



Red Hat Enterprise Linux 9

使用 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

Red Hat Enterprise Linux 9 使用 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

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

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 系统的一致且可重复的管理。使用 RHEL 系统角色，您可以通过从一个系统运行配置 playbook 来高效地管理大型系统清单。

目录

对红帽文档提供反馈	6
第 1 章 RHEL 系统角色简介	7
第 2 章 准备一个控制节点和受管节点以使用 RHEL 系统角色	9
2.1. 在 RHEL 9 上准备一个控制节点	9
2.2. 准备受管节点	11
第 3 章 ANSIBLE VAULT	14
第 4 章 RHEL 中的 ANSIBLE IPMI 模块	16
4.1. RHEL_MGMT 集合	16
4.2. 使用 IPMI_BOOT 模块	17
4.3. 使用 IPMI_POWER 模块	18
第 5 章 RHEL 中的 REDFISH 模块	20
5.1. REDFISH 模块	20
5.2. REDFISH 模块参数	20
5.3. 使用 REDFISH_INFO 模块	21
5.4. 使用 REDFISH_COMMAND 模块	22
5.5. 使用 REDFISH_CONFIG 模块	23
第 6 章 使用 RHEL 系统角色将 RHEL 系统直接集成到 AD	25
6.1. AD_INTEGRATION RHEL 系统角色	25
第 7 章 使用 RHEL 系统角色请求证书	26
7.1. CERTIFICATE RHEL 系统角色	26
7.2. 使用 CERTIFICATE RHEL 系统角色请求一个新的自签名证书	26
7.3. 使用 CERTIFICATE RHEL 系统角色从 IDM CA 请求一个新证书	27
7.4. 使用 CERTIFICATE RHEL 系统角色指定在证书颁发之前或之后要运行的命令	28
第 8 章 使用 RHEL 系统角色安装和配置 WEB 控制台	30
8.1. COCKPIT RHEL 系统角色	30
8.2. 使用 COCKPIT RHEL 系统角色安装 WEB 控制台	30
第 9 章 使用 RHEL 系统角色设置自定义加密策略	32
9.1. 使用 CRYPTO_POLICIES RHEL 系统角色设置自定义加密策略	32
第 10 章 使用 RHEL 系统角色配置 FIREWALLD	34
10.1. FIREWALL RHEL 系统角色简介	34
10.2. 使用 FIREWALL RHEL 系统角色重置 FIREWALLD 设置	34
10.3. 使用 FIREWALL RHEL 系统角色，将 FIREWALLD 中的传入流量从一个本地端口转发到不同的本地端口	35
10.4. 使用 FIREWALL RHEL 系统角色管理 FIREWALLD 中的端口	36
10.5. 使用 FIREWALL RHEL 系统角色配置 FIREWALLD DMZ 区域	37
第 11 章 使用 RHEL 系统角色配置高可用性集群	40
11.1. HA_CLUSTER RHEL 系统角色的变量	40
11.2. 为 HA_CLUSTER RHEL 系统角色指定一个清单	57
11.3. 为高可用性集群创建 PCSD TLS 证书和密钥文件	59
11.4. 配置不运行任何资源的高可用性集群	60
11.5. 配置带有隔离和资源的高可用性集群	61
11.6. 配置具有资源和资源操作默认值的高可用性集群	64
11.7. 配置具有隔离级别的高可用性集群	66
11.8. 使用资源限制配置高可用性集群	68

11.9. 在高可用性集群中配置 COROSYNC 值	71
11.10. 使用 SBD 节点隔离配置高可用性集群	73
11.11. 使用仲裁设备配置高可用性集群	74
11.12. 配置具有节点属性的高可用性集群	77
11.13. 使用 HA_CLUSTER RHEL 系统角色在高可用性集群中配置一个 APACHE HTTP 服务器	78
第 12 章 使用 RHEL 系统角色配置 SYSTEMD 日志	83
12.1. 使用 JOURNALD RHEL 系统角色配置持久性日志记录	83
第 13 章 使用 RHEL 系统角色配置自动崩溃转储	85
13.1. 使用 KDUMP RHEL 系统角色配置内核崩溃转储机制	85
第 14 章 使用 RHEL 系统角色永久配置内核参数	87
14.1. KERNEL_SETTINGS RHEL 系统角色简介	87
14.2. 使用 KERNEL_SETTINGS RHEL 系统角色应用所选的内核参数	87
第 15 章 使用 RHEL 系统角色配置日志记录	90
15.1. LOGGING RHEL 系统角色	90
15.2. 应用一个本地 LOGGING RHEL 系统角色	90
15.3. 在本地 LOGGING RHEL 系统角色中过滤日志	92
15.4. 使用 LOGGING RHEL 系统角色应用一个远程日志解决方案	93
15.5. 使用带有 TLS 的 LOGGING RHEL 系统角色	96
15.6. 将 RELP 与 LOGGING RHEL 系统角色一起使用	100
第 16 章 使用 RHEL 系统角色监控性能	106
16.1. METRICS RHEL 系统角色简介	106
16.2. 使用 METRICS RHEL 系统角色以可视化方式监控本地系统	106
16.3. 使用 METRICS RHEL 系统角色将单独的系统设置成一组来监控其自身	107
16.4. 使用 METRICS RHEL 系统角色，使用本地机器集中监控一组机器	108
16.5. 使用 METRICS RHEL 系统角色，在监控系统时设置身份验证	109
16.6. 使用 METRICS RHEL 系统角色为 SQL SERVER 配置并启用指标集合	110
第 17 章 使用 RHEL 系统角色配置 MICROSOFT SQL SERVER	112
17.1. 使用带有现有证书文件的 MICROSOFT.SQL.SERVER 系统角色安装和配置 SQL 服务器	112
17.2. 使用 MICROSOFT.SQL.SERVER 系统角色和 CERTIFICATE RHEL 系统角色安装和配置 SQL 服务器	113
17.3. 为数据和日志设置自定义存储路径	114
第 18 章 使用 RHEL 系统角色配置 NBDE	116
18.1. NBDE_CLIENT 和 NBDE_SERVER RHEL 系统角色简介(CLEVIS 和 TANG)	116
18.2. 使用 NBDE_SERVER RHEL 系统角色设置多个 TANG 服务器	116
18.3. 使用 NBDE_CLIENT RHEL 系统角色设置多个 CLEVIS 客户端	117
第 19 章 使用 RHEL 系统角色配置网络设置	120
19.1. 使用 NETWORK RHEL 系统角色和接口名称，配置具有静态 IP 地址的以太网连接	120
19.2. 使用 NETWORK RHEL 系统角色和设备路径，配置具有静态 IP 地址的以太网连接	121
19.3. 使用 NETWORK RHEL 系统角色和接口名称，配置具有动态 IP 地址的以太网连接	123
19.4. 使用 NETWORK RHEL 系统角色和设备路径，配置具有动态 IP 地址的以太网连接	124
19.5. 使用 NETWORK RHEL 系统角色配置 VLAN 标记	125
19.6. 使用 NETWORK RHEL 系统角色配置网桥	127
19.7. 使用 NETWORK RHEL 系统角色配置网络绑定	129
19.8. 使用 NETWORK RHEL 系统角色配置 IPOIB 连接	130
19.9. 使用 NETWORK RHEL 系统角色将特定子网的流量路由到不同的默认网关	132
19.10. 使用 NETWORK RHEL 系统角色配置一个具有 802.1X 网络身份验证的静态以太网连接	136
19.11. 使用 NETWORK RHEL 系统角色配置一个具有 802.1X 网络身份验证的 WIFI 连接	138
19.12. 使用 NETWORK RHEL 系统角色在现有连接上设置默认网关	140

19.13. 使用 NETWORK RHEL 系统角色配置一个静态路由	142
19.14. 使用 NETWORK RHEL 系统角色配置一个 ETHTOOL 卸载功能	144
19.15. 使用 NETWORK RHEL 系统角色配置 ETHTOOL 合并设置	145
19.16. 使用 NETWORK RHEL 系统角色增加环缓冲区的大小，以减少高数据包丢弃率	147
19.17. NETWORK RHEL 系统角色的网络状态	149
第 20 章 使用 PODMAN RHEL 系统角色管理容器	151
20.1. 创建一个带有绑定挂载的无根容器	151
20.2. 使用 PODMAN 卷创建有根容器	152
20.3. 使用 SECRET 创建一个 QUADLET 应用程序	154
第 21 章 使用 RHEL 系统角色配置 POSTFIX MTA	157
21.1. 使用 POSTFIX RHEL 系统角色自动化基本 POSTFIX MTA 管理	157
第 22 章 使用 RHEL 系统角色安装和配置 POSTGRESQL	159
22.1. POSTGRESQL RHEL 系统角色简介	159
22.2. 使用 POSTGRESQL RHEL 系统角色配置 POSTGRESQL 服务器	159
第 23 章 使用 RHEL 系统角色注册系统	161
23.1. RHC RHEL 系统角色简介	161
23.2. 使用 RHC RHEL 系统角色注册系统	161
23.3. 使用 RHC RHEL 系统角色，使用 SATELLITE 注册系统	162
23.4. 使用 RHC RHEL 系统角色在注册后禁用到 INSIGHTS 的连接	164
23.5. 使用 RHC RHEL 系统角色启用存储库	165
23.6. 使用 RHC RHEL 系统角色设置发行版本	166
23.7. 在使用 RHC RHEL 系统角色注册主机时使用代理服务器	166
23.8. 使用 RHC RHEL 系统角色禁用 INSIGHTS 规则的自动更新	168
23.9. 使用 RHC RHEL 系统角色禁用 INSIGHTS 补救	169
23.10. 使用 RHC RHEL 系统角色配置 INSIGHTS 标签	170
23.11. 使用 RHC RHEL 系统角色取消系统注册	172
第 24 章 使用 RHEL 系统角色配置 SELINUX	173
24.1. SELINUX RHEL 系统角色简介	173
24.2. 使用 SELINUX RHEL 系统角色在多个系统上应用 SELINUX 设置	173
24.3. 使用 SELINUX RHEL 系统角色管理端口	174
第 25 章 使用 RHEL 系统角色保护文件访问	176
25.1. 使用 FAPOLICYD RHEL 系统角色配置对未知代码执行的保护	176
第 26 章 使用 RHEL 系统角色配置安全通信	178
26.1. SSHD RHEL 系统角色的变量	178
26.2. 使用 SSHD RHEL 系统角色配置 OPENSSH 服务器	178
26.3. 对非独占配置使用 SSHD RHEL 系统角色	180
26.4. 使用 SSHD RHEL 系统角色覆盖 SSH 服务器上系统范围的加密策略	182
26.5. SSH RHEL 系统角色的变量	183
26.6. 使用 SSH RHEL 系统角色配置 OPENSSH 客户端	184
第 27 章 使用 RHEL 系统角色管理本地存储	186
27.1. STORAGE RHEL 系统角色简介	186
27.2. 使用 STORAGE RHEL 系统角色在块设备上创建 XFS 文件系统	186
27.3. 使用 STORAGE RHEL 系统角色永久挂载文件系统	188
27.4. 使用 STORAGE RHEL 系统角色管理逻辑卷	189
27.5. 使用 STORAGE RHEL 系统角色启用在线块丢弃	190
27.6. 使用 STORAGE RHEL 系统角色创建并挂载 EXT4 文件系统	190
27.7. 使用 STORAGE RHEL 系统角色创建并挂载 EXT3 文件系统	191

27.8. 使用 STORAGE RHEL 系统角色调整 LVM 上现有文件系统的大小	192
27.9. 使用 STORAGE RHEL 系统角色创建交换卷	193
27.10. 使用 STORAGE RHEL 系统角色配置 RAID 卷	194
27.11. 使用 STORAGE RHEL 系统角色配置带有 RAID 的 LVM 池	196
27.12. 使用 STORAGE RHEL 系统角色为 RAID LVM 卷配置条带大小	197
27.13. 使用 STORAGE RHEL 系统角色压缩和去重 LVM 上的 VDO 卷	198
27.14. 使用 STORAGE RHEL 系统角色创建 LUKS2 加密的卷	199
27.15. 使用 STORAGE RHEL 系统角色将池卷大小表示为百分比	200
第 28 章 使用 RHEL 系统角色管理 SYSTEMD 单元	202
28.1. 使用 SYSTEMD RHEL 系统角色部署并启动 SYSTEMD 单元	202
第 29 章 使用 RHEL 系统角色配置时间同步	204
29.1. TIMESYNC RHEL 系统角色	204
29.2. 为单个服务器池应用 TIMESYNC RHEL 系统角色	204
29.3. 在客户端服务器上应用 TIMESYNC RHEL 系统角色	205
第 30 章 使用 RHEL 系统角色为会话记录配置一个系统	207
30.1. TLOG RHEL 系统角色	207
30.2. TLOG RHEL 系统角色的组件和参数	207
30.3. 部署 TLOG RHEL 系统角色	207
30.4. 部署 TLOG RHEL 系统角色，以排除组或用户的列表	208
第 31 章 使用 RHEL 系统角色配置带有 IPSEC 的 VPN 连接	211
31.1. 使用 VPN RHEL 系统角色创建一个带有 IPSEC 的主机到主机的 VPN	211
31.2. 使用 VPN RHEL 系统角色创建一个带有 IPSEC 的机会网状 VPN 连接	213

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. 在顶部导航栏中点 **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。

重要术语和概念

下面描述了 Ansible 环境中的重要术语和概念：

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 9 上准备一个控制节点](#)。

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 [准备一个受管节点](#)。

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，您可以列出受管节点并指定每个受管节点的信息，如 IP 地址。在清单中，您还可以通过创建和嵌套组来组织受管节点，以便于扩展。清单文件有时也称为 hostfile。

Red Hat Enterprise Linux 9 控制节点上可用的角色

在 Red Hat Enterprise Linux 9 控制节点上，**rhel-system-roles** 软件包提供以下角色：

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
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

角色名称	角色描述	章节标题
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
postgresql	PostgreSQL	使用 postgresql RHEL 系统角色安装和配置 PostgreSQL
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	Storage	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 系统角色配置带有 IPsec 的 VPN 连接

其他资源

- [Red Hat Enterprise Linux \(RHEL\)系统角色](#)
- `/usr/share/ansible/roles/rhel-system-roles.<role_name>/README.md` 文件
- `/usr/share/doc/rhel-system-roles/<role_name>/` 目录

第 2 章 准备一个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前，您必须准备控制节点和受管节点。

2.1. 在 RHEL 9 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，此系统根据 playbook 从清单中配置受管主机。

先决条件

- 该系统已注册到客户门户网站。
- **Red Hat Enterprise Linux Server** 订阅被附加到系统上。
- 可选：**Ansible Automation Platform** 订阅被附加到系统上。

步骤

1. 创建一个名为 **ansible** 的用户，来管理并运行 playbook：

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

2. 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。

3. 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):
Enter passphrase (empty for no passphrase): <password>
Enter same passphrase again: <password>
...
```

为密钥文件使用推荐的默认位置。

4. 可选：要防止 Ansible 在每次建立连接时提示您输入 SSH 密钥密码，请配置一个 SSH 代理。
5. 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，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，建议这样做。
6. 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。

7. 安装 RHEL 系统角色：

- 在没有 Ansible Automation Platform 的 RHEL 主机上，安装 **rhel-system-roles** 软件包：

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

此命令在 **/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/** 目录中安装集合，并且 **ansible-core** 软件包作为依赖项。

- 在 Ansible Automation Platform 上，以 **ansible** 用户身份执行以下步骤：
 - i. 对 **~/ansible.cfg** 文件中的内容 将 [Red Hat Automation hub](#) 定义为主要源。
 - ii. 从 Red Hat Automation Hub 安装 **redhat.rhel_system_roles** 集合：

```
[ansible@control-node]$ ansible-galaxy collection install
redhat.rhel_system_roles
```

此命令在 **~/ansible/collections/ansible_collections/redhat/rhel_system_roles/** 目录中安装集合。

后续步骤

- 准备受管节点。如需更多信息，请参阅 [准备一个受管节点](#)。

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](#)

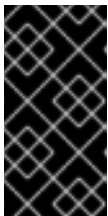
- [如何使用 subscription-manager 在红帽客户门户网站中注册和订阅系统](#)
- [ssh-keygen \(1\) 手册页](#)
- [通过 ssh-agent，使用 SSH 密钥连接到远程机器](#)
- [Ansible 配置设置](#)
- [如何构建清单](#)

2.2. 准备受管节点

受管节点是在清单中列出的系统，它由控制节点根据 playbook 进行配置。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 9 上准备一个控制节点](#)。
- 您从控制节点进行 SSH 访问的权限。



重要

以 **root** 用户身份进行直接的 SSH 访问是一个安全风险。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到受管节点，并以不同的用户身份（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** 用户身份登录到控制节点，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. 当提示时，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. 当提示时，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

```

与普通编辑器相比，使用 **visudo** 的好处是，该工具在安装文件前提供基本的检查，如，检查解析错误。

- 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. 验证您可以在所有受管节点上执行来自控制节点命令：


```
[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可以正常工作：

```
[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**，则您在受管节点上正确地配置了 **sudo**。

其他资源

- [在 RHEL 9 上准备一个控制节点](#)
- **sudoers (5)** 手册页

第 3 章 ANSIBLE VAULT

有时，您的 playbook 需要使用敏感数据，如密码、API 密钥和其他 secret 来配置受管主机。将此信息以纯文本形式存储在变量或其他与 Ansible 兼容的文件中存在安全风险，因为有权访问这些文件的任何用户都可以读取敏感数据。

使用 Ansible vault，您可以加密、解密、查看和编辑敏感信息。它们可能包括：

- 在 Ansible Playbook 中插入的变量文件
- 主机和组变量
- 执行 playbook 时作为参数传递的变量文件
- Ansible 角色中定义的变量

您可以使用 Ansible vault 安全地管理单个变量、整个文件，甚至像 YAML 文件这样的结构化的数据。然后，此数据可以被安全地存储在版本控制系统中，或者与团队成员共享，而不会暴露敏感信息。



重要

文件通过高级加密标准(AES256)的对称加密进行了保护，其中单个密码或密码短语用于加密和解密数据。请注意，这一操作方式尚未经过第三方的正式审核。

为了简化管理，设置您的 Ansible 项目，以便敏感变量和所有其他变量都保存在单独的文件或目录中是有意义的。然后，您可以使用 **ansible-vault** 命令保护包含敏感变量的文件。

创建加密的文件

以下命令提示您输入新的 vault 密码。然后，它使用默认的编辑器打开一个文件来存储敏感变量。

```
# ansible-vault create vault.yml
New Vault password: <vault_password>
Confirm New Vault password: <vault_password>
```

查看加密的文件

以下命令提示您输入现有的 vault 密码。然后，它显示已加密的文件的敏感内容。

```
# ansible-vault view vault.yml
Vault password: <vault_password>
my_secret: "yJJvPqhsiusmmPPZdnjndkdnYNDjdj782meUZcw"
```

