

FISHOS OPENSTACK IRONIC

S A R D I N A

Written by: Technical Sales

Contact: support@sardinasystems.com

Tel: +442034689857

FISHOS OPENSTACK IRONIC BARE METAL SERVICE

WHAT IS IRONIC

OpenStack Ironic Bare Metal Service provisions bare metal (as opposed to virtual) machines. It may be used independently or as part of an OpenStack Cloud, and integrates with the OpenStack Identity (keystone), Compute (nova), Network (neutron), Image (glance) and Object (swift) services.

Ironic, configured with Compute and Network services, enables provisioning of physical machines via the Compute service's API.

Due to characteristics of physical servers and switch hardware, live migration cannot be performed on bare metal instances.

Ironic features a set of reference drivers that leverage open standards (eg. PXE, iPXE and IPMI) to support a wide range of hardware.

This document will focus on IPMI, which is most common. Other drivers supported in OpenStack (Yoga release) are iBMC, iDRAC, iLO, Intel IPMI, iRMC, Redfish, SNMP, and XClarity.

Bare Metal Use Cases

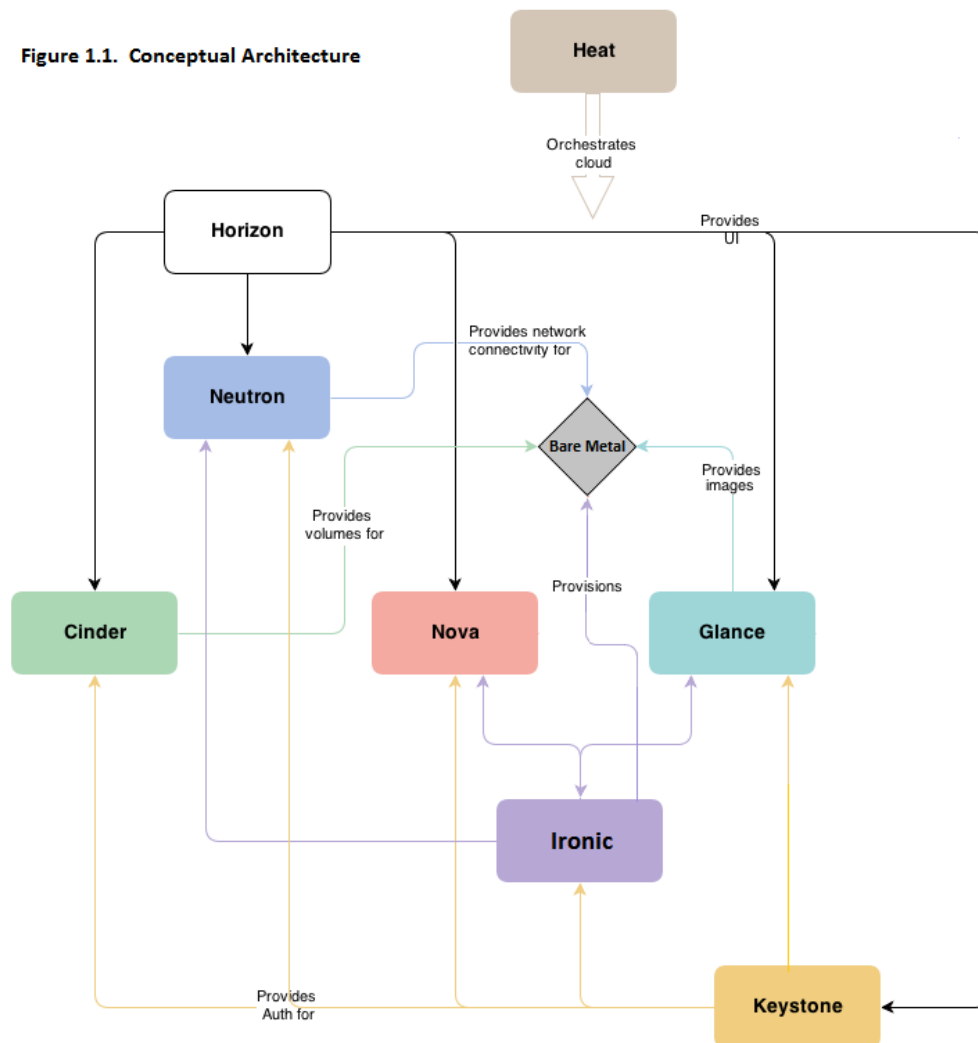
Here are a few use cases for bare metal (physical server) provisioning in cloud:

