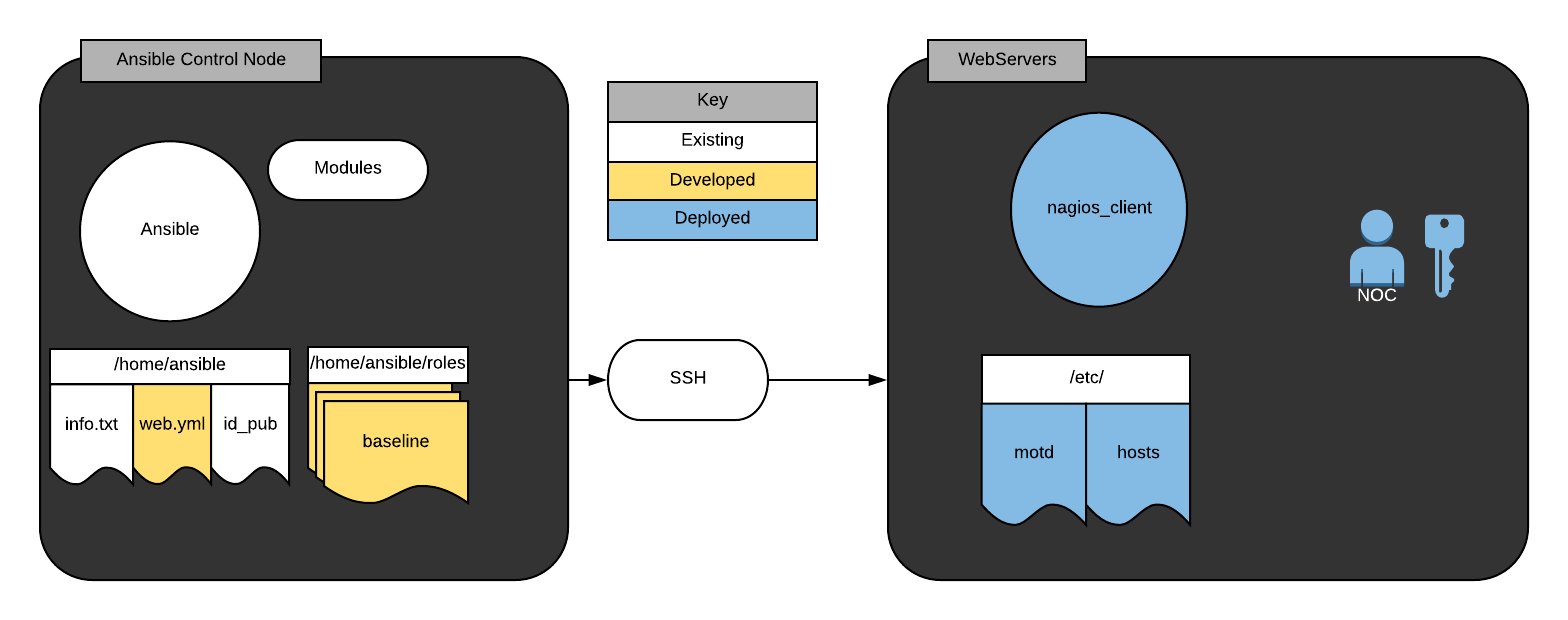
# Ansible Professional

PROFESSIONAL

# LAB 1 : 1. Working with Ansible Roles



**NOTES:**

CONTROL 1 USER DATA :

[cloud\_user@control1 ~]$ curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo 'VvSW]-y8' | /bin/passwd cloud\_user —stdin

/bin/echo "StrictHostKeyChecking no" >> /etc/ssh/ssh\_config

sed -i s/requiretty/\!requiretty/ /etc/sudoers

hostnamectl set-hostname control1

/bin/echo 10.0.1.127 node1>> /etc/hosts

/bin/echo 10.0.1.150 node2>> /etc/hosts

yum-config-manager —enable rhui-REGION-rhel-server-extras

cd /tmp

wget <https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm>

yum install -y epel-release-latest-7.noarch.rpm

yum install -y ansible

yum install -y sshpass

/sbin/useradd ansible

/bin/echo 'VvSW]-y8' | /bin/passwd ansible —stdin

/bin/echo "ansible ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

sudo -u ansible /bin/mkdir -p /home/ansible/.ssh

sudo -u ansible /bin/ssh-keygen -q -N "" -f /home/ansible/.ssh/id\_rsa

sudo -u ansible sshpass -p 'VvSW]-y8' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost

sudo -u ansible sshpass -p 'VvSW]-y8' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@node1

sudo -u ansible sshpass -p 'VvSW]-y8' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@node2

echo "localhost" >> /etc/ansible/hosts

echo "[webservers]" >> /etc/ansible/hosts

echo "node1" >> /etc/ansible/hosts

echo "node2" >> /etc/ansible/hosts

mkdir -p /home/ansible/keys/noc/.ssh

yum install -y git

git clone <https://github.com/linuxacademy/content-rh-ansible.git> /root/content-rh-ansible

/bin/mv /root/content-rh-ansible/lab7/resources /home/ansible/resources

/bin/rm -rf /root/content-rh-ansible

/bin/echo 10.0.1.127 nagios.example.com >> /home/ansible/resources/nagios\_info.txt

mkdir /home/ansible/resources/keys

/bin/ssh-keygen -q -N "" -f /home/ansible/resources/keys/noc\_id\_rsa

/bin/cat /home/ansible/resources/keys/noc\_id\_rsa.pub > /home/ansible/resources/authorized\_keys

chown -R ansible:ansible /home/ansible

rpm -ivh <https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm>

rpm --import <https://epel.mirror.constant.com/RPM-GPG-KEY-EPEL-7>

/usr/bin/pip3 install <https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-py3-latest.tar.gz>

/usr/local/bin/cfn-signal -e 0 —stack cfst-1583-6b34fade7aba2e32b752811f9f3f4e01 —resource Control1 —region us-east-1

[cloud\_user@control1 ~]$

NODE 1 and NODE 2 :

[cloud\_user@node1 ~]$ curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo 'VvSW]-y8' | /bin/passwd cloud\_user —stdin

/bin/echo "StrictHostKeyChecking no" >> /etc/ssh/ssh\_config

/usr/bin/hostnamectl set-hostname node1

/usr/bin/yum-config-manager —enable rhui-REGION-rhel-server-extras

cd /tmp

wget <https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm>

yum install -y epel-release-latest-7.noarch.rpm

/sbin/useradd ansible

/bin/echo 'VvSW]-y8' | /bin/passwd ansible —stdin

/bin/echo "ansible ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

rpm -ivh <https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm>

rpm --import <https://epel.mirror.constant.com/RPM-GPG-KEY-EPEL-7>

/usr/bin/pip3 install <https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-py3-latest.tar.gz>

/usr/local/bin/cfn-signal -e 0 —stack cfst-1583-6b34fade7aba2e32b752811f9f3f4e01 —resource Node1 —region us-east-1

[cloud\_user@node1 ~]$

[cloud\_user@control1 ~]$ sudo cat /etc/\*-release

NAME="Red Hat Enterprise Linux Server"

VERSION="7.9 (Maipo)"

ID="rhel"

ID\_LIKE="fedora"

VARIANT="Server"

VARIANT\_ID="server"

VERSION\_ID="7.9"

PRETTY\_NAME="Red Hat Enterprise Linux Server 7.9 (Maipo)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:redhat:enterprise\_linux:7.9:GA:server"

HOME\_URL="<https://www.redhat.com/>“

BUG\_REPORT\_URL="<https://bugzilla.redhat.com/>“

REDHAT\_BUGZILLA\_PRODUCT="Red Hat Enterprise Linux 7"

REDHAT\_BUGZILLA\_PRODUCT\_VERSION=7.9

REDHAT\_SUPPORT\_PRODUCT="Red Hat Enterprise Linux"

REDHAT\_SUPPORT\_PRODUCT\_VERSION="7.9"

Red Hat Enterprise Linux Server release 7.9 (Maipo)

Red Hat Enterprise Linux Server release 7.9 (Maipo)

[cloud\_user@control1 ~]$

[cloud\_user@control1 ~]$ ansible —version

ansible 2.9.27

config file = /etc/ansible/ansible.cfg

configured module search path = [u'/home/cloud\_user/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']

ansible python module location = /usr/lib/python2.7/site-packages/ansible

executable location = /usr/bin/ansible

python version = 2.7.5 (default, May 27 2022, 11:27:32) [GCC 4.8.5 20150623 (Red Hat 4.8.5-44)]

[cloud\_user@control1 ~]$

NOTES :

Server baseline:

In Server Configuration Monitor (SCM), each node can have a snapshot of all configuration items from all profiles at a particular date set as its baseline configuration. A baseline is the ideal or standard configuration for that node. It is the configuration against which you want to judge that node going forward.

MOTD:

pam\_motd is a PAM module that can be used to display arbitrary motd (message of the day) files after a successful login. By default, pam\_motd shows files in the following locations: /etc/motd /run/motd /usr/lib/motd /etc/motd.

The contents of /etc/motd are displayed by login(1) after a successful login but just before it executes the login shell. The abbreviation "motd" stands for "message of the day", and this file has been traditionally used for exactly that (it requires much less disk space than mail to all users).

COPY vs TEMPLATE modeule :

copy: just copy the file

template: can generate dynamic file based on variables and system properties

NRPE allows you to remotely execute Nagios plugins on other Linux/Unix machines. This allows you to monitor remote machine metrics (disk usage, CPU load, etc.). NRPE can also communicate with Windows agent addons like NSClient++, so you can check metrics on remote Windows machines as well.20 Sept 2019

Nagios provides complete monitoring of Linux services. Nagios is capable of monitoring the state of any Linux service (Apache, MySQL, BIND, etc.) and alerting you when the service is stopped or crashes.

noc user :

NOC engineers, also known as Network Operations Center engineers, monitor and control large computer networks from a central location. These skilled IT professionals are responsible for network troubleshooting and alarm response.

