Enrollment

« (configure-tenant-networks.html) » (enabling-https.html) (https://bugs.launchpad.net/ironic/+filebug?field.title=Enrollment%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/enrollment.rst%0AURL: http://docs.openstack.org/project-install-guide/baremetal/draft/enrollment.html&field.tags=install-guide)

UPDATED: 2016-11-09 20:35

Contents (index.html)

After all the services have been properly configured, you should enroll your hardware with the Bare Metal service, and confirm that the Compute service sees the available hardware. The nodes will be visible to the Compute service once they are in the `available` provision state.

> **Note**
>
> After enrolling nodes with the Bare Metal service, the Compute service will not be immediately notified of the new resources. The Compute service's resource tracker syncs periodically, and so any changes made directly to the Bare Metal service's resources will become visible in the Compute service only after the next run of that periodic task. More information is in the Troubleshooting (troubleshooting.html#troubleshooting) section.

> **Note**
>
> Any bare metal node that is visible to the Compute service may have a workload scheduled to it, if both the `power` and `deploy` interfaces pass the `validate` check. If you wish to exclude a node from the Compute service's scheduler, for instance so that you can perform maintenance on it, you can set the node to "maintenance" mode. For more information see the Maintenance mode (troubleshooting.html#maintenance-mode) section.

Enrollment process¶

This section describes the main steps to enroll a node and make it available for provisioning. Some steps are shown separately for illustration purposes, and may be combined if desired.

1. Create a node in the Bare Metal service. At a minimum, you must specify the driver name (for example, "pxe_ipmitool"). This will return the node UUID along with other information about the node. The node's provision state will be `available`. (The example assumes that the client is using the default API version.):

```
ironic node-create -d pxe_ipmitool
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| uuid         | dfc6189f-ad83-4261-9bda-b27258eb1987 |
| driver_info  | {}                                   |
| extra        | {}                                   |
| driver       | pxe_ipmitool                         |
| chassis_uuid |                                      |
| properties   | {}                                   |
| name         | None                                 |
+--------------+--------------------------------------+

ironic node-show dfc6189f-ad83-4261-9bda-b27258eb1987
+------------------------+--------------------------------------+
| Property               | Value                                |
+------------------------+--------------------------------------+
| target_power_state     | None                                 |
| extra                  | {}                                   |
| last_error             | None                                 |
| maintenance_reason     | None                                 |
| provision_state        | available                            |
| uuid                   | dfc6189f-ad83-4261-9bda-b27258eb1987 |
| console_enabled        | False                                |
| target_provision_state | None                                 |
| provision_updated_at   | None                                 |
| maintenance            | False                                |
| power_state            | None                                 |
| driver                 | pxe_ipmitool                         |
| properties             | {}                                   |
| instance_uuid          | None                                 |
| name                   | None                                 |
| driver_info            | {}                                   |
| ...                    | ...                                  |
+------------------------+--------------------------------------+
```

Beginning with the Kilo release a node may also be referred to by a logical name as well as its UUID. To utilize this new feature a name must be assigned to the node. This can be done when the node is created by adding the `-n` option to the `node-create` command or by updating an existing node with the `node-update` command. See Logical Names for examples.

Beginning with the Liberty release, with API version 1.11 and above, a newly created node will have an initial provision state of `enroll` as opposed to `available`. See Enrolling a node for more details.

2. Update the node `driver_info` so that Bare Metal service can manage the node. Different drivers may require different information about the node. You can determine this with the `driver-properties` command, as follows:

```
ironic driver-properties pxe_ipmitool
+----------------------+-------------------------------------------------------------------------------------------------------
| Property             | Description
+----------------------+-------------------------------------------------------------------------------------------------------
| ipmi_address         | IP address or hostname of the node. Required.
| ipmi_password        | password. Optional.
| ipmi_username        | username; default is NULL user. Optional.
| ...                  | ...
| deploy_kernel        | UUID (from Glance) of the deployment kernel. Required.
| deploy_ramdisk       | UUID (from Glance) of the ramdisk that is mounted at boot time. Required.
+----------------------+-------------------------------------------------------------------------------------------------------

ironic node-update $NODE_UUID add \
driver_info/ipmi_username=$USER \
driver_info/ipmi_password=$PASS \
driver_info/ipmi_address=$ADDRESS
```

Note

If IPMI is running on a port other than 623 (the default). The port must be added to `driver_info` by specifying the `ipmi_port` value. Example:

```
ironic node-update $NODE_UUID add driver_info/ipmi_port=$PORT_NUMBER
```

Note that you may also specify all `driver_info` parameters during `node-create` by passing the **-i** option multiple times.

3. Update the node's properties to match the bare metal flavor you created earlier:

```
ironic node-update $NODE_UUID add \
properties/cpus=$CPU \
properties/memory_mb=$RAM_MB \
properties/local_gb=$DISK_GB \
properties/cpu_arch=$ARCH
```

As above, these can also be specified at node creation by passing the **-p** option to `node-create` multiple times.

