Red Hat Enterprise Linux Automation with Ansible

FEEDBACK TRANSLATIONS CERTIFICATE OF ATTENDANCE

Lab: Simplifying Playbooks with Roles ☆

Performance Checklist

In this lab, you will create Ansible roles that use variables, files, templates, tasks, and handlers.

Outcomes

You should be able to:

- Create Ansible roles that use variables, files, templates, tasks, and handlers to configure a development web server.
- Use a role that is hosted in a remote repository in a playbook.
- Use a Red Hat Enterprise Linux system role in a playbook.

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the `lab role-review start` command. The script creates the working directory, `/home/student/role-review`, and populates it with an Ansible configuration file, host inventory, and other lab files.

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

Procedure 7.5. Instructions

Your organization must provide a single web server to host development code for all web developers. You are tasked with writing a playbook to configure this development web server.

The development web server must satisfy several requirements:

- The development server configuration matches the production server configuration. The production server is configured using an Ansible role, developed by the organization's infrastructure team.
- Each developer is given a directory on the development server to host code and content. Each developer's content is accessed using an assigned, nonstandard port.
- SELinux is set to enforcing and targeted.

Your playbook will:

- Use a role to configure directories and ports for each developer on the web server. You must write this role.

This role has a dependency on a role written by the organization to configure Apache. You should define the dependency using version `v1.4` of the organizational role. The URL of the dependency's repository is: `git@workstation.lab.example.com:infra/apache`

- Use the `rhel-system-roles.selinux` role to configure SELinux for the nonstandard HTTP ports used by your web server. You will be provided with a `selinux.yml` variable file that can be installed as a `group_vars` file to pass the correct settings to the role.

1. Change to the `/home/student/role-review` working directory.

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

HIDE SOLUTION

2. Create a playbook named `web_dev_server.yml` with a single play named `Configure Dev Web Server`. Configure the play to target the host group `dev_webserver`. Do not add any roles or tasks to the play yet.

   Ensure that the play forces handlers to execute, because you may encounter an error while developing the playbook.

   Once complete, the `/home/student/role-review/web_dev_server.yml` playbook contains:

```
---
- name: Configure Dev Web Server
  hosts: dev_webserver
  force_handlers: yes
```

   HIDE SOLUTION

3. Check the syntax of the playbook. Run the playbook. The syntax check should pass and the playbook should run successfully.

```
[student@workstation role-review]$ ansible-playbook \
> --syntax-check web_dev_server.yml

playbook: web_dev_server.yml
[student@workstation role-review]$ ansible-playbook web_dev_server.yml
PLAY [Configure Dev Web Server] ************************************************

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

PLAY RECAP *********************************************************************
servera.lab.example.com    : ok=1    changed=0    unreachable=0    failed=0
```

   HIDE SOLUTION

4. Make sure that playbook's role dependencies are installed.

   The `apache.developer_configs` role that you will create depends on the `infra.apache` role. Create a `roles/requirements.yml` file. It should install the role from the Git repository at `git@workstation.lab.example.com:infra/apache`, use version `v1.4`, and name it `infra.apache` locally. You can assume that your SSH keys are configured to allow you to get roles from that repository automatically. Install the role with the `ansible-galaxy` command.

   In addition, install the rhel-system-roles package if not present.

   4.1. Create a `roles` subdirectory for the playbook project.

```
[student@workstation role-review]$ mkdir -v roles
mkdir: created directory 'roles'
```

4.2. Create a `roles/requirements.yml` file and add an entry for the `infra.apache` role. Use version `v1.4` from the role's git repository.

Once complete, the `roles/requirements.yml` file contains:

```
- name: infra.apache
  src: git@workstation.lab.example.com:infra/apache
  scm: git
  version: v1.4
```

4.3. Install the project dependencies.

```
[student@workstation role-review]$ ansible-galaxy install \
> -r roles/requirements.yml -p roles
- extracting infra.apache to /home/student/role-review/roles/infra.apache
- infra.apache (v1.4) was installed successfully
```

4.4. Install the RHEL System Roles package if not present. This was installed during an earlier exercise.

```
[student@workstation role-review]$ sudo yum install rhel-system-roles
```

HIDE SOLUTION

5. Initialize a new role named `apache.developer_configs` in the `roles` subdirectory.

Add the `infra.apache` role as a dependency for the new role, using the same information for name, source, version, and version control system as the `roles/requirements.yml` file.

The `developer_tasks.yml` file in the project directory contains tasks for the role. Move this file to the correct location to be the tasks file for this role, and replace the existing file in that location.

The `developer.conf.j2` file in the project directory is a Jinja2 template used by the tasks file. Move it to the correct location for template files used by this role.

