Integration with other OpenStack services

« (install-obs.html) » (configure-cleaning.html) 🐞 (https://bugs.launchpad.net/ironic/+filebug?field.title=Integration%20with%20other%20OpenStack%20services%20in%20Installation%20Guide%20for%20Bare%20Metal%20Service&field.comment=%0A%0A-----------------------------------%0ARelease:%200.1%20on%202016-11-09%2020:35%0ASHA:%20e3bedc4eadbafbf11c1e26a216e9d40a1839a838%0ASource:%20http://git.openstack.org/cgit/openstack/ironic/tree/install-guide/source/configure-integration.rst%0AURL: http://docs.openstack.org/project-install-guide/baremetal/draft/configure-integration.html&field.tags=install-guide)

UPDATED: 2016-11-09 20:35

Contents (index.html)

Configure the Identity service for the Bare Metal service¶

1. Create the Bare Metal service user (for example, `ironic`). The service uses this to authenticate with the Identity service. Use the `service` tenant and give the user the `admin` role:

```
$ openstack user create --password IRONIC_PASSWORD \
    --email ironic@example.com ironic
$ openstack role add --project service --user ironic admin
```

2. You must register the Bare Metal service with the Identity service so that other OpenStack services can locate it. To register the service:

```
$ openstack service create --name ironic --description \
    "Ironic baremetal provisioning service" baremetal
```

3. Use the `id` property that is returned from the Identity service when registering the service (above), to create the endpoint, and replace `IRONIC_NODE` with your Bare Metal service's API node:

```
$ openstack endpoint create --region RegionOne \
    baremetal admin http://$IRONIC_NODE:6385
$ openstack endpoint create --region RegionOne \
    baremetal public http://$IRONIC_NODE:6385
$ openstack endpoint create --region RegionOne \
    baremetal internal http://$IRONIC_NODE:6385
```

If only keystone v2 API is available, use this command instead:

```
$ openstack endpoint create --region RegionOne \
    --publicurl http://$IRONIC_NODE:6385 \
    --internalurl http://$IRONIC_NODE:6385 \
    --adminurl http://$IRONIC_NODE:6385 \
    baremetal
```

4. You may delegate limited privileges related to the Bare Metal service to your Users by creating Roles with the OpenStack Identity service. By default, the Bare Metal service expects the "baremetal_admin" and "baremetal_observer" Roles to exist, in addition to the default "admin" Role. There is no negative consequence if you choose not to create these Roles. They can be created with the following commands:

```
$ openstack role create baremetal_admin
$ openstack role create baremetal_observer
```

If you choose to customize the names of Roles used with the Bare Metal service, do so by changing the "is_member", "is_observer", and "is_admin" policy settings in `/etc/ironic/policy.json`.

More complete documentation on managing Users and Roles within your OpenStack deployment are outside the scope of this document, but may be found here (http://docs.openstack.org/admin-guide/identity-concepts.html#user-management).

5. You can further restrict access to the Bare Metal service by creating a separate "baremetal" Project, so that Bare Metal resources (Nodes, Ports, etc) are only accessible to members of this Project:

```
$ openstack project create baremetal
```

At this point, you may grant read-only access to the Bare Metal service API without granting any other access by issuing the following commands:

```
$ openstack user create \
    --domain default --project-domain default --project baremetal \
    --password PASSWORD USERNAME
$ openstack role add \
    --user-domain default --project-domain default --project baremetal \
    --user USERNAME baremetal_observer
```

6. Further documentation is available elsewhere for the `openstack` command-line client (http://docs.openstack.org/admin-guide/cli-manage-projects-users-and-roles.html) and the Identity (http://docs.openstack.org/admin-guide/identity-management.html) service. A policy.json.sample (https://github.com/openstack/ironic/blob/master/etc/ironic/policy.json.sample) file, which enumerates the service's default policies, is provided for your convenience with the Bare Metal Service.

Configure Compute to use the Bare Metal service¶

The Compute service needs to be configured to use the Bare Metal service's driver. The configuration file for the Compute service is typically located at `/etc/nova/nova.conf`.

> **Note**
>
> This configuration file must be modified on the Compute service's controller nodes and compute nodes.

