



# IaaS Cloud (OpenStack) overview

**Antonio Messina** <antonio.messina@s3it.uzh.ch>

S<sup>3</sup>IT - Services and Support for Science IT, University of Zurich

# What is OpenStack

*OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.*

[www.openstack.org](http://www.openstack.org)

# OpenStack vs. rest of the (virtualization) world

What's different from KVM/VMWare/Virtualbox?

- specs of the VMs are chosen from a list of predefined **flavors** that define:
  - Nr. of CPUs
  - amount of RAM
  - size disk
- complex network setup are possible
- OS already installed (but adapted automatically to the current instance)
- multiple options for storage (volumes and object storage)
- VMs are spawned on possibly thousands of nodes

yes, it's a *clone* of Amazon

# OpenStack vs. rest of the (virtualization) world

What's different from KVM/VMWare/Virtualbox?

- specs of the VMs are chosen from a list of predefined **flavors** that define:
  - Nr. of CPUs
  - amount of RAM
  - size disk
- complex network setup are possible
- OS already installed (but adapted automatically to the current instance)
- multiple options for storage (volumes and object storage)
- VMs are spawned on possibly thousands of nodes

yes, it's a *clone* of Amazon

# OpenStack vs. rest of the (virtualization) world

What's different from KVM/VMWare/Virtualbox?

- specs of the VMs are chosen from a list of predefined **flavors** that define:
  - Nr. of CPUs
  - amount of RAM
  - size disk
- complex network setup are possible
- OS already installed (but adapted automatically to the current instance)
- multiple options for storage (volumes and object storage)
- VMs are spawned on possibly thousands of nodes

yes, it's a *clone* of Amazon

# OpenStack vs. rest of the (virtualization) world

What's different from KVM/VMWare/Virtualbox?

- specs of the VMs are chosen from a list of predefined **flavors** that define:
  - Nr. of CPUs
  - amount of RAM
  - size disk
- complex network setup are possible
- OS already installed (but adapted automatically to the current instance)
- multiple options for storage (volumes and object storage)
- VMs are spawned on possibly thousands of nodes

yes, it's a *clone* of Amazon

# OpenStack vs. rest of the (virtualization) world

What's different from KVM/VMWare/Virtualbox?

- specs of the VMs are chosen from a list of predefined **flavors** that define:
  - Nr. of CPUs
  - amount of RAM
  - size disk
- complex network setup are possible
- OS already installed (but adapted automatically to the current instance)
- multiple options for storage (volumes and object storage)
- VMs are spawned on possibly thousands of nodes

yes, it's a *clone* of Amazon

# OpenStack vs. rest of the (virtualization) world

What's different from KVM/VMWare/Virtualbox?

- specs of the VMs are chosen from a list of predefined **flavors** that define:
  - Nr. of CPUs
  - amount of RAM
  - size disk
- complex network setup are possible
- OS already installed (but adapted automatically to the current instance)
- multiple options for storage (volumes and object storage)
- VMs are spawned on possibly thousands of nodes

yes, it's a *clone* of Amazon



# OpenStack vs. rest of the (virtualization) world

What's different from KVM/VMWare/Virtualbox?

- specs of the VMs are chosen from a list of predefined **flavors** that define:
  - Nr. of CPUs
  - amount of RAM
  - size disk
- complex network setup are possible
- OS already installed (but adapted automatically to the current instance)
- multiple options for storage (volumes and object storage)
- VMs are spawned on possibly thousands of nodes

yes, it's a *clone* of Amazon

## Above all

creation is done via Web GUI, CLI or **network APIs**  
⇒ it can be *scripted*

You just have to setup the cloud.

Actual provisioning of the VMs can be delegated **to the user**

## Above all

creation is done via Web GUI, CLI or **network APIs**  
⇒ it can be *scripted*

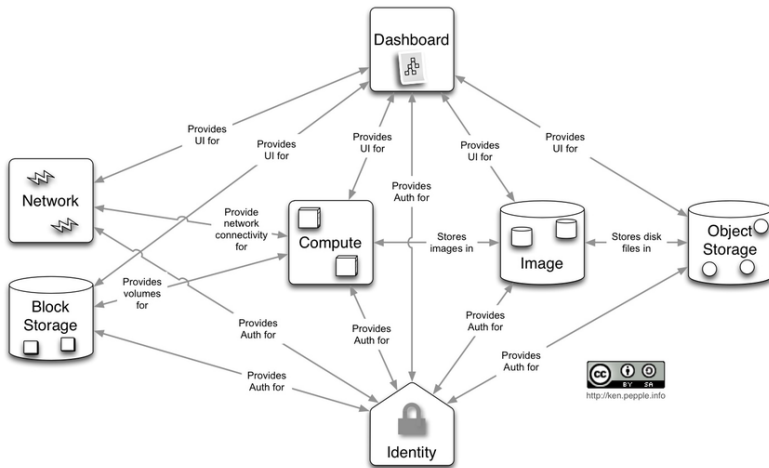
You just have to setup the cloud.