5.1. Use the `ansible-galaxy init` to create a role skeleton for the `apache.developer_configs` role.

```
[student@workstation role-review]$ cd roles
[student@workstation roles]$ ansible-galaxy init apache.developer_configs
- apache.developer_configs was created successfully
[student@workstation roles]$ cd ..
[student@workstation role-review]$
```

5.2. Update the `roles/apache.developer_configs/meta/main.yml` file of the `apache.developer_configs` role to reflect a dependency on the `infra.apache` role.

After editing, the `dependencies` variable is defined as follows:

```
dependencies:
  - name: infra.apache
    src: git@workstation.lab.example.com:infra/apache
    scm: git
    version: v1.4
```

5.3. Overwrite the role's `tasks/main.yml` file with the `developer_tasks.yml` file.

```
[student@workstation role-review]$ mv -v developer_tasks.yml \
> roles/apache.developer_configs/tasks/main.yml
renamed 'developer_tasks.yml' -> 'roles/apache.developer_configs/tasks/main.yml'
```

5.4. Place the `developer.conf.j2` file in the role's `templates` directory.

```
[student@workstation role-review]$ mv -v developer.conf.j2 \
> roles/apache.developer_configs/templates/
renamed 'developer.conf.j2' -> 'roles/apache.developer_configs/templates/developer.conf.j2'
```

HIDE SOLUTION

6. The `apache.developer_configs` role will process a list of users defined in a variable named `web_developers`. The `web_developers.yml` file in the project directory defines the `web_developers` user list variable. Review this file and use it to define the `web_developers` variable for the development web server host group.

6.1. Review the `web_developers.yml` file.

```
---
web_developers:
  - username: jdoe
    name: John Doe
    user_port: 9081
  - username: jdoe2
    name: Jane Doe
    user_port: 9082
```

A `name`, `username`, `user_port` is defined for each web developer.

6.2. Place the `web_developers.yml` in the `group_vars/dev_webserver` subdirectory.

```
[student@workstation role-review]$ mkdir -pv group_vars/dev_webserver
mkdir: created directory 'group_vars'
mkdir: created directory 'group_vars/dev_webserver'
[student@workstation role-review]$ mv -v web_developers.yml \
> group_vars/dev_webserver/
renamed 'web_developers.yml' -> 'group_vars/dev_webserver/web_developers.yml'
```

HIDE SOLUTION

7. Add the role `apache.developer_configs` to the play in the `web_dev_server.yml` playbook.

The edited playbook:

```
---
- name: Configure Dev Web Server
  hosts: dev_webserver
  force_handlers: yes
  roles:
    - apache.developer_configs
```

HIDE SOLUTION

8. Check the syntax of the playbook. Run the playbook. The syntax check should pass, but the playbook should fail when the `infra.apache` role attempts to restart Apache HTTPD.

```
[student@workstation role-review]$ ansible-playbook \
> --syntax-check web_dev_server.yml

playbook: web_dev_server.yml
[student@workstation role-review]$ ansible-playbook web_dev_server.yml

PLAY [Configure Dev Web Server] ************************************************

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

...output omitted...

TASK [infra.apache : Install a skeleton index.html] ****************************
skipping: [servera.lab.example.com]

TASK [apache.developer_configs : Create user accounts] *************************
changed: [servera.lab.example.com] => (item={u'username': u'jdoe', u'user_port': 9081, u'name': u'John Doe'})
changed: [servera.lab.example.com] => (item={u'username': u'jdoe2', u'user_port': 9082, u'name': u'Jane Doe'})

...output omitted...

RUNNING HANDLER [infra.apache : restart firewalld] *****************************
changed: [servera.lab.example.com]

RUNNING HANDLER [infra.apache : restart apache] ********************************
fatal: [servera.lab.example.com]: FAILED! => {"changed": false, "msg": "Unable to restart service httpd: Job for
httpd.service failed because the control process exited with error code. See \"systemctl status httpd.service\"
and \"journalctl -xe\" for details.\n"}

NO MORE HOSTS LEFT *************************************************************
  to retry, use: --limit @/home/student/role-review/web_dev_server.retry

PLAY RECAP *********************************************************************
servera.lab.example.com    : ok=13   changed=11   unreachable=0    failed=1
skipped=1    rescued=0    ignored=0
```

An error occurs when the `httpd` service is restarted. The `httpd` service daemon cannot bind to the non-standard HTTP ports, due to the SELinux context on those ports.

HIDE SOLUTION

9. Apache HTTPD failed to restart in the preceding step because the network ports it uses for your developers are labeled with the wrong SELinux contexts. You have been provided with a variable file, `selinux.yml`, which can be used with the `rhel-system-roles.selinux` role to fix the issue.

   Create a `pre_tasks` section for your play in the `web_dev_server.yml` playbook. In that section, use a task to include the `rhel-system-roles.selinux` role in a `block/rescue` structure so that it is properly applied. Review the lecture or the documentation for this role to see how to do this.

