Linux Academy
Study Guide

OpenStack Essentials

Contents

Welcome to OpenStack

About the Instructor

- Cloud Virtualization Admin for Rackspace
- RedHat Certified Systems Administrator
- 5 years Linux and OpenStack experience
- Self-identified OpenStack nerd
- Loves cartoons, animals, cartoon animals, computers, stickers, and fashion!
- Great dancer and a hit at parties (not really)
- Twitter @OGtrilliams || LinkedIn @trilliams || #LinuxAcademy on Freenode @PagliaccisCloud

What to Know Before Beginning

About This Course

- This course is targeted towards those who are new to OpenStack technology
- We will cover the origins of OpenStack technology, the OpenStack community, and resources available for those interested in expanding their learning.
- By the end of this course you will have a deep fundamental understanding of the core components needed to build and manage a public or private cloud environment.

Course Prerequisites

- Linux systems administration
  - CLI - command line interface
  - Directory structure and navigation
- Networking
  - Ethernet
  - VLANs
  - Routing
  - IP addresses
- Optional: Virtualization and hypervisor technology
  - Citrix Xen

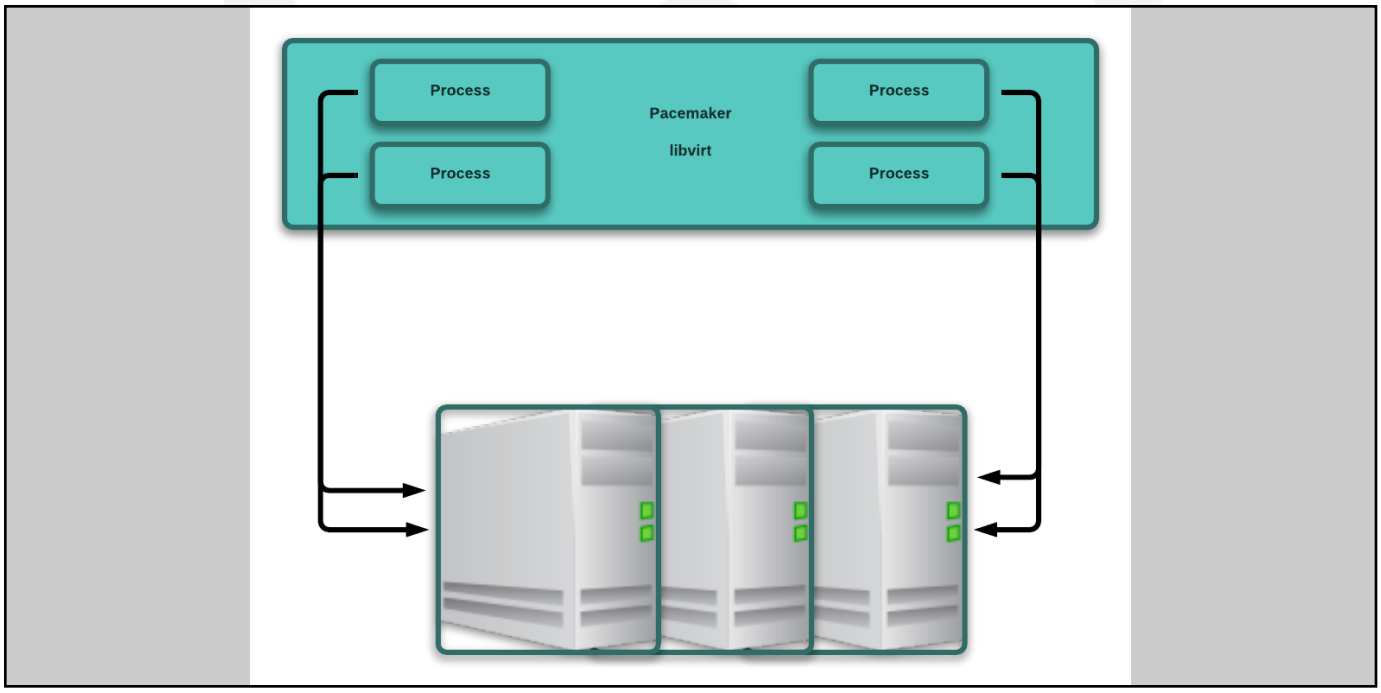
- VMWare
- VirtualBox
- KVM

End Goals

- A core understanding of OpenStack
- A conceptual knowledge of OpenStack services covered in this course:
  - Keystone (Identity)
  - Horizon (Dashboard)
  - Nova (Compute)
  - Neutron (Networking)
  - Glance (Imaging)
  - Swift (Object Storage)
  - Cinder (Block Storage)

About OpenStack

What is a Cloud?

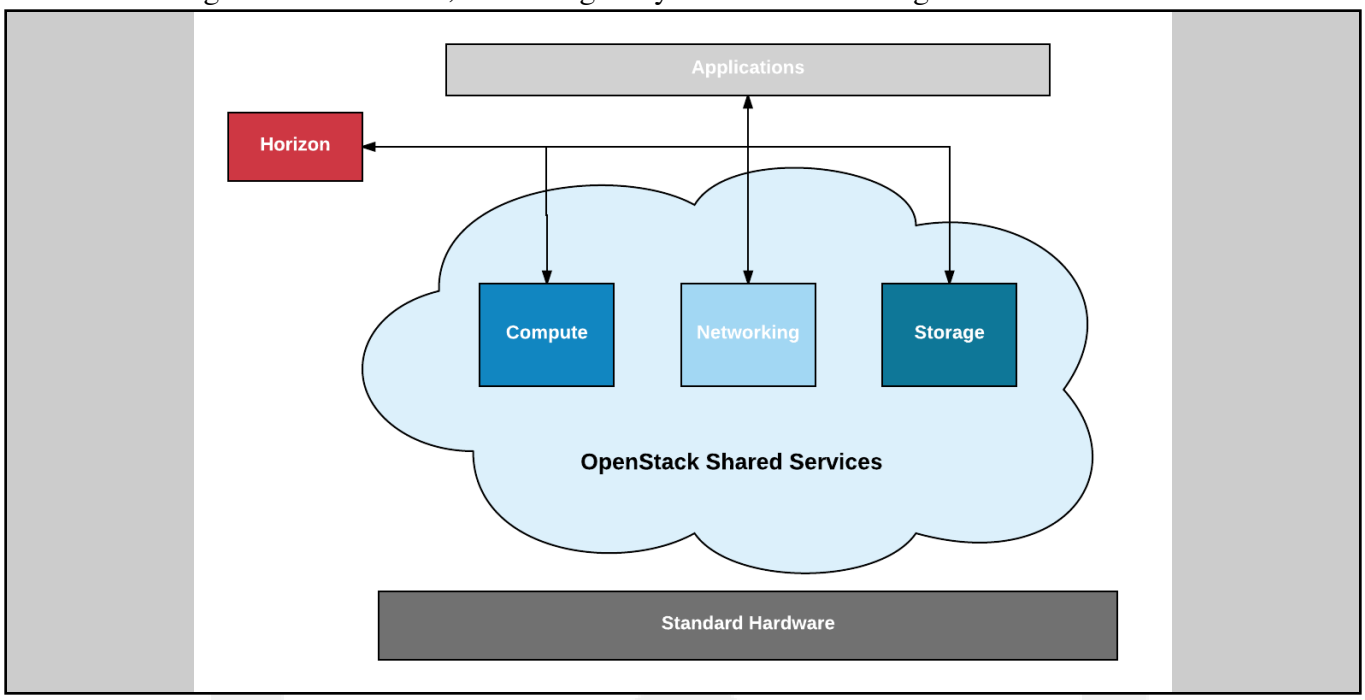


In a Linux environment, virtualization architecture most likely contains:

- Centralized storage
- SAN
- Virtualization nodes

What is OpenStack?

OpenStack is an open source cloud platform that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed by a dashboard that gives administrators control while



empowering users to provision resources through a web interface.

- A collaboration between NASA and Rackspace started in 2010
- Written mostly in Python and free under the Apache 2.0 license
- Turns hypervisors in a datacenter, or across several datacentrs, into a pool of resources that can be accessed from a centralized location.
- OpenStack can be viewed as the control layer that sits above other virtualized resources in your datacenter.
- You can run many types of hypervisors, from KVM to Hyper-V, but the way you access them all is the same. That's the magic of OpenStack.
- A framework that allows administrators to efficiently and dynamically manage virtualization and storage so users can request computing power, networks, block storage, etc. as needed.

Use Cases

OpenStack is being run by research institutions, government agencies, financial institutions, e-commerce

and media companies, and biomedical research and pharmaceutical companies globally in both public and private cloud environments. Notable users include:

- Walmart
- PayPal
- eBay
- Twitter
- CERN
- Pinterest
- Overstock.com
- Rackspace
- Ubuntu
- And many more!

OpenStack Modules

- **Daemon** • Daemons run as background processes. On Linux platforms, a daemon is usually installed as a service.
- **Script** • Installs a virtual environment and runs tests
- **Command line interface** • CLI enables users to submit API calls to OpenStack services through commands.

OpenStack Origins

RAX/Slicehost, NASA, OpenStack Foundation, Contribution, and Expansion

OpenStack Origins

- In an effort to consolidate their web space into a unified platform, NASA started Nebula in 2008 under the project name nasa.net.
- NASA noticed the similarities between the Nova controller and storage systems developed by Rackspace as they began development for a compute controller.
- In July 2010, Rackspace Hosting and NASA jointly launched an open source cloud software

initiative known as OpenStack, intended to help organizations offer cloud computing services running on standard hardware.

- The community's first official release, codename Austin, appeared four months later with plans to release regular updates of the software every few months.

NASA's Departure

- In July 2013, NASA released an internal audit citing lack of technical progress and other factors as the agency's primary reason for dropping out as an active developer of the project.

Citing the increase of contributors and popularity, NASA decided to instead focus on the use of public clouds.

OpenStack Foundation

Created in 2012, the OpenStack Foundation promotes the global development, distribution, and adoption of the OpenStack cloud operating system.

- Serves over 60,000 individual members from over 180 countries
- The foundation's mission statement is "...to serve developers, users, and the entire ecosystem globally by providing a set of shared resources to grow the footprint of public and private OpenStack clouds, enable technology vendors targeting the platform and assist developers in producing the best cloud software in the industry."

Board of Directors

The OpenStack Foundation Board of Directors is comprised of 24 total members elected to protect, empower, and promote OpenStack software and the community around it, including users, developers, and the entire ecosystem.

- 8 platinum members appointed by members
- 8 gold members elected by member class
- 8 individual members elected by individual members

Technical Committee

The OpenStack Technical Committee members define and steward the technical direction of OpenStack software, including cross-program issues.

- 13 individuals fully elected by the project's active technical contributors
- Members are affiliated with Red Hat, Mirantis, IBM, Rackspace, and Intel in addition to OpenStack Foundation

User Committee

Represented by Tim Bell, OS and infrastructure services group leader for CERN, the User Committee was created to represent a broad set of enterprise, academic, and service provider users with the Technical Committee and Board of Directors.

- User Committee Mission Statement:
    - Consolidate user requirements and present these to the management board and technical committee
    - Provide guidance for the development teams where user feedback is requested
    - Track OpenStack deployments and usage, helping to share user stories and experiences
    - Work with the user groups worldwide to keep the OpenStack community vibrant and informed.
- Currently led by a core group of 3 individuals who provide oversight and guidance to several working groups that target specific areas for improvement.

Foundation Membership

The OpenStack Foundation offers Platinum, Gold, Corporate, and Individual membership levels

Platinum membership

An initially set list of companies, but those interested can apply for future consideration if a spot opens. Must contribute $500,000 per year to the foundation and must have at least two full time employees contributing to OpenStack.

Current members:

- AT&T
- Ubuntu, sponsored by Canonical
- HP Enterprise
- IBM
- Intel
- Rackspace (also an infrastructure donor)
- Red Hat
- SUSE

Gold Membership

Organizations can apply for Gold membership, subject to board approval. Must make an annual contribution equal to .025% of their revenue, with a minimum contribution of $50,000 per year and maximum of $200,000 per year.

Current members:

- Cisco
- Ericsson
- NetApp
- Hitachi
- Huawei
- Mirantis
- Fujitsu
- Dell EMC
- China Telecom

Corporate Sponsorship

Corporate sponsorship is available at two levels, Startup, for companies less than 2 years old with less than $5 million dollars in revenue, for $10,000 per year. Companies older than 5 years can sign up for Established Company membership, for $25,000 per year.

- Access to OpenStack restricted use logos for commercial use and as long as product meets technical requirements
- Designation as Startup or Corporate sponsor with logo placement in supporter section of openstack.org
- Profile on openstack.org and a link to your site
- OpenStack Startup/Corporate sponsor badge for corporate website or marketing materials
- Opportunity for inclusion in OpenStack newsletter
- Ability to purchase “OpenStack” as a keyword for relevant advertising efforts consistent with brand guidelines
- Potential to promote qualifying products in OpenStack Marketplace
- Early access to selected Summit sponsorships
- Some current sponsors include:
    - Linux Academy

- VMWare
- Brocade
- GoDaddy
- PayPal
- Accenture
- F5 Networks
- Google Cloud Platform
- Verizon
- Plus dozens more!

