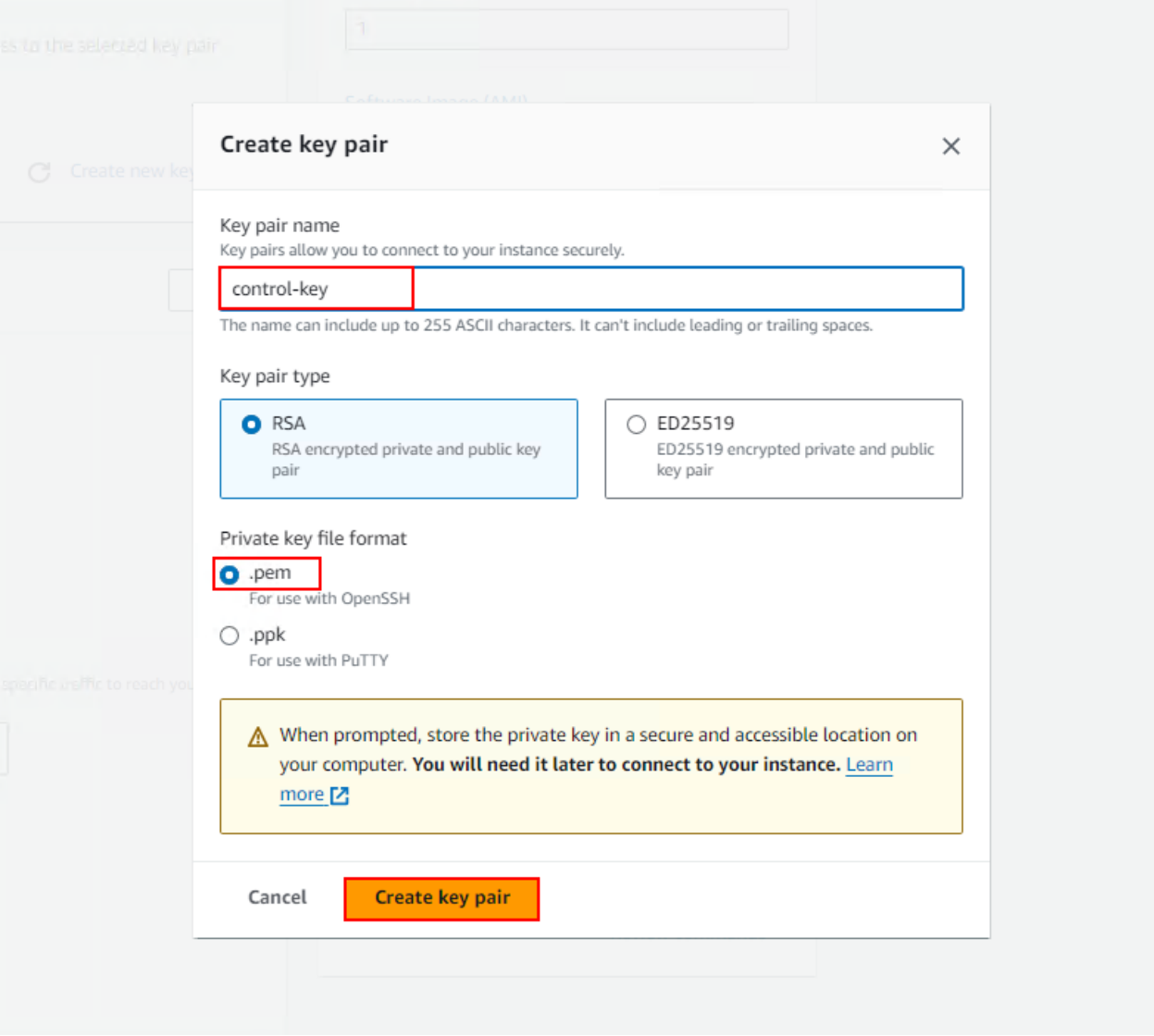
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

In the Key Pair section, click on the **Create New Key Pair** link.



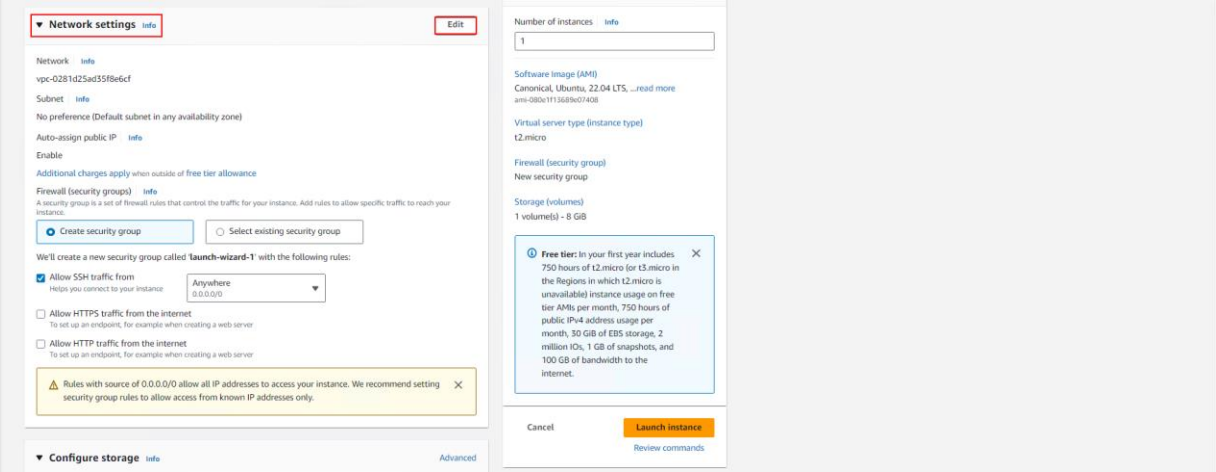
At this stage, specify a name for the key pair, set the private key format to **PEM**, and then click on the **Create Key Pair** button.



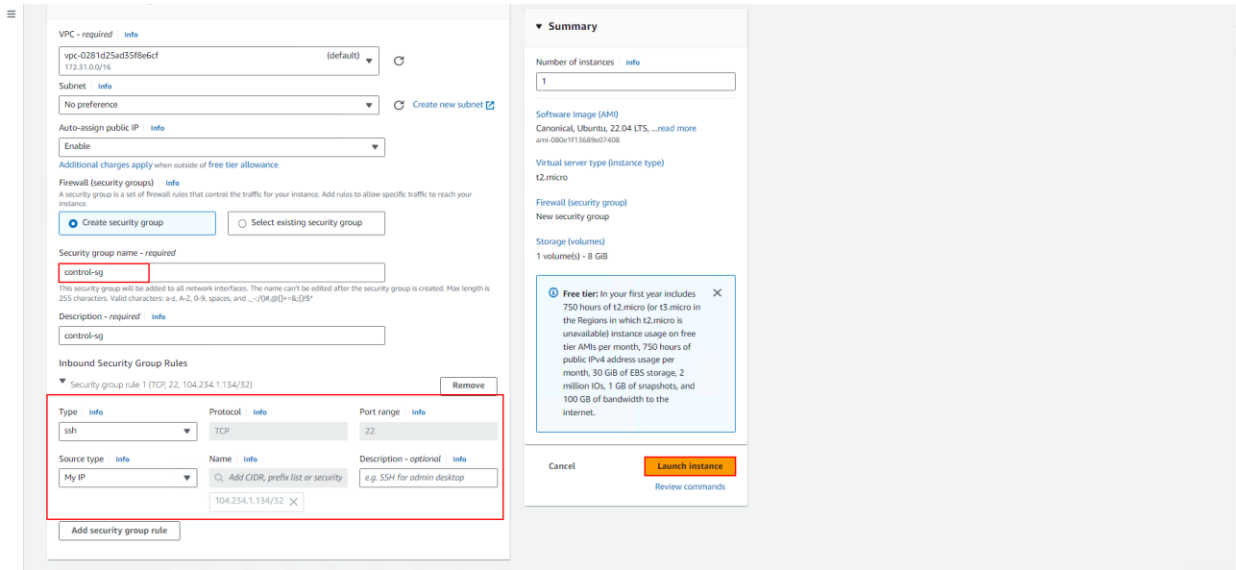
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

In the Network Settings section, click on the **Edit** button.



Then, specify a name for the security group and create an inbound rule to allow access to this instance via port 22. Finally, click on the **Launch Instance** button to create the instance.



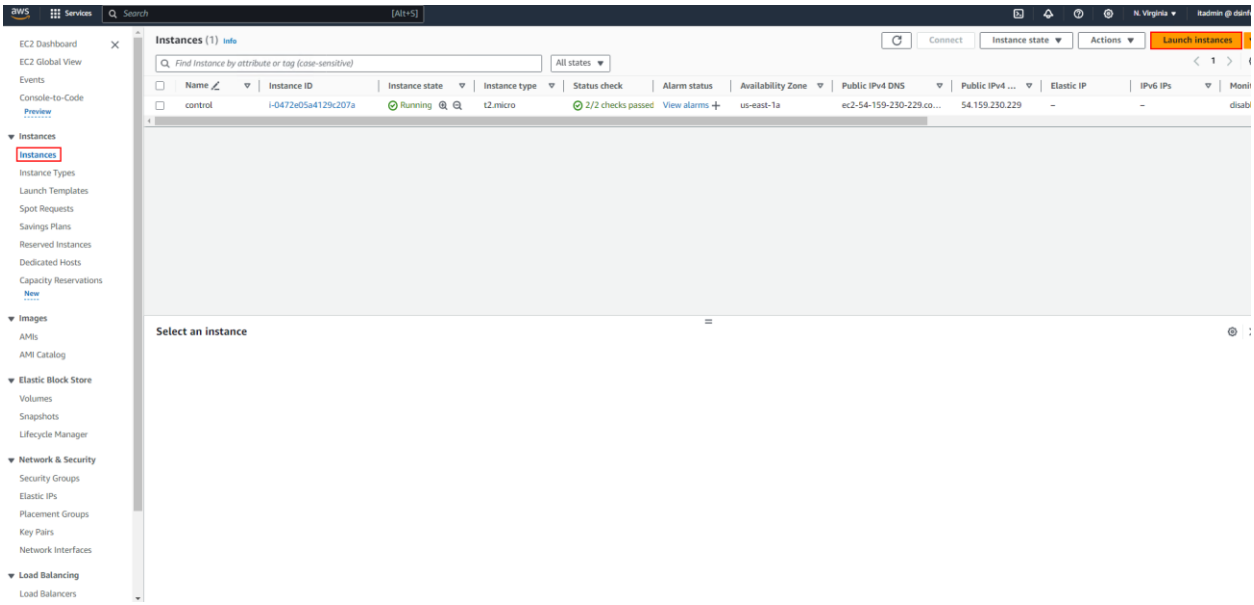
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

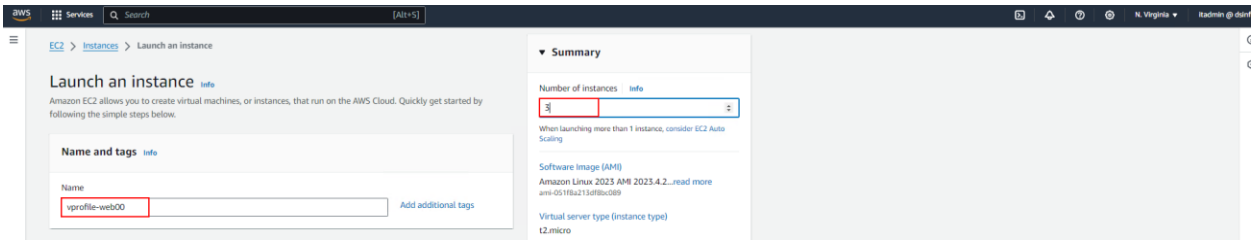
How to Create 3 CentOS EC2 Instances

At this stage, you need to create 3 EC2 instances with CentOS as the operating system.

At this stage, go to the Instances section and click on the **Launch Instance** button.



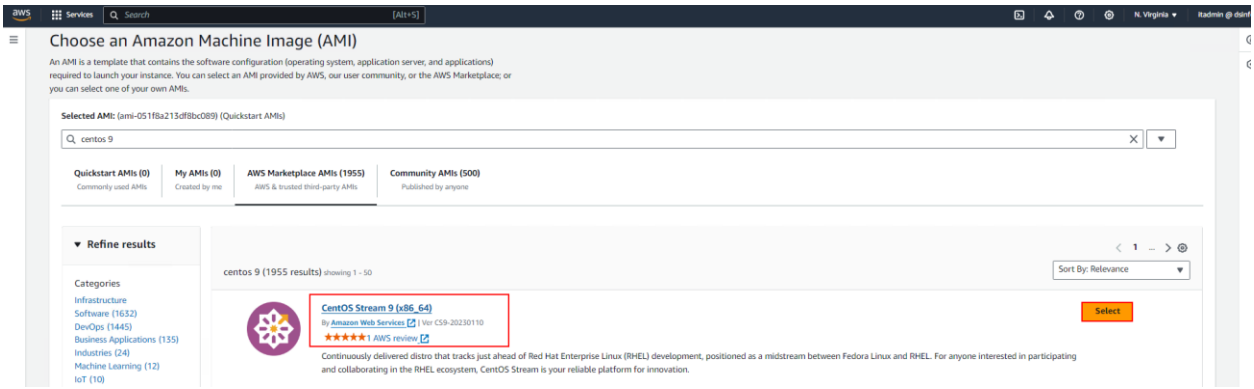
At this stage, specify a name for the instance and set the **Number of Instances** to 3.



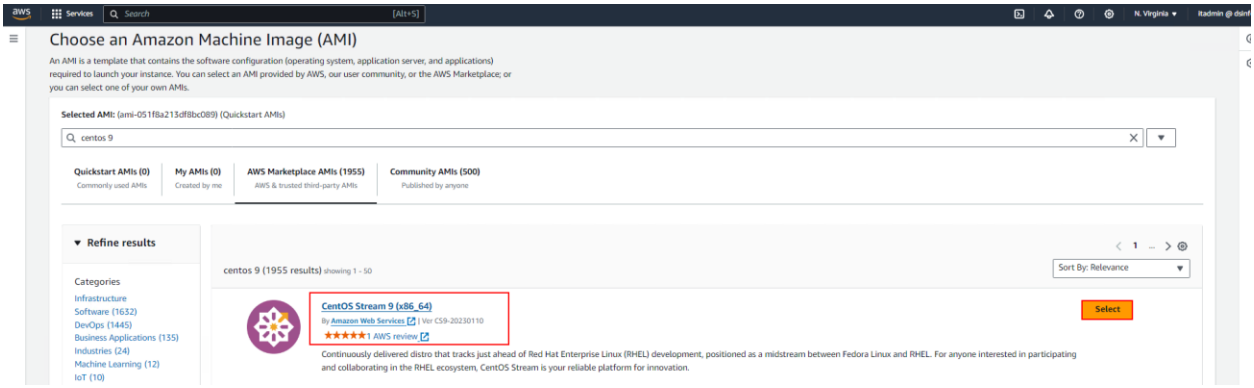
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

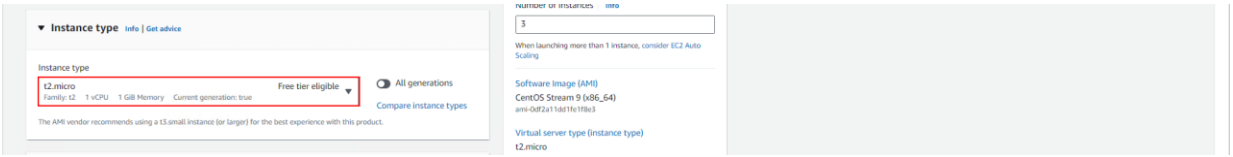
At this stage, in the **AWS AMI Marketplace** section, you need to select **CentOS 9** as the operating system.



Then, at this stage, click on the **Subscribe Now** button.



At this stage, set the Instance Type to **t2.micro**.

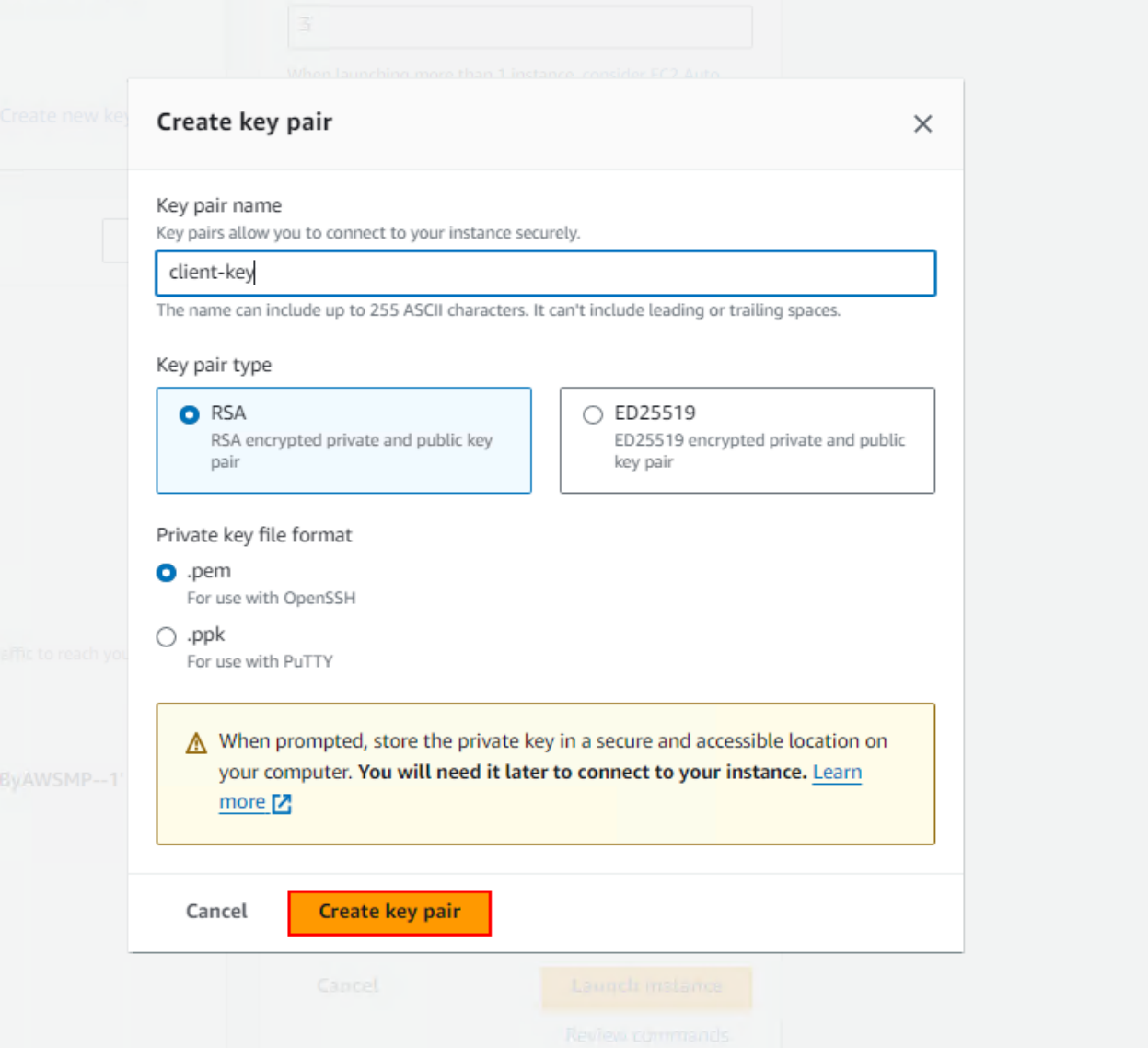


At this stage, in the **Key Pair** section, click on the **Create New Key Pair** button.

Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

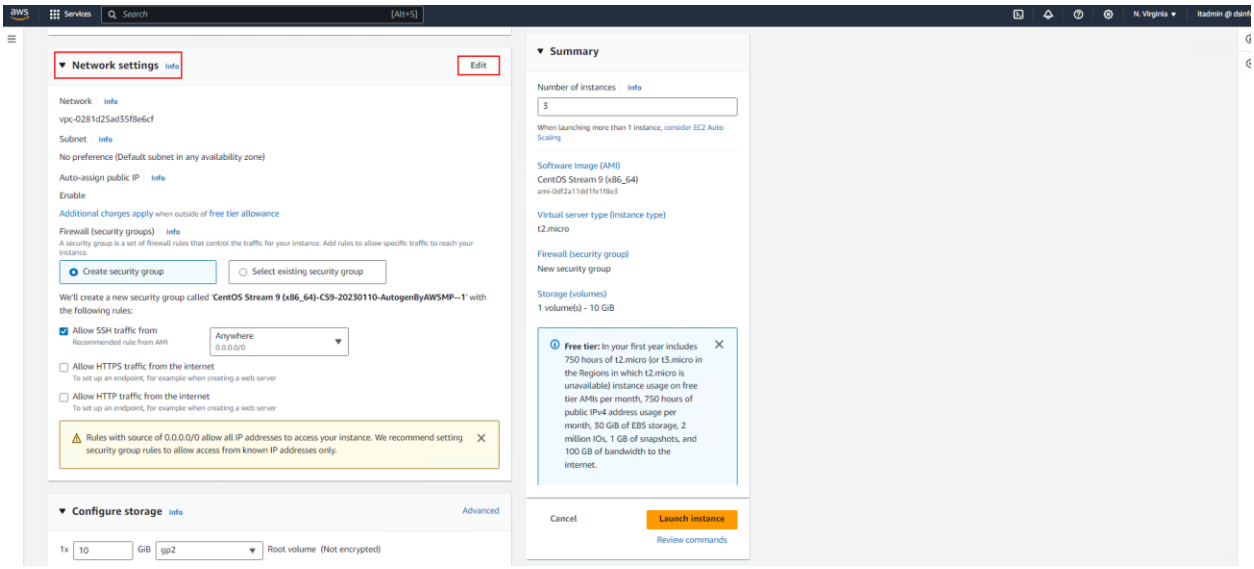
At this stage, specify a name for the key pair and then click on the **Create Key Pair** button.



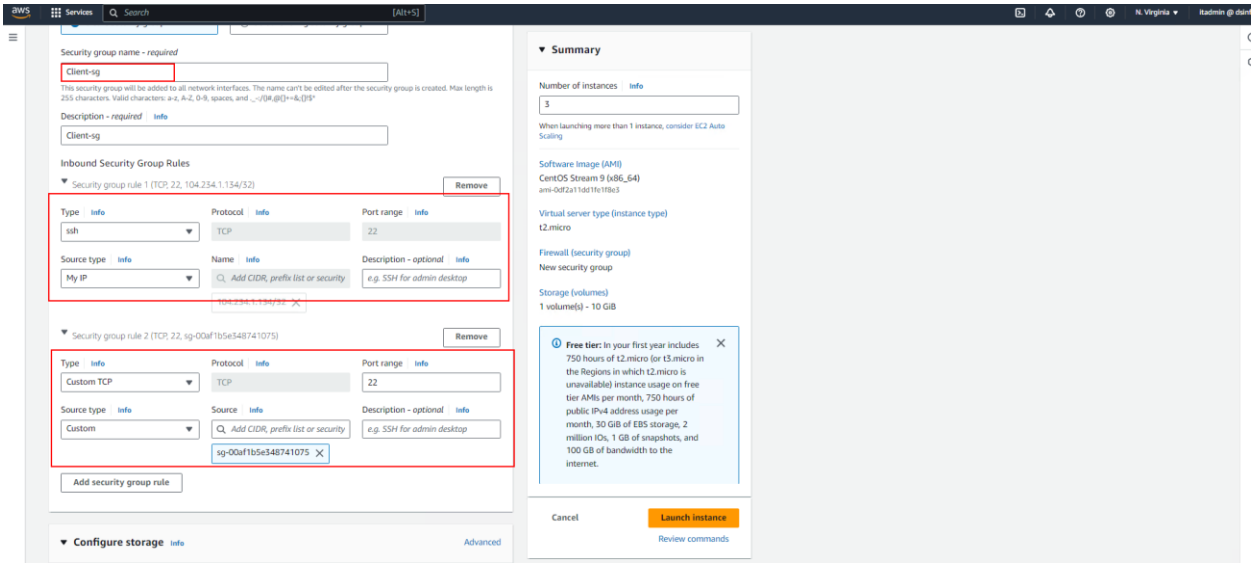
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

In the **Network Settings** section, click on the **Edit** button.



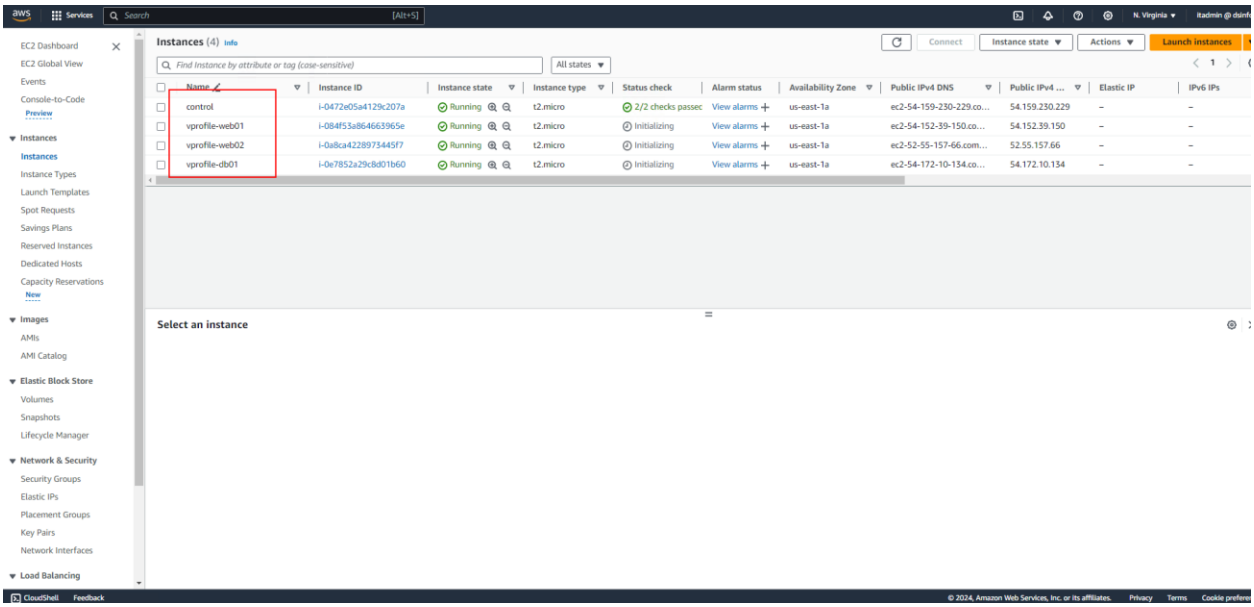
At this stage, specify a name for the security group, then create two inbound rules—one for accessing the instances and another for allowing Ansible to access these instances. Finally, click on the **Launch Instance** button to create the instances.