GUIDE :

# Working with Ansible Roles

# *This course is not approved or sponsored by Red Hat.*

## Introduction

## Working with Ansible roles is a key concept covered on the Red Hat Certified Ansible Specialist Exam (EX407). This should not be a surprise, considering how much functionality roles provide. This exercise covers how to create a role and how to use roles within a playbook. In order to complete this exercise, you will need to have basic proficiency with several common Ansible modules and using Ansible playbooks. After completing this learning activity, you will better understand how to work with Ansible roles.

## Solution

## Log in to the Ansible control node as ansible using the credentials provided:

ssh ansible@<PUBLIC\_IP\_ADDRESS>

**Note:** *When copying and pasting code into Vim from the lab guide, first enter :set paste (and then i to enter insert mode) to avoid adding unnecessary spaces and hashes.*

### Create a Role Called baseline in /etc/ansible/roles

Create the structure needed for the role:

cd /etc/ansible/roles/  
sudo mkdir baseline && sudo chown ansible.ansible /etc/ansible/roles/baseline  
mkdir /etc/ansible/roles/baseline/{templates,tasks,files}  
echo "---" > baseline/tasks/main.yml

### Configure the Role to Deploy the /etc/motd Template

Copy the file:

cp /home/ansible/resources/motd.j2 baseline/templates

Create a file called deploy\_motd.yml:

vim baseline/tasks/deploy\_motd.yml

Add the following content:

---  
- template:  
    src: motd.j2  
    dest: /etc/motd

Save and exit with **Escape** followed by :wq.

Open main.yml:

vim baseline/tasks/main.yml

Add the following lines to the file:

- name: configure motd  
  import\_tasks: deploy\_motd.yml

Save and exit with **Escape** followed by :wq.

### Configure the Role to Install the Latest Nagios Client

Find the package we need to install by reading a text file in our home directory:

cat /home/ansible/resources/nagios\_info.txt

That file tells us the package we need to install is nrpe.x86\_64.

Copy the IP of the Nagios server that's in the file and paste it into a text file, as we'll need it later.

Create a file, which will install the package, called deploy\_nagios.yml:

vim baseline/tasks/deploy\_nagios.yml

Add the following content:

---  
- yum: name=nrpe state=latest

Save and exit with **Escape** followed by :wq.

Open main.yml:

vim baseline/tasks/main.yml

Add the following lines to the bottom of the file:

- name: deploy nagios client  
  import\_tasks: deploy\_nagios.yml

Save and exit with **Escape** followed by :wq.

### Configure the Role to Add an Entry to /etc/hosts for the Nagios Server

Create a file called edit\_hosts.yml:

vim baseline/tasks/edit\_hosts.yml

Add the following content, substituting <IP\_ADDRESS> with the Nagios server IP you copied earlier:

---  
- lineinfile:  
    line: "<IP\_ADDRESS> nagios.example.com"  
    path: /etc/hosts

Save and exit with **Escape** followed by :wq.

Open main.yml:

vim  baseline/tasks/main.yml

Add the following lines to the bottom of the file:

- name: edit hosts file  
  import\_tasks: edit\_hosts.yml

Save and exit with **Escape** followed by :wq.

### Configure the Role to Create the noc User and Deploy the Provided Public Key for the noc User on Target Systems

Copy the provided authorized\_keys file to our files directory:

cp /home/ansible/resources/authorized\_keys /etc/ansible/roles/baseline/files/

Create a file called deploy\_noc\_user.yml:

vim baseline/tasks/deploy\_noc\_user.yml

Add the following content:

---  
- user: name=noc  
- file:  
     state: directory  
     path: /home/noc/.ssh  
     mode: 0600  
     owner: noc  
     group: noc  
- copy:  
     src: authorized\_keys  
     dest: /home/noc/.ssh/authorized\_keys  
     mode: 0600  
     owner: noc  
     group: noc

Save and exit with **Escape** followed by :wq.

Open main.yml:

vim baseline/tasks/main.yml

Add the following lines to the bottom of the file:

- name: set up noc user and key  
  import\_tasks: deploy\_noc\_user.yml

Save and exit with **Escape** followed by :wq.

### Edit web.yml to Deploy the baseline Role

Change back to the home directory:

cd /home/ansible/

Open web.yml:

vim resources/web.yml

Edit it to match the following:

---  
- hosts: webservers  
  become: yes  
  roles:  
    - baseline  
  tasks:  
    - name: install httpd  
      yum: name=httpd state=latest  
    - name: start and enable httpd  
      service: name=httpd state=started enabled=yes

Save and exit with **Escape** followed by :wq.

### Run Your Playbook Using the Default Inventory

Deploy the playbook:

ansible-playbook resources/web.yml

### Check Our Work

Log in to one of the nodes (the IP addresses are on the hands-on lab overview page):

ssh node1

We should see a new MOTD, so we know that play worked.

See if the noc user was set up:

id noc

Check to see if the nrpe package was installed:

sudo yum list nrpe

## Conclusion

## All these plays ran, and now we've got a playbook that we can edit when we want to keep things consistent across our webservers. Congratulations!

## Additional Resources

You have just started a new job as the operations lead at a small company. There is currently no formal server baseline, and it makes for a mixed configuration environment that is consuming more support and maintenance than it should. You have decided to create a baseline process using Ansible by creating a baseline role. You have noted the following commonalities that should be included in the baseline role:

* Set */etc/motd* based on a template.
* Install the latest Nagios client.
* Add the Nagios server to */etc/hosts*.
* Create a *noc* user.
* Import the *noc* user's public key (copy authorized keys to */home/noc/.ssh/authorized\_keys*).

The role should be called "baseline" and should reside in */etc/ansible/roles* on the *ansible control* node.

You will test your role on some newly requested webservers. A playbook called *web.yml* has been provided for you and deploys httpd to all servers in the web group (defined in your default inventory). You will need to edit the playbook to deploy the *baseline* role to the servers in the *web* group as well.

You will find the *motd* template, Nagios server IP information, the *noc* user's public key, and the *web.yml* playbook in */home/ansible/resources* on the *ansible control* node.

Summary tasks list:

* Create the necessary directories and files for the *baseline* role.
* Configure the role to deploy the */etc/motd* template.
* Configure the role to install the latest Nagios client.
* Configure the role to add an entry to */etc/hosts* for the Nagios server.
* Configure the role to create the *noc* user and deploy the provided public key for the *noc* user on target systems (copy *authorized\_keys* to */home/noc/.ssh/authorized\_keys* with the *owner* and *group owner* set as *noc* and the mode as *0600*).
* Edit *web.yml* to deploy the *baseline* role in addition to what it already does.
* Verify that your role works by deploying *web.yml* with Ansible.

Important notes:

* For your convenience, Ansible is already installed on the control node.
* The user *ansible* is on all servers with the appropriate shared keys for access to necessary servers from the control node.
* The *ansible* user has sudo access with no password. It uses the same password as *cloud\_user*.
* All the necessary Ansible inventories have been created for you.

## Learning Objectives

0 of 7 completed

Create a Role Called baseline in /etc/ansible/roles

Run the following commands to create the structure needed for the role:

* sudo mkdir /etc/ansible/roles/baseline && sudo chown ansible.ansible /etc/ansible/roles/baseline
* mkdir /etc/ansible/roles/baseline/{templates,tasks,files}
* echo "---" > /etc/ansible/roles/baseline/tasks/main.yml

Configure the Role to Deploy the /etc/motd Template

cp /home/ansible/resources/motd.j2 /etc/ansible/roles/baseline/templates

Create a file called /etc/ansible/roles/baseline/tasks/deploy\_motd.yml with the following content:

---  
- template:  
    src: motd.j2  
    dest: /etc/motd

Edit /etc/ansible/roles/baseline/tasks/main.yml to include the following lines at the bottom of the file:

- name: configure motd  
  import\_tasks: deploy\_motd.yml

Configure the Role to Install the Latest Nagios Client

Create a file called /etc/ansible/roles/baseline/tasks/deploy\_nagios.yml with the following content:

---  
- yum: name=nrpe state=latest

Edit /etc/ansible/roles/baseline/tasks/main.yml to include the following lines at the bottom of

the file (take care with the formatting.):

  - name: deploy nagios client  
    import\_tasks: deploy\_nagios.yml

Configure the Role to Add an Entry to /etc/hosts for the Nagios Server

Create a file called */etc/ansible/roles/baseline/tasks/edit\_hosts.yml* with the following content, substituting <<PROVIDED>PROVIDED> with the IP specified in */home/ansible/resources/nagios\_info.txt*:

---  
- lineinfile:  
    line: "<<PROVIDED>PROVIDED> nagios.example.com"  
    path: /etc/hosts

Edit */etc/ansible/roles/baseline/tasks/main.yml* to include the following lines at the bottom of the file:

     - name: edit hosts file  
       import\_tasks: edit\_hosts.yml

Configure the Role to Create the noc User and Deploy the Provided Public Key for the noc User on Target Systems

Copy the file /home/ansible/resources/authorized\_keys to /etc/ansible/roles/baseline/files/.

