OpenShift Container Platform 3.11

安装集群

OpenShift Container Platform 3.11 安装集群

Last Updated: 2023-03-25

OpenShift Container Platform 3.11 安装集群

OpenShift Container Platform 3.11 安装集群

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

摘要

使用本指南安装 OpenShift Container Platform 3.11 集群

目录

第 1 章 规划安装 **6**
1.1. 最初规划 6
1.1.1. IBM POWER 安装的限制和注意事项 6
1.2. 与环境规模相关的注意事项 7
1.3. 环境场景 7
1.3.1. 在单一系统上运行单个 master 和节点 7
1.3.2. 单个 master 和多个节点 7
1.3.3. 使用原生 HA 的多个 master 7
1.3.4. 多个使用原生 HA 的、带有外部集群 etcd 的 Master。 8
1.3.5. 独立 registry 9
1.4. 支持的操作系统的安装类型 9
1.4.1. 系统容器所需的镜像 9
1.4.2. systemd 服务名称 10
1.4.3. 文件路径位置 10
1.4.4. 存储要求 10

第 2 章 系统和环境要求 **11**
2.1. 系统要求 11
2.1.1. 红帽订阅 11
2.1.2. 最低硬件要求 11
2.1.3. 产品级别硬件要求 13
2.1.4. 存储管理 13
2.1.5. Red Hat Gluster Storage 硬件要求 14
2.1.6. 监控硬件要求 15
2.1.7. SELinux 要求 15
2.1.8. 可选：配置内核用法 15
2.1.9. 可选：使用 OverlayFS 15
2.1.10. 安全警告 15
2.2. 环境要求 16
2.2.1. DNS 要求 16
2.2.1.1. 配置主机使用 DNS 17
2.2.1.2. 配置 DNS 通配符 17
2.2.1.3. 配置节点主机名 18
2.2.2. 网络访问要求 18
2.2.2.1. NetworkManager 18
2.2.2.2. 配置 firewalld 防火墙 19
2.2.2.3. 有多个网络接口的主机 19
2.2.2.4. 所需端口 19
2.2.3. 持久性存储 21
2.2.4. 使用云供应商的注意事项 22
2.2.4.1. 覆盖检测到的 IP 地址和主机名 22
2.2.4.2. Cloud Provider 的安装后配置 23

第 3 章 准备主机 **24**
3.1. 操作系统要求 24
3.2. 服务器类型要求 24
3.3. 设置 PATH 24
3.4. 确保主机访问 24
3.5. 设置代理覆盖 25
3.6. 注册主机 25
3.7. 安装基本软件包 27

3.8. 安装 DOCKER 28
3.9. 配置 DOCKER 存储 28
3.9.1. 配置 OverlayFS 29
3.9.2. 配置精简池存储 29
3.9.3. 配置 Docker 存储 32
3.9.4. 启用镜像签名支持 32
3.9.5. 管理容器日志 34
3.9.6. 查看可用的容器日志 34
3.9.7. 阻塞使用本地卷 34
3.10. RED HAT GLUSTER STORAGE 软件要求 35
3.11. 后续步骤 36

第 4 章 配置清单文件 37
4.1. 为您的集群自定义清单文件 37
4.2. 配置集群变量 37
4.3. 配置部署类型 44
4.4. 配置主机变量 44
4.5. 定义节点组和主机映射 45
4.5.1. 节点 ConfigMap 45
4.5.2. 节点组定义 46
4.5.3. 将主机映射到节点组 48
4.5.4. 节点主机标签 48
4.5.4.1. 主 pod 调度功能 49
4.5.4.2. 节点上的 Pod 调度功能 49
4.5.4.3. 配置 Dedicated Infrastructure 节点 49
4.6. 配置项目参数 50
4.7. 配置主 API 端口 51
4.8. 配置集群预安装检查 51
4.9. 配置 REGISTRY 位置 53
4.10. 配置 REGISTRY 路由 54
4.11. 配置路由器分片 55
4.12. 配置 RED HAT GLUSTER STORAGE PERSISTENT STORAGE 55
4.12.1. 配置聚合模式 55
4.12.2. 配置独立模式 57
4.13. 配置 OPENSHIFT CONTAINER REGISTRY 58
4.13.1. 配置 registry 存储 58
选项 A：NFS 主机组 58
选项 B：外部 NFS 主机 58
升级或安装带有 NFS 的 OpenShift Container Platform 59
选项 C：OpenStack Platform 59
选项 D:：WS 或者其它 S3 存储解决方案 59
选项 E：聚合模式 59
选项 F： Google Compute Engine（GCE）上的 Google Cloud Storage（GCS）存储桶 60
选项 G：使用 vSphere Cloud Provider（VCP）的 vSphere 卷 60
4.14. 配置全局代理选项 61
4.15. 配置防火墙 62
4.16. 配置会话选项 63
4.17. 配置自定义证书 63
4.18. 配置证书验证 64
4.19. 配置集群监控 65
4.20. 配置集群指标 65
4.20.1. 配置 Metrics 存储 65
选项 A：动态 66

选项 B：NFS 主机组 66
选项 C：外部 NFS 主机 66
升级或安装带有 NFS 的 OpenShift Container Platform 67
4.21. 配置集群日志记录 67
4.21.1. 配置日志存储 68
选项 A：动态 68
选项 B：NFS 主机组 68
选项 C：外部 NFS 主机 69
升级或安装带有 NFS 的 OpenShift Container Platform 69
4.22. 自定义服务目录选项 70
4.22.1. 配置 OpenShift Ansible Broker 70
4.22.1.1. 为 OpenShift Ansible Broker 配置持久性存储 71
4.22.1.2. 为本地 APB 开发配置 OpenShift Ansible Broker 72
4.22.2. 配置 Template Service Broker 72
4.23. 配置 WEB 控制台自定义 72
4.24. 配置集群控制台 74
4.25. 配置 OPERATOR LIFECYCLE MANAGER 74

第 5 章 清单文件示例 76
5.1. 概述 76
5.2. 单 MASTER 示例 76
5.2.1. 单 Master、单一 etcd 和多个节点 76
5.2.2. 单个 Master、多个 etcd 和多个节点 77
5.3. 多个主页示例 79
5.3.1. 多个使用原生 HA 的、带有外部集群 etcd 的 Master。 79
5.3.2. 多个使用原生 HA 的 Master 81

第 6 章 安装 OPENSHIFT CONTAINER PLATFORM 84
6.1. 先决条件 84
6.1.1. 运行基于 RPM 的安装程序 84
6.1.2. 运行容器化安装程序 85
6.1.2.1. 将安装程序作为系统容器运行 85
6.1.2.2. 运行其他 playbook 86
6.1.2.3. 将安装程序作为容器运行 86
6.1.2.4. 为 OpenStack 运行安装 Playbook 87
6.1.3. 关于安装 playbook 88
6.2. 重试安装 88
6.3. 验证安装 90
验证多个 etcd 主机 90
使用 HAProxy 验证多个 Master 91
6.4. （可选）安全构建 91
6.5. 已知问题 91
6.6. 下一步是什么？ 92

第 7 章 断开连接的安装 93
7.1. 先决条件 93
7.2. 获取所需的软件包和镜像 93
7.2.1. 获取 OpenShift Container Platform 软件包 93
7.2.2. 获取镜像 96
7.2.3. 导出镜像 99
7.3. 准备并填充存储库服务器 102
7.4. 填充 REGISTRY 103
7.5. 准备集群主机 104
7.6. 安装 OPENSHIFT CONTAINER PLATFORM 104

第 8 章 安装独立部署的 OPENSHIFT 容器镜像 REGISTRY 106
8.1. 最低硬件要求 106
8.2. 支持的系统拓扑 107
8.3. 安装 OPENSHIFT CONTAINER REGISTRY 107

第 9 章 卸载 OPENSHIFT CONTAINER PLATFORM 111
9.1. 卸载 OPENSHIFT CONTAINER PLATFORM 集群 111
9.2. 卸载节点 111

第 1 章 规划安装

您可以通过运行一系列 Ansible playbook 来安装 OpenShift Container Platform。在准备安装集群时，可以创建代表环境和 OpenShift Container Platform 集群配置的清单（inventory）文件。对 Ansible 有一定了解将有助于简化这个过程，但并不是必须的。

您可以在 Ansible 的 官方文档 中了解更多有关 Ansible 及其基本使用情况的信息。

1.1. 最初规划

在安装生产环境中的 OpenShift Container Platform 集群前，需要考虑以下问题：

- *您的内部服务器使用 IBM POWER 还是 x86_64 处理器？*OpenShift Container Platform 可以在使用这两种架构的服务器上安装。如果使用 POWER 服务器，请参阅在 IBM POWER 中安装的限制和考量。
- *集群需要多少个 pod?*大小考虑部分提供了与节点和 pod 的限制相关的信息，以便您规划您的环境。
- *集群中需要多少个主机？*环境场景 部分提供了多个单 Master 和多 Master 配置的多个示例。
- *您需要 高可用性 集群吗？*高可用性配置提高了容错性。在这种情况下，您可以使用 原生 HA 的多 Master 环境示例来设置您的环境。
- *是否需要 集群监控?*监控（monitoring）堆栈需要额外的 系统资源。请注意，监控堆栈会被默认安装。如需更多信息，请参阅集群监控文档。
- *您希望使用 Red Hat Enterprise Linux（RHEL）或 RHEL Atomic Host 作为集群节点的操作系统？*如果在 RHEL 上安装 OpenShift Container Platform，使用基于 RPM 的安装。在 RHEL Atomic Host 上，您需要使用一个系统容器。两种安装类型都提供一个有效的 OpenShift Container Platform 环境。
- *使用什么身份提供机制（indentity provider）进行身份验证？*如果您已使用受支持的用户提供机制，请将 OpenShift Container Platform 配置为在安装过程中使用该身份提供机制。
- *如果我的安装集成了其他技术，我的环境是否被支持？*如需已经过测试的集成列表，请参阅 OpenShift Container Platform Tested Integrations 。

1.1.1. IBM POWER 安装的限制和注意事项

从 3.10.45 开始，您可以在 IBM POWER 服务器上安装 OpenShift Container Platform。

- 您的集群必须只使用 Power 节点和 master。由于镜像的标记方式,OpenShift Container Platform 无法区分 x86 镜像和 Power 镜像。
- 升级时不会默认安装或更新镜像流和模板。您可以手动安装和更新镜像流。
- 您只能在内部 Power 服务器上安装。不能在云供应商的节点上安装 OpenShift Container Platform。
- 且并不支持所有存储。您只能使用以下存储环境：
  - GlusterFS
  - NFS

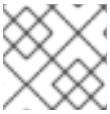
- 本地存储

1.2. 与环境规模相关的注意事项

确定 OpenShift Container Platform 集群需要多少个节点和 pod。集群的可扩展性与集群环境中的 pod 数量有关。它会影响设置中的其他设置。有关 OpenShift Container Platform 中对象的最新限制信息，请参阅集群限制。

1.3. 环境场景

使用这些环境场景帮助根据大小需求规划 OpenShift Container Platform 集群。



注意

在安装后从单一 master 集群移动到多个 master 集群不被支持。

在所有环境中，如果 etcd 主机与 master 主机在一起，则 etcd 会作为静态 pod 在主机上运行。如果 etcd 主机没有与 master 主机在一起，则会以独立进程的形式运行 etcd。

注意

如果使用 RHEL Atomic Host，则只能在 master 主机上配置 etcd。

1.3.1. 在单一系统上运行单个 master 和节点

只在单一系统上安装 OpenShift Container Platform 只适用于开发环境。您不能在生产环境中使用这个 *all-in-one 环境*。

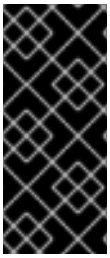
1.3.2. 单个 master 和多个节点

下表描述了单个 master（etcd 在同一个主机上安装）和两个节点的示例环境：

主机名	安装的基础架构组件
master.example.com	Master、etcd 和 node
node1.example.com	节点
node2.example.com	

1.3.3. 使用原生 HA 的多个 master

以下描述了一个示例环境，它有三个 master，一个 HAProxy 负载均衡器和两个使用 **原生 HA** 的 节点。etcd 在 master 节点上作为静态 pod 运行：



重要

路由器和 master 节点必须带有负载均衡才能实现一个具有高度性且容错的环境。红帽建议在生产环境中使用企业级外部负载均衡器。此负载均衡适用于 master 和节点，以及运行 OpenShift Container Platform 路由器的主机。推荐在传输控制协议（TCP）4 层进行负载均衡，其中的负载在 IP 地址之间分布。您可以参阅 外部负载均衡器与 OpenShift Enterprise 3 的集成，但不推荐在生产环境中使用。

主机名	安装的基础架构组件
master1.example.com	Master（使用原生 HA 进行集群）、节点和集群的 etcd
master2.example.com	
master3.example.com	
lb.example.com	用于负载均衡器 API 主端点的 HAProxy
node1.example.com	节点
node2.example.com	

1.3.4. 多个使用原生 HA 的、带有外部集群 etcd 的 Master。

以下是三个 master 的示例环境，一个 HAProxy 负载均衡器，三个外部集群的 etcd 主机，以及使用 **原生** HA 方法的两个 节点:

主机名	安装的基础架构组件
master1.example.com	Master（使用原生 HA 进行集群）和节点
master2.example.com	
master3.example.com	
lb.example.com	用于负载均衡器 API 主端点的 HAProxy
etcd1.example.com	集群的 etcd
etcd2.example.com	
etcd3.example.com	
node1.example.com	节点
node2.example.com	

1.3.5. 独立 registry

您还可以安装 OpenShift Container Platform 作为一个独立的 registry，它使用 OpenShift Container Platform 集成的 registry。有关此场景的详情，请参阅安装独立 registry。

1.4. 支持的操作系统的安装类型

从 OpenShift Container Platform 3.10 开始，如果您使用 RHEL 作为主机的底层操作系统，则使用 RPM 方法在该主机上安装 OpenShift Container Platform 组件。如果使用 RHEL Atomic Host，则在该主机上使用系统容器。这两种安装类型都为集群提供相同的功能，但您使用的操作系统决定了如何管理服务和主机更新。

注意

自 OpenShift Container Platform 3.10 起，Red Hat Enterprise Linux 系统不再支持 containzerized 安装方法。

RPM 通过软件包管理和配置服务安装所有服务在同一用户空间中运行，而系统容器使用系统容器镜像安装服务，并在独立容器中运行单独的服务。

当在 RHEL 上使用 RPM 时，所有服务都通过软件包管理从外部源安装和更新服务。这些软件包会修改主机在同一用户空间中的现有配置。在 RHEL Atomic Host 上安装系统容器时，OpenShift Container Platform 的每个组件都会作为一个容器（独立软件包）提供，该容器使用主机的内核运行。更新的、新版本的容器会替换了主机上的现有容器。

下表和小节概述了安装类型的区别：

表 1.1. 安装类型之间的区别

	Red Hat Enterprise Linux	RHEL Atomic Host
安装类型	基于 RPM	系统容器
交付机制	RPM 软件包，使用 **yum**	系统容器镜像，使用 **docker**
服务管理	**systemd**	**docker** 和 **systemd** 单元

1.4.1. 系统容器所需的镜像

系统容器安装类型使用以下镜像：

- **openshift3/ose-node**

默认情况下，所有上述镜像都是从 registry.redhat.io 的 Red Hat Registry 中拉取的。

如果需要在安装过程中使用私有 registry 来拉取镜像，可以提前指定 registry 信息。根据需要，在清单文件中设置以下 Ansible 变量：

```
oreg_url='<registry_hostname>/openshift3/ose-${component}:${version}'
openshift_docker_insecure_registries=<registry_hostname>
openshift_docker_blocked_registries=<registry_hostname>
```

注意

您还可以将 **openshift_docker_insecure_registries** 变量设置为主机的 IP 地址。**0.0.0.0/0** 不是一个有效设置。

默认组件从 **oreg_url** 值继承镜像前缀和版本。

额外的、不安全的以及受阻的容器 registry 的配置会在安装过程开始时生效，以确保在尝试拉取任何所需的镜像前应用这些设置。

1.4.2. systemd 服务名称

安装过程会创建相关的 **systemd** 单元，以便可以使用一般的 **systemctl** 命令启动、停止和轮询服务。对于系统容器安装，这些单元名称与 RPM 安装的名称相匹配。

1.4.3. 文件路径位置

容器化安装的所有 OpenShift Container Platform 配置文件和基于 RPM 安装的配置文件位于同一个位置，并会在 **os-tree** 升级后保留下来。

但是，对于 Atomic Host 安装，默认镜像流和模板文件 安装在 ***/etc/origin/examples/*** 中，而不是标准的 ***/usr/share/openshift/examples/*** 中，因为在 RHEL Atomic Host 中这个目录是只读的。

1.4.4. 存储要求

RHEL Atomic Host 安装通常有一个非常小的 root 文件系统。但是，etcd、master 和节点容器会在 ***/var/lib/*** 目录中保存数据。在安装 OpenShift Container Platform 前，请确保您在 root 文件系统中有足够的空间。详情请查看系统要求一节。

第 2 章 系统和环境要求

2.1. 系统要求

OpenShift Container Platform 环境中的主机必须满足以下硬件规格和系统级别要求。

2.1.1. 红帽订阅

您的红帽帐户必须具有有效的 OpenShift Container Platform 订阅。如果没有，请与您的销售代表联系以了解更多信息。

2.1.2. 最低硬件要求

系统要求因主机类型而异：

Master	• 物理或虚拟系统，或者在一个公共或私有 IaaS 环境中运行的实例。
	• 基础操作系统：使用 "Minimal" 安装选项以及来自 Extras 频道的最新软件包的 Red Hat Enterprise Linux（RHEL）7.5 或更新版本，或Atomic Host 7.4.5 或更新版本。
	  ◦ IBM POWER9: RHEL-ALT 7.5，使用 "Minimal" 安装选项以及来自 Extras 频道中的最新软件包。
	  ◦ IBM POWER8: RHEL 7.5，使用 "Minimal" 安装选项以及来自 Extras 频道的最新软件包。如果使用 RHEL，则必须使用以下最小内核版本：
	  ◦ RHEL 7.5: 3.10.0-862.31.1
	  ◦ RHEL 7.6: 3.10.0-957.27.2
	  ◦ RHEL 7.7: 3.10.0-1062
	• 最少 4 个 vCPU（强烈建议使用更多）。
	• 最少 16 GB RAM（强烈建议使用更多内存，特别是 etcd 和 master 在一起）。
	• 最小 40 GB 硬盘空间，用于包含 */var/* 的文件系统。 1
	• 最小 1 GB 硬盘空间，用于包含 */usr/local/bin/* 的文件系统。
	• 最小1 GB 硬盘空间，用于包含系统临时目录的文件系统。 2
	• etcd 和 master 在一起至少需要 4 个内核。双核系统无法工作。

节点	• 物理或虚拟系统，或在公有或私有 IaaS 上运行的实例。
	• 基础操作系统：使用 "Minimal" 安装选项的 RHEL 7.5 或更高版本，或 RHEL Atomic Host 7.4.5 或更新版本。
	  ◦ IBM POWER9: RHEL-ALT 7.5，使用 "Minimal" 安装选项以及来自 Extras 频道中的最新软件包。
	  ◦ IBM POWER8: RHEL 7.5，使用 "Minimal" 安装选项以及来自 Extras 频道的最新软件包。如果使用 RHEL，则必须使用以下最小内核版本：
	  ◦ RHEL 7.5: 3.10.0-862.31.1
	  ◦ RHEL 7.6: 3.10.0-957.27.2
	  ◦ RHEL 7.7: 3.10.0-1062
	• NetworkManager 1.0 或更高版本。
	• 1 个 vCPU。
	• 最小 8 GB RAM。
	• 最小 15 GB 硬盘空间，用于包含 */var/* 的文件系统。 1
	• 最小 1 GB 硬盘空间，用于包含 */usr/local/bin/* 的文件系统。
	• 最小1 GB 硬盘空间，用于包含系统临时目录的文件系统。 2
	• 为 Docker 的存储后端运行容器的每个系统都最少需要一个 15 GB 未分配空间 ; 请参阅 配置 Docker 存储。可能还需要额外空间，这要看节点上运行的容器的大小和数量。

外部 etcd 节点	• etcd 数据需要最少 20 GB 硬盘空间。 • 有关如何正确配置 etcd 节点的信息，请参阅 CoreOS etcd 文档中的硬件建议部分。 • 目前，OpenShift Container Platform 在 etcd 中存储镜像、构建和部署元数据。您需要定期 修剪旧资源。如果您计划大量使用这些资源，请将 etcd 放置到有大量内存和快速 SSD 驱动器的机器上。
Ansible 控制器	运行 Ansible playbook 的主机，对于清单（inventory）中的每个主机都至少需要有 75MiB 可用内存。

1 满足 RHEL Atomic Host 中 */var/* 文件系统大小的要求需要对默认配置进行修改。有关在安装过程中或安装后配置此功能的说明，请参阅使用 Docker 格式容器管理存储。

2 系统的临时目录根据 Python 标准库中 **tempfile 模块中定义的规则确定。**

您必须为每个运行容器守护进程的系统配置存储。对于容器化安装，您需要在 master 上存储。另外，在默认情况下，Web 控制台在 master 上的容器中运行，masters 需要存储来运行 Web 控制台。容器在节点上运行，因此节点始终需要存储。存储的大小取决于工作负载、容器数量、运行容器的大小以及容器的存储要求。您还必须配置存储以运行容器化 etcd。

强烈建议您使用带有可快速处理串口写入(fsync)的存储的 etcd，如 NVMe 或者 SSD。不建议使用 Ceph、NFS 和 spinning 磁盘。

2.1.3. 产品级别硬件要求

满足最低要求的测试或示例环境功能。对于生产环境，建议使用以下设置：

Master 主机

在带有外部 etcd 的具有高可用性功能的 OpenShift Container Platform 集群中，master 主机需要满足最低要求，每 1000 个 Pod 需要 1 个 CPU 内核和 1.5 GB 内存。因此，建议有 2000 个 pod 的 OpenShift Container Platform集群中的 master 主机最少需要 2 个 CPU 内核，16 GB RAM，再加上 2 个 CPU 内核和 3 GB RAM，总计 4 个 CPU 内核和 19 GB RAM。

更多详情，请参阅 OpenShift Container Platform master 主机的建议实践。

节点主机

节点主机的大小取决于其预期的工作负载大小。作为 OpenShift Container Platform 集群管理员，您必须计算预期的工作负载，再加上大约 10% 的开销。对于生产环境，请分配足够的资源，以防止节点主机故障影响您的最大容量。

如需更多信息，请参阅 大小考虑 和 集群限制。

重要

在节点中过度订阅物理资源会影响在 pod 放置过程中对 Kubernetes 调度程序的资源保证。了解可以采取什么措施避免出现内存交换问题。

2.1.4. 存储管理

表 2.1. OpenShift Container Platform 组件写入数据的主目录

目录	备注	大小	预期的增长
/var/lib/openshift	只用于 etcd 存储，当单一 master 模式且 etcd 嵌入到 **atomic-openshift-master** 进程中时。	小于 10GB。	随着环境增长会缓慢增长。只存储元数据。
/var/lib/etcd	用于 etcd 存储，当多 Master 模式或由管理员创建独立 etcd 时 。	小于 20 GB。	随着环境增长会缓慢增长。只存储元数据。
/var/lib/docker	当运行时为 docker 时，这是挂载点。用于活跃容器运行时（包括 Pod）和本地镜像存储（不用于 registry 存储）。挂载点应当由 docker-storage 管理，而不是手动管理。	有 16 GB 内存的节点需要 50 GB。 每多加 8 GB 内存需要额外 20-25 GB。	增长受运行容器容量的限制。

目录	备注	大小	预期的增长
/var/lib/containers	当运行时为 CRI-O 时，这是挂载点。用于活跃容器运行时（包括 Pod）和本地镜像存储（不用于 registry 存储）。	有 16 GB 内存的节点需要 50 GB。 每多加 8 GB 内存需要额外 20-25 GB。	增长受运行容器容量的限制。
/var/lib/origin/openshift.local.volumes	pod 的临时卷（Ephemeral volume）存储。这包括在运行时挂载到容器的任何外部存储。包括环境变量、kube secrets 以及不受持久性存储 PV 支持的数据卷。	可变	如果需要存储的 pod 使用持久性卷，则最小。如果使用临时存储，可能会快速增长。
/var/log	所有组件的日志文件。	10 到 30 GB。	日志文件可能会快速增长；大小可由通过增加磁盘或对日志进行轮转进行控制。

2.1.5. Red Hat Gluster Storage 硬件要求

任何在聚合模式或独立模式集群中使用的节点都被视为存储节点。虽然单个节点不能在多个组中，但存储节点可以被分到不同的集群组。对于每个存储节点组：

- **根据存储 gluster volumetype 选项，每个组最少需要一个或多个存储节点。**
- **每个存储节点必须至少有 8 GB RAM。这可允许运行 Red Hat Gluster Storage Pod，以及其他应用程序和底层操作系统。**
  - **每个 GlusterFS 卷也在其存储集群中的每个存储节点上消耗内存，大约是 30MB。RAM 总量应该根据计划或预期的并发卷数量来决定。**
- **每个存储节点必须至少有一个没有当前数据或元数据的原始块设备。这些块设备将完全用于 GlusterFS 存储。确保不存在以下内容：**
  - **分区表（GPT 或 MSDOS）**
  - **文件系统或者文件系统签名**
  - **前卷组和逻辑卷的 LVM2 签名**
  - **LVM2 物理卷的 LVM2 元数据**

  如果有疑问，使用 wipefs -a <device> 命令清除以上数据。

重要

建议规划两个集群：一个用于存储基础架构应用程序（如 OpenShift Container Registry）的专用存储集群，和一个用于常规应用程序的专用存储集群。这需要总共 6 个存储节点。此项建议是为了避免在创建 I/O 和卷时对性能造成潜在影响。

2.1.6. 监控硬件要求

监控堆栈需要额外的系统资源，它会被默认安装。请参阅 计算资源建议 和 集群监控文档。

2.1.7. SELinux 要求

在安装 OpenShift Container Platform 之前，必须在所有服务器上启用 Security-Enhanced Linux（SELinux），否则安装程序将失败。另外，在 */etc/selinux/config* 文件中配置 **SELINUX =enforcing 和 SELINUXTYPE=targeted**:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

2.1.8. 可选：配置内核用法

默认情况下，OpenShift Container Platform master 和节点使用它们运行的系统中的所有可用内核。您可以通过设置 **GOMAXPROCS** 环境变量来选择 OpenShift Container Platform 使用的内核数。如需更多信息，请参阅 Go Language 文档，包括 **GOMAXPROCS** 环境变量是如何工作的。

例如，在启动服务器前运行以下命令使 OpenShift Container Platform 只在一个核心上运行：

```
# export GOMAXPROCS=1
```

2.1.9. 可选：使用 OverlayFS

OverlayFS 是一个联合文件系统，允许用户在一个文件系统上使用另一个文件系统。

从 Red Hat Enterprise Linux 7.4 开始，您可以选择将 OpenShift Container Platform 环境配置为使用 OverlayFS。除了老的 **overlay** 驱动程序外，还完全支持 **overlay2** 图形驱动程序。但是，红帽建议使用 **overlay2** 而不是 **overlay**，因为其有速度和简单实施的优点。

比较 Overlay 和 Overlay2 Graph Drivers提供了更多与 overlay 和 overlay2 驱动程序相关的信息。

有关如何为 Docker 服务启用 **overlay2** 图形驱动程序的说明，请参见 Atomic Host 文档中的Overlay 图形驱动程序部分。

2.1.10. 安全警告

OpenShift Container Platform 在集群中的主机上运行容器，在一些情况下（如构建操作和 registry 服务），它使用特权容器进行操作。另外，这些容器会访问主机的 Docker 守护进程，并执行 **docker build** 和 **docker push** 操作。因此，集群管理员必须了解对镜像执行 **docker run** 操作相关的固有安全风险，因为它们实际具有根访问权限。这对 **docker build** 操作尤其重要。

可将有风险的容器分配给特定的节点，从而使可能的风险只限制在这些特定的节点中。更多信息，请参阅开发指南中的指定构建到特定节点 部分。对于集群管理员，请参阅 配置全局构建默认值和覆盖标题。

