DISTRIBUTED CLOUD COMPUTING TERM PROJECT

Designing Business Enterprise Level Cloud Computing Solution by Deploying Public and Private Cloud

DECEMBER 11, 2018

GEETIKA KONERU

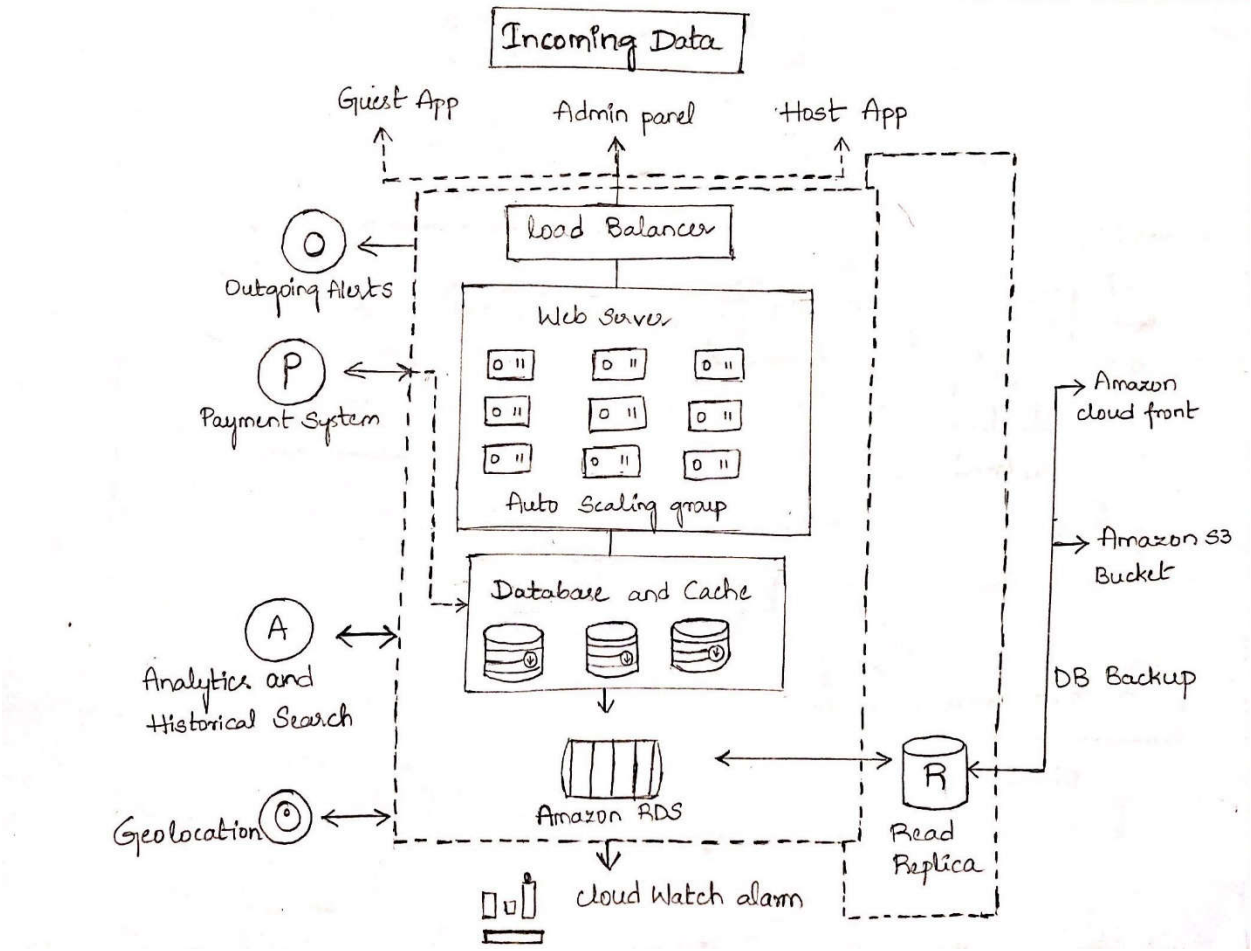
Distributed Cloud Computing Term Project:

Business Objective: The main motivation of the business proposed is to provide an online reservation system which aids the customers to make bookings for their stay while on vacation or to fit in their travel needs, they will be provided with suggestions.

This report aims at providing the high-level system architecture along with the detailed elaboration of individual components involved in the system development. The application is implemented on public and private platforms using AWS (Amazon Web Services) and Open stack respectively. This report clearly depicts the costs incurred in procuring and establishing the system in both the environments and gives a comparison of both the system costs to evaluate and conclude on the cloud platform that perfectly adopts to the business objective.

High Level Architecture:

The key components of the high-level architecture proposed are as follows:



Each of the modules labelled above are defined in sequences as follows: -

1. Incoming data
2. Load Balancer
3. Web Server – Auto Scaling Group
4. Database and Cache
5. Amazon RDS
6. Outgoing alerts
7. Analytics and Historical Search
8. Payment System and
9. Geo location

The database backup is handled through read replica, Amazon cloud front and Amazon S3 bucket.

Detailed Internal Application Flow:

1. **Incoming data module:**

The application is designed such that it is deployed on web and mobile platforms considering the customer ease as a top priority. Hence, the application is expected to receive data from multiple sources as described below:

- The API's (Application Programming Interfaces) often use machine-based interactions such as REST and SOAP. The restful API'S are based majorly on the HTTP methods to access resources via URL- encoded parameters and which use JSON or XML to transmit data. This incoming data from API's flow through the stream alert apps for any notifications of unauthorized access.
- The mundane expected source for the streaming and live input is from the user laptops and workstation where the data is passed to the web servers for storage and retrieval. Hereby in server-less computing it flows to the Amazon kinesis and queries the database which here is Mongo DB. Kinesis is capable of processing hundreds of terabytes per hour from high volumes of streaming data from the sources such as operating logs, financial transactions.
- SaaS applications input is handled by S3 bucket which is used to store data such as from websites, mobile applications, IOT devices. The management features of S3 are easy to use for organization of the data and for configuring finely-tuned access control for 99.99% durability. Hence S3 instances are used in the application.
- The messaging service is handled by SNS, Simple Notification Service.

