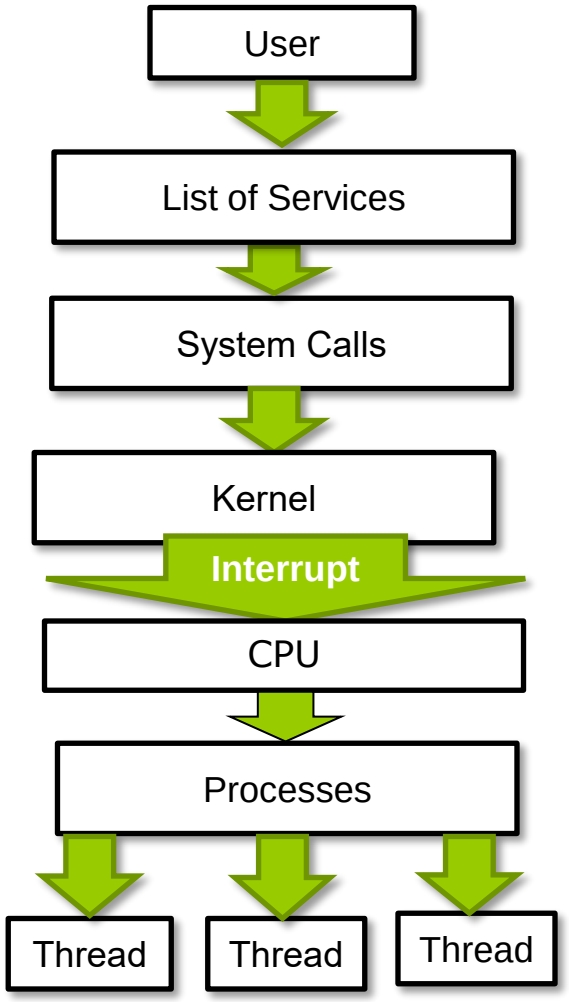
Part 7

Big Pic of OS

The Big Pic of OS



- **Detailed Analysis of Each Part in the OS Diagram**
- This diagram presents a high-level view of how an **Operating System (OS)** functions, showing the flow from user interaction to thread execution. Let’s break down each component:

- **1. User**
- The **user** interacts with the computer through applications or commands. These interactions involve actions like opening a web browser, editing a document, or playing a video.
- The OS serves as a bridge between the user and the hardware.
- Users indirectly communicate with the system via the **List of Services** provided by the OS.

- **2. List of Services**
- This layer represents the **set of functionalities** that the OS provides. Examples include:
- ✅ **File Management** – Opening, closing, reading, and writing files.
  ✅ **Process Management** – Running applications and allocating CPU resources.
  ✅ **Device Management** – Handling input/output devices like keyboards, printers, and storage.
  ✅ **Memory Management** – Allocating and managing RAM for processes.
- These services allow users and applications to perform tasks without needing to interact directly with hardware.

- **3. System Calls**

- **System calls are the interface between user applications and the OS kernel. When a program needs to perform an action requiring OS intervention, it makes a system call.**

Examples of system calls:

- 📌 **open() – Opens a file.**
- 📌 **read() – Reads data from a file or input device.**
- 📌 **write() – Writes data to a file or output device.**
- 📌 **fork() – Creates a new process.**
- 📌 **exec() – Executes a new program.**

- **The system call layer ensures security and controlled access to system resources by managing user requests properly.**

- **4. Kernel**
- The **kernel** is the core component of the OS, responsible for **managing system resources** like CPU, memory, and I/O devices.
- Key responsibilities of the **kernel**:
- **Process Management:** Schedules processes for execution.
- **Memory Management:** Allocates and deallocates RAM for processes.
- **Device Management:** Communicates with hardware components using device drivers.
- **File System Management:** Organizes and retrieves files.
- **Security and Access Control:** Protects system resources from unauthorized access.
- The kernel runs in **privileged mode (kernel mode)**, meaning it has unrestricted access to all system resources.

- **5. Interrupt**
- The **interrupt** system allows the CPU to respond to urgent events by temporarily stopping its current task and executing a special routine to handle the event.
- **Hardware Interrupts** – Triggered by devices (e.g., pressing a key, mouse click, network packet arrival).
- Interrupts ensure the system is responsive and efficient by allowing the CPU to quickly address important tasks.

