

DEPLOY A WEBSITE ON AN APACHE WEB SERVER ON MULTIPLE EC2 INSTANCES USING ANSIBLE PLAYBOOK:

By P.Lokesh(Mech)

Why Ansible?

No matter your role, or what your automation goals are, Ansible can help you demonstrate value, connect teams, and deliver efficiencies for your organization. Ansible delivers simple IT automation that ends repetitive tasks and frees up DevOps teams for more strategic work.

Agenda

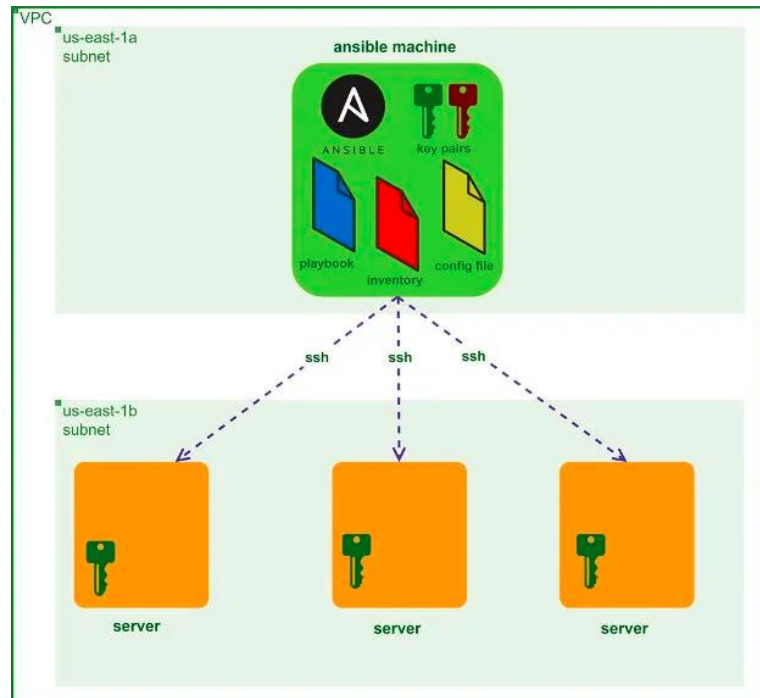
In this blog, we will learn how to deploy a website on multiple AWS EC2 Instances using Ansible Playbooks without the need to install the service on each server separately through the power of Automation which is the ultimate need for **DevOps** practice and 'Automate everything' is the key principle of **DevOps**.

Pre-requisites:

1. AWS account
2. Basic Knowledge of Ansible & Ansible playbooks
3. Basic Linux Knowledge

Project Architecture:

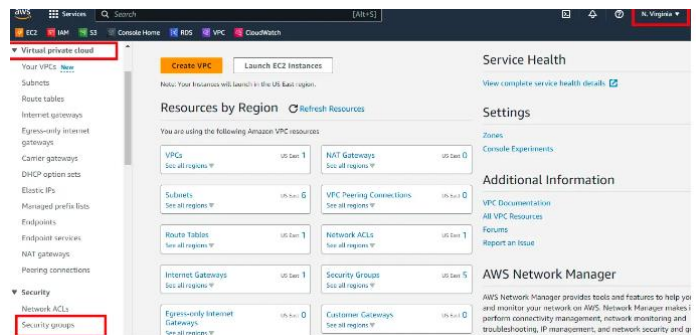
The below image shows the reference architecture of our project which we would follow in this blog.



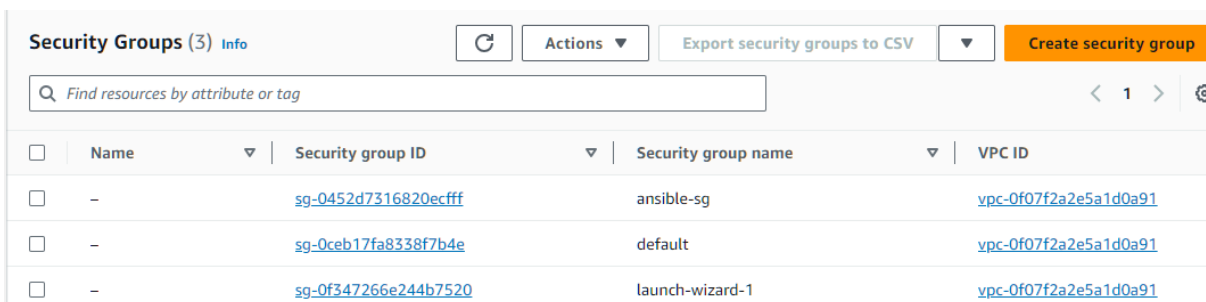
Steps Involved:

Step 1.

