Cloud Computing

Chapter 3 : Cloud Provider Platforms

By : Prof. Lamri SAYAD

2025

Content

- Introduction
- Commercial Platforms
  - AWS
  - Microsoft Azure
  - Google Cloud GCP
- Opensource platforms
  - OpenStack
  - OpenNebula

Introduction

Introduction

What Is a Cloud Provider Platform?

A **cloud platform** is:

- **A technical environment and infrastructure** offered by a cloud service provider (such as AWS, Microsoft Azure, Google Cloud Platform, or IBM Cloud) that enables users to **deploy, manage, and scale computing resources and services** over the internet.
- It offers services through **well-defined layers of abstraction**, known as **service models**.
  - **IaaS** : Provider offers virtualized infrastructure; user manages OS & apps (like AWS EC2, Azure VM, Google Compute Engine)
  - **PaaS** : Provider manages infrastructure; user deploys applications (Google App Engine, Azure App Service)
  - **SaaS** : User uses software; provider manages everything (Gmail, Microsoft 365, Salesforce)

Each provider platform combines:

- **Physical infrastructure** (data centers and hardware)
- **Virtualized resources** (compute, storage, network)
- **Cloud services** (databases, analytics, AI, IoT, etc.)
- **Management and security tools** (monitoring, billing, IAM)

Introduction
Existing cloud infrastructure

- The cloud computing infrastructure at Amazon, Google, and Microsoft (as of mid 2012).
    - Amazon is a pioneer in Infrastructure-as-a-Service (IaaS).
    - Google's efforts are focused on Software-as-a-Service (SaaS) and Platform-as-a-Service (PaaS).
    - Microsoft is involved in PaaS.
- Private clouds are an alternative to public clouds. Open-source cloud computing platforms such as:
    - Eucalyptus,
    - OpenNebula,
    - Nimbus,
    - OpenStack
- Can be used as a control infrastructure for a private cloud.

Reminder : Infrastructure as a service

- Users can ask for
  - Computing nodes (servers)
    - Elastic Compute Cloud (EC2)
  - Storage
    - Simple Storage Services (S3)
    - Elastic Block Stores (EBS)
    - Glaciar
    - DynamoDB
  - Networking (between nodes)

Reminder : Platform as a service

- Provides a runtime environment and development tools for building, testing, and deploying applications without managing underlying infrastructure.
- Use cases :
  - Web app development.
  - API hosting.
  - Rapid software deployment pipelines.
- Examples :
  - Google App Engine,
  - AWS Elastic Beanstalk,
  - Azure App Service.

Reminder : Software as a service

- Cloud-based software delivered via the internet, accessible through a web interface or API, fully managed by the provider.
- Use cases :
  - Email,
  - Collaboration tools,
  - CRM, ERP.
- Examples :
  - Gmail,
  - Microsoft 365,
  - Salesforce,
  - Dropbox.

Amazon Web Services (AWS)

Amazon Web Services (AWS)

- **Amazon Web Services (AWS)** :
  - is a **comprehensive cloud computing platform** offered by **Amazon**,
  - Offers **a complete environment** — infrastructure, development tools, APIs, and management systems — to build and run applications in the cloud.
  - Provides **on-demand access** to computing, storage, networking, databases, AI/ML tools, and other IT resources through the internet
  - uses a **pay-as-you-go** pricing model.

- **Launched:** March 2006

- **Operated by:** Amazon.com, Inc.

Amazon Web Services (AWS)
Historical background

Year	Milestone
2002	AWS announced as internal Amazon web services platform
2006	Official launch: EC2 (compute) and S3 (storage) released
2012	Introduction of AWS Marketplace
2015	Amazon becomes top cloud provider globally
2020+	Expansion into AI, IoT, and serverless computing (e.g., Lambda, SageMaker, Greengrass)

AWS Global Infrastructure

- Amazon offers cloud services through a network of data centers on several continents.
- AWS global infrastructure is organized hierarchically into **Regions**, **Availability Zones**, and **Edge Locations**.
- **Region**
  - A **geographical area** that contains multiple data centers (Availability Zones).
  - Each Region is isolated for fault tolerance and compliance.
- **Availability Zone (AZ)**
  - Each Region has **2 or more Availability Zones** — physically separate data centers connected by high-speed, low-latency links.
- **Edge Location**
  - Smaller data centers located close to users for **content delivery and caching**.

- The AWS Cloud spans 120 Availability Zones within 38 Geographic Regions.
- Regions do not share resources and communicate through the Internet.

Region	Location	Availability zones	Cost
US West	Oregon	us-west-2a/2b/2c	Low
US West	North California	us-west-1a/1b/1c	High
US East	North Virginia	us-east-1a/2a/3a/4a	Low
Europe	Ireland	eu-west-1a/1b/1c	Medium
South America	Sao Paulo, Brazil	sa-east-1a/1b	Very high
Asia Pacific	Tokyo, Japan	ap-northeast-1a/1b	High
Asia Pacific	Singapore	ap-southeast-1a/1b	Medium


US West
(Oregon)
US West
(N. California)
US East
(N. Virginia)
EU
(Ireland)
Asia Pacific
(Tokyo)
Asia Pacific
(Singapore)
South America
(San Paulo)
Low cost region
Medium cost region
High cost region
Very high cost region

AWS Global Infrastructure
Choosing the right region

- Selection depends on:
  - Proximity to users (lower latency)
  - Service availability (not all regions have every service)
  - Cost differences
  - Regulatory compliance

- Example: Hosting an EU e-commerce site → choose eu-west-3 (Paris) for compliance and low latency.

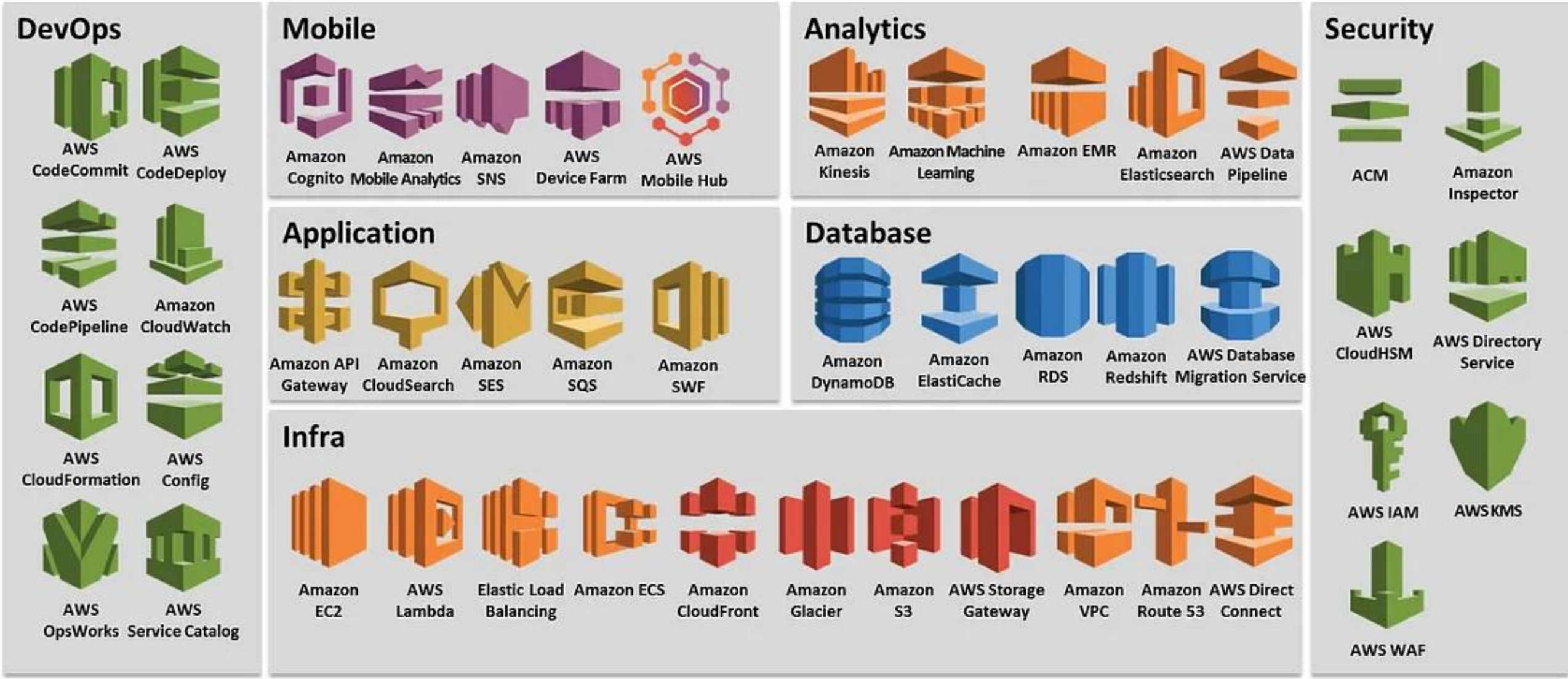
Accessing AWS Services

- The AWS Management Console : A **web-based graphical interface** that allows users to interact with AWS services.
  - **URL:** https://aws.amazon.com/console/
  - **Access:** Login using AWS account credentials or IAM user credentials.
- AWS Command Line Interface (CLI) : A **text-based interface** that allows users to interact with AWS services using terminal commands.
  - **Installation**: **awscli** tool (available for Windows, macOS, Linux)
  - Examples :
    - aws s3 ls
    - aws ec2 start-instances --instance-ids i-1234567890abcdef
- AWS Software Development Kits (SDKs) : Libraries that allow developers to access AWS services **programmatically** from apps.
  - **Available SDKs: Python (boto3), Java, JavaScript, C#, Go, PHP, Ruby, etc.**
- AWS APIs (Application Programming Interfaces) : RESTful **HTTP-based APIs** that expose AWS services directly over the web.
  - Every SDK and CLI command ultimately calls these APIs.