- **6. CPU (Central Processing Unit)**
- The **CPU** executes instructions from running programs, including:
- Processing system calls.
- Managing process execution.
- Handling interrupts and context switching.
- The CPU operates in **two modes**:
- **User Mode:** Executes user applications with limited system access.
- **Kernel Mode:** Executes OS code with full system control.

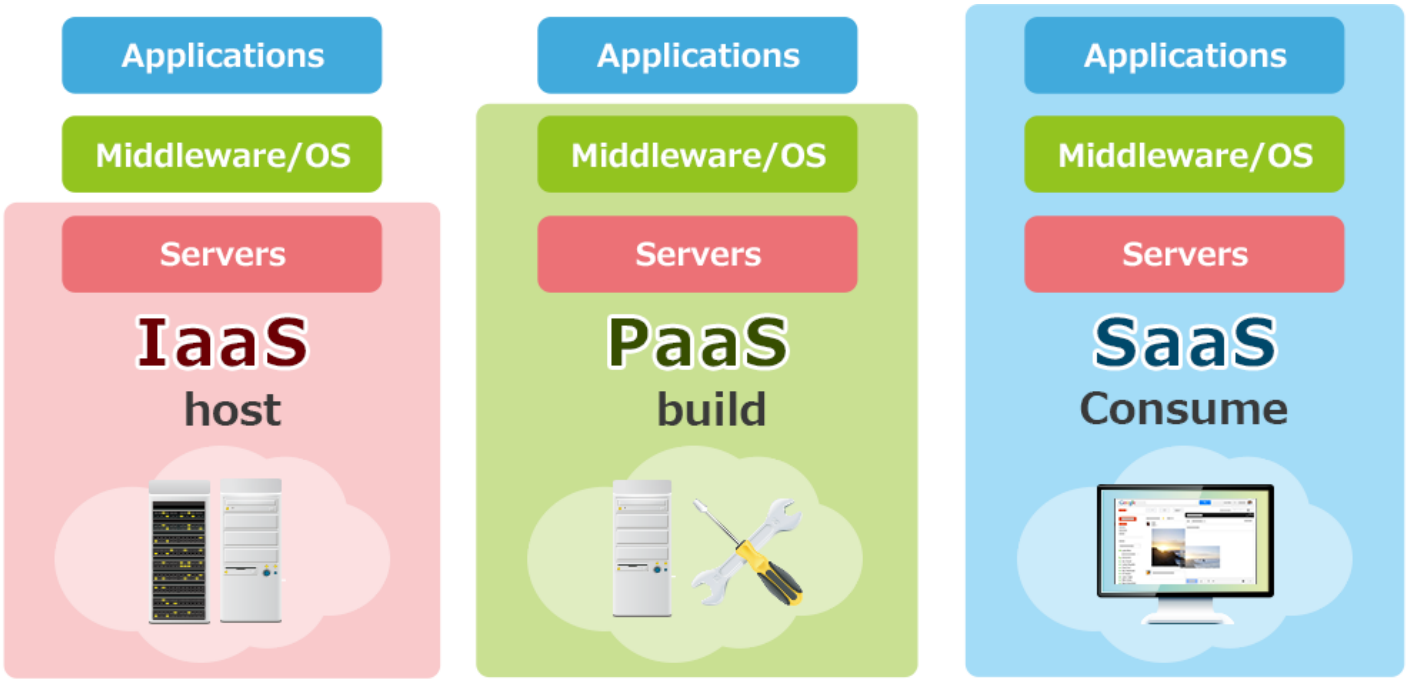
- **7. Processes**
- A **process** is an instance of a program being executed. The OS is responsible for **creating, scheduling, and terminating processes**.
- **Process Lifecycle:**
  1. **New** – Process is created.
  2. **Ready** – Waiting for CPU allocation.
  3. **Running** – Executing on the CPU.
  4. **Waiting** – Blocked, waiting for I/O or resources.
  5. **Terminated** – Process execution ends.
- Each process runs in its **own memory space** to ensure isolation and security.

