



**G H Raisonni College of Engineering &
Management, Pune – 412 207**

Department of Computer Engineering

Lab Manual (2024-2025)

Class: TY B. Tech. Computer

Term: V

Cloud Computing Lab

G H Raisonni College of Engineering and Management, Pune 412207

Course Details

Course: Cloud Computing Lab-(UCOP305)

Class: T Y B.Tech. Division: A, B, C

Internal Marks: 25

Internal marks: NA

Credits: 2

Pattern: 2023

COURSE OUTCOME	
CO1:	Understand and analyze cloud services provided by AWS, Google cloud, Microsoft azure and open stack.
CO2:	Implement virtualization by creating and configuring virtual machine
CO3:	Design cloud simulator by using CloudSim simulator software
CO4:	Construct cloud application in PaaS environment like sales force, kubernetes, DevOps and MLOPS

List of Experiment

Sr. No.	Experiment List	CO Mapping
1	Explore AWS cloud based Iaas Service	CO1
2	Implement Virtualization using VirtualBox / VMware workstation	CO2
3	Explore Cloud Services using CloudSim Simulator	CO3
4	Explore Cloud Service using Google Cloud	CO3
5	Explore Cloud Service and table formation using Microsoft Azure Cloud	CO3
6	Creating a Warehouse Application in Salesforce.com Paas	CO4
7	Explore open-source cloud platform Open stack	CO4
8	Implement container management with Kubernetes	CO4
Content Beyond Syllabus		
9	Implement DevOps and MLOPS using Cloud	C04
10	Explore Fog Computing Framework	C04

Experiment. No: 1

Aim: Explore AWS cloud based IaaS Service

Theory:

AWS IaaS (Infrastructure as a service)



IaaS or Infrastructure as a service is one of Amazon Web Services that focuses on providing infrastructure services based on cloud computing technology. It's not difficult to name an organization that provides IaaS but among the top providers worldwide, AWS stands out as the front runner for IaaS cloud services. It has clients in 190 countries and 66 available Zones within 21 geographic regions. IaaS Amazon Service is used to replace physical resources, such as servers, with virtual resources hosted and managed by Amazon. System users can run any operating system or application on these leased servers, without incurring any extra fees for maintenance and operation.

. AWS IaaS Benefits

Availability of separate development environment

Hardware and operating system specifications for the service can be selected and used directly from the network

Allow expanding the resources of the server in terms of quantity and

functionality No errors or extra costs arise while upgrading the system

AWS IaaS Characteristics

Characteristics that define IaaS include:

- Resources are available as a service
- Cost depends on the consumption
- Highly scalable
- Allow multiple users to access a single piece of hardware
- The organization has complete control of the infrastructure
- Dynamic and flexible

When to use Amazon AWS IaaS

AWS IaaS is suitable for:

Startups and small companies which want to avoid spending time and money in producing hardware and software.

Larger companies which prefer complete control over their applications and infrastructure, but want to purchase only what is needed for the app development.

Companies with rapid growth potential that like to keep altering hardware and software as per their demands for scaling up.

AWS IaaS services Drawbacks

- Security threats which may arise from the host or other virtual machines
- Customers' inability to access their data when vendor outages happen
- Required team training to learn how to manage new infrastructure

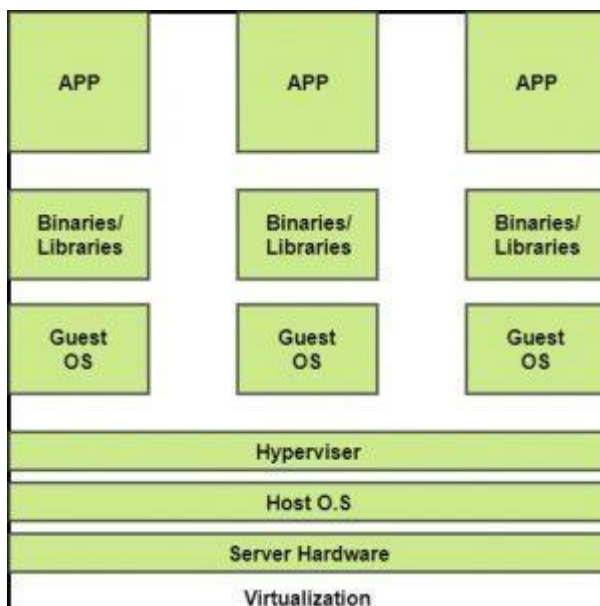
Experiment No. 2

Aim: Implement Virtualization using VirtualBox/ VMware workstation

Theory:

Virtualization is a technique of how to separate a service from the underlying physical delivery of that service. It is the process of creating a virtual version of something like computer hardware. It was initially developed during the mainframe era. It involves using specialized software to create a virtual or software-created version of a computing resource rather than the actual version of the same resource. With the help of Virtualization, multiple operating systems and applications can run on same machine and its same hardware at the same time, increasing the utilization and flexibility of hardware.

In other words, one of the main cost effective, hardware reducing, and energy saving techniques used by cloud providers is virtualization. Virtualization allows to share a single physical instance of a resource or an application among multiple customers and organizations at one time. It does this by assigning a logical name to a physical storage and providing a pointer to that physical resource on demand. The term virtualization is often synonymous with hardware virtualization, which plays a fundamental role in efficiently delivering Infrastructure-as-a-Service (IaaS) solutions for cloud computing. Moreover, virtualization technologies provide a virtual environment for not only executing applications but also for storage, memory, and networking.

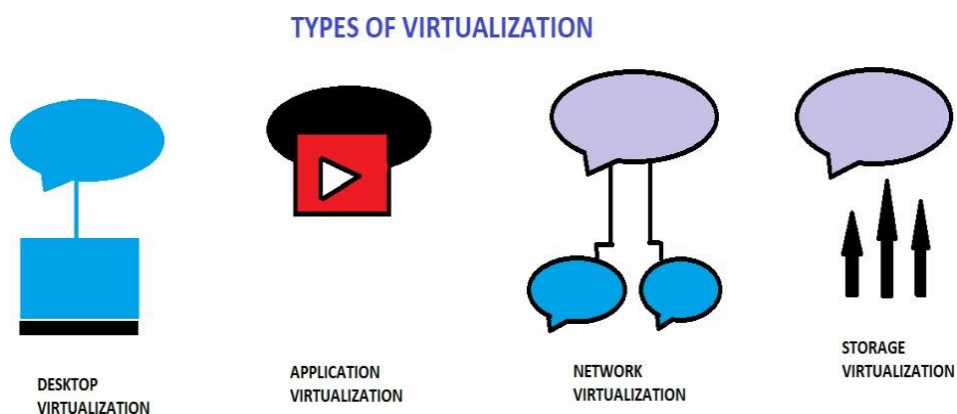


BENEFITS OF VIRTUALIZATION

1. More flexible and efficient allocation of resources.
2. Enhance development productivity.
3. It lowers the cost of IT infrastructure.
4. Remote access and rapid scalability.
5. High availability and disaster recovery.
6. Pay peruse of the IT infrastructure on demand.
7. Enables running multiple operating systems.

Types of Virtualization:

- 1.Application Virtualization.
- 2.Network Virtualization.
- 3.Desktop Virtualization.
- 4.Storage Virtualization.
- 5.Server Virtualization.
- 6.Data virtualization.



1. Application Virtualization:

Application virtualization helps a user to have remote access of an application from a server. The server stores all personal information and other characteristics of the application but can

still run on a local workstation through the internet. Example of this would be a user who needs to run two different versions of the same software. Technologies that use application virtualization are hosted applications and packaged applications.

2. Network Virtualization:

The ability to run multiple virtual networks with each has a separate control and data plan. It co-exists together on top of one physical network. It can be managed by individual parties that potentially confidential to each other.

Network virtualization provides a facility to create and provision virtual networks—logical switches, routers, firewalls, load balancer, Virtual Private Network (VPN), and workload security within days or even in weeks.

3. Desktop Virtualization:

Desktop virtualization allows the users' OS to be remotely stored on a server in the data centre. It allows the user to access their desktop virtually, from any location by a different machine. Users who want specific operating systems other than Windows Server will need to have a virtual desktop. Main benefits of desktop virtualization are user mobility, portability, easy management of software installation, updates, and patches.

4. Storage Virtualization:

Storage virtualization is an array of servers that are managed by a virtual storage system. The servers aren't aware of exactly where their data is stored, and instead function more like worker bees in a hive. It makes managing storage from multiple sources to be managed and utilized as a single repository. storage virtualization software maintains smooth operations, consistent performance and a continuous suite of advanced functions despite changes, break down and differences in the underlying equipment.

5. Server Virtualization:

This is a kind of virtualization in which masking of server resources takes place. Here, the central-server(physical server) is divided into multiple different virtual servers by changing the identity number, processors. So, each system can operate its own operating systems in isolate manner. Where each sub-server knows the identity of the central server. It causes an increase in the performance and reduces the operating cost by the deployment of main server resources into a sub-server resource. It's beneficial in virtual migration, reduce energy consumption, reduce infrastructural cost, etc.

6. Data virtualization:

This is the kind of virtualization in which the data is collected from various sources and managed that at a single place without knowing more about the technical information like how data is collected, stored & formatted then arranged that data logically so that its virtual

view can be accessed by its interested people and stakeholders, and users through the various cloud services remotely. Many big giant companies are providing their services like Oracle, IBM, At scale, Cdata, etc.

Experiment No: 03

Title: Explore Cloud Services using CloudSim Simulator

CloudSim is an open-source framework, which is used to simulate cloud computing infrastructure and services. It is developed by the CLOUDS Lab organization and is written entirely in Java. It is used for modelling and simulating a cloud computing environment as a means for evaluating a hypothesis prior to software development in order to reproduce tests and results.

For example, if you were to deploy an application or a website on the cloud and wanted to test the services and load that your product can handle and also tune its performance to overcome bottlenecks before risking deployment, then such evaluations could be performed by simply coding a simulation of that environment with the help of various flexible and scalable classes provided by the CloudSim package, free of cost.

Benefits of Simulation over the Actual Deployment:

Following are the benefits of CloudSim:

- **No capital investment involved.** With a simulation tool like CloudSim there is no installation or maintenance cost.
- **Easy to use and Scalable.** You can change the requirements such as adding or deleting resources by changing just a few lines of code.
- **Risks can be evaluated at an earlier stage.** In Cloud Computing utilization of real testbeds limits the experiments to the scale of the testbed and makes the reproduction of results an extremely difficult undertaking. With simulation, you can test your product against test cases and resolve issues before actual deployment without any limitations.
- **No need for try-and-error approaches.** Instead of relying on theoretical and imprecise evaluations which can lead to inefficient service performance and revenue generation, you can test your services in a repeatable and controlled environment free of cost with CloudSim.