您还可以使用 安全上下文约束来控制 Pod 可以执行的操作，以及它有权访问的内容。有关如何在 Dockerfile 中使用 USER 来启用镜像运行的信息，请参阅管理安全性上下文约束 （需要具有 cluster-admin 特权的用户）。

如需了解更多相关信息，请参阅以下文章：

- http://opensource.com/business/14/7/docker-security-selinux
- https://docs.docker.com/engine/security/security/

2.2. 环境要求

以下部分定义包含 OpenShift Container Platform 配置的环境要求。这包括网络注意事项，以及对外部服务的访问，如 Git 存储库访问、存储和云基础架构供应商。

2.2.1. DNS 要求

OpenShift Container Platform 需要环境中具有一个可以全面正常工作的 DNS 服务器。理想的情况是，在一个单独的主机上运行 DNS 软件，为平台上运行的主机和容器提供名称解析服务。

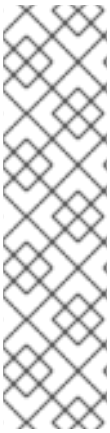
重要

只在每个主机的 */etc/hosts* 文件中添加条目并不够。此文件不会被复制到平台上运行的容器中。

OpenShift Container Platform 的主要组件在容器内运行，并使用以下流程进行名称解析：

1. 默认情况下，容器从它们的主机中接收 DNS 配置文件（*/etc/resolv.conf*）。
2. 然后，OpenShift Container Platform 将 pod 的第一个名称服务器设置为节点的 IP 地址。

从 OpenShift Container Platform 3.2 开始，在所有 master 和节点上自动配置 dnsmasq。pod 使用节点作为其 DNS，节点会转发请求。默认情况下，dnsmasq 被配置为侦听节点上的端口 53，因此节点无法运行任何其他类型的 DNS 应用程序。



注意

NetworkManager 是提供自动连接到网络的系统的检测和配置程序。节点上需要它以便使用 DNS IP 地址填充 dnsmasq。

默认将 **NM_CONTROLLED** 设置为 **yes**。如果 **NM_CONTROLLED** 被设置为 **no**，则 NetworkManager 分配脚本不会创建相关的 *origin-upstream-dns.conf* dnsmasq 文件，您必须手动配置 dnsmasq。

同样，如果在网络脚本中将 **PEERDNS** 参数设置为 **no**，例如 */etc/sysconfig/network-scripts/ifcfg-em1*，则 dnsmasq 文件不会被生成，Ansible 安装将失败。确定将 **PEERDNS** 设置设定为 **yes**。

以下是 DNS 记录示例：

```
master1    A   10.64.33.100
master2    A   10.64.33.103
node1      A   10.64.33.101
node2      A   10.64.33.102
```

如果没有可以正常工作的 DNS 环境，则可能会遇到以下问题：

- 通过引用基于 Ansible 的脚本进行产品安装
- 部署基础架构容器（registry、路由器）
- 访问 OpenShift Container Platform Web 控制台，因为它只能通过 IP 地址访问

2.2.1.1. 配置主机使用 DNS

确保将环境中的每个主机配置为从 DNS 服务器解析主机名。主机 DNS 解析的配置取决于是否启用了 DHCP。如果 DHCP 是：

- 禁用，将网络接口配置为静态，并在 NetworkManager 中添加 DNS 名称服务器。
- 启用，NetworkManager 分配脚本根据 DHCP 配置自动配置 DNS。

验证您的 DNS 服务器可以解析主机：

1. 检查 */etc/resolv.conf* 文件的内容：

```
$ cat /etc/resolv.conf
# Generated by NetworkManager
search example.com
nameserver 10.64.33.1
# nameserver updated by /etc/NetworkManager/dispatcher.d/99-origin-dns.sh
```

   在这个示例中，10.64.33.1 是 DNS 服务器的地址。

2. 测试 */etc/resolv.conf* 中列出的 DNS 服务器是否可以解析到 OpenShift Container Platform 环境中所有 master 和节点的 IP 地址：

```
$ dig <node_hostname> @<IP_address> +short
```

   例如：

```
$ dig master.example.com @10.64.33.1 +short
10.64.33.100
$ dig node1.example.com @10.64.33.1 +short
10.64.33.101
```

2.2.1.2. 配置 DNS 通配符

另外，还可为路由器配置通配符，以便在添加新路由时无需更新 DNS 配置。如果您为路由器配置通配符，请在配置 Ansible 清单文件 时将 **openshift_master_default_subdomain** 参数设置为这个值。

DNS 区的通配符最终必须解析为 OpenShift Container Platform router 的 IP 地址。

例如，为 cloudapp 创建通配符 DNS 条目，该条目的 Time-to-live 值较低（TTL），并指向要部署路由器的主机的公共 IP 地址：

```
*.cloudapps.example.com. 300 IN  A 192.168.133.2
```

警告

在每个节点主机上的 */etc/resolv.conf* 文件中，确保具有通配符条目的 DNS 服务器没有列为名称服务器，或者通配符域不在搜索列表中列出。否则，由 OpenShift Container Platform 管理的容器可能无法正确解析主机名。

2.2.1.3. 配置节点主机名

在设置不使用云供应商环境的集群时，您必须正确设置节点的主机名。每个节点的主机名必须可以被解析，每个节点必须能够访问其他节点。

确认节点可以访问另一个节点：

1. **在一个节点上，获取主机名：**

```
$ hostname

master-1.example.com
```

2. **在同一节点上，获取主机的完全限定域名：**

```
$ hostname -f

master-1.example.com
```

3. **在另外一个节点中，确认您可以访问到第一个节点：**

```
$ ping master-1.example.com -c 1

PING master-1.example.com (172.16.122.9) 56(84) bytes of data.
64 bytes from master-1.example.com (172.16.122.9): icmp_seq=1 ttl=64 time=0.319 ms

--- master-1.example.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.319/0.319/0.319/0.000 ms
```

2.2.2. 网络访问要求

master 和节点主机之间必须存在一个共享网络。如果您计划使用标准集群安装过程 为高可用性配置多个 master，则必须在安装过程中选择将 IP 配置为您的 虚拟 IP (VIP)。您选择的 IP 必须在所有节点之间路由，如果使用 FQDN 进行配置，则必须在所有节点上解析。

2.2.2.1. NetworkManager

NetworkManager 是提供自动连接到网络的系统的检测和配置程序。节点上需要它以便使用 DNS IP 地址填充 dnsmasq。

默认将 NM_CONTROLLED 设置为 yes。如果 NM_CONTROLLED 被设置为 no，则 NetworkManager 分配脚本不会创建相关的 *origin-upstream-dns.conf* dnsmasq 文件，您必须手动配置 dnsmasq。

2.2.2.2. 配置 firewalld 防火墙

虽然 iptables 是默认的防火墙，但推荐在新的安装中使用 firewalld。您可以通过在 Ansible 清单文件 中设置 **os_firewall_use_firewalld=true** 来启用 firewalld。

```
[OSEv3:vars]
os_firewall_use_firewalld=True
```

将这个变量设置为 **true** 将打开所需的端口，并在默认区中添加规则，这样可保证正确配置了 firewalld。

注意

使用 firewalld 的默认配置会带有有限的配置选项，它们不能被覆盖。例如，当您可以在多个区中使用接口设置存储网络时，节点通信的接口必须位于默认区。

2.2.2.3. 有多个网络接口的主机

如果主机有一个网络接口，OpenShift Container Platform 只使用一个网络接口来安装、集群网络和服务网络。您可以使用其他网络接口来与 OpenShift Container Platform 相关的通信，但不支持通过一个网络接口路由一些集群相关流量，并通过另一个网络接口将不同集群相关的流量路由。

2.2.2.4. 所需端口

OpenShift Container Platform 安装会在每台主机上使用 iptables 自动创建一组内部防火墙规则。然而，如果您的网络配置使用外部防火墙，如基于硬件的防火墙，您必须确保基础架构组件可以通过作为特定进程或服务的通信端点的特定端口相互通信。

确保 OpenShift Container Platform 所需的以下端口在网络上打开，并配置为允许主机间的访问。根据您的配置和使用方法，有些端口是可选的。

表 2.2. 节点到节点

4789	UDP	对于独立主机上的 pod 间的 SDN 通信需要此项。

表 2.3. 节点到 Master

4789	UDP	对于独立主机上的 pod 间的 SDN 通信需要此项。
443 或 8443	TCP	节点主机与 master API 通信、节点主机返回状态、接收任务等需要这个端口。

表 2.4. Master 到节点

4789	UDP	对于独立主机上的 pod 间的 SDN 通信需要此项。
10250	TCP	**oc** 命令通过Kubelet 的 master 代理到节点的通信需要它。这个端口必须允许从 master 和 infra 节点到任何 master 和节点的通信。对于指标数据（metrics），源必须是 infra 节点。
10010	TCP	如果使用 CRI-O，打开这个端口来允许 **oc exec** 和 **oc rsh** 操作。

表 2.5. Master 到 Master

2049	TCP/UDP	作为安装程序的一部分置备 NFS 主机时需要。
2379	TCP	用于独立 etcd（clustered）接受状态更改。
2380	TCP	etcd 需要此端口在 master 之间打开，以便在使用独立 etcd（clustered）时进行领导选举和对等连接。
4789	UDP	对于独立主机上的 pod 间的 SDN 通信需要此项。

表 2.6. 外部到 Load Balancer

9000	TCP	如果选择了 **原生** HA 方法，则可以选择允许访问 HAProxy 统计页面。

表 2.7. 外部到 Master

443 或 8443	TCP	节点主机与 master API 通信、节点主机返回状态、接收任务等需要这个端口。
8444	TCP	控制器管理器和调度程序服务侦听的端口。需要为 **/metrics** 和 **/healthz** 端点打开。

表 2.8. IaaS 部署

22	TCP	安装程序或者系统管理员需要 SSH。
53 或 8053	TCP/UDP	集群服务的 DNS 解析需要（SkyDNS）。3.2 版本之前的安装或升级到 3.2 的环境使用端口 53。新的安装将默认使用 8053，以便可以配置 **dnsmasq**。仅需要在 master 主机内部间打开。
80 或 443	TCP	用于路由器的 HTTP/HTTPS。需要在节点主机上对外部打开，特别是在运行路由器的节点中。
1936	TCP	（**可选**）运行模板路由器时需要打开访问统计信息。可以为外部或者内部打开连接，取决于是否想公开显示统计数据。可能需要额外配置来打开。详情请查看下面的备注部分。
2379 和 2380	TCP	供独立 etcd 使用。仅需在 master 主机上内部打开。**2379** 用于服务器/客户端连接。**2380** 用于服务器/服务器连接，只在有集群的 etcd 时才需要。
4789	UDP	对于 VxLAN 使用（OpenShift SDN）。仅在节点主机内部需要。
8443	TCP	与 API 服务器共享，供 OpenShift Container Platform Web 控制台使用。
10250	TCP	供 Kubelet 使用。需要在节点上打开为外部打开。

备注

- 在上面的例子中，端口 4789 用于 User Datagram Protocol（UDP）。
- 当部署使用 SDN 时，pod 网络可通过服务代理访问，除非它从 registry 部署的同一节点访问 registry。
- OpenShift Container Platform 内部 DNS 无法通过 SDN 接收。对于非云的部署，这会默认为与 master 主机上默认路由关联的 IP 地址。对于云部署，它将默认为与云元数据定义的第一个内部接口关联的 IP 地址。
- master 主机使用端口 10250 访问节点，而不通过 SDN。它依赖于部署的目标主机，并使用计算的 **openshift_public_hostname** 值。
- 取决于 iptables 规则的具体设置，端口 1936 可能仍然不能访问。使用以下内容配置 iptables 打开端口 1936:

```
# iptables -A OS_FIREWALL_ALLOW -p tcp -m state --state NEW -m tcp \
    --dport 1936 -j ACCEPT
```

表 2.9. 聚合的日志记录（Logging）和指标数据（Metrics）

9200	TCP	用于 Elasticsearch API。需要内部在任何基础架构节点上打开，以便 Kibana 能够检索日志以进行显示。它可以在外部开放，可通过路由直接访问 Elasticsearch。路由可以使用 **oc expose** 创建。
9300	TCP	用于 Elasticsearch 集群间使用。需要在所有基础架构节点上内部打开，以便 Elasticsearch 集群成员可以相互通信。
9090	TCP	用于 Prometheus API 和 Web 控制台。
9100	TCP	对于 Prometheus Node-Exporter，它会导出硬件和操作系统指标数据。需要在每个 OpenShift Container Platform 主机上打开端口 9100，以便 Prometheus 服务器提取指标。
8443	TCP	节点主机与 master API 通信，用于节点主机返回状态、接收任务等。这个端口必须允许从 master 和 infra 节点到任何 master 和节点的通信。
10250	TCP	对于 Kubernetes cAdvisor（容器资源使用和性能分析代理）。这个端口必须允许从 master 和 infra 节点到任何 master 和节点的通信。对于指标数据（metrics），源必须是 infra 节点。
8444	TCP	控制器管理器和调度程序服务侦听的端口。在每个 OpenShift Container Platform 主机上都必须打开端口 8444。
1936	TCP	（可选）运行模板路由器时需要打开访问统计信息。如果在路由器上启用了 Prometheus metrics，则必须允许从 infra 节点到所有托管路由器的infra 节点。可以为外部或者内部打开连接，取决于是否想公开显示统计数据。可能需要额外配置来打开。如需更多信息，请参阅上面的备注部分。

备注

2.2.3. 持久性存储

Kubernetes 持久性卷 框架允许您使用环境中可用的联网存储来置备带有持久性存储的 OpenShift Container Platform 集群。这可在根据应用程序需要完成初始 OpenShift Container Platform 安装后完成，为用户提供在不了解底层基础架构的情况下请求这些资源的方法。

《配置集群指南》为集群管理员提供了使用 NFS、GlusterFS、Ceph RBD、OpenStack Cinder、AWS Elastic Block Store(EBS)、GCE Persistent Disks 和 iSCSI 使用持久性存储置备 OpenShift Container Platform 集群的说明。

2.2.4. 使用云供应商的注意事项

如果在云供应商环境上安装 OpenShift Container Platform 时，需要考虑某些方面。

- 对于 Amazon Web Services，请参阅 权限 和 配置安全组 部分。
- 对于 OpenStack，请参阅 权限 和 配置安全组 部分。

2.2.4.1. 覆盖检测到的 IP 地址和主机名

有些部署需要用户覆盖检测到的主机名和主机的 IP 地址。要查看默认值，请切换到 playbook 目录并运行 **openshift_facts** playbook:

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook  [-i /path/to/inventory] \
  playbooks/byo/openshift_facts.yml
```

重要

对于 Amazon Web Services，请参阅 覆盖已检测到的 IP 地址和主机名称 部分。

现在，验证检测到的常见设置。如果他们不是您所期望的，可以重载它们。

配置清单文件 更详细地讨论可用的 Ansible 变量。

变量	使用方法
hostname	• 从实例本身解析到内部 IP 地址。
ip	• 实例的内部 IP 地址。
public_hostname	• 解析到云外主机的外部 IP。 • Provider **openshift_public_hostname** overrides.
public_ip	• 与实例关联的外部访问 IP 地址。 • **openshift_public_ip** 覆盖。

变量	使用方法
use_openshift_sdn	• 对于 GCE 以外的所有云，设置为 **true**。 • **openshift_use_openshift_sdn** 覆盖。

2.2.4.2. Cloud Provider 的安装后配置

在安装过程后，您可以为 AWS、 OpenStack 或 GCE 配置 OpenShift Container Platform。

第 3 章 准备主机

在安装 OpenShift Container Platform 之前，您必须准备节点主机。它们必须满足以下要求。

3.1. 操作系统要求

master 和节点主机的操作系统要求根据您的服务器架构的不同而有所不同。

- 对于使用 x86_64 构架的服务器，使用 Red Hat Enterprise Linux（RHEL）7.5 或更新版本的基本安装，以及来自 Extras 频道或者 RHEL Atomic Host 7.4.2 或更新版本中的最新软件包。
- 对于基于云的安装，请使用 RHEL 7.5 或更高版本的基本安装，以及来自 Extras 频道的最新软件包。
- 对于使用 IBM POWER8 架构的服务器，使用 RHEL 7.5 或更新的版本以及来自 Extras 频道的最新软件包。
- 对于使用 IBM POWER9 架构的服务器，使用 RHEL-ALT 7.5 或更高版本的基本安装，以及来自 Extras 频道的最新软件包。

如果需要，请参阅以下文档中的相关安装说明：

- Red Hat Enterprise Linux 7 安装指南
- Red Hat Enterprise Linux Atomic Host 7 安装和配置指南

3.2. 服务器类型要求

如果您在节点中使用 IBM POWER 服务器，则只能使用 IBM POWER 服务器。您不能将运行在 IBM POWER 服务器上的节点添加到使用 x86_64 服务器的现有集群中，或者将集群节点混合部署在 IBM POWER 和 x86_64 服务器中。

3.3. 设置 PATH

每个主机上的 root 用户的 **PATH** 必须包含以下目录：

- */bin*
- */sbin*
- */usr/bin*
- */usr/sbin*

这些目录默认在新的 RHEL 7.x 安装中设置。

3.4. 确保主机访问

OpenShift Container Platform 安装程序需要一个可以访问所有主机的用户。如果您想以非 root 用户身份运行安装程序，首先为每个主机配置无密码 sudo 权利：

1. 在运行安装 playbook 的主机上生成 SSH 密钥：

```
# ssh-keygen
```

不要 使用密码。

2. 将密钥分配到其他集群主机。您可以使用 bash 循环：

```
# for host in master.example.com \ 1
  node1.example.com \ 2
  node2.example.com; \ 3
  do ssh-copy-id -i ~/.ssh/id_rsa.pub $host; \
  done
```

1 2 3 提供每个集群主机的主机名。

3. 确认您可以通过 SSH 访问循环中列出的每个主机。

3.5. 设置代理覆盖

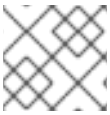
如果节点上的 */etc/environment* 文件包含 http_proxy 或 https_proxy 值，则必须在该文件中设置 no_proxy 值，以允许 OpenShift Container Platform 组件之间的开放通信。

注意

/etc/environment 文件中的 no_proxy 参数与清单文件中设置的全局代理值不同。全局代理值使用您的代理设置配置特定的 OpenShift Container Platform 服务。详情请查看 配置全局代理选项。

如果 */etc/environment* 文件包含代理值，在每个节点上的 no_proxy 参数中定义以下值：

- master 和节点主机名或其域后缀。
- 其他内部主机名或者它们的域后缀。
- etcd IP 地址。您必须提供 IP 地址，而不是主机名，因为 etcd 访问是由 IP 地址控制的。
- Kubernetes IP 地址，默认为 172.30.0.1。必须是清单文件中 openshift_portal_net 参数中设置的值。
- Kubernetes 内部域后缀 cluster.local。
- Kubernetes 内部域后缀 .svc。



注意

因为 no_proxy 不支持 CIDR，所以可以使用域后缀。

如果使用 http_proxy 或 https_proxy 值，则 no_proxy 参数值类似以下示例：

```
no_proxy=.internal.example.com,10.0.0.1,10.0.0.2,10.0.0.3,.cluster.local,.svc,localhost,127.0.0.1,172.30
0.1
```

3.6. 注册主机

要访问安装软件包，您必须使用 Red Hat Subscription Manager（RHSM）注册每个主机，并附加有效的 OpenShift Container Platform 订阅。

1. 在每一主机上进行 RHSM 注册：

```
# subscription-manager register --username=<user_name> --password=<password>
```

2. 从 RHSM 获取最新的订阅数据：

```
# subscription-manager refresh
```

3. 列出可用的订阅：

```
# subscription-manager list --available --matches '*OpenShift*'
```

4. 在上一命令的输出中，找到 OpenShift Container Platform 订阅的池 ID 并附加该池：

```
# subscription-manager attach --pool=<pool_id>
```

5. 禁用所有 yum 存储库：

   a. 禁用所有已启用的 RHSM 存储库：

   ```
   # subscription-manager repos --disable="*"
   ```

   b. 列出剩余的 yum 存储库，并记录它们在 repo id 下的名称（若有）：

   ```
   # yum repolist
   ```

   c. 使用 yum-config-manager 禁用剩余的 yum 存储库：

   ```
   # yum-config-manager --disable <repo_id>
   ```

   或者，禁用所有存储库：

   ```
    yum-config-manager --disable \*
   ```

   请注意，有大量可用存储库时可能需要花费几分钟

6. 仅启用 OpenShift Container Platform 3.11 需要的存储库：

   - 对于 x86_64 服务器中的云安装和内部安装，请运行以下命令：

   ```
   # subscription-manager repos \
       --enable="rhel-7-server-rpms" \
       --enable="rhel-7-server-extras-rpms" \
       --enable="rhel-7-server-ose-3.11-rpms" \
       --enable="rhel-7-server-ansible-2.9-rpms"
   ```

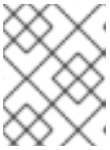
   - 对于 IBM POWER8 服务器中的内部安装，请运行以下命令：

   ```
   # subscription-manager repos \
       --enable="rhel-7-for-power-le-rpms" \
   ```

```
    --enable="rhel-7-for-power-le-extras-rpms" \
    --enable="rhel-7-for-power-le-optional-rpms" \
    --enable="rhel-7-server-ansible-2.9-for-power-le-rpms" \
    --enable="rhel-7-server-for-power-le-rhscl-rpms" \
    --enable="rhel-7-for-power-le-ose-3.11-rpms"
```

- 对于 IBM POWER9 服务器中的内部安装，请运行以下命令：

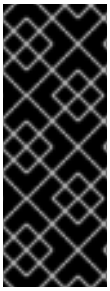
```
# subscription-manager repos \
    --enable="rhel-7-for-power-9-rpms" \
    --enable="rhel-7-for-power-9-extras-rpms" \
    --enable="rhel-7-for-power-9-optional-rpms" \
    --enable="rhel-7-server-ansible-2.9-for-power-9-rpms" \
    --enable="rhel-7-server-for-power-9-rhscl-rpms" \
    --enable="rhel-7-for-power-9-ose-3.11-rpms"
```



注意

旧版本的 OpenShift Container Platform 3.11 仅支持 Ansible 2.6。Playbook 的最新版本现在支持 Ansible 2.9，这是首选的版本。

3.7. 安装基本软件包



重要

如果您的主机使用 RHEL 7.5，且您想要接受 OpenShift Container Platform 的默认 docker 配置（使用 OverlayFS 存储和所有默认日志选项），请不要手动安装这些软件包。安装过程中运行 prerequisites.yml playbook 时会安装这些软件包。

如果您的主机使用 RHEL 7.4，或者使用 RHEL 7.5 且您要自定义 docker 配置，请安装这些软件包。

对于 RHEL 7 系统：

1. 安装以下基本软件包：

```
# yum install wget git net-tools bind-utils yum-utils iptables-services bridge-utils bash-completion kexec-tools sos psacct
```

2. 将系统更新至最新的软件包：

```
# yum update
# reboot
```

3. 安装您要使用的安装方法所需的软件包：

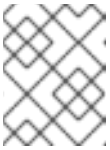
- 如果计划使用 容器化安装程序，请安装以下软件包：

```
# yum install atomic
```

- 如果计划使用 基于 RPM 的安装程序，请安装以下软件包：

```
# yum install openshift-ansible
```

-



注意

如果在运行 yum install openshift-ansible 命令时遇到错误，请查看解决方案来修复此错误。

这个软件包提供安装程序实用程序，并拉取集群安装过程所需的其他软件包，如 Ansible、playbook 和相关配置文件

对于 RHEL Atomic Host 7 系统：

1. 如果有可用的 Atomic 树，请升级到最新的 Atomic 树以确保主机已更新：

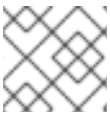
```
# atomic host upgrade
```

2. 升级完成后，并为下次引导做好准备，重启主机：

```
# reboot
```

3.8. 安装 DOCKER

此时会在所有 master 和节点主机上安装 Docker。这允许您在安装 OpenShift Container Platform 前配置 Docker 存储选项。



注意

集群安装过程自动修改 */etc/sysconfig/docker* 文件。

对于 RHEL 7 系统：

1. 安装 Docker 1.13:

```
# yum install docker-1.13.1
```

2. 验证 1.13 版本是否已安装：

```
# rpm -V docker-1.13.1
# docker version
```

对于 RHEL Atomic Host 7 系统：

不需要操作。Docker 被默认安装、配置并运行。

3.9. 配置 DOCKER 存储

容器及其创建的镜像存储在 Docker 的存储后端中。这个存储是临时的，与分配用来满足应用程序需要的持久性存储分开。使用 *临时（Ephemeral）存储*时，容器保存的数据会在容器被删除时丢失。使用 *持久性存储*时，如果容器被删除,容器保存的数据会保留。

您必须为所有 master 和节点主机配置存储，因为默认每个系统都运行容器守护进程。对于容器化安装，您需要在 master 上存储。另外，默认情况下，需要存储的 Web 控制台和 etcd 在 master 上的容器中运行。容器在节点上运行，它们始终需要存储。

存储的大小取决于工作负载、容器数量、运行的容器大小以及容器的存储要求。

重要

如果您的主机使用 RHEL 7.5，且您想要接受 OpenShift Container Platform 的默认 docker 配置（使用 OverlayFS 存储和所有默认日志选项），请不要手动安装这些软件包。安装过程中运行 prerequisites.yml playbook 时会安装这些软件包。

如果您的主机使用 RHEL 7.4，或者使用 RHEL 7.5 且您要自定义 docker 配置，请安装这些软件包。

对于 RHEL 7 系统：

RHEL 7 上 Docker 的默认存储后端是回送设备的一个精简池，在生产环境中不被支持，仅适用于概念验证环境。对于生产环境，您必须创建一个精简池逻辑卷，并重新配置 Docker 来使用这个卷。

Docker 将镜像和容器存储在图形驱动程序中，它是一个可插入的存储技术，如 DeviceMapper、OverlayFS 和 Btrfs。它们都有各自的优缺点。例如，OverlayFS 在启动和停止容器时比 DeviceMapper 快，但因为联合文件系统的架构限制，OverlayFS 无法兼容 Unix（POSIX）的可移植操作系统接口。有关在您的 RHEL 版本中使用 OverlayFS 的详情，请查看 Red Hat Enterprise Linux 发行注记。

有关 DeviceMapper 和 OverlayFS 的优点和局限性的更多信息，请参阅 Choosing a Graph Driver。

对于 RHEL Atomic Host 7 系统：

RHEL Atomic Host 上 Docker 的默认存储后端是一个精简逻辑卷，在生产环境中被支持。您必须确保根据系统要求中提到的 Docker 存储要求为这个卷分配足够空间。

如果您没有足够的空间，请参阅使用 Docker 格式化容器管理存储，以了解有关使用 docker-storage-setup 和 RHEL Atomic Host 中存储管理的基本说明。

3.9.1. 配置 OverlayFS

OverlayFS 是一种联合文件系统。OverlayFS 允许您在另一个文件系统上覆盖一个文件系统。更改记录在上面的文件系统中，而较小的文件系统则未修改。

比较 Overlay 和 Overlay2 Graph Drivers提供了更多与 overlay 和 overlay2 驱动程序相关的信息。

要使用 overlay2 驱动程序，下层必须使用 XFS 文件系统。较低层文件系统是保持未修改的文件系统。

有关为 Docker 服务启用 OverlayFS 存储驱动程序的详情，请查看 Red Hat Enterprise Linux Atomic Host 文档。

3.9.2. 配置精简池存储

您可以使用 Docker 中包含的 docker-storage-setup 脚本来创建精简池设备并配置 Docker 的存储驱动程序。安装 Docker 后，可以进行此操作，且必须在创建镜像或容器前进行此操作。该脚本从 */etc/sysconfig/docker-storage-setup* 文件中读取配置选项，并支持 3 个创建逻辑卷的选项：

- 使用附加块设备。
- 使用一个已存在的指定的卷组。
- 使用 root 文件系统所在卷组中的剩余空间。

使用附加块设备是最可靠的选项，但需要在配置 Docker 存储前向您的主机添加另一个块设备。其他选项都需要在置备主机时保留可用空间。使用 root 文件系统卷组中剩余的可用空间会导致一些应用程序出现问题，例如 Red Hat Mobile Application Platform（RHMAP）。

1. 使用以下三个选项之一创建 docker-pool 卷：

- 使用附加块设备：

a. 在 */etc/sysconfig/docker-storage-setup* 中，将 DEVS 设置为要使用的块设备的路径。将 VG 设置为要创建的卷组名称，如 docker-vg。例如：

```
# cat <<EOF > /etc/sysconfig/docker-storage-setup
DEVS=/dev/vdc
VG=docker-vg
EOF
```

b. 运行 docker-storage-setup 并查看输出以确保创建 docker-pool 卷：

```
# docker-storage-setup
[5/1868]
0
Checking that no-one is using this disk right now ...
OK

Disk /dev/vdc: 31207 cylinders, 16 heads, 63 sectors/track
sfdisk:  /dev/vdc: unrecognized partition table type

Old situation:
sfdisk: No partitions found

New situation:
Units: sectors of 512 bytes, counting from 0

   Device Boot    Start       End   #sectors  Id  System
/dev/vdc1         2048  31457279   31455232  8e  Linux LVM
/dev/vdc2            0         -          0   0  Empty
/dev/vdc3            0         -          0   0  Empty
/dev/vdc4            0         -          0   0  Empty
Warning: partition 1 does not start at a cylinder boundary
Warning: partition 1 does not end at a cylinder boundary
Warning: no primary partition is marked bootable (active)
This does not matter for LILO, but the DOS MBR will not boot this disk.
Successfully wrote the new partition table

Re-reading the partition table ...