1. Change these configuration options in the `default` section, as follows:

```
[default]

# Driver to use for controlling virtualization. Options
# include: libvirt.LibvirtDriver, xenapi.XenAPIDriver,
# fake.FakeDriver, baremetal.BareMetalDriver,
# vmwareapi.VMwareESXDriver, vmwareapi.VMwareVCDriver (string
# value)
#compute_driver=<None>
compute_driver=ironic.IronicDriver

# Firewall driver (defaults to hypervisor specific iptables
# driver) (string value)
#firewall_driver=<None>
firewall_driver=nova.virt.firewall.NoopFirewallDriver

# The scheduler host manager class to use (string value)
#scheduler_host_manager=host_manager
scheduler_host_manager=ironic_host_manager

# Virtual ram to physical ram allocation ratio which affects
# all ram filters. This configuration specifies a global ratio
# for RamFilter. For AggregateRamFilter, it will fall back to
# this configuration value if no per-aggregate setting found.
# (floating point value)
#ram_allocation_ratio=1.5
ram_allocation_ratio=1.0

# Amount of disk in MB to reserve for the host (integer value)
#reserved_host_disk_mb=0
reserved_host_memory_mb=0

# Flag to decide whether to use baremetal_scheduler_default_filters or not.
# (boolean value)
#scheduler_use_baremetal_filters=False
scheduler_use_baremetal_filters=True

# Determines if the Scheduler tracks changes to instances to help with
# its filtering decisions (boolean value)
#scheduler_tracks_instance_changes=True
scheduler_tracks_instance_changes=False

# New instances will be scheduled on a host chosen randomly from a subset
# of the N best hosts, where N is the value set by this option.  Valid
# values are 1 or greater. Any value less than one will be treated as 1.
# For ironic, this should be set to a number >= the number of ironic nodes
# to more evenly distribute instances across the nodes.
#scheduler_host_subset_size=1
scheduler_host_subset_size=9999999
```

2. Change these configuration options in the `ironic` section. Replace:

- `IRONIC_PASSWORD` with the password you chose for the `ironic` user in the Identity Service
- `IRONIC_NODE` with the hostname or IP address of the ironic-api node
- `IDENTITY_IP` with the IP of the Identity server

```
[ironic]

# Ironic authentication type
auth_type=password

# Keystone API endpoint
auth_url=http://IDENTITY_IP:35357/v3

# Ironic keystone project name
project_name=service

# Ironic keystone admin name
username=ironic

# Ironic keystone admin password
password=IRONIC_PASSWORD

# Ironic keystone project domain
# or set project_domain_id
project_domain_name=Default

# Ironic keystone user domain
# or set user_domain_id
user_domain_name=Default
```

3. On the Compute service's controller nodes, restart the `nova-scheduler` process:

```
Fedora/RHEL7/CentOS7:
  sudo systemctl restart openstack-nova-scheduler

Ubuntu:
  sudo service nova-scheduler restart
```

4. On the Compute service's compute nodes, restart the `nova-compute` process:

```
Fedora/RHEL7/CentOS7:
  sudo systemctl restart openstack-nova-compute

Ubuntu:
  sudo service nova-compute restart
```

Create Compute flavors for use with the Bare Metal service¶

You'll need to create a special bare metal flavor in the Compute service. The flavor is mapped to the bare metal node through the hardware specifications.

1. Change these to match your hardware:

```
$ RAM_MB=1024
$ CPU=2
$ DISK_GB=100
$ ARCH={i686|x86_64}
```

2. Create the bare metal flavor by executing the following command:

```
$ nova flavor-create my-baremetal-flavor auto $RAM_MB $DISK_GB $CPU
```

> **Note**
>
> You can replace `auto` with your own flavor id.

3. Set the architecture as extra_specs information of the flavor. This will be used to match against the properties of bare metal nodes:

```
$ nova flavor-key my-baremetal-flavor set cpu_arch=$ARCH
```

4. Associate the deploy ramdisk and kernel images with the ironic node:

   ```
   $ ironic node-update $NODE_UUID add \
       driver_info/deploy_kernel=$DEPLOY_VMLINUZ_UUID \
       driver_info/deploy_ramdisk=$DEPLOY_INITRD_UUID
   ```

Configure Networking to communicate with the bare metal server¶

You need to configure Networking so that the bare metal server can communicate with the Networking service for DHCP, PXE boot and other requirements. This section covers configuring Networking for a single flat network for bare metal provisioning.

You will also need to provide Bare Metal service with the MAC address(es) of each node that it is provisioning; Bare Metal service in turn will pass this information to Networking service for DHCP and PXE boot configuration. An example of this is shown in the Enrollment (enrollment.html#enrollment) section.

