Setup the drivers for the Bare Metal service

« (deploy-ramdisk.html) » (advanced.html) (https://bugs.launchpad.net/ironic/+filebug?field.title=Setup%20the%20drivers%20for%20the%20Bare%20Metal%20service%20in%20Installation%20Guide%20for%20Bare%20Metal%20Service&field.comment=9----------%0ARelease:%200.1%20on%202016-11-09%2020:35%0ASHA:%20e3bedc4eadbafbf11c1e26a216e9d40a1839a838%0ASource:%20http://git.openstack.org/cgit/openstack/ironic/tree/install-guide/source/setup-drivers.rst%0AURL: http://docs.openstack.org/project-install-guide/baremetal/draft/setup-drivers.html&field.tags=install-guide)

UPDATED: 2016-11-09 20:35

Contents (index.html)

PXE setup
PXE UEFI setup
Elilo: an alternative to Grub2
iPXE setup
PXE Multi-Arch setup
Networking service configuration
IPMI support
Configure node web console
Boot mode support
Choosing the disk label
When used with Compute service
When used in standalone mode

PXE setup¶

If you will be using PXE, it needs to be set up on the Bare Metal service node(s) where `ironic-conductor` is running.

1. Make sure the tftp root directory exist and can be written to by the user the `ironic-conductor` is running as. For example:

```
sudo mkdir -p /tftpboot
sudo chown -R ironic /tftpboot
```

2. Install tftp server and the syslinux package with the PXE boot images:

```
Ubuntu: (Up to and including 14.04)
    sudo apt-get install xinetd tftpd-hpa syslinux-common syslinux

Ubuntu: (14.10 and after)
    sudo apt-get install xinetd tftpd-hpa syslinux-common pxelinux

Fedora 21/RHEL7/CentOS7:
    sudo yum install tftp-server syslinux-tftpboot xinetd

Fedora 22 or higher:
     sudo dnf install tftp-server syslinux-tftpboot xinetd
```

3. Using xinetd to provide a tftp server setup to serve `/tftpboot` . Create or edit `/etc/xinetd.d/tftp` as below:

```
service tftp
{
  protocol        = udp
  port            = 69
  socket_type     = dgram
  wait            = yes
  user            = root
  server          = /usr/sbin/in.tftpd
  server_args     = -v -v -v -v -v --map-file /tftpboot/map-file /tftpboot
  disable         = no
  # This is a workaround for Fedora, where TFTP will listen only on
  # IPv6 endpoint, if IPv4 flag is not used.
  flags           = IPv4
}
```

and restart xinetd service:

```
Ubuntu:
    sudo service xinetd restart

Fedora:
    sudo systemctl restart xinetd
```

4. Copy the PXE image to `/tftpboot` . The PXE image might be found at [1]:

```
Ubuntu (Up to and including 14.04):
    sudo cp /usr/lib/syslinux/pxelinux.0 /tftpboot

Ubuntu (14.10 and after):
    sudo cp /usr/lib/PXELINUX/pxelinux.0 /tftpboot
```

5. If whole disk images need to be deployed via PXE-netboot, copy the chain.c32 image to `/tftpboot` to support it. The chain.c32 image might be found at:

```
Ubuntu (Up to and including 14.04):
    sudo cp /usr/lib/syslinux/chain.c32 /tftpboot

Ubuntu (14.10 and after):
    sudo cp /usr/lib/syslinux/modules/bios/chain.c32 /tftpboot

Fedora/RHEL7/CentOS7:
    sudo cp /boot/extlinux/chain.c32 /tftpboot
```

6. If the version of syslinux is **greater than** 4 we also need to make sure that we copy the library modules into the `/tftpboot` directory [2] [1]:

```
Ubuntu:
    sudo cp /usr/lib/syslinux/modules/*/ldlinux.* /tftpboot
```

7. Create a map file in the tftp boot directory (`/tftpboot`):

```
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
```

[1] *(1, 2)* On **Fedora/RHEL** the `syslinux-tftpboot` package already install the library modules and PXE image at `/tftpboot` . If the TFTP server is configured to listen to a different directory you should copy the contents of `/tftpboot` to the configured directory

[2] http://www.syslinux.org/wiki/index.php/Library_modules (http://www.syslinux.org/wiki/index.php/Library_modules)

PXE UEFI setup¶

If you want to deploy on a UEFI supported bare metal, perform these additional steps on the ironic conductor node to configure the PXE UEFI environment.

