Advanced features

« (setup-drivers.html) » (troubleshooting.html) (https://bugs.launchpad.net/ironic/+filebug?field.title=Advanced%20features%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/advanced.rst%0AURL: http://docs.openstack.org/project-install-guide/baremetal/draft/advanced.html&field.tags=install-guide)

UPDATED: 2016-11-09 20:35

Contents (index.html)

Local boot with partition images¶

Starting with the Kilo release, Bare Metal service supports local boot with partition images, meaning that after the deployment the node's subsequent reboots won't happen via PXE or Virtual Media. Instead, it will boot from a local boot loader installed on the disk.

It's important to note that in order for this to work the image being deployed with Bare Metal service **must** contain `grub2` installed within it.

Enabling the local boot is different when Bare Metal service is used with Compute service and without it. The following sections will describe both methods.

> **Note**
>
> The local boot feature is dependent upon a updated deploy ramdisk built with diskimage-builder (http://docs.openstack.org/developer/diskimage-builder/) **version >= 0.1.42** or ironic-python-agent (http://docs.openstack.org/developer/ironic-python-agent/) in the kilo-era.

Enabling local boot with Compute service¶

To enable local boot we need to set a capability on the bare metal node, for example:

```
ironic node-update <node-uuid> add properties/capabilities="boot_option:local"
```

Nodes having `boot_option` set to `local` may be requested by adding an `extra_spec` to the Compute service flavor, for example:

```
nova flavor-key baremetal set capabilities:boot_option="local"
```

> **Note**
>
> If the node is configured to use `UEFI`, Bare Metal service will create an `EFI partition` on the disk and switch the partition table format to `gpt`. The `EFI partition` will be used later by the boot loader (which is installed from the deploy ramdisk).

Enabling local boot without Compute¶

Since adding `capabilities` to the node's properties is only used by the nova scheduler to perform more advanced scheduling of instances, we need a way to enable local boot when Compute is not present. To do that we can simply specify the capability via the `instance_info` attribute of the node, for example:

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

Specifying the disk for deployment (root device hints)¶

Starting with the Kilo release, Bare Metal service supports passing hints to the deploy ramdisk about which disk it should pick for the deployment. The list of support hints is:

- model (STRING): device identifier
- vendor (STRING): device vendor
- serial (STRING): disk serial number
- size (INT): size of the device in GiB

> **Note**
>
> A node's 'local_gb' property is often set to a value 1 GiB less than the actual disk size to account for partitioning (this is how DevStack, TripleO and Ironic Inspector work, to name a few). However, in this case `size` should be the actual size. For example, for a 128 GiB disk `local_gb` will be 127, but size hint will be 128.

- wwn (STRING): unique storage identifier
- wwn_with_extension (STRING): unique storage identifier with the vendor extension appended
- wwn_vendor_extension (STRING): unique vendor storage identifier
- rotational (BOOLEAN): whether it's a rotational device or not. This hint makes it easier to distinguish HDDs (rotational) and SSDs (not rotational) when choosing which disk Ironic should deploy the image onto.
- name (STRING): the device name, e.g /dev/md0

> **Warning**
>
> The root device hint name should only be used for devices with constant names (e.g RAID volumes). For SATA, SCSI and IDE disk controllers this hint is not recommended because the order in which the device nodes are added in Linux is arbitrary, resulting in devices like /dev/sda and /dev/sdb switching around at boot time (https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Storage_Administration_Guide/persistent_naming.html).

To associate one or more hints with a node, update the node's properties with a `root_device` key, for example:

```
ironic node-update <node-uuid> add properties/root_device='{"wwn": "0x4000cca77fc4dba1"}'
```

That will guarantee that Bare Metal service will pick the disk device that has the `wwn` equal to the specified wwn value, or fail the deployment if it can not be found.

> **Note**
>
> If multiple hints are specified, a device must satisfy all the hints.

Appending kernel parameters to boot instances¶

The Bare Metal service supports passing custom kernel parameters to boot instances to fit users' requirements. The way to append the kernel parameters is depending on how to boot instances.

Network boot¶

Currently, the Bare Metal service supports assigning unified kernel parameters to PXE booted instances by:

- Modifying the `[pxe]/pxe_append_params` configuration option, for example:

  ```
  [pxe]

  pxe_append_params = quiet splash
  ```

- Copying a template from shipped templates to another place, for example:

  ```
  https://git.openstack.org/cgit/openstack/ironic/tree/ironic/drivers/modules/pxe_config.template
  ```

  Making the modifications and pointing to the custom template via the configuration options: `[pxe]/pxe_config_template` and `[pxe]/uefi_pxe_config_template`.