If you created or changed a DOS partition, /dev/foo7, say, then use dd(1)
to zero the first 512 bytes:  dd if=/dev/zero of=/dev/foo7 bs=512 count=1
(See fdisk(8).)
  Physical volume "/dev/vdc1" successfully created
  Volume group "docker-vg" successfully created
  Rounding up size to full physical extent 16.00 MiB
  Logical volume "docker-poolmeta" created.
  Logical volume "docker-pool" created.
  WARNING: Converting logical volume docker-vg/docker-pool and docker-
vg/docker-poolmeta to pool's data and metadata volumes.
```

```
  THIS WILL DESTROY CONTENT OF LOGICAL VOLUME (filesystem etc.)
  Converted docker-vg/docker-pool to thin pool.
  Logical volume "docker-pool" changed.
```

- **使用现有的、指定的卷组：**

  a. **在 *etc/sysconfig/docker-storage-setup* 中，将 VG 设置为卷组。例如：**

```
# cat <<EOF > /etc/sysconfig/docker-storage-setup
VG=docker-vg
EOF
```

  b. **然后运行 docker-storage-setup 并查看输出以确保创建 docker-pool 卷：**

```
# docker-storage-setup
  Rounding up size to full physical extent 16.00 MiB
  Logical volume "docker-poolmeta" created.
  Logical volume "docker-pool" created.
  WARNING: Converting logical volume docker-vg/docker-pool and docker-
vg/docker-poolmeta to pool's data and metadata volumes.
  THIS WILL DESTROY CONTENT OF LOGICAL VOLUME (filesystem etc.)
  Converted docker-vg/docker-pool to thin pool.
  Logical volume "docker-pool" changed.
```

- **使用您的根文件系统所在卷组中的剩余空间：**

  a. **验证 root 文件系统所在的卷组是否有所需可用空间，然后运行 docker-storage-setup 并查看输出以确保创建了 docker-pool 卷：**

```
# docker-storage-setup
  Rounding up size to full physical extent 32.00 MiB
  Logical volume "docker-poolmeta" created.
  Logical volume "docker-pool" created.
  WARNING: Converting logical volume rhel/docker-pool and rhel/docker-poolmeta to
pool's data and metadata volumes.
  THIS WILL DESTROY CONTENT OF LOGICAL VOLUME (filesystem etc.)
  Converted rhel/docker-pool to thin pool.
  Logical volume "docker-pool" changed.
```

2. **验证您的配置。确认 *etc/sysconfig/docker-storage* 文件有 dm.thinpooldev 和 docker-pool 逻辑卷值：**

```
# cat /etc/sysconfig/docker-storage
DOCKER_STORAGE_OPTIONS="--storage-driver devicemapper --storage-opt dm.fs=xfs --
storage-opt dm.thinpooldev=/dev/mapper/rhel-docker--pool --storage-opt
dm.use_deferred_removal=true --storage-opt dm.use_deferred_deletion=true "

# lvs
  LV          VG   Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  docker-pool rhel twi-a-t---  9.29g             0.00   0.12
```

重要

在使用 Docker 或 OpenShift Container Platform 前，请验证 docker-pool 逻辑卷是否足够大，以满足您的需要。使 docker-pool 卷 60% 成为可用卷组，它将通过 LVM 监控来充满卷组。

3. 启动或重启 Docker。

- **如果 Docker 从未在主机上运行，请启用并启动该服务，然后验证它是否正在运行：**

```
# systemctl enable docker
# systemctl start docker
# systemctl is-active docker
```

- **如果 Docker 已在运行：**

a. **重新初始化 Docker:**

警告

这将破坏当前主机上的所有容器或镜像。

```
# systemctl stop docker
# rm -rf /var/lib/docker/*
# systemctl restart docker
```

b. **删除 */var/lib/docker/* 目录中的所有内容。**

3.9.3. 配置 Docker 存储

如果您需要在创建 docker-pool 后重新配置 Docker 存储：

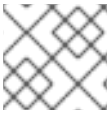
1. **删除 docker-pool 逻辑卷。**
2. **如果您使用专用卷组，请删除卷组和任何关联的物理卷。**
3. **再次运行 docker-storage-setup。**

有关 LVM 管理的详情，请查看逻辑卷管理器管理。

3.9.4. 启用镜像签名支持

OpenShift Container Platform 能够检查镜像是否来自可信源。容器安全指南介绍了相关内容。

您可以使用 atomic 命令行界面（CLI）版本 1.12.5 或更高版本来配置镜像签名验证。在 RHEL Atomic Host 系统中预先安装 atomic CLI。



注意

如需有关 atomic CLI 的更多信息，请参阅 Atomic CLI 文档。

以下文件和目录组成主机的信任配置：

- */etc/containers/registries.d/**
- */etc/containers/policy.json*

您可以直接管理每个节点上的信任配置，或者管理独立主机上的文件，或使用 Ansible 将这些文件分发到适当的节点。有关使用 Ansible 自动分发文件的实例，请参阅容器镜像签名集成指南。

1. 如果还没有安装，在主机系统中安装 atomic 软件包：

```
$ yum install atomic
```

2. 查看当前的信任配置：

```
$ atomic trust show
* (default)                    accept
```

默认配置是将所有 registry 列入白名单，这代表没有配置签名验证。

3. 自定义信任配置。在以下示例中，将一个 registry 或命名空间列入白名单，将不信任的 registry 列入黑名单（reject），这代表需要在供应商的 registry 中进行签名验证：

```
$ atomic trust add --type insecureAcceptAnything 172.30.1.1:5000

$ atomic trust add --sigstoretype atomic \
  --pubkeys pub@example.com \
  172.30.1.1:5000/production

$ atomic trust add --sigstoretype atomic \
  --pubkeys /etc/pki/example.com.pub \
  172.30.1.1:5000/production

$ atomic trust add --sigstoretype web \
  --sigstore https://access.redhat.com/webassets/docker/content/sigstore \
  --pubkeys /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release \
  registry.redhat.io

# atomic trust show
* (default)                    accept
172.30.1.1:5000                 accept
172.30.1.1:5000/production        signed security@example.com
registry.redhat.io              signed security@redhat.com,security@redhat.com
```

4. 您可以通过添加全局 reject（拒绝）默认信任来进一步强化节点：

```
$ atomic trust default reject

$ atomic trust show
* (default)                    reject
```

```
172.30.1.1:5000                 accept
172.30.1.1:5000/production      signed security@example.com
registry.redhat.io              signed security@redhat.com,security@redhat.com
```

5. （可选）查看 **atomic** man page **man atomic-trust** 来获得更多配置选项。

3.9.5. 管理容器日志

为防止容器的日志文件（容器运行在的节点上的 */var/lib/docker/containers/<hash>/<hash>-json.log* 文件）的大小增长到会出问题的大小，您可以配置 Docker 的 **json-file** 日志记录驱动程序来限制日志文件的大小和数量。

选项	用途
--log-opt max-size	设置创建新日志文件的大小。
--log-opt max-file	设置每个主机要保留的日志文件的最大数量。

1. 要配置日志文件，请编辑 */etc/sysconfig/docker* 文件。例如：要将最大文件大小设定为 1MB，并且始终保留最后三个日志文件，在 **OPTIONS=** 行中附加 **max-size=1M** 和 **max-file=3**，确保值使用单引号格式：

```
OPTIONS='--insecure-registry=172.30.0.0/16 --selinux-enabled --log-opt max-size=1M --log-opt max-file=3'
```

如需了解如何 配置日志记录驱动程序 的更多信息，请参阅 Docker 文档。

2. 重启 Docker 服务：

```
# systemctl restart docker
```

3.9.6. 查看可用的容器日志

您可以在容器运行的节点上的 */var/lib/docker/containers/<hash>/* 目录中查看容器日志。例如：

```
# ls -lh
/var/lib/docker/containers/f088349cceac173305d3e2c2e4790051799efe363842fdab5732f51f5b001fd8/

total 2.6M
-rw-r--r--. 1 root root 5.6K Nov 24 00:12 config.json
-rw-r--r--. 1 root root 649K Nov 24 00:15
f088349cceac173305d3e2c2e4790051799efe363842fdab5732f51f5b001fd8-json.log
-rw-r--r--. 1 root root 977K Nov 24 00:15
f088349cceac173305d3e2c2e4790051799efe363842fdab5732f51f5b001fd8-json.log.1
-rw-r--r--. 1 root root 977K Nov 24 00:15
f088349cceac173305d3e2c2e4790051799efe363842fdab5732f51f5b001fd8-json.log.2
-rw-r--r--. 1 root root 1.3K Nov 24 00:12 hostconfig.json
drwx------. 2 root root    6 Nov 24 00:12 secrets
```

3.9.7. 阻塞使用本地卷

当使用 *Dockerfile* 中的 **VOLUME** 指令置备卷或使用 **docker run -v <volumename>** 命令时，会使用主机的存储空间。使用这个存储会导致意外地出现存储空间的问题，并可能导致主机关闭。

在 OpenShift Container Platform 中，试图运行自己的镜像可能会导致占用完节点主机上的所有存储空间。解决这个问题的一种方法是防止用户使用卷运行镜像。这样，用户唯一可以访问的存储会被限制，集群管理员可以分配存储配额。

使用 docker-novolume-plugin 可通过禁止使用定义的本地卷启动容器来解决这个问题。特别是，插件块 **docker run** 命令包含以下内容：

- **--volumes-from** 选项
- 定义了 **VOLUME** 的镜像
- 对已存在的、由 **docker volume** 命令置备的卷的引用

该插件不会阻止对绑定挂载的引用。

要启用 docker-novolume-plugin，在每个节点主机上执行以下步骤：

1. 安装 docker-novolume-plugin 软件包：

```
$ yum install docker-novolume-plugin
```

2. 启用并启动 docker-novolume-plugin 服务：

```
$ systemctl enable docker-novolume-plugin
$ systemctl start docker-novolume-plugin
```

3. 编辑 */etc/sysconfig/docker* 文件并在 **OPTIONS** 列表中附加以下内容：

```
--authorization-plugin=docker-novolume-plugin
```

4. 重启 **docker** 服务：

```
$ systemctl restart docker
```

启用此插件后，定义了本地卷的容器无法启动并显示以下错误消息：

```
runContainer: API error (500): authorization denied by plugin
docker-novolume-plugin: volumes are not allowed
```

3.10. RED HAT GLUSTER STORAGE 软件要求

要访问 GlusterFS 卷，必须在所有可调度节点上使用 **mount.glusterfs** 命令。对于基于 RPM 的系统，必须安装 glusterfs-fuse 软件包：

```
# yum install glusterfs-fuse
```

这个软件包会在每个 RHEL 系统上安装。但是，如果您的服务器使用 x86_64 架构，则建议您将 Red Hat Gluster Storage 升级到最新的可用版本。要做到这一点，必须启用以下 RPM 存储库：

```
# subscription-manager repos --enable=rh-gluster-3-client-for-rhel-7-server-rpms
```

如果已在节点上安装了 glusterfs-fuse，请确保安装了最新版本：

```
# yum update glusterfs-fuse
```

3.11. 后续步骤

在主机准备完后，如果安装了 OpenShift Container Platform，需要配置清单文件。如果要安装独立 registry，请继续 安装独立 registry。

第 4 章 配置清单文件

4.1. 为您的集群自定义清单文件

Ansible 清单文件描述了集群中主机的详情，以及 OpenShift Container Platform 安装的集群配置详情。OpenShift Container Platform 的安装 playbook 会读取您的清单文件，了解如何在您的主机集合中安装 OpenShift Container Platform。

注意

如需有关清单文件格式的详细信息，请参阅 Ansible 文档，包括 YAML 语法的基本信息。

安装 openshift-ansible RPM 软件包时，如主机准备中所述，Ansible 依赖项会在 */etc/ansible/hosts* 的默认位置创建一个文件。但是，该文件只是默认 Ansible 示例，没有与 OpenShift Container Platform 具体配置相关的变量。要成功安装 OpenShift Container Platform，*必须*根据集群的拓扑和要求，将文件的默认内容替换为您自己的配置。

以下小节描述了在集群安装过程中在清单文件中设置的常用变量。许多 Ansible 变量都是可选的。对于开发环境中，可以使用所需参数的默认值，但在生产环境中必须为它们设置适当的值。

您可以查看 清单文件示例作为集群安装的起点。

注意

镜像需要一个版本号策略才能维护更新。如需更多信息，请参阅架构指南中的镜像版本标签策略 部分。

4.2. 配置集群变量

要在 Ansible 安装过程中分配全局集群环境变量，请将它们添加到 */etc/ansible/hosts* 文件的 **[OSEv3:vars]** 部分。您必须将每个参数值放在单独的行中。例如：

```
[OSEv3:vars]

openshift_master_identity_providers=[{'name': 'htpasswd_auth',
'login': 'true', 'challenge': 'true',
'kind': 'HTPasswdPasswordIdentityProvider',}]

openshift_master_default_subdomain=apps.test.example.com
```

重要

如果 Ansible 清单文件中的参数值包含特殊字符，如 **#**、**{** 或 **}**，则需要双重转义（以单引号和双引号包括该值）。例如，要将 **mypasswordwith###hashsigns** 作为变量 **openshift_cloudprovider_openstack_password** 的值，在 Ansible 主机清单文件中将其声明为 **openshift_cloudprovider_openstack_password='"mypasswordwith###hashsigns"'**。

下表描述了与 Ansible 安装程序搭配使用的全局集群变量：

表 4.1. 常规集群变量

变量	用途
ansible_ssh_user	这个变量为安装程序设定要使用的 SSH 用户，默认为 **root** 用户。此用户必须在 不需要密码的情况下允许基于 SSH 的身份验证。如果使用基于 SSH 密钥的身份验证，则密钥必须由 SSH 代理管理。
ansible_become	如果 **ansible_ssh_user** 不是 **root**，则此变量必须设为 **true**，且必须为此用户配置无需密码的 **sudo**。
debug_level	这个变量设定了将哪些 INFO 信息记录到 **systemd-journald.service**。设置以下内容之一： ● **0** 只记录错误（error）和警告（warning）信息 ● **2** 记录普通信息（这是默认级别） ● **4** 记录调试级别信息 ● **6** 记录 API 级调试信息（request / response） ● **8** 记录 body 级 API 调试信息 如需有关 debug 日志级别的更多信息，请参阅 配置日志记录级别。
openshift_clock_enabled	是否在集群节点上启用网络时间协议（NTP）。默认值为 **true**。 如果安装了 **chrony** 软件包，则会将其配置为提供 NTP 服务。如果没有安装 **chrony** 软件包，安装 playbook 会安装和配置 **ntp** 软件包以提供 NTP 服务。  **重要** 要防止集群中的 master 和节点不同步，请不要更改此参数的默认值。

变量	用途
openshift_master_admission_plugin_config	此变量根据清单主机文件中的要求设置参数和任意 JSON 值。例如： openshift_master_admission_plugin_config={"ClusterResourceOverride":{"configuration":{"apiVersion":"v1","kind":"ClusterResourceOverrideConfig","memoryRequestToLimitPercent":"25","cpuRequestToLimitPercent":"25","limitCPUToMemoryPercent":"200"}}} 在这个值中，**openshift_master_admission_plugin_config={"openshift.io/ImagePolicy":{"configuration":{"apiVersion":"v1","executionRules":[{"matchImageAnnotations":[{"key":"images.openshift.io/deny-execution","value":"true"}],"name":"execution-denied","onResources":[{"resource":"pods"},{"resource":"builds"}],"reject":true,"skipOnResolutionFailure":true}],"kind":"ImagePolicyConfig"}}}** 是默认的参数值。 **重要** 即使要添加自定义设置，也需要包含默认的 **openshift_master_admission_plugin_config** 值。
openshift_master_audit_config	此变量启用 API 服务审计。如需更多信息，请参阅 审计配置。
openshift_master_audit_policyfile	提供审计策略文件的位置。如需更多信息，请参阅 审计策略配置。
openshift_master_cluster_hostname	此变量覆盖集群的主机名，默认为 master 的主机名。
openshift_master_cluster_public_hostname	此变量覆盖集群的公共主机名，默认为 master 的主机名。如果您使用外部负载均衡器，指定外部负载均衡器的地址。 例如： openshift_master_cluster_public_hostname=openshift-ansible.public.example.com

变量	用途
openshift_master_cluster_method	可选。此变量定义了部署多个 master 时的 HA 方法。支持 **native** 方法。如需更多信息，请参阅多 Master。
openshift_rolling_restart_mode	此变量会在 直接运行升级 playbook 时启用 HA master 的滚动重启（例如，每次只会有一个 masters 停机）。默认为 **services**，它允许在 master 上滚动重启服务。可将其设定为 **system**，这样可启用滚动、完全重启 master 节点。 在升级过程中，可能会需要滚动重启 master，以使用提供的 Ansible hook 应用其他更改。根据您选择执行的任务而定，您可能想重启主机来重启您的服务。
openshift_master_identity_providers	此变量设置 身份提供程序。默认值为 Deny All。如果使用受支持的身份提供程序，请配置 OpenShift Container Platform 来使用它。您可以配置多个身份提供程序。
openshift_master_named_certificates	这些变量用于配置部署在安装过程中的 自定义证书。如需更多信息，请参阅配置自定义证书。
openshift_master_overwrite_named_certificates	
openshift_hosted_router_certificate	为托管路由器提供自定义证书的位置。
openshift_master_ca_certificate	提供为 OpenShift Container Platform 证书签名的单个证书和密钥。请参阅 重新部署新的或自定义 OpenShift Container Platform CA
openshift_additional_ca	如果 **openshift_master_ca_certificate** 参数的证书由中间证书签名，请提供包含 CA 中间证书和 root 证书链的捆绑证书。请参阅 重新部署新的或自定义 OpenShift Container Platform CA
openshift_redeploy_service_signer	如果 参数设置为 **false**，则运行 **openshift-master/redeploy-certificates.yml** playbook 时不会重新部署 signer。默认值为 **true**。
openshift_hosted_registry_cert_expire_days	自动生成的 registry 证书的有效性。（以天为单位）默认为 **730**（2 年）。
openshift_ca_cert_expire_days	自动生成的 CA 证书的有效期（以天为单位）。默认值为 **1825**（5 年）。
openshift_master_cert_expire_days	自动生成的 master 证书的有效期。（以天为单位）默认为 **730**（2 年）。

变量	用途
etcd_ca_default_days	自动生成的外部 etcd 证书的有效性。（以天为单位）控制 etcd CA、peer、服务器和客户端证书的有效性。默认值为 **1825**（5 年）。
openshift_certificate_expiry_warning_days	停止在这个天数内或更短的时间内过期的集群进行升级。默认值为 **365**（1 年）。
openshift_certificate_expiry_fail_on_warn	自动生成的证书在 **openshift_certificate_expiry_warning_days** 参数指定的期间无效时，升级是否失败。默认值为 **True**。
os_firewall_use_firewalld	设置为 **true** 来使用 firewalld 而不是默认的 iptables。在 RHEL Atomic Host 上不可用。如需更多信息，请参阅配置防火墙部分。
openshift_master_session_name	这些变量覆盖 OAuth 配置中的 会话选项 的默认设置。如需更多信息，请参阅配置会话选项。
openshift_master_session_max_seconds	
openshift_master_session_auth_secrets	
openshift_master_session_encryption_secrets	
openshift_master_image_policy_config	在 master 配置中设置 **imagePolicyConfig**。详情请参阅 镜像配置。
openshift_router_selector	自动部署路由器 Pod 的默认节点选择器。有关详细信息，请参阅配置节点主机标签。
openshift_registry_selector	用于自动部署 registry Pod 的默认节点选择器。有关详细信息，请参阅配置节点主机标签。
openshift_template_service_broker_namespaces	此变量通过指定由代理提供模板的一个或多个命名空间来启用模板服务代理。
openshift_master_bootstrap_auto_approve	此变量启用 TLS bootstrapping 自动批准，它允许节点在提供 bootstrap 凭证时自动加入集群。如果将在 Amazon Web Services（AWS）集群上启用cluster auto-scaler，则设置为 **true**。默认值为 **false**。
ansible_service_broker_node_selector	用于自动部署 Ansible 服务代理 Pod 的默认节点选择器，默认为 **{"node-role.kubernetes.io/infra":"true"}**。有关详细信息，请参阅配置节点主机标签。

变量	用途
osm_default_node_selector	此变量覆盖了项目在放置 pod 时默认使用的节点选择器，这由主配置文件中的 **projectConfig.defaultNodeSelector** 定义。如果未定义，则默认为 **node-role.kubernetes.io/compute=true**。
openshift_docker_additional_registries	OpenShift Container Platform 将指定的额外 registry 或 registry 添加到 **docker** 配置中。这些是要搜索的 registry。如果 registry 需要访问 **80** 以外的端口，以 **<address>:<port>** 的形式包括所需的端口号。 例如： `openshift_docker_additional_registries=example.com:443`  **注意** 如果需要将集群配置为使用备用 registry，设置 **oreg_url** 而不是依赖于 **openshift_docker_additional_registries**。
openshift_docker_insecure_registries	OpenShift Container Platform 将指定的额外不安全的 registry 添加到 **docker** 配置中。对于这些 registry，无法验证安全套接字层(SSL)。可以设置为主机的主机名或 IP 地址。**0.0.0.0/0** 不是 IP 地址的有效设置。
openshift_docker_blocked_registries	OpenShift Container Platform 将指定的受阻 registry 添加到 **docker** 配置中。阻止列出的 registry。把它设置为 **all** 会阻止没有包括在其他变量中的所有项。
openshift_docker_ent_reg	当 **openshift_deployment_type** 设置为 **openshift-enterprise** 时，容器运行时信任的额外 registry 。默认为 **registry.redhat.io**。如果设置 **openshift_docker_ent_reg=''**，则 **registry.redhat.io** 不会被添加到 **docker** 配置中。
openshift_metrics_hawkular_hostname	此变量通过覆盖集群指标主配置中的 **metricsPublicURL** 来设置与指标控制台集成的主机名。如果更改了这个变量，请确保主机名可以通过路由器访问。
openshift_clusterid	此变量是 AWS Availability Zone 中唯一的集群标识符。使用这个功能可避免在带有多个区或多个集群的 Amazon Web Services（AWS）中的潜在问题。如需了解详细信息，请参阅 为 AWS 标记集群。

变量	用途
openshift_encryption_config	使用此变量配置数据存储层加密。
openshift_image_tag	使用此变量指定要安装或配置的容器镜像标签。
openshift_pkg_version	使用这个变量指定 RPM 版本来安装或配置。

警告

如果在集群设置后修改 openshift_image_tag 或 openshift_pkg_version 变量，则可能会触发升级过程，从而导致停机。

- **如果设置了 openshift_image_tag，则其值将用于系统容器中的所有主机，包括安装了另一个版本的主机。如果**
- **设定了openshift_pkg_version，其值用于基于 RPM 的环境中的所有主机，包括安装了另一个版本的主机。**

表 4.2. 网络变量

变量	用途
openshift_master_default_subdomain	此变量覆盖用于公开 路由 的默认子域。此变量的值必须包含小写字母数字字符或横线**(-)**。它必须以字母字符开头，并以字母数字字符结尾。
os_sdn_network_plugin_name	此变量配置了将哪些 OpenShift SDN 插件 用于 pod 网络。对于标准 SDN 插件，默认为 **redhat/openshift-ovs-subnet**。将变量设置为 **redhat/openshift-ovs-multitenant** 以使用多租户 SDN 插件。
osm_cluster_network_cidr	此变量覆盖 SDN 集群网络 CIDR 块。这是分配 Pod IP 的网络。指定与基础架构中现有网络块没有冲突的私有块，pod、节点或 master 需要访问它。默认为 **10.128.0.0/14**，且在部署后无法任意重新配置，但可在 SDN master 配置中进行某些更改。
openshift_portal_net	此变量配置子网，services 会在 OpenShift Container Platform SDN 的这个子网中创建。指定与基础架构中任何现有网络块没有冲突的私有块， pod、节点或 master 需要可以访问它，否则安装会失败。默认为 **172.30.0.0/16**，部署后无法重新配置。如果改变默认的值，需要避免使用 **172.17.0.0/16**（因为 **docker0** 网桥会默认使用它）或修改 **docker0** 网络。

变量	用途
osm_host_subnet_length	此变量指定 OpenShift Container Platform SDN为 pod IP 分配的每个主机子网的大小。默认为 **9**，表示每个主机分配了 /23 大小的子网 ; 例如，如果默认为 10.128.0.0/14 集群网络，则会分配 10.128.0.0/23、10.128.2.0/23、10.128.4.0/23 等。这在部署后无法重新配置。
openshift_node_proxy_mode	这个变量指定要使用的 服务代理模式:默认为 **iptables**（纯 **iptables** 的实施），或 **userspace** 用于用户空间代理。
openshift_use_flannel	这个变量启用 **flannel** 作为替代网络层而不是默认的 SDN。如果启用 **flannel**，则使用 **openshift_use_openshift_sdn** 变量禁用默认 SDN。如需更多信息，请参阅 使用 Flannel。
openshift_use_openshift_sdn	设置为 **false** 以禁用 OpenShift SDN 插件。
openshift_sdn_vxlan_port	此变量设置 **cluster 网络**的 **vxlan 端口**。默认为 **4789**。如需更多信息请参阅为集群网络更改 VXLAN PORT。
openshift_node_sdn_mtu	此变量指定用于 OpenShift SDN 的 MTU 大小。该值必须小于节点的主网络接口的 MTU **50** 字节。例如，如果主网络接口的 MTU 为 **1500**，则这个值将是 **1450**。默认值为 **1450**。

4.3. 配置部署类型

安装程序在 playbook 和角色中所使用的各种默认值是基于部署类型配置（通常在 Ansible 清单文件中定义）。

确保将清单文件 [OSEv3:vars] 部分中的 openshift_deployment_type 参数设置为 openshift-enterprise 以安装 OpenShift Container Platform 变体：

```
[OSEv3:vars]
openshift_deployment_type=openshift-enterprise
```

4.4. 配置主机变量

要在 Ansible 安装过程中为主机分配环境变量，在 *./etc/ansible/hosts* 中的[masters] 或 [nodes] 项中的 host 条目的后面设置。例如：