Accessing AWS Services

Access Method	Type	Best For	Example Tool / URL
AWS Management Console	GUI (Web)	Beginners / Admin tasks	console.aws.amazon.com
AWS CLI	Command-line	DevOps / Automation	awscli
AWS SDKs	Programming	Developers / Integration	boto3, AWS SDK for Java
AWS APIs	HTTP REST	Custom Integrations	https://ec2.amazonaws.com
AWS CloudShell / Tools	Specialized	Quick access / scripting	CloudShell, Mobile App

AWS Core Services



AWS IaaS services

Service	Description
EC2	Virtual servers in the cloud (compute)
EBS	Block storage for EC2 instances
S3	Object storage (storage service)
VPC	Virtual private cloud, networking control
Elastic IP	Static public IP addresses
Route 53	DNS and domain registration
Direct Connect	Dedicated network connections to AWS
Elastic Load Balancing (ELB)	Distributes traffic across EC2 instances

AWS PaaS services

Service	Description
Elastic Beanstalk	Deploy and scale web applications automatically
AWS Lambda	Serverless compute (Function as a Service)
RDS	Managed relational database service
Aurora	High-performance managed relational database
API Gateway	Build, deploy, and manage APIs
App Runner	Run containerized web applications without managing servers
AWS Fargate	Serverless container service for ECS/EKS

AWS SaaS services

Service	Description
Amazon WorkMail	Managed business email service
Amazon Chime	Online meetings and communication
Amazon Connect	Cloud contact center solution
AWS Managed Microsoft AD	Directory services as a managed solution
Amazon QuickSight	Business intelligence / analytics dashboards
AWS IQ	On-demand AWS expert help platform

Compute Services

Service	Description	Example Use
EC2 (Elastic Compute Cloud)	Virtual servers in the cloud	Host web apps, simulations
Lambda	Serverless compute; runs code on demand	Event-driven apps
Elastic Beanstalk	PaaS for web app deployment	Deploy apps without managing servers
ECS / EKS	Container management (Docker/Kubernetes)	Microservices deployment

For more details, see : https://aws.amazon.com/products/compute/?nc2=h_prod_cp_hub

AWS instances

- An instance is a virtual server with a well specified set of resources including: CPU cycles, main memory, secondary storage, communication and I/O bandwidth.
- The user chooses:
  - The region and the availability zone where this virtual server should be placed.
  - An instance type from a limited menu of instance types.
- When launched, an instance is provided with a DNS name; this name maps to a
  - *private IP address* → for internal communication within the internal EC2 communication network.
  - *public IP address* → for communication outside the internal Amazon network, e.g., for communication with the user that launched the instance.

AWS instances

- Network Address Translation (NAT) maps external IP addresses to internal ones.

Temporary

- The public IP address is assigned for the lifetime of an instance.

What if the user want a parmanent IP ?

- An instance can request an *elastic IP address*, rather than a public IP address. The elastic IP address is a static public IP address allocated to an instance from the available pool of the availability zone.

- An elastic IP address is **not released** when the instance is **stopped** or terminated and must be released when no longer needed.

Amazon Machine Image (AMI)

- An **Amazon Machine Image (AMI)** is a **preconfigured template** that contains all the information required to **a virtual machine** (a virtual server) in AWS.
- It contains (components) :
  1. **Root Volume Template** : The base operating system (Linux, Windows, etc.) + preinstalled applications + configurations.
  2. **Launch Permissions**: Control which AWS accounts can use this AMI to launch instances.
  3. **Block Device Mapping** : Specifies the volumes (EBS or instance store) that will be attached to the instance when it launches.
  4. **Metadata** : Information about the image (name, ID, architecture type, virtualization type, region)
- When users ask for a specific system : AWS loads the corresponding AMI and creates the virtual machine (VM)

Amazon machine images(AMIs)

- **Public AMIs:** Pre-configured free to use images provided by AWS or community (ex : Fedora, Amazon Linux, Ubuntu, Debian).

- **Private AMIs:** User created images, that are visible only to this user, containing its applications, libraries, data and associated configuration settings (ex : Custom web server image, company environment).

- **Paid AMIs (Third-party / Marketplace AMIs):** Provided by software vendors (e.g., Bitnami, Fortinet, Palo Alto, Red Hat, Canonical) and billed hourly (ex : AMIs with commercial DBMS like Oracle or SQL Server).

EC2 – Elastic Cloud Computing

- **Amazon EC2** stands for **Elastic Compute Cloud**.

- It is the **primary compute service** in Amazon Web Services (AWS) that lets users **create and manage virtual servers** in the cloud.

- In simple terms : EC2 = Virtual Machines in the AWS Cloud.

- **Purpose :**
  - EC2 provides **resizable compute capacity** — meaning that the user can launch, stop, or resize virtual servers (called **instances**) anytime he wants.
  - It's designed to make cloud computing **flexible**, **scalable**, and **cost-efficient**.

EC2 instance components

Each EC2 **instance** includes:

- **CPU & Memory** → compute resources
- **Storage** → EBS (Elastic Block Store) or instance store
- **Networking** → private and/or public IP addresses
- **Operating System** → from the AMI you selected

EC2 - Instance types

- AWS provides many instance "families," each optimized for different workloads:
  - Standard instances: micro (StdM), small (StdS), large (StdL), extra large (StdXL); small is the default.
  - High memory instances: high-memory extra large (HmXL), high-memory double extra large (Hm2XL), and high-memory quadruple extra large (Hm4XL).
  - High CPU instances: high-CPU extra large (HcpuXL).
  - Cluster computing: cluster computing quadruple extra large (Cl4XL).

Instance name	API name	Platform (32/64-bit)	Memory (GB)	Max EC2 compute units	I-memory (GB)	I/O (M/H)
StdM		32 and 64	0. 633	1 VC; 2 CUs		
StdS	`m1.small`	32	1.7	1 VC; 1 CU	160	M
StdL	`m1.large`	64	7.5	2 VCs; 2 × 2 CUs	85	H
StdXL	`m1.xlarge`	64	15	4 VCs; 4 × 2 CUs	1,690	H
HmXL	`m2.xlarge`	64	17.1	2 VCs; 2 × 3.25 CUs	420	M
Hm2XL	`m2.2xlarge`	64	34.2	4 VCs; 4 × 3.25 CUs	850	H
Hm4XL	`m2.4xlarge`	64	68.4	8 VCs; 8 × 3.25 CUs	1,690	H
HcpuXL	`c1.xlarge`	64	7	8 VCs; 8 × 2.5 CUs	1,690	H
Cl4XL	`cc1.4xlarge`	64	18	33.5 CUs	1,690	H

EC2 - Instance cost

- A main attraction of the Amazon cloud computing is the low cost.

Instance	Linux/Unix	Windows
StdM	0.007	0.013
StdS	0.03	0.048
StdL	0.124	0.208
StdXL	0.249	0.381
HmXL	0.175	0.231
Hm2XL	0.4	0.575
Hm4XL	0.799	1.1
HcpuXL	0.246	0.516
Cl4XL	0.544	N/A

How EC2 works

1. **Select an AMI** (e.g., Ubuntu, Windows, or a custom image)
2. **Choose an instance type** (e.g., t3.micro, m5.large, etc. — each has defined CPU/RAM)
3. **Configure networking** (choose a VPC, subnet, assign IPs, security groups)
4. **Configure Storage** : Choose the size and type of the root EBS volume
5. **Launch instance** : AWS provisions a virtual machine for you within seconds.
6. **Connect** : Access it via SSH (Linux) or RDP (Windows) just like a normal computer

EC2 - Instance types

- For more details explore the following link:
- https://aws.amazon.com/fr/ec2/instance-types/

EC2 - Example Use Cases

- Host websites and web apps
- Run enterprise servers or APIs
- Perform big data analytics or simulations
- Train AI and ML models
- Run Dev/Test environments on demand

AWS Lambda

- **AWS Lambda** is :
  - A **serverless compute service**
  - AWS Lambda = serverless, event-driven, auto-scaling compute service:
    - **serverless** : No servers to manage
    - **event-driven** : It runs only when triggered
    - **auto-scaling** : It automatically scales with demand
    - Users only pay for the compute time their code consumes
- **Purpose :**
  - Lambda is designed to run **short-lived functions** in response to events.
  - Users write their code as a **function**, and AWS handles the execution, scaling, and infrastructure automatically.

AWS Lambda
How it works

1. The user uploads his function code (Node.js, Python, Java, Go, C#, Ruby, etc.) to AWS Lambda.
2. The user configures a trigger for his function. Common triggers include:
   - API Gateway requests (run Lambda when an HTTP request arrives)
   - S3 events (run when a file is uploaded)
   - DynamoDB streams (on data changes)
   - CloudWatch Events / Scheduled triggers
3. When the event happens (the trigger occurs):
   - Lambda starts a function container.
   - Lambda automatically executes the code.
   - It scales automatically to handle multiple requests concurrently.
   - The user is billed only for the compute time used (in milliseconds).
4. AWS destroys the container after execution

For more details, see : https://aws.amazon.com/tutorials/run-serverless-code/

EC2 vs Lambda

Feature	Lambda	EC2
Server management	None	You manage servers
Pricing	Pay per execution	Pay per hour/second
Runtime	Up to 15 minutes	Unlimited
Scaling	Auto, instant	You must configure
Use cases	Event-driven apps	Full control workloads

Storage Services

Service	Description	Example Use
S3 (Simple Storage Service)	Object storage for files and backups	Store website assets
EBS (Elastic Block Store)	Block storage attached to EC2	Persistent storage
EFS (Elastic File System)	Shared file storage for EC2 instances	File sharing between VMs
Glacier	Low-cost archival storage	Backup and long-term storage

https://aws.amazon.com/products/storage/?nc2=h_prod_st_hub

S3 – Simple Storage System

- **Amazon S3 (Simple Storage Service)** is a **highly scalable, durable, and secure object storage service** provided by AWS.
- **Purpose :** S3 is used for storing files such as:
  - Documents, images, videos
  - Backups and archives
  - Static website content
  - Data for big data analytics and machine learning
- Service designed to store large objects; an application can handle an unlimited number of objects ranging in size from 1 byte to 5 TB.
- The object names are **global**.

