## Openstack Mitaka版本Ironic组件的基本部署

前面部署ironic都是基于Centos7系统，然后采用packstack+rdo方式安装部署的openstack mitaka版本，再手动部署ironic组件。

这里如果采用TECS 3.0，那么请自行确保nova、glance、neutron组件正常，尤其是neutron里面的dhcp服务、l3服务正常。

Ironic的部署跟Kilo版本还是有点小区别，不再采用keystone CLI命令，而是直接采用openstack CLI命令。下面结合ironic mitaka版本的安装手册给出前面的部署过程。社区文档路径：http://docs.openstack.org/project-install-guide/baremetal/draft/install-rdo.html

### 配置Identity service

执行下面4个命令：

openstack user create --password $IRONIC\_PASSWORD --email ironic@example.com ironic

openstack role add --project services --user ironic admin

openstack service create --name ironic --description "Ironic baremetal provisioning service" baremetal

openstack endpoint create --region RegionOne \

--publicurl http://$IRONIC\_NODE:6385 \

--internalurl http://$IRONIC\_NODE:6385 \

--adminurl http://$IRONIC\_NODE:6385 \

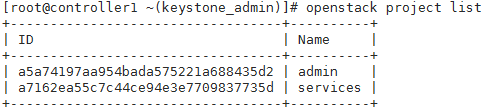
$(openstack service list | awk '/baremetal/ {print $2}')

说明：

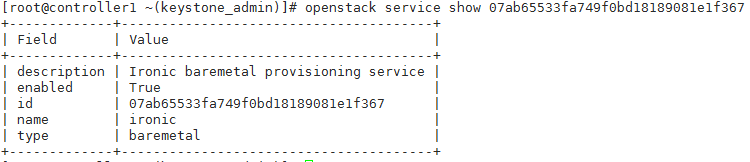
1. 上面$IRONIC\_PASSWORD，包括后面的$IRONIC\_NODE，需要用自己实际的字符串或IP、密码等替代，后面雷同；
2. 第二个命令增加role权限，这里” --project services”后面的services需要参考实际的openstack部署环境，建议通过openstack project list看一下到底是service还是services，社区文档这块全部是按照”service”来举例说明的，包括后面的ironic.conf里面的相关配置；
3. 上面这种写法是考虑在脚本中自动实现，下面雷同。

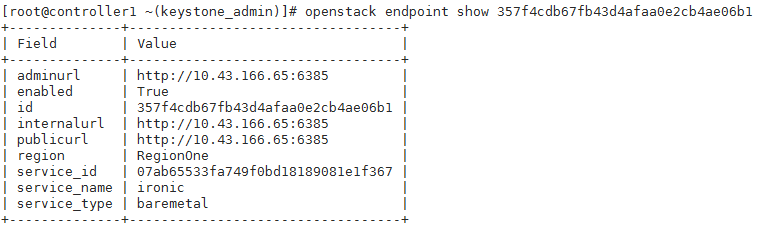
执行后环境贴图：











### 创建bare metal数据库

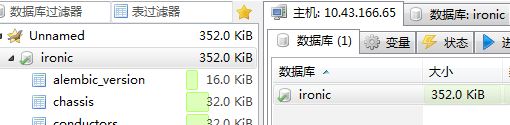
这里既可以按照社区文档里面的方式操作，也可以按照下面的方式直接在脚本中实现。

mysql -uroot -p$MARIADB\_PASSWORD -h127.0.0.1 -e "CREATE DATABASE ironic CHARACTER SET utf8;"

mysql -uroot -p$MARIADB\_PASSWORD -h127.0.0.1 -e "GRANT ALL PRIVILEGES ON ironic.\* TO 'ironic'@'localhost' IDENTIFIED BY '$IRONIC\_DBPASSWORD';"

mysql -uroot -p$MARIADB\_PASSWORD -h127.0.0.1 -e "GRANT ALL PRIVILEGES ON ironic.\* TO 'ironic'@'%' IDENTIFIED BY '$IRONIC\_DBPASSWORD';"

在数据库里创建ironic数据库并给ironic用户权限，可以用HeidiSQL工具看。



### 安装ironic组件

yum -y install openstack-ironic-api openstack-ironic-conductor python-ironicclient

然后需要手动创建ironic的日志文件所在文件夹，并修改属性，否则后面的enable和start操作会失败：

mkdir -p /var/log/ironic

chown -R ironic:ironic /var/log/ironic/

使能激活ironic组件：

systemctl enable openstack-ironic-api openstack-ironic-conductor

systemctl start openstack-ironic-api openstack-ironic-conductor

这里可以吧tecs3.0的bin包解压做成yum源操作。

### 配置ironic-api服务

这里还是建议严格按照社区文档给出的配置项进行操作，配置后还要保证/etc/ironic/ironic.conf文件的所有者和组用户为root:ironic。

前面搭建的时候这块是采用awk脚本自动完成的，下面给出awk脚本和如何使用。

Awk脚本：



使用方法：

ironic\_conf=/etc/ironic/ironic.conf

ironic\_conf\_new=/etc/ironic/ironic.conf.new

./replace\_section\_key.awk -v section=[database] -v key=connection \

-v keyvalue=mysql://ironic:$IRONIC\_DBPASSWORD@$DB\_IP/ironic?charset=utf8 \

-v outfile=$ironic\_conf\_new $ironic\_conf

mv -f $ironic\_conf\_new $ironic\_conf

上面一句话就是想把[database]里面的connection赋值为mysql://ironic:$IRONIC\_DBPASSWORD@$DB\_IP/ironic?charset=utf8。

#### 修改配置文件

Ironic-api组件需要修改ironic.conf的配置项有下面这些：

**[database]部分**

connection= mysql://ironic:$IRONIC\_DBPASSWORD@$DB\_IP/ironic?charset=utf8

**[oslo\_messaging\_rabbit]部分**

rabbit\_host= $RABBIT\_HOST

**[keystone\_authtoken]部分**

auth\_uri= <http://$IDENTITY_IP:5000/>

identity\_uri= http://$IDENTITY\_IP:35357/

admin\_user= ironic

admin\_password= $IRONIC\_PASSWORD

admin\_tenant\_name= services

注意：

1. 文档中要求给IP赋值的部分还是建议附上，比如后面的my\_ip；
2. 上面的admin\_tenant\_name跟前面创建ironic用户权限一样，社区文档给出的是service，这里需要通过openstack project list确认实际环境。

#### 创建bare metal服务的数据库表

mysql -uroot -p$MARIADB\_PASSWORD -h127.0.0.1 -e "show databases;" | grep ironic

if [ $? -eq 0 ];then

ironic-dbsync --config-file $ironic\_conf upgrade

else

ironic-dbsync --config-file $ironic\_conf create\_schema

fi

这里因为是通过脚本实现的，所以防止脚本被多次修改执行，所以先判断表是否存在，如果存在，只需要upgrade，而不是create\_schema。

注意：

这里如果创建数据库失败，可以先检查一下/etc/my.cnf.d/mariadb-server.cnf中[mysqld]配置项下面有skip-name-resolve这么一句，如果没有，则添加一下，可以保证在域名解析失败的情况下同步数据库正常。