Individual Membership

Individual membership has two account levels - Foundation Member and Community Member. Membership is free, but Foundation Membership comes with specific responsibilities.

- Submit on proposed Summit talks
- Vote on proposed Summit talks
- Profile on openstack.org
- Commit code to OpenStack via Gerrit - Foundation Level
- Vote on Foundation Board Member elections - Foundation level
- Foundation Members are required to vote on Foundation Board member elections or their membership can be suspended or moved to Community membership.
- New Foundation Members must be members for 180 days before they are eligible to vote in elections
- New members can sign up at https://www.openstack.org/join

OpenStack versions A-??

First Release

- The first release of OpenStack, codename Austin, was released October 21, 2010
- Included first iterations of Nova (Compute) and Swift (Object Storage) projects
- Supported by more than 35 corporate partners over three months
- Object Storage was production-ready, while Compute was intended for testing and limited deployment.

Typical Release Schedule

Under normal circumstances, OpenStack is developed and released around 6-month cycles.

- After initial release, additional stable point releases are released in each series.
- Ocata was released 4 months after Newton to address issues with stability

Current Release

- At the creation of this video, OpenStack **Ocata** (16) is the current release, launched officially on February 22, 2017.
- Ocata replaced Newton (15) 4 months after first release
- Designed as a stabilization release to address scalability and performance of OpenStack Compute and Networking
- Contributions from 1,925 developers, operators, and users from 265 organizations
- Networking and driver enhancements for Ironic
- Keystone-to-keystone federation support for Horizon
- Swift backend support for Zaqar messaging service

Who's Next?

As of the creation of this video, the next upcoming release will be **Pike** (16), currently under development.

How Do They Come Up with These Names?

Names for OpenStack releases are proposed and chosen by the OpenStack community through a Condorcet poll.

Names are taken from landmarks or distinguishing features related to the corresponding Design Summit. For example, Newton was named for a historical home in Austin, Texas - where the OpenStack design summit was held for 2016.

- Each potential release name must start with the letter of the basic Latin alphabet following the initial letter of the previous release (alphabetical order)
    - After "Z," the cycle will start over with "A"
- The name must be composed of only the 26 characters of the ISO basic Latin alphabet, or transliterated into this character set.
- The name must refer to the physical or human geography of the region encompassing the location of the OpenStack summit for the corresponding release.
- The name must be a single word with a maximum of 10 characters

- "Names which do not meet these criteria but otherwise sound really cool should be added to a separate section of the wiki page and the TC may make an exception for one or more of them to be considered in the Condorcet poll."

Commercial OpenStack: Canonical, RDO, Suse, etc.

Ubuntu OpenStack

- In 2011, developers of the Ubuntu Linux distribution adopted OpenStack with an unsupported technology preview of the OpenStack Bexar release for Ubuntu 11.04
- Ubuntu's sponsor, Canonical, then introduced full support for OpenStack cloud, starting with the Cactus release.
- Ubuntu OpenStack offers a fully integrated, optimized combination of Ubuntu LTS, OpenStack Icehouse - Kilo, and powerful tools to deploy manage and scale your cloud environment.
- Features 24/7 production grade support worldwide
- Contains OpenStack software validated through testing to provide API compatibility for core services
- Supports OpenStack Federated Identity, allowing it to connect to other OpenStack cloud environments for authentication and authorization.

Supported Hypervisors

- KVM
- Qemu
- LXC
- ESXi
- Hyper-V
- Supported guests:
  - Windows
  - Linux

Enabled Services

- Block Storage
- Compute API and extensions

- Dashboard
- Identity API and extensions
- Image Service API
- Networking API and extensions
- Object Storage API and Extensions
- Orchestration API
- Telemetry API

SUSE OpenStack Cloud

SUSE OpenStack Cloud is the enterprise private cloud solution of choice. Designed for HA, rapid deployment and ease of use, along with the widest hypervisor support and interoperability, it delivers the agility and control to drive innovation.

- In October 2011, SUSE announced the public preview of the industry's first fully configured OpenStack powered appliance based on the Diablo OpenStack release.
    - In August 2012, SUSE announced it's commercially-supported enterprise OpenStack distribution based on the Essex release.
- Built using open source software code, allowing clients to avoid vendor lock-in and control overhead costs
- Contains OpenStack software validated through testing to provide API compatibility for core services
- Supports OpenStack Federated Identity, allowing it to connect to other OpenStack cloud environments for authentication and authorization.

Key Features

- Latest release (SOSC7) based on OpenStack Newton
- Streamlined installation process
- Integrated with SUSE portfolio
- Production-ready and reliable
- Supports SUSE Linux Enterprise Support Pack 1
- Full support for Docker, using Kubernetes as the container orchestration framework delivered via the integration of OpenStack Magnum.
- Delivery of non-disruptive upgrade capabilities to avoid downtime and service interruption experienced when migrating to new OpenStack releases.

- Additional high availability capabilities, including HA protection for virtual machines and workloads to complement existing HA support for the control plane and compute nodes.
- Integration of OpenStack Manila with CephFS, through SUSE Enterprise Storage.

Enabled Services

- Block Storage API and extensions
- Compute API and extensions
- Dashboard
- Database API
- Elastic Map Reduce API
- Identity API and extensions
- Image Service API
- Networking API and extensions
- Orchestration API
- Telemetry API

BYOH (Bring Your Own Hypervisor)

- KVM
- Xen
- VMware vSphere
- Hyper-V

Supported Guests:

- Linux
- Windows

Red Hat OpenStack Platform

- In 2012, Red Hat announced a preview of their OpenStack distribution, beginning with the Essex release.
    - July 2013, after another preview release, Red Hat introduced commercial support for OpenStack with the Grizzly release.
- Uniquely co-engineered together with Red Hat Enterprise Linux to ensure a stable and production-ready cloud

Features

- **Built-in security** • SELinux military-grade security technologies
- **Managed cloud** • Comes integrated with Red Hat CloudForms, a unified management and automation system
- **No vendor lock-in** • Proprietary solutions are restricted to the providers' ecosystem.

Lifecycle support options:

- Long life versions (every 3rd release)
  - Offers standard 3-year lifecycle
  - Optional 1-2 years of extended lifecycle support

		Long life			Long life		
RHOSP 8 Liberty	RHOSP 9 Mitaka	RHOSP 10 Newton	RHOSP 11 Ocata	RHOSP 12 Pike	RHOSP 13 Queens	RHOSP 14 R....	RHOSP 15 S....
3 years	3 years	3 years (+2 years)	1 year	1 year	3 years (+2 years)	1 year	1 year

- Latest versions (each release)
  - Supported for 1 year
  - Automated upgrades and continuous updates

Enabled Services

- Block Storage API and extensions
- Compute API and Extensions
- Dashboard
- Elastic Map Reduce API
- Identity API and extensions
- Image Service API
- Networking API and extensions
- Orchestration API

Hypervisors

- KVM
- ESXi

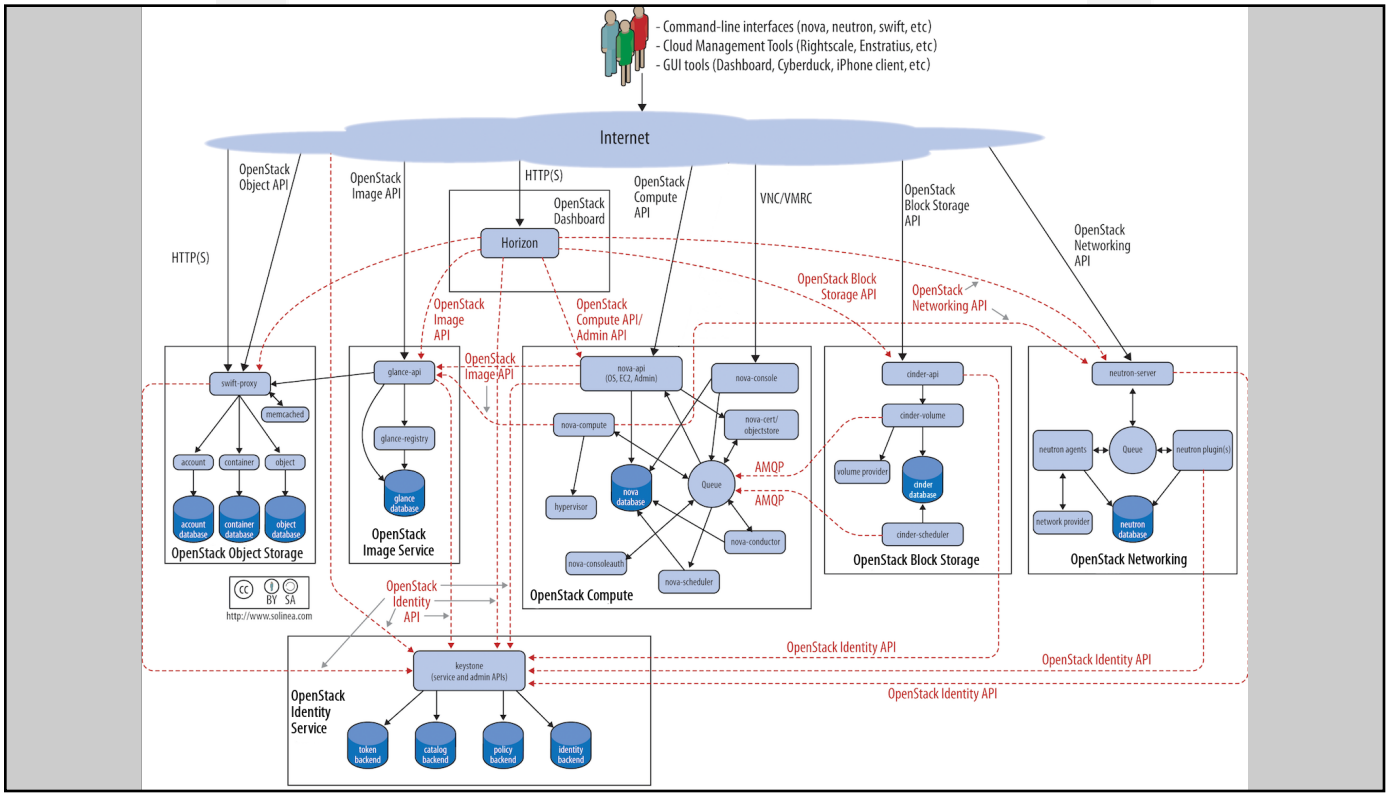
- HPE Helion OpenStack
- IBM Cloud Manager with OpenStack
- Mirantis OpenStack
- Oracle OpenStack for Oracle Linux (O3L)
- Oracle OpenStack for Oracle Solaris
- Red Hat OpenStack
- Stratoscale
- VMware Integrated OpenStack (VIO)
- Rackspace OpenStack Private Cloud software

Key Components of the Cloud

Basic Architecture

OpenStack logical architecture diagram - from openstack.org

- End users can interact through the dashboard, CLIs, and APIs.
- All services authenticate through a common identity service.



- Individual services interact with each other through public APIs, except where privileged administrator commands are necessary.

Advanced Message Queue Protocol

- OpenStack Compute uses a central hub for passing messages between daemons.
- The most commonly used Advanced Message Queueing Protocol (AMQP) is RabbitMQ, but ZeroMQ can also be used.

Databases

- Most build-time and runtime states for Cloud infrastructure are stored in a SQL database, including:
  - Available instance types
  - Instances in use
  - Available networks
  - Projects
  - Etc!
- The most commonly used databases are MySQL/MariaDB, PostgreSQL for production, and SQLite for development.

Keystone

What is Keystone?

Keystone is the Identity Service used by OpenStack for authentication and high-level authorization. It provides API client authentication, service discovery, and distributed multi-tenant authorization by implementing OpenStack's identity API.

- Acts as a common authentication system across the cloud.
- Currently supports token-based authorization and user-service authorization.
- The Identity Service (v3) generates authentication tokens that permit access to the OpenStack services REST APIs
- Provides authentication not only to users but also to OpenStack services
- Provides Identity, Token, Catalog and policy services using OpenStack Identity API v2 or v3
- Like most OpenStack projects, OpenStack Identity protects its APIs by defining policy rules based on an RBAC approach, stored in a .json policy file.
- Each Identity API v3 calls has a line in the policy file that dictates which level of governance of access applies.

- Keystone is a foundation service, and should be the first service installed

Keystone Concepts

User

A digital representation of a person, system, or service who uses OpenStack services and has associated information such as username, password and (optional) email.