1. Install Grub2 and shim packages:

```
Ubuntu: (14.04LTS and later)
    sudo apt-get install grub-efi-amd64-signed shim-signed

Fedora 21/RHEL7/CentOS7:
    sudo yum install grub2-efi shim

Fedora 22 or higher:
    sudo dnf install grub2-efi shim
```

2. Copy grub and shim boot loader images to `/tftpboot` directory:

```
Ubuntu: (14.04LTS and later)
    sudo cp /usr/lib/shim/shim.efi.signed /tftpboot/bootx64.efi
    sudo cp /usr/lib/grub/x86_64-efi-signed/grubnetx64.efi.signed  \
    /tftpboot/grubx64.efi

Fedora: (21 and later)
    sudo cp /boot/efi/EFI/fedora/shim.efi /tftpboot/bootx64.efi
    sudo cp /boot/efi/EFI/fedora/grubx64.efi /tftpboot/grubx64.efi

CentOS: (7 and later)
    sudo cp /boot/efi/EFI/centos/shim.efi /tftpboot/bootx64.efi
    sudo cp /boot/efi/EFI/centos/grubx64.efi /tftpboot/grubx64.efi
```

3. Create master grub.cfg:

```
Ubuntu: Create grub.cfg under ``/tftpboot/grub`` directory.
    GRUB_DIR=/tftpboot/grub

Fedora: Create grub.cfg under ``/tftpboot/EFI/fedora`` directory.
     GRUB_DIR=/tftpboot/EFI/fedora

CentOS: Create grub.cfg under ``/tftpboot/EFI/centos`` directory.
    GRUB_DIR=/tftpboot/EFI/centos

Create directory GRUB_DIR
  sudo mkdir –p $GRUB_DIR
```

This file is used to redirect grub to baremetal node specific config file. It redirects it to specific grub config file based on DHCP IP assigned to baremetal node.

```
set default=master
set timeout=5
set hidden_timeout_quiet=false

menuentry "master"  {
configfile /tftpboot/$net_default_ip.conf
}
```

Change the permission of grub.cfg:

```
sudo chmod 644 $GRUB_DIR/grub.cfg
```

4. Update the bare metal node with `boot_mode` capability in node's properties field:

```
ironic node–update <node–uuid> add properties/capabilities='boot_mode:uefi'
```

5. Make sure that bare metal node is configured to boot in UEFI boot mode and boot device is set to network/pxe.

   NOTE: `pxe_ilo` driver supports automatic setting of UEFI boot mode and boot device on the bare metal node. So this step is not required for `pxe_ilo` driver.

> **Note**
>
> For more information on configuring boot modes, see boot_mode_support.

Elilo: an alternative to Grub2¶

Elilo is a UEFI bootloader. It is an alternative to Grub2, although it isn't recommended since it is not being supported.

1. Download and untar the elilo bootloader version >= 3.16 from http://sourceforge.net/projects/elilo/ (http://sourceforge.net/projects/elilo/):

```
sudo tar zxvf elilo–3.16–all.tar.gz
```

2. Copy the elilo boot loader image to `/tftpboot` directory:

```
sudo cp ./elilo–3.16–x86_64.efi /tftpboot/elilo.efi
```

3. Update bootfile and template file configuration parameters for UEFI PXE boot in the Bare Metal Service's configuration file (/etc/ironic/ironic.conf):

```
[pxe]

# Bootfile DHCP parameter for UEFI boot mode. (string value)
uefi_pxe_bootfile_name=elilo.efi

# Template file for PXE configuration for UEFI boot loader.
# (string value)
uefi_pxe_config_template=$pybasedir/drivers/modules/elilo_efi_pxe_config.template
```

iPXE setup¶

An alternative to PXE boot, iPXE was introduced in the Juno release (2014.2.0) of Bare Metal service.