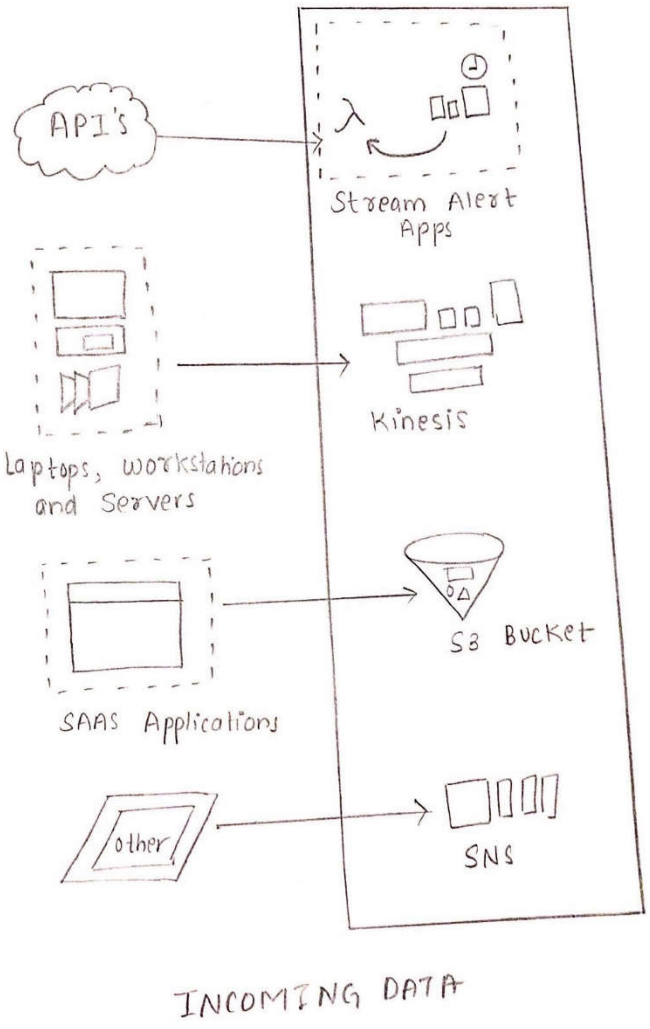


Figure 1Figure showing Incoming Data Module

2. **Load Balancer: -**

Elastic Load Balancing automatically routes incoming web traffic caused by the excess of usage across EC2 instances. By attaching Load balancer to Auto scaling, it uses metrics to scale the application.

Elastic load balancer sends data about the application load balancer and EC2 instances to Amazon CloudWatch. This application uses **Amazon CloudWatch** with Auto Scaling groups to monitor CPU usage and increase the usage of instances during peak stage automatically.

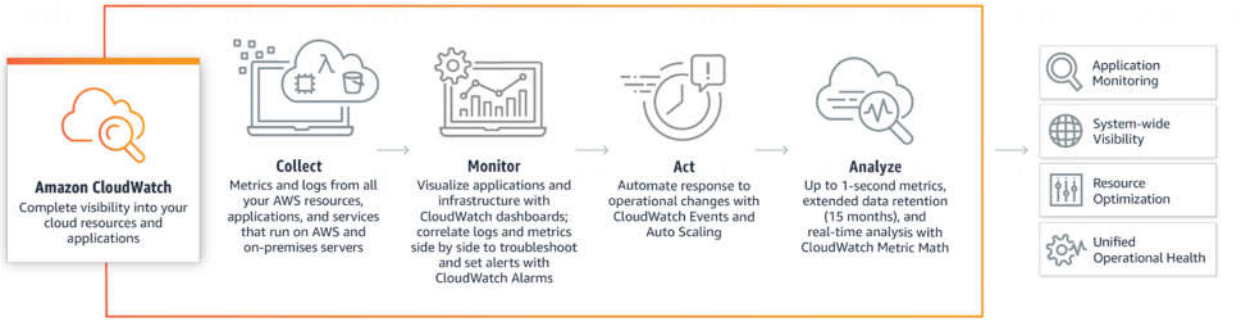
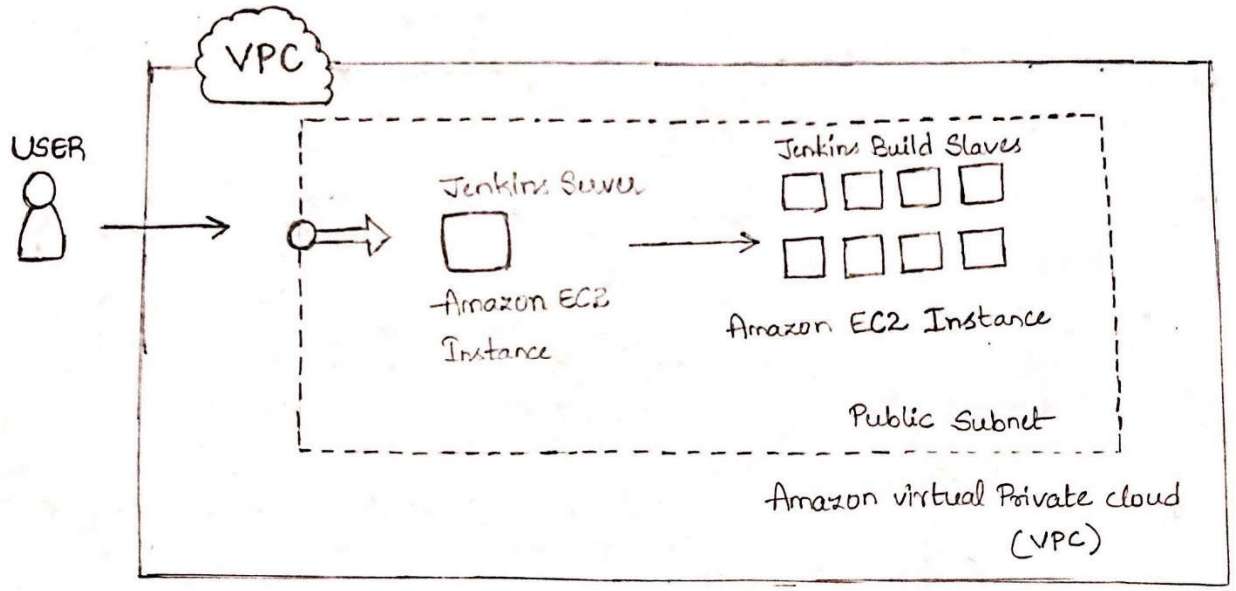


Figure 2 Figure Showing CloudWatch Operations Flow

3. Amazon EC2 Auto Scaling ensures to have a required number of EC2 instances to build this application. Auto Scaling can launch or terminate instances based on demand of the application. It removes excess resource capacity to avoid over spending. Virtual private cloud (VPC) handles AWS resources in a virtual network that would operate in data center. VPC includes public subnet which enables communication over the internet. Private subnet instances only need a private IP and internet traffic is routed through the NAT in the public subnet.

In this application, we create a public-facing subnet for the users to access its websites and mobile apps which have access to the Internet and placing the backend systems such as databases or application servers in a private-facing subnet with no Internet access.



SERVER SETUP AUTO SCALING GROUP

Subnets in VPC handle security groups, NACL for control access to EC2 instances. Launching **EC2 Instances** on VPC and hosting application websites using those instances and can also host multiple websites using single instance.

With the EC2 plug-in, if Jenkins notices that your build cluster is overloaded, it will start instances using the EC2 API and automatically connect them as Jenkins slaves. When the load goes down, excess EC2 instances will be terminated. If one of the build slaves gets damaged, we deploy a particular build instead of deploying each build slave.

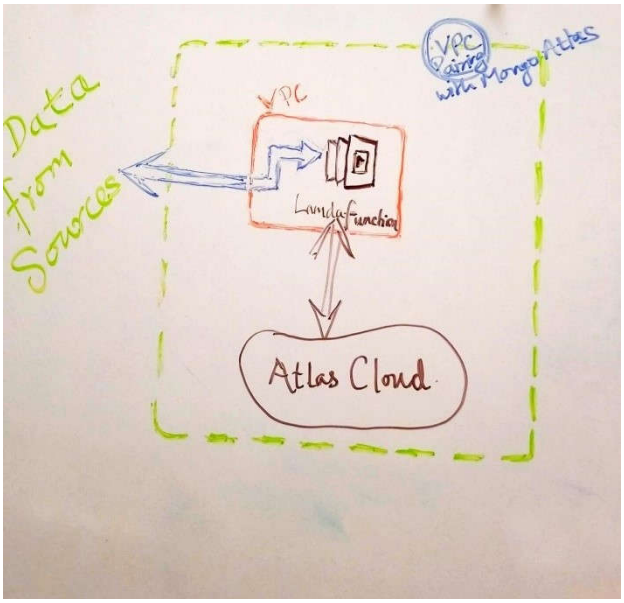
4 Database:

MongoDB is one of the very few databases in the industry that offers scalable, reliable database solution that can constitute all sorts of data. As we are talking about data from online sources, it is highly helpful when choosing a database system that can deal with high variety and velocity of data.