Instructor: Fariborz Fallahzadeh

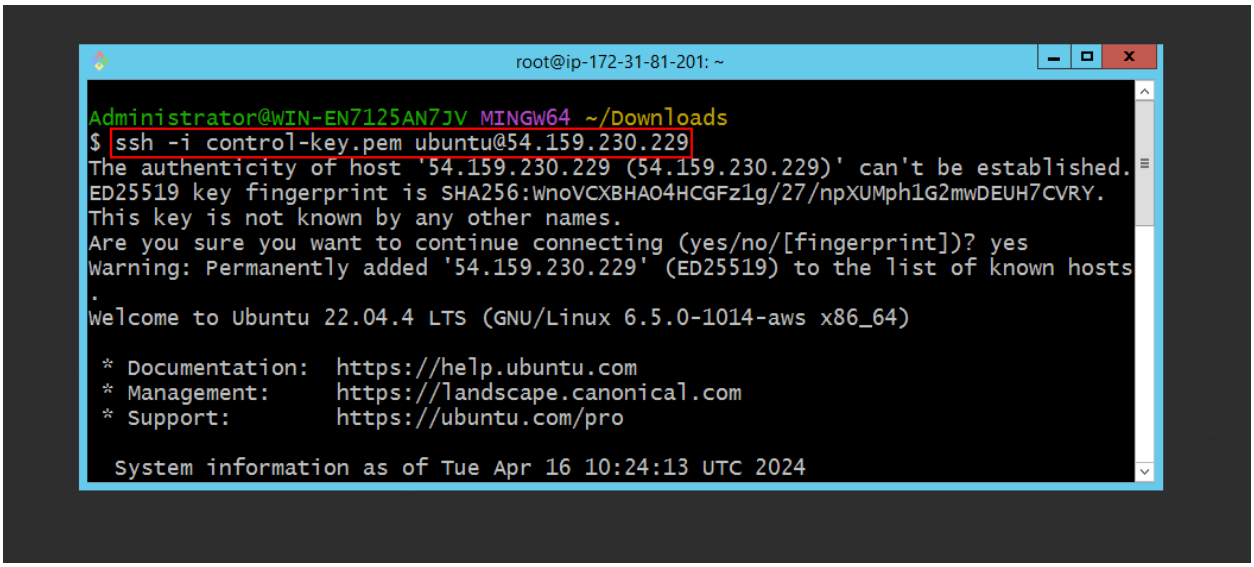
Email Address: fariborz.fallahzadeh@gmail.com

At this stage, edit the name of the instance.



How to Install Ansible on the Control Machine

To install Ansible, you first need to SSH into the corresponding EC2 instance.



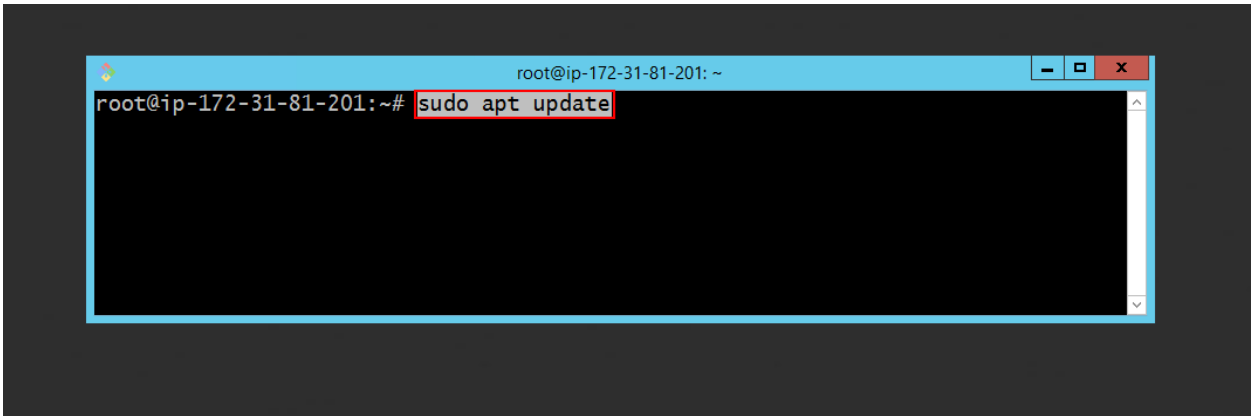
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

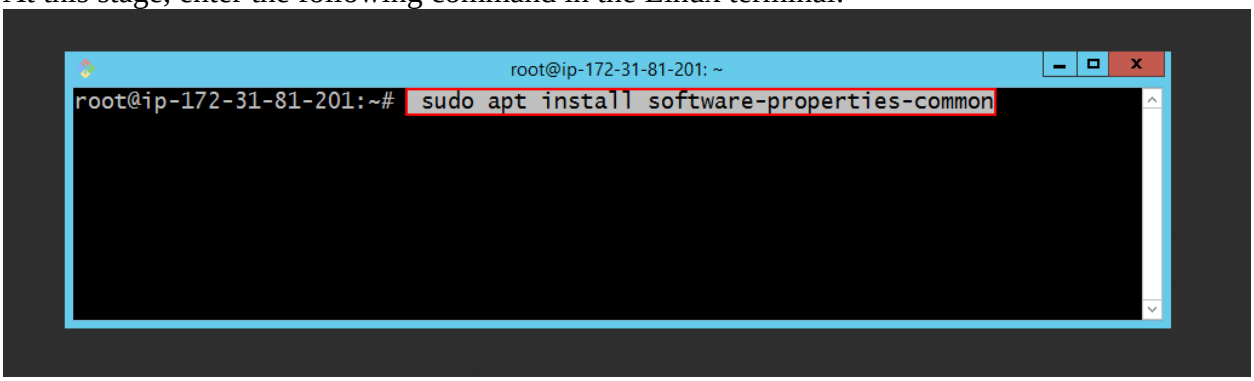
To install Ansible on the Ubuntu operating system, we use the steps provided on the following website.

https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html#installing-ansible-on-ubuntu

In the first step, update the repository once.



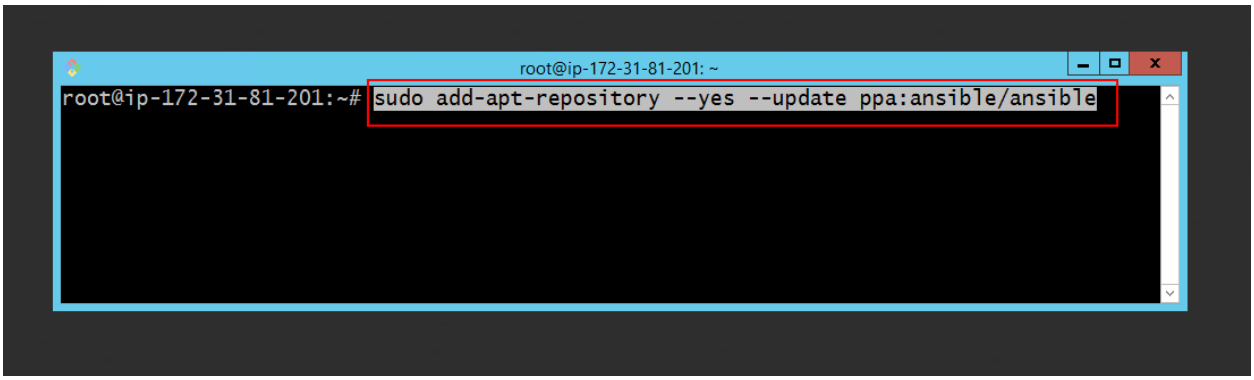
At this stage, enter the following command in the Linux terminal:



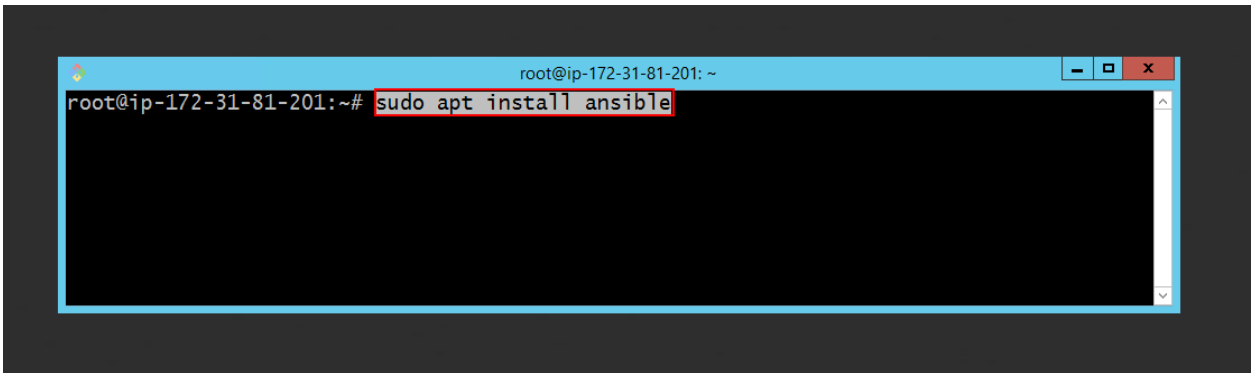
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

Then, you need to add the Ansible repository to the Ubuntu operating system repository using the following command.



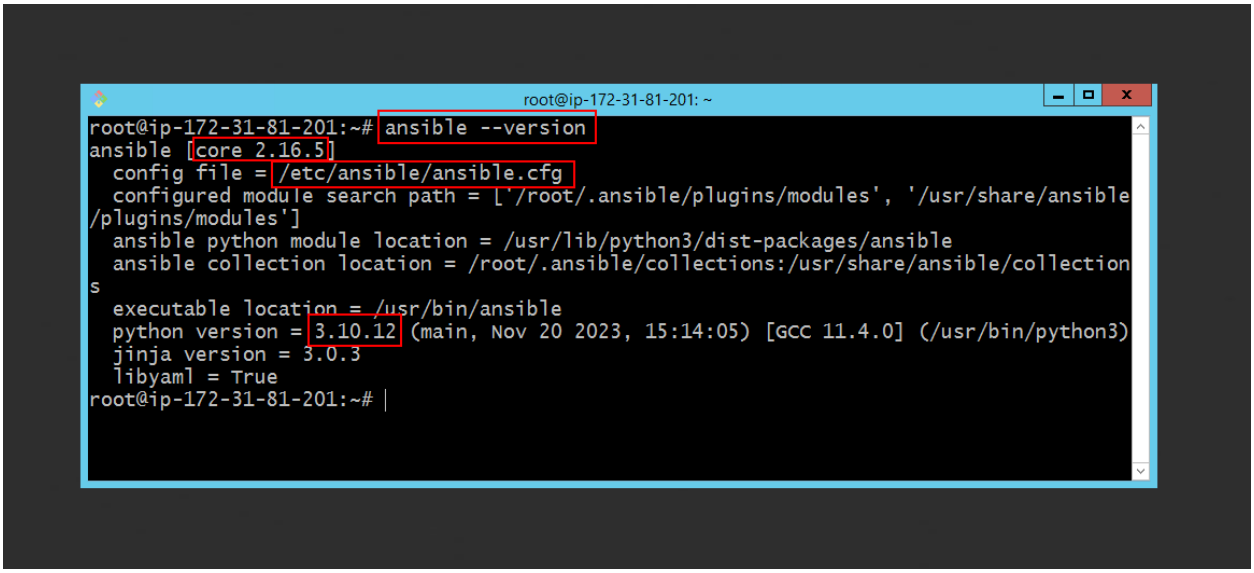
Finally, install Ansible using the following command.



Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

At this stage, check the Ansible version using the following command.



Instructor: Fariborz Fallahzadeh

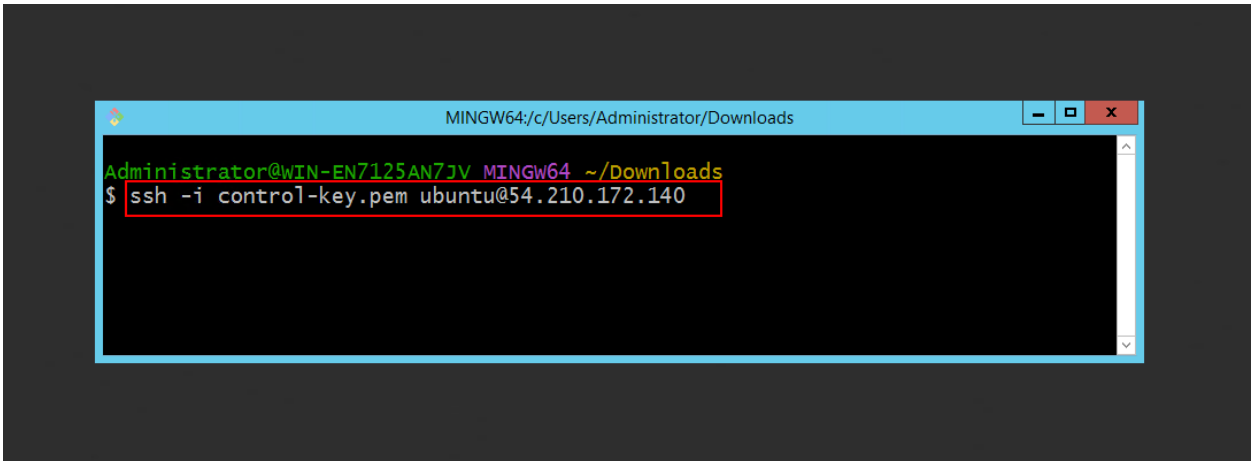
Email Address: fariborz.fallahzadeh@gmail.com

Introduction to Creating an Inventory

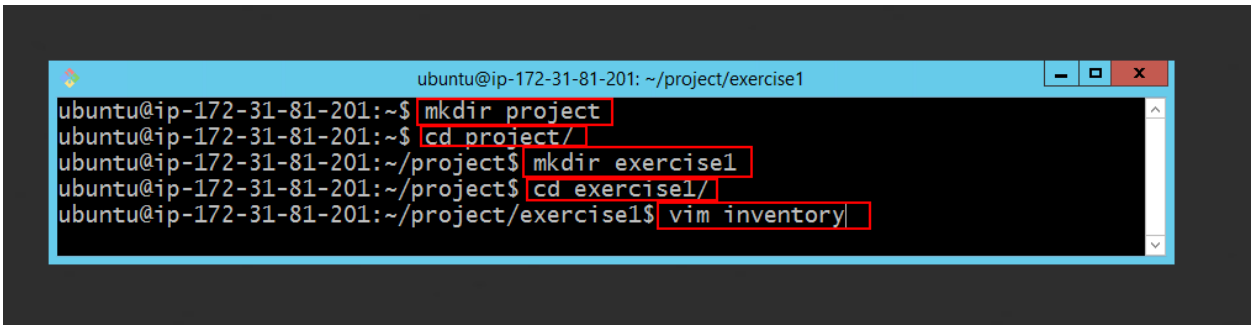
In Ansible, an inventory file is used to connect to targets. An inventory includes necessary information such as IP addresses, usernames, and passwords for connecting to the target machines.

An inventory file can be written in two formats: INI or YAML.

To create the inventory file, SSH into your Ansible EC2 instance.



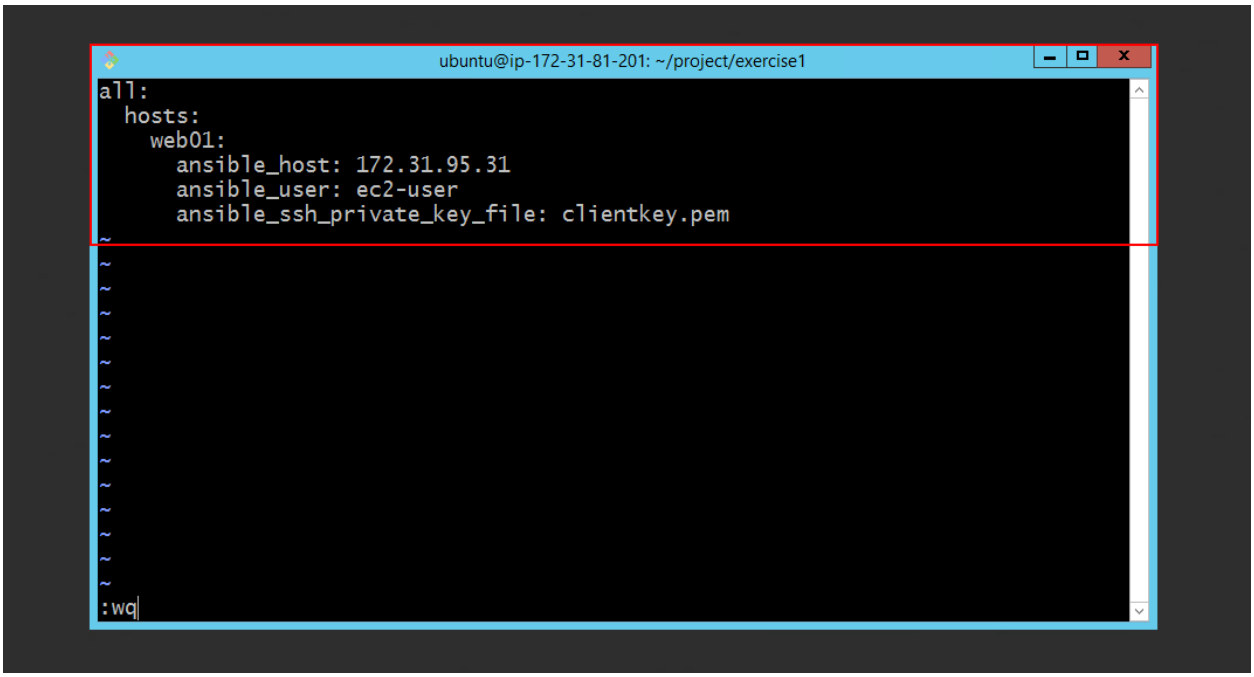
At this stage, create a folder for the project, then create another folder named **Exercise1** for your practice. Inside this folder, use the VIM editor to create a file named **inventory**. You can choose any name for this file.



Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

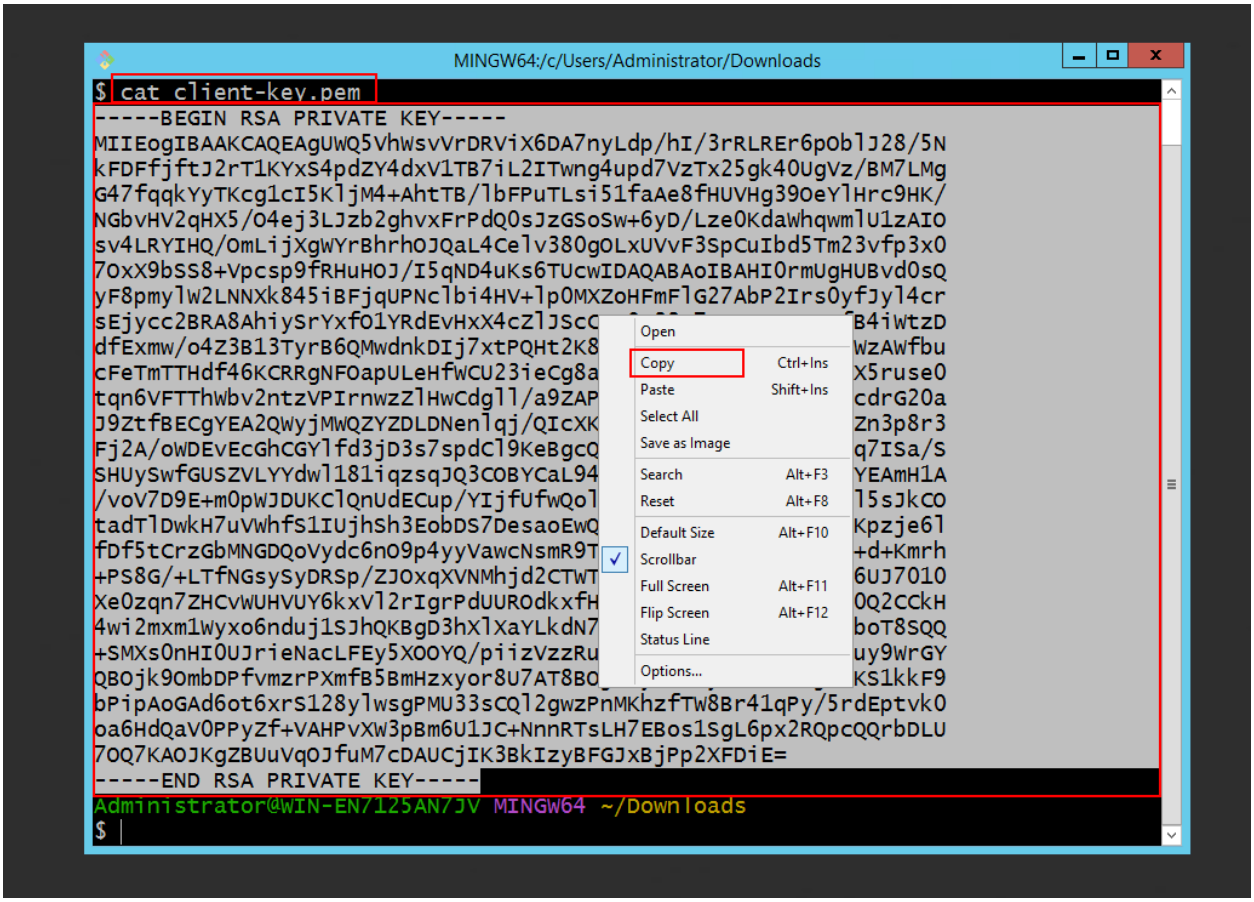
Inside the inventory file, enter the information related to the target.



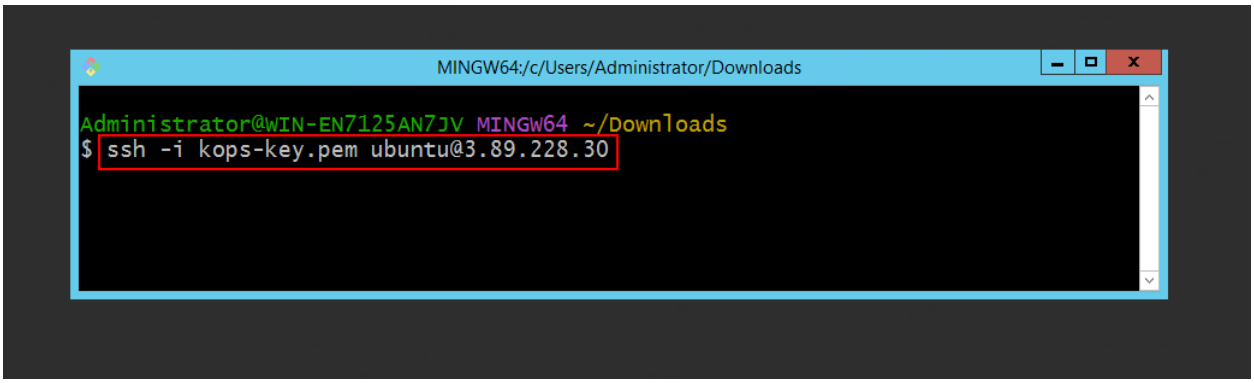
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

Exit from your control machine, and then use the following command to copy the contents of the private key file related to your target.



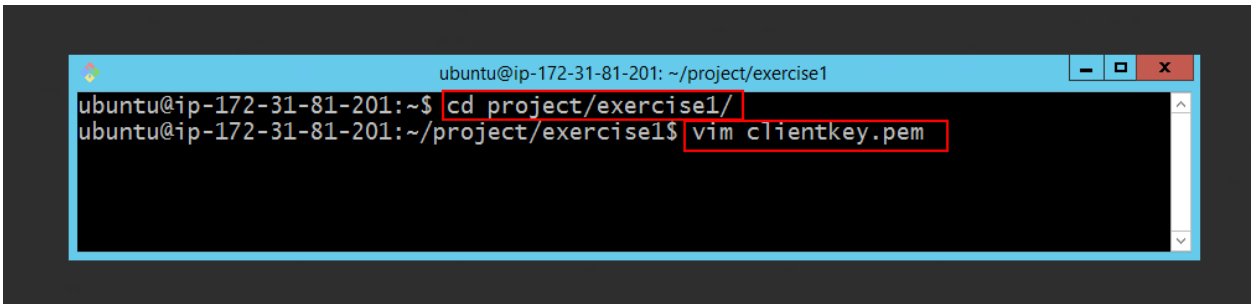
Then, SSH back into your control machine.