Create a file called /etc/ansible/roles/baseline/tasks/deploy\_noc\_user.yml with the following content:

---  
- user: name=noc  
- file:  
     state: directory  
     path: /home/noc/.ssh  
     mode: 0600  
     owner: noc  
     group: noc  
- copy:  
        src: authorized\_keys  
        dest: /home/noc/.ssh/authorized\_keys  
        mode: 0600  
        owner: noc  
        group: noc

Edit /etc/ansible/roles/baseline/tasks/main.yml to include the following lines at the bottom of the file:

      - name: set up noc user and key  
        import\_tasks: deploy\_noc\_user.yml

Edit web.yml to Deploy the baseline Role

Edit */home/ansible/resources/web.yml* to the following:

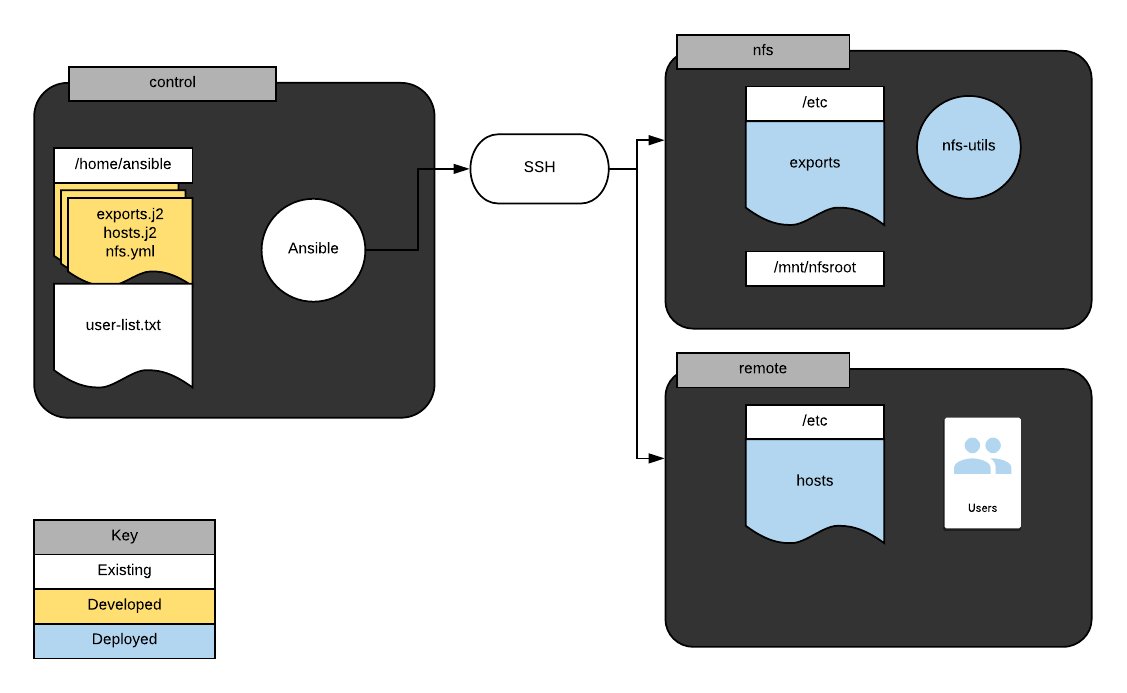
---  
- hosts: webservers  
  become: yes  
  roles:  
    - baseline  
  tasks:  
    - name: install httpd  
      yum: name=httpd state=latest  
    - name: start and enable httpd  
      service: name=httpd state=started enabled=yes

Run Your Playbook Using the Default Inventory

Run ansible-playbook /home/ansible/resources/web.yml.

PROFESSIONAL   
LAB 2 :

2 :  Deploying Services Using Ansible



NOTES :

Deploying Services Using Ansible

CONTROL 1 USER DATA :

[ansible@control ~]$ curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo 'G0rRUcD-' | /bin/passwd root —stdin

/bin/echo 'G0rRUcD-' | /bin/passwd cloud\_user —stdin

/bin/echo "StrictHostKeyChecking no" >> /etc/ssh/ssh\_config

sed -i s/requiretty/\!requiretty/ /etc/sudoers

hostnamectl set-hostname control

/bin/echo 10.0.1.65 nfs node1>> /etc/hosts

/bin/echo 10.0.1.211 node2>> /etc/hosts

yum install -y epel-release

yum install -y ansible

yum install -y sshpass

yum install -y python36

/sbin/useradd ansible

/bin/echo 'G0rRUcD-' | /bin/passwd ansible —stdin/bin/echo "ansible ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

sudo -u ansible /bin/mkdir -p /home/ansible/.sshsudo -u ansible /bin/ssh-keygen -q -N "" -f /home/ansible/.ssh/id\_rsa

sudo -u ansible sshpass -p 'G0rRUcD-' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost

sudo -u ansible sshpass -p 'G0rRUcD-' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@node1

sudo -u ansible sshpass -p 'G0rRUcD-' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@node2

echo "users:" > /home/ansible/user-list.txt

echo "- frank" >> /home/ansible/user-list.txt

echo "- judy" >> /home/ansible/user-list.txt

echo "- joe" >> /home/ansible/user-list.txt

echo "- sarah" >> /home/ansible/user-list.txt

echo "- sam" >> /home/ansible/user-list.txt

echo "- carry" >> /home/ansible/user-list.txt

chown ansible.ansible /home/ansible/\*

echo "localhost" >> /etc/ansible/hosts

echo "nfs" >> /etc/ansible/hosts

echo "[remote]" >> /etc/ansible/hosts

echo "node2" >> /etc/ansible/hosts

/usr/bin/pip3 install <https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-py3-latest.tar.gz>

/usr/local/bin/cfn-signal -e 0 —stack cfst-1748-2d6f8e9916119d1cdaaab02af77dd4f3 —resource Control —region us-east-1

[ansible@control ~]$

Cloud Server of nfs USER DATA :

[cloud\_user@nfs ~]$ curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo 'G0rRUcD-' | /bin/passwd cloud\_user —stdin

/bin/echo "StrictHostKeyChecking no" >> /etc/ssh/ssh\_config

/usr/bin/hostnamectl set-hostname nfs

/sbin/useradd ansible

/bin/echo 'G0rRUcD-' | /bin/passwd ansible —stdin

/bin/echo "ansible ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

mkdir /mnt/nfsroot

/sbin/setenforce 0

/bin/firewall-cmd —permanent —add-service=nfs

/bin/firewall-cmd —reload

/sbin/setenforce 1

yum install -y python36

/usr/bin/pip3 install <https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-py3-latest.tar.gz>

/usr/local/bin/cfn-signal -e 0 —stack cfst-1748-2d6f8e9916119d1cdaaab02af77dd4f3 —resource Node1 —region us-east-1

[cloud\_user@nfs ~]$

Cloud Server Node 2 USER DATA :

[cloud\_user@node2 ~]$ curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo 'G0rRUcD-' | /bin/passwd cloud\_user —stdin

/bin/echo "StrictHostKeyChecking no" >> /etc/ssh/ssh\_config

/usr/bin/hostnamectl set-hostname node2

/sbin/useradd ansible

/bin/echo 'G0rRUcD-' | /bin/passwd ansible —stdin

/bin/echo "ansible ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

/bin/echo There may be users on this system > /opt/user-agreement.txt

yum install -y python36

/usr/bin/pip3 install <https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-py3-latest.tar.gz>

/usr/local/bin/cfn-signal -e 0 —stack cfst-1748-2d6f8e9916119d1cdaaab02af77dd4f3 —resource Node2 —region us-east-1

System info :

[ansible@control ~]$ sudo cat /etc/\*-release

CentOS Linux release 7.9.2009 (Core)

NAME="CentOS Linux"

VERSION="7 (Core)"

ID="centos"

ID\_LIKE="rhel fedora"

VERSION\_ID="7"

PRETTY\_NAME="CentOS Linux 7 (Core)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:centos:centos:7"

HOME\_URL="<https://www.centos.org/>“

BUG\_REPORT\_URL="<https://bugs.centos.org/>“

CENTOS\_MANTISBT\_PROJECT="CentOS-7"

CENTOS\_MANTISBT\_PROJECT\_VERSION="7"REDHAT\_SUPPORT\_PRODUCT="centos"

REDHAT\_SUPPORT\_PRODUCT\_VERSION="7"

CentOS Linux release 7.9.2009 (Core)

CentOS Linux release 7.9.2009 (Core)

[ansible@control ~]$

Ansible Version :

ansible@control ~]$ ansible —version

ansible 2.9.27

config file = /etc/ansible/ansible.cfg

configured module search path = [u'/home/ansible/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']

ansible python module location = /usr/lib/python2.7/site-packages/ansible

executable location = /usr/bin/ansible

python version = 2.7.5 (default, Nov 16 2020, 22:23:17) [GCC 4.8.5 20150623 (Red Hat 4.8.5-44)]

[ansible@control ~]$

Deploying Services Using Ansible  
Introduction  
Ansible is frequently used to manage software configuration and deployment. This exercise allows students to deploy and configure an NFS server using key features of Ansible.  
  
The team has been working on automating the builds for common services using Ansible. You have been tasked with creating the build for a common NFS server. Write a playbook that satisfies the following requirements:  
  
On nfs:  
  
Make sure nfs-utils is installed.  
  
Configure /etc/exports via an Ansible template stored in /home/ansible/exports.j2. Deploy the template so that /mnt/nfsroot is exported with read and write to all hosts.  
  
Note: Your template file should contain the following with the variable {{ share\_path }} being defined within your playbook:  
  
  {{ share\_path }} \*(rw)  
Note: The file /etc/exports on nfs should have the following content once deployed:  
  
  /mnt/nfsroot \*(rw)  
Create a handler that runs the command exportfs -a if the file /etc/exports is modified in a playbook task.  
  
You can assume all necessary firewall rules have been deployed.  
  
