Red Hat build of MicroShift 4.16

配置

配置 MicroShift

Last Updated: 2024-06-28

Red Hat build of MicroShift 4.16 配置

配置 MicroShift

法律通告

摘要

本文档提供有关配置 MicroShift 的说明。

目录

第 1 章 配置工具的工作方式

通过一个 YAML 文件，可根据您的偏好、设置和参数自定义 MicroShift 实例。

注意

如果要使用 **kustomize** 清单以外的工具通过 MicroShift API 进行配置更改或部署应用程序，您必须等待 Greenboot 健康检查完成。这样可确保，如果 Greenboot 将 **rpm-ostree** 系统回滚到较早的状态，则您的更改不会丢失。

1.1. 默认设置

如果没有创建 **config.yaml** 文件，则使用默认值。以下示例显示了默认配置设置。

- 要查看默认值，请运行以下命令：

```
$ microshift show-config
```

YAML 格式的默认值示例

```
apiServer:
  advertiseAddress: 10.44.0.0/32 1
  auditLog:
    maxFileAge: 0 2
    maxFileSize: 200 3
    maxFiles: 10 4
    profile: Default 5
  namedCertificates:
    - certPath: ""
      keyPath: ""
      names:
        - ""
  subjectAltNames: [] 6
debugging:
  logLevel: "Normal" 7
dns:
  baseDomain: microshift.example.com 8
etcd:
  memoryLimitMB: 0 9
ingress:
  listenAddress:
    - "" 10
  ports: 11
    http: 80
    https: 443
  routeAdmissionPolicy:
    namespaceOwnership: InterNamespaceAllowed 12
  status: Managed 13
manifests: 14
  kustomizePaths:
    - /usr/lib/microshift/manifests
    - /usr/lib/microshift/manifests.d/*
```

```
    - /etc/microshift/manifests
    - /etc/microshift/manifests.d/*
network:
  clusterNetwork:
    - 10.42.0.0/16 15
  serviceNetwork:
    - 10.43.0.0/16 16
  serviceNodePortRange: 30000-32767 17
node:
  hostnameOverride: "" 18
  nodeIP: "" 19
```

1. 指定 API 服务器公告给集群成员的 IP 地址的字符串。默认值根据服务网络的地址计算。
2. 在自动删除前保留日志文件的时长。**maxFileAge** 参数中的默认值为 **0** 表示日志文件永远不会根据年龄删除。可以配置这个值。
3. 默认情况下，当 **audit.log** 文件达到 **maxFileSize** 限制时，**audit.log** 文件会被轮转，MicroShift 开始写入新的 **audit.log** 文件。可以配置这个值。
4. 保存的日志文件总数。默认情况下，MicroShift 保留 10 个日志文件。创建过量文件时，会删除最旧的文件。可以配置这个值。
5. 仅记录读取和写入请求的日志元数据 ；除了 OAuth 访问令牌请求外，不记录请求正文。如果没有指定此字段，则使用 **Default** 配置集。
6. API 服务器证书的主题备用名称。
7. 日志详细程度。此字段的有效值为 **Normal**,**Debug**,**Trace**, 或 **TraceAll**。
8. 默认情况下，**etcd** 根据需要使用内存来处理系统上的负载。但是，在内存限制的系统中，可能需要在给定时间限制 **etcd** 可以使用的内存量。
9. 集群的基域。所有管理的 DNS 记录都是这个基础的子域。
10. **ingress.listenAddress** 值默认为主机的整个网络。有效可配置的值是一个列表，可以是单个 IP 地址或 NIC 名称，也可以是多个 IP 地址和 NIC 名称。
11. 显示的默认端口。可配置。两个端口条目的有效值为 1-65535 范围内的单个唯一端口。**ports.http** 和 **ports.https** 字段的值不能相同。
12. 描述如何处理跨命名空间的主机名声明。默认情况下，允许路由在命名空间间声明相同主机名的不同路径。有效值为 **Strict** 和 **InterNamespaceAllowed**。指定 **Strict** 可防止不同命名空间中的路由声明相同的主机名。如果在自定义 MicroShift **config.yaml** 中删除了该值，则会自动设置 **InterNamespaceAllowed** 值。
13. 默认路由器状态，可以是 **Managed** 或 **Removed**。
14. 用于扫描 **kustomization** 文件的位置，用于加载清单。设置为仅扫描这些路径的路径列表。设置为空列表以禁用加载清单。列表中的条目可以是 glob 模式，以匹配多个子目录。
15. 从中分配 Pod IP 地址的 IP 地址块。在安装后此字段是不可变的。
16. Kubernetes 服务的虚拟 IP 地址块。服务的 IP 地址池.支持单个条目。在安装后此字段是不可变的。