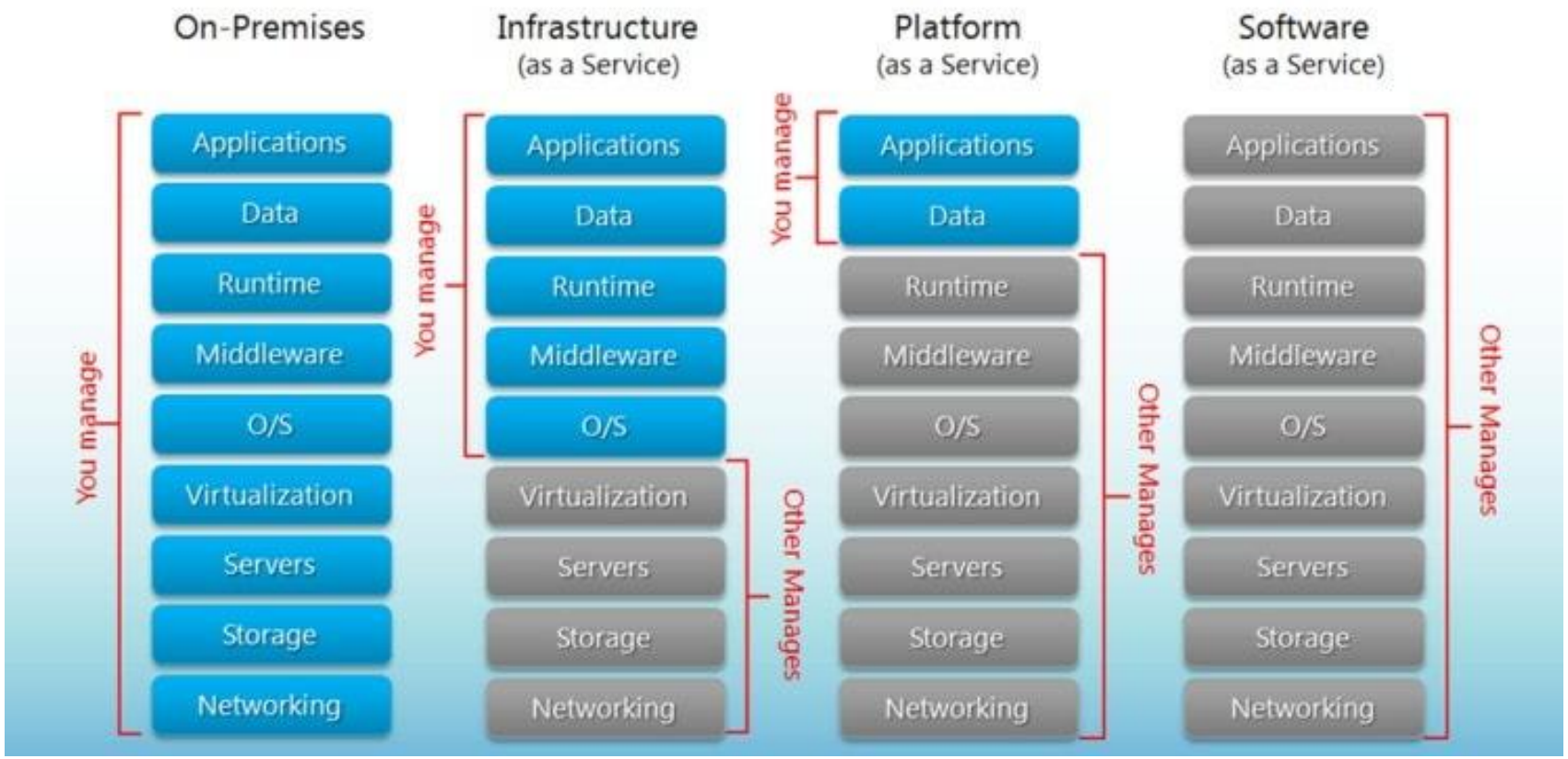
- **8. Threads**
- A **thread** is a lightweight execution unit within a process. A single process can have **multiple threads** running concurrently, improving efficiency.
- 🔹 **Types of Threads:**
- **User-Level Threads:** Managed by the application.
- **Kernel-Level Threads:** Managed by the OS.
- **Example:** A web browser has different threads for:
  ✅ Rendering the UI.
  ✅ Downloading files.
  ✅ Playing videos.
- Multithreading allows programs to **run faster and perform multiple tasks simultaneously**.