On remote:  
  
Configure /etc/hosts from a template file stored on control at /home/ansible/etc.hosts.j2 with the following entries:  
  
  127.0.0.1 localhost, {{ ansible\_hostname }}  
  {{ nfs\_ip }}  {{ nfs\_hostname }}  
Note: You should populate the variables {{ nfs\_ip }} and {{ nfs\_hostname }} using magic variables in your playbook.  
Create users from file stored on control at /home/ansible/user-list.txt only if the remote host has the file /opt/user-agreement.txt.  
  
The Ansible control node has been configured for you and all other servers have already been configured for use with Ansible. The default inventory has been configured to include the group remote and server nfs.  
  
Solution  
Begin in by logging in to the Ansible control node as the ansible user using the credentials on the hands-on lab page (its the same password as cloud\_user):  
  
ssh ansible@IP\_ADDRESS  
Create the necessary template files on the Ansible control node  
Create the files exports.j2 in /home/ansible with the noted content:  
  
vim /home/ansible/exports.j2  
Provide the following content for this file:  
  
{{ share\_path }} \*(rw)  
Create the file etc.hosts.j2 in /home/ansible with the noted content:  
  
vim /home/ansible/etc.hosts.j2  
Provide the following content for this file:  
  
127.0.0.1    localhost {{ ansible\_hostname}}  
{{ nfs\_ip }}    {{ nfs\_hostname }}  
Create a playbook for the server 'nfs' in the Ansible inventory  
Create a playbook in /home/ansible/nfs.yml that meets the provided specification for the server nfs in the ansible inventory.  
  
Edit the /home/ansible/nfs.yml file:  
  
vim /home/ansible/nfs.yml  
Add a playbook in /home/ansible/nfs.yml with the following content:  
  
    - hosts: nfs  
      become: yes  
      vars:  
        share\_path: /mnt/nfsroot  
      tasks:  
        - name: install nfs  
          yum:  
            name: nfs-utils  
            state: latest  
        - name: start and enable nfs-server  
          service:  
            name: nfs-server  
            state: started  
            enabled: yes  
        - name: configure exports  
          template:  
            src: /home/ansible/exports.j2  
            dest: /etc/exports  
          notify: update nfs  
      handlers:  
        - name: update nfs exports  
          command: exportfs -a  
          listen: update nfs  
Add a play for the remote host group  
Edit /home/ansible/nfs.yml such that changes are executed on the remote host group to meet the provided requirements.  
  
Open the /home/ansible/nfs.yml file:  
  
vim /home/ansible/nfs.yml  
Edit the /home/ansible/nfs.yml file:to contain the following content after the nfs play:  
  
    - hosts: remote  
      become: yes  
      vars:  
        nfs\_ip: "{{ hostvars['nfs']['ansible\_default\_ipv4']['address'] }}"  
        nfs\_hostname: "{{ hostvars['nfs']['ansible\_hostname'] }}"  
      vars\_files:  
        - /home/ansible/user-list.txt  
      tasks:  
        - name: configure hostsfile  
          template:  
            src: /home/ansible/etc.hosts.j2  
            dest: /etc/hosts  
        - name: get file status  
          stat:  
            path: /opt/user-agreement.txt  
          register: filestat  
        - name: debug info  
          debug:  
            var: filestat  
        - name: create users  
          user:  
            name: "{{ item }}"  
          when:  filestat.stat.exists  
          loop: "{{ users }}"  
Execute playbook to verify your playbook works correctly  
Execute playbook /home/ansible/nfs.yml to verify your playbook works correctly.  
  
Run the command below from the control node.  
  
ansible-playbook /home/ansible/nfs.yml  
Conclusion  
Congratulations, you've completed this hands-on lab!

## Additional Resources

The team has been working on automating the builds for common services using Ansible. You have been tasked with creating the build for a common NFS server. Write a playbook that satisfies the following requirements:

**On nfs**:

* Make sure nfs-utils is installed.
* Configure /etc/exports via an Ansible template stored in /home/ansible/exports.j2. Deploy the template so that /mnt/nfsroot is exported with read and write to all hosts.

**Note**: Your template file should contain the following with the variable {{ share\_path }} being defined within your playbook:

{{ share\_path }} \*(rw)

**Note**: The file /etc/exports on nfs should have the following content once deployed:

/mnt/nfsroot \*(rw)

* Create a handler that runs the command exportfs -a if the file /etc/exports is modified in a playbook task.
* You can assume all necessary firewall rules have been deployed.

**On remote**:

Configure /etc/hosts from a template file stored on control at /home/ansible/etc.hosts.j2 with the following entries:

  127.0.0.1 localhost {{ ansible\_hostname }}  
  {{ nfs\_ip }}  {{ nfs\_hostname }}

* + **Note**: You should populate the variables {{ nfs\_ip }} and {{ nfs\_hostname }} using magic variables in your playbook.

Create users from file stored on control at /home/ansible/user-list.txt only if the remote host has the file /opt/user-agreement.txt.

The Ansible control node has been configured for you and all other servers have already been configured for use with Ansible. The default inventory has been configured to include the group remote and server nfs.

## Learning Objectives

0 of 4 completed

Create the necessary template files on the Ansible control node.

Create the files exports.j2 and etc.hosts.j2 in /home/ansible with the noted content:

*/home/ansible/exports.j2*

{{ share\_path }} \*(rw)

*/home/ansible/etc.hosts.j2*

127.0.0.1    localhost {{ ansible\_hostname}}  
{{ nfs\_ip }}    {{ nfs\_hostname }}

Create a playbook for the server 'nfs' in the Ansible inventory.

Create a playbook in /home/ansible/nfs.yml that meets the provided specification for the server nfs in the ansible inventory.

Create a playbook in /home/ansible/nfs.yml with the following content: - hosts: nfs become: yes vars: share\_path: /mnt/nfsroot tasks: - name: install nfs yum: name: nfs-utils state: latest - name: start and enable nfs-server service: name: nfs-server state: started enabled: yes - name: configure exports template: src: /home/ansible/exports.j2 dest: /etc/exports notify: update nfs handlers: - name: update nfs exports command: exportfs -a listen: update nfs

Add a play for the remote host group.

Edit /home/ansible/nfs.yml such that changes are executed on the remote host group to meet the provided requirements.

Edit /home/ansible/nfs.yml to contain the following content after the nfs play: - hosts: remote become: yes vars: nfs\_ip: "{{ hostvars['nfs']['ansible\_default\_ipv4']['address'] }}" nfs\_hostname: "{{ hostvars['nfs']['ansible\_hostname'] }}" vars\_files: - /home/ansible/user-list.txt tasks: - name: configure hostsfile template: src: /home/ansible/etc.hosts.j2 dest: /etc/hosts.nfslab - name: get file status stat: path: /opt/user-agreement.txt register: filestat - name: debug info debug: var: filestat - name: create users user: name: "{{ item }}" when: filestat.stat.exists loop: "{{ users }}"

Execute playbook to verify your playbook works correctly.

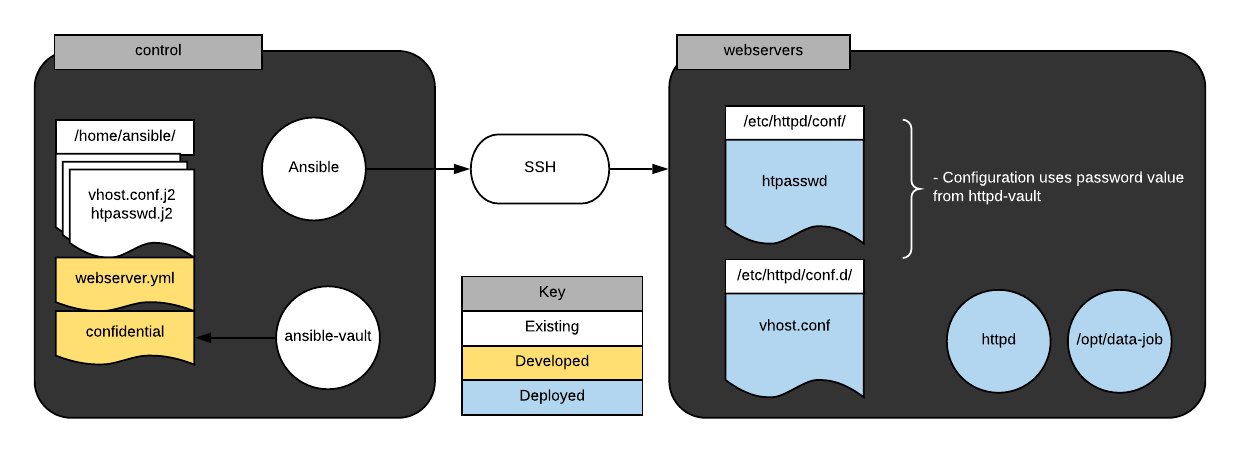
Execute playbook /home/ansible/nfs.yml to verify your playbook works correctly.

Run ansible-playbook /home/ansible/nfs.yml from the control node.

