
MIRANTIS
OpenStack Overview
What is OpenStack?
training.mirantis.com
Copyright © 2017 Mirantis, Inc. All rights reserved

Agenda

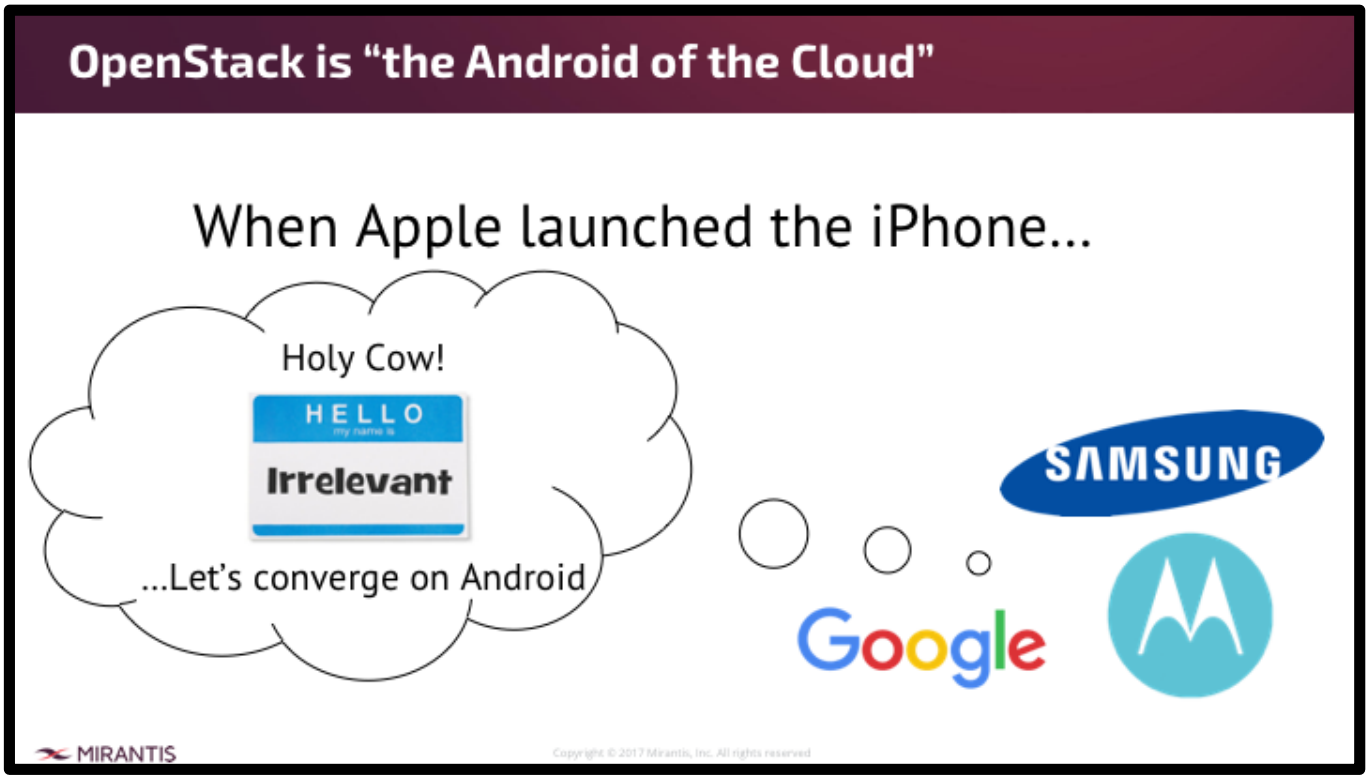
- Why is OpenStack Important?
- What is OpenStack?
  - Definition of cloud
  - Purpose and history
  - OpenStack initiatives – programs
  - Relationships between programs
  - Control Plane Communication Protocols
  - Deployment Topologies

MIRANTIS

Why is OpenStack Important?

Plugin Model

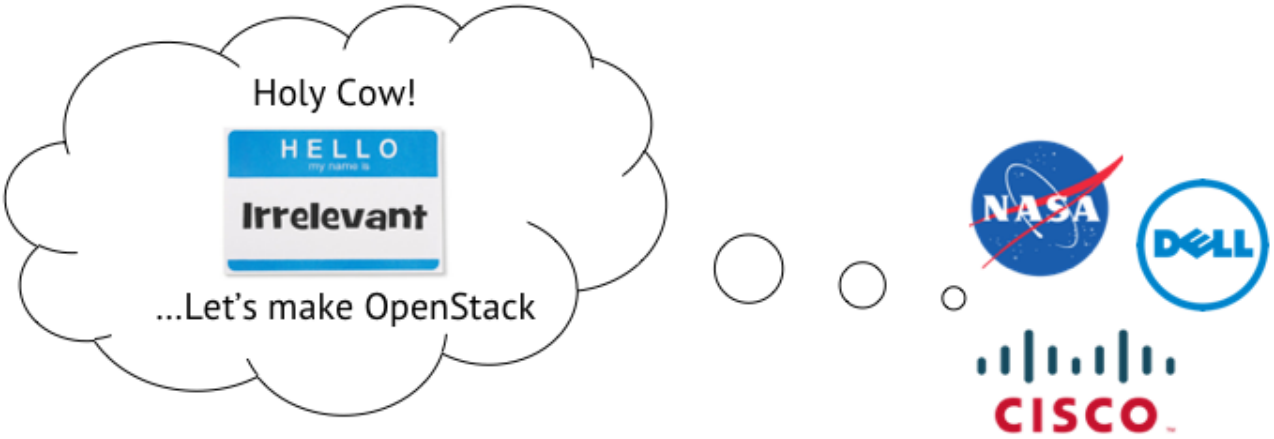
MIRANTIS

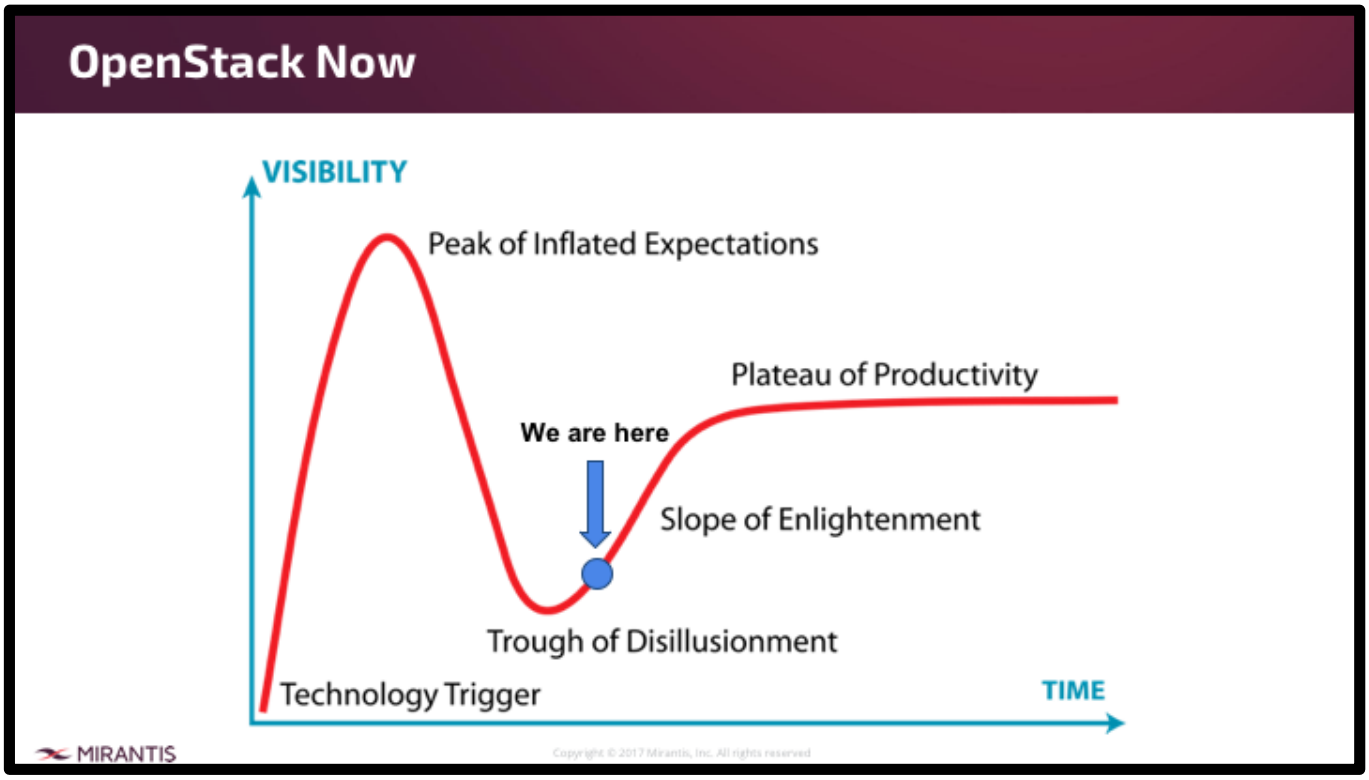


We believe that Openstack will become android of the cloud(at least private).

OpenStack is “the Android of the Cloud”

When Amazon launched AWS…





This slide is an interpretation of the diagrams from the previous slides and a preambule for next section, which is based on the latest OpenStack User Survey.

We at Mirantis believe that although the *visibility* of OpenStack started going down in 2016 - 2017, "Plateau of Productivity" in terms of "hype cycle" is the next phase for OpenStack. The "hype cycle" identifies the phase "Trough of Disillusionment" as "less than 5% of the potential audience has adopted fully", while "Plateau of Productivity" is identified as "20%-30% of the potential audience has adopted". The OpenStack User Survey revealed widespread adoption, maturity and growth, with increased user interest in innovation and agility (see the following slides and the survey).

Source: http://www.prweb.com/releases/2016/10/prweb13785976.htm

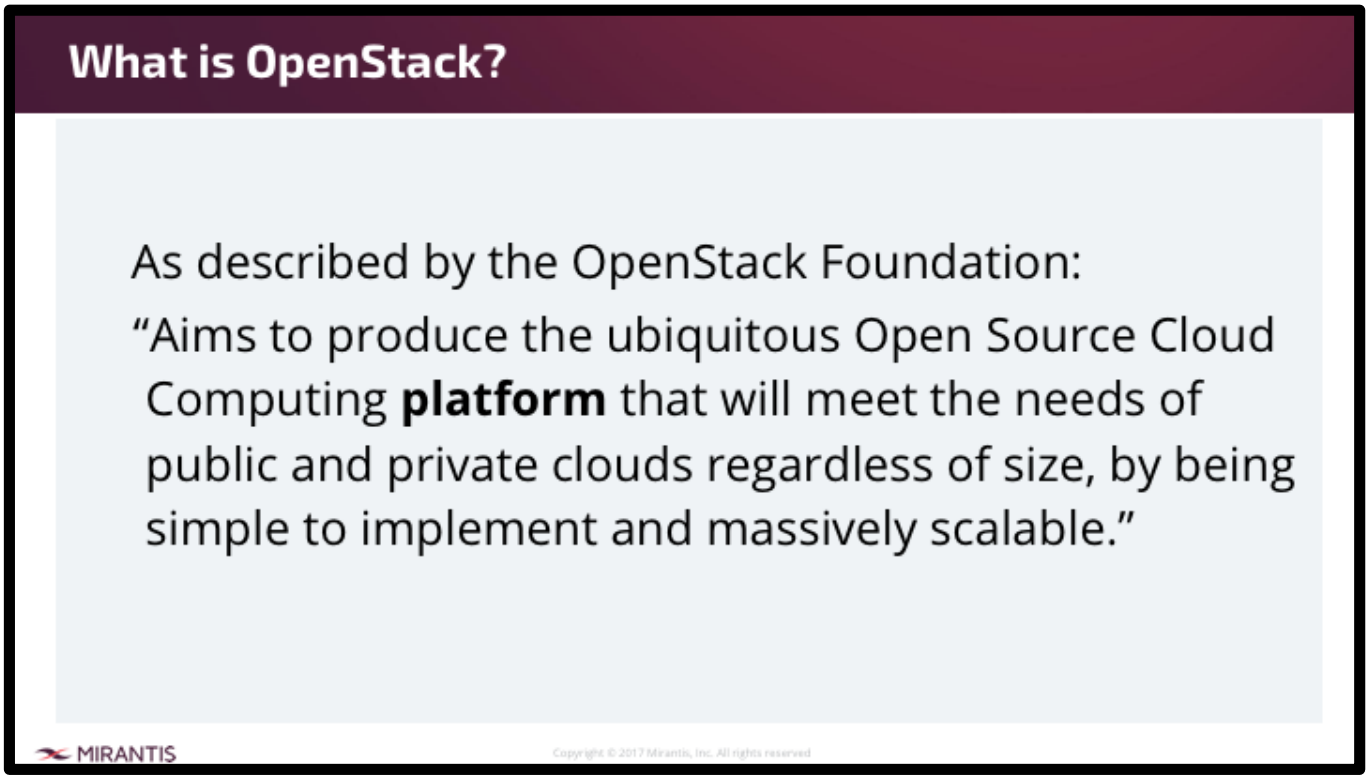
451 Research's Market Monitor service expects revenue from OpenStack business models to exceed $5bn by 2020 and grow at a 35% CAGR*

*) CAGR (Compound annual growth rate) is a business and investing specific term for the geometric progression ratio that provides a constant rate of return over the time period

OpenStack Cloud

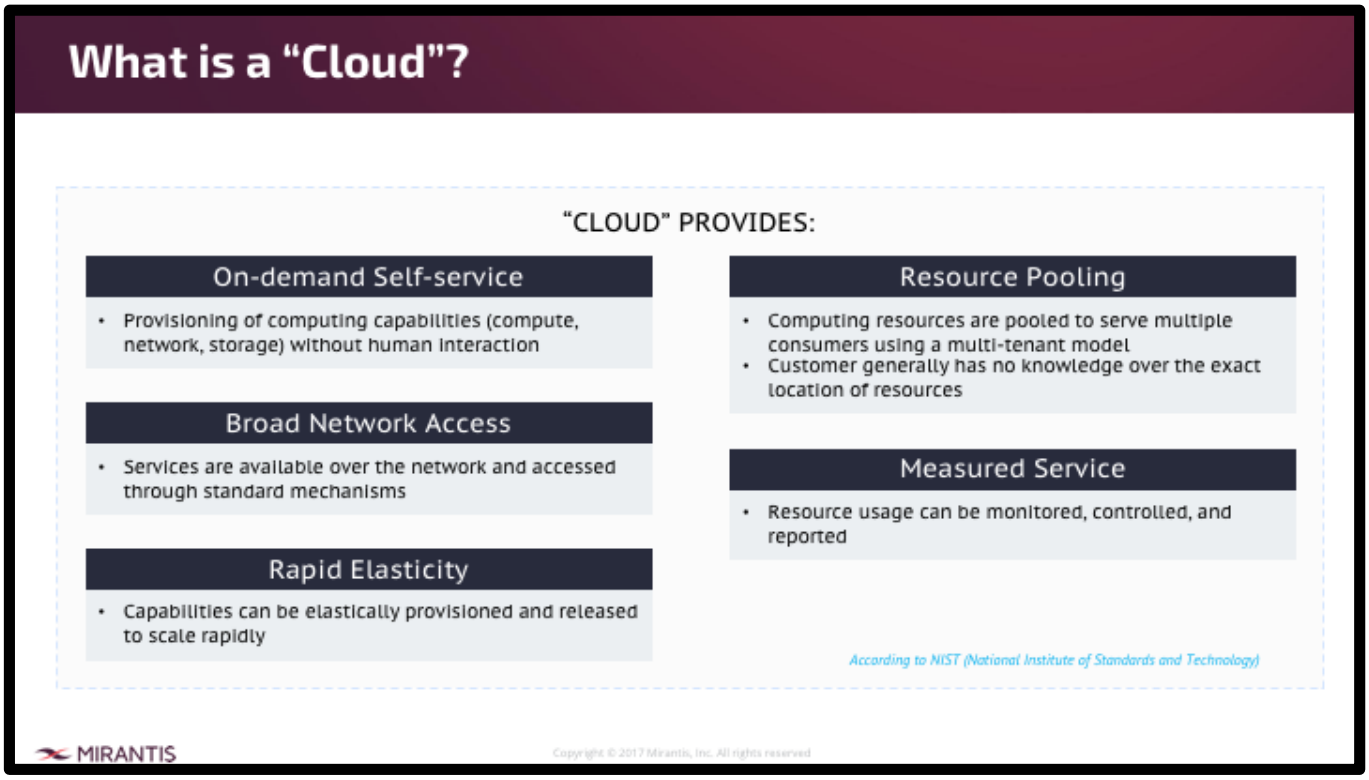
The Cloud Operating System

MIRANTIS



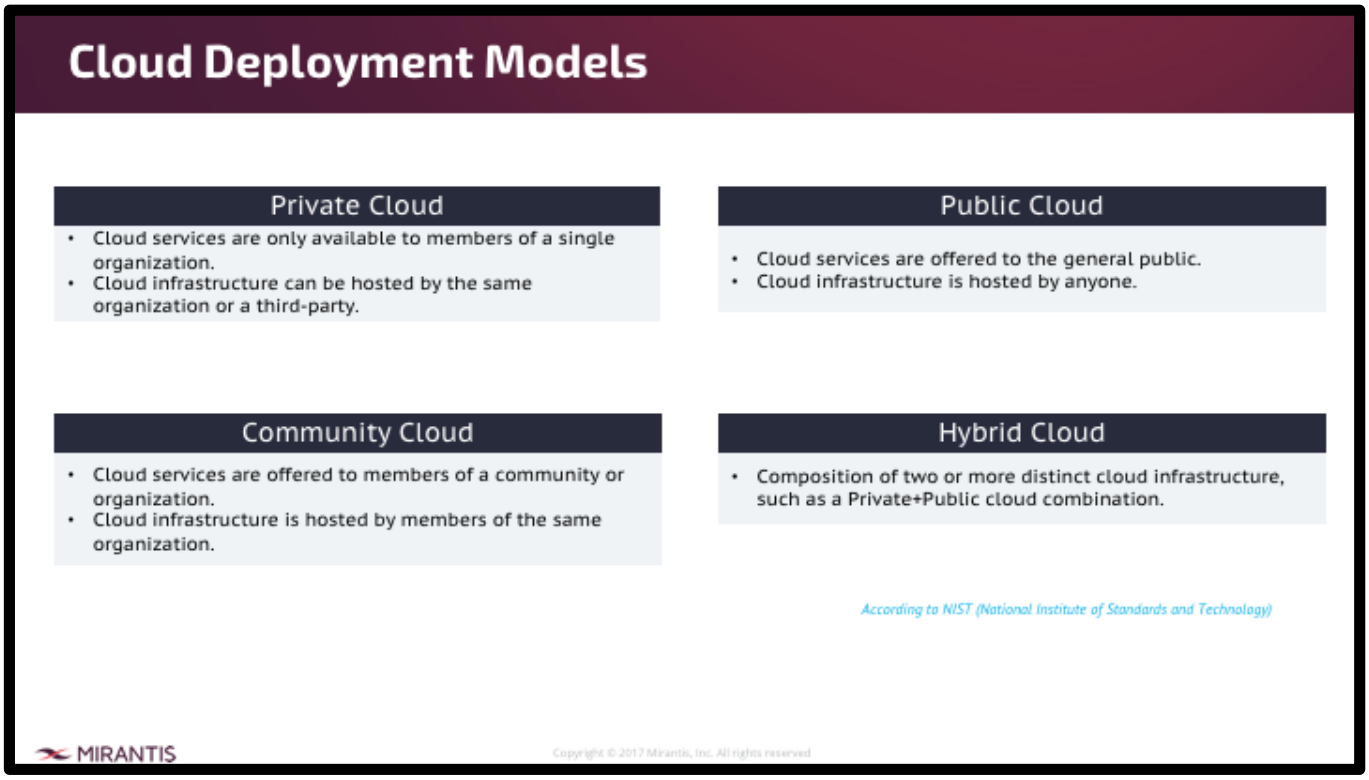
Openstack Foundation plays the same role for Openstack as Linux Foundation for Linux. No single company is behind Openstack. Anyone can join. All trademarks are owned by foundation.

