Red Hat Enterprise Linux Automation with Ansible

Managing Network Configuration ☆

Objectives

After completing this section, you should be able to configure network settings and name resolution on managed hosts, and collect network-related Ansible facts.

Configuring Networking with the Network System Role

Red Hat Enterprise Linux 8 includes a collection of system Ansible roles to configure RHEL-based systems. The rhel-system-roles package installs those system roles which, for example, support the configuration of time synchronization or networking. You can list the currently installed system roles with the `ansible-galaxy list` command.

```
[user@controlnode ~]$ ansible-galaxy list
- linux-system-roles.kdump, (unknown version)
- linux-system-roles.network, (unknown version)
- linux-system-roles.postfix, (unknown version)
- linux-system-roles.selinux, (unknown version)
- linux-system-roles.timesync, (unknown version)
- rhel-system-roles.kdump, (unknown version)
- rhel-system-roles.network, (unknown version)
- rhel-system-roles.postfix, (unknown version)
- rhel-system-roles.selinux, (unknown version)
- rhel-system-roles.timesync, (unknown version)
```

Roles are located in the `/usr/share/ansible/roles` directory. A role beginning with `linux-system-roles` is a symlink to the matching `rhel-system-roles` role.

The network system role supports the configuration of networking on managed hosts. This role supports the configuration of ethernet interfaces, bridge interfaces, bonded interfaces, VLAN interfaces, MacVLAN interfaces, and Infiniband interfaces. The network role is configured with two variables, `network_provider` and `network_connections`.

```
---
network_provider: nm
network_connections:
  - name: ens4
    type: ethernet
    ip:
      address:
        - 172.25.250.30/24
```

The `network_provider` variable configures the back end provider, either `nm` (NetworkManager) or `initscripts`. On Red Hat Enterprise Linux 8, the network role uses the `nm` (NetworkManager) as a default networking provider. The `initscripts` provider is used for RHEL 6 systems, and requires the `network` service to be available. The `network_connections` variable configures the different connections, specified as a list of dictionaries, using the interface name as the connection name.

The following table lists the options for the `network_connections` variable.

Option name	Description
name	Identifies the connection profile.
state	The runtime state of a connection profile. Either `up`, if the connection profile is active, or `down` if it is not.
persistent_state	Identifies if a connection profile is persistent. Either `present` if the connection profile is persistent, or `absent` if it is not.
type	Identifies the connection type. Valid values are `ethernet`, `bridge`, `bond`, `team`, `vlan`, `macvlan`, and `infiniband`.
autoconnect	Determines if the connection automatically starts.
mac	Restricts the connection to be used on devices with a specific MAC address.
interface_name	Restricts the connection profile to be used by a specific interface.
zone	Configures the FirewallD zone for the interface.
ip	Determines the IP configuration for the connection. Supports options like for example `address`, to specify a static IP address, or `dns` to configure a DNS server.

The following example uses some of the previous options:

```
network_connections:
- name: eth0 (1)
   persistent_state: present (2)
   type: ethernet (3)
   autoconnect: yes (4)
   mac: 00:00:5e:00:53:5d (5)
   ip:
     address:
       - 172.25.250.40/24 (6)
   zone: external (7)
```

(1) Uses `eth0` as the connection name.

(2) Makes the connection persistent. This is the default value.

(3) Sets the connection type to `ethernet`.

(4) Automatically starts the connection at boot. This is the default value.

(5) Restricts the connection usage to a device with that MAC address.

(6) Configures the `172.25.250.40/24` IP address for the connection.

(7) Configures the `external` zone as the FirewallD zone of the connection.

To use the network system role, you need to specify the role name under the `roles` clause in your playbook as follows:

```
- name: NIC Configuration
  hosts: webservers
  vars:
    network_connections:
      - name: ens4
        type: ethernet
        ip:
          address:
            - 172.25.250.30/24
  roles:
    - rhel-system-roles.network
```

You can specify variables for the network role with the `vars` clause, as in the previous example, or create a YAML file with those variables under the `group_vars` or `host_vars` directories, depending on your use case.

Configuring Networking with Modules

As an alternative to the `network` system role, Ansible includes modules which support the networking configuration on a system. The `nmcli` module supports the management of both network connections and devices. This module supports the configuration of both teaming and bonding for network interfaces, as well as IPv4 and IPv6 addressing.

The following table lists some of the parameters for the `nmcli` module.

Parameter name	Description
conn_name	Configures the connection name.
autoconnect	Enables automatic connection activation on boot.
dns4	Configures DNS servers for IPv4 (up to 3).
gw4	Configures the IPv4 gateway for the interface.
ifname	Interface to be bound to the connection.
ip4	IP address (IPv4) for the interface.
state	Enables or disables the network interface.
type	Type of device or network connection.

The following example configures a static IP configuration for a network connection and device.

```
- name: NIC configuration
  nmcli:
    conn_name: ens4-conn (1)
    ifname: ens4 (2)
    type: ethernet (3)
    ip4: 172.25.250.30/24 (4)
    gw4: 172.25.250.1 (5)
    state: present (6)
```

(1) Configures `ens4-conn` as the connection name.

(2) Binds the `ens4-conn` connection to the `ens4` network interface.