编辑加密的文件

以下命令提示您输入现有的 vault 密码。然后，它打开已加密的文件，以供您使用默认编辑器更新敏感变量。

```
# ansible-vault edit vault.yml
Vault password: <vault_password>
```

加密现有文件

以下命令提示您输入新的 vault 密码。然后，它加密现有的未加密的文件。

```
# ansible-vault encrypt vault.yml
New Vault password: <vault_password>
Confirm New Vault password: <vault_password>
Encryption successful
```

解密现有文件

以下命令提示您输入现有的 vault 密码。然后，它会解密现有的加密文件。

```
# ansible-vault decrypt vault.yml
Vault password: <vault_password>
Decryption successful
```

更改加密的文件的密码

以下命令提示您输入原始 vault 密码，然后提示输入新的 vault 密码。

```
# ansible-vault rekey vault.yml
Vault password: <vault_password>
New Vault password: <vault_password>
Confirm New Vault password: <vault_password>
Rekey successful
```

playbook 中 Ansible vault 变量的基本应用程序

```
---
- name: Create user accounts for all servers
  hosts: managed-node-01.example.com
  vars_files:
    - vault.yml
  tasks:
    - name: Create user from vault.yml file
      user:
        name: "{{ username }}"
        password: "{{ pwhash }}"
```

您使用 Ansible Playbook 的 **vars_files** 部分中的变量(vault.yml)读取文件，并且就像使用普通变量一样使用大括号。然后，您可以使用 **ansible-playbook --ask-vault-pass** 命令运行 playbook，并手动输入密码。或者，您可以将密码保存到单独的文件中，并使用 **ansible-playbook --vault-password-file /path/to/my/vault-password-file** 命令运行 playbook。

其他资源

- [ansible-vault \(1\), ansible-playbook \(1\) 手册页](#)
- [Ansible vault](#)
- [Ansible vault 最佳实践](#)

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 8 AppStream，作为新 **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 * 网络 * 软盘 * 硬盘 * 安全 * 光盘 * 设置 * 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

模块名称	描述
password	连接到 BMC 的密码
user	连接到 BMC 的用户名
状态	<p>检查机器是否处于所需的状态</p> <ul style="list-style-type: none"> * 开 * 关 * 关闭 * 重置 * 启动

4.2. 使用 IPMI_BOOT 模块

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

- 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。
- **ansible-collection-redhat-rhel_mgmt** 软件包已安装。
- **python3-pyghmi** 软件包已安装在控制节点或受管节点上。
- 您要控制的 IPMI BMC 可以从控制节点或受管主机（如果不使用 **localhost** 作为受管主机）通过网络访问。请注意，其 BMC 由模块配置的主机通常与受管主机不同，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。

步骤

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

```
---
- name: Set boot device to be used on next boot
  hosts: managed-node-01.example.com
  tasks:
    - name: Ensure boot device is HD
      redhat.rhel_mgmt.ipmi_boot:
        user: <admin_user>
        password: <password>
        bootdev: hd
```

- 2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

- 3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 运行 playbook 时，Ansible 返回 **success**。

其他资源

- `/usr/share/ansible/collections/ansible_collections/redhat/rhel_mgmt/README.md` 文件

4.3. 使用 IPMI_POWER 模块

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

- 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。
- **ansible-collection-redhat-rhel_mgmt** 软件包已安装。
- **python3-pyghmi** 软件包已安装在控制节点或受管节点上。
- 您要控制的 IPMI BMC 可以从控制节点或受管主机（如果不使用 **localhost** 作为受管主机）通过网络访问。请注意，其 BMC 由模块配置的主机通常与受管主机不同，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。

步骤

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Power management
  hosts: managed-node-01.example.com
  tasks:
    - name: Ensure machine is powered on
      redhat.rhel_mgmt.ipmi_power:
```

```
user: <admin_user>
password: <password>
state: on
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 运行 playbook 时，Ansible 返回 **true**。

其他资源

- `/usr/share/ansible/collections/ansible_collections/redhat/rhel_mgmt/README.md` 文件

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

- 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。
- **ansible-collection-redhat-rhel_mgmt** 软件包已安装。
- **python3-pyghmi** 软件包已安装在控制节点或受管节点上。
- OOB 控制器访问详细信息。

步骤

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

```
---
- name: Manage out-of-band controllers using Redfish APIs
  hosts: managed-node-01.example.com
  tasks:
    - name: Get CPU inventory
```

```
redhat.rhel_mgmt.redfish_info:
  baseuri: "<URI>"
  username: "<username>"
  password: "<password>"
  category: Systems
  command: GetCpuInventory
  register: result
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 运行 playbook 时，Ansible 返回 CPU 清单详细信息。

其他资源

- `/usr/share/ansible/collections/ansible_collections/redhat/rhel_mgmt/README.md` 文件

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

- 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。
- ansible-collection-redhat-rhel_mgmt** 软件包已安装。
- python3-pyghmi** 软件包已安装在控制节点或受管节点上。
- OOB 控制器访问详细信息。

步骤

- 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Manage out-of-band controllers using Redfish APIs
  hosts: managed-node-01.example.com
  tasks:
    - name: Power on system
```

```
redhat.rhel_mgmt.redfish_command:
  baseuri: "<URI>"
  username: "<username>"
  password: "<password>"
  category: Systems
  command: PowerOn
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 系统开机。

其他资源

- `/usr/share/ansible/collections/ansible_collections/redhat/rhel_mgmt/README.md` 文件

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

- 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。
- **ansible-collection-redhat-rhel_mgmt** 软件包已安装。
- **python3-pyghmi** 软件包已安装在控制节点或受管节点上。
- OOB 控制器访问详细信息。

步骤

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Manages out-of-band controllers using Redfish APIs
  hosts: managed-node-01.example.com
  tasks:
    - name: Set BootMode to UEFI
      redhat.rhel_mgmt.redfish_config:
```

```
baseuri: "<URI>"
username: "<username>"
password: "<password>"
category: Systems
command: SetBiosAttributes
bios_attributes:
  BootMode: Uefi
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 系统引导模式被设置为 UEFI。

其他资源

- `/usr/share/ansible/collections/ansible_collections/redhat/rhel_mgmt/README.md` 文件

第 6 章 使用 RHEL 系统角色将 RHEL 系统直接集成到 AD

使用 **ad_integration** 系统角色，您可以使用 Red Hat Ansible Automation Platform 自动将 RHEL 系统直接与活动目录(AD)集成。

6.1. AD_INTEGRATION RHEL 系统角色

使用 **ad_integration** 系统角色，您可以直接将 RHEL 系统连接到活动目录(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

- `/usr/share/ansible/roles/rhel-system-roles.ad_integration/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ad_integration/` 目录
- [使用 SSSD 将 RHEL 系统直接连接到 AD](#)

第 7 章 使用 RHEL 系统角色请求证书

您可以使用 **certificate** 系统角色发布和管理证书。

7.1. CERTIFICATE RHEL 系统角色

使用 **certificate** 系统角色，您可以使用 Ansible Core 管理发布和更新 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)。

您可以使用 **certificate** 系统角色，在 Ansible playbook 中使用以下变量：

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

- `/usr/share/ansible/roles/rhel-system-roles.certificate/README.md` 文件
- `/usr/share/doc/rhel-system-roles/certificate/` 目录

7.2. 使用 CERTIFICATE RHEL 系统角色请求一个新的自签名证书

使用 **certificate** 系统角色，您可以使用 Ansible Core 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。

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。

步骤

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.certificate
  vars:
    certificate_requests:
      - name: mycert
        dns: "*.example.com"
        ca: self-sign
```

- 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**。

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。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md** 文件
- **/usr/share/doc/rhel-system-roles/certificate/** 目录

7.3. 使用 CERTIFICATE RHEL 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的证书颁发机构(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** 提供者，并通过 **getcert** 命令请求证书。

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。

步骤

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

```
---
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.certificate
  vars:
    certificate_requests:
      - name: mycert
        dns: www.example.com
        principal: HTTP/www.example.com@EXAMPLE.COM
        ca: ipa
```

- 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**。

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。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md** 文件
- **/usr/share/doc/rhel-system-roles/certificate/** 目录

7.4. 使用 CERTIFICATE RHEL 系统角色指定在证书颁发之前或之后要运行的命令

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。

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。

步骤

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

```
---
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.certificate
  vars:
    certificate_requests:
      - name: mycert
        dns: www.example.com
```



```
ca: self-sign
run_before: systemctl stop httpd.service
run_after: systemctl start httpd.service
```

- 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**。

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。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md** 文件
- **/usr/share/doc/rhel-system-roles/certificate/** 目录

第 8 章 使用 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。

8.1. COCKPIT RHEL 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，从而能够从 Web 浏览器管理 RHEL 系统。

8.2. 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

- 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。

步骤

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

```
---
- name: Manage the RHEL web console
  hosts: managed-node-01.example.com
  tasks:
    - name: Install RHEL web console
      ansible.builtin.include_role:
        name: rhel-system-roles.cockpit
  vars:
    cockpit_packages: default
    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. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles-cockpit/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles-cockpit](#) 目录
- [使用 RHEL 系统角色请求证书。](#)

第 9 章 使用 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，使用 Ansible Core 软件包快速且一致地跨多个不同的系统来配置自定义加密策略。

9.1. 使用 CRYPTO_POLICIES RHEL 系统角色设置自定义加密策略

您可以使用 **crypto_policies** 系统角色，从单个控制节点一致地配置大量的受管节点。

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。

步骤

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

```
---
- name: Configure crypto policies
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure crypto policies
      ansible.builtin.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 设置导致系统在系统角色更改了加密策略后重启。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```



警告

因为 **FIPS:OSPP** 系统范围的子策略包含对通用标准(CC)认证所需的加密算法的进一步限制，所以在设置它后系统的互操作性较差。例如，您无法使用少于 3072 位的 RSA 和 DH 密钥、其它 SSH 算法和几个 TLS 组。设置 **FIPS:OSPP** 也会阻止连接到 Red Hat Content Delivery Network (CDN) 结构。另外，您无法将活动目录(AD)集成到使用 **FIPS:OSPP** 的 IdM 部署中，使用 **FIPS:OSPP** 的 RHEL 主机和 AD 域之间的通信可能无法工作，或者某些 AD 帐户可能无法进行身份验证。

请注意，在设置了 **FIPS:OSPP** 加密子策略后，您的系统不符合 CC。使 RHEL 系统符合 CC 标准的唯一正确方法是通过安装 **cc-config** 软件包。有关认证 RHEL 版本、验证报告列表以及 [国家信息保障合作伙伴\(NIAP\)](#) 网站上托管的 CC 指南的列表，请参阅合规性活动和政府标准知识库文章中的 [通用标准](#)。

验证

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

```
---
- name: Verification
  hosts: managed-node-01.example.com
  tasks:
    - name: Verify active crypto policy
      ansible.builtin.include_role:
        name: rhel-system-roles.crypto_policies
    - debug:
        var: crypto_policies_active
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/verify_playbook.yml
```

3. 运行 playbook：

```
$ ansible-playbook ~/verify_playbook.yml
TASK [debug] *****
ok: [host] => {
  "crypto_policies_active": "FUTURE"
}
```

crypto_policies_active 变量显示受管节点上活跃的策略。

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.crypto_policies/README.md` 文件
- `/usr/share/doc/rhel-system-roles/crypto_policies/` 目录

第 10 章 使用 RHEL 系统角色配置 FIREWALLD

您可以使用 **firewall** RHEL 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 **firewall** 角色后，RHEL 系统角色会立即将 **firewalld** 参数应用到受管节点，并使其在重启后保持不变。

10.1. FIREWALL RHEL 系统角色简介

RHEL 系统角色是 Ansible 自动化工具的一组内容。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。

为自动化 **firewalld** 服务的配置引入了 RHEL 系统角色中的 **rhel-system-roles.firewall** 角色。**rhel-system-roles** 软件包包含此 RHEL 系统角色，以及参考文档。

要以自动化方式在一个或多个系统上应用 **firewalld** 参数，请在 playbook 中使用 **firewall** RHEL 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。

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.firewall/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/firewall/](#) 目录
- [使用 playbook](#)
- [如何构建清单](#)

10.2. 使用 FIREWALL RHEL 系统角色重置 FIREWALLD 设置

使用 **firewall** RHEL 系统角色，您可以将 **firewalld** 设置重置为默认状态。如果您将 **previous:replaced** 参数添加到变量列表中，则 RHEL 系统角色会删除所有现有用户定义的设置，并将 **firewalld** 重置回默认设置。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具有 **sudo** 权限。

步骤

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

```
---
- name: Reset firewalld example
  hosts: managed-node-01.example.com
  tasks:
    - name: Reset firewalld
      ansible.builtin.include_role:
        name: rhel-system-roles.firewall
  vars:
    firewall:
      - previous: replaced
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.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
```

其他资源

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

10.3. 使用 FIREWALL RHEL 系统角色，将 FIREWALLD 中的传入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具有 **sudo** 权限。

步骤

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Configure firewall
  hosts: managed-node-01.example.com
  tasks:
    - name: Forward incoming traffic on port 8080 to 443
      ansible.builtin.include_role:
        name: rhel-system-roles.firewall
      vars:
        firewall:
          - { forward_port: 8080/tcp;443;, state: enabled, runtime: true, permanent: true }
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 在受管主机上显示 **firewalld** 设置：

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

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.firewall/README.md` 文件
- `/usr/share/doc/rhel-system-roles/firewall/` 目录

10.4. 使用 FIREWALL RHEL 系统角色管理 FIREWALLD 中的端口

您可以使用 **firewall** RHEL 系统角色在本地防火墙中为传入的流量打开或关闭端口，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具有 **sudo** 权限。

步骤

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Configure firewall
  hosts: managed-node-01.example.com
  tasks:
    - name: Allow incoming HTTPS traffic to the local host
      ansible.builtin.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. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.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
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.firewall/README.md` 文件
- `/usr/share/doc/rhel-system-roles/firewall/` 目录

10.5. 使用 FIREWALL RHEL 系统角色配置 FIREWALLD DMZ 区域

作为系统管理员，您可以使用 **firewall** RHEL 系统角色在 `enp1s0` 接口上配置 **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具有 **sudo** 权限。

步骤

1. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.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
      ansible.builtin.include_role:
        name: rhel-system-roles.firewall
      vars:
        firewall:
          - zone: dmz
            interface: enp1s0
            service: https
            state: enabled
            runtime: true
            permanent: true
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。


3. 运行 playbook：

```
$ ansible-playbook ~/playbook.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:

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.firewall/README.md` 文件
- `/usr/share/doc/rhel-system-roles/firewall/` 目录

第 11 章 使用 RHEL 系统角色配置高可用性集群

通过 **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。

11.1. HA_CLUSTER RHEL 系统角色的变量

在 **ha_cluster** 系统角色 playbook 中，您可以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

启用存储库的布尔值标志，该存储库包含 **ha_cluster** 系统角色所需的软件包。当此变量被设置为默认值 **true** 时，您必须在将用作集群成员的系统上有活跃的覆盖 RHEL 和 RHEL 高可用性附加组件订阅，否则系统角色将失败。

ha_cluster_enable_repos_resilient_storage

(RHEL 9.4 及更高版本)一个布尔值标志，它启用包含弹性存储软件包（如 **dlm** 或 **gfs2**）的存储库。要使用此选项生效，**ha_cluster_enable_repos** 必须被设为 **true**。此变量的默认值为 **false**。

ha_cluster_manage_firewall

(RHEL 9.2 及更高版本)一个布尔值标志，其决定 **ha_cluster** 系统角色是否管理防火墙。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** 被设为 **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 9.2 开始，防火墙不再会被默认配置，因为它仅在 **ha_cluster_manage_firewall** 被设为 **true** 时才进行配置。

ha_cluster_manage_selinux

(RHEL 9.2 及更高版本)一个布尔值标志，其决定 **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。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** 被设为 **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。playbook 中没有指定且不被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。为保护敏感数据，vault 会加密密码，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_hacluster_qdevice_password

(RHEL 9.3 及更高版本)指定仲裁设备的 **hacluster** 用户的密码的字符串值。只有在

ha_cluster_quorum 参数被配置为使用类型 **net** 的仲裁设备，且仲裁设备上 **hacluster** 用户的密码与 **ha_cluster_hacluster_password** 参数指定的 **hacluster** 用户的密码不同时，才需要此参数。**hacluster** 用户对集群具有完全访问权限。为保护敏感数据，vault 会加密密码，如使用 [Ansible Vault 加密内容](#) 中所述。此密码没有默认值。

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一个新密钥，则您应该将密钥复制到节点的 hypervisor，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_src, 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

(RHEL 9.2 及更高版本)使用 **certificate** 系统角色创建一个 **pcs**d 私钥和证书。

如果您的系统没有使用 **pcs**d 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** 系统角色被在内部使用，它按照定义为 **pcs**d 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。如果您没有设置任何这些变量，则 **ha_cluster** 系统角色将使用 **pcs**d 本身创建 **pcs**d 证书。**ha_cluster_pcsd_certificates** 的值被设置为变量 **certificate_requests** 的值，如 **certificate** 系统角色中所指定的。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：

- 除非使用 IPA，并将系统加入到 IPA 域，否则 **certificate** 系统角色创建自签名证书。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 **pcs** 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`

(RHEL 9.1 及更高版本) 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 [在高可用性集群中配置 Corosync 值](#)。

ha_cluster_totem

(RHEL 9.1 及更高版本) 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)** 手册页的 **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 的示例，请参阅在 [在高可用性集群中配置 Corosync 值](#)。

ha_cluster_quorum

(RHEL 9.1 及更高版本) 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）- (RHEL 9.2 及更高版本)将集群配置为使用仲裁设备。默认情况下，不使用仲裁设备。
 - **model**（必需）- 指定仲裁设备型号。仅支持 **net**
 - **model_options**（可选）- 配置指定仲裁设备型号的名称-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** 选项设置连接到 **qnetd** 主机的自定义 **pcsd** 地址和端口。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）- 不特定于型号的名称-值字典设置仲裁设备选项的列表。
 - **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 的示例，请参阅在 [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

(RHEL 9.1 和更高版本) 一个布尔值标记，它决定集群是否可以使用 SBD 节点隔离机制。此变量的默认值为 **false**。

有关启用 SBD 的 **ha_cluster** 系统角色 playbook 的示例，请参阅 [配置具有 SBD 节点隔离的高可用性集群](#)。

ha_cluster_sbd_options

(RHEL 9.1 及更高版本) 指定 SBD 选项的 name-value 字典列表。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。

有关配置 SBD 选项的 **ha_cluster** 系统角色 playbook 的示例，请参阅 [配置具有 SBD 节点隔离的高可用性集群](#)。

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_node_options

(RHEL 9.4 及更高版本)此变量定义各种不同设置，它们因集群节点而异。它为指定节点设置选项，但不指定哪些节点组成集群。您可以在清单或 playbook 中使用 **hosts** 参数指定哪些节点组成集群。

您使用这个变量配置的项目如下：

- **node_name**（必需）- 为其定义 Pacemaker 节点属性的节点的名称。

- **attributes**（可选）- 节点的 Pacemaker 节点属性的集合的列表。目前支持每个节点不超过一个集合。

ha_cluster_node_options 变量的结构如下：

