Draft Study Material

JOB ROLE

JUNIOR CLOUD COMPUTING ASSOCIATE

Qualification Pack
QG-04-IT-00354-2023-V1-NIELIT

SECTOR: IT-ITeS
Grades XII

विद्यया ऽ मृतमश्नुते

एन सी ई आर टी
NCERT

PSS CENTRAL INSTITUTE OF VOCATIONAL EDUCATION

(a constituent unit of National Council of Educational Research and Training (NCERT) under Ministry of Education, Government of India)

www.psscive.ac.in

Junior Cloud Computing Associate

Grade – XII

Qualifications Pack
QG-04-IT-00354-2023-V1-NIELIT

PSS Central Institute of Vocational Education

[A constituent unit of NCERT, Under Ministry of Education, Government of India)

Shyamla Hills, Bhopal - 462 002, Madhya Pradesh, India
www.psscive.ac.in

Published by:
Joint Director
PSS Central Institute of Vocational Education, NCERT,
Shyamla Hills, Bhopal

PREFACE

The National Education Policy, 2020 emphasizes upon removing hard distinction between arts, science and commerce; and between curricular, co-curricular and extracurricular activities; and between vocational and general education. NEP focuses on flexible curricular structure and multidisciplinary learning. The secondary stage is for students aged between 14 and 18 and is divided into two phases: Phase 1 — Grades 9 and 10 and Phase 2 – Grades 11 and 12. The secondary stage for students aged 14-18 is divided into two phases, with the guidelines presented for grades 11 and 12. The National Curriculum Framework for School Education (NCFSE) 2023 advocates for choice-based courses, aiming to provide flexibility, remove separations between disciplines, and align with industry needs. Vocational education in the secondary stage will be an integral part of the educational system designed to provide students with practical skills and knowledge that directly prepare them for specific careers or trades. The focus should be on the holistic development of each child, addressing not only vocational skills but also social, emotional, and life skills. In schools, vocational courses are expected to align with the National Skill Qualifications Framework (NSQF) falling within NSQF levels 3 and 4. The NSQF is a quality assurance framework that organizes qualifications in a series of eight levels, in increasing order of complexity and competency. These levels are defined in terms of learning outcomes which are an explicit description of what a learner should know, understand and be able to do as a result of learning, regardless of whether these competencies were acquired through formal, non-formal or informal learning.

The NEP underscores the importance of vocational education, preparing students with practical skills aligned with industry needs. In the context of web development, this translates to equipping learners with not just theoretical knowledge but also hands-on experience. We should strive to provide comprehensive training that empowers individuals to tackle real-world challenges in the digital landscape.

Just as the NEP emphasizes the holistic development of students, our approach to web development should extend beyond technical skills. Let's prioritize the development of essential soft skills such as problem-solving, collaboration, and adaptability, ensuring that learners are well-rounded professionals capable of thriving in diverse environments.

I thank all other members for completing this task on time and in such an admirable way. I am also thankful to all the institutions and organizations which have generously extended their help and assistance in making this possible. As an organization committed to reforming school education in Bharat and continuously improving the quality of all learning and teaching material that it develops, NCERT looks forward to critical comments and suggestions from all its stakeholders to further improve upon this textbook.

Professor Dinesh Prasad Saklani

Director

National Council of Educational Research and Training

New Delhi

Foreword

Vocational Education and Training (VET) plays a significant role in willing youth for relevant occupation and meeting the skill demand of the changing labour market. This is even more relevant, at India is witnessing accelerated youth population and the need for preparing skilled workforce for the growing economy. The strong partnership with the industry partners characterizes India's National Shill Qualification Framework (NSQF], The Vocationalisation of Education in Schools under *Samagra Shiksha* by the Ministry of Education, Government of India is spearheading and catalyzing the role of vocational education and training in equipping young people with skills.

The recent reforms through National Educational Policy (NEP) 2020 have focused on making VET system more coherent and flexible to both the needs of the labour market and social challenges. Improving the learning pathways and bridging the gap between vocational and general education and avoiding dead ends is another goal The ultimate goal is to ensure flexibility and responsiveness to the needs through education and training and to provide a strong framework for lifelong learning.

Reflecting on vocational education and training priorities, and recent developments in the system, priority has to be placed on developing vocational teachers of trainers to act as the link between education and training and employment. Preparing a cadre of professionally trained. vocational teachers is vital for imparting quality vocational education and developing skilled workforce in different sectors In this perspective, the PSS Central Institute of Vocational Education (PSSCIVE), Bhopal has introduced a 'Diploma in Vocational Education and Training' through distance mode, with the aim to develop a pool of trained vocational teachers or resource persons in spearheading the effective, Implementation of the scheme an vocationalisation of education in schools across India, The Diploma in VET is a one year programme, which will be taught in four blocks of tri-semester. It aims at providing the learners with the latest knowledge, skills and competencies in the field of vocational education and training Among others, the programme will also enable the learners to appreciate the ethical dimension of teacher professionalism in Vocational Education. The goal is to. equip the learners with a strong theoretical and practical understanding of VET while integrating ICT in their teaching.

I acknowledge the contributions of the material development team, reviewers and the support team for their contributions in the development of this self- learning material. We would welcome suggestions, which would help us to improve further the quality of this programme.

Wish you all the very best in this endeavor.

Dr. Deepak Paliwal

Joint Director

PSSCIVE, Bhopal

About the Textbook

“Junior Cloud Computing Associate for class 12” is a concise yet comprehensive textbook designed to equip students with essential cloud computing. Through clear explanations and practical exercises, students learn the foundational concepts of Advance Linux, Cloud Architecture with Open Stack and Cloud Computing Services. By engaging in hands-on projects, students gain practical experience in managing troubleshooting OpenStack environments, understanding Cloud Computing services, Cloud Infrastructure and Architecture, and working with Cloud Storage and Databases, building confidence and problem-solving skills in real-world IT scenarios.

This book provides a structured pathway for students to acquire valuable skills that are increasingly in demand. Whether students aim to pursue further education or enter the workforce, this textbook serves as a stepping stone to success in the dynamic field of cloud computing. The book is divided into three units, the book meticulously covers every aspect of Advance Linux, Cloud Architecture with Open Stack and Cloud Computing Services.

Throughout the book, emphasis is placed on developing critical thinking and problem-solving skills, enabling students to manage and troubleshoot system-level and network-level challenges effectively. Practical exercises and case studies accompany each chapter, providing students with hands-on experience and reinforcing theoretical concepts.

Dr. Munesh Chandra

Professor

Department of Engineering and Technology

PSSCIVE, NCERT, Bhopal

Textbook Development Team

1. Dr. Kanika Taneja, ABES Institute of Technology, Ghaziabad
2. Dr. Nandita Goyal, AKG Engineering College, Ghaziabad
3. Mr. Mayank Bhushan, Directorate of Education, Delhi, Lecturer
4. Dr. Monika Sharma, PSSCIVE, Bhopal
5. Ms. Soumya Trivedi, AKG Engineering College, Ghaziabad

MEMBER-COORDINATOR

Dr. Munesh Chandra,

Professor & Head ICTC & Computer Centre

Department of Engineering and Technology

PSSCIVE, NCERT, Bhopal

Acknowledgement

On behalf of team at the PSS Central Institute of Vocational Education (PSSCIVE), Bhopal we are grateful to the officials of the Department of School Education and Literacy, Ministry of Education, Government of India for the guidance and support at all times and levels.

We are obliged to the Director, NCERT for his care and leadership. We are indebted to the PAC NCERT for financial support.

We acknowledge the contributions of our colleagues at PSSCIVE and other experts for their academic support, untiring efforts and contributions in development of this material. The names of all the experts are acknowledged in the list of contributors.

The contributions made by the Administration and the supporting staffs of PSSCIVE are duly acknowledged.

TEAM PSSCIVE

Table of Content

Unit-1.
Introduction to Advance Linux

In its most fundamental form, Linux is an open-source operating system (OS) that is constructed on the Linux kernel. It is a system that enables several users to simultaneously run applications while simultaneously managing the resources possessed by the system. An operating system is comprised of several fundamental components, including the kernel, the shell, and the file system.

The Linux operating system is built on top of the UNIX computing platform. The UNIX operating system dates back to the 1970s and was developed by AT&T Bell Labs. It is capable of supporting multiple users and tasks at the same time. It is a source of thankfulness for a great number of contemporary operating systems, including Linux.

Linux, in contrast to UNIX, is totally free to use for everyone and everyone because it is an open-source operating system and there are no license requirements. Individuals from all across the world are able to view the source code, which fosters global collaboration and advancement. The robust security strategy and effective performance of this product can be advantageous to a wide variety of devices and businesses.

Linux has developed into a dependable and secure operating system that is used to power a broad variety of devices. These devices include desktop computers, mobile phones, and massive supercomputers. Because of its capacity to finish a number of jobs in a short amount of time, it has earned a reputation for being efficient and having low operating expenses.

In this unit, you will Linux Networking Concepts, Network Troubleshooting and Monitoring and concepts related to remote access.

Chapter-1

Linux Networking Concepts

Ritesh loved exploring and learning new concepts. He already had idea about Operating systems and had explored some basic concepts related to Windows and Linux operating systems. Now, he heard about networking concepts for Linux. While doing this, he faced some challenges but finally he overcame them. He found a lot of digital information about them but then he realized that there are many more related concepts about Linux Networking.

In this chapter 1, you will learn about OSI model for networking along with Domain Name System and its related concepts.

1.1. Definition of OSI Model & layers

Devices from different manufacturers had a hard time communicating in the early days of networking due to the prevalence of proprietary systems. To address this issue and promote interoperability a conceptual framework for standardizing the way different computer systems communicate over a network, the OSI model was developed by the International Organization for Standardization (ISO). In order to facilitate communication between various pieces of hardware and software, it partitions network communication into seven separate layers, each of which performs a unique role. Create new technologies, create networks, and troubleshoot existing ones with ease using this tiered methodology.

Key Features:

Conceptual framework:
To clarify, the Open Systems Interconnection (OSI) model is more of a framework for thinking about how networks work than a physical design.

Seven Layers:
The physical, data link, network, transport, session, presentation, and application layers are the ones that structure the functions of a network.

Interoperability:
Establishing a common set of protocols and guidelines to enable systems from many suppliers to communicate is the main objective.

Standardization:
It facilitates the creation, testing, and upkeep of network systems by providing a shared vocabulary for communicating across networks.

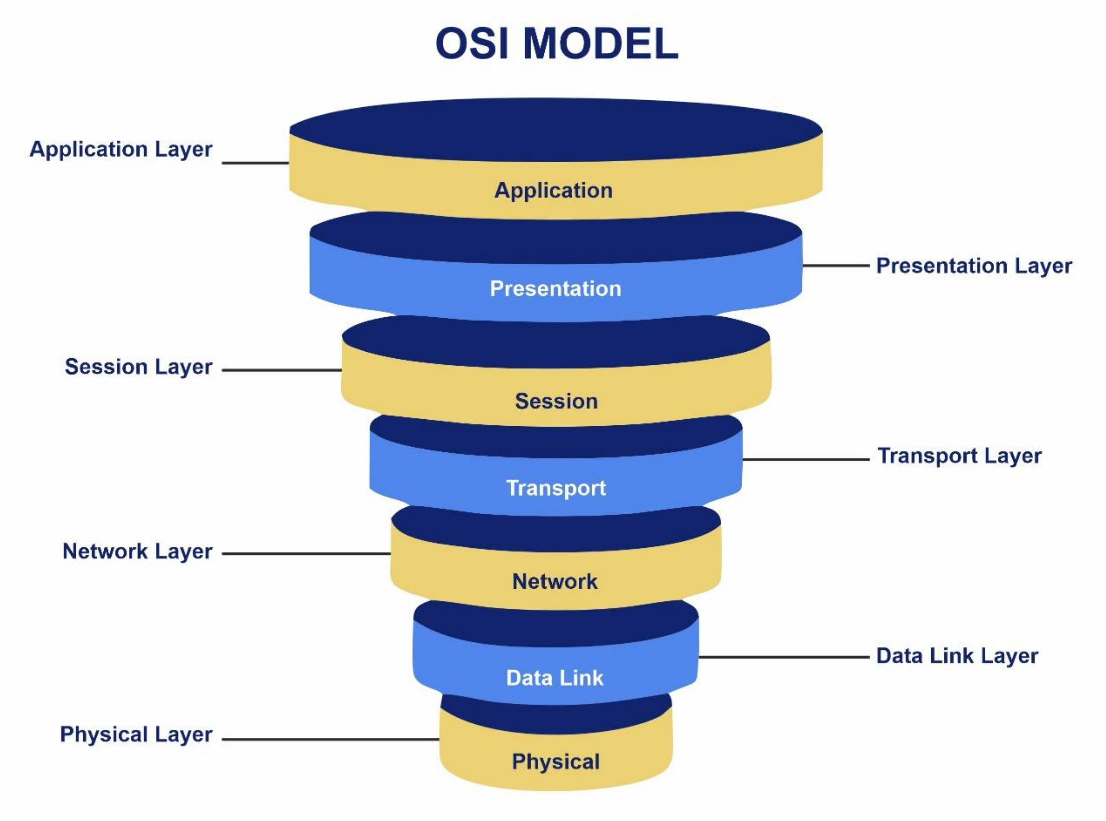


Fig 1.1 OSI Model

Layers of OSI Model

1. Physical Layer:
Data transmission via physical and electrical media encompasses a wide variety of methods, such as cables, voltage, and bit transmission.

2. Data Link Layer:
This layer is responsible for detecting and correcting faults in data transmission between directly connected nodes.

3. Network Layer:
Finds the optimal route for data packets across networks by using IP addresses.

4. Transport Layer:
With capabilities like segmentation, reassembly, and flow management, it guarantees dependable and transparent data transmission between end systems.

5. Session Layer:
Creates, maintains, and ends sessions—the link between programs.

6. Presentation Layer:
Ensures data usability for the application layer by handling data format and encryption.

7. Application Layer:
Applications like email, file transfer, and web browsing rely on network services provided by the layer nearest to the user.

Assignment 1.1
a. Write down the names and functions of all the layers of OSI model.
b. Name some of the main applications supported by application layer.

1.2. Domain Name System (DNS)

The Domain Name System (DNS) is the core directory function of the Internet. Websites such as espn.com and nytimes.com enable consumers to acquire information online. Internet Protocol (IP) addresses enable web browsers to communicate with each other. DNS transforms domain names to IP addresses, which allow browsers to access Internet resources.

Other computers may identify any Internet-connected device by searching for its unique IP address. DNS servers have eliminated the need for humans to memorize Internet Protocol (IP) addresses, whether they are simple integers like 192.168.1.1 (IPv4) or more complex ones like 2400:cb00:2048:1::c629:d7a2 (IPv6).

DNS' principal function is to quickly provide the information required to link users to remote hosts. It is essential for online browsing and the majority of internet activities. There is a hierarchical distribution of DNS mapping on the internet. Companies, governments, and educational institutions all have unique IP address ranges and domain names assigned to them. Furthermore, they are in responsible of running the DNS servers, which transform domain names into IP addresses. The domain name of the web server that processes client requests serves as the foundation for the majority of Uniform Resource Locators.

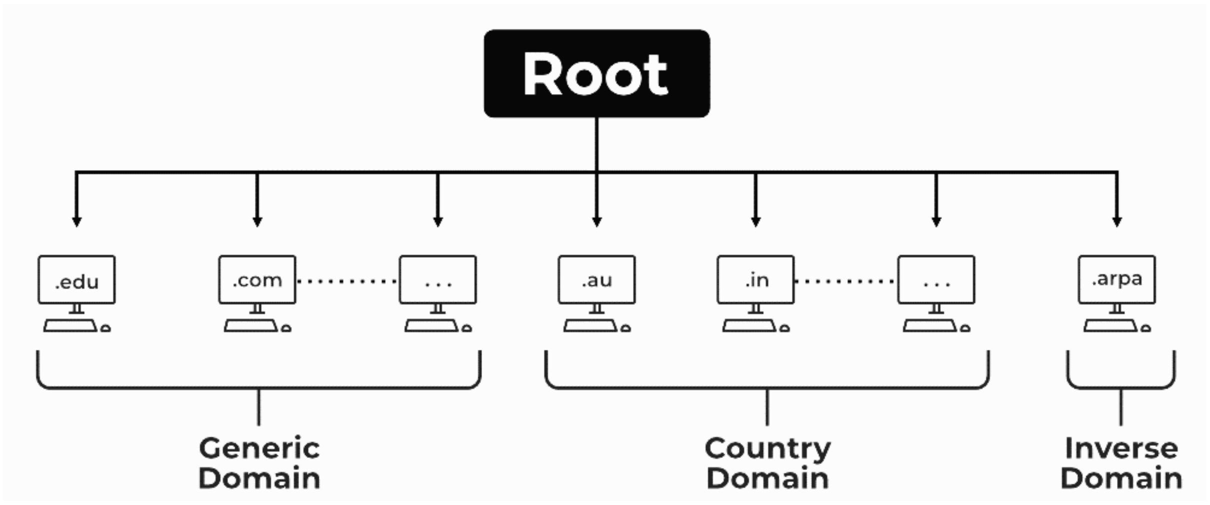


Fig 1.2 Domain Name Systems

Working of DNS

To make IP addresses usable by computers, DNS servers convert domain names and URLs. For a computer to access a website, they transform the user's input into a format that a browser can understand. Domain name system (DNS) resolution entails translating and searching for domain names.

The following are the main components of the DNS resolution process:

1. A user opens a web browser and types in a domain name or URL.
2. To find the IP address that goes with the domain, the browser runs a recursive DNS query on the network.
3. The request is forwarded to an ISP-maintained recursive DNS server, sometimes called a recursive resolver. The user receives the address back from the recursive resolver, and the webpage loads, if it finds it.
4. To find a response, the recursive DNS server will first query the DNS root name servers, then the name servers for top-level domains (TLDs), and finally the authoritative name servers.
5. The asked-for IP address is part of a DNS record, and the three kinds of servers work together to keep redirecting until they discover it. With this data sent to the recursive

DNS server, the user's requested URL gets loaded. The Domain Name System (DNS) and Top-Level Domain (TLD) servers typically reroute queries rather than offering direct solutions.

6. The A record for the domain name, which contains the IP address, is saved or cached by the recursive server. It will then be able to bypass intermediate servers and respond directly to the user the next time that domain name is requested.
7. A message is sent if the authoritative server is unable to find the information when the query reaches it.

Users rarely notice anything because the process of requesting many servers takes fractions of a second.

Any user, whether within or outside of their own domain, can send an inquiry to a DNS server. When someone from outside the domain requests information about a name or address within the domain, the server provides the official response.

Servers generally route requests for external names or address from their domains to other servers, usually those maintained by their Internet service providers (ISPs).

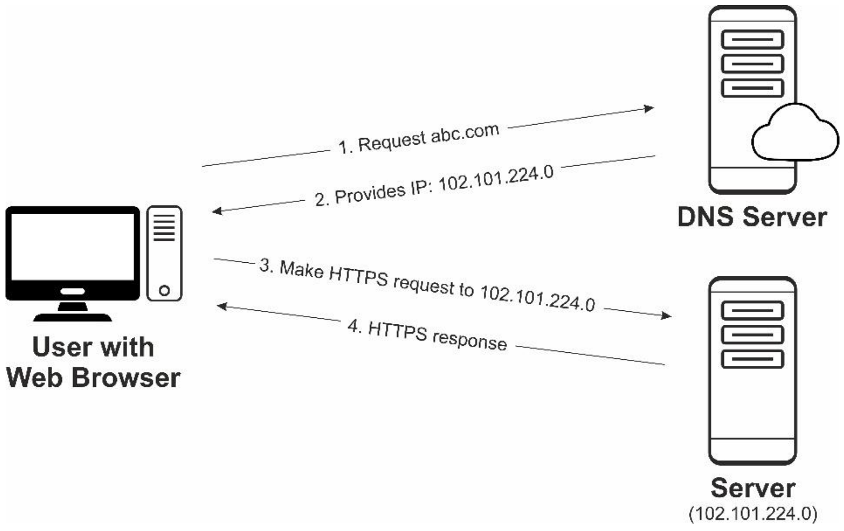


Fig 1.3 Working of DNS

Assignment 1.2
a. What is the main role of DNS?
b. Explain the process for working of DNS.

Practical Activity 1.1: Use browser developer tools to observe DNS activity.

You may monitor DNS activity in your browser's developer tools by reloading the page and selecting the Network tab (typically accessed via F12 or Ctrl+Shift+I). The Network tab displays the page's network queries, which include DNS lookups. You can refine DNS-related queries by selecting options such as "All," "XHR," "Doc," etc. Chrome and other browsers provide detailed DNS information via specific URLs such as chrome://net-internals/#dns.

Steps for viewing DNS traffic with Chrome:

1. **Open Developer Tools:**
   Right-click on the webpage and select "Inspect" or press F12, or Ctrl+Shift+I.
2. **Navigate to the Network Tab:**
   Within the developer tools, select the “Network” tab

3. **Enable Preserve Log (Optional):**
   Enable "Preserve log" in order to capture DNS activity for loading page and subsequent actions.
4. **Reload the page:**
   This will trigger the browser to perform DNS lookups and display them in the Network tab.
5. **Observe Network Requests:**
   Examine the list of network requests. DNS lookups will be visible as part of the overall network activity. You can filter requests by type (e.g., "Doc", "XHR", "Fetch") to focus on DNS-related entries.
6. **Inspect DNS Details:**
   Click on a specific request to view more details, including timing information like the DNS lookup time.

Points to Remember:
What you should mostly look for under the Network tab:
The time it takes the browser to translate a domain name to its numerical IP address is referred to as the DNS lookup time.

- Request Initiator: Displays the section of the website that requested the domain name system (DNS).
- Status Code: Shows the result of the DNS query (200 denotes success).
- The response headers might contain information about DNS prefetching.

Following these instructions will allow you to utilize your browser's developer tools to monitor and analyse a webpage's DNS activity.

1.3. BIND Zone

The Berkeley Internet Name Domain (BIND) program maintains a set of DNS zones, and each of these zones has its own text file that provides the configuration information. In doing so, it converts human-readable URLs into machine-readable IP addresses and vice versa for other resource records. One popular open-source DNS server software is BIND.

Important features of a Bind map file:

- A zone file primarily serves as a database for the DNS server, outlining the DNS entries for a particular domain or subdomain.
- Organization: RFC 1035 normally defines the format that zone files adhere to. In these you'll find records of several kinds, including: Among other zone-wide attributes, the SOA (Start of Authority) specifies which server is the principal authoritative one.
- The "name servers" (NS) that handle zone inquiries are specified here. “A” stands for “Address,” and it converts domain names to IPv4 numbers. The AAAA protocol converts domain names into IPv6 addresses.
- Mail Exchanger (MX): This record specifies which mail servers are authorized to receive emails on behalf of the domain.
- Canonical Name (CNAME): Generates domain name aliases.
- Setting up BIND: Zone files are used to serve DNS queries for the zones that are defined in them. In response to a domain name system query, the BIND server looks up the associated IP address and other details in the appropriate zone file.

- Environments with dynamic IP addresses or frequent changes are well-suited to BIND because of its support for dynamic updates, which enable automatic updates to zone files based on changes in the network.
- Serial Number: Whenever a zone file is updated, its serial number must be incremented. This makes ensuring that the most recent data is cached by resolvers.

Here are a few instances:

Domain name to Internet Protocol address translations are stored in the forward zone. For reverse DNS lookups, the Reverse Zone stores IP address to domain name mappings. Some of the capabilities supported by BIND include load balancing through the use of several resource records for a single name and DNSSEC (DNS Security Extensions), which provide improved security.

History of BIND

BIND, which stands for Berkeley Internet Name Domain, is a collection of tools that includes the DNS server software that is the most widely used in the world. This all-encompassing DNS service and tool implementation is designed to be completely compliant with all standards, and it has the potential to act as a model for the development of prospective DNS software. The Berkeley campus of the University of California, Berkeley, was the location where the free and open-source software package known as BIND was developed in the 1980s. BIND 9, the most recent major version, was initially released in the year 2000. The Internet Systems Consortium is responsible for the regular release and release of updates for BIND 9. When it comes to networks that are relatively small or straightforward, BIND is capable of handling all DNS-related tasks on its own. One of the capabilities of BIND is the ability to simultaneously operate authoritative and cached DNS servers.

Use of BIND

If you need DNS server software, BIND is the best option. BIND DNS servers are typically managed on a daily basis by Linux/UNIX-proficient system administrators or network administrators. Running BIND on Windows hosts is possible, but it requires a thorough understanding of the system requirements for running open-source services. Many administrators prefer BIND to alternative solutions, such as Microsoft DNS, because it is open-source and closely follows IETF standards (RFCs). Custom tools customized to specific DNS use cases and operational requirements can be simply created using BIND. However, keep in mind that BIND is solely responsible for DNS management; it does not handle IP address management or any other associated services.

Need of BIND

There are various scenarios in which being how to configure a BIND DNS server is useful. If you want to work on a network or a development team, you must understand how to configure and use BIND. Furthermore, BIND enables fine-grained management of DNS servers. It will help you quickly understand the principles of providing core network services. Last but not least, the core skills you develop while using BIND will be beneficial even if you encounter a network that does not utilize BIND or is transitioning away from it. The majority of BIND software tools, with the exception of the DNS server, are interoperable with other DNS servers because they use the standard DNS message protocol.

Features of BIND

- Whether it's a primary or secondary server, authoritative DNS enables you to publish DNS zones and records under your control.
- Partition the Domain Name System (DNS) so that different groups of people, both within and external to the firm, can access different data sets. Despite the fact that each view has typically been treated as its own virtual server, BIND has lately added facilities to allow data sharing between views.
- Recursive DNS, also known as a caching resolver, is used to retrieve information from other DNS servers on behalf of client systems such as mobile devices, desktop workstations, and servers.
- According to RFC 2136, dynamic DNS (DDNS) is the process of updating a main server by adding or removing entries via a certain type of DNS message.
- Data replication made simple: quickly and precisely copy data from main to secondary servers, including change notifications from main to secondary and incremental zone transfer requests from secondary to primary.
- Verify incoming data on a cache server, then use DNSSEC to cryptographically sign authoritative data. BIND supports both elliptic curve encryption and the newest versions of DNSSEC.
- Verify and cryptographically sign communications with a pre-shared key or a dynamically negotiated key; this is known as Transaction Signatures (TSIG) and Keys (TKEY). Many recent standard signature algorithms work with BIND. Microsoft Active Directory uses one of these.
- Reduce the impact of distributed denial of service (DDOS) attacks by implementing a number of response strategies.
- Support Internet Protocol Version 6 (IPv6) by publishing name servers with IPv6 addresses and participating in IPv6 networking directly.
- The advantages of With BIND. You can customize BIND to your desire. If you know how to use Perl, Python, BASH, or Powershell, you can create your own specialized network tools. There is no upfront fee to utilize BIND. BIND allows you to get up and running for free, unlike commercial DNS solutions such as BlueCat, Microsoft, or Infoblox. The majority of Linux and UNIX distributions already provide an installation of BIND in their repositories.
- BIND is supported by a substantial user base. The BIND community and knowledge base are vast and global, covering subjects such as utilizing and troubleshooting BIND.
- BIND is an excellent platform to start with. Throughout your career, you will meet the majority of commercial DNS solutions based on the BIND platform. It will be useful to have a basic understanding of how to configure a BIND server.

Limitations of BIND

- The DNS server BIND scales effectively. It has previously been stated that additional commercially available, open-source, or homegrown solutions are required to handle BIND at scale. Not to note that BIND only supports DNS services and utilities. This necessitates a more complete administrative architecture to ensure that BIND and related services such as DHCP and IPAM work seamlessly together. This prevents data from diverging or clashing, causing outages.

- Complete network visibility cannot be gained just using BIND. In terms of DNS traffic, each DNS server runs separately, and BIND does not provide an overview of your network's DNS traffic.
- Breaking BIND is straightforward. Because of the numerous and complex configuration options, a syntax error could ground your network to a halt. Sometimes syntactic changes occur in the configuration between various versions of BIND, exacerbating the difficulty.

BIND Zone Configuration

In order to configure BIND zones, the major sites for doing so are the named.conf file and the zone files that are associated with it. It is in the named.conf file that the global settings, zones, and permissions of BIND are defined. Zone files, on the other hand, are where the real domain DNS entries are stored. These records include A, CNAME, and MX records.

Practical Activity 1.2: Demonstration of BIND (named) Server & BIND Zone Configuration.

BIND is a robust domain name system (DNS) server that follows both the final and draft DNS specifications set by the Internet Engineering Task Force (IETF). The following is an example of how administrators usually utilize BIND:

- **Local DNS** cache server.
- Server that stores DNS cache locally and is the **authoritative server for zones**.
- High availability for zones will be provided by the **secondary server**.
- **When determining whether to run BIND in a change-root environment or using SELinux to secure it, there are a few things to take into consideration.**

To set up BIND installation:

a. It is advised to run the **named** service without using a change-root environment. In this instance, SELinux in **enforcing** mode prevents the use of known security vulnerabilities like BIND. SELinux is set to function in **enforcing** mode by default in Red Hat Enterprise Linux.

b. Execute the **named-chroot** service while the root is being changed. A process's root directory, as well as any subdirectories it may have, can be defined by administrators to be distinct from the / directory using the change-root functionality. A change to the /var/named/chroot/directory occurs in BIND whenever the **named-chroot** service is started. Consequently, the service binds the files and directories listed in /etc/named-chroot.files to the directory /var/named/chroot/ using mount commands. The process cannot access any files outside of this directory.

c. When utilizing BIND: Using the named service is how standard mode works. Use the named-chroot service in a change-root environment. Installing the named-chroot package is an additional requirement for this.

BIND caching DNS server configuration:

Successful and unsuccessful lookups are resolved and cached by default by the BIND DNS server. After that, the service uses its cache to respond to queries that target the same records. The performance of DNS lookups is greatly enhanced by this.

Prerequisites

- The server must have a static IP address.

Procedure

1. Install the bind and bind-utils packages into your system:

```
    # dnf install bind bind-utils
```

2. To use BIND in a change-root environment, install the bind-chroot package:

```
   # dnf install bind-chroot
```

3. Make the following edits to the options statement in the file /etc/named.conf:

   - Change the listen-on and listen-on-v6 directives to direct BIND to listen on specific IPv4 and IPv6 interfaces:

```
  listen-on port 53 { 127.0.0.1; 192.0.2.1; };

  listen-on-v6 port 53 { ::1; 2001:db8:1::1; };
```

   - Revise the allow-query statement to specify the ranges and IP addresses that clients are allowed to use to query this DNS server:

```
allow-query { localhost; 192.0.2.0/24; 2001:db8:1::
```

   - The IP addresses and ranges that BIND accepts for recursive queries can be defined with an allow-recursion statement:

```
allow-recursion { localhost; 192.0.2.0/24; 2001:db8
```

   - As a default, BIND will recursively ask the root servers until it finds an authoritative DNS server to resolve the query. Also, your provider's DNS servers, among others, can be pointed to by configuring BIND to do so. To tell BIND which DNS servers to send queries to in this scenario, you can include their IP addresses in a forwarders statement:

```
    forwarders { 198.51.100.1; 203.0.113.5; };
```

     If the forwarder servers fall down, BIND will recursively resolve the queries as a backup. Inserting a forward only; statement will turn off this behavior.

4. Make sure the /etc/named.conf file is correctly formatted:

```
    # named-checkconf
```

   If the command does not produce any output, then the syntax is valid.

5. Modify the firewall rules so that DNS traffic can enter:

```
# firewall-cmd --permanent --add-service=dns

# firewall-cmd --reload
```

6. Get BIND up and running:

```
# systemctl enable --now named
```

For the purpose of starting and enabling the BIND service in a change-root environment, the systemctl enable --now named-chroot command should be utilized.

Setting up a zone on a BIND primary server.

1. Forward Zone

Through the use of forward zones, names are converted into IP addresses and other pertinent data. Through the configuration of a forward zone in BIND, it is possible to resolve names such as example.com, for example, by handling queries such as www.example.com

Prerequisites

- An example of how BIND is already set up is as a name server that caches results.
- You can see that the named or named-chroot service is active.

Procedure

I. Update the /etc/named.conf file with a zone definition:

```
zone "example.com" {

    type master;

    file "example.com.zone";

    allow-query { any; };

    allow-transfer { none; };

};
```

According to these parameters,

- The example.com zone uses this server as its main server, also known as a master server.
- You may locate the zone file at /var/named/example.com.zone. In the options statement, you can give a relative route, as seen in this example. This path will be relative to the directory that you choose.
- This zone accepts inquiries from any host. To restrict access, select IP ranges or nicknames from the BIND access control list (ACL).
- There is no host that is able to do zoning. The zone transfers should only be enabled when secondary servers are being configured, and even then, only for their IP addresses that are being transferred.

II. Double check the syntax of the file located at /etc/named.conf:

```
# named-checkconf
```

In the event that the command does not provide any output, the syntax is accurate.

III. As an illustration, the following is the content that you ought to provide in the /var/named/example.com.zone file:

```
$TTL 8h
@ IN SOA ns1.example.com. hostmaster.example.com. (
                          2022070601 ; serial number
                          1d         ; refresh period
                          3h         ; retry period
                          3d         ; expire time
                          3h )       ; minimum TTL

                  IN NS   ns1.example.com.
                  IN MX   10 mail.example.com.

www               IN A    192.0.2.30
www               IN AAAA 2001:db8:1::30
ns1               IN A    192.0.2.1
ns1               IN AAAA 2001:db8:1::1
mail              IN A    192.0.2.20
mail              IN AAAA 2001:db8:1::20
```

This is the zone file:

The default value for resource records' time-to-live (TTL) is 8 hours. Values without a time suffix, such as h for hour, are regarded as seconds by BIND. This includes the essential SOA resource record that describes the zone. This command specifies ns1.example.com as the primary DNS server for the given domain. A name server (NS) record is required for a zone to operate. Two name servers are required to comply with RFC 1912. The mail exchanger (MX) for the example.com domain is configured as mail.example.com. The numerical value preceding the host name indicates the record's priority. We give higher weight to entries with lower values. Sets up the network settings for example.com, mail.example.com, and ns1.example.com in both IPv4 and IPv6.

IV. Make sure that only the people you've authorized can access the zone file:

```
# chown root:named /var/named/example.com.zone
# chmod 640 /var/named/example.com.zone
```

V. Please check the syntax of the file located at /var/named/example.com.zone.

```
# named-checkzone example.com /var/named/example.com.z
zone example.com/IN: loaded serial 2022070601
OK
```

VI. Reload BIND.

```
# systemctl reload named
```

If you are running BIND in a change-root environment, use the systemctl reload named- chroot command to restart the service.

2. Reverse Zone

Reverse zones map IP addresses to names. For example, if you are in control of the IP range 192.0.2.0/24, you can set up a reverse zone in BIND to resolve IP addresses in that range to hostnames.

Prerequisites

- An example of how BIND is already set up is as a name server that caches results.
- You can see that the named or named-chroot service is active.

Procedure

1. Update the /etc/named.conf file with a zone definition:

```
zone "2.0.192.in-addr.arpa" {
    type master;
    file "2.0.192.in-addr.arpa.zone";
    allow-query { any; };
    allow-transfer { none; };
};
```

Based on these parameters,

- This server serves as the primary, or master, server for the 2.0.192.in-addr.arpa reverse zone.
- You may find the zone file in /var/named/2.0.192.in-addr.arpa.zone. In the options statement, you can give a relative route, as seen in this example. This path will be relative to the directory that you choose.
- Any host can query this zone. In addition, the BIND ACL allows you to restrict access by specifying IP ranges or nicknames. Zoning cannot be performed by any host.
- Zone transfers should only be enabled when secondary servers are being configured, and only for their IP addresses at that.

2. Make sure the /etc/named.conf file is correctly formatted:

```
# named-checkconf
```

If the command does not produce any output, then the syntax is valid.

3. For example, create the following content in the /var/named/2.0.192.in-addr.arpa.zone file:

```
$TTL 8h
@ IN SOA ns1.example.com. hostmaster.example.com. (
                          2022070601 ; serial number
                          1d         ; refresh period
                          3h         ; retry period
                          3d         ; expire time
                          3h )       ; minimum TTL

                  IN NS   ns1.example.com.

1                 IN PTR  ns1.example.com.
30                IN PTR  www.example.com.
```

The following files:

- Sets 8 hours as the default time-to-live (TTL) for resource records. If the number does not include a time suffix, such as h for hour, BIND will treat it as seconds. Include the relevant zone information in the SOA resource entry.
- This command configures ns1.example.com as the primary DNS server for the specified domain. To function, a zone requires an NS record, or name server.
- To meet RFC 1912's criteria, however, two name servers are required.
- This command configures the pointer record (PTR) for the IP addresses 192.0.2.1 and 192.0.2.30.

4. Ensure that only the selected group has access to the zone file:

```
# chown root:named /var/named/2.0.192.in-addr.arpa.zon
# chmod 640 /var/named/2.0.192.in-addr.arpa.zone
```

5. Check the /var/named/2.0.192.in-addr.arpa.zone file to ensure it's correct:

```
# named-checkzone 2.0.192.in-addr.arpa /var/named/2.0.
zone 2.0.192.in-addr.arpa/IN: loaded serial 2022070601
OK
```

6. Reload BIND:

```
# systemctl reload named
```

If you are using BIND in a change-root environment, you can restart the service with systemctl reload named-chroot.

DNS Security

To prevent data theft, service outages, or malicious website redirections, the Domain Name System (DNS) must be secure. It covers protocols for ensuring the validity, accessibility, and integrity of DNS records.

Crucial DNS security measures-Safeguarding the DNS Backbone:

- This involves the deployment of specialized appliances, regular software updates and backups, as well as other severe DNS server security procedures.
- Protecting data transferred using DNS: Encrypting DNS communication with protocols like DNS over TLS (DoT) or DNS over HTTPS (DoH) prevents eavesdroppers from accessing the system.
- Validation and Authentication DNSSEC digitally signs DNS data, ensuring its validity and integrity.
- The use of threat intelligence feeds to filter out potentially hazardous domains and websites. To keep a watch out for unusual activity and potential attacks, DNS logging and monitoring must be configured.

Standard DNS attacks:

- Attackers use DNS spoofing to fool users into visiting malicious websites by altering DNS responses.
- When hackers gain control of a domain name system (DNS) server or record, they are performing DNS hijack.
- Using weak open DNS resolvers, fraudsters can magnify traffic and overwhelm their targets, resulting in a denial of service.
- Attackers can steal data from a compromised system by DNS tunneling, which requires probing the DNS.

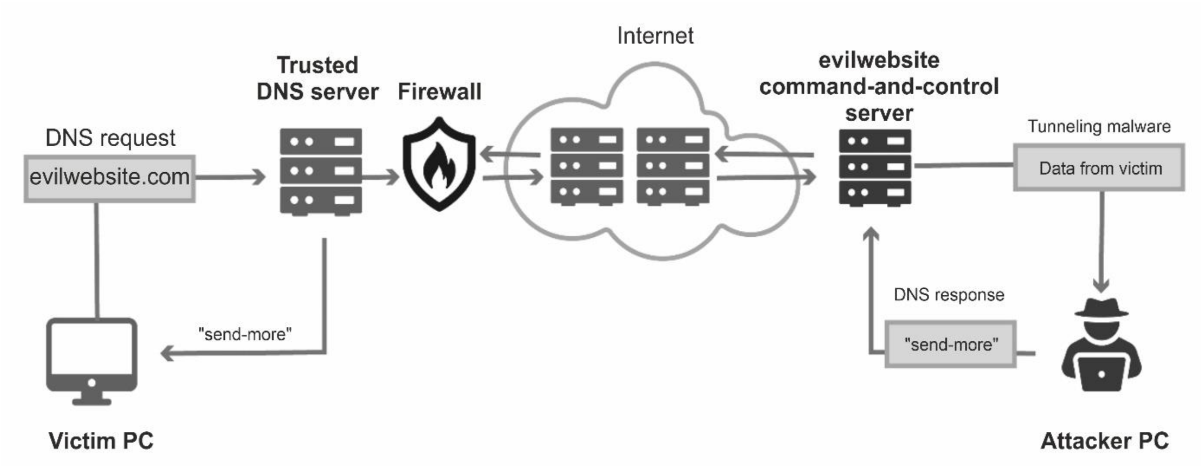


Fig 1.4. DNS tunneling

Conventional thought for DNS security:

- Enabling DNSSEC is crucial for ensuring DNS data validity.
- Implement secure DNS resolvers. Choose a resolver with built-in security features and DNS via TLS or HTTPS.

- Implement DNS filtering on your system: Put an end to any websites or domains that are known to contain malware.
- Keep an eye out for unexpected DNS traffic by examining the logs on a regular basis.
- Inform consumers: Educate people on how to identify and avoid social engineering frauds such as phishing.

Assignment 1.3:
a. Explain the DNS resolution process.
b. Comment on DNS security.

1.4. System Services:

It is necessary to manage system services, which are programs that operate in the background on a computer, in terms of launching, shutting down, and setting their state. The network configuration of the system, which is applied both during startup and during runtime, is what determines the system's capacity to connect to additional networks and interact with those networks. A safe connection to the internet can be obtained through the use of OpenVPN, which is a virtual private network (VPN) tool.

Services are provided by the system

The maintenance of network connections, the provision of security features, and the execution of system duties are all examples of various system services. In the background of a computer, these operations are being carried out.

These are just two examples of network services; others include print spoolers and security services such as firewalls. DHCP clients and DNS resolvers are two examples of network services.

1.5. Managing System Services:

The goal of the System Services Management program is to: At the same time as you are managing the status of services (starting, stopping, and restarting), you are also responsible for controlling the startup behavior of services (for instance, automated startup on boot or manual start). the utilities: command-line applications that can be used to manage services include systemctl (for Linux) and net start/stop (for Windows). Both of these programs are available for usage. The Windows Services applet and the graphical user interface tools that may be found in Linux desktop environments are examples of tools that are included in the category of graphical user interfaces.

1.6. Run time Network Configuration:

The practice of managing network settings while the system is running is referred to as runtime network configuration. This setting management method enables dynamic changes to be made. As an illustration, you are able to make adjustments to the settings of the network interface, the DNS server, and the IP address while the system is operating. Methods: The Network and Sharing Center in Windows, the ip command in Linux, and NetworkManager (on systems that support it) are all examples of tools that can be used to configure networks.

1.7. Boot Time Network Configuration:

When referring to the process of initializing the system's network settings during startup, the term "boot time network configuration" is generally used. It is possible, for instance, that you will be required to establish routing rules, assign static IP addresses, or specify which network interface you will be utilizing. Proceed with: The configuration files that you used to set up your network are analyzed by the system whenever you switch on your computer. This could be found in the system's UEFI or BIOS, the network configuration files that are maintained by NetworkManager, or the /etc/network/interfaces directory on older systems that are based on Debian.

1.8. Introduction to OpenVPN:

OpenVPN is a technology that allows virtual private network connections to be encrypted. Through the construction of tunnels that make use of the SSL/TLS protocols, it is technically capable of encrypting data that is transmitted over the internet. When it comes to connecting to private networks and accessing resources on public networks, its primary objective is to offer consumers a safe and secure method of interaction. Among the notables are: Provides support for both the TCP and UDP protocols.

There are many other authentication mechanisms that can be utilized, such as smart cards, certificates, and combinations of personal identification number and password. VPN configurations supported include client-server and site-to-site configurations. Certificates, keys, and network configurations are some of the components that are shared between the OpenVPN server setup and the client configuration.

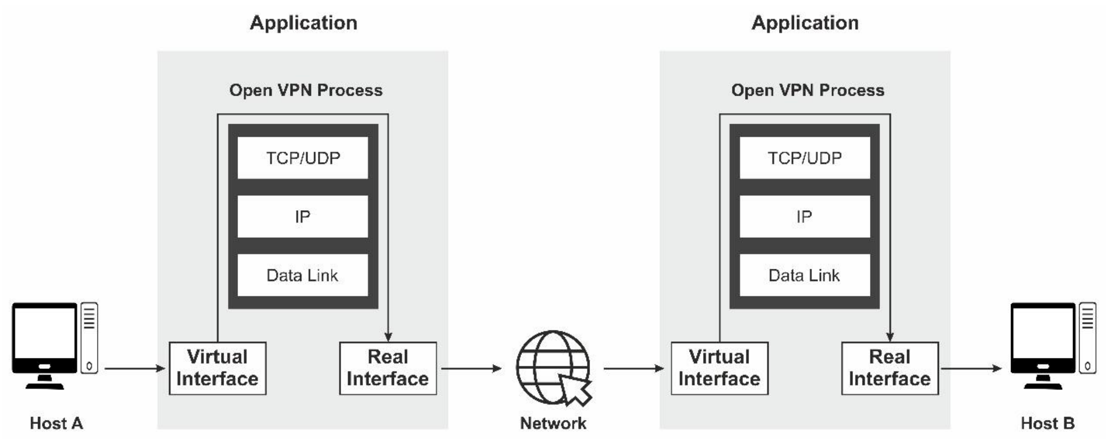


Fig 1.5. OpenVPN

Practical Activity 1.3: Demonstration of DNS and OpenVPN

Scenario: In order to ensure that clients connecting to your OpenVPN server on a Raspberry Pi use your local DNS server for internal resources (such accessing your home server), you may set it up so that it uses a public DNS server for broad web browsing.

- Get everything ready: Since your home IP address could change, it's a good idea to enroll your Raspberry Pi in a dynamic DNS service, such as DuckDNS.
- Create an OpenVPN account: The following would be seen in the client-side configuration file (.ovpn):
- Access the domain name server at duckdns.org: Communicates with the OpenVPN server. dhcp-option may DNS 192.168.1.1: This command tells the client the IP address of the DNS server in your local area network. The register-dns block-

outside-dns command prevents DNS leaks and verifies that the client is using the correct DNS server.

- In addition to configuring OpenVPN to transmit clients their local DNS settings, you must also configure the server to resolve zones for DNS on its end.

Verification

- To confirm that the VPN client is using the correct DNS server, Spiceworks suggests running ipconfig /all on Windows or ifconfig on Linux or macOS after connecting to the VPN. Try accessing both internal and external resources, as well as public websites, to ensure that DNS resolution is functioning properly.

SUMMARY

- OSI model was developed by the International Organization for Standardization (ISO), in order to facilitate communication between various pieces of hardware and software, it partitions network communication into seven separate layers, each of which performs a unique role.
- The physical, data link, network, transport, session, presentation, and application layers are the ones that structure the functions of a network.
- Establishing a common set of protocols and guidelines to enable systems from many suppliers to communicate is the main objective of OSI model.
- OSI model facilitates the creation, testing, and upkeep of network systems by providing a shared vocabulary for communicating across networks.
- The Domain Name System (DNS) is the core directory function of the Internet.
- DNS' principal function is to quickly provide the information required to link users to remote hosts. It is essential for online browsing and the majority of internet activities.
- There is a hierarchical distribution of DNS mapping on the internet. Companies, governments, and educational institutions all have unique IP address ranges and domain names assigned to them.
- BIND, which stands for Berkeley Internet Name Domain, is a collection of tools that includes the DNS server software that is the most widely used in the world.
- The Berkeley campus of the University of California, Berkeley, was the location where the free and open-source software package known as BIND was developed in the 1980s.
- The Berkeley Internet Name Domain (BIND) program maintains a set of DNS zones, and each of these zones has its own text file that provides the configuration information.
- Through the use of forward zones, names are converted into IP addresses and other pertinent data. Through the configuration of a forward zone in BIND, it is possible to resolve names.
- Reverse zones map IP addresses to names. For example, if you are in control of the IP range 192.0.2.0/24, you can set up a reverse zone in BIND to resolve IP addresses in that range to hostnames.
- To prevent data theft, service outages, or malicious website redirections, the Domain Name System (DNS) must be secure. It covers protocols for ensuring the validity, accessibility, and integrity of DNS records.