17 端口范围允许用于 **NodePort** 类型的 Kubernetes 服务。如果没有指定，则使用默认 30000-32767 范围。没有指定 **NodePort** 的服务会自动从这个范围内分配一个。此参数可

18 节点的名称。默认值为 hostname。如果非空，则使用此字符串来识别节点，而不是主机名。

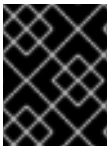
19 节点的 IP 地址。默认值是默认路由的 IP 地址。

1.2. 使用 YAML 配置文件

启动时，MicroShift 会检查系统范围的 **/etc/microshift/** 目录，以查找名为 **config.yaml** 的配置文件。如果目录中不存在配置文件，则使用内置默认值来启动服务。

1.2.1. 自定义设置

要创建自定义配置，您必须在 **/etc/microshift/** 目录中创建 **config.yaml** 文件，然后在启动或重启 MicroShift 前更改要覆盖默认值的设置。



重要

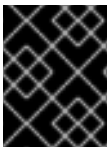
在更改任何配置设置后，重新启动 MicroShift 使其生效。**config.yaml** 文件仅在 MicroShift 启动时读取。

提示

如果同时添加您需要的所有配置，您可以最小化系统重启。

1.2.2. 配置公告地址网络标记

apiserver.advertiseAddress 标志指定要将 API 服务器公告给集群成员的 IP 地址。集群必须可以访问这个地址。您可以在此处设置自定义 IP 地址，但还必须将 IP 地址添加到主机接口。自定义此参数会抢占 MicroShift，将默认 IP 地址添加到 **br-ex** 网络接口。



重要

如果自定义 **advertiseAddress** IP 地址，请通过将 IP 地址添加到主机接口来确保集群可在 MicroShift 启动时被集群访问。

如果未设置，则默认值会在服务网络后设置为下一个即时子网。例如，当服务网络为 **10.43.0.0/16** 时，**广告地址** 被设置为 **10.44.0.0/32**。

1.2.3. 为 NodePort 服务扩展端口范围

serviceNodePortRange 设置扩展可用于 NodePort 服务的端口范围。当需要公开 **30000-32767** 范围下的特定标准端口时，这个选项很有用。例如，如果您的设备需要公开网络上的 **1883/tcp** MQ 遥测传输 (MQTT)端口，因为客户端设备无法使用不同的端口。

重要

NodePort 可以与系统端口重叠，从而导致系统或 MicroShift 出现故障。

配置 NodePort 服务范围时请考虑以下几点：

- 不要在没有明确选择了 **nodePort** 的情况下创建任何 NodePort 服务。如果没有指定显式 **nodePort**，则 **kube-apiserver** 会随机分配端口，且无法预测。
- 不要为在设备 **HostNetwork** 上公开的系统服务端口、MicroShift 端口或其他服务创建任何 NodePort 服务。
- 表一指定在扩展端口范围时要避免的端口：

表 1.1. 避免的端口。

端口	描述
22/tcp	SSH 端口
80/tcp	OpenShift Router HTTP 端点
443/tcp	OpenShift Router HTTPS 端点
1936/tcp	openshift-router 的指标服务，目前不会公开
2379/tcp	etcd 端口
2380/tcp	etcd 端口
6443	kubernetes API
8445/tcp	openshift-route-controller-manager
9537/tcp	cri-o 指标
10250/tcp	kubelet
10248/tcp	kubelet healthz port
10259/tcp	kube 调度程序

1.3. 其他资源

- 检查 Greenboot 状态

第 2 章 使用 KUBECONFIG 的集群访问权限

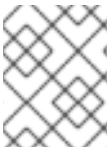
了解 **kubeconfig** 文件如何与 MicroShift 部署一起使用。CLI 工具使用 **kubeconfig** 文件与集群的 API 服务器通信。这些文件提供集群详情、IP 地址以及身份验证所需的其他信息。

2.1. 用于配置集群访问的 KUBECONFIG 文件

MicroShift 中使用的两种 **kubeconfig** 文件是本地访问和远程访问。每次 MicroShift 启动时，都会生成一组用于本地和远程访问 API 服务器的 **kubeconfig** 文件。这些文件使用预先存在的配置信息在 **/var/lib/microshift/resources/kubeadmin/** 目录中生成。

每个访问类型都需要不同的身份验证证书，由不同的证书颁发机构(CA)签名。生成多个 **kubeconfig** 文件来满足这一需求。

您可以将适当的 **kubeconfig** 文件用于每个情况下所需的访问类型，以提供身份验证详情。MicroShift **kubeconfig** 文件的内容由默认的内置值或 **config.yaml** 文件决定。