Create a couple of **Security Groups** under AWS VPC, first for our Ansible Control Master Server and then another for Ansible nodes. Log into the **AWS console**, choose your preferred region, head back to VPC, and select security groups on the left navigation bar as shown below:



Click on **Create security group**:



Provide the basic details of your security group such as name, and description, and

Basic details

Security group name Info

ansible-sg

Name cannot be edited after creation.

Description Info

ansible-sg

VPC Info

vpc-0f07f2a2e5a1d0a91

Under Inbound rules allow ssh from anywhere for this demonstration and click on **Create security group**.

Inbound rules Info

Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	Source <small>Info</small>	Description - optional <small>Info</small>
SSH	TCP	22	Any...	

0.0.0.0/0 X

Delet e

Add rule

Now let's create another security group that would be common for all three Ansible nodes on which we would deploy the web server.

Basic details

Security group name Info

ansible-nodes-sg

Name cannot be edited after creation.

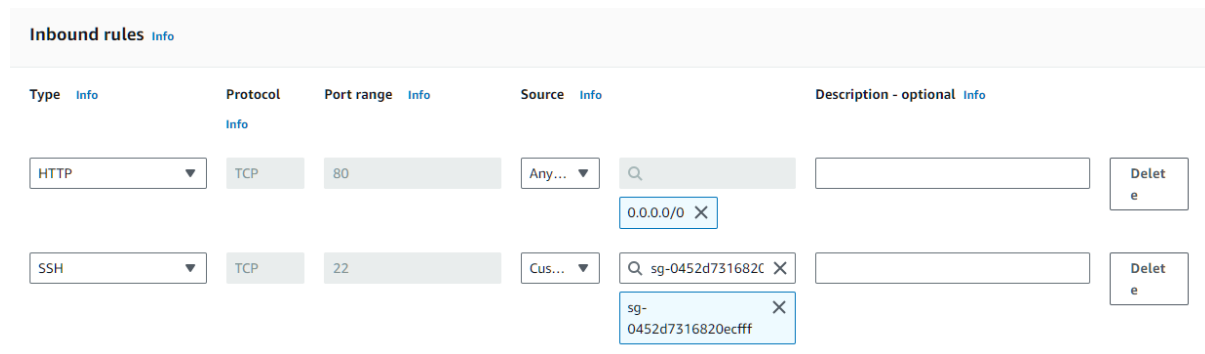
Description Info

ansible-nodes-sg

VPC Info

vpc-0f07f2a2e5a1d0a91

Under Inbound Rules allow HTTP from anywhere and allow SSH from the master server's security group only and click on **Create security group** as depicted below:



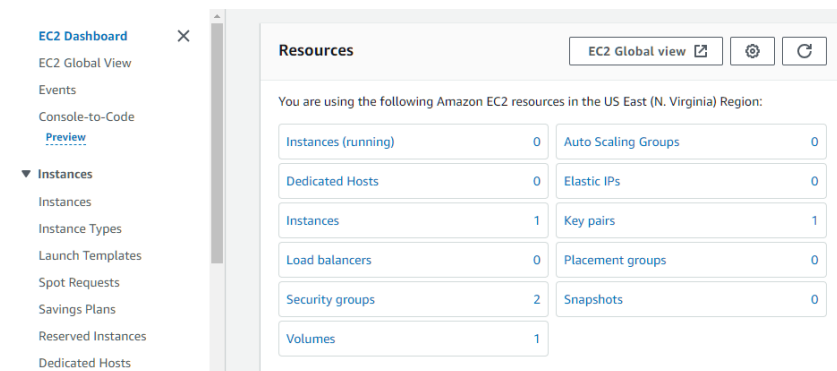
The screenshot shows the 'Inbound rules' configuration page in the AWS Management Console. It displays two rules: an HTTP rule allowing traffic from anywhere (0.0.0.0/0) on port 80, and an SSH rule allowing traffic from a specific security group (sg-0452d7316820ecff) on port 22. The SSH rule's source is currently set to 'Cus...' and a dropdown menu is open showing the selected security group ID.