```
ha_cluster_node_options:
  - node_name: node1
    attributes:
      - attrs:
          - name: attribute1
            value: value1_node1
          - name: attribute2
            value: value2_node1
  - node_name: node2
    attributes:
      - attrs:
          - name: attribute1
            value: value1_node2
          - name: attribute2
            value: value2_node2
```

默认情况下，没有定义节点选项。

有关包含节点选项配置的 **ha_cluster** 系统角色 playbook 的示例，请参阅 [配置带有节点属性的高可用性集群](#)。

ha_cluster_resource_primitives

此变量定义系统角色配置的 pacemaker 资源，包括隔离资源。您可以为每个资源配置以下项目：

- **id**（必需）- 资源的 ID。
- **agent**（必需）- 资源或隔离代理的名称，如 **ocf:pacemaker:Dummy** 或 **stonith:fence_xvm**。对于 STONITH 代理，必须指定 **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，以及它们是否是必需的，取决于资源或隔离代理。
- **meta_attrs**（可选）- 资源的元属性集合列表。目前，只支持一个集合。
- **copy_operations_from_agent**（可选）- (RHEL 9.3 及更新版本)资源代理通常会为资源操作定义默认设置，如 **interval** 和 **timeout**，为特定代理进行了优化。如果此变量设为 **true**，则这些设置将被复制到资源配置。否则，集群范围的默认值应用到资源。如果您使用 **ha_cluster_resource_operation_defaults** 角色变量为资源定义资源操作默认值，则您可以将其设置为 **false**。此变量的默认值为 **true**。
- **operations**（可选）- 资源操作列表。
 - **action**（必需）- pacemaker 以及资源或隔离代理定义的操作。
 - **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
  copy_operations_from_agent: bool
  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 [配置带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 [配置带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 [配置带有隔离和资源的高可用性集群](#)。

ha_cluster_resource_defaults

(RHEL 9.3 及更新版本)此变量定义资源默认值的集合。您可以定义多个默认值集合，并使用规则将它们应用到特定代理的资源。您使用 **ha_cluster_resource_defaults** 变量指定的默认值不会应用到使用它们自己定义值的覆盖它们的资源。

只有 meta 属性可以被指定为默认值。

您可以为每个默认值集合配置以下项目：

- **id**（可选）- 默认值集合的 ID。如果没有指定，它会自动生成。
- **rule**（可选）- 使用 **pcs** 语法编写的规则，定义何时以及集合适用于哪些资源。有关指定规则的详情，请查看 **pcs(8)**手册页中的 [资源默认值集合创建](#) 部分。
- **score**（可选）- 默认值集合的权重。

- **attrs**（可选）- 作为默认值应用到资源的元数据属性。

ha_cluster_resource_defaults 变量的结构如下：

```
ha_cluster_resource_defaults:
  meta_attrs:
    - id: defaults-set-1-id
      rule: rule-string
      score: score-value
      attrs:
        - name: meta_attribute1_name
          value: meta_attribute1_value
        - name: meta_attribute2_name
          value: meta_attribute2_value
    - id: defaults-set-2-id
      rule: rule-string
      score: score-value
      attrs:
        - name: meta_attribute3_name
          value: meta_attribute3_value
        - name: meta_attribute4_name
          value: meta_attribute4_value
```

有关配置资源默认值的 **ha_cluster** 系统角色 playbook 的示例，请参阅 [默认具有资源和资源操作默认值的高可用性集群](#)。

ha_cluster_resource_operation_defaults

(RHEL 9.3 及更新版本)此变量定义资源操作默认值集合。您可以定义多个默认值集合，并使用规则将它们应用到特定代理的资源和特定的资源操作。使用 **ha_cluster_resource_operation_defaults** 变量指定的默认值不应用到使用它们自己定义的值覆盖它们的资源操作。默认情况下，**ha_cluster** 系统角色配置资源，来为资源操作定义自己的值。有关使用 **ha_cluster_resource_operations_defaults** 变量覆盖这些默认值的详情，请查看 **ha_cluster_resource_primitives** 中的 **copy_operations_from_agent** 项的描述。

只有 meta 属性可以被指定为默认值。

ha_cluster_resource_operations_defaults 变量的结构与 **ha_cluster_resource_defaults** 变量的结构一样，但指定规则的方式除外。有关指定规则来描述集合应用到的资源操作的详情，请查看 **pcs(8)** 手册页中的 **资源操作默认值集合创建** 部分。

ha_cluster_stonith_levels

(RHEL 9.4 及更高版本)此变量定义 STONITH 级别，也称为隔离拓扑。隔离级别将集群为使用多个设备来隔离节点。如果一个设备失败了，您可以定义替代设备，且您可以要求多个设备成功执行，来将节点视为成功隔离。有关隔离级别的更多信息，请参阅 [配置和管理高可用性集群](#) 中的 [配置隔离级别](#)。您可以在定义隔离级别时配置以下项目：

- **level**（必需）- 尝试隔离级别的顺序。Pacemaker 会按照升序尝试级别，直到一个成功。
- **target**（可选）- 此级别应用到的节点的名称。
- 您必须指定以下三个选择之一：
 - **target_pattern** - 与这个级别应用到的节点名称匹配的 POSIX 扩展正则表达式。
 - **target_attribute** - 为此级别应用到的节点设置的节点属性的名称。

- **target_attribute** 和 **target_value** - 为此级别应用到的节点设置的节点属性的名称和值。
- **resource_ids**（必需）- 必须为此级别尝试的所有隔离资源的列表。
默认情况下，没有定义隔离级别。

使用 **ha_cluster** 系统角色配置的隔离级别定义的结构如下：

```
ha_cluster_stonith_levels:
- level: 1..9
  target: node_name
  target_pattern: node_name_regular_expression
  target_attribute: node_attribute_name
  target_value: node_attribute_value
  resource_ids:
    - fence_device_1
    - fence_device_2
- level: 1..9
  target: node_name
  target_pattern: node_name_regular_expression
  target_attribute: node_attribute_name
  target_value: node_attribute_value
  resource_ids:
    - fence_device_1
    - fence_device_2
```

有关配置隔离默认值的 **ha_cluster** 系统角色 playbook 的示例，请参阅 [配置具有隔离级别的高可用性集群](#)。

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 的示例，请参阅 [配置具有资源限制的高可用性集群](#)。

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 的示例，请参阅 [配置具有资源限制的高可用性集群](#)。

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
```

有关创建具有资源限制的集群的 **ha_cluster** 系统角色 playbook 的示例，请参阅 [配置具有资源限制的高可用性集群](#)。

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 的示例，请参阅 [配置具有资源限制的高可用性集群](#)。

ha_cluster_qnetd

(RHEL 9.2 及更高版本)此变量配置一个 **qnetd** 主机，然后其充当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。

有关配置使用仲裁设备的集群的 **ha_cluster** 系统角色 playbook 的示例，请参阅 [配置使用仲裁设备的集群](#)。

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.ha_cluster/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ha_cluster/` 目录

11.2. 为 HA_CLUSTER RHEL 系统角色指定一个清单

使用 **ha_cluster** 系统角色 playbook 配置 HA 集群时，您可以为清单中的集群配置节点的名称和地址。

11.2.1. 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
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles/ha_cluster/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ha_cluster/` 目录

11.2.2. 在清单中配置 watchdog 和 SBD 设备

(RHEL 9.1 及更高版本)使用 SBD 时，您可以选择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并可从所有节点访问，但每个节点可以对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 中设置的 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` 条目。

对于清单中的每个节点，您可以选择指定以下项目：

- **sbd_watchdog_modules** (可选) - (RHEL 9.3 及更新版本)要载入的 Watchdog 内核模块，这将创建 `/dev/watchdog*` 设备。如果没有设置，则默认为空列表。
- **sbd_watchdog_modules_blocklist** (可选) - (RHEL 9.3 及更新版本)要卸载和阻止的 Watchdog 内核模块。如果没有设置，则默认为空列表。
- **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_modules:
      - module1
      - module2
    sbd_watchdog: /dev/watchdog2
    sbd_devices:
      - /dev/vdx
      - /dev/vdy
node2:
  ha_cluster:
    sbd_watchdog_modules:
      - module1
    sbd_watchdog_modules_blocklist:
      - module2
    sbd_watchdog: /dev/watchdog1
    sbd_devices:
      - /dev/vdw
      - /dev/vdz

```

有关创建使用 SBD 隔离的高可用性集群的详情，请参考 [配置具有 SBD 节点隔离的高可用性集群](#)。

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.ha_cluster/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/ha_cluster/](#) 目录

11.3. 为高可用性集群创建 PCSD TLS 证书和密钥文件

(RHEL 9.2 及更高版本)

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** 系统角色在内部使用 **certificate** 系统角色来管理 TLS 证书。



警告

ha_cluster 系统角色替换指定节点上任何现有的群集配置。playbook 中未指定的任何设置都将丢失。

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。
- 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。
- 清单文件指定集群节点，如 [为 ha_cluster 系统角色指定清单](#) 中所述。

步骤

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Create TLS certificates and key files in a high availability cluster
  hosts: node1 node2
  roles:
    - rhel-system-roles.ha_cluster
  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
```

此 playbook 配置运行 **firewalld** 和 **selinux** 服务的集群，并在 `/var/lib/pcs` 中创建自签名 **pcs** 证书和私钥文件。**pcs** 证书具有文件名 **FILENAME.crt**，密钥文件名为 **FILENAME.key**。

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。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.ha_cluster/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ha_cluster/` 目录 [使用 RHEL 系统角色请求证书](#)

11.4. 配置不运行任何资源的高可用性集群

以下流程使用 **ha_cluster** 系统角色来创建没有配置隔离并且没有运行任何资源的高可用性集群。



警告

ha_cluster 系统角色替换指定节点上任何现有的群集配置。playbook 中未指定的任何设置都将丢失。

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。
- 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。
- 清单文件指定集群节点，如 [为 ha_cluster 系统角色指定清单](#) 中所述。

步骤

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Create a high availability cluster with no fencing and which runs no resources
  hosts: node1 node2
  roles:
    - rhel-system-roles.ha_cluster
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: <password>
    ha_cluster_manage_firewall: true
    ha_cluster_manage_selinux: true
```

这个示例 playbook 文件配置一个运行 **firewalld** 和 **selinux** 服务，而没有配置隔离的集群，其没有运行任何资源。

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。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.ha_cluster/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ha_cluster/` 目录

11.5. 配置带有隔离和资源的高可用性集群

以下流程使用 **ha_cluster** 系统角色来创建高可用性集群，该集群包含隔离设备、集群资源、资源组和克隆的资源。



警告

ha_cluster 系统角色替换指定节点上任何现有的群集配置。playbook 中未指定的任何设置都将丢失。

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。
- 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。
- 清单文件指定集群节点，如 [为 ha_cluster 系统角色指定清单](#) 中所述。

步骤

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Create a high availability cluster that includes a fencing device and resources
  hosts: node1 node2
  roles:
    - rhel-system-roles.ha_cluster
  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

```

这个示例 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。

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。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles/ha_cluster/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ha_cluster/` 目录

11.6. 配置具有资源和资源操作默认值的高可用性集群

(RHEL 9.3 及更高版本)以下流程使用 **ha_cluster** 系统角色创建一个定义资源和资源操作默认值的高可用性集群。



警告

ha_cluster 系统角色替换指定节点上任何现有的群集配置。playbook 中未指定的任何设置都将丢失。

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。
- 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。
- 清单文件指定集群节点，如 [为 ha_cluster 系统角色指定清单](#) 中所述。

步骤

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Create a high availability cluster that defines resource and resource operation
  defaults
    hosts: node1 node2
    roles:
```

```

- rhel-system-roles.ha_cluster
vars:
  ha_cluster_cluster_name: my-new-cluster
  ha_cluster_hacluster_password: <password>
  ha_cluster_manage_firewall: true
  ha_cluster_manage_selinux: true
  # Set a different resource-stickiness value during
  # and outside work hours. This allows resources to
  # automatically move back to their most
  # preferred hosts, but at a time that
  # does not interfere with business activities.
  ha_cluster_resource_defaults:
    meta_attrs:
      - id: core-hours
        rule: date-spec hours=9-16 weekdays=1-5
        score: 2
      attrs:
        - name: resource-stickiness
          value: INFINITY
      - id: after-hours
        score: 1
      attrs:
        - name: resource-stickiness
          value: 0
    # Default the timeout on all 10-second-interval
    # monitor actions on IPAddr2 resources to 8 seconds.
  ha_cluster_resource_operation_defaults:
    meta_attrs:
      - rule: resource ::IPAddr2 and op monitor interval=10s
        score: INFINITY
      attrs:
        - name: timeout
          value: 8s

```

这个示例 playbook 文件配置一个运行 **firewalld** 和 **selinux** 服务的集群。集群包括资源和资源操作默认值。

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。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.ha_cluster/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ha_cluster/` 目录

11.7. 配置具有隔离级别的高可用性集群

(RHEL 9.4 及更高版本)以下流程使用 **ha_cluster** 系统角色创建一个定义了隔离级别的高可用性集群。



警告

ha_cluster 系统角色替换指定节点上任何现有的群集配置。playbook 中未指定的任何设置都将丢失。

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。
- 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。
- 清单文件指定集群节点，如 [为 ha_cluster 系统角色指定清单](#) 中所述。有关创建清单文件的常规信息，请参阅 [在 RHEL 9 上准备一个控制节点](#)。

流程

1. 将您的敏感变量存储在一个加密文件中：

- a. 创建 vault：

```
$ ansible-vault create vault.yml
New Vault password: <vault_password>
Confirm New Vault password: <vault_password>
```

- b. 在 **ansible-vault create** 命令打开编辑器后，以 **<key>: <value>** 格式输入敏感数据：

```
cluster_password: <cluster_password>
fence1_password: <fence1_password>
fence2_password: <fence2_password>
```

- c. 保存更改，并关闭编辑器。Ansible 加密 vault 中的数据。

2. 创建一个 playbook 文件，如 **~/playbook.yml**。这个示例 playbook 文件配置一个运行 **firewalld** 和 **selinux** 服务的集群。

```
---
- name: Create a high availability cluster
  hosts: node1 node2
  vars_files:
    - vault.yml
  tasks:
    - name: Configure a cluster that defines fencing levels
```

```

ansible.builtin.include_role:
  name: rhel-system-roles.ha_cluster
vars:
  ha_cluster_cluster_name: my-new-cluster
  ha_cluster_hacluster_password: "{{ cluster_password }}"
  ha_cluster_manage_firewall: true
  ha_cluster_manage_selinux: true
  ha_cluster_resource_primitives:
    - id: apc1
      agent: 'stonith:fence_apc_snmp'
      instance_attrs:
        - attrs:
            - name: ip
              value: apc1.example.com
            - name: username
              value: user
            - name: password
              value: "{{ fence1_password }}"
            - name: pcmk_host_map
              value: node1:1;node2:2
    - id: apc2
      agent: 'stonith:fence_apc_snmp'
      instance_attrs:
        - attrs:
            - name: ip
              value: apc2.example.com
            - name: username
              value: user
            - name: password
              value: "{{ fence2_password }}"
            - name: pcmk_host_map
              value: node1:1;node2:2
  # Nodes have redundant power supplies, apc1 and apc2. Cluster must
  # ensure that when attempting to reboot a node, both power
  # supplies # are turned off before either power supply is turned
  # back on.
  ha_cluster_stonith_levels:
    - level: 1
      target: node1
      resource_ids:
        - apc1
        - apc2
    - level: 1
      target: node2
      resource_ids:
        - apc1
        - apc2

```

3. 验证 playbook 语法 :

```
$ ansible-playbook --syntax-check --ask-vault-pass ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

4. 运行 playbook :

```
$ ansible-playbook --ask-vault-pass ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.ha_cluster/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ha_cluster/` 目录
- [Ansible vault](#)

11.8. 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。



警告

ha_cluster 系统角色替换指定节点上任何现有的群集配置。playbook 中未指定的任何设置都将丢失。

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。
- 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。
- 清单文件指定集群节点，如 [为 ha_cluster 系统角色指定清单](#) 中所述。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Create a high availability cluster with resource constraints
  hosts: node1 node2
  roles:
    - rhel-system-roles.ha_cluster
  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

```

这个示例 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。

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。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles/ha_cluster/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ha_cluster/` 目录

11.9. 在高可用性集群中配置 COROSYNC 值

(RHEL 9.1 及更高版本)以下流程使用 **ha_cluster** 系统角色创建一个配置了 Corosync 值的高可用性集群。



警告

ha_cluster 系统角色替换指定节点上任何现有的群集配置。playbook 中未指定的任何设置都将丢失。

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。
- 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。
- 清单文件指定集群节点，如 [为 ha_cluster 系统角色指定清单](#) 中所述。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Create a high availability cluster that configures Corosync values
  hosts: node1 node2
```

```

roles:
  - rhel-system-roles.ha_cluster
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

```

这个示例 playbook 文件配置一个运行 **firewalld** 和 **selinux** 服务的集群，该集群配置了 Corosync 属性。

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。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.ha_cluster/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ha_cluster/` 目录

11.10. 使用 SBD 节点隔离配置高可用性集群

(RHEL 9.1 及更高版本)以下流程使用 **ha_cluster** 系统角色创建一个使用 SBD 节点隔离的高可用性集群。



警告

ha_cluster 系统角色替换指定节点上任何现有的群集配置。playbook 中未指定的任何设置都将丢失。

此 playbook 使用一个载入 watchdog 模块（在 RHEL 9.3 及之后版本中支持）的清单文件，如 [在清单中配置 watchdog 和 SBD 设备](#) 中所述。

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。
- 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。
- 清单文件指定集群节点，如 [为 ha_cluster 系统角色指定清单](#) 中所述。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Create a high availability cluster that uses SBD node fencing
  hosts: node1 node2
  roles:
    - rhel-system-roles.ha_cluster
  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: 30
# Suggested optimal values for SBD timeouts:
# watchdog-timeout * 2 = msgwait-timeout (set automatically)
# msgwait-timeout * 1.2 = stonith-timeout
ha_cluster_cluster_properties:
  - attrs:
    - name: stonith-timeout
      value: 72
ha_cluster_resource_primitives:
  - id: fence_sbd
    agent: 'stonith:fence_sbd'
    instance_attrs:
      - attrs:
        # taken from host_vars
        - name: devices
          value: "{{ ha_cluster.sbd_devices | join(',') }}"
        - name: pcmk_delay_base
          value: 30

```

这个示例 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，并创建 SBD Stonith 资源。

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。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.ha_cluster/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ha_cluster/` 目录

11.11. 使用仲裁设备配置高可用性集群

(RHEL 9.2 及更高版本)要使用 **ha_cluster** 系统角色配置带有独立仲裁设备的高可用性集群，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。