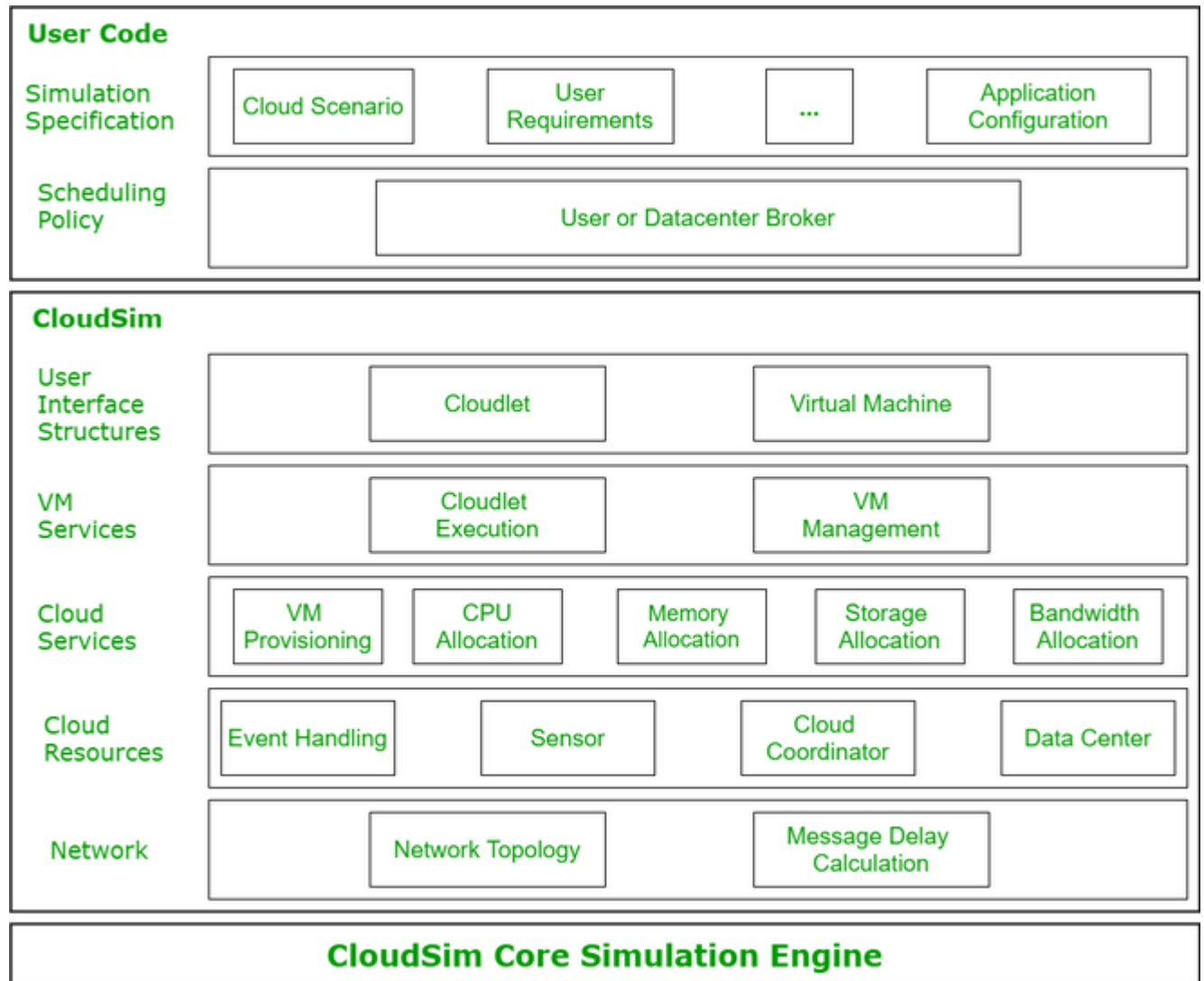
Why use CloudSim?

Below are a few reasons to opt for CloudSim:

- *Open source* and *free of cost*, so it favours researchers/developers working in the field.
- Easy to download and set-up.
- It is more *generalized* and *extensible* to support modelling and experimentation.
- Does not require any high-specs computer to work on.
- Provides *pre-defined allocation policies* and *utilization models* for managing resources, and allows implementation of user-defined algorithms as well.

- The documentation provides *pre-coded examples* for new developers to get familiar with the basic classes and functions.
- Tackle bottlenecks before deployment to reduce risk, lower costs, increase performance, and raise revenue.

CloudSim Architecture:



CloudSim Layered Architecture

CloudSim Core Simulation Engine provides interfaces for the management of resources such as VM, memory and bandwidth of virtualized Datacenters.

CloudSim layer manages the creation and execution of core entities such as VMs, Cloudlets, Hosts etc. It also handles network-related execution along with the provisioning of resources and their execution and management.

User Code is the layer controlled by the user. The developer can write the requirements of the hardware specifications in this layer according to the scenario.

Some of the most common classes used during simulation are:

- **Datacenter:** used for modelling the foundational hardware equipment of any cloud environment, that is the Datacenter. This class provides methods to specify the functional requirements of the Datacenter as well as methods to set the allocation policies of the VMs etc.
- **Host:** this class executes actions related to management of virtual machines. It also defines policies for provisioning memory and bandwidth to the virtual machines, as well as allocating CPU cores to the virtual machines.
- **VM:** this class represents a virtual machine by providing data members defining a VM's bandwidth, RAM, mips (million instructions per second), size while also providing setter and getter methods for these parameters.
- **Cloudlet:** a cloudlet class represents any task that is run on a VM, like a processing task, or a memory access task, or a file updating task etc. It stores parameters defining the characteristics of a task such as its length, size, mi (million instructions) and provides methods similarly to VM class while also providing methods that define a task's execution time, status, cost and history.
- **DatacenterBroker:** is an entity acting on behalf of the user/customer. It is responsible for functioning of VMs, including VM creation, management, destruction and submission of cloudlets to the VM.
- **CloudSim:** this is the class responsible for initializing and starting the simulation environment after all the necessary cloud entities have been defined and later stopping after all the entities have been destroyed.

Features of CloudSim:

CloudSim provides support for simulation and modelling of:

1. Large scale virtualized Datacenters, servers and hosts.
2. Customizable policies for provisioning host to virtual machines.
3. Energy-aware computational resources.
4. Application containers and federated clouds (joining and management of multiple public clouds).
5. Datacenter network topologies and message-passing applications.
6. Dynamic insertion of simulation entities with stop and resume of simulation.
7. User-defined allocation and provisioning policies.

Installation:

Prerequisites:

- Knowledge of Core Java language features such as [OOP](#) and [Collections](#).
- Basics of [Cloud Computing](#).
- CloudSim is available for download [here](#).
- For this tutorial, we have downloaded zip file of CloudSim 3.0.3.

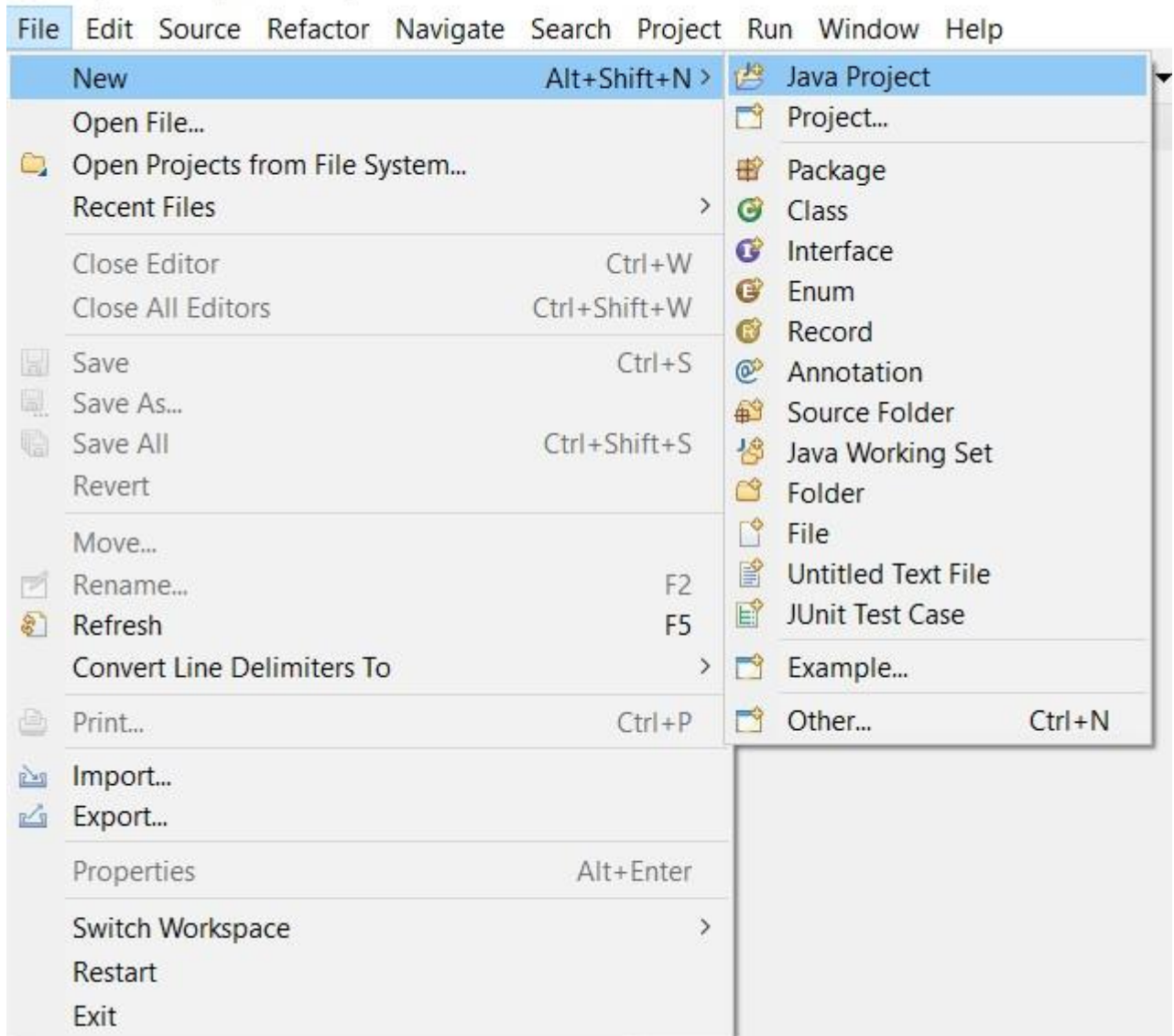
Note: CloudSim also uses some utilities of Apache's commons-math3 library. Download its Binaries zip file from [here](#).

Step 1: From the zip folder extracts *cloudsim-3.0.3* into a folder. Also, extract the *commons-math3-3.6.1 jar* into the same folder.



Step 2: Open Eclipse IDE and go to *File -> New -> Java Project*.

eclipse-workspace - Eclipse IDE



Step 3: Enter any name for your project and then uncheck the *Use default location* box just under it and click on *Browse*.

New Java Project

Create a Java Project

Enter a location for the project.

Project name: CloudSimTutorial

☐ Use default location

Location: [Browse...](#)

JRE

☒ Use an execution environment JRE: JavaSE-11

☐ Use a project specific JRE: jdk-11.0.9

☐ Use default JRE 'jdk-11.0.9' and workspace compiler preferences [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

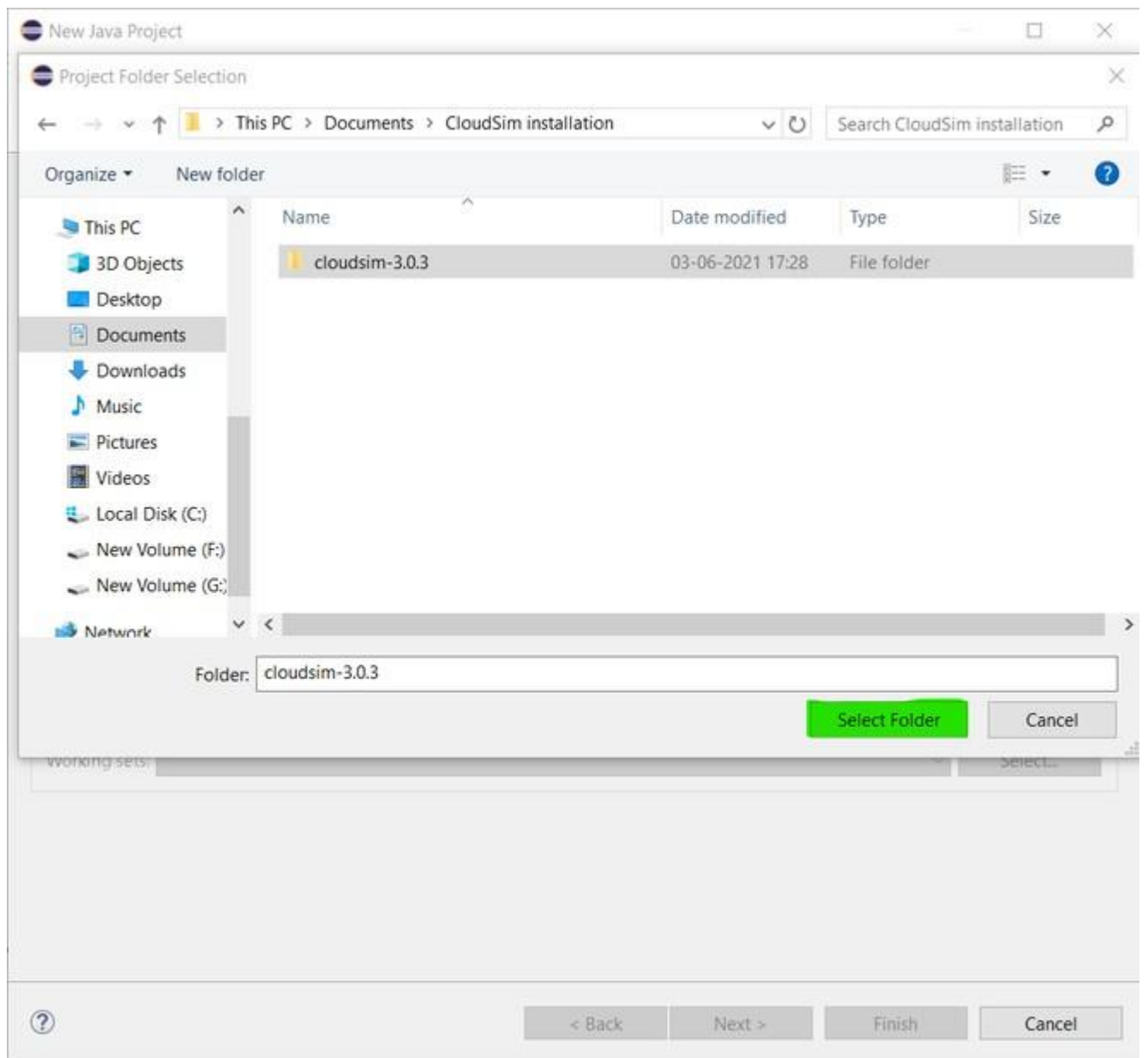
Working sets

☐ Add project to working sets [New...](#)

Working sets: [Select...](#)

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

Browse to the folder where you extracted your files and select the *cloudsim-3.0.3* folder.