OpenStack Mission - https://wiki.openstack.org/wiki/Main_Page
Mention benefits of open source



Word "Cloud" is very popular. People keep something on the cloud, get something from the cloud. Cloud definition is provided by National Institute of Standards and Technologies should meet 5 requirements:

1. On-demand self service - no human interaction. When you need VM or you want to get new volume or new network you don't need to call any person. You can do it on your own either using command-line or GUI. But you always talk to API. This is the way how Openstack provides exposes functions via API.
2. Service should be accessed via any type of network, like enterprise network or internet using well-known, adopted protocols. Openstack API offers HTTP API to make it possible.
3. Rapid Elasticity. Scale Out and In(horizontal) rapidly and instantly. It can be automated, depending on the load - it is called auto-scaling. In traditional datacenters, e.g. VMWare provided, the whole idea was about consolidation and utilizing existing resources in better way - running big servers with a lot of VM and growing VM. In the cloud we care about scalability, not about consolidation. Adding/removing more units is more cost efficient than improving parameters of single server.
4. Resource Pooling - using a pool of hardware resources and share it between many customers. It can be done by hypervisors, vlans, overlay networks(gre, vxlan..), enterprise luns, lvm. Any resource pooling technology can be plugged into Openstack.
5. Measured service in openstack provided by Ceilometer component. It measures, CPU time, Memory, Storage etc… This information can be used for billing, planning, reporting, autoscaling etc…



NIST defines 4 deployments models of the cloud.
Private cloud is not about ownership. It can be managed or owned by anyone. It basically about physical separation of resources. In private cloud you cannot mix your resource with other customers. Only one organization has access to physical(!) resources in private cloud.

In Public cloud your VM potentially can run on the same physical machine as another customer's VM. One of the most infamous data leak in public cloud was in Digital Ocean - they were didn't erasing data after the customer(in order to extend device lifetime). Security is one of the reason to use Private cloud. Specific cloud tasks may be cheaper to run on the private cloud.

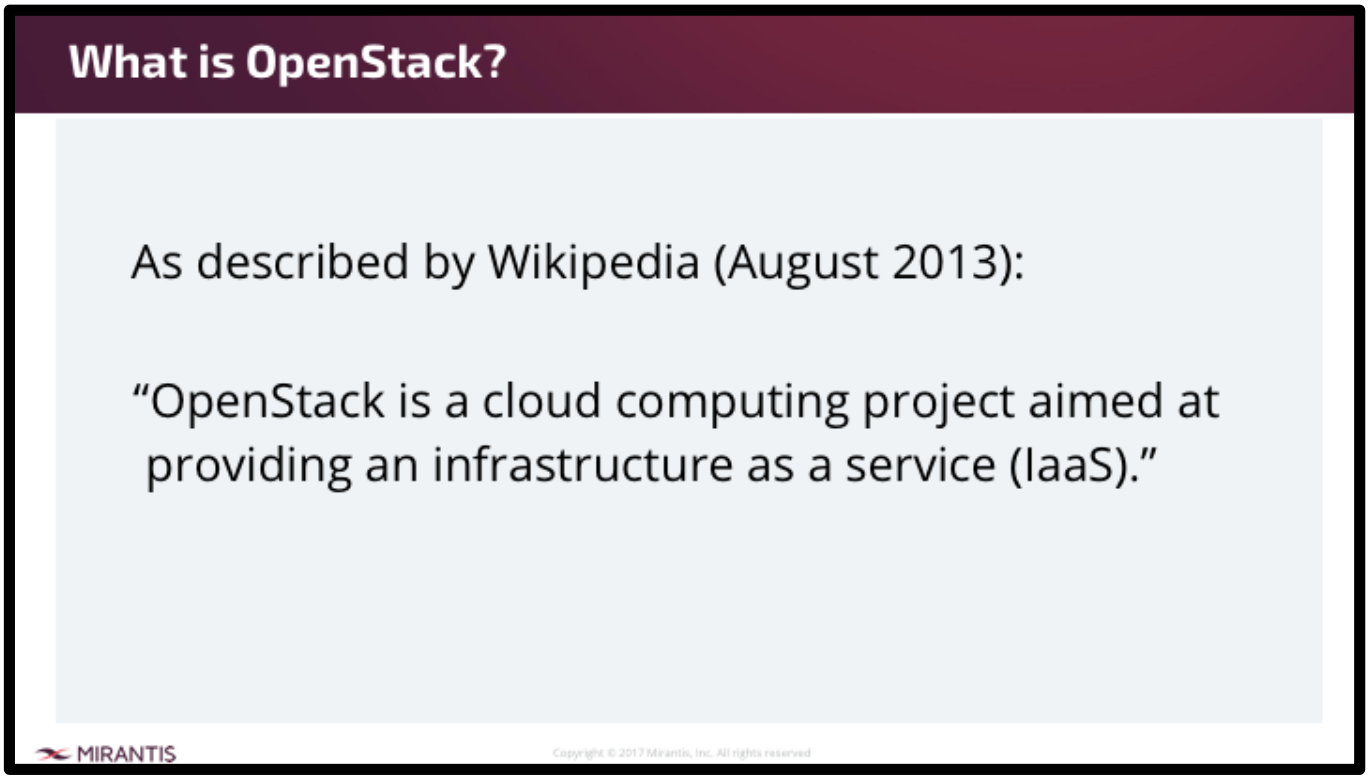
When you run basic workloads on the private cloud, and you need more capacity for limited time - you may use public cloud. This is called hybrid model mixed model.

Community cloud is popular among government, healthcare, universities. It is like club - you have to be invited to participate.

OpenStack Is An IaaS Cloud

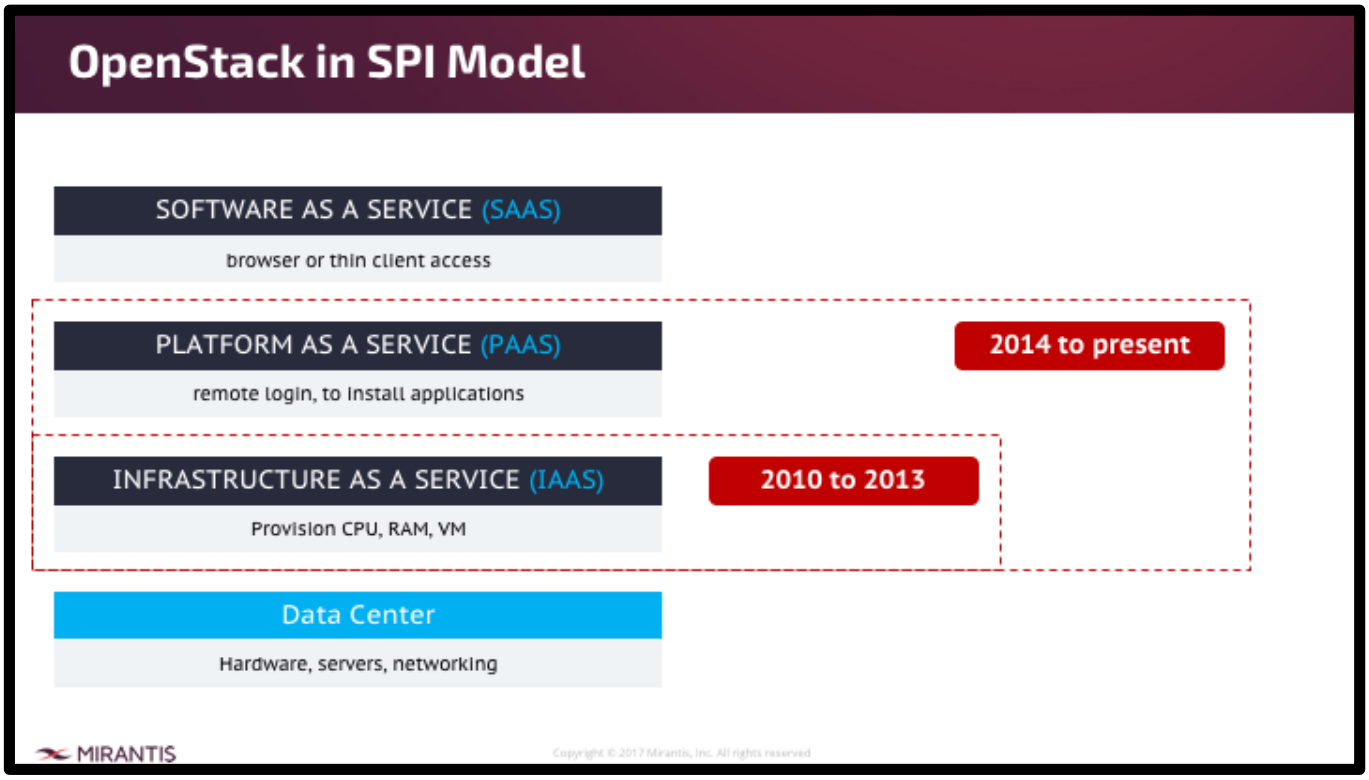
Infrastructure As A Service Cloud

MIRANTIS



From openstack wiki:
The OpenStack Mission: to produce a ubiquitous Open Source Cloud Computing platform that is easy to use, simple to implement, interoperable between deployments, works well at all scales, and meets the needs of users and operators of both public and private clouds.

Version of 2017 openstack.org: Open source software for creating private and public clouds.



SaaS example is Google Docs, Cisco WebEx.. etc. You may use software as service via thin client. You don't have to install software, you don't control the software.

PaaS is used mostly by software developers. PaaS cloud provide environment with additional services like backup your application, provide database, message bus etc...
IaaS expose maximum capabilities of the cloud. Openstack is IaaS

The exposed capabilities of the cloud increases as we move down the SPI hierarchy:

OpenStack IaaS Capabilities

- Compute instances on demand
  - VMs (as well as containers, bare-metal) life-cycle management
    - Provisioning
    - Snapshotting
- Networks to connect compute instances to:
  - Each other
  - External networks
- Storage for compute instances and arbitrary objects
- Multi-tenancy
  - Quotas for different projects, users
  - User can be associated with multiple projects

MIRANTIS

What's Unique about OpenStack?

OpenStack Is Not A Product

MIRANTIS

Openstack is a community which produces a different software components used to

build private and public clouds.

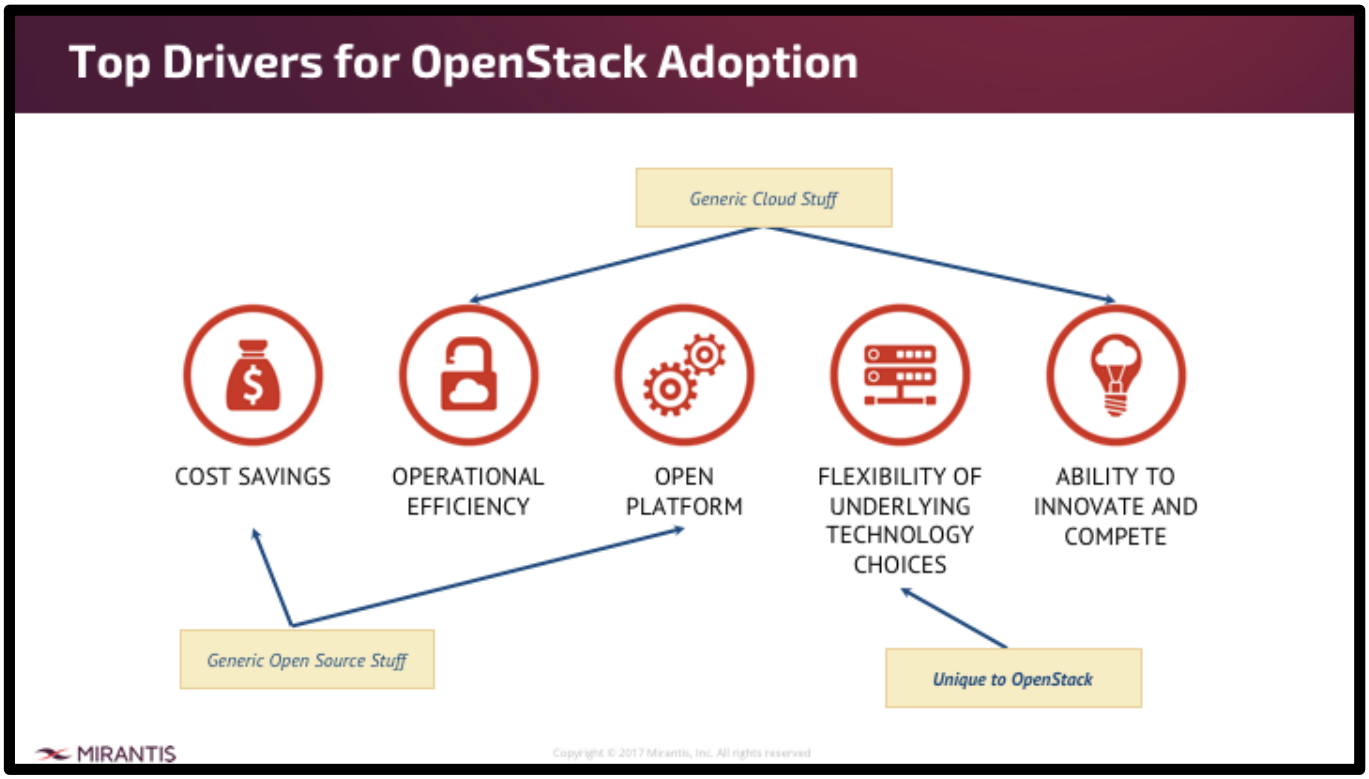
OpenStack Core Programs

Project Name	Description	AWS Equiv	Codename(s)
Compute	Provision and manage large pools of on-demand computing resources	EC2	Nova
Block Storage	Volumes on commodity storage gear, and drivers for turn-key block storage solutions	EBS	Cinder
Object Storage	Petabytes of reliable storage on standard gear	S3	Swift
Networking	L2-focused on-demand networking with some L3 capabilities	VPC	Neutron
Dashboard	Self-service, role-based web interface for users and administrators	Console	Horizon
Metering	Centralized metering data for all services for integration to external billing	N/A	Ceilometer
Identity	Multi-tenant authentication system that ties to existing stores (e.g. LDAP) and Image Service	IAM	Keystone
Image Management	Upload, download, and manage VM images for the compute service	VM Import/Export	Glance
Orchestration	Application orchestration layer that runs on top of and manages OpenStack Compute	CloudFormation, CloudWatch	Heat

MIRANTIS

This is a list of most important openstack components:

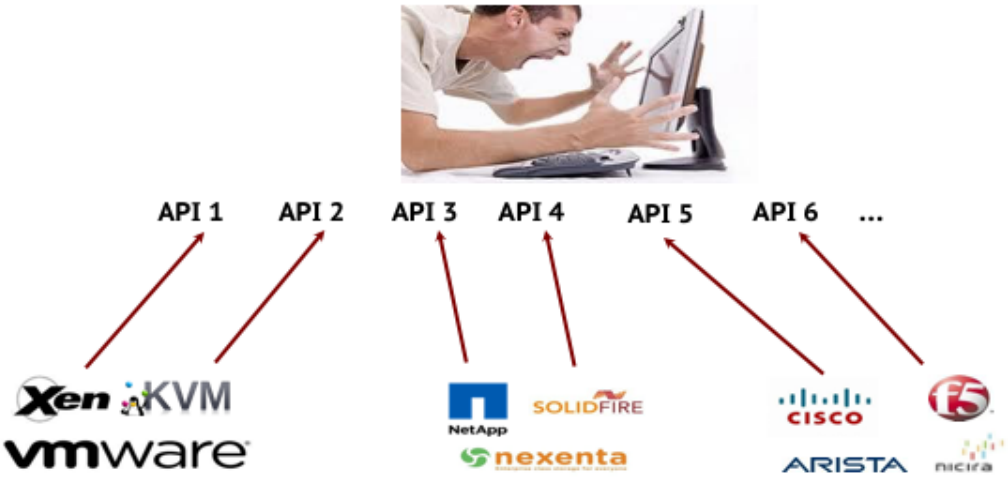
- **Nova**(analogue of EC2) - manages the lifecycle of compute instances in an OpenStack environment. Responsibilities include spawning, scheduling and decommissioning of machines on demand.
- **Cinder**(analogue of EBS) - provides persistent block storage to running instances. Its pluggable driver architecture facilitates the creation and management of block storage devices.
- **Swift**(analogue of S3) - stores and retrieves arbitrary unstructured data objects via a RESTful, HTTP based API. It is highly fault tolerant with its data replication and scale out architecture. Its implementation is not like a file server with mountable directories.
- **Neutron**(analogue of VPC) - enables network connectivity as a service for other OpenStack services, such as OpenStack Compute. Provides an API for users to define networks and the attachments into them. Has a pluggable architecture that supports many popular networking vendors and technologies.
- **Horizon** - provides a web-based self-service portal to interact with underlying OpenStack services, such as launching an instance, assigning IP addresses and configuring access controls.
- **Ceilometer** - monitors and meters the OpenStack cloud for billing, benchmarking, scalability, and statistical purposes.
- **Keystone** - provides an authentication and authorization service for other OpenStack services. Provides a catalog of endpoints for all OpenStack services.
- **Glance** - stores and retrieves virtual machine disk images. OpenStack Compute makes use of this during instance provisioning.
- **Heat** - orchestrates multiple composite cloud applications by using either the native HOT template format or the AWS CloudFormation template format, through both an OpenStack-native REST API and a CloudFormation-compatible Query API.