4. If you wish to perform more advanced scheduling of the instances based on hardware capabilities, you may add metadata to each node that will be exposed to the nova scheduler (see: ComputeCapabilitiesFilter (http://docs.openstack.org/developer/nova/devref/filter_scheduler.html?highlight=computecapabilitiesfilter)). A full explanation of this is outside of the scope of this document. It can be done through the special `capabilities` member of node properties:

```
ironic node-update $NODE_UUID add \
properties/capabilities=key1:val1,key2:val2
```

5. As mentioned in the Create Compute flavors for use with the Bare Metal service (configure-integration.html#flavor-creation) section, if using the Kilo or later release of Bare Metal service, you should specify a deploy kernel and ramdisk which correspond to the node's driver, for example:

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

6. You must also inform Bare Metal service of the network interface cards which are part of the node by creating a port with each NIC's MAC address. These MAC addresses are passed to the Networking service during instance provisioning and used to configure the network appropriately:

```
ironic port-create -n $NODE_UUID -a $MAC_ADDRESS
```

7. To check if Bare Metal service has the minimum information necessary for a node's driver to function, you may `validate` it:

```
ironic node-validate $NODE_UUID

+------------+--------+--------+
| Interface  | Result | Reason |
+------------+--------+--------+
| console    | True   |        |
| deploy     | True   |        |
| management | True   |        |
| power      | True   |        |
+------------+--------+--------+
```

If the node fails validation, each driver will return information as to why it failed:

```
ironic node-validate $NODE_UUID

+------------+--------+-------------------------------------------------------------------------------------------------------------
| Interface  | Result | Reason
+------------+--------+-------------------------------------------------------------------------------------------------------------
| console    | None   | not supported
| deploy     | False  | Cannot validate iSCSI deploy. Some parameters were missing in node's instance_info. Missing are: ['ro
| management | False  | Missing the following IPMI credentials in node's driver_info: ['ipmi_address'].
| power      | False  | Missing the following IPMI credentials in node's driver_info: ['ipmi_address'].
+------------+--------+-------------------------------------------------------------------------------------------------------------
```

8. If using API version 1.11 or above, the node was created in the `enroll` provision state. In order for the node to be available for deploying a workload (for example, by the Compute service), it needs to be in the `available` provision state. To do this, it must be moved into the `manageable` state and then moved into the `available` state. The API version 1.11 and above section describes the commands for this.

Enrolling a node¶

In the Liberty cycle, starting with API version 1.11, the Bare Metal service added a new initial provision state of `enroll` to its state machine.

Existing automation tooling that use an API version lower than 1.11 are not affected, since the initial provision state is still `available`. However, using API version 1.11 or above may break existing automation tooling with respect to node creation.

The default API version used by (the most recent) python-ironicclient is 1.9.

The examples below set the API version for each command. To set the API version for all commands, you can set the environment variable `IRONIC_API_VERSION`.

API version 1.10 and below¶

Below is an example of creating a node with API version 1.10. After creation, the node will be in the `available` provision state. Other API versions below 1.10 may be substituted in place of 1.10.

```
ironic --ironic-api-version 1.10 node-create -d agent_ilo -n pre11

+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| uuid         | cc4998a0-f726-4927-9473-0582458c6789 |
| driver_info  | {}                                   |
| extra        | {}                                   |
| driver       | agent_ilo                            |
| chassis_uuid |                                      |
| properties   | {}                                   |
| name         | pre11                                |
+--------------+--------------------------------------+


ironic --ironic-api-version 1.10 node-list

+--------------------------------------+-------+---------------+-------------+--------------------+-------------+
| UUID                                 | Name  | Instance UUID | Power State | Provisioning State | Maintenance |
+--------------------------------------+-------+---------------+-------------+--------------------+-------------+
| cc4998a0-f726-4927-9473-0582458c6789 | pre11 | None          | None        | available          | False       |
+--------------------------------------+-------+---------------+-------------+--------------------+-------------+
```

API version 1.11 and above¶

Beginning with API version 1.11, the initial provision state for newly created nodes is `enroll`. In the examples below, other API versions above 1.11 may be substituted in place of 1.11.

```
ironic --ironic-api-version 1.11 node-create -d agent_ilo -n post11

+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| uuid         | 0eb013bb-1e4b-4f4c-94b5-2e7468242611 |
| driver_info  | {}                                   |
| extra        | {}                                   |
| driver       | agent_ilo                            |
| chassis_uuid |                                      |
| properties   | {}                                   |
| name         | post11                               |
+--------------+--------------------------------------+


ironic --ironic-api-version 1.11 node-list

+--------------------------------------+--------+---------------+-------------+--------------------+-------------+
| UUID                                 | Name   | Instance UUID | Power State | Provisioning State | Maintenance |
+--------------------------------------+--------+---------------+-------------+--------------------+-------------+
| 0eb013bb-1e4b-4f4c-94b5-2e7468242611 | post11 | None          | None        | enroll             | False       |
+--------------------------------------+--------+---------------+-------------+--------------------+-------------+
```

In order for nodes to be available for deploying workloads on them, nodes must be in the `available` provision state. To do this, nodes created with API version 1.11 and above must be moved from the `enroll` state to the `manageable` state and then to the `available` state.