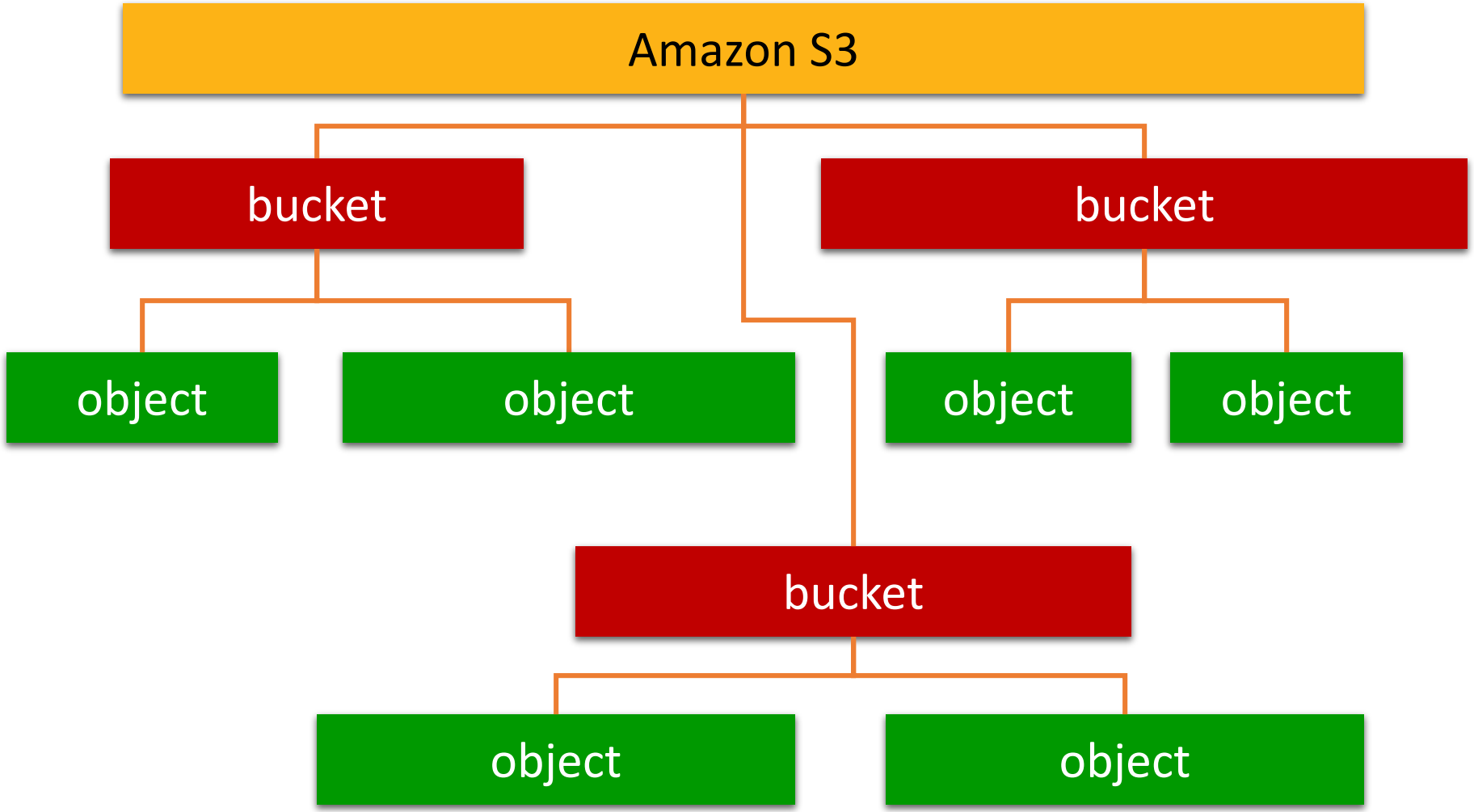
S3 – Key concepts

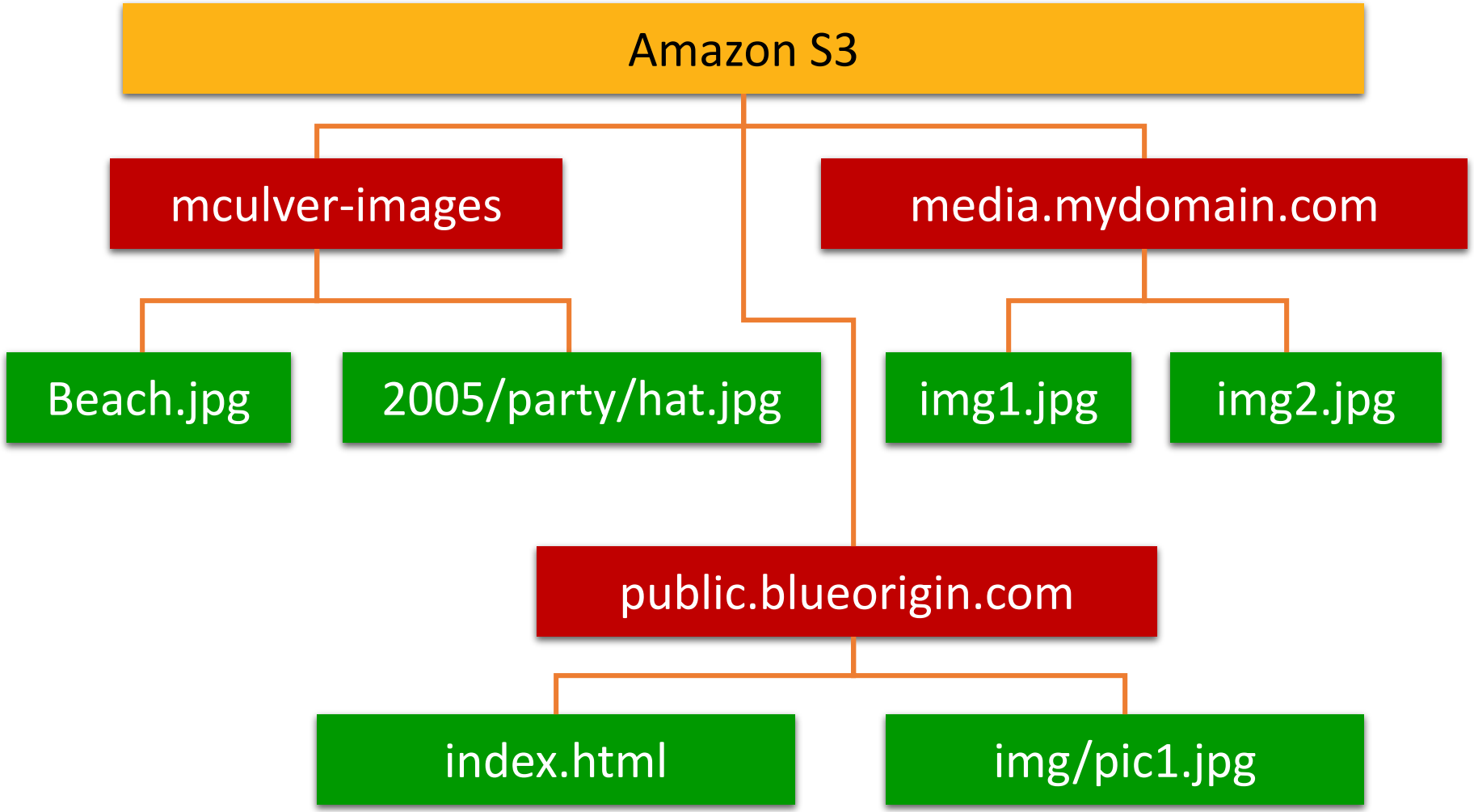
- **Buckets**: Containers for storing objects (files). Each bucket has a unique name in AWS.
- **Objects**: The files stored in S3. Each object consists of data, metadata, and a unique key (name).
- **Regions**: You can choose the AWS region where your bucket is stored.
- **Keys**: Unique identifiers for objects within a bucket.

S3 – Simple Storage System

- An object is stored in a bucket and retrieved via a unique, developer-assigned key; a bucket can be stored in a Region selected by the user.
- Supports a minimal set of functions: write, read, and delete; it does not support primitives to copy, to rename, or to move an object from one bucket to another.
- S3 maintains for each object: the name, modification time, an access control list, and up to 4 KB of user-defined metadata.

S3 namespace

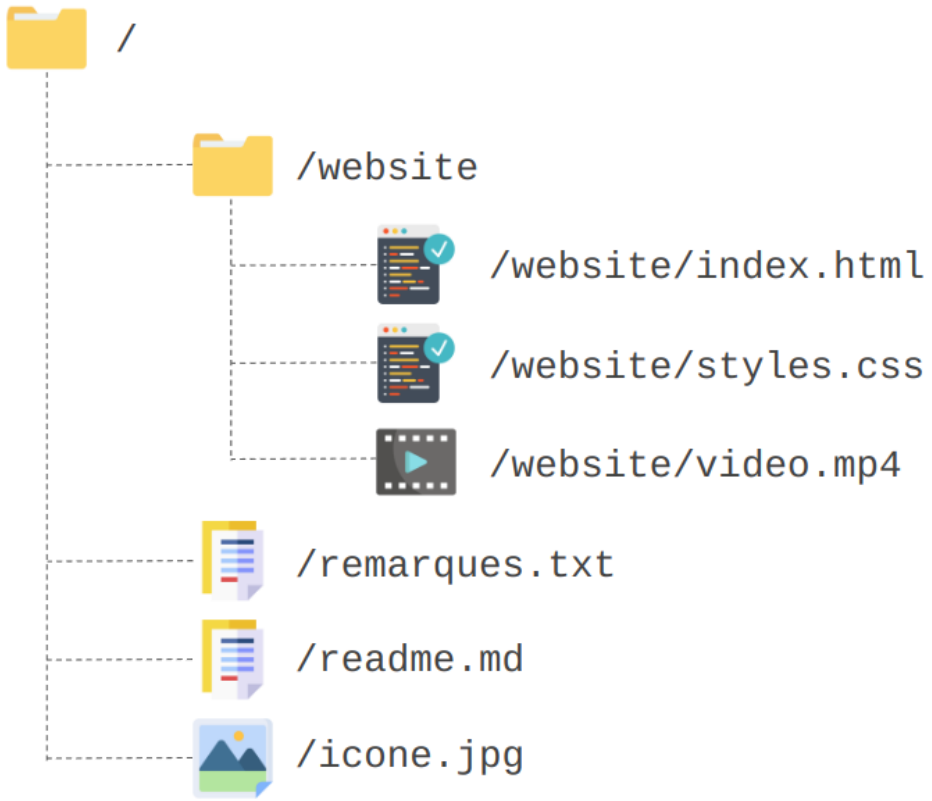



Amazon S3
mculver-images
media.mydomain.com
Beach.jpg
2005/party/hat.jpg
img1.jpg
img2.jpg
public.blueorigin.com
index.html
img/pic1.jpg

Example

Compartiment

- website/index.html
- website/styles.css
- website/video.mp4
- remarques.txt
- readme.md
- icone.jpg



Elastic Block Store (EBS)

- **Amazon EBS (Elastic Block Store)** is a **block storage service** provided by AWS that you can attach to **EC2 instances** to store persistent data.
- Unlike S3, which is object storage, EBS acts like a **hard drive in the cloud**.
- **Purpose** :
  - EBS provides **persistent storage** for EC2 instances.
  - This means that even if your EC2 instance stops or restarts, the data on the attached EBS volume remains intact.
- **In short:**
  - **EBS = cloud hard drive** for your EC2 instances, providing persistent, high-performance block storage suitable for databases, applications, and operating system volumes.

EBS – Key concepts

- **Volumes**: EBS storage units (like virtual hard disks) that can be attached to EC2 instances. The range 1 GB to 1 TB.
- **Snapshots**: Point-in-time backups of EBS volumes stored in S3.
- **Attachment**: A volume can be attached to one EC2 instance at a time (for standard volumes), but an *EC2* instance may mount multiple volumes.
- **IOPS**: Performance metric (Input/Output Operations Per Second) which can vary depending on the volume type.