- Computing tasks that require access to hardware devices which cannot be virtualized
- High performance computing clusters
- Single tenant, dedicated hardware for performance, security, dependability and other regulatory requirements
- Or, rapidly deploying a cloud infrastructure

Conceptual Architecture

The diagram in Figure 1.1 shows the relationships between services in provisioning of a physical server by Ironic.

Figure 1.1. Conceptual Architecture



IRONIC BARE METAL SERVICE

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

Bare Metal Service Components

The Bare Metal service includes the following components:

- **ironic-api:** A RESTful API that processes application requests by sending them to ironic-conductor over remote procedure call (RPC). It can be run through WSGI or as a separate process.
- **ironic-conductor:** Adds/edits/deletes nodes; powers on/off nodes with IPMI or other vendor-specific protocol; provisions/deploys/cleans bare metal nodes. ironic-conductor uses drivers to execute operations on hardware.
- **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:

- **database:** A database to store hardware information and state. While database back-end type and location can be set, 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.
- **Oslo Messaging:** An oslo.messaging compatible queue, such as RabbitMQ. It may use the same deployment/instance as that of the Compute service, but that is not a requirement. It is used to implement RPC between ironic-api and ironic-conductor.

Deployment Architecture

The Bare Metal RESTful API service is used to enroll hardware that the Bare Metal service will manage. A cloud administrator usually registers it, specifying their attributes such as MAC addresses and IPMI credentials. There can be multiple instances of the API service.

The ironic-conductor process does the bulk of the work in an Ironic service. **For security reasons, it is advisable to place it on an isolated host, as it is the only service that requires access to both the data plane and IPMI control plane.**

There can be multiple instances of the Ironic Conductor service to support various class of drivers and also to manage fail over. Instances of the Conductor service should be on separate nodes. Each Conductor can itself run many drivers to operate heterogeneous hardware.

DEPLOYMENT CONSIDERATIONS

Some of the key considerations when deploying Ironi service are as follows:

1. The Ironi Bare Metal API service (ironi-api) should be deployed in a similar way as the control plane API services. The exact location will depend on the architecture used.
2. The Bare Metal Conductor service (ironi-conductor) is where most of the provisioning logic lives. The following considerations are the most important when deciding on the way to deploy it:
3. Ironi Conductor manages a certain proportion of nodes, distributed to it via a hash ring. This includes constantly polling these nodes for their current power state and hardware sensor data (where applicable, if enabled and supported by hardware).
4. Conductor requires access to the management controller of each node it manages.
5. Conductor co-exists with TFTP (for PXE) and/or HTTP (for iPXE) services that provide the kernel and ramdisk to boot the nodes. Conductor manages them by writing files to their root directories.
6. If serial console is used, Conductor launches console processes locally. If the nova-serialproxy service (part of the Compute service) is used, it has to be able to reach the conductors. Otherwise, they have to be directly accessible by the users.
7. There must be bi-directional connectivity between the conductor and the nodes being deployed or cleaned.
8. The provisioning ramdisk which runs the ironi-python-agent service on start up.

Note: The ironi-python-agent service is not intended to be used or executed anywhere other than a provisioning/cleaning/rescue ramdisk.