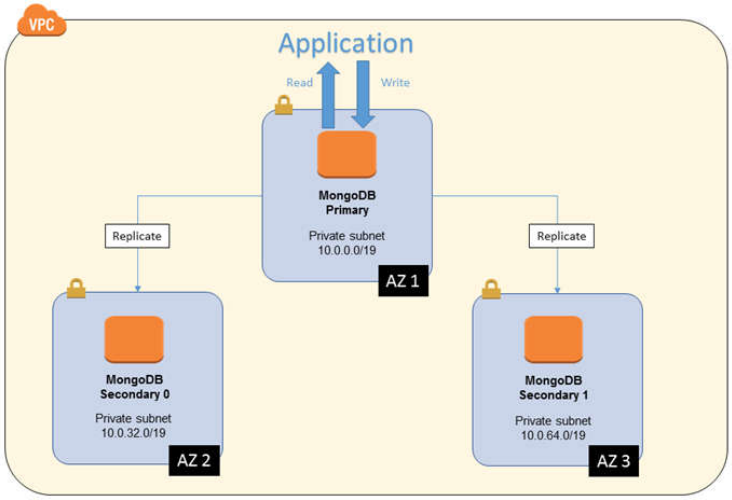
For choosing the database system in designing of this Public cloud infrastructure various other database services were used like DynamoDB, but it was figured that DynamoDB is not as powerful as MongoDB. Also, it was found out that there is no aggregation, the query language isn't as elegant, and it cannot be used to store something complex.

For the database part of the cloud we would be using the MongoDB Atlas which is a fully managed database service. Atlas provides more customization options, functionality and better integration with all the services offered in AWS which was why MongoDB Atlas was used. Also, Atlas provides live migration service which provides an easy interface to migrate services from any other platform into the AWS.

The perks of using MongoDB Atlas is it is highly feasible, flexible, and scalable. As data can be deed in clusters the processing is much faster with better fault tolerance and better faster accessing and analytics possible from the data in the system.



With the help of Virtual Private Cloud connectivity with the Amazon AWS service the MongoDB not only offers versatility but also offers compatibility to the other Big Data Solutions like Apache Spark. The figure demonstrates the basic architecture of the MongoDB Atlas Cloud.



Replication Architecture.

sed in this reference deployment. The of mongoDB instances that holds the data is parallelly stored in order to deal that the data is highly available. The rent availability zones. In the case, the basic workflow of MongoDB constructs includes primary node for read and write operations. The secondary node could also be used as a preference while read operation is being carried out. Normally the writing operations go asynchronously on the primary and the secondary node

Also, the database does sharding, which is distribution of data across multiple no`des. MongoDB infrastructure takes care of the database distribution and the read and write operations being maintained on the various level so that a single node is not bottlenecked, and the system is highly scalable and can handle different variety, velocity of data.

There are also options of growing instances horizontally, but it is not normally recommended. Also, the options to choose different volumes of data along with each node is available. Below are the various plans and options of MongoDB fully compatible with the Amazon Web Services: -

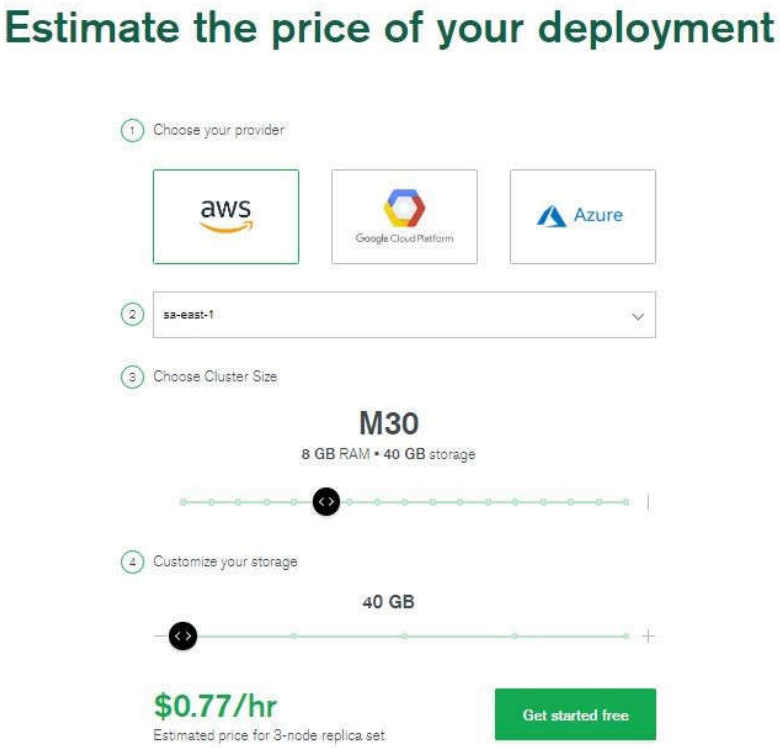
There are various plans like free to basic that differ in the features and the option they provide. The basic features start with scalable RAM and a lot of other customizable options that are available to optimize the performance to the storage as well as the security features of the public cloud being deployed are as follows.

Cluster Customization				
Highly available clusters	●	●	●	●
Uptime SLA		●	●	●
Elastic scalability		●	●	●
Cross-region replication		●	●	●
Global clusters		●	●	●
Security				
Always-on authentication	●	●	●	●
IP whitelists	●	●	●	●
End-to-end encryption	●	●	●	●
Dedicated private network		●	●	●
Enterprise security features		Available as add-on	Available as add-on	●

The figure above shows the various options about the cluster optimization that differ in terms of the custom plan being selected and the features being included in one of those plans. The platform excels in the fact that it includes most of the security features for all the plans except the free plan. The Enterprise level plan includes the enterprise security features and other features like end-to-end encryption and dedicated private network.

