**12.1** Use Ansible playbook to Configure Reverse Proxy on AWS i.e., HAProxy and update its configuration file automatically on each time new Managed node (Configured with Apache Webserver) join the inventory.

1. What is Load Balancer?

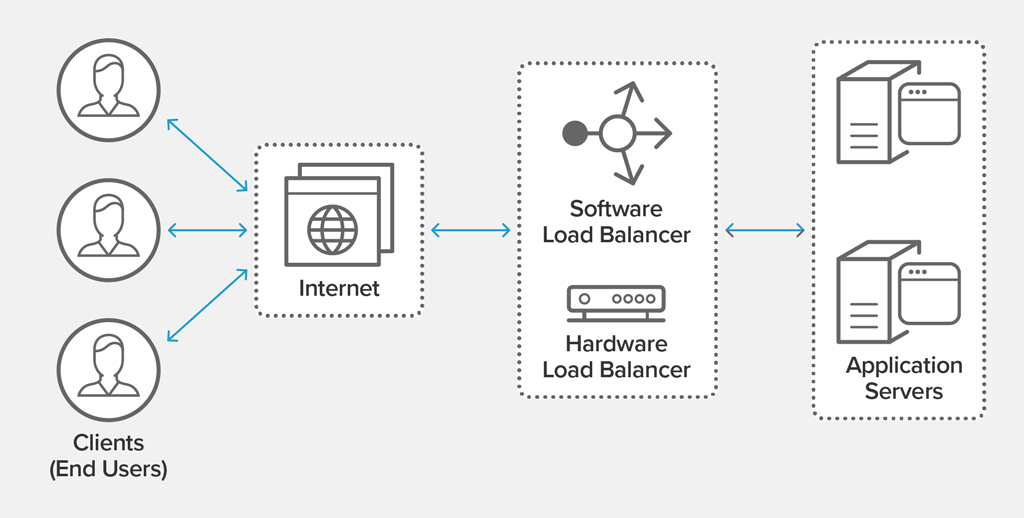
**Load balancing** refers to efficiently distributing incoming network traffic across a group of backend servers, also known as a *server farm* or *server pool*.

Modern high‑traffic websites must serve hundreds of thousands, if not millions, of concurrent requests from users or clients and return the correct text, images, video, or application data, all in a fast and reliable manner. To cost‑effectively scale to meet these high volumes, modern computing best practice generally requires adding more servers.

A [load balancer](https://www.nginx.com/solutions/adc) acts as the “traffic cop” sitting in front of your servers and routing client requests across all servers capable of fulfilling those requests in a manner that maximizes speed and capacity utilization and ensures that no one server is overworked, which could degrade performance. If a single server goes down, the load balancer redirects traffic to the remaining online servers. When a new server is added to the server group, the load balancer automatically starts to send requests to it.

In this manner, a load balancer performs the following functions:

* Distributes client requests or network load efficiently across multiple servers.
* Ensures high availability and reliability by sending requests only to servers that are online.
* Provides the flexibility to add or subtract servers as demand dictates.



Benefits of Load Balancing

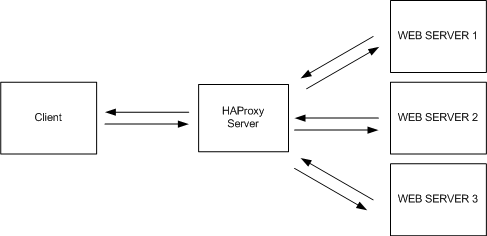
* Reduced downtime
* Scalable
* Redundancy
* Flexibility
* Efficiency

1. What is HAProxy?

HAProxy, which stands for High Availability Proxy, is a popular open-source software TCP/HTTP Load Balancer and proxying solution which can be run on Linux, Solaris, and FreeBSD. Its most common use is to improve the performance and reliability of a server environment by distributing the workload across multiple servers (e.g., web, application, database). It is used in many high-profile environments, including GitHub, Imgur, Instagram, and Twitter.

Ansible:

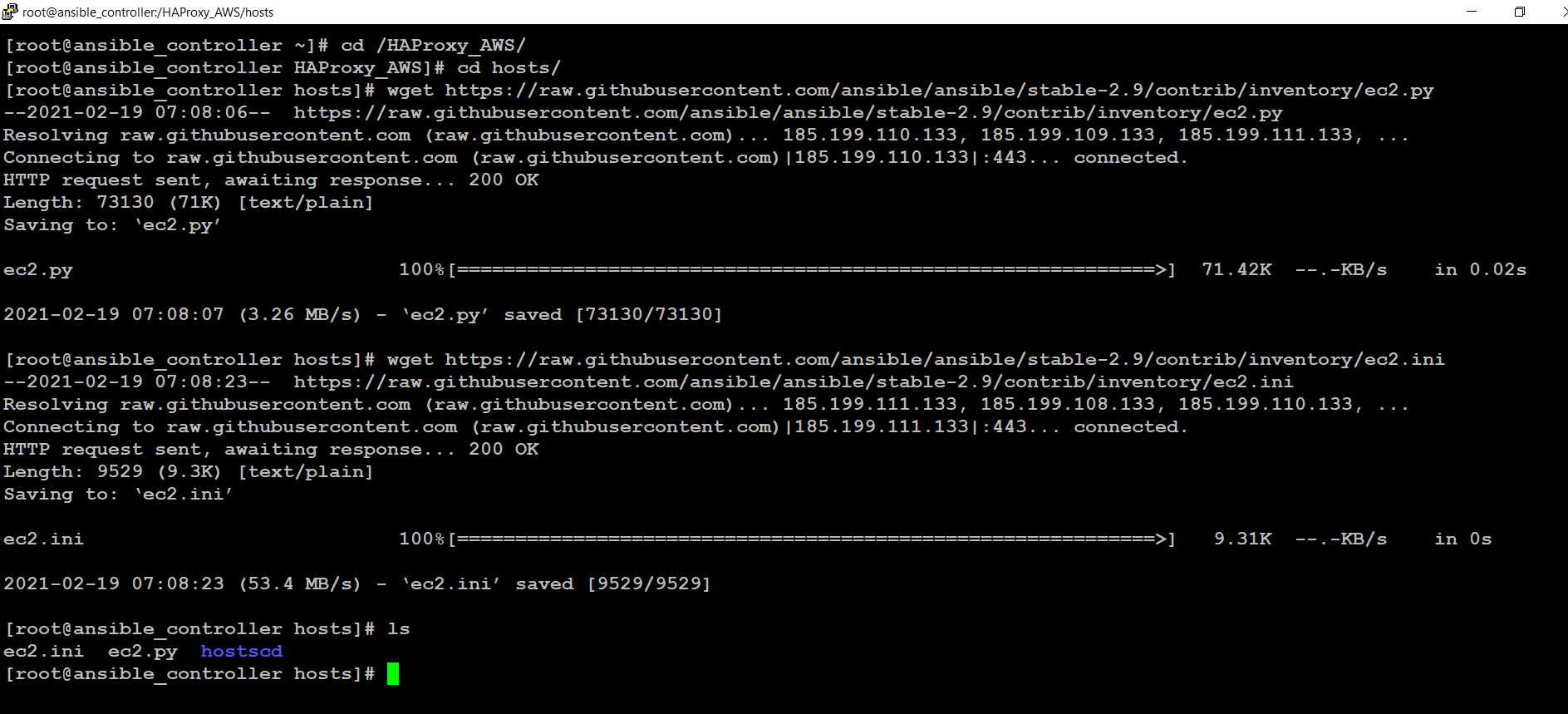
[Ansible](https://www.ansible.com/) is a software tool that provides simple but powerful automation for cross-platform computer support. It is primarily intended for IT professionals, who use it for application deployment, updates on workstations and servers, cloud provisioning, configuration management, intra-service orchestration, and nearly anything a systems administrator does on a weekly or daily basis. Ansible does not depend on agent software and has no additional security infrastructure, so it's easy to deploy.



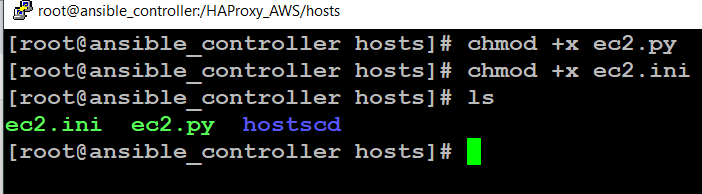
So, you are going to design this architecture using the Automation tool Ansible.

Step1: Download ec2.py and ec2.ini using:

* wget <https://raw.githubusercontent.com/ansible/ansible/stable-2.9/contrib/inventory/ec2.py>
* wget <https://raw.githubusercontent.com/ansible/ansible/stable-2.9/contrib/inventory/ec2.ini>

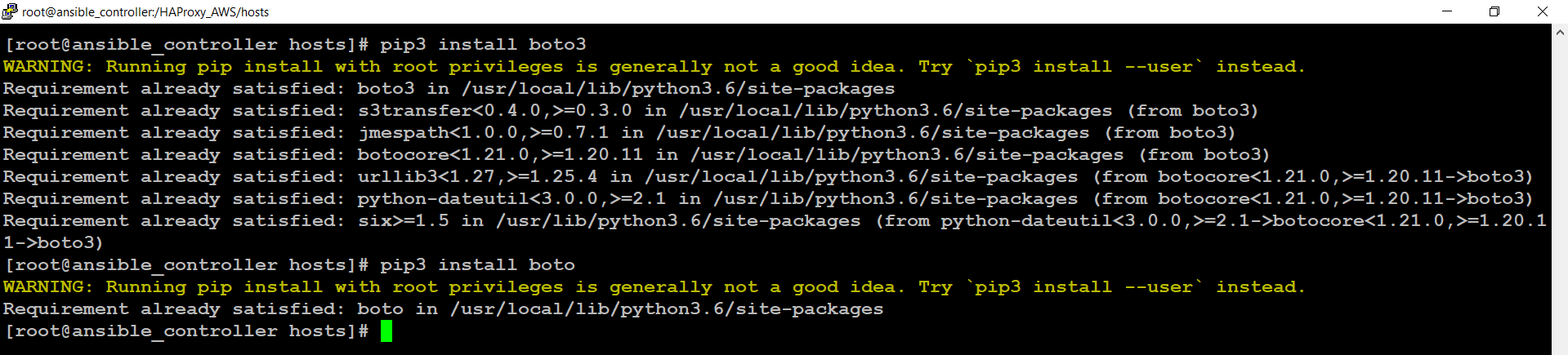


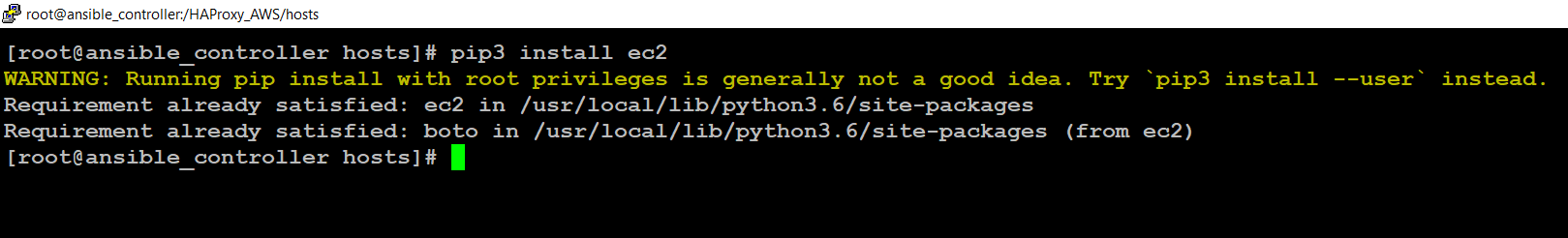
* Make ec2.py and ec2.ini executable using:
* chmod +x ec2.py
* chmod +x ec2.ini



