OpenStack:
Nova e Glance

Aula Teórica nº5

2020/2021

Compute Service - Nova

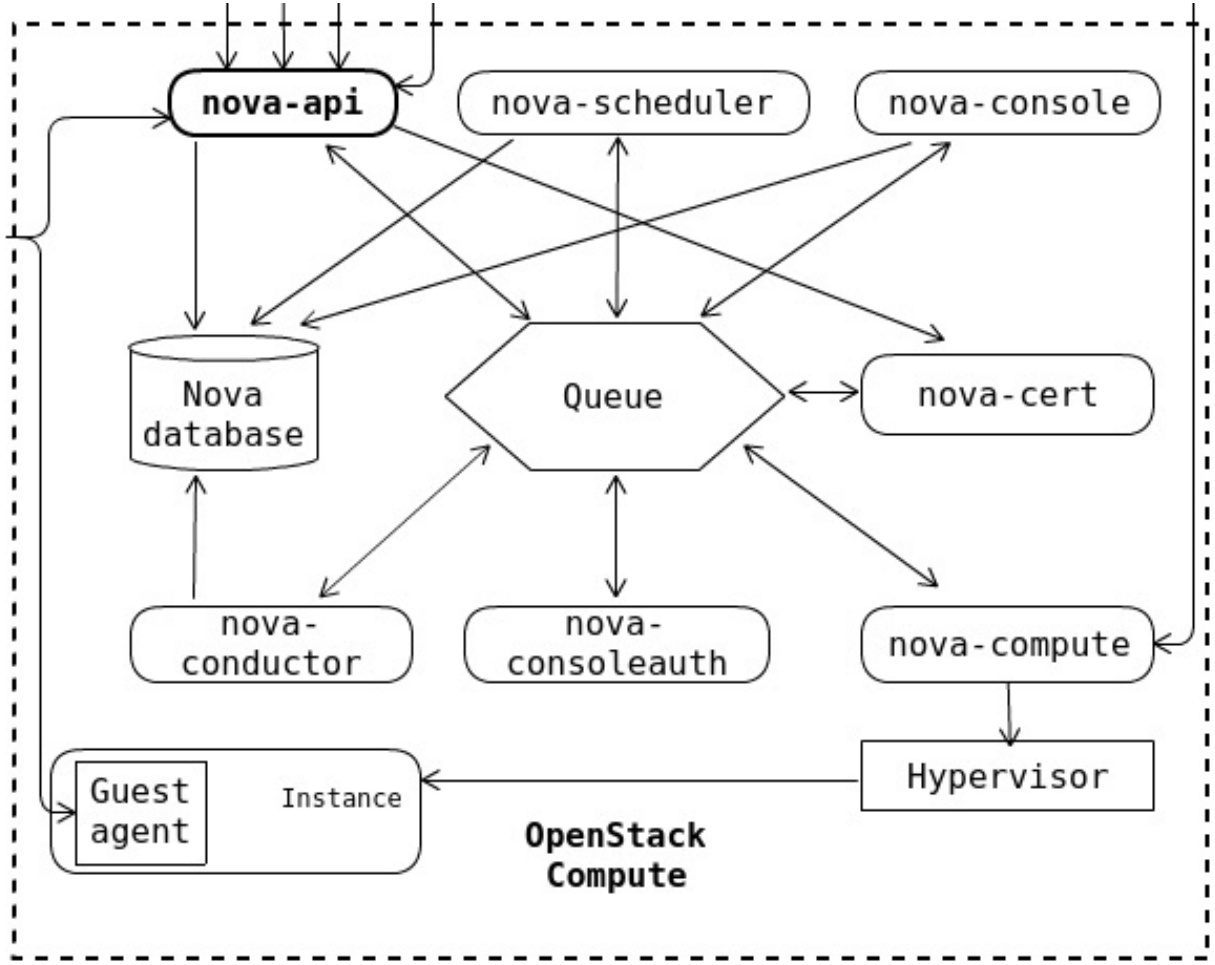
OpenStack Compute

- OpenStack Compute (Nova) is a cloud computing fabric controller (the main part of an IaaS system)
- Provides a highly scalable management framework for virtual machines
- Designed to manage and automate pools of compute resources
- Supporting wide variety of virtualization technologies (KVM, XEN, LXC, VMWare, ...)
- Scale up and down the infrastructure to meet demand
- Written in Python and uses many external libraries such as Eventlet (concurrent programming), Kombu (for AMQP communication), and SQLAlchemy (for database access)

Nova Logical Architecture

- Two types:
  - Web Services Gateway Interface (WSGI) applications to provide API
    - nova-api
  - Worker daemons to carry out orchestration tasks
    - nova-compute
    - nova-scheduler
    - Nova-conductor, etc
- Nova components use AMPQ based queues for inter process communication and a SQL database for information sharing

Nova Architecture



From OpenStack Folsom release nova-volume and nova-network separated as Cinder and Neutron respectively.