EBS - Volume

- An EBS volume is a **virtual disk** of a fixed size with a block read/write interface. It can be **mounted** as a filesystem on a running EC2 instance where it can be **updated incrementally.** Unlike an instance store, an EBS volume is **persistent.**

- Fundamental operations:
  - CREATE a new volume (1GB-1TB)
  - COPY a volume from an existing EBS volume or S3 object.
  - MOUNT on one instance at a time.
  - SNAPSHOT current state to an S3 object.

Amazon EBS Volume Types

Volume Type	EBS General Purpose (SSD)	EBS Provisioned IOPS (SSD)	EBS Magnetic
Use Cases	Boot volumes Small to Med DBs Dev and Test	I/O intensive Relational DBs NoSQL DBs	Infrequent Data Access
Storage Media	SSD-backed	SSD-backed	Magnetic disk-backed
Max Volume Size	1TB	1TB	1TB
Max IOPS/volume	3,000 (burst)	4,000	40 - 200
Max throughput/volume	128MBps	128MBps	40 - 90MBps
Max IOPS/instance	48,000	48,000	48,000
Max throughput/instance	800MBps	800MBps	800MBps
API Name	gp2	io1	standard
Price*	$.10/GB - Month	$.125/GB - Month $.065/provisioned IOPS	$.05/GB - Month $.05/million I/O

EBS is approx. 3x more expensive by volume and 10x more expensive by IOPS than S3.

S3 vs EBS

Feature	Amazon S3	Amazon EBS
Storage type	Object storage	Block storage
Accessible via	HTTP REST API, SDK	Mounted filesystem on EC2
Persistence	Independent of EC2	Independent but attached to EC2
Scalability	Virtually unlimited	Limited per volume (16 TiB)
Performance	Higher latency	Low latency, high IOPS
Durability	11 nines	99.999% (EBS snapshots for backup)
Use case	Files, backups, static content	OS disks, databases, transactional apps
Price model	Pay for storage, requests, transfer	Pay for provisioned size and IOPS

Databases Services

Service	Type	Database type	Example Use
RDS (Relational Database Service)	SQL (MySQL, PostgreSQL, etc.)	Relational	Web app database
DynamoDB	NoSQL, key-value	Key-value	Fast, scalable storage
Aurora	High-performance relational DB	Relational	Enterprise-grade apps
Redshift	Data warehouse	Relational	Analytics, BI
Neptune		Graph	Fraud detection, social networking, recommendation engines, generative AI

DynamoDB

- Serverless, fully managed, distributed NoSQL database
- Designed to store and retrieve any amount of data.
- Simple table like storage
  - Weak schema
  - Back-end: a key-value store
- Features
  - Scalable: Dynamo architecture
  - Reliable
    - Replicas over multiple data centers
  - Speed
    - Fast, single-digit milliseconds
  - Secure

DynamoDB - Data Model

- table
  - Container, similar to a worksheet in excel,
  - Cannot query across domains
- Item
  - Item name
  - item name ->(Attribute, value) pairs
  - An item is stored in a domain (a row in a worksheet. Attributes are column names)
- Example
  - domain: “cars”
  - Item 1: “car1”:{“make”:”BMW”, “year”:”2009”}

DynamoDB - Data Model

- Primary key of table
  - Single key (hash)
  - Hash-range key
    - A pair of attributes: first one is hash key, 2nd one is range key.
    - Example: Reply(Id, datetime, …)
- Data type
  - Simple: string and number
  - Multi-valued: string set and number set

Example

Example items

```
{
   Id = 201
   ProductName = "18-Bicycle 201"
   Description = "201 description"
   BicycleType = "Road"
   Brand = "Brand-Company A"
   Price = 100
   Gender = "M"
   Color = [ "Red", "Black" ]
   ProductCategory = "Bike"
}
```

```
{
   Id = 202
   ProductName = "21-Bicycle 202"
   Description = "202 description"
   BicycleType = "Road"
   Brand = "Brand-Company A"
   Price = 200
   Gender = "M"
   Color = [ "Green", "Black" ]
   ProductCategory = "Bike"
}
```

Access methods

- Amazon DynamoDB is a web service that uses HTTP and HTTPS as the transport method
- JavaScript Object Notation (JSON) as a message serialization format
- APIs
  - Java, PHP, .Net
  - Boto

SimpleDB

- Non-relational data store. Supports store and query functions traditionally provided only by relational databases.
- Supports high performance Web applications; users can store and query data items via Web services requests.
- Creates multiple geographically distributed copies of each data item.
- It manages automatically:
    - The infrastructure provisioning.
    - Hardware and software maintenance.
    - Replication and indexing of data items.
    - Performance tuning.

Case studies and Practical Examples

- **Example 1: Host a Static Website on S3**

1. Upload HTML/CSS/JS files to an S3 bucket.
2. Enable “Static website hosting.”
3. Use Route 53 for custom domain.

Case studies and Practical Examples

- **Example 2: Deploy a Web App on EC2 + RDS**

1. Launch EC2 instance, install web server.
2. Connect to RDS database.
3. Configure security groups and load balancing.

Case studies and Practical Examples

- **Example 3: Build a Serverless App (Lambda + API Gateway)**

1. Create a Lambda function triggered by an HTTP API Gateway event.
2. Store data in DynamoDB.
3. Visualize performance in CloudWatch.

Open source platforms for private clouds

Open-source platforms for private clouds

- *OpenStack* - Started by NASA, now an open source platform
- *Eucalyptus* - can be regarded as an open-source counterpart of Amazon's EC2.
- *Open-Nebula* - a private cloud with users actually logging into the head node to access cloud functions. The system is centralized and its default configuration uses the NFS file system.
- *Nimbus* - a cloud solution for scientific applications based on Globus software; inherits from Globus:
    - The image storage.
    - The credentials for user authentication.
    - The requirement that a running Nimbus process can **ssh** into all compute nodes.

OpenStack

OpenStack

- **Origin** : NASA (Nebula Cloud platform), with Rackspace (Cloud Files technology)
- **First release**: Oct 2010
- **Community** : Open source
- **Purpose:** Build and manage your own cloud infrastructure, similar to AWS but on-premise.
- **Service Model:** IaaS
- **Deployment Model:** Private Cloud (main), Hybrid Cloud (common), Community Cloud (possible)

OpenStack

- **IaaS ??!!**
- It provides users with everything at the **infrastructure** level :
  - Virtual machines (Nova, hypervisors)
  - Networks (Neutron)
  - Storage (Cinder, Swift)
  - Images (Glance)
  - Authentication (Keystone)
- **Equivalent to:**
  - AWS EC2, VPC, and EBS

OpenStack
Applications

- OpenStack is most commonly used to create a **private cloud**
- **It is ideal when:**
  1) You want AWS-like capabilities but on your own servers
  2) You need data sovereignty (government, telecom, enterprises)
  3) You require full control over hardware, security, and network
  4) You want to build a research or academic cloud (common in universities)
- It can also be used for:
  - **Hybrid cloud** (combining OpenStack private cloud + public cloud)
  - **Community cloud** (multiple institutions sharing resources)

What is OpenStack ?

- OpenStack is an **open-source cloud computing platform** designed to build and manage **IaaS (Infrastructure as a Service)** clouds.
- It provides the software to manage compute, storage, and networking resources in a data center through APIs, dashboards, and automation tools.
- In short:
  ➞ **OpenStack = the open-source equivalent of AWS EC2 + EBS + VPC, but running inside your own datacenter.**

How OpenStack is Delivered

OpenStack is available via (distributions):

- **DevStack** (development/testing)
- **RDO** (Red Hat)
- **Canonical Charmed OpenStack** (Ubuntu)
- **Mirantis OpenStack**
- **OpenStack-Ansible**
- **Kolla (Containerized OpenStack)**

High-Level OpenStack Workflow

1. **Request a VM** via CLI, API, or Horizon dashboard
2. **Nova** schedules and boots the instance
3. **Glance** provides the system image
4. **Neutron** configures the network
5. **Cinder** attaches persistent storage (optional)
6. **Keystone** authenticates and authorizes the user
7. **Telemetry** collects metrics

OpenStack architecture

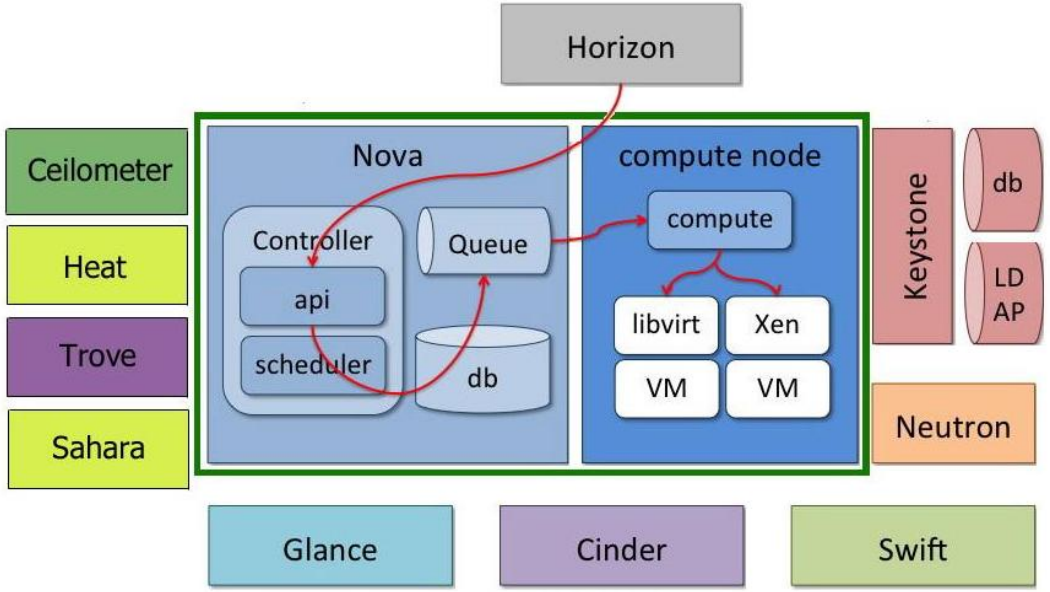
OpenStack operates on a **distributed architecture :**