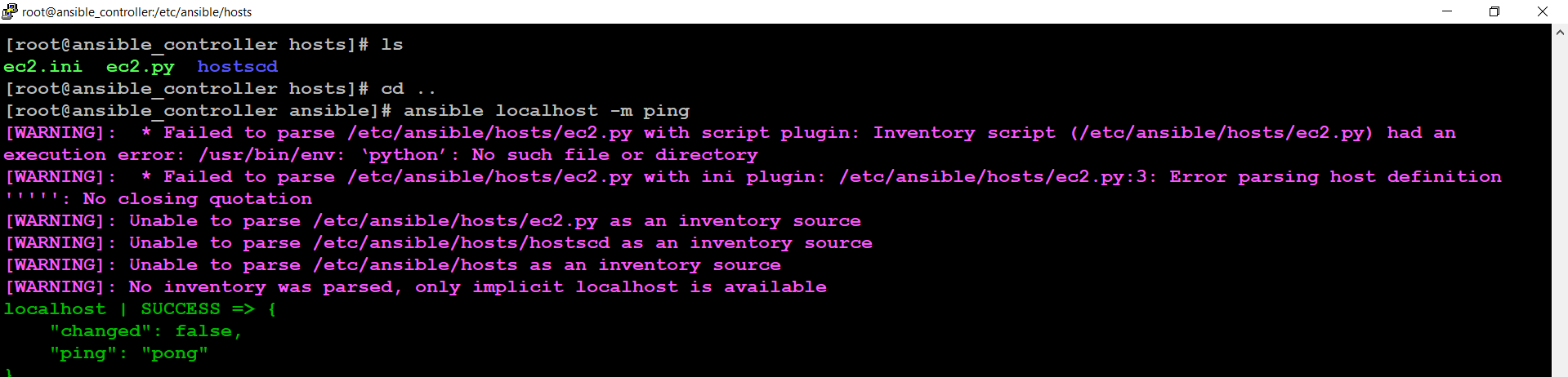
Step2: Download Boto3, Boto, ec2 packages using:

* pip3 install boto3
* pip3 install boto
* pip3 install ec2



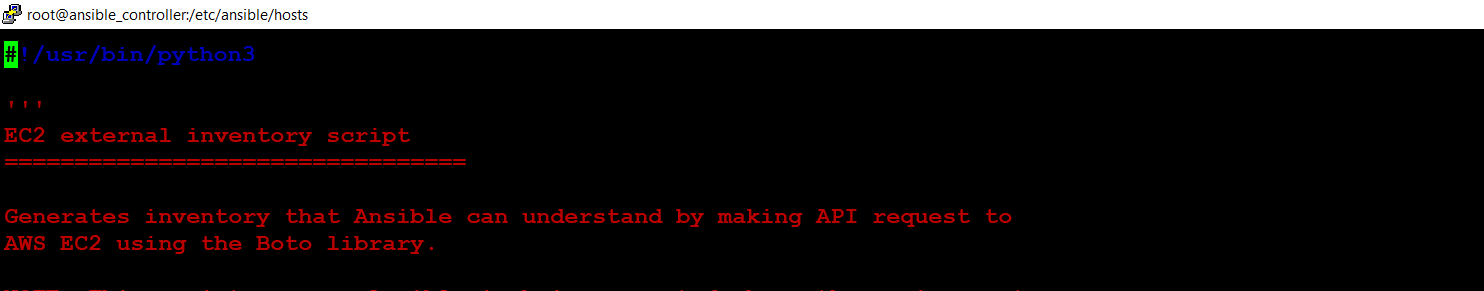


* Check the dynamic inventory is working fine or not.
* You will get these kinds of warnings.

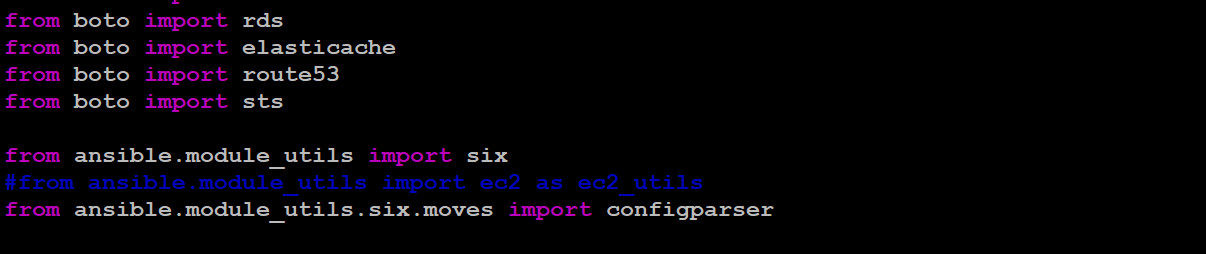


* To solve this Warnings
* Open ec2.py, In line1 Change

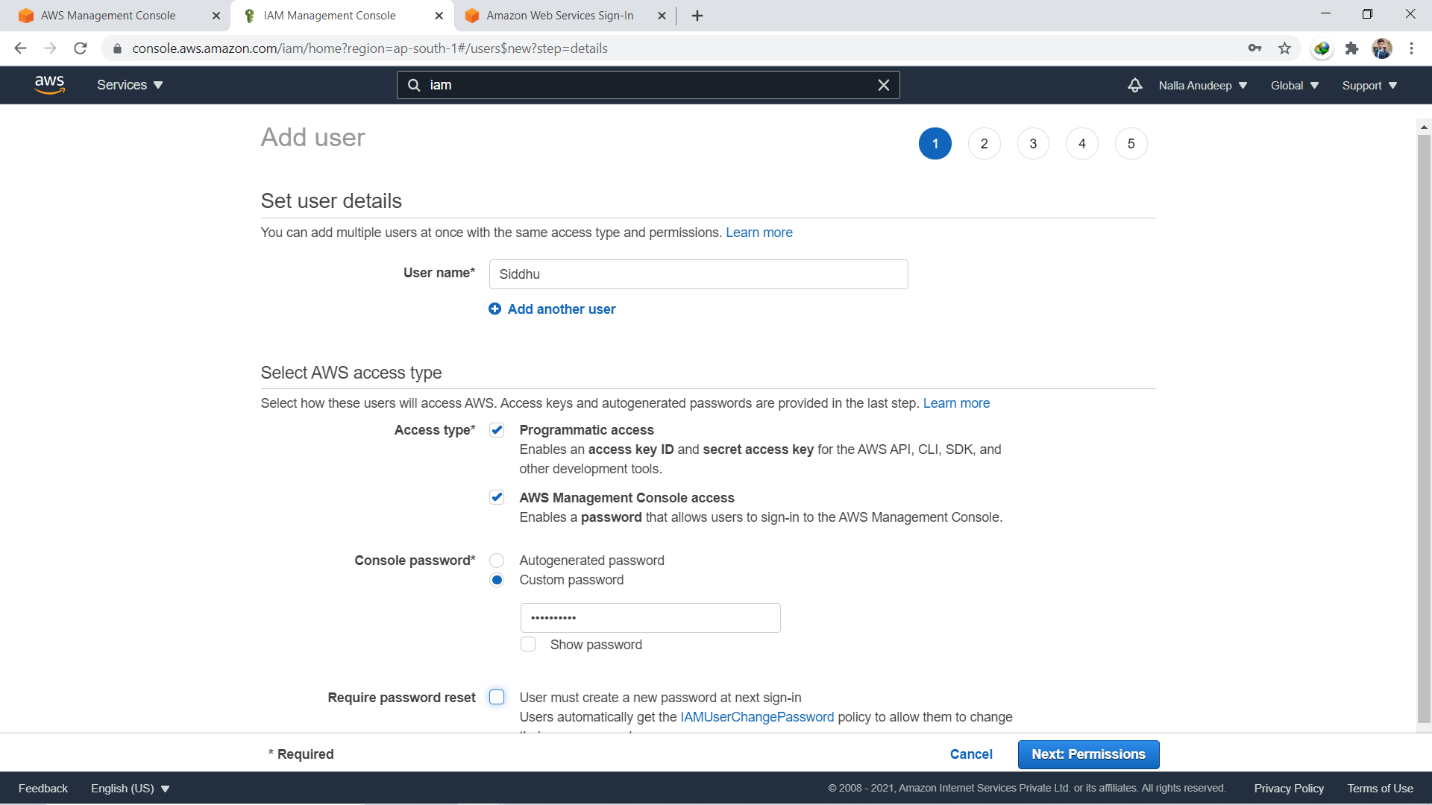
#!/usr/bin/env python 🡪 #!/usr/bin/python3