If you will be using iPXE to boot instead of PXE, iPXE needs to be set up on the Bare Metal service node(s) where `ironic–conductor` is running.

1. Make sure these directories exist and can be written to by the user the `ironic–conductor` is running as. For example:

```
sudo mkdir -p /tftpboot
sudo mkdir -p /httpboot
sudo chown -R ironic /tftpboot
sudo chown -R ironic /httpboot
```

2. Create a map file in the tftp boot directory (`/tftpboot`):

```
echo 'r ^([^/]) /tftpboot/\1' > /tftpboot/map-file
echo 'r ^(/tftpboot/) /tftpboot/\2' >> /tftpboot/map-file
```

3. Set up TFTP and HTTP servers.

   These servers should be running and configured to use the local /tftpboot and /httpboot directories respectively, as their root directories. (Setting up these servers is outside the scope of this install guide.)

   These root directories need to be mounted locally to the `ironic-conductor` services, so that the services can access them.

   The Bare Metal service's configuration file (/etc/ironic/ironic.conf) should be edited accordingly to specify the TFTP and HTTP root directories and server addresses. For example:

```
[pxe]

# Ironic compute node's tftp root path. (string value)
tftp_root=/tftpboot

# IP address of Ironic compute node's tftp server. (string
# value)
tftp_server=192.168.0.2

[deploy]
# Ironic compute node's http root path. (string value)
http_root=/httpboot

# Ironic compute node's HTTP server URL. Example:
# http://192.1.2.3:8080 (string value)
http_url=http://192.168.0.2:8080
```

4. Install the iPXE package with the boot images:

```
Ubuntu:
    apt-get install ipxe

Fedora 21/RHEL7/CentOS7:
    yum install ipxe-bootimgs

Fedora 22 or higher:
    dnf install ipxe-bootimgs
```

5. Copy the iPXE boot image (`undionly.kpxe` for **BIOS** and `ipxe.efi` for **UEFI**) to `/tftpboot` . The binary might be found at:

```
Ubuntu:
    cp /usr/lib/ipxe/{undionly.kpxe,ipxe.efi} /tftpboot

Fedora/RHEL7/CentOS7:
    cp /usr/share/ipxe/{undionly.kpxe,ipxe.efi} /tftpboot
```

> **Note**
>
> If the packaged version of the iPXE boot image doesn't work, you can download a prebuilt one from http://boot.ipxe.org (http://boot.ipxe.org) or build one image from source, see http://ipxe.org/download (http://ipxe.org/download) for more information.

6. Enable/Configure iPXE in the Bare Metal Service's configuration file (/etc/ironic/ironic.conf):

```
[pxe]

# Enable iPXE boot. (boolean value)
ipxe_enabled=True

# Neutron bootfile DHCP parameter. (string value)
pxe_bootfile_name=undionly.kpxe

# Bootfile DHCP parameter for UEFI boot mode. (string value)
uefi_pxe_bootfile_name=ipxe.efi

# Template file for PXE configuration. (string value)
pxe_config_template=$pybasedir/drivers/modules/ipxe_config.template

# Template file for PXE configuration for UEFI boot loader.
# (string value)
uefi_pxe_config_template=$pybasedir/drivers/modules/ipxe_config.template
```

7. It is possible to configure the Bare Metal service in such a way that nodes will boot into the deploy image directly from Object Storage. Doing this avoids having to cache the images on the ironic-conductor host and serving them via the ironic-conductor's HTTP server. This can be done if:

   a. the Image Service is used for image storage;
   b. the images in the Image Service are internally stored in Object Storage;
   c. the Object Storage supports generating temporary URLs for accessing objects stored in it. Both the OpenStack Swift and RADOS Gateway provide support for this.
      - See Ceph Object Gateway support (http://docs.openstack.org/developer/ironic/deploy/radosgw.html) on how to configure the Bare Metal Service with RADOS Gateway as the Object Storage.

