nova-dpm Documentation

Release 3.0.1.dev9

OpenStack Foundation

Apr 10, 2018

Contents

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.

CHAPTER 1

Overview

1.1 Release Notes

1.1.1 2.0.0

Upgrade Notes

- The max_instances config parameter is removed as NumInstancesFilter is used inorder to filter the number of instances on each host.
- Minimum value of 512 has been set for the configuration option `max_memory`.This is the minimum memory needed by the host to launch an instance.

1.2 Topology

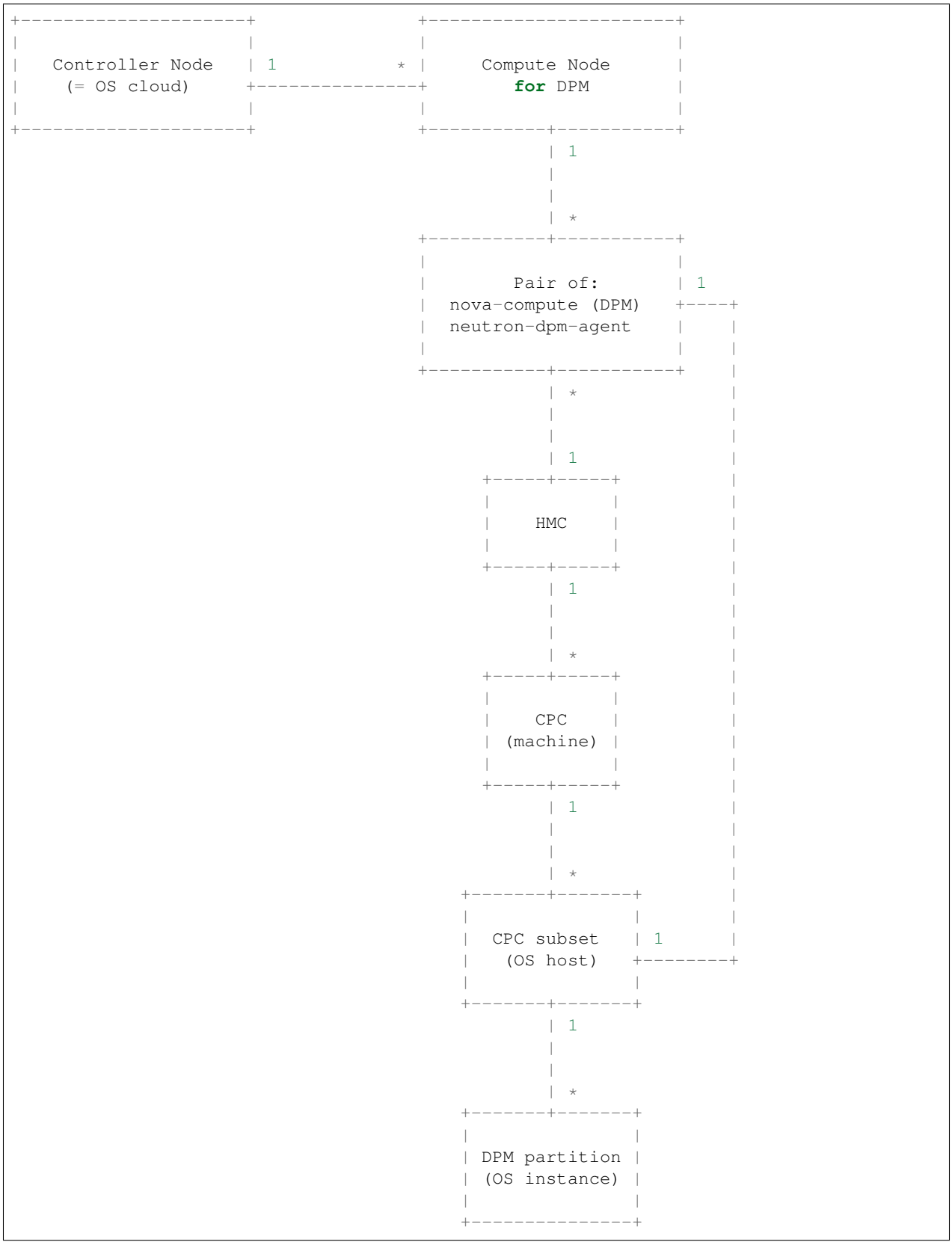
This section describes the topology between the OpenStack compute node for DPM, the z Systems Hardware Management Console (HMC) and the managed machines (CPCs).

1.2.1 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.
- A particular HMC can manage multiple CPCs. Therefore, there may be multiple pairs of nova-compute and neutron-dpm-agent services on possibly multiple compute nodes connecting to the same or different HMCs, for managing OpenStack hosts (CPC subsets) on the same or on different CPCs.
- Finally, the OpenStack host (CPC subset) powers the OpenStack instances (DPM partitions), like on any other OpenStack Nova compute platform.

1.2.2 General Topology

The general case is nearly like the case of a single OpenStack cloud, except that the compute nodes can now belong to different OpenStack clouds.

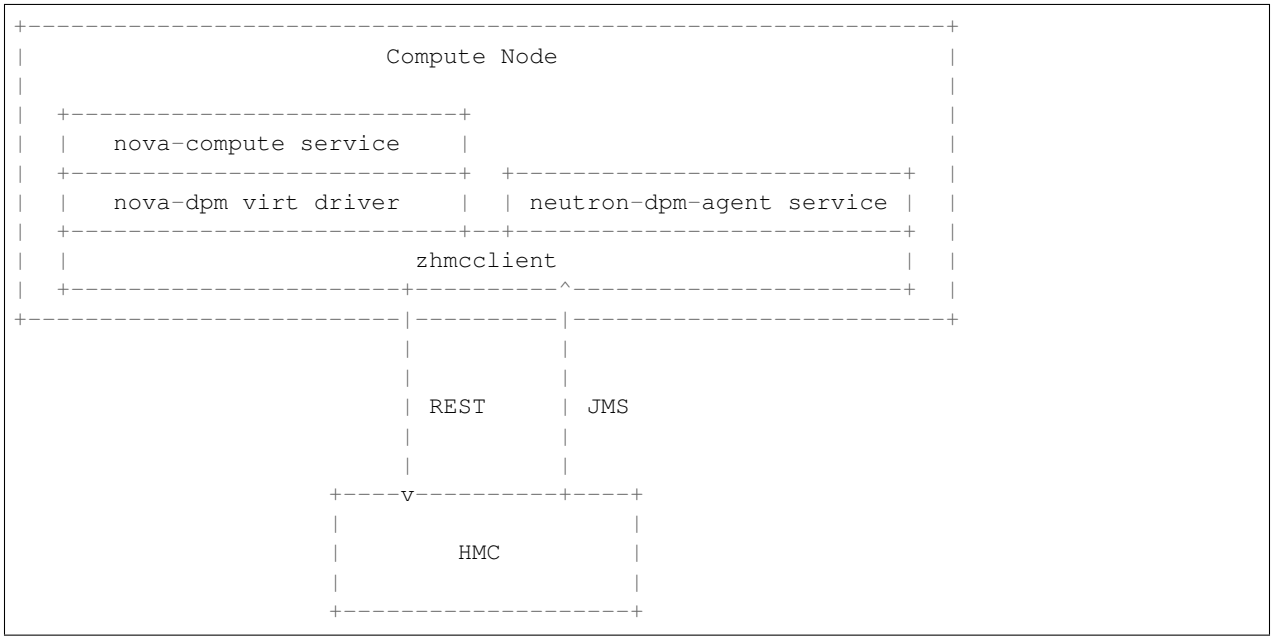
1.2.3 Interaction between OpenStack compute node and HMC

All interactions of OpenStack for DPM with an HMC go through a compute node for DPM. On the compute node, the nova-dpm virtualization driver within the nova-compute service and the neutron-dpm-agent service connect to the HMC. These are the only OpenStack components that interface with the HMC.

The HMC supports a Web Services API that uses REST over HTTPS for client-driven request/response style interactions, and JMS over STOMP for event-style notifications.

The *zhmcclient* Python package is used to isolate the OpenStack code from the details of the HMC Web Services API.

The following diagram shows how the OpenStack components on the compute node use the zhmcclient package to connect to the HMC:



1.3 Feature Support Matrix

Warning: Please note, while this document is still being maintained, this is slowly being updated to re-group and classify features.