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	Any... 0.0.0.0/0	
SSH	TCP	22	Cus... sg-0452d7316820ecff	

Step 2.

Now that we have created both our security groups it's time to **create the Ansible master EC2 Instance**.


Go to the EC2 dashboard, and click on launch instance:



The screenshot shows the AWS EC2 Dashboard 'Resources' section. It lists various EC2 resources in the US East (N. Virginia) Region. The 'Instances' count is 1, and 'Security groups' count is 2.

Resource	Count
Instances (running)	0
Dedicated Hosts	0
Instances	1
Load balancers	0
Security groups	2
Volumes	1
Auto Scaling Groups	0
Elastic IPs	0
Key pairs	1
Placement groups	0
Snapshots	0

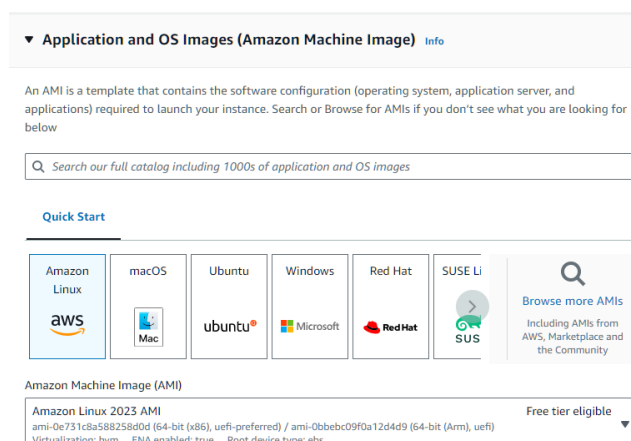
Provide the name for your EC2 Instance:



The screenshot shows the 'Name and tags' configuration page in the AWS Management Console. The 'Name' field is filled with 'ansible master'.

Name:

Under **AMI**, select Amazon Linux 2023 AMI which is free-tier eligible:



The screenshot shows the 'Application and OS Images (Amazon Machine Image)' selection page in the AWS Management Console. It displays a search bar and a 'Quick Start' section with various operating system logos. The 'Amazon Linux 2023 AMI' is highlighted as 'Free tier eligible'.

Application and OS Images (Amazon Machine Image)

Search:

Quick Start

Amazon Linux 2023 AMI
ami-0e731c8a588258d0d (64-bit x86, uefi-preferred) / ami-0bbebc09f0a12d4d9 (64-bit Arm, uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Under **Instance type**, choose t2.micro which provides 1 vCPU, 1 GiB Memory, and is also free-tier eligible.

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

Instance type

t2.micro

Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.0116 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour

Free tier eligible

☐ All generations

[Compare instance types](#)

Additional costs apply for AMLs with pre-installed software

Under **key pair**, allowing us to ssh into the EC2 instance, we can use an existing or create a new pair. For this demo, we will create a new 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*

ansiblekp

[Create new key pair](#)

Now edit your **Network Settings**. Select the default VPC, select the subnet of your choice, or leave with no preference. Auto-assign public IP should be enabled and under security, groups choose the existing security group for the ansible-master server we created in step-1.

▼ Network settings [Info](#)

VPC - *required* [Info](#)

vpc-0f07f2a2e5a1d0a91 (default)

Subnet [Info](#)

No preference

Auto-assign public IP [Info](#)

Enable

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

ansible-sg sg-0452d7316820ecff X

[Compare security group rules](#)

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

Rest of the settings we can keep them at default and click on the **Launch instance**.

Now on the Connect to instance windows, copy the following commands and paste them into your terminal in order to ssh into the server:

Step 3.

Creating a key pair on the Ansible master server that would be used to establish a connection between the Ansible master and Ansible nodes with the command:

ssh-keygen -t rsa -b 2048

```
[ec2-user@ip-172-31-40-98 ~]$ chmod 400 ansiblekp.pem
chmod: cannot access 'ansiblekp.pem': No such file or directory
[ec2-user@ip-172-31-40-98 ~]$ ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ec2-user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ec2-user/.ssh/id_rsa
Your public key has been saved in /home/ec2-user/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Nhjh/nUlvQVONMuOHsaur81SPjRM78aAtgepT7tIs ec2-user@ip-172-31-40-98.ec2.internal
The key's randomart image is:
+----[RSA 2048]-----+
|  .  oo. |
|  . . +. |
|  o .+. |
|  . to .o. |
|  B S+o+. |
|  + oBo+ o |
|  . o.oE + |
|  o .o+. = |
|  .+++=. |
+-----[SHA256]-----+
[ec2-user@ip-172-31-40-98 ~]$
```

Step 4.