Top 5 answers to question why do they want to adopt Openstack:

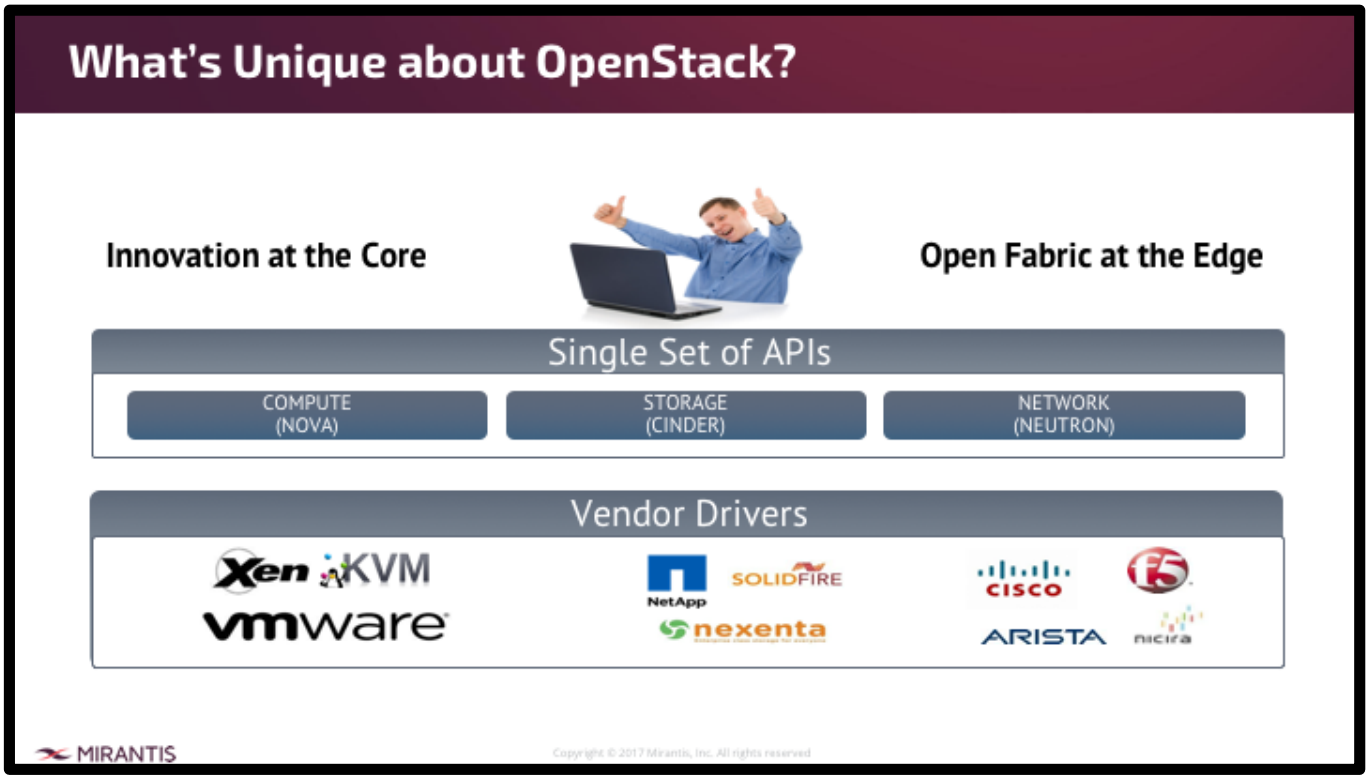
1. Save money. It is not unique to openstack. It is generic open source stuff.
2. Operate their IT infrastructure in more efficient way. They want to automate processes. It is not unique to openstack too. It is generic cloud stuff.
3. People do not to be forced to use any particular operating system or server to run their workload. It is not unique again.
4. People want to innovate and compete - not unique. Generic cloud stuff.
5. You can plug into openstack any technology backend, any hypervisor, any networking backend, any storage backend. Openstack is so open and community is so huge so it will support virtually all the existing backend technologies. Stack of technologies is completely Open(OpenStack). Richard Stallman said in 2007 that compute cloud is something completely evil because all the vendors behind compute cloud want to force you to closed stack of technologies and lock you.

What's Unique about OpenStack?



MIRANTIS

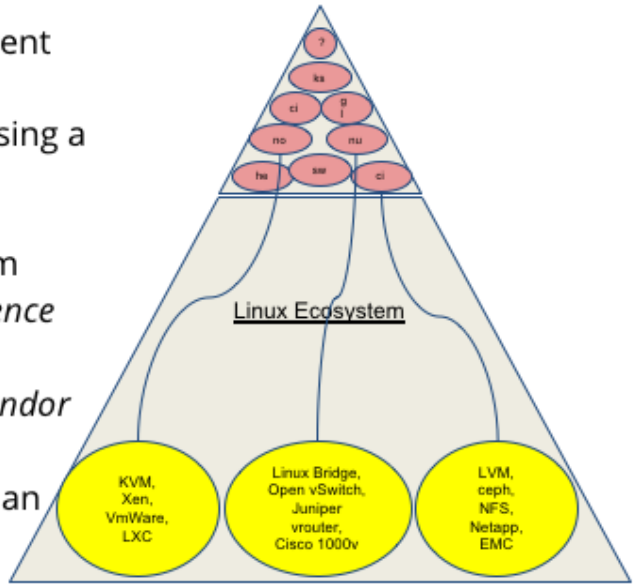
This is how data center looks without Openstack. Different backends, hypervisors, different storages, networking backends etc... Have to use different tools. Different set of administrators.



Openstack provides kind of overlay on top of the APIs and expose it as single consistent set of APIs. API can be consumed by single tool, e.g. Horizon - GUI provided by openstack and it will be actually no matter what is backend. End user is not be aware about underlying API.

Platform + Reference Implementation Plugins

- OpenStack is a platform of independent "*programs*"
- Each program has to be actualized using a plugin
  - Swift Object Store is a product
- To integrate/test/release the platform OpenStack also releases some *Reference Implementation* plugins
- Which also serve as blueprints for *Vendor* plugin implementations
- The features/behavior of programs can change by changing the plugin



Reference Implementation vs. Vendor Plugins

- Reference plugins are always based on open source Linux applications
- When used, OS features will be limited by the features of the underlying Linux application
- While this may work for some OS installation, it will not have the necessary features for all
- Vendor plugins may be necessary to fill the gaps
- This is the OS "*model*"

Market	Needed Features
Top	Large on premise private clouds, customized/demanding workloads (NFV)
Middle	Resiliency (high availability - HA)
	Performance (disk, CPU, network)
	Scalability (number of users)
	Monitoring (alerts, self management/healing)
	Security (prevention, detection)
	Life cycle management (upgrade, update)
Bottom	Small cluster, minimal requirements

OpenStack Programs Overview

Historical Progression

MIRANTIS

OpenStack History

** Pre-July 2010 is predicated by Rackspace Cloud Files project (Swift), NASA Nebula project (Nova)*

DATE	REL	PROGRAMS	TYPE	NOTE
Jul 2010	N/A		PoC	** Rackspace Hosting & NASA joint launch*
Oct 2010	**AUSTIN**	Nova, Swift	PoC	
Feb 2011	**BEXAR**	Nova, Glance, Swift	PoC	
Apr 2011	**CACTUS**	Nova, Glance, Swift	PoC	*** 6 month development cycle starts*
Sep 2011	**DIABLO**	Nova, Glance, Swift	**Prod**	*1st production release (Cactus) at Internap (10/27)*
Apr 2012	**ESSEX**	Nova, Glance, Swift, **Horizon, Keystone**	**Prod**	*Common web UI and shared authentication mechanism added*
Sep 2012	**FOLSOM**	Nova, Glance, Swift, Horizon, Keystone, **Quantum, Cinder**	**Prod**	*OpenStack Foundation Established*
Apr 2013	**GRIZZLY**	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder	**Prod**	*Ceilometer and Heat incubation projects added*
Oct 2013	**HAVANA**	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, **Heat, Ceilometer**	**Prod**	*Quantum is renamed to Neutron*

MIRANTIS

Pre-July 2010 is predicated by Rackspace Cloud Files, NASA, Anso Labs, etc.

Cactus – first commercial cloud

Diablo – multi host networking

Essex – sufficiently stable to use for many purposes

Folsom – Everything as a service

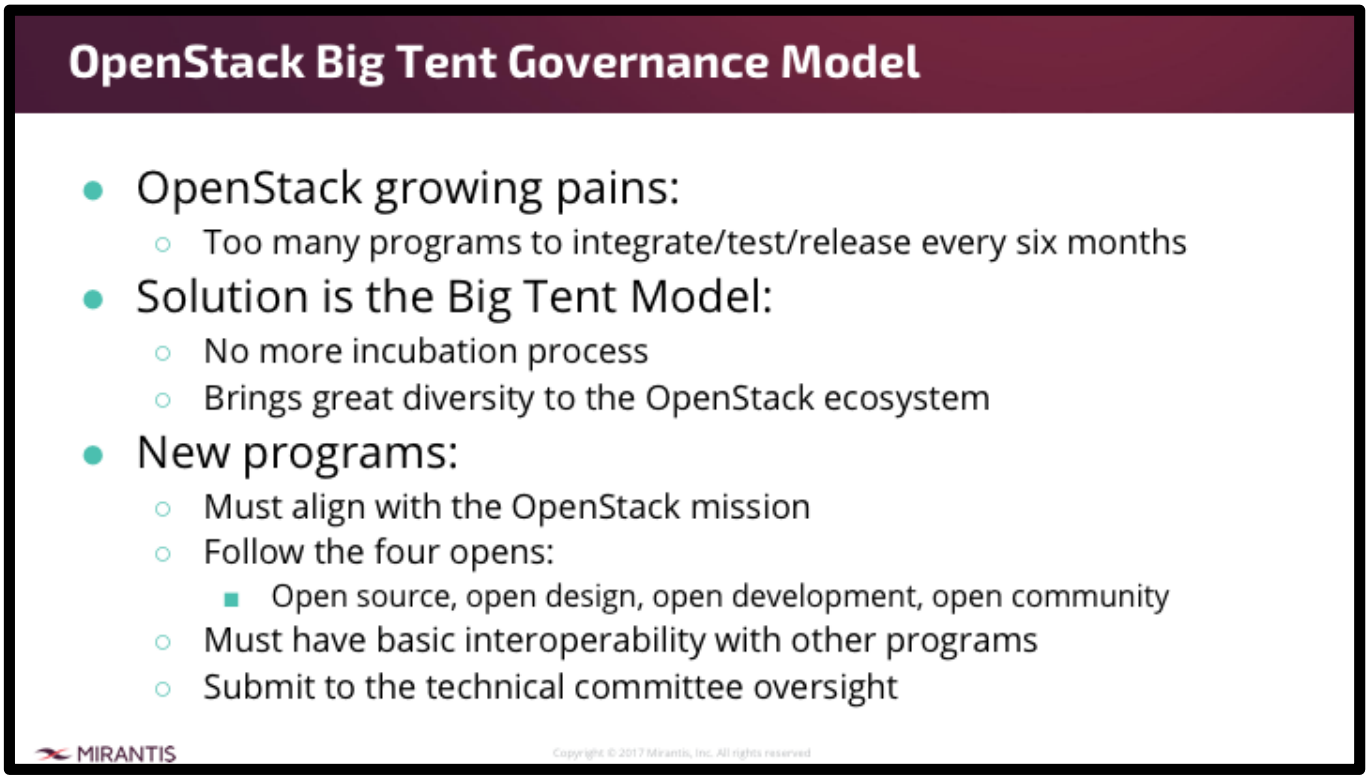
Grizzly – Nova volumes removed, compute nodes do not directly access database, security groups for Neutron are fixed

Havana – added Ceilometer and Heat, expecting Trove and possibly Savanna

OpenStack History (cont'd)

DATE	REL	PROGRAMS	TYPE	NOTE
Apr 2014	**ICEHOUSE**	All above, **Trove**	**Prod**	*More focus on PaaS*
Oct 2014	**JUNO**	All above, **Sahara**	**Prod**	*First release of Data Processing service*
Apr 2015	**KILO**	All above, **Ironic**	**Prod**	*First release of Bare Metal Service*
Oct 2015	**LIBERTY**	All above, *Zaqar, Barbican, Manila, Designate, Mistral, Magnum, Murano, Rally*	**Prod**	*Big tent governance model*
Apr 2016	**MITAKA**	All above	**Prod**	*User experience, manageability and scalability*
Oct 2016	**NEWTON**	All above	**Prod**	*Heterogeneous environments (containers, bare metal)*
Feb 2017	**OCATA**	All above	**Prod**	*Stabilization, user experience, manageability, scalability*
Autumn	**PIKE**	All above	**Prod**	*Coming soon!*

MIRANTIS

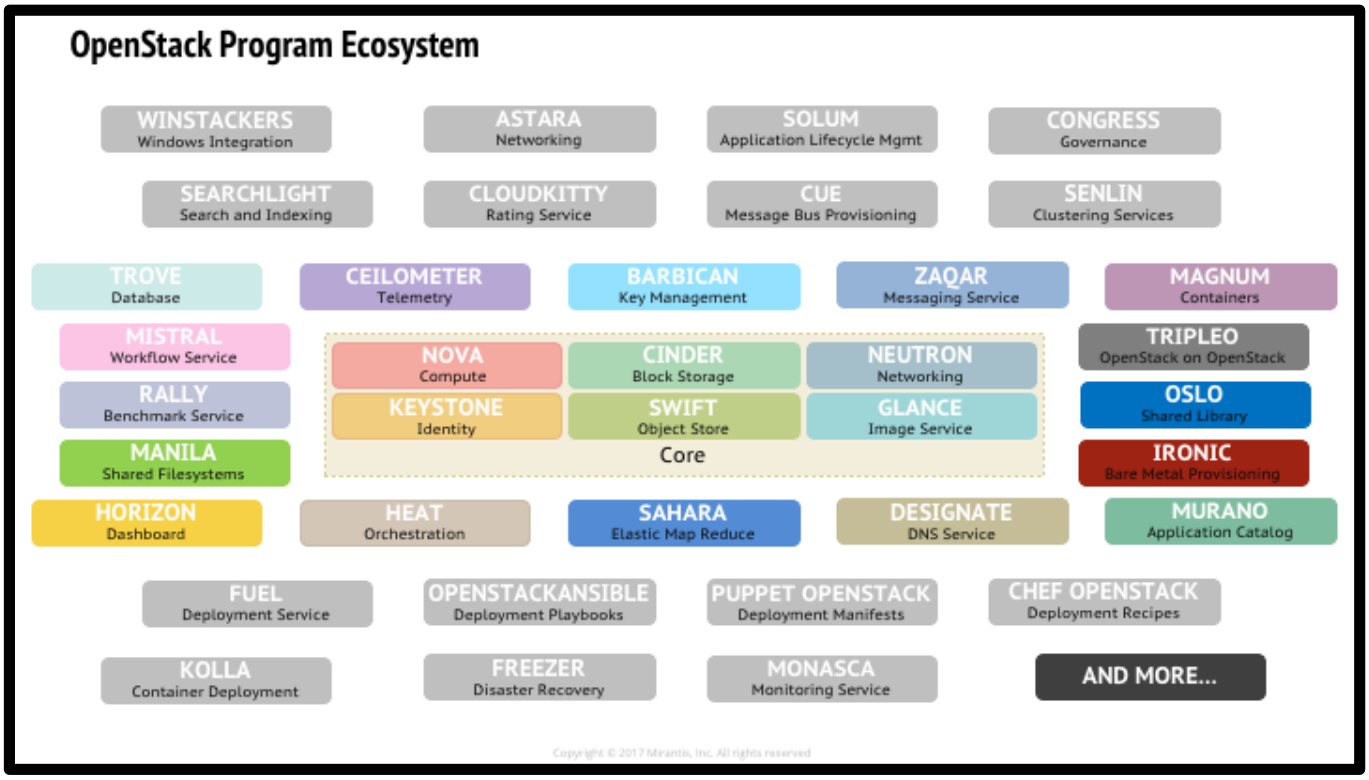


video from Vancouver summit: https://www.openstack.org/videos/vancouver-2015/the-big-tent-a-look-at-the-new-openstack-projects-governance

big-tent description: http://drbacchus.com/bigtent/
https://gist.github.com/rbowen/320263de7c974d5a98a2

big-tent project tags: https://governance.openstack.org/tc/reference/tags/
starter-kit:compute tag: https://governance.openstack.org/tc/reference/tags/starter-kit_compute.html
tc:approved-release tag: https://governance.openstack.org/tc/reference/tags/tc_approved-release.html
team:diverse-affiliation tag: https://governance.openstack.org/tc/reference/tags/team_diverse-affiliation.html

Interop working group (formerly DefCore):
https://github.com/openstack/interop/blob/master/doc/source/process/CoreDefinition.rst
https://github.com/openstack/interop/blob/master/doc/source/process/CoreCriteria.rst



[12/01/2015] List derived from sources below:
http://governance.openstack.org/reference/projects/index.html
https://github.com/openstack/governance/blob/master/reference/projects.yaml

The 6 core programs as per what is one the project navigator page:
https://www.openstack.org/software/project-navigator

The programs with a coloured box represent those programs that were either previously known as Core, Integrated, and Incubated programs. The grey coloured boxes are for the programs that were accepted into Big Tent.

What the Instructor can talk about

The major take away from this slide is OpenStack's growth and diversity. I think this slide is a good representation of the ecosystem. OpenStack started off with a few programs, evolved with the break out of services like Cinder, Quantum/Neutron, then saw programs introduced to fill gaps that were not addressed at all (Ceilometer, Heat, Trove, etc.). With the change to the big tent, the barrier to entry is far lower, giving deployers more possibilities.

Synopsis of each program that is "new" on this slide:

Astara (https://wiki.openstack.org/wiki/Astara): To provide an integrated network service orchestration (routing, firewall, load balancing, vpn) for connecting and security multi-tenant OpenStack environments.

Chef OpenStack (https://wiki.openstack.org/wiki/Chef): The Chef cookbooks for OpenStack allow to automate the building, operation and consumption of OpenStack cloud deployments.

Cloudkitty (https://wiki.openstack.org/wiki/CloudKitty): CloudKitty is a rating component for OpenStack. Its goal is to process data from different metric backends and implement rating rule creation. Its role is to fit in-between the raw metrics from OpenStack and the billing system of a provider for chargeback purposes.

Congress (https://wiki.openstack.org/wiki/Congress): To provide governance as a service across any collection of cloud services in order to monitor, enforce, and audit policy over dynamic infrastructure.