- **The Control Plane**
- **The Data Plane**

OpenStack architecture
The control plane

It runs

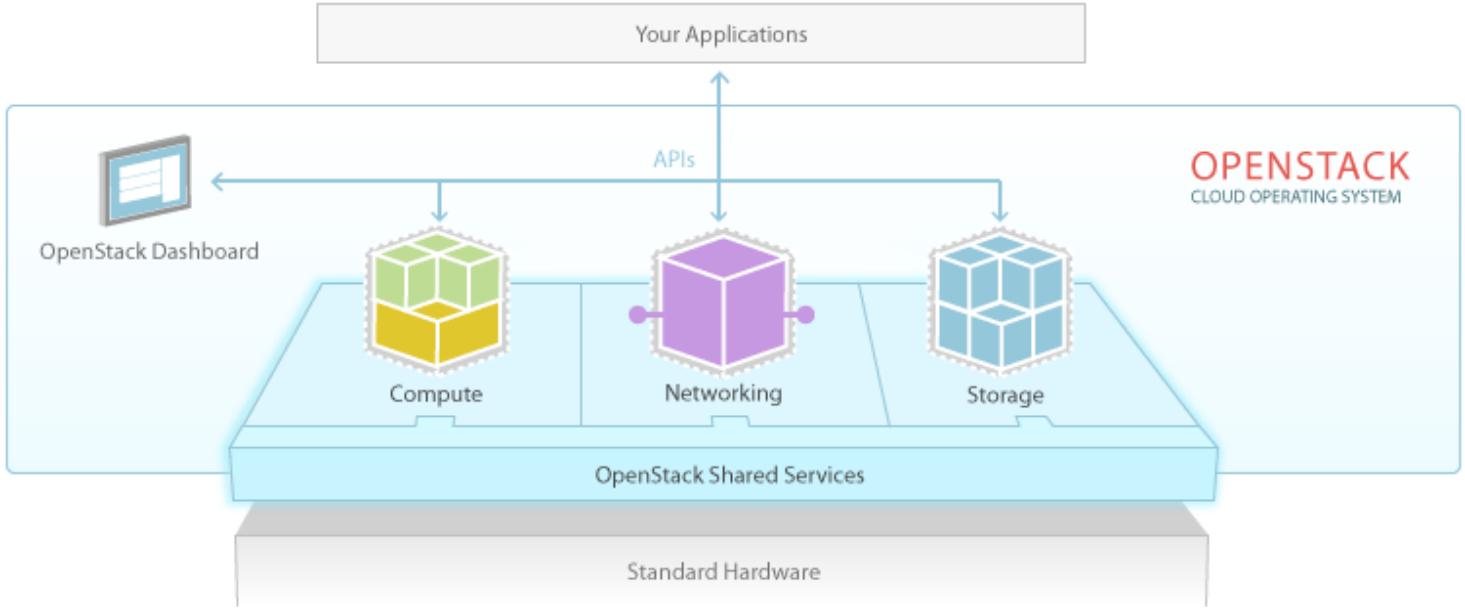
- the central **API** services (one for each major service, like Nova or Neutron),
- the **Database** (to maintain the state of the cloud, e.g., which VM is running where),
- and the **Messaging Queue** (a critical component, often RabbitMQ, that allows all services to communicate asynchronously and reliably).
- All management requests from users or other services go through the Control Plane.



OpenStack architecture
The data plane

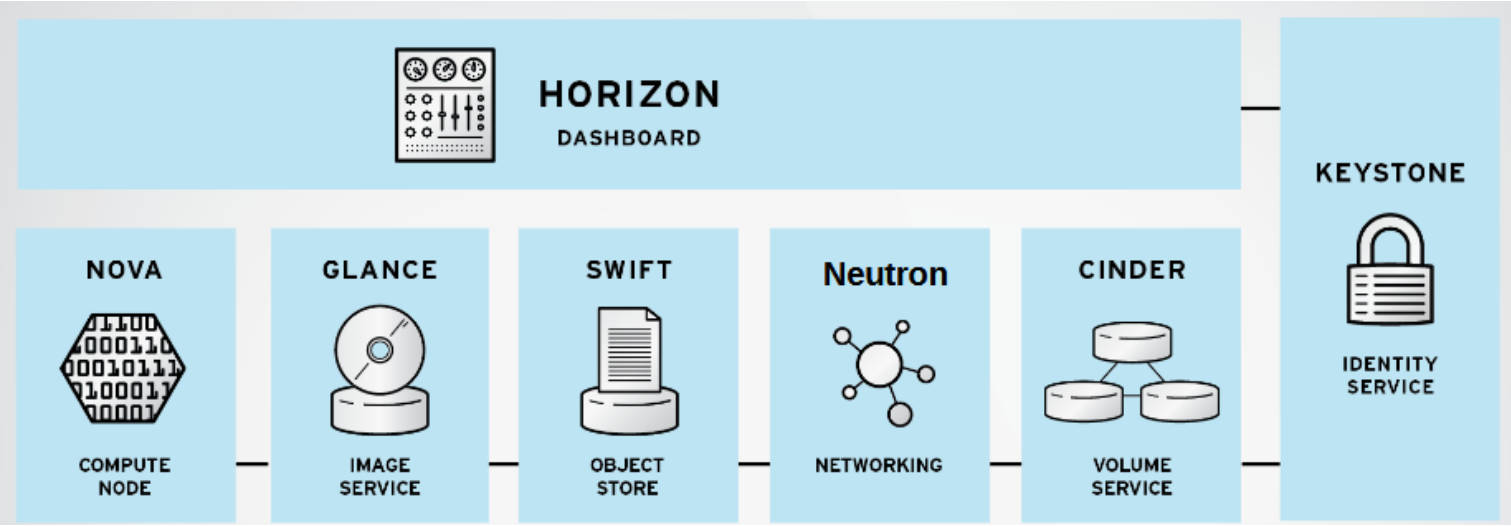
It consists of the physical servers dedicated to hosting user resources.

- Compute Nodes: Run the hypervisors (like KVM) to host Virtual Machines (VMs). The Nova-compute agent runs here to manage the VM lifecycle.
- Storage Nodes: Servers dedicated to providing Block Storage (Cinder) or Object Storage (Swift).
- Network Nodes: Servers often dedicated to running network agents (Neutron) to handle complex networking functions like routing and DHCP.



OpenStack architecture
The foundational modules

- Modular architecture
- Designed to easily scale out
- Based on (growing) set of core services



OpenStack Components

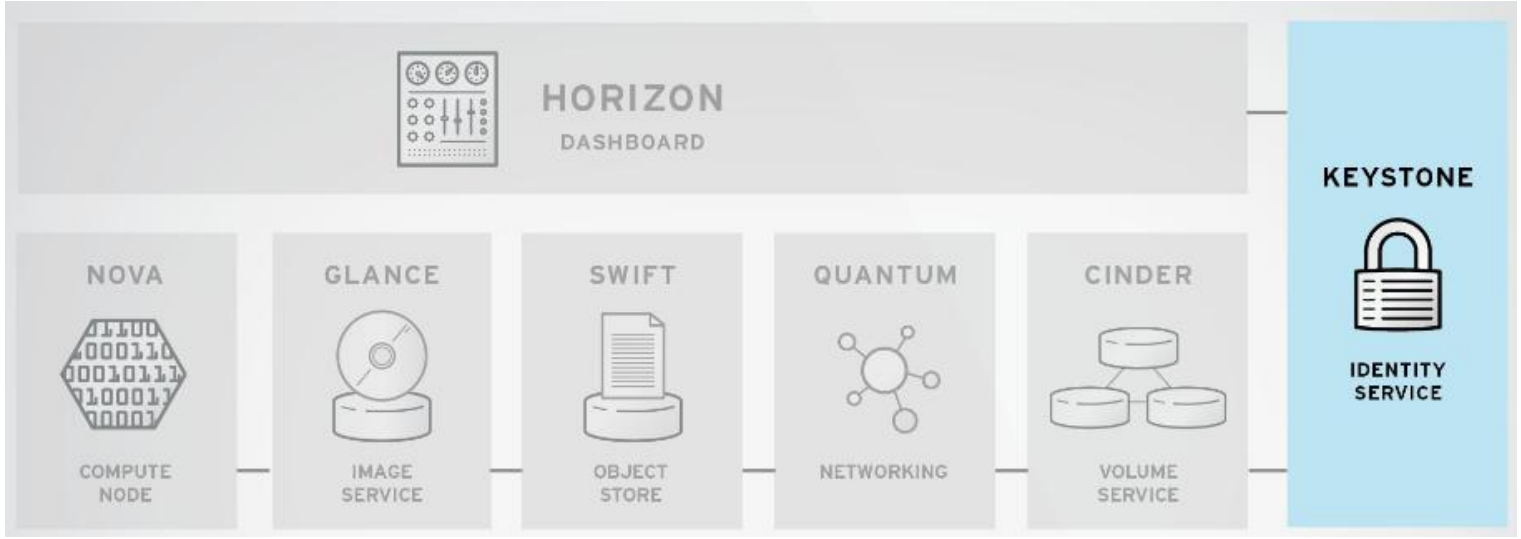
- Nova – Compute (equivalent to EC2)
- Swift – object storage (S3)
- Glance - Image service (AMI)
- Neutron - Networking (virtual network)
- Cinder - Block storage (Elastic block storage)
- Keystone – Authentication
- Horizon - Dashboard (AWS web console)

-- mostly implemented with python

OpenStack architecture

Keystone – Identity Service

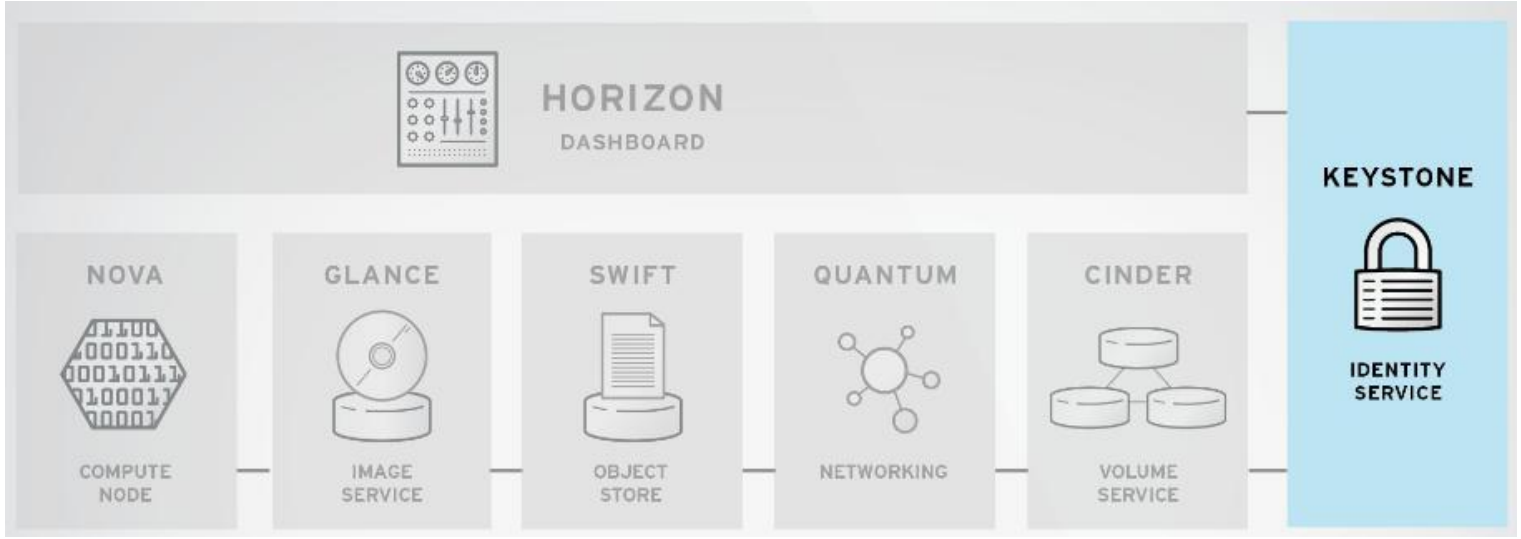
- Central authentication & authorization.
- Manages users, roles, projects, tokens.
- Pluggable backends (SQL, PAM, LDAP, IDM, etc)