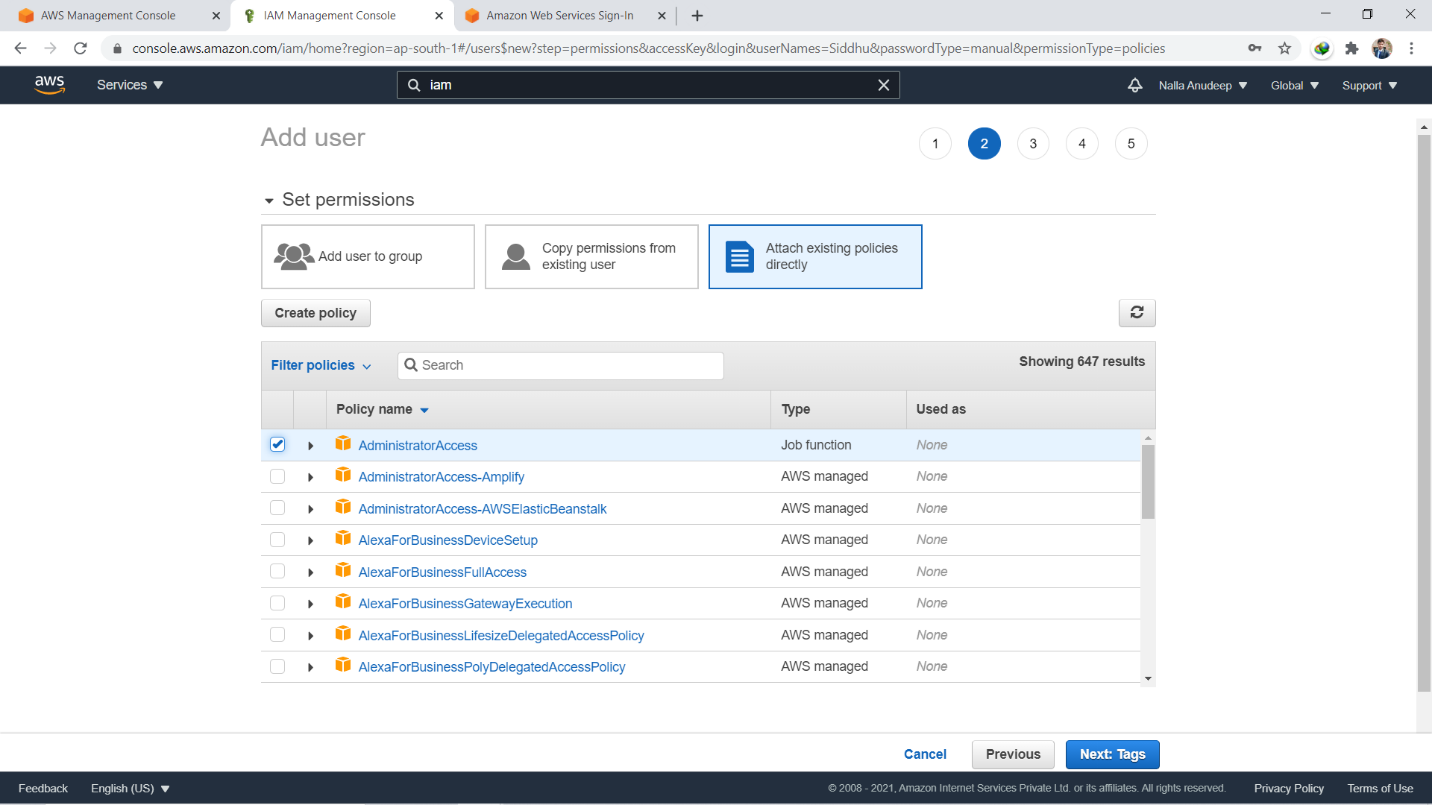


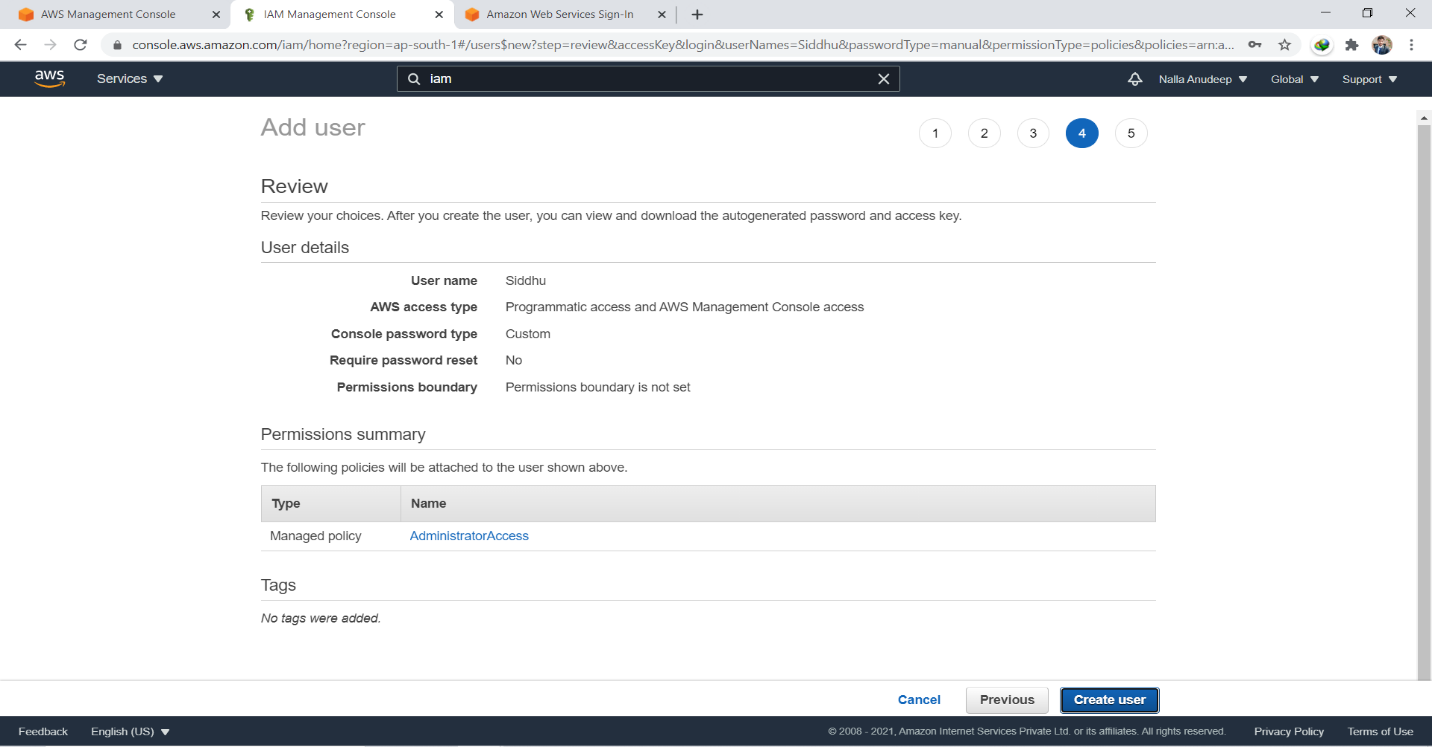
* Comment the Line 172 in ec2.py i.e.,
* from ansible.module\_utils import ec2 as ec2\_utils



Step3: Create the IAM user and export Access key and Secret key of user.



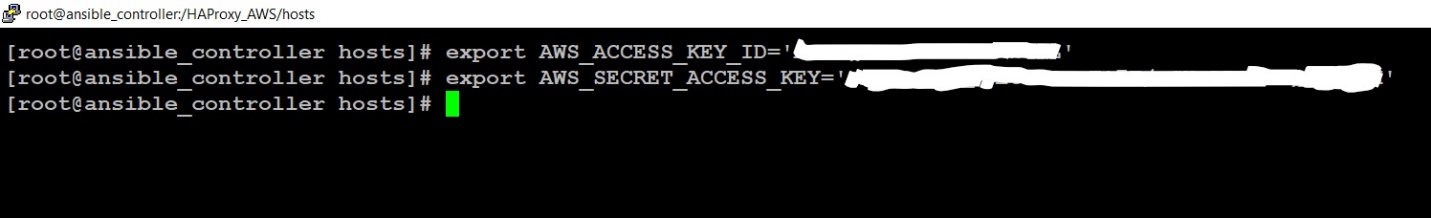




The keyword used to set key is –

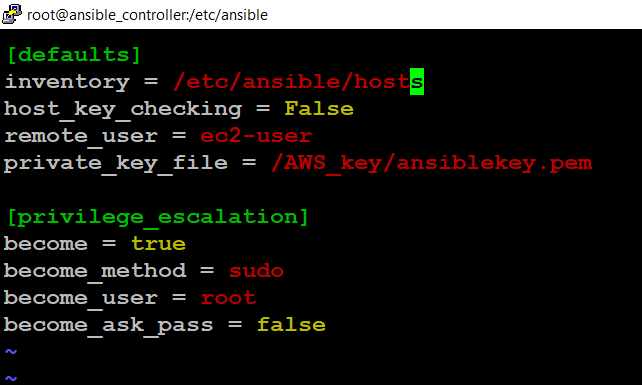
**🡪 export AWS\_ACCESS\_KEY\_ID=’<Your\_Access\_key>’**

**🡪 export AWS\_SECRET\_ACCESS\_KEY=’<Your\_Secret\_Key>’**

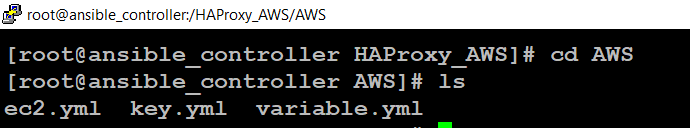


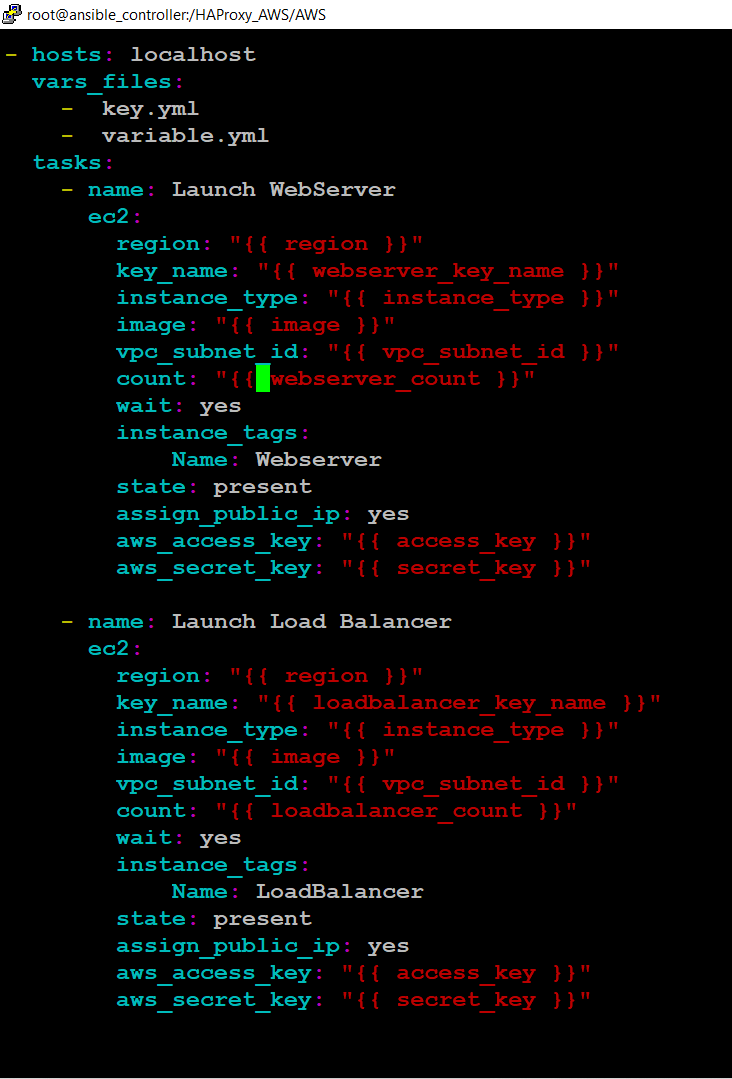
Step4: Setup the ansible.cfg file:

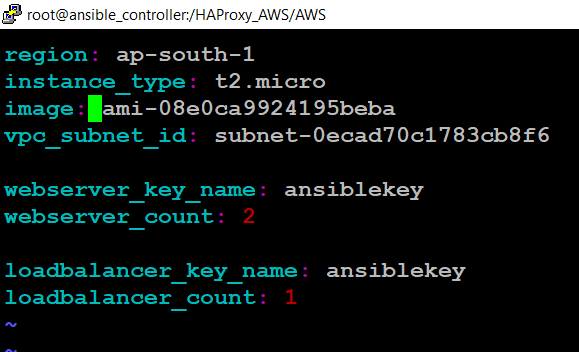
* **Set the path of dynamic inventory (i.e., path of ec2.ini and ec2.py)**
* **Set host key checking to false.**
* **Set remote user as ec2-user.**
* **Set the path of private key (The key used for launch instances i.e., ansiblekey.pem)**
* **Set Privilege escalation**
* **Copy the ansiblekey.pem from windows to CN using Winscp**



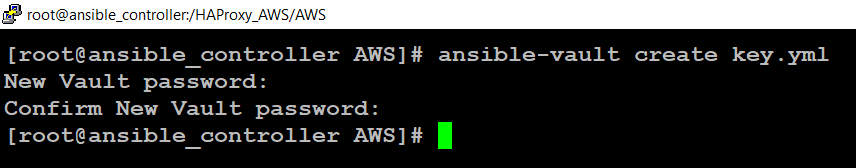
Step5: Create a directory and create ec2.yml, variable.yml.





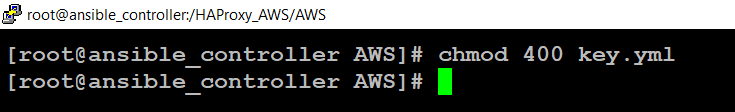


* Create the key.yml in secure mode using the concept of ansible vault and change its permission to read mode.