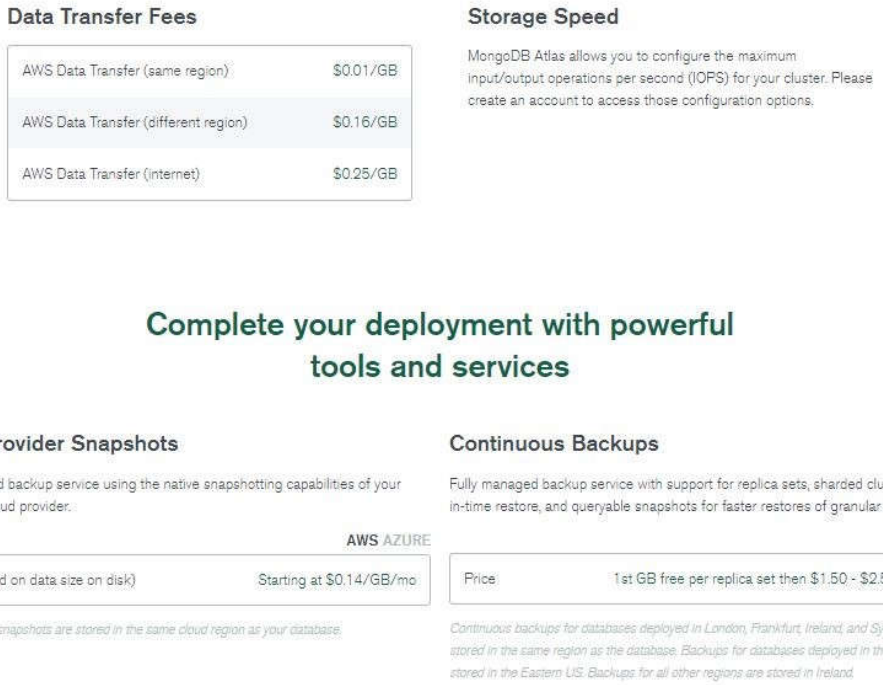
Performance Optimization				
Monitoring and alerts	●	●	●	●
100+ metrics dashboard		●	●	●
Real-time performance panel		●	●	●
Performance advisor		●	●	●
Disaster Recovery				
Fully managed backups		●	●	●
Point-in-time recovery		●	●	●
Queryable snapshots		●	●	●
Support & Learning				
Platform and UI support	●	●	●	●
End-to-end database support	30 days free	30 days free	●	●
Response time SLA			2 hrs	1 hr
On-demand access to MongoDB University				●
Additional Software & Tools				
MongoDB Compass			●	●
MongoDB Connector for BI		Available as add-on	Available as add-on	●

As shown in the above figure there are also other features that can make public cloud database platform selection an easy choice with more additional support and learning features.



There is a proper tool available to estimate the price of service deployment as per the requirement selected in the MongoDB Atlas and it is separate from the pricing of hosting and integration in the Amazon Cloud Services AWS.

Moreover, the price for inclusion of the replica set in terms of the velocity, variety, volume of data as the system and the project being deployed requires.



As shown in the figure above there are additional functionalities that can be deployed like continuous backups which have additional cost of their own.

5. Amazon RDS: Amazon RDS for relational data handling can be used to setup cost-efficient and resizable capacity while performing administrative tasks such as hardware provisioning, database setup, patching and backup. For this system, RDS service for Mongo DB has been adopted for the optimization of memory, I/O operations. AWS database migration service is used to replicate and easily mitigate or replicate data in Amazon RDS. RDS also helps in running the database instances in Amazon Virtual Private Cloud (Amazon VPC), which helps in isolating the database instances and to connect the existing IT infrastructure through industry-standard encrypted IPsec VPN.

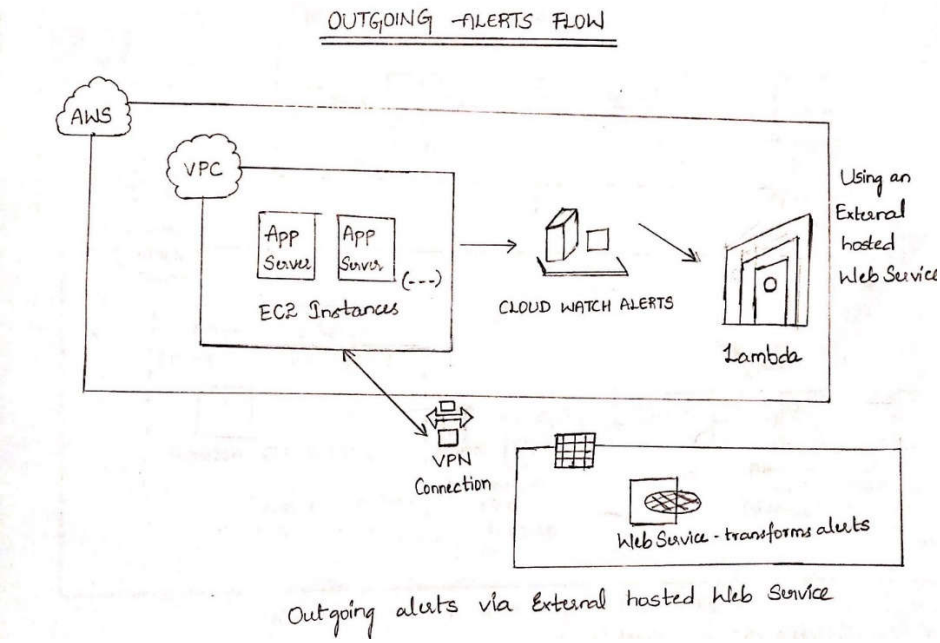
The system uses Amazon RDS since it simplifies much of the time-consuming administrative tasks typically associated with databases. Our system is expected to use Multi- Availability Zone (Multi-

AZ) deployment to further automate database replication and augmented durability of the data. The down time also can be much reduced for any expected migrations in the future.

6. Outgoing alerts: The outgoing alerts are designed using externally hosted web service that transforms alerts and sends the notifications to the customers via SMS adapter, PagerDuty, Slack and Amazon cloud watch alerts. The design flow for the same is as follows. Multiple application servers with multiple EC2 instances connected via VPC are connected to follow the alerting procedure to cloud watch alerts and Lambda over AWS VPC connection which intern using VPN connection, redirects the alerts to the web services to transform them.

External alerts are sent via webservice to the customers while the internal alerting is monitored by CloudWatch which monitors and manages the data and actionable insights to understand and respond to the system-wide performance changes, optimize resource utilization. It can be used to set high resolution alarms, visualize logs and metrics side-by-side.