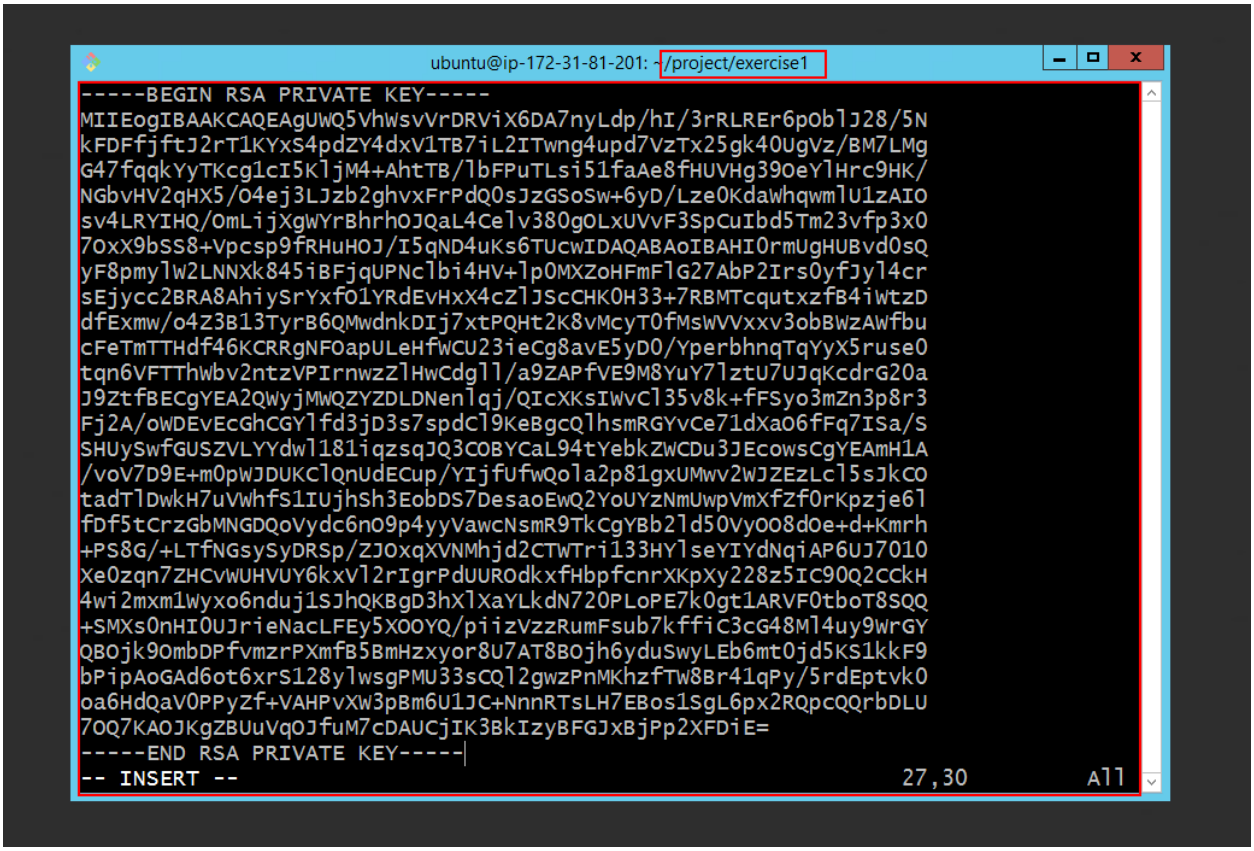
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

Go to the project directory, and then use the VIM editor to create a file with exactly the same name as the private key specified in the inventory file.



Then, paste the contents of the target's private key file that you copied in the previous steps into this file.



Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

Then, at this stage, use the following command to test the connection to the target.

In the following command:

-web01

is the name of the target.

-m

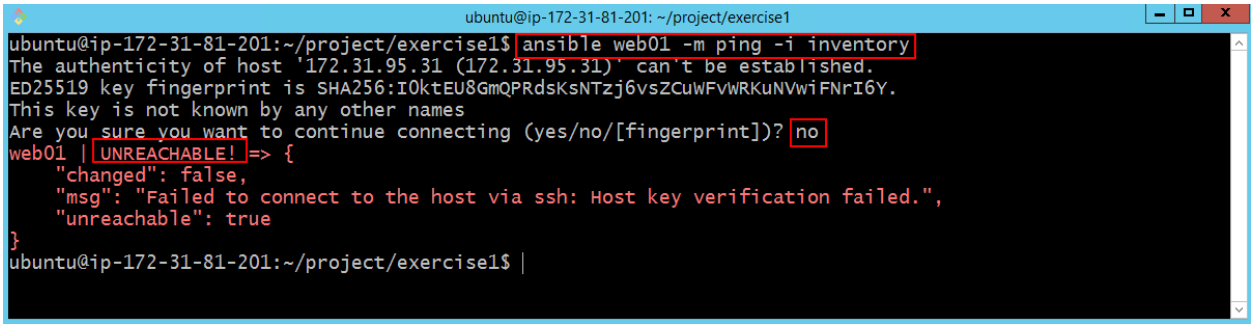
means module.

-Ping

is one of Ansible's modules used for SSH and does not function like a network ping.

-i

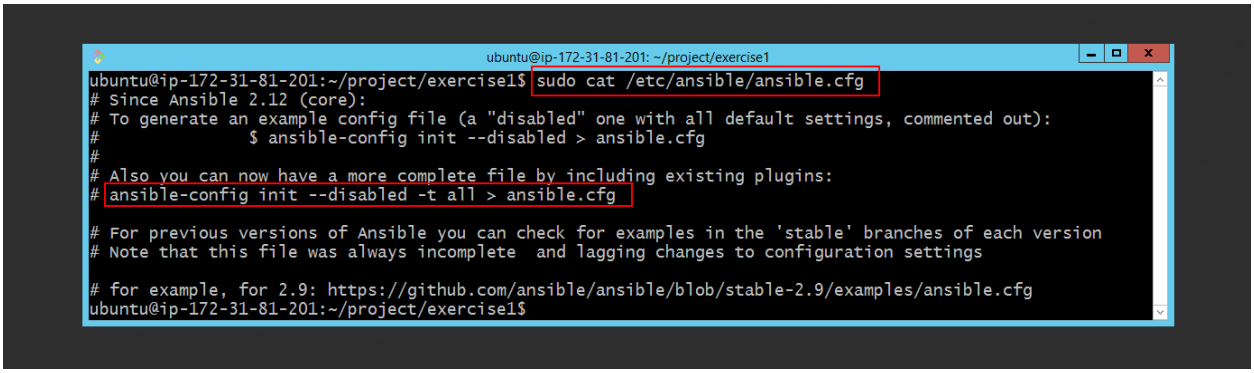
is used to specify the path to the inventory file.



As shown in the image above, when the above command is executed, it asks you to confirm the public key, which is not ideal because it interrupts the automation process. This default behavior should be changed so that no confirmation is requested when using the above command.

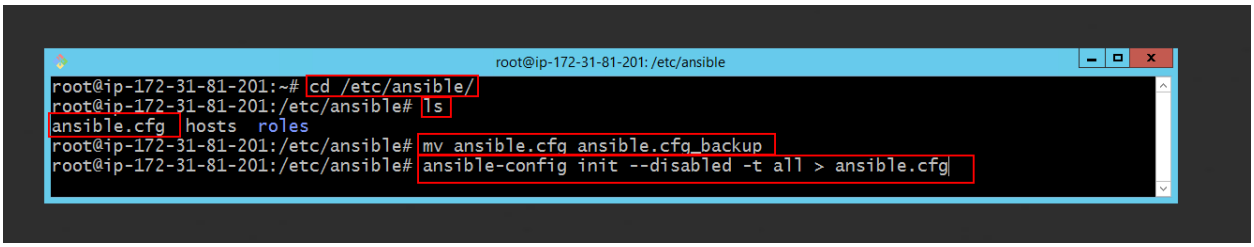
To change Ansible's default settings, you need to check the `ansible.cfg` file, which contains Ansible configurations. The path to this file is shown below, but by default, it doesn't contain any specific content.

You need to create an Ansible configuration to have more settings, which can be done using the `ansible-config` command.



First, use the following command to enter the root user environment.

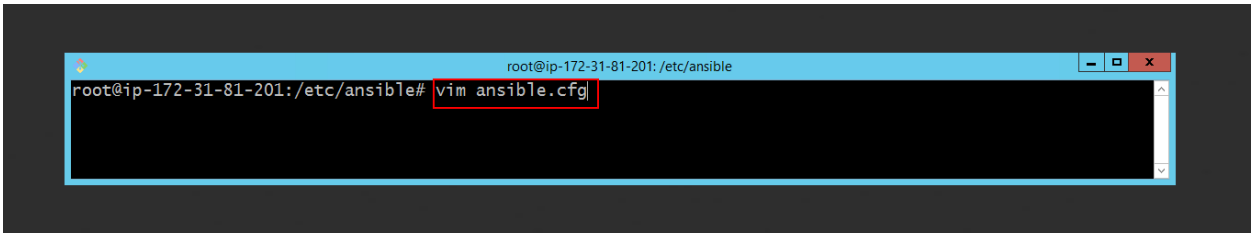
Then, go to the Ansible directory, and use the `mv` command to rename the original Ansible file—in other words, to create a backup of it. After that, use the `ansible-config init` command to generate a new Ansible configuration file with more settings.



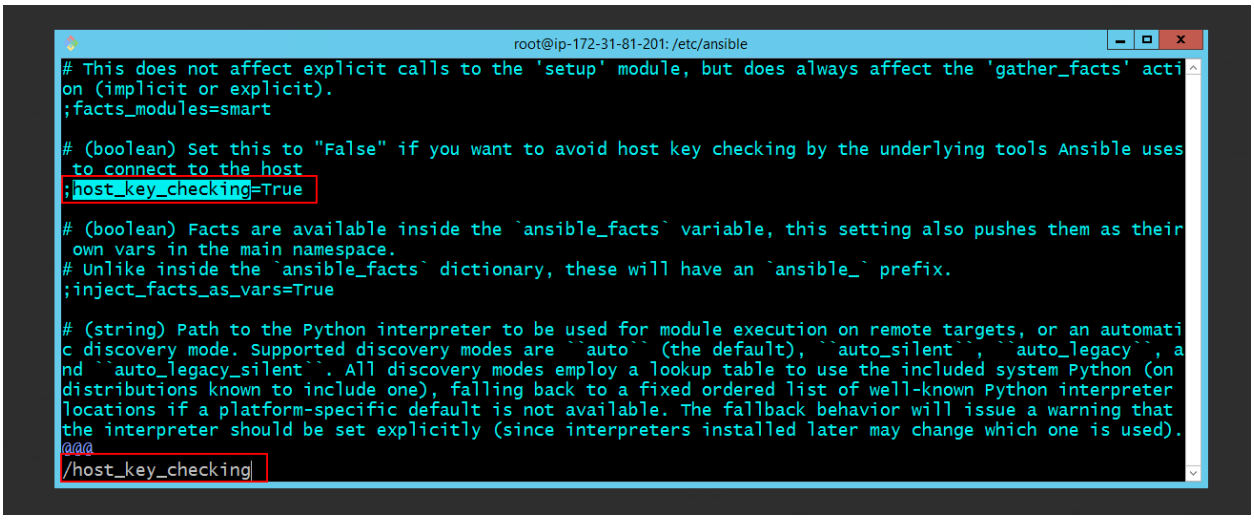
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

At this stage, open the new `ansible.cfg` file using the VIM editor.



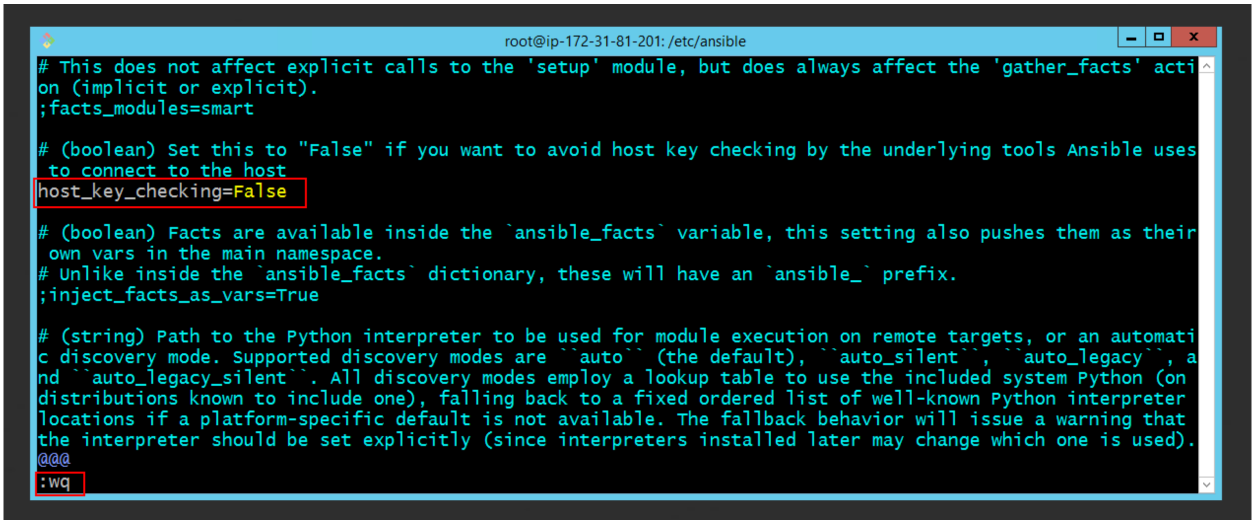
Inside the file, search for the variable `host_key_checking`.



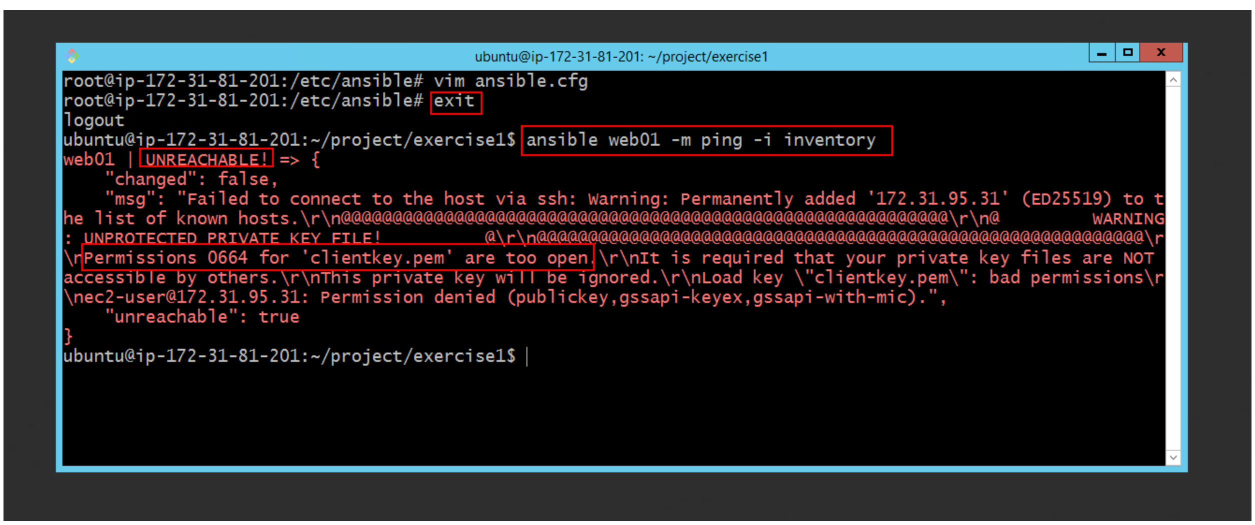
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

At this stage, remove the `;` at the beginning of the line to uncomment the instruction, then change its value from `True` to `False`. This way, Ansible will no longer ask for public key confirmation when connecting to the target. Finally, save the file.



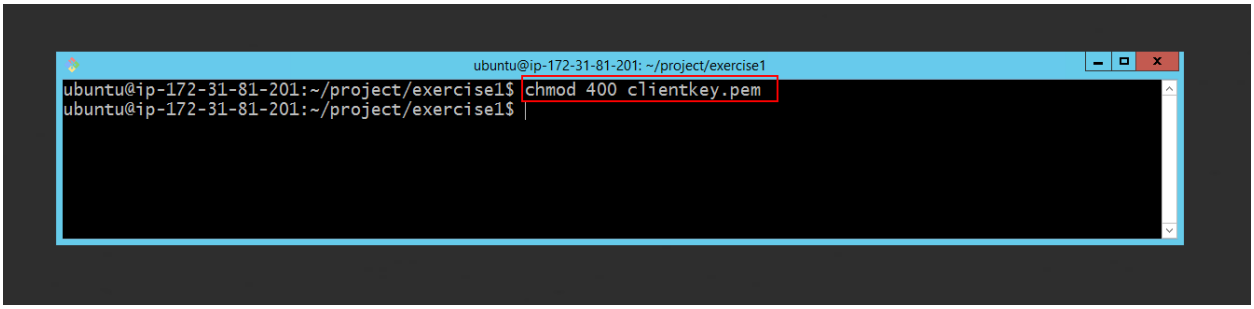
Then, exit the root user environment and use the command below again to connect to the target. However, as shown in the image below, you encounter a permission error when accessing the `Clientkey.pem` file. This means Ansible cannot read the contents of this file. To fix this issue, you need to set the correct permission for the file.



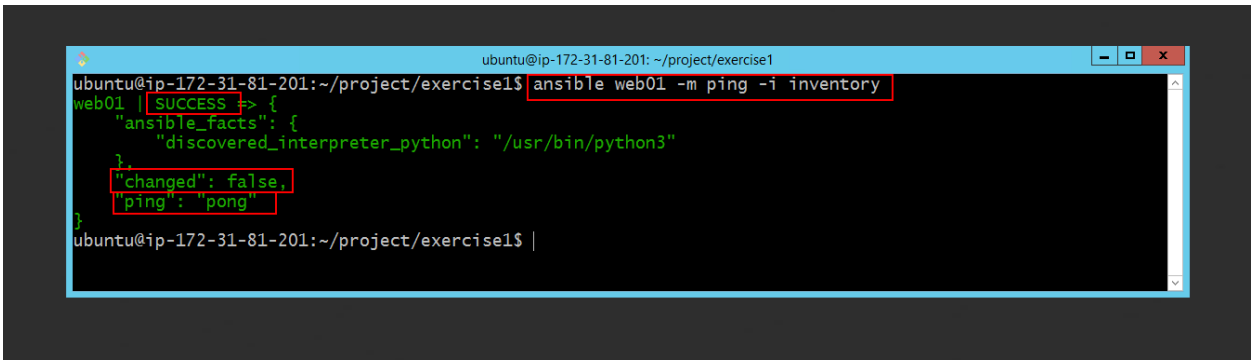
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

Use the following command to change the permission of the `clientkey.pem` file.



As shown in the image below, when the following command is executed, it returns **Success**, which means the connection between the control machine and the target machine was successfully established. The word **False** under the **Changed** section indicates that no changes were made to the target, as only an SSH connection was made without modifying anything. In the **ping** module section, the word **pong** appears, which is the output of the ping module.



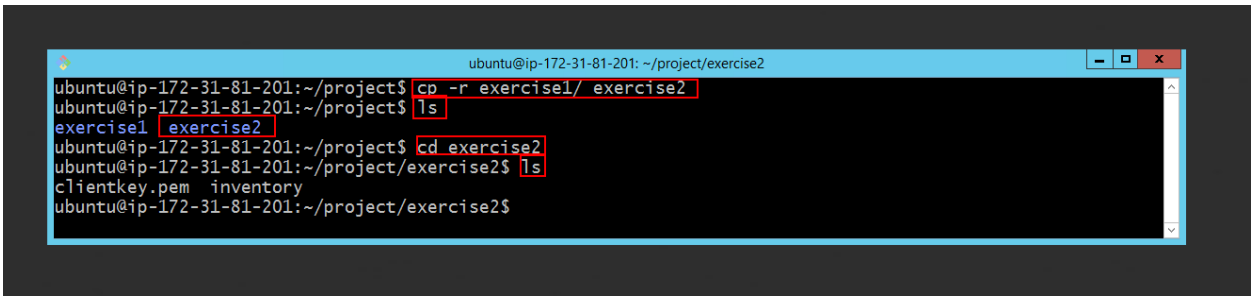
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

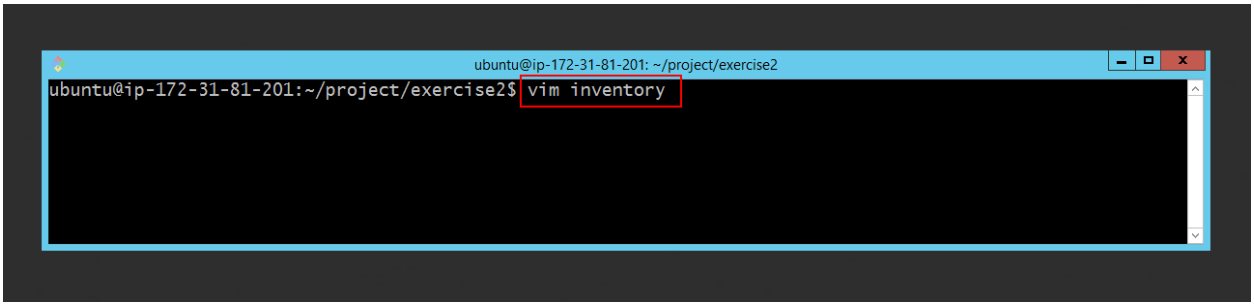
How to Group Targets in the Inventory

One of the things you can do inside the inventory is grouping the targets. By using grouping, you can easily apply certain configurations to one or multiple groups. For example, you can apply specific settings to a group of web servers or a group of databases, and so on.

To get familiar with grouping targets in the inventory, first make a copy of the previous project folder with a new name, and then enter the **exercise2** folder.



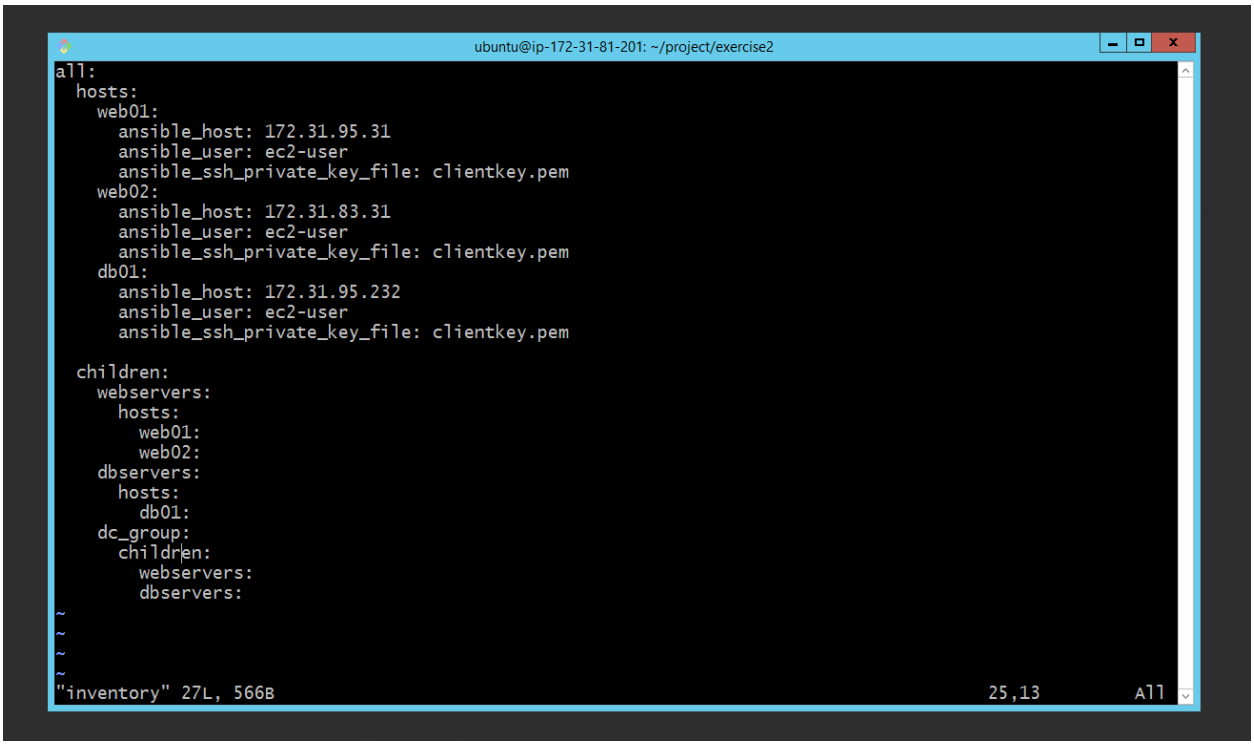
At this stage, open the **inventory** file using the VIM editor.



Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

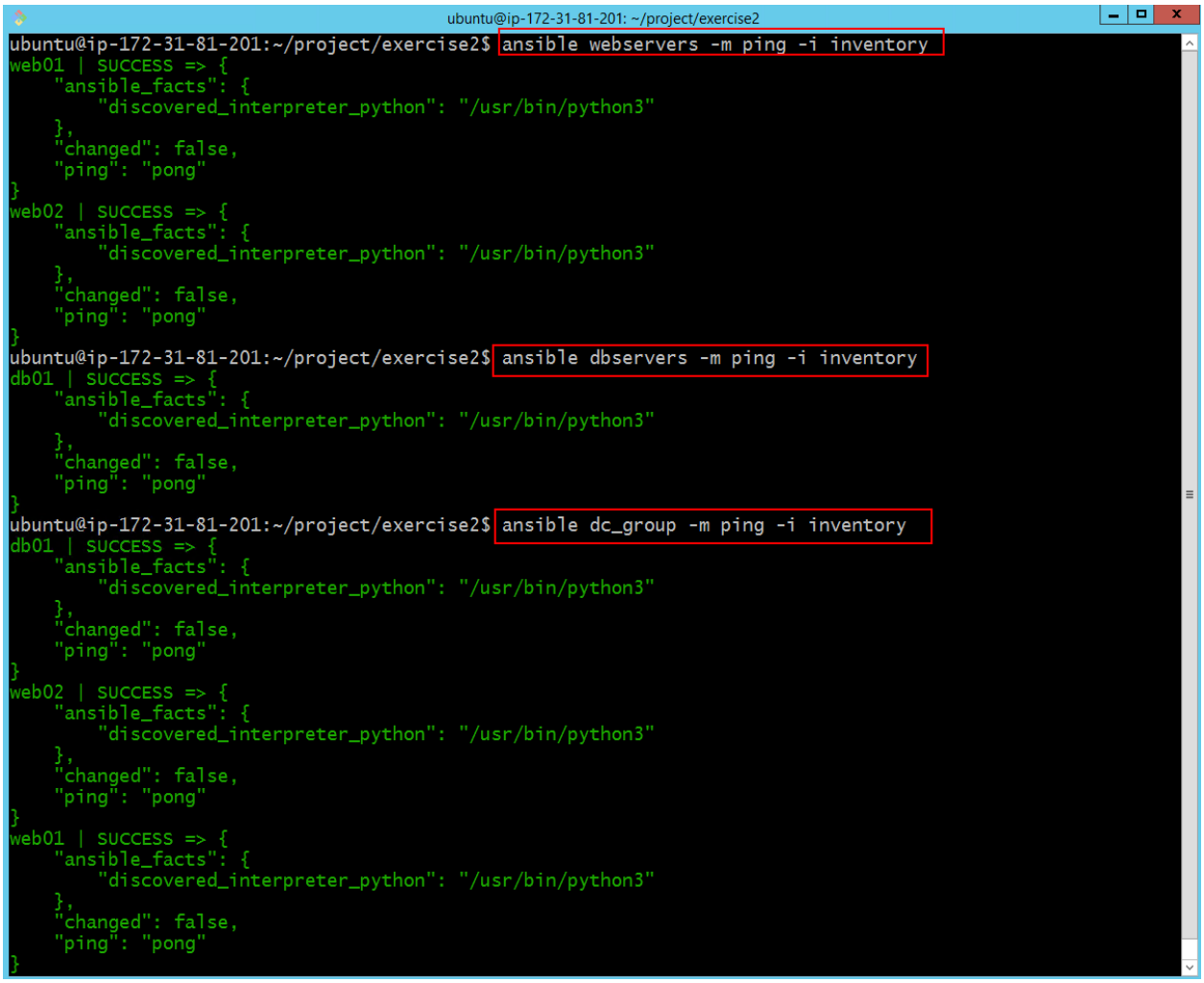
As shown in the image below, we have created three groups in this inventory: **webservers**, **dbservers**, and **dc_group**. The **webservers** group contains the web servers, the **dbservers** group contains the database servers, and the **dc_group** group includes both the **webservers** and **dbservers** groups.



Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

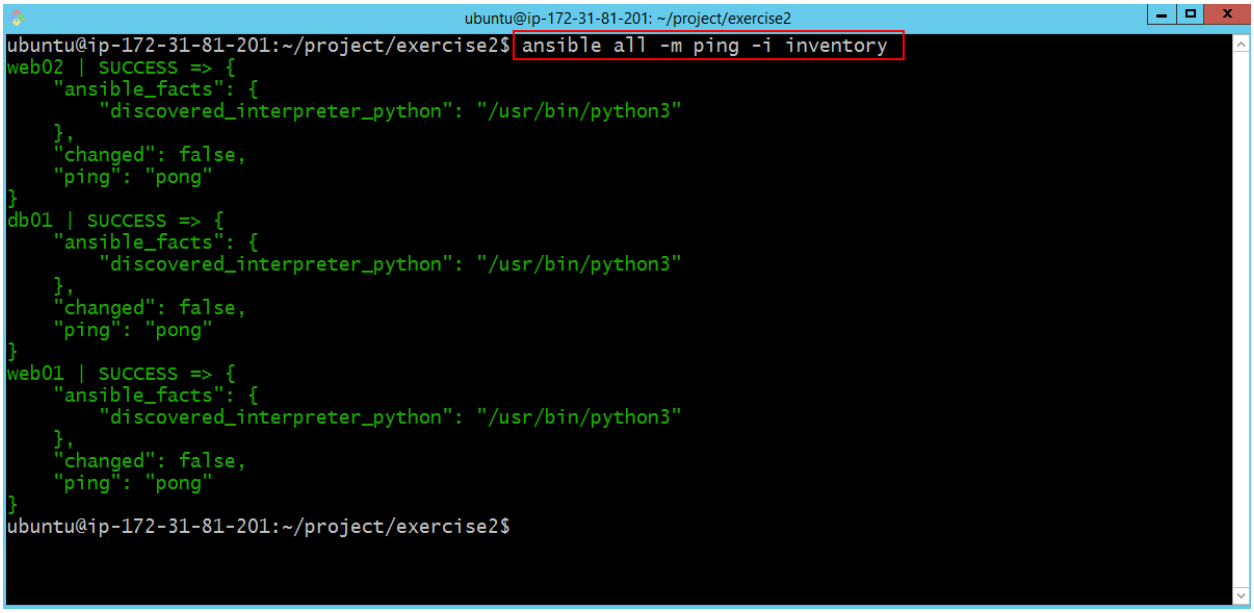
As shown in the image below, when the `ansible` command is used with the **webservers** group, the settings are applied only to the web server group. When used with the **dbservers** group, the settings are applied only to the database server group. And when the **dc_group** group is used in the `ansible` command, the settings are applied to both the **webservers** and **dbservers** groups.



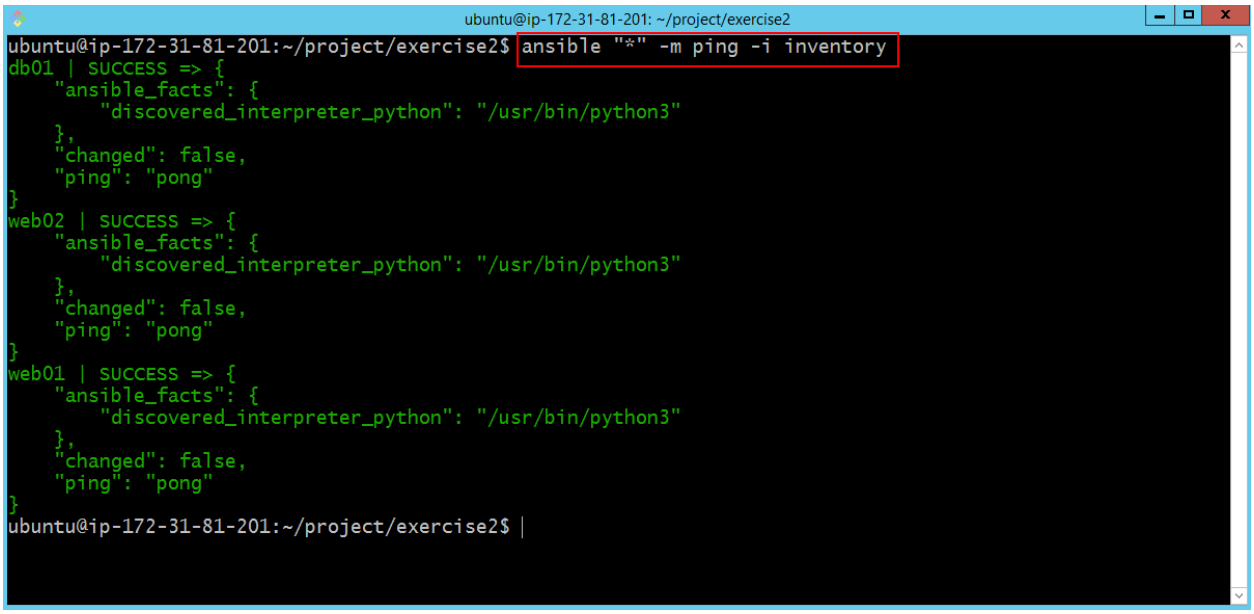
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

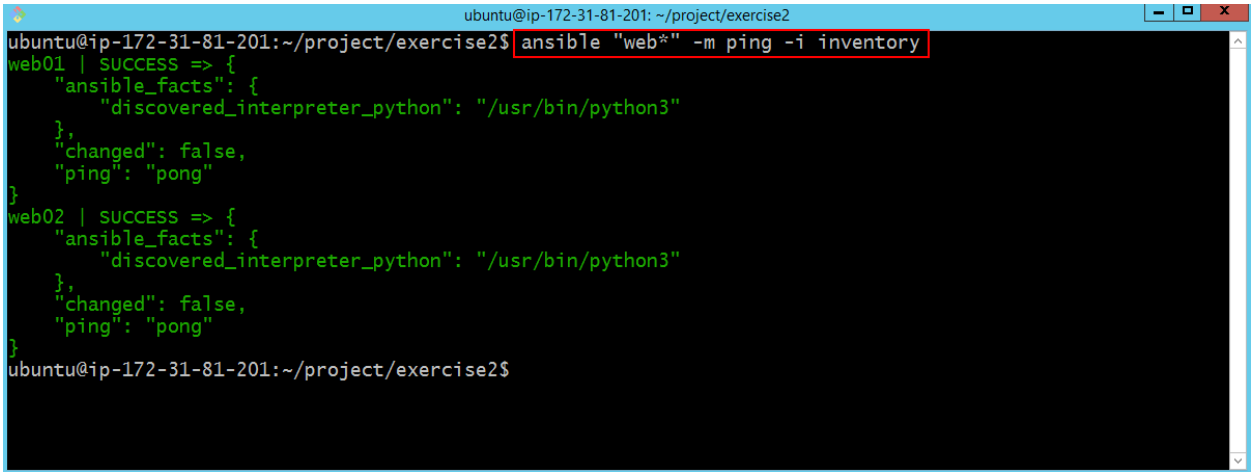
If you use **all** after the `ansible` command, the settings will be applied to all targets.



As shown in the image below, you can also use `*` as a substitute for `all`, and in this case, the settings will be applied to all targets.



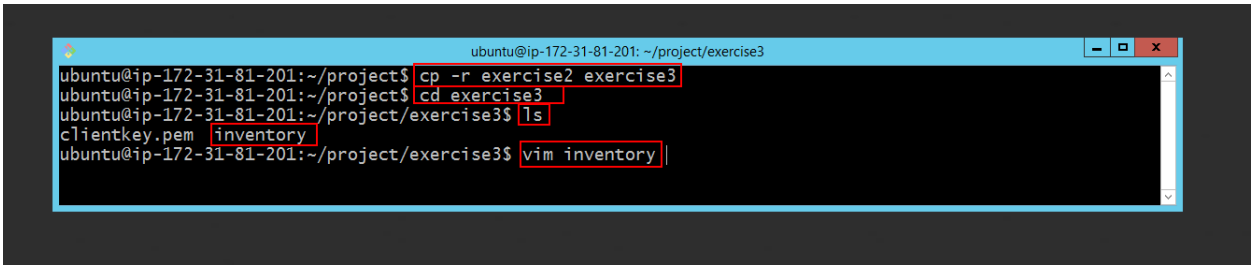
You can also use regular expressions, like in the example below where `*` is used after the word `web`. This means the target includes all machines that start with `web` and are followed by any characters.



How to Use Variables in the Inventory File

We can use variables for repetitive information. Variables defined at the host level have a higher priority than those defined at the group level. In this section, instead of defining the username and SSH private key under each host, we define them at the group level. Variables also help prevent repetition of information.

At this stage, make a copy of the **exercise2** folder with the name **exercise3**, then enter the new folder and edit it using the VIM editor.

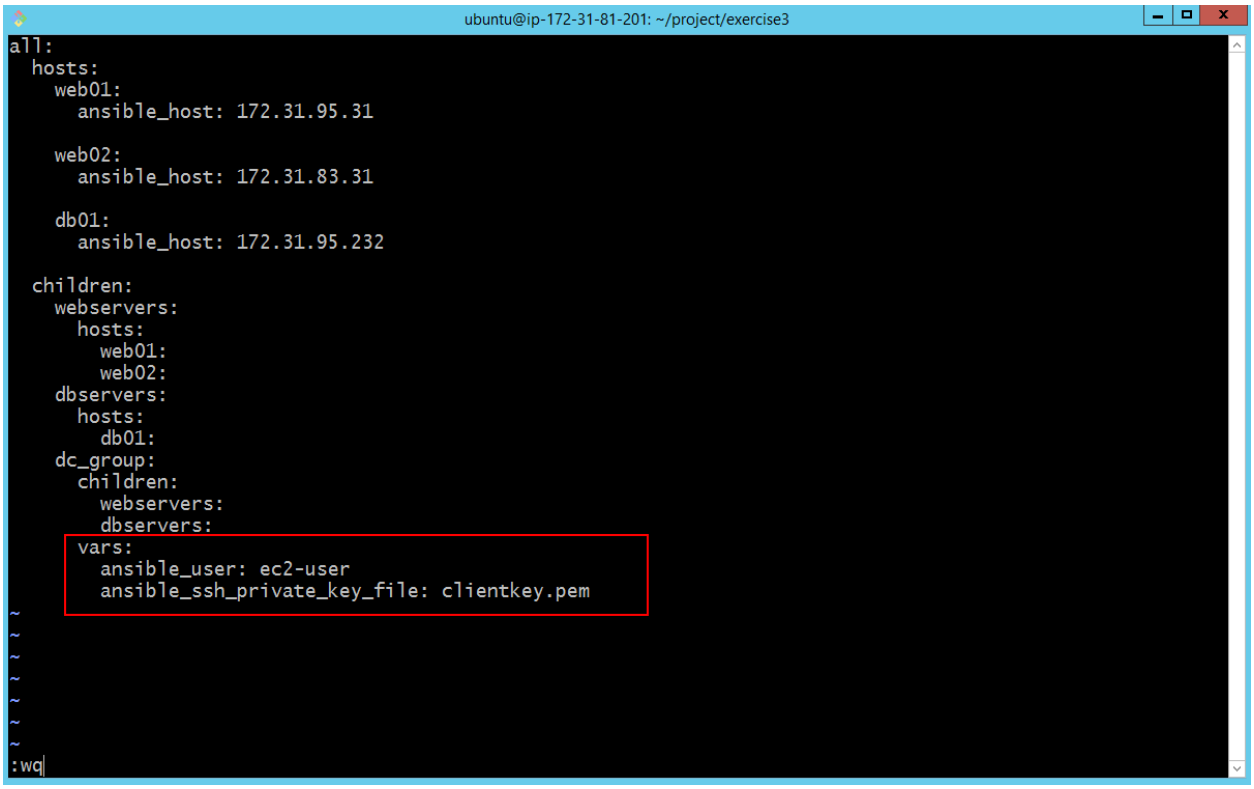


In the inventory file, remove the variables `ansible_user` and `ansible_ssh_private_key` from the host section, and define them under the `dc_group` in the `vars` section. Since this group includes

Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

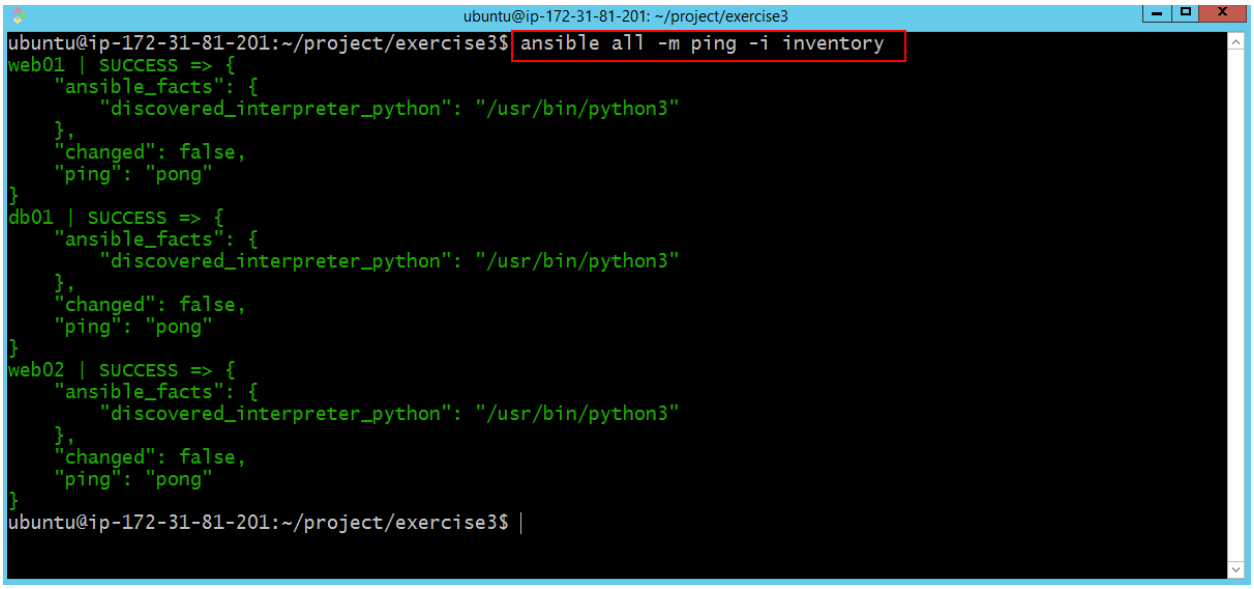
all the hosts, the values of these variables will apply to all of them, avoiding repetition and making the file structure simpler.



Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

At this stage, test the connection to the targets using the following command. As shown in the image below, all connections to the targets have been successfully established.



Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

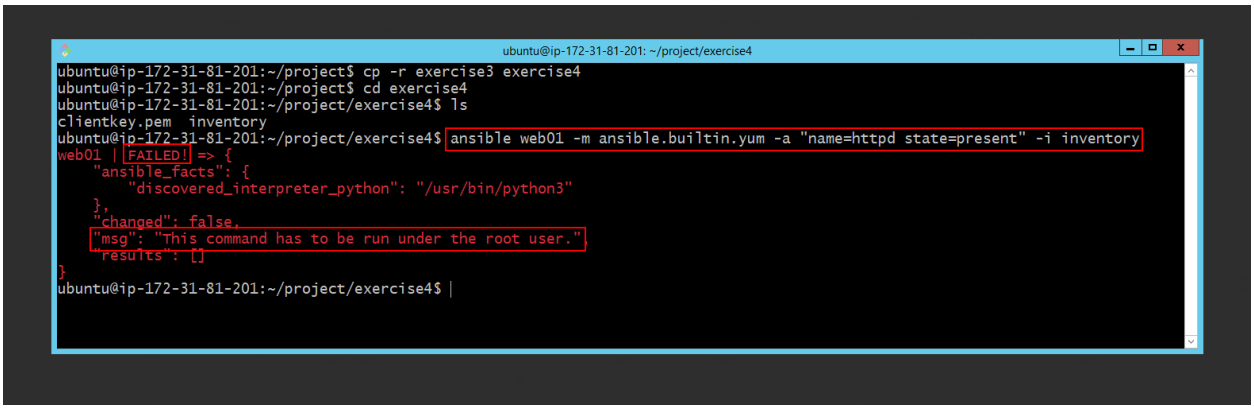
Introduction to Ad Hoc Command

In the previous section, we executed the ping module on the targets, and in this section, we intend to use Ad Hoc commands.

Ansible scripts are known as Ansible Playbooks. Before writing a Playbook, you can execute and test the commands intended for the Playbook in the form of Ad Hoc commands.

In other words, executing commands in the command line using the `ansible` command is known as an Ad Hoc command.

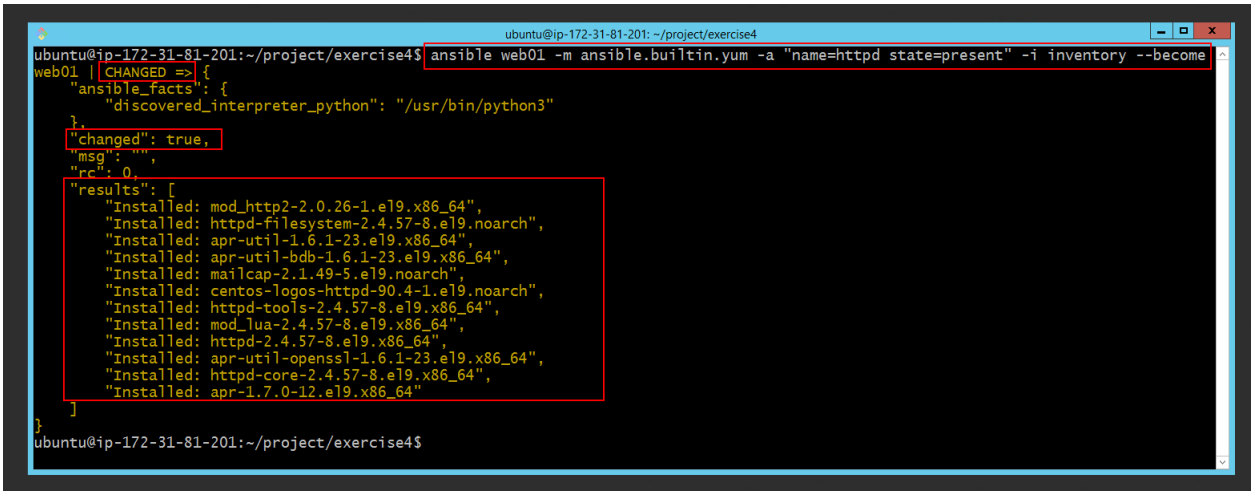
For example, suppose we want to install the `httpd` package on a CentOS Linux system. To do this, we use the `ansible.builtin.yum` module and then use the `-a` switch, which allows you to specify required parameters such as the package name and state. In this example, we use `present` as the state, which means that if the package is not installed on the target, it should be installed. However, as shown in the image below, when we run this Ad Hoc command, we encounter the following error, which means that the command must be executed on the destination system with root privileges.



Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

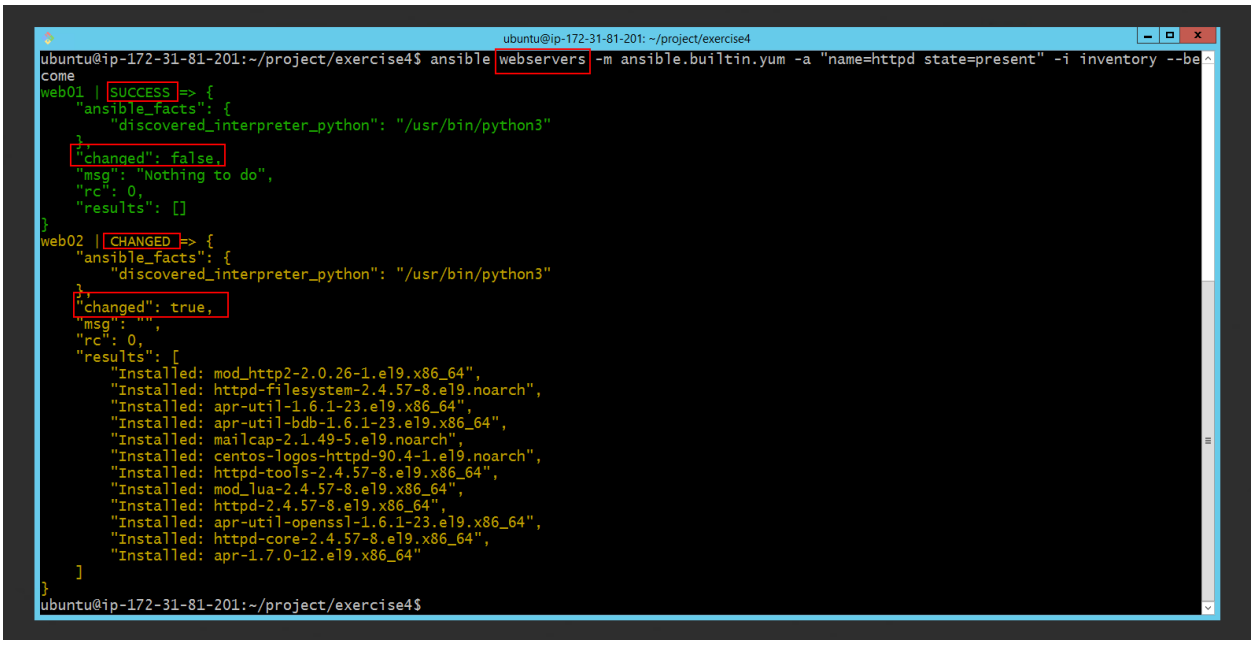
To run the following command with root privileges, we use the `--become` switch.



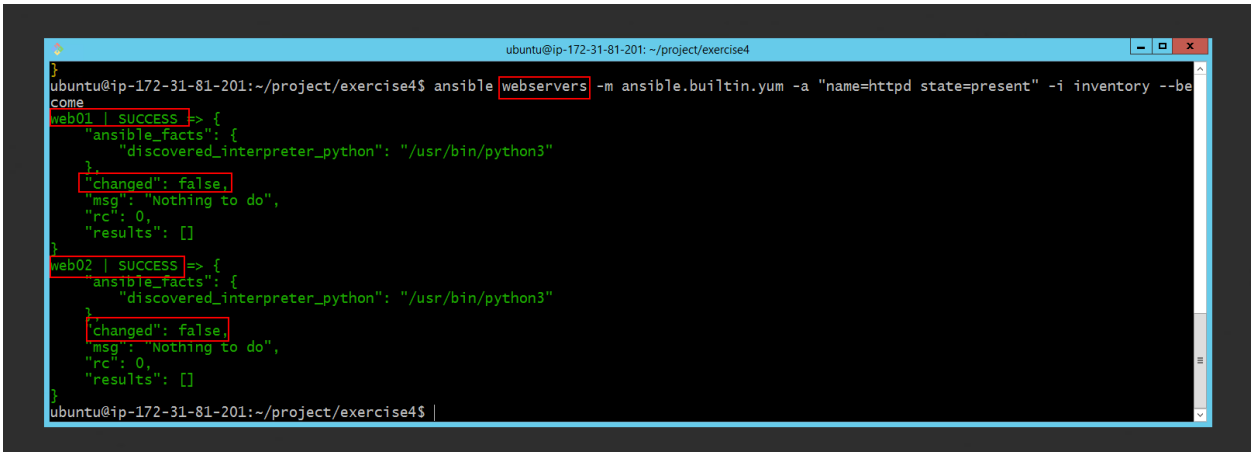
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

In the image below, pay attention to the word **changed**. Here you can see that Ansible can maintain configuration state. When you want to apply a configuration to a target, it first compares the configuration with the previous state. If the configurations are the same, it does not apply any changes to the target, and the value of **changed** will be **false**. However, if the changes differ from the current configuration state, Ansible applies the changes to the target, and the value of **changed** becomes **true**.