Don't click on *Finish* yet, because we need to add a jar file to our project.

Step 4 Click *Next* and go to **Libraries -> Add External JARs**. Now browse to the same folder where you extracted your *commons-math3* jar file and *Open* it.

Java Settings

Define the Java build settings.



Source Projects Libraries Order and Export Module Dependencies

JARs and class folders on the build path:

- > cloudsim-3.0.3.jar - CloudSimTutorial/jars
- > cloudsim-3.0.3-sources.jar - CloudSimTutorial/jars
- > cloudsim-examples-3.0.3.jar - CloudSimTutorial/jars
- > cloudsim-examples-3.0.3-sources.jar - CloudSimTutorial/jars
- > JRE System Library [jdk-11.0.9]

Add JARs...

Add External JARs...

Add Variable...

Add Library...

Add Class Folder...

Add External Class Folder...

Edit...

Remove

Migrate JAR File...

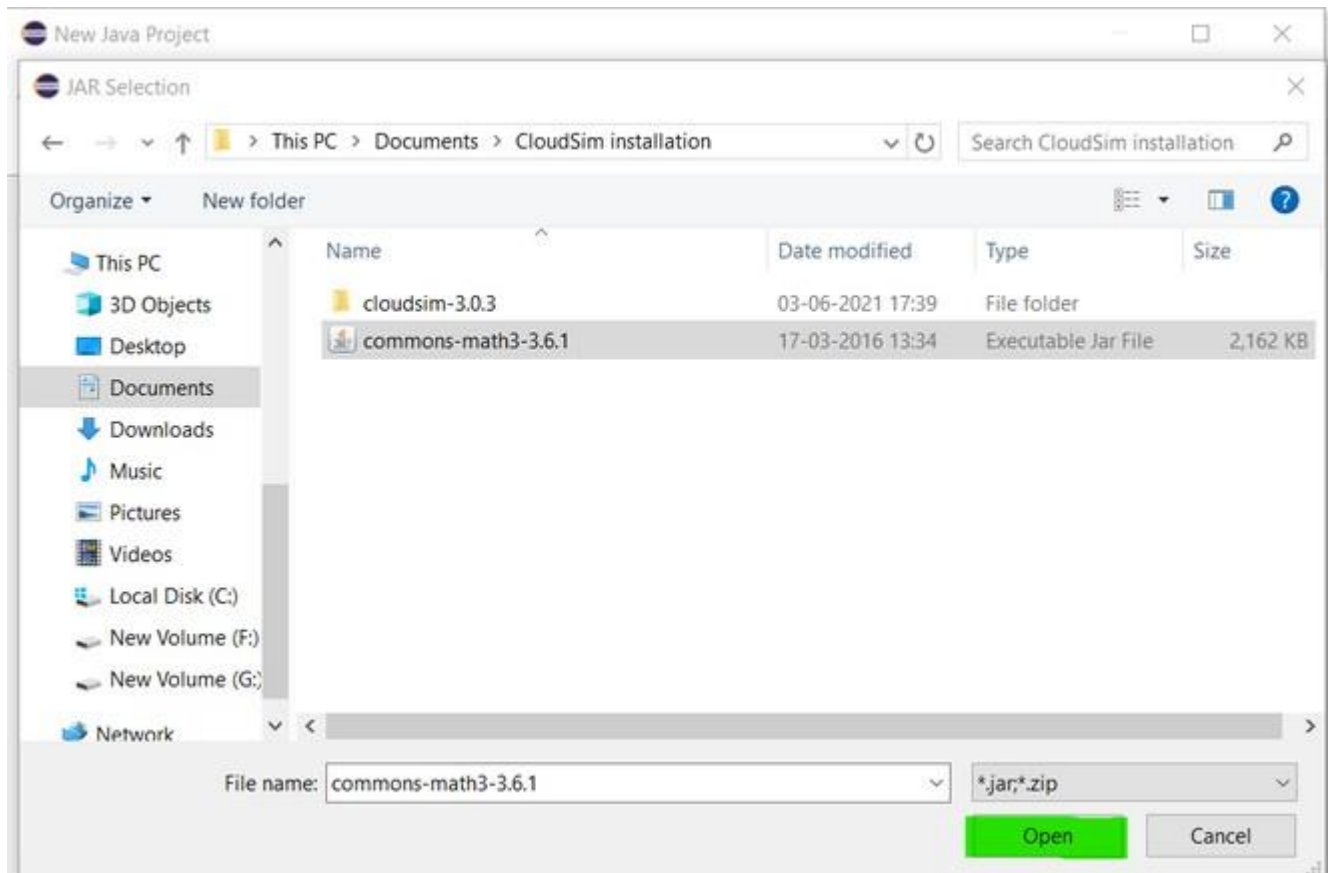


< Back

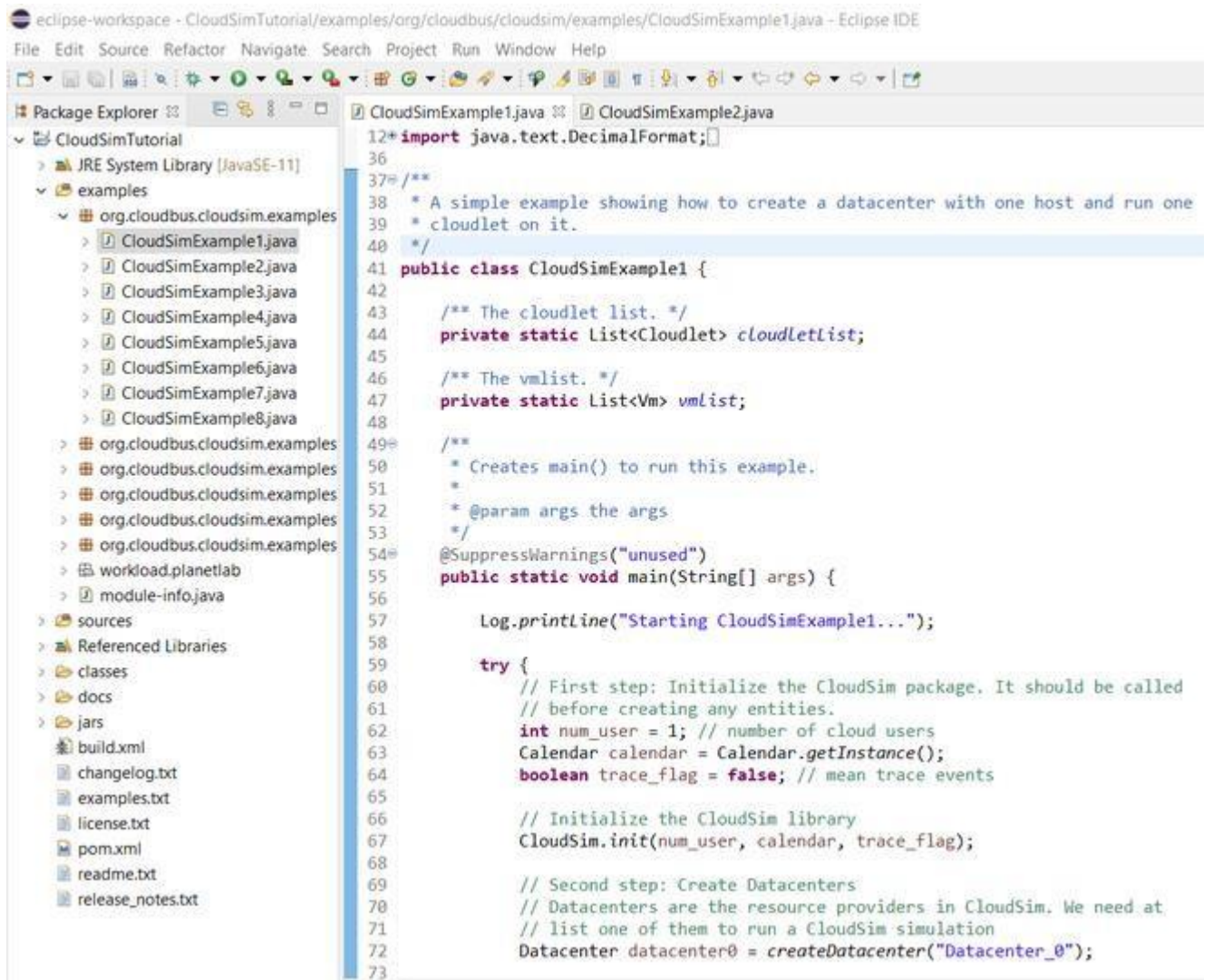
Next >

Finish

Cancel



Step 5 Finally click on *Finish* and wait for the project to build. After the project has been built, from the *Project Explorer* you can click on your project and from the dropdown go to *examples* -> *org.cloudbus.cloudsim.examples* where you can find pre-written sample codes and try to run them.



```
12*import java.text.DecimalFormat;[]
36
37= /**
38  * A simple example showing how to create a datacenter with one host and run one
39  * cloudlet on it.
40  */
41 public class CloudSimExample1 {
42
43     /** The cloudlet list. */
44     private static List<Cloudlet> cloudletlist;
45
46     /** The vmlist. */
47     private static List<Vm> vmlist;
48
49= /**
50  * Creates main() to run this example.
51  *
52  * @param args the args
53  */
54= @SuppressWarnings("unused")
55     public static void main(String[] args) {
56
57         Log.println("Starting CloudSimExample1...");
58
59         try {
60             // First step: Initialize the CloudSim package. It should be called
61             // before creating any entities.
62             int num_user = 1; // number of cloud users
63             Calendar calendar = Calendar.getInstance();
64             boolean trace_flag = false; // mean trace events
65
66             // Initialize the CloudSim library
67             CloudSim.init(num_user, calendar, trace_flag);
68
69             // Second step: Create Datacenters
70             // Datacenters are the resource providers in CloudSim. We need at
71             // list one of them to run a CloudSim simulation
72             Datacenter datacenter0 = createDatacenter("Datacenter_0");
73
```

Scope

With the flexibility and generalizability of the CloudSim framework, it is easy to model heavy cloud environments which would otherwise require experimentation on paid computing infrastructures. Extensible capabilities of scaling the infrastructure and resources to fit any scenario helps in fast and efficient research of several topics in cloud computing.

CloudSim has been used in several areas of research such as:

- Task Scheduling
- Green Computing
- Resource Provisioning
- Secure Log Forensics.

Conclusion: The cloudsim technology is supportive for the generation of several cloud applications with the creation of virtual machines and data centers. It is the easiest way for the analyzing process. There are many cloud simulators which is useful for the cloud research.

Experiment No: 04

Aim: Explore Cloud Service using Google Cloud

Theory:

1. Database Services

Teams can build and deploy faster, deliver transformative applications, and maintain portability and control of data with Google Cloud database.

The 5 Database Types are,

1. Relational
2. Key value
3. Document
4. In-memory
5. Additional NoSQL

Several SQL and NoSQL database services are provided by Google Cloud. A SQL database in Cloud SQL, which provides either MySQL or PostgreSQL databases.

A fully managed, mission-critical, relational database service in Cloud Spanner that offers transactional consistency at global scale, schemas, SQL querying, and automatic, synchronous replication for high availability.

Google Cloud provides 2 options for NoSQL data storage. They are,

1. Cloud Bigtable used for tabular data
2. Fire store used for document-like data

By using persistent disks, on Compute Engine, one can easily set up their preferred database technology. A MongoDB can be set up for NoSQL document storage.

Google Cloud Partner Services are MongoDB, Datastax, Redis Labs, and Neo4j.