Now after creating the SSH key pairs we need to import the public key into the EC2 console which would allow the Ansible nodes to use that key in order to establish communication with the Ansible master server.

cd .ssh

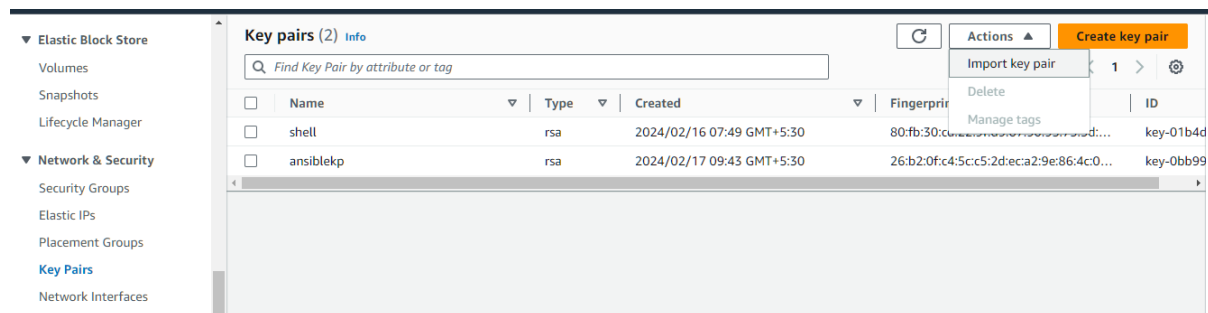
ls

cat id_rsa.pub

Copy the public key by following the below steps as shown in the output:

```
(ec2-user@ip-172-31-40-98 ~)$ cd .ssh
(ec2-user@ip-172-31-40-98 .ssh)$ ls
authorized_keys  id_rsa  id_rsa.pub
(ec2-user@ip-172-31-40-98 .ssh)$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDN8oCIBovR4w7Az1Shqi6GJAnIxADwVyn7Xz1VoSI/0L9bDZq0Cqu9eQ9d2JlG0lVvYBn9P9UfghxNY+WR/wwmVuEX8E0UuCz45gHwx2+91
iteKLEbiodMMAGnpfPTc3XY+18bEyMhKdVlpjaa8m45mR/iJN5Syte73T2QidSKTeuXq2lkg3wBF1N9Vn8sa0ud4lLfQZoubgsJvJwjgRIRD183DvYkiBHDjkEQLG7vcJ28vm9uBsx29d0D
YmKXKdIjfaTmjaPU3RMO4Y1/Q94a2njz6BD1fbJ4DsbYVWx69SHI6xkPscD/PeF01USAURQ/rL+mG60TApS1seFMBn6X ec2-user@ip-172-31-40-98.ec2.internal
```

After copying the public key, go to the EC2 management console, and on the left navigation bar under **Network & Security**, select **Key Pairs**. Then under **Actions** click on **Import key pair**.



Provide the name of the new key being imported and paste the public key from the terminal. Then click on **Import key pair**.

Import key pair

Import settings

Name

ansible-pub-key

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair file

Browse

Choose **Browse** and navigate to your public key. You may change the name of your key. Alternatively, paste the contents of your public key into the **Public key contents** text box.

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDN8oCIBovR4w7Az1Shqi6GJAnIxADwVyn7Xz1VoSI/0L9bDZq0Cqu9eQ9d2JlG0lVvYBn9P9UfghxNY+WR/wwmVuEX8E0UuCz45gHwx2+91iteKLEbiodMMAGnpfPTc3XY+18bEyMhKdVlpjaa8m45mR/iJN5Syte73T2QidSKTeuXq2lkg3wBF1N9Vn8sa0ud4lLfQZoubgsJvJwjgRIRD183DvYkiBHDjkEQLG7vcJ28vm9uBsx29d0DYmKXKdIjfaTmjaPU3RMO4Y1/Q94a2njz6BD1fbJ4DsbYVWx69SHI6xkPscD/PeF01USAURQ/rL+mG60TApS1seFMBn6X ec2-user@ip-172-31-40-98.ec2.internal

Step 5.