OpenStack architecture

Keystone – Identity Service

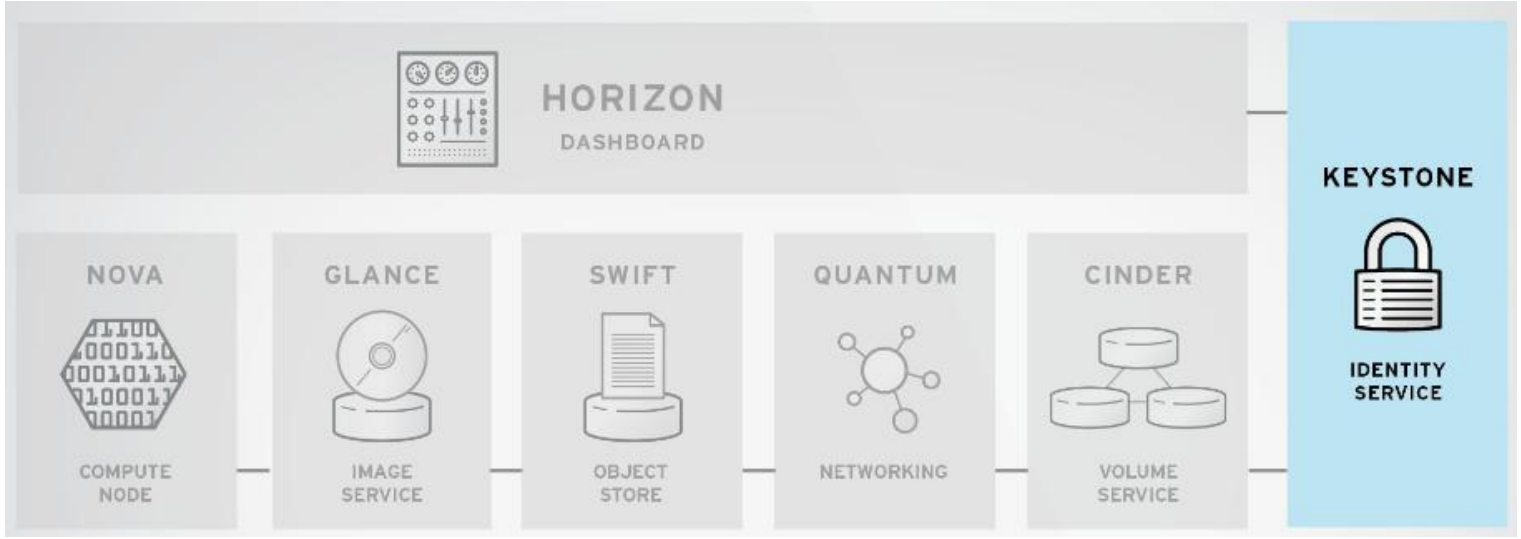
- A **User** is an individual or service entity.
- A **Project** (Tenant) is a container for organizing resources (VMs, networks, storage). Resources are consumed within a Project.
- A **Role** defines the level of access a user has within a specific project (e.g., admin or member).
- **Service Catalog:** Keystone provides a list of all available OpenStack services and their API endpoints (network addresses) so that users and services know how to communicate with them.



OpenStack architecture

Keystone – Identity Service : Keystone provides four essential functions

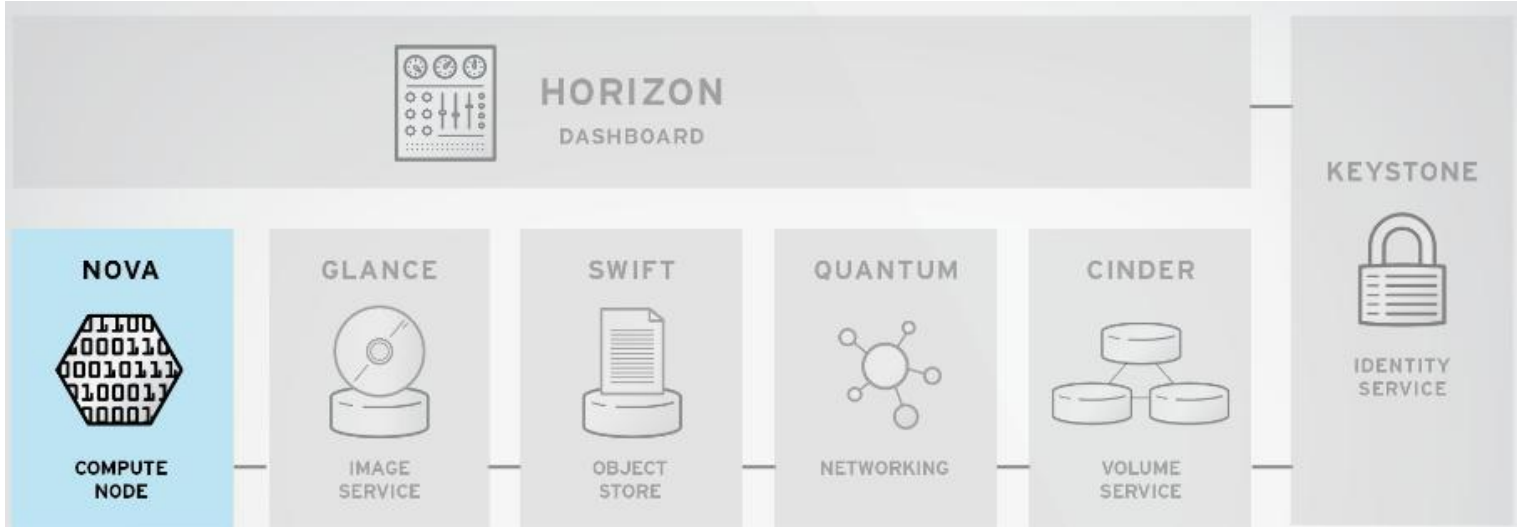
- Authentication (Who are you?)
- Authorization (What are you allowed to do?)
- Service Catalog (Where is everything?)
- Token Management



OpenStack architecture

NOVA – Compute Service

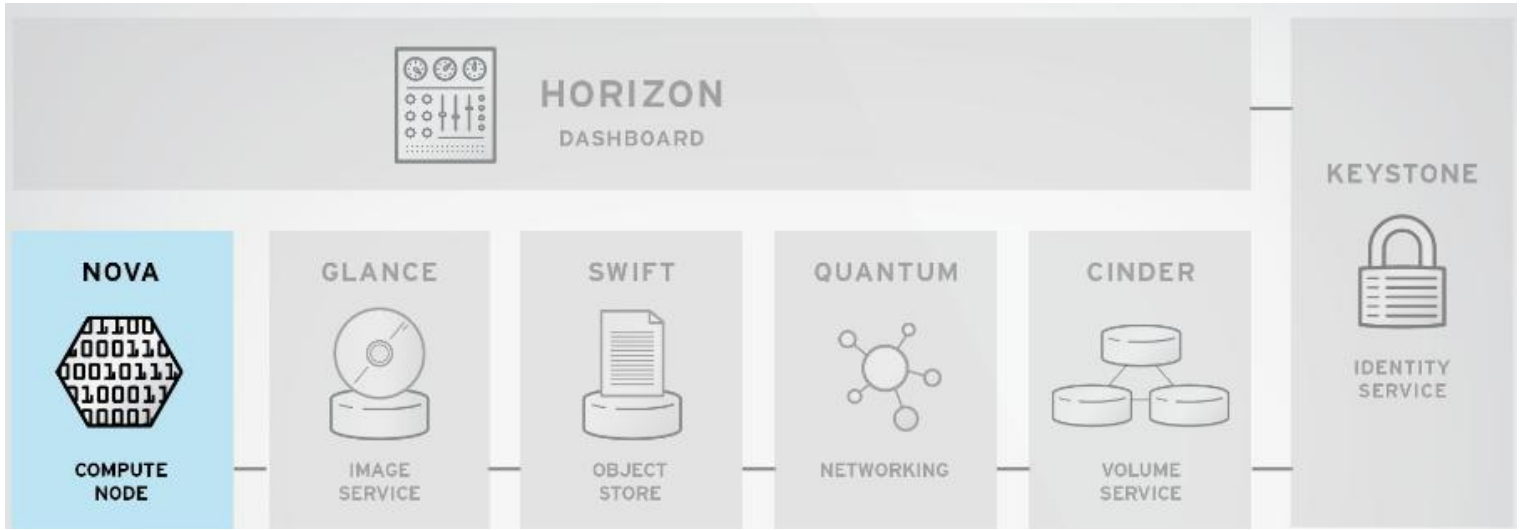
- The central service for managing the lifecycle of Virtual Machines (VMs), from launching to termination (create, start, resize, migrate, delete).
- Orchestrates virtualization.
- Interacts with hypervisors (KVM, Xen, VMware).
- Schedules VMs on compute nodes.