1. Edit `/etc/neutron/plugins/ml2/ml2_conf.ini` and modify these:

   ```
   [ml2]
   type_drivers = flat
   tenant_network_types = flat
   mechanism_drivers = openvswitch

   [ml2_type_flat]
   flat_networks = physnet1

   [securitygroup]
   firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
   enable_security_group = True

   [ovs]
   bridge_mappings = physnet1:br-eth2
   # Replace eth2 with the interface on the neutron node which you
   # are using to connect to the bare metal server
   ```

2. If neutron-openvswitch-agent runs with `ovs_neutron_plugin.ini` as the input config-file, edit `ovs_neutron_plugin.ini` to configure the bridge mappings by adding the [ovs] section described in the previous step, and restart the neutron-openvswitch-agent.

3. Add the integration bridge to Open vSwitch:

   ```
   $ ovs-vsctl add-br br-int
   ```

4. Create the br-eth2 network bridge to handle communication between the OpenStack services (and the Bare Metal services) and the bare metal nodes using eth2. Replace eth2 with the interface on the network node which you are using to connect to the Bare Metal service:

   ```
   $ ovs-vsctl add-br br-eth2
   $ ovs-vsctl add-port br-eth2 eth2
   ```

5. Restart the Open vSwitch agent:

   ```
   # service neutron-plugin-openvswitch-agent restart
   ```

6. On restarting the Networking service Open vSwitch agent, the veth pair between the bridges br-int and br-eth2 is automatically created.

   Your Open vSwitch bridges should look something like this after following the above steps:

```
$ ovs-vsctl show

    Bridge br-int
        fail_mode: secure
        Port "int-br-eth2"
            Interface "int-br-eth2"
                type: patch
                options: {peer="phy-br-eth2"}
        Port br-int
            Interface br-int
                type: internal
    Bridge "br-eth2"
        Port "phy-br-eth2"
            Interface "phy-br-eth2"
                type: patch
                options: {peer="int-br-eth2"}
        Port "eth2"
            Interface "eth2"
        Port "br-eth2"
            Interface "br-eth2"
                type: internal
    ovs_version: "2.3.0"
```

7. Create the flat network on which you are going to launch the instances:

   ```
   $ neutron net-create --tenant-id $TENANT_ID sharednet1 --shared \
         --provider:network_type flat --provider:physical_network physnet1
   ```

8. Create the subnet on the newly created network:

   ```
   $ neutron subnet-create sharednet1 $NETWORK_CIDR --name $SUBNET_NAME \
         --ip-version=4 --gateway=$GATEWAY_IP --allocation-pool \
         start=$START_IP,end=$END_IP --enable-dhcp
   ```

Create and add images to the Image service¶

Bare Metal provisioning requires two sets of images: the deploy images and the user images. The deploy images are used by the Bare Metal service to prepare the bare metal server for actual OS deployment. Whereas the user images are installed on the bare metal server to be used by the end user. Below are the steps to create the required images and add them to the Image service:

1. The disk-image-builder (http://docs.openstack.org/developer/diskimage-builder/) can be used to create images required for deployment and the actual OS which the user is going to run.

   - Install diskimage-builder package (use virtualenv, if you don’t want to install anything globally):

     ```
     # pip install diskimage-builder
     ```

   - Build the image your users will run (Ubuntu image has been taken as an example):

     - Partition images

       ```
       $ disk-image-create ubuntu baremetal dhcp-all-interfaces grub2 -o my-image
       ```

     - Whole disk images

       ```
       $ disk-image-create ubuntu vm dhcp-all-interfaces -o my-image
       ```

   The partition image command creates `my-image.qcow2` , `my-image.vmlinuz` and `my-image.initrd` files. The `grub2` element in the partition image creation command is only needed if local boot will be used to deploy `my-image.qcow2` , otherwise the images `my-image.vmlinuz` and `my-image.initrd` will be used for PXE booting after deploying the bare metal with `my-image.qcow2` .

   If you want to use Fedora image, replace `ubuntu` with `fedora` in the chosen command.

> **Note**
>
> To build the deploy image take a look at the Building or downloading a deploy ramdisk image (deploy-ramdisk.html#deploy-ramdisk) section.

2. Add the user images to the Image service

   Load all the images created in the below steps into the Image service, and note the image UUIDs in the Image service for each one as it is generated.

   - Add the kernel and ramdisk images to the Image service:

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

Store the image uuid obtained from the above step as `MY_VMLINUZ_UUID` .

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

Store the image UUID obtained from the above step as `MY_INITRD_UUID` .

