



ANSIBLE



Amazon  
EC2

# Task 2

Statement : Deploy Web Server on AWS through ANSIBLE!

- ◆ Provision EC2 instance through ansible.
- ◆ Retrieve the IP Address of instance using dynamic inventory concept.
- ◆ Configure the web server through ansible!
- ◆ Create role for webserver to customize the Instance and deploy the webpage to root directory.

So let start this task 2 :-

Ansible is great tool in doing configuration on any OS. What we have to launch the OS using Ansible. So this can be possible using Ansible. But ansible is meant for configuration and for provisioning OS we cab use Terraform. Though Ansible can manage Configuration as well as provisioning there i am using ansible to provision ec2 instance and also for configuring webserver inside that ec2 instance. The task description can be cleared using below figure.

That we can use our localhost ip address to behave as a managed node and we will use the SDK to launch ec2 instance on AWS as Ansible is built on python language so we will be using boto3. Boto as it an API so it has the capability to contact to AWS.

**pip3 install boto**

So we have to insert localhost IP address in hosts of ansible.

```
[localhost]  
127.0.0.1      ansible_ssh_pass=pratik      ansible_ssh_pass=pratik
```

After that try to ping using the ping command. and then try via ansible localhost -m ping.

Now we are ready to create ansible-playbook. Ansible Playbook is a file where we have to write play's i.e a book of play's means what you would like to configure or provision then we have to write code in that file. Normally ansible-playbook supports YAML language. so we have to create a playbook.

```
- hosts: "localhost"
vars_files:
  - secret_key.yml
tasks:
- name: "provisioning OS on AWS using Ansible"
  ec2:
    key_name: "openstack"
    instance_type: "t2.micro"
    image: "ami-052c08d70def0ac62"
    wait: yes
    count: 1
    instance_tags:
      Name: aws_ansible_ec2
    vpc_subnet_id: "subnet-3c161654"
    assign_public_ip: yes
    region: "ap-south-1"
    state: present
    group_id: "sg-06f34c397d23b4c97"
    aws_access_key: "{{aws_access_key}}"
    aws_secret_key: "{{aws_secret_key}}"
  register: X

- debug:
  var: X.instances[0].private_ip
  register: IP

- debug:
  var: IP["X.instances[0].private_ip"]

~
~
~
"aws_ec2.yml" 29L, 709C
```

Extraction of the IP of newly launched EC2 instance.

```
[root@Ansible ansible_yask2]# ansible-playbook --vault-id ec2_launch@prompt aws_ec2.yml
Vault password (ec2_launch):

PLAY [localhost] *****

TASK [Gathering Facts] *****
ok: [127.0.0.1]

TASK [provisioning OS on AWS using Ansible] *****
changed: [127.0.0.1]

TASK [debug] *****
ok: [127.0.0.1] => {
  "X.instances[0].private_ip": "172.31.34.43"
}

TASK [debug] *****
ok: [127.0.0.1] => {
  "IP[\"X.instances[0].private_ip\"]": "172.31.34.43"
}

PLAY RECAP *****
127.0.0.1 : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[root@Ansible ansible_yask2]#
```

After Running code Successfully. we can see below that the new ec2 instance named "ansible\_ec2" is has been launched.

The screenshot displays the AWS Management Console interface. On the left, the navigation menu includes options like 'New EC2 Experience', 'EC2 Dashboard', 'Events', 'Tags', 'Limits', 'Instances', 'Images', and 'Elastic Block Store'. The main content area shows a table of EC2 instances. One instance, named 'ansible\_ec2', is listed with the following details:

NAME	Name	App	Enviroment	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
<input checked="" type="checkbox"/>	ansible_ec2			i-0884d1549390c0c9e	t2.micro	ap-south-1a	running	2/2 checks ...

Below the table, the 'ansible\_ec2' instance details are expanded, showing various attributes such as 'Private DNS', 'Private IPs' (172.31.34.43), 'Secondary private IPs', 'VPC ID' (vpc-e722398f), 'Subnet ID' (subnet-3c161654), 'Network interfaces' (eth0), and 'IAM role' (-). A tooltip is visible over the 'AMI ID' field, which reads: 'The ID of the AMI with which the instance was launched.' The AMI ID is 'RHEL-8.2.0\_HVM-20200423-x86\_64-0-Hourly2-GP2 (ami-052c08d70def0ac62)'.

