# **Day 20 Task**

## **1. Inventory Plugins**

* **Activity**: Configure a dynamic inventory plugin to manage a growing number of web servers dynamically. Integrate the plugin with Ansible to automatically detect and configure servers in various environments.
* **Deliverable**: Dynamic inventory configuration file or script, demonstrating the ability to automatically update the inventory based on real-time server data.

ansible.cfg

[defaults]

inventory = /home/einfochips/Desktop/devops-training/day20-Ansible/inventory.ini

private\_key\_file = /home/einfochips/Downloads/ansible-shital.pem

remote\_user = ubuntu

host\_key\_checking = False

forks = 50

aws\_ec2.yml

plugin: amazon.aws.aws\_ec2

regions:

- us-west-2

filters:

instance-state-name: running

tag:Name:

- shital

hostnames:

- dns-name

compose:

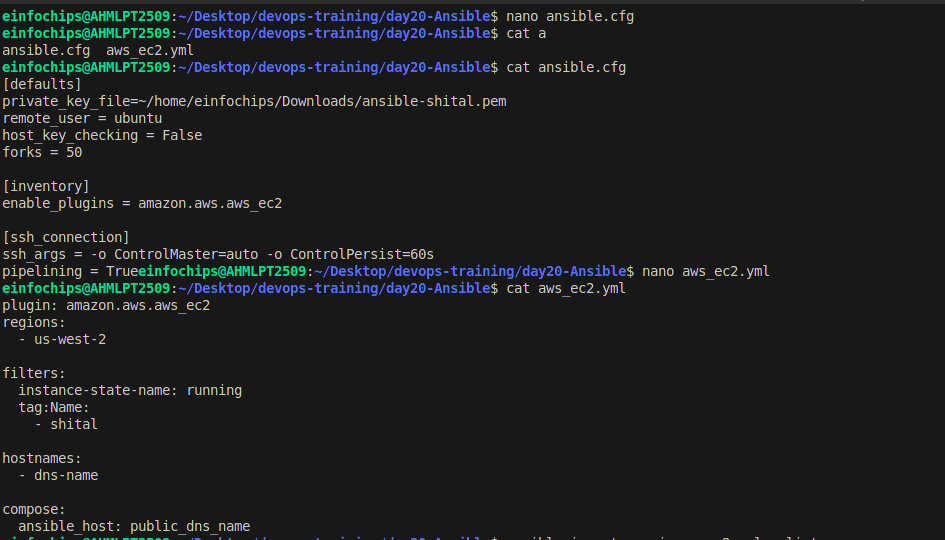
ansible\_host: public\_dns\_name

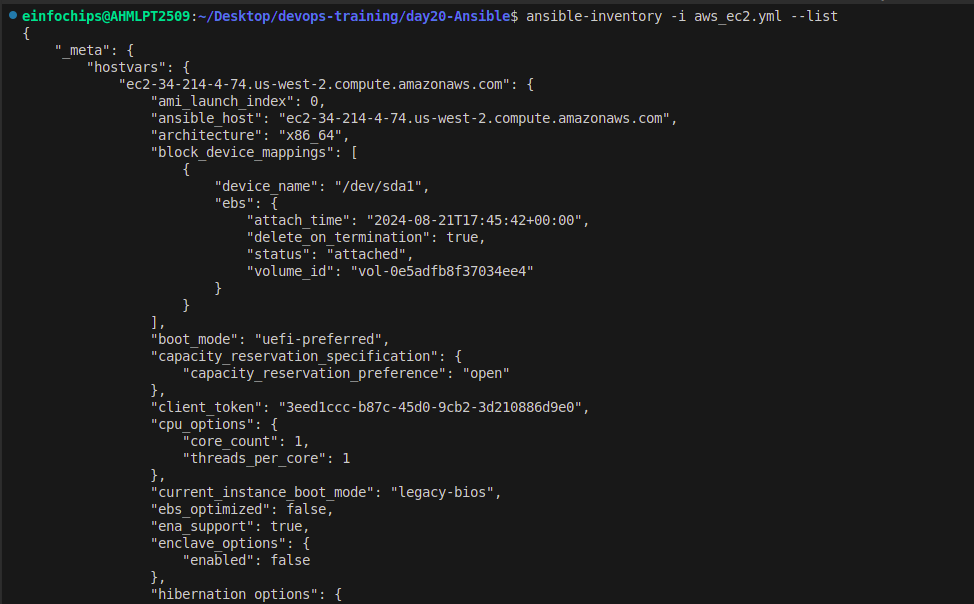
Commands:

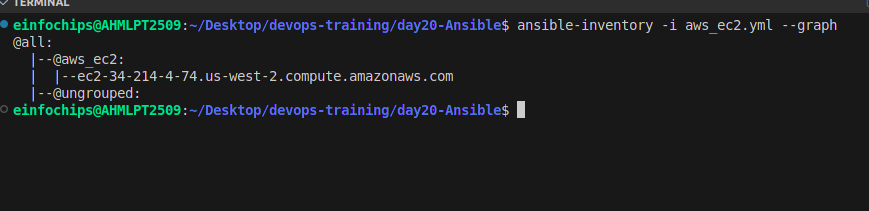
# to get detailed information about running instances

ansible-inventory -i aws\_ec2.yml --list

ansible-inventory -i aws\_ec2.yml --graph







## **2. Performance Tuning**

* **Activity**: Tune Ansible performance by adjusting settings such as parallel execution (forks), optimizing playbook tasks, and reducing playbook run time.
* **Deliverable**: Optimized ansible.cfg configuration file, performance benchmarks, and documentation detailing changes made for performance improvement.

[defaults]

forks = 50

[ssh\_connection]

ssh\_args = -o ControlMaster=auto -o ControlPersist=60s

pipelining = True

## **3. Debugging and Troubleshooting Playbooks**

* **Activity**: Implement debugging strategies to identify and resolve issues in playbooks, including setting up verbose output and advanced error handling.
* **Deliverable**: Debugged playbooks with enhanced error handling and logging, including a troubleshooting guide with common issues and solutions.

**Commands:**

# -v -> task result

ansible-playbook -i inventory.ini docker-playbook.yml -v

# -vv -> task input and output

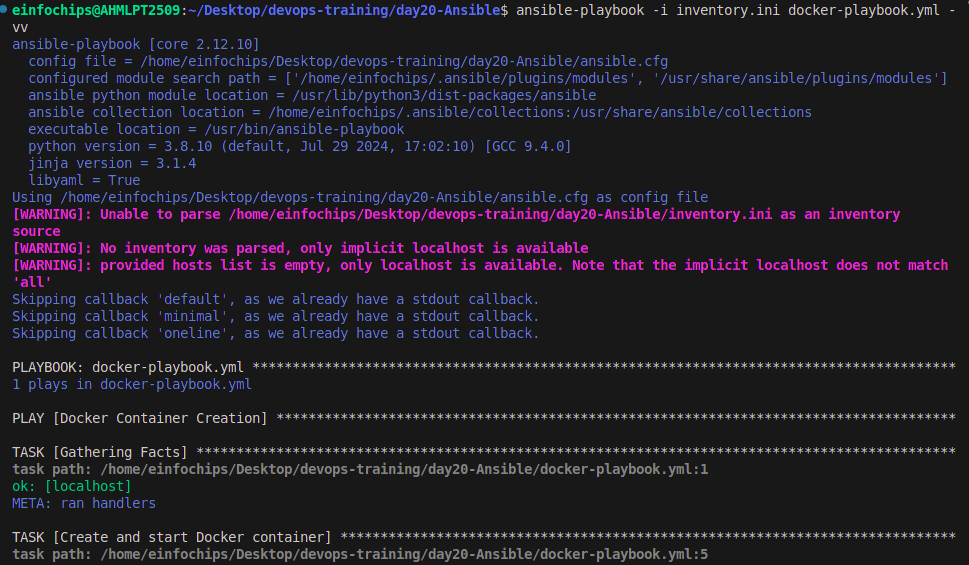
ansible-playbook -i inventory.ini docker-playbook.yml -vv

# -vvv -> task execution

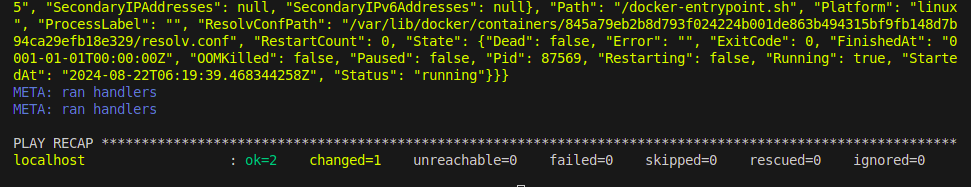
ansible-playbook -i inventory.ini docker-playbook.yml -vvv