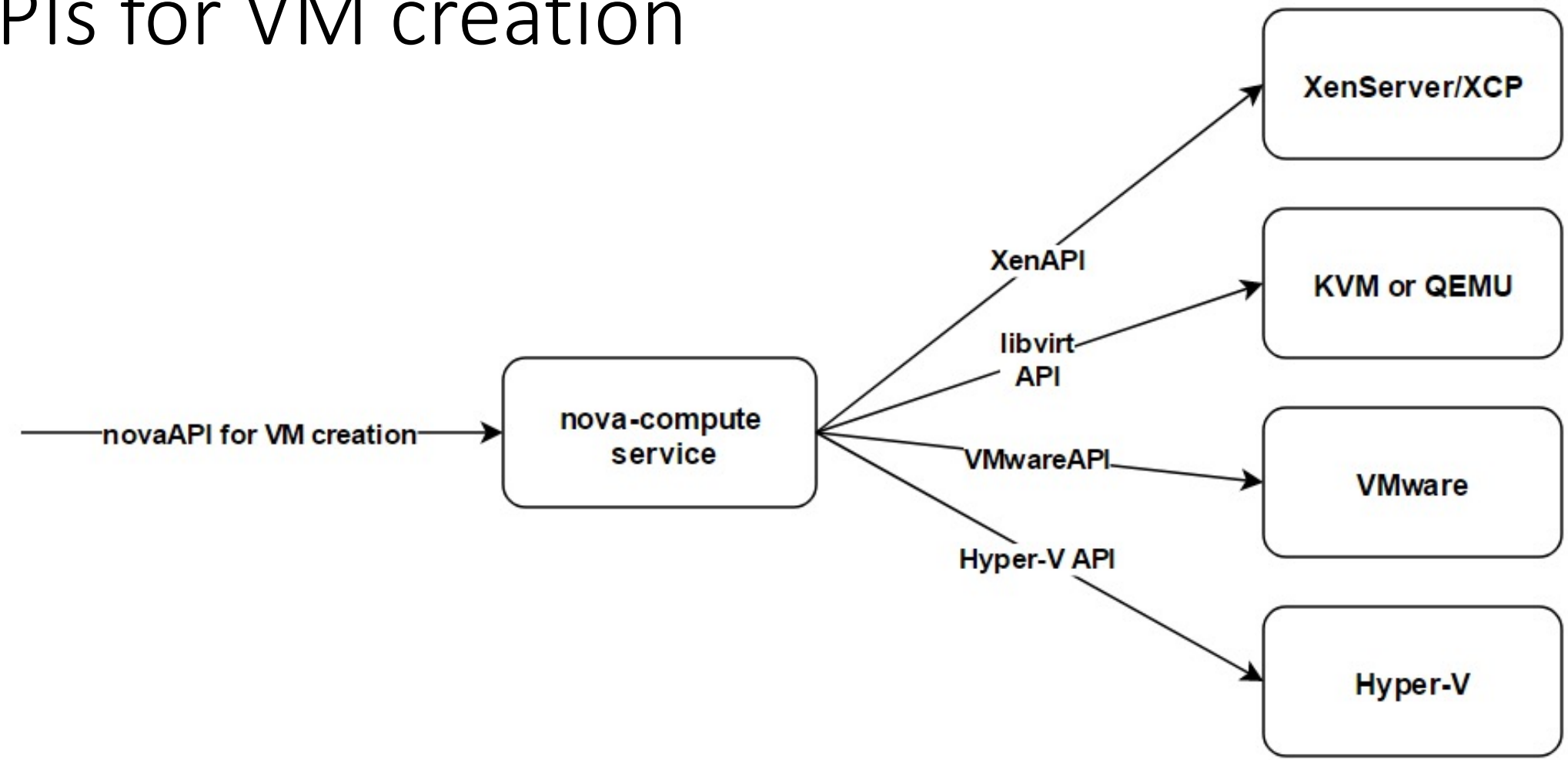
nova-api

- Nova provides a REST API for client interaction
- Nova expects you to be authenticated against Keystone and be able to provide a valid authentication token
- The nova-api process takes care of initiating most orchestration activities, such as provisioning new virtual machines

nova-compute

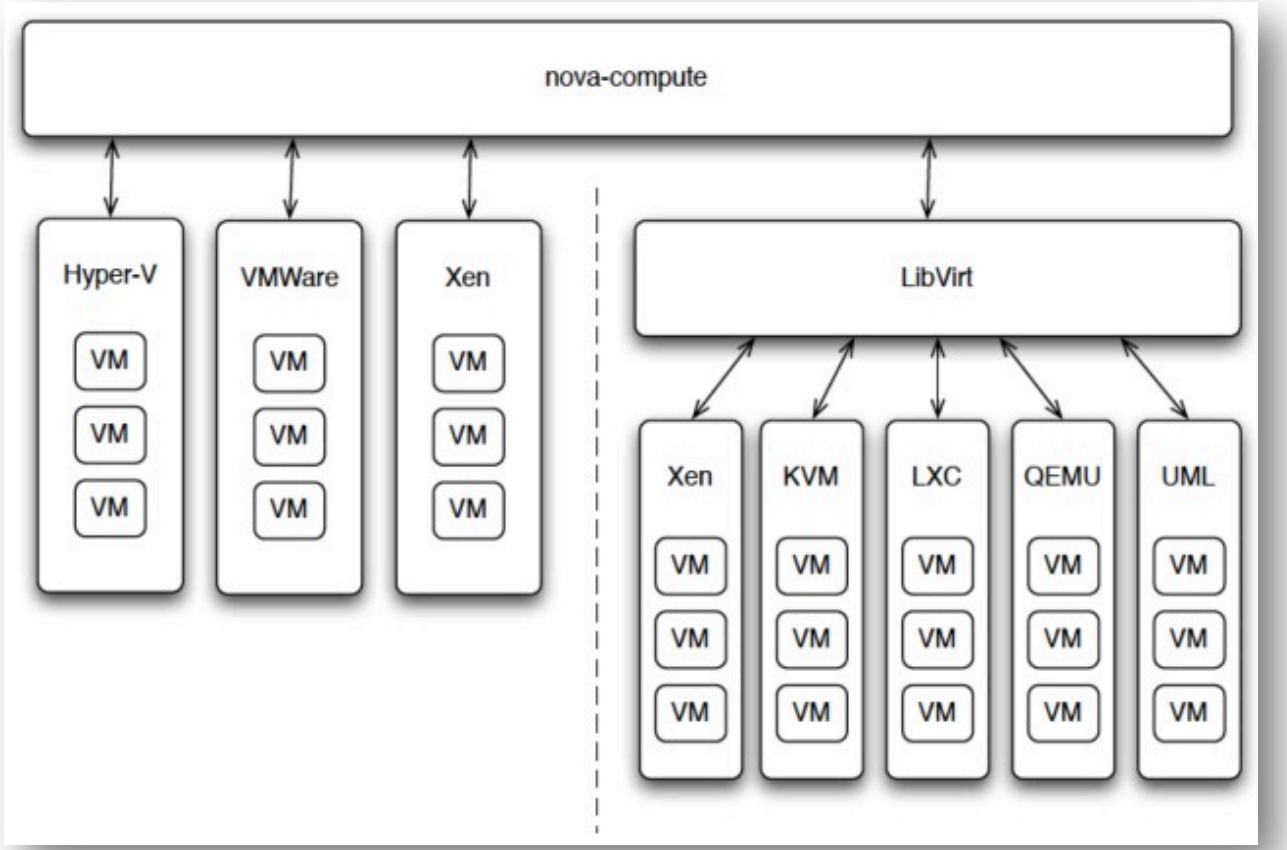
- nova-compute is primarily a worker daemon that creates and terminates virtual machine instances
- The entire lifecycle of the virtual machine is managed by the hypervisors
- Accepts the actions from queue, performs a series of system commands (e.g. launching a KVM instance) and updates the state in the database
- Flexibility to use multi-hypervisor environments in the same setup (KVM, Vmware, etc)
- the nova-compute process will take care of triggering the suitable APIs for the hypervisors to manage the virtual machine lifecycle.