- OpenVPN is a technology that allows virtual private network connections to be encrypted. Through the construction of tunnels that make use of the SSL/TLS protocols, it is technically capable of encrypting data that is transmitted over the internet.

ASSESSMENT

A. MULTIPLE CHOICE QUESTIONS

1. What is the primary function of DNS?
   a) To store website content
   b) To translate domain names into IP addresses
   c) To manage email routing
   d) To control internet traffic

2. Which command is used to change the ownership of a file or directory?
   a) chmod
   b) chown
   c) chgrp
   d) chattr

3. What is the purpose of the umask command?
   a) To change the default permissions of newly created files and directories
   b) To change the owner of a file or directory
   c) To change the group of a file or directory
   d) To list all files and directories

4. Who is the creator of the Linux kernel?
   a) Linus Torvalds
   b) Richard Stallman
   c) Bill Gates
   d) Steve Jobs

5. Which of the following is a popular Linux distribution?
   a) Windows
   b) Ubuntu
   c) MacOS
   d) Android

6. In Linux, what is the purpose of the 'root' user?
   a) A standard user with limited permissions
   b) The system administrator with full privileges
   c) A guest user
   d) A service account

7. Which command is used to list all files and directories in a Linux system?
   a) ls
   b) dir
   c) list
   d) show
8. Which command is used to display the current working directory?
   a) pwd
   b) current
   c) dir
   d) ls

9. Which command will change the current directory to the home directory?

a) cd ~
b) cd /home
c) cd.
d) cd..

10. Which command is used to create a new directory?
   a) newdir
   b) mkdir
   c) createdir
   d) dircreate

B. Fill in the blanks.

1. ____________model establishes a common set of protocols and guidelines to enable systems from many suppliers to communicate.
2. ____________ is a collection of tools that includes the DNS server software that is the most widely used in the world.
3. _____________is a technology that allows virtual private network connections to be encrypted. Through the construction of tunnels that make use of the ____________ protocols, it is technically capable of encrypting data that is transmitted over the internet.
4. Through the use of____________, names are converted into IP addresses and other pertinent data.
5. To make IP addresses usable by computers, ___________ servers convert domain names and URLs.
6. _____________ layer is responsible for detecting and correcting faults in data transmission between directly connected nodes.
7. _______layer ensures data usability for the application layer by handling data format and encryption.
8. __________ maps IP addresses to names.
9. Attackers use DNS _____________ to fool users into visiting malicious websites by altering DNS responses.
10. Using weak open DNS resolvers, fraudsters can magnify traffic and overwhelm their targets, resulting in a _____________attack.

C. True or False

1. Linux is a single-user operating system.

2. Linux filenames are case-insensitive.

3. The pwd command prints the current working directory.

4. The root user in Linux is identified by the UID of 0.

5. The ls command lists the path to the current directory.

6. Linux is an open-source operating system.

7. The true and false commands in Linux are primarily used in shell scripting.

8. Linux is a proprietary operating system developed by Microsoft.

9. The su command is used to switch user contexts.

10. Every UNIX file structure has a root directory named /.

D. Short Answer Questions

1. Define the OSI model and its purpose.
2. Name the seven layers of the OSI model.
3. What is interoperability in networking?
4. What does the "A" record do in BIND?
5. Explain the role of the SOA record in DNS.
6. What is the difference between runtime and boot time network configuration?
7. Which command is used in Linux to manage system services?
8. Write two uses of OpenVPN.
9. What is the role of the Internet Systems Consortium in BIND?
10. List two limitations of BIND.

Answer Key

A. Multiple Choice Questions

1. b, 2. b, 3. a, 4. a, 5. b, 6. b, 7. a, 8. a, 9. a, 10. b

B. Fill-in-the-blanks

1. OSI, 2. BIND, 3. OpenVPN, SSL/TLS, 4. DNS, 5. DNS, 6. Data Link, 7. Presentation, 8. Reverse DNS, 9. Spoofing, 10. denial of service (DoS)

C. True/False questions

1. False, 2. False, 3. True, 4. True, 5. False, 6. True, 7. True, 8. False, 9. True, 10. True

Chapter-2

Network Troubleshooting and Monitoring

Beena was curious and keen to learn new concepts. One day she visited a book shop in a marketplace where she came across a book on network troubleshooting and monitoring. She learnt in depth concepts on Network Troubleshooting, Client-Side Troubleshooting, Server-Side Troubleshooting, and network monitoring. Then she started helping her friends to identify and fix issues in machines, systems, or processes. With every success, Beena felt happy and understood the power of troubleshooting.

2.1. Troubleshooting

It is common practice to employ troubleshooting, which is a methodical approach to problem solving, to successfully detect and fix machines, systems, or processes. To restore functioning, it is necessary to first determine the precise cause of the issue in a methodical manner and then implement the appropriate solutions. It is necessary to employ a combination of observation, reasoning, and, in some instances, trial and error to effectively troubleshoot a problem.

Steps for troubleshooting

1. Understanding the Problem:

Collect as much information as you can regarding the situation, including any recent modifications, your system settings, and any error messages that may have been displayed.

2. Systematic Approach:

- Getting down to business: Start with the most basic (such as the power supply and connections) and ensure that everything is in working order.
- Identify the problem. Experiment with various parts or processes to determine the root cause of the problem.
- Develop a working hypothesis: Using the information you have gathered, identify the most likely source of the problem.
- Take steps to ensure the accuracy of your theory.

Make a note of what you discover. Make sure you document all the actions, results, and solutions that were implemented.

3. Solving the Problem:

- Execute the repair: Make use of the fix that resolves the issue.
- Check that the problem has been resolved and that the system is running normally.
- Prevent future occurrences: To avoid the problem from recurring, it is best to identify and resolve the root cause, if possible.

4. Seeking Help:

- See the manuals, webpages, or databases for more information.
- Consult with professionals: If you are still unable to solve the problem, consult with a more qualified individual.

The skill to troubleshoot is useful in a variety of fields, including engineering, computer science, customer service, and even everyday life. When dealing with unexpected situations, being knowledgeable in this area can save you time, energy, and annoyance.

2.2. Network Troubleshooting

The term "network troubleshooting" refers to the process of systematically locating and fixing problems that occur contained within a network. It entails identifying the factors that are producing issues, such as poor performance, security breaches, or connectivity problems, and then correcting those issues.

Steps for Network Troubleshooting

Problems and difficulties may arise at any point in the network. To begin troubleshooting, you must first identify the problem, its cause(s), the persons affected, and the time frame of the problem. If you can pinpoint the actual nature of the problem and gather pertinent information, you will be in a much better position to find a solution without wasting time on unnecessary attempts. If you are not sure what the problem is, try these basic network troubleshooting techniques.

i. Verify the hardware

Before you begin troubleshooting, ensure that all your hardware is properly connected, powered on, and operational. A weak wire or someone turning off a critical router could be the source of your networking issues. It is pointless to deal with network troubles when all you must do is plug in a cord. Check that no switches have been accidentally bumped and that they are in the correct locations.

Next, power cycle the device. While this may appear to be a straightforward solution, it is the foundation of IT difficulties. If your computer, router, or modem is having simple troubles, try shutting it off and back on after 60 seconds.

ii. Use ipconfig

Put "ipconfig" (without the quotations) into the terminal when you open the command prompt. If your router is listed last, its IP address is the Default Gateway. You can find your computer's IP address by looking at the number next to "IP Address." Your computer is not obtaining a valid IP address if it starts with 169. Your PC is being assigned a genuine IP address by your router if it starts with a number other than 169.

You can assign a new IP address and remove the old one by entering "ipconfig /release" and then "ipconfig /renew" in that order. In certain instances, this will resolve the issue. If your computer is still not receiving an IP address from your router after trying that, try connecting it directly to your modem using an ethernet cable. Assuming it is functional, the router is at fault.

iii. Use ping and tracert

If your router is functioning properly and your IP address does not begin with 169, then the issue is probably somewhere in the connection between your router and the internet. The ping tool should be used now. If your router is not able to establish a connection, try pinging a popular, huge server like Google's. To access the Google DNS servers, open the command prompt and type "ping 8.8.8.8"; to make it continuously ping the servers while you troubleshoot, add "-t" to the end of the command (ping 8.8.8.8 -t). If the pings are unable to be transmitted, the command prompt will provide basic details regarding the problem.

Typing "tracert 8.8.8.8" into the command prompt will accomplish the same thing; it will display each "hop" that your router takes to reach the Google DNS servers. The issue is occurring at a specific point along the pathway. The problem is probably on your local network if the error appears early in the process.

iv. Perform a DNS check

Try connecting to a different server and seeing whether the problem persists by using the "nslookup" command. Problems with the DNS server for your destination could be the cause of DNS errors like "Timed Out," "Server Failure," "Refused," "No Response from Server," or "Network Is Unreachable" when you try to check your domain name with Google.com, for instance. If you want to verify your own DNS server, you may do that with nslookup as well.

v. Contact the ISP

If none of those solutions work, you could want to check with your ISP to see if they are experiencing problems as well. To find out if anyone else is experiencing the same issue, you can use your smartphone to search for outage maps and associated information.

vi. Virus and malware protection should be in place

The next thing to do is check that your malware and virus scanners are up and running properly and have not detected anything that might be causing your network to malfunction.

vii. Check database logs

Make that the databases are working as they should by reviewing all of the logs. Even if your network is operational, issues with your database, such as an overflow or malfunction, could be impacting the functionality of your network.

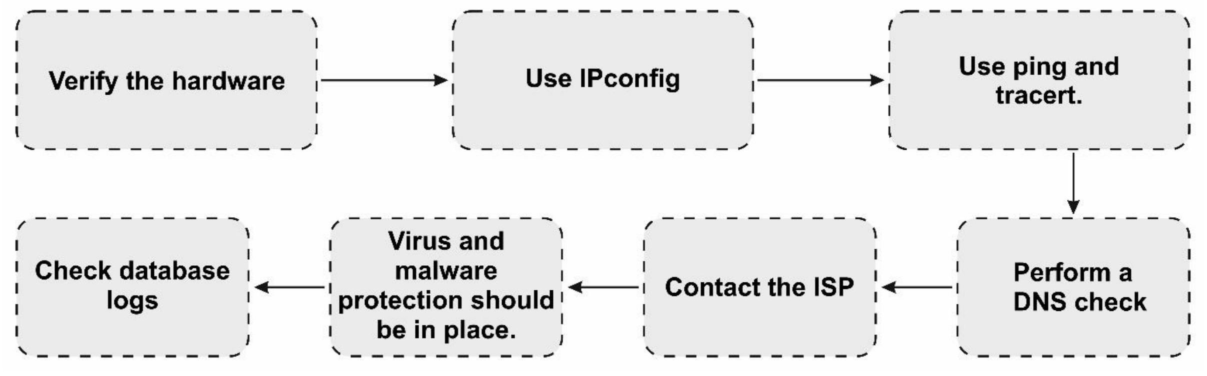


Fig 2.1 Step by step process for Network Troubleshooting

Troubleshooting a network can be a challenging endeavor, even when performing the operation in perfect conditions. The method may be made to run more smoothly with the assistance of well-defined processes, a collection of best practices, and a powerful monitoring tool such as Network Performance Monitor.

Assignment 2.1:
a. Write down the reasons justifying that "Network troubleshooting is important."
b. Explain the role of Network Performance Monitor.

2.3. Client-Server Model

One distributed application architecture is the client-server model, which uses clients to seek resources and servers to offer those resources. Here, the data is usually handled on the server side once a client makes a request for it. Following this, the client receives the requested data from the server.

In most cases, clients will not pool their resources but will instead ask the server to supply them. Both email and the World Wide Web use the client-server architecture; in the former, clients communicate with mail servers, while in the latter, browsers make resource requests to servers.

Working of Client Server model

Client

In this context, a "Client" is any device (often a computer, smartphone, or application) that interacts with a "Server" by requesting and receiving services. The party requesting information or services from the server is known as the client. Common client apps that fetch data from servers in order to display online pages include web browsers such as Google Chrome, Mozilla Firefox, or Safari.

Server

Contrarily, a server is an off-site computer or network that makes information, resources, or services available to users. It takes in requests from clients, processes them, and then returns the necessary data. It is possible for a server to process numerous requests from clients all at once.

Database servers store and provide programs with databases, while web servers house websites. If the requested data or service is already in the server's system, the client just sends a request, and the server fulfils the request.

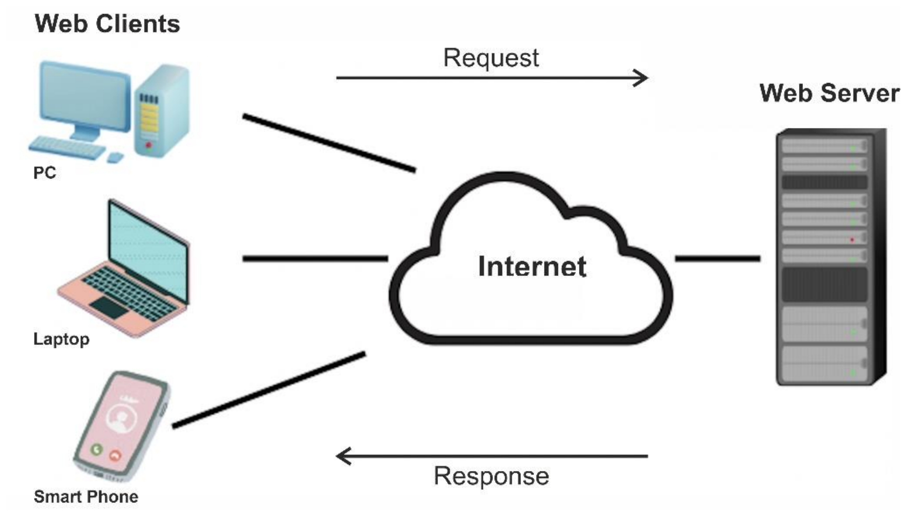


Fig 2.2 Client-Server Computing

Assignment 2.2:
a. Write down a few examples of Client-server models.
b. How does a Client-Server model works.

2.4. Client-Side Troubleshooting

When an issue arises with the front end of an online program and occurs in the user's web browser or device, this type of troubleshooting is known as client-side troubleshooting. Some examples of these issues include incorrect rendering and malfunctioning interactive features. Errors in the code that runs on the client side, such as HTML, CSS, or JavaScript, are the most common culprits.

Common Client-Side Issues:

- **JavaScript Errors:** These are common and can cause scripts to not load, client-side scripts to have syntax mistakes, or to refer to items that do not exist.
- **Rendering Problems:** Pages or elements on the web may not display properly, or they may look distorted or damaged, depending on the browser or device you are using.
- **Interactivity Issues:** Forms and animations, which depend on JavaScript, might not work properly.
- **Performance Issues:** Inefficient JavaScript or excessive DOM manipulation are examples of client-side flaws that can cause slow loading times or sluggish performance.

Troubleshooting Techniques:

- **Browser Developer Tools:** Code inspection, debugging JavaScript, monitoring network requests, and examining web page rendering are all possible with the built-in developer tools that are available in modern browsers (often accessed by hitting F12).
- **Error Logging:** It is common practice for the browser's console to record client-side failures. Developers can analyse these logs to see where the mistake occurred and what caused it.
- **Code Review**: Typos, syntax mistakes, and logical flaws can be found by thoroughly analysing the client-side code, which includes HTML, CSS, and JavaScript.

- **Cross-Browser Testing:** Consistent behaviour and the identification of browser-specific rendering issues can be achieved through testing the application on many devices and browsers.
- **User Feedback:** You may learn a lot about the app's behaviour and any usability issues it may have by asking users for their input.
- **Error Tracking Systems:** Developers can keep an eye out for and fix repeating client-side problems by using error tracking tools (such as Sentry or Rollbar).
- **Performance Optimization:** Improving client-side performance can be achieved by methods such as image compression, code minification, and browser caching.

Consider the following scenario: a user complains that an online form is not submitting properly. With the help of the browser's developer tools, a programmer can make:

1. Launch the console and look for any submission-related error messages.
2. Scan the form's accompanying HTML and JavaScript code for logic and grammar issues.
3. Keep an eye on the network requests to verify the proper transmission of form data to the server.
4. Experiment with various devices and browsers to see if there are any compatibility concerns.

Developers can efficiently fix client-side bugs and provide a dependable user experience by following these strategies in a methodical way.

2.5. Server-Side Troubleshooting

Problems that occur on the server side are caused by the web server that is responsible for hosting the website. It is usual for these issues to originate from the responses that the server provides to requests made by clients (users). A server that is too slow or that provides an inaccurate answer could lead to an undesirable experience for the user.

Common Server-Side Issues:

- **A Slow Response Time from the Server**

  Possible causes include the server being overcrowded, having poor configuration, or experiencing problems with the database.

  An answer is:

  ➢ Optimize the configurations and code of the server (for example, by caching the answers from the server).

  ➢ Change to cloud hosting alternatives that have superior load handling, such as Amazon Web Services or Microsoft Azure, or scale the server resources (RAM and CPU).

  ➢ In order to disperse traffic over multiple servers, load balancers should be utilized. Database queries and indexes should be optimized in order to reduce latency.

- **Error 500: Internal Server Not Found**

  Possible Reasons: Incorrect settings, an excessive number of requests, or broken.htaccess files.

The answer is:

- ➢ Look for specifics of the error in the server logs. Correct any errors in the server's configuration files, such as.htaccess or the Nginx/Apache settings.
- ➢ If the error is caused by server overload, try increasing resources or clearing the cache.

- **The "404 Not Found" Issue:**

  Reason: The sought-after resource is not available on the server.
  The answer is:

  - ➢ Verify the existence of resources on the server and ensure URLs are correct.
  - ➢ Set up individualized 404 error pages and redirection plan for malfunctioning links.

- **Troubleshooting Database Connections:**

  Reason: The server may not be able to establish a connection to the database because either the credentials are incorrect or the database is unavailable.

  The answer is:

  - ➢ See how the database server is doing. Check the credentials and permissions for the connection.
  - ➢ Optimize queries or upgrade hardware to make sure the database is not overwhelmed.

- **Security Flaws (SQL Injection, Distributed Denial of Service Attacks):**

  Reason: Weak server security setups or badly written code.

  The answer is:

  - ➢ To avoid SQL injection, use prepared statements.
  - ➢ To filter traffic and identify potential dangers, set up a firewall.
  - ➢ To prevent distributed denial of service attacks, use a web application firewall (WAF).
  - ➢ Update your server software and plugins regularly to protect it from vulnerabilities.

- **Network Delays or High Latency:**

  The reason for the delay is that the network is either too busy or the users are too far away from the server.

  The answer is:

  - ➢ Distribute static files to users more quickly by using a Content Delivery Network (CDN).
  - ➢ In order to accelerate the delivery of content, optimize the networking settings of your server and make sure that HTTP/2 is used.

- **Issues with Session Management:**

  The reason behind this is that user sessions are not properly managed, which might result in replay assaults or session hijacking.
  The answer is:
  - ➢ Make sure to use secure, HTTP-only cookies that have appropriate expiration dates.
  - ➢ When managing sessions, make use of secure tokens and robust hashing techniques, such as JWTs.
  - ➢ Prevent session fixation attacks by regenerating session IDs following login.

- **Weaknesses in File Upload Security:**

  Reason: Security issues, such as malware or code execution on the server, can arise when users are allowed to upload files without appropriate inspection.

  The answer is:

  - ➢ Check the server and client sides for valid file types and sizes.
  - ➢ You should analyse all submitted files for malware and store them somewhere other than the root site directory.
  - ➢ Restrict file upload sizes and impose rate limits.

- **Data corruption or loss:**

  Reason: Database data loss or corruption can occur due to power outages, hardware issues, or software faults.

  The answer is:

  - ➢ Set up off-site backups on a regular basis.
  - ➢ Make sure your database is replicated correctly and implement storage redundancy using RAID (Redundant Array of Independent Disks) configurations.
  - ➢ When writing data, make use of tools for checking for errors and validating the information.

- **Problems with Authorization and Authentication:**

  Reason: Unauthorized access might occur due to insecure login procedures or inadequate access control.

  The answer is:

  - ➢ Make sure that both users and administrators are using multi-factor authentication (MFA). It is imperative that permits and duties are rigorously upheld.
  - ➢ Choose safe authentication frameworks such as OAuth2.

- **Fatigue of the Memory:**

  Reason: The server's memory is full due to an excessive number of requests or improper management of memory.

  The answer is:

  - ➢ To keep tabs on memory consumption, you can use monitoring tools like as Datadog or New Relic.
  - ➢ To decrease memory utilization, either optimize your program or increase the memory limitations in your server parameters.
  - ➢ To prevent memory leaks, make sure your code uses rate limiting and effective garbage collection.

- **Problems with API Rate Limitation:**

  Reason: If the server gets overwhelmed with API requests, it can cause downtime or performance issues.

  The answer is:

  - ➢ To regulate the number of requests made by each user, set up API rate restriction.

- To cut down on API calls that are not needed, use caching methods. Make use of many servers to distribute API traffic evenly.

- **Problems with the Configuration of the DNS:**

  Reason: The website may become inaccessible or have lengthy periods of outage if the DNS settings are incorrect.

  The answer is:

  - Verify that the A, CNAME, and MX records in your DNS are set up correctly.
  - For better DNS management and optimization, use a service like Cloudflare.
  - Implement a failsafe system by utilizing numerous DNS providers.

- **Certificate Issues with SSL/TLS:**

  Reason: If the SSL certificate is invalid or has expired, your browser may display a warning or even prohibit access.

  The answer is:

  - Keep an eye on SSL/TLS certificates and renew them before they expire.
  - For SSL certificate management, use an automated tool like Let's Encrypt.
  - To make sure your server is properly configuring SSL, you can use tools like SSL Labs to check it.

- **Connections to Databases Are Overloaded:**

  Reason: If there are too many database connections happening at once, the server performance will suffer or perhaps crash.

  The answer is:

  - Reduce the amount of open database connections by using connection pooling.
  - Reduce database demand by optimizing your queries.
  - Raise the maximum number of connections to your database or expand it horizontally if necessary.

Methods for Determining and Fixing Problems on the Server Side:

- With tools like New Relic and Datadog, you can keep tabs on your server's speed, response time, and any bottlenecks as they happen.
- UptimeRobot: Keeps tabs on server availability and uptime and alerts you when there's a problem.
- Logs from Apache or Nginx: When troubleshooting server responses or setups, server logs are priceless.
- By using Fail2Ban, you can prevent brute-force assaults and other forms of server infiltration.
- Tools for monitoring SQL performance, such as MySQL Workbench and pgAdmin, can aid in the diagnosis and optimization of sluggish database queries.
- To keep tabs on server performance and identify configuration problems, use Nginx Amplify, a monitoring tool developed for Nginx servers.

- Loggly is a log management solution that lets you aggregate server logs and keep an eye out for any unexpected patterns, warnings, or failures.
- The Elastic Stack (ELK) is a well-liked collection of tools for managing logs and monitoring server infrastructure. It includes Elasticsearch, Logstash, and Kibana.
- The open-source application Prometheus allows you to keep tabs on server data in real-time, monitor uptime, and find performance bottlenecks.
- The resolution of difficulties on both the client and server sides is required in order to maintain the functionality and user-friendliness of a website. The user experience can be negatively impacted by client-side issues such as delayed page loading, browser incompatibilities, and JavaScript failures, as was mentioned earlier. This can lead to dissatisfaction among users and missed opportunities for businesses. The pace of your website can be significantly slowed down by server-side difficulties such as database errors, server outages, or security vulnerabilities. This can make your website unsecure or even unavailable to users.

However, addressing these issues is not merely a technical effort; rather, it is a vital component of a more comprehensive strategy for raising the level of satisfaction experienced by customers and the level of productivity achieved by businesses. By resolving server-side and client-side issues as soon as they arise, you can make certain that your website runs without any hiccups and earn the trust and reliability of your customers. When you perform preventative maintenance, perform frequent monitoring, and make use of the proper technologies, you can reduce the frequency of these problems, which will allow you to save both time and resources.

"Ensure that you have a strong presence on the internet, work to develop your website, and solve any important concerns! Inboundsys is here to handle issues on both the server and the client side before they have an impact on your corporate operations. For more information on how our professional solutions may assist improve the speed of your website, please get in touch with us as soon as possible.

"Make sure you have a solid presence on the internet, optimize your website, and address any important problems! Before they influence your business, Inboundsys is here to fix problems that occur on the server and the client side. Get in touch with us right away if you are interested in learning how the speed of your website could be improved by utilizing our professional solutions.

Practical Activity 2.1: Create Client-Server Model

The client and server programs must be designed and implemented in a way that allows them to communicate over a network using proper protocols to form a client-server model. When a client makes a request for a service or resource, the server fulfils that request.

Procedure:

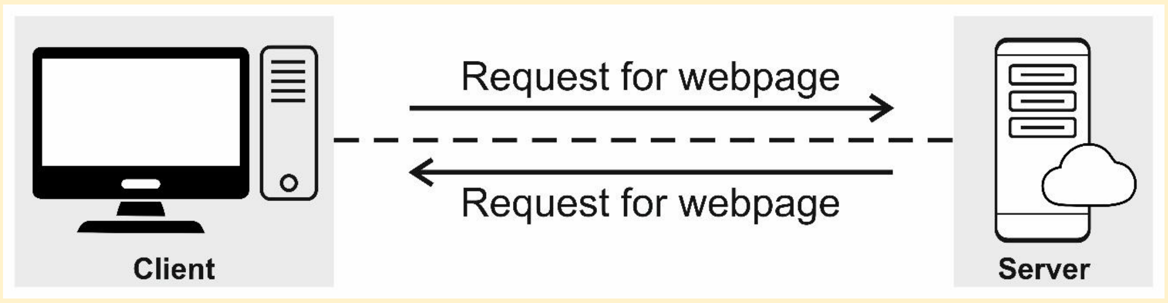
- **Make Your Goals and Scope Clear:** Specify what you want to achieve with your client-server system and the services it will provide.
- **Pick Out Some Technology:** Consider protocols for networking, frameworks for programming, and languages for data transfer (e.g., TCP/IP, HTTP, WebSockets).
- **Build the Server:** Build and deploy the server program, which can include integrating with databases, creating APIs, and processing business logic.
- **Design the user interface** and build the client application or interface, which includes integrating it with the server's application programming interface (API).

- **Set Up Communication:** Make Sure the Client and Server Can Communicate by Setting Up the Network and Protocols.
- **Optimize performance and scalability** after extensive testing to find and fix any problems.
- **Launch:** Launch the server and client apps in a production setting, making sure every part works as it should.
- Keeping an eye on how the system is doing and fixing or updating it as needed is what **monitoring and maintenance** are all about.

Example:

The client-server paradigm can be best represented as a web application.

- To access a webpage, the client—which could be a web browser—sends a request to the server.
- The request is received by the server (e.g., a web server), which processes it and accesses a database or other source to retrieve the required data (e.g., HTML, CSS, JavaScript).
- After receiving the data, the server returns it to the client, which in turn displays the webpage to the user.



Working of web application

2.6. Network Monitoring:

The process of continuously monitoring a computer network for any problems or flaws to guarantee that the network will continue to function properly is referred to as network monitoring, which is also sometimes referred to as network management. Network management and network monitoring are believed to be equal in practice, even though network monitoring can be considered a subset of network management.

To identify frequent network problems, malfunctions in network devices, and resource overloads, the process of network monitoring requires the utilization of specialized systems. This is the case regardless of whether the resources in question are located on-premises, in a data centre, or hosted in the cloud. Devices such as routers, switches, web servers, and firewalls are some examples of common network devices.

Process of monitoring a network.

The term "network monitoring" refers to the process of gathering and relaying information about various components of a computer network. These components include routers, switches, firewalls, load balancers, and even endpoints like servers and workstations. To detect a wide range of network issues and the underlying causes of those issues, the data that has been collected is filtered and evaluated. Device failures, link outages, interface errors, packet loss, application response time, and configuration changes are all examples of the types of network issues that might arise.

From each of the building's components, below are some examples:

- **Events.** The devices that have reported the highest number of events or the events that have occurred most frequently.
- **Change management for network configurations (NCCM)**. Which devices have had the most configuration modifications, or which devices have changed the most essential settings?
- **Performance.** Most heavily used interfaces, interfaces with a high error rate (in terms of both number and percentage of packets), or devices that use a lot of CPU or memory.
- **Handle administration**. Address spaces in some subnets are rapidly becoming exhausted.
- **A study of topology.** Technology that has the greatest number of neighbours undergoes adjustments.
- **Tracking digital experiences**. A list of the systems that have reported apps running slowly or pathways with the most issues.

Keep in mind that many organizations often call for specialized tools for each architectural component. By bringing all the reports together, you can see the network's performance in a nutshell. For network administrators to do in-depth troubleshooting, the reports should include links to the gathered data.

The roles that network monitoring plays

Various subsystems inside a network management and monitoring system carry out distinct but complementary tasks.

1. **Gathering and analysing event data:** The data collected from network events is based on syslog and Simple Network Management Protocol (SNMP) traps. Without polling network devices, the network may now notify administrators of critical occurrences. Reduce the number of warnings that network administrators must deal with by using event processing to detect critical occurrences.
2. **Network configuration and change management**: You can automate configuration updates and store network device configurations with NCCM. The command-line interface (CLI), simple network management protocol (SNMP), RESTCONF, and NETCONF are among the methods that can be used to retrieve and change configurations.

   Everyday changes (drift) and audit compliance exceptions (when settings do not fit network design policies) can be found through configuration analysis.

   To guarantee that the settings of a network are in accordance with its intended purpose and operation, the drift and audit capabilities are required.
3. **Tracking efficiency**: Data on the device's performance, including its temperature, power supply voltages, fan activity, CPU, and memory utilization, and so on, is gathered through performance monitoring. To detect issues like congestion, failures, and packet loss, interface performance data is utilized.

   Tools like Windows Management Instrumentation (WMI), SNMP, the command line interface (CLI), and telemetry are used to gather data. While Windows-based devices use the WMI remote protocol, network devices and Linux-based endpoints usually use SNMP or telemetry. Using the Common Information Model—a client-server framework—Windows Management Instrumentation (WMI) allows system administration. The model represents the OS's components.

4. **Network Telemetry:** To transmit data about the network's functioning to a monitoring system, newer devices and systems may use network telemetry. The data used by telemetry can be encoded in either JavaScript Object Notation or Extensible Markup Language. To gather data in these same formats, certain network monitoring systems and associated network devices employ representational state transfer interfaces.
5. **Management of Internet Protocol addresses**: The assignment of IP addresses to devices on a network and their use can be monitored and controlled by an IP address management system. In most cases, this feature communicates with other network management systems through an API or command line interface (CLI).
6. **Mapping the topology:** The physical and logical topology maps are created by the topology and mapping function using data collected from device connections. These maps serve as the basis for basic troubleshooting. Data about routing neighbours (Layer 3), switching neighbours (Layer 2), address translation tables (Layer 2 to Layer 3 mapping), and neighbour discovery methods (such Link Layer Discovery Protocol) can be obtained by SNMP polling or the command line interface (CLI).
7. **Tracking digital experiences:** The purpose of digital experience monitoring is to ensure that the network is functioning properly by using active testing tools like ping, traceroute, and synthetic monitoring. It may also use software agents installed on endpoints (such servers and workstations) to gather information regarding the efficiency of applications and networks. Information technology companies can determine if an application issue is network-related or caused by something else, such as external networks, by integrating application performance monitoring with network monitoring.
8. **Automation and security:** Security and automation should permeate the entire architecture. Network security is still crucial to a well-functioning network, and automation ensures that policies are consistently applied. Devices for intrusion detection and prevention, as well as software for their monitoring and management, should be part of the security architecture. Tools that offer automation might be standalone or integrated into an NCCM system.

The advantages of network monitoring are:

- **Less downtime:** Proactive monitoring helps find and fix problems before they affect users, which lowers downtime and makes the network more available overall.
- **Better Resource utilization:** Network monitoring helps keep networks running at their best by finding bottlenecks and making sure that resources are used in the best way possible.
- **Better Security:** Monitoring can find efforts to get in without permission, strange traffic patterns, and other security issues, which can assist stop cyberattacks.
- **More efficient:** Automated monitoring and alerting make the troubleshooting process easier, so administrators may spend less time putting out fires and more time on strategic duties.
- **Better Visibility and Control:** Network monitoring gives you a centralized view of the whole network infrastructure, which helps you understand how the network works and gives you more control over its resources.

Network monitoring: Best Practices

- Make sure your goals and standards are very clear.
- Ensure thorough monitoring is carried out.

- Get proactive with your monitoring.
- Put important resources first
- Keep monitoring setups up-to-date by reviewing them often. Review facts from the past.
- Establish a system for tracking security
- Educate the group.

In short, network monitoring is an important IT task that keeps computer networks running smoothly, reliably, and securely by always being able to see what is going on and fixing problems before they happen.

Practical Activity 2.2: Demonstration using packet tracer of the following

- Network Troubleshooting
- Client-Side Troubleshooting
- Server-Side Troubleshooting
- Network Monitoring

Materials Required

- Computer with Cisco Packet Tracer
- Devices in Packet Tracer: 1 Router, 1 Switch, 2 PCs, 1 Server

Steps

Step 1 – Create the Network

1. Open Packet Tracer.
2. Drag and drop: 1 Router, 1 Switch, 2 PCs, and 1 Server.
3. Connect them with straight cables.
4. Save your file.

Step 2 – Assign IP Addresses

1. Give Router IP: 192.168.10.1
2. PC0: IP 192.168.10.10, Gateway 192.168.10.1
3. PC1: IP 192.168.10.11, Gateway 192.168.10.1
4. Server: IP 192.168.10.50, Gateway 192.168.10.1

Step 3 – Network Troubleshooting

1. From PC0, open Command Prompt → type ping 192.168.10.50.

o If reply comes → network is fine.

2. To create an error → change PC1's IP to 192.168.20.11.
3. Again try ping. It will fail.
4. Correct the IP back to 192.168.10.11.
5. Now ping will succeed.

Step 4 – Client-Side Troubleshooting

1. On PC0, set wrong gateway (e.g., 192.168.1.1).
2. Try ping 192.168.10.50 → it will fail.
3. Check IP using ipconfig.

4. Correct gateway to 192.168.10.1.
5. Test again → works.

Step 5 – Server-Side Troubleshooting

1. Go to Server → Services → HTTP.
2. Switch OFF the HTTP service.
3. On PC0, open Web Browser → type http://192.168.10.50.

o It won't open.

4. Turn HTTP service ON again.
5. Refresh browser → page opens.

Step 6 – Network Monitoring

1. Switch Packet Tracer to Simulation Mode (bottom-right).
2. From PC0, open browser → type http://192.168.10.50.
3. Click Play / Step to watch packets moving across devices.
4. Observe: ARP requests, TCP connection, HTTP request &amp; response.

SUMMARY

- The term "network troubleshooting" refers to the process of systematically locating and fixing problems that occur contained within a network.
- One distributed application architecture is the client-server model, which uses clients to seek resources and servers to offer those resources.
- Client" is any device (often a computer, smartphone, or application) that interacts with a "Server" by requesting and receiving services.
- Common client apps that fetch data from servers in order to display online pages include web browsers such as Google Chrome, Mozilla Firefox, or Safari.
- A server is an off-site computer or network that makes information, resources, or services available to users. It takes in requests from clients, processes them, and then returns the necessary data.
- When an issue arises with the front end of an online program and occurs in the user's web browser or device, this type of troubleshooting is known as client-side troubleshooting.
- Common client-side issues include Java script errors, rendering problems, interactivity issues and performance issues.
- Problems that occur on the server side are caused by the web server that is responsible for hosting the website. It is usual for these issues to originate from the responses that the server provides to requests made by clients (users).
- Common server-side issues include slow server response, "Error 500": Internal Server Not Found, "404 Not Found" Issue, Database Connections issues and SQL Injection, Distributed Denial of Service Attack.
- The process of continuously monitoring a computer network for any problems or flaws to guarantee that the network will continue to function properly is referred to as network monitoring, which is also sometimes referred to as network management.

ASSESSMENT

A. Multiple Choice Questions

1. In isolation technique each stage is isolated from the end point on by one in troubleshooting of power supply unit.
   a) True
   b) False

2. A technician is troubleshooting a slow network connection. Which tool would be MOST useful for identifying bottlenecks?
   a) Network sniffer
   b) Protocol analyser
   c) Network monitoring tools
   d) All of the above
   e) None of the above

3. A user reports that they can connect to the internet but cannot access a specific website. What should the technician check first?
   a) The user's IP address
   b) The user's default gateway
   c) The website's DNS records
   d) The user's firewall settings
   e) The user's antivirus software

4. What does "BSOD" stand for in the context of computer errors?
   a) Blue Screen of Death
   b) Bad Sector on Disk
   c) Broken Software on Drive
   d) Basic System Overload

5. When troubleshooting network connectivity issues, what should you check first?
   a) Keyboard and mouse
   b) Monitor settings
   c) Internet Service Provider (ISP) status
   d) Ethernet or Wi-Fi connection

6. If you receive a "No bootable device" error message on startup, what should you check first?
   a) Monitor connection
   b) Hard drive status
   c) RAM modules
   d) Keyboard functionality

7. What is the purpose of "Safe Mode" in Windows troubleshooting?
   a) To run the computer at maximum performance
   b) To test hardware components
   c) To start Windows with a minimal set of drivers and services
   d) To uninstall software

8. What is the purpose of "System Restore" in Windows troubleshooting?
   a) To delete system files
   b) To restore the computer to factory settings
   c) To undo system changes to a previous state
   d) To update device drivers

9. If your computer is displaying a "No signal" message on the monitor, what should you check first?
   a) RAM modules
   b) Hard drive status
   c) Monitor cable connections
   d) CPU temperature

10. What should you do if your computer's sound is not working?
   a) Update the keyboard drivers
   b) Check the monitor settings
   c) Verify the speakers or headphones are properly connected
   d) Replace the graphics card

B. Fill-in-the-blanks

1. The term ______________refers to the process of systematically locating and fixing problems that occur contained within a network.
2. One distributed application architecture is the___________, which uses clients to seek resources and servers to offer those resources.
3. A _________ is any device (often a computer, smartphone, or application) that interacts with a "Server" by requesting and receiving services.
4. A ____________is an off-site computer or network that makes information, resources, or services available to users.
5. The process of continuously monitoring a computer network for any problems or flaws to guarantee that the network will continue to function properly is referred to as _____________.
6. _____________ is a powerful simulation tool that allows users to design, configure, and troubleshoot network environments.
7. Common client-side issues include ________________________.
8. Common client apps that fetch data from servers in order to display online pages include web browsers such as_______ and ___________.
9. When an issue arises with the front end of an online program and occurs in the user's web browser or device, this type of troubleshooting is known as _____________.
10. Common server-side issues include_____, ______ and ______.

C. True/False questions

1. After a problem is resolved, there is no further action you need to take.
2. A switch is more intelligent than a hub.
3. A bus topology is easy to maintain and troubleshoot.

4. A router is a device that connects two or more networks.
5. The physical layer is the layer closest to the transmission medium in the OSI model.
6. Wireless networks are usually used in small local area networks (LANs).
7. All computers on a network need a unique IP address.
8. Parity bits are used to encrypt data.
9. A patch is a replacement for an entire software package.
10. Attackers can steal data from a compromised system by DNS tunnelling, which requires probing the DNS.

D. Short Answer Questions

1. What is the primary goal of troubleshooting?
2. Name the first step in the systematic troubleshooting process.
3. What is the purpose of using the "ipconfig" command during network troubleshooting?
4. What does a client do in a client-server model?
5. Give two common examples of client-side issues.
6. What is the main function of a server in a network?
7. Why is network monitoring important for organizations?
8. Name two tools or commands used to check network connectivity.
9. What should a technician do if the DNS server is suspected to be the problem?
10. Define client-side troubleshooting.

Answer Key

A. Multiple Choice Questions

1.a, 2.d, 3.c, 4.a, 5.d, 6.b, 7.c, 8.c, 9.c, 10.c

B. Fill-in-the-blanks

1. Network troubleshooting, 2. Client-server model, 3. Client, 4. Server, 5. Network monitoring, 6. Packet tracer, 7. Slow performance, connectivity problems, application errors, 8. Chrome, Firefox, 9. Client-side troubleshooting, 10. Hardware failures, software errors, configuration issues

C. True/False questions

1. False 2. True 3. False 4. True 5. True 6. True 7. True 8. False 9. False 10. True

Chapter-3

Remote Access

Isha was a class XII student. One she wanted to access her system for some urgent data requirement but could not access it, which due to a lack of its physical presence. She shared this issue with one of her friends, Rohan. Rohan told her about the concept of remote access. They both went to library and read some books on this interesting concept.

In many cases, when you do not have physical access to your desktop, the cloud is not sufficient. Take this hypothetical situation: you require access to a specific file but are unable to upload it to the cloud because of business policy.

Perhaps you require the use of a program that can only be installed on a specific computer. This might be since the software is only valid on your work computer or that the resource-intensive program you require cannot be executed on any other workstations save your desktop.

Working remotely is never easy, but it becomes much more difficult when you don't have access to a computer. We really wish we could take our computer with us everywhere we go because of this.

Fortunately, you do not even need to bring your computer with you because remote access makes it possible. A remote access tool, such as Splashtop, will allow you to safely connect to the target computer and manage it from another device as if you were physically there. Companies can take advantage of this technology to let their employees work remotely. It can be used to improve distance learning in schools and institutions.

In this Chapter, we will learn about remote access, its types, secure remote access and remote graphics.

3.1. Remote access

When you have remote access, you can log in to a computer or other device from another device, and you can do so from anywhere in the world, at any time. The ability to access a remote computer and all its data and programs from another device and manage it as if you were physically present is what remote access software is all about.

A few examples of typical remote access applications are:

- **Continuity of Business and Remote Work**: Workers have the ability to safely access company computers from any location.

- **Support and troubleshooting from IT departments:** IT staff can identify problems and implement solutions remotely.
- **Training and Education**: Instructors and educators can offer online help and demonstrations.
- **Remote server and system monitoring and management** is now possible for IT admins.

Working of Remote access

The use of software, hardware, and an active network connection allows for remote access. Before internet access became widely available, for instance, traditional remote access meant controlling access through a physical modem linked to a telephone network using terminal emulation software.

These days, the most typical methods for achieving remote access are:

- **Software.** By employing a VPN or other secure software solution.
- **Pieces of hardware**. Linking hosts using a wireless or hardwired network connection.
- **Connected devices.** Linking up over the web.

Assignment 3.1:
- What is remote access?
- Write down an example of remote access.

3.2. Ways to get remote access

Different remote access technologies, such as virtual private networks (VPNs), desktop sharing (VNCs), and remote desktop protocols (RDPs), make it possible for users to access and completely operate the computer of another user from a remote location. On the other hand, other technologies block users from sharing resources or accessing them.

Virtual private network (VPN)

Users have the ability to encrypt and decrypt data that is sent and received between devices or across public networks when they utilize a virtual private network, also known as a VPN. Both users need to be connected to the same virtual private network (VPN) and use the same access software in order to access the computer of another user. With the help of the Dynamic Domain Name System (DDNS), domain names are constantly mapped to changing IP addresses. This makes it possible for networks that use dynamic IP addresses to continually communicate with one another.

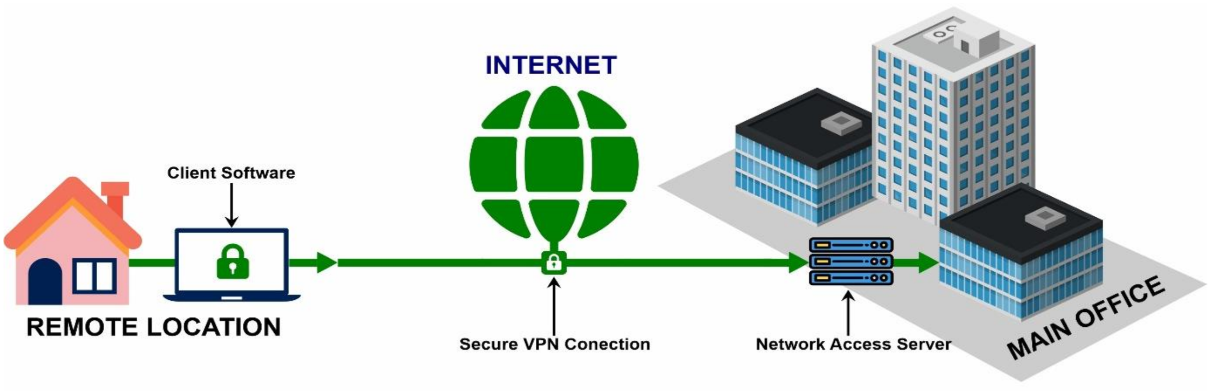


Fig 3.1. Remote access VPN

Virtual Network Computing (VNC)

The graphical user interface for desktop sharing is provided via VNC, which stands for virtual network computing. Through the process of sending the user's input to the other device, the remote user is able to manipulate it as if they were physically present, while the other user is able to view what the remote user is doing on their screen.

Remote Desktop Protocol (RDP)

Among several protocols is the remote desktop protocol, also known as RDP. RDP, which stands for Remote Desktop Protocol, is a Windows application that allows users to connect two computers to one another through the use of a graphical user interface. During the time that the RDP application is being run on the computer of the other user, the user is in charge of the RDP client software.

Proxy Server

Proxy servers that are secure to use online. An intermediary computer acts as a go-between for you and another computer that is part of the proxy server environment when you use an internet proxy server like the one described above. By connecting both workstations to the same proxy server, a user is able to acquire access to the computer of another user.

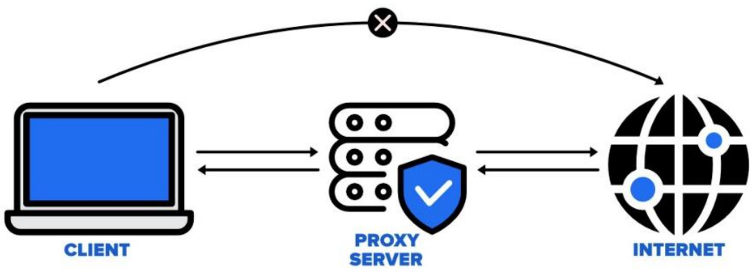


Fig 3.2 Proxy Server

Practical activity 3.1: Network establishment using remote access

Example 1: Initial Configuration of a New Router via SSH

Scenario: A new branch office has received a pre-configured Cisco router (with basic management IP and SSH enabled). You are in the main office and need to configure its interfaces, routing, and security policies remotely.

Steps:

Step 1: On the Remote Router (Physical or Initial Access):

1) Initially, someone at the branch office (or a pre-provisioning team) connects to the router via console cable and configures a management IP address on an interface (e.g., GigabitEthernet0/0 with 192.168.1.10/24) and enables SSH. This is the only physical touch needed.

   - configure terminal
     interface GigabitEthernet0/0
     ip address 192.168.1.10 255.255.255.0
     no shutdown
     ip domain-name mycompany.com
     crypto key generate rsa (generate RSA keys for SSH)

```
line vty 0 4
transport input ssh
login local
username admin privilege 15 secret myStrongPassword
exit
```

2) Ensure the router has basic internet access to be reachable from your location.

Step 2: From Your Location (Main Office) using SSH:

1. Open an SSH client (e.g., PuTTY on Windows, ssh command on Linux/macOS).
2. Connect to the router's management IP: ssh admin@192.168.1.10
3. Enter the password myStrongPassword.
4. Now you are remotely establishing the network:
5. Configure additional interfaces:

-
  ```
  configure terminal
  interface GigabitEthernet0/1
  ip address 10.0.0.1 255.255.255.0
  no shutdown
  description "Connection to Branch LAN"
  exit
  ```

Step 3: Set up static routes or dynamic routing protocols (OSPF/EIGRP):

- ip route 0.0.0.0 0.0.0.0 192.168.1.1 (default route to internet)
- router ospf 1
- network 10.0.0.0 0.0.0.255 area 0
- network 192.168.1.0 0.0.0.255 area 0

Step 4: Implement security (ACLs/Firewall rules):

- ip access-list extended ALLOW_BRANCH_TRAFFIC
- permit ip 10.0.0.0 0.0.0.255 any
- deny ip any any log
- interface GigabitEthernet0/1
- ip access-group ALLOW_BRANCH_TRAFFIC in
- Save configuration: write memory or copy running-config startup-config

Example 2: Setting up a Windows Server Role via RDP

Scenario: You need to deploy a new DHCP server on a Windows Server 2022 instance located in a data center. The server has a basic OS installation and network connectivity.

