

---

# **nova-dpm Documentation**

*Release 1.0.0.0rc3.dev3*

**OpenStack Foundation**

February 23, 2017



<b>1</b>	<b>Overview</b>	<b>3</b>
1.1	Release Notes . . . . .	3
1.2	Topology . . . . .	4
1.3	Feature Support Matrix . . . . .	6
<b>2</b>	<b>Using the driver</b>	<b>17</b>
2.1	Installation . . . . .	17
2.2	Configuration . . . . .	18
2.3	DPM Guest Image Tools . . . . .	19
<b>3</b>	<b>Creating DPM Images</b>	<b>23</b>
3.1	Creating a qcow2 image for RHEL . . . . .	23
<b>4</b>	<b>Contributing to the project</b>	<b>31</b>
4.1	Contributing . . . . .	31
4.2	Developer Guide . . . . .	31
4.3	Nova-dpm specifications for Ocata . . . . .	32
<b>5</b>	<b>Links</b>	<b>49</b>



On IBM z Systems and IBM LinuxOne machines, certain workloads run better in a partition of the firmware-based PR/SM (Processor Resource/System Manager) hypervisor, than in a virtual machine of a software hypervisor such as KVM or z/VM.

This project provides a Nova virtualization driver for the PR/SM hypervisor of IBM z Systems and IBM LinuxOne machines that are in the DPM (Dynamic Partition Manager) administrative mode.

The DPM mode enables dynamic capabilities of the firmware-based PR/SM hypervisor that are usually known from software-based hypervisors, such as creation, deletion and modification of partitions (i.e. virtual machines) and virtual devices within these partitions, and dynamic assignment of these virtual devices to physical I/O adapters.

The z/VM and KVM hypervisors on z Systems and LinuxONE machines are supported by separate Nova virtualization drivers:

- KVM is supported by the standard libvirt/KVM driver in the [openstack/nova](#) project.
- z/VM is supported by the z/VM driver in the [openstack/nova-zvm-virt-driver](#) project.



---

## Release Notes

### 1.0.0

*nova-dpm* 1.0.0 is the first release of the Nova virtualization driver for the PR/SM hypervisor of IBM z Systems and IBM LinuxOne machines that are in the DPM (Dynamic Partition Manager) administrative mode.

#### New Features

- Configure a compute node to manage and consume only a subset of a *z Systems* CPC in DPM mode.
- CPC subsetting is hidden from users and they are treated like normal hosts in OpenStack.
- Spawn instance from FCP volume.
- Instance lifecycle management.
- Use flat networking.

#### Known Issues

- VLAN and tunneled networks are not supported in this release.
- Cinder driver for Storwize V7000 Unified returns additional WWPN's which are tagged as NAS, which are used for internal connections. The invalid target WWPN's need to be blacklisted in nova configuration parameters.
- Fibre Channel Multipathing is not supported.
- The configuration parameter [DEFAULT] `.host` cannot be more than 17 characters in length.
- Networking: 12 ports per partition at the maximum.
- Networking: In the guest image, always port 0 of a network adapter gets autoconfigured. If port 1 should be used, manually deconfigure port 0 and configure port 1 in the operating system of the launched instance.
- Boot from image is not available. Boot from volume has to be used.
- Only a single fibre channel network is supported. Configured storage adapters and cinder fibre channel backends must all use the same fibre channel network.
- The configured maximum number of partitions (`[dpm].max_instances`) is not yet enforced.
- All bug reports are listed at: <https://bugs.launchpad.net/nova-dpm>

## Topology

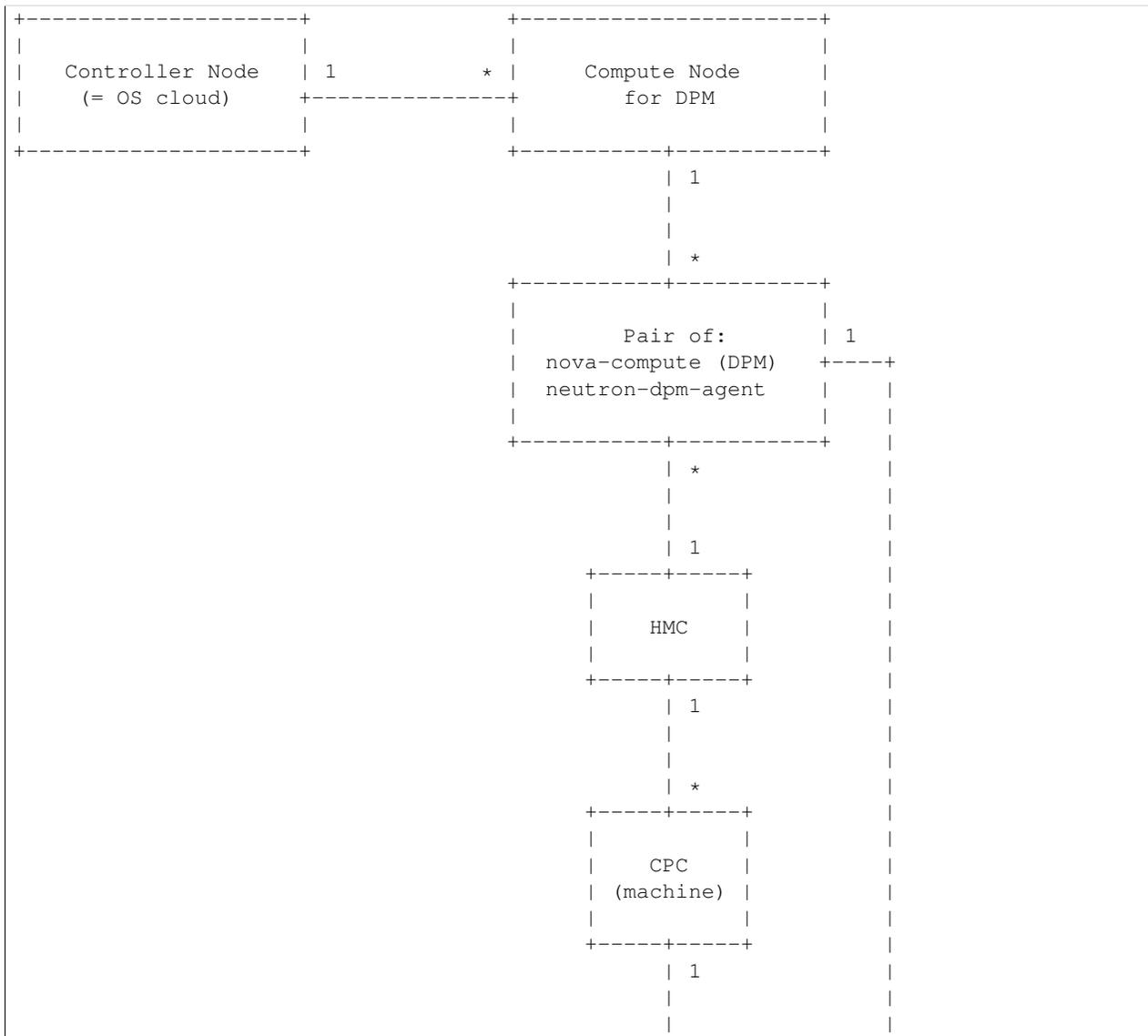
This section describes the topology between the OpenStack compute node for DPM, the z Systems Hardware Management Console (HMC) and the managed machines (CPCs).

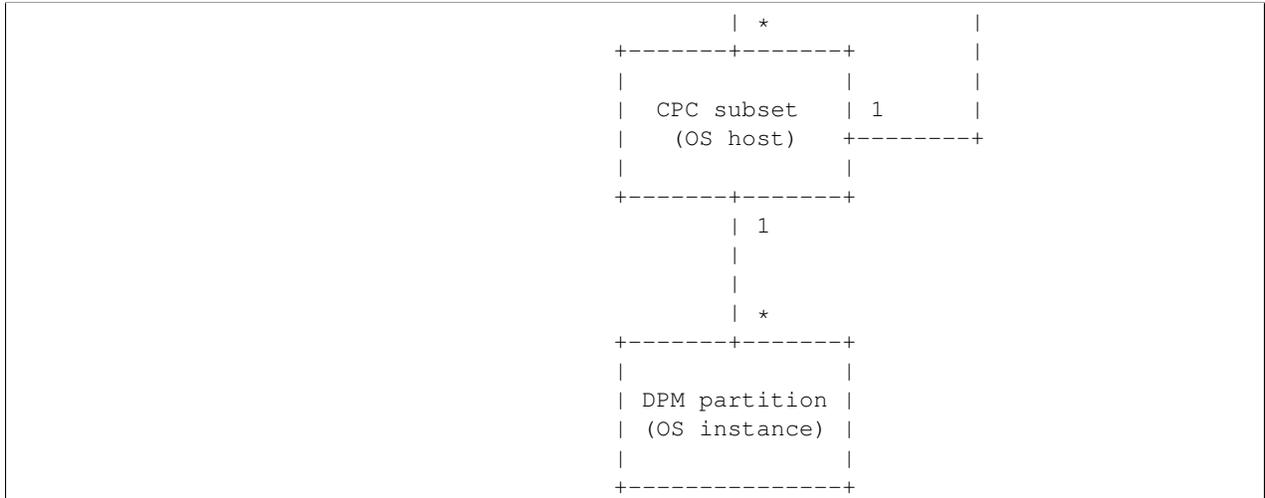
### Topology for a single OpenStack cloud

To keep it simple, we start with explaining the topology for a single OpenStack cloud with compute nodes for DPM. The controller node is only shown as a means to denote the OpenStack cloud.

The following entity-relationship diagram shows the entities related to OpenStack compute nodes for DPM for a single OpenStack cloud.

The diagram presents multiplicities (cardinalities) on the relations using the look-across semantics known from UML associations (e.g. The “1” on the left side of the relation between controller node and compute node means that one compute node is related to “1” controller node, and the “\*” on the right side of that relation means that one controller node is related to “\*” (= 0 to N) compute nodes.





Explanation:

- The controller node is the management plane of the OpenStack cloud - it is only shown to relate the compute nodes to an OpenStack cloud. It can run on any (supported) operating system and hardware architecture.
- Within an OpenStack cloud, there can be many compute nodes for DPM (along with compute nodes for other hypervisor types and hardware architectures).
- Each compute node for DPM can run the services for multiple OpenStack “hosts”. For OpenStack, a “host” is a hypervisor instance that can run multiple virtual systems (the OpenStack instances). The OpenStack instances are DPM partitions on a CPC.
- An OpenStack host is established by defining a subset of a CPC. A CPC subset is defined in the DPM-specific part of the Nova config files of its compute node with the following characteristics:
  - A maximum number of DPM partitions that can be created.
  - A maximum number of physical CPUs that can be used.
  - A maximum amount of physical memory that can be used.