Actual provisioning of the VMs can be delegated **to the user**

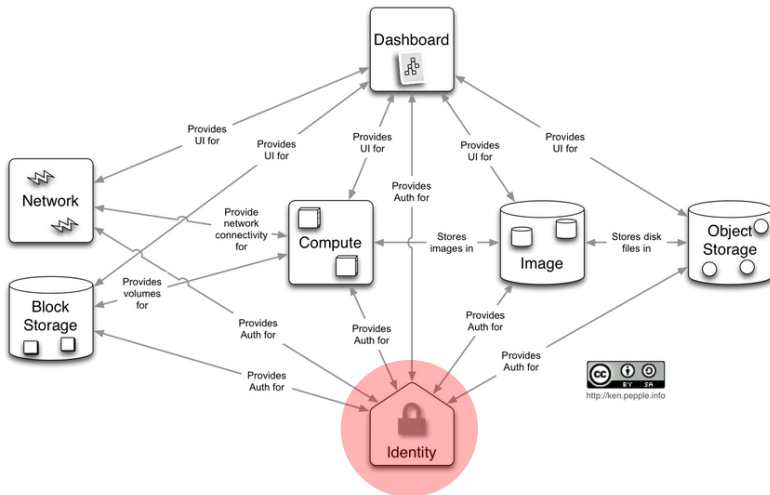
# OpenStack Architecture

- written in Python (plus auxiliary shell scripts)
- built around **independent components**
- **highly distributed** architecture
  - designed for very big installations
- **intrinsic HA** of *most* OpenStack services (MySQL and RabbitMQ have to be properly configured)
- **\*SQL** database used to store persistent data
- **RabbitMQ** used for RPC and notification
- **RESTful APIs** for all the services

# OpenStack logical view

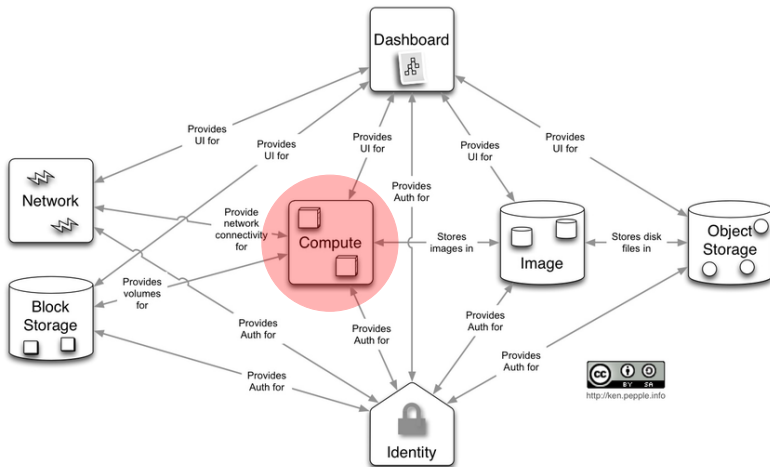


# OpenStack logical view



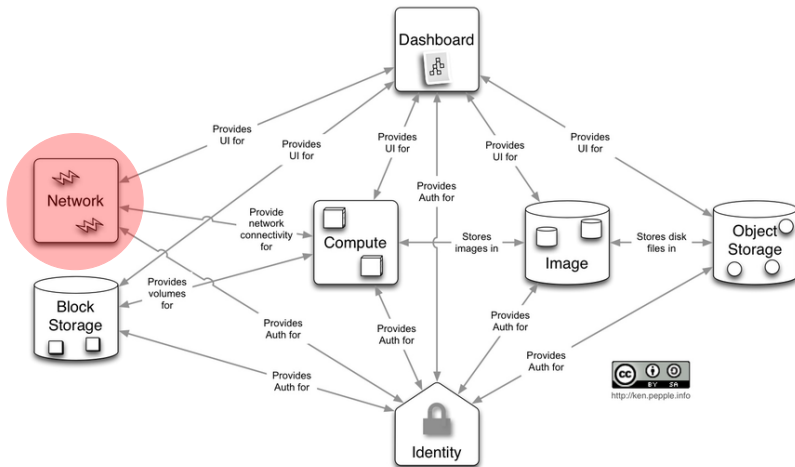
**Keystone** provides the authentication service