Steps:

1) On the Windows Server (Remote Data Center):

- Ensure RDP is enabled. This is often done during initial server provisioning or can be enabled via console access if absolutely necessary.
- Ensure the server has a static IP address and can communicate on the network.

2) From Your Workstation (Using RDP Client):

- Open "Remote Desktop Connection" (mstsc.exe).
- Enter the IP address or hostname of the Windows Server (e.g., 192.168.50.10).
- Enter administrator credentials.

3) Now you are remotely establishing the network service:

4) Once connected, the server's GUI appears.

5) Open "Server Manager."

6) Click "Add Roles and Features."

7) Select "DHCP Server" and proceed through the wizard to install the role.

8) After installation, complete the DHCP configuration (scope creation, exclusions, lease duration, DNS servers) directly through the server's GUI.

Example 3: Configuring a Wireless Access Point (WAP) via Web Interface

Scenario: You've deployed new Cisco Meraki WAPs in a remote office. They are powered on and connected to the network. You need to join them to your Meraki Dashboard (cloud-based network establishment).

Steps:

1. **On the Meraki Dashboard (Cloud Service):**
   - Log in to your Meraki Dashboard from your web browser.
   - Navigate to "Network-wide" > "Add Devices."
   - Enter the serial numbers of the new WAPs.
   - Assign them to the correct network configuration profiles (which include SSIDs, VLANs, security settings).
2. **On the WAP (Initial Network Connection):**
   - The WAP powers on, obtains an IP address via DHCP (typically).
   - It then automatically connects to the Meraki cloud and downloads its configuration from the dashboard.

Example 4: Establishing a VPN Tunnel to Secure Remote Access

Scenario: You need to securely access network resources in a remote data center (e.g., a server for configuration or a specific network segment) over the public internet.

Steps:

1. **On the Data Center Firewall/Router (Remote Network Establishment):**
   - Someone (or a prior configuration) configures the VPN server role on the firewall/router. This involves defining VPN users/groups, authentication methods (pre-shared key, certificates), and encryption protocols (IPsec, SSL VPN).
   - Example CLI (conceptual for a firewall):
   - vpn-server enable
   - vpn-profile MY_VPN_PROFILE
   - authentication pre-shared-key MySuperSecretKey
   - tunnel-protocol ipsec
   - remote-access-users LOCAL_VPN_USERS

2. **On Your Workstation (Remote Access Client):**
   - Install and configure a VPN client (e.g., Cisco AnyConnect, OpenVPN client, Windows built-in VPN client).
   - Enter the public IP address of the data centre’s firewall/router and your VPN credentials.
   - Establish the VPN connection.

3. **Post-VPN Establishment:**
   - Once the VPN is connected, your workstation is logically "inside" the data centre network.
   - You can now use SSH, RDP, or web interfaces to securely configure any other devices (servers, switches, firewalls) within that data centre, as if you were physically connected to its LAN.
   - Benefit: VPN is the cornerstone of secure remote network establishment, providing a trusted tunnel over untrusted networks.

4. **Security Considerations for Remote Access**

   While incredibly convenient, remote access introduces security risks if not properly managed. When establishing networks remotely, always prioritize:
   - **Strong Authentication:** Use complex passwords, multi-factor authentication (MFA) whenever possible.
   - **Encryption:** Always prefer SSH over Telnet, HTTPS over HTTP, and use strong VPN protocols.
   - **Least Privilege:** Grant only the necessary permissions to remote access users.
   - Network Segmentation: Isolate management networks from production data networks.
   - **Firewall Rules**: Restrict remote access to specific source IP addresses and ports.
   - **Logging and Auditing:** Monitor and log remote access attempts and activities to detect suspicious behaviour.
   - **Regular Updates:** Keep remote access software, operating systems, and device firmware updated to patch vulnerabilities.

Advantages of remote access

1. When employees have access to remote access, they are able to work from any location, whether they are traveling, working from home, or at a client's site. This provides them with increased flexibility and mobility. Increased productivity and overall job happiness are two potential outcomes of this freedom.
2. By allowing employees to work from home, businesses can reduce the expenses that are connected with office space, utilities, and other overhead expenses. In addition, employees conserve both time and money by not having to commute.
3. The ability to access files and apps in real time is made possible by tools such as remote desktop software, which boosts productivity and makes it simpler to maintain productivity.
4. Remote access ensures that corporate activities can continue without severe disruptions in the case of a disaster or other unexpected occurrence. This improves the continuity of business operations. With the ability to access their work environment from secure areas, employees are able to retain their productivity.
5. Companies do not have to restrict themselves to recruiting local talent; they have access to a larger talent pool. Using remote access, they can attract the most qualified people from any location in the world, thereby contributing a wide range of abilities and points of view to the team.

Limitations of remote access

1. Security risks – Vulnerable to hacking, malware, and unauthorized access if not properly secured.
2. Internet dependence – Requires a reliable, high-speed connection for smooth operation.
3. Performance issues – Can face latency, lag, or slow data transfer.

4. High maintenance – Needs regular updates, monitoring, and management.
5. Compatibility limits – Some applications or devices may not work efficiently over remote access.

3.3. Cryptography

Cryptography is the science of protecting information by turning it into a secret code so that only authorized people can read it. It changes a normal message (plaintext) into an unreadable form (ciphertext) using a process called encryption. The ciphertext can only be changed back (decryption) if you have the correct key.

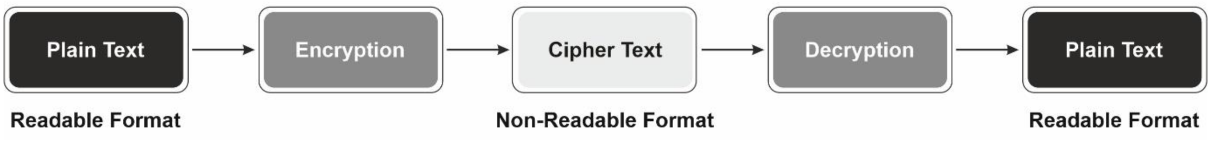


Fig 3.3 Cryptography

Public Key and Private Key

In Asymmetric Cryptography, two different keys are used:

Public Key:

- This key is shared openly with everyone.
- It is used to encrypt data.
- Example: If you want to send me a secret message, you use my public key to lock (encrypt) it.

Private Key:

- This key is kept secret and only known by the owner.
- It is used to decrypt data.
- Example: After you send me the encrypted message, only I can open (decrypt) it with my private key.

Data transported between devices must be encrypted using cryptography to prevent unauthorized access, which is why it is vital for safeguarding remote access. Secure shell (SSH), virtual private networks (VPNs) (IPsec and SSL/TLS), and encrypted web access all use encryption to keep private data safe while in transit over the internet.

Take a closer look at this:

1**. SSH encryption: Secure Shell (SSH):** allows users to remotely access equipment across untrusted networks in a safe and secure manner. Managing Unix-like systems remotely and transferring files securely are two of its most common uses.

2. Virtual Private Networks (VPNs): To safeguard data while it is in transit, VPNs construct an encrypted tunnel over a public network, such as the internet. Virtual private networks (VPNs) that use IPsec provide strong security for connections at the network level, whereas VPNs that use SSL/TLS provide safe access via web browsers.

3. Web-based Remote Access: A lot of systems encrypt data sent during remote access sessions using web-based interfaces with HTTPS. This makes sure that the communication route remains secure even when devices are accessed through a browser.

4. Encryption's Significance: A protective layer, encryption prevents data interception and unauthorized access while distant sessions are in progress. Key lengths of various algorithms, such as AES (Advanced Encryption Standard), provide varied degrees of security (e.g., 128-bit, 256-bit). For strong security, it is essential to have secure key management and strong encryption techniques.

5. RDP: RDP is a protocol that allows users to access their desktops remotely; it frequently uses encryption to ensure the security of the connection. TLS/SSL encryption is one way to protect RDP; further security methods to consider include strong passwords, two-factor authentication, and authentication at the network level.

6. Data-at-Rest Encryption: To further safeguard information from loss or theft, it is crucial to encrypt data kept on remote devices as well as data in transit.

3.4. Secure remote access

No matter the protocol you pick, keep in mind that your computer will be accessible to other users whenever you engage in remote access. Given the ease of file communication between computers, there is always the risk of malware transmission or unwanted access by an unauthorized third party.

For safe remote access, follow these guidelines.

- Ensure the safety of every device participating in the remote connection by utilizing endpoint security. Security measures such as firewalls and antivirus software are usually part of this.
- Make use of a secure browser: Both users are at risk when using public Wi-Fi. With a reliable and secure connection, you may establish a direct link that keeps unwanted users out.
- Be sure to use lengthy passwords that contain a mix of uppercase and lowercase letters, digits, and symbols. Install MFA (multi-factor authentication) tools:
- An additional hurdle for a malicious actor to overcome in order to obtain access is multi-factor authentication, which combines traditional login credentials with biometric data or SMS. Deploy a policy to prevent unauthorized access to your account:
- An account lockout policy prevents a user from attempting to connect after a specific number of incorrect password entries.
- Be sure to update your software regularly: Updating your software is one way to protect your computer from newly discovered viruses and other forms of malware.
- Reduce the maximum number of users: The greater the number of users, the greater the opportunity for malicious actors to gain access. Reducing the user base lessens the likelihood of intrusion.

3.5. Remote Graphics

With remote access graphics, you may view and manipulate the visual output of a distant computer. Working on a computer from afar is typically accomplished over a network connection. To do this, users often employ remote desktop protocols (RDP) or specialized software that grants access to the resources and processing capacity of a remote system, which is particularly useful for graphically intensive jobs.

Remote Graphics working

- **Remote Desktop Protocol:** One way to access a remote computer's graphical user interface is through the usage of Remote Desktop Protocol (RDP), which was created by Microsoft. One computer processes the user's activities and sends back an updated display; the other computer operates as a server. The user's computer displays the remote desktop.
- **Expert Software**: Graphic design-specific remote access tools include TeamViewer and Splashtop, which facilitate file sharing, teamwork, and, in some cases, GPU acceleration. Using the graphics processing unit (GPU) of the distant system, certain remote access solutions can speed up graphics rendering, making applications like video editing and 3D modeling much more efficient.
- **Graphics Encoding:** In order to compress and send the visual data across the network, protocols like RDP require encoding techniques. This procedure is optimized for various workloads by modern protocols such as RDP's Graphics Pipeline Extension.

Possible applications:

- **Graphic Design:** With remote access, designers may access powerful workstations or servers from any location, allowing them to work on projects from anywhere. CAD/CAM: Engineers and architects can collaborate on intricate 3D models and designs from diverse places using remote access and powerful workstations.
- **Software Development**: Remote access allows developers to access servers and development environments, which improves workflow efficiency and fosters collaboration. For gamers, remote access means they can enjoy graphically demanding games on more powerful gaming PCs from any device, regardless of how powerful their home system is.

Features:

- **Functionality**: Latency may be introduced by remote access, particularly when dealing with high-resolution graphics or sluggish network connections.
- **Security:** Accessing remote machines, particularly through public networks, requires secure connections and authentication.
- **Software Compatibility:** It is critical to select the correct tool due to software compatibility issues, as not all remote access solutions support all graphical APIs or hardware combinations.
- **Licensing:** Compliance with the conditions of any applicable commercial licenses is mandatory while using some remote access technologies.

Assignment 3.2:
a. What are the uses of remote graphics.
b. Discuss some important application of remote access

Practical Activity 3.2

i) Demonstration of Secure Remote Access. ii) Demonstration of remote graphics

Materials Required

- Router (with SSH support)

- PC/Laptop with terminal/command prompt
- Ethernet cable
- Router console/command line access (for initial setup)

Steps:

- Connect the PC/laptop to the router using an Ethernet cable.
- Assign IP addresses:
- Router: 192.168.1.1
- PC: 192.168.1.10 (same network).
- Configure router for SSH (on router console):
- Set hostname → hostname Secure_Router
- Set domain name → ip domain-name school.local
- Generate RSA keys → crypto key generate rsa
- Create user → username admin password Secure@123
- Enable SSH on VTY lines → line vty 0 4, login local, transport input ssh
- Save configuration → write memory.
- On PC, open Command Prompt/Terminal.
- Type: 'ssh admin@192.168.1.1'
- Enter password → Secure@123.
- Router CLI will open securely via SSH.
- Run command show ip interface brief to verify access.

ii) Demonstration of remote graphics

Materials Required

- Two computers (PC-A as server, PC-B as client)
- Local network connection (LAN/Wi-Fi)
- Remote desktop software (e.g., Remote Desktop Connection in Windows, VNC Viewer, or AnyDesk)
- User credentials for remote login

Simple Steps

1. Setup the Remote Computer (PC-A / Server):
   - Enable Remote Desktop (Windows: Settings → System → Remote Desktop).
   - Note down the IP address of PC-A (use ipconfig).
   - Create or use an existing user account with a password.
2. Prepare the Client Computer (PC-B):
   - Open Remote Desktop Connection (Windows: search mstsc).
   - In the Computer field, enter the IP address of PC-A.
3. Login Remotely:
   - Enter the username and password of PC-A when prompted.
   - A graphical desktop session of PC-A will appear on PC-B.
4. Demonstrate Remote Graphics:
   - Open a graphical application on PC-A (e.g., Paint, Word, or a browser) from PC-B.
   - Show that graphics and user interface are displayed on PC-B but processed by PC-A.
5. End the Session:
   - Close the Remote Desktop window.
   - Log off or disconnect the session properly.

Practical Activity 3.3: Using SSH to Manage Devices Securely

Materials Required

- Router, switch, or Linux server with SSH enabled
- PC/Laptop with terminal/command prompt (Windows/Linux/macOS)

- Ethernet cable or LAN connection
- IP address of the device
- Username and password for SSH login

Simple Steps

1. Connect the PC to the network device (router/switch/server) using Ethernet or LAN.
2. Assign IP addresses so both devices are in the same network (e.g., PC: 192.168.1.10, Device: 192.168.1.1).
3. Enable SSH on the device (if router/switch):

- Set hostname → hostname SecureDevice
- Set domain name → ip domain-name school.local
- Generate RSA keys → crypto key generate rsa
- Create user → username admin password Secure@123
- Configure VTY lines →
    - line vty 0 4
    - login local
    - transport input ssh

4. Save the configuration → write memory.
5. On the PC terminal, type: ‘ssh admin@192.168.1.1’
6. Enter the password → Secure@123.
7. Once connected, run basic commands such as:

- show ip interface brief (for router/switch)
- ls or pwd (if Linux server).

8. Exit the session → type exit.

Example 3.1: Using RDP to control a Windows Server

A system administrator needs to install a new program on a Windows Server 2022 instance in a data center. The app needs a graphical interface.

Steps for the demonstration:

- **Materials Required:** * A Windows Server (like 192.168.10.50) with Remote Desktop turned on and set up to let connections. It is expected that you have a static IP address and can connect to the network.
- **Client:** A Windows computer with the "Remote Desktop Connection" software or a Linux or macOS computer with an RDP client (like FreeRDP).
- **Protection:** The RDP connection should ideally go through a VPN for maximum protection. If that isn't possible, a firewall should only allow trusted source IPs to access the RDP port (TCP 3389).

1. * Setting up Windows Server (in a simulated way):
    - Action: On the Windows Server, click "Server Manager," then "Local Server," and finally "Remote Desktop." Make sure it is turned on and that the right users and groups are set up.

- Action: Check if the server's firewall rules let RDP connections come in.

2. To connect to RDP from a client:
    - Action (Windows Client): Search for mstsc.exe and open "Remote Desktop Connection."
    - Action: Type in the Windows Server's IP address or hostname (for example, 192.168.10.50).
3. What to do: Click "Connect." You will be asked for your credentials.
    - Action: Type in the Windows Server administrator's username and password.

Note: The desktop environment of the distant server shows up on your local screen. You can now use the server's GUI as if you were there in person.

Practical Activity 3.4: Using VNC to get to a Linux desktop

Materials Required

- One Linux computer (as VNC server)
- One Windows/Linux computer (as VNC client)
- VNC software: TigerVNC (server) and VNC Viewer (client)
- Local network connection (LAN/Wi-Fi)

Simple Steps

1. On Linux computer (Server):
    - Install VNC server: type ‘sudo apt install tigervnc-standalone-server’
    - Start the server: type ‘vncserver’
    - Set a password when asked (e.g., Class12@123).
    - Note the server IP and display (e.g., 192.168.1.20:1).
2. On Client computer (Windows/Linux):
    - Open VNC Viewer.
    - Enter the server address (e.g., 192.168.1.20:1).
    - Enter the password (Class12@123).
3. Remote Desktop Access:

- The Linux desktop appears on the client PC.
- Open apps like Firefox, LibreOffice, or Terminal to demonstrate.

4. Close Session:

Exit VNC Viewer when finished.

SUMMARY

- The ability to access a remote computer and all its data and programs from another device and manage it as if you were physically present is what remote access software is all about.
- The use of software, hardware, and an active network connection allows for remote access.
- Different remote access technologies, such as virtual private networks (VPNs), desktop

sharing (VNCs), and remote desktop protocols (RDPs), make it possible for users to access and completely operate the computer of another user from a remote location.

- Among several protocols is the remote desktop protocol, also known as RDP. RDP, which stands for Remote Desktop Protocol, is a Windows application that allows users to connect two computers to one another through the use of a graphical user interface.
- Data transported between devices must be encrypted using cryptography to prevent unauthorized access, which is why it is vital for safeguarding remote access. Secure shell (SSH), virtual private networks (VPNs) (IPsec and SSL/TLS), and encrypted web access all use encryption to keep private data safe while in transit over the internet.
- Given the ease of file communication between computers, there is always the risk of malware transmission or unwanted access by an unauthorized third party.
- With remote access graphics, you may view and manipulate the visual output of a distant computer. Working on a computer from afar is typically accomplished over a network connection.

ASSESSMENT

A. Multiple Choice Questions

1. What is the main benefit of employing remote access?
   a) More secure physical space
   b) More space for data storage
   c) Better teamwork and productivity
   d) Less use of network bandwidth

2. What does VPN mean when it comes to accessing things from far away?
   a) VPN
   b) Very Protected Network
   c) Private Network That Is Useful
   d) Public Network That Is Easy to See

3. What protocol is most often used to securely access Windows desktops from afar?
   a) SSH
   b) RDP
   c) HTTP
   d) FTP

4. What is a major security issue with remote access?
   a) Faster network speed
   b) Access to sensitive data without permission
   c) Better user experience
   d) Less need for IT assistance

5. Which of these is NOT a popular way to access anything from afar?
   a) VPN
   b) RDP
   c) Direct Access
   d) Local Area Network (LAN)

6. In the context of remote access security, what does MFA mean?
   a) Maximum File Access
   b) Multi-Factor Authentication
   c) Manage File Access
   d) Minimal File Access

7. What is a feature of Remote Desktop Protocol (RDP)?
   a) It's a protocol that is free and open source.
   b) Its main use is to move files.
   c) It lets people who are not on the same network use a graphical interface.
   d) It needs a separate physical line to connect.

8. What is one big benefit of using Direct Access instead of a VPN?
   a) It is safer.
   b) It needs to be started by hand.
   c) It automatically sets up a secure connection.
   d) It works better with older operating systems.

9. Which of the following security protocols makes sure that network traffic is private and safe?
   a) HTTP
   b) FTP
   c) IPsec
   d)TCP

10. What is the main job of a VPN or other remote access solution?
   a) To limit access to the network.
   b) To make a private network on top of a public one.
   c) To keep track of user accounts.
   d) To keep an eye on network traffic.

B. Fill-in-the-blanks

1. Different remote access technologies, such as_________, ____________ and _____________, make it possible for users to access and completely operate the computer of another user from a remote location.
2. With the help of the________________, domain names are constantly mapped to changing IP addresses.
3. The graphical user interface for desktop sharing is provided via ____________, which stands for virtual network computing.
4. ______________is a Windows application that allows users to connect two computers to one another through the use of a graphical user interface.
5. An intermediary computer acts as a go-between for you and another computer that is part of the _____________ when you use an internet proxy server like the one described above.
6. Graphic design-specific remote access tools include _______ and __________, which facilitate file sharing, teamwork, and, in some cases, GPU acceleration.
7. _____________- refers to technologies that allow a user to view and interact with a graphical desktop environment running on a remote computer, as if they were sitting directly in front of it.
8. ___________ allows users to remotely access equipment across untrusted networks in a safe and secure manner.

9. For gamers, ______________means they can enjoy graphically demanding games on more powerful gaming PCs from any device, regardless of how powerful their home system is.
10. Given the ease of file communication between computers, there is always the risk of ___________ or unwanted access by an unauthorized third party.

C. True/False questions

1. A VPN is a way to connect to a private network from a distance.
2. You can operate another computer from a distance using Remote Desktop Protocol (RDP).
3. SSH (Secure Shell) is a protocol used for encrypting data transfers and establishing secure remote connections between computers.
4. A Virtual Network Computing (VNC) connection is just as safe as a VPN connection.
5. A strong password is all that's needed for safe remote access; multi-factor authentication isn't necessary.
6. Remote access can only be used for accessing files and folders on a remote computer.
7. Remote access is only used by large corporations and not by individuals.
8. Direct Access is a type of VPN that automatically connects a user to a network when they connect to the internet.
9. You can handle and keep an eye on industrial equipment from a distance via remote access.
10. As long as a business has a solid firewall, it doesn't need a secure remote access solution.

D. Short Answer Questions

1. What is remote access, and what does it allow a user to do?
2. Name three common purposes of remote access in business or education.
3. What are the main methods for achieving remote access today?
4. List three remote access technologies that enable users to control another computer.
5. How does a VPN help secure remote access connections?
6. What is the role of VNC in remote access?
7. How does RDP differ from VNC in providing remote access?
8. Name two security measures to ensure safe remote access.
9. What are remote graphics, and how are they useful in graphic-intensive tasks?
10. Mention two software tools used for remote access in graphic design and their advantages.

Answer Key

A. Multiple Choice Questions

1. c, 2. a, 3. B, 4. b, 5. d, 6. b, 7. c, 8. c, 9. c, 10. b

B. Fill-in-the-blanks

1. VPN, RDP, Direct Access, 2. Dynamic DNS, 3. VNC (Virtual Network Computing),
4. Remote Desktop, 5. Internet, 6. TeamViewer, AnyDesk, 7. Remote Desktop Technology,
8. VPN (Virtual Private Network), 9. Cloud Gaming, 10. Hacking

C. True/False questions

1. True, 2. True, 3. True, 4. False, 5. False, 6. False, 7. False, 8. True, 9. True, 10. False

Unit-2. Managing Cloud Architecture with Open Stack

The proliferation of cloud computing has fundamentally reshaped the technological landscape, demanding robust, flexible, and scalable infrastructure management solutions. In this dynamic environment, OpenStack has emerged as a premier open-source cloud computing platform, empowering organizations to build and manage both public and private clouds with unparalleled control and customization. This unit serves as a comprehensive gateway into the world of OpenStack, designed to equip you with the foundational knowledge and initial skills necessary to navigate its complex yet powerful ecosystem.

As we embark on this exploration, our primary objective is to demystify OpenStack and its integral role in contemporary cloud architecture. We will begin by establishing a solid understanding of the basics of OpenStack, clarifying its core purpose, key characteristics, and its significance in providing Infrastructure-as-a-Service (IaaS). You will learn what OpenStack is, why it is a critical tool for modern cloud deployments, and the fundamental problems it solves.

Subsequently, this unit will guide you through the intricacies of the OpenStack architecture. We will dissect its modular design, examining the core services such as Nova for compute, Neutron for networking, Cinder for block storage, Glance for image management, and Keystone for identity services and understanding how these components interoperate to deliver a cohesive cloud environment. By grasping this architectural framework, students will appreciate the synergy between different services and how they collectively enable the provisioning and management of cloud resources.

Finally, recognizing that operational efficacy is paramount, this unit will introduce students to the basics of troubleshooting in OpenStack. While a comprehensive mastery of troubleshooting requires extensive experience, we will explore common issues, diagnostic methodologies, and essential tools. This foundational exposure will enable students to approach problem-solving systematically, identify potential points of failure, and understand how to gather relevant information for issue resolution within an OpenStack deployment.

Upon successful completion of this unit, students will not only possess a theoretical understanding of OpenStack's core concepts and architecture but will also have taken their first steps towards practical application and problem-solving within this powerful cloud operating system. This knowledge forms an essential building block for anyone aspiring to design, deploy, manage, or develop for cloud environments leveraging the OpenStack platform.

Chapter-4

Basics of Open Stack

Amit was studying cloud computing and heard about OpenStack. His teacher explained that OpenStack is like a toolkit for building a private cloud, where you can create and manage virtual machines, networks, and storage just like big cloud providers do. One day, Amit imagined his college needed a small cloud to run projects. He learned that Nova helps to create virtual machines, Cinder gives block storage, Swift stores files as objects, Neutron manages networking, Glance provides images, and Keystone handles login and identity. To make it easy, Horizon offers a dashboard for users.

By seeing how these services work together, Anita understood that OpenStack is basically a collection of services that allow organizations to run their own cloud platform.

4.1 Introduction to OpenStack

OpenStack is a free, open-source software platform for cloud computing, mostly deployed as Infrastructure as a Service (IaaS). It was initially started in 2010 as a joint project by Rackspace Hosting and NASA. Its primary goal is to enable organizations to build and manage private and public cloud computing platforms.

Unlike monolithic cloud solutions, OpenStack is designed as a collection of modular components (services) that interact with each other via Application Programming Interfaces (APIs). This modular design allows organizations to deploy only the services they need and scale specific components independently.

OpenStack provides a control plane to manage pools of compute, storage, and networking resources throughout a data center. Users interact with OpenStack through a web dashboard, command-line tools, or directly via its APIs to provision and manage their cloud resources.

OpenStack was founded on principles of open source, open design, and open development. This philosophy fosters a large, diverse community of developers, users, and vendors who contribute to its continuous evolution.

Key milestones include:

- **2010:** Launched by Rackspace and NASA with projects like Nova (Compute) and Swift (Object Storage).
- **2011:** First official release ("Austin"). Rapid growth in community and number of core projects.
- **2012:** Formation of the OpenStack Foundation (now the Open Infrastructure Foundation) to govern and manage the project.
- **Ongoing:** Regular six-monthly release cycles (named alphabetically, e.g., "Rocky," "Stein," "Train," "Ussuri," "Victoria," etc.), introducing new features, services, and improvements.

The philosophy emphasizes transparency, collaboration, and flexibility. It aims to avoid vendor lock-in and provide users with the freedom to choose the underlying hardware and software components.

4.1.1 Core Services of OpenStack

While OpenStack has grown to include numerous services, some of the foundational components include:

- **Nova (Compute):** Manages the lifecycle of virtual machines (instances). It interacts with hypervisors like KVM, VMware, Xen, etc.
- **Swift (Object Storage):** Provides highly scalable, redundant object storage using a flat namespace. Ideal for unstructured data like images, backups, and archives.
- **Cinder (Block Storage):** Provides persistent block-level storage volumes that can be attached to compute instances. Suitable for traditional file systems or database storage.
- **Neutron (Networking):** Manages network connectivity (IP addresses, routing, firewalls, load balancing) within the cloud. Allows users to create their own virtual networks.
- **Keystone (Identity):** Provides authentication and authorization for all OpenStack services. Manages users, projects, roles, and service catalogs. (This is the focus of subsequent chapters).
- **Glance (Image Service):** Stores and manages virtual machine disk images.
- **Horizon (Dashboard):** A web-based graphical user interface for interacting with OpenStack services.
- **Heat (Orchestration):** Templates and orchestrates the deployment of complex cloud applications and infrastructure.

These core services form the backbone of a typical OpenStack deployment. Numerous other projects exist for services like telemetry, database as a service, container orchestration integration, etc.

Assignment 4.1

1. What is OpenStack, and what are its main goals as a cloud computing platform?
2. Explain the modular design of OpenStack. How does it benefit organizations in deploying and managing cloud services?
3. List and briefly describe any four core services of OpenStack.

4.1.2 OpenStack Architecture Overview

OpenStack's architecture is distributed and API-driven. Each service typically runs as one or more processes, exposing a public API endpoint. These services communicate with each other and with the underlying infrastructure (hypervisors, storage devices, network switches).

A central component is the **Message Queue** (like RabbitMQ or Qpid), which facilitates asynchronous communication between services. When a user requests to launch an instance via the Nova API, the Nova API service might send a message to a Nova scheduler service via the message queue. The scheduler then determines which compute host should run the instance and sends another message via the queue to the Nova compute service on that host.

All services rely on **Keystone** for authentication and often for authorization. Before a service (like Nova) processes an API request, it will validate the user's identity and permissions using Keystone.

A shared **Database** (typically SQL like MySQL or PostgreSQL) is used by most services to store their state and configuration.

4.1.3 Importance of Identity and Security in Cloud

In any multi-tenant or private cloud environment, security is paramount. Users need to be confident that their data and resources are isolated and protected from other users. A fundamental layer of this security is **Identity and Access Management (IAM)**.

IAM in OpenStack addresses critical questions:

1. **Authentication:** *Who are you?* (Verifying the identity of a user or service)
2. **Authorization:** *What are you allowed to do?* (Granting permissions to perform specific actions on specific resources)

Without a robust identity system, a cloud platform cannot securely separate tenants, control resource access, or audit user actions. Keystone is the OpenStack project responsible for providing these essential IAM capabilities.

4.2 OpenStack Authentication System

OpenStack manages valuable computing resources – virtual servers, sensitive data in storage, and control over network traffic. In a multi-user or multi-tenant environment, it is critical to know *who* is making requests to these resources and to ensure that only authorized entities can access them.

The OpenStack authentication system serves several vital purposes:

- **User Identification:** Verifying the identity of individuals, applications, or services interacting with the cloud.
- **Resource Isolation:** Ensuring that users and projects can only access resources that belong to them or to which they have been granted explicit access.
- **Security:** Protecting the cloud infrastructure and the data stored within it from unauthorized access and malicious activities.
- **Accountability and Auditing:** Logging actions performed by authenticated users for auditing, troubleshooting, and compliance purposes.

- **Service-to-Service Security:** Enabling secure communication between different OpenStack services themselves, which often need to make requests to one another.

Every interaction with an OpenStack service API begins with authentication (or validating a prior authentication).

Keystone is the OpenStack project that provides identity, authentication, and authorization services for the entire cloud platform. It acts as a central directory and authentication hub for all other OpenStack services.

Think of Keystone as the "bouncer" or "security desk" for your OpenStack cloud. Before any user or service can access resources managed by Nova, Cinder, Neutron, etc., they must first prove their identity to Keystone and obtain a token that represents their authenticated session and authorized scope.

Keystone's primary responsibilities include:

- **Identity Management:** Storing and managing information about users, groups, projects (tenants), roles, and domains.
- **Authentication:** Verifying the credentials of users and services.
- **Token Management:** Issuing, managing, and validating tokens that represent authenticated sessions.
- **Service Catalog:** Providing a catalog of the API endpoints for all other OpenStack services running in the cloud, which is typically included in the authentication response.
- **Authorization (via Roles):** Providing the framework for assigning permissions based on roles, although the enforcement of these permissions happens within the individual OpenStack services.

4.2.1 Role of Keystone in the OpenStack Ecosystem

Keystone is a foundational service upon which all other OpenStack services depend. When a user interacts with OpenStack, via the CLI, dashboard, or direct API calls, the initial request flow involves Keystone:

1. The user provides credentials (e.g., username/password) to the Keystone API.
2. Keystone authenticates the user against its backend (database, LDAP, etc.).
3. If authentication is successful, Keystone issues a unique **token**. This token acts as a session key.
4. Keystone also provides the user with a **Service Catalog**, which lists the endpoints (API addresses) for all OpenStack services the user is authorized to access within their authenticated scope.
5. For subsequent API requests to other services (e.g., Nova to launch a VM), the user includes this token in the request header.
6. The receiving service (e.g., Nova) receives the request and the token. Before processing the request, it sends the token to Keystone for **validation**.
7. Keystone validates the token (checks if it's valid, not expired, etc.) and returns information about the token's owner (the user, their roles, their project context) to the requesting service (Nova).

8. Based on the user's roles and the requested action, the service (Nova) performs an **authorization check** to determine if the user is permitted to perform that action in the specified project context.
9. If authorized, the service processes the request. If not, it denies the request.

This flow highlights Keystone's central role: it's the gatekeeper for accessing any OpenStack service API.

4.2.2 Authentication vs. Authorization

It's crucial to distinguish between authentication and authorization, as Keystone facilitates both aspects:

- **Authentication:** The process of verifying the identity of a user or service. It answers the question "Who are you?" Authentication typically involves providing credentials (like username/password, a certificate, or a token) and having them verified.
- **Authorization:** The process of determining what an authenticated user or service is permitted to do. It answers the question "What are you allowed to do?" Authorization in OpenStack is typically handled by assigning roles to users (or groups) within specific projects (or domains). The individual OpenStack services then use the role information provided by Keystone (during token validation) to enforce access control policies.

Keystone handles the authentication and provides the necessary identity and role information to other services. The *enforcement* of authorization rules based on these roles primarily resides within each individual OpenStack service's policy engine.

Assignment 4.2

1. Explain the role of the Message Queue in OpenStack's architecture.
2. Describe the importance of Identity and Access Management (IAM) in OpenStack.
3. What is Keystone in OpenStack, and how does it manage authentication and authorization?
4. Differentiate between Authentication and Authorization in the context of OpenStack.

4.3 OpenStack Identity Management (Keystone Concepts)

Keystone manages several fundamental identity primitives that define who users are, what groups they belong to, which resource containers they operate within, what permissions they possess, and how these identities are isolated.

The primary identity components are:

- Users
- Groups
- Projects (Tenants)
- Roles
- Domains

Users

A **User** represents an individual, application, or service that interacts with OpenStack. Each user has a unique identifier and associated credentials used for authentication. Users are the entities that perform actions within the cloud.

- **Individuals:** Human administrators, developers, or end-users.

- **Applications:** Software processes that need to interact with OpenStack APIs (e.g., a monitoring script, a CI/CD tool). These often use specific user accounts or API keys.
- **Other Services:** OpenStack services themselves (like Nova, Cinder) may need to authenticate with Keystone to perform service-to-service operations or access the service catalog.

Users exist within a **Domain**, which provides a namespace and isolation boundary for identity information.

User Management

User management involves operations like:

- Creating new users
- Deleting users
- Modifying user attributes (password, email, enabled/disabled status)
- Assigning users to groups
- Assigning roles to users within specific projects or domains

These operations are typically performed by administrators via the Keystone API, the Horizon dashboard, or command-line tools.

Authentication Methods

Keystone supports various methods for users to authenticate:

- **Password Authentication:** The most common method. Users provide a username and password. Keystone verifies these credentials against its configured backend.
- **Token Authentication:** Users provide an existing, valid Keystone token as their credential. This is how users make subsequent API calls *after* initially authenticating with password (or other method).
- **Application Credential Authentication:** A method for applications or scripts to authenticate without using a user's password. An application credential is created by a user, linked to their identity, and granted specific access permissions. It's a more secure alternative to password authentication for automated tasks.
- **Federated Authentication:** Allows users from an external Identity Provider (IdP) (like a corporate Active Directory or Shibboleth) to log in to OpenStack using their existing enterprise credentials. Keystone acts as a Service Provider (SP), interacting with the external IdP via protocols like SAML or OpenID Connect.
- **Certificates:** Authentication can potentially be done using client-side SSL certificates, although this is less common for end-users compared to other methods.

Groups

A **Group** is a collection of users. Groups simplify the process of assigning roles and managing permissions. Instead of assigning a role to multiple individual users, you can assign the role to a group, and all members of that group inherit that role.

- **Example:** Create a group called "developers" and assign the "developer" role to this group within the "projectX" project. Any user added to the "developers" group will automatically have the "developer" role within "projectX".

Group Management: Group management involves:

- Creating and deleting groups
- Adding or removing users from groups
- Assigning roles to groups within projects or domains

Projects (Tenants)

A **Project** (often historically referred to as a Tenant) is a container for OpenStack resources and users. It provides a logical isolation boundary. Resources (like virtual machines, volumes, networks, etc.) are allocated and managed within a specific project.

- **Purpose:**
  - **Resource Grouping:** Resources are grouped together under a project, making them manageable as a unit.
  - **Resource Isolation:** Resources within one project are typically isolated from resources in another project by default. Users in one project cannot access or even see resources in another project unless explicitly granted permissions.
  - **Billing/Quota:** Projects are often used as the unit for tracking resource usage for billing or enforcing resource quotas.

Resource Isolation and Accounting

Isolation between projects is primarily enforced by the individual OpenStack services (Nova, Cinder, Neutron, etc.). When a service receives an API request with a project-scoped token, it knows the request is made *on behalf of that project*. The service then uses its internal logic and the underlying infrastructure (like network namespaces in Neutron) to ensure that the request only affects resources within that specific project.

Quotas (limits on the number of VMs, volumes, IPs, etc., that a project can consume) are also managed on a per-project basis by the individual services, often configured via their APIs or configuration files.

Project Management

Project management includes:

- Creating and deleting projects
- Enabling or disabling projects
- Assigning users and groups to projects
- Assigning roles to users and groups *within* projects

Roles

A **Role** represents a set of permissions that can be assigned to a user or a group within a specific scope (typically a project or a domain). Roles do *not* define the permissions themselves; they are simply labels (e.g., "admin," "member," "reader"). The *actual permissions* granted by a role are defined in **policy files** specific to each OpenStack service (e.g., `nova/policy.json`, `cinder/policy.yaml`, `neutron/policy.yaml`).

When an OpenStack service receives an API request with a token, Keystone provides the service with the roles the authenticated user has within the project (or domain) scope of the token. The service then consults its internal policy engine, which maps API actions (e.g., `os_compute_api:servers:create`, `os_volume_api:volumes:delete`) to required roles. If the user possesses one of the required roles for that action, the action is authorized; otherwise, it is denied.

Role Assignment (User/Group to Project)

Roles are assigned to a **principal** (a user or a group) within a **scope** (a project or a domain).

Common assignment structures:

- User `alice` has role `member` on project `prod_web`.
- Group `admins` has role `admin` on project `prod_db`.
- User `bob` has role `reader` on domain `default`.

This granular assignment allows for fine-grained control over permissions.

Keystone deployments often come with standard roles. The most common is the `_member_` role (often displayed as "member" or "user" in the dashboard). Users assigned the `_member_` role within a project are typically granted basic permissions by most services to create and manage their own resources within that project. The `admin` role typically grants permissions to manage the project itself and all resources within it.

Administrators can create custom roles in Keystone and then define policies in the service policy files that grant specific permissions to these custom roles. This allows for highly customized access control.

Domains

A **Domain** provides a higher level of isolation for identity resources (users, groups, projects). Introduced in Keystone v3, domains allow for multi-tenancy of the identity system itself.

- **Use Cases:**
  - Representing different organizations or departments within a single OpenStack cloud.
  - Integrating with multiple distinct external identity backends (like different LDAP directories).
  - Delegating the administration of users, groups, and projects within a domain to a domain-specific administrator, without granting them full control over the entire Keystone deployment.

Users, groups, and projects all belong to a specific domain. Usernames, group names, and project names are unique *within* a domain but can be duplicated *across* different domains (e.g., user 'admin' can exist in both 'default' and 'finance' domains).

Assigning a role to a user or group on a domain grants them permissions related to the identity resources within that domain (e.g., permission to create users or projects in that domain), as opposed to granting permissions on cloud resources within a project.

Domain management includes:

- Creating and deleting domains
- Assigning administrators to domains
- Configuring identity backends per domain

Identity Backends

Keystone needs to store and retrieve information about users, groups, projects, roles, and domains. It supports various backend drivers for this purpose:

- **SQL Database:** The default and most common backend. Keystone stores all identity data in a relational database (MySQL, PostgreSQL, etc.). Simple to set up and manage for most deployments.
- **LDAP/Active Directory:** Allows Keystone to authenticate users and retrieve user/group information directly from an external LDAP or Active Directory server. This is crucial for integrating OpenStack into existing enterprise identity infrastructure. While user and group *attributes* can come from LDAP, project and role assignments are typically still managed within Keystone's SQL database (though more advanced configurations can externalize some assignments).
- **Federated Identity:** As mentioned earlier, this backend handles interaction with external Identity Providers (IdPs) using protocols like SAML or OpenID Connect. When a user authenticates via federation, Keystone maps the external identity attributes (like group memberships) to internal Keystone users and roles, often assigning them to a specific project based on configuration (Identity Mapping).

Keystone can be configured to use different backends for different domains, providing flexibility in how identities are managed across the cloud.

4.4 Tokens and Token Validation API

Once a user successfully authenticates with Keystone, Keystone issues a **Token**. A token is a digital key or credential that represents the user's authenticated session and the context (scope) within which they are authorized to operate.

The primary purpose of tokens is to avoid requiring users to re-authenticate with their primary credentials (like username/password) for every single API request. Instead of sending sensitive credentials repeatedly, the user sends the token.

- **Stateless API Interaction (for the client):** After obtaining a token, subsequent API calls are made by including the token in the request header (`X-Auth-Token`). The client doesn't need to manage connection state based on username/password.
- **Session Management:** Tokens define a session's validity period (they expire) and scope.
- **Information Carrier:** Tokens often carry information about the authenticated user, their roles, and the service catalog, reducing the need for the client to make separate calls for this information.

4.4.1 Information Contained in a Token

When Keystone issues a token, it typically contains or is linked to information about:

- **User:** The ID and name of the authenticated user.
- **Expiration:** The timestamp when the token becomes invalid.
- **Issued At:** The timestamp when the token was created.
- **Methods:** The authentication method used to obtain the token (e.g., password, application_credential, token).
- **Project (if scoped):** The ID and name of the project the token is scoped to.
- **Domain (if scoped):** The ID and name of the domain the token is scoped to.
- **Roles (if scoped):** A list of roles assigned to the user within the token's scope (project or domain).

- **Service Catalog:** A list of all OpenStack services the user can access within the token's scope, including their API endpoints. (This allows clients to discover service locations without hardcoding them).

The exact information contained *within* the token itself versus information retrieved by the service during validation depends on the token format used.

4.4.2 Token Formats

Keystone has supported several token formats over its history. The choice of format impacts performance, scalability, and statefulness.

UUID Tokens

- **Description:** The simplest format. A UUID token is just a unique, random identifier (a UUID).
- **Mechanism:** When Keystone issues a UUID token, it stores all the token's associated information (user, roles, scope, expiration, service catalog) in its backend database.
- **Validation:** When a service receives a UUID token, it sends the UUID to Keystone. Keystone looks up the UUID in its database, retrieves the stored information, and sends it back to the service.
- **Pros:** Simple to implement.
- **Cons: Stateful.** Requires Keystone to maintain state for every active token in its database. This can become a performance bottleneck and scalability challenge in large deployments, as every API request validation requires a database lookup within Keystone. Keystone nodes must share the same database.

PKI and PKIv3 Tokens

- **Description:** Cryptographically signed tokens using Public Key Infrastructure.
- **Mechanism:** Keystone signs the token data (user, roles, scope, expiration, etc.) using a private key. The token itself contains the signed data.
- **Validation:** Services validate PKI tokens by using Keystone's public key to verify the signature. If the signature is valid and the token hasn't expired, the service trusts the information contained within the token *without needing to contact Keystone*.
- **Pros: Stateless (for validation).** Services don't need to contact Keystone for every validation, significantly reducing the load on Keystone's validation API and database. Improves performance and scalability. Keystone nodes only need access to the public key, not necessarily a shared database for tokens (though they still need a database for identity data).
- **Cons:** Token size can be large because they contain all the information and the signature. Key management requires distributing the public key to all services. Validation involves cryptographic operations, which have a small CPU cost.

Fernet Tokens (Preferred Stateless)

- **Description:** Encrypted and signed tokens using the Fernet specification (a symmetric encryption scheme).
- **Mechanism:** Keystone encrypts the token data using a symmetric key and appends a message authentication code (MAC) for integrity verification. The token is a base64 encoded string containing the encrypted data and MAC.

- **Validation:** Services validate Fernet tokens by decrypting the token using the same symmetric key and verifying the MAC. If successful and not expired, the token is valid.
- **Pros: Stateless.** Like PKI, validation does not require contacting Keystone. Much smaller token size compared to PKI. Faster validation than PKI (symmetric encryption is faster than asymmetric signing/verification).
- **Cons:** Requires secure distribution and rotation of symmetric keys to all services. If a key is compromised, tokens issued with that key could be forged or decrypted (though revocation mechanisms help mitigate this).

Recommendation: Fernet tokens are the recommended and default format in modern OpenStack deployments due to their stateless nature, performance, and smaller size compared to PKI. They offer a good balance of security and efficiency.

4.4.3 Token Generation Process

The process of obtaining a token involves an authentication request to the Keystone API.

The Authentication Request (POST /v3/auth/tokens)

This is the primary API endpoint used by users (or their clients like CLI/dashboard) to authenticate and receive a token. The request is a POST operation to the/v3/auth/tokens endpoint of the Keystone service.

The request body contains the user's credentials and specifies the desired scope of the token.

Authentication Methods and Request Body

The POST /v3/auth/tokens request body uses a JSON structure. The auth object within the body contains the authentication method and credentials.

4.4.4 The Authentication Response (The Token and Service Catalog)

If authentication is successful, Keystone returns a 201 Created HTTP status code.

Crucially, the response includes:

1. **The Token ID:** In the X-Subject-Token HTTP header. This is the value the client must include in the X-Auth-Token header for subsequent requests to other services.
2. **The Response Body:** Contains detailed information about the authenticated session, including:
   - User details (ID, name, domain)
   - Expiration time
   - Issued-at time
   - Authentication methods used
   - **Scope details:** Information about the project or domain the token is scoped to (if requested).
   - **Roles:** The list of roles the user has within the token's scope.
   - **Service Catalog:** A list of endpoints for all OpenStack services accessible within the token's scope. This is essential for clients to know where to find the Nova, Cinder, Neutron, etc., APIs.

4.4.5 Token Validation API

When an OpenStack service (e.g., Nova) receives an API request with an X-Auth-Token header, it cannot inherently trust the token or know who it represents. It needs to verify the token's validity and get information about the principal (user/group) and scope associated with it. This is done by calling the Keystone Token Validation API.

The validation request is a GET operation to the same /v3/auth/tokens endpoint used for generation, but this time the request is made by one OpenStack service to Keystone, and the token to be validated is included in the X-Auth-Token header of this GET request.

The service validating the token includes:

- The token to be validated in the X-Auth-Token header.
- Optionally, its *own* service token in the X-Auth-Token header (if required by Keystone policy for validation requests).
- Optionally, including the ?nocatalog query parameter is a common optimization for service-to-service validation, as the service doesn't need the catalog; it just needs the user/role/scope info.

The workflow within a service receiving a user request with a token is:

1. Service (Nova) receives GET /servers request with X-Auth-Token: UserToken.
2. Nova extracts UserToken.
3. Nova sends a GET /v3/auth/tokens request to Keystone, including UserToken in its own X-Auth-Token header.
4. Keystone receives the validation request for UserToken.
5. Keystone verifies UserToken (checks format, signature/decryption, expiration, revocation status).
    - If using UUID tokens, this involves a database lookup.
    - If using PKI/Fernet tokens, this involves cryptographic verification without a database lookup (unless checking revocation).
6. If UserToken is valid, Keystone returns a 200 OK HTTP status code with a response body containing the token details (user, roles, scope, etc., excluding the service catalog if ?nocatalog was used).
7. If UserToken is invalid (expired, revoked, malformed), Keystone returns a 401 Unauthorized or 404 Not Found HTTP status code.
8. Nova receives the validation response.
9. If valid (200 OK), Nova extracts the user ID, project ID, and roles from the response body.
10. Nova performs an authorization check based on the requested API action (GET /servers), the user's roles, the project scope, and Nova's internal policy rules (nova/policy.yaml).
11. If authorized, Nova processes the GET /servers request.
12. If unauthorized (e.g., user doesn't have the reader or member role in the project), Nova returns a 403 Forbidden error to the user.
13. If the token was invalid (401/404), Nova returns a 401 Unauthorized error, prompting the user to re-authenticate.