```
[masters]
ec2-52-6-179-239.compute-1.amazonaws.com openshift_public_hostname=ose3-
master.public.example.com
```

下表描述了可用于 Ansible 安装程序的变量，可分配给各个 host 条目：

表 4.3. Host 变量

变量	用途
openshift_public_hostname	该变量覆盖系统的公共主机名。在云安装中、或使用网络地址转换（NAT）的网络主机中使用它。
openshift_public_ip	这个变量会覆盖系统的公共 IP 地址。在云安装中、或使用网络地址转换（NAT）的网络主机中使用它。
openshift_node_labels	此变量已弃用，请参阅 定义节点组和主机映射了解当前设置节点标签的方法。
openshift_docker_options	此变量在 */etc/sysconfig/docker* 中配置额外的 **docker** 选项，如管理容器日志中使用的选项。建议您使用 **json-file**。 以下示例显示了使用 **json-file** 日志驱动程序的 Docker 配置，Docker 在三个 1 MB 日志文件中轮转，且不需要签名验证。提供额外选项时，请确定您使用单引号格式： `OPTIONS=' --selinux-enabled --log-opt max-size=1M --log-opt max-file=3 --insecure-registry 172.30.0.0/16 --log-driver=json-file --signature-verification=false'`
openshift_schedulable	此变量配置主机是否标记为可调度节点，即是否可以放置新 pod。请参阅在 Master 中配置调度。
openshift_node_problem_detector_install	此变量用于激活 节点问题检测程序（Node Problem Detector）。如果设置为 **false, the default**， Node Problem Detector 不会被安装或启动。

4.5. 定义节点组和主机映射

节点配置从 master 启动。当节点启动和服务时，节点会在加入集群前检查 *kubeconfig* 和其他节点配置文件是否存在。如果不存在，节点将从 master 中拉取配置，然后加入集群。

通过使用这个方法，管理员不需要在每个节点主机上手动维护节点配置。节点主机的 */etc/origin/node/node-config.yaml* 文件的内容由来自 master 的 ConfigMaps 提供。

4.5.1. 节点 ConfigMap

用于定义节点配置的 Configmap 必须在 openshift-node 项目中可用。现在，ConfigMap 也是节点标签的权威定义，旧的 openshift_node_labels 值会被实际忽略。

默认情况下，安装程序会在集群安装过程中创建以下默认 ConfigMap:

- **node-config-master**

- **node-config-infra**
- **node-config-compute**

另外还会创建以下 ConfigMap，将节点标记为多个角色：

- **node-config-all-in-one**
- **node-config-master-infra**

以下 ConfigMap 是每个现有默认节点组的 CRI-O 变体：

- **node-config-master-crio**
- **node-config-infra-crio**
- **node-config-compute-crio**
- **node-config-all-in-one-crio**
- **node-config-master-infra-crio**

重要

您不能修改节点主机的 *./etc/origin/node/node-config.yaml* 文件。任何更改都会被节点使用的 ConfigMap 中定义的配置覆盖。

4.5.2. 节点组定义

安装最新的 openshift-ansible 软件包后，您可以在 *./usr/share/ansible/openshift-ansible/roles/openshift_facts/defaults/main.yml* 文件中以 YAML 格式查看默认节点组定义集：

```
openshift_node_groups:
  - name: node-config-master (1)
    labels:
      - 'node-role.kubernetes.io/master=true' (2)
    edits: [] (3)
  - name: node-config-infra
    labels:
      - 'node-role.kubernetes.io/infra=true'
    edits: []
  - name: node-config-compute
    labels:
      - 'node-role.kubernetes.io/compute=true'
    edits: []
  - name: node-config-master-infra
    labels:
      - 'node-role.kubernetes.io/infra=true,node-role.kubernetes.io/master=true'
    edits: []
  - name: node-config-all-in-one
    labels:
      - 'node-role.kubernetes.io/infra=true,node-role.kubernetes.io/master=true,node-
role.kubernetes.io/compute=true'
    edits: []
```

1. 节点组名称。
2. 与节点组关联的节点标签列表。有关详细信息，请参阅节点主机标签。
3. 任何对节点组配置的编辑。

如果您没有在清单文件中的 [OSEv3:vars] 组中设置 openshift_node_groups 变量，则会使用这些默认值。但是，如果要设置自定义节点组，则必须在清单文件中定义整个 openshift_node_groups 结构，包括所有计划的节点组。

openshift_node_groups 值没有与默认值合并，您必须将 YAML 定义转换为 Python 字典。然后，您可以使用 edits 字段指定键值对来修改任何节点配置变量。

注意

有关可配置节点变量的参考，请参阅Master 和节点配置文件。

例如，以下清单文件中的条目定义了名为 node-config-master、node-config-infra 和 node-config-compute 的组。

```
openshift_node_groups=[{'name': 'node-config-master', 'labels': ['node-
role.kubernetes.io/master=true']}, {'name': 'node-config-infra', 'labels': ['node-
role.kubernetes.io/infra=true']}, {'name': 'node-config-compute', 'labels': ['node-
role.kubernetes.io/compute=true']}]
```

您还可以使用其他标签定义新节点组名称，清单文件中的以下条目定义了名为 node-config-master、node- config-infra、node- config-compute 和 node-config-compute-storage 的组。

```
openshift_node_groups=[{'name': 'node-config-master', 'labels': ['node-
role.kubernetes.io/master=true']}, {'name': 'node-config-infra', 'labels': ['node-
role.kubernetes.io/infra=true']}, {'name': 'node-config-compute', 'labels': ['node-
role.kubernetes.io/compute=true']}, {'name': 'node-config-compute-storage', 'labels': ['node-
role.kubernetes.io/compute-storage=true']}]
```

当在清单文件中设置条目时，您还可编辑节点组的 ConfigMap:

- 您可以使用列表，如修改的 node-config-compute 将 kubeletArguments.pods-per-core 设置为 20:

```
openshift_node_groups=[{'name': 'node-config-master', 'labels': ['node-
role.kubernetes.io/master=true']}, {'name': 'node-config-infra', 'labels': ['node-
role.kubernetes.io/infra=true']}, {'name': 'node-config-compute', 'labels': ['node-
role.kubernetes.io/compute=true'], 'edits': [{ 'key': 'kubeletArguments.pods-per-core','value': ['20']}]}]
```

- 您可以使用列表来修改多个键值对，例如修改 node-config-compute 组，为 kubelet 添加两个参数：

```
openshift_node_groups=[{'name': 'node-config-master', 'labels': ['node-
role.kubernetes.io/master=true']}, {'name': 'node-config-infra', 'labels': ['node-
role.kubernetes.io/infra=true']}, {'name': 'node-config-compute', 'labels': ['node-
role.kubernetes.io/compute=true'], 'edits': [{ 'key': 'kubeletArguments.experimental-allocatable-ignore-
eviction','value': ['true']}, {'key': 'kubeletArguments.eviction-hard', 'value': ['memory.available<1Ki']}]}]
```

- 您还可以使用字典作为值，如修改 node-config-compute 组把 perFSGroup 设置为 512Mi

```
openshift_node_groups=[{'name': 'node-config-master', 'labels': ['node-
role.kubernetes.io/master=true']}, {'name': 'node-config-infra', 'labels': ['node-
role.kubernetes.io/infra=true']}, {'name': 'node-config-compute', 'labels': ['node-
role.kubernetes.io/compute=true'], 'edits': [{ 'key': 'volumeConfig.localQuota','value':
{'perFSGroup':'512Mi'}}]}]
```

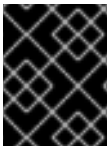
每当运行 *openshift_node_group.yml* playbook 时，在 edits 字段中定义的更改会更新相关的 ConfigMap（本例中的node-config-compute），这最终将影响主机上的节点配置文件。

注意

运行 *openshift_node_group.yaml* playbook 只会更新新节点。无法运行它来更新集群中的现有节点。

4.5.3. 将主机映射到节点组

要把 ConfigMap 映射到节点，在清单的 [nodes] 组中定义的所有主机都需要使用 openshift_node_group_name 变量分配到一个 *节点组*。



重要

对于所有集群安装，无论是使用默认节点组定义和 ConfigMap 还是自行自定义，都需要为每个主机将 openshift_node_group_name 设置为节点组。

openshift_node_group_name 的值用于选择配置每个节点的 ConfigMap。例如：

```
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'
node1.example.com openshift_node_group_name='node-config-compute'
node2.example.com openshift_node_group_name='node-config-compute'
```

如果在 openshift_node_groups 中定义了其他自定义 ConfigMap，它们也可以被使用。例如：

```
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'
node1.example.com openshift_node_group_name='node-config-compute'
node2.example.com openshift_node_group_name='node-config-compute'
gluster[1:6].example.com openshift_node_group_name='node-config-compute-storage'
```

4.5.4. 节点主机标签

您可以在集群安装过程中为节点主机分配 标签（Label）。通过标签，调度程序（scheduler）可以决定 pod 放置的节点位置。

如果要修改分配给节点主机的默认标签，必须创建自己的自定义节点组。您无法再设置 openshift_node_labels 变量来更改标签。请参阅 节点组定义来修改默认节点组。

除 node-role.kubernetes.io/infra=true （使用此组的主机也称为 *专用基础架构节点*，在 配置 Dedicated Infrastructure 节点一节中会进一步讨论） 外，其他实际标签名称和值都是任意的并可随意分配，但最好根据集群的要求进行分配。

4.5.4.1. 主 pod 调度功能

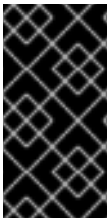
配置在安装过程中指定为 master 的所有主机。通过这样做，masters 会被配置为 OpenShift SDN 的一部分。您必须将 master 主机的条目添加到 [nodes] 部分：

```
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master'
```

如果要在安装后更改主机的调度性，请参阅 将节点标记为 Unschedulable 或 Schedulable。

4.5.4.2. 节点上的 Pod 调度功能

默认情况下，masters 标记为可调度节点，在集群安装过程中会默认设置默认节点选择器。master 配置文件的 projectConfig.defaultNodeSelector 字段中定义了默认节点选择器，以确定在放置 pod 时将默认使用哪些节点项目。除非使用 osm_default_node_selector 变量覆盖，否则会设置为 node-role.kubernetes.io/compute=true。



重要

如果您在安装过程中接受 node-role.kubernetes.io/compute=true 的默认节点选择器，请确保在集群中定义了非 master 节点时具有除专用基础架构节点之外的其他节点。否则，应用程序 Pod 会无法部署。这是因为在为项目调度 pod 时，没有可用的具有 node-role.kubernetes.io/compute=true 标签的节点与默认节点选择器匹配。

如需了解调整安装后设置此设置的步骤，请参阅 设置集群范围默认 Node Selector。

4.5.4.3. 配置 Dedicated Infrastructure 节点

建议您在生产环境中使用专用基础架构节点来运行 registry 和路由器 Pod，这样可以使它们独立于用于用户应用程序的 Pod。

openshift_router_selector 和 openshift_registry_selector Ansible 设置决定了在放置 registry 和路由器 Pod 时使用的标签选择器。默认情况下，它们被设置为 node-role.kubernetes.io/infra=true:

```
# default selectors for router and registry services
# openshift_router_selector='node-role.kubernetes.io/infra=true'
# openshift_registry_selector='node-role.kubernetes.io/infra=true'
```

registry 和路由器只能在使用 node-role.kubernetes.io/infra=true 标签的节点主机上运行，这些标签被视为专用基础架构节点。确保 OpenShift Container Platform 环境中至少有一个节点主机具有 node-role.kubernetes.io/infra=true 标签。您可以使用默认 node-config-infra 设置这个标签：

```
[nodes]
infra-node1.example.com openshift_node_group_name='node-config-infra'
```

重要

如果 [nodes] 部分中没有与选择器设置匹配的节点，则默认路由器和 registry 将部署失败并带有 Pending 状态。

如果您不打算使用 OpenShift Container Platform 管理 registry 和路由器，请配置以下 Ansible 设置：

```
openshift_hosted_manage_registry=false
openshift_hosted_manage_router=false
```

如果使用默认 registry.redhat.io 以外的镜像 registry，则必须在 */etc/ansible/hosts* 文件中 指定 registry。

如在 Master 中配置调度功能所述，master 主机默认标记为可调度。如果标记了带有 node-role.kubernetes.io/infra=true 的 master 主机，且没有其他专用基础架构节点，则 master 主机还必须标记为可以调度。否则，registry 和路由器 Pod 将无法放置到任何节点。

您可以使用默认的 node-config-master-infra 节点组来实现这一目的：

```
[nodes]
master.example.com openshift_node_group_name='node-config-master-infra'
```

4.6. 配置项目参数

要配置默认项目设置，请在 */etc/ansible/hosts* 文件中配置以下变量：

表 4.4. 项目参数

参数	描述	类型	默认值
osm_project_request_message	当用户无法通过 **projectrequest** API 端点请求一个项目时，为用户显示的字符串。	字符串	null
osm_project_request_template	响应 **projectrequest** 以创建项目时使用的模板。如果没有指定，则使用默认模板。	带有 **<namespace>/<template>** 格式的字符串	null
osm_mcs_allocator_range	定义分配给命名空间的 MCS 类别范围。如果此值在启动后改变，新项目可能会收到已经分配给其他项目的标签。前缀可以是任意有效的 SELinux 术语集合，包括 user、role 和 type但是，使用默认前缀可让服务器自动设置它们。例如：**s0:/2** 分配 s0:c0,c0 到 s0:c511,c511 的标签，**s0:/2,512** 分配从 s0:c0,c0,c0 到 s0:c511,c511,511 的标签。	格式为 **<prefix>/<numberOfLabels>[,<maxCategory>]** 的字符串	**s0:/2**

参数	描述	类型	默认值
osm_mcs_labels_per_project	定义每个项目要保留的标签数。	整数	**5**
osm_uid_allocator_range	定义自动分配给项目的总 Unix 用户 ID（UID）以及每个命名空间获取的块的大小。例如，**1000-1999/10** 代表为每个命名空间分配十个 UID，并在耗尽前最多分配 100 个块。默认值是用户命名空间启动时容器镜像的预期范围大小。	格式为 **<block_range>/<number_of_UIDs>** 的字符串。	**1000000000-1999999999/10000**

4.7. 配置主 API 端口

要配置 master API 使用的默认端口，在 */etc/ansible/hosts* 文件中配置以下变量：

表 4.5. Master API 端口

变量	用途
openshift_master_api_port	此变量设置了用于访问 OpenShift Container Platform API 的端口号。

例如：

```
openshift_master_api_port=3443
```

Web 控制台端口设置（openshift_master_console_port）必须与 API 服务器端口（openshift_master_api_port）匹配。

4.8. 配置集群预安装检查

预安装检查是一组作为 openshift_health_checker Ansible 角色一部分运行的诊断任务。它们在 OpenShift Container Platform 安装 Ansible 之前运行，确保设置了所需的清单值，并识别主机上可能会阻止或影响成功安装的潜在问题。

下表描述了在每次 OpenShift Container Platform 安装 Ansible 之前运行的可用的预安装检查：

表 4.6. 预安装检查

检查名称	用途

检查名称	用途
memory_availability	此检查可确保主机具有为 OpenShift Container Platform 的特定部署提供推荐的内存量。默认值可从最新的安装文档中获得。用户定义的用于最小内存要求的值，可通过在清单文件中设置 **openshift_check_min_host_memory_gb** 集群变量进行设置。
disk_availability	此检查只会在 etcd、master 和节点主机上运行。它确保 OpenShift Container Platform 安装的挂载路径有足够的磁盘空间。推荐的磁盘值请参考最新的安装文档。用户定义的用于最小磁盘要求的值，可通过在清单文件中设置 **openshift_check_min_host_disk_gb** 集群变量进行设置。
docker_storage	仅在依赖于 **docker** 守护进程的主机上运行（节点和系统容器安装）。检查 **docker** 的总用量没有超过用户定义的限制。如果没有设置用户自定义的限制，则 **docker** 的最大用量阈值默认为可用总大小的 90%。使用总百分比的阈值可通过清单文件中的变量来设置：**max_thinpool_data_usage_percent=90**。用户定义的用于最大 thinpool 使用限制的值，可通过设置清单文件中的 **max_thinpool_data_usage_percent** 集群变量进行设置。
docker_storage_driver	确保 **docker** 守护进程使用 OpenShift Container Platform 支持的存储驱动程序。如果使用 **devicemapper** 存储驱动程序，这个检查还会检查是否没有使用一个 loopback 设备。如需更多信息，请参阅 Docker 使用设备映射器存储驱动程序指南。
docker_image_availability	尝试确保 OpenShift Container Platform 安装所需的镜像在本地或主机上至少一个配置的容器镜像 registry 中可用。
package_version	在基于 **yum** 的系统上运行，确定所需的 OpenShift Container Platform 软件包有多个可用版本。在一个**企业级** OpenShift 安装过程中提供多个软件包发行版本表示在不同版本中启用了多个 **yum** 存储库，这可能会导致安装问题。
package_availability	在 OpenShift Container Platform 的 RPM 安装前运行。确保当前安装所需的 RPM 软件包可用。
package_update	检查 **yum** 更新或软件包安装是否会成功，而并不实际执行更新或升级，或在主机上运行 **yum**。

要禁用特定的预安装检查，请在清单文件中使用逗号分隔的检查名称列表包括 openshift_disable_check 变量。例如：

```
openshift_disable_check=memory_availability,disk_availability
```

注意

更多信息，请参阅基于 Ansible 的健康检查。关于检查证书过期的信息，请参阅重新部署证书。

4.9. 配置 REGISTRY 位置

如果您在 registry.redhat.io 中使用默认 registry，则必须设置以下变量：

```
oreg_url=registry.redhat.io/openshift3/ose-${component}:${version}
oreg_auth_user="<user>"
oreg_auth_password="<password>"
```

有关设置 registry 访问令牌的更多信息，请参阅 Red Hat Container Registry 身份验证。

如果您使用 registry.redhat.io 以外的默认镜像 registry，请在 */etc/ansible/hosts* 文件中指定 registry。

```
oreg_url=example.com/openshift3/ose-${component}:${version}
openshift_examples_modify_imagestreams=true
```

表 4.7. registry 变量

变量	用途
oreg_url	设置备用镜像位置。如果您没有使用默认的 **registry.redhat.io**，则需要设置它。默认组件从 **oreg_url** 值继承镜像前缀和版本。对于需要身份验证的默认 registry，需要指定 **oreg_auth_user** 和 **oreg_auth_password**。
openshift_examples_modify_imagestreams	如果指向默认 registry 以外的 registry，则设置为 **true**。将镜像流位置改为 **oreg_url**。
oreg_auth_user	如果 **oreg_url** 指向需要身份验证的 registry，使用 **oreg_auth_user** 变量来提供您的用户名。您还必须提供密码作为 **oreg_auth_password** 的参数值。如果使用默认 registry，请指定可访问 **registry.redhat.io** 的用户。
oreg_auth_password	如果 **oreg_url** 指向需要身份验证的 registry，使用 **oreg_auth_password** 变量来提供您的密码。还必须将您的用户名作为 **oreg_auth_user** 参数值提供。如果使用默认 registry，指定该用户的密码或令牌。

注意

默认 registry 需要身份验证令牌。如需更多信息，请参阅访问并配置 Red Hat Registry

例如：

```
oreg_url=example.com/openshift3/ose-${component}:${version}
oreg_auth_user=${user_name}
oreg_auth_password=${password}
openshift_examples_modify_imagestreams=true
```

4.10. 配置 REGISTRY 路由

要允许用户从 OpenShift Container Platform 集群以外将镜像推送或拉取到内部容器镜像 registry，在 *`/etc/ansible/hosts`* 文件中配置 registry 路由。默认情况下，registry 路由是*docker-registry-default.router.default.svc.cluster.local*。

表 4.8. registry 路由变量

变量	用途
openshift_hosted_registry_routehost	设置为所需 registry 路由的值。该路由包含一个解析到路由器管理通信的基础架构节点的名称，或您设置为默认应用程序子域通配符值的子域。例如，如果您将 **openshift_master_default_subdomain** 参数设置为 **apps.example.com**，**.apps.example.com** 解析到基础架构节点或一个负载均衡器，则可以使用 **registry.apps.example.com** 作为 registry 路由。
openshift_hosted_registry_routecertificates	设置 registry 证书的路径。如果没有为证书位置提供值，则会生成证书。您可以定义以下证书的位置： • **certfile** • **keyfile** • **cafile**
openshift_hosted_registry_routetermination	设置为以下值之一： • 设置为 **reencrypt** 会在边缘路由器中终止加密，并使用目的地提供的新证书重新加密。 • 设置为 **passthrough** 会在目的地终止加密。目的地负责对数据进行解密。

例如：

```
openshift_hosted_registry_routehost=<path>
openshift_hosted_registry_routetermination=reencrypt
openshift_hosted_registry_routecertificates= "{'certfile': '<path>/org-cert.pem', 'keyfile': '<path>/org-
```

```
privkey.pem', 'cafile': '<path>/org-chain.pem'}"
```

4.11. 配置路由器分片

通过向清单提供正确的数据启用路由器分片 支持。变量 **openshift_hosted_routers** 包含数据，它是一个列表。如果没有传递数据，则创建一个默认路由器。路由器分片有多种组合。以下示例支持独立节点上的路由器：

```
openshift_hosted_routers=[{'name': 'router1', 'certificate': {'certfile': '/path/to/certificate/abc.crt',
'keyfile': '/path/to/certificate/abc.key', 'cafile':
'/path/to/certificate/ca.crt'}, 'replicas': 1, 'serviceaccount': 'router',
'namespace': 'default', 'stats_port': 1936, 'edits': [], 'images':
'openshift3/ose-${component}:${version}', 'selector': 'type=router1', 'ports':
['80:80', '443:443']},

{'name': 'router2', 'certificate': {'certfile': '/path/to/certificate/xyz.crt',
'keyfile': '/path/to/certificate/xyz.key', 'cafile':
'/path/to/certificate/ca.crt'}, 'replicas': 1, 'serviceaccount': 'router',
'namespace': 'default', 'stats_port': 1936, 'edits': [{'action': 'append',
'key': 'spec.template.spec.containers[0].env', 'value': {'name': 'ROUTE_LABELS',
'value': 'route=external'}}], 'images':
'openshift3/ose-${component}:${version}', 'selector': 'type=router2', 'ports':
['80:80', '443:443']}]
```

4.12. 配置 RED HAT GLUSTER STORAGE PERSISTENT STORAGE

Red Hat Gluster Storage 可以配置为为 OpenShift Container Platform 提供 持久性存储和动态置备。它可用于 OpenShift Container Platform 中容器化（聚合模式）和在其自身节点上的非容器化（独立模式）。

您可以使用变量配置 Red Hat Gluster Storage 集群，这些变量与 OpenShift Container Platform 集群交互。您在 **[OSEv3:vars]** 组中定义的变量包括主机变量、角色变量以及镜像名称和版本标签变量。

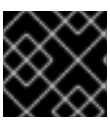
您可以使用 **glusterfs_devices** 主机变量定义块设备列表来管理 Red Hat Gluster Storage 集群。您配置中的每个主机必须至少有一个 **glusterfs_devices** 变量，对于每次配置来说，必须至少有一个裸机设备没有分区或 LVM PV。

角色变量控制将 Red Hat Gluster Storage 集群整合到新的或现有的 OpenShift Container Platform 集群。您可以定义多个角色变量，其中每个变量都有一个对应变量，可选择性配置一个独立的 Red Hat Gluster Storage 集群，用作集成 Docker registry 的存储。

您可以定义镜像名称和版本标签变量，以防止 OpenShift Container Platform pod 在中断后进行升级，这可能导致使用不同 OpenShift Container Platform 版本的集群。您还可以定义这些变量，以指定所有容器化组件的镜像名称和版本标签。

额外的信息和示例，包括以下信息及示例，可在 使用 Red Hat Gluster Storage 的持久性存储中找到。

4.12.1. 配置聚合模式



重要

如需具体主机的准备和先决条件，请参阅 聚合模式考虑。

1. 在清单文件的 [OSEv3:vars] 部分中包含以下变量，并根据您的配置需要调整它们：

```
[OSEv3:vars]
...
openshift_storage_glusterfs_namespace=app-storage
openshift_storage_glusterfs_storageclass=true
openshift_storage_glusterfs_storageclass_default=false
openshift_storage_glusterfs_block_deploy=true
openshift_storage_glusterfs_block_host_vol_size=100
openshift_storage_glusterfs_block_storageclass=true
openshift_storage_glusterfs_block_storageclass_default=false
```

2. 在 [OSEv3: Child] 部分添加 glusterfs 来启用 [glusterfs] 组：

```
[OSEv3:children]
masters
nodes
glusterfs
```

3. 添加 [glusterfs] 部分，其中包含托管 GlusterFS 存储的每个存储节点的条目。对于每个节点，将 glusterfs_devices 设置为作为 GlusterFS 集群一部分完全管理的原始块设备列表。必须至少列出一个设备。每个设备都必须是空的，没有分区或 LVM PV。以以下形式指定变量：

```
<hostname_or_ip> glusterfs_devices='[ "</path/to/device1/>", "</path/to/device2>", ... ]'
```

例如：

```
[glusterfs]
node11.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
node12.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
node13.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
```

4. 将 [glusterfs] 下列出的主机添加到 [nodes] 组中：

```
[nodes]
...
node11.example.com openshift_node_group_name="node-config-compute"
node12.example.com openshift_node_group_name="node-config-compute"
node13.example.com openshift_node_group_name="node-config-compute"
```

为了使部署成功，需要有效的镜像标签。将 <tag> 替换为与 OpenShift Container Platform 3.11 兼容的 Red Hat Gluster Storage 版本，如清单文件中以下变量互操作性表格 中所述：

- openshift_storage_glusterfs_image=registry.redhat.io/rhgs3/rhgs-server-rhel7:<tag>
- openshift_storage_glusterfs_block_image=registry.redhat.io/rhgs3/rhgs-gluster-block-prov-rhel7:<tag>
- openshift_storage_glusterfs_s3_image=registry.redhat.io/rhgs3/rhgs-s3-server-rhel7:<tag>
- openshift_storage_glusterfs_heketi_image=registry.redhat.io/rhgs3/rhgs-volmanager-rhel7:<tag>

- **openshift_storage_glusterfs_registry_image=registry.redhat.io/rhgs3/rhgs-server-rhel7:<tag>**
- **openshift_storage_glusterfs_block_registry_registry.redhat.io/rhgs3/rhgs-gluster-block-prov-rhel7:<tag>**
- **openshift_storage_glusterfs_s3_registry_image=registry.redhat.io/rhgs3/rhgs-s3-server-rhel7:<tag>**
- **openshift_storage_glusterfs_heketi_registry_image=registry.redhat.io/rhgs3/rhgs-volmanager-rhel7:<tag>**

4.12.2. 配置独立模式

1. **在清单文件的 [OSEv3:vars] 部分中包含以下变量，并根据您的配置需要调整它们：**

```
[OSEv3:vars]
...
openshift_storage_glusterfs_namespace=app-storage
openshift_storage_glusterfs_storageclass=true
openshift_storage_glusterfs_storageclass_default=false
openshift_storage_glusterfs_block_deploy=true
openshift_storage_glusterfs_block_host_vol_size=100
openshift_storage_glusterfs_block_storageclass=true
openshift_storage_glusterfs_block_storageclass_default=false
openshift_storage_glusterfs_is_native=false
openshift_storage_glusterfs_heketi_is_native=true
openshift_storage_glusterfs_heketi_executor=ssh
openshift_storage_glusterfs_heketi_ssh_port=22
openshift_storage_glusterfs_heketi_ssh_user=root
openshift_storage_glusterfs_heketi_ssh_sudo=false
openshift_storage_glusterfs_heketi_ssh_keyfile="/root/.ssh/id_rsa"
```

2. **在 [OSEv3: Child] 部分添加 glusterfs 来启用 [glusterfs] 组：**

```
[OSEv3:children]
masters
nodes
glusterfs
```

3. **添加 [glusterfs] 部分，其中包含托管 GlusterFS 存储的每个存储节点的条目。对于每个节点，将 glusterfs_devices 设置为作为 GlusterFS 集群一部分完全管理的原始块设备列表。必须至少列出一个设备。每个设备都必须是空的，没有分区或 LVM PV。另外，将 glusterfs_ip 设置为节点的 IP 地址。以以下形式指定变量：**

```
<hostname_or_ip> glusterfs_ip=<ip_address> glusterfs_devices='[ "</path/to/device1/>", "
</path/to/device2>", ... ]'
```

例如：

```
[glusterfs]
gluster1.example.com glusterfs_ip=192.168.10.11 glusterfs_devices='[ "/dev/xvdc",
"/dev/xvdd" ]'
gluster2.example.com glusterfs_ip=192.168.10.12 glusterfs_devices='[ "/dev/xvdc",
```