APIs for VM creation



nova-compute

- Runs on each compute node to create and terminate virtual instances
- The compute service interacts with the hypervisor to launch new instances, and ensures that the instance state is maintained in the Compute database



nova-conductor

- New service introduced in the Grizzly release (April 2013)
- Enables OpenStack to function without compute nodes accessing the database (without it, if one of the compute nodes is compromised, the attacker has almost full access to the database)
- Places where nova-compute previously performed database access are now talking to nova-conductor
- Nova-conductor provides benefits of security, ease of upgrade and performance

For example, if there is a need to update certain state of a VM instance in nova-compute, instead of connecting to the database directly, make a RPC call to nova-conductor, which connects to the database and makes the actual DB update.

nova-consoleauth

- The nova-consoleauth process takes care of authorizing the tokens for the end users, to access a remote console of the guest virtual machines provided by the following control proxies:
  - The nova-novncproxy daemon provides a proxy for accessing running instances through a VNC connection
  - The nova-spicehtml5proxy daemon provides a proxy through a SPICE connection

SPICE (Simple Protocol for Independent Computing Environments) is a communication protocol for virtual environments. It allows users to see the console of virtual machines (VM) from anywhere via the Internet. It is a client-server model that imagines Virtualization Station as a host and users can connect to VMs via the SPICE client.

Nova Concept - Flavors

- A flavor is an available hardware configuration for a server.
- Flavors define the configuration of the instance, including:
  - Number of Virtual CPUs
  - Amount of RAM
  - Size of ephemeral disks
- OpenStack provides number of predefined flavors which cloud administrators may edit or add to