   Configure this by setting the `[pxe]/ipxe_use_swift` configuration option to `True` as follows:

```
[pxe]

# Download deploy images directly from swift using temporary
# URLs. If set to false (default), images are downloaded to
# the ironic-conductor node and served over its local HTTP
# server. Applicable only when 'ipxe_enabled' option is set to
# true. (boolean value)
ipxe_use_swift=True
```

   Although the HTTP server still has to be deployed and configured (as it will serve iPXE boot script and boot configuration files for nodes), such configuration will shift some load from ironic-conductor hosts to the Object Storage service which can be scaled horizontally.

   Note that when SSL is enabled on the Object Storage service you have to ensure that iPXE firmware on the nodes can indeed boot from generated temporary URLs that use HTTPS protocol.

8. Restart the `ironic-conductor` process:

```
Fedora/RHEL7/CentOS7:
  sudo systemctl restart openstack-ironic-conductor

Ubuntu:
  sudo service ironic-conductor restart
```

PXE Multi-Arch setup¶

It is possible to deploy servers of different architecture by one conductor.

To support this feature, architecture specific boot and template files must be configured correctly in the options listed below:

- `pxe_bootfile_name_by_arch`
- `pxe_config_template_by_arch`

These two options are dictionary values. Node's `cpu_arch` property is used as the key to find according boot file and template. If according `cpu_arch` is not found in the dictionary, `pxe_bootfile_name`, `pxe_config_template`, `uefi_pxe_bootfile_name` and `uefi_pxe_config_template` are referenced as usual.

In below example, x86 and x86_64 nodes will be deployed by bootf1 or bootf2 based on `boot_mode` capability('bios' or 'uefi') as there's no 'x86' or 'x86_64' keys available in `pxe_bootfile_name_by_arch`. While aarch64 nodes will be deployed by bootf3, and ppc64 nodes by bootf4:

```
pxe_bootfile_name = bootf1
uefi_pxe_bootfile_name = bootf2
pxe_bootfile_name_by_arch = aarch64:bootf3,ppc64:bootf4
```

Following example assumes you are provisioning x86_64 and aarch64 servers, both in UEFI boot mode.

Update bootfile and template file configuration parameters in the Bare Metal Service's configuration file (/etc/ironic/ironic.conf):

```
[pxe]

# Bootfile DHCP parameter for UEFI boot mode. (string value)
uefi_pxe_bootfile_name=bootx64.efi

# Template file for PXE configuration for UEFI boot loader.
# (string value)
uefi_pxe_config_template=$pybasedir/drivers/modules/pxe_grub_config.template

# Bootfile DHCP parameter per node architecture. (dictionary value)
pxe_bootfile_name_by_arch=aarch64:grubaa64.efi

# Template file for PXE configuration per node architecture.
# (dictionary value)
pxe_config_template_by_arch=aarch64:pxe_grubaa64_config.template
```

Networking service configuration¶

DHCP requests from iPXE need to have a DHCP tag called `ipxe` , in order for the DHCP server to tell the client to get the boot.ipxe script via HTTP. Otherwise, if the tag isn't there, the DHCP server will tell the DHCP client to chainload the iPXE image (undionly.kpxe). The Networking service needs to be configured to create this DHCP tag, since it isn't created by default.

