# NestOS Documentation


NestOS 是一款基于openEuler开发的自动更新的最小化操作系统。它将配置工具 ignition 与自动更新模型 rpm-ostree，OCI 支持，SELinux 强化等技术集成在一起。采用双系统分区，容器技术和集群架构的设计思路，可以适应各种不同的基础设施环境，使系统具备十分便捷的集群组件能力，而且可以独立运行，并针对运行Kubernetes进行了优化。

同时 NestOS 可以安全的进行系统的升级与回滚，可用于无需任何维护即可运行的的容器系统，与 OKD 紧密集成。NestOS 的目标即为提供最佳的容器主机，用于大规模下安全的运行容器化工作负载。

# NestOS 入门

## 介绍

### Streams

NestOS 有三种更新stream可供选择：stable，testing，next，在一般情况下，通常会选择 stable，但还是建议在一些场景下使用 testing 和 next 并给我们提供反馈。

### 概要

与经典操作系统相比， NestOS 没有单独的安装盘，每个实例都从通用的磁盘镜像开始，在首次启动时需要通过 Ignitiion 来进行自定义安装。

Ignition 是一个供应实用程序，它通过读取配置文件（JSON 格式）部署NestOS。可配置的组件包括存储和文件系统、systemd 单元和用户。Ignition 仅在系统第一次引导期间运行一次（在 initramfs 中）。因为 Ignition 在启动过程的早期运行，所以它可以在用户空间开始启动之前重新分区磁盘、格式化文件系统、创建用户和写入文件。因此，systemd 服务在 systemd 启动时已经写入磁盘，从而加快了启动速度。

每个平台都有特定的机制来识别系统首次安装。对于云部署，Ignition 通过用户数据机制收集配置；对于裸机，Ignition可以从磁盘或远程来源获取其配置。

## 快速开始

## 在 VMware 上部署 NestOS

本指南展示了如何在VMware虚拟机管理程序上配置最新的 NestOS。

### 开始之前

在开始部署 NestOS 之前，需要做如下准备工作：

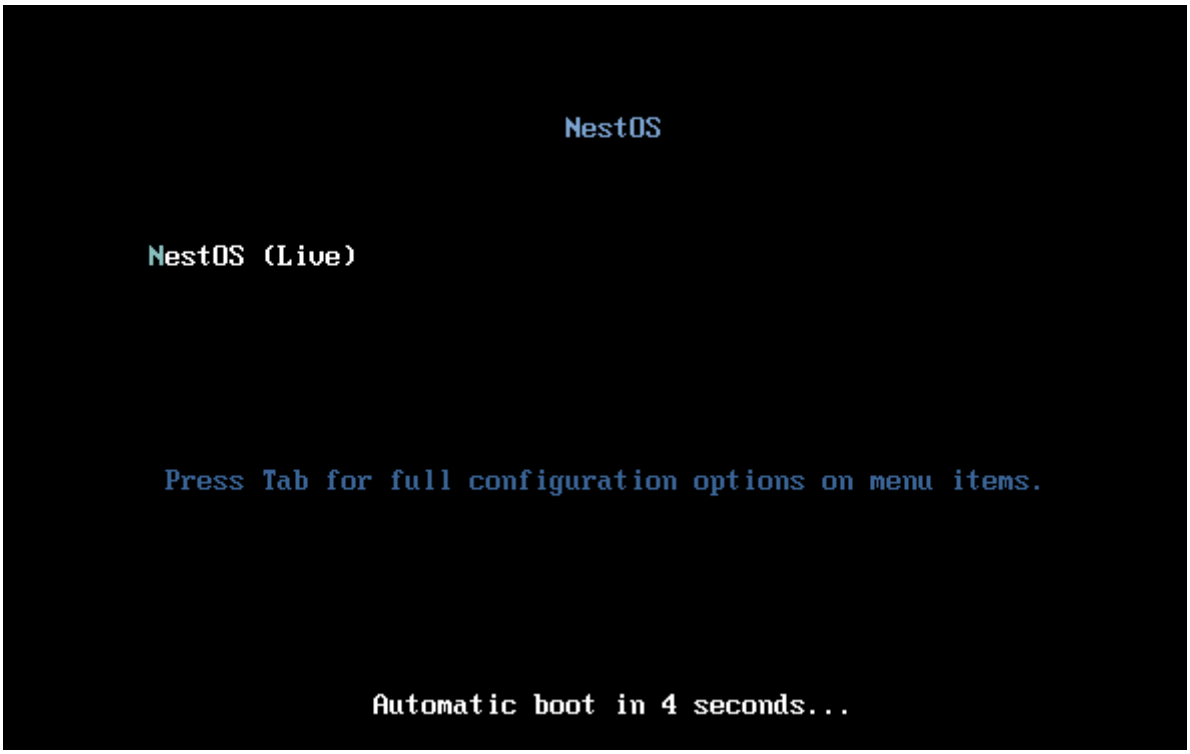
- 下载 NestOS ISO
- 准备 config.bu 文件
- 配置 butane 工具（Linux环境/win10环境）
- 安装有VMware的宿主机