Flavors

By default, each size (generally) doubles the previous one

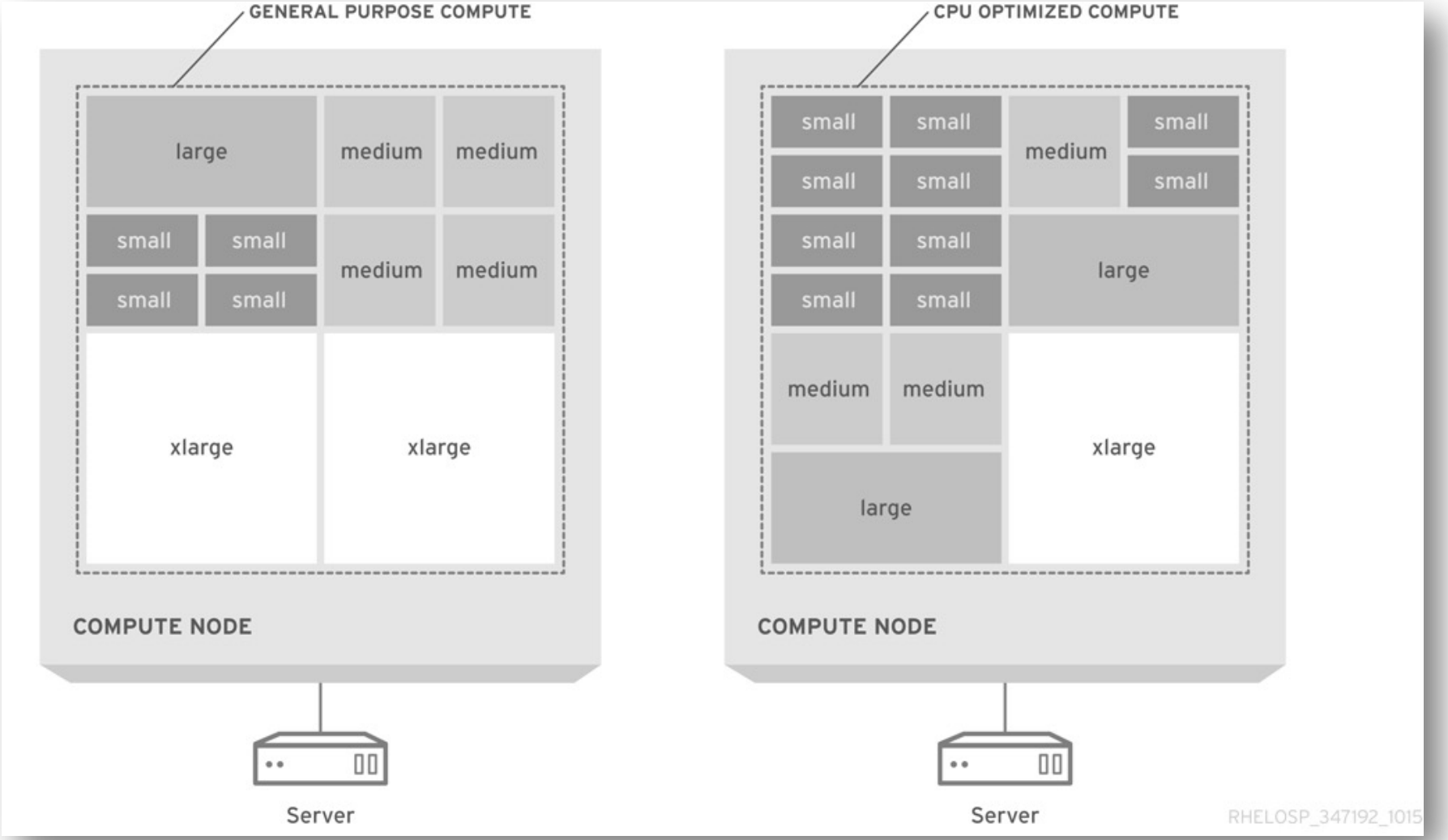


Image courtesy of RedHat

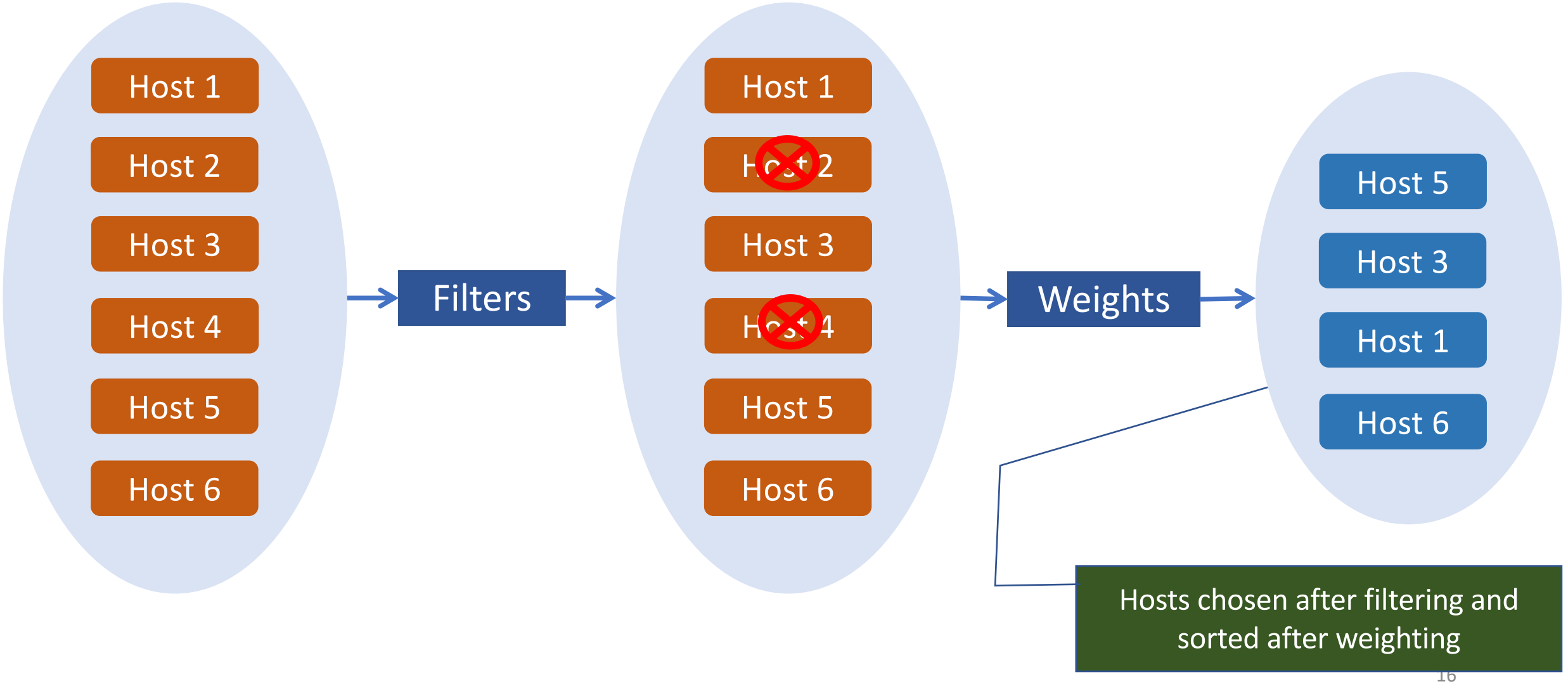
nova-scheduler

- Nova uses nova-scheduler service to determine how to dispatch compute and volume requests
- A Host is a physical or virtual machine that has a nova-compute service running on it
- All hosts periodically publish their status, available resources and hardware capabilities to nova-scheduler through the queue
- nova-scheduler then collects this data and uses it to make decisions when a request comes in
- Interacts with other components through rabbitmq queues and central Nova database

nova-scheduler: Filter Scheduler

- Filter Scheduler
  - The Filter Scheduler supports filtering and weighting to make informed decisions on where a new instance should be created.
- Numerous filters available
  - Are in the requested availability zone (AvailabilityZoneFilter).
    - Have sufficient RAM available (RAMFilter).
    - Are capable of servicing the request (ComputeFilter).
    - Schedule the instance on a different host from a set of instances (DifferentHostFilter)
- Provides ability to create custom filters in Python

nova-scheduler: Filter Scheduler



Nova Scheduler – Examples of Filters and Weights

- Examples of Filters:
  - **AllHostsFilter**: No filtering
  - **ComputeFilter:** Is the specific host active?
  - **CoreFilter:** Are there enough vCPU's available?
  - **RAMFilter:** Is there enough vRAM available?
  - **ImagePropertiesFilter:** Does the host support the image requirements?
- Example of Weights:
  - Prefer a host with the most free vRAM
  - Prefer a host with the most free CPU cycles
- Hints
  - Example: Tell Nova to launch on a different host than another instance

```
$ nova boot --image cedef40a-ed67-4d10-800e-17455edce175 --flavor 1 \
--hint different_host=a0cf03a5-d921-4877-bb5c-86d26cf818e1 \
--hint different_host=8c19174f-4220-44f0-824a-cd1eeef10287 server-1
```

Scale Nova Compute Resources

Segregation Helps Scale in Large Production Environments

- Enable Logical Groupings
  - Geographical
  - Specific Data Centers
  - By Rack, Power Sources, networks etc
- Group by specific Capabilities
  - Faster CPU's or NIC's
  - Storage requirements
  - Specific devices

Use these capabilities according to your own environment!