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**
Attach block volume to instance	optional	
Detach block volume from instance	optional	
Attach virtual network interface to instance	optional	
Detach virtual network interface from instance	optional	
Set the host in a maintenance mode	optional	
Evacuate instances from a host	optional	
Rebuild instance	optional	
Guest instance status	mandatory	
Guest host status	optional	
Live migrate instance across hosts	optional	
Force live migration to complete	optional	
Launch instance	mandatory	
Stop instance CPUs (pause)	optional	
Reboot instance	optional	
Rescue instance	optional	
Resize instance	optional	
Restore instance	optional	
Service control	optional	
Set instance admin password	optional	
Save snapshot of instance disk	optional	
Suspend instance	optional	
Swap block volumes	optional	
Shutdown instance	mandatory	
Trigger crash dump	optional	
Resume instance CPUs (unpause)	optional	
Auto configure disk	optional	
Instance disk I/O limits	optional	
Config drive support	choice	
Inject files into disk image	optional	
Inject guest networking config	optional	
Remote desktop over RDP	choice	
View serial console logs	choice	
Remote interactive serial console	choice	
Remote desktop over SPICE	choice	
Remote desktop over VNC	choice	
Block storage support	optional	
Block storage over fibre channel	optional	
Block storage over iSCSI	condition	
CHAP authentication for iSCSI	optional	
Image storage support	mandatory	
Network firewall rules	optional	
Network routing	optional	
Network security groups	optional	
Flat networking	choice	
VLAN networking	choice	
uefi boot	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 cummulative 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

1.4 Storage Support

This section describes the storage setup for OpenStack DPM.

1.4.1 Supported Storage types

The following Storage types exist in OpenStack [1]:

- Ephemeral storage
  - Boot: Not supported by OpenStack DPM
  - Additional attachment: n/a
- Block storage
  - Boot: Only fibre channel protocol (FCP) block storage is supported by OpenStack DPM
  - Additional attachment: Only FCP block storage is supported by OpenStack DPM
- Object Storage
  - Boot: n/a
  - Additional attachment: Access happens via network - supported by OpenStack for DPM
- File Storage
  - Boot: n/a
  - Additional attachment: Access happens via network - supported by OpenStack for DPM

1.4.2 Block Storage setup

The Block Storage service (Cinder) must be configured to use a FCP back-end storage subsystem.

1.4.3 DPM FCP Architecture

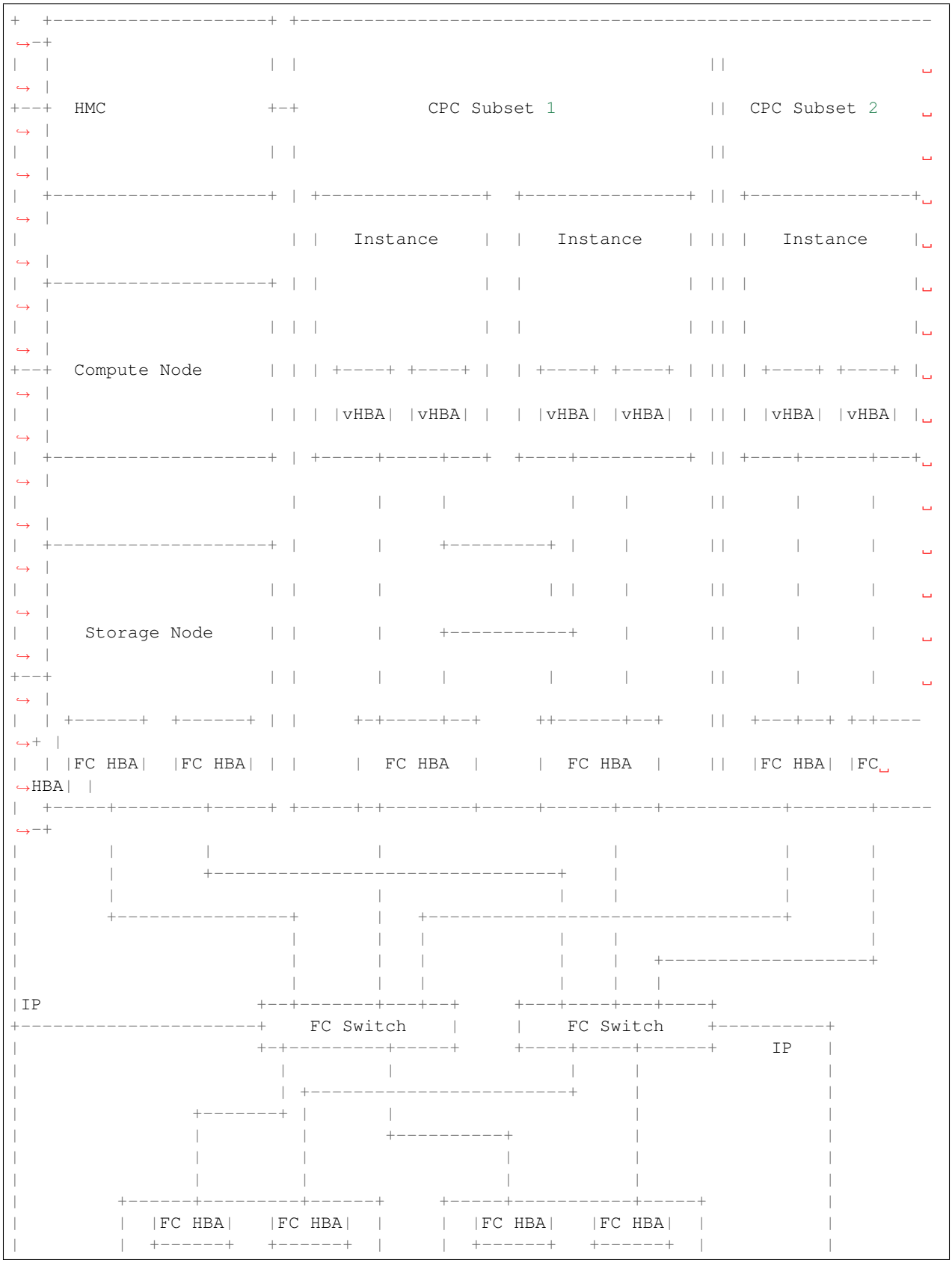
A CPC needs to be configured into CPC subsets. Each CPC subset defines the amount of processors, memory and FCP adapter ports for OpenStack to manage. For more details on CPC subsets see the chapter *Topology*.

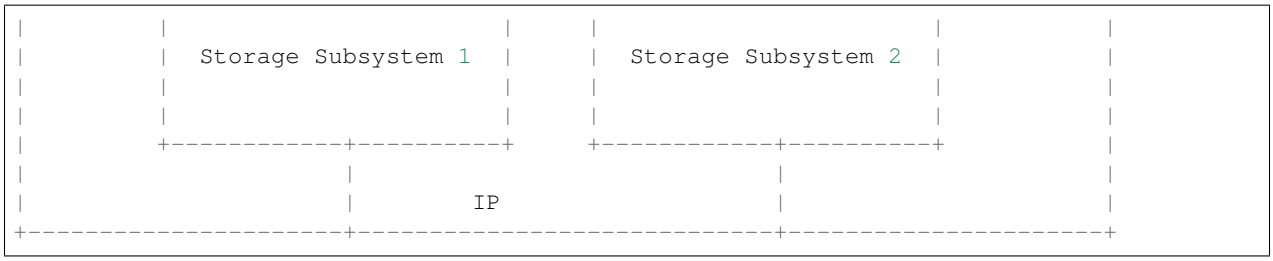
An instance requires access to a virtual Host Bus Adapter (vHBA) to be able to access a FCP volume. Each vHBA behaves like a physical HBA. It has a World Wide Port Number (WWPN) assigned to it. OpenStack DPM will create a vHBA for each FCP adapter port configured to each instance. It's important that all configured FCP adapter ports provide physical connectivity to each target storage subsystem.

The *storage node* requires connectivity to FCP adapter ports on the target storage subsystems to allow Cinder to deploy images. In addition, it requires IP connectivity to the storage subsystem to trigger the creation of a volume and to do the LUN (logical unit number) masking. An IP connection to the FCP switches is required to configure FCP zoning.