Hardware and Drivers

Power and Management Interfaces

The minimum set of capabilities that the hardware has to provide and the driver has to support are as follows:

1. getting and setting the power state of the machine
2. getting and setting the current boot device
3. booting an image provided by the Bare Metal service (in the simplest case, support booting using PXE and/or iPXE)

If hardware support for IPMI is available, the IPMI driver can be used.

Boot Interface

The boot interface of a node manages booting of both the deploy ramdisk and the user instances on the bare metal node.

The deploy interface orchestrates the deployment and defines how the image gets transferred to the target disk.

The main alternatives are to use PXE/iPXE or virtual media. If a virtual media implementation is available for the hardware, it is recommended using it for better scalability and security. Otherwise, it is recommended to use iPXE, when it is supported by target hardware.

Deploy Interface

There are several deploy interfaces supported natively within Ironic, namely, Ansible, Anaconda, direct and custom-agent interfaces. Depending on the configuration and chosen deploy interface, most tasks will be performed either on the Conductor or the baremetal node. If the Object Storage service (swift) or RadosGW is present in the environment, it is recommended to use the direct deploy method for better scalability and reliability.

Hardware Specifications

Ironic Bare Metal services does not impose too many restrictions on the characteristics of hardware itself. However, keep in mind that:

By default, the Bare Metal service will pick the smallest hard drive that is larger than 4 GiB for deployment. Another hard drive can be used, but it requires setting root device hints. (Note: This device does not have to match the boot device set in BIOS (or similar firmware)).

The machines should have enough RAM to fit the deployment/cleaning ramdisk to run.

Image Types

Ironic Bare Metal service can deploy two types of images:

1. Whole-disk images that contain a complete partitioning table with all necessary partitions and a bootloader. Such images are the most universal, but they may be harder to build.
2. Partition images that contain only the root partition. Ironic Bare Metal service will create the necessary partitions and install a boot loader, if needed. This is only supported for Linux images.

Note: If use of local boot (see below) is planned, partition images must contain GRUB2 bootloader tools to enable Ironic to set up the bootloader during deploy process.

Local vs Network Boot

Ironic Bare Metal service supports booting user instances either via a local bootloader or via the driver's boot interface (eg: via PXE or iPXE protocol in case of the PXE interface).

Network boot cannot be used with certain architectures (eg: when no tenant networks have access to the control plane).

Additional considerations are related to the pxe boot interface, and other boot interfaces based on it:

1. Local boot makes node's boot process independent of the Ironic Bare Metal Conductor instance managing it. Thus, nodes are able to reboot correctly, even if the Ironic Bare Metal TFTP or HTTP service is down.
2. Network boot (and iPXE) must be used when booting nodes from remote volumes, if the driver does not support attaching volumes in an out-of-band manner.

The default boot option for the cloud can be changed via Ironic Bare Metal service's configuration file.

Networking

There are several recommended network topologies to be used with Ironic Bare Metal service. They are explained in depth in specific architecture documentation. However, several considerations are common for all of them:

1. There has to be a provisioning network, which is used by nodes during the deployment process. If allowed by the architecture, this network should not be accessible by end users, and should not have access to the internet.
2. There has to be a cleaning network, which is used by nodes during the cleaning process.
3. There should be a rescuing network, which is used by nodes during the rescue process. It can be skipped if the rescue process is not supported.
4. The bare metal nodes must have access to the Bare Metal API while connected to the provisioning/cleaning/rescuing network.
5. If the PXE boot interface (or any boot interface based on it) is used, then the bare metal nodes should have untagged (access mode) connectivity to the provisioning/cleaning/rescuing networks. It allows PXE firmware, which does not support VLANs, to communicate with the services required for provisioning.
6. Sometimes it may be necessary to disable the spanning tree protocol delay on the switch (eg: in the event that there is a problem with reliability of DHCP during PXE/iPXE).