Launch Ansible Node Servers:

Now we will launch three EC2 Instances for this demo.

Go to the EC2 dashboard and click on **Launch instances**.

Provide the name and under the **Number of instances** type 3.

Launch an instance [Info](#)

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

Name and tags [Info](#)

Name

ansible-node

Select **Amazon Linux 2023 AMI** and Instance type as **t2.micro**.

Here we have to make sure that under the key pair, we need select **ansible-public-key** which we created earlier in our demo.

Key pair (login) [Info](#)

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

Key pair name - required

ansible-pub-key

Under **Network settings** select the default VPC, Auto-assign public IP as enabled. Make sure to select the existing security group **ansible-nodes-sg** which we created in earlier steps.

Network settings [Info](#)

VPC - required [Info](#)

vpc-0f07f2a2e5a1d0a91 (default)

Subnet [Info](#)

No preference

Create new subnet

Auto-assign public IP [Info](#)

Enable

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

ansible-nodes-sg sg-0eb8067784fc4978 X

Compare security group rules

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

Keep the rest of the settings at default and launch the instances.

After successfully launching, we can view the instances and tag them appropriately as shown below:

Instances (4) Info								
<div>Find Instance by attribute or tag (case-sensitive)</div>								
Running < 1 > ⚙								
<input type="checkbox"/>	Name ↗	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability :	
<input type="checkbox"/>	ansile master	i-0b8d88aa38ce26247	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	
<input type="checkbox"/>	ansible-node 1	i-0f76374315c43f173	Running	t2.micro	Initializing	View alarms	us-east-1a	
<input type="checkbox"/>	ansible-node 2	i-0da614942e7a04c36	Running	t2.micro	Initializing	View alarms	us-east-1a	
<input type="checkbox"/>	ansible-node 3	i-02f35b8b92f03673a	Running	t2.micro	Initializing	View alarms	us-east-1a	

Step 6.

Test the connectivity between Ansible master and node servers:

To test the ssh connection between the master and worker/slave nodes, we would log into the master Ansible server, and take the private IP of the first node server from the console as shown below:

The screenshot shows the AWS Management Console. The top section displays a list of instances under the 'Instances (1/4)' heading. The table below shows the details of the instances:

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	ansible master	i-0b8d88aa38ce26247	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a
<input checked="" type="checkbox"/>	ansible-node 1	i-0f76374315c43f173	Running	t2.micro	Initializing	View alarms	us-east-1a
<input type="checkbox"/>	ansible-node 2	i-0da614942e7a04c36	Running	t2.micro	Initializing	View alarms	us-east-1a
<input type="checkbox"/>	ansible-node 3	i-02f35b8b92f03673a	Running	t2.micro	Initializing	View alarms	us-east-1a

The bottom section shows the 'Instance summary for i-0f76374315c43f173 (ansible-node 1)'. It provides details for this specific instance:

- Instance ID: i-0f76374315c43f173 (ansible-node 1)
- Public IPv4 address: 18.212.140.154
- Private IPv4 addresses: 172.31.37.251
- Instance state: Running
- Public IPv4 DNS: ec2-18-212-140-154.compute-1.amazonaws.com

Copy the Private IPV4 address

Go to the terminal of the Ansible master server and type the following command as shown in the below screenshot.

```
[ec2-user@ip-172-31-40-98 .ssh]$ ssh 172.31.37.251
The authenticity of host '172.31.37.251 (172.31.37.251)' can't be established.
ED25519 key fingerprint is SHA256:fCLMe9//m5WMBOr9TFdRi844dLM4V3jFjW2d4bUm4AM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.31.37.251' (ED25519) to the list of known hosts.

      #
     _#_
    _###_
   _###_
  _###_
 _###_
v~'-'>

      Amazon Linux 2023

      https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-172-31-37-251 ~]$
```

In the above output, check the private IP which should be the same one that we copied earlier from the AWS console.

In the same way, test the connectivity between the other two servers and the master server.