The *compute node* itself does not require a FCP or IP connection to the storage subsystem or the FCP switches at all.

The *controller node* does not require FCP connectivity at all.





Note: The *controller node* is not shown on this picture. It just requires IP connectivity to the *storage node* as well as to the *compute node*. FCP connectivity is not required for the *controller node*.

- FCP Cinder drivers take care of LUN masking and zoning configuration.
- The nova-dpm driver creates a vHBA for each adapter configured in nova.conf. For more details see the chapter *Configuration*.
- Instances do not support multipathing between the adapter and the switch during boot.

  Even though multiple FCP adapters are configured to the partition, only a single adapter is chosen for boot. After the operating system is up and running, the second vHBA can be enabled inband and used for multipathing.

1.4.4 References

[1] https://docs.openstack.org/admin-guide/common/get-started-storage-concepts.html

CHAPTER 2

Using the driver

2.1 Getting Started with OpenStack for DPM

This section gets you started with OpenStack for DPM.

It assumes

- an existing OpenStack controller, in which the OpenStack components specific to DPM will be integrated
- A system where the new compute node for DPM can be installed.

  This can either be a separate system but could also be the controller node itself (single node setup). Fore more information on topology see the doc *Topology*.

2.1.1 Compute

The following actions are done for the compute node.

1. Install the following packages

- Nova
- nova-dpm (see *Installation*)

1. Configure the nova-compute service for DPM

   Make sure you can answer the following questions:

   - On which CPC should the OpenStack instances run?
   - Which HMC manages that CPC?

   The following questions help to define the scope of the CPC subset. For more details on CPC subsets see the chapter *Topology*.

   - Which name should be used to identify this subset of the CPC? By default the hostname of the compute node is taken.

- How many partitions of this CPC should OpenStack be able to consume?
- What is the maximum number of shared IFL processors that OpenStack can assign to a single instance?

Note: OpenStack does *not* use those processors exclusively. These processors are still shared with non-OpenStack partitions.

- What is the maximum amount of memory that can be consumed by all OpenStack instances on this CEC?

Note: This memory is *not* being reserved upfront. It's only consumed when an instance is using it. It can happen, that none-OpenStack partitions consume that much memory that the defined maximum cannot be exploited by OpenStack. Therefore it's important to consider this number in the CECs memory pre-planing.

- Which storage adapter ports should be used for attaching FCP storage?

With this information you should be able to create the corresponding configuration file with the help of the *Configuration* documentation.

2. Start the nova-compute service

```
# /usr/bin/python /usr/local/bin/nova-compute --config-file /etc/nova/
↪nova.conf
```

Note: The file *nova.conf* must contain all the DPM specific configuration options.

---

3. Verify if the service is running

```
# openstack compute service list
+----+------------------+----------+----------+---------+-------+---------
↪-------------------+
| ID | Binary           | Host     | Zone     | Status  | State | Updated␣
↪At                  |
+----+------------------+----------+----------+---------+-------+---------
↪-------------------+
|  4 | nova-conductor   | control1 | internal | enabled | up    | 2017-02-
↪22T10:31:08.000000 |
|  5 | nova-scheduler   | control1 | internal | enabled | up    | 2017-02-
↪22T10:31:01.000000 |
|  6 | nova-consoleauth | control1 | internal | enabled | up    | 2017-02-
↪22T10:31:08.000000 |
|  7 | nova-compute     | subset_1 | nova     | enabled | up    | 2017-02-
↪22T10:30:59.000000 |
+----+------------------+----------+----------+---------+-------+---------
↪-------------------+
```

-> Amongst others a nova-compute service for the chosen subset host name (subset_1) should show up.

```
# openstack hypervisor list
+----+---------------------+-----------------+-----------------+-------+
| ID | Hypervisor Hostname | Hypervisor Type | Host IP         | State |
```

```
+----+--------------------+-----------------+-----------------+-------+
|  1 | subset_1           | PRSM            | xxx.xxx.xxx.xxx | up    |
+----+--------------------+-----------------+-----------------+-------+
```

-> A hypervisor of type PRSM using the same hostname like above should show up.

2.1.2 Storage

There is no storage related service required on the compute node. The compute node does not require a FCP storage attachment at all.

On the storage node the cinder volume service must be configured for FCP usage.

For more details see *Storage Support*.

2.1.3 Networking

1. Install the following Python packages on the compute node for DPM and the existing controller node

- Neutron
- networking-dpm (see‘documentation <http://networking-dpm.readthedocs.io/en/latest/installation.html>‘_)

1. Configure the Neutron DPM mechanism driver on the existing controller node

   The DPM mechanism driver must be configured to be used by the Neutron server's ML2 plug-in. Other drivers required by the network node might be configured in parallel. For more details see the configuration documentation. After the configuration change the Neutron server must be restarted to apply the changes.

2. Configure the Neutron DPM agent on the compute node for DPM

   Make sure you can answer the following questions:

   - Which network adapter ports should be used for instances created by OpenStack?
     - The list of supported network adapters can be found here.
   - How many logical networks are required? A dedicated adapter port is required for each physical (and therefore for each logical) network (with flat networking, the mapping between a physical and a logical network is 1:1).

   With this information and the help of the configuration documentation you should be able to create the Neutron DPM agent configuration file.

3. Start the Neutron DPM agent on the compute node for DPM

```
/usr/bin/python /usr/local/bin/neutron-dpm-agent --config-file /etc/
↪neutron/plugins/ml2/neutron_dpm_agent.conf
```

Note: The file *neutron_dpm_agent.conf* must contain all the DPM specific configuration options. In addition it must specify the CPCSubset that it belongs to in the *host* variable of the *DEFAULT* section.

4. Verify if the agent is running

```
# openstack network agent list
+--------------------------------------+--------------------+----------+--
↪-----------------+-------+-------+---------------------------+
| ID                                   | Agent Type         | Host     |␣
↪Availability Zone | Alive | State | Binary                    |
+--------------------------------------+--------------------+----------+--
↪-----------------+-------+-------+---------------------------+
| 0d9ec043-9dcf-478c-a4df-56c93e516ca8 | DPM agent          | subset_1 |␣
↪None              | True  | UP    | neutron-dpm-agent         |
| 42264083-e90d-4e7e-9b4f-0675e282d1ef | Metadata agent     | control1 |␣
↪None              | True  | UP    | neutron-metadata-agent    |
| 6d2dbc59-db7b-4f34-9c5f-8fe9935ad824 | Open vSwitch agent | control1 |␣
↪None              | True  | UP    | neutron-openvswitch-agent |
| af25dea7-1895-4b81-b087-8e30101d2475 | DHCP agent         | control1 |␣
↪nova              | True  | UP    | neutron-dhcp-agent        |
+--------------------------------------+--------------------+----------+--
↪-----------------+-------+-------+---------------------------+
```

-> Amongst others a neutron-dpm-agent for the chosen subset host name (subset_1) should be alive.

```
# openstack network agent show 0d9ec043-9dcf-478c-a4df-56c93e516ca8
+-------------------+-----------------------------------------------------
↪--------------------------------------------------------+
| Field             | Value                                               ␣
↪                                                       |
+-------------------+-----------------------------------------------------
↪--------------------------------------------------------+
| admin_state_up    | UP                                                  ␣
↪                                                       |
| agent_type        | DPM agent                                           ␣
↪                                                       |
| alive             | True                                                ␣
↪                                                       |
| availability_zone | None                                                ␣
↪                                                       |
| binary            | neutron-dpm-agent                                   ␣
↪                                                       |
| configuration     | {u'extensions': [], u'adapter_mappings': {u'provider
↪': [u'3ea09d2a-b18d-11e6-89a4-42f2e9ef1641']}, u'devices': 0} |
| created_at        | 2017-02-22 11:47:57                                 ␣
↪                                                       |
| description       | None                                                ␣
↪                                                       |
| host              | subset_1                                            ␣
↪                                                       |
| id                | 0d9ec043-9dcf-478c-a4df-56c93e516ca8                ␣
↪                                                       |
| last_heartbeat_at | 2017-02-22 12:12:57                                 ␣
↪                                                       |
| name              | None                                                ␣
↪                                                       |
| started_at        | 2017-02-22 11:47:57                                 ␣
↪                                                       |
| topic             | N/A                                                 ␣
↪                                                       |
+-------------------+-----------------------------------------------------
↪--------------------------------------------------------+
```

-> The configuration option should show an adapter mapping. It's not exactly the same mapping as it was provided in the agents configuration file. It's a translated mapping, where the physical network is mapped to a vswitch object-id.

