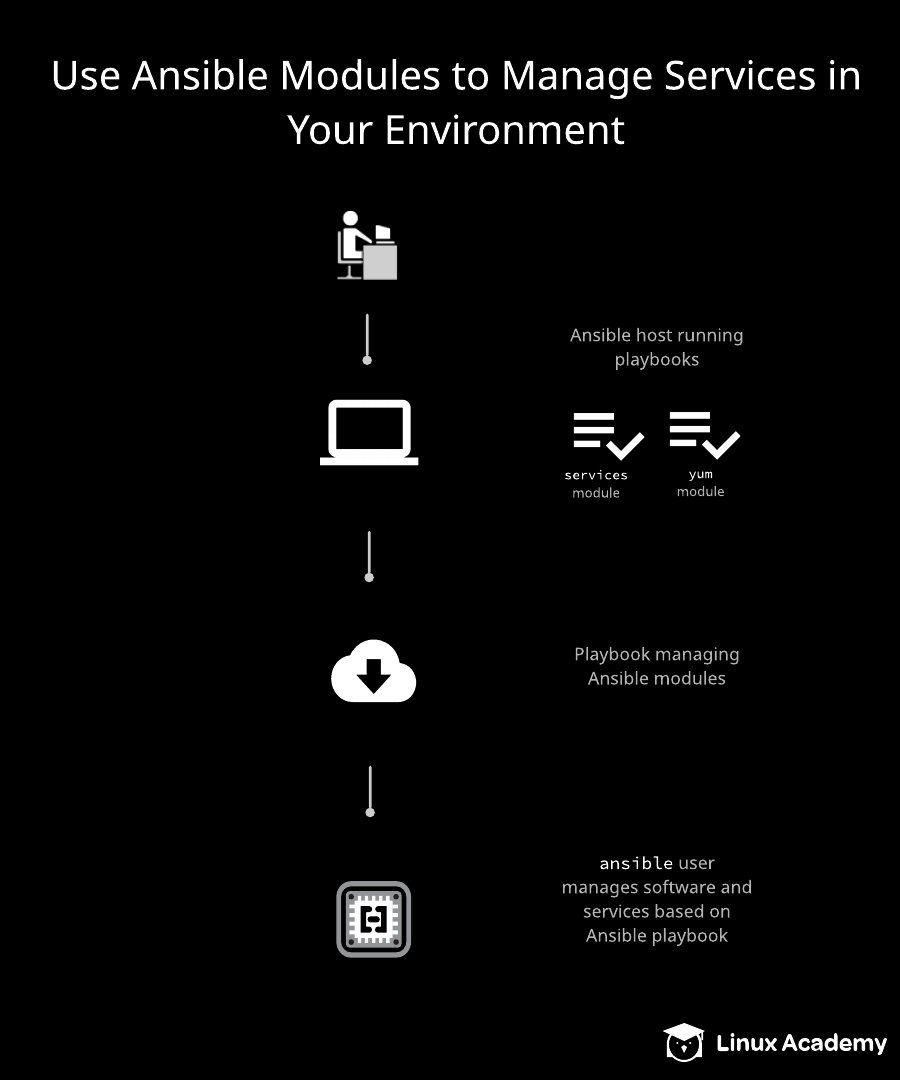
# . Ansible Practioner 23-36

22 :

## Using Ansible Modules to Manage Services in Your Environment.



# Using Ansible Modules to Manage Services in Your Environment

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

## Introduction

## Connecting to a single server to restart a service is tedious. Now imagine having to do it on a hundred servers. That kind of task is what tools like Ansible are great at. This lab will help reinforce some of the lesson on managing services.

## The Scenario

## There are reports of some servers not starting their required services when they rebooting after getting patches last night. Our task is to ensure all postgresql and httpd services are started and enabled throughout the environment. Web servers do not have PostgreSQL installed, while database and administrative servers do. Web servers only have httpd installed.

In addition, because the databases sometimes take a significant amount of time to start, we've been asked to make sure these restarts are asynchronous.

## Logging In

## Use the credentials provided on the hands-on lab page to get into Server1 to begin with. Since we need root privileges, let's just run sudo -i right off and become root.

## Start httpd on the Clients in the webserver Group

## We need to create services.yml and make a section for starting the httpd service. When we're done, it should look like this:

---  
- name: Start httpd  
  hosts: webservers  
  become: yes  
  
  tasks:  
  - service:  
     name: httpd  
     state: started  
     enabled: yes

## Start PostgreSQL on Clients in the dbservers and adminservers Groups

## While we're in there, let's add a section to start the database service. Because the services need to start asynchronously, we have to use the systemd module:

- name: Start postgresql  
  hosts: dbservers:admins  
  become: yes  
  
  tasks:  
  - systemd:  
     name: postgresql  
     state: started  
     no\_block: yes

## Run the Playbook

## The playbook is done, now let's see what happens when we run it. Execute ansible-playbook services.yml. We'll see output showing us that the appropriate services were in fact started on the correct servers.

If we run it again, we'll get all **ok:** messages, because the playbook is only starting services that aren't currently running.

## Conclusion

## We did it. Now, managing a fleet of many servers, as far as starting services goes, is a whole lot easier. Congratulations!

## Additional Resources

Notice: Ansible is installed as the root user, so please work on all tasks after elevating to the root user.

There are reports of some servers not starting their required services when they rebooting after getting patches last night. Our task is to ensure all postgresql and httpd services are started and enabled throughout the environment. Web servers do not have PostgreSQL installed, while database and administrative servers do. Web servers only have httpd installed.

In addition, because the databases sometimes take a significant amount of time to start, we've been asked to make sure these restarts are asynchronous.

## Learning Objectives

0 of 2 completed

Start httpd on the Clients in the webserver Group

Create and edit services.yml, and add a section to start httpd. The block will look like this:

---  
- name: Start httpd  
  hosts: webservers  
  become: yes  
  
  tasks:  
  - service:  
     name: httpd  
     state: started  
     enabled: yes

Start PostgreSQL on Clients in the dbservers and adminservers Groups

Edit services.yml further and add a section to start PostgreSQL. Because the services need to start asynchronously, we have to use the systemd module:

- name: Start postgresql  
  hosts: dbservers:admins  
  become: yes  
  
  tasks:  
  - systemd:  
     name: postgresql  
     state: started  
     no\_block: yes  
       
     

Host Machine :

[root@Server1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/usr/bin/hostnamectl set-hostname Server1

/usr/bin/yum install wget git -y

/bin/echo 'source /root/ansible/hacking/env-setup -q' >> /root/.bashrc

/bin/ssh-keygen -N '' -t rsa -f ~/.ssh/id\_rsa

/bin/cat /root/.ssh/id\_rsa.pub >> /home/ansible/.ssh/authorized\_keys

/bin/echo 'python\_interpreter=auto' >> /etc/ansible/ansible.cfg

/bin/echo 'host\_key\_checking = False' >> /etc/ansible/ansible.cfg

/sbin/alternatives —set python /usr/bin/python3

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/privkey> -O /tmp/pkey

/bin/cat /tmp/pkey > /root/.ssh/id\_rsa

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/authkey> -O /tmp/akey

/bin/cat /tmp/akey > /root/.ssh/id\_rsa.pub

/bin/cat /tmp/akey > /home/ansible/.ssh/id\_rsa.pub

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/etchosts> -O /tmp/hosts

/bin/cat /tmp/hosts > /etc/hosts

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/inventory> -O /tmp/inv

/bin/cat /tmp/inv > /etc/ansible/hosts

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

/usr/bin/echo 'cloud\_user:m5^#j)(K' | /usr/sbin/chpasswd

/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-2281-260e748e5f55c9e8944d9689ee49688f —resource PublicInstance —region us-east-1

[root@Server1 cloud\_user]# sudo cat /etc/os-release

NAME="Red Hat Enterprise Linux"

VERSION="8.7 (Ootpa)"

ID="rhel"

ID\_LIKE="fedora"

VERSION\_ID="8.7"

PLATFORM\_ID="platform:el8"

PRETTY\_NAME="Red Hat Enterprise Linux 8.7 (Ootpa)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:redhat:enterprise\_linux:8::baseos"

HOME\_URL="<https://www.redhat.com/>“

DOCUMENTATION\_URL="<https://access.redhat.com/documentation/red_hat_enterprise_linux/8/>“

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

REDHAT\_BUGZILLA\_PRODUCT="Red Hat Enterprise Linux 8"

REDHAT\_BUGZILLA\_PRODUCT\_VERSION=8.7

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

REDHAT\_SUPPORT\_PRODUCT\_VERSION="8.7"

[root@Server1 cloud\_user]# ansible —version

ansible 2.10.0.dev0

config file = /etc/ansible/ansible.cfg

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

ansible python module location = /root/ansible/lib/ansible

executable location = /root/ansible/bin/ansible

python version = 3.6.8 (default, Jan 11 2023, 08:43:50) [GCC 8.5.0 20210514 (Red Hat 8.5.0-16)]

[root@Server1 cloud\_user]#

Server 1 :

[root@WebServer1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/usr/bin/hostnamectl set-hostname WebServer1

/usr/bin/yum install wget httpd -y

/sbin/useradd ansible

/usr/bin/echo 'ansible:m5^#j)(K' | /usr/sbin/chpasswd

/sbin/alternatives —set python /usr/bin/python3

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/etchosts> -O /tmp/hosts

/bin/cat /tmp/hosts > /etc/hosts

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/inventory> -O /tmp/inv

/bin/cat /tmp/inv > /etc/ansible/hosts

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/authkey> -O /tmp/key

/bin/mkdir /home/ansible/.ssh

/usr/bin/chmod 0700 /home/ansible/.ssh/

/bin/cat /tmp/key > /home/ansible/.ssh/authorized\_keys

/usr/bin/chmod 0600 /home/ansible/.ssh/authorized\_keys

/usr/bin/chown ansible.ansible /home/ansible -R

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

/usr/bin/echo 'cloud\_user:m5^#j)(K' | /usr/sbin/chpasswd

/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-2281-260e748e5f55c9e8944d9689ee49688f —resource PublicInstance2 —region us-east-1

[root@WebServer1 cloud\_user]# sudo cat /etc/os-release

NAME="Red Hat Enterprise Linux"

VERSION="8.9 (Ootpa)"

ID="rhel"

ID\_LIKE="fedora"

VERSION\_ID="8.9"

PLATFORM\_ID="platform:el8"

PRETTY\_NAME="Red Hat Enterprise Linux 8.9 (Ootpa)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:redhat:enterprise\_linux:8::baseos"

HOME\_URL="<https://www.redhat.com/>“

DOCUMENTATION\_URL="<https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8>“

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

REDHAT\_BUGZILLA\_PRODUCT="Red Hat Enterprise Linux 8"

REDHAT\_BUGZILLA\_PRODUCT\_VERSION=8.9

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

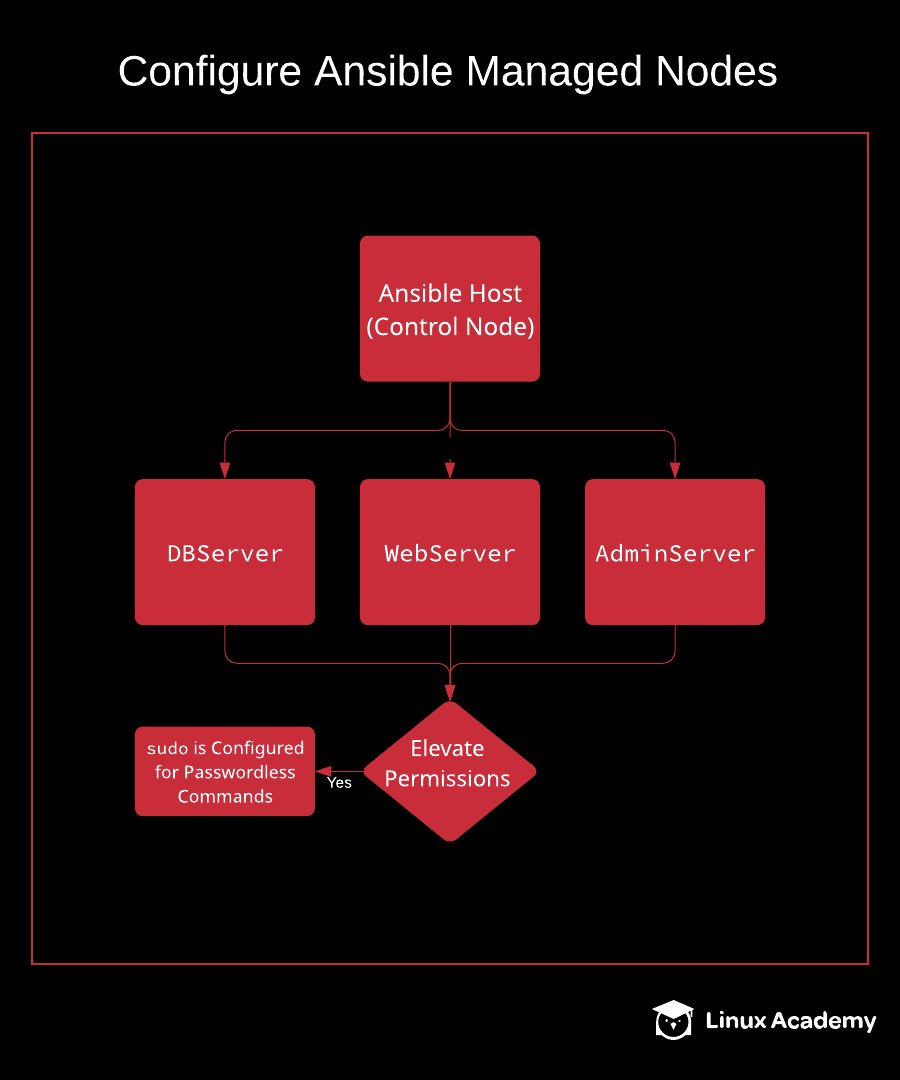
REDHAT\_SUPPORT\_PRODUCT\_VERSION="8.9"

[root@WebServer1 cloud\_user]# ansible —version

bash: ansible: command not found...

23

## Configure Ansible Managed Nodes



# Configure Ansible Managed Nodes

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

## Introduction

## The power of Ansible comes in being able to remotely manage a fleet of servers. For that to happen though we need to configure our managed nodes. In this lab we'll practice configuring these nodes.

## The Scenario

## We're in the midst of a proof of concept project. We have gotten approval for the combination of inventories and facts, software management, modules, playbooks, and variables in a single-server environment. Now we can move on to configuring this in a multi-server setup. We've been given three servers, and some tasks to set up for each: WebServer1, DBServer1, and AdminServer1.

We've got to set the three servers up so that Ansible can connect and run commands as root without entering a password.

## Logging In

## Use the credentials provided on the hands-on lab page to get into Server1 to begin with. Since we need root privileges, let's just run sudo -i right off and become root.

## Set up the Ansible Inventory

## We need to make sure Ansible has the new servers in its inventory so let's edit /etc/ansible/hosts and add lines similar to the following:

[webservers]  
WebServer1  
  
[dbservers]  
DBServer1  
  
[admins]  
AdminServer1

## Set up Passwordless SSH Between Nodes

## If we take a look in /root/.ssh/, we'll see that there's already have an id.rsa file in there. Rather than create a new one, let's just re-use what we have.

Copy that key to one of the new servers:

ssh-copy-id ansible@WebServer1

Enter the password at the prompt, the key will be copied over. Repeat this with the other two servers, DBServer1 and AdminServer1. To test, we can just try logging into either of the other two servers with no password.

## Configure Passwordless Privilege Escalation

## Connect to WebServer1 as cloud\_user. From there, use sudo to escalate and have root privileges.

As root, run visudo. Add ansible ALL=(ALL) NOPASSWD: ALL to the end of the file, then get out and repeat this process on the other two servers.

## Ensure Correct Configuration

## Make sure that everything was done correctly by running ansible all -m ping --become. This will have Ansible attempt to connect to all the other servers, and escalate privileges once it does.

## Conclusion

## We've done it. We set Ansible up so that it can get into any of the three servers, and have privileges to do whatever we need it to. Congratulations!

## Additional Resources

Notice: Ansible is installed as the root user, so please work on all tasks after elevating to the root user.

We're in the midst of a proof of concept project. We have gotten approval for the combination of inventories and facts, software management, modules, playbooks, and variables in a single-server environment. Now we can move on to configuring this in a multi-server setup. We've been given three servers, and some tasks to set up for each: WebServer1, DBServer1, and AdminServer1.

We've got to set the three servers up so that Ansible can connect and run commands as root without entering a password.

## Learning Objectives

0 of 4 completed

Set up the Ansible Inventory

We need to make sure Ansible has the new servers in its inventory. Edit /etc/ansible/hosts and add lines similar to the following:

[webservers]  
WebServer1  
  
[dbservers]  
DBServer1  
  
[admins]  
AdminServer1

Set up Passwordless SSH Between Nodes

Note that we already have a SSH key in root's .ssh directory. We don't have to create a new one though. Let's just re-use what we have.

Copy that key to one of the new servers:

ssh-copy-id ansible@WebServer1

Put in the password and the key will be copied over. Repeat this with the other two servers.

Configure Passwordless Privilege Escalation

Connect to WebServer1 as cloud\_user. From there, use sudo to escalate and have root privileges.

As root, run visudo. Add ansible ALL=(ALL) NOPASSWD: ALL to the end of the file.

Repeat this on the other two servers.

Ensure Correct Configuration

Make sure that everything was done correctly by running ansible all -m ping --become to have Ansible attempt to connect to all the other servers and escalate privileges.

Ansible Host :

[root@Server1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/usr/bin/hostnamectl set-hostname Server1

/usr/bin/yum install wget git -y

/bin/echo 'source /root/ansible/hacking/env-setup -q' >> /root/.bashrc

/bin/ssh-keygen -N '' -t rsa -f ~/.ssh/id\_rsa

/bin/cat /root/.ssh/id\_rsa.pub >> /home/ansible/.ssh/authorized\_keys

/bin/echo 'python\_interpreter=auto' >> /etc/ansible/ansible.cfg

/sbin/alternatives —set python /usr/bin/python3

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/etchosts> -O /tmp/hosts

/bin/cat /tmp/hosts > /etc/hosts

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

/usr/bin/echo 'cloud\_user:#ra\_2L%#' | /usr/sbin/chpasswd

/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-2250-923cfc45a0f33aadae7b6aad68f89775 —resource PublicInstance —region us-east-1

[root@Server1 cloud\_user]# sudo cat /etc/os-release

NAME="Red Hat Enterprise Linux"

VERSION="8.7 (Ootpa)"

ID="rhel"

ID\_LIKE="fedora"

VERSION\_ID="8.7"

PLATFORM\_ID="platform:el8"

PRETTY\_NAME="Red Hat Enterprise Linux 8.7 (Ootpa)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:redhat:enterprise\_linux:8::baseos"

HOME\_URL="<https://www.redhat.com/>“

DOCUMENTATION\_URL="<https://access.redhat.com/documentation/red_hat_enterprise_linux/8/>“

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

REDHAT\_BUGZILLA\_PRODUCT="Red Hat Enterprise Linux 8"

REDHAT\_BUGZILLA\_PRODUCT\_VERSION=8.7

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

REDHAT\_SUPPORT\_PRODUCT\_VERSION="8.7"

[root@Server1 cloud\_user]# ansible —version

ansible 2.10.0.dev0

config file = /etc/ansible/ansible.cfg

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

ansible python module location = /root/ansible/lib/ansible

executable location = /root/ansible/bin/ansible

python version = 3.6.8 (default, Jan 11 2023, 08:43:50) [GCC 8.5.0 20210514 (Red Hat 8.5.0-16)]

[root@Server1 cloud\_user]#

[root@Server1 cloud\_user]#

Server 1 :

[root@WebServer1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/usr/bin/hostnamectl set-hostname WebServer1

/usr/bin/yum install wget -y

/sbin/useradd ansible

/usr/bin/echo 'ansible:#ra\_2L%#' | /usr/sbin/chpasswd

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/etchosts> -O /tmp/hosts

/bin/cat /tmp/hosts > /etc/hosts

/usr/bin/echo 'cloud\_user:#ra\_2L%#' | /usr/sbin/chpasswd

/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-2250-923cfc45a0f33aadae7b6aad68f89775 —resource PublicInstance2 —region us-east-1

[root@WebServer1 cloud\_user]# sudo cat /etc/os-release

NAME="Red Hat Enterprise Linux"

VERSION="8.9 (Ootpa)"

ID="rhel"

ID\_LIKE="fedora"

VERSION\_ID="8.9"

PLATFORM\_ID="platform:el8"

PRETTY\_NAME="Red Hat Enterprise Linux 8.9 (Ootpa)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:redhat:enterprise\_linux:8::baseos"

HOME\_URL="<https://www.redhat.com/>“

DOCUMENTATION\_URL="<https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8>“

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

REDHAT\_BUGZILLA\_PRODUCT="Red Hat Enterprise Linux 8"

REDHAT\_BUGZILLA\_PRODUCT\_VERSION=8.9

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

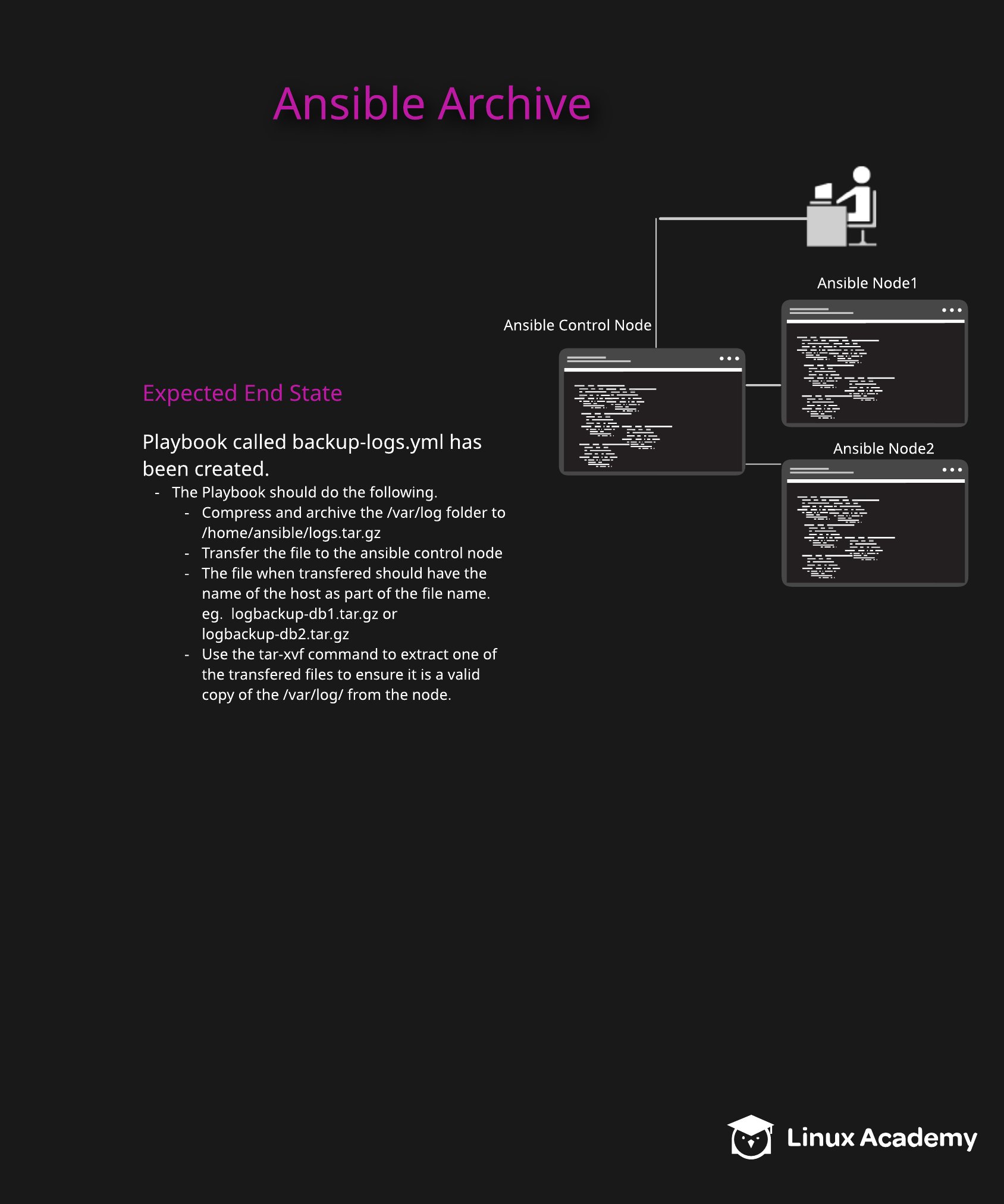
REDHAT\_SUPPORT\_PRODUCT\_VERSION="8.9"

[root@WebServer1 cloud\_user]# ansible —version

bash: ansible: command not found...