Cue (https://wiki.openstack.org/wiki/Cue): To provide a multi-tenant service that offers scalable and reliable provisioning and management capabilities for off-the-shelf message brokers.

Freezer (https://wiki.openstack.org/wiki/Freezer): To provide integrated tooling for backing up, restoring, and recovering file systems, instances, or database backups. Disaster Recovery features are also provided, as Freezer allows to recover your data in case of one or more core components are not avaialble (i.e. Swift, Keystone, DB). This is achieved by supporting multiple and parallel media to store backups.

Kolla (https://wiki.openstack.org/wiki/Kolla): To provide production-ready containers and deployment tools for operating OpenStack clouds.

Monasca (https://wiki.openstack.org/wiki/Monasca): To provide a multi-tenant, highly scalable, performant, fault-tolerant monitoring-as-a-service solution for metrics, complex event processing and logging. To build an extensible platform for advanced monitoring services that can be used by both operators and tenants to gain operational insight and visibility, ensuring availability and stability.

OpenStackAnsible (https://wiki.openstack.org/wiki/OpenStackAnsible): Deploying OpenStack from source in a way that makes it scalable while also being simple to operate, upgrade, and grow.

Winstackers (https://wiki.openstack.org/wiki/Os-win): To facilitate the integration of Hyper-V, Windows and related components into OpenStack. This includes producing projects containing Hyper-V / Windows related code, commonly used in OpenStack scenarios.

Puppet OpenStack (https://wiki.openstack.org/wiki/Puppet): The Puppet modules for OpenStack bring scalable and reliable IT automation to OpenStack cloud deployments.

Searchlight (https://wiki.openstack.org/wiki/Searchlight): To provide advanced and scalable indexing and search across multi-tenant cloud resources.

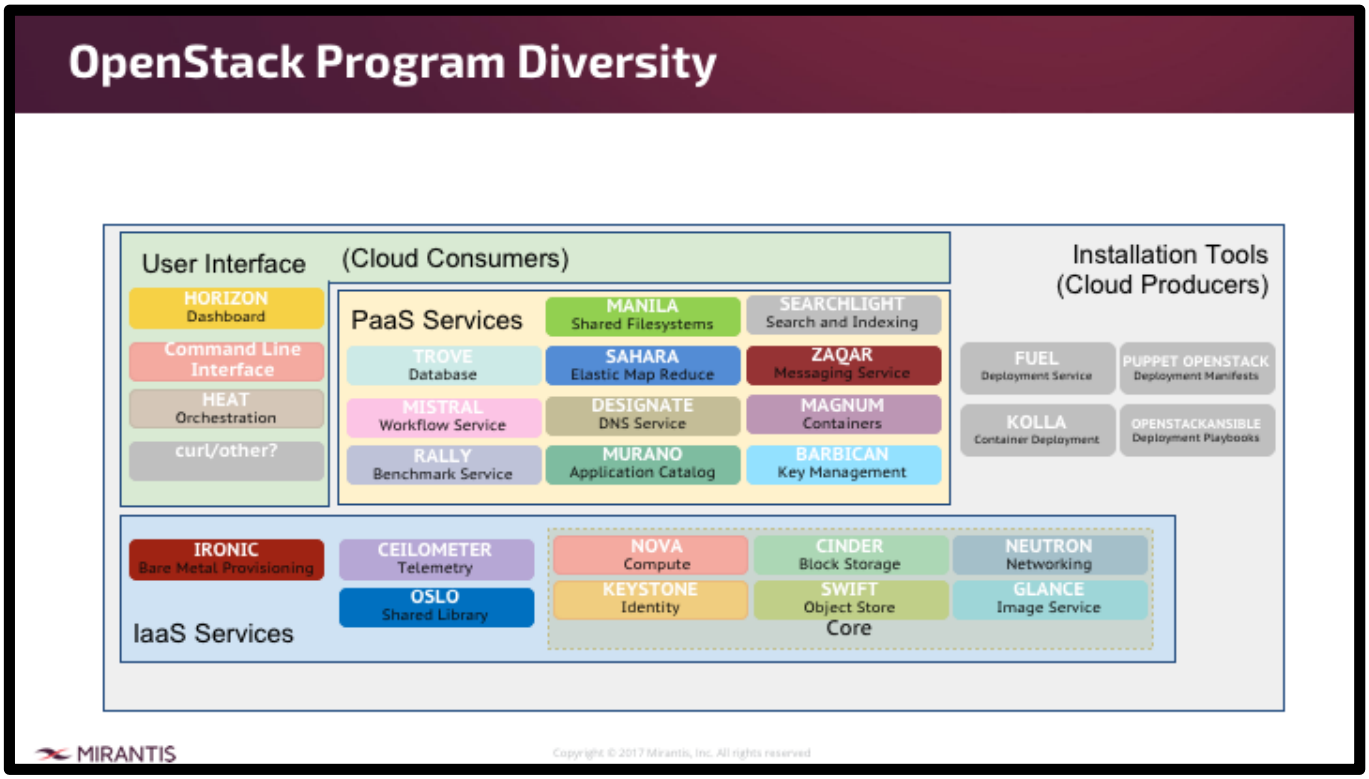
Senlin (https://wiki.openstack.org/wiki/Senlin): To implement clustering services and libraries for the management of groups of homogeneous objects exposed by other OpenStack services.

Solum (https://wiki.openstack.org/wiki/Solum): To make cloud services easier to consume and integrate with your application development process by automating the source-to-image process, and simplifying app-centric deployment.

Each OpenStack Program

- Is also a “top-level” OpenStack component
- Has an elected “Project Technical Lead” (PTL)
- Has separate developers and design teams
- Has separate development and release cycles
- Has a well defined public API
  - With the exception of Horizon, which is the Web GUI, all other projects have a RESTful (JSON/HTTP) API
- Has a separate persistent database layer

MIRANTIS

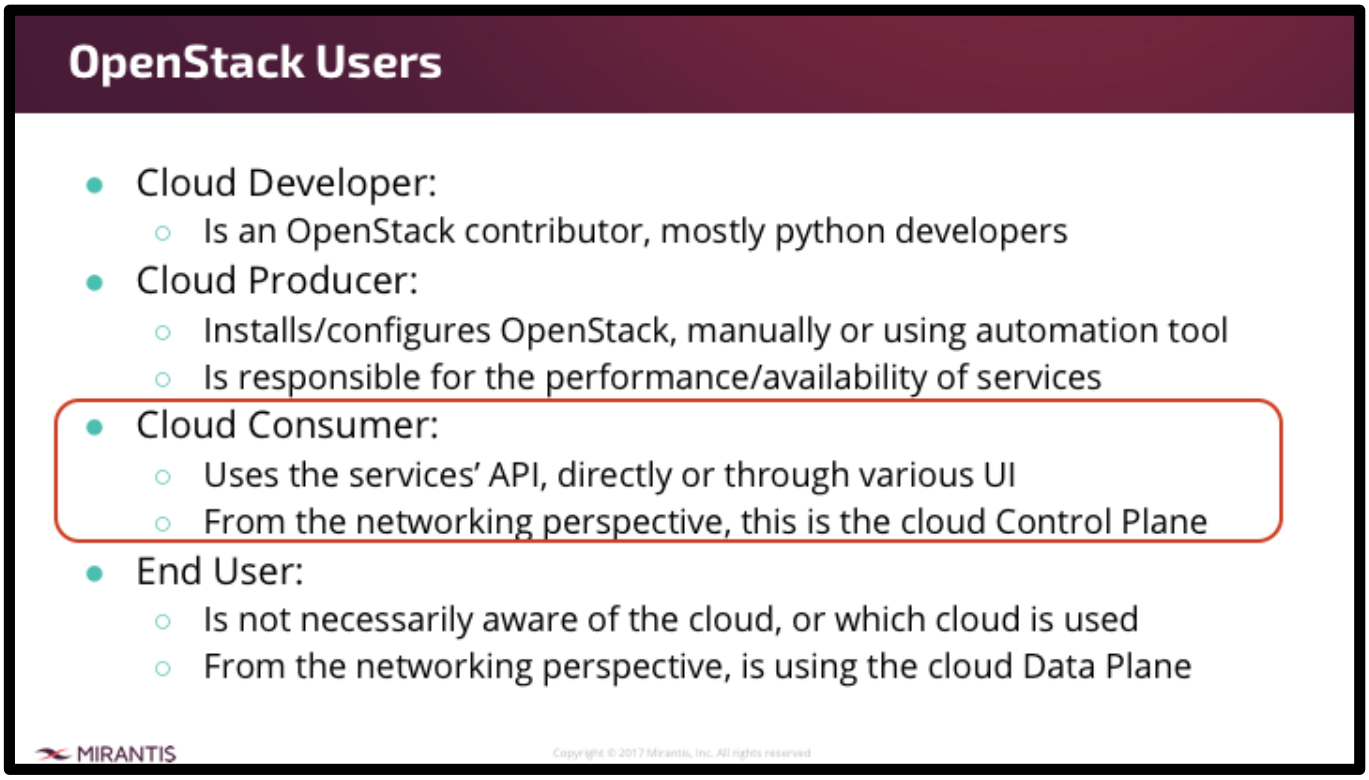


https://www.openstack.org/software/project-navigator/

OpenStack Control Plane

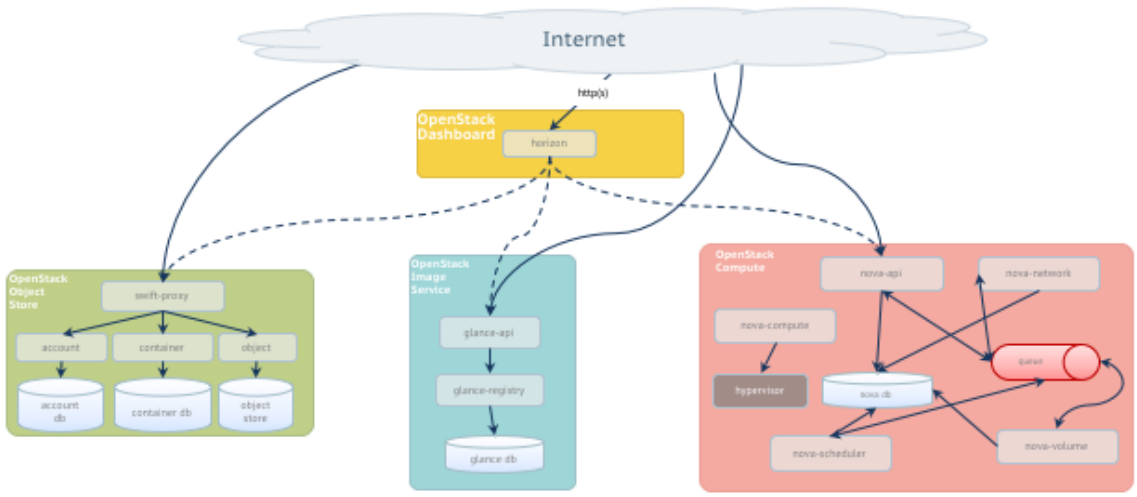
Service API Access

MIRANTIS



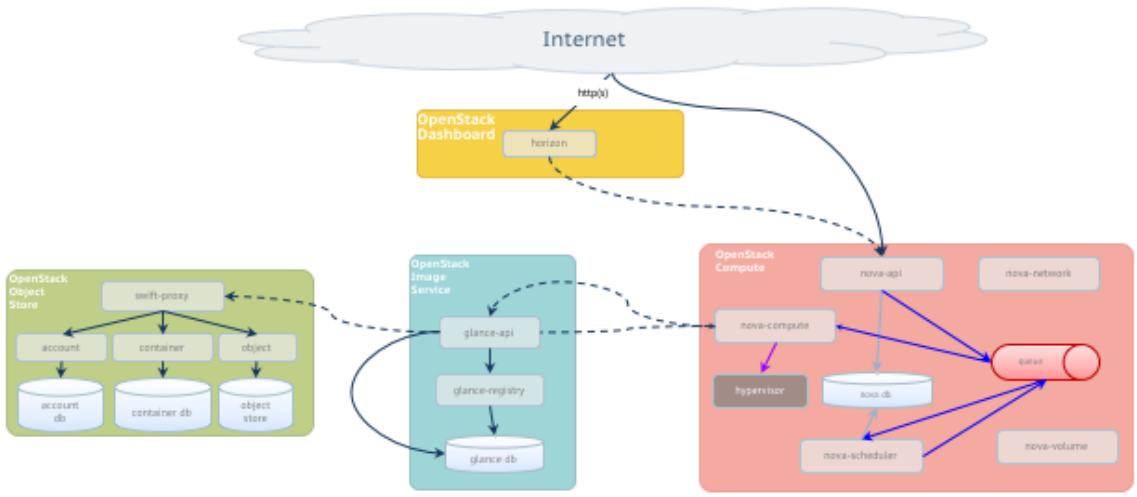
List of different type of users who come in contact with OpenStack.

OpenStack Beginning (Cactus)



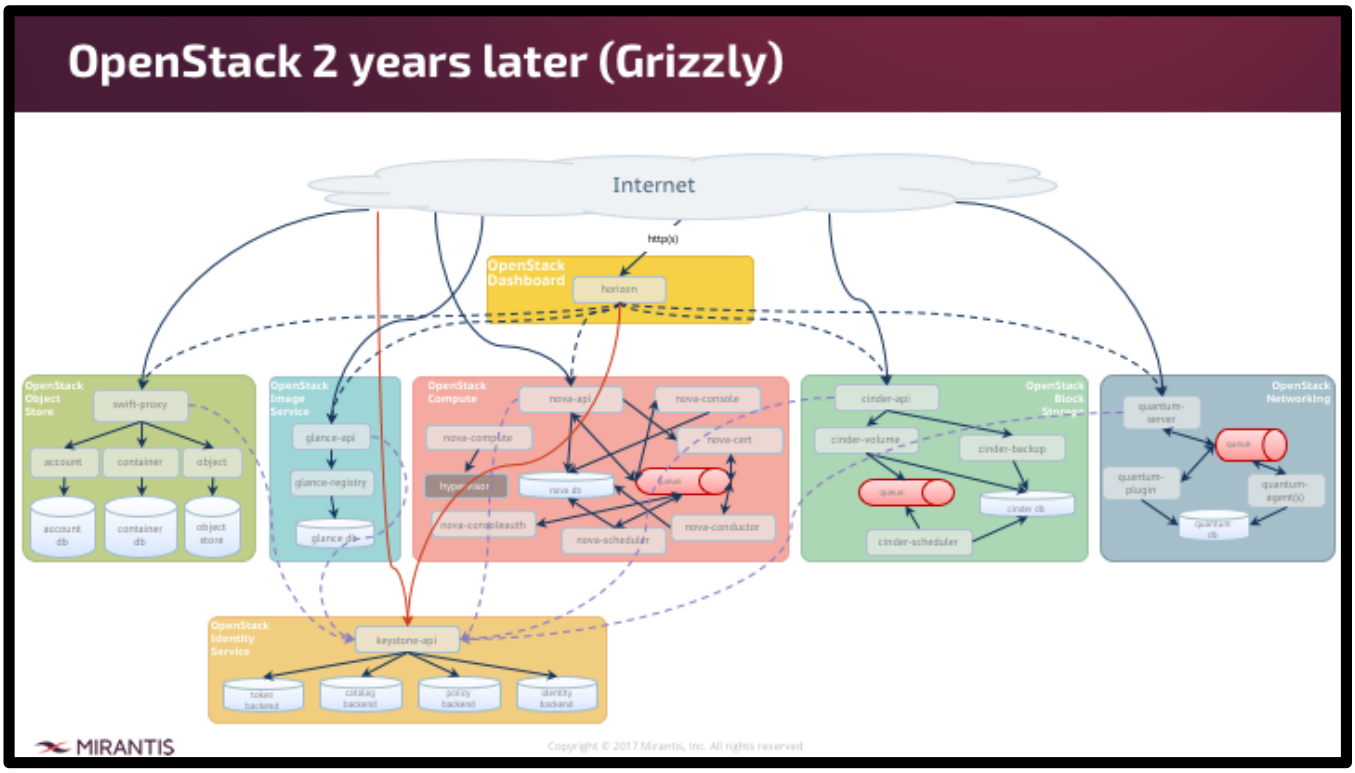
We have only glance + nova + dashboard

OpenStack Launch VM (Cactus)