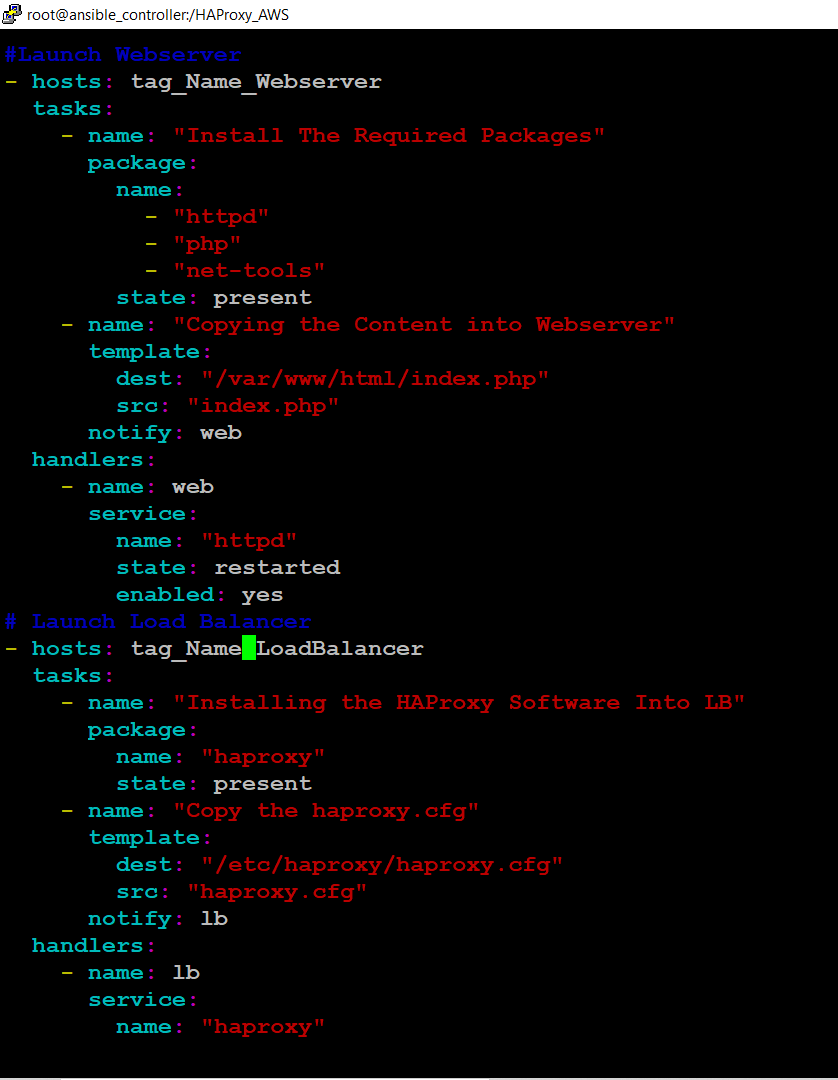


* Key.yml





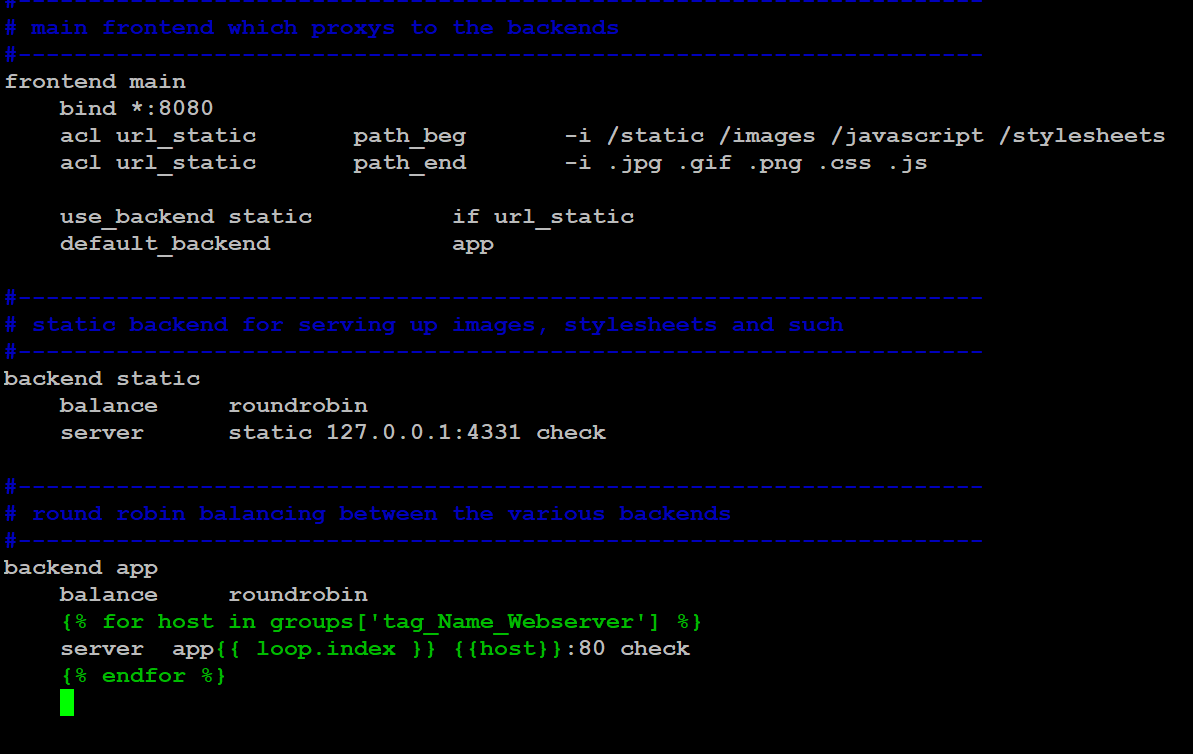
Step6: Create a playbook to configure HAProxy and Webservers.



# Binding with port 8080

* Open the haproxy.cfg file and bind the **port 8080**. And

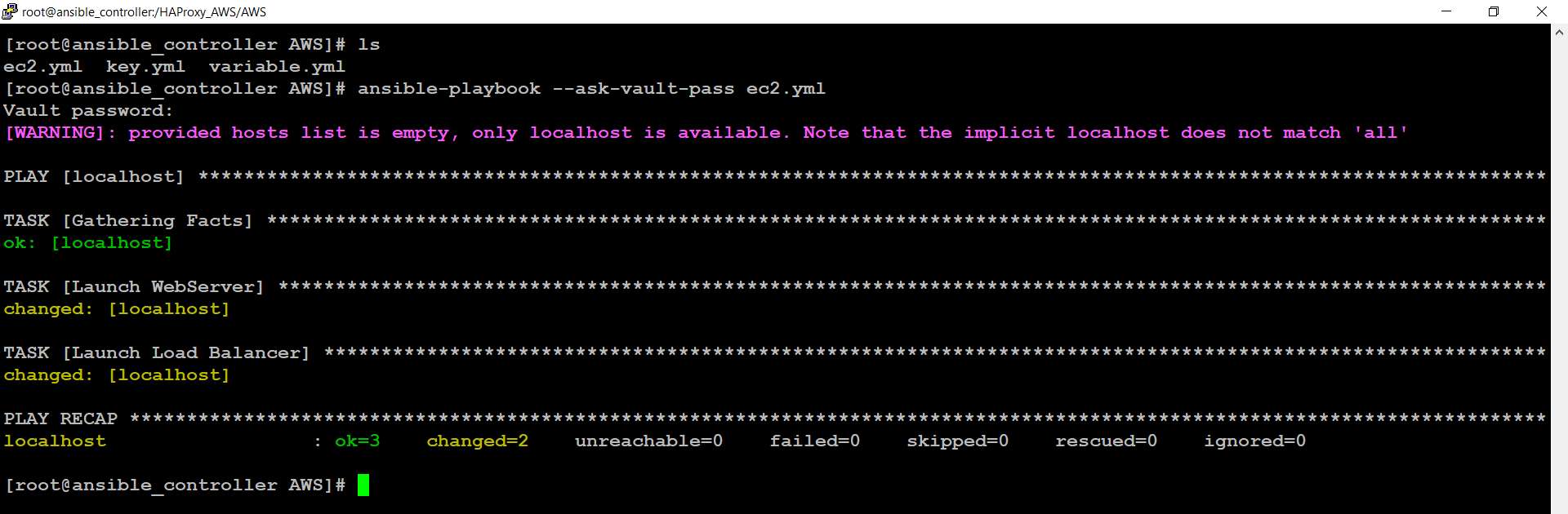
Also, write the jinja code to update the **haproxy.cfg** file to perform Load Balancer Dynamically.



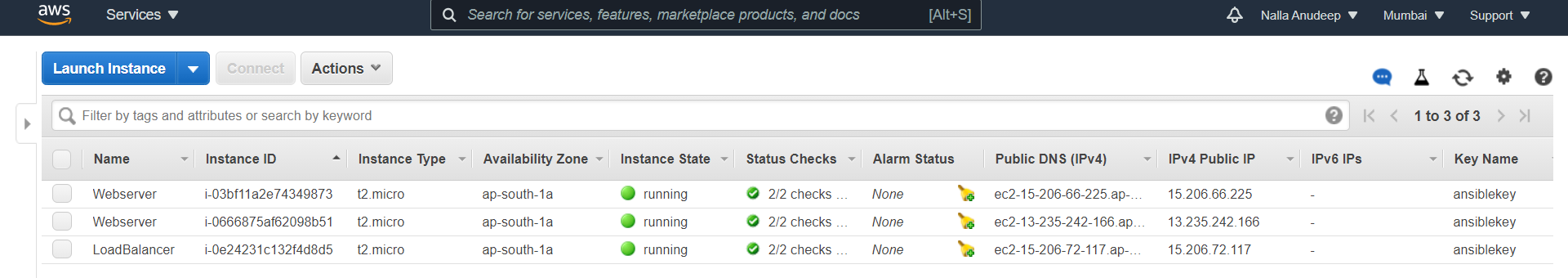
Step7: Launching EC2 instance

Here, I am launching 2 instances as webservers and 1 instance as load balancer.

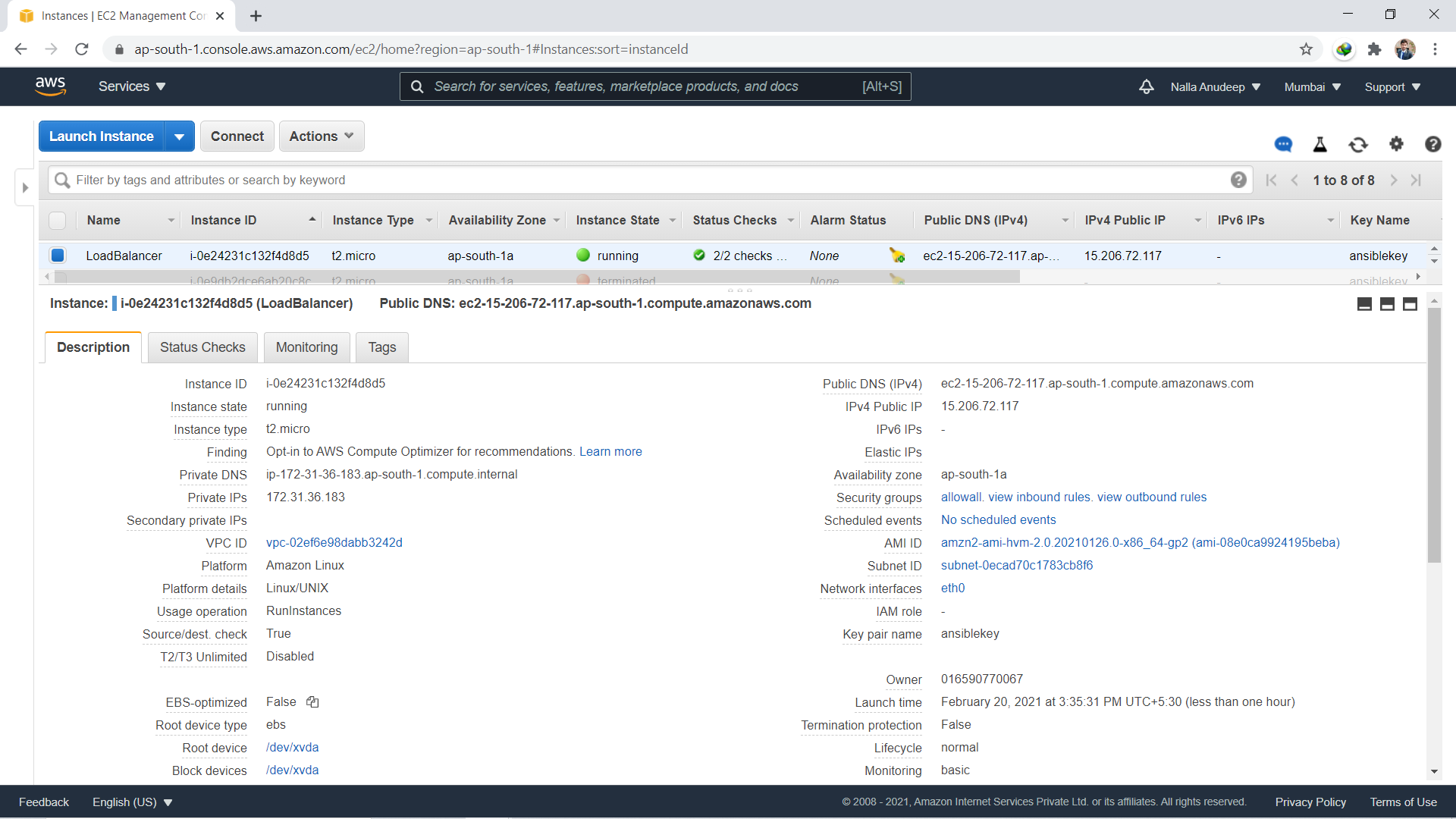
Command: ansible-playbook –ask-vault-pass ec2.yml