Tenants/Projects

- A container used to group or isolate resources.
- Tenant must be specified to make requests to OpenStack services.
- May map to a customer, account, or organization.
- Users can be directly assigned to a particular tenant or project.

Role

A role includes a set of rights and privileges that specifies what operations a user is permitted to perform in a given tenant/project they are part of.

Token

- An arbitrary bit of text used to access resources.
- Each token has a scope, which describes the resources accessible with it.

Endpoint

- An endpoint is a network-accessible address, usually a URL, from where you can access an OpenStack service.
- URLs are available for all projects/tenants by default

Service

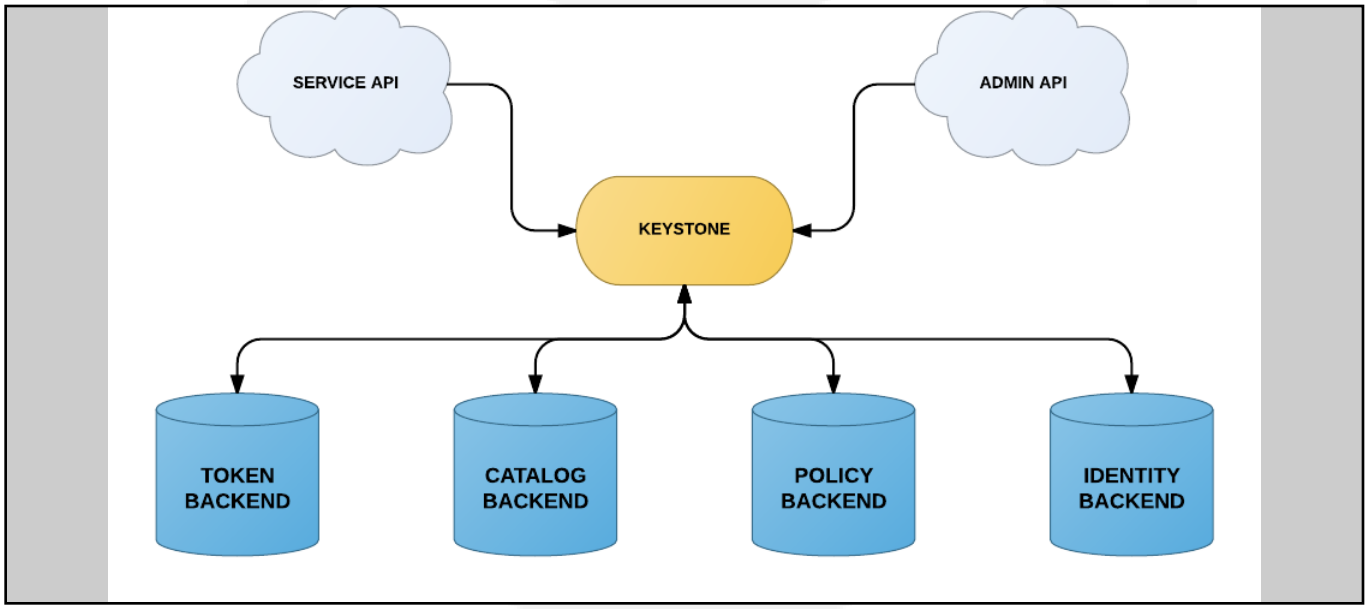
- Provides one or more endpoints through which users can access resources and perform operations
- Each service is identified by an ID number, name, service type, and description
- Services are associated with Public, Internal, and Admin endpoint URLs
    - **PublicURL** • Accessible over the public internet
    - **InternalURL** • Communication between services
    - **AdminURL** • Administrative management of OpenStack services
- The service catalog is a list of services and their respective endpoints

Keystone Authentication Process

- When an incoming functional call arises, Keystone confirms if it is from an authorized user via credentials (username, password, authURL)
- Once identity is confirmed, a token is provided. Tokens are a string of ACSII characters representing a successful authentication request
- The token includes a list of the user's projects/tenants and roles. Users can now submit tokens instead of re-authenticating on each request.
- Token expiration time, validity, and life span is customizable (default is 1 hour as of Icehouse release)

Keystone Architecture

- **Token Backend** • Contains temporary tokens
- **Catalog Backend** • Contains endpoint registry
- **Policy Backend** • Rule management interface and rule-based authorization
- **Identity Backend** • Contains users and groups. Deploys with its own database but can also be substituted with LDAP or other EAS



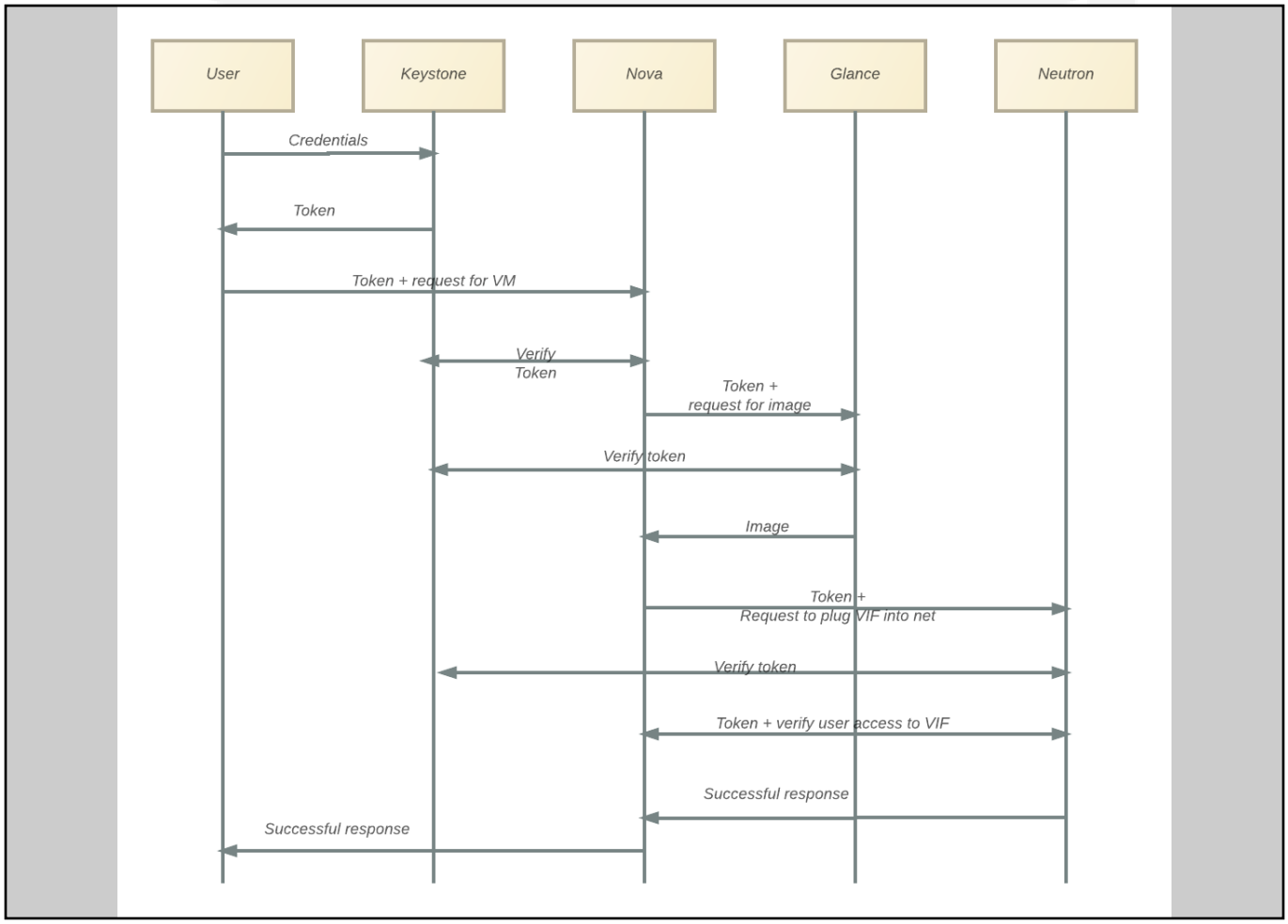
- **Assignments Backend** • Contains domains, projects (tenants), roles, and role assignments
- **Credentials Backend** • Contains credentials
- Uses Python-based library and daemon(s) to receive service and admin API requests
- Pluggable backend architecture helps Keystone to integrate with heterogeneous environments such as:
  - **Pluggable Authentication Module** • Uses local system's PAM service to authenticate

- **LDAP** • Connects via LDAP to AD to authenticate users and obtain role information
- **Key Value Store (KVS)** • In-memory
- **memcached** • Distributed memory caching system
- **SQL** • SQLite, MySQL, MariaDB

- Keystone tracks which OpenStack services are installed and where to locate them on the network

Workflow Example - Create A New VM

1. The user enters their credentials, which are verified by **Keystone**.
2. **Keystone** responds with an auth token.
3. The user sends a request to **Nova** to build a new instance, using the token provided by **Keystone**.
4. Nova verifies the token with **Keystone**.



5. Once the token is verified, Nova sends a request to **Glance**, requesting access to an image with the user's token.

6. **Glance** reaches over to **Keystone** to verify the token.
7. Once token is verified, **Glance** will serve an image over to **Nova** to create an instance with.
8. After instance has been created, **Nova** sends a request to **Neutron** containing the user token, requesting a VIF to plugin to the instance.
9. **Neutron** reaches over to **Keystone** to verify the auth token.
10. **Neutron** then reaches out to **Nova** to verify that the user has access to the specified VIF.
11. Upon verification, **Neutron** returns a Successful response to **Nova**.
12. The new instance was successfully created. **Nova** returns a Successful response to the user.

Horizon

Horizon is a Django-based project aimed at providing a complete OpenStack dashboard, along with an extensible framework for building new dashboards from reusable components.

- Horizon provides a web-based graphical user interface (GUI) to OpenStack services including Nova, Swift, Keystone, Cinder, Neutron, etc.
- Can be used to access, provision, and automate deployment of Cloud-based resources.
- Design allows for third-party products and services such as:
  - Billing
  - Monitoring
  - Additional management tools
- Can be customized and rebranded by service providers and commercial vendors using custom .css files

History

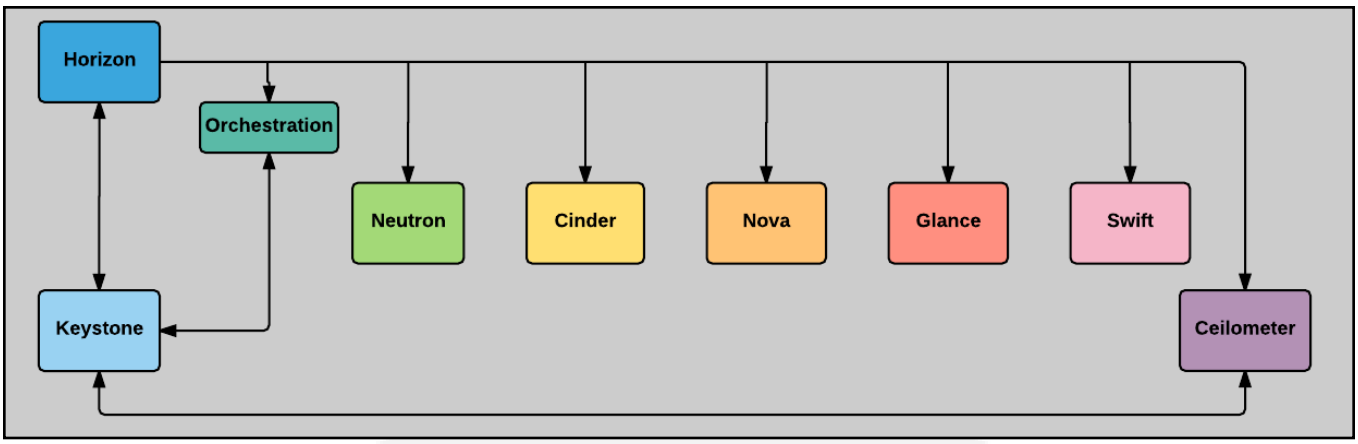
- Horizon began as a single application to manage the OpenStack Compute project.
- Horizon gradually expanded support to multiple OpenStack projects and APIs, rigidly arranged into “dash” and “syspanel” groupings.
- During the Diablo release cycle, an initial plugin system was added using signals to hook in additional URL patterns and add links into the “dash” and “syspanel” navigation.

Horizon Architecture

Horizon, at its core, should be a registration pattern for applications to hook into. What that means in relation to OpenStack values:

- **Core Support** • Out-of-the-box support for all core OpenStack projects.

- **Extensible** • Anyone can add a new component as a “first-class” citizen.



- **Manageable** • The core codebase should be simple and easy to navigate.
- **Consistent** • Visual and interaction paradigms are maintained.
- **Stable** • A reliable API with an emphasis on backwards-compatibility.
- **Usable** • Provides an awesome interface that people want to use.