The construct of a CPC subset limits the resources used for an OpenStack host, ensuring that the OpenStack host cannot exhaust the resources of an entire CPC. This allows other OpenStack hosts or non-OpenStack workload to coexist on the same CPC.

- For each OpenStack host, the compute node needs a pair of:
  - the nova-compute service for DPM (that is, with the nova-dpm virtualization driver)
  - the neutron-dpm-agent service

The multi-host capability at the level of the nova-compute service is not exploited for DPM; multiple hosts are supported by having multiple pairs of services.

- There is no need to run all pairs of nova-compute and neutron-dpm-agent services on the same compute node; they can also be spread across multiple compute nodes.
- The services on a compute node for DPM connect to an HMC over a network and therefore the compute node can run on any (supported) operating system and hardware architecture.
- The HMC can be duplicated into a primary and alternate HMC. In this OpenStack release, the nova-compute service for DPM and the neutron-dpm-agent service can be configured to connect to only one HMC.



When considering which capabilities should be marked as mandatory the following general guiding principles were applied

- **Inclusivity** - people have shown ability to make effective use of a wide range of virtualization technologies with broadly varying featuresets. Aiming to keep the requirements as inclusive as possible, avoids second-guessing what a user may wish to use the cloud compute service for.
- **Bootstrapping** - a practical use case test is to consider that starting point for the compute deploy is an empty data center with new machines and network connectivity. The look at what are the minimum features required of a compute service, in order to get user instances running and processing work over the network.
- **Competition** - an early leader in the cloud compute service space was Amazon EC2. A sanity check for whether a feature should be mandatory is to consider whether it was available in the first public release of EC2. This had quite a narrow featureset, but none the less found very high usage in many use cases. So it serves to illustrate that many features need not be considered mandatory in order to get useful work done.
- **Reality** - there are many virt drivers currently shipped with Nova, each with their own supported feature set. Any feature which is missing in at least one virt driver that is already in-tree, must by inference be considered optional until all in-tree drivers support it. This does not rule out the possibility of a currently optional feature becoming mandatory at a later date, based on other principles above.

Summary

Feature	Status	IBM DPM
<i>Attach block volume to instance</i>	optional	
<i>Detach block volume from instance</i>	optional	
<i>Attach virtual network interface to instance</i>	optional	
<i>Detach virtual network interface from instance</i>	optional	
<i>Set the host in a maintenance mode</i>	optional	
<i>Evacuate instances from a host</i>	optional	
<i>Rebuild instance</i>	optional	
<i>Guest instance status</i>	mandatory	
<i>Guest host status</i>	optional	
<i>Live migrate instance across hosts</i>	optional	
<i>Force live migration to complete</i>	optional	
<i>Launch instance</i>	mandatory	
<i>Stop instance CPUs (pause)</i>	optional	
<i>Reboot instance</i>	optional	
<i>Rescue instance</i>	optional	
<i>Resize instance</i>	optional	
<i>Restore instance</i>	optional	
<i>Service control</i>	optional	
<i>Set instance admin password</i>	optional	
<i>Save snapshot of instance disk</i>	optional	
<i>Suspend instance</i>	optional	
<i>Swap block volumes</i>	optional	
<i>Shutdown instance</i>	mandatory	
<i>Trigger crash dump</i>	optional	
<i>Resume instance CPUs (unpause)</i>	optional	
<i>Auto configure disk</i>	optional	
<i>Instance disk I/O limits</i>	optional	
<i>Config drive support</i>	choice	
<i>Inject files into disk image</i>	optional	
<i>Inject guest networking config</i>	optional	
Continued on next page		

Table 1.1 – continued from previous page

<i>Feature</i>	<i>Status</i>	<b>IBM DPM</b>
<i>Remote desktop over RDP</i>	choice	
<i>View serial console logs</i>	choice	
<i>Remote interactive serial console</i>	choice	
<i>Remote desktop over SPICE</i>	choice	
<i>Remote desktop over VNC</i>	choice	
<i>Block storage support</i>	optional	
<i>Block storage over fibre channel</i>	optional	
<i>Block storage over iSCSI</i>	condition	
<i>CHAP authentication for iSCSI</i>	optional	
<i>Image storage support</i>	mandatory	
<i>Network firewall rules</i>	optional	
<i>Network routing</i>	optional	
<i>Network security groups</i>	optional	
<i>Flat networking</i>	choice	
<i>VLAN networking</i>	choice	
<i>uefi boot</i>	optional	

#### Details

- **Attach block volume to instance Status: optional.** The attach volume operation provides a means to hotplug additional block storage to a running instance. This allows storage capabilities to be expanded without interruption of service. In a cloud model it would be more typical to just spin up a new instance with large storage, so the ability to hotplug extra storage is for those cases where the instance is considered to be more of a pet than cattle. Therefore this operation is not considered to be mandatory to support.

#### CLI commands:

```
- nova volume-attach <server> <volume>
```

#### drivers:

```
- IBM DPM: missing
```

- **Detach block volume from instance Status: optional.** See notes for attach volume operation.

#### CLI commands:

```
- nova volume-detach <server> <volume>
```

#### drivers:

```
- IBM DPM: missing
```

- **Attach virtual network interface to instance Status: optional.** The attach interface operation provides a means to hotplug additional interfaces to a running instance. Hotplug support varies between guest OSes and some guests require a reboot for new interfaces to be detected. This operation allows interface capabilities to be expanded without interruption of service. In a cloud model it would be more typical to just spin up a new instance with more interfaces.

#### CLI commands:

```
- nova interface-attach <server>
```

#### drivers:

```
- IBM DPM: missing
```

- **Detach virtual network interface from instance Status: optional.** See notes for attach-interface operation.

**CLI commands:**

- nova interface-detach <server> <port\_id>

**drivers:**

- **IBM DPM:** missing

- **Set the host in a maintenance mode Status: optional.** This operation allows a host to be placed into maintenance mode, automatically triggering migration of any running instances to an alternative host and preventing new instances from being launched. This is not considered to be a mandatory operation to support. The driver methods to implement are “host\_maintenance\_mode” and “set\_host\_enabled”.

**CLI commands:**

- nova host-update <host>

**drivers:**

- **IBM DPM:** missing

- **Evacuate instances from a host Status: optional.** A possible failure scenario in a cloud environment is the outage of one of the compute nodes. In such a case the instances of the down host can be evacuated to another host. It is assumed that the old host is unlikely ever to be powered back on, otherwise the evacuation attempt will be rejected. When the instances get moved to the new host, their volumes get re-attached and the locally stored data is dropped. That happens in the same way as a rebuild. This is not considered to be a mandatory operation to support.

**CLI commands:**

- nova evacuate <server>
- nova host-evacuate <host>

**drivers:**

- **IBM DPM:** missing

- **Rebuild instance Status: optional.** A possible use case is additional attributes need to be set to the instance, nova will purge all existing data from the system and remakes the VM with given information such as ‘metadata’ and ‘personalities’. Though this is not considered to be a mandatory operation to support.

**CLI commands:**

- nova rebuild <server> <image>

**drivers:**

- **IBM DPM:** missing

- **Guest instance status Status: mandatory.** Provides a quick report on information about the guest instance, including the power state, memory allocation, CPU allocation, number of vCPUs and cumulative CPU execution time. As well as being informational, the power state is used by the compute manager for tracking changes in guests. Therefore this operation is considered mandatory to support.

**drivers:**

- **IBM DPM:** complete

- **Guest host status Status: optional.** Unclear what this refers to

**drivers:**

- **IBM DPM:** missing

- **Live migrate instance across hosts Status: optional.** Live migration provides a way to move an instance off one compute host, to another compute host. Administrators may use this to evacuate instances from a host that needs to undergo maintenance tasks, though of course this may not help if the host is already suffering a failure. In general instances are considered cattle rather than pets, so it is expected that an instance is liable to be killed if host maintenance is required. It is technically challenging for some hypervisors to provide support for the live migration operation, particularly those built on the container based virtualization. Therefore this operation is not considered mandatory to support.

**CLI commands:**

- `nova live-migration <server>`
- `nova host-evacuate-live <host>`

**drivers:**

- **IBM DPM:** `missing`

- **Force live migration to complete Status: optional.** Live migration provides a way to move a running instance to another compute host. But it can sometimes fail to complete if an instance has a high rate of memory or disk page access. This operation provides the user with an option to assist the progress of the live migration. The mechanism used to complete the live migration depends on the underlying virtualization subsystem capabilities. If libvirt/qemu is used and the post-copy feature is available and enabled then the force complete operation will cause a switch to post-copy mode. Otherwise the instance will be suspended until the migration is completed or aborted.

**CLI commands:**

- `nova live-migration-force-complete <server> <migration>`

**drivers:**

- **IBM DPM:** `missing`

- **Launch instance Status: mandatory.** Importing pre-existing running virtual machines on a host is considered out of scope of the cloud paradigm. Therefore this operation is mandatory to support in drivers.

**drivers:**

- **IBM DPM:** `complete`

- **Stop instance CPUs (pause) Status: optional.** Stopping an instances CPUs can be thought of as roughly equivalent to suspend-to-RAM. The instance is still present in memory, but execution has stopped. The problem, however, is that there is no mechanism to inform the guest OS that this takes place, so upon unpausing, its clocks will no longer report correct time. For this reason hypervisor vendors generally discourage use of this feature and some do not even implement it. Therefore this operation is considered optional to support in drivers.

**CLI commands:**

- `nova pause <server>`

**drivers:**

- **IBM DPM:** `missing`

- **Reboot instance Status: optional.** It is reasonable for a guest OS administrator to trigger a graceful reboot from inside the instance. A host initiated graceful reboot requires guest co-operation and a non-graceful reboot can be achieved by a combination of stop+start. Therefore this operation is considered optional.

**CLI commands:**

- `nova reboot <server>`

**drivers:**

- **IBM DPM:** complete Please note that this is will always be a hard reboot, as the hypervisor doesn't support soft reboots.

