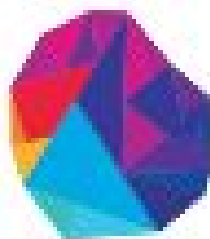


ASSESSMENT

ON:

ANSIBLE (2)

**TO
THE
NEW.**



1. Create two nodes with tag:key role and tag:value master & slave respectively. Setup the dynamic inventory on ansible control nodes.

STEP 1: Launch an ec2 instance

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic or add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one.

Assign a security group: ☒ Create a new security group ☐ Select an existing security group

Security group name:

Description:

Type <small>i</small>	Protocol <small>i</small>	Port Range <small>i</small>
SSH	TCP	22
All ICMP - IPv4	ICMP	0 - 65535

STEP 2: SSH into the instance and create a ssh key and add it to master and slaves authorised_key file.

<input type="checkbox"/>	Name	Instance ID	Instance Type
<input checked="" type="checkbox"/>	Master	i-05114e2c32a886160	t2.micro

STEP 3: Now launch another instance from that ami.

Launch Instance Actions

Key Name : diksha_awskey2

<input type="checkbox"/>	Name	Instance ID	Availability Zone	Instance State	Status Check
<input checked="" type="checkbox"/>	Master	i-05114e2c32a886160	us-east-1c	running	2/2 checks passed
<input type="checkbox"/>	Slave	i-07e88e72e6fb0b1c	us-east-1c	terminated	
<input type="checkbox"/>	Master	i-0ea862b94b7a1c	us-east-1c	running	

Connect

Get Windows Password

Create Template From Instance

Launch More Like This

Instance State

Instance Settings

Image

Networking

CloudWatch Monitoring

Create Image

Bundle Instance (instance store AMI)

Step 1: Choose an Amazon Machine Image (AMI)

[Cancel and Exit](#)


An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start (0)

My AMIs (1)

AWS Marketplace (0)


Community AMIs (0)



ansible_masterimage - ami-0f76ccc29a37cf6a8
ansible
Root device type: ebs Virtualization type: hvm Owner: 187632318301 ENA Enabled: Yes

Select


64-bit (x86)

 **ansible_slave**

i-0faa95ee7a90d0102

t2.micro

us-east-1c

 **running**

STEP 4: Then create a role for ec2 full access and attach it to the ansible-master node.

Create role

Select type of trusted entity

 **AWS service**
EC2, Lambda and others

 **Another AWS user**
Belonging to you

Allows AWS services to perform actions on your behalf. [Learn](#)

Choose a use case

Common use cases

EC2

Allows EC2 instances to call AWS services on your behalf.

Create role

Review

Provide the required information below and review this role before you create it.

Role name*

Use alphanumeric and '+=,._@-_' characters. Maximum 64 characters.

Role description

Maximum 1000 characters. Use alphanumeric and '+=,._@-_' characters.

Trusted entities AWS service: ec2.amazonaws.com

Policies  AmazonEC2FullAccess [↗](#)

Permissions boundary Permissions boundary is not set

Attach/Replace IAM Role

Select an IAM role to attach to your instance. If you don't have any IAM roles, choose Create new IAM role. If an IAM role is already attached to your instance, the IAM role you choose will replace the existing role.

Instance ID i-05114e2c32a886160 (Master) ⓘ

IAM role* masterec2full ▼ ↻

STEP 5: Download the ec2.py and ec2.ini files and put them in /etc/ansible directory and give ec2.py the execute permissions.

```
ubuntu@ip-172-31-46-248:/etc/ansible$ ls
ansible.cfg ec2.ini ec2.py hosts
ubuntu@ip-172-31-46-248:/etc/ansible$ sudo chmod +x ec2.py
```

STEP 6: Set RDS and ElastiCache to false in ec2.ini

```
# To exclude RDS instance
rds = False
elasticsearch = False
```

* Pre-requisites:

- **Boto** must be installed on the master and slave server to make AWS API Calls.
- Ansible servers should be able to do SSH on nodes.