#### 重启ironic-api服务

systemctl restart openstack-ironic-api

### 配置ironic-conductor服务

如果部署的时候ironic-conductor和ironic-api在同一个节点上，下面有些配置项就不需要执行，比如数据库的connection、rabbitmq配置、keystone相关配置。

#### 修改配置文件

这里还是修改/etc/ironic/ironic.conf，配置项有：

**[DEFAULT]部分**

my\_ip=$HOST\_IP

enabled\_drivers=$MY\_DRIVERS

**[conductor]部分**

api\_url=http://$IRONIC\_API\_IP:6385

**[glance]部分**

glance\_host=$GLANCE\_IP

**[neutron]部分**

url=http://$NEUTRON\_IP:9696

**[database]部分**

connection= mysql://ironic:$IRONIC\_DBPASSWORD@$DB\_IP/ironic?charset=utf8

**[oslo\_messaging\_rabbit]部分**

rabbit\_host= $RABBIT\_HOST

**[keystone\_authtoken]部分**

auth\_uri= <http://$IDENTITY_IP:5000/>

identity\_uri= http://$IDENTITY\_IP:35357/

admin\_user= ironic

admin\_password= $IRONIC\_PASSWORD

admin\_tenant\_name= services

注意：

1. 如果conductor所在主机有多个IP，那么这里的my\_ip需要配成跟bare metal nodes所在同一网段的那个IP；
2. enabled\_drivers按照社区文档，同一个conductor host是可以支持多个driver，这里没有验证过，支持多个不同driver，是不是在这里注册成pxe\_ipmitool,pxe\_ilo这样的？

#### 重启ironic-conductor服务

systemctl restart openstack-ironic-conductor

### 配置两个服务的总结

两个服务在一个主机上时：

vi /etc/ironic/ironic.conf

目前的修改项：

[defualt]

auth\_strategy=keystone

enabled\_drivers=pxe\_ipmitool,agent\_ipmitool

my\_ip=10.43.166.65

#transport\_url=rabbit://guest:guest@10.43.166.65:5672/（出自官方N版本）

[conductor]

api\_url=http://10.43.166.65:6385

automated\_clean=true

[database]

connection=mysql://ironic:ironic@10.43.166.65/ironic?charset=utf8

[glance]

glance\_host=10.43.166.65

[keystone\_authtoken]

auth\_uri=http://10.43.166.65:5000/

identity\_uri=http://10.43.166.65:35357/

# Authentication type to load (string value)

auth\_type=v3password

# Complete public Identity API endpoint (string value)

auth\_uri=http://PUBLIC\_IDENTITY\_IP:5000/v3/

# Complete admin Identity API endpoint. (string value)

auth\_url=http://PRIVATE\_IDENTITY\_IP:35357/v3/这两项在N版本里改了，tecs3采用M版本，这里不能加这个认证

admin\_user=ironic

admin\_password=ironic

admin\_tenant\_name=services

[neutron]

url=http://10.43.166.65:9696

cleaning\_network\_uuid=f3d3a257-0f45-4b5e-bf08-6db1abd228f0

[oslo\_messaging\_rabbit]

#rabbit\_host=10.43.166.65问题：rabbitmq是在这里设还是在default选项里用transport\_url设。

设置完之后要重新更新或创建数据库，然后重启api和conductor服务。

ironic-dbsync --config-file $ironic\_conf upgrade或\

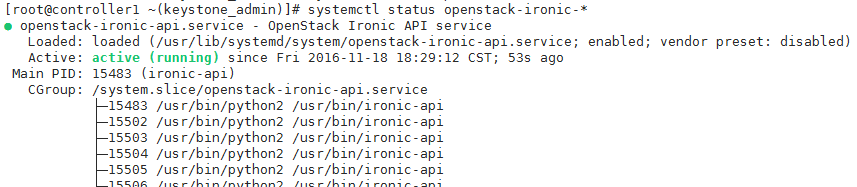
ironic-dbsync --config-file $ironic\_conf create\_schema

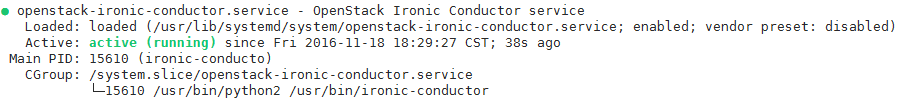
systemctl restart openstack-ironic-api

systemctl restart openstack-ironic-conductor

至此：







### mod\_wsgi

这块前面安装失败，没有深究。里面提到的/etc/apache2/ironic文件在ironic源码的etc目录下可以找到。

### 配置compute服务

#### 修改配置文件

/etc/nova/nova.conf要修改的内容如下：

**官方M版本**

**[DEFAULT]部分**

compute\_driver=ironic.IronicDriver

firewall\_driver=nova.virt.firewall.NoopFirewallDriver

scheduler\_host\_manager= nova.scheduler.ironic\_host\_manager.IronicHostManager

ram\_allocation\_ratio=1.0

reserved\_host\_memory\_mb=0

compute\_manager=ironic.nova.compute.manager.ClusteredComputeManager

scheduler\_use\_baremetal\_filters=True

scheduler\_tracks\_instance\_changes=False

**[ironic]部分**

admin\_username= ironic

admin\_password= $IRONIC\_PASSWORD

admin\_url= http://$IDENTITY\_IP:35357/v2.0

admin\_tenant\_name= services

api\_endpoint=http://$IRONIC\_NODE:6385/v1

测试环境中：

**[DEFAULT]部分**

compute\_manager=ironic.nova.compute.manager.ClusteredComputeManager（注释中显示会在之后版本忽略）

memcached\_servers=localhost:11211（解决nova500问题）

scheduler\_use\_baremetal\_filters=True

scheduler\_tracks\_instance\_changes=False

scheduler\_host\_manager=nova.scheduler.ironic\_host\_manager.IronicHostManager

compute\_driver=ironic.IronicDriver

firewall\_driver=nova.virt.firewall.NoopFirewallDriver

**[ironic]部分**

api\_endpoint=http://10.43.166.65:6385

admin\_username=ironic

admin\_password=ironic

admin\_url=http://10.43.166.65:35357

admin\_tenant\_name=services

注意：

1. 上面的admin\_tenant\_name跟前面ironic一样；
2. 文档中提到nova配置文件必须要修改compute service所在的控制节点和计算节点，这里没有仔细研究过哪些是应该在控制节点上，哪些是在计算节点上。前面验证的时候是节点合一的环境。

#### 重启服务

控制节点

systemctl restart openstack-nova-scheduler

计算节点

systemctl restart openstack-nova-compute

### 配置Neutron服务

#### 修改配置文件

这里要修改两部分文件：

/etc/neutron/plugins/ml2/ml2\_conf.ini修改的内容如下：

**官方文档**

**[ml2]部分**

type\_drivers配置项要增加对flat的支持，比如：

type\_drivers=flat,vlan

tenant\_network\_types配置项要增加对flat的支持

mechanism\_drivers =openvswitch

注意：这里文档中写的是type\_drivers=flat，但考虑neutron不会仅支持ironic，所以这里是采用增加flat，而不是仅flat一种。

**[ml2\_type\_flat]部分**

flat\_networks= \*

注意：这里文档中写的是flat\_networks=physnet1，这样就必须要求后面创建flat子网的名称必须跟这里一致，如果这里配置文件配成\*，就无这些麻烦。