   Inspect the `selinux.yml` file. Move it to the correct location so that its variables are set for the `dev_webserver` host group.

   9.1. The `pre_tasks` section can be added to the end of the play in the `web_dev_server.yml` playbook.

   You can look at the block in `/usr/share/doc/rhel-system-roles/selinux/example-selinux-playbook.yml` for a basic outline of how to apply the role. Replace the complex `shell` and `wait_for_connection` logic with the `reboot` module.

   The `pre_tasks` section should contain:

```
  pre_tasks:
    - name: Check SELinux configuration
      block:
        - include_role:
            name: rhel-system-roles.selinux
      rescue:
        # Fail if failed for a different reason than selinux_reboot_required.
        - name: Check for general failure
          fail:
            msg: "SELinux role failed."
          when: not selinux_reboot_required

        - name: Restart managed host
          reboot:
            msg: "Ansible rebooting system for updates."

        - name: Reapply SELinux role to complete changes
          include_role:
            name: rhel-system-roles.selinux
```

9.2. The `selinux.yml` file contains variable definitions for the `rhel-system-roles.selinux` role. Use the file to define variables for the play's host group.

```
[student@workstation role-review]$ cat selinux.yml
---
# variables used by rhel-system-roles.selinux

selinux_policy: targeted
selinux_state: enforcing

selinux_ports:
  - ports:
      - "9081"
      - "9082"
    proto: 'tcp'
    setype: 'http_port_t'
    state: 'present'

[student@workstation role-review]$ mv -v selinux.yml \
> group_vars/dev_webserver/
renamed 'selinux.yml' -> 'group_vars/dev_webserver/selinux.yml'
```

HIDE SOLUTION

0. Verify the final `web_dev_server.yml` playbook and run a syntax check. The syntax check should pass.

> **NOTE**
>
> Whether `pre_tasks` is at the end of the play or in the "correct" position in terms of execution order in the playbook file does not matter to `ansible-playbook`. It will still run the play's tasks in the correct order.

Validate that the `web_dev_server.yml` playbook passes a syntax check.

The final `web_dev_server.yml` playbook should read as follows:

```
---
- name: Configure Dev Web Server
  hosts: dev_webserver
  force_handlers: yes
  roles:
    - apache.developer_configs
  pre_tasks:
    - name: Check SELinux configuration
      block:
        - include_role:
            name: rhel-system-roles.selinux
      rescue:
        # Fail if failed for a different reason than selinux_reboot_required.
        - name: Check for general failure
          fail:
            msg: "SELinux role failed."
          when: not selinux_reboot_required

        - name: Restart managed host
          reboot:
            msg: "Ansible rebooting system for updates."

        - name: Reapply SELinux role to complete changes
          include_role:
            name: rhel-system-roles.selinux
```

```
[student@workstation role-review]$ ansible-playbook \
> --syntax-check web_dev_server.yml

playbook: web_dev_server.yml
```

HIDE SOLUTION

11. Run the playbook. It should succeed.

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

PLAY [Configure Dev Web Server] ************************************************

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

TASK [include_role : rhel-system-roles.selinux] ********************************

TASK [rhel-system-roles.selinux : Install SELinux python3 tools] *************
ok: [servera.lab.example.com]

...output omitted...

TASK [infra.apache : Apache Service is started] ********************************
changed: [servera.lab.example.com]

...output omitted...

TASK [apache.developer_configs : Copy Per-Developer Config files] ************
ok: [servera.lab.example.com] => (item={'username': 'jdoe', 'name': 'John Doe', 'user_port': 9081})
ok: [servera.lab.example.com] => (item={'username': 'jdoe2', 'name': 'Jane Doe', 'user_port': 9082})

PLAY RECAP *********************************************************************
servera.lab.example.com    : ok=19   changed=3    unreachable=0    failed=0
skipped=14   rescued=0    ignored=0
```

HIDE SOLUTION

2. Test the configuration of the development web server. Verify that all endpoints are accessible and serving each developer's content.

```
[student@workstation role-review]$ curl servera
This is the production server on servera.lab.example.com
[student@workstation role-review]$ curl servera:9081
This is index.html for user: John Doe (jdoe)
[student@workstation role-review]$ curl servera:9082
This is index.html for user: Jane Doe (jdoe2)
[student@workstation role-review]$
```

HIDE SOLUTION

Evaluation

Grade your work by running the `lab role-review grade` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

Finish

On `workstation`, run the `lab role-review finish` script to clean up this exercise.

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

This concludes the lab.

← PREVIOUS (/ROL/APP/COURSES/RH294-8.4/PAGES/CH07S10) → NEXT (/ROL/APP/COURSES/RH294-8.4/PAGES/CH07S12)