# OpenStack logical view



**Nova** provides computational services

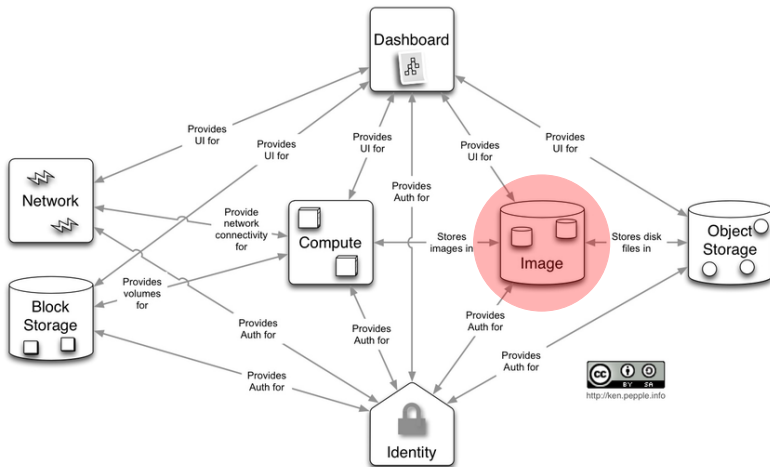
# OpenStack logical view



**Neutron** provides network services

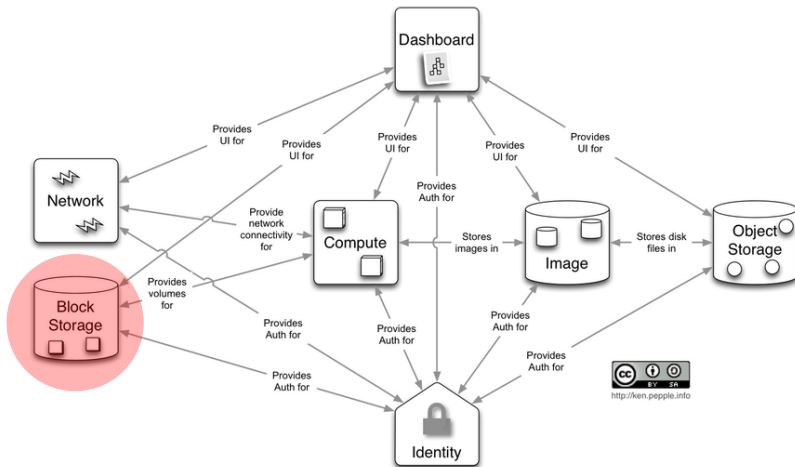


# OpenStack logical view



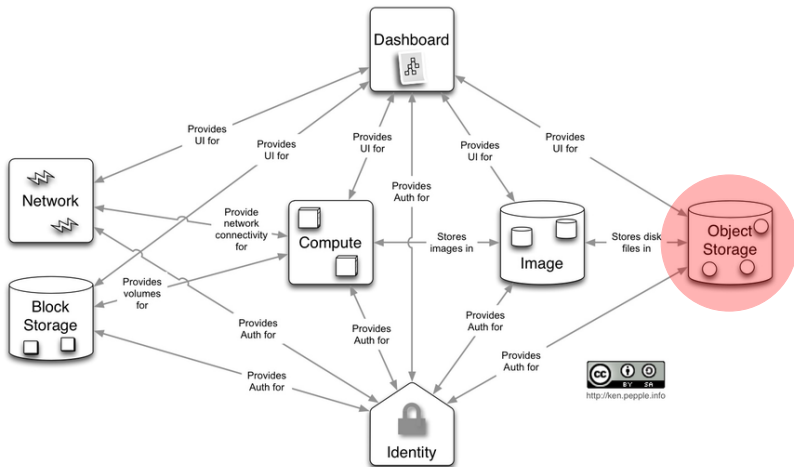
**Glance** provides image store

# OpenStack logical view



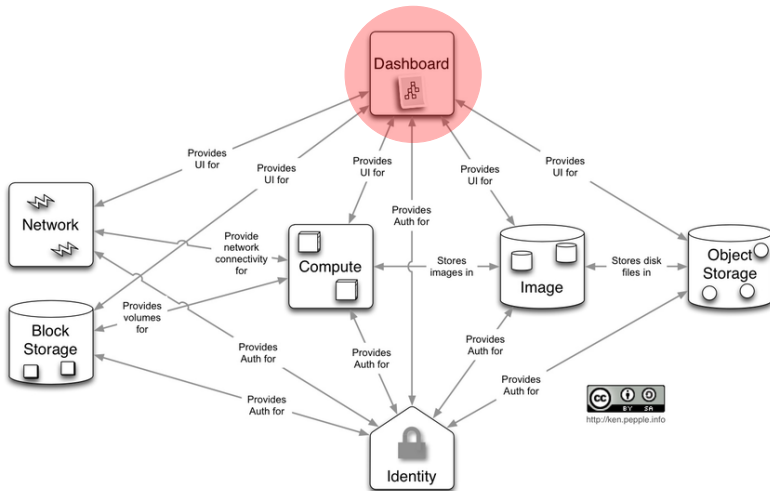
**Cinder** provides block persistent store

# OpenStack logical view



**Swift** provides object persistent store

# OpenStack logical view



**Horizon** provides web user interface

## keystone - authentication service

- It's the **entry point** for OpenStack API.
- Stores authentication information (*users, passwords, tokens, projects, roles*)
- Holds a catalog of available services and their endpoints.
- Can use different backends (SQL database, LDAP)

## nova - compute service



Service responsible of managing virtual instances.

**nova-api** Web API frontend, accepts requests, validates them and contact other services if needed.

**nova-scheduler** decides where to start an instance

**nova-compute** running on each compute node, interacts with the hypervisor and actually starts the vm.

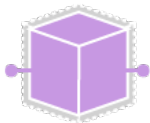
## glance - image service



Service responsible of storing image informations and, optionally, image files.

- Holds information about available images.
- Optionally allow to download and upload images.
- Images can be stored on **different backends** (RDB, S3, Swift, filesystem)

## neutron - network service



Service responsible of creating and managing networks. It is supposed to replace.

Still not widely used, but very feature rich.

- L2 and L3 networks.
- Allow creation of multiple networks and subnets.
- Plugin architecture.
- Supports advanced network services (Load Balancer, Firewall, DNS as a service)
- Integrates with network devices (Cisco, Brocade. . .)



## cinder - block storage



- Creates and export volumes via iSCSI to the compute node.
- Volumes are mounted **transparently** from the virtual machines.
- Supports **multiple storage backends** (NFS, LVM, Ceph, GlusterFS but also SAN/NAS devices from IBM, NetApp etc. . . )

composed of **multiple services**:

**cinder-api** Web API frontend.

**cinder-volume** Manages block storage devices. You can have many of these.

**cinder-scheduler** Decides which cinder-volume has to provide the volume for an instance.

## swift - object storage



Object storage distributed service.

- Redundant, scalable object storage on commodity hardware.
- Not a POSIX filesystem.
- Scales horizontally simply by adding new servers.

It's not the only choice: **Ceph**, **GlusterFS** and others can be used instead.

## Life of a virtual machine

1. Authentication is performed either by the web interface **horizon** or **nova** command line tool:
2. **nova-api** is contacted and a new request is created:
3. **nova-scheduler** find an appropriate host
4. **nova-compute** reads the request and start an instance:
5. (if requested) **nova-compute** contacts **cinder** to provision the volume
6. **neutron** configure the network
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

## Life of a virtual machine

1. Authentication is performed either by the web interface **horizon** or **nova** command line tool:
  - 1.1 keystone is contacted and authentication is performed
  - 1.2 a **token** is saved in the database and returned to the client to be used with later interactions with OpenStack services for this request.
2. **nova-api** is contacted and a new request is created:
3. **nova-scheduler** find an appropriate host
4. **nova-compute** reads the request and start an instance:
5. (if requested) **nova-compute** contacts **cinder** to provision the volume
6. **neutron** configure the network
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

## Life of a virtual machine

1. Authentication is performed either by the web interface **horizon** or **nova** command line tool:
2. **nova-api** is contacted and a new request is created:
  - 2.1 checks via **keystone** the validity of the token
  - 2.2 checks the authorization of the user
  - 2.3 validates parameters and create a new request in the database
  - 2.4 calls the scheduler via queue
3. **nova-scheduler** find an appropriate host
4. **nova-compute** reads the request and start an instance:
5. (if requested) **nova-compute** contacts **cinder** to provision the volume
6. **neutron** configure the network
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