Core Support

- Core Support is provided through three central dashboards. Between the three, they cover the core OpenStack applications and deliver core support.
  - User Dashboard
  - System Dashboard
  - Settings Dashboard

Extensible

- A Horizon dashboard application is based around the dashboard class that provides a consistent API and set of capabilities for both core OpenStack dashboard apps shipped with Horizon and equally for third-party applications.
- The dashboard class is treated as a top-level navigation item.
- Should a developer wish to provide functionality within an existing dashboard, the simple registration pattern makes it possible to write an app that hooks into other dashboards just as easily as creating a new dashboard by importing the dashboard you wish to modify.

Manageable

- There is a simple method for registering a panel (sub-navigation items) within the application.
- Each panel contains the necessary logic (views, forms, tests, etc) for that interface.

- This granular breakdown prevents files (such as api.py) from becoming thousands of lines long and makes code easy to find by correlating it directly to the navigation.

Consistent

Horizon maintains consistency by providing the necessary core classes to build from, as well as a solid set of reusable templates and additional tools such as base form classes, base widget classes, template tags, and class-based views.

Stable

- By architecting around these core classes and reusable components, Horizon developers have created an implicit contract that changes to these components will be made in the most backwards-compatible ways whenever possible.

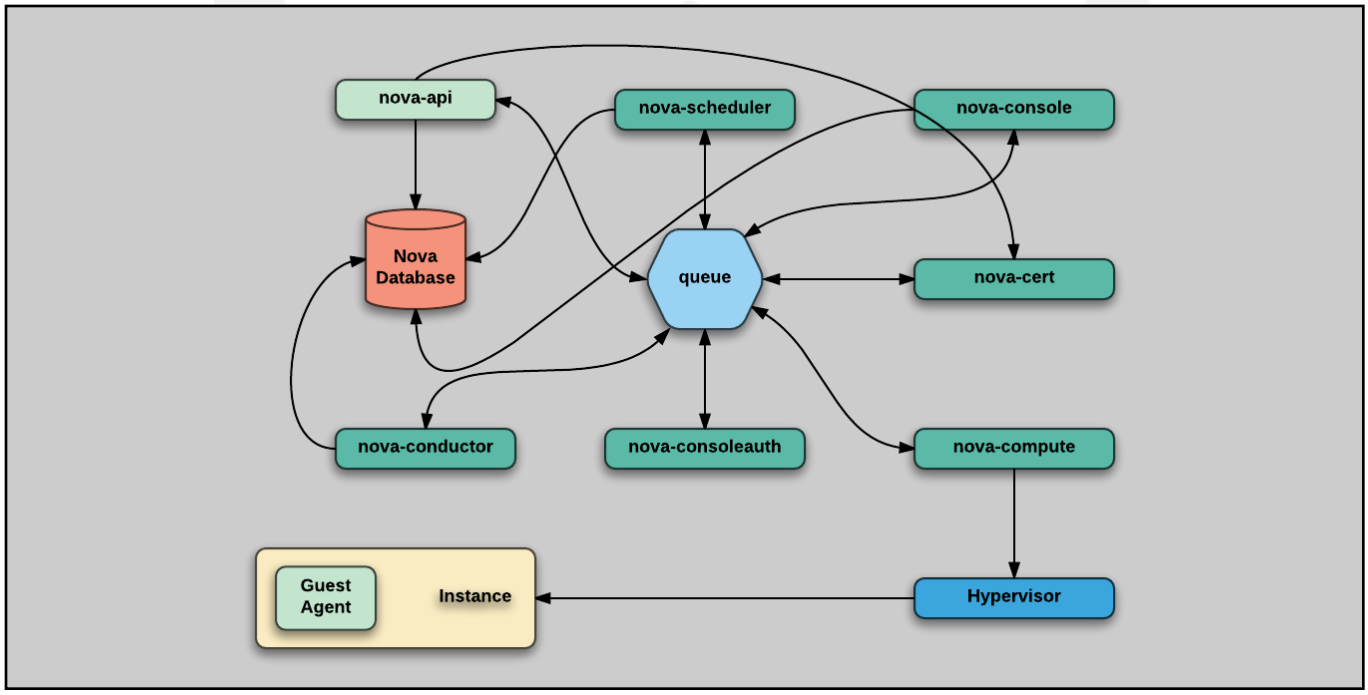
Usable

- This is ultimately left up to every developer that touches the code, but as all other goals are accomplished, they're freed to focus on the best possible experience.

Nova

OpenStack Compute (codename Nova) is an open source software designed to provision and manage large networks of virtual machines, creating a redundant and scalable cloud computing platform.

- Provides software, control panels, and APIs required to orchestrate a cloud.



- Running instances, managing networks, and controlling access through users and projects.

- Strives to be both hardware and hypervisor agnostic; currently supporting a variety of standard hardware configurations and seven major hypervisors.
- Interacts with Keystone (authentication), Glance (images), and Horizon (dashboard) for both admin and end users.
- When paired with Glance, Nova is an integral part of quick horizontal environmental scaling.

Detailed Feature List

- **Multi-tenancy** • An initial feature of OpenStack Compute, all facets of the platform are multi-tenant including billing, logging, auditing, and end-user control panels.
- **Massive scalability** • Compute architecture is designed to scale horizontally on standard hardware, with no proprietary hardware or software requirements. Compute nodes can be scaled into the thousands.
- **Multiple network models** • Supports a number of pluggable back-end networking drivers, such as:
  - **VLAN** • Configured instances live on a private network on a per-customer VLAN. Access to the private network and public NAT is provided through an OpenVPN gateway.
  - **FlatDHCP** • Public IP addresses are shared from a pool, and instance IP addresses are controlled via a DHCP server running on the host.
  - **Flat** • Public IP addresses are assigned from a pool of IP addresses. IPs can be “injected” into the client machine, or DHCP managed by an external DHCP infrastructure.
- **Pluggable Authentication** • A pluggable authentication system makes it possible to easily integrate an existing authentication system. Currently implemented backends:
  - **Local Auth** • standalone internal authentication system
  - **LDAP** • An example authentication module that integrates with an LDAP backend.
- **Block Storage Support** • Offers a variety of options as a supplementary volume:
  - **AOE** • ATA over Ethernet
  - **IET iSCSI** • Provisions IET iSCSI volumes
  - **LVM volume RBD** • RADOS block device, a network block device backed by objects in a Ceph distributed object store
  - **Sheepdog** • A distributed storage system for KVM using commodity hardware
  - **Solaris iSCSI** • iSCSI target running on Solaris/ZFS
  - **HP SAN** • HP StorageWorks P4000 SAN target
- **Control Panel** • A modern, AJAX based web control panel suitable for rebranding. Can be used as a customer-facing CP, or as a sample integration with existing control panels.

    - Django-based, and can be hosted on Apache or other highly scalable web servers.

- **Android and iOS Clients** • Includes reference Android and iOS clients. Applications can be used as-is, or rebranded to provide a provider-specific experience to customers.

- **Language Bindings** • Multiple language bindings are available for both legacy EC2 API and the OpenStack APIs.

    - Most Rackspace Cloud server bindings will work against an OpenStack Compute installation as well.

Nova Architecture

- `nova-api` • The nova-api daemon provides access to Nova services. Can communicate with other APIs, such as the Amazon EC2 API.

- `nova-scheduler` • Determines appropriate host for requests received from the queue.

- `nova-consoleauth` • Authorizes tokens for users that console proxies provide. Must be running for console proxies to work.

- `nova-cert` • A daemon that serves the nova cert service for x509 certificates. Used to generate certs for `euca-bundle-image`. This module is only needed for EC2 API.

- `nova-conductor` • Mediates interactions between `nova-compute` and the database, eliminating direct access to the database made by `nova-compute`.

- `nova-consoleauth` • A daemon that authorizes tokens for users that console proxies provide. Must be running for console proxies to work.

- `nova-compute` • A worker daemon that manages instance lifecycles through API.

- `nova-network` • Similar to `nova-compute`, `nova-network` accepts networking tasks from the queue and manipulates the network by performing tasks such as bridging interfaces or changing iptables rules.

- `nova-*-proxy` • Browser-based proxy for access to running instances.

VNC Proxy types

- `nova-novncproxy` • Provides access through a VNC connection. Supports browser-based novnc clients.

- `nova-spicehtml5proxy` • Provides access through SPICE. Supports browser-based HTML5 clients.

- `nova-xvpvncproxy` • Provides access through a VNC connection. Supports an OpenStack-specific Java client.

Hypervisor Support

Nova can use many different hypervisors as a back-end virtualization target for an OpenStack Compute

cluster.

- Xen/XenServer
- KVM
- Microsoft Hyper-V
- VMWare/ESX
- LXC (Linux Containers)
- Qemu
- UML

Popular Uses

- Service providers offering an IaaS Compute platform
- IT departments provisioning Compute resources to teams and projects
- Processing big data with tools like Hadoop
- Scaling Compute up and down to meet demands for web resources and applications
- OpenStack is a full open source project under an OSI-approved license. It is free to all for modification or enhancement without the worry of viral licensing.

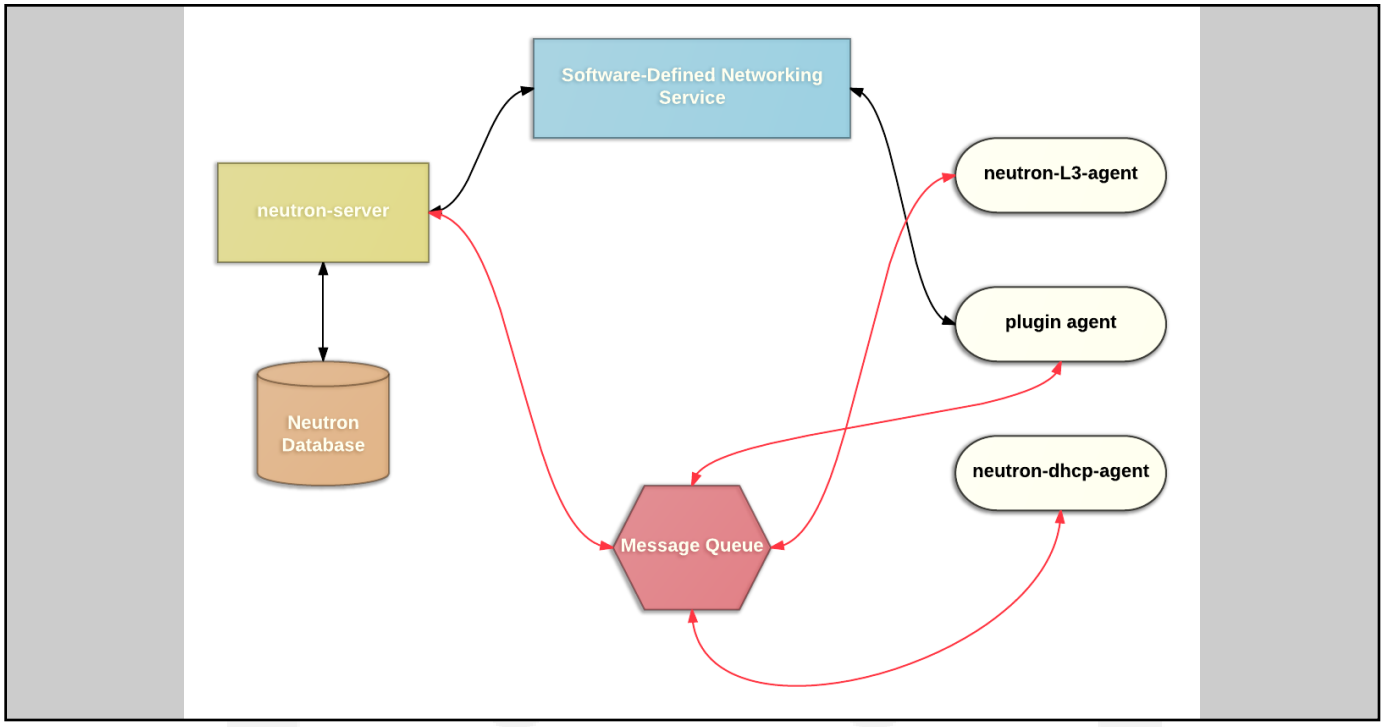
Neutron

Neutron is an OpenStack project created to provide Networking as a Service (NaaS) between interface devices managed by other OpenStack services.

- Responsible for setting up virtual network infrastructure in an OpenStack environment.
- Provides flexibility through plugins, drivers and agents.
- Core components provide switching and routing for virtual machines.
- Specialized virtual network functions:
  - VPNaaS
  - FWaaS
  - LBaaS
- Formerly known as Quantum
  - Name was changed to Neutron due to copyright issues

Networking Components

- Neutron Server (`neutron-server` and `neutron-*-plugin`)
  - Runs on the network node to service the Networking API and its extensions.
  - Enforces the network model and IP addressing of each port.
  - Requires indirect access to a persistent database, accomplished through plugins which



communicate with the database using AMQP (Advanced Message Queueing Protocol).