- **Summary of OS Flow:**
- 1 **User interacts** with the system via services.
  2 **System calls** request OS functionalities.
  3 The **kernel** manages resources.
  4 **Interrupts** handle system events.
  5 The **CPU** executes processes.
  6 **Processes** contain multiple **threads** for parallel execution.
- This structure ensures an OS can efficiently **manage resources, provide security, and allow multitasking**.

OpenStack: The Cloud Operating System

XaaS


Applications
Middleware/OS
Servers
IaaS
host
Applications
Middleware/OS
Servers
PaaS
build
Applications
Middleware/OS
Servers
SaaS
Consume


On-Premises
Infrastructure
(as a Service)
Platform
(as a Service)
Software
(as a Service)
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
You manage
You manage
You manage
Other Manages
Other Manages
Other Manages

Types of cloud computing

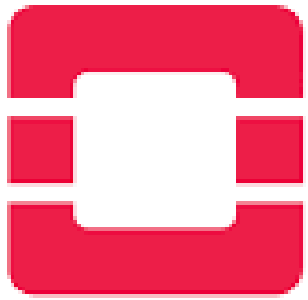
- **Public Cloud**: In Public Cloud the computing infrastructure is hosted by the cloud vendor at the vendor's premises. (AWS)
- The customer has no visibility and control over where the computing infrastructure is hosted.
- The computing infrastructure is shared between any organizations.

- **Private Cloud**: The computing infrastructure is dedicated to a particular organization and not shared with other organizations.
- Private Clouds are more expensive and more secure when compared to Public Clouds.

- Private Cloud is what used to be called your company network.
- It may be accessed whilst inside the firewall, or outside via some form of secure VPN.
- But it is a resource controlled and consumed by your internal IT department.

- **Hybrid Cloud**: Organizations may **host critical** applications on Private Clouds and **applications with relatively less security** concerns on the Public Cloud.
- The usage of both Private and Public Clouds together is called Hybrid Cloud. A related term is ***Cloud Bursting***.

- In **Cloud Bursting** organization use their own computing infrastructure for normal usage, but access the cloud using services like Salesforce cloud computing for high/peak load requirements.
- This ensures that a sudden increase in computing requirement is handled gracefully.