PROFESSIONAL

LAB 3

3:  Advanced Features in Ansible Playbooks



Advanced Features in Ansible Playbooks

CONTROL NODE :

[ansible@control ~]$ curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo 'BQ7|w4sp' | /bin/passwd root —stdin

/bin/echo 'BQ7|w4sp' | /bin/passwd cloud\_user —stdin

/bin/echo "StrictHostKeyChecking no" >> /etc/ssh/ssh\_config

sed -i s/requiretty/\!requiretty/ /etc/sudoers

hostnamectl set-hostname control

/bin/echo 10.0.1.68 node1>> /etc/hosts

/bin/echo 10.0.1.193 node2>> /etc/hosts

yum install -y epel-release

yum install -y ansible

yum install -y sshpass

/sbin/useradd ansible

/bin/echo 'BQ7|w4sp' | /bin/passwd ansible —stdin

/bin/echo "ansible ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

sudo -u ansible /bin/mkdir -p /home/ansible/.ssh

sudo -u ansible /bin/ssh-keygen -q -N "" -f /home/ansible/.ssh/id\_rsa

sudo -u ansible sshpass -p 'BQ7|w4sp' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost

sudo -u ansible sshpass -p 'BQ7|w4sp' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@node1

sudo -u ansible sshpass -p 'BQ7|w4sp' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@node2

echo "htpasswd: $apr1$dWF.Kizb$HAo.dpR44mEP5eIv6X90n1" > /home/ansible/confidential

echo "I love ansible" >> /home/ansible/.vault

echo "agent:{{ htpasswd }}" > /home/ansible/htpasswd.j2

echo "<VirtualHost {{ ansible\_default\_ipv4.address }}:80>" >> /home/ansible/vhost.conf.j2

echo "ServerAdmin [webmaster@xyzcorp.com](mailto:webmaster@xyzcorp.com)" >> /home/ansible/vhost.conf.j2

echo "DocumentRoot /var/www/html" >> /home/ansible/vhost.conf.j2

echo "" >> /home/ansible/vhost.conf.j2

echo '<Directory "/var/www/html"> ' >> /home/ansible/vhost.conf.j2

echo " AuthType Basic" >> /home/ansible/vhost.conf.j2

echo ' Authname "Authorized Users Only"' >> /home/ansible/vhost.conf.j2

echo " AuthUserFile /etc/httpd/conf/htpasswd" >> /home/ansible/vhost.conf.j2

echo " Require valid-user" >> /home/ansible/vhost.conf.j2

echo "</Directory>" >> /home/ansible/vhost.conf.j2

echo "</VirtualHost>" >> /home/ansible/vhost.conf.j2

echo "localhost" >> /etc/ansible/hosts

echo "[webservers]" >> /etc/ansible/hosts

echo "node1" >> /etc/ansible/hostsecho "node2" >> /etc/ansible/hosts

chown ansible.ansible /home/ansible/\*[ansible@control ~]$

[ansible@control ~]$

[ansible@control ~]$

[ansible@control ~]$ sudo cat /etc/os-release

NAME="CentOS Linux"

VERSION="7 (Core)"

ID="centos"

ID\_LIKE="rhel fedora"

VERSION\_ID="7"

PRETTY\_NAME="CentOS Linux 7 (Core)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:centos:centos:7"

HOME\_URL="<https://www.centos.org/>“

BUG\_REPORT\_URL="<https://bugs.centos.org/>“

CENTOS\_MANTISBT\_PROJECT="CentOS-7"

CENTOS\_MANTISBT\_PROJECT\_VERSION="7"

REDHAT\_SUPPORT\_PRODUCT="centos"

REDHAT\_SUPPORT\_PRODUCT\_VERSION="7"

[ansible@control ~]$

[ansible@control ~]$

[ansible@control ~]$

[ansible@control ~]$

[ansible@control ~]$ ansible —version

ansible 2.9.27

config file = /etc/ansible/ansible.cfg

configured module search path = [u'/home/ansible/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']

ansible python module location = /usr/lib/python2.7/site-packages/ansible

executable location = /usr/bin/ansible

python version = 2.7.5 (default, Nov 16 2020, 22:23:17) [GCC 4.8.5 20150623 (Red Hat 4.8.5-44)]

[ansible@control ~]$

NODE 1 and 2 :

[cloud\_user@node2 ~]$ curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo 'BQ7|w4sp' | /bin/passwd cloud\_user —stdin

/bin/echo "StrictHostKeyChecking no" >> /etc/ssh/ssh\_config

/usr/bin/hostnamectl set-hostname node2

/sbin/useradd ansible

/bin/echo 'BQ7|w4sp' | /bin/passwd ansible —stdin

/bin/echo "ansible ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

/sbin/setenforce 0

/bin/firewall-cmd —permanent —add-service=http

/bin/firewall-cmd —reload

/sbin/setenforce 1

/bin/echo "#!/bin/sh">/opt/data-job.sh

/bin/echo "#Data is going to be worked on!">>/opt/data-job.sh

/bin/echo "sleep 300">>/opt/data-job.sh

/bin/echo echo "Data is done!">>/opt/data-job.sh

/bin/chmod +x /opt/data-job.sh

[cloud\_user@node2 ~]$

Advanced Features in Ansible Playbooks  
Introduction  
There are a number of features unique to Ansible playbooks which provide robust functionality. This exercise explores many of these features in a practical scenario of deploying a web server. Most notably, this exercise deals with confidential data in an Ansible vault and working with tags in Ansible playbooks.  
  
You must create a modular playbook used for webserver management. Create a playbook called /home/ansible/webserver.yml that meets the following requirements:  
  
On the host group webservers:  
  
Deploy httpd.  
You can assume all necessary firewall rules have been deployed.  
Start and enable httpd.  
Configure virtual host using the provided templates in /home/ansible/vhost.conf.j2 and /home/ansible/htpasswd.j2.  
Note: The template references a variable defined in /home/ansible/confidential which must be included as a variable file in your playbook.  
Use ansible vault to secure /home/ansible/confidential with password "I love ansible".  
Run the data job stored in /opt/data-job.sh on each node in webservers asynchronously, without polling for status.  
Create tags for the following tasks:  
base-install for httpd installation and service configuration.  
vhost for virtual host deployment.  
data-job to execute the asynchronous data job.  
The Ansible control node has been configured for you and each webserver has already been configured for use with Ansible. The default inventory has been configured to include the webservers group and sample hosts.  
  
Solution  
Begin by logging in to the Ansible control node using the credentials on the hands-on lab page:  
  
ssh cloud\_user@PUBLIC\_IP\_ADDRESS  
Become the ansible user:  
  
sudo -i -u ansible      
Use ansible-vault to protect the confidential information  
Use ansible-vault to encrypt /home/ansible/confidential to protect the confidential information stored within using the password "I love ansible".  
  
Run ansible-vault encrypt /home/ansible/confidential and supply the password "I love ansible".  
Create a playbook that deploys httpd on webservers  
Create a playbook in /home/ansible/webserver.yml that deploys httpd on webservers. It should be tagged with base-install and contain a handler that restarts the httpd daemon that is flagged by both installation and service manipulation for httpd.  
  
Create the file /home/ansible/webserver.yml and add the following content:  
vim /home/ansible/webserver.yml  
      
---  
- hosts: webservers  
  become: yes  
  vars\_files:  
    - /home/ansible/confidential  
  tasks:  
    - name: install httpd  
      yum:  
        name: httpd  
        state: latest  
      notify: httpd service  
      tags:  
        - base-install  
  handlers:  
    - name: Restart and enable httpd  
      service:  
        name: httpd  
        state: restarted  
        enabled: yes  
      listen: httpd service  
Deploy the templates stored on the control node to the webservers group  
Configure /home/ansible/webserver.yml to deploy the templates /home/ansible/vhost.conf.j2 and /home/ansible/htpasswd.j2 stored on the control node to the webservers group. httpd must restart on config change. The tasks should be tagged vhost.  
  
Add the following text to /home/ansible/webserver.yml just before the handler section:  
vim /home/ansible/webserver.yml  
  
    - name: configure virtual host  
      template:  
        src: /home/ansible/vhost.conf.j2  
        dest: /etc/httpd/conf.d/vhost.conf  
      notify: httpd service  
      tags:  
        - vhost  
    - name: configure site auth  
      template:  
        src: /home/ansible/htpasswd.j2  
        dest: /etc/httpd/conf/htpasswd  
      notify: httpd service  
      tags:  
        - vhost  
Asynchronously execute data-job on webservers  
Configure /home/ansible/webserver.yml to asynchronously execute /opt/data-job.sh located on webservers with a timeout of 600 seconds and no polling. The task should be tagged with data-job.  
  
Add the following text to /home/ansible/webserver.yml just before the handler section:  
vim /home/ansible/webserver.yml  
  
    - name: run data job  
      command: /opt/data-job.sh  
      async: 600  
      poll: 0  
      tags:  
        - data-job  
Your complete file should look similar to the below:  
---  
- hosts: webservers  
  become: yes  
  vars\_files:  
    - /home/ansible/confidential  
  tasks:  
    - name: install httpd  
      yum:  
        name: httpd  
        state: latest  
      notify: httpd service  
      tags:  
        - base-install  
    - name: configure virtual host  
      template:  
        src: /home/ansible/vhost.conf.j2  
        dest: /etc/httpd/conf.d/vhost.conf  
      notify: httpd service  
      tags:  
        - vhost  
    - name: configure site auth  
      template:  
        src: /home/ansible/htpasswd.j2  
        dest: /etc/httpd/conf/htpasswd  
      notify: httpd service  
      tags:  
        - vhost  
    - name: run data job  
      command: /opt/data-job.sh  
      async: 600  
      poll: 0  
      tags:  
        - data-job  
  handlers:  
    - name: Restart and enable httpd  
      service:  
        name: httpd  
        state: restarted  
        enabled: yes  
      listen: httpd service  
Execute playbook to verify your playbook works correctly  
Execute playbook /home/ansible/webserver.yml to verify your playbook works correctly.  
  
Run ansible-playbook --ask-vault-pass /home/ansible/webserver.yml from the control node and provide the vault password "I love ansible".  
Conclusion  
Congratulations, you've completed this hands-on lab!