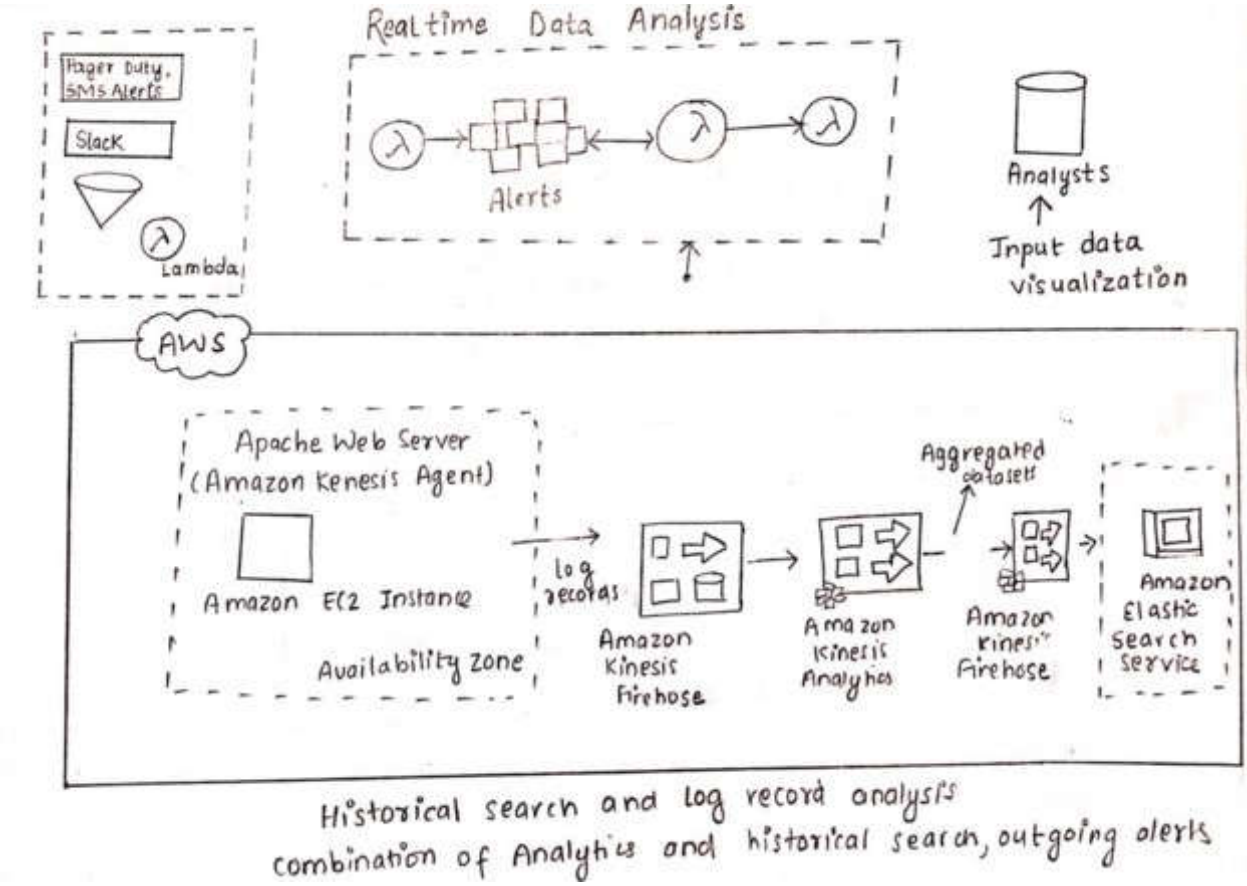
Slack is a communication channel for internal communication between the teams and developers.



7. Analytics and Historical Search:

Based on the customer needs they need to be provided with the recommendations and suggestions if they are willing to make any reservations in the future. For any instance if they are planning for booking with us, they would need to be marketed with existing offers and availabilities. Here comes the analytics and historical search into picture.

The design for the same is as follows:



The real time data analysis is handled by the lambda function. The output for the analysts in the form of visualizations or the transformed data is received by handling the data at EC2 instances by the amazon kinesis agent. The logs are sent to the amazon kinesis firehouse which is an easy way

to load streaming data into AWS. This is again sent to the amazon kinesis analytics which can read application data continuously and process stream data in real time.

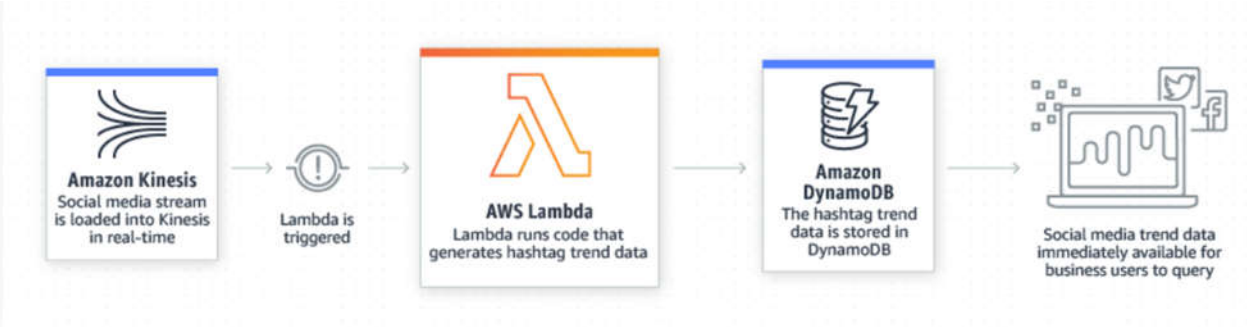


Figure 4 Figure showing Amazon Kinesis flow of processing real-time data

The application code can be written using SQL to process incoming streaming data and produce the output. We have planned to write the application output to a custom destination instead of Amazon S3 or Amazon Redshift. Amazon Lambda is configured to receive the stream data as input. In Lambda code, the kinesis data delivery stream to write results to Elastic search service (Amazon ES).

8. Payment system: The entire payment system is handled in Mainframe applications. The transactions are managed in the DB2 data bases and the communication is handled using the message queue services. The service in use to establish this connection is the enterprise platform as a service.

9. Geo Location: Geo location feature has been included to provide location specific services while using the mobile app or for the web-based user interface. The service has been provided using google maps service.

Back up services: For efficient backup and service restoration in case of disasters, the application is set up at 3 Availability zones. Each region is completely independent, and each availability zone is isolated. Elastic IP addresses are used to mask the failure of an instance in one availability zone by rapidly remapping the address to an instance in another availability zone.

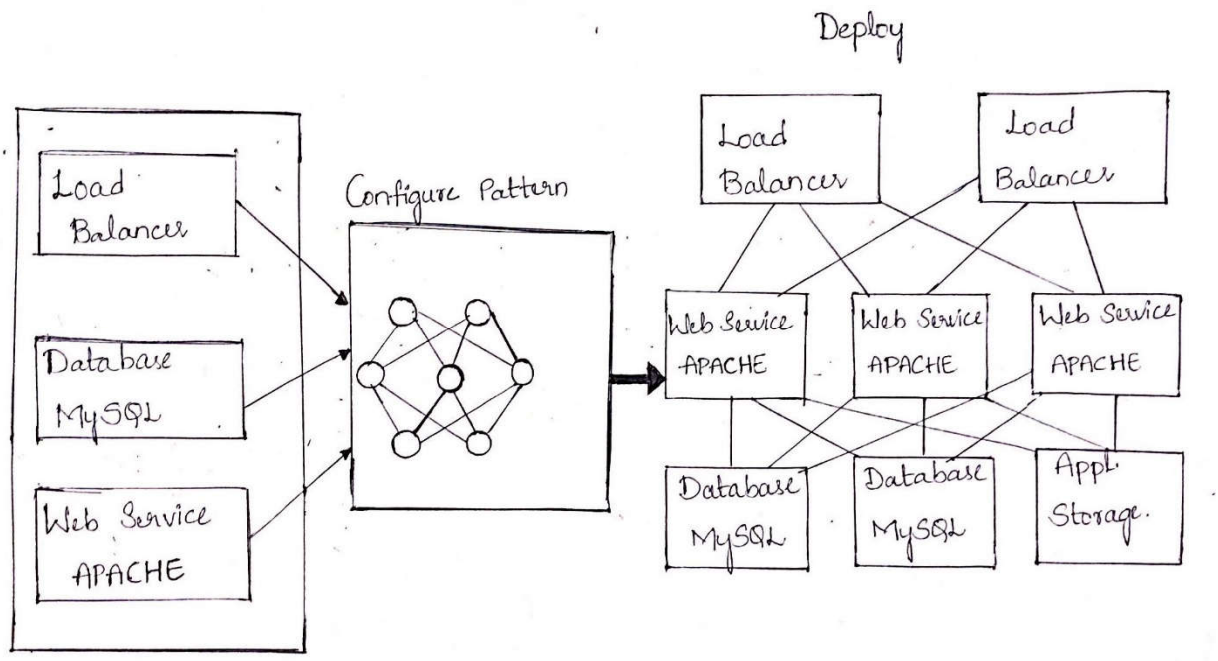
Application data from websites, mobile applications is stored as objects/buckets and protected in **Amazon S3**. This application houses backup data and static data files on Amazon S3. Restores and retrieve the lost data.

