目 录

1 第一个例子

1.1 配置 baremetal provisioning 驱动

可以修改配置文件/etc/ironic/ironic.conf 来设置 openstack 启用对应驱动：

```
# 可以用逗号分隔来指定多个驱动
enabled_drivers=pxe_ipmitool
```

修改后需要重启服务:

```
systemctl restart openstack-ironic-conductor.service
```

1.2 上传镜像到 glance 服务器

openstack 要实现部署裸机需要用到的镜像有 5 个。这 5 个镜像有两个是用作 deploy，即被用来在安装操作系统前对裸机节点进行准备。有两镜像个用作系统的启动引导，还有一个就是系统镜像。

命令如下：

```
# 上传用于deploy的镜像
glance image-create --name deploy_kernel --is-public true \
--disk-format aki \
--file deploy.kernel
glance image-create --name deploy_initramfs --is-public true \
--disk-format ari \
--file deploy.initramfs

# 上传用于boot镜像
glance image-create --name boot_kernel --is-public true \
--disk-format aki \
--file boot.vmlinuz
glance image-create --name boot_initrd --is-public true \
--disk-format ari \
--file boot.initrd

# 上传系统镜像
glance image-create --name NAME --is-public true \
--disk-format qcow2 \
--container-format bare \
--property kernel_id=$boot_kernel_uuid \
--property ramdisk_id=$boot_initrd_uuid \
--property hypervisor_type=ironic \
--file image.qcow2
```

1.3 把物理机注册为裸机节点

命令如下:

```
# 创建新节点
ironic node-create -d pxe_ipmitool

# 创建逻辑名
ironic node-update <node-uuid> add name=<node-name>

# 可以通过下面的命令查看对于pxe_ipmitool，哪些驱动信息必须被添加
ironic driver-properties pxe_ipmitool

# 为主机添加IPMI驱动信息
ironic node-update <node-uuid> add \
    driver_info/ipmi_username=<username> \
    driver_info/ipmi_password=<password> \
    driver_info/ipmi_address=<HOST-IP>

# 添加用于deploy的镜像的uuid
ironic node-update <node-uuid> add \
    driver_info/pxe_deploy_kernel=<deploy-kernel-uuid> \
    driver_info/pxe_deploy_ramdisk=<deploy-ramdisk-uuid>

# 设置裸机硬件的规格
ironic node-update <node-uuid> add \
    properties/cpus=4 \
    properties/memory_mb=98304 \
    properties/local_gb=80 \
    properties/cpu_arch=x86_64

#  配置为本地引导（pxe初始化实施后的引导方式,flavor也需要设置）
ironic node-update <node-uuid> add  \
    properties/capabilities="boot_option:local"

# 添加mac port(需要分配ip的所有网卡都要添加)
ironic port-create -n <node-uuid> -a <mac-address>

# 检验节点的设置
ironic node-validate <node-uuid>
```

1.4 创建物理机需要的 flavor

命令如下:

```
nova flavor-create <flavor-name> auto 512 20 1
nova flavor-key <flavor-name> set cpu_arch="x86_64"
nova flavor-key <flavor-name> set capabilities:boot_option="local"
```

openstack 通过指定实例的 flavor 来确定该实例生成在哪一个裸机中:

```
ironic node-update $NODE_UUID add \
```

```
    properties/capabilities='profile:baremetal,boot_option:local'

nova flavor-key $FLAVOR_NAME set capabilities:profile="baremetal"
```