24

## Ansible Archiving



# Ansible Archiving

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

## Introduction

## In this Hands-On Lab we will use a playbook with the archive module for backup purposes.

Note: Ansible has been setup and configured for your use.

## Solution

## To access the lab environment, log into the control node as cloud\_user, using the IP address and login credentials provided in the hands-on lab page.

### Sign onto the Ansible Control Node Server as cloud\_user and Change to the ansible User. Test to Ensure Ansible Has Been Set up for You

### Sign into the server called Ansible Control Node using the cloud\_user and change to the ansible user via the su - ansible command.

Test that Ansible is working via an ad-hoc command.

A possible command to use is the following:

ansible all -m ping

### Create an Ansible Playbook Called backup-logs.yml. The Playbook Should Create a Compressed Backup of the /var/log Folder on Each of the Nodes

### Before we do anything, let's have a look at our hosts file:

cat /etc/ansible/hosts

In there, we'll see a couple of hosts (down near the bottom of the file):

[dbsystems]  
db1  
db2

The Ansible playbook should create a backup of the /var/log folder to the /home/ansible folder and call the compressed copy of the log files, logs.tar.gz. To make this happen, we've got to create a YAML file (we'll call it backup-logs.yml. Here's that file:

---  
- hosts: dbsystems  
  user: ansible  
  become: yes  
  gather\_facts: no  
  tasks:  
  
  - name: Compress directory /var/log/ into /home/ansible/logs.tar.gz  
    archive:  
      path: /var/log  
      dest: /home/ansible/logs.tar.gz  
      owner: ansible  
      group: ansible  
      format: gz

Now let's run that playbook and see what happens with: ansible-playbook backup-logs.yml. If everything looks good, we can log in and check it: ssh node1. run a quick ll, and we'll see that there is a logs.tar.gz sitting in our home directory. Go ahead and get out of node1 (with the exit command) and we'll be back in the control1 machine.

### Using Another Ansible Module, Copy the Compressed Log Files to the Control Node. Ensure the Name Is Unique and Related to Which Node It Has Come From

### So, we're sure that the files we need do in fact exist on the node, now we need to bring them back to control1. We'll do that with the **fetch** module, fetch from each node a copy of the compressed log file back to the control node.

We've got to go out and grab the remote logs. Let's add this to backup-logs.yml:

  - name: Fetch the log files to the local filesystem  
    fetch:  
      src: /home/ansible/logs.tar.gz  
      dest: logbackup-{{ inventory\_hostname }}.tar.gz  
      flat: yes

The whole file should look like this when we're done:

---  
- hosts: dbsystems  
  user: ansible  
  become: yes  
  gather\_facts: no  
  tasks:  
  
  - name: Compress directory /var/log/ into /home/ansible/logs.zip  
    archive:  
      path: /var/log  
      dest: /home/ansible/logs.tar.gz  
      owner: ansible  
      group: ansible  
      format: gz  
  
  - name: Fetch the log files to the local filesystem  
    fetch:  
      src: /home/ansible/logs.tar.gz  
      dest: logbackup-{{ inventory\_hostname }}.tar.gz  
      flat: yes

Now when we run ansible-playbook backup-logs.yml, and then run a quick ll, we should see tar.gz files sitting there that came from both db nodes.

### Uncompress One of the Files That Has Been Downloaded and Make Sure It Is a Copy of the Files That Are in the /var/log Folder on the Node

### To *really* check what we got, let's extract contents from one of the archived files that has been copied to the control node, and test to ensure it is a valid copy of the files as they exist on the node:

tar -xvf logbackup-db1.tar.gz

Once that's done extracting, run ll log (log being the directory that was extracted from the tar.gz file), and we should see the same thing we'd see when running ll /var/log on the relevant node.

## Conclusion

## That's really all there is to it. We've created an Ansible playbook that grabs log file directories from two different node servers. Congratulations!

## Additional Resources

## The Scenario:

## We are doing some work for a charity that does not have the money to pay for a commercial backup solution. We have told them we are able to back up some folders and transfer them to a central location using Ansible. We have been asked to show a proof of concept by archiving and downloading a folder from all the servers.

## The Task:

## In this hands-on lab we will make use of the archive module to archive a folder on our Nodes and transfer the resulting compressed file to our Ansible control node.

To summarize, using an Ansible playbook you must do the following:

1. Create a playbook called backup-logs.yml
2. The playbook should archive the /var/log folder on each node to a file in the ansible user's home directory.
3. Transfer the archived file to the control node with the name logbackup- and the node's hostname as part of the file name.

Here are some important things to take into consideration while completing each task:

* For convenience, Ansible is already on the control node. If we connect to the server by clicking on the Public IP address in a browser, we need to make sure we change to the ansible user with the su - ansible command.
* The user ansible is present on all servers, with appropriate shared keys for access to managed servers from the control node. Make sure to use the ansible user to complete the commands.
* The ansible user has the same password as cloud\_user.
* The default Ansible inventory has been configured for with the appropriate hosts and groups.
* /etc/hosts entries are present on control1 for the managed servers.

## Learning Objectives

0 of 4 completed

Sign onto the Ansible Control Node Server as cloud\_user and Change to the ansible User. Test to Ensure Ansible Has Been Set up for You

Sign into the server called Ansible Control Node using the cloud\_user and change to the ansible user via the su - ansible command.

Test that ansible is working via an ad-hoc command.

A possible command to use is the following:

ansible all -m ping

Note: It may take several minutes after the start of the lab for the test to work as expected.

Create an Ansible Playbook Called backup-logs.yml. The Playbook Should Create a Compressed Backup of the /var/log Folder on Each of the Nodes

The Ansible playbook should create a backup of the /var/log folder to the /home/ansible folder and call the compressed copy of the log files, logs.tar.gz.

Using Another Ansible Module, Copy the Compressed Log Files to the Control Node. Ensure the Name Is Unique and Related to Which Node It Has Come From

Using the **fetch** module, fetch from each node a copy of the compressed log file back to the control node. The file on the control node should be a name that relates to which node it came from.

Uncompress One of the Files That Has Been Downloaded and Make Sure It Is a Copy of the Files That Are in the /var/log Folder on the Node

Uncompress one of the archived files that has been copied to the control node and test to ensure it is a valid copy of the files as they exist on the node.

HOST :

[root@control1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo '+Vpkb9RZ' | /bin/passwd root —stdin

/bin/echo '+Vpkb9RZ' | /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 10.0.1.73 db1 node1>> /etc/hosts

/bin/echo 10.0.1.55 db2 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

/sbin/useradd ansible

yum install -y ansible

yum install -y sshpass

/sbin/useradd ansible

/bin/echo '+Vpkb9RZ' | /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 '+Vpkb9RZ' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost

sudo -u ansible sshpass -p '+Vpkb9RZ' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@db1

sudo -u ansible sshpass -p '+Vpkb9RZ' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@db2

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

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

echo "db1" >> /etc/ansible/hosts

echo "db2" >> /etc/ansible/hosts

/bin/yum install -y git

/bin/chmod -R ansible:ansible /home/ansible

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

[root@control1 cloud\_user]# sudo cat /etc/os-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"

[root@control1 cloud\_user]# ansible —version

ansible 2.9.27

config file = /etc/ansible/ansible.cfg

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

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

executable location = /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)]

[root@control1 cloud\_user]#

[root@control1 cloud\_user]#

Node :

[root@node1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo '+Vpkb9RZ' | /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 '+Vpkb9RZ' | /bin/passwd ansible —stdin

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

yum install -y mariadb-server

systemctl start mariadb

systemctl enable mariadb

systemctl disable auditd

[root@node1 cloud\_user]# sudo cat /etc/os-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"

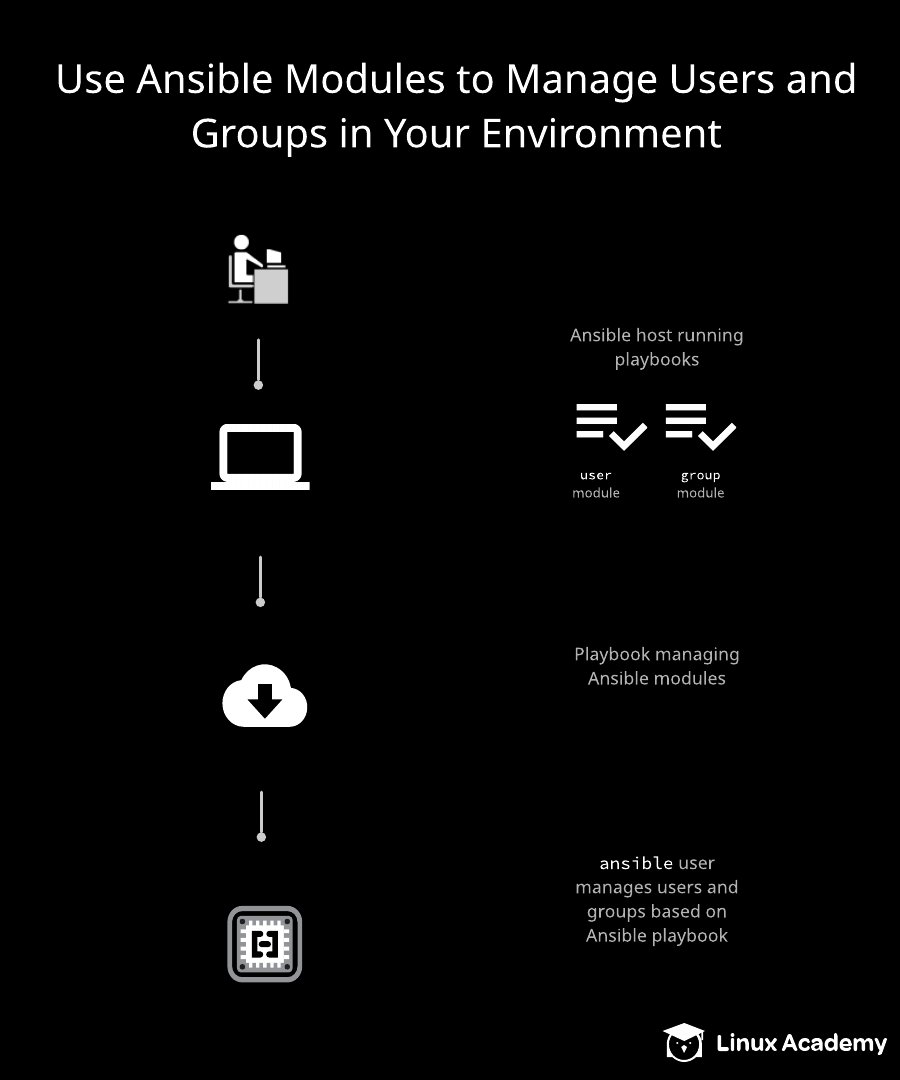
[root@node1 cloud\_user]# ansible —version

bash: ansible: command not found

[root@node1 cloud\_user]#

25

## Using Ansible Modules to Manage Users and Groups in Your Environment



Using Ansible Modules to Manage Users and Groups in Your Environment  
This course is not approved or sponsored by Red Hat.  
  
Introduction  
While many environments have some sort of central authentication, there's still use cases for managing users on servers outside of that. This lab will help drive home managing users and groups with Ansible.  
  
Solution  
Log in to the lab server using the credentials provided:  
  
ssh cloud\_user@<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. To save and quit the file, press Escape followed by :wq. To exit the file without saving, press Escape followed by :q!.  
  
(Note: Ansible is installed as the root user, so please work on all tasks after elevating to the root user.)  
  
Remove the 'old\_backup' User and Create the 'new\_backup' User on All Servers with the Required Settings  
Open users.yml file:  
vim users.yml  
Enter the following information:  
---  
- name: User and Group playbook for backups  
  hosts: all  
  become: yes  
  
  tasks:  
    - group:  
        gid: 12310  
        name: backups  
        state: present  
    - user:  
        name: old\_backup  
        state: absent  
    - user:  
        name: new\_backup  
        uid: 12427  
        shell: /bin/false  
        password\_lock: yes  
        groups: backups  
Save and exit the file:  
:wq  
Run ansible playbook:  
ansible-playbook users.yml  
Create the New Group and Add the 'new\_backup' User  
Edit users.yml file:  
vim users.yml  
Update the file to reflect the following information:  
---  
- name: User and Group playbook for backups  
  hosts: all  
  become: yes  
  
  tasks:  
    - group:  
       gid: 12310  
       name: backups  
       state: present  
    - group:  
       gid: 12311  
       name: new\_backups  
       state: present  
    - user:  
       name: old\_backup  
       state: absent  
    - user:  
       name: new\_backup  
       uid: 12427  
       shell: /bin/false  
       password\_lock: yes  
       groups: backups, new\_backups  
Save and exit the file:  
:wq  
Run ansible playbook:  
ansible-playbook users.yml  
Conclusion  
Congratulations — you've completed this hands-on lab!

## Additional Resources

Your backup team has come to you with a problem. They've switched software vendors and the new software requires the use of a specific username. The scripts they've written to manage backups require a specific user id to perform correctly. They'd like you to use Ansible to remove the old user and create the new one using the same uid, so that they don't have to change all of their scripts. Additionally, the old backup user didn't exist everywhere in the environment, but the new one must - and it needs that user id.

Write a playbook that removes the old\_backup user and creates the new\_backup user with the following:

* The uid is 1242.
* The shell needs to be set to /bin/false.
* The account should be password locked.
* Needs to be a member of the backups group. Some servers may not have this group created already.
* The group ID should be 12310.

After you push these changes out, the backup team comes to you and mentions the new backup software also needs a new group created, named new\_backups. The new\_backup user must be a member of that group, but also keep all of its old groups. The GID for this group must be 12311. Modify and execute your playbook with this new requirement.

*(Note: Ansible is installed as the root user, so please work on all tasks after elevating to the root user.)*

## Learning Objectives

0 of 4 completed

Ensure the 'backups' Group Exists with the Correct GID on All Servers

This section of your playbook should look something like this:

---  
- name: verify group exists  
 hosts: all  
 become: yes  
  
 tasks:  
  - group:   
     gid: 12310  
     name: backups  
     state: present

Remove the 'old\_backup' User on All Servers

You can add this section to the existing playbook:

   - user:  
     name: old\_backup  
     state: absent

Create the 'new\_backup' User on All Servers with the Required Settings

You can add this section to your existing playbook:

   - user:  
     name: new\_backup  
     uid: 12427  
     shell: /bin/false  
     password\_lock: yes  
     groups: backups

Create the New Group and Add the 'new\_backup' User

Your complete playbook should look similar to the following:

---  
- name: verify group exists  
 hosts: all  
 become: yes  
  
 tasks:  
  - group:   
     gid: 12310  
     name: backups  
     state: present  
  - group:  
     gid: 12311  
     name: new\_backups  
     state: present  
  - user:  
     name: old\_backup  
     state: absent  
  - user:  
     name: new\_backup  
     uid: 12427  
     shell: /bin/false  
     password\_lock: yes  
     groups: backups, new\_backups

HOST SERVER :

[root@Server1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/usr/bin/hostnamectl set-hostname Server1

/usr/bin/yum install wget git -y

/bin/echo 'source /root/ansible/hacking/env-setup -q' >> /root/.bashrc

/bin/ssh-keygen -N '' -t rsa -f ~/.ssh/id\_rsa

/bin/cat /root/.ssh/id\_rsa.pub >> /home/ansible/.ssh/authorized\_keys

/bin/echo 'python\_interpreter=auto' >> /etc/ansible/ansible.cfg

/bin/echo 'host\_key\_checking = False' >> /etc/ansible/ansible.cfg

/sbin/alternatives —set python /usr/bin/python3

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/privkey> -O /tmp/pkey

/bin/cat /tmp/pkey > /root/.ssh/id\_rsa

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/authkey> -O /tmp/akey

/bin/cat /tmp/akey > /root/.ssh/id\_rsa.pub

/bin/cat /tmp/akey > /home/ansible/.ssh/id\_rsa.pub

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/etchosts> -O /tmp/hosts

/bin/cat /tmp/hosts > /etc/hosts

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/inventory> -O /tmp/inv

/bin/cat /tmp/inv > /etc/ansible/hosts

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

/usr/bin/echo 'cloud\_user:6GxwYHd|' | /usr/sbin/chpasswd

/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-2259-3a304e476c36a1009956556794fc7ecf —resource PublicInstance —region us-east-1

[root@Server1 cloud\_user]# sudo cat /etc/os-release

NAME="Red Hat Enterprise Linux"

VERSION="8.7 (Ootpa)"

ID="rhel"

ID\_LIKE="fedora"

VERSION\_ID="8.7"

PLATFORM\_ID="platform:el8"

PRETTY\_NAME="Red Hat Enterprise Linux 8.7 (Ootpa)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:redhat:enterprise\_linux:8::baseos"

HOME\_URL="<https://www.redhat.com/>“

DOCUMENTATION\_URL="<https://access.redhat.com/documentation/red_hat_enterprise_linux/8/>“

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

REDHAT\_BUGZILLA\_PRODUCT="Red Hat Enterprise Linux 8"

REDHAT\_BUGZILLA\_PRODUCT\_VERSION=8.7

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

REDHAT\_SUPPORT\_PRODUCT\_VERSION="8.7"

[root@Server1 cloud\_user]# ansible —version

ansible 2.10.0.dev0

config file = /etc/ansible/ansible.cfg

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

ansible python module location = /root/ansible/lib/ansible

executable location = /root/ansible/bin/ansible

python version = 3.6.8 (default, Jan 11 2023, 08:43:50) [GCC 8.5.0 20210514 (Red Hat 8.5.0-16)]

[root@Server1 cloud\_user]#

Node:

[root@DBServer1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/usr/bin/hostnamectl set-hostname DBServer1

/usr/bin/yum install wget -y

/sbin/useradd ansible

/usr/bin/echo 'ansible:6GxwYHd|' | /usr/sbin/chpasswd

/sbin/groupadd -g 12310 backups

/sbin/useradd -G backups -M -s /bin/false -u 12427 old\_backup

/sbin/alternatives —set python /usr/bin/python3

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/etchosts> -O /tmp/hosts

/bin/cat /tmp/hosts > /etc/hosts

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/inventory> -O /tmp/inv

/bin/cat /tmp/inv > /etc/ansible/hosts

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/authkey> -O /tmp/key

/bin/mkdir /home/ansible/.ssh

/usr/bin/chmod 0700 /home/ansible/.ssh/

/bin/cat /tmp/key > /home/ansible/.ssh/authorized\_keys

/usr/bin/chmod 0600 /home/ansible/.ssh/authorized\_keys

/usr/bin/chown ansible.ansible /home/ansible -R

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

/usr/bin/echo 'cloud\_user:6GxwYHd|' | /usr/sbin/chpasswd

/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-2259-3a304e476c36a1009956556794fc7ecf —resource PublicInstance3 —region us-east-1

[root@DBServer1 cloud\_user]# sudo cat /etc/os-release

NAME="Red Hat Enterprise Linux"

VERSION="8.9 (Ootpa)"

ID="rhel"

ID\_LIKE="fedora"

VERSION\_ID="8.9"

PLATFORM\_ID="platform:el8"

PRETTY\_NAME="Red Hat Enterprise Linux 8.9 (Ootpa)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:redhat:enterprise\_linux:8::baseos"

HOME\_URL="<https://www.redhat.com/>“

DOCUMENTATION\_URL="<https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8>“

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

REDHAT\_BUGZILLA\_PRODUCT="Red Hat Enterprise Linux 8"

REDHAT\_BUGZILLA\_PRODUCT\_VERSION=8.9

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

REDHAT\_SUPPORT\_PRODUCT\_VERSION="8.9"

[root@DBServer1 cloud\_user]# ansible —version

bash: ansible: command not found...

