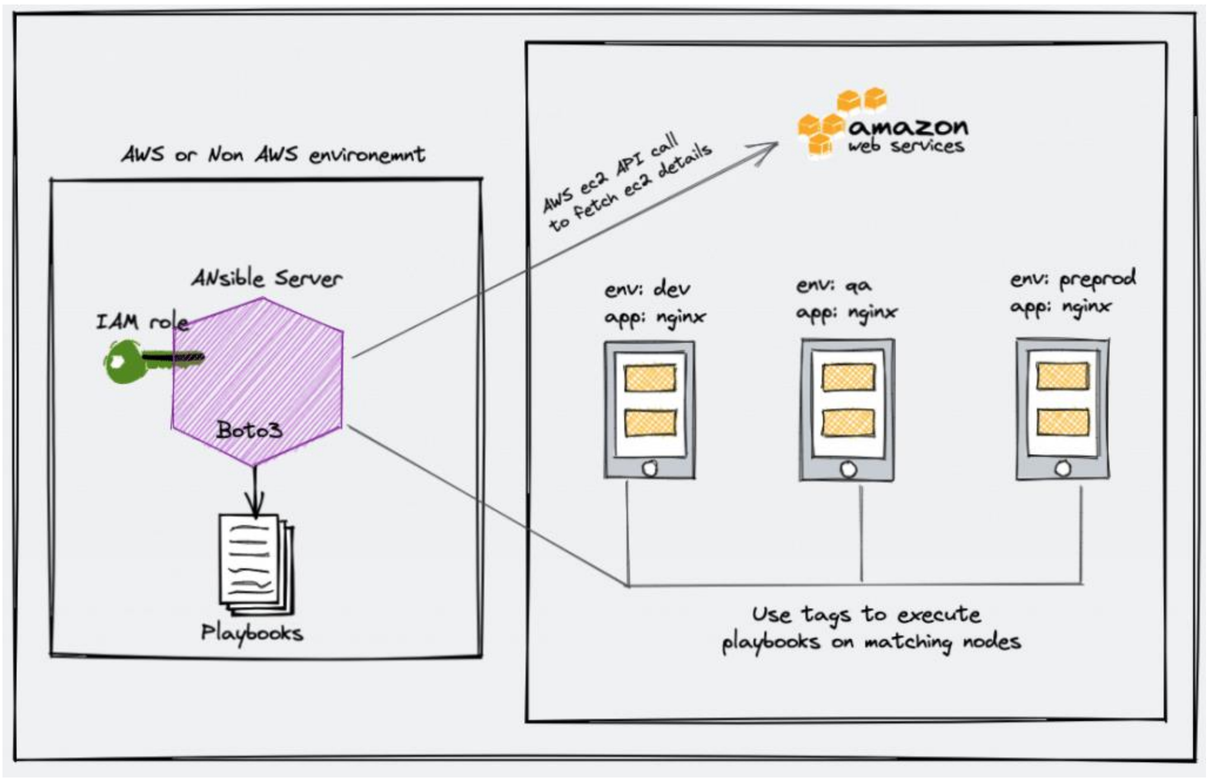
Working With Dynamic Inventory

To working with AWS dynamic inventory, we need **boto3** and **botocore** python modules.

https://docs.ansible.com/ansible/latest/collections/amazon/aws/aws_ec2_inventory.html



- Get inventory hosts from Amazon Web Services EC2.
- Uses a YAML **configuration file that ends with** `aws_ec2.(yml|yaml`).

Requirements

The below requirements are needed on the local controller node that executes this inventory.

- boto3
- botocore

First, install **python3** if you haven’t installed it yet.

```
$ sudo yum install -y python3
```

Install “**boto3**”

```
$ pip3 install --user boto3

$ pip3 install --upgrade requests --user
```

Create a file named `inventory_aws_ec2.yml` in the project directory.

Note: The file name needs to be ended with **aws_ec2.yaml/yml**.

```
$ vi inventory_aws_ec2.yml
```

Paste the content below into the **inventory_aws_ec2.yml** file. As you see that this file begins with defined the plugin: **aws_ec2**.

Note: In this example, I added one tag to the target nodes (via AWS Console) “Name” to groups them. And use filter to see only **running** instances.

```
plugin: aws_ec2

regions:
  - ap-south-1

filters:
  instance-state-name : running

keyed_groups:
  - key: tags.Name
    prefix: ""
    separator: ""

hostnames:
  - private-ip-address

compose:
  ansible_host: private_ip_address
```

But at this point, the Control node needs authentication to access the AWS resources.

If you want, you can add your AWS access key and secret to the config file.

But I think it is not a safer way, and I prefer to use the **IAM role** instead. So Ansible will automatically use this role to make the AWS API calls.

Step 3: Add An IAM Role And Attached It To Control Node

At **AWS Console**, go to **Identity and Access Management (IAM)** service and click the "**Create role**" button and then create a role with "**AmazonEC2ReadOnlyAccess**".