2 第二个例子

2.1 创建虚拟机

```
sudo -E su pengsida -c '/opt/stack/ironic/devstack/tools/ironic/scripts/create-node.sh -n node-4 -c 1 -m 1280 -d 10 -a x86_64 -b brbm -e /usr/bin/qemu-system-x86_64 -E qemu -p 6233 -o 4 -f qcow2 -l /home/pengsida/temp/ironic-bm-logs'
```

```
# 设置虚拟机的存储池
virsh pool-define-as --name default dir --target /var/lib/libvirt/images
virsh pool-autostart default
virsh pool-start default
```

```
domain_name="node-4"
# 创建虚拟机使用的网桥
brctl addbr br-$domain_name
ip link set br-$domain_name up

# 在网桥brbm上创建新的端口
ovs-vsctl add-port brbm ovs-$domain_name -- set Interface ovs-$domain_name type=internal
ip link set ovs-$domain_name up

# 将ovs-node端口加入br-node这个网桥
brctl addif br-$domain_name ovs-$domain_name

# 创建虚拟机的磁盘镜像文件
virsh vol-create-as default $domain_name.qcow2 11G --format qcow2 --prealloc-metadata
virsh vol-path --pool default $domain_name.qcow2
touch /var/lib/libvirt/images/$domain_name.qcow2
chattr +C /var/lib/libvirt/images/$domain_name.qcow2
/opt/stack/ironic/devstack/tools/ironic/scripts/configure-vm.py --bootdev network --name $domain_name --image /var/lib/libvirt/images/$domain_name.qcow2 --arch x86_64 --cpus 1 --memory 1310720 --libvirt-nic-driver virtio --bridge br-$domain_name --disk-format qcow2 --console-log /home/pengsida/temp/ironic-bm-logs/"$domain_name"_console.log --engine qemu --emulator /usr/bin/qemu-system-x86_64

# 设置虚拟机的监听端口
vbmc add $domain_name --port 6234
# 启动虚拟机的监听端口
vbmc start $domain_name
```

2.2 注册物理机

```
sudo -E su pengsida -c '/opt/stack/ironic/devstack/tools/ironic/scripts/create-node.sh -n node-3 -c 1 -m 1280 -d 10 -a x86_64 -b brbm -e /usr/bin/qemu-system-x86_64 -E qemu -p 6233 -o 3 -f qcow2 -l /home/pengsida/temp/ironic-bm-logs'
```

```
# 注册物理机
domain_name="node-4"
node_uuid=`uuidgen`
first_chassis_uuid=`openstack baremetal chassis list -f value | head -1 | cut -d" " -f1`
ramdisk_uuid=`openstack image list | grep "\.initramfs" | cut -d"|" -f2 | cut -d" " -f2`
kernel_uuid=`openstack image list | grep "\.kernel" | cut -d"|" -f2 | cut -d" " -f2`
port_num=`vbmc list | grep $domain_name | egrep -o "| [0-9]+ |" | cut -d" " -f2`
vcpu_num=`virsh dumpxml $domain_name | grep vcpu | cut -d">" -f2 | cut -d"<" -f1`
memory=$(echo "`virsh dumpxml $domain_name | grep "memory" | cut -d">" -f2 | cut -d"<" -f1`/1024" | bc)
arch=`virsh dumpxml $domain_name | grep arch | cut -d"'" -f2 | cut -d"'" -f1`
disk_path=`virsh dumpxml $domain_name | grep "source file" | cut -d"'" -f2 | cut -d"'" -f1`
disk_size=`qemu-img info $disk_path | grep "virtual size" | egrep -o "[0-9]+G" | cut -d"G" -f1`
local_size=$[ $disk_size-1 ]
driver_address=`openstack endpoint list | grep ironic | grep admin | cut -d"/" -f3 | cut -d":" -f1`
domain_address=`virsh dumpxml $domain_name | grep 'mac address' | head -1 | cut -d"'" -f2`

openstack flavor create --ephemeral 0 --ram $memory --disk $local_size --vcpus $vcpu_num $domain_name
openstack flavor set $domain_name --property cpu_arch=$arch
# 设置与domain关联
nova flavor-key $domain_name set capabilities:profile="$domain_name"

ironic node-create --uuid $node_uuid --chassis_uuid $first_chassis_uuid --driver agent_ipmitool --name $domain_name -p cpus=$vcpu_num -p memory_mb=$memory -p local_gb=$local_size -p cpu_arch=$arch -i ipmi_address=$driver_address -i ipmi_username=admin -i ipmi_password=password -i deploy_kernel=$kernel_uuid -i deploy_ramdisk=$ramdisk_uuid -i ipmi_port=$port_num
ironic port-create --address $domain_address --node $node_uuid
# 设置与flavor关联
ironic node-update $NODE_UUID add properties/capabilities="profile:$domain_name"
```

2.3 启动虚拟机

```
private_network=$(openstack network list -f value -c ID --name private)
domain_name="node-0"
openstack server create --flavor $domain_name --image cirros-0.3.5-x86_64-disk
    --nic net-id=$private_network $domain_name
```

