

**CLOUD COMPUTING UNIT – 6 PYQ'S**➤ **MAY / JUN 2022****Q7****a) Discuss Energy Aware Cloud Computing with suitable example. [9 Marks]**

**Energy Aware Cloud Computing** refers to techniques and strategies aimed at **reducing energy consumption** in cloud data centers while **maintaining performance and availability**. It focuses on achieving **energy efficiency**, **cost reduction**, and **environmental sustainability**.

**Need for Energy Awareness:**

- Cloud data centers consume a **large amount of electricity**.
- High energy usage leads to **higher operational costs** and **carbon emissions**.
- Growing demand for cloud services calls for **eco-friendly infrastructure**.

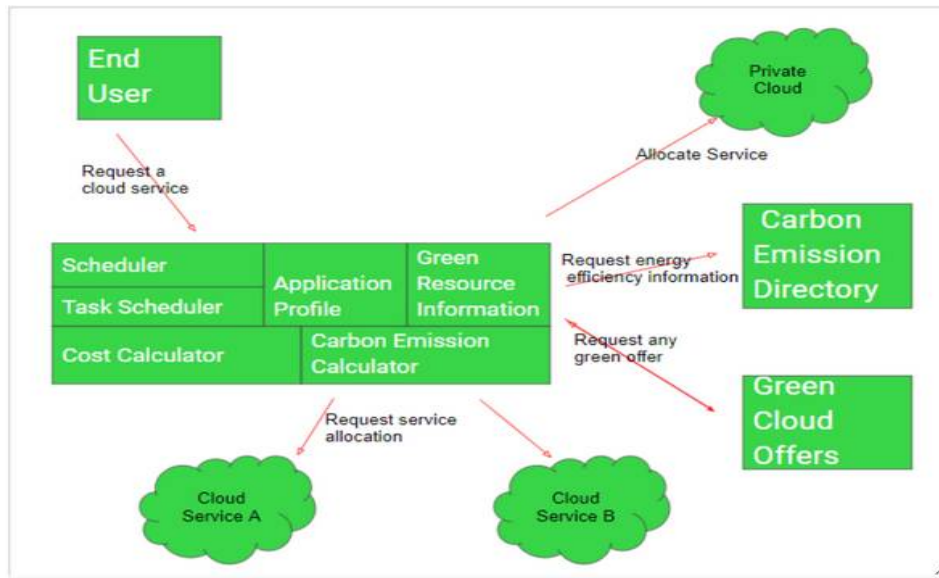
**Techniques for Energy Efficiency:**

<b>Technique</b>	<b>Description</b>
<b>Dynamic VM Consolidation</b>	Migrating VMs to fewer servers during low usage to switch off idle servers.
<b>Load Balancing</b>	Distributes workloads efficiently to avoid overloading any single server.
<b>DVFS (Dynamic Voltage and Frequency Scaling)</b>	Reduces CPU voltage/frequency during low load to save power.
<b>Energy-aware Scheduling</b>	Assigns tasks to resources based on energy consumption patterns.
<b>Renewable Energy Integration</b>	Using solar, wind, or hydroelectric power for green cloud computing.

**Example:**

A cloud provider like **Google Cloud** or **Amazon Web Services (AWS)** implements energy-aware strategies by:

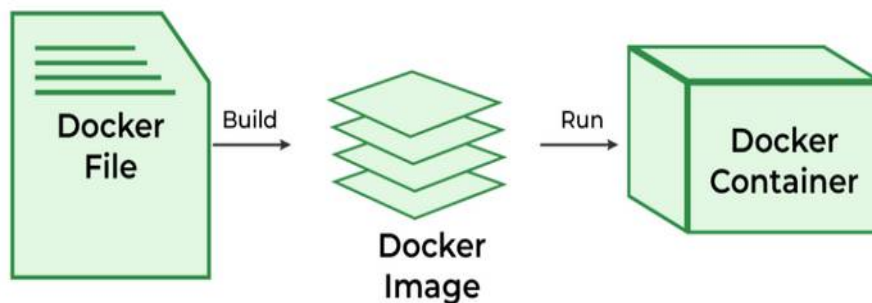
- **Using AI-based workload prediction** to schedule low-priority tasks during non-peak hours.
- **Shutting down unused servers** in off-peak times.
- **Cooling systems optimized** using Machine Learning to save electricity.



If u want u can draw diagram , nahi keli tari chalel !!

b) Explain with example, working of Docker. [9 Marks]

**Docker** is an open-source platform used to **build, ship, and run applications** in **lightweight containers**. Containers package an application and its dependencies together, ensuring it runs consistently across different environments.



**Key Concepts:**

Term	Description
Image	A blueprint of the application (including code, runtime, libraries).
Container	A running instance of an image; isolated from the host system.

---

<b>Dockerfile</b>	A script with instructions to build a Docker image.
-------------------	---

---

<b>Docker Hub</b>	A public registry to store and share Docker images.
-------------------	---

---

### 3. Working of Docker (Step-by-Step):

#### Step 1: Write a Dockerfile

```
dockerfile                                                                    Copy Edit

FROM python:3.8
COPY app.py /app/app.py
WORKDIR /app
RUN pip install flask
CMD ["python", "app.py"]
```

#### Step 2: Build the Docker Image

```
bash                                                                    Copy Edit

docker build -t flask-app .
```

#### Step 3: Run the Container

```
bash                                                                    Copy Edit

docker run -p 5000:5000 flask-app
```

This launches the container, exposing the Flask web app on port 5000.

#### Example:

You have a **Flask web application** (app.py). With Docker:

- You write a **Dockerfile** to define environment setup.
- Docker **packages the app** and all dependencies into an image.
- You can run it **anywhere** (local machine, cloud, testing server) with the same behavior.

#### Advantages of Docker:

- **Portability:** Runs on any system with Docker.
- **Isolation:** Each container is independent.
- **Efficiency:** Uses fewer resources than virtual machines.
- **Scalability:** Easily scale containers up/down.

Q8

a) How do the Cloud and IoT work together for Home Automation? [9 Marks]

**IoT (Internet of Things)** and **Cloud Computing** are key enablers of **smart home automation**. IoT devices collect and exchange data, while the cloud provides **storage, processing power, and remote access**.

## 2. Working of Cloud and IoT in Home Automation:

### Step-by-Step Process:

1. **Sensors & Devices:** Smart devices like lights, thermostats, cameras, and door locks sense data (e.g., temperature, motion).
2. **Connectivity:** Devices connect via Wi-Fi, Zigbee, or Bluetooth to a local gateway or directly to the internet.
3. **Data Transfer to Cloud:** Collected data is sent securely to a **cloud platform** (like AWS IoT, Google Cloud IoT).
4. **Cloud Processing & Storage:**
  - Analyzes data using cloud-based AI/ML algorithms.
  - Stores user preferences and device logs.
5. **Remote Access & Control:**
  - Users interact with devices via **mobile apps** or **web dashboards**.
  - Cloud sends control signals to IoT devices.
6. **Automation & Alerts:**
  - Triggers automated actions (e.g., turn on lights at sunset).
  - Sends alerts (e.g., intrusion detected, smoke alarm).

## 3. Example:

A **smart thermostat** uses temperature sensors to monitor room conditions. It sends data to the **cloud**, which:

- Analyzes the pattern of user preferences.
- Automