Red Hat Enterprise Linux Automation with Ansible

Lab: Implementing Task Control

Performance Checklist

In this lab, you will install the Apache web server and secure it using `mod_ssl`. You will use conditions, handlers, and task failure handling in your playbook to deploy the environment.

Outcomes

You should be able to define conditionals in Ansible Playbooks, set up loops that iterate over elements, define handlers in playbooks, and handle task errors.

Log in as the `student` user on `workstation` and run `lab control-review start`. This script ensures that the managed host, `serverb`, is reachable on the network. It also ensures that the correct Ansible configuration file and inventory are installed on the control node.

```
[student@workstation ~]$ lab control-review start
```

Procedure 4.4. Instructions

On `workstation`, change to the `/home/student/control-review` project directory.

```
[student@workstation ~]$ cd ~/control-review
[student@workstation control-review]$
```

HIDE SOLUTION

The project directory contains a partially completed playbook, `playbook.yml`. Using a text editor, add a task that uses the `fail` module under the `#Fail Fast Message` comment. Be sure to provide an appropriate name for the task. This task should only be executed when the remote system does not meet the minimum requirements.

The minimum requirements for the remote host are listed below:

Has at least the amount of RAM specified by the `min_ram_mb` variable. The `min_ram_mb` variable is defined in the `vars.yml` file and has a value of `256`.

Is running Red Hat Enterprise Linux.

The completed task matches:

```
tasks:
  #Fail Fast Message
  - name: Show Failed System Requirements Message
    fail:
      msg: "The {{ inventory_hostname }} did not meet minimum reqs."
    when: >
      ansible_memtotal_mb < min_ram_mb or
      ansible_distribution != "RedHat"
```

HIDE SOLUTION

Add a single task to the playbook under the `#Install all Packages` comment to install the latest version of any missing packages. Required packages are specified by the `packages` variable, which is defined in the `vars.yml` file.

The task name should be `Ensure required packages are present`.

The completed task matches:

```
  #Install all Packages
  - name: Ensure required packages are present
    yum:
      name: "{{ packages }}"
      state: latest
```

HIDE SOLUTION

Add a single task to the playbook under the `#Enable and start services` comment to start all services. All services specified by the `services` variable, which is defined in the `vars.yml` file, should be started and enabled. Be sure to provide an appropriate name for the task.

The completed task matches:

```
#Enable and start services
- name: Ensure services are started and enabled
  service:
    name: "{{ item }}"
    state: started
    enabled: yes
  loop: "{{ services }}"
```

HIDE SOLUTION

Add a task block to the playbook under the `#Block of config tasks` comment. This block contains two tasks:

A task to ensure the directory specified by the `ssl_cert_dir` variable exists on the remote host. This directory stores the web server's certificates.

A task to copy all files specified by the `web_config_files` variable to the remote host. Examine the structure of the `web_config_files` variable in the `vars.yml` file. Configure the task to copy each file to the correct destination on the remote host.

This task should trigger the `restart web service` handler if any of these files are changed on the remote server.

Additionally, a debug task is executed if either of the two tasks above fail. In this case, the task prints the message: `One or more of the configuration changes failed, but the web service is still active.`

Be sure to provide an appropriate name for all tasks.

The completed task block matches below:

```
#Block of config tasks
- name: Setting up the SSL cert directory and config files
  block:
    - name: Create SSL cert directory
      file:
        path: "{{ ssl_cert_dir }}"
        state: directory

    - name: Copy Config Files
      copy:
        src: "{{ item.src }}"
        dest: "{{ item.dest }}"
      loop: "{{ web_config_files }}"
      notify: restart web service

  rescue:
    - name: Configuration Error Message
      debug:
        msg: >
          One or more of the configuration
          changes failed, but the web service
          is still active.
```

HIDE SOLUTION

The playbook configures the remote host to listen for standard HTTPS requests. Add a single task to the playbook under the `#Configure the firewall` comment to configure `firewalld`.

This task should ensure that the remote host allows standard HTTP and HTTPS connections. These configuration changes should be effective immediately and persist after a system reboot. Be sure to provide an appropriate name for the task.

The completed task matches:

```
  #Configure the firewall
  - name: ensure web server ports are open
    firewalld:
      service: "{{ item }}"
      immediate: true
      permanent: true
      state: enabled
    loop:
      - http
      - https
```

HIDE SOLUTION

Define the `restart web service` handler.

When triggered, this task should restart the web service defined by the `web_service` variable, defined in the `vars.yml` file.