## Additional Resources

You must create a modular playbook used for webserver management. Create a playbook called /home/ansible/webserver.yml that meets the following requirements:

On the host group webservers:

* Deploy httpd.
* You can assume all necessary firewall rules have been deployed.
* Start and enable httpd.
* Configure virtual host using the provided templates in /home/ansible/vhost.conf.j2 and /home/ansible/htpasswd.j2.
  + **Note**: The template references a variable defined in /home/ansible/confidential which must be included as a variable file in your playbook.
  + Use ansible vault to secure /home/ansible/confidential with password "I love ansible".
* Run the data job stored in /opt/data-job.sh on each node in webservers asynchronously, without polling for status.
* Create tags for the following tasks:
  + base-install for httpd installation and service configuration.
  + vhost for virtual host deployment.
  + data-job to execute the asynchronous data job.

The Ansible control node has been configured for you and each webserver has already been configured for use with Ansible. The default inventory has been configured to include the webservers group and sample hosts.

## Learning Objectives

0 of 5 completed

Use ansible-vault to protect the confidential information.

Use ansible-vault to encrypt /home/ansible/confidential to protect the confidential information stored within using the password "I love ansible".

Run ansible-vault encrypt /home/ansible/confidential and supply the password "I love ansible".

Create a playbook that deploys httpd on webservers.

Create a playbook in /home/ansible/webserver.yml that deploys httpd on webservers. It should be tagged with base-install and contain a handler that restarts the httpd daemon that is flagged by both installation and service manipulation for httpd.

Create the file /home/ansible/webserver.yml and add the following content:

- hosts: webservers  
  become: yes  
  vars\_files:  
    - /home/ansible/confidential  
  tasks:  
    - name: install httpd  
      yum:  
        name: httpd  
        state: latest  
      notify: httpd service  
      tags:  
        - base-install  
  handlers:  
    - name: Restart and enable httpd  
      service:  
        name: httpd  
        state: restarted  
        enabled: yes  
      listen: httpd service

Deploy the templates stored on the control node to the webservers group.

Configure /home/ansible/webserver.yml to deploy the templates /home/ansible/vhost.conf.j2 and /home/ansible/htpasswd.j2 stored on the control node to the webservers group. httpd must restart on config change. The tasks should be tagged vhost.

Add the following text to /home/ansible/webserver.yml just before the handler section:

    - name: configure virtual host  
      template:  
        src: /home/ansible/vhost.conf.j2  
        dest: /etc/httpd/conf.d/vhost.conf  
      notify: httpd service  
      tags:  
        - vhost  
    - name: configure site auth  
      template:  
        src: /home/ansible/htpasswd.j2  
        dest: /etc/httpd/conf/htpasswd  
      notify: httpd service  
      tags:  
        - vhost

Asynchronously execute data-job on webservers.

Configure /home/ansible/webserver.yml to asynchronously execute /opt/data-job.sh located on webservers with a timeout of 600 seconds and no polling. The task should be tagged with data-job.

Add the following text to /home/ansible/webserver.yml just before the handler section: - name: run data job command: /opt/data-job.sh async: 600 poll: 0 tags: - data-job

Execute playbook to verify your playbook works correctly.

Execute playbook /home/ansible/webserver.yml to verify your playbook works correctly.

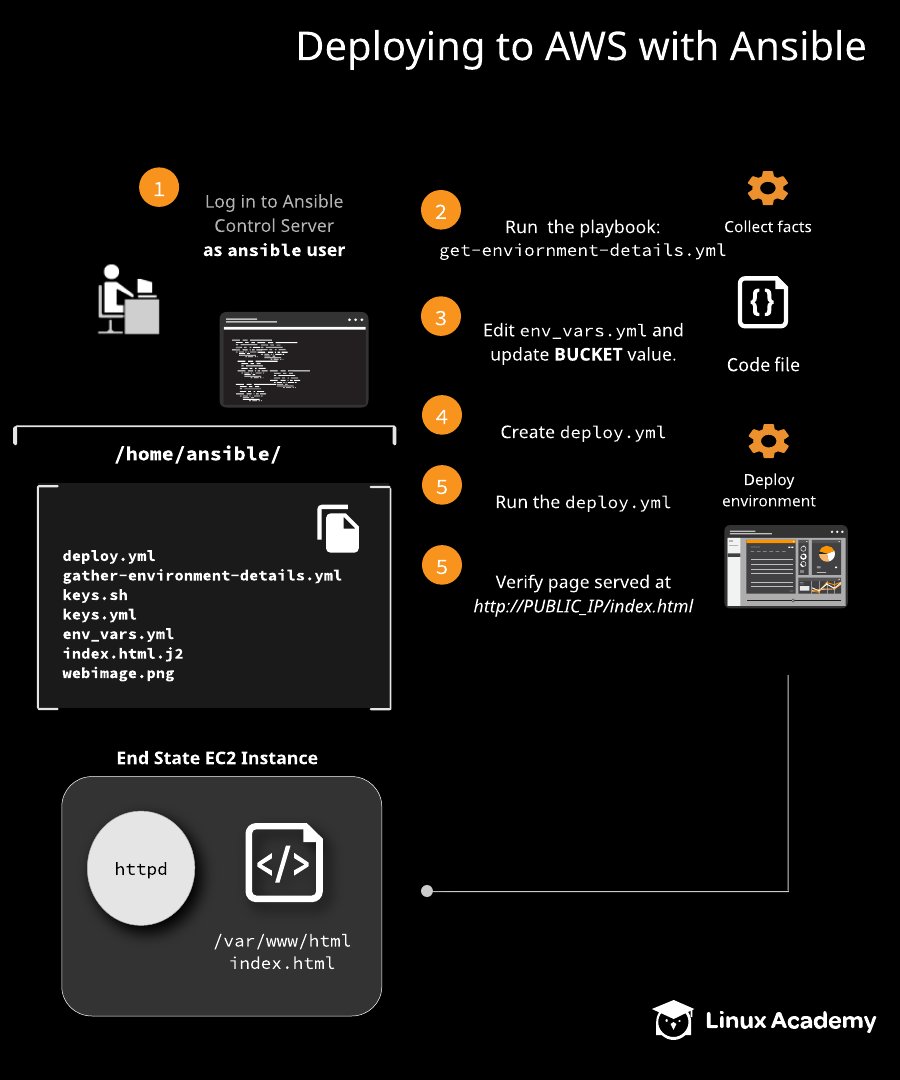
Run ansible-playbook --ask-vault-pass /home/ansible/webserver.yml from the control node providing the vault password "I love ansible".

\*\* ps -ef | grep data

PROFESSIONAL  
LAB 4.

4: Deploying to AWS with Ansible

GETTING MULTIPLE ERRORS



Deploying to AWS with Ansible  
Introduction  
This exercise provides a simple yet realistic task of deploying a basic website to dynamically provisioned AWS infrastructure. We will work with multiple AWS components through Ansible, and also perform basic web server configuration against a Linux host with Ansible. This exercise will help demonstrate a portion of the power provided by rolling cloud provisioning into deployment automation.  
  
Solution  
Log in to the provided lab server using the credentials provided:  
  
ssh cloud\_user@<PUBLIC\_IP\_ADDRESS>  
Become the ansible user:  
  
su - ansible  
Note: When copying and pasting code into Vim from the lab guide, first enter :set paste (and then i to enter insert mode) to avoid adding unnecessary spaces and hashes.  
  
Run the Provided Playbook to Collect Necessary Environment Details  
Run the following command:  
  
ansible-playbook /home/ansible/get-environment-details.yml  
This playbook populates our /home/ansible/env\_vars.yml with the necessary values.  
  
Replace the placeholder in /home/ansible/env\_vars.yml  
Open /home/ansible/env\_vars.yml:  
  
vim /home/ansible/env\_vars.yml  
Change the value placeholder to a globally unique S3 bucket name of your choice.  
  
Save and exit the file by pressing Escape followed by :wq.  
  
Create a Playbook and Add a Play  
Create the playbook:  
  
vim /home/ansible/deploy.yml  
Edit it such that it resembles the following:  
  
- hosts: localhost  
  gather\_facts: no  
  vars\_files:  
    - /home/ansible/env\_vars.yml  
  tasks:  
    - name: Create AWS key pair using Ansible's key.  
      local\_action: ec2\_key  
      args:  
        region: "{{ AWS\_REGION }}"  
        name: ansible\_keypair  
        key\_material: "{{ lookup('file', '/home/ansible/.ssh/id\_rsa.pub') }}"  
  
    - name: Provision instances  
      local\_action: ec2  
      args:  
        region: "{{ AWS\_REGION }}"  
        instance\_type: t2.micro  
        group: "{{ SEC\_GROUP\_NAME }}"  
        keypair: ansible\_keypair  
        image: "{{ AMI\_ID }}"  
        assign\_public\_ip: yes  
        vpc\_subnet\_id: "{{ DEFAULT\_VPC\_SUBNET }}"  
        wait: yes  
        instance\_tags:  
          type: web  
        count: 1  
      register: ec2  
  
    - name: Add host to inventory  
      add\_host:  
        hostname: "{{ item.public\_ip }}"  
        groupname: webservers  
        ansible\_ssh\_common\_args: "-o StrictHostKeyChecking=no"  
        ansible\_ssh\_private\_key\_file: /home/ansible/.ssh/id\_rsa  
      loop: "{{ ec2.instances }}"  
  
    - name: Create S3 Bucket  
      local\_action: aws\_s3  
      args:  
        bucket: "{{ BUCKET\_NAME }}"  
        mode: create  
        permission: public-read  
  
    - name: Add file to bucket  
      local\_action: aws\_s3  
      args:  
        bucket: "{{ BUCKET\_NAME }}"  
        mode: put  
        object: /webimage.png  
        src: /home/ansible/webimage.png  
        permission: public-read  
Add Another Play to Further Configure the New EC2 Instance  
Add another play to /home/ansible/deploy.yml that resembles the following:  
  
- hosts: webservers  
  gather\_facts: no  
  vars\_files:  
    - /home/ansible/env\_vars.yml  
  remote\_user: ec2-user  
  tasks:  
    - name: Wait for SSH to come up  
      wait\_for\_connection:  
        delay: 5  
        timeout: 90  
  
    - name: Collect instance facts  
      ec2\_metadata\_facts:  
  
    - name: Install HTTPD  
      become: yes  
      yum:  
        name: httpd  
        state: present  
  
    - name: Start HTTPD  
      become: yes  
      service:  
        name: httpd  
        state: started  
        enabled: yes  
  
    - name: Deploy content  
      become: yes  
      template:  
        src: /home/ansible/index.html.j2  
        dest: /var/www/html/index.html  
Run the Playbook  
Source our keys.sh file:  
  
source keys.sh  
Run the following command:  
  
ansible-playbook /home/ansible/deploy.yml  
It may take about 5–10 minutes to finish.  
  