OpenStack architecture

NOVA – Compute Service

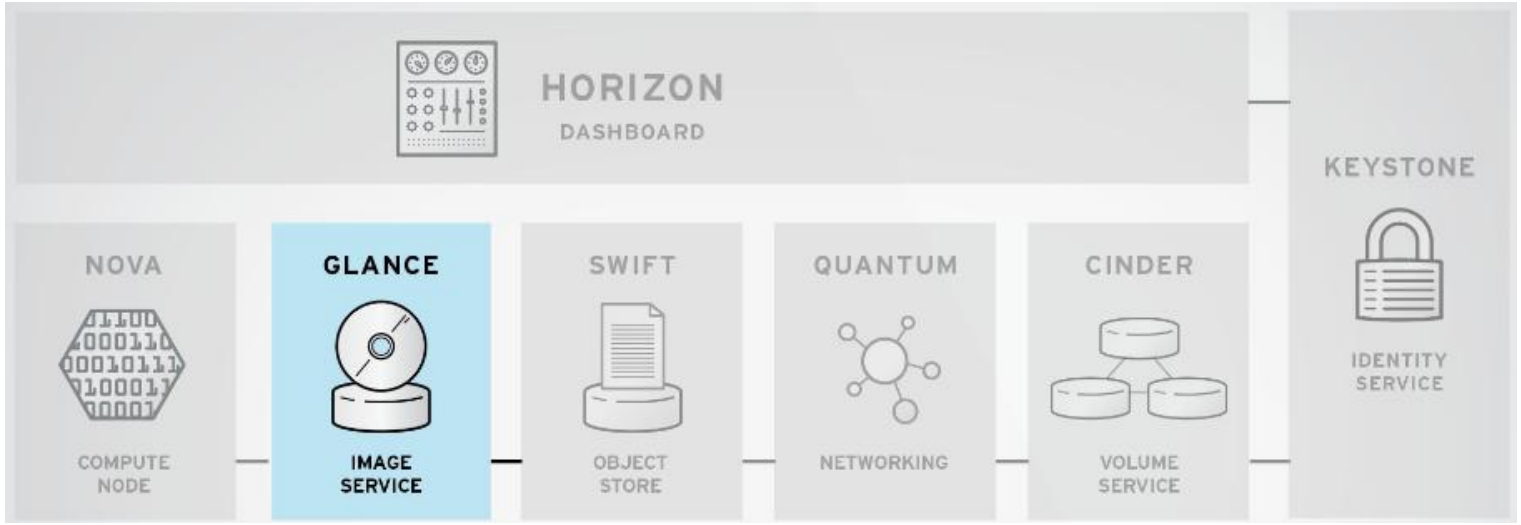
- Its components :
  - Compute Nodes : Hosts the VMs and runs the **Nova-compute** agent to manage the local hypervisor.
  - Distributed controllers that handle scheduling (scheduler), API calls :
    - API server : Receives and validates all VM launch requests.
    - **Scheduler:** Determines the best Compute Node to run a new VM based on load and available resources.



OpenStack architecture

Glance – Image Service

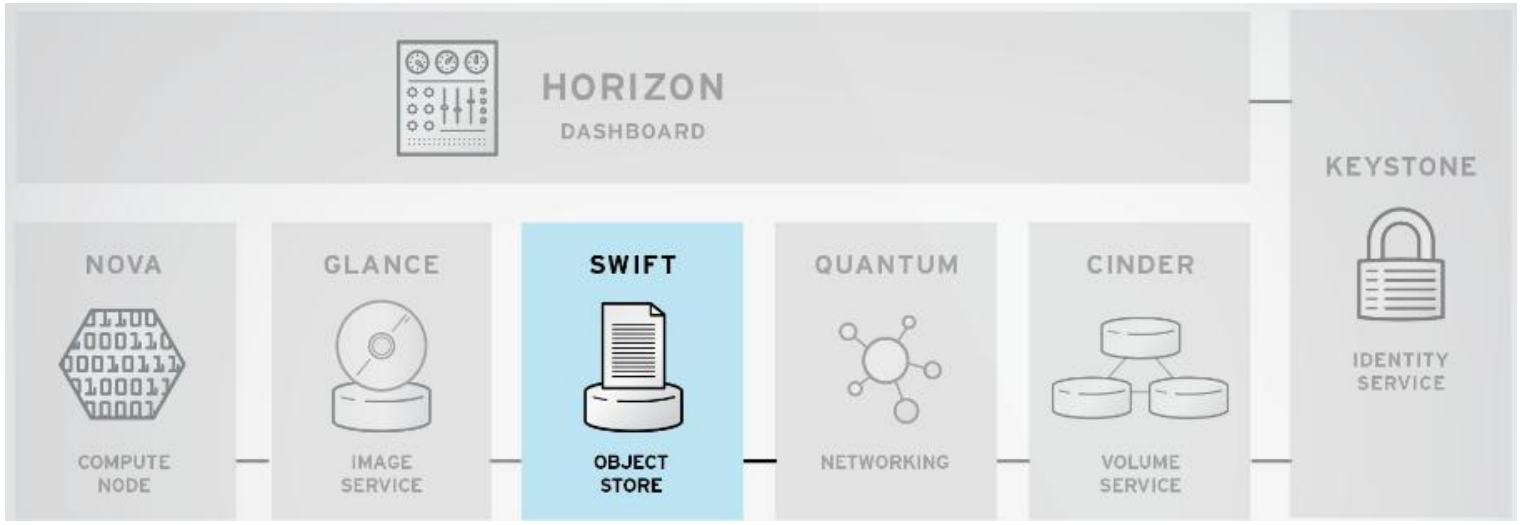
- Stores and manages disk images (templates) used to create VMs,
- Images can be uploaded, versioned, and used by Nova to spawn VMs.
- Supports Raw, QCOW, VMDK, VHD, ISO, OVF & AMI/AKI
- Backend storage : Filesystem, Swift, Gluster, Amazon S3



OpenStack architecture

Swift – Object Storage service

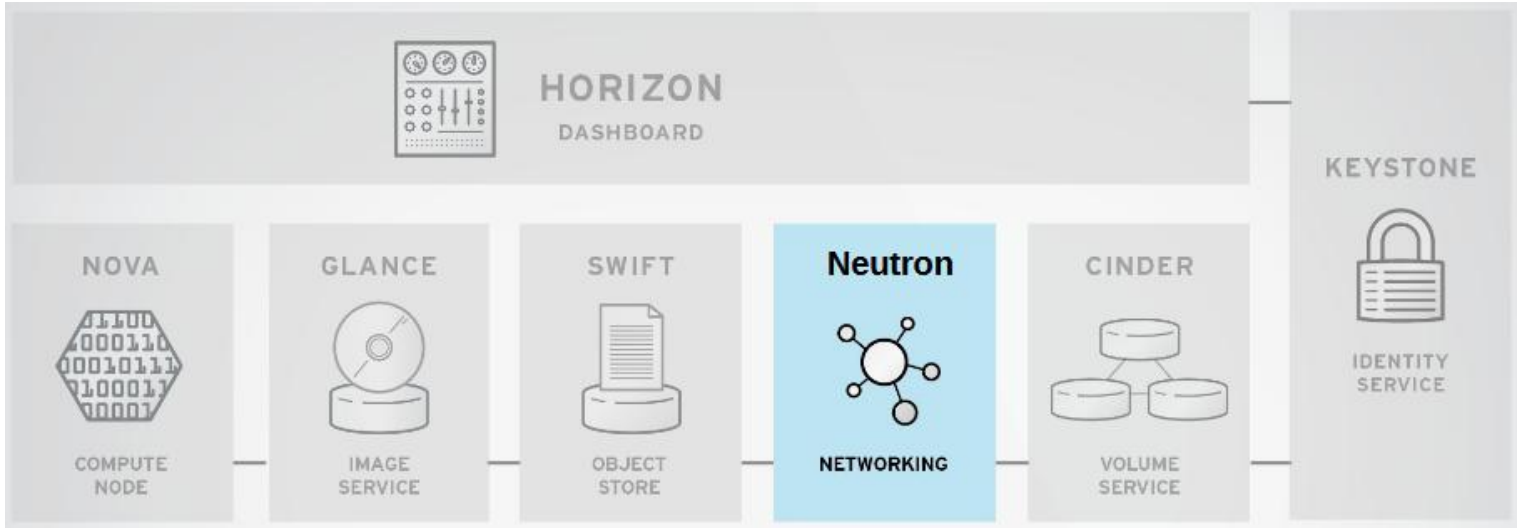
- Native API and S3 compatible API
- Provides highly scalable and redundant **storage for unstructured data** (e.g., backups, archives, photos) accessible via an API.



OpenStack architecture

Neutron – Network Service

- Provides framework for Software Defined Network (SDN)
- handles virtual networks, subnets, routers, and firewalls.

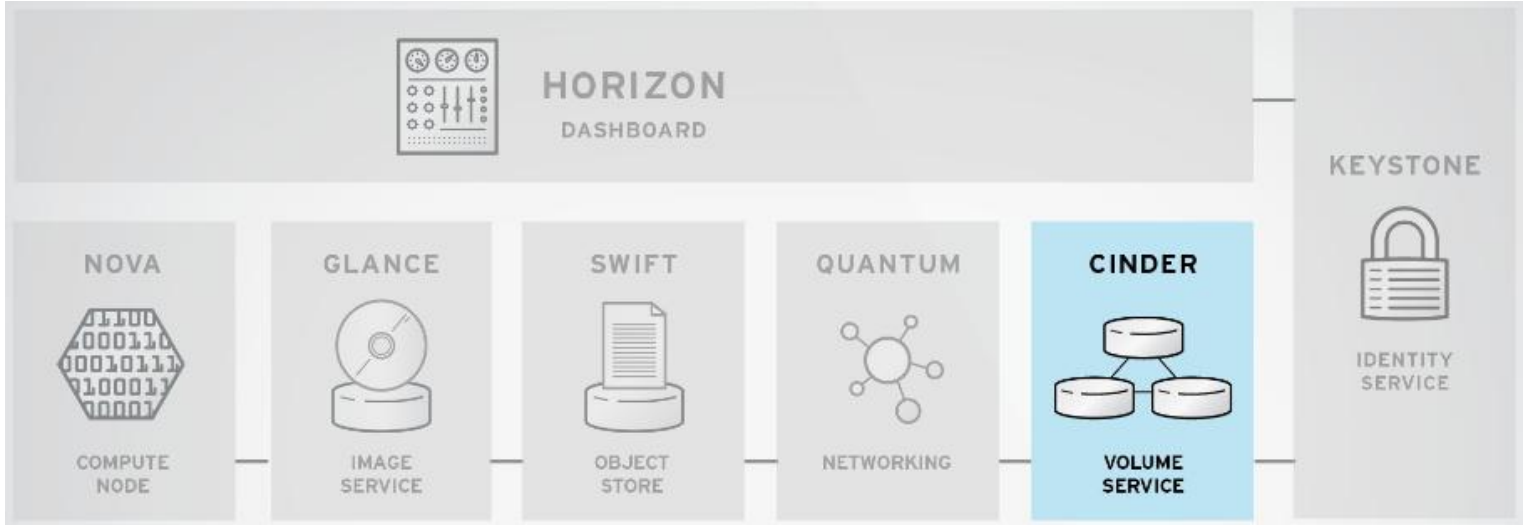


OpenStack architecture

Cinder – Block Storage (Volume) Service

- Provides block storage for virtual machines (persistent disks)
- Similar to Amazon EBS service
- Plugin architecture for vendor extensions