11.11.1. 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。



警告

ha_cluster 系统角色替换指定节点上任何现有的群集配置。playbook 中未指定的任何设置都将丢失。

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。
- 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。
- 清单文件指定仲裁设备，如 [为 ha_cluster 系统角色指定清单](#) 中所述。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Configure a quorum device
  hosts: nodeQ
  roles:
    - rhel-system-roles.ha_cluster
  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
```

这个示例 playbook 文件在运行 **firewalld** 和 **selinux** 服务的系统上配置一个仲裁设备。

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。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.ha_cluster/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ha_cluster/` 目录

11.11.2. 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。



警告

ha_cluster 系统角色替换指定节点上任何现有的群集配置。playbook 中未指定的任何设置都将丢失。

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。
- 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。
- 清单文件指定集群节点，如 [为 ha_cluster 系统角色指定清单](#) 中所述。
- 您已配置了一个仲裁设备。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Configure a cluster to use a quorum device
  hosts: node1 node2
  roles:
    - rhel-system-roles.ha_cluster
  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
```

这个示例 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。

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。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.ha_cluster/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ha_cluster/` 目录

11.12. 配置具有节点属性的高可用性集群

(RHEL 9.4 及更高版本)以下流程使用 **ha_cluster** 系统角色创建一个配置了节点属性的高可用性集群。

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任何现有的群集配置。playbook 中未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 [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，且为集群中的节点配置了节点属性。

```
- 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_node_options:
      - node_name: node1
        attributes:
          - attrs:
              - name: attribute1
                value: value1A
              - name: attribute2
                value: value2A
      - node_name: node2
        attributes:
          - attrs:
              - name: attribute1
                value: value1B
              - name: attribute2
                value: value2B

  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
```

11.13. 使用 HA_CLUSTER RHEL 系统角色在高可用性集群中配置一个 APACHE HTTP 服务器

此流程使用 **ha_cluster** 系统角色在双节点 Red Hat Enterprise Linux 高可用性附加组件集群中配置一个主动/被动 Apache HTTP 服务器。



警告

ha_cluster 系统角色替换指定节点上任何现有的群集配置。playbook 中未指定的任何设置都将丢失。

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。
- 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。
- 清单文件指定集群节点，如 [为 ha_cluster 系统角色指定清单](#) 中所述。
- 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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Configure active/passive Apache server in a high availability cluster
  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

```

这个示例 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。

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。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook :

```
$ ansible-playbook ~/playbook.yml
```

- 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

- 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
```

- 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
```

- 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
```

- 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。

- 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](#)。

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.ha_cluster/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ha_cluster/` 目录

第 12 章 使用 RHEL 系统角色配置 SYSTEMD 日志

使用 **journald** RHEL 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。

12.1. 使用 JOURNALD RHEL 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** RHEL 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** RHEL 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。

流程

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

```
---
- name: Configure persistent logging
  hosts: managed-node-01.example.com
  vars:
    journald_persistent: true
    journald_max_disk_size: 2048
    journald_per_user: true
    journald_sync_interval: 1
  roles:
    - rhel-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. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.journald/README.md` 文件
- `/usr/share/doc/rhel-system-roles/journald/` 目录

第 13 章 使用 RHEL 系统角色配置自动崩溃转储

要使用 Ansible 管理 kdump，您可以使用 **kdump** 角色，这是 RHEL 9 中提供的 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。

13.1. 使用 **KDUMP** RHEL 系统角色配置内核崩溃转储机制

您可以通过运行 Ansible playbook，使用 **kdump** 系统角色在多个系统上设置基本内核转储参数。



警告

kdump 系统角色通过替换 `/etc/kdump.conf` 文件完全替换了受管主机的 **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

- 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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.kdump
  vars:
    kdump_path: /var/crash
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.kdump/README.md` 文件

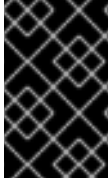
- **/usr/share/doc/rhel-system-roles/kdump/** 目录

第 14 章 使用 RHEL 系统角色永久配置内核参数

您可以使用 **kernel_settings** RHEL 系统角色一次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 渠道提供的 RHEL 系统角色在默认的 AppStream 存储库中作为 RPM 软件包提供给 RHEL 客户。RHEL 系统角色还可作为通过 Ansible Automation Hub 为客户提供 Ansible 订阅的集合。

14.1. KERNEL_SETTINGS RHEL 系统角色简介

RHEL 系统角色是一组角色，其为远程管理多个系统提供一致的配置接口。

引入 RHEL 系统角色是为了使用 **kernel_settings** RHEL 系统角色自动化内核的配置。**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/ansible/roles/rhel-system-roles.kernel_settings/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/kernel_settings/](#) 目录
- [使用 playbook](#)
- [如何构建清单](#)

14.2. 使用 KERNEL_SETTINGS RHEL 系统角色应用所选的内核参数

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。

流程

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

```
---
- name: Configure kernel settings
  hosts: managed-node-01.example.com
  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:** 可选键，其将任意字符串与 play 关联来作为标签，并标识 play 的用途。
- **hosts:** 在 play 中的键，其指定运行 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:** playbook 部分，其表示包含所选内核参数名称和必须设置成的值的变量的列表。
- **role:** 键，其指定哪个 RHEL 系统角色将配置参数和 **vars** 部分中提到的值。



注意

您可以修改 playbook 中的内核参数及其值以符合您的需要。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

4. 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

- `/usr/share/ansible/roles/rhel-system-roles.kernel_settings/README.md` 文件
- `/usr/share/doc/rhel-system-roles/kernel_settings/` 目录
- [使用 Playbook](#)
- [使用变量](#)
- [角色](#)

第 15 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** RHEL 系统角色将 Red Hat Enterprise Linux 主机配置为日志服务器，以从多个客户端系统收集日志。

15.1. LOGGING RHEL 系统角色

使用 **logging** RHEL 系统角色，您可以在本地和远程主机上部署日志配置。

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**
- 网络上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** RHEL 系统角色，您可以组合输入和输出，以适应您的场景。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。

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.logging/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/logging/](#) 目录
- [RHEL 系统角色](#)

15.2. 应用一个本地 LOGGING RHEL 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

- [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。



注意

您不必安装 **rsyslog** 软件包，因为 RHEL 系统角色在部署时会安装 **rsyslog**。

流程

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

```
---
- name: Deploying basics input and implicit files output
  hosts: managed-node-01.example.com
  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. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/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**。

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.logging/README.md` 文件
- `/usr/share/doc/rhel-system-roles/logging/` 目录

15.3. 在本地 LOGGING RHEL 系统角色中过滤日志

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

- 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。



注意

您不必安装 **rsyslog** 软件包，因为 RHEL 系统角色在部署时会安装 **rsyslog**。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Deploying files input and configured files output
  hosts: managed-node-01.example.com
  roles:
    - rhel-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，所有包含 **error** 字符串的消息都记录在 `/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** 属性值替换为您要用来过滤的字符串。

您可以根据您的偏好修改变量。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/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

- `/usr/share/ansible/roles/rhel-system-roles/logging/README.md` 文件
- `/usr/share/doc/rhel-system-roles/logging/` 目录

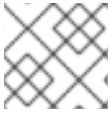
15.4. 使用 LOGGING RHEL 系统角色应用一个远程日志解决方案

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

- 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。



注意

您不必安装 **rsyslog** 软件包，因为 RHEL 系统角色在部署时会安装 **rsyslog**。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Deploying remote input and remote_files output
  hosts: managed-node-01.example.com
  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: managed-node-02.example.com
  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>** 是日志记录服务器。



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. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

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/<host2.example.com>/messages** 日志，例如：

```
# cat /var/log/<host2.example.com>/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

- `/usr/share/ansible/roles/rhel-system-roles.logging/README.md` 文件
- `/usr/share/doc/rhel-system-roles/logging/` 目录

15.5. 使用带有 TLS 的 LOGGING RHEL 系统角色

传输层安全性(TLS)是一种加密协议，旨在允许通过计算机网络的安全通信。

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。

15.5.1. 配置带有 TLS 的客户端日志

您可以使用 Ansible playbook 和 **logging** RHEL 系统角色，来在 RHEL 客户端上配置日志，并使用 TLS 加密将日志传送到远程日志系统上。

此流程创建一个私钥和证书，并在 Ansible 清单中的客户端组的所有主机上配置 TLS。TLS 对信息的传输进行加密，确保日志在网络安全传输。



注意

您不必在 playbook 中调用 **certificate** RHEL 系统角色来创建证书。**logging** RHEL 系统角色会自动调用它。

要让 CA 能够为创建的证书签名，受管节点必须在 IdM 域中注册。

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。
- 受管节点已在 IdM 域中注册。
- 如果要在管理节点上配置的日志服务器运行 RHEL 9.2 或更高版本，且启用了 FIPS 模式，客户端必须支持扩展 Master Secret (EMS) 扩展或使用 TLS 1.3。没有 EMS 的 TLS 1.2 连接会失败。如需更多信息，请参阅 [强制 TLS 扩展"Extended Master Secret"](#) 知识库文章。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Deploying files input and forwards output with certs
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_certificates:
      - name: logging_cert
        dns: ['localhost', 'www.example.com']
```

```

ca: ipa
logging_pki_files:
- ca_cert: /local/path/to/ca_cert.pem
  cert: /local/path/to/logging_cert.pem
  private_key: /local/path/to/logging_cert.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_certificates

此参数的值传递给 **certificate** RHEL 系统角色中的 **certificate_requests**，并用来创建一个私钥和证书。

logging_pki_files

使用这个参数，您可以配置日志记录用来查找用于 TLS 的 CA、证书和密钥文件的路径和其他设置，使用以下一个或多个子参数指定：**ca_cert**、**ca_cert_src**、**cert**、**cert_src**、**private_key**、**private_key_src** 和 **tls**。



注意

如果您使用 **logging_certificates** 在目标节点上创建文件，请不要使用 **ca_cert_src**、**cert_src** 和 **private_key_src**，它们用于复制不是由 **logging_certificates** 创建的文件。

ca_cert

表示目标节点上 CA 证书文件的路径。默认路径为 **/etc/pki/tls/certs/ca.pem**，文件名由用户设置。

cert

表示目标节点上证书文件的路径。默认路径为 **/etc/pki/tls/certs/server-cert.pem**，文件名由用户设置。

private_key

表示目标节点上私钥文件的路径。默认路径为 **/etc/pki/tls/private/server-key.pem**，文件名由用户设置。

ca_cert_src

代表控制节点上 CA 证书文件的路径，该路径将复制到目标主机上 **ca_cert** 指定的位置。如果使用 **logging_certificates**，请不要使用它。

cert_src

表示控制节点上证书文件的路径，其将被复制到目标主机上 **cert** 指定的位置。如果使用 **logging_certificates**，请不要使用它。

private_key_src

表示控制节点上私钥文件的路径，其将被复制到目标主机上 **private_key** 指定的位置。如果使用 **logging_certificates**，请不要使用它。

tls

将此参数设置为 **true** 可确保通过网络安全地传输日志。如果您不想要安全封装器，您可以设置 **tls: false**。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles/logging/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/logging/](#) 目录
- [使用 RHEL 系统角色请求证书。](#)

15.5.2. 配置带有 TLS 的服务器日志

您可以使用 Ansible playbook 和 **logging** RHEL 系统角色，来在 RHEL 服务器上配置日志，并将它们设置为使用 TLS 加密从远程日志系统接收日志。

此流程创建一个私钥和证书，并在 Ansible 清单的服务器组中的所有主机上配置 TLS。



注意

您不必在 playbook 中调用 **certificate** RHEL 系统角色来创建证书。**logging** RHEL 系统角色会自动调用它。

要让 CA 能够为创建的证书签名，受管节点必须在 IdM 域中注册。

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。
- 受管节点已在 IdM 域中注册。
- 如果要在管理节点上配置的日志服务器运行 RHEL 9.2 或更高版本，且启用了 FIPS 模式，客户端必须支持扩展 Master Secret (EMS) 扩展或使用 TLS 1.3。没有 EMS 的 TLS 1.2 连接会失败。如需更多信息，请参阅 [强制 TLS 扩展"Extended Master Secret"](#) 知识库文章。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Deploying remote input and remote_files output with certs
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.logging
  vars:
    logging_certificates:
      - name: logging_cert
        dns: ['localhost', 'www.example.com']
        ca: ipa
    logging_pki_files:
      - ca_cert: /local/path/to/ca_cert.pem
        cert: /local/path/to/logging_cert.pem
        private_key: /local/path/to/logging_cert.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_certificates

此参数的值传递给 **certificate** RHEL 系统角色中的 **certificate_requests**，并用来创建一个私钥和证书。

logging_pki_files

使用这个参数，您可以配置日志记录用来查找用于 TLS 的 CA、证书和密钥文件的路径和其他设置，使用以下一个或多个子参数指定：**ca_cert**、**ca_cert_src**、**cert**、**cert_src**、**private_key**、**private_key_src** 和 **tls**。



注意

如果您使用 **logging_certificates** 在目标节点上创建文件，请不要使用 **ca_cert_src**、**cert_src** 和 **private_key_src**，它们用于复制不是由 **logging_certificates** 创建的文件。

ca_cert

表示目标节点上 CA 证书文件的路径。默认路径为 `/etc/pki/tls/certs/ca.pem`，文件名由用户设置。

cert

表示目标节点上证书文件的路径。默认路径为 `/etc/pki/tls/certs/server-cert.pem`，文件名由用户设置。

private_key

表示目标节点上私钥文件的路径。默认路径为 `/etc/pki/tls/private/server-key.pem`，文件名由用户设置。

ca_cert_src

代表控制节点上 CA 证书文件的路径，该路径将复制到目标主机上 **ca_cert** 指定的位置。如果使用 **logging_certificates**，请不要使用它。

cert_src

表示控制节点上证书文件的路径，其将被复制到目标主机上 **cert** 指定的位置。如果使用 **logging_certificates**，请不要使用它。

private_key_src

表示控制节点上私钥文件的路径，其将被复制到目标主机上 **private_key** 指定的位置。如果使用 **logging_certificates**，请不要使用它。

tls

将此参数设置为 **true** 可确保通过网络安全地传输日志。如果您不想要安全封装器，您可以设置 **tls: false**。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles/logging/README.md` 文件
- `/usr/share/doc/rhel-system-roles/logging/` 目录
- [使用 RHEL 系统角色请求证书](#)。

15.6. 将 RELP 与 LOGGING RHEL 系统角色一起使用

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** RHEL 系统角色将日志记录系统配置为可靠发送和接收日志条目。

15.6.1. 使用 RELP 配置客户端日志

您可以使用 **logging** RHEL 系统角色在本地机器上记录日志的 RHEL 系统中配置日志，并通过运行 Ansible playbook，使用 RELP 将日志转发到远程日志系统。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Deploying basic input and relp output
  hosts: managed-node-01.example.com
  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。

port

远程日志记录系统正在监听的端口号。

tls

此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。

确保日志在网络上安全地传输。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}** 三元组，则默认位置 **/etc/pki/tls/certs** 和 **/etc/pki/tls/private** 被用作受管节点上的目的地，以便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}** 三元组，则文件在日志配置前应位于默认路径上。
- 如果两个三元组都设置了，则文件将从控制节点的本地路径传输到受管节点的特定路径。

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

表示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

表示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。

pki_authmode

接受身份验证模式为 **name** 或 **fingerprint**。

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

日志输入字典列表。

输出

日志输出字典列表。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

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

- `/usr/share/doc/rhel-system-roles/logging/` 目录

15.6.2. 配置带有 RELP 的服务器日志

您可以使用 **logging** RHEL 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，使用 RELP 从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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Deploying remote input and remote_files output
  hosts: managed-node-01.example.com
  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：

port

远程日志记录系统正在监听的端口号。

tls

确保日志在网络上安全地传输。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}** 三元组，则默认位置 **/etc/pki/tls/certs** 和 **/etc/pki/tls/private** 被用作受管节点上的目的地，以便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}** 三元组，则文件在日志配置前应位于默认路径上。
- 如果两个三元组都设置了，则文件将从控制节点的本地路径传输到受管节点的特定路径。

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

表示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

表示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。

pki_authmode

接受身份验证模式为 **name** 或 **fingerprint**。

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

日志输入字典列表。

输出

日志输出字典列表。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

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

- `/usr/share/doc/rhel-system-roles/logging/` 目录

第 16 章 使用 RHEL 系统角色监控性能

作为系统管理员，您可以使用 **metrics** RHEL 系统角色监控系统的性能。

16.1. METRICS RHEL 系统角色简介

RHEL 系统角色是 Ansible 角色和模块的集合，其提供了一个一致的配置接口来远程管理多个 RHEL 系统。**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** 来监控您的系统性能，而无需单独配置 **pcp**，因为 **pcp** 的设置和部署已由 playbook 处理。

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.metrics/README.md` 文件
- `/usr/share/doc/rhel-system-roles/metrics/` 目录

16.2. 使用 METRICS RHEL 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

- 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。
- **localhost** 在控制节点上的清单文件中被配置：

```
localhost ansible_connection=local
```

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Manage metrics
  hosts: localhost
  roles:
    - rhel-system-roles.metrics
  vars:
    metrics_graph_service: yes
    metrics_manage_firewall: true
    metrics_manage_selinux: true
```

因为 **metrics_graph_service** 布尔值被设置为 **value="yes"**，所以 **Grafana** 被自动安装和置备，并使用 **pcp** 添加为一个数据源。因为 **metrics_manage_firewall** 和 **metrics_manage_selinux** 都被设为 **true**，所以 **metrics** 角色使用 **firewall** 和 **selinux** 系统角色来管理 **metrics** 角色使用的端口。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 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。

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.metrics/README.md` 文件
- `/usr/share/doc/rhel-system-roles/metrics/` 目录

16.3. 使用 METRICS RHEL 系统角色将单独的系统设置成一组来监控其自身

此流程描述了如何使用 **metrics** 系统角色将机器设置成一组来监控其自身。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Configure a fleet of machines to monitor themselves
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.metrics
  vars:
    metrics_retention_days: 0
    metrics_manage_firewall: true
    metrics_manage_selinux: true
```

因为 **metrics_manage_firewall** 和 **metrics_manage_selinux** 都被设为 **true**，所以 **metrics** 角色使用 **firewall** 和 **selinux** 角色来管理 **metrics** 角色使用的端口。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook :

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.metrics/README.md` 文件
- `/usr/share/doc/rhel-system-roles/metrics/` 目录

16.4. 使用 METRICS RHEL 系统角色，使用本地机器集中监控一组机器

此流程描述了如何使用 **metrics** 系统角色设置本地机器，来集中监控一组机器，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

- 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。
- **localhost** 在控制节点上的清单文件中被配置：

```
localhost ansible_connection=local
```

