**Task 9: Create and Work with Roles**

Create a role called **sample-mysql** and store it in /home/automation/plays/roles. The role should satisfy the following requirements:

* A primary partition number 1 of size 800MB on device /dev/sdb is created.
* An LVM volume group called vg\_database is created that uses the primary partition created above.
* An LVM logical volume called lv\_mysql is created of size 512MB in the volume group vg\_database.
* An XFS filesystem on the logical volume lv\_mysql is created.
* Logical volume lv\_mysql is permanently mounted on /mnt/mysql\_backups.
* **mysql-community-server** package is installed.
* Firewall is configured to allow all incoming traffic on MySQL port TCP 3306.
* MySQL root user password should be set from the variable **database\_password** (see task #5).
* MySQL server should be started and enabled on boot.
* MySQL server configuration file is generated from the my.cnf.j2 Jinja2 template with the following content:

[mysqld]

bind\_address = {{ ansible\_default\_ipv4.address }}

skip\_name\_resolve

datadir=/var/lib/mysql

socket=/var/lib/mysql/mysql.sock

symbolic-links=0

sql\_mode=NO\_ENGINE\_SUBSTITUTION,STRICT\_TRANS\_TABLES

[mysqld\_safe]

log-error=/var/log/mysqld.log

pid-file=/var/run/mysqld/mysqld.pid

Create a playbook /home/automation/plays/mysql.yml that uses the role and runs on hosts in the **database** host group.

To create the role template:

$ cd roles

$ ansible-galaxy role init --offline sample-mysql

The playbook:

---

- name: task 9

hosts: database

become: true

vars\_files:

- secret.yml

roles:

- sample-mysql

The role file:

---

# tasks file for sample-mysql

I spent quite some time in this one. First of all, `mysql-community-server` is not available in the configure repos, so I used `mysql-server` instead. But it took some time to figure out that `mysql` and `python3-PyMySQL` are also required.

- name: create primary partition

parted:

device: /dev/nvme0n2

number: 1

flags: [ lvm ]

state: present

part\_end: 800MB

- name: create VG vg\_database using the primary partition created above

lvg:

vg: vg\_database

pvs: /dev/nvme0n2p1

- name: create LV lv\_mysql size 512MB in the VG vg\_database

lvol:

vg: vg\_database

lv: lv\_mysql

size: 512m

- name: create an XFS filesystem on lv\_mysql

filesystem:

fstype: xfs

dev: /dev/vg\_database/lv\_mysql

- name: ensure mount point /mnt/mysql\_backups exists

file:

path: /mnt/mysql\_backups

state: directory

owner: root

group: root

mode: 775

- name: permanently mount filesystem

mount:

path: /mnt/mysql\_backups

src: /dev/vg\_database/lv\_mysql

fstype: xfs

state: mounted

- name: install mysql-server

yum:

name: "{{ item }}"

state: latest

loop:

- mysql-server

- mysql

- python3-PyMySQL

- name: allow mysql traffic

firewalld:

service: mysql

permanent: true

immediate: true

state: enabled

- name: start and enable mysql

service:

name: mysqld

state: started

enabled: true

- name: configure root user

mysql\_user:

check\_implicit\_admin: true

login\_host: localhost

login\_user: root

login\_password: ''

name: root

password: "{{ database\_password }}"

state: present

update\_password: always

# no\_log: true

- name: deploy configuration

template:

src: mysql.j2

dest: /etc/my.cnf

owner: root

group: root

mode: 0644

**Task 10: Create and Work with Roles (Some More)**

Create a role called **sample-apache** and store it in /home/automation/plays/roles. The role should satisfy the following requirements:

* The **httpd**, **mod\_ssl** and **php** packages are installed. Apache service is running and enabled on boot.
* Firewall is configured to allow all incoming traffic on HTTP port TCP 80 and HTTPS port TCP 443.
* Apache service should be restarted every time the file /var/www/html/index.html is modified.
* A Jinja2 template file index.html.j2 is used to create the file /var/www/html/index.html with the following content:

The address of the server is: IPV4ADDRESS

IPV4ADDRESS is the IP address of the managed node.

Create a playbook /home/automation/plays/apache.yml that uses the role and runs on hosts in the **webservers** host group.

The playbook:

---

- name: task 10

hosts: webservers

become: true

roles:

- sample-apache

The role:

---

# tasks file for roles/sample-apache

- name: install apache

yum:

name: "{{ item }}"

state: latest

loop:

- httpd

- mod\_ssl

- php

- name: allow incoming http/https traffic

firewalld:

service: "{{ item }}"

immediate: true

permanent: true

state: enabled

loop:

- http

- https

- name: start and enable the apache service

service:

name: httpd

state: started

enabled: true

- name: update index.html

template:

src: index.html.j2

dest: /var/www/html/index.html

owner: root

group: root

mode: 0644

notify: restart apache

The handlers file:

---

# handlers file for roles/sample-apache

- name: restart apache

service:

name: httpd

state: restarted

**Task 11: Download Roles From Ansible Galaxy and Use Them**

Use Ansible Galaxy to download and install **geerlingguy.haproxy** role in /home/automation/plays/roles.

Create a playbook /home/automation/plays/haproxy.yml that runs on servers in the **proxy** host group and does the following:

* Use geerlingguy.haproxy role to load balance request between hosts in the **webservers** host group.
* Use **roundrobin** load balancing method.
* HAProxy backend servers should be configured for HTTP only (port 80).
* Firewall is configured to allow all incoming traffic on port TCP 80.

If your playbook works, then doing “**curl http://ansible2.hl.local/**” should return output from the web server (see task #10). Running the command again should return output from the other web server.

Install the role:

ansible-galaxy install geerlingguy.haproxy

The playbook:

---

- name: task 11

hosts: proxy

become: true

vars:

haproxy\_backend\_servers:

- name: ansible3

address: 172.16.10.203:80

- name: ansible4

address: 172.16.10.204:80

haproxy\_backend\_balance\_method: 'roundrobin'

haproxy\_backend\_mode: 'http'

roles:

- geerlingguy.haproxy

tasks:

- name: enable http traffic to proxy

firewalld:

service: http

state: enabled

immediate: true

permanent: true

Note: it may be a good idea to add firewalld installation and configuration just in case.

**Install Role from ansible-galaxy repository**

Create a file called requirements.yml in /home/ismat/ansible/roles to install two roles. The source for the first role is geerlingguy.haproxy and geerlingguy.php. Name the first haproxy-role and the second php-role. The roles should be installed in /home/ismat/ansible/roles.

Solution:

Create a file requirements.yml in /home/ismat/ansible/roles and write the following code

- name: haproxy-role

src: geerlingguy.haproxy

- name: php-role

src: geerlingguy.php

Navigate to the roles directory and run the following command

$ ansible-galaxy install -r requirements.yml -p /home/sandy/ansible/roles/

Check the roles were installed

$ ls \*

haproxy-role php-role sample-apache

**Question: 1**

* Use ansible Galaxy to download and Install the ansible-galaxy role named geerlingguy.nginx
* The requirement file should install this role.
* change the name to Nginx

**Answer:1**

**Step 1:** Type "ansible-galaxy search nginx --author geerlingguy --platforms EL" to find the geeglinguy.nginx role

Step 2: "ansible-galaxy info geerlingguy.nginx" to check more information about this role.

Step 3: Create requirement.yml file as show below:

- src: geerlingguy.nginx

version: "2.7.0"

name: Nginx

Step 4: Run this commad "ansible-galaxy install -r requirement.yml"

Step 5: Run "ansible-galaxy list" to make sure that new role has been installed sucessfully. [Note] Prior to this task you need to have ansible-galaxy in your system. Make sure you have added the correct path of roles directory in your ansible.cfg file with roles\_path =< Path of your ansible directory to have the roles >

Top of Form

**Task 5:**

- create a requirements file under roles directory your created somewhere in this exam requirements.yml to download the following roles:

  o source: https://github.com/bennojoy/nginx  
  o version: master  
  o name: nginx\_role

- download the above role **ONLY** to the roles directory.

* 

**$ cat requirements.yml**

* 1. ---
  2. - src: https://github.com/bennojoy/nginx
  3. version: master
  4. name: nginx\_role

**$ ansible-galaxy install -r requirments.yml -p ./roles/**

***- extracting nginx\_role to /etc/ansible/roles/nginx\_role***

***nginx\_role (master) was installed successfully***

Top of Form

Task 7:

- download **geerlingguy.haproxy** role using ansible galaxy  – USING Requirements file  
- create a playbook balancers.yml to apply role on balancers group  
- make sure you do required edits to create a loadbalance between webservers nodes.  
- Validate by running curl http://node3.exam1.loc twice, each time you should get a different response from one different webserver node.

* 

**$ cat requirments.yml**

* 1. - name: geerlingguy.haproxy

**$ cat balancers.yml**

* 1. ---
  2. - hosts: balancers
  3. become: yes
  4. roles:
  5. - role: geerlingguy.haproxy
  6. haproxy\_backend\_servers:
  7. - name: node4
  8. address: 192.168.20.204:80
  9. - name: node5
  10. address: 192.168.20.205:80

**you may need to run the following to open port 80 on balancers:**

**ansible -m firewalld -a "port=80/tcp permanent=yes state=enabled immediate=yes" -b balancers**

Question 6:

Task 6:  Create ansible role **webconfig** to do the following:

- enable and start **httpd\_service** variable  
- allow **http\_port** on firewall  
- deploy **index.html.j2** to index\_path variable  
- **httpd\_service** must be restart each time **index.html** get edited

With in **webconfig** role, configure the following variables:  
**httpd\_service** = httpd  
**http\_port** = 443 TCP  
**index\_path** = /var/www/   
  
With in webconfig role create  
template **index.html.j2** that contains   
       Welcome to Webserver <node\_inventoy\_name> on <node\_ip>   
  
Then create **webconfig.yml** to run on **webservers** nodes to:  
  - use webconfig role created above and make sure variables  
**http\_port**and **index\_path** values **MUST** be **changed** to:  
**http\_port**= 80 TCP  
**index\_path** = /var/www/html

- deploy index.html.j2 to the new index\_path and make sure the **owner/group** should be apache:apaech and **mode** 0644

Top of Form

Task 4:

  - Create a role motd and a playbook motd.yml and apply it to all managed hosts to do edit the /etc/motd file on managed hosts as the following:  
    o if the managed host is in dev environments the content should be:  
         Welcome to Dev environment this is <inventory\_hostname>  
    o if the managed host is in qual environment the content should be:  
          Welcome to Qual environment this is <inventory\_hostname>  
    o if the managed host is in prod the content should be:  
          Welcome to Production environment this is <inventory\_hostname>

* 

**$ cat motd.yml**

* 1. ---
  2. - hosts: all
  3. become: yes
  4. roles:
  5. - motd

**Question: 3**

* Create a playbook of any name, and that playbook should do as shown below:
* The playbook runs over all the managed hosts and uses the time sync role ( Red Hat system role)
* This role should be able to change the time server to nl.pool.ntp.org .

**Answer: 3**

Step 1: you must have RHEL system roles installed and settled all the paths.

Step 2: Copy the roles into your ansible directory or change the path of roles\_path in ansible.cfg file.

Step 3: Copy /usr/share/doc/rhel-sytem-roles/timesync/example-timesync-playbook.yml to home directory.

Step 4: Edit the file has shome below:

---

- hosts: all

vars:

timesync\_ntp\_servers:

- hostname: nl.pool.ntp.org

ibrust: yes

roles:

- rhel-system-roles.timesync

Step 5: Run shown above yaml file to settle the ntp server.

question 4: Create Role including template

Create a role called sample-apache in /home/ismat/ansible/roles that enables and starts httpd, enables and starts the firewall; and allows the webserver service. Create a template called index.html.j2 which creates and serves a message from /var/www/html/index.html. Whenever the content of the file changes, restart the webserver service.  
**Welcome to [FQDN] on [IP]**  
Replace the FQDN with the fully qualified domain name and IP with the ip address of the node using ansible facts. Lastly, create a playbook in /home/ismat/ansible named apache.yml and use the role to serve the index file on webserver hosts.

Solution:

Navigate to /home/ismat/ansible/roles directory and run command

$ ansible-galaxy init sample-apache

vim sample-apache/tasks/main.yml

---

# tasks file for sample-apache

- name: Enable httpd

service:

name: httpd

state: started

enabled: yes

- name: Enable firewalld

service:

name: firewalld

state: started

enabled: yes

- name: Allow webserver service

firewalld:

service: http

state: enabled

permanent: yes

immediate: yes

- name: Create index file from index.html.j2

template:

src: index.html.j2

dest: /var/www/html/index.html

notify:

- restart\_webservers

vim sample-apache/templates/index.html.j2

Welcome to {{ ansible\_fqdn }} on {{ ansible\_default\_ipv4.address }}

**Tips:**There is a chance you can’t remember ansible\_fqdn or ansible\_default\_ipv4.address these kinds of facts. You can run adhoc command of the setup module to get the whole remote machine facts and save them to a file. **ansible node2.example.com -m setup > facts.json.** Then vim facts.json file and find the facts variables. It is extremely helpful if you are afraid of a spelling mistake.

vim sample-apache/handlers/main.yml

---

# handlers file for sample-apache

- name: restart\_webservers

service:

name: httpd

state: restarted

Navigate back to your ansible working directory /home/ismat/ansible and write in apache.yml ( vim apache.yml)

---

- name: Install apachec from apache-role

hosts: webservers

roles:

- name: sample-apache

...

Top of Form

**Task 11: work with system roles**

- Create a playbook **timesync.yml**to use system roles to do the following on all managed nodes:

   o Use the **currently configured** NTP client to sync time with

   o Use both **0.pool.ntp.org**, **1.pool.ntp.org** as NTP servers

   o Enable **iburst**

   o Time zone is **UTC**

* 

**# you maybe need to enable extras repository - in the exam you don't need this step**

**$ sudo yum-config-manager --enablerepo=extras**

**# then install rhel-system-roles**

**$ sudo yum install rhel-system-roles**

**$ cat timesync.yml**

* 1. - hosts: servers
  2. become: yes
  3. vars:
  4. timesync\_ntp\_servers:
  5. - hostname: 0.rhel.pool.ntp.org
  6. iburst: yes
  7. timezone: UTC
  8. roles:
  9. - rhel-system-roles.timesync
  10. tasks:
  11. - name: Set timezone
  12. timezone:
  13. name: "{{ timezone }}"

**Question: 2**

Create a role called **apache** and store it in your ansible directory “roles”. This role should satisfy the requirements below:

* The **httpd**, **mod\_ssl** and **php** packages are installed. Apache service is running and enabled on boot.
* The firewall is configured to allow all incoming traffic on HTTP port TCP 80 and HTTPS port TCP 443.
* Apache service should be restarted every time the file /var/www/html/index.html is modified.
* A Jinja2 template file index.html.j2 is used to create the file /var/www/html/index.html with the following content:

The address of the server is: IPV4ADDRESS

IPV4ADDRESS is the IP address of the managed node.

Create a playbook /home/automation/plays/apache.yml that uses the role and runs on hosts in the **webservers** host grou **Answer: 2**

Step 1: Create the empty role by "ansible-galaxy init apache" run this command in your roles directory, for it is /home/ali/ansible/roles

Step 2: Create the handler file roles/apache/handlers/main.yml.

---

- name: restart\_apache

service:

name: httpd

state: restarted

enabled: yes

Step 3: Create the main task file roles/apache/tasks/main.yml

---

- name: Ensure the packages httpd, mod\_ssl and php are installed

yum:

name: "{{ item }}"

state: latest

loop:

- httpd

- mod\_ssl

- php

- name: Ensure that the service httpd is enabled

service:

name: httpd

state: started

enabled: yes

- name: Ensure the firewall ports 80 and 443 are open

firewalld:

service: "{{ item }}"

permanent: yes

immediate: yes

state: enabled

loop:

- http

- https

- name: Create index.html from template

template:

src: index.html.j2

dest: /var/www/html/index.html

notify: restart\_apache

...

Step 4: Create a template file index.html.j2.

The address of the server is: {{ ansible\_facts['default\_ipv4']['address']}}

Step 5: Create a general apache file to run the role.

---

- hosts: webservers

become: yes

roles:

- apache

**Question: 3**

* Create a playbook of any name, and that playbook should do as shown below:
* The playbook runs over all the managed hosts and uses the time sync role ( Red Hat system role)
* This role should be able to change the time server to nl.pool.ntp.org .

**Answer: 3**

Step 1: you must have RHEL system roles installed and settled all the paths.

Step 2: Copy the roles into your ansible directory or change the path of roles\_path in ansible.cfg file.

Step 3: Copy /usr/share/doc/rhel-sytem-roles/timesync/example-timesync-playbook.yml to home directory.

Step 4: Edit the file has shome below:

---

- hosts: all

vars:

timesync\_ntp\_servers:

- hostname: nl.pool.ntp.org

ibrust: yes

roles:

- rhel-system-roles.timesync

Step 5: Run shown above yaml file to settle the ntp server.

**Task 12: Security**

Create a playbook /home/automation/plays/selinux.yml that runs on hosts in the **webservers** host group and does the following:

* Uses the selinux **RHEL system role**.
* Enables **httpd\_can\_network\_connect** SELinux boolean.
* The change must survive system reboot.
* Install roles:
* yum install rhel-system-roles
* Since the installed roles are not in the path, the ansible.cfg has to be modified:
* [defaults]
* roles\_path = /home/automation/plays/roles:/usr/share/ansible/roles
* [...]
* The playbook:
* ---
* - name: task 12
* hosts: webservers
* become: true
* vars:
* - selinux\_booleans:
* - name: httpd\_can\_network\_connect
* state: on
* persistent: true
* roles:
* - rhel-system-roles.selinux