Ansible Collections

Why Ansible Collections Are Needed

Ansible Collections help organize and distribute modules, roles, and plugins in a structured way. Here's why they are needed:

- Structured Organization: Collections group related modules, plugins, and roles, making them easier to manage and use.
- Modular Approach: Instead of managing individual modules, Collections allow bundling them into reusable packages.
- Efficient Infrastructure Provisioning: When provisioning infrastructure via AWS APIs:
- The module runs on the control node instead of the managed node.
- It communicates with AWS using API keys, instead of running tasks via SSH.
- Simplifies Installation and Updates: Collections provide a streamlined way to install, update, and distribute Ansible modules.

General Ansible Workflow

- 1. Traditional Setup: The control node sends modules to the managed node and executes tasks over SSH.
- 2. Infrastructure Provisioning with AWS: The module remains on the control node and communicates with AWS using API keys.

AWS Collection Prerequisites

To provision infrastructure using Ansible on AWS, install the required dependencies:

1. Install boto3 (AWS SDK for Python):

COMMAND: pipx install boto3

2. Install the AWS Ansible Collection:

COMMAND: ansible-galaxy collection install amazon.aws

root@Sujal:/mnt/c/Users/91620/OneDrive/Desktop/DevOps/ansible/an
ansible-galaxy collection install amazon.aws
Starting galaxy collection install process
Nothing to do. All requested collections are already installed.

- 3. Generate AWS Credentials (Access Key & Secret Key):
- Go to AWS IAM Console → Users → Create Access Key.

Secure AWS Credentials Using Ansible Vault

1. Create a Vault Password File

Generate a secure password for Vault:

COMMAND: openssl rand-base64 2048 > vault.pass

root@Sujal:/mnt/c/Users/91620/OneDrive/Desktop/DevOps/ansible/anisble_collections/Ansible/Ansible_Collec tions# openssl rand -base64 2048 > vault.pass

2. Store AWS Credentials Securely

Use Ansible Vault to create a credentials file:

COMMAND: ansible-vault create group_vars/all/pass.yml --vault-password-file /root/vault.pass

root@Sujal:/mnt/c/Users/91620/OneDrive/Desktop/DevOps/ansible/anisble_collections/Ansible/Ansible_Collections# ansible-vault create group_vars/all/pass.yml --vault-password-file /root/vault.pass[WARNING]: group_vars/all does not exist, creating...

Inside pass.yml, store:

ec2_access_key: "YOUR_ACCESS_KEY" ec2_secret_key: "YOUR_SECRET_KEY"

Using Roles in Ansible

To maintain a structured setup, we use roles. The tasks for launching the EC2 instance are placed inside the tasks section under a role.

1. Create a Role

Run the following command to create a role named ec2:

COMMAND: ansible-galaxy init ec2

```
root@Sujal:/mnt/c/Users/91620/OneDrive/Desktop/DevOps/ansible/anisble_collections/Ansible/Ansibl
tions# ansible-galaxy role init ec2
- Role ec2 was created successfully
```

2. Define the Task in roles/ec2/tasks/main.yml

Inside roles/ec2/tasks/main.yml, add:

```
# tasks file for ec2
- name: start an instance with a public IP address
 amazon.aws.ec2 instance:
   name: "ansible-instance"
   # key_name: "prod-ssh-key"
    # vpc subnet id: subnet-013744e41e8088axx
    instance_type: t2.micro
   security_group: sg-0f1e60d41e7d112d0
   region: us-east-1
    vpc subnet id: subnet-0f3b8e79585b1ee3f
    aws_access_key: "{{ec2_access_key}}" # From vault as defined
   aws_secret_key: "{{ec2_secret_key}}" # From vault as defined
   network:
     assign public ip: true
    image id: ami-04b70fa74e45c3917
     Environment: Testing
```

3. Create the Playbook (ec2_create.yml)

This playbook will execute the ec2 role:

```
Ansible > Ansible_Collections > ML ec2_create.yml > {}

1 ---
2 - hosts: localhost
3 | connection: local
4 | roles:
5 | - ec2
6
```

4. Run the Playbook

Execute the playbook with:

COMMAND: ansible-playbook -i inventory.ini ec2_create.yml --vault-password-file /root/vault.pass

root@Sujal:/mnt/c/Users/91620/OneDrive/Desktop/DevOps/ansible/anisble_collections/Ansible/Ansible_Collec tions# ansible-playbook -i inventory.ini ec2_create.yml --vault-password-file /root/vault.pass

Output

```
root@Sujal:/mnt/c/Users/91620/OneDrive/Desktop/DevOps/ansible/ansible_collections/Ansible_Collections# ansible-playbook -i inventory.ini ec2_create.yml --vault-pas sword-file /root/vault.pass

ile /root/vault.pass

PLAY [localhost]

TASK [Gathering Facts]

TASK [Gathering Facts]

*********

ok: [localhost]

TASK [ec2 : start an instance with a public IP address]

TASK [ec2 : start an instance with a public IP address]

*********

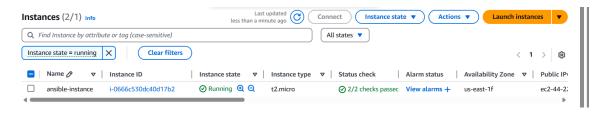
[DEPRECATION WARNING]: The network parameter has been deprecated, please use network_interfaces and/or network_interfaces_ids instead. This feature will be removed from amazon.aws in a release after 2026-12-01. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.

PLAY RECAP

*********

| Dealbost | oks2 | changed=0 | unreachable=0 | failed=0 | skinged=0 | prescued=0 | ignored=0 | ignored=0
```

Instance Launched



Summary

- 1. Installed boto3 and AWS Ansible Collection.
- 2. Generated AWS Access Key and Secret Key.
- 3. Secured credentials using Ansible Vault.
- 4. Created an Ansible role for EC2 provisioning.
- 5. Defined tasks inside the role's tasks/main.yml.
- 6. Used the role inside a playbook (ec2_create.yml).
- 7. Retrieved the public IP of the EC2 instance.