Red Hat Enterprise Linux Automation with Ansible

FEEDBACK TRANSLATIONS CERTIFICATE OF ATTENDANCE

Reusing Content with System Roles ☆

Objectives

After completing this section, you should be able to write playbooks that take advantage of Red Hat Enterprise Linux System Roles to perform standard operations.

Red Hat Enterprise Linux System Roles

Beginning with Red Hat Enterprise Linux 7.4, a number of Ansible roles have been provided with the operating system as part of the rhel-system-roles package. In Red Hat Enterprise Linux 8 the package is available in the AppStream channel. A brief description of each role:

Table 7.2. RHEL System Roles

Name	State	Role Description
`rhel-system-roles.kdump`	Fully Supported	Configures the `kdump` crash recovery service.
`rhel-system-roles.network`	Fully Supported	Configures network interfaces.
`rhel-system-roles.selinux`	Fully Supported	Configures and manages SELinux customization, including SELinux mode, file and port contexts, Boolean settings, and SELinux users.
`rhel-system-roles.timesync`	Fully Supported	Configures time synchronization using Network Time Protocol or Precision Time Protocol.
`rhel-system-roles.postfix`	Technology Preview	Configures each host as a Mail Transfer Agent using the Postfix service.
`rhel-system-roles.firewall`	In Development	Configures a host's firewall.
`rhel-system-roles.tuned`	In Development	Configures the `tuned` service to tune system performance.

System roles aim to standardize the configuration of Red Hat Enterprise Linux subsystems across multiple versions. Use system roles to configure any Red Hat Enterprise Linux, version 6.10 and onward.

Simplified Configuration Management

As an example, the recommended time synchronization service for Red Hat Enterprise Linux 7 is the `chronyd` service. In Red Hat Enterprise Linux 6 however, the recommended service is the `ntpd` service. In an environment with a mixture of Red Hat Enterprise Linux 6 and 7 hosts, an administrator must manage the configuration files for both services.

With RHEL System Roles, administrators no longer need to maintain configuration files for both services. Administrators can use `rhel-system-roles.timesync` role to configure time synchronization for both Red Hat Enterprise Linux 6 and 7 hosts. A simplified YAML file containing role variables defines the configuration of time synchronization for both types of hosts.

Support for RHEL System Roles

RHEL System Roles are derived from the open source Linux System Roles project, found on Ansible Galaxy. Unlike Linux System Roles, RHEL System Roles are supported by Red Hat as part of a standard Red Hat Enterprise Linux subscription. RHEL System Roles have the same life cycle support benefits that come with a Red Hat Enterprise Linux subscription.

Every system role is tested and stable. The `Fully Supported` system roles also have stable interfaces. For any `Fully Supported` system role, Red Hat will endeavour to ensure that role variables are unchanged in future versions. Playbook refactoring due to system role changes should be minimal.

The `Technology Preview` system roles may utilize different role variables in future versions. Integration testing is recommended for playbooks that incorporate any `Technology Preview` role. Playbooks may require refactoring if role variables change in a future version of the role.

Other roles are in development in the upstream Linux System Roles project, but are not yet available through a RHEL subscription. These roles are available through Ansible Galaxy.

Installing RHEL System Roles

The RHEL System Roles are provided by the rhel-system-roles package, which is available in the `AppStream` channel. Install this package on the Ansible control node.

Use the following procedure to install the rhel-system-roles package. The procedure assumes the control node is registered to a Red Hat Enterprise Linux subscription and that Ansible is installed. See the section on *Installing Ansible* for more information.

1. Install RHEL System Roles.

```
[root@host ~]# yum install rhel-system-roles
```

After installation, the RHEL System roles are located in the `/usr/share/ansible/roles` directory:

```
[root@host ~]# ls -l /usr/share/ansible/roles/
total 20
...output omitted... linux-system-roles.kdump -> rhel-system-roles.kdump
...output omitted... linux-system-roles.network -> rhel-system-roles.network
...output omitted... linux-system-roles.postfix -> rhel-system-roles.postfix
...output omitted... linux-system-roles.selinux -> rhel-system-roles.selinux
...output omitted... linux-system-roles.timesync -> rhel-system-roles.timesync
...output omitted... rhel-system-roles.kdump
...output omitted... rhel-system-roles.network
...output omitted... rhel-system-roles.postfix
...output omitted... rhel-system-roles.selinux
...output omitted... rhel-system-roles.timesync
```

The corresponding upstream name of each role is linked to the RHEL System Role. This allows a role to be referenced in a playbook by either name.