Technical Procedure for the Deployment of the public cloud:

- A load balancer, Web server, and database server appliances should be selected from a library of preconfigured virtual machine images.
- Configuring each component to make a custom image should be made. Load balancer is configured accordingly; web server should be populated with the static contents by uploading them to the storage cloud whereas the database servers are populated with the dynamic content of the site.
- The developer, as in we then feeds the custom code in to the new architecture making components meet their specific requirements.
- The developer chooses a pattern that takes the images for each layer and deploys them, handling networking, security, and scalability issues.

- The secure, high-availability Web application will be up and running. When the application needs to be updated, the virtual machine images can be updated, copied across the development chain, and the entire infrastructure can be redeployed.
- In this example, a standard set of components can be used to quickly deploy an application. With this model, enterprise business needs can be met quickly, without the need for the time-consuming, manual purchase, installation, cabling, and configuration of servers, storage, and network infrastructure.

The deployment tools AWS CodeDeploy helps in managing the deployments. It automates the variety of compute services.



Cluster support functionalities:

For the application to provide cost-effective, fast and analytics at high scale, our proposed application uses Amazon EMR. Amazon EMR provides a managed Hadoop framework to process data across dynamically scalable Amazon EC2 instances. We can run MapReduce in EMR and

interact with data in other AWS data stores such as Amazon S3 and Amazon DynamoDB. EMR securely and reliably handles a broad set of big data use cases, including log analysis, web indexing and ETL part. Amazon EMR also includes EMRFS, a connector allowing Hadoop to use Amazon S3 as a storage layer. The Amazon EMR cluster is scalable in that we can size the cluster for CPU and memory required for our workloads instead of having extra nodes in the cluster to maximize on-cluster storage.

HDFS is automatically installed with Hadoop on the Amazon EMR cluster, and hence we would be using HDFS along with Amazon S3 to store your input and output data.

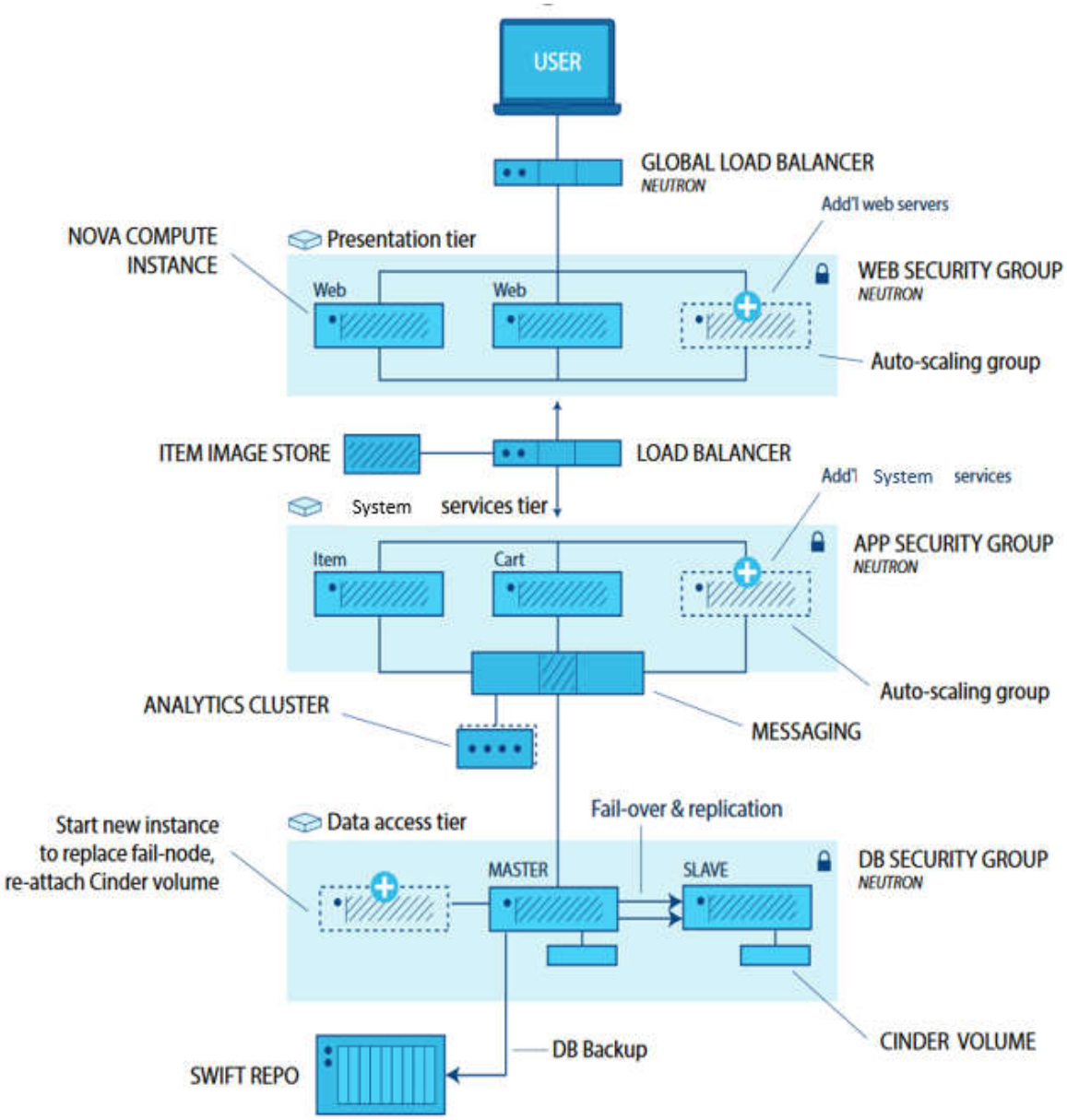
By default, Amazon EMR uses YARN (Yet Another Resource Manager) to centrally manage cluster resources for multiple data-processing frameworks. EMR supports SPARK cluster framework that uses directed acyclic graphs for execution plans and in-memory caching for datasets. When we run Spark on Amazon EMR, we can use EMRFS to directly access your data in Amazon S3. Spark supports multiple interactive query modules such as SparkSQL.

Private cloud:

Private cloud can be on-or-off-premises. Openstack helps in providing a modular framework with components that manage resources for computing, networking, storage and user interface functionalities.

The point of this model is the customer experience drives the need for dynamic content generation at a scale. This leads to a service-oriented architecture (SOA) for the applications. Each customer view is a result of collaboration between many services. To illustrate, when a customer is using the search box, the application is talking to one service, the images on the page are coming from another service, the related product recommendations are coming from yet another service, the item details from another one, and so on.

To have our own system (DIY) which provides unlimited flexibility and customization for the proposed online reservation system, the architecture is as follows:



The architecture and working of the application i.e. the functionalities are same however the components used are different. The global load balancing is handled by neutron, there are multiple NOVA compute instances of web connected, protected by WEB SECURITY GROUP where the security features are editable and modifiable.