7. The bare metal nodes need to have access to any services required for provisioning/cleaning/rescue, while connected to the provisioning/cleaning/rescuing network. This may include:
 1. a TFTP server for PXE boot and also an HTTP server when iPXE is enabled
 2. either an HTTP server or the Object Storage service in case of the direct deploy interface and some virtual media boot interfaces
8. The Ironic Bare Metal Conductors need to have access to the booted bare metal nodes during provisioning/cleaning/rescue. Conductor communicates with an internal API, provided by ironic-python-agent, to conduct actions on nodes.

Note that the same network can be used for provisioning, cleaning and rescue purposes.

HIGH AVAILABILITY AND SCALABILITY

ironic-api

The Bare Metal API service is stateless, and thus can be easily scaled horizontally. It is recommended to deploy it as a WSGI application behind Apache or another WSGI container.

Note: This service accesses Ironic's database for reading entities (eg: in response to GET /v1/nodes request) and in rare cases for writing.

ironic-conductor

High availability

The Ironic Bare Metal Conductor service has an active/active HA model. Each Conductor instance manages a certain subset of nodes. The nodes are organized in a hash ring that tries to keep the load spread more or less uniformly across the Conductors. When a Conductor is considered offline, its nodes are taken over by other Conductors. As a result of this, at least 2 Conductor hosts will be needed for an HA deployment.

Performance

Ironic Conductors can be resource intensive, so it is recommended (but not required) to keep all Conductors separate from other services in the cloud. The minimum required number of Conductors in a deployment depends on several factors:

1. the performance of the hardware where the conductors will be running,
2. the speed and reliability of the management controller of the bare metal nodes (for example, handling slower controllers may require having less nodes per conductor),
3. the frequency, at which the management controllers are polled by the Bare Metal service (see the `sync_power_state_interval` option),
4. the bare metal driver used for nodes (see Hardware and drivers above),
5. the network performance,
6. the maximum number of bare metal nodes that are provisioned simultaneously (see the `max_concurrent_builds` option for the Compute service).

There is some tolerance in ability of Ironic Conductor to manage a larger number of bare metal nodes. However, for reliability and performance, it is recommended that each Conductor instance manage up to 100 bare metal nodes.

Disk Space

Each Conductor needs enough free disk space to cache images it uses. Depending on the combination of the deploy interface and the boot option, the space requirements may be different. The considerations are as follows:

1. The deployment kernel and ramdisk are always cached during the deployment.
2. While the direct deploy interface is used, the deployment ramdisk downloads the image and either streams it to the disk or caches it in memory.
3. When network boot is used, the instance image kernel and ramdisk are cached locally while the instance is active.

IRONIC BARE METAL DEPLOYMENT PROCESS

What happens when a boot instance request comes in? The below diagram walks through the steps involved during the provisioning of a bare metal instance.

The following prerequisites must be met before the deployment process can commence:

1. Dependent packages to be configured on the Bare Metal service node(s) where ironic-conductor is running like tftp-server, ipmi, syslinux, etc. for bare metal provisioning.
2. Nova must be configured to make use of the bare metal service endpoint and compute driver should be configured to use ironic driver on the Nova compute node(s).
3. Flavors to be created for the available hardware. Nova must know the flavor to boot from.
4. Images to be made available in Glance. Listed below are some image types required for successful bare metal deployment:
 - bm-deploy-kernel
 - bm-deploy-ramdisk
 - user-image
 - user-image-vmlinuz
 - user-image-initrd
5. Hardware to be enrolled via Ironic RESTful API service.