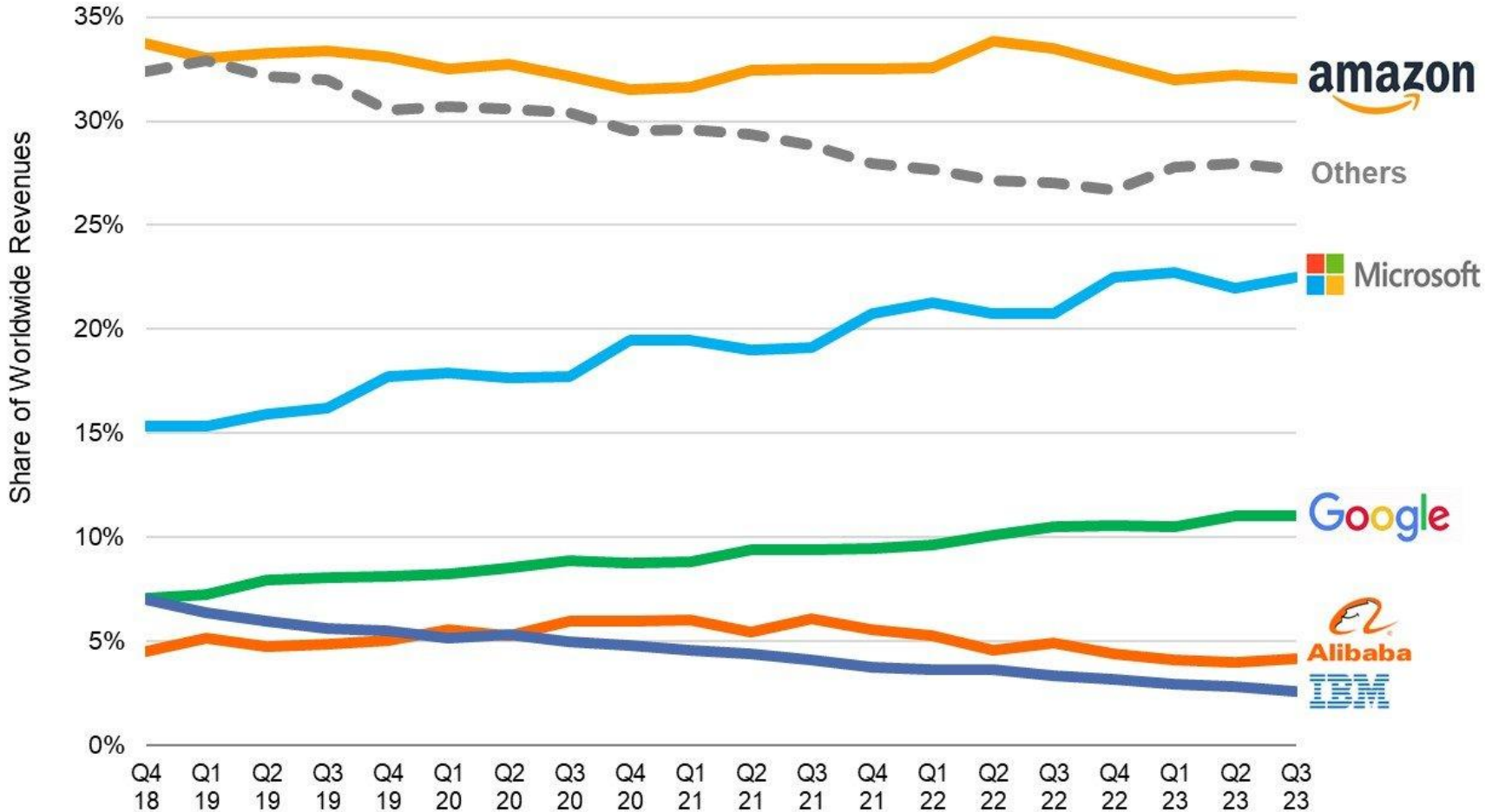


openstack®

OpenStack: The Cloud Operating System

Cloud Provider Market Share Trend

(IaaS, PaaS, Hosted Private Cloud)



Source: Synergy Research Group

Releases Home

Contents

- OpenStack Releases
  - Release Series
  - Series-Independent Releases
  - Teams
  - Cryptographic Signatures
  - References