### 初步安装与启动

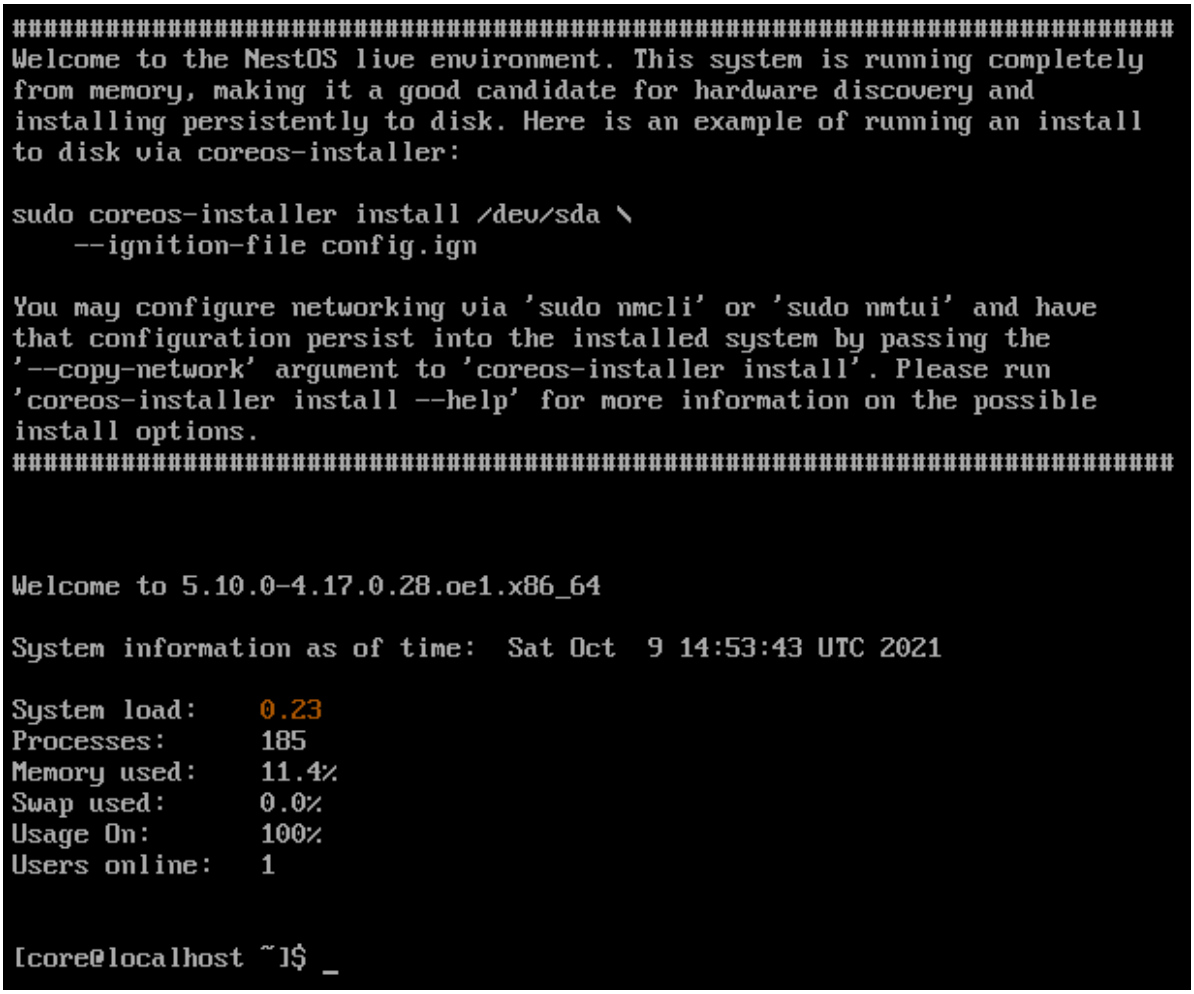
#### 在 VMware 中添加 iso


#### 启动 NestOS

NestOS 初次启动如下图所示




初次启动 NestOS ， ignition 尚未安装，可根据系统提示使用 nestos-installer 组件进行 ignition的安装。




**配置 ignition 文件**

**获取 Butane**

可以通过 Butane 将 bu 文件转化为 igniton 文件。ignition 配置文件被设计为可读但难以编写，是为了阻止用户尝试手动编写配置。

Butane 提供了多种环境的支持，可以在 linux/windows 宿主机中或容器环境中进行配置。

```
docker pull quay.io/coreos/butane:release
```

**生成登录密码**

在宿主机执行如下命令，并输入你的密码。

```
# openssl passwd -1 -salt yoursalt
Password:
$1$yoursalt$1QskegeyhtMG2tdhOldQN0
```

**生成ssh-key**

在宿主机执行如下命令，获取公钥和私钥以供后续 ssh 登录。

```
# ssh-keygen -N '' -f ./id_rsa
Generating public/private rsa key pair.
Your identification has been saved in ./id_rsa
Your public key has been saved in ./id_rsa.pub
The key fingerprint is:
SHA256:4fFpDDyGHOYEd2fPaprKvvqst3T1xBQuk3mbdon+0Xs root@host-12-0-0-141
```

```
The key's randomart image is:
+---[RSA 3072]----+
|    ..= . o .    |
|     * = o * .   |
|      + B = *    |
|       o B O + . |
|        S O B o  |
|         * = . . |
|      . +   o . .|
|     +.o     . .E|
|    o*Oo      ...|
+----[SHA256]-----+
```

可以在当前目录查看id_rsa.pub公钥

```
# cat id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDjf+I9QQZ+3vNWomqxpHkZq7ONHcEYBzs4C9RZahmLYVPBf/3y
HF5wTtfl5CBviERUnLGFn8c4Ua9MNWcJL6zEO1xXtDZ2db7vaPwP3Qbo1lKJg1BVw6u+5bMKCJxEnN9+
aOiX3A2XpkUVxhCoGlei3j78oLRU3ucCLn6m7wVE+P53tNQ5364xWqbAsDXdze4xnNZjlzH9JvjJ5IJY
WjwrD7UUkfI8qDj5ub9Gz+nSenaaSboWADsKe4JTLoU2Gz5fPLCj+uuNFpZAUc/GCe47He5UO6IbHjDI
bxqhzYZQTXdwIgKIM1PL19IkAAYO7gU53b4gDSDj7SZYB+jjtgG8VoFF4m7nCJgRDeUKGTNT5fsLPKAZ
tmBvy9Mg5qkK/LisEzjUwPPh1NEb8bgN251wPXmPMjQ1aMzD8t9blq4OKEyod2Eg05nW2q5/9OICNQBa
r9AkQrQ/3j8WsejvqseWIi1kq68pqvtcBJkCMiIfzIoUgCgcolw3fZprDhgfau8= root@host-12-0-
0-141
```

#### 编写bu文件

进行最简单的初始配置，如需更多详细的配置，参考后面的 ignition 详解。

如下为最简单的 config.bu 文件

```
variant: fcos
version: 1.1.0
passwd:
  users:
    - name: core
      password_hash: "$1$yoursalt$1QskegeyhtMG2tdhOldQN0"
      ssh_authorized_keys:
        - "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDjf+I9QQZ+3vNWomqxpHkZq7ONHcEYBzs4C9RZahmLYVPBf/3y
HF5wTtfl5CBviERUnLGFn8c4Ua9MNWcJL6zEO1xXtDZ2db7vaPwP3Qbo1lKJg1BVw6u+5bMKCJxEnN9+
aOiX3A2XpkUVxhCoGlei3j78oLRU3ucCLn6m7wVE+P53tNQ5364xWqbAsDXdze4xnNZjlzH9JvjJ5IJY
WjwrD7UUkfI8qDj5ub9Gz+nSenaaSboWADsKe4JTLoU2Gz5fPLCj+uuNFpZAUc/GCe47He5UO6IbHjDI
bxqhzYZQTXdwIgKIM1PL19IkAAYO7gU53b4gDSDj7SZYB+jjtgG8VoFF4m7nCJgRDeUKGTNT5fsLPKAZ
tmBvy9Mg5qkK/LisEzjUwPPh1NEb8bgN251wPXmPMjQ1aMzD8t9blq4OKEyod2Eg05nW2q5/9OICNQBa
r9AkQrQ/3j8WsejvqseWIi1kq68pqvtcBJkCMiIfzIoUgCgcolw3fZprDhgfau8= root@host-12-0-
0-141"
```