2.1.4 Spawning an instance

1. Creating a initial network

   Assuming that the Neutron DPM agent configuration *physical_network_adapter_mappings* contains a physical network called *provider*.

```
# openstack network create --provider-physical-network provider --
↪provider-network-type flat provider
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2017-02-22T12:46:35Z                 |
| description               |                                      |
| dns_domain                | None                                 |
| id                        | 49887552-ea35-41ca-aba2-2df2bb59896d |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | None                                 |
| mtu                       | 1500                                 |
| name                      | test-net                             |
| port_security_enabled     | True                                 |
| project_id                | 561a226832eb4eabb50b05d21c46d9bb     |
| provider:network_type     | flat                                 |
| provider:physical_network | provider                             |
| provider:segmentation_id  | None                                 |
| qos_policy_id             | None                                 |
| revision_number           | 3                                    |
| router:external           | Internal                             |
| segments                  | None                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| updated_at                | 2017-02-22T12:46:35Z                 |
+---------------------------+--------------------------------------+
```

```
# openstack subnet create --dhcp --subnet-range 192.168.222.0/24 --
↪network provider provider_subnet
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| allocation_pools  | 192.168.222.2-192.168.222.254        |
| cidr              | 192.168.222.0/24                     |
| created_at        | 2017-02-22T12:47:09Z                 |
| description       |                                      |
| dns_nameservers   |                                      |
| enable_dhcp       | True                                 |
| gateway_ip        | 192.168.222.1                        |
| host_routes       |                                      |
| id                | d6e641a7-8c42-43a6-a3e1-193de297f494 |
```

```
| ip_version         | 4                                    |
| ipv6_address_mode  | None                                 |
| ipv6_ra_mode       | None                                 |
| name               | provider_subnet                      |
| network_id         | 49887552-ea35-41ca-aba2-2df2bb59896d |
| project_id         | 561a226832eb4eabb50b05d21c46d9bb     |
| revision_number    | 2                                    |
| segment_id         | None                                 |
| service_types      |                                      |
| subnetpool_id      | None                                 |
| updated_at         | 2017-02-22T12:47:09Z                 |
+--------------------+--------------------------------------+
```

2. Check the existing images:

```
# openstack image list
+--------------------------------------+--------------------------+--------+
| ID                                   | Name                     | Status |
+--------------------------------------+--------------------------+--------+
| a249ef36-74d1-48fb-8d65-c4d532fa68e6 | dpm_image                | active |
+--------------------------------------+--------------------------+--------+
```

3. Create a volume based on an image:

```
# openstack volume create  --image a249ef36-74d1-48fb-8d65-c4d532fa68e6  --size␣
↪15 dpm_volume1
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| attachments         | []                                   |
| availability_zone   | nova                                 |
| bootable            | true                                 |
| consistencygroup_id | None                                 |
| created_at          | 2017-02-22T14:42:27.013674           |
| description         | None                                 |
| encrypted           | False                                |
| id                  | 25307859-e227-4f2b-82f8-b3ff3d5caefd |
| migration_status    | None                                 |
| multiattach         | False                                |
| name                | vol_andreas                          |
| properties          |                                      |
| replication_status  | None                                 |
| size                | 15                                   |
| snapshot_id         | None                                 |
| source_volid        | 3d5f72ec-9f1d-41fe-8bac-77bc0dc1e930 |
| status              | creating                             |
| type                | v7kuni                               |
| updated_at          | None                                 |
| user_id             | 0a6eceb0f73f4f37a0fce8936a1023c4     |
+---------------------+--------------------------------------+
```

4. Wait until the volume status changed to “available”:

```
# openstack volume list
+--------------------------------------+--------------+-----------+------+--------
↪-----+
| ID                                   | Display Name | Status    | Size |␣
↪Attached to |
```

```
+--------------------------------------+--------------+----------+------+--------
↪-----+
| 25307859-e227-4f2b-82f8-b3ff3d5caefd | dpm_volume1  | available |   15 |        ␣
↪      |
+--------------------------------------+--------------+----------+------+--------
↪-----+
```

5. Check the existing flavors:

```
# openstack flavor list
+-------+-----------+-------+------+-----------+-------+-----------+
| ID    | Name      |   RAM | Disk | Ephemeral | VCPUs | Is Public |
+-------+-----------+-------+------+-----------+-------+-----------+
| 1     | m1.tiny   |   512 |    1 |         0 |     1 | True      |
| 2     | m1.small  |  2048 |   20 |         0 |     1 | True      |
+-------+-----------+-------+------+-----------+-------+-----------+
```

6. Boot the instance:

```
# openstack server create --flavor m1.small --volume dpm_volume1 --nic net-id=
↪$(openstack network list | awk '/test-net/ {print $2}') dpm_server1
+--------------------------------------+--------------------------------------+
| Field                                | Value                                |
+--------------------------------------+--------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                               |
| OS-EXT-AZ:availability_zone          |                                      |
| OS-EXT-SRV-ATTR:host                 | None                                 |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | None                                 |
| OS-EXT-SRV-ATTR:instance_name        |                                      |
| OS-EXT-STS:power_state               | NOSTATE                              |
| OS-EXT-STS:task_state                | scheduling                           |
| OS-EXT-STS:vm_state                  | building                             |
| OS-SRV-USG:launched_at               | None                                 |
| OS-SRV-USG:terminated_at             | None                                 |
| accessIPv4                           |                                      |
| accessIPv6                           |                                      |
| addresses                            |                                      |
| adminPass                            | TbLsiNT8rN3n                         |
| config_drive                         |                                      |
| created                              | 2017-02-22T14:46:24Z                 |
| flavor                               | m1.small (2)                         |
| hostId                               |                                      |
| id                                   | 9b44589c-cd91-4b67-9a9f-2ec88ad1c27d |
| image                                |                                      |
| key_name                             | None                                 |
| name                                 | dpm_server1                          |
| progress                             | 0                                    |
| project_id                           | e2e0784ca1b64d6cae07d3c6e8d4bcff     |
| properties                           |                                      |
| security_groups                      | name='default'                       |
| status                               | BUILD                                |
| updated                              | 2017-02-22T14:46:24Z                 |
| user_id                              | 0a6eceb0f73f4f37a0fce8936a1023c4     |
| volumes_attached                     |                                      |
+--------------------------------------+--------------------------------------+
```

2.2 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?h=stable/ocata
$ 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?h=stable/ocata
$ 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?h=master
$ 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/.

2.3 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)
# Minimum value: 512
#max_memory = <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 (DPM Object-Id value)
#cpc_object_id = <None>
```

2.4 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.

2.4.1 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

2.4.2 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

CHAPTER 3

Creating DPM Images

3.1 Creating a qcow2 image for RHEL

This section explains the qcow2 image creation for RHEL.

3.1.1 Precondition

Partition with RHEL-7.3 installed and root user access

3.1.2 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\.[0-9a-fA-F]\.[0-9a-fA-F]\{4\},0x[0-9a-fA-F]\
↪{16\},0x[0-9a-fA-F]\{16\}' >/dev/null
        < 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/firmware/ipl/lun)"
        > 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 zfcp_arg in $(getargs rd.zfcp -d 'rd_ZFCP='); do
    if [ "$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/firmware/ipl/lun)"
    fi
    (
       IFS=","
        set $zfcp_arg
        echo "$@" >> /etc/zfcp.conf
    )
done

zfcp_cio_free
```

6. **Rebuild the ramdisk** *dracut -f*
7. **To apply the above changes to the contents of boot loader script** *zipl -V*

3.1.3 Installation of Cloud-init

Install cloud-utils-growpart

This package is required for growing the filesystem.