The default `roles_path` on Red Hat Enterprise Linux includes `/usr/share/ansible/roles` in the path, so Ansible should automatically find those roles when referenced by a playbook.

> **NOTE**
>
> Ansible might not find the system roles if `roles_path` has been overridden in the current Ansible configuration file, if the environment variable `ANSIBLE_ROLES_PATH` is set, or if there is another role of the same name in a directory listed earlier in `roles_path`.

Accessing Documentation for RHEL System Roles

After installation, documentation for the RHEL System Roles is found in the `/usr/share/doc/rhel-system-roles-<version>/` directory. Documentation is organized into subdirectories by subsystem:

```
[root@host ~]# ls -l /usr/share/doc/rhel-system-roles/
total 4
drwxr-xr-x. ...output omitted... kdump
drwxr-xr-x. ...output omitted... network
drwxr-xr-x. ...output omitted... postfix
drwxr-xr-x. ...output omitted... selinux
drwxr-xr-x. ...output omitted... timesync
```

Each role's documentation directory contains a `README.md` file. The `README.md` file contains a description of the role, along with role usage information.

The `README.md` file also describes role variables that affect the behavior of the role. Often the `README.md` file contains a playbook snippet that demonstrates variable settings for a common configuration scenario.

Some role documentation directories contain example playbooks. When using a role for the first time, review any additional example playbooks in the documentation directory.

Role documentation for RHEL System Roles matches the documentation for Linux System Roles. Use a web browser to access role documentation for the upstream roles at the Ansible Galaxy site, `https://galaxy.ansible.com`.

Time Synchronization Role Example

Suppose you need to configure NTP time synchronization on your servers. You could write automation yourself to perform each of the necessary tasks. But RHEL System Roles includes a role that can do this, `rhel-system-roles.timesync`.

The role is documented in its `README.md` in the `/usr/share/doc/rhel-system-roles/timesync` directory. The file describes all the variables that affect the role's behavior and contains three playbook snippets illustrating different time synchronization configurations.

To manually configure NTP servers, the role has a variable named `timesync_ntp_servers`. It takes a list of NTP servers to use. Each item in the list is made up of one or more attributes. The two key attributes are:

Table 7.3. timesync_ntp_servers attributes

Attribute	Purpose
`hostname`	The hostname of an NTP server with which to synchronize.
`iburst`	A Boolean that enables or disables fast initial synchronization. Defaults to `no` in the role, you should normally set this to `yes`.

Given this information, the following example is a play that uses the `rhel-system-roles.timesync` role to configure managed hosts to get time from three NTP servers using fast initial synchronization. In addition, a task has been added that uses the `timezone` module to set the hosts' time zone to UTC.

```
- name: Time Synchronization Play
  hosts: servers
  vars:
    timesync_ntp_servers:
      - hostname: 0.rhel.pool.ntp.org
        iburst: yes
      - hostname: 1.rhel.pool.ntp.org
        iburst: yes
      - hostname: 2.rhel.pool.ntp.org
        iburst: yes
    timezone: UTC

  roles:
    - rhel-system-roles.timesync

  tasks:
    - name: Set timezone
      timezone:
        name: "{{ timezone }}"
```

> **NOTE**
>
> If you want to set a different time zone, you can use the `tzselect` command to look up other valid values. You can also use the `timedatectl` command to check current clock settings.

This example sets the role variables in a `vars` section of the play, but a better practice might be to configure them as inventory variables for hosts or host groups.