26

## Using Ansible Modules to Manage Security in Your Environment



# Using Ansible Modules to Manage Security in Your Environment

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

## Introduction

## Security is paramount in many industries today. Ansible can help make sure your environment is set up in the way your security office requires by pushing out changes and ensuring that current settings are live. In this lab, we'll practice firewall and SELinux configuration.

Your security team is interested in your Ansible proof of concept setup. They'd like to see if it could help them in ensuring that firewalls are set up correctly and that SELinux is running throughout your environment.

Write a playbook to ensure you have relevant ports open on webservers (http & ssh) and dbservers (postgresql & ssh), and enable SELinux on all servers in the environment.

## Solution

## Log in to the lab using the credentials provided:

ssh cloud\_user@<PUBLIC\_IP\_ADDRESS>

Become the root user:

sudo -i

### Ensure Ports 80 and 22 Are Open on Webservers

Create and open the playbook.yml file:

vim playbook.yml

The first section to set up the firewalls on servers should look like this:

**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. To save and quit the file, press **Escape** followed by :wq. To exit the file *without* saving, press **Escape** followed by :q!.

---  
- name: webserver firewall rules  
  hosts: webservers  
  become: yes  
  
  tasks:  
   - name: ssh firewall rules  
     firewalld:  
      permanent: yes  
      state: enabled  
      immediate: yes  
      service: ssh  
  
   - name: http firewall rules  
     firewalld:  
      permanent: yes  
      state: enabled  
      immediate: yes  
      service: http

To save and quit the file, press **Escape** followed by :wq.

Run the Ansible playbook against playbook.yml:

ansible-playbook playbook.yml

You should see the ssh firewall port was previously set, but the Apache firewall is now added.

### Ensure Ports 5432 and 22 Are Open on dbservers

Reopen the playbook:

vim playbook.yml

This playbook section sets up the firewalls on the dbservers and should look similar to the below. Add this below the last section you created:

- name: Set up firewalls on dbservers  
  hosts: dbservers  
  become: yes  
  
  tasks:  
   - name: ssh firewall rules  
     firewalld:  
      permanent: yes  
      state: enabled  
      immediate: yes  
      service: ssh  
  
   - name: postgres rules  
     firewalld:  
      permanent: yes  
      state: enabled  
      immediate: yes  
      service: postgresql

To save and quit the file, press **Escape** followed by :wq.

Re-run the Ansible playbook against playbook.yml:

ansible-playbook playbook.yml

You should see the postgres firewall is now set up.

### Enable SELinux on All Servers

This section of the playbook enables SELinux. Add the following as a new section, below the last one:

- name: SELinux  
  hosts: all  
  become: yes  
  
  tasks:  
   - name: Enable SELinux  
     selinux:  
      state: enforcing  
      policy: targeted

To save and quit the file, press **Escape** followed by :wq.

Re-run the Ansible playbook against playbook.yml:

ansible-playbook playbook.yml

SELinux is now enabled on all 3 servers.

## Conclusion

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

Additional Resources  
Scenario  
Your security team is interested in your Ansible proof of concept setup. They'd like to see if it could help them in ensuring that firewalls are set up correctly and that SELinux is running throughout your environment.  
  
Write a playbook to ensure you have relevant ports open on webservers (http & ssh) and dbservers (postgresql & ssh), and enable SELinux on all servers in the environment.  
  
Log in to the lab using the credentials provided:  
  
ssh cloud\_user@<PUBLIC IP ADDRESS>  
Become the root user:  
  
sudo -i  
Learning Objectives  
0 of 3 completed  
  
Ensure Ports 80 and 22 Are Open on Webservers  
  
Your playbook should look similar to the following:  
  
```  
---  
- name: webserver firewall rules  
  hosts: webservers  
  become: yes  
  
  tasks:  
   - name: ssh firewall rules  
     firewalld:  
      permanent: yes  
      state: enabled  
      immediate: yes  
      service: ssh  
  
   - name: http firewall rules  
     firewalld:  
      permanent: yes  
      state: enabled  
      immediate: yes  
      service: http  
```  
Ensure Ports 5432 and 22 Are Open on dbservers  
  
Your playbook should contain something similar to the following:  
  
```  
- name: Set up firewalls on dbservers  
  hosts: dbservers  
  become: yes  
  
  tasks:  
   - name: ssh firewall rules  
     firewalld:  
      permanent: yes  
      state: enabled  
      immediate: yes  
      service: ssh  
  
   - name: postgres rules  
     firewalld:  
      permanent: yes  
      state: enabled  
      immediate: yes  
      service: postgresql  
```  
Enable SELinux on All Servers  
  
Your playbook should look similar to the following:  
  
```  
- name: SELinux  
  hosts: all  
  become: yes  
  
  tasks:  
   - name: Enable SELinux  
     selinux:  
      state: enforcing  
      policy: targeted  
```

HOST :

[root@Server1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/usr/bin/hostnamectl set-hostname Server1

/usr/bin/yum install wget git -y

/bin/echo 'source /root/ansible/hacking/env-setup -q' >> /root/.bashrc

/bin/ssh-keygen -N '' -t rsa -f ~/.ssh/id\_rsa

/bin/cat /root/.ssh/id\_rsa.pub >> /home/ansible/.ssh/authorized\_keys

/bin/echo 'python\_interpreter=auto' >> /etc/ansible/ansible.cfg

/bin/echo 'host\_key\_checking = False' >> /etc/ansible/ansible.cfg

/sbin/alternatives —set python /usr/bin/python3

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/privkey> -O /tmp/pkey

/bin/cat /tmp/pkey > /root/.ssh/id\_rsa

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/authkey> -O /tmp/akey

/bin/cat /tmp/akey > /root/.ssh/id\_rsa.pub

/bin/cat /tmp/akey > /home/ansible/.ssh/id\_rsa.pub

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/etchosts> -O /tmp/hosts

/bin/cat /tmp/hosts > /etc/hosts

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/inventory> -O /tmp/inv

/bin/cat /tmp/inv > /etc/ansible/hosts

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

/usr/bin/echo 'cloud\_user:g6k\_+8Uc' | /usr/sbin/chpasswd

/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-2268-02ba3c30ebc93c8d6ce5b8c89a615b9d —resource PublicInstance —region us-east-1

[root@Server1 cloud\_user]# sudo cat /etc/os-release

NAME="Red Hat Enterprise Linux"

VERSION="8.7 (Ootpa)"

ID="rhel"

ID\_LIKE="fedora"

VERSION\_ID="8.7"

PLATFORM\_ID="platform:el8"

PRETTY\_NAME="Red Hat Enterprise Linux 8.7 (Ootpa)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:redhat:enterprise\_linux:8::baseos"

HOME\_URL="<https://www.redhat.com/>“

DOCUMENTATION\_URL="<https://access.redhat.com/documentation/red_hat_enterprise_linux/8/>“

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

REDHAT\_BUGZILLA\_PRODUCT="Red Hat Enterprise Linux 8"

REDHAT\_BUGZILLA\_PRODUCT\_VERSION=8.7

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

REDHAT\_SUPPORT\_PRODUCT\_VERSION="8.7"

[root@Server1 cloud\_user]# ansible —version

ansible 2.10.0.dev0

config file = /etc/ansible/ansible.cfg

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

ansible python module location = /root/ansible/lib/ansible

executable location = /root/ansible/bin/ansible

python version = 3.6.8 (default, Jan 11 2023, 08:43:50) [GCC 8.5.0 20210514 (Red Hat 8.5.0-16)]

[root@Server1 cloud\_user]#

[root@Server1 cloud\_user]#

Node :

[root@DBServer1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/usr/bin/hostnamectl set-hostname DBServer1

/usr/bin/yum install wget postgresql firewalld python3-firewall -y

/sbin/useradd ansible

/usr/bin/echo 'ansible:g6k\_+8Uc' | /usr/sbin/chpasswd

/sbin/alternatives —set python /usr/bin/python3

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/etchosts> -O /tmp/hosts

/bin/cat /tmp/hosts > /etc/hosts

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/inventory> -O /tmp/inv

/bin/cat /tmp/inv > /etc/ansible/hosts

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/authkey> -O /tmp/key

/bin/mkdir /home/ansible/.ssh

/usr/bin/chmod 0700 /home/ansible/.ssh/

/bin/cat /tmp/key > /home/ansible/.ssh/authorized\_keys

/usr/bin/chmod 0600 /home/ansible/.ssh/authorized\_keys

/usr/bin/chown ansible.ansible /home/ansible -R

/usr/bin/systemctl start postgresql

/usr/bin/systemctl start firewalld

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

/sbin/setenforce 0

/usr/bin/echo 'cloud\_user:g6k\_+8Uc' | /usr/sbin/chpasswd

/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-2268-02ba3c30ebc93c8d6ce5b8c89a615b9d —resource PublicInstance3 —region us-east-1

[root@DBServer1 cloud\_user]# sudo cat /etc/os-release

NAME="Red Hat Enterprise Linux"

VERSION="8.9 (Ootpa)"

ID="rhel"

ID\_LIKE="fedora"

VERSION\_ID="8.9"

PLATFORM\_ID="platform:el8"

PRETTY\_NAME="Red Hat Enterprise Linux 8.9 (Ootpa)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:redhat:enterprise\_linux:8::baseos"

HOME\_URL="<https://www.redhat.com/>“

DOCUMENTATION\_URL="<https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8>“

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

REDHAT\_BUGZILLA\_PRODUCT="Red Hat Enterprise Linux 8"

REDHAT\_BUGZILLA\_PRODUCT\_VERSION=8.9

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

REDHAT\_SUPPORT\_PRODUCT\_VERSION="8.9"

[root@DBServer1 cloud\_user]# ansible —version

bash: ansible: command not found...

27

# Using Ansible Modules to Manage Scheduled Tasks in Your Environment



# Using Ansible Modules to Manage Scheduled Tasks in Your Environment

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

## Introduction

## Cron jobs are tools that help system administrators automate tasks. The downside is that configuring them across hundreds of systems is a daunting task. Ansible has a module to help here as well. This hands-on lab will help you understand how that module is configured.

## The Scenario

## Recently the security was breached on the webservers in your environment. One of the postmortem findings was that the breached servers were not up to date with current security patches. Because of this, you've been tasked with writing an Ansible Playbook to set up three cron jobs, and then running the playbook on every server in our environment.

Here are the three cron jobs:

1. Running yum update weekly.
2. Execute /usr/bin/status upon reboot to check in with your monitoring system and give a current status of critical services and their package versions.
3. Running /usr/bin/faillock\_report at 7:30AM on Mondays, Wednesdays, and Fridays to report excessive login attempts.

## Solution

**Note:** Please allow extra time for the lab to finish provisioning before connecting to the server. All tasks will need to be executed as the root user.

Log in to the server using the credentials provided:

ssh cloud\_user@<PUBLIC\_IP\_ADDRESS>

## Write an Ansible Playbook for the Three Cron Jobs and Verify Success

1. Become the root user:

sudo -i

1. Create a playbook:

vim cron.yml

1. Enter the following information:

- name: Cron jobs  
  hosts: all  
  become: yes  
  
  tasks:  
  - cron:  
     name: "Weekly YUM Update"  
     special\_time: weekly  
     job: "/usr/bin/yum update -y"  
  
  - cron:  
     name: "Reboot Status"  
     special\_time: reboot  
     job: "/usr/bin/status"  
  - cron:  
     name: "Faillock Report"  
     job: /usr/bin/faillock\_report  
     weekday: "1,3,5"  
     minute: "30"  
     hour: "7"

1. Save and quit the file:

:wq

1. Run the Ansible Playbook:

ansible-playbook cron.yml

1. Log into other server to verify work:

ssh ansible@webserver1

1. Become the root user:

sudo -i

1. Verify the three cron jobs are listed:

crontab -e

## Conclusion

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

## Additional Resources

Recently the security was breached on the webservers in your environment. One of the postmortem findings was that the breached servers were not up to date with current security patches. Because of this, you've been tasked with writing an Ansible Playbook to set up three cron jobs, and then running the playbook on every server in our environment.

Here are the three cron jobs:

1. Running yum update weekly.
2. Execute /usr/bin/status upon reboot to check in with your monitoring system and give a current status of critical services and their package versions.
3. Running /usr/bin/faillock\_report at 7:30AM on Mondays, Wednesdays, and Fridays to report excessive login attempts.

**Note:** Please allow extra time for the lab to finish provisioning before connecting to the server. All tasks will need to be executed as the root user.

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

## Learning Objectives

0 of 3 completed

Write a Playbook to Create a Weekly YUM Update Job in Cron

Your playbook entry for this job should look similar to this:

  - cron:  
     name: "Weekly YUM Update"  
     special\_time: weekly  
     job: "/usr/bin/yum update -y"

Write a Playbook to Execute /usr/bin/status on Reboot

This section of your playbook should look like this:

  - cron:  
     name: "Reboot Status"  
     special\_time: reboot  
     job: "/usr/bin/status"

Write a Playbook to Execute /usr/bin/faillock\_report at the Required Time

This section of the playbook should look like this:

  - cron:  
     name: "Faillock Report"  
     job: /usr/bin/faillock\_report  
     weekday: "1,3,5"  
     minute: "30"  
     hour: "7"  
       
       
     

HOST :

[root@Server1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/usr/bin/hostnamectl set-hostname Server1

/usr/bin/yum install wget git -y

/bin/echo 'source /root/ansible/hacking/env-setup -q' >> /root/.bashrc

/bin/ssh-keygen -N '' -t rsa -f ~/.ssh/id\_rsa

/bin/cat /root/.ssh/id\_rsa.pub >> /home/ansible/.ssh/authorized\_keys

/bin/echo 'python\_interpreter=auto' >> /etc/ansible/ansible.cfg

/bin/echo 'host\_key\_checking = False' >> /etc/ansible/ansible.cfg

/sbin/alternatives —set python /usr/bin/python3

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/privkey> -O /tmp/pkey

/bin/cat /tmp/pkey > /root/.ssh/id\_rsa

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/authkey> -O /tmp/akey

/bin/cat /tmp/akey > /root/.ssh/id\_rsa.pub

/bin/cat /tmp/akey > /home/ansible/.ssh/id\_rsa.pub

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/etchosts> -O /tmp/hosts

/bin/cat /tmp/hosts > /etc/hosts

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/inventory> -O /tmp/inv

/bin/cat /tmp/inv > /etc/ansible/hosts

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

/usr/bin/echo 'cloud\_user:6BiL|l#E' | /usr/sbin/chpasswd

/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-2262-243bec1f4fa64ef798a585e8b44394bb —resource PublicInstance —region us-east-1

[root@Server1 cloud\_user]# sudo cat /etc/os-release

NAME="Red Hat Enterprise Linux"

VERSION="8.7 (Ootpa)"

ID="rhel"

ID\_LIKE="fedora"

VERSION\_ID="8.7"

PLATFORM\_ID="platform:el8"

PRETTY\_NAME="Red Hat Enterprise Linux 8.7 (Ootpa)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:redhat:enterprise\_linux:8::baseos"

HOME\_URL="<https://www.redhat.com/>“

DOCUMENTATION\_URL="<https://access.redhat.com/documentation/red_hat_enterprise_linux/8/>“

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

REDHAT\_BUGZILLA\_PRODUCT="Red Hat Enterprise Linux 8"

REDHAT\_BUGZILLA\_PRODUCT\_VERSION=8.7

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

REDHAT\_SUPPORT\_PRODUCT\_VERSION="8.7"

[root@Server1 cloud\_user]# ansible —version

ansible 2.10.0.dev0

config file = /etc/ansible/ansible.cfg

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

ansible python module location = /root/ansible/lib/ansible

executable location = /root/ansible/bin/ansible

python version = 3.6.8 (default, Jan 11 2023, 08:43:50) [GCC 8.5.0 20210514 (Red Hat 8.5.0-16)]

[root@Server1 cloud\_user]#

SERVER 1 :

[root@WebServer1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/usr/bin/hostnamectl set-hostname WebServer1

/usr/bin/yum install wget -y

/sbin/useradd ansible

/usr/bin/echo 'ansible:6BiL|l#E' | /usr/sbin/chpasswd

/sbin/alternatives —set python /usr/bin/python3

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/etchosts> -O /tmp/hosts

/bin/cat /tmp/hosts > /etc/hosts

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/inventory> -O /tmp/inv

/bin/cat /tmp/inv > /etc/ansible/hosts

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/authkey> -O /tmp/key

/bin/mkdir /home/ansible/.ssh

/usr/bin/chmod 0700 /home/ansible/.ssh/

/bin/cat /tmp/key > /home/ansible/.ssh/authorized\_keys

/usr/bin/chmod 0600 /home/ansible/.ssh/authorized\_keys

/usr/bin/chown ansible.ansible /home/ansible -R

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

/usr/bin/echo 'cloud\_user:6BiL|l#E' | /usr/sbin/chpasswd

/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-2262-243bec1f4fa64ef798a585e8b44394bb —resource PublicInstance2 —region us-east-1

[root@WebServer1 cloud\_user]# sudo cat /etc/os-release

NAME="Red Hat Enterprise Linux"

VERSION="8.9 (Ootpa)"

ID="rhel"

ID\_LIKE="fedora"

VERSION\_ID="8.9"

PLATFORM\_ID="platform:el8"

PRETTY\_NAME="Red Hat Enterprise Linux 8.9 (Ootpa)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:redhat:enterprise\_linux:8::baseos"

HOME\_URL="<https://www.redhat.com/>“

DOCUMENTATION\_URL="<https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8>“

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

REDHAT\_BUGZILLA\_PRODUCT="Red Hat Enterprise Linux 8"

REDHAT\_BUGZILLA\_PRODUCT\_VERSION=8.9

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

REDHAT\_SUPPORT\_PRODUCT\_VERSION="8.9"

[root@WebServer1 cloud\_user]# ansible —version

bash: ansible: command not found...

28

## Working with Tags and Delegation in Ansible Playbooks



Working with Tags and Delegation in Ansible Playbooks  
This course is not approved or sponsored by Red Hat.  
  
Introduction  
The company you work for has purchased some new backup software. The design of the environment has a backup server that will host most of the software and a web server to host the UI. These servers have been deployed and are ready to be configured.  
  
A coworker started a playbook to get everything ready but he was pulled away onto another project. You've been tasked with finishing what he started.  
  
We need to make sure that neither server has unneeded software installed on it. We also want to make sure that in the future when changes are pushed toindex.html, we restart Apache (this is a vendor requirement). Finally, we want to verify that the web server works outside of the firewall, so the Ansible host should test it.  
  
Solution  
Log in to the Ansible host server using the credentials provided:  
  
ssh cloud\_user@<PUBLIC\_IP\_ADDRESS>  
Note: Make sure to elevate to root once logged in. The reason for this is because of how the lab server’s version of Ansible is installed.  
  
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. To save and quit the file, press Escape followed by :wq. To exit the file without saving, press Escape followed by :q!.  
  
Modify the Provided Playbook to Filter Packages Based on Tags  
Elevate to root-level permissions using sudo -i.  
  
Edit the tagsconditionals.yml file:  
  
vim tagsconditionals.yml  
Add appropriate tags to each of the playbook tasks:  
  
- hosts: all  
  become: yes  
  
  tasks:  
    - name: Install Apache on Webservers  
      yum:  
        state: present  
        name: httpd  
      tags: webserver  
  
    - name: Install CSH on Backup servers (required per vendor)  
      yum:  
        state: present  
        name: csh  
      tags: backup  
  
    - name: Configure /var/www/html/index.html  
      copy:  
        dest: "/var/www/html/index.html"  
        content: |  
          This is your new website content.  
          Please replace this with the actual content from the developers before going production.  
        tags: webserver  
  
    - name: Create backup software directory (Per vendor, directory must exist before software is installed)  
      file:  
        dest: "/opt/backup"  
        state: directory  
      tags: backup  
  
    - name: Configure backup streams (Per vendor, file must exist before software is installed)  
      copy:  
        dest: "/opt/backup/streams"  
        content: "192"  
      tags: backup  
Add a Conditional for Web Services to Restart if an Index Is Modified  
Under name: Configure /var/www/html/index.html, register the indexchange variable :  
  
- name: Configure /var/www/html/index.html  
copy:  
  dest: "/var/www/html/index.html"  
  content: |  
    This is your new website content.  
    Please replace this with the actual content from the developers before going production.  
 tags: webserver  
 register: indexchange  
Create a new task to restart Apache when the index has changed:  
  
- name: Restart apache when index changes  
  service:  
    state: restarted  
    name: httpd  
    enabled: yes  
  when: indexchange.changed  
  tags: webserver  
Add a Task That Has the Local Ansible Host Verify the Web Server Is Working  
Create the verification task :  
  
- name: Verify webserver is working correctly  
  uri:  
    url: http://{{ inventory\_hostname }}  
    method: GET  
    timeout: 30  
    status\_code: 200  
    return\_content: yes  
  delegate\_to: localhost  
  register: result  
  tags: webserver  
  
- name: Show the verification results  
  debug:  
    var: result  
  tags: webserver  
To save and exit the file, press Escape, type :wq, and hit Enter.  
  
Run the playbook on backup servers:  
  
ansible-playbook -i inventory tagsconditionals.yml --tags=backup --limit backups  
Run the playbook on the web server :  
  
ansible-playbook -i inventory tagsconditionals.yml --tags=webserver --limit webservers  
Conclusion  
Congratulations — you've completed this hands-on lab!