The Messaging, load balancing, auto-scaling and analytics support has been added as in public for providing the same functionalities.

web presentation tier:

The web presentation tier holds cluster of web servers that renders the content generated for the user. The dashboard provides the web-based interface. Horizon is being used currently in our enterprise-based system for launching virtual machine instances, managing the networking part, viewing the size and current state of OpenStack cloud deployment. The horizon here provides three versions of dashboard management versions namely, a system dashboard, a user dashboard and a setting dashboard.

The Nova here provides a way to provision compute instances. It aids in the provisioning of compute instances. It supports the creation of virtual machine, BareMetal servers, and has limited support for system containers. Nova runs as a set of deamons on the top of existing Linux servers to provide the service.

Web security group helps in providing the Identity services. Keystone provides API authentication and high-level authentication. It holds the central directory of users who can make modifications to the code and acts as a common authentication system across the cloud operating system. All CRUD operations associated with the data managed by the identity services supports token-based authN and user-service authorization.

Openstack Neutron delivers Networking as a Service (NaaS) to virtual computing environments. Neutron address the issues of tenant control in multi-tenant environments. Helps tenants in creating their own multiple private networks and control the IP address. API extension helps the organizations for additional control over security, QoS and troubleshooting.

Services tier:

This tier consists of the services that implement the business logic functionality; some visible to the end user and some not. Each service focuses on its core competency, like search, cart, checkout, recommendation, item, etc. The services interact with each other using the published web service API.

Database tier – The processing done in an application is totally dependent on the flow of data and the user and merchant insert and retrievals. The database layer is responsible for providing the storage and retrieval mechanisms for data. The object storage is featured by SWIFT and the Block storage by CINDER.

Openstack block storage Swift is a redundant scalable data storage where objects and files can be written to multiple disk drives. This is spread throughout the servers in the data center to ensures data distribution and replication across the cluster. The data distribution and replication in various devices which help is cost-reduction and scale-out storage. It helps organizations to store data safely and efficiently.

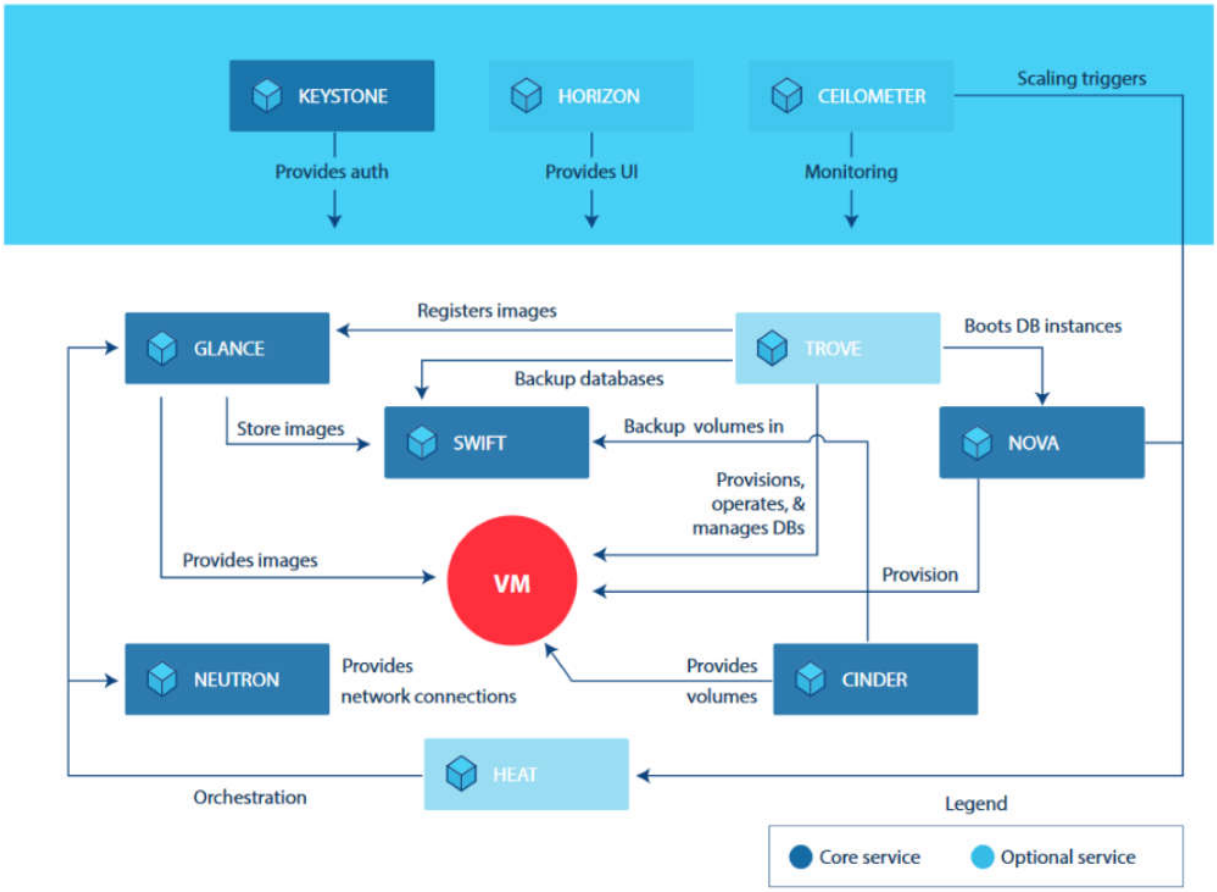
CINDER is blocks storage service that can be consumed by NOVA. It helps in permitting the organization to make a catalogue of block-based storage available to the devices with varied characteristics. It acts like an abstraction for the storage management.

Messaging: The API for each service provides a synchronous communication mechanism. There is also a need for an asynchronous communication mechanism provided by messaging.

Analytics: When the workload meets the criteria for big data: volume, variety, and velocity. The analytics layer is responsible for ensuring the transformation of the raw logs, metrics, and meters into information and insights, which are then stored in databases, and consumed by services online (customer interaction) or offline (business decisions).

Payment and billing: OpenStack Telemetry Ceilometer delivers a single point of contact for billing system. It is used to collect measurements of the utilization of the physical and virtual resources which consists of deployed cloud. Developers can collect the data and configure the type of data to meet their operating system requirements by monitoring notifications from existing services.

Orchestration Program- Heat is the main orchestration program in open stack. An orchestration engine is implemented to launch multiple composite cloud application that are based on templated in the form of text files which can also be treated like code. Those text files are readable and writable by humans and can be checked into version control. This helps in creating a human and machine accessible service for managing the complete lifecycle of infrastructure and applications within open stack clouds.

