

Enterprise Software Platform

Homework 1 - Ansible

Monica Dommaraju
014545336
San Jose State University

[Introduction](#)

[Web server development](#)

[Environment setup](#)

[Ansible Installation](#)

[Web Server deployment using Ansible](#)

[Deploying the flask app](#)

[Undeploying the flask app](#)

[Github Repository](#)

[Addendum](#)

[Virtual Machine Setup](#)

Introduction

This document explains the process of setting up virtual machine, downloading and installing Ansible, configuring playbook along with deploying the web server to virtual machine server using Ansible.

Web server development

We have chosen Python Flask web server framework for this assignment because of its simplicity in configuration and deployment.

- Created a new Python flask project using PyCharm IDE.
- Added a resource file to accept GET / requests from the user and returns **Hello World!**
- Configured the server to listen on port 8080
- Code sample

```
# app.py
# -----
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello World!"

if __name__ == '__main__':
    app.run(host="0.0.0.0", port="8080")
```

- Ran the flask application and made sure that the server is responding to GET / requests with **Hello World!!** Response.

Environment setup

- Boot two new amazon ec2 instances and name them as
 - Control Node server
 - Node server

aws Services Resource Groups

EC2 Dashboard

Launch Instance Connect Actions

Filter by tags and attributes or search by keyword

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP
Control Node	i-0785e1a4351905274	t2.micro	us-east-1b	running	2/2 checks ...	None	ec2-3-91-243-52.compute-1.amazonaws.com	3.91.243.52
Node	i-0914ff2bc501de6ab	t2.micro	us-east-1b	running	2/2 checks ...	None	ec2-34-236-153-73.co...	34.236.153.73

Instance: i-0785e1a4351905274 (Control Node) Public DNS: ec2-3-91-243-52.compute-1.amazonaws.com

Description Status Checks Monitoring Tags Usage Instructions

Instance ID	i-0785e1a4351905274	Public DNS (IPv4)	ec2-3-91-243-52.compute-1.amazonaws.com
Instance state	running	IPv4 Public IP	3.91.243.52
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs	-	Private DNS	ip-172-31-30-238.ec2.internal
Availability zone	us-east-1b	Private IPs	172.31.30.238
Security groups	CentOS 7 -x86_64- - with Updates HVM-1901_01-AutogenByAWSMP-1. view inbound rules. view outbound rules	Secondary private IPs	-
Scheduled events	No scheduled events	VPC ID	vpc-14e8ad6e
AMI ID	CentOS Linux 7 x86_64 HVM EBS ENA 1901_01-b7ee8a69-ee97-4a49-9e68-afae216db2e-ami-05713873c6794f575.4 (ami-02eac2c0129f6376b)	Subnet ID	subnet-28a21a65
Platform	-	Network interfaces	eth0
IAM role	-	Source/dest. check	True
Key pair name	ec2	T2/T3 Unlimited	Disabled

- Edit security group to add TCP traffic on port 8080

aws Services Resource Groups

Launch Templates Spot Requests Reserved Instances Dedicated Hosts Scheduled Instances Capacity Reservations

IMAGES AMIs Bundle Tasks

ELASTIC BLOCK STORE Volumes Snapshots Lifecycle Manager

NETWORK & SECURITY Security Groups Elastic IPs Placement Groups Key Pairs Network Interfaces

LOAD BALANCING Load Balancers

Create Security Group Actions

Filter by tags and attributes or search by keyword

Name	Group ID	Group Name	VPC ID	Owner
sg-07dc0e613ddb6a57e	sg-07dc0e613ddb6a57e	CentOS 7 -x86_64- - with Up...	vpc-14e8ad6e	210625281936
sg-1119994f	sg-1119994f	default	vpc-14e8ad6e	210625281936

Security Group: sg-07dc0e613ddb6a57e

Description Inbound Outbound Tags

Edit

Type	Protocol	Port Range	Source
Custom TCP Rule	TCP	8080	0.0.0.0/0
Custom TCP Rule	TCP	8080	:::0
SSH	TCP	22	0.0.0.0/0

- Save the **key pair** to local machine and store it in a secure location.
- Change the permissions of the file using **chmod** so that only admin can read the file.

```
chmod 400 ~/.ssh/ec2.pem
```

