# Red Hat Enterprise Linux 7

# 使用 RHEL 7.9 中的 RHEL 系统角色自动化系统管理

使用 Red Hat Ansible Automation Platform playbook 在多个主机上进行一致且可重复配置的 RHEL 部署

Last Updated: 2024-01-24

# Red Hat Enterprise Linux 7 使用 RHEL 7.9 中的 RHEL 系统角色自动化系统管理

使用 Red Hat Ansible Automation Platform playbook 在多个主机上进行一致且可重复配置的 RHEL 部署

## 法律通告



## 摘要

Red Hat Enterprise Linux (RHEL)系统角色是 Ansible 角色、模块和 playbook 的集合，帮助自动化 RHEL 系统的一致性和可重复管理。使用 RHEL 系统角色，您可以通过从单一系统运行配置 playbook 来高效地管理大型系统清单。

# 目录

**使开源包含更多** ..... 6

对红帽文档提供反馈 ..... 7

**第 1 章 RHEL 系统角色简介** ..... 8

**第 2 章 准备控制节点和受管节点以使用 RHEL 系统角色** ..... 10
2.1. 在 RHEL 8 上准备一个控制节点 10
2.2. 准备受管节点 11

**第 3 章 安装和使用 COLLECTIONS** ..... 15
3.1. ANSIBLE COLLECTIONS 简介 15
3.2. 集合结构 15
3.3. 使用 CLI 安装 COLLECTIONS 15
3.4. 从 AUTOMATION HUB 安装 COLLECTIONS 16
3.5. 使用 COLLECTIONS 应用本地日志系统角色 17

**第 4 章 RHEL 中的 ANSIBLE IPMI 模块** ..... 19
4.1. RHEL_MGMT 集合 19
4.2. 使用 CLI 安装 RHEL MGMT COLLECTION 20
4.3. 使用 IPMI_BOOT 模块的示例 20
4.4. 使用 IPMI_POWER 模块的示例 21

**第 5 章 RHEL 中的 REDFISH 模块** ..... 23
5.1. REDFISH 模块 23
5.2. REDFISH 模块参数 23
5.3. 使用 REDFISH_INFO 模块 24
5.4. 使用 REDFISH_COMMAND 模块 25
5.5. 使用 REDFISH_CONFIG 模块 25

**第 6 章 使用 KERNEL_SETTINGS RHEL 系统角色永久配置内核参数** ..... 27
6.1. KERNEL_SETTINGS 角色简介 27
6.2. 使用 KERNEL_SETTINGS 角色应用所选内核参数 27

**第 7 章 使用 RHC 系统角色注册系统** ..... 31
7.1. RHC 系统角色简介 31
7.2. 使用 RHC 系统角色注册系统 31
7.3. 使用 RHC 系统角色通过 SATELLITE 注册系统 32
7.4. 使用 RHC 系统角色在注册后禁用到 INSIGHTS 的连接 34
7.5. 使用 RHC 系统角色启用存储库 34
7.6. 使用 RHC 系统角色设置发行版本 35
7.7. 在使用 RHC 系统角色注册主机时使用代理服务器 36
7.8. 使用 RHC 系统角色禁用 INSIGHTS 规则的自动更新 37
7.9. 使用 RHC RHEL 系统角色禁用 INSIGHTS 修复 39
7.10. 使用 RHC 系统角色配置 INSIGHTS 标签 40
7.11. 使用 RHC 系统角色取消系统注册 41

**第 8 章 使用 RHEL 系统角色配置网络设置** ..... 42
8.1. 使用网络 RHEL 系统角色和接口名称，使用静态 IP 地址配置以太网连接 42
8.2. 使用带有设备路径的网络 RHEL 系统角色使用静态 IP 地址配置以太网连接 43
8.3. 使用的网络 RHEL 系统角色和接口名称，使用动态 IP 地址配置以太网连接 45
8.4. 使用带有设备路径的网络 RHEL 系统角色，使用动态 IP 地址配置以太网连接 46
8.5. 使用 NETWORK RHEL 系统角色配置 VLAN 标记 47
8.6. 使用 NETWORK RHEL 系统角色配置网络桥接 48

8.7. 使用 NETWORK RHEL 系统角色配置网络绑定 50
8.8. 使用 NETWORK RHEL 系统角色配置 IPOIB 连接 51
8.9. 使用 NETWORK RHEL 系统角色将来自特定子网的流量路由到不同的默认网关 53
8.10. 使用 NETWORK RHEL 系统角色配置带有 802.1X 网络身份验证的静态以太网连接 57
8.11. 使用 NETWORK RHEL 系统角色在现有连接上设置默认网关 58
8.12. 使用 NETWORK RHEL 系统角色配置一个静态路由 60
8.13. 使用 NETWORK RHEL 系统角色配置 ETHTOOL 卸载功能 62
8.14. 网络 RHEL 系统角色的网络状态 63

**第 9 章 使用系统角色配置 FIREWALLD** 66
9.1. FIREWALL RHEL 系统角色简介 66
9.2. 使用防火墙 RHEL 系统角色重置 FIREWALLD 设置 66
9.3. 将传入的流量从一个本地端口转发到不同的本地端口 67
9.4. 使用系统角色配置端口 68
9.5. 使用 FIREWALLD RHEL 系统角色配置 DMZ FIREWALLD 区域 69

**第 10 章 系统角色中的 POSTFIX 角色的变量** 71
10.1. 其他资源 72

**第 11 章 使用系统角色配置 SELINUX** 73
11.1. SELINUX 系统角色简介 73
11.2. 使用 SELINUX 系统角色在多个系统中应用 SELINUX 设置 74

**第 12 章 使用 RHEL 系统角色配置日志记录** 76
12.1. 日志记录系统角色 76
12.2. LOGGING 系统角色参数 76
12.3. 应用 本地 日志记录系统角色 77
12.4. 过滤本地日志记录系统角色中的 LOGGING 79
12.5. 使用 LOGGING 系统角色应用远程日志解决方案 81
12.6. 使用带有 TLS 的 LOGGING 系统角色 84
12.7. 使用带有 RELP 的 LOGGING 系统角色 87
12.8. 其他资源 91

**第 13 章 使用 JOURNALD RHEL 系统角色配置 SYSTEMD 日志** 92
13.1. JOURNALD RHEL 系统角色的变量 92
13.2. 使用 JOURNALD 系统角色配置持久性日志记录 92
13.3. 其他资源 93

**第 14 章 使用 SSH 和 SSHD RHEL 系统角色配置安全通信** 94
14.1. SSH 服务器系统角色变量 94
14.2. 使用 SSHD 系统角色配置 OPENSSH 服务器 96
14.3. SSH 系统角色变量 98
14.4. 使用 SSH 系统角色配置 OPENSSH 客户端 100
14.5. 将 SSHD 系统角色用于非排除配置 101

**第 15 章 使用 VPN RHEL 系统角色使用 IPSEC 配置 VPN 连接** 103
15.1. 使用 VPN 系统角色使用 IPSEC 创建主机到主机的 VPN 103
15.2. 使用 VPN 系统角色创建与 IPSEC 的 OPPORTUNISTIC MESH VPN 连接 105
15.3. 其他资源 107

**第 16 章 使用 CRYPTO-POLICIES RHEL 系统角色设置自定义加密策略** 108
16.1. CRYPTO_POLICIES 系统角色变量和事实 108
16.2. 使用 CRYPTO_POLICIES 系统角色设置自定义加密策略 108
16.3. 其它资源 110

**第 17 章 使用 RHEL 系统角色配置 NBDE** 111

17.1. NBDE_CLIENT 和 NBDE_SERVER 系统角色 (CLEVIS 和 TANG) 简介 111
17.2. 使用 NBDE_SERVER 系统角色设置多个 TANG 服务器 111
17.3. 使用 NBDE_CLIENT 系统角色设置多个 CLEVIS 客户端 113

**第 18 章 使用 RHEL 系统角色请求证书 115**
18.1. CERTIFICATE 系统角色 115
18.2. 使用 CERTIFICATE 系统角色请求新的自签名证书 115
18.3. 使用 CERTIFICATE 系统角色从 IDM CA 请求一个新证书 116
18.4. 使用 CERTIFICATE 系统角色指定证书颁发前或后要运行的命令 118

**第 19 章 使用 KDUMP RHEL 系统角色配置自动化崩溃转储 120**
19.1. KDUMP RHEL 系统角色 120
19.2. KDUMP 角色参数 120
19.3. 使用 RHEL 系统角色配置 KDUMP 120

**第 20 章 使用 RHEL 系统角色管理本地存储 122**
20.1. STORAGE RHEL 系统角色简介 122
20.2. 在 STORAGE RHEL 系统角色中识别存储设备的参数 122
20.3. 在块设备中创建 XFS 文件系统的 ANSIBLE PLAYBOOK 示例 123
20.4. 永久挂载文件系统的 ANSIBLE PLAYBOOK 示例 124
20.5. 管理逻辑卷的 ANSIBLE PLAYBOOK 示例 124
20.6. 启用在线块丢弃的 ANSIBLE PLAYBOOK 示例 125
20.7. 创建和挂载 EXT4 文件系统的 ANSIBLE PLAYBOOK 示例 125
20.8. 创建和挂载 EXT3 文件系统的 ANSIBLE PLAYBOOK 示例 126
20.9. 使用 STORAGE RHEL 系统角色调整现有 EXT4 或 EXT3 文件系统大小的 ANSIBLE PLAYBOOK 示例 127
20.10. 使用 STORAGE RHEL 系统角色在 LVM 上调整现有文件系统的大小的 ANSIBLE PLAYBOOK 示例 128
20.11. 使用 STORAGE RHEL 系统角色创建交换卷的 ANSIBLE PLAYBOOK 示例 129
20.12. 使用存储系统角色配置 RAID 卷 129
20.13. 使用 STORAGE RHEL 系统角色配置带有 RAID 的 LVM 池 130
20.14. 使用 STORAGE RHEL 系统角色在 LVM 上压缩和去掉重复数据的 VDO 卷的 ANSIBLE PLAYBOOK 示例 131
20.15. 使用 STORAGE RHEL 系统角色创建 LUKS2 加密的卷 132
20.16. 使用 STORAGE RHEL 系统角色以百分比形式表示池卷大小的 ANSIBLE PLAYBOOK 示例 134
20.17. 其他资源 135

**第 21 章 使用 TIMESYNC RHEL 系统角色配置时间同步 136**
21.1. TIMESYNC RHEL 系统角色 136
21.2. 为单一服务器池应用 TIMESYNC 系统角色 136
21.3. 在客户端服务器上应用 TIMESYNC 系统角色 137
21.4. TIMESYNC 系统角色变量 138

**第 22 章 使用 METRICS RHEL 系统角色监控性能 139**
22.1. METRICS 系统角色简介 139
22.2. 使用 METRICS 系统角色以可视化方式监控本地系统 140
22.3. 使用 METRICS 系统角色设置监控其自身的独立系统 140
22.4. 使用 METRICS 系统角色通过本地机器监控机器的数量 141
22.5. 在使用 METRICS 系统角色监控系统时设置身份验证 142
22.6. 使用 METRICS 系统角色为 SQL SERVER 配置并启用指标集合 143

**第 23 章 使用 MICROSOFT.SQL.SERVER ANSIBLE 角色配置 MICROSOFT SQL SERVER 145**
23.1. 先决条件 145
23.2. 安装 MICROSOFT.SQL.SERVER ANSIBLE 角色 145
23.3. 使用 MICROSOFT.SQL.SERVER ANSIBLE 角色安装并配置 SQL 服务器 145
23.4. TLS 变量 146
23.5. 接受 MLSERVICE 的 EULA 147
23.6. 接受 MICROSOFT ODBC 17 的 EULA。 147

23.7. 高可用性变量 148

**第 24 章 配置系统以使用 TLOG RHEL 系统角色记录会话记录 151**
24.1. TLOG 系统角色 151
24.2. TLOG 系统角色的组件和参数 151
24.3. 部署 TLOG RHEL 系统角色 151
24.4. 为排除组或用户列表部署 TLOG RHEL 系统角色 153
24.5. 使用在 CLI 中部署的 TLOG 系统角色记录会话 154
24.6. 使用 CLI 监视记录的会话 155

**第 25 章 使用 HA_CLUSTER RHEL 系统角色配置高可用性集群 157**
25.1. HA_CLUSTER 系统角色变量 157
25.2. 为 HA_CLUSTER 系统角色指定清单 172
25.3. 为高可用性集群创建 PCSD TLS 证书和密钥文件 173
25.4. 配置不运行任何资源的高可用性集群 174
25.5. 配置带有隔离和资源的高可用性集群 175
25.6. 使用资源限制配置高可用性集群 177
25.7. 在高可用性集群中配置 COROSYNC 值 180
25.8. 使用 SBD 节点隔离配置高可用性集群 182
25.9. 使用仲裁设备配置高可用性集群 183
25.10. 使用 HA_CLUSTER 系统角色在高可用性集群中配置 APACHE HTTP 服务器 186
25.11. 其他资源 190

**第 26 章 使用 COCKPIT RHEL 系统角色安装和配置 WEB 控制台 191**
26.1. COCKPIT 系统角色 191
26.2. COCKPIT RHEL 系统角色的变量 191
26.3. 使用 COCKPIT RHEL 系统角色安装 WEB 控制台 192

**第 27 章 使用 PODMAN RHEL 系统角色管理容器 194**
27.1. PODMAN RHEL 系统角色 194
27.2. PODMAN RHEL 系统角色的变量 194
27.3. 其他资源 197

**第 28 章 使用 RHEL 系统角色将 RHEL 系统直接与 AD 集成 198**
28.1. AD_INTEGRATION 系统角色 198
28.2. AD_INTEGRATION RHEL 系统角色的变量 198
28.3. 使用 AD_INTEGRATION 系统角色将 RHEL 系统直接连接到 AD 199
28.4. 其他资源 201

# 使开源包含更多

红帽承诺替换我们的代码、文档和网页属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

# 对红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

**通过 Jira 提交反馈（需要帐户）**

1. 登录到 Jira 网站。
2. 单击顶部导航栏中的 **Create**。
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您对改进的建议。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

# 第 1 章 RHEL 系统角色简介

通过使用 RHEL 系统角色，您可以远程管理跨 RHEL 主版本的多个 RHEL 系统的系统配置。RHEL 系统角色是 Ansible 角色和模块的集合。要使用它来配置系统，您必须使用以下组件：

**控制节点**

控制节点是您运行 Ansible 命令和 playbook 的系统。您的控制节点可以是 Ansible Automation Platform、Red Hat Satellite 或 RHEL 9、8 或 7 主机。如需更多信息，请参阅 在 RHEL 8 上准备控制节点。

**受管节点**

受管节点是您通过 Ansible 管理的服务器和网络设备。受管节点有时也称为主机。Ansible 不必安装到受管节点上。如需更多信息，请参阅 准备受管节点。

**Ansible playbook**

在 playbook 中，您可以定义要在受管节点上实现的配置，或受管节点上系统要执行的一组步骤。Playbook 是 Ansible 的配置、部署和编配语言。

**清单（Inventory）**

在清单文件中，您列出受管节点，并指定信息，如每个受管节点的 IP 地址等。在清单中，您还可以组织受管节点，创建并嵌套组以更轻松地扩展。清单文件有时也称为 hostfile。

在 Red Hat Enterprise Linux 8 中，您可以使用 **rhel-system-roles** 软件包提供的以下角色，该软件包在 **AppStream** 存储库中提供：

| 角色名称 | 角色描述 | 章节标题 |
|---|---|---|
| **certificate** | 证书问题和续订 | 使用 RHEL 系统角色请求证书 |
| **cockpit** | Web 控制台 | 使用 cockpit RHEL 系统角色安装和配置 web 控制台 |
| **crypto_policies** | 系统范围的加密策略 | 设置跨系统的自定义加密策略 |
| **firewall** | Firewalld | 使用系统角色配置 firewalld |
| **ha_cluster** | HA 集群 | 使用系统角色配置高可用性集群 |
| **kdump** | 内核转储 | 使用 RHEL 系统角色配置 kdump |
| **kernel_settings** | 内核设置 | 使用 Ansible 角色永久配置内核参数 |
| **logging** | 日志记录 | 使用 logging 系统角色 |
| **metrics** | 指标(PCP) | 使用 RHEL 系统角色监控性能 |
| **microsoft.sql.server** | Microsoft SQL Server | 使用 microsoft.sql.server Ansible 角色配置 Microsoft SQL Server |

| 角色名称 | 角色描述 | 章节标题 |
| --- | --- | --- |
| **network** | 网络 | 使用 network RHEL 系统角色管理 InfiniBand 连接 |
| **nbde_client** | 网络绑定磁盘加密客户端 | 使用 nbde_client 和 nbde_server 系统角色 |
| **nbde_server** | 网络绑定磁盘加密服务器 | 使用 nbde_client 和 nbde_server 系统角色 |
| **postfix** | postfix | 系统角色中 postfix 角色的变量 |
| **selinux** | SELinux | 使用系统角色配置 SELinux |
| **ssh** | SSH 客户端 | 使用 ssh 系统角色配置安全通信 |
| **sshd** | SSH 服务器 | 使用 ssh 系统角色配置安全通信 |
| **storage** | 存储 | 使用 RHEL 系统角色管理本地存储 |
| **tlog** | 终端会话记录 | 使用 tlog RHEL 系统角色为会话记录配置系统 |
| **timesync** | 时间同步 | 使用 RHEL 系统角色配置时间同步 |
| **vpn** | VPN | 使用 vpn RHEL 系统角色配置具有 IPsec 的 VPN 连接 |

**其他资源**

- Red Hat Enterprise Linux (RHEL) 系统角色
- **rhel-system-roles** 软件包提供的 **/usr/share/doc/rhel-system-roles/**

# 第 2 章 准备控制节点和受管节点以使用 RHEL 系统角色

在使用单独的 RHEL 系统角色管理服务和设置之前，您必须准备控制节点和受管节点。

## 2.1. 在 RHEL 8 上准备一个控制节点

在使用 RHEL 系统角色前，您必须配置一个控制节点。然后，此系统会根据 playbook 从清单中配置受管主机。

**先决条件**

- RHEL 7.9 已安装。有关安装 RHEL 的更多信息，请参阅 安装指南。
- 该系统已注册到客户门户网站。
- **Red Hat Enterprise Linux Server** 订阅已附加到系统。
- 如果在客户门户网站帐户中可用，则会将 **Ansible Automation Platform** 订阅附加到系统。

**流程**

1. 安装 **rhel-system-roles** 软件包：

```
[root@control-node]# yum install rhel-system-roles
```

   此命令将 **ansible-core** 软件包安装为依赖项。

2. 创建名为 **ansible** 的用户，以管理并运行 playbook：

```
[root@control-node]# useradd ansible
```

3. 切换到新创建的 **ansible** 用户：

```
[root@control-node]# su - ansible
```

   以这个用户身份执行其余步骤。

4. 创建一个 SSH 公钥和私钥：

```
[ansible@control-node]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ansible/.ssh/id_rsa): <password>
...
```

   为密钥文件使用推荐的默认位置。

5. 可选： 要防止 Ansible 每次建立连接时提示您输入 SSH 密钥密码，请配置 SSH 代理。

6. 使用以下内容创建 **~/.ansible.cfg** 文件：

```
[defaults]
inventory = /home/ansible/inventory
remote_user = ansible
```

```
[privilege_escalation]
become = True
become_method = sudo
become_user = root
become_ask_pass = True
```

**注意**

~/**.ansible.cfg** 文件中的设置具有更高的优先级，并覆盖全局 **/etc/ansible/ansible.cfg** 文件中的设置。

使用这些设置，Ansible 执行以下操作：

- 管理指定清单文件中的主机。
- 当帐户建立到受管节点的 SSH 连接时，请使用 **remote_user** 参数中设置的帐户。
- 使用 **sudo** 工具，以 **root** 用户身份在受管节点上执行任务。
- 每次应用 playbook 时，都会提示输入远程用户的 root 密码。出于安全考虑，建议使用它。

7. 创建一个列出受管主机主机名的 INI 或 YAML 格式的 ~/**inventory** 文件。您还可以在清单文件中定义主机组。例如，以下是 INI 格式的清单文件，它有三个主机，以及一个名为 **US** 的主机组：

```
managed-node-01.example.com

[US]
managed-node-02.example.com ansible_host=192.0.2.100
managed-node-03.example.com
```

请注意，控制节点必须能够解析主机名。如果 DNS 服务器无法解析某些主机名，请在主机条目旁边添加 **ansible_host** 参数来指定其 IP 地址。

**后续步骤**

- 准备受管节点。如需更多信息，请参阅 准备受管节点。

**其他资源**

- RHEL 9 和 RHEL 8.6 及更新的 AppStream 软件仓库中包含的 Ansible Core 软件包支持范围
- 如何使用 subscription-manager 在红帽客户门户网站中注册和订阅系统
- **ssh-keygen (1)** man page
- 通过 ssh-agent，使用 SSH 密钥连接到远程机器
- Ansible 配置设置
- 如何构建清单
- 在 RHEL 8.6 和 9.0 中使用 Ansible 的更新

## 2.2. 准备受管节点

受管节点是清单中列出的系统，它将根据 playbook 由控制节点配置。您不必在受管主机上安装 Ansible。

**先决条件**

- 您已准备好了控制节点。如需更多信息，请参阅 在 RHEL 8 上准备控制节点 。
- 您有从控制节点进行 SSH 访问的权限。

**重要**

以 **root** 用户身份直接访问 SSH 有安全风险。要降低这个风险，您将在此节点上创建一个本地用户，并在准备受管节点时配置一个 **sudo** 策略。然后，控制节点上的 Ansible 可以使用本地用户帐户登录受管节点，并以不同的用户（如 **root** ）运行 playbook。

**流程**

1. 创建一个名为 **ansible** 的用户：

```
[root@managed-node-01]# useradd ansible
```

控制节点稍后使用这个用户建立与这个主机的 SSH 连接。

2. 为 **ansible** 用户设置密码：

```
[root@managed-node-01]# passwd ansible
Changing password for user ansible.
New password: <password>
Retype new password: <password>
passwd: all authentication tokens updated successfully.
```

当 Ansible 使用 **sudo** 以 **root** 用户身份执行任务时，您必须输入此密码。

3. 在受管主机上安装 **ansible** 用户的 SSH 公钥：

   a. 以 **ansible** 用户身份登录控制节点，并将 SSH 公钥复制到受管节点：

```
[ansible@control-node]$ ssh-copy-id managed-node-01.example.com
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/home/ansible/.ssh/id_rsa.pub"
The authenticity of host 'managed-node-01.example.com (192.0.2.100)' can't be
established.
ECDSA key fingerprint is
SHA256:9bZ33GJNODK3zbNhybokN/6Mq7hu3vpBXDrCxe7NAvo.
```

   b. 提示时，输入 **yes** 进行连接：

```
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that
are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is
to install the new keys
```

   c. 提示时，输入密码：

```
ansible@managed-node-01.example.com's password: <password>

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '<managed-node-01.example.com>'"
and check to make sure that only the key(s) you wanted were added.
```

d. 通过在控制节点上远程执行命令来验证 SSH 连接：

```
[ansible@control-node]$ ssh <managed-node-01.example.com> whoami
ansible
```

4. 为 **ansible** 用户创建一个 **sudo** 配置：

a. 使用 **visudo** 命令创建并编辑 **/etc/sudoers.d/ansible** 文件：

```
[root@managed-node-01]# visudo /etc/sudoers.d/ansible
```

在正常编辑器中使用 **visudo** 的好处是，该实用程序提供基本的健全检查和检查是否有解析错误，然后再安装该文件。

b. 在 **/etc/sudoers.d/ansible** 文件中配置满足您要求的 **sudoers** 策略，例如：

- 要为 **ansible** 用户授予权限，以便在输入 **ansible** 用户密码后以此主机上的任何用户和组身份运行所有命令，请使用：

```
ansible ALL=(ALL) ALL
```

- 要向 **ansible** 用户授予权限，以便在不输入 **ansible** 用户密码的情况下以该主机上任何用户和组的身份运行所有命令，请使用：

```
ansible ALL=(ALL) NOPASSWD: ALL
```

或者，配置匹配您安全要求的更精细的策略。有关 **sudoers** 策略的详情，请查看 **sudoers(5)** 手册页。

**验证**

1. 验证您可以是否可以在所有受管节点上执行控制节点的命令：

```
[ansible@control-node]$ ansible all -m ping
BECOME password: <password>
managed-node-01.example.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
...
```

硬编码的所有组都会动态包含清单文件中列出的所有主机。

2. 使用 Ansible **command** 模块在受管主机上运行 **whoami** 工具来验证特权升级是否正常工作：

```
[ansible@control-node]$ ansible managed-node-01.example.com -m command -a
whoami
BECOME password: <password>
managed-node-01.example.com | CHANGED | rc=0 >>
root
```

如果命令返回 root，则您在受管节点上配置的 **sudo** 正确。

**其他资源**

- 在 RHEL 8 上准备一个控制节点。
- **sudoers (5)** 手册页

# 第 3 章 安装和使用 COLLECTIONS

## 3.1. ANSIBLE COLLECTIONS 简介

Ansible Collections 是一种发布、维护和使用自动化的新方法。通过组合多种类型的 Ansible 内容，如 playbook、角色、模块和插件，您可以从灵活性和可扩展性的改进中受益。

Ansible Collections 是传统 RHEL 系统角色格式的一个选项。以 Ansible Collection 格式使用 RHEL 系统角色与以传统 RHEL 系统角色格式使用它几乎相同。区别是 Ansible Collections 使用 **完全限定集合名称** (FQCN)的概念，其由 **名字空间** 和 **集合名称** 组成。我们使用的 **名字空间** 是 **redhat**，**集合名称** 是 **rhel_system_roles**。因此，虽然 **kernel_settings** 角色的传统 RHEL 系统角色格式显示为 **rhel-system-roles.kernel_settings** （带横线），但使用 **kernel_settings** 角色的 Collection **完全限定集合名称** 将显示为 **redhat.rhel_system_roles.kernel_settings** （带下划线）。

**名字空间** 和 **集合名称** 的组合确保对象是唯一的。它也确保对象在 Ansible Collections 和名字空间之间共享，且没有任何冲突。

**其他资源**

- 要通过访问 Automation Hub 来使用红帽认证的集合，您必须有一个 Ansible Automation Platform（AAP 订阅）。

## 3.2. 集合结构

Collections 是 Ansible 内容的软件包格式。数据结构如下：

- docs/：集合的本地文档，以及示例（如果角色提供了文档）
- galaxy.yml：将成为 Ansible Collection 软件包一部分的 MANIFEST.json 的源数据
- playbooks/: playbook 位于此处
  - tasks/：包含用于 include_tasks/import_tasks 的"任务列表文件"
- plugins/：此处提供所有 Ansible 插件和模块，各自位于其子目录中
  - modules/：Ansible 模块
  - modules_utils/：用于开发模块的通用代码
  - lookup/：搜索插件
  - filter/: Jinja2 过滤器插件
  - connection/：所需的连接插件（如果不使用默认的）
- roles/：Ansible 角色的目录
- test/：对集合的内容进行测试

## 3.3. 使用 CLI 安装 COLLECTIONS

集合是 Ansible 内容的分发格式，可包含 playbook、角色、模块和插件。

您可以通过 Ansible Galaxy、浏览器或使用命令行来安装 Collections。

**先决条件**

- 访问一个或多个 *受管节点*。
- 对 *控制节点* 的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。
  在控制节点上：
  - **ansible-core** 和 **rhel-system-roles** 软件包已安装 。
  - 列出受管节点的清单文件。

**流程**

- 通过 RPM 软件包安装集合：

```
# yum install rhel-system-roles
```

安装完成后，角色作为 **redhat.rhel_system_roles.<role_name>** 提供。另外，您可以在 **/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/roles/<role_name>/README.md** 找到每个角色的文档。

**验证步骤**

要验证安装，请在 localhost 上以 **check** 模式运行 **kernel_settings** 角色。您还必须使用 **--become** 参数，因为对于 Ansible **package** 模块，它是必需的。但是，该参数不会更改您的系统：

1. 运行以下命令:

```
$ ansible-playbook -c local -i localhost, --check --become
/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/tests/kernel_settings
ests_default.yml
```

命令输出的最后一行应包含值 **failed=0**。

**注意**

**localhost** 之后的逗号是必需的。即使列表中只有一个主机，您页必须添加它。如果没有它，**ansible-playbook** 会将 **localhost** 识别为文件或目录。

**其他资源**

- **ansible-playbook** 手册页。
- **ansible-playbook** 命令的 **-i** 选项

## 3.4. 从 AUTOMATION HUB 安装 COLLECTIONS

如果使用 Automation Hub，您可以安装托管在 Automation Hub 上的 RHEL 系统角色集合。

**先决条件**

- 访问一个或多个 *受管节点*。
- 对 *控制节点* 的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。
  在控制节点上：

- **ansible-core** 和 **rhel-system-roles** 软件包已安装 。
- 列出受管节点的清单文件。

**流程**

1. 定义 Red Hat Automation Hub 作为 **ansible.cfg** 配置文件中内容的默认源。请参阅 将 Red Hat Automation Hub 配置为内容的主源。
2. 从 Automation Hub 安装 **redhat.rhel_system_roles** 集合：

```
# ansible-galaxy collection install redhat.rhel_system_roles
```

安装完成后，角色作为 **redhat.rhel_system_roles.<role_name>** 提供。另外，您可以在 **/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/roles/<role_name>/README.md** 找到每个角色的文档。

**验证步骤**

要验证安装，请在 localhost 上以 **check** 模式运行 **kernel_settings** 角色。您还必须使用 **--become** 参数，因为对于 Ansible **package** 模块，它是必需的。但是，该参数不会更改您的系统：

1. 运行以下命令:

```
$ ansible-playbook -c local -i localhost, --check --become
/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/tests/kernel_settings
ests_default.yml
```

命令输出的最后一行应包含值 **failed=0**。

**注意**

**localhost** 之后的逗号是必需的。即使列表中只有一个主机，您页必须添加它。如果没有它，**ansible-playbook** 会将 **localhost** 识别为文件或目录。

**其他资源**

- **ansible-playbook** 手册页。
- **ansible-playbook** 命令的 **-i** 选项

## 3.5. 使用 COLLECTIONS 应用本地日志系统角色

下例中使用集合来准备和应用 Ansible playbook，以在一组独立的计算机上配置日志记录解决方案。

**先决条件**

- rhel-system-roles 的 Collection 格式是从 rpm 软件包或从 Automation Hub 安装的。

**流程**

1. 创建定义所需角色的 playbook:
   a. 创建新 YAML 文件，并在文本编辑器中打开，例如：
      ▪

```
# vi logging-playbook.yml
```

b. 将以下内容插入 YAML 文件中：

```
---
- name: Deploying basics input and implicit files output
  hosts: all
  roles:
    - redhat.rhel_system_roles.logging
  vars:
    logging_inputs:
      - name: system_input
        type: basics
    logging_outputs:
      - name: files_output
        type: files
    logging_flows:
      - name: flow1
        inputs: [system_input]
        outputs: [files_output]
```

2. 在特定清单上执行 playbook:

```
# ansible-playbook -i inventory-file logging-playbook.yml
```

其中：

- *inventory-file* 是清单文件的名称。
- *logging-playbook.yml* 是您使用的 playbook。

**验证步骤**

1. 测试配置文件 **/etc/rsyslog.conf** 和 **/etc/rsyslog.d** 的语法：

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

2. 验证系统是否向日志发送信息：

a. 发送测试信息：

```
# logger test
```

b. 查看 **/var/log/messages** 日志，例如：

```
# cat /var/log/messages
Aug  5 13:48:31 hostname root[6778]: test
```

**hostname** 是客户端系统的主机名。日志显示输入 logger 命令的用户的用户名，在本例中是 **root**。

# 第 4 章 RHEL 中的 ANSIBLE IPMI 模块

## 4.1. RHEL_MGMT 集合

智能平台管理接口(IPMI)是一组标准协议的规范，用来与基板管理控制器(BMC)设备通信。**IPMI** 模块允许您启用和支持硬件管理自动化。**IPMI** 模块由以下产品提供：

- **rhel_mgmt** Collection。软件包名称为 **ansible-collection-redhat-rhel_mgmt**。
- RHEL 7.9 AppStream，作为新的 **ansible-collection-redhat-rhel_mgmt** 软件包的一部分。

rhel_mgmt 集合中提供以下 IPMI 模块：

- **ipmi_boot** ：管理引导设备顺序
- **ipmi_power** ：机器的电源管理

用于 IPMI 模块的必要参数有：

- **ipmi_boot** 参数:

| 模块名称 | 描述 |
| --- | --- |
| name | BMC 的主机名或 IP 地址 |
| password | 连接到 BMC 的密码 |
| bootdev | 在下次引导时使用的设备<br>* 网络<br>* 软盘<br>* 硬盘<br>* 安全<br>* 光盘<br>* 设置<br>* 默认 |
| 用户 | 连接到 BMC 的用户名 |

- **ipmi_power** 参数:

| 模块名称 | 描述 |
| --- | --- |
| name | BMC 主机名或 IP 地址 |
| password | 连接到 BMC 的密码 |

| 模块名称 | 描述 |
| --- | --- |
| user | 连接到 BMC 的用户名 |
| 状态 | 检查机器是否处于所需的状态<br>* 开<br>* 关<br>* 关闭<br>* 重置<br>* 启动 |

## 4.2. 使用 CLI 安装 RHEL MGMT COLLECTION

您可以使用命令行安装 **rhel_mgmt** Collection。

**先决条件**

- **ansible-core** 软件包已安装。

**流程**

- 通过 RPM 软件包安装集合：

```
# yum install ansible-collection-redhat-rhel_mgmt
```

安装完成后，IPMI 模块位于 **redhat.rhel_mgmt** Ansible 集合中。

**其他资源**

- **ansible-playbook** 手册页。

## 4.3. 使用 IPMI_BOOT 模块的示例

以下示例演示了如何在 playbook 中使用 **ipmi_boot** 模块来为下次引导设置引导设备。为了简单起见，示例使用与 Ansible 控制主机和受管主机相同的主机，从而在执行 playbook 的同一主机上执行模块。

**先决条件**

- rhel_mgmt 集合已安装。
- **python3-pyghmi** 软件包中的 **pyghmi** 库已安装在以下位置之一：
  - 执行 playbook 的主机。

    - 受管主机。如果您使用 localhost 作为受管主机，请在您要执行 playbook 的主机上安装 **python3-pyghmi** 软件包。

- 您要控制的 IPMI BMC 可以从您执行 playbook 的主机访问，或从受管主机（如果不使用 localhost 作为受管主机）访问。请注意，其 BMC 由模块配置的主机通常与执行该模块的主机（Ansible 受管主机）不同，因为模块使用 IPMI 协议通过网络联系 BMC。
- 您拥有使用适当访问级别访问 BMC 的凭证。

**流程**

1. 创建包含以下内容的 *playbook.yml* 文件：

```
---
- name: Sets which boot device will be used on next boot
  hosts: localhost
    tasks:
    - redhat.rhel_mgmt.ipmi_boot:
        name: bmc.host.example.com
          user: admin_user
          password: basics
          bootdev: hd
```

2. 针对 localhost 执行 playbook：

```
# ansible-playbook playbook.yml
```

因此，输出会返回值 "success"。

## 4.4. 使用 IPMI_POWER 模块的示例

本示例演示了如何在 playbook 中使用 **ipmi_boot** 模块来检查系统是否已开启。为了简单起见，示例使用与 Ansible 控制主机和受管主机相同的主机，从而在执行 playbook 的同一主机上执行模块。

**先决条件**

- rhel_mgmt 集合已安装。
- **python3-pyghmi** 软件包中的 **pyghmi** 库已安装在以下位置之一：
    - 执行 playbook 的主机。
    - 受管主机。如果您使用 localhost 作为受管主机，请在您要执行 playbook 的主机上安装 **python3-pyghmi** 软件包。
- 您要控制的 IPMI BMC 可以从您执行 playbook 的主机访问，或从受管主机（如果不使用 localhost 作为受管主机）访问。请注意，其 BMC 由模块配置的主机通常与执行该模块的主机（Ansible 受管主机）不同，因为模块使用 IPMI 协议通过网络联系 BMC。
- 您拥有使用适当访问级别访问 BMC 的凭证。

**流程**

1. 创建包含以下内容的 *playbook.yml* 文件：

```
---
- name: Turn the host on
  hosts: localhost
    tasks:
    - redhat.rhel_mgmt.ipmi_power:
        name: bmc.host.example.com
          user: admin_user
          password: basics
          state: on
```

2. 执行 playbook：

```
# ansible-playbook playbook.yml
```

输出返回值 "true"。

# 第 5 章 RHEL 中的 REDFISH 模块

用于远程管理设备的 Redfish 模块现在是 **redhat.rhel_mgmt** Ansible 集合的一部分。通过 Redfish 模块，您可以使用标准 HTTPS 传输和 JSON 格式获取服务器或控制它们的信息，轻松在裸机服务器和平台硬件上轻松使用管理自动化。

## 5.1. REDFISH 模块

**redhat.rhel_mgmt** Ansible 集合提供了 Redfish 模块，以支持 Ansible over Redfish 中的硬件管理。**redhat.rhel_mgmt** 集合位于 **ansible-collection-redhat-rhel_mgmt** 软件包中。要安装它，请参阅使用 CLI 安装 redhat.rhel_mgmt Collection。

以下 Redfish 模块包括在 **redhat.rhel_mgmt** 集合中：

1. **redfish_info** ：**redfish_info** 模块检索有关远程 Out-Of-Band (OOB)控制器的信息，如系统清单。
2. **redfish_command** ：**redfish_command** 模块执行 Out-Of-Band (OOB)控制器操作，如日志管理和用户管理，以及电源操作，如系统重启、开机和关机。
3. **redfish_config**: **redfish_config** 模块执行 OOB 控制器操作，如更改 OOB 配置或设置 BIOS 配置。

## 5.2. REDFISH 模块参数

用于 Redfish 模块的参数包括：

| redfish_info 参数： | 描述 |
|---|---|
| **baseuri** | （必需）- OOB 控制器的基本 URI。 |
| **category** | （必需）- 在 OOB 控制器上执行的类别列表。默认值为 ["Systems"]。 |
| **命令** | （必需）- 在 OOB 控制器上执行的命令列表。 |
| **username** | OOB 控制器身份验证的用户名。 |
| **password** | OOB 控制器身份验证的密码。 |

| redfish_command 参数： | 描述 |
|---|---|
| **baseuri** | （必需）- OOB 控制器的基本 URI。 |
| **category** | （必需）- 在 OOB 控制器上执行的类别列表。默认值为 ["Systems"]。 |
| **命令** | （必需）- 在 OOB 控制器上执行的命令列表。 |

| redfish_command 参数： | 描述 |
| --- | --- |
| **username** | OOB 控制器身份验证的用户名。 |
| **password** | OOB 控制器身份验证的密码。 |

| redfish_config 参数： | 描述 |
| --- | --- |
| **baseuri** | （必需）- OOB 控制器的基本 URI。 |
| **category** | （必需）- 在 OOB 控制器上执行的类别列表。默认值为 ["Systems"]。 |
| **命令** | （必需）- 在 OOB 控制器上执行的命令列表。 |
| **username** | OOB 控制器身份验证的用户名。 |
| **password** | OOB 控制器身份验证的密码。 |
| **bios_attributes** | 更新的 BIOS 属性。 |

## 5.3. 使用 REDFISH_INFO 模块

以下示例演示了如何在 playbook 中使用 **redfish_info** 模块来获取 CPU 清单的信息。为了简单起见，示例使用与 Ansible 控制主机和受管主机相同的主机，从而在执行 playbook 的同一主机上执行模块。

**先决条件**

- 已安装 **redhat.rhel_mgmt** 集合。
- **python3-pyghmi** 软件包中的 **pyghmi** 库被安装到受管主机上。如果使用 localhost 作为受管主机，请在执行 playbook 的主机上安装 **python3-pyghmi** 软件包。
- OOB 控制器访问详细信息。