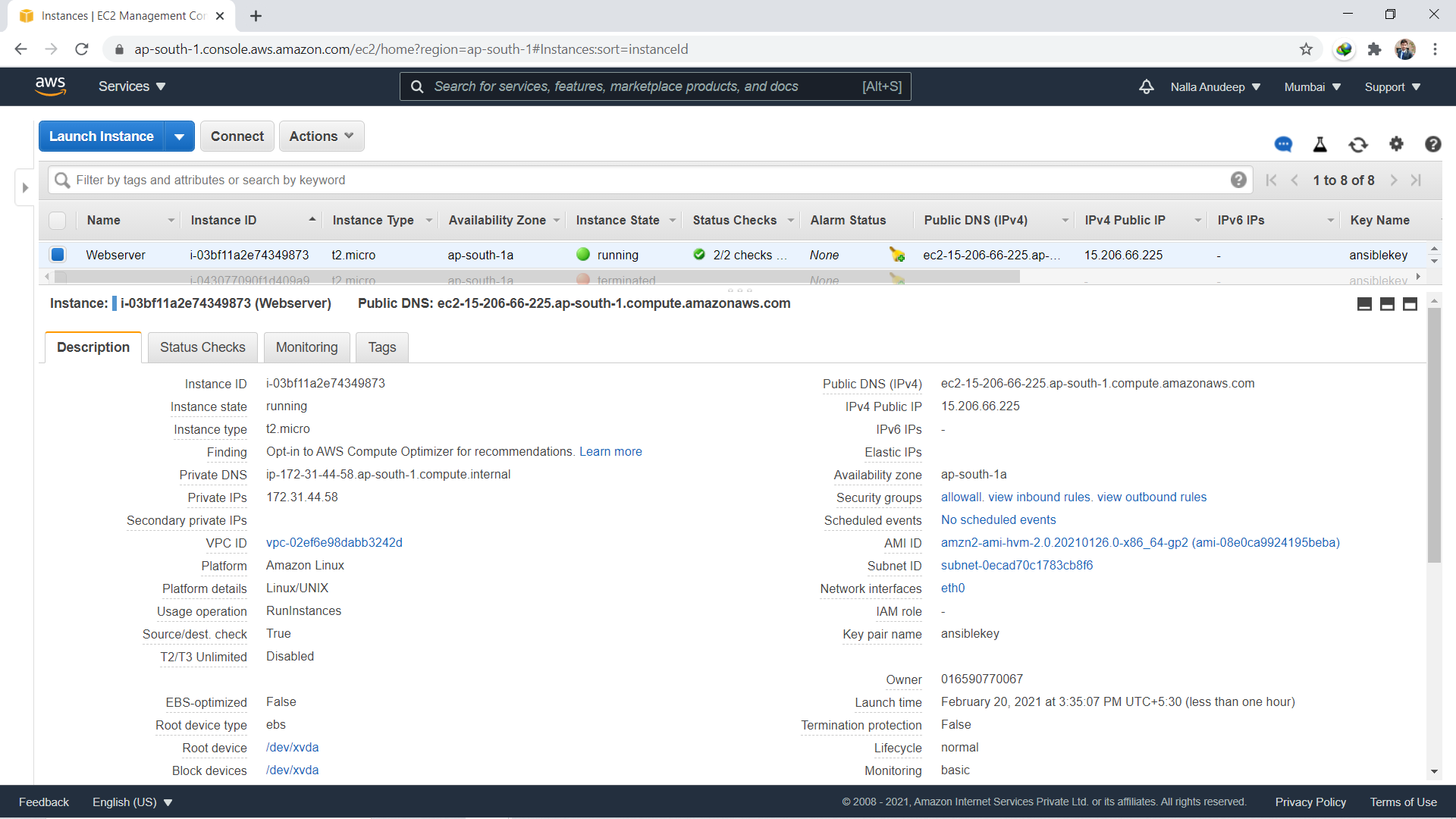
* Load Balancer and Webservers are Successfully Launched.

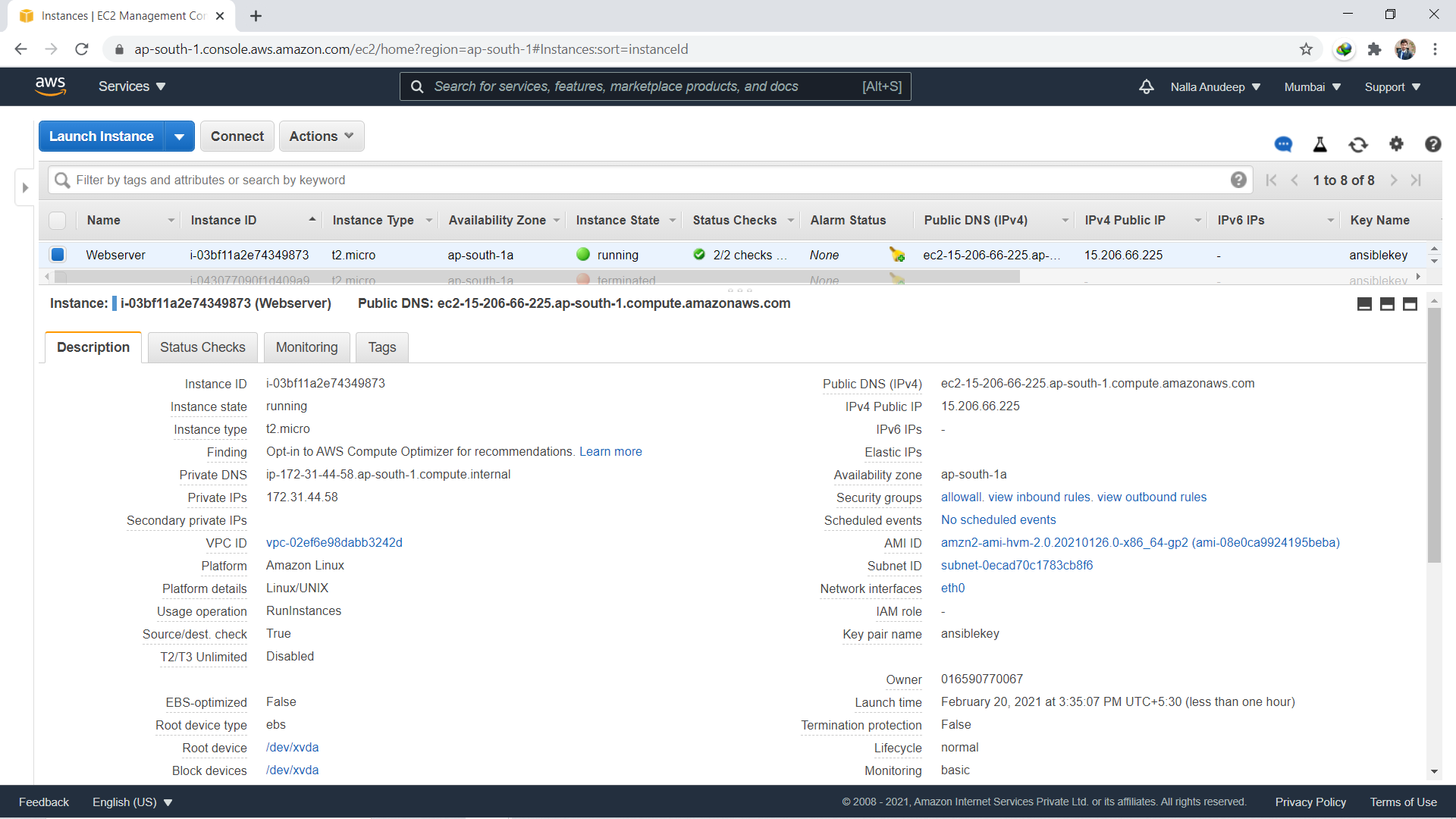


* Load Balancer

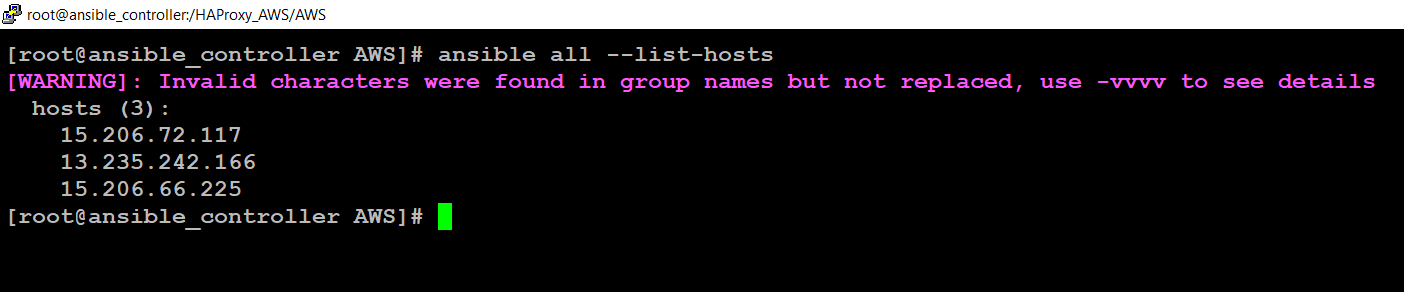


* Webserver





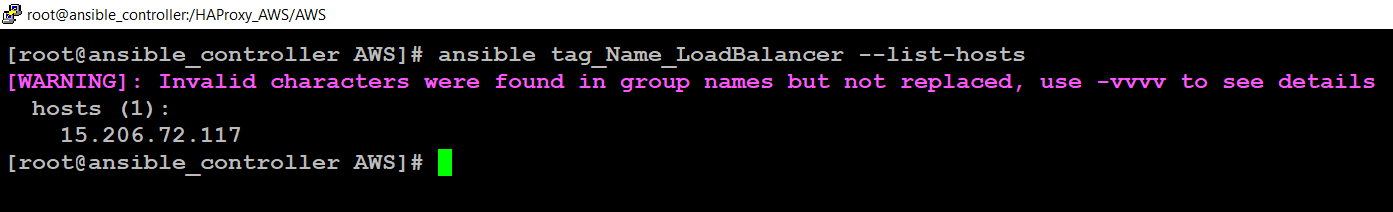
* Checking Dynamic Inventory Hosts
* Command: ansible all –list-hosts.