- Make sure we can login to instances using **ssh** like this

```
ssh centos@34.236.153.73 -i ~/.ssh/ec2.pem
```

In order to deploy flask app on **Node Server**, **Control Node Server** should be able to ssh into the Node Server. For that we need to make the following changes

- Generate **rsa** public and private key pair on control node server.

```
[centos@ip-172-31-30-238 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/centos/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/centos/.ssh/id_rsa.
Your public key has been saved in /home/centos/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:jRVeWngGVGjo4brBWne6IjL42uvMOVCKL93UsNw72tI
centos@ip-172-31-30-238.ec2.internal
The key's randomart image is:
+---[RSA 2048]-----+
|           o+=+      |
|           o.+=o     |
|           o o+o     |
|      . .   o+       |
|.o . * .S .         |
|+   + B . .         |
| = o + = o          |
|o+*.=.E .           |
|.=0=.+.o..          |
+-----[SHA256]-----+
[centos@ip-172-31-30-238 ~]$
```

- Enable **passwordAuthentication** to yes on **Node Server**, so that public key of Control Node can be copied to Node server authorized_keys using **ssh-copy-id**

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

- Enter new password for centos user

```
[centos@ip-172-31-24-41 ~]$ sudo passwd centos
Changing password for user centos.
New password:
BAD PASSWORD: The password fails the dictionary check - it does not contain
enough DIFFERENT characters
Retype new password:
passwd: all authentication tokens updated successfully.
```

- Restart the sshd service.

```
[centos@ip-172-31-24-41 ~]$ sudo service sshd restart
Redirecting to /bin/systemctl restart sshd.service
```

- Now copy the Control Node server public key to Node server using **ssh-copy-id** and make sure you control node can ssh into the node server without asking for any password.

```
[centos@ip-172-31-30-238 ~]$ ssh-copy-id centos@172.31.24.41
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/home/centos/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to
filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
centos@172.31.24.41's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'centos@172.31.24.41'"
and check to make sure that only the key(s) you wanted were added.
```

```
[centos@ip-172-31-30-238 ~]$ ssh centos@172.31.24.41
Last login: Sun Sep 15 19:49:31 2019 from ip-172-31-30-238.ec2.internal
```

Ansible Installation

- Install Ansible on Control Node Server

```
[centos@ip-172-31-24-41 Ansible]$ sudo yum install ansible
```

- Make sure to verify that ansible is able to connect to the Node server by running the **ping** module.

```
[centos@ip-172-31-30-238 Ansible]$ ansible all -m ping -i hosts -k -K
SSH password:
SUDO password[defaults to SSH password]:
172.31.24.41 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

- Now lets clone the playbook script to run from github

```
[centos@ip-172-31-30-238 Ansible]$ git clone
https://github.com/monicasjsu/Ansible.git
Cloning into 'Ansible'...
remote: Enumerating objects: 70, done.
remote: Counting objects: 100% (70/70), done.
remote: Compressing objects: 100% (54/54), done.
remote: Total 70 (delta 29), reused 35 (delta 11), pack-reused 0
Unpacking objects: 100% (70/70), done.
```

Web Server deployment using Ansible

Deploying the flask app

- Run the playbook to deploy the flask app as a **systemd** service.

```
- name: Install flask webserver
  hosts: all
  become: yes
  become_user: root

  tasks:
    - name: Install epel-release
```

```
  yum:
    name: "{{ packages }}"
    state: present
  vars:
    packages:
      - epel-release

- name: Install pip
  yum:
    name: "{{ packages }}"
  vars:
    packages:
      - python-pip
      - git

- name: Install virtualenv
  pip:
    name: virtualenv

- name: Clone app from Git
  git:
    repo: 'https://github.com/monicasjsu/Ansible.git'
    dest: /home/centos/Ansible/
    force: yes

- name: Install python requirements
  pip:
    requirements: /home/centos/Ansible/flask/requirements.txt
    virtualenv: /home/centos/Ansible/flask/venv
    virtualenv_python: python

- name: Copy service daemon(flaskapp.service) to /etc/service.d/service/
  copy:
    src: "{{ item }}"
    dest: /etc/systemd/system/
  with_fileglob:
    - /home/centos/Ansible/flaskapp.service