Conclusion  
If the playbook ran with no errors, then we can fetch our instance's web content, using its public IP listed in the final output in the terminal, with:  
  
curl <PUBLIC\_IP\_ADDRESS>/index.html  
If all goes well, we've got a running web page. Congratulations!

## Additional Resources

In an effort to get senior management on board with using AWS for web content, we have been asked to produce a quick proof of concept. We need to demonstrate how quickly we can stand up a new website using EC2 for compute and S3 for content. A colleague has started the work but taken ill. We can pick up where they left off.

We have been provided an Ansible Control node and a sandbox AWS environment.

We'll start our work on the Ansible Control node:

* Run the provided playbook /home/ansible/get-environment-details.yml to complete another provided file, /home/ansible/env\_vars.yml with environment-specific configuration information.
* Edit /home/ansible/env\_vars.yml and update the placeholder value for the **BUCKET** variable. It should be a unique S3 bucket name that meets the [S3 naming restrictions](https://docs.aws.amazon.com/AmazonS3/latest/dev/BucketRestrictions.html).
* Create the playbook /home/ansible/deploy.yml to perform the following tasks:
  + Using the provided SSH key for the ansible system user in /home/ansible/.ssh/id\_rsa.pub, create a new AWS key pair named **ansible\_keypair**.
  + Create a new EC2 instance that meets the following requirements:
    - Use the subnet, security group, and AMI defined in /home/ansible/env\_vars.yml.
    - Use **ansible\_keypair**, created in the initial task, as the instance login key pair.
    - Set the instance type to **t2.micro**.
    - The instance should be deployed with a public IP.
    - The instance should have a tag with the key **type** and value **web**.
    - We only need a single instance for the purpose of the proof of concept.
  + Create a new S3 bucket meeting the following requirements:
    - The bucket should be named after the **BUCKET** variable in /home/ansible/env\_vars.yml.
    - The bucket should have public read permissions.
  + Upload the provided image /home/ansible/webimage.png to the new S3 bucket with an object name of /webimage.png
  + The playbook will need to configure the new EC2 instance as a web server in another play in the same playbook.
  + The following tasks must be performed on the EC2 instance after it is provisioned:
    - Install the httpd package using yum.
    - Start and enable the httpd package.
    - Deploy the provided template /home/ansible/index.html.j2 to /var/www/html/index.html. We will need to include the variable file /home/ansible/env\_vars.yml in the play to successfully deploy the template.
* Run the playbook /home/ansible/deploy.yml to build the environment.
* Verify the work by loading [*http:///index.html*](http://<PUBLIC_IP_ADDRESS_OF_NEW_EC2_INSTANCE>/index.html) in a web browser. If we did everything correctly, we should see a statement and an image in a browser.

The Ansible control node has been configured, and Ansible is installed. The control node also has a system user named ansible configured with SSH access keys and necessary system privileges.

An IAM user called ansible has been created on the provided AWS sandbox account. The access keys for the ansible IAM user are stored in /home/ansible/keys.sh and /home/ansible/keys.yml for whichever authentication method we prefer. The ansible IAM user has appropriate permissions to perform the required task.

The default Ansible inventory has been configured to include the Ansible control host as *localhost*.

## Learning Objectives

0 of 5 completed

Run the Provided Playbook `/home/ansible/get-environment-details.yml` to Collect Necessary Environment Details

After logging into the EC2 instance, become the ansible user:

su - ansible

The password is the same as it is for cloud\_user.

Run the following command:

ansible-playbook /home/ansible/get-environment-details.yml

Replace the word "placeholder" in `/home/ansible/env\_vars.yml` with a Unique S3 Bucket Name

1. Open /home/ansible/env\_vars.yml with a text editor.
2. Change the value placeholder to a unique S3 bucket name of your choosing.
   * Be sure to stick to the S3 naming conventions (<https://docs.aws.amazon.com/AmazonS3/latest/dev/BucketRestrictions.html> for details).

Create a Playbook and Add a Play per the Lab Instructions

1. Create /home/ansible/deploy.yml
2. Add an Ansible play that configures your EC2 key pair, EC2 instance, and S3 bucket.
3. Use the provided variable file for required parameter configuration.

Add Another Play to Further Configure the New EC2 Instance

1. Add another play that will:
   * Configure the new EC2 instance.
   * Install the httpd package
   * Start and enable the httpd service.
   * Deploy the provided template file into /var/www/html.
2. Use /home/ansible/env\_vars.yml for required parameter configuration values.

Run `/home/ansible/deploy.yml` to Perform the Required Tasks

Run the following command:

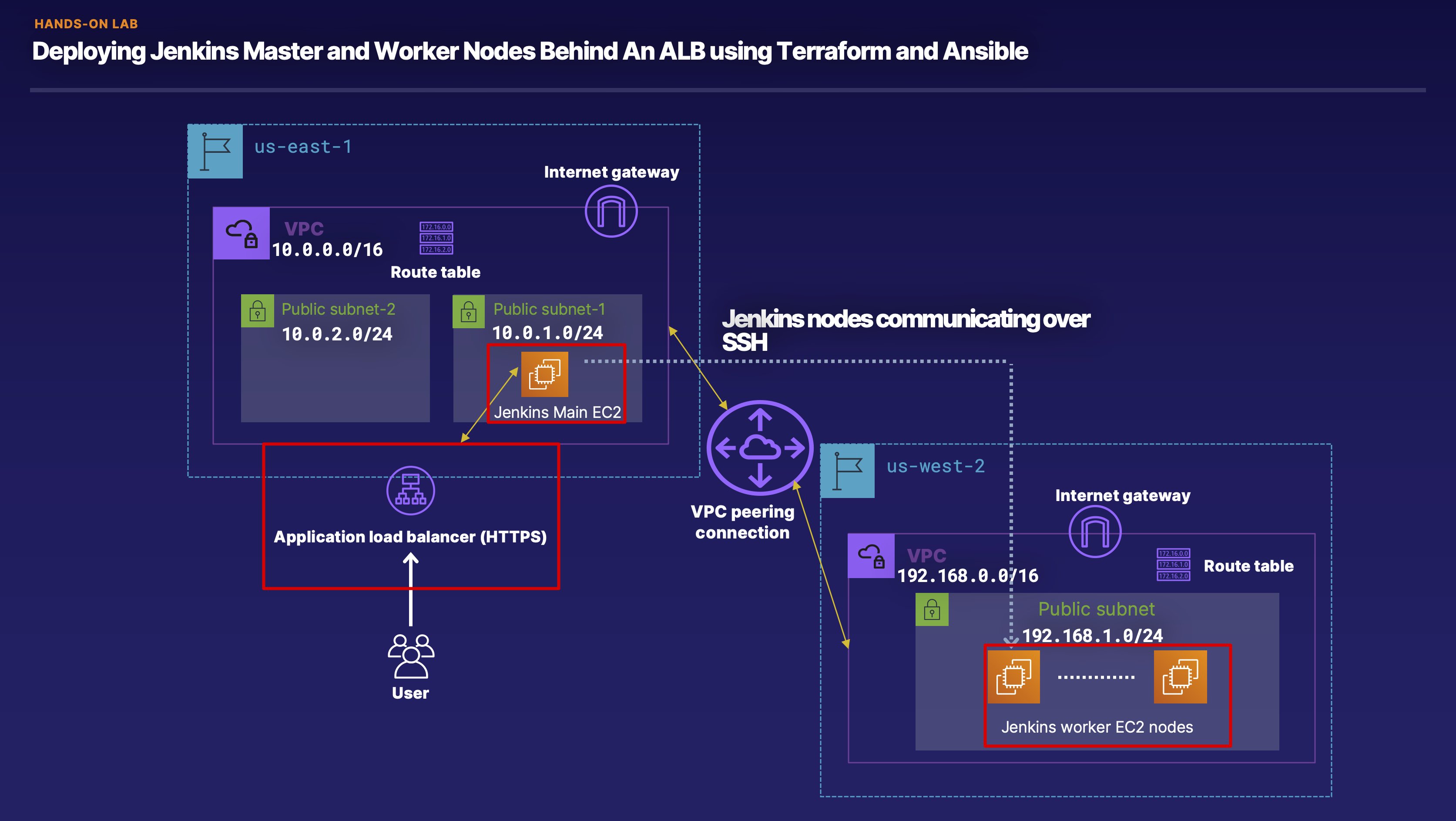
ansible-playbook /home/ansible/deploy.yml

IMP  GIT REPO FOR TERRAFORM

PROFESSIONAL

LAB 5

5 : Deploying Jenkins Master and Worker Nodes in AWS Behind an ALB Using Terraform and Ansible.