注意

kubeconfig 文件必须存在，集群才能访问。这些值从内置默认值或 **config.yaml** 应用（如果创建了）。

kubeconfig 文件的内容示例

```
/var/lib/microshift/resources/kubeadmin/
├── kubeconfig (1)
├── alt-name-1 (2)
│   └── kubeconfig
├── 1.2.3.4 (3)
│   └── kubeconfig
└── microshift-rhel9 (4)
    └── kubeconfig
```

1. 本地主机名。主机的主 IP 地址始终是默认值。
2. API 服务器证书的主题备用名称。
3. DNS 名称。
4. MicroShift 主机名。

2.2. 本地访问 KUBECONFIG 文件

本地访问 **kubeconfig** 文件被写入 **/var/lib/microshift/resources/kubeadmin/kubeconfig**。此 **kubeconfig** 文件可使用 **localhost** 访问 API 服务器。当您在本地连接集群时，选择此文件。

用于本地访问的 kubeconfig 的内容示例

```
clusters:
- cluster:
    certificate-authority-data: <base64 CA>
    server: https://localhost:6443
```

-

localhost kubeconfig 文件只能从从同一主机上连接到 API 服务器的客户端使用。文件中的证书不适用于远程连接。

2.2.1. 本地访问 MicroShift 集群

使用以下步骤使用 **kubeconfig** 文件在本地访问 MicroShift 集群。

先决条件

- 已安装 **oc** 二进制文件。

流程

1. 可选：如果您的 RHEL 机器没有 **~/.kube/** 文件夹，请运行以下命令：

```
$ mkdir -p ~/.kube/
```

2. 运行以下命令，将生成的本地访问 **kubeconfig** 文件复制到 **~/.kube/** 目录中：

```
$ sudo cat /var/lib/microshift/resources/kubeadmin/kubeconfig > ~/.kube/config
```

3. 运行以下命令更新 **~/.kube/config** 文件的权限：

```
$ chmod go-r ~/.kube/config
```

验证

- 输入以下命令验证 MicroShift 是否正在运行：

```
$ oc get all -A
```

2.3. 远程访问 KUBECONFIG 文件

当 MicroShift 集群从外部源连接到 API 服务器时，会使用 SAN 字段中所有替代名称的证书进行验证。MicroShift 使用 **hostname** 值为外部访问生成默认的 **kubeconfig**。默认值在默认的 **kubeconfig** 文件的 < **node.hostnameOverride> , <node.nodeIP** > 和 **api.<dns.baseDomain** > 参数值中设置。

/var/lib/microshift/resources/kubeadmin/<hostname>/kubeconfig 文件使用机器的 **hostname** 或 **node.hostnameOverride**（如果设置了这个选项）来访问 API 服务器。**kubeconfig** 文件的 CA 可以在外部访问时验证证书。

用于远程访问的默认 kubeconfig 文件的内容示例

```
clusters:
- cluster:
    certificate-authority-data: <base64 CA>
    server: https://microshift-rhel9:6443
```

2.3.1. 远程访问自定义

可以生成多个远程访问 **kubeconfig** 文件值来访问具有不同 IP 地址或主机名的集群。为 **apiServer.subjectAltNames** 参数中的每个条目生成额外的 **kubeconfig** 文件。您可以在 IP 连接期间从主机复制 **kubeconfig** 文件，然后使用它们从其他工作站访问 API 服务器。

2.4. 为远程访问生成额外的 KUBECONFIG 文件

如果需要更多主机名或 IP 地址超过默认的远程访问文件提供，您可以生成额外的 **kubeconfig** 文件。

重要

您必须重启 MicroShift 才能实现配置更改。

先决条件

- 您已为 MicroShift 创建了一个 **config.yaml**。

流程

1. 可选： 您可以显示 **config.yaml** 的内容。运行以下命令:

```
$ cat /etc/microshift/config.yaml
```

2. 可选： 您可以显示远程访问 **kubeconfig** 文件的内容。运行以下命令:

```
$ cat /var/lib/microshift/resources/kubeadmin/<hostname>/kubeconfig
```

重要

其他远程访问 **kubeconfig** 文件必须包含红帽构建的 MicroShift **config.yaml** 文件中列出的一个服务器名称。其他 **kubeconfig** 文件还必须使用相同的 CA 进行验证。

3. 要为额外的 DNS 名称 SAN 或外部 IP 地址生成额外的 **kubeconfig** 文件，请将您需要的条目添加到 **apiServer.subjectAltNames** 字段。在以下示例中，所用的 DNS 名称为 **alt-name-1**，IP 地址为 **1.2.3.4**。

带有额外身份验证值的 config.yaml 示例

```
dns:
  baseDomain: example.com
node:
  hostnameOverride: "microshift-rhel9" (1)
  nodeIP: 10.0.0.1
apiServer:
  subjectAltNames:
  - alt-name-1 (2)
  - 1.2.3.4 (3)
```