- Download a version of cloud-utils-growpart < 0.27.13

> **Note:** Greater version can not deal with rhel7s old 'util-linux' package version

- install it:

```
rpm -i cloud-utils-growpart-0.27-9.fc20.noarch.rpm
```

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/system/cloud-init.service

sed -i '/^\[Unit\]$/,/^\[/ s/^DefaultDependencies=no/#DefaultDependencies=no/' /
↪usr/lib/systemd/system/cloud-init-local.service
```

- Remove ordering for sysinit.target

```
sed -i '/^\[Unit\]$/,/^\[/ s/^Before=sysinit.target/#Before=sysinit.target/' /usr/
↪lib/systemd/system/cloud-init.service

sed -i '/^\[Unit\]$/,/^\[/ s/^Before=sysinit.target/#Before=sysinit.target/' /usr/
↪lib/systemd/system/cloud-init-local.service
```

- order with systemd-hostnamed.service

```
sed -i '/^\[Unit\]$/,/^\[/ s/^After=networking.service/After=networking.
↪service\nAfter=systemd-hostnamed.service/' /usr/lib/systemd/system/cloud-init.
↪service
```

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.

3.1.4 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/guest_image_tools/usr/bin/autoconfigure_networking.sh  /usr/bin/
↪autoconfigure_networking.sh

cp nova-dpm/guest_image_tools/usr/lib/systemd/system/autoconfigure_networking.
↪service  /usr/lib/systemd/system/autoconfigure_networking.service

cp nova-dpm/guest_image_tools/usr/bin/setmac.sh  /usr/bin/setmac.sh

cp nova-dpm/guest_image_tools/usr/bin/dpm_guest_image_tools_common  /usr/bin/

cp nova-dpm/guest_image_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

3.1.5 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 ~/**

3.1.6 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

3.1.7 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

CHAPTER 4

Contributing to the project

4.1 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

4.2 Developer Guide

4.2.1 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: https://docs.openstack.org/reno/latest/index.html

4.2.2 Release Management

Note: Be aware that the description below is only a simplified summary of the official release management documentation which can be found at https://docs.openstack.org/project-team-guide/release-management.html Reading the section below doesn't replace reading the official docs.

Getting Started

Milestone release

- Checkout the master branch
- Create a tag `<version>.0b<milestone>`, e.g. `1.0.0.0b1` for the first milestone of the Ocata release. For more details, see *Tag releases*.
- Push the tag along *Tag releases*.
- Update the launchpad project
  - Release the corresponding milestone along *Release a Milestone*.

First release candidate

- Checkout the master branch
- Create a tag `<version>.0rc1`, e.g. `1.0.0.0rc1`. For more details, see *Tag releases*.
- Push the tag along *Tag releases*.
- Create a stable branch `stable/<release>`, e.g. `stable/ocata`. For more details, see *New stable branch*.
- Update the launchpad project
  - Create a Milestone for the release candidate along *Create a Milestone for a Series*.
  - Release the milestone along *Release a Milestone*.
  - Change the Focus of development along *Change focus of development*.
- Make the documentation for the new stable branch available on RTD along *Activate/deactivate docs for a branch or tag*.

Follow up release candidates

- Checkout the `stable/<release>` branch
- Create a tag `<version>.0rc2`, e.g. `1.0.0.0rc2`. For more details, see *Tag releases*.
- Push the tag along *Tag releases*.
- Update the launchpad project
  - Create a Milestone for the release candidate along *Create a Milestone for a Series*.
  - Release the milestone along *Release a Milestone*.

Release

- Checkout the `stable/<release>` branch:

```
git fetch
git checkout -t stable/<release>
```

- Create a tag `<version>` e.g. `1.0.0` with the description *<release> release*. For more details, see *Tag releases*.
- Push the tag along *Tag releases*.

Model

We follow the *Common cycle with development milestones* like Nova does. In short, this mean we will produce:

- one *full release* at the end of each development cycle
- AND three *milestone releases* during each development cycle.

The versioning of those releases will also follow the rules Nova uses. In short, this means we will have releases which looks like this:

- The first full release based on *Ocata* has version `1.0.0`.
- A (possible) 2nd full release based on *Ocata* has version `1.0.1`

- The first milestone release in *Pike* has version `2.0.0.0b1`
- The second milestone release in *Pike* has version `2.0.0.0b2`
- The third milestone release in *Pike* has version `2.0.0.0b3`
- The first release candidate for *Pike* has version `2.0.0.0rc1`
- The second full release based on *Pike* has version `2.0.0`

The versioning happens with *git* tags on specific commits which we will define during the (full/milestone) release process.

Process

When creating a new full release, the usual order of action is:

- start during the RC phase (usually ~3 weeks before the release)
- merge the open changes which need to make the release into master
- tag the last commit in `master` with the *release candidate* tag
- create a `stable/<release>` branch from that tag (master is now open for changes for the next release)
- double-check if that release candidate needs fixes
- tag the final release candidate 1 week before the actual release
- tag the final full release

Note: As a project which is not under the Openstack governance, we don't use the `openstack/releases` repository to create releases and stable branches. See *New stable branch* for the HOW-TO.

Tag releases

Releases are done via *Git tags*. The list of releases can be found at https://github.com/openstack/nova-dpm/releases . To tag the first release candidate (RC) for the next release, follow the steps below. We use the *Ocata* release as an example:

1. You need a key to sign the tag:

```
$ gpg --list-keys
```

2. If this is not yet done, create one:

```
$ gpg --gen-key
```

3. Go to the commit you want to tag (usually the latest one in `master`):

```
$ git checkout master
$ git pull
```

4. (Optional) Double-check the list of current tags:

```
$ git tag -l
```

5. Create a signed tag:

```
$ git tag -s 1.0.0.0rc1 -m "RC1 for the Ocata release"
```

6. Push that tag via the *gerrit* remote (no Gerrit change will be created):

```
$ git push gerrit 1.0.0.0rc1
```

7. (Optional) Wait for ~5m, then you can check if the automatic release process was executed:

```
$ git os-job 1.0.0.0rc1
```

At this point we are done with the release of a version. You might want to check if the artifacts show the new version number:

- The read-only github repo: https://github.com/openstack/nova-dpm/releases
- The package on PyPi: https://pypi.python.org/pypi/nova-dpm
- The docs on RTD: http://nova-dpm.readthedocs.io/en/latest/

Note: RTD uses `pbr` to determine the version number and shows a version number higher than that you pushed before, that's fine and nothing to worry about.

Warning: Further release candidates and the final release must be tagged in the `stable/<release>` branch and **not** in the `master` branch.

4.2.3 Stable Branches

Note: Be aware that the description below is only a simplified summary of the official stable branch documentation which can be found at https://docs.openstack.org/project-team-guide/stable-branches.html Reading the section below doesn't replace reading the official docs.

Supported releases

We will have 3 simultaneously maintained branches as a maximum. These are:

- master (`N`)
- the latest stable release (`N-1`)
- the older stable release (`N-2`)

Branches older than these will be deleted after a `<release-eol>` tag was applied to the last commit of that branch.

Backports

Again, we follow the same rules Nova does. In short, this means:

- for the latest stable branch (`N-1`)
  - No backports of features are allowed
  - All kinds of bugfixes are allowed

- for the older stable branch (N-2)
  - Only critical bugfixes and security patches

Fixes need to be first done in the master branch (N) and then cherry-picked into the stable branches (first N-1 and after that, if necessary, N-2).

The original Change-Id needs to be kept intact when a backport is proposed for review.

The short version of the technical side of creating a backport:

```
$ git checkout -t origin/stable/ocata
$ git cherry-pick -x $master_commit_id
$ git review stable/ocata
```

New stable branch

After the first release candidate is tagged in master, you should create the stable branch in *Gerrit* based on that:

1. Check if you are a member of the Gerrit group nova-dpm-release: https://review.openstack.org/#/admin/groups/1633,members
2. This release group is allowed to create references and tags: https://review.openstack.org/#/admin/projects/openstack/nova-dpm,access
3. Go to https://review.openstack.org/#/admin/projects/openstack/nova-dpm,branches and enter the branch name stable/<release> and the initial revision it is based on (the release candidate tag).

   (a) Example for Ocata:

   ```
   Branch Name: stable/ocata
   Initial Revision: 1.0.0.0rc1
   ```

   (b) Example for Pike:

   ```
   Branch Name: stable/pike
   Initial Revision: 2.0.0.0rc1
   ```

After this is done, every open change in Gerrit which uses master as target branch will be (if it will merge) part of the next release.

4.2.4 Launchpad

Create a new Series with milestones

1. Go to https://launchpad.net/nova-dpm/+addseries to register a new release series using
   - name: <release>, e.g. pike
   - description: Development series for the Pike release <version>., e.g. Development series for the Pike release 2.0.0.
2. Create the milestones for the new release along *Create a Milestone for a Series*. Information about the milestones can be found at https://releases.openstack.org/<release>/schedule.html . E.g. https://releases.openstack.org/pike/schedule.html for the 'Pike' release.

   Do this for all 3 milestones.

Create a Milestone for a Series

Go to https://launchpad.net/nova-dpm/<release> and click on "Create milestone". Provide the following information

- name
  - Milestone: `<release>-<milestone>`, e.g. `pike-1`
  - Release candidate: `<release>-rc<candidate>`, e.g. `pike-rc1`
- code name
  - Milestone: `<short-release><milestone>`, e.g. `p1`
  - Release candidate: `RC<candidate>`, e.g. `RC1`
- date targeted

Release a Milestone

1. Open the Milestone using https://launchpad.net/nova-dpm/+milestone/ocata-rc1/+addrelease.
2. Specify the release date

Change focus of development

Go to the projects edit page https://launchpad.net/nova-dpm/+edit. Set 'Development focus' to the upcoming release series.

4.2.5 Read The Docs (RTD)

Activate/deactivate docs for a branch or tag

To create documentation for the stable stable branch, go to https://readthedocs.org/projects/nova-dpm/versions/. Edit the version you want to change and tick or untick "Active". Exit with "Save".

Note: The strategy is to provide documentation for stable branches only (instead of release tags). Doing so, the backported documentation is available without having a new release required.

4.2.6 Requirements

This chapter describes how requirements are handled. The most important requirements are the library `os-dpm` and the `zhmcclient`.

Each project specifies its requirements using the `requirements.txt` and `test-requirements.txt` files.

In addition to that, requirements are also managed OpenStack wide in the requirements repository https://github.com/openstack/requirements. The following files are of importance

- `global-requirements.txt`

  Specifies a requirement and its minimum version. All requirements that are listed in a projects `requirements.txt` file must be listed in this file as well. There's a Jenkins job ensuring that the version in the projects `requirements.txt` always matches the exact version listed in this file.

Note: Exact really means exact, including white spaces and so on!

This file has to be updated manually.

- upper-constraints.txt

  This file specifies the upper version limit for a package. For each requirement listed in `global-requirements.txt` a corresponding entry must exist in this file. In addition an upper constraint for all indirect requirements must be specified in this file as well (e.g. zhmccclient uses `click-spinner`. An upper constraint must be specified for `click-spinner` as well, although no entry in `global-requirements.txt` exists).

  This file is being updated by the OpenStack Proposal Bot.

  - OpenStack libraries: The release job will trigger the Bot directly
  - External libraries: Bot is triggered on a daily bases (except if the branch is frozen due to a pending release)

  Also manual updates can be proposed.

- projects.txt

  The OpenStack Proposal Bot proposes changes made to *global-requirements* to the listed projects `requirements.txt` and `test-requirements.txt` file.

How to use a new version of a package?

The new version must be specified in `upper-constraints.txt` of the requirements repository. Usually the OpenStack Proposal Bot takes care about that. Alternatively a patch can be submitted manually.

TBD: When is the OpenStack Proposal Bot being triggered for OpenStack libraries vs. external libraries.

How to increase the minimum version for a package?

Propose a patch to the `global-requirements.txt` file of the requirements repository. The OpenStack Proposal Bot will propose a change to your project once that patch is merged.

If also the version in `upper-constraints.txt` should be bumped, do both with the same commit.

Note: The OpenStack Proposal Bot proposes changes made to *global-requirements* only to projects listed in `projects.txt` of the requirements repo.

How to avoid that a new version of a package gets applied to a project?

The upper constraint cannot be controlled on a project basis.

The only way to mark a invalid version is to propose a change to the `global-requirements.txt` file of the requirements repository to exclude the invalid version.

Note: If you plan to use that version in the future do not propose an update to `global-requirements.txt`. Rather focus on fixing the issue with the new version in your project right now!

Note: On a version bump, the unittests of the main projects are run to ensure those are not breaking. But this is only for the major projects.

CHAPTER 5

Specs

5.1 Template

5.1.1 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 https://docs.openstack.org/nova/latest/blueprints.html
- 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/compute/schemas

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 additionaProperties 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.

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 5.1: Revisions

Release Name	Description
Ocata	Introduced

5.2 Ocata

5.2.1 Base support Nova Neutron Communication

https://blueprints.launchpad.net/nova/+spec/neutron-integration

The nova-dpm virt driver needs integration into Neutron to allow instance networking using OSA and hipersockets. RoCE attachments are not supported.

Problem Description

The nova-dpm virt driver needs integration into Neutron to allow instance networking. The initial integration comes with the following restrictions:

- There's no way to specify the MAC address for a DPM NIC. Therefore the MAC address of Neutron port will not be equal the actual used MAC address of the NIC. This has the following known impact
  - Addresses from Neutron DHCP server do not get assigned
- Only flat networking supported
- No live attach of an interface (attaching only works on boot)

Use Cases

- Starting a DPM instance with OSA and hipersockets NICs attached. RoCE attachments are not supported.

Proposed Change

Worksplit Nova Neutron

- Nova is responsible for the partition object and all its sub resources. This includes the NIC object.
- Neutron is responsible for the vSwitch and the (network) adapter objects and all its sub resources.
- In addition to that, Neutron should manage everything networking related that can be changed while a port is attached to an instance. Therefore some overlap in the NIC object might exist. Nova will create it, but Neutron might read it or update it after creation.

Background:

```
* Attaching the instance to the network is job of Nova, also in other virt
  drivers like for libvirt.

* In fully virtualized systems there's is a differentiation between the
  actual NIC (managed by Nova) and the virtual switch port (managed by
  Neutron). DPM does not offer this differentiation.
```

DPM Objects manged by Nova

- Partition
  - Create NIC: `POST /api/partitions/{partition-id}/nics`
  - Delete NIC: `DELETE /api/partitions/{partition-id}/nics/{nic-id}`
  - Get NIC Properties: `GET /api/partitions/{partition-id}/nics/{nic-id}`

```
"class":"nic",
"description":"",
"device-number":"000A",
"element-id":"97de6092-c049-11e5-8f1f-020000000108",
"element-uri":"/api/partitions/67782eb8-c041-11e5-92b8-020000000108/nics/
97de6092-c049-11e5-8f1f-020000000108",
"name":"jn",
"parent":"/api/partitions/67782eb8-c041-11e5-92b8-020000000108",
"type":"osd",
"virtual-switch-uri":"/api/virtual-switches/6a428720-b9d3-11e5-9e34-
↪020000000108"
```

  - Update NIC Properties: `POST /api/partitions/{partition-id}/nics/{nic-id}`

DPM Objects managed by Neutron

- Adapter
  - Get Adapter Properties: `GET /api/adapters/{adapter-id}`


```
"adapter-family":"ficon",
"adapter-id":"141",
"allowed-capacity":32,
"card-location":"Z22B-D207-J.01",
"channel-path-id":"01",
"class":"adapter",
"configured-capacity":1,
"description":"",
"detected-card-type":"ficon-express-8",
"maximum-total-capacity":255,
"name":"FCP 0141 Z22B-07",
"object-id":"d71902a4-c930-11e5-a978-020000000338",
"object-uri":"/api/adapters/d71902a4-c930-11e5-a978-020000000338",
"parent":"/api/cpcs/87dbe268-0b43-362f-9f80-c79923cc4a29",
"physical-channel-status":"operating",
"port-count":1,
"state":"online",
"status":"active",
"storage-port-uris":[
"/api/adapters/d71902a4-c930-11e5-a978-020000000338/storage-ports/0"
],
"type":"fcp",
"used-capacity":1
```


  - Update Adapter Properties: `POST /api/adapters/{adapter-id}`
  - Create Hipersockets: `POST /api/cpcs/{cpc-id}/adapters`
  - Delete Hipersockets: `DELETE /api/adapters/{adapter-id}`
  - Get Network Port Properties: `GET /api/adapters/{adapter-id}/network-ports/{network-port-id}`