Local boot¶

For local boot instances, users can make use of configuration drive (see Enabling the configuration drive (configdrive) (configdrive.html#configdrive)) to pass a custom script to append kernel parameters when creating an instance. This is more flexible and can vary per instance. Here is an example for grub2 with ubuntu, users can customize it to fit their use case:

```
#!/usr/bin/env python
import os

# Default grub2 config file in Ubuntu
grub_file = '/etc/default/grub'
# Add parameters here to pass to instance.
kernel_parameters = ['quiet', 'splash']
grub_cmd = 'GRUB_CMDLINE_LINUX'
old_grub_file = grub_file+'~'
os.rename(grub_file, old_grub_file)
cmdline_existed = False
with open(grub_file, 'w') as writer, \
       open(old_grub_file, 'r') as reader:
       for line in reader:
           key = line.split('=')[0]
           if key == grub_cmd:
               #If there is already some value:
               if line.strip()[-1] == '"':
                   line = line.strip()[:-1] + ' ' + ' '.join(kernel_parameters) + '"'
               cmdline_existed = True
           writer.write(line)
       if not cmdline_existed:
           line = grub_cmd + '=' + '"' + ' '.join(kernel_parameters) + '"'
           writer.write(line)

os.remove(old_grub_file)
os.system('update-grub')
os.system('reboot')
```

Trusted boot with partition image¶

Starting with the Liberty release, Ironic supports trusted boot with partition image. This means at the end of the deployment process, when the node is rebooted with the new user image, `trusted boot` will be performed. It will measure the node's BIOS, boot loader, Option ROM and the Kernel/Ramdisk, to determine whether a bare metal node deployed by Ironic should be trusted.

It's important to note that in order for this to work the node being deployed **must** have Intel TXT (http://en.wikipedia.org/wiki/Trusted_Execution_Technology) hardware support. The image being deployed with Ironic must have `oat-client` installed within it.

The following will describe how to enable `trusted boot` and boot with PXE and Nova:

1. Create a customized user image with `oat-client` installed:

   ```
   disk-image-create -u fedora baremetal oat-client -o $TRUST_IMG
   ```

   For more information on creating customized images, see Create and add images to the Image service (configure-integration.html#image-requirements).

2. Enable VT-x, VT-d, TXT and TPM on the node. This can be done manually through the BIOS. Depending on the platform, several reboots may be needed.

3. Enroll the node and update the node capability value:

   ```
   ironic node-create -d pxe_ipmitool

   ironic node-update $NODE_UUID add properties/capabilities={'trusted_boot':true}
   ```

4. Create a special flavor:

   ```
   nova flavor-key $TRUST_FLAVOR_UUID set 'capabilities:trusted_boot'=true
   ```

5. Prepare tboot (https://sourceforge.net/projects/tboot) and mboot.c32 and put them into tftp_root or http_root directory on all nodes with the ironic-conductor processes:

   ```
   Ubuntu:
       cp /usr/lib/syslinux/mboot.c32 /tftpboot/

   Fedora:
       cp /usr/share/syslinux/mboot.c32 /tftpboot/
   ```

   Note: The actual location of mboot.c32 varies among different distribution versions.

   tboot can be downloaded from https://sourceforge.net/projects/tboot/files/latest/download (https://sourceforge.net/projects/tboot/files/latest/download)

6. Install an OAT Server. An OAT Server (https://github.com/OpenAttestation/OpenAttestation/wiki) should be running and configured correctly.

7. Boot an instance with Nova:

```
nova boot --flavor $TRUST_FLAVOR_UUID --image $TRUST_IMG --user-data $TRUST_SCRIPT trusted_instance
```

Note that the node will be measured during `trusted boot` and the hash values saved into TPM (http://en.wikipedia.org/wiki/Trusted_Platform_Module). An example of TRUST_SCRIPT can be found in trust script example (https://wiki.openstack.org/wiki/Bare-metal-trust#Trust_Script_Example).

8. Verify the result via OAT Server.

   This is outside the scope of Ironic. At the moment, users can manually verify the result by following the manual verify steps (https://wiki.openstack.org/wiki/Bare-metal-trust#Manual_verify_result).

« (setup-drivers.html) » (troubleshooting.html) (https://bugs.launchpad.net/ironic/+filebug?field.title=Advanced%20features%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/advanced.rst%0AURL: http://docs.openstack.org/project-install-guide/baremetal/draft/advanced.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 (setup-drivers.html)
- Advanced features ()
  - Local boot with partition images
  - Specifying the disk for deployment (root device hints)
  - Appending kernel parameters to boot instances
  - Trusted boot with partition image
- 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/)