- **Rescue instance Status: optional.** The rescue operation starts an instance in a special configuration whereby it is booted from an special root disk image. The goal is to allow an administrator to recover the state of a broken virtual machine. In general the cloud model considers instances to be cattle, so if an instance breaks the general expectation is that it be thrown away and a new instance created. Therefore this operation is considered optional to support in drivers.

**CLI commands:**

- `nova rescue <server>`

**drivers:**

- **IBM DPM:** missing

- **Resize instance Status: optional.** The resize operation allows the user to change a running instance to match the size of a different flavor from the one it was initially launched with. There are many different flavor attributes that potentially need to be updated. In general it is technically challenging for a hypervisor to support the alteration of all relevant config settings for a running instance. Therefore this operation is considered optional to support in drivers.

**CLI commands:**

- `nova resize <server> <flavor>`

**drivers:**

- **IBM DPM:** missing

- **Restore instance Status: optional.** See notes for the suspend operation

**CLI commands:**

- `nova resume <server>`

**drivers:**

- **IBM DPM:** missing

- **Service control Status: optional.** Something something, dark side, something something. Hard to claim this is mandatory when no one seems to know what “Service control” refers to in the context of virt drivers.

**drivers:**

- **IBM DPM:** missing

- **Set instance admin password Status: optional.** Provides a mechanism to (re)set the password of the administrator account inside the instance operating system. This requires that the hypervisor has a way to communicate with the running guest operating system. Given the wide range of operating systems in existence it is unreasonable to expect this to be practical in the general case. The configdrive and metadata service both provide a mechanism for setting the administrator password at initial boot time. In the case where this operation were not available, the administrator would simply have to login to the guest and change the password in the normal manner, so this is just a convenient optimization. Therefore this operation is not considered mandatory for drivers to support.

**CLI commands:**

- `nova set-password <server>`

**drivers:**

- **IBM DPM:** missing

- **Save snapshot of instance disk Status: optional.** The snapshot operation allows the current state of the instance root disk to be saved and uploaded back into the glance image repository. The instance can later be booted again using this saved image. This is in effect making the ephemeral instance root disk into a semi-persistent storage, in so much as it is preserved even though the guest is no longer running. In general though, the expectation is that the root disks are ephemeral so the ability to take a snapshot cannot be assumed. Therefore this operation is not considered mandatory to support.

**CLI commands:**

```
- nova image-create <server> <name>
```

**drivers:**

```
- IBM DPM: missing
```

- **Suspend instance Status: optional.** Suspending an instance can be thought of as roughly equivalent to suspend-to-disk. The instance no longer consumes any RAM or CPUs, with its live running state having been preserved in a file on disk. It can later be restored, at which point it should continue execution where it left off. As with stopping instance CPUs, it suffers from the fact that the guest OS will typically be left with a clock that is no longer telling correct time. For container based virtualization solutions, this operation is particularly technically challenging to implement and is an area of active research. This operation tends to make more sense when thinking of instances as pets, rather than cattle, since with cattle it would be simpler to just terminate the instance instead of suspending. Therefore this operation is considered optional to support.

**CLI commands:**

```
- nova suspend <server>
```

**drivers:**

```
- IBM DPM: missing
```

- **Swap block volumes Status: optional.** The swap volume operation is a mechanism for changing a running instance so that its attached volume(s) are backed by different storage in the host. An alternative to this would be to simply terminate the existing instance and spawn a new instance with the new storage. In other words this operation is primarily targeted towards the pet use case rather than cattle, however, it is required for volume migration to work in the volume service. This is considered optional to support.

**CLI commands:**

```
- nova volume-update <server> <attachment> <volume>
```

**drivers:**

```
- IBM DPM: missing
```

- **Shutdown instance Status: mandatory.** The ability to terminate a virtual machine is required in order for a cloud user to stop utilizing resources and thus avoid indefinitely ongoing billing. Therefore this operation is mandatory to support in drivers.

**CLI commands:**

```
- nova delete <server>
```

**drivers:**

```
- IBM DPM: complete
```

- **Trigger crash dump Status: optional.** The trigger crash dump operation is a mechanism for triggering a crash dump in an instance. The feature is typically implemented by injecting an NMI (Non-maskable Interrupt) into the instance. It provides a means to dump the production memory image as a dump file which is useful for users. Therefore this operation is considered optional to support.

**drivers:**

- **IBM DPM:** `missing`

- **Resume instance CPUs (unpause) Status: optional.** See notes for the “Stop instance CPUs” operation

**CLI commands:**

- `nova unpause <server>`

**drivers:**

- **IBM DPM:** `missing`

- **Auto configure disk Status: optional.** something something, dark side, something something. Unclear just what this is about.

**drivers:**

- **IBM DPM:** `missing`

- **Instance disk I/O limits Status: optional.** The ability to set rate limits on virtual disks allows for greater performance isolation between instances running on the same host storage. It is valid to delegate scheduling of I/O operations to the hypervisor with its default settings, instead of doing fine grained tuning. Therefore this is not considered to be an mandatory configuration to support.

**CLI commands:**

- `nova limits`

**drivers:**

- **IBM DPM:** `missing`

- **Config drive support Status: choice(guest.setup).** The config drive provides an information channel into the guest operating system, to enable configuration of the administrator password, file injection, registration of SSH keys, etc. Since cloud images typically ship with all login methods locked, a mechanism to set the administrator password of keys is required to get login access. Alternatives include the metadata service and disk injection. At least one of the guest setup mechanisms is required to be supported by drivers, in order to enable login access.

**drivers:**

- **IBM DPM:** `missing`

- **Inject files into disk image Status: optional.** This allows for the end user to provide data for multiple files to be injected into the root filesystem before an instance is booted. This requires that the compute node understand the format of the filesystem and any partitioning scheme it might use on the block device. This is a non-trivial problem considering the vast number of filesystems in existence. The problem of injecting files to a guest OS is better solved by obtaining via the metadata service or config drive. Therefore this operation is considered optional to support.

**drivers:**

- **IBM DPM:** `missing`

- **Inject guest networking config Status: optional.** This allows for static networking configuration (IP address, netmask, gateway and routes) to be injected directly into the root filesystem before an instance is booted. This requires that the compute node understand how networking is configured in the guest OS which is a non-trivial problem considering the vast number of operating system types. The problem of configuring networking is better solved by DHCP or by obtaining static config via config drive. Therefore this operation is considered optional to support.

**drivers:**

- **IBM DPM:** `missing` Only DHCP is supported.

- **Remote desktop over RDP Status: choice(console).** This allows the administrator to interact with the graphical console of the guest OS via RDP. This provides a way to see boot up messages and login to the instance when networking configuration has failed, thus preventing a network based login. Some operating systems may prefer to emit messages via the serial console for easier consumption. Therefore support for this operation is not mandatory, however, a driver is required to support at least one of the listed console access operations.

**CLI commands:**

```
- nova get-rdp-console <server> <console-type>
```

**drivers:**

```
- IBM DPM: missing
```

- **View serial console logs Status: choice(console).** This allows the administrator to query the logs of data emitted by the guest OS on its virtualized serial port. For UNIX guests this typically includes all boot up messages and so is useful for diagnosing problems when an instance fails to successfully boot. Not all guest operating systems will be able to emit boot information on a serial console, others may only support graphical consoles. Therefore support for this operation is not mandatory, however, a driver is required to support at least one of the listed console access operations.

**drivers:**

```
- IBM DPM: missing
```

- **Remote interactive serial console Status: choice(console).** This allows the administrator to interact with the serial console of the guest OS. This provides a way to see boot up messages and login to the instance when networking configuration has failed, thus preventing a network based login. Not all guest operating systems will be able to emit boot information on a serial console, others may only support graphical consoles. Therefore support for this operation is not mandatory, however, a driver is required to support at least one of the listed console access operations. This feature was introduced in the Juno release with blueprint <https://blueprints.launchpad.net/nova/+spec/serial-ports>

**CLI commands:**

```
- nova get-serial-console <server>
```

**drivers:**

```
- IBM DPM: missing
```

- **Remote desktop over SPICE Status: choice(console).** This allows the administrator to interact with the graphical console of the guest OS via SPICE. This provides a way to see boot up messages and login to the instance when networking configuration has failed, thus preventing a network based login. Some operating systems may prefer to emit messages via the serial console for easier consumption. Therefore support for this operation is not mandatory, however, a driver is required to support at least one of the listed console access operations.

**CLI commands:**

```
- nova get-spice-console <server> <console-type>
```

**drivers:**

```
- IBM DPM: missing
```

- **Remote desktop over VNC Status: choice(console).** This allows the administrator to interact with the graphical console of the guest OS via VNC. This provides a way to see boot up messages and login to the instance when networking configuration has failed, thus preventing a network based login. Some operating systems may prefer to emit messages via the serial console for easier consumption. Therefore support for this operation is not mandatory, however, a driver is required to support at least one of the listed console access operations.

**CLI commands:**

```
- nova get-vnc-console <server> <console-type>
```

**drivers:**

```
- IBM DPM: missing
```

- **Block storage support Status: optional.** Block storage provides instances with direct attached virtual disks that can be used for persistent storage of data. As an alternative to direct attached disks, an instance may choose to use network based persistent storage. OpenStack provides object storage via the Swift service, or a traditional filesystem such as NFS/GlusterFS may be used. Some types of instances may not require persistent storage at all, being simple transaction processing systems reading requests & sending results to and from the network. Therefore support for this configuration is not considered mandatory for drivers to support.

**drivers:**

```
- IBM DPM: complete
```

- **Block storage over fibre channel Status: optional.** To maximise performance of the block storage, it may be desirable to directly access fibre channel LUNs from the underlying storage technology on the compute hosts. Since this is just a performance optimization of the I/O path it is not considered mandatory to support.

**drivers:**

```
- IBM DPM: complete
```

- **Block storage over iSCSI Status: condition(storage.block==missing).** If the driver wishes to support block storage, it is common to provide an iSCSI based backend to access the storage from cinder. This isolates the compute layer for knowledge of the specific storage technology used by Cinder, albeit at a potential performance cost due to the longer I/O path involved. If the driver chooses to support block storage, then this is considered mandatory to support, otherwise it is considered optional.

**drivers:**

```
- IBM DPM: missing
```

- **CHAP authentication for iSCSI Status: optional.** If accessing the cinder iSCSI service over an untrusted LAN it is desirable to be able to enable authentication for the iSCSI protocol. CHAP is the commonly used authentication protocol for iSCSI. This is not considered mandatory to support. (?)