#### 生成ignition文件

将 config.bu 通过 Butane 工具转换为 config.ign 文件，如下为在容器环境下进行转换。

```
# docker run --interactive --rm quay.io/coreos/butane:release \
--pretty --strict < your_config.bu > transpiled_config.ign
```

也可在其他环境下进行转换，Butane提供了多种转换的方式，可在如下地址查看。

https://github.com/coreos/butane

**安装 NestOS**

将宿主机生成的的config.ign文件通过scp拷贝到前面初步启动的 NestOS 中，该OS目前运行在内存中，并没有安装到硬盘。

```
sudo -i
scp root@12.1.10.88:/root/config.ign /root     #例子从12.1.10.88拷贝，请自行更换你的生成ign文件的机器的ip，我们在官网提供了一份.ign可自行下载
```

根据系统所给提示，执行如下指令完成安装。

```
nestos-installer install /dev/sda --ignition-file config.ign
```

安装完成后重启 NestOS 。

```
systemctl reboot
```

# Rpm-ostree 使用

## rpm-ostree安装软件包

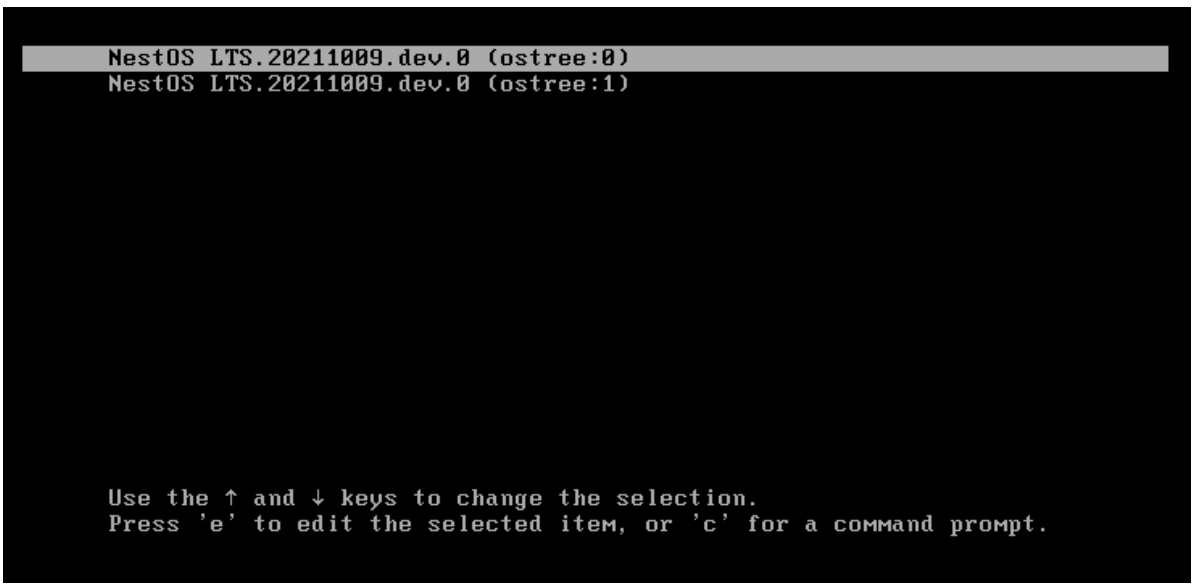
安装wget

```
rpm-ostree install wget
```

```
[root@localhost ~]# rpm-ostree install wget
Checking out tree fe2408e... done
Enabled rpm-md repositories: updates OS everything EPOL de
Updating metadata for 'updates'... done
rpm-md repo 'updates'; generated: 2021-04-08T08:32:39Z
Updating metadata for 'OS'... done
rpm-md repo 'OS'; generated: 2020-03-25T03:42:26Z
Updating metadata for 'everything'... done
rpm-md repo 'everything'; generated: 2020-03-25T06:14:16Z
Updating metadata for 'EPOL'... done
rpm-md repo 'EPOL'; generated: 2020-03-26T17:59:31Z
Updating metadata for 'debuginfo'... done
rpm-md repo 'debuginfo'; generated: 2020-03-25T04:28:18Z
Updating metadata for 'source'... done
rpm-md repo 'source'; generated: 2020-05-11T03:29:48Z
Updating metadata for 'update'... done
rpm-md repo 'update'; generated: 2021-04-08T08:32:39Z
Importing rpm-md... done
Resolving dependencies... done
Will download: 1 package (659.9?kB)
Downloading from 'updates'... done
Importing packages... done
Checking out packages... done
Running pre scripts... done
Running post scripts... done
Running posttrans scripts... done
Writing rpmdb... done
Writing OSTree commit... done
Staging deployment... done
Added:
  wget-1.20.3-2.oe1.x86_64
Run "systemctl reboot" to start a reboot
```

重启系统，可在启动时通过键盘上下按键选择rpm包安装完成后或安装前的系统状态，其中【ostree:0】为安装之后的版本。

```
systemctl reboot
```



查看wget是否安装成功

```
rpm -qa | grep wget
```

```
[root@localhost ~]# rpm -qa | grep wget
wget-1.20.3-2.oe1.x86_64
```

## rpm-ostree 手动更新升级 NestOS

在NestOS中执行命令可查看当前rpm-ostree状态，可看到当前版本为LTS.20210927.dev.0

```
rpm-ostree status
```

```
[root@localhost ~]# rpm-ostree status
State: idle
Deployments:
* ostree://openEuler:openEuler/x86_64/nestos/stable
                   Version: LTS.20210927.dev.0 (2021-09-27T03:23:10Z)
                    Commit: fe2408e341487c9d6971348091bf48a03f125e2dbd4cb5b4e634ecfc1284711c
```