```
"/dev/xvdd" ]'
gluster3.example.com glusterfs_ip=192.168.10.13 glusterfs_devices='[ "/dev/xvdc",
"/dev/xvdd" ]'
```

4.13. 配置 OPENSHIFT CONTAINER REGISTRY

可使用安装程序部署集成的 OpenShift Container Registry。

4.13.1. 配置 registry 存储

如果没有使用 registry 存储选项，默认的 OpenShift Container Registry 为临时存储。当 pod 不再存在时会丢失所有数据。

重要

测试显示，使用 RHEL NFS 服务器作为容器镜像 registry 的存储后端会出现问题。这包括 OpenShift Container Registry 和 Quay。因此，不建议使用 RHEL NFS 服务器来备份核心服务使用的 PV。

市场上的其他 NFS 实现可能没有这些问题。如需了解更多与此问题相关的信息，请联络相关的 NFS 厂商。

在使用高级安装程序时启用 registry 存储有几个选项：

选项 A：**NFS 主机组**
当设置以下变量时，会在集群安装过程中在 [nfs]主机组中的主机上创建一个 NFS 卷（路径为 *<nfs_directory>/<volume_name>*）例如，使用这些选项的卷路径是 */exports/registry*:

```
[OSEv3:vars]

# nfs_directory must conform to DNS-1123 subdomain must consist of lower case
# alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character

openshift_hosted_registry_storage_kind=nfs
openshift_hosted_registry_storage_access_modes=['ReadWriteMany']
openshift_hosted_registry_storage_nfs_directory=/exports
openshift_hosted_registry_storage_nfs_options='*(rw,root_squash)'
openshift_hosted_registry_storage_volume_name=registry
openshift_hosted_registry_storage_volume_size=10Gi
```

选项 B：**外部 NFS 主机**
要使用外部 NFS 卷，必须在存储主机上有 *<nfs_directory>/<volume_name>* 路径。使用以下选项的远程卷路径为 *nfs.example.com:/exports/registry*。

```
[OSEv3:vars]

# nfs_directory must conform to DNS-1123 subdomain must consist of lower case
# alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character

openshift_hosted_registry_storage_kind=nfs
openshift_hosted_registry_storage_access_modes=['ReadWriteMany']
openshift_hosted_registry_storage_host=nfs.example.com
```

```
openshift_hosted_registry_storage_nfs_directory=/exports
openshift_hosted_registry_storage_volume_name=registry
openshift_hosted_registry_storage_volume_size=10Gi
```

升级或安装带有 NFS 的 OpenShift Container Platform
选项 C：OpenStack Platform
OpenStack 存储配置必须已经存在。

```
[OSEv3:vars]

openshift_hosted_registry_storage_kind=openstack
openshift_hosted_registry_storage_access_modes=['ReadWriteOnce']
openshift_hosted_registry_storage_openstack_filesystem=ext4
openshift_hosted_registry_storage_openstack_volumeID=3a650b4f-c8c5-4e0a-8ca5-eaee11f16c57
openshift_hosted_registry_storage_volume_size=10Gi
```

选项 D:：WS 或者其它 S3 存储解决方案
简单的存储解决方案（S3）存储桶必须已经存在。

```
[OSEv3:vars]

#openshift_hosted_registry_storage_kind=object
#openshift_hosted_registry_storage_provider=s3
#openshift_hosted_registry_storage_s3_accesskey=access_key_id
#openshift_hosted_registry_storage_s3_secretkey=secret_access_key
#openshift_hosted_registry_storage_s3_bucket=bucket_name
#openshift_hosted_registry_storage_s3_region=bucket_region
#openshift_hosted_registry_storage_s3_chunksize=26214400
#openshift_hosted_registry_storage_s3_rootdirectory=/registry
#openshift_hosted_registry_pullthrough=true
#openshift_hosted_registry_acceptschema2=true
#openshift_hosted_registry_enforcequota=true
```

如果您使用不同的 S3 服务，如 Minio 或 ExoScale，还要添加 region 端点参数：

```
openshift_hosted_registry_storage_s3_regionendpoint=https://myendpoint.example.com/
```

选项 E：聚合模式
与 配置聚合模式 类似，可将 Red Hat Gluster Storage 配置为在初始安装集群期间为 OpenShift Container Registry 提供存储，以便为 registry 提供冗余可靠的存储。

重要

如需具体主机的准备和先决条件，请参阅 聚合模式考虑。

1. 在清单文件的 [OSEv3:vars] 部分中设置以下变量，并根据您的配置需要调整它们：

```
[OSEv3:vars]
...
openshift_hosted_registry_storage_kind=glusterfs (1)
openshift_hosted_registry_storage_volume_size=5Gi
openshift_hosted_registry_selector='node-role.kubernetes.io/infra=true'
```

1 建议在基础架构节点上运行集成的 OpenShift Container Registry。基础架构节点是专用于运行管理员部署的应用程序的节点,用于为 OpenShift Container Platform 集群提供服务。

2. 在 **[OSEv3: Child]** 部分添加 **glusterfs_registry** 来启用 **[glusterfs_registry]** 组：

```
[OSEv3:children]
masters
nodes
glusterfs_registry
```

3. 添加 **[glusterfs_registry]** 部分，其中包含托管 GlusterFS 存储的每个存储节点的条目。对于每个节点，将 **glusterfs_devices** 设置为作为 GlusterFS 集群一部分完全管理的原始块设备列表。必须至少列出一个设备。每个设备都必须是空的，没有分区或 LVM PV。以以下形式指定变量：

```
<hostname_or_ip> glusterfs_devices='[ "</path/to/device1/>", "</path/to/device2>", ... ]'
```

例如：

```
[glusterfs_registry]
node11.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
node12.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
node13.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
```

4. 将 **[glusterfs_registry]** 下列出的主机添加到 **[nodes]** 组中：

```
[nodes]
...
node11.example.com openshift_node_group_name="node-config-infra"
node12.example.com openshift_node_group_name="node-config-infra"
node13.example.com openshift_node_group_name="node-config-infra"
```

选项 F：Google Compute Engine（GCE）上的 Google Cloud Storage（GCS）存储桶
GCS 存储桶必须已经存在。

```
[OSEv3:vars]

openshift_hosted_registry_storage_provider=gcs
openshift_hosted_registry_storage_gcs_bucket=bucket01
openshift_hosted_registry_storage_gcs_keyfile=test.key
openshift_hosted_registry_storage_gcs_rootdirectory=/registry
```

选项 G：使用 vSphere Cloud Provider（VCP）的 vSphere 卷
vSphere Cloud Provider 必须通过 OpenShift Container Platform 节点访问的数据存储进行配置。

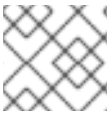
在为 registry 使用 vSphere 卷时，您必须将存储访问模式设置为 **ReadWriteOnce**，并将副本数设置为 **1**:

```
[OSEv3:vars]

openshift_hosted_registry_storage_kind=vsphere
openshift_hosted_registry_storage_access_modes=['ReadWriteOnce']
openshift_hosted_registry_storage_annotations=['volume.beta.kubernetes.io/storage-provisioner:
kubernetes.io/vsphere-volume']
openshift_hosted_registry_replicas=1
```

4.14. 配置全局代理选项

如果您的主机需要使用 HTTP 或 HTTPS 代理来连接到外部主机，则必须配置多个组件来使用代理，包括 master、Docker 和构建。只连接 master API 的节点服务不需要外部访问，因此不需要配置为使用代理。

为简化此配置，可在集群或主机级别指定以下 Ansible 变量，以便在您的环境中统一应用这些设置。



注意

如需了解有关如何为构建定义代理环境的更多信息，请参阅 配置全局构建默认值和覆盖。

表 4.9. Cluster Proxy 变量

变量	用途
openshift_http_proxy	此变量指定 master 和 Docker 守护进程的 **HTTP_PROXY** 环境变量。
openshift_https_proxy	此变量指定 master 和 Docker 守护进程的 **HTTPS_PROXY** 环境变量。
openshift_no_proxy	此变量用于为 master 和 Docker 守护进程设置 **NO_PROXY** 环境变量。提供不使用定义的代理的主机名、域名或通配符主机名的以逗号分隔的列表。默认情况下，此列表包含所有定义的 OpenShift Container Platform 主机名列表。 不使用定义的代理的主机名包括： • Master 和节点主机名。您必须包含域后缀。 • 其他内部主机名。您必须包含域后缀。 • etcd IP 地址。您必须提供 IP 地址，因为 etcd 访问是由 IP 地址管理的。 • 容器镜像 registry IP 地址。 • Kubernetes IP 地址。在默认情况下，这个值为 **172.30.0.1**。如果提供，则为 **openshift_portal_net** 参数的值。 • **cluster.local** Kubernetes 内部域后缀。 • **svc** Kubernetes 内部域后缀。
openshift_generate_no_proxy_hosts	此布尔变量指定是否自动附加所有定义的 OpenShift 主机的名称和 ***.cluster.local** 附加到**NO_PROXY** 列表。默认值为 **true**; 设置为 **false** 以覆盖这个选项。

变量	用途
openshift_builddefaults_http_proxy	此变量使用 **BuildDefaults** 准入控制器定义插入到构建中的 **HTTP_PROXY** 环境变量。如果您没有定义此参数，但定义 **openshift_http_proxy** 参数，则使用 **openshift_http_proxy** 值。将 **openshift_builddefaults_http_proxy** 值设置为 **False** 以禁用构建的默认 http 代理服务器，而不考虑 **openshift_http_proxy** 值。
openshift_builddefaults_https_proxy	此变量使用 **BuildDefaults** 准入控制器定义插入到构建中的 **HTTPS_PROXY** 环境变量。如果您没有定义此参数，但定义了 **openshift_http_proxy** 参数，则使用 **openshift_https_proxy** 值。将 **openshift_builddefaults_https_proxy** 值设置为 **False**，以禁用构建的默认 https 代理服务器，而不考虑 **openshift_https_proxy** 值。
openshift_builddefaults_no_proxy	此变量使用 **BuildDefaults** 准入控制器定义插入到构建中的 **NO_PROXY** 环境变量。将 **openshift_builddefaults_no_proxy** 值设置为 **False**，以禁用构建的默认 no proxy 设置，而不考虑 **openshift_no_proxy** 值。
openshift_builddefaults_git_http_proxy	此变量定义了 **git clone** 操作在构建期间使用的 HTTP 代理，该代理使用 **BuildDefaults** 准入控制器定义。将 **openshift_builddefaults_git_http_proxy** 值设置为 **False**，在构建期间为 **git clone** 操作禁用默认 http 代理，而不考虑 **openshift_http_proxy** 值。
openshift_builddefaults_git_https_proxy	此变量定义了 **git clone** 操作在构建期间使用的 HTTPS 代理，该代理使用 **BuildDefaults** 准入控制器定义。将 **openshift_builddefaults_git_https_proxy** 值设置为 **False**，在构建期间为 **git clone** 操作禁用默认 https 代理，而不考虑 **openshift_https_proxy** 值。

4.15. 配置防火墙

重要

- **如果要更改默认防火墙，请确保集群中的每个主机都使用相同的防火墙类型以防止不一致。**
- **不要在 Atomic Host 上安装的 OpenShift Container Platform 中使用 firewalld。Atomic 主机上不支持 firewalld。**

注意

虽然 iptables 是默认的防火墙，但推荐在新的安装中使用 firewalld。

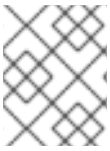
OpenShift Container Platform 使用 iptables 作为默认防火墙，但您可以将集群配置为在安装过程中使用 firewalld。

因为 iptables 是默认防火墙，所以 OpenShift Container Platform 旨在自动进行配置。但是，如果未正确配置，iptables 规则可能会破坏 OpenShift Container Platform。firewalld 的优点包括允许多个对象安全地共享防火墙规则。

要将 firewalld 用作 OpenShift Container Platform 安装的防火墙，请在安装时将 **os_firewall_use_firewalld** 变量添加到 Ansible 主机文件的配置变量列表中：

```
[OSEv3:vars]
os_firewall_use_firewalld=True (1)
```

1. 将这个变量设置为 **true** 可打开所需的端口并在默认区中添加规则，确保正确配置了 firewalld。



注意

使用 firewalld 的默认配置会带有有限的配置选项，它们不能被覆盖。例如，当您可以在多个区中使用接口设置存储网络时，节点通信的接口必须位于默认区。

4.16. 配置会话选项

OAuth 配置中的会话选项可在清单文件中配置。默认情况下，Ansible 使用生成的身份验证和加密 secret 填充 **sessionSecretsFile**，以便由一个 master 生成的会话可以被其他解码。默认位置为 */etc/origin/master/session-secrets.yaml*，只有在所有 master 上都被删除时才会重新创建该文件。

您可以使用 **openshift_master_session_name** 和 **openshift_master_session_max_seconds** 设置会话名称和最大秒数：

```
openshift_master_session_name=ssn
openshift_master_session_max_seconds=3600
```

如果提供，**openshift_master_session_auth_secrets** 和 **openshift_master_encryption_secrets** 的长度必须相等。

对于 **openshift_master_session_auth_secrets**，用于使用 HMAC 验证会话，建议使用 32 或 64 字节的 secret:

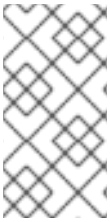
```
openshift_master_session_auth_secrets=['DONT+USE+THIS+SECRET+b4NV+pmZNSO']
```

对于 **openshift_master_encryption_secrets**，用于加密会话，secret 必须为 16、24 或 32 个字符长，才能选择 AES-128、AES-192 或 AES-256:

```
openshift_master_session_encryption_secrets=['DONT+USE+THIS+SECRET+b4NV+pmZNSO']
```

4.17. 配置自定义证书

OpenShift Container Platform API 和 Web 控制台的公共主机名的自定义服务证书 可以在集群安装过程中部署，并可在清单文件中配置。



注意

为与 **publicMasterURL** 关联的主机名配置自定义证书，该证书设置为 **openshift_master_cluster_public_hostname** 参数值。为与 **masterURL**（**openshift_master_cluster_hostname**）关联的主机名使用自定义服务证书会导致 TLS 错误，因为基础架构组件尝试使用内部 **masterURL** 主机联系主 API。

可使用 **openshift_master_named_certificates** 集群变量配置证书和密钥文件路径：

```
openshift_master_named_certificates=[{"certfile": "/path/to/custom1.crt", "keyfile":
"/path/to/custom1.key", "cafile": "/path/to/custom-ca1.crt"}]
```

文件路径对于运行 Ansible 的系统来说必须是本地的。证书会复制到 master 主机，并部署到 */etc/origin/master/named_certificates/* 目录中。

Ansible 检测到证书的 **Common Name** 和 **Subject Alternative Names**。在设置 **openshift_master_named_certificates** 时，可以通过提供 **"names"** 键来覆盖检测到的名称：

```
openshift_master_named_certificates=[{"certfile": "/path/to/custom1.crt", "keyfile":
"/path/to/custom1.key", "names": ["public-master-host.com"], "cafile": "/path/to/custom-ca1.crt"}]
```

使用 **openshift_master_named_certificates** 配置的证书会在 master 上缓存。这意味着每个额外的 Ansible 运行使用不同的证书集合，会导致以前部署的所有证书都保留在 master 主机和 master 配置文件中。

如果要使用提供的值（或无值）覆盖 **openshift_master_named_certificates**，请指定 **openshift_master_overwrite_named_certificates** 集群变量：

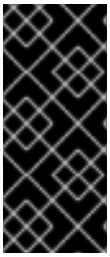
```
openshift_master_overwrite_named_certificates=true
```

如需更完整的示例，请考虑清单文件中的以下集群变量：

```
openshift_master_cluster_method=native
openshift_master_cluster_hostname=lb-internal.openshift.com
openshift_master_cluster_public_hostname=custom.openshift.com
```

要覆盖后续 Ansible 运行上的证书，请设置以下参数值：

```
openshift_master_named_certificates=[{"certfile": "/root/STAR.openshift.com.crt", "keyfile":
"/root/STAR.openshift.com.key", "names": ["custom.openshift.com"], "cafile": "/root/ca-file.crt"}]
openshift_master_overwrite_named_certificates=true
```



重要

cafile 证书在安装过程中或重新部署证书期间导入到 master 上的 **ca-bundle.crt** 文件中。**ca-bundle.crt** 文件挂载到在 OpenShift Container Platform 中运行的每个 pod 中。几个 OpenShift Container Platform 组件会在访问 **masterPublicURL** 端点时自动信任命名的证书。如果在证书参数中省略 **cafile** 选项，Web 控制台的功能和几个其他组件会减少。

4.18. 配置证书验证

默认情况下，用于管理 etcd、master 和 kubelet 的证书会在 2 到 5 年后过期。自动生成的 registry、CA、节点和 master 证书的有效性（以天为单位直到其过期）在安装过程中可使用以下变量配置：

```
[OSEv3:vars]

openshift_hosted_registry_cert_expire_days=730
openshift_ca_cert_expire_days=1825
openshift_master_cert_expire_days=730
etcd_ca_default_days=1825
```

这些值也用于通过 Ansible 安装后 重新部署证书。

4.19. 配置集群监控

Prometheus Cluster Monitoring 被设置为自动部署。要防止自动部署，请设置以下内容：

```
[OSEv3:vars]

openshift_cluster_monitoring_operator_install=false
```

如需有关 Prometheus Cluster Monitoring 及其配置的更多信息，请参阅 Prometheus Cluster Monitoring 文档。

4.20. 配置集群指标

集群指标（metrics）不会被设置为自动部署。将以下内容设置为在集群安装过程中启用集群指标：

```
[OSEv3:vars]

openshift_metrics_install_metrics=true
```

指标公共 URL 可以在集群安装过程中使用 openshift_metrics_hawkular_hostname Ansible 变量设置，默认值为：

https://hawkular-metrics.{{openshift_master_default_subdomain}}/hawkular/metrics

如果更改了这个变量，请确保主机名可以通过路由器访问。

```
openshift_metrics_hawkular_hostname=hawkular-metrics.
{{openshift_master_default_subdomain}}
```

重要

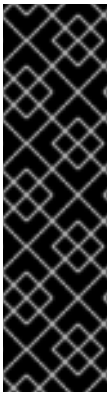
根据上游 Kubernetes 规则，指标只能从 eth0 的默认接口收集。

注意

您必须设置 openshift_master_default_subdomain 值才能部署指标。

4.20.1. 配置 Metrics 存储

必须设置 openshift_metrics_cassandra_storage_type 变量才能将持久性存储用于指标数据。如果没有设置 openshift_metrics_cassandra_storage_type，则集群指标数据将存储在 emptyDir 卷中，该卷将在 Cassandra pod 终止时被删除。



重要

测试显示，使用 RHEL NFS 服务器作为容器镜像 registry 的存储后端会出现问题。这包括用于 metrics 存储的 Cassandra。因此，不建议使用 RHEL NFS 服务器来备份核心服务使用的 PV。

Cassandra 旨在通过多个独立实例提供冗余性。因此，在数据目录中使用 NFS 或 SAN 是一个反模式，我们不推荐这样做。

然而，市场中的其他 NFS/SAN 实现可能没有这个问题。如需了解更多与此问题相关的信息，请联络相关的 NFS/SAN 厂商。

在集群安装过程中启用集群指标存储有三个选项：

选项 A：动态
如果您的 OpenShift Container Platform 环境支持为云供应商进行 动态卷置备，请使用以下变量：

```
[OSEv3:vars]

openshift_metrics_cassandra_storage_type=dynamic
```

如果有多个默认动态置备的卷类型，如 gluster-storage 和 glusterfs-storage-block，您可以根据变量指定置备的卷类型。使用以下变量：

```
[OSEv3:vars]

openshift_metrics_cassandra_storage_type=pv
openshift_metrics_cassandra_pvc_storage_class_name=glusterfs-storage-block
```

有关使用 DynamicProvisioningEnabled 来启用或禁用动态置备的详情，请查看 卷配置。

选项 B：NFS 主机组
当设置以下变量时，会在集群安装过程中在 [nfs]主机组中的主机上创建一个 NFS 卷（路径为 *<nfs_directory>/<volume_name>*）例如，使用这些选项的卷路径是 */exports/metrics*:

```
[OSEv3:vars]

# nfs_directory must conform to DNS-1123 subdomain must consist of lower case
# alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character

openshift_metrics_storage_kind=nfs
openshift_metrics_storage_access_modes=['ReadWriteOnce']
openshift_metrics_storage_nfs_directory=/exports
openshift_metrics_storage_nfs_options='*(rw,root_squash)'
openshift_metrics_storage_volume_name=metrics
openshift_metrics_storage_volume_size=10Gi
```

选项 C：外部 NFS 主机
要使用外部 NFS 卷，必须在存储主机上有 *<nfs_directory>/<volume_name>* 路径。

```
[OSEv3:vars]

# nfs_directory must conform to DNS-1123 subdomain must consist of lower case
# alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character

openshift_metrics_storage_kind=nfs
openshift_metrics_storage_access_modes=['ReadWriteOnce']
openshift_metrics_storage_host=nfs.example.com
openshift_metrics_storage_nfs_directory=/exports
openshift_metrics_storage_volume_name=metrics
openshift_metrics_storage_volume_size=10Gi
```

使用以下选项的远程卷路径是 *nfs.example.com:/exports/metrics*。

升级或安装带有 NFS 的 OpenShift Container Platform
不建议将 NFS 用于 OpenShift Container Platform 核心组件，因为 NFS（和 NFS 协议）没有提供组成 OpenShift Container Platform 基础架构的应用程序所需的一致性。

因此，安装程序和更新 playbook 需要一个选项来启用使用带核心基础架构组件的 NFS。

```
# Enable unsupported configurations, things that will yield a partially
# functioning cluster but would not be supported for production use
#openshift_enable_unsupported_configurations=false
```

如果您在升级或安装集群时看到以下信息，则需要额外的步骤。

```
TASK [Run variable sanity checks] ***********************************************
fatal: [host.example.com]: FAILED! => {"failed": true, "msg": "last_checked_host: host.example.com,
last_checked_var: openshift_hosted_registry_storage_kind;nfs is an unsupported type for
openshift_hosted_registry_storage_kind. openshift_enable_unsupported_configurations=True
mustbe specified to continue with this configuration."}
```

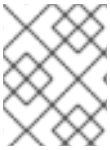
在 Ansible 清单文件中指定以下参数：

```
[OSEv3:vars]
openshift_enable_unsupported_configurations=True
```

4.21. 配置集群日志记录

默认情况下，集群日志被设置为不进行自动部署。设置以下内容以在集群安装过程中启用集群日志记录：

```
[OSEv3:vars]

openshift_logging_install_logging=true
```



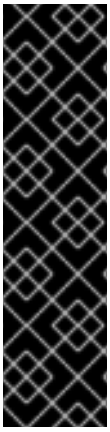
注意

安装集群日志记录时，还必须在 Ansible 清单文件中指定一个节点选择器，如 **openshift_logging_es_nodeselector={"node-role.kubernetes.io/infra": "true"}**。

如需有关可用集群日志记录变量的更多信息，请参阅 指定日志记录 Ansible 变量。

4.21.1. 配置日志存储

必须设置 openshift_logging_es_pvc_dynamic 变量，以便使用持久性存储进行日志记录。如果没有设置 openshift_logging_es_pvc_dynamic，则集群日志记录数据存储在 emptyDir 卷中，当 Elasticsearch Pod 终止时会删除它。



重要

测试显示，使用 RHEL NFS 服务器作为容器镜像 registry 的存储后端会出现问题。这包括用于日志存储的 ElasticSearch。因此，不建议使用 RHEL NFS 服务器来备份核心服务使用的 PV。

由于 ElasticSearch 没有实现自定义 deletionPolicy，所以 Elasticsearch 存储不支持将 NFS 存储用作卷或持久性卷，因为 Lucene 和默认的 deletePolicy 依赖于 NFS 不提供的文件系统行为。数据崩溃和其他问题可能会发生。

市场上的其他 NFS 实现可能没有这些问题。如需了解更多与此问题相关的信息，请联络相关的 NFS 厂商。

在集群安装过程中启用集群日志记录存储有三个选项：

选项 A：动态
如果您的 OpenShift Container Platform 环境有动态卷置备，它可以通过云供应商或一个独立的存储供应商进行配置。例如，云供应商可以在 GCE 上带有 provisioner kubernetes.io/gce-pd 的 StorageClass，而一个独立的存储供应商（如 GlusterFS）则可能具有带有 provisioner kubernetes.io/glusterfs 的 StorageClass。在这两种情况下，请使用以下变量：

```
[OSEv3:vars]

openshift_logging_es_pvc_dynamic=true
```

有关动态置备的额外信息，请参阅 动态置备和创建存储类。

如果有多个默认动态置备的卷类型，如 gluster-storage 和 glusterfs-storage-block，您可以根据变量指定置备的卷类型。使用以下变量：

```
[OSEv3:vars]

openshift_logging_elasticsearch_storage_type=pvc
openshift_logging_es_pvc_storage_class_name=glusterfs-storage-block
```

有关使用 DynamicProvisioningEnabled 来启用或禁用动态置备的详情，请查看 卷配置。

选项 B：NFS 主机组
当设置以下变量时，会在集群安装过程中在 [nfs]主机组中的主机上创建一个 NFS 卷（路径为 *<nfs_directory>/<volume_name>*）例如，使用这些选项的卷路径是 */exports/logging*:

```
[OSEv3:vars]

# nfs_directory must conform to DNS-1123 subdomain must consist of lower case
# alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character

openshift_logging_storage_kind=nfs
openshift_logging_storage_access_modes=['ReadWriteOnce']
openshift_logging_storage_nfs_directory=/exports 1
```

```
openshift_logging_storage_nfs_options='*(rw,root_squash)' 2
openshift_logging_storage_volume_name=logging 3
openshift_logging_storage_volume_size=10Gi
openshift_enable_unsupported_configurations=true
openshift_logging_elasticsearch_storage_type=pvc
openshift_logging_es_pvc_size=10Gi
openshift_logging_es_pvc_storage_class_name=''
openshift_logging_es_pvc_dynamic=true
openshift_logging_es_pvc_prefix=logging
```

1 2 这些参数只适用于 **/usr/share/ansible/openshift-ansible/playbooks/deploy_cluster.yml** 安装 playbook。这些参数无法用于 **/usr/share/ansible/openshift-ansible/playbooks/openshift-logging/config.yml** playbook。

3 NFS 卷名称必须是 日志记录。

选项 C：外部 NFS 主机
要使用外部 NFS 卷，必须在存储主机上有 *<nfs_directory>/<volume_name>* 路径。

```
[OSEv3:vars]

# nfs_directory must conform to DNS-1123 subdomain must consist of lower case
# alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character

openshift_logging_storage_kind=nfs
openshift_logging_storage_access_modes=['ReadWriteOnce']
openshift_logging_storage_host=nfs.example.com 1
openshift_logging_storage_nfs_directory=/exports 2
openshift_logging_storage_volume_name=logging 3
openshift_logging_storage_volume_size=10Gi
openshift_enable_unsupported_configurations=true
openshift_logging_elasticsearch_storage_type=pvc
openshift_logging_es_pvc_size=10Gi
openshift_logging_es_pvc_storage_class_name=''
openshift_logging_es_pvc_dynamic=true
openshift_logging_es_pvc_prefix=logging
```

1 2 这些参数只适用于 **/usr/share/ansible/openshift-ansible/playbooks/deploy_cluster.yml** 安装 playbook。这些参数无法用于 **/usr/share/ansible/openshift-ansible/playbooks/openshift-logging/config.yml** playbook。

3 NFS 卷名称必须是 日志记录。

使用以下选项的远程卷路径为 *nfs.example.com:/exports/logging*。

升级或安装带有 NFS 的 OpenShift Container Platform
不建议将 NFS 用于 OpenShift Container Platform 核心组件，因为 NFS（和 NFS 协议）没有提供组成 OpenShift Container Platform 基础架构的应用程序所需的一致性。

因此，安装程序和更新 playbook 需要一个选项来启用使用带核心基础架构组件的 NFS。

```
# Enable unsupported configurations, things that will yield a partially
# functioning cluster but would not be supported for production use
#openshift_enable_unsupported_configurations=false
```

如果您在升级或安装集群时看到以下信息，则需要额外的步骤。

```
TASK [Run variable sanity checks] ***********************************************
fatal: [host.example.com]: FAILED! => {"failed": true, "msg": "last_checked_host: host.example.com,
last_checked_var: openshift_hosted_registry_storage_kind;nfs is an unsupported type for
openshift_hosted_registry_storage_kind. openshift_enable_unsupported_configurations=True
mustbe specified to continue with this configuration."}
```

在 Ansible 清单文件中指定以下参数：