2. Big Data Services

Big data services processes and query big data in the cloud to get fast answers to complicated questions. It is always time consuming and expensive to store and query huge datasets. And to do so without the right hardware and infrastructure support is a complete nightmare.

Google's BigQuery is a data analysis tool/service that helps users to create schemas to organize data into tables and datasets. The key factor about BigQuery is that it is fully managed, Hence the need to deploy resources is not required.

Enterprises (big data for enterprises) prefer BigQuery for storing and querying huge datasets. BigQuery with the power of Google's infrastructure enables super-fast SQL queries. With BigQuery, enterprises can control access to both the data and project based on your business needs.

BigQuery also accommodates multiple 3rd party tools such as visualizing the data or loading the data.

Advantages of BigQuery are,

1. Loads data (Includes Streaming data) from multiple sources
2. SQL-like commands are used to query massive datasets very quickly.
3. Designed and optimized for speed.
4. Use the web UI, command-line interface, or API.
5. Manage data and protect it by using permissions.

3. Networking Services

Kubernetes model is used by GKE whereas the app engine manages the networking. A set of networking services is provided by compute engine. These services help you to load-balance traffic across resources, create DNS records, and connect your existing network to Google's network, to balance the load and to create DNS records — these services are used.

We shall now investigate the main aspects of Google Kubernetes Engine networking. Application deploying, communication between applications, app communication with Kubernetes control panel — all can be defined through Kubernetes.

It is highly important and effective to change the way we think about the application's network design and their hosts while Kubernetes run the application. Instead of focusing on the host's or VMs connections, with Kubernetes in place, think about the communication between external clients, pods, and services.

Kubernetes along with Google Cloud configures IP filtering rules dynamically, routing tables, and firewall rules on each node. All these are completely dependent on the way the Kubernetes deployments are done and the way cluster configuration is done on Google Cloud.

4. Machine Learning Services

A variety of APIs is provided by Google Cloud helping you to enable and take advantage of Google's ML without creating and training your own models.

A few of the APIs are as follows:

1. Video Intelligence API for video analysis technology

2. Speech-to-Text API to convert audio to text
3. Cloud Vision API to integrate vision detection features
4. Cloud Natural Language API to add sentiment analysis
5. Cloud Translation API to translate the source text
6. Dialogflow API to build conversational interfaces for websites, mobile applications, popular messaging platforms, and IoT devices.

5. Storage Services

Google Cloud provides a variety of storage services such as,

1. Consistent
2. Scalable
3. Large-capacity data storage in Cloud Storage.

Cloud Storage comes in several flavours such as,

1. Standard Cloud Storage for maximum availability
2. Cloud Storage Nearline for low-cost archival storage
3. Cloud Storage Coldline for even lower-cost archival storage
4. Cloud Storage Archive for the lowest-cost archival storage

Filestore instances can be used to store data from applications running on Compute Engine VM instances or GKE clusters.

6. Computing & Hosting Services

Computing and hosting options from Google Cloud are as follows:

1. Serverless environment
2. Managed application platform.
3. Leverage container technologies
4. Build own cloud-based infrastructure

7. Serverless computing :

Google Cloud's functions provides a serverless environment for building and connecting cloud services. A Simple, single-purpose functions can be written using Cloud Functions that are attached to events emitted from the cloud infrastructure and services.

8.Application platform:

App Engine is Google Cloud's platform as a service (PaaS). Google handles the management of the resources with App engine.

9.Containers:

With container-based computing, the focus can be only on the application code. Google Kubernetes Engine (GKE), Google Cloud's containers as a service (CaaS) offering, is built on the open source Kubernetes system providing the flexibility of on-premises or hybrid clouds, in addition to Google Cloud's public cloud infrastructure.

Experiment No: 05

Title: Explore Cloud Service and table formation using Microsoft Azure Cloud

Theory:

What is Azure?

Azure is Microsoft's cloud platform, just like Google has its Google Cloud and Amazon has its Amazon Web Service or AWS.000. Generally, it is a platform through which we can use Microsoft's resource. For example, to set up a huge server, we will require huge investment, effort, physical space and so on. In such situations, Microsoft Azure comes to our rescue. It will provide us with virtual machines, fast processing of data, analytical and monitoring tools and so on to make our work simpler. The pricing of Azure is also simpler and cost-effective. Popularly termed as "Pay As You Go", which means how much you use, pay only for that.

Azure History

Microsoft unveiled Windows Azure in early October 2008 but it went to live after February 2010. Later in 2014, Microsoft changed its name from Windows Azure to Microsoft Azure. Azure provided a service platform for .NET services, SQL Services, and many Live Services. Many people were still very skeptical about "the cloud". As an industry, we were entering a brave new world with many possibilities. Microsoft Azure is getting bigger and better in coming days. More tools and more functionalities are getting added. It has two releases as of now. It's famous version **Microsoft Azure v1** and later **Microsoft Azure v2**. Microsoft Azure v1 was more like JSON script driven then the new version v2, which has interactive UI for simplification and easy learning. Microsoft Azure v2 is still in the preview version.

How Azure can help in business?

Azure can help in our business in the following ways-

- **Capital less:** We don't have to worry about the capital as Azure cuts out the high cost of hardware. You simply pay as you go and enjoy a subscription-based model that's kind to your cash flow. Also, to set up an Azure account is very easy. You simply register in Azure Portal and select your required subscription and get going.
- **Less Operational Cost:** Azure has low operational cost because it runs on its own servers whose only job is to make the cloud functional and bug-free, it's usually a whole lot more reliable than your own, on-location server.
- **Cost Effective:** If we set up a server on our own, we need to hire a tech support team to monitor them and make sure things are working fine. Also, there might be a situation where the tech support team is taking too much time to solve the issue incurred in the server. So, in this regard is way too pocket-friendly.
- **Easy Back Up and Recovery options:** Azure keep backups of all your valuable data. In disaster situations, you can recover all your data in a single click without your business getting affected. Cloud-based backup and recovery solutions save time, avoid large up-front investment and roll up third-party expertise as part of the deal.
- **Easy to implement:** It is very easy to implement your business models in Azure. With a couple of on-click activities, you are good to go. Even there are several tutorials to make you learn and deploy faster.
- **Better Security:** Azure provides more security than local servers. Be carefree about your critical data and business applications. As it stays safe in the Azure Cloud. Even,

in natural disasters, where the resources can be harmed, Azure is a rescue. The cloud is always on.

- **Work from anywhere:** Azure gives you the freedom to work from anywhere and everywhere. It just requires a network connection and credentials. And with most serious Azure cloud services offering mobile apps, you're not restricted to which device you've got to hand.
- **Increased collaboration:** With Azure, teams can access, edit and share documents anytime, from anywhere. They can work and achieve future goals hand in hand. Another advantage of the Azure is that it preserves records of activity and data. Timestamps are one example of the Azure's record keeping. Timestamps improve team collaboration by establishing transparency and increasing accountability.

Microsoft Azure Services

Some following are the services of Microsoft Azure offers:

1. **Compute:** Includes Virtual Machines, Virtual Machine Scale Sets, Functions for serverless computing, Batch for containerized batch workloads, Service Fabric for microservices and container orchestration, and Cloud Services for building cloud-based apps and APIs.
2. **Networking:** With Azure you can use variety of networking tools, like the Virtual Network, which can connect to on-premise data centers; Load Balancer; Application Gateway; VPN Gateway; Azure DNS for domain hosting, Content Delivery Network, Traffic Manager, ExpressRoute dedicated private network fiber connections; and Network Watcher monitoring and diagnostics
3. **Storage:** Includes Blob, Queue, File and Disk Storage, as well as a Data Lake Store, Backup and Site Recovery, among others.
4. **Web + Mobile:** Creating Web + Mobile applications is very easy as it includes several services for building and deploying applications.
5. **Containers:** Azure has a property which includes Container Service, which supports Kubernetes, DC/OS or Docker Swarm, and Container Registry, as well as tools for microservices.
6. **Databases:** Azure has also includes several SQL-based databases and related tools.
7. **Data + Analytics:** Azure has some big data tools like HDInsight for Hadoop Spark, R Server, HBase and Storm clusters
8. **AI + Cognitive Services:** With Azure developing applications with artificial intelligence capabilities, like the Computer Vision API, Face API, Bing Web Search, Video Indexer, Language Understanding Intelligent.
9. **Internet of Things:** Includes IoT Hub and IoT Edge services that can be combined with a variety of machine learning, analytics, and communications services.
10. **Security + Identity:** Includes Security Center, Azure Active Directory, Key Vault and Multi-Factor Authentication Services.
11. **Developer Tools:** Includes cloud development services like Visual Studio Team Services, Azure DevTest Labs, HockeyApp mobile app deployment and monitoring, Xamarin cross-platform mobile development and more.

Difference between AWS (Amazon Web Services), Google Cloud and Azure

	AWS	Google Cloud	Azure
Technology	EC2 (Elastic Compute Cloud)	Google Compute Engine (GCE)	VHD (Virtual Hard Disk)
Databases Supported	AWS fully supports relational and NoSQL databases and Big Data.	Technologies pioneered by Google, like Big Query, Big Table, and Hadoop, are naturally fully supported.	Azure supports both relational and NoSQL databases, and Big Data, through Windows Azure Table and HDInsight.
Pricing	Per hour – rounded up	Per minute – rounded up (minimum 10 minutes)	Per minute – rounded up commitments (pre-paid or monthly)
Models	On demand, reserved, spot	On demand – sustained use	On demand – short term commitments (pre-paid or monthly)
Difficulties	Many enterprises find it difficult to understand the company's cost structure	Fewer features and services.	Less “enterprise-ready”
Storage Services	<ul style="list-style-type: none"> Simple Storage Service (S3) Elastic Block Storage (EBS) Elastic Block Storage (EBS) 	<ul style="list-style-type: none"> Blob Storage Queue Storage File Storage Disk Storage Data Lake Store 	<ul style="list-style-type: none"> Cloud Storage Persistent Disk Transfer Appliance
Machine Learning	<ul style="list-style-type: none"> Sage Maker Lex Polly And many more 	<ul style="list-style-type: none"> Machine Learning Azure Bot Service Cognitive Service 	<ul style="list-style-type: none"> Cloud Speech API Cloud Video Intelligence Cloud Machine Learning Engine And many more.

Conclusion:

In conclusion, cloud computing is recently new technological development that has the potential to have a great impact on the world. It has many benefits that it provides to its users and businesses.

Experiment No:6

Aim: Creating a Warehouse Application in SalesForce.com Paas

Theory:

Paas(Platform as a service):

Platform as a service (PaaS) is a complete development and deployment environment in the cloud, with resources that enable you to deliver everything from simple cloud-based apps to sophisticated, cloud-enabled enterprise applications. You purchase the resources you need from a cloud service provider on a pay-as-you-go basis and access them over a secure Internet connection.

The main offerings included by PaaS vendors are:

- Development tools
- Middleware
- Operating systems
- Database management
- Infrastructure

Different vendors may include other services as well, but these are the core PaaS services.

Development tools:

PaaS vendors offer a variety of tools that are necessary for software development, including a source code editor, a debugger, a compiler, and other essential tools. These tools may be offered together as a framework. The specific tools offered will depend on the vendor, but PaaS offerings should include everything a developer needs to build their application.

Middleware:

Platforms offered as a service usually include middleware, so that developers don't have to build it themselves. Middleware is software that sits in between user-facing applications and the machine's operating system; for example, middleware is what allows software to access

input from the keyboard and mouse. Middleware is necessary for running an application, but end users don't interact with it.

Operating systems:

A PaaS vendor will provide and maintain the operating system that developers work on and the application runs on.

Databases:

PaaS providers administer and maintain databases. They will usually provide developers with a database management system as well.

Infrastructure:

PaaS is the next layer up from IaaS in the cloud computing service model, and everything included in IaaS is also included in PaaS. A PaaS provider either manages servers, storage, and physical data centers, or purchases them from an IaaS provider.

Advantages of Paas:

Faster time to market PaaS is used to build applications more quickly than would be possible if developers had to worry about building, configuring, and provisioning their own platforms and backend infrastructure. With PaaS, all they need to do is write the code and test the application, and the vendor handles the rest.

One environment from start to finish: PaaS permits developers to build, test, debug, deploy, host, and update their applications all in the same environment. This enables developers to be sure a web application will function properly as hosted before they release, and it simplifies the application development lifecycle.

Price: PaaS is more cost-effective than leveraging IaaS in many cases. Overhead is reduced because PaaS customers don't need to manage and provision virtual machines. In addition, some providers have a pay-as-you-go pricing structure, in which the vendor only charges for the computing resources used by the application, usually saving customers money. However, each vendor has a slightly different pricing structure, and some platform providers charge a flat fee per month.