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
- name: Set up your local machine to centrally monitor a fleet of machines
  hosts: localhost
  roles:
    - rhel-system-roles.metrics
  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
```

因为 **metrics_graph_service** 和 **metrics_query_service** 布尔值都被设置为 **value="yes"**，所以 **grafana** 被自动安装和置备，并使用 **pcp** 添加为一个数据源，使用 **pcp** 将数据记录索引到 **redis** 中，允许 **pcp** 查询语言用于复杂的数据查询。因为 **metrics_manage_firewall** 和 **metrics_manage_selinux** 都被设为 **true**，所以 **metrics** 角色使用 **firewall** 和 **selinux** 角色来管理 **metrics** 角色使用的端口。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook :

```
$ ansible-playbook ~/playbook.yml
```

验证

- 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。

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.metrics/README.md` 文件
- `/usr/share/doc/rhel-system-roles/metrics/` 目录

16.5. 使用 METRICS RHEL 系统角色，在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** 身份验证机制自动化设置身份验证的步骤。这个流程描述了如何使用 **metrics** RHEL 系统角色设置身份验证。

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。

流程

1. 编辑一个现有的 playbook 文件，如 `~/playbook.yml`，并添加与身份验证相关的变量：

```
---
- name: Set up authentication by using the scram-sha-256 authentication mechanism
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.metrics
  vars:
    metrics_retention_days: 0
    metrics_manage_firewall: true
    metrics_manage_selinux: true
    metrics_username: <username>
    metrics_password: <password>
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 验证 **sasl** 配置：

```
# pminfo -f -h "pcp://managed-node-01.example.com?username=<username>"
disk.dev.read
Password: <password>
disk.dev.read
inst [0 or "sda"] value 19540
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.metrics/README.md` 文件
- `/usr/share/doc/rhel-system-roles/metrics/` 目录

16.6. 使用 METRICS RHEL 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，通过本地系统上的 **pcp**，来为 Microsoft SQL Server 自动化配置，并启用指标集合。

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。
- 您已为 Red Hat Enterprise Linux 安装了 Microsoft SQL Server，并建立了一条到 SQL 服务器的可信连接。
- 您已为 Red Hat Enterprise Linux 的 SQL Server 安装了 Microsoft ODBC 驱动程序。
- **localhost** 在控制节点上的清单文件中被配置：

```
localhost ansible_connection=local
```

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Configure and enable metrics collection for Microsoft SQL Server
  hosts: localhost
  roles:
    - rhel-system-roles.metrics
  vars:
    metrics_from_mssql: true
    metrics_manage_firewall: true
    metrics_manage_selinux: true
```

因为 `metrics_manage_firewall` 和 `metrics_manage_selinux` 都被设为 `true`，所以 `metrics` 角色使用 `firewall` 和 `selinux` 角色来管理 `metrics` 角色使用的端口。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 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 sqlserver.example.com 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/sqlserver.example.com/20200326.16.31
pmie: primary engine: /var/log/pcp/pmie/sqlserver.example.com/pmie.log
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.metrics/README.md` 文件
- `/usr/share/doc/rhel-system-roles/metrics/` 目录

第 17 章 使用 RHEL 系统角色配置 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。角色使用推荐的设置简化和自动化了 Red Hat Enterprise Linux 主机的配置，以运行 SQL Server 工作负载。

17.1. 使用带有现有证书文件的 MICROSOFT.SQL.SERVER 系统角色安装和配置 SQL 服务器

您可以使用 **microsoft.sql.server** Ansible 角色来安装和配置 SQL Server 版本 2019。本例中的 playbook 还将服务器配置为使用现有的 **sql_cert** 证书和私钥文件，以进行 TLS 加密。

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。
- 最小 2 GB RAM
- **ansible-collection-microsoft-sql** 软件包已安装在受管节点上。
- 受管节点使用以下版本之一：RHEL 7.9、RHEL 8、RHEL 9.4 或更高版本。

流程

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

```
---
- name: Install and configure SQL Server
  hosts: managed-node-01.example.com
  roles:
    - microsoft.sql.server
  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_version: 2019
    mssql_manage_firewall: true
    mssql_tls_enable: true
    mssql_tls_cert: sql_cert.pem
    mssql_tls_private_key: sql_cert.key
    mssql_tls_version: 1.2
    mssql_tls_force: false
    mssql_password: <password>
    mssql_edition: Developer
    mssql_tcp_port: 1433
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook :

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/microsoft.sql-server/README.md` 文件

17.2. 使用 MICROSOFT.SQL.SERVER 系统角色和 CERTIFICATE RHEL 系统角色安装和配置 SQL 服务器

您可以使用 **microsoft.sql.server** Ansible 角色来安装和配置 SQL Server 版本 2019。本例中的 playbook 还将服务器配置为使用 TLS 加密，并使用 **certificate** 系统角色创建一个自签名的 **sql_cert** 证书文件和私钥。

您不必在 playbook 中调用 **certificate** 系统角色来创建证书。**microsoft.sql.server** Ansible 角色会自动调用它。

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。
- 最小 2 GB RAM
- **ansible-collection-microsoft-sql** 软件包已安装在受管节点上。
- 受管节点已在 Red Hat Identity Management (IdM)域中注册。
- 受管节点使用以下版本之一：RHEL 7.9、RHEL 8、RHEL 9.4 或更高版本。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Install and configure SQL Server
  hosts: managed-node-01.example.com
  roles:
    - microsoft.sql.server
  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_version: 2019
    mssql_manage_firewall: true
    mssql_tls_enable: true
    mssql_tls_certificates:
      - name: sql_cert
```

```

dns: *.example.com
ca: self-sign
mssql_password: <password>
mssql_edition: Developer
mssql_tcp_port: 1433

```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- [/usr/share/ansible/roles/microsoft.sql-server/README.md](#) 文件
- [使用 RHEL 系统角色请求证书](#)

17.3. 为数据和日志设置自定义存储路径

要将数据或日志存储在与默认目录不同的目录中，请使用

mssql_datadir、**mssql_datadir_mode**、**mssql_logdir** 和 **mssql_logdir_mode** 变量在现有 playbook 中指定自定义存储路径。当您定义自定义路径时，角色创建提供的目录，并确保其有正确权限和所有权。



重要

如果您稍后决定删除变量，则存储路径不会改回到默认路径，而是在最新定义的路径中保存数据。

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。
- 最小 2 GB RAM
- **ansible-collection-microsoft-sql** 软件包已安装在受管节点上。
- 受管节点使用以下版本之一：RHEL 7.9、RHEL 8、RHEL 9.4 或更高版本。

流程

1. 编辑一个现有的 playbook 文件，如 **~/playbook.yml**，并添加存储和日志相关的变量：

```

---
- name: Install and configure SQL Server
  hosts: managed-node-01.example.com

```

```

roles:
  - microsoft.sql.server
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_version: 2019
  mssql_manage_firewall: true
  mssql_tls_enable: true
  mssql_tls_cert: sql_cert.pem
  mssql_tls_private_key: sql_cert.key
  mssql_tls_version: 1.2
  mssql_tls_force: false
  mssql_password: <password>
  mssql_edition: Developer
  mssql_tcp_port: 1433
  mssql_datadir: /var/lib/mssql/
  mssql_datadir_mode: '0700'
  mssql_logdir: /var/log/mssql/
  mssql_logdir_mode: '0700'

```

在单引号中输入权限模式，以便 Ansible 将其作为字符串解析，而不是作为八进制数解析。

如果没有指定模式，且目标目录不存在，则角色在设置模式时使用系统上的默认 umask。如果没有指定模式，且目标目录存在，则角色使用现有目录的模式。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/microsoft.sql-server/README.md` 文件

第 18 章 使用 RHEL 系统角色配置 NBDE

18.1. NBDE_CLIENT 和 NBDE_SERVER RHEL 系统角色简介(CLEVIS 和 TANG)

RHEL 系统角色是 Ansible 角色和模块的集合，其提供了一个一致的配置接口来远程管理多个 RHEL 系统。

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 系统角色使您能够以自动化的方式部署多个 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。当卷最初是使用时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您可以有多个绑定。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，来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

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.nbde_server/README.md` 文件
- `/usr/share/ansible/roles/rhel-system-roles.nbde_client/README.md` 文件
- `/usr/share/doc/rhel-system-roles/nbde_server/` 目录
- `/usr/share/doc/rhel-system-roles/nbde_client/` 目录

18.2. 使用 NBDE_SERVER RHEL 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

- 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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.nbde_server
  vars:
    nbde_server_rotate_keys: yes
    nbde_server_manage_firewall: true
    nbde_server_manage_selinux: true
```

此示例 playbook 确保部署 Tang 服务器和密钥轮转。

当 `nbde_server_manage_firewall` 和 `nbde_server_manage_selinux` 都被设置为 `true` 时，`nbde_server` 角色使用 `firewall` 和 `selinux` 角色来管理 `nbde_server` 角色使用的端口。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 要通过使用安装了 Clevis 的系统上的 `grubby` 工具来确保 Tang pin 的网络在早期引导过程中可用，请输入：

```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.nbde_server/README.md` 文件
- `/usr/share/doc/rhel-system-roles/nbde_server/` 目录

18.3. 使用 NBDE_CLIENT RHEL 系统角色设置多个 CLEVIS 客户端

使用 `nbde_client` RHEL 系统角色，您可以在多个系统上准备并应用包含 Clevis 客户端设置的 Ansible playbook。



注意

`nbde_client` 系统角色只支持 Tang 绑定。因此，您无法将其用于 TPM2 绑定。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.nbde_client
  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
```

这个示例 playbook 配置了 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 加密的卷

nbde_client 系统角色只支持使用动态主机配置协议(DHCP)的情况。要对具有静态 IP 配置的客户端使用 NBDE，请使用以下 playbook：

```
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.nbde_client
  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
  tasks:
    - name: Configure a client with a static IP address during early boot
      ansible.builtin.command:
        cmd: grubby --update-kernel=ALL --args='GRUB_CMDLINE_LINUX_DEFAULT="ip={{
          <ansible_default_ipv4.address> }}:{{ <ansible_default_ipv4.gateway> }}:{{
          <ansible_default_ipv4.netmask> }}:{{ <ansible_default_ipv4.alias> }}:none"'
```

在这个 playbook 中，将 `<ansible_default_ipv4.*>` 字符串替换为您网络的 IP 地址，例如：`ip={{ 192.0.2.10 }}:{{ 192.0.2.1 }}:{{ 255.255.255.0 }}:{{ ens3 }}:none`。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.nbde_client/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/nbde_client/](#) 目录
- [期待初始 ramdisk \(initrd\)中的 Linux 网络配置](#) 文章

第 19 章 使用 RHEL 系统角色配置网络设置

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

19.1. 使用 **NETWORK** RHEL 系统角色和接口名称，配置具有静态 IP 地址的以太网连接

您可以使用 **network** RHEL 系统角色远程配置一个以太网连接。

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。
- 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. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml**：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with static IP
      ansible.builtin.include_role:
        name: rhel-system-roles.network
      vars:
        network_connections:
          - name: enp1s0
            interface_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
            state: up
```

这些设置使用以下设置为 **enp1s0** 设备定义一个以太网连接配置文件：

- 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**

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件
- `/usr/share/doc/rhel-system-roles/network/` 目录

19.2. 使用 NETWORK RHEL 系统角色和设备路径，配置具有静态 IP 地址的以太网连接

您可以使用 **network** RHEL 系统角色远程配置一个以太网连接。

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=
```

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。
- 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. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
```

```

- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with static IP
      ansible.builtin.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

```

这些设置使用以下设置定义以太网连接配置文件：

- 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**

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. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件
- `/usr/share/doc/rhel-system-roles/network/` 目录

19.3. 使用 NETWORK RHEL 系统角色和接口名称，配置具有动态 IP 地址的以太网连接

您可以使用 **network** RHEL 系统角色远程配置一个以太网连接。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。

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。
- 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. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with dynamic IP
      ansible.builtin.include_role:
        name: rhel-system-roles.network
      vars:
        network_connections:
          - name: enp1s0
            interface_name: enp1s0
            type: ethernet
            autoconnect: yes
            ip:
              dhcp4: yes
              auto6: yes
            state: up
```

这些设置为 **enp1s0** 设备定义了一个以太网连接配置文件。连接从 DHCP 服务器检索 IPv4 地址、IPv6 地址、默认网关、路由、DNS 服务器和搜索域,以及 IPv6 无状态地址自动配置 (SLAAC)。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件
- `/usr/share/doc/rhel-system-roles/network/` 目录

19.4. 使用 NETWORK RHEL 系统角色和设备路径，配置具有动态 IP 地址的以太网连接

您可以使用 **network** RHEL 系统角色远程配置一个以太网连接。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=
```

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。
- 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. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with dynamic IP
      ansible.builtin.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

```

这些设置定义一个以太网连接配置文件。连接从 DHCP 服务器检索 IPv4 地址、IPv6 地址、默认网关、路由、DNS 服务器和搜索域,以及 IPv6 无状态地址自动配置(SLAAC)。

match 参数定义 Ansible 将 play 应用到与 PCI ID **0000:00:0[1-3].0** , 而不是 **0000:00:02.0** 匹配的设备。

2. 验证 playbook 语法 :

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意, 这个命令只验证语法, 不会防止错误但有效的配置。

3. 运行 playbook :

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件
- `/usr/share/doc/rhel-system-roles/network/` 目录

19.5. 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。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure a VLAN that uses an Ethernet connection
      ansible.builtin.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
```

这些设置定义在 **enp1s0** 设备之上操作的 VLAN。VLAN 接口有以下设置：

- 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**

- VLAN ID - **10**
VLAN 配置文件中的 **parent** 属性将 VLAN 配置为在 **enp1s0** 设备之上运行。作为子设备，VLAN 连接包含 IP、默认网关和 DNS 配置。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件
- `/usr/share/doc/rhel-system-roles/network/` 目录

19.6. 使用 NETWORK RHEL 系统角色配置网桥

您可以使用 **network** RHEL 系统角色远程配置网桥。

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。
- 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. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure a network bridge that uses two Ethernet ports
      ansible.builtin.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

```

这些设置使用以下设置定义一个网桥：

- 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**
- 网桥的端口 - **enp7s0** 和 **enp8s0**



注意

在网桥上设置 IP 配置,而不是在 Linux 网桥的端口上设置。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

■

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件
- `/usr/share/doc/rhel-system-roles/network/` 目录

19.7. 使用 NETWORK RHEL 系统角色配置网络绑定

您可以使用 **network** RHEL 系统角色远程配置网络绑定。

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。
- 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. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure a network bond that uses two Ethernet ports
      ansible.builtin.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

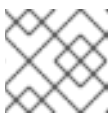
          # Add an Ethernet profile to the bond
```

```
- 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
```

这些设置使用以下设置定义网络绑定：

- 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**
- 绑定的端口 - **enp7s0** 和 **enp8s0**
- 绑定模式 - **active-backup**



注意

在绑定上设置 IP 配置，而不是在 Linux 绑定的端口上设置。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件
- `/usr/share/doc/rhel-system-roles/network/` 目录

19.8. 使用 NETWORK RHEL 系统角色配置 IPOIB 连接

您可以使用 **network** 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，使用以下设置为 **mlx4_ib0** 接口远程添加一个 InfiniBand 连接：

- 一个 IPoIB 设备 - **mlx4_ib0.8002**
- 一个分区密钥 **p_key** - **0x8002**
- 一个静态 **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。](#)
- 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。
- 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. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml**：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure IPoIB
      ansible.builtin.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. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.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

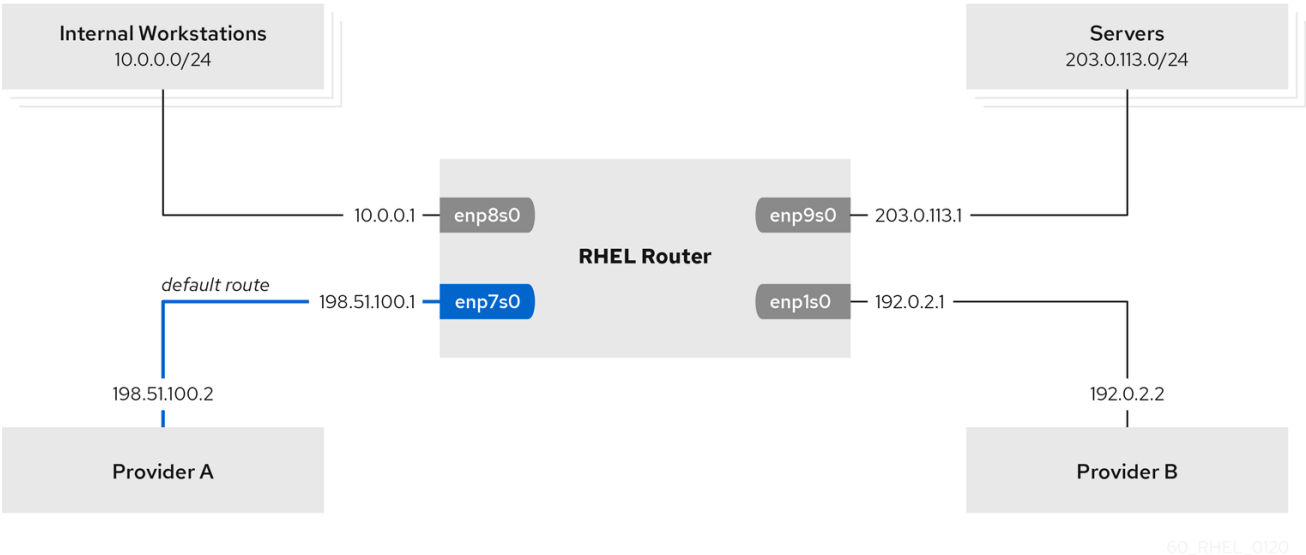
- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件
- **/usr/share/doc/rhel-system-roles/network/** 目录

19.9. 使用 NETWORK RHEL 系统角色将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，使用默认路由将所有流量路由到互联网提供商 A。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 **network** RHEL 系统角色。

此流程假设以下网络拓扑：



60_RHEL_0120

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。
- 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. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.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
      ansible.builtin.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. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

1. 在内部工作站子网的 RHEL 主机上：

- a. 安装 **traceroute** 软件包：

```
# dnf 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：

```
# dnf 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
- `/usr/share/doc/rhel-system-roles/network/` 目录

19.10. 使用 NETWORK RHEL 系统角色配置一个具有 802.1X 网络身份验证的静态以太网连接

您可以使用 **network** RHEL 系统角色远程配置一个具有 802.1X 网络身份验证的以太网连接。

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。
- 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. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.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
      ansible.builtin.copy:
        src: "/srv/data/client.key"
        dest: "/etc/pki/tls/private/client.key"
        mode: 0600