(1) 主机名

(2) DNS 名称

3 IP 地址或范围

4. 运行以下命令，重启 MicroShift 以应用配置更改并自动生成您需要的 **kubeconfig** 文件：

```
$ sudo systemctl restart microshift
```

5. 要检查额外的远程访问 **kubeconfig** 文件的内容，请将 **config.yaml** 中列出的名称或 IP 地址插入到 **cat** 命令中。例如，以下示例命令中使用 **alt-name-1** ：

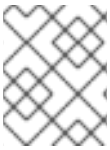
```
$ cat /var/lib/microshift/resources/kubeadmin/alt-name-1/kubeconfig
```

6. 选择 **kubeconfig** 文件，以使用其中包含您要用于连接集群的 SAN 或 IP 地址。在本例中，在 **cluster.server** 字段中包含'alt-name-1' 的 **kubeconfig** 是正确的文件。

其他 kubeconfig 文件的内容示例

```
clusters:
- cluster:
    certificate-authority-data: <base64 CA>
    server: https://alt-name-1:6443 1
```

1 **/var/lib/microshift/resources/kubeadmin/alt-name-1/kubeconfig** 文件值来自 **apiServer.subjectAltNames** 配置值。



注意

所有这些参数都作为通用名称(CN)和主题备用名称(SAN)包含在 API 服务器的外部服务证书中。

2.4.1. 打开防火墙以远程访问 MicroShift 集群

使用以下步骤打开防火墙，以便远程用户可以访问 MicroShift 集群。必须在 workstation 用户可以访问集群前完成此步骤。

对于此过程，**user@microshift** 是 MicroShift 主机上的用户，负责设置该机器，使其可以被单独的工作站上的远程用户访问。

先决条件

- 已安装 **oc** 二进制文件。
- 您的帐户具有集群管理特权。

流程

- 在 MicroShift 主机上以 **user@microshift** 的身份，运行以下命令来打开 Kubernetes API 服务器的防火墙端口 (**6443/tcp**)：

```
[user@microshift]$ sudo firewall-cmd --permanent --zone=public --add-port=6443/tcp &&
sudo firewall-cmd --reload
```

验证

- 以 **user@microshift** 的身份，输入以下命令验证 MicroShift 是否正在运行：

```
[user@microshift]$ oc get all -A
```

2.4.2. 远程访问 MicroShift 集群

使用以下步骤使用 **kubeconfig** 文件从远程位置访问 MicroShift 集群。

user@workstation 登录用于远程访问主机计算机。该流程中的 **<user>** 值是 **user@workstation** 登录到 MicroShift 主机所使用的用户名。

先决条件

- 已安装 **oc** 二进制文件。
- **user@microshift** 已从本地主机打开防火墙。

流程

1. 以 **user@workstation** 的身份，运行以下命令，创建 **~/.kube/** 文件夹：

```
[user@workstation]$ mkdir -p ~/.kube/
```

2. 以 **user@workstation** 的身份，运行以下命令来为您的 MicroShift 主机的主机名设置变量：

```
[user@workstation]$ MICROSHIFT_MACHINE=<name or IP address of MicroShift machine>
```

3. 以 **user@workstation** 的身份，通过运行以下命令复制生成的 **kubeconfig** 文件，其中包含您要从运行 MicroShift 的 RHEL 机器连接到您的本地机器的主机名或 IP 地址：

```
[user@workstation]$ ssh <user>@$MICROSHIFT_MACHINE "sudo cat
/var/lib/microshift/resources/kubeadmin/$MICROSHIFT_MACHINE/kubeconfig" >
~/.kube/config
```

注意

要为此步骤生成 **kubeconfig** 文件，请参阅为远程访问生成额外的 kubeconfig 文件。

4. 以 **user@workstation** 的身份，运行以下命令来更新 **~/.kube/config** 文件的权限：

```
$ chmod go-r ~/.kube/config
```

验证

- 以 **user@workstation** 的身份，输入以下命令验证 MicroShift 是否正在运行：

```
[user@workstation]$ oc get all -A
```

第 3 章 配置自定义证书颁发机构

您可以将自定义证书颁发机构(CA)与 MicroShift 服务一起使用来加密连接。

3.1. 在 MICROSHIFT 中自定义证书颁发机构如何工作

默认 API 服务器证书由内部 MicroShift 集群证书颁发机构(CA)发布。默认情况下，集群外的客户端无法验证 API 服务器证书。此证书可以被一个由客户端信任的自定义 CA 外部发布的自定义服务器证书替代。以下步骤演示了 MicroShift 中的工作流：

1. 将证书和密钥复制到主机操作系统中首选目录中。确保仅可由 root 访问这些文件。
2. 通过在 MicroShift **/etc/microshift/config.yaml** 配置文件中指定证书名称和新的完全限定域名(FQDN)来更新每个自定义 CA 的 MicroShift 配置。
   每个证书配置都可以包含以下值：
   - 证书文件位置是一个所需的值。
   - 包含 API 服务器 DNS 和 IP 地址或 IP 地址范围的单一通用名称。