Ease of licensing: PaaS providers handle all licensing for operating systems, development tools, and everything else included in their platform.

Potential drawbacks of using PaaS:

Vendor lock-in: It may become hard to switch PaaS providers, since the application is built using the vendor's tools and specifically for their platform. Each vendor may have different architecture requirements. Different vendors may not support the same languages, libraries, APIs, architecture, or operating system used to build and run the application. To switch vendors, developers may need to either rebuild or heavily alter their application.

Vendor dependency:

The effort and resources involved in changing PaaS vendors may make companies more dependent on their current vendor. A small change in the vendor's internal processes or infrastructure could have a huge impact on the performance of an application designed to run efficiently on the old configuration. Additionally, if the vendor changes their pricing model, an application may suddenly become more expensive to operate.

Security and compliance challenges:

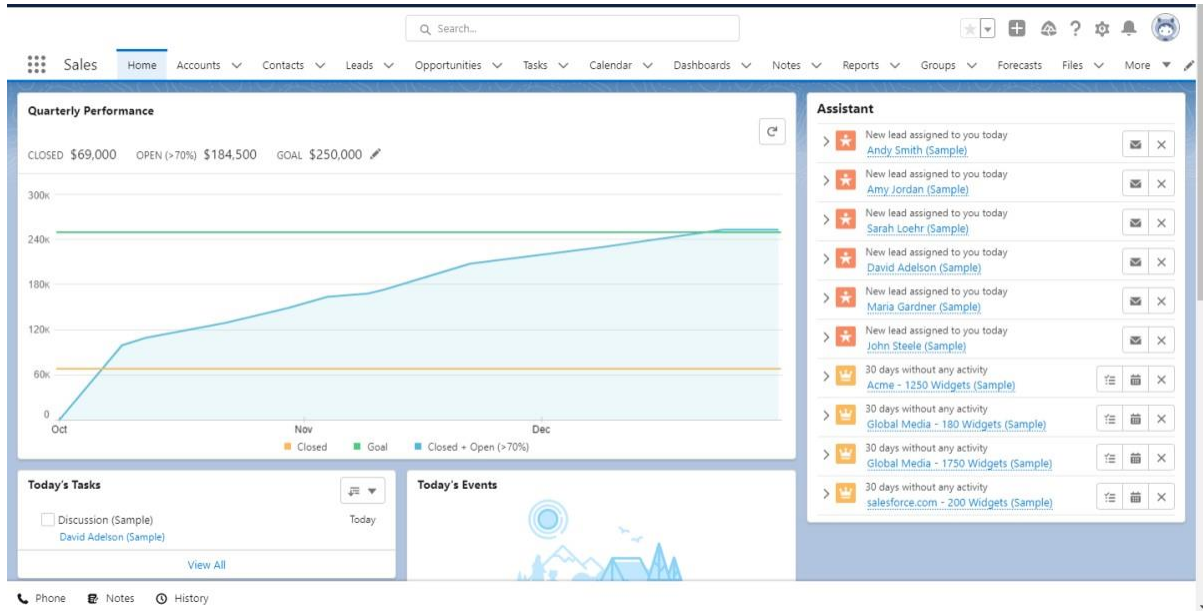
In a PaaS architecture, the external vendor will store most or all of an application's data, along with hosting its code. In some cases the vendor may actually store the databases via a further third party, an IaaS provider. Though most PaaS vendors are large companies with strong security in place, this makes it difficult to fully assess and test the security measures protecting the application and its data. In addition, for companies that have to comply with strict data security regulations, verifying the compliance of additional external vendors will add more hurdles to going to market.

What is salesforce?

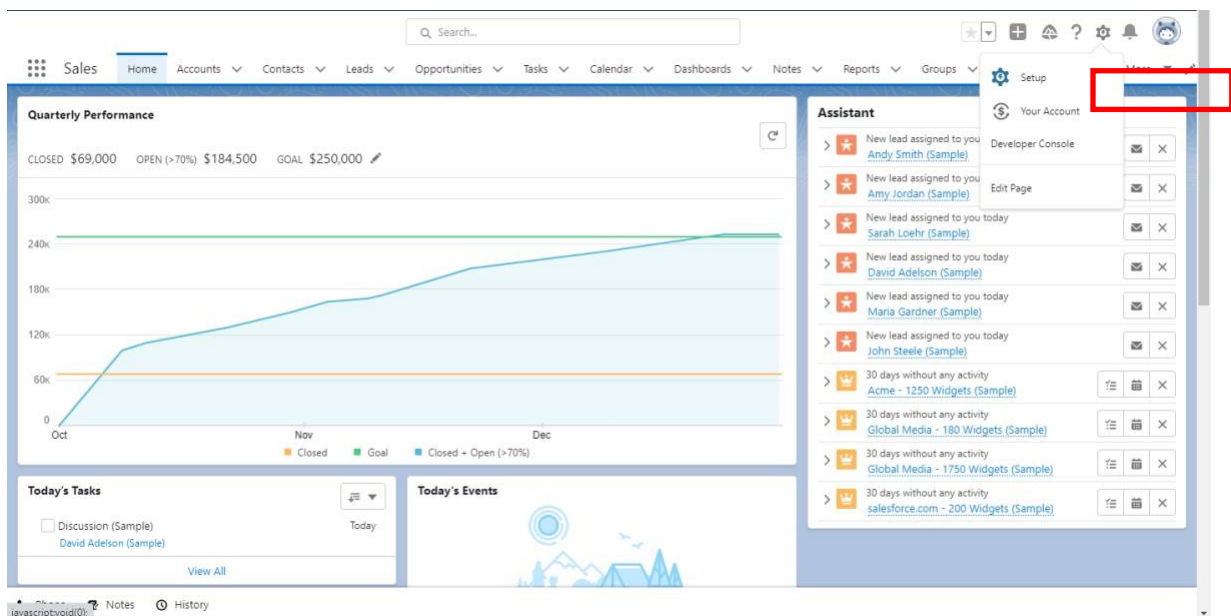
Salesforce is a company that makes cloud-based software designed to help businesses find more prospects, close more deals, and wow customers with amazing service. It also helps teams work better together. We can use a single Customer 360 app, or a combination of many. By improving team communications and productivity, business drive greater success..

Steps to create a warehouse application:

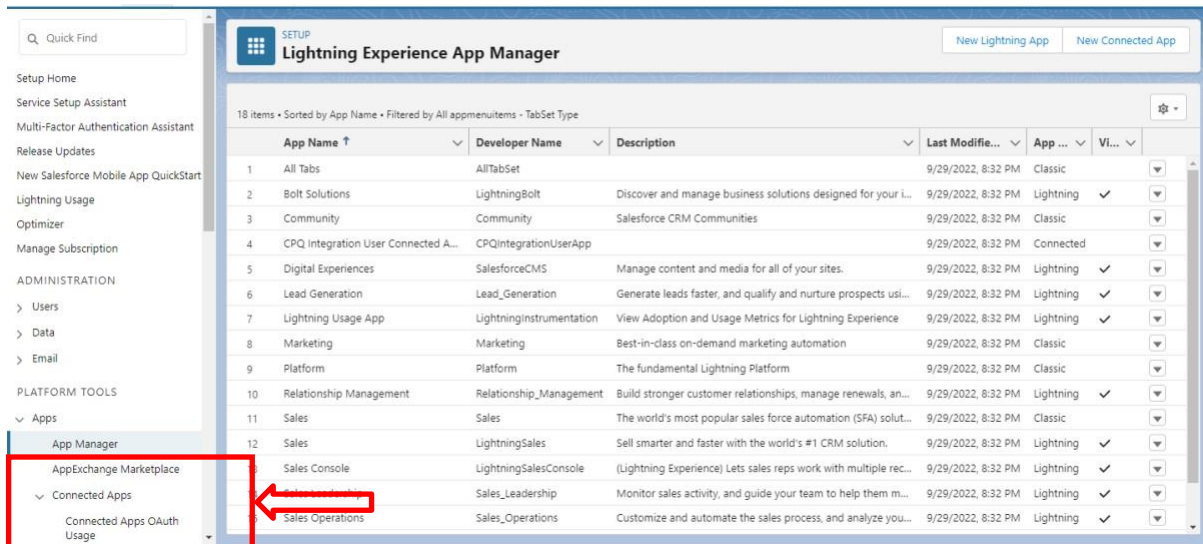
- 1) Go to Home page:



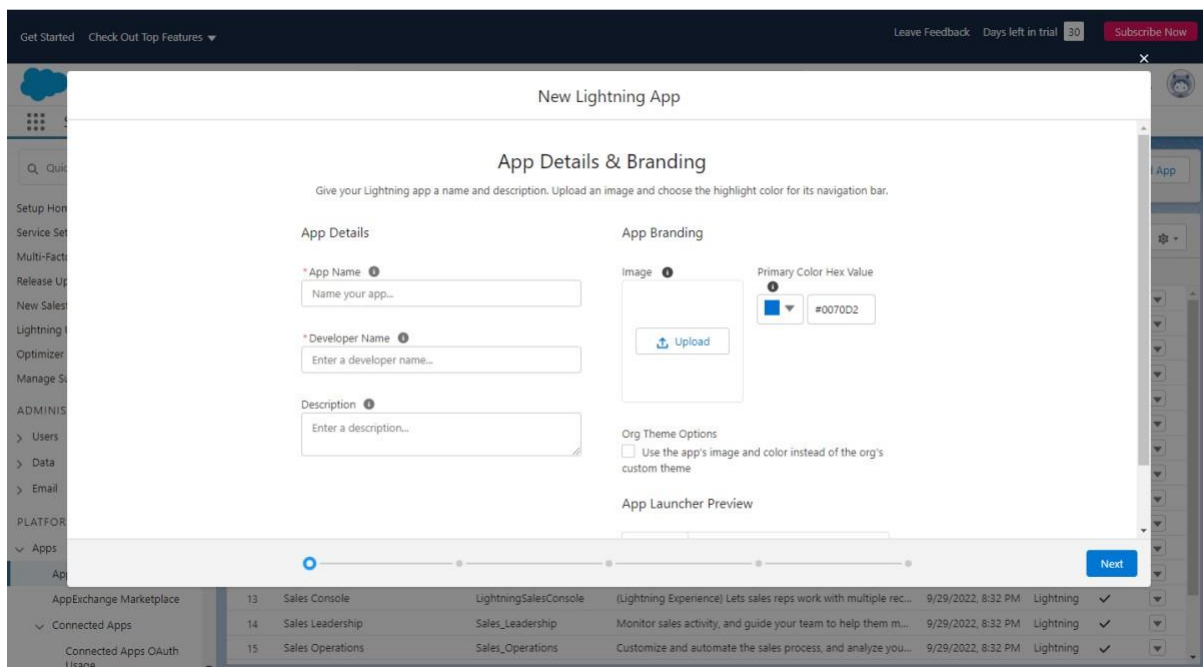
2) On the right hand side click on the setting and then setup:



3) On the left hand side click on the app manager.



4)



5)

Get Started | Check Out Top Features | Leave Feedback | Days left in trial: 30 | Subscribe Now

New Lightning App

App Details

* App Name ⓘ
Warehouse

* Developer Name ⓘ
Warehouse

Description ⓘ
This app is a warehouse app

App Branding

Image ⓘ

Primary Color Hex Value ⓘ
#0070D2

Org Theme Options
☒ Use the app's image and color instead of the org's custom theme

App Launcher Preview

This name appears in the navigation bar so your users can see the name of the app they're currently using.

