

Contents (index.html)

The Bare Metal service is a collection of components that provides support to manage and provision physical machines.

Also known as the `ironic` project, the Bare Metal service may, depending upon configuration, interact with several other OpenStack services. This includes:

- the OpenStack Telemetry module ( `ceilometer` ) for consuming the IPMI metrics
- the OpenStack Identity service ( `keystone` ) for request authentication and to locate other OpenStack services
- the OpenStack Image service ( `glance` ) from which to retrieve images and image meta-data
- the OpenStack Networking service ( `neutron` ) for DHCP and network configuration
- the OpenStack Compute service ( `nova` ) works with the Bare Metal service and acts as a user-facing API for instance management, while the Bare Metal service provides the admin/operator API for hardware management. The OpenStack Compute service also provides scheduling facilities (matching flavors <-> images <-> hardware), tenant quotas, IP assignment, and other services which the Bare Metal service does not, in and of itself, provide.
- the OpenStack Object Storage ( `swift` ) provides temporary storage for the configdrive, user images, deployment logs and inspection data.

The Bare Metal service includes the following components:

- ironic-api**  
A RESTful API that processes application requests by sending them to the ironic-conductor over [remote procedure call \(RPC\)](#) ([https://en.wikipedia.org/wiki/Remote\\_procedure\\_call](https://en.wikipedia.org/wiki/Remote_procedure_call)).
- ironic-conductor**  
Adds/edits/deletes nodes; powers on/off nodes with ipmi or ssh; provisions/deploys/cleans bare metal nodes.
- ironic-python-agent**  
A python service which is run in a temporary ramdisk to provide ironic-conductor and ironic-inspector services with remote access, in-band hardware control, and hardware introspection.

Additionally, the Bare Metal service has certain external dependencies, which are very similar to other OpenStack services:

- A database to store hardware information and state. You can set the database back-end type and location. A simple approach is to use the same database back end as the Compute service. Another approach is to use a separate database back-end to further isolate bare metal resources (and associated metadata) from users.
- An oslo.messaging compatible queue, such as RabbitMQ. It may use the same implementation as that of the Compute service, but that is not a requirement.

Optionally, one may wish to utilize the following associated projects for additional functionality:

- python-ironicclient** (<http://docs.openstack.org/developer/python-ironicclient/>)  
A command-line interface (CLI) and python bindings for interacting with the Bare Metal service.
- ironic-inspector** (<http://docs.openstack.org/developer/ironic-inspector/>)  
An associated service which performs in-band hardware introspection by PXE booting unregistered hardware into the ironic-python-agent ramdisk.
- diskimage-builder** (<http://docs.openstack.org/developer/diskimage-builder/>)  
A related project to help facilitate the creation of ramdisks and machine images, such as those running the ironic-python-agent.
- bifrost** (<http://docs.openstack.org/developer/bifrost/>)  
A set of Ansible playbooks that automates the task of deploying a base image onto a set of known hardware using ironic in a standalone mode.



## Contents

(index.html)

- Bare Metal service overview ()
- Install and configure the Bare Metal service (install.html)
- Integration with other OpenStack services (configure-integration.html)
- Configure the Bare Metal service for cleaning (configure-cleaning.html)
- Configure tenant networks (configure-tenant-networks.html)
- Enrollment (enrollment.html)
- Enabling HTTPS (enabling-https.html)
- Using Bare Metal service as a standalone service (standalone.html)
- Enabling the configuration drive (configdrive) (configdrive.html)
- Building or downloading a deploy ramdisk image (deploy-ramdisk.html)
- Setup the drivers for the Bare Metal service (setup-drivers.html)
- Advanced features (advanced.html)
- Troubleshooting (troubleshooting.html)
- Next steps (next-steps.html)

## 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](https://wiki.openstack.org/wiki/How_To_Contribute#Contributor_License_Agreement))

## Stay In Touch

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

The OpenStack project is provided under the [Apache 2.0 license \(http://www.apache.org/licenses/LICENSE-2.0\)](http://www.apache.org/licenses/LICENSE-2.0). Openstack.org is powered by [Rackspace Cloud Computing \(http://rackspace.com\)](http://rackspace.com).