**drivers:**

```
- IBM DPM: missing
```

- **Image storage support Status: mandatory.** This refers to the ability to boot an instance from an image stored in the glance image repository. Without this feature it would not be possible to bootstrap from a clean environment, since there would be no way to get block volumes populated and reliance on external PXE servers is out of scope. Therefore this is considered a mandatory storage feature to support.

**CLI commands:**

```
- nova boot --image <image> <name>
```

**drivers:**

```
- IBM DPM: partial The image needs to be deployed into a volume first. As the system z DPM machine doesn't provide local storage, booting from image, which gets downloaded to local storage first, is not supported.
```

- **Network firewall rules Status: optional.** Unclear how this is different from security groups

**drivers:**

```
- IBM DPM: missing
```

- **Network routing Status: optional.** Unclear what this refers to

**drivers:**

- **IBM DPM:** `missing`

- **Network security groups Status: optional.** The security groups feature provides a way to define rules to isolate the network traffic of different instances running on a compute host. This would prevent actions such as MAC and IP address spoofing, or the ability to setup rogue DHCP servers. In a private cloud environment this may be considered to be a superfluous requirement. Therefore this is considered to be an optional configuration to support.

**drivers:**

- **IBM DPM:** `missing`

- **Flat networking Status: choice(networking.topology).** Provide network connectivity to guests using a flat topology across all compute nodes. At least one of the networking configurations is mandatory to support in the drivers.

**drivers:**

- **IBM DPM:** `complete`

- **VLAN networking Status: choice(networking.topology).** Provide network connectivity to guests using VLANs to define the topology. At least one of the networking configurations is mandatory to support in the drivers.

**drivers:**

- **IBM DPM:** `missing` A config driver, which is not yet supported, is a prerequisite for that.

- **uefi boot Status: optional.** This allows users to boot a guest with uefi firmware.

**drivers:**

- **IBM DPM:** `missing`

Notes

---

## Using the driver

---

### Installation

The nova-dpm virtualization driver must be installed on every OpenStack compute node for DPM.

This section describes the manual installation of the nova-dpm driver onto a compute node that has already been installed by some means.

The nova-dpm virtualization driver is released on PyPI as package [nova-dpm](#).

The following table indicates which version of the nova-dpm package on PyPI to use for a particular OpenStack release:

OpenStack release	nova-dpm version
Ocata	1.x.x

Typically, the nova-dpm package will increase its major version number by one for each new OpenStack release.

If you want to install the package for a particular OpenStack release, it is recommended to use the packages that have been released to PyPI, rather than installing from a particular branch of a Git repository.

To do that, identify the major version number for the desired OpenStack release from the table above, and install the latest minor and fix version of the package for that major version, also specifying the global upper constraints file for the desired OpenStack release (the latter ensures that you get the right versions of any dependent packages).

For example, for Ocata:

```
$ constraints_file=https://git.openstack.org/cgit/openstack/requirements/plain/upper-constraints.txt
$ pip install -c$constraints_file "nova-dpm >=1,<2"
```

If you have good reasons to install the latest not yet released fix level of the nova-dpm package for a particular (released) OpenStack release, install the nova-dpm package from the stable branch of the GitHub repo for that OpenStack release:

For example, for Ocata:

```
$ constraints_file=https://git.openstack.org/cgit/openstack/requirements/plain/upper-constraints.txt
$ pip install -c$constraints_file git+https://git.openstack.org/openstack/nova-dpm@stable/ocata
```

If you are a developer and want to install the latest code of the nova-dpm package for the OpenStack release that is in development:

```
$ constraints_file=https://git.openstack.org/cgit/openstack/requirements/plain/upper-constraints.txt
$ pip install -c$constraints_file git+https://git.openstack.org/openstack/nova-dpm@master
```

The pip commands above install the packages into the currently active Python environment.

If your active Python environment is a virtual Python environment, the commands above can be issued from a userid without sudo rights.

If you need to install the packages into the system Python environment, you need sudo rights:

```
$ sudo pip install ...
```

After installing the nova-dpm driver, proceed with its *Configuration*.

Note that you will also need to install and configure the networking-dpm package on the compute node. For its documentation, see <http://networking-dpm.readthedocs.io/en/latest/>.

## Configuration

The following is a sample `nova_dpm.conf` configuration file for the nova-dpm driver, for adaptation and use.

It is auto-generated from the nova-dpm project when this documentation is built, so if you are having issues with an option, please compare your version of the nova-dpm Python package with the version of this documentation.

The sample configuration can also be viewed in file form.

```
[DEFAULT]

[dpm]
#
# Configuration options for IBM z Systems and IBM LinuxONE in DPM (Dynamic
# Partition Manager) administrative mode. A z Systems or LinuxONE machine is
# termed "CPC" (Central Processor Complex). The CPCs are managed via the Web
# Services API exposed by the "HMC" (Hardware Management Console). One HMC can
# manage multiple CPCs.
#
#
# DPM config options for the Nova compute service (one for each OpenStack
# hypervisor host) specify the target CPC, the HMC managing it, and limits on
# the
# resource usage on the target CPC. These limits ensure that only a subset of
# the
# target CPC is used for the OpenStack hypervisor host. To use the Nova driver
# for DPM, the `[DEFAULT].compute_driver` config option needs to be set to the
# value `dpm.DPMDriver`.
#
# From nova_dpm
#
#
#     Maximum number of shared physical IFL processors on the target CPC that
#     can
#     be used for this OpenStack hypervisor host (integer value)
#max_processors = <None>
#
#
#     Maximum amount of memory (in MiB) on the target CPC that can be used for
#     this OpenStack hypervisor host (integer value)
#max_memory = <None>
```

```

#
#   Maximum number of instances (partitions) that can be created for this
#   OpenStack hypervisor host (integer value)
#max_instances = <None>

#
#   Physical storage adapter with port details for hba creation (multi valued)
#physical_storage_adapter_mappings =

#
#   list of target/remote wwpns can be used for example to exclude NAS/file
#   WWPNS returned by the V7000 Unified. (list value)
#target_wwpn_ignore_list =

#
#   Hostname or IP address of the HMC that manages the target CPC (string
#   value)
#hmc = <None>

#
#   User name for connection to the HMC (string value)
#hmc_username = <None>

#
#   Password for connection to the HMC (string value)
#hmc_password = <None>

#
#   DPM Object-id of the target CPC (string value)
#cpc_object_id = <None>

```

## DPM Guest Image Tools

The DPM Guest Image Tools must be installed within a DPM OpenStack image. The purpose of the tools are to dynamically configure the network interfaces.

Doing IP configuration is not part of the tools. This is handled like usual with cloud-init.

### autoconfigure\_networking

#### Description

Is used to configure all network interfaces that are listed in the kernels cmdline */proc/cmdline* with the given adapter port. All interfaces are configured in layer2 mode.

The format of the data in the cmdline must be

```
<devno>,<port>[,<mac>];
```

Example

```
0001,1,0a0000000011;0004,0;
```

This will result in

- 0001 being configured with port 1

- 0004 being configured with port 0

## Content

- systemd service `autoconfigure_networking.service`
- shell script `autoconfigure_networking.sh`

## Trigger

The systemd service `autoconfigure_networking.service` is configured to run before cloud-init during boot. It's job is to trigger the shell script.

Manual execution of the shell script

```
/usr/bin/autoconfigure_networking.sh
```

## Installation

- Place the following files in the guest image
  - `dpm_guest_tools/usr/bin/autoconfigure_networking.sh`  
  -> `/usr/bin/autoconfigure_networking.sh`
  - `dpm_guest_tools/usr/lib/systemd/system/autoconfigure_networking.service`  
  -> `/usr/lib/systemd/systemd/autoconfigure_networking.service`
- Ensure permissions

```
chmod 644 /usr/lib/systemd/system/autoconfigure_networking.service
```
- Enable the service for autostart
  - `systemctl enable autoconfigure_networking.service`

## setmac

### Description

Is used to reconfigure the MAC address of a network interface. The mapping must be provided via the kernels cmdline `/proc/cmdline`.

The format of the data in the cmdline must be

```
<devno>,<portno>,<mac>;
```

Example

```
0001,1,0a0000000011;0004,0;
```

- 0001: corresponding interface will be set to mac `0a:00:00:00:00:11`
- 0004: mac will not be changed

## Content

- shell script setmac.sh
- udev rule 80-setmac.rules

## Trigger

If a new network interface gets configured (e.g. for device 0.0.0001), the udev rules triggers the shell script passing in the device-bus-id.

If a service instance for a certain device-bus-id already exists, it will not get started again.

Manual execution of the shell script

```
/usr/bin/setmac.sh <dev-bus-id>
```

## Installation

- Place the following files in the guest image
  - dpm\_guest\_tools/usr/bin/setmac.sh
    - > /usr/bin/setmac.sh
  - dpm\_guest\_tools/etc/udev/rules.d/80-setmac.rules
    - > /etc/udev/rules.d/80-setmac.rules



---

## Creating DPM Images

---

### Creating a qcow2 image for RHEL

This section explains the qcow2 image creation for RHEL.