Consider a playbook project with the following structure:

```
[root@host playbook-project]# tree
.
├── ansible.cfg
├── group_vars
│   └── servers
│       └── timesync.yml (1)
├── inventory
└── timesync_playbook.yml (2)
```

1. Defines the time synchronization variables overriding the role defaults for hosts in group `servers` in the inventory. This file would look something like:

   ```
   timesync_ntp_servers:
     - hostname: 0.rhel.pool.ntp.org
       iburst: yes
     - hostname: 1.rhel.pool.ntp.org
       iburst: yes
     - hostname: 2.rhel.pool.ntp.org
       iburst: yes
   timezone: UTC
   ```

2. The content of the playbook simplifies to:

```
- name: Time Synchronization Play
  hosts: servers
  roles:
    - rhel-system-roles.timesync
  tasks:
    - name: Set timezone
      timezone:
        name: "{{ timezone }}"
```

This structure cleanly separates the role, the playbook code, and configuration settings. The playbook code is simple, easy to read, and should not require complex refactoring. The role content is maintained and supported by Red Hat. All the settings are handled as inventory variables.

This structure also supports a dynamic, heterogeneous environment. Hosts with new time synchronization requirements may be placed in a new host group. Appropriate variables are defined in a YAML file, and placed in the appropriate `group_vars` (or `host_vars`) subdirectory.

SELinux Role Example

As another example, the `rhel-system-roles.selinux` role simplifies management of SELinux configuration settings. It is implemented using the SELinux-related Ansible modules. The advantage of using this role instead of writing your own tasks is that it relieves you from the responsibility of writing those tasks. Instead, you provide variables to the role to configure it, and the maintained code in the role will ensure your desired SELinux configuration is applied.

Among the tasks this role can perform:

- Set enforcing or permissive mode
- Run `restorecon` on parts of the file system hierarchy
- Set SELinux Boolean values
- Set SELinux file contexts persistently
- Set SELinux user mappings

Calling the SELinux Role

Sometimes, the SELinux role must ensure the managed hosts are rebooted in order to completely apply its changes. However, it does not ever reboot hosts itself. This is so that you can control how the reboot is handled. But it means that it is a little more complicated than usual to properly use this role in a play.

The way this works is that the role will set a Boolean variable, `selinux_reboot_required`, to `true` and fail if a reboot is needed. You can use a `block`/`rescue` structure to recover from the failure, by failing the play if that variable is not set to `true` or rebooting the managed host and rerunning the role if it is `true`. The block in your play should look something like this:

```
    - name: Apply SELinux role
      block:
        - include_role:
            name: rhel-system-roles.selinux
      rescue:
        - name: Check for failure for other reasons than required reboot
          fail:
          when: not selinux_reboot_required

        - name: Restart managed host
          reboot:

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

Configuring the SELinux Role

The variables used to configure the `rhel-system-roles.selinux` role are documented in its `README.md` file. The following examples show some ways to use this role.

The `selinux_state` variable sets the mode SELinux runs in. It can be set to `enforcing`, `permissive`, or `disabled`. If it is not set, the mode is not changed.

```
selinux_state: enforcing
```

The `selinux_booleans` variable takes a list of SELinux Boolean values to adjust. Each item in the list is a hash/dictionary of variables: the `name` of the Boolean, the `state` (whether it should be `on` or `off`), and whether the setting should be `persistent` across reboots.

This example sets `httpd_enable_homedirs` to `on` persistently:

```
selinux_booleans:
  - name: 'httpd_enable_homedirs'
    state: 'on'
    persistent: 'yes'
```

The `selinux_fcontext` variable takes a list of file contexts to persistently set (or remove). It works much like the `selinux fcontext` command.

The following example ensures the policy has a rule to set the default SELinux type for all files under `/srv/www` to `httpd_sys_content_t`.

```
selinux_fcontexts:
  - target: '/srv/www(/.*)?'
    setype: 'httpd_sys_content_t'
    state: 'present'
```

The `selinux_restore_dirs` variable specifies a list of directories on which to run `restorecon`:

```
selinux_restore_dirs:
  - /srv/www
```

The `selinux_ports` variable takes a list of ports that should have a specific SELinux type.

```
selinux_ports:
  - ports: '82'
    setype: 'http_port_t'
    proto: 'tcp'
    state: 'present'
```

There are other variables and options for this role. See its `README.md` file for more information.

REFERENCES

Red Hat Enterprise Linux (RHEL) System Roles (https://access.redhat.com/articles/3050101)

Linux System Roles (https://linux-system-roles.github.io/)

← PREVIOUS (/ROL/APP/COURSES/RH294-8.4/PAGES/CH07S02) → NEXT (/ROL/APP/COURSES/RH294-8.4/PAGES/CH07S04)