MIRANTIS

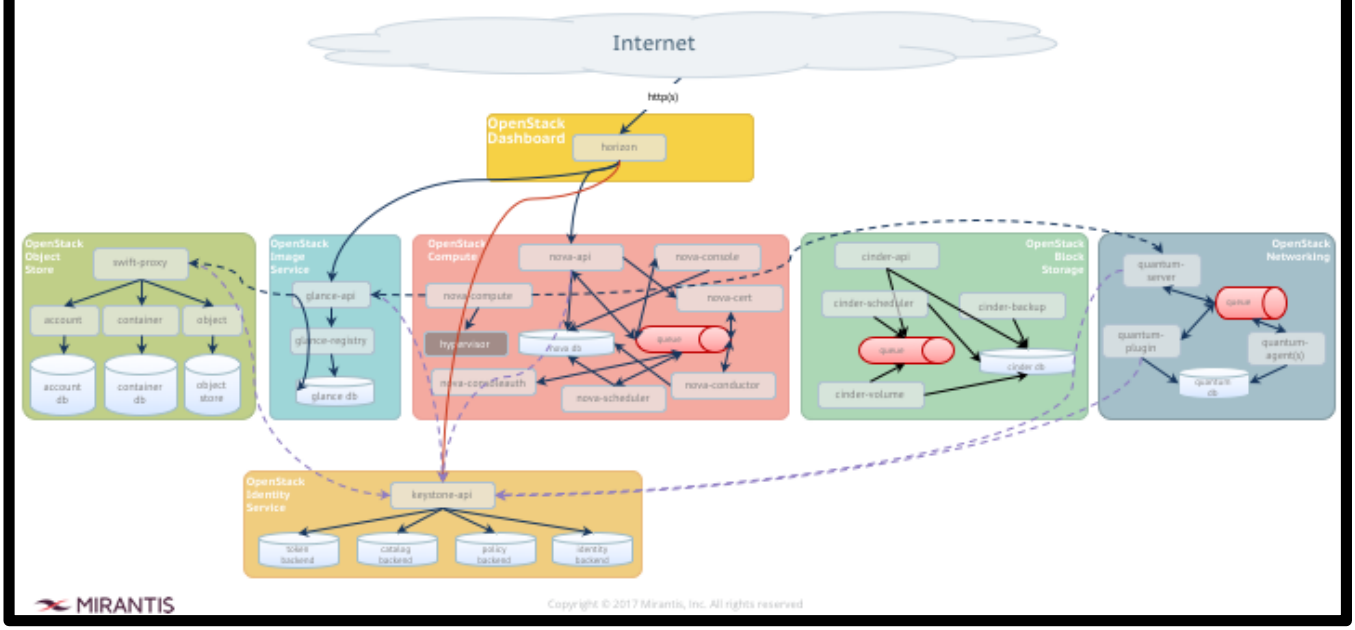
We have only glance + nova + dashboard

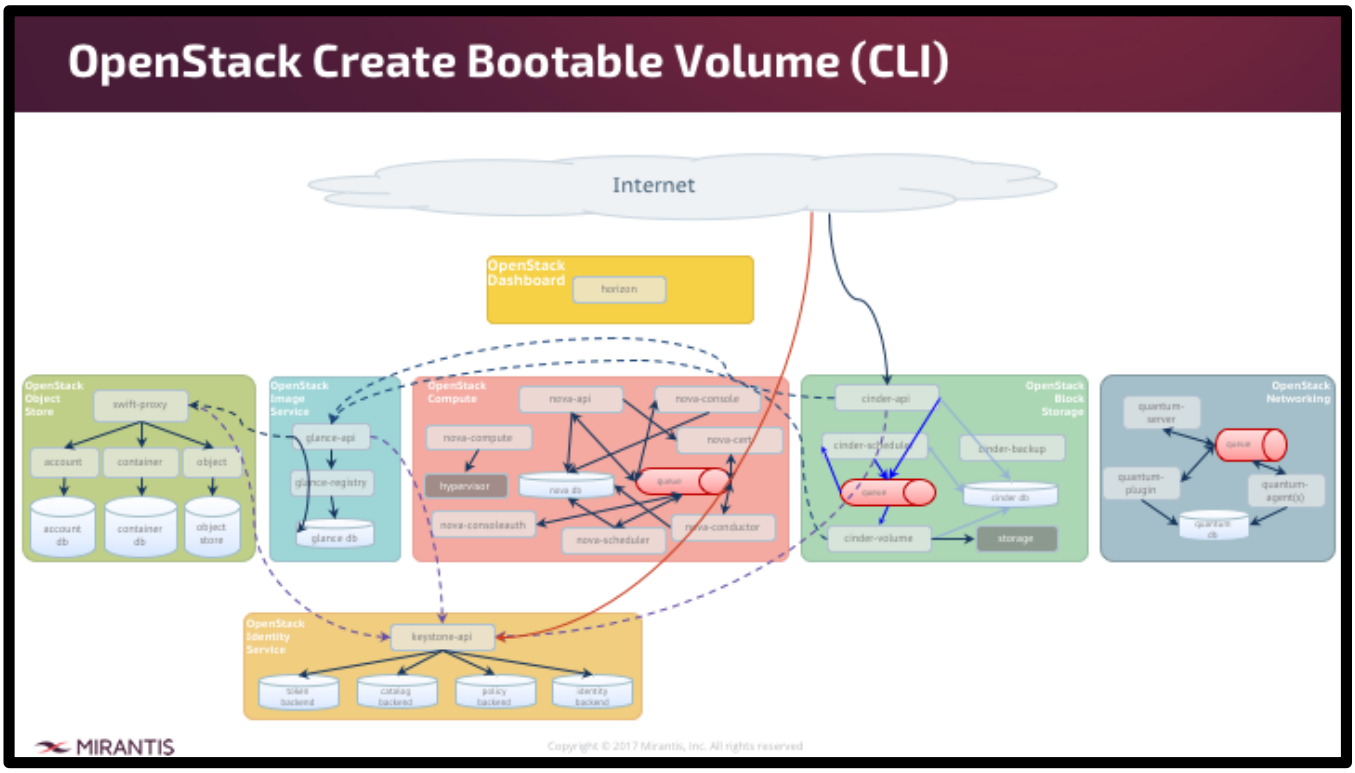


Two years later, when OpenStack architecture is finalized.

Original post:
http://www.solinea.com/2013/06/15/openstack-grizzly-architecture-revisited/

OpenStack VM Boot (ephemeral storage)





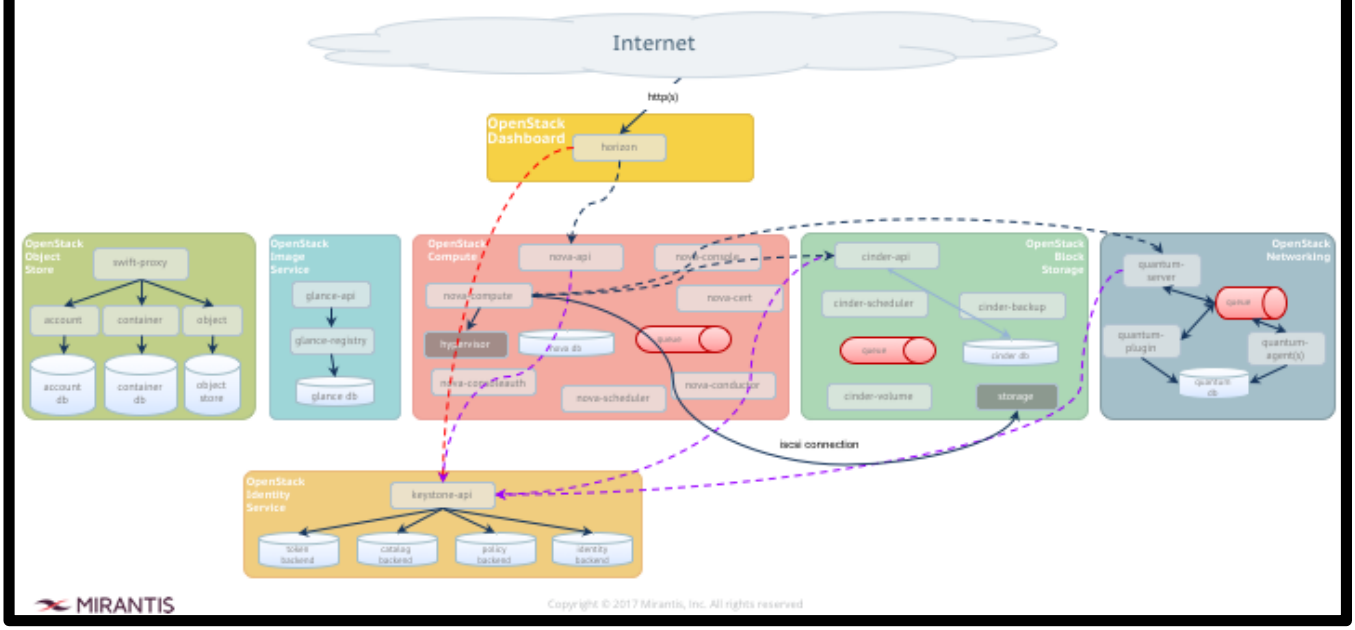
grep "Cloning " or "Attempting download" or "Temporary image " c_*.log for flow.
Source in:

cinder/volume/flows/manager/create_volume.py: _copy_image_to_volume()

cinder/volume/drivers/lvm.py: fetch_to_raw()

Image conversion: `LOG.debug("%s was %s, converting to %s ", image_id, fmt, volume_format)`

OpenStack VM Boot From Volume



Developer Trends

- Decoupling of features
  - e.g. Nova-volume became Cinder
- APIs to communicate
- Common generic API/Infrastructure (Oslo)
- Backends & drivers (everything's pluggable)
- Delegation of operations
  - e.g. Call Heat to provision infrastructure

MIRANTIS

Conflict – Neutron vs nova-network

Lab And Break

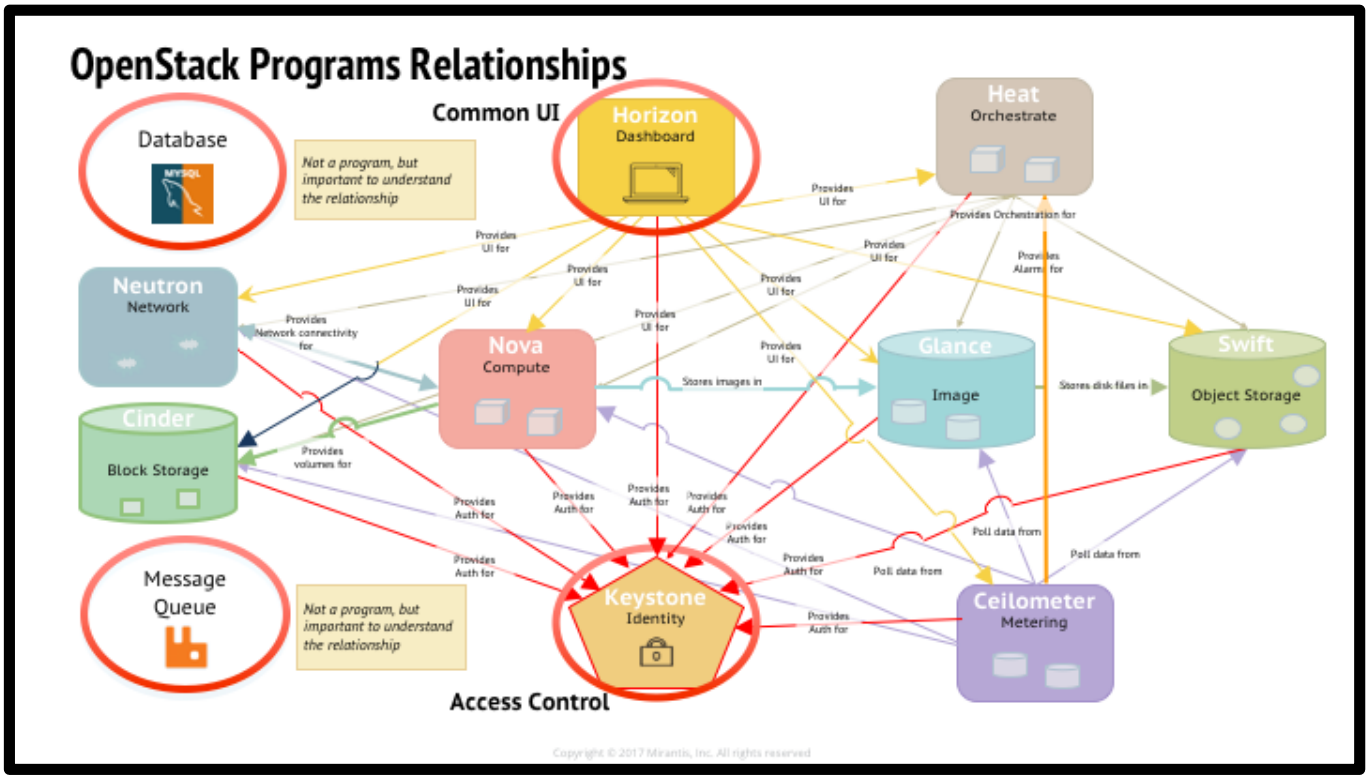
- Take a 10 minute break
- Lab Explore the OpenStack Environment (15 minutes)
- Lab discussion (5 minutes)

MIRANTIS

OpenStack API

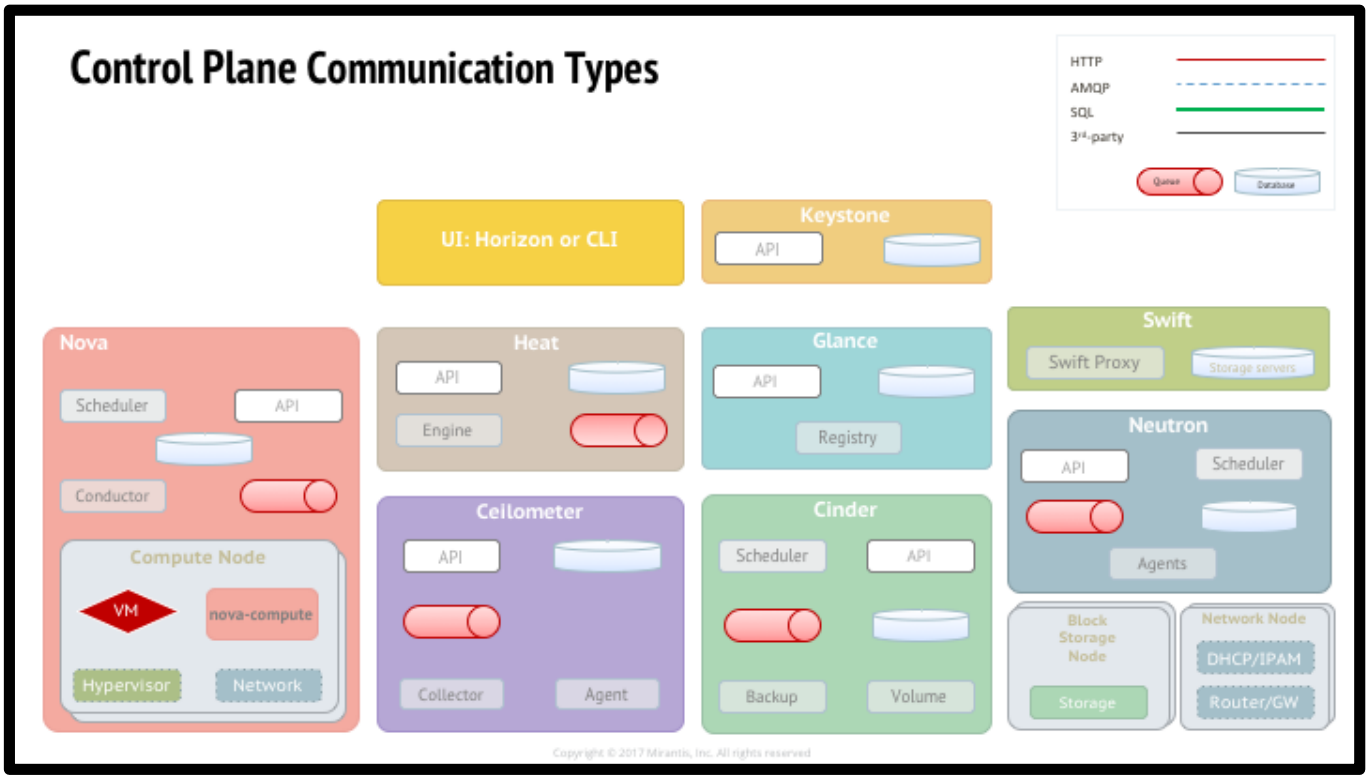
Component Relationship

MIRANTIS



For Ceilometer it directly polls only from Nova, Glance and Swift for now, for everything else just a subscriber to rpc events (see http://docs.openstack.org/developer/ceilometer/architecture.html)

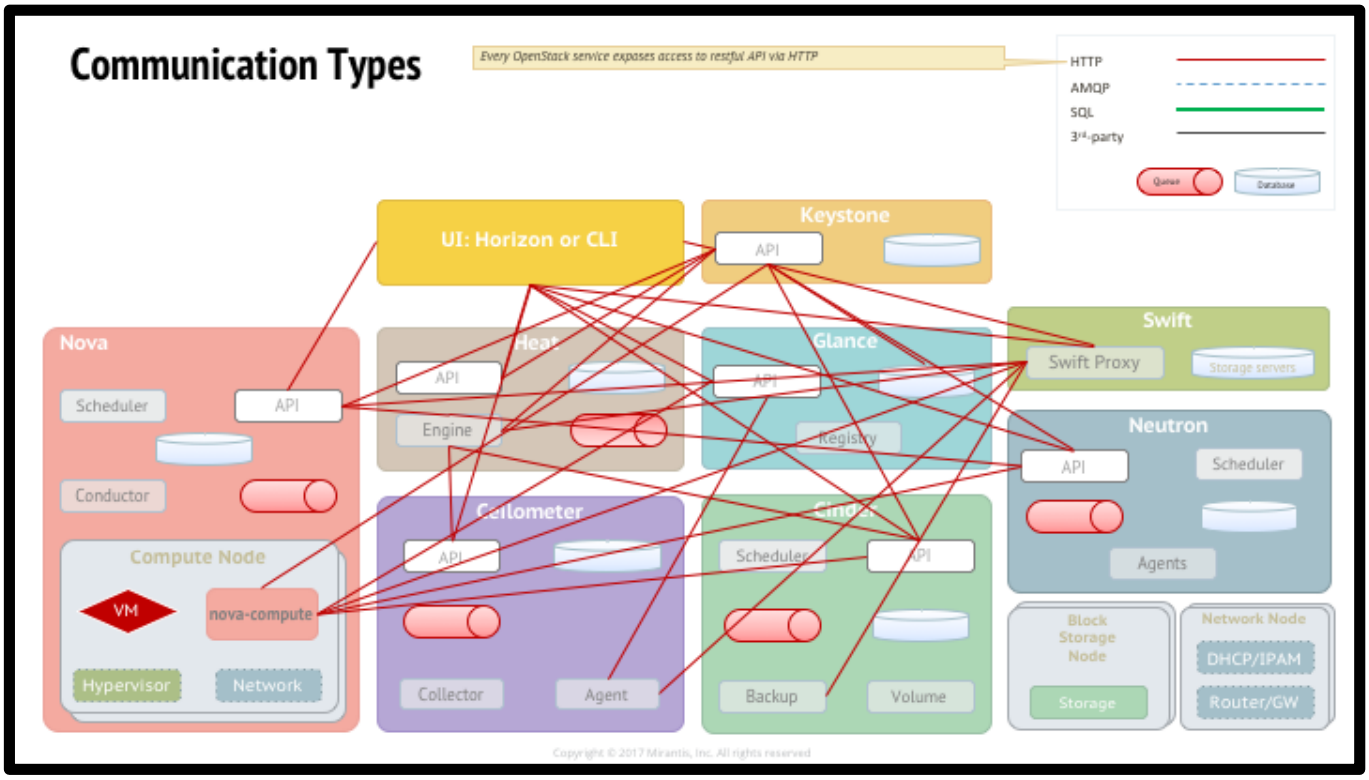
DB: sql database for data storage. Used by all components
Web Dashboard: potential external component that talks to the api
api: component that receives http requests, converts commands and communicates with other components via the queue or http (in the case of object store)
Auth Manager: component responsible for users/projects/and roles. Can backend to DB or LDAP. This is not a separate binary, but rather a python class that is used by most components in the system.
object store: http server that replicates s3 api and allows storage and retrieval of images
scheduler: decides which host gets each vm
network: manages ip forwarding, bridges, and vlans
compute: manages communication with hypervisor and virtual machines.