1. Create a custom `dnsmasq.conf` file with a setting for the ipxe tag. For example, create the file `/etc/dnsmasq-ironic.conf` with the content:

   ```
   # Create the "ipxe" tag if request comes from iPXE user class
   dhcp-userclass=set:ipxe,iPXE

   # Alternatively, create the "ipxe" tag if request comes from DHCP option 175
   # dhcp-match=set:ipxe,175
   ```

2. In the Networking service DHCP Agent configuration file (typically located at /etc/neutron/dhcp_agent.ini), set the custom `/etc/dnsmasq-ironic.conf` file as the dnsmasq configuration file:

   ```
   [DEFAULT]
   dnsmasq_config_file = /etc/dnsmasq-ironic.conf
   ```

3. Restart the `neutron-dhcp-agent` process:

   ```
   service neutron-dhcp-agent restart
   ```

IPMI support¶

If using the IPMITool driver, the `ipmitool` command must be present on the service node(s) where `ironic-conductor` is running. On most distros, this is provided as part of the `ipmitool` package. Source code is available at http://ipmitool.sourceforge.net/ (http://ipmitool.sourceforge.net/)

Note that certain distros, notably Mac OS X and SLES, install `openipmi` instead of `ipmitool` by default. THIS DRIVER IS NOT COMPATIBLE WITH `openipmi` AS IT RELIES ON ERROR HANDLING OPTIONS NOT PROVIDED BY THIS TOOL.

Check that you can connect to and authenticate with the IPMI controller in your bare metal server by using `ipmitool` :

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

<ip-address> = The IP of the IPMI controller you want to access

Note:

1. This is not the bare metal node's main IP. The IPMI controller should have its own unique IP.
2. In case the above command doesn't return the power status of the bare metal server, check for these:
   - `ipmitool` is installed.
   - The IPMI controller on your bare metal server is turned on.
   - The IPMI controller credentials passed in the command are right.
   - The conductor node has a route to the IPMI controller. This can be checked by just pinging the IPMI controller IP from the conductor node.

> **Note**
>
> If there are slow or unresponsive BMCs in the environment, the retry_timeout configuration option in the [ipmi] section may need to be lowered. The default is fairly conservative, as setting this timeout too low can cause older BMCs to crash and require a hard-reset.

Bare Metal service supports sending IPMI sensor data to Telemetry with pxe_ipmitool, pxe_ipminative, agent_ipmitool, agent_pyghmi, agent_ilo, iscsi_ilo, pxe_ilo, and with pxe_irmc driver starting from Kilo release. By default, support for sending IPMI sensor data to Telemetry is disabled. If you want to enable it, you should make the following two changes in `ironic.conf` :

- `notification_driver = messaging` in the `DEFAULT` section
- `send_sensor_data = true` in the `conductor` section

If you want to customize the sensor types which will be sent to Telemetry, change the `send_sensor_data_types` option. For example, the below settings will send temperature, fan, voltage and these three sensor types of data to Telemetry:

- send_sensor_data_types=Temperature,Fan,Voltage

If we use default value 'All' for all the sensor types which are supported by Telemetry, they are:

- Temperature, Fan, Voltage, Current

Configure node web console¶

See Configuring Web or Serial Console (http://docs.openstack.org/developer/ironic/deploy/console.html).

Boot mode support¶

The following drivers support setting of boot mode (Legacy BIOS or UEFI).

- `pxe_ipmitool`

The boot modes can be configured in Bare Metal service in the following way:

- When no boot mode setting is provided, these drivers default the boot_mode to Legacy BIOS.
- Only one boot mode (either `uefi` or `bios`) can be configured for the node.
- If the operator wants a node to boot always in `uefi` mode or `bios` mode, then they may use `capabilities` parameter within `properties` field of an bare metal node. The operator must manually set the appropriate boot mode on the bare metal node.