Regions

- Share or separate any OpenStack services as needed
  - Example. Share Keystone and Horizon. Separate Nova and other core projects by Data Center
- Targetable API endpoints with unique compute, storage and networks
- By default, all OpenStack services are placed in “RegionOne”

```
keystone endpoint-create --region "RegionTwo" ...
```

- Run OpenStack commands per region

```
nova --os-region-name "RegionTwo" boot
```

* Alternate syntax using openstack client: “openstack endpoint create”

Host Aggregates

Assign Metadata to Specific Hosts

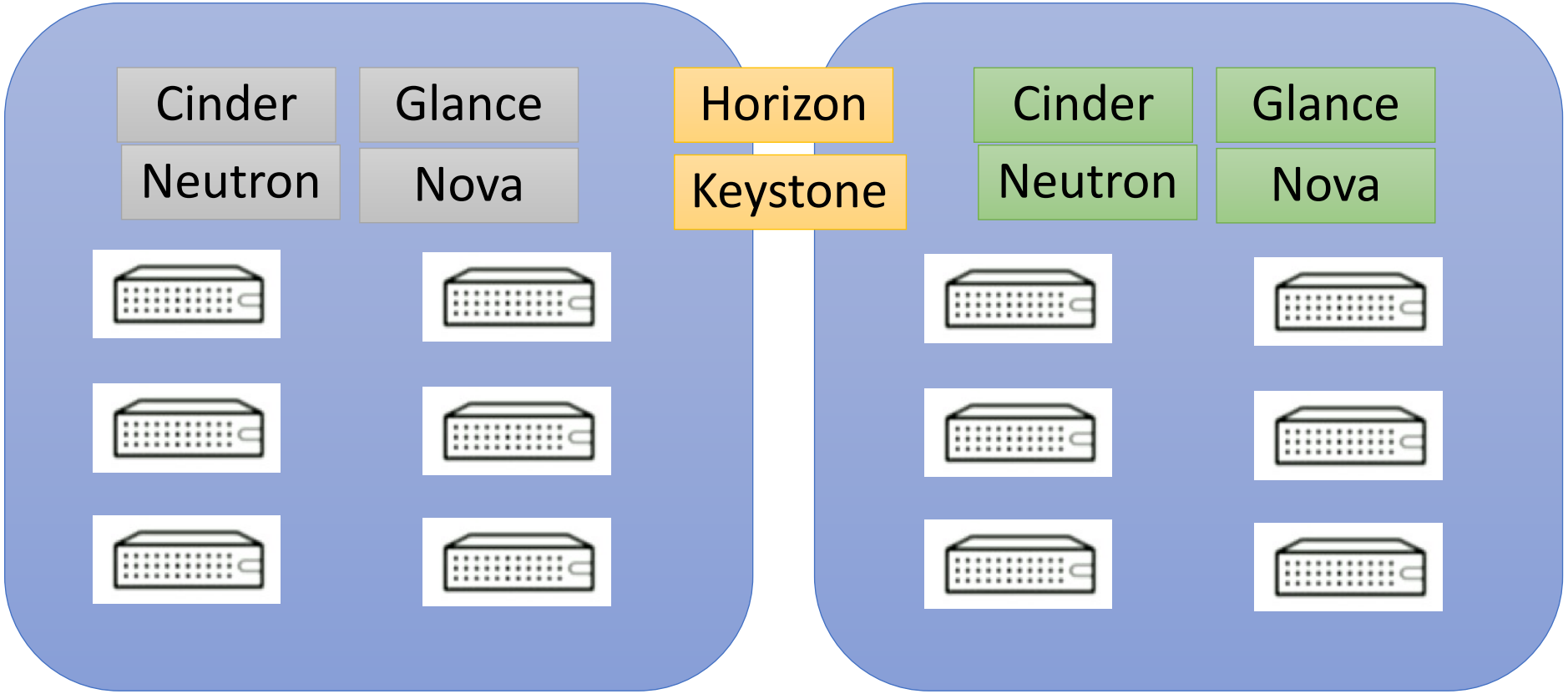
- Attach a label to a host or group of hosts
  - Example: "SSD Hosts". Ensure your instance only boots on nodes with SSD's
  - Example: "PCI Pass through"…
  - Example: "Node1", Node2",…
- Nodes are not limited to a single Host Aggregate
  - Using the above examples, Node1 might be in "Node1", PCI Pass through", *AND* "SSD Hosts" at the same time
- To boot an instance to a desired Host Aggregate – Filter Scheduler eliminates all other hosts at time of launch

Availability Zones

Assign Metadata to Specific Hosts ***BUT NO OVERLAP ALLOWED***

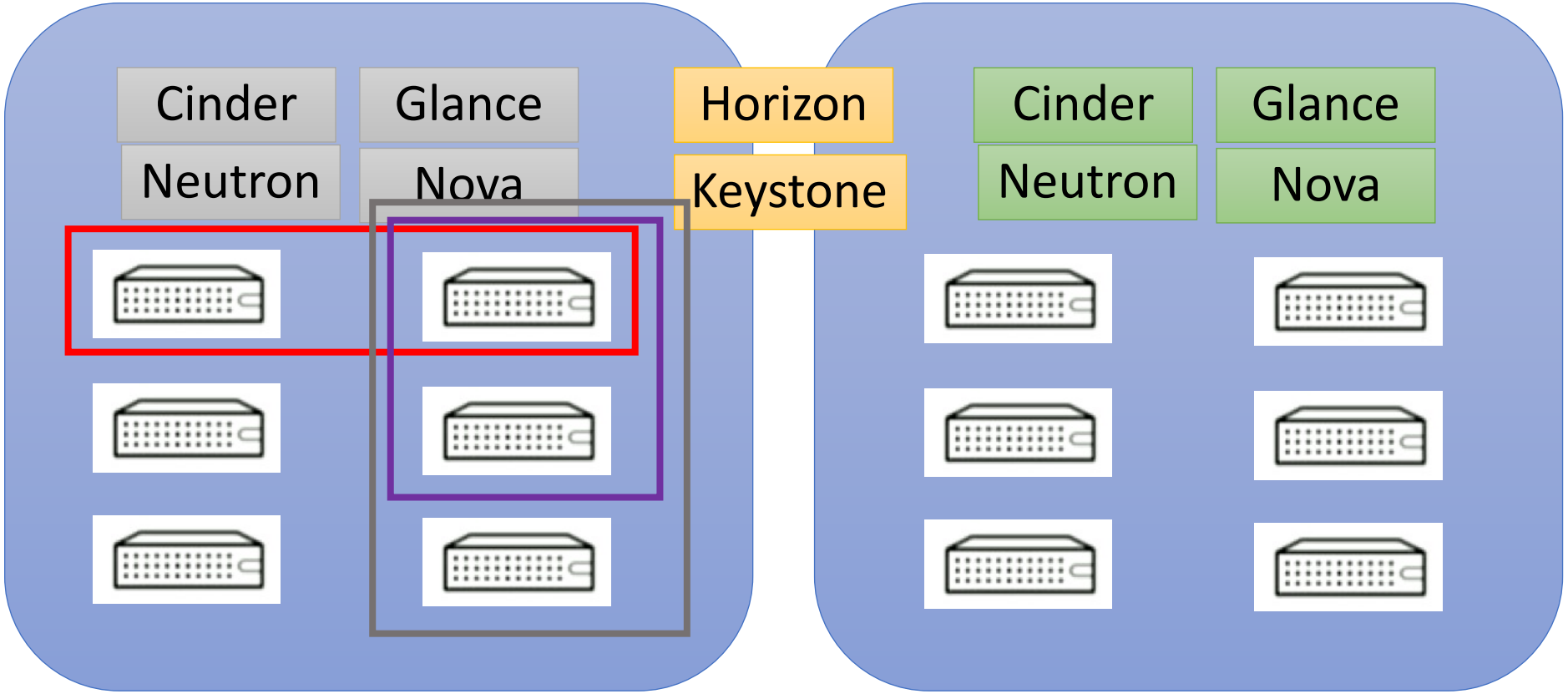
- Attach a label to a host or group of hosts
  - Example: “SSD Hosts”. Ensure your instance only boots on nodes with SSD’s
  - Example: “PCI Pass through”…
  - Example: “Datacenter1”, Datacenter2”,…
- Like Host Aggregate, when booting an instance to a desired Availability Zone – Filter Scheduler eliminates all other hosts at time of launch