## Additional Resources

There are three tasks to accomplish for this hands on lab:

1. Modify the provided playbook to filter tasks based on tags.
2. Add a conditional for web services to restart if index.html is modified.
3. Add a task that has the local Ansible host verify the web server is working.

## Learning Objectives

0 of 3 completed

Modify the Provided Playbook to Filter Packages Based on Tags

The provided playbook doesn't have any tags in it, so if it's run as is, unneeded software will be installed on the servers. Add tags so that tasks can be filtered based on which group is being targeted.

Add a Conditional for Web Services to Restart if an Index Is Modified

There is a vendor requirement to restart Apache services if the index.html is modified. Build that into the provided playbook.

Add a Task That Has the Local Ansible Host Verify the Web Server Is Working

There is a firewall in place that we want to ensure isn't blocking traffic on the web server. Verify from the Ansible host that pages load correctly.

HOST :

[root@Server1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/usr/bin/hostnamectl set-hostname Server1

/usr/bin/yum install wget -y

/usr/bin/yum install <https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm> -y

/usr/bin/yum install sshpass -y

/usr/bin/echo 'cloud\_user:d88[qSeF' | /usr/sbin/chpasswd

/bin/echo 'source /root/ansible/hacking/env-setup -q' >> /root/.bashrc

/bin/echo 'python\_interpreter=auto' >> /etc/ansible/ansible.cfg

/bin/echo 'host\_key\_checking = False' >> /etc/ansible/ansible.cfg

/sbin/alternatives —set python /usr/bin/python3

/usr/bin/wget <https://raw.githubusercontent.com/ACloudGuru-Resources/content-ex447-ansible-best-practice/master/ansible/etchosts2> -O /tmp/hosts

/bin/cat /tmp/hosts >> /etc/hosts

/usr/bin/wget <https://raw.githubusercontent.com/ACloudGuru-Resources/content-ex447-ansible-best-practice/master/ansible/inv3> -O /tmp/inv

/bin/cat /tmp/inv > /root/inventory

/usr/bin/wget <https://raw.githubusercontent.com/ACloudGuru-Resources/content-ex447-ansible-best-practice/master/ansible/tagsconditionals.yml> -O /root/tagsconditionals.yml

/bin/ssh-keygen -t rsa -q -f '/root/.ssh/id\_rsa' -N ''

SSHPASS='d88[qSeF' /bin/rsync —rsh='sshpass -e ssh -o StrictHostKeyChecking=no -l cloud\_user' /root/.ssh/id\_rsa.pub 10.0.0.41:/tmp

SSHPASS='d88[qSeF' /bin/sshpass -e ssh -o StrictHostKeyChecking=no cloud\_user@10.0.0.41 'cat /tmp/id\_rsa.pub >> /home/cloud\_user/.ssh/authorized\_keys'

SSHPASS='d88[qSeF' /bin/rsync —rsh='sshpass -e ssh -o StrictHostKeyChecking=no -l cloud\_user' /root/.ssh/id\_rsa.pub 10.0.0.42:/tmp

SSHPASS='d88[qSeF' /bin/sshpass -e ssh -o StrictHostKeyChecking=no cloud\_user@10.0.0.42 'cat /tmp/id\_rsa.pub >> /home/cloud\_user/.ssh/authorized\_keys'

/bin/echo 'cloud\_user ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers.d/90-cloud-init-users

curl -O <https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-py3-latest.tar.gz>

tar xvzf aws-cfn-bootstrap-py3-latest.tar.gz

#easy\_install aws-cfn-bootstrap-latest

cd aws-cfn-bootstrap-\*/

python3 setup.py install

#python setup.py build

#python setup.py install

#mkdir -p /opt/aws/bin

#ln -s /usr/init/redhat/cfn-hup /etc/init.d/cfn-hup

#chmod 775 /usr/init/redhat/cfn-hup

#ln -s /usr/bin/cfn-hup /opt/aws/bin/cfn-hup

#ln -s /usr/bin/cfn-signal /opt/aws/bin/cfn-signal

#ln -s /usr/bin/cfn-init /opt/aws/bin/cfn-init

#ln -s /usr/bin/cfn-get-metadata /opt/aws/bin/cfn-get-metadata

#ln -s /usr/bin/cfn-send-cmd-event /opt/aws/bin/cfn-send-cmd-event

#ln -s /usr/bin/cfn-send-cmd-result /opt/aws/bin/cfn-send-cmd-result

cfn-signal -s true "<https://cloudformation-waitcondition-us-east-1.s3.amazonaws.com/arn%3Aaws%3Acloudformation%3Aus-east-1%3A690784770211%3Astack/cfst-3433-7d6114c409cd7e46f5999c4f30d3bc74/df681470-1046-11ef-89a8-0e87f4101add/df6a3750-1046-11ef-89a8-0e87f4101add/PublicInstanceWaitHandle?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20240512T100328Z&X-Amz-SignedHeaders=host&X-Amz-Expires=86399&X-Amz-Credential=AKIA6L7Q4OWT3JRXU3BZ%2F20240512%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Signature=e7110e12e097713a56a29b7c240c3013a25ec1d6217123d55fd2549301cb14be>“

[root@Server1 cloud\_user]# sudo cat /etc/os-release

NAME="Red Hat Enterprise Linux"

VERSION="8.7 (Ootpa)"

ID="rhel"

ID\_LIKE="fedora"

VERSION\_ID="8.7"

PLATFORM\_ID="platform:el8"

PRETTY\_NAME="Red Hat Enterprise Linux 8.7 (Ootpa)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:redhat:enterprise\_linux:8::baseos"

HOME\_URL="<https://www.redhat.com/>“

DOCUMENTATION\_URL="<https://access.redhat.com/documentation/red_hat_enterprise_linux/8/>“

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

REDHAT\_BUGZILLA\_PRODUCT="Red Hat Enterprise Linux 8"

REDHAT\_BUGZILLA\_PRODUCT\_VERSION=8.7

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

REDHAT\_SUPPORT\_PRODUCT\_VERSION="8.7"

[root@Server1 cloud\_user]# ansible —version

ansible 2.10.0.dev0

config file = /etc/ansible/ansible.cfg

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

ansible python module location = /root/ansible/lib/ansible

executable location = /root/ansible/bin/ansible

python version = 3.6.8 (default, Jan 11 2023, 08:43:50) [GCC 8.5.0 20210514 (Red Hat 8.5.0-16)]

[root@Server1 cloud\_user]#

SERVER :

[cloud\_user@WebServer1 ~]$ curl 169.254.169.254/latest/user-data

#!/bin/bash

/usr/bin/hostnamectl set-hostname WebServer1

/sbin/alternatives —set python /usr/bin/python3/usr/bin/echo 'cloud\_user:d88[qSeF' | /usr/sbin/chpasswd/sbin/setenforce 0/bin/echo 'cloud\_user ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers.d/90-cloud-init-users

[cloud\_user@WebServer1 ~]$ sudo cat /etc/os-release

NAME="Red Hat Enterprise Linux"

VERSION="8.9 (Ootpa)"

ID="rhel"

ID\_LIKE="fedora"

VERSION\_ID="8.9"

PLATFORM\_ID="platform:el8"

PRETTY\_NAME="Red Hat Enterprise Linux 8.9 (Ootpa)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:redhat:enterprise\_linux:8::baseos"

HOME\_URL="<https://www.redhat.com/>“

DOCUMENTATION\_URL="<https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8>“

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

REDHAT\_BUGZILLA\_PRODUCT="Red Hat Enterprise Linux 8"

REDHAT\_BUGZILLA\_PRODUCT\_VERSION=8.9

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

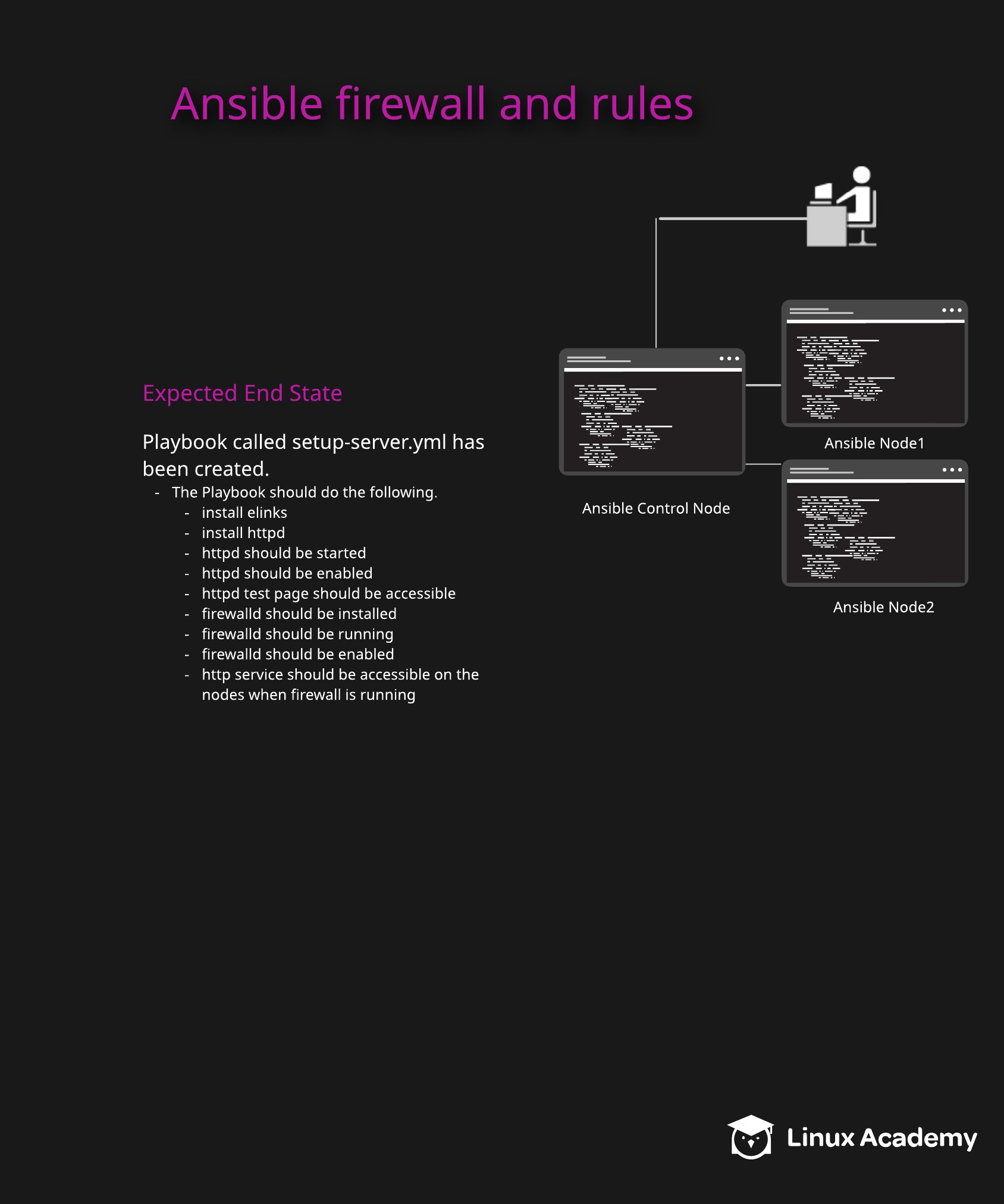
REDHAT\_SUPPORT\_PRODUCT\_VERSION="8.9"

[cloud\_user@WebServer1 ~]$ ansible —version

bash: ansible: command not found...

29

## Using Ansible Firewall Rules



Using Ansible Firewall Rules  
This course is not approved or sponsored by Red Hat.  
  
Introduction  
In this hands-on lab, we will use an Ansible playbook to install the Apache Web Server, and then install a firewall. Afterwards, we will add a rule to the firewall, that will allow Apache to be accessed externally.  
  
Note: Ansible has been set up and configured for use.  
  
Solution  
To access the lab environment, log in to the control node server as cloud\_user, using the IP address and login information provided in the Credentials section of the hands-on lab page.  
  
Sign onto the Ansible Control Node Server as cloud\_user and Change to the ansible User  
Sign in to the server called Ansible Control Node using the cloud\_user, and then change to the ansible user via the su - ansible command.  
Test that Ansible is working via an ad hoc command. We can use the following:  
ansible all -m ping  
Ensure That firewalld Is Not Installed on the Ansible nodes, node1 and node2  
Check that firewalld is not installed on the nodes:  
ssh node1  
sudo systemctl status firewalld  
Use exit to move onto the next node check:  
ssh node2  
sudo systemctl status firewalld  
Note: For both nodes we should encounter a "service not be found message."  
  
Use exit to close node2, and then clear the screen.  
Using an Ansible Playbook, Install the Apache Web Server, httpd and elinks onto the Ansible Nodes  
Using a playbook, we should install httpd and elinks onto the Ansible nodes.  
  
Create the playbook:  
vim setup-server.yml  
The playbook file could appear like so for the installations, plus enabling and starting the Apache Web Server:  
---  
- hosts: all  
  user: ansible  
  become: yes  
  gather\_facts: no  
  tasks:  
    - name: install elinks  
      action: yum name=elinks state=installed  
    - name: install httpd  
      action: yum name=httpd state=installed  
    - name: Enable Apache on system reboot  
      service: name=httpd enabled=yes  
    - name: Start service httpd, if not started  
      service:  
        name: httpd  
        state: started  
Run the playbook:  
ansible-playbook setup-server.yml  
Verify that the Apache HTTP Server on Red Had Enterprise Test Linux page can be seen:  
elinks https://localhost  
Ensure the Web Server is Working and Test That Apache on the Nodes Can Be Accessed  
Using the elinks command. We will verify that we can see the Apache HTTP Server test page on the 2 Ansible nodes with the following commands:  
elinks http://node1  
elinks http://node2  
Using the Playbook, Install firewalld on the ansible Nodes  
We will ensure that firewalld is running and verify that the Apache test pages are not available.  
  
Make changes to the playbook for installing firewalld:  
vim setup-server.yml  
We need to add the firewall section at the bottom. It should look like this when we're done:  
  
---  
- hosts: all  
  user: ansible  
  become: yes  
  gather\_facts: no  
  tasks:  
    - name: install elinks  
      action: yum name=elinks state=installed  
    - name: install httpd  
      action: yum name=httpd state=installed  
    - name: Enable Apache on system reboot  
      service: name=httpd enabled=yes  
    - name: Start service httpd, if not started  
      service:  
        name: httpd  
        state: started  
    - name: install firewalld  
      action: yum name=firewalld state=installed  
    - name: Enable firewalld on system reboot  
      service: name=firewalld enabled=yes  
    - name: Start service firewalld, if not started  
      service:  
        name: firewalld  
        state: started  
Run the playbook again to apply the changes:  
ansible-playbook setup-server.yml  
Verify that we are unable to access the test pages on node1 and node2 with the following commands:  
elinks http://node1  
elinks http://node2  
Note: We will be unable to access the test pages because the firewall blocks them.  
  
Change the Ansible Playbook to Add the Firewall Rule to Allow Port 80  
Make changes to allow port 80 in the playbook and restart the firewalld service. It should look like this when we're done:  
---  
- hosts: all  
  user: ansible  
  become: yes  
  gather\_facts: no  
  tasks:  
    - name: install elinks  
      action: yum name=elinks state=installed  
    - name: install httpd  
      action: yum name=httpd state=installed  
    - name: Enable Apache on system reboot  
      service: name=httpd enabled=yes  
    - name: Start service httpd, if not started  
      service:  
        name: httpd  
        state: started  
    - name: install firewalld  
      action: yum name=firewalld state=installed  
    - name: Enable firewalld on system reboot  
      service: name=firewalld enabled=yes  
    - firewalld:  
        service: http  
        permanent: yes  
        state: enabled  
    - name: Restart service firewalld  
      service:  
        name: firewalld  
        state: restarted  
Test That You Can access the Test Pages on the Web Servers on the 2 Ansible Nodes.  
Run the playbook:  
ansible-playbook setup-server.yml  
Using the elinks command again:  
 elinks http://node1  
 elinks http://node2  
This time we should be able to access the test web pages on the two Ansible nodes.  
  
Conclusion  
Congratulations — you've completed this hands-on lab.

## Additional Resources

In this Hands-On Lab, it has been noted that the security on the servers needs to be updated. Because of this, you have been tasked with adding a firewall to the organization's servers. Since the servers are Centos 7, you have been asked to use firewalld.

You have also been asked to check to see what happens when a firewall is in place but a specific firewall rule is not.

Our task is to create a playbook called setup-server.yml that installs elinks and Apache (HTTPD). Use this playbook to start the httpd service and enable it to work after a server reboot. Use the elinks or another method to verify that Apache works, and that we can access the default web page.

Install firewalld with the same playbook, and start it. Test the access to Apache on the nodes again. It is not expected to work, as firewalld should be blocking port 80.

Do what is required within the playbook so that traffic is allowed through on port 80.

To summarize, using an Ansible playbook we must do the following:

1. Ensure that firewalld is *not* installed on the nodes.
2. Install Apache Web Server, httpd, and elinks on the nodes.
3. Perform a test to see if the Apache Web Server is working and that the default web page is accessible.
4. Install firewalld on the nodes.
5. Perform a test to verify that the Apache Web Server is no longer accessible.
6. Use an Ansible recipe to add a firewall rule to the nodes that will allow traffic on port 80 from external connections.
7. Perform a test to verify that the Apache Web Server is accessible on the nodes.

Here are some important things to take into consideration while completing each task:

* For convenience, Ansible is already on the control node. If we connect to the server by clicking on the Public IP address in a browser, we need to make sure we change to the ansible user with the su - ansible command.
* The user ansible is present on all servers, with appropriate shared keys for access to managed servers from the control node. Make sure to use the ansible user to complete the commands.
* The ansible user has the same password as cloud\_user.
* The default Ansible inventory has been configured for with the appropriate hosts and groups.
* /etc/hosts entries are present on control1 for the managed servers.

For detailed instructions on how to complete these tasks, expand each learning objective below, or click the *Guide* tab above the video player.

## Learning Objectives

0 of 7 completed

Sign on to the Ansible Control Node Server as cloud\_user and change to the `ansible` user. Test to ensure that Ansible has been set up for you.

Sign in to the server called *Ansible Control Node* using the cloud\_user, and then change to the ansible user via the su - ansible command.

Test that Ansible is working via an ad-hoc command. We can use the following:

ansible all -m ping

Ensure that `firewalld` is not installed on the Ansible nodes, `node1` and node2

1. Check that firewalld is not installed on the nodes:

ssh node1  
sudo systemctl status firewalld

1. Use exit to move onto the next node check:

ssh node2  
sudo systemctl status firewalld

Using an Ansible playbook, install the Apache Web Server, `httpd` and also `elinks` onto the Ansible nodes and start the Apache Web Server.

1. Create the playbook:

vim setup-server.yml

1. The playbook file could appear like so for the installations, plus enabling and starting the Apache Web Server:

---  
- hosts: all  
  user: ansible  
  become: yes  
  gather\_facts: no  
  tasks:  
    - name: install elinks  
      action: yum name=elinks state=installed  
    - name: install httpd  
      action: yum name=httpd state=installed  
    - name: Enable Apache on system reboot  
      service: name=httpd enabled=yes  
    - name: Start service httpd, if not started  
      service:  
        name: httpd  
        state: started

Ensure the webserver is working and test that apache on the nodes can be accessed.

1. Using the elinks command. We will verify that we can see the *Apache HTTP Server* test page on the 2 Ansible nodes with the following commands:

elinks http://node1  
elinks http://node2

Using the playbook, install `firewalld` on the Ansible nodes. Ensure the `firewalld` is running. Test the Apache test pages are not avaiable.

1. Make changes to the playbook for installing firewalld:

---  
- hosts: all  
  user: ansible  
  become: yes  
  gather\_facts: no  
  tasks:  
    - name: install elinks  
      action: yum name=elinks state=installed  
    - name: install httpd  
      action: yum name=httpd state=installed  
    - name: Enable Apache on system reboot  
      service: name=httpd enabled=yes  
    - name: Start service httpd, if not started  
      service:  
        name: httpd  
        state: started  
    - name: install firewalld  
      action: yum name=firewalld state=installed  
    - name: Enable firewalld on system reboot  
      service: name=firewalld enabled=yes  
    - name: Start service firewalld, if not started  
      service:  
        name: firewalld  
        state: started

1. Save and exit.
2. Run the playbook with the changes applied:

vim setup-server.yml  
ansible-playbook setup-server.yml

1. Verify that we are unable to access the test pages on node1 and node2 with the following commands:

elinks http://node1  
elinks http://node2

**Note:** We will be unable to access the test pages because the firewall will be blocking them.

Change the Ansible playbook to add the firewall rule to allow port 80.

1. Make changes to the playbook for installing firewalld:

---  
- hosts: all  
  user: ansible  
  become: yes  
  gather\_facts: no  
  tasks:  
    - name: install elinks  
      action: yum name=elinks state=installed  
    - name: install httpd  
      action: yum name=httpd state=installed  
    - name: Enable Apache on system reboot  
      service: name=httpd enabled=yes  
    - name: Start service httpd, if not started  
      service:  
        name: httpd  
        state: started  
    - name: install firewalld  
      action: yum name=firewalld state=installed  
    - name: Enable firewalld on system reboot  
      service: name=firewalld enabled=yes  
    - firewalld:  
        service: http  
          permanent: yes  
          state: enabled  
      - name: Restart service firewalld  
        service:  
        name: firewalld  
        state: restarted

1. Save and exit.
2. Run the playbook with the changes applied:

vim setup-server.yml  
ansible-playbook setup-server.yml

1. Verify that we are unable to access the test pages on node1 and node2 with the following commands:

elinks http://node1  
elinks http://node2

**Note:** We will be unable to access the test pages because the firewall will be blocking them.

Test that you can access the test pages on the web servers on the 2 Ansible nodes.

1. Run the playbook:

vim setup-server.yml  
ansible-playbook setup-server.yml

1. Using the elinks command again:

 elinks http://node1  
 elinks http://node2

HOST :

[root@control1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo '4Mc4|dM5' | /bin/passwd root —stdin

/bin/echo '4Mc4|dM5' | /bin/passwd cloud\_user —stdin/bin/echo "StrictHostKeyChecking no" >> /etc/ssh/ssh\_configsed -i s/requiretty/\!requiretty/ /etc/sudoershostnamectl set-hostname control1

/bin/echo 10.0.1.53 db1 node1>> /etc/hosts

/bin/echo 10.0.1.87 db2 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 '4Mc4|dM5' | /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 '4Mc4|dM5' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost

sudo -u ansible sshpass -p '4Mc4|dM5' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@db1

sudo -u ansible sshpass -p '4Mc4|dM5' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@db2

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

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

echo "db1" >> /etc/ansible/hosts

echo "db2" >> /etc/ansible/hosts

/bin/yum install -y git

/bin/chmod -R ansible:ansible /home/ansible

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

[root@control1 cloud\_user]# sudo cat /etc/os-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"

[root@control1 cloud\_user]# ansible —version

ansible 2.9.27

config file = /etc/ansible/ansible.cfg

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

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

executable location = /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)]

[root@control1 cloud\_user]#

SERVER 1 :

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

#!/bin/bash

/bin/echo '4Mc4|dM5' | /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 '4Mc4|dM5' | /bin/passwd ansible —stdin

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

yum install -y mariadb-server

systemctl start mariadb

systemctl enable mariadb

systemctl disable auditd

[cloud\_user@node1 ~]$ sudo cat /etc/os-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"

[cloud\_user@node1 ~]$ ansible —version

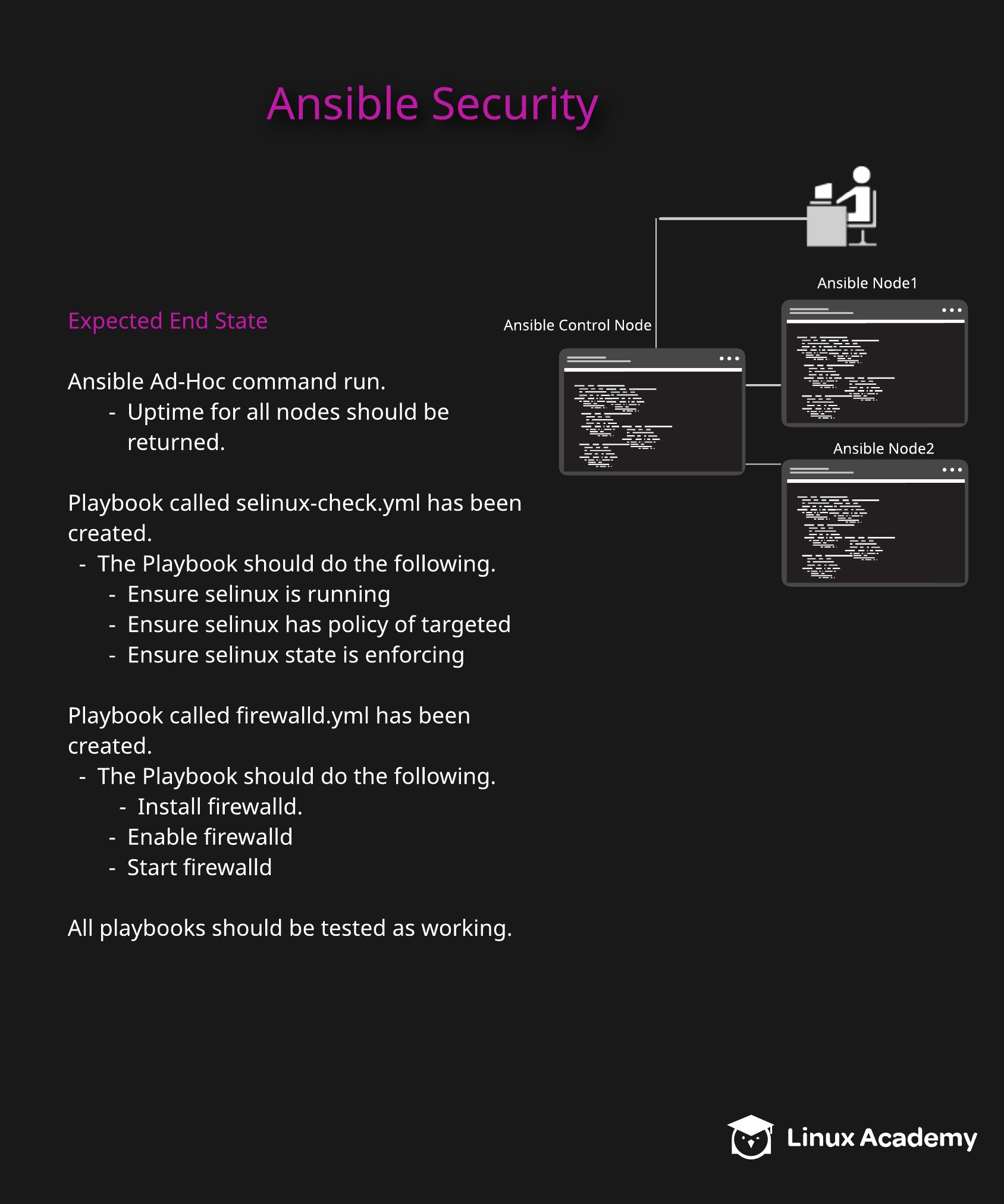
-bash: ansible: command not found

[cloud\_user@node1 ~]$

[cloud\_user@node1 ~]$

30

## Increasing Ansible Security



# Increasing Ansible Security

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

## Introduction

## In this hands-on lab, we will use a playbook and assorted modules to increase the security of your Ansible nodes.

**Note:** Ansible has been set up and configured for your use.

## Solution

## Log in to the Ansible Control Node server using the credentials provided:

ssh cloud\_user@<ANSIBLE\_CONTROL\_NODE\_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.*

### Sign onto the Ansible Control Node Server as cloud\_user and Change to the ansible User

Become the ansible user:

su - ansible

Test whether Ansible is working:

ansible all -m ping

You should see success messages.

**Note:** If you get a "service is not working" message, this may be because it can take several minutes after the start of the lab for the test to work as expected. Wait a few minutes before trying the command again.

### Use an Ad Hoc Ansible Command to Check the Uptime on All the Nodes

Show the uptime on all the servers:

ansible all -a /usr/bin/uptime

### Create a Playbook Called selinux.yml and Ensure SELinux Is Enabled on All Nodes

Create a playbook called selinux.yml:

vim selinux-check.yml

Check the playbook, ensuring all the nodes have SELinux installed and it's set to enabled on all nodes:

---  
- hosts: all  
  user: ansible  
  become: yes  
  gather\_facts: no  
  tasks:  
  
  - name: Enable SELinux  
    selinux:  
      policy: targeted  
      state: enforcing

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

### Test the Playbook selinux.yml to Ensure It Works

Test the playbook selinux.yml against all the nodes, and ensure there are no errors:

ansible-playbook selinux-check.yml

### Create a Playbook Called firewalld.yml and Verify the Installation of firewalld

Create the firewalld.yml playbook that will install, enable, and start firewalld:

vim firewalld.yml

Enter the following information into the playbook:

---  
- hosts: all  
  user: ansible  
  become: yes  
  gather\_facts: no  
  tasks:  
    - name: install firewalld  
      action: yum name=firewalld state=installed  
    - name: Enable firewalld on system reboot  
      service: name=firewalld enabled=yes  
    - name: Start service firewalld, if not started  
      service:  
        name: firewalld  
        state: started

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

Run the playbook to install and enable firewalld:

ansible-playbook firewalld.yml

### Test the firewalld.yml Playbook and Verify Task Completion

Test the playbook and ensure firewalld is installed, enabled, and started on each of the nodes:

sudo systemctl status firewalld

## Conclusion

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

## Additional Resources

You recently joined an organization, and find out that security on the servers could be better than it currently is. Your manager agrees and has tasked you with performing some security-related tasks on all the nodes under control of Ansible.

In this hands-on lab, we will make use of the various modules to help secure your nodes in a reproducible manner. This hands-on lab is not meant to be an exhaustive example of security; instead, it showcases a couple of items to enhance your nodes security.

To summarize, perform the following security-related tasks:

1. Find the uptime of all the Ansible nodes using an Ansible ad-hoc command.
2. Create a playbook called selinux-check.yml and use it to check all the nodes for the status of SELinux. SELinux should get set to an enforcing so you should ensure all nodes get set correctly.
3. Using a playbook called firewalld.yml, ensure that firewalld gets installed on all the nodes.

Here are some important things to consider when completing each task:

* For your convenience, Ansible is already on the control node. If you connect to the server by clicking on the Public IP address in your browser, make sure to change to the ansible user with the su - ansible command.
* The user ansible is present on all servers with appropriate shared keys for access to managed servers from the control node. Make sure to use this user to complete the commands.
* The ansible user has the same password as cloud\_user.
* The default Ansible inventory has been configured for you with the appropriate hosts and groups.
* /etc/hosts entries are present on control1 for the managed servers.

For detailed instructions on how to complete these tasks, expand each learning objective below, or click the *Guide* tab above the video player.

## Learning Objectives

0 of 6 completed

Log in to the Ansible Control Node Server and Change to the `ansible` User

Sign in to the Ansible Control Node server using the cloud\_user credentials, and change to the ansible user.

Test that Ansible is working:

ansible all -m ping

**Note:** It may take several minutes after the start of the lab, for the test to work as expected.

Use an Ad Hoc Ansible Command to Check the Uptime on All the Nodes

Show the uptime on all the servers:

ansible all -a /usr/bin/uptime

Create a Playbook Called `selinux-check.yml`and Ensure SELinux Is Enabled on All Nodes

Use the selinux module and create a playbook called selinux-check.yml:

vim selinux-check.yml

Check the playbook, ensuring all the nodes have SELinux installed and that it's set to enabled on all nodes.

**Note**: More information about the selinux module can be found here: <https://docs.ansible.com/ansible/latest/modules/selinux_module.html>

Test the Playbook `selinux-check.yml` to Ensure It Works

Test the playbook selinux-check.yml against all the nodes and ensure there are no errors:

ansible-playbook selinux-check.yml

Create a Playbook Called `firewalld.yml` and Verify the Installation of firewalld

1. Use the firewalld module (and any other modules as required) in a playbook called firewalld.yml to ensure firewalld is installed, enabled, and started on all the nodes.

**Note**: More information about the firewalld module can be found here: <https://docs.ansible.com/ansible/latest/modules/firewalld_module.html?highlight=firewalld>

Test the `firewalld.yml` Playbook and Verify Task Completion

Test the playbook and ensure that firewalld is installed, enabled, and started on each of the nodes:

sudo systemctl status firewalld

HOST :

[root@control1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo 'BK4\_7r2m' | /bin/passwd root —stdin

/bin/echo 'BK4\_7r2m' | /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 10.0.1.218 db1 node1>> /etc/hosts

/bin/echo 10.0.1.111 db2 node2>> /etc/hosts

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

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

yum install -y ansible

yum install -y sshpass

/sbin/useradd ansible

/bin/echo 'BK4\_7r2m' | /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 'BK4\_7r2m' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost

sudo -u ansible sshpass -p 'BK4\_7r2m' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@db1

sudo -u ansible sshpass -p 'BK4\_7r2m' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@db2

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

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

echo "db1" >> /etc/ansible/hosts

echo "db2" >> /etc/ansible/hosts

/bin/yum install -y git

/bin/chmod -R ansible:ansible /home/ansible

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

[root@control1 cloud\_user]# sudo cat /etc/os-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"

[root@control1 cloud\_user]# ansible —version

ansible 2.9.27

config file = /etc/ansible/ansible.cfg

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

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

executable location = /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)]

[root@control1 cloud\_user]#

NODE 1 :

[root@node1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo 'BK4\_7r2m' | /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

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

yum install -y ansible

/sbin/useradd ansible

/bin/echo 'BK4\_7r2m' | /bin/passwd ansible —stdin

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

yum install -y mariadb-server

systemctl start mariadb

systemctl enable mariadb

systemctl disable auditd

[root@node1 cloud\_user]# sudo cat /etc/os-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"

[root@node1 cloud\_user]# ansible —version

ansible 2.9.27

config file = /etc/ansible/ansible.cfg

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

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

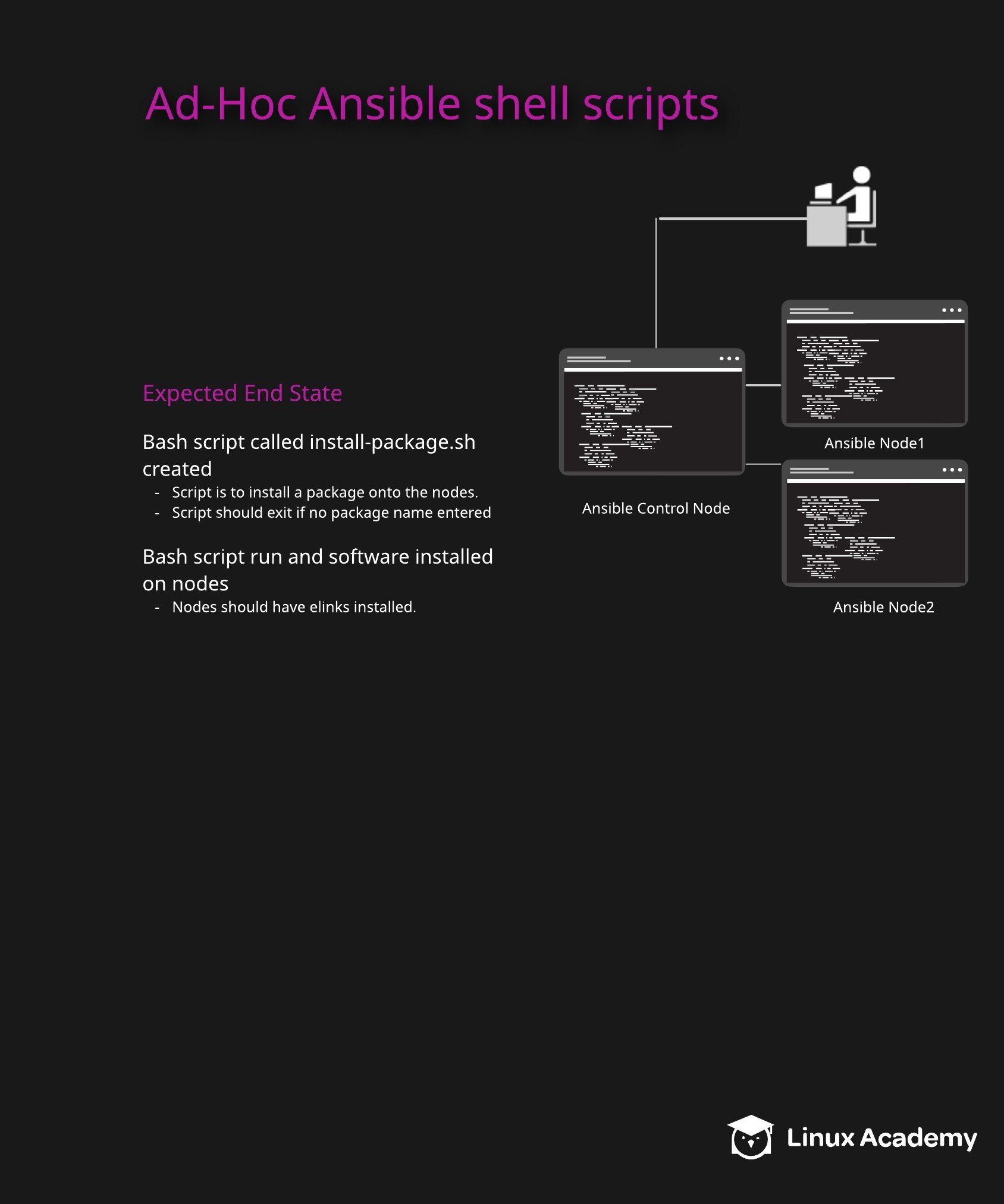
executable location = /bin/ansible

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

[root@node1 cloud\_user]#

31

## Ad-Hoc Ansible Shell Scripts



# Ad-Hoc Ansible Shell Scripts

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

## Introduction

## One of the keys to success with Ansible is being able to run ad-hoc commands. In this particular exercise we will make use of scripting and ad-hoc commands to perform tasks.

The value of ad-hoc commands in scripts is underscored by the fact that it is an objective of the *Red Hat Certified Ansible Specialist* exam.

In this lab we will create a Bash script and use it to perform ad-hoc Ansible tasks.

Note: Ansible has been setup and configured for use.

## Instructions

## We have been tasked with creating a Bash script that will make use of an ad-hoc Ansible command to install software on our Ansible servers.

The purpose of the script is so that a user who has no experience with Ansible can just run it and install software as needed.

To summarize, we must do the following:

1. Create a Bash script to use an ad-hoc Ansible command to install software.
2. Make sure the script accepts a command line variable with the software to install.
3. If no software to install is included on the command line, the script must exit and alert the user.

Important notes:

* Ansible is already on the control node. If we connect to the server by clicking on the Public IP address in a web browser, we need to make sure we change to the ansible user, with the su - ansible command.
* The user ansible is present on all servers with appropriate shared keys for access to managed servers from the control node. We need to make sure we use this user to complete the commands.
* The ansible user has the same password as cloud\_user.
* The default Ansible inventory has already been configured with the appropriate hosts and groups.
* /etc/hosts entries are present on the control1 host for the managed servers.

## Get Logged In

## Login credentials are all on the lab overview page. Once we're logged into the control1 server, become the ansible user (su - ansible) and we can get going.

## Sign onto the Ansible Control Node Server as cloud\_user and Change to the ansible User, Then Test Ansible Has Been Set Up

## Let's look at our hosts file:

cat /etc/hosts

We should see node1 and node2 in there, each with a private IP address. And we should be able to log in (with SSH) to either of them without a password.

Back in control1 as ansible, test that Ansible is working with an ad-hoc command. One possible command to use is:

ansible all -m ping

## Create a Bash Script Called install-package.sh That Accepts a Parameter and Uses an Ad-Hoc Ansible Command to Install Software

## Using Bash, create a script that accepts a variable on the command line and uses an ad-hoc Ansible command to install software on the node1 and node2 servers.

The script should exit if no program to install has been typed in the command line.

Here is the script:

#!/bin/bash  
  
# Command line variable is $1  
  
if [ -n "$1" ]; then  
  echo "Package to install is $1"  
else  
  echo "Package to install not supplied."  
  exit  
fi  
  
ansible all -b -m yum -a "name=$1 state=present"

## Test the Script to Ensure It Will Exit If a Command Line Argument Has Not Been Entered

## If no command line argument has been entered, the script should exit. Let's make it executable (chmod +x install-package.sh), the run it (./install-package.sh). Since we didn't supply any package names, it should just dump us back out to the Bash prompt.

## Use the Script to Install ELinks on the node1 and node2 Servers

## Now let's run it again, with a package name:

./install-package.sh elinks

We should see ELinks get installed on both nodes. Let's check to be sure though...

## Log into node1 and node2 and Ensure the ELinks Package Has Been Installed

## As the ansible user, log into the node1 and node2 servers, and ensure that ELinks has been installed. Test on each node by running elinks http://google.com. It should work on both of them.

## Conclusion

## Well, we've reached the end of the lab. We have a working script now that installs software on nodes, using ad-hoc Ansible commands on a control node. Congratulations!

## Additional Resources

We have been tasked with creating a Bash script that will make use of an ad-hoc Ansible command to install software on our Ansible servers.

The purpose of the script is so that a user who has no experience with Ansible can just run it and install software as needed.

To summarize, we must do the following:

1. Create a Bash script to use an ad-hoc Ansible command to install software.
2. Make sure the script accepts a command line variable with the software to install.
3. If no software to install is included on the command line, the script must exit and alert the user.

Important notes:

* Ansible is already on the control node. If we connect to the server by clicking on the Public IP address in a web browser, we need to make sure we change to the ansible user, with the su - ansible command.
* The user ansible is present on all servers with appropriate shared keys for access to managed servers from the control node. We need to make sure we use this user to complete the commands.
* The ansible user has the same password as cloud\_user.
* The default Ansible inventory has already been configured with the appropriate hosts and groups.
* /etc/hosts entries are present on the control1 host for the managed servers.

## Learning Objectives

0 of 5 completed

Sign onto the Ansible Control Node Server as cloud\_user and Change to the ansible User, Then Test Ansible Has Been Set Up

Sign into the server called Ansible Control Node as cloud\_user and change to the ansible user via the su - ansible command.

Test that Ansible is working, with an ad-hoc command. One possible command to use is:

ansible all -m ping

Create a Bash Script Called install-package.sh That Accepts a Parameter and Uses an Ad-Hoc Ansible Command to Install Software

Using Bash, create a script that accepts a variable on the command line and uses an ad-hoc Ansible command to install software on the node1 and node2 servers.

The script should exit if no program to install has been typed in the command line.

Test the Script to Ensure It Will Exit If a Command Line Argument Has Not Been Entered

If no command line argument has been entered, the script should exit.

Make Sure the Script is Executable and Use it to Install Elinks on the node1 and node2 Servers

Ensure the script is executable, and run it to install ELinks on the nodes.

Log into node1 and node2 and Ensure the ELinks Package Has Been Installed

As the ansible user, log into the node1 and node2 servers, and ensure that ELinks has been installed.