Next

ID	Name	Label	Description	Created	Theme	Status
11	Sales	Sales	The world's most popular sales force automation (SFA) solution...	9/29/2022, 8:32 PM	Classic	
12	Sales	LightningSales	Sell smarter and faster with the world's #1 CRM solution.	9/29/2022, 8:32 PM	Lightning	✓
13	Sales Console	LightningSalesConsole	(Lightning Experience) Lets sales reps work with multiple records...	9/29/2022, 8:32 PM	Lightning	✓

6)

New Lightning App

App Options

Navigation and Form Factor ⓘ

* Navigation Style
☒ Standard navigation
☐ Console navigation

* Supported Form Factors
☒ Desktop and phone
☐ Desktop
☐ Phone

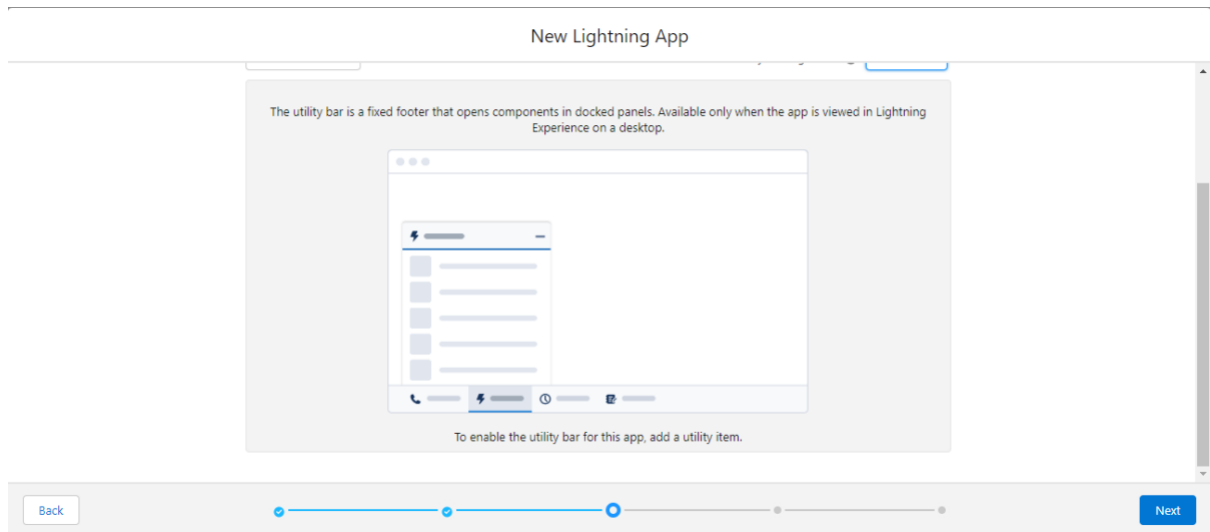
Setup and Personalization ⓘ

App Personalization Settings
☐ Disable end user personalization of nav items in this app
☐ Disable temporary tabs for items outside of this app

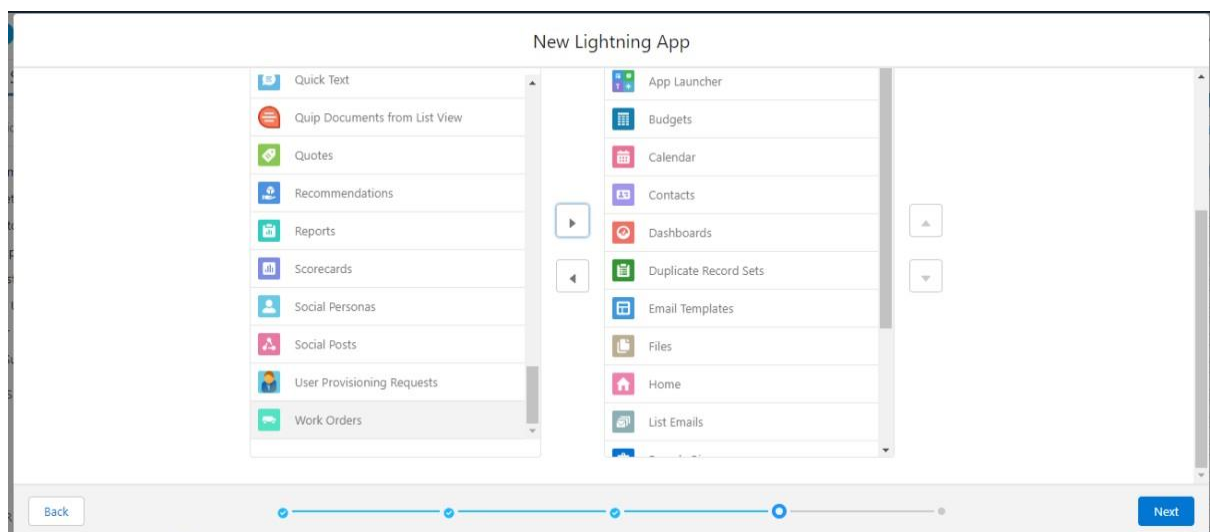
Back

Next

7)



8)



9)

New Lightning App

Type to filter list...

Contract Manager

CPQ Integration User

Executive Sponsor

Identity User

Marketing User

Minimum Access - Salesforce

Solution Manager

▶

◀

End User

Standard User

System Administrator

Read Only

Back

●

●

●

●

●

Save & Finish

Experiment No:7

Aim: Explore open-source cloud platform Open stack

Theory:

What is OpenStack?

OpenStack is an open source platform that uses pooled virtual resources to build and manage private and public clouds. The tools that comprise the OpenStack platform, called "projects," handle the core cloud-computing services of compute, networking, storage, identity, and image services. More than a dozen optional projects can also be bundled together to create unique, deployable clouds.

In virtualization, resources such as storage, CPU, and RAM are abstracted from a variety of vendor-specific programs and split by a hypervisor before being distributed as needed. OpenStack uses a consistent set of application programming interfaces (APIs) to abstract those virtual resources 1 step further into discrete pools used to power standard cloud computing tools that administrators and users interact with directly.

Working Of Open Stack:

OpenStack is essentially a series of commands known as scripts. Those scripts are bundled into packages called projects that relay tasks that create cloud environments. In order to create those environments, OpenStack relies on 2 other types of software:

- Virtualization that creates a layer of virtual resources abstracted from hardware.
- A base operating system (OS) that carries out commands given by OpenStack scripts.

The OpenStack components:

OpenStack's architecture is made up of numerous open-source projects. These projects are used to set up OpenStack's undercloud and overcloud—used by system admins and cloud users, respectively. Underclouds contain the core components system admins need to set up and manage end users' OpenStack environments, known as overclouds.

There are 6 stable, core services that handle compute, networking, storage, identity, and images while more than a dozen optional ones vary in developmental maturity. Those 6 core services are the infrastructure that allows the rest of the projects to handle dashboarding, orchestration, bare-metal provisioning, messaging, containers, and governance.

1) Nova:

Nova is a full management and access tool to OpenStack compute resources—handling scheduling, creation, and deletion.

2) Neutron:

Neutron connects the networks across other OpenStack services.

3) Swift:

Swift is a highly fault-tolerant object storage service that stores and retrieves unstructured data objects using a RESTful API.

4) Cinder:

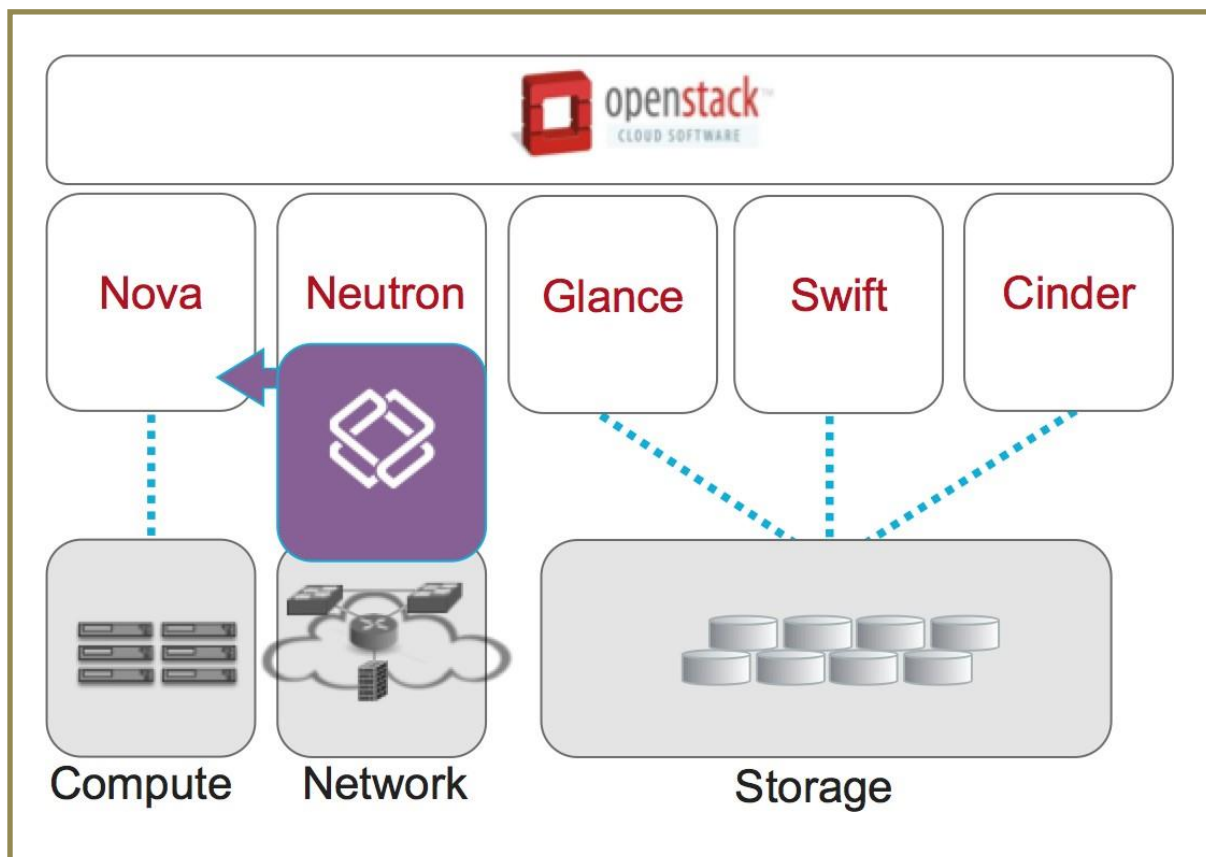
Cinder provides persistent block storage accessible through a self-service API.

5) Keystone:

Keystone authenticates and authorizes all OpenStack services. It's also the endpoint catalog for all services.

6) Glance:

Glance stores and retrieves virtual machine disk images from a variety of locations.



Applications of Open Stack:

i) Private clouds:

Private cloud distributions run on OpenStack can provide more substantial benefits than private clouds built using custom code. IDC evaluated the value of Red Hat OpenStack Platform for private clouds and found that organizations realized annual benefits of \$6.81 million.

ii) Network functions virtualization:

451 Research found that using OpenStack for network functions virtualization (NFV)—which involves separating a network's key functions so they can be distributed among environments—could very well be the next big thing. It's on the agenda of virtually every global communications services provider surveyed by the analyst.

iii) Public clouds

OpenStack is the leading open source option for building public cloud environments. Whether your company is a multibillion-dollar publicly traded enterprise or a startup, you can use OpenStack to set up public clouds with services that compete with major public cloud providers.

iv) Containers:

OpenStack is a stable foundation for public and private clouds. Containers speed up application delivery while simplifying application deployment and management. Running containers on OpenStack can scale containers' benefits from single, siloed teams to enterprise-wide interdepartmental operations.

Experiment No: 08

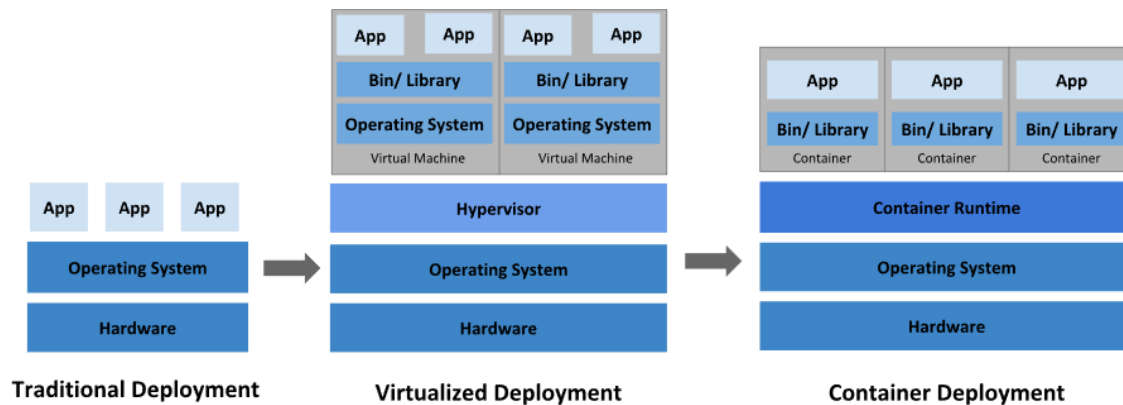
Aim: Implement container management with kubernetes.