Regions


Cinder
Glance
Neutron
Nova
Horizon
Keystone
Cinder
Glance
Neutron
Nova

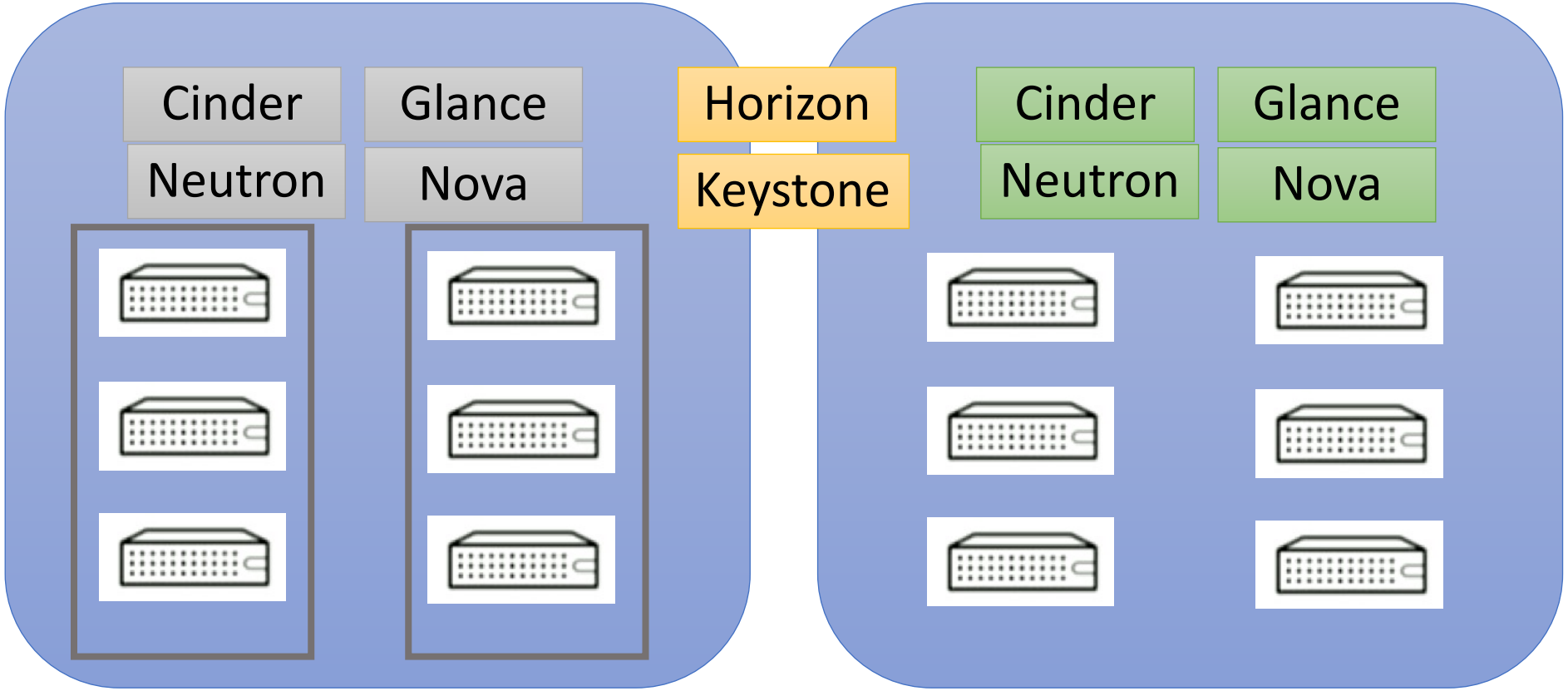
Host Aggregates

- **RED: SSD Drives** **BLACK: >med CPU** **Purple: > High CPU**



Availability Zones

- **Rack 1 vs Rack 2**



Nova Backend Hypervisor Support

- KVM
- Xen
- VMware (VCenter)
- HyperV
- LPAR
- Docker/LXC
- Via Ironic: Bare Metal

Compute Server Architecture Overview

- Considerations to be taken into account:
  - number of processors, amount of memory, network requirements, and the quantity of storage required for each hypervisor
  - any requirements for bare metal hosts provisioned through ironic
  - compute resources will be provided in a single pool or in multiple pools or availability zones? Use host aggregates?
  - For NFV or HPC based clouds, there may even be specific network configurations that should be reserved for those specific workloads on specific compute nodes