- name: Start flask service
  systemd:
    state: started
    name: flaskapp
```


- Output after running the deployment playbook

```
[centos@ip-172-31-30-238 Ansible]$ ansible-playbook flask-playbook.yaml -i
hosts -k -K
SSH password:
SUDO password[defaults to SSH password]:

PLAY [Install flask webserver]
*****

TASK [Gathering Facts]
*****
ok: [172.31.24.41]

TASK [Install epel-release]
*****
ok: [172.31.24.41]

TASK [Install pip]
*****
ok: [172.31.24.41]

TASK [Install virtualenv]
*****
ok: [172.31.24.41]

TASK [Clone app from Git]
*****
ok: [172.31.24.41]

TASK [Install python requirements]
*****
ok: [172.31.24.41]

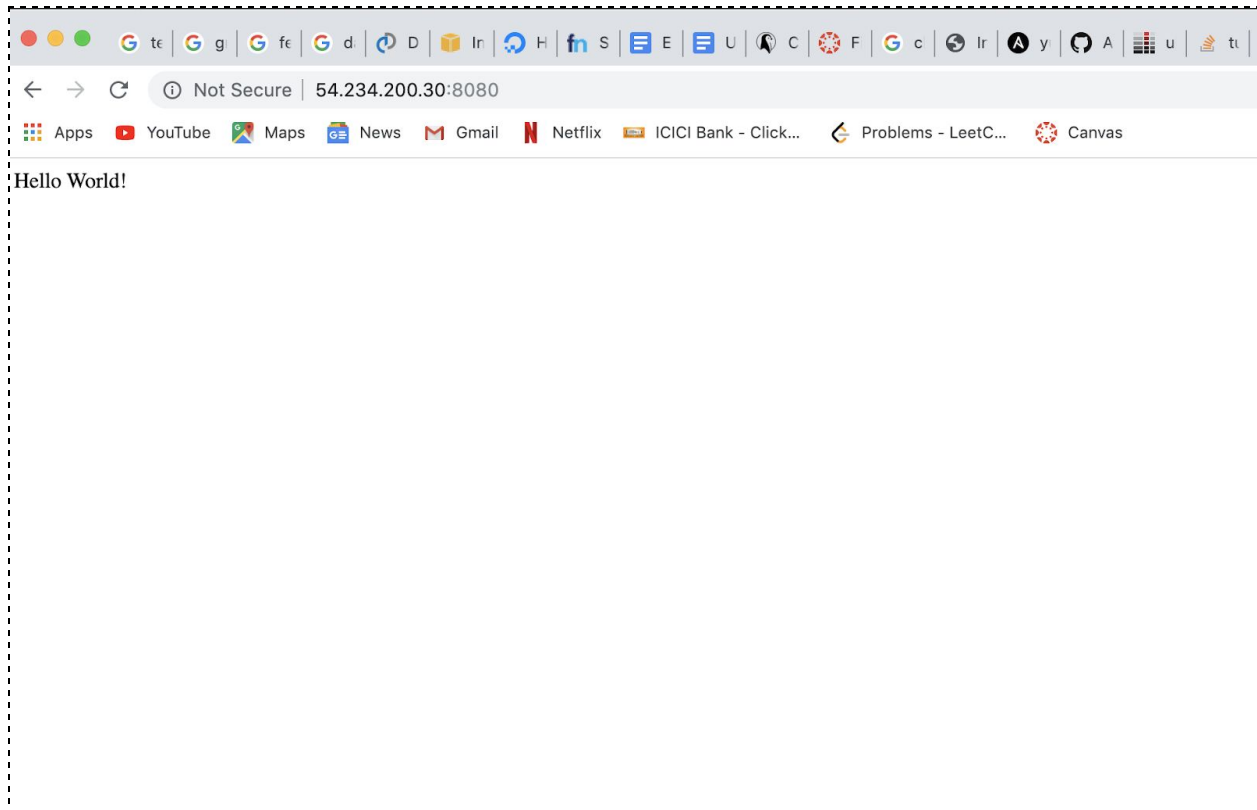
TASK [Copy service daemon(flaskapp.service) to /etc/service.d/service/]
*****
ok: [172.31.24.41] => (item=/home/centos/Ansible/flaskapp.service)

TASK [Start flask service]
*****
ok: [172.31.24.41]
```

PLAY RECAP

```
*****
**
172.31.24.41           : ok=8    changed=0    unreachable=0    failed=0
```

- Verify that the web server can be accessed using the public ip of the ec2 node server



Undeploying the flask app

- Running flask application can be stopped and undeployed by running the following playbook.