     提示

     在大多数情况下，MicroShift 为您的自定义 CA 生成一个新的 **kubeconfig**，其中包括您指定的 IP 地址或范围。例外是为 IP 地址指定通配符时。在这种情况下，MicroShift 使用服务器的公共 IP 地址生成 **kubeconfig**。要使用通配符，您必须使用特定详情更新 **kubeconfig** 文件。

   - 包含 API 服务器 DNS 和 IP 地址或通配符证书的多个主题备用名称(SAN)。
   - 您可以为每个证书提供额外的 DNS 名称。
3. MicroShift 服务重启后，您必须将生成的 **kubeconfig** 文件复制到客户端。
4. 在客户端系统上配置其他 CA。例如，您可以在 Red Hat Enterprise Linux (RHEL)信任存储中更新 CA 捆绑包。
5. 证书和密钥从主机上的指定文件位置读取。配置的测试和验证是从客户端进行的。
6. 外部服务器证书不会自动续订。您必须手动轮转外部证书。

注意

如果有任何验证失败，MicroShift 服务将跳过自定义配置，并使用默认证书启动。优先级是继续服务不间断。MicroShift 在服务启动时记录错误。常见错误包括过期的证书、缺少文件或不正确的 IP 地址。

重要

自定义服务器证书必须针对在主机操作系统信任根目录中配置的 CA 数据进行验证。如需更多信息，请参阅 系统范围的信任存储。

3.2. 配置自定义证书颁发机构

要使用自定义证书颁发机构(CA)配置外部生成的证书和域名，请将它们添加到 MicroShift **/etc/microshift/config.yaml** 配置文件中。您还必须配置主机操作系统信任 root。

注意

外部生成的 **kubeconfig** 文件在 **/var/lib/microshift/resources/kubeadmin/<hostname>/kubeconfig** 目录中创建。如果您需要在外部生成的配置之外使用 **localhost**，请在其默认位置保留原始 **kubeconfig** 文件。**localhost kubeconfig** 文件使用自签名证书颁发机构。

先决条件

- 已安装 OpenShift CLI (**oc**)。
- 您可以使用具有集群管理角色的用户访问集群。
- 证书颁发机构已发布自定义证书。
- MicroShift **/etc/microshift/config.yaml** 配置文件存在。

流程

1. 复制您要添加到 MicroShift 主机的信任根目录中的自定义证书。确保证书和私钥只能被 MicroShift 访问。
2. 对于每个您需要的自定义 CA，使用以下示例 **将名为Certificates 的 apiServer** 部分添加到 **/etc/microshift/config.yaml** MicroShift 配置文件中：

```
apiServer:
  namedCertificates:
  - certPath: ~/certs/api_fqdn_1.crt 1
    keyPath:  ~/certs/api_fqdn_1.key 2
  - certPath: ~/certs/api_fqdn_2.crt
    keyPath:  ~/certs/api_fqdn_2.key
    names: 3
    - api_fqdn_1
    - *.apps.external.com
```

1 添加证书的完整路径。

2 添加证书密钥的完整路径。

3 可选。添加显式 DNS 名称列表。允许使用前导通配符。如果没有提供名称，则会从证书中提取隐式名称。

3. 运行以下命令重启 {microshift-service} 以应用证书：

```
$ systemctl microshift restart
```

4. 等待几分钟，让系统重新启动并应用自定义服务器。新的 **kubeconfig** 文件在 **/var/lib/microshift/resources/kubeadmin/** 目录中生成。
5. 将 **kubeconfig** 文件复制到客户端。如果您为 IP 地址指定了通配符，请更新 **kubeconfig** 以删除服务器的公共 IP 地址，并将该 IP 地址替换为您要使用的特定通配符范围。

6. 在客户端中，执行以下步骤：

a. 运行以下命令指定要使用的 **kubeconfig** ：

```
$ export KUBECONFIG=~/custom-kubeconfigs/kubeconfig (1)
```