  To configure a node in `uefi` mode, then set `capabilities` as below:

  ```
  ironic node-update <node-uuid> add properties/capabilities='boot_mode:uefi'
  ```

  Nodes having `boot_mode` set to `uefi` may be requested by adding an `extra_spec` to the Compute service flavor:

  ```
  nova flavor-key ironic-test-3 set capabilities:boot_mode="uefi"
  nova boot --flavor ironic-test-3 --image test-image instance-1
  ```

  If `capabilities` is used in `extra_spec` as above, nova scheduler (`ComputeCapabilitiesFilter`) will match only bare metal nodes which have the `boot_mode` set appropriately in `properties/capabilities`. It will filter out rest of the nodes.

  The above facility for matching in the Compute service can be used in heterogeneous environments where there is a mix of `uefi` and `bios` machines, and operator wants to provide a choice to the user regarding boot modes. If the flavor doesn't contain `boot_mode` and `boot_mode` is configured for bare metal nodes, then nova scheduler will consider all nodes and user may get either `bios` or `uefi` machine.

Choosing the disk label¶

> **Note**
>
> The term `disk label` is historically used in Ironic and was taken from parted (https://www.gnu.org/software/parted). Apparently everyone seems to have a different word for `disk label` - these are all the same thing: disk type, partition table, partition map and so on...

Ironic allows operators to choose which disk label they want their bare metal node to be deployed with when Ironic is responsible for partitioning the disk; therefore choosing the disk label does not apply when the image being deployed is a `whole disk image`.

There are some edge cases where someone may want to choose a specific disk label for the images being deployed, including but not limited to:

- For machines in `bios` boot mode with disks larger than 2 terabytes it's recommended to use a `gpt` disk label. That's because a capacity beyond 2 terabytes is not addressable by using the MBR partitioning type. But, although GPT claims to be backward compatible with legacy BIOS systems that's not always the case (http://www.rodsbooks.com/gdisk/bios.html).
- Operators may want to force the partitioning to be always MBR (even if the machine is deployed with boot mode `uefi`) to avoid breakage of applications and tools running on those instances.

The disk label can be configured in two ways; when Ironic is used with the Compute service or in standalone mode. The following bullet points and sections will describe both methods:

- When no disk label is provided Ironic will configure it according to the boot mode; `bios` boot mode will use `msdos` and `uefi` boot mode will use `gpt`.
- Only one disk label - either `msdos` or `gpt` - can be configured for the node.

When used with Compute service¶

When Ironic is used with the Compute service the disk label should be set to node's `properties/capabilities` field and also to the flavor which will request such capability, for example:

```
ironic node-update <node-uuid> add properties/capabilities='disk_label:gpt'
```

As for the flavor:

```
nova flavor-key baremetal set capabilities:disk_label="gpt"
```

When used in standalone mode¶

When used without the Compute service, the disk label should be set directly to the node's `instance_info` field, as below:

```
ironic node-update <node-uuid> add instance_info/capabilities='{"disk_label": "gpt"}'
```

« (deploy-ramdisk.html) » (advanced.html) (https://bugs.launchpad.net/ironic/+filebug?field.title=Setup%20the%20drivers%20for%20the%20Bare%20Metal%20service%20in%20Installation%20Guide%20for%20Bare%20Metal%20Service&field.comment----------%0ARelease:%200.1%20on%202016-11-09%2020:35%0ASHA:%20e3bedc4eadbafbf11c1e26a216e9d40a1839a838%0ASource:%20http://git.openstack.org/cgit/openstack/ironic/tree/install-guide/source/setup-drivers.rst%0AURL: http://docs.openstack.org/project-install-guide/baremetal/draft/setup-drivers.html&field.tags=install-guide)

UPDATED: 2016-11-09 20:35

OpenStack Documentation

Contents

(index.html)

- Bare Metal service overview (get_started.html)
- 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 ()
  - PXE setup
  - PXE UEFI setup
  - iPXE setup
  - PXE Multi-Arch setup
  - Networking service configuration
  - IPMI support
  - Configure node web console
  - Boot mode support
  - Choosing the disk label
- 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/)