The HTTP connectors from Heat are drawn based on https://github.com/openstack/heat/blob/master/heat/engine/clients.py

For AMQP:

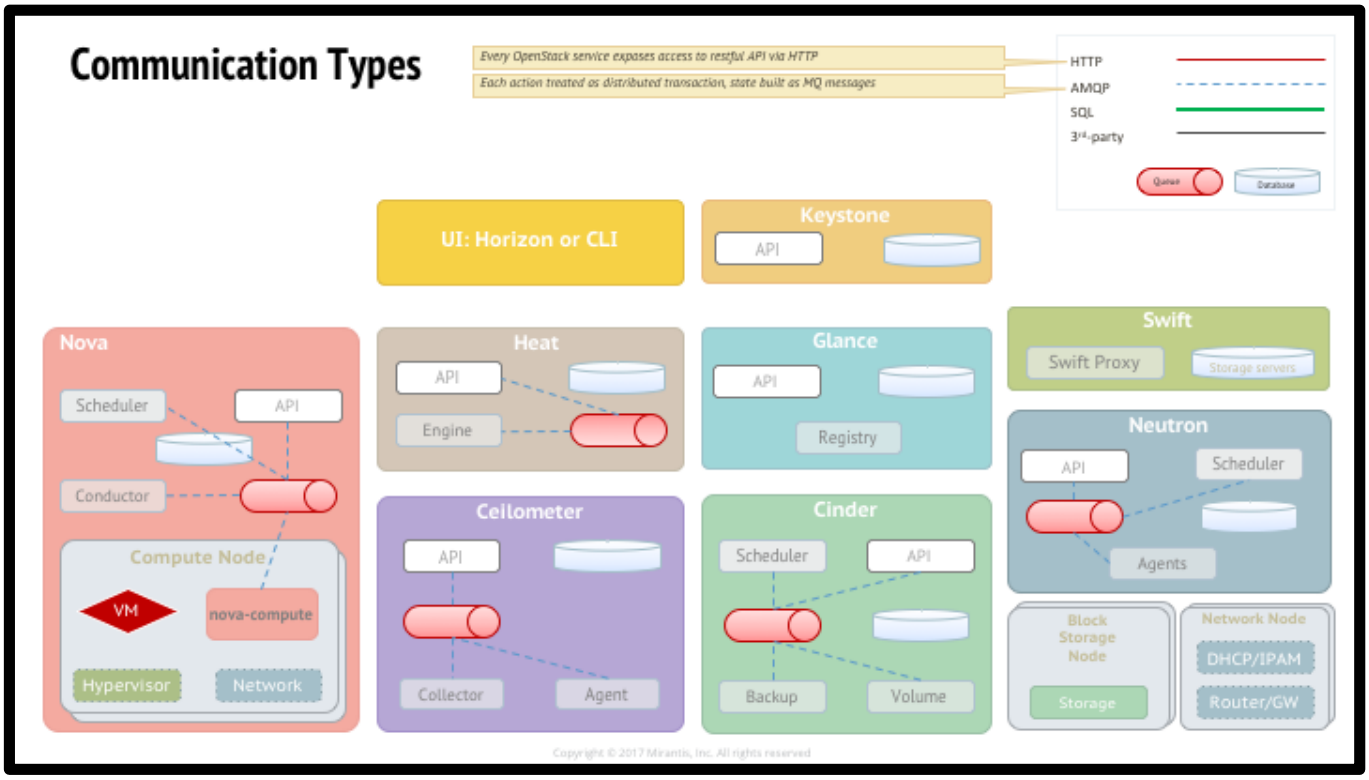
- All the subcomponents talk between each other
- Also Ceilometer uses it to get the data from Cinder and Neutron (components like Neutron and Cinder can publish the notification messages into the queue, Ceilometer Collectors are listening on those), this is not shown on this diagram. Also, there are different conversations to replace that mechanism with more flexible agents.



The HTTP connectors from Heat are drawn based on https://github.com/openstack/heat/blob/master/heat/engine/clients.py

For AMQP:

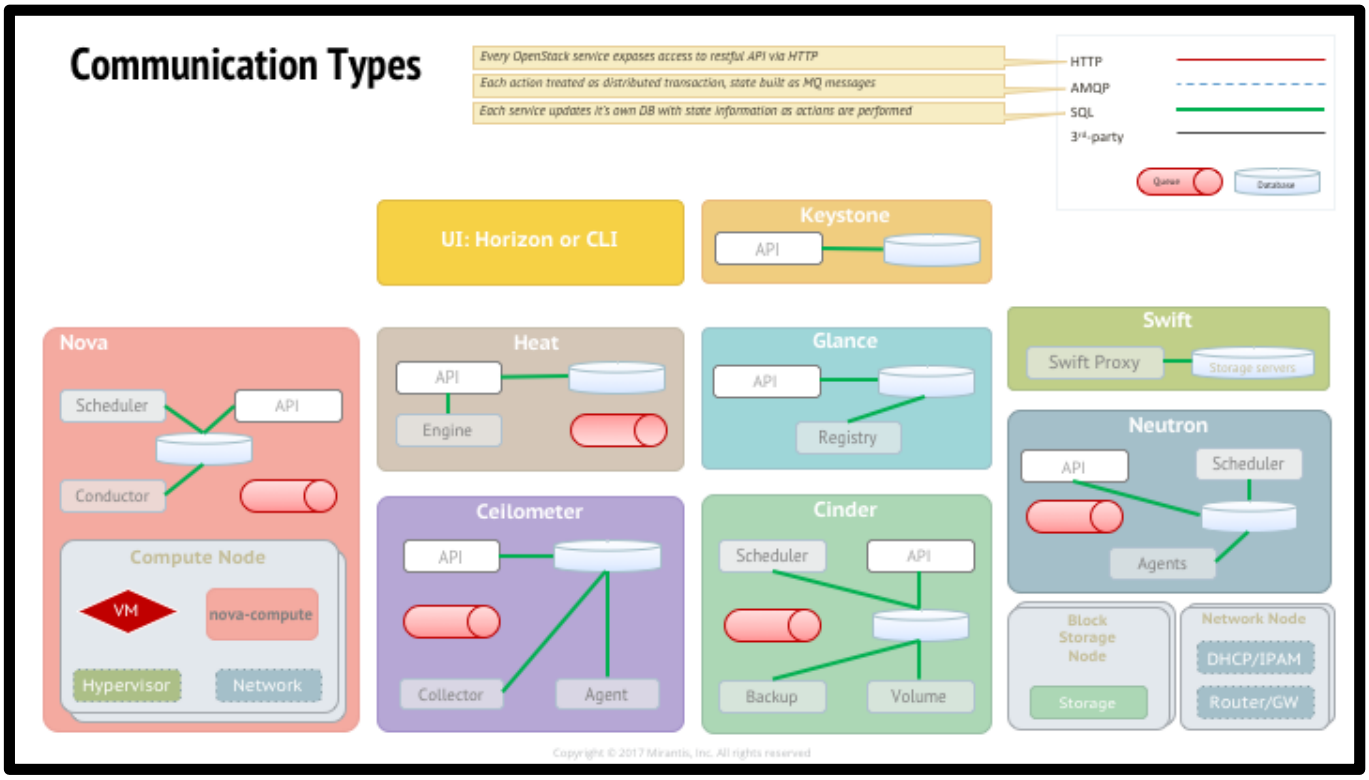
- All the subcomponents talk between each other
- Also Ceilometer uses it to get the data from Cinder and Neutron (components like Neutron and Cinder can publish the notification messages into the queue, Ceilometer Collectors are listening on those), this is not shown on this diagram. Also, there are different conversations to replace that mechanism with more flexible agents.



The HTTP connectors from Heat are drawn based on https://github.com/openstack/heat/blob/master/heat/engine/clients.py

For AMQP:

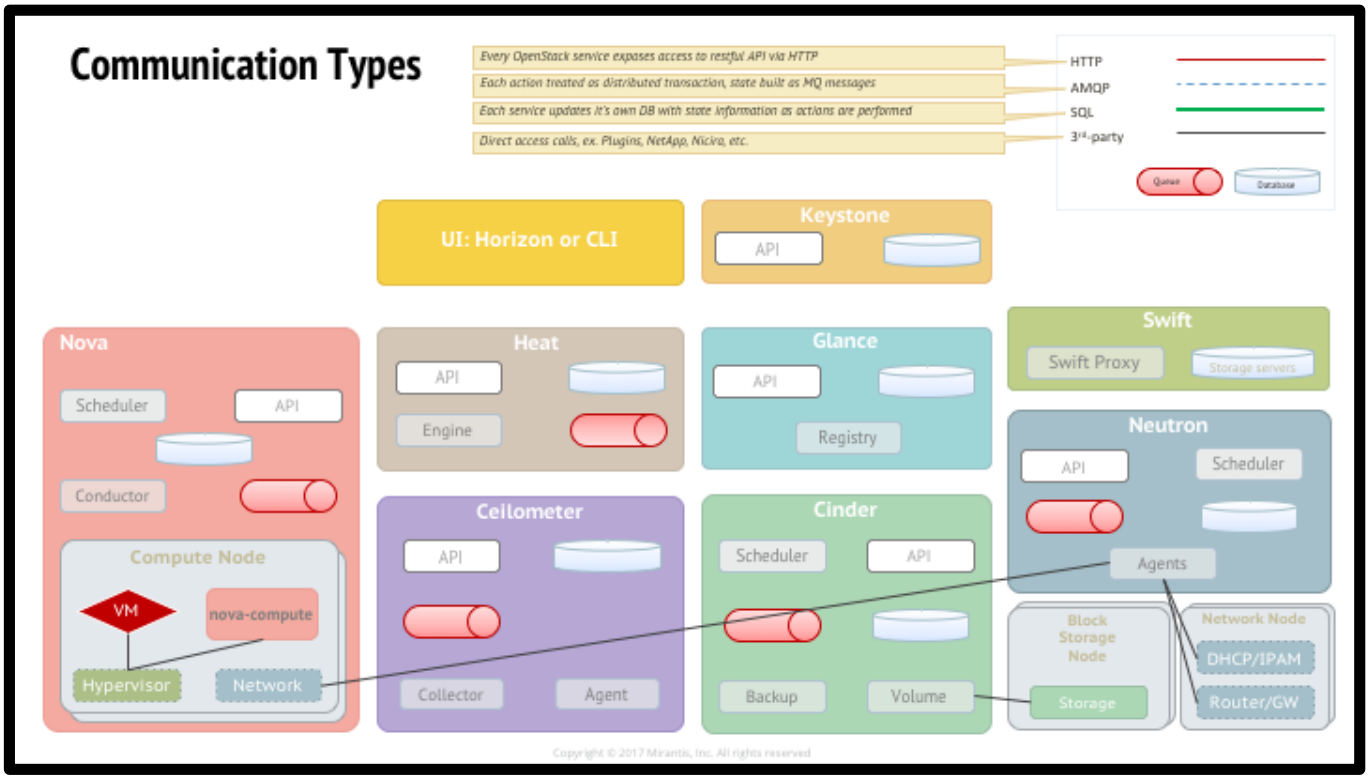
- All the subcomponents talk between each other
- Also Ceilometer uses it to get the data from Cinder and Neutron (components like Neutron and Cinder can publish the notification messages into the queue, Ceilometer Collectors are listening on those), this is not shown on this diagram. Also, there are different conversations to replace that mechanism with more flexible agents.



The HTTP connectors from Heat are drawn based on https://github.com/openstack/heat/blob/master/heat/engine/clients.py

For AMQP:

- All the subcomponents talk between each other
- Also Ceilometer uses it to get the data from Cinder and Neutron (components like Neutron and Cinder can publish the notification messages into the queue, Ceilometer Collectors are listening on those), this is not shown on this diagram. Also, there are different conversations to replace that mechanism with more flexible agents.



The HTTP connectors from Heat are drawn based on https://github.com/openstack/heat/blob/master/heat/engine/clients.py

For AMQP:

- All the subcomponents talk between each other
- Also Ceilometer uses it to get the data from Cinder and Neutron (components like Neutron and Cinder can publish the notification messages into the queue, Ceilometer Collectors are listening on those), this is not shown on this diagram. Also, there are different conversations to replace that mechanism with more flexible agents.

OpenStack REST API

- OpenStack public API is a RESTful API
  - REST stands for Representational State Transfer
  - REST is a stateless client/server protocol with a uniform interface for accessing the object model
  - OpenStack RESTful API is implemented using HTTP GET/PUT/POST/DELETE in combination with JSON for data
- Calling the API leads to calls to:
  - SQL calls to the database
  - AMQP messages on the message bus
  - 3rd-party protocols to other devices/applications

MIRANTIS

OpenStack Deployment

What Happens At Runtime

MIRANTIS

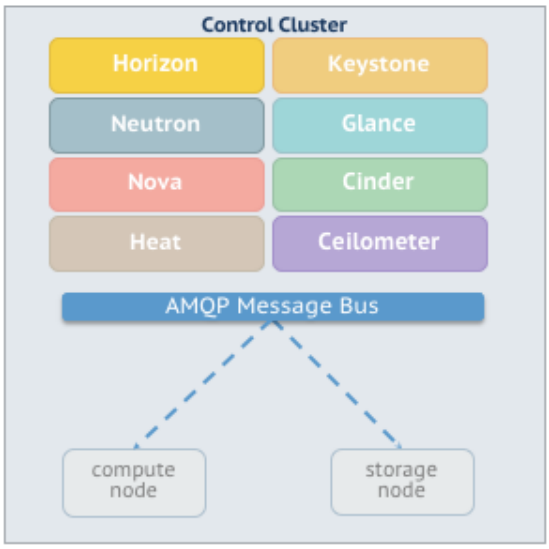
Deployment: Pick What You Want

- There is not one “OpenStack”
  - Different mix of programs and plugins leads to different products
- The components can be mixed & matched
  - A very basic OpenStack:
    - Nova
    - Keystone
    - Dashboard
    - Glance
  - Some choices are harder than others to change later

MIRANTIS

For example if you decided not use Cinder to begin with, you can add it later.
But if you decided to go with nova-network, switching to neutron is practically impossible

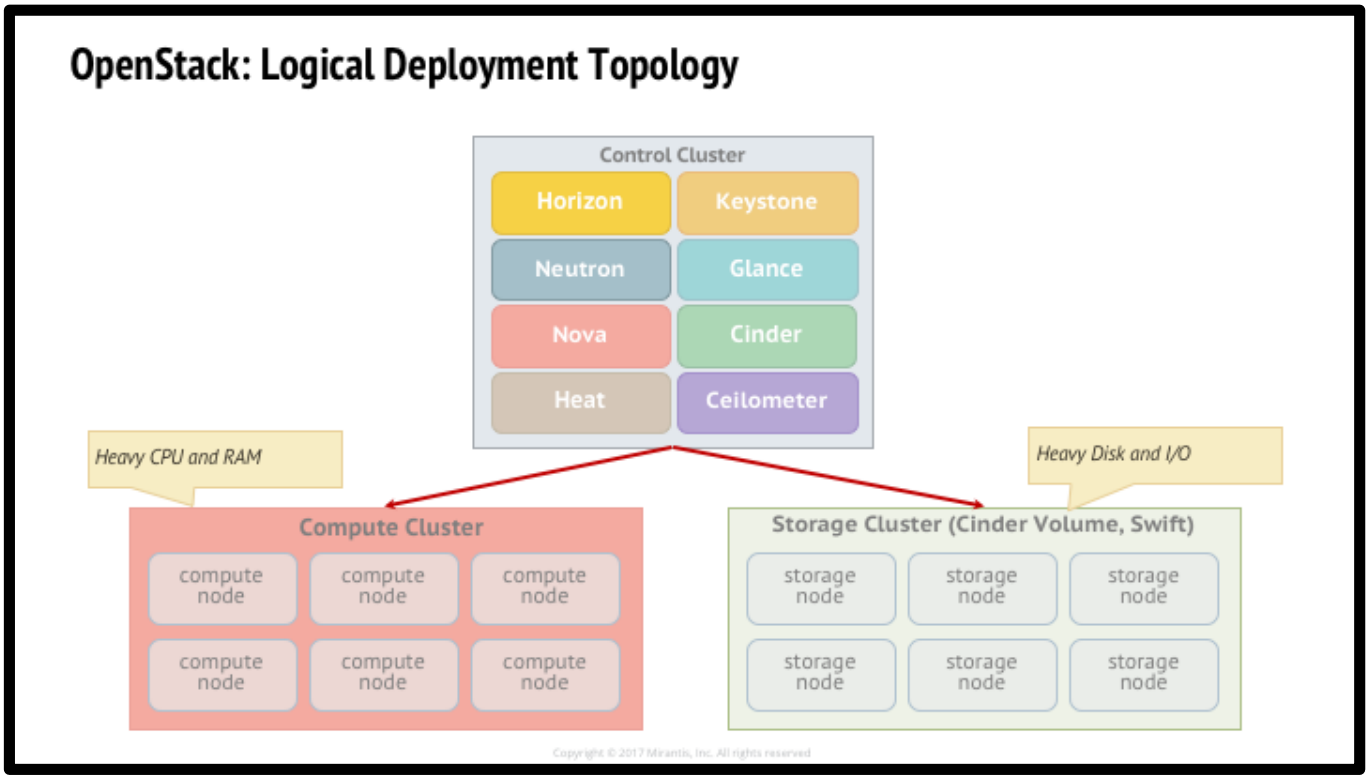
OpenStack: All-in-One Deployment



- Architecturally flexible enough to put everything on one node
  - This is not much of a cloud
  - Good for demos, or developers
- Typically installed using devstack or manual installation
  - Devstack is shell-script installation of OpenStack
  - http://devstack.org

MIRANTIS

All-in-One deployment of openstack is used for demo or development purposes only - but it also shows the architectural flexibility of the platform.