To move a node to a different provision state, use the `node-set-provision-state` command.

Note

Since it is an asynchronous call, the response for `ironic node-set-provision-state` will not indicate whether the transition succeeded or not. You can check the status of the operation via `ironic node-show`. If it was successful, `provision_state` will be in the desired state. If it failed, there will be information in the node's `last_error`.

After creating a node and before moving it from its initial provision state of `enroll`, basic power and port information needs to be configured on the node. The Bare Metal service needs this information because it verifies that it is capable of controlling the node when transitioning the node from `enroll` to `manageable` state.

To move a node from `enroll` to `manageable` provision state:

```
ironic --ironic-api-version 1.11 node-set-provision-state $NODE_UUID manage

ironic node-show $NODE_UUID

+------------------------+--------------------------------------------------------------------+
| Property               | Value                                                              |
+------------------------+--------------------------------------------------------------------+
| ...                    | ...                                                                |
| provision_state        | manageable                                                         | <- verify correct state
| uuid                   | 0eb013bb-1e4b-4f4c-94b5-2e7468242611                               |
| ...                    | ...                                                                |
+------------------------+--------------------------------------------------------------------+
```

When a node is moved from the `manageable` to `available` provision state, the node will go through automated cleaning if configured to do so (see Configure the Bare Metal service for cleaning (configure-cleaning.html#configure-cleaning)). To move a node from `manageable` to `available` provision state:

```
ironic --ironic-api-version 1.11 node-set-provision-state $NODE_UUID provide

ironic node-show $NODE_UUID

+------------------------+--------------------------------------------------------------------+
| Property               | Value                                                              |
+------------------------+--------------------------------------------------------------------+
| ...                    | ...                                                                |
| provision_state        | available                                                          | < - verify correct state
| uuid                   | 0eb013bb-1e4b-4f4c-94b5-2e7468242611                               |
| ...                    | ...                                                                |
+------------------------+--------------------------------------------------------------------+
```

For more details on the Bare Metal service's state machine, see the state machine (http://docs.openstack.org/developer/ironic/dev/states.html) documentation.

Logical names¶

Beginning with the Kilo release a Node may also be referred to by a logical name as well as its UUID. Names can be assigned either when creating the node by adding the `-n` option to the `node-create` command or by updating an existing node with the `node-update` command.

Node names must be unique, and conform to:

- rfc952 (http://tools.ietf.org/html/rfc952)
- rfc1123 (http://tools.ietf.org/html/rfc1123)
- wiki_hostname (http://en.wikipedia.org/wiki/Hostname)

The node is named 'example' in the following examples:

```
ironic node-create -d agent_ipmitool -n example
```

or:

```
ironic node-update $NODE_UUID add name=example
```

Once assigned a logical name, a node can then be referred to by name or UUID interchangeably.

```
ironic node-create -d agent_ipmitool -n example

+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| uuid         | 71e01002-8662-434d-aafd-f068f69bb85e |
| driver_info  | {}                                   |
| extra        | {}                                   |
| driver       | agent_ipmitool                       |
| chassis_uuid |                                      |
| properties   | {}                                   |
| name         | example                              |
+--------------+--------------------------------------+


ironic node-show example

+------------------------+--------------------------------------+
| Property               | Value                                |
+------------------------+--------------------------------------+
| target_power_state     | None                                 |
| extra                  | {}                                   |
| last_error             | None                                 |
| updated_at             | 2015-04-24T16:23:46+00:00            |
| ...                    | ...                                  |
| instance_info          | {}                                   |
+------------------------+--------------------------------------+
```

Hardware Inspection¶

Starting with the Kilo release, Bare Metal service supports hardware inspection that simplifies enrolling nodes - please see inspection (http://docs.openstack.org/developer/ironic/deploy/inspection.html) for details.

« (configure-tenant-networks.html) » (enabling-https.html) (https://bugs.launchpad.net/ironic/+filebug?field.title=Enrollment%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/enrollment.rst%0AURL: http://docs.openstack.org/project-install-guide/baremetal/draft/enrollment.html&field.tags=install-guide)

UPDATED: 2016-11-09 20:35

(https://creativecommons.org/licenses/by/3.0/)
Except where otherwise noted, this document is licensed under Creative Commons Attribution 3.0 License (https://creativecommons.org/licenses/by/3.0/). See all OpenStack Legal Documents (http://www.openstack.org/legal).

FOUND AN ERROR? REPORT A BUG (HTTPS://BUGS.LAUNCHPAD.NET/IRONIC/+FILEBUG?FIELD.TITLE=ENROLLMENT%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/ENROLLMENT.RST%0AURL: HTTP://DOCS.OPENSTACK.ORG/PROJECT-INSTALL-GUIDE/BAREMETAL/DRAFT/ENROLLMENT.HTML&FIELD.TAGS=INSTALL-GUIDE)

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

OpenStack Documentation ▾

Contents