SERVER :

[root@control1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo '6x0TW!(v' | /bin/passwd root —stdin

/bin/echo '6x0TW!(v' | /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 10.0.1.34 db1 node1>> /etc/hosts

/bin/echo 10.0.1.162 db2 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 '6x0TW!(v' | /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 '6x0TW!(v' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost

sudo -u ansible sshpass -p '6x0TW!(v' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@db1

sudo -u ansible sshpass -p '6x0TW!(v' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@db2

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

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

echo "db1" >> /etc/ansible/hosts

echo "db2" >> /etc/ansible/hosts

/bin/yum install -y git

/bin/chmod -R ansible:ansible /home/ansible

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

[root@control1 cloud\_user]# sudo cat /etc/os-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"

[root@control1 cloud\_user]# ansible —version

ansible 2.9.27

config file = /etc/ansible/ansible.cfg

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

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

executable location = /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)]

[root@control1 cloud\_user]#

NODE 1 :

[root@node1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo '6x0TW!(v' | /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 '6x0TW!(v' | /bin/passwd ansible —stdin

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

yum install -y mariadb-server

systemctl start mariadb

systemctl enable mariadb

systemctl disable auditd

[root@node1 cloud\_user]# sudo cat /etc/os-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"

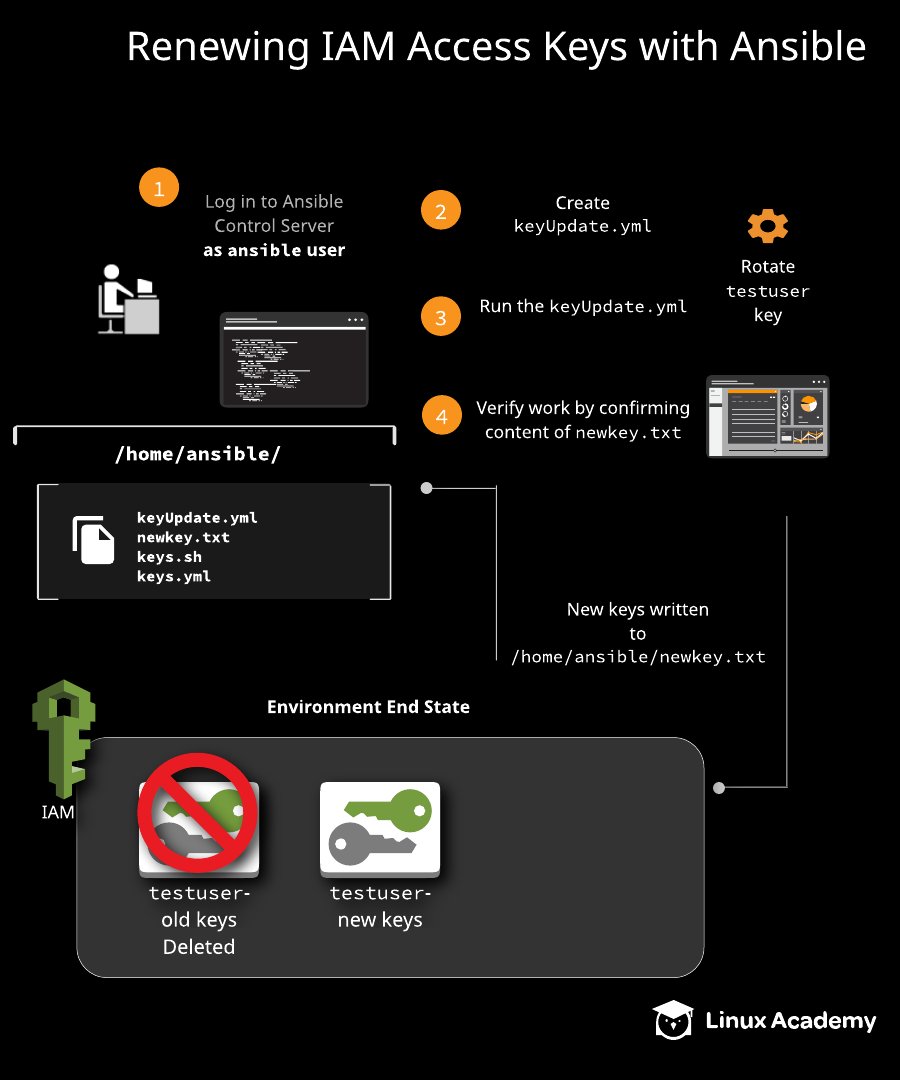
[root@node1 cloud\_user]# ansible —version

bash: ansible: command not found

[root@node1 cloud\_user]#

32

## Renewing IAM Access Keys with Ansible



Renewing IAM Access Keys with Ansible  
Introduction  
Rotating AWS access keys is an important part of an overall security strategy. Ansible can help us manage this process. In this exercise, we will see how to work with AWS IAM user keys using Ansible.  
  
Instructions  
Our boss heard about another company's AWS environment being compromised after IAM user credentials were stolen. In order to protect our company's AWS environment, our boss has tasked us with creating automation that updates IAM keys for a provided user. The company is heavily invested in Ansible, so we have been asked to create the automation using Ansible.  
  
We have been provided an Ansible Control node and a sandbox AWS environment.  
  
Start work on the Ansible Control node:  
  
Create the playbook /home/ansible/keyUpdate.yml to perform the following tasks:  
Look up the current IAM access key for the testuser IAM user.  
Remove the old access id key and secret access key.  
Create new keys for the testuser IAM user.  
Write the new keys to /home/ansible/newkey.txt.  
Run the playbook /home/ansible/keyUpdate.yml.  
The Ansible control node has been configured and already has Ansible installed. The control node also has a system user named ansible configured with SSH access keys and necessary system privileges.  
  
An IAM user 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.  
  
Logging In  
Use the hands-on lab page to get the public IP of the cloud server we need to log into (making sure to use cloud\_user as a username), then switch to the ansible user using the command su - ansible. The password for the two users is the same.  
  
Create /home/ansible/keyUpdate.yml and Add an Ansible Play that Removes the Old Access Keys for the IAM User testuser and Creates a New Set, Stored in /home/ansible/newkey.txt  
Create the playbook and edit it such that it resembles the following:  
  
- hosts: localhost  
  gather\_facts: no  
  vars\_files:  
    - keys.yml  
  tasks:  
    - name: Get access key  
      iam:  
        aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
        aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
        region: "{{ AWS\_REGION}}"  
        iam\_type: user  
        name: testuser  
        state: present  
      register: iam\_info  
  
    - name: Remove original key  
      iam:  
        aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
        aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
        region: "{{ AWS\_REGION}}"  
        iam\_type: user  
        name: testuser  
        state: update  
        access\_key\_ids: "{{ iam\_info.user\_meta.access\_keys[0].access\_key\_id }}"  
        access\_key\_state: remove  
  
    - name: Create new key  
      iam:  
        aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
        aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
        region: "{{ AWS\_REGION}}"  
        iam\_type: user  
        name: testuser  
        state: update  
        access\_key\_state: create  
      register: new\_key  
  
    - name: Store new access key information  
      lineinfile:  
        create: yes  
        path: /home/ansible/newkey.txt  
        mode: 0600  
        line: "{{ new\_key.created\_keys[0].access\_key\_id }}"  
  
    - name: Store new secret key information  
      lineinfile:  
        path: /home/ansible/newkey.txt  
        line: "{{ new\_key.created\_keys[0].secret\_access\_key }}"  
Run the /home/ansible/keyUpdate.yml Playbook to Perform the Required Tasks  
Run the following command:  
ansible-playbook /home/ansible/keyUpdate.yml  
Conclusion  
If all goes well, we should see a newkey.txt file in our home directory. Run cat newkey.txt to see that it in fact has new keys in it. We're done. Congratulations!

Additional Resources  
Our boss heard about another company's AWS environment being compromised after IAM user credentials were stolen. In order to protect our company's AWS environment, our boss has tasked us with creating automation that updates IAM keys for a provided user. The company is heavily invested in Ansible, so we have been asked to create the automation using Ansible.  
  
We have been provided an Ansible Control node and a sandbox AWS environment.  
  
Start work on the Ansible Control node:  
  
Create the playbook /home/ansible/keyUpdate.yml to perform the following tasks:  
Look up the current IAM access key for the testuser IAM user.  
Remove the old access id key and secret access key.  
Create new keys for the testuser IAM user.  
Write the new keys to /home/ansible/newkey.txt.  
Run the playbook /home/ansible/keyUpdate.yml.  
The Ansible control node has been configured and already has Ansible installed. The control node also has a system user named ansible configured with SSH access keys and necessary system privileges.  
  
An IAM user 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 2 completed  
  
Create `/home/ansible/keyUpdate.yml` and Add an Ansible Play that Removes the Old Access Keys for the IAM User `testuser` and Creates a New Set, Stored in `/home/ansible/newkey.txt`  
  
After logging into the EC2 instance, run su - ansible to become the ansible user. The password is the same as it is for cloud\_user.  
  
Create the playbook and edit it such that it resembles the following:  
  
- hosts: localhost  
  gather\_facts: no  
  vars\_files:  
    - keys.yml  
  tasks:  
    - name: Get access key  
      iam:  
        aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
        aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
        region: "{{ AWS\_REGION}}"  
        iam\_type: user  
        name: testuser  
        state: present  
      register: iam\_info  
  
    - name: Remove original key  
      iam:  
        aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
        aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
        region: "{{ AWS\_REGION}}"  
        iam\_type: user  
        name: testuser  
        state: update  
        access\_key\_ids: "{{ iam\_info.user\_meta.access\_keys[0].access\_key\_id }}"  
        access\_key\_state: remove  
  
    - name: Create new key  
      iam:  
        aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
        aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
        region: "{{ AWS\_REGION}}"  
        iam\_type: user  
        name: testuser  
        state: update  
        access\_key\_state: create  
      register: new\_key  
  
    - name: Store new access key information  
      lineinfile:  
        create: yes  
        path: /home/ansible/newkey.txt  
        mode: 0600  
        line: "{{ new\_key.created\_keys[0].access\_key\_id }}"  
  
    - name: Store new secret key information  
      lineinfile:  
        path: /home/ansible/newkey.txt  
        line: "{{ new\_key.created\_keys[0].secret\_access\_key }}"  
Run the `/home/ansible/keyUpdate.yml` Playbook to Perform the Required Tasks  
  
Run the following command:  
ansible-playbook /home/ansible/keyUpdate.yml

ANSIBLE SERVER :

[root@control cloud\_user]# curl 169.254.169.254/latest/user-data#!/bin/bash/bin/echo 'Jf1UC8&9' | /bin/passwd root —stdin/bin/echo 'Jf1UC8&9' | /bin/passwd cloud\_user —stdin/bin/echo "StrictHostKeyChecking no" >> /etc/ssh/ssh\_configsed -i s/requiretty/\!requiretty/ /etc/sudoers

hostnamectl set-hostname control

yum install -y epel-release

yum install -y ansible sshpass

yum install -y python-boto python-boto3

/sbin/useradd ansible

/bin/echo 'Jf1UC8&9' | /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 'Jf1UC8&9' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost

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

ACCESS=AKIAWBW4S75VUV6XRSXN

SECRET=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

echo "export AWS\_ACCESS\_KEY\_ID='KEYACCESS'" > /home/ansible/accesstemplate

echo "export AWS\_SECRET\_ACCESS\_KEY='KEYSECRET'" > /home/ansible/secrettemplate

echo "#!/bin/sh" >> /home/ansible/keys.shsed -e "s|KEYACCESS|$ACCESS|" /home/ansible/accesstemplate >> /home/ansible/keys.shsed -e "s|KEYSECRET|$SECRET|" /home/ansible/secrettemplate >> /home/ansible/keys.shecho "export AWS\_REGION='us-east-1'" >> /home/ansible/keys.shchmod 700 /home/ansible/keys.shrm -f /home/ansible/accesstemplate /home/ansible/secrettemplateecho "AWS\_ACCESS\_KEY\_ID: KEYACCESS" > /home/ansible/accesstemplateecho "AWS\_SECRET\_ACCESS\_KEY: KEYSECRET" > /home/ansible/secrettemplatesed -e "s|KEYACCESS|$ACCESS|" /home/ansible/accesstemplate >> /home/ansible/keys.ymlsed -e "s|KEYSECRET|$SECRET|" /home/ansible/secrettemplate >> /home/ansible/keys.ymlecho "AWS\_REGION: us-east-1" >> /home/ansible/keys.ymlchmod 600 /home/ansible/keys.ymlrm -f /home/ansible/accesstemplate /home/ansible/secrettemplatechown ansible.ansible /home/ansible/\*echo "finished" > /home/cloud\_user/.done[root@control cloud\_user]# 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"

[root@control cloud\_user]# ansible —version

ansible 2.9.27

config file = /etc/ansible/ansible.cfg

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

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

executable location = /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)]

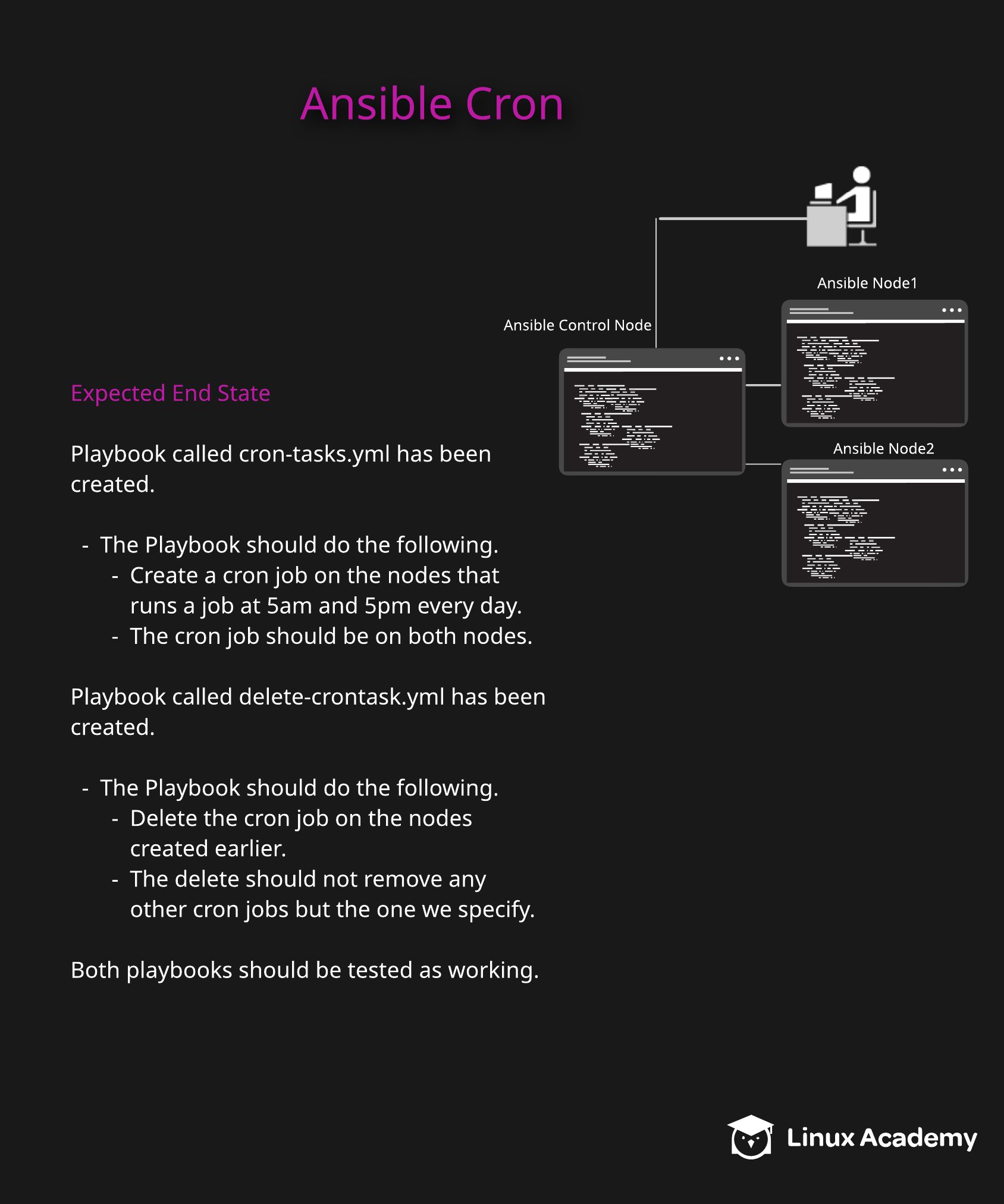
[root@control cloud\_user]#

CFN TEMPLATE :

{  
 "AWSTemplateFormatVersion" : "2010-09-09",  
 "Description" : "Renewing IAM Access Keys with Ansible",  
  
 "Mappings" : {  
  
   "SubnetConfig" : {  
     "VPC"     : { "CIDR" : "10.0.0.0/16" },  
     "Public1"  : { "CIDR" : "10.0.1.0/24" }  
   }  
 },  
  
 "Resources" : {  
  
   "VPC" : {  
     "Type" : "AWS::EC2::VPC",  
     "Properties" : {  
  
       "EnableDnsSupport" : "true",  
       "EnableDnsHostnames" : "true",  
       "CidrBlock" : { "Fn::FindInMap" : [ "SubnetConfig", "VPC", "CIDR" ]},  
       "Tags" : [  
        { "Key" : "Name", "Value" : "LinuxAcademy" },  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "VPC" }  
       ]  
     }  
   },  
  
  
   "PublicSubnet1" : {  
     "Type" : "AWS::EC2::Subnet",  
         "Properties" : {  
       "VpcId" : { "Ref" : "VPC" },  
       "AvailabilityZone" : {  
         "Fn::Select" : [ "0", { "Fn::GetAZs" : "" } ]  
       },  
       "CidrBlock" : { "Fn::FindInMap" : [ "SubnetConfig", "Public1", "CIDR" ]},  
       "Tags" : [  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "Public1" }  
       ]  
     }  
   },  
  
   "InternetGateway" : {  
     "Type" : "AWS::EC2::InternetGateway",  
     "Properties" : {  
       "Tags" : [  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "Public" }  
       ]  
     }  
   },  
  
   "GatewayToInternet" : {  
      "Type" : "AWS::EC2::VPCGatewayAttachment",  
      "Properties" : {  
        "VpcId" : { "Ref" : "VPC" },  
        "InternetGatewayId" : { "Ref" : "InternetGateway" }  
      }  
   },  
  
   "PublicRouteTable" : {  
     "Type" : "AWS::EC2::RouteTable",  
     "Properties" : {  
       "VpcId" : { "Ref" : "VPC" },  
       "Tags" : [  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "Public" }  
       ]  
     }  
   },  
  
   "PublicRoute" : {  
     "Type" : "AWS::EC2::Route",  
     "DependsOn" : "GatewayToInternet",  
     "Properties" : {  
       "RouteTableId" : { "Ref" : "PublicRouteTable" },  
       "DestinationCidrBlock" : "0.0.0.0/0",  
       "GatewayId" : { "Ref" : "InternetGateway" }  
     }  
   },  
  
   "PublicSubnetRouteTableAssociation1" : {  
     "Type" : "AWS::EC2::SubnetRouteTableAssociation",  
     "Properties" : {  
       "SubnetId" : { "Ref" : "PublicSubnet1" },  
       "RouteTableId" : { "Ref" : "PublicRouteTable" }  
     }  
   },  
  
   "PublicNetworkAcl" : {  
     "Type" : "AWS::EC2::NetworkAcl",  
     "Properties" : {  
       "VpcId" : { "Ref" : "VPC" },  
       "Tags" : [  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "Public" }  
       ]  
     }  
   },  
  
   "InboundHTTPPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "100",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "false",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "80", "To" : "80" }  
     }  
   },  
  
   "InboundHTTPSPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "101",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "false",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "443", "To" : "443" }  
     }  
   },  
  
   "InboundSSHPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "102",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "false",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "22", "To" : "22" }  
     }  
   },  
  
   "InboundEmphemeralPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "103",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "false",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "1024", "To" : "65535" }  
     }  
   },  
  
   "OutboundPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "100",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "true",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "0", "To" : "65535" }  
     }  
   },  
  
   "PublicSubnetNetworkAclAssociation1" : {  
     "Type" : "AWS::EC2::SubnetNetworkAclAssociation",  
     "Properties" : {  
       "SubnetId" : { "Ref" : "PublicSubnet1" },  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" }  
     }  
   },  
  
"EC2SecurityGroup" : {  
     "Type" : "AWS::EC2::SecurityGroup",  
     "Properties" : {  
       "GroupDescription" : "Enable access to the EC2 host",  
       "VpcId" : { "Ref" : "VPC" },  
       "SecurityGroupIngress" : [  
         { "IpProtocol" : "tcp", "FromPort" : "22",  "ToPort" : "22",  "CidrIp" : "0.0.0.0/0" },  
         { "IpProtocol" : "tcp", "FromPort" : "80",  "ToPort" : "80",  "CidrIp" : "0.0.0.0/0" },  
         { "IpProtocol" : "tcp", "FromPort" : "443",  "ToPort" : "443",  "CidrIp" : "0.0.0.0/0" },  
         { "IpProtocol" : "icmp", "FromPort" : "-1",  "ToPort" : "-1",  "CidrIp" : "0.0.0.0/0" }  
       ]  
     }  
   },  
  "SGBaseIngress": {  
     "Type": "AWS::EC2::SecurityGroupIngress",  
     "Properties": {  
       "GroupId": { "Ref": "EC2SecurityGroup" },  
       "IpProtocol": "tcp",  
       "FromPort": "80",  
       "ToPort": "80",  
       "SourceSecurityGroupId": { "Ref": "EC2SecurityGroup" }  
     }  
   },  
   "ansible": {  
     "Type": "AWS::IAM::User",  
     "Properties": {  
       "Path": "/",  
       "ManagedPolicyArns" : [ "arn:aws:iam::aws:policy/IAMFullAccess" ],  
       "UserName": "ansible"  
     }  
   },  
   "ansiblekey" : {  
   "Type" : "AWS::IAM::AccessKey",  
   "Properties" : {  
      "UserName" : { "Ref" : "ansible" }  
      }  
   },  
   "testuser": {  
     "Type": "AWS::IAM::User",  
     "Properties": {  
       "Path": "/",  
       "UserName": "testuser"  
     }  
   },  
   "testuserkey" : {  
   "Type" : "AWS::IAM::AccessKey",  
   "Properties" : {  
      "UserName" : { "Ref" : "testuser" }  
      }  
   },  
   "Control" : {  
       "Type" : "AWS::EC2::Instance",  
       "Properties" : {  
           "InstanceType" : "t3.micro",  
           "ImageId" : "ami-09eb574578ce81bb9",  
           "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [  
               "#!/bin/bash\n",  
               "/bin/echo 'Jf1UC8&9' | /bin/passwd root --stdin\n",  
               "/bin/echo 'Jf1UC8&9' | /bin/passwd cloud\_user --stdin\n",  
               "/bin/echo \"StrictHostKeyChecking no\" >> /etc/ssh/ssh\_config\n",  
               "sed -i s/requiretty/\\!requiretty/ /etc/sudoers\n",  
               "hostnamectl set-hostname control\n",  
  