```
- name: Stop flask webserver
  hosts: all
  become: yes
  become_user: root
```

```
tasks:
- name: stop flask service
  systemd:
    state: stopped
    name: flaskapp
```

```
[centos@ip-172-31-30-238 Ansible]$ ansible-playbook
```

```
flask-playbook-undeploy.yaml -i hosts -k -K
```

```
SSH password:
```

```
SUDO password[defaults to SSH password]:
```

```
PLAY [Install flask webserver]
```

```
*****
```

```
TASK [Gathering Facts]
```

```
*****
```

```
ok: [172.31.24.41]
```

```
TASK [stop flask service]
```

```
*****
```

```
changed: [172.31.24.41]
```

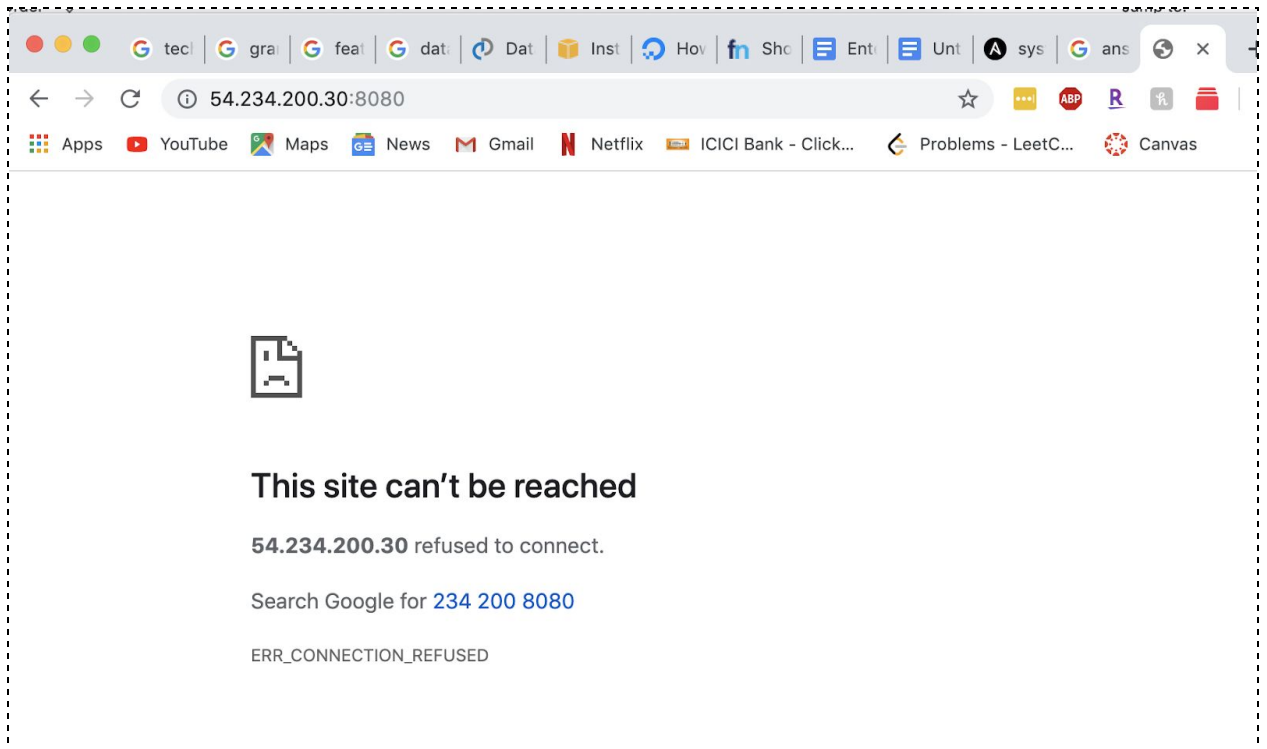
```
PLAY RECAP
```