执行检查命令查看是否有升级可用，发现存在LTS.20210928.dev.0版本

```
rpm-ostree upgrade --check
```

```
[root@localhost ~]# rpm-ostree upgrade --check
2 metadata, 0 content objects fetched; 20 KiB transferred in 0 seconds; 0 bytes content written
AvailableUpdate:
        Version: LTS.20210928.dev.0 (2021-09-28T01:33:53Z)
         Commit: 55eed9bfc5ec32cf8333fddc775a12ad38f82516c5b5c5477132614c24a2f6a5
           Diff: 1 added
```

预览版本的差异

```
rpm-ostree upgrade --preview
```

```
[root@localhost ~]# rpm-ostree upgrade --preview
1 metadata, 0 content objects fetched; 313 B transferred in 0 seconds; 0 bytes content written
AvailableUpdate:
        Version: LTS.20210928.dev.0 (2021-09-28T01:33:53Z)
         Commit: 55eed9bfc5ec32cf8333fddc775a12ad38f82516c5b5c5477132614c24a2f6a5
          Added: wget-1.20.3-1.oe1.x86_64
```

可以看到，在0928的最新版本中，我们将wget包做了引入。

下载最新的ostree和RPM数据，不需要进行部署

```
rpm-ostree upgrade --download-only
```

```
[root@localhost ~]# rpm-ostree upgrade --download-only
⠒ Writing objects: 1
Writing objects: 1... done
Update downloaded.
```

重启NestOS，重启后可看到系统的新旧版本两个状态，选择最新版本的分支进入

```
rpm-ostree upgrade --reboot
```

# 比较NestOS版本差别

检查状态，确认此时ostree有两个版本，分别为LTS.20210927.dev.0和LTS.20210928.dev.0

```
rpm-ostree status
```

```
[root@localhost ~]# rpm-ostree status
State: idle
Deployments:
* ostree://openEuler:openEuler/x86_64/nestos/stable
                   Version: LTS.20210928.dev.0 (2021-09-28T01:33:53Z)
                    Commit: 55eed9bfc5ec32cf8333fddc775a12ad38f82516c5b5c5477132614c24a2f6a5

  ostree://openEuler:openEuler/x86_64/nestos/stable
                   Version: LTS.20210927.dev.0 (2021-09-27T03:23:10Z)
                    Commit: fe2408e341487c9d6971348091bf48a03f125e2dbd4cb5b4e634ecfc1284711c
```

根据commit号比较2个ostree的差别

```
rpm-ostree db diff 55eed9bfc5ec fe2408e34148
```

```
[root@localhost ~]# rpm-ostree db diff 55eed9bfc5ec fe2408e34148
ostree diff commit from: 55eed9bfc5ec (55eed9bfc5ec32cf8333fddc775a12ad38f82516c5b5c5477132614
ostree diff commit to:   fe2408e34148 (fe2408e341487c9d6971348091bf48a03f125e2dbd4cb5b4e634ecf
Removed:
  wget-1.20.3-1.oe1.x86_64
```

# 系统回滚

当一个系统更新完成，之前的NestOS部署仍然在磁盘上，如果更新导致了系统出现问题，可以使用之前的部署回滚系统。

## 临时回滚

要临时回滚到之前的OS部署，在系统启动过程中按住shift键，当引导加载菜单出现时，在菜单中选择相关的分支。

## 永久回滚

要永久回滚到之前的操作系统部署，登录到目标节点，运行rpm-ostree rollback，此操作将使用之前的系统部署作为默认部署，并重新启动到其中。

执行命令，回滚到前面更新前的系统。

```
rpm-ostree rollback
```

```
[root@localhost ~]# rpm-ostree rollback
Moving 'fe2408e341487c9d6971348091bf48a03f125e2dbd4cb5b4e634ecfc1284711c.0' to be first deployment
Bootloader updated; bootconfig swap: yes; bootversion: boot.0.1, deployment count change: 0
Removed:
  wget-1.20.3-1.oe1.x86_64
Run "systemctl reboot" to start a reboot
```

重启后生效。

# 切换版本

在上一步将NestOS回滚到了LTS.20210927.dev.0版本，可以通过命令切换当前 NestOS 使用的rpm-ostree版本，将LTS.20210927.dev.0切换为LTS.20210928.dev.0版本。

```
rpm-ostree  deploy -r LTS.20210928.dev.0
```

```
[root@localhost ~]# rpm-ostree  deploy -r LTS.20210928.dev.0
Resolving version 'LTS.20210928.dev.0'
1 metadata, 0 content objects fetched; 313 B transferred in 0 seconds; 0 bytes content written
1 metadata, 0 content objects fetched; 313 B transferred in 0 seconds; 0 bytes content written
Staging deployment... done
Freed: 608.8?kB (pkgcache branches: 1)
```

重启后确认目前NestOS已经使用的是LTS.20210928.dev.0版本的ostree了。

```
[root@localhost ~]# rpm-ostree status
State: idle
Deployments:
* ostree://openEuler:openEuler/x86_64/nestos/stable
                   Version: LTS.20210928.dev.0 (2021-09-28T01:33:53Z)
                    Commit: 55eed9bfc5ec32cf8333fddc775a12ad38f82516c5b5c5477132614c24a2f6a5

  ostree://openEuler:openEuler/x86_64/nestos/stable
                   Version: LTS.20210927.dev.0 (2021-09-27T03:23:10Z)
                    Commit: fe2408e341487c9d6971348091bf48a03f125e2dbd4cb5b4e634ecfc1284711c
```

# Zincati 自动更新

zincati负责NestOS的自动更新，zincati通过cincinnati提供的后端来检查当前是否有可更新版本，若检测到有可能新版本，会通过rpm-ostree进行下载。cincinnati根据配置的json文件的版本号、commit号等信息生成一个有向无环图（节点为各个版本，节点之间存在边代表两个版本之间可进行更新），可通过指令查看cincinnati返回的图。

```
curl --verbose "http://nestos.org.cn:8080/v1/graph?
basearch=x86_64&stream=stable"
```

目前系统默认关闭zincati自动更新服务，可通过修改配置文件设置为开机自动启动自动更新服务。

```
vi /etc/zincati/config.d/95-disable-on-dev.toml
```

将updates.enabled设置为true

同时增加配置文件，修改cincincati后端地址

```
vi /etc/zincati/config.d/update-cincinnati.toml
```