Run the following modules using tag key-value:

1.1 Ping master node and slave node separately.

```
ubuntu@ip-172-31-46-248:/etc/ansible$ ansible -i ec2.py tag_role_master -m ping
[ERROR]:

The authenticity of host '3.92.50.160 (3.92.50.160)' can't be established.
ECDSA key fingerprint is SHA256:egitsuiYsttgWR6bizjlvEOAflid0DamCl0M9gFxd4s.
Are you sure you want to continue connecting (yes/no)? 18.208.156.61 | SUCCESS =
> {
  "changed": false,
  "ping": "pong"
}
```

```
ubuntu@ip-172-31-46-248:/etc/ansible$ ansible -i ec2.py tag_role_slave -m ping
[ERROR]:
3.92.225.200 | UNREACHABLE! => {
  "changed": false,
  "msg": "Failed to connect to the host via ssh: ubuntu@3.92.225.200: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).\r\n",
  "unreachable": true
}
3.83.42.122 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

1.2 To check all running processes on slave node.

```
ubuntu@ip-172-31-46-248:/etc/ansible$ ansible -i ec2.py tag_role_slave -m shell
-a "ps -aux"
[ERROR]:
3.92.225.200 | UNREACHABLE! => {
  "changed": false,
  "msg": "Failed to connect to the host via ssh: ubuntu@3.92.225.200: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).\r\n",
  "unreachable": true
}
3.83.42.122 | SUCCESS | rc=0 >>
USER      PID  %CPU  %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1   0.0   0.9 225408  9220 ?        Ss   16:27   0:05 /lib/systemd/sy
stemd --system --deserialize 19
root         2   0.0   0.0      0     0 ?        S    16:27   0:00 [kthreadd]
root         4   0.0   0.0      0     0 ?        I<   16:27   0:00 [kworker/0:0H]
root         6   0.0   0.0      0     0 ?        I<   16:27   0:00 [mm_percpu_wq]
root         7   0.0   0.0      0     0 ?        S    16:27   0:00 [ksoftirqd/0]
root         8   0.0   0.0      0     0 ?        I    16:27   0:00 [rcu_sched]
root         9   0.0   0.0      0     0 ?        I    16:27   0:00 [rcu_bh]
root        10   0.0   0.0      0     0 ?        S    16:27   0:00 [migration/0]
root        11   0.0   0.0      0     0 ?        S    16:27   0:00 [watchdog/0]
```

1.3 To copying files to both nodes concurrently.

```
ubuntu@ip-172-31-46-248:/etc/ansible$ ansible -i ec2.py ec2 -m copy -a "src=/etc
/hosts dest=/home/ubuntu"
```



```
18.208.156.61 | SUCCESS => {
  "changed": true,
  "checksum": "a777be51c79c607edd61c8e12cd9b775ddc8a6c6",
  "dest": "/home/ubuntu/hosts",
  "gid": 1000,
  "group": "ubuntu",
  "md5sum": "1463508f28edb4d6d5ae349b20e00409",
  "mode": "0664",
  "owner": "ubuntu",
  "size": 221,
  "src": "/home/ubuntu/.ansible/tmp/ansible-tmp-1586024595.55-6373230439234/source",
  "state": "file",
  "uid": 1000
}
```

```
3.83.42.122 | SUCCESS => {
  "changed": true,
  "checksum": "a777be51c79c607edd61c8e12cd9b775ddc8a6c6",
  "dest": "/home/ubuntu/hosts",
  "gid": 1000,
  "group": "ubuntu",
  "md5sum": "1463508f28edb4d6d5ae349b20e00409",
  "mode": "0664",
  "owner": "ubuntu",
  "size": 221,
  "src": "/home/ubuntu/.ansible/tmp/ansible-tmp-1586024597.19-203267895781573/source",
  "state": "file",
  "uid": 1000
}
```

```
ubuntu@ip-172-31-218-181:~$ ls
hosts
ubuntu@ip-172-31-46-248:~$
```

```
ubuntu@ip-172-31-218-181:~$ ls
hosts
ubuntu@ip-172-31-218-181:~$
```

2. Setup nginx on both nodes with a single custom configuration template, on master nginx should run on 8000 while on slave nginx would listen on port 80. [Jinja2+conditional]

STEP 1: Create a role named “custom”

```
ubuntu@ip-172-31-46-248:~$ cd /etc/ansible/
ubuntu@ip-172-31-46-248:/etc/ansible$ sudo mkdir roles
ubuntu@ip-172-31-46-248:/etc/ansible$ cd roles/
ubuntu@ip-172-31-46-248:/etc/ansible/roles$ ls
ubuntu@ip-172-31-46-248:/etc/ansible/roles$ sudo ansible-galaxy init /etc/ansible/roles/custom
- /etc/ansible/roles/custom was created successfully
ubuntu@ip-172-31-46-248:/etc/ansible/roles$
```