(1) 使用复制的 **kubeconfig** 文件的位置作为路径。

b. 使用以下命令检查是否应用证书：

```
$ oc --certificate-authority ~/certs/ca.ca get node
```

输出示例

```
oc get node
NAME                                STATUS   ROLES                         AGE   VERSION
dhcp-1-235-195.arm.example.com   Ready    control-plane,master,worker   76m   v1.29.2
```

c. 运行以下命令，将新的 CA 文件添加到 $KUBECONFIG 环境变量中：

```
$ oc config set clusters.microshift.certificate-authority /tmp/certificate-authority-data-new.crt
```

d. 运行以下命令，验证新的 **kubeconfig** 文件是否包含新的 CA：

```
$ oc config view --flatten
```

外部生成的 kubeconfig 文件示例

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /tmp/certificate-authority-data-new.crt (1)
    server: https://api.ci-ln-k0gim2b-76ef8.aws-2.ci.openshift.org:6443
  name: ci-ln-k0gim2b-76ef8
contexts:
- context:
    cluster: ci-ln-k0gim2b-76ef8
    user:
  name:
current-context:
kind: Config
preferences: {}
```

(1) 外部生成的 **kubeconfig** 文件没有 **certificate-authority-data** 部分。它通过前面使用的 **oc config set** 命令添加。

e. 运行以下命令，验证自定义 API 服务器证书颁发机构的 **主题和签发者** ：

```
$ curl --cacert /tmp/caCert.pem https://${fqdn_name}:6443/healthz -v
```

输出示例

```
Server certificate:
  subject: CN=kas-test-cert_server
  start date: Mar 12 11:39:46 2024 GMT
  expire date: Mar 12 11:39:46 2025 GMT
  subjectAltName: host "dhcp-1-235-3.arm.eng.rdu2.redhat.com" matched cert's "dhcp-1-235-3.arm.eng.rdu2.redhat.com"
  issuer: CN=kas-test-cert_ca
  SSL certificate verify ok.
```

重要

将生成的 kubeconfig 文件中的 certificate-authority-data 替换为新的 rootCA，或者将 certificate-authority-data 添加到操作系统的信任 root 中。不要同时使用这两种方法。

f. 在操作系统的信任根中配置额外的 CA。例如，在客户端系统的 RHEL 客户端信任存储中。详情请查看 系统范围的信任存储。

- 建议使用包含 CA 的配置更新证书捆绑包。
- 如果您不想配置证书捆绑包，您可以使用 oc login localhost:8443 --certificate-authority=/path/to/cert.crt 命令，但这不是首选的方法。

3.3. 自定义证书保留名称值

以下证书问题会导致 MicroShift 动态忽略证书并记录错误：

- 证书文件不存在于磁盘上，或者不可读。
- 证书不可解析。
-

证书覆盖 SubjectAlternativeNames (SAN)字段中的内部证书 IPAddress/DNSNames。在配置 SAN 时不要使用保留名称。

表 3.1. 保留名称值

地址	类型	注释
localhost	DNS	
127.0.0.1	IP 地址	
10.42.0.0	IP 地址	集群网络
10.43.0.0/16,10.44.0.0/16	IP 地址	服务网络
169.254.169.2/29	IP 地址	br-ex Network
kubernetes.default.svc	DNS	
openshift.default.svc	DNS	
svc.cluster.local	DNS	

3.4. 自定义证书故障排除

要排除自定义证书的实现，您可以执行以下步骤。

流程

1. 在 MicroShift 中，通过运行以下命令，确保 kube-apiserver 提供证书，并验证证书路径是否已附加到 --tls-sni-cert-key FLAG 中：

```
$ journalctl -u microshift -b0 | grep tls-sni-cert-key
```

输出示例

```
Jan 24 14:53:00 localhost.localdomain microshift[45313]: kube-apiserver I0124
14:53:00.649099   45313 flags.go:64] FLAG: --tls-sni-cert-key="
[/home/eslutsky/dev/certs/server.crt,/home/eslutsky/dev/certs/server.key;/var/lib/micro
shift/certs/kube-apiserver-external-signer/kube-external-
serving/server.crt,/var/lib/microshift/certs/kube-apiserver-external-signer/kube-
external-serving/server.key;/var/lib/microshift/certs/kube-apiserver-localhost-
signer/kube-apiserver-localhost-serving/server.crt,/var/lib/microshift/certs/kube-
```

```
apiserver-localhost-signer/kube-apiserver-localhost-
serving/server.key;/var/lib/microshift/certs/kube-apiserver-service-network-
signer/kube-apiserver-service-network-
serving/server.crt,/var/lib/microshift/certs/kube-apiserver-service-network-
signer/kube-apiserver-service-network-serving/server.key
```

2. 从客户端，运行以下命令来确保 kube-apiserver 提供正确的证书：

```
$ openssl s_client -connect <SNI_ADDRESS>:6443 -showcerts | openssl x509 -text -
noout -in - | grep -C 1 "Alternative\|CN"
```

3.5. 清理和重新创建自定义证书

要停止 MicroShift 服务，请清理自定义证书并重新创建自定义证书，请使用以下步骤。

流程

1. 运行以下命令，停止 MicroShift 服务并清理自定义证书：

```
$ sudo microshift-cleanup-data --cert
```

输出示例

```
Stopping MicroShift services
Removing MicroShift certificates
MicroShift service was stopped
Cleanup succeeded
```

2. 运行以下命令重启 MicroShift 服务以重新创建自定义证书：