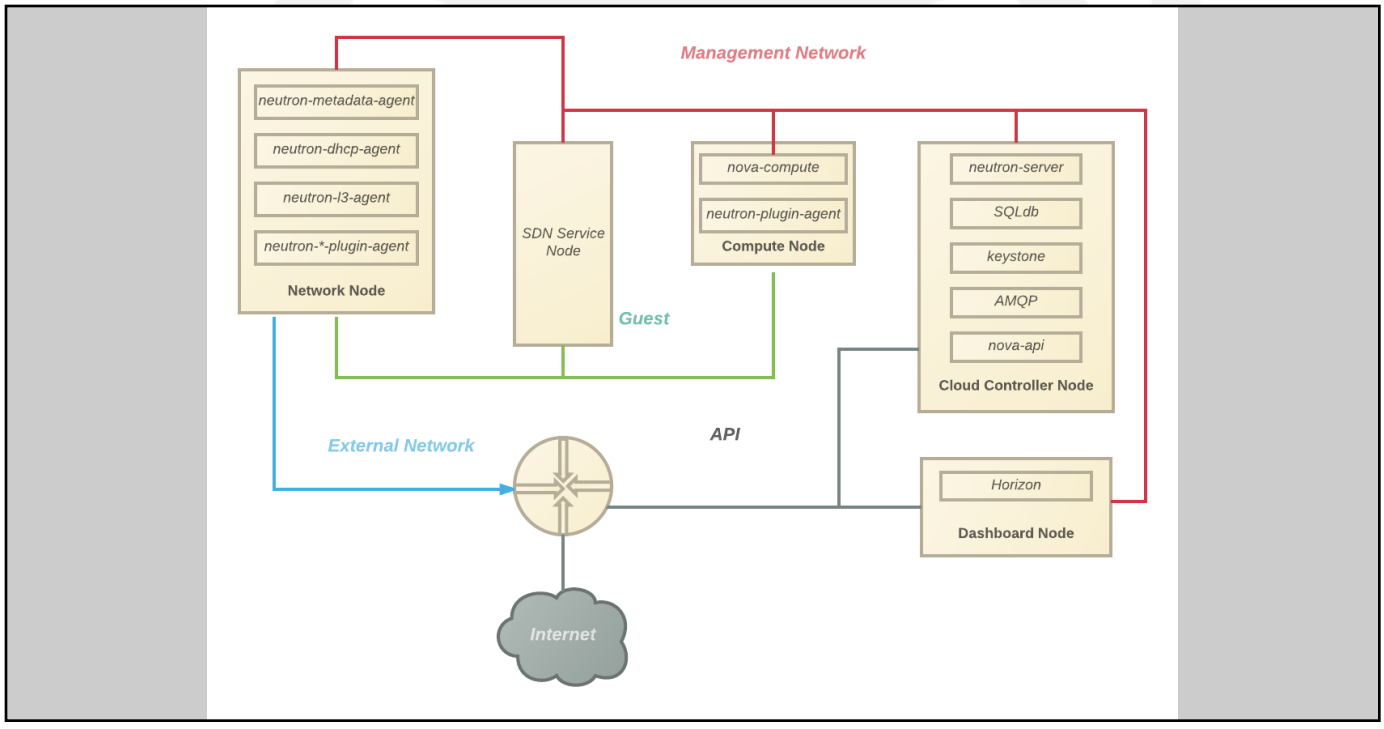
- Plugin Agent (`neutron-*-agent`)
  - Runs on each compute node to manage local virtual switch (vSwitch) configuration.
  - Agents are determined by which plugin is used in your environment.
  - Requires message queue access and depends on the plugin used.
  - Some plugins, like OpenDaylight (ODL) and Open Virtual Network (OVN) do not require any Python agents on compute nodes.
- DHCP Agent (`neutron-dhcp-agent`)
  - Optional, depending on plug-in
  - Provides DHCP services to tenant networks.
  - The DHCP agent is the same across all plug-ins and is responsible for maintaining DHCP configuration.
  - Requires message queue access.

- L3 Agent (`neutron-l3-agent`)
  - Optional, depending on plugin
  - Provides L3/NAT forwarding for external network access of VMs on tenant networks.
  - Requires message queue access
- Network Provider Services (SDN server/services)
  - Provides additional networking services to tenant networks.
  - May interact with neutron-server, neutron-plugin, and plugin-agents through communication channels, such as REST APIs.

Network Types

A standard OpenStack networking environment has up to four distinct physical data center networks:

- **Management Network** • For internal communication between OpenStack components. The IP address on this network should be reachable only from within the datacenter and is considered the Management Security Domain.



- **Guest Network** • Used for VM communication within the cloud deployment. IP addressing requirements depend on the OpenStack Networking plugin uses and the network configuration choices of the virtual networks made by the tenant. Considered the Guest Security Domain.

- **External Network** • Used to provide eVMs with Internet access in some deployment scenarios. The IP addresses on this network should be reachable by anyone on the Internet. Considered Public Security Domain.

- **API network** • Exposes all OpenStack APIs, including the OpenStack Networking API, to tenants. IP addresses should be reachable by anyone on the Internet. It may be on the same network as the external network, as it is possible to create a subnet for the external network that uses API allocation ranges to use less than the full range of IP addresses in an IP block. Considered the Public Security Domain.

Glance

OpenStack Compute (Nova) relies on an external image service to store virtual machine images and maintain a catalog of available images. By default, Compute is configured to use the Glance image service, which is currently (as of Newton release) the only supported image service.

- The Glance project provides a service that allows users to upload and discover data assets, including images and metadata definitions, meant to be used with other services.
- Glance image services include discovering, registering, and retrieving virtual machine images.
- Has a RESTful API that allows querying of VM image metadata as well as retrieval of the actual image.
- Virtual images made available by the image service can be stored in a variety of locations, from simple file systems to object-storage systems like OpenStack Object Storage (Swift).
- The Images API v1 has been DEPRECATED in the Newton release in favor of API v2. The Images API v1 will ultimately be removed following the standard OpenStack deprecation policy.

Design Guidelines

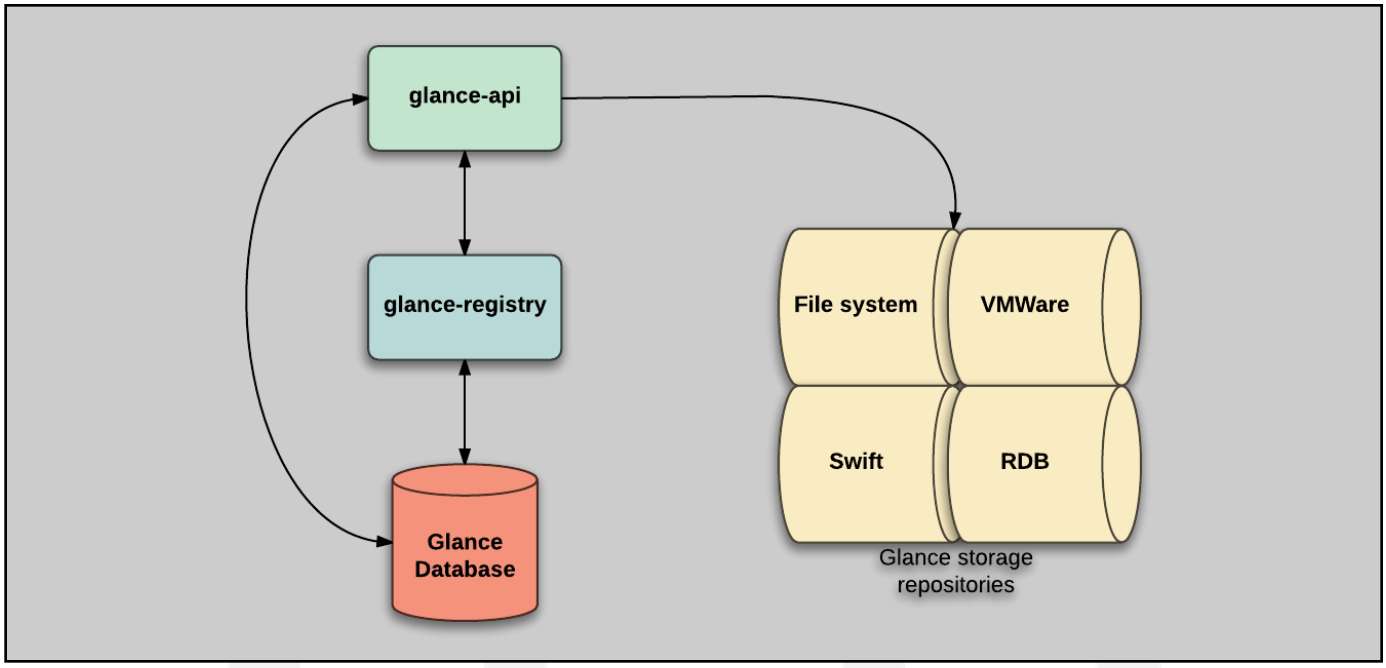
- **Component-based architecture** • Quickly add new behaviors
- **Highly available** • Scale to very serious workloads
- **Fault tolerant** • Isolated processes avoid cascading failures
- **Recoverable** • Failures should be easy to diagnose, debug, and rectify
- **Open standards** • A reference implementation for a community-driven API

Glance API

- Glance has two APIs:
  - **User-facing API** • Accessible over the public Internet
  - **Registry API** • For internal requests; requires access to the database.
  - Both APIs have two major versions, v1 (supported) and v2 (current).
  - You can run either or both versions by enabling the correct value in your glance_api configuration file.

Glance Overview

- The OpenStack image service is central to Infrastructure as a Service (IaaS)
- It accepts API requests for disk or server images, and metadata definitions from end users or OpenStack Compute components.
- Supports the storage of disk or server images on various repository types, including OpenStack



Object Storage (Swift)

Glance Architecture

The OpenStack image service includes the following components:

- **glance-api** • Accepts image API calls for image discovery, retrieval and storage
- **glance-registry** • Stores, processes, and retrieves metadata about images. Metadata includes items such as size and type
  - The registry is a private internal service meant for use by OpenStack image service and should not be exposed to users
- **Database** • The database (can be MySQL or SQLite, depending on preference) stores image metadata
- **Storage repository for image files** • Various repository types are supported, including:
  - Normal file systems (or any filesystem mounted on the glance-api controller node)
  - Object Storage (Swift)

- RADOS block devices
- VMWare datastore
- HTTP

- **Metadata definition service** • A common API for vendors, administrators, services, and users to meaningfully define their own custom metadata. This metadata can be used on different types of resources, such as:
  - Images
  - Artifacts
  - Volumes
  - Flavors
  - Aggregates
  - A definition includes the new property's key, description, constraints, and the resource types it can be associated with.

Swift

The OpenStack Object Store project known as Swift is a highly available, distributed, eventually consistent object store that can be used to store lots of data efficiently, safely, and cheaply.

