Unit 3: OpenStack Cloud Platform & Serverless Computing

Part A – OpenStack Cloud Platform

1. Introduction to OpenStack Cloud Platform

OpenStack is an **open-source cloud computing platform** used for building and managing public and private clouds.

- It offers **Infrastructure-as-a-Service (IaaS)**.
- Provides a set of interrelated services that manage computing, storage, and networking resources in a data center.
- **Free and open-source**: Licensed under Apache License 2.0.
- Developed by **Rackspace Hosting** and **NASA** in 2010.
- Supported by a large community and companies like IBM, Intel, Red Hat, and Canonical.

Key Features:

- **Scalability** – Handles small setups to large enterprise-scale deployments.
- **Multi-tenancy** – Supports multiple isolated users/projects.
- **API-driven** – Access resources through RESTful APIs.
- **Modular architecture** – Different services for different tasks.
- **Hypervisor support** – Works with KVM, Xen, Hyper-V, VMware ESXi, etc.

Use Cases:

- Private cloud for enterprises.
- Public cloud services.
- Research and scientific computing.
- Development and testing environments.

2. Components of OpenStack

OpenStack has a **modular architecture** with several core components:

Service	Code Name	Function
Compute	**Nova**	Manages virtual machines and compute resources.
Networking	**Neutron**	Manages networking (IP addresses, routers, VLANs).
Storage (Block)	**Cinder**	Provides block storage for VMs.
Storage (Object)	**Swift**	Stores large amounts of unstructured data as objects.
Dashboard	**Horizon**	Web-based UI for managing OpenStack resources.
Identity	**Keystone**	Authentication and authorization service.
Image	**Glance**	Stores and retrieves VM images.
Telemetry	**Ceilometer**	Monitoring and metering usage for billing.
Orchestration	**Heat**	Automates cloud application deployment.

3. Modes of Operation

OpenStack can be deployed in different modes based on requirements:

1. **Private Cloud** – Deployed within an organization's own data center for internal use.
2. **Public Cloud** – Available to the general public over the internet.
3. **Hybrid Cloud** – Combination of private and public clouds for flexibility.
4. **Community Cloud** – Shared by several organizations with common goals.

4. Architecture of OpenStack

The OpenStack architecture is modular and service-oriented:

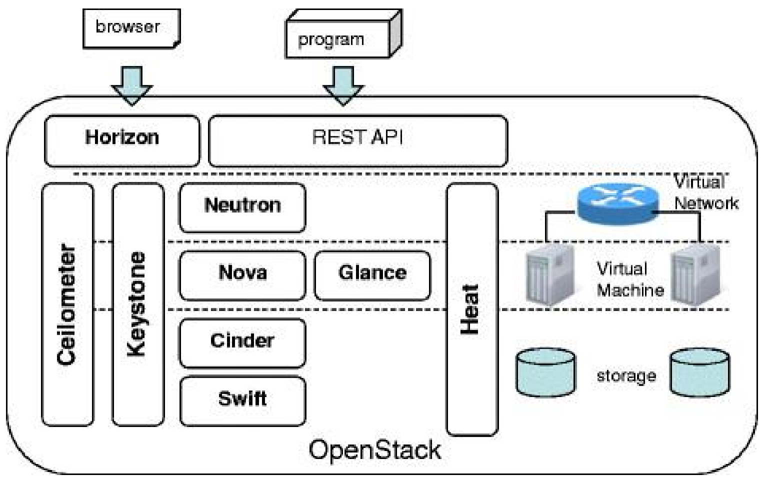
Key Layers:

1. **Dashboard Layer** – Horizon provides a web-based interface for users/admins.
2. **Identity Layer** – Keystone handles authentication/authorization.
3. **Compute Layer** – Nova manages VM lifecycle (launch, terminate, resize).

4. **Networking Layer** – Neutron manages networking services for VMs.
5. **Storage Layer** –
   - Swift for object storage.
   - Cinder for block storage.
6. **Image Service Layer** – Glance stores and retrieves OS images.
7. **Telemetry Layer** – Ceilometer collects usage data.

Diagram:

```
[ Horizon (UI) ]

     ↓

[ Keystone (Identity) ]

     ↓

[ Nova (Compute) ] — [ Neutron (Networking) ]

     ↓

[ Cinder (Block Storage) ] — [ Swift (Object Storage) ]

     ↓

[ Glance (Image Service) ]

     ↓

[ Ceilometer, Heat, etc. ]
```



Part B – Serverless Computing

1. Introduction to Serverless Computing

- **Definition:** A cloud execution model where the cloud provider dynamically manages server infrastructure.
- Users write and deploy **functions**; servers are abstracted away.
- The term "serverless" does not mean no servers exist — it means **server management is handled by the provider**.
- Popular platforms: **AWS Lambda**, **Azure Functions**, **Google Cloud Functions**.

Benefits:

- No server provisioning or management.
- Automatic scaling.
- Pay-per-execution pricing.
- Fast development and deployment.

Limitations:

- Cold start latency.
- Limited execution time.
- Dependency on the cloud provider’s ecosystem.

2. Working with a Serverless Environment

Steps:

1. Write a function in a supported language (Python, Node.js, Java, etc.).
2. Upload to the cloud provider’s serverless platform.
3. Define **triggers/events** that will invoke the function (e.g., API request, file upload).
4. The provider runs the function when the event occurs.
5. User is charged only for the execution time and resources used.

3. Basics of Serverless Events and Functions

- **Event:** A trigger that starts the function (e.g., HTTP request, file upload, database update).
- **Function:** Small, stateless code that runs in response to an event.
- **Handler:** The entry point of the function code.
- **Context:** Provides runtime information to the function (e.g., request details).

4. AWS Lambda (Example)

AWS Lambda is Amazon's serverless computing service.

Key Features:

- Supports multiple languages (Python, Node.js, Java, Go, C#).
- Fully managed execution environment.
- Integrates with AWS services like S3, DynamoDB, API Gateway.

Workflow:

1. Create a Lambda function in AWS Console.
2. Choose a trigger (e.g., S3 file upload).
3. Write/upload code.
4. Deploy the function.
5. Lambda automatically scales and executes the function when triggered.

Example:

- **Event:** User uploads an image to S3.
- **Trigger:** S3 sends event to Lambda.
- **Function:** Lambda resizes the image and stores it in another bucket.

5. AWS Core Services Related to Serverless

- **Amazon S3** – Object storage for triggers/data storage.
- **Amazon DynamoDB** – NoSQL database.
- **Amazon API Gateway** – Create/manage APIs to trigger Lambda.
- **Amazon CloudWatch** – Monitoring and logging.

- **AWS Step Functions** – Orchestrates multiple Lambda functions into workflows.

Summary Table

Topic	**Key Points**
OpenStack	Open-source IaaS platform; modular architecture; services like Nova, Neutron, Cinder, Swift, Horizon.
Modes of Operation	Private, Public, Hybrid, Community clouds.
Serverless	No server management; pay-per-execution; event-driven functions.
AWS Lambda	Fully managed serverless service; integrates with AWS core services.