[cloud\_user@ip-10-0-1-160 ~]$ curl 169.254.169.254/latest/user-data#!/bin/bashecho 'cloud\_user:+Q8(Zh7e' | chpasswdyum -y install git wget jq unzip python3 python3-pippip3 install boto3 botocore ansiblepip3 install —upgrade --force pipwget -P /tmp -c <https://releases.hashicorp.com/terraform/0.12.13/terraform_0.12.13_linux_amd64.zip>

alternatives —set python /usr/bin/python3

unzip /tmp/terraform\_0.12.13\_linux\_amd64.zip -d /bin/

sudo -iu cloud\_user pip3 install ansible —usersudo -iu cloud\_user pip3 install boto3 —user

sudo -iu cloud\_user aws configure set default.region us-east-1/opt/aws/bin/cfn-signal —exit-code 0 —resource TerraformController —region us-east-1 —stack cfst-2896-1620784f08042e15523bb68af58ae36a[cloud\_user@ip-10-0-1-160 ~]$[cloud\_user@ip-10-0-1-160 ~]$[cloud\_user@ip-10-0-1-160 ~]$[cloud\_user@ip-10-0-1-160 ~]$ sudo cat /etc/os-releaseNAME="Amazon Linux"

VERSION="2"

ID="amzn"

ID\_LIKE="centos rhel fedora"

VERSION\_ID="2"

PRETTY\_NAME="Amazon Linux 2"

ANSI\_COLOR="0;33"

CPE\_NAME="cpe:2.3:o:amazon:amazon\_linux:2"

HOME\_URL="<https://amazonlinux.com/>“

SUPPORT\_END="2025-06-30"

[cloud\_user@ip-10-0-1-160 ~]$[cloud\_user@ip-10-0-1-160 ~]$

[cloud\_user@ip-10-0-1-160 ~]$

[cloud\_user@ip-10-0-1-160 ~]$ ansible —version

[DEPRECATION WARNING]: Ansible will require Python 3.8 or newer on the controller starting with Ansible 2.12.

Current version: 3.7.16 (default, Apr 2 2024, 23:33:39) [GCC 7.3.1 20180712 (Red Hat 7.3.1-17)]. This feature will

be removed from ansible-core in version 2.12. Deprecation warnings can be disabled by setting

deprecation\_warnings=False in ansible.cfg.

ansible [core 2.11.12]

config file = None

configured module search path = ['/home/cloud\_user/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']

ansible python module location = /usr/local/lib/python3.7/site-packages/ansible

ansible collection location = /home/cloud\_user/.ansible/collections:/usr/share/ansible/collections

executable location = /usr/local/bin/ansible

python version = 3.7.16 (default, Apr 2 2024, 23:33:39) [GCC 7.3.1 20180712 (Red Hat 7.3.1-17)]

jinja version = 3.1.4

libyaml = True

[cloud\_user@ip-10-0-1-160 ~]$[cloud\_user@ip-10-0-1-160 ~]$

[cloud\_user@ip-10-0-1-160 ~]$

[cloud\_user@ip-10-0-1-160 ~]$ terraform -v

Terraform v0.12.13

Your version of Terraform is out of date! The latest version

is 1.8.2. You can update by downloading from www.terraform.io/downloads.html

[cloud\_user@ip-10-0-1-160 ~]$

# Deploying Jenkins Master and Worker Nodes in AWS Behind an ALB Using Terraform and Ansible

## Introduction

## In this hands-on lab, the student will deploy Jenkins master and worker nodes on AWS EC2 instances across regions through Terraform and manage the software and integration with Ansible.

## Solution

### Log in to the Terraform Controller Node EC2 Instance

Find the details for logging in to the Terraform Controller node provided by the hands-on lab interface and log in to the node using SSH:

ssh cloud\_user@<IP-OF-TERRAFORM-CONTROLLER>

**Note**: This instance already has an EC2 instance profile (role) attached to it and has all necessary AWS API permissions required for this lab. It also has the AWS CLI set up and configured with the AWS account attached to this lab, for which the console login credentials are also provided in the lab interface page once the lab spins up.

After logging in, verify the version of Terraform installed (should be 12.13). Execute the following command to check:

terraform version

### Clone the GitHub Repo for Terraform Code

### Use the git command to clone the GitHub repo which has the Terraform code for deploying the solution of this lab. [GitHub repo URL](https://github.com/linuxacademy/content-deploying-to-aws-ansible-terraform.git).

Execute the following command:

git clone https://github.com/ACloudGuru-Resources/content-deploying-to-aws-ansible-terraform.git

Change to the directory for lab Terraform code:

cd content-deploying-to-aws-ansible-terraform/lab\_jenkins\_master\_worker

Examine the contents of the directory you're in:

ls

### Run the gen\_ssh\_key.yaml Ansible Playbook to Generate SSH Key Pair

Run the Ansible Playbook:

ansible-playbook ansible\_templates/gen\_ssh\_key.yaml

This Ansible Playbook will generate an SSH key pair for you user cloud\_user which is required for deploying EC2 key pairs in our code.

**Note**: Alternatively, you may also run the following Linux command to do the same:

ssh-keygen -t rsa

When this command prompts for input, **keep pressing enter** until you're returned to the prompt. **Do not enter a passphrase**.

### Deploy the Terraform Code

Initialize the Terraform directory you changed into to download the required provider

terraform init

Ensure Terraform code is formatted properly:

terraform fmt

Ensure code has proper syntax and no errors:

terraform validate

See the execution plan and note the number of resources that will be created:

terraform plan

Deploy resources:

terraform apply

Enter yes when prompted.

After terraform apply has run successfully, you can either use the AWS CLI on the Controller node to list and describe created resources or you can log in to the AWS Console to verify and investigate created resources.

After a successful terraform apply, you will get the DNS URL of the ALB. Test it out to see if you can reach your Jenkins deployment.

Jenkins credentials:

* + username: admin
  + password: password

Finally, on the Terraform Controller node CLI, delete all resources which were created and ensure that it runs through successfully.

terraform destroy

## Conclusion

## Congratulations — you've completed this hands-on lab!

## Additional Resources

Please ensure that only regions **us-east-1** and **us-west-2** are used to deploy resources.

You're an automation engineer at CloudWare Inc. and you have been tasked with deploying a CI/CD application across 2 AWS regions. You've been provided with the Git repo URL which hosts all the required code here:

https://github.com/ACloudGuru-Resources/content-deploying-to-aws-ansible-terraform.git

The code that is relevant to this module of the project resides within this folder:

content-deploying-to-aws-ansible-terraform/lab\_jenkins\_master\_worker

The system provided to you is a Terraform controller node that comes pre-configured with Terraform and Ansible. All you need to do is generate an SSH key pair on this controller node to be used by Ansible for bootstrapping and then deploy the Terraform code!

**Hint**: For generating the SSH key pair once you are within the correct folder in the code base, use the following command:

ansible-playbook ansible\_templates/gen\_ssh\_key.yaml

The end result should be that you'll have an ALB DNS name that your Jenkins application will be accessible from.

## Learning Objectives

0 of 4 completed

Log in to the Terraform Controller Node EC2 Instance

Find the details for logging in to the Terraform Controller node provided by the hands-on lab interface and log in to the node using SSH:

ssh cloud\_user@<IP-OF-TERRAFORM-CONTROLLER>

**Note**: This instance already has an EC2 instance profile (role) attached to it and has all necessary AWS API permissions required for this lab. It also has the AWS CLI set up and configured with the AWS account attached to this lab, for which the console login credentials are also provided in the lab interface page once the lab spins up.

After logging in, verify the version of Terraform installed (should be 12.13). Execute the following command to check:

terraform version

Clone the GitHub Repo for Terraform Code

Use the git command to clone the GitHub repo which has the Terraform code for deploying the solution of this lab. [GitHub repo URL](https://github.com/linuxacademy/content-deploying-to-aws-ansible-terraform.git).

Execute the following command:

git clone https://github.com/ACloudGuru-Resources/content-deploying-to-aws-ansible-terraform.git

Change to the directory for lab Terraform code:

cd content-deploying-to-aws-ansible-terraform/lab\_jenkins\_master\_worker

Examine the contents of the directory you're in:

ls

Run the `gen\_ssh\_key.yaml` Ansible Playbook to Generate SSH Key Pair

Run the Ansible Playbook:

ansible-playbook ansible\_templates/gen\_ssh\_key.yaml

This Ansible Playbook will generate an SSH key pair for you user cloud\_user which is required for deploying EC2 key pairs in our code.

**Note**: Alternatively, you may also run the following Linux command to do the same:

ssh-keygen -t rsa

When this command prompts for input, **keep pressing enter** until you're returned to the prompt. **Do not enter a passphrase**.

Deploy the Terraform Code

Initialize the Terraform directory you changed into to download the required provider

terraform init

Ensure Terraform code is formatted properly:

terraform fmt

Ensure code has proper syntax and no errors:

terraform validate

See the execution plan and note the number of resources that will be created:

terraform plan

Enter yes when prompted.

Deploy resources:

terraform apply

Enter yes when prompted.

After terraform apply has run successfully, you can either use the AWS CLI on the Controller node to list and describe created resources or you can log in to the AWS Console to verify and investigate created resources.

After a successful terraform apply, you will get the DNS URL of the ALB. Test it out to see if you can reach your Jenkins deployment.

Jenkins credentials:

* + username: admin
  + password: password

Finally, on the Terraform Controller node CLI, delete all resources which were created and ensure that it runs through successfully.

terraform destroy