Horizontal Scaling	Add compute nodes to an OpenStack cloud	Vertical Scaling	Upgrade CPUs with more cores or increase overall server memory on compute nodes

Some Design Decisions

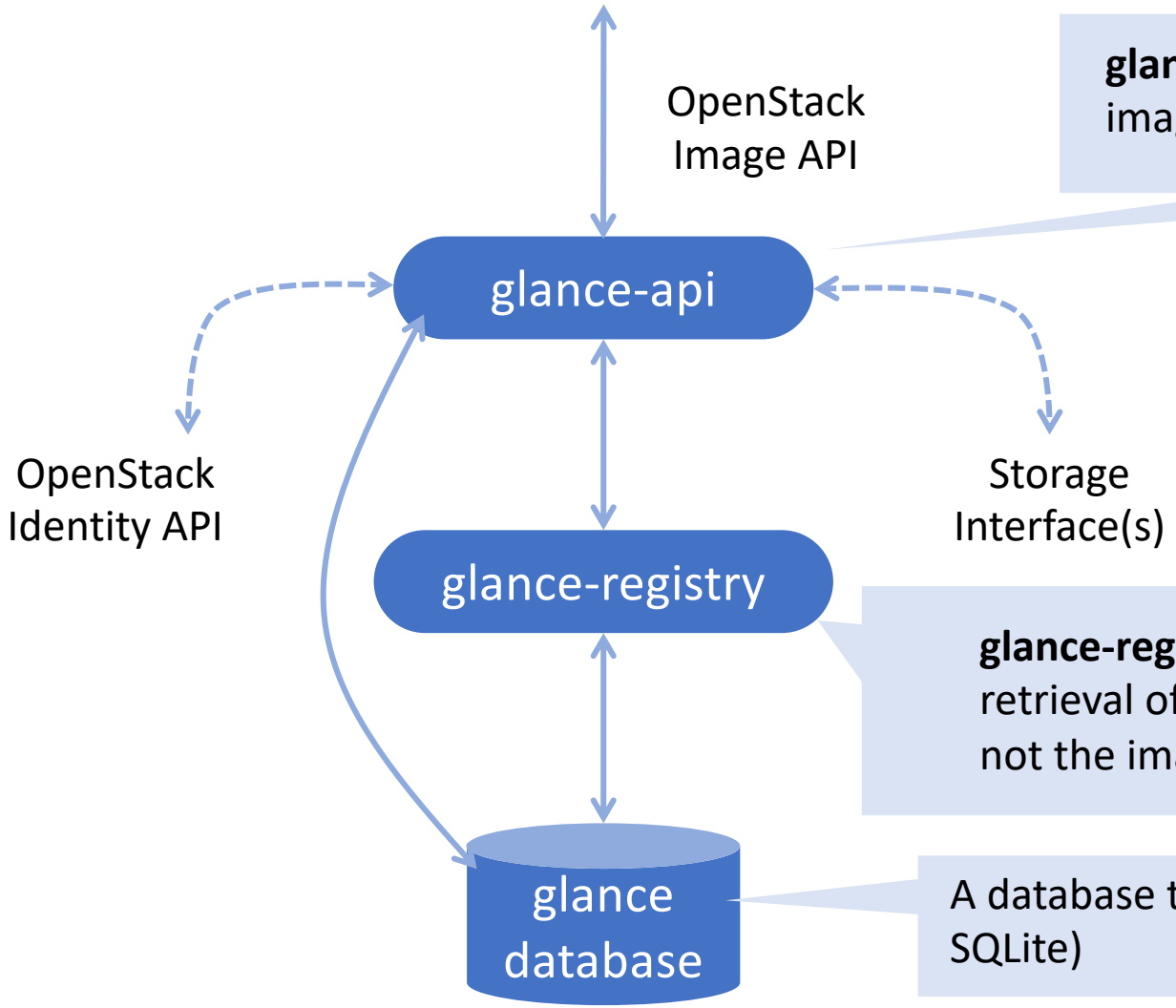
- CPUs should support virtualization (VT-x Intel chips , AMD-v AMD chips)
- Number of cores per server = # of cores / CPU * 2 (hyper-threading) * # of CPUs
- Overcommitting CPU (16:1) and RAM (1.5:1) – default values
- When CPU pinning is requested for a guest, it is assumed there is no overcommit (or, an overcommit ratio of 1.0)
- Server density (#servers / RU), Resource capacity (#CPU, #RAM, #storage / server)
- It is possible to run multiple hypervisors in a single deployment using host aggregates (KVM is the most widely adopted hypervisor in the OpenStack community)

Image Service - Glance

OpenStack Imaging Service

- OpenStack imaging service codenamed as Glance
- Glance provide services for discovering, retrieving and storage of virtual machine images
- REST API enabled
- Image can be stored on local file system or OpenStack based distributed storage

Glance Architecture



glance-api accepts Image API calls for image discovery, image retrieval and image storage

A storage repository for the actual image files. In addition to Swift, Glance supports normal filesystems, RADOS block devices, Amazon S3 and HTTP

glance-registry deals with the storage, processing and retrieval of metadata about images (size, type, etc.), not the image itself

A database to store the image metadata. Most used: MySQL or SQLite)

Glance

- Both public and private images can be stored and retrieved
- List of available images and its configurations can be obtained
- Snapshot can also be managed (migrate from snapshot to image)
- Nova interacts with Glance to replicate images to compute hosts

Images and Instances

- Virtual machine images contain a virtual disk that holds a bootable operating system on it
- The Image service (Glance) controls image storage and management
- Users can launch any number of instances from the same image: each launched instance runs from a copy of the base image
- Snapshots capture the state of an instances running disk
- Users can create a snapshot, and build a new image based on these snapshots
- The Compute service (Nova) controls instance, image, and snapshot storage and management