添加如下内容

```
[cincinnati]
base_url="http://nestos.org.cn:8080"
```

重新启动zincati服务

```
systemctl restart zincati.service
```

当有新版本时，zincati会自动检测到可更新版本，此时查看rpm-ostree状态，可以看到状态是“busy”，说明系统正在升级中。

```
[root@localhost config.d]# rpm-ostree status
State: busy
Transaction: deploy --lock-finalization revision=27a0ac355f0e21fd2c75a3bab9cdddabf5ce9995117c4705da60b9e4de2b162f --disallow-downgrade
  Initiator: caller :1.44
Deployments:
● ostree://openEuler:openEuler/x86_64/nestos/stable
                   Version: LTS.20211012.dev.0 (2021-10-12T07:04:20Z)
                    Commit: bc5e9f56050897f180fbb8c6515d7a1d039a734eab8758ba9d6006c2ac2d433a
```

一段时间后NestOS将自动重启，此时再次登录NestOS，可以再次确认rpm-ostree的状态，其中状态转为"idle"，而且当前版本已经是“20211013”，这说明rpm-ostree版本已经升级了。

```
[root@localhost ~]# rpm-ostree status
State: idle
Deployments:
● ostree://openEuler:openEuler/x86_64/nestos/stable
                   Version: LTS.20211013.dev.0 (2021-10-13T01:05:07Z)
                    Commit: 27a0ac355f0e21fd2c75a3bab9cdddabf5ce9995117c4705da60b9e4de2b162f

  ostree://openEuler:openEuler/x86_64/nestos/stable
                   Version: LTS.20211012.dev.0 (2021-10-12T07:04:20Z)
                    Commit: bc5e9f56050897f180fbb8c6515d7a1d039a734eab8758ba9d6006c2ac2d433a
```

查看zincati服务的日志，确认升级的过程和重启系统的日志。另外日志显示的"auto-updates logic enabled"也说明更新是自动的

```
10月 13 03:27:13 localhost zincati[1418]: [INFO ] starting update agent (zincati 0.0.18)
10月 13 03:27:13 localhost zincati[1418]: [INFO ] Cincinnati service: http://10.1.110.88:8080
10月 13 03:27:13 localhost zincati[1418]: [INFO ] agent running on node 'f2da8badca0f4fcbb00f0efd7e1b1a84', in update group 'default'
10月 13 03:27:14 localhost zincati[1418]: [INFO ] initialization complete, auto-updates logic enabled
10月 13 03:27:14 localhost zincati[1418]: [INFO ] update strategy: immediate
10月 13 03:27:14 localhost systemd[1]: Started Zincati Update Agent.
   Subject: zincati.service 单元已结束启动
   Defined-By: systemd
   Support: https://lists.freedesktop.org/mailman/listinfo/systemd-devel

   zincati.service 单元已结束启动。

   启动结果为“done”。
10月 13 03:27:14 localhost zincati[1418]: [INFO ] current release detected as not a dead-end
10月 13 03:27:14 localhost zincati[1418]: [INFO ] target release 'LTS.20211013.dev.0' selected, proceeding to stage it
10月 13 03:27:19 localhost zincati[1418]: [INFO ] staged deployment 'LTS.20211013.dev.0' available, proceeding to finalize it and reboot
10月 13 03:27:20 localhost zincati[1418]: [INFO ] update applied, waiting for reboot: LTS.20211013.dev.0
```

# K8S+iSulad 搭建

**以下步骤在master节点和node节点均需执行**，本教程以master为例

## 开始之前

需准备如下内容

1.NestOS-LTS-live.x86_64.iso

2.一台主机用作master，一台主机用作node

## 组件下载

编辑源文件，添加k8s的阿里云源

```
vi /etc/yum.repos.d/openEuler.repo
```

添加如下内容

```
[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64/
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
```

下载k8s组件以及同步系统时间所用组件

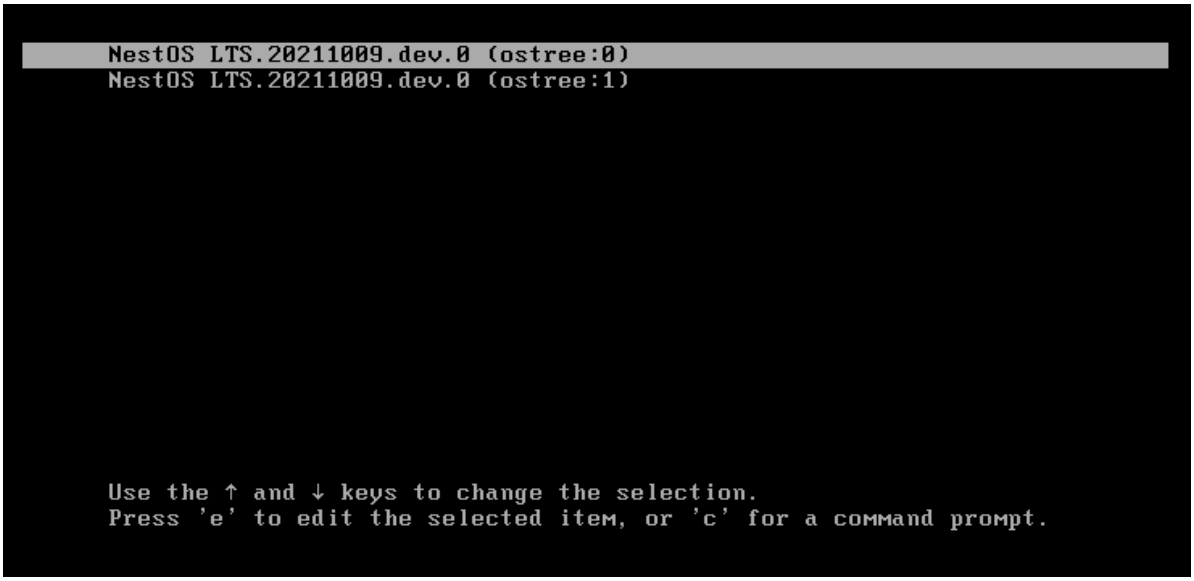
```
rpm-ostree install kubelet kubeadm kubectl ntp ntpdate wget
```

```
Writing rpmdb... done
Writing OSTree commit... done
Staging deployment... done
Added:
  conntrack-tools-1.4.6-1.oe1.x86_64
  cri-tools-1.13.0-0.x86_64
  kubeadm-1.22.2-0.x86_64
  kubectl-1.22.2-0.x86_64
  kubelet-1.22.2-0.x86_64
  libnetfilter_cthelper-1.0.0-15.oe1.x86_64
  libnetfilter_cttimeout-1.0.0-13.oe1.x86_64
  libnetfilter_queue-1.0.2-13.oe1.x86_64
Run "systemctl reboot" to start a reboot
```