OpenStack Releases

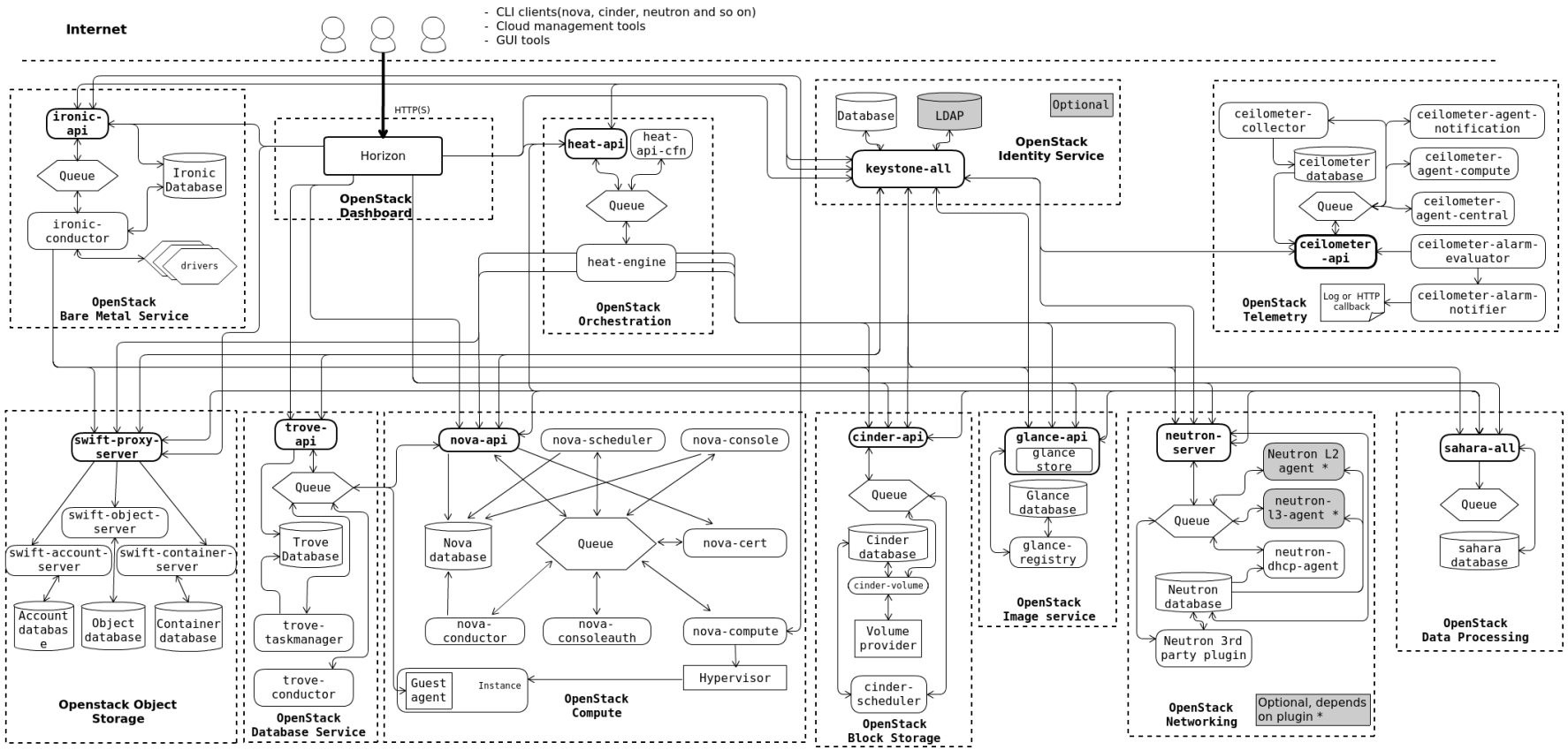
Release Series

OpenStack is developed and released around 6-month cycles. After the initial release, additional stable point releases will be released in each release series. You can find the detail of the various release series here on their series page. Subscribe to the combined release calendar for continual updates.

Series	Status	Initial Release Date	Next Phase	EOL Date
Ussuri	Development	2020-05-13 *estimated* (schedule)	Maintained *estimated 2020-05-13*	
Train	Maintained	2019-10-16	Extended Maintenance *estimated 2021-04-16*	
Stein	Maintained	2019-04-10	Extended Maintenance *estimated 2020-10-10*	
Rocky	Maintained	2018-08-30	Extended Maintenance *estimated 2020-02-24*	
Queens	Maintained	2018-02-28	Extended Maintenance *estimated 2019-10-25*	
Pike	Extended Maintenance	2017-08-30	Unmaintained *estimated TBD*	
Ocata	Extended Maintenance	2017-02-22	Unmaintained *estimated TBD*	
Newton	End Of Life	2016-10-06		2017-10-25
Mitaka	End Of Life	2016-04-07		2017-04-10
Liberty	End Of Life	2015-10-15		2016-11-17
Kilo	End Of Life	2015-04-30		2016-05-02

https://releases.openstack.org/

Logical architecture



OpenStack: The Cloud Operating System

- **OpenStack** is a global collaboration of developers and technologists producing an open source cloud computing platform for public and private clouds.

- OpenStack software delivers a massively **scalable cloud operating system**.
- It is an open source *infrastructure as a service* (IaaS) initiative for creating and managing large groups of virtual private servers in a cloud computing environment.

- The goals of the OpenStack initiative are to support ***interoperability*** between cloud services and allow businesses to build cloud services in their own data centers.

- OpenStack lets users deploy virtual machines and other instances which handle different tasks for managing a cloud environment **on the fly.**
- It makes **horizontal scaling easy**, which means that tasks which benefit from running concurrently can easily serve more or less users on the fly by just spinning up more instances.