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

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

    - name: Configure connection
      ansible.builtin.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
```

这些设置使用以下设置为 **enp1s0** 设备定义一个以太网连接配置文件：

- 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**
- 使用 **TLS** 可扩展身份验证协议(EAP)的 802.1X 网络身份验证。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

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

19.11. 使用 NETWORK RHEL 系统角色配置一个具有 802.1X 网络身份验证的 WIFI 连接

使用 RHEL 系统角色，您可以自动化一个 wifi 连接的创建。例如，您可以使用 Ansible Playbook，远程为 **wlp1s0** 接口添加无线连接配置文件。创建的配置集使用 802.1X 标准将客户端验证到 Wi-Fi 网络。该 playbook 将连接配置集配置为使用 DHCP。要配置静态 IP 设置，相应地调整 **ip** 字典中的参数。

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。

- 网络支持 802.1X 网络身份验证。
- 您已在受管节点上安装了 **wpa_supplicant** 软件包。
- DHCP 位于受管节点的网络中。
- 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。
 - CA 证书存储在 **/srv/data/ca.crt** 文件中。

流程

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

```
---
- name: Configure a wifi connection with 802.1X authentication
  hosts: managed-node-01.example.com
  tasks:
    - name: Copy client key for 802.1X authentication
      ansible.builtin.copy:
        src: "/srv/data/client.key"
        dest: "/etc/pki/tls/private/client.key"
        mode: 0400

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

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

    - block:
      - ansible.builtin.import_role:
          name: rhel-system-roles.network
        vars:
          network_connections:
            - name: Configure the Example-wifi profile
              interface_name: wlp1s0
              state: up
              type: wireless
              autoconnect: yes
              ip:
                dhcp4: true
                auto6: true
              wireless:
                ssid: "Example-wifi"
                key_mgmt: "wpa-eap"
              ieee802_1x:
                identity: "user_name"
```

```
eap: tls
private_key: "/etc/pki/tls/client.key"
private_key_password: "password"
private_key_password_flags: none
client_cert: "/etc/pki/tls/client.pem"
ca_cert: "/etc/pki/tls/cacert.pem"
domain_suffix_match: "example.com"
```

这些设置为 **wlp1s0** 接口定义一个 wifi 连接配置文件。该配置文件使用 802.1X 标准向 wifi 网络验证客户端。连接从 DHCP 服务器检索 IPv4 地址、IPv6 地址、默认网关、路由、DNS 服务器和搜索域,以及 IPv6 无状态地址自动配置(SLAAC)。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

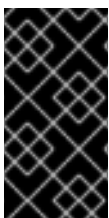
```
$ ansible-playbook ~/playbook.yml
```

其他资源

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

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

在大多数情况下，管理员在创建连接时设置默认网关。但是，您也可以使用 **network** RHEL 系统角色，在之前创建的连接上设置或更新默认网关设置，以设置默认网关。



重要

当您运行一个使用 **network** RHEL 系统角色的 play 时，如果设置值与 play 中指定的值不匹配，则角色会使用同样的名称覆盖现有的连接配置文件。要防止将这些值重置为其默认值，请始终在 play 中指定网络连接配置文件的整个配置，即使配置（如 IP 配置）已存在。

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::ffff**
- IPv4 DNS 服务器 - **198.51.100.200**
- IPv6 DNS 服务器 - **2001:db8:1::ffbb**

- DNS 搜索域 - **example.com**

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。

流程

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

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with static IP and default gateway
      ansible.builtin.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. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

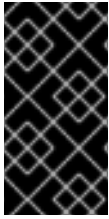
```
$ ansible-playbook ~/playbook.yml
```

其他资源

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

19.13. 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

当您运行一个使用 **network** RHEL 系统角色的 play 时，如果设置值与 play 中指定的值不匹配，则角色会使用同样的名称覆盖现有的连接配置文件。要防止将这些值重置为其默认值，请始终在 play 中指定网络连接配置文件的整个配置，即使配置（如 IP 配置）已存在。

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。

流程

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

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

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**

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.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：

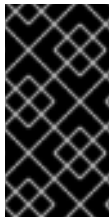
```
# ip -6 route
...
2001:db8:2::/64 via 2001:db8:1::10 dev enp7s0 metric 1024 pref medium
```

其他资源

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

19.14. 使用 NETWORK RHEL 系统角色配置一个 ETHTOOL 卸载功能

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



重要

当您运行一个使用 **network** RHEL 系统角色的 play 时，如果设置值与 play 中指定的值不匹配，则角色会使用同样的名称覆盖现有的连接配置文件。要防止将这些值重置为其默认值，请始终在 play 中指定网络连接配置文件的整个配置，即使配置（如 IP 配置）已存在。

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。

流程

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

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with ethtool features
      ansible.builtin.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
```

此 playbook 创建具有以下设置的 **enp1s0** 连接配置文件，或者在配置文件已存在时更新它：

- 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

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

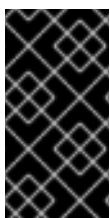
```
$ ansible-playbook ~/playbook.yml
```

其他资源

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

19.15. 使用 NETWORK RHEL 系统角色配置 ETHTOOL 合并设置

您可以使用 **network** RHEL 系统角色配置 NetworkManager 连接的 **ethtool** 合并设置。



重要

当您运行一个使用 **network** RHEL 系统角色的 play 时，如果设置值与 play 中指定的值不匹配，则角色会使用同样的名称覆盖现有的连接配置文件。要防止将这些值重置为其默认值，请始终在 play 中指定网络连接配置文件的整个配置，即使配置（如 IP 配置）已存在。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with ethtool coalesce settings
      ansible.builtin.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:
              coalesce:
                rx_frames: 128
                tx_frames: 128
            state: up
```

此 playbook 创建具有以下设置的 **enp1s0** 连接配置文件，或者在配置文件已存在时更新它：

- 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**
- **ethtool** coalesce 设置：
 - RX 帧：**128**
 - TX 帧：**128**

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件
- `/usr/share/doc/rhel-system-roles/network/` 目录

19.16. 使用 NETWORK RHEL 系统角色增加环缓冲区的大小，以减少高数据包丢弃率

如果数据包丢弃率导致应用程序报告数据丢失、超时或其他问题，请增加以太网设备的环缓冲区的大小。

环缓冲区是循环缓冲区，溢出会覆盖现有数据。网卡分配一个传输(TX)和接收(RX)环缓冲区。接收环缓冲区在设备驱动程序和网络接口控制器(NIC)之间共享。数据可以通过硬件中断或软件中断（也称为 SoftIRQ）从 NIC 移到内核。

内核使用 RX 环缓冲区存储传入的数据包，直到设备驱动程序可以处理它们。设备驱动程序排空 RX 环，通常是使用 SoftIRQ，其将传入的数据包放在名为 `sk_buff` 或 `skb` 的内核数据结构中，以通过内核开始其过程，直到拥有相关套接字的应用程序。

内核使用 TX 环缓冲区来存放应发送到网络的传出数据包。这些环缓冲区位于堆栈的底部，是可能发生数据包丢弃的关键点，这反过来会对网络性能造成负面影响。



重要

当您运行一个使用 **network** RHEL 系统角色的 play 时，如果设置值与 play 中指定的值不匹配，则角色会使用同样的名称覆盖现有的连接配置文件。要防止将这些值重置为其默认值，请始终在 play 中指定网络连接配置文件的整个配置，即使配置（如 IP 配置）已存在。

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。
- 您知道设备支持的最大环缓冲区大小。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
```

```

- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure an Ethernet connection with increased ring buffer sizes
      ansible.builtin.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:
              ring:
                rx: 4096
                tx: 4096
            state: up

```

此 playbook 创建具有以下设置的 **enp1s0** 连接配置文件，或者在配置文件已存在时更新它：

- 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**
- 环缓冲区条目的最大数量：
 - 接收(RX)：4096
 - 传输(TX)：4096

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

\$ ansible-playbook ~/playbook.yml

其他资源

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

19.17. NETWORK 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
<pre>vars: network_state: interfaces: - name: enp7s0 type: ethernet state: up ipv4: enabled: true auto-dns: true auto-gateway: true auto-routes: true dhcp: true ipv6: enabled: true auto-dns: true auto-gateway: true auto-routes: true autoconf: true dhcp: true</pre>	<pre>vars: network_connections: - name: enp7s0 interface_name: enp7s0 type: ethernet autoconnect: yes ip: dhcp4: yes auto6: yes state: up</pre>

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:
  network_state:
    interfaces:
      - name: enp7s0
        type: ethernet
        state: down
```

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

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件
- `/usr/share/doc/rhel-system-roles/network/` 目录

第 20 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、容器以及运行 Podman 容器的 **systemd** 服务。

20.1. 创建一个带有绑定挂载的无根容器

您可以通过运行 Ansible playbook，使用 **podman** RHEL 系统角色创建带有绑定挂载的无根容器，并使用它来管理应用程序配置

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。

流程

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

```
- hosts: managed-node-01.example.com
vars:
  podman_create_host_directories: true
  podman_firewall:
    - port: 8080-8081/tcp
      state: enabled
    - port: 12340/tcp
      state: enabled
  podman_selinux_ports:
    - ports: 8080-8081
      setype: http_port_t
  podman_kube_specs:
    - state: started
      run_as_user: dbuser
      run_as_group: dbgroup
      kube_file_content:
        apiVersion: v1
        kind: Pod
        metadata:
          name: db
        spec:
          containers:
            - name: db
              image: quay.io/db/db:stable
              ports:
                - containerPort: 1234
                  hostPort: 12340
              volumeMounts:
                - mountPath: /var/lib/db:Z
                  name: db
          volumes:
            - name: db
              hostPath:
```

```

        path: /var/lib/db
    - state: started
      run_as_user: webapp
      run_as_group: webapp
      kube_file_src: /path/to/webapp.yml
  roles:
    - linux-system-roles.podman

```

此流程创建一个有两个容器的 pod。**podman_kube_specs** 角色变量描述了 pod。

- **run_as_user** 和 **run_as_group** 字段指定容器是无根的。
- 包含 Kubernetes YAML 文件的 **kube_file_content** 字段定义了名为 **db** 的第一个容器。您可以使用 **podman kube generate** 命令生成 Kubernetes YAML 文件。
 - **db** 容器是基于 **quay.io/db/db:stable** 容器镜像。
 - **db** 绑定挂载将主机上的 **/var/lib/db** 目录映射到容器中的 **/var/lib/db** 目录。**Z** 标志使用私有的 **unshared** 标签标记内容，因此只有 **db** 容器才能访问内容。
- **kube_file_src** 字段定义第二个容器。控制器节点上 **/path/to/webapp.yml** 文件的内容将复制到受管节点上的 **kube_file** 字段中。
- 设置 **podman_create_host_directories: true** 以在主机上创建目录。这指示角色检查 **hostPath** 卷的 kube 规格，并在主机上创建这些目录。如果您需要对所有权和权限有更多的控制，请使用 **podman_host_directories**。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check --ask-vault-pass ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook --ask-vault-pass ~/playbook.yml
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.podman/README.md** 文件
- **/usr/share/doc/rhel-system-roles/podman/** 目录

20.2. 使用 PODMAN 卷创建有根容器

您可以通过运行 Ansible playbook，使用 **podman** RHEL 系统角色创建带有 Podman 卷的有根容器，并使用它来管理应用程序配置。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
- hosts: managed-node-01.example.com
  vars:
    podman_firewall:
      - port: 8080/tcp
        state: enabled
    podman_kube_specs:
      - state: started
        kube_file_content:
          apiVersion: v1
          kind: Pod
          metadata:
            name: ubi8-httpd
          spec:
            containers:
              - name: ubi8-httpd
                image: registry.access.redhat.com/ubi8/httpd-24
                ports:
                  - containerPort: 8080
                    hostPort: 8080
                volumeMounts:
                  - mountPath: /var/www/html:Z
                    name: ubi8-html
            volumes:
              - name: ubi8-html
                persistentVolumeClaim:
                  claimName: ubi8-html-volume
  roles:
    - linux-system-roles.podman
```

此流程创建一个具有一个容器的 pod。`podman_kube_specs` 角色变量描述了 pod。

- 默认情况下，`podman` 角色创建有根容器。
- 包含 Kubernetes YAML 文件的 `kube_file_content` 字段定义了名为 `ubi8-httpd` 的容器。
 - `ubi8-httpd` 容器是基于 `registry.access.redhat.com/ubi8/httpd-24` 容器镜像。
 - `ubi8-html-volume` 将主机上的 `/var/www/html` 目录映射到容器。`Z` 标志使用私有 `unshared` 标签标记内容，因此只有 `ubi8-httpd` 容器可以访问内容。
 - pod 使用挂载路径 `/var/www/html` 挂载名为 `ubi8-html-volume` 的现有持久性卷。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.podman/README.md` 文件
- `/usr/share/doc/rhel-system-roles/podman/` 目录

20.3. 使用 SECRET 创建一个 QUADLET 应用程序

您可以通过运行 Ansible playbook，使用 **podman** RHEL 系统角色创建带有 secret 的 Quadlet 应用程序。

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。
- 容器中的 web 服务器应使用的证书和对应的私钥存储在 `~/certificate.pem` 和 `~/key.pem` 文件中。

流程

1. 显示证书和私钥文件的内容：

```
$ cat ~/certificate.pem
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----

$ cat ~/key.pem
-----BEGIN PRIVATE KEY-----
...
-----END PRIVATE KEY-----
```

在后续步骤中需要此信息。

2. 将您的敏感变量存储在一个加密文件中：

- a. 创建 vault：

```
$ ansible-vault create vault.yml
New Vault password: <vault_password>
Confirm New Vault password: <vault_password>
```

- b. 在 **ansible-vault create** 命令打开编辑器后，以 **<key>: <value>** 格式输入敏感数据：

```
root_password: <root_password>
certificate: |-
  -----BEGIN CERTIFICATE-----
  ...
  -----END CERTIFICATE-----
key: |-
```

```
-----BEGIN PRIVATE KEY-----
...
-----END PRIVATE KEY-----
```

确保 **certificate** 和 **key** 变量中的所有行都以两个空格开头。

c. 保存更改，并关闭编辑器。Ansible 加密 vault 中的数据。

3. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml**：

```
- name: Deploy a wordpress CMS with MySQL database
  hosts: managed-node-01.example.com
  vars_files:
    - vault.yml
  tasks:
    - name: Create and run the container
      ansible.builtin.include_role:
        name: rhel-system-roles.podman
      vars:
        podman_create_host_directories: true
        podman_activate_systemd_unit: false
        podman_quadlet_specs:
          - name: quadlet-demo
            type: network
            file_content: |
              [Network]
              Subnet=192.168.30.0/24
              Gateway=192.168.30.1
              Label=app=wordpress
          - file_src: quadlet-demo-mysql.volume
          - template_src: quadlet-demo-mysql.container.j2
          - file_src: envoy-proxy-configmap.yml
          - file_src: quadlet-demo.yml
          - file_src: quadlet-demo.kube
            activate_systemd_unit: true
        podman_firewall:
          - port: 8000/tcp
            state: enabled
          - port: 9000/tcp
            state: enabled
        podman_secrets:
          - name: mysql-root-password-container
            state: present
            skip_existing: true
            data: "{{ root_password }}"
          - name: mysql-root-password-kube
            state: present
            skip_existing: true
            data: |
              apiVersion: v1
              data:
                password: "{{ root_password | b64encode }}"
                kind: Secret
              metadata:
                name: mysql-root-password-kube
          - name: envoy-certificates
```

```

state: present
skip_existing: true
data: |
  apiVersion: v1
  data:
    certificate.key: {{ key | b64encode }}
    certificate.pem: {{ certificate | b64encode }}
  kind: Secret
  metadata:
    name: envoy-certificates

```

流程创建一个与 MySQL 数据库配对的 WordPress 内容管理系统。**podman_quadlet_specs** 角色变量为 Quadlet 定义一组配置，它指向以某种方式一起工作的一组容器或服务。它包括以下规范：

- Wordpress 网络由 **quadlet-demo** 网络单元定义。
- MySQL 容器的卷配置由 **file_src: quadlet-demo-mysql.volume** 字段定义。
- **template_src: quadlet-demo-mysql.container.j2** 字段用于为 MySQL 容器生成一个配置。
- 两个 YAML 文件如下：**file_src: envoy-proxy-configmap.yml** 和 **file_src: quadlet-demo.yml**。请注意，.yml 不是一个有效的 Quadlet 单元类型，因此这些文件将只被复制，且不会作为 Quadlet 规范来处理。
- Wordpress 和 envoy 代理容器和配置由 **file_src: quadlet-demo.kube** 字段定义。kube 单元将之前 **[Kube]** 部分中的 YAML 文件称为 **Yaml=quadlet-demo.yml** 和 **ConfigMap=envoy-proxy-configmap.yml**。

4. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check --ask-vault-pass ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

5. 运行 playbook：

```
$ ansible-playbook --ask-vault-pass ~/playbook.yml
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.podman/README.md** 文件
- **/usr/share/doc/rhel-system-roles/podman/** 目录

第 21 章 使用 RHEL 系统角色配置 POSTFIX MTA

使用 **postfix** RHEL 系统角色，您可以统一简化 Postfix 服务的自动配置，带有模块化设计的与 Sendmail 兼容的邮件传输代理(MTA)，以及各种配置选项。**rhel-system-roles** 软件包包含此 RHEL 系统角色，以及参考文档。

21.1. 使用 POSTFIX RHEL 系统角色自动化基本 POSTFIX MTA 管理

您可以使用 **postfix** RHEL 系统角色在受管节点上安装、配置和启动 Postfix 邮件传输代理。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Manage postfix
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.postfix
  vars:
    postfix_conf:
      relay_domains: $mydestination
      relayhost: example.com
```

- 如果您希望 Postfix 使用与 **gethostname** () 函数返回的完全限定域名(FQDN)不同的主机名，请在文件中的 **postfix_conf**: 行下添加 **myhostname** 参数：

```
myhostname = smtp.example.com
```

- 如果域名与 **myhostname** 参数中的域名不同，请添加 **mydomain** 参数。否则，会使用 **\$myhostname** 减去第一个组件。

```
mydomain = <example.com>
```

- 使用 **postfix_manage_firewall: true** 变量来确保 SMTP 端口在服务器的防火墙中打开。管理 SMTP 相关的端口 **25/tcp**、**465/tcp** 和 **587/tcp**。如果变量设为 **false**，则 **postfix** 角色不管理防火墙。默认值为 **false**。



注意

postfix_manage_firewall 变量仅限于添加端口。它不能用来删除端口。如果要删除端口，请直接使用 **firewall** RHEL 系统角色。

- 如果您的场景涉及使用非标准端口，请设置 **postfix_manage_selinux: true** 变量，以确保端口在服务器上为 SELinux 正确标记。



注意

postfix_manage_selinux 变量仅限于向 SELinux 策略中添加规则。它不能从策略中删除规则。如果要删除规则，请直接使用 **selinux** RHEL 系统角色。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.postfix/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/postfix/](#) 目录