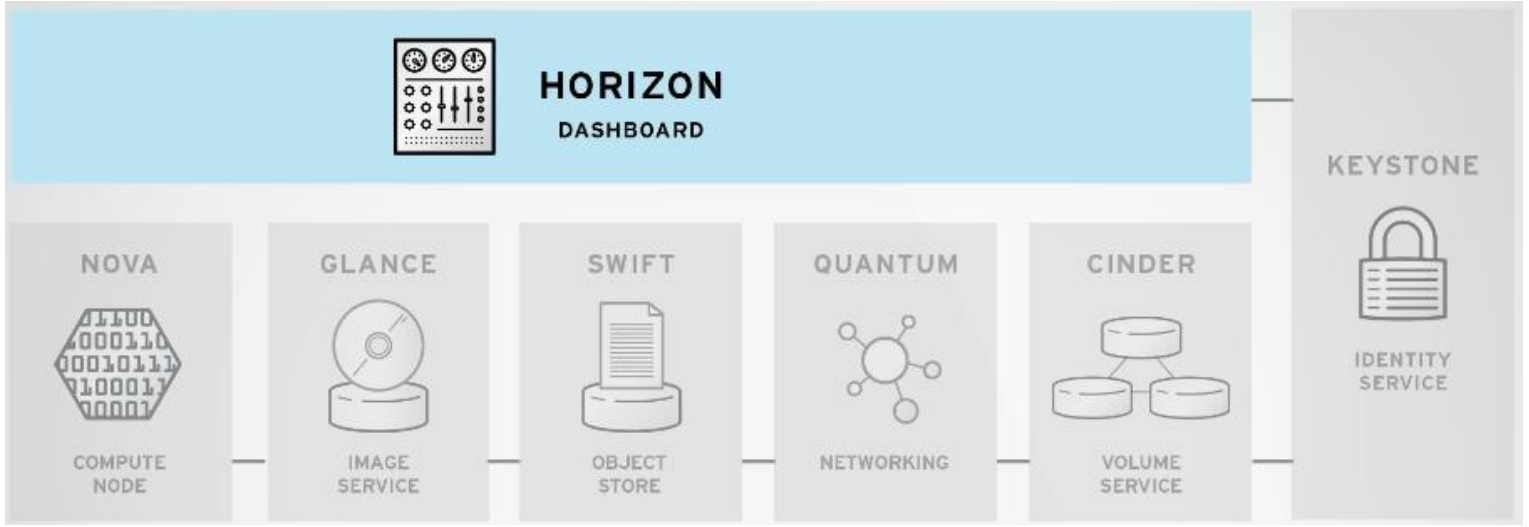
  eg. NetApp driver for Cinder



OpenStack architecture

Horizon – Dashboard

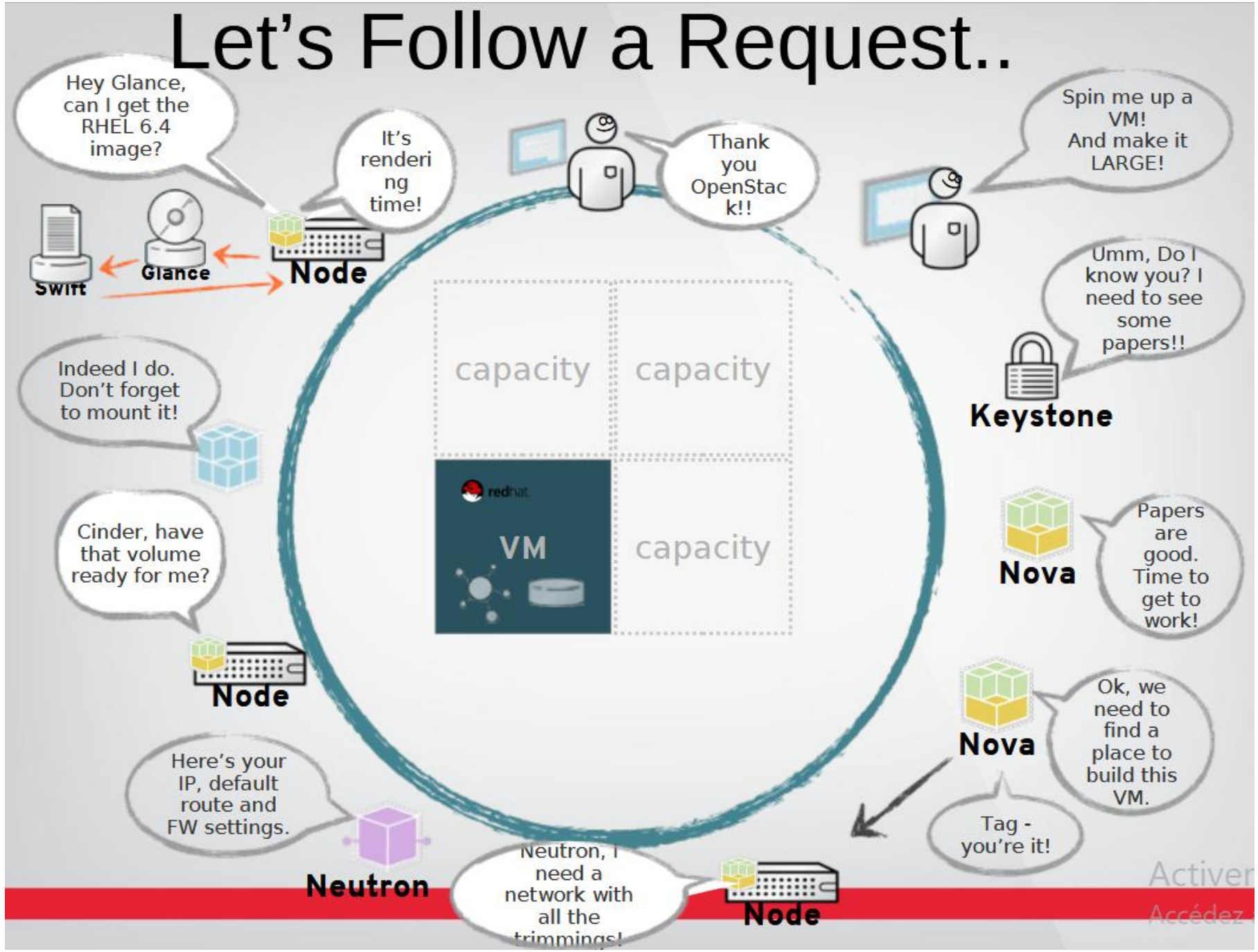
- Provides **web-based GUI**
- Allows users and administrators to interact with and manage all OpenStack services.

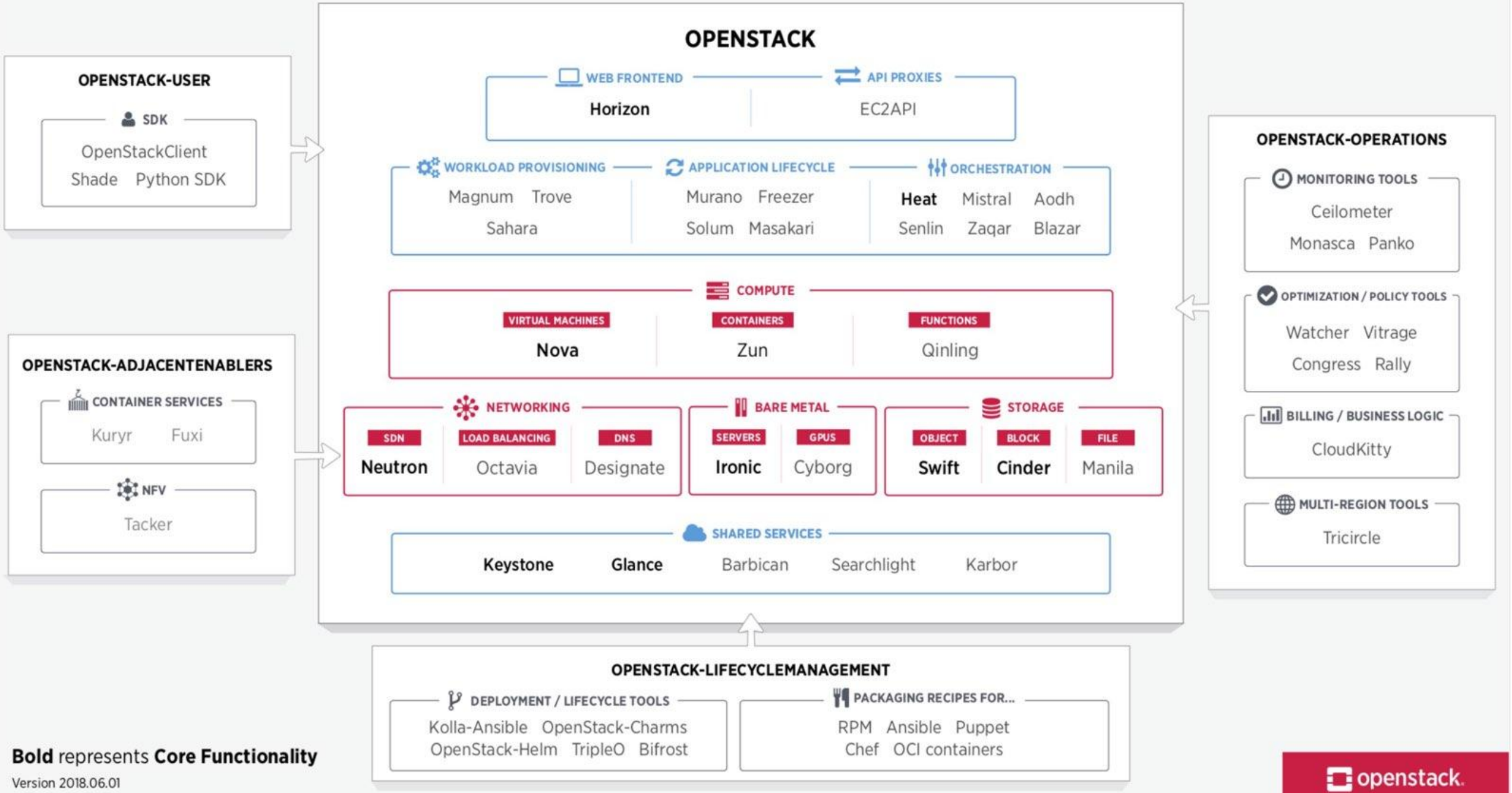


OpenStack architecture
Interaction between components

- **Nova and Neutron**: Nova interacts with Neutron to provide and manage network resources for virtual machines, ensuring connectivity and network isolation.
- **Cinder and Nova**: When a virtual machine needs additional storage, Nova communicates with Cinder to attach block storage volumes, providing persistent storage for applications.
- **Keystone and All Services**: Keystone integrates with all other components to manage authentication and authorization, ensuring secure access and user management.
- **Glance and Nova**: During the launch of new instances, Nova retrieves the required images from Glance, ensuring that the instances boot with the correct configurations.

Let's Follow a Request..





Bold represents **Core Functionality**

Version 2018.06.01

openstack.