**[ml2\_type\_vlan]部分**

network\_vlan\_ranges=physnet1

**[securitygroup]部分**

firewall\_driver=neutron.agent.linux.iptables\_firewall.OVSHybridIptablesFirewallDriver

enable\_security\_group=True

**[ovs]部分**

bridge\_mappings=physnet1:br-data1

注意：这里网桥名称到底是br-data1还是其他，得先看一下目前neutron里面是否有这个网桥，这里新增的br-data1网桥是用于neutron节点跟bare metal服务器通信的。

修改/etc/neutron/plugins/ml2/openvswitch\_agent.ini文件

这里是增加ovs网桥：

bridge\_mappings=physnet1:br-data1

注意：

这里文档中描述的有点不同，因为不同的环境或版本，neutron-openvswitch-agent运行的配置文件不一样，需要通过systemctl status neutron-openvswitch-agent –l看一下，一般都会列出下面这样的信息，其中可以看到其配置文件是什么。

/usr/bin/python2 /usr/bin/neutron-openvswitch-agent --config-file /usr/share/neutron/neutron-dist.conf --config-file /etc/neutron/neutron.conf --config-file /etc/neutron/plugins/ml2/openvswitch\_agent.ini --config-dir /etc/neutron/conf.d/common --config-dir /etc/neutron/conf.d/neutron-openvswitch-agent --log-file /var/log/neutron/openvswitch-agent.log

#### OVS网桥配置

ovs-vsctl add-br br-int

注意：这一步一般都可以省略，可以先执行ovs-vsctl show看一下就知道是否有br-int

ovs-vsctl add-br br-data1

ovs-vsctl add-port br-data1 eth2

注意：这里eth2要根据实际环境替换，也就是实际环境Neutron节点是通过哪个物理网口跟Bare Metal节点通信。

#### 重启服务

systemctl restart neutron-openvswitch-agent

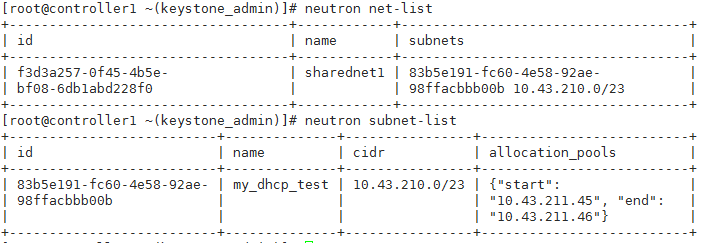
文档中介绍的是neutron-plugin-openvswitch-agent服务，这里没有安装。另外重启后可以通过ovs-vsctl show看一下是否br-data1桥，里面是否有前面物理网口eth2这个Port。

#### 创建子网

neutron net-create --tenant-id $TENANT\_ID sharednet1 --shared --provider:network\_type flat --provider:physical\_network physnet1

neutron subnet-create sharednet1 10.43.210.0/23 --name my\_dhcp\_test --ip-version=4 --gateway=10.43.210.1 --allocation-pool start=10.43.211.45,end=10.43.211.46 --enable-dhcp

注意：上面的tenant id是采用admin这个租户的id，通过openstack project list可以获取到，也可以不填；另外physnet1一定要跟前面配置文件一致，至于子网的名称、IP根据实际情况填写。



### 配置ironic支持cleaning操作

这部分具体有何作用也没有深究。需要修改/etc/ironic/ironic.conf文件。

先需要把automated\_clean配成true；

然后通过neutron net-list看一下前面创建的physnet1网络所对应的UUID，然后修改[neutron]部分的cleaning\_network\_uuid=$NETWORK\_UUID

最后需要重启ironic-conductor服务

### 总结

到此为止，ironic组件基本安装完成，如果ironic不提供物理机安装，上面的Neutron配置相关部分都不需要。

## Mitaka版本中ironic部署物理机

### 镜像制作

Ironic部署一台物理机需要两组镜像：deploy images和user images。这两组镜像都可以通过diskimage-builder工具创建。这里建议此工具采用源码安装的方式安装，方便代码修改跟踪：

git clone https://git.openstack.org/openstack/diskimage-builder

git clone <https://git.openstack.org/openstack/dib-utils>

export PATH=$PATH:$(pwd)/diskimage-builder/bin:$(pwd)/dib-utils/bin

或者就按照文档中提到的pip安装：

pip install diskimage-builder

#### deploy image

在Mitaka版本中部署镜像和物理机发现下载的镜像是同一套镜像，都是通过下面命令实现的：

disk-image-create ironic-agent centos7 -o ironic-deploy

这里面使用到了一个重要的模块：Ironic-python-agent，它既完成目标主机在部署阶段提供tgt+iscsi服务，又提供物理机发现阶段收集上报目标主机的物理信息。

成功执行后会生成两个文件：ironic-deploy.vmlinuz和ironic-deploy.initramfs，这里实际上是ironic-deploy. Kernel和ironic-deploy.initramfs，对于fedora这样的OS来说，都不会生成ironic-deploy.vmlinuz。

#### user image

对于partition images制作，采用下面命令：

disk-image-create mimosa baremetal dhcp-all-interfaces grub2 -o my-image -x &> mkimgnew.txt

成功完成后，会生成my-image.qcow2、my-image.vmlinuz和my-image.initrd三个文件。

存在的问题：上面命令的执行过程可以看出，他先从centos官网下载一个云镜像到/root/.cache/image-create目录下，如：CentOS-7-x86\_64-GenericCloud.qcow2.xz，然后下载安装一些必备包，如：cache-url、epel，最后做成上面3个文件。那么后面对于我们自己的TFG来说，如何实现？TFG项目只能提供后缀为qcow2的镜像文件，对于user image，社区还推荐了whole disk images制作，此方式完成后仅生成my-image.qcow2文件，但这种whole disk images方式如何部署，还没研究过。

**还有，上面grub2这个选项很重要，对于后面的Local boot with partition images，要求必须安装grub2。**

#### 添加镜像到glance服务

这里还是以partition images来介绍，先添加3个user images到glance服务：

glance image-create --name my-image.initrd --visibility public --disk-format ari --container-format ari < my-image.initrd

glance image-create --name my-kernel --visibility public --disk-format aki --container-format aki < my-image.vmlinuz

glance image-create --name my-image --visibility public --disk-format qcow2 --container-format bare --property kernel\_id=$(openstack image list | awk '/my-kernel/ {print $2}') --property ramdisk\_id=$(openstack image list | awk '/my-image.initrd/ {print $2}') < my-image.qcow2

然后添加2个deploy images到glance服务：

glance image-create --name deploy-vmlinuz --visibility public --disk-format aki --container-format aki < ironic-deploy.kernel

glance image-create --name deploy-initrd --visibility public --disk-format ari --container-format ari < ironic-deploy.initramfs