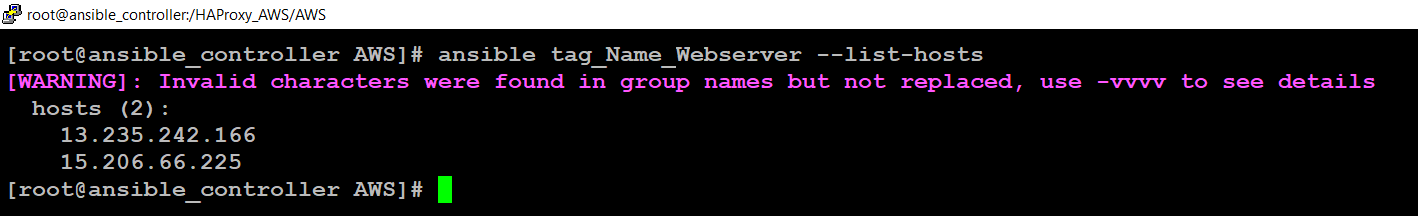
* As we have given tags to each instance so we can check hosts by using tag names.

Command: **“ ansible tag\_Key\_Value — list-hosts ”**

* **Load Balancer**

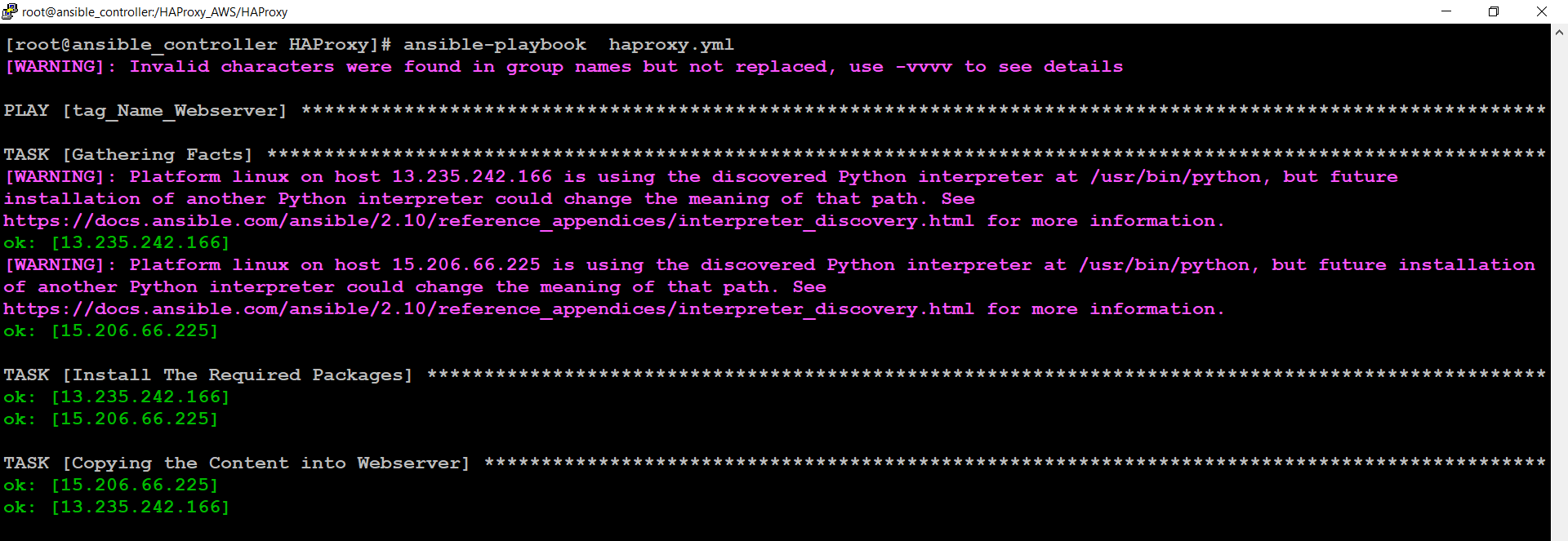


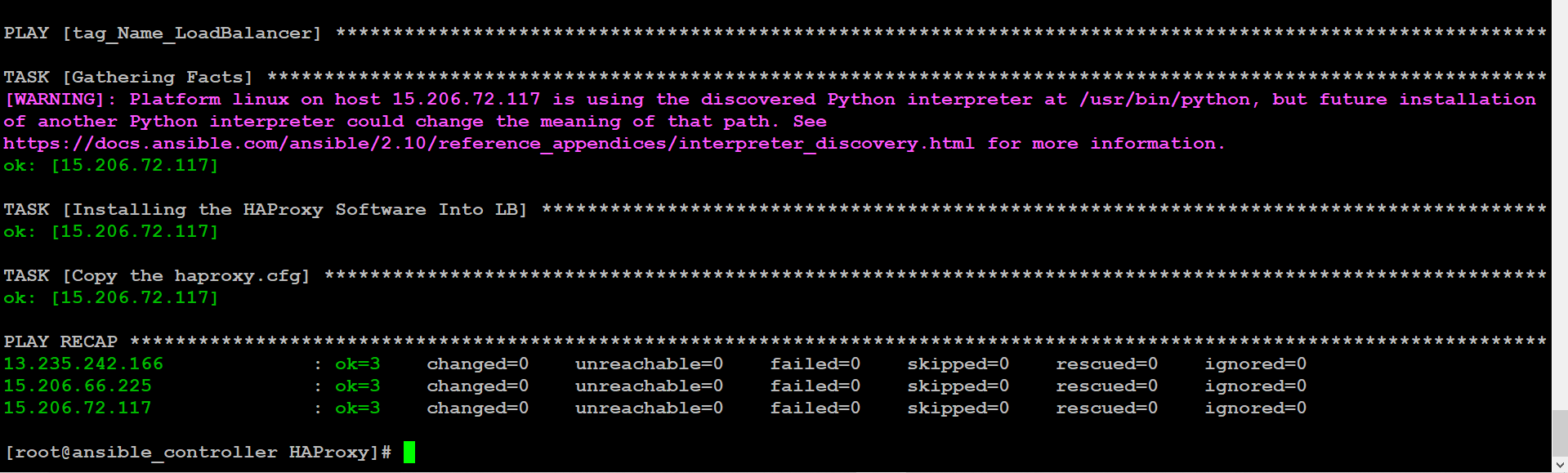
* **Web Server**



**Step8: LAUNCH REVERSE PROXY SETUP**

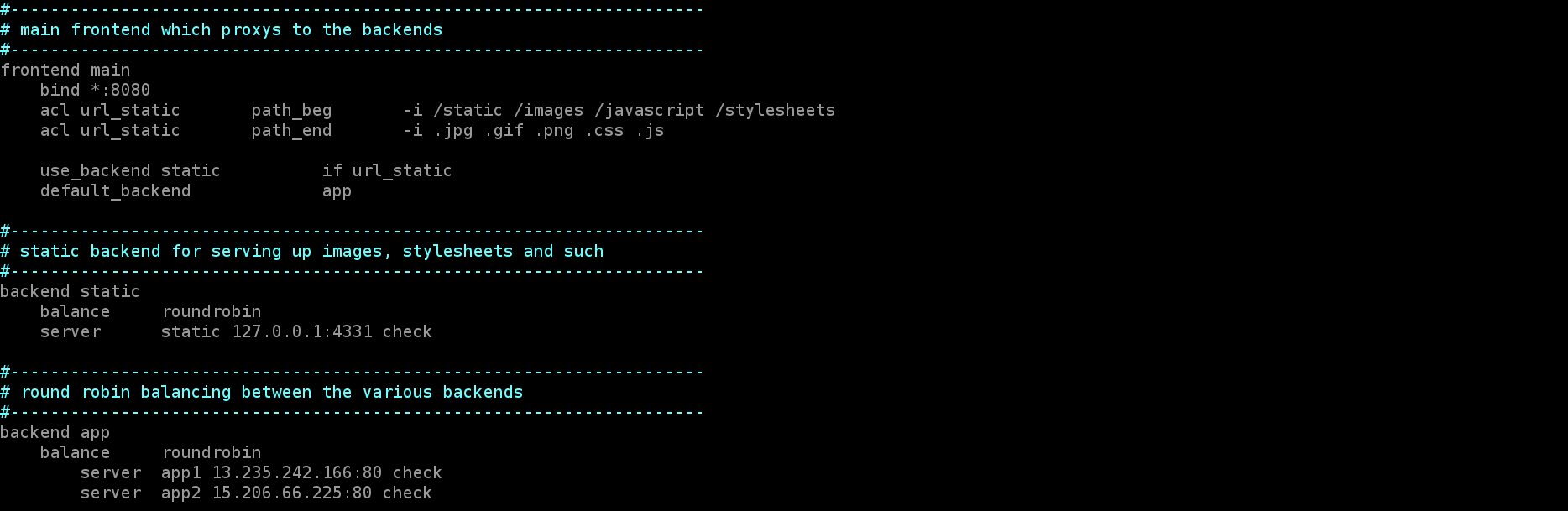
* **Launch the HAProxy playbook using:**
* **ansible-playbook <file\_name>.yml**
* **use verbose -v if you want to see more details:**



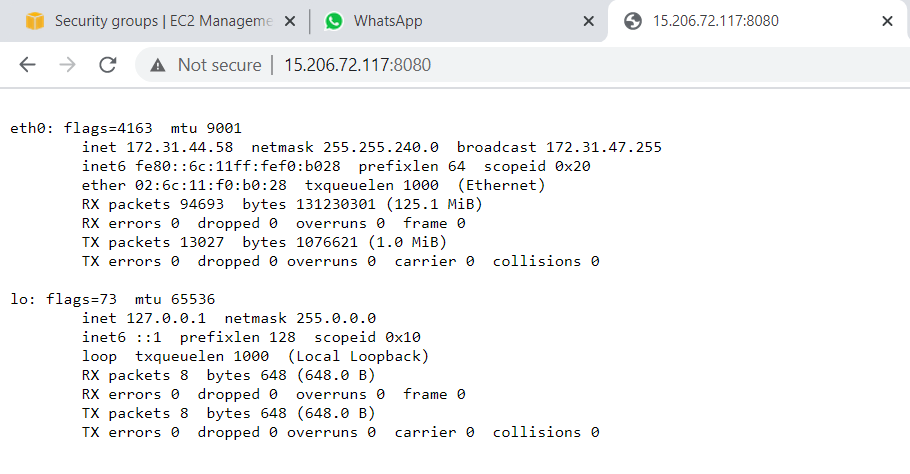


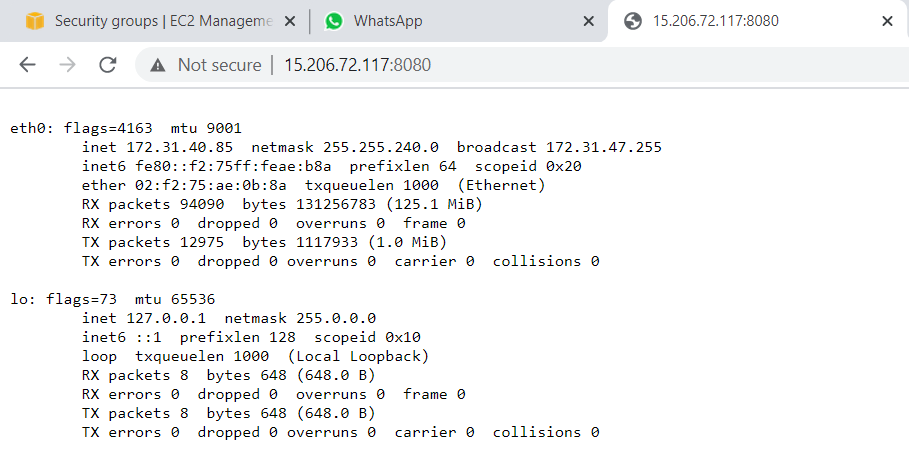
* Check the **haproxy.cfg** file is updated, in LB instance using:

Vim /etc/haproxy/haproxy.cfg

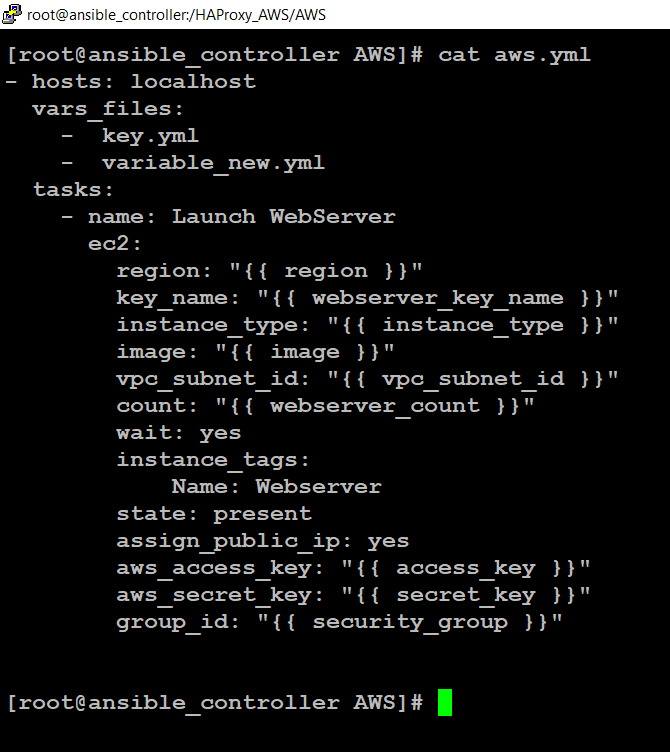


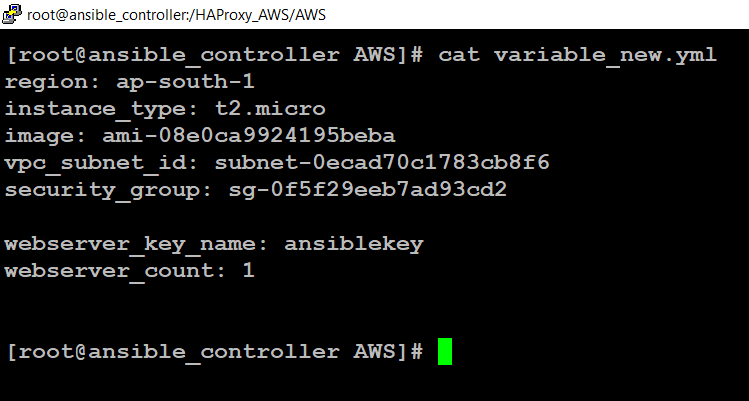
* You can see in the above screenshot, the **haproxy.cfg** file is updated.
* Let’s check the Load Balancer is working or not.





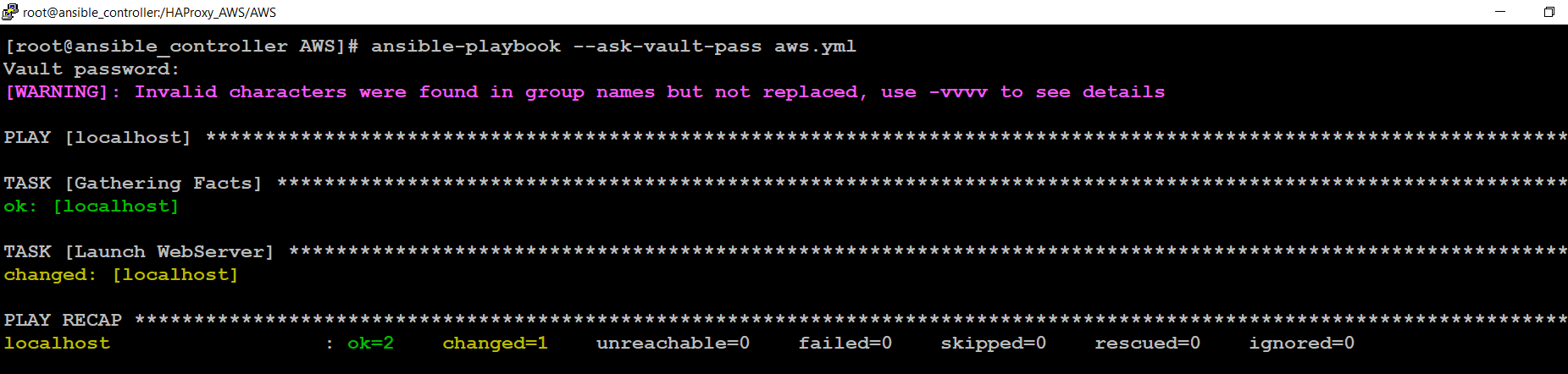
* It works well with two web servers.
* Let’s Launch one more webserver instance using aws.yml

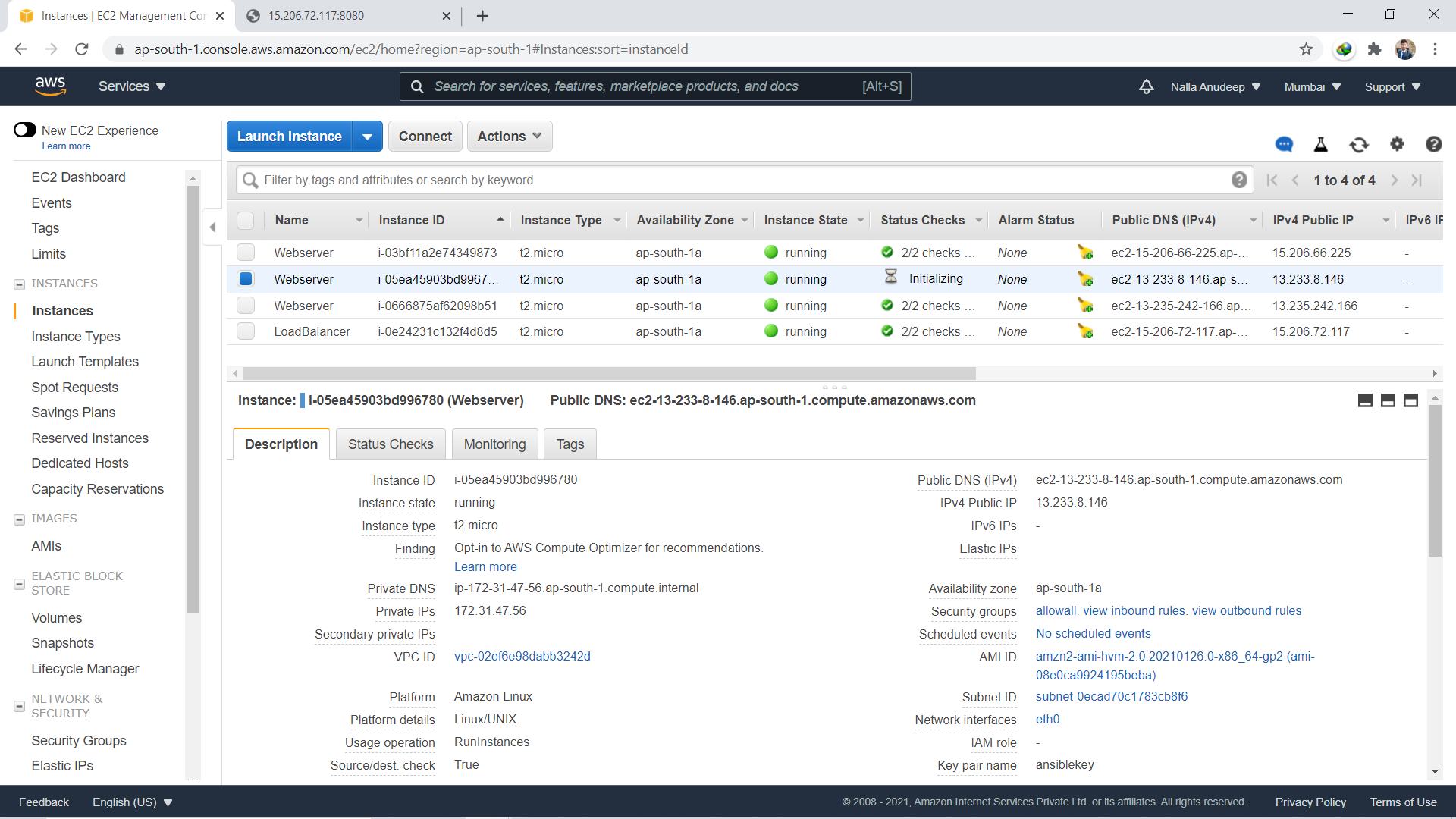




Step9: Launching new Webserver instance.

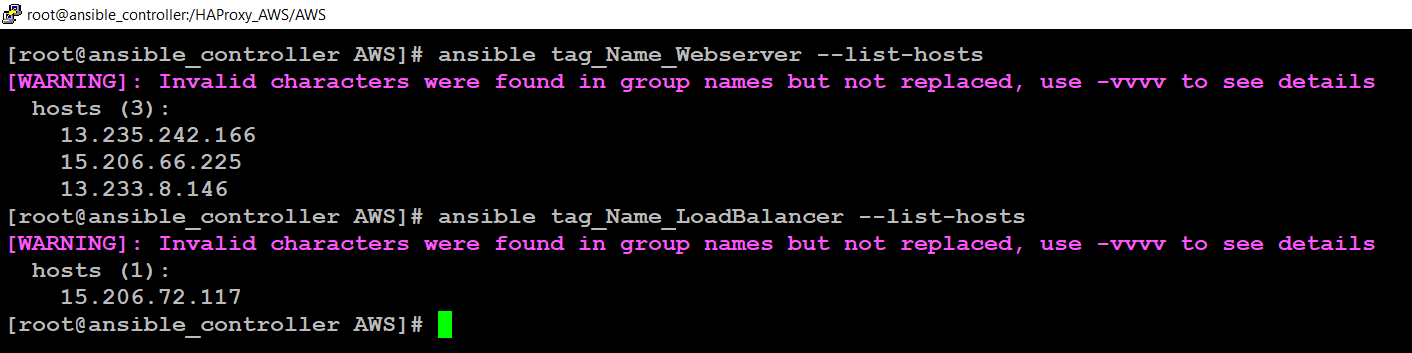
Command: ansible-playbook –ask-vault-pass aws.yml