Validation Response

The response body from a successful GET /v3/auth/tokens validation request is similar to the initial authentication response body (POST /v3/auth/tokens), but typically omits the catalog section if the ?nocatalog parameter was used. It provides the validating service with all the necessary information about the token's owner and context.

4.4.6 Scope of Tokens

Tokens can be issued with different scopes, affecting the information they contain (especially the service catalog and roles) and the context in which they can be used.

- **Unscoped Tokens:**
  - Obtained by authenticating without specifying a scope in the POST /v3/auth/tokens request.
  - Contain user details, expiration, and a service catalog that lists services accessible globally or services that don't require a project context.
  - Do not contain project/domain ID or project/domain-specific roles.
  - Cannot be used to interact with most OpenStack APIs that require a project context (like creating a VM in Nova or a volume in Cinder).
  - Primarily used to list available projects (GET /v3/auth/projects) or domains and then request a scoped token for a specific project/domain.
- **Project-Scoped Tokens:**
  - Obtained by specifying a scope.project in the POST /v3/auth/tokens request.
  - Contain user details, expiration, project ID/name, and roles the user has within that specific project.
  - The service catalog is filtered to show only endpoints for services accessible within that project's scope.
  - Required for interacting with most core OpenStack services (Nova, Cinder, Neutron, Glance when uploading images to a project, etc.) that manage project-specific resources.
- **Domain-Scoped Tokens:**
  - Obtained by specifying a scope.domain in the POST /v3/auth/tokens request.
  - Contain user details, expiration, domain ID/name, and roles the user has within that specific domain.
  - Used for administrative actions related to managing identity resources within a specific domain (e.g., creating users, groups, or projects inside that domain), provided the user has the necessary domain-level roles.
  - Cannot be used for project-specific resource management (creating VMs, etc.).

Users typically obtain an unscoped token first, list their available projects from the catalog in the unscoped token or by calling the list projects API, and then request a project-scoped token for the project they wish to work in.

4.4.7 Token Lifecycle

Tokens have a finite life cycle:

Expiration

Tokens are issued with an expiration time (expires_at). Once expired, a token is no longer valid and cannot be used for authentication or validation. The default token expiration time is configurable in Keystone (typically 1 hour for X-Auth-Token and longer for X-Subject-Token if it's an unscoped token used only to get project list).

Renewal (Implicit via Re-authentication)

There is no explicit "token renewal" API in OpenStack. To get a new, valid token, the user must perform the authentication process again (POST /v3/auth/tokens). They can use their primary credentials (username/password) or use their *existing, non-expired token* with a methods: ["token"] request body to obtain a new token.

Revocation

Administrators (or sometimes users for their own tokens) can explicitly **revoke** tokens before they expire. Revocation invalidates the token immediately. This is important if credentials might have been compromised or if a user's access needs to be terminated instantly.

Keystone maintains a revocation list. For stateless tokens (PKI/Fernet), validation *might* still involve checking this revocation list, depending on configuration, adding a small stateful element back into validation.

API Endpoints Related to Tokens

Beyond the core POST /v3/auth/tokens (generate) and GET /v3/auth/tokens (validate), other related token APIs exist:

- GET /v3/auth/tokens: As seen, used for validation (by services) or checking details of a valid token (by users). Requires the token in the X-Auth-Token header.
- DELETE /v3/auth/tokens: Revokes the token provided in the X-Auth-Token header. Allows users to log out or invalidate their current token.
- GET /v3/auth/tokens?subject_token_id=...: Allows administrators (with appropriate roles) to inspect the details of a specific token ID, potentially without needing to possess the token itself.
- DELETE /v3/auth/tokens?subject_token_id=...: Allows administrators to revoke a specific token ID.
- GET /v3/auth/tokens/: (Often requires admin roles) List issued tokens. This is less common and can be performance-intensive.

These endpoints provide flexibility in managing the token lifecycle.

OpenStack's identity and authentication system, primarily embodied by the Keystone service, is a cornerstone of its architecture and security model. It provides the essential mechanisms for verifying who is accessing the cloud and defining the context in which they operate.

Assignment 4.3

- Explain the role of a "User" in OpenStack Keystone. Who can be considered a user and what are the ways to manage users?
- What are "Groups" in Keystone and how do they simplify role management? Give an example scenario.
- Describe the concept of "Projects" in OpenStack Keystone. How do projects help in resource isolation and billing?
- What is a Keystone token? Describe the different types of tokens and how they help in authentication.

Practical Activity 4.1. Demonstration of Openstack authentication

Materials Needed

- A computer with **Ubuntu 20.04 LTS** or above
- **Internet connection**
- OpenStack (DevStack or Packstack) installed

- Pre-configured OpenStack environment with:
  - Keystone (Identity Service)
  - Horizon Dashboard (Web Interface) or CLI tools
- OpenStack admin or user credentials

Step 1: Open Terminal and Source the OpenStack RC File

The RC file contains authentication details like username, password, project, and authentication URL.

```
bash
source admin-openrc.sh
```

If using a normal user: 'source demo-openrc.sh'
You can open and inspect the file: 'cat admin-openrc.sh'

Sample content:

```
export OS_USERNAME=admin
export OS_PASSWORD=secret123
export OS_PROJECT_NAME=admin
export OS_AUTH_URL=http://controller:5000/v3
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_IDENTITY_API_VERSION=3
```

Step 2: Request an Authentication Token from Keystone
Run this command:
'openstack token issue'
Expected Output:

```
+------------+----------------------------------+
| Field      | Value                            |
+------------+----------------------------------+
| expires    | 2025-05-19T12:34:56.000000Z      |
| id         | gAAAAABkD...                     |
| project_id | 76b1b9078a...                    |
| user_id    | 134a1e0bde...                    |
+------------+----------------------------------+
```

This shows authentication was successful and a **token** was generated.
Step 3: Verify User Identity

```
openstack token issue
or
openstack user show admin
```

This will display details about the logged-in user:

```
+-------------+------------------+
| Field       | Value            |
+-------------+------------------+
| name        | admin            |
| id          | abc123...        |
| domain_id   | default          |
| enabled     | True             |
+-------------+------------------+
```

Step 4: Demonstrate Authentication via Web Interface (Horizon)

1. Open browser and go to:
   'http://<controller-ip>/dashboard'
2. Enter credentials:
   - Username: admin
   - Password: secret123
3. Click "Sign In"
4. If login is successful, the **OpenStack Dashboard** appears.
5. Navigate to **Project** > **Compute** > **Instances** to see running instances.

Step 5: Verify Access with Token

You can use the token to manually access other services:
Example: Access list of images (Glance service):
'openstack image list'
Expected Output:

```
+--------------------------------------+----------+--------+
| ID                                   | Name     | Status |
+--------------------------------------+----------+--------+
| f7c124aa-7f1b-4b23-...               | Ubuntu20 | active |
+--------------------------------------+----------+--------+
```

This means the token is valid and the user is authorized.

Summary

- The fundamental concepts of OpenStack and the necessity of a robust identity system in a cloud environment.
- Keystone's central role as the identity service, managing users, groups, projects, roles, and domains.
- The different identity concepts – Users (who), Groups (collections of users), Projects (resource containers), Roles (permission labels), and Domains (identity isolation boundaries) – and how they interrelate.
- The critical role of Tokens as session keys that represent authenticated users and their scope.
- The different token formats (UUID, PKI, Fernet) and the strong advantages of modern stateless formats like Fernet.
- The detailed process of obtaining a token via the `POST /v3/auth/tokens` API, understanding the request body and the valuable information (like the service catalog) returned in the response.
- The mechanism by which other OpenStack services validate tokens using the `GET /v3/auth/tokens` API, including the workflow and response.
- The different scopes of tokens (unscoped, project-scoped, domain-scoped) and their specific use cases.
- The lifecycle of tokens, including expiration and revocation.
- Advanced topics like Federated Identity for enterprise integration and Service-to-Service authentication for securing internal communication.
- Key security best practices for deploying and managing the identity system.

ASSESSMENT

A. Multiple Choice Questions

1. What is the primary OpenStack service responsible for identity, authentication, and authorization for the entire cloud platform?
   a) Nova
   b) Swift
   c) Keystone
   d) Neutron

2. According to the document, which Keystone concept answers the question "What are you allowed to do?"
   a) Authentication
   b) Token Management
   c) Authorization
   d) User Identification

3. In Keystone, what is a "Project" (historically referred to as a "Tenant") primarily used for?
   a) Storing only user credentials and their passwords
   b) Defining network topologies for virtual machines
   c) A container for OpenStack resources, providing isolation and a unit for quotas
   d) Managing the lifecycle of hypervisors

4. Where are the actual permissions granted by a Role (e.g., "admin," "member") defined in an OpenStack deployment?
   a) In policy files specific to each OpenStack service (e.g., nova/policy.json)
   b) Directly within the Keystone Role object itself
   c) In the user's metadata stored in LDAP
   d) Within the Domain configuration that the Role belongs to

5. Which token format is described as "recommended and default in modern OpenStack deployments" due to its stateless nature, performance, and smaller size compared to PKI?
   a) UUID Tokens
   b) PKIv3 Tokens
   c) Fernet Tokens
   d) SAML Tokens

6. What HTTP method and API endpoint are typically used to request an authentication token from Keystone?
   a) GET /v3/auth/tokens
   b) POST /v3/auth/tokens
   c) PUT /v3/users/{user_id}/token
   d) GET /v3/service_catalog

7. When an OpenStack service (e.g., Nova) receives an API request with an X-Auth-Token, what is its immediate next step involving Keystone?
   a) It processes the request immediately, trusting the token.
   b) It sends the token to Keystone for validation.
   c) It checks the token against a local service cache without contacting Keystone.
   d) It issues a new token to the user.

8. What is a primary use case for "Domains" introduced in Keystone v3, as stated in the document?
   a) To group virtual machines by their operating system
   b) To provide a higher level of isolation for identity resources (users, groups, projects) and allow delegation
   c) To define specific network segments for projects
   d) To manage different types of storage backends for Cinder

9. Which of the following is typically NOT contained directly within a project-scoped token or retrieved during its validation, according to the document?
   a) The ID of the authenticated user
   b) The expiration time of the token
   c) The user's raw password
   d) A list of roles assigned to the user within the token's project scope

10. OpenStack was initially launched in 2010 as a joint project by Rackspace Hosting and NASA, with which two core projects?
    a) Keystone (Identity) and Horizon (Dashboard)
    b) Nova (Compute) and Swift (Object Storage)
    c) Cinder (Block Storage) and Neutron (Networking)
    d) Heat (Orchestration) and Glance (Image Service)

B. Fill-in-the-blanks

1. OpenStack was initially started in 2010 as a joint project by Rackspace Hosting and ____.
2. The OpenStack service responsible for managing the lifecycle of virtual machines is called ____.
3. Keystone provides authentication and authorization for all OpenStack services and manages users, projects, roles, and ____ catalogs.
4. A central component in OpenStack's architecture that facilitates asynchronous communication between services is the ____ Queue.
5. ____ is the OpenStack project responsible for providing Identity and Access Management (IAM) capabilities.
6. In Keystone, a ____ is a collection of users, simplifying role assignment.
7. The actual permissions granted by a Role are defined in ____ files specific to each OpenStack service.
8. The recommended and default stateless token format in modern OpenStack deployments is ____ tokens.
9. To obtain an authentication token, a user typically makes a POST request to the Keystone API endpoint /v3/auth/____.
10. Sourcing an OpenStack RC file sets environment variables like OS_USERNAME and OS_AUTH_URL which are used by ____ tools.

C. True/False questions

1. OpenStack is a proprietary, closed-source software platform.
2. Nova is the OpenStack service that provides persistent block-level storage.
3. The OpenStack Foundation was formed in 2012 to govern and manage the project.
4. Authorization in OpenStack answers the question "Who are you?".
5. Keystone is responsible for enforcing all authorization rules directly within itself for all other OpenStack services.
6. A "Project" in OpenStack is also historically referred to as a "Tenant".

7. UUID tokens are stateless and allow services to validate them without contacting Keystone.
8. An unscoped token can be used to launch a virtual machine in Nova.
9. To renew an OpenStack token, a user must use a specific "token renewal" API endpoint.
10. The openstack token issue command revokes an existing token.

D. Short Answer type questions.

1. What are the two primary goals of OpenStack as mentioned in the introduction?
2. List three core services of OpenStack (other than Keystone) and briefly describe their function.
3. What is the difference between authentication and authorization in OpenStack?
4. What are the three main types of identity primitives managed by Keystone mentioned under "OpenStack Identity Management (Keystone Concepts)" besides Roles and Domains?
5. Name two authentication methods supported by Keystone besides password authentication.
6. What is the primary purpose of a "Service Catalog" provided by Keystone?
7. Why are Fernet tokens generally preferred over UUID tokens in OpenStack?
8. What information is typically included in the X-Subject-Token HTTP header of an authentication response from Keystone?
9. What is the difference in usability between a project-scoped token and a domain-scoped token?
10. What is the purpose of sourcing an OpenStack RC file (e.g., admin-openrc.sh) before using OpenStack CLI commands?

Answer Key
A. Multiple Choice Questions
1.c, 2.c, 3.c, 4.a, 5.c, 6.b, 7.b, 8.b, 9.c, 10.b

B. Fill-in-the-blanks
1. NASA, 2. Nova, 3. Service, 4. Message, 5. Keystone, 6. Group, 7. Policy, 8. Fernet, 9. Tokens, 10. CLI (Command-Line Interface)

C. True/False questions
1.False, 2. False, 3. True, 4. False, 5. False, 6. True, 7. False, 8. False, 9. False, 10. False

Chapter-5

OpenStack Architecture

In the vast digital library of a school, Mr. Sharma, the IT head, faced a mountain of work. He had to set up new computers and storage for every student project, a task that required plugging in countless wires and installing software manually. Then, he discovered the magic of OpenStack.

This was not a physical tool, but a dashboard that gave him superpowers. From a single screen, he could now create new virtual computers for the robotics club, allocate storage for the art class's giant files, and connect them all instantly. OpenStack allowed him to manage the entire library's technology like a master puppeteer, making resources available to anyone who needed them, with just a few clicks. It was the future of computing, right in his hands.

OpenStack is a modular cloud computing platform that provides Infrastructure as a Service (IaaS) through a collection of interrelated components. Its architecture is designed to manage large pools of compute, storage, and networking resources, all controlled through a web-based dashboard, command-line tools, or RESTful APIs. At the heart of OpenStack are its core services: Nova (Compute), Neutron (Networking), Cinder (Block Storage), Glance (Image Service), Keystone (Identity Service), and Horizon (Dashboard). These services communicate with each other using message queues and APIs, ensuring flexibility, scalability, and integration across various environments.

OpenStack's architecture is highly distributed and scalable, allowing for deployment on commodity hardware across data centers. It follows a microservices design, where each component performs a specific function but works in harmony with others. For example, Nova handles virtual machine provisioning, while Neutron manages IP addresses and network routing. Keystone ensures secure authentication and authorization, and Glance provides access to virtual machine images. This modular approach allows organizations to choose only the components they need, simplifying deployment and maintenance while enabling seamless integration with third-party tools and services.

5.1 NOVA (OpenStack Compute Service)

NOVA is the core component of the OpenStack architecture that provides the compute functionality in an Infrastructure as a Service (IaaS) environment. It acts as a controller for managing and automating pools of computer resources and is responsible for provisioning and managing virtual machines (VMs) on demand. NOVA interacts with other OpenStack services such as Keystone for authentication, Glance for image management, Neutron for networking, and Cinder for block storage, to deliver fully functional compute instances.

NOVA is written in Python and is designed to be horizontally scalable, fault-tolerant, and modular. It supports multiple hypervisors, including KVM, QEMU, Xen, VMware, and Hyper-V, offering great flexibility in terms of deployment. Additionally, it supports bare-metal provisioning through integration with the Ironic project and container technologies via plugins.

5.1.1 Key Features of NOVA:

- **Instance Management:** Handles lifecycle operations of virtual machine instances such as create, start, stop, pause, suspend, migrate, and terminate.
- **Hypervisor Abstraction:** Abstracts the underlying hypervisor technologies using a pluggable driver architecture.
- **Multi-Tenant Support:** Supports multi-tenancy and isolates instances between tenants using projects and role-based access control.
- **Scalability:** Supports distributed architecture to manage thousands of virtual instances across clusters.
- **Scheduling:** Includes a scheduler to intelligently place VM instances on appropriate compute hosts based on resource availability and policies.
- **Fault Tolerance:** Integrates with tools for monitoring, high availability, and automatic failover to ensure uptime and reliability.

5.1.2 NOVA Architecture:

NOVA follows a distributed architecture composed of several key components, each responsible for a specific function. The main components include:

1. **nova-api:** This component handles RESTful API requests and routes them to appropriate NOVA services. It provides a user interface for developers and users to interact with the compute infrastructure.
2. **nova-scheduler:** It decides where a new virtual machine instance should be hosted. The scheduler evaluates multiple compute nodes and selects the best one based on filters and weightings (like available RAM, CPU, or affinity rules).
3. **nova-compute:** This is the core component that interacts with the hypervisor on each compute node. It is responsible for spawning, managing, and terminating VM instances on the physical host.
4. **nova-conductor:** Acts as a mediator between the nova-compute services and the database. It handles complex object conversions and reduces direct database access from compute nodes, enhancing security.
5. **nova-network or Neutron:** Earlier versions of NOVA used nova-network for networking, but it has largely been deprecated in favor of Neutron, OpenStack's dedicated networking service.

6. **nova-consoleauth, nova-novncproxy, and nova-spicehtml5proxy:** These services support remote console access to instances, enabling users to access VM terminals through web interfaces.
7. **Message Queue (RabbitMQ, Qpid, etc.):** All internal services communicate asynchronously using a message queue system. This decouples services and improves system resilience.

5.1.3 Workflow of Instance Provisioning:

When a user requests a new VM through the dashboard (Horizon) or CLI/API, the following steps are executed:

1. **API Request:** The request is received by nova-api.
2. **Authentication:** Keystone authenticates the user credentials.
3. **Scheduling:** The nova-scheduler determines the most suitable compute node for the instance.
4. **Instance Creation:** The nova-compute service on the selected node interacts with the hypervisor to launch the instance.
5. **Networking & Storage:** NOVA interacts with Neutron to attach networking interfaces and with Cinder/Glance to provision storage or retrieve disk images.
6. **Monitoring:** Once launched, the instance is monitored for performance and health, with options for live migration or scaling as needed.

5.1.4 Hypervisor Support and Plugins:

NOVA is designed to be hypervisor-agnostic and supports various hypervisors through drivers. Some commonly supported hypervisors include:

- **KVM/QEMU:** Most widely used and preferred hypervisors in OpenStack deployments.
- **VMware vSphere:** Supported through dedicated drivers allowing enterprises to integrate existing VMware infrastructure.
- **Xen and XenServer:** Supported with specific configurations.
- **Hyper-V:** Offers Microsoft ecosystem integration.
- **LXC and Containers:** NOVA can be extended to work with container technologies through additional plugins.

NOVA is the backbone of the OpenStack compute service and plays a crucial role in providing cloud-based virtual machine management. Its modular and scalable architecture allows cloud administrators to manage compute workloads efficiently while offering flexibility across different hypervisor platforms. By coordinating with other OpenStack components, NOVA ensures that compute resources are provisioned securely, efficiently, and in a way that supports both private and public cloud deployment models. It continues to evolve as cloud technologies advance, with growing support for edge computing, bare-metal provisioning, and container orchestration platforms.

Lab Implementation Activity 5.1

Steps: Setting up NOVA (Compute Service)

Setup is for the **controller node** and **compute node** on separate systems.

I. On the Controller Node

1. Install Required Packages

```
sudo apt update
sudo apt install nova-api nova-conductor nova-novncproxy nova-scheduler
```

2. Configure /etc/nova/nova.conf

```
[DEFAULT]
transport_url = rabbit://openstack:RABBIT_PASS@controller
auth_strategy = keystone
my_ip = 192.168.0.11
[api_database]
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api
[database]
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova
[keystone_authtoken]
auth_url = http://controller:5000/v3
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS
```

(Continue configuration for other sections like [vnc], [glance], [oslo_concurrency])

3. Sync Database

```
sudo nova-manage api_db sync
sudo nova-manage cell_v2 map_cell0
sudo nova-manage cell_v2 create_cell --name=cell1 --verbose
sudo nova-manage db sync
```

4. Restart NOVA Services

```
sudo systemctl restart nova-api nova-scheduler nova-conductor nova-novncproxy
sudo systemctl enable nova-api nova-scheduler nova-conductor nova-novncproxy
```

II. On the Compute Node

1. Install Nova Compute

```
sudo apt update
sudo apt install nova-compute
```

2. Configure /etc/nova/nova.conf

```
[DEFAULT]
transport_url = rabbit://openstack:RABBIT_PASS@controller
auth_strategy = keystone
my_ip = 192.168.0.31
[keystone_authtoken]
auth_url = http://controller:5000/v3
…
```

```
[libvirt]
virt_type = kvm
[glance]
api_servers = http://controller:9292
```

3. Restart Compute Service

```
sudo systemctl restart nova-compute
sudo systemctl enable nova-compute
```

III. Verification

On the controller node, check for registered compute services:

```
openstack compute service list
```

To launch a test instance:

1. Create a flavor
2. Upload an image to Glance
3. Create a network via Neutron
4. Launch instance:

```
openstack server create --flavor m1.small --image cirros --network private --security-group default --key-name mykey test-vm
```

Practical Activity 5.1 To understand and explore the basic functionality and usage of Nova (Compute Service) in OpenStack:

Materials Required:

- Operating System: Ubuntu Server (20.04 LTS or compatible)
- Virtualization Platform: VirtualBox / KVM / VMware Workstation
- OpenStack distribution (DevStack or PackStack for practice environment)
- Python (latest supported version for OpenStack services)
- Git (for cloning repositories like DevStack)
- Database server (MySQL/MariaDB)
- Message Queue (RabbitMQ)
- Web Server (Apache HTTP Server with mod_wsgi)

OpenStack is a free, open-source cloud computing platform that manages large pools of compute, storage, and networking resources. It consists of multiple interrelated services. This practical introduces students to the core services that power cloud infrastructure.

➢ Nova (Compute Service)

Nova is responsible for provisioning and managing virtual machines (VMs) or compute instances.

Key Features:

- Launch, stop, suspend VMs
- Schedule VMs across hosts
- Connect with networking and storage services

Step 1: Source credentials:

```
bash
source admin-openrc.sh
```

Step 2: Launch an instance:

```
bash
```

```
openstack server create --flavor m1.small --image Ubuntu20 --nic net-id=<netid> --security-group default --key-name mykey myVM
```

Step 3: Check VM status:

```
bash
openstack server list
```

5.2 Cinder (OpenStack Block Storage Service)

Cinder is the block storage component of the OpenStack platform. It provides persistent block storage to running virtual machines, similar to how traditional hard drives work in physical systems. These block volumes can be dynamically created, attached to instances, detached, resized, and even backed up or cloned. The design of Cinder is modular and extensible, allowing it to support a wide variety of storage backends including LVM, iSCSI, NFS, Ceph, NetApp, Dell EMC, and more.

Cinder provides the flexibility of abstracting the block storage layer from the underlying physical storage infrastructure, making it easier to manage and automate storage provisioning in a cloud environment. It supports features like volume snapshots, backup and restore, volume replication, and quality of service (QoS) for I/O operations.

- **Persistent Storage:** Cinder volumes retain data independently of the lifecycle of a virtual machine.
- **Multi-Backend Support:** Works with a variety of commercial and open-source storage systems.
- **Snapshot and Cloning:** Volumes can be snapshotted and cloned, supporting backup and disaster recovery.
- **Volume Encryption:** Supports encrypted volumes for data protection.
- **Backup Integration:** Allows volume data to be backed up to secondary storage systems.
- **QoS and Scheduling:** Cinder supports rules for provisioning volumes based on performance needs or resource availability.

5.2.1 Cinder Architecture:

Cinder follows a distributed, service-oriented architecture. It includes the following key components:

1. **cinder-api:** This is the front-end service that receives and processes API requests. It validates requests, enforces quotas, and sends commands to other Cinder services via a message queue.
2. **cinder-scheduler:** Determines where a new volume should be created based on filters and weightings such as available capacity or storage type.
3. **cinder-volume:** This is the heart of Cinder. It interacts with the backend storage system (such as LVM, Ceph, NetApp) to manage the lifecycle of volumes.
4. **cinder-backup:** Manages backup and restore operations for volumes. Backups can be stored in object storage like Swift or other systems.
5. **Message Queue (RabbitMQ):** Facilitates communication between various Cinder services asynchronously.
6. **Database (MySQL/MariaDB):** Stores metadata about volumes, snapshots, backups, quotas, etc.

5.2.2 Volume Provisioning Workflow:

Here's a typical workflow when a volume is created and attached to an instance:

1. **API Request:** User requests a new volume via Horizon (dashboard), CLI, or REST API.
2. **cinder-api:** Processes the request and places it into the message queue.
3. **cinder-scheduler:** Selects the appropriate backend based on current resource status and policy.
4. **cinder-volume:** Interacts with the chosen backend (e.g., creates a volume using LVM or Ceph).
5. **Nova Integration:** Once created, the volume is attached to a running instance by Nova (compute service).
6. **Mounting:** Inside the VM, the volume appears as a new block device and can be formatted and used.

Supported Backends and Drivers:

Cinder is highly extensible and supports various drivers for backend integration. These include:

- LVM (default reference implementation)
- Ceph RBD (RADOS Block Device)
- NFS
- iSCSI targets
- Fibre Channel SANs
- Vendor Drivers: Dell EMC, NetApp, HPE 3PAR, Hitachi, IBM Storwize, Pure Storage, etc.

Admins can configure multiple backends simultaneously using **volume types** and **backend names**.

Important Configuration Parameters (in **cinder.conf**):

```
[DEFAULT]

enabled_backends = lvm

auth_strategy = keystone

transport_url = rabbit://openstack:RABBIT_PASS@controller

my_ip = 192.168.0.11

[lvm]

volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver

volume_group = cinder-volumes

iscsi_protocol = iscsi

iscsi_helper = tgtadm
```

This configuration sets up an LVM-based backend with iSCSI support.

Security and Access Control:

- **Volume Encryption:** Cinder supports in-transit and at-rest volume encryption using Barbican.
- **Access Control:** Only the tenant who owns the volume can attach or detach it unless explicitly shared.
- **Multi-Attach:** Some backends support attaching a volume to multiple instances simultaneously (read-only or read-write modes).

Use Cases of Cinder:

- **Database Storage:** Block storage is ideal for relational and NoSQL databases.
- **Persistent Application Data:** Applications that require reliable, long-term storage (e.g., ERP systems).
- **Backup and Restore Operations:** Easily clone or snapshot volumes for backup purposes.
- **High Availability Deployments:** Cinder volumes can be used for storing critical state data across multiple nodes.

Cinder is a powerful and essential component of the OpenStack ecosystem that enables flexible, enterprise-grade block storage management. With its modular architecture and wide range of backend support, it can be tailored to meet the needs of both small deployments and large-scale cloud environments. Its ability to integrate with other OpenStack services like Nova, Keystone, and Glance ensures seamless data flow and efficient storage lifecycle management. As organizations increasingly move toward hybrid and private cloud models, Cinder provides a robust solution for delivering dynamic, persistent, and secure storage services.

Assignment 5.1

1. What is the primary role of the NOVA service in OpenStack?
2. What type of storage does the Cinder service provide to virtual machines?
3. Does the NOVA service support different types of hypervisors?
4. Does Cinder only work with a single type of storage backend, or can it work with multiple?

Practical Activity 5.2 To understand and explore the basic functionality and usage of Cinder (Block Storage) in OpenStack:

Materials Required:

- Operating System: Ubuntu Server (20.04 LTS or compatible)
- Virtualization Platform: VirtualBox / KVM / VMware Workstation
- OpenStack distribution (DevStack or PackStack for practice environment)
- Python (latest supported version for OpenStack services)
- Git (for cloning repositories like DevStack)
- Database server (MySQL/MariaDB)
- Message Queue (RabbitMQ)
- Web Server (Apache HTTP Server with mod_wsgi)

  ➢ Cinder (Block Storage)

Cinder provides block-level storage to instances.

Key Features:

- Persistent volumes
- Attach/detach volumes to/from VMs
- Snapshot and backup support

Step 1: Create a volume:

```
bash
openstack volume create --size 5 demo-volume
```

Step 2: Attach volume:

```
bash
openstack server add volume myVM demo-volume
```

Step 3: Check status:

```
bash
openstack volume list
```

5.3 Glance – OpenStack Image Service

Glance is the image service in OpenStack, responsible for discovering, registering, and retrieving virtual machine images. These images are templates used to create new virtual machine instances. Glance acts as a central image catalog for OpenStack and supports various disk formats such as QCOW2, RAW, VMDK, and VHD. It allows users and administrators to upload and manage images that can be used across different projects and instances.

Glance stores metadata about images (name, size, type, checksum, etc.) in a database, while the actual image files can be stored in a variety of backends including the local file system, OpenStack Swift, Amazon S3, and Ceph.

- Image discovery, registration, and delivery
- Support for multiple image formats
- Pluggable backend storage
- Image versioning and metadata support
- Secure access using Keystone
- Integration with Nova to launch instances from images

5.3.1 Glance Architecture:

1. **glance-api:** Handles all API requests, including image upload, retrieval, and deletion.
2. **glance-registry:** Stores and retrieves image metadata (deprecated in newer versions, replaced by API DB handling).
3. **Database:** Stores image metadata (e.g., image name, size, owner).
4. **Image Storage Backends:** Stores actual image data. Options include file, Swift, Ceph, etc.
5. **Keystone:** Provides authentication and access control for Glance APIs.

5.3.2 Use Cases:

- Upload and maintain a library of base operating system images
- Use custom application images for rapid deployment

- Store snapshots of running instances for backup or migration

Glance is a foundational service in OpenStack environments. It decouples image storage and metadata management from compute operations and enables consistent image provisioning across cloud environments.

Practical Activity 5.3 To understand and explore the basic functionality and usage of Glance (Image Service) in OpenStack:

Materials Required:

- Operating System: Ubuntu Server (20.04 LTS or compatible)
- Virtualization Platform: VirtualBox / KVM / VMware Workstation
- OpenStack distribution (DevStack or PackStack for practice environment)
- Python (latest supported version for OpenStack services)
- Git (for cloning repositories like DevStack)
- Database server (MySQL/MariaDB)
- Message Queue (RabbitMQ)
- Web Server (Apache HTTP Server with mod_wsgi)

➢ Glance (Image Service)

Glance stores and retrieves virtual machine disk images.

- Upload, download, share images
- Store images in different formats (qcow2, raw)
- Public/private image control

Step 1: List available images:

```
bash
openstack image list
```

Step 2: Upload an image:

```
bash
openstack image create "Ubuntu22" --disk-format qcow2 --container-format bare --file ubuntu22.qcow2 --public
```

5.4 Neutron - OpenStack Networking Service

Neutron is the networking component of OpenStack, providing "Networking as a Service" between interface devices managed by other OpenStack services (like Nova). It allows users to define network connectivity and IP addressing for instances, manage subnets, and configure network topologies dynamically.

Neutron supports advanced features such as load balancing, firewalls, VPNs, and software-defined networking (SDN) integration.

- Self-service network creation
- IP address management (IPAM)
- VLAN, GRE, VXLAN support
- DHCP, DNS, and NAT support
- Floating IPs for external access
- Integration with SDN controllers like Open vSwitch, OVN, Cisco ACI, and more

5.4.1 Neutron Architecture:

1. **neutron-server:** Receives API calls and interacts with the database and plugins.

2. **ML2 Plugin:** Modular Layer 2 plugin that supports multiple networking mechanisms.
3. **Agents:**
   - **L3 agent:** Manages routing and NAT.
   - **DHCP agent:** Assigns IP addresses.
   - **Metadata agent:** Provides instance metadata.
4. **Message Queue:** Facilitates communication between agents and server.
5. **Database:** Stores network topology and configuration data.

5.4.2 Use Cases:

- Provide tenant-isolated networks in multi-tenant cloud
- Assign floating IPs for public access
- Create custom routing and firewall rules
- Automate network provisioning for DevOps environments

Neutron enables flexible, programmable networking in OpenStack. It decouples network provisioning from hardware and provides the building blocks for creating complex, multi-tier cloud architectures.

Practical Activity 5.4 To understand and explore the basic functionality and usage of Neutron (Networking) in OpenStack:

Materials Required:

- Operating System: Ubuntu Server (20.04 LTS or compatible)
- Virtualization Platform: VirtualBox / KVM / VMware Workstation
- OpenStack distribution (DevStack or PackStack for practice environment)
- Python (latest supported version for OpenStack services)
- Git (for cloning repositories like DevStack)
- Database server (MySQL/MariaDB)
- Message Queue (RabbitMQ)
- Web Server (Apache HTTP Server with mod_wsgi)
  - ➢ Neutron (Networking)

Neutron provides network-as-a-service (NaaS) functionality.

Key Features:

- Manage networks, subnets, routers
- Support for private/public IPs
- Security groups and floating IPs

Step 1: Create a network:

```
bash
openstack network create demo-net
```

Step 2: Create a subnet:

```
bash
openstack subnet create --network demo-net --subnet-range 192.168.1.0/24 demo-subnet
```

Step 3: Assign a floating IP:

```
bash
openstack floating ip create public
openstack server add floating ip myVM 203.0.113.15
```

5.5 Horizon – OpenStack Dashboard

Horizon is the official web-based dashboard for OpenStack. It provides a graphical user interface (GUI) to interact with the various OpenStack services like Nova, Glance, Neutron, Cinder, and Keystone. It is built using Django, a Python web framework.

Horizon simplifies cloud management by providing easy access to features such as launching instances, creating networks, managing users, and monitoring usage all from a browser.

5.5.1 Key Features of Horizon

1. Web-Based Interface

Horizon offers an intuitive, responsive, and interactive interface that can be accessed via any modern web browser. Users can view dashboards, launch instances, configure security groups, and perform other cloud operations without needing to interact with CLI or API tools.

2. Multi-Tenancy Support

The dashboard supports multiple projects (tenants), allowing users and administrators to work within isolated resource boundaries. Users only see resources and operations allowed by their assigned roles and permissions.

3. Role-Based Access Control

Horizon integrates with Keystone, the OpenStack identity service, to enforce fine-grained access control. Depending on their roles (admin, member, read-only, etc.), users can view and manage specific resources or perform restricted actions.

4. Service Integration

Horizon supports seamless interaction with multiple OpenStack services:

- **Nova**: Launch, pause, resize, or delete instances.
- **Glance**: Manage image repositories.
- **Cinder**: Create and attach volumes.
- **Neutron**: Configure networks, routers, floating IPs.
- **Heat**: Deploy orchestration stacks.
- **Swift**: Manage object storage containers.
- **Keystone**: Manage users, roles, domains, and projects.

5.5.2 Architecture of Horizon

1. Django Framework

Horizon is built using Django, a high-level Python web framework. This provides a robust MVC (Model-View-Controller) pattern and simplifies web development tasks.

2. Horizon and Dashboards

Horizon is the top-level project containing multiple dashboards. Each dashboard is a collection of panels and views corresponding to a specific OpenStack service.

- **Admin Dashboard**: For administrators to manage global settings and resources.
- **Project Dashboard**: For users to manage instances, storage, and networking in their own projects.
- **Identity Dashboard**: For managing user roles, domains, and authentication settings.

3. REST API Interaction

Horizon communicates with OpenStack services via REST APIs. It acts as a client that sends API calls to services such as Nova, Glance, and Neutron. The responses are rendered as web pages using Django templates.

4. Keystone Integration

Authentication and session management are handled by Keystone. Users are authenticated before accessing the dashboard, and their session tokens determine their permissions and visibility.

5.5.3 Advantages of Using Horizon

- **Accessibility**: Provides a centralized, GUI-based access point for all OpenStack services.
- **Ease of Use**: Simplifies cloud management for non-technical users or developers new to OpenStack.
- **No Local Installation**: Can be accessed through any browser without requiring software installation.
- **Comprehensive Control**: Nearly all OpenStack functionalities can be accessed and controlled via the dashboard.
- **Multi-language Support**: Supports internationalization (i18n) for multilingual users.

Horizon enhances OpenStack's usability, particularly for those not comfortable with CLI tools. Its intuitive interface makes it easy to learn and operate a cloud environment.

Assignment 5.2

1. What is the main function of the Glance service in OpenStack?
2. What kind of service does OpenStack's Neutron component provide?
3. What is Horizon and what does it offer users?
4. Does the text say that Horizon requires users to install software locally?

Practical Activity 5.5 To understand and explore the basic functionality and usage of Horizon (Web Dashboard) in OpenStack:

Materials Required:

- Operating System: Ubuntu Server (20.04 LTS or compatible)
- Virtualization Platform: VirtualBox / KVM / VMware Workstation
- OpenStack distribution (DevStack or PackStack for practice environment)
- Python (latest supported version for OpenStack services)
- Git (for cloning repositories like DevStack)
- Database server (MySQL/MariaDB)
- Message Queue (RabbitMQ)
- Web Server (Apache HTTP Server with mod_wsgi)
  - ➢ Horizon (Web Dashboard)

Horizon is the web-based GUI to interact with OpenStack services.

Key Features:

- Manage projects, users, VMs, volumes, networks
- Launch instances easily
- View usage and monitoring data

Step 1: Access: http://<controller-ip>/dashboard

Step 2: Login with admin credentials

Step 3: Navigate:

- Compute → Instances
- Network → Topology
- Project → Volumes

5.6 Heat – OpenStack Orchestration Service

Heat is the orchestration component of OpenStack that enables the automation of cloud resource provisioning and management. It allows users to define the infrastructure required for their applications using a human-readable, template-driven approach. These templates, written in YAML, describe the relationships and configurations of resources such as servers, floating IPs, volumes, networks, security groups, and more.

Heat automates the deployment of composite cloud applications and services, reducing manual steps and the possibility of human error. It plays a key role in Infrastructure-as-Code (IaC), allowing users to version, replicate, and scale their infrastructure as easily as they manage software code.

1. Template-Driven Orchestration

At the heart of Heat is the **Heat Orchestration Template (HOT)** format. These YAML-based templates define all the resources, their dependencies, and properties. Heat interprets the template and provisions the resources accordingly.

2. Infrastructure-as-Code

Heat templates can be stored in version control systems (like Git), reused across environments, and tested for consistency, embodying the principles of DevOps and IaC.

3. Resource Management

Heat interacts with various OpenStack services:

- **Nova** for compute instances
- **Glance** for images
- **Neutron** for networks
- **Cinder** for block storage
- **Keystone** for authentication
- **Swift** for object storage

4. Stack Creation

Users can launch a group of interdependent resources as a **stack**, which is the fundamental unit of deployment in Heat. A stack can be created, updated, or deleted in one operation.

5. Auto-Scaling and Monitoring

Heat integrates with **Ceilometer** and **Aodh** to support auto-scaling and alarms. For example, when CPU usage crosses a threshold, Heat can trigger the launch of new instances automatically.

5.6.1 Benefits of Using Heat

1. Automation

Heat automates the provisioning and configuration of multiple resources, reducing manual errors and saving time.

2. Consistency

Templates ensure that environments can be reproduced reliably across dev, test, and production environments.

3. Scalability

With auto-scaling policies, Heat can dynamically adjust resources based on workload or custom-defined alarms.

4. Modularity

Templates can be split into reusable components (nested stacks), promoting clean, modular design of infrastructure.

5. Rollback and Recovery

In the event of deployment failures, Heat can automatically roll back the changes, preventing partial or inconsistent states.

5.6.2 Typical Use Cases of Heat

- **Web App Deployment**: Deploying a full stack of web servers, databases, load balancers, and networks.
- **DevOps CI/CD Pipelines**: Automating infrastructure setup as part of software delivery.
- **Disaster Recovery**: Reproducing the same infrastructure in another region in case of failure.
- **Demo Environments**: Spinning up test environments for demos or training purposes.

Heat brings powerful orchestration capabilities to OpenStack. It enables consistent, automated deployments, saving time and reducing configuration errors.

Practical Activity 5.6 To understand and explore the basic functionality and usage of Heat (Orchestration) in OpenStack:

Materials Required:

- Operating System: Ubuntu Server (20.04 LTS or compatible)
- Virtualization Platform: VirtualBox / KVM / VMware Workstation
- OpenStack distribution (DevStack or PackStack for practice environment)
- Python (latest supported version for OpenStack services)
- Git (for cloning repositories like DevStack)
- Database server (MySQL/MariaDB)
- Message Queue (RabbitMQ)
- Web Server (Apache HTTP Server with mod_wsgi)
    - ➢ Heat (Orchestration)

Heat automates the deployment of cloud resources using templates (HOT - Heat Orchestration Templates).

Key Features:

- Template-driven resource creation
- Automates multi-tier app deployment
- Dependency handling

Step 1: Write a simple YAML template (mytemplate.yaml)

```
yaml
heat_template_version: 2013-05-23
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      image: Ubuntu20
      flavor: m1.small
      networks:
        - network: demo-net
```

Step 2: Create stack:

```
bash
openstack stack create -t mytemplate.yaml mystack
```

Step 3: View stack status:

```
bash
openstack stack list
```

5.7 Ceilometer – OpenStack Telemetry (Billing and Monitoring)

Ceilometer is the telemetry and metering component of the OpenStack ecosystem. It plays a central role in **monitoring resource usage**, **collecting performance metrics**, and **generating billing information** in OpenStack cloud environments. As cloud providers increasingly need to track and charge users based on their consumption of compute, storage, and network resources, Ceilometer fulfills this critical requirement by offering data collection, aggregation, and forwarding capabilities.

Originally designed for metering and billing, Ceilometer has evolved into a robust telemetry system that collects resource usage statistics and can integrate with **Aodh** for alarming and **Gnocchi** for time-series data storage. Its flexibility and extensibility make it an essential service for cloud administrators who require detailed insights into infrastructure operations and tenant-level usage.

- Resource usage tracking (CPU, RAM, Disk, Network)
- Event and alarm triggering
- Metering for billing
- REST API for data access
- Pluggable backend support (MongoDB, Gnocchi)

5.7.1 Ceilometer Architecture

The architecture of Ceilometer consists of several modular components:

1. Pollsters

These agents collect data by polling other OpenStack components such as Nova (compute), Cinder (block storage), Neutron (networking), and Glance (image service). Pollsters generate samples that represent usage metrics.

2. Notification Agents

OpenStack services can emit notifications via the message queue (such as RabbitMQ). Notification agents listen to these messages and convert them into meters for storage and analysis.

3. Pipeline

The pipeline configuration determines how collected data flows through the system. It defines sources (pollsters, notifications), transformations (unit conversions, calculations), and sinks (destinations such as file, database, or HTTP endpoint).

4. Dispatcher

The dispatcher sends the final metered data to storage backends like Gnocchi, MongoDB (deprecated), or other supported databases.

5. API Server

The Ceilometer API provides access to stored metering data. Users or applications can retrieve usage statistics using RESTful queries.

5.7.2 Typical Ceilometer Workflow

1. **User Launches an Instance**: A VM is launched using Nova.
2. **Ceilometer Polls Nova**: A pollster queries Nova for resource metrics like CPU time, disk I/O, etc.
3. **Notification Received**: Nova also sends events (like instance creation) to the message queue.
4. **Data Processing**: Notification agents capture events and generate meters.
5. **Data Stored**: Metrics are pushed to a backend like Gnocchi.
6. **API Access**: Users or admins query the API to retrieve usage data for billing or monitoring.

Ceilometer is a vital service in OpenStack for collecting, managing, and analyzing cloud usage data. Its integration with other components like Aodh (for alarming) and Gnocchi (for time-series data storage) makes it a comprehensive solution for billing, monitoring, and infrastructure management. For cloud providers and enterprises adopting OpenStack, Ceilometer enables cost visibility, proactive resource management, and detailed operational insights.

As OpenStack environments grow, Ceilometer continues to evolve, focusing more on performance and integration with modern data platforms, ensuring that OpenStack remains a competitive and capable open-source cloud solution.

Assignment 5.3

1. What is the main purpose of the Heat component in OpenStack?
2. What is the role of Ceilometer in OpenStack environments?
3. What format are the templates used by Heat written in?
4. Does Ceilometer help with monitoring resource usage?

Practical Activity 5.7 To understand and explore the basic functionality and usage of Ceilometer (Telemetry/Metering) in OpenStack:

Materials Required:

- Operating System: Ubuntu Server (20.04 LTS or compatible)
- Virtualization Platform: VirtualBox / KVM / VMware Workstation
- OpenStack distribution (DevStack or PackStack for practice environment)
- Python (latest supported version for OpenStack services)
- Git (for cloning repositories like DevStack)
- Database server (MySQL/MariaDB)
- Message Queue (RabbitMQ)
- Web Server (Apache HTTP Server with mod_wsgi)

➢ Ceilometer (Telemetry/Metering)

Ceilometer collects usage and performance data from OpenStack services.

Key Features:

- Monitor resource usage
- Enable billing and chargeback
- Collect CPU, memory, disk, and bandwidth metrics

Step 1: List meters:

```
bash
openstack metric resource list
```

Step 2: Show CPU metrics:

```
bash
openstack metric measures show --resource-id <resource-id> cpu_util
```

5.8 Trove – OpenStack Database-as-a-Service

Trove is the Database-as-a-Service (DBaaS) component of the OpenStack cloud ecosystem. It provides scalable and reliable cloud-based database management capabilities, allowing users to **provision and manage different types of databases** without worrying about the underlying infrastructure. Trove supports a variety of relational and non-relational databases including MySQL, MariaDB, PostgreSQL, MongoDB, Cassandra, and Redis.

As modern applications rely heavily on database services, Trove simplifies database provisioning, monitoring, scaling, and management delivering a managed experience similar to what is offered by public cloud platforms like Amazon RDS or Google Cloud SQL, but within an open-source and self-managed environment.

Key Features:

- Support for multiple database engines
- Automated backups and restores
- Configuration management
- Horizontal and vertical scaling
- Multi-tenant secure isolation

5.8.1 Core Features of Trove

1. Multi-Database Support

Trove supports both relational and NoSQL databases:

- **Relational**: MySQL, MariaDB, PostgreSQL, Oracle, Percona
- **NoSQL**: MongoDB, Cassandra, Redis, Couchbase

This flexibility allows users to choose the right database technology for their application needs.

2. Automated Provisioning

Using Trove, users can launch a fully functional database instance in just a few clicks or API calls. Trove handles the backend orchestration including VM or container setup, database installation, configuration, and security.

3. Backup and Restore

Trove provides built-in support for creating backups of database instances. These backups can later be restored into new or existing instances, facilitating disaster recovery and data migration.

4. Horizontal and Vertical Scaling

Trove allows users to resize databases (changing RAM or vCPU) and scale out by adding replica sets or sharded clusters, depending on the database engine's capabilities.

5. Monitoring and Health Checks

Trove integrates with OpenStack telemetry tools such as **Ceilometer** to monitor resource usage and with **Nagios or custom agents** for health checks, ensuring high availability and performance.

5.8.2 Trove Architecture Overview

Trove's architecture consists of several core components:

1. **Trove API**
   A RESTful interface for users and services to manage database instances. It accepts client requests and forwards them to the task manager.
2. **Trove Task Manager**
   The heart of Trove, it handles the orchestration of provisioning, resizing, backup, and other operations. It interacts with Nova (for VMs), Cinder (for volumes), Neutron (for networking), and Glance (for images).
3. **Trove Conductor**
   Facilitates secure communication between Trove components and handles messaging between guest agents and the control plane.
4. **Guest Agent**
   Runs inside the provisioned database instance and is responsible for executing commands like starting/stopping the database, managing backups, or performing status checks.
5. **Database Image**
   Each database type requires a pre-built Glance image with the database software and the guest agent installed. Trove uses these images when provisioning new instances.