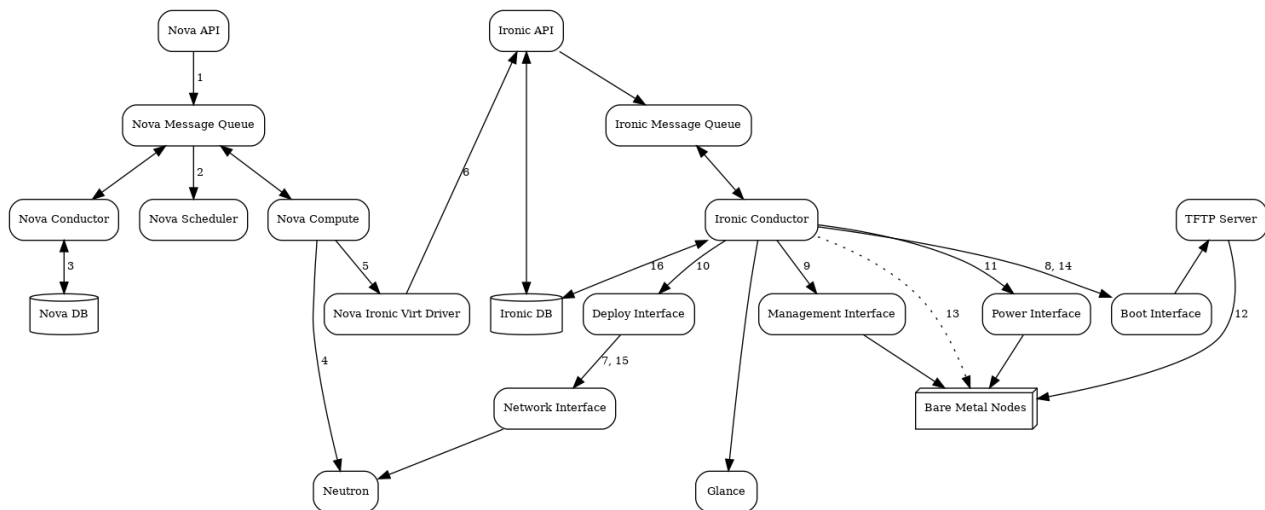
Deploy Process

A typical Ironic node deployment using PXE and the Ironic Python Agent (IPA) will following the sequence below. Depending on the Ironic driver interfaces used, some of the steps might be marginally different, however the majority of them will remain the same.

1. A request to initiate an instance is received via Nova API, and is passed through the message queue to the Nova Scheduler.
2. Nova Scheduler applies filters and finds the eligible hypervisor. Nova Scheduler also uses the flavor's extra_specs, such as cpu_arch, to match the target physical node.
3. Nova Compute Manager claims the resources of the selected hypervisor.
4. Nova Compute Manager creates (unbound) tenant virtual interfaces (VIFs) in Neutron Networking service according to the network interfaces requested in Nova's boot request. Note that the MACs of the ports will be randomly generated, and will be updated when the VIF is attached to a particular node to correspond to the node network interface card's (or bond's) MAC.

5. A spawn task is created by Nova Compute which contains all the information such as which image to boot from etc. It invokes the driver.spawn from the virt layer of Nova Compute. During the spawn process, the virt driver does the following:
 1. Updates the target Ironic node with the information about deploy image, instance UUID, requested capabilities and various flavor properties.
 2. Validates the node's power and deploy interfaces, by calling Ironic API.
 3. Attaches the previously created VIFs to the node. Each Neutron port can be attached to any Ironic port or port group, with port groups having higher priority than ports. On the Ironic side, this work is done by the network interface. Attachment here means saving the VIF identifier into Ironic port or port group and updating VIF MAC to match the port's or port group's MAC, as described above.
 4. Generates config drive, if requested.
6. Nova's Ironic virt driver issues a deploy request via the Ironic API to the Ironic Conductor servicing the bare metal node.
7. Virtual interfaces are plugged in and Neutron API updates DHCP port to set PXE/TFTP options. Where Neutron network interface is used, Ironic creates separate provisioning ports in the Neutron Networking service, while in case of flat network interface, the ports created by Nova are used both for provisioning and for the deployed instance's network.
8. The Ironic node's boot interface prepares PXE/iPXE configuration and caches deploy kernel and ramdisk.
9. The Ironic node's management interface issues commands to enable network boot of a node.
10. The Ironic node's deploy interface caches the instance image, and kernel and ramdisk if needed (it is needed in case of netboot for example).
11. The Ironic node's power interface instructs the node to power on.
12. The Ironic node boots the deploy ramdisk.
13. This step depends on the specific driver used. The deploy ramdisk downloads the image from a temporary URL (Direct deploy). The temporary URL can be generated by Swift API-compatible object stores, for example Swift itself or RadosGW. Following that, image deployment is complete.
14. The Ironic node's boot interface switches PXE configuration to refer to instance images (or, in case of local boot, sets boot device to disk), and asks the ramdisk agent to soft power off the node. If the soft power off by the ramdisk agent fails, the Ironic bare metal node would be powered off via IPMI/BMC.