### 配置iscsi客户端

Ironic通过pxe\_\*部署一个物理机的12个步骤里面提到了物理主机会通过iscsi服务把硬盘暴露给ironic，然后ironic conductor就可以直接打开iscsi设备，把镜像写到目标主机硬盘上。

这里还是描述一下iscsi服务端和客户端过程，这也是在调试过程中遇到的问题。

Iscsi服务端，这里是目标主机提供，也就是前面镜像制作中提到的deploy image，会用到ironic-python-agent模块，此模块实现了iscsi服务端操作配置（在kilo版本中，deploy image是通过ramdisk-image-create centos7 deploy-ironic -o my-deploy-ramdisk-centos7这样的命令实现的，也就是直接在deploy-ironic这个element里面提供了tgtadm操作）。Iscsi服务端是靠tgtd服务来实现的，也就是靠安装scsi-target-utils-1.0.46-3.el7.x86\_64.rpm及其依赖包来提供tgtd服务的。tgtd服务起来后，一般都会通过下面3条命令来注册一个scsi设备：

tgtadm --lld iscsi --mode target --op new --tid 1 -T iqn.tmcNode1:disk1 //蓝色部分是主机名

tgtadm --lld iscsi --mode logicalunit --op new --tid 1 --lun 1 -b /dev/sda5 //蓝色部分是需要把哪个硬盘分区挂到scsi上

tgtadm --lld iscsi --mode target --op bind --tid 1 -I ALL

注意：对于ironic来说，scsi设备命令是采用NODE\_UUID加固定格式命名的，iqn.2008-10.org.openstack:fbda3e57-fdd7-4d9d-a45b-cae07b897503。这里还没研究NODE\_UUID是如何传递给目标主机的。

Iscsi客户端，这里也就是ironic conductor所在主机。客户端一般通过discovery、login后就能在/dev/disk/by-path下面看到scsi设备。客户端需要安装iscsi-initiator-utils-6.2.0.873-21.el7.x86\_64.rpm包，也就是客户端需要提供iscsid服务后才能执行iscsi相关命令：

iscsiadm -m discovery -t sendtargets -p 10.43.311.34:3260 //这里的IP是scsi服务端所在的IP

iscsiadm -m node -T iqn.tmcNode1:disk1 -p 10.43.311.34:3260 --login

此时再查询/dev/disk/by-path目录就会看到下面内容：

[root@controller43 by-path]# ls

ip-10.43.166.53:3260-iscsi-iqn.zhuxltest:disk1-lun-1 pci-0000:00:1f.2-ata-6.0 pci-0000:00:1f.2-ata-6.0-part2

ip-10.43.211.34:3260-iscsi-iqn.tmcNode1:disk1-lun-1 pci-0000:00:1f.2-ata-6.0-part1

[root@controller43 by-path]#

总结：这里谈了这么多，只是iscsid在ironic部署物理机中的作用太重要了，而对于ironic这边，其实只需每次部署物理机的时候iscsid服务是激活的就可以了。

### 创建flavor

这里由于ironic inspector流程还没跑通，所以，这里flavor创建，是提前通过其他手段读的硬件信息。

RAM\_MB= 55784

CPU=48

DISK\_GB=100

ARCH=x86\_64

nova flavor-create my-baremetal-flavor 20 $RAM\_MB $DISK\_GB $CPU

nova flavor-key my-baremetal-flavor set cpu\_arch=$ARCH

nova flavor-key my-baremetal-flavor set capabilities:boot\_option="local"

注意：上面创建的时候flavor\_id指定为20，这里是建立在你确定使用这个id作为当前bare metal的flavor id，否则就使用auto替代。还有，最后一个属性非常重要，此是保证物理裸机部署完用户镜像后重启从本地硬盘加载，否则还是走PXE，加载主机侧的用户镜像。

当然，在创建flavor之前，必须先确保quota额度够了。通过nova quota-show可以看一下，不够可以分别通过下面命令进行cores和ram的扩展（后面的ID是tenant为admin的ID）

nova quota-update --cores 48 817176c38de94b32af9e49a5bb792e3d

nova quota-update --ram 55784 817176c38de94b32af9e49a5bb792e3d

### IPMI支持

这里IPMI不需要做什么，系统会默认支持ipmitool工具，这里可以先提前验证一下功能就行：

ipmitool -I lanplus -H <ip-address> -U <username> -P <password> chassis power status

注意：替换Bare Metal Node所在的BMC的IP，用户名和密码。另外需要注意BMC的版本，我司的刀片BMC版本分重构前和重构后，里面有个set boot device命令，重构前的版本基本不支持，重构后的版本，也只有16年7月以后的版本才支持完善。

### PXE配置

先要创建几个目录，修改一下其所有者。

mkdir -p /tftpboot

mkdir -p /tftpboot/pxelinux.cfg

chown -R ironic /tftpboot

注意：这里创建pxelinux.cfg在ironic社区安装文档中并没有提到。但实际中conductor还是会先检查此目录是否会存在，否则创建Node的MAC文件失败。

然后要确保几个RPM包成功安装：

yum -y install tftp-server syslinux-tftpboot

xinetd也要安装，并且安装后需要进行enable和start操作：

rpm -ivh xinetd-2.3.15-12.el7.x86\_64.rpm

systemctl enable xinetd

systemctl start xinetd

这个社区文档中没有提到，只是说使用xinetd提供tftp服务器。

配置/etc/xinetd.d/tftp：

service tftp

{

socket\_type = dgram

protocol = udp

wait = yes

user = root

server = /usr/sbin/in.tftpd

server\_args = -v -v -v -v -v --map-file /tftpboot/map-file /tftpboot

disable = no

flags = IPv4

port = 69

}

重启xinetd服务

然后拷贝一个pxelinux.0文件到/tftpboot下面：

cp /var/lib/tftpboot/pxelinux.0 /tftpboot

最后在/tftpboot下面创建一个map文件

echo 're ^(/tftpboot/) /tftpboot/\2' > /tftpboot/map-file

echo 're ^/tftpboot/ /tftpboot/' >> /tftpboot/map-file

echo 're ^(^/) /tftpboot/\1' >> /tftpboot/map-file

echo 're ^([^/]) /tftpboot/\1' >> /tftpboot/map-file

### 正式开始部署物理机

执行下面命令：

NODE\_UUID=$(ironic node-create -d pxe\_ipmitool | awk '/^\| uuid / {print $4}')

ironic node-update $NODE\_UUID add driver\_info/ipmi\_username=zteroot driver\_info/ipmi\_password=superuser driver\_info/ipmi\_address=192.168.5.10

DEPLOY\_VMLINUZ\_UUID=$(glance image-list | awk '/deploy-vmlinuz/ {print $2}')

DEPLOY\_INITRD\_UUID=$(glance image-list | awk '/deploy-initrd/ {print $2}')

IMG=$(glance image-list | awk '/ my-image / {print $2}')

KERNEL=$(glance image-list | awk '/my-kernel/ {print $2}')

RAMDISK=$(glance image-list | awk '/ my-image.initrd / {print $2}')