Theory:

Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

The name Kubernetes originates from Greek, meaning helmsman or pilot. K8s as an abbreviation results from counting the eight letters between the "K" and the "s". Google open-sourced the Kubernetes project in 2014. Kubernetes combines over 15 years of Google's experience running production workloads at scale with best-of-breed ideas and practices from the community.

Let's take a look at why Kubernetes is so useful by going back in time.



Traditional deployment era: Early on, organizations ran applications on physical servers. There was no way to define resource boundaries for applications in a physical server, and this caused resource allocation issues. For example, if multiple applications run on a physical server, there can be instances where one application would take up most of the resources, and as a result, the other applications would underperform. A solution for this would be to run

each application on a different physical server. But this did not scale as resources were underutilized, and it was expensive for organizations to maintain many physical servers.

Virtualized deployment era: As a solution, virtualization was introduced. It allows you to run multiple Virtual Machines (VMs) on a single physical server's CPU. Virtualization allows applications to be isolated between VMs and provides a level of security as the information of one application cannot be freely accessed by another application.

Virtualization allows better utilization of resources in a physical server and allows better scalability because an application can be added or updated easily, reduces hardware costs, and much more. With virtualization you can present a set of physical resources as a cluster of disposable virtual machines.

Each VM is a full machine running all the components, including its own operating system, on top of the virtualized hardware.

Container deployment era: Containers are similar to VMs, but they have relaxed isolation properties to share the Operating System (OS) among the applications. Therefore, containers are considered lightweight. Similar to a VM, a container has its own filesystem, share of CPU, memory, process space, and more. As they are decoupled from the underlying infrastructure, they are portable across clouds and OS distributions.

Containers have become popular because they provide extra benefits, such as:

- Agile application creation and deployment: increased ease and efficiency of container image creation compared to VM image use.
- Continuous development, integration, and deployment: provides for reliable and frequent container image build and deployment with quick and efficient rollbacks (due to image immutability).
- Dev and Ops separation of concerns: create application container images at build/release time rather than deployment time, thereby decoupling applications from infrastructure.
- Observability: not only surfaces OS-level information and metrics, but also application health and other signals.
- Environmental consistency across development, testing, and production: Runs the same on a laptop as it does in the cloud.
- Cloud and OS distribution portability: Runs on Ubuntu, RHEL, CoreOS, on-premises, on major public clouds, and anywhere else.
- Application-centric management: Raises the level of abstraction from running an OS on virtual hardware to running an application on an OS using logical resources.
- Loosely coupled, distributed, elastic, liberated micro-services: applications are broken into smaller, independent pieces and can be deployed and managed dynamically – not a monolithic stack running on one big single-purpose machine.
- Resource isolation: predictable application performance.
- Resource utilization: high efficiency and density.

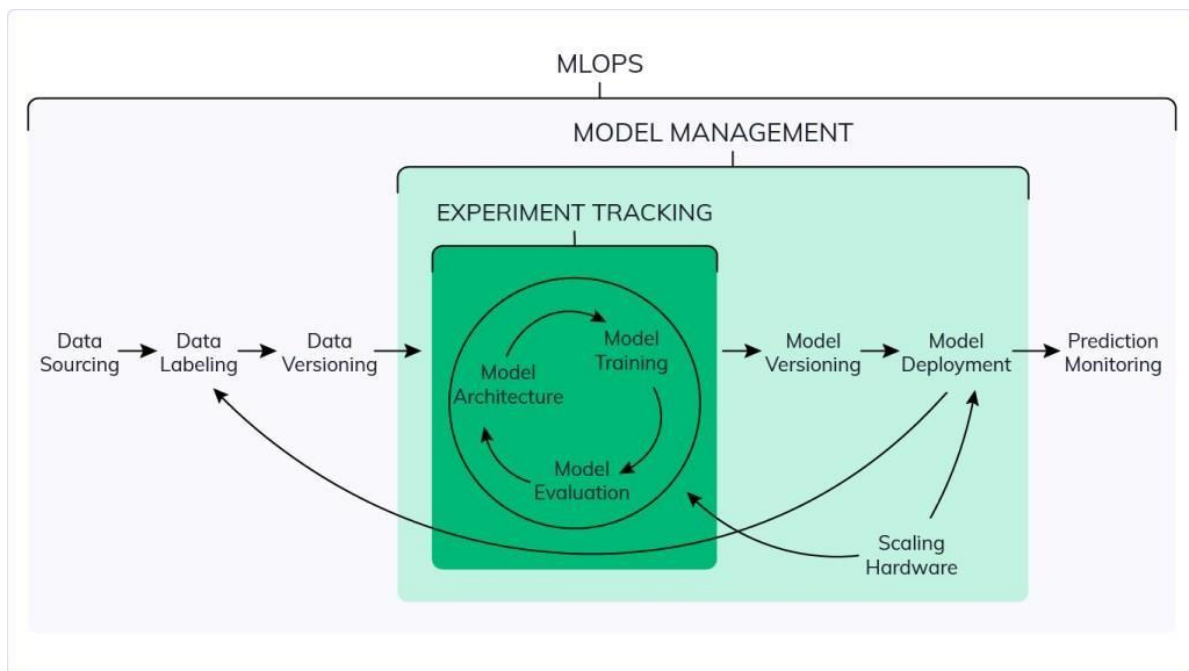
Kubernetes provides :

- **Service discovery and load balancing** Kubernetes can expose a container using the DNS name or using their own IP address. If traffic to a container is high, Kubernetes is able to load balance and distribute the network traffic so that the deployment is stable.
- **Storage orchestration** Kubernetes allows you to automatically mount a storage system of your choice, such as local storages, public cloud providers, and more.
- **Automated rollouts and rollbacks** You can describe the desired state for your deployed containers using Kubernetes, and it can change the actual state to the desired state at a controlled rate. For example, you can automate Kubernetes to create new containers for your deployment, remove existing containers and adopt all their resources to the new container.
- **Automatic bin packing** You provide Kubernetes with a cluster of nodes that it can use to run containerized tasks. You tell Kubernetes how much CPU and memory (RAM) each container needs. Kubernetes can fit containers onto your nodes to make the best use of your resources.
- **Self-healing** Kubernetes restarts containers that fail, replaces containers, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.
- **Secret and configuration management** Kubernetes lets you store and manage sensitive information, such as passwords, OAuth tokens, and SSH keys. You can deploy and update secrets and application configuration without rebuilding your container images, and without exposing secrets in your stack configuration

Experiment-9

Aim:- Implement DevOps and MLOPS using Cloud

Theory: **MLOps** is a set of practices for collaboration and communication between data scientists and operations professionals. Applying these practices increases the quality, simplifies the management process, and automates the deployment of Machine Learning and Deep Learning models in large-scale production environments. It's easier to align models with business needs, as well as regulatory requirements.



MLOps is slowly evolving into an independent approach to ML lifecycle management. It applies to the entire lifecycle – data gathering, model creation (software

development lifecycle, continuous integration/continuous delivery), orchestration, deployment, health, diagnostics, governance, and business metrics.

The key phases of MLOps are:

- Data gathering
- Data analysis
- Data transformation/preparation
- Model training & development
- Model validation
- Model serving
- Model monitoring
- Model re-training.

There are 3 ways you can go about implementing MLOps:

- MLOps level 0 (Manual process)
- MLOps level 1 (ML pipeline automation)
- MLOps level 2 (CI/CD pipeline automation)

MLOps level 0

This is typical for companies that are just starting out with ML. An entirely manual ML workflow and the data-scientist-driven process might be enough if your models are rarely changed or trained.

Characteristics

- **Manual, script-driven, and interactive process:** every step is manual, including data analysis, data preparation, model training, and validation. It requires manual execution of each step and manual transition from one step to another.
- **Disconnect between ML and operations:** the process separates data scientists who create the model, and engineers who serve the model as a prediction service. The data scientists hand over a trained model as an artifact for the engineering team to deploy on their API infrastructure.
- **Infrequent release iterations:** the assumption is that your data science team manages a few models that don't change frequently—either changing model implementation or retraining the model with new data. A new model version is implementation or retraining the model with new data. A new model version is deployed only a couple of times per year.
- **No Continuous Integration (CI):** because few implementation changes are assumed, you ignore CI. Usually, testing the code is part of the notebooks or script execution.
- **No Continuous Deployment (CD):** because there aren't frequent model version deployments, CD isn't considered.
- **Deployment refers to the prediction service** (i.e. a microservice with REST API)

- **Lack of active performance monitoring:** the process doesn't track or log model predictions and actions.

There are 3 ways you can go about implementing MLOps:

- MLOps level 0 (Manual process)
- MLOps level 1 (ML pipeline automation)
- MLOps level 2 (CI/CD pipeline automation)

MLOps level 0

This is typical for companies that are just starting out with ML. An entirely manual ML workflow and the data-scientist-driven process might be enough if your models are rarely changed or trained.

Characteristics

- **Manual, script-driven, and interactive process:** every step is manual, including data analysis, data preparation, model training, and validation. It requires manual execution of each step and manual transition from one step to another.
- **Disconnect between ML and operations:** the process separates data scientists who create the model, and engineers who serve the model as a prediction service. The data scientists hand over a trained model as an artifact for the engineering team to deploy on their API infrastructure.
- **Infrequent release iterations:** the assumption is that your data science team manages a few models that don't change frequently—either changing model implementation or retraining the model with new data. A new model version is deployed only a couple of times per year.
- **No Continuous Integration (CI):** because few implementation changes are assumed, you ignore CI. Usually, testing the code is part of the notebooks or script execution.
- **No Continuous Deployment (CD):** because there aren't frequent model version deployments, CD isn't considered.
- **Deployment refers to the prediction service** (i.e. a microservice with REST API)
- **Lack of active performance monitoring:** the process doesn't track or log model predictions and actions.

MLOps level 1

The goal of MLOps level 1 is to perform continuous training (CT) of the model by automating the ML pipeline. This way, you achieve continuous delivery of model prediction service.

This scenario may be helpful for solutions that operate in a constantly changing environment and need to proactively address shifts in customer behavior, price rates, and other indicators.

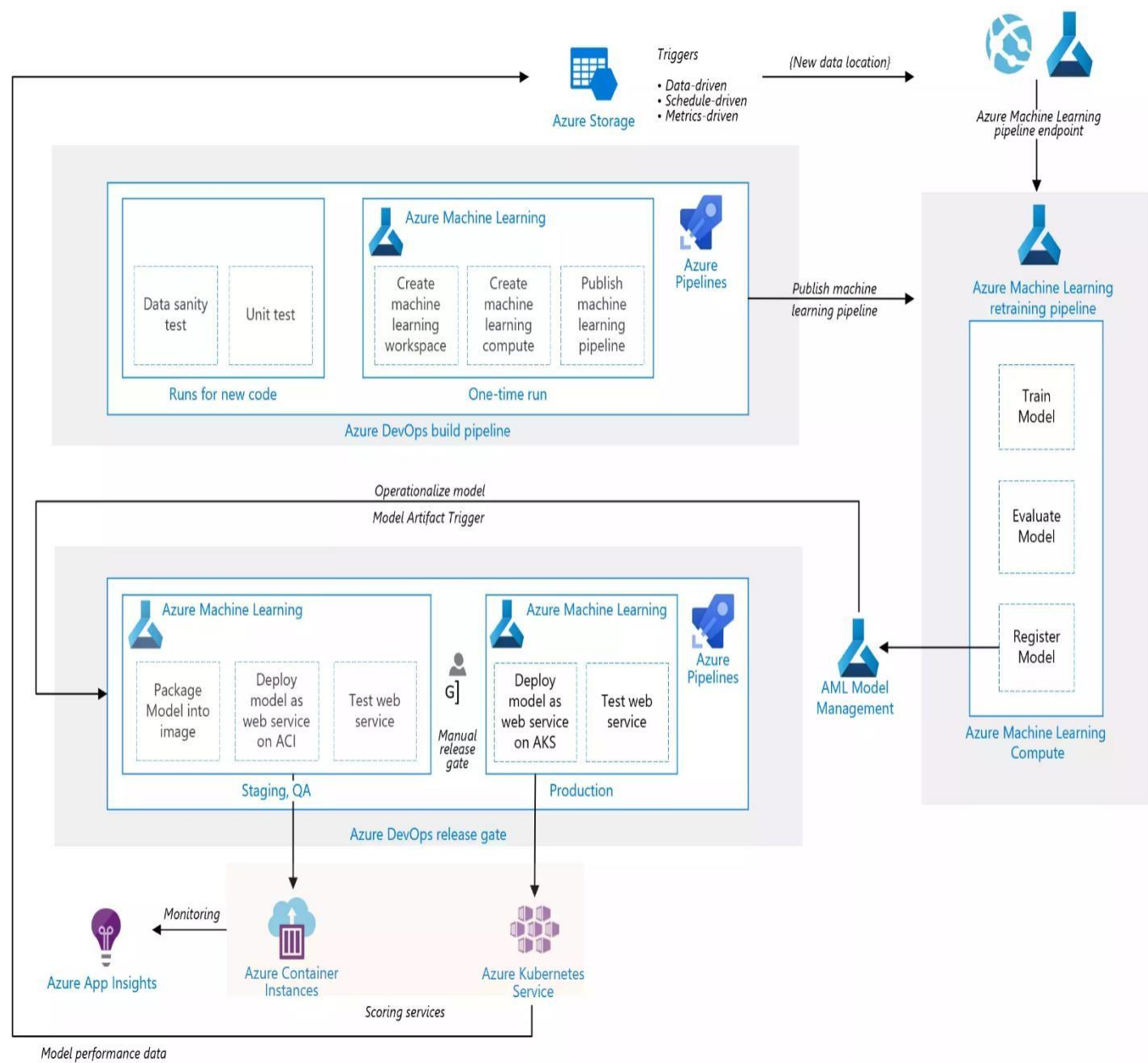
MLOps level 2

For a rapid and reliable update of pipelines in production, you need a robust automated CI/CD system. With this automated CI/CD system, your data scientists rapidly explore new ideas around feature engineering, model architecture, and hyperparameters.