```
$ sudo systemctl start microshift
```

3.6. 其他资源

- openshift：添加名为 certificate 的 API 服务器
- RHEL：创建和管理 TLS 密钥和证书
- 系统范围的信任存储
- OpenShift CLI Reference: oc login

第 4 章 检查 GREENBOOT 脚本状态

要使用 kustomize 清单以外的工具通过 MicroShift API 部署应用程序或进行其他更改，您必须等待 Greenboot 健康检查完成。这样可确保，如果 Greenboot 将 rpm-ostree 系统回滚到较早的状态，则您的更改不会丢失。

greenboot-healthcheck 服务会运行一个时间，然后退出。在 Greenboot 退出且系统处于健康状态后，您可以继续进行配置更改和部署。

4.1. 检查 GREENBOOT 健康检查的状态

在对系统进行更改或故障排除期间，检查 Greenboot 健康检查的状态。您可以使用以下任一命令来帮助确保 Greenboot 脚本已运行。

流程

- 要查看健康检查状态的报告，请使用以下命令：

```
$ systemctl show --property=SubState --value greenboot-healthcheck.service
```

  - start 的输出表示 Greenboot 检查仍在运行。
  - 退出 的输出表示检查已通过，Greenboot 已退出。当系统处于健康状态时，greenboot 在 green.d 目录中运行脚本。
  - 失败 的输出意味着检查还没有通过。greenboot 在系统处于此状态时在 red.d 目录中运行脚本，并可能重启系统。

- 要查看显示服务的数字退出代码的报告，其中 0 表示成功，非零值表示发生失败，请使用以下命令：

```
$ systemctl show --property=ExecMainStatus --value greenboot-healthcheck.service
```

- 要查看显示引导状态的消息的报告，如 Boot Status 为 GREEN - Health Check SUCCESS，请使用以下命令：

```
$ cat /run/motd.d/boot-status
```

第 5 章 配置审计日志记录策略

您可以使用配置值控制审计日志文件轮转和保留。

5.1. 关于设置审计日志文件的限制

使用配置值控制审计日志文件的轮转和保留，有助于防止超过边缘设备的有限存储容量。在这样的设备中，日志记录数据积累可能会限制主机系统或集群工作负载，从而导致设备停止工作。设置审计日志策略有助于确保持续使用关键处理空间。

您设置为限制审计日志的值可让您强制审计日志备份的大小、数量和年龄限制。字段值独立于另一个字段处理，且没有优先级。

您可以组合设置字段来为保留日志定义最大存储限制。例如：

- 设置 **maxFileSize** 和 **maxFiles**，以创建日志存储上限。
- 设置 **maxFileAge** 值，以自动删除文件名中时间戳旧的文件，而不考虑 **maxFiles** 值。

5.1.1. 默认审计日志值

MicroShift 包括以下默认审计日志轮转值：

表 5.1. MicroShift 默认审计日志值

审计日志参数	默认设置	定义
maxFileAge:	**0**	在自动删除前保留日志文件的时长。默认值表示，日志文件永远不会根据年龄删除。可以配置这个值。
maxFiles:	**10**	保留的日志文件总数。默认情况下，MicroShift 保留 10 个日志文件。创建过量文件时，会删除最旧的文件。可以配置这个值。
maxFileSize:	**200**	默认情况下，当 **audit.log** 文件达到 **maxFileSize** 限制时，**audit.log** 文件会被轮转，MicroShift 开始写入新的 **audit.log** 文件。这个值以 MB 为单位，可以进行配置。

审计日志参数	默认设置	定义
配置集 ：	**default**	**Default** 配置集只为读取和写入请求记录元数据；除了 OAuth 访问令牌请求外，请求正文不会被记录。如果没有指定此字段，则使用 **Default** 配置集。

如果文件有 10 个或更少，则审计日志保留的最大默认存储使用量为 2000Mb。

如果没有为某个字段指定值，则使用默认值。如果您删除了之前设置的字段值，则默认值会在下一个 MicroShift 服务重启后恢复。

5.2. 关于审计日志策略配置集

审计日志配置集定义了如何记录 **OpenShift API 服务器和 Kubernetes API 服务器的请求。**

MicroShift 支持以下预定义的审计策略配置集：

profile	描述
default	仅记录读取和写入请求的日志元数据 ；除了 OAuth 访问令牌请求外，不记录请求正文。这是默认策略。
WriteRequestBodies	除了记录所有请求的元数据外，还会记录对 API 服务器的写入请求（**create**, **update**, **patch**, **delete**, **deletecollection**）。这个配置集的资源开销比 **Default** 配置集大。[1]
AllRequestBodies	除了记录所有请求的元数据外，对 API 服务器的每个读写请求（**get**、**list**、**create**、**update**、**patch**）都进行日志记录。这个配置集的资源开销最大。[1]

profile	描述
None	没有记录请求，包括 OAuth 访问令牌请求和 OAuth 授权令牌请求。 **警告** 除非完全了解在对问题进行故障排除时无法记录数据的风险，否则不要使用 **None** 配置集禁用审计日志记录。如果禁用审计日志记录并出现支持情况，您可能需要启用审计日志记录并重现问题以正确排除故障。

1. **敏感资源（如 Secret、 Route 和 OAuthClient 对象）仅在元数据级别上记录。**

默认情况下，MicroShift 使用 Default 审计日志配置集。您可以使用另一个审计策略配置集来记录请求正文，但请注意 CPU、内存和 I/O 等资源使用量增加。

5.3. 配置审计日志值

您可以使用 MicroShift 服务配置文件配置审计日志设置。

流程

1. **在 /etc/microshift/ 目录中复制提供的 config.yaml.default 文件，将它重命名为 config.yaml。将您创建的新 MicroShift config.yaml 保留在 /etc/microshift/ 目录中。每当 MicroShift 服务启动时都会读取新的 config.yaml。创建后，config.yaml 文件优先于内置设置。**

2. **将 YAML 的 auditLog 部分中的默认值替换为您需要的有效值。**

默认 auditLog 配置示例