重启生效

```
systemctl reboot
```

选择正确的分支




查看软件包是否已安装

```
rpm -qa | grep kube
```

```
[root@localhost ~]# rpm -qa | grep kube
kubelet-1.22.2-0.x86_64
kubectl-1.22.2-0.x86_64
kubeadm-1.22.2-0.x86_64
```

# 配置环境

## 修改主机名，以master为例

```
hostnamectl set-hostname k8s-master
sudo -i
```

编辑/etc/hosts

```
vi /etc/hosts
```

```
添加如下内容，ip为主机ip
192.168.237.133  k8s-master
192.168.237.135  k8s-node01
```

### 同步系统时间

```
ntpdate time.windows.com
systemctl enable ntpd
```

### 关闭swap分区，防火墙，selinux

NestOS默认无swap分区，默认关闭防火墙

关闭selinux如下

```
vi /etc/sysconfig/selinux
修改为SELINUX=disabled
```

### 网络配置，开启相应的转发机制

创建配置文件

```
vi /etc/sysctl.d/k8s.conf
```

添加如下内容

```
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-ip6tables=1
net.ipv4.ip_forward=1
```

使配置生效

```
modprobe br_netfilter
sysctl -p /etc/sysctl.d/k8s.conf
```

## 配置iSula

查看k8s需要的系统镜像，需注意pause的版本号

```
kubeadm config images list
```

```
 version: v1.22.2
k8s.gcr.io/kube-apiserver:v1.22.2
k8s.gcr.io/kube-controller-manager:v1.22.2
k8s.gcr.io/kube-scheduler:v1.22.2
k8s.gcr.io/kube-proxy:v1.22.2
k8s.gcr.io/pause:3.5
k8s.gcr.io/etcd:3.5.0-0
k8s.gcr.io/coredns/coredns:v1.8.4
```

修改daemon配置文件

```
vi /etc/isulad/daemon.json
```

```
##关于添加项的解释说明##
registry-mirrors 设置为"docker.io"
insecure-registries 设置为"rnd-dockerhub.huawei.com"
pod-sandbox-image 设置为"registry.aliyuncs.com/google_containers/pause:3.5"（使用阿里云，pause版本可在上一步查看）
network-plugin 设置为"cni"。
cni-bin-dir 设置为"/opt/cni/bin"；
cni-conf-dir 设置为"/etc/cni/net.d"
```

修改后的完整文件如下

```
{
    "group": "isula",
    "default-runtime": "lcr",
    "graph": "/var/lib/isulad",
    "state": "/var/run/isulad",
    "engine": "lcr",
    "log-level": "ERROR",
    "pidfile": "/var/run/isulad.pid",
    "log-opts": {
        "log-file-mode": "0600",
        "log-path": "/var/lib/isulad",
        "max-file": "1",
        "max-size": "30KB"
    },
    "log-driver": "stdout",
    "container-log": {
        "driver": "json-file"
    },
    "hook-spec": "/etc/default/isulad/hooks/default.json",
    "start-timeout": "2m",
    "storage-driver": "overlay2",
    "storage-opts": [
        "overlay2.override_kernel_check=true"
    ],
    "registry-mirrors": [
           "docker.io"
    ],
    "insecure-registries": [
           "rnd-dockerhub.huawei.com"
    ],
    "pod-sandbox-image": "registry.aliyuncs.com/google_containers/pause:3.5",
    "native.umask": "secure",
    "network-plugin": "cni",
    "cni-bin-dir": "/opt/cni/bin",
    "cni-conf-dir": "/etc/cni/net.d",
    "image-layer-check": false,
    "use-decrypted-key": true,
    "insecure-skip-verify-enforce": false
}
```

启动相关服务

```
systemctl restart isulad
systemctl enable isulad
systemctl enable kubelet
```

**以上为master，node节点均需执行的操作。**

# master节点初始化

**该部分仅master节点执行。**

初始化，在这一步会拉取镜像，需等待一小段时间。也可在该步骤之前手动拉取镜像。

```
kubeadm init --kubernetes-version=1.22.2 --apiserver-advertise-
address=192.168.237.133 --cri-socket=/var/run/isulad.sock --image-repository
registry.aliyuncs.com/google_containers --service-cidr=10.10.0.0/16 --pod-
network-cidr=10.122.0.0/16
```

```
##关于初始化参数的解释说明##

kubernetes-version 为当前安装的版本
apiserver-advertise-address 为master节点ip
cri-socket 指定引擎为isulad
image-repository 指定镜像源为阿里云，可省去修改tag的步骤
service-cidr 指定service分配的ip段
pod-network-cidr 指定pod分配的ip段
```

初始化成功后可看到如下界面：

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.237.131:6443 --token d9aau5.yzyehe49b5k9eixm \
        --discovery-token-ca-cert-hash sha256:2f514c6c009f00a3a11aac8c2591326f7e87a4be1f20590d097577e91856d7a5
```

复制最后两行内容方便后续node节点加入使用

```
kubeadm join 192.168.237.133:6443 --token j7kufw.yl1gte0v9qgxjzjw --discovery-
token-ca-cert-hash
sha256:73d337f5edd79dd4db997d98d329bd98020b712f8d7833c33a85d8fe44d0a4f5 --cri-
socket=/var/run/isulad.sock
```

**注意**：添加--cri-socket=/var/run/isulad.sock以使用isula为容器引擎

查看下载好的镜像

```
isula images
```

```
[root@k8s-master ~]# isula images
REPOSITORY                                                    TAG         IMAGE ID
registry.aliyuncs.com/google_containers/kube-apiserver        v1.22.2     e64579b7d886
registry.aliyuncs.com/google_containers/kube-controller-manager v1.22.2   5425bcbd23c5
registry.aliyuncs.com/google_containers/kube-scheduler        v1.22.2     b51ddc1014b0
registry.aliyuncs.com/google_containers/kube-proxy            v1.22.2     873127efbc8a
registry.aliyuncs.com/google_containers/etcd                  3.5.0-0     004811815584
registry.aliyuncs.com/google_containers/coredns               v1.8.4      8d147537fb7d
registry.aliyuncs.com/google_containers/pause                 3.5         ed210e3e4a5b
```

按照初始化成功所提示，配置集群

```
mkdir -p $HOME/.kube
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
chown $(id -u):$(id -g) $HOME/.kube/config
export KUBECONFIG=/etc/kubernetes/admin.conf
source /etc/profile
```

查看健康状态