Architecture of OpenStack

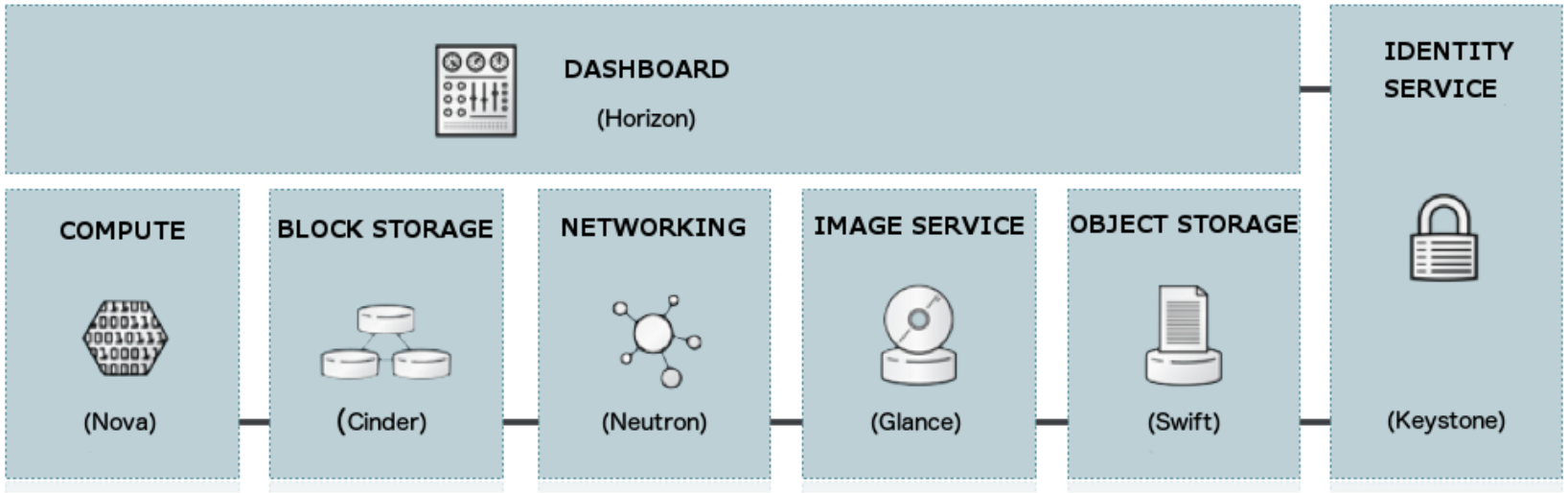
- OpenStack has **a modular architecture** that currently has three components: *compute, storage and image service* :