Note: Since we are able to connect between the Ansible Master Server and all other nodes, it is to be noted that this connection is made possible by the SSH itself and not Ansible as so far we haven't installed the Ansible in the master server.

Step 7.

Install Ansible on the Master Server:

First, we will update our Ansible master server with the below command:

```
sudo yum update -y
```

Since we are using Amazon Linux 2023 AML, the installation steps would be different than that for Amazon Linux2 AML.

Copy and paste the below commands to install Ansible:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

```
python3 get-pip.py --user
```

```
python3 -m pip install --user ansible
```

Output:

```
Collecting resolvelib<1.1.0,>=0.5.3 (from ansible-core==2.15.7->ansible)
  Downloading resolvelib-1.0.1-py2.py3-none-any.whl.metadata (4.0 kB)
Collecting importlib-resources<5.1,>=5.0 (from ansible-core==2.15.7->ansible)
  Downloading importlib_resources-5.0.7-py3-none-any.whl (24 kB)
Collecting MarkupSafe>=2.0 (from Jinja2==3.0.0->ansible-core==2.15.7->ansible)
  Downloading MarkupSafe-2.1.5-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.0 kB)
Requirement already satisfied: cffi>=1.12 in /usr/lib64/python3.9/site-packages (from cryptography->ansible-core==2.15.7->ansible) (1.14.5)
Requirement already satisfied: pycparser in /usr/lib/python3.9/site-packages (from cffi>=1.12->cryptography->ansible-core==2.15.7->ansible) (2.20)
Requirement already satisfied: ply==3.11 in /usr/lib/python3.9/site-packages (from pycparser->ffi>=1.12->cryptography->ansible-core==2.15.7->ansible) (3.11)
Downloading ansible-8.7.0-py3-none-any.whl (48.4 MB)
   48.4/48.4 MB 28.4 MB/s eta 0:00:00
Downloading ansible_core-2.15.9-py3-none-any.whl (2.2 MB)
   2.2/2.2 MB 58.4 MB/s eta 0:00:00
Downloading Jinja2-3.1.3-py3-none-any.whl (133 kB)
   133.2/133.2 kB 14.7 MB/s eta 0:00:00
Downloading resolvelib-1.0.1-py2.py3-none-any.whl (17 kB)
Downloading packaging-23.2-py3-none-any.whl (53 kB)
   53.0/53.0 kB 7.0 MB/s eta 0:00:00
Downloading MarkupSafe-2.1.5-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (25 kB)
Installing collected packages: resolvelib, packaging, MarkupSafe, importlib-resources, Jinja2, ansible-core, ansible
Successfully installed MarkupSafe-2.1.5 ansible-8.7.0 ansible-core-2.15.9 importlib-resources-5.0.7 Jinja2-3.1.3 packaging-23.2 resolvelib-1.0.1
[ec2-user@ip-172-31-40-98 ~]$
```

To check if Ansible was successfully installed check the version with the below command:

```
[ec2-user@ip-172-31-40-98 ~]$ ansible --version
ansible [core 2.15.9]
  config file = None
  configured module search path = ['/home/ec2-user/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /home/ec2-user/.local/lib/python3.9/site-packages/ansible
  ansible collection location = /home/ec2-user/.ansible/collections:/usr/share/ansible/collections
  executable location = /home/ec2-user/.local/bin/ansible
  python version = 3.9.16 (main, Sep  8 2023, 00:00:00) [GCC 11.4.1 20230605 (Red Hat 11.4.1-2)] (/usr/bin/python3)
  Jinja2 version = 3.1.3
  libyaml = True
[ec2-user@ip-172-31-40-98 ~]$
```

Step 8.

Create an Ansible Inventory file:

Ansible automates tasks on managed nodes or “hosts” in your infrastructure, using a list or group of lists known as inventory. You can pass host names at the command line, but most Ansible users create inventory files. Your inventory defines the managed nodes you automate, with groups so you can run automation tasks on multiple hosts at the same time.

Within the inventory file, you can organize your servers into different groups and subgroups.

Let's create an Inventory file on the Ansible Master Server with the below command:
sudo vim inventory

Create a group name **webservers** and under it type the IP addresses of all the node servers as shown below:

```
[ec2-user@ip-172-31-40-98 ~]$ cat inventory
[webservers]
172.31.37.251
172.31.40.53
172.31.44.193
```

Step 9.