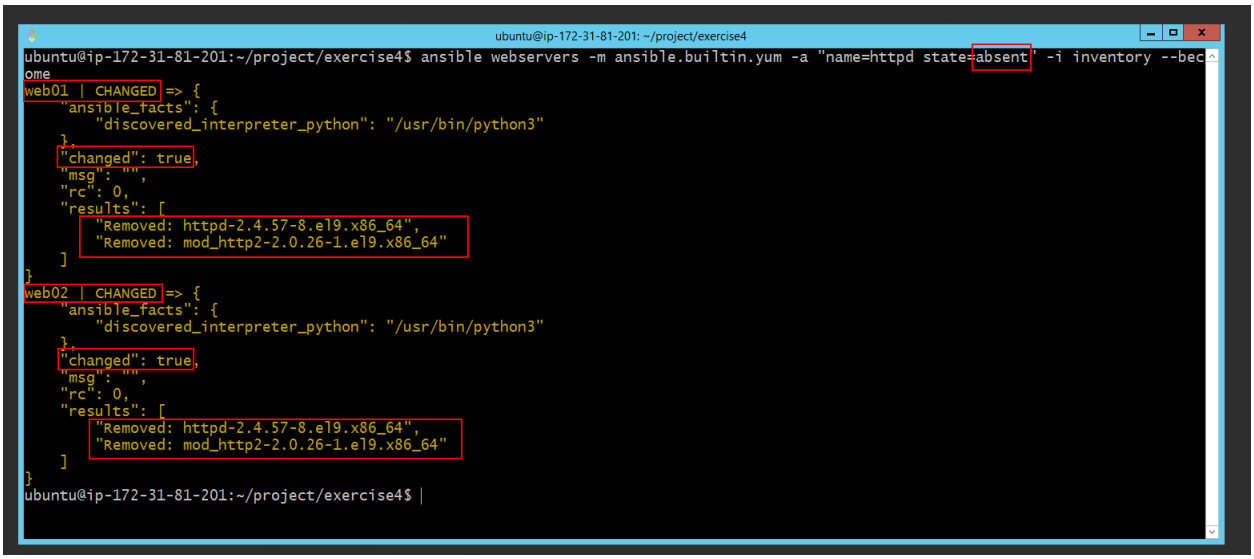
As shown in the image below, because we are trying to apply the same settings again to the target, Ansible does not apply these changes to the target.



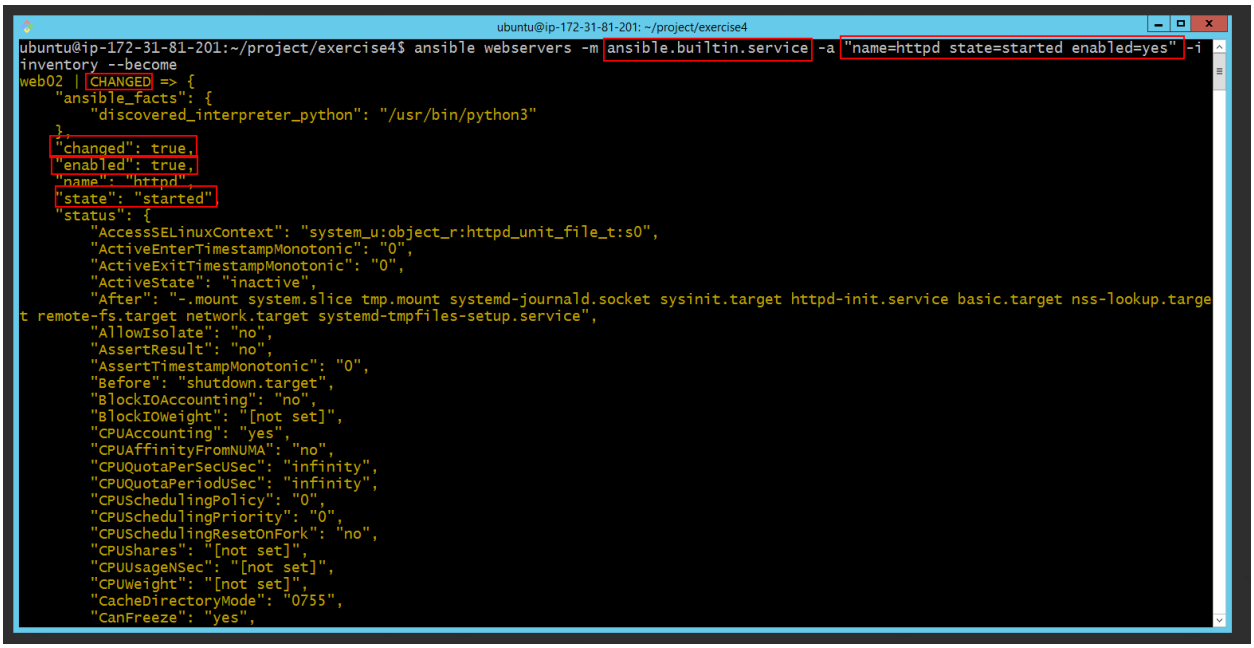
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

Now, at this stage, we change the state to **absent**. This state is used for uninstalling or removing configurations.



At this stage, we use another module called `ansible.builtin.service` for working with services. The `name` parameter specifies the name of the service, the `state` parameter defines the service status, and the `enabled` parameter is used to enable the service.

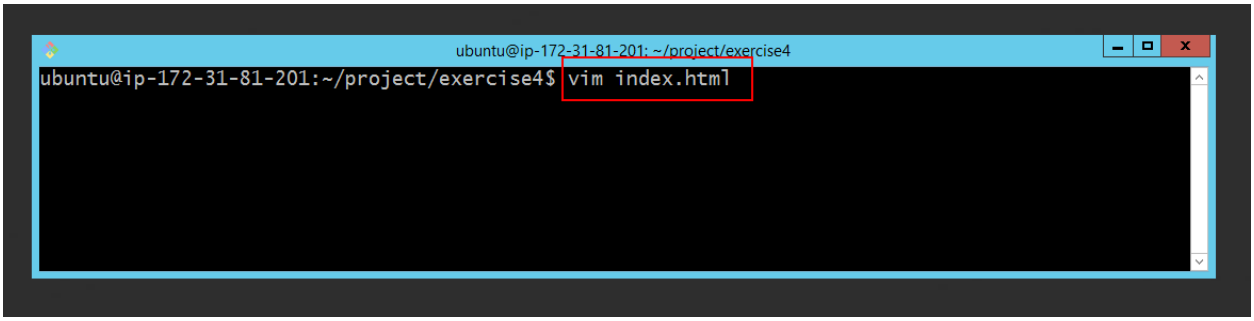


Instructor: Fariborz Fallahzadeh

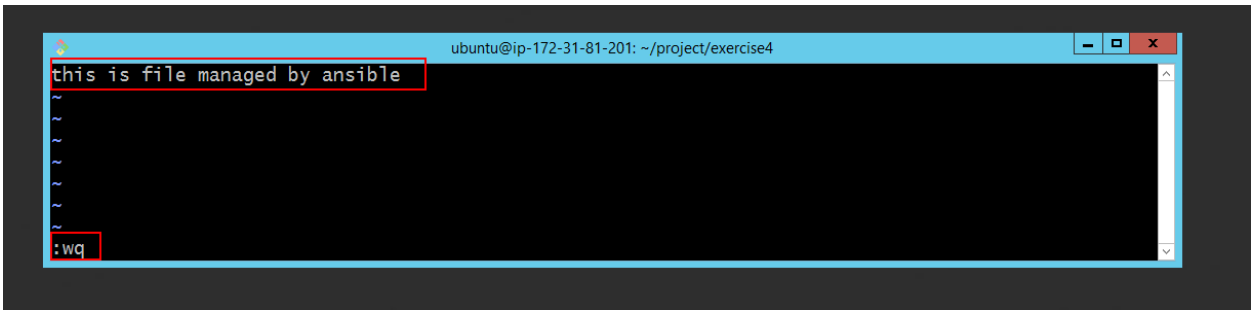
Email Address: fariborz.fallahzadeh@gmail.com

At this stage, we intend to create an HTML file on the control machine and then copy it to the target machine using Ansible modules.

Use the following command to create an HTML file.



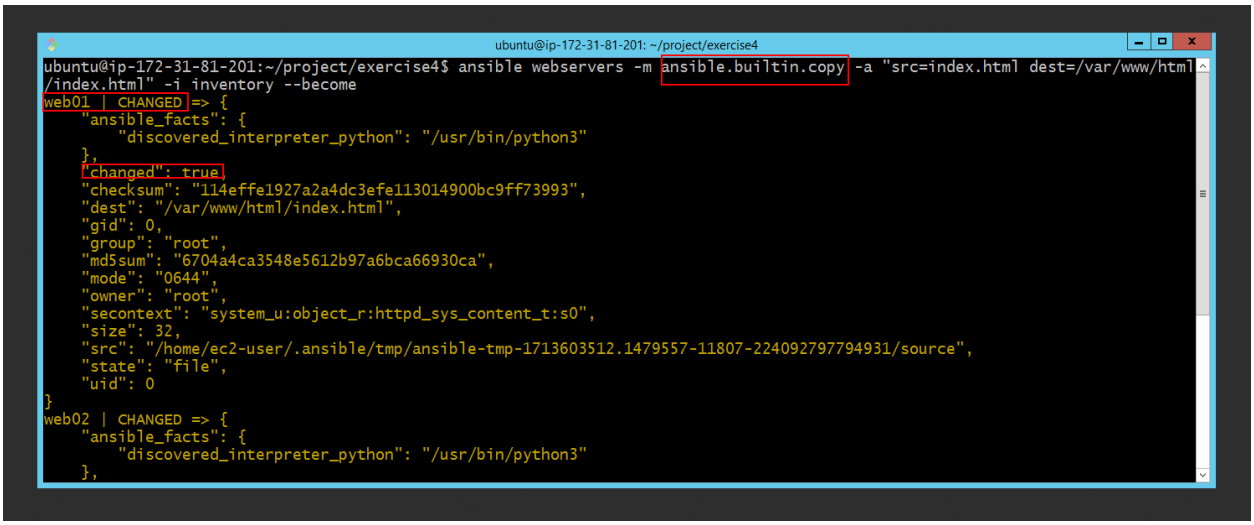
At this stage, write a test message inside the HTML file and then save it.



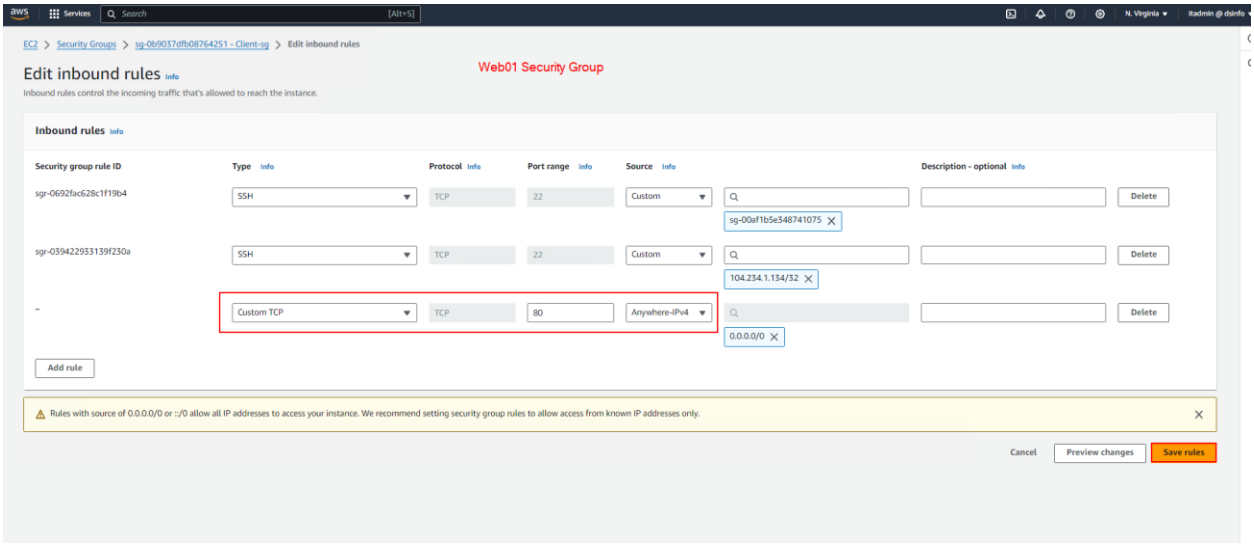
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

To copy a file from the Ansible machine to the target machine, you can use the `ansible.builtin.copy` module. In this module, you can use the `src` and `dest` parameters, which specify the source and destination paths for the file copy.



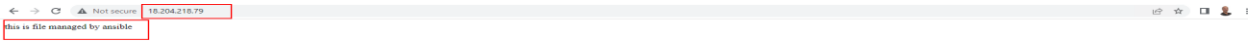
To test whether the file has been copied to the target, simply open port 80 on the target. Note that the target is a web server and the HTML file has been copied to the web server's directory.



Instructor: Fariborz Fallahzadeh

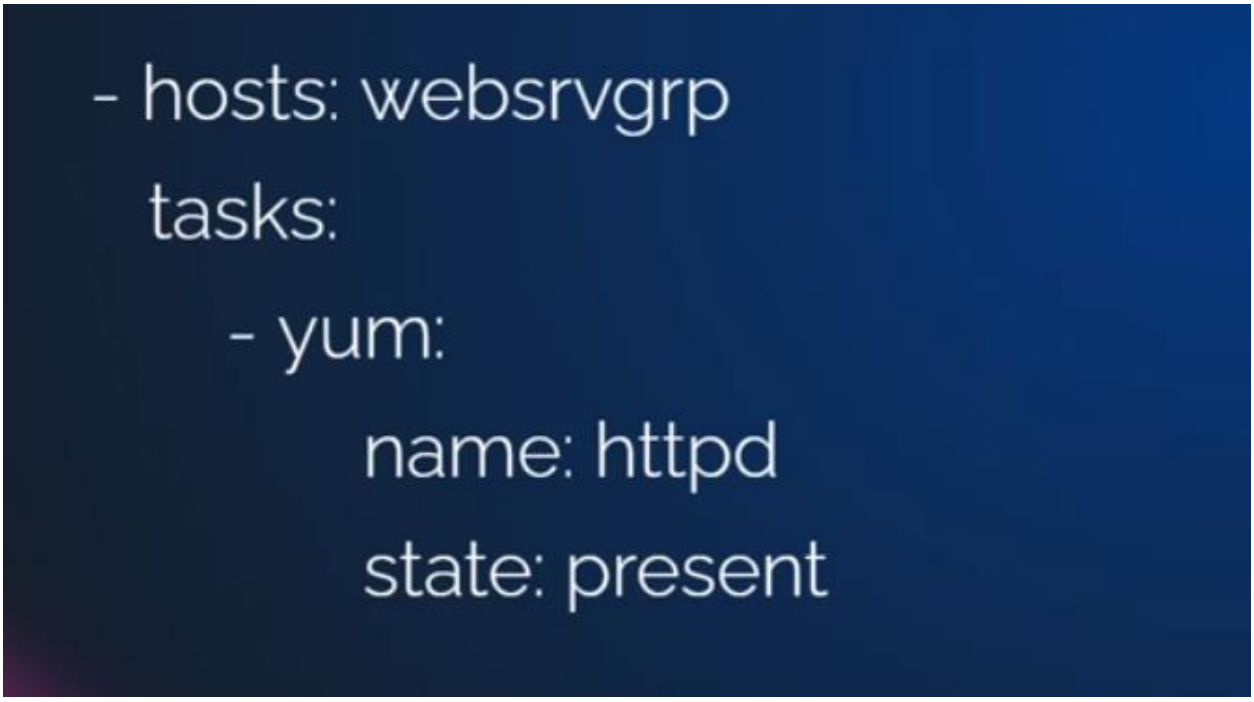
Email Address: fariborz.fallahzadeh@gmail.com

As shown in the image below, when we enter the target's IP address in the browser, we can see the contents of the HTML file.



Introduction to Playbook and Module

A Playbook is like a script, similar to a Bash script. A Playbook is a collection of plays and is written in YAML format. In a Playbook, the **hosts** section specifies the targets, which in the example below is the **webservers** group. The **tasks** section contains the actions to be performed on the targets. Then, a module can be used—in the example below, **yum** is the name of the module used in the Playbook. For each module, a series of options or arguments can be specified in the form of key-value pairs.

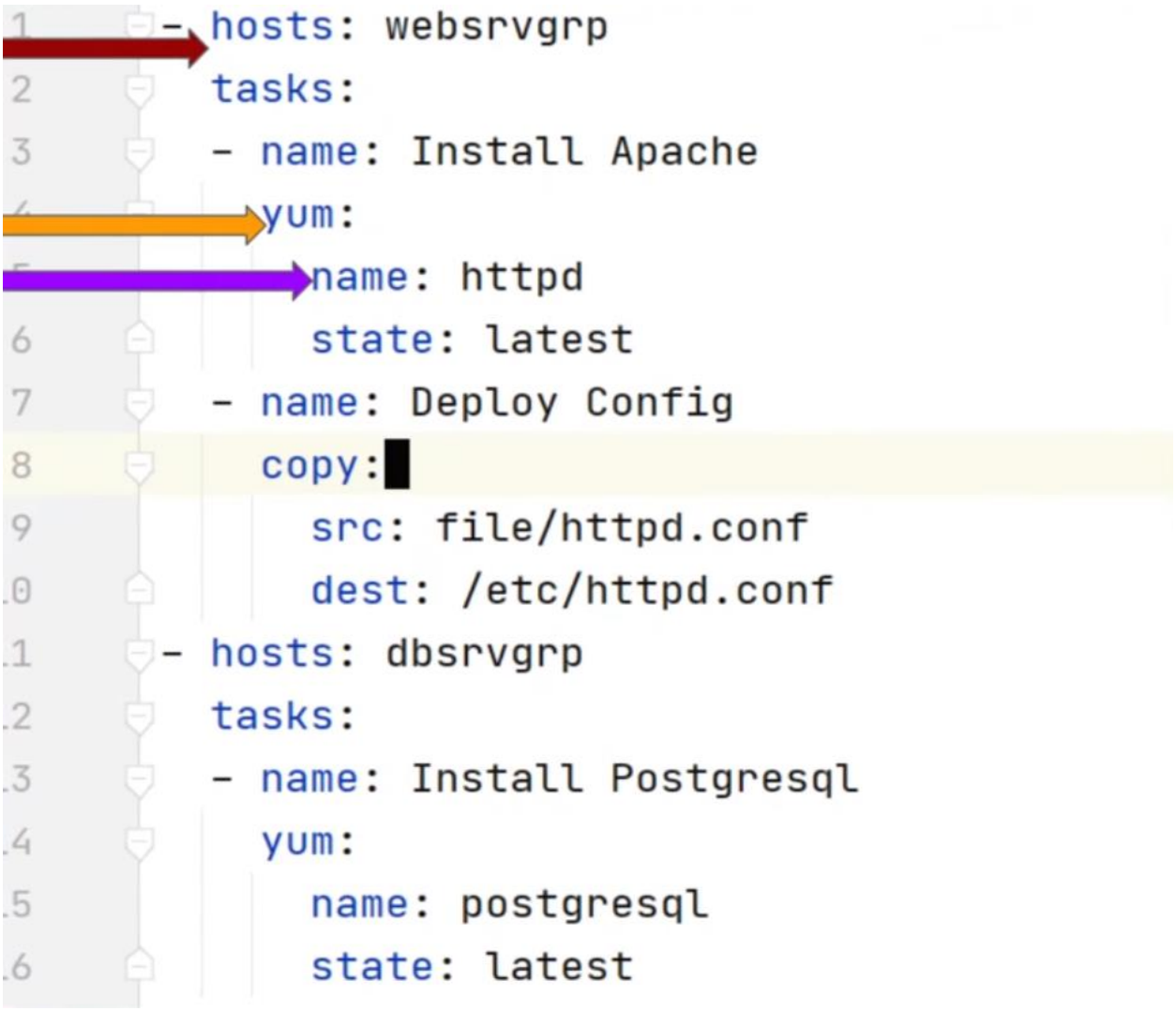


Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

In a Playbook, we can have multiple plays, and for each play, multiple tasks can be defined.

In the example below, we have a Playbook that contains two plays. The first one includes the **webserver** group, and the second includes the **dbserver** group, each specified with `- hosts`.



Each play in the example above contains a series of tasks. In the first play, which is related to the **webserver** group, there are two tasks.

Each task has a name specified by `name`. For example, the name of the first task is **Install Apache**, and the module used in this task is **yum**. Two options are defined for this module: **name** and **state**. This Playbook includes a three-level block structure. The first level is the **global area**, which contains **hosts** and **tasks**.

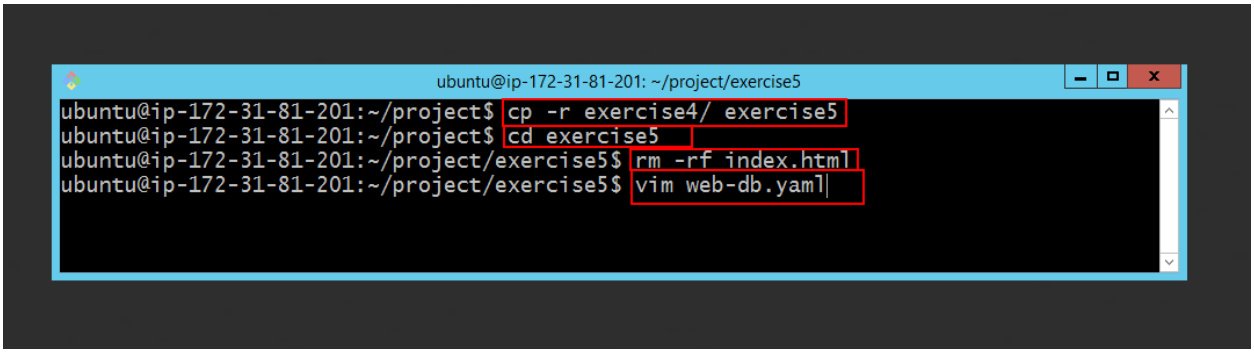
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

The second level is the **module name**, and the third level includes the **module options**.

How to Create a Playbook

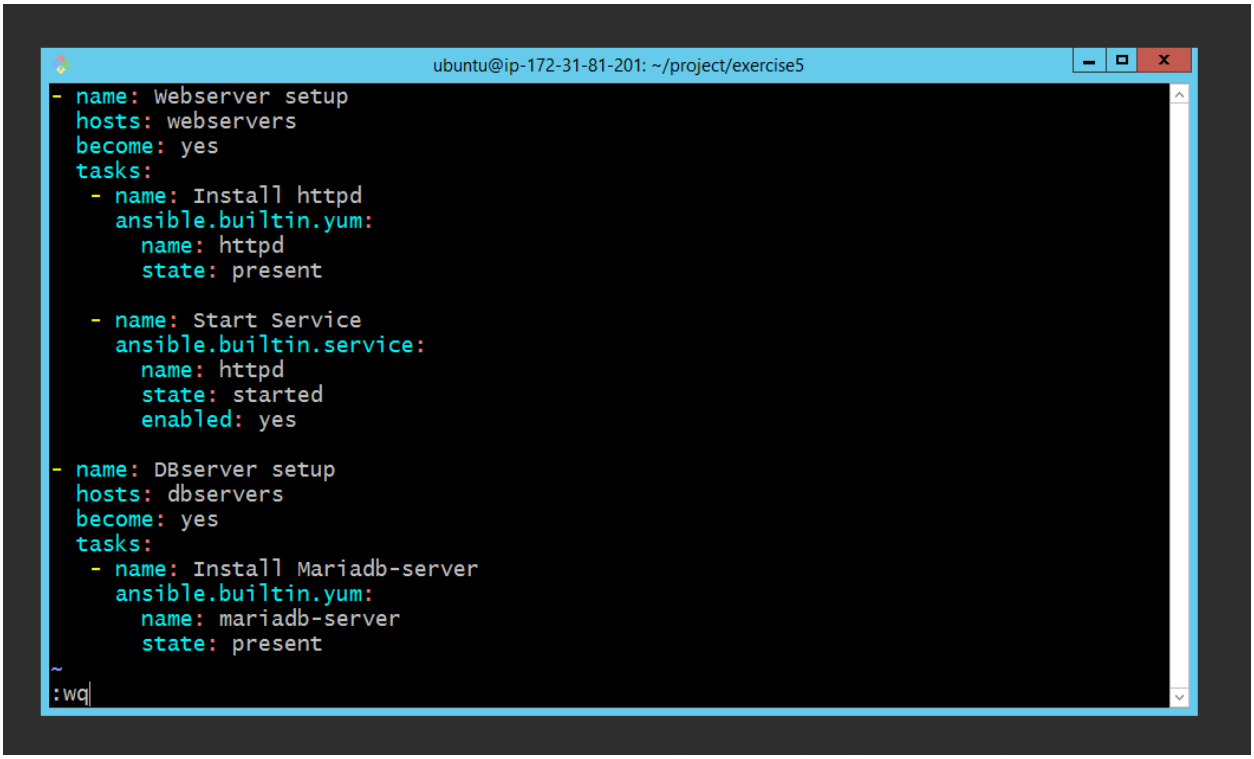
In the project directory, create a file with the `.yaml` extension. Just like the inventory file, you can choose any name for it.



Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

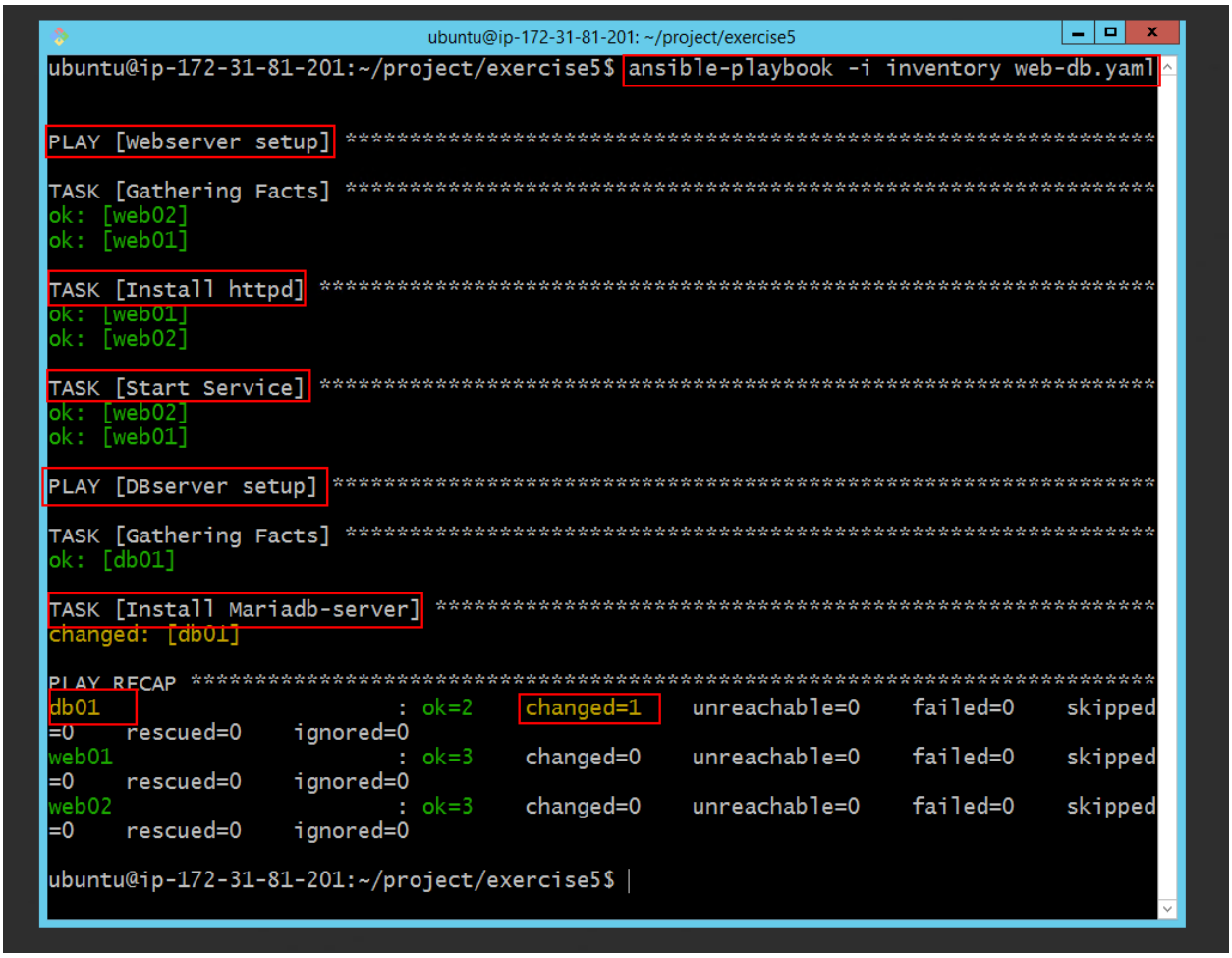
Inside the file, type the script below. In the first play, it runs two tasks on the web server group: installing the `httpd` service and starting and enabling it. In the second play, it installs the database service on the `dbserver` group.



Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

Use the following command to run the Playbook.

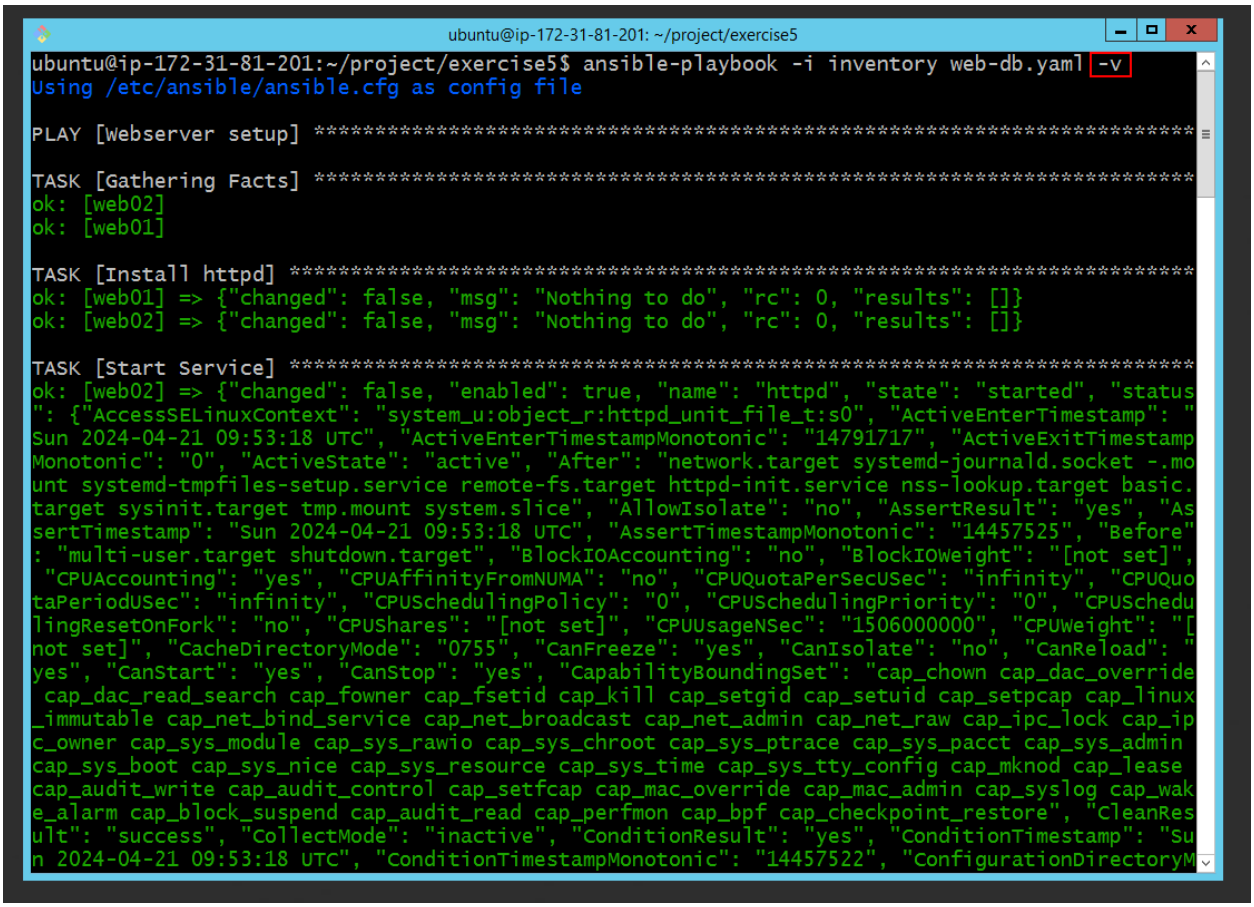


In the output of the above command, there is a default task called **Gathering Facts**, which uses a module named **setup** to collect some information about the target.

Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

If you add the `-v` switch at the end of the `ansible-playbook` command, it will provide detailed information during script execution, which is useful for troubleshooting.

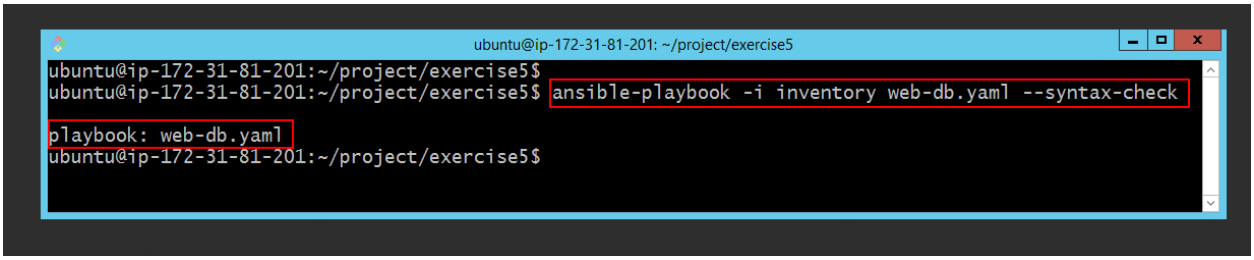


Instructor: Fariborz Fallahzadeh

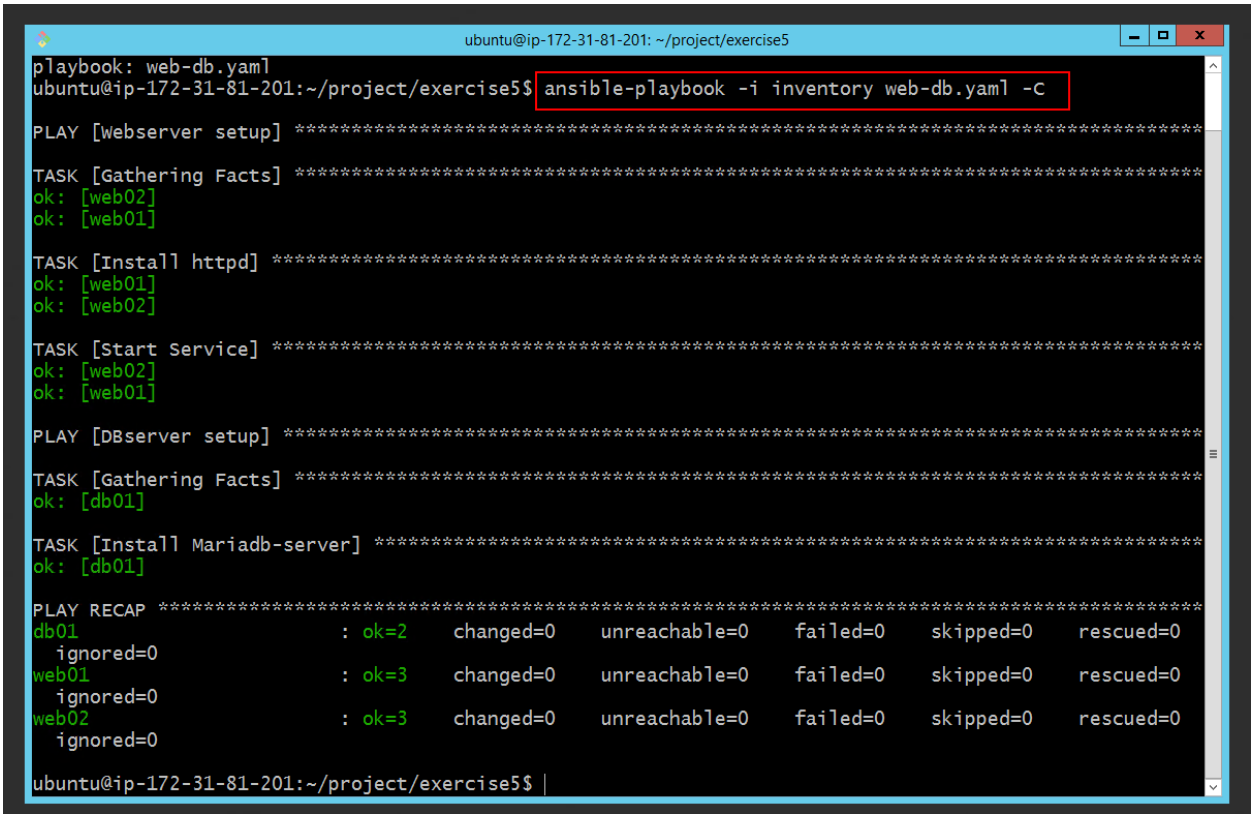
Email Address: fariborz.fallahzadeh@gmail.com

The `-v` switch stands for **verbose**, and you can use up to four `v`s in the command. The more `v`s you use, the more detailed the output will be.

To check the syntax of the YAML file, you can use the following switch:



One of the switches you can use with the `ansible-playbook` command is the `-C` switch, which tests the Playbook on the targets before actual execution. If there are any issues, it reports them in the output. This is known as a **dry run**, and it is recommended to use it in production environments.



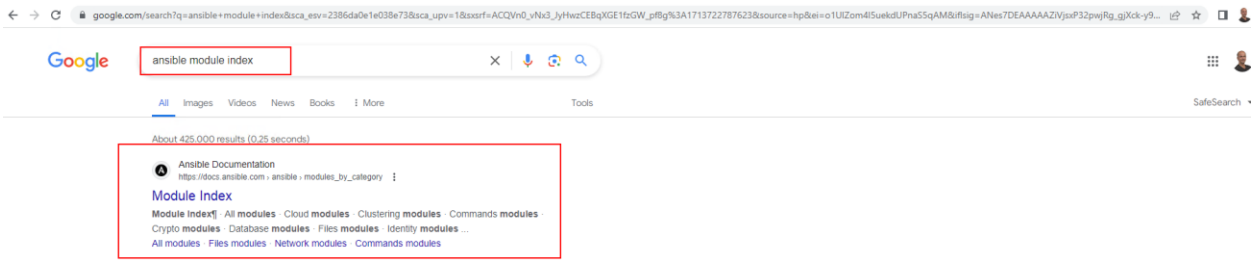
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

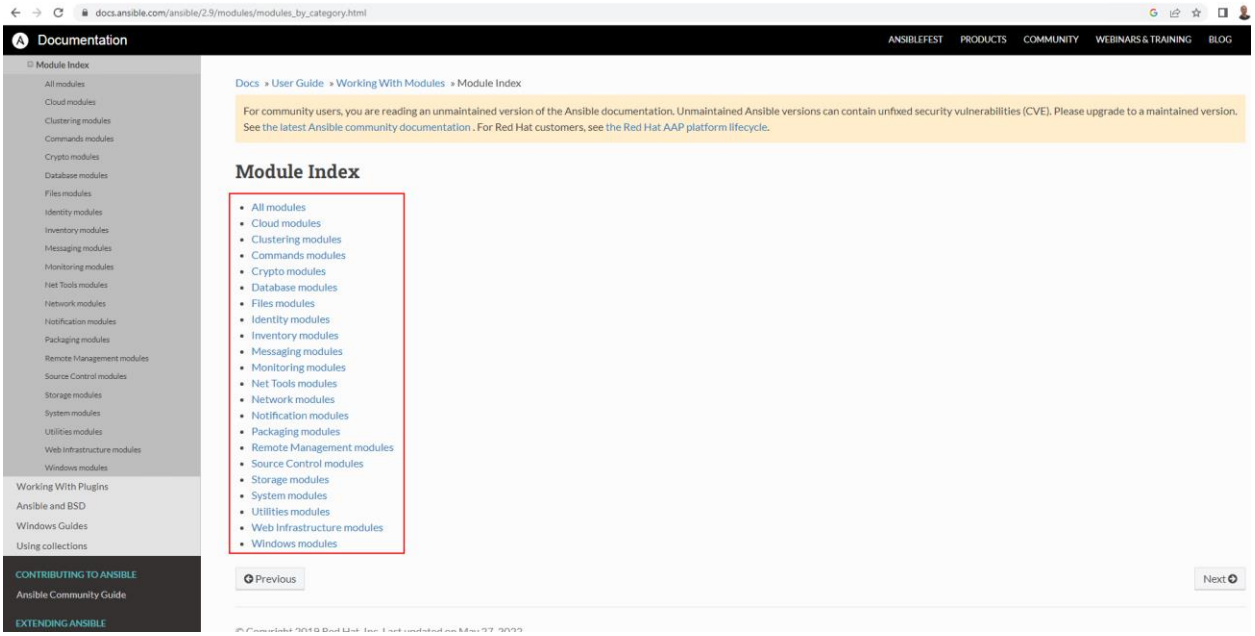
How to Find and Use Modules

One of the tasks a DevOps specialist must be able to perform is finding and using modules based on the tasks in a Playbook.

To find modules, simply search the following phrase on Google:



In this section, we can view all Ansible modules categorized and organized.

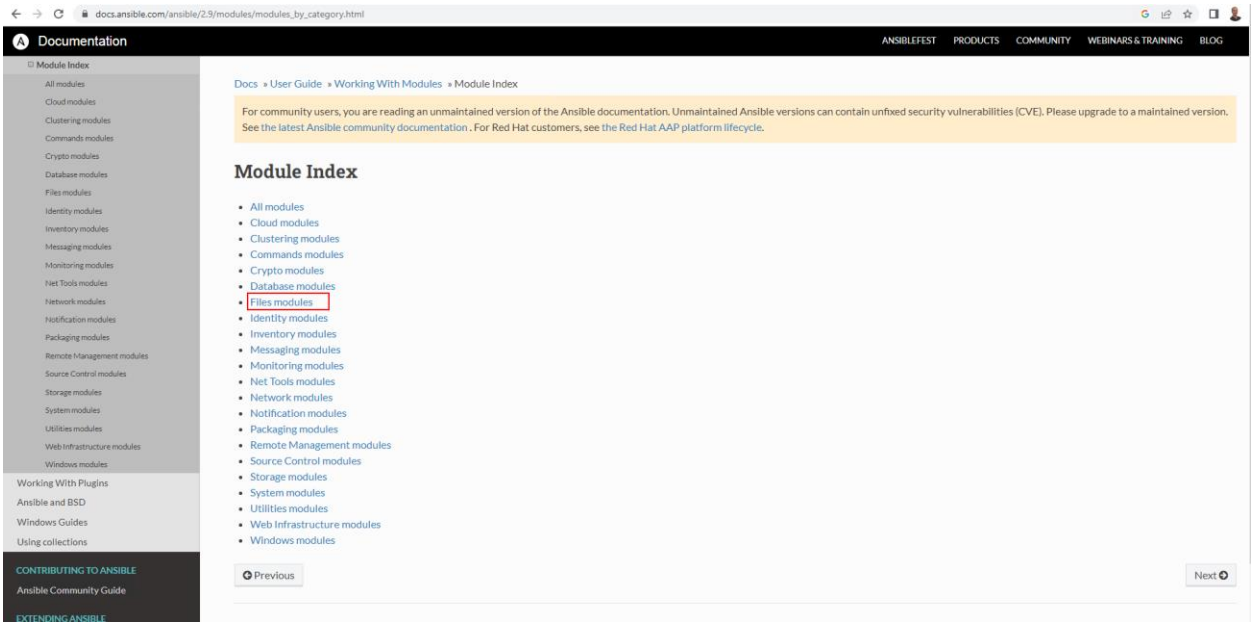


Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

For example, suppose we want to copy a file from the control machine to the web server machine, and on the other hand, we want to create a database and a database user on the DB server machine.

To copy a file, we can use the **Files** module, so we click on this module.

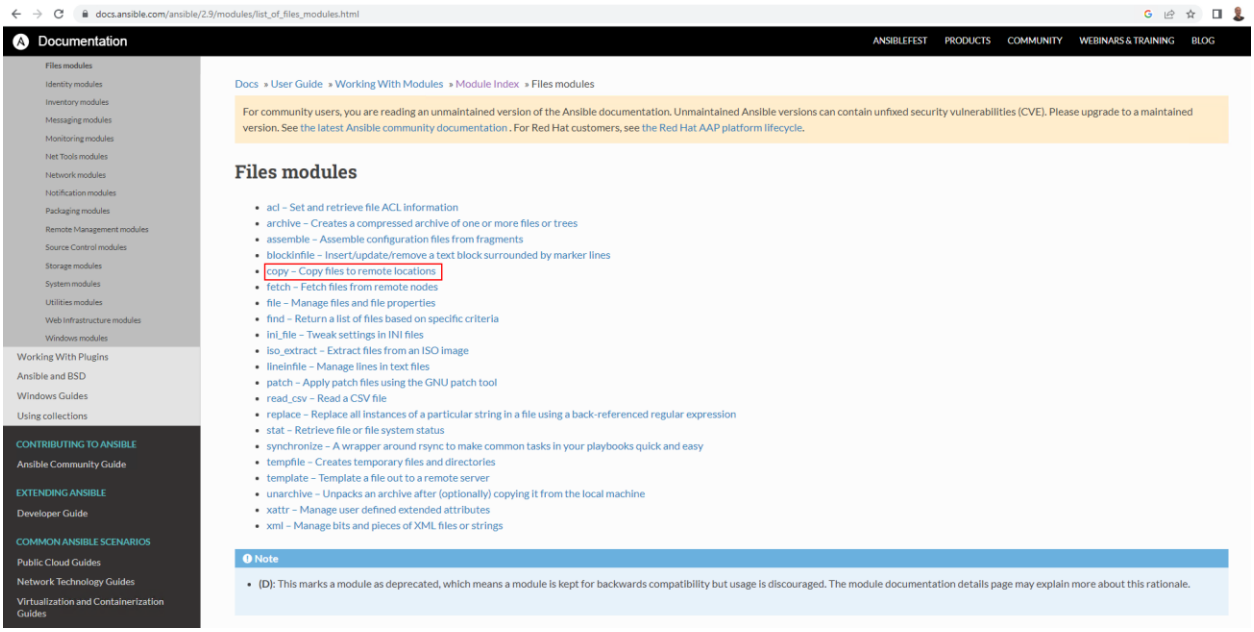


Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

As shown in the image below, there are many file-related modules available in this section.

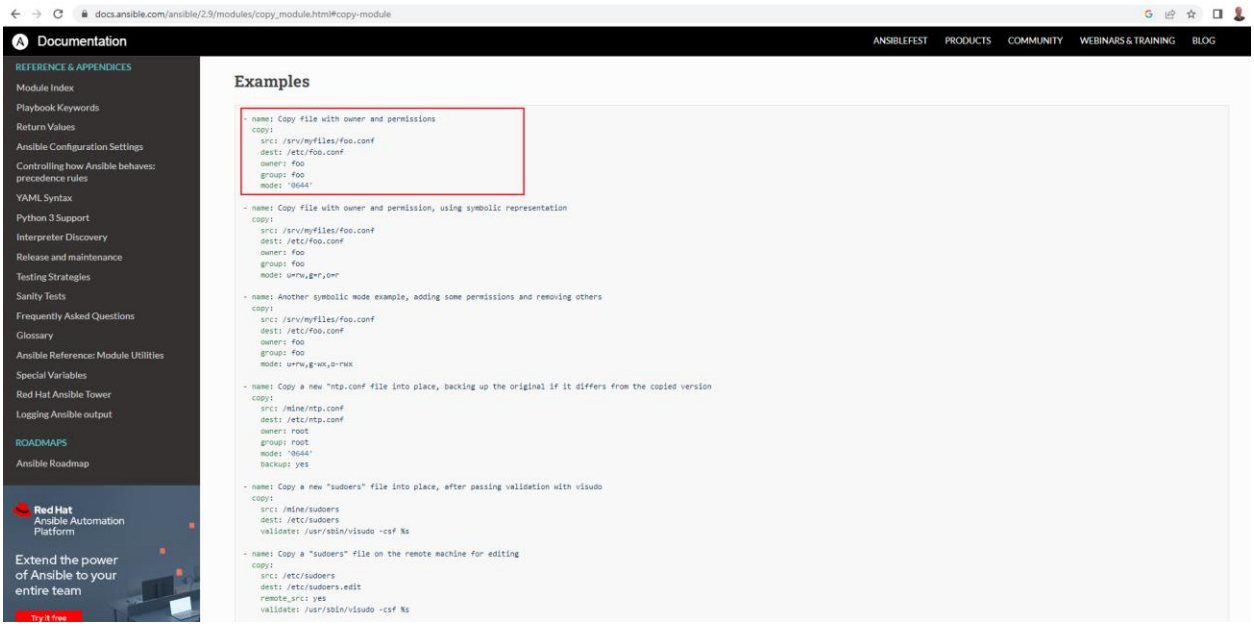
Since we want to copy a file from the control machine to the web server, we click on the following module: **copy**.



Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

After opening the module, as you can see, there are various examples of file copying available.



At this stage, create a folder for the project and then open the Playbook file using the VIM editor.

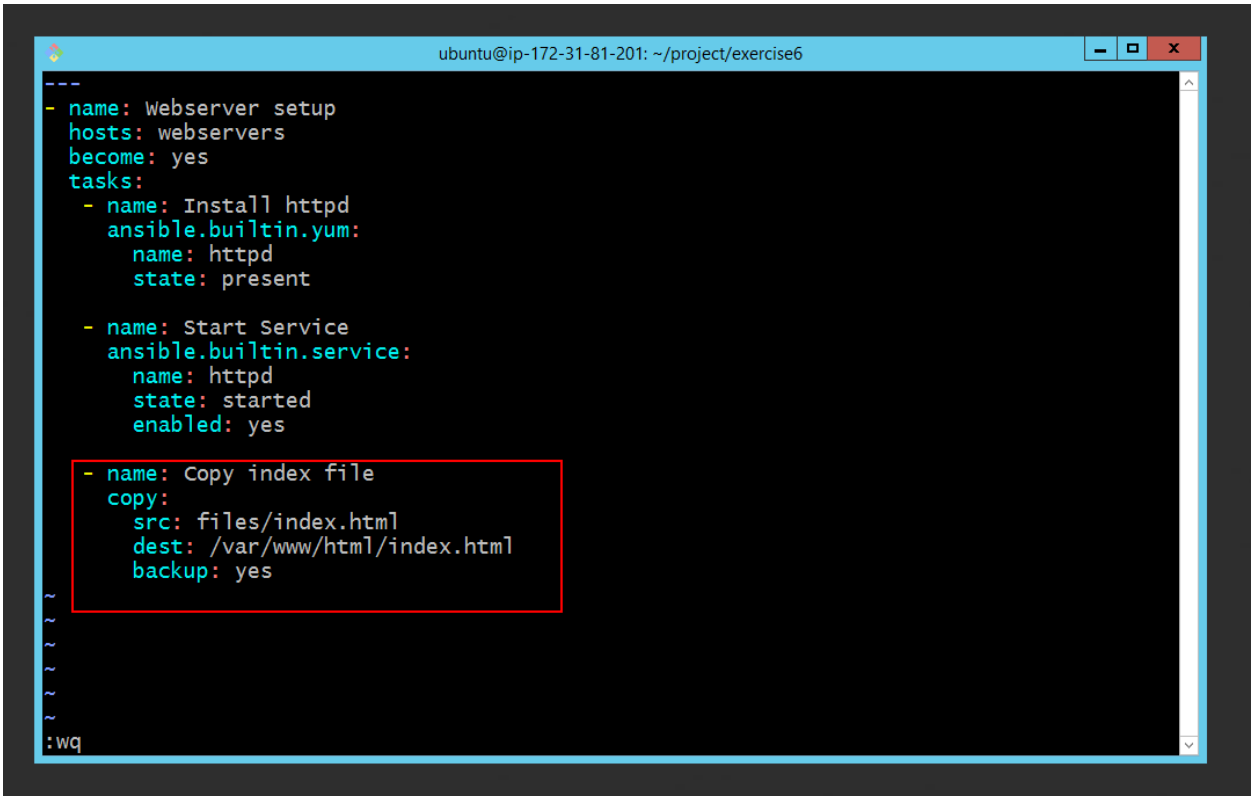
In the Playbook file, create a new task to copy the `index.html` file from the control machine to the web server's directory. Use the **backup** option as well, which will first create a backup of the existing file at the destination if it exists, and then copy the new file.