- Objects and files are written to multiple disk drives spread over multiple servers in a datacenter, with OpenStack software responsible for ensuring data replication and integrity across the cluster.
- Clusters can be scaled horizontally simply by adding additional servers

Swift Architecture

The account, container, and object hierarchy affects how you interact with the Object Storage API

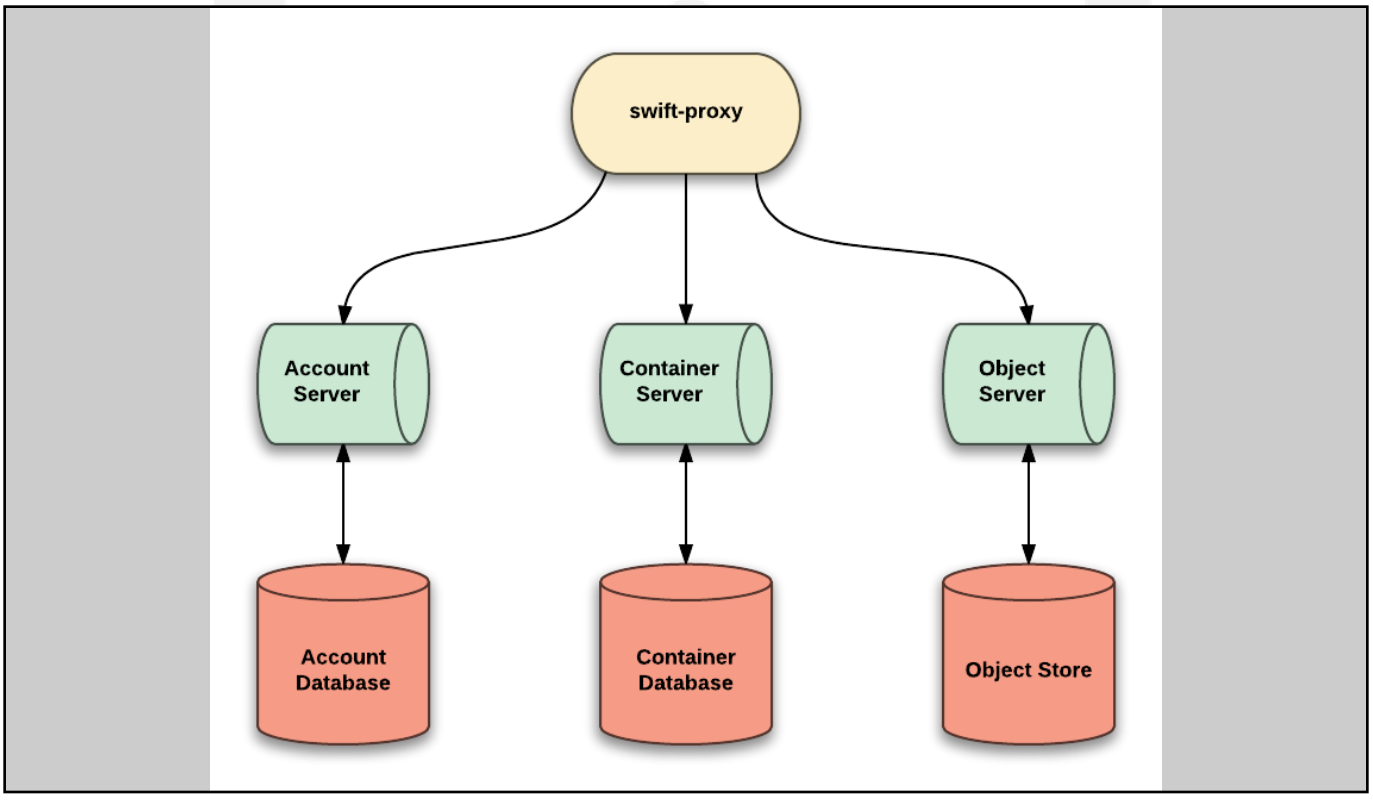
- **Account** • Represents the top-level of the hierarchy. Your account (created by the service provider) defines a name space for containers. In the OpenStack environment, account is synonymous with a project/tenant.
- **Container** • Defines a name space for objects. An object with the same name in two different containers represents two different objects. Any number of containers can be created within an account.
- **Object** • Stores data content such as documents, images, etc. Objects can also store custom metadata. The object storage API can:
  - Store an unlimited number of objects as large as 5GB each (default). Maximum object size can be adjusted.
  - Upload and store objects of any size with large object creation

- Use cross-origin resource sharing to manage object security
- Compress files using content-encoding metadata
- Override browser behavior for an object using content-disposition metadata
- Schedule objects for deletion
- Bulk-delete up to 10,000 objects in a single request
- Auto-extract archive files
- Generate a URL that provides time-limited GET access to an object
- Upload objects directly to the Object Storage system from a browser by using the form POST middleware

- Specifically, the resource path reflects a structure with this format:
  - `/v1/[ACCOUNT]/[CONTAINER]/[OBJECT]`
  - For example, if a user with account #555555 wanted to access file.jpg in a container named *Example*, the path would be:
    - `/v1/555555/Example/file.jpg`

Cinder

- OpenStack Block Storage (codename Cinder) is designed to present storage resources to end users



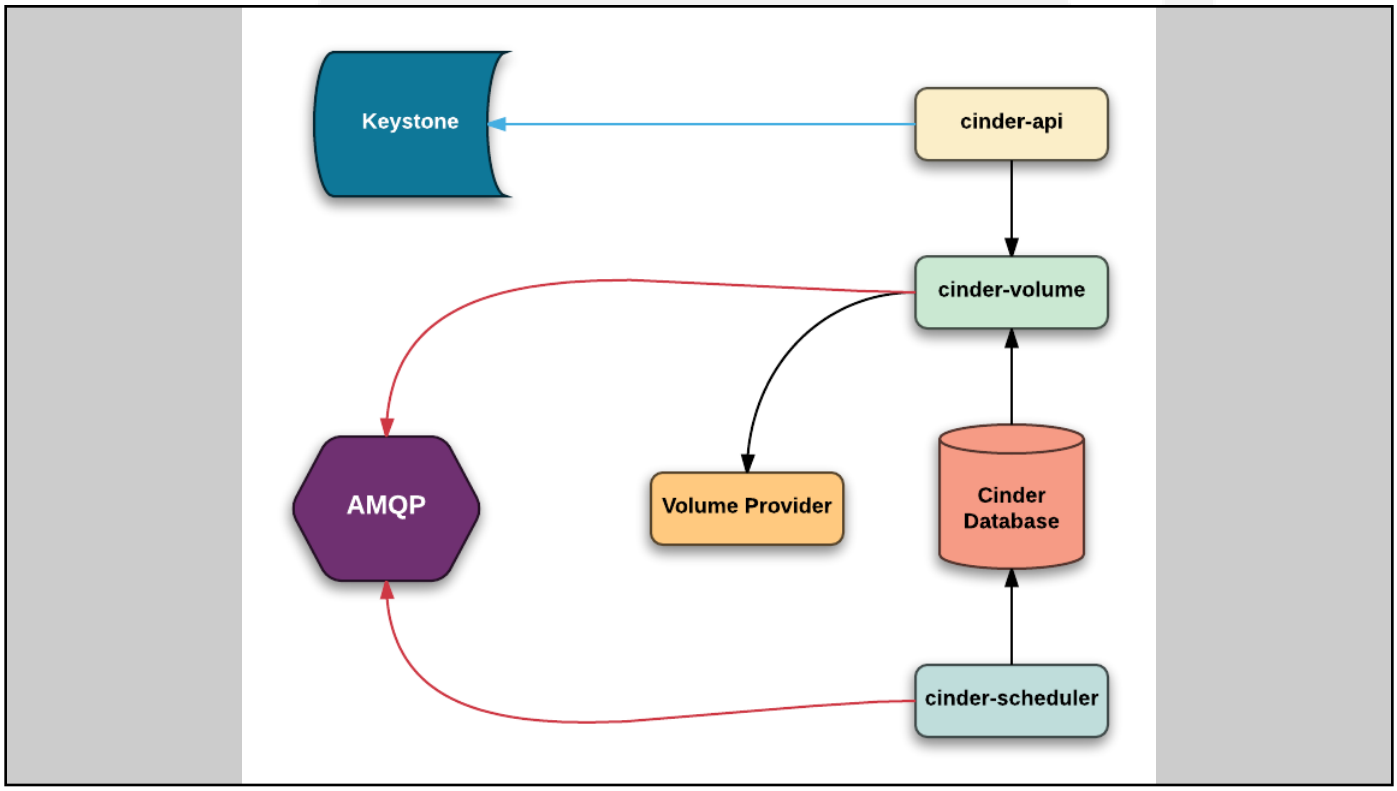
that can be consumed by the OpenStack Compute project (Nova).

- This is done through the use of either a reference implementation (LVM) or plugin drivers for other storage.
- Cinder manages the creation, attachment and detachment of block storage devices to servers.
- Fully integrated with OpenStack compute and Horizon, Cinder allows users to quickly and easily manage their own storage needs.
- Cinder block storage service is intended to run on one or more nodes
- In short, Cinder virtualizes the management of block storage devices and provides end users with a self-service API to request and consume said resources without requiring any knowledge of where their storage is actually deployed or on what type of device.

Cinder Architecture

Cinder is composed of three main services:

- `cinder-api` • Component that receives HTTP requests, converts commands and communicates



with cinder-scheduler (creation) and cinder-volume (management) via AMQP or HTTP

- `cinder-scheduler` • In charge of host placement for newly created volumes. Forwards request to `cinder-volume`.
- `cinder-volume` • Manages dynamically attachable block devices. Receives volume management requests from `cinder-api` and `cinder-scheduler`, and rotates them to storage

backends using vendor-supplied drivers.

- **SQL Database** • Central database that provides shared data storage for all components

Cinder Backend Support

- A Cinder backend is the configuration object that represents a single provider of block storage upon which provisioning requests may be fulfilled. A Cinder backend communicates with the storage system through a Cinder driver.
- The default OpenStack Block Storage service implementation is an iSCSI solution that uses LVM for Linux.
- Until the Grizzly release, a separate instance of cinder-volume service was run for each pool managed by Cinder.
- When you configured multiple-storage backends, several backend storage solutions that serve the same OpenStack Compute configuration and one cinder-volume is launched for each backend storage or backend storage pool
- **Multi-backend support** • Added in the Grizzly release, allows a single instance of cinder-volume to manage all storage pools through a single configuration file.
- This has become the preferred method to run Cinder, as it concentrates configuration into one file and uses less resources.

Cinder Use Cases

- Easy migration and storage of non-disposable data
- Quick recovery
- Horizontally scalable
- Can be used as boot disk

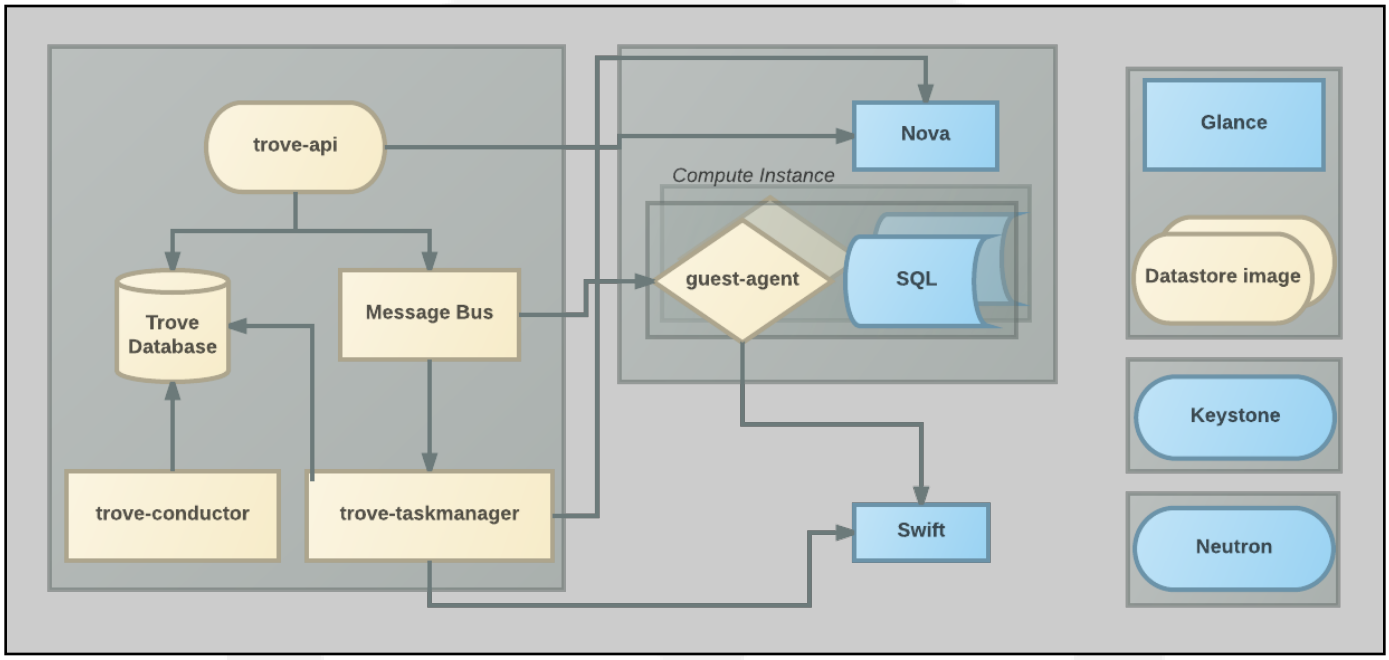
Optional Services to Expand Your Cloud

Trove (formerly known as RedDwarf)

- DBaaS for OpenStack, designed to run entirely on OpenStack.
- Allows users to quickly and easily utilize features for both relational and non-relational database engines without the burden of handling complex administrative tasks.
- Allows cloud users and administrators to provision and manage multiple database instances as needed.
- Focused on providing resource isolation at high performance while automating complex administrative tasks including:

- Deployment
- Configuration
- Patching
- Backups
- Restores
- Monitoring

Trove Components



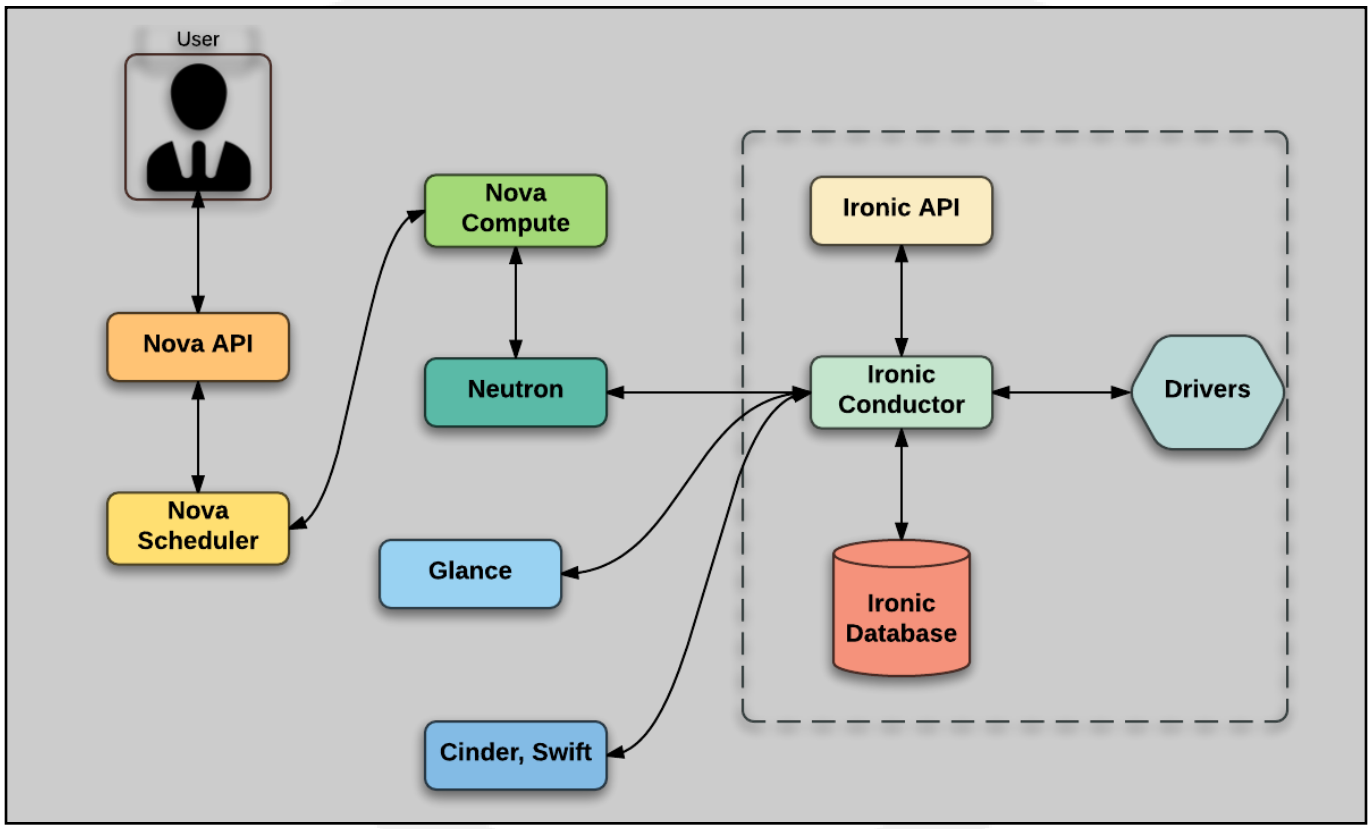
Trove is designed to support a single-tenant database within a Nova instance. Since Trove interacts with other OpenStack components purely through API, there will be no restrictions on how Nova is configured.

- `trove-api` • Provides a RESTful API that supports JSON and XML to provision and manage Trove instances
  - A RESTful component
  - Uses a WSGI launcher configured by Trove
  - The API class wires the REST paths to the appropriate controllers
- `trove-taskmanager` • Manages lifecycle of instances and performs operations on the database instance.
  - Listens on a RabbitMQ topic
  - Requests for this component are pushed to MQ from another component using the `trove-`

`taskmanager` API module and putting the method's name as a parameter

- `trove-guestagent` • A service that runs within the guest instance, responsible for managing and performing operations on the database itself. The guest agent listens for RPC messages through the message bus and performs the requested operation.
    - The guest agent is similar to `trove-taskmanager` in the sense that it runs as a service that listens on a RabbitMQ topic
    - Guest agent runs on every DB instance and a dedicated MQ topic is used, identified with the instance's ID

Ironic



OpenStack bare metal provisioning (codename Ironic) is an integrated OpenStack program which aims to provision bare metal instances instead of virtual machines, forked from a bare metal driver.

- It is best thought of as a bare metal hypervisor API and set of plugins which interact with the bare metal hypervisors.
- By default, bare metal uses PXE (Preboot Execution Environment) and IPMI (Intelligent Platform Management Interface) to provision and turn on/off machines but will also support vendor-specific plugins which may implement additional functionality

Bare Metal Components

- A RESTful API service, through which operators and other services may interact with managed bare metal servers
- A Conductor service, which does the bulk of the work. Functionality is exposed through the API. The Conductor and API services communicate via RPC.
- Various drivers that support heterogeneous hardware
- A message queue (RabbitMQ, ZeroMQ, etc)
- A database for storing resource information details such as state of the conductors, nodes, and drivers.

Bare Metal Deployment Process

- A user request to boot an instance is passed to the Nova Compute service via Nova API and Nova Scheduler.
- The Compute service hands over this request to the Ironic service, where the request passes from the Ironic API, to the Conductor, to a Driver to successfully provision a physical server for the user.
- Just as the Nova Compute service talks to various OpenStack services like Glance, Neutron, Swift, etc. to provision a virtual machine instance, here the Ironic service talks to the same OpenStack services for image, network and other resource needs to provision a bare metal instance.

Heat

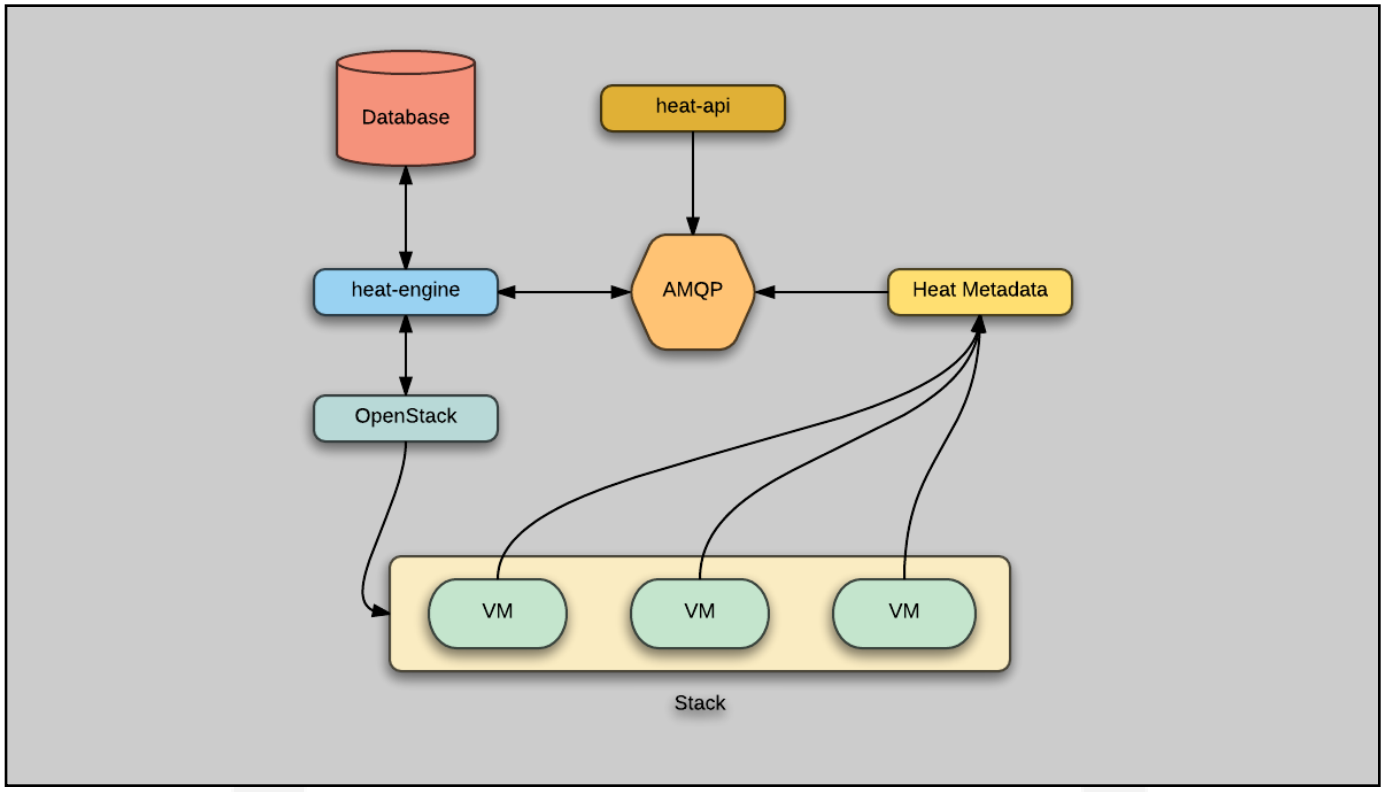
Heat is the main project in the OpenStack Orchestration program. It implements an orchestration engine to launch multiple composite cloud applications based on templates in the form of text files that can be treated like code. Heat provides both an OpenStack Native REST API and a CloudFormation compatibule Query API.

Heat Deployment Process

- A Heat template describes the infrastructure for a cloud application in a human +R-W text file that can be checked into version control, diffed, etc.
- Infrastructure resources that can be described include: servers, floating IPs, volumes, security groups, users, etc.
- Also provides an autoscaling service that integrates with Telemetry
- Templates can specify the relationships between resources. This enables Heat to call out the OpenStack APIs to create your infrastructure in the correct order.
- Manages the whole lifecycle of an application. Modify templates to update or change existing infrastructure.
- Heat primarily manages infrastructure, but the templates integrate well with software configuration management tools such as Puppet and Chef.

Heat Architecture

- `heat` • The heat tool is a CLI which communicates with heat-api to execute AWS



CloudFormation APIs.

- `heat-api` • Provides an OpenStack-native RESTful API that processes API requests by sending them to the heat-engine over RPC.

- `heat-api-cfn` • Provides an AWS-style Query API that is compatible with AWS CloudFormation and processes API requests by sending them to the heat-engine over RPC.

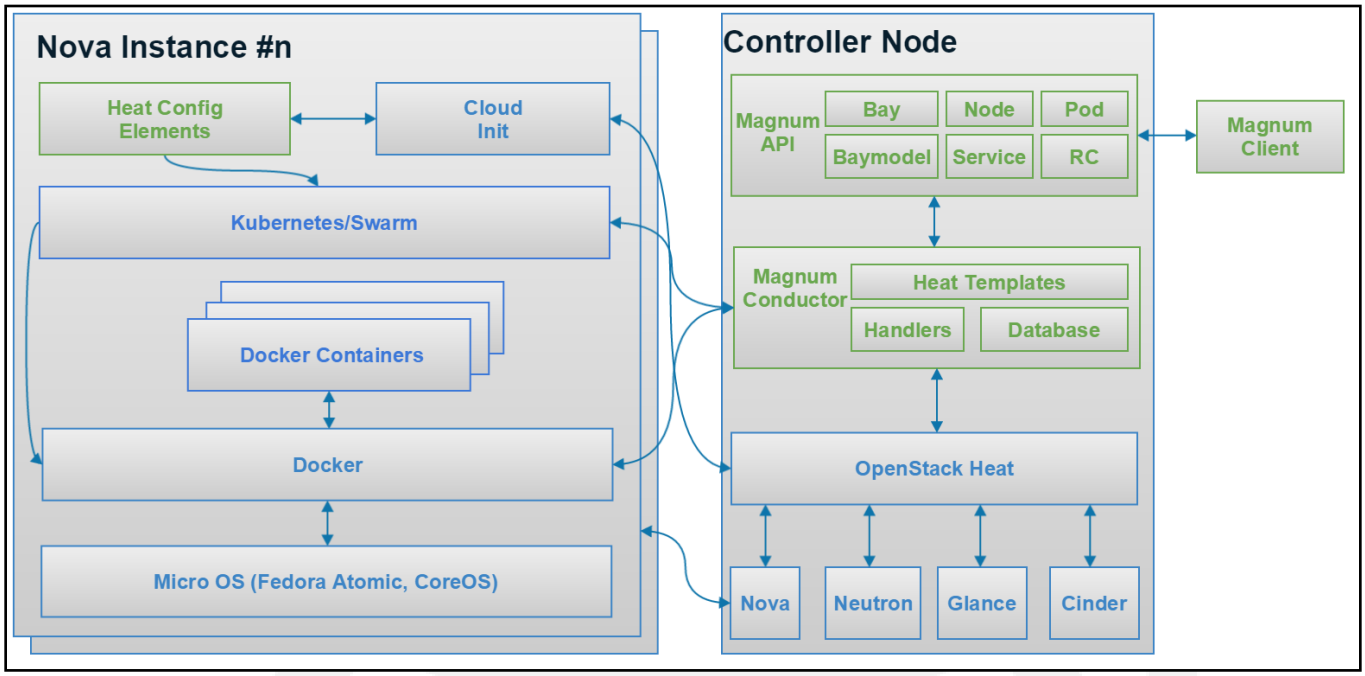
- `heat-engine` • Does the main work of orchestrating the launch of templates and providing events back to the API consumer.