```
"class":"network-port",
"description":"",
"element-id":"0",
"element-uri":"/api/adapters/e77d39f8-c930-11e5-a978-020000000338/network-
↪ports/0",
"index":0,
"name":"Port 0",
"parent":"/api/adapters/e77d39f8-c930-11e5-a978-020000000338"
```


  - Update Network Port Properties: `POST /api/adapters/{adapter-id}/network-ports/{network-port-id}`
- Virtual Switch
  - Get Virtual Switch Properties: `GET /api/virtual-switches/{vswitch-id}`


```
"backing-adapter-uri":"/api/adapters/f718c7a0-d490-11e4-a555-020000003058",
↪"class":"virtual-switch",
"description":"",
"name":"PrimeIQDVSwitch1",
```

```
"object-id":"f6b4c70e-d491-11e4-a555-020000003058",
"object-uri":"/api/virtual-switches/f6b4c70e-d491-11e4-a555-020000003058",
"parent":"/api/cpcs/8e543aa6-1c26-3544-8197-4400110ef5ef",
"port":0,
"type":"hipersockets"
```

  - Update Virtual Switch Properties: `POST /api/virtual-switches/{vswitch-id}`

Potential overlap between Nova and Neutron

There's no doubt about that Nova should create the NIC object. However some attributes of the NIC object might need to be managed by Neutron.

The questionable attribute would be

- device-number

  The device number auto assignment of DPM will be used. Due to that Neutron is not aware of the device numbers at all. Only Nova needs to know about device numbers for passing this information into the partitions operating system.

Note: In future dpm releases there might additional questionable attributes like the anti spoofing feature or setting the link up/down.

Mapping OpenStack API - DPM API

This is a mapping of OpenStack API calls and resulting DPM API calls.

Table 5.2: OpenStack API - DPM API Mapping

OpenStack API	DPM API
Nova: Create instance on network	Create NIC
Nova: Delete instance with attached	Delete NIC
Nova: Attach interface	Create NIC
Nova: Detach interface	Delete NIC
Neutron: Create Port	n/a (4)
Neutron: Delete Port	n/a (4)
Neutron: Update Port - change MAC	n/a (1)

Out of scope are

- Quality of service (2)
- Security Groups (2)
- Setting MTU (3)

(1) If a port is bound to a certain host (the corresponding DPM NIC object exists), changing the MAC is denied by Neutron. If the port is unbound, updating the MAC is allowed by Neutron.

(2) Not available in DPM rel. 1

(3) Not required. Automatically done by Operating System. MTU is part of DHCP offers or cloud-init configuration.

(4) Creating does not result in a NIC creation. Create port only creates the ports DB entry. The corresponding NIC gets created once the partition gets started. Same applies for delete port.

The 'host' identifier

The 'host' attribute is an unique identifier for a hypervisor. It is used by Nova and by Neutron. During spawn call, Nova requests a Neutron port to be created for a certain hypervisor. The hypervisor is identified by this host identifier. It's part of the create port call. For more details, see the flow diagram further below.

Nova Spawn Instance

The Nova driver call "spawn instance" attaches the partition to the networks. The following steps are required:

- Retrieve the relevant information from Neutron
- Create the NIC

Retrieving the relevant information from Neutron

The nova compute manager already does this. Then it calls the virt drivers "spawn" method passing a list of VIF (Virtual Interface) dicts. This list is named *network_info*. A VIF dict represents a Neutron port (1:1 mapping) and contains all relevant information that Nova needs. A VIF dict (or Neutron port) is represented by 0-1 DPM NICs (Neutron port can exist without a corresponding DPM NIC object).

Note: There is currently a transition going on, to transform all VIF dicts into an os-vif object [6]. Nova already started that transition in the VIFDriver (see below). The final goal is to use this object for Neutron as well. But Neutron did not yet adopt to it and only a few Nova vif_types already switched to the new object.

Generation of the *network_info* list and its VIF dicts happens in *neutronv2/api.py* method *_build_network_info_model* [7]. a The VIF dict is defined in *network/model.py* [5].

Create the NIC

Nova needs to create the NIC on the partition object.

First Nova needs to check the *vif_type* to assess if it can support such a network attachment. At the beginning, the nova dpm driver will only support the type "dpm_vswitch". If a port has another *vif_type*, processing should fail. The *vif_types* are defined by Neutron, in this particular case by the networking-dpm project [2].

Note: The Nova libvirt driver implements a VIFDriver framework, to support different *vif_type* attachments [3]. A vif driver does 3 things: Define the configuration of the NIC, do some plumbing that is required to do the NIC creation (plug) and do some cleanup after a NIC got deleted (unplug). As we do not need any plumbing for dpm done, the plan is to not implement such a framework in the initial release. This will also speeds up development.

Support for the VIFDriver framework and os-vif will be introduced in a later release.

This is how the main path of the code could look like:

```
host = cfg.CONF.host
# Waiting for vif-plugged event can be skipped in the first prototypes
with wait_for_vif_plugged_event:
    for vif in network_info:
       # do something with vif
      port_id = vif['id']
      vif_type = vif['type']
      mac = vif['address']
      vif_details = vif['details']
      dpm_object_id = vif_details['object_id']