第 22 章 使用 RHEL 系统角色安装和配置 POSTGRESQL

作为系统管理员，您可以使用 **postgresql** RHEL 系统角色安装、配置、管理、启动和提高 PostgreSQL 服务器的性能。

22.1. POSTGRESQL RHEL 系统角色简介

要使用 Ansible 安装、配置、管理和启动 PostgreSQL 服务器，您可以使用 **postgresql** RHEL 系统角色。

您还可以使用 **postgresql** 角色来优化数据库服务器设置并提高性能。

角色支持 RHEL 8 和 RHEL 9 受管节点上当前发布的和支持的 PostgreSQL 版本。

22.2. 使用 POSTGRESQL RHEL 系统角色配置 POSTGRESQL 服务器

您可以使用 **postgresql** RHEL 系统角色安装、配置、管理和启动 PostgreSQL 服务器。



警告

postgresql 角色替换了受管主机上 `/var/lib/pgsql/data/` 目录中的 PostgreSQL 配置文件。之前的设置变为角色变量中指定的变量，如果它们没有在角色变量中指定，则会丢失。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Manage PostgreSQL
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.postgresql
  vars:
    postgresql_version: "13"
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook :

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.postgresql/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/postgresql/](#) 目录
- [使用 PostgreSQL](#)

第 23 章 使用 RHEL 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。

23.1. RHC RHEL 系统角色简介

RHEL 系统角色是一组角色，其提供一致的配置接口来远程管理多个系统。远程主机配置(**rhc**) RHEL 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** RHEL 系统角色注册系统时，系统连接到 Insights。另外，使用 **rhc** RHEL 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

23.2. 使用 RHC RHEL 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 系统角色在注册时将系统连接到 Red Hat Insights。

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。

流程

1. 将您的敏感变量存储在一个加密文件中：

- a. 创建 vault：

```
$ ansible-vault create vault.yml
New Vault password: <password>
Confirm New Vault password: <vault_password>
```

- b. 在 **ansible-vault create** 命令打开编辑器后，以 **<key>: <value>** 格式输入敏感数据：

```
activationKey: <activation_key>
username: <username>
password: <password>
```

- c. 保存更改，并关闭编辑器。Ansible 加密 vault 中的数据。

2. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

- 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:
    - vault.yml
  roles:
    - role: rhel-system-roles.rhc
  vars:
    rhc_auth:
      activation_keys:
        keys:
          - "{{ activationKey }}"
    rhc_organization: organizationID
```

- 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:
    - vault.yml
  vars:
    rhc_auth:
      login:
        username: "{{ username }}"
        password: "{{ password }}"
  roles:
    - role: rhel-system-roles.rhc
```

3. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check --ask-vault-pass ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

4. 运行 playbook：

```
$ ansible-playbook --ask-vault-pass ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.rhc/README.md` 文件
- `/usr/share/doc/rhel-system-roles/rhc/` 目录
- [Ansible Vault](#)

23.3. 使用 RHC RHEL 系统角色，使用 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，使用 Satellite 远程注册您的系统。

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。

流程

1. 将您的敏感变量存储在一个加密文件中：

- a. 创建 vault：

```
$ ansible-vault create vault.yml
New Vault password: <password>
Confirm New Vault password: <vault_password>
```

- b. 在 **ansible-vault create** 命令打开编辑器后，以 **<key>: <value>** 格式输入敏感数据：

```
activationKey: <activation_key>
```

- c. 保存更改，并关闭编辑器。Ansible 加密 vault 中的数据。

2. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml**：

```
---
- name: Register to the custom registration server and CDN
  hosts: managed-node-01.example.com
  vars_files:
    - vault.yml
  roles:
    - role: rhel-system-roles.rhc
  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
```

3. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check --ask-vault-pass ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

4. 运行 playbook :

```
$ ansible-playbook --ask-vault-pass ~/playbook.yml
```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.rhc/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/rhc/](#) 目录
- [Ansible Vault](#)

23.4. 使用 RHC RHEL 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，角色默认启用到 Red Hat Insights 的连接。如果不需要，您可以使用 **rhc** RHEL 系统角色禁用它。

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。
- 您已注册系统。

流程

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

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

2. 验证 playbook 语法 :

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook :

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.rhc/README.md](#) 文件

- `/usr/share/doc/rhel-system-roles/rhc/` 目录

23.5. 使用 RHC RHEL 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

- 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。
- 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. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

- 要启用存储库：

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

- 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. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.rhc/README.md` 文件
- `/usr/share/doc/rhel-system-roles/rhc/` 目录

23.6. 使用 RHC RHEL 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

- [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。
- 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. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

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

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.rhc/README.md` 文件
- `/usr/share/doc/rhel-system-roles/rhc/` 目录

23.7. 在使用 RHC RHEL 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

- 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。

流程

1. 将您的敏感变量存储在一个加密文件中：

- a. 创建 `vault`：

```
$ ansible-vault create vault.yml
New Vault password: <password>
Confirm New Vault password: <vault_password>
```

- b. 在 **`ansible-vault create`** 命令打开编辑器后，以 **`<key>: <value>`** 格式输入敏感数据：

```
username: <username>
password: <password>
proxy_username: <proxyusername>
proxy_password: <proxypassword>
```

- c. 保存更改，并关闭编辑器。Ansible 加密 `vault` 中的数据。

2. 创建一个包含以下内容的 `playbook` 文件，如 `~/playbook.yml`：

- 要使用代理注册到红帽客户门户网站：

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

- 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:
    - vault.yml
  vars:
    rhc_auth:
      login:
        username: "{{ username }}"
        password: "{{ password }}"
    rhc_proxy: {"state":"absent"}
  roles:
    - role: rhel-system-roles.rhc
```

3. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check --ask-vault-pass ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

4. 运行 playbook：

```
$ ansible-playbook --ask-vault-pass ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.rhc/README.md` 文件
- `/usr/share/doc/rhel-system-roles/rhc/` 目录
- [Ansible Vault](#)

23.8. 使用 RHC RHEL 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

- 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。
- 您已注册系统。

流程

1. 将您的敏感变量存储在一个加密文件中：

- a. 创建 vault :

```
$ ansible-vault create vault.yml
New Vault password: <password>
Confirm New Vault password: <vault_password>
```

- b. 在 **ansible-vault create** 命令打开编辑器后, 以 **<key>: <value>** 格式输入敏感数据 :

```
username: <username>
password: <password>
```

- c. 保存更改, 并关闭编辑器。Ansible 加密 vault 中的数据。

2. 创建一个包含以下内容的 playbook 文件, 如 **~/playbook.yml** :

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

3. 验证 playbook 语法 :

```
$ ansible-playbook --syntax-check --ask-vault-pass ~/playbook.yml
```

请注意, 这个命令只验证语法, 不会防止错误但有效的配置。

4. 运行 playbook :

```
$ ansible-playbook --ask-vault-pass ~/playbook.yml
```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.rhc/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/rhc/](#) 目录
- [Ansible Vault](#)

23.9. 使用 RHC RHEL 系统角色禁用 INSIGHTS 补救

您可以使用 **rhc** RHEL 系统角色配置系统, 来自动更新动态配置。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** RHEL 系统角色启用补救，以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

- 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。
- 您已启用了 Insights 补救。
- 您已注册系统。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

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

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.rhc/README.md` 文件
- `/usr/share/doc/rhel-system-roles/rhc/` 目录

23.10. 使用 RHC RHEL 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

- 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。

流程

1. 将您的敏感变量存储在一个加密文件中：

- a. 创建 vault：

```
$ ansible-vault create vault.yml
New Vault password: <password>
Confirm New Vault password: <vault_password>
```

- b. 在 **ansible-vault create** 命令打开编辑器后，以 **<key>: <value>** 格式输入敏感数据：

```
username: <username>
password: <password>
```

- c. 保存更改，并关闭编辑器。Ansible 加密 vault 中的数据。

2. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml**：

```
---
- name: Creating tags
  hosts: managed-node-01.example.com
  vars_files:
    - vault.yml
  roles:
    - role: rhel-system-roles.rhc
  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
```

3. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check --ask-vault-pass ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

4. 运行 playbook：

```
$ ansible-playbook --ask-vault-pass ~/playbook.yml
```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.rhc/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/rhc/](#) 目录
- [系统过滤和对 Red Hat Insights 进行分组](#)。
- [Ansible Vault](#)

23.11. 使用 RHC RHEL 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

- [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。
- 系统已经注册。

流程

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

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

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.rhc/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/rhc/](#) 目录

第 24 章 使用 RHEL 系统角色配置 SELINUX

您可以使用 **selinux** RHEL 系统角色在其他系统上配置和管理 SELinux 权限。

24.1. SELINUX RHEL 系统角色简介

RHEL 系统角色是 Ansible 角色和模块的集合，其提供了一个一致的配置接口来远程管理多个 RHEL 系统。您可以使用 **selinux** RHEL 系统角色执行以下操作：

- 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。

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。

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.selinux/README.md` 文件
- `/usr/share/doc/rhel-system-roles/selinux/` 目录

24.2. 使用 SELINUX RHEL 系统角色在多个系统上应用 SELINUX 设置

使用 **selinux** RHEL 系统角色，您可以准备并应用带有验证的 SELinux 设置的 Ansible playbook。

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。

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. 验证 playbook 语法：

```
# ansible-playbook <my-selinux-playbook.yml> --syntax-check
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

5. 运行您的 playbook:

```
# ansible-playbook <my-selinux-playbook.yml>
```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.selinux/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/selinux/](#) 目录
- [使用 Ansible 进行 SELinux 强化](#) 知识库文章

24.3. 使用 SELINUX RHEL 系统角色管理端口

您可以使用 **selinux** RHEL 系统角色在多个系统间一致地自动管理 SELinux 中的端口访问。这可能很有用，例如，在将 Apache HTTP 服务器配置为侦听不同端口时。您可以通过使用 **selinux** RHEL 系统角色创建一个 playbook 来实现，其将 **http_port_t** SELinux 类型分配给特定的端口号。在受管节点上运行 playbook 后，SELinux 策略中定义的特定服务可以访问此端口。

您可以使用 **seport** 模块自动管理 SELinux 中的端口访问，这比使用整个角色更快，或者通过使用 **selinux** RHEL 系统角色，这在 SELinux 配置中进行其他更改时更有用。这些方法是等同的，实际上 **selinux** RHEL 系统角色在配置端口时使用 **seport** 模块。在受管节点上输入 **semanage port -a -t http_port_t -p tcp <port_number>** 命令时，每个方法都有同样的效果。

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。
- 可选：要使用 **semanage** 命令验证端口状态，必须安装 **polycoreutils-python-utils** 软件包。

流程

- 要在不进行其他更改的情况下只配置端口号，请使用 **seport** 模块：

```
- name: Allow Apache to listen on tcp port <port_number>
  community.general.seport:
    ports: <port_number>
    proto: tcp
    setype: http_port_t
    state: present
```

将 **<port_number>** 替换为您要为其分配 **http_port_t** 类型的端口号。

- 对于涉及其他 SELinux 自定义的受管节点的更复杂的配置，请使用 **selinux** RHEL 系统角色。创建一个 playbook 文件，如 **~/playbook.yml**，并添加以下内容：

```
---
- name: Modify SELinux port mapping example
  hosts: all
  vars:
    # Map tcp port <port_number> to the 'http_port_t' SELinux port type
    selinux_ports:
      - ports: <port_number>
        proto: tcp
        setype: http_port_t
        state: present

  tasks:
    - name: Include selinux role
      ansible.builtin.include_role:
        name: rhel-system-roles.selinux
```

将 **<port_number>** 替换为您要为其分配 **http_port_t** 类型的端口号。

验证

- 验证端口是否被分配了 **http_port_t** 类型：

```
# semanage port --list | grep http_port_t
http_port_t          tcp  <port_number>, 80, 81, 443, 488, 8008, 8009, 8443, 9000
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.selinux/README.md** 文件
- **/usr/share/doc/rhel-system-roles/selinux/** 目录

第 25 章 使用 RHEL 系统角色保护文件访问

使用 **fapolicyd** 系统角色，您可以使用 Red Hat Ansible Automation Platform 防止在 RHEL 上执行未知代码。

25.1. 使用 FAPOLICYD RHEL 系统角色配置对未知代码执行的保护

您可以使用 **fapolicyd** 系统角色，通过运行 Ansible playbook 来防止执行未知代码。

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。

流程

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

```
---
- name: Preventing execution of unknown code
  hosts: all
  vars:
    fapolicyd_setup_integrity: sha256
    fapolicyd_setup_trust: rpmdb,file
    fapolicyd_add_trusted_file:
      - </usr/bin/my-ls>
      - </opt/third-party/app1>
      - </opt/third-party/app2>
  roles:
    - rhel-system-roles.fapolicyd
```

您可以使用 **linux-system-roles.fapolicyd** RHEL 系统角色的以下变量来进一步自定义保护：

fapolicyd_setup_integrity

您可以设置以下一种完整性类型：**none**、**sha256** 和 **size**。

fapolicyd_setup_trust

您可以设置信任文件类型 **file**、**rpmd** 和 **deb**。

fapolicyd_add_trusted_file

您可以列出您信任且 **fapolicyd** 不阻止其执行的可执行文件。

2. 验证 playbook 语法：

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

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

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

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.fapolicyd/README.md` 文件

第 26 章 使用 RHEL 系统角色配置安全通信

作为管理员，您可以使用 **sshd** 系统角色配置 SSH 服务器，使用 **ssh** 系统角色，使用 Ansible Core 软件包同时在任意数量的 RHEL 系统上一致地配置 SSH 客户端。

26.1. SSHD RHEL 系统角色的变量

在 **sshd** 系统角色 playbook 中，您可以根据您的偏好和限制为 SSH 配置文件定义参数。

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 **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 ::
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles/sshd/README.md` 文件
- `/usr/share/doc/rhel-system-roles/sshd/` 目录

26.2. 使用 SSHD RHEL 系统角色配置 OPENSSH 服务器

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



注意

您可以将 **sshd** 系统角色与更改 SSH 和 SSHD 配置的其他系统角色（如身份管理 RHEL 系统角色）一起使用。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

- 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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
```

```

- name: SSH server configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure sshd to prevent root and password login except from particular subnet
      ansible.builtin.include_role:
        name: rhel-system-roles.sshd
      vars:
        sshd:
          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

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

1. 登录到 SSH 服务器：

```
$ ssh <username>@<ssh_server>
```

2. 验证 SSH 服务器上 **sshd_config** 文件的内容：

```

$ cat /etc/ssh/sshd_config.d/00-ansible_system_role.conf
#
# Ansible managed
#
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@<ssh_server>
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.sshd/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ssh/` 目录

26.3. 对非独占配置使用 sshd RHEL 系统角色

通常，应用 **sshd** 系统角色会覆盖整个配置。如果您之前已调整了配置，例如使用不同的系统角色或 playbook，这可能会有问题。要只对所选的配置选项应用 **sshd** 系统角色，同时保留其他选项，您可以使用非独占配置。

您可以应用一个非独占配置：

- 在 RHEL 8 及更早版本中，使用配置片断。
- 在 RHEL 9 及更高版本中，使用置入目录中的文件。默认配置文件已放入随时可访问的目录中，存为 `/etc/ssh/sshd_config.d/00-ansible_system_role.conf`。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

- 对于运行 RHEL 8 或更早版本的受管节点：

```
---
- name: Non-exclusive sshd configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: <Configure SSHD to accept some useful environment variables>
      ansible.builtin.include_role:
        name: rhel-system-roles.sshd
  vars:
    sshd_config_namespace: <my-application>
  sshd:
    # Environment variables to accept
    AcceptEnv:
      LANG
      LS_COLORS
      EDITOR
```


- 对于运行 RHEL 9 或更高版本的受管节点：

```
- name: Non-exclusive sshd configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: <Configure sshd to accept some useful environment variables>
      ansible.builtin.include_role:
        name: rhel-system-roles.sshd
      vars:
        sshd_config_file: /etc/ssh/sshd_config.d/<42-my-application>.conf
      sshd:
        # Environment variables to accept
        AcceptEnv:
          LANG
          LS_COLORS
          EDITOR
```

在 `sshd_config_file` 变量中，定义 `sshd` 系统角色在其中写入配置选项的 `.conf` 文件。使用两位前缀，例如 `42-` 来指定应用配置文件的顺序。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 验证 SSH 服务器上的配置：
 - 对于运行 RHEL 8 或更早版本的受管节点：

```
# cat /etc/ssh/sshd_config.d/42-my-application.conf
# Ansible managed
#
AcceptEnv LANG LS_COLORS EDITOR
```

- 对于运行 RHEL 9 或更高版本的受管节点：

```
# cat /etc/ssh/sshd_config
...
# 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>
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.sshd/README.md` 文件

- `/usr/share/doc/rhel-system-roles/sshd/` 目录

26.4. 使用 SSHD RHEL 系统角色覆盖 SSH 服务器上系统范围的加密策略

您可以使用 **sshd** RHEL 系统角色覆盖 SSH 服务器上系统范围的加密策略。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
- name: Overriding the system-wide cryptographic policy
  hosts: managed-node-01.example.com
  roles:
    - rhel_system_roles.sshd
  vars:
    sshd_sysconfig: true
    sshd_sysconfig_override_crypto_policy: true
    sshd_KexAlgorithms: ecdh-sha2-nistp521
    sshd_Ciphers: aes256-ctr
    sshd_MACs: hmac-sha2-512-etm@openssh.com
    sshd_HostKeyAlgorithms: rsa-sha2-512,rsa-sha2-256
```

- **sshd_KexAlgorithms**:: 您可以选择密钥交换算法，例如 **ecdh-sha2-nistp256**、**ecdh-sha2-nistp384**、**ecdh-sha2-nistp521**、**diffie-hellman-group14-sha1** 或 **diffie-hellman-group-exchange-sha256**。
- **sshd_Ciphers**:: 您可以选择密码，例如 **aes128-ctr**、**aes192-ctr** 或 **aes256-ctr**。
- **sshd_MACs**:: 您可以选择 MAC，例如 **hmac-sha2-256**、**hmac-sha2-512** 或 **hmac-sha1**。
- **sshd_HostKeyAlgorithms**:: 您可以选择公钥算法，如 **ecdsa-sha2-nistp256**、**ecdsa-sha2-nistp384**、**ecdsa-sha2-nistp521**、**ssh-rsa** 或 **ssh-dss**。

在 RHEL 9 受管节点上，系统角色会将配置写入到 `/etc/ssh/sshd_config.d/00-ansible_system_role.conf` 文件中，其中加密选项会被自动应用。您可以使用 **sshd_config_file** 变量更改文件。但是，要确保配置有效，请使用一个字典顺序在 `/etc/ssh/sshd_config.d/50-redhat.conf` 文件之前的文件名，其包括配置的加密策略。

在 RHEL 8 受管节点上，您必须通过将 **sshd_sysconfig_override_crypto_policy** 和 **sshd_sysconfig** 变量设置为 **true** 来启用覆盖。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook :

```
$ ansible-playbook ~/playbook.yml
```