NIC\_UUID=$(neutron net-list | awk '/ sharednet1 / {print $2}')

ironic node-update $NODE\_UUID add driver\_info/deploy\_kernel=$DEPLOY\_VMLINUZ\_UUID driver\_info/deploy\_ramdisk=$DEPLOY\_INITRD\_UUID

ironic node-update $NODE\_UUID add properties/cpus=48 properties/memory\_mb=55784 properties/local\_gb=100 properties/cpu\_arch=x86\_64

ironic port-create -n $NODE\_UUID -a 00:22:93:6e:da:b6

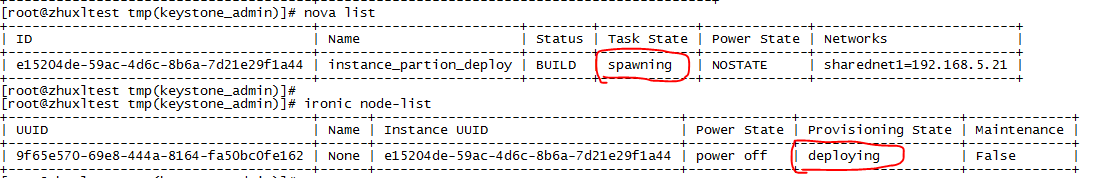
ironic node-update $NODE\_UUID add instance\_info/image\_source=$IMG instance\_info/kernel=$KERNEL instance\_info/ramdisk=$RAMDISK instance\_info/root\_gb=100

ironic node-update $NODE\_UUID add properties/capabilities="boot\_option:local"

然后执行一下ironic node-validate $NODE\_UUID 看一下结果，确保boot、deploy、management和power相关接口是True状态。注意：这里boot一定要True，也就是上面指令的最后一行，确保部署完后再次重启走本地硬盘，而不是走PXE。

最后执行：nova boot --flavor 20 --image my-image --nic net-id=$NIC\_UUID instance\_partion\_deploy

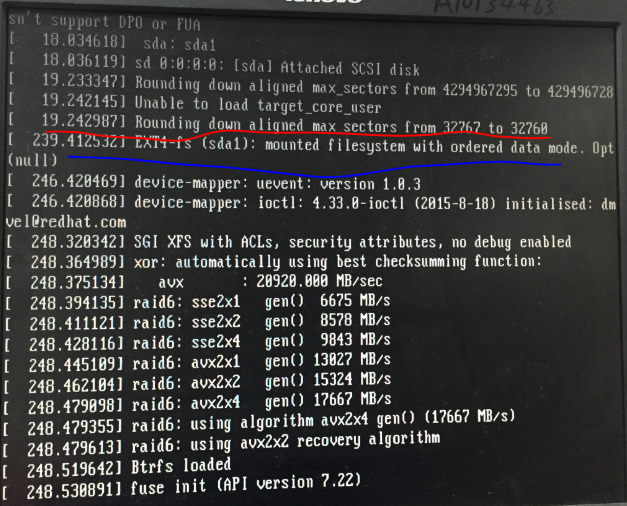
此时通过nova list确保instance进入spawning状态，通过ironic node-list确保NODE进入deploying状态，如图：



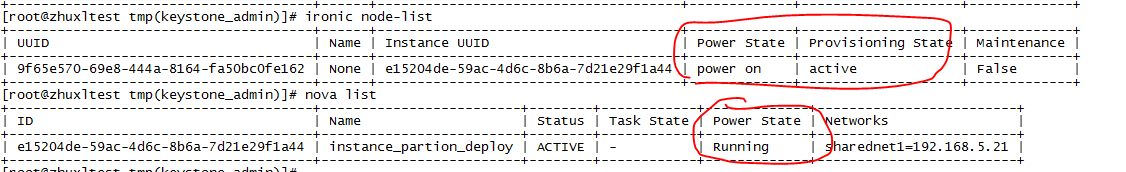
此时Node就会从deploying状态进入wait call-back，开始真正部署阶段。

注意：对于一个Node来说，必须在其provisioning state为available，也就是对于nova来说，仅部署状态为available的Node。

等物理机下载完部署镜像的两个文件，也就是deploy\_kernel和deploy\_ramdisk后会开始跑tgtd等相关服务，此时截图如下：



上面红色部分表示开始写镜像到本地硬盘，到蓝色部分看已经结束，后面会自动重启，此时从ironic和nova侧来看，已经成功部署：



此时，物理裸机会从本地启动，这里有3个前提，也就是前面着重强调的：第一、镜像制作安装社区方式，有grub2、部署镜像支持IPA；第二、flavor添加boot option为local；第三、ironic注册Node的时候要添加注册boot option为local。否则物理裸机重启后会走PXE，下载/tftpboot/ pxelinux.cfg/下面的MAC文件里面的boot镜像，也就是还是通过PXE下载用户镜像。

附件是这次调试，物理裸机被部署了用户镜像后从本地启动的过程：



### 部署中存在的问题

这里是列出前面部署过程中可能会出现的问题，只要按照上面步骤全部按序实现，则一般都不会出现下面这些问题，除了iscsi的一个问题。

#### Quota不够

这个在前面创建flavor的时候就说过，要确保quota额度满足flavor，否则在nova boot一开始就会报错，如下：

[root@zhuxltest tmp(keystone\_admin)]# nova boot --flavor 20 --image my-image --nic net-id=$NIC\_UUID instance\_partion\_deploy

ERROR (Forbidden): Quota exceeded for cores, ram: Requested 48, 55784, but already used 0, 0 of 20, 51200 cores, ram (HTTP 403) (Request-ID: req-c3991bae-e202-4810-ad3c-455bb8607605)

解决方法在前面创建flavor章节已经描述。

#### 部署失败后的处理

nova boot出错后重新部署前，一般需要做下面几件事：

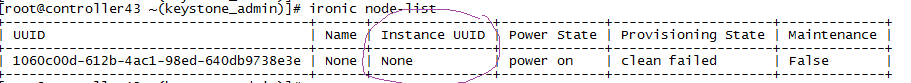
1、

nova delete $id //$id是nova boot后显示的id，也是nova list里面显示的instance id

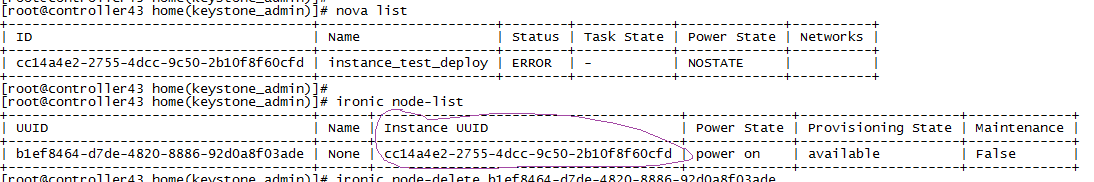
2、

ironic node-delete $id //这里的$id是ironic node-list里面的被上面nova list管理的Node ID，如果环境上ironic node-list显示就一个，好办，如果有多个，必须找到你要删除的哪一个。