This level fits tech-driven companies that have to retrain their models daily, if not hourly, update them in minutes, and redeploy on thousands of servers simultaneously. Without an end-to-end MLOps cycle, such organizations just won't survive.

This MLOps setup includes the following components:

- Source control
- Test and build services
- Deployment services
- Model registry
- Feature store
- ML metadata store
- ML pipeline orchestrator.



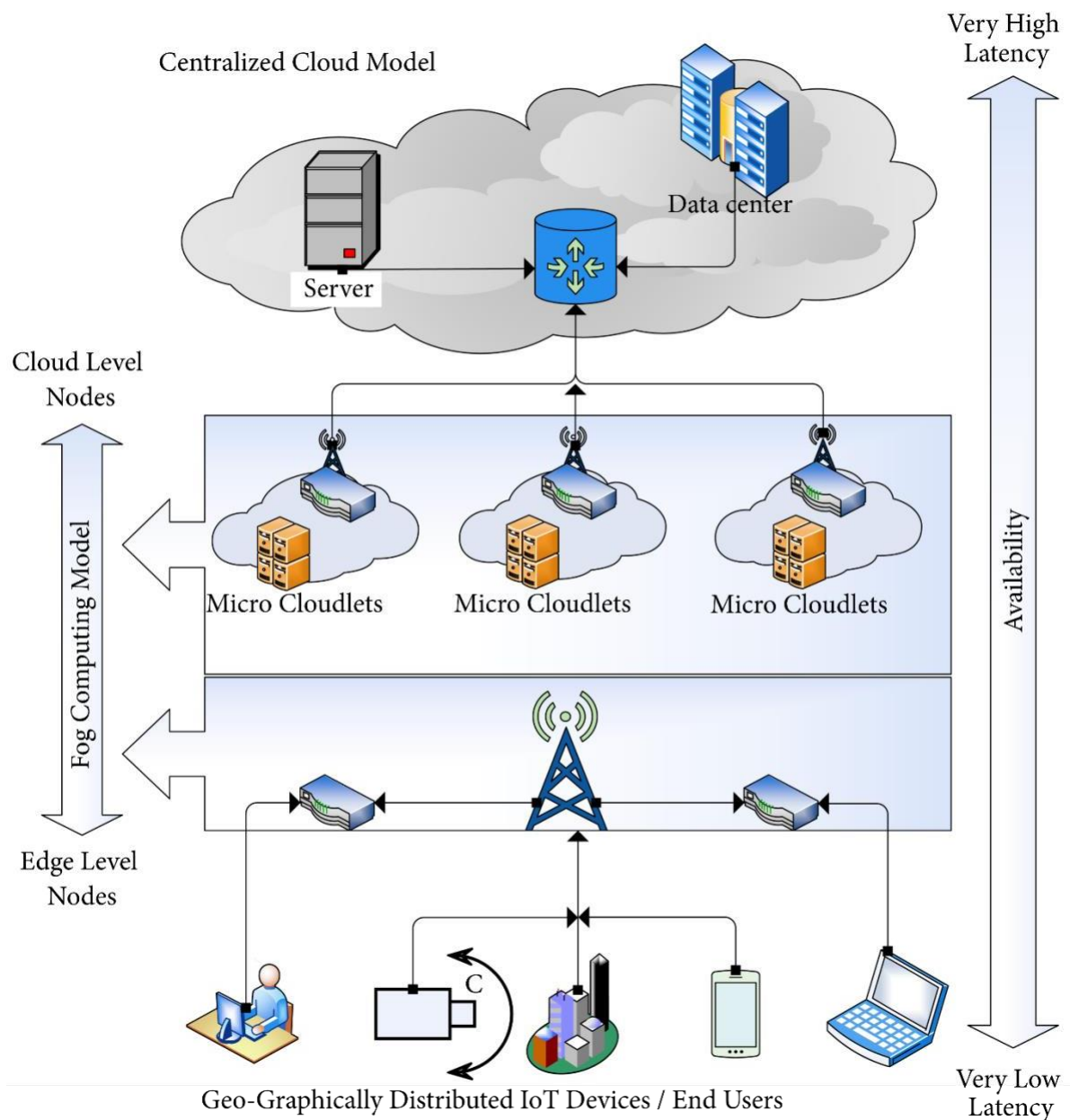
Experiment no-10

Aim: Explore Fog Computing Framework

Theory:

Fog computing or fog networking, also known as fogging, is pushing frontiers of computing applications, data, and services away from centralized cloud to the logical stream of the network edge. Fog networking system works on to build the control, configuration, and management over the Internet backbone rather than the primarily control by network gateways and switches those which are embedded in the LTE network. We can illuminate the fog computing framework as highly virtualized computing infrastructure which provides hierarchical computing facilities with the help of edge server nodes. These fog nodes organize the wide applications and services to store and process the contents in close proximity of end users. Sometimes, fog computing used frequently and often interchangeably the term “edge computing.” However, there is a little bit difference between those two concepts. Fog and edge computing both involve pushing the processing and intelligence capabilities down to the proximity where the information is originating. The main difference between both architectures is exactly where the computing and intelligence power is placed. In both structures data is sent by the same sources or physical assets, like pumps, relays, motors, sensors, and so on.

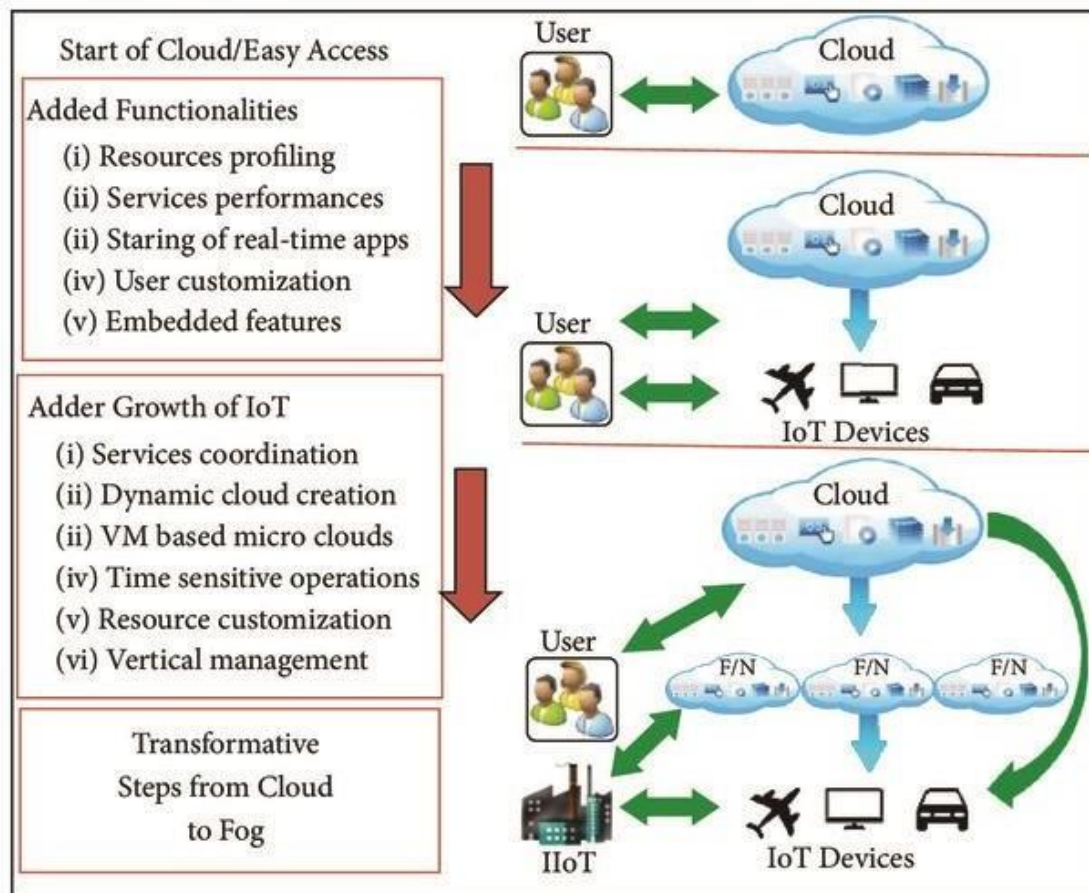
Fog computing presents a hierarchical distributed architecture with support of the integration of technological components and services might be in near future like smart cities, smart grid system, connected vehicles, and smart homes



Fog Interfaces with Cloud, IoT, and Other Fog Nodes

As defined before, fog computing expands the CC functionality with more elasticity to the edge level of the core network and shared same processing strategies and features (virtualization, multitendency, etc.) to make extendable nontrivial computation services. Some existing features of transformations make the fog computing more significant: (i) high-speed itinerant applications (e.g., smart connected vehicle and connected rail), (ii) low latency required surroundings applications, (iii) distributed control systems in large scale (e.g., smart grids, connected tracks, and STLS), and (iv) geologically distributed applications such as sensor networks to monitor the different environments.

Fog computing architecture allows processing, networking, and storage services to dynamically transfer at the fog node, cloud, and IoT continuum. However, the interfaces for fog to interact with the cloud, other fogs, and the things or users must facilitate flexibility and dynamic relocation of the computing, storage, and control functions among these different entities. It enabled well-situated end user assessment for fog computing services and also allowed capable and effectual QoS management.



Fog to Cloud. From fog to cloud interface can be considered compulsory to support fog to cloud and cloud to fog collaboration which provides back-to-back services. It also supports functionalities, for example, (i) some functions at fog to be supervised or managed with the cloud computing ability, (ii) cloud and fog which can transfer data to each other for processing and comparing and for other required functionalities, (iii) cloud which can decide to distribute or schedule fog node(s) for allocation of the services on demand, (iv) cloud and fog both mutually which can differentiate for better management computing services, with each other, and (v) cloud which can make availability of its services through fog to the end users. It is essential to find out which information and services should be transversely passed at fog and cloud. Regularity and granularity of such data and information should decide how fog or cloud can response to that information or data.

Fog to Fog. Fog nodes must have pool resources functionality to support processing with each other. For example, all deployed fog nodes can share their data storage, computing, and

processing capability tasks with prioritized node functionality system for one or several user applications. Multiple fog nodes might also act together with service for backups of each other.

Fog to IoT/End User. Fog computing provides its services to a widely distributed structure of IoT devices (e.g., smart devices and sensors) with the differential identification recognizing system. Fog to IoT interface or fog to user interface essentially needs to allow IoT access for fog services in user friendly environment and resource efficient and within secure ways.

Figure 2 explains about interfaces of fog with cloud and IoT both, through hierarchically distributed fog computing structure iteratively continuum. The right side of figure describes an emerging era of technology world from traditional cloud computing towards nearly deployed fog computing. It is also visualized that which type of interface is also included in different type of era (e.g., fog to cloud, fog to fog, and fog to IoT). In the left side of picture, it is pointed out that why we have a single and combined platform (fog computing) of these essential technologies. In the end, an important dispute or a question is raised that how all these interfaces and protocols to derive fog nodes functionality in existing or nearly distributed fog structure which will work together be designed. How the cost of required infrastructure could be reduced with providing high QoS to the end users or IoT.