```
[OSEv3:vars]
openshift_enable_unsupported_configurations=True
```

4.22. 自定义服务目录选项

服务目录**在安装过程中默认启用。启用服务代理允许您在目录中注册服务代理。启用服务目录后，OpenShift Ansible 代理和模板服务代理也会被安装。如需更多信息，请参阅**配置 OpenShift Ansible Broker **和**配置 Template Service Broker**。如果您禁用服务目录，则不会安装 OpenShift Ansible 代理和模板服务代理。**

要禁用服务目录的自动部署，请在清单文件中设置以下集群变量：

```
openshift_enable_service_catalog=false
```

如果使用自己的 registry，您必须添加：

- **openshift_service_catalog_image_prefix：拉取服务目录镜像时，强制使用特定的前缀（如 registry）。您必须提供镜像名称前的完整 registry 名称。**
- **openshift_service_catalog_image_version：拉取服务目录镜像时，强制使用特定镜像版本。**

例如：

```
openshift_service_catalog_image="docker-registry.default.example.com/openshift/ose-service-
catalog:${version}"
openshift_service_catalog_image_prefix="docker-registry-default.example.com/openshift/ose-"
openshift_service_catalog_image_version="v3.9.30"
```

4.22.1. 配置 OpenShift Ansible Broker

OpenShift Ansible 代理**（OAB）在安装过程中默认启用。**

如果您不想安装 OAB，请在清单文件中将 ansible_service_broker_install 参数值设置为 false:

```
ansible_service_broker_install=false
```

表 4.10. 服务代理自定义变量

变量	用途
openshift_service_catalog_image_prefix	指定服务目录组件镜像的前缀。

4.22.1.1. 为 OpenShift Ansible Broker 配置持久性存储

OAB 部署其自身的 etcd 实例与 OpenShift Container Platform 集群的其他集群使用的 etcd 实例不同。OAB 的 etcd 实例需要使用持久性卷（PV）的独立存储才能正常工作。如果没有 PV 可用，etcd 将等待到 PV 满足为止。OAB 应用程序将进入 CrashLoop 状态，直到 etcd 实例可用。

为了进行部署，一些 Ansible playbook 捆绑包（APB）还需要一个 PV。例如，每个数据库 APB 都有两个计划：Development 计划使用临时存储且不需要 PV，而 Production 计划会使用持久性存储，且需要一个 PV。

APB	PV 需要吗？
postgresql-apb	需要，但只适用于 Production 计划
mysql-apb	需要，但只适用于 Production 计划
mariadb-apb	需要，但只适用于 Production 计划
mediawiki-apb	是

为 OAB 配置持久性存储：

注意

以下示例显示使用 NFS 主机来提供所需的 PV，但可使用其他持久性存储供应商。

1. **在清单文件中，在 [OSEv3: Child] 部分添加 nfs 来启用 [nfs] 组：**

```
[OSEv3:children]
masters
nodes
nfs
```

2. **添加 [nfs] 组部分并为系统添加将会是 NFS 主机的主机名：**

```
[nfs]
master1.example.com
```

3. **在 [OSEv3:vars] 部分添加以下内容：**

```
# nfs_directory must conform to DNS-1123 subdomain must consist of lower case
# alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character

openshift_hosted_etcd_storage_kind=nfs
openshift_hosted_etcd_storage_nfs_options="*(rw,root_squash,sync,no_wdelay)"
```

```
openshift_hosted_etcd_storage_nfs_directory=/opt/osev3-etcd (1)
openshift_hosted_etcd_storage_volume_name=etcd-vol2 (2)
openshift_hosted_etcd_storage_access_modes=["ReadWriteOnce"]
openshift_hosted_etcd_storage_volume_size=1G
openshift_hosted_etcd_storage_labels={'storage': 'etcd'}
```

(1) (2) **在 [nfs] 组中的主机上将使用路径 <nfs_directory>/<volume_name> 创建 NFS 卷。例如：使用这些选项的卷路径为 */opt/osev3-etcd/etcd-vol2*。**

这些设置会创建一个持久性卷，在集群安装过程中附加到 OAB 的 etcd 实例。

4.22.1.2. 为本地 APB 开发配置 OpenShift Ansible Broker

要将 APB 开发与 OpenShift Container Registry 和 OAB 结合，必须定义 OAB 可访问的镜像白名单。如果没有定义白名单，代理将忽略 APB，用户将不会看到可用的 APB。

默认情况下，白名单为空，用户在没有配置代理的情况下无法将 APB 镜像添加到代理中。将所有以 -apb 结尾的镜像列入白名单：

1. **在清单文件中，将以下内容添加到 [OSEv3:vars] 部分：**

```
ansible_service_broker_local_registry_whitelist=['.*-apb$']
```

4.22.2. 配置 Template Service Broker

模板服务代理（TSB）在安装过程中默认启用。

如果不想安装 TSB，将 template_service_broker_install 参数值设置为 false:

```
template_service_broker_install=false
```

要配置 TSB，则必须将一个或多个项目定义为代理的源命名空间，用于将模板和镜像流加载到服务目录中。通过修改清单文件中的 [OSEv3:vars] 部分中的以下内容来设置源项目：

```
openshift_template_service_broker_namespaces=['openshift','myproject']
```

表 4.11. 模板服务代理自定义变量

变量	用途
template_service_broker_prefix	指定模板 service broker 组件镜像的前缀。
ansible_service_broker_image_prefix	指定 ansible 服务代理组件镜像的前缀。

4.23. 配置 WEB 控制台自定义

以下 Ansible 变量设置了 master 配置选项以自定义 Web 控制台。如需这些自定义选项的详细信息，请参阅自定义 Web 控制台。

表 4.12. Web 控制台自定义变量

变量	用途
openshift_web_console_install	决定是否安装 Web 控制台。可以设置为 **true** 或 **false**。默认值为 **true**。
openshift_web_console_prefix	指定 Web 控制台镜像的前缀。
openshift_master_logout_url	在 web 控制台配置中设置 **clusterInfo.logoutPublicURL**。详情请参阅 更改 Logout URL。示例值：**https://example.com/logout**
openshift_web_console_extension_script_urls	在 web 控制台配置中设置 **extensions.scriptURL**。详情请查看 载入扩展脚本和 Stylesheets。示例值：**['https://example.com/scripts/menu-customization.js','https://example.com/scripts/nav-customization.js ']**
openshift_web_console_extension_stylesheet_urls	在 web 控制台配置中设置 **extensions.stylesheetURLs**。详情请查看 载入扩展脚本和 Stylesheets。示例值：**['https://example.com/styles/logo.css','https://example.com/styles/custom-styles.css ']**
openshift_master_oauth_template	在 master 配置中设置 OAuth 模板。详情请参阅 自定义登录页面。示例值：**['/path/to/login-template.html']**
openshift_master_metrics_public_url	在 master 配置中设置 **metricsPublicURL**。详情请参阅 设置指标公共 URL。示例值：**https://hawkular-metrics.example.com/hawkular/metrics**
openshift_master_logging_public_url	在 master 配置中设置 **loggingPublicURL**。详情请参阅 Kibana。示例值：**https://kibana.example.com**
openshift_web_console_inactivity_timeout_minutes	配置 web 控制台以在一段时间不活跃后自动注销。必须是一个大于或等于 5 的整数， 使用 0 禁用该功能。默认值为 0（禁用）。
openshift_web_console_cluster_resource_overrides_enabled	布尔值表示是否为集群配置使用过量分配（overcommit）。为 **true** 时，Web 控制台会在编辑资源限制时隐藏 CPU 请求、CPU 限制和内存请求的字段，因为您必须使用集群资源覆盖配置来设置这些值。
openshift_web_console_enable_context_selector	在 web 控制台和 admin 控制台 mastheads 中启用上下文选择器，以便在两个控制台间快速切换。当两个控制台都被安装时，默认为 **true**。

4.24. 配置集群控制台

集群控制台是 Web 控制台等额外 Web 界面，但着重执行 admin 任务。集群控制台与 web 控制台支持多个相同的常见 OpenShift Container Platform 资源，但它也允许您查看集群的指标数据，并管理集群范围的资源，如节点、持久性卷、集群角色和自定义资源定义。以下变量可用于自定义集群控制台。

表 4.13. 集群控制台自定义变量

变量	用途
openshift_console_install	决定是否安装集群控制台。可以设置为 **true** 或 **false**。默认值为 **true**。
openshift_console_hostname	设置集群控制台的主机名。默认为 **console.<openshift_master_default_subdomain>**。如果更改了这个变量，请确保主机名可以通过路由器访问。
openshift_console_cert	用于集群控制台路由的可选证书。这只在使用自定义主机名时才需要。
openshift_console_key	集群控制台路由使用的可选密钥。这只在使用自定义主机名时才需要。
openshift_console_ca	集群控制台路由使用的可选 CA。这只在使用自定义主机名时才需要。
openshift_base_path	集群控制台的可选基本路径。如果设置，它应该以斜杠开头和结束，类似 **/console/**。默认为 / （无基本路径）。
openshift_console_auth_ca_file	用于连接到 OAuth 服务器的可选 CA 文件。默认为 **/var/run/secrets/kubernetes.io/serviceaccount/ca.crt**。通常，这个值不需要更改。您必须创建一个 ConfigMap，其中包含 CA 文件，并将其挂载到 **openshift-console** 命名空间中的**控制台** Deployment 中，其位置与您指定的位置相同。

4.25. 配置 OPERATOR LIFECYCLE MANAGER

重要

Operator Framework 是一个技术预览功能。技术预览功能不包括在红帽生产服务级别协议（SLA）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需红帽技术预览功能支持范围的更多信息，请参阅 https://access.redhat.com/support/offerings/techpreview/。

技术预览 Operator Framework 包含 Operator Lifecycle Manager（OLM）。您可选择在集群安装过程中安装 OLM，方法是在清单文件中设置以下变量：

注意

另外，技术预览 Operator Framework 可在集群安装后再安装。如需了解单独的说明，请参阅 使用 Ansible 安装 Operator Lifecycle Manager。

1. 在 [OSEv3:vars] 部分添加 openshift_enable_olm 变量，将其设置为 true:

```
openshift_enable_olm=true
```

2. 在 [OSEv3:vars] 部分添加 openshift_additional_registry_credentials 变量，设置拉取 Operator 容器所需的凭证：

```
openshift_additional_registry_credentials=
[{'host':'registry.connect.redhat.com','user':'<your_user_name>','password':'<your_password>','t
est_image':'mongodb/enterprise-operator:0.3.2'}]
```

将 user 和 password 设置为您用来登录到红帽客户门户网站的凭证，地址为 https://access.redhat.com。

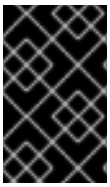
test_image 代表将用来测试您提供的凭证的镜像。

在集群安装成功后，请参阅 启动第一个 Operator 以了解在这个技术预览阶段将 OLM 作为集群管理员的进一步步骤。

第 5 章 清单文件示例

5.1. 概述

了解配置自己的清单文件的基本知识后，您可以查看以下示例清单，这些清单描述了各种环境提示，包括使用多个 master 进行高可用性。您可以选择一个符合您的要求的示例，对其进行修改以符合您自己的环境，并在运行安装时将其用作清单文件。

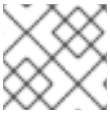


重要

以下示例清单在 [nodes] 组中的每个主机设置 openshift_node_group_name 时使用默认节点组集。要定义和使用您自己的自定义节点组定义，请在清单文件中设置 openshift_node_groups 变量。有关详细信息，请参阅定义节点组和主机映射。

5.2. 单 MASTER 示例

您可以使用单个 master 和多个节点配置环境，也可以配置单个或多个 etcd 主机。



注意

在安装后从单一 master 集群移动到多个 master 集群不被支持。

5.2.1. 单 Master、单一 etcd 和多个节点

下表描述了单个 master （单一 etcd 实例在同一主机上运行的静态 pod）、两个托管用户应用程序的节点以及两个具有 node-role.kubernetes.io/infra=true 标签的节点，用于托管专用基础架构的示例环境：

主机名	安装的组件/角色
master.example.com	Master、etcd 和 node
node1.example.com	Compute 节点
node2.example.com	
infra-node1.example.com	Infrastructure 节点
infra-node2.example.com	

您可以看到以下清单文件的 [masters]、[etcd] 和 [nodes] 部分中出现的这些示例主机：

单个 Master、单一 etcd 和多节点清单文件

```
# Create an OSEv3 group that contains the masters, nodes, and etcd groups
[OSEv3:children]
masters
nodes
etcd

# Set variables common for all OSEv3 hosts
```

```
[OSEv3:vars]
# SSH user, this user should allow ssh based auth without requiring a password
ansible_ssh_user=root

# If ansible_ssh_user is not root, ansible_become must be set to true
#ansible_become=true

openshift_deployment_type=openshift-enterprise

# uncomment the following to enable htpasswd authentication; defaults to
DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge': 'true', 'kind':
'HTPasswdPasswordIdentityProvider'}]

# host group for masters
[masters]
master.example.com

# host group for etcd
[etcd]
master.example.com

# host group for nodes, includes region info
[nodes]
master.example.com openshift_node_group_name='node-config-master'
node1.example.com openshift_node_group_name='node-config-compute'
node2.example.com openshift_node_group_name='node-config-compute'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'
```

重要

请参阅 配置节点主机标签，以确保您了解从 OpenShift Container Platform 3.9 开始的默认节点选择器要求和节点标签注意事项。

要使用本例，修改该文件以符合您的环境和规格，并将它保存为 *~/etc/ansible/hosts*。

5.2.2. 单个 Master、多个 etcd 和多个节点

下表描述了单个 master、三个 etcd 主机、两个用于托管用户应用程序的节点，以及两个具有 node-role.kubernetes.io/infra=true 标签的节点：

主机名	安装的组件/角色
master.example.com	Master 和节点
etcd1.example.com	etcd
etcd2.example.com	
etcd3.example.com	

主机名	安装的组件/角色
node1.example.com	Compute 节点
node2.example.com	
infra-node1.example.com	专用基础架构节点
infra-node2.example.com	

您可以看到以下清单文件的 [masters]、[nodes] 和 [etcd] 部分中出现的这些示例主机：

单个 Master、多个 etcd 和多节点清单文件

```
# Create an OSEv3 group that contains the masters, nodes, and etcd groups
[OSEv3:children]
masters
nodes
etcd

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=openshift-enterprise

# uncomment the following to enable htpasswd authentication; defaults to
DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge': 'true', 'kind':
'HTPasswdPasswordIdentityProvider'}]

# host group for masters
[masters]
master.example.com

# host group for etcd
[etcd]
etcd1.example.com
etcd2.example.com
etcd3.example.com

# host group for nodes, includes region info
[nodes]
master.example.com openshift_node_group_name='node-config-master'
node1.example.com openshift_node_group_name='node-config-compute'
node2.example.com openshift_node_group_name='node-config-compute'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'
```

重要

请参阅 **配置节点主机标签**，**以确保您了解从 OpenShift Container Platform 3.9 开始的默认节点选择器要求和节点标签注意事项。**

要使用本例，修改该文件以符合您的环境和规格，并将它保存为 */etc/ansible/hosts*。

5.3. 多个主页示例

您可以配置具有多个 master、多个 etcd 主机和多个节点的环境。为高可用性（HA）配置多个 master 可确保集群没有单点故障。

注意

在安装后从单一 master 集群移动到多个 master 集群不被支持。

在配置多个 master 时，集群安装过程支持原生的高可用性（HA）方法。此方法利用了 OpenShift Container Platform 中内置的原生 HA master 功能，并可与任何负载均衡解决方案结合使用。

如果在清单文件的 [lb] 部分中定义了主机，Ansible 会自动安装并配置 HAProxy 作为 master 的负载均衡解决方案。如果没有定义主机，则假设您已预先配置了您选择的外部负载均衡解决方案，以在所有 master 主机上平衡 master API（端口 8443）。

注意

此 HAProxy 负载均衡器旨在演示 API 服务器的 HA 模式，不建议在生产环境中使用。如果您要部署到云供应商中，红帽建议部署基于云原生 TCP 的负载均衡器，或者采取其他步骤来提供高度可用的负载均衡器。

HAProxy 负载均衡器仅用于对 API 服务器的流量进行负载平衡，且不会对任何用户应用程序流量进行负载平衡。

对于外部负载平衡解决方案，您必须有：

- 为 SSL passthrough 配置预创建的负载均衡器虚拟 IP(VIP)。
- 由 openshift_master_api_port 值 （默认为 8443）指定的端口上的 VIP 监听，并从那个端口上返回到所有 master 主机的代理。
- 在 DNS 中注册的 VIP 的域名。
  - 域名将成为 OpenShift Container Platform 安装程序中的 **openshift_master_cluster_public_hostname** 和 **openshift_master_cluster_hostname** 的值。

如需更多信息，请参阅 Github 中的 External Load Balancer 集成示例。有关高可用性 master 架构的更多信息，请参阅 Kubernetes Infrastructure。

注意

集群安装过程目前不支持主动被动（active-passibe）设置中的多个 HAProxy 负载均衡器。有关安装后的详情，请查看 Load Balancer 管理文档。

要配置多个 master，请参阅使用 多个 etcd 的多 Master

5.3.1. 多个使用原生 HA 的、带有外部集群 etcd 的 Master。

下面描述了使用 原生 HA 方法的三个 master 的示例环境：一个 HAProxy 负载均衡器，三个 etcd 主机，两个用于托管用户应用程序的节点，具有 用于托管专用基础架构 的 node-role.kubernetes.io/infra=true 标签的两个节点：

主机名	安装的组件/角色
master1.example.com	Master（使用原生 HA 进行集群）和节点
master2.example.com	
master3.example.com	
lb.example.com	仅平衡 API 主端点的 HAProxy
etcd1.example.com	etcd
etcd2.example.com	
etcd3.example.com	
node1.example.com	Compute 节点
node2.example.com	
infra-node1.example.com	专用基础架构节点
infra-node2.example.com	

您可以看到以下清单文件的 [masters]、[etcd]、[lb] 和 [nodes] 部分中出现的这些示例主机：

使用 HAProxy 清单文件的多个 Master

```
# Create an OSEv3 group that contains the master, nodes, etcd, and lb groups.
# The lb group lets Ansible configure HAProxy as the load balancing solution.
# Comment lb out if your load balancer is pre-configured.
[OSEv3:children]
masters
nodes
etcd
lb

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=openshift-enterprise

# uncomment the following to enable htpasswd authentication; defaults to
DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge': 'true', 'kind':
'HTPasswdPasswordIdentityProvider'}]
```

```
# Native high availbility cluster method with optional load balancer.
# If no lb group is defined installer assumes that a load balancer has
# been preconfigured. For installation the value of
# openshift_master_cluster_hostname must resolve to the load balancer
# or to one or all of the masters defined in the inventory if no load
# balancer is present.
openshift_master_cluster_method=native
openshift_master_cluster_hostname=openshift-internal.example.com
openshift_master_cluster_public_hostname=openshift-cluster.example.com

# apply updated node defaults
openshift_node_groups=[{'name': 'node-config-all-in-one', 'labels': ['node-
role.kubernetes.io/master=true', 'node-role.kubernetes.io/infra=true', 'node-
role.kubernetes.io/compute=true'], 'edits': [{ 'key': 'kubeletArguments.pods-per-core','value': ['20']}]}]

# host group for masters
[masters]
master1.example.com
master2.example.com
master3.example.com

# host group for etcd
[etcd]
etcd1.example.com
etcd2.example.com
etcd3.example.com

# Specify load balancer host
[lb]
lb.example.com

# host group for nodes, includes region info
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master'
node1.example.com openshift_node_group_name='node-config-compute'
node2.example.com openshift_node_group_name='node-config-compute'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'
```

重要

请参阅 **配置节点主机标签**，**以确保您了解从 OpenShift Container Platform 3.9 开始的默认节点选择器要求和节点标签注意事项。**

要使用本例，修改该文件以符合您的环境和规格，并将它保存为 *~/etc/ansible/hosts*。

5.3.2. 多个使用原生 HA 的 Master

下面描述了使用 原生 HA 方法的三个 master 的示例环境（每个主机上的 etcd 作为静态 pod 运行），一个 HAProxy 负载均衡器，两个用于托管用户应用程序的 节点，具有 用于托管专用基础架构 的 node-role.kubernetes.io/infra=true 标签的两个节点：

主机名	安装的组件/角色
master1.example.com	Master（使用原生 HA 进行集群）和 etcd 在每个主机上运行作为静态 pod 的节点
master2.example.com	
master3.example.com	
lb.example.com	仅平衡 API 主端点的 HAProxy
node1.example.com	Compute 节点
node2.example.com	
infra-node1.example.com	专用基础架构节点
infra-node2.example.com	

您可以看到以下清单文件的 [masters]、[etcd]、[lb] 和 [nodes] 部分中出现的这些示例主机：

```
# Create an OSEv3 group that contains the master, nodes, etcd, and lb groups.
# The lb group lets Ansible configure HAProxy as the load balancing solution.
# Comment lb out if your load balancer is pre-configured.
[OSEv3:children]
masters
nodes
etcd
lb

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=openshift-enterprise

# uncomment the following to enable htpasswd authentication; defaults to
DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge': 'true', 'kind':
'HTPasswdPasswordIdentityProvider'}]

# Native high availability cluster method with optional load balancer.
# If no lb group is defined installer assumes that a load balancer has
# been preconfigured. For installation the value of
# openshift_master_cluster_hostname must resolve to the load balancer
# or to one or all of the masters defined in the inventory if no load
# balancer is present.
openshift_master_cluster_method=native
openshift_master_cluster_hostname=openshift-internal.example.com
openshift_master_cluster_public_hostname=openshift-cluster.example.com

# host group for masters
[masters]
```

```
master1.example.com
master2.example.com
master3.example.com

# host group for etcd
[etcd]
master1.example.com
master2.example.com
master3.example.com

# Specify load balancer host
[lb]
lb.example.com

# host group for nodes, includes region info
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master'
node1.example.com openshift_node_group_name='node-config-compute'
node2.example.com openshift_node_group_name='node-config-compute'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'
```

重要

请参阅 **配置节点主机标签**，以确保您了解从 OpenShift Container Platform 3.9 开始的默认节点选择器要求和节点标签注意事项。

要使用本例，修改该文件以符合您的环境和规格，并将它保存为 */etc/ansible/hosts*。

第 6 章 安装 OPENSHIFT CONTAINER PLATFORM

要安装 OpenShift Container Platform 集群，您需要运行一系列 Ansible playbook。

重要

红帽不支持使用 **--tags** 或 **--check** 选项运行 Ansible playbook。

注意

要将 OpenShift Container Platform 作为独立 registry 安装，请参阅安装独立 registry。

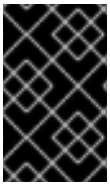
6.1. 先决条件

安装 OpenShift Container Platform 前，准备集群主机：

- 查看 系统和环境要求。
- 如果您的集群规模较大，请查看《缩放和性能指南以优化安装时间。
- 准备主机。此过程包括为每个组件类型验证系统和环境要求，安装和配置 docker 服务，以及安装 Ansible 版本 2.6 或更高版本。您必须安装 Ansible 以运行安装 playbook。
- 配置清单文件以定义您的环境和 OpenShift Container Platform 集群配置。您的初始安装和将来的集群升级都基于此清单文件。
- 如果要在 Red Hat Enterprise Linux 上安装 OpenShift Container Platform，请决定是使用 RPM 还是系统容器安装方法。RHEL Atomic Host 系统需要使用系统容器。

6.1.1. 运行基于 RPM 的安装程序

基于 RPM 的安装程序使用 RPM 软件包安装的 Ansible 运行本地主机上可用的 playbook 和配置文件。



重要

不要在 **nohup** 下运行 OpenShift Ansible playbook。将 **nohup** 与 playbook 结合使用会导致创建文件描述符但不会关闭。因此，系统会消耗掉所有打开的文件，playbook 会失败。

要运行基于 RPM 的安装程序：

1. 进入 playbook 目录并运行 *prerequisites.yml* playbook。这个 playbook 会安装所需的软件包（若有），并修改容器运行时。除非需要配置容器运行时，在首次部署集群前仅运行一次此 playbook:

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook [-i /path/to/inventory] \ 1
  playbooks/prerequisites.yml
```

1 如果您的清单文件不在 */etc/ansible/hosts* 目录中，使用 **-i** 指定清单文件的路径。

2. 切换到 playbook 目录，运行 *deploy_cluster.yml* playbook 来启动集群安装：

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook [-i /path/to/inventory] \ 1
    playbooks/deploy_cluster.yml
```

1 如果您的清单文件不在 */etc/ansible/hosts* 目录中，使用 **-i** 指定清单文件的路径。

. 如果安装成功，验证安装。 如果安装失败，重试安装。

6.1.2. 运行容器化安装程序

openshift3/ose-ansible 镜像是 OpenShift Container Platform 安装程序的容器化版本。这个安装程序镜像提供和基于 RPM 的安装程序相同的功能，但它可在容器化环境中运行，该环境中提供所有依赖项，而不是直接安装到主机上。使用它的唯一要求是能够运行容器。

6.1.2.1. 将安装程序作为系统容器运行

安装程序镜像可用作 系统容器。系统容器在传统 docker 服务外存储并运行。这可启用从其中一个目标主机运行安装程序镜像，而无需考虑在主机中重启 docker。

要使用 Atomic CLI 作为运行一次系统容器运行安装程序，以 root 用户身份执行以下步骤：

1. 运行 *prerequisites.yml* playbook:

```
# atomic install --system \
    --storage=ostree \
    --set INVENTORY_FILE=/path/to/inventory \ 1
    --set PLAYBOOK_FILE=/usr/share/ansible/openshift-ansible/playbooks/prerequisites.yml \
    --set OPTS="-v" \
    registry.redhat.io/openshift3/ose-ansible:v3.11
```

1 指定清单文件本地主机上的位置。

此命令使用指定的清单文件和 root 用户的 SSH 配置来运行一组需要完成的任务。

2. 运行 *deploy_cluster.yml* playbook:

```
# atomic install --system \
    --storage=ostree \
    --set INVENTORY_FILE=/path/to/inventory \ 1
    --set PLAYBOOK_FILE=/usr/share/ansible/openshift-ansible/playbooks/deploy_cluster.yml
\
    --set OPTS="-v" \
    registry.redhat.io/openshift3/ose-ansible:v3.11
```

1 指定清单文件本地主机上的位置。

此命令使用指定的清单文件和 root 用户的 SSH 配置启动集群安装。它记录终端的输出，并将数据保存到 */var/log/ansible.log* 文件。第一次运行这个命令时，镜像会导入到 OSTree 存储中（系统容器使用这个而不是 docker 守护进程存储）。在后续运行时，它将重复使用存储的镜像。

如果出于某种原因安装失败，在重新运行安装程序前查看 已知问题 以查看任何具体步骤或临时解决方案。

6.1.2.2. 运行其他 playbook

您可以使用 **PLAYBOOK_FILE** 环境变量指定您要使用容器化安装程序运行的其他 playbook。**PLAYBOOK_FILE** 的默认值为 */usr/share/ansible/openshift-ansible/playbooks/deploy_cluster.yml*，它是主要的集群安装 playbook，当您可以把它设置为容器内的另一个 playbook 的路径。

例如，要在安装前运行 预安装检查 playbook，请使用以下命令：

```
# atomic install --system \
    --storage=ostree \
    --set INVENTORY_FILE=/path/to/inventory \
    --set PLAYBOOK_FILE=/usr/share/ansible/openshift-ansible/playbooks/openshift-checks/pre-
install.yml \ 1
    --set OPTS="-v" \ 2
    registry.redhat.io/openshift3/ose-ansible:v3.11
```

1. 将 **PLAYBOOK_FILE** 设置为从 *playbooks/* 目录开始的 playbook 的完整路径。playbook 位于与基于 RPM 的安装程序相同的位置。
2. 将 **OPTS** 设置为在 **ansible-playbook** 中添加命令行的选项。

6.1.2.3. 将安装程序作为容器运行

安装程序镜像也可以作为一个 docker 容器在 docker 可以运行的任何地方运行。

警告

这个方法不能在被配置的一个主机上运行的安装程序使用，因为安装程序可能会在主机上重启 docker，并破坏安装过程。

注意

虽然此方法和上述系统容器方法使用相同的镜像，但它们在不同入口点和上下文中运行，因此运行时参数会不同。

至少，当作为 docker 容器运行安装程序时，您必须提供：

- SSH 密钥，以便 Ansible 可以访问您的主机。
- Ansible 清单文件。
- 针对该清单运行的 Ansible playbook 的位置。

以下是如何使用 **docker** 运行安装的示例，必须由具有 **docker** 访问权限的非root 用户运行：

1. 首先，运行 *prerequisites.yml* playbook:

```
$ docker run -t -u `id -u` \ 1
```

```
    -v $HOME/.ssh/id_rsa:/opt/app-root/src/.ssh/id_rsa:Z \ 2
    -v $HOME/ansible/hosts:/tmp/inventory:Z \ 3
    -e INVENTORY_FILE=/tmp/inventory \ 4
    -e PLAYBOOK_FILE=playbooks/prerequisites.yml \ 5
    -e OPTS="-v" \ 6
    registry.redhat.io/openshift3/ose-ansible:v3.11
```

1. **-U 'id -u' 使容器与当前用户使用相同的 UID 运行，这样用户就可以使用容器中的 SSH 密钥。只有所有者才有 SSH 私钥的读取权限。**

2. **-v $HOME/.ssh/id_rsa:/opt/app-root/src/.ssh/id_rsa:Z 挂载您的 SSH 密钥 $HOME/.ssh/id_rsa，它位于容器用户的 $HOME/.ssh 目录下。*/opt/app-root/src* 是容器中用户的 $HOME。如果您将 SSH 密钥挂载到不同位置，使用 -e ANSIBLE_PRIVATE_KEY_FILE=/the/mount/point 或设置 ansible_ssh_private_key_file=/the/mount/point 作为清单中的变量来将 Ansible 指向它。请注意，SSH 密钥使用 :Z 标志挂载。这个标志是必需的，以便容器可以读取其受限 SELinux 上下文下的 SSH 密钥。这也意味着您的原始 SSH 密钥文件将重新标记为类似 system_u:object_r:container_file_t:s0:c113,c247。有关 :Z 的详情，请查看 docker-run(1) 手册页。请记住，在提供这些卷挂载规格时，这可能会造成意外后果。例如，如果您挂载（因此重新标记）整个 $HOME/.ssh 目录，它将阻断主机的 sshd 访问要登录的公钥。因此，您可以考虑使用 SSH 密钥或目录的副本，这样就不会影响原始文件标签。**

3. 4. **-v $HOME/ansible/hosts:/tmp/inventory:Z 和 -e INVENTORY_FILE=/tmp/inventory 将静态 Ansible 清单文件挂载到容器中，作为 */tmp/inventory*，并将对应的环境变量设置为指向它。与使用 SSH 密钥一样，可能需要使用 :Z 标志来根据现有标签在容器中读取来重新标记清单文件 SELinux 标签。对于用户 $HOME 目录中的文件，这很可能是必需的。您可能希望在挂载清单前将清单复制到专用位置。您可以指定 INVENTORY_URL 环境变量来从 web 服务器下载清单文件，或通过使用 DYNAMIC_SCRIPT_URL 参数指定一个可以提供动态清单的可执行脚步来动态生成清单文件。**

5. **-e PLAYBOOK_FILE=playbooks/prerequisites.yml 指定作为 openshift-ansible 内容顶级目录中的相对路径运行的 playbook。在本例中，可以指定先决条件 playbook。您还可以指定 RPM 的完整路径，或者指定到容器中任何其他 playbook 文件的路径。**

6. **-e OPTS="-v" 为容器中运行的 ansible-playbook 命令提供任意命令行选项。在这个示例中，使用 -v 来提高输出的详细程度。**

2. **接下来，运行 *deploy_cluster.yml* playbook 来启动集群安装：**

```
$ docker run -t -u `id -u` \
    -v $HOME/.ssh/id_rsa:/opt/app-root/src/.ssh/id_rsa:Z \
    -v $HOME/ansible/hosts:/tmp/inventory:Z \
    -e INVENTORY_FILE=/tmp/inventory \
    -e PLAYBOOK_FILE=playbooks/deploy_cluster.yml \
    -e OPTS="-v" \
    registry.redhat.io/openshift3/ose-ansible:v3.11
```

6.1.2.4. 为 OpenStack 运行安装 Playbook

要在现有 OpenStack 安装上安装 OpenShift Container Platform，使用 OpenStack playbook。如需有关 playbook 的更多信息，包括详细的先决条件，请参阅 OpenStack Provisioning readme 文件。

要运行 playbook，运行以下命令：

```
$ ansible-playbook --user openshift \
  -i openshift-ansible/playbooks/openstack/inventory.py \
  -i inventory \
  openshift-ansible/playbooks/openstack/openshift-cluster/provision_install.yml
```

6.1.3. 关于安装 playbook

安装程序使用模块化 playbook，以便管理员可以根据需要安装特定的组件。通过分隔角色和 playbook 可以更好地将临时管理任务作为目标。这会增加安装过程中的控件并节省时间。

主要的安装 playbook */usr/share/ansible/openshift-ansible/playbooks/deploy_cluster.yml* 以特定顺序运行一组独立组件 playbook，安装程序会回到已完成阶段的最后进行报告。如果安装失败，会收到哪个阶段失败的信息以及 Ansible 运行中的错误信息。

重要

虽然 RHEL Atomic Host 支持以系统容器的形式运行 OpenShift Container Platform 服务，但安装方法使用 Ansible（在 RHEL Atomic Host 中不提供）。因此，基于 RPM 的安装程序必须在 RHEL 7 系统中运行。启动安装的主机可以，但不需要一定包含在 OpenShift Container Platform 集群中。另外，安装程序的容器化版本 也可作为系统容器使用，该容器可在 RHEL Atomic Host 系统中运行。

6.2. 重试安装

如果 Ansible 安装程序失败，您仍可以安装 OpenShift Container Platform:

1. 查看 已知问题以检查是否有具体步骤或临时解决方案。
2. 解决安装中的错误。
3. 确定是否需要卸载、重新安装或重试安装：
   - 如果您没有修改 SDN 配置或生成新证书，请重试安装。
   - 如果您修改了 SDN 配置，生成新证书，或者安装程序再次失败，您必须开始安装干净的操作系统，或者重新 卸载并安装。
   - 如果您使用虚拟机，请从新镜像启动或者 卸载并再次安装。
   - 如果使用裸机，需要先卸载并再次安装。
4. 重试安装：
   - 您可以再次运行 *deploy_cluster.yml* playbook。
   - 您可以运行剩余的单独安装 playbook。
     如果您只运行剩余的 playbook，请从失败的阶段开始运行 playbook，然后按顺序运行每个剩余的 playbook。使用以下命令运行每个 playbook：

     ```
     # ansible-playbook [-i /path/to/inventory] <playbook_file_location>
     ```

     下表按必须运行的顺序列出 playbook：

     表 6.1. 独立组件 Playbook 运行顺序

Playbook 名称	文件位置
健康检查	*/usr/share/ansible/openshift-ansible/playbooks/openshift-checks/pre-install.yml*
Node Bootstrap	*/usr/share/ansible/openshift-ansible/playbooks/openshift-node/bootstrap.yml*
etcd Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-etcd/config.yml*
NFS Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-nfs/config.yml*
Load Balancer Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-loadbalancer/config.yml*
Master Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-master/config.yml*
Master Additional Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-master/additional_config.yml*
Node Join	*/usr/share/ansible/openshift-ansible/playbooks/openshift-node/join.yml*
GlusterFS Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-glusterfs/config.yml*
Hosted Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-hosted/config.yml*
Monitoring Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-monitoring/config.yml*
Web Console Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-web-console/config.yml*
Admin Console Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-console/config.yml*
Metrics Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-metrics/config.yml*
metrics-server	*/usr/share/ansible/openshift-ansible/playbooks/metrics-server/config.yml*
Logging Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-logging/config.yml*

Playbook 名称	文件位置
Availability Monitoring Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-monitor-availability/config.yml*
Service Catalog Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-service-catalog/config.yml*
Management Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-management/config.yml*
Descheduler Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-descheduler/config.yml*
Node Problem Detector Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-node-problem-detector/config.yml*
Operator Lifecycle Manager (OLM) Install（技术预览）	*/usr/share/ansible/openshift-ansible/playbooks/olm/config.yml*

6.3. 验证安装

安装完成后：

1. 验证 master 是否已启动，节点是否已注册并报告为 Ready 状态。*在 master 主机*上，作为 root 运行以下命令：

```
# oc get nodes
NAME                STATUS    ROLES     AGE       VERSION
master.example.com  Ready     master    7h        v1.9.1+a0ce1bc657
node1.example.com   Ready     compute   7h        v1.9.1+a0ce1bc657
node2.example.com   Ready     compute   7h        v1.9.1+a0ce1bc657
```

2. 要验证是否已正确安装了 Web 控制台，使用 master 主机名和 Web 控制台端口号通过 Web 浏览器访问 Web 控制台。
例如，对于主机名为 **master.openshift.com** 并使用默认的 **8443** 端口的 master 主机，Web 控制台 URL 为 **https://master.openshift.com:8443/console**。

验证多个 etcd 主机
如果安装了多个 etcd 主机：

1. 首先，验证是否安装了提供 **etcdctl** 命令的 etcd 软件包：

```
# yum install etcd
```

2. 在 master 主机上，验证 etcd 集群健康状况，在以下内容中使用 etcd 主机的 FQDN 替换相应的内容：

```
# etcdctl -C \
```

```
https://etcd1.example.com:2379,https://etcd2.example.com:2379,https://etcd3.example.com:237
9 \
    --ca-file=/etc/origin/master/master.etcd-ca.crt \
    --cert-file=/etc/origin/master/master.etcd-client.crt \
    --key-file=/etc/origin/master/master.etcd-client.key cluster-health
```

3. 同时还要验证成员列表是否正确：

```
# etcdctl -C \

https://etcd1.example.com:2379,https://etcd2.example.com:2379,https://etcd3.example.com:237
9 \
    --ca-file=/etc/origin/master/master.etcd-ca.crt \
    --cert-file=/etc/origin/master/master.etcd-client.crt \
    --key-file=/etc/origin/master/master.etcd-client.key member list
```

使用 HAProxy 验证多个 Master
如果使用 HAProxy 作为负载均衡器安装多个 master，请打开以下 URL 并检查 HAProxy 的状态：

```
http://<lb_hostname>:9000 (1)
```

(1) **提供清单文件的 [lb] 部分中列出的负载均衡器主机名。**

您可以通过咨询 HAProxy 配置文档来验证您的安装。

6.4. （可选）安全构建

运行 docker build 是一个需要特权的过程，因此容器可能需要访问比某些多租户环境中允许的更多节点。如果您不信任您的用户，可以在安装后配置更安全的选项。禁用集群上的 Docker 构建，要求用户在集群外构建镜像。有关此可选过程的更多信息，请参阅通过策略保护构建。

6.5. 已知问题

- **在多个 master 集群中的故障切换中，控制器管理器可能会过度修正，这会导致系统运行超过预期的 pod 数。然而，这只是一个瞬时事件，系统会随时进行修正。详情请查看 https://github.com/kubernetes/kubernetes/issues/10030。**
- **由于一个已知问题，在运行安装后，如果为任何组件置备了 NFS 卷，则可能会创建以下目录，无论它们的组件是否被部署到 NFS 卷中：**
  - */exports/logging-es*
  - */exports/logging-es-ops/*
  - */exports/metrics/*
  - */exports/prometheus*
  - */exports/prometheus-alertbuffer/*
  - */exports/prometheus-alertmanager/*
    您可根据需要在安装后删除这些目录。

6.6. 下一步是什么？

现在，您已有一个可以正常工作的 OpenShift Container Platform 实例，您可以：

- 部署 集成的容器镜像 registry。
- 部署 路由器。

第 7 章 断开连接的安装

通常情况下，数据中心的部分环境可能无法访问互联网，甚至无法通过代理服务器访问。您仍可在这些环境中安装 OpenShift Container Platform，但需要先下载所需的软件和镜像，并将其提供给离线环境中。

在节点主机安装组件可用后，按照标准安装步骤安装 OpenShift Container Platform。

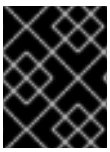
安装 OpenShift Container Platform 后，您必须使拉取的 S2I 构建器镜像可供集群使用。

7.1. 先决条件

- 查看 OpenShift Container Platform 的整体架构并规划您的环境拓扑。
- 有一个 Red Hat Enterprise Linux（RHEL）7 服务器，并有 root 访问权限，且可以访问互联网和至少 110 GB 磁盘空间。您可以将所需的软件存储库和容器镜像下载到这台计算机。
- 计划在断开连接的环境中维护一个 webserver，以便为镜像的存储库提供服务。 您可以通过网络或者断开连接的部署中的物理介质将库从互联网连接的主机复制到这个 webserver 服务器中。
- 提供源控制存储库。安装后，您的节点必须访问源代码存储库中的源代码，如 Git。
  在 OpenShift Container Platform 中构建应用程序时，您的构建可能会包含外部依赖项，如 Maven Repository 或 Ruby 应用程序的 Gem 文件。
- 在断开连接的环境中提供 registry。选项包括：
  - 安装一个独立 OpenShift Container Platform registry。
  - 使用作为容器镜像 registry 的 Red Hat Satellite 6.1服务器。

7.2. 获取所需的软件包和镜像

在断开连接的环境中安装 OpenShift Container Platform 前，先获取所需的镜像和组件并将它们存储在存储库中。



重要

您必须在与断开连接的环境中具有与集群相同的架构的系统上获取所需的镜像和软件组件。

7.2.1. 获取 OpenShift Container Platform 软件包

在有互联网连接的 RHEL 7 服务器中，同步软件仓库：

1. 要确保在同步该存储库后不会删除软件包，请导入 GPG 密钥：

```
$ rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

2. 使用红帽客户门户网站注册服务器。您必须使用与可访问 OpenShift Container Platform 订阅的帐户关联的凭证：

```
$ subscription-manager register
```

3. 从 RHSM 获取最新的订阅数据：

```
$ subscription-manager refresh
```

4. 附加提供 OpenShift Container Platform 频道的订阅。

   a. 找到提供 OpenShift Container Platform 频道的可用订阅池：

   ```
   $ subscription-manager list --available --matches '*OpenShift*'
   ```

   b. 为提供 OpenShift Container Platform 的订阅附加一个池 ID:

   ```
   $ subscription-manager attach --pool=<pool_id>
   $ subscription-manager repos --disable="*"
   ```

5. 仅启用 OpenShift Container Platform 3.11 需要的存储库：

   - 对于 x86_64 服务器中的云安装和内部安装，请运行以下命令：

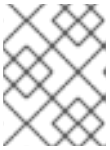
   ```
   # subscription-manager repos \
       --enable="rhel-7-server-rpms" \
       --enable="rhel-7-server-extras-rpms" \
       --enable="rhel-7-server-ose-3.11-rpms" \
       --enable="rhel-7-server-ansible-2.9-rpms"
   ```

   - 对于 IBM POWER8 服务器中的内部安装，请运行以下命令：

   ```
   # subscription-manager repos \
       --enable="rhel-7-for-power-le-rpms" \
       --enable="rhel-7-for-power-le-extras-rpms" \
       --enable="rhel-7-for-power-le-optional-rpms" \
       --enable="rhel-7-server-ansible-2.9-for-power-le-rpms" \
       --enable="rhel-7-server-for-power-le-rhscl-rpms" \
       --enable="rhel-7-for-power-le-ose-3.11-rpms"
   ```

   - 对于 IBM POWER9 服务器中的内部安装，请运行以下命令：

   ```
   # subscription-manager repos \
       --enable="rhel-7-for-power-9-rpms" \
       --enable="rhel-7-for-power-9-extras-rpms" \
       --enable="rhel-7-for-power-9-optional-rpms" \
       --enable="rhel-7-server-ansible-2.9-for-power-9-rpms" \
       --enable="rhel-7-server-for-power-9-rhscl-rpms" \
       --enable="rhel-7-for-power-9-ose-3.11-rpms"
   ```



注意

旧版本的 OpenShift Container Platform 3.11 仅支持 Ansible 2.6。Playbook 的最新版本现在支持 Ansible 2.9，这是首选的版本。

6. 安装所需的软件包：

```
$ sudo yum -y install yum-utils createrepo docker git
```

yum-utils 软件包提供 reposync 工具，可让您镜像 yum 存储库，您可以使用 createrepo 软件包从目录中创建可用的 yum 存储库。

7. 创建一个目录将软件保存在服务器的存储中，或者 USB 驱动器或者其它外部设备中：

```
$ mkdir -p </path/to/repos>
```

重要

如果您可以将这个服务器重新连接到断开连接的 LAN，并使用它作为存储库服务器，请在本地存储这些文件。如果无法使用 USB 连接的存储，可以将软件传送到断开连接的 LAN 中的存储库服务器中。

8. 同步软件包并为每个软件包创建程序库。

- 对于 x86_64 服务器中的内部安装，请运行以下命令：

```
$ for repo in \
  rhel-7-server-rpms \
  rhel-7-server-extras-rpms \
  rhel-7-server-ansible-2.9-rpms \
  rhel-7-server-ose-3.11-rpms
do
  reposync --gpgcheck -lm --repoid=${repo} --download_path=</path/to/repos> (1)
  createrepo -v </path/to/repos/>${repo} -o </path/to/repos/>${repo} (2)
done
```

(1)(2) 提供您创建的目录的路径。

- 对于 IBM POWER8 服务器中的内部安装，请运行以下命令：

```
$ for repo in \
  rhel-7-for-power-le-rpms \
  rhel-7-for-power-le-extras-rpms \
  rhel-7-for-power-le-optional-rpms \
  rhel-7-server-ansible-2.9-for-power-le-rpms \
  rhel-7-server-for-power-le-rhscl-rpms \
  rhel-7-for-power-le-ose-3.11-rpms
do
  reposync --gpgcheck -lm --repoid=${repo} --download_path=</path/to/repos> (1)
  createrepo -v </path/to/repos/>${repo} -o </path/to/repos/>${repo} (2)
done
```

(1)(2) 提供您创建的目录的路径。

- 对于 IBM POWER9 服务器中的内部安装，请运行以下命令：

```
$ for repo in \
  rhel-7-for-power-9-rpms \
  rhel-7-for-power-9-extras-rpms \
  rhel-7-for-power-9-optional-rpms \
  rhel-7-server-ansible-2.9-for-power-9-rpms \
```

```
  rhel-7-server-for-power-9-rhscl-rpms \
  rhel-7-for-power-9-ose-3.11-rpms
do
  reposync --gpgcheck -lm --repoid=${repo} --download_path=/<path/to/repos> (1)
  createrepo -v </path/to/repos/>${repo} -o </path/to/repos/>${repo} (2)
done
```

(1)(2)提供您创建的目录的路径。

7.2.2. 获取镜像

拉取所需的容器镜像：

1. 启动 Docker 守护进程：

```
$ systemctl start docker
```

2. 拉取所有所需的 OpenShift Container Platform 基础架构组件镜像。将 <tag> 替换为要安装的版本。例如，为最新版本指定 v3.11.634。您可以指定不同的次版本。如果使用容器化安装程序，除了拉取这些所需镜像外，还需要拉取 registry.redhat.io/openshift3/ose-ansible:v3.11：

```
$ docker pull registry.redhat.io/openshift3/apb-base:<tag>
$ docker pull registry.redhat.io/openshift3/apb-tools:<tag>
$ docker pull registry.redhat.io/openshift3/automation-broker-apb:<tag>
$ docker pull registry.redhat.io/openshift3/csi-attacher:<tag>
$ docker pull registry.redhat.io/openshift3/csi-driver-registrar:<tag>
$ docker pull registry.redhat.io/openshift3/csi-livenessprobe:<tag>
$ docker pull registry.redhat.io/openshift3/csi-provisioner:<tag>
$ docker pull registry.redhat.io/openshift3/grafana:<tag>
$ docker pull registry.redhat.io/openshift3/kuryr-controller:<tag>
$ docker pull registry.redhat.io/openshift3/kuryr-cni:<tag>
$ docker pull registry.redhat.io/openshift3/local-storage-provisioner:<tag>
$ docker pull registry.redhat.io/openshift3/manila-provisioner:<tag>
$ docker pull registry.redhat.io/openshift3/mariadb-apb:<tag>
$ docker pull registry.redhat.io/openshift3/mediawiki:<tag>
$ docker pull registry.redhat.io/openshift3/mediawiki-apb:<tag>
$ docker pull registry.redhat.io/openshift3/mysql-apb:<tag>
$ docker pull registry.redhat.io/openshift3/ose-ansible-service-broker:<tag>
$ docker pull registry.redhat.io/openshift3/ose-cli:<tag>
$ docker pull registry.redhat.io/openshift3/ose-cluster-autoscaler:<tag>
$ docker pull registry.redhat.io/openshift3/ose-cluster-capacity:<tag>
$ docker pull registry.redhat.io/openshift3/ose-cluster-monitoring-operator:<tag>
$ docker pull registry.redhat.io/openshift3/ose-console:<tag>
$ docker pull registry.redhat.io/openshift3/ose-configmap-reloader:<tag>
$ docker pull registry.redhat.io/openshift3/ose-control-plane:<tag>
$ docker pull registry.redhat.io/openshift3/ose-deployer:<tag>
$ docker pull registry.redhat.io/openshift3/ose-descheduler:<tag>
$ docker pull registry.redhat.io/openshift3/ose-docker-builder:<tag>
$ docker pull registry.redhat.io/openshift3/ose-docker-registry:<tag>
$ docker pull registry.redhat.io/openshift3/ose-efs-provisioner:<tag>
$ docker pull registry.redhat.io/openshift3/ose-egress-dns-proxy:<tag>
$ docker pull registry.redhat.io/openshift3/ose-egress-http-proxy:<tag>
$ docker pull registry.redhat.io/openshift3/ose-egress-router:<tag>
$ docker pull registry.redhat.io/openshift3/ose-haproxy-router:<tag>
```

```
$ docker pull registry.redhat.io/openshift3/ose-hyperkube:<tag>
$ docker pull registry.redhat.io/openshift3/ose-hypershift:<tag>
$ docker pull registry.redhat.io/openshift3/ose-keepalived-ipfailover:<tag>
$ docker pull registry.redhat.io/openshift3/ose-kube-rbac-proxy:<tag>
$ docker pull registry.redhat.io/openshift3/ose-kube-state-metrics:<tag>
$ docker pull registry.redhat.io/openshift3/ose-metrics-server:<tag>
$ docker pull registry.redhat.io/openshift3/ose-node:<tag>
$ docker pull registry.redhat.io/openshift3/ose-node-problem-detector:<tag>
$ docker pull registry.redhat.io/openshift3/ose-operator-lifecycle-manager:<tag>
$ docker pull registry.redhat.io/openshift3/ose-ovn-kubernetes:<tag>
$ docker pull registry.redhat.io/openshift3/ose-pod:<tag>
$ docker pull registry.redhat.io/openshift3/ose-prometheus-config-reloader:<tag>
$ docker pull registry.redhat.io/openshift3/ose-prometheus-operator:<tag>
$ docker pull registry.redhat.io/openshift3/ose-recycler:<tag>
$ docker pull registry.redhat.io/openshift3/ose-service-catalog:<tag>
$ docker pull registry.redhat.io/openshift3/ose-template-service-broker:<tag>
$ docker pull registry.redhat.io/openshift3/ose-tests:<tag>
$ docker pull registry.redhat.io/openshift3/ose-web-console:<tag>
$ docker pull registry.redhat.io/openshift3/postgresql-apb:<tag>
$ docker pull registry.redhat.io/openshift3/registry-console:<tag>
$ docker pull registry.redhat.io/openshift3/snapshot-controller:<tag>
$ docker pull registry.redhat.io/openshift3/snapshot-provisioner:<tag>
$ docker pull registry.redhat.io/rhel7/etcd:3.2.28
```

3. 对于 x86_64 服务器中的内部安装，请拉取以下镜像。将 **<tag>** 替换为要安装的版本。例如，为最新版本指定 **v3.11.634**。您可以指定不同的次版本。

```
$ docker pull registry.redhat.io/openshift3/ose-efs-provisioner:<tag>
```

4. 为可选组件拉取所有所需的 OpenShift Container Platform 组件镜像。将 **<tag>** 替换为要安装的版本。例如，为最新版本指定 **v3.11.634**。您可以指定不同的次版本。

   - 对于 x86_64 服务器中的内部安装，请运行以下命令：

```
$ docker pull registry.redhat.io/openshift3/metrics-cassandra:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-hawkular-metrics:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-hawkular-openshift-agent:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-heapster:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-schema-installer:<tag>
$ docker pull registry.redhat.io/openshift3/oauth-proxy:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-curator5:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-elasticsearch5:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-eventrouter:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-fluentd:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-kibana5:<tag>
$ docker pull registry.redhat.io/openshift3/prometheus:<tag>
$ docker pull registry.redhat.io/openshift3/prometheus-alertmanager:<tag>
$ docker pull registry.redhat.io/openshift3/prometheus-node-exporter:<tag>
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-postgresql
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-memcached
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-app-ui
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-app
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-embedded-ansible
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-httpd
$ docker pull registry.redhat.io/cloudforms46/cfme-httpd-configmap-generator
$ docker pull registry.redhat.io/rhgs3/rhgs-server-rhel7
```

```
$ docker pull registry.redhat.io/rhgs3/rhgs-volmanager-rhel7
$ docker pull registry.redhat.io/rhgs3/rhgs-gluster-block-prov-rhel7
$ docker pull registry.redhat.io/rhgs3/rhgs-s3-server-rhel7
```

- 对于 IBM POWER8 或 IBM POWER9 服务器中的内部安装，请运行以下命令：

```
$ docker pull registry.redhat.io/openshift3/metrics-cassandra:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-hawkular-openshift-agent:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-heapster:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-schema-installer:<tag>
$ docker pull registry.redhat.io/openshift3/oauth-proxy:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-curator5:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-elasticsearch5:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-eventrouter:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-fluentd:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-kibana5:<tag>
$ docker pull registry.redhat.io/openshift3/prometheus:<tag>
$ docker pull registry.redhat.io/openshift3/prometheus-alert-buffer:<tag>
$ docker pull registry.redhat.io/openshift3/prometheus-alertmanager:<tag>
$ docker pull registry.redhat.io/openshift3/prometheus-node-exporter:<tag>
```

重要

对于红帽支持，rhgs3/ 镜像需要一个聚合模式订阅。

5. 拉取要在 OpenShift Container Platform 环境中使用的经过认证的 Source-to-Image（S2I） 构建器镜像。
通过指定版本号来确保指定正确的标签。如需了解有关镜像兼容性的详细信息，请参阅 OpenShift 和 Atomic Platform Tested Integrations 中的 S2I 表。

   您可以拉取以下镜像：

```
$ docker pull registry.redhat.io/jboss-amq-6/amq63-openshift:<tag>
$ docker pull registry.redhat.io/jboss-datagrid-7/datagrid71-openshift:<tag>
$ docker pull registry.redhat.io/jboss-datagrid-7/datagrid71-client-openshift:<tag>
$ docker pull registry.redhat.io/jboss-datavirt-6/datavirt63-openshift:<tag>
$ docker pull registry.redhat.io/jboss-datavirt-6/datavirt63-driver-openshift:<tag>
$ docker pull registry.redhat.io/jboss-decisionserver-6/decisionserver64-openshift:
<tag>
$ docker pull registry.redhat.io/jboss-processserver-6/processserver64-openshift:
<tag>
$ docker pull registry.redhat.io/jboss-eap-6/eap64-openshift:<tag>
$ docker pull registry.redhat.io/jboss-eap-7/eap71-openshift:<tag>
$ docker pull registry.redhat.io/jboss-webserver-3/webserver31-tomcat7-openshift:
<tag>
$ docker pull registry.redhat.io/jboss-webserver-3/webserver31-tomcat8-openshift:
<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-2-rhel7:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-agent-maven-35-rhel7:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-agent-nodejs-8-rhel7:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-slave-base-rhel7:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-slave-maven-rhel7:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-slave-nodejs-rhel7:<tag>
$ docker pull registry.redhat.io/rhscl/mongodb-32-rhel7:<tag>
```

```
$ docker pull registry.redhat.io/rhscl/mysql-57-rhel7:<tag>
$ docker pull registry.redhat.io/rhscl/perl-524-rhel7:<tag>
$ docker pull registry.redhat.io/rhscl/php-56-rhel7:<tag>
$ docker pull registry.redhat.io/rhscl/postgresql-95-rhel7:<tag>
$ docker pull registry.redhat.io/rhscl/python-35-rhel7:<tag>
$ docker pull registry.redhat.io/redhat-sso-7/sso70-openshift:<tag>
$ docker pull registry.redhat.io/rhscl/ruby-24-rhel7:<tag>
$ docker pull registry.redhat.io/redhat-openjdk-18/openjdk18-openshift:<tag>
$ docker pull registry.redhat.io/redhat-sso-7/sso71-openshift:<tag>
$ docker pull registry.redhat.io/rhscl/nodejs-6-rhel7:<tag>
$ docker pull registry.redhat.io/rhscl/mariadb-101-rhel7:<tag>
```