The screenshot displays the AWS Management Console interface. On the left, the navigation menu includes sections for EC2 Dashboard, Events, Tags, Limits, Instances, Images, Elastic Block Store, and Network & Security. The main content area shows a list of EC2 instances with columns for NAME, Name, App, Environment, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, Alarm Status, Public DNS (IPv4), IPv4 Public IP, and IPv6 IPs. One instance, 'ansible\_ec2', is listed with ID 'i-0884d1549390c0c9e', type 't2.micro', in 'ap-south-1a' availability zone, and state 'running'. Below the list, the details for this instance are shown, including its Public DNS, IPv4 Public IP (3.6.40.88), and various configuration options like Elastic IPs, Availability zone, and Security groups.

Now we have to set the user name and password of the newly ec2 instance launched and also we have to set the hostname.

```
Last login: Wed Aug 19 05:45:08 UTC 2020 on pts/1
[root@ec2 ~]# useradd pratik
[root@ec2 ~]# passwd pratik
Changing password for user pratik.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
[root@ec2 ~]#
```

Updation of the user in /etc/sudoers

```
ec2-user      ALL=(ALL)      NOPASSWD: ALL
pratik        ALL=(ALL)      NOPASSWD: ALL
```

we need to give password authentication in `/etc/ssh/sshd_config`. and restart the sshd services.

```
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no
#PasswordAuthentication no
```

Restart the services using `service sshd restart`.

Then we have to generate the ssh key. using the command `ssh-keygen`.



```

[ec2-user@Ansible ~]$ ssh-keygen
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:uNq7WuGH2w2/2Q1lrYgDdrjl4HFC500aAcSMAzOtUY4 ec2-user@Ansible
The key's randomart image is:
+---[RSA 3072]---+
|      ++.=o..      |
|      .=+ o   .    |
|      Eo... o   .   |
|      .  o + =     . |
|      o S * . o .  |
|      . * @ . + .   |
|      = = + o .     |
|      + + + + o     |
|      o.=o. =.. .   |
+-----[SHA256]-----+
[ec2-user@Ansible ~]$ █

```

After generation of a sshkey we have to copy the sshkey in ec2 instance. using the command `ssh-copy-id`.

```

pratik@172.31.34.43 's password:
Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'pratik@172.31.34.43'"
and check to make sure that only the key(s) you wanted were added.

```

After copying ssh key to newly launched ec2 instance then we have to update the private IP address in the ansible hosts.

write the access permission code in the ansible.cfg file.

```
[defaults]
inventory = /etc/myhosts.txt
host_key_checking=False
command_warnings=false
remote_user = pratik
ask_pass = false

[privilege_escalation]
become = true
become_method = sudo
become_user = root
become_ask_pass = false

~
~
```

```
[localhost]
127.0.0.1      ansible_ssh_pass=pratik      ansible_ssh_pass=pratik

[ec2]
172.31.34.43
```

Here no need to give password as we have already generated ssh key and copied to the desired ec2 instance. Now we can ping the ec2 Node.



```
[ec2-user@Ansible ~]$ ansible ec2 -m ping
172.31.34.43 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/libexec/platform-python"
  },
  "changed": false,
  "ping": "pong"
}
[ec2-user@Ansible ~]$
```

We have to configure the web server in that ec2 node with the help of ansible.

```
- hosts: "ec2"
  tasks:
    - package:
        name: "httpd"
        state: present
    - copy:
        src: "index.html"
        dest: "/var/www/html"
    - service:
        name: "httpd"
        state: restarted
```

~

~

~

The above code will configure the webserver in that ec2 node.

```
[root@Ansible ansible_yask2]# ansible-playbook ec2.yml

PLAY [ec2] *****

TASK [Gathering Facts] *****
ok: [172.31.34.43]
```

```
[root@Ansible ansible_yask2]# ansible-playbook ec2.yml

PLAY [ec2] *****

TASK [Gathering Facts] *****
ok: [172.31.34.43]

TASK [package] *****
changed: [172.31.34.43]

TASK [copy] *****
[WARNING]: File '/var/www/html/index.html' created with default permissions '600'. The previous default was '666'.
Specify 'mode' to avoid this warning.
changed: [172.31.34.43]

TASK [service] *****
changed: [172.31.34.43]

PLAY RECAP *****
172.31.34.43 : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Now we see our code run successfully now you can check the webserver using public IP of that ec2 instance.

# Congratulations on your success

## Welcome to the second Task of Ansible

Launching EC2 instance on AWS using ansible-playbook

**The webserver is successfully configured !!!**

**Thanks for reading!!!**