   Trove simplifies and automates the management of databases in OpenStack, enabling cloud administrators and developers to deploy, scale, and maintain databases efficiently. It brings enterprise-grade DBaaS features into private and hybrid cloud environments while maintaining the flexibility of open-source tools. With support for multiple database engines and deep integration with other OpenStack components, Trove is a powerful tool for building cloud-native applications that require reliable and scalable data services.

   As OpenStack continues to evolve, Trove plays a pivotal role in offering database capabilities that match the scale, automation, and self-service expectations of modern cloud users.

Practical Activity 5.8 To understand and explore the basic functionality and usage of Trove (Database as a Service) in OpenStack:

Materials Required:

- Operating System: Ubuntu Server (20.04 LTS or compatible)
- Virtualization Platform: VirtualBox / KVM / VMware Workstation
- OpenStack distribution (DevStack or PackStack for practice environment)
- Python (latest supported version for OpenStack services)
- Git (for cloning repositories like DevStack)
- Database server (MySQL/MariaDB)
- Message Queue (RabbitMQ)
- Web Server (Apache HTTP Server with mod_wsgi)
  - ➢ Trove (Database as a Service)

Trove offers relational and non-relational DBMS provisioning.

Key Features:

- Deploy MySQL, PostgreSQL, MongoDB, etc.
- Manage databases, users, and backups
- Automate database clustering

Step 1: Launch a DB instance:

```
bash
openstack database instance create db1 --flavor m1.small --size 5 --datastore mysql
--datastore-version 5.7
```

Step 2: Create a database inside:

```
bash
openstack database database create db1 mydb
```

5.9 Swift – OpenStack Object Storage

Swift is the **Object Storage** component of OpenStack. It is designed to store and retrieve unstructured data such as documents, images, videos, backups, and virtual machine snapshots. Unlike traditional file systems or block storage solutions, Swift stores data as objects with metadata in a flat namespace, offering **scalability, durability, and accessibility** in distributed environments.

Swift is highly fault-tolerant and replicates data across multiple nodes and locations. It supports **eventual consistency**, meaning data remains available even in the event of hardware failures. This design makes it ideal for cloud applications, backups, archival storage, and large-scale content distribution platforms.

Some key features are:

1. Object Storage Architecture

Swift uses a distributed architecture without a central point of failure. Objects are stored across a cluster of commodity servers, and data replication ensures high availability and fault tolerance.

2. Scalability

Swift is designed to scale horizontally. New nodes can be added without downtime, and the system automatically rebalances data across the cluster.

3. High Availability
Through data replication (typically three copies), Swift ensures that data remains accessible even if one or more storage nodes fail.

4. Multi-Tenancy Support
Swift integrates with **Keystone**, OpenStack's identity service, to support secure multi-tenant object storage. Each tenant (project) can manage its own containers and objects securely.

5. RESTful API Access
Swift provides a REST-based API that allows users and applications to upload, retrieve, and manage objects programmatically or via CLI.

5.9.1 Swift Architecture Components

Swift is made up of several logical components that work together to deliver object storage functionality:

Component	Description
Proxy Server	Handles all incoming API requests and routes them to appropriate storage nodes.
Object Server	Stores the actual data objects and their metadata on disk.
Container Server	Maintains listings of objects in a container (similar to a directory index).
Account Server	Tracks containers under a particular account (user/project).
Ring	A mapping system that determines where data resides in the cluster.
Replicator	Ensures consistent replication of objects, containers, and accounts across nodes.
Auditor	Verifies integrity and consistency of data.

5.9.2 Swift Stores Data
Swift stores data as **objects** in a flat namespace using a three-level hierarchy:

- **Account**: Represents a tenant or user.
- **Container**: A logical grouping of objects (similar to a folder).
- **Object**: The actual data (file/image/etc.) and its associated metadata.

For example:
makefile
Account: project123
Container: images
Object: logo.png
When a file is uploaded, Swift stores it in a specific location determined by the **ring** and creates redundant copies for high availability.

5.9.3 Replication and Consistency
Swift uses **asynchronous replication** to maintain three (or more) copies of each object across different physical devices or nodes. If a node becomes unavailable, Swift automatically redirects traffic to other nodes and later synchronizes the failed node when it comes back online.

Unlike traditional storage systems, Swift is designed for **eventual consistency**, ensuring availability and resilience over strict real-time synchronization.

5.9.4 Use Cases

1. **Backup and Archival Storage**

   Swift is ideal for storing backups, logs, and archives due to its durability and cost-effective scalability.

2. **Media Storage and Streaming**

   Web applications and content delivery platforms use Swift to store and serve media files such as images, videos, and audio.

3. **Cloud-Native Application Storage**

   Applications designed for the cloud can store configuration files, app data, or logs directly in Swift using its API.

4. **VM Image Repository**

   OpenStack Glance can use Swift as a backend for storing virtual machine images.

5.9.5 Advantages of Using Swift

Advantage	Explanation
Durability	Data replication ensures fault tolerance and high durability.
Elastic Scalability	Easily scales to petabytes of data by adding storage nodes.
No Central Point of Failure	Fully distributed architecture ensures uptime.
API Accessibility	REST API enables integration with other tools and services.
Efficient Metadata	Object metadata can be customized and retrieved easily.

5.9.6 Limitations and Considerations

- **Not POSIX-Compliant**: Swift is not suitable for applications requiring traditional file system interfaces.
- **Latency Overhead**: Object storage typically has higher latency than block or file storage.
- **Write-Once Paradigm**: Objects are immutable updating requires re-uploading the full object.
- **Eventual Consistency**: May not be ideal for applications requiring real-time consistency.

Swift is a powerful and reliable object storage solution within the OpenStack ecosystem. Its distributed, fault-tolerant design makes it an excellent choice for large-scale storage of unstructured data. With its RESTful interface, multi-tenant support, and seamless integration with other OpenStack services, Swift enables developers and cloud administrators to build resilient, scalable, and cost-effective storage infrastructure.

Whether you're creating a media hosting service, a backup solution, or a data lake, Swift provides the necessary features and flexibility to support cloud-native applications and services in private, public, or hybrid cloud environments.

Practical Activity 5.9 To understand and explore the basic functionality and usage of Swift (Object Storage) in OpenStack:

Materials Required:

- Operating System: Ubuntu Server (20.04 LTS or compatible)

- Virtualization Platform: VirtualBox / KVM / VMware Workstation
- OpenStack distribution (DevStack or PackStack for practice environment)
- Python (latest supported version for OpenStack services)
- Git (for cloning repositories like DevStack)
- Database server (MySQL/MariaDB)
- Message Queue (RabbitMQ)
- Web Server (Apache HTTP Server with mod_wsgi)
  - ➢ Swift (Object Storage)

Swift provides scalable object storage for unstructured data like images, backups, and logs.

Key Features:

- Store/retrieve objects via API
- Highly available and redundant
- Access control and container-based storage

Step 1: Create a container:

```
bash
openstack container create mycontainer
```

Step 2: Upload a file:

```
bash
openstack object create mycontainer demo.txt
```

Step 3: List objects:

```
bash
openstack object list mycontainer
```

SUMMARY

- A modular, distributed cloud platform providing IaaS through interrelated components (Nova, Neutron, Cinder, Glance, Keystone, Horizon) that communicate via APIs and message queues.
- Core service for provisioning and managing virtual machines (VMs), supporting multiple hypervisors (KVM, VMware, etc.) and featuring components like nova-api, nova-scheduler, and nova-compute.
- API requests are authenticated by Keystone, scheduled by nova-scheduler, and VMs are launched by nova-compute on a host, interacting with Neutron for networking and Cinder/Glance for storage/images.
- Provides persistent block storage volumes for VMs, supporting various backends (LVM, Ceph, NFS), snapshots, and volume encryption. Key components include cinder-api, cinder-scheduler, and cinder-volume.
- Manages discovery, registration, and retrieval of VM images (templates like QCOW2, RAW), storing metadata in a database and image files in backends like Swift or Ceph.
- Delivers "Networking as a Service," allowing users to define networks, subnets, routers, and manage IP addressing (including floating IPs), supporting VLANs, VXLANs, and SDN integration through ML2 plugins and agents (L3, DHCP).
- The official web-based GUI for OpenStack, built on Django, enabling users to manage resources across services (Nova, Cinder, Neutron, etc.) with multi-tenancy and role-based access control via Keystone.
- Automates cloud resource provisioning using template-driven (YAML-based HOT - Heat Orchestration Templates) infrastructure-as-Code, managing interdependent resources as "stacks."

- Collects resource usage and performance metrics (CPU, RAM, disk, network) for monitoring, billing, and alarming, using pollsters and notification agents.
- Provides scalable and reliable DBaaS, allowing provisioning and management of various database engines (MySQL, PostgreSQL, MongoDB) with features like automated backups and scaling.
- Stores and retrieves unstructured data (documents, images, backups) as objects in a distributed, fault-tolerant, and scalable manner using a RESTful API and components like proxy, object, container, and account servers.
- Organizes data in a hierarchy of Account, Container, and Object, with data replication (typically three copies) ensuring high availability and eventual consistency.
- Each OpenStack service performs a specific function but integrates with others (e.g., Nova uses Glance for images, Cinder for volumes, Neutron for networks, and Keystone for auth).
- All services expose RESTful APIs, which are used for inter-service communication and user interaction (via CLI, SDKs, or Horizon).
- The distributed nature of OpenStack services allows them to scale horizontally, and the pluggable architecture (e.g., hypervisor drivers in Nova, storage backends in Cinder) offers deployment flexibility.

ASSESSMENT

A. Multiple Choice Questions

1. Which OpenStack service is primarily responsible for provisioning and managing virtual machines?
   a) Cinder
   b) Neutron
   c) Nova
   d) Glance

2. What is the name of the OpenStack service that provides persistent block storage to instances?
   a) Swift
   b) Cinder
   c) Heat
   d) Keystone

3. Glance is the OpenStack service responsible for:
   a) Network management
   b) User authentication
   c) Storing and retrieving virtual machine disk images
   d) Orchestrating application deployment

4. Which component of Nova decides where a new virtual machine instance should be hosted?
   a) nova-api
   b) nova-compute
   c) nova-conductor
   d) nova-scheduler

5. Heat Orchestration Templates (HOT) are typically written in which format?
   a) JSON
   b) XML
   c) YAML
   d) Python

6. Which OpenStack service provides "Networking as a Service" and manages IP addresses, subnets, and routers?
   a) Horizon
   b) Neutron
   c) Ceilometer
   d) Trove

7. The web-based dashboard for OpenStack that provides a GUI to interact with various services is called:
   a) Swift
   b) Horizon
   c) Keystone
   d) Heat

8. Ceilometer in OpenStack is primarily used for:
   a) Object storage
   b) Database management
   c) Telemetry, metering, and billing
   d) Identity management

9. Which OpenStack service allows users to provision and manage different types of databases like MySQL and MongoDB?
   a) Cinder
   b) Glance
   c) Trove
   d) Nova

10. Swift stores data as objects in a three-level hierarchy. Which of the following is the top level of this hierarchy?
   a) Object
   b) Container
   c) Bucket
   d) Account

B. Fill-in-the-blanks

1. The core component of the OpenStack architecture that provides compute functionality and manages virtual machines is called ____.
2. In NOVA's architecture, the ____ component is responsible for deciding where a new virtual machine instance should be hosted.
3. Cinder provides persistent ____ storage to running virtual machines, similar to traditional hard drives.
4. Glance is the image service in OpenStack, responsible for discovering, registering, and retrieving virtual machine ____.

5. Neutron's ____ Plugin is a Modular Layer 2 plugin that supports multiple networking mechanisms.
6. Horizon, the OpenStack dashboard, is built using ____, a Python web framework.
7. Heat uses ____ Orchestration Templates (HOT) written in YAML to define infrastructure.
8. Ceilometer's ____ are agents that collect data by polling other OpenStack components like Nova and Cinder.
9. In Trove's architecture, the ____ Agent runs inside the provisioned database instance and executes commands like starting/stopping the database.
10. Swift uses a mapping system called the ____ to determine where data resides in the cluster.

C. True/False questions

1. Nova is written in Java and supports only the KVM hypervisor.
2. Cinder volumes retain data independently of the lifecycle of a virtual machine.
3. Glance can only store images in the RAW disk format.
4. Neutron uses an L2 agent to manage routing and NAT.
5. Horizon communicates with OpenStack services via command-line interfaces only.
6. Heat uses "stacks" as the fundamental unit of deployment for a group of interdependent resources.
7. Ceilometer's primary function is to provide block storage to virtual machines.
8. Trove is capable of managing both relational and NoSQL databases.
9. Swift is designed for storing structured data and offers POSIX-compliant file system interfaces.
10. In Swift, data replication ensures high availability, typically storing three copies of an object.

D. Short Answer type questions.

1. What is the primary function of the nova-compute component within the Nova service?
2. List two key features of the Cinder block storage service.
3. How does Glance facilitate the creation of new virtual machine instances?
4. What is the role of the ML2 (Modular Layer 2) plugin in Neutron?
5. Describe the main purpose of the Horizon dashboard in an OpenStack environment.
6. What does "Infrastructure-as-Code" mean in the context of the Heat orchestration service?
7. Name two types of agents used by Ceilometer to collect data.
8. What benefit does Trove provide to application developers in an OpenStack cloud?
9. Explain the concept of "eventual consistency" as it applies to Swift.
10. What are the three levels of hierarchy Swift uses to store data?

Answer Key

A. Multiple Choice Questions

1.c, 2.b, 3.c, 4.d, 5.c, 6.b, 7.b, 8.c, 9.c, 10.d

B. Fill-in-the-blanks

1. NOVA, 2. nova-scheduler, 3. Block, 4. Images, 5. ML2, 6. Django, 7. Heat, 8. Pollsters, 9. Guest, 10. Ring

C. True/False questions

1.False, 2. True 3. False, 4. False, 5. False, 6. True, 7. False, 8. True, 9. False, 10. True

Chapter-6

Basics of Troubleshooting in OpenStack

Ravi was a new cloud administrator at a company that used OpenStack. One morning, a developer reported that a virtual machine was not starting. Ravi decided to troubleshoot the issue step by step. First, he checked the logs in Nova and saw an error message. Then, he verified the network service (Neutron) to make sure the VM had the correct IP address. Next, he looked at the storage (Cinder) to confirm the volume was properly attached. By following a simple process — identify the problem, check logs, test connectivity, and verify service status — Ravi found that the issue was a misconfigured network. He fixed it, and the VM launched successfully.

Ravi learned that troubleshooting in OpenStack is like solving a puzzle: break the problem into smaller parts, check each service, and use logs and commands to find the root cause.

OpenStack, an open-source cloud computing platform, follows a modular architecture where different services are distributed among various nodes. These nodes interact with each other to deliver Infrastructure as a Service (IaaS) functionalities such as compute power, networking, storage, and orchestration. The major components in a standard OpenStack deployment include the Controller Node, Network Node, and one or more Compute Nodes. Each plays a critical role in the functioning of the cloud, and understanding their responsibilities is essential for both cloud administrators and developers.

6.1 Controller Node and Its Functions

The **Controller Node** acts as the central brain of the OpenStack cloud. It hosts most of the shared services and databases required for orchestrating and coordinating the cloud's resources. This node typically runs the API services, management processes, authentication services, image services, orchestration engines, and dashboards.

One of the most critical components on the Controller Node is **Keystone**, the identity service. Keystone provides authentication and authorization services for all other OpenStack components. It maintains a list of users, their credentials, roles, and access permissions. Every OpenStack API call is first routed through Keystone to verify the identity of the user or service.

Another significant component is **Glance**, the image service. It manages disk images and snapshots of virtual machines. When users launch instances, the image is pulled from Glance and used as the base disk. Glance supports multiple backends such as file, Swift, or Ceph, allowing administrators to configure storage flexibly.

The **Nova API service** also resides on the Controller Node. While Nova-compute runs on Compute Nodes, the API and scheduler services operate on the Controller. The **Nova scheduler** decides which Compute Node should host a particular instance based on factors like resource availability, load, and filters defined by the administrator.

Neutron server, which manages the networking API and coordination of network components, also runs here. It communicates with agents on other nodes to configure actual networking.

Horizon, the web-based dashboard, is another prominent feature hosted on the Controller Node. This interface allows administrators and users to interact with OpenStack services through a GUI, simplifying management and deployment.

Additionally, the **Cinder API** and scheduler services are part of the Controller Node. While volume creation is executed by volume services, the orchestration and initial API requests are managed here.

The **Heat orchestration engine**, if installed, operates from the Controller Node. It enables users to create complex environments with templates, similar to AWS CloudFormation.

Monitoring and telemetry components such as **Ceilometer**, **Gnocchi**, or **Panko** are also typically deployed on the Controller. These services collect data for billing, monitoring, and resource management.

A database server, commonly **MariaDB** or **MySQL**, stores configuration and state data for all OpenStack services. Each component writes its metadata here, and queries it for decisions. Messaging between services is handled by **RabbitMQ** or **Qpid**, both of which are message queueing systems deployed on the Controller Node.

Therefore, the Controller Node is essential in coordinating the entire cloud environment. It is a critical single point of control and must be configured for high availability and security in a production environment.

Practical Activity 6.1 Demonstration of Environment set up

Materials Needed

- Ubuntu 20.04 Server (x3)
- OpenStack packages (Victoria or Wallaby release recommended)
- OpenStack client tools
- MySQL/MariaDB
- RabbitMQ
- NTP (chrony)
- etcd
- Python3-pip, Git

Step 1: Update and Upgrade Systems

```
bash
sudo apt update && sudo apt upgrade -y
```

Step 2: Set Hostnames

- **Controller Node**:
  ```
  bash
  hostnamectl set-hostname controller
  ```

- **Network Node**:
  ```
  bash
  hostnamectl set-hostname network
  ```
- **Compute Node**:
  ```
  bash
  hostnamectl set-hostname compute
  ```

Step 3: Configure /etc/hosts

Edit /etc/hosts file on each node to include:

```
192.168.0.10 controller

192.168.0.20 network

192.168.0.30 compute
```

Step 4: Install NTP via Chrony

```
bash

sudo apt install chrony -y
```

- Set controller as time server.
- Configure chrony.conf:
  On controller:
  ```
  bash
  server ntp.ubuntu.com iburst
  allow 192.168.0.0/24
  ```
  On other nodes:
  ```
  bash
  server controller iburst
  ```

Step 5: Install OpenStack Packages

```
bash
sudo add-apt-repository cloud-archive:wallaby
sudo apt update && sudo apt dist-upgrade -y
```

6.2 Network Node and Its Functions

The **Network Node** in OpenStack is responsible for handling all network-related services and traffic routing between instances, external networks, and different tenant networks. It acts as the virtual switchboard of the OpenStack cloud, and houses essential Neutron agents and services.

Key components on the Network Node include the **L3 agent**, **DHCP agent**, and **metadata agent**. These agents are vital for implementing network topologies defined by the users and administrators.

The **Neutron L3 agent** provides layer 3 (routing) functionality, enabling communication between different subnets and access to external networks. This agent creates virtual routers and uses iptables or nftables to apply NAT rules and firewalling. When an instance requests internet access, its traffic is routed through the L3 agent, which performs SNAT or DNAT operations accordingly.

The **Neutron DHCP agent** handles dynamic IP address allocation for instances. It ensures that each virtual machine connected to an OpenStack-managed network receives an IP address automatically. It uses dnsmasq as the backend DHCP server. This agent listens for DHCP requests from VMs and responds with the allocated IP configuration, including subnet mask, DNS server, and default gateway.

The **metadata agent** is another critical service on the Network Node. It enables instances to retrieve instance-specific metadata during boot, such as user data scripts or SSH keys. It acts as a proxy to the Nova metadata API and is necessary for instances without direct access to the Controller.

The Network Node is also where **Open vSwitch (OVS)** or **Linux Bridge** is deployed. These software switches connect tenant networks, manage VLANs or VXLANs, and bridge connections between virtual NICs and physical network interfaces. Open vSwitch is preferred in many deployments due to its support for tunneling, quality of service (QoS), and firewall integration.

In environments using **ML2 plugins**, the Network Node implements the logic required to interface with various network backends. Whether using VLANs, GRE, VXLANs, or flat networks, the plugins handle the creation and teardown of logical network elements.

Security in networking is enforced using **Security Groups** and **Firewall-as-a-Service (FWaaS)**. These services define rules to control ingress and egress traffic at the instance level. The rules are implemented using iptables rules programmed by the Neutron agents.

To connect the internal OpenStack networks to the internet, the Network Node uses **external bridges** connected to physical NICs. Floating IPs, which provide internet access to instances, are routed through the external interface of the Network Node.

High availability of the Network Node is vital in production. Multiple Network Nodes can be deployed with **VRRP-based failover** mechanisms such as Keepalived or HAProxy. Load balancing across Network Nodes helps distribute traffic and ensures that services continue operating even during failure of one node.

Overall, the Network Node is responsible for providing robust and programmable network infrastructure. It transforms the virtual networks defined by users into actual packet-forwarding paths, ensuring seamless connectivity.

Practical Activity 6.2. Building a controller node

Materials Needed

- Ubuntu Server LTS (20.04 / 22.04)
- OpenStack packages (Victoria or Wallaby release recommended)
- OpenStack client tools
- MySQL/MariaDB
- RabbitMQ
- NTP (chrony)
- etcd
- Python3-pip, Git

Step 1: Install SQL Database

bash
```
sudo apt install mariadb-server python3-pymysql -y
```
Configure /etc/mysql/mariadb.conf.d/50-server.cnf:

ini
```
bind-address = 0.0.0.0
```
Restart:

bash
```
sudo systemctl restart mariadb
```

Step 2: Install RabbitMQ

bash
```
sudo apt install rabbitmq-server -y
sudo rabbitmqctl add_user openstack RABBIT_PASS
sudo rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

Step 3: Install and Configure etcd

bash
```
sudo apt install etcd -y
```
Update /etc/default/etcd:

ini
```
ETCD_LISTEN_PEER_URLS="http://0.0.0.0:2380"
ETCD_LISTEN_CLIENT_URLS="http://0.0.0.0:2379"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://controller:2380"
ETCD_ADVERTISE_CLIENT_URLS="http://controller:2379"
ETCD_INITIAL_CLUSTER="controller=http://controller:2380"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-01"
ETCD_INITIAL_CLUSTER_STATE="new"
```

Step 4: Install Keystone (Identity Service)

bash
```
sudo apt install keystone apache2 libapache2-mod-wsgi-py3 -y
```
Edit /etc/keystone/keystone.conf:

ini
```
[database]
connection =
mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone
[token]
provider = fernet
```
Create DB, initialize Fernet, and restart:

bash
```
sudo keystone-manage db_sync
sudo keystone-manage fernet_setup --keystone-user keystone --keystone-group
keystone
sudo keystone-manage credential_setup --keystone-user keystone --keystone-
group keystone
sudo keystone-manage bootstrap --bootstrap-password ADMIN_PASS \
  --bootstrap-admin-url http://controller:5000/v3/ \
  --bootstrap-internal-url http://controller:5000/v3/ \
  --bootstrap-public-url http://controller:5000/v3/ \
  --bootstrap-region-id RegionOne
```
Enable Apache:

bash
```
sudo systemctl restart apache2
```

Step 5: Install Horizon

bash
```
sudo apt install openstack-dashboard -y
```
Access via browser:

arduino
```
http://controller/dashboard
```

Practical Activity 6.3. Building a network node

Materials Needed

- Ubuntu Server LTS (20.04 / 22.04)
- OpenStack packages (Victoria or Wallaby release recommended)
- OpenStack client tools
- MySQL/MariaDB
- RabbitMQ
- NTP (chrony)
- etcd
- Python3-pip, Git

Step 1: Install Neutron Agents

```
bash
sudo apt install neutron-linuxbridge-agent neutron-dhcp-agent neutron-
metadata-agent -y
```

Step 2: **Edit the file**

```
/etc/neutron/neutron.conf:
ini
 [DEFAULT]
transport_url = rabbit://openstack:RABBIT_PASS@controller
auth_strategy = keystone
```

Step 3: Edit Linux bridge agent config

```
/etc/neutron/plugins/ml2/linuxbridge_agent.ini:
ini
[linux_bridge]
physical_interface_mappings = provider:ens3
[vxlan]
enable_vxlan = true
local_ip = 192.168.0.20
l2_population = true
[securitygroup]
enable_security_group = true
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

Step 4: Restart all services:

```
bash
sudo systemctl restart neutron-linuxbridge-agent neutron-dhcp-agent neutron-metadata-
agent
```

Assignment 6.1

1. Which node in OpenStack often hosts the Keystone identity service?
2. What is the primary function of the Network Node?
3. What web-based dashboard, used for interacting with OpenStack services, is mentioned as being hosted on the Controller Node?

6.3 Compute Node and Functions

The **Compute Node** is the backbone of an OpenStack cloud as it is the physical or virtual server where actual instances (virtual machines) are hosted. Multiple Compute Nodes can exist in a deployment to provide scalable and distributed compute capacity. Each Compute Node runs the **nova-compute service**, which is responsible for launching and managing VMs.

The nova-compute service interacts with a hypervisor, which could be KVM, QEMU, Xen, or VMware. The choice of hypervisor determines how virtual machines are created and managed. KVM is the most widely used hypervisor in OpenStack due to its performance and integration.

When a user launches an instance, the Nova scheduler (on the Controller Node) selects an appropriate Compute Node. The nova-compute service on that node then handles the lifecycle of the instance spawning, pausing, resuming, stopping, or terminating it.

Each Compute Node must have access to the image repository, typically served by Glance. The image is downloaded onto the local disk of the Compute Node and used to create the root disk for the virtual machine.

In terms of networking, the Compute Node is integrated with **Open vSwitch** or **Linux Bridge**, depending on the Neutron plugin used. These virtual switches connect the VM interfaces to tenant networks and apply the required security group rules. The actual network configuration is orchestrated by Neutron agents, which may also reside on the Compute Node.

Compute Nodes also interact with **Cinder volumes**. If an instance uses a block volume for storage, it is attached to the VM through the Cinder volume driver. This allows for persistent storage independent of the instance lifecycle.

The Compute Node reports its resource usage and availability to the Controller Node. This includes CPU, RAM, disk space, and the number of running instances. These metrics help the Nova scheduler make informed decisions when placing new instances.

Each Compute Node also maintains logs of its activities. Logs from nova-compute, libvirt (KVM), and Neutron agents provide crucial information for troubleshooting and performance tuning. These logs are often centralized using log aggregation tools such as Fluentd or Logstash.

For high availability and fault tolerance, Compute Nodes can be configured with **live migration** capabilities. This allows administrators to move running instances between Compute Nodes with minimal downtime. This is useful during maintenance or load balancing operations.

Monitoring tools such as **Ceilometer**, **Prometheus**, or **Zabbix** collect performance metrics from the Compute Nodes. This helps administrators track resource usage, detect anomalies, and plan for scaling.

In edge deployments, lightweight Compute Nodes may operate closer to end-users, processing data with reduced latency. Such configurations are part of **distributed cloud architectures**, making OpenStack suitable for telco and IoT applications.

The Compute Node is thus a highly resource-intensive component and needs to be equipped with powerful CPUs, sufficient memory, and high-speed storage to ensure optimal performance of the hosted virtual machines.

Practical Activity 6.4. Building a compute node

Materials Needed

- Ubuntu Server LTS (20.04 / 22.04)
- OpenStack packages (Victoria or Wallaby release recommended)
- OpenStack client tools
- MySQL/MariaDB
- RabbitMQ
- NTP (chrony)
- etcd
- Python3-pip, Git

Step 1: Install Nova Compute

bash

```
sudo apt install nova-compute -y
```

Edit /etc/nova/nova.conf:

ini

```
[DEFAULT]
transport_url = rabbit://openstack:RABBIT_PASS@controller
auth_strategy = keystone
my_ip = 192.168.0.30
[api]
auth_strategy = keystone
[keystone_authtoken]
auth_url = http://controller:5000/v3
...
[libvirt]
virt_type = kvm
```

Step 2: Install Neutron Agent (Same as Network Node)

Also install:

bash

```
sudo apt install neutron-linuxbridge-agent -y
```

Configure and restart services:

bash

```
sudo systemctl restart nova-compute neutron-linuxbridge-agent
```

Assignment 6.2

1. What kind of server hosts the actual instances (virtual machines) in an OpenStack cloud?
2. What is the name of the OpenStack service that runs on the Compute Node to launch and manage virtual machines?
3. Where does the Compute Node get the images it needs to create virtual machines?

OpenStack's architecture is distributed and modular, with each node fulfilling specialized roles. The Controller Node provides centralized management and orchestration; the Network Node handles all aspects of virtual and physical networking; and the Compute Node is responsible for hosting and managing virtual machines. Together, these nodes form the core infrastructure of a powerful and scalable cloud platform. Understanding their individual roles and inter-node communications is key to deploying, managing, and troubleshooting an OpenStack cloud. Proper configuration, security, and scaling of these nodes ensure the reliability and efficiency of the cloud environment.

Summary

- The Controller Node is the central brain of an OpenStack cloud, hosting shared services and databases.
- Keystone, on the Controller Node, provides identity, authentication, and authorization for all OpenStack services.
- Glance, on the Controller Node, manages virtual machine disk images and snapshots.
- The Nova API and scheduler services operate on the Controller Node to manage compute instance placement.
- Neutron server on the Controller Node manages networking APIs and coordinates network components.
- Horizon, the web-based dashboard, is hosted on the Controller Node for GUI-based OpenStack management.
- Cinder API and scheduler services for block storage management reside on the Controller Node.
- The Heat orchestration engine, if used, operates from the Controller Node for template-based environment creation.
- Monitoring and telemetry services like Ceilometer are typically deployed on the Controller Node.
- The Network Node handles all network-related services, traffic routing, and hosts Neutron agents like L3 and DHCP.
- The Neutron L3 agent on the Network Node provides routing functionality between subnets and external networks.
- The Neutron DHCP agent on the Network Node manages dynamic IP address allocation for instances.
- Open vSwitch or Linux Bridge on the Network Node connects tenant networks and manages VLANs/VXLANs.
- The Compute Node is where actual virtual machine instances are hosted and runs the nova-compute service.
- The nova-compute service on Compute Nodes interacts with hypervisors (like KVM) to manage VM lifecycles.

ASSESSMENT

A. Multiple Choice Questions

1. Which OpenStack node acts as the central brain, hosting most shared services and APIs?
   a) Compute Node
   b) Network Node
   c) Storage Node
   d) Controller Node

2. What is the primary function of the Keystone service in OpenStack?
   a) Managing virtual machine images
   b) Providing identity, authentication, and authorization
   c) Handling network traffic routing
   d) Orchestrating complex cloud environments

3. Which service on the Controller Node manages disk images and snapshots for virtual machines?
   a) Nova
   b) Neutron
   c) Glance
   d) Cinder

4. The Neutron L3 agent, responsible for routing, typically resides on which node?
   a) Controller Node
   b) Compute Node
   c) Network Node
   d) Swift Node

5. What is the primary role of the nova-compute service?
   a) Scheduling new instances
   b) Providing a web dashboard
   c) Launching and managing VMs on a Compute Node
   d) Managing block storage volumes

6. Which of these is a message queueing system commonly deployed on the Controller Node for inter-service communication?
   a) MySQL
   b) Etcd
   c) RabbitMQ
   d) Chrony

7. What is the main purpose of the Neutron DHCP agent?
   a) To provide routing between subnets
   b) To handle dynamic IP address allocation for instances
   c) To store virtual machine images
   d) To offer a web-based dashboard

8. Which hypervisor is mentioned as the most widely used in OpenStack due to its performance and integration?
   a) Xen
   b) Hyper-V
   c) VMware
   d) KVM

9. Horizon provides which of the following for OpenStack?
   a) Command-line interface
   b) Block storage service
   c) Web-based dashboard (GUI)
   d) Object storage service

10. The nova-scheduler service, which decides where an instance should be hosted, runs on the:
    a) Compute Node
    b) Network Node
    c) Controller Node
    d) Storage Node

B. Fill-in-the-blanks

1. The Controller Node hosts most of the shared services and ____ required for orchestrating the cloud.
2. ____ is the identity service in OpenStack, providing authentication and authorization.
3. The Glance service manages ____ ____ and snapshots of virtual machines.
4. The Neutron ____ agent handles dynamic IP address allocation for instances.
5. The ____ Node is the physical or virtual server where actual instances (virtual machines) are hosted.
6. The nova-compute service interacts with a ____, which could be KVM, QEMU, Xen, or VMware.
7. ____ is the web-based dashboard hosted on the Controller Node.
8. The Neutron ____ agent provides layer 3 (routing) functionality.
9. Open vSwitch or ____ ____ is often deployed on the Network Node to connect tenant networks.
10. The __________ is the backbone of an OpenStack cloud.

C. True/False questions

1. The Compute Node hosts the Keystone identity service.
2. Glance is responsible for managing network routing between instances.
3. The Nova scheduler runs on each Compute Node to decide instance placement.
4. Horizon provides a command-line interface for interacting with OpenStack.
5. The Network Node houses essential Neutron agents like the L3 agent and DHCP agent.
6. The nova-compute service is responsible for spawning, managing, and terminating VM instances on a physical host.
7. KVM is a message queueing system used in OpenStack.
8. Compute Nodes report their resource usage and availability to the Controller Node.
9. The Neutron metadata agent enables instances to retrieve instance-specific metadata.
10. Cinder API and scheduler services primarily run on the Compute Nodes.

D. Short Answer type questions.

1. What are the three major types of nodes in a standard OpenStack deployment as described in the chapter?
2. Which service on the Controller Node is responsible for authentication and authorization of all other OpenStack components?

3. What is the primary function of the Glance service in an OpenStack environment?
4. Name two key Neutron agents that typically run on the Network Node and describe their primary responsibilities.
5. Which software component on the Compute Node interacts directly with the hypervisor to launch and manage VMs?
6. What is the purpose of Horizon in an OpenStack deployment?
7. Name two common message queueing systems mentioned that are deployed on the Controller Node for inter-service communication.
8. What is the role of the Nova scheduler, and on which node does it typically operate?
9. How do Compute Nodes typically access the virtual machine images they need to launch instances?
10. Briefly explain the importance of configuring NTP (e.g., via Chrony) in an OpenStack deployment.

Answer Key

A. Multiple Choice Questions

1.d, 2.b, 3.c, 4.c, 5.c, 6.c, 7.b, 8.d, 9.c, 10.c

B. Fill-in-the-blanks

1. databases, 2. Keystone, 3. disk images, 4. DHCP, 5. Compute, 6. Hypervisor, 7. Horizon, 8. L3 Linux, 9. Bridge, 10. Compute Node

C. True/False questions

1. False, 2. False, 3. False, 4. False, 5. True, 6. True, 7. False, 8. True, 9. True, 10. False

Unit-3.
EMPHASIZING CLOUD COMPUTING SERVICES

Cloud computing is considered as a new technology that enables individuals to access and utilize computational resources such as storage, software, and processing on the web, without installing them on their own devices. Rather than purchasing and supporting costly hardware or software, individuals can merely rent what they require from cloud providers such as Google, Microsoft, or Amazon. Cloud services are employed daily in programs like Google Drive, Netflix, and Zoom. Whether you're saving a file on the internet, streaming a video, or taking an online course, cloud computing is quietly at work. It speeds up computing, makes it more agile, and more accessible to individuals and businesses alike.

In this unit, students will explore the world of cloud computing, which enables access to shared computing resources over the internet. Session 1 starts with comprehension of the primary concepts and classifications of cloud computing services. Through this, the students will gain knowledge on how cloud services promote flexibility, scalability, and economic benefits for customers and companies. Theoretical sessions encompass cloud deployment models, Public, Private, Community, and Hybrid clouds and hands-on activities such as the discussion of cloud types and mastering the creation and sharing of documents using cloud platforms like Google Drive and OneDrive.

In Session 2, learners will explore cloud infrastructure and architecture more deeply. They will study virtualization and the functioning of virtual machines in cloud environments. The course covers topics such as containers (e.g., Docker), serverless computing, and cloud data centers to present students with the overall view of the backend setup of cloud services. Practical activities will involve the creation and operation of virtual machines on platforms like Google Cloud, virtual machine versus container comparison, and live collaboration through document sharing and co-editing.

Cloud storage and databases are covered in Session 3. The various forms of cloud storage (object, block, and file storage) and how data is synchronized and managed in the cloud will be comprehended by the students. The use and relevance of cloud-based databases will be researched by the students. In the practical aspect, students will create cloud storage accounts adhering to best practices and conduct projects using Google Workspace, Microsoft OneDrive, or Teams. The session concludes with the study of actual case studies from sites like Netflix, Amazon, and Zoom to learn about how cloud storage and services operate in large-scale systems.

Chapter-7

Cloud Computing Services

Ravi started an online business but soon realized his single computer couldn't handle customer orders and data. His friend suggested moving to the cloud. Ravi learned that cloud infrastructure is like the foundation of a building—it provides servers, storage, and networks. The architecture decides how all these parts work together to deliver services smoothly.

With cloud infrastructure and architecture, Ravi's business could easily scale up during sales, stay secure, and serve customers faster. He understood that just like a strong building needs a good design, a reliable online service needs strong cloud infrastructure and smart architecture.

Cloud computing is a technology that allows users to access, retrieve and store data, apps, and services on the internet rather than using only their local devices. It provides flexibility, remote access, and secure storage, all without having to spend certain amount on expensive hardware or complicated installations. Be it collaborating on Google Docs, taking online classes through Zoom, or streaming movies on Netflix, cloud computing is an essential part of our everyday digital existence. It makes individuals and organizations smarter, quicker, and able to work from anywhere globally.

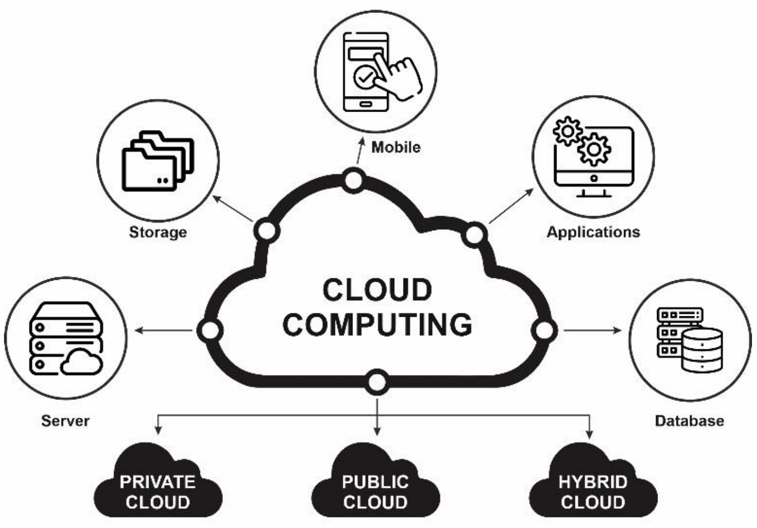


Fig 7.1 Services provided by cloud computing

Cloud Computing refers to accessing the internet to stock, process, and manage data rather than using your computer or local server. The data is stocked on distant servers, which belong to companies known as cloud providers, like Amazon, Microsoft, and Google. These corporations charge you according to how greatly you have utilized their services.

7.1. Revision of Cloud Computing Services

7.1.1 Types of Cloud Computing

The majority of cloud computing offerings are classified under five general groupings:

- Software as a service (SaaS)
- Platform as a service (PaaS)
- Infrastructure as a service (IaaS)
- Anything as a service (XaaS)
- Function as a Service (FaaS)

These are occasionally called the cloud computing heap because they are built on top of one another. Knowing what they are and how they are different makes it unambiguous to accomplish your goals. These abstraction layers can also be observed as a layered construction where services of a higher layer can be composed of services of the fundamental layer, i.e, SaaS can provide Infrastructure.

1. Software as a Service (SaaS)

Software-as-a-Service, commonly abbreviated as SaaS, is the delivery of software application over the web. In traditional software, the user must buy a software program, install it on his/her computer, and update it every so often. With SaaS, the user accesses the application via the web browser. There is nothing to install and no long download process. All the technical work of maintenance, updates, security and everything else is all done by the service provider.

One of the most common examples of SaaS is Google Docs. To use it, you don't need to download or install any software onto your computer. You simply open your Internet browser, log in to your Google account, and start typing in your document. All of your work is saved online, and Google takes care of making sure the software is automatically updated and functioning as intended. This enables the user to only think about using the service. The service provider handles the rest.

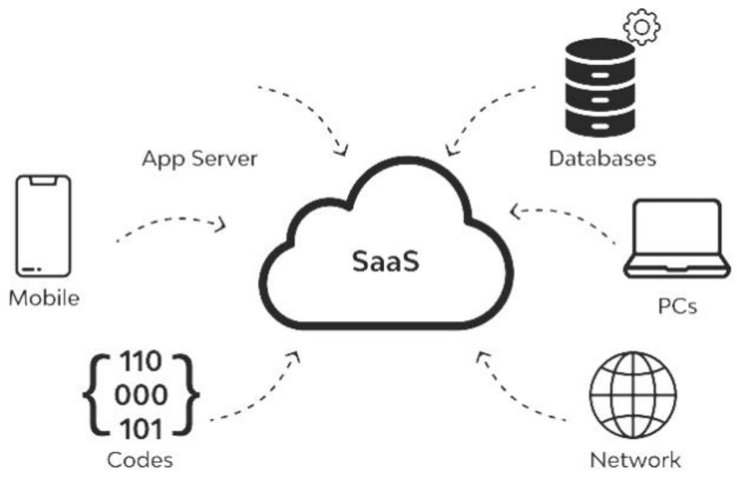


Fig 7.2. SaaS

Advantages of SaaS: -

- Affordable: Users only pay for what they use, meaning it is inexpensive.
- Saves Time: It runs directly from the web browser (doesn't require installation).
- Accessible: Users can access anywhere, anytime with internet connectivity.
- Automatic Updates: The service provider automatically updates the offering meaning users get the latest versions.
- Easily Scalable: Users can easily increase or decrease needed services.

The companies that offer Software as a service include Cloud9 Analytics, Salesforce.com, Cloud Switch, Cloud Tran, Big Commerce, and Microsoft Office 365.

Disadvantages of SaaS: -

- Reduced Adaptability: SaaS applications will have far less flexibility than traditional software-based solutions, because they require users to use the vendor's platform.
- Dependency on the Internet: As SaaS runs on the cloud, a reliable connection to the Internet is needed for functionality. Poor connectivity can negatively impact usage.
- Security Issues: Data will be stored on the vendor's server, posing some risk in terms of breaches or unauthorized access.

2. Platform as a Service (PaaS)

Platform as a Service is a cloud service that provides developers with a complete environment to build and run applications on the web without worrying about setting up their hardware or software. With PaaS, the service provider looks after the servers, storage, and operating systems, while the developer is focused only on writing and maintaining the app. Much like renting a ready-to-go venue for the annual day function at your school—there's no staging or arranging of lights, only the event. PaaS provides time, energy, and costs by providing everything behind the scenes and leaving the developer concerned only with the application and how or why it works.

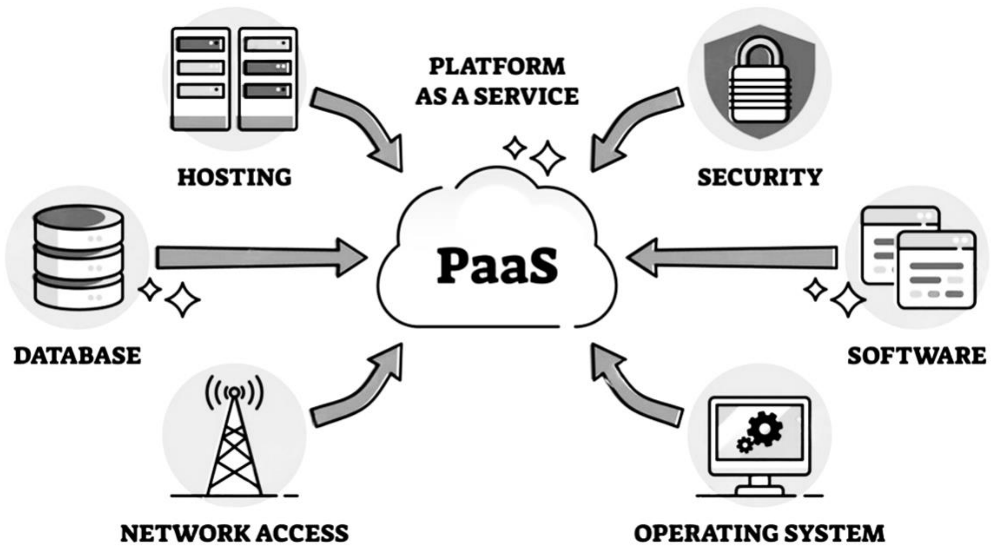


Fig 7.3 PaaS

Advantages of PaaS:

- Simple & Accessible: Equipped IT and infrastructure that can be accessed from anywhere that has a web browser.

- Cost Effective: Only paying for what is used, avoiding the upfront cost of hardware or software purchases and ongoing costs of maintenance.
- Encompasses the overall application lifecycle: It covers the full app development process from design, development, testing, deployment, operation and ultimately, maintenance.
- More productive: More higher-level application programming with less complexity when compared with on-site hardware, resulting in quicker app development - a better app.

The different companies offering Platform as a service are AWS, Elastic Beanstalk, Salesforce, Windows Azure, Google App Engine, and IBM smart cloud.

Disadvantages of PaaS:

- Restricted Control: Users have limited control over the underlying infrastructure, since it is completely managed by the provider (and customizability is limited).
- Trusting the Vendor: The availability and performance of applications is dependent upon the PaaS vendor, and any outage on the provider's end could affect you.
- Limited Flexibility: There are some workloads, service types, and/or applications that may not be supported by PaaS that can limit its overall usefulness for some organizations.

3. Infrastructure as a Service (IaaS)

Infrastructure-as-a-Service (IaaS) is a model of computing in which companies get IT infrastructure (i.e., servers, storage, networking) from a cloud vendor instead of buying and maintaining them in-house. It is like outsourcing your computer hardware. The vendor provides the underlying resource, as a provisioned resource (like a virtual machine), storage (disk), and internet connection, and you use their resource to host your applications and services.

In this model, you only have to pay for what you use, hourly, weekly, or monthly, which is a cost-effective way to use the IT resource you need. For instance, if you want to launch a website, you do not need to buy and maintain an expensive server. You can rent space or services on a cloud provider's server and run your website or application in their infrastructure.

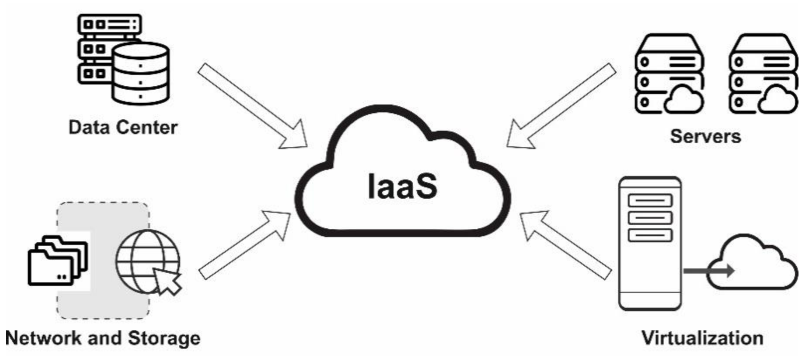


Fig 7.4 Infrastructure-as-a-Service (IaaS)

Advantages of IaaS:

- Cost Effective: No expensive startup costs. Customers only pay for what they use and they normally pay on an hourly, weekly, or monthly basis.
- Website Hosting: Hosting websites on IaaS often means enhanced flexibility and is often less expensive than traditional web hosting.