A `handlers` section is added to the end of the playbook:

```
handlers:
  - name: restart web service
    service:
      name: "{{ web_service }}"
      state: restarted
```

The completed playbook contains:

```
---
- name: Playbook Control Lab
  hosts: webservers
  vars_files: vars.yml
  tasks:
    #Fail Fast Message
    - name: Show Failed System Requirements Message
      fail:
        msg: "The {{ inventory_hostname }} did not meet minimum reqs."
      when: >
        ansible_memtotal_mb < min_ram_mb or
        ansible_distribution != "RedHat"

    #Install all Packages
    - name: Ensure required packages are present
      yum:
        name: "{{ packages }}"
        state: latest

    #Enable and start services
    - name: Ensure services are started and enabled
      service:
        name: "{{ item }}"
        state: started
        enabled: yes
      loop: "{{ services }}"

    #Block of config tasks
    - name: Setting up the SSL cert directory and config files
      block:
        - name: Create SSL cert directory
          file:
            path: "{{ ssl_cert_dir }}"
            state: directory

        - name: Copy Config Files
          copy:
            src: "{{ item.src }}"
            dest: "{{ item.dest }}"
          loop: "{{ web_config_files }}"
          notify: restart web service

      rescue:
        - name: Configuration Error Message
          debug:
            msg: >
              One or more of the configuration
              changes failed, but the web service
              is still active.

    #Configure the firewall
    - name: ensure web server ports are open
      firewalld:
        service: "{{ item }}"
        immediate: true
        permanent: true
        state: enabled
      loop:
        - http
        - https

  #Add handlers
  handlers:
    - name: restart web service
      service:
        name: "{{ web_service }}"
        state: restarted
```

HIDE SOLUTION

From the project directory, `~/control-review`, run the `playbook.yml` playbook. The playbook should execute without errors, and trigger the execution of the handler task.

```
[student@workstation control-review]$ ansible-playbook playbook.yml

PLAY [Playbook Control Lab] ***************************************************

TASK [Gathering Facts] *********************************************************
ok: [serverb.lab.example.com]

TASK [Show Failed System Requirements Message] *********************************
skipping: [serverb.lab.example.com]

TASK [Ensure required packages are present] ************************************
changed: [serverb.lab.example.com]

TASK [Ensure services are started and enabled] *********************************
changed: [serverb.lab.example.com] => (item=httpd)
ok: [serverb.lab.example.com] => (item=firewalld)

TASK [Create SSL cert directory] ***********************************************
changed: [serverb.lab.example.com]

TASK [Copy Config Files] *******************************************************
changed: [serverb.lab.example.com] => (item={'src': 'server.key', 'dest': '/etc/httpd/conf.d/ssl'})
changed: [serverb.lab.example.com] => (item={'src': 'server.crt', 'dest': '/etc/httpd/conf.d/ssl'})
changed: [serverb.lab.example.com] => (item={'src': 'ssl.conf', 'dest': '/etc/httpd/conf.d'})
changed: [serverb.lab.example.com] => (item={'src': 'index.html', 'dest': '/var/www/html'})

TASK [ensure web server ports are open] ****************************************
changed: [serverb.lab.example.com] => (item=http)
changed: [serverb.lab.example.com] => (item=https)

RUNNING HANDLER [restart web service] ******************************************
changed: [serverb.lab.example.com]

PLAY RECAP *********************************************************************
serverb.lab.example.com    : ok=7    changed=6    unreachable=0    failed=0
```

HIDE SOLUTION

Verify that the web server now responds to HTTPS requests, using the self-signed custom certificate to encrypt the connection. The web server response should match the string `Configured for both HTTP and HTTPS.`

```
[student@workstation control-review]$ curl -k -vvv https://serverb.lab.example.com
* About to connect() to serverb.lab.example.com port 443 (#0)
*   Trying 172.25.250.11...
* Connected to serverb.lab.example.com (172.25.250.11) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* skipping SSL peer certificate verification
* SSL connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate:
...output omitted...
*       start date: Nov 13 15:52:18 2018 GMT
*       expire date: Aug 09 15:52:18 2021 GMT
*       common name: serverb.lab.example.com
...output omitted...
< Accept-Ranges: bytes
< Content-Length: 36
< Content-Type: text/html; charset=UTF-8
<
Configured for both HTTP and HTTPS.
* Connection #0 to host serverb.lab.example.com left intact
```

HIDE SOLUTION

Evaluation

Run the `lab control-review grade` command on `workstation` to confirm success on this exercise. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab control-review grade
```

Finish

Run the `lab control-review finish` command to clean up after the lab.

```
[student@workstation ~]$ lab control-review finish
```

This concludes the lab.

← PREVIOUS (/ROL/APP/COURSES/RH294-8.4/PAGES/CH04S06) → NEXT (/ROL/APP/COURSES/RH294-8.4/PAGES/CH04S08)

Privacy Policy (http://s.bl-1.com/h/cZrgWbQn?url=https://www.redhat.com/en/about/privacy-policy?extIdCarryOver=true&sc_cid=701f2000001D8QoAAK)

Terms of Use (https://www.redhat.com/en/about/terms-use)

Release Notes (https://learn.redhat.com/t5/Red-Hat-Learning-Subscription/Red-Hat-Learning-Subscription-Release-Notes/ba-p/22952)

Red Hat Training Policies (http://s.bl-1.com/h/cZrb2DXG?url=https://www.redhat.com/en/about/red-hat-training-policies)

All policies and guidelines (https://www.redhat.com/en/about/all-policies-guidelines)

Cookie Preferences