#### Precondition

Partition with RHEL-7.3 installed and root user access

#### Update boot loader

1. Go to `/etc/zipl.conf` and remove all occurrences of `rd.zfcp=`
2. Add a new `rd.zfcp` entry
 

```
rd.zfcp=ipldev
```
3. Empty `/etc/zfcp.conf` file
 

```
echo "" > /etc/zfcp.conf
```
4. Create the `dasd.conf` file # In order to avoid error messages related to dasd configuration
 

```
touch /etc/dasd.conf
```
5. Go to `/usr/lib/dracut/modules.d/95zfcp/parse-zfcp.sh`

Apply the following differences:

```
$ diff old_parse-zfcp.sh new_parse-zfcp.sh
8,9c8,10 < echo $zfcp_arg | grep '\.[0-9a-fA-F]\.\.[0-9a-fA-F]\{4\},0x[0-9a-fA-F]\{16\},0x[0-9a-f
< test $? -ne 0 && die "For argument 'rd.zfcp=$zfcp_arg'\nSorry, invalid format."
--- > if [ "$zfcp_arg" == "ipldev" -a "$(cat /sys/firmware/ipl/ipl_type)" == "fcp" ] ; then
> zfcp_arg="$(cat /sys/firmware/ipl/device),$(cat /sys/firmware/ipl/wwpn),$(cat /sys/fi
> fi
```

The result should look like this:

```
#!/bin/sh
# -*- mode: shell-script; indent-tabs-mode: nil; sh-basic-offset: 4; -*-
# ex: ts=8 sw=4 sts=4 et filetype=sh

getargbool 1 rd.zfcp.conf -d -n rd_NO_ZFCPCONF || rm /etc/zfcp.conf
```

```
for zfcplib in $(getargs rd.zfcplib -d 'rd_ZFCP='); do
    if [ "$zfcplib" == "ipldev" -a "$(cat /sys/firmware/ipl/ipl_type)" == "fcp" ]; then
        zfcplib="$(cat /sys/firmware/ipl/device),$(cat /sys/firmware/ipl/wwpn),$(cat /sys/firmware/ipl/wwpn)"
    fi
    (
        IFS=","
        set $zfcplib
        echo "$@" >> /etc/zfcplib.conf
    )
done

zfcplib_free
```

6. **Rebuild the ramdisk** *dracut -f*

7. **To apply the above changes to the contents of boot loader script** *zipl -V*

## Installation of Cloud-init

### Add the RHEL7.3 yum repository

- Add the yum repository file that points to a network resource

```
cat <<EOT > /etc/yum.repos.d/rhel.repo
[RHEL7.3]
name=Red Hat Enterprise Linux Repository
baseurl=https://x.x.x.x
enabled=1
gpgcheck=0
EOT
```

### Install cloud-init 0.7.9

Download latest cloud-init from <https://launchpad.net/cloud-init/+download>

- Install python setuptools

```
yum install python-setuptools
```

- Extract it

```
tar -xf cloud-init-0.7.9.tar.gz
```

- Enter the extracted directory

```
cd cloud-init-0.7.9
```

- Build and install it:

```
python setup.py build
```

```
python setup.py install --init-system systemd
```

### Update cloud-init service files

- Remove Default dependencies

```
sed -i '/^\[Unit\]\$/,/^\[/ s/^DefaultDependencies=no/#DefaultDependencies=no/' /usr/lib/systemd/sy
sed -i '/^\[Unit\]\$/,/^\[/ s/^DefaultDependencies=no/#DefaultDependencies=no/' /usr/lib/systemd/sy
```

- Remove ordering for sysinit.target

```
sed -i '/^\[Unit\]\$/,/^\[/ s/^Before=sysinit.target/#Before=sysinit.target/' /usr/lib/systemd/sy
sed -i '/^\[Unit\]\$/,/^\[/ s/^Before=sysinit.target/#Before=sysinit.target/' /usr/lib/systemd/sy
```

- order with systemd-hostnamed.service

```
sed -i '/^\[Unit\]\$/,/^\[/ s/^After=networking.service/After=networking.service\nAfter=systemd-h
```

The result should look like this:

```
cat /usr/lib/systemd/system/cloud-init.service
```

```
[Unit]
Description=Initial cloud-init job (metadata service crawler)
#DefaultDependencies=no
Wants=cloud-init-local.service
Wants=sshd-keygen.service
Wants=sshd.service
After=cloud-init-local.service
After=networking.service
After=systemd-hostnamed.service
Before=network-online.target
Before=sshd-keygen.service
Before=sshd.service
#Before=sysinit.target
Before=systemd-user-sessions.service
Conflicts=shutdown.target

[Service]
Type=oneshot
ExecStart=/usr/bin/cloud-init init
RemainAfterExit=yes
TimeoutSec=0

# Output needs to appear in instance console output
StandardOutput=journal+console

[Install]
WantedBy=cloud-init.target
```

```
cat /usr/lib/systemd/system/cloud-init-local.service
```

```
[Unit]
Description=Initial cloud-init job (pre-networking)
#DefaultDependencies=no
Wants=network-pre.target
After=systemd-remount-fs.service
Before=NetworkManager.service
Before=network-pre.target
Before=shutdown.target
#Before=sysinit.target
Conflicts=shutdown.target
RequiresMountsFor=/var/lib/cloud
```

```
[Service]
Type=oneshot
ExecStart=/usr/bin/cloud-init init --local
ExecStart=/bin/touch /run/cloud-init/network-config-ready
RemainAfterExit=yes
TimeoutSec=0

# Output needs to appear in instance console output
StandardOutput=journal+console

[Install]
WantedBy=cloud-init.target
```

### Configure cloud-init for autostart

```
systemctl daemon-reload
systemctl enable cloud-init.service
systemctl enable cloud-init-local.service
systemctl enable cloud-final.service
systemctl enable cloud-config.service
```

### Use the following cloud.cfg file

- Keep this cloud.cfg file in /etc/cloud/

```
# The top level settings are used as module
# and system configuration.

# A set of users which may be applied and/or used by various modules
# when a 'default' entry is found it will reference the 'default_user'
# from the distro configuration specified below
users:
  - default

# If this is set, 'root' will not be able to ssh in and they
# will get a message to login instead as the above $user (ubuntu)
disable_root: false

# This will cause the set+update hostname module to not operate (if true)
preserve_hostname: false

#datasource_list: [ ConfigDrive, None ]

# Example datasource config
# datasource:
#   Ec2:
#     metadata_urls: [ 'blah.com' ]
#     timeout: 5 # (defaults to 50 seconds)
#     max_wait: 10 # (defaults to 120 seconds)

# The modules that run in the 'init' stage
cloud_init_modules:
  - migrator
```

```
# - ubuntu-init-switch
- seed_random
- bootcmd
- write-files
- growpart
- resizefs
- disk_setup
- mounts
- set_hostname
- update_hostname
- update_etc_hosts
- ca-certs
- rsyslog
- users-groups
- ssh

# The modules that run in the 'config' stage
cloud_config_modules:
# Emit the cloud config ready event
# this can be used by upstart jobs for 'start on cloud-config'.
- emit_upstart
- snap_config
- ssh-import-id
- locale
- set-passwords
# - grub-dpkg
# - apt-pipelining
# - apt-configure
- ntp
- timezone
- disable-ec2-metadata
- runcmd
- byobu

# The modules that run in the 'final' stage
cloud_final_modules:
- snappy
- package-update-upgrade-install
- fan
- landscape
- lxd
- puppet
- chef
- salt-minion
- mcollective
- rightscale_userdata
- scripts-vendor
- scripts-per-once
- scripts-per-boot
- scripts-per-instance
- scripts-user
- ssh-authkey-fingerprints
- keys-to-console
- phone-home
- final-message
- power-state-change

# System and/or distro specific settings
```

```
# (not accessible to handlers/transforms)
system_info:
  # This will affect which distro class gets used
  distro: rhel
```

## Test-It

Run it once to see if things are working

```
cloud-init -init
```

---

**Note:** This might take a few minutes, as cloud-init tries to access various network datasources, which probably are not available in your image build environment. But they should be available in your OpenStack cloud. For debugging you might need to set “datasource\_list: [ ConfigDrive, None ]” in cloud.cfg. This excludes those network data sources and boot is pretty fast.

---

## Add DPM-Guest Tools

- Install *git* and clone *nova-dpm* repository into the guest image.

```
git clone https://github.com/openstack/nova-dpm.git
```

- Copy the following files from *nova-dpm* directory into the guest image

```
cp nova-dpm/dpm_guest_tools/usr/bin/autoconfigure_networking.sh /usr/bin/autoconfigure_networking.sh
cp nova-dpm/dpm_guest_tools/usr/lib/systemd/system/autoconfigure_networking.service /usr/lib/systemd/system/autoconfigure_networking.service
cp nova-dpm/dpm_guest_tools/usr/bin/setmac.sh /usr/bin/setmac.sh
cp nova-dpm/dpm_guest_tools/etc/udev/rules.d/80-setmac.rules /etc/udev/rules.d/80-setmac.rules
```

- Ensure permissions

```
chmod 644 /usr/lib/systemd/system/autoconfigure_networking.service
```

- Enable the service for autostart

```
systemctl enable autoconfigure_networking.service
```

## Cleanup

- Cleanup logs and journalctl

```
rm -rf /var/log/*
```

- Remove repo file and update repo

```
rm -f /etc/yum.repos.d/rhel.repo
```

```
yum clean all
```

```
yum update
```

```
yum repolist
```

- Remove data from last cloud-init run

```
rm -rf /var/lib/cloud/*
```

- Remove persistent mac address interface mappings

```
rm -f /etc/udev/rules.d/70-persistent-net.rules
```

- Remove persistent network configs

```
rm -f /etc/sysconfig/network-scripts/ifcfg-enc*
```

- Clear /etc/hostname

```
echo "" > /etc/hostname
```

- Cleanup home directory

```
rm -rf ~/*
```

## Create qcow2 image

- In order to nullify space

```
dd if=/dev/zero of=~/.tmpfile
```

```
rm -rf ~/.tmpfile
```

- Now stop the partition and access the LUN used for image creation from other machine

- copy disk content byte-by-byte into a raw image

```
dd status=progress if=/path/to/installed/lun of=RHEL.img
```

- Convert this raw image to qcow

```
qemu-img convert -f raw -O qcow2 RHEL.img RHEL.qcow
```

## Test qcow2 image

- Deploy this image on another LUN

```
qemu-img convert RHEL.qcow /path/to/new/lun
```

- Use this new LUN to boot the machine



---

## Contributing to the project

---

### Contributing

If you would like to contribute to the development of the nova-dpm project, you must follow the rules for OpenStack contributions described in the “If you’re a developer, start here” section of this page:

<http://wiki.openstack.org/HowToContribute>

Once those steps have been completed, changes to the nova-dpm project should be submitted for review via the Gerrit tool, following the workflow documented at:

<http://wiki.openstack.org/GerritWorkflow>

Pull requests submitted through GitHub will be ignored.

The Git repository for the nova-dpm project is here:

<http://git.openstack.org/cgit/openstack/nova-dpm>

Bugs against the nova-dpm project should be filed on Launchpad (not on GitHub):

<https://bugs.launchpad.net/nova-dpm>

Pending changes for the nova-dpm project can be seen on its Gerrit page:

<https://review.openstack.org/#/q/project:openstack/nova-dpm>

### Developer Guide

#### Release Notes

##### Guidelines

- Release note files **MUST** be part of the code changes which introduce the noteworthy behavior change. Noteworthy behavior changes are:
  - a deprecation of a config option
  - a change of the default value of a config option
  - the removal of a config option
  - upgrade relevant actions (e.g. new required config options)
  - security fixes

- When important bug fixes or features are done, release note files COULD be part of those code changes.

