目录

[Firecracker使用手册 1](#_Toc89868077)

[1先决条件 1](#_Toc89868078)

[1.1 /dev/kvm读写权限 1](#_Toc89868079)

[1.2 内核版本 1](#_Toc89868080)

[2 Firecracker二进制文件 1](#_Toc89868081)

[2.1获得最新的firecracker二进制文件 1](#_Toc89868082)

[2.2重命名二进制文件“firecracker” 2](#_Toc89868083)

[3 运行firecracker 2](#_Toc89868084)

[3.1启动firecracker 2](#_Toc89868085)

[3.2 获得内核kernel和跟文件系统rootfs 2](#_Toc89868086)

[3.3通过REST API配置kernel和rootfs 3](#_Toc89868087)

[3.4通过REST API启动虚拟机 4](#_Toc89868088)

[3.5通过REST API关闭虚拟机 5](#_Toc89868089)

[4 扩展：配置虚拟机 6](#_Toc89868090)

[5 扩展实验 6](#_Toc89868091)

[5.1 内核kernel裁剪 6](#_Toc89868092)

[5.2 跟文件系统rootfs制作 7](#_Toc89868093)

[5.3 firecracker启动裁剪内核和rootfs 8](#_Toc89868094)

[6 扩展实验参考 8](#_Toc89868095)

# Firecracker使用手册

## 1先决条件

### 1.1 /dev/kvm读写权限

Firecracker需要使用kvm；当前登录linux 服务器的用户，需要增加读写权限到 /dev/kvm

sudo setfacl -m u:${USER}:rw /dev/kvm

### 1.2 内核版本

内核版本需要4.14+，才可以支持firecracker的运行，在ubuntu查看内核版本的命令：

uname -r



确保kvm虚拟化已经在linux服务器中开启：

kvm-ok

屏幕上有字

描述已自动生成

## 2 Firecracker二进制文件

### 2.1获得最新的firecracker二进制文件

release\_url="https://github.com/firecracker-microvm/firecracker/releases"

latest=$(basename $(curl -fsSLI -o /dev/null -w %{url\_effective} ${release\_url}/latest))

arch=`uname -m`

curl -L ${release\_url}/download/${latest}/firecracker-${latest}-${arch}.tgz | tar -xz

图形用户界面

中度可信度描述已自动生成

### 2.2重命名二进制文件“firecracker”

mv firecracker-${latest}-$(uname -m) firecracker



## 3 运行firecracker

### 3.1启动firecracker

开启一个终端，按如下操作：

#### 3.1.1保证firecracker可以创建API socket：

rm -f /tmp/firecracker.socket



#### 3.1.2启动firecracker

在firecracker目录下执行

./firecracker --api-sock /tmp/firecracker.socket



开启第二个终端，按如下操作

### 3.2 获得内核kernel和跟文件系统rootfs

运行run0.sh获得kernel和rootfs，



run0.sh脚本内容如下：

arch=`uname -m`

dest\_kernel="hello-vmlinux.bin"

dest\_rootfs="hello-rootfs.ext4"

image\_bucket\_url="https://s3.amazonaws.com/spec.ccfc.min/img/quickstart\_guide/$arch"

if [ ${arch} = "x86\_64" ]; then

kernel="${image\_bucket\_url}/kernels/vmlinux.bin"

rootfs="${image\_bucket\_url}/rootfs/bionic.rootfs.ext4"

elif [ ${arch} = "aarch64" ]; then

kernel="${image\_bucket\_url}/kernels/vmlinux.bin"

rootfs="${image\_bucket\_url}/rootfs/bionic.rootfs.ext4"

else

echo "Cannot run firecracker on $arch architecture!"

exit 1

fi

echo "Downloading $kernel..."

curl -fsSL -o $dest\_kernel $kernel

echo "Downloading $rootfs..."

curl -fsSL -o $dest\_rootfs $rootfs

echo "Saved kernel file to $dest\_kernel and root block device to $dest\_rootfs."

### 3.3通过REST API配置kernel和rootfs

运行run1.sh 配置kernel和rootfs



run1.sh脚本内容如下

arch=`uname -m`

kernel\_path=$(pwd)"/hello-vmlinux.bin"

if [ ${arch} = "x86\_64" ]; then

curl --unix-socket /tmp/firecracker.socket -i \

-X PUT 'http://localhost/boot-source' \

-H 'Accept: application/json' \

-H 'Content-Type: application/json' \

-d "{

\"kernel\_image\_path\": \"${kernel\_path}\",

\"boot\_args\": \"console=ttyS0 reboot=k panic=1 pci=off\"

}"

elif [ ${arch} = "aarch64" ]; then

curl --unix-socket /tmp/firecracker.socket -i \

-X PUT 'http://localhost/boot-source' \

-H 'Accept: application/json' \

-H 'Content-Type: application/json' \

-d "{

\"kernel\_image\_path\": \"${kernel\_path}\",

\"boot\_args\": \"keep\_bootcon console=ttyS0 reboot=k panic=1 pci=off\"