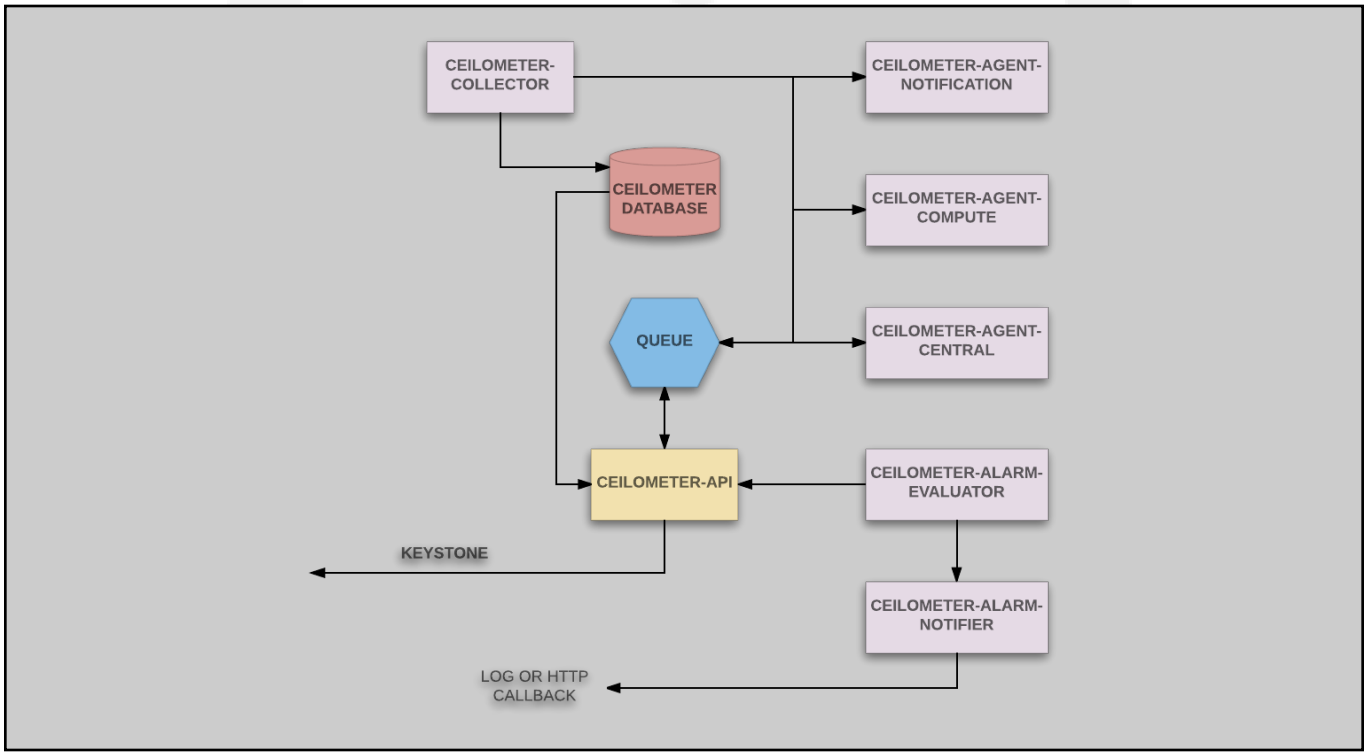
Magnum

OpenStack Magnum is a multi-tenant Containers as a Service combining the best of infrastructure software with the best of container software.

- Created to make current container technology and tools work with OpenStack.
- Allows cloud operators to offer CaaS as a hosting service.
- A collaboration between 101 Engineers from 28 different companies
- First released three months into Liberty release of OpenStack
- Primarily focused on container infrastructure, while other services are used for container

management.





OpenStack Telemetry

OpenStack Telemetry (Ceilometer) provides a single point of contact for billing systems, providing counters

to establish customer billing across all current and future OpenStack components.

- Use cases:
  - Metering
  - Monitoring
  - Alarming, etc

Telemetry Services

The Telemetry project provides a set of functionality split across multiple projects, with each project designed to provide a discrete service in the Telemetry space.

- **Aodh** • An alarm service. Enables the ability to trigger actions based on defined rules against sample or event data collected by Ceilometer
- **Ceilometer** • A data collection service that efficiently collects, normalizes, and transforms data produced by OpenStack services. The data collected is intended to be used to create different views and help solve various telemetry use cases. Aodh and Gnocchi are two examples of services extending Ceilometer data.
- **Gnocchi** • A time-series database and resource indexing service which enables users to capture OpenStack resources and the metrics associated with them. Using rolling aggregation set by user-defined archival policies, its aim is to provide a scalable means of storing both short and long-term data and provide a statistical view based on the input data (i.e. Ceilometer)
- **Panko** • An event and metadata indexing service which enables users to capture the state information of OpenStack resources at a given time. Panko's aim is to enable a scalable means of storing both short and long-term data for use cases such as auditing and system debugging.

Other Services

- Sahara (formerly known as Savanna)
  - Elastic Map Reduce, a component to easily and rapidly provision Hadoop clusters.
  - Also provides means to scale a pre-existing Hadoop cluster by adding and removing worker nodes on demand.
- Designate (formerly known as Moniker)
  - A multi-tenant REST API for DNS management.
  - Provides DNSaaS and is compatible with many backend technologies, including PowerDNS and BIND.
  - Was created to interface with existing DNS servers to manage DNS zones on a per-tenant basis and does not provide a DNS service.

DIYCloud

VirtualBox

- Compatible with Windows, Linux, or OSX hosts
- Simple click-through installation
- Network device needs to be bridged to your network
    - Bridged mode will get an IP on the primary network, which will allow access to the OpenStack env from any other node on that same network. If NAT mode is used, access from outside of the network will not be allowed.
- Promiscuous mode enabled

Note for Windows Users

For first-time users and Windows users, there may not be a 64-bit option to install; usually if you don't have virtualization enabled in BIOS. To resolve this, reboot your server into the BIOS menu and enable `Virtualization or VTX technology`. If you're running Windows 10, there may be conflicts with running other hypervisors, including VirtualBox if the hypervisor for Hyper-V is enabled. You will need to disable Hyper-V to resolve the conflicts.

- Download the latest version of VirtualBox here: https://www.virtualbox.org/wiki/Downloads

DevStack on Ubuntu

DevStack is a series of extensible scripts used to quickly bring up a complete OpenStack environment based on the latest versions of everything on the master Git branch. It is used interactively as a development environment and as the bases for much of the OpenStack project's functional testing.

DevStack attempts to support Ubuntu 16.04/17.04, Fedora 24/25, CentOS/RHEL 7, Debian, and OpenSUSE.

- All-in-one single machine setup
- Network device needs to be bridged to your network
- Network device needs Promiscuous mode allowed
- May use Windows, Linux, or OSX host
- Download Ubuntu 16.04 Minimal Server image

  - Minimum requirements:
    - 6144 MB RAM
    - 20GB virtual disk
- To download: https://www.ubuntu.com/download/server
- On install screen, choose **Ubuntu 16.04 minimal install**
- Software to install:
  - Standard system utilities
  - OpenSSH server
- Install GRUB
- Reboot system
- After restart completes, log in to server with the username created during installation:

```
ssh $USER@$IP
```

- Update system:

```
apt-get update && apt-get -y upgrade
```

- Print network configuration settings:

```
ifconfig
```

- Configure static networking.
  - Edit `/etc/network/interfaces`.
  - **NOTE:** Depending on your local settings, your primary public network may have a name other than `eth0`. Please adjust the following as needed to work in your environment.

```
iface eth0 inet static
address $IPADDRESS
netmask 255.255.255.0
gateway $GW-IP
dns-nameservers  $GW-IP 8.8.8.8
```

- Restart network to set changes:

```
ifdown eth0
```

```
ifup eth0
```

- Verify networking functionality:

```
ping google.com
```

- Add stack user for interaction with DevStack:

```
adduser stack
Adding user `stack' ...
Adding new group `stack' (1001) ...
Adding new user `stack' (1001) with group `stack' ...
Creating home directory `/home/stack' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for stack
Enter the new value, or press ENTER for the default
Full Name []:
Room Number []:
Work Phone []:
Home Phone []:
Other []:
Is the information correct? [Y/n] y
```

- Grant passwordless sudo privileges to `stack` user:

```
echo “stack ALL=(ALL)       NOPASSWD: ALL” >> /etc/sudoers
```

- Switch to `stack` user:

```
su - stack
```

- Install Git:

```
sudo apt-get install git -y
```

- Pull DevStack Ocata repo from Git.
  - Note: you can pull the latest version of OpenStack by omitting the `-b stable/ocata` in the clone call.

```
git clone https://git.openstack.org/openstack-dev/devstack -b
stable/ocata
```

- Move into `devstack` folder:

```
cd devstack
```

- Edit `local.conf`:

```
[[local|localrc]]
ADMIN_PASSWORD=openstack
DATABASE_PASSWORD=openstack
RABBIT_PASSWORD=openstack
SERVICE_PASSWORD=openstack
```

- Launch DevStack installation script:
  - Note: DevStack installation can take upwards of 45 minutes depending on system resources.

```
./stack.sh
```

- Upon completion, you should have a message similar to this in your terminal:

```
This is your host IP address: 111.222.333.4
This is your host IPv6 address: 9999:8888:7777:6666:5555:4444:3333
Horizon is now available at http://111.222.333.4/dashboard
Keystone is serving at http://111.222.333.4/identity/
The default users are: admin and demo
The password: openstack
```

Packstack on CentOS

Packstack is a utility that uses Puppet modules to deploy various OpenStack services over SSH automatically on Red Hat and CentOS operating systems. Packstack can be used to install OpenStack either in single-node or multi-node environments.

- All-in-one single machine setup
- CentOS 7 minimal server installation in VirtualBox
  - Minimum requirements:
    - 6144 MB RAM
    - 20GB virtual disk
- Network device needs to be bridged to your network
  - Bridged mode will get an IP on the primary network, which will allow access to the OpenStack environment from any other node on that same network. If NAT mode is used, access from outside of the network will not be allowed.
  - Promiscuous mode set to Allow All

  - Windows, Linux, or OSX host

- Download latest CentOS 7 Minimal Server image: http://isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7-x86_64-Minimal-1611.iso

- After creating virtual machine in VirtualBox, log in to instance as *root*:

```
ssh root@$USER
```

- Update system:

```
yum -y update
```

- Install `qemu-ev` dependency:

```
yum -y install centos-release-qemu-ev
```

- Uninstall `mariadb-libs` to avoid MariaDB version conflicts:

```
yum -y remove mariadb-libs
```

- Grant passwordless sudo privileges to your admin user:

```
visudo
```

- Uncomment the following line, then save and exit the file:

```
%wheel ALL=(ALL)       NOPASSWD: ALL
```

- Disable SELinux:

```
sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/
config
```

- Disable NetworkManager and firewalld:

```
systemctl disable NetworkManager firewalld
```

- Locate and install OpenStack Ocata repository:

```
yum search openstack
sudo yum -y install centos-release-openstack-ocata
```

- Install OpenStack Packstack:

```
sudo yum -y install openstack-packstack
```

- Generate answer file:

```
packstack --gen-answer-file=/home/$USER/answers.txt
```

- Edit the following in `answers.txt`:

```
CONFIG_DEFAULT_PASSWORD=openstack
CONFIG_HEAT_INSTALL=y
CONFIG_MAGNUM_INSTALL=y
CONFIG_KEYSTONE_ADMIN_PW=openstack
CONFIG_KEYSTONE_DEMO_PW=openstack
```

- Start PackStack installation
  - Note: Installation can run anywhere from 10-45 minutes depending on system resources.

```
packstack --answer-file=/home/$USER/answers.txt
```

- Upon build completion, Packstack will return a message similar to this:

```
****Installation completed successfully****
Additional information:
 * Time synchronization installation was skipped. Please note that
unsynchronized time on server instances might be problem for some
OpenStack components.
 * File /root/keystonerc_admin has been created on OpenStack
client host 123.456.789.0. To use the command line tools you need
to source the file.
 * To access the OpenStack Dashboard browse to
http://123.456.789.0/dashboard .
Please, find your login credentials stored in the keystonerc_admin
in your home directory.
 * Because of the kernel update the host 123.456.789.0 requires
reboot.
 * The installation log file is available at: /var/tmp/
packstack/20170904-112058-Is6ox9/openstack-setup.log
 * The generated manifests are available at: /var/tmp/
packstack/20170904-112058-Is6ox9/manifests
```