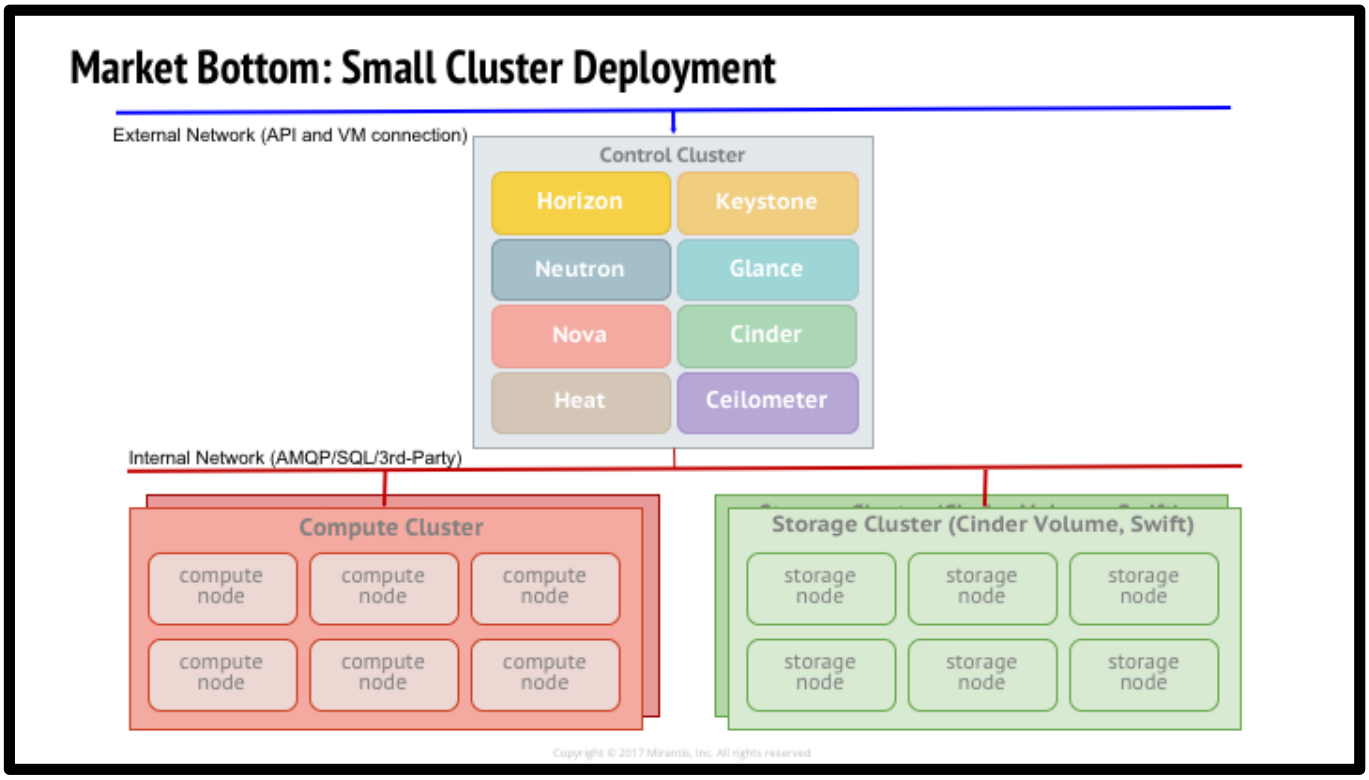


The motivation for separation is

1. The HW and resource demand is different for each service, i.e. lots of CPU and memory for compute nodes not needed on other services
2. You don't want to affect core services by move, adds, changes on compute or storage resources
3. Generally a many: one relationship between compute and storage vs. shared / platform services

- Think of it as 3 clusters within the OpenStack deployment topology
- Can be deployed on a single server or VM for lab purposes
- Production environments often segregate and customize services
- Separated for HW differences

Control Cluster is a brain

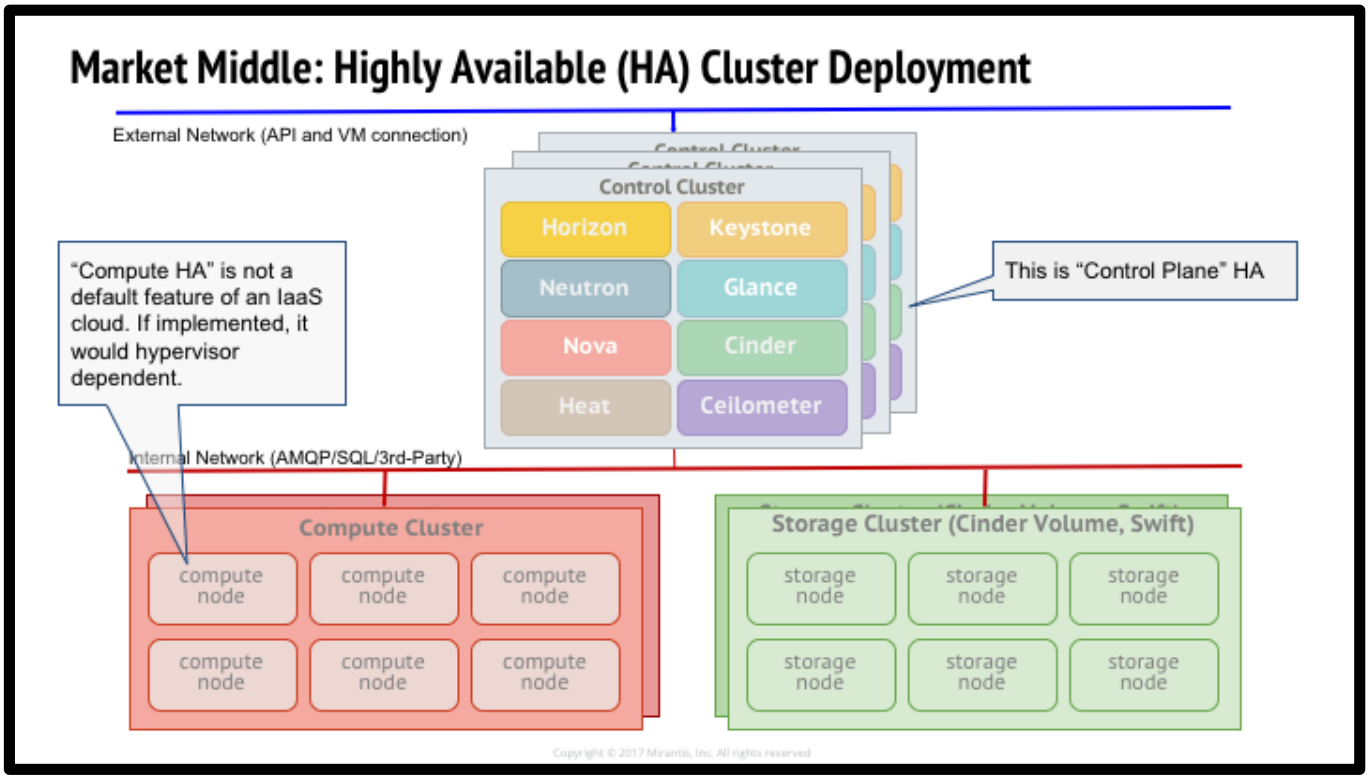


The motivation for separation is

1. The HW and resource demand is different for each service, i.e. lots of CPU and memory for compute nodes not needed on other services
2. You don't want to affect core services by move, adds, changes on compute or storage resources
3. Generally a many: one relationship between compute and storage vs. shared / platform services

- Think of it as 3 clusters within the OpenStack deployment topology
- Can be deployed on a single server or VM for lab purposes
- Production environments often segregate and customize services
- Separated for HW differences

Control Cluster is a brain

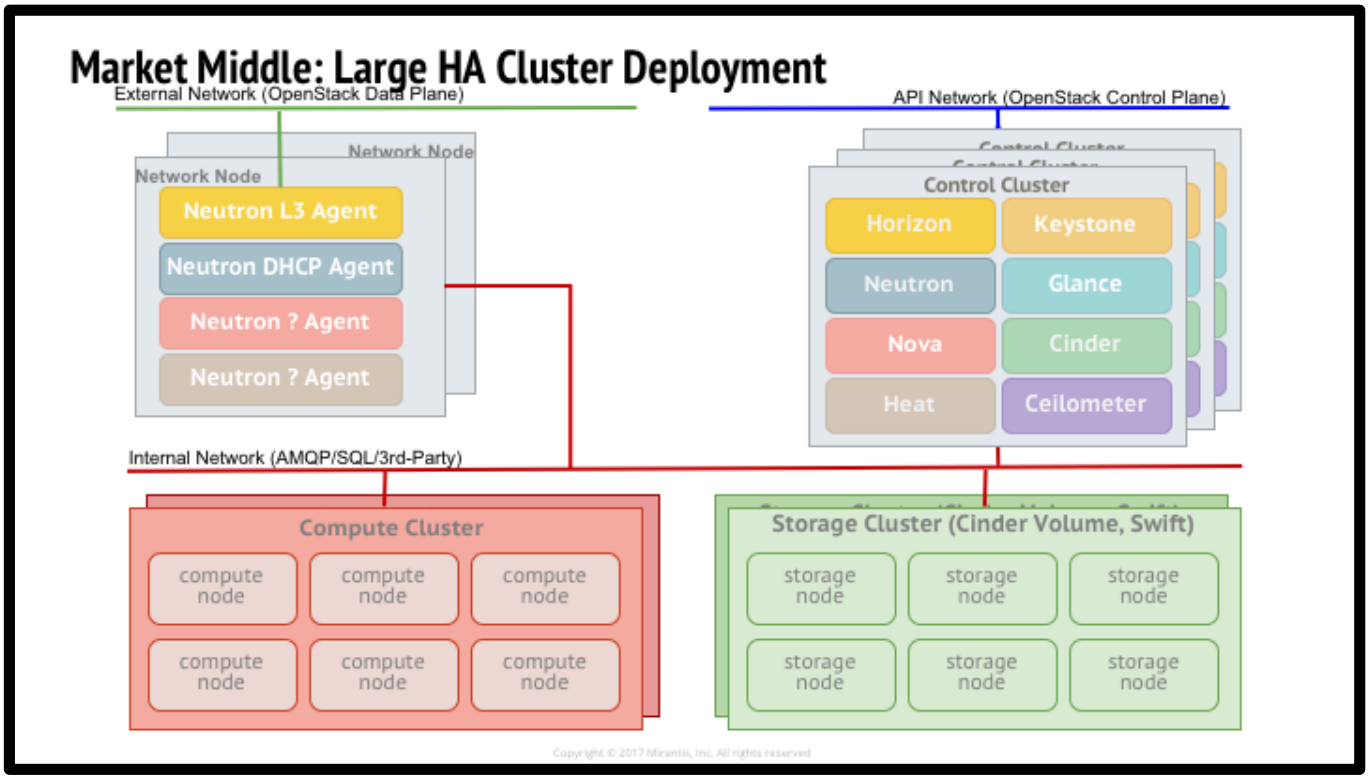


The motivation for separation is

1. The HW and resource demand is different for each service, i.e. lots of CPU and memory for compute nodes not needed on other services
2. You don't want to affect core services by move, adds, changes on compute or storage resources
3. Generally a many: one relationship between compute and storage vs. shared / platform services

- Think of it as 3 clusters within the OpenStack deployment topology
- Can be deployed on a single server or VM for lab purposes
- Production environments often segregate and customize services
- Separated for HW differences

Control Cluster is a brain



The motivation for separation is

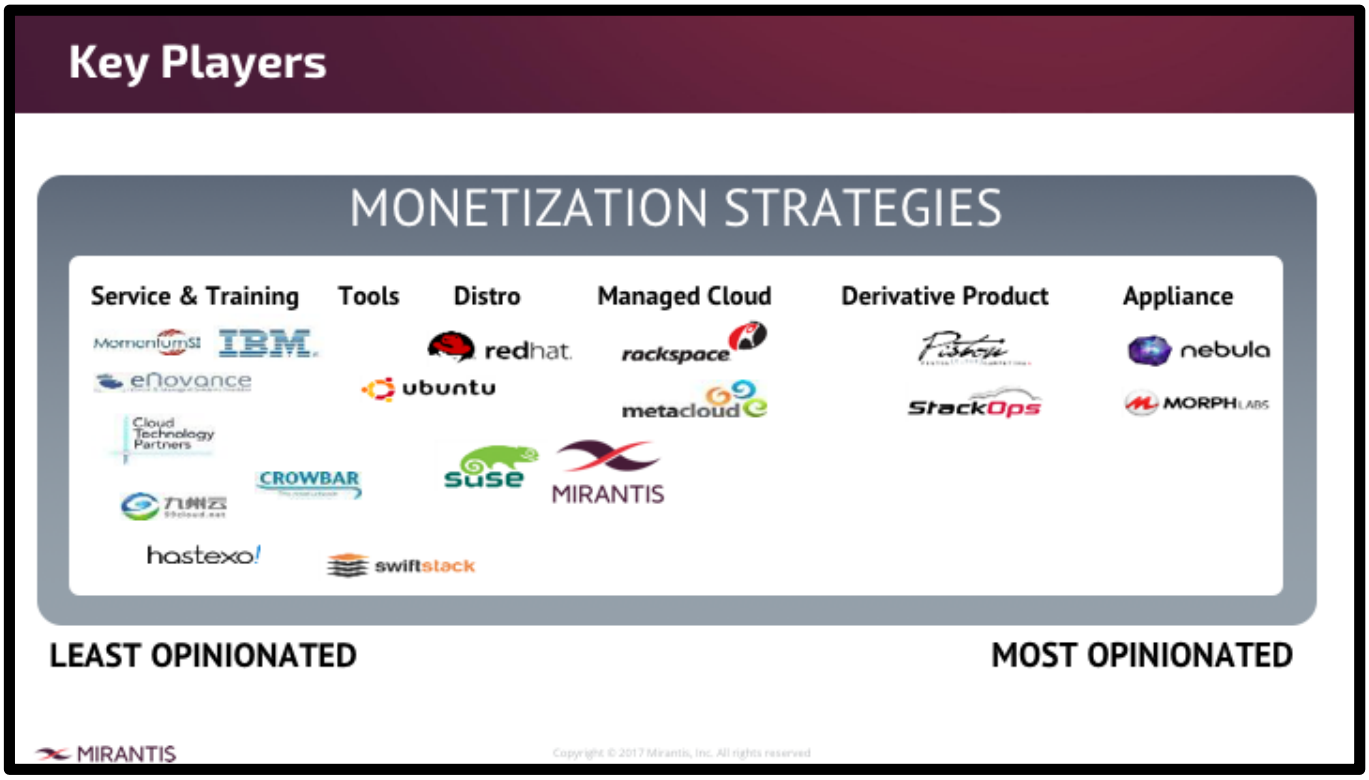
1. The HW and resource demand is different for each service, i.e. lots of CPU and memory for compute nodes not needed on other services
2. You don't want to affect core services by move, adds, changes on compute or storage resources
3. Generally a many: one relationship between compute and storage vs. shared / platform services

- Think of it as 3 clusters within the OpenStack deployment topology
- Can be deployed on a single server or VM for lab purposes
- Production environments often segregate and customize services
- Separated for HW differences

Control Cluster is a brain

OpenStack Ecosystem

MIRANTIS

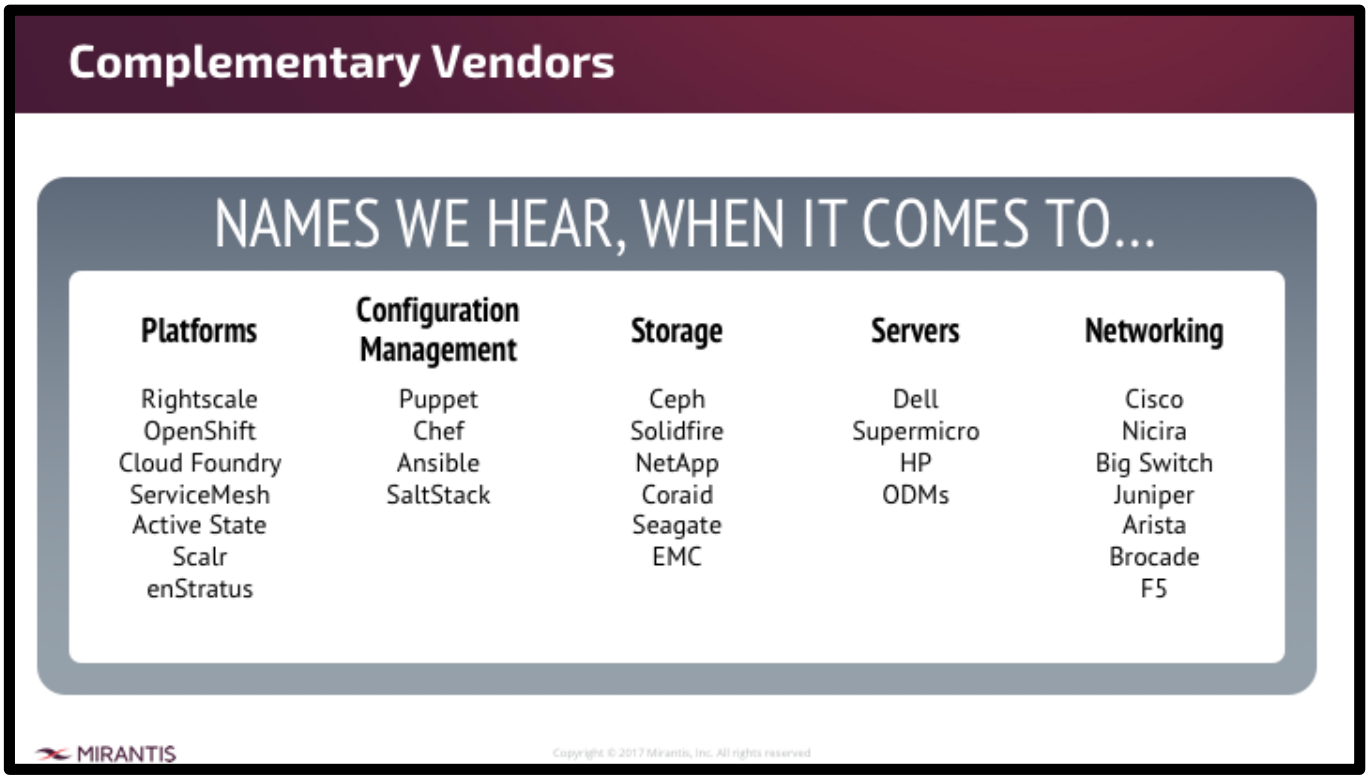


How to make money on Opentack?
There a lot of companies that provides service and training, consulting.

There are vendors providing tools, distributions. Same idea as with Linux distribution - you compile kernel on your own or take distribution. You can install openstack from source code but it is much easy to rely on distribution bundle, e.g. Mirantis Openstack with Fuel. *In 2016-2017, Mirantis started transforming its distro-centric product Fuel into ops-centric Mirantis Cloud Platform (MCP) moving from distro provider to managed cloud provider niche. Mirantis OpenStack, a a distro remains supported and publicly available.*

Managed Cloud providers. E.g. rackspace is the biggest openstack public cloud provider.

Derivative Products. E.g. storages based on swift without other components.
Appliance providers will provide with hardware and service, integrated racks with openstack preinstalled and preconfigured.