- Better Security: Cloud providers generally outperform many of the security measures in in-place with local systems.
- No Maintenance: Since the provider manages the data center, hardware, and updates, users benefit by minimizing maintenance.

The different companies offering Infrastructure as a service include Amazon web services, Bluestack and IBM.

Disadvantages of IaaS:

- Minimal control: users cannot customize and have limited influence in the management of the infrastructure that the cloud provider provides.
- Security issues: users have the responsibility of securing their data and applications which can be iffy at times.
- Unavailability: there may be areas in which cloud services may not operate due to government restrictions or regulations.

4. Anything as a Service (XaaS)

This is also referred to as Everything-as-a-Service (XaaS). Most cloud service providers today not only deliver SaaS, PaaS, or IaaS, they bundle a combination of these together with many other services. This means that almost any IT service can now be consumed via the cloud, as a service.

Advantages of XaaS:-

- Scalability: XaaS can be easily expanded or reduced based on the changing needs of an organization.
- Flexibility: It can provide numerous services, including storage, networking, and software, that can be adjusted based on requirements.
- Cost-effectiveness: XaaS is often cheaper than traditional systems, as users pay for only the services they consume.

Disadvantages of XaaS:-

- Provider Dependence: Users are dependent on the XaaS provider for consistent and reliable service, which is risky if there are any outages or failures.
- Limited Flexibility: There may be workloads or applications that aren't supported, which limits its utility for some organizations.
- Integration Issues: Integrating the XaaS solution with existing systems and data may be difficult, limiting the effectiveness of the solutions.

5. Function as a Service (FaaS)

Function as a Service (FaaS) is a cloud platform that allows you to run short blocks of code functions without any need to manage servers at all. You just write your code, upload it and it only executes when something happens, say the click of a button or the upload of a file. FaaS it is an event based so the code only executes when a certain event occurs. You don't have a server to run in the background so it starts up when you need it and shuts down when the job is done. The term "serverless" also comes from this concept. Consider an online image application where images are resized as someone uploads a photo for example. In FaaS, you build simple functions to resize the image. The Function executes only when an image is uploaded and you pay for the execution.

Advantages of FaaS:-

- Scalability: cloud vendors automatically adjust the number of resources used based on the number of requests.
- Cost: you will only be charged by how many times your function runs.
- Simplicity: you can upload small independent functions rather than having to manage a whole server or application.
- No Server Management: you only focus on code the provider manages the server.
- Languages: functions can be written in more than one programming language.
- Less Control: users have less control of the system since the cloud vendor is completely responsible for servers.

The different entities offering Function as a Service include Amazon Web Services, Oracle - Fn, Apache OpenWhisk - IBM, OpenFaaS,

Disadvantages of FaaS:-

- Startup Lag: Since FaaS only executes when an event is invoked, the first request could take a little longer because the function needs to start.
- Little Customization: Since the cloud provider manages the servers and infrastructure, users can't apply many customizations.
- Security Issues: Users are responsible to secure their data and apps, and maintaining that security can be complicated.
- Not Designed for Heavy Load: FaaS may not handle very heavy traffic or large amounts of requests efficiently.

Did you know?
Among the three main cloud computing models — IaaS, PaaS, and SaaS model gives users the most control over the computing resources, allowing them to manage virtual machines, storage, and networks just like managing their own data center — but without having to own any physical hardware!

7.2 Models for cloud deployment

A cloud deployment model essentially decides where your deployment's infrastructure is located and who retains and operates that structure. It also determines the nature and purpose of the cloud.

The first place that any company wanting to adopt cloud services should go is to learn about the deployment replicas available. Once these are known, a better decision can be made as to what paths the business should take. Each model will has its pros and cons in governance, scalability, cost, flexibility, security and management

Types of Deployment Models in Cloud Computing

The cloud deployment model categorizes the nature and purpose of the cloud environment grounded on ownership, size, and access, and the nature of the cloud itself. A cloud deployment determines where the servers that you are using are located, and who owns them is determined by a cloud deployment model. It indicates how your cloud environment will be, what you can modify, and whether you will be provided with services or will have to develop everything manually. Cloud deployment types also establish relationships between your users and infrastructure. Various kinds of cloud computing deployment models are explained below.

a) Public Cloud
b) Private Cloud
c) Hybrid Cloud
d) Community Cloud
e) Multi-Cloud

a) Public Cloud

The public cloud allows anyone to acquire systems and services. The public cloud can be less protected since it is accessible to all. The public cloud is a cloud that offers cloud infrastructure services across the internet to the public or large industry segments. The infrastructure within this cloud model is held by the organization that offers the cloud services, rather than by the customer. It is a form of cloud hosting that provides customers and users simple access to systems and services. This type of cloud computing is a perfect demonstration of cloud hosting, where service providers offer services to multiple clients. Under this arrangement, storage backup and retrieval services are provided at no cost, as a subscription, or on a per-user basis. Like Google App Engine, etc.

Advantages of the Public Cloud Model

- Minimal Investment: Since it is a pay-as-per-use service, there isn't much initial cost, and hence it is ideal for companies that need to access resources immediately.
- No arrangement cost: All the infrastructure is completely supported by the cloud vendors, so no setup of any hardware is necessary.
- No Infrastructure Management is needed: Utilizing the public cloud doesn't require infrastructure administration.
- No maintenance: The service provider performs the maintenance task.
- Dynamic extensibility: For meeting your company's requirements, on-demand resources are available.

Disadvantages of the Public Cloud Model

- Less protected: Public cloud is less secure since capitals are public, so no assurance of high-level security.
- Less customization: Many public accesses it, so it can't be customized as per individual needs.

b) Private Cloud

The private cloud belongs to only one organization and can either be hosted by the organization or a trusted external supplier; the cloud environment is not shared with anyone else. The organization retains full control over their data and resources. This model is a great fit for organizations that need the utmost privacy and security such as banks or government organizations. While the private cloud has significantly more control and customization and fits the need, it may be much more costly to maintain and administer than public clouds.

Advantages of the Private Cloud Model

- Improved Control: You are the exclusive holder of the property. You acquire full control over service integration, IT operations, guidelines, and user behavior.
- Data Security and Privacy: It's appropriate for holding commercial information to which only sanctioned personnel have access. Through segregation of resources in the same infrastructure, enhanced admission and security can be obtained.

- Provisions Legacy Systems: This model is set to support legacy systems that cannot tap in the public cloud.
- Customization: In contrast to a public cloud deployment, a private cloud will enable an organization to customize its resolution to suit its unique requirements.

Disadvantages of the Private Cloud Model

- Less scalable: Private clouds scale inside a range because there is a lower amount of clients.
- Expensive: Private clouds are more expensive as they provide personalized facilities.

c) Hybrid Cloud

A hybrid cloud is a combination of both public and private cloud functions. An organization can access the public cloud for non-sensitive workloads and utilize a private cloud for critical workloads because the two types of clouds are integrated and share data and applications. This model offers the flexibility to interchange workloads from one cloud to another, depending on potential cost savings. It also allows the organization to keep a high level of protection around its most important data. Many large organizations use hybrid clouds primarily due to their flexibility and adaptability.

Advantages of the Hybrid Cloud Model

- Flexibility and control: Extra flexible Companies can create custom-made solutions according to their specific needs.
- Cost-efficient: Since public clouds offer scalability, you'll pay only for extra capacity when you need it.
- Security: Since information is efficiently segregated, opportunities for data capture by intruders are much less.

Disadvantages of the Hybrid Cloud Model

- Tough to manage: Hybrid clouds are tough to manage because it is a mix of both public and private cloud. Therefore, it is complicated.
- Deliberate data transmission: Data transmission in the hybrid cloud is done through the public cloud, so there is dormancy.

d) Community Cloud

A community cloud is a cloud that is shared between a group of organizations that all have similar organizational requirements such as data security or compliance with specific legal or regulatory mandates that require organizations to work together to build or use a cloud. For any organizations that have similar needs, such a community cloud is less expensive than a private cloud, but cost is shared amongst members. The community cloud offers lower costs to those entities using it as found in sectors such as healthcare, education, or government, wherein similar types of organizations sometimes need to cooperatively share resources. It provides high privacy while sharing resources.

Advantages of the Community Cloud Model

- Cost-Efficient: It is cost-efficient since the cloud is shared among various administrations or communities.

- Secure: Community cloud offers greater safety.
- Shared resources: It enables you to share resources, infrastructure, etc, among various administrations.
- Association and data sharing: It is appropriate for association as well as data sharing.

Disadvantages of the Community Cloud Model

- Limited Scalability: Community cloud is not very extensible in nature because there are numerous organizations using the same resources as per their collective needs.
- Rigid to customize: Since the data and resources are combined between various organizations based on their common interests, if one organization desires some alterations as per their requirements, they cannot since it will affect other organizations.

e) Multi-Cloud

We are referring to having more than one cloud provider simultaneously under this model, as the title suggests.

This is comparable to the hybrid cloud deployment strategy, which brings public and private clouds together. As an alternative of integrating public and private clouds, multi-cloud integrates a variety of public clouds. Even though public cloud vendors offer various instruments to make their services more reliable, accidents can still happen. It is very unlikely that two different clouds would experience an occurrence at the same time. Consequently, multi-cloud deployment enhances the high accessibility of your services even further.

Advantages of the Multi-Cloud Model

You can combine the best of each cloud provider's services to meet the needs of your apps, workloads, and business by selecting different cloud providers.

- Lower Latency: To lower latency and enhance the knowledge of users, you can select cloud locations and regions that are nearer to your customers.
- Service high accessibility: Unusually, two separate clouds partake an event at the same time. Therefore, the multi-cloud deployment enhances the high availability of your facilities.

Disadvantages of the Multi-Cloud Model

- Ambiguous: That several clouds combined make the system complicated, and there can be blockages.
- Security issue: Because of the complicated design, loopholes might exist through which a hacker can gain access, hence making the data unsafe.

7.3 Applications of Cloud Computing

1. Google Drive / OneDrive / Dropbox – Cloud Storage

Cloud storage solutions such as Google Drive and Microsoft OneDrive enable one to save their files online rather than just on their device. That implies that you would be able to use your documents, presentation, and photos from any device—be it a computer at school, your phone, or a laptop at home. These websites enable easier organization of files into folders, sharing them with others, and even online editing of documents. They also back up your files, so you don't lose them in case your device gets destroyed. For students, this is particularly useful because your projects, notes, and assignments are always within reach.

Did you know?

Many popular apps you use every day, like Netflix, Google Drive, and Spotify, run on cloud computing! This lets them stream videos, store your files, and play music instantly without needing powerful hardware on your device.

2. Google Docs / Microsoft Office 365 – Online Collaboration

Online collaboration tools like Google Docs and Microsoft Office 365 allow numerous people to work on the same document at the same time, even if they are in different places. For example, if you're doing a group project, each member can open the same file, edit it together in real time, and leave comments. This saves time and helps avoid confusion over different file versions. These resources automatically save your work as you write, avoiding the loss of your work. They also offer features such as adding images, tables, and even citations, which make your work presentable. These resources are particularly helpful in schools and group study rooms.

3. Gmail / Outlook – Cloud-Based Email Services

Email services such as Gmail and Outlook are examples of cloud-based communication systems. When you send and receive mail, all your messages and attachments are kept on cloud servers. That way, no matter what device you use to log in to your mail, you will still have access to your contacts, messages, and folders. These services also provide convenient features such as spam filtering, calendar integration, and file sharing. For students, email is a crucial means of communicating with instructors, getting school news, or sending in assignments. Because all information is saved on the web, there's no fear of losing vital messages or having to save them manually.

Fig 7.5 Cloud-Based Email services

4. Zoom / Google Meet / Microsoft Teams – Video Conferencing

Video conferencing tools such as Google Meet, Zoom and Microsoft Teams have attained huge popularity, particularly for online lectures and virtual meetings. These web-based applications enable users to host or join virtual sessions with audio, video, and screen sharing capabilities. Teachers in a classroom can give live lectures, share study documents, and even record the session for future reference. Students can pose questions, provide presentations, or participate in breakout rooms for group discussions. Chat, file sharing, and whiteboards are also supported by these platforms. Since they are cloud-based, we don't require powerful hardware—just an internet connection and a login.

Fig 7.6 Video conferencing

5. YouTube / Netflix / Spotify – Streaming Services

Streaming services such as YouTube, Netflix, and Spotify enable you to view video and listen to music straight from the cloud without needing to download hefty files. These services upload their content to powerful cloud servers and stream it to your device on demand. For students, this translates to being able to view educational videos, documentaries, and tutorials in an instant. Without having to think about storage capacity, you can view a huge library of content whenever you want. These services also make recommendations tailored to your usage. The cloud provides instant delivery, automatic updates, and accessibility across devices.

Fig 7.7 Streaming services

6. Social Media (Instagram, Facebook, WhatsApp) – Cloud-Based Communication

Social networking sites like Facebook, Instagram, and WhatsApp are based on cloud computing to host and store the content of the users, like messages, posts, images, and videos. Each time you upload a picture, send a message, or post an update, that information gets stored in the cloud so you can connect it from any device. Even when you change your phone, all your information remains accessible when you log back in. These applications take advantage of cloud services to render content speedily, store gigantic amounts of customer information, and maintain security. Social media, for students, is an access method that lets them keep up with buddies, connect to academic communities, and exchange course material.

Fig 7.8 Social Media applications

7. Online Shopping & Banking (Amazon, Flipkart, UPI apps) – Cloud Transactions

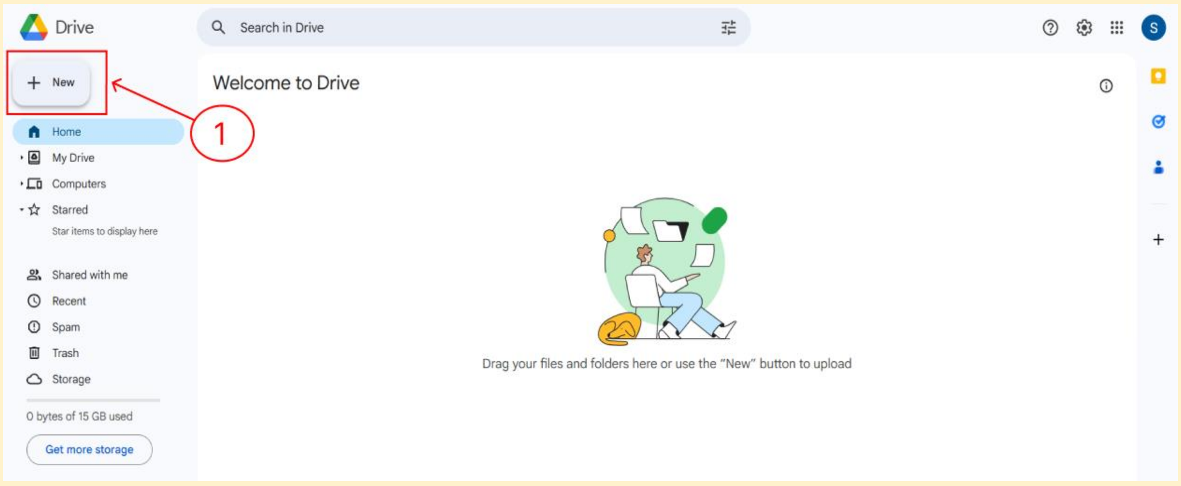
Online services like Amazon, Flipkart, and UPI-based apps (Google Pay, PhonePe, etc.) use cloud computing to manage transactions, store customer data, and provide secure access. When you make a purchase or transfer money, the cloud handles everything from product selection to payment processing. These platforms work 24/7 and use strong security measures like encryption to keep your data safe. For students, online applications facilitate online purchases of books, payment of fees, or even refilling phones without having to use money. They also give instant reminders and maintain a record of all transactions, which can be accessed at any time from any machine.

Practical Activity 7.1:

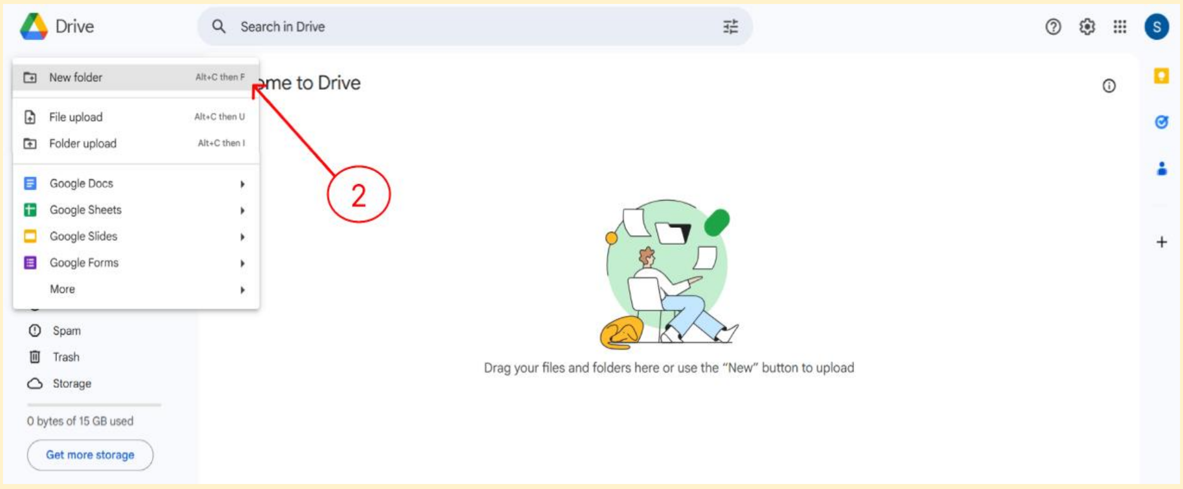
1. Creation and sharing files using Google Drive

Material required: Google drive, Netlify hosting

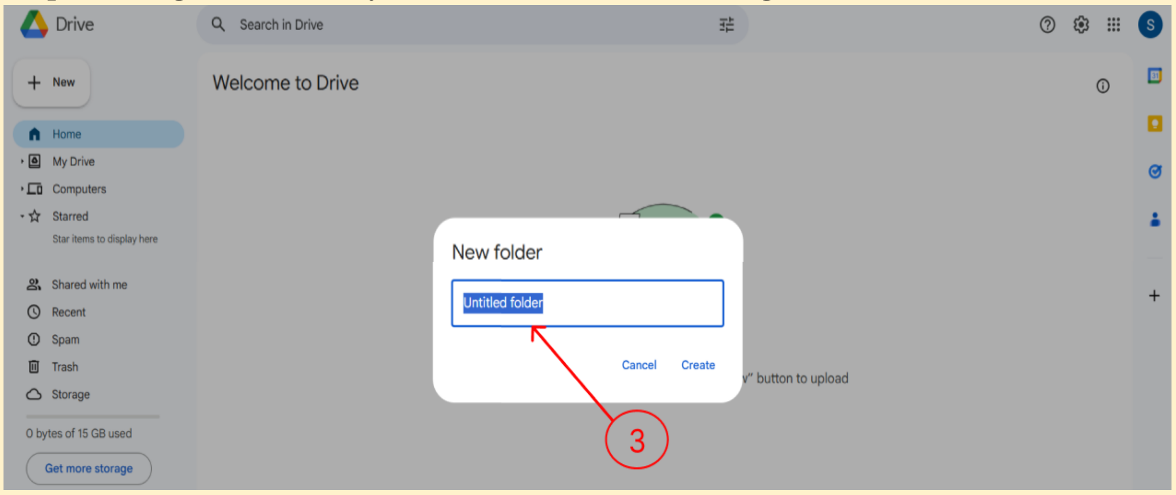
Step 1: Go to a browser and open Google Drive. Create/open an account using your email id. Now click on New option to create a new document as shown in figure below



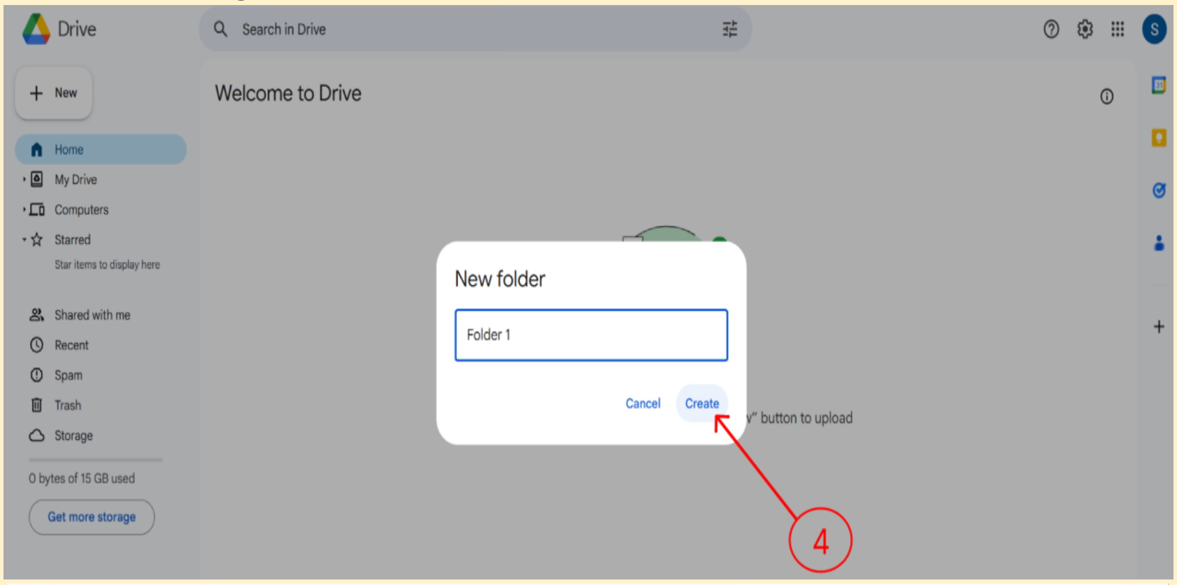
Step 2: In the dropbox click on New folder to create a new folder, as shown in figure below

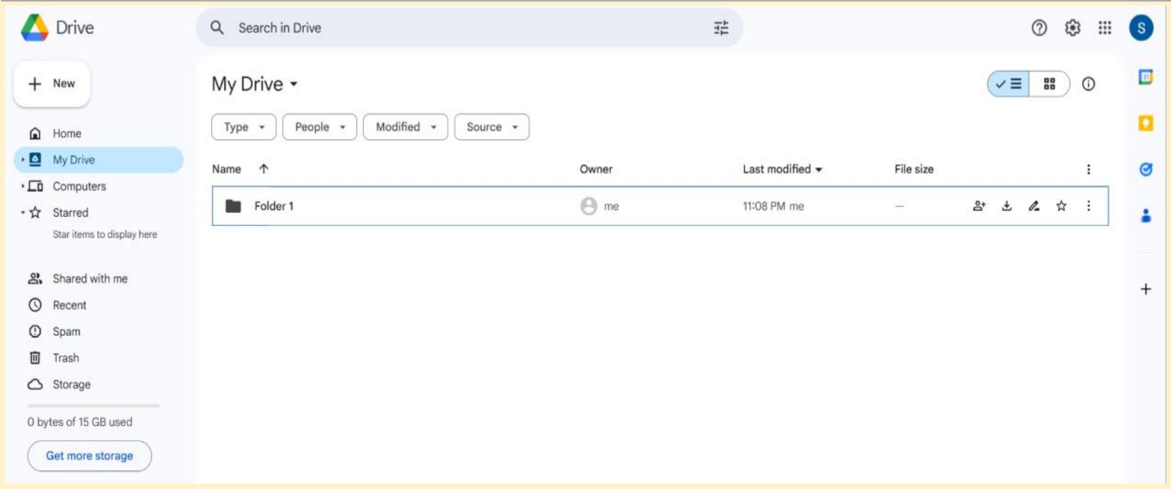


Step 3: Now give a name to your folder, as shown in the figure below

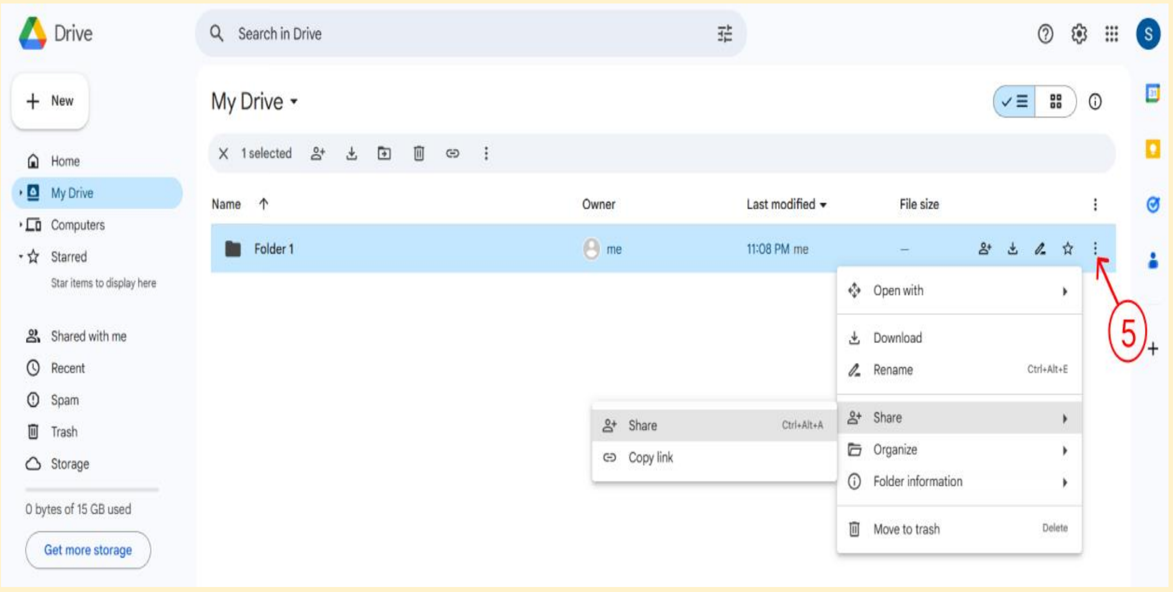


Step 4: Click on Create option. Now you can observe a new folder created on your screen, as shown in the figure below



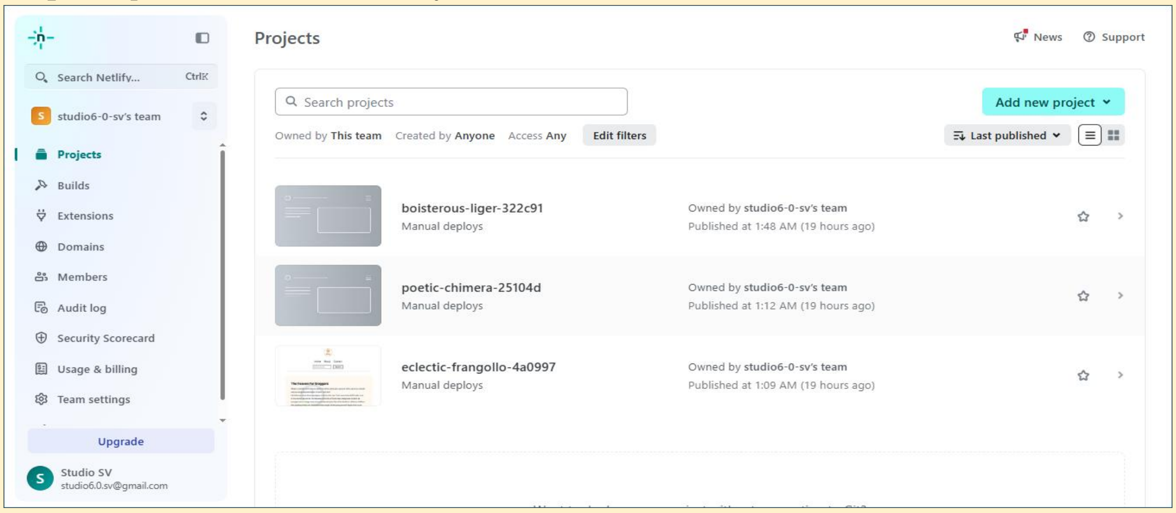


Step 5: By clicking on the three dots on the right of the folder created, you can click on the share option to share the folder with anyone or copy the link to your folder for further transmission of the folder



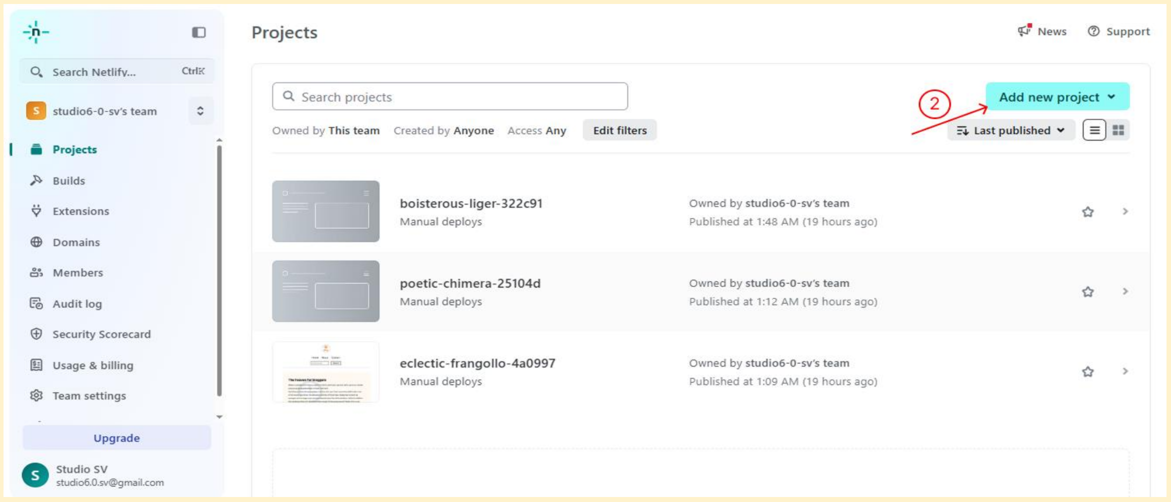
2. Host a simple website using cloud storage

Step 1: Open dashboard of Netlify.



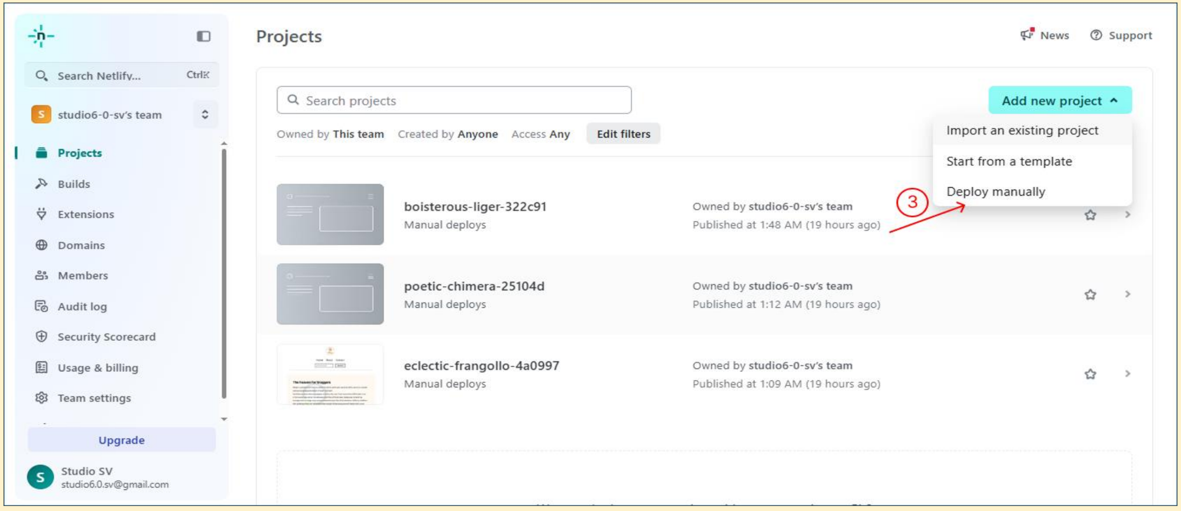
A user would navigate to the Netlify dashboard in their web browser and log in with their credentials.

Step 2: Click on the "Add new project" button.



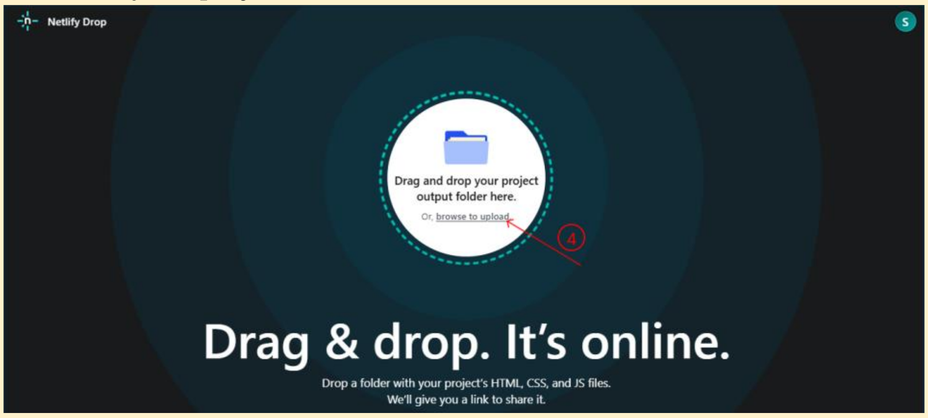
Select on this button to select your project. It involves selecting a template, connecting a Git repository, or manually uploading files.

Step 3: Select Deploy manually.



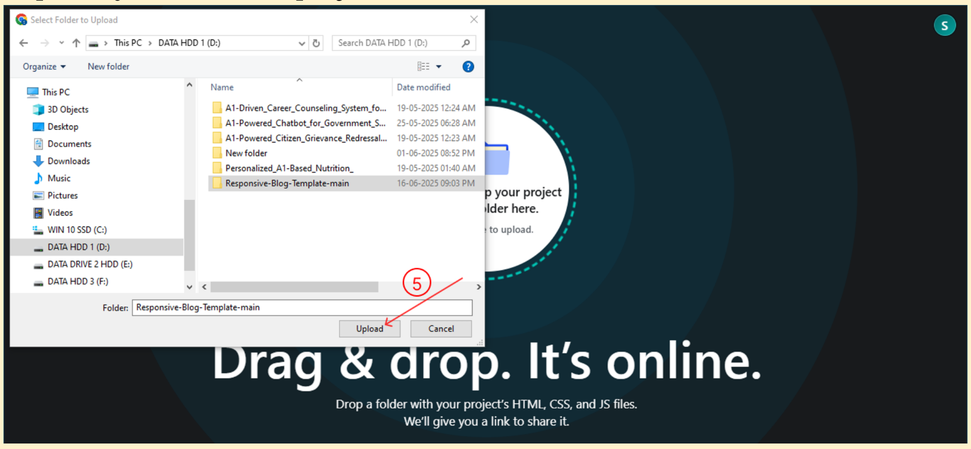
From the "Add new project" dropdown menu, click on the "Deploy manually" option. Here you can drag and drop your project files or browse your local system to upload them for deployment.

Step 4: Browse for your project folder.


Netlify Drop
S
Drag and drop your project output folder here.
Or, browse to upload.
4
Drag & drop. It's online.
Drop a folder with your project's HTML, CSS, and JS files.
We'll give you a link to share it.

On the Netlify Drop page, click on the "browse to upload link" so that you can select a project through your PC.

Step 5: Upload your selected project folder.

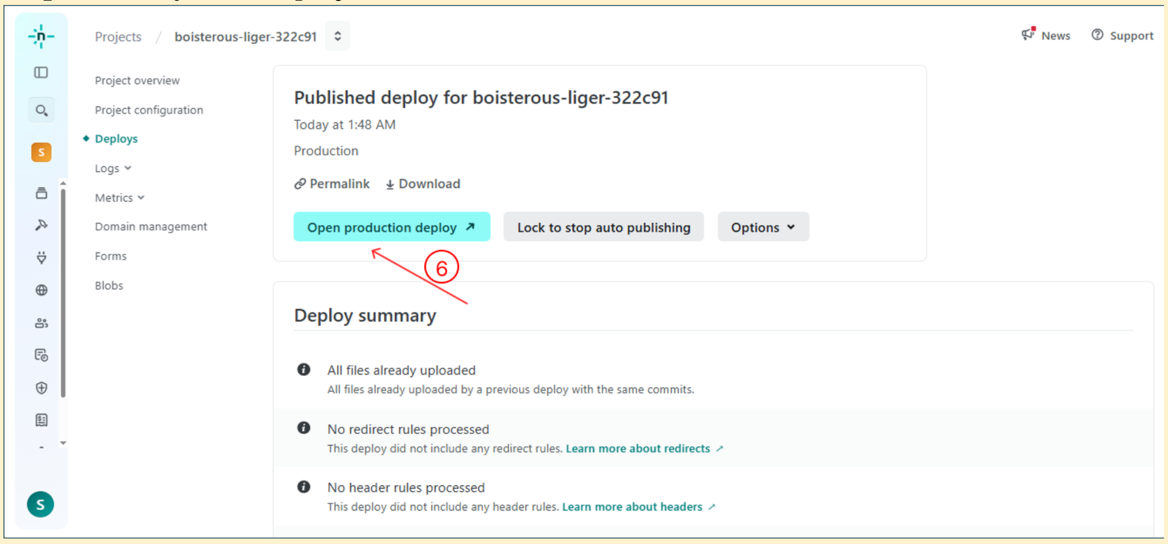


Name	Date modified
A1-Driven_Career_Counseling_System_fo...	19-05-2025 12:24 AM
A1-Powered_Chatbot_for_Government_S...	25-05-2025 06:28 AM
A1-Powered_Citizen_Grievance_Redressal...	19-05-2025 12:23 AM
New folder	01-06-2025 08:52 PM
Personalized_A1-Based_Nutrition_	19-05-2025 01:40 AM
Responsive-Blog-Template-main	16-06-2025 09:03 PM

Folder: Responsive-Blog-Template-main

In the "Select Folder to Upload" window, ensure your project's output folder. Click on the Upload button to initiate the upload process.

Step 6: View your live project.



Published deploy for boisterous-liger-322c91
Today at 1:48 AM
Production

Deploy summary

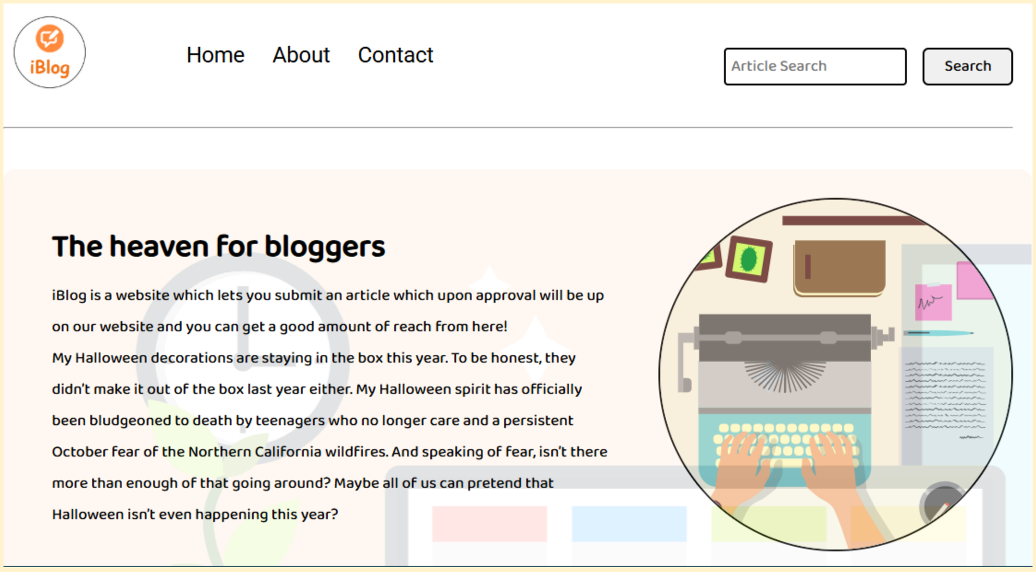
All files already uploaded
All files already uploaded by a previous deploy with the same commits.

No redirect rules processed
This deploy did not include any redirect rules. Learn more about redirects

No header rules processed
This deploy did not include any header rules. Learn more about headers

On the project overview page, click the Open production deploy button to view your newly published website.

Step 7: Verify your deployed website.



The heaven for bloggers

iBlog is a website which lets you submit an article which upon approval will be up on our website and you can get a good amount of reach from here!
My Halloween decorations are staying in the box this year. To be honest, they didn't make it out of the box last year either. My Halloween spirit has officially been bludgeoned to death by teenagers who no longer care and a persistent October fear of the Northern California wildfires. And speaking of fear, isn't there more than enough of that going around? Maybe all of us can pretend that Halloween isn't even happening this year?

After clicking "Open production deploy" (or navigating to your project's URL), confirm that your website is successfully live and appears as expected in your browser.

ASSESSMENT

A. Multiple Choice Questions

1. Cloud computing allows users to access:
   a) Only local files
   b) Data, apps, and services over the internet
   c) Only hardware
   d) Only software installed on their computer

2. Which of the following is not a benefit of cloud computing mentioned in the text?
   a) Flexibility
   b) Remote access
   c) High cost of hardware
   d) Secure storage

3. Companies that provide cloud services are called:
   a) Users
   b) Developers
   c) Cloud providers
   d) Hardware vendors

4. SaaS stands for:
   a) Software as a Service
   b) Storage as a Service
   c) System as a Service
   d) Security as a Service

5. In SaaS, who is accountable for software maintenance and updates?
   a) The user
   b) The computer manufacturer
   c) The internet service provider
   d) The SaaS provider

6. Which among the following is an advantage of SaaS?
   a) High customization
   b) Internet connectivity dependence
   c) Automatic updates
   d) High security risks

7. PaaS provides developers with:
   a) Only hardware
   b) Only software
   c) Everything they need to develop and publish apps online
   d) Just a web browser

8. In PaaS, who is responsible for servers, storage, and operating systems?
   a) The developer
   b) The user
   c) The PaaS provider
   d) The network administrator

9. Which of the following is an advantage of PaaS?
   a) Limited infrastructure control
   b) High cost
   c) Easy and user-friendly
   d) Restricted flexibility

10. IaaS stands for:
    a) Internet as a Service
    b) Infrastructure a Service
    c) Information as a Service
    d) Integration as a Service

B. Fill-in-the-blanks

1. Cloud computing is a technology that allows users to access and store data, apps, and services on the __________ instead of using only their local devices.
2. Cloud Computing refers to accessing the __________ to store, process, and manage data rather than via your own computer or local server.
3. The data is stored on distant servers, which belong to companies known as __________ like Amazon, Google, and Microsoft.
4. The majority of cloud computing offerings are classified under __________ general categories.
5. SaaS stands for __________.
6. XaaS is also referred to as __________.
7. SaaS means using software over the __________ instead of installing it on your computer.
8. SaaS is typically provided on a __________ model.
9. SaaS is also referred to as web-based software or __________ software.
10. Most SaaS solutions are cloud-based, hence reliant on a stable __________ to work.

C. True/False questions

1. Cloud computing requires users to purchase expensive hardware.
2. Cloud computing provides flexibility and remote access to data.

3. Cloud providers charge users based on their usage of services.
4. SaaS involves installing software directly on your computer.
5. In SaaS, the provider handles software updates.
6. SaaS applications are highly customizable.
7. PaaS provides developers with infrastructure like servers and storage.
8. Developers using PaaS have full control over the back-end infrastructure.
9. PaaS can help make application development more efficient.
10. IaaS involves leasing IT infrastructure.

D. Short Questions and Answers

1. What is Cloud computing?
2. Name three cloud providers.
3. List the five general categories of cloud computing offerings.
4. What is SaaS? Give two advantages and disadvantages of SaaS.
5. What is PaaS? Give two advantages and disadvantages of PaaS.
6. Explain Multi-cloud model. Also provide its benefits and drawbacks.
7. What is XaaS? Give two advantages of XaaS.
8. Explain the public cloud model.
9. Explain the hybrid cloud model.
10. Write down the applications of cloud computing.

Answer Key

A. Multiple Choice Questions

1.b, 2.c, 3.c, 4.a, 5.d, 6.c, 7.c, 8.c, 9.c, 10.b

B. Fill-in-the-blanks

1. internet, 2. Internet, 3. cloud providers, 4. Three, 5. Software as a Service, 6. Anything as a Service, 7. Internet, 8. Subscription, 9. on-demand, 10. internet connection

C. True/False questions

1. False, 2. True, 3. True, 4. False, 5. True, 6. False, 7. True, 8. False, 9. True, 10. True

Chapter-8

Cloud Infrastructure and Architecture

Ravi started an online business but soon realized his single computer couldn't handle customer orders and data. His friend suggested moving to the cloud. Ravi learned that cloud infrastructure is like the foundation of a building—it provides servers, storage, and networks. The architecture decides how all these parts work together to deliver services smoothly.

With cloud infrastructure and architecture, Ravi's business could easily scale up during sales, stay secure, and serve customers faster. He understood that just like a strong building needs a good design, a reliable online service needs strong cloud infrastructure and smart architecture.

8.1 Virtualization

Imagine working on an important assignment using a personal computer at home. Suddenly, the device starts running out of storage space, and installing new software seems complicated and expensive. Buying extra hardware would take both time and money. At this moment, there is a simpler solution, just connect to the internet and use services already available online. With only a few clicks, extra storage can be added, powerful applications can be accessed, and even large amounts of data can be processed without any additional installation.

Behind this convenience are companies known as cloud providers. Well known examples include Amazon Web Services (AWS), Microsoft Azure etc. These providers maintain huge data centres with powerful servers located across the world. Instead of each individual or organization purchasing costly equipment, cloud providers allow them to "rent" only the amount of storage, computing power, or services they actually need. This model has made technology more affordable, flexible, and accessible to anyone with an internet connection.

This idea has revolutionized the utilization of computing resources. Before virtualization, each operating system or application required its physical machine. This resulted in hardware wastage, as numerous machines were typically underutilized. With virtualization, we are able to utilize several operating systems and applications on a single physical device, making more efficient use of resources and saving costs.

Did you Know?
Virtualization allows one physical computer to run **multiple virtual machines**, each acting like a separate computer — meaning a single server can do the work of many, saving space, energy, and costs!

The key concept behind virtualization is to isolate the software from the hardware. It makes a layer between the software using it and the physical system. That is, the programs are no longer required to know what real hardware they are on. Rather, they communicate with a virtualized version of the hardware.

8.2 Virtual Machine

A Virtual Machine is a computer that exists within a physical computer and executes software. It acts exactly like a regular computer, with its own operating system, memory, disk space, and applications. But in fact, it is only part of a big system, which shares the physical resources such as CPU, RAM, and disk space with other virtual machines.

All virtual machines are provisioned with a special program known as a hypervisor. The hypervisor allocates physical resources to these virtual machines and controls them. There are two primary forms of hypervisors: Type 1 hypervisors execute immediately on the hardware (bare-metal hypervisors), and Type 2 hypervisors execute on a host operating system (hosted hypervisors).

VMs are isolated from each other, so if one VM crashes or breaks, it doesn't touch the others. This separation makes them extremely useful for testing new software, having multiple operating systems running, or creating secure environments.

8.2.1 Working of Virtualization

The most important technology that supports virtualization is the hypervisor, which is the middleman between the hardware and the virtual machines. When you launch a virtual machine, the hypervisor assigns a chunk of the computer's memory, CPU, and other resources to it. As far as the VM is concerned, it has a complete system of its own, and it can execute software as a physical machine.

For instance, you may be on a Windows 11 computer, and using virtualization, you may launch a virtual machine with Linux. You may run Linux applications within this VM, test them, and even access the web, all without making any impact on your Windows system. The VM occupies a window, just as any other application.

Hypervisors are intelligent enough to optimize the needs of all the virtual machines that are running. If a VM requires additional memory or CPU, and the physical infrastructure can accommodate it, the hypervisor can dole it out as required. This adaptability is one reason why virtualization has gained such widespread usage.