## Life of a virtual machine

1. Authentication is performed either by the web interface **horizon** or **nova** command line tool:
2. **nova-api** is contacted and a new request is created:
3. **nova-scheduler** find an appropriate host
  - 3.1 reads the request
  - 3.2 find an appropriate host via filtering and weighting
  - 3.3 calls the chosen **nova-compute** host via queue
4. **nova-compute** reads the request and start an instance:
5. (if requested) **nova-compute** contacts **cinder** to provision the volume
6. **neutron** configure the network
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

## Life of a virtual machine

1. Authentication is performed either by the web interface **horizon** or **nova** command line tool:
2. **nova-api** is contacted and a new request is created:
3. **nova-scheduler** find an appropriate host
4. **nova-compute** reads the request and start an instance :
  - 4.1 generates a proper configuration for the hypervisor
  - 4.2 get image URI via image id
  - 4.3 download the image
  - 4.4 request to allocate network via queue
5. (if requested) **nova-compute** contacts **cinder** to provision the volume
6. **neutron** configure the network
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

## Life of a virtual machine

1. Authentication is performed either by the web interface **horizon** or **nova** command line tool:
2. **nova-api** is contacted and a new request is created:
3. **nova-scheduler** find an appropriate host
4. **nova-compute** reads the request and start an instance:
5. **nova-compute** contacts **cinder** to provision the volume(if requested) **nova-compute** contacts **cinder** to provision the volume
  - 5.1 gets connection parameters from cinder
  - 5.2 uses iscsi to make the volume available on the local machine
  - 5.3 asks the hypervisor to provision the local volume as virtual volume of the specified virtual machine
6. **neutron** configure the network
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.



## Life of a virtual machine

1. Authentication is performed either by the web interface **horizon** or **nova** command line tool:
2. **nova-api** is contacted and a new request is created:
3. **nova-scheduler** find an appropriate host
4. **nova-compute** reads the request and start an instance:
5. (if requested) **nova-compute** contacts **cinder** to provision the volume
6. **neutron** configure the network
  - 6.1 allocates a valid private ip
  - 6.2 if requested, it allocates a floating ip
  - 6.3 configures the host as needed (dnsmasq, iptables, Open VSwitch. . .)
  - 6.4 updates the request status
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

## Life of a virtual machine

1. Authentication is performed either by the web interface **horizon** or **nova** command line tool:
2. **nova-api** is contacted and a new request is created:
3. **nova-scheduler** find an appropriate host
4. **nova-compute** reads the request and start an instance:
5. (if requested) **nova-compute** contacts **cinder** to provision the volume
6. **neutron** configure the network
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

## Life of a virtual machine

1. Authentication is performed either by the web interface **horizon** or **nova** command line tool:
2. **nova-api** is contacted and a new request is created:
3. **nova-scheduler** find an appropriate host
4. **nova-compute** reads the request and start an instance:
5. (if requested) **nova-compute** contacts **cinder** to provision the volume
6. **neutron** configure the network
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

## Notes on installation

- Please, please, please, use a **deployment and configuration manager**. There are many: **Puppet**, **Chef**, **CFEngine**, **Ansible**, **SaltStack**. . . Just pick the one you like most.
- Do not underestimate the **complexity** of the system.
- Plan in advance, and **plan for failures**.
- RTFM: the OpenStack website is now plenty of documentation<sup>1</sup>
  - **Install Guide (for Ubuntu 12.04/14.04)**
  - **Architecture Design Guide**
  - **Cloud Administrator Guide**
  - **Training guide**
  - **Operations Guide**
  - **High Availability Guide**
  - **Security Guide**

---

<sup>1</sup>it wasn't like this 2 years ago. . .

- [illegible]

## 2 September 2014

## Other OpenStack services

Projects **integrated** in Icehouse:

- **Ceilometer** (Metering)
- **Heat** (Orchestration)
- **Trove** (Database as a service)
- **Sahara** (Data Processing - Hadoop)

Projects in **incubation**:

- **IroniC** (Bare metal provisioning)
- **Zaqar (aka Marconi)** (Messaging service)
- **Barbican** (Secure storage of secrets)
- **Designate** (DNSaaS)
- **TripleO** (OpenStack-on-OpenStack)