PaaS providers, storage backend providers, tools, networking backends, servers provide support to integrate with Openstack

OpenStack Overview: Recap

- OpenStack – open source software for building clouds
- OpenStack release cycle is every 6 months
- OpenStack is an umbrella over multiple independent programs (components)
- All OpenStack components talk RESTful API
- Most OpenStack components have dedicated DB (SQL) and MQ (QP), some talk to 3rd party components using their native APIs

MIRANTIS

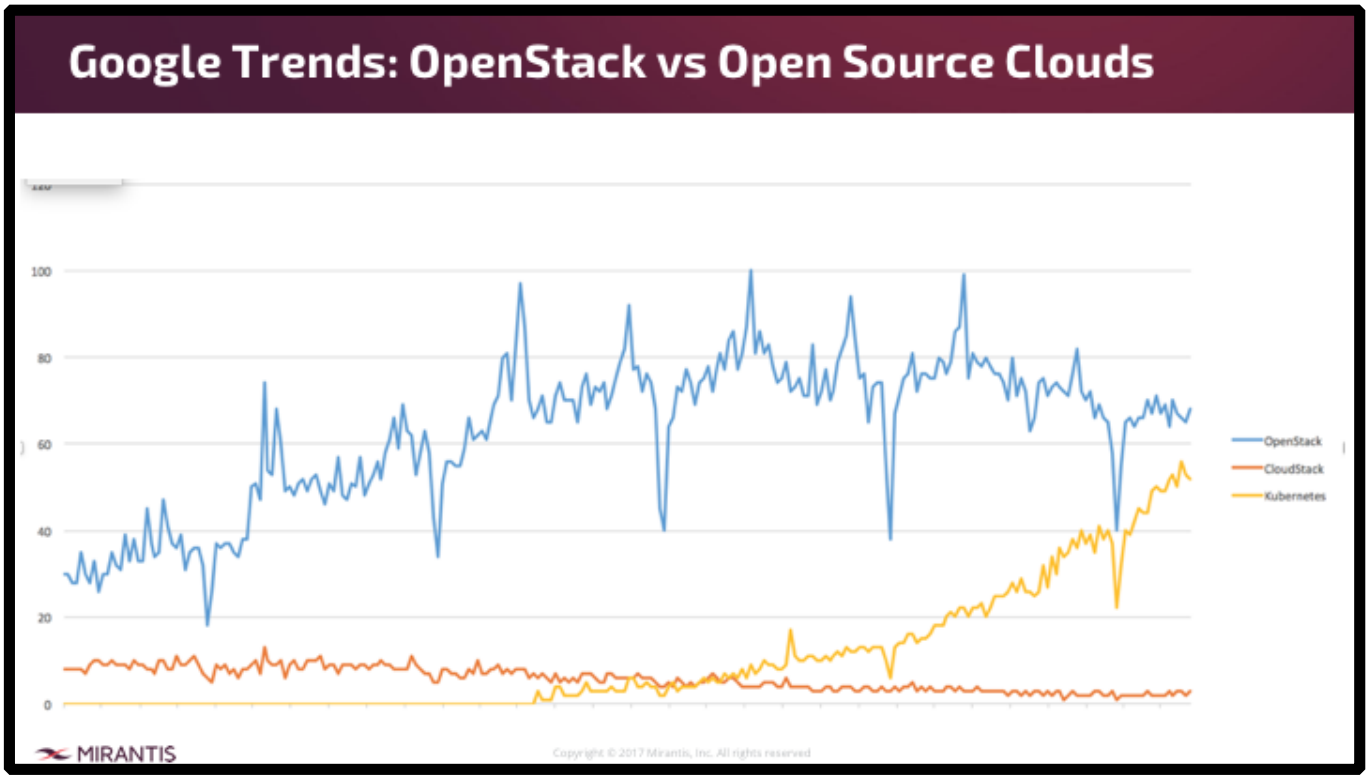
We have finished part one, and are about to begin part two, the flow during the provisioning of a VM.

This is a good place to stop and answer any questions you have about part one.

Let's make sure you are clear about the logical architecture before we start part two.

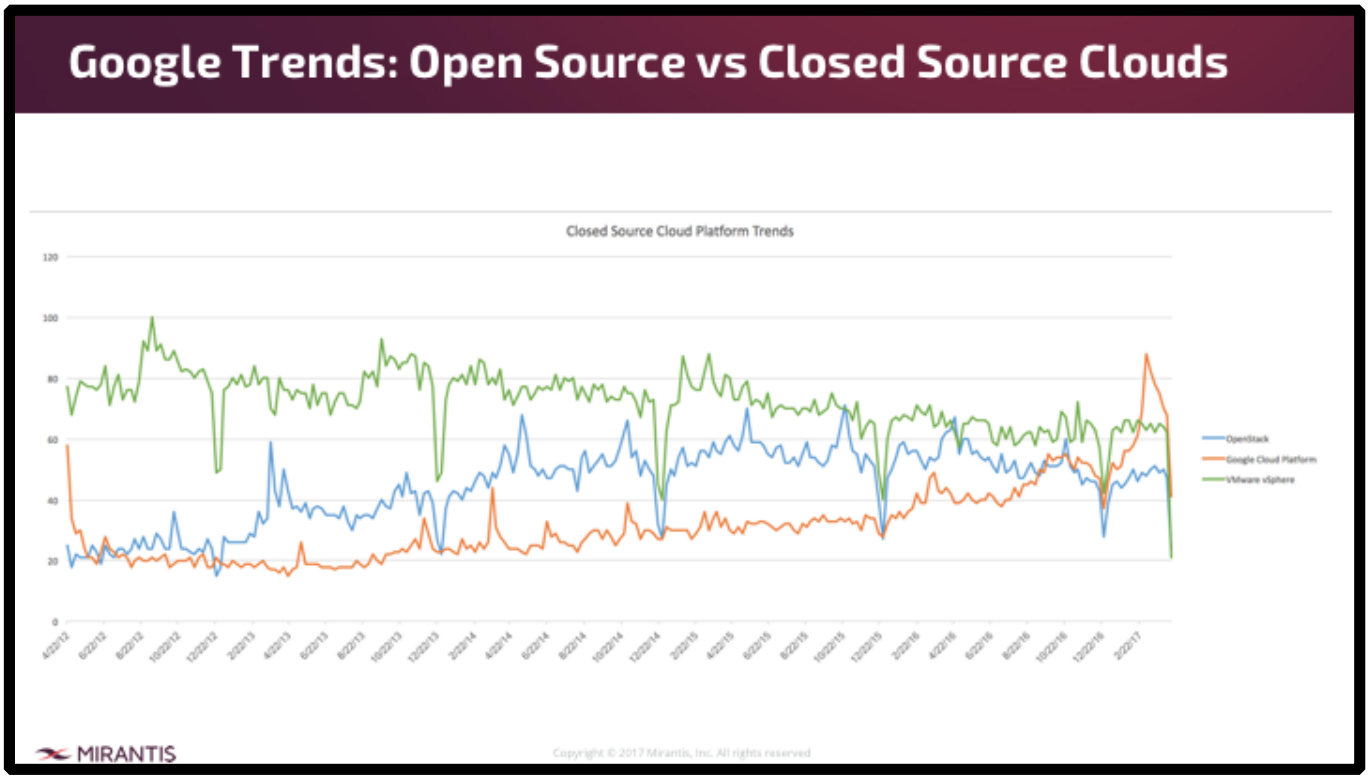
Reference

MIRANTIS



This picture demonstrates interest to Openstack comparing to other open source cloud software. As you can see, Openstack has been the top contender when it comes to Open Source Cloud Software for over the past 5 years. On a separate note, We have seen a steady decline in CloudStack due to Citrix pulling out of its backing while we see a much larger trend in Kubernetes since its inception.

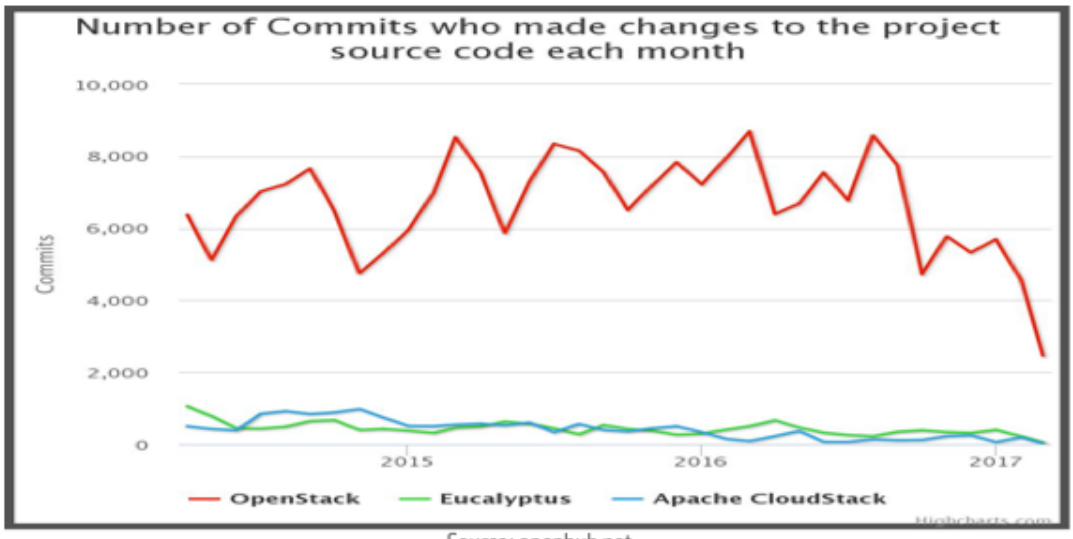
This information is based on a recent poll of Google Trends which can be pulled by anyone.



As you can see, Openstack again has had a large following when comparing it to proprietary software. More recently though , Google Cloud Platform has started it's climb to the top. This is due to the GA (Go Ahead) release of GCP in February of 2016. The giant spike in GCP was due to the recent partnership with the startup company Elastic to bring Elasticsearch and Analytics to GCP. Finally, you can see that VMWare's vSphere has been a leading platform in Closed Source Cloud Platforms because it is one of the oldest and well known platforms.

This information is based on a recent poll of Google Trends which can be pulled by anyone.

Commits per Month



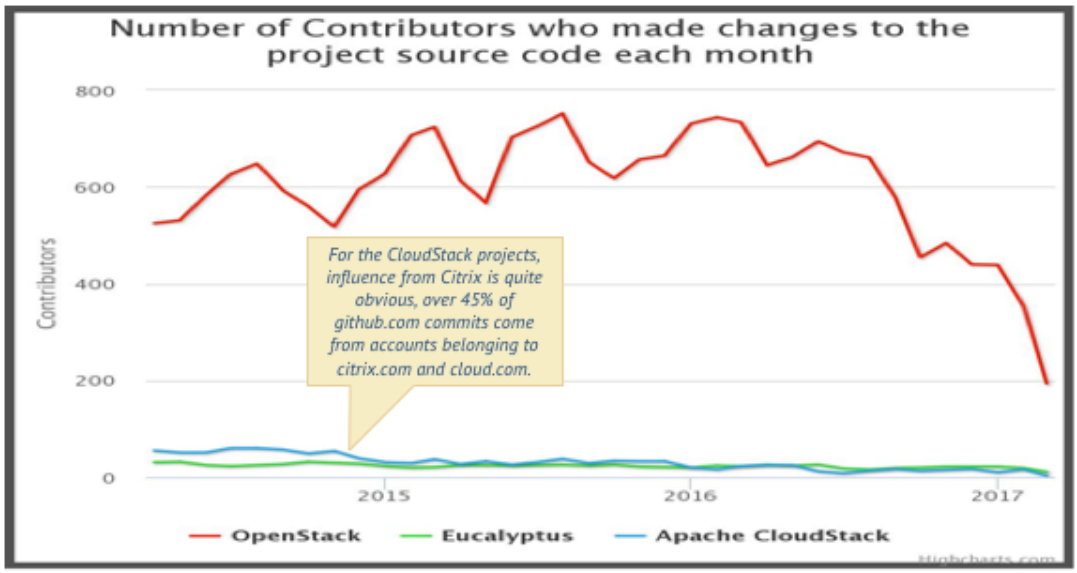
Source: openhub.net

MIRANTIS

How frequently people commit their code to Openstack. It obvious that Openstack is more alive than other open source clouds. You can see there apears to be a decline in Openstack more recently. The huge drop is due to this data being pulled in the middle of March of 2017. This means developers hadn't committed their code to show the proper statistics.

Data pulled from openhub.net

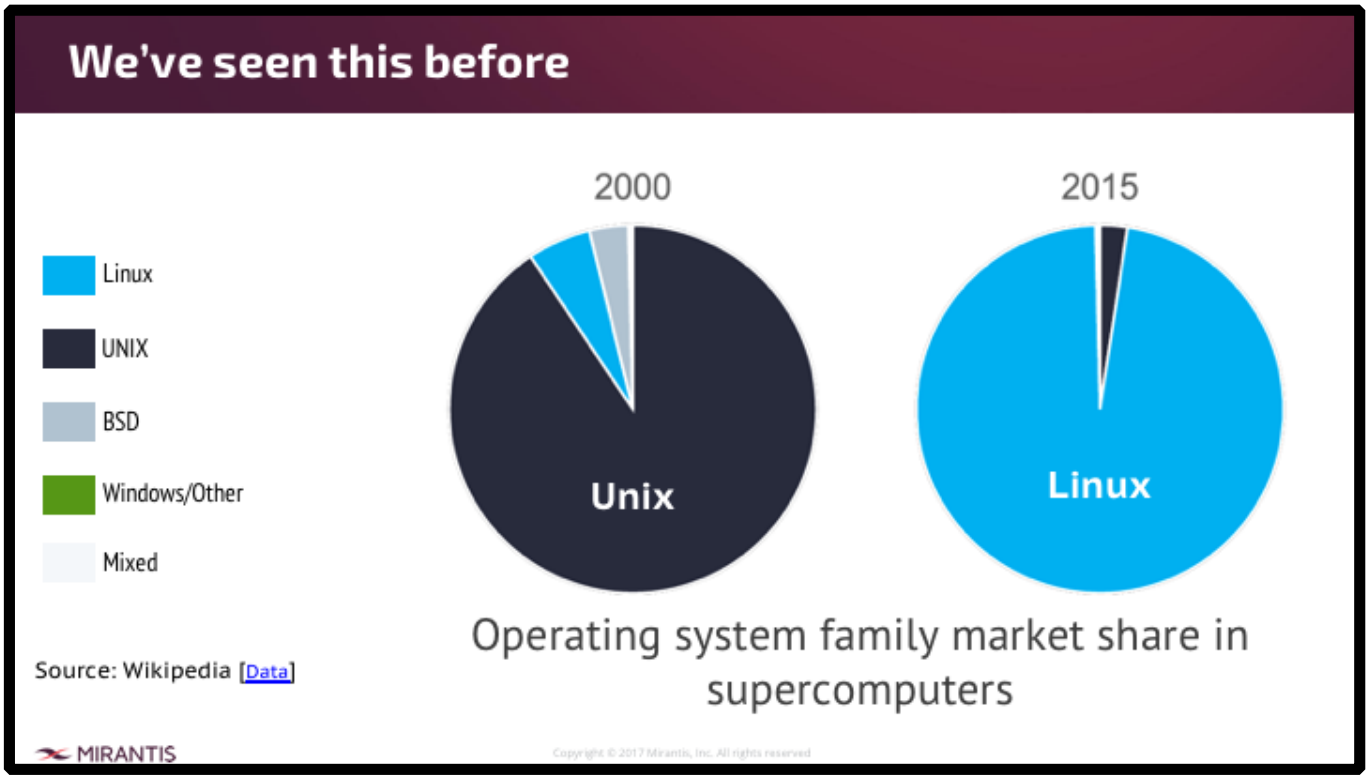
Contributor Diversity



MIRANTIS

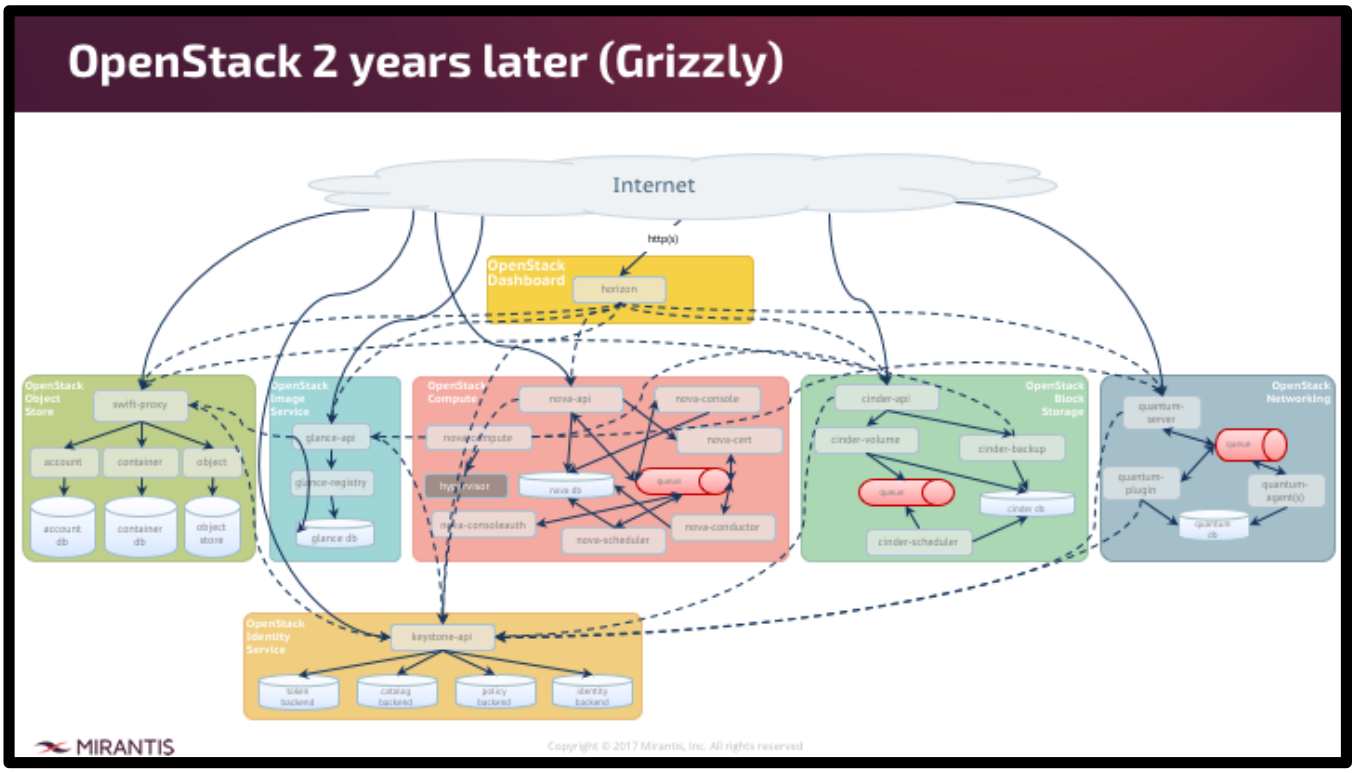
It is important that there is a huge diversity behind developers. They work for many different companies, there is no single company behind Openstack. The huge drop is due to this data being pulled in the middle of March of 2017. This means developers hadn't quite committed their code to show the proper statistics.

Data pulled from openhub.net



We believe that OpenStack will follow the same path of Linux and grow up to majority of market share in cloud-defined data centers.

https://commons.wikimedia.org/wiki/File:Operating_systems_used_on_top_500_supercomputers.svg



Two years later, when OpenStack architecture is finalized.

Original post:
http://www.solinea.com/2013/06/15/openstack-grizzly-architecture-revisited/