Advantages of Virtualization

Virtualization has numerous benefits, particularly in large organizations, data centers, and cloud computing. The largest advantage is the efficiency of resources. With several virtual machines operating on a single physical system, businesses can minimize the number of physical servers required. This reduces hardware, electricity, and maintenance costs.

Another significant advantage is flexibility. Virtual machines are easily created, changed, and deleted. When a developer has to test an application under a different operating system, they can install a virtual machine in minutes. In the same way, companies can adapt to changing requirements by scaling out or in their virtual infrastructure with ease.

Security is also a major advantage. Since virtual machines are virtualized and do not directly interact with each other, executing dangerous or experimental software within a VM insulates the central system. If things do not work as they should, only the VM gets impacted, and it can be restored or removed without damaging the host.

Virtualization aids disaster recovery as well. As VMs are files stored on a disk, they can be easily backed up and copied. If a physical server goes down, its virtual machines can be easily transferred to another server and restarted, minimizing downtime.

8.2.2. Types of Virtualization

There are various forms of virtualization, all with their own applications. The most widely used form is server virtualization, where several virtual servers can run on one physical server. It is commonly employed in the data center to optimize the use of servers and lower expenses. Shown in Figure 8.1.

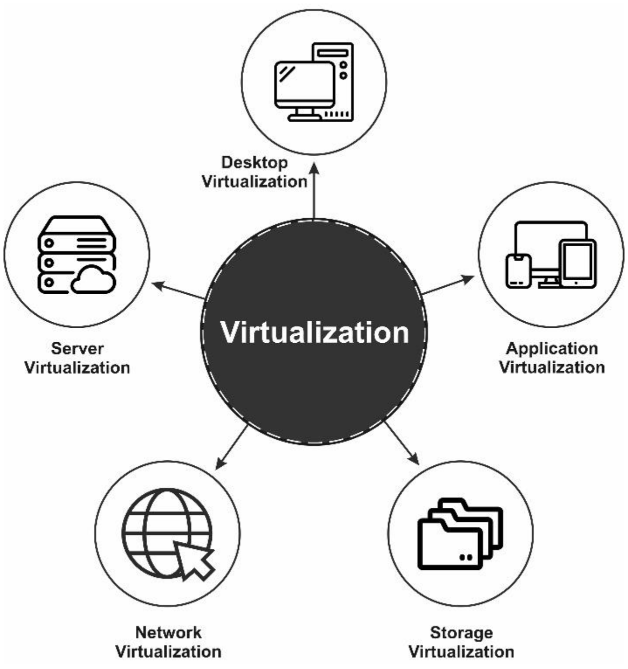


Fig 8.1 Types of virtualization

- Application Virtualization: This technology permits users to access applications remotely, as if they were installed directly on their computer. The application runs on a server, but users interact with it over the internet. This is helpful for using different versions of the same software or accessing applications from various devices. Examples include hosted and packaged applications.
- Network Virtualization: This involves creating virtual networks within a physical network. It allows multiple virtual networks to operate independently on the same hardware. Virtual components like switches, routers, firewalls, and VPNs can be added easily, increasing network flexibility and management efficiency.

- Desktop Virtualization: This technology permits the creation of virtual desktop environments, which users can access from devices like laptops or tablets. It provides users with flexibility and simplifies tasks like software updates and improves portability.
- Storage Virtualization: This technology combines storage space from multiple physical storage devices into an only, unified storage system. It simplifies storage management and ensures continuous operation, even if there are hardware changes or failures.
- Server Virtualization: This technology divides one physical server into several independent virtual servers. Each virtual server can execute its own operating system and applications, maximizing the use of server resources.

8.2.3. Real-Life Applications of Virtualization

Virtualization is employed in numerous fields these days. In colleges and schools, virtual machines are employed to provide laboratories wherein students can experiment with programming and software in a safe environment. Virtualization in companies decreases IT expenses and enhances productivity by executing numerous applications on a few machines.

Cloud services such as Amazon Web Services (AWS), and Microsoft Azure leverage virtualization to provide computing capability to consumers globally. When you utilize these services to host a website or retain data, you tend to utilize virtual machines located in a remote data center.

Even in the home, virtualization can be employed. Technology fans employ virtual machines to experiment with new operating systems or to operate older software that could be incompatible with their systems at present. It's also a wonderful tool for cybersecurity training, where users can practice attacks within a safe and controlled virtual setting.

8.3 Docker

Docker is a tool that allows developers to package an application and all the things it needs (like libraries, code, and system tools) into a container. A container is like a small, lightweight box that holds everything an application needs to run. This makes the app run the same way no matter where it is – on a personal computer, on a company server, or in the cloud.

In cloud computing, applications are often run on many different servers and environments. Without Docker, apps may face issues like “it works on my computer but not on the cloud.” Docker solves this problem by:

- Making applications portable (run anywhere – local system, server, or cloud).
- Saving resources (containers are lighter than virtual machines).
- Allowing faster deployment (apps can be launched in seconds).
- Helping in scaling applications easily in the cloud (adding more containers when needed).

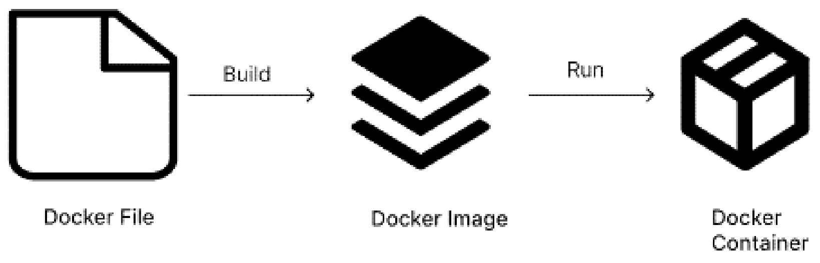


Fig 8.2 Docker

Although it might seem like a virtual machine (VM), Docker containers are far more efficient. A virtual machine requires a complete operating system for every instance, which eats up a lot of disk space and memory. Docker containers, on the other hand, share the host operating system's kernel, so they start quicker, use less resources, and can be run in huge quantities on one system. This effectiveness has made Docker a darling among the DevOps and cloud computing communities.

Its portability is another feature that makes Docker so powerful. Developers can create their applications within Docker containers and then transport them across environments without giving a thought to compatibility. A program created on a laptop may be shipped to a cloud server without any modification. This is vital in today's software development when applications are deployed across multiple servers as well as geographical locations.

Docker constructs containers using images. A Docker image is analogous to a recipe or blueprint of what a container should be like. It contains the code of the application, required files, and other instructions on how to construct the container. Images are built from a Dockerfile, which is a plain text file containing commands like installing packages of software packages or copying files. After the image is constructed, it can be published to online repositories such as Docker Hub, which is a repository where developers worldwide can upload and download container images.

Docker is also extensively used within continuous integration and constant deployment (CI/CD) pipelines. Within a CI/CD environment, whenever a developer commits any change to the code, it gets tested and deployed automatically via tools with Docker integration. Thus, new functionality and bug fixes can be delivered to users more rapidly and securely. For instance, an app for a mobile device can be updated every day with new functionality, and everything is automated with Docker containers for testing and delivery.

Security is also enhanced using Docker containers. Because every container is separate from others and the host machine, an issue in one container does not impact others. That is particularly significant when several applications or users use one server. In addition, Docker offers functionality such as image scanning and signed containers, which ensure that software executed in containers is secure and reliable.

In cloud computing, Docker has occurred as a basic building block. Cloud providers such as Amazon Web Services (AWS) and Google Cloud Platform (GCP) provide complete support for Docker containers. Applications in the cloud typically employ containers to scale up and down with user demand. As an example, in an online test, the number of students using the platform may suddenly rise. With Docker containers, the application will be able to scale rapidly by adding more containers to carry the load and then scaling them down once traffic reduces.

Lastly, Docker is not only a professional tool—it's also extremely applicable to students and teachers. Most computer science and IT programs now incorporate Docker in their syllabi. It promotes experiential learning and trial. Even schools and universities are implementing Docker to develop standardized environments for teaching programming, database management, and web development. This provides the same learning environment to all students, whether they use a different computer or not.

8.4 Containers

In cloud computing, containers are now one of the most important and influential instruments to deploy and manage software. Since cloud computing is concerned with delivering resources such as storage, processing capacity, and applications over the internet, it also expects these resources to be used efficiently and reliably. This is where containers fit in—they provide a sensible solution to one of the greatest challenges in cloud space: executing software reliably on many systems and at enormous scale.

Containers in cloud computing are similar to portable, self-contained packages that have everything an application requires to execute—its code, configuration files, libraries, and system tools. This is why they are best suited for cloud environments where the same application can be deployed on various servers, across various geographies, or even for thousands of users simultaneously. Because containers are light and don't need a full-fledged operating system to be packaged within them (as with virtual machines), they boot quickly, take up less memory, and are more efficient in their use of computing resources. This enables cloud service providers such as Google Cloud Platform, Amazon Web Services (AWS) and Microsoft Azure to host and operate applications at a significantly cheaper price while providing high performance and reliability.

One of the key benefits of containers in cloud computing is scalability. Suppose a website for online shopping experiences an unexpected influx of traffic during a promotion. If that website is hosted within containers on a cloud infrastructure, the cloud company is able to automatically spawn additional containers in real time to cope with the increased traffic. This is referred to as auto-scaling, and it prevents the users from having any delays or downtime. When the traffic returns to normal, the additional containers can be shut down, meaning that the organization will not be charged for unused resources. This type of flexibility and cost-effectiveness is only achievable with the use of containers and cloud technology.

Another huge benefit of containers in the cloud is portability. Because containers encapsulate everything an application requires, they can very easily be transported from one place to another. This allows a developer to create an application on their laptop, test it on a staging server, and publish it on a cloud platform and never once touch the application's underlying code. The container makes sure that the application will act the same way in all environments. This is crucial in cloud computing, where applications might need to be published on more than one server, more than one location, or even more than one cloud provider.

Additionally, containers enhance cloud application security and isolation. Every container executes in its isolated environment, distinct from the host system as well as other containers. This implies that if one container is hacked or fails, it does not impact other containers or the system. This form of isolation is particularly helpful in multi-tenant clouds, where several users or administrations are sharing the same cloud infrastructure. Containers ensure that every user's application runs safely and independently, without interrupting others.

Today's cloud-native technologies, such as Kubernetes have further amplified the capability of containers. Kubernetes is a container orchestrator that assists cloud companies and developers in running hundreds or thousands of containers with ease. It performs tasks such as launching, halting, scaling, updating, and monitoring containers. Without Kubernetes and similar tools, it would be very challenging and cumbersome to do things manually in the cloud. With it, organizations can develop dependable and scalable cloud applications much more effortlessly.

In short, containers are now the building blocks of contemporary cloud computing. They give a light, portable, and consistent means of running applications on different cloud environments. Their capacity to isolate, scale, and interact with orchestration tools such as Kubernetes makes them the deployment method of choice for cloud-native applications. It is crucial for students and future professionals to learn how containers work in the cloud in order to keep pace with the changing technology and software development world.

8.5 Resource Pooling

In cloud computing, resource pooling is one of the underlying features that makes it possible to deliver services efficiently and flexibly to many users. It is a term that is used to describe how cloud providers handle their physical and virtual computing resources, servers, storage, processing capacity, memory, and network bandwidth, by pooling them into a single pool. These shared resources are further dynamically allocated and reallocated based on the changing requirements of users on demand. Rather than each user being allocated a fixed number of hardware, users can access a common pool of resources, which are assigned automatically whenever they are needed. This sharing is accomplished in a secure and efficient manner, with no delay or interference to users.

One of the fundamental principles of resource pooling is that of multi-tenancy, whereby numerous users (also referred to as tenants) access a common physical infrastructure but have their data and applications separated from one another. Each user gets the impression that they are working on an individual system but actually are working on virtual segments of the combined infrastructure. It is facilitated by virtualization, which is technology that enables one physical machine to be partitioned into multiple independent virtual machines (VMs). Every VM behaves like a distinct computer with its own operating system and applications. Virtualization provides robust isolation between users and simplifies the management and relocation of workloads in the cloud space.

Cost-effectiveness is one of the key benefits of resource pooling. As several users use the same infrastructure and hardware, the cost is lower overall. Cloud providers do not have to purchase and maintain individual machines for each user, and users only pay for what they use. This approach is also very efficient, as resources are not idle. When a given amount of computing power or storage is no longer needed by one user, such resources are released back into the pool and can be utilized by another person instantly. This is what assists in the minimization of waste and the optimized use of available resources.

Scalability is another essential feature. Resource pooling makes it simple and rapid for cloud services to scale down or scale up based on the requirement. For instance, when a site suddenly gets more traffic, the cloud environment can provision additional processing capabilities and memory from the pool to ensure the site remains in operation. The resources can be removed once traffic levels drop again and returned to the pool. This is what provides cloud computing with such great potential for companies that have unpredictable or dynamic loads.

A second advantage is location independence. The users need not be aware of where the processing and storage of their data is taking place. Regardless of whether the resources are in Mumbai, Singapore, or New York, the users deal with the same virtual interface. The location of the servers from the user's perspective is kept concealed, and the cloud provider has control over how the resources are allocated to various data centers. This not only makes cloud computing convenient but also world-accessible, facilitating operations across a world divided by geography without any extra effort on the user's part.

For better clarity, imagine a real-world example like an electricity grid. When you turn on a light bulb at home, you are tapping into an inclusive network. You don't own the power plant; you just consume electricity when needed and pay for its usage. Likewise, in cloud computing, users don't own the storage devices or the servers. They just utilize computing resources from a shared pool, and they are charged accordingly for what they consume. This approach makes the system more flexible, manageable, and cost-effective.

In addition, resource pooling makes centralized maintenance and control simpler. It is simpler to update, patch, and secure shared infrastructure. If a single physical machine malfunctions, the system can immediately reallocate workloads to a second machine in the pool without interrupting service. This provides high availability and fault tolerance, which are essential characteristics for any enterprise dependent on cloud services.

In summary, resource pooling is a fundamental principle that drives cloud computing's flexibility and efficiency. By consolidating and sharing computing resources of various users through virtualization, cloud providers are able to provide dependable, affordable, and scalable services. This not only aids providers by maximizing their infrastructure but also aids users by providing them with a hassle-free, secure, and flexible computing experience. Without resource pooling, the cloud would lack most of its power, versatility, and worth.

8.6 Cloud Data Centres

A cloud data centre is an expert centre in which huge quantities of computer servers, networking equipment, storage units, and software platforms are hosted to deliver cloud computing services. Data centres make up the physical infrastructure of cloud computing. Although cloud services seem like virtual platforms or software applications running on the internet to users, in the background, all this processing takes place through physical assets within enormous, secure structures known as data centres.

Cloud data centers are typically constructed by large cloud service vendors like Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), IBM Cloud, and so on. These organizations own and manage enormous data center networks globally, guaranteeing that cloud services are accessed at all times, in several regions. The aim is to provide computing resources, like processing power, storage capacity, databases, and applications over the internet, without the users having to own or maintain physical hardware.

Every cloud data centre has thousands of servers stacked in racks. These servers are always up and processing requests of users worldwide. In addition to servers, a data centre also has cooling systems, power backup systems, fire suppression systems, and high-end networking devices to keep everything operational 24x7. High-speed internet connectivity and several layers of security—both physical and online are also integral components of a data centre.

A key characteristic of cloud data centres is virtualization. Rather than allocating entire physical computers to single users, cloud vendors utilize virtualization software to configure multiple virtual machines (VMs) or containers for each piece of hardware. This enables multiple users to use the same hardware, with their data and applications being kept isolated. Such virtual worlds are dynamically controlled by cloud orchestration software that allocates the workloads, balances traffic, and guarantees efficiency in resource usage.

Cloud data centres are also architected to be highly available and redundant. This implies that even if one server fails or one component of the system goes down, an alternate portion of the data centre will be able to take over instantly without impacting the experience of the user. In other cases, the same data is replicated in different data centres at different places (regions or zones) so that services will continue even if a large failure like power outage or natural disaster occurs. This type of design is also referred to as fault tolerance or disaster recovery.

Another significant feature of cloud data centres is scalability. These data centers have the ability to increase or decrease resources in accordance with user demand. For instance, when there are festive seasons or international events, the users visiting a particular website or application surge drastically. Cloud data centers have the ability to rapidly deploy additional servers and bandwidth to manage the huge surge. When the demand declines, they can again decrease the resources. This is one of the main reasons why businesses prefer cloud computing—it adapts automatically to changing needs.

From an economic and environmental perspective, contemporary cloud data centers are designed to be efficient in terms of energy. These tend to employ novel cooling mechanisms, clean energy sources like wind and solar, and smart power management mechanisms to minimize electricity consumption. This not only minimizes costs but also decreases the environmental footprint of mass computation.

To the users, the whereabouts and functioning of a data center are largely transparent. When you post a picture, watch a video online, or run online applications, your data is really processed in one or more data centers, which could be somewhere in the world. But cloud providers make the system so quick and consistent that users perceive that they are dealing with local software.

So, cloud data centers are the spine of the entire cloud ecosystem. Without them, the cloud would not be able to deliver the flexible, on-demand, and affordable services that have become essential in today's digital world.

8.7 Concept of Serverless Computing

In cloud computing, a new model called serverless computing has emerged, which changes the approach being followed in the building, deployment, and execution of applications. From the name itself, it can be inferred that serverless computing points towards the lack of servers; however, it is a very misleading name. The servers are indeed present and are performing their significant roles in the background, but the concept is that the developers do not need to fear about or monitor the servers. All the infrastructure-related operations like server configuration, provisioning, scaling, monitoring, and maintenance are all handled by the cloud service provider. This arrangement enables developers to concentrate only on writing the logic and code of the application, thereby making the development process simple and avoiding operational complexities.

Historically, when hosting an application online, the programmers would need to lease or buy a server, whether physical or virtual. They would then need to host the software and operating system necessary, set up the server to serve their application and keep it running continuously. On top of this, they would need to make intelligent predictions of how many users would use the application and allocate sufficient computational resources to handle the anticipated traffic. Should more users than expected use the application, the server would be swamped with the load and crash. Should fewer users than expected use the application, the provisioned resources would be wasted, incurring unnecessary expense. This was an ongoing issue with the tradeoff of cost, performance, and reliability.

Serverless computing solves these issues by offering a completely different model. Developers code functions small pieces of code that have the purpose of performing a single task and then deploy them to the cloud provider. This is generally referred to as Functions as a Service (FaaS). The infrastructure management issues are taken care of by the cloud provider. When something happens for e.g., a user presses a button, submits a form, or uploads a file the associated function is invoked automatically. It executes in a managed cloud environment, performs the single task, and then exists. Developers do not pre-book server time, deal with background processes, or load balancing because the platform dynamically and automatically solves all these issues.

One of the most important features of serverless computing is that it scales automatically as an inherent component of its architecture. What this means is that when there are a thousand users of the application simultaneously, the cloud provider can call thousand instances of the function—one instance per user—without delay. If there are fewer users, the instances scale down automatically and hence no resources are wasted. Serverless computing's scalability is one of the key reasons for its greater efficiency and its capability to respond immediately to changing loads of users.

To demonstrate this principle, a good analogy is taken from everyday life. Consider the server-based approach as having a restaurant. This entails leasing the building, purchasing kitchen equipment, employing staff, and maintaining the business no matter the customers. Serverless computing is analogous to a street food stall with a van that operates only when there is a request. It cooks, serves, and remains in stand-by until called once again. Costs incurred here are only for the used services without necessarily maintaining kitchens and staff continuously.

The Actual Working of Serverless Architecture Serverless computing consists of individual functions running in isolation and with their own compute and memory resources and invoked by some events. The events could be a user (e.g., a form submission), a system (e.g., file upload), or a schedule (e.g., a batch task that runs on a daily basis). The function is run only for the time required with the users paying only for the runtime and the resource utilization during the function execution. There is no cost, therefore, when the function is not running, making it very cost-effective.

These stateless serverless functions retain nothing from past runs. If there is an application that needs to store or retrieve data between events, it will have to do this through external storage or databases, which also are hosted by the cloud provider. While this statelessness makes the functions light and quick, it also means developers will need to ensure that they are mindful of how their app stores and loads data.

Some of the most common serverless platforms include

- AWS (Amazon Web Services)
- IBM Cloud Functions
- Microsoft Azure Functions
- Google Cloud Functions

Important Features of Serverless Computing

Serverless computing possesses certain strong attributes which are making it more and more popular:

- **No Server Management:** The developers don't need to worry about the infrastructure at all. Right from updates, maintenance, security, and availability, the cloud provider handles everything.
- **Event-Driven Execution:** Events such as user input, database updates, or scheduled tasks trigger methods to be called.
- **Automatic Scaling:** Scales automatically up and down based on the level of incoming requests.
- **Micro-billing:** Customers are charged for only the actual compute time used, in milliseconds.
- **Faster Development and Deployment:** Since developers do not need to worry about servers, they can simply write and deploy working code.

Illustrative Application Scenario: Serverless Implementation in Practice

Let's say a business is going to make an image-resizing application to run online. Instead of using a 24/7-running server, the developer writes a function that is called whenever a user submits a picture. The function reduces the picture to the desired size and stores it in a cloud storage bucket. The function ends. The developer doesn't concern himself with what is happening behind the scenes, how many users are submitting images or if the system can handle large loads. All this is managed by the cloud provider.

Advantages of Serverless Computing

1. Cost-Effectiveness: Pay only when your code executes. No charge for idle server time.
2. Simplified Infrastructure Management: The cloud providers manage provisioning, patching, and server scaling.
3. High Availability: Cloud environments guarantee that your application is always available using in-built redundancy mechanisms.
4. Rapid Innovation: Release new features rapidly, test code rapidly, and experiment with no risk.
5. Global Accessibility: Serverless apps may be deployed from various geographic locations around the world with virtually zero latency and fast response times for global users.

Restrictions and Challenges

Although serverless computing is very useful, it is not for every situation: Execution Time Constraints: All serverless platforms have limitations on the execution time of functions that make them inappropriate for lengthy operations. Cold Starts: A function that has not been called in a long time will take longer to start the first time. That can have a minor effect on response time. Complex Architectures: Large programs can mean splitting up functionality into lots of tiny functions, which can make the architecture complex. Constrained Power: The developers have only limited control over the environment where their functions run, and that could constrain some types of software. Conclusion In short, serverless computing is an evolutionary cloud computing paradigm that provides a very efficient, flexible, and cost-saving way of running applications. By removing server management, it enables developers to focus on the valuable aspects writing functional, scalable, and responsive applications. This paradigm is well-suited for startups, mobile apps, real-time apps, and businesses that need to scale rapidly without investing much in infrastructure. With its inherent constraints, serverless computing is becoming an integral part of cloud architecture today and one of the main drivers of digital innovation.

8.8 Difference between Virtual Machine and Containers

Feature	Virtual Machines (VMs)	Containers
Definition	A Virtual Machine is like a complete computer system inside another computer. It has its own virtual hardware, operating system, and apps.	A Container is similar to a small box that contains just the app and what it needs to run. It uses the host computer's operating system.
Startup Time	VMs take longer to start because they load a complete operating system every time, like starting a real computer.	Containers start very quickly, typically in seconds, because they don't need to load an entire OS.
Resource Usage	Virtual Machines utilizes a lot of memory and storage since each one runs its own operating system and apps.	Containers are lightweight and use less system resources because they share the same OS.

Operating System	Each VM can run a unlike OS (like Windows or Linux), even if the host is different. This allows more flexibility.	All containers share the same OS as the host.
Isolation	VMs provide strong isolation as everything is separated, with the OS and apps. It's like having separate computers.	Containers are remote at the application level. They don't have their own OS, so isolation is weaker but enough for most uses.
Use Case	VMs are useful when you want to run different operating systems or need robust security between apps.	Containers are ideal for running numerous versions of the same app or service quickly and easily.
Examples	Common VM software includes VMware, VirtualBox, and Hyper-V. They allow you to make and manage virtual computers.	Popular container tools comprise Docker and Kubernetes. They help developers package apps and run them anywhere.

Practical activity 8.1

1. Create and run a virtual machine on free-tier platform

Material required: On Google Cloud Free Tier, you can create and run a virtual machine by signing up, creating a project, enabling Compute Engine, selecting a free-tier eligible machine type, and connecting via SSH or RDP to use it like a remote computer in the cloud.

Step 1: Sign Up on Google Cloud Free Tier. First Visit- https://cloud.google.com/free, then click on "Get started for free". Sign in with your Google account, fill your details and verify yourself

Step 2: Now Go to Google Cloud Console, visit- https://console.cloud.google.com/, you will land in the Google Cloud Console dashboard.

Step 3: Create a New Project, at the top-left corner, click on the Project drop-down, click on "New Project", name your project (e.g., "My First VM") and then click Create

Step 4: Enable Compute Engine, in the search bar at the top, type "Compute Engine", click on Compute Engine then click VM instances, click "Enable" (this may take a few seconds)

Step 5: Create Your Virtual Machine, once Compute Engine is ready, click "Create Instance"

Set these basic settings:

1. Name: Give your VM a name (e.g., my-vm)
2. Region: Choose a Free Tier eligible region like us-west1
3. Machine type: Select e2-micro (this is free-tier eligible)
4. Boot disk: Choose an OS like Debian, Ubuntu, or Windows Server

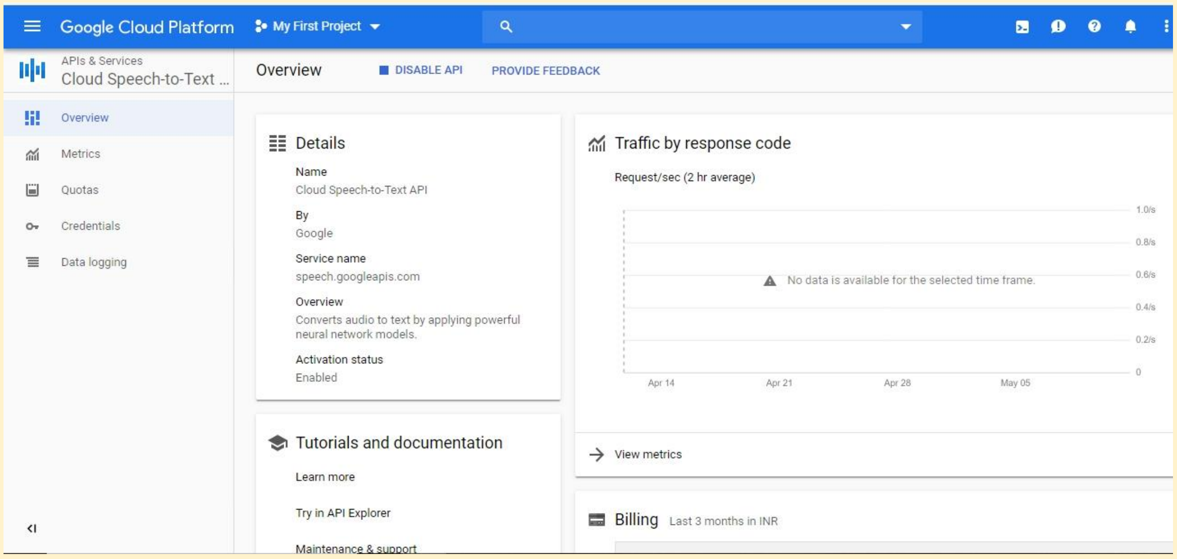
Now click "Create"

Step 6: Start and Use Your VM, once created, your VM will appear in the list

Click on "SSH" to open a terminal window to your VM

1. If you chose Linux, a black terminal will open
2. If you chose Windows, you can connect using RDP (Remote Desktop Protocol)

Your virtual machine is now running on the cloud, similar to a computer inside Google data center. You can use it to run programs, learn Linux commands, or practice coding.



Google Cloud Platform · My First Project

APIs & Services
Cloud Speech-to-Text ...

Overview · DISABLE API · PROVIDE FEEDBACK

Overview
Metrics
Quotas
Credentials
Data logging

Details

Name
Cloud Speech-to-Text API

By
Google

Service name
speech.googleapis.com

Overview
Converts audio to text by applying powerful neural network models.

Activation status
Enabled

Traffic by response code

Request/sec (2 hr average)

No data is available for the selected time frame.

View metrics

Tutorials and documentation

Learn more

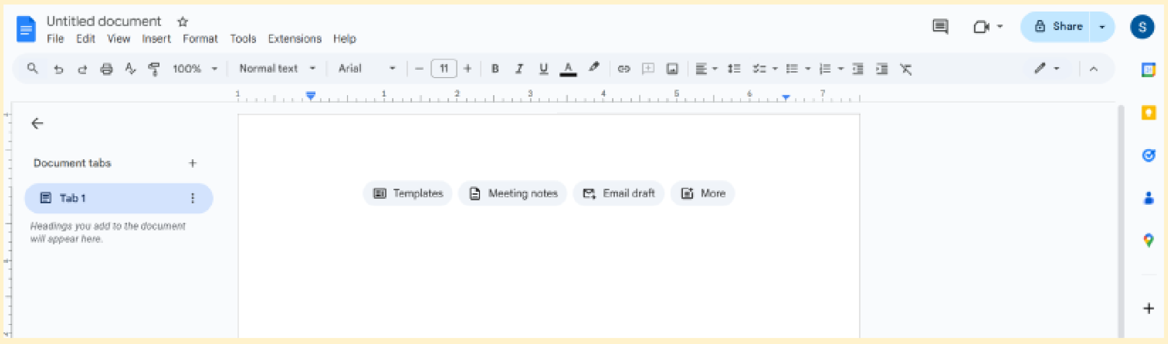
Try in API Explorer

Maintenance & support

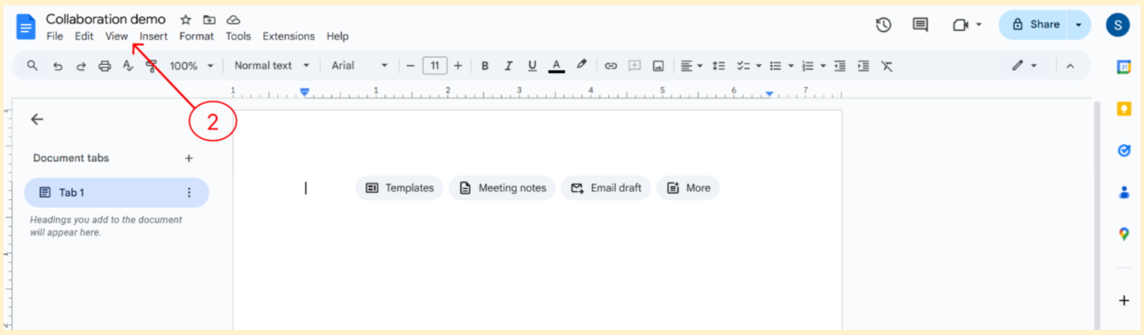
Billing Last 3 months in INR

2. Demonstration of collaboration of documents in real time

Step 1: Create a Document and access Google Docs. Click on "Blank" to begin a new document.

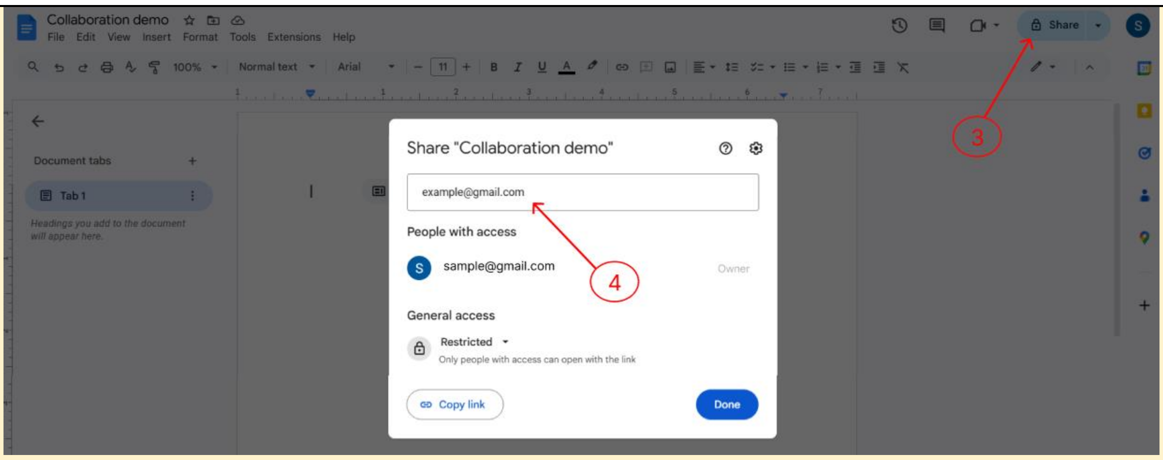

Untitled document
File Edit View Insert Format Tools Extensions Help
Share
100%
Normal text
Arial
11
Document tabs
Tab 1
Headings you add to the document will appear here.
Templates
Meeting notes
Email draft
More

Step 2: Give your new document a name (for example, "Collaboration Sample").

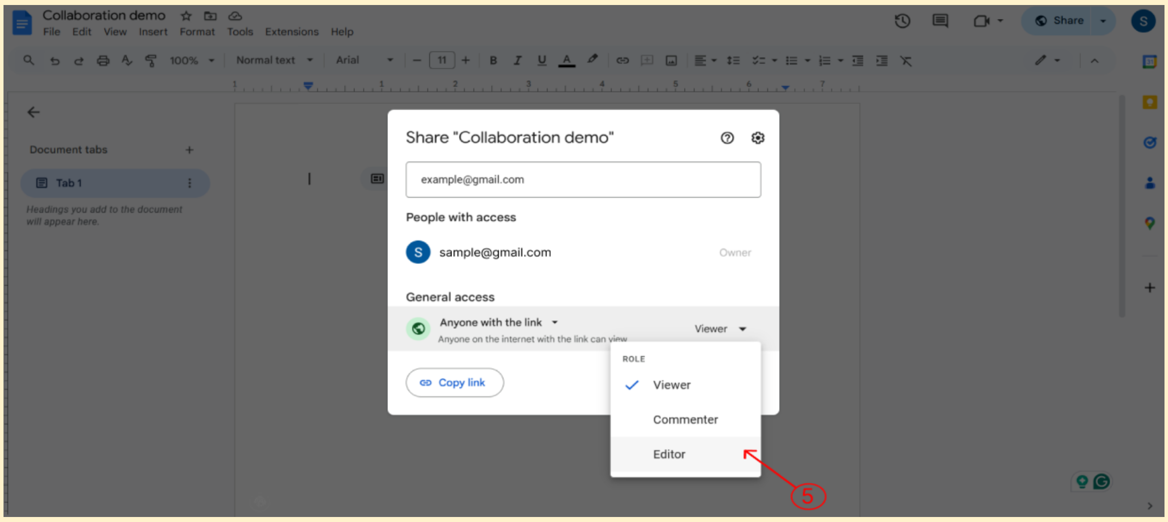


Step 3: Share the Document. Locate and then click on the "Share" button, in the top right corner of the page.

Type in the email addresses of your classmates or friends (they need to have a Google email) into the pop-up window.


Collaboration demo
File Edit View Insert Format Tools Extensions Help
Share
Document tabs
Tab 1
Headings you add to the document will appear here.
Share "Collaboration demo"
example@gmail.com
People with access
sample@gmail.com
Owner
General access
Restricted
Only people with access can open with the link
Copy link
Done
3
4

Step 4: Change their permission to Editor (so that they can have input). Click Send.



Step 5: Begin Working Together. All the users invited (the document is now shared) are able to access the document at the same time. One person will see a cursor that is colored different from other people, it will tell the other people who is drafting and what they are doing.

Everyone can add text, make changes to each other's work, add comments. By highlighting text and clicking "Insert → Comment" and be able to communicate in a chat box (not available with all versions) while in the document.

ASSESSMENT

A. Multiple Choice Questions

1. What is cloud computing?
   a) A type of local storage
   b) Accessing computing resources over the internet
   c) Using only hardware devices
   d) Offline software applications

2. Which among the following is a key characteristic of cloud computing?
   a) High initial investment in hardware
   b) Limited accessibility

c) On-demand access to resources
d) Single-device dependency

3. In the SaaS model, who manages the software?
a) The user
b) The device manufacturer
c) The cloud provider
d) The internet service provider

4. Which cloud service model provides hardware and software tools available over the internet?
a) SaaS
b) FaaS
c) PaaS
d) IaaS

5. What does IaaS involve?
a) Leasing software applications
b) Providing a platform for app development
c) Leasing IT infrastructure
d) Executing code in response to events

6. Which cloud deployment model offers services to the general-public over the internet?
a) Private cloud
b) Public cloud
c) Hybrid cloud
d) Community cloud

7. Which cloud deployment model is exclusive to a sole organization?
a) Public cloud
b) Private cloud
c) Hybrid cloud
d) Community cloud

8. What does a hybrid cloud combine?
a) Multiple public clouds
b) Public and private clouds
c) Multiple private clouds
d) Community and public clouds

9. Which of the following is a advantage of cloud storage?
a) Data is only accessible from one device
b) Reduced data security
c) Accessibility of files from any device
d) No file sharing capabilities

10. What is the purpose of cloud file syncing?
a) To keep files only on a local device
b) To share files via email attachments
c) To automatically update files across multiple devices
d) To prevent file sharing

B. Fill-in-the-blanks

1. Cloud computing aids users to access data and applications over the ___________.
2. ___________ is a cloud service model that delivers software applications over the internet.
3. PaaS provides developers with a ___________ to make and deploy applications.
4. IaaS allows businesses to lease IT ___________ such as servers and storage.
5. In a ___________ cloud, the cloud infrastructure is shared by numerous organizations with a common interest.
6. A ___________ cloud combines public and private cloud environments.
7. ___________ cloud storage permits access to files from any device with an internet connection.
8. ___________ enables many users to work on the same document simultaneously.
9. FaaS is also known as ___________ computing.
10. Within the public cloud model, the Cloud infrastructure is owned by the ___________.

C. True/False questions

1. Cloud computing requires users to purchase and maintain their own hardware.
2. SaaS allows users to have a high degree of control over the underlying infrastructure.
3. PaaS is designed to shorten the process of developing and deploying applications.
4. IaaS provides the most control over computing resources compared to other cloud service models.
5. A public cloud offers greater security as compared to a private cloud.
6. A private cloud is typically more expensive than a public cloud.
7. A hybrid cloud combines the features of public as well as community clouds.
8. Cloud storage enables users to access their files from any device.
9. Cloud syncing automatically updates files across all devices connected to the same account.
10. FaaS requires users to manage the underlying server infrastructure.

D. Short Answer Questions

1. What is cloud computing and how does it differ from traditional computing?
2. Explain the Software as a Service (SaaS) cloud service model also provide an example.
3. Describe the Platform as a Service (PaaS) and its benefits for developers.
4. What is Infrastructure as a Service (IaaS) also classify what type of control does it offer to users?
5. What are the key characteristics of public cloud?
6. How does private cloud differ from a public cloud in terms of access and security?

7. Explain the concept of a hybrid cloud and its advantages for organizations.
8. What is a community cloud and what are its primary use cases?
9. Describe the multi-cloud deployment model and its benefits related to availability.
10. How do cloud storage services alike Google Drive and Dropbox benefit users?

Answer Key

A. Multiple Choice Questions

1.b, 2.c, 3.c, 4.c, 5.c, 6.b, 7.b, 8.b, 9.c, 10.c

B. Fill-in-the-blanks

1. internet, 2. SaaS, 3. Platform, 4. Infrastructure, 5. Community, 6. Hybrid, 7. Cloud, 8. Online collaboration, 9. Serverless, 10. Cloud provider

C. True/False questions

1. False, 2. False, 3. True, 4. True, 5. False, 6. True, 7. False, 8. True, 9. True, 10. False

Chapter-9

Cloud Storage and Databases

Meena was tired of carrying USB drives to move her college project files between her laptop and the computer lab. One day, her teacher introduced her to cloud storage. She learned that by saving her work on the cloud, she could access it anytime from her laptop, phone, or even the lab computers—just by logging in with the internet. Later, when her project grew bigger, she needed to handle lots of data. That's when she discovered cloud databases. Instead of installing heavy software on her laptop, she could use a cloud database managed by providers like AWS or Google Cloud. The service stored her data securely, updated automatically, and was always available.

Meena realized that cloud storage keeps files safe and accessible, while cloud databases help organize and manage large amounts of information. From then on, she never worried about losing files or managing big data.

9.1 Cloud Storage

On one evening, a group of students was working through the preparations for their final presentation. Each student had been working on different sections of the project- documents, images, and data files. Rather than passing around multiple USB drives or wondering who had the last file of each version, the students decided to place everything in Google Drive. From their homes, all students could upload, update, and view the same files in real time. Even when one student could not access her laptop because it stopped working, the files were secure and all students could access the files from whatever device they chose without worry or concern.

This common scenario illustrates the true value of cloud storage. Today, in our digital world, protecting our information, ensuring that we can access it from anywhere, and making it expandable as training or organizational needs grow is imperative, not only from a student's perspective, but from a large organization point of view. Services such as Google Drive, Dropbox, Microsoft OneDrive, and Amazon S3 have changed the way individuals and companies store, access, and manage data, allowing them to store their critical information to ensure secure protection, sharing where needed, and accurate information that can be accessed reliably.

Cloud storage is like keeping your computer files on many faraway computers instead of just on your device, like your laptop's hard drive or your phone. These big computer systems are owned and run by companies that specialize in cloud services. They take care of keeping the data safe and making sure it's always there. Shown in the Figure 9.1.

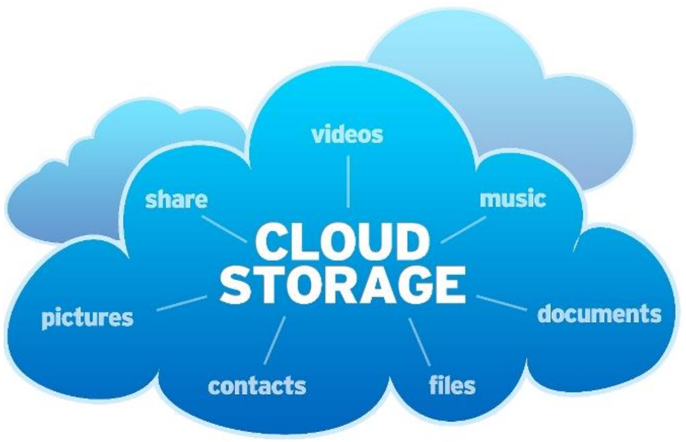


Fig 9.1 cloud storage

Imagine it as a digital locker that you can open from anywhere, anytime. Instead of your files being stuck on one computer or in one place, they're sent over the internet to these huge data centres. Then, whenever you need them, you can grab them back as long as you have internet and your login details.

Cloud storage came about because the old way of storing things on our computers had problems. We'd run out of space, have to constantly update things, and worry about our computers breaking down and losing everything. By letting these cloud companies handle the storage, we get better access to our files, more space when we need it, it costs less in the long run, and our data is usually much safer.

9.2 Various Types of Cloud Storage

Cloud storage types can be primarily categorized based on how they structure and manage data. The three main types are Object Storage, Block Storage, and File Storage, each designed for different use cases and data types.

1. Object Storage
2. File Storage
3. Block Storage

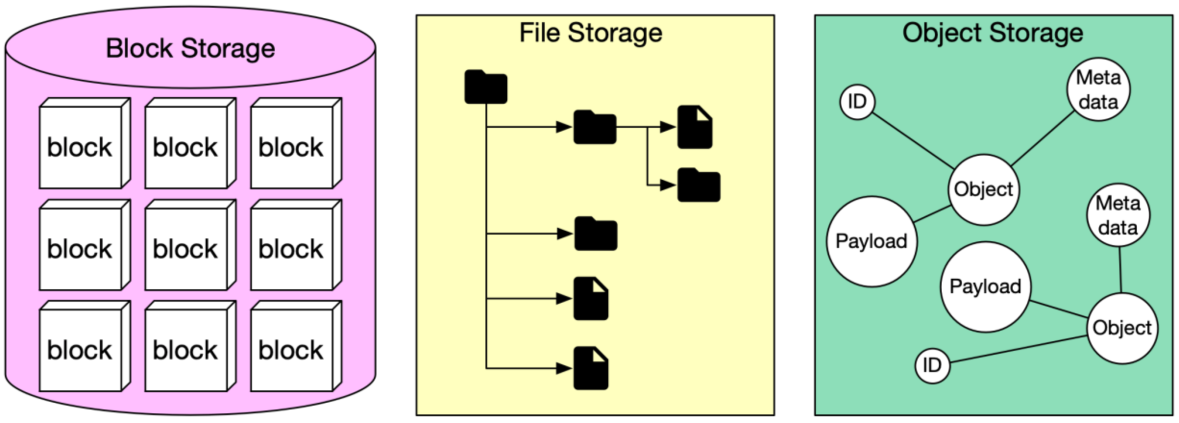


Fig 9.2 Various Types of Cloud Storage

a) Object Storage

Object Storage treats data like discrete units called objects, which are stored in a flat address space known as buckets. Each object includes the data itself, metadata (descriptive information about the data), and a unique identifier. This structure allows for massive scalability and efficient handling of unstructured data like images, videos, and log files. Object storage is ideal for cloud-native applications, data lakes, content delivery, backup, and archiving because of its scalability, cost-effectiveness for large volumes, and rich metadata capabilities that enhance searchability and data management.

Advantages of Object Storage:

- Scalability: Object storage is virtually limitless, allowing users to store petabytes of data without worrying about capacity constraints. It scales seamlessly as data grows.
- Cost-Effectiveness: It is generally cheaper than block storage, especially for large-scale, infrequently accessed data. Tiered storage options (e.g., AWS S3 Glacier, Azure Cool Blob) further reduce costs for archival use.
- Simplicity: Object storage uses a flat namespace with unique identifiers (keys) for objects, eliminating complex hierarchies and making it easy to manage huge amounts of data.
- Support for Unstructured Data: It is suitable for storing unstructured data like images, videos, logs, backups, and machine learning datasets, which don't require the structured access of block storage.
- Data Sharing and Collaboration: Object storage supports easy sharing through pre-signed URLs or public buckets, making it suitable for distributed teams or public content delivery.
- Built-in Features: Many object storage services offer features like versioning, lifecycle policies, event notifications, and integration with analytics or serverless computing, enhancing functionality.
- Global Distribution: Data can be stored in multiple regions or edge locations, improving latency and redundancy for global applications.

Disadvantages of Object Storage:

- Slower Performance: Object storage is slower than block storage, with higher delays and fewer operations per second. It's not good for fast apps like databases or real-time systems.
- Not Great for Frequent Changes: You can't easily edit objects. To update, you must replace or create a new version, which is slow and inefficient for data that changes often.
- No Traditional File System: It doesn't work like a regular hard drive. You can't do things like lock files or edit them directly, which limits its use for some apps.
- Tricky Permissions: Setting up who can access data can get complicated, especially with lots of data. Mistakes can accidentally make data public.
- Temporary Inconsistencies: Updates or deletions might not show up instantly everywhere, which can cause issues for apps needing immediate accuracy.
- Expensive and Slow Retrieval: Storing data in low-cost archival tiers is cheap, but getting it back can cost a lot and take hours.
- No Direct Connection to Machines: You can't attach object storage directly to a virtual machine, like block storage. You need to use APIs, which can be more complex.
- Hard to Switch Providers: Different cloud providers use different systems for object storage, so moving your data or apps to another provider can be difficult.

b) File Storage

File Storage, also called file-level or file-based storage, organizes data in a tiered structure of files and folders, similar to traditional file systems. It uses common file-level protocols like NFS (Network File System) and SMB/CIFS (Server Message Block/Common Internet File System), making it suitable for applications that need shared file access and a familiar file system interface. Multiple users or applications can often access and collaborate on files stored within the same file share, with varying levels of permissions and access control. It is providing a central location for teams to store and collaborate on documents and project files.