System state prior launching an instance

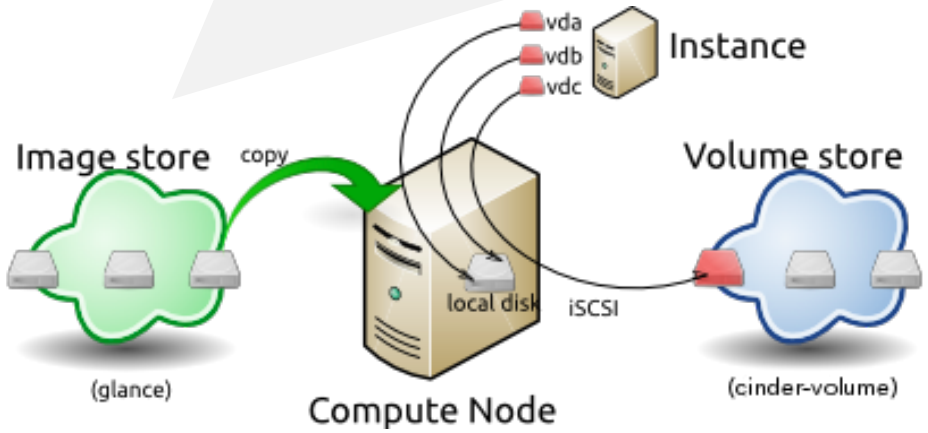
The image store has a number of predefined images, supported by the Image service

the **cinder-volume** service stores predefined volumes

a compute node contains the available vCPU, memory, and local disk resources

Instance launch

The Image service copies the base image from the image store to the local disk. The local disk is the first disk that the instance accesses, which is the root volume labeled **vda**



The compute node connects to the attached cinder-volume using iSCSI. The cinder-volume is mapped to the third disk, labeled **vdc** in this diagram

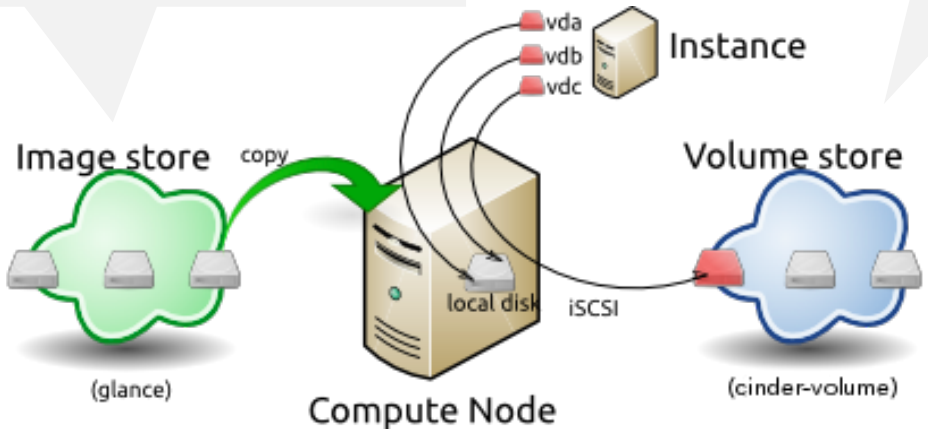
The new empty ephemeral disk is also created, labeled **vdb**. This disk is deleted when you delete the instance

After the compute node provisions the vCPU and memory resources, the instance boots up from root volume **vda**. The instance runs and changes data on the disks

Delete an instance

The image remains unchanged throughout this process

The persistent volume is maintained in its state



The ephemeral storage, whether encrypted or not, is purged. Memory and vCPU resources are released

- In OpenStack, the base operating system is usually copied from an image stored in the OpenStack Image service. This results in an ephemeral instance that starts from a known template state and loses all accumulated states on shutdown.
- You can also put an operating system on a persistent volume in Compute or the Block Storage volume system. This gives a more traditional, persistent system that accumulates states that are preserved across restarts.

Glance Replicator

- Glance Replicator is an in-built tool used to populate a new glance server using the images stored in an existing glance server
- The images in replicated glance servers preserve uuid, metadata and image data from original.

Glance replicator command	Description
compare	What is missing from the slave glance?
dump	Dump the contents of a glance instance to local disk.
livecopy	Load the contents of one glance instance into another.
load	Load the contents of a local directory into glance.
size	Determine the size of a glance instance if dumped to disk.

Supported Backend Storages

- File system
- Swift
- Amazon S3
- RADOS Block Device – CEPH storage cluster using RBD interface
- Cinder

In the default configuration Glance uses the file storage backend.
This means that uploaded disk images are stored in
the **/var/lib/glance/images/** directory on the **local file system** of the Glance server.

Disk Format of a VM

The disk format of a virtual machine image is the format of the underlying disk image. Virtual appliance vendors have different formats for laying out the information contained in a virtual machine disk image.

Type	Description
raw	an unstructured disk image format
vhd	a common disk format used by virtual machine monitors from VMWare, Xen, Microsoft, VirtualBox, and others
vmdk	common disk format supported by many common virtual machine monitors
vdi	supported by VirtualBox virtual machine monitor and the QEMU emulator
iso	An archive format for the data contents of an optical disc (e.g. CDROM).
qcow2	A disk format supported by the QEMU emulator
aki	Amazon kernel image
ari	Amazon ramdisk image
ami	Amazon machine image

Container Format - VM + Metadata

The container format indicates whether the virtual machine image is in a file format that also contains metadata about the actual virtual machine

Type	Description
bare	indicates there is no container or metadata envelope for the image
ovf	OVF Container format
aki	Amazon Kernel Image
ari	Amazon ramdisk Image
ami	Amazon Machine Image
ova	OVA tar archive

Images

- Easiest way is to download an image – repositories exist
- Most of the images contain the cloud-init package to support the SSH key pair and user data injection
- Image creation (other than snapshots) is principally done outside of OpenStack: image created manually on a system and then upload the image to the cloud
- Common model is:
  - Start with an ISO or physical media for a guest OS
  - Install via a virtualization manager on a laptop (VMware, VirtualBox, etc.)
- Boot an instance from an ISO image (and use the new image for future instances)

Conversion between Image Formats

- Conversion between image formats are done through set of commands based on the image formats.

Tool	
qemu-img convert	raw, qcow2, VDI (VirtualBox), VMDK (VMware) and VHD (Hyper-V)
VBoxManage	VDI (VirtualBox) to raw