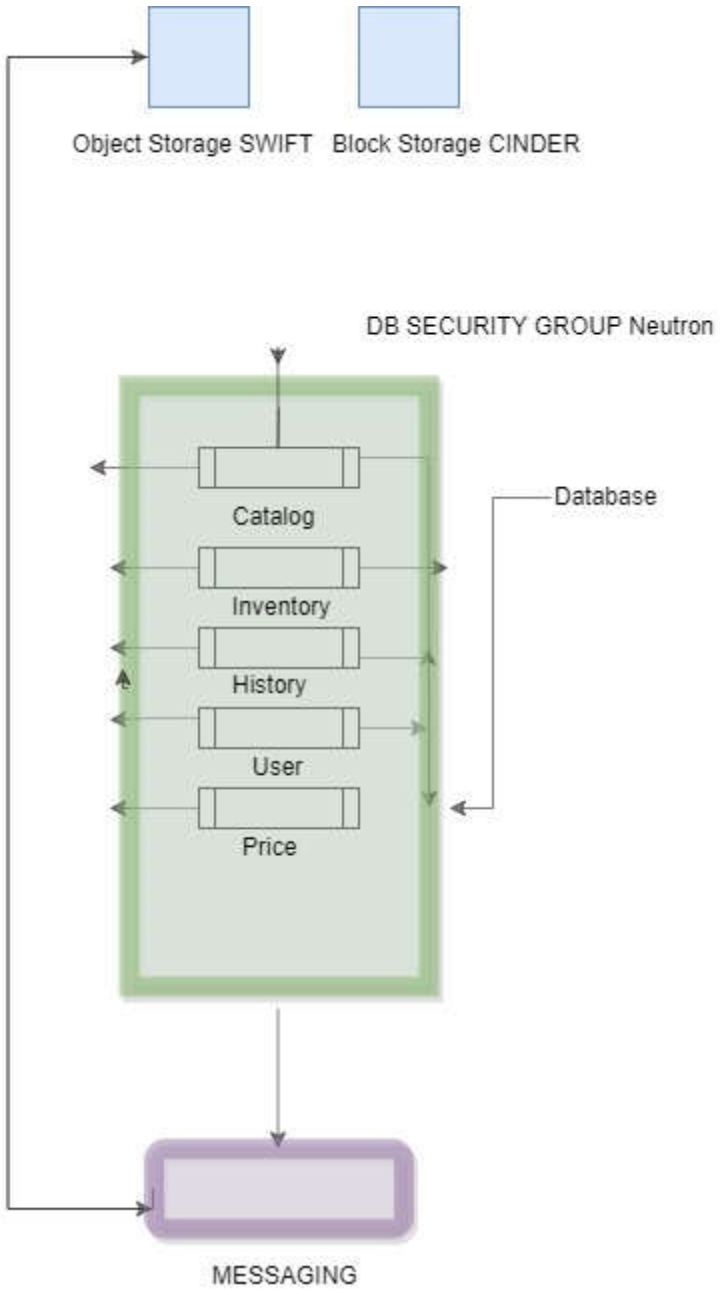


Each service provides a specific set of operations and the interconnections between them is as follows:

1. Keystone provides authentication for the User interface handled by Horizon. The Ceilometer Helps in monitoring and scaling triggers from ceilometer are directed to Heat service for Orchestration.
2. The Virtual Machine is connected to Glance, Swift, Trove, Nova, Cinder and Neutron for multiple purposes.
3. The register images from Trove goes to Glance which in turn adds store images to the Swift. Trove sends the Backup of the databases to swift.

4. The backup volumes, provisions like operations and management of databases is handled by Swift which in turn are sent to Virtual Machine.
5. Core services and Operational services are marked.

Database Architecture



Data Processing Services:

The data processing services for the open stack are handled with the help of Sahara service which aims to provide users with the means of data processing clusters i.e Hadoop and Spark clusters.

They specify several parameters like Hadoop version, cluster topology, node hardware details.

After all the user fills in all the required parameters, the cluster can be deployed in few minutes. Analytics-as- a-service is supported by the Sahara. The analytical queries are done with the help of Hive or Pig. However, for this application, we have adopted Hive.

Deployment of private cloud:

The deployment of private cloud can happen using managed Openstack services provided by third party vendors. For our proposed model, we are using ansible, where the deployment steps for the same are as follows:

Openstack-Ansible has a flexible deployment configuration model that can deploy all services in separate containers or on the designated hosts.

1. Firstly, the infrastructure components are installed. These include Memcached, Repository that holds the reference set of artifacts like python wheels, load balancer, utility container, Log aggregation host, Unbound DNS container.
2. Prepare the deployment host.
    - Install the operating system.
    - Configure operating system.
    - Configure the network.
    - Install the source and dependencies.
    - Configure SSH keys.
3. Prepare the target hosts

- Configuring the operating systems and storage.
- Network configuration

4. Authorize servers:

   Authorizing servers is simple. Just select what roles these servers should be assigned:

   Hypervisor: Can run virtual machine instances (Nova Compute)
   Image Library: Can serve VM images to hypervisors (Glance)
   Storage Endpoint: Can interface with block storage (Cinder)
   Network Endpoint: Can interface with networks (Nova or Neutron)

5. Import images and customize flavours:

   The Image Library can automatically discover existing VM disk images from its storage backing. Or, we can import pre-configured images that are maintained by Platform9.

6. Configure the deployment
   - Initial environment configuration
   - Service configurations

7. Configure service credentials like add users and go live:

   The final step is to bring on self-service users and define tenant groups. Users can be assigned roles within each tenant. You can also control which tenants have access to which VM images, flavors and networks.

We are now live in production with OpenStack. The maintenance activities will be handled. For any further deployments, continuous integration and build tools are utilized.

Hardware Requirements:

ADMIN NODE	1U server enclosure Single 8 core CPU processor 16 GB DDR-3 memory, 1600 MT/s 3 HDDs, 3 TBs each, SATA
STORAGE NODES (3)— CINDER / CEPH (Expandable to 4)	(3) Object Storage Device NODES 2U server enclosure Dual 12 core CPU processors 64 GB DDR-3 memory, 1600 MT/s Internal SFF HDDs (internal or rear-mounted) (2) 128GB SSD (OS—RAID1 mirror) External SFF HDDs (24) 1TB SATA SSD 2 x 10G and 2 x 1G Ethernet ports
CONTROLLER NODE (4)— Controller / RGW	1U server enclosure Single 8 core CPU processor 24 GB DDR-3 memory, 1600 MT/s 4 HDDs, 2 TBs each, SATA 2 x 10G and 2 x 1G Ethernet ports

COMPUTE NODES (12) (Expandable to 25)	1U server enclosure Dual 8 core CPU processors 256 GB DDR-3 memory, 1600 MT/s 6 1TB SATA SSD (data) 2 128GB SSD RAID1 (OS) 2 x 10G and 2 x 1G Ethernet ports
NETWORK	2 1Gb Managed Ethernet Switch (48P) IPMI network for server management (public) Management network for deployment and configuration (private) 2 10Ge Ethernet switch (48P) Cluster, application and data traffic (public)
RACK AND POWER	42U-19" rack Dual PDUs 208VAC single-phase inputs-L6-30 plugs

Software requirements:

Ensure that all hosts within an OpenStack-Ansible (OSA) environment meet the following minimum requirements:

- Ubuntu
    - Ubuntu 16.04 LTS (Xenial Xerus) or Ubuntu 18.04 LTS (Bionic Beaver)
    - Linux kernel version 3.13.0-34-generic or later is required.

- CentOS (support is experimental)
    - Centos 7, fully updated.
    - Linux kernel version 3.10.0 or later.
- openSUSE (support is experimental)
    - Leap 42.X, fully updated.
    - Linux kernel version 4.4.X or later.
- Secure Shell (SSH) client and server that support public key authentication
- Network Time Protocol (NTP) client for time synchronization (such as ntpd or chronyd)
- Python 2.7.*x*
- en_US.UTF-8 as the locale
- Compute hosts should have multicore processors with hardware-assisted virtualization extensions. These extensions provide a significant performance boost and improve security in virtualized environments.
- Infrastructure (control plane) hosts should have multicore processors for best performance. Some services, such as MySQL, benefit from additional CPU cores and other technologies, such as Hyper-threading.