Create an Ansible Playbook:

An Ansible Playbook is a blueprint of [automation](#) tasks — which are complex IT actions executed with limited or no human involvement. Ansible Playbooks are executed on a set, group, or classification of hosts, which together make up an Ansible inventory.

Let's create the Playbook on the Ansible Master Serve in the home directory where we have our inventory file as well:

sudo vim website.yml

Once the file gets opened paste the code into the playbook file. You can find the link to the GitHub repo at the end of this blog.

Before running the playbook to install our website on the Apache server let's first test the connectivity between the master and all the nodes using the Ansible command:

ansible all --key-file ~/.ssh/id_rsa -i inventory -m ping -u ec2-user

The successful output of the above command should look like this:

```
[ec2-user@ip-172-31-40-98 ~]$ ansible all --key-file ~/.ssh/id_rsa -i inventory -m ping -u ec2-user
[WARNING]: Platform linux on host 172.31.37.251 is using the discovered Python interpreter at /usr/bin/python3.9, but future installation of
another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.15/reference_appendices/interpreter_discovery.html for more information.
172.31.37.251 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3.9"
  },
  "changed": false,
  "ping": "pong"
}
[WARNING]: Platform linux on host 172.31.44.193 is using the discovered Python interpreter at /usr/bin/python3.9, but future installation of
another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.15/reference_appendices/interpreter_discovery.html for more information.
172.31.44.193 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3.9"
  },
  "changed": false,
  "ping": "pong"
}
[WARNING]: Platform linux on host 172.31.40.53 is using the discovered Python interpreter at /usr/bin/python3.9, but future installation of
another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.15/reference_appendices/interpreter_discovery.html for more information.
172.31.40.53 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3.9"
  },
  "changed": false,
  "ping": "pong"
}
```

The above output in green shows that we are able to connect with the nodes using Ansible.

One more thing to do before running our Ansible Playbook is to create an **Ansible configuration file** in which we will list the path to our key pair, inventory file, and the default username of ec2-user so that we don't have to mention every time we run our playbook.

To create the Ansible config file type
sudo vim ansible.cfg

Once the file is open enter the below details as follows:

```
[defaults]
remote_user = ec2-user
inventory = inventory
private_key_file = ~/.ssh/id_rsa
```

Save the file and exit.

Step 10.

Run the Playbook to Install Webserver on the Servers:

To run the playbook type the below command on the Ansible Master Server:

ansible-playbook website.yml

The successful output of the above command should look like this:

```
PLAY [deploy bootstrap-website] *****

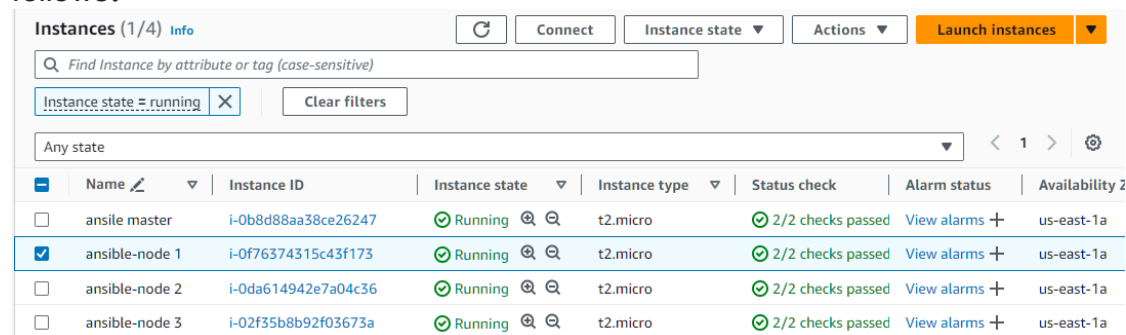
TASK [Gathering Facts] *****
[WARNING]: Platform linux on host 172.31.40.53 is using the discovered Python interpreter at /usr/bin/python3.9, but future installation of
another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.15/reference_appendices/interpreter_discovery.html for more information.
ok: [172.31.40.53]
[WARNING]: Platform linux on host 172.31.37.251 is using the discovered Python interpreter at /usr/bin/python3.9, but future installation of
another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.15/reference_appendices/interpreter_discovery.html for more information.
ok: [172.31.37.251]
[WARNING]: Platform linux on host 172.31.44.193 is using the discovered Python interpreter at /usr/bin/python3.9, but future installation of
another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.15/reference_appendices/interpreter_discovery.html for more information.
ok: [172.31.44.193]

TASK [update ec2 instance] *****
ok: [172.31.37.251]
ok: [172.31.44.193]
ok: [172.31.40.53]
```