}"

else

echo "Cannot run firecracker on $arch architecture!"

exit 1

fi

rootfs\_path=$(pwd)"/hello-rootfs.ext4"

curl --unix-socket /tmp/firecracker.socket -i \

-X PUT 'http://localhost/drives/rootfs' \

-H 'Accept: application/json' \

-H 'Content-Type: application/json' \

-d "{

\"drive\_id\": \"rootfs\",

\"path\_on\_host\": \"${rootfs\_path}\",

\"is\_root\_device\": true,

\"is\_read\_only\": false

}"

### 3.4通过REST API启动虚拟机

运行run2.sh 启动虚拟机

文本

描述已自动生成

run2.sh脚本内容如下:

curl --unix-socket /tmp/firecracker.socket -i \

-X PUT 'http://localhost/actions' \

-H 'Accept: application/json' \

-H 'Content-Type: application/json' \

-d '{

"action\_type": "InstanceStart"

}'

返回第一个终端，可以看到虚拟机已经启动

文本

描述已自动生成

### 3.5通过REST API关闭虚拟机

运行run3.sh 关闭虚拟机

文本

描述已自动生成

run3.sh脚本内容如下:

curl --unix-socket /tmp/firecracker.socket -i \

-X PUT "http://localhost/actions" \

-H "accept: application/json" \

-H "Content-Type: application/json" \

-d '{"action\_type": "SendCtrlAltDel"}'

返回第一个终端可以看到虚拟机已经关闭，终端返回到ubuntu操作系统

文本

描述已自动生成

## 4 扩展：配置虚拟机

上面默认开启虚拟机只有一个CPU，运行内存是128M，在启动虚拟机（运行脚本run2.sh）之前，可以个性化的配置虚拟机的资源，例如配置2个CPU，运行内存是1024M。

运行run4.sh 配置虚拟机，run4.sh脚本内容如下:

curl --unix-socket /tmp/firecracker.socket -i \

-X PUT 'http://localhost/machine-config' \

-H 'Accept: application/json' \

-H 'Content-Type: application/json' \

-d '{

"vcpu\_count": 2,

"mem\_size\_mib": 1024,

"ht\_enabled": false

}'

## 5 扩展实验

### 5.1 内核kernel裁剪

当前，在X86\_64架构中firecracker支持未压缩过的ELF内核镜像。

1）获得内核源代码

git clone https://github.com/torvalds/linux.git linux.git

cd linux.git

2）check out到指定的内核版本

git checkout v4.20

3）配置内核编译选项

make menuconfig（可以省略此步骤）

.config 文件可以参考github

https://github.com/juliwang0205/firecracker-experiment/tree/master

图形用户界面, 文本, 应用程序

描述已自动生成

4）编译内核镜像

make vmlinux

在linux.git目录下，生产成vmlinux内核镜像。

### 5.2 跟文件系统rootfs制作

根文件系统镜像，就是一个文件系统镜像，需要最少有一个init系统。

1. 制作一个虚拟的硬盘

dd if=/dev/zero of=rootfs.ext4 bs=1M count=50

1. 格式化虚拟硬盘

mkfs.ext4 rootfs.ext4

现在已经有一个空的EXT4镜像，接下来填充它。实验以 Alpine-based rootfs为例，创建跟文件系统。

1. 挂载到 /tmp/my-rootfs

mkdir /tmp/my-rootfs

sudo mount rootfs.ext4 /tmp/my-rootfs

1. 开始Alpine 容器，绑定到/tmp/my-rootfs

docker run -it --rm -v /tmp/my-rootfs:/my-rootfs alpine

1. 安装openrc和基础的linux工具

# apk add openrc

# apk add util-linux

1. 建立一个用户态的init

# ln -s agetty /etc/init.d/agetty.ttyS0

# echo ttyS0 > /etc/securetty

# rc-update add agetty.ttyS0 default

# rc-update add devfs boot

# rc-update add procfs boot

# rc-update add sysfs boot

# for d in bin etc lib root sbin usr; do tar c "/$d" | tar x -C /my-rootfs; done

# for dir in dev proc run sys var; do mkdir /my-rootfs/${dir}; done

# exit

4）卸载rootfs.ext4镜像

rootfs.ext4制作完成之前，修改/etc/passwd文件

root:x:0:0:root:/root:/bin/sh

修改后为root::0:0::/root:/bin/sh

保证root可以免密码登陆

# umount /tmp/my-rootfs

### 5.3 firecracker启动裁剪内核和rootfs

替换run1.sh内核和文件系统，重启运行run1.sh

裁剪的内核已经启动：

手机屏幕的截图

描述已自动生成

## 6 扩展实验参考

扩展实验的kernel和rootfs可以参考下面链接的内容。

https://github.com/juliwang0205/firecracker-experiment/tree/master