**Openstack – Nova Compute Components and Other OpenStack Terms**

**Keystone:**

Keystone provides Identity, Token, Catalog, and Policy services for use specifically by projects in the OpenStack family

**Keystone terms:**

* User:
  + Digital representation of a person, system, or service who uses OpenStack cloud services
* Credentials
  + Data that is known only be a user that proves who they are such as using username and password, username and API key, auth token
* Token
  + Identifying credential associated with a user, an arbitrary bit of text that is used to access resources
* Group
  + Collection of users
* Project
  + A container used to group or isolate resources and/or identity objects such as end-points, running instances, quotas, and more
* Domain
  + Represents a distinct set of individual users, user groups, and projects
* Role
  + A first-class piece of metadata associated with the user for the tenant, assigned directly to users or groups for projects or inherited from domains
* Service
  + An OpenStack service, such as Nova (Compute), Swift (Object Storage), or Glance (Image Service) and provides one or more endpoints through which users can access resources and perform operations
* Endpoint
  + A network-accessible address, usually described by URL, from where you access an OpenStack service
* Rule
  + A set of requirements for performing an action over the endpoint

**Nova API – this is what manages the hypervisors and other Compute pieces:**

* Nova API is a RESTful API web service which is used to interact with Nova
* Provides way to manage multiple APIs on different sub-domains
  + EC2-compatability – note that this is being deprecated
  + Compute API – This is where all the innovation and development is happening in OpenStack
* Nova API exposes REST API via HTTP
* Nova API is stateless and is ready for HA (high availability) deployments

**Nova Database – stores current state of all objects in the compute cluster**

* Usually MySQL (which we will be using) and PostgreSQL can also be used
* Nova API talks to the MySQL Nova DB via SQLAlchemy
  + <http://www.sqlalchemy.org/>
* Nova Database HA is not handled by OpenStack. You must use third party external tools such as Galera or Multi-Master replication Model for MySQL (MMM)
  + <https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster-replication-multi-master.html>

**Message Queue**

* Message queue (RabbitMQ) is a unified way for collaboration between all of the sub-components
* Can use multiple queues within a single RabbitMQ instance
  + Used by the OpenStack services to build machine state
  + Each compute node has a queue
* Message traffic is not intensive
* Does not send broadcast messages
  + Monitor using API polling
* HA must be configured separately from OpenStack such as using mirrored queues that are not handled with OpenStack natively
  + OpenStack uses the following Message Queue nodes:
    - Rpc.cast – Don’t wait for result (fire and forget)
    - Rpc.call – Wait for result (When there is something to return)
* How the queue works (This explanation needs improved)
  + *Nova API creates or updates an entry in the Nova Database*
  + *The Nova API sends a JSON message with receiver and entry ID to the RabbitMQ queue*
  + *The Nova Scheduler listens to queue channels, grabs the message when it casts onto a channel, unpacks the entry ID and processes it in the Nova Database*
  + *Nova Scheduler gets entry by the entry ID from the Nova Database for further processing*

**Nova Scheduler**

* Nova Scheduler is a daemon, which determines, on which compute host the request should go
  + This component only does provisioning time and is not like VMware’s DRS (?)
  + This daemon service is usually located on the Cloud Controller host
* Scheduler types:
  + Chance – picks a host that is available
  + Filter – Picks the best-suited host that satisfies selected filter(s)

**Nova Compute**

* Nova Compute is a worker daemon, which primarily creates and terminates virtual machines via Hypervisor API
  + Allows multiple hypervisor types per cloud
  + Libvert / KVM is most commonly used in deployments
  + New in Kilo we can also use Nova Compute to create Bare Metal servers as well

**Nova Conductor: Middle man between Nova Computer and Nova Database**

* Nova Conductor is called from the Nova Compute for Virtual Machine information from the database
  + Nova Compute does an rpc.call to Nova Conductor
* The Nova Conductor service is key to completing the no-db-compute
  + Eliminates remote DB access for better security
  + Horizontal scalability – spawn multiple worker threads operating in parallel
  + Hides DB implementation/schema from the Nova Compute
    - This allows upgrading Nova Compute in the future without taking down production servers
  + Possible offloading of long-running operations from other services, not just Nova Compute
  + Nova Conductor is beneficial as well for operations that cross multiple compute nodes for future migrations