      # Only dpm_vswitch attachments are supported for now
      if vif_type != "dpm_vswitch":
         raise Exception()
```

```
dpm_nic_dict = {
  "name": "OpenStack Neutron Port" + port_id,
  "description": "OpenStack mac=" + mac + ", CPCSubset=" + host,
  "virtual-switch-uri": "/api/virtual-switches/" + object_id
}
partition.nics.create(dpm_nic_dict)
```

Note: Having the NICs *name* starting with 'OpenStack' is not mandatory. It's just there to indicate an HMC user that this NIC is managed by OpenStack and he better not touches it.

Note: Having the uuid of the Neutron port in the DPM NICs name field is important to ensure uniqueness. For DPM the NIC name is a mandatory field that must be unique within the scope of a partition. Therefore the Neutron UUID comes into the play. The Neutron ports name must not be used, as it is an optional attribute.

Note: Having the NICs *description* starting with 'OpenStack' is important. The DPM Neutron Agent uses this to identify if a NIC is managed by OpenStack or not.

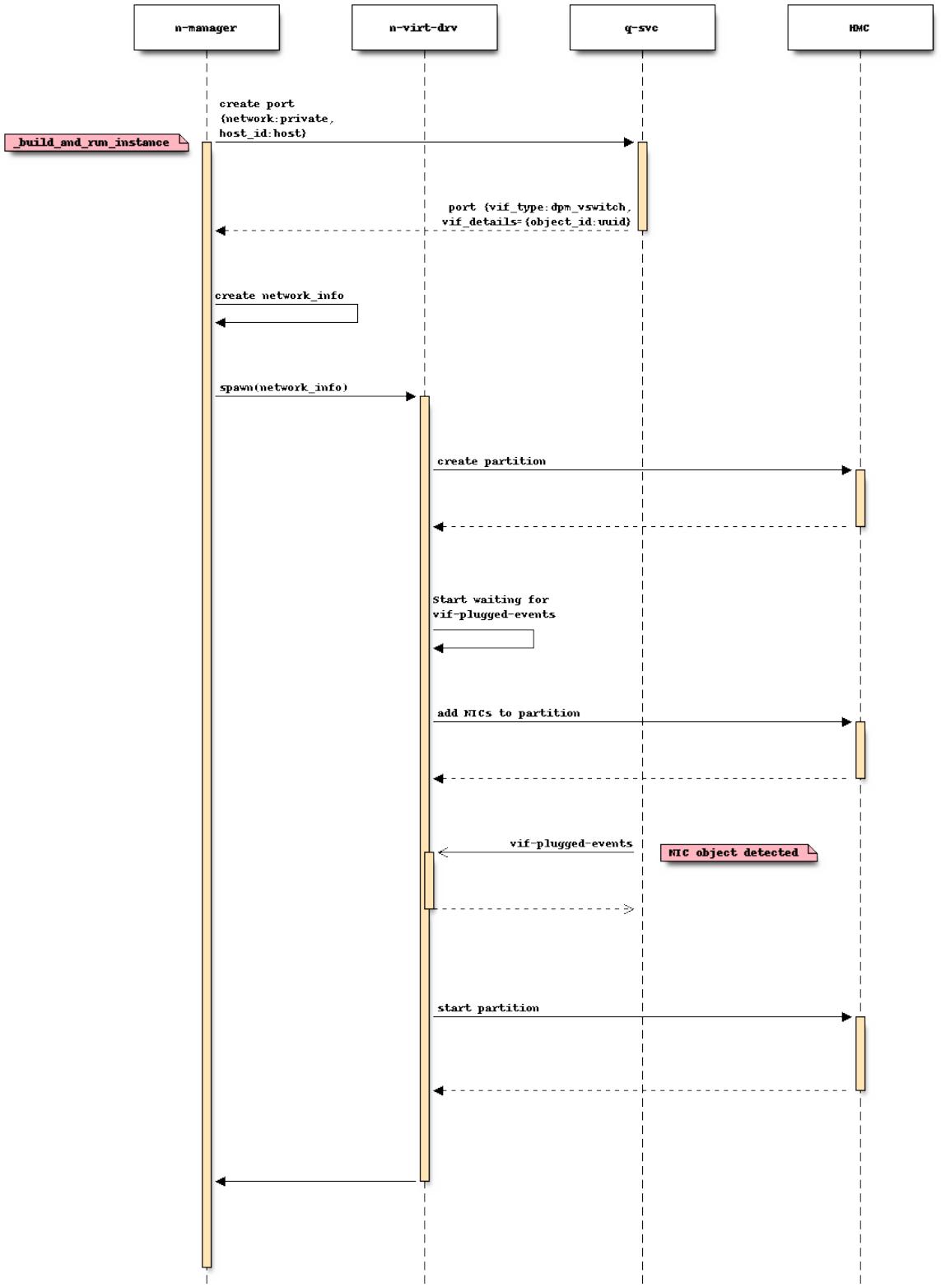
Note: Having the mac of the Neutron port in the description attribute is important. It will later on be used by the Neutron agent to map a NIC object to a Neutron port! Also Novas detach interface probably needs to identify the NIC along the ports MAC.

Note: Having the host-identifier at the NIC is also of importance. The same adapter might be used by different CPC-Subsets. Adding the host-identifier we can ensure, that only the neutron agent that is responsible for the CPCSubset handles those NICs. Otherwise those NICs would be reported on both agents a "up" which leads to confusion in the neutron-server. The proposal is to add the host-identifier somewhere in the description field. Neutron will check for this.

Note: There is no need for Nova to know if the vswitch object corresponds to an OSA adapter or an Hipersockets adapter. The DPM API for attaching those types is the same.

Note: The RoCE adapter is not supported at all. Once it becomes supported a new *vif_type* 'dpm_adapter' will be avaiable.

Spawn Instance Flow


n-manager
n-virt-drv
q-svc
HMC
create port
{network:private,
host_id:host}
_build_and_run_instance
port {vif_type:dpm_vswitch,
vif_details={object_id:uuid}
create network_info
spawn(network_info)
create partition
Start waiting for
vif-plugged-events
add NICs to partition
vif-plugged-events
NIC object detected
start partition

Note: There's an effort going on to move the Port creation from nova-manager to nova conductor [4].

- **On _build_and_run_instance, nova compute manager (n-manager) asks Neutron** to create a port with the following relevant details
  - host = the host identifier (hypervisor) on which the instance should be spawned
  - network = the network that the instance was launched on
- Nova manager creates the *network_info* list out of this information
- Nova manager calls the nova virt-driver (n-virt-drv) to spawn the instance passing in the *network_info* list
- Nova virt-driver creates the Partition (This can also done before the port details are requested).
- Nova start waiting for the vif-plugged events and then attaches the NICs to the partition
- The Neutron server sends the vif-plugged-events to Nova (after it detected the NIC).
- After all events have been received, Nova virt-driver starts the partition

Alternatives

None

Data model impact

None

REST API impact

None

Security impact

None

Notifications impact

None

Other end user impact

None

Performance Impact

None

Other deployer impact

None

Developer impact

None

Implementation

Assignee(s)

Primary assignee: <launchpad-id or None>

Other contributors: <launchpad-id or None>

Work Items

- All in one :)

Dependencies

Neutron DPM ML2 mechanism driver and agent https://bugs.launchpad.net/networking-dpm/+bug/1646095

Testing

- Unittest

Documentation Impact

TBD

References

[1] https://blueprints.launchpad.net/nova-dpm/+spec/neutron-integration-set-mac [2] https://github.com/openstack/networking-dpm [3] https://github.com/openstack/nova/blob/master/nova/virt/libvirt/vif.py [4] http://lists.openstack.org/pipermail/openstack-dev/2016-November/107476.html [5] https://github.com/openstack/nova/blob/15.0.0.0rc2/nova/network/model.py#L356 [6] https://github.com/openstack/os-vif [7] https://github.com/openstack/nova/blob/15.0.0.0rc2/nova/network/neutronv2/api.py#L2144

5.2.2 Neutron Integration - Setting the MAC address

https://blueprints.launchpad.net/nova-dpm/+spec/neutron-integration-set-mac

DPM does not support setting the MAC address for a NIC on its NIC creation. To workaround that, this blueprint proposes to set the MAC address from inside of the guest operating system.

Problem Description

DPM does not support setting the MAC address for a NIC on its NIC creation. It does also not offer a way to query the MAC address of an already created NIC. Therefore the MAC of the DPM NIC should be changed from within the operating system to ensure that the actual used MAC is equal the MAC of the Neutron port. That happens via a udev rule and a corresponding shell script.

Use Cases

- Attaching an instance to the network while using the correct MAC address
- Assignment of IP addresses from the OpenStack DHCP
- Using cloud-init to setup the networking with static ips (requires the MAC to be set correctly)

Proposed Change

Nova adds the relevant data to the 'boot-os-specific-parameters' property of an instance. This gets attached to the kernels cmdline during boot. The boot process itself ignores this data. Finally it's available in the guest operating system under */proc/cmdline*. If a new qeth interface gets defined in the guest, a udev rule triggers a script which is able to read the relevant information from the cmdline and set the MAC for that interface accordingly. This also works for interfaces that are brought up during boot.

Note: Updating the 'boot-os-specific-parameters' property on a running partition will not be reflected in /proc/cmdline. This requires the partition to stopped and started again (A inband reboot is not sufficient).

The format of the parameters is the following:

```
<devno>,<port-no>,<mac>;
```

Where

- <devno> is the first device number, of the qeth device added, e.g. '0001'
- <port-no> is the port number of the adapter, e.g. '0' or '1'
- <mac> is the MAC address to be set, without the ':' delimiter, e.g. 'aabbccddeeff'

The udev rule should call the script to configure the MAC for qeth devices only. As argument it passes in something that contains the interfaces device number.

Due to the limitations of the 'boot-os-specific-parameters' property the maximum amount of Ports (NICs) per instance (partiton) is limited.

- Max len of 'boot-os-specifc-parameters' = 256
- len required per NIC = 20 (0001,1,aabbccddeeff;)
- Max number of NICs = 256/20 = 12,8

-> A single partition can not have more than 12 NICs.

Note: This number goes with the assumption that the 'boot-os-specific-parameters' property is used by this feature exclusively.

Alternatives

Using the ConfigDrive to provide the relevant information. This would add certain complexity to the scripts in the operating system. Those scripts would need to find and mount that ConfigDrive. A bunch of code would be required for that. In addition the ConfigDrive is not available in Ocata.

Keeping the DPM chosen MAC and pushing this one back to Neutron. The problem with this approach is, that the Neutron port is created in the nova manager before any virt-driver gets invoked. Nova manager requests the port creation and binds it to the target host. Once the port is bound, changing the MAC is not possible anymore. That means, that the nova-dpm driver would need to unbind the port, change the MAC, bind it again.

In addition there's no way to get the DPM chosen MAC address, as an appropriate attribute on the DPM NIC object does not exist.

Data model impact

None

REST API impact

None

Security impact

None

Notifications impact

None

Other end user impact

The max number of Ports (NICs) per Instance (partition) is limited to 12.

Performance Impact

None

Other deployer impact

None

Developer impact

None

Implementation

Assignee(s)

Primary assignee: <andreas-scheuring>

Other contributors: None

Work Items

- All in one :)

Dependencies

- Initial Neutron integration: https://blueprints.launchpad.net/nova-dpm/+spec/neutron-integration

Testing

- Unittest

Documentation Impact

Document the limitation of maximum 12 Ports per Instance.

References

- Nova ConfigDrive support: https://blueprints.launchpad.net/nova-dpm/+spec/config-drive

CHAPTER 6

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