另外也得看node是否已经关联上nova list里面的id，比如下面这个是已经解除了关联关系或者还没有建立关联关系：



下面这个就已经建立了关联关系

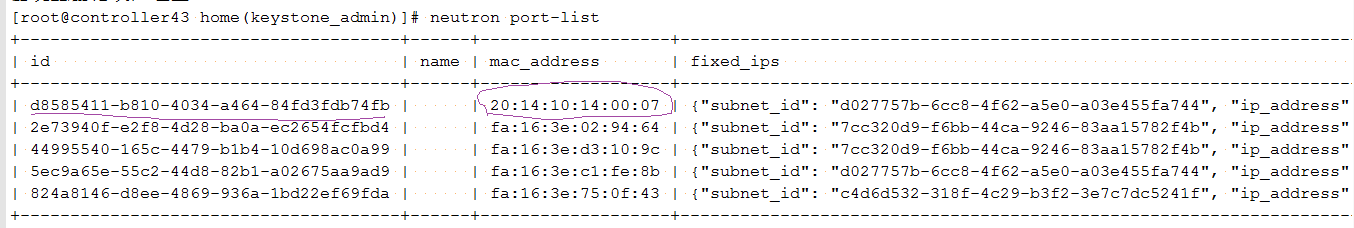


对于这种有关联关系的，要么等nova boot失败后自己解除关联关系，要么就只能手动删除，即手动进入数据库删除这一条Node。

另外ironic这边，只有当node进入稳态后才能删除。如available这样的稳态。

3、

neutron port-list ，如果nova boot已经走到了bare metal进入DHCP阶段，那么neutron port-list里面就能显示出此bare metal 的MAC跟port的对应关系，需要删除：



此时需要执行neutron port-delete $id，因为对于neutron来说，这种用户自己创建的端口，删除虚拟机的时候不会主动删除端口对应关系。

4、重启nova compute、neutron、ironic相关服务，但目前来看不知道是哪里配置的不对还是其他问题，貌似重启服务有时无法清除当前instance、node信息。建议重启。

5、每次重启后建议重点检查httpd、iscsid和xinetd服务，如果httpd服务起不来，执行ps -ef | grep dnsmasq并杀死dnsmasq所有进程

注意：如果不按照这个步骤进行部署失败后的删除处理，重新部署会莫名其妙的出现nova boot后直接进入ERROR状态，查询是过滤器没有选中的错误。比如下面这种

File "/usr/lib/python2.7/site-packages/oslo\_messaging/rpc/server.py", line 150, in inner

return func(\*args, \*\*kwargs)

File "/usr/lib/python2.7/site-packages/nova/scheduler/manager.py", line 104, in select\_destinations

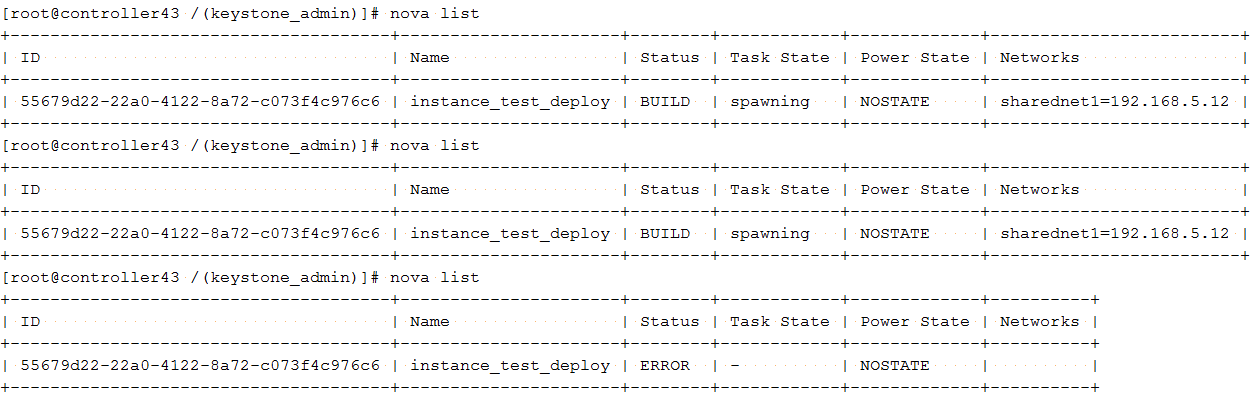
dests = self.driver.select\_destinations(ctxt, spec\_obj)

File "/usr/lib/python2.7/site-packages/nova/scheduler/filter\_scheduler.py", line 74, in select\_destinations

raise exception.NoValidHost(reason=reason)

#### Node没有进入上电流程

这里执行nova boot后开始进入spawning，但后续失败了。



此时从KVM来看，是一直停留在No Signal画面，查询ironic conductor日志，有下面错误：

[root@controller43 ironic]# tail --lines=20 ironic-conductor.log

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager task.driver.boot.prepare\_ramdisk(task, ramdisk\_opts)

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager File "/usr/lib/python2.7/site-packages/ironic/drivers/modules/pxe.py", line 484, in prepare\_ramdisk

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager deploy\_utils.try\_set\_boot\_device(task, boot\_devices.PXE)

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager File "/usr/lib/python2.7/site-packages/ironic/drivers/modules/deploy\_utils.py", line 659, in try\_set\_boot\_device

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager persistent=persistent)

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager File "/usr/lib/python2.7/site-packages/ironic/conductor/task\_manager.py", line 141, in wrapper

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager return f(\*args, \*\*kwargs)

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager File "/usr/lib/python2.7/site-packages/ironic/conductor/utils.py", line 56, in node\_set\_boot\_device

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager persistent=persistent)

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager File "/usr/lib/python2.7/site-packages/ironic/conductor/task\_manager.py", line 141, in wrapper

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager return f(\*args, \*\*kwargs)

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager File "/usr/lib/python2.7/site-packages/ironic/drivers/modules/ipmitool.py", line 889, in set\_boot\_device

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager send\_raw(task, timeout\_disable)

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager File "/usr/lib/python2.7/site-packages/ironic/conductor/task\_manager.py", line 141, in wrapper

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager return f(\*args, \*\*kwargs)

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager File "/usr/lib/python2.7/site-packages/ironic/drivers/modules/ipmitool.py", line 675, in send\_raw

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager raise exception.IPMIFailure(cmd=cmd)

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager IPMIFailure: IPMI call failed: raw 0x00 0x08 0x03 0x08.

2016-06-14 16:36:14.872 1495 ERROR ironic.conductor.manager

2016-06-14 16:36:20.084 1495 WARNING ironic.conductor.utils [req-12bff997-2194-4907-ac71-d38ff277c611 - - - - -] Not going to change node power state because current state = requested state = 'power off'.