Some options of a module are mandatory, and next to them, the word **Required** is listed, meaning they must be used. Other options are **Optional**, and you can use them if needed.

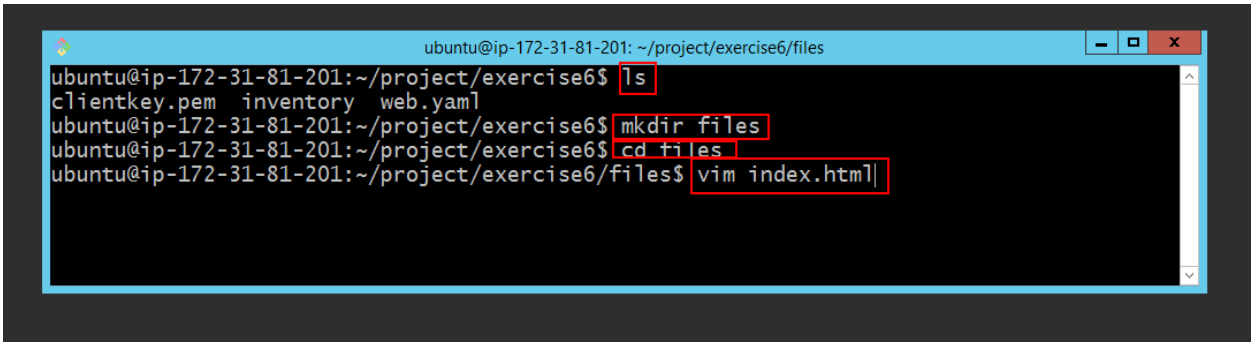
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

Add the following task to the Playbook and then save the file.



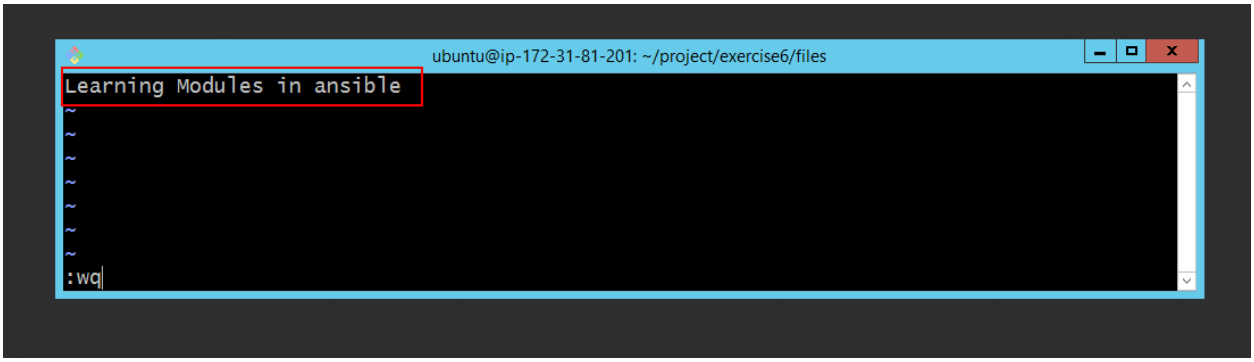
Now, since the `index.html` file needs to be copied from the control machine, you should copy the **files** folder and the `index.html` file to the control machine.



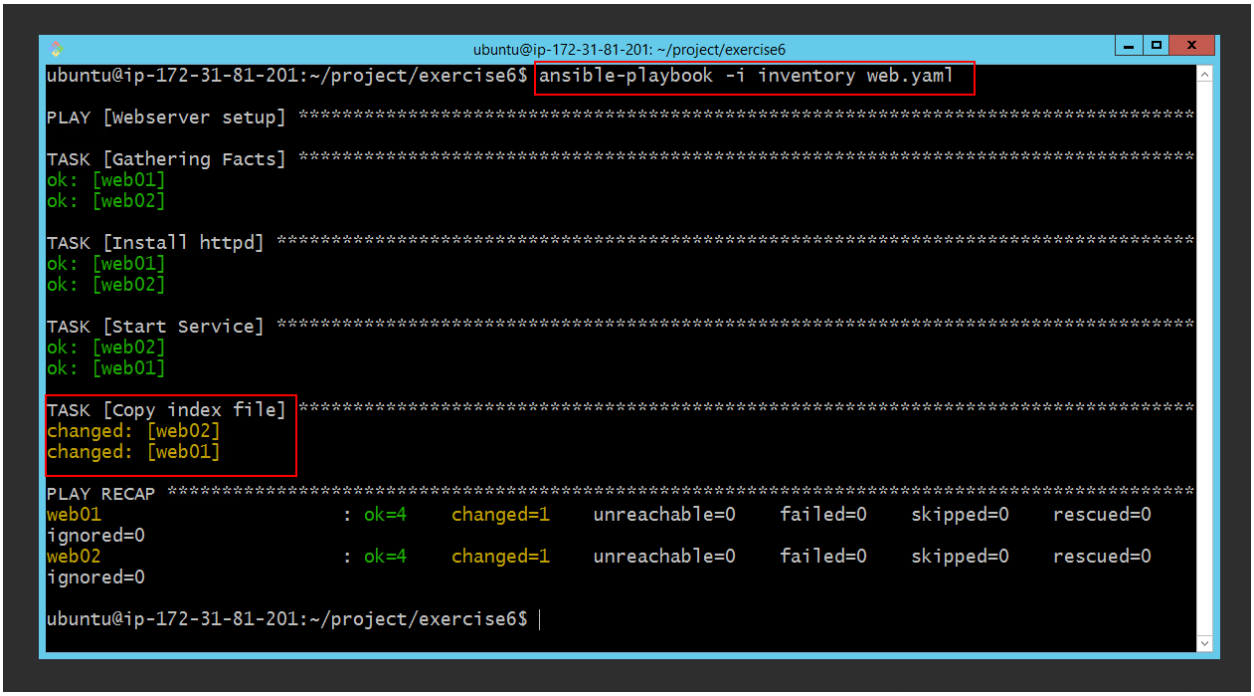
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

At this stage, create some test content inside the `index.html` file and then save the file.



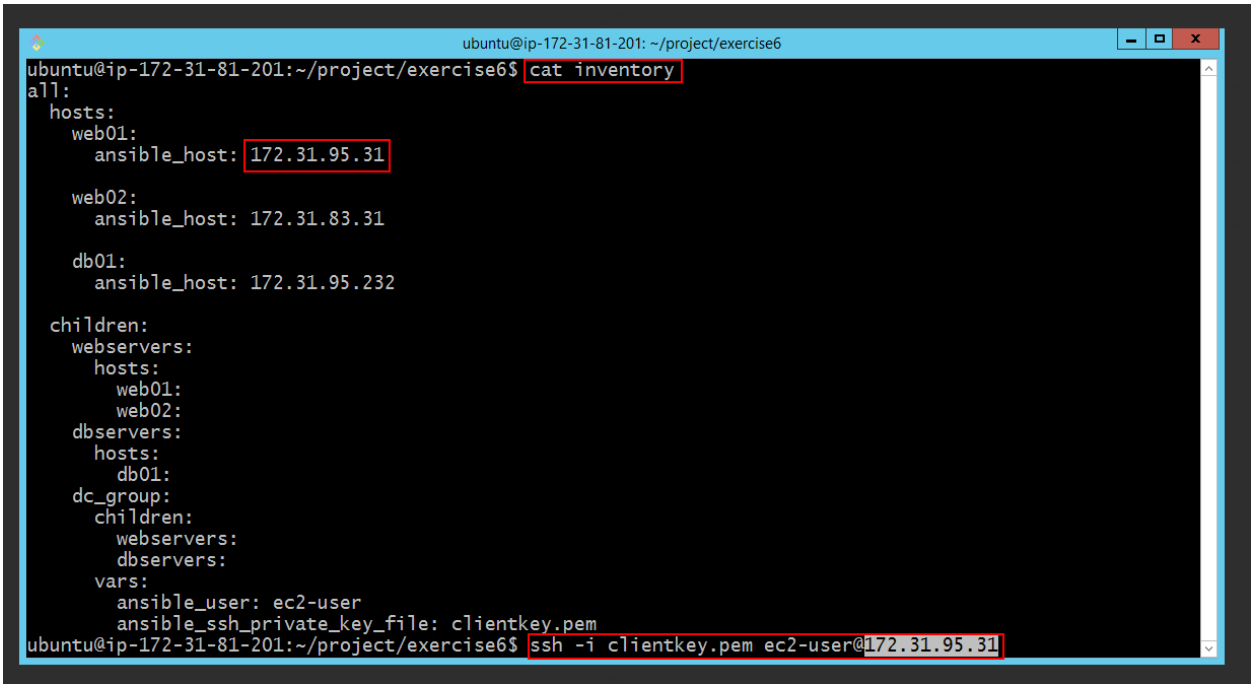
At this stage, use the following command to test the Playbook.



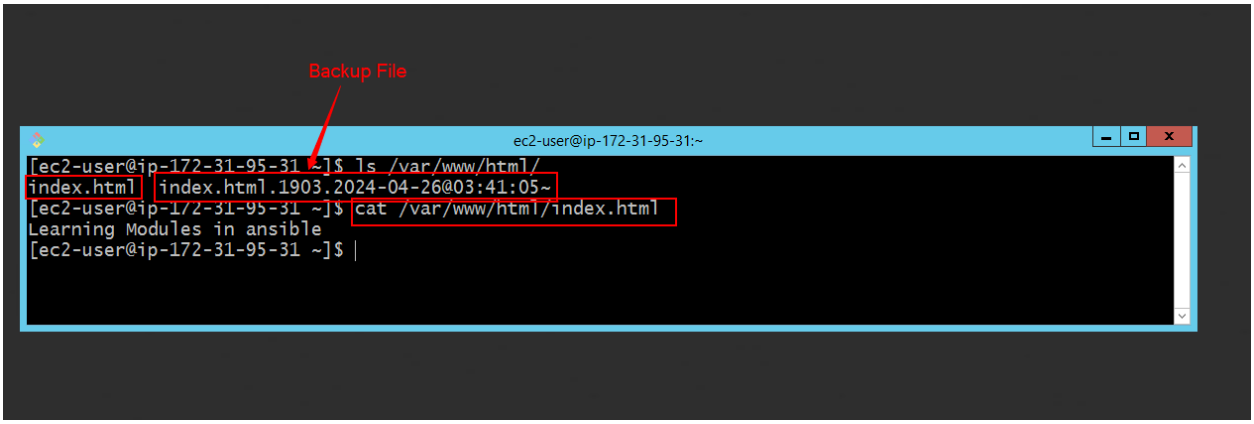
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

At this stage, to ensure that the file has been copied to the web server, SSH into the web server.



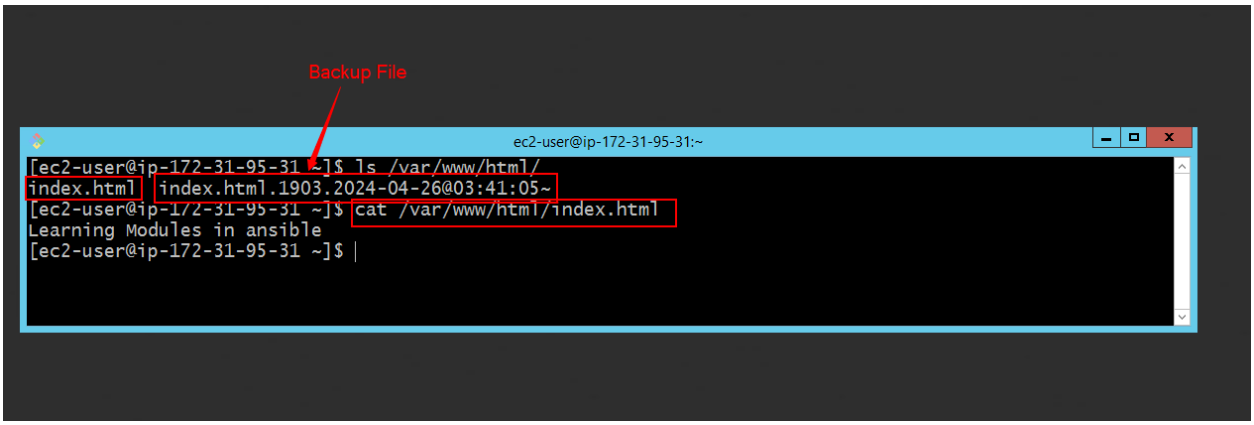
As shown in the image below, the file has been correctly placed in the web server's directory. Since the file already existed at this path, Ansible first created a backup of the existing file and then copied the new file.



Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

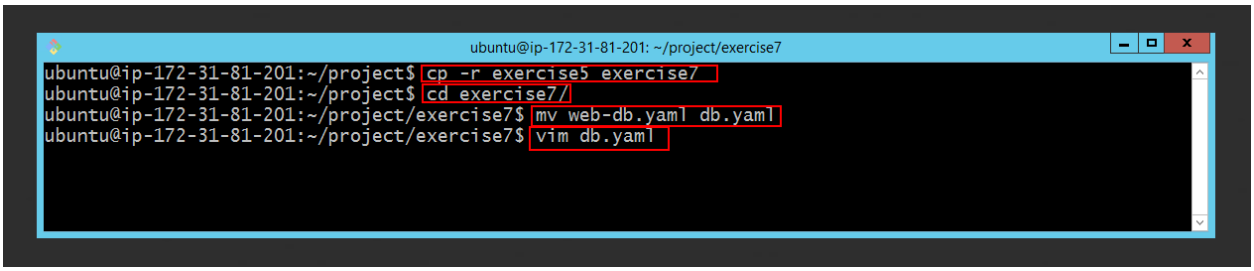
At this stage, use the `exit` command to exit the web server environment and return to the Control Machine terminal.



Note:

Some modules require dependencies.

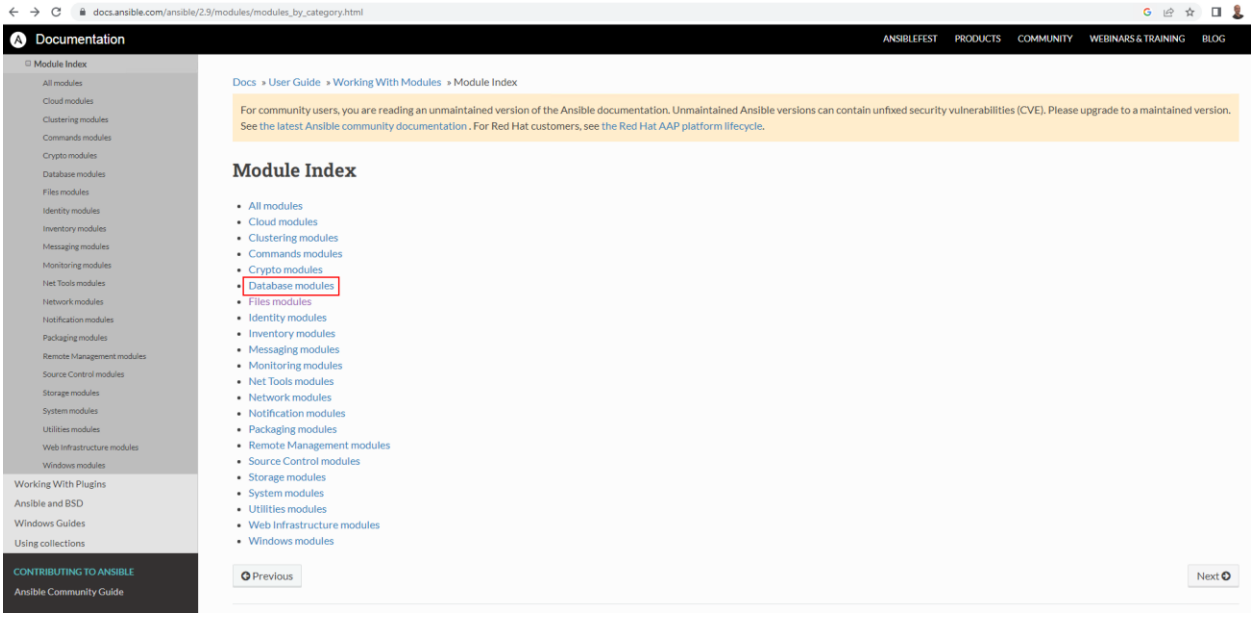
To get familiar with these dependencies, we will create a new project in this stage. The goal is to use a module to install a database and then start its service. After that, we will create a database and a database user using a Playbook.



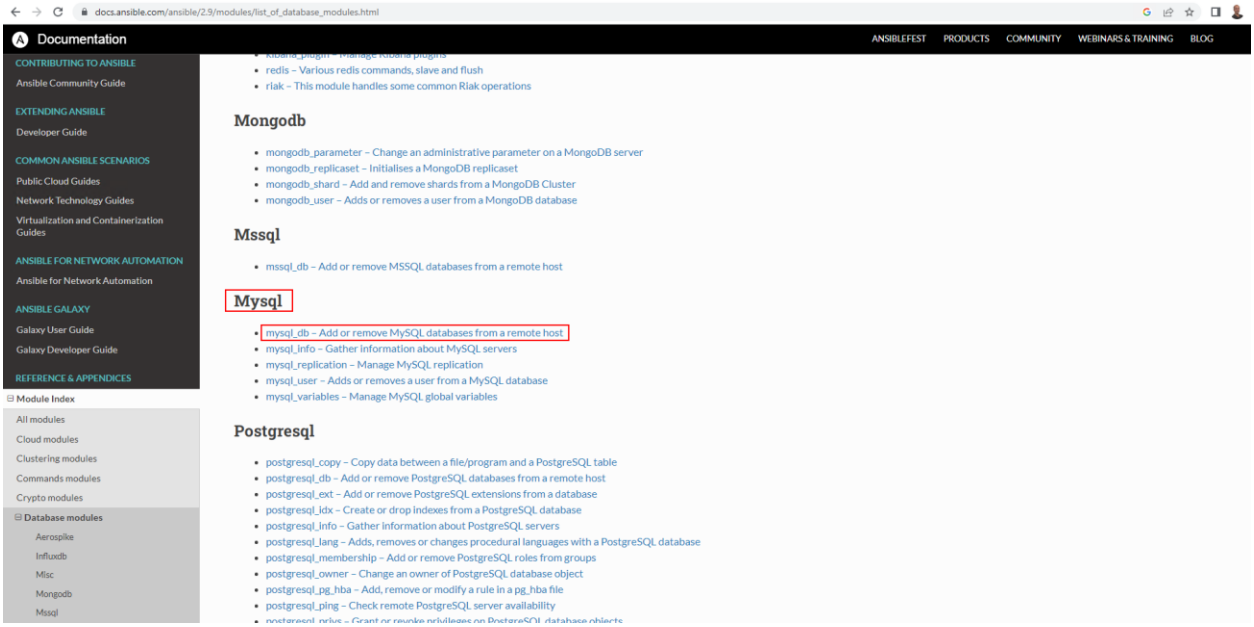
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

To work with databases, click on the **Database Module** section.



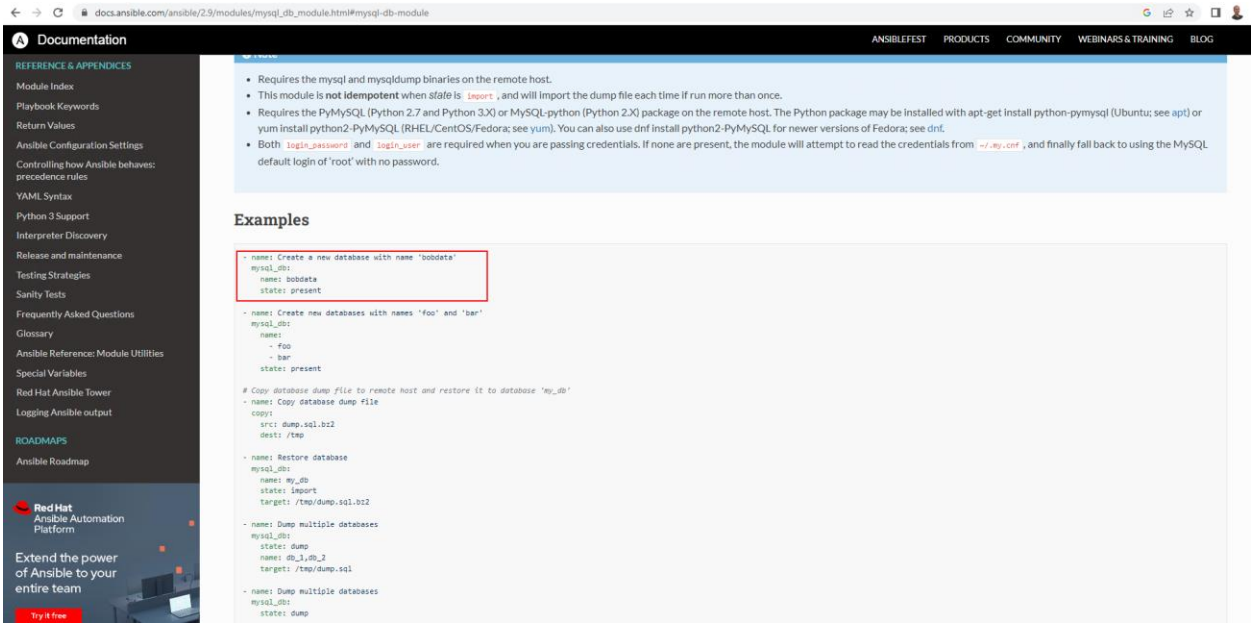
Since we intend to create a database on MySQL, click on the following module: **mysql_db**.



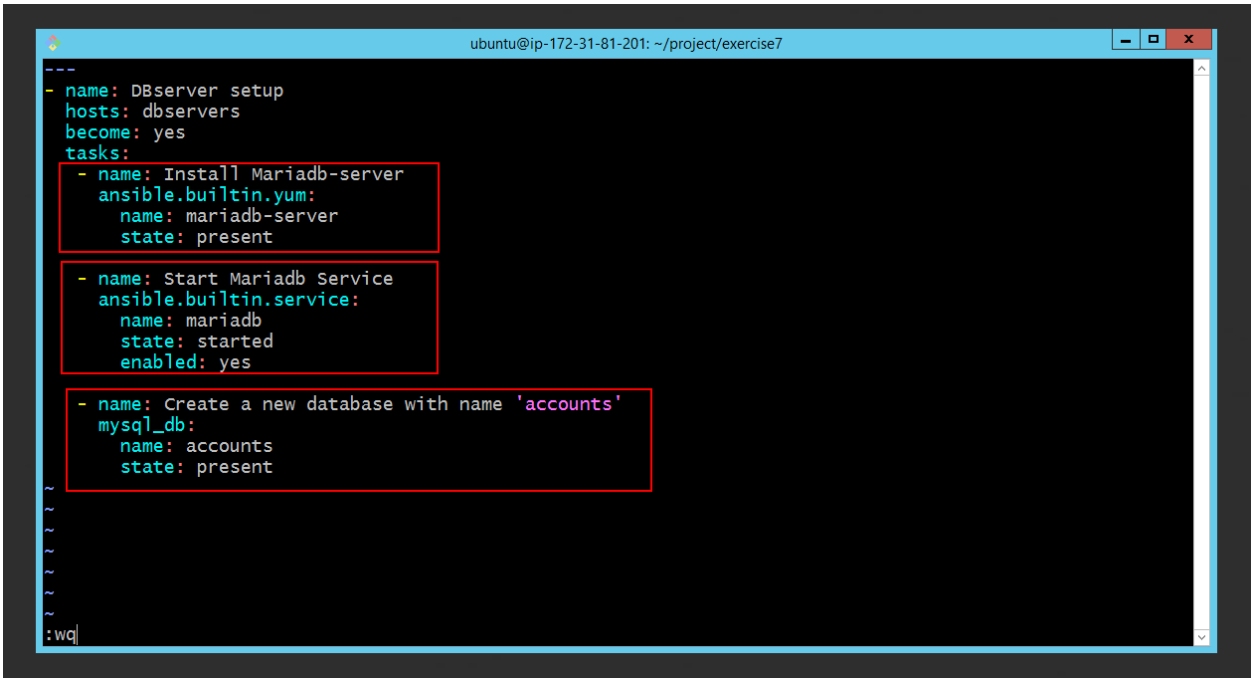
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

In the **Example** section, you can see examples of how to create a database.