验证

- 您可以通过使用详细的 SSH 连接验证并在以下输出中检查定义的变量，来验证流程是否成功：

```
$ ssh -vvv <ssh_server>
...
debug2: peer server KEXINIT proposal
debug2: KEX algorithms: ecdh-sha2-nistp521
debug2: host key algorithms: rsa-sha2-512,rsa-sha2-256
debug2: ciphers ctos: aes256-ctr
debug2: ciphers stoc: aes256-ctr
debug2: MACs ctos: hmac-sha2-512-etm@openssh.com
debug2: MACs stoc: hmac-sha2-512-etm@openssh.com
...
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.sshd/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ssh/` 目录

26.5. ssh RHEL 系统角色的变量

在 **ssh** 系统角色 playbook 中，您可以根据您的偏好和限制为客户端 SSH 配置文件定义参数。

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。

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.ssh/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ssh/` 目录

26.6. 使用 ssh RHEL 系统角色配置 OPENSSH 客户端

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



注意

您可以将 **ssh** 系统角色与更改 SSH 和 SSHD 配置的其他系统角色（如身份管理 RHEL 系统角色）一起使用。要防止配置被覆盖，请确保 **ssh** 角色使用置入目录（在 RHEL 8 及更新版本中默认使用）。

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。

流程

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

```
---
- name: SSH client configuration
  hosts: managed-node-01.example.com
  tasks:
    - name: "Configure ssh clients"
      ansible.builtin.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: server.example.com
              User: user1
        ssh_FowardX11: 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 **server.example.com** 主机的 **example** 别名是 **user1**。
- 创建了 **example** 主机别名，它代表使用 **user1** 用户名到 **server.example.com** 主机的连接。
- X11 转发被禁用。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 通过显示 SSH 配置文件来验证受管节点是否有正确的配置：

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

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.ssh/README.md` 文件
- `/usr/share/doc/rhel-system-roles/ssh/` 目录

第 27 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 9 中可用的 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。

27.1. 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

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 文件
- `/usr/share/doc/rhel-system-roles/storage/` 目录

27.2. 使用 STORAGE RHEL 系统角色在块设备上创建 XFS 文件系统

示例 Ansible playbook 应用 **storage** 角色，来使用默认参数在块设备上创建 XFS 文件系统。



注意

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。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.storage
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
```

- 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 9 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[使用 storage RHEL 系统角色管理逻辑卷](#)。
不要提供到 LV 设备的路径。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 文件

- [/usr/share/doc/rhel-system-roles/storage/](#) 目录

27.3. 使用 STORAGE RHEL 系统角色永久挂载文件系统

示例 Ansible 应用 **storage** 角色，来立即且永久挂载 XFS 文件系统。

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。

流程

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

```
---
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.storage
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        mount_point: /mnt/data
        mount_user: somebody
        mount_group: somegroup
        mount_mode: 0755
```

- 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。
2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.storage/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/storage/](#) 目录

27.4. 使用 STORAGE RHEL 系统角色管理逻辑卷

示例 Ansible playbook 应用 **storage** 角色，来在卷组中创建 LVM 逻辑卷。

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。

流程

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

```
- hosts: managed-node-01.example.com
roles:
  - rhel-system-roles.storage
vars:
  storage_pools:
    - name: myvg
      disks:
        - sda
        - sdb
        - sdc
      volumes:
        - name: mylv
          size: 2G
          fs_type: ext4
          mount_point: /mnt/dat
```

- **myvg** 卷组由以下磁盘组成：**/dev/sda**、**/dev/sdb** 和 **/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。
2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件
- **/usr/share/doc/rhel-system-roles/storage/** 目录

27.5. 使用 STORAGE RHEL 系统角色启用在线块丢弃

示例 Ansible playbook 应用 **storage** 角色，来挂载启用了在线块丢弃的 XFS 文件系统。

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。

流程

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

```
---
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.storage
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        mount_point: /mnt/data
        mount_options: discard
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件
- **/usr/share/doc/rhel-system-roles/storage/** 目录

27.6. 使用 STORAGE RHEL 系统角色创建并挂载 EXT4 文件系统

示例 Ansible playbook 应用 **storage** 角色，来创建和挂载 Ext4 文件系统。

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。

流程

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

```
---
- hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.storage
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: ext4
        fs_label: label-name
        mount_point: /mnt/data
```

- playbook 在 **/dev/sdb** 磁盘上创建文件系统。
 - playbook 在 **/mnt/data** 目录永久挂载文件系统。
 - 文件系统的标签是 **label-name**。
2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件
- **/usr/share/doc/rhel-system-roles/storage/** 目录

27.7. 使用 STORAGE RHEL 系统角色创建并挂载 EXT3 文件系统

示例 Ansible playbook 应用 **storage** 角色，来创建和挂载 Ext3 文件系统。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- hosts: all
  roles:
    - rhel-system-roles.storage
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: ext3
        fs_label: label-name
        mount_point: /mnt/data
        mount_user: somebody
        mount_group: somegroup
        mount_mode: 0755
```

- playbook 在 **/dev/sdb** 磁盘上创建文件系统。
 - playbook 在 **/mnt/data** 目录永久挂载文件系统。
 - 文件系统的标签是 **label-name**。
2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 文件
- `/usr/share/doc/rhel-system-roles/storage/` 目录

27.8. 使用 STORAGE RHEL 系统角色调整 LVM 上现有文件系统的大小

示例 Ansible playbook 应用 **storage** RHEL 系统角色，来调整带有文件系统的 LVM 逻辑卷的大小。

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。

流程

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

```
---
- name: Create LVM pool over three disks
  hosts: managed-node-01.example.com
  tasks:
    - name: Resize LVM logical volume with file system
      ansible.builtin.include_role:
        name: rhel-system-roles.storage
      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
```

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。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件
- **/usr/share/doc/rhel-system-roles/storage/** 目录

27.9. 使用 STORAGE RHEL 系统角色创建交换卷

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），或者使用默认参数在块设备上修改交换卷（如果已存在）。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Create a disk device with swap
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.storage
  vars:
    storage_volumes:
      - name: swap_fs
        type: disk
        disks:
          - /dev/sdb
        size: 15 GiB
        fs_type: swap
```

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。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 文件
- `/usr/share/doc/rhel-system-roles/storage/` 目录

27.10. 使用 STORAGE RHEL 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。



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。

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。

流程

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

```
---
- name: Configure the storage
  hosts: managed-node-01.example.com
  tasks:
    - name: Create a RAID on sdd, sde, sdf, and sdg
      ansible.builtin.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
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 文件
- `/usr/share/doc/rhel-system-roles/storage/` 目录

27.11. 使用 STORAGE RHEL 系统角色配置带有 RAID 的 LVM 池

使用 **storage** 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。您可以使用可用参数建立一个 Ansible playbook，来配置带有 RAID 的 LVM 池。

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。

流程

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

```
---
- name: Configure LVM pool with RAID
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.storage
  vars:
    storage_safe_mode: false
  storage_pools:
    - name: my_pool
      type: lvm
      disks: [sdh, sdi]
      raid_level: raid1
      volumes:
        - name: my_volume
          size: "1 GiB"
          mount_point: "/mnt/app/shared"
          fs_type: xfs
          state: present
```

要创建带有 RAID 的 LVM 池，您必须使用 **raid_level** 参数指定 RAID 类型。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 文件
- **/usr/share/doc/rhel-system-roles/storage/** 目录
- [管理 RAID](#)

27.12. 使用 STORAGE RHEL 系统角色为 RAID LVM 卷配置条带大小

使用 **storage** 系统角色，您可以使用 Red Hat Ansible Automation Platform 在 RHEL 上为 RAID LVM 卷配置条带大小。您可以使用可用参数建立一个 Ansible playbook，来配置带有 RAID 的 LVM 池。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Configure stripe size for RAID LVM volumes
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.storage
  vars:
    storage_safe_mode: false
  storage_pools:
    - name: my_pool
      type: lvm
      disks: [sdh, sdi]
      volumes:
        - name: my_volume
          size: "1 GiB"
          mount_point: "/mnt/app/shared"
          fs_type: xfs
          raid_level: raid1
          raid_stripe_size: "256 KiB"
          state: present
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 文件
- `/usr/share/doc/rhel-system-roles/storage/` 目录
- [管理 RAID](#)

27.13. 使用 STORAGE RHEL 系统角色压缩和去重 LVM 上的 VDO 卷

示例 Ansible playbook 应用 **storage** RHEL 系统角色，来使用虚拟数据优化器(VDO)启用逻辑卷(LVM)的压缩和去重。



注意

由于 **storage** 系统角色使用 LVM VDO，因此每个池只有一个卷可以使用压缩和去重。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
- name: Create LVM VDO volume under volume group 'myvg'
  hosts: managed-node-01.example.com
  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。
2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 文件
- `/usr/share/doc/rhel-system-roles/storage/` 目录

27.14. 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

- 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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Create and configure a volume encrypted with LUKS
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.storage
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        fs_label: label-name
        mount_point: /mnt/data
        encryption: true
        encryption_password: <password>
```

您还可以将其他加密参数（如 `encryption_key`, `encryption_cipher`, `encryption_key_size` 和 `encryption_luks`）添加到 playbook 文件中。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/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]
...
```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.storage/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/storage/](#) 目录
- [使用 LUKS 加密块设备](#)

27.15. 使用 STORAGE RHEL 系统角色将池卷大小表示为百分比

示例 Ansible playbook 应用 **storage** 系统角色，使您能够将逻辑卷管理器卷(LVM)卷大小表示为池总大小的百分比。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Express volume sizes as a percentage of the pool's total size
  hosts: managed-node-01.example.com
  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 卷的大小指定为文件系统的人类可读大小的池大小的百分比（例如 **10g** 或 **50 GiB**）。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 文件
- `/usr/share/doc/rhel-system-roles/storage/` 目录

第 28 章 使用 RHEL 系统角色管理 systemd 单元

使用 **systemd** RHEL 系统角色，您可以使用 Red Hat Ansible Automation Platform 在多个系统上部署单元文件和管理 **systemd** 单元。

您可以在 **systemd** RHEL 系统角色 playbook 中使用 **systemd_units** 变量来深入了解目标系统上 **systemd** 单元的状态。变量显示一个字典列表。每个字典条目描述了受管主机上存在的一个 **systemd** 单元的状态和配置。**systemd_units** 变量作为任务执行的最后一步被更新，并在角色运行所有任务后捕获状态。

28.1. 使用 systemd RHEL 系统角色部署并启动 systemd 单元

您可以应用 **systemd** RHEL 系统角色，来在目标主机上执行与 **systemd** 单元管理相关的任务。您将在 playbook 中设置 **systemd** RHEL 系统角色变量，来定义 **systemd** 管理、启动和启用哪些单元文件。

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。

流程

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

```
---
- name: Deploy and start systemd unit
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.systemd
  vars:
    systemd_unit_files:
      - <name1>.service
      - <name2>.service
      - <name3>.service
    systemd_started_units:
      - <name1>.service
      - <name2>.service
      - <name3>.service
    systemd_enabled_units:
      - <name1>.service
      - <name2>.service
      - <name3>.service
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.systemd/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/systemd/](#) 目录

第 29 章 使用 RHEL 系统角色配置时间同步

使用 **timesync** RHEL 系统角色，您可以在 RHEL 上使用 Red Hat Ansible Automation Platform 在多台目标机器上管理时间同步。

29.1. TIMESYNC RHEL 系统角色

您可以使用 **timesync** RHEL 系统角色在多台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** 角色还有助于 [迁移到 chrony](#)，因为您可以在 RHEL 6 开始的所有 Red Hat Enterprise Linux 版本上使用相同的 playbook，无论系统是否使用 **ntp** 或 **chrony** 来实现 NTP 协议。

- `/usr/share/ansible/roles/rhel-system-roles/timesync/README.md` 文件
- `/usr/share/doc/rhel-system-roles/timesync/` 目录

29.2. 为单个服务器池应用 TIMESYNC RHEL 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

- 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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Manage time synchronization
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.timesync
  vars:
    timesync_ntp_servers:
```



```
- hostname: 2.rhel.pool.ntp.org
  pool: yes
  iburst: yes
```

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.timesync/README.md` 文件
- `/usr/share/doc/rhel-system-roles/timesync/` 目录

29.3. 在客户端服务器上应用 TIMESYNC RHEL 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。
- **chrony** NTP 提供者版本为 4.0 或更高版本。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Enable Network Time Security on NTP clients
  hosts: managed-node-01.example.com
```

```
roles:
  - rhel-system-roles.timesync
vars:
  timesync_ntp_servers:
    - hostname: ptbtime1.ptb.de
      iburst: yes
      nts: yes
```

ptbtime1.ptb.de 是公共服务器的一个示例。您可能想要使用不同的公共服务器或您自己的服务器。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/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

- `/usr/share/ansible/roles/rhel-system-roles.timesync/README.md` 文件
- `/usr/share/doc/rhel-system-roles/timesync/` 目录

第 30 章 使用 RHEL 系统角色为会话记录配置一个系统

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。

30.1. tLog RHEL 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一个 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

- [/usr/share/ansible/roles/rhel-system-roles/ha_cluster/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/ha_cluster/](#) 目录

30.2. tLog RHEL 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

其他资源

- [/usr/share/ansible/roles/rhel-system-roles/ha_cluster/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/ha_cluster/](#) 目录

30.3. 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。

playbook 在您指定的系统上安装 **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。

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。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Deploy session recording
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.tlog
  vars:
    tlog_scope_sssd: some
    tlog_users_sssd:
      - recorded-user
```

tlog_scope_sssd

一些 值指定只记录某些用户和组，而不是 **all** 或 **none**。

tlog_users_sssd

指定要从其记录会话的用户。请注意，这不会为您添加用户。您必须自行设置该用户。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

1. 进入创建 SSSD 配置丢弃文件的文件夹：

```
# cd /etc/sss/conf.d/
```

2. 检查文件内容：

```
# cat /etc/sss/conf.d/sss-session-recording.conf
```

您可以看到该文件包含您在 playbook 中设置的参数。

3. 以其会话将被记录的用户的身份登录。
4. [回放记录的会话](#)。

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.tlog/README.md` 文件
- `/usr/share/doc/rhel-system-roles/tlog/` 目录

30.4. 部署 TLOG RHEL 系统角色，以排除组或用户的列表

您可以使用 **tlog** 系统角色来支持 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。

playbook 在您指定的系统上安装 **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/sss/conf.d/sss-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。

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。

流程

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

```
---
- name: Deploy session recording excluding users and groups
  hosts: managed-node-01.example.com
  roles:
    - rhel-system-roles.tlog
  vars:
    tlog_scope_sssd: all
    tlog_exclude_users_sssd:
      - jeff
      - james
    tlog_exclude_groups_sssd:
      - admins
```

tlog_scope_sssd

值 **all** 指定您要记录所有用户和组。

tlog_exclude_users_sssd

指定您要从会话记录中排除的用户的用户名。

tlog_exclude_groups_sssd

指定您要从会话记录中排除的组。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

1. 进入创建 SSSD 配置丢弃文件的文件夹：

```
# cd /etc/sss/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。

3. 以其会话将被记录的用户的身分登录。
4. [回放记录的会话](#)。

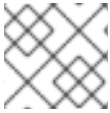
其他资源

- `/usr/share/ansible/roles/rhel-system-roles.tlog/README.md` 文件
- `/usr/share/doc/rhel-system-roles/tlog/` 目录

第 31 章 使用 RHEL 系统角色配置带有 IPSEC 的 VPN 连接

使用 **vpn** 系统角色，您可以使用 Red Hat Ansible Automation Platform 在 RHEL 系统上配置 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络配置。

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 机会主义网络配置。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（其是一种 IPsec 实现）作为 VPN 提供者。

31.1. 使用 **vpn** RHEL 系统角色创建一个带有 IPSEC 的主机到主机的 VPN

您可以使用 **vpn** 系统角色，通过在控制节点上运行 Ansible playbook 来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

- 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。

流程

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

```
- name: Host to host VPN
  hosts: managed-node-01.example.com, managed-node-02.example.com
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - hosts:
          managed-node-01.example.com:
          managed-node-02.example.com:
    vpn_manage_firewall: true
    vpn_manage_selinux: true
```

此 playbook 通过将预共享身份验证与系统角色自动生成的密钥一起使用，来配置连接 **managed-node-01.example.com-to-managed-node-02.example.com**。因为 **vpn_manage_firewall** 和 **vpn_manage_selinux** 都被设为 **true**，因此 **vpn** 角色使用 **firewall** 和 **selinux** 角色来管理 **vpn** 角色使用的端口。

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

```
vpn_connections:
  - hosts:
      managed-node-01.example.com:
```

```
<external_node>:
  hostname: <IP_address_or_hostname>
```

这将配置一个额外的连接：**managed-node-01.example.com-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。

2. 可选：您可以使用 **vpn_connections** 中的其它部分为受管节点指定多个 VPN 连接，如 control plane 和 data plane：

```
- name: Multiple VPN
  hosts: managed-node-01.example.com, managed-node-02.example.com
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - name: control_plane_vpn
        hosts:
          managed-node-01.example.com:
            hostname: 192.0.2.0 # IP for the control plane
          managed-node-02.example.com:
            hostname: 192.0.2.1
      - name: data_plane_vpn
        hosts:
          managed-node-01.example.com:
            hostname: 10.0.0.1 # IP for the data plane
          managed-node-02.example.com:
            hostname: 10.0.0.2
```

3. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

4. 运行 playbook：

```
$ ansible-playbook ~/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`。

2. 在受管节点上，确认连接是否成功启动：

```
# ipsec trafficstatus | grep <connection_name>
```

3. 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。这提供了更具体的信息，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会出现的任何错误都在 `/var/log/pluto.log` 文件中报告。由于这些日志很难解析，因此改为手动添加连接，以从标准输出获得日志消息。

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.vpn/README.md` 文件
- `/usr/share/doc/rhel-system-roles/vpn/` 目录

31.2. 使用 VPN RHEL 系统角色创建一个带有 IPSEC 的机会网状 VPN 连接

您可以使用 `vpn` 系统角色来配置一个机会网状 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，其将配置清单文件中列出的所有受管节点。

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。
- `/etc/ipsec.d/` 目录中的 IPsec 网络安全服务(NSS)加密库包含必需的证书。

流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
- name: Mesh VPN
  hosts: managed-node-01.example.com, managed-node-02.example.com, managed-node-03.example.com
  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
```

通过在 playbook 中定义 **auth_method: cert** 参数来配置用证书进行身份验证。默认情况下，节点名称用作证书的昵称。在本例中，这是 **managed-node-01.example.com**。您可以使用清单中的 **cert_name** 属性来定义不同的证书名称。

在本例流程中，控制节点是您将运行 Ansible playbook 的系统，与两个受管节点(192.0.2.0/24)共享同样的无类别域间路由(CIDR)数，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。

因为 **vpn_manage_firewall** 和 **vpn_manage_selinux** 都被设为 **true**，因此 **vpn** 角色使用 **firewall** 和 **selinux** 角色来管理 **vpn** 角色使用的端口。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

其他资源

- **/usr/share/ansible/roles/rhel-system-roles/vpn/README.md** 文件
- **/usr/share/doc/rhel-system-roles/vpn/** 目录