```
*****
```

```
**
```

```
172.31.24.41 : ok=2 changed=1 unreachable=0 failed=0
```

- Test to verify that the web server is not available.



Github Repository

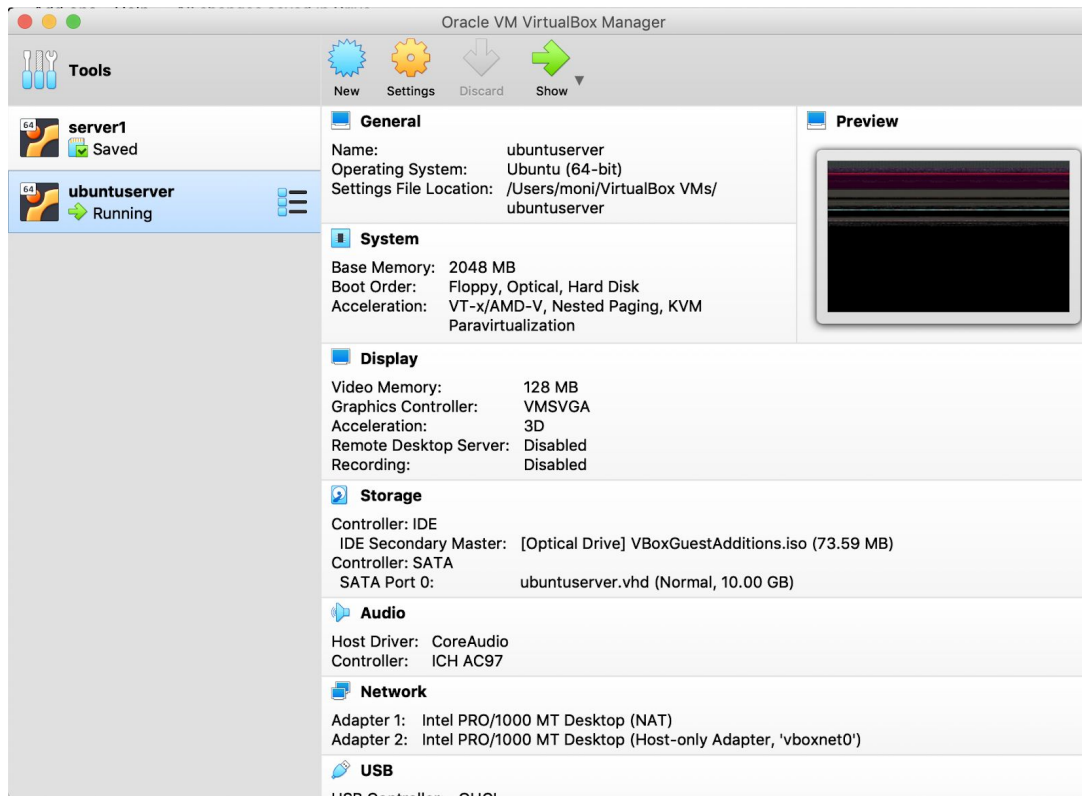
<https://github.com/monicasjsu/Ansible>

Addendum

Virtual Machine Setup

Ansible was also tested on ubuntu virtual machine installed on Mac. Installation steps are mentioned below.

- Downloaded and installed Oracle Virtual Box.
- Downloaded iso image file of Ubuntu 18.04.
- Created and booted a new virtual machine using the image downloaded.
- Installed VboxGuest additions to support full scale resolution and clipboard.
- Added network adapters to the Virtual machine.
 - NAT network adapter to support Internet
 - Host Only network adapter to support access between host and VM



- Verified that host is able to communicate with VM using Ping command

```
~ ping 192.168.56.4
```

```
PING 192.168.56.4 (192.168.56.4): 56 data bytes
64 bytes from 192.168.56.4: icmp_seq=0 ttl=64 time=0.316 ms
64 bytes from 192.168.56.4: icmp_seq=1 ttl=64 time=0.312 ms
64 bytes from 192.168.56.4: icmp_seq=2 ttl=64 time=0.422 ms
64 bytes from 192.168.56.4: icmp_seq=3 ttl=64 time=0.426 ms
```

- Installed **openssh-server** program using apt package manager in the server to accept SSH connections from the host. This will enable the control server (host machine) to run ansible modules.

```
~ sudo apt update
~ sudo apt install openssh-server
```