Advantages of File Storage:

- Shared Access: File storage supports simultaneous access by multiple users or systems via protocols similar to NFS (Network File System) or SMB (Server Message Block), making it ideal for collaborative environments, such as file shares for teams or applications.
- Ease of Use: File storage integrates seamlessly with operating systems and applications, requiring minimal configuration for users to read, write, or manage files.
- Compatibility: It supports standard file-based applications and workflows, such as content management systems, media editing, or enterprise software, without needing significant modifications.
- Scalability: Cloud-based file storage can scale capacity and performance dynamically, accommodating growing data needs without major infrastructure changes (e.g., AWS EFS, Azure Files).
- Centralized Management: It provides centralized control for permissions, access policies, and backups, streamlining administration for IT teams managing shared resources.
- Backup and Recovery: File storage often includes snapshot or versioning capabilities, enabling point-in-time backups and easy recovery from accidental deletions or corruption.
- Security Features: Cloud file storage typically offers encryption (at rest and in transit) and fine-grained access controls, ensuring data security besides compliance with regulations.
- Cost-Effective for Shared Workloads: For workloads requiring shared access, file storage can be more cost-efficient than provisioning multiple block storage volumes for individual instances.

Disadvantages of File Storage:

- Performance Limitations: File storage generally offers lower performance (e.g., latency, IOPS) related to block storage, making it less suitable for high-performance applications like databases or transactional systems.
- Cost for Large-Scale Use: While cost-effective for shared access, file storage can become expensive for large datasets or high-throughput needs, as pricing often scales with capacity and performance.
- Complexity in Distributed Systems: Managing file storage across multiple regions or ensuring consistency in globally distributed setups can be challenging due to latency or synchronization issues.
- Limited Flexibility: Unlike block storage, file storage doesn’t allow custom file system formatting or direct attachment to instances as a raw device, limiting its use for certain specialized workloads.

- Not Perfect for Unstructured Data: File storage is less efficient for massive, unstructured datasets (e.g., multimedia archives, big data analytics) compared to object storage, which is optimized for such use cases. Management Overhead: While centralized, managing permissions, quotas, and access controls for large user bases or complex folder structures can become time-consuming.
- Scalability Costs: While scalable, achieving high performance (e.g., increased throughput) often requires premium tiers, significantly raising costs compared to basic configurations.

c) Block Storage

Block Storage breaks data into set-size pieces, respectively having its ID, and keeps these pieces separate. It doesn't sort data into files or folders, instead offering basic storage volumes that connect to servers or virtual machines like they are local drives. This type of storage makes it easy to update data since only the changed pieces need to be rewritten, and it usually includes cool features like snapshots and replication to keep data safe and recoverable in case of disasters. Block storage services typically offer functionalities such as point-in-time snapshots to ensure data protection, as well as replication capabilities to enhance high availability and facilitate disaster recovery.

Advantages of Block Storage:

- Flexibility: Block storage volumes can be easily mounted to different operating systems and file systems, offering versatility in how storage is utilized.
- Scalability: Cloud-based block storage allows for easy scaling of storage capacity up or down as needed without a significant performance impact. You can add or remove block volumes dynamically.
- Redundancy and Reliability: Cloud providers often implement redundancy at the block storage level, distributing blocks across numerous physical devices to enhance data availability and durability.
- High Performance: Block storage provides low-latency and higher IOPS (Input/Output Operations Per Second), ideal for performance-sensitive applications like databases, virtual machines, and transactional systems.
- Data Security: Features like encryption at rest and in transfer, along with access controls, ensure data protection and compliance with regulatory standards.

Disadvantages of Block Storage:

- Higher Cost: Block storage is generally more expensive per gigabyte compared to other cloud storage options like object storage, especially for large capacities.
- Limited Use Case Suitability: It’s optimized for structured data and high-performance workloads (e.g., databases, VMs) but not ideal for unstructured data, large-scale media storage, or scenarios requiring frequent data sharing, where object storage excels.
- Complexity in Management: Managing block storage volumes, including partitioning, formatting, and ensuring data consistency across blocks, can be more complex than managing simple files or objects.
- Scalability Constraints: While scalable, block storage may have upper limits on volume size or performance (e.g., max IOPS or throughput), and resizing or reconfiguring volumes can sometimes involve operational overhead.

- No Native Data Sharing: Unlike file storage (e.g., NFS) or object storage (e.g., S3), block loading is typically attached to a single instance or VM, making it less suitable for shared access across multiple systems without additional setup.
- Dependency on Compute Instances: Block storage is often tied to specific virtual machines or instances, meaning it cannot be accessed independently, unlike object storage, which can be accessed via APIs or URLs.
- Data Organization Overhead: Since block storage doesn't inherently understand file structures, the responsibility of organizing and managing data at a file system level falls on the user or the applications using the storage.

9.3 Concept of Cloud Databases

A cloud database is a database which is executed on a Cloud computing system and is accessed via the internet. Imagine it this way: rather than keeping all your school notes in a notebook that you carry around in your backpack (your local computer), you're keeping them in a safe locker (the cloud) from which you can access them from anywhere via your phone, laptop, or tablet. Cloud databases are operated and maintained by service providers such as Amazon Web Services (AWS), IBM Cloud, or Google Cloud.

All these providers handle everything for you—safe storage of data, automatic updating of software, backing up, and scaling resources when more individuals begin to use the app. This translates into no need for businesses or even schools to purchase costly hardware or concern themselves with physical damage, maintenance, or loss of data. For instance, a school that uses a cloud-based system to store student records can quickly access marksheets or attendance reports anywhere, without access to a special server room. Cloud databases are not only economical but also ensure handling of data is faster, more secure, and reliable.

9.4 Usage of Cloud Databases

Cloud databases are employed in numerous industries these days due to their speed, flexibility, and capacity to scale as required. Here's a closer examination of how they're employed:

Gaming
New computer games are not merely a piece of data on your machine. They comprise massive, interactive worlds with real-time occurrences, player information, leaderboards, and in-game buying. Cloud databases handle all this data storage and management. Ponder a game such as PUBG or Fortnite—it supports millions of users playing simultaneously, across various regions of the globe. Cloud databases enable this by processing all information regarding players, game stages, and updates in an instant. The developers can patch the game for every user without requiring them to download massive files; information is simply altered on the cloud and shared with all automatically.

Data Storage and Backup
Those days of having files saved on pen drives or hard disks are gone. Tools such as Google Drive and OneDrive allow you one upload your files to the internet. The cloud database of such services holds your files securely and makes them accessible from anywhere. For instance, if you leave your homework file in home, you can access your Google Drive account in school and download it. Such services also include automatic backup. Therefore, if your laptop crashes or your smartphone is stolen, your essential files won't go missing—they're still securely stored in the cloud.

Business Uses
Cloud databases are utilized by businesses for enhance their workflow. Software such as Slack and Asana enables teams to work together and collaborate on projects online. They can delegate tasks, post comments, exchange files, and monitor progress in real-time. Likewise, accounting tools like QuickBooks Online keep financial information in the cloud. Companies no longer have to store sensitive financial information on local devices. Cloud databases keep the data up to date in real time, backed up, and available to only authorized users.

Internet of Things (IoT)
IoT is a organization of intelligent devices—such as fitness trackers, home automation devices, or factory equipment—that gather and transfer data to the cloud. Cloud databases hold and analyze this data. Your smartwatch, for instance, monitors your heartbeat, steps, and sleep cycle. It uploads this data to the cloud, where it can be processed and viewed through your mobile app. It is the same in factories, where machines with sensors upload data to the cloud so that the operator can track performance and anticipate problems before they occur.

Entertainment and Media Streaming
Streaming providers such as Netflix and Spotify don't hold content on your hardware. Instead, they employ cloud databases to retrieve your preferences, watch history, and stream movies or tracks. When many people watch shows at the same time, the cloud infrastructure powering these providers scales automatically to handle the load. You don't need to wait for downloads; you click and it plays, all because of robust cloud databases providing the content and user experience behind the scenes.

Education and E-Learning
The pandemic later saw educational institutions such as schools and colleges adopting online learning platforms such as Zoom, Google Classroom, and Coursera. These learning platforms keep videos, assignments, notes, and student information in cloud databases. When a teacher uploads a lesson, it gets saved in the cloud, and students can view it whenever they wish. Teachers can mark attendance, administer tests, and offer feedback without the use of paper or offline mechanisms. Cloud databases do all this in the background and enable students to learn from anywhere, at any time.

Financial Services and Banking
Banks and fintech organizations use cloud databases to store transaction histories, account information, and financial analytics. These are highly secure and fast systems. For example, when you pay using UPI or view your bank balance on your mobile app, you're accessing data from a cloud database. Cloud computing assists with fraud detection, where questionable transactions can be marked immediately. Cloud computing also cuts down on the necessity for physical branches, banking thus becoming more convenient and within reach for customers.

9.5 Management of Files in the Cloud

Suppose you have a school locker that you can access from your home, school, or even a friend's place with the use of your mobile. Your cloud storage is this locker, and it keeps all your digital files secure. Cloud file management is the process of organizing, storing, accessing, sharing, and securing your files online through tools such as Google Drive, Dropbox, or OneDrive.

9.5.1 Cloud File Management

Digital file management entails storing, organizing, accessing, and sharing files like documents, photos, videos, and presentations on cloud storage services. Instead of saving files on physical devices like USB drives, desktop hard drives, or memory cards, users can now securely store their data on the internet on services offered by cloud platforms like Google Drive, Microsoft OneDrive, Dropbox, and even iCloud.

Such platforms allow users to upload files to a central online location, also known as the "cloud," which is accessible through the internet on any device including computers, tablets, and smartphones. This development liberates users from being bound to a single device. For instance, a student can complete a school project at home and save it on Google Drive. Later, the student can open the project on a different computer at school or even from their mobile phone while they're on-the-go.

Cloud file management is essential now more than ever because of the increased need for flexibility, collaboration, and security of information. Every student, professional, organization, and even businesses are now able to store and manage their crucial files in a single location which can be accessed anytime from any place. This ability to centralize file storage is beneficial for everyone. Furthermore

9.5.2 Working of Cloud File Management

Here's how managing files in the cloud works in simple steps:

1. Saving Files: Uploading files involves dragging them onto your cloud drive. For instance, if you have completed a Biology project, you save the Word document or PDF in your Google Drive. The cloud permits saving any size and type of document, subject to your free space.

2. Organizing Files: Similar to organizing your school bag, you can make folders and label them for every subject, such as "Class 12 Physics" or "English Assignments." Within these folders, you can organize files based on date, type, or tags so that they're easily accessible. This keeps your drive free of clutter and helps you study efficiently.

3. Accessing Files: Home or school, you can view your cloud files by logging into your account. Even when offline, some cloud services permit access to downloaded or synced files. Therefore, you'll always have notes whenever you need them.

4. Sharing Files: Cloud storage makes it easy to share files with friends or teachers. Rather than emailing large attachments, you can share a link. You can choose if others can only look at the file, can make suggestions for changes, or can edit it with you. It's ideal for group projects.

5. Storing Files Safely: Cloud platforms employ robust protection, such as encryption, to secure your files. They also support features like file recovery (if you delete something) and version history (to view older versions). You can also turn on two-step verification foradded security.

9.6 Syncing of Files

Syncing files is magic—it updates everything everywhere! If you add a note to your notes on your computer, syncing will make that same change on your phone and other machines right away. It's like changing one copy of a notebook, and the other copies change themselves.

9.6.1 Cloud File Syncing

Cloud syncing of files guarantees that any updates to files in one place automatically propagate to all other places linked to the same cloud account. Therefore, if you write your Chemistry notes on your school computer and save them in your synced Google Drive folder, when you get home and open Google Drive on your mobile, the updated file will already be loaded. This makes things simple for students who work on more than one machine or collaborate on projects with fellow students.

9.6.2 Working of Cloud File Syncing

1. Uploading Changes: As you save or modify a file in your synced folder, the cloud storage program notices the change and sends it to the internet.

2. Server Updates: The cloud server saves and stores this updated version. Others even store old versions in case you want to undo something.

3. Downloading to Other Devices: With updates on the cloud, the file automatically downloads to your other devices that are signed in using the same cloud account.

4. Two-Way Syncing: You can modify on any device—laptop, phone, or tablet—and all remains in sync. Two-way communication.

5. Conflict Resolution: When changes are done on two devices simultaneously, cloud systems create a second copy or enable the user to merge changes, such as in Google Docs.

6. Offline Syncing: Your changes are not only stored on your device even offline but are also synced with the cloud and other devices once you are back online.

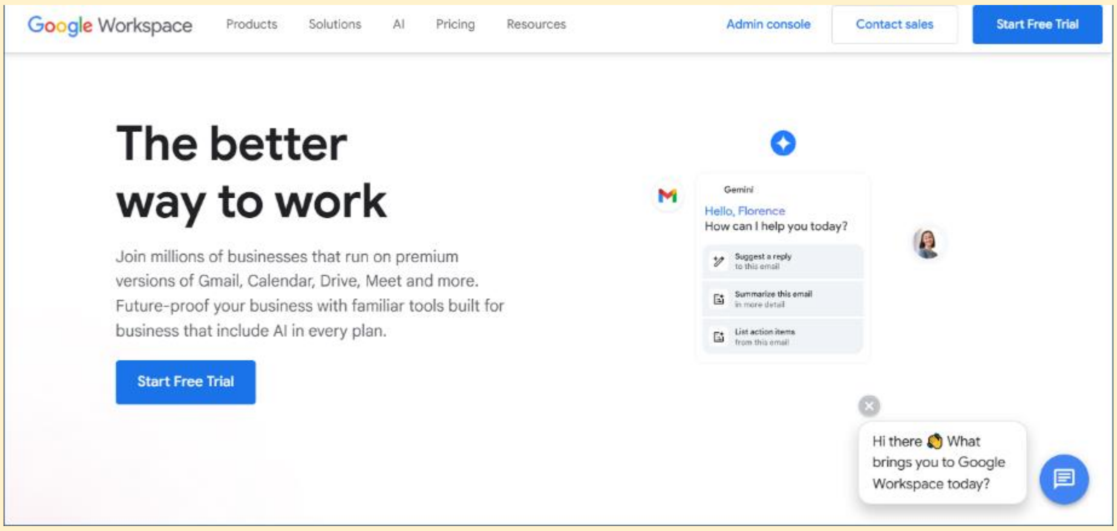
Example: You begin a PowerPoint presentation for your Business Studies project in school and save it in your synced OneDrive folder. At home, you open your laptop, and the latest copy is already ready to go—no USB drive, no email required!

Practical activity 9.1 Demonstrate Setting up cloud storage account and cloud-based project using Google Drive, Microsoft Teams

1. **Setting up cloud storage account**

   Material required: Google workspace

Step 1: Go to google workspace using https://workspace.google.com/.



The better way to work

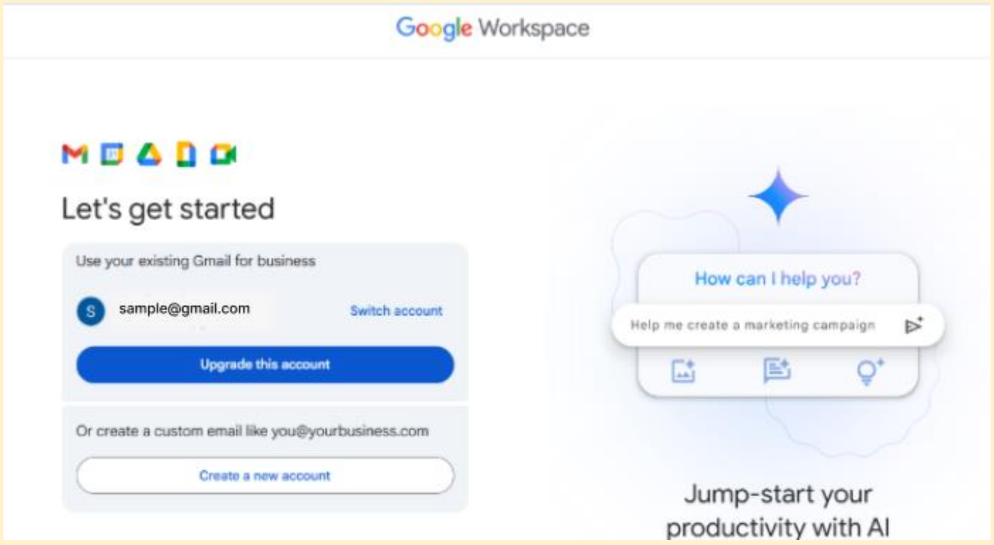
Join millions of businesses that run on premium versions of Gmail, Calendar, Drive, Meet and more. Future-proof your business with familiar tools built for business that include AI in every plan.

Start Free Trial

Hello, Florence
How can I help you today?

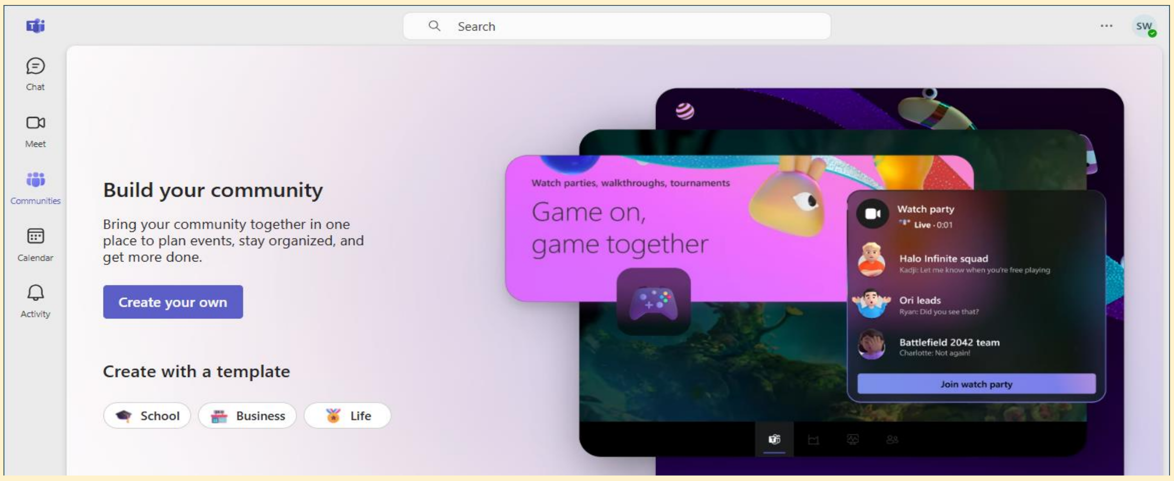
Hi there 👋 What brings you to Google Workspace today?

Step 2: Click on Start free trial.


Google Workspace
Let's get started
Use your existing Gmail for business
sample@gmail.com
Switch account
Upgrade this account
Or create a custom email like you@yourbusiness.com
Create a new account
How can I help you?
Help me create a marketing campaign
Jump-start your productivity with AI

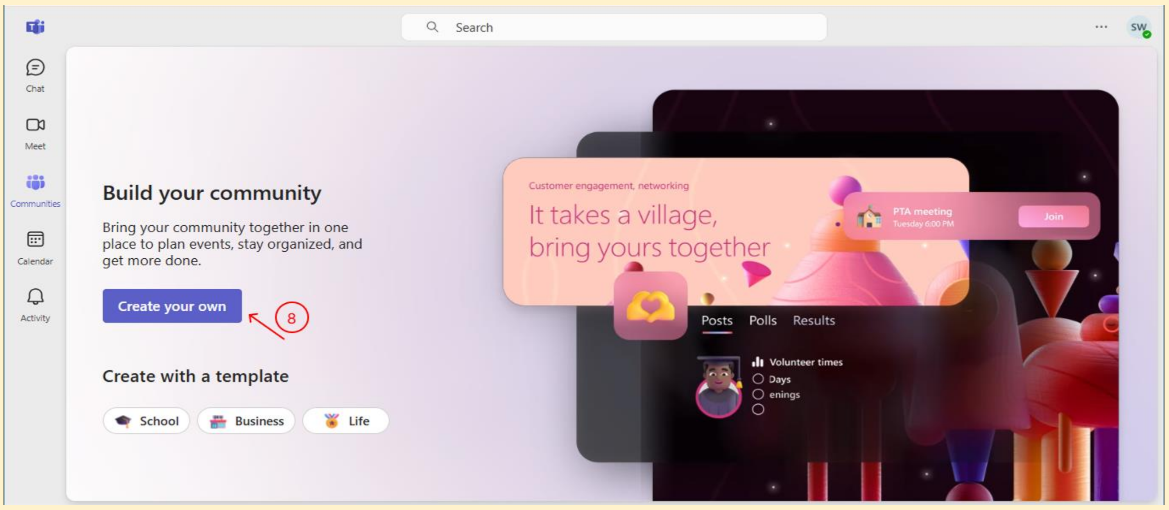
2. Setting up cloud-based project using Google Drive, Microsoft Teams

Step 1: Go to https://www.microsoft.com/en-us/microsoft-teams.



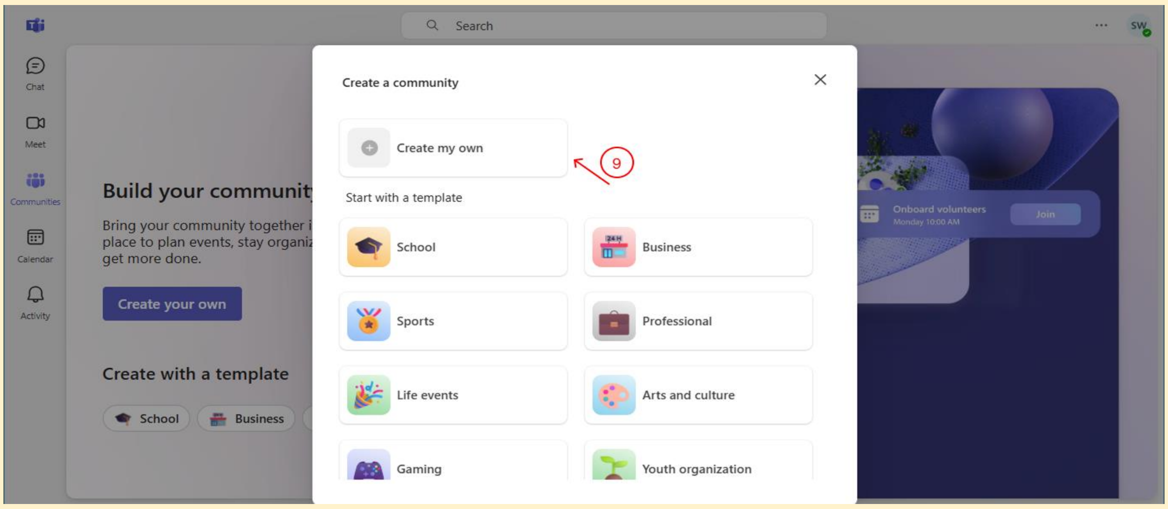
Now, go to the Microsoft team by this link.

Step 2: Click on the Create your own button.


Search
Chat
Meet
Communities
Calendar
Activity
Build your community
Bring your community together in one place to plan events, stay organized, and get more done.
Create your own
8
Create with a template
School
Business
Life
Customer engagement, networking
It takes a village, bring yours together
PTA meeting
Join
Posts
Polls
Results
Volunteer times

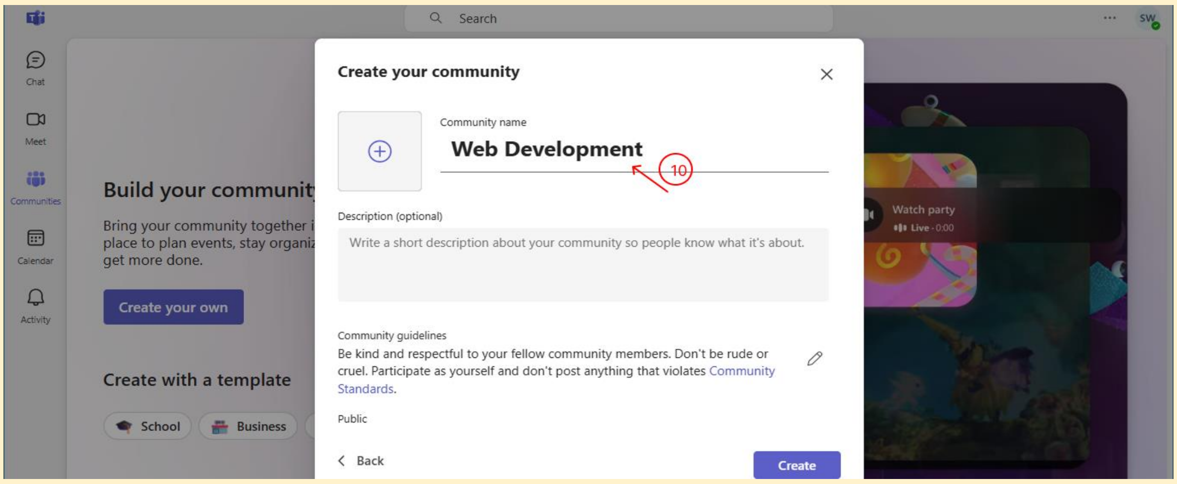
Select this button to create a new team or community.

Step 3: Select the Create my own button.



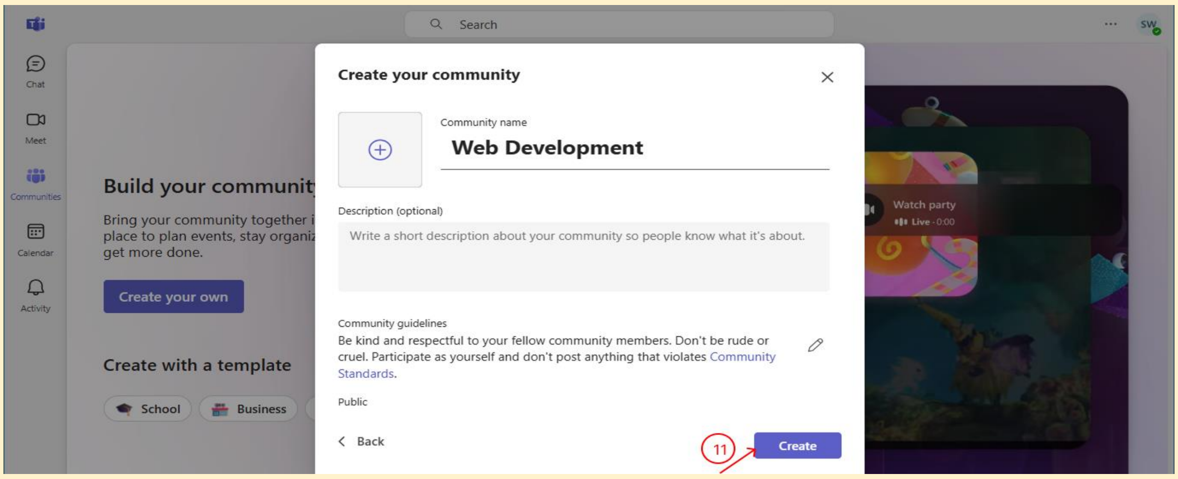
For creating your own team with your team members, select on Create my own button.

Step 4: Write your team/community name here.


Search
Create your community
Community name
Web Development
10
Description (optional)
Write a short description about your community so people know what it's about.
Community guidelines
Be kind and respectful to your fellow community members. Don't be rude or cruel. Participate as yourself and don't post anything that violates Community Standards.
Public
Back
Create
Watch party

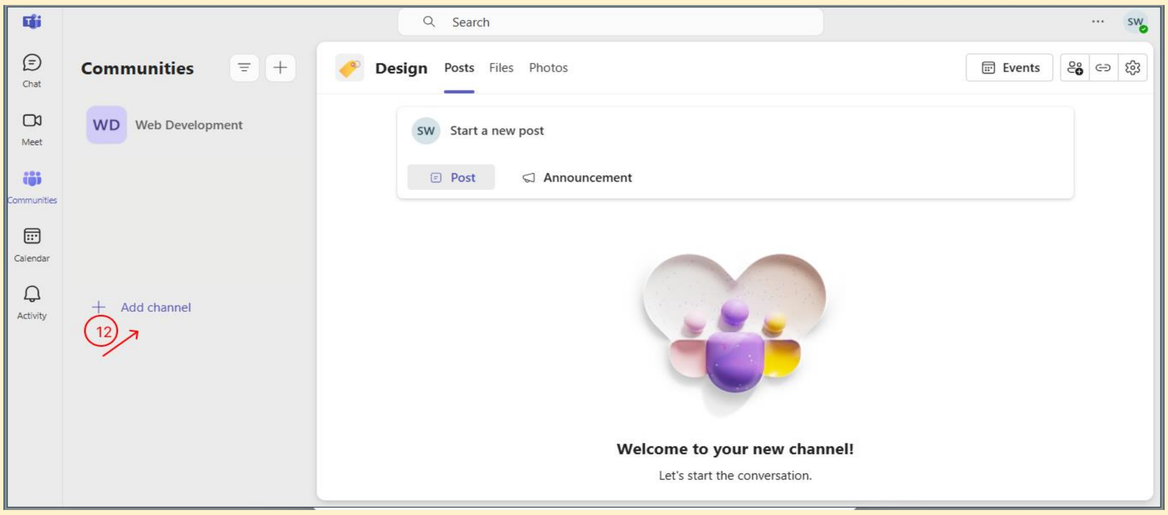
Now, mention your team or community name.

Step 5: Click on the Create button.


Search
Create your community
Community name
Web Development
Description (optional)
Write a short description about your community so people know what it's about.
Community guidelines
Be kind and respectful to your fellow community members. Don't be rude or cruel. Participate as yourself and don't post anything that violates Community Standards.
Public
Back
11
Create

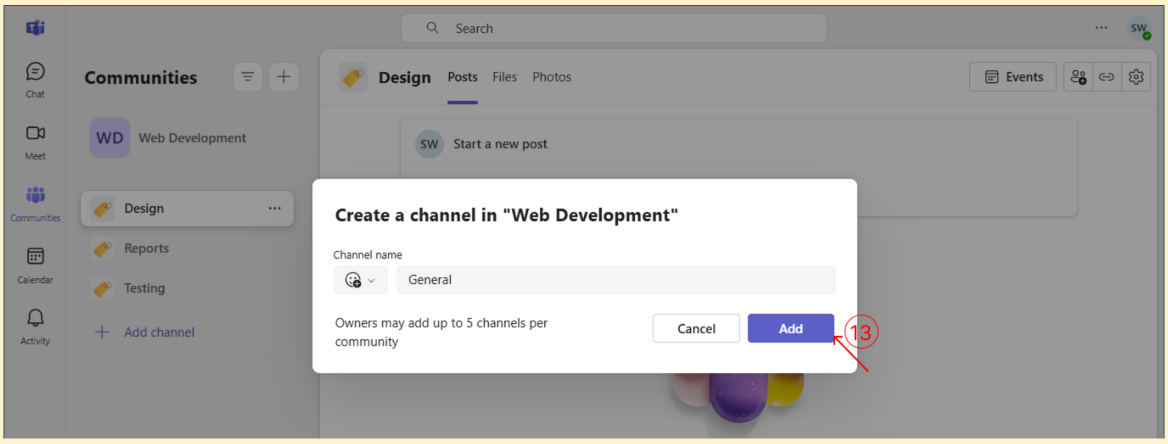
Once you have filled in the necessary details and are satisfied with the settings, click the "Create" button located at the bottom right of the dialog box to finalize and create your new community.

Step 6: Click on Add channels to create channels.



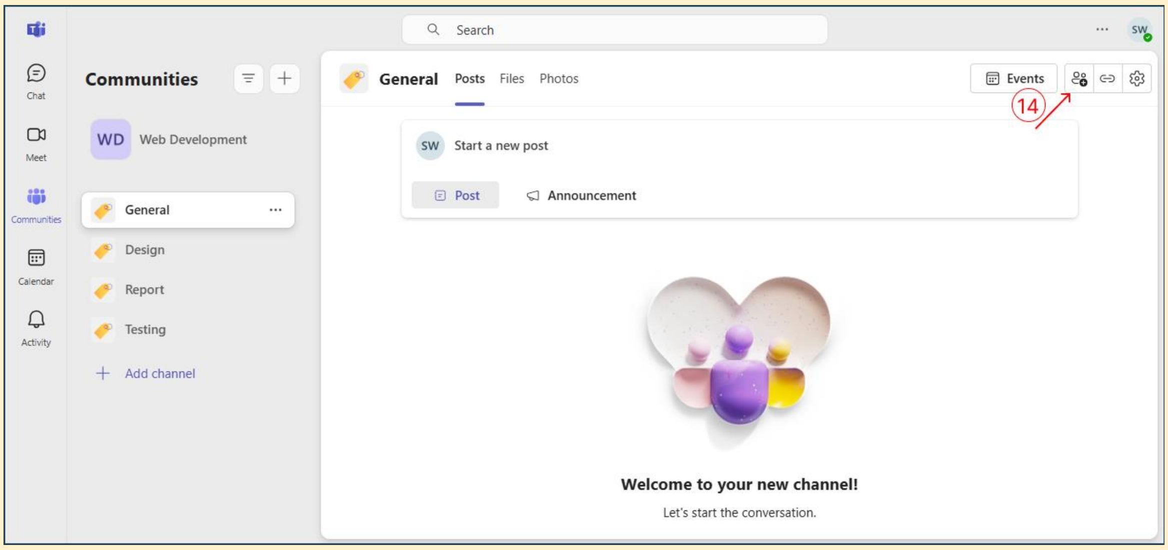
This action will open a dialog box to configure your new channel.

Step 7: Create channels and click on the Add button.



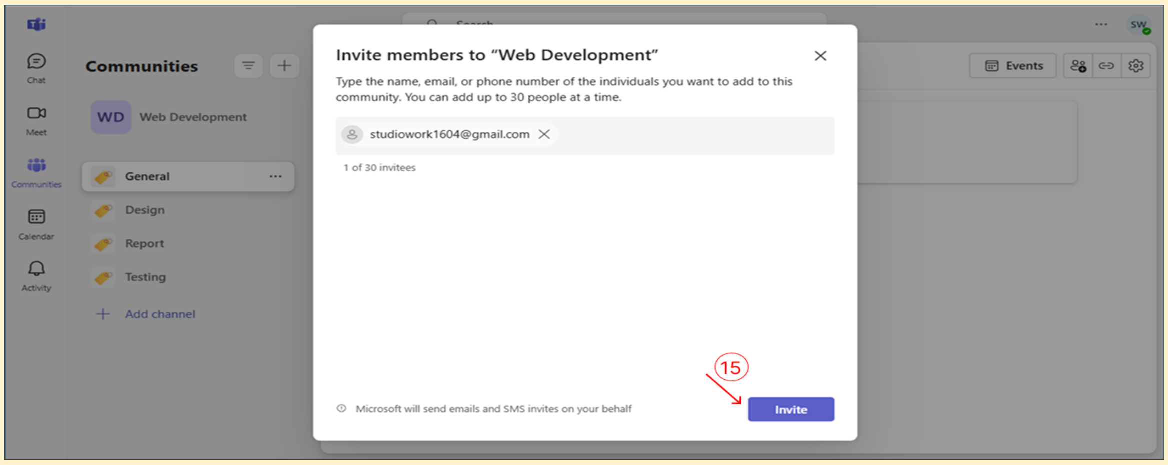
Create all channels like General, Design, Reports, Testing for different docs, PPT slides, and other work.

Step 8: Click on the invite button.



Through this, you can invite or add your team members in your team.

Step 9: Invite a team member.



Invite members to "Web Development"

Type the name, email, or phone number of the individuals you want to add to this community. You can add up to 30 people at a time.

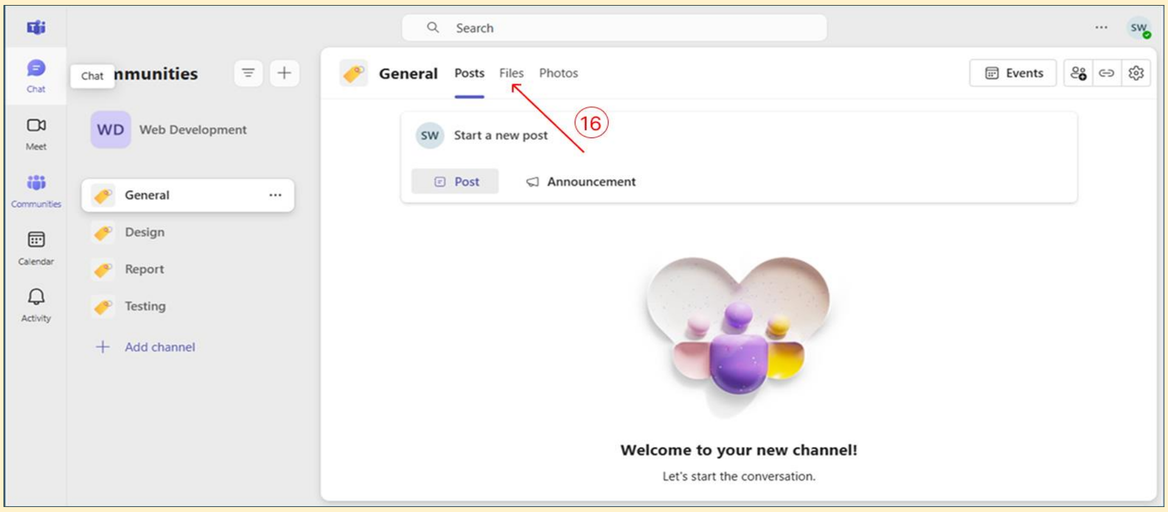
studiowork1604@gmail.com

1 of 30 invitees

Microsoft will send emails and SMS invites on your behalf

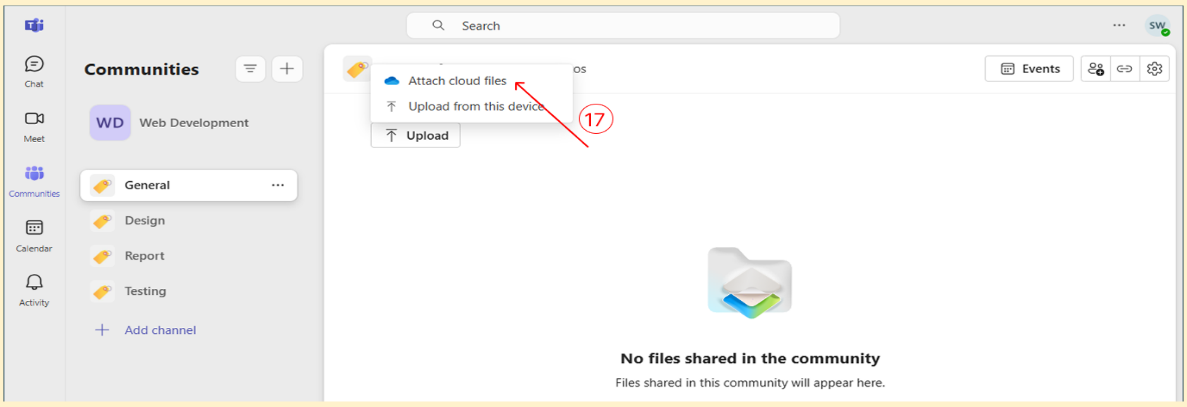
Mention your team member's phone number or email ID. And then click on the invite button.

Step 10: Select on Files.



For uploading files, go to the Files section.

Step 11: Select Attach Cloud Files.



For the cloud project, select Attach cloud files option to share the cloud project with your team members.

Case studies based on Cloud computing

1. Netflix

Netflix is among the leading video streaming sites globally, providing users with thousands of titles in the form of movies, television shows, and original series. To host so much content and stream it without interruption or delay, Netflix utilizes cloud computing, primarily through Amazon Web Services (AWS). Earlier, Netflix stored videos and served users from its own data centers, but as its popularity increased, it was not easy to manage traffic, scale services, and maintain smooth performance. That's when Netflix shifted entirely to the cloud.

By leveraging AWS, Netflix is able to store and deliver massive video files effectively. Cloud computing enables the company to stream to millions of viewers simultaneously, even at times of peak usage when numerous individuals are streaming. For instance, when a new show such as Stranger Things is launched, millions of users view it across various countries. With the cloud, Netflix doesn't have to concern itself with server saturation, it can dynamically scale up its services when demand is high and scale down when demand is low.

The cloud also enables other features of Netflix, such as personalized suggestions, subtitles in several languages, and features such as "resume watching." All these services need quick access to user data, which is handled in the cloud. Through the utilization of cloud storage and computing, Netflix provides high availability, minimal buffering, and consistent streaming quality for all its viewers worldwide. This case overwhelmingly indicates how cloud computing makes companies go global, efficient, and user-friendly.

2. Amazon

Amazon is one of the world's robust e-commerce giants, with millions of customers from various countries visiting their sites daily. Many are unaware that Amazon is also topmost in cloud computing through its platform, Amazon Web Services (AWS). While Amazon.com facilitates individuals purchasing products on the internet, AWS is the behind-the-scenes power engine driving not just Amazon's own business but also offering cloud services to other large organizations. Cloud computing is at the heart of Amazon's success because it enables the company to process huge amounts of data, respond to customers quickly, and stay online continuously.

In a typical configuration, a company would require numerous physical servers to hold customer information, contain product listings, process payments, and monitor deliveries. Such systems are difficult to control, costly, and challenging to scale. Amazon fixes this issue through cloud computing. With AWS, Amazon holds all its valuable data, such as customer records, inventory, and orders—in the cloud. This makes the information accessible from anywhere and makes it safe and backed up. Amazon is also able to make quick changes to its services through the cloud, like introducing new product categories or selling flash sales.

High-traffic events such as Amazon Prime Day or Black Friday, when millions of visitors come to the website concurrently, are also made possible by cloud computing. Without the cloud, Amazon's servers could crash from being overwhelmed. But with AWS, the system is able to scale resources such as storage capacity and processing capabilities automatically when traffic is high. After the event, the additional resources are relinquished, saving costs. This flexibility is one of the utmost advantages of cloud computing because it allows businesses to expand and contract resources based on demand.

Another key fact is that AWS is not just utilized by Amazon itself but also provided to other entities globally. Nowadays, well-known services such as Netflix, Spotify, Adobe, and even NASA utilize AWS for their own products. AWS offers capabilities for website hosting, data storage, artificial intelligence, analytics, and more. Through offering these facilities, Amazon has made cloud computing a significant portion of its business. This case study readily demonstrates how cloud computing enables scalability, reliability, flexibility, and innovation, not only for Amazon, but also for thousands of companies worldwide.

3. Zoom Meetings

Zoom was a common name amid the COVID-19 pandemic when offices, schools, and organizations moved to digital meetings and classes. In the background, Zoom employs cloud computing to back its video conferencing solution. Zoom is assisted by the cloud to handle millions of video calls on a daily basis, without slowdown or disruptions. Whether it's a student participating in an online course or a business conducting an international meeting, Zoom makes it possible for anyone to participate in real time from any corner of the globe.

One of the most significant aspects of Zoom's Cloud infrastructure is scalability. When more individuals began to use Zoom during lockdowns, the platform needed to scale immediately. With cloud computing, Zoom was able to install more servers and bandwidth in an instant, without having to construct physical infrastructure. This enabled it to keep up with the sudden surge in usage. It also stores recorded meetings, chat history, and user preferences in the cloud securely.

In addition, cloud services allow Zoom to preserve video and audio quality, identify connectivity problems, and secure data using encryption. If it were not for the cloud, Zoom could not have grown so fast and offer a stable service to millions of users on various devices and networks. This example illustrates how cloud computing enables more flexible, efficient, and reliable communication, even in times of international crises.

ASSESSMENT

A. Multiple Choice Questions

1. Cloud storage is best described as:
   a) Storing files only on your local device.
   b) Keeping your computer files on faraway computers and accessing them via the internet.
   c) Using physical lockers to store digital files.
   d) Storing data only on USB drives.

2. Which among the following is NOT a benefit of cloud storage?
   a) Better access to files from anywhere.
   b) More storage space when needed.
   c) Higher long-term costs.
   d) Improved data safety.

3. Object storage is best suited for:
   a) Storing operating systems.
   b) Handling unstructured data like images and videos.
   c) Storing data in a hierarchical structure of files and folders.
   d) Applications requiring high-speed, frequent data changes.

4. In object storage, data is stored as:
   a) Files and folders.
   b) Blocks with IDs.
   c) Discrete units called objects in buckets.
   d) Tables and rows.

5. Which of the following is a merit of object storage?
   a) Slower performance.
   b) Difficult to edit objects.
   c) Scalability.
   d) Tricky permissions.

6. File storage organizes data in a:
   a) Flat address space.
   b) Hierarchical structure of files and folders.
   c) Series of data blocks.
   d) Database format.

7. Which protocol is commonly used in file storage?
   a) HTTP
   b) TCP/IP
   c) NFS
   d) SMTP

8. Which among the following is a disadvantage of file storage?
   a) Shared access.
   b) Ease of use.
   c) Performance limitations.
   d) Centralized management.

9. Block storage breaks data into:
    a) Objects.
    b) Files.
    c) Set-size pieces with IDs.
    d) Folders.

10. Block storage is idyllic for:
    a) Storing unstructured data.
    b) High-performance applications like databases.
    c) Large-scale media storage.
    d) Scenarios requiring frequent data sharing.

B. Fill-in-the-blanks

1. Cloud storage is like keeping your computer files on many faraway _________ instead of just on your device.
2. Object Storage treats data as distinct units called _________, which are stored in a flat address space known as _________.
3. Each object includes the data itself, _________ (descriptive information about the data), and a unique _________.
4. Object storage is ideal for cloud-native applications, data _________, content delivery, backup, and archiving.
5. Object storage uses a flat _________ with unique identifiers (keys) for objects, eliminating complex hierarchies.
6. File Storage arranges data in a _________ structure of files and folders, similar to traditional file systems.
7. Common file-level protocols used in file storage include NFS (Network File System) and _________.
8. Block Storage breaks data into _________-size pieces, each having its ID, and keeps these pieces separate.
9. A cloud database is a database that is executed on a cloud computation system and is accessed via the _________.
10. Cloud databases are operated and maintained by service providers like Amazon Web Services (AWS), Microsoft Azure, or _________.

C. True/False questions

1. Cloud storage involves storing files on your local device's hard drive.
2. Cloud storage permits you to contact your files from wherever with an internet connection.
3. Object storage arranges data in a hierarchical structure of files and folders.
4. Object storage is ideal for packing unstructured data like videos.
5. Object storage is generally faster than block storage.
6. File storage uses protocols like NFS and SMB.
7. File storage is well-suited for high-performance applications like databases.

8. Block storage breaks data into set-size pieces with IDs.
9. Block storage provides high performance and low latency.
10. Block storage is typically cheaper than object storage.

Short Questions and Answers:

1. What is cloud storage? Name the three main kinds of cloud storage.
2. How does an object storage treat data? Give two advantages and disadvantages of object storage.
3. How does file storage organize data? What are two common file-level protocols used in file storage?
4. How does block storage break up data? Give two advantages and disadvantages of block storage.
5. What is a cloud database?
6. Name three cloud service providers that offer cloud databases.
7. Give two benefits of using cloud databases.
8. How are cloud databases used for data storage and backup?
9. How are cloud databases used in the Internet of Things (IoT)?
10. How are cloud databases used in entertainment and media streaming?

Answer Key

A. Multiple Choice Questions

1.b, 2.c, 3.b, 4.c, 5.c, 6.b, 7.c, 8.c, 9.c, 10.b

B. Fill-in-the-blanks

1. servers, 2. objects, buckets, 3. metadata, identifier (ID), 4. Analytics, 5. Namespace, 6. Hierarchical, 7. SMB (Server Message Block), 8. Fixed, 9. Internet, 10. Google Cloud

C. True/False questions

1.False, 2. True, 3. False, 4. True, 5. False, 6. True, 7. False, 8. True, 9. True, 10. False

विद्यया ऽमृतमश्नुते

एन सी ई आर टी
NCERT

PSS CENTRAL INSTITUTE OF VOCATIONAL EDUCATION

(a constituent unit of NCERT, under Ministry of Education, Government of India)
Shyamla Hills, Bhopal- 462 002, M.P., INDIA

www.psscive.ac.in