               "yum install -y epel-release\n",  
               "yum install -y ansible sshpass\n",  
               "yum install -y python-boto python-boto3\n",  
  
               "/sbin/useradd ansible\n",  
               "/bin/echo 'Jf1UC8&9' | /bin/passwd ansible --stdin\n",  
               "/bin/echo \"ansible        ALL=(ALL)       NOPASSWD: ALL\" >> /etc/sudoers\n",  
               "sudo -u ansible /bin/mkdir -p /home/ansible/.ssh\n",  
               "sudo -u ansible /bin/ssh-keygen -q -N \"\" -f /home/ansible/.ssh/id\_rsa\n",  
               "sudo -u ansible sshpass -p 'Jf1UC8&9' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost\n",  
               "echo \"localhost\" >> /etc/ansible/hosts\n",  
  
               "ACCESS=",  
               {"Ref" : "ansiblekey" },  
               "\n",  
               "SECRET=",  
               {"Fn::GetAtt" : [  
                   "ansiblekey",  
                   "SecretAccessKey"  
                 ]  
               },  
               "\n",  
  
               "echo \"export AWS\_ACCESS\_KEY\_ID='KEYACCESS'\" > /home/ansible/accesstemplate\n",  
               "echo \"export AWS\_SECRET\_ACCESS\_KEY='KEYSECRET'\" > /home/ansible/secrettemplate\n",  
               "echo \"#!/bin/sh\"  >> /home/ansible/keys.sh\n",  
               "sed -e \"s|KEYACCESS|$ACCESS|\" /home/ansible/accesstemplate  >> /home/ansible/keys.sh\n",  
               "sed -e \"s|KEYSECRET|$SECRET|\" /home/ansible/secrettemplate  >> /home/ansible/keys.sh\n",  
               "echo \"export AWS\_REGION='us-east-1'\"  >> /home/ansible/keys.sh\n",  
               "chmod 700 /home/ansible/keys.sh\n",  
               "rm -f /home/ansible/accesstemplate /home/ansible/secrettemplate\n",  
  
               "echo \"AWS\_ACCESS\_KEY\_ID: KEYACCESS\" > /home/ansible/accesstemplate\n",  
               "echo \"AWS\_SECRET\_ACCESS\_KEY: KEYSECRET\" > /home/ansible/secrettemplate\n",  
               "sed -e \"s|KEYACCESS|$ACCESS|\" /home/ansible/accesstemplate  >> /home/ansible/keys.yml\n",  
               "sed -e \"s|KEYSECRET|$SECRET|\" /home/ansible/secrettemplate  >> /home/ansible/keys.yml\n",  
               "echo \"AWS\_REGION: us-east-1\"  >> /home/ansible/keys.yml\n",  
               "chmod 600 /home/ansible/keys.yml\n",  
               "rm -f /home/ansible/accesstemplate /home/ansible/secrettemplate\n",  
  
               "chown ansible.ansible /home/ansible/\*\n",  
               "echo \"finished\" > /home/cloud\_user/.done\n"  
           ]]}},  
           "NetworkInterfaces" : [{  
               "GroupSet"                 : [{ "Ref" : "EC2SecurityGroup" }],  
               "AssociatePublicIpAddress": "true",  
               "DeviceIndex"              : "0",  
               "DeleteOnTermination"      : "true",  
               "SubnetId"                 : { "Ref" : "PublicSubnet1" }  
           }]  
       }  
   }  
  },  
 "Outputs" : {  
    "pubIpAddress1" : {  
       "Description" : "Public IP of Ansible Control Node",  
       "Value" : { "Fn::GetAtt" : [ "Control", "PublicIp" ] }  
     },  
     "privIpAddress3" : {  
       "Description" : "Private IP of Ansible Control Node",  
       "Value" : { "Fn::GetAtt" : [ "Control", "PrivateIp" ] }  
     }  
 }  
}

33

## Ansible Cron



# Ansible Cron

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

## Introduction

## In this Hands-On Lab we will use a playbook with the archive module for backup purposes.

Note: Ansible has been setup and configured for your use.

## Solution

## To access the lab environment, log into the control node as cloud\_user, using the IP address and login credentials provided in the hands-on lab page.

### Sign onto the Ansible Control Node Server as cloud\_user and Change to the ansible User. Test to Ensure Ansible Has Been Set up for You

### Sign into the server called Ansible Control Node, as cloud\_user, and become ansible with su - ansible.

Test that Ansible is working via an ad-hoc command.

A possible command to use is the following:

ansible all -m ping

### Create a Playbook Called cron-tasks.yml That Performs a Task (df -h >> /tmp/diskspace) on the Nodes at 5:00AM and 5:00PM

### Using the **cron** module, create a playbook that adds a cron job to the nodes. The job should run at 5AM and 5PM. The task to run is df -h >> /tmp/diskspace.

Here is cron-tasks.yml:

---  
- hosts: dbsystems  
  user: ansible  
  become: yes  
  gather\_facts: no  
  tasks:  
  - name: Ensure a job that runs at 5am and 5pm exists. Creates an entry like "0 5,17 \* \* \* df -h >> /tmp/diskspace"  
    cron:  
      name: "Job 0001"  
      minute: "0"  
      hour: "5,17"  
      job: "df -h >> /tmp/diskspace"

### Run the Playbook and Test the Cron Job Exists on the Nodes

### Run the playbook (with ansible-playbook cron-tasks.yml) and then log into one of the nodes. Check that the cron job exists, with sudo crontab -l.

It should look similar to this:

0 5,17 \* \* \* df -h >> /tmp/diskspace

### Create a Playbook to Remove the Cron Job. It Should Be Called delete-crontask.yml and Should Only Delete That Cron Job

### Now log out of the node, and let's create a new playbook. This new playbook should delete the cron job we just created, but leave any other cron jobs alone. This delete-crontask.yml should look like this:

---  
- hosts: dbsystems  
  user: ansible  
  become: yes  
  gather\_facts: no  
  tasks:  
  - name: 'Ensure an old job is no longer present. Removes any job that is prefixed by "#Ansible: Job 0001" from the crontab'  
    cron:  
      name: "Job 0001"  
      state: absent

Before we go putting this into play, let's get back into cron-tasks.yml. Change the time to 4:00AM, instead of 5:00, and change the name: to be 0002. Now run that playbook again with ansible-playbook cron-tasks.yml. Just like last time, let's log into one of the nodes and check. Run a sudo crontab -l, and we should now see **Job 0001** and **Job 0002**. Now we can exit out of this node.

### Run the Playbook, Then Confirm That the Job Added Earlier Has Been Deleted

### Back on the control machine, run the new playbook (ansible-playbook delete-crontask.yml). Then, once again, log into a node and confirm that the original cron job has been deleted.

## Conclusion

## That's all folks. We've created an Ansible playbook that sets up a cron job, and another playbook that will delete a job. This is what we wanted. Congratulations!

## Additional Resources

Our manager wants us to use Ansible for scheduling a task across all of our servers, using the Cron module. This is to show management that Ansible can be used for this task.

In this hands-on lab, we will make use of the Cron module to create scheduled tasks. Then we will show how to set up and remove a task from crontab.

To summarize, using an Ansible playbook we must do the following:

1. Create a playbook called cron-tasks.yml that performs a task (df -h >> /tmp/diskspace) on the nodes at 5am and 5pm.
2. Run the playbook and test that the cron job exists on the nodes.
3. Create a playbook that removes that task.
4. Run the playbook and confirm that the cron job has been removed from the nodes.

Here are some important things to take into consideration while completing each task:

* For convenience, Ansible is already on the control node. If we connect to the server by clicking on the Public IP address in a browser, we need to make sure we change to the ansible user with the su - ansible command.
* The user ansible is present on all servers, with appropriate shared keys for access to managed servers from the control node. Make sure to use the ansible user to complete the commands.
* The ansible user has the same password as cloud\_user.
* The default Ansible inventory has been configured for with the appropriate hosts and groups.
* /etc/hosts entries are present on control1 for the managed servers.

## Learning Objectives

0 of 5 completed

Sign onto the Ansible Control Node Server as cloud\_user and Change to the ansible User. Test to Ensure Ansible Has Been Set up for You

Sign into the server called Ansible Control Node using the cloud\_user and change to the ansible user via the su - ansible command.

Test that Ansible is working via an ad-hoc command.

A possible command to use is the following:

ansible all -m ping

Note: It may take several minutes after the start of the lab, for the test to work as expected.

Create a Playbook Called cron-tasks.yml That Performs a Task ('df -h >> /tmp/diskspace') on the Nodes at 5:00AM and 5:00PM.

Using the **cron** module, create a playbook that adds a cron job to the nodes. The job should run at 5AM and 5PM. The task to run is df -h >> /tmp/diskspace.

Run the Playbook and Test Whether the Cron Job Exists on the Nodes

Run the playbook and then log into the nodes and check that the cron job exists.

It should look similar to this:

0 5,17 \* \* \* df -h >> /tmp/diskspace

Create a Playbook to Remove the Cron Job. It Should Be Called delete-crontask.yml and Should Only Delete That Cron Job

Create a new playbook. This new playbook should delete the cron job you just created, but leave any other cron jobs there.

Run the Playbook, Then Confirm That the Job Added Earlier Has Been Deleted

Run the playbook and confirm that the original cron job has been deleted.

ANSIBL EHOST :

[root@control1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash -xe

/bin/echo 'nnDz-H1A' | /bin/passwd root —stdin

/bin/echo 'nnDz-H1A' | /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 10.0.1.252 db1 node1>> /etc/hosts

/bin/echo 10.0.1.243 db2 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 'nnDz-H1A' | /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 'nnDz-H1A' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost

sudo -u ansible sshpass -p 'nnDz-H1A' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@db1

sudo -u ansible sshpass -p 'nnDz-H1A' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@db2

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

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

echo "db1" >> /etc/ansible/hosts

echo "db2" >> /etc/ansible/hosts

/bin/yum install -y git

/bin/chmod -R ansible:ansible /home/ansible

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

[root@control1 cloud\_user]# sudo cat /etc/os-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"

[root@control1 cloud\_user]# ansible —version

ansible 2.9.27

config file = /etc/ansible/ansible.cfg

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

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

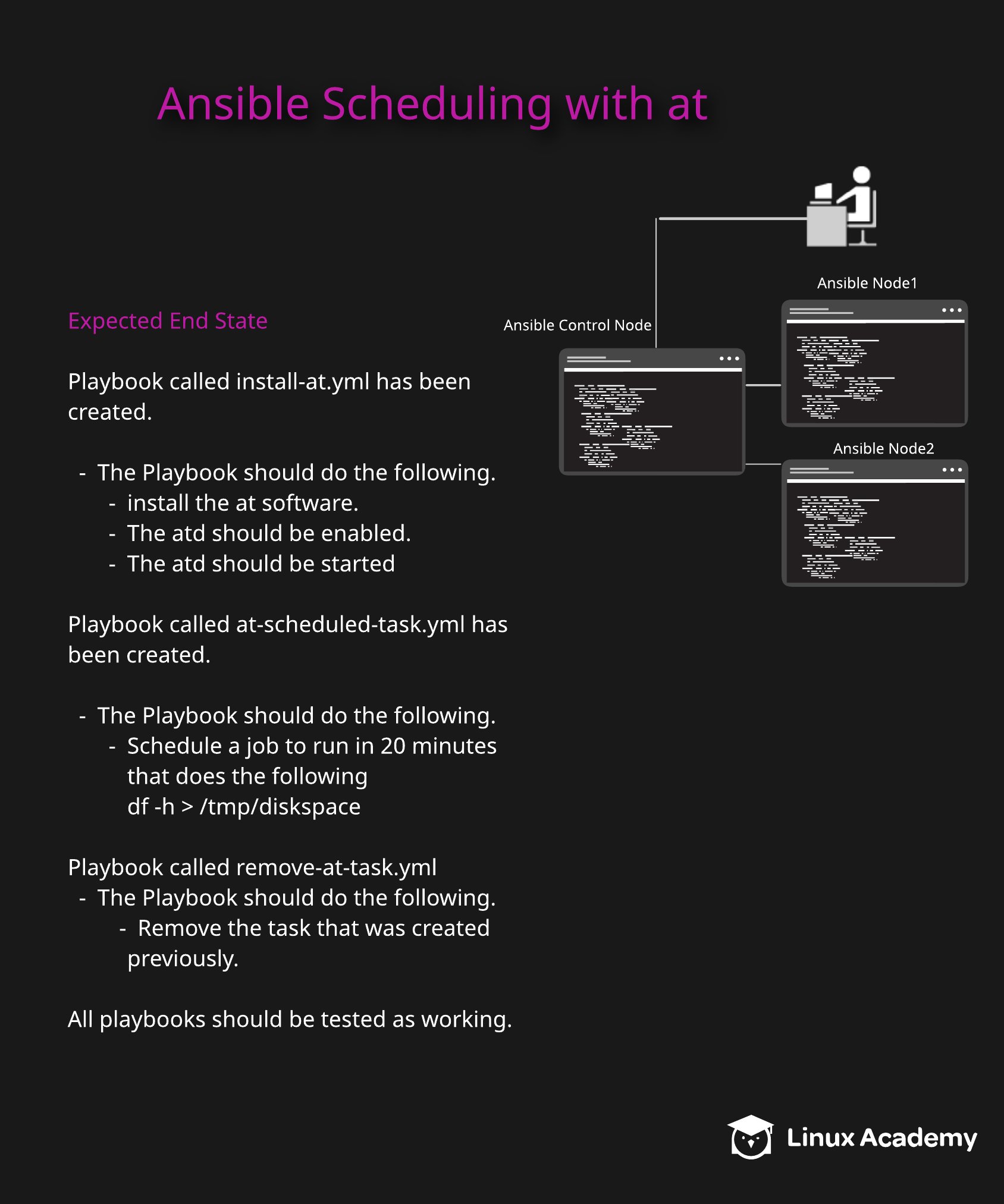
executable location = /bin/ansible

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

[root@control1 cloud\_user]#

33

## Tackling Ansible Scheduling with the `at` Command



# ackling Ansible Scheduling with the at Command

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

## Introduction

## In this hands-on lab, we will use a playbook with the at module for job scheduling purposes.

**Note:** Ansible has been set up and configured already for use.

## Solution

## Log in to the Ansible Control Node server using the credentials provided:

ssh cloud\_user@<ANSIBLE\_CONTROL\_NODE\_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.*

### Log in to the Ansible Control Node Server and Change to the ansible User

Become the ansible user:

su - ansible

Test whether Ansible is working:

ansible all -m ping

You should see success messages.

**Note:** If you get a "service is not working" message, this may be because it can take several minutes after the start of the lab for the test to work as expected. Wait a few minutes before trying the command again.

### Create and Run a Playbook Called install-at.yml that Installs, Enables, and Starts the at Service on All Nodes

Create a playbook called install-at.yml that will :

vim install-at.yml

Enter the following contents, which will install the at service on all nodes as well as enable and start the atd service:

---  
- hosts: all  
  user: ansible  
  become: yes  
  gather\_facts: no  
  tasks:  
    - name: install the at command for job scheduling  
      action: yum name=at state=installed  
  
    - name: Enable and Start service at if not started  
      service:  
        name: atd  
        enabled: yes  
        state: started

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

Run the playbook and ensure it installs, enables, and starts the service correctly:

ansible-playbook install-at.yml

View more information about our at command:

man at

Press q to get back to the command prompt.

### Create a Playbook Called at-scheduled-task.yml to Perform a Task on the Nodes in 20 Minutes

Create a playbook called at-scheduled-task.yml:

vim at-scheduled-task.yml

Enter the following:

---  
- hosts: all  
  user: ansible  
  become: no  
  gather\_facts: no  
  tasks:  
    - name: Schedule a command to execute in 20 minutes as the ansible user  
      at:  
        command: df -h > /tmp/diskspace  
        count: 20  
        units: minutes

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

Run the playbook:

ansible-playbook at-scheduled-task.yml

### Run the at-scheduled-task Playbook and Test Each Node to Ensure the Task Is Scheduled

Verify the job has been scheduled:

atq

We'll see a job listed that's set to run.

See what the job is:

at -c 1

There will be a lot in the output — including the actual command we set: df -h > /tmp/diskspace

### Create a Playbook Called remove-at-task.yml to Remove the Previously Scheduled Task

Create the playbook:

vim remove-at-task.yml

Enter the following, which will remove the task that was set in the at-scheduled-task.yml:

---  
- hosts: all  
  user: ansible  
  become: no  
  gather\_facts: no  
  tasks:  
  
  - name: Match a command to an existing job and delete the job  
    at:  
      command: df -h > /tmp/diskspace  
      state: absent

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

Run the playbook:

ansible-playbook remove-at-task.yml

Ensure the tasks have been removed:

atq

There shouldn't be anything returned this time, as the job has been deleted.

## Conclusion

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

## Additional Resources

Your manager has been told that Ansible can be used to schedule one-off tasks on multiple servers at the same time. To show this is possible, you will need to use the at module in several Ansible playbooks to manage the scheduling of a one-time task.

To do this assignment, you will make use of the at module to create a scheduled task, which will help demonstrate you know how to set up and remove a task from the list of jobs.

Using an Ansible playbook requires you to perform the following steps:

1. Create a playbook called install-at.yml that will install and start the atd service.
2. Create a playbook called at-scheduled-task.yml to perform a task (df -h > /tmp/diskspace) on the nodes set to start in 20 minutes.
3. Run the playbook, and ensure the job exists on the nodes.
4. Create a playbook called remove-at-task.yml that removes the task.
5. Run the playbook and confirm the task has been removed from the nodes.

Here are some important things to consider while completing each task:

* Ansible is already on the control node. If you connect to the server by clicking on the **Public IP** address in your browser, make sure to change to the ansible user with the su - ansible command.
* The ansible user is present on all servers with appropriate shared keys for access to managed servers from the control node.
* The ansible user has the same password as cloud\_user.
* The default Ansible inventory has been preconfigured for you with the appropriate hosts and groups.
* /etc/hosts entries are present on control1 for the managed servers.

For detailed instructions on how to complete these tasks, click the *Guide* tab above the video player.

## Learning Objectives

0 of 5 completed

Log in to the Ansible Control Node Server and Change to the `ansible` User

Sign in to the Ansible Control Node server using the cloud\_user credentials, and change to the ansible user.

Test that Ansible is working:

ansible all -m ping

**Note:** It may take several minutes after the start of the lab for the test to work as expected.

Create and Run a Playbook Called `install-at.yml` that Installs, Enables, and Starts the `at` Service on All Nodes

1. Create a playbook called install-at.yml that will install the at service on all nodes.
2. Enable the atd service and start the service as part of the playbook.
3. Run the playbook and ensure it installs, enables, and starts the service correctly.

Create a Playbook Called `at-scheduled-task.yml` to Perform a Task on the Nodes in 20 Minutes

1. Using the at module, create a playbook called at-scheduled-task.yml that adds a scheduled task to the nodes.
2. The job should run in 20 minutes. The task to run is df -h > /tmp/diskspace.

Run the `at-scheduled-task` Playbook and Test Each Node to Ensure the Task Is Scheduled

1. Use the atq command to ensure the df -h task has been scheduled.

Create a Playbook Called `remove-at-task.yml` to Remove the Previously Scheduled Task

The remove-at-task.yml playbook should remove the task that was set in the at-scheduled-task.yml playbook.

We can test with the atq command to ensure the task has been removed.