原来是BMC不支持netfn=0x0、cmd=0x8，请求数据为0x3/0x8的命令，让BMC临时增加提供新版本后正常。

#### Neutron分配的子网和ironic的URL不在同一网段

这里主要是一般环境都喜欢把openstack装到大网IP上，所以，刚开始考虑，让neutron建立一个大网DHCP服务器，会不会影响大网，或者大网其他机器如果也配置了DHCP服务器会不会冲突，所以让neutron建立的子网是192网段：192.168.5.0/24。PXE阶段正常，能够通过DHCP下载到版本并开始运行：



但运行后进入第二阶段加载真正的大镜像的时候出错：



也就是ironic部署一个bare metal对于pxe\_\*驱动来说会有两次镜像下载过程，第一次是直接通过DHCP下载，走neutron配的小网没有问题，但第二次跟Irionic API的URL建立连接，此时如果不在同一网段就会出现上面错误。

#### Failed to erase beginning of disk

这个问题困扰了很久。一般现象是，物理主机下到了部署镜像，开始部署，但在部署过程中失败，ironic conductor日志有下面错误：

[req-7e6bb273-41fb-4193-961c-1eef5df43600 - - - - -] Initial lookup for node fbda3e57-fdd7-4d9d-a45b-cae07b897503 succeeded, agent is running and waiting for commands

2016-06-22 15:24:17.304 1480 ERROR ironic\_lib.disk\_utils [req-2aab2042-e136-4ce0-8f81-6455a8cb42af - - - - -] Failed to erase beginning of disk for node fbda3e57-fdd7-4d9d-a45b-cae07b897503. Command: sudo ironic-rootwrap /etc/ironic/rootwrap.conf dd if=/dev/zero of=/dev/disk/by-path/ip-10.43.211.46:3260-iscsi-iqn.2008-10.org.openstack:fbda3e57-fdd7-4d9d-a45b-cae07b897503-lun-1 bs=512 count=36. Error: /bin/dd: error writing '/dev/disk/by-path/ip-10.43.211.46:3260-iscsi-iqn.2008-10.org.openstack:fbda3e57-fdd7-4d9d-a45b-cae07b897503-lun-1': Input/output error

1+0 records in

0+0 records out

0 bytes (0 B) copied, 0.086388 s, 0.0 kB/s

这个问题刚开始还在社区上提了一个bug，也没有人答复，只好自己分析。上面错误打印是发生在函数destroy\_disk\_metadata（disk\_utils.py）里面，也就是开始清硬盘的时候发生的错误：

try:

utils.dd('/dev/zero', dev, 'bs=512', 'count=36')

except processutils.ProcessExecutionError as err:

with excutils.save\_and\_reraise\_exception():

LOG.error(\_LE("Failed to erase beginning of disk for node "

"%(node)s. Command: %(command)s. Error: %(error)s."),

{'node': node\_uuid,

'command': err.cmd,

'error': err.stderr})

也就是清磁盘空间，大小512\*36=18K

再对照协议跟踪代码，应该是ironic先iscsi discovery、login\_iscsi，然后才是dd写，在deploy\_utils.py里面函数deploy\_partition\_image增加打印：

LOG.info(\_LI('Zhuxuanlong test ,address=%s,'), address)

LOG.info(\_LI('Zhuxuanlong test ,port=%s,'), port)

LOG.info(\_LI('Zhuxuanlong test ,iqn=%s,'), iqn)

LOG.info(\_LI('Zhuxuanlong test ,lun=%s,'), lun)

with \_iscsi\_setup\_and\_handle\_errors(address, port, iqn, lun) as dev:

uuid\_dict\_returned = disk\_utils.work\_on\_disk(

dev, root\_mb, swap\_mb, ephemeral\_mb, ephemeral\_format, image\_path,

node\_uuid, preserve\_ephemeral=preserve\_ephemeral,

configdrive=configdrive, boot\_option=boot\_option,

boot\_mode=boot\_mode, disk\_label=disk\_label)

他是在紫色部分进行iscsi发现登录操作，然后在红色部分进行dd操作。同时放开utils.py里面的函数execute打印部分，也就是每条命令都记录结果，找到了出错的真正地方：

2016-06-23 19:03:09.493 1505 INFO ironic.drivers.modules.agent\_base\_vendor [req-ccd41119-d19f-4de9-b07b-732ab7a0e76e - - - - -] Initial lookup for node ec1eb574-34af-465b-8d37-31ce994950be succeeded, agent is running and waiting for commands

2016-06-23 19:03:10.592 1505 INFO ironic.drivers.modules.deploy\_utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Zhuxuanlong test ,address=10.43.211.46,

2016-06-23 19:03:10.600 1505 INFO ironic.drivers.modules.deploy\_utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Zhuxuanlong test ,port=3260,

2016-06-23 19:03:10.605 1505 INFO ironic.drivers.modules.deploy\_utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Zhuxuanlong test ,iqn=iqn.2008-10.org.openstack:ec1eb574-34af-465b-8d37-31ce994950be,

2016-06-23 19:03:10.609 1505 INFO ironic.drivers.modules.deploy\_utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Zhuxuanlong test ,lun=1,

2016-06-23 19:03:10.908 1505 ERROR ironic.common.utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Execution completed, command line is "iscsiadm -m discovery -t st -p 10.43.211.46:3260"

2016-06-23 19:03:10.915 1505 ERROR ironic.common.utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Command stdout is: "10.43.211.46:3260,1 iqn.2008-10.org.openstack:ec1eb574-34af-465b-8d37-31ce994950be

"

2016-06-23 19:03:10.920 1505 ERROR ironic.common.utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Command stderr is: ""

2016-06-23 19:03:11.251 1505 ERROR ironic.common.utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Execution completed, command line is "iscsiadm -m node -p 10.43.211.46:3260 -T iqn.2008-10.org.openstack:ec1eb574-34af-465b-8d37-31ce994950be --login"

2016-06-23 19:03:11.259 1505 ERROR ironic.common.utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Command stdout is: "Logging in to [iface: default, target: iqn.2008-10.org.openstack:ec1eb574-34af-465b-8d37-31ce994950be, portal: 10.43.211.46,3260] (multiple)

Login to [iface: default, target: iqn.2008-10.org.openstack:ec1eb574-34af-465b-8d37-31ce994950be, portal: 10.43.211.46,3260] successful.

"

2016-06-23 19:03:11.264 1505 ERROR ironic.common.utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Command stderr is: ""

2016-06-23 19:03:11.374 1505 ERROR ironic.common.utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Execution completed, command line is "iscsiadm -m node -S"

2016-06-23 19:03:11.375 1505 ERROR ironic.common.utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Command stdout is: "10.43.211.46:3260,1 iqn.2008-10.org.openstack:ec1eb574-34af-465b-8d37-31ce994950be

"

2016-06-23 19:03:11.375 1505 ERROR ironic.common.utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Command stderr is: ""

2016-06-23 19:03:12.060 1505 ERROR ironic.common.utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Execution completed, command line is "iscsiadm -m node -T iqn.2008-10.org.openstack:ec1eb574-34af-465b-8d37-31ce994950be -R"

2016-06-23 19:03:12.061 1505 ERROR ironic.common.utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Command stdout is: "Rescanning session [sid: 1, target: iqn.2008-10.org.openstack:ec1eb574-34af-465b-8d37-31ce994950be, portal: 10.43.211.46,3260]

"

2016-06-23 19:03:12.061 1505 ERROR ironic.common.utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Command stderr is: ""

2016-06-23 19:03:13.061 1505 ERROR ironic.drivers.modules.deploy\_utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] iSCSI connection not seen by file system. Rechecking. Attempt 1 out of 3

2016-06-23 19:03:13.415 1505 ERROR ironic\_lib.disk\_utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Failed to erase beginning of disk for node ec1eb574-34af-465b-8d37-31ce994950be. Command: sudo ironic-rootwrap /etc/ironic/rootwrap.conf dd if=/dev/zero of=/dev/disk/by-path/ip-10.43.211.46:3260-iscsi-iqn.2008-10.org.openstack:ec1eb574-34af-465b-8d37-31ce994950be-lun-1 bs=512 count=36. Error: /bin/dd: error writing '/dev/disk/by-path/ip-10.43.211.46:3260-iscsi-iqn.2008-10.org.openstack:ec1eb574-34af-465b-8d37-31ce994950be-lun-1': Input/output error

1+0 records in

0+0 records out

0 bytes (0 B) copied, 0.118621 s, 0.0 kB/s

.