15. The deploy interface triggers the network interface to remove provisioning ports if they were created, and binds the tenant ports to the node if not already bound. Then the node is powered on.
16. The ironic bare metal node's provisioning state is updated to active.



MULTI-TENANCY IN THE BARE METAL SERVICE

Overview

It is possible to use dedicated tenant networks for provisioned nodes, which extends the current Bare Metal service capabilities of providing flat networks. This works in conjunction with the Networking service to allow provisioning of nodes in a separate provisioning network. The result of this is that multiple tenants can use nodes in an isolated fashion. However, this configuration does not support trunk ports belonging to multiple networks.

Concepts

Network interfaces

Network interface is one of the driver interfaces that manages network switching for nodes. There are 3 network interfaces available in Ironic Bare Metal service:

1. noop interface is used for standalone deployments, and does not perform any network switching.
2. flat interface places all nodes into a single provider network that is pre-configured on Neutron Networking service and physical equipment. Nodes remain physically connected to this network during their entire life cycle.
3. neutron interface provides tenant-defined networking through the Neutron Networking service, separating tenant networks from each other and from the provisioning and cleaning provider networks. Nodes will move between these networks during their life cycle. This interface requires Neutron Networking service support for the switches attached to the baremetal servers so they can be programmed.

Security Overview

While Ironic Bare Metal service is intended to be a secure application, it is important to understand what it does and does not cover today.

REST API: user roles and policy settings

Beginning with the Newton (6.1.0) release, the Bare Metal service allows operators significant control over API access:

1. Access may be restricted to each method (GET, PUT, etc) for each REST resource. Defaults are provided with the release and defined in code.
2. Access may be divided between an "administrative" role with full access and "observer" role with read-only access. By default, these roles are assigned the names baremetal_admin and baremetal_observer, respectively.

3. By default, passwords and instance secrets are hidden in `driver_info` and `instance_info`, respectively. In case of debugging or diagnosing, the behavior can be overridden by changing the policy file.

Multi-tenancy

There are two aspects of multitenancy to consider when evaluating a deployment of Ironi Bare Metal Service: interactions between tenants on the network, and actions one tenant can take on a machine that will affect the next tenant.

Network Interactions

Interactions between tenants' workloads running simultaneously on separate servers may include, but are not limited to: IP spoofing, packet sniffing, and network man-in-the-middle attacks.

By default, the Bare Metal service provisions all nodes on a "flat" network, and does not take any precautions to avoid or prevent interaction between tenants. This can be addressed by integration with the OpenStack Identity, Compute, and Networking services, so as to provide tenant-network isolation.

Lingering Effects

Interactions between tenants placed sequentially on the same server include, but are not limited to: changes in BIOS settings, modifications to firmware, or files left on disk or peripheral storage devices (if these devices are not erased between uses).

By default, the Bare Metal service will erase (clean) the local disk drives during the "cleaning" phase, after deleting an instance. It does not reset BIOS or reflash firmware or peripheral devices. This can be addressed through customizing the utility ramdisk used during the "cleaning" phase.