```
apiServer:
# ....
  auditLog:
    maxFileAge: 7 (1)
    maxFileSize: 200 (2)
    maxFiles: 1 (3)
    profile: Default (4)
# ....
```

(1) 指定日志文件保留的最长时间（以天为单位）。超过这个限制的文件会被删除。在本例中，日志文件超过 7 天后，它将被删除。无论 live 日志是否已达到 maxFileSize 字段中指定的最大文件大小，文件都会被删除。文件年龄由轮转日志文件名称写入的时间戳决定，例如 audit-2024-05-16T17-03-59.994.log。当值为 0 时，会禁用限制。

(2) 以 MB 为单位的最大审计日志文件大小。在本例中，当实时日志达到 200 MB 限制时，该文件就会立即轮转。当值设为 0 时，会禁用限制。

(3) 轮转审计日志文件的最大数量。达到限制后，日志文件将从最旧的到最新的顺序删除。在这个示例中，除了当前活跃的日志外，值 1 只会保留 1 个大小 maxFileSize 的文件。当值设为 0 时，会禁用限制。

(4) 仅记录读取和写入请求的日志元数据 ；除了 OAuth 访问令牌请求外，不记录请求正文。如果没有指定此字段，则使用 Default 配置集。

3. 可选： 要为日志指定新目录，您可以停止 MicroShift，然后将 /var/log/kube-apiserver 目录移到所需位置：

   a. 运行以下命令来停止 MicroShift：

```
$ sudo systemctl stop microshift
```

b. 运行以下命令，将 /var/log/kube-apiserver 目录移到所需的位置：

```
$ sudo mv /var/log/kube-apiserver <~/kube-apiserver> 1
```

1 将 < ~/*kube-apiserver* > 替换为您要使用的目录的路径。

c. 如果您为日志指定了新目录，请运行以下命令在 /var/log/kube-apiserver 上创建一个指向自定义目录的符号链接：

```
$ sudo ln -s <~/kube-apiserver> /var/log/kube-apiserver 1
```

1 将 < ~/*kube-apiserver* > 替换为您要使用的目录的路径。这会在 sos 报告中启用日志集合。

4. 如果要在正在运行的实例中配置审计日志策略，请输入以下命令重启 MicroShift：

```
$ sudo systemctl restart microsohift
```

5.4. 审计日志配置故障排除

使用以下步骤对自定义审计日志设置和文件位置进行故障排除。

流程

- 运行以下命令检查配置的当前值：

```
$ sudo microshift show-config --mode effective
```

输出示例

```
auditLog:
  maxFileSize: 200
  maxFiles: 1
```

```
    maxFileAge: 7
    profile: AllRequestBodies
```

- 运行以下命令检查 audit.log 文件权限：

```
$ sudo ls -ltrh /var/log/kube-apiserver/audit.log
```

输出示例

```
-rw-------. 1 root root 46M Mar 12 09:52 /var/log/kube-apiserver/audit.log
```

- 运行以下命令，列出当前日志目录的内容：

```
$ sudo ls -ltrh /var/log/kube-apiserver/
```

输出示例

```
total 6.0M
-rw-------. 1 root root 2.0M Mar 12 10:56 audit-2024-03-12T14-56-16.267.log
-rw-------. 1 root root 2.0M Mar 12 10:56 audit-2024-03-12T14-56-49.444.log
-rw-------. 1 root root 962K Mar 12 10:57 audit.log
```