## How-To

To create a new release note:

```
$ reno --rel-notes-dir=doc/source/releasenotes/ new file-name-goes-here
```

To list existing release notes:

```
$ reno --rel-notes-dir=doc/source/releasenotes/ list .
```

To build the release notes:

```
$ tox -e docs
```

---

**Note:** If you build the release notes locally, please be aware that *reno* only scans release note files (*\*.yaml*) which are committed in your local repository of this project.

---

More information about *reno* can be found at: <http://docs.openstack.org/developer/reno/index.html>

## Nova-dpm specifications for Ocata

**TODO:** Figure out what to do with this page and its child pages.

Template:

### Example Spec - The title of your blueprint

Include the URL of your launchpad blueprint:

<https://blueprints.launchpad.net/nova/+spec/example>

Introduction paragraph – why are we doing anything? A single paragraph of prose that operators can understand. The title and this first paragraph should be used as the subject line and body of the commit message respectively.

Some notes about the nova-spec and blueprint process:

- Not all blueprints need a spec. For more information see <http://docs.openstack.org/developer/nova/devref/kilo.blueprints.html#what-is-a-blueprint-needed>
- The aim of this document is first to define the problem we need to solve, and second agree the overall approach to solve that problem.
- This is not intended to be extensive documentation for a new feature. For example, there is no need to specify the exact configuration changes, nor the exact details of any DB model changes. But you should still define that such changes are required, and be clear on how that will affect upgrades.
- You should aim to get your spec approved before writing your code. While you are free to write prototypes and code before getting your spec approved, its possible that the outcome of the spec review process leads you towards a fundamentally different solution than you first envisaged.

- But, API changes are held to a much higher level of scrutiny. As soon as an API change merges, we must assume it could be in production somewhere, and as such, we then need to support that API change forever. To avoid getting that wrong, we do want lots of details about API changes upfront.

Some notes about using this template:

- Your spec should be in ReSTructured text, like this template.
- Please wrap text at 79 columns.
- The filename in the git repository should match the launchpad URL, for example a URL of: <https://blueprints.launchpad.net/nova/+spec/awesome-thing> should be named awesome-thing.rst
- Please do not delete any of the sections in this template. If you have nothing to say for a whole section, just write: None
- For help with syntax, see <http://sphinx-doc.org/rest.html>
- To test out your formatting, build the docs using tox and see the generated HTML file in `doc/build/html/specs/<path_of_your_file>`
- If you would like to provide a diagram with your spec, ascii diagrams are required. <http://asciiflow.com/> is a very nice tool to assist with making ascii diagrams. The reason for this is that the tool used to review specs is based purely on plain text. Plain text will allow review to proceed without having to look at additional files which can not be viewed in gerrit. It will also allow inline feedback on the diagram itself.
- If your specification proposes any changes to the Nova REST API such as changing parameters which can be returned or accepted, or even the semantics of what happens when a client calls into the API, then you should add the APIImpact flag to the commit message. Specifications with the APIImpact flag can be found with the following query:

<https://review.openstack.org/#/q/status:open+project:openstack/nova-specs+message:apiimpact,n,z>

## Problem description

A detailed description of the problem. What problem is this blueprint addressing?

## Use Cases

What use cases does this address? What impact on actors does this change have? Ensure you are clear about the actors in each use case: Developer, End User, Deployer etc.

## Proposed change

Here is where you cover the change you propose to make in detail. How do you propose to solve this problem?

If this is one part of a larger effort make it clear where this piece ends. In other words, what's the scope of this effort?

At this point, if you would like to just get feedback on if the problem and proposed change fit in nova, you can stop here and post this for review to get preliminary feedback. If so please say: Posting to get preliminary feedback on the scope of this spec.

## Alternatives

What other ways could we do this thing? Why aren't we using those? This doesn't have to be a full literature review, but it should demonstrate that thought has been put into why the proposed solution is an appropriate one.

## Data model impact

Changes which require modifications to the data model often have a wider impact on the system. The community often has strong opinions on how the data model should be evolved, from both a functional and performance perspective. It is therefore important to capture and gain agreement as early as possible on any proposed changes to the data model.

Questions which need to be addressed by this section include:

- What new data objects and/or database schema changes is this going to require?
- What database migrations will accompany this change.
- How will the initial set of new data objects be generated, for example if you need to take into account existing instances, or modify other existing data describe how that will work.

## REST API impact

Each API method which is either added or changed should have the following

- Specification for the method
  - A description of what the method does suitable for use in user documentation
  - Method type (POST/PUT/GET/DELETE)
  - Normal http response code(s)
  - Expected error http response code(s)
    - \* A description for each possible error code should be included describing semantic errors which can cause it such as inconsistent parameters supplied to the method, or when an instance is not in an appropriate state for the request to succeed. Errors caused by syntactic problems covered by the JSON schema definition do not need to be included.
  - URL for the resource
    - \* URL should not include underscores, and use hyphens instead.
  - Parameters which can be passed via the url
  - JSON schema definition for the request body data if allowed
    - \* Field names should use snake\_case style, not CamelCase or MixedCase style.
  - JSON schema definition for the response body data if any
    - \* Field names should use snake\_case style, not CamelCase or MixedCase style.
- Example use case including typical API samples for both data supplied by the caller and the response
- Discuss any policy changes, and discuss what things a deployer needs to think about when defining their policy.

Example JSON schema definitions can be found in the Nova tree <http://git.openstack.org/cgit/openstack/nova/tree/nova/api/openstack/com>

Note that the schema should be defined as restrictively as possible. Parameters which are required should be marked as such and only under exceptional circumstances should additional parameters which are not defined in the schema be permitted (eg additionalProperties should be False).

Reuse of existing predefined parameter types such as regexps for passwords and user defined names is highly encouraged.

### Security impact

Describe any potential security impact on the system. Some of the items to consider include:

- Does this change touch sensitive data such as tokens, keys, or user data?
- Does this change alter the API in a way that may impact security, such as a new way to access sensitive information or a new way to login?
- Does this change involve cryptography or hashing?
- Does this change require the use of sudo or any elevated privileges?
- Does this change involve using or parsing user-provided data? This could be directly at the API level or indirectly such as changes to a cache layer.
- Can this change enable a resource exhaustion attack, such as allowing a single API interaction to consume significant server resources? Some examples of this include launching subprocesses for each connection, or entity expansion attacks in XML.

For more detailed guidance, please see the OpenStack Security Guidelines as a reference (<https://wiki.openstack.org/wiki/Security/Guidelines>). These guidelines are a work in progress and are designed to help you identify security best practices. For further information, feel free to reach out to the OpenStack Security Group at [openstack-security@lists.openstack.org](mailto:openstack-security@lists.openstack.org).

### Notifications impact

Please specify any changes to notifications. Be that an extra notification, changes to an existing notification, or removing a notification.

### Other end user impact

Aside from the API, are there other ways a user will interact with this feature?

- Does this change have an impact on python-novaclient? What does the user interface there look like?

### Performance Impact

Describe any potential performance impact on the system, for example how often will new code be called, and is there a major change to the calling pattern of existing code.

Examples of things to consider here include:

- A periodic task might look like a small addition but if it calls conductor or another service the load is multiplied by the number of nodes in the system.
- Scheduler filters get called once per host for every instance being created, so any latency they introduce is linear with the size of the system.
- A small change in a utility function or a commonly used decorator can have a large impacts on performance.
- Calls which result in a database queries (whether direct or via conductor) can have a profound impact on performance when called in critical sections of the code.
- Will the change include any locking, and if so what considerations are there on holding the lock?

### Other deployer impact

Discuss things that will affect how you deploy and configure OpenStack that have not already been mentioned, such as:

- What config options are being added? Should they be more generic than proposed (for example a flag that other hypervisor drivers might want to implement as well)? Are the default values ones which will work well in real deployments?
- Is this a change that takes immediate effect after its merged, or is it something that has to be explicitly enabled?
- If this change is a new binary, how would it be deployed?
- Please state anything that those doing continuous deployment, or those upgrading from the previous release, need to be aware of. Also describe any plans to deprecate configuration values or features. For example, if we change the directory name that instances are stored in, how do we handle instance directories created before the change landed? Do we move them? Do we have a special case in the code? Do we assume that the operator will recreate all the instances in their cloud?

### Developer impact

Discuss things that will affect other developers working on OpenStack, such as:

- If the blueprint proposes a change to the driver API, discussion of how other hypervisors would implement the feature is required.

### Implementation

#### Assignee(s)

Who is leading the writing of the code? Or is this a blueprint where you're throwing it out there to see who picks it up?

If more than one person is working on the implementation, please designate the primary author and contact.

**Primary assignee:** <launchpad-id or None>

**Other contributors:** <launchpad-id or None>

#### Work Items

Work items or tasks – break the feature up into the things that need to be done to implement it. Those parts might end up being done by different people, but we're mostly trying to understand the timeline for implementation.

### Dependencies

- Include specific references to specs and/or blueprints in nova, or in other projects, that this one either depends on or is related to.
- If this requires functionality of another project that is not currently used by Nova (such as the glance v2 API when we previously only required v1), document that fact.
- Does this feature require any new library dependencies or code otherwise not included in OpenStack? Or does it depend on a specific version of library?

## Testing

Please discuss the important scenarios needed to test here, as well as specific edge cases we should be ensuring work correctly. For each scenario please specify if this requires specialized hardware, a full openstack environment, or can be simulated inside the Nova tree.

Please discuss how the change will be tested. We especially want to know what tempest tests will be added. It is assumed that unit test coverage will be added so that doesn't need to be mentioned explicitly, but discussion of why you think unit tests are sufficient and we don't need to add more tempest tests would need to be included.

Is this untestable in gate given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans (3rd party testing, gate enhancements, etc).

## Documentation Impact

Which audiences are affected most by this change, and which documentation titles on docs.openstack.org should be updated because of this change? Don't repeat details discussed above, but reference them here in the context of documentation for multiple audiences. For example, the Operations Guide targets cloud operators, and the End User Guide would need to be updated if the change offers a new feature available through the CLI or dashboard. If a config option changes or is deprecated, note here that the documentation needs to be updated to reflect this specification's change.

## References

Please add any useful references here. You are not required to have any reference. Moreover, this specification should still make sense when your references are unavailable. Examples of what you could include are:

- Links to mailing list or IRC discussions
- Links to notes from a summit session
- Links to relevant research, if appropriate
- Related specifications as appropriate (e.g. if it's an EC2 thing, link the EC2 docs)
- Anything else you feel it is worthwhile to refer to

## History

Optional section intended to be used each time the spec is updated to describe new design, API or any database schema updated. Useful to let reader understand what's happened along the time.