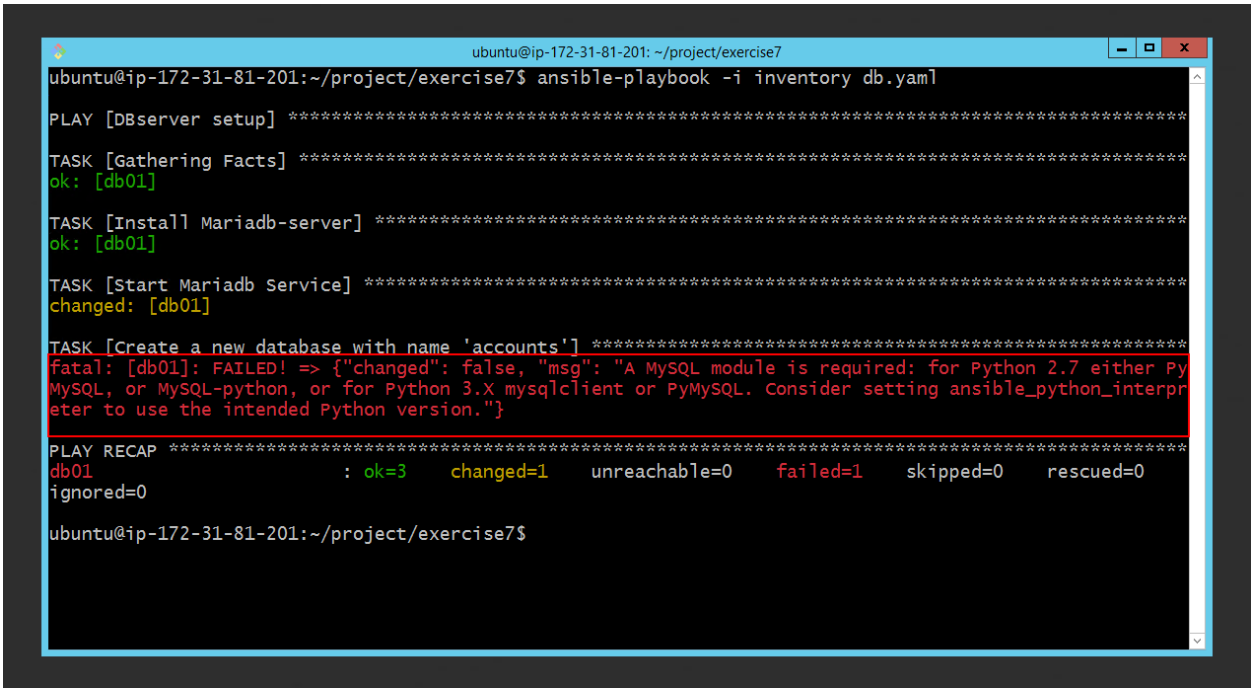
As seen in this scenario, to create a database, you first need to ensure that the database is installed using a module. Then, you need to use another module to start the database service, and finally, the database can be created.



Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

At this stage, we run the Playbook. As you can see, during execution, it throws an error. The reason for this is that most Ansible modules require certain dependencies before they can be used, and most of them depend on Python.



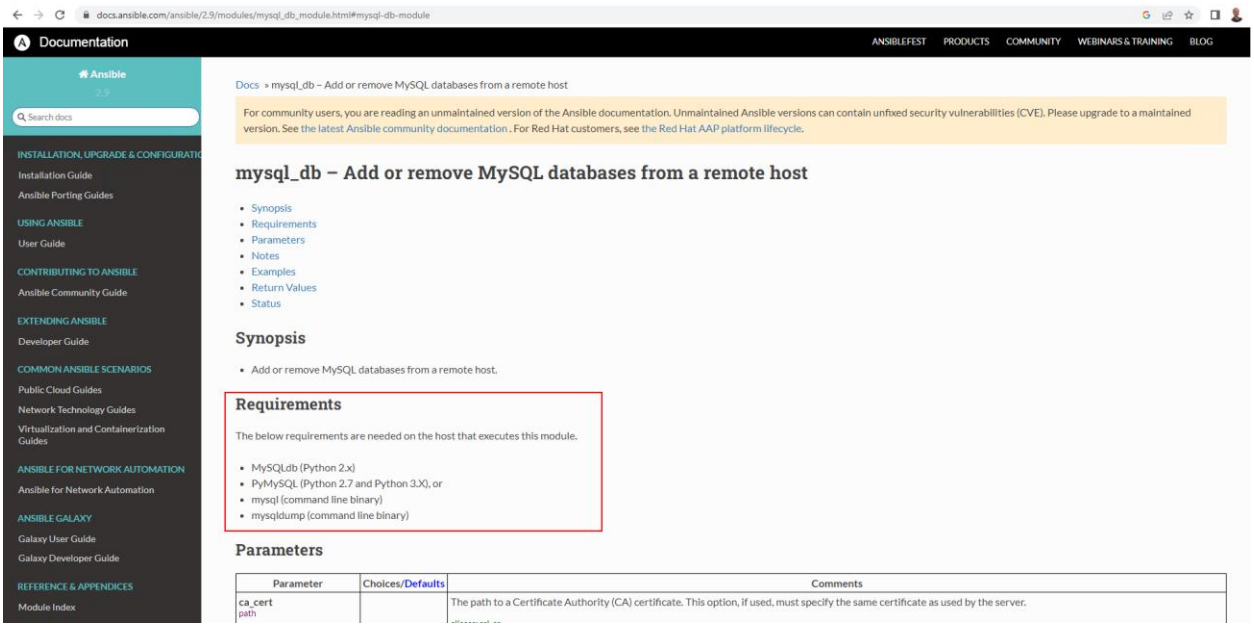
Now, why did this happen during the execution of the Playbook?

We know that Ansible creates a Python script, which is uploaded to the DB server and then executed by the DB server. Some modules require specific dependencies before execution. For example, the MySQL module requires Python 2.7, and Python 2.7 needs the **PyMySQL** or **MySQL-python** module to connect to the database.

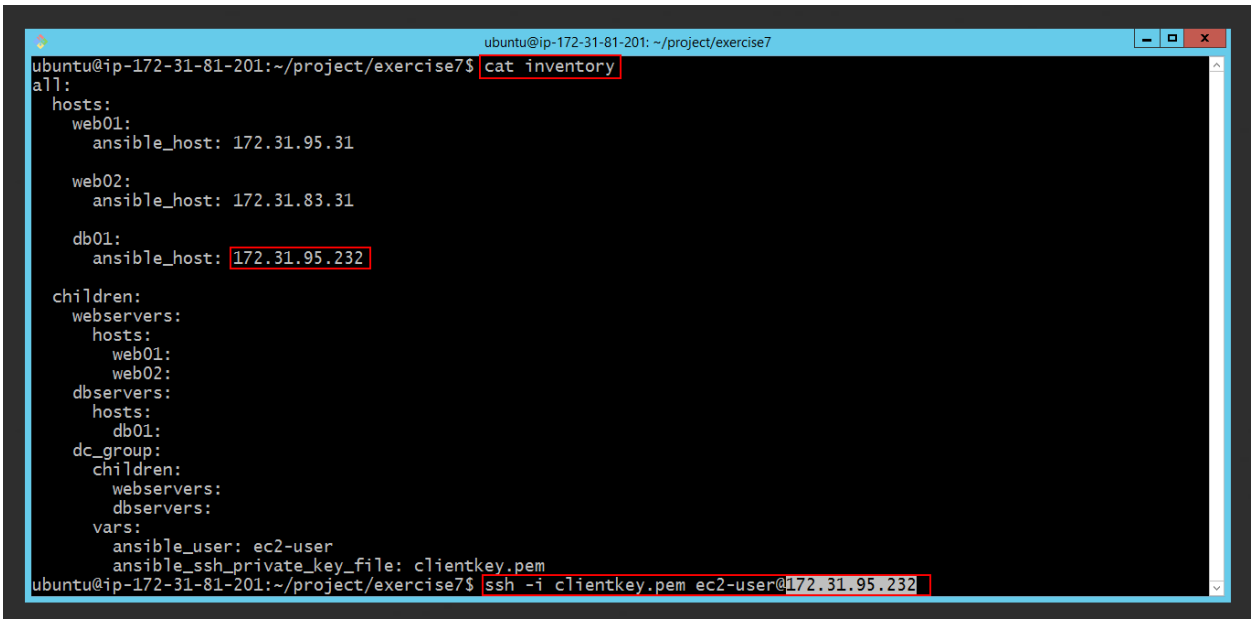
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

Before using a module in Ansible, always check the **Requirements** section of the module you want to use.



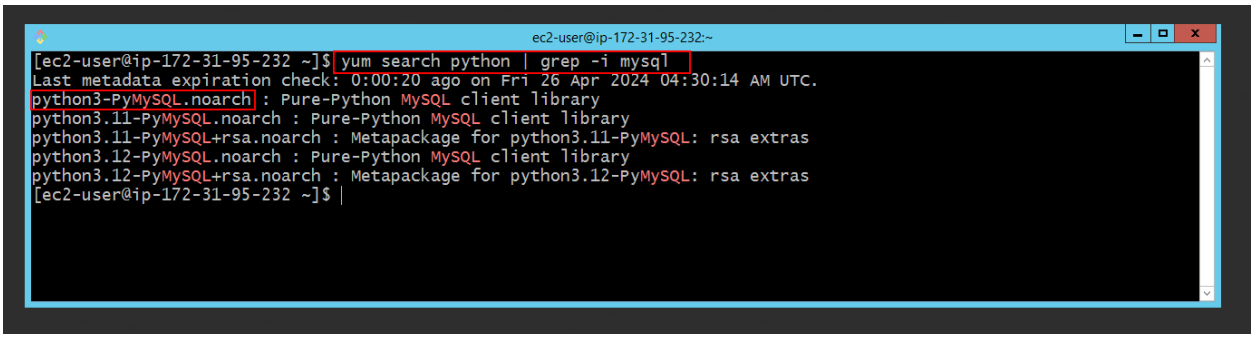
To install the correct module on the DB server, first SSH into the DB server of the scenario.



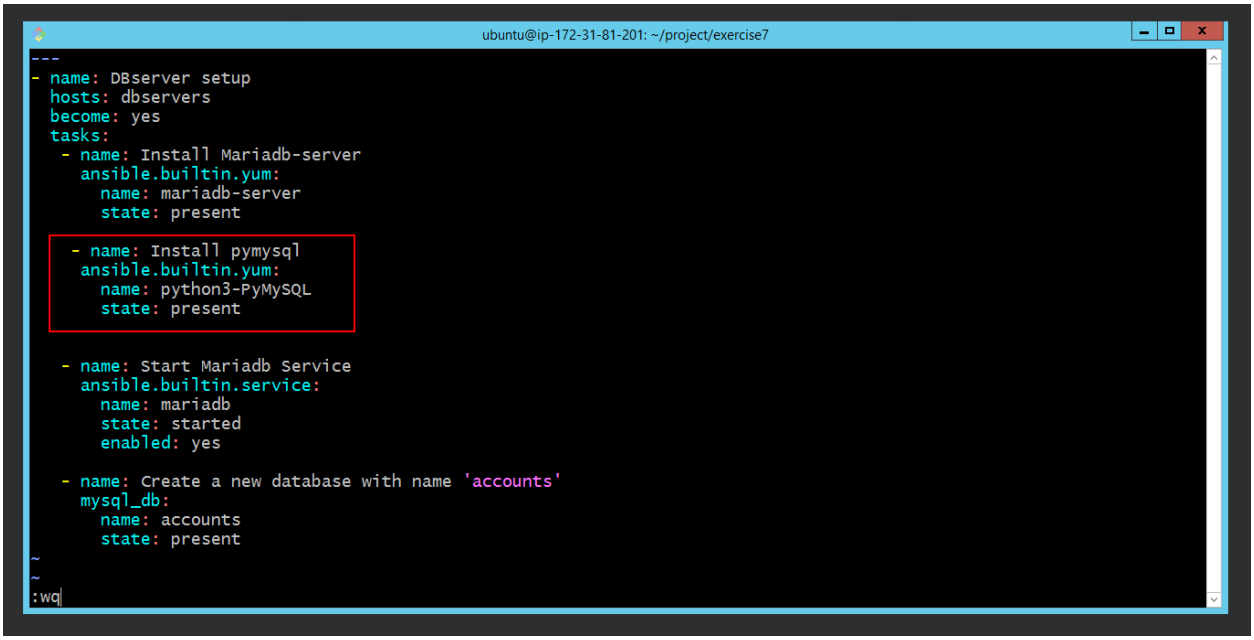
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

At this stage, use the following command to search for the required dependency to connect to the MySQL database in the Linux repository.



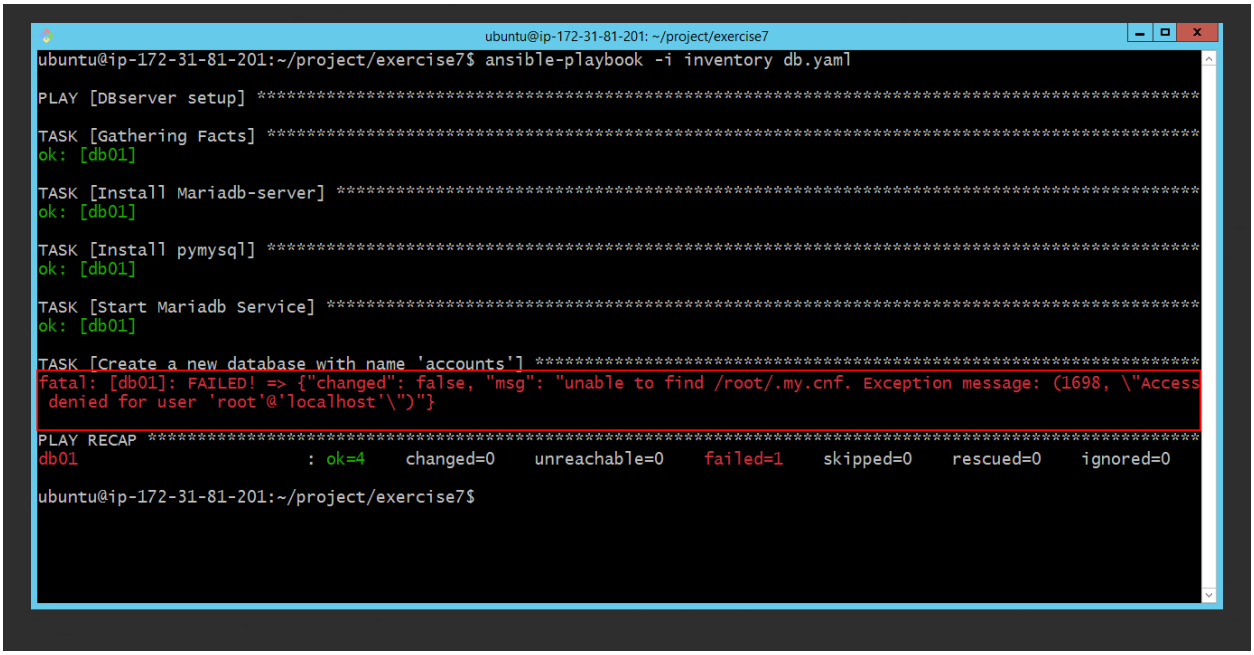
Then, in the Playbook file, use the **yum** module to add the Python package, so it can be installed on the DB server.



Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

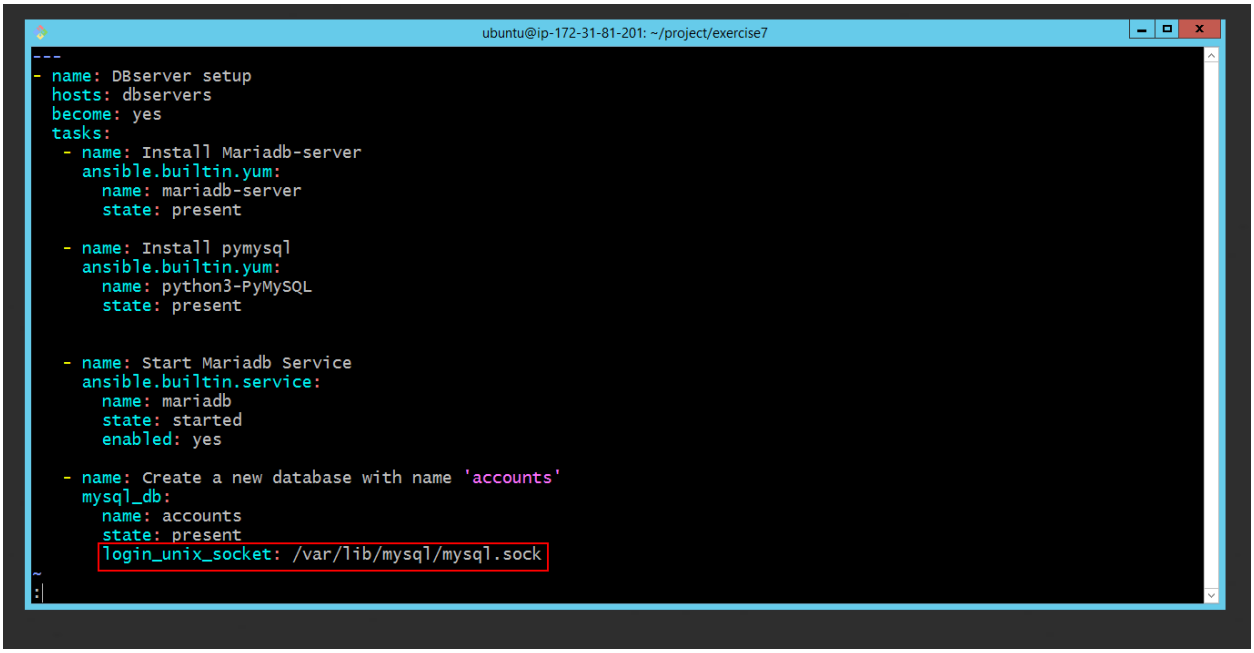
We test the Playbook again.



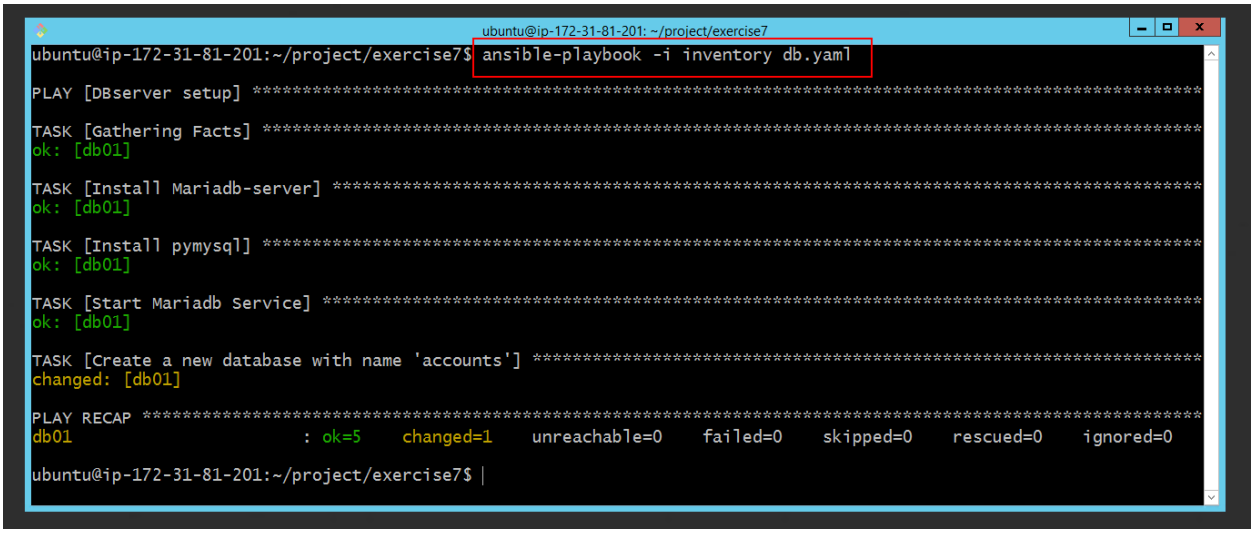
Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

As shown in the image above, after executing the Playbook, we encountered a new error. The reason for this is that the file socket path in the CentOS operating system is located elsewhere, and this needs to be corrected in the Playbook file.

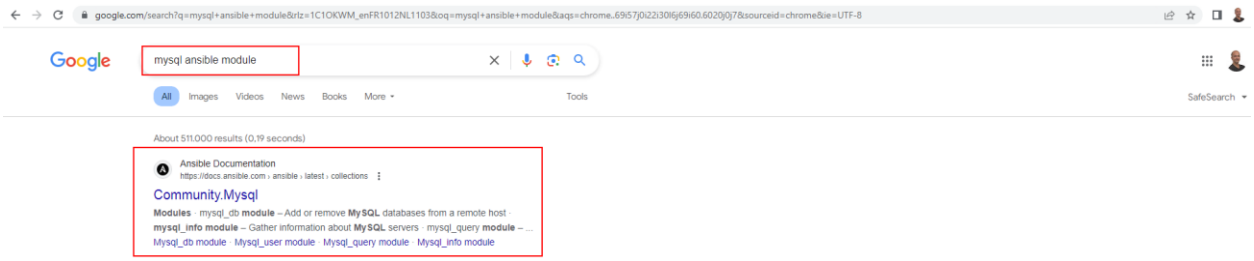


We run the Playbook again. As shown in the image below, this time it runs without any errors.

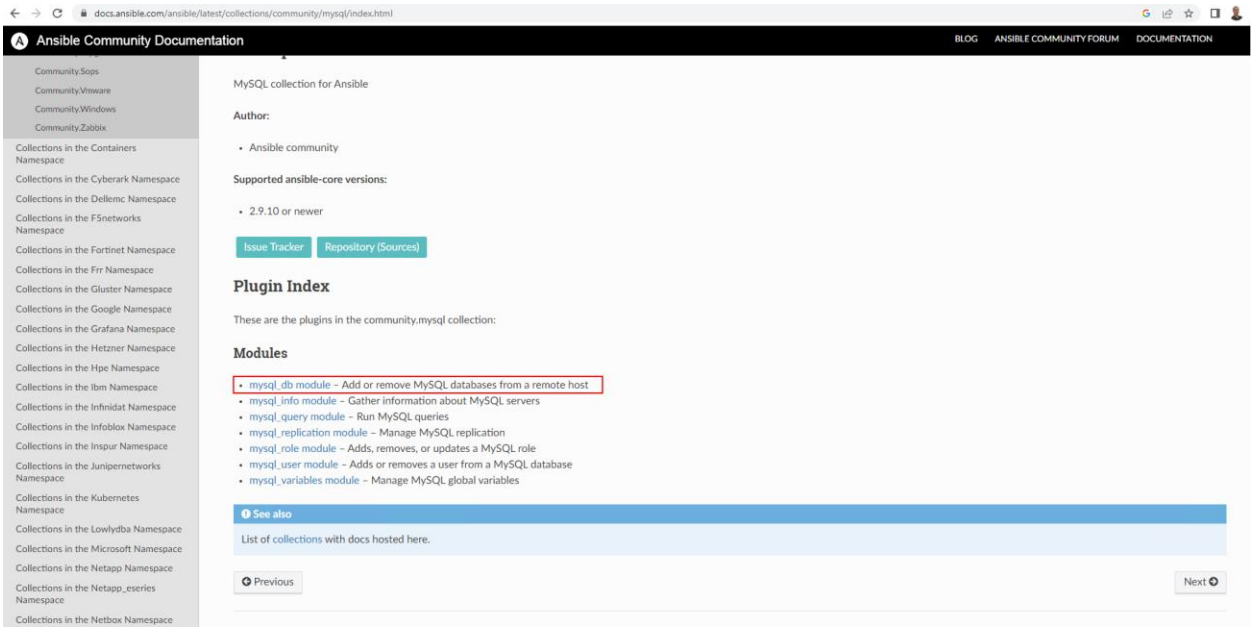


To find errors related to a module and how to resolve them, we can refer to the community discussions or forums related to that module.

For example, simply search for the following phrase on Google and click on the **Community MySQL** link.



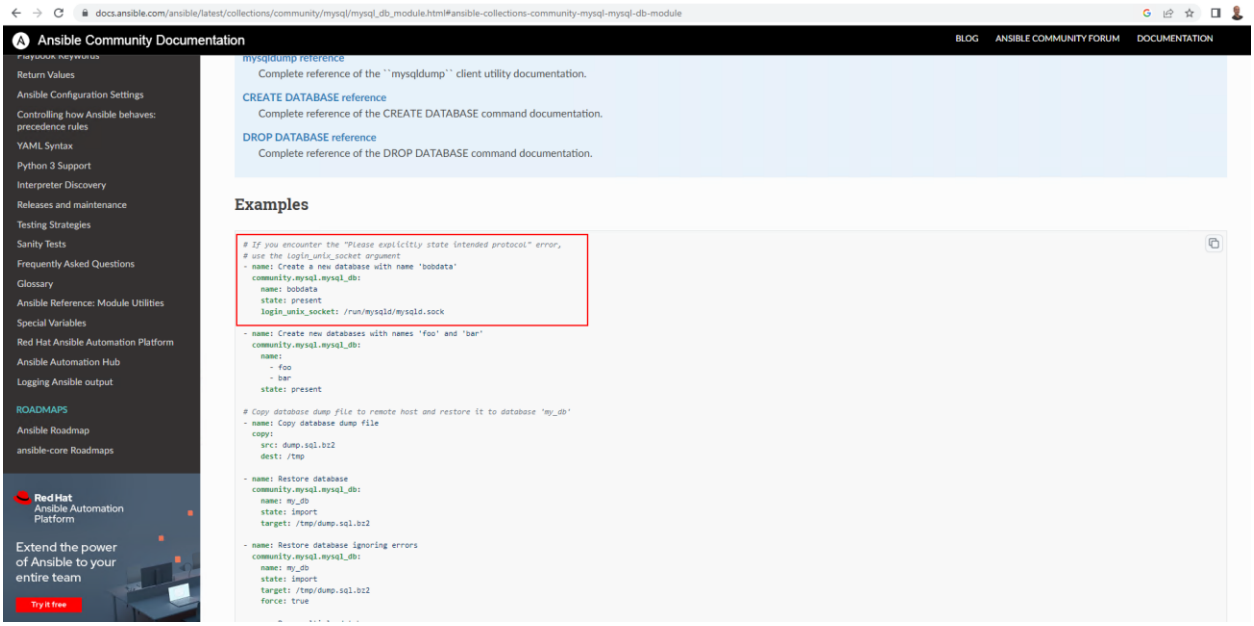
Then, click on the link for the specific module you are interested in.



Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com

As seen in the **Community Module** section, here you can find examples of using the module without errors, exactly like the one we created in the Playbook.



Instructor: Fariborz Fallahzadeh

Email Address: fariborz.fallahzadeh@gmail.com