```
ubuntu@ip-172-31-46-248:/etc/ansible/roles$ tree custom/
custom/
├── README.md
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml

8 directories, 8 files
```

STEP 2: Create a template for nginx configuration file

```
ubuntu@ip-172-31-46-248:/etc/ansible/roles$ cat custom/templates/nginx_temp.j2
{% if ec2_tag_role == 'master' %}
server{
    listen 8000;
    server_name _;
    index index.nginx-debian.html index.html;
    root /var/www/html;
}
{% else %}
server{
    listen 80;
    server_name _;
    index index.nginx-debian.html index.html;
    root /var/www/html;
}
{% endif %}
```

```
ubuntu@ip-172-31-46-248:/etc/ansible/roles$ cat custom/tasks/main.yml
---
# tasks file for /etc/ansible/roles/custom
- name: Updating apt
  apt:
    update_cache: yes
- name: Installing nginx
  apt:
    name: nginx
    state: latest
- name: Copy custom conf files (use template)
  templates:
    src: nginx_temp.j2
    dest: /etc/nginx/sites-available/nginx
- name: Remove unnecessary file in sites-enabled
  file:
    state: absent
    path: "/etc/nginx/sites-enabled/default"
- name: Creating soft link
  raw: test -L /etc/nginx/sites-enabled/nginx || ln -s /etc/nginx/sites-available/nginx /etc/nginx/sites-enabled/nginx
- name: Reloading service
  service:
    name: nginx
    state: reloaded
```

STEP 3: Create Playbook

```
ubuntu@ip-172-31-46-248:/etc/ansible/roles$ cat playbook.yml
- name: PLAYBOOK
  hosts: ec2
  become: yes
  gather_facts: yes
  roles:
```

STEP 4: Run playbook

```
ubuntu@ip-172-31-46-248:/etc/ansible/roles$ ansible-playbook -i /etc/ansible/ec2.py playbook.yml
[ERROR]:

PLAY [PLAYBOOK] *****

TASK [Gathering Facts] *****
The authenticity of host '3.83.220.82 (3.83.220.82)' can't be established.
ECDSA key fingerprint is SHA256:1yl608UMss5W4Vz8kKsgZcIQHGseJLKtmn89XmN39Ho.
Are you sure you want to continue connecting (yes/no)? ok: [3.83.42.122]
ok: [18.208.156.61]
```



```

TASK [custom : Updating apt] *****
changed: [3.83.42.122]
changed: [18.208.156.61]

TASK [custom : Installing nginx] *****
changed: [3.83.42.122]
changed: [18.208.156.61]

TASK [custom : Copy custom conf files (use template)] *****
changed: [3.83.42.122]
changed: [18.208.156.61]

TASK [custom : Remove unnecessary file in sites-enabled] *****
changed: [3.83.42.122]
changed: [18.208.156.61]

TASK [custom : Creating soft link] *****
changed: [18.208.156.61]
changed: [3.83.42.122]

TASK [custom : Reloading service] *****
[WARNING]: Consider using the service module rather than running service. If you need to use command because service
is insufficient you can add warn=False to this command task or set command_warnings=False in ansible.cfg to get rid of
this message.

changed: [3.83.42.122]
changed: [18.208.156.61]
[WARNING]: Could not create retry file '/etc/ansible/roles/playbook.retry'. [Errno 13] Permission denied:
u'/etc/ansible/roles/playbook.retry'

PLAY RECAP *****
18.208.156.61      : ok=7    changed=6    unreachable=0    failed=0
3.83.220.82      : ok=0    changed=0    unreachable=1    failed=0
3.83.42.122      : ok=7    changed=6    unreachable=0    failed=0

```

STEP 5: Check for file on master node

```

ubuntu@ip-172-31-46-248:/etc/ansible/roles$ cd ../../nginx/sites-enabled/
ubuntu@ip-172-31-46-248:/etc/nginx/sites-enabled$ ls
nginx
ubuntu@ip-172-31-46-248:/etc/nginx/sites-enabled$ cat nginx
server{
    listen 8000;
    server_name _;
    index index.nginx-debian.html index.html;
    root /var/www/html;
}

```

Welcome to nginx!

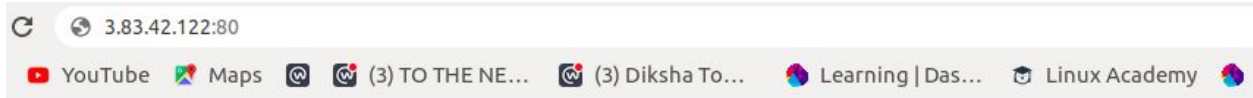
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

STEP 6: Similarly on slave node