```
kubectl get cs
```

```
[root@k8s-master ~]# kubectl get cs
Warning: v1 ComponentStatus is deprecated in v1.19+
NAME                 STATUS      MESSAGE
controller-manager   Unhealthy   Get "https://127.0.0.1:10257/h
scheduler            Unhealthy   Get "http://127.0.0.1:10251/he
etcd-0               Healthy     {"health":"true","reason":""}
```

可以看到controller-manager，scheduler状态为unhealthy，解决方法如下：

编辑相关配置文件

```
vi /etc/kubernetes/manifests/kube-controller-manager.yaml
```

```
注释如下内容：
- --port=0
修改hostpath：
将所有/usr/libexec/kubernetes/kubelet-plugins/volume/exec 修改为/opt/libexec/...
```

```
vi /etc/kubernetes/manifests/kube-scheduler.yaml
```

```
注释如下内容：
- --port=0
```

等待片刻后，再次查看健康状态

```
[root@k8s-master manifests]# kubectl get cs
Warning: v1 ComponentStatus is deprecated in v1.19+
NAME                 STATUS    MESSAGE                         ERROR
scheduler            Healthy   ok
controller-manager   Healthy   ok
etcd-0               Healthy   {"health":"true","reason":""}
```

# 配置网络插件

仅需要在master节点配置网络插件，但是要在**所有节点**提前拉取镜像，拉取镜像指令如下。

```
isula pull calico/node:v3.19.3
isula pull calico/cni:v3.19.3
isula pull calico/kube-controllers:v3.19.3
isula pull calico/pod2daemon-flexvol:v3.19.3
```

**以下步骤仅在master节点执行**

获取配置文件

```
wget https://docs.projectcalico.org/v3.19/manifests/calico.yaml
```

编辑calico.yaml 修改所有/usr/libexec/... 为 /opt/libexec/...

然后执行如下命令完成calico的安装：

```
kubectl apply -f calico.yaml
```

通过kubectl get pod -n kube-system查看calico是否安装成功
通过kubectl get pod -n kube-system查看是否所有pod状态都为running

```
[root@k8s-master ~]# kubectl get pod -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
calico-kube-controllers-6b59cd85f8-wb76k   1/1     Running   0          25s
calico-node-zx4wf                          1/1     Running   0          25s
coredns-7f6cbbb7b8-klsj5                   1/1     Running   0          5m14s
coredns-7f6cbbb7b8-qh6t5                   1/1     Running   0          5m14s
etcd-k8s-master                            1/1     Running   0          8m49s
kube-apiserver-k8s-master                  1/1     Running   0          8m49s
kube-controller-manager-k8s-master         1/1     Running   0          5m29s
kube-proxy-lm9kq                           1/1     Running   0          5m14s
kube-scheduler-k8s-master                  1/1     Running   0          5m9s
```

# node节点加入集群

在node节点执行如下指令，将node节点加入集群

```
kubeadm join 192.168.237.133:6443 --token j7kufw.yl1gteOv9qgxjzjw --discovery-
token-ca-cert-hash
sha256:73d337f5edd79dd4db997d98d329bd98020b712f8d7833c33a85d8fe44d0a4f5 --cri-
socket=/var/run/isulad.sock
```

通过kubectl get node 查看master，node节点状态是否为ready

```
[root@k8s-master ~]# kubectl get node
NAME         STATUS   ROLES                  AGE   VERSION
k8s-master   Ready    control-plane,master   10m   v1.22.2
k8s-node01   Ready    <none>                 83s   v1.22.2
```

再次查看pod