At the end of the output, you can see all the tasks executed successfully without any failure:

```
PLAY RECAP *****
172.31.37.251      : ok=10   changed=8    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.40.53      : ok=10   changed=8    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.44.193     : ok=10   changed=8    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

We can also confirm by accessing any one of the Node Server public IP from our browser. Go to the EC2 console and copy the Public IP of Ansible-node-1 server as follows:



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability zone
<input type="checkbox"/>	ansible master	i-0b8d88aa38ce26247	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a
<input checked="" type="checkbox"/>	ansible-node 1	i-0f76374315c43f173	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a
<input type="checkbox"/>	ansible-node 2	i-0da614942e7a04c36	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a
<input type="checkbox"/>	ansible-node 3	i-02f35b8b92f03673a	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a



Instance summary for i-0f76374315c43f173 (ansible-node 1)

Updated less than a minute ago

Instance ID: i-0f76374315c43f173 (ansible-node 1)

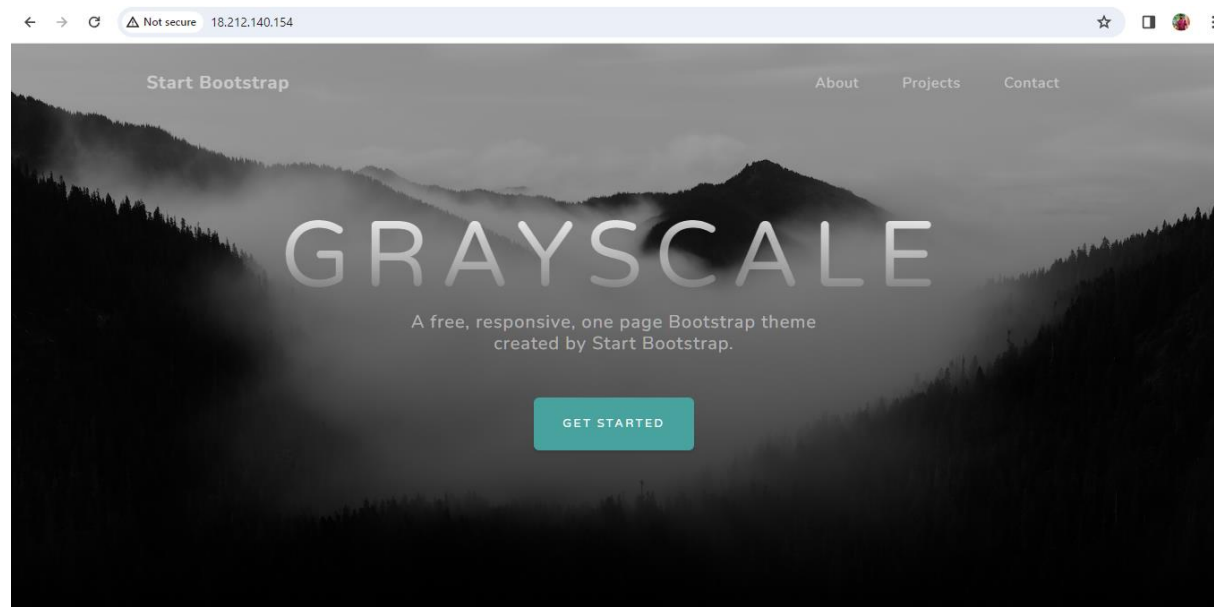
Public IPv4 address: 18.212.140.154 [open address](#)

Private IPv4 addresses: 172.31.37.251

Instance state: Running

Public IPv4 DNS: ec2-18-212-140-154.compute-1.amazonaws.com [open address](#)

Then paste the Public IP on your browser and you should see your website as I am able to see mine:



In the same way, you can confirm the successful deployment of our website on other node servers as well.

Conclusion:

In this blog, we learned how to write a playbook using Ansible on an EC2 instance and deploy a website on an Apache web server on multiple EC2 instances. Make sure you clean up the resources on the AWS cloud.