# -vvvv -> ansible internal details

ansible-playbook -i inventory.ini docker-playbook.yml -vvvv







## **4. Exploring Advanced Modules**

* **Activity**: Use advanced Ansible modules such as docker\_container to manage containerized applications and aws\_ec2 for AWS infrastructure management, demonstrating their integration and usage.
* **Deliverable**: Playbooks showcasing the deployment and management of Docker containers and AWS EC2 instances, along with documentation on the benefits and configurations of these advanced modules.

**Docker Container Module Example :**

- name: Docker Container Creation

hosts: my\_group

become: yes

tasks:

- name: Install Docker

apt:

name: docker.io

state: present

update\_cache: yes

- name: Ensure Docker is running

service:

name: docker

state: started

enabled: yes

- name: Create and start Docker container

community.docker.docker\_container:

name: my\_nginx

image: shital37/my\_nginx\_image:latest

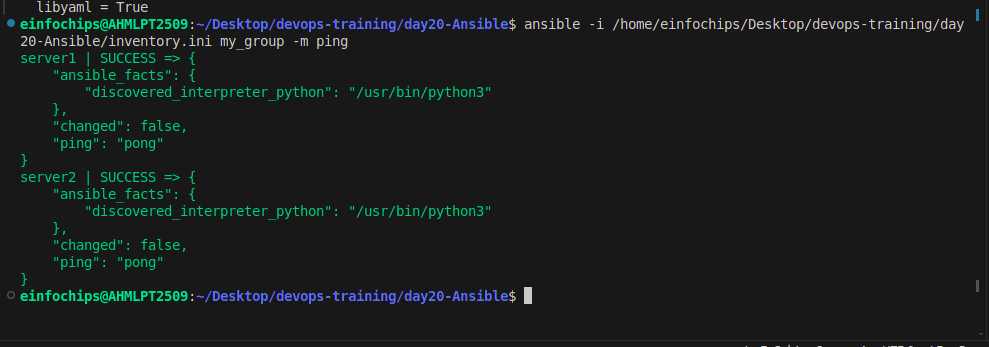
state: started

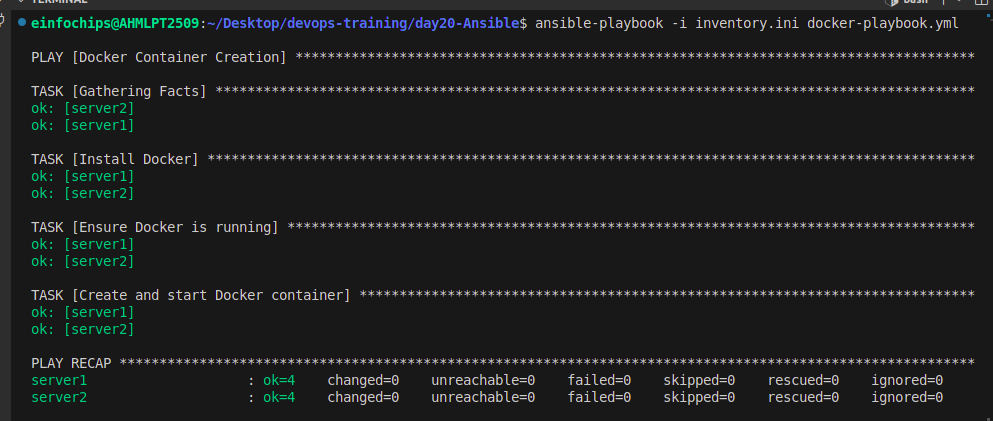
ports:

- "8084:80"

volumes:

- /my/local/path:/usr/share/nginx/html





**AWS ec2 Module Example :**

* Write a playbook to manage AWS EC2 instances using the aws\_ec2 module.
* Sample playbook for creating an security group:

- name: Launch an EC2 instance

hosts: localhost

collections:

- amazon.aws

tasks:

- name: Create security group

amazon.aws.ec2\_security\_group:

name: "new-security-group"

description: "Sec group for app"

rules:

- proto: tcp

ports:

- 22

cidr\_ip: 0.0.0.0/0

rule\_desc: allow all on ssh port

state: present

delegate\_to: localhost