```
[root@k8s-master ~]# kubectl get pod -n kube-system
NAME                                       READY   STATUS    RESTARTS      AGE
calico-kube-controllers-6b59cd85f8-wb76k   1/1     Running   0             3m22s
calico-node-sxndw                          1/1     Running   0             2m34s
calico-node-zx4wf                          1/1     Running   0             3m22s
coredns-7f6cbbb7b8-klsj5                   1/1     Running   1 (10s ago)   8m11s
coredns-7f6cbbb7b8-qh6t5                   1/1     Running   1 (10s ago)   8m11s
etcd-k8s-master                            1/1     Running   0             11m
kube-apiserver-k8s-master                  1/1     Running   0             11m
kube-controller-manager-k8s-master         1/1     Running   0             8m26s
kube-proxy-bpmjm                           1/1     Running   0             2m34s
kube-proxy-lm9kq                           1/1     Running   0             8m11s
kube-scheduler-k8s-master                  1/1     Running   0             8m6s
```

至此，k8s部署成功。

# 功能特性

## 容器技术

NestOS通过容器化 (containerized) 的运算环境向应用程序提供运算资源。应用程序之间共享系统内核和资源，但是彼此之间又互不可见。这样就意味着应用程序将不会再被直接安装到操作系统中，而是通过 iSulad 运行在容器中。这种方式使得操作系统、应用程序及运行环境之间的耦合度大大降低。相对于传统的部署方式而言，在NestOS 集群中部署应用程序更加灵活便捷，应用程序运行环境之间的干扰更少，而且操作系统自身的维护也更加容易。

## Rpm-ostree

### 系统更新

rpm-ostree是一种镜像/包混合系统，可以看成是rpm和ostree的结合体。一方面它提供了基于rpm的软件包安装，另一方面它提供了基于ostree的操作系统更新升级，而且rpm-ostree将这两种操作都视为对操作系统的更新，每次对系统的更新都像提交“Transaction-事务”，从而确保更新全部成功或全部失败，并且允许在更新系统后回滚到更新前的状态。

rpm-ostree在更新操作系统的时候会有2个bootable区域，一个是更新前一个是更新后的，更新升级只有在重启操作系统后才生效。如果软件安装或升级出现问题，回滚和重新启动会使NestOS系统返回到先前的状态。我们可以查看NestOS的“/ostree/”和“/boot/”目录，它们是ostree repository环境和boot使用的哪个ostree。

### 文件系统

在rpm-ostree的文件系统布局中，只有/etc和/var是唯一可写的目录，/var中的任何数据不会被触及，而是在升级过程中共享。在升级时，该过程采用新的默认值/etc，并将更改添加到顶部。这意味着升级将会接收/etc中新的默认文件，这是一个非常关键的特性。/usr始终具有只读绑定挂载。

Ostree旨在并行安装多个独立操作系统的版本，ostree依赖于一个新的ostree 目录，该目录实际上可以并行安装在现有的操作系统或者是占据物理/root目录的发行版本中。每台客戶机和每组部署上都存储在 /ostree/deploy/$STATEROOT/$CHECKSUM 上，而且还有一个ostree存储库存储在 /ostree/repo 中。每个部署主要由一组指向存储库的硬链接组成，这意味着每个版本都进行了重复数据的删除并且升级过程中只消耗了与新文件成比例的磁盘空间加上一些恒定的开销。

Ostree模型强调的是OS只读内容保存在 /usr 中，它附带了用于创建Linux只读安装以防止无意损坏的代码，对于给定的操作系统，每个部署之间都有一个 /var 共享的可供读写的目录。Ostree核心代码不触及该目录的内容，如何管理和升级状态取决于每个操作系统中的代码。

### 系统扩展

出于安全性和可维护性的考虑，NestOS让基础镜像尽可能保持小巧和精简。但是在某些情况下，需要向基本操作系统本身添加软件，例如驱动软件，VPN

等等，因为它们比较难容器化。为此，rpm-ostree install将这些包视为拓展，它们拓展了操作系统的功能，而不是仅仅在用户运行时提供。也就是说，对于实际安装哪些包没有限制，默认情况下，软件包是从openEuler仓库下载的。

要对软件包进行分层， 需要重新编写一个systemd单元来执行rpm-ostree命令安装所需要的包，所做的更改应用于新部署，重新启动才能生效。

## Nestos-installer

nestos-installer是一个帮助安装NestOS的程序，它可以执行以下操作：

（1）安装操作系统到一个目标磁盘，可使用Ignition和首次引导内核参数对其进行对其进行自定义(nestos-installer install)

（2）下载并验证各种云平台，虚拟化或者裸机平台的操作系统映像(nestos-installer download)

（3）列出可供下载的Coreos镜像(nestos-installer list-stream)

（4）在ISO中嵌入一个Ignition配置，以自定义地从中启动操作系统(nestos-installer iso ignition)

（5）将Ignition配置包装在initd映像中，该映像可以被加入到PXE initramfs中以自定义从中启动的操作系统(nestos-installer pxe ignition)

## Zincati

Zincati是NestOS自动更新的代理，它作为Cincinnati和rpm-ostree的客户端，负责自动更新/重启机器。Zincati有如下特点：

（1）支持自动更新代理，支持分阶段推出

（2）通过TOML dropins和重叠目录配置

（3）用于完成/重启的多种更新策略

（4）每周计划的本地维护窗口计划升级

（5）以Prometheue格式通过本地端口公开的内部指标

（6）具有可配置优先级的日志记录

（7）通过Cincinnati协议支持复杂的更新图

（8）通过外部锁管理器支持集群范围的重启编排

## Cincinnati

Cincinnati是一个更新协议，旨在促进自动更新。Cincinnati使用有向无环图存入版本信息表示项目版本之间的转换。Cincinati建立两个服务端，一个更新版本信息的有向无环图，另一个接收处理客户端zincati自动更新服务的请求，允许客户端在这些版本之间执行自动更新。

## Ignition 系统初始化

Ignition 是一个与分发无关的配置实用程序，不仅用于安装，还读取配置文件（JSON 格式）并根据该配置NestOS系统。可配置的组件包括存储和文件系统、systemd单元和用户。

Ignition仅在系统第一次引导期间运行一次（在initramfs中）。因为 Ignition 在启动过程的早期运行，所以它可以在用户空间开始启动之前重新分区磁盘、格式化文件系统、创建用户和写入文件。 因此，systemd 服务在 systemd 启动时已经写入磁盘，从而加快了启动速度。

（1）Ignition 仅在第一次启动时运行

Ignition 旨在用作配置工具，而不是配置管理工具。 Ignition 鼓励不可变的基础设施，其中机器修改要求用户丢弃旧节点并重新配置机器。

（2）Ignition不是在任何情况下都可以完成配置

Ignition 执行它需要的操作，使系统与 Ignition 配置中描述的状态相匹配。 如果由于任何原因 Ignition 无法提供配置要求的确切机器，Ignition 会阻止机器成功启动。例如，如果用户想要获取托管在 https://example.com/foo.conf 的文档并将其写入磁盘，如果无法解析给定的 URL，Ignition 将阻止机器启动。

（3）Ignition只是声明性配置

Ignition配置只描述了系统的状态，没有列出 Ignition 应该采取的一系列步骤。

Ignition 配置不允许用户提供任意逻辑（包括 Ignition 运行的脚本）。 用户描述哪些文件系统必须存在、哪些文件必须创建、哪些用户必须存在等等。任何进一步的定制都必须使用由 Ignition 创建的 systemd 服务。

（4）Ignition配置不应手写

Ignition 配置被设计为人类可读但难以编写，是为了阻止用户尝试手动编写配置。可以使用Butane或类似工具生成或转化生成Ignition 配置。

## Afterburn

Afterburn是类似于云平台一样的一次性代理，他可以用于与特定的提供者的元数据端点进行交互，通常和Ignition结合使用。

Afterburn包含了很多可以在实例生命周期中不同时间段运行的模块。下面的服务是根据特定的平台可能在第一次启动时在initramfs中运行的：

（1）设置本地主机名

（2）加入网络命令行参数

以下的功能是在一定条件下，作为systemd服务单元在一些平台上才能使用的：

（1）为本地系统用户安装公共SSH密钥

（2）从实例元数据中检索属性

（3）给提供者登记以便报道成功的启动或实例供应

# 性能对比测试

使用NestOS 20211009版本，横向对比 docker，podman，isula 容器引擎性能。测试结果如下。

x86 machine :

| Configuration | Information |
| --- | --- |
| OS | NestOS |
| Kernel | linux 5.10.0 |
| CPU | 8 cores |
| Memory | 16 GB |

软件版本：

| Name | Version |
| --- | --- |
| iSulad | Version 2.0.8, commit 9aa57ef27d3719729097f75c65125519497b8b66 |
| docker | Version: 18.09.0, Git commit: 1c709d9 |
| podman | version 0.10.1 |

测试结果：

| operator (ms) | Docker | Podman | iSulad | vs Docker | vs Podman |
| --- | --- | --- | --- | --- | --- |
| 100 * create | 1745 | 22919 | 1122 | -36% | -95% |
| 100 * start | 8561 | 8133 | 1561 | -82% | -81% |
| 100 * stop | 1483 | 1445 | 296 | -80% | -80% |
| 100 * rm | 1691 | 5268 | 741 | -56% | -86% |

.