Table 4.1: Revisions

Release Name	Description
Ocata	Introduced

Specs implemented in Ocata:

## Example Spec - The title of your blueprint

Include the URL of your launchpad blueprint:

<https://blueprints.launchpad.net/nova/+spec/example>

Introduction paragraph – why are we doing anything? A single paragraph of prose that operators can understand. The title and this first paragraph should be used as the subject line and body of the commit message respectively.

Some notes about the nova-spec and blueprint process:

- Not all blueprints need a spec. For more information see <http://docs.openstack.org/developer/nova/devref/kilo.blueprints.html#what-is-a-blueprint-needed>
- The aim of this document is first to define the problem we need to solve, and second agree the overall approach to solve that problem.
- This is not intended to be extensive documentation for a new feature. For example, there is no need to specify the exact configuration changes, nor the exact details of any DB model changes. But you should still define that such changes are required, and be clear on how that will affect upgrades.
- You should aim to get your spec approved before writing your code. While you are free to write prototypes and code before getting your spec approved, its possible that the outcome of the spec review process leads you towards a fundamentally different solution than you first envisaged.
- But, API changes are held to a much higher level of scrutiny. As soon as an API change merges, we must assume it could be in production somewhere, and as such, we then need to support that API change forever. To avoid getting that wrong, we do want lots of details about API changes upfront.

Some notes about using this template:

- Your spec should be in ReSTructured text, like this template.
- Please wrap text at 79 columns.
- The filename in the git repository should match the launchpad URL, for example a URL of: <https://blueprints.launchpad.net/nova/+spec/awesome-thing> should be named awesome-thing.rst
- Please do not delete any of the sections in this template. If you have nothing to say for a whole section, just write: None
- For help with syntax, see <http://sphinx-doc.org/rest.html>
- To test out your formatting, build the docs using tox and see the generated HTML file in `doc/build/html/specs/<path_of_your_file>`
- If you would like to provide a diagram with your spec, ascii diagrams are required. <http://asciiflow.com/> is a very nice tool to assist with making ascii diagrams. The reason for this is that the tool used to review specs is based purely on plain text. Plain text will allow review to proceed without having to look at additional files which can not be viewed in gerrit. It will also allow inline feedback on the diagram itself.
- If your specification proposes any changes to the Nova REST API such as changing parameters which can be returned or accepted, or even the semantics of what happens when a client calls into the API, then you should add the APIImpact flag to the commit message. Specifications with the APIImpact flag can be found with the following query:  
<https://review.openstack.org/#/q/status:open+project:openstack/nova-specs+message:apiimpact,n,z>

### Problem description

A detailed description of the problem. What problem is this blueprint addressing?

## Use Cases

What use cases does this address? What impact on actors does this change have? Ensure you are clear about the actors in each use case: Developer, End User, Deployer etc.

## Proposed change

Here is where you cover the change you propose to make in detail. How do you propose to solve this problem?

If this is one part of a larger effort make it clear where this piece ends. In other words, what's the scope of this effort?

At this point, if you would like to just get feedback on if the problem and proposed change fit in nova, you can stop here and post this for review to get preliminary feedback. If so please say: Posting to get preliminary feedback on the scope of this spec.

## Alternatives

What other ways could we do this thing? Why aren't we using those? This doesn't have to be a full literature review, but it should demonstrate that thought has been put into why the proposed solution is an appropriate one.

## Data model impact

Changes which require modifications to the data model often have a wider impact on the system. The community often has strong opinions on how the data model should be evolved, from both a functional and performance perspective. It is therefore important to capture and gain agreement as early as possible on any proposed changes to the data model.

Questions which need to be addressed by this section include:

- What new data objects and/or database schema changes is this going to require?
- What database migrations will accompany this change.
- How will the initial set of new data objects be generated, for example if you need to take into account existing instances, or modify other existing data describe how that will work.

## REST API impact

Each API method which is either added or changed should have the following

- Specification for the method
  - A description of what the method does suitable for use in user documentation
  - Method type (POST/PUT/GET/DELETE)
  - Normal http response code(s)
  - Expected error http response code(s)
    - \* A description for each possible error code should be included describing semantic errors which can cause it such as inconsistent parameters supplied to the method, or when an instance is not in an appropriate state for the request to succeed. Errors caused by syntactic problems covered by the JSON schema definition do not need to be included.
  - URL for the resource
    - \* URL should not include underscores, and use hyphens instead.

- Parameters which can be passed via the url
- JSON schema definition for the request body data if allowed
  - \* Field names should use snake\_case style, not CamelCase or MixedCase style.
- JSON schema definition for the response body data if any
  - \* Field names should use snake\_case style, not CamelCase or MixedCase style.
- Example use case including typical API samples for both data supplied by the caller and the response
- Discuss any policy changes, and discuss what things a deployer needs to think about when defining their policy.

Example JSON schema definitions can be found in the Nova tree <http://git.openstack.org/cgit/openstack/nova/tree/nova/api/openstack/com>

Note that the schema should be defined as restrictively as possible. Parameters which are required should be marked as such and only under exceptional circumstances should additional parameters which are not defined in the schema be permitted (eg `additionalProperties` should be `False`).

Reuse of existing predefined parameter types such as regexps for passwords and user defined names is highly encouraged.

### Security impact

Describe any potential security impact on the system. Some of the items to consider include:

- Does this change touch sensitive data such as tokens, keys, or user data?
- Does this change alter the API in a way that may impact security, such as a new way to access sensitive information or a new way to login?
- Does this change involve cryptography or hashing?
- Does this change require the use of sudo or any elevated privileges?
- Does this change involve using or parsing user-provided data? This could be directly at the API level or indirectly such as changes to a cache layer.
- Can this change enable a resource exhaustion attack, such as allowing a single API interaction to consume significant server resources? Some examples of this include launching subprocesses for each connection, or entity expansion attacks in XML.

For more detailed guidance, please see the OpenStack Security Guidelines as a reference (<https://wiki.openstack.org/wiki/Security/Guidelines>). These guidelines are a work in progress and are designed to help you identify security best practices. For further information, feel free to reach out to the OpenStack Security Group at [openstack-security@lists.openstack.org](mailto:openstack-security@lists.openstack.org).

### Notifications impact

Please specify any changes to notifications. Be that an extra notification, changes to an existing notification, or removing a notification.

### Other end user impact

Aside from the API, are there other ways a user will interact with this feature?

- Does this change have an impact on python-novaclient? What does the user interface there look like?

### Performance Impact

Describe any potential performance impact on the system, for example how often will new code be called, and is there a major change to the calling pattern of existing code.

Examples of things to consider here include:

- A periodic task might look like a small addition but if it calls conductor or another service the load is multiplied by the number of nodes in the system.
- Scheduler filters get called once per host for every instance being created, so any latency they introduce is linear with the size of the system.
- A small change in a utility function or a commonly used decorator can have a large impacts on performance.
- Calls which result in a database queries (whether direct or via conductor) can have a profound impact on performance when called in critical sections of the code.
- Will the change include any locking, and if so what considerations are there on holding the lock?

### Other deployer impact

Discuss things that will affect how you deploy and configure OpenStack that have not already been mentioned, such as:

- What config options are being added? Should they be more generic than proposed (for example a flag that other hypervisor drivers might want to implement as well)? Are the default values ones which will work well in real deployments?
- Is this a change that takes immediate effect after its merged, or is it something that has to be explicitly enabled?
- If this change is a new binary, how would it be deployed?
- Please state anything that those doing continuous deployment, or those upgrading from the previous release, need to be aware of. Also describe any plans to deprecate configuration values or features. For example, if we change the directory name that instances are stored in, how do we handle instance directories created before the change landed? Do we move them? Do we have a special case in the code? Do we assume that the operator will recreate all the instances in their cloud?

### Developer impact

Discuss things that will affect other developers working on OpenStack, such as:

- If the blueprint proposes a change to the driver API, discussion of how other hypervisors would implement the feature is required.

### Implementation

#### Assignee(s)

Who is leading the writing of the code? Or is this a blueprint where you're throwing it out there to see who picks it up?

If more than one person is working on the implementation, please designate the primary author and contact.

**Primary assignee:** <launchpad-id or None>

**Other contributors:** <launchpad-id or None>

### Work Items

Work items or tasks – break the feature up into the things that need to be done to implement it. Those parts might end up being done by different people, but we’re mostly trying to understand the timeline for implementation.

### Dependencies

- Include specific references to specs and/or blueprints in nova, or in other projects, that this one either depends on or is related to.
- If this requires functionality of another project that is not currently used by Nova (such as the glance v2 API when we previously only required v1), document that fact.
- Does this feature require any new library dependencies or code otherwise not included in OpenStack? Or does it depend on a specific version of library?

### Testing

Please discuss the important scenarios needed to test here, as well as specific edge cases we should be ensuring work correctly. For each scenario please specify if this requires specialized hardware, a full openstack environment, or can be simulated inside the Nova tree.

Please discuss how the change will be tested. We especially want to know what tempest tests will be added. It is assumed that unit test coverage will be added so that doesn’t need to be mentioned explicitly, but discussion of why you think unit tests are sufficient and we don’t need to add more tempest tests would need to be included.

Is this untestable in gate given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans (3rd party testing, gate enhancements, etc).

### Documentation Impact

Which audiences are affected most by this change, and which documentation titles on docs.openstack.org should be updated because of this change? Don’t repeat details discussed above, but reference them here in the context of documentation for multiple audiences. For example, the Operations Guide targets cloud operators, and the End User Guide would need to be updated if the change offers a new feature available through the CLI or dashboard. If a config option changes or is deprecated, note here that the documentation needs to be updated to reflect this specification’s change.

### References

Please add any useful references here. You are not required to have any reference. Moreover, this specification should still make sense when your references are unavailable. Examples of what you could include are:

- Links to mailing list or IRC discussions
- Links to notes from a summit session
- Links to relevant research, if appropriate
- Related specifications as appropriate (e.g. if it’s an EC2 thing, link the EC2 docs)
- Anything else you feel it is worthwhile to refer to

## History

Optional section intended to be used each time the spec is updated to describe new design, API or any database schema updated. Useful to let reader understand what's happened along the time.

Table 4.2: Revisions

Release Name	Description
Ocata	Introduced

Specs approved (but not implemented) in Ocata:

## Example Spec - The title of your blueprint

Include the URL of your launchpad blueprint:

<https://blueprints.launchpad.net/nova/+spec/example>

Introduction paragraph – why are we doing anything? A single paragraph of prose that operators can understand. The title and this first paragraph should be used as the subject line and body of the commit message respectively.