3 Configures the network interface as `ethernet`.

4 Configures the `172.25.250.30/24` IP address on the interface.

5 Sets the gateway to `172.25.250.1`.

6 Makes sure the connection is available.

The `hostname` module sets the hostname for a managed host without modifying the `/etc/hosts` file. This module uses the `name` parameter to specify the new hostname, as on the task shown below:

```
- name: Change hostname
  hostname:
    name: managedhost1
```

The `firewalld` module supports the management of FirewallD on managed hosts. This modules supports the configuration of FirewallD rules for services and ports. It also supports the zone management, including the association or network interfaces and rules to a specific zone.

The following task shows how to create a FirewallD rule for the `http` service on the default zone (`public`). The task configures the rule as permanent, and makes sure it is active.

```
- name: Enabling http rule
  firewalld:
    service: http
    permanent: yes
    state: enabled
```

This task configures the `eth0` in the `external` FirewallD zone.

```
- name: Moving eth0 to external
  firewalld:
    zone: external
    interface: eth0
    permanent: yes
    state: enabled
```

The following table lists some of the parameters for the `firewalld` module.

Parameter name	Description
interface	The interface name to manage with FirewallD.
port	Port or port range. Uses the port/protocol or port-port/protocol format.
rich_rule	Rich rule for FirewallD.
service	Service name to manage with FirewallD.
source	Source network to manage with FirewallD.
zone	The FirewallD zone.
state	Enables or disables a FirewallD configuration.
type	Type of device or network connection.

Ansible Facts for Network Configuration

Ansible uses facts to retrieve information to the control node about the configuration of the managed hosts. You can use the `setup` Ansible module to retrieve all the Ansible facts for a managed host.

```
[user@controlnode ~]$ ansible webservers -m setup
host.lab.example.com | SUCCESS => {
    "ansible_facts": {
...output omitted...
    }
```

All network interfaces for a managed host are available under the `ansible_interfaces` element. You can use the `gather_subset=network` parameter for the `setup` module to restrict the facts to those included in the `network` subset. The `filter` option for the `setup` module supports fine-grained filtering based on shell-style wildcards.

```
[user@controlnode ~]$ ansible webservers -m setup \
> -a 'gather_subset=network filter=ansible_interfaces'
host.lab.example.com | SUCCESS => {
    "ansible_facts": {
        "ansible_interfaces": [
            "ens4",
            "lo",
            "ens3"
        ]
    },
    "changed": false
}
```

The previous command shows that three network interfaces are available on the managed host, `host.lab.example.com`: `lo`, `ens3`, and `ens4`.

You can retrieve additional information about the configuration for a network interface with the `ansible_`*`NIC_name`* filter for the `setup` module. For example, to retrieve the configuration for the `ens4` network interface, use the `ansible_ens4` filter.

```
[user@controlnode ~]$ ansible webservers -m setup \
> -a 'gather_subset=network filter=ansible_ens4'
host.lab.example.com | SUCCESS => {
    "ansible_facts": {
        "ansible_ens4": {
            "active": true,
            "device": "ens4",
            "features": {
            },
            "hw_timestamp_filters": [],
            "ipv4": {
                "address": "172.25.250.30",
                "broadcast": "172.25.250.255",
                "netmask": "255.255.255.0",
                "network": "172.25.250.0"
            },
            "ipv6": [
                {
                    "address": "fe80::5b42:8c94:1fc7:40ae",
                    "prefix": "64",
                    "scope": "link"
                }
            ],
            "macaddress": "52:54:00:01:fa:0a",
            "module": "virtio_net",
            "mtu": 1500,
            "pciid": "virtio1",
            "promisc": false,
            "speed": -1,
            "timestamping": [
                "tx_software",
                "rx_software",
                "software"
            ],
            "type": "ether"
        }
    },
    "changed": false
}
```

The previous command displays additional configuration details like the IP address configuration both for IPv4 and IPv6, the associated device, and the type.

The following table lists some of the available facts for the `network` subset.

Fact name	Description
ansible_dns	Includes the DNS server(s) IP address, and the search domain(s).
ansible_domain	Includes the subdomain for the managed host.
ansible_all_ipv4_addresses	Includes all the IPv4 addresses configured on the managed host.
ansible_all_ipv6_addresses	Includes all the IPv6 addresses configured on the managed host.
ansible_fqdn	Includes the FQDN for the managed host.
ansible_hostname	Includes the unqualified hostname, the string in the FQDN before the first period.
ansible_nodename	Includes the hostname for the managed host as reported by the system.

NOTE

Ansible also provides the `inventory_hostname` variable which includes the hostname as configured in Ansible's inventory file.

REFERENCES

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

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

nmcli Module Documentation (https://docs.ansible.com/ansible/2.9/modules/nmcli_module.html)

hostname Module Documentation (https://docs.ansible.com/ansible/2.9/modules/hostname_module.html)

firewalld Module Documentation (https://docs.ansible.com/ansible/2.9/modules/firewalld_module.html)

← PREVIOUS (/ROL/APP/COURSES/RH294-8.4/PAGES/CH09S08)

→ NEXT (/ROL/APP/COURSES/RH294-8.4/PAGES/CH09S10)