Firmware security

When the Bare Metal service deploys an operating system image to a server, the image is run natively on the server without virtualization. Any user with administrative access to the deployed instance has administrative access to the underlying hardware.

Most servers' default settings do not prevent a privileged local user from gaining direct access to hardware devices. Such a user could modify device or firmware settings, and potentially flash new firmware to the device, before deleting their instance and allowing the server to be allocated to another user.

If the `[conductor]/automated_clean` configuration option is enabled (and the `[deploy]/erase_devices_priority` configuration option is not zero), the Bare Metal service will securely erase all local disk devices within a machine during instance deletion. However, the service does not ship with any code that will validate the integrity of, or make any modifications to, system or device firmware or firmware settings.

These risks can be mitigated by:

1. installing signed firmware for BIOS and peripheral devices
2. using a TPM (Trusted Platform Module) to validate signatures at boot time
3. booting machines in UEFI Secure Boot Support, rather than BIOS mode, to validate kernel signatures
4. disabling local (in-band) access from the host OS to the management controller (BMC)
5. disabling modifications to boot settings from the host OS

Other considerations

Internal networks

Access to networks which the Bare Metal service uses internally should be prohibited from outside. These networks are the ones used for management (with the nodes' BMC controllers), provisioning, cleaning (if used) and rescuing (if used).

This can be done with physical or logical network isolation, traffic filtering, etc.

Management interface technologies

Some nodes support more than one management interface technology (vendor and IPMI for example). If only one modern technology for out-of-band node access is used, it is recommended that you disable IPMI as the IPMI protocol is not secure. If IPMI is enabled, in most cases a local OS administrator will be able to work in-band with IPMI settings without specifying any credentials, as this is a DCMI specification requirement.

Tenant network isolation

If tenant network isolation is used, services (TFTP or HTTP) that handle the nodes' boot files should serve requests only from the internal networks that are used for the nodes being deployed and cleaned.

TFTP protocol does not support per-user access control at all.

For HTTP, there is no generic and safe way to transfer credentials to the node.

Also, tenant network isolation is not intended to work with network-booting a node by default, once the node has been provisioned.

API endpoints for RAM disk use

There are two (unauthorized) endpoints in the Bare Metal API that are intended for use by the ironic-python-agent RAM disk. They are not intended for public use.

RECOMMENDED CONFIGURATION

Configuration

Boot interface: PXE boot

Deploy interfaces: Direct, Ramdisk

Hardware driver: IPMI, iDRAC, iLO

Risks

Considerations for incompatibilities with current configuration

1. Network boot cannot be used in situations where tenant networks do not have access to the control plane.
2. Network boot (and iPXE) must be used when booting nodes from remote volumes, if the driver does not support attaching volumes out-of-band.
3. There has to be a provisioning network, used by nodes during the deployment process. If allowed by the architecture, this network should not be accessible by end users, and should not have access to public Internet.
4. There has to be a cleaning network, which is used by nodes during the cleaning process.
5. There should be a rescuing network, which is used by nodes during the rescue process. It can be skipped if the rescue process is not supported.
6. If the PXE boot interface (or a boot interface based on it) is used, then the bare metal nodes should have untagged (access mode) connectivity to the provisioning/cleaning/rescuing networks. It allows PXE firmware, which does not support VLANs, to communicate with the services required for provisioning.
7. The bare metal nodes need to have access to any services required for provisioning/cleaning/rescue, while connected to the provisioning/cleaning/rescuing network.
8. Ironi Conductor requires access to the management controller of each node it manages.
9. For security reasons, it is advisable to place Ironi Conductor on an isolated host, since it is the only service that requires access to both the data plane and IPMI control plane.
10. Tenant network isolation is not intended to work with network-booting a node by default, once the node has been provisioned.

Contact us

To discover more about these unique capabilities, please contact Sardina Sales team at sales@sardinasystems.com.