Some notes about the nova-spec and blueprint process:

- Not all blueprints need a spec. For more information see <http://docs.openstack.org/developer/nova/devref/kilo.blueprints.html#what-is-a-blueprint-needed>
- The aim of this document is first to define the problem we need to solve, and second agree the overall approach to solve that problem.
- This is not intended to be extensive documentation for a new feature. For example, there is no need to specify the exact configuration changes, nor the exact details of any DB model changes. But you should still define that such changes are required, and be clear on how that will affect upgrades.
- You should aim to get your spec approved before writing your code. While you are free to write prototypes and code before getting your spec approved, its possible that the outcome of the spec review process leads you towards a fundamentally different solution than you first envisaged.
- But, API changes are held to a much higher level of scrutiny. As soon as an API change merges, we must assume it could be in production somewhere, and as such, we then need to support that API change forever. To avoid getting that wrong, we do want lots of details about API changes upfront.

Some notes about using this template:

- Your spec should be in ReSTructured text, like this template.
- Please wrap text at 79 columns.
- The filename in the git repository should match the launchpad URL, for example a URL of: <https://blueprints.launchpad.net/nova/+spec/awesome-thing> should be named awesome-thing.rst
- Please do not delete any of the sections in this template. If you have nothing to say for a whole section, just write: None
- For help with syntax, see <http://sphinx-doc.org/rest.html>
- To test out your formatting, build the docs using tox and see the generated HTML file in doc/build/html/specs/<path\_of\_your\_file>
- If you would like to provide a diagram with your spec, ascii diagrams are required. <http://asciiflow.com/> is a very nice tool to assist with making ascii diagrams. The reason for this is that the tool used to review specs

is based purely on plain text. Plain text will allow review to proceed without having to look at additional files which can not be viewed in gerrit. It will also allow inline feedback on the diagram itself.

- If your specification proposes any changes to the Nova REST API such as changing parameters which can be returned or accepted, or even the semantics of what happens when a client calls into the API, then you should add the APIImpact flag to the commit message. Specifications with the APIImpact flag can be found with the following query:

<https://review.openstack.org/#/q/status:open+project:openstack/nova-specs+message:apiimpact,n,z>

## Problem description

A detailed description of the problem. What problem is this blueprint addressing?

## Use Cases

What use cases does this address? What impact on actors does this change have? Ensure you are clear about the actors in each use case: Developer, End User, Deployer etc.

## Proposed change

Here is where you cover the change you propose to make in detail. How do you propose to solve this problem?

If this is one part of a larger effort make it clear where this piece ends. In other words, what's the scope of this effort?

At this point, if you would like to just get feedback on if the problem and proposed change fit in nova, you can stop here and post this for review to get preliminary feedback. If so please say: Posting to get preliminary feedback on the scope of this spec.

## Alternatives

What other ways could we do this thing? Why aren't we using those? This doesn't have to be a full literature review, but it should demonstrate that thought has been put into why the proposed solution is an appropriate one.

## Data model impact

Changes which require modifications to the data model often have a wider impact on the system. The community often has strong opinions on how the data model should be evolved, from both a functional and performance perspective. It is therefore important to capture and gain agreement as early as possible on any proposed changes to the data model.

Questions which need to be addressed by this section include:

- What new data objects and/or database schema changes is this going to require?
- What database migrations will accompany this change.
- How will the initial set of new data objects be generated, for example if you need to take into account existing instances, or modify other existing data describe how that will work.

## REST API impact

Each API method which is either added or changed should have the following

- Specification for the method
  - A description of what the method does suitable for use in user documentation
  - Method type (POST/PUT/GET/DELETE)
  - Normal http response code(s)
  - Expected error http response code(s)
    - \* A description for each possible error code should be included describing semantic errors which can cause it such as inconsistent parameters supplied to the method, or when an instance is not in an appropriate state for the request to succeed. Errors caused by syntactic problems covered by the JSON schema definition do not need to be included.
  - URL for the resource
    - \* URL should not include underscores, and use hyphens instead.
  - Parameters which can be passed via the url
  - JSON schema definition for the request body data if allowed
    - \* Field names should use snake\_case style, not CamelCase or MixedCase style.
  - JSON schema definition for the response body data if any
    - \* Field names should use snake\_case style, not CamelCase or MixedCase style.
- Example use case including typical API samples for both data supplied by the caller and the response
- Discuss any policy changes, and discuss what things a deployer needs to think about when defining their policy.

Example JSON schema definitions can be found in the Nova tree <http://git.openstack.org/cgit/openstack/nova/tree/nova/api/openstack/com>

Note that the schema should be defined as restrictively as possible. Parameters which are required should be marked as such and only under exceptional circumstances should additional parameters which are not defined in the schema be permitted (eg additionalProperties should be False).

Reuse of existing predefined parameter types such as regexps for passwords and user defined names is highly encouraged.

## Security impact

Describe any potential security impact on the system. Some of the items to consider include:

- Does this change touch sensitive data such as tokens, keys, or user data?
- Does this change alter the API in a way that may impact security, such as a new way to access sensitive information or a new way to login?
- Does this change involve cryptography or hashing?
- Does this change require the use of sudo or any elevated privileges?
- Does this change involve using or parsing user-provided data? This could be directly at the API level or indirectly such as changes to a cache layer.
- Can this change enable a resource exhaustion attack, such as allowing a single API interaction to consume significant server resources? Some examples of this include launching subprocesses for each connection, or entity expansion attacks in XML.

For more detailed guidance, please see the OpenStack Security Guidelines as a reference (<https://wiki.openstack.org/wiki/Security/Guidelines>). These guidelines are a work in progress and are designed to help you identify security best practices. For further information, feel free to reach out to the OpenStack Security Group at [openstack-security@lists.openstack.org](mailto:openstack-security@lists.openstack.org).

### **Notifications impact**

Please specify any changes to notifications. Be that an extra notification, changes to an existing notification, or removing a notification.

### **Other end user impact**

Aside from the API, are there other ways a user will interact with this feature?

- Does this change have an impact on python-novaclient? What does the user interface there look like?

### **Performance Impact**

Describe any potential performance impact on the system, for example how often will new code be called, and is there a major change to the calling pattern of existing code.

Examples of things to consider here include:

- A periodic task might look like a small addition but if it calls conductor or another service the load is multiplied by the number of nodes in the system.
- Scheduler filters get called once per host for every instance being created, so any latency they introduce is linear with the size of the system.
- A small change in a utility function or a commonly used decorator can have a large impacts on performance.
- Calls which result in a database queries (whether direct or via conductor) can have a profound impact on performance when called in critical sections of the code.
- Will the change include any locking, and if so what considerations are there on holding the lock?

### **Other deployer impact**

Discuss things that will affect how you deploy and configure OpenStack that have not already been mentioned, such as:

- What config options are being added? Should they be more generic than proposed (for example a flag that other hypervisor drivers might want to implement as well)? Are the default values ones which will work well in real deployments?
- Is this a change that takes immediate effect after its merged, or is it something that has to be explicitly enabled?
- If this change is a new binary, how would it be deployed?
- Please state anything that those doing continuous deployment, or those upgrading from the previous release, need to be aware of. Also describe any plans to deprecate configuration values or features. For example, if we change the directory name that instances are stored in, how do we handle instance directories created before the change landed? Do we move them? Do we have a special case in the code? Do we assume that the operator will recreate all the instances in their cloud?

## Developer impact

Discuss things that will affect other developers working on OpenStack, such as:

- If the blueprint proposes a change to the driver API, discussion of how other hypervisors would implement the feature is required.

## Implementation

### Assignee(s)

Who is leading the writing of the code? Or is this a blueprint where you're throwing it out there to see who picks it up?

If more than one person is working on the implementation, please designate the primary author and contact.

**Primary assignee:** <launchpad-id or None>

**Other contributors:** <launchpad-id or None>

### Work Items

Work items or tasks – break the feature up into the things that need to be done to implement it. Those parts might end up being done by different people, but we're mostly trying to understand the timeline for implementation.

## Dependencies

- Include specific references to specs and/or blueprints in nova, or in other projects, that this one either depends on or is related to.
- If this requires functionality of another project that is not currently used by Nova (such as the glance v2 API when we previously only required v1), document that fact.
- Does this feature require any new library dependencies or code otherwise not included in OpenStack? Or does it depend on a specific version of library?

## Testing

Please discuss the important scenarios needed to test here, as well as specific edge cases we should be ensuring work correctly. For each scenario please specify if this requires specialized hardware, a full openstack environment, or can be simulated inside the Nova tree.

Please discuss how the change will be tested. We especially want to know what tempest tests will be added. It is assumed that unit test coverage will be added so that doesn't need to be mentioned explicitly, but discussion of why you think unit tests are sufficient and we don't need to add more tempest tests would need to be included.

Is this untestable in gate given current limitations (specific hardware / software configurations available)? If so, are there mitigation plans (3rd party testing, gate enhancements, etc).

## Documentation Impact

Which audiences are affected most by this change, and which documentation titles on docs.openstack.org should be updated because of this change? Don't repeat details discussed above, but reference them here in the context of documentation for multiple audiences. For example, the Operations Guide targets cloud operators, and the End User Guide would need to be updated if the change offers a new feature available through the CLI or dashboard. If a config option changes or is deprecated, note here that the documentation needs to be updated to reflect this specification's change.

## References

Please add any useful references here. You are not required to have any reference. Moreover, this specification should still make sense when your references are unavailable. Examples of what you could include are:

- Links to mailing list or IRC discussions
- Links to notes from a summit session
- Links to relevant research, if appropriate
- Related specifications as appropriate (e.g. if it's an EC2 thing, link the EC2 docs)
- Anything else you feel it is worthwhile to refer to

## History

Optional section intended to be used each time the spec is updated to describe new design, API or any database schema updated. Useful to let reader understand what's happened along the time.

Table 4.3: Revisions

Release Name	Description
Ocata	Introduced

---

### Links

---

- Documentation: <http://nova-dpm.readthedocs.io/en/latest/>
- Source: <http://git.openstack.org/cgit/openstack/nova-dpm>
- Github shadow: <https://github.com/openstack/nova-dpm>
- Bugs: <http://bugs.launchpad.net/nova-dpm>
- Gerrit: <https://review.openstack.org/#/q/project:openstack/nova-dpm>