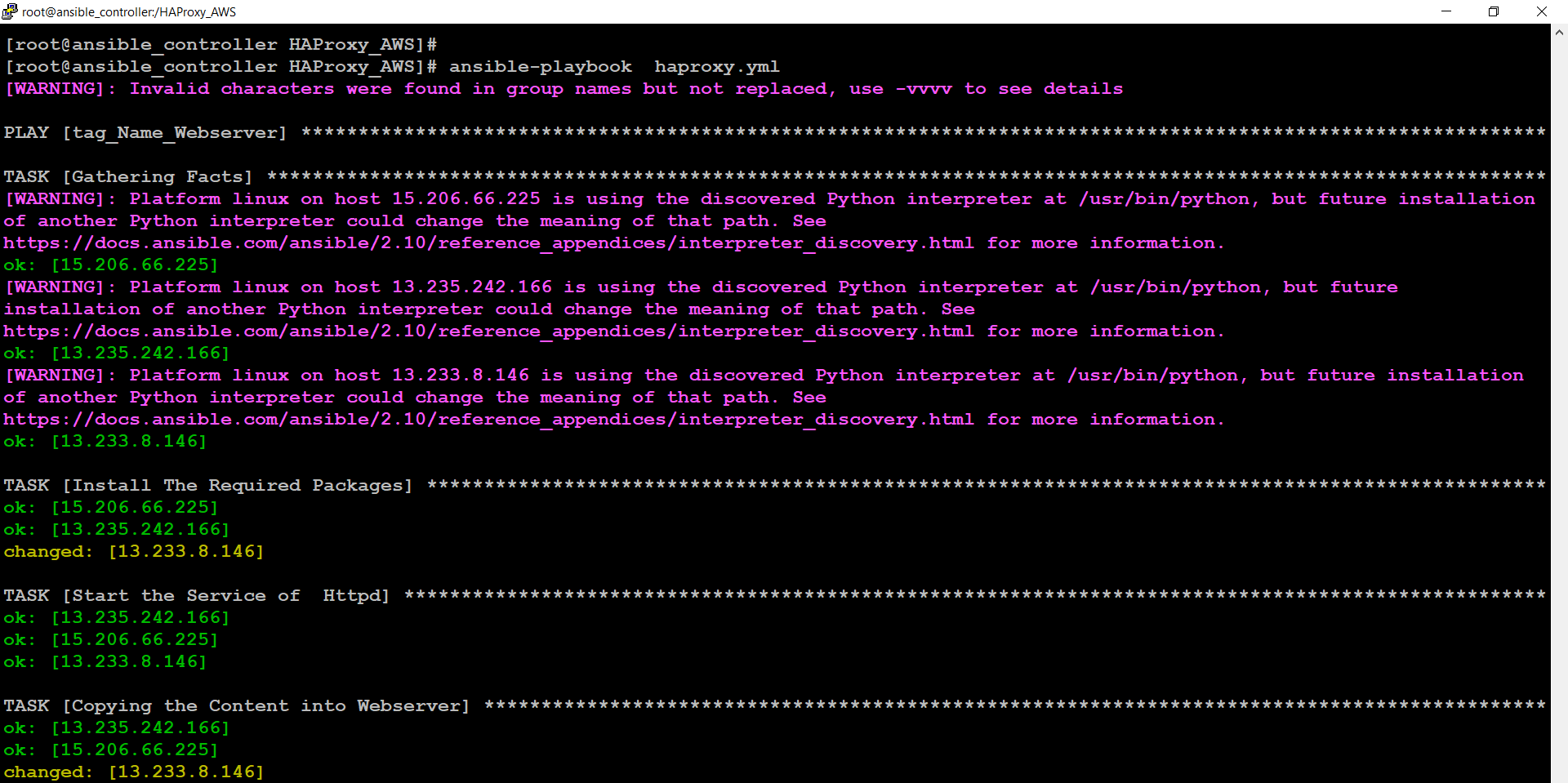
* One more new webserver is launched.
* As we have given tags to each instance so we can check hosts by using tag names.

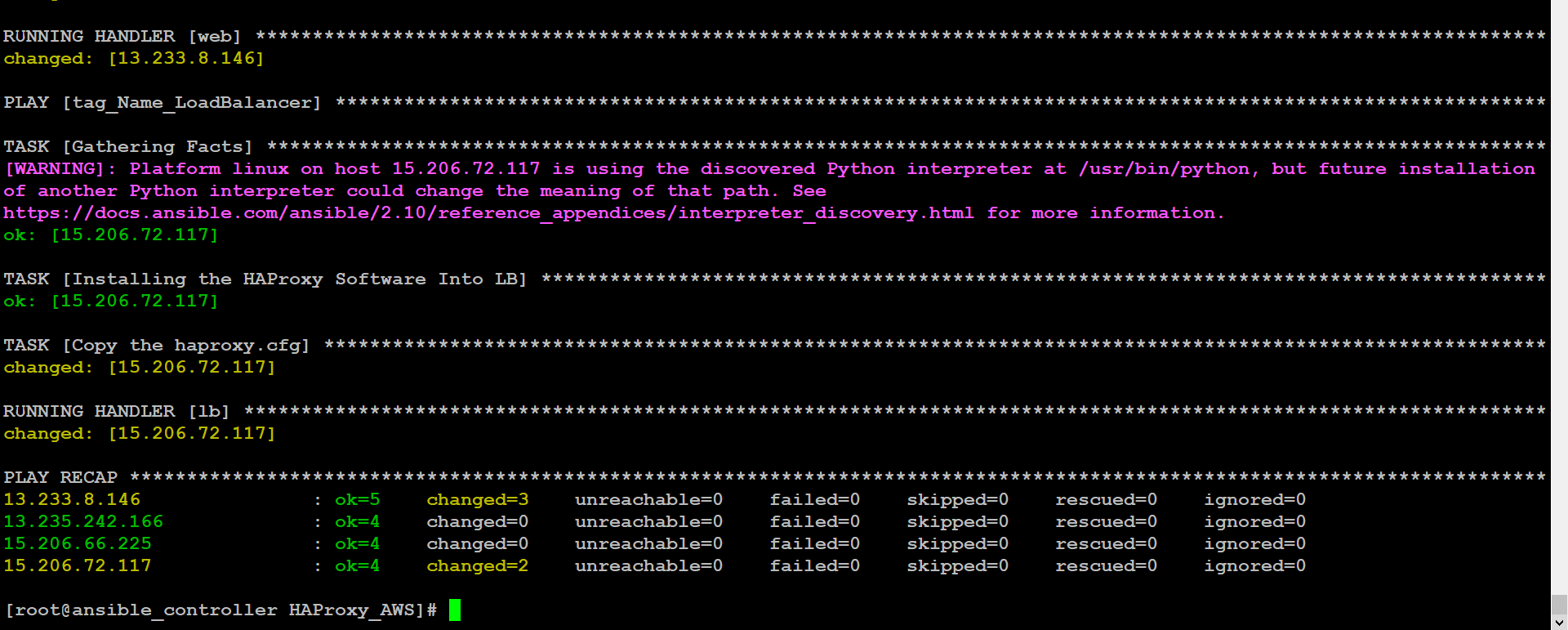
Command: **“ ansible tag\_Key\_Value — list-hosts ”**



**Step10: LAUNCH REVERSE PROXY SETUP**

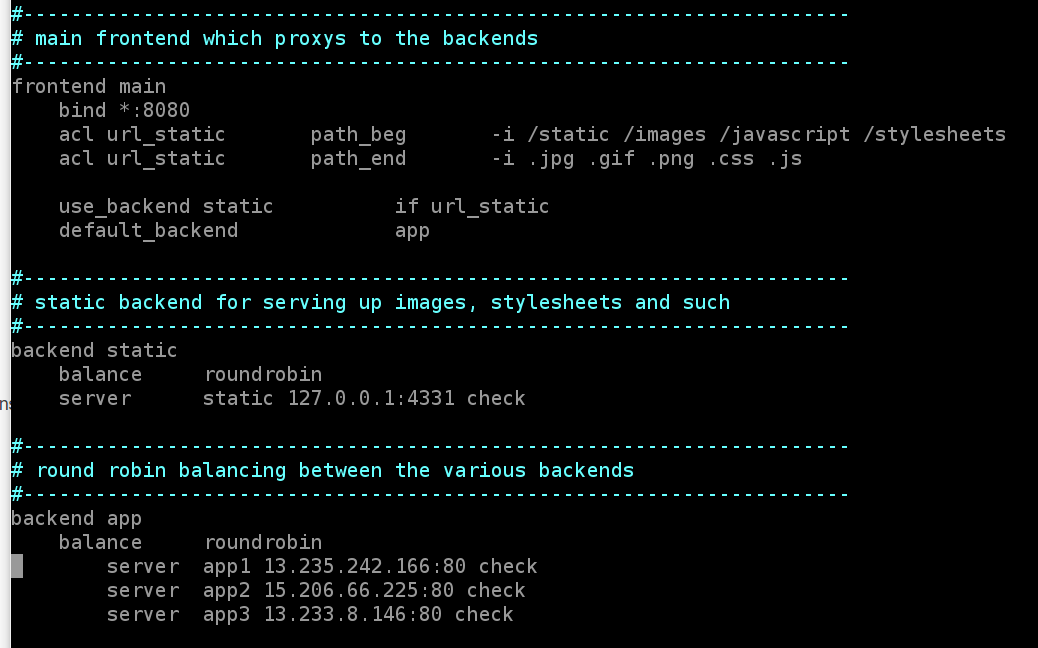
* **Launch the HAProxy playbook using:**
* **ansible-playbook <file\_name>.yml**
* **use verbose -v if you want to see more details:**



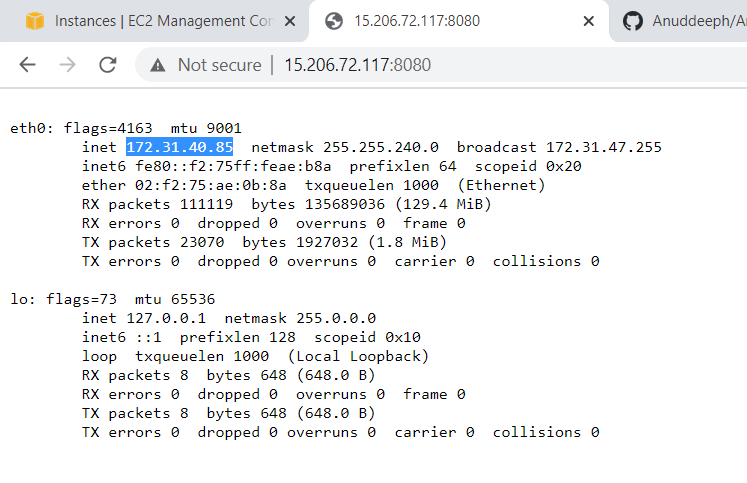
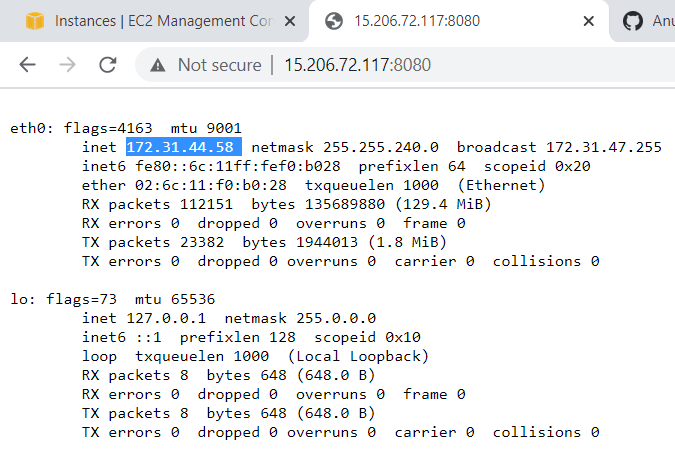
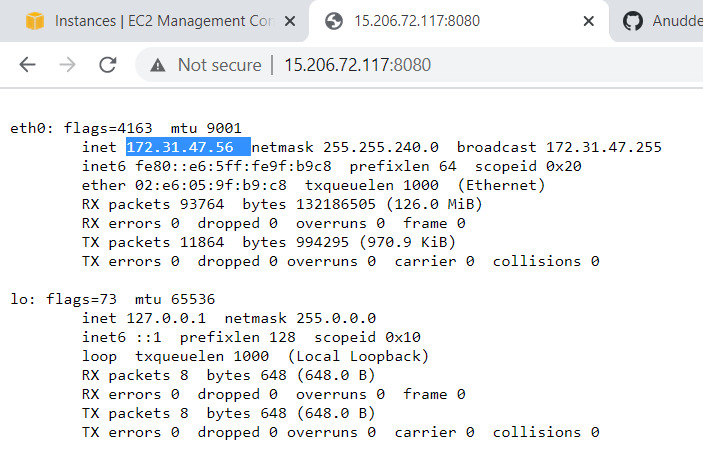


* Check the **haproxy.cfg** file is updated, in LB instance using:

Vim /etc/haproxy/haproxy.cfg



* You can see in the above screenshot, the **haproxy.cfg** file is updated.
* Let’s check the Load Balancer is working or not.

* Load Balancer is also working great with all three web servers.

GitHub link: https://github.com/Anuddeeph/Ansible\_AWS\_Reverse\_Proxy.git