2016-06-23 19:03:13.415 1505 ERROR ironic.drivers.modules.deploy\_utils [req-19f0d7c4-d017-41dc-a644-ad5639309b91 - - - - -] Deploy to address 10.43.211.46 failed.

也就是文件deploy\_utils.py里面函数check\_file\_system\_for\_iscsi\_device出错：

def check\_file\_system\_for\_iscsi\_device(portal\_address,

portal\_port,

target\_iqn):

"""Ensure the file system sees the iSCSI block device."""

check\_dir = "/dev/disk/by-path/ip-%s:%s-iscsi-%s-lun-1" % (portal\_address,

portal\_port,

target\_iqn)

total\_checks = CONF.disk\_utils.iscsi\_verify\_attempts

for attempt in range(total\_checks):

if os.path.exists(check\_dir):

break

time.sleep(1)

LOG.error("iSCSI connection not seen by file system. Rechecking. "

"Attempt %(attempt)d out of %(total)d",

{"attempt": attempt + 1,

"total": total\_checks})

Iscsi过程的调用链如下：

def login\_iscsi(portal\_address, portal\_port, target\_iqn):

"""Login to an iSCSI target."""

utils.execute('iscsiadm',

'-m', 'node',

'-p', '%s:%s' % (portal\_address, portal\_port),

'-T', target\_iqn,

'--login',

run\_as\_root=True,

check\_exit\_code=[0],

attempts=5,

delay\_on\_retry=True)

# Ensure the login complete

verify\_iscsi\_connection(target\_iqn)

# force iSCSI initiator to re-read luns

force\_iscsi\_lun\_update(target\_iqn)

# ensure file system sees the block device

check\_file\_system\_for\_iscsi\_device(portal\_address,

portal\_port,

target\_iqn)

也就是iscsi discovery后，开始login，login是先执行login命令，然后检查connection、更新lun，最后检查文件系统下是否可以看到iscsi设备，在这里失败了。

上面问题搞了好久，最终发现了两个方向：

1. 硬盘坏了，一个坏硬盘，由于部署镜像是运行在内存中，discovery、login都正常，但挂载到文件系统就无法正常了，这里需要重点检查BIOS阶段的打印，能否正常打印出本物理机上挂的硬盘信息，如果打印不出则肯定硬盘坏了；
2. 在调试中还发现了另外一个问题，函数login\_iscsi是在login后立马检查文件系统是否挂载scsi设备成功，这里检查函数check\_file\_system\_for\_iscsi\_device里面会重复3次，每次延迟1s，也就是这里在login一个scsi设备后，必须在2s就得完成scsi设备挂载。调试的时候发现不同的主机，login到文件系统挂载的时间不太一样，在CSNB这种刀片上，ironic执行完login后，2s无法完成一个scsi设备挂载，调试的时候延迟了3s就正常了，但在SBCJ这种刀片上，由于性能比较差，运行非常慢，不加延迟也能正常发现。**所以，这里建议后续团队要重点调试一下这块，也就是函数login\_iscsi，login后，理论上一般要多少时间完成一个scsi设备挂载。**

## Ironic Mitaka版本的一些新特性

Mitaka版本中还是有很多新特性没有实验过，比如agent\_\*驱动，也就是部署物理机12个步骤中既可以采用pxe\_\*驱动，也可以采用agent\_\*驱动。另外由于部署镜像制作过程中，disk-image-builder工具已经把IPA(Ironic-python-agent)作为element编译进了部署镜像里面，那么ironic conductor这侧也应该把agent\_\*驱动支持起来。

另外Ironic API version 1.10和1.11和文档中提到的Logical names也是一个非常不错的新特性，前面ironic node-create –d pxe\_ipmitool这样的方式注册一个Node节点，无法让软件从代码中把当前Node跟实际环境对应起来（这里不考虑后面会绑定BMC信息），但这里如果采用新特性，创建Node的时候直接指定名称就可以对应上了：

ironic node-create -d pxe\_ipmitool -n slot1

ironic --ironic-api-version 1.11 node-create -d agent\_ipmitool -n slot2

还有，ironic可以作为一个standalone service存在，也就是不依赖nova等openstack组件单独部署。

## Ironic inspector以及跟ironic冲突问题

Ironic hardware inspector没有正式调通过，inspector这块的文档也不够完善，这里仅把前面摸索的过程记录一下。

前面说过，ironic部署一个物理机，需要知道物理机的一些硬件信息，比如MAC、CPU、Memory、Disk，也就是这些信息需要注册到Node里面。而对于inspector来说，先只需要注册一个node节点，然后把ipmi信息注册进去，也就是BMC的IP、用户名和密码信息注册进去，然后确保inspector已经在正常运行，此时ironic通过下面两条命令更改Node的状态，可以让Node上电进入PXE模式，下载部署镜像，收集硬件信息并上报，然后自动更新Node中的MAC、硬盘、内存和CPU信息：

ironic node-set-provision-state <UUID> manage

ironic node-set-provision-state <node\_UUID> inspect

Inspector里面的安装文档部分，对数据库这块仅提了下面一句：

ironic-inspector-dbsync --config-file /etc/ironic-inspector/inspector.conf upgrade

并没有提到如何创建数据库，是否需要创建数据库，前面摸索过程中发现必须仿照ironic数据库创建一个inspector的数据库才行。

Inspector跟ironic冲突，或者说inspector后续如何运行的问题：这里目前只想到dhcp问题。在ironic部署物理机中，dhcp是通过neutron提供的，而neutron 的dhcp功能是跟MAC绑定的，也就是必须先有MAC，然后注册端口，neutron才能提供此端口的dhcp功能。而对于inspector来说，显然不行。那么对于一个既要支持inspector、又要支持ironic部署物理机来说，就得上面跑两套DHCP，比如一个dhcpd服务提供一般的DHCP服务，一个neutron的DHCP服务，inspector阶段让物理裸机通过普通的dhcpd提供的DHCP服务下载完部署镜像后上报物理信息结束后，此时需要关闭dhcpd服务，然后再配置neutron，由neutron提供dhcp服务再次下载部署镜像。

另外这里inspector社区上要求的镜像是部署镜像，对于我们来说，也可以通过自己的单独的镜像专门做上报功能，只是数据库接口就得变更。