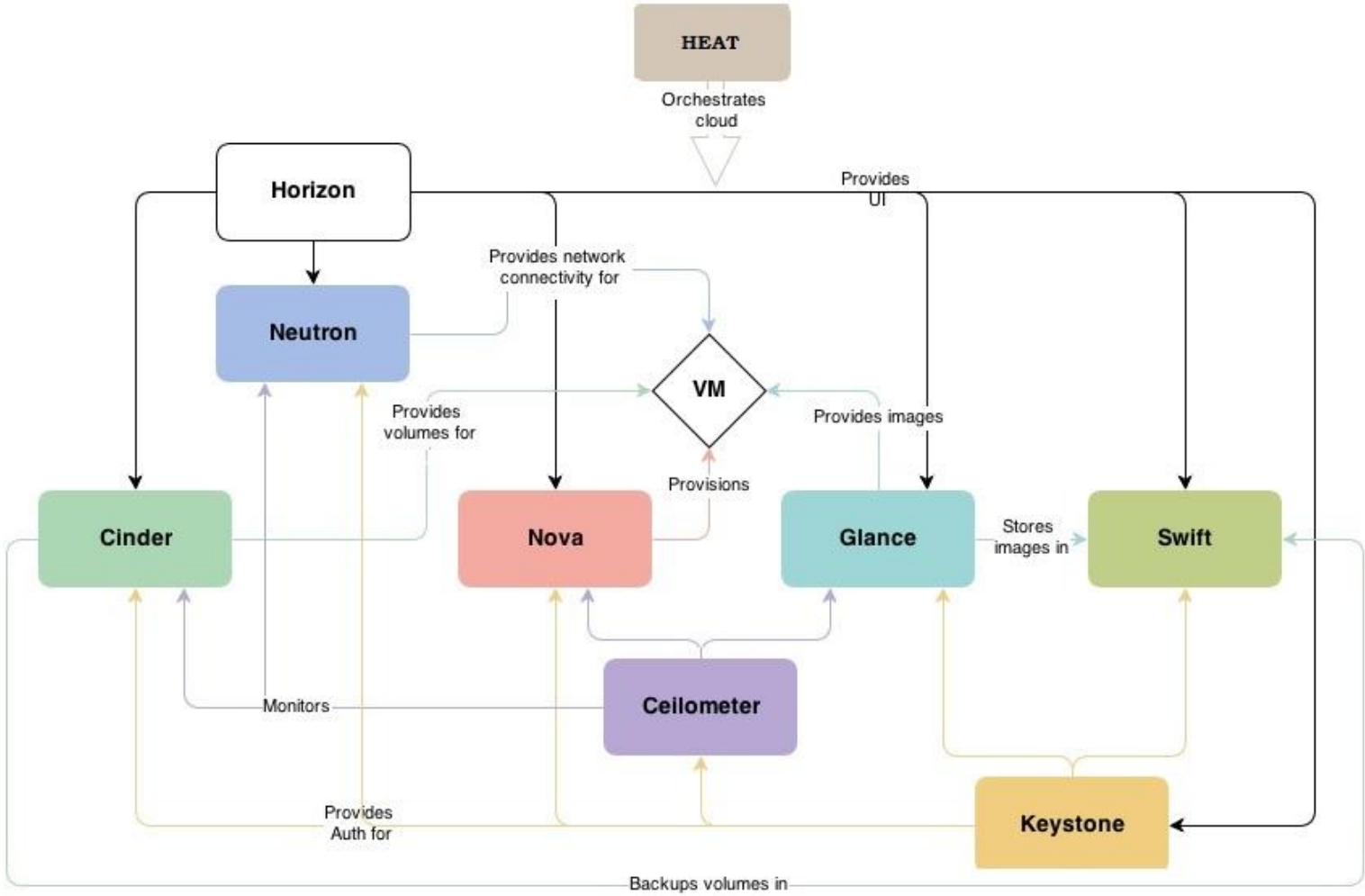
- **Compute:** open source software designed to *provision* and *manage* **large networks of virtual machines**, creating a redundant and scalable cloud computing platform.

- **Storage**: open source software for creating **redundant, scalable object storage** using clusters of standardized servers to store ***petabytes*** of accessible data

- **Image Service**: provides **discovery, registration, and delivery** services for virtual disk images.

Core Components of OpenStack




HEAT
Orchestrates cloud
Horizon
Provides UI
Provides network connectivity for
Neutron
VM
Provides volumes for
Provides images
Provisions
Cinder
Nova
Glance
Stores images in
Swift
Monitors
Ceilometer
Provides Auth for
Keystone
Backups volumes in

- **Nova** is the primary **computing engine** behind OpenStack.
- It is a "fabric controller," which is used for deploying and managing large numbers of virtual machines and other instances to handle computing tasks.

- **Swift** is a **storage system** for objects and files.
- Developers can refer to a unique identifier referring to the file or piece of information and let OpenStack decide where to store this information.

- **Cinder** is **a block storage component**, which is more analogous to the traditional notion of a computer being able to access specific locations on a disk drive.

- **Neutron** provides the **networking capability** for OpenStack.
- It helps to ensure that each of the components of an OpenStack deployment can communicate with one another quickly and efficiently.

- **Horizon** is the **dashboard** behind OpenStack.
- It is the only graphical interface to OpenStack.
- The dashboard provides system administrators a look at what is going on in the cloud, and manage it.

- **Keystone** provides **identity services** for OpenStack.
- It is essentially a central list of all of the users of the OpenStack cloud, mapped against all of the services provided by the cloud which they have permission to use.

- **Glance** provides **image services** to OpenStack.
- In this case, "images" refers to images (or virtual copies) of hard disks.
- Glance allows these images to be used as templates when deploying new virtual machine instances.

- **Ceilometer** provides **telemetry services**, which allow the cloud to provide **billing services** to individual users of the cloud.
- It also keeps a verifiable count of each user's system usage of each of the various components of an OpenStack cloud.

- **Heat** is the **orchestration** component of OpenStack, which allows developers to store the requirements of a cloud application in a file that defines what resources are necessary for that application.