**Note:** If they have not, we must troubleshoot why they have not been removed.

Server " :

[root@control1 ~]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo 'D]!l5mOh' | /bin/passwd root —stdin

/bin/echo 'D]!l5mOh' | /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 10.0.1.55 db1 node1>> /etc/hosts

/bin/echo 10.0.1.76 db2 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 'D]!l5mOh' | /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 'D]!l5mOh' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost

sudo -u ansible sshpass -p 'D]!l5mOh' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@db1

sudo -u ansible sshpass -p 'D]!l5mOh' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@db2

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

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

echo "db1" >> /etc/ansible/hosts

echo "db2" >> /etc/ansible/hosts

/bin/yum install -y git

/bin/chmod -R ansible:ansible /home/ansible

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

[root@control1 ~]# sudo cat /etc/os-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"

[root@control1 ~]# ansible —version

ansible 2.9.27

config file = /etc/ansible/ansible.cfg

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

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

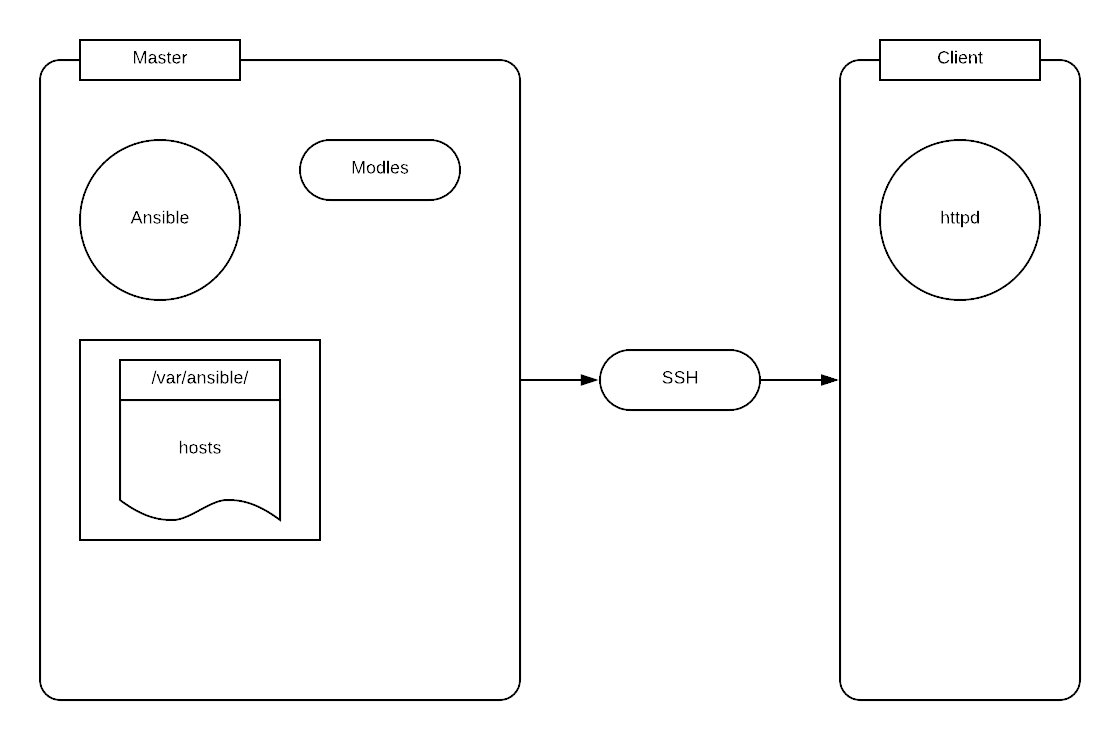
executable location = /bin/ansible

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

[root@control1 ~]#

34

## Exchange SSH Keys and Run Ansible Playbooks



# Exchange SSH Keys and Run Ansible Playbooks

## Introduction

## Ansible allows you to manage the configuration of your hardware and software. Since Ansible is agentless, it can be run from any system that has account credentials on any other system using standard SSH credentials and commands. In this hands-on lab, we will practice running a small playbook on a remote host to install a web server and then test it on our primary server.

## Setting Up the Environment

### Log In to the Master Server

1. Open your terminal application, and log in to the master server using the credentials provided on the lab instructions page.

ssh cloud\_user@&lt;MASTER\_PUBLIC\_IP&gt;

1. Type yes at the prompt.
2. Enter your password at the prompt.

### Log In to the Client Server

1. Open a new window of your terminal, and log in to the client server using the credentials provided on the lab instructions page.

ssh cloud\_user@&lt;CLIENT\_PUBLIC\_IP&gt;

1. Type yes at the prompt.
2. Enter your password at the prompt.

## Create the SSH Keys

1. In your master server terminal window, add a new user called ansible.

adduser ansible

* Set up a password for the new user.

passwd ansible

* Enter a password for the user at the prompt. Enter it again when you're asked to confirm.
* Enable the ansible user to run commands as the root user without using a password.

visudo

* In the sudoers file, locate the section that says Allow root to run any commands anywhere.
* Under the line below it (which starts with root), add the following:

ansible     ALL=(ALL)     NOPASSWD: ALL

* Press **ESC** and type :wq to save our changes and exit the vi text editor.
* Switch to your client server terminal window, and repeat steps 1-7.
* Go back to your master server terminal window.
* Change to the ansible user.

su ansible

* Create an SSH key.

ssh-keygen

* Press **Enter** to accept the defaults.
* Copy the SSH key to the client.

ssh-copy-id &lt;IP\_ADDRESS&gt;

* Type yes at the prompt.
* Enter your password at the prompt.
* Exit Ansible.

exit

* Update the Ansible hosts file.

vi /etc/ansible/hosts

* At the top of the page, add the word client.
* Press **ESC** and type :wq to save the change and exit the vi text editor.
* Change to the /home/cloud\_user directory.

cd /home/cloud\_user

* List the contents of the directory.

ls

* Copy the playbook in this directory to the /ansible directory.

cp playbook.yml /home/ansible

* Change to the /ansible directory.

cd /home/ansible

* List the contents of the /ansible directory to verify that a copy of the playbook.yml file is there.

ls

* Change the permissions on the playbook.yml file.

chown ansible:ansible playbook.yml

* Check the permissions of the directory.

ls -la

* Change to the ansible user.

su ansible -

* List the contents of the current directory.

ls

* Execute the Ansible playbook.

ansible-playbook playbook.yml

## Install the Elinks Package on the Remote Client Server

1. In the client server window of your terminal application, run the following command:

elinks

* You should see an ELinks Welcome menu.

## Conclusion

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

## Additional Resources

## Scenario

## Your development team is working on a new web application service and needs a basic cluster of web servers that they can throw load against in order to test performance.

As a result, you have been asked to provide a basic cluster that encompasses one management node and one client node. You will need to create the manager and join the client to the cluster and confirm they are all registered appropriately.

## Instructions

## You have been asked to help your team work with Ansible to manage software packages. As a result, you have been provided credentials and connectivity information for two servers, one master and one client.

When you connect to the master server, you will see that Ansible has already been installed for you. You will find that there is a simple playbook in the /home/cloud\_user directory that will install the elinks web browser on the remote client system as a test. However, they have been unable to get it to run.

On the master and client, create a new user called ansible. Give the user a password. You can use the same password you used for cloud\_user. Make sure you have created SSH keys for the ansible user and exchanged that key with the remote system. Once you verify the keys, allow access without a password, and be sure that the ansible account on the local *and* remote systems does not require a password when running commands needing escalation privileges (e.g., sudo). Copy playbook.yml from /home/cloud\_user to /home/ansible and make sure that the playbook has the correct owner permissions.

Finally, run the playbook called playbook.yml in the /home/ansible directory and then log in to the client system and verify that the package is installed. Your team can then use that playbook template and configuration to install any software they want thereafter.

## Learning Objectives

0 of 2 completed

Create the SSH Keys for Exchanging between Master and Client Servers

1. Create a new user called ansible and set the password.

adduser ansible  
passwd ansible

1. Add the ansible user to the sudoers file and make sure that it can use sudo without a password.

visudo

1. While logged in as ansible user, create the necessary keys.

ssh-keygen

1. Exchange the key with the remote client server.

ssh-copy-id 10.0.1.101

1. Add the client to the Ansible host file.

vi /etc/ansible/hosts

Install the Elinks Package on the Remote Client Server

1. Run the playbook on the master.

ansible-playbook /home/cloud\_user/playbook.yml

1. Once the software is installed (it should show a success message), log in to the remote system and run the following:

elinks

1. You should see an *About* screen on your console.

35

## Using Ansible Modules to Manage Archiving in Your Environment.



# Using Ansible Modules to Manage Archiving in Your Environment

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

## Introduction

## Dealing with compressed files is one of the less glamorous things that a System Administrator is called upon to do. Fortunately, Ansible has modules to help work with archived files as well. This lab will help cement those skills.

## The Scenario

## We've gotten a ticket from the DBA team that says they need a couple of tarballs unpacked on the DB server. One is on the Ansible host, /root/file.tar.gz, while the other is /tmp/devstuff.tar.gz on the DB Server. Both need to be extracted into /opt on the DB Server.

We have another ticket requesting that we compress /var/log/messages on the DB server and put the result in /tmp on the Ansible host. We've decided to do both tasks at once.

We need to write a playbook that will unpack all of the mentioned tarballs, compress /var/log/messages on dbserver1, and then get that tarball into /tmp on the Ansible Host.

## Logging In

## Use the credentials provided on the hands-on lab page to get into Server1 to begin with. Since we need root privileges, let's just run sudo -i right off and become root.

## Unpack the /root/file.tar.gz Tarball into /opt on DBServer1

## Let's create a playbook:

vim archive.yml

We'll put the first task in, which will be what extracts a local archive. It should look similar to this when we're done:

---  
- name: Copy data dev stuff and inflate data and dev stuff  
  hosts: dbservers  
  become: yes  
  
  tasks:  
  - unarchive:  
     src: /root/file.tar.gz  
     dest: /opt/

## Unpack the devstuff Tarball into /opt on dbserver1

## This part will extract a remote archive. It should look similar to this when we're finished:

  - unarchive:  
     src: /tmp/devstuff.tar.gz  
     dest: /opt/  
     remote\_src: yes

## Compress and Copy /var/log/messages from dbserver1 to the Ansible Host

## The next play in the book creates an archive on dbserver1, then retrieve it. When we're done, the section should look like this:

  - archive:  
     path: /var/log/messages  
     dest: /tmp/messages.tar.gz  
  
  - fetch:  
     src: /tmp/messages.tar.gz  
     dest: /tmp/

## Run the Playbook

## As long as we don't have any typos in our playbook, we should see all of the successful task output march across the screen.

## Conclusion

## We've done it. There's a playbook now that will extract remote archives, then create others that we fetch to the Ansible host. That's exactly what we wanted. Congratulations!

## Additional Resources

Notice: Ansible is installed as the root user, so please work on all tasks after elevating to the root user.

We've gotten a ticket from the DBA team that says they need a couple of tarballs unpacked on the DB server. One is on the Ansible Host, /root/file.tar.gz, while the other is /tmp/devstuff.tar.gz on the DB Server. Both need to be extracted into /opt on the DB Server.

We have another ticket requesting that we compress /var/log/messages on the DB server and put the result in /tmp on the Ansible host. We've decided to do both tasks at once.

We need to write a playbook that will unpack all of the mentioned tarballs, compress /var/log/messages on DBServer1, and then get that tarball into /tmp on the Ansible Host.

## Learning Objectives

0 of 3 completed

Unpack the /root/file.tar.gz Tarball into /opt on DBServer1

Your playbook should look similar to:

- name: Copy data dev stuff and inflate data and dev stuff  
  hosts: dbservers  
  become: yes  
  
  tasks:  
  - unarchive:  
     src: /root/file.tar.gz  
     dest: /opt/

Unpack the devstuff Tarball into /opt on dbserver1

This part of your playbook should look similar to:

- unarchive:  
   src: /tmp/devstuff.tar.gz  
   dest: /opt/  
   remote\_src: yes

Compress and Copy /var/log/messages from dbserver1 to the Ansible Host

The last part of your playbook should look like this:

  - archive:  
     path: /var/log/messages  
     dest: /tmp/messages.tar.gz  
  
  - fetch:  
     src: /tmp/messages.tar.gz  
     dest: /tmp/

HOST :

[root@Server1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/usr/bin/hostnamectl set-hostname Server1

/usr/bin/yum install wget git -y

/bin/echo 'source /root/ansible/hacking/env-setup -q' >> /root/.bashrc

/bin/ssh-keygen -N '' -t rsa -f ~/.ssh/id\_rsa

/bin/cat /root/.ssh/id\_rsa.pub >> /home/ansible/.ssh/authorized\_keys

/bin/echo 'python\_interpreter=auto' >> /etc/ansible/ansible.cfg

/bin/echo 'host\_key\_checking = False' >> /etc/ansible/ansible.cfg

/sbin/alternatives —set python /usr/bin/python3

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/privkey> -O /tmp/pkey

/bin/cat /tmp/pkey > /root/.ssh/id\_rsa

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/authkey> -O /tmp/akey

/bin/cat /tmp/akey > /root/.ssh/id\_rsa.pub

/bin/cat /tmp/akey > /home/ansible/.ssh/id\_rsa.pub

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/etchosts> -O /tmp/hosts

/bin/cat /tmp/hosts > /etc/hosts

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/inventory> -O /tmp/inv

/bin/cat /tmp/inv > /etc/ansible/hosts

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

/bin/dd if=/dev/random of=/root/filename.sql bs=10M count=50

/bin/tar -czf /root/file.tar.gz /root/filename.sql

/usr/bin/echo 'cloud\_user:b|9dKN(6' | /usr/sbin/chpasswd

/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-2289-73e5ee0bbae43242a5625d91f11322e2 —resource PublicInstance —region us-east-1

[root@Server1 cloud\_user]# sudo cat /etc/os-release

NAME="Red Hat Enterprise Linux"

VERSION="8.7 (Ootpa)"

ID="rhel"

ID\_LIKE="fedora"

VERSION\_ID="8.7"

PLATFORM\_ID="platform:el8"

PRETTY\_NAME="Red Hat Enterprise Linux 8.7 (Ootpa)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:redhat:enterprise\_linux:8::baseos"

HOME\_URL="<https://www.redhat.com/>“

DOCUMENTATION\_URL="<https://access.redhat.com/documentation/red_hat_enterprise_linux/8/>“

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

REDHAT\_BUGZILLA\_PRODUCT="Red Hat Enterprise Linux 8"

REDHAT\_BUGZILLA\_PRODUCT\_VERSION=8.7

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

REDHAT\_SUPPORT\_PRODUCT\_VERSION="8.7"

[root@Server1 cloud\_user]# ansible —version

ansible 2.10.0.dev0

config file = /etc/ansible/ansible.cfg

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

ansible python module location = /root/ansible/lib/ansible

executable location = /root/ansible/bin/ansible

python version = 3.6.8 (default, Jan 11 2023, 08:43:50) [GCC 8.5.0 20210514 (Red Hat 8.5.0-16)]

[root@Server1 cloud\_user]#

NODE :

[root@WebServer1 cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/usr/bin/hostnamectl set-hostname WebServer1

/usr/bin/yum install wget -y

/sbin/useradd ansible

/usr/bin/echo 'ansible:b|9dKN(6' | /usr/sbin/chpasswd

/sbin/alternatives —set python /usr/bin/python3

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/etchosts> -O /tmp/hosts

/bin/cat /tmp/hosts > /etc/hosts

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/inventory> -O /tmp/inv

/bin/cat /tmp/inv > /etc/ansible/hosts

/usr/bin/wget <https://raw.githubusercontent.com/linuxacademy/content-linux-rhcev8/master/authkey> -O /tmp/key

/bin/mkdir /home/ansible/.ssh

/usr/bin/chmod 0700 /home/ansible/.ssh/

/bin/cat /tmp/key > /home/ansible/.ssh/authorized\_keys

/usr/bin/chmod 0600 /home/ansible/.ssh/authorized\_keys

/usr/bin/chown ansible.ansible /home/ansible -R

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

/usr/bin/echo 'cloud\_user:b|9dKN(6' | /usr/sbin/chpasswd

/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-2289-73e5ee0bbae43242a5625d91f11322e2 —resource PublicInstance2 —region us-east-1

[root@WebServer1 cloud\_user]# sudo cat /etc/os-release

NAME="Red Hat Enterprise Linux"

VERSION="8.9 (Ootpa)"

ID="rhel"

ID\_LIKE="fedora"

VERSION\_ID="8.9"

PLATFORM\_ID="platform:el8"

PRETTY\_NAME="Red Hat Enterprise Linux 8.9 (Ootpa)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:redhat:enterprise\_linux:8::baseos"

HOME\_URL="<https://www.redhat.com/>“

DOCUMENTATION\_URL="<https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8>“

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

REDHAT\_BUGZILLA\_PRODUCT="Red Hat Enterprise Linux 8"

REDHAT\_BUGZILLA\_PRODUCT\_VERSION=8.9

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

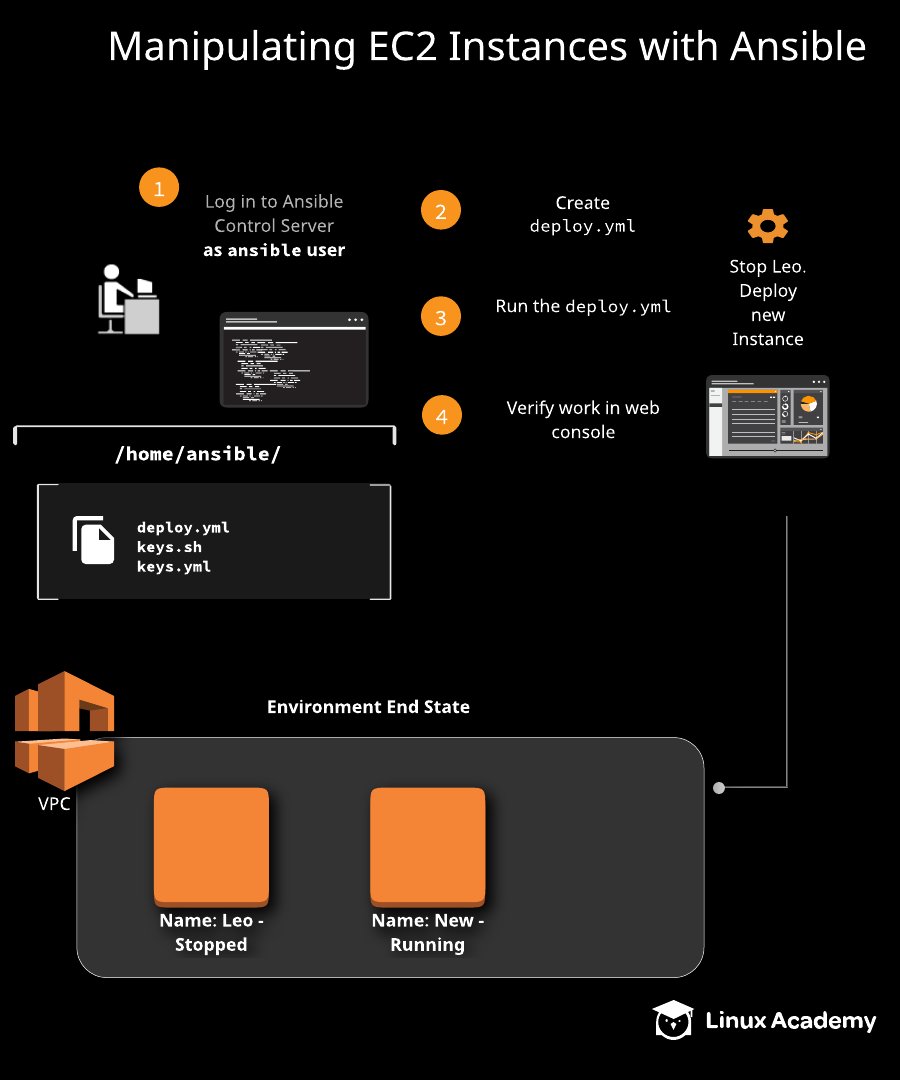
REDHAT\_SUPPORT\_PRODUCT\_VERSION="8.9"

[root@WebServer1 cloud\_user]# ansible —version

bash: ansible: command not found...

36

## Manipulating EC2 Instances with Ansible.



Manipulating EC2 Instances with Ansible  
Introduction  
EC2 is at the heart of AWS as the primary compute resource on the platform. Ansible provides several modules that allow us to interact with EC2 instances. Being able to provision and manipulate EC2 instances within Ansible allows for infrastructure automation to be built into a deployment strategy. This exercise will allow students to explore the EC2 functionality in Ansible.  
  
Instructions  
We have been tasked with creating automation that will redeploy an EC2 instance from an updated AMI. We will need to stop the currently running instance, tagged with Name: Leo. Then we will need to deploy a single new EC2 instance meeting the following requirements:  
  
Type: t2.micro  
AMI: Same as existing instance  
Region: us-east-1  
Public IP: Yes  
VPC Subnet ID: Same as existing instance  
Assign the tag Name: New to the new instance  
We will need to consult the AWS console (or, alternatively, use Ansible facts) to determine the subnet ID and AMI ID of the existing instance, in order to assign the correct value for the new instance.  
  
From the Ansible Control node:  
  
Create the playbook /home/ansible/deploy.yml to perform the following tasks:  
  
Stop the EC2 instance tagged Name: Leo.  
Deploy a new EC2 instance meeting the described properties.  
Run the playbook /home/ansible/deploy.yml  
  
Validate that our work in the AWS Web Console is correct.  
  
The Ansible control node has been configured and already has Ansible installed. The control node also has a system user named ansible configured with SSH access keys and necessary system privileges.  
  
An IAM user 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.  
  
Logging In  
Use the hands-on lab page to get the public IP of the cloud server we need to log into (making sure to use cloud\_user as a username), then switch to the ansible user. The password for the two users is the same.  
  
Create and Edit /home/ansible/deploy.yml and Add Ansible Tasks to Stop the Existing EC2 Instance, by Tag, Then Deploy a New EC2 Instance That Meets the Specification Described in the Instructions.  
Create and edit the playbook (/home/ansible/deploy.yml) so that it resembles the following:  
  
- hosts: localhost  
  gather\_facts: no  
  vars\_files:  
    - /home/ansible/keys.yml  
  tasks:  
    - name: Get Subnet ID and AMI ID from existing server.  
      ec2\_instance\_facts:  
        aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
        aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
        region: "{{ AWS\_REGION }}"  
        filters:  
          tag:Name: Leo  
      register: ec2\_facts  
  
    - name: Stop Leo Instance  
      ec2:  
        aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
        aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
        ec2\_region: us-east-1  
        state: stopped  
        instance\_tags:  
          Name: Leo  
  
    - name: Deploy new EC2 Instance  
      ec2:  
       aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
       aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
       ec2\_region: us-east-1  
       instance\_type: t2.micro  
       image: "{{ ec2\_facts.instances[0].image\_id }}"  
       assign\_public\_ip: yes  
       vpc\_subnet\_id: "{{ ec2\_facts.instances[0].subnet\_id }}"  
       instance\_tags:  
         Name: New  
Run the Playbook /home/ansible/deploy.yml to Perform the Required Tasks, Then Log into the AWS Console to Validate that Everything Works  
Run the following command:  
ansible-playbook /home/ansible/deploy.yml  
Log into the AWS Console, and in the EC2 Dashboard (find it by searching for EC2 in the Find Services search box) confirm the new instance's existence and state.  
It might be best to wait a bit before checking. Once everything is finished processing though, we'll see a Leo instance that's stopped, and a new one called New that is running.  
Conclusion  
We've finished. We were able to do exactly what people required of us, which was to create an Ansible playbook that fires up an EC2 instance named New after stopping one named Leo. Congratulations