**流程**

1. 创建包含以下内容的 *playbook.yml* 文件：

```
---
- name: Get CPU inventory
  hosts: localhost
  tasks:
    - redhat.rhel_mgmt.redfish_info:
        baseuri: "{{ baseuri }}"
        username: "{{ username }}"
        password: "{{ password }}"
```

```
      category: Systems
      command: GetCpuInventory
    register: result
```

2. 针对 localhost 执行 playbook：

```
# ansible-playbook playbook.yml
```

因此，输出会返回 CPU 清单详情。

## 5.4. 使用 REDFISH_COMMAND 模块

以下示例演示了如何在 playbook 中使用 **redfish_command** 模块来打开系统。为了简单起见，示例使用与 Ansible 控制主机和受管主机相同的主机，从而在执行 playbook 的同一主机上执行模块。

**先决条件**

- 已安装 **redhat.rhel_mgmt** 集合。
- **python3-pyghmi** 软件包中的 **pyghmi** 库被安装到受管主机上。如果使用 localhost 作为受管主机，请在执行 playbook 的主机上安装 **python3-pyghmi** 软件包。
- OOB 控制器访问详细信息。

**流程**

1. 创建包含以下内容的 *playbook.yml* 文件：

```
---
- name: Power on system
  hosts: localhost
  tasks:
    - redhat.rhel_mgmt.redfish_command:
        baseuri: "{{ baseuri }}"
        username: "{{ username }}"
        password: "{{ password }}"
        category: Systems
        command: PowerOn
```

2. 针对 localhost 执行 playbook：

```
# ansible-playbook playbook.yml
```

因此，系统会开机。

## 5.5. 使用 REDFISH_CONFIG 模块

以下示例演示了如何在 playbook 中使用 **redfish_config** 模块将系统配置为使用 UEFI 引导。为了简单起见，示例使用与 Ansible 控制主机和受管主机相同的主机，从而在执行 playbook 的同一主机上执行模块。

**先决条件**

- 已安装 **redhat.rhel_mgmt** 集合。

- **python3-pyghmi** 软件包中的 **pyghmi** 库被安装到受管主机上。如果使用 localhost 作为受管主机，请在执行 playbook 的主机上安装 **python3-pyghmi** 软件包。
- OOB 控制器访问详细信息。

**流程**

1. 创建包含以下内容的 *playbook.yml* 文件：

```
---
- name: "Set BootMode to UEFI"
  hosts: localhost
  tasks:
    - redhat.rhel_mgmt.redfish_config:
        baseuri: "{{ baseuri }}"
        username: "{{ username }}"
        password: "{{ password }}"
        category: Systems
        command: SetBiosAttributes
        bios_attributes:
          BootMode: Uefi
```

2. 针对 localhost 执行 playbook：

```
# ansible-playbook playbook.yml
```

因此，系统引导模式被设置为 UEFI。

# 第 6 章 使用 KERNEL_SETTINGS RHEL 系统角色永久配置内核参数

作为熟悉 Red Hat Ansible 的一名经验丰富的用户，您可以使用 **kernel_settings** 角色同时在多个客户端上配置内核参数。这个解决方案：

- 提供带有有效输入设置的友好接口。
- 保留所有预期的内核参数。

从控制计算机运行 **kernel_settings** 角色后，内核参数将立即应用于受管系统，并在重新启动后保留。

**重要**

请注意，通过 RHEL 频道提供的 RHEL 系统角色可在默认 AppStream 软件仓库中作为 RPM 软件包提供给 RHEL 客户。RHEL 系统角色还可以通过 Ansible Automation Hub 为客户提供 Ansible 订阅的集合。

## 6.1. KERNEL_SETTINGS 角色简介

RHEL System Roles（系统角色）是一组角色，为远程管理多个系统提供一致的配置接口。

RHEL 系统角色是使用 **kernel_settings** 系统角色自动配置内核。**rhel-system-roles** 软件包包含这个系统角色以及参考文档。

要将内核参数以自动化方式应用到一个或多个系统，请在 playbook 中使用 **kernel_settings** 角色和您选择的一个或多个角色变量。playbook 是一个或多个人类可读的 play 的列表，采用 YAML 格式编写。

您可以使用清单文件来定义一组您希望 Ansible 根据 playbook 配置的系统。

使用 **kernel_settings** 角色，您可以配置：

- 使用 **kernel_settings_sysctl** 角色变量的内核参数
- 使用 **kernel_settings_sysfs** 角色变量的各种内核子系统、硬件设备和设备驱动程序
- **systemd** 服务管理器的 CPU 相关性，并使用 **kernel_settings_systemd_cpu_affinity** 角色变量处理其分叉
- 内核内存子系统使用 **kernel_settings_transparent_hugepages** 和 **kernel_settings_transparent_hugepages_defrag** 角色变量透明巨页

**其他资源**

- **/usr/share/doc/rhel-system-roles/kernel_settings/kernel_settings/** 目录中的 **README.md** 和 **README.html** 文件
- 使用 playbook
- 如何构建清单

## 6.2. 使用 KERNEL_SETTINGS 角色应用所选内核参数

按照以下步骤准备并应用 Ansible playbook 来远程配置内核参数，从而对多个受管操作系统产生持久性。

**先决条件**

- 您有 **root** 权限。
- 根据您的 RHEL 订阅，您可以在控制机器上安装 **ansible-core** 和 **rhel-system-roles** 软件包。
- 受管主机清单存在于控制计算机上，Ansible 能够连接它们。

**流程**

1. 另外，还可查看 **清单（inventory）**文件 ：

```
# cat /home/jdoe/<ansible_project_name>/inventory
[testingservers]
pdoe@192.168.122.98
fdoe@192.168.122.226

[db-servers]
db1.example.com
db2.example.com

[webservers]
web1.example.com
web2.example.com
192.0.2.42
```

   该文件定义 **[testingservers]** 组和其他组。它允许您对特定的系统集合更有效地运行 Ansible。

2. 创建配置文件来为 Ansible 操作设置默认值和特权升级。

   a. 创建新 YAML 文件，并在文本编辑器中打开，例如：

```
# vi /home/jdoe/<ansible_project_name>/ansible.cfg
```

   b. 将以下内容插入到文件中：

```
[defaults]
inventory = ./inventory

[privilege_escalation]
become = true
become_method = sudo
become_user = root
become_ask_pass = true
```

   **[defaults]** 部分指定受管主机清单文件的路径。**[privilege_escalation]** 部分定义用户特权转移到指定受管主机上的 **root**。这对成功配置内核参数是必需的。运行 Ansible playbook 时，会提示您输入用户密码。用户在连接到受管主机后，通过 **sudo** 自动切换为 **root**。

3. 创建使用 **kernel_settings** 角色的 Ansible playbook。

   a. 创建新 YAML 文件，并在文本编辑器中打开，例如：

```
# vi /home/jdoe/<ansible_project_name>/kernel-roles.yml
```

此文件代表一个 playbook，通常包含了一组有特定顺序的任务（也称为 *play* ）列表。这些任何会根据 **inventory** 文件中选择的特定管理主机进行。

b. 将以下内容插入到文件中：

```
---
-
  hosts: testingservers
  name: "Configure kernel settings"
  roles:
    - rhel-system-roles.kernel_settings
  vars:
    kernel_settings_sysctl:
      - name: fs.file-max
        value: 400000
      - name: kernel.threads-max
        value: 65536
    kernel_settings_sysfs:
      - name: /sys/class/net/lo/mtu
        value: 65000
    kernel_settings_transparent_hugepages: madvise
```

**name** 键是可选的。它将一个任意字符串作为标签与该 play 关联，并确定该 play 的用途。Play 中的 **hosts** 键指定对其运行 play 的主机。此键的值或值可以作为被管理的主机的单独名称提供，也可以作为 **inventory** 文件中定义的一组主机提供。

**vars** 部分代表包含所选内核参数名称和值的变量列表。

**roles** 键指定系统角色将配置 **vars** 部分中提到的参数和值。

**注意**

您可以修改 playbook 中的内核参数及其值以符合您的需要。

4. （可选）验证 play 中的语法是否正确。

```
#  ansible-playbook --syntax-check kernel-roles.yml

playbook: kernel-roles.yml
```

本例演示了对 playbook 的成功验证。

5. 执行 playbook。

```
#  ansible-playbook kernel-roles.yml

...

BECOME password:

PLAY [Configure kernel settings]
*********************************************************************************
```

```
PLAY RECAP
*********************************************************************************************************
fdoe@192.168.122.226      : ok=10   changed=4    unreachable=0    failed=0    skipped=6
rescued=0    ignored=0
pdoe@192.168.122.98       : ok=10   changed=4    unreachable=0    failed=0    skipped=6
rescued=0    ignored=0
```

在 Ansible 运行 playbook 之前，会提示您输入密码，因此在受管主机上的用户能够切换到 **root**，这对于配置内核参数是必需的。

recap 部分显示 play 成功完成所有受管主机 (**failed=0**)，并且已应用了 4 个内核参数 (**changed=4**)。

6. 重启您的受管主机并检查受影响的内核参数,以验证是否应用了更改并在重启后保留。

**其他资源**

- 准备控制节点和受管节点以使用 RHEL 系统角色
- **/usr/share/doc/rhel-system-roles/kernel_settings/** 目录中的 **README.html** 和 **README.md** 文件
- 构建清单
- 配置 Ansible
- 使用 Playbook
- 使用变量
- 角色

# 第 7 章 使用 RHC 系统角色注册系统

**rhc** RHEL 系统角色使管理员能够使用红帽订阅管理(RHSM)和 Satellite 服务器自动注册多个系统。该角色还支持使用 Ansible，进行与 Insights 相关配置和管理任务。

## 7.1. RHC 系统角色简介

RHEL 系统角色是一组角色，为远程管理多个系统提供一致的配置接口。远程主机配置(**rhc**)系统角色使管理员能够轻松地将 RHEL 系统注册到 Red Hat Subscription Management (RHSM)和 Satellite 服务器。默认情况下，当使用 **rhc** 系统角色注册系统时，系统会连接到 Insights。另外，使用 **rhc** 系统角色，您可以：

- 配置到 Red Hat Insights 的连接
- 启用和禁用存储库
- 配置用于连接的代理
- 配置 Insights 修复以及自动更新
- 设置系统发行版本
- 配置 insights 标签

## 7.2. 使用 RHC 系统角色注册系统

您可以使用 **rhc** RHEL 系统角色将您的系统注册到红帽。默认情况下，**rhc** RHEL 系统角色在注册系统时，会将系统连接到 Red Hat Insights。

**先决条件**

- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。

**流程**

1. 创建一个 vault 来保存敏感信息：

```
$ ansible-vault create secrets.yml
New Vault password: password
Confirm New Vault password: password
```

2. **ansible-vault create** 命令会创建一个加密的 vault 文件，并在编辑器中打开它。输入您要保存在密码库中的敏感数据，例如：

```
activationKey: activation_key
username: username
password: password
```

3. 保存更改，并关闭编辑器。Ansible 加密 vault 中的数据。
之后，您可以使用 **ansible-vault edit *secrets.yml*** 命令编辑 vault 中的数据。

4. 可选：显示 vault 内容：

```
$ ansible-vault view secrets.yml
```

5. 创建一个 playbook 文件，如 ~/**registration.yml**，并根据您要执行的操作使用以下选项之一：

   a. 要使用激活码和机构 ID （推荐）注册，请使用以下 playbook：

```
---
- name: Registering system using activation key and organization ID
  hosts: managed-node-01.example.com
  vars_files:
    - secrets.yml
  vars:
    rhc_auth:
      activation_keys:
        keys:
          - "{{ activationKey }}"
     rhc_organization: organizationID
  roles:
    - role: rhel-system-roles.rhc
```

   b. 要使用用户名和密码注册，请使用以下 playbook：

```
---
- name: Registering system with username and password
  hosts:  managed-node-01.example.com
  vars_files:
    - secrets.yml
  vars:
    rhc_auth:
      login:
        username: "{{ username }}"
        password: "{{ password }}"
  roles:
    - role: rhel-system-roles.rhc
```

6. 运行 playbook：

```
# ansible-playbook ~/registration.yml --ask-vault-pass
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md** 文件

## 7.3. 使用 RHC 系统角色通过 SATELLITE 注册系统

当组织使用 Satellite 管理系统时，需要通过 Satellite 注册系统。您可以使用 **rhc** RHEL 系统角色通过 Satellite 远程注册您的系统。

**先决条件**

- 以可在受管主机上运行 playbook 的用户登录到控制节点。

- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。

**流程**

1. 创建一个 vault 来保存敏感信息：

```
$ ansible-vault create secrets.yml
New Vault password: password
Confirm New Vault password: password
```

2. **ansible-vault create** 命令创建一个加密的文件，并在编辑器中打开它。输入您要保存在密码库中的敏感数据，例如：

```
activationKey: activation_key
```

3. 保存更改，并关闭编辑器。Ansible 加密 vault 中的数据。
之后，您可以使用 **ansible-vault edit *secrets.yml*** 命令编辑 vault 中的数据。

4. 可选：显示 vault 内容：

```
$ ansible-vault view secrets.yml
```

5. 创建一个 playbook 文件，如 ~/**registration-sat.yml**。

6. 使用 ~/**registration-sat.yml** 中的以下文本，使用激活码和机构 ID 注册系统：

```
---
- name: Register to the custom registration server and CDN
  hosts: managed-node-01.example.com
  vars_files:
    - secrets.yml
  vars:
    rhc_auth:
      login:
        activation_keys:
          keys:
            - "{{ activationKey }}"
        rhc_organization: organizationID
    rhc_server:
      hostname: example.com
        port: 443
        prefix: /rhsm
    rhc_baseurl: http://example.com/pulp/content
  roles:
    - role: rhel-system-roles.rhc
```

7. 运行 playbook：

```
# ansible-playbook ~/registration-sat.yml --ask-vault-pass
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md** 文件

## 7.4. 使用 RHC 系统角色在注册后禁用到 INSIGHTS 的连接

当使用 **rhc** RHEL 系统角色注册系统时，默认情况下角色会启用到 Red Hat Insights 的连接。如果需要，您可以使用 **rhc** 系统角色禁用它。

**先决条件**

- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。
- 系统已经注册。

**流程**

1. 创建一个 playbook 文件，如 ~/**dis-insights.yml**，并在其中添加以下内容：

```
---
- name: Disable Insights connection
  hosts: managed-node-01.example.com
  vars:
    rhc_insights:
      state: absent
  roles:
    - role: rhel-system-roles.rhc
```

2. 运行 playbook：

```
# ansible-playbook ~/dis-insights.yml
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md** 文件

## 7.5. 使用 RHC 系统角色启用存储库

您可以使用 **rhc** RHEL 系统角色远程启用或禁用受管节点上的存储库。

**先决条件**

- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。
- 您有要在受管节点上启用或禁用的存储库的详情。
- 您已注册系统。

**流程**

1. 创建一个 playbook 文件，如 ~/**configure-repos.yml** ：

   a. 要启用存储库：

```
---
- name: Enable repository
  hosts: managed-node-01.example.com
  vars:
    rhc_repositories:
      - {name: "RepositoryName", state: enabled}
  roles:
    - role: rhel-system-roles.rhc
```

   b. 要禁用存储库：

```
---
- name: Disable repository
  hosts: managed-node-01.example.com
  vars:
    rhc_repositories:
      - {name: "RepositoryName", state: disabled}
  roles:
    - role: rhel-system-roles.rhc
```

2. 运行 playbook：

```
# ansible-playbook ~/configure-repos.yml
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md** 文件

## 7.6. 使用 RHC 系统角色设置发行版本

您可以将系统限制为只使用特定 RHEL 次版本的存储库，而不是最新版本的存储库。这样，您可以将您的系统锁定到特定的次版本。

**先决条件**

- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。
- 您知道您要锁定系统的次 RHEL 版本。请注意，您只能将系统锁定到主机当前运行的 RHEL 次版本或之后的次版本。
- 您已注册系统。

**流程**

1. 创建一个 playbook 文件，如 ~/**release.yml** ：

```
---
- name: Set Release
  hosts: managed-node-01.example.com
  vars:
    rhc_release: "8.6"
  roles:
    - role: rhel-system-roles.rhc
```

2. 运行 playbook：

```
# ansible-playbook ~/release.yml
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md** 文件

## 7.7. 在使用 RHC 系统角色注册主机时使用代理服务器

如果您的安全限制只允许通过代理服务器访问互联网，您可以在使用 **rhc** RHEL 系统角色注册系统时在 playbook 中指定代理设置。

**先决条件**

- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。

**流程**

1. 创建一个 vault 来保存敏感信息：

```
$ ansible-vault create secrets.yml
New Vault password: password
Confirm New Vault password: password
```

2. **ansible-vault create** 命令创建一个加密的文件，并在编辑器中打开它。输入您要保存在密码库中的敏感数据，例如：

```
username: username
password: password
proxy_username: proxyusernme
proxy_password: proxypassword
```

3. 保存更改，并关闭编辑器。Ansible 加密 vault 中的数据。
   之后，您可以使用 **ansible-vault edit *secrets.yml*** 命令编辑 vault 中的数据。

4. 可选：显示 vault 内容：

```
$ ansible-vault view secrets.yml
```

5. 创建一个 playbook 文件，如 ~/**configure-proxy.yml** ：

   a. 要使用代理注册到 RHEL 客户门户网站：

```
---
- name: Register using proxy
  hosts: managed-node-01.example.com
  vars_files:
    - secrets.yml
  vars:
    rhc_auth:
      login:
        username: "{{ username }}"
        password: "{{ password }}"
    rhc_proxy:
      hostname: proxy.example.com
      port: 3128
      username: "{{ proxy_username }}"
      password: "{{ proxy_password }}"
  roles:
    - role: rhel-system-roles.rhc
```

   b. 要从 Red Hat Subscription Manager 服务的配置中删除代理服务器：

```
---
- name: To stop using proxy server for registration
  hosts: managed-node-01.example.com
  vars_files:
    - secrets.yml
  vars:
    rhc_auth:
      login:
        username: "{{ username }}"
        password: "{{ password }}"
     rhc_proxy: {"state":"absent"}
  roles:
    - role: rhel-system-roles.rhc
```

6. 运行 playbook：

```
# ansible-playbook ~/configure-proxy.yml --ask-vault-pass
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md** 文件

## 7.8. 使用 RHC 系统角色禁用 INSIGHTS 规则的自动更新

您可以使用 **rhc** RHEL 系统角色禁用 Red Hat Insights 的自动集合规则更新。默认情况下，当您的系统连接到 Red Hat Insights 时，这个选项就启用了。您可以使用 **rhc** RHEL 系统角色禁用它。

**注意**

如果禁用了此功能，您就存在使用过时的规则定义文件，且没有获得最新验证更新的风险。

**先决条件**

- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。
- 您已注册系统。

**流程**

1. 创建一个 vault 来保存敏感信息：

```
$ ansible-vault create secrets.yml
New Vault password: password
Confirm New Vault password: password
```

2. **ansible-vault create** 命令创建一个加密的文件，并在编辑器中打开它。输入您要保存在密码库中的敏感数据，例如：

```
username: username
password: password
```

3. 保存更改，并关闭编辑器。Ansible 加密 vault 中的数据。
之后，您可以使用 **ansible-vault edit *secrets.yml*** 命令编辑 vault 中的数据。

4. 可选：显示 vault 内容：

```
$ ansible-vault view secrets.yml
```

5. 创建一个 playbook 文件，如 ~/**auto-update.yml**，并将以下内容添加到其中：

```
---
 - name: Disable Red Hat Insights autoupdates
   hosts: managed-node-01.example.com
   vars_files:
     - secrets.yml
   vars:
    rhc_auth:
      login:
        username: "{{ username }}"
        password: "{{ password }}"
    rhc_insights:
      autoupdate: false
      state: present
   roles:
     - role: rhel-system-roles.rhc
```

6. 运行 playbook：

```
# ansible-playbook ~/auto-update.yml --ask-vault-pass
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md** 文件

## 7.9. 使用 RHC RHEL 系统角色禁用 INSIGHTS 修复

您可以使用 **rhc** RHEL 系统角色将系统配置为自动更新动态配置。当将您的系统连接到 Red Hat Insights 时，它默认启用。如果需要，您可以禁用它。

**注意**

使用 **rhc** 系统角色启用补救可确保您的系统在直接连接到红帽时已准备好补救。对于连接到 Satellite 或 Capsule 的系统，必须以不同的方式实现补救。有关 Red Hat Insights 补救的更多信息，请参阅 Red Hat Insights 补救指南 。

**先决条件**

- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。
- 您已启用了 Insights 补救。
- 您已注册系统。

**流程**

1. 要启用补救，请创建一个 playbook 文件，如 **~/remediation.yml** ：

```
---
- name: Disable remediation
  hosts: managed-node-01.example.com
  vars:
    rhc_insights:
      remediation: absent
      state: present
  roles:
    - role: rhel-system-roles.rhc
```

2. 运行 playbook：

```
# ansible-playbook ~/remediation.yml
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md** 文件

## 7.10. 使用 RHC 系统角色配置 INSIGHTS 标签

您可以对系统过滤和分组使用标签。您还可以根据要求自定义标签。

**先决条件**

- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。

**流程**

1. 创建一个 vault 来保存敏感信息：

```
$ ansible-vault create secrets.yml
New Vault password: password
Confirm New Vault password: password
```

2. **ansible-vault create** 命令创建一个加密的文件，并在编辑器中打开它。输入您要保存在密码库中的敏感数据，例如：

```
username: username
password: password
```

3. 保存更改，并关闭编辑器。Ansible 加密 vault 中的数据。
之后，您可以使用 **ansible-vault edit *secrets.yml*** 命令编辑 vault 中的数据。

4. 可选：显示 vault 内容：

```
$ ansible-vault view secrets.yml
```

5. 创建一个 playbook 文件，如 ~/**tags.yml**，并将以下内容添加到其中：

```
---
- name: Creating tags
  hosts: managed-node-01.example.com
  vars_files:
    - secrets.yml
  vars:
    rhc_auth:
      login:
        username: "{{ username }}"
        password: "{{ password }}"
    rhc_insights:
      tags:
        group: group-name-value
          location: location-name-value
          description:
            - RHEL8
            - SAP
           sample_key:value
```

```
      state: present
  roles:
    - role: rhel-system-roles.rhc
```

6. 运行 playbook：

```
# ansible-playbook ~/remediation.yml --ask-vault-pass
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md** 文件
- 如需更多信息，请参阅 系统过滤和对 Red Hat Insights 分组 。

## 7.11. 使用 RHC 系统角色取消系统注册

如果您不再需要订阅服务，您可以从红帽取消注册系统。

**先决条件**

- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。
- 系统已经注册。

**流程**

1. 要取消注册，请创建一个 playbook 文件，如 **~/unregister.yml** ，并在其中添加以下内容：

```
---
- name: Unregister the system
  hosts: managed-node-01.example.com
  vars:
    rhc_state: absent
  roles:
    - role: rhel-system-roles.rhc
```

2. 运行 playbook：

```
# ansible-playbook ~/unregister.yml
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.rhc/README.md** 文件

# 第 8 章 使用 RHEL 系统角色配置网络设置

管理员可以使用 **network** RHEL 系统角色自动化与网络相关的配置和管理任务。

## 8.1. 使用网络 RHEL 系统角色和接口名称，使用静态 IP 地址配置以太网连接

您可以使用 **network** RHEL 系统角色远程配置以太网连接。

例如，以下流程使用以下设置为 **enp7s0** 设备创建 NetworkManager 连接配置文件：

- 静态 IPv4 地址 - **192.0.2.1** 和 **/24** 子网掩码
- 静态 IPv6 地址 - **2001:db8:1::1** 和 **/64** 子网掩码
- IPv4 默认网关 - **192.0.2.254**
- IPv6 默认网关 - **2001:db8:1::fffe**
- IPv4 DNS 服务器 - **192.0.2.200**
- IPv6 DNS 服务器 - **2001:db8:1::ffbb**
- DNS 搜索域 - **example.com**

在 Ansible 控制节点上执行此步骤。

**先决条件**

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。
- 服务器配置中有一个物理或者虚拟以太网设备。
- 受管节点使用 NetworkManager 配置网络。

**流程**

1. 创建包含以下内容的 playbook 文件，如 ***~/ethernet-static-IP.yml*** ：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure an Ethernet connection with static IP
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        - name: enp7s0
          interface_name: enp7s0
```

```
    type: ethernet
    autoconnect: yes
    ip:
      address:
        - 192.0.2.1/24
        - 2001:db8:1::1/64
      gateway4: 192.0.2.254
      gateway6: 2001:db8:1::fffe
      dns:
        - 192.0.2.200
        - 2001:db8:1::ffbb
      dns_search:
        - example.com
    state: up
```

2. 运行 playbook：

```
# ansible-playbook ~/ethernet-static-IP.yml
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件

## 8.2. 使用带有设备路径的网络 RHEL 系统角色使用静态 IP 地址配置以太网连接

您可以使用 **network** RHEL 系统角色远程配置以太网连接。

您可以使用以下命令识别设备路径：

```
# udevadm info /sys/class/net/<device_name> | grep ID_PATH=
```

例如，以下流程会使用以下设置为与 PCI ID **0000:00:0[1-3].0** 表达式，而不是 **0000:00:02.0** 匹配的设备创建 NetworkManager 连接配置文件：

- 静态 IPv4 地址 - **192.0.2.1** 和 **/24** 子网掩码
- 静态 IPv6 地址 - **2001:db8:1::1** 和 **/64** 子网掩码
- IPv4 默认网关 - **192.0.2.254**
- IPv6 默认网关 - **2001:db8:1::fffe**
- IPv4 DNS 服务器 - **192.0.2.200**
- IPv6 DNS 服务器 - **2001:db8:1::ffbb**
- DNS 搜索域 - **example.com**

在 Ansible 控制节点上执行此步骤。

**先决条件**

- 您已准备好控制节点和受管节点

- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。
- 服务器配置中有一个物理或者虚拟以太网设备。
- 受管节点使用 NetworkManager 配置网络。

**流程**

1. 创建包含以下内容的 playbook 文件，如 ***~/ethernet-static-IP.yml*** ：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure an Ethernet connection with static IP
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        - name: example
          match:
            path:
              - pci-0000:00:0[1-3].0
              - &!pci-0000:00:02.0
          type: ethernet
          autoconnect: yes
          ip:
            address:
              - 192.0.2.1/24
              - 2001:db8:1::1/64
            gateway4: 192.0.2.254
            gateway6: 2001:db8:1::fffe
            dns:
              - 192.0.2.200
              - 2001:db8:1::ffbb
            dns_search:
              - example.com
          state: up
```

本例中的 **match** 参数定义了 Ansible 将剧本应用到与 PCI ID **0000:00:0[1-3].0** 匹配的设备，但没有 **0000:00:02.0** 设备。有关可以使用的特殊修饰符和通配符的详情，请查看 **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件中的 **match** 参数描述。

2. 运行 playbook：

```
# ansible-playbook ~/ethernet-static-IP.yml
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件

## 8.3. 使用的网络 RHEL 系统角色和接口名称，使用动态 IP 地址配置以太网连接

您可以使用 **network** RHEL 系统角色远程配置以太网连接。对于具有动态 IP 地址设置的连接，NetworkManager 会为来自 DHCP 服务器的连接请求 IP 设置。

在 Ansible 控制节点上执行此步骤。

**先决条件**

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。
- 服务器配置中有一个物理或者虚拟以太网设备。
- 网络中有 DHCP 服务器
- 受管节点使用 NetworkManager 配置网络。

**流程**

1. 创建包含以下内容的 playbook 文件，如 ***~/ethernet-dynamic-IP.yml*** ：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure an Ethernet connection with dynamic IP
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        - name: enp7s0
          interface_name: enp7s0
          type: ethernet
          autoconnect: yes
          ip:
            dhcp4: yes
            auto6: yes
          state: up
```

2. 运行 playbook：

```
# ansible-playbook ~/ethernet-dynamic-IP.yml
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件

## 8.4. 使用带有设备路径的网络 RHEL 系统角色，使用动态 IP 地址配置以太网连接

您可以使用 **network** RHEL 系统角色远程配置以太网连接。对于具有动态 IP 地址设置的连接，NetworkManager 会为来自 DHCP 服务器的连接请求 IP 设置。

您可以使用以下命令识别设备路径：

```
# udevadm info /sys/class/net/<device_name> | grep ID_PATH=
```

在 Ansible 控制节点上执行此步骤。

**先决条件**

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。
- 服务器配置中有一个物理或者虚拟以太网设备。
- 网络中有 DHCP 服务器。
- 受管主机使用 NetworkManager 配置网络。

**流程**

1. 创建包含以下内容的 playbook 文件，如 ***~/ethernet-dynamic-IP.yml*** ：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure an Ethernet connection with dynamic IP
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        - name: example
          match:
            path:
              - pci-0000:00:0[1-3].0
              - &!pci-0000:00:02.0
          type: ethernet
          autoconnect: yes
          ip:
            dhcp4: yes
            auto6: yes
          state: up
```

本例中的 **match** 参数定义了 Ansible 将剧本应用到与 PCI ID **0000:00:0[1-3].0** 匹配的设备，但没有 **0000:00:02.0** 设备。有关可以使用的特殊修饰符和通配符的详情，请查看 **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件中的 **match** 参数描述。

2. 运行 playbook：

```
# ansible-playbook ~/ethernet-dynamic-IP.yml
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件

## 8.5. 使用 NETWORK RHEL 系统角色配置 VLAN 标记

您可以使用 **network** RHEL 系统角色配置 VLAN 标记。这个示例在这个以太网连接之上添加了一个以太网连接和 ID 为 **10** 的 VLAN。作为子设备，VLAN 连接包含 IP、默认网关和 DNS 配置。

根据您的环境，相应地进行调整。例如：

- 要在其他连接中将 VLAN 用作端口（如绑定），请省略 **ip** 属性，并在子配置中设置 IP 配置。
- 若要在 VLAN 中使用 team、bridge 或 bond 设备，请调整您在 VLAN 中使用的端口的 **interface_name** 和 **类型** 属性。

在 Ansible 控制节点上执行此步骤。

**先决条件**

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。

**流程**

1. 创建包含以下内容的 playbook 文件，如 ***~/vlan-ethernet.yml***：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure a VLAN that uses an Ethernet connection
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        # Add an Ethernet profile for the underlying device of the VLAN
        - name: enp1s0
          type: ethernet
```

```
    interface_name: enp1s0
    autoconnect: yes
    state: up
    ip:
      dhcp4: no
      auto6: no

  # Define the VLAN profile
  - name: enp1s0.10
    type: vlan
    ip:
      address:
        - "192.0.2.1/24"
        - "2001:db8:1::1/64"
      gateway4: 192.0.2.254
      gateway6: 2001:db8:1::fffe
      dns:
        - 192.0.2.200
        - 2001:db8:1::ffbb
      dns_search:
        - example.com
    vlan_id: 10
    parent: enp1s0
    state: up
```

VLAN 配置文件中的 **parent** 属性将 VLAN 配置为在 **enp1s0** 设备之上运行。

2. 运行 playbook：

```
# ansible-playbook ~/vlan-ethernet.yml
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件

## 8.6. 使用 NETWORK RHEL 系统角色配置网络桥接

您可以使用 **network** RHEL 系统角色配置 Linux 网桥。例如，使用它来配置使用两个以太网设备的网桥，并设置 IPv4 和 IPv6 地址、默认网关和 DNS 配置。

**注意**

在网桥上设置 IP 配置,而不是在 Linux 网桥的端口上设置。

在 Ansible 控制节点上执行此步骤。

**先决条件**

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。

- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。
- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。

**流程**

1. 创建包含以下内容的 playbook 文件，如 ***~/bridge-ethernet.yml*** ：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure a network bridge that uses two Ethernet ports
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        # Define the bridge profile
        - name: bridge0
          type: bridge
          interface_name: bridge0
          ip:
            address:
              - "192.0.2.1/24"
              - "2001:db8:1::1/64"
            gateway4: 192.0.2.254
            gateway6: 2001:db8:1::fffe
            dns:
              - 192.0.2.200
              - 2001:db8:1::ffbb
            dns_search:
              - example.com
          state: up

        # Add an Ethernet profile to the bridge
        - name: bridge0-port1
          interface_name: enp7s0
          type: ethernet
          controller: bridge0
          port_type: bridge
          state: up

        # Add a second Ethernet profile to the bridge
        - name: bridge0-port2
          interface_name: enp8s0
          type: ethernet
          controller: bridge0
          port_type: bridge
          state: up
```

2. 运行 playbook：

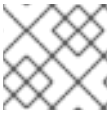
```
# ansible-playbook ~/bridge-ethernet.yml
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件

## 8.7. 使用 NETWORK RHEL 系统角色配置网络绑定

您可以使用 **network** RHEL 系统角色来配置 Linux 绑定。例如，使用它来在 active-backup 模式中配置使用两个以太网设备的网络绑定，并设置 IPv4 和 IPv6 地址、默认网关和 DNS 配置。



**注意**

在绑定上设置 IP 配置，而不是在 Linux 绑定的端口上设置。

在 Ansible 控制节点上执行此步骤。

**先决条件**

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。
- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。

**流程**

1. 创建包含以下内容的 playbook 文件，如 ***~/bond-ethernet.yml*** ：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure a network bond that uses two Ethernet ports
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        # Define the bond profile
        - name: bond0
          type: bond
          interface_name: bond0
          ip:
            address:
              - "192.0.2.1/24"
              - "2001:db8:1::1/64"
            gateway4: 192.0.2.254
            gateway6: 2001:db8:1::fffe
            dns:
              - 192.0.2.200
              - 2001:db8:1::ffbb
            dns_search:
```

```
      - example.com
    bond:
      mode: active-backup
    state: up

  # Add an Ethernet profile to the bond
  - name: bond0-port1
    interface_name: enp7s0
    type: ethernet
    controller: bond0
    state: up

  # Add a second Ethernet profile to the bond
  - name: bond0-port2
    interface_name: enp8s0
    type: ethernet
    controller: bond0
    state: up
```

2. 运行 playbook：

```
# ansible-playbook ~/bond-ethernet.yml
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件

## 8.8. 使用 NETWORK RHEL 系统角色配置 IPOIB 连接

您可以使用 **网络** RHEL 系统角色为InfiniBand (IPoIB)设备上的 IP 远程创建 NetworkManager 连接配置文件。例如，通过运行 Ansible playbook，为 **mlx4_ib0** 接口远程添加具有以下设置的 InfiniBand 连接：

- 一个 IPoIB 设备 - **mlx4_ib0.8002**
- 一个分区密钥 **p_key** - **0x8002**
- 静态 **IPv4** 地址 - **192.0.2.1**，子网掩码为 **/24**
- 一个静态 **IPv6** 地址 - **2001:db8:1::1** ，子网掩码为 **/64**

在 Ansible 控制节点上执行此步骤。

**先决条件**

- 您已准备好控制节点和受管节点
- 您可以以可在受管主机上运行 playbook 的用户身份登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。
- 一个名为 **mlx4_ib0** 的 InfiniBand 设备被安装在受管节点上。
- 受管节点使用 NetworkManager 配置网络。

**流程**

1. 创建一个 playbook 文件，如包含以下内容的 ***~/IPoIB.yml*** ：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure IPoIB
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:

        # InfiniBand connection mlx4_ib0
        - name: mlx4_ib0
          interface_name: mlx4_ib0
          type: infiniband

        # IPoIB device mlx4_ib0.8002 on top of mlx4_ib0
        - name: mlx4_ib0.8002
          type: infiniband
          autoconnect: yes
          infiniband:
            p_key: 0x8002
            transport_mode: datagram
          parent: mlx4_ib0
          ip:
            address:
              - 192.0.2.1/24
              - 2001:db8:1::1/64
          state: up
```

如果您像本例所示设置了 **p_key** 参数，请不要在 IPoIB 设备上设置 **interface_name** 参数。

2. 运行 playbook：

```
# ansible-playbook ~/IPoIB.yml
```

**验证**

1. 在 **managed-node-01.example.com** 主机上显示 **mlx4_ib0.8002** 设备的 IP 设置：

```
# ip address show mlx4_ib0.8002
...
inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute ib0.8002
   valid_lft forever preferred_lft forever
inet6 2001:db8:1::1/64 scope link tentative noprefixroute
   valid_lft forever preferred_lft forever
```

2. 显示 **mlx4_ib0.8002** 设备的分区密钥(P_Key)：

```
# cat /sys/class/net/mlx4_ib0.8002/pkey
0x8002
```

3. 显示 **mlx4_ib0.8002** 设备的模式：

```
# cat /sys/class/net/mlx4_ib0.8002/mode
datagram
```

**其他资源**

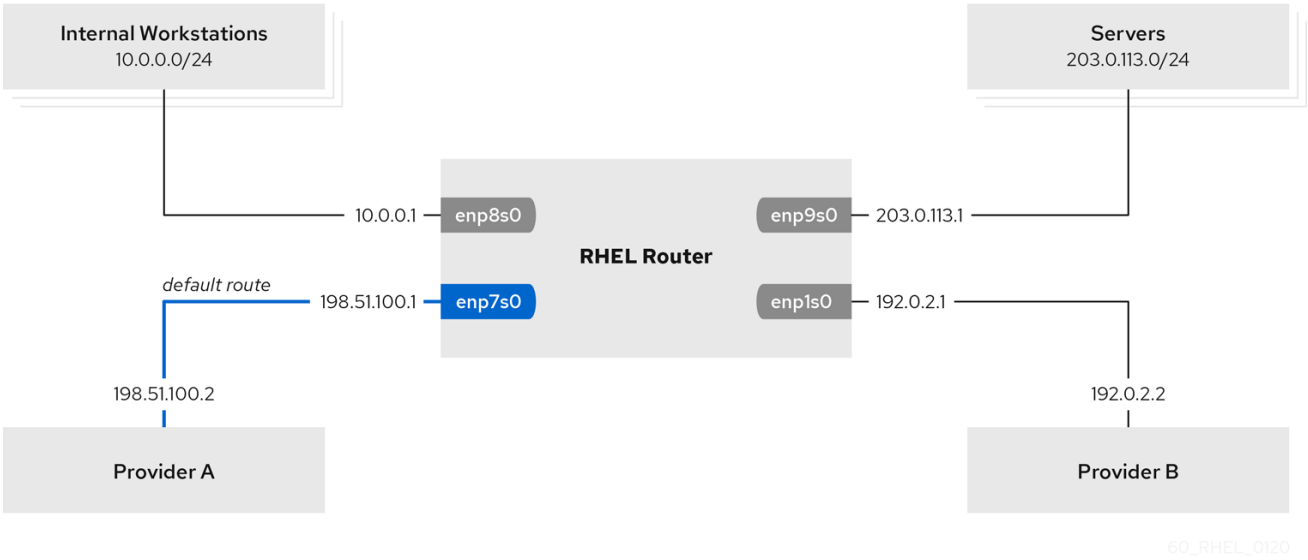
- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件

## 8.9. 使用 NETWORK RHEL 系统角色将来自特定子网的流量路由到不同的默认网关

您可以使用基于策略的路由为来自特定子网的流量配置不同的默认网关。例如，您可以将 RHEL 配置为默认路由将所有流量路由到互联网供应商 A 的路由器。但是，从内部工作站子网接收的流量路由到供应商 B。

要远程和在多个节点上配置基于策略的路由，您可以使用 RHEL **network** 系统角色。在 Ansible 控制节点上执行此步骤。

此流程假设以下网络拓扑：




**先决条件**

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户对其具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。
- 受管节点使用 **NetworkManager** 和 **firewalld** 服务。
- 您要配置的受管节点有 4 个网络接口：
  - **enp7s0** 接口已连接到提供商 A 的网络。提供商网络中的网关 IP 为 **198.51.100.2**，网络使用 **/30** 网络掩码。

  - **enp1s0** 接口连接到提供商 B 的网络。提供商网络中的网关 IP 为 **192.0.2.2**，网络使用 **/30** 网络掩码。
  - **enp8s0** 接口已与连有内部工作站的 **10.0.0.0/24** 子网相连。
  - **enp9s0** 接口已与连有公司服务器的 **203.0.113.0/24** 子网相连。
- 内部工作站子网中的主机使用 **10.0.0.1** 作为默认网关。在此流程中，您可以将这个 IP 地址分配给路由器的 **enp8s0** 网络接口。
- 服务器子网中的主机使用 **203.0.113.1** 作为默认网关。在此流程中，您可以将这个 IP 地址分配给路由器的 **enp9s0** 网络接口。

**流程**

1. 创建包含以下内容的 playbook 文件，如 **~/pbr.yml** ：

```
---
- name: Configuring policy-based routing
  hosts: managed-node-01.example.com
  tasks:
  - name: Routing traffic from a specific subnet to a different default gateway
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        - name: Provider-A
          interface_name: enp7s0
          type: ethernet
          autoconnect: True
          ip:
            address:
              - 198.51.100.1/30
            gateway4: 198.51.100.2
            dns:
              - 198.51.100.200
          state: up
          zone: external

        - name: Provider-B
          interface_name: enp1s0
          type: ethernet
          autoconnect: True
          ip:
            address:
              - 192.0.2.1/30
            route:
              - network: 0.0.0.0
                prefix: 0
                gateway: 192.0.2.2
                table: 5000
          state: up
          zone: external

        - name: Internal-Workstations
```

```
    interface_name: enp8s0
    type: ethernet
    autoconnect: True
    ip:
      address:
        - 10.0.0.1/24
      route:
        - network: 10.0.0.0
          prefix: 24
          table: 5000
      routing_rule:
        - priority: 5
          from: 10.0.0.0/24
          table: 5000
    state: up
    zone: trusted

  - name: Servers
    interface_name: enp9s0
    type: ethernet
    autoconnect: True
    ip:
      address:
        - 203.0.113.1/24
    state: up
    zone: trusted
```

2. 运行 playbook：

```
# ansible-playbook ~/pbr.yml
```

**验证**

1. 在内部工作站子网的 RHEL 主机上：
   a. 安装 **traceroute** 软件包：

```
# yum install traceroute
```

   b. 使用 **traceroute** 工具显示到互联网上主机的路由：

```
# traceroute redhat.com
traceroute to redhat.com (209.132.183.105), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)     0.337 ms  0.260 ms  0.223 ms
 2  192.0.2.1 (192.0.2.1)   0.884 ms  1.066 ms  1.248 ms
 ...
```

   命令的输出显示路由器通过 **192.0.2.1**，即提供商 B 的网络来发送数据包。

2. 在服务器子网的 RHEL 主机上：
   a. 安装 **traceroute** 软件包：

```
# yum install traceroute
```

b. 使用 **traceroute** 工具显示到互联网上主机的路由：

```
# traceroute redhat.com
traceroute to redhat.com (209.132.183.105), 30 hops max, 60 byte packets
 1  203.0.113.1 (203.0.113.1)    2.179 ms  2.073 ms  1.944 ms
 2  198.51.100.2 (198.51.100.2)  1.868 ms  1.798 ms  1.549 ms
 ...
```

命令的输出显示路由器通过 **198.51.100.2**，即供应商 A 的网络来发送数据包。

3. 在使用 RHEL 系统角色配置的 RHEL 路由器上：

a. 显示规则列表：

```
# ip rule list
0:      from all lookup local
5:    from 10.0.0.0/24 lookup 5000
32766:  from all lookup main
32767:  from all lookup default
```

默认情况下，RHEL 包含表 **local**、**main** 和 **default** 的规则。

b. 显示表 **5000** 中的路由：

```
# ip route list table 5000
0.0.0.0/0 via 192.0.2.2 dev enp1s0 proto static metric 100
10.0.0.0/24 dev enp8s0 proto static scope link src 192.0.2.1 metric 102
```

c. 显示接口和防火墙区：

```
# firewall-cmd --get-active-zones
external
  interfaces: enp1s0 enp7s0
trusted
  interfaces: enp8s0 enp9s0
```

d. 验证 **external** 区是否启用了伪装：

```
# firewall-cmd --info-zone=external
external (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp1s0 enp7s0
  sources:
  services: ssh
  ports:
  protocols:
  masquerade: yes
  ...
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件

## 8.10. 使用 NETWORK RHEL 系统角色配置带有 802.1X 网络身份验证的静态以太网连接

使用 **network** RHEL 系统角色，您可以自动创建使用 802.1X 标准的以太网连接来验证客户端。例如，通过运行 Ansible playbook，使用以下设置，为 **enp1s0** 接口远程添加一个以太网连接：

- 静态 IPv4 地址 - **192.0.2.1** 和 **/24** 子网掩码
- 静态 IPv6 地址 - **2001:db8:1::1** 和 **/64** 子网掩码
- IPv4 默认网关 - **192.0.2.254**
- IPv6 默认网关 - **2001:db8:1::fffe**
- IPv4 DNS 服务器 - **192.0.2.200**
- IPv6 DNS 服务器 - **2001:db8:1::ffbb**
- DNS 搜索域 - **example.com**
- 使用 **TLS** 可扩展身份验证协议(EAP)的 802.1X 网络身份验证.