After that, we need to attach this role with the Control node.

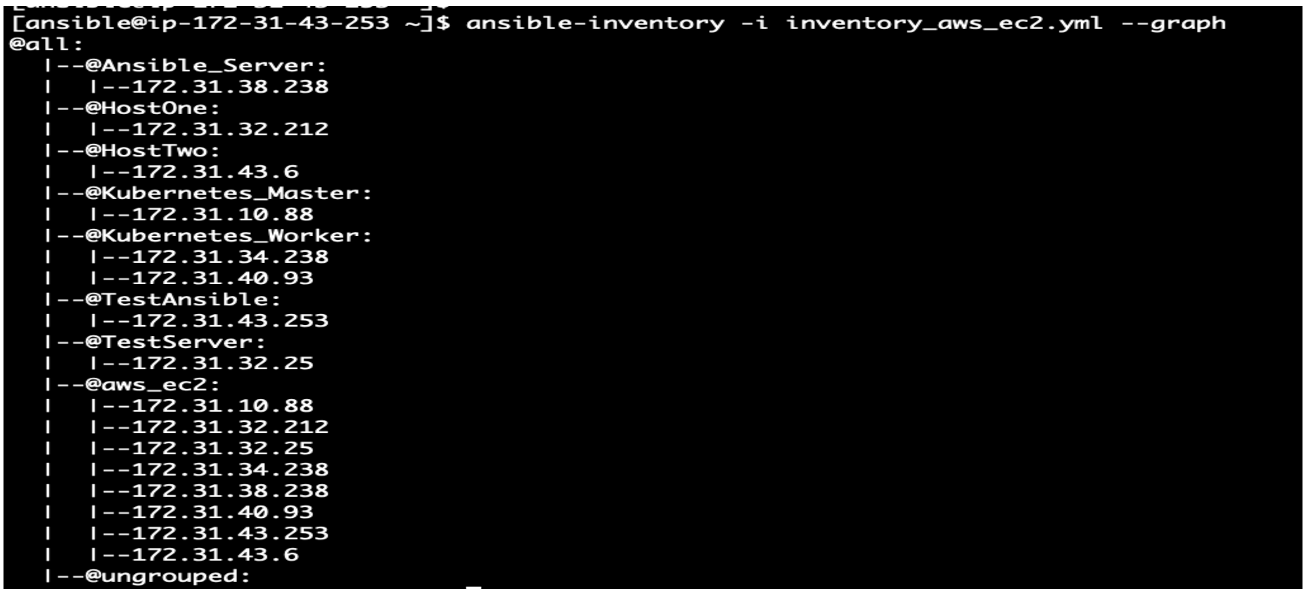
- Go to **EC2 Dashboard**, and select the control-node instance
- Select "**actions" → "security" → "modify IAM role"**
- Select the role that has "**AmazonEC2ReadOnlyAccess**" and **save** it.

Step 4: Pinging The Target Nodes With Dynamic Inventory

First, check the inventory.

Note: We will use the "**-i"** flag to refer to the **inventory_aws_ec2.yml** file because we haven't changed the inventory variable in the config file yet.

```
$ ansible-inventory --graph -i inventory_aws_ec2.yml
```



```
[ansible@ip-172-31-43-253 ~]$ ansible-inventory -i inventory_aws_ec2.yml --graph
@all:
  |--@Ansible_Server:
  |  |--172.31.38.238
  |--@HostOne:
  |  |--172.31.32.212
  |--@HostTwo:
  |  |--172.31.43.6
  |--@Kubernetes_Master:
  |  |--172.31.10.88
  |--@Kubernetes_Worker:
  |  |--172.31.34.238
  |  |--172.31.40.93
  |--@TestAnsible:
  |  |--172.31.43.253
  |--@TestServer:
  |  |--172.31.32.25
  |--@aws_ec2:
  |  |--172.31.10.88
  |  |--172.31.32.212
  |  |--172.31.32.25
  |  |--172.31.34.238
  |  |--172.31.38.238
  |  |--172.31.40.93
  |  |--172.31.43.253
  |  |--172.31.43.6
  |--@ungrouped:
```

Using Dynamic Inventory Inside Playbook

If you want to use dynamic inventory inside the playbook, you just need to mention the group name in the hosts variable as shown below.

```
- hosts: HostOne
  gather_facts: false
  tasks:
    - name: Run Shell Command
      command: echo "Hello World"
```

and execute like

```
$ ansible-playbook -i inventory_aws_ec2.yml <playbookName>.yaml


$ ansible-playbook -i inventory_aws_ec2.yml <playbookName>.yaml -u=
<username> --private-key=<PemFilePath>.pem


$ ansible-playbook -i inventory_aws_ec2.yml <playbookName>.yaml -
u=<username> --private-key=<PemFilePath>.pem -l <groupName>
```