- Add the *my-image* to the Image service which is going to be the OS that the user is going to run. Also associate the above created images with this OS image. These two operations can be done by executing the following command:

```
$ glance image-create --name my-image --visibility public \
  --disk-format qcow2 --container-format bare --property \
  kernel_id=$MY_VMLINUZ_UUID --property \
  ramdisk_id=$MY_INITRD_UUID < my-image.qcow2
```

> **Note**
>
> To deploy a whole disk image, a kernel_id and a ramdisk_id shouldn't be associated with the image. For example,
>
> ```
> $ glance image-create --name my-whole-disk-image --visibility public \
>   --disk-format qcow2 \
>   --container-format bare < my-whole-disk-image.qcow2
> ```

3. Add the deploy images to the Image service

Add the *my-deploy-ramdisk.kernel* and *my-deploy-ramdisk.initramfs* images to the Image service:

```
$ glance image-create --name deploy-vmlinuz --visibility public \
  --disk-format aki --container-format aki < my-deploy-ramdisk.kernel
```

Store the image UUID obtained from the above step as `DEPLOY_VMLINUZ_UUID` .

```
$ glance image-create --name deploy-initrd --visibility public \
  --disk-format ari --container-format ari < my-deploy-ramdisk.initramfs
```

Store the image UUID obtained from the above step as `DEPLOY_INITRD_UUID` .

« (install-obs.html) » (configure-cleaning.html) (https://bugs.launchpad.net/ironic/+filebug?field.title=Integration%20with%20other%20OpenStack%20services%20in%20Installation%20Guide%20for%20Bare%20Metal%20Service&field.comment=%0A%0A-----------------------------------%0ARelease:%200.1%20on%202016-11-09%2020:35%0ASHA:%20e3bedc4eadbafbf11c1e26a216e9d40a1839a838%0ASource:%20http://git.openstack.org/cgit/openstack/ironic/tree/install-guide/source/configure-integration.rst%0AURL: http://docs.openstack.org/project-install-guide/baremetal/draft/configure-integration.html&field.tags=install-guide)

UPDATED: 2016-11-09 20:35

FOUND AN ERROR? REPORT A BUG (HTTPS://BUGS.LAUNCHPAD.NET/IRONIC/+FILEBUG?FIELD.TITLE=INTEGRATION%20WITH%20OTHER%20OPENSTACK%20SERVICES%20IN%20INSTALLATION%20GUIDE%20FOR%20BARE%20METAL%20SERVICE&FIELD.COMMENT=%0A%0A-----------------------------------%0ARELEASE:%200.1%20ON%202016-11-09%2020:35%0ASHA:%20E3BEDC4EADBAFBF11C1E26A216E9D40A1839A838%0ASOURCE:%20HTTP://GIT.OPENSTACK.ORG/CGIT/OPENSTACK/IRONIC/TREE/INSTALL-GUIDE/SOURCE/CONFIGURE-INTEGRATION.RST%0AURL: HTTP://DOCS.OPENSTACK.ORG/PROJECT-INSTALL-GUIDE/BAREMETAL/DRAFT/CONFIGURE-INTEGRATION.HTML&FIELD.TAGS=INSTALL-GUIDE)

QUESTIONS? (HTTP://ASK.OPENSTACK.ORG)

OpenStack Documentation ▾

Contents

OpenStack

- Projects (http://openstack.org/projects/)
- OpenStack Security (http://openstack.org/projects/openstack-security/)
- Common Questions (http://openstack.org/projects/openstack-faq/)
- Blog (http://openstack.org/blog/)
- News (http://openstack.org/news/)

Community

- User Groups (http://openstack.org/community/)
- Events (http://openstack.org/community/events/)
- Jobs (http://openstack.org/community/jobs/)
- Companies (http://openstack.org/foundation/companies/)
- Contribute (http://docs.openstack.org/infra/manual/developers.html)

Documentation

- OpenStack Manuals (http://docs.openstack.org)
- Getting Started (http://openstack.org/software/start/)
- API Documentation (http://developer.openstack.org)
- Wiki (https://wiki.openstack.org)

Branding & Legal

- Logos & Guidelines (http://openstack.org/brand/)
- Trademark Policy (http://openstack.org/brand/openstack-trademark-policy/)
- Privacy Policy (http://openstack.org/privacy/)
- OpenStack CLA (https://wiki.openstack.org/wiki/How_To_Contribute#Contributor_License_Agreement)

Stay In Touch

(https://twitter.com/OpenStack)(https://www.facebook.com/company/openstack)(https://www.youtube.com/user/OpenStackFoundation)