3 第三个例子

```
# 创建虚拟机
export domain_name="node-0"
export port="6230"
export num=$(echo $domain_name | cut -d"-" -f2)
export port=$[ $port + $num ]
export outlet=$[ $num + 1 ]
sudo vbmc delete $domain_name
sudo virsh vol-delete $domain_name.qcow2 default
sudo virsh undefine $domain_name
sudo ip link set br-$domain_name down
sudo brctl delbr br-$domain_name
sudo ovs-vsctl del-port brbm ovs-$domain_name
sudo -E su pengsida -c '/opt/stack/ironic/devstack/tools/ironic/scripts/create-
    node.sh -n $domain_name -c 1 -m 1280 -d 10 -a x86_64 -b brbm -e /usr/bin/
    qemu-system-x86_64 -E qemu -p $port -o $outlet -f qcow2 -l /home/pengsida/
    temp/ironic-bm-logs'

# 注册虚拟机
node_uuid=`uuidgen`
# ironic chassis-create -d 'ironic test chassis'
first_chassis_uuid=`openstack baremetal chassis list -f value | head -1 | cut -
    d" " -f1`
if [ -z $first_chassis_uuid ]; then ironic chassis-create -d 'ironic test
    chassis'; first_chassis_uuid=`openstack baremetal chassis list -f value |
    head -1 | cut -d" " -f1`; fi
ramdisk_uuid=`openstack image list | grep "\.initramfs" | cut -d"|" -f2 | cut -
    d" " -f2`
kernel_uuid=`openstack image list | grep "\.kernel" | cut -d"|" -f2 | cut -d" "
    -f2`
port_num=`vbmc list | grep $domain_name | egrep -o "| [0-9]+ |" | cut -d" " -f2
    `
vcpu_num=`sudo virsh dumpxml $domain_name | grep vcpu | cut -d">" -f2 | cut -d"
    <" -f1`
memory=$(echo "`sudo virsh dumpxml $domain_name | grep "memory" | cut -d">" -f2
    | cut -d"<" -f1`/1024" | bc)
arch=`sudo virsh dumpxml $domain_name | grep arch | cut -d"'" -f2 | cut -d"'" -
    f1`
disk_path=`sudo virsh dumpxml $domain_name | grep "source file" | cut -d"'" -f2
    | cut -d"'" -f1`
disk_size=`sudo qemu-img info $disk_path | grep "virtual size" | egrep -o "
    [0-9]+G" | cut -d"G" -f1`
local_size=$[ $disk_size-1 ]
driver_address=`openstack endpoint list | grep ironic | grep admin | cut -d"/"
    -f3 | cut -d":" -f1`
```

```
30      domain_address=`sudo virsh dumpxml $domain_name | grep 'mac address' | head -1
            | cut -d"'" -f2`
31
32      # 创建相应的flavor
33      openstack flavor delete $domain_name
34      openstack flavor create --ephemeral 0 --ram $memory --disk $local_size --vcpus
            $vcpu_num $domain_name
35      openstack flavor set $domain_name --property cpu_arch=$arch
36
37      # 设置flavor与domain关联
38      nova flavor-key $domain_name set capabilities:profile="$domain_name"
39
40      ironic node-delete $domain_name
41
42      ironic node-create --uuid $node_uuid --chassis_uuid $first_chassis_uuid --
            driver agent_ipmitool --name $domain_name -p cpus=$vcpu_num -p memory_mb=
            $memory -p local_gb=$local_size -p cpu_arch=$arch -i ipmi_address=
            $driver_address -i ipmi_username=admin -i ipmi_password=password -i
            deploy_kernel=$kernel_uuid -i deploy_ramdisk=$ramdisk_uuid -i ipmi_port=
            $port_num
43      ironic port-create --address $domain_address --node $node_uuid
44      # 设置与flavor关联
45      ironic node-update $node_uuid add properties/capabilities="profile:$domain_name
            "
46
47      # 远程在裸机上安装操作系统
48      nova-manage cell_v2 discover_hosts --verbose
49      PRIVATE_NETWORK_NAME="private"
50      net_id=$(openstack network list | egrep "$PRIVATE_NETWORK_NAME" '[^-]' | awk '{
            print $2 }')
51      openstack server create --flavor $domain_name --nic net-id=$net_id --image
            cirros-0.3.5-x86_64-disk testing
```