7.2.3. 导出镜像

如果您的环境无法访问您的内部网络，并需要使用物理介质来传输内容，请导出镜像到压缩文件。如果您的主机同时连接到互联网和您的内部网络，可以跳过以下步骤，然后继续准备并填充存储库服务器。

1. 创建用于存储压缩镜像的目录并进行以下改变：

```
$ mkdir </path/to/images>
$ cd </path/to/images>
```

2. 导出 OpenShift Container Platform 基础架构组件镜像。如果使用容器化安装程序，除这些所需镜像外还需要额外导出 registry.redhat.io/openshift3/ose-ansible:v3.11:

   - 对于 x86_64 服务器中的内部安装，请运行以下命令：

```
$ docker save -o ose3-images.tar \
    registry.redhat.io/openshift3/apb-base \
    registry.redhat.io/openshift3/apb-tools \
    registry.redhat.io/openshift3/automation-broker-apb \
    registry.redhat.io/openshift3/csi-attacher \
    registry.redhat.io/openshift3/csi-driver-registrar \
    registry.redhat.io/openshift3/csi-livenessprobe \
    registry.redhat.io/openshift3/csi-provisioner \
    registry.redhat.io/openshift3/grafana \
    registry.redhat.io/openshift3/kuryr-controller \
    registry.redhat.io/openshift3/kuryr-cni \
    registry.redhat.io/openshift3/local-storage-provisioner \
    registry.redhat.io/openshift3/manila-provisioner \
    registry.redhat.io/openshift3/mariadb-apb \
    registry.redhat.io/openshift3/mediawiki \
    registry.redhat.io/openshift3/mediawiki-apb \
    registry.redhat.io/openshift3/mysql-apb \
    registry.redhat.io/openshift3/ose-ansible-service-broker \
    registry.redhat.io/openshift3/ose-cli \
    registry.redhat.io/openshift3/ose-cluster-autoscaler \
    registry.redhat.io/openshift3/ose-cluster-capacity \
    registry.redhat.io/openshift3/ose-cluster-monitoring-operator \
    registry.redhat.io/openshift3/ose-console \
    registry.redhat.io/openshift3/ose-configmap-reloader \
    registry.redhat.io/openshift3/ose-control-plane \
    registry.redhat.io/openshift3/ose-deployer \
    registry.redhat.io/openshift3/ose-descheduler \
    registry.redhat.io/openshift3/ose-docker-builder \
```

```
    registry.redhat.io/openshift3/ose-docker-registry \
    registry.redhat.io/openshift3/ose-efs-provisioner \
    registry.redhat.io/openshift3/ose-egress-dns-proxy \
    registry.redhat.io/openshift3/ose-egress-http-proxy \
    registry.redhat.io/openshift3/ose-egress-router \
    registry.redhat.io/openshift3/ose-haproxy-router \
    registry.redhat.io/openshift3/ose-hyperkube \
    registry.redhat.io/openshift3/ose-hypershift \
    registry.redhat.io/openshift3/ose-keepalived-ipfailover \
    registry.redhat.io/openshift3/ose-kube-rbac-proxy \
    registry.redhat.io/openshift3/ose-kube-state-metrics \
    registry.redhat.io/openshift3/ose-metrics-server \
    registry.redhat.io/openshift3/ose-node \
    registry.redhat.io/openshift3/ose-node-problem-detector \
    registry.redhat.io/openshift3/ose-operator-lifecycle-manager \
    registry.redhat.io/openshift3/ose-ovn-kubernetes \
    registry.redhat.io/openshift3/ose-pod \
    registry.redhat.io/openshift3/ose-prometheus-config-reloader \
    registry.redhat.io/openshift3/ose-prometheus-operator \
    registry.redhat.io/openshift3/ose-recycler \
    registry.redhat.io/openshift3/ose-service-catalog \
    registry.redhat.io/openshift3/ose-template-service-broker \
    registry.redhat.io/openshift3/ose-tests \
    registry.redhat.io/openshift3/ose-web-console \
    registry.redhat.io/openshift3/postgresql-apb \
    registry.redhat.io/openshift3/registry-console \
    registry.redhat.io/openshift3/snapshot-controller \
    registry.redhat.io/openshift3/snapshot-provisioner \
    registry.redhat.io/rhel7/etcd:3.2.28 \
```

- 对于 IBM POWER8 或 IBM POWER9 服务器中的内部安装，请运行以下命令：

```
$ docker save -o ose3-images.tar \
    registry.redhat.io/openshift3/apb-base \
    registry.redhat.io/openshift3/apb-tools \
    registry.redhat.io/openshift3/automation-broker-apb \
    registry.redhat.io/openshift3/csi-attacher \
    registry.redhat.io/openshift3/csi-driver-registrar \
    registry.redhat.io/openshift3/csi-livenessprobe \
    registry.redhat.io/openshift3/csi-provisioner \
    registry.redhat.io/openshift3/grafana \
    registry.redhat.io/openshift3/kuryr-controller \
    registry.redhat.io/openshift3/kuryr-cni \
    registry.redhat.io/openshift3/local-storage-provisioner \
    registry.redhat.io/openshift3/manila-provisioner \
    registry.redhat.io/openshift3/mariadb-apb \
    registry.redhat.io/openshift3/mediawiki \
    registry.redhat.io/openshift3/mediawiki-apb \
    registry.redhat.io/openshift3/mysql-apb \
    registry.redhat.io/openshift3/ose-ansible-service-broker \
    registry.redhat.io/openshift3/ose-cli \
    registry.redhat.io/openshift3/ose-cluster-autoscaler \
    registry.redhat.io/openshift3/ose-cluster-capacity \
    registry.redhat.io/openshift3/ose-cluster-monitoring-operator \
    registry.redhat.io/openshift3/ose-console \
    registry.redhat.io/openshift3/ose-configmap-reloader \
```

```
    registry.redhat.io/openshift3/ose-control-plane \
    registry.redhat.io/openshift3/ose-deployer \
    registry.redhat.io/openshift3/ose-descheduler \
    registry.redhat.io/openshift3/ose-docker-builder \
    registry.redhat.io/openshift3/ose-docker-registry \
    registry.redhat.io/openshift3/ose-egress-dns-proxy \
    registry.redhat.io/openshift3/ose-egress-http-proxy \
    registry.redhat.io/openshift3/ose-egress-router \
    registry.redhat.io/openshift3/ose-haproxy-router \
    registry.redhat.io/openshift3/ose-hyperkube \
    registry.redhat.io/openshift3/ose-hypershift \
    registry.redhat.io/openshift3/ose-keepalived-ipfailover \
    registry.redhat.io/openshift3/ose-kube-rbac-proxy \
    registry.redhat.io/openshift3/ose-kube-state-metrics \
    registry.redhat.io/openshift3/ose-metrics-server \
    registry.redhat.io/openshift3/ose-node \
    registry.redhat.io/openshift3/ose-node-problem-detector \
    registry.redhat.io/openshift3/ose-operator-lifecycle-manager \
    registry.redhat.io/openshift3/ose-ovn-kubernetes \
    registry.redhat.io/openshift3/ose-pod \
    registry.redhat.io/openshift3/ose-prometheus-config-reloader \
    registry.redhat.io/openshift3/ose-prometheus-operator \
    registry.redhat.io/openshift3/ose-recycler \
    registry.redhat.io/openshift3/ose-service-catalog \
    registry.redhat.io/openshift3/ose-template-service-broker \
    registry.redhat.io/openshift3/ose-tests \
    registry.redhat.io/openshift3/ose-web-console \
    registry.redhat.io/openshift3/postgresql-apb \
    registry.redhat.io/openshift3/registry-console \
    registry.redhat.io/openshift3/snapshot-controller \
    registry.redhat.io/openshift3/snapshot-provisioner \
    registry.redhat.io/rhel7/etcd:3.2.28 \
```

3. **如果您为可选组件同步了镜像，请导出它们：**

   - **对于 x86_64 服务器中的内部安装，请运行以下命令：**

```
$ docker save -o ose3-optional-imags.tar \
    registry.redhat.io/openshift3/metrics-cassandra \
    registry.redhat.io/openshift3/metrics-hawkular-metrics \
    registry.redhat.io/openshift3/metrics-hawkular-openshift-agent \
    registry.redhat.io/openshift3/metrics-heapster \
    registry.redhat.io/openshift3/metrics-schema-installer \
    registry.redhat.io/openshift3/oauth-proxy \
    registry.redhat.io/openshift3/ose-logging-curator5 \
    registry.redhat.io/openshift3/ose-logging-elasticsearch5 \
    registry.redhat.io/openshift3/ose-logging-eventrouter \
    registry.redhat.io/openshift3/ose-logging-fluentd \
    registry.redhat.io/openshift3/ose-logging-kibana5 \
    registry.redhat.io/openshift3/prometheus \
    registry.redhat.io/openshift3/prometheus-alertmanager \
    registry.redhat.io/openshift3/prometheus-node-exporter \
    registry.redhat.io/cloudforms46/cfme-openshift-postgresql \
    registry.redhat.io/cloudforms46/cfme-openshift-memcached \
    registry.redhat.io/cloudforms46/cfme-openshift-app-ui \
    registry.redhat.io/cloudforms46/cfme-openshift-app \
```

```
    registry.redhat.io/cloudforms46/cfme-openshift-embedded-ansible \
    registry.redhat.io/cloudforms46/cfme-openshift-httpd \
    registry.redhat.io/cloudforms46/cfme-httpd-configmap-generator \
    registry.redhat.io/rhgs3/rhgs-server-rhel7 \
    registry.redhat.io/rhgs3/rhgs-volmanager-rhel7 \
    registry.redhat.io/rhgs3/rhgs-gluster-block-prov-rhel7 \
    registry.redhat.io/rhgs3/rhgs-s3-server-rhel7 \
```

- 对于 IBM POWER8 或 IBM POWER9 服务器中的内部安装，请运行以下命令：

```
$ docker save -o ose3-optional-imags.tar \
    registry.redhat.io/openshift3/metrics-cassandra \
    registry.redhat.io/openshift3/metrics-hawkular-openshift-agent \
    registry.redhat.io/openshift3/metrics-heapster \
    registry.redhat.io/openshift3/metrics-schema-installer \
    registry.redhat.io/openshift3/oauth-proxy \
    registry.redhat.io/openshift3/ose-logging-curator5 \
    registry.redhat.io/openshift3/ose-logging-elasticsearch5 \
    registry.redhat.io/openshift3/ose-logging-eventrouter \
    registry.redhat.io/openshift3/ose-logging-fluentd \
    registry.redhat.io/openshift3/ose-logging-kibana5 \
    registry.redhat.io/openshift3/prometheus \
    registry.redhat.io/openshift3/prometheus-alert-buffer \
    registry.redhat.io/openshift3/prometheus-alertmanager \
    registry.redhat.io/openshift3/prometheus-node-exporter \
```

4. 导出您拉取的 S2I 构建器镜像。例如，如果只同步了 Jenkins 和 Tomcat 镜像：

```
$ docker save -o ose3-builder-images.tar \
    registry.redhat.io/jboss-webserver-3/webserver31-tomcat7-openshift:<tag> \
    registry.redhat.io/jboss-webserver-3/webserver31-tomcat8-openshift:<tag> \
    registry.redhat.io/openshift3/jenkins-2-rhel7:<tag> \
    registry.redhat.io/openshift3/jenkins-agent-maven-35-rhel7:<tag> \
    registry.redhat.io/openshift3/jenkins-agent-nodejs-8-rhel7:<tag> \
    registry.redhat.io/openshift3/jenkins-slave-base-rhel7:<tag> \
    registry.redhat.io/openshift3/jenkins-slave-maven-rhel7:<tag> \
    registry.redhat.io/openshift3/jenkins-slave-nodejs-rhel7:<tag> \
```

5. 将互联网连接主机中压缩的文件复制到您的内部主机中。

6. 加载您复制的镜像：

```
$ docker load -i ose3-images.tar
$ docker load -i ose3-builder-images.tar
$ docker load -i ose3-optional-images.tar
```

7.3. 准备并填充存储库服务器

在安装过程中，以及将来的更新，您需要一个 webserver 来托管软件。RHEL 7 可以提供 Apache webserver。

1. 准备 webserver：

a. 如果您需要在断开连接的环境中安装一个新的 webserver，在您的 LAN 中安装至少110 GB 空间的新 RHEL 7 系统。在 RHEL 安装过程中，选择 Basic Web Server选项。

b. 如果要重新使用下载 OpenShift Container Platform 软件和所需镜像的服务器，请在服务器上安装 Apache:

```
$ sudo yum install httpd
```

2. 将存储库文件放在 Apache 的根目录中。

- 如果您要重新使用服务器：

```
$ mv /path/to/repos /var/www/html/
$ chmod -R +r /var/www/html/repos
$ restorecon -vR /var/www/html
```

- 如果您安装了一个新的服务器，附加外部存储，然后复制文件：

```
$ cp -a /path/to/repos /var/www/html/
$ chmod -R +r /var/www/html/repos
$ restorecon -vR /var/www/html
```

3. 添加防火墙规则：

```
$ sudo firewall-cmd --permanent --add-service=http
$ sudo firewall-cmd --reload
```

4. 启用并启动 Apache 以使更改生效：

```
$ systemctl enable httpd
$ systemctl start httpd
```

7.4. 填充 REGISTRY

在断开连接的环境中，标记并将镜像推送到内部 registry:

重要

以下步骤是将镜像加载到 registry 中的通用过程。您可能需要执行更多或不同的操作来加载镜像。

1. 在将镜像推送到 registry 之前，请重新标记各个镜像。

- 对于 openshift3 存储库中的镜像，将镜像标记为主版本和次版本号。例如，要标记 OpenShift Container Platform 节点镜像，请执行以下操作：

```
$ docker tag registry.redhat.io/openshift3/ose-node:<tag>
registry.example.com/openshift3/ose-node:<tag>
$ docker tag registry.redhat.io/openshift3/ose-node:<tag>
registry.example.com/openshift3/ose-node:{major-tag}
```

- 对于其他镜像，使用准确版本号标记镜像。例如，要标记 etcd 镜像，执行以下操作：

```
$ docker tag registry.redhat.io/rhel7/etcd:3.2.28
registry.example.com/rhel7/etcd:3.2.28
```

2. 将每个镜像推送到 registry。例如，推送 OpenShift Container Platform 节点镜像：

```
$ docker push registry.example.com/openshift3/ose-node:<tag>
$ docker push registry.example.com/openshift3/ose-node:{major-tag}
```

7.5. 准备集群主机

现在已有安装文件，准备您的主机。

1. 为 OpenShift Container Platform 集群创建主机。建议您使用最新版本的 RHEL 7 并执行最小安装。确定主机满足系统要求。

2. 在每个节点主机上创建存储库定义。将以下内容放在 */etc/yum.repos.d/ose.repo* 文件中：

```
[rhel-7-server-rpms]
name=rhel-7-server-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-rpms (1)
enabled=1
gpgcheck=0
[rhel-7-server-extras-rpms]
name=rhel-7-server-extras-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-extras-rpms (2)
enabled=1
gpgcheck=0
[rhel-7-server-ansible-2.9-rpms]
name=rhel-7-server-ansible-2.9-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-ansible-2.9-rpms (3)
enabled=1
gpgcheck=0
[rhel-7-server-ose-3.11-rpms]
name=rhel-7-server-ose-3.11-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-ose-3.11-rpms (4)
enabled=1
gpgcheck=0
```

(1)(2)(3)(4) 将 **<server_IP>** 替换为托管软件程序库的 Apache 服务器的 IP 地址或主机名。

3. 完成主机安装准备。按照 准备主机的步骤，忽略主机注册部分中的步骤。

7.6. 安装 OPENSHIFT CONTAINER PLATFORM

准备软件、镜像和主机后，您可以使用标准安装方法安装 OpenShift Container Platform:

1. 配置清单文件以引用内部 registry:

   - 对于内部 registry:

```
oreg_url=registry.example.com/openshift3/ose-<component>:<version> (1)
openshift_examples_modify_imagestreams=true
```

1 **指定 ose 组件名称和版本号。**

- 对于 Satellite 镜像 registry:

```
oreg_url=satellite.example.com/oreg-prod-openshift3_ose-<component>:<version> 1
osm_etcd_image=satellite.example.com/oreg-prod-rhel7_etcd:3.2.28 2
openshift_examples_modify_imagestreams=true
```

1 **指定 ose 组件名称和版本号。**

2 **如果 etcd 镜像的 URL 前缀在您的 Satellite 服务器上有所不同，您必须在 osm_etcd_image 参数中指定 etcd 镜像的位置和名称。**

2. 运行 Installation Playbook。

第 8 章 安装独立部署的 OPENSHIFT 容器镜像 REGISTRY

OpenShift Container Platform 是一个功能齐全的企业级解决方案，其中包含一个名为 OpenShift Container Registry（OCR）集成的容器镜像 registry。对于开发人员，您可以安装 OCR 作为一个在内部或云环境中的独立容器镜像 registry，而不必把 OpenShift Container Platform 部署为一个完整的 Platform-as-a-Service 环境。

当安装独立 OCR 部署时，仍会安装 master 和节点集群，类似于典型的 OpenShift Container Platform 安装。然后，容器镜像 registry 被部署在集群中运行。此独立部署选项对希望容器镜像 registry 但不需要包括面向开发者的 Web 控制台及应用程序构建和部署工具的完整 OpenShift Container Platform 环境的管理员有用。

OCR 提供以下功能：

- 以用户为主的 registry Web 控制台 Cockpit。
- 默认安全流量，通过 TLS 提供。
- 全局身份提供程序身份验证。
- 项目命名空间 模型,使团队能够通过 基于角色的访问控制(RBAC) 授权进行协作。
- 基于 Kubernetes 的集群来管理服务。
- 名为镜像流（image streams）的镜像提取，用于增强镜像管理。

管理员可以部署一个独立的 OCR 来单独管理支持多个 OpenShift Container Platform 集群的 registry。独立 OCR 还可让管理员分离其 registry，以满足自己的安全或合规要求。

8.1. 最低硬件要求

安装独立 OCR 有以下硬件要求：

- 物理或虚拟系统，或者在一个公共或私有 IaaS 环境中运行的实例。
- 基础操作系统：使用 "Minimal" 安装选项以及 RHEL 7 Extras 频道中的最新软件包的 RHEL 7.5 或 RHEL Atomic Host 7.4.5 或更新版本。
- NetworkManager 1.0 或更高版本。
- 2 个 vCPU。
- 最小 16 GB RAM。
- 最小 15 GB 硬盘空间，用于包含 */var/* 的文件系统。
- Docker 的存储后端最少需要 15 GB 未分配空间 ; 详情请参阅 配置 Docker 存储。

重要

OpenShift Container Platform 支持有 x86_64 或 IBM POWER 架构的服务器。如果您使用 IBM POWER 服务器托管集群节点，则只能使用 IBM POWER 服务器。

注意

为了满足 RHEL Atomic Host 中 */var/* 文件系统大小的要求，需要修改默认配置。有关在安装过程中或安装后配置此功能的说明，请参阅 Red Hat Enterprise Linux Atomic Host 中的管理存储。

8.2. 支持的系统拓扑

以下系统拓扑支持独立 OCR：

All-in-one	包括 master、节点和 registry 组件的单个主机。
多个 master（高可用性）	三个带有所有组件（master、node 和 registry）的主机，在每个主机上都配置了用于原生高可用性的 master。

8.3. 安装 OPENSHIFT CONTAINER REGISTRY

1. 参阅 规划您的安装开。安装 OCR 使用相同的过程，但需要在清单文件中进行一些特定的设置。安装文档包含清单文件可用 Ansible 变量的完整列表。
2. 完成主机准备步骤。
3. 在 */etc/ansible/hosts* 目录中创建 清单文件:

重要

要安装独立 OCR，您必须在 **[OSEv3:vars]** 部分的清单文件中设置 **deployment_subtype=registry**。

为不同的系统拓扑使用以下示例清单文件：

All-in-one 独立 OpenShift Container Registry 清单文件

```
# Create an OSEv3 group that contains the masters and nodes groups
[OSEv3:children]
masters
nodes
etcd

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
# SSH user, this user should allow ssh based auth without requiring a password
ansible_ssh_user=root

openshift_master_default_subdomain=apps.test.example.com

# If ansible_ssh_user is not root, ansible_become must be set to true
#ansible_become=true

openshift_deployment_type=openshift-enterprise
deployment_subtype=registry (1)
```

```
openshift_hosted_infra_selector="" (2)

# uncomment the following to enable htpasswd authentication; defaults to
DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge':
'true', 'kind': 'HTPasswdPasswordIdentityProvider'}]

# host group for masters
[masters]
registry.example.com

# host group for etcd
[etcd]
registry.example.com

# host group for nodes
[nodes]
registry.example.com openshift_node_group_name='node-config-all-in-one'
```

1. **设置 deployment_subtype=registry 以确保安装独立 OCR 而不是完整的 OpenShift Container Platform 环境。**
2. **允许 registry 及其 Web 控制台调度到单一主机上。**

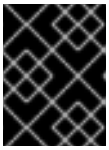
多个 master（高可用性）独立 OpenShift Container Registry 清单文件

```
# Create an OSEv3 group that contains the master, nodes, etcd, and lb groups.
# The lb group lets Ansible configure HAProxy as the load balancing solution.
# Comment lb out if your load balancer is pre-configured.
[OSEv3:children]
masters
nodes
etcd
lb

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=openshift-enterprise
deployment_subtype=registry (1)

openshift_master_default_subdomain=apps.test.example.com

# Uncomment the following to enable htpasswd authentication; defaults to
# DenyAllPasswordIdentityProvider.
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge':
'true', 'kind': 'HTPasswdPasswordIdentityProvider'}]

# Native high availability cluster method with optional load balancer.
# If no lb group is defined installer assumes that a load balancer has
# been preconfigured. For installation the value of
# openshift_master_cluster_hostname must resolve to the load balancer
# or to one or all of the masters defined in the inventory if no load
# balancer is present.
```

```
openshift_master_cluster_method=native
openshift_master_cluster_hostname=openshift-internal.example.com
openshift_master_cluster_public_hostname=openshift-cluster.example.com

# apply updated node-config-compute group defaults
openshift_node_groups=[{'name': 'node-config-compute', 'labels': ['node-
role.kubernetes.io/compute=true'], 'edits': [{'key': 'kubeletArguments.pods-per-core','value':
['20']}, {'key': 'kubeletArguments.max-pods','value': ['250']}, {'key': 'kubeletArguments.image-
gc-high-threshold', 'value':['90']}, {'key': 'kubeletArguments.image-gc-low-threshold', 'value':
['80']}]}]

# enable ntp on masters to ensure proper failover
openshift_clock_enabled=true

# host group for masters
[masters]
master1.example.com
master2.example.com
master3.example.com

# host group for etcd
[etcd]
etcd1.example.com
etcd2.example.com
etcd3.example.com

# Specify load balancer host
[lb]
lb.example.com

# host group for nodes, includes region info
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master-infra'
node1.example.com       openshift_node_group_name='node-config-compute'
node2.example.com       openshift_node_group_name='node-config-compute'
```

1 设置 **deployment_subtype=registry** 以确保安装独立 OCR 而不是完整的 OpenShift Container Platform 环境。

4. 安装独立 OCR。这个过程与完整集群安装过程类似。



重要

运行 Ansible playbook 的主机，对于清单（inventory）文件中的每个主机都至少需要有 75MiB 可用内存。

a. 在部署新集群前，请切换到集群目录，并运行 *prerequisites.yml* playbook:

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook  [-i /path/to/inventory] \ 1
    playbooks/prerequisites.yml
```

1 如果您的清单文件不在 */etc/ansible/hosts* 目录中，使用 **-i** 指定清单文件的路径。

您必须只运行此 playbook 一次。

b. **要启动安装，切换到 playbook 目录并运行 *deploy_cluster.yml* playbook:**

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook  [-i /path/to/inventory] \ 1
    playbooks/deploy_cluster.yml
```

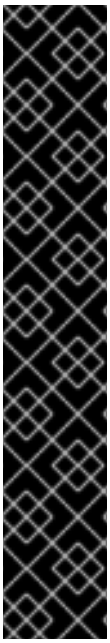
1 **如果您的清单文件不在 */etc/ansible/hosts* 目录中，使用 -i 指定清单文件的路径。**

第 9 章 卸载 OPENSHIFT CONTAINER PLATFORM

您可以通过运行 *uninstall.yml* playbook 来卸载集群中的 OpenShift Container Platform 主机。此 playbook 删除 Ansible 安装的 OpenShift Container Platform 内容，包括：

- Configuration
- 容器
- 默认模板和镜像流
- 镜像
- RPM 软件包

playbook 删除您在运行 playbook 时指定的清单文件中定义的任何主机的内容。



重要

在卸载集群前，请查看以下情况列表，并确定卸载是最佳选项：

- 如果安装过程失败，且想继续这个过程，您可以重试安装。安装 playbook 旨在如果无法安装集群，可以在不需要卸载集群的情况下再次运行它们。
- 如果要从开始重启失败的安装，可以通过运行 *uninstall.yml* playbook 来卸载集群中的 OpenShift Container Platform 主机，如以下部分所述。此 playbook 仅为您安装的最新版本卸载 OpenShift Container Platform 资产。
- 如果必须更改主机名或证书名称，必须在重试安装前重新创建证书，方法是运行 *uninstall.yml* playbook。再次运行安装 playbook 不会重新创建证书。
- 如果要重新使用之前安装的 OpenShift Container Platform 的主机，如使用概念验证安装，或想要安装不同的 OpenShift Container Platform 次要或异步版本，您必须在生产集群中使用主机前重新制作这些主机的镜像。运行 *uninstall.yml* playbook 后，一些主机资产可能会保持在更改的状态。

9.1. 卸载 OPENSHIFT CONTAINER PLATFORM 集群

要在集群中的所有主机中卸载 OpenShift Container Platform，进入 playbook 目录，并使用您最近使用的清单文件运行 playbook:

```
# ansible-playbook [-i /path/to/file] \ 1
    /usr/share/ansible/openshift-ansible/playbooks/adhoc/uninstall.yml
```

1 如果您的清单文件不在 */etc/ansible/hosts* 目录中，使用 **-i** 指定清单文件的路径。

9.2. 卸载节点

在只保留剩余的主机和集群时，使用 *uninstall.yml* playbook 从特定主机中卸载节点组件：

警告

仅在尝试卸载特定节点主机时使用此方法，而不使用特定 master 或 etcd 主机。卸载 master 或 etcd 主机需要在集群中更改更多配置。

1. 按照删除节点中的步骤，从集群中删除节点对象。
2. 创建仅引用那些主机的不同清单文件。例如，要从只有一个节点的环境中删除内容：

```
[OSEv3:children]
nodes (1)

[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=openshift-enterprise

[nodes]
node3.example.com openshift_node_group_name='node-config-infra' (2)
```

(1) 仅包含应用到主机卸载的部分。

(2) 仅包含要卸载的主机。

3. 进入 playbook 目录并运行 *uninstall.yml* playbook：

```
# ansible-playbook -i /path/to/new/file \ (1)
    /usr/share/ansible/openshift-ansible/playbooks/adhoc/uninstall.yml
```

(1) 指定新清单文件的路径。

playbook 完成后，所有 OpenShift Container Platform 内容都会从指定的主机中移除。