```
ubuntu@ip-172-31-218-181:~$ cd /etc/nginx/sites-enabled
ubuntu@ip-172-31-218-181:/etc/nginx/sites-enabled$ ls
nginx
ubuntu@ip-172-31-218-181:/etc/nginx/sites-enabled$ cat nginx
server{
    listen 80;
    server_name _;
    index index.nginx-debian.html index.html;
    root /var/www/html;
}
```



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

3. Setup mysql on a remote server, create a user with password. Passwords should be encrypted using Ansible vault. Verify the setup by log in to mysql.

STEP 1: Create password

```
ubuntu@ip-172-31-46-248:/etc/ansible$ sudo ansible-vault encrypt_string diksha
New Vault password:
Confirm New Vault password:
!vault |
     $ANSIBLE_VAULT;1.1;AES256
     36356339616139623230646264653934653333353635323335366362363935376566306532613263
     3464336137336531666538393165643461616530303462660a633436316339653664633566303562
     64346261336533383534633834336563343663323862346139383662363132633537623338646532
     6464613464393961390a663233356337306430386230343634633336616663323331346261323165
     3262
Encryption successful
```

STEP 2: Make entry of both master and slave in hosts file

```
ubuntu@ip-172-31-46-248: /etc/ansible
File Edit View Search Terminal Tabs Help
ubuntu@ip-172-31-46-248: /etc... x ubuntu@ip-172-31-218-181: /et... x diksha@diksha: /etc/ansible/
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
# - Comments begin with the '#' character
# - Blank lines are ignored
# - Groups of hosts are delimited by [header] elements
# - You can enter hostnames or ip addresses
# - A hostname/ip can be a member of multiple groups
#
# Ex 1: Ungrouped hosts, specify before any group headers.
ubuntu@18.208.156.61
ubuntu@3.83.42.122
```

STEP 3: Create a playbook

```
ubuntu@ip-172-31-46-248:/etc/ansible$ cat mysqlplaybook.yml
- hosts: all
  gather_facts: no
  become: yes
  vars:
    db_password: !vault |
    $ANSIBLE_VAULT;1.1;AES256
    36356339616139623230646264653934653333353635323335366362363935376566306532613263
    3464336137336531666538393165643461616530303462660a633436316339653664633566303562
    64346261336533383534633834336563343663323862346139383662363132633537623338646532
    6464613464393961390a663233356337306430386230343634633336616663323331346261323165
    3262
  tasks:
    - name: Installing mysql
      apt:
        name: mysql-server
        state: present
        update-cache: yes

    - name : Installing dependencies for db
      apt:
        name: python2.7-mysqldb
        state: present
        update-cache: yes

    - name: wait 10 sec
      pause:
        seconds: 10

    - name: create db user
      mysql_user:
        name: wordpress
        password: '{{ db_password }}'
        priv: ' *.*:ALL'
        state: present
```


STEP 4: Run playbook

```
ubuntu@ip-172-31-46-248:/etc/ansible$ ansible-playbook mysqlplaybook.yml --ask-vault-pass
Vault password:

PLAY [all] *****

TASK [Installing mysql] *****
changed: [ubuntu@18.208.156.61]
changed: [ubuntu@3.83.42.122]

TASK [Installing dependencies for db] *****
changed: [ubuntu@18.208.156.61]
changed: [ubuntu@3.83.42.122]

TASK [wait 10 sec] *****
Pausing for 10 seconds
(ctrl+C then 'C' = continue early, ctrl+C then 'A' = abort)
ok: [ubuntu@18.208.156.61]

TASK [create db user] *****
changed: [ubuntu@18.208.156.61]
changed: [ubuntu@3.83.42.122]

PLAY RECAP *****
ubuntu@18.208.156.61      : ok=4    changed=3    unreachable=0    failed=0
ubuntu@3.83.42.122      : ok=3    changed=3    unreachable=0    failed=0
```

STEP 5: Verify the setup by log in to mysql.

```
ubuntu@ip-172-31-46-248:~$ sudo mysql -u wordpress -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.29-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █
```