在 Ansible 控制节点上执行此步骤。

**先决条件**

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中
- 网络支持 802.1X 网络身份验证。
- 受管节点使用 NetworkManager。
- control 节点上存在 TLS 身份验证所需的以下文件：
  - 客户端密钥存储在 **/srv/data/client.key** 文件中。
  - 客户端证书存储在 **/srv/data/client.crt** 文件中。
  - 证书颁发机构(CA)证书存储在 **/srv/data/ca.crt** 文件中。

**流程**

1. 创建包含以下内容的 playbook 文件，如 ***~/enable-802.1x.yml*** ：

```
---
- name: Configure an Ethernet connection with 802.1X authentication
  hosts: managed-node-01.example.com
  tasks:
    - name: Copy client key for 802.1X authentication
      copy:
```

```
    src: "/srv/data/client.key"
    dest: "/etc/pki/tls/private/client.key"
    mode: 0600

- name: Copy client certificate for 802.1X authentication
  copy:
    src: "/srv/data/client.crt"
    dest: "/etc/pki/tls/certs/client.crt"

- name: Copy CA certificate for 802.1X authentication
  copy:
    src: "/srv/data/ca.crt"
    dest: "/etc/pki/ca-trust/source/anchors/ca.crt"

- include_role:
    name: rhel-system-roles.network

  vars:
    network_connections:
      - name: enp1s0
        type: ethernet
        autoconnect: yes
        ip:
          address:
            - 192.0.2.1/24
            - 2001:db8:1::1/64
          gateway4: 192.0.2.254
          gateway6: 2001:db8:1::fffe
          dns:
            - 192.0.2.200
            - 2001:db8:1::ffbb
          dns_search:
            - example.com
        ieee802_1x:
          identity: user_name
          eap: tls
          private_key: "/etc/pki/tls/private/client.key"
          private_key_password: "password"
          client_cert: "/etc/pki/tls/certs/client.crt"
          ca_cert: "/etc/pki/ca-trust/source/anchors/ca.crt"
          domain_suffix_match: example.com
        state: up
```

2. 运行 playbook：

```
# ansible-playbook ~/enable-802.1x.yml
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件

## 8.11. 使用 NETWORK RHEL 系统角色在现有连接上设置默认网关

您可以使用 **network** RHEL 系统角色来设置默认网关。

**重要**

当您运行使用 **network** RHEL 系统角色的剧本时，如果设置的值与剧本中指定的名称不匹配，则系统角色会覆盖具有相同名称的现有的连接配置文件。因此，始终在剧本中指定网络连接配置文件的整个配置，即使 IP 配置已经存在。否则，角色会将这些值重置为默认值。

根据它是否已存在，流程使用如下设置创建或更新 **enp1s0** 连接配置文件：

- 静态 IPv4 地址 - **198.51.100.20**，子网掩码为 **/24**
- 静态 IPv6 地址 - **2001:db8:1::1** 和 **/64** 子网掩码
- IPv4 默认网关 - **198.51.100.254**
- IPv6 默认网关 - **2001:db8:1::fffe**
- IPv4 DNS 服务器 - **198.51.100.200**
- IPv6 DNS 服务器 - **2001:db8:1::ffbb**
- DNS 搜索域 - **example.com**

在 Ansible 控制节点上执行此步骤。

**先决条件**

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。

**流程**

1. 创建包含以下内容的 playbook 文件，如 ***~/ethernet-connection.yml***：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure an Ethernet connection with static IP and default gateway
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        - name: enp1s0
          type: ethernet
          autoconnect: yes
          ip:
            address:
              - 198.51.100.20/24
              - 2001:db8:1::1/64
```

```
    gateway4: 198.51.100.254
    gateway6: 2001:db8:1::fffe
    dns:
      - 198.51.100.200
      - 2001:db8:1::ffbb
    dns_search:
      - example.com
  state: up
```

2. 运行 playbook：

```
# ansible-playbook ~/ethernet-connection.yml
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件

## 8.12. 使用 NETWORK RHEL 系统角色配置一个静态路由

您可以使用 **network** RHEL 系统角色配置静态路由。

**重要**

当您运行使用 **network** RHEL 系统角色的剧本时，如果设置的值与剧本中指定的名称不匹配，则系统角色会覆盖具有相同名称的现有的连接配置文件。因此，始终在剧本中指定网络连接配置文件的整个配置，即使 IP 配置已经存在。否则，角色会将这些值重置为默认值。

根据它是否已存在，流程使用以下设置创建或更新 **enp7s0** 连接配置文件：

- 静态 IPv4 地址 - **192.0.2.1** 和 **/24** 子网掩码
- 静态 IPv6 地址 - **2001:db8:1::1** 和 **/64** 子网掩码
- IPv4 默认网关 - **192.0.2.254**
- IPv6 默认网关 - **2001:db8:1::fffe**
- IPv4 DNS 服务器 - **192.0.2.200**
- IPv6 DNS 服务器 - **2001:db8:1::ffbb**
- DNS 搜索域 - **example.com**
- 静态路由：
  - **198.51.100.0/24**，网关为 **192.0.2.10**
  - **2001:db8:2::/64**，网关为 **2001:db8:1::10**

在 Ansible 控制节点上执行此步骤。

**先决条件**

- 您已准备好控制节点和受管节点

- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。

**流程**

1. 创建包含以下内容的 playbook 文件，如 ***~/add-static-routes.yml*** ：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure an Ethernet connection with static IP and additional routes
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        - name: enp7s0
          type: ethernet
          autoconnect: yes
          ip:
            address:
              - 192.0.2.1/24
              - 2001:db8:1::1/64
            gateway4: 192.0.2.254
            gateway6: 2001:db8:1::fffe
            dns:
              - 192.0.2.200
              - 2001:db8:1::ffbb
            dns_search:
              - example.com
            route:
              - network: 198.51.100.0
                prefix: 24
                gateway: 192.0.2.10
              - network: 2001:db8:2::
                prefix: 64
                gateway: 2001:db8:1::10
          state: up
```

2. 运行 playbook：

```
# ansible-playbook ~/add-static-routes.yml
```

**验证**

1. 在受管节点上：
   a. 显示 IPv4 路由：

```
# ip -4 route
...
198.51.100.0/24 via 192.0.2.10 dev enp7s0
```

b. 显示 IPv6 路由：

```
# ip -6 route
...
2001:db8:2::/64 via 2001:db8:1::10 dev enp7s0 metric 1024 pref medium
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件

## 8.13. 使用 NETWORK RHEL 系统角色配置 ETHTOOL 卸载功能

您可以使用 **network** RHEL 系统角色配置 NetworkManager 连接中的 **ethtool** 功能。

**重要**

当您运行使用 **network** RHEL 系统角色的剧本时，如果设置的值与剧本中指定的名称不匹配，则系统角色会覆盖具有相同名称的现有的连接配置文件。因此，始终在剧本中指定网络连接配置文件的整个配置，例如，即使 IP 配置已经存在。否则，角色会将这些值重置为默认值。

根据它是否已存在，流程使用如下设置创建或更新 **enp1s0** 连接配置文件：

- 静态 **IPv4** 地址 - **198.51.100.20**，子网掩码为 **/24**
- 一个静态 **IPv6** 地址 - **2001:db8:1::1**，子网掩码为 **/64**
- **IPv4** 默认网关 - **198.51.100.254**
- **IPv6** 默认网关 - **2001:db8:1::fffe**
- **IPv4** DNS 服务器 - **198.51.100.200**
- **IPv6** DNS 服务器 - **2001:db8:1::ffbb**
- DNS 搜索域 - **example.com**
- **ethtool** 功能：
  - 通用接收卸载(GRO)：禁用
  - 通用分段卸载(GSO)：启用
  - TX 流控制传输协议(SCTP)分段：禁用

在 Ansible 控制节点上执行此步骤。

**先决条件**

- 您已准备好控制节点和受管节点

- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。

**流程**

1. 创建一个 playbook 文件，如 ***~/configure-ethernet-device-with-ethtool-features.yml***，其内容如下：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
  - name: Configure an Ethernet connection with ethtool features
    include_role:
      name: rhel-system-roles.network

    vars:
      network_connections:
        - name: enp1s0
          type: ethernet
          autoconnect: yes
          ip:
            address:
              - 198.51.100.20/24
              - 2001:db8:1::1/64
            gateway4: 198.51.100.254
            gateway6: 2001:db8:1::fffe
            dns:
              - 198.51.100.200
              - 2001:db8:1::ffbb
            dns_search:
              - example.com
          ethtool:
            features:
              gro: "no"
              gso: "yes"
              tx_sctp_segmentation: "no"
          state: up
```

2. 运行 playbook：

```
# ansible-playbook ~/configure-ethernet-device-with-ethtool-features.yml
```

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件

## 8.14. 网络 RHEL 系统角色的网络状态

**network** RHEL 系统角色支持 playbook 中的状态配置来配置设备。为此，请使用 **network_state** 变量，后面跟上状态配置。

在 playbook 中使用 **network_state** 变量的好处：

- 通过与状态配置结合使用声明方法，您可以配置接口，NetworkManager 会在后台为这些接口创建一个配置集。
- 使用 **network_state** 变量，您可以指定您需要更改的选项，所有其他选项将保持不变。但是，使用 **network_connections** 变量，您必须指定所有设置来更改网络连接配置集。

例如，要使用动态 IP 地址设置创建以太网连接，请在 playbook 中使用以下 **vars** 块：

| 带有状态配置的 playbook | 常规 playbook |
| --- | --- |
| vars:<br>  network_state:<br>  interfaces:<br>  - name: enp7s0<br>  type: ethernet<br>  state: up<br>  ipv4:<br>  enabled: true<br>  auto-dns: true<br>  auto-gateway: true<br>  auto-routes: true<br>  dhcp: true<br>  ipv6:<br>  enabled: true<br>  auto-dns: true<br>  auto-gateway: true<br>  auto-routes: true<br>  autoconf: true<br>  dhcp: true | vars:<br>  network_connections:<br>  - name: enp7s0<br>  interface_name: enp7s0<br>  type: ethernet<br>  autoconnect: yes<br>  ip:<br>  dhcp4: yes<br>  auto6: yes<br>  state: up |

例如，要仅更改您之前创建的动态 IP 地址设置的连接状态，请在 playbook 中使用以下 **vars** 块：

| 带有状态配置的 playbook | 常规 playbook |
| --- | --- |
| vars:<br>  network_state:<br>  interfaces:<br>  - name: enp7s0<br>  type: ethernet<br>  state: down | vars:<br>  network_connections:<br>  - name: enp7s0<br>  interface_name: enp7s0<br>  type: ethernet<br>  autoconnect: yes<br>  ip:<br>  dhcp4: yes<br>  auto6: yes<br>  state: down |

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件

# 第 9 章 使用系统角色配置 FIREWALLD

您可以使用 **firewall** 系统角色一次性在多个客户端上配置 **firewalld** 服务的设置。这个解决方案：

- 提供具有有效输入设置的接口。
- 将所有预期的 **firewalld** 参数保存在一个地方。

在控制节点上运行 **防火墙** 角色后，系统角色会立即向受管节点应用 **firewalld** 参数，并使其在重启后保持不变。

## 9.1. FIREWALL RHEL 系统角色简介

RHEL 系统角色是一组用于 Ansible 自动化实用程序的内容。此内容与 Ansible 自动化工具一起提供了一致的配置界面，来远程管理多个系统。

RHEL 系统角色中的 **rhel-system-roles.firewall** 角色是为 **firewalld** 服务的自动配置而引入的。**rhel-system-roles** 软件包中包含这个系统角色以及参考文档。

要以自动化的方式在一个或多个系统上应用 **firewalld** 参数，请在 playbook 中使用 **firewall** 系统角色变量。playbook 是一个或多个以基于文本的 YAML 格式编写的 play 的列表。

您可以使用清单文件来定义您希望 Ansible 来配置的一组系统。

使用 **firewall** 角色，您可以配置许多不同的 **firewalld** 参数，例如：

- 区。
- 应允许哪些数据包的服务。
- 授权、拒绝或丢弃访问端口的流量。
- 区的端口或端口范围的转发。

**其它资源**

- **/usr/share/doc/rhel-system-roles/firewall/** 目录中的 **README.md** 和 **README.html**
- 使用 playbook
- 如何构建清单

## 9.2. 使用防火墙 RHEL 系统角色重置 FIREWALLD 设置

使用 **firewall** RHEL 系统角色，您可以将 **firewalld** 设置重置为其默认状态。如果在变量列表中添加 **previous:replaced** 参数，则系统角色会删除所有现有用户定义的设置，并将 **firewalld** 重置为默认值。如果将 **previous:replaced** 参数与其他设置相结合，则 **firewall** 角色会在应用新设置前删除所有现有设置。

在 Ansible 控制节点上执行此步骤。

**先决条件**

- 您已准备好控制节点和受管节点

- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户对其具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。

**流程**

1. 创建包含以下内容的 playbook 文件，如 ***~/reset-firewalld.yml*** ：

```
---
- name: Reset firewalld example
  hosts: managed-node-01.example.com
  tasks:
  - name: Reset firewalld
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
        - previous: replaced
```

2. 运行 playbook：

```
# ansible-playbook ~/configuring-a-dmz.yml
```

验证

- 在受管节点上以 **root** 用户身份运行这个命令，以检查所有区域：

```
# firewall-cmd --list-all-zones
```

**其它资源**

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md**
- **ansible-playbook(1)**
- **firewalld(1)**

## 9.3. 将传入的流量从一个本地端口转发到不同的本地端口

使用 **firewall** 角色，您可以远程配置 **firewalld** 参数，使其对多个受管主机有效。

在 Ansible 控制节点上执行此步骤。

**先决条件**

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户对其具有 **sudo** 权限。

- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。

**流程**

1. 创建包含以下内容的 playbook 文件，如 ***~/port_forwarding.yml*** ：

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
  - name: Forward incoming traffic on port 8080 to 443
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
        - { forward_port: 8080/tcp;443;, state: enabled, runtime: true, permanent: true }
```

2. 运行 playbook：

```
# ansible-playbook ~/port_forwarding.yml
```

验证

- 在受管主机上显示 **firewalld** 设置：

```
# firewall-cmd --list-forward-ports
```

**其它资源**

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md**

## 9.4. 使用系统角色配置端口

您可以使用 RHEL **firewall** 系统角色为入站流量在本地防火墙中打开或关闭端口，并使新配置在重启后保留不变。例如，您可以配置默认区域，以允许 HTTPS 服务的传入流量。

在 Ansible 控制节点上执行此步骤。

**先决条件**

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户对其具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。

**流程**

1. 创建包含以下内容的 playbook 文件，如 ***~/opening-a-a-port.yml*** ：

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
  - name: Allow incoming HTTPS traffic to the local host
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
        - port: 443/tcp
          service: http
          state: enabled
          runtime: true
          permanent: true
```

**permanent: true** 选项可使新设置在重启后保持不变。

2. 运行 playbook：

```
# ansible-playbook ~/opening-a-port.yml
```

**验证**

- 在受管节点上，验证与 **HTTPS** 服务关联的 **443/tcp** 端口是否已打开：

```
# firewall-cmd --list-ports
443/tcp
```

**其它资源**

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md**

## 9.5. 使用 FIREWALLD RHEL 系统角色配置 DMZ FIREWALLD 区域

作为系统管理员，您可以使用 **firewall** 系统角色在 **enp1s0** 接口上配置一个 **dmz** 区域，以允许到区域的 **HTTPS** 流量。这样，您可以让外部用户访问您的 web 服务器。

在 Ansible 控制节点上执行此步骤。

**先决条件**

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户对其具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。

**流程**

1. 创建包含以下内容的 playbook 文件，如 ***~/configuring-a-dmz.yml*** ：

```
---
- name: Configure firewalld
  hosts: managed-node-01.example.com
  tasks:
  - name: Creating a DMZ with access to HTTPS port and masquerading for hosts in DMZ
    include_role:
      name: rhel-system-roles.firewall

    vars:
      firewall:
        - zone: dmz
          interface: enp1s0
          service: https
          state: enabled
          runtime: true
          permanent: true
```

2. 运行 playbook：

```
# ansible-playbook ~/configuring-a-dmz.yml
```

**验证**

- 在受管节点上，查看关于 **dmz** 区的详细信息：

```
# firewall-cmd --zone=dmz --list-all
dmz (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp1s0
  sources:
  services: https ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
```

**其它资源**

- **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md**

# 第 10 章 系统角色中的 POSTFIX 角色的变量

**postfix** 角色变量允许用户安装、配置和启动 **postfix** 邮件传输代理 (MTA)。

本节中定义了以下角色变量：

- **postfix_conf** ：它包含所有支持的 **postfix** 配置参数的键/值对。默认情况下，**postfix_conf** 没有值。

```
postfix_conf:
  relayhost: example.com
```

如果您的场景需要删除任何现有配置并在干净 **postfix** 安装之上应用所需的配置，请在 **postfix_conf** 字典中指定 **previous: replaced** 选项：

带有 **previous: replaced** 选项的示例：

```
postfix_conf:
  relayhost: example.com
  previous: replaced
```

- **postfix_check** ：它会确定在启动 **postfix** 前是否执行了检查以验证配置更改。默认值为 true。

例如：

```
postfix_check: true
```

- **postfix_backup** ：它决定是否已创建了配置的一个备份副本。默认情况下，postfix **_backup** 值为 false。

要覆盖以前的备份，请运行以下命令：

```
# *cp /etc/postfix/main.cf /etc/postfix/main.cf.backup*
```

如果 **postfix_backup** 值更改为 **true**，则还必须将 **postfix_backup_multiple** 值设为 false。

例如：

```
postfix_backup: true
postfix_backup_multiple: false
```

- **postfix_backup_multiple** ：它决定角色是否生成配置的时间戳备份副本。

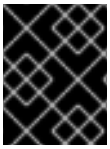
要保留多个备份副本，请运行以下命令：

```
# *cp /etc/postfix/main.cf /etc/postfix/main.cf.$(date -Isec)*
```

默认情况下，postfix **_backup_multiple** 的值为 true。**postfix_backup_multiple:true** 设置将覆盖 **postfix_backup**。如果要使用 **postfix_backup**，您必须设置 **postfix_backup_multiple:false**。

- **postfix_manage_firewall** ：将 **postfix** 角色与 **firewall** 角色集成，以管理端口访问。默认情况下，变量设为 **false**。如果要从 **postfix** 角色自动管理端口访问，请将变量设置为 **true**。

- **postfix_manage_selinux** ：将 **postfix** 角色与 **selinux** 角色集成，以管理端口访问。默认情况下，变量设为 **false**。如果要从 **postfix** 角色自动管理端口访问，请将变量设置为 **true**。



**重要**

不能删除配置参数。在运行 **postfix** 角色之前，请将 **postfix_conf** 设置为所有必需的配置参数，并使用 file 模块删除 **/etc/postfix/main.cf**。

## 10.1. 其他资源

- **/usr/share/doc/rhel-system-roles/postfix/README.md**

# 第 11 章 使用系统角色配置 SELINUX

## 11.1. SELINUX 系统角色简介

RHEL 系统角色是 Ansible 角色和模块的集合,可为远程管理多个 RHEL 系统提供一致的配置界面。**selinux** 系统角色启用以下操作：

- 清理与 SELinux 布尔值、文件上下文、端口和登录相关的本地策略修改。
- 设置 SELinux 策略布尔值、文件上下文、端口和登录。
- 在指定文件或目录中恢复文件上下文。
- 管理 SELinux 模块。

下表提供了 **selinux** 系统角色中提供的输入变量概述。

**表 11.1. SELinux 系统角色变量**

| 角色变量 | 描述 | CLI 的替代方案 |
|---|---|---|
| selinux_policy | 选择保护目标进程或多级别安全保护的策略。 | **/etc/selinux/config**中的 **SELINUXTYPE** |
| selinux_state | 切换 SELinux 模式。 | **/etc/selinux/config** 中的 **setenforce** 和 **SELINUX**. |
| selinux_booleans | 启用和禁用 SELinux 布尔值。 | **setsebool** |
| selinux_fcontexts | 添加或删除 SELinux 文件上下文映射。 | **semanage fcontext** |
| selinux_restore_dirs | 在文件系统树中恢复 SELinux 标签。 | **restorecon -R** |
| selinux_ports | 在端口上设置 SELinux 标签。 | **semanage port** |
| selinux_logins | 将用户设置为 SELinux 用户映射。 | **semanage login** |
| selinux_modules | 安装、启用、禁用或删除 SELinux 模块。 | **semodule** |

**rhel-system-roles** 软件包安装的 **/usr/share/doc/rhel-system-roles/selinux/example-selinux-playbook.yml** 示例 playbook 演示了如何在 enforcing 模式下设置目标策略。playbook 还应用多个本地策略修改，并在 **/tmp/test_dir/** 目录中恢复文件上下文。

有关 **selinux** 角色变量的详细参考，请安装 **rhel-system-roles** 软件包，并参阅 **/usr/share/doc/rhel-system-roles/selinux/** 目录中的 **README.md** 或 **README.html** 文件。

**其他资源**

- RHEL 系统角色简介

## 11.2. 使用 SELINUX 系统角色在多个系统中应用 SELINUX 设置

按照以下步骤，在已验证的 SELinux 设置中准备并应用 Ansible playbook。

**前提条件**

- 有访问一个或多个*受管节点*的权限，这是您要使用 **selinux** 系统角色配置的系统。
- 对 *控制节点* 的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。
  在控制节点上：
  - **ansible-core** 和 **rhel-system-roles** 软件包已安装 。
  - 列出受管节点的清单文件。

**重要**

RHEL 8.0-8.5 提供对基于 Ansible 的自动化需要 Ansible Engine 2.9 的独立 Ansible 存储库的访问权限。Ansible Engine 包含命令行实用程序，如 **ansible**、**ansible-playbook**、连接器（如 **docker** 和 **podman** ）以及许多插件和模块。有关如何获取和安装 Ansible Engine 的详情，请参考 如何下载和安装 Red Hat Ansible Engine 知识库文章。

RHEL 8.6 和 9.0 引入了 Ansible Core（作为 **ansible-core** 软件包提供），其中包含 Ansible 命令行工具、命令以及小型内置 Ansible 插件。RHEL 通过 AppStream 软件仓库提供此软件包，它有一个有限的支持范围。如需更多信息，请参阅 RHEL 9 和 RHEL 8.6 及更新的 AppStream 软件仓库文档中的 Ansible Core 软件包的支持范围。

- 列出受管节点的清单文件。

**流程**

1. 准备您的 playbook。您可以从头开始，或修改作为 **rhel-system-roles** 软件包的一部分安装的示例 playbook：

```
# cp /usr/share/doc/rhel-system-roles/selinux/example-selinux-playbook.yml my-selinux-playbook.yml
# vi my-selinux-playbook.yml
```

2. 更改 playbook 的内容，使其适合您的场景。例如，以下部分确保系统安装并启用 **selinux-local-1.pp** SELinux 模块：

```
selinux_modules:
- { path: "selinux-local-1.pp", priority: "400" }
```

3. 保存更改，然后退出文本编辑器。
4. 在 *host1*、*host2* 和 *host3* 系统上运行您的 playbook：

```
# ansible-playbook -i host1,host2,host3 my-selinux-playbook.yml
```

**其他资源**

- 如需更多信息，请安装 **rhel-system-roles** 软件包，并查看 **/usr/share/doc/rhel-system-roles/selinux/** 和 **/usr/share/ansible/roles/rhel-system-roles.selinux/** 目录。

# 第 12 章 使用 RHEL 系统角色配置日志记录

作为系统管理员，您可以使用 **logging** 系统角色将 RHEL 主机配置为日志服务器，从很多客户端系统收集日志。

## 12.1. 日志记录系统角色

**使用日志记录系统角色**，您可以在本地和远程主机上部署日志配置。

要在一个或多个系统中应用 **logging** 系统角色，您可以在 *playbook* 中定义日志配置。playbook 是一个或多个 play 的列表。playbook 是人类可读的，它们采用 YAML 格式编写。如需有关 playbook 的更多信息，请参阅 Ansible 文档中的 使用 playbook。

您要根据 playbook 配置的系统集合在 *清单文件中* 定义。如需有关创建和使用清单的更多信息，请参阅 Ansible 文档中的如何构建您的清单。

日志记录解决方案提供多种读取日志和多个日志记录输出的方法。

例如，日志记录系统可接受以下输入：

- 本地文件、
- **systemd/journal**,
- 网络中的另一个日志记录系统。

另外，日志记录系统还可有以下输出：

- 日志存储在 **/var/log** 目录中的本地文件中，
- 日志被发送到 Elasticsearch ，
- 日志被转发到另一个日志系统。

使用 **logging** 系统角色，您可以组合输入和输出以适合您的场景。例如，您可以配置一个日志解决方案，将来自 **日志** 的输入存储在本地文件中，而从文件读取的输入则被转发到另一个日志系统，并存储在本地日志文件中。

## 12.2. LOGGING 系统角色参数

在 **logging** 系统角色 playbook 中，您可以在 **logging_inputs** 参数中定义输入，在 **logging_outputs** 参数中定义输出，以及在 **logging_flows** 参数中定义输入和输出之间的关系。**logging** 系统角色使用附加选项处理这些变量来配置日志记录系统。您还可以启用加密或自动端口管理。

**注意**

目前，**logging** 系统角色中唯一可用的日志记录系统是 **Rsyslog**。

- **logging_inputs** ：日志记录解决方案的输入列表。
  - **名称** ：输入的唯一名称。用于 **logging_flows**：输入列表以及生成的 **config** 文件名称的一部分。
  - **类型** ：输入元素的类型。type 指定与 **roles/rsyslog/{tasks,vars}/inputs/** 中的目录名称相对应的任务类型。

- **基本** ：配置 **systemd** 日志或 **unix** 套接字输入。
  - **kernel_message** ：如果设为 **true**，则加载 **imklog**。默认值为 **false**。
  - **使用_imuxsock** ：使用 **imuxsock** 而不是 **imjournal**。默认值为 **false**。
  - **ratelimit_burst** ：可在 **ratelimit_interval** 内发送的最大消息数。如果 **use_imuxsock** 为 false，则默认为 **20000**。如果 **use_imuxsock** 为 true，则默认为 **200**。
  - **ratelimit_interval**: 用于评估 **ratelimit_burst** 的间隔。如果 **use_imuxsock** 为 false，则默认为 600 秒。如果 **use_imuxsock** 为 true，则默认为 0。0 表示关闭速率限制。
  - **persist_state_interval**: Journal 状态保留每个 **值** 的消息。默认为 **10**。仅在 **use_imuxsock** 为 false 时有效。
- **文件** ：输入配置本地文件输入。
- **远程** ：输入通过网络配置来自其他日志记录系统的输入。

- **state**：配置文件的状态。**present** 或 **absent**。默认为 **present**。

- **logging_outputs** ：日志记录解决方案的输出列表。
  - **文件** ：输出将输出配置为本地文件。
  - **转发**: 输出配置输出到另一个日志记录系统。
  - **remote_files** ：输出将输出配置为另一个日志记录系统到本地文件。
- **logging_flows** ：定义 **logging_inputs** 和 **logging_outputs** 之间关系的流列表。**logging_flows** 变量具有以下键：
  - **名称** ：流的唯一名称
  - **输入** ： **logging_inputs** 名称值列表
  - **输出** ： **logging_outputs** 名称值列表。
- **logging_manage_firewall** ：如果设为 **true**，变量将使用 **firewall** 角色自动管理来自 **logging** 角色的端口访问。
- **logging_manage_selinux** ：如果设为 **true**，变量使用 **selinux** 角色自动管理 **logging** 角色的端口访问。

**其他资源**

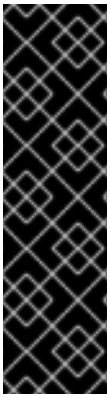
- 与 **rhel-system-roles** 软件包一起安装的文档在 **/usr/share/ansible/roles/rhel-system-roles.logging/README.html**

## 12.3. 应用 **本地** 日志记录系统角色

准备并应用 Ansible playbook，以便对一组独立的机器配置日志记录解决方案。每台机器在本地记录日志。

**先决条件**

- 对一个或多个 *受管节点* 的访问和权限，这些节点是您要使用 **logging** 系统角色配置的系统。
- 对 *控制节点* 的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。
  在控制节点上：
  - **ansible-core** 和 **rhel-system-roles** 软件包已安装 。

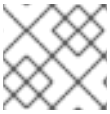


**重要**

RHEL 8.0-8.5 提供对基于 Ansible 的自动化需要 Ansible Engine 2.9 的独立 Ansible 存储库的访问权限。Ansible Engine 包含命令行实用程序，如 **ansible**、**ansible-playbook**、连接器（如 **docker** 和 **podman** ）以及许多插件和模块。有关如何获取和安装 Ansible Engine 的详情，请参考 如何下载和安装 Red Hat Ansible Engine 知识库文章。

RHEL 8.6 和 9.0 引入了 Ansible Core（作为 **ansible-core** 软件包提供），其中包含 Ansible 命令行工具、命令以及小型内置 Ansible 插件。RHEL 通过 AppStream 软件仓库提供此软件包，它有一个有限的支持范围。如需更多信息，请参阅 RHEL 9 和 RHEL 8.6 及更新的 AppStream 软件仓库文档中的 Ansible Core 软件包的支持范围。

- 列出受管节点的清单文件。



**注意**

您不必安装 **rsyslog** 软件包，因为系统管理员在部署时会安装 **rsyslog**。

**流程**

1. 创建定义所需角色的 playbook:
   a. 创建新 YAML 文件，并在文本编辑器中打开，例如：

```
# vi logging-playbook.yml
```

   b. 插入以下内容：

```
---
- name: Deploying basics input and implicit files output
  hosts: all
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: system_input
        type: basics
    logging_outputs:
      - name: files_output
        type: files
    logging_flows:
      - name: flow1
        inputs: [system_input]
        outputs: [files_output]
```

2. 对特定清单运行 playbook:

```
# ansible-playbook -i inventory-file /path/to/file/logging-playbook.yml
```

其中：

- ***inventory-file*** 是清单文件。
- ***logging-playbook.yml*** 是您使用的 playbook。

**验证**

1. 测试 **/etc/rsyslog.conf** 文件的语法：

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run...
rsyslogd: End of config validation run. Bye.
```

2. 验证系统是否向日志发送信息：
   a. 发送测试信息：

```
# logger test
```

   b. 查看 **/var/log/messages** 日志，例如：

```
# cat /var/log/messages
Aug  5 13:48:31 hostname root[6778]: test
```

其中 *'hostname'* 是客户端系统的主机名。请注意，该日志包含输入 logger 命令的用户的用户名，本例中为 **root**。

## 12.4. 过滤本地日志记录系统角色中的 LOGGING

您可以部署一个日志解决方案，该方案基于 **rsyslog** 属性的过滤器过滤日志。

**先决条件**

- 对一个或多个 *受管节点* 的访问和权限，这些节点是您要使用 **logging** 系统角色配置的系统。
- 对 *控制节点* 的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。
  在控制节点上：
  - 安装了 Red Hat Ansible Core
  - **rhel-system-roles** 软件包已安装
  - 列出受管节点的清单文件。

**注意**

您不必安装 **rsyslog** 软件包，因为系统管理员在部署时会安装 **rsyslog**。

**流程**

1. 使用以下内容创建新的 ***playbook.yml*** 文件：

```
---
- name: Deploying files input and configured files output
  hosts: all
  roles:
    - linux-system-roles.logging
  vars:
    logging_inputs:
      - name: files_input
        type: basics
    logging_outputs:
      - name: files_output0
        type: files
        property: msg
        property_op: contains
        property_value: error
        path: /var/log/errors.log
      - name: files_output1
        type: files
        property: msg
        property_op: "!contains"
        property_value: error
        path: /var/log/others.log
    logging_flows:
      - name: flow0
        inputs: [files_input]
        outputs: [files_output0, files_output1]
```

使用这个配置，包含 ***error*** 字符串的所有消息都会记录在 **/var/log/errors.log** 中，所有其他消息都记录在 **/var/log/others.log** 中。

您可以将 ***error*** 属性值替换为您要过滤的字符串。

您可以根据您的偏好修改变量。

2. 可选：验证 playbook 语法。

```
# ansible-playbook --syntax-check playbook.yml
```

3. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

**验证**

1. 测试 **/etc/rsyslog.conf** 文件的语法：

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run...
rsyslogd: End of config validation run. Bye.
```

2. 验证系统是否向日志发送包含 **error** 字符串的信息：

   a. 发送测试信息：

```
# logger error
```

b. 查看 **/var/log/errors.log** 日志，例如：

```
# cat /var/log/errors.log
Aug  5 13:48:31 hostname root[6778]: error
```

其中 ***hostname*** 是客户端系统的主机名。请注意，该日志包含输入 logger 命令的用户的用户名，本例中为 **root**。

**其他资源**

- 与 **rhel-system-roles** 软件包一起安装的文档在 **/usr/share/ansible/roles/rhel-system-roles.logging/README.html**

## 12.5. 使用 LOGGING 系统角色应用远程日志解决方案

按照以下步骤准备并应用 Red Hat Ansible Core playbook 来配置远程日志记录解决方案。在本 playbook 中，一个或多个客户端从 **systemd-journal** 获取日志，并将它们转发到远程服务器。服务器从 **remote_rsyslog** 和 **remote_files** 接收远程输入，并将日志输出到由远程主机名命名的目录中的本地文件。

**先决条件**

- 对一个或多个 *受管节点* 的访问和权限，这些节点是您要使用 **logging** 系统角色配置的系统。
- 对 *控制节点* 的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。
  在控制节点上：
  - **ansible-core** 和 **rhel-system-roles** 软件包已安装 。
  - 列出受管节点的清单文件。

**注意**

您不必安装 **rsyslog** 软件包，因为系统管理员在部署时会安装 **rsyslog**。

**流程**

1. 创建定义所需角色的 playbook:

   a. 创建新 YAML 文件，并在文本编辑器中打开，例如：

```
# vi logging-playbook.yml
```

   b. 将以下内容插入到文件中：

```
---
- name: Deploying remote input and remote_files output
  hosts: server
  roles:
    - rhel-system-roles.logging
  vars:
```

```
    logging_inputs:
      - name: remote_udp_input
        type: remote
        udp_ports: [ 601 ]
      - name: remote_tcp_input
        type: remote
        tcp_ports: [ 601 ]
    logging_outputs:
      - name: remote_files_output
        type: remote_files
    logging_flows:
      - name: flow_0
        inputs: [remote_udp_input, remote_tcp_input]
        outputs: [remote_files_output]

- name: Deploying basics input and forwards output
  hosts: clients
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: basic_input
        type: basics
    logging_outputs:
      - name: forward_output0
        type: forwards
        severity: info
        target: _host1.example.com_
        udp_port: 601
      - name: forward_output1
        type: forwards
        facility: mail
        target: _host1.example.com_
        tcp_port: 601
    logging_flows:
      - name: flows0
        inputs: [basic_input]
        outputs: [forward_output0, forward_output1]

[basic_input]
[forward_output0, forward_output1]
```

其中 ***host1.example.com*** 是日志服务器。

**注意**

您可以修改 playbook 中的参数以符合您的需要。

**警告**

日志解决方案只适用于在服务器或者客户端系统的 SELinux 策略中定义的端口并在防火墙中打开。默认 SELinux 策略包括端口 601、514、6514、10514 和 20514。要使用其他端口，请修改客户端和服务器系统上的 SELinux 策略。

2. 创建列出您的服务器和客户端的清单文件：

   a. 创建新文件并在文本编辑器中打开该文件，例如：

   ```
   # vi inventory.ini
   ```

   b. 将以下内容插入到清单文件中：

   ```
   [servers]
   server ansible_host=host1.example.com
   [clients]
   client ansible_host=host2.example.com
   ```

   其中：

   - ***host1.example.com*** 是日志服务器。
   - ***host2.example.com*** 是日志客户端。

3. 对清单运行 playbook。

   ```
   # ansible-playbook -i /path/to/file/inventory.ini /path/to/file/_logging-playbook.yml
   ```

   其中：

   - ***inventory.ini*** 是清单文件。
   - ***logging-playbook.yml*** 是您创建的 playbook。

**验证**

1. 在客户端和服务器系统上测试 **/etc/rsyslog.conf** 文件的语法：

   ```
   # rsyslogd -N 1
   rsyslogd: version 8.1911.0-6.el8, config validation run (level 1), master config
   /etc/rsyslog.conf
   rsyslogd: End of config validation run. Bye.
   ```

2. 验证客户端系统向服务器发送信息：

   a. 在客户端系统中发送测试信息：

   ```
   # logger test
   ```

b. 在服务器系统上，查看 **/var/log/messages** 日志，例如：

```
# cat /var/log/messages
Aug  5 13:48:31 host2.example.com root[6778]: test
```

其中 ***host2.example.com*** 是客户端系统的主机名。请注意，该日志包含输入 logger 命令的用户的用户名，本例中为 **root**。

**其他资源**

- 准备控制节点和受管节点以使用 RHEL 系统角色
- 与 **rhel-system-roles** 软件包一起安装的文档在 **/usr/share/ansible/roles/rhel-system-roles.logging/README.html**
- RHEL 系统角色 知识库文章

## 12.6. 使用带有 TLS 的 LOGGING 系统角色

传输层安全性(TLS)是一种加密协议，旨在在计算机网络上安全地进行通信。

作为管理员，您可以使用 **logging** RHEL 系统角色使用红帽 Ansible Automation Platform 配置日志的安全传输。

### 12.6.1. 配置带有 TLS 的客户端日志

您可以使用 **logging** 系统角色在 RHEL 系统上配置日志，这些日志记录在本地计算机上，并通过运行 Ansible playbook 将日志转发到带有 TLS 的远程日志系统。

此流程对 Ansible 清单中所有客户端组中的主机配置 TLS。TLS 对信息的传输进行加密，确保日志在网络安全传输。

**先决条件**

- 您有在要配置 TLS 的受管节点上运行 playbook 的权限。
- 受管节点列在控制节点上的清单文件中。
- **ansible** 和 **rhel-system-roles** 软件包已安装在控制节点上。

**流程**

1. 使用以下内容创建一个 ***playbook.yml*** 文件：

```
---
- name: Deploying files input and forwards output with certs
  hosts: clients
  roles:
    - rhel-system-roles.logging
  vars:
    logging_pki_files:
      - ca_cert_src: /local/path/to/ca_cert.pem
        cert_src: /local/path/to/cert.pem
        private_key_src: /local/path/to/key.pem
```

```
logging_inputs:
  - name: input_name
    type: files
    input_log_path: /var/log/containers/*.log
logging_outputs:
  - name: output_name
    type: forwards
    target: your_target_host
    tcp_port: 514
    tls: true
    pki_authmode: x509/name
    permitted_server: 'server.example.com'
logging_flows:
  - name: flow_name
    inputs: [input_name]
    outputs: [output_name]
```

playbook 使用以下参数：

**logging_pki_files**

使用这个参数您可以配置 TLS，并且必须传递 **ca_cert_src**、**cert_src** 和 **private_key_src** 参数。

**ca_cert**

表示 CA 证书的路径。默认路径为 **/etc/pki/tls/certs/ca.pem**，文件名由用户设置。

**cert**

表示证书的路径。默认路径为 **/etc/pki/tls/certs/server-cert.pem**，文件名由用户设置。

**private_key**

表示私钥的路径。默认路径为 **/etc/pki/tls/private/server-key.pem**，文件名由用户设置。

**ca_cert_src**

代表复制到目标主机的本地 CA 证书文件路径。如果指定了 **ca_cert**，则会将其复制到该位置。

**cert_src**

代表复制到目标主机的本地证书文件路径。如果指定了 **cert**，则会将其复制到该位置。

**private_key_src**

表示复制到目标主机的本地密钥文件的路径。如果指定了 **private_key**，则会将其复制到该位置。

**tls**

使用此参数可确保在网络上安全地传输日志。如果不想要安全包装程序，您可设置 **tls: true**。

2. 验证 playbook 语法：

```
# ansible-playbook --syntax-check playbook.yml
```

3. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory_file playbook.yml
```

### 12.6.2. 配置带有 TLS 的服务器日志

您可以使用 **logging** 系统角色在 RHEL 系统中将日志配置为服务器，并通过运行 Ansible playbook 从带有 TLS 的远程日志系统接收日志。

此流程对 Ansible 清单中主机组中的所有主机配置 TLS。

**先决条件**

- 您有在要配置 TLS 的受管节点上运行 playbook 的权限。
- 受管节点列在控制节点上的清单文件中。
- **ansible** 和 **rhel-system-roles** 软件包已安装在控制节点上。

**流程**

1. 使用以下内容创建一个 ***playbook.yml*** 文件：

```
---
- name: Deploying remote input and remote_files output with certs
  hosts: server
  roles:
    - rhel-system-roles.logging
  vars:
    logging_pki_files:
      - ca_cert_src: /local/path/to/ca_cert.pem
        cert_src: /local/path/to/cert.pem
        private_key_src: /local/path/to/key.pem
    logging_inputs:
      - name: input_name
        type: remote
        tcp_ports: 514
        tls: true
        permitted_clients: ['clients.example.com']
    logging_outputs:
      - name: output_name
        type: remote_files
        remote_log_path: /var/log/remote/%FROMHOST%/%PROGRAMNAME:::secpath-
replace%.log
        async_writing: true
        client_count: 20
        io_buffer_size: 8192
    logging_flows:
      - name: flow_name
        inputs: [input_name]
        outputs: [output_name]
```

playbook 使用以下参数：

**logging_pki_files**

使用这个参数您可以配置 TLS，并且必须传递 **ca_cert_src**、**cert_src** 和 **private_key_src** 参数。

**ca_cert**

表示 CA 证书的路径。默认路径为 **/etc/pki/tls/certs/ca.pem**，文件名由用户设置。

**cert**

表示证书的路径。默认路径为 **/etc/pki/tls/certs/server-cert.pem**，文件名由用户设置。

**private_key**

表示私钥的路径。默认路径为 **/etc/pki/tls/private/server-key.pem**，文件名由用户设置。

**ca_cert_src**

代表复制到目标主机的本地 CA 证书文件路径。如果指定了 **ca_cert**，则会将其复制到该位置。

**cert_src**

代表复制到目标主机的本地证书文件路径。如果指定了 **cert**，则会将其复制到该位置。

**private_key_src**

表示复制到目标主机的本地密钥文件的路径。如果指定了 **private_key**，则会将其复制到该位置。

**tls**

使用此参数可确保在网络上安全地传输日志。如果不想要安全包装程序，您可设置 **tls: true**。

2. 验证 playbook 语法：

```
# ansible-playbook --syntax-check playbook.yml
```

3. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory_file playbook.yml
```

## 12.7. 使用带有 RELP 的 LOGGING 系统角色

可靠的事件日志协议(RELP)是一种通过 TCP 网络记录数据和消息的网络协议。它确保了事件消息的可靠传递，您可以在不容许任何消息丢失的环境中使用它。

RELP 发送者以命令的形式传输日志条目，接收者在处理后确认这些条目。为确保一致性，RELP 将事务数保存到传输的命令中，以便进行任何类型的消息恢复。

您可以考虑在 RELP 客户端和 RELP Server 间的远程日志系统。RELP 客户端将日志传送给远程日志系统，RELP 服务器接收由远程日志系统发送的所有日志。

管理员可以使用 **logging** 系统角色将日志系统配置为可靠地发送和接收日志条目。

### 12.7.1. 配置带有 RELP 的客户端日志

您可以使用 **logging** 系统角色在 RHEL 系统中配置日志，这些日志记录在本地机器上，并通过运行 Ansible playbook 来将日志传送到带有 RELP 的远程日志系统。

此流程对 Ansible 清单中 **客户端** 组中的所有主机配置 RELP。RELP 配置使用传输层安全(TLS)来加密消息传输，保证日志在网络上安全传输。

**先决条件**

- 您有在要配置 RELP 的受管节点上运行 playbook 的权限。
- 受管节点列在控制节点上的清单文件中。
- **ansible** 和 **rhel-system-roles** 软件包已安装在控制节点上。

**流程**

1. 使用以下内容创建一个 ***playbook.yml*** 文件：

```
---
- name: Deploying basic input and relp output
  hosts: clients
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: basic_input
        type: basics
    logging_outputs:
      - name: relp_client
        type: relp
        target: _logging.server.com_
        port: 20514
        tls: true
        ca_cert: _/etc/pki/tls/certs/ca.pem_
        cert: _/etc/pki/tls/certs/client-cert.pem_
        private_key: _/etc/pki/tls/private/client-key.pem_
        pki_authmode: name
        permitted_servers:
          - '*.server.example.com'
    logging_flows:
      - name: _example_flow_
        inputs: [basic_input]
        outputs: [relp_client]
```

playbook 使用以下设置：

- **target**: 这是一个必需的参数，用于指定运行远程日志系统的主机名。
- **端口** ：显示远程日志记录系统正在侦听的端口号。
- **TLS** ：确保通过网络安全地传输日志。如果您不想要安全打包程序，可以将 **tls** 变量设置为 **false**。在与 RELP 工作时，默认的 **tls** 参数被设置为 true，且需要密钥/证书和 triplets {**ca_cert**、**cert**、**private_key**} 和/或 {**ca_cert_src**,**cert_src**,**private_key_src**}。
  - 如果设置了 {**ca_cert_src**, **cert_src** ,**private_key_src**} triplet ， 默认位置 **/etc/pki/tls/certs** 和 **/etc/pki/tls/private** 被用作受管节点上的目的地，来从控制节点传输文件。在这种情况下，文件名与 triplet 中的原始名称相同
  - 如果设置了 {**ca_cert** ， **cert** ， **private_key**} triplet，则在日志配置之前，文件应位于默认路径上。
  - 如果同时设置了 triplet，则会将文件从本地路径从控制节点传输到受管节点的特定路径。
- **ca_cert** ：代表 CA 证书的路径。默认路径为 **/etc/pki/tls/certs/ca.pem**，文件名由用户设置。
- **认证** ：代表证书的路径。默认路径为 **/etc/pki/tls/certs/server-cert.pem**，文件名由用户设置。
- **private_key** ：代表私钥的路径。默认路径为 **/etc/pki/tls/private/server-key.pem**，文件名由用户设置。

- **ca_cert_src**:代表本地 CA 证书文件路径，该文件路径被复制到目标主机。如果指定了 ca_cert，则会将其复制到该位置。
- **cert_src**: 代表复制到目标主机的本地证书文件路径。如果指定了 cert，则会将其复制到该位置。
- **private_key_src**: 代表复制到目标主机的本地密钥文件路径。如果指定了 private_key，则会将其复制到该位置。
- **pki_authmode** ：接受身份验证模式作为**名称**或**指纹**。
- **permitted_servers**：日志客户端允许通过 TLS 连接和发送日志的服务器列表。
- **输入** ：日志记录输入字典列表。
- **输出** ：日志输出字典列表。

2. 可选：验证 playbook 语法。

```
# ansible-playbook --syntax-check playbook.yml
```

3. 运行 playbook：

```
# ansible-playbook -i inventory_file playbook.yml
```

### 12.7.2. 配置带有 RELP 的服务器日志

您可以使用 **logging** 系统角色将 RHEL 系统中的日志配置为服务器，并通过运行 Ansible playbook 从带有 RELP 的远程日志系统接收日志。

此流程对 Ansible 清单中 **服务器** 组中的所有主机配置 RELP。RELP 配置使用 TLS 加密消息传输，以保证在网络上安全地传输日志。

**先决条件**

- 您有在要配置 RELP 的受管节点上运行 playbook 的权限。
- 受管节点列在控制节点上的清单文件中。
- **ansible** 和 **rhel-system-roles** 软件包已安装在控制节点上。

**流程**

1. 使用以下内容创建一个 ***playbook.yml*** 文件：

```
---
- name: Deploying remote input and remote_files output
  hosts: server
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: relp_server
        type: relp
        port: 20514
```

```
      tls: true
      ca_cert: _/etc/pki/tls/certs/ca.pem_
      cert: _/etc/pki/tls/certs/server-cert.pem_
      private_key: _/etc/pki/tls/private/server-key.pem_
      pki_authmode: name
      permitted_clients:
        - '_*example.client.com_'
  logging_outputs:
    - name: _remote_files_output_
      type: _remote_files_
  logging_flows:
    - name: _example_flow_
      inputs: _relp_server_
      outputs: _remote_files_output_
```

playbook 使用以下设置：

- **端口** ：显示远程日志记录系统正在侦听的端口号。
- **TLS** ：确保通过网络安全地传输日志。如果您不想要安全打包程序，可以将 **tls** 变量设置为 **false**。在与 RELP 工作时，默认的 **tls** 参数被设置为 true，且需要密钥/证书和 triplets {**ca_cert**、**cert**、**private_key**} 和/或 {**ca_cert_src**,**cert_src**,**private_key_src**}。
  - 如果设置了 {**ca_cert_src**, **cert_src** ,**private_key_src**} triplet ，默认位置 **/etc/pki/tls/certs** 和 **/etc/pki/tls/private** 被用作受管节点上的目的地，来从控制节点传输文件。在这种情况下，文件名与 triplet 中的原始名称相同
  - 如果设置了 {**ca_cert** ， **cert** ， **private_key**} triplet，则在日志配置之前，文件应位于默认路径上。
  - 如果同时设置了 triplet，则会将文件从本地路径从控制节点传输到受管节点的特定路径。
- **ca_cert** ：代表 CA 证书的路径。默认路径为 **/etc/pki/tls/certs/ca.pem**，文件名由用户设置。
- **认证** ：代表证书的路径。默认路径为 **/etc/pki/tls/certs/server-cert.pem**，文件名由用户设置。
- **private_key** ：代表私钥的路径。默认路径为 **/etc/pki/tls/private/server-key.pem**，文件名由用户设置。
- **ca_cert_src**:代表本地 CA 证书文件路径，该文件路径被复制到目标主机。如果指定了 ca_cert，则会将其复制到该位置。
- **cert_src**: 代表复制到目标主机的本地证书文件路径。如果指定了 cert ，则会将其复制到该位置。
- **private_key_src**: 代表复制到目标主机的本地密钥文件路径。如果指定了 private_key，则会将其复制到该位置。
- **pki_authmode** ：接受身份验证模式作为**名称**或**指纹**。
- **permitted_clients**：日志记录服务器允许通过 TLS 连接和发送日志的客户端列表。
- **输入** ：日志记录输入字典列表。
- **输出** ：日志输出字典列表。

2. 可选：验证 playbook 语法。

```
# ansible-playbook --syntax-check playbook.yml
```

3. 运行 playbook：

```
# ansible-playbook -i inventory_file playbook.yml
```

## 12.8. 其他资源

- 准备控制节点和受管节点以使用 RHEL 系统角色
- 随 **rhel-system-roles** 软件包安装在 **/usr/share/ansible/roles/rhel-system-roles.logging/README.html** 中的文档。
- RHEL 系统角色
- **ansible-playbook(1)** 手册页。

# 第 13 章 使用 JOURNALD RHEL 系统角色配置 SYSTEMD 日志

使用 **journald** 系统角色，您可以自动化 **systemd** 日志，并使用 Red Hat Ansible Automation Platform 配置持久性日志记录。

## 13.1. JOURNALD RHEL 系统角色的变量

**journald** 系统角色提供一组自定义 **journald** 日志记录服务行为的变量。角色包括以下变量：

| 角色变量 | Description |
| --- | --- |
| **journald_persistent** | 使用此布尔值变量来配置 **journald**，以将日志文件存储在磁盘上的 **/var/log/journal/** 目录中。当您将此变量设置为 **true** 时，日志被存储在磁盘上，否则存储在易失性内存中。默认值为 **false**。 |
| **journald_max_disk_size** | 使用此变量指定日志文件可在磁盘上占用的最大大小（以 MB 为单位）。请参阅 **journald.conf (5)** 手册页中描述的默认大小计算。 |
| **journald_max_files** | 使用此变量指定在遵守日志的 **journal_max_disk_size** 设置时要保留的日志文件的最大数量。 |
| **journald_max_file_size** | 使用此变量指定单个日志文件的最大大小（以 MB 为单位）。 |
| **journald_per_user** | 使用此布尔值变量配置 **journald**，以为每个用户分开保持日志数据。默认值为 **true**，非特权用户可以从自己的用户服务中读取系统日志。请注意，只有在 **journald_persistent** 变量设置为 **true** 时，每个用户日志文件才可用。 |
| **journald_compression** | 使用此布尔值变量将压缩应用到大于默认 512 字节的 **journald** 数据对象。默认值为 **true**。 |
| **journald_sync_interval** | 使用此变量指定 **journald** 将当前使用的日志文件同步到磁盘的时间（以分钟为单位）。默认情况下，该角色不会更改当前值。 |

**其他资源**

- **journald.conf (5)** 手册页。

## 13.2. 使用 JOURNALD 系统角色配置持久性日志记录

作为系统管理员，您可以使用 **journald** 系统角色配置持久性日志记录。以下示例演示了如何在 playbook 中设置 **journald** 系统角色变量以达到以下目标：

- 配置持久性日志记录

- 为日志文件指定最大磁盘空间大小
- 配置 **journald** 以为每个用户单独保留日志数据
- 定义同步间隔

**先决条件**

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 要在其上运行此 playbook 的受管节点或受管节点组列在 Ansible 清单文件中。

**流程**

1. 创建包含以下内容的 **playbook.yml** 文件：

```
---
- hosts: all
  vars:
    journald_persistent: true
    journald_max_disk_size: 2048
    journald_per_user: true
    journald_sync_interval: 1
  roles:
    - linux-system-roles.journald
---
```

因此，**journald** 服务会将日志在磁盘上永久保存到 2048 MB 的最大大小，并为每个用户单独保留日志数据。同步每分钟发生。

2. 可选：验证 playbook 语法。

```
# ansible-playbook --syntax-check playbook.yml -i inventory_file
```

3. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

## 13.3. 其他资源

- **journald.conf (5)** 手册页
- **ansible-playbook (1)** 手册页

# 第 14 章 使用 SSH 和 SSHD RHEL 系统角色配置安全通信

作为管理员，您可以使用 **sshd** 系统角色来配置 SSH 服务器，使用 **ssh** 系统角色来通过Red Hat Ansible Automation Platform 在任意数量的 RHEL 系统上同时配置 SSH 客户端。

## 14.1. SSH 服务器系统角色变量

在 **sshd** 系统角色 playbook 中，您可以根据您的首选项和限制定义 SSH 配置文件的参数。

如果您没有配置这些变量，则系统角色会生成与 RHEL 默认值匹配的 **sshd_config** 文件。

在所有情况下，布尔值在 **sshd** 配置中都正确呈现为 **yes** 和 **no**。您可以使用 list 来定义多行配置项。例如：

```
sshd_ListenAddress:
  - 0.0.0.0
  - '::'
```

呈现为：

```
ListenAddress 0.0.0.0
ListenAddress ::
```

**sshd 系统角色的变量**

**sshd_enable**

如果设置为 **False**，则角色将被完全禁用。默认值为 **True**。

**sshd_skip_defaults**

如果设置为 **True**，则系统角色不会应用默认值。相反，您可以使用 **sshd** dict 或 **sshd_Key** 变量来指定完整的配置默认值集合。默认值为 **False**。

**sshd_manage_service**

如果设置为 **False**，则服务不会被管理，这意味着它不会在引导时启用，也不会启动或重新加载。除非在容器内或 AIX 中运行，否则默认为 **True**，因为 Ansible 服务模块目前不支持对 AIX 的 **启用** 。

**sshd_allow_reload**

如果设置为 **False**，则**sshd** 不会在配置更改后重新加载。这可帮助进行故障排除。要应用更改后的配置，请手动重新加载 **sshd**。默认为与 **sshd_manage_service** 相同的值，但 AIX 除外，其中 **sshd_manage_service** 默认为 **False**，但 **sshd_allow_reload** 默认为 **True**。

**sshd_install_service**

如果设置为 **True**，该角色将为 **sshd** 服务安装服务文件。这会覆盖操作系统中提供的文件。除非您要配置第二个实例，否则不要设置为 **True**，您也可以更改 **sshd_service** 变量，。默认值为 **False**。
该角色使用以下变量指向的文件作为模板：

```
sshd_service_template_service (default: templates/sshd.service.j2)
sshd_service_template_at_service (default: templates/sshd@.service.j2)
sshd_service_template_socket (default: templates/sshd.socket.j2)
```

**sshd_service**

此变量更改 **sshd** 服务名称，这对于配置第二个 **sshd** 服务实例非常有用。

**sshd**

包含配置的字典。例如：

```
sshd:
  Compression: yes
  ListenAddress:
    - 0.0.0.0
```

**sshd_*OptionName***

您可以使用由 **sshd_** 前缀和选项名称而不是 dict 组成的简单变量来定义选项。简单的变量覆盖 **sshd** 字典中的值。例如：

```
sshd_Compression: no
```

**sshd_match 和 sshd_match_1 到 sshd_match_9**

字典列表或 只是匹配部分的字典。请注意，这些变量不会覆盖 **sshd** 字典中定义的匹配块。所有源都会反映在生成的配置文件中。

### sshd 系统角色的二级变量

您可以使用这些变量来覆盖与每个支持的平台对应的默认值。

**sshd_packages**

您可以使用此变量来覆盖安装的软件包的默认列表。

**sshd_config_owner、sshd_config_group 和 sshd_config_mode**

您可以使用这些变量为该角色生成的 **openssh** 配置文件设置所有权和权限。

**sshd_config_file**

此角色保存生成的 **openssh** 服务器配置的路径。

**sshd_config_namespace**

此变量的默认值为 null，这意味着角色定义配置文件的整个内容，包括系统默认值。或者，您也可以使用此变量从其他角色或从不支随时可访问目录的系统上的单个 playbook 中的多个位置调用此角色。**sshd_skip_defaults** 变量将被忽略，本例中没有使用系统默认值。
设置此变量时，角色会将您指定的配置放置在给定命名空间下的现有配置段中。如果您的场景需要多次应用角色，您需要为每个应用程序选择不同的命名空间。

**注意**

**openssh** 配置文件的限制仍然适用。例如，对大多数配置选项，只有配置文件中指定的第一个选项有效。

从技术上讲，角色会将段放在"Match all"块中，除非它们包含其他匹配块，以确保无论现有配置文件中之前匹配的块如何都将应用它们。这允许从不同角色调用中配置任何不冲突的选项。

**sshd_binary**

**openssh** 的 **sshd** 可执行文件的路径。

**sshd_service**

**sshd** 服务的名称。默认情况下，此变量包含目标平台所使用的 **sshd** 服务的名称。当角色使用 **sshd_install_service** 变量时，您还可以使用它来设置自定义 **sshd** 服务的名称。

**sshd_verify_hostkeys**

默认值为 **auto**。当设置为 **auto** 时，这将列出生成的配置文件中存在的所有主机密钥，并生成所有不存在的路径。此外，权限和文件所有者被设置为默认值。如果该角色用于部署阶段来确保服务能够在第一次尝试时启动，则这非常有用。若要禁用此检查，可将此变量设置为空列表 **[]**。

**sshd_hostkey_owner,sshd_hostkey_group,sshd_hostkey_mode**

使用这些变量来设置 **sshd_verify_hostkeys** 的主机密钥的所有权和权限。

**sshd_sysconfig**

在基于 RHEL 的系统上，这个变量配置 **sshd** 服务的其它详细信息。如果设置为 **true**，则此角色还会根据以下配置来管理 **/etc/sysconfig/sshd** 配置文件：默认值为 **false**。

**sshd_sysconfig_override_crypto_policy**

在 RHEL 中，当设为 **true** 时，这个变量会覆盖系统范围的加密策略。默认值为 **false**。

**sshd_sysconfig_use_strong_rng**

在基于 RHEL 的系统上，此变量可以强制 **sshd** 使用给定的字节数作为参数来重新设置 **openssl** 随机数字生成器的种子。默认值为 **0**，它会禁用此功能。如果系统没有硬件随机数字生成器，请不要打开此选项。

## 14.2. 使用 SSHD 系统角色配置 OPENSSH 服务器

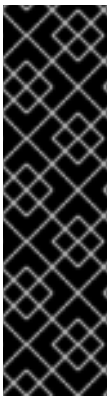
您可以通过运行 Ansible playbook，使用 **sshd** 系统角色配置多个 SSH 服务器。

**注意**

您可以将 **sshd** 系统角色用于更改 SSH 和 SSHD 配置的其他系统角色，例如身份管理 RHEL 系统角色。要防止配置被覆盖，请确保 **sshd** 角色使用命名空间(RHEL 8 和更早的版本)或 drop-in 目录(RHEL 9)。

**先决条件**

- 访问一个或多个 *受管节点* 的权限，这是您要使用 **sshd** 系统角色配置的系统。
- 对 *控制节点* 的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。
  在控制节点上：
  - **ansible-core** 和 **rhel-system-roles** 软件包已安装 。



**重要**

RHEL 8.0-8.5 提供对基于 Ansible 的自动化需要 Ansible Engine 2.9 的独立 Ansible 存储库的访问权限。Ansible Engine 包含命令行实用程序，如 **ansible**、**ansible-playbook**、连接器（如 **docker** 和 **podman** ）以及许多插件和模块。有关如何获取和安装 Ansible Engine 的详情，请参考 如何下载和安装 Red Hat Ansible Engine 知识库文章。

RHEL 8.6 和 9.0 引入了 Ansible Core（作为 **ansible-core** 软件包提供），其中包含 Ansible 命令行工具、命令以及小型内置 Ansible 插件。RHEL 通过 AppStream 软件仓库提供此软件包，它有一个有限的支持范围。如需更多信息，请参阅 RHEL 9 和 RHEL 8.6 及更新的 AppStream 软件仓库文档中的 Ansible Core 软件包的支持范围。

- 列出受管节点的清单文件。

**流程**

1. 复制 **sshd** 系统角色的示例 playbook：

```
-
```

```
# cp /usr/share/doc/rhel-system-roles/sshd/example-root-login-playbook.yml path/custom-playbook.yml
```

2. 使用文本编辑器打开复制的 playbook，例如：

```
# vim path/custom-playbook.yml

---
- hosts: all
  tasks:
  - name: Configure sshd to prevent root and password login except from particular subnet
    include_role:
      name: rhel-system-roles.sshd
    vars:
      sshd:
        # root login and password login is enabled only from a particular subnet
        PermitRootLogin: no
        PasswordAuthentication: no
        Match:
        - Condition: "Address 192.0.2.0/24"
          PermitRootLogin: yes
          PasswordAuthentication: yes
```

playbook 将受管节点配置为 SSH 服务器，以便：

- 禁用密码和 **root** 用户登录
- 只对子网 **192.0.2.0/24** 启用密码和 **root** 用户登录

您可以根据您的偏好修改变量。如需了解更多详细信息，请参阅 SSH 服务器系统角色变量。

3. 可选：验证 playbook 语法。

```
# ansible-playbook --syntax-check path/custom-playbook.yml
```

4. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory_file path/custom-playbook.yml

...

PLAY RECAP
**************************************************

localhost : ok=12 changed=2 unreachable=0 failed=0
skipped=10 rescued=0 ignored=0
```

**验证**

1. 登录到 SSH 服务器：

```
$ ssh user1@10.1.1.1
```

其中：

- ***user1*** 是 SSH 服务器上的用户。
- ***10.1.1.1*** 是 SSH 服务器的 IP 地址。

2. 检查 SSH 服务器上的 **sshd_config** 文件的内容：

```
$ cat /etc/ssh/sshd_config
…
PasswordAuthentication no
PermitRootLogin no
…
Match Address 192.0.2.0/24
  PasswordAuthentication yes
  PermitRootLogin yes
…
```

3. 检查您是否可以以 root 用户身份从 **192.0.2.0/24** 子网连接到服务器：

   a. 确定您的 IP 地址：

   ```
   $ hostname -I
   192.0.2.1
   ```

   如果 IP 地址在 **192.0.2.1** - **192.0.2.254** 范围内，您可以连接到服务器。

   b. 以 **root** 用户身份连接到服务器：

   ```
   $ ssh root@10.1.1.1
   ```

**其他资源**

- **/usr/share/doc/rhel-system-roles/sshd/README.md** 文件。
- **ansible-playbook(1)** 手册页。

## 14.3. SSH 系统角色变量

在 **ssh** 系统角色 playbook 中，您可以根据您的首选项和限制定义客户端 SSH 配置文件的参数。

如果没有配置这些变量，系统角色会生成一个与 RHEL 默认值匹配的全局 **ssh_config** 文件。

在所有情况下，布尔值在 **ssh** 配置中都正确地呈现为 **yes** 或 **no**。您可以使用 list 来定义多行配置项。例如：

```
LocalForward:
  - 22 localhost:2222
  - 403 localhost:4003
```

呈现为：

```
LocalForward 22 localhost:2222
LocalForward 403 localhost:4003
```

**注意**

配置选项区分大小写。

**ssh 系统角色的变量**

**ssh_user**

您可以定义系统角色修改用户特定配置的现有用户名。用户特定配置保存在给定用户的 **~/.ssh/config** 中。默认值为 null，它会修改所有用户的全局配置。

**ssh_skip_defaults**

默认值为 **auto**。如果设置为 **auto**，则系统角色将写入系统范围的配置文件 **/etc/ssh/ssh_config**，并在其中保留定义 RHEL 的默认值。例如，通过定义 **ssh_drop_in_name** 变量来创建一个 drop-in 配置文件，将自动禁用 **ssh_skip_defaults** 变量。

**ssh_drop_in_name**

定义 drop-in 配置文件的名称，该文件放在系统范围的 drop-in 目录中。该名称在模板 **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf** 中使用，以引用要修改的配置文件。如果系统不支持 drop-in 目录，则默认值为 null。如果系统支持 drop-in 目录，则默认值为 **00-ansible**。

**警告**

如果系统不支持 drop-in 目录，设置此选项将使 play 失败。

建议的格式是 **NN-name**，其中 **NN** 是用于订购配置文件的两位数字，**name** 是内容或文件所有者的任何描述性名称。

**ssh**

包含配置选项和其相应值的字典。

**ssh_*OptionName***

您可以使用由 **ssh_** 前缀和选项名称而不是字典组成的简单变量来定义选项。简单的变量覆盖 **ssh** 字典中的值。

**ssh_additional_packages**

此角色会自动安装 **openssh** 和 **openssh-clients** 软件包，这是最常见用例所需要的。如果您需要安装其他软件包，例如 **openssh-keysign** 以用于基于主机的身份验证，您可以在此变量中指定它们。

**ssh_config_file**

角色保存产生的配置文件的路径。默认值：

- 如果系统有一个 drop-in 目录，则默认值通过模板 **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf** 来定义。
- 如果系统没有 drop-in 目录，则默认值为 **/etc/ssh/ssh_config**。
- 如果定义了 **ssh_user** 变量，则默认值为 **~/.ssh/config**。

**ssh_config_owner,ssh_config_group,ssh_config_mode**

所创建的配置文件的所有者、组和模式。默认情况下，文件的所有者是 **root:root**，模式是 **0644**。如果定义了 **ssh_user**，则模式为 **0600**，所有者和组派生自 **ssh_user** 变量中指定的用户名。

## 14.4. 使用 SSH 系统角色配置 OPENSSH 客户端

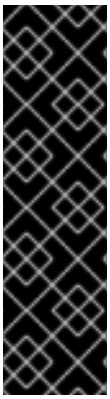
您可以通过运行 Ansible playbook，使用 **ssh** 系统角色配置多个 SSH 客户端。

**注意**

您可以将 **ssh** 系统角色用于更改 SSH 和 SSHD 配置的其他系统角色，如身份管理 RHEL 系统角色。要防止配置被覆盖，请确保 **ssh** 角色使用置入目录（默认为 RHEL 8）。

**先决条件**

- 访问一个或多个 *受管节点* 的权限，这是您要使用 **ssh** 系统角色配置的系统。
- 对 *控制节点* 的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。
  在控制节点上：
  - **ansible-core** 和 **rhel-system-roles** 软件包已安装 。



**重要**

RHEL 8.0-8.5 提供对基于 Ansible 的自动化需要 Ansible Engine 2.9 的独立 Ansible 存储库的访问权限。Ansible Engine 包含命令行实用程序，如 **ansible**、**ansible-playbook**、连接器（如 **docker** 和 **podman** ）以及许多插件和模块。有关如何获取和安装 Ansible Engine 的详情，请参考 如何下载和安装 Red Hat Ansible Engine 知识库文章。

RHEL 8.6 和 9.0 引入了 Ansible Core（作为 **ansible-core** 软件包提供），其中包含 Ansible 命令行工具、命令以及小型内置 Ansible 插件。RHEL 通过 AppStream 软件仓库提供此软件包，它有一个有限的支持范围。如需更多信息，请参阅 RHEL 9 和 RHEL 8.6 及更新的 AppStream 软件仓库文档中的 Ansible Core 软件包的支持范围。

- 列出受管节点的清单文件。

**流程**

1. 使用以下内容创建一个新的 ***playbook.yml*** 文件：

```
---
- hosts: all
  tasks:
  - name: "Configure ssh clients"
    include_role:
      name: rhel-system-roles.ssh
    vars:
      ssh_user: root
      ssh:
        Compression: true
        GSSAPIAuthentication: no
        ControlMaster: auto
        ControlPath: ~/.ssh/.cm%C
        Host:
          - Condition: example
            Hostname: example.com
            User: user1
      ssh_ForwardX11: no
```

此 playbook 使用以下配置在受管节点上配置 **root** 用户的 SSH 客户端首选项：

- 压缩已启用。
- ControlMaster 多路复用设置为 **auto**。
- 连接到 ***example.com*** 主机的***example*** 别名是 ***user1***。
- ***example*** 主机别名已创建，它表示使用 ***user1*** 用户名连接到 ***example.com*** 主机。
- X11 转发被禁用。

另外，您还可以根据您的偏好修改这些变量。如需了解更多详细信息，请参阅 **ssh** 系统角色变量。

2. 可选：验证 playbook 语法。

```
# ansible-playbook --syntax-check path/custom-playbook.yml
```

3. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory_file path/custom-playbook.yml
```

**验证**

- 通过在文本编辑器中打开 SSH 配置文件来验证受管节点是否具有正确的配置，例如：

```
# vi ~root/.ssh/config
```

在应用了上述示例 playbook 后，配置文件应具有以下内容：

```
# Ansible managed
Compression yes
ControlMaster auto
ControlPath ~/.ssh/.cm%C
ForwardX11 no
GSSAPIAuthentication no
Host example
  Hostname example.com
  User user1
```

## 14.5. 将 SSHD 系统角色用于非排除配置

通常，应用 **sshd** 系统角色会覆盖整个配置。如果您之前调整了配置，例如使用不同的系统角色或 playbook，这可能会出现问题。要只对所选配置选项应用 **sshd** 系统角色，同时保持其他选项，您可以使用非排除的配置。

在 RHEL 8 和更早版本中，您可以使用配置段来应用非独占配置。

**先决条件**

- 访问一个或多个 *受管节点* 的权限，这是您要使用 **sshd** 系统角色配置的系统。
- 对 *控制节点* 的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。

在控制节点上：

- **ansible-core** 软件包已安装 。
- 列出受管节点的清单文件。
- 不同 RHEL 系统角色的 playbook。

**流程**

1. 在 playbook 中添加带有 **sshd_config_namespace** 变量的配置片断：

```
---
- hosts: all
  tasks:
  - name: <Configure SSHD to accept some useful environment variables>
    include_role:
      name: rhel-system-roles.sshd
    vars:
      sshd_config_namespace: <my-application>
      sshd:
        # Environment variables to accept
        AcceptEnv:
          LANG
          LS_COLORS
          EDITOR
```

当您将 playbook 应用到清单时，角色会在 **/etc/ssh/sshd_config** 文件中添加下列代码片段（如果没有的话）。

```
# BEGIN sshd system role managed block: namespace <my-application>
Match all
  AcceptEnv LANG LS_COLORS EDITOR
# END sshd system role managed block: namespace <my-application>
```

**验证**

- 可选：验证 playbook 语法。

```
# ansible-playbook --syntax-check playbook.yml -i inventory_file
```

**其他资源**

- **/usr/share/doc/rhel-system-roles/sshd/README.md** 文件。
- **ansible-playbook(1)** 手册页。

# 第 15 章 使用 VPN RHEL 系统角色使用 IPSEC 配置 VPN 连接

使用 **vpn** 系统角色，您可以使用 Red Hat Ansible Automation Platform 在 RHEL 系统中配置 VPN 连接。您可以使用它来设置主机到主机、网络到网络、VPN 远程访问服务器和网格配置。

对于主机到主机连接，角色使用默认参数在 **vpn_connections** 列表中的每一对主机之间设置 VPN 通道，包括根据需要生成密钥。另外，您还可以将其配置为在列出的所有主机之间创建 机会主义网格配置。该角色假定 **hosts** 下的主机名称与 Ansible 清单中使用的主机的名称相同，并且您可以使用这些名称来配置通道。

**注意**

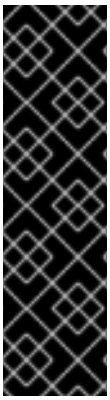
**vpn** RHEL 系统角色目前仅支持 Libreswan （即 IPsec 实现），作为 VPN 供应商。

## 15.1. 使用 VPN 系统角色使用 IPSEC 创建主机到主机的 VPN

您可以通过在控制节点上运行 Ansible playbook 来使用 **vpn** 系统角色配置主机到主机的连接，这将配置清单文件中列出的所有受管节点。

**先决条件**

- 对一个或多个 *受管节点* 的访问权限（即您要使用 **vpn** 系统角色配置的系统）。
- 对 *控制节点* 的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。
  在控制节点上：
  - **ansible-core** 和 **rhel-system-roles** 软件包已安装 。



**重要**

RHEL 8.0-8.5 提供对基于 Ansible 的自动化需要 Ansible Engine 2.9 的独立 Ansible 存储库的访问权限。Ansible Engine 包含命令行实用程序，如 **ansible**、**ansible-playbook**、连接器（如 **docker** 和 **podman** ）以及许多插件和模块。有关如何获取和安装 Ansible Engine 的详情，请参考 如何下载和安装 Red Hat Ansible Engine 知识库文章。

RHEL 8.6 和 9.0 引入了 Ansible Core（作为 **ansible-core** 软件包提供），其中包含 Ansible 命令行工具、命令以及小型内置 Ansible 插件。RHEL 通过 AppStream 软件仓库提供此软件包，它有一个有限的支持范围。如需更多信息，请参阅 RHEL 9 和 RHEL 8.6 及更新的 AppStream 软件仓库文档中的 Ansible Core 软件包的支持范围。

- 列出受管节点的清单文件。

**流程**

1. 使用以下内容创建一个新的 ***playbook.yml*** 文件：

```
- name: Host to host VPN
  hosts: managed_node1, managed_node2
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - hosts:
```

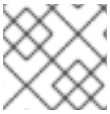
```
      managed_node1:
      managed_node2:
  vpn_manage_firewall: true
  vpn_manage_selinux: true
```

此 playbook 使用系统角色自动生成的密钥进行预共享密钥身份验证，来配置 ***managed_node1*-to-*managed_node2*** 的连接。由于 **vpn_manage_firewall** 和 **vpn_manage_selinux** 都被设置为 true，**vpn** 角色将使用 **firewall** 和 **selinux** 角色来管理 **vpn** 角色使用的端口。

2. 可选：将以下部分添加到主机的 **vpn_connections** 列表中，来配置从受管主机到清单文件中未列出的外部主机之间的连接：

```
  vpn_connections:
    - hosts:
        managed_node1:
        managed_node2:
        external_node:
          hostname: 192.0.2.2
```

这将配置另外两个连接：***managed_node1*-to-*external_node*** 和 ***managed_node2*-to-*external_node***。



**注意**

连接仅在受管节点上配置，而不在外部节点上配置。

1. 可选： 您可以使用 **vpn_connections** 中的附加部分为受管节点指定多个 VPN 连接，如控制平面和数据平面：

```
- name: Multiple VPN
  hosts: managed_node1, managed_node2
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - name: control_plane_vpn
        hosts:
          managed_node1:
            hostname: 192.0.2.0 # IP for the control plane
          managed_node2:
            hostname: 192.0.2.1
      - name: data_plane_vpn
        hosts:
          managed_node1:
            hostname: 10.0.0.1 # IP for the data plane
          managed_node2:
            hostname: 10.0.0.2
```

2. 可选： 您可以根据您的喜好修改变量。详情请查看 **/usr/share/doc/rhel-system-roles/vpn/README.md** 文件。

3. 可选：验证 playbook 语法。

```
# ansible-playbook --syntax-check /path/to/file/playbook.yml -i /path/to/file/inventory_file
```

4. 在清单文件上运行 playbook:

```
# ansible-playbook -i /path/to/file/inventory_file /path/to/file/playbook.yml
```

**验证**

1. 在受管节点上，确认连接已成功载入：

```
# ipsec status | grep connection.name
```

将 *connection.name* 替换为来自此节点的连接的名称，如 **managed_node1-to-managed_node2**。

**注意**

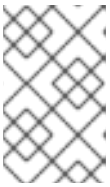
默认情况下，从每个系统的角度来看，角色为其创建的每个连接生成一个描述性名称。例如，当在 **managed_node1** 和 **managed_node2** 之间创建连接时，此连接在 **managed_node1** 上的描述性名称为 **managed_node1-to-managed_node2**，但在 **managed_node2** 上，连接的描述性名称为 **managed_node2-to-managed_node1**。

1. 在受管节点上，确认连接是否成功启动：

```
# ipsec trafficstatus | grep connection.name
```

2. 可选：如果连接没有成功加载，请输入以下命令来手动添加连接。这将提供更具体的信息，说明连接未能建立的原因：

```
# ipsec auto --add connection.name
```



**注意**

加载和启动连接过程中可能出现的任何错误都会在日志中报告，这些错误可以在 **/var/log/pluto.log** 中找到。由于这些日志难以解析，因此请尝试手动添加连接来获得日志消息，而不是从标准输出中获得。

## 15.2. 使用 VPN 系统角色创建与 IPSEC 的 OPPORTUNISTIC MESH VPN 连接

您可以使用 **vpn** 系统角色来配置机会主义网格 VPN 连接，该连接通过在控制节点上运行 Ansible playbook 来使用证书进行身份验证，它将配置清单文件中列出的所有受管节点。

通过在 playbook 中定义 **auth_method: cert** 参数来配置用证书进行身份验证。**vpn** 系统角色假设 IPsec 网络安全服务(NSS)加密库（在 **/etc/ipsec.d** 目录中定义）包含必要的证书。默认情况下，节点名称用作证书的昵称。在本例中，这是 **managed_node1**。您可以使用清单中的 **cert_name** 属性来定义不同的证书名称。

在以下示例流程中，控制节点（您要从其运行 Ansible playbook 的系统）与两个受管节点(192.0.2.0/24)共享相同的无类别域间路由(CIDR)数，并且 IP 地址为 192.0.2.7。因此，控制节点属于为 CIDR 192.0.2.0/24 自动创建的私有策略。

为防止在操作期间出现 SSH 连接丢失，控制节点的清晰策略包含在策略列表中。请注意，在策略列表中还有一个项 CIDR 等于 default。这是因为此 playbook 覆盖了默认策略中的规则，以使其为私有，而非私有或清晰。

**先决条件**

- 对一个或多个 *受管节点* 的访问权限（即您要使用 **vpn** 系统角色配置的系统）。
  - 在所有受管节点上，**/etc/ipsec.d** 目录中的 NSS 数据库包含对等身份验证所需的所有证书。默认情况下，节点名称用作证书的昵称。
- 对 *控制节点* 的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。
  在控制节点上：
  - **ansible-core** 和 **rhel-system-roles** 软件包已安装 。

**重要**

RHEL 8.0-8.5 提供对基于 Ansible 的自动化需要 Ansible Engine 2.9 的独立 Ansible 存储库的访问权限。Ansible Engine 包含命令行实用程序，如 **ansible**、**ansible-playbook**、连接器（如 **docker** 和 **podman** ）以及许多插件和模块。有关如何获取和安装 Ansible Engine 的详情，请参考 如何下载和安装 Red Hat Ansible Engine 知识库文章。

RHEL 8.6 和 9.0 引入了 Ansible Core（作为 **ansible-core** 软件包提供），其中包含 Ansible 命令行工具、命令以及小型内置 Ansible 插件。RHEL 通过 AppStream 软件仓库提供此软件包，它有一个有限的支持范围。如需更多信息，请参阅 RHEL 9 和 RHEL 8.6 及更新的 AppStream 软件仓库文档中的 Ansible Core 软件包的支持范围。

- 列出受管节点的清单文件。

**流程**

1. 使用以下内容创建新的 ***playbook.yml*** 文件：

```
- name: Mesh VPN
  hosts: managed_node1, managed_node2, managed_node3
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - opportunistic: true
        auth_method: cert
        policies:
          - policy: private
            cidr: default
          - policy: private-or-clear
            cidr: 198.51.100.0/24
          - policy: private
            cidr: 192.0.2.0/24
          - policy: clear
            cidr: 192.0.2.7/32
    vpn_manage_firewall: true
    vpn_manage_selinux: true
```

**注意**

由于 **vpn_manage_firewall** 和 **vpn_manage_selinux** 都被设置为 true，**vpn** 角色将使用 **firewall** 和 **selinux** 角色来管理 **vpn** 角色使用的端口。

2. 可选： 您可以根据您的喜好修改变量。详情请查看 **/usr/share/doc/rhel-system-roles/vpn/README.md** 文件。

3. 可选：验证 playbook 语法。

```
# ansible-playbook --syntax-check playbook.yml
```

4. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

## 15.3. 其他资源

- 有关 **vpn** 系统角色中使用的参数以及角色的附加信息，请参阅 **/usr/share/doc/rhel-system-roles/vpn/README.md** 文件。
- 有关 **ansible-playbook** 命令的详情，请查看 **ansible-playbook(1)** 手册页。

# 第 16 章 使用 CRYPTO-POLICIES RHEL 系统角色设置自定义加密策略

作为管理员，您可以使用 **crypto_policies** RHEL 系统角色在使用 Ansible Core 软件包的许多不同系统中快速一致地配置自定义加密策略。

## 16.1. CRYPTO_POLICIES 系统角色变量和事实

在 **crypto_policies** 系统角色 playbook 中，您可以根据您的首选项和限制定义 **crypto_policies** 配置文件的参数。

如果没有配置任何变量，系统角色不会配置系统，只会报告事实。

### 为 crypto_policies 系统角色选择的变量

**crypto_policies_policy**

确定系统角色应用到受管节点的加密策略。有关不同加密策略的详情，请参阅 系统范围的加密策略。

**crypto_policies_reload**

如果设置为 **yes**，则目前受影响的服务 **ipsec**、**bind** 和 **sshd** 服务，在应用加密策略后重新加载。默认值为 **yes**。

**crypto_policies_reboot_ok**

如果设置为 **yes**，在系统角色更改了加密策略后需要重新启动，它会将 **crypto_policies_reboot_required** 设置为 **yes**。默认值为 **no**。

### crypto_policies 系统角色设置的事实

**crypto_policies_active**

列出当前选择的策略。

**crypto_policies_available_policies**

列出系统上所有可用的策略。

**crypto_policies_available_subpolicies**

列出系统上所有可用的子策略。

### 其他资源

- 创建并设置自定义系统范围的加密策略。

## 16.2. 使用 CRYPTO_POLICIES 系统角色设置自定义加密策略

您可以使用 **crypto_policies** 系统角色从单个控制节点配置大量的受管节点。

### 先决条件

- 访问一个或多个 *受管节点* 的权限，这是您要使用 **crypto_policies** 系统角色配置的系统。
- 对控制节点的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。
  在控制节点上：
  - **ansible-core** 和 **rhel-system-roles** 软件包已安装 。

**重要**

RHEL 8.0-8.5 提供对基于 Ansible 的自动化需要 Ansible Engine 2.9 的独立 Ansible 存储库的访问权限。Ansible Engine 包含命令行实用程序，如 **ansible**、**ansible-playbook**、连接器（如 **docker** 和 **podman** ）以及许多插件和模块。有关如何获取和安装 Ansible Engine 的详情，请参考 如何下载和安装 Red Hat Ansible Engine 知识库文章。

RHEL 8.6 和 9.0 引入了 Ansible Core（作为 **ansible-core** 软件包提供），其中包含 Ansible 命令行工具、命令以及小型内置 Ansible 插件。RHEL 通过 AppStream 软件仓库提供此软件包，它有一个有限的支持范围。如需更多信息，请参阅 RHEL 9 和 RHEL 8.6 及更新的 AppStream 软件仓库文档中的 Ansible Core 软件包的支持范围。

- 列出受管节点的清单文件。

**流程**

1. 使用以下内容创建新 ***playbook.yml*** 文件：

```
---
- hosts: all
  tasks:
  - name: Configure crypto policies
    include_role:
      name: rhel-system-roles.crypto_policies
    vars:
      - crypto_policies_policy: FUTURE
      - crypto_policies_reboot_ok: true
```

   您可以将 *FUTURE* 值替换为您喜欢的加密策略，例如：**DEFAULT**、 **LEGACY** 和 **FIPS:OSPP**。

   **crypto_policies_reboot_ok: true** 变量会导致系统在系统角色更改加密策略后重启系统。

   如需了解更多详细信息，请参阅 crypto_policies 系统角色变量和事实。

2. 可选：验证 playbook 语法。

```
# ansible-playbook --syntax-check playbook.yml
```

3. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory_file playbook.yml
```

**验证**

1. 在控制节点上，创建另一个 playbook，例如,名为 ***verify_playbook.yml***:

```
- hosts: all
  tasks:
  - name: Verify active crypto policy
    include_role:
      name: rhel-system-roles.crypto_policies

  - debug:
      var: crypto_policies_active
```

此 playbook 不更改系统上的任何配置，只报告受管节点上的活动策略。

2. 运行同一个清单文件上的 playbook:

```
# ansible-playbook -i inventory_file verify_playbook.yml

TASK [debug] ***************************
ok: [host] => {
    "crypto_policies_active": "FUTURE"
}
```

**"crypto_policies_active":** 变量显示受管节点上的活动策略。

## 16.3. 其它资源

- **/usr/share/ansible/roles/rhel-system-roles.crypto_policies/README.md** 文件。
- **ansible-playbook(1)** 手册页。
- 准备一个控制节点和受管节点以使用 RHEL 系统角色。

# 第 17 章 使用 RHEL 系统角色配置 NBDE

## 17.1. NBDE_CLIENT 和 NBDE_SERVER 系统角色 (CLEVIS 和 TANG) 简介

RHEL 系统角色是 Ansible 角色和模块的集合,可为远程管理多个 RHEL 系统提供一致的配置界面。

您可以使用 Ansible 角色使用 Clevis 和 Tang 自动部署基于策略的解密(PBD)解决方案。**rhel-system-roles** 包中包含了这些系统角色、相关的例子以及参考文档。

**nbde_client** 系统角色可让您以自动的方式部署多个 Clevis 客户端。请注意，**nbde_client** 角色只支持 Tang 绑定，您目前无法将其用于 TPM2 绑定。

**nbde_client** 角色需要已经使用 LUKS 加密的卷。此角色支持将 LUKS 加密卷绑定到一个或多个网络绑定(NBDE)服务器 - Tang 服务器。您可以使用密码短语保留现有的卷加密，或者将其删除。删除密码短语后，您只能使用 NBDE 解锁卷。当卷最初是使用在置备系统后会删除的临时密钥或密码进行加密时，这非常有用，

如果您同时提供密语和密钥文件，角色将使用您首先提供的那一个。如果找不到任何有效密语或密码，它将尝试从现有的绑定中检索密码短语。

PBD 将绑定定义为设备到插槽的映射。这意味着对同一个设备你可以有多个绑定。默认插槽是插槽 1。

**nbde_client** 角色也提供了 **state** 变量。使用 **present** 值来创建新绑定或更新现有绑定。与 **clevis luks bind** 命令不同，您可以使用 **state: present** 来覆盖其设备插槽中的现有绑定。**absent** 的值会删除指定的绑定。

使用 **nbde_client** 系统角色，您可以部署和管理 Tang 服务器作为自动磁盘加密解决方案的一部分。此角色支持以下功能：

- 轮转 Tang 密钥
- 部署和备份 Tang 密钥

**其它资源**

- 有关网络绑定磁盘加密(NBDE)角色变量的详细参考，请安装 **rhel-system-roles** 软件包，并查看 **/usr/share/doc/rhel-system-roles/nbde_client/** 和 **/usr/share/doc/rhel-system-roles/nbde_server/** 目录中的 **README.md** 和**README.html** 文件。
- 关于系统角色 playbook 示例，请安装 **rhel-system-roles** 软件包，并查看 **/usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/** 目录。
- 有关 RHEL 系统角色的更多信息，请参阅 准备控制节点和受管节点以使用 RHEL 系统角色。

## 17.2. 使用 NBDE_SERVER 系统角色设置多个 TANG 服务器

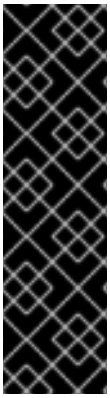
按照以下步骤准备和应用包含您的 Tang 服务器设置的 Ansible playbook。

**先决条件**

- 对一个或多个 *受管节点* 的访问和权限，这些节点是您要使用 **nbde_server** 系统角色配置的系统。
- 对控制节点的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。

在控制节点上：

- **ansible-core** 和 **rhel-system-roles** 软件包已安装 。



**重要**

RHEL 8.0-8.5 提供对基于 Ansible 的自动化需要 Ansible Engine 2.9 的独立 Ansible 存储库的访问权限。Ansible Engine 包含命令行实用程序，如 **ansible**、**ansible-playbook**、连接器（如 **docker** 和 **podman** ）以及许多插件和模块。有关如何获取和安装 Ansible Engine 的详情，请参考 如何下载和安装 Red Hat Ansible Engine 知识库文章。

RHEL 8.6 和 9.0 引入了 Ansible Core（作为 **ansible-core** 软件包提供），其中包含 Ansible 命令行工具、命令以及小型内置 Ansible 插件。RHEL 通过 AppStream 软件仓库提供此软件包，它有一个有限的支持范围。如需更多信息，请参阅 RHEL 9 和 RHEL 8.6 及更新的 AppStream 软件仓库文档中的 Ansible Core 软件包的支持范围。

- 列出受管节点的清单文件。

**流程**

1. 准备包含 Tang 服务器设置的 playbook。您可以从头开始，或使用 **/usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/** 目录中的一个 playbook 示例。

```
# cp /usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/simple_deploy.yml
./my-tang-playbook.yml
```

2. 在您选择的文本编辑器中编辑 playbook，例如：

```
# vi my-tang-playbook.yml
```

3. 添加所需参数。以下 playbook 示例确保部署 Tang 服务器和密钥轮转：

```
---
- hosts: all

  vars:
    nbde_server_rotate_keys: yes
    nbde_server_manage_firewall: true
    nbde_server_manage_selinux: true

  roles:
    - rhel-system-roles.nbde_server
```

**注意**

因为 **nbde_server_manage_firewall** 和 **nbde_server_manage_selinux** 都被设置为 true，所以 **nbde_server** 角色将使用 **firewall** 和 **selinux** 角色来管理 **nbde_server** 角色使用的端口。

4. 应用完成的 playbook:

```
# ansible-playbook -i inventory-file my-tang-playbook.yml
```

其中：* ***inventory-file*** 是清单文件。* ***logging-playbook.yml*** 是您使用的 playbook。

**重要**

在安装了 Clevis 的系统上使用 **grubby** 工具来确保 Tang pin 的网络可用：

```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

**其它资源**

- 如需更多信息，请安装 **rhel-system-roles** 软件包，并查看 **/usr/share/doc/rhel-system-roles/nbde_server/** 和 **usr/share/ansible/roles/rhel-system-roles.nbde_server/** 目录。

## 17.3. 使用 NBDE_CLIENT 系统角色设置多个 CLEVIS 客户端

按照步骤准备和应用包含 Clevis 客户端设置的 Ansible playbook。

**注意**

**nbde_client** 系统角色只支持 Tang 绑定。这意味着您目前无法将其用于 TPM2 绑定。

**先决条件**

- 对一个或多个 *受管节点* 的访问和权限，这些节点是您要使用 **nbde_client** 系统角色配置的系统。
- 对 *控制节点* 的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。
- Ansible Core 软件包安装在控制机器上。
- **rhel-system-roles** 软件包安装在您要运行 playbook 的系统中。

**流程**

1. 准备包含 Clevis 客户端设置的 playbook。您可以从头开始，或使用 **/usr/share/ansible/roles/rhel-system-roles.nbde_client/examples/** 目录中的一个 playbook 示例。

```
# cp /usr/share/ansible/roles/rhel-system-roles.nbde_client/examples/high_availability.yml
./my-clevis-playbook.yml
```

2. 在您选择的文本编辑器中编辑 playbook，例如：

```
# vi my-clevis-playbook.yml
```

3. 添加所需参数。以下 playbook 示例配置 Clevis 客户端，以便在两个 Tang 服务器中至少有一个可用时自动解锁两个 LUKS 加密卷：

```
---
- hosts: all

  vars:
    nbde_client_bindings:
      - device: /dev/rhel/root
```

```
      encryption_key_src: /etc/luks/keyfile
      servers:
        - http://server1.example.com
        - http://server2.example.com
    - device: /dev/rhel/swap
      encryption_key_src: /etc/luks/keyfile
      servers:
        - http://server1.example.com
        - http://server2.example.com

  roles:
    - rhel-system-roles.nbde_client
```

4. 应用完成的 playbook:

```
# ansible-playbook -i host1,host2,host3 my-clevis-playbook.yml
```

**重要**

通过使用在安装了 Clevis 的系统上的 **grubby** 工具来确保在早期引导期间 Tang pin 的网络可用：

```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

**其它资源**

- 有关 NBDE 客户端系统角色的参数和附加信息，请安装 **rhel-system-roles** 软件包，并查看 **/usr/share/doc/rhel-system-roles/nbde_client/** 和 **/usr/share/ansible/roles/rhel-system-roles.nbde_client/** 目录。

# 第 18 章 使用 RHEL 系统角色请求证书

使用 **certificate** 系统角色，您可以使用 Red Hat Ansible Core 来发布和管理证书。

本章涵盖了以下主题：

- **certificate** 系统角色
- 使用 **certificate** 系统角色请求新的自签名证书
- 使用 **certificate** 系统角色从 IdM CA 请求一个新证书

## 18.1. CERTIFICATE 系统角色

使用 **certificate** 系统角色，您可以使用 Ansible Core 管理签发和续订 TLS 和 SSL 证书。

该角色使用 **certmonger** 作为证书提供者，目前支持发布和续订自签名证书及使用 IdM 集成认证机构 (CA)。

您可以将 Ansible playbook 中的以下变量与 **certificate** 系统角色结合使用：

**certificate_wait**
来指定任务是否应该等待要发布的证书。

**certificate_requests**
来表示要发布的每个证书及其参数。

**其他资源**

- 请参阅 **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md** 文件。
- 准备控制节点和受管节点以使用 RHEL 系统角色

## 18.2. 使用 CERTIFICATE 系统角色请求新的自签名证书

使用 **certificate** 系统角色，您可以使用 Ansible Core 来发布自签名证书。

此过程使用 **certmonger** 提供者，并通过 **getcert** 命令请求证书。

**注意**

默认情况下，**certmonger** 会在证书过期前自动尝试续订证书。您可以通过将 Ansible playbook 中的 **auto_renew** 参数设置为 **no** 来禁用此功能。

**先决条件**

- Ansible Core 软件包安装在控制机器上。
- 您已在要运行 playbook 的系统上安装了 **rhel-system-roles** 软件包。

**流程**

1. *可选：* 创建一个清单文件，如 **inventory.file** ：

```
$ *touch inventory.file*
```

2. 打开清单文件并定义要请求证书的主机，例如：

```
[webserver]
server.idm.example.com
```

3. 创建 playbook 文件，如 **request-certificate.yml**:

   - 将 **hosts** 设置为包含您要请求证书的主机，如 **webserver**。
   - 将 **certificate_requests** 变量设置为包含以下项：
     - 将 **name** 参数设置为证书的所需名称，如 **mycert**。
     - 将 **dns** 参数设置为证书中包含的域，如 ***.example.com**。
     - 将 **ca** 参数设置为 **self-sign**。
   - 在 **roles** 下设置 **rhel-system-roles.certificate** 角色。
     这是本例的 playbook 文件：

```
---
- hosts: webserver

  vars:
    certificate_requests:
      - name: mycert
        dns: "*.example.com"
        ca: self-sign

  roles:
    - rhel-system-roles.certificate
```

4. 保存该文件。

5. 运行 playbook:

```
$ *ansible-playbook -i inventory.file request-certificate.yml*
```

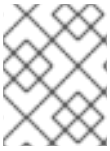
**其他资源**

- 请参阅 **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md** 文件。
- 请参阅 **ansible-playbook(1)** 手册页。

## 18.3. 使用 CERTIFICATE 系统角色从 IDM CA 请求一个新证书

使用 **certificate** 系统角色，您可以在使用带有集成证书颁发机构(CA)的 IdM 服务器时，使用 **anible-core** 来发布证书。因此，当使用 IdM 作为 CA 时，您可以高效且一致地为多个系统管理证书信任链。

此过程使用 **certmonger 供应商**，并通过 **getcert** 命令请求证书。



**注意**

默认情况下，**certmonger** 会在证书过期前自动尝试续订证书。您可以通过将 Ansible playbook 中的 **auto_renew** 参数设置为 **no** 来禁用此功能。

**先决条件**

- Ansible Core 软件包安装在控制机器上。
- 您已在要运行 playbook 的系统上安装了 **rhel-system-roles** 软件包。

**流程**

1. *可选：* 创建一个清单文件，如 **inventory.file** ：

```
$ *touch inventory.file*
```

2. 打开清单文件并定义要请求证书的主机，例如：

```
[webserver]
server.idm.example.com
```

3. 创建 playbook 文件，如 **request-certificate.yml**:
   - 将 **hosts** 设置为包含您要请求证书的主机，如 **webserver**。
   - 将 **certificate_requests** 变量设置为包含以下项：
     - 将 **name** 参数设置为证书的所需名称，如 **mycert**。
     - 将 **dns** 参数设置为证书中包含的域，如 **www.example.com**。
     - 将 **principal** 参数设置为指定 Kerberos 主体，如 **HTTP/www.example.com@EXAMPLE.COM**。
     - 将 **ca** 参数设置为 **ipa**。
   - 在 **roles** 下设置 **rhel-system-roles.certificate** 角色。
     这是本例的 playbook 文件：

```
---
- hosts: webserver
  vars:
    certificate_requests:
      - name: mycert
        dns: www.example.com
        principal: HTTP/www.example.com@EXAMPLE.COM
        ca: ipa

  roles:
    - rhel-system-roles.certificate
```

4. 保存该文件。
5. 运行 playbook:

```
$ *ansible-playbook -i inventory.file request-certificate.yml*
```

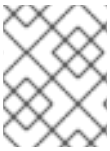
**其他资源**

- 请参阅 **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md** 文件。
- 请参阅 **ansible-playbook(1)** 手册页。

## 18.4. 使用 CERTIFICATE 系统角色指定证书颁发前或后要运行的命令

使用 **certificate** 系统角色，您可以使用 Ansible Core 在签发或更新证书前后执行命令。

在以下示例中，管理员确保在为 **www.example.com** 发布或更新自签名证书前停止 **httpd** 服务，然后再重启该服务。



**注意**

默认情况下，**certmonger** 会在证书过期前自动尝试续订证书。您可以通过将 Ansible playbook 中的 **auto_renew** 参数设置为 **no** 来禁用此功能。

**先决条件**

- Ansible Core 软件包安装在控制机器上。
- 您已在要运行 playbook 的系统上安装了 **rhel-system-roles** 软件包。

**流程**

1. *可选：* 创建一个清单文件，如 **inventory.file** ：

```
$ *touch inventory.file*
```

2. 打开清单文件并定义要请求证书的主机，例如：

```
[webserver]
server.idm.example.com
```

3. 创建 playbook 文件，如 **request-certificate.yml**:
   - 将 **hosts** 设置为包含您要请求证书的主机，如 **webserver**。
   - 将 **certificate_requests** 变量设置为包含以下项：
     - 将 **name** 参数设置为证书的所需名称，如 **mycert**。
     - 将 **dns** 参数设置为证书中包含的域，如 **www.example.com**。
     - 将 **ca** 参数设置为您要用来发布证书的 CA，如 **自签名**。
     - 将 **run_before** 参数设置为在签发或续订证书之前要执行的命令，如 **systemctl stop httpd.service**。
     - 将 **run_after** 参数设置为在签发或续订此证书后要执行的命令，如 **systemctl start httpd.service**。

- 在 **roles** 下设置 **rhel-system-roles.certificate** 角色。
  这是本例的 playbook 文件：

```
---
- hosts: webserver
  vars:
    certificate_requests:
      - name: mycert
        dns: www.example.com
        ca: self-sign
        run_before: systemctl stop httpd.service
        run_after: systemctl start httpd.service

  roles:
    - rhel-system-roles.certificate
```

4. 保存该文件。
5. 运行 playbook:

```
$ *ansible-playbook -i inventory.file request-certificate.yml*
```

**其他资源**

- 请参阅 **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md** 文件。
- 请参阅 **ansible-playbook(1)** 手册页。

# 第 19 章 使用 KDUMP RHEL 系统角色配置自动化崩溃转储

要使用 Ansible 管理 kdump，您可以使用 **kdump** 角色，这是 RHEL 7.9 中可用的 RHEL 系统角色之一。

使用 **kdump** 角色可让您指定保存系统内存内容的位置，以便稍后进行分析。

## 19.1. KDUMP RHEL 系统角色

**kdump** 系统角色可让您在多个系统上设置基本内核转储参数。

## 19.2. KDUMP 角色参数

**kdump** RHEL 系统角色使用的参数有：

| 角色变量 | 描述 |
| --- | --- |
| kdump_path | 写入 **vmcore** 的路径。如果 **kdump_target** 不是 null，则路径相对于那个转储目标。否则，它必须是 root 文件系统的绝对路径。 |

**其他资源**

- makedumpfile(8)手册页。
- 有关 **kdump** 中使用的参数的详细信息，以及有关 **kdump** 系统角色的额外信息，请参阅 **/usr/share/ansible/roles/rhel-system-roles.tlog/README.md** 文件。

## 19.3. 使用 RHEL 系统角色配置 KDUMP

您可以通过运行 Ansible playbook，在多个系统上使用 **kdump** 系统角色来设置基本内核转储参数。

**警告**

通过替换 **/etc/kdump.conf**文件，**kdump** 角色完全取代了受管主机的 kdump 配置。另外,如果应用了 **kdump** 角色，则之前的所有 **kdump** 设置也会被替换，即使它们没有被角色变量指定，也可以替换 **/etc/sysconfig/kdump** 文件。

**先决条件**

- Ansible Core 软件包安装在控制机器上。
- 您已在要运行 playbook 的系统上安装了 **rhel-system-roles** 软件包。
- 您有一个清单文件，其中列出了您要在其上部署 **kdump** 的系统。

**流程**

1. 使用以下内容创建新的 ***playbook.yml*** 文件：

```
---
- hosts: kdump-test
  vars:
    kdump_path: /var/crash
  roles:
    - rhel-system-roles.kdump
```

2. 可选：验证 playbook 语法。

```
# ansible-playbook --syntax-check playbook.yml
```

3. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

**其他资源**

- 有关 **kdump** 角色变量的详情，请查看 /usr/share/doc/rhel-system-roles/kdump 目录中的 README.md 或 README.html 文件。
- 请参阅准备控制节点和受管节点以使用 RHEL 系统角色
- 随 **rhel-system-roles** 软件包安装的文档 **/usr/share/ansible/roles/rhel-system-roles.kdump/README.html**

# 第 20 章 使用 RHEL 系统角色管理本地存储

要使用 Ansible 管理 LVM 和本地文件系统(FS)，您可以使用 **storage** 角色，这是 RHEL 8 中提供的 RHEL 系统角色之一。

使用 **存储** 角色可让您自动管理多台机器上的磁盘和逻辑卷上的文件系统，以及从 RHEL 7.7 开始的所有 RHEL 版本。

有关 RHEL 系统角色的更多信息，以及如何应用它们，请参阅 RHEL 系统角色简介。

## 20.1. STORAGE RHEL 系统角色简介

**存储** 角色可以管理：

- 磁盘上未被分区的文件系统
- 完整的 LVM 卷组，包括其逻辑卷和文件系统
- MD RAID 卷及其文件系统

使用 **storage** 角色，您可以执行以下任务：

- 创建文件系统
- 删除文件系统
- 挂载文件系统
- 卸载文件系统
- 创建 LVM 卷组
- 删除 LVM 卷组
- 创建逻辑卷
- 删除逻辑卷
- 创建 RAID 卷
- 删除 RAID 卷
- 使用 RAID 创建 LVM 卷组
- 使用 RAID 删除 LVM 卷组
- 创建加密的 LVM 卷组
- 使用 RAID 创建 LVM 逻辑卷

## 20.2. 在 STORAGE RHEL 系统角色中识别存储设备的参数

您的 **存储** 角色配置只会影响您在以下变量中列出的文件系统、卷和池：

**storage_volumes**

在所有要管理的未分区磁盘中的文件系统列表。
**storage_volumes** 也可以包含 **raid** 卷。

当前不支持的分区。

**storage_pools**

要管理的池列表。
目前唯一支持的池类型是 LVM。使用 LVM 时，池代表卷组（VG）。每个池中都有一个要由角色管理的卷列表。对于 LVM，每个卷对应一个带文件系统的逻辑卷（LV）。

## 20.3. 在块设备中创建 XFS 文件系统的 ANSIBLE PLAYBOOK 示例

本节提供了一个 Ansible playbook 示例。此 playbook 应用 **存储** 角色，来使用默认参数在块设备上创建 XFS 文件系统。

**警告**

**存储** 角色只能在未分区、整个磁盘或逻辑卷(LV)上创建文件系统。它不能在分区中创建文件系统。

**例 20.1. 在 /dev/sdb 上创建 XFS 的 playbook**

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
  roles:
    - rhel-system-roles.storage
```

- 卷名称（示例中为 ***barefs***）目前是任意的。**存储** 角色根据 **disks:** 属性下列出的磁盘设备来识别卷。
- 您可以省略 **fs_type: xfs** 行，因为 XFS 是 RHEL 8 中的默认文件系统。
- 要在 LV 上创建文件系统，请在 **disks:** 属性下提供 LVM 设置，包括括起的卷组。详情请参阅管理逻辑卷的 Ansible playbook 示例。
  不要提供到 LV 设备的路径。

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件。

## 20.4. 永久挂载文件系统的 ANSIBLE PLAYBOOK 示例

本节提供了一个 Ansible playbook 示例。此 playbook 应用 **存储** 角色来立即且永久挂载 XFS 文件系统。

**例 20.2. 将 /dev/sdb 上的文件系统挂载到 /mnt/data 的 playbook**

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- 此 playbook 将文件系统添加到 **/etc/fstab** 文件中，并立即挂载文件系统。
- 如果 **/dev/sdb** 设备上的文件系统或挂载点目录不存在，则 playbook 会创建它们。

**其它资源**

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件。

## 20.5. 管理逻辑卷的 ANSIBLE PLAYBOOK 示例

本节提供了一个 Ansible playbook 示例。此 playbook 应用**存储** 角色来在卷组中创建 LVM 逻辑卷。

**例 20.3. 在 myvg 卷组中创建 mylv 逻辑卷的 playbook**

```
- hosts: all
  vars:
    storage_pools:
      - name: myvg
        disks:
          - sda
          - sdb
          - sdc
        volumes:
          - name: mylv
            size: 2G
            fs_type: ext4
            mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- **myvg** 卷组由以下磁盘组成：
  - **/dev/sda**

    - **/dev/sdb**
    - **/dev/sdc**
  - 如果 **myvg** 卷组已存在，则 playbook 会将逻辑卷添加到卷组。
  - 如果 **myvg** 卷组不存在，则 playbook 会创建它。
  - playbook 在 **mylv** 逻辑卷上创建 Ext4 文件系统，并在 **/mnt** 上永久挂载文件系统。

**其它资源**

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件。

## 20.6. 启用在线块丢弃的 ANSIBLE PLAYBOOK 示例

本节提供了一个 Ansible playbook 示例。此 playbook 应用 **存储** 角色来挂载启用了在线块丢弃的 XFS 文件系统。

**例 20.4. 一个 playbook，它在 /mnt/data/ 上启用在线块丢弃功能**

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        mount_point: /mnt/data
        mount_options: discard
  roles:
    - rhel-system-roles.storage
```

**其他资源**

- 永久挂载文件系统的 Ansible playbook 示例
- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件。

## 20.7. 创建和挂载 EXT4 文件系统的 ANSIBLE PLAYBOOK 示例

本节提供了一个 Ansible playbook 示例。此 playbook 应用 **存储** 角色来创建和挂载 Ext4 文件系统。

**例 20.5. 在 /dev/sdb 上创建 Ext4 并挂载到 /mnt/data 的 playbook**

```
---
- hosts: all
  vars:
    storage_volumes:
```

```
    - name: barefs
      type: disk
      disks:
        - sdb
      fs_type: ext4
      fs_label: label-name
      mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- playbook 在 **/dev/sdb** 磁盘上创建文件系统。
- playbook 永久将文件系统挂载在 ***/mnt/data*** 目录。
- 文件系统的标签是 ***label-name***。

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件。

## 20.8. 创建和挂载 EXT3 文件系统的 ANSIBLE PLAYBOOK 示例

本节提供了一个 Ansible playbook 示例。此 playbook 应用 **存储** 角色来创建和挂载 Ext3 文件系统。

**例 20.6. 在 /dev/sdb 上创建 Ext3，并将其挂载到/mnt/data 的 playbook**

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: ext3
        fs_label: label-name
        mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- playbook 在 **/dev/sdb** 磁盘上创建文件系统。
- playbook 永久将文件系统挂载在 ***/mnt/data*** 目录。
- 文件系统的标签是 ***label-name***。

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件。

## 20.9. 使用 STORAGE RHEL 系统角色调整现有 EXT4 或 EXT3 文件系统大小的 ANSIBLE PLAYBOOK 示例

本节提供了一个 Ansible playbook 示例。此 playbook 应用 **存储** 角色来调整块设备上现有的 Ext4 或 Ext3 文件系统的大小。

**例 20.7. 在磁盘上设置单个卷的 playbook**

```
---
- name: Create a disk device mounted on /opt/barefs
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - /dev/sdb
        size: 12 GiB
        fs_type: ext4
        mount_point: /opt/barefs
  roles:
    - rhel-system-roles.storage
```

- 如果上例中的卷已存在，若要调整卷大小，您需要运行相同的 playbook，只是参数 **size** 的值不同。例如：

**例 20.8. 在 /dev/sdb 上调整 ext4 大小的 playbook**

```
---
- name: Create a disk device mounted on /opt/barefs
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - /dev/sdb
        size: 10 GiB
        fs_type: ext4
        mount_point: /opt/barefs
  roles:
    - rhel-system-roles.storage
```

- 卷名称（示例中为 barefs）当前是任意的。Storage 角色根据 disks: attribute 中列出的磁盘设备标识卷。

**注意**

在其他文件系统中使用 **Resizing** 操作可能会破坏您正在使用的设备上的数据。

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件。

## 20.10. 使用 STORAGE RHEL 系统角色在 LVM 上调整现有文件系统的大小的 ANSIBLE PLAYBOOK 示例

本节提供了一个 Ansible playbook 示例。此 playbook 应用 **storage** RHEL 系统角色，以使用文件系统重新定义 LVM 逻辑卷大小。

**警告**

在其他文件系统中使用 **Resizing** 操作可能会破坏您正在使用的设备上的数据。

**例 20.9. 调整 myvg 卷组中现有 mylv1 和 myvl2 逻辑卷大小的 playbook**

```
---

- hosts: all
  vars:
   storage_pools:
     - name: myvg
       disks:
         - /dev/sda
         - /dev/sdb
         - /dev/sdc
       volumes:
          - name: mylv1
            size: 10 GiB
            fs_type: ext4
            mount_point: /opt/mount1
          - name: mylv2
            size: 50 GiB
            fs_type: ext4
            mount_point: /opt/mount2

- name: Create LVM pool over three disks
  include_role:
    name: rhel-system-roles.storage
```

- 此 playbook 调整以下现有文件系统的大小：
  - 挂载在 **/opt/mount1** 上的 **mylv1** 卷上的 Ext4 文件系统，大小调整为 10 GiB。
  - 挂载在 **/opt/mount2** 上的 **mylv2** 卷上的 Ext4 文件系统，大小调整为 50 GiB。

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件。

## 20.11. 使用 STORAGE RHEL 系统角色创建交换卷的 ANSIBLE PLAYBOOK 示例

本节提供了一个 Ansible playbook 示例。此 playbook 应用 **storage** 角色来创建交换卷（如果不存在），或者修改交换卷（如果已存在）在使用默认参数的块设备上。

**例 20.10. 创建或修改 /dev/sdb 上现有 XFS 的 playbook**

```
---
- name: Create a disk device with swap
- hosts: all
  vars:
    storage_volumes:
      - name: swap_fs
        type: disk
        disks:
          - /dev/sdb
  size: 15 GiB
        fs_type: swap
  roles:
    - rhel-system-roles.storage
```

- 卷名称（示例中的 ***swap_fs***）目前是任意的。**存储** 角色根据 **disks:** 属性下列出的磁盘设备来识别卷。

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件。

## 20.12. 使用存储系统角色配置 RAID 卷

使用 **storage** 系统角色，您可以使用 Red Hat Ansible Automation Platform 和 Ansible-Core 在 RHEL 上配置 RAID 卷。使用参数创建一个 Ansible playbook，以配置 RAID 卷以满足您的要求。

**先决条件**

- Ansible Core 软件包安装在控制机器上。
- 您已在要运行 playbook 的系统上安装了 **rhel-system-roles** 软件包。
- 您有一个清单文件详细描述了您要使用 **存储** 系统角色部署 RAID 卷的系统。

**流程**

1. 创建包含以下内容的 *playbook.yml* 文件：

```
---
- name: Configure the storage
  hosts: managed-node-01.example.com
  tasks:
  - name: Create a RAID on sdd, sde, sdf, and sdg
    include_role:
```

```
    name: rhel-system-roles.storage
vars:
storage_safe_mode: false
storage_volumes:
  - name: data
    type: raid
    disks: [sdd, sde, sdf, sdg]
    raid_level: raid0
    raid_chunk_size: 32 KiB
    mount_point: /mnt/data
    state: present
```

**警告**

设备名称在某些情况下可能会改变，例如：当您在系统中添加新磁盘时。因此，为了避免数据丢失，请不要在 playbook 中使用特定的磁盘名称。

2. 可选：验证 playbook 语法：

```
# ansible-playbook --syntax-check playbook.yml
```

3. 运行 playbook：

```
# ansible-playbook -i inventory.file /path/to/file/playbook.yml
```

**其他资源**

- 管理 RAID
- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件
- 准备一个控制节点和一个受管节点以使用 RHEL 系统角色。

## 20.13. 使用 STORAGE RHEL 系统角色配置带有 RAID 的 LVM 池

使用 **存储** 系统角色，您可以使用 Red Hat Ansible Automation Platform 在 RHEL 上配置带有 RAID 的 LVM 池。在本小节中，您将了解如何使用可用参数设置 Ansible playbook，以配置使用 RAID 的 LVM 池。

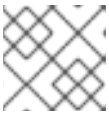
**先决条件**

- Ansible Core 软件包安装在控制机器上。
- 您已在要运行 playbook 的系统上安装了 **rhel-system-roles** 软件包。
- 您有一个清单文件详细描述了您要使用 **存储** 系统角色在其上配置带有 RAID 的 LVM 池的系统。

**流程**

1. 使用以下内容创建新的 ***playbook.yml*** 文件：

```
- hosts: all
  vars:
    storage_safe_mode: false
    storage_pools:
      - name: my_pool
        type: lvm
        disks: [sdh, sdi]
        raid_level: raid1
        volumes:
          - name: my_pool
            size: "1 GiB"
            mount_point: "/mnt/app/shared"
            fs_type: xfs
            state: present
  roles:
    - name: rhel-system-roles.storage
```



**注意**

要使用 RAID 创建 LVM 池，您必须使用 **raid_level** 参数来指定 RAID 类型。

2. 可选。验证 playbook 语法。

```
# ansible-playbook --syntax-check playbook.yml
```

3. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory.file /path/to/file/playbook.yml
```

**其他资源**

- 管理 RAID 。
- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件。

## 20.14. 使用 STORAGE RHEL 系统角色在 LVM 上压缩和去掉重复数据的 VDO 卷的 ANSIBLE PLAYBOOK 示例

本节提供了一个 Ansible playbook 示例。此 playbook 应用 **storage** RHEL 系统角色，以便使用虚拟数据优化器 (VDO) 启用逻辑卷 (LVM) 的压缩和重复数据删除。

**例 20.11. 在 myvg 卷组中创建 mylv1 LVM VDO 卷的 playbook**

```
---
- name: Create LVM VDO volume under volume group 'myvg'
  hosts: all
  roles:
    -rhel-system-roles.storage
  vars:
    storage_pools:
```

```
- name: myvg
  disks:
    - /dev/sdb
  volumes:
    - name: mylv1
      compression: true
      deduplication: true
      vdo_pool_size: 10 GiB
      size: 30 GiB
      mount_point: /mnt/app/shared
```

在本例中，**compression** 和 **deduplication** 池被设为 true，这指定使用 VDO。下面描述了这些参数的用法：

- **deduplication** 用于去除存储在存储卷上的重复数据。
- compression 用于压缩存储在存储卷上的数据，从而提高存储量。
- vdo_pool_size 指定卷在设备上占用的实际大小。VDO 卷的虚拟大小由 **size** 参数设置。注：由于 Storage 角色使用 LVM VDO，因此每个池只有一个卷可以使用压缩和重复数据删除。

## 20.15. 使用 STORAGE RHEL 系统角色创建 LUKS2 加密的卷

您可以通过运行 Ansible playbook，使用 **storage** 角色来创建和配置使用 LUKS 加密的卷。

**先决条件**

- 对一个或多个受管节点的访问和权限，这些节点是您要使用 **crypto_policies** 系统角色配置的系统。
- 清单文件，其列出受管节点。
- 对控制节点的访问和权限，这是 Red Hat Ansible Core 配置其他系统的系统。在控制节点上，已安装了 **ansible-core** 和 **rhel-system-roles** 软件包。

**重要**

RHEL 8.0-8.5 提供对基于 Ansible 的自动化需要 Ansible Engine 2.9 的独立 Ansible 存储库的访问权限。Ansible Engine 包含命令行实用程序，如 **ansible**、**ansible-playbook**、连接器（如 **docker** 和 **podman** ）以及许多插件和模块。有关如何获取和安装 Ansible Engine 的详情，请参考 如何下载和安装 Red Hat Ansible Engine 知识库文章。

RHEL 8.6 和 9.0 引入了 Ansible Core（作为 **ansible-core** 软件包提供），其中包含 Ansible 命令行工具、命令以及小型内置 Ansible 插件。RHEL 通过 AppStream 软件仓库提供此软件包，它有一个有限的支持范围。如需更多信息，请参阅 RHEL 9 和 RHEL 8.6 及更新的 AppStream 软件仓库文档中的 Ansible Core 软件包的支持范围。

**流程**

1. 使用以下内容创建新的 ***playbook.yml*** 文件：

```
- hosts: all
  vars:
```

```
    storage_volumes:
      - name: barefs
        type: disk
        disks:
         - sdb
        fs_type: xfs
        fs_label: label-name
        mount_point: /mnt/data
        encryption: true
        encryption_password: your-password
  roles:
   - rhel-system-roles.storage
```

您还可以在 *playbook.yml* 文件中添加其他加密参数，如 **encryption_key**、**encryption_cipher**、**encryption_key_size** 和 **encryption_luks** 版本。

2. 可选：验证 playbook 语法：

```
# ansible-playbook --syntax-check playbook.yml
```

3. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory.file /path/to/file/playbook.yml
```

**验证**

1. 查看加密状态：

```
# cryptsetup status sdb

/dev/mapper/sdb is active and is in use.
type: LUKS2
cipher: aes-xts-plain64
keysize: 512 bits
key location: keyring
device: /dev/sdb
[...]
```

2. 验证创建的 LUKS 加密的卷：

```
# cryptsetup luksDump /dev/sdb

Version:        2
Epoch:          6
Metadata area:  16384 [bytes]
Keyslots area:  33521664 [bytes]
UUID:           a4c6be82-7347-4a91-a8ad-9479b72c9426
Label:          (no label)
Subsystem:      (no subsystem)
Flags:          allow-discards

Data segments:
  0: crypt
 offset: 33554432 [bytes]
```

```
  length: (whole device)
  cipher: aes-xts-plain64
  sector: 4096 [bytes]
 [...]
```

3. 查看 ***playbook.yml*** 文件中的 **cryptsetup** 参数， **storage** 角色对其支持：

```
# cat ~/playbook.yml

    - hosts: all
      vars:
        storage_volumes:
          - name: foo
            type: disk
            disks:
             - nvme0n1
            fs_type: xfs
            fs_label: label-name
            mount_point: /mnt/data
            encryption: true
            #encryption_password: passwdpasswd
            encryption_key: /home/passwd_key
            encryption_cipher: aes-xts-plain64
            encryption_key_size: 512
            encryption_luks_version: luks2

      roles:
       - rhel-system-roles.storage
```

**其他资源**

- 使用 LUKS 加密块设备
- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件

## 20.16. 使用 STORAGE RHEL 系统角色以百分比形式表示池卷大小的 ANSIBLE PLAYBOOK 示例

本节提供了一个 Ansible playbook 示例。此 playbook 应用 **storage** 系统角色，以表达作为池总大小的百分比的逻辑卷管理器卷(LVM)卷大小。

**例 20.12. 将卷大小表示为池总大小百分比形式的 playbook**

```
---
- name: Express volume sizes as a percentage of the pool's total size
  hosts: all
  roles
    - rhel-system-roles.storage
  vars:
    storage_pools:
    - name: myvg
      disks:
        - /dev/sdb
      volumes:
```

```
    - name: data
      size: 60%
      mount_point: /opt/mount/data
    - name: web
      size: 30%
      mount_point: /opt/mount/web
    - name: cache
      size: 10%
      mount_point: /opt/cache/mount
```

这个示例将 LVM 卷的大小指定为池大小的百分比，例如："60%"。另外，您还可以将 LVM 卷的大小指定为人类可读的文件系统形式（如 "10g" 或 "50 GiB"）的池大小的百分比。

## 20.17. 其他资源

- **/usr/share/doc/rhel-system-roles/storage/**
- **/usr/share/ansible/roles/rhel-system-roles.storage/**

# 第 21 章 使用 TIMESYNC RHEL 系统角色配置时间同步

借助 **timesync** RHEL 系统角色，您可以使用 Red Hat Ansible Automation Platform 在 RHEL 上的多个目标机器上管理时间同步。

## 21.1. TIMESYNC RHEL 系统角色

您可以使用 **timesync** RHEL 系统角色在多个目标机器上管理时间同步。

**timesync** 角色安装和配置 NTP 或 PTP 实现，来作为 NTP 客户端或 PTP 副本进行操作，以便将系统时钟与 NTP 服务器或 PTP 域中的 Pumasters 同步。

请注意，使用 **timesync** 角色也有助于 迁移到 chrony，因为您可以在从 RHEL 6 开始的所有 Red Hat Enterprise Linux 版本上使用相同的 playbook，而无论系统是使用 **ntp** 还是 **chrony** 来实现 NTP 协议。

## 21.2. 为单一服务器池应用 TIMESYNC 系统角色

以下示例演示了如何在只有一个服务器池的情况下应用 **timesync** 角色。

**警告**

**timesync 角色**替换了受管主机上给定或检测到的供应商服务的配置。之前的设置即使没有在角色变量中指定，也会丢失。如果没有定义 **timesync_ntp_provider** 变量，唯一保留的设置就是供应商选择。

**先决条件**

- Ansible Core 软件包安装在控制机器上。
- 您已在要运行 playbook 的系统上安装了 **rhel-system-roles** 软件包。
- 您有一个清单文件，其中列出了您要在其上部署 **timesync** 系统角色的系统。

**流程**

1. 使用以下内容创建新的 ***playbook.yml*** 文件：

```
---
- hosts: timesync-test
  vars:
    timesync_ntp_servers:
      - hostname: 2.rhel.pool.ntp.org
        pool: yes
        iburst: yes
  roles:
    - rhel-system-roles.timesync
```

2. 可选：验证 playbook 语法。

```
# ansible-playbook --syntax-check playbook.yml
```

3. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

## 21.3. 在客户端服务器上应用 TIMESYNC 系统角色

您可以使用 **timesync** 角色来在 NTP 客户端上启用网络时间安全(NTS)。网络时间安全(NTS)是一个针对网络时间协议(NTP)指定的身份验证机制。它验证在服务器和客户端之间交换的 NTP 数据包是否未被更改。

**警告**

**timesync 角色**替换了受管主机上给定或检测到的供应商服务的配置。即使未在角色变量中指定，之前的设置也会丢失。如果没有定义 **timesync_ntp_provider** 变量，唯一保留的设置就是供应商选择。

**先决条件**

- 您不必在要部署 **timesync** 解决方案的系统上安装 Red Hat Ansible Automation Platform。
- 您已在要运行 playbook 的系统上安装了 **rhel-system-roles** 软件包。
- 您有一个清单文件，其中列出了您要在其上部署 **timesync** 系统角色的系统。
- **chrony** NTP 提供者版本为 4.0 或更高版本。

**流程**

1. 使用以下内容创建一个 ***playbook.yml*** 文件：

```
---
- hosts: timesync-test
  vars:
    timesync_ntp_servers:
      - hostname: ptbtime1.ptb.de
        iburst: yes
        nts: yes
  roles:
    - rhel-system-roles.timesync
```

**ptbtime1.ptb.de** 是一个公共服务器的示例。您可能想要使用不同的公共服务器或您自己的服务器。

2. 可选：验证 playbook 语法。

```
# ansible-playbook --syntax-check playbook.yml
```

3. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

**验证**

1. 在客户端机器上执行测试：

```
# chronyc -N authdata

Name/IP address         Mode KeyID Type KLen Last Atmp  NAK Cook CLen
==============================================================================
ptbtime1.ptb.de         NTS     1   15  256  157    0    0    8  100
```

2. 检查是否报告的 cookies 数量大于零。

**其他资源**

- **chrony.conf(5)** 手册页

## 21.4. TIMESYNC 系统角色变量

您可以将以下变量传递给 **timesync** 角色：

- **timesync_ntp_servers**:

| 角色变量设置 | 描述 |
|---|---|
| hostname: host.example.com | 服务器的主机名或地址 |
| minpoll: *number* | 最小轮询间隔。默认：6 |
| maxpoll: *number* | 最大轮询间隔。默认：10 |
| iburst: yes | 标志启用快速初始同步。默认： no |
| pool: yes | 指示每个主机名解析地址都是一个单独的 NTP 服务器的标志。默认： no |
| nts: yes | 用于启用网络时间安全性(NTS)的标记。默认：no。只支持 chrony >= 4.0。 |

**其他资源**

- 有关 **timesync** 角色变量的详细参考，请安装 rhel-system-roles 软件包，并参阅 **/usr/share/doc/rhel-system-roles/timesync** 目录中的 README.md 或 README.html 文件。

# 第 22 章 使用 METRICS RHEL 系统角色监控性能

作为系统管理员，您可以将 **metrics** RHEL 系统角色与任何 Ansible Automation Platform 控制节点一起使用，来监控系统性能。

## 22.1. METRICS 系统角色简介

RHEL 系统角色是 Ansible 角色和模块的集合,可为远程管理多个 RHEL 系统提供一致的配置界面。**metrics** 系统角色为本地系统配置性能分析服务，并可以选择包含要由本地系统监控的远程系统的列表。**metrics** 系统角色可让您使用 **pcp** 来监控系统性能，而无需单独配置 **pcp**，因为 playbook 处理 **pcp** 的设置和部署。

**表 22.1. metrics 系统角色变量**

| 角色变量 | 描述 | 用法示例 |
|---|---|---|
| metrics_monitored_hosts | 要通过目标主机分析的远程主机的列表。这些主机将在目标主机上记录指标，因此要确保每个主机的 **/var/log** 下有足够的磁盘空间。 | **metrics_monitored_hosts: ["*webserver.example.com*", "*database.example.com*"]** |
| metrics_retention_days | 在删除前配置性能数据保留的天数。 | **metrics_retention_days: 14** |
| metrics_graph_service | 一个布尔值标志，使主机能够通过 **pcp** 和 **grafana** 设置性能数据可视化服务。默认设置为 false。 | **metrics_graph_service: no** |
| metrics_query_service | 一个布尔值标志，使主机能够通过 **redis** 设置时间序列查询服务，来查询记录的 **pcp** 指标。默认设置为 false。 | **metrics_query_service: no** |
| metrics_provider | 指定要用于提供指标的指标收集器。目前，**pcp** 是唯一受支持的指标提供者。 | **metrics_provider: "pcp"** |
| metrics_manage_firewall | 使用 **firewall** 角色管理直接从 **metrics** 角色访问的端口。默认设置为 false。 | **metrics_manage_firewall: true** |
| metrics_manage_selinux | 使用 **selinux** 角色管理直接从 **metrics** 角色访问的端口。默认设置为 false。 | **metrics_manage_selinux: true** |

**注意**

如需有关 **metrics_connections** 中使用的参数，以及有关 **metrics** 系统角色的其他信息，请参阅 **/usr/share/ansible/roles/rhel-system-roles.metrics/README.md** 文件。

## 22.2. 使用 METRICS 系统角色以可视化方式监控本地系统

此流程描述了如何使用 **metrics** RHEL 系统角色来监控您的本地系统，同时通过 **Grafana** 提供数据可视化。

**先决条件**

- Ansible Core 软件包安装在控制机器上。
- 您已在要监控的机器上安装了 **rhel-system-roles** 软件包。

**流程**

1. 通过在清单中添加以下内容，在 **/etc/ansible/hosts** Ansible 清单中配置 **localhost** ：

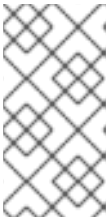
```
localhost ansible_connection=local
```

2. 使用以下内容创建一个 Ansible playbook:

```
---
- name: Manage metrics
  hosts: localhost
  vars:
    metrics_graph_service: yes
    metrics_manage_firewall: true
    metrics_manage_selinux: true
  roles:
    - rhel-system-roles.metrics
```

3. 运行 Ansible playbook:

```
# ansible-playbook name_of_your_playbook.yml
```



**注意**

由于 **metrics_graph_service** 布尔值被设置为 value="yes"，因此会使用 **pcp** 自动安装并提供 **Grafana** ，并添加为数据源。由于 metrics_manage_firewall 和 metrics_manage_selinux 都被设置为 true，因此 metrics 角色将使用 firewall 和 selinux 系统角色来管理 metrics 角色使用的端口。

4. 要查看机器上收集的指标的视图，请访问 **grafanaweb** 界面 ， 如 访问 Grafana web UI 中所述。

## 22.3. 使用 METRICS 系统角色设置监控其自身的独立系统

此流程描述了如何使用 **metrics** 系统角色设置一组机器来监控其自身。

**先决条件**

- Ansible Core 软件包安装在控制机器上。
- 您已在要用来运行 playbook 的机器上安装了 **rhel-system-roles** 软件包。
- 您已建立 SSH 连接。

**流程**

1. 将您要通过 playbook 监控的机器的名称或 IP 添加到方括号括起来的标识组名称下的 **/etc/ansible/hosts** Ansible 清单文件中：

```
[remotes]
webserver.example.com
database.example.com
```

2. 使用以下内容创建一个 Ansible playbook:

```
---
- hosts: remotes
  vars:
    metrics_retention_days: 0
    metrics_manage_firewall: true
    metrics_manage_selinux: true
  roles:
    - rhel-system-roles.metrics
```

**注意**

由于 **metrics_manage_firewall** 和 **metrics_manage_selinux** 都被设置为 true，因此 metrics 角色将使用 **firewall** 和 **selinux** 角色来管理 **metrics** 角色使用的端口。

3. 运行 Ansible playbook:

```
# ansible-playbook name_of_your_playbook.yml -k
```

其中 **-k** 提示连接到远程系统的密码。

## 22.4. 使用 METRICS 系统角色通过本地机器监控机器的数量

此流程描述了如何使用 **metrics** 系统角色设置本地机器来集中一组监控机器，同时通过 **grafana** 提供数据的可视化，并通过 **redis** 提供数据的查询。

**先决条件**

- Ansible Core 软件包安装在控制机器上。
- 您已在要用来运行 playbook 的机器上安装了 **rhel-system-roles** 软件包。

**流程**

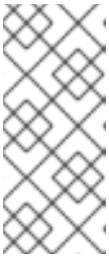
1. 使用以下内容创建一个 Ansible playbook:

```
---
- hosts: localhost
  vars:
    metrics_graph_service: yes
    metrics_query_service: yes
    metrics_retention_days: 10
```

```
    metrics_monitored_hosts: ["database.example.com", "webserver.example.com"]
    metrics_manage_firewall: yes
    metrics_manage_selinux: yes
  roles:
    - rhel-system-roles.metrics
```

2. 运行 Ansible playbook:

```
# ansible-playbook name_of_your_playbook.yml
```



**注意**

由于 **metrics_graph_service** 和 **metrics_query_service** 布尔值被设置为 value="yes"，因此会使用**pcp** 来自动安装并提供 **grafana**，并添加为带有 **pcp** 数据记录索引到 **redis** 的数据源，允许 **pcp** 查询语言用于复杂的数据查询。由于 **metrics_manage_firewall** 和 **metrics_manage_selinux** 都被设置为 true，因此 **metrics** 角色将使用 **firewall** 和 **selinux** 角色来管理 **metrics** 角色使用的端口。

3. 要查看机器集中收集的指标的图形表示，并查询数据，请访问 **grafana** web 界面，如 访问 Grafana Web UI 中所述。

## 22.5. 在使用 METRICS 系统角色监控系统时设置身份验证

PCP 通过简单身份验证安全层 (SASL) 框架支持 **scram-sha-256** 验证机制。**metrics** RHEL 系统角色使用 **scram-sha-256** 身份验证机制自动设置身份验证的步骤。这个步骤描述了如何使用 **metrics** RHEL 系统角色设置身份验证。

**先决条件**

- Ansible Core 软件包安装在控制机器上。
- 您已在要用来运行 playbook 的机器上安装了 **rhel-system-roles** 软件包。

**流程**

1. 在您要为其设置身份验证的 Ansible playbook 中包含以下变量：

```
---
  vars:
    metrics_username: your_username
    metrics_password: your_password
    metrics_manage_firewall: true
    metrics_manage_selinux: true
```

**注意**

由于 **metrics_manage_firewall** 和 **metrics_manage_selinux** 都被设置为 true，因此 **metrics** 角色将使用 **firewall** 和 **selinux** 角色来管理 **metrics** 角色使用的端口。

2. 运行 Ansible playbook:

```
# ansible-playbook name_of_your_playbook.yml
```

**验证步骤**

- 验证 **sasl** 配置：

```
# pminfo -f -h "pcp://ip_adress?username=your_username" disk.dev.read
Password:
disk.dev.read
inst [0 or "sda"] value 19540
```

*ip_adress* 应替换为主机的 IP 地址。

## 22.6. 使用 METRICS 系统角色为 SQL SERVER 配置并启用指标集合

此流程描述了如何使用 **metrics** RHEL 系统角色通过您本地系统中的 **pcp** 自动配置和为 Microsoft SQL Server 启用指标集合。

**先决条件**

- Ansible Core 软件包安装在控制机器上。
- 您已在要监控的机器上安装了 **rhel-system-roles** 软件包。
- 您已安装了用于 Red Hat Enterprise Linux 的 Microsoft SQL Server，并建立了与 SQL 服务器的"信任"连接。请参阅安装 SQL Server 并在红帽上创建数据库。
- 您已为 Red Hat Enterprise Linux 安装了用于 SQL Server 的 Microsoft ODBC 驱动程序。请参阅 Red Hat Enterprise Server 和 Oracle Linux 。

**流程**

1. 通过在清单中添加以下内容，在 **/etc/ansible/hosts** Ansible 清单中配置 **localhost** ：

```
localhost ansible_connection=local
```

2. 创建一个包含以下内容的 Ansible playbook：

```
---
- hosts: localhost
  vars:
   metrics_from_mssql: true
   metrics_manage_firewall: true
   metrics_manage_selinux: true
  roles:
   - role: rhel-system-roles.metrics
```

**注意**

由于 **metrics_manage_firewall** 和 **metrics_manage_selinux** 都被设置为 true，因此 **metrics** 角色将使用 **firewall** 和 **selinux** 角色来管理 **metrics** 角色使用的端口。

3. 运行 Ansible playbook:

```
# ansible-playbook name_of_your_playbook.yml
```

**验证步骤**

- 使用 **pcp** 命令来验证 SQL Server PMDA 代理 (mssql) 是否已加载并在运行：

```
# pcp
platform: Linux rhel82-2.local 4.18.0-167.el8.x86_64 #1 SMP Sun Dec 15 01:24:23 UTC
2019 x86_64
 hardware: 2 cpus, 1 disk, 1 node, 2770MB RAM
 timezone: PDT+7
 services: pmcd pmproxy
     pmcd: Version 5.0.2-1, 12 agents, 4 clients
     pmda: root pmcd proc pmproxy xfs linux nfsclient mmv kvm mssql
           jbd2 dm
 pmlogger: primary logger: /var/log/pcp/pmlogger/rhel82-2.local/20200326.16.31
     pmie: primary engine: /var/log/pcp/pmie/rhel82-2.local/pmie.log
```

**其它资源**

- 有关用于 Microsoft SQL Server 的 Performance CoPilot 的更多信息，请参阅此红帽开发者博客文章。

# 第 23 章 使用 MICROSOFT.SQL.SERVER ANSIBLE 角色配置 MICROSOFT SQL SERVER

作为管理员，您可以使用 **microsoft.sql.server** Ansible 角色来安装、配置和启动 Microsoft SQL Server(SQL Server)。The **microsoft.sql.server** Ansible 角色优化您的操作系统，以提高 SQL Server 的性能和吞吐量。该角色使用推荐的设置来简化和自动化 RHEL 主机配置，以运行 SQL Server 工作负载。

## 23.1. 先决条件

- 2 GB RAM
- **root** 访问您要配置 SQL Server 的受管节点
- 预配置的防火墙
  您可以将 **mssql_manage_firewall** 变量设置为 **true**，以便角色可以自动管理防火墙。

  另外，还可在使用 **mssql_tcp_port** 变量设置的 SQL Server TCP 端口上启用连接。如果没有定义此变量，则角色默认为 TCP 端口号 **1433**。

  要添加新端口，请使用：

  ```
  # firewall-cmd --add-port=xxxx/tcp --permanent
  # firewall-cmd --reload
  ```

  将 *xxxx* 替换为 TCP 端口号，然后重新加载防火墙规则。
- *可选*：创建一个包含 SQL 语句和存储进程的带 **.sql** 扩展名的文件，来将它们输入到 SQL 服务器。

## 23.2. 安装 MICROSOFT.SQL.SERVER ANSIBLE 角色

**microsoft.sql.server** Ansible 角色是 **ansible-collection-microsoft-sql** 软件包的一部分。

**先决条件**

- **root** 访问权限

**流程**

1. 安装 Ansible Core，它在 RHEL 7.9 AppStream 存储库中提供：

   ```
   # *yum install ansible-core*
   ```

2. 安装 **microsoft.sql.server** Ansible 角色：

   ```
   # *yum install ansible-collection-microsoft-sql*
   ```

## 23.3. 使用 MICROSOFT.SQL.SERVER ANSIBLE 角色安装并配置 SQL 服务器

您可以使用 **microsoft.sql.server** Ansible 角色来安装和配置 SQL server。

**先决条件**

- Ansible 清单已创建

**流程**

1. 创建一个扩展名为 **.yml** 的文件。例如，**mssql-server.yml**。
2. 在 **.yml** 文件中添加以下内容：

```
---
- hosts: all
  vars:
    mssql_accept_microsoft_odbc_driver_17_for_sql_server_eula: true
    mssql_accept_microsoft_cli_utilities_for_sql_server_eula: true
    mssql_accept_microsoft_sql_server_standard_eula: true
    mssql_password: <password>
    mssql_edition: Developer
    mssql_tcp_port: 1433
  roles:
    - microsoft.sql.server
```

将 *<password>* 替换为您的 SQL Server 密码。

3. 运行 **mssql-server.yml** ansible playbook:

```
# *ansible-playbook mssql-server.yml*
```

## 23.4. TLS 变量

您可以使用以下变量来配置传输层安全(TLS)协议。

**表 23.1. TLS 角色变量**

| 角色变量 | 描述 |
| --- | --- |
| mssql_tls_enable | 此变量启用或禁用 TLS 加密。<br><br>当变量设置为 **true** 时，**microsoft.sql.server** Ansible 角色执行以下任务：<br><br>● 将 TLS 证书复制到 SQL Server 上的 **/etc/pki/tls/certs/**<br><br>● 将私钥复制到 SQL Server 上的 **/etc/pki/tls/private/**<br><br>● 将 SQL Server 配置为使用 TLS 证书和私钥来加密连接<br><br>如果设置为 **false**，则禁用 TLS 加密。该角色不会删除现有的证书和私钥文件。 |
| mssql_tls_cert | 要定义此变量，请输入 TLS 证书文件的路径。 |

| 角色变量 | 描述 |
| --- | --- |
| mssql_tls_private_key | 要定义此变量，请输入私钥文件的路径。 |
| mssql_tls_remote_src | 定义该角色是否在远程或控制节点上搜索 **mssql_tls_cert** 和 **mssql_tls_private_key** 文件。<br><br>当设置为默认 **false** 时，该角色在 Ansible 控制节点上搜索 **mssql_tls_cert** 或 **mssql_tls_private_key** 文件。<br><br>当设置为 **true** 时，该角色在 Ansible 受管节点上搜索 **mssql_tls_cert** 或 **mssql_tls_private_key** 文件。 |
| mssql_tls_version | 定义此变量来选择要使用的 TSL 版本。<br><br>默认值为 **1.2** |
| mssql_tls_force | 将此变量设为 **true** 以替换主机上的证书和私钥文件。这些文件必须在 **/etc/pki/tls/certs/** 和 **/etc/pki/tls/private/** 目录下。<br><br>默认值为 **false**。 |

## 23.5. 接受 MLSERVICE 的 EULA

您必须接受 Python 和 R 软件包开源分发的所有 EULA，才能安装所需的 SQL Server 机器学习服务 (MLService)。

有关许可证条款，请参阅 **/usr/share/doc/mssql-server**。

**表 23.2. SQL Server 机器学习服务 EULA 变量**

| 角色变量 | 描述 |
| --- | --- |
| mssql_accept_microsoft_sql_server_standard_eula | 此变量决定是否接受安装 **mssql-conf** 软件包的条款和条件。<br><br>要接受条款和条件，请将这个变量设为 **true**。<br><br>默认值为 **false**。 |

## 23.6. 接受 MICROSOFT ODBC 17 的 EULA。

您必须接受所有 EULA，才能安装 Microsoft Open Database Connectivity(ODBC)驱动程序。

有关许可证条款，请参阅 **/usr/share/doc/msodbcsql17/LICENSE.txt** 和 **/usr/share/doc/mssql-tools/LICENSE.txt**。

**表 23.3. Microsoft ODBC 17 EULA 变量**

| 角色变量 | 描述 |
| --- | --- |
| mssql_accept_microsoft_odbc_driver_17_for_sql_server_eula | 此变量决定是否接受安装 **msodbcsql17** 软件包的条款和条件。<br><br>要接受条款和条件，请将这个变量设为 **true**。<br><br>默认值为 **false**。 |
| mssql_accept_microsoft_cli_utilities_for_sql_server_eula | 此变量决定是否接受安装 **mssql-tools** 软件包的条款和条件。<br><br>要接受条款和条件，请将这个变量设为 **true**。<br><br>默认值为 **false**。 |

## 23.7. 高可用性变量

您可以使用下表中的变量为 Microsoft SQL Server 配置高可用性。

**表 23.4. 高可用性配置变量**

| 变量 | 描述 |
| --- | --- |
| **mssql_ha_configure** | 默认值为 **false**。<br><br>当它被设置为 **true** 时，执行以下操作：<br><br>- 通过打开 **mssql_ha_listener_port** 变量的端口来配置防火墙，并在防火墙中启用 **high-availability** 服务。<br>- 配置 SQL Server 以实现高可用性。<br>  - 启用 Always On Health 事件。<br>  - 在主副本上创建证书并将其分发给其他副本。<br>  - 配置端点和资源组。<br>  - 为 Pacemaker 配置来自 **mssql_ha_login** 变量的用户。<br>- 可选：包含系统角色 **ha_cluster** 角色，以配置 Pacemaker。您必须将 **mssql_ha_cluster_run_role** 设置为 **true**，并提供 **ha_cluster** 角色为 Pacemaker 集群配置所需的所有变量。 |
| **mssql_ha_replica_type** | 此变量指定您可以在主机上配置哪些类型的副本。您可以将此变量设置为 **primary**, **synchronous**, 和 **witness**。您必须只在一个主机上将其设置为 **primary**。 |

| 变量 | 描述 |
| --- | --- |
| **mssql_ha_listener_port** | 默认端口为 **5022**。<br><br>该角色使用此 TCP 端口为 Always On availability 组复制数据。 |
| **mssql_ha_cert_name** | 您必须定义证书的名称，以便在 Always On Availability 组成员间保护事务。 |
| **mssql_ha_master_key_password** | 您必须设置用于证书的 master 密钥的密码。 |
| **mssql_ha_private_key_password** | 您必须设置用于证书的私钥的密码。 |
| **mssql_ha_reset_cert** | 默认值为 **false**。<br><br>如果设置为 **true**，请重置 Always On availability 组使用的证书。 |
| **mssql_ha_endpoint_name** | 您必须定义要配置的端点的名称。 |
| **mssql_ha_ag_name** | 您必须定义要配置的可用区的名称。 |
| **mssql_ha_db_names** | 您可以定义要复制的数据库列表，否则角色会创建集群，而不复制数据库。 |
| **mssql_ha_login** | SQL Server Pacemaker 资源代理利用此用户来执行数据库健康检查，并管理从副本转换到主服务器的状态。 |
| **mssql_ha_login_password** | SQL Server 中的 **mssql_ha_login** 用户的密码。 |
| **mssql_ha_cluster_run_role** | 默认值为 **false**。<br><br>此变量定义此角色是否运行 **ha_cluster** 角色。<br><br>请注意，**ha_cluster** 角色替换了指定节点上 HA 集群的配置，当前为 HA 集群配置的所有变量都会被清除并覆盖。<br><br>要临时解决这个问题，**microsoft.sql.server** 角色不会为 **ha_cluster** 角色设置任何变量，以确保它不会覆盖任何现有的 Pacemaker 配置。<br><br>如果您希望 **microsoft.sql.server** 运行 **ha_cluster** 角色，请将此变量设置为 **true**，并为 **ha_cluster** 角色提供变量，并带有 **microsoft.sql.server** 角色调用。 |

请注意，此角色会将数据库备份到 **/var/opt/mssql/data/** 目录。

有关如何为 Microsoft SQL Server 使用高可用性变量的示例：

- 如果从 Automation Hub 安装角色，请查看服务器中的 **~/.ansible/collections/ansible_collections/microsoft/sql/roles/server/README.md** 文件。
- 如果从软件包安装角色，请在浏览器中打开 **/usr/share/microsoft/sql-server/README.html** 文件。

# 第 24 章 配置系统以使用 TLOG RHEL 系统角色记录会话记录

使用 **tlog** RHEL 系统角色，您可以使用 Red Hat Ansible Automation Platform 为 RHEL 上的终端会话记录配置一个系统。

## 24.1. TLOG 系统角色

您可以使用 **tlog** RHEL 系统角色为 RHEL 上的终端会话记录配置 RHEL 系统。

您可以使用 **SSSD** 服务将记录配置为为每个用户或用户组进行。

**其他资源**

- 有关 RHEL 中会话记录的详情，请参阅记录会话。

## 24.2. TLOG 系统角色的组件和参数

Session Recording 解决方案有以下组件：

- **tlog** 工具
- 系统安全性服务守护进程（SSSD）
- 可选： Web 控制台界面

用于 **tlog** RHEL 系统角色的参数有：

| 角色变量 | 描述 |
| --- | --- |
| tlog_use_sssd (default: yes) | 使用 SSSD 配置会话记录，这是管理记录的用户或组的首选方法 |
| tlog_scope_sssd（默认值：none） | 配置 SSSD 记录范围 - all / some / none |
| tlog_users_sssd（默认值：[]） | 要记录的用户的 YAML 列表 |
| tlog_groups_sssd（默认值：[]） | 要记录的组的 YAML 列表 |

- 有关 **tlog** 中使用的参数以及 **tlog** 系统角色的附加信息，请查看 **/usr/share/ansible/roles/rhel-system-roles.tlog/README.md** 文件。

## 24.3. 部署 TLOG RHEL 系统角色

按照以下步骤准备并应用 Ansible playbook 来配置 RHEL 系统，以将数据记录到 systemd 日志中。

**先决条件**

- 您已设置了 SSH 密钥，用于从控制节点访问将配置 **tlog** 系统角色的目标系统。
- 您至少有一个要配置 **tlog** 系统角色的系统。

- Ansible Core 软件包安装在控制机器上。
- **rhel-system-roles** 软件包安装在控制机器上。

**流程**

1. 使用以下内容创建新的 ***playbook.yml*** 文件：

```
---
- name: Deploy session recording
  hosts: all
  vars:
    tlog_scope_sssd: some
    tlog_users_sssd:
      - recorded-user

  roles:
    - rhel-system-roles.tlog
```

其中,

- **tlog_scope_sssd**:
  - **some** 指定您只记录某些用户和组，而不是 **all** 或 **none**。
- **tlog_users_sssd**:
  - **recorded-user** 指定您要记录会话的用户。请注意，这不会为您添加用户。您必须自行设置该用户。

2. 另外，还可以验证 playbook 语法。

```
# ansible-playbook --syntax-check playbook.yml
```

3. 在清单文件上运行 playbook:

```
# ansible-playbook -i IP_Address /path/to/file/playbook.yml -v
```

因此，playbook 会在您指定的系统中安装 **tlog** RHEL 系统角色。该角色包括 **tlog-rec-session** （终端会话 I/O 日志记录程序），它充当用户的登录 shell。它还创建一个 SSSD 配置丢弃文件，可供您定义的用户和组使用。SSSD 解析并读取这些用户和组，并使用 **tlog-rec-session** 替换其用户 shell。另外，如果系统上安装了 **cockpit** 软件包，playbook 也会安装 **cockpit-session-recording** 软件包，它是一个 **Cockpit** 模块，供您在 web 控制台界面中查看和播放记录。

**验证步骤**

要验证 SSSD 配置文件是否在系统中创建了，请执行以下步骤：

1. 进入创建 SSSD 配置丢弃文件的文件夹：

```
# cd /etc/sssd/conf.d
```

2. 检查文件内容：

```
# cat /etc/sssd/conf.d/sssd-session-recording.conf
```

您可以看到该文件包含您在 playbook 中设置的参数。

## 24.4. 为排除组或用户列表部署 TLOG RHEL 系统角色

您可以使用 **tlog** 系统角色支持 SSSD 会话记录配置选项 **exclude_users** 和 **exclude_groups**。按照以下步骤准备和应用 Ansible playbook，来配置 RHEL 系统，以便在 systemd 日志中排除用户或组的会话记录。

**先决条件**

- 您已设置了从控制节点访问您要配置 **tlog** 系统角色的目标系统的 SSH 密钥。
- 您至少有一个要在其上配置 **tlog** 系统角色的系统。
- Ansible Core 软件包安装在控制机器上。
- **rhel-system-roles** 软件包安装在控制机器上。

**流程**

1. 使用以下内容创建新的 ***playbook.yml*** 文件：

```
---
- name: Deploy session recording excluding users and groups
  hosts: all
  vars:
    tlog_scope_sssd: all
    tlog_exclude_users_sssd:
      - jeff
      - james
    tlog_exclude_groups_sssd:
      - admins

  roles:
    - rhel-system-roles.tlog
```

其中,

- **tlog_scope_sssd**:
  - **all** ：指定您要记录所有用户和组。
- **tlog_exclude_users_sssd**:
  - user names：指定您要从会话记录中排除的用户的用户名。
- **tlog_exclude_groups_sssd**:
  - **admins** 指定要从会话记录中排除的组。

2. （可选）验证 playbook 语法；

```
# ansible-playbook --syntax-check playbook.yml
```

3. 在清单文件上运行 playbook:

```
# ansible-playbook -i IP_Address /path/to/file/playbook.yml -v
```

因此，playbook 会在您指定的系统中安装 **tlog** RHEL 系统角色。该角色包括 **tlog-rec-session** （终端会话 I/O 日志记录程序），它充当用户的登录 shell。它还会创建一个 **/etc/sssd/conf.d/sssd-session-recording.conf** SSSD 配置丢弃文件，供用户和组使用，但您定义为排除的用户和组除外。SSSD 解析并读取这些用户和组，并使用 **tlog-rec-session** 替换其用户 shell。另外，如果系统上安装了 **cockpit** 软件包，playbook 也会安装 **cockpit-session-recording** 软件包，它是一个 **Cockpit** 模块，供您在 web 控制台界面中查看和播放记录。

**验证步骤**

要验证 SSSD 配置文件是否在系统中创建了，请执行以下步骤：

1. 进入创建 SSSD 配置丢弃文件的文件夹：

```
# cd /etc/sssd/conf.d
```

2. 检查文件内容：

```
# cat sssd-session-recording.conf
```

您可以看到该文件包含您在 playbook 中设置的参数。

**其他资源**

- 请参阅 **/usr/share/doc/rhel-system-roles/tlog/** 和 **/usr/share/ansible/roles/rhel-system-roles.tlog/** 目录。
- 在 CLI 中使用部署的 Terminal Session Recording 系统角色记录会话 。

## 24.5. 使用在 CLI 中部署的 TLOG 系统角色记录会话

在您指定的系统中部署了 **tlog** 系统角色后，就可以使用命令行界面 (CLI) 记录用户终端会话。

**先决条件**

- 您已在目标系统中部署了 **tlog** 系统角色。
- SSSD 配置丢弃文件在 **/etc/sssd/conf.d** 目录中创建。请参阅部署 Terminal Session Recording RHEL 系统角色。

**流程**

1. 创建一个用户并为这个用户分配密码：

```
# useradd recorded-user
# passwd recorded-user
```

2. 以您刚刚创建的用户身份登录到该系统：

```
# ssh recorded-user@localhost
```

3. 当系统提示您输入 yes 或 no 进行身份验证时请输入 "yes"。

4. 插入 *record-user* 的密码。
   系统显示有关记录会话的信息。

   ```
   ATTENTION! Your session is being recorded!
   ```

5. 记录完会话后，键入：

   ```
   # exit
   ```

   系统从用户注销并关闭与本地主机的连接。

用户会话会被记录，并被保存，您可以使用 journal 进行播放。

**验证步骤**

要在日志中查看您记录的会话，请执行以下步骤：

1. 运行以下命令：

   ```
   # journalctl -o verbose -r
   ```

2. 搜索 **tlog-rec** 记录的日志条目的 **MESSAGE** 字段。

   ```
   # journalctl -xel _EXE=/usr/bin/tlog-rec-session
   ```

## 24.6. 使用 CLI 监视记录的会话

您可以使用命令行界面（CLI）从日志中执行用户会话记录。

**先决条件**

- 您已经记录了一个用户会话。请参阅 使用 CLI 中部署的 tlog 系统角色来记录会话。

**流程**

1. 在 CLI 终端中，播放用户会话记录：

   ```
   # journalctl -o verbose -r
   ```

2. 搜索 **tlog** 记录：

   ```
   $ /tlog-rec
   ```

   您可以查看详情，例如：

   - 用户会话记录的用户名
   - **out_txt** 字段，这是记录的会话的原始输出编码
   - 标识符号 TLOG_REC=*ID_number*

3. 复制标识符号 TLOG_REC=*ID_number*。

4. 使用标识符号 TLOG_REC=*ID_number* 回放记录。

```
# tlog-play -r journal -M TLOG_REC=ID_number
```

您可以看到记录的用户会话被回放。

# 第 25 章 使用 HA_CLUSTER RHEL 系统角色配置高可用性集群

使用 **ha_cluster** 系统角色，您可以配置和管理使用 Pacemaker 高可用性集群资源管理器的高可用性集群。

## 25.1. HA_CLUSTER 系统角色变量

在 **ha_cluster** 系统角色 playbook 中，根据集群部署的要求为高可用性集群定义变量。

您可以为 **ha_cluster** 系统角色设置的变量如下：

**ha_cluster_enable_repos**

启用包含 **ha_cluster** 系统角色所需软件包的存储库的布尔值标志。当此变量被设置为默认值 **true** 时，您在用作集群成员的系统上有活动的 RHEL 和 RHEL High Availability 附加组件的订阅，否则系统角色将失败。

**ha_cluster_manage_firewall**

确定 **ha_cluster** 系统角色是否管理防火墙的布尔值标志。当 **ha_cluster_manage_firewall** 设为 **true** 时，防火墙高可用性服务和**fence-virt** 端口被启用。当 **ha_cluster_manage_firewall** 设为 **false** 时，**ha_cluster** 系统角色不管理防火墙。如果您的系统正在运行 **firewalld** 服务，则必须在 playbook 中将该参数设置为 **true**。
您可以使用 **ha_cluster_manage_firewall** 参数来添加端口，但您无法使用该参数删除端口。要删除端口，请直接使用 **firewall** 系统角色。

从 RHEL 7.9 开始，防火墙不再被默认配置，因为它仅在 **ha_cluster_manage_firewall** 设为 **true** 时才配置。

**ha_cluster_manage_selinux**

确定 **ha_cluster** 系统角色是否使用 **selinux** 系统角色管理属于防火墙高可用性服务的端口的布尔值标志。当 **ha_cluster_manage_selinux** 设为 **true** 时，属于防火墙高可用性服务的端口与 SELinux 端口类型 **cluster_port_t** 相关联。当 **ha_cluster_manage_selinux** 设为 **false** 时，**ha_cluster** 系统角色不管理 SELinux。
如果您的系统正在运行 **selinux** 服务，则必须在 playbook 中将此参数设置为 **true**。防火墙配置是管理 SELinux 的先决条件。如果没有安装防火墙，则管理 SELinux 策略会被跳过。

您可以使用 **ha_cluster_manage_selinux** 参数添加策略，但您无法使用该参数删除策略。要删除策略，请直接使用 **selinux** 系统角色。

**ha_cluster_cluster_present**

布尔值标志，如果设为 **true**，则会根据传递给角色的变量，决定是否在主机上配置 HA 集群。角色中没有指定且不受角色支持的任何集群配置都将丢失。
如果 **ha_cluster_cluster_present** 设为 **false**，则会从目标主机中删除所有 HA 集群配置。

此变量的默认值为 **true**。

以下示例 playbook 删除了 **node1** 和 **node2** 上的所有集群配置

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_present: false

  roles:
    - rhel-system-roles.ha_cluster
```

**ha_cluster_start_on_boot**

确定是否将集群服务配置为在引导时启动的布尔值标志。此变量的默认值为 **true**。

**ha_cluster_fence_agent_packages**

要安装的隔离代理软件包列表。此变量的默认值为 **fence-agents-all**，**fence-virt**。

**ha_cluster_extra_packages**

要安装的其他软件包列表。此变量的默认值是 no packages。
此变量可用于安装角色未自动安装的其他软件包，如自定义资源代理。

可以将隔离代理指定为这个列表的成员。但是，**ha_cluster_fence_agent_packages** 是用于指定隔离代理的推荐的角色变量，因此其默认值会被覆盖。

**ha_cluster_hacluster_password**

指定 **hacluster** 用户的密码的字符串值。**hacluster** 用户对集群具有完全访问权限。建议您加密密码，如 使用 Ansible Vault 加密内容 中所述。没有默认密码值，必须指定此变量。

**ha_cluster_corosync_key_src**

Corosync **authkey** 文件的路径，它是 Corosync 通信的身份验证和加密密钥。强烈建议您对每个集群都有一个唯一的 **authkey** 值。密钥应为 256 字节的随机数据。
如果为此变量指定一个密钥，则建议您使用 vault 加密密钥，如 使用 Ansible Vault 加密内容 中所述。

如果没有指定密钥，则使用节点上已存在的密钥。如果节点没有相同的密钥，则一个节点的密钥将被分发到其他节点，以便所有节点都有相同的密钥。如果节点都没有密钥，则将生成一个新的密钥，并将其分发到节点。

如果设置了此变量，则忽略这个密钥的 **ha_cluster_regenerate_keys**。

此变量的默认值为 null。

**ha_cluster_pacemaker_key_src**

Pacemaker **authkey** 文件的路径，它是 Pacemaker 通信的身份验证和加密密钥。强烈建议您对每个集群都有一个唯一的 **authkey** 值。密钥应为 256 字节的随机数据。
如果为此变量指定一个密钥，则建议您使用 vault 加密密钥，如 使用 Ansible Vault 加密内容 中所述。

如果没有指定密钥，则使用节点上已存在的密钥。如果节点没有相同的密钥，则一个节点的密钥将被分发到其他节点，以便所有节点都有相同的密钥。如果节点都没有密钥，则将生成一个新的密钥，并将其分发到节点。

如果设置了此变量，则忽略这个密钥的 **ha_cluster_regenerate_keys**。

此变量的默认值为 null。

**ha_cluster_fence_virt_key_src**

**fence-virt** 或 **fence-xvm** 预共享密钥文件的路径，它是 **fence-virt** 或 **fence-xvm** 隔离代理验证密钥的位置。
如果为此变量指定一个密钥，则建议您使用 vault 加密密钥，如 使用 Ansible Vault 加密内容 中所述。

如果没有指定密钥，则使用节点上已存在的密钥。如果节点没有相同的密钥，则一个节点的密钥将被分发到其他节点，以便所有节点都有相同的密钥。如果节点都没有密钥，则将生成一个新的密钥，并将其分发到节点。如果 **ha_cluster** 系统角色以这种方式生成新的密钥，您应该将密钥复制到节点的虚拟机监控程序，以确保隔离正常工作。

如果设置了此变量，则忽略这个密钥的 **ha_cluster_regenerate_keys**。

此变量的默认值为 null。

**ha_cluster_pcsd_public_key_srcr, ha_cluster_pcsd_private_key_src**

**pcsd** TLS 证书和私钥的路径。如果没有指定，则使用节点上已存在的证书密钥对。如果没有证书密钥对，则会生成一个随机的新密钥对。
如果为此变量指定了私钥值，则建议您使用 vault 加密密钥，如 使用 Ansible Vault 加密内容 中所述。

如果设置了这些变量，则将忽略此证书密钥对的 **ha_cluster_regenerate_keys**。

这些变量的默认值为 null。

**ha_cluster_pcsd_certificates**

使用 **certificate** 系统角色创建 **pcsd** 私钥和证书。
如果您的系统没有使用 **pcsd** 私钥和证书配置，则您可以使用以下两种方式之一创建它们：

- 设置 **ha_cluster_pcsd_certificates** 变量。当您设置 **ha_cluster_pcsd_certificates** 变量时，**certificate** 系统角色会在内部使用，它会如定义的那样为 **pcsd** 创建私钥和证书。
- 不要设置 **ha_cluster_pcsd_public_key_src**、**ha_cluster_pcsd_private_key_src** 或 **ha_cluster_pcsd_certificates** 变量。如果没有设置这些变量，则 **ha_cluster** 系统角色将使用 **pcsd** 本身创建 **pcsd** 证书。**ha_cluster_pcsd_certificates** 的值被设置为 **certificate_requests** 的值，如 **certificate** 系统角色中指定的那样。有关 **certificate** 系统角色的更多信息，请参阅 使用 RHEL 系统角色请求证书。

以下操作注意事项适用于 **ha_cluster_pcsd_certificate** 变量的使用：

- 除非您使用 IPA 并将系统加入 IPA 域，否则 **certificate** 系统角色会创建自签名证书。在这种情况下，您必须在 RHEL 系统角色上下文之外明确配置信任设置。系统角色不支持配置信任设置。
- 当您设置 **ha_cluster_pcsd_certificates** 变量时，不要设置 **ha_cluster_pcsd_public_key_src** 和 **ha_cluster_pcsd_private_key_src** 变量。
- 当您设置 **ha_cluster_pcsd_certificates** 变量时，此证书-密钥对会忽略 **ha_cluster_regenerate_keys**。

此变量的默认值为 **[]**。
有关在高可用性集群中创建 TLS 证书和密钥文件的 **ha_cluster** 系统角色 playbook 的示例，请参阅 为高可用性集群创建 pcsd TLS 证书和密钥文件。

**ha_cluster_regenerate_keys**

布尔值标志，当设为 **true** 时，决定将重新生成预共享密钥和 TLS 证书。有关重新生成密钥和证书的更多信息，请参阅 **ha_cluster_corosync_key_src**、**ha_cluster_pacemaker_key_src** **ha_cluster_fence_virt_key_src**、**ha_cluster_pcsd_public_key_src** 和**ha_cluster_pcsd_private_key_src** 变量的描述。

此变量的默认值为 **false**。

**ha_cluster_pcs_permission_list**

配置使用 **pcsd** 管理集群的权限。您使用这个变量配置的项目如下：

- **type** - 用户 或 组
- **name** - 用户或组名称

- **allow_list** - 对指定的用户或组允许的操作：
  - **read** - 查看集群状态和设置
  - **write** - 修改集群设置，权限和 ACL 除外
  - **grant** - 修改集群权限和 ACL
  - **full** - 对集群的无限制访问，包括添加和删除节点，以及访问密钥和证书

**ha_cluster_pcs_permission_list** 变量的结构及其默认值如下：

```
ha_cluster_pcs_permission_list:
  - type: group
    name: hacluster
    allow_list:
      - grant
      - read
      - write
```

**ha_cluster_cluster_name**

集群的名称。这是一个字符串值，默认值为 **my-cluster**。

**ha_cluster_transport**

设置集群传输方法。您使用这个变量配置的项目如下：

- **type** (可选) - 传输类型：**knet**, **udp**, 或 **udpu**。**udp** 和 **udpu** 传输类型只支持一个链接。对于 **udp** 和 **udpu**，始终禁用加密。若未指定，则默认为 **knet**。
- **options** （可选）- 带有传输选项的"名称-值"的字典列表。
- **links** （可选）- "名称-值"的字典列表。每个 name-value 字典列表都包含适用于一个 Corosync 链接的选项。建议您为每个链接设置 **linknumber** 值。否则，第一个字典列表被默认分配给第一个链接，第二个分配给第二个链接，以此类推。
- **compression** （可选）- 配置传输压缩的 name-value 字典列表。仅支持 **knet** 传输类型。
- **crypto** （可选）- 配置传输加密的 name-value 字典列表。默认情况下启用加密。仅支持 **knet** 传输类型。
  有关允许的选项列表，请查看 **pcs -h cluster setup** 帮助页或 **pcs**(8) man page 的 **cluster** 部分中的 **setup** 描述。有关更详细的描述，请查看 **corosync.conf**(5) man page。

  **ha_cluster_transport** 变量的结构如下：

```
ha_cluster_transport:
  type: knet
  options:
    - name: option1_name
      value: option1_value
    - name: option2_name
      value: option2_value
  links:
    -
      - name: option1_name
        value: option1_value
      - name: option2_name
```

```
      value: option2_value
  -
    - name: option1_name
      value: option1_value
    - name: option2_name
      value: option2_value
compression:
  - name: option1_name
    value: option1_value
  - name: option2_name
    value: option2_value
crypto:
  - name: option1_name
    value: option1_value
  - name: option2_name
    value: option2_value
```

有关配置传输方法的 **ha_cluster** 系统角色 playbook 的示例，请参阅 在高可用性集群中配置 Corosync 值。

**ha_cluster_totem**

配置 Corosync totem。有关允许的选项列表，请查看 **pcs -h cluster setup** 帮助页或 **pcs**(8) man page 的 **cluster** 部分中的 **setup** 描述。有关更详细的说明，请查看 **corosync.conf**(5) man page。**ha_cluster_totem** 变量的结构如下：

```
ha_cluster_totem:
  options:
    - name: option1_name
      value: option1_value
    - name: option2_name
      value: option2_value
```

有关配置 Corosync totem 的 **ha_cluster** 系统角色 playbook 示例，请参阅 在高可用性集群中配置 Corosync 值。

**ha_cluster_quorum**

配置集群仲裁。您可以为集群仲裁配置以下项目：

- **options** （可选）- 配置仲裁的名称-值字典的列表。允许的选项有：**auto_tie_breaker**、**last_man_standing**、**last_man_standing_window** 和 **wait_for_all**。有关仲裁选项的详情，请查看 **votequorum**(5)手册页。

- **device** （可选）-

配置使用仲裁设备的集群。默认情况下，不使用仲裁设备。** **model**（必需的）- 指定仲裁设备模型。仅支持 **net**

+ ** **model_options** （可选）- 配置指定仲裁设备模型的名称-值字典的列表。对于型号 **net**，您必须指定 **host** 和 **algorithm** 选项。

+ 使用 **pcs-address** 选项设置自定义 **pcsd** 地址和端口来连接到 **qnetd** 主机。如果没有指定这个选项，角色会连接到 **主机** 上的默认 **pcsd** 端口。

+ ** **generic_options** （可选）- 设置不是特定于型号的仲裁设备选项的名称-值字典的列表。

+ ** **heuristics_options**（可选）- 配置仲裁设备启发式的名称-值字典的列表。

+ 有关仲裁设备选项的详情，请查看 **corosync-qdevice**(8)手册页。通用选项为 **sync_timeout** 和 **timeout**。有关型号 **net** 选项，请查看 **quorum.device.net** 部分。有关启发式选项，请查看 **quorum.device.heuristics** 部分。

+ 要重新生成仲裁设备 TLS 证书，请将 **ha_cluster_regenerate_keys** 变量设置为 **true**。

+:: **ha_cluster_quorum** 变量的结构如下：

+

```
ha_cluster_quorum:
  options:
    - name: option1_name
      value: option1_value
    - name: option2_name
      value: option2_value
  device:
    model: string
    model_options:
      - name: option1_name
        value: option1_value
      - name: option2_name
        value: option2_value
    generic_options:
      - name: option1_name
        value: option1_value
      - name: option2_name
        value: option2_value
    heuristics_options:
      - name: option1_name
        value: option1_value
      - name: option2_name
        value: option2_value
```

+ 有关配置集群仲裁的 **ha_cluster** 系统角色 playbook 示例，请参阅在 高可用性集群中配置 Corosync 值。有关使用仲裁设备配置集群的 **ha_cluster** 系统角色 playbook 的示例，请参阅 使用仲裁设备配置高可用性集群。

**ha_cluster_sbd_enabled**

决定集群是否可以使用 SBD 节点隔离机制的布尔值标志。此变量的默认值为 **false**。
有关启用 SBD 的 **ha_cluster** 系统角色 playbook 示例，请参阅 使用 SBD 节点隔离配置高可用性集群。

**ha_cluster_sbd_options**

指定 SBD 选项的名称-值字典的列表。支持的选项包括：

- **delay-start** - 默认为 **no**
- **startmode** - 默认为 **always**
- **timeout-action** - 默认为 **flush,reboot**
- **watchdog-timeout** - 默认为 **5**

有关这些选项的详情，请参考 **sbd**(8)手册页中的 **Configuration via environment** 部分。

有关用于配置 SBD 选项的 **ha_cluster** 系统角色的 playbook 示例，请参阅 使用 SBD 节点隔离配置高可用性集群。
使用 SBD 时，您可以选择性地为清单中的每个节点配置 watchdog 和 SBD 设备。有关在清单文件中配置 watchdog 和 SBD 设备的详情，请参阅 为 ha_cluster 系统角色指定清单。

**ha_cluster_cluster_properties**

Pacemaker 集群范围配置的集群属性集列表。仅支持一组集群属性。
一组集群属性的结构如下：

```
ha_cluster_cluster_properties:
  - attrs:
      - name: property1_name
        value: property1_value
      - name: property2_name
        value: property2_value
```

默认情况下，不设置任何属性。

以下示例 playbook 配置包含 **node1** 和 **node2** 的集群，并设置 **stonith-enabled** 和 **no-quorum-policy** 集群属性。

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_cluster_properties:
      - attrs:
          - name: stonith-enabled
            value: 'true'
          - name: no-quorum-policy
            value: stop

  roles:
    - rhel-system-roles.ha_cluster
```

**ha_cluster_resource_primitives**

此变量定义由系统角色配置的 pacemaker 资源，包括 stonith 资源。您可以为每个资源配置以下项目：

- **id** （必需）- 资源的 ID。
- **agent** (必需)- 资源或 stonith 代理的名称，如 **ocf:pacemaker:Dummy** 或 **stonith:fence_xvm** 。必须为 stonith 代理指定 **stonith:** 。对于资源代理，可以使用短名称，如 **Dummy**，而不是 **ocf:pacemaker:Dummy**。但是，如果安装了多个具有相同短名称的代理，则角色将失败，因为它将无法决定应使用哪个代理。因此，建议您在指定资源代理时使用全名。
- **instance_attrs** （可选） - 资源的实例属性集合列表。目前，只支持一个集合。属性的确切名称和值，以及它们是否为必选的，这取决于资源还是 stonith 代理。
- **meta_attrs** （可选） - 资源的元属性集合列表。目前，只支持一个集合。

- **operations** （可选） - 资源操作列表。
  - **action** （必需）- pacemaker 以及资源或 stonith 代理定义的操作。
  - **attrs** （必需）- 操作选项，必须至少指定一个选项。

使用 **ha_cluster** 系统角色配置的资源定义的结构如下：

```
- id: resource-id
  agent: resource-agent
  instance_attrs:
    - attrs:
        - name: attribute1_name
          value: attribute1_value
        - name: attribute2_name
          value: attribute2_value
  meta_attrs:
    - attrs:
        - name: meta_attribute1_name
          value: meta_attribute1_value
        - name: meta_attribute2_name
          value: meta_attribute2_value
  operations:
    - action: operation1-action
      attrs:
        - name: operation1_attribute1_name
          value: operation1_attribute1_value
        - name: operation1_attribute2_name
          value: operation1_attribute2_value
    - action: operation2-action
      attrs:
        - name: operation2_attribute1_name
          value: operation2_attribute1_value
        - name: operation2_attribute2_name
          value: operation2_attribute2_value
```

默认情况下不定义任何资源。

有关包含资源配置的 **ha_cluster** 系统角色 playbook 的示例，请参阅 配置具有隔离和资源的高可用性集群。

**ha_cluster_resource_groups**

此变量定义由系统角色配置的 pacemaker 资源组。您可以为每个资源组配置以下项目：

- **id** （必需）- 组的 ID.
- **resources** （必需） - 组的资源列表.每个资源通过其 ID 引用，资源必须在 **ha_cluster_resource_primitives** 变量中定义。必须至少列出一个资源。
- **meta_attrs** （可选） - 组的元属性集合列表。目前，只支持一个集合。

使用 **ha_cluster** 系统角色配置的资源组定义的结构如下：

```
ha_cluster_resource_groups:
  - id: group-id
```

```
    resource_ids:
      - resource1-id
      - resource2-id
    meta_attrs:
      - attrs:
          - name: group_meta_attribute1_name
            value: group_meta_attribute1_value
          - name: group_meta_attribute2_name
            value: group_meta_attribute2_value
```

默认情况下，不定义任何资源组。

有关包含资源组配置的 **ha_cluster** 系统角色 playbook 的示例，请参阅 配置具有隔离和资源的高可用性集群。

**ha_cluster_resource_clones**

此变量定义由系统角色配置的 pacemaker 资源克隆。您可以为资源克隆配置以下项目：

- **resource_id** （必需）- 要克隆的资源.资源必须在 **ha_cluster_resource_primitives** 变量或 **ha_cluster_resource_groups** 变量中定义。
- **promotable**（可选）- 表示要创建的资源克隆是否是可升级的克隆，用 **true** 或 **false** 表示。
- **id** （可选）- 克隆的自定义 ID。如果未指定 ID，将会生成它。如果集群不支持这个选项，则会显示一个警告。
- **meta_attrs** （可选）- 克隆的元属性集合列表。目前，只支持一个集合。

使用 **ha_cluster** 系统角色配置的资源克隆定义的结构如下。

```
ha_cluster_resource_clones:
  - resource_id: resource-to-be-cloned
    promotable: true
    id: custom-clone-id
    meta_attrs:
      - attrs:
          - name: clone_meta_attribute1_name
            value: clone_meta_attribute1_value
          - name: clone_meta_attribute2_name
            value: clone_meta_attribute2_value
```

默认情况下，没有定义资源克隆。

有关包含资源克隆配置的 **ha_cluster** 系统角色 playbook 的示例，请参阅 配置具有隔离和资源的高可用性集群。

**ha_cluster_constraints_location**

此变量定义资源位置限制。资源位置限制表示资源可在哪些节点上运行。您可以指定资源 ID 或模式匹配的资源，它们可以匹配多个资源。您可以通过节点名称或规则指定节点。
您可以为资源位置约束配置以下项目：

- **resource** （必需）- 约束应用到的资源规格。
- **node** （必需）- 资源应首选或避免的节点的名称。

- **id** （可选）- 约束 ID。如果未指定，它将自动生成。
- **options** （可选）- "名称-值"字典列表。
  - **score** - 设置约束的权重。
    - 正 **score** 值表示资源首选在节点上运行。
    - 负 **score** 值表示资源应避免在节点上运行。
    - **score** 值为 **-INFINITY** 表示资源必须避免在节点上运行。
    - 如果没有指定 **score**，分数值默认为 **INFINITY**。

默认情况下，没有定义资源位置限制。
指定资源 ID 和节点名称的资源位置约束的结构如下：

```
ha_cluster_constraints_location:
  - resource:
      id: resource-id
    node: node-name
    id: constraint-id
    options:
      - name: score
        value: score-value
      - name: option-name
        value: option-value
```

您为资源位置约束配置的项目，用于指定资源模式是为资源位置约束配置的相同项目，用于指定资源 ID，但资源规格本身除外。您为资源规格指定的项目如下：

- **pattern** （必需）- POSIX 扩展正则表达式资源 ID 与.

指定资源模式和节点名称的资源位置约束结构如下：

```
ha_cluster_constraints_location:
  - resource:
      pattern: resource-pattern
    node: node-name
    id: constraint-id
    options:
      - name: score
        value: score-value
      - name: resource-discovery
        value: resource-discovery-value
```

您可以为指定资源 ID 和规则的资源位置约束配置以下项目：

- **resource** （必需）- 约束应用到的资源规格。
  - **id** （必需）- 资源 ID。
  - **role**（可选）- 约束限制的资源角色：**Started**、**Unpromoted**,**Promoted**。
- **rule** （必需）- 使用 **pcs** 语法编写的 Constraint 规则。如需更多信息，请参阅 **pcs**(8)man page **的约束位置** 部分。

- 指定的其他项目的含义与未指定规则的资源约束相同。

指定资源 ID 和规则的资源位置约束的结构如下：

```
ha_cluster_constraints_location:
  - resource:
      id: resource-id
      role: resource-role
    rule: rule-string
    id: constraint-id
    options:
      - name: score
        value: score-value
      - name: resource-discovery
        value: resource-discovery-value
```

为资源位置约束配置的项目，用于指定资源模式，规则是用于资源位置约束的相同项目，用于指定资源 ID 和规则，但资源规格本身除外。您为资源规格指定的项目如下：

- **pattern** （必需）- POSIX 扩展正则表达式资源 ID 与.

指定资源模式和规则的资源位置约束结构如下：

```
ha_cluster_constraints_location:
  - resource:
      pattern: resource-pattern
      role: resource-role
    rule: rule-string
    id: constraint-id
    options:
      - name: score
        value: score-value
      - name: resource-discovery
        value: resource-discovery-value
```

有关创建具有资源限制的集群的 **ha_cluster** 系统角色 playbook 示例 ， 请参阅使用资源限制配置高可用性集群。

**ha_cluster_constraints_colocation**

此变量定义资源 colocation 约束。资源共存限制表示一个资源的位置取决于另一个资源的位置。存在两种类型的 colocation 约束：两个资源的一个简单 colocation 约束，并为多个资源设置 colocation 约束。
您可以为简单资源托管约束配置以下项目：

- **resource_follower** (必需)- 应相对于 **resource_leader** 的资源。
  - **id** （必需）- 资源 ID。
  - **role**（可选）- 约束限制的资源角色：**Started**、**Unpromoted**,**Promoted**。
- **resource_leader** （必需）- 集群将决定优先放置此资源的位置，然后决定放置 **resource_follower** 的位置。
  - **id** （必需）- 资源 ID。

    - **role**（可选）- 约束限制的资源角色：**Started**、**Unpromoted**,**Promoted**。
- **id**（可选）- 约束 ID。如果未指定，它将自动生成。
- **options**（可选）- "名称-值"字典列表。
    - **score** - 设置约束的权重。
        - 正 **score** 值表示资源应该在同一节点上运行。
        - 负 **score** 值表示资源应在不同的节点上运行。
        - **score** 值为 **+INFINITY** 表示资源必须在同一节点上运行。
        - **score** 值为 **-INFINITY** 表示资源必须在不同的节点上运行。
        - 如果没有指定 **score**，分数值默认为 **INFINITY**。

默认情况下，没有定义资源 colocation 约束。
简单资源 colocation 约束的结构如下：

```
ha_cluster_constraints_colocation:
  - resource_follower:
      id: resource-id1
      role: resource-role1
    resource_leader:
      id: resource-id2
      role: resource-role2
    id: constraint-id
    options:
      - name: score
        value: score-value
      - name: option-name
        value: option-value
```

您可以为资源集托管约束配置以下项目：

- **resource_sets**（必需）- 资源集合列表。
    - **resource_ids**（必需）- 资源列表。
    - **options**（可选）- "名称/值"字典列表精细调整集合中资源如何被约束处理。
- **id**（可选）- Same 值作为简单 colocation 约束。
- **options**（可选）- Same 值作为简单 colocation 约束。

资源集 colocation 约束的结构如下：

```
ha_cluster_constraints_colocation:
  - resource_sets:
      - resource_ids:
          - resource-id1
          - resource-id2
        options:
          - name: option-name
```

```
      value: option-value
  id: constraint-id
  options:
    - name: score
      value: score-value
    - name: option-name
      value: option-value
```

有关创建具有资源限制的集群的 **ha_cluster** 系统角色 playbook 示例，请参阅使用资源限制配置高可用性集群。

**ha_cluster_constraints_order**

此变量定义资源顺序约束。资源顺序限制表示应发生某些资源操作的顺序。有两种资源顺序约束：两个资源的简单顺序约束，以及多个资源的设置顺序约束。
您可以为简单资源顺序约束配置以下项目：

- **resource_first** (必需)- **resource_then** 资源依赖的资源。
  - **id** （必需）- 资源 ID。
  - **action** （可选）- 在为 **resource_then** 资源启动操作前必须完成的操作。允许的值：**start**、**stop**、**promote**、**demote**。
- **resource_then** （必需）- 依赖资源。
  - **id** （必需）- 资源 ID。
  - **action** （可选）- 资源只能在 **resource_first** 资源执行操作后执行的操作。允许的值：**start**、**stop**、**promote**、**demote**。
- **id** （可选）- 约束 ID。如果未指定，它将自动生成。
- **options** （可选）- “名称-值”字典列表。

默认情况下，没有定义资源顺序限制。
简单资源顺序约束的结构如下：

```
ha_cluster_constraints_order:
  - resource_first:
      id: resource-id1
      action: resource-action1
    resource_then:
      id: resource-id2
      action: resource-action2
    id: constraint-id
    options:
      - name: score
        value: score-value
      - name: option-name
        value: option-value
```

您可以为资源集顺序约束配置以下项目：

- **resource_sets** （必需）- 资源集合列表。
  - **resource_ids** （必需）- 资源列表。

    - **options** （可选）- “名称/值”字典列表精细调整集合中资源如何被约束处理。
  - **id** （可选）- 相同值作为简单顺序约束。
  - **options** （可选）- 相同值作为简单顺序约束。

资源集顺序约束的结构如下：

```
ha_cluster_constraints_order:
  - resource_sets:
      - resource_ids:
          - resource-id1
          - resource-id2
        options:
          - name: option-name
            value: option-value
    id: constraint-id
    options:
      - name: score
        value: score-value
      - name: option-name
        value: option-value
```

有关创建具有资源约束的集群的 **ha_cluster** 系统角色 playbook 是示例，请参阅 配置具有资源约束的高可用性集群。

**ha_cluster_constraints_ticket**

此变量定义资源 ticket 约束。资源票据限制表示依赖于特定票据的资源。有两种类型的资源 ticket 约束：一个资源的简单 ticket 约束，多个资源的 ticket 顺序约束。
您可以为简单资源票据约束配置以下项目：

- **resource** （必需）- 约束应用到的资源规格。
  - **id** （必需）- 资源 ID。
  - **role**（可选）- 约束限制的资源角色：**Started**、**Unpromoted**,**Promoted**。
- **ticket** (必需)- 资源所依赖的票据的名称。
- **id** （可选）- 约束 ID。如果未指定，它将自动生成。
- **options** （可选）- “名称-值”字典列表。
  - **loss-policy** （可选）- 在撤销 ticket 时要对资源执行的操作。

默认情况下，没有定义资源 ticket 约束。
简单资源票据约束的结构如下：

```
ha_cluster_constraints_ticket:
  - resource:
      id: resource-id
      role: resource-role
    ticket: ticket-name
    id: constraint-id
    options:
```

```
    - name: loss-policy
      value: loss-policy-value
    - name: option-name
      value: option-value
```

您可以为资源集票据约束配置以下项目：

- **resource_sets** （必需）- 资源集合列表。
  - **resource_ids** （必需）- 资源列表。
  - **options** （可选）- "名称/值"字典列表精细调整集合中资源如何被约束处理。
- **ticket** (必需)- 相同值作为一个简单 ticket 约束。
- **id** （可选）- 相同值作为简单票据约束。
- **选项** （可选）- 相同值作为简单票据约束。

资源集 ticket 约束的结构如下：

```
ha_cluster_constraints_ticket:
  - resource_sets:
      - resource_ids:
          - resource-id1
          - resource-id2
        options:
          - name: option-name
            value: option-value
    ticket: ticket-name
    id: constraint-id
    options:
      - name: option-name
        value: option-value
```

有关创建具有资源限制的集群的 **ha_cluster** 系统角色 playbook 示例，请参阅使用资源限制配置高可用性集群。

**ha_cluster_qnetd**

(RHEL 8.8 及更高版本)此变量配置 **qnetd** 主机，然后其可以作为集群的外部仲裁设备。
您可以为 **qnetd** 主机配置以下项目：

- **present** （可选）- 如果为 **true**，则在主机上配置 **qnetd** 实例。如果为 **false**，请从主机中删除 **qnetd** 配置。默认值为 **false**。如果将其设置为 **true**，则必须将 **ha_cluster_cluster_present** 设置为 **false**。
- **start_on_boot**（可选）- 配置 **qnetd** 实例是否应在引导时自动启动。默认值为 **true**。
- **regenerate_keys**（可选）- 将此变量设置为 **true** 以重新生成 **qnetd** TLS 证书。如果重新生成了证书，则必须重新运行角色，以便每个集群再次连接到 **qnetd** 主机，或者手动运行 **pcs**。

您无法在集群节点上运行 **qnetd**，因为隔离会破坏 **qnetd** 操作。
有关使用仲裁设备配置集群的 **ha_cluster** 系统角色 playbook 的示例，请参阅 使用仲裁设备配置集群。

## 25.2. 为 HA_CLUSTER 系统角色指定清单

使用 **ha_cluster** 系统角色 playbook 配置 HA 集群时，您可以在清单中为集群配置节点的名称和地址。

### 25.2.1. 在清单中配置节点名称和地址

对于清单中的每个节点，您可以选择指定以下项目：

- **node_name** - 集群中节点的名称。
- **pcs_address** - **pcs** 用于与节点进行通信的地址。它可以是名称、FQDN 或 IP 地址，并且可以包含端口号。
- **corosync_addresses** - Corosync 使用的地址列表.组成特定集群的所有节点必须具有相同数量的地址，并且地址的顺序也很重要。

以下示例显示了一个具有目标 **node1** 和 **node2** 的清单。**node1** 和 **node2** 必须是完全限定域名，或者必须能够连接到节点，例如，名称可以通过 **/etc/hosts** 文件解析。

```
all:
  hosts:
    node1:
      ha_cluster:
        node_name: node-A
        pcs_address: node1-address
        corosync_addresses:
          - 192.168.1.11
          - 192.168.2.11
    node2:
      ha_cluster:
        node_name: node-B
        pcs_address: node2-address:2224
        corosync_addresses:
          - 192.168.1.12
          - 192.168.2.12
```

### 25.2.2. 在清单中配置 watchdog 和 SBD 设备

使用 SBD 时，您可以选择性地为清单中的每个节点配置 watchdog 和 SBD 设备。虽然所有 SBD 设备都必须与所有节点共享，但每个节点也可以与设备使用不同的名称。每个节点的 watchdog 设备也可以不同。有关您可以在系统角色 playbook 中设置 SBD 变量的信息，请参阅 **ha_cluster** 系统角色变量中的 **ha_cluster_sbd_enabled** 和 **ha_cluster_sbd_options** 部分。

对于清单中的每个节点，您可以选择指定以下项目：

- **sbd_watchdog** - SBD 使用的 Watchdog 设备。如果没有设置，则默认为 **/dev/watchdog**。
- **sbd_devices** - 用于交换 SBD 信息和监控的设备。如果没有设置，则默认为空列表。

以下示例显示了为目标 **node1** 和 **node2** 配置 watchdog 和 SBD 设备的清单。

```
all:
  hosts:
    node1:
      ha_cluster:
```

```
    sbd_watchdog: /dev/watchdog2
    sbd_devices:
      - /dev/vdx
      - /dev/vdy
  node2:
    ha_cluster:
      sbd_watchdog: /dev/watchdog1
      sbd_devices:
        - /dev/vdw
        - /dev/vdz
```

## 25.3. 为高可用性集群创建 PCSD TLS 证书和密钥文件

您可以使用 **ha_cluster** 系统角色在高可用性集群中创建 TLS 证书和密钥文件。运行此 playbook 时，**ha_cluster** 系统角色会在内部使用 **certificate** 系统角色来管理 TLS 证书。

### 先决条件

- **ansible-core** 和 **rhel-system-roles** 软件包已安装在您要运行 playbook 的节点上。

**注意**

您不需要在集群成员节点上安装 **ansible-core**。

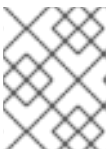
- 作为集群成员运行的系统必须拥有对 RHEL 和 RHEL 高可用性附加组件的有效订阅。

**警告**

**ha_cluster** 系统角色替换指定节点上的任何现有集群配置。没有在角色中指定的任何设置都将丢失。

### 流程

1. 创建一个指定集群中节点的清单文件，如在 为 ha_cluster 系统角色指定一个清单 中所述。
2. 创建一个 playbook 文件，如 **new-cluster.yml**。



**注意**

为生产环境创建 playbook 文件时，vault 会加密密码，如在 使用 Ansible Vault 加密内容 中所述。

以下示例 playbook 文件配置一个运行 **firewalld** 和 **selinux** 服务的集群，并在 **/var/lib/pcsd** 中创建自签名 **pcsd** 证书和私钥文件。**pcsd** 证书具有文件名 **FILENAME.crt**，密钥文件名为 **FILENAME.key**。

```
- hosts: node1 node2
  vars:
```

```
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
    ha_cluster_pcsd_certificates:
      - name: FILENAME
        common_name: "{{ ansible_hostname }}"
        ca: self-sign
  roles:
    - linux-system-roles.ha_cluster
```

3. 保存该文件。

4. 运行 playbook，指定在第 1 步中创建的*清单文件清单*的路径。

```
# ansible-playbook -i inventory new-cluster.yml
```

**其他资源**

- 使用 RHEL 系统角色请求证书

## 25.4. 配置不运行任何资源的高可用性集群

以下流程使用 **ha_cluster** 系统角色，创建没有配置隔离且没有运行任何资源的高可用性集群。

**先决条件**

- 您已在要运行 playbook 的节点上安装了 **ansible-core**。

**注意**

您不需要在集群成员节点上安装 **ansible-core**。

- 您已在要运行 playbook 的系统上安装了 **rhel-system-roles** 软件包。
- 作为集群成员运行的系统必须拥有对 RHEL 和 RHEL 高可用性附加组件的有效订阅。

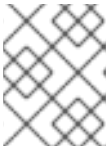
**警告**

**ha_cluster** 系统角色替换指定节点上的任何现有集群配置。没有在角色中指定的任何设置都将丢失。

**流程**

1. 创建一个指定集群中节点的清单文件，如在 为 ha_cluster 系统角色指定一个清单 中所述。

2. 创建一个 playbook 文件，如 **new-cluster.yml**。



**注意**

为生产环境创建 playbook 文件时，vault 会加密密码，如在 使用 Ansible Vault 加密内容 中所述。

以下示例 playbook 文件配置一个运行 **firewalld** 和 **selinux** 服务，而没有配置隔离的集群，且没有运行任何资源。

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true

  roles:
    - rhel-system-roles.ha_cluster
```

3. 保存该文件。
4. 运行 playbook，指定在第 1 步中创建的*清单文件清单*的路径。

```
# ansible-playbook -i inventory new-cluster.yml
```

## 25.5. 配置带有隔离和资源的高可用性集群

以下流程使用 **ha_cluster** 系统角色创建包括隔离设备、集群资源、资源组和克隆资源的高可用性集群。

**先决条件**

- 您已在要运行 playbook 的节点上安装了 **ansible-core**。

**注意**

您不需要在集群成员节点上安装 **ansible-core**。

- 您已在要运行 playbook 的系统上安装了 **rhel-system-roles** 软件包。
- 作为集群成员运行的系统必须拥有对 RHEL 和 RHEL 高可用性附加组件的有效订阅。

**警告**

**ha_cluster** 系统角色替换指定节点上的任何现有集群配置。没有在角色中指定的任何设置都将丢失。

**流程**

1. 创建一个指定集群中节点的清单文件，如在 为 ha_cluster 系统角色指定一个清单 中所述。

2. 创建一个 playbook 文件，如 **new-cluster.yml**。

**注意**

为生产环境创建 playbook 文件时，vault 会加密密码，如在 使用 Ansible Vault 加密内容 中所述。

以下示例 playbook 文件配置一个运行 **firewalld** 和 **selinux** 服务的集群。集群包括隔离、多个资源和资源组。它还包含资源组的资源克隆。

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
    ha_cluster_resource_primitives:
      - id: xvm-fencing
        agent: 'stonith:fence_xvm'
        instance_attrs:
          - attrs:
              - name: pcmk_host_list
                value: node1 node2
      - id: simple-resource
        agent: 'ocf:pacemaker:Dummy'
      - id: resource-with-options
        agent: 'ocf:pacemaker:Dummy'
        instance_attrs:
          - attrs:
              - name: fake
                value: fake-value
              - name: passwd
                value: passwd-value
        meta_attrs:
          - attrs:
              - name: target-role
                value: Started
              - name: is-managed
                value: 'true'
        operations:
          - action: start
            attrs:
              - name: timeout
                value: '30s'
          - action: monitor
            attrs:
              - name: timeout
                value: '5'
              - name: interval
                value: '1min'
      - id: dummy-1
        agent: 'ocf:pacemaker:Dummy'
      - id: dummy-2
        agent: 'ocf:pacemaker:Dummy'
      - id: dummy-3
```

```
      agent: 'ocf:pacemaker:Dummy'
    - id: simple-clone
      agent: 'ocf:pacemaker:Dummy'
    - id: clone-with-options
      agent: 'ocf:pacemaker:Dummy'
  ha_cluster_resource_groups:
    - id: simple-group
      resource_ids:
        - dummy-1
        - dummy-2
      meta_attrs:
        - attrs:
            - name: target-role
              value: Started
            - name: is-managed
              value: 'true'
    - id: cloned-group
      resource_ids:
        - dummy-3
  ha_cluster_resource_clones:
    - resource_id: simple-clone
    - resource_id: clone-with-options
      promotable: yes
      id: custom-clone-id
      meta_attrs:
        - attrs:
            - name: clone-max
              value: '2'
            - name: clone-node-max
              value: '1'
    - resource_id: cloned-group
      promotable: yes

roles:
  - rhel-system-roles.ha_cluster
```

3. 保存该文件。

4. 运行 playbook，指定在第 1 步中创建的*清单文件清单*的路径。

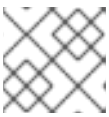
```
# ansible-playbook -i inventory new-cluster.yml
```

## 25.6. 使用资源限制配置高可用性集群

以下流程使用 **ha_cluster** 系统角色创建高可用性集群，其包含资源位置约束、资源 colocation 约束、资源顺序限制和资源票据限制。

**先决条件**

- 您已在要运行 playbook 的节点上安装了 **ansible-core**。



**注意**

您不需要在集群成员节点上安装 **ansible-core**。

- 您已在要运行 playbook 的系统上安装了 **rhel-system-roles** 软件包。
- 作为集群成员运行的系统必须拥有对 RHEL 和 RHEL 高可用性附加组件的有效订阅。

**警告**

**ha_cluster** 系统角色替换指定节点上任何现有的群集配置。没有在角色中指定的任何设置都将丢失。

**流程**

1. 创建一个指定集群中节点的清单文件，如在 为 ha_cluster 系统角色指定一个清单 中所述。
2. 创建一个 playbook 文件，如 **new-cluster.yml**。

**注意**

为生产环境创建 playbook 文件时，vault 会加密密码，如在 使用 Ansible Vault 加密内容 中所述。

以下示例 playbook 文件配置一个运行 **firewalld** 和 **selinux** 服务的集群。集群包括资源位置约束、资源托管约束、资源顺序约束和资源票据约束。

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
    # In order to use constraints, we need resources the constraints will apply
    # to.
    ha_cluster_resource_primitives:
      - id: xvm-fencing
        agent: 'stonith:fence_xvm'
        instance_attrs:
          - attrs:
              - name: pcmk_host_list
                value: node1 node2
      - id: dummy-1
        agent: 'ocf:pacemaker:Dummy'
      - id: dummy-2
        agent: 'ocf:pacemaker:Dummy'
      - id: dummy-3
        agent: 'ocf:pacemaker:Dummy'
      - id: dummy-4
        agent: 'ocf:pacemaker:Dummy'
      - id: dummy-5
        agent: 'ocf:pacemaker:Dummy'
      - id: dummy-6
        agent: 'ocf:pacemaker:Dummy'
```

```
# location constraints
ha_cluster_constraints_location:
  # resource ID and node name
  - resource:
      id: dummy-1
    node: node1
    options:
      - name: score
        value: 20
  # resource pattern and node name
  - resource:
      pattern: dummy-\d+
    node: node1
    options:
      - name: score
        value: 10
  # resource ID and rule
  - resource:
      id: dummy-2
    rule: '#uname eq node2 and date in_range 2022-01-01 to 2022-02-28'
  # resource pattern and rule
  - resource:
      pattern: dummy-\d+
    rule: node-type eq weekend and date-spec weekdays=6-7
# colocation constraints
ha_cluster_constraints_colocation:
  # simple constraint
  - resource_leader:
      id: dummy-3
    resource_follower:
      id: dummy-4
    options:
      - name: score
        value: -5
  # set constraint
  - resource_sets:
      - resource_ids:
          - dummy-1
          - dummy-2
      - resource_ids:
          - dummy-5
          - dummy-6
        options:
          - name: sequential
            value: "false"
    options:
      - name: score
        value: 20
# order constraints
ha_cluster_constraints_order:
  # simple constraint
  - resource_first:
      id: dummy-1
    resource_then:
      id: dummy-6
    options:
```

```
        - name: symmetrical
          value: "false"
    # set constraint
    - resource_sets:
        - resource_ids:
            - dummy-1
            - dummy-2
          options:
            - name: require-all
              value: "false"
            - name: sequential
              value: "false"
        - resource_ids:
            - dummy-3
        - resource_ids:
            - dummy-4
            - dummy-5
          options:
            - name: sequential
              value: "false"
  # ticket constraints
  ha_cluster_constraints_ticket:
    # simple constraint
    - resource:
        id: dummy-1
      ticket: ticket1
      options:
        - name: loss-policy
          value: stop
    # set constraint
    - resource_sets:
        - resource_ids:
            - dummy-3
            - dummy-4
            - dummy-5
      ticket: ticket2
      options:
        - name: loss-policy
          value: fence

roles:
  - linux-system-roles.ha_cluster
```

3. 保存该文件。

4. 运行 playbook，指定在第 1 步中创建的*清单文件清单*的路径。

```
# ansible-playbook -i inventory new-cluster.yml
```

## 25.7. 在高可用性集群中配置 COROSYNC 值

以下流程使用 **ha_cluster** 系统角色来创建配置 Corosync 值的高可用性集群。

**先决条件**

- 您已在要运行 playbook 的节点上安装了 **ansible-core**。

**注意**

您不需要在集群成员节点上安装 **ansible-core**。

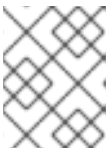
- 您已在要运行 playbook 的系统上安装了 **rhel-system-roles** 软件包。
- 作为集群成员运行的系统必须拥有对 RHEL 和 RHEL 高可用性附加组件的有效订阅。

**警告**

**ha_cluster** 系统角色替换指定节点上的任何现有集群配置。没有在角色中指定的任何设置都将丢失。

**流程**

1. 创建一个指定集群中节点的清单文件，如在 为 ha_cluster 系统角色指定一个清单 中所述。
2. 创建一个 playbook 文件，如 **new-cluster.yml**。



**注意**

为生产环境创建 playbook 文件时，Vault 会加密密码，如 使用 Ansible Vault 加密内容 中所述。

以下示例 playbook 文件配置一个运行 **firewalld** 和 **selinux** 服务的集群，该集群配置 Corosync 属性。

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
    ha_cluster_transport:
      type: knet
      options:
        - name: ip_version
          value: ipv4-6
        - name: link_mode
          value: active
      links:
        -
          - name: linknumber
            value: 1
          - name: link_priority
            value: 5
        -
```

```
        - name: linknumber
          value: 0
        - name: link_priority
          value: 10
    compression:
      - name: level
        value: 5
      - name: model
        value: zlib
    crypto:
      - name: cipher
        value: none
      - name: hash
        value: none
  ha_cluster_totem:
    options:
      - name: block_unlisted_ips
        value: 'yes'
      - name: send_join
        value: 0
  ha_cluster_quorum:
    options:
      - name: auto_tie_breaker
        value: 1
      - name: wait_for_all
        value: 1

roles:
  - linux-system-roles.ha_cluster
```

3. 保存该文件。

4. 运行 playbook，指定在第 1 步中创建的*清单文件清单*的路径。

```
# ansible-playbook -i inventory new-cluster.yml
```

## 25.8. 使用 SBD 节点隔离配置高可用性集群

以下流程使用 **ha_cluster** 系统角色创建使用 SBD 节点隔离的高可用性集群。

**先决条件**

- 您已在要运行 playbook 的节点上安装了 **ansible-core**。

**注意**

您不需要在集群成员节点上安装 **ansible-core**。

- 您已在要运行 playbook 的系统上安装了 **rhel-system-roles** 软件包。
- 作为集群成员运行的系统必须拥有对 RHEL 和 RHEL 高可用性附加组件的有效订阅。

**警告**

**ha_cluster** 系统角色替换指定节点上的任何现有集群配置。没有在角色中指定的任何设置都将丢失。

**流程**

1. 创建一个指定集群中节点的清单文件，如在 为 ha_cluster 系统角色指定一个清单 中所述。您可以选择在清单文件中为集群中的每个节点配置 watchdog 和 SBD 设备。
2. 创建一个 playbook 文件，如 **new-cluster.yml**。

**注意**

为生产环境创建 playbook 文件时，vault 会加密密码，如在 使用 Ansible Vault 加密内容 中所述。

以下示例 playbook 文件配置一个运行 **firewalld** 和 **selinux** 服务的集群，该集群使用 SBD 隔离。

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
    ha_cluster_sbd_enabled: yes
    ha_cluster_sbd_options:
      - name: delay-start
        value: 'no'
      - name: startmode
        value: always
      - name: timeout-action
        value: 'flush,reboot'
      - name: watchdog-timeout
        value: 5

  roles:
    - linux-system-roles.ha_cluster
```

3. 保存该文件。
4. 运行 playbook，指定在第 1 步中创建的*清单文件清单*的路径。

```
# ansible-playbook -i inventory new-cluster.yml
```

## 25.9. 使用仲裁设备配置高可用性集群

要使用 **ha_cluster** 系统角色，使用单独的仲裁设备配置一个高可用性集群，首先要设置仲裁设备。设置仲裁设备后，您可以在任意数量的集群中使用该设备。

### 25.9.1. 配置仲裁设备

要使用 **ha_cluster** 系统角色配置仲裁设备，请按照以下步骤操作。请注意，您不能在集群节点上运行仲裁设备。

**先决条件**

- **ansible-core** 和 **rhel-system-roles** 软件包已安装在您要运行 playbook 的节点上。

**注意**

您不需要在集群成员节点上安装 **ansible-core**。

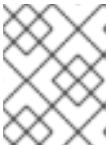
- 您要用来运行仲裁设备的系统有涵盖 RHEL 和 RHEL High Availability 附加组件的有效订阅。

**警告**

**ha_cluster** 系统角色替换指定节点上的任何现有集群配置。没有在角色中指定的任何设置都将丢失。

**流程**

1. 创建一个 playbook 文件，如 **qdev-playbook.yml**。



**注意**

为生产环境创建 playbook 文件时，vault 会加密密码，如在 使用 Ansible Vault 加密内容 中所述。

以下示例 playbook 文件在运行 **firewalld** 和 **selinux** 服务的系统上配置一个仲裁设备。

```
- hosts: nodeQ
  vars:
    ha_cluster_cluster_present: false
    ha_cluster_hacluster_password: password
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
    ha_cluster_qnetd:
      present: true

  roles:
    - linux-system-roles.ha_cluster
```

2. 保存该文件。

3. 运行 playbook，为仲裁设备指定主机节点。

```
# ansible-playbook -i nodeQ, qdev-playbook.yml
```

### 25.9.2. 配置一个集群以使用仲裁设备

要将集群配置为使用仲裁设备，请按照以下步骤操作。

**先决条件**

- 您已在要运行 playbook 的节点上安装了 **ansible-core**。

**注意**

您不需要在集群成员节点上安装 **ansible-core**。

- 您已在要运行 playbook 的系统上安装了 **rhel-system-roles** 软件包。
- 作为集群成员运行的系统必须拥有对 RHEL 和 RHEL 高可用性附加组件的有效订阅。
- 您已配置了一个仲裁设备。

**警告**

**ha_cluster** 系统角色替换指定节点上任何现有的群集配置。没有在角色中指定的任何设置都将丢失。

**流程**

1. 创建一个指定集群中节点的清单文件，如在 为 ha_cluster 系统角色指定一个清单 中所述。
2. 创建一个 playbook 文件，如 **new-cluster.yml**。

**注意**

为生产环境创建 playbook 文件时，vault 会加密密码，如在 使用 Ansible Vault 加密内容 中所述。

以下示例 playbook 文件配置一个运行 **firewalld** 和 **selinux** 服务的集群，该集群使用一个仲裁设备。

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
    ha_cluster_quorum:
      device:
        model: net
```

```
    model_options:
      - name: host
        value: nodeQ
      - name: algorithm
        value: lms

roles:
  - linux-system-roles.ha_cluster
```

3. 保存该文件。

4. 运行 playbook，指定在第 1 步中创建的*清单文件清单*的路径。

```
# ansible-playbook -i inventory new-cluster.yml
```

## 25.10. 使用 HA_CLUSTER 系统角色在高可用性集群中配置 APACHE HTTP 服务器

这个过程使用 **ha_cluster** 系统角色在双节点 Red Hat Enterprise Linux High Availability Add-On 集群中配置主动/被动 Apache HTTP 服务器。

**先决条件**

- 您已在要运行 playbook 的节点上安装了 **ansible-core**。

**注意**

您不需要在集群成员节点上安装 **ansible-core**。

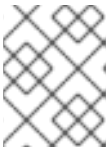
- 您已在要运行 playbook 的系统上安装了 **rhel-system-roles** 软件包。
- 作为集群成员运行的系统必须拥有对 RHEL 和 RHEL 高可用性附加组件的有效订阅。
- 您的系统包括 Apache 需要的一个公共虚拟 IP 地址。
- 您的系统包括集群中节点的共享存储，使用 iSCSI、光纤通道或其他共享网络块的设备。
- 您已配置了具有 XFS 文件系统的 LVM 逻辑卷，如在 在 Pacemaker 集群中配置具有 XFS 文件系统的 LVM 卷 中所述。
- 您已配置了 Apache HTTP 服务器，如 配置 Apache HTTP 服务器 中所述。
- 您的系统包含一个用于隔离群集节点的 APC 电源开关。

**警告**

**ha_cluster** 系统角色替换指定节点上的任何现有集群配置。没有在角色中指定的任何设置都将丢失。

**流程**

1. 创建一个指定集群中节点的清单文件，如在 为 ha_cluster 系统角色指定一个清单 中所述。
2. 创建一个 playbook 文件，如 **http-cluster.yml**。



**注意**

为生产环境创建 playbook 文件时，vault 会加密密码，如在 使用 Ansible Vault 加密内容 中所述。

以下示例 playbook 文件在运行 **firewalld** 和 **selinux** 服务的主动/被动双节点 HA 集群中配置一个之前创建的 Apache HTTP 服务器。

这个示例使用主机名为 **zapc.example.com** 的 APC 电源开关。如果集群不使用任何其他隔离代理，则您可以选择在定义 **ha_cluster_fence_agent_packages** 变量时只列出集群所需的隔离代理。

```
- hosts: z1.example.com z2.example.com
  roles:
    - rhel-system-roles.ha_cluster
  vars:
    ha_cluster_hacluster_password: password
    ha_cluster_cluster_name: my_cluster
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
    ha_cluster_fence_agent_packages:
      - fence-agents-apc-snmp
    ha_cluster_resource_primitives:
      - id: myapc
        agent: stonith:fence_apc_snmp
        instance_attrs:
          - attrs:
              - name: ipaddr
                value: zapc.example.com
              - name: pcmk_host_map
                value: z1.example.com:1;z2.example.com:2
              - name: login
                value: apc
              - name: passwd
                value: apc
      - id: my_lvm
        agent: ocf:heartbeat:LVM-activate
        instance_attrs:
          - attrs:
              - name: vgname
                value: my_vg
              - name: vg_access_mode
                value: system_id
      - id: my_fs
        agent: Filesystem
        instance_attrs:
          - attrs:
              - name: device
                value: /dev/my_vg/my_lv
```

```
        - name: directory
          value: /var/www
        - name: fstype
          value: xfs
    - id: VirtualIP
      agent: IPaddr2
      instance_attrs:
        - attrs:
        - name: ip
          value: 198.51.100.3
        - name: cidr_netmask
          value: 24
    - id: Website
      agent: apache
      instance_attrs:
        - attrs:
        - name: configfile
          value: /etc/httpd/conf/httpd.conf
        - name: statusurl
          value: http://127.0.0.1/server-status
  ha_cluster_resource_groups:
    - id: apachegroup
      resource_ids:
        - my_lvm
        - my_fs
        - VirtualIP
        - Website
```

3. 保存该文件。

4. 运行 playbook，指定在第 1 步中创建的*清单文件清单*的路径。

```
# ansible-playbook -i inventory http-cluster.yml
```

5. 当您使用 **apache** 资源代理来管理 Apache 时，它不会使用 **systemd**。因此，您必须编辑 Apache 提供的 **logrotate** 脚本，使其不使用 **systemctl** 重新加载 Apache。
在集群中的每个节点上删除 **/etc/logrotate.d/httpd** 文件中的以下行：

```
/bin/systemctl reload httpd.service > /dev/null 2>/dev/null || true
```

- 使用以下三行替换您删除的行，将 **/var/run/httpd-*website*.pid** 指定为 PID 文件路径，其中 *website* 是 Apache 资源的名称。在本例中，Apache 资源名称是 **Website**。

```
/usr/bin/test -f /var/run/httpd-Website.pid >/dev/null 2>/dev/null &&
/usr/bin/ps -q $(/usr/bin/cat /var/run/httpd-Website.pid) >/dev/null 2>/dev/null &&
/usr/sbin/httpd -f /etc/httpd/conf/httpd.conf -c "PidFile /var/run/httpd-Website.pid" -k
graceful > /dev/null 2>/dev/null || true
```

**验证步骤**

1. 从集群中的一个节点检查集群的状态。请注意，所有四个资源都运行在同一个节点上，**z1.example.com**。
如果发现配置的资源没有运行，则您可以运行 **pcs resource debug-start *resource*** 命令来测试资源配置。

```
[root@z1 ~]# pcs status
Cluster name: my_cluster
Last updated: Wed Jul 31 16:38:51 2013
Last change: Wed Jul 31 16:42:14 2013 via crm_attribute on z1.example.com
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.10-5.el7-9abe687
2 Nodes configured
6 Resources configured

Online: [ z1.example.com z2.example.com ]

Full list of resources:
 myapc  (stonith:fence_apc_snmp):       Started z1.example.com
 Resource Group: apachegroup
     my_lvm     (ocf::heartbeat:LVM-activate):   Started z1.example.com
     my_fs      (ocf::heartbeat:Filesystem):    Started z1.example.com
     VirtualIP  (ocf::heartbeat:IPaddr2):       Started z1.example.com
     Website    (ocf::heartbeat:apache):        Started z1.example.com
```

2. 集群启动并运行后，您可以将浏览器指向定义为 **IPaddr2** 资源的 IP 地址，来查看示例显示，包含简单的单词"Hello"。

```
Hello
```

3. 要测试运行在 **z1.example.com** 上的资源组是否可以切换到节点 **z2.example.com**，请将节点 **z1.example.com** 置于 **待机** 模式，之后该节点将不能再托管资源。

```
[root@z1 ~]# pcs node standby z1.example.com
```

4. 将节点 **z1** 置于 **待机** 模式后，从集群中的某个节点检查集群状态。请注意，资源现在都应运行在 **z2** 上。

```
[root@z1 ~]# pcs status
Cluster name: my_cluster
Last updated: Wed Jul 31 17:16:17 2013
Last change: Wed Jul 31 17:18:34 2013 via crm_attribute on z1.example.com
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.10-5.el7-9abe687
2 Nodes configured
6 Resources configured

Node z1.example.com (1): standby
Online: [ z2.example.com ]

Full list of resources:

 myapc  (stonith:fence_apc_snmp):       Started z1.example.com
 Resource Group: apachegroup
     my_lvm     (ocf::heartbeat:LVM-activate):   Started z2.example.com
     my_fs      (ocf::heartbeat:Filesystem):    Started z2.example.com
     VirtualIP  (ocf::heartbeat:IPaddr2):       Started z2.example.com
     Website    (ocf::heartbeat:apache):        Started z2.example.com
```

定义的 IP 地址的网页仍会显示，而不中断。

5. 要从 **待机** 模式中删除 **z1**，请输入以下命令。

```
[root@z1 ~]# pcs node unstandby z1.example.com
```

**注意**

从 **待机** 模式中删除节点本身不会导致资源切换到该节点。这将依赖于资源的 **resource-stickiness** 值。有关 **resource-stickiness** 元属性的详情，请参考 配置资源以首选其当前节点。

## 25.11. 其他资源

- 准备一个控制节点和一个受管节点以使用 RHEL 系统角色。
- 与 **rhel-system-roles** 软件包一起安装的文档在 **/usr/share/ansible/roles/rhel-system-roles.logging/README.html**
- RHEL 系统角色知识库文章
- **ansible-playbook(1)** 手册页。

# 第 26 章 使用 COCKPIT RHEL 系统角色安装和配置 WEB 控制台

使用 **cockpit** RHEL 系统角色，您可以在系统中安装和配置 Web 控制台。

## 26.1. COCKPIT 系统角色

您可以使用 **cockpit** 系统角色自动部署和启用 Web 控制台，从而从 Web 浏览器管理 RHEL 系统。

## 26.2. COCKPIT RHEL 系统角色的变量

用于 **cockpit** RHEL 系统角色的参数有：

| 角色变量 | 描述 |
| --- | --- |
| cockpit_packages: (默认为 default) | 设置其中一个预定义的软件包集：default、min 或 full。<br><br>* cockpit_packages:(默认为 default)- 最常见的页面和按需安装 UI<br><br>* cockpit_packages:(默认为 minimal)- 仅 Overview、Terminal、Logs、accounts 和 Metrics 页面；最小依赖项<br><br>* cockpit_packages:(默认为 full)- 所有可用的页面<br><br>（可选）指定您自己的 cockpit 软件包选择。 |
| cockpit_enabled: (默认值:true) | 配置是否启用了 web 控制台 web 服务器，以便在系统引导时自动启动 |
| cockpit_started: (默认值:true) | 配置 Web 控制台是否应启动 |
| cockpit_config:（默认为 nothing） | 您可以在 **/etc/cockpit/cockpit.conf** 文件中应用设置。注意：以前的设置文件将会丢失。 |
| cockpit_port: (默认值: 9090) | 默认情况下，Web 控制台在端口 9090 上运行。您可以使用这个选项更改端口。 |
| cockpit_manage_firewall: （默认值：false） | 允许 **cockpit** 角色控制 **firewall** 角色来添加端口。它不能用来删除端口。如果要删除端口，则需要直接使用 firewall 系统角色。 |
| cockpit_manage_selinux: (默认值: false) | 允许 **cockpit** 角色使用 **selinux** 角色配置 SELinux。默认 SELinux 策略不允许 Cockpit 侦听端口 9090 以外的任何端口。如果您更改了端口，请将这个选项设置为 **true**，以便 **selinux** 角色可以设置正确的端口权限(**websm_port_t**)。 |

| 角色变量 | 描述 |
| --- | --- |
| cockpit_certificates: （默认值：nothing） | 允许 **cockpit** 角色使用 **certificate** 角色生成新证书。**cockpit_certificates** 的值被传递给 **certificate** 角色的 **certificate_requests** 变量。此角色由 **cockpit** 角色在内部调用，它会生成私钥和证书。 |

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.cockpit/README.md** 文件。
- Cockpit 配置文件 man page。

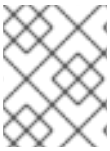
## 26.3. 使用 COCKPIT RHEL 系统角色安装 WEB 控制台

您可以使用 **cockpit** 系统角色安装并启用 RHEL web 控制台。

默认情况下，RHEL web 控制台使用自签名证书。为了安全起见，您可以指定由可信证书颁发机构发布的证书。

在本例中，您可以使用 **cockpit** 系统角色来：

- 安装 RHEL web 控制台。
- 允许 web 控制台管理 **firewalld**。
- 将 web 控制台设置为使用 **ipa** trusted 证书颁发机构的证书，而不使用自签名证书。
- 将 web 控制台设置为使用自定义端口 9050。



**注意**

您不必在 playbook 中调用 **firewall** 或 **certificate** 系统角色来管理防火墙或创建证书。**cockpit** 系统角色根据需要自动调用它们。

**先决条件**

- 访问一个或多个 *受管节点*。
- 对 *控制节点* 的访问和权限。
  在控制节点上：
  - Red Hat Ansible Engine 已安装。
  - **rhel-system-roles** 软件包已安装。
  - 存在一个列出受管节点的清单文件。

**流程**

1. 使用以下内容创建新的 ***playbook.yml*** 文件：

```
---
- hosts: all
  tasks:
    - name: Install RHEL web console
      include_role:
        name: rhel-system-roles.cockpit
      vars:
        cockpit_packages: default
        #cockpit_packages: minimal
        #cockpit_packages: full
        cockpit_port:9050
        cockpit_manage_selinux: true
        cockpit_manage_firewall: true
        cockpit_certificates:
          - name: /etc/cockpit/ws-certs.d/01-certificate
            dns: ['localhost', 'www.example.com']
            ca: ipa
            group: cockpit-ws
```

2. 可选：验证 playbook 语法：

```
# ansible-playbook --syntax-check -i inventory_file playbook.yml
```

3. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

**其他资源**

- 安装并启用 Web 控制台。
- 使用 RHEL 系统角色请求证书.

# 第 27 章 使用 PODMAN RHEL 系统角色管理容器

使用 **podman** RHEL 系统角色，您可以管理 Podman 配置、容器和运行 Podman 容器的 **systemd** 服务。

## 27.1. PODMAN RHEL 系统角色

您可以使用 **podman** RHEL 系统角色管理 Podman 配置、容器和运行 Podman 容器的 systemd 服务。

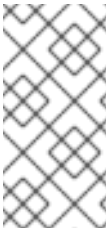
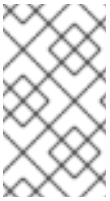
**其他资源**

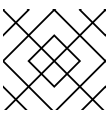
- 安装 RHEL 系统角色
- 有关 **podman** 中使用的参数以及 **podman** RHEL 系统角色的附加信息，请查看 **/usr/share/ansible/roles/rhel-system-roles.podman/README.md** 文件。

## 27.2. PODMAN RHEL 系统角色的变量

用于 **podman** RHEL 系统角色的参数有：

| 变量 | 描述 |
| --- | --- |
| **podman_kube_spec** | 描述要管理的 podman pod 和对应的 systemd 单元。<br>• **state** ：（默认：**created**）- 表示一个要通过 **systemd** 服务和 pod 执行的操作：<br>  ○ **created** ：创建 pod 和**systemd** 服务，但不运行它们<br>  ○ **started**: 创建 pod 和**systemd** 服务并启动它们<br>  ○ **absent** ：删除 pod 和**systemd** 服务<br>• **run_as_user** ：（默认：**podman_run_as_user**）- 每 pod 用户。如果没有指定，则使用 root。<br>**注意**<br>用户必须已经存在。<br>• **run_as_group** (默认: **podman_run_as_group**)- 每 pod 组。如果没有指定，则使用 root。<br>**注意**<br>组必须已经存在。<br>• **systemd_unit_scope** (默认: **podman_systemd_unit_scope**)- 用于 **systemd** 单元的范围。如果没有指定，**system** 用于 root 容器，**user** 用于用户容器。 |

| 变量 | 描述 |
| --- | --- |
| | • **kube_file_src** - 控制器节点上 Kubernetes YAML 文件的名称，该文件将被复制到受管节点上的 **kube_file**<br><br>  <br>**注意**<br>如果指定了 **kube_file_content** 变量，则不要指定 **kube_file_src** 变量。**kube_file_content** 优先于 **kube_file_src**。<br><br>• **kube_file_content** - Kubernetes YAML 格式的字符串，或 Kubernetes YAML 格式的字典。它指定受管节点上 **kube_file** 的内容。<br><br>  <br>**注意**<br>如果指定了 **kube_file_src** 变量，则不要指定 **kube_file_content** 变量。**kube_file_content** 优先于 **kube_file_src**。<br><br>• **kube_file** - 受管节点上包含容器或 pod 的 Kubernetes 规范的文件的名称。除非需要将 **kube_file** 文件复制到角色外的受管节点，否则通常不必指定 **kube_file** 变量。如果指定了 **kube_file_src** 或 **kube_file_content**，则不必指定此变量。<br><br>  <br>**注意**<br>强烈建议省略 **kube_file**，而指定 **kube_file_src** 或 **kube_file_content**，让角色管理文件路径和名称。<br><br>◦ 文件 basename 是 K8s yaml 中的 metadata.name 值，并附加了 **.yml** 后缀。<br><br>◦ 对于系统服务，目录为 **/etc/containers/ansible-kubernetes.d**。<br><br>◦ 对于用户服务，目录为 **$HOME/.config/containers/ansible-kubernetes.d**。<br><br>◦ 这将被复制到受管节点上的 **/etc/containers/ansible-kubernetes.d/*<application_name>*.yml** 文件。 |

| 变量 | 描述 |
| --- | --- |
| **podman_create_host_directories** | 如果为 true，该角色确保在 **podman_kube_specs** 中给定的 Kubernetes YAML 中的 **volumes.hostPath** 规范中的主机挂载中指定的主机目录。默认值为 false。<br><br>  <br>**注意**<br>目录必须指定为绝对路径（用于 root 容器），或者相对于主目录的路径（对于非 root 容器），以便角色管理它们。其他任何内容都被忽略。<br><br>该角色将其默认所有权或权限应用到目录。如果需要设置所有权或权限，请参阅 **podman_host_directory**。 |
| **podman_host_directories** | 它是字典。如果使用 **podman_create_host_directories** 告知角色为卷挂载创建主机目录，并且您需要指定应用到这些创建的主机目录的权限或所有权，请使用 **podman_host_directory**。每个键都是要管理的主机目录的绝对路径。该值采用 file 模块的参数格式。如果没有指定值，该角色将使用其内置的默认值。如果要指定一个用于所有主机目录的值，请使用特殊键 **DEFAULT**。 |
| **podman_firewall** | 它是一个字典列表。指定您希望角色在防火墙中管理的端口。这使用的格式与 firewall RHEL 系统角色使用的格式相同。 |
| **podman_selinux_ports** | 它是一个字典列表。指定您希望角色为角色使用的端口管理 SELinux 策略的端口。这使用的格式与 selinux RHEL 系统角色使用的格式相同。 |
| **podman_run_as_user** | 指定用于所有无根容器的用户名称。您还可以使用 **podman_kube_specs** 中的 **run_as_user** 指定每容器用户名。<br><br>  <br>**注意**<br>用户必须已经存在。 |
| **podman_run_as_group** | 指定用于所有无根容器的组名称。您还可以在 **podman_kube_specs** 中使用 **run_as_group** 指定每容器组名称。<br><br>  <br>**注意**<br>组必须已经存在。 |

| 变量 | 描述 |
| --- | --- |
| **podman_systemd_unit_scope** | 定义所有 **systemd** 单元默认使用的 **systemd** 范围。您还可以在 **podman_kube_specs** 中使用 **systemd_unit_scope** 指定每容器范围。默认情况下，无根容器使用 **user**，root 容器使用 **system**。 |
| **podman_containers_conf** | 将 **containers.conf (5)** 设置定义为字典。此设置在 **containers.conf.d** 目录中的置入文件中提供。如果以 root 身份运行（请参阅 **podman_run_as_user**），将管理 **system** 设置。否则，将管理 **user** 设置。有关目录位置，请参阅 **containers.conf** 手册页。 |
| **podman_registries_conf** | 将 **containers-registries.conf (5)** 设置定义为字典。设置在 **registry.conf.d** 目录中的置入文件中提供。如果以 root 身份运行（请参阅 **podman_run_as_user**），将管理 **system** 设置。否则，将管理 **user** 设置。有关目录位置，请参见 **registries.conf** 手册页。 |
| **podman_storage_conf** | 将 **containers-storage.conf (5)** 设置定义为字典。如果以 root 身份运行（请参阅 **podman_run_as_user**），将管理 **system** 设置。否则，将管理 **user** 设置。有关目录位置，请参见 **storage.conf** 手册页。 |
| **podman_policy_json** | 将 **containers-policy.conf (5)** 设置定义为字典。如果以 root 身份运行（请参阅 **podman_run_as_user**），将管理 **system** 设置。否则，将管理 **user** 设置。有关目录位置，请参阅 **policy.json** 手册页。 |

**其他资源**

- 安装 RHEL 系统角色
- 有关 **podman** 中使用的参数以及 **podman** RHEL 系统角色的附加信息，请查看 **/usr/share/ansible/roles/rhel-system-roles.podman/README.md** 文件。

## 27.3. 其他资源

- 有关 **podman** 中使用的参数以及 **podman** RHEL 系统角色的附加信息，请查看 **/usr/share/ansible/roles/rhel-system-roles.podman/README.md** 文件。
- 有关 **ansible-playbook** 命令的详情，请查看 **ansible-playbook(1)** 手册页。

# 第 28 章 使用 RHEL 系统角色将 RHEL 系统直接与 AD 集成

使用 **ad_integration** 系统角色，您可以使用 Red Hat Ansible Automation Platform 自动将 RHEL 系统与活动目录(AD)集成。

本章涵盖了以下主题：

- **ad_integration** 系统角色
- ad_integration RHEL 系统角色的变量
- 使用 **ad_integration** 系统角色将 RHEL 系统直接连接到 AD

## 28.1. AD_INTEGRATION 系统角色

使用 **ad_integration** 系统角色，您可以将 RHEL 系统直接连接到 活动目录(AD)。

该角色使用以下组件：

- SSSD 与中央身份和身份验证源交互
- **realmd** 来检测可用的 AD 域，并配置底层 RHEL 系统服务（在本例中为 SSSD）来连接到所选 AD 域

**注意**

**ad_integration** 角色用于使用没有身份管理(IdM)环境的直接 AD 集成的部署。对于 IdM 环境，请使用 **ansible-freeipa** 角色。

**其他资源**

- 使用 SSSD 将 RHEL 系统直接连接到 AD 。

## 28.2. AD_INTEGRATION RHEL 系统角色的变量

**ad_integration** RHEL 系统角色使用以下参数：

| 角色变量 | 描述 |
| --- | --- |
| ad_integration_realm | 活动目录领域或要加入的域名。 |
| ad_integration_password | 在将机器加入到域时用来进行身份验证的用户密码。不要使用纯文本。反之，使用 Ansible Vault 来加密值。 |
| ad_integration_manage_crypto_policies | 如果为 **true**，**ad_integration** 角色将根据需要使用 **fedora.linux_system_roles.crypto_policies**。<br><br>默认：**false** |

| 角色变量 | 描述 |
| --- | --- |
| ad_integration_allow_rc4_crypto | 如果为 **true**，**ad_integration** 角色将设置加密策略以允许 RC4 加密。<br><br>提供此变量会自动将 **ad_integration_manage_crypto_policies** 设置为 **true**。<br><br>默认：**false** |
| ad_integration_timesync_source | 与系统时钟同步的时间源的主机名或 IP 地址。提供此变量会自动将 **ad_integration_manage_timesync** 设置为 **true**。 |

**其他资源**

- **/usr/share/ansible/roles/rhel-system-roles.ad_integration/README.md** 文件。

## 28.3. 使用 AD_INTEGRATION 系统角色将 RHEL 系统直接连接到 AD

您可以使用 **ad_integration** 系统角色，通过运行 Ansible playbook 配置 RHEL 系统和 AD 域之间的直接集成。

**注意**

从 RHEL8 开始，RHEL 默认不再支持 RC4 加密。如果无法在 AD 域中启用 AES，您必须启用 **AD-SUPPORT** 加密策略，并在 playbook 中允许 RC4 加密。

**重要**

RHEL 服务器和 AD 之间的时间必须同步。您可以通过在 playbook 中使用 **timesync** 系统角色来确保这一点。

在本例中，RHEL 系统使用 AD **Administrator** 用户和存储在 Ansible vault 中的此用户的密码加入 **domain.example.com** AD 域。playbook 还设置 **AD-SUPPORT** 加密策略，并允许 RC4 加密。为确保 RHEL 系统和 AD 之间的时间同步，playbook 会将 **adserver.domain.example.com** 服务器设置为 **timesync** 源。

**先决条件**

- 访问一个或多个 *受管节点*。
- 对 *控制节点* 的访问和权限。
  在控制节点上：
  - Red Hat Ansible Engine 已安装。
  - **rhel-system-roles** 软件包已安装。
  - 列出受管节点的清单文件。

- AD 域控制器上的以下端口已开放，并可从 RHEL 服务器访问：

**表 28.1. 使用 ad_integration 系统角色将 Linux 系统直接集成到 AD 所需的端口**

| 源端口 | 目的地端口 | 协议 | 服务 |
|---|---|---|---|
| 1024:65535 | 53 | UDP 和 TCP | DNS |
| 1024:65535 | 389 | UDP 和 TCP | LDAP |
| 1024:65535 | 636 | TCP | LDAPS |
| 1024:65535 | 88 | UDP 和 TCP | Kerberos |
| 1024:65535 | 464 | UDP 和 TCP | Kerberos 更改/设置密码(**kadmin**) |
| 1024:65535 | 3268 | TCP | LDAP 全局目录 |
| 1024:65535 | 3269 | TCP | LDAP 全局目录 SSL/TLS |
| 1024:65535 | 123 | UDP | NTP/Chrony （可选） |
| 1024:65535 | 323 | UDP | NTP/Chrony （可选） |

**流程**

1. 使用以下内容创建一个新的 **ad_integration.yml** 文件：

```
---
- hosts: all
  vars:
    ad_integration_realm: "domain.example.com"
    ad_integration_password: !vault | vault encrypted password
    ad_integration_manage_crypto_policies: true
    ad_integration_allow_rc4_crypto: true
    ad_integration_timesync_source: "adserver.domain.example.com"
  roles:
    - linux-system-roles.ad_integration
---
```

2. 可选：验证 playbook 语法。

```
# ansible-playbook --syntax-check ad_integration.yml -i inventory_file
```

3. 在清单文件上运行 playbook:

```
# ansible-playbook -i inventory_file /path/to/file/ad_integration.yml
```

**验证**

**验证**

- 显示 AD 用户详情，如 **administrator** 用户：

```
getent passwd administrator@ad.example.com
administrator@ad.example.com:*:1450400500:1450400513:Administrator:/home/administrator
@ad.example.com:/bin/bash
```

## 28.4. 其他资源

- **/usr/share/ansible/roles/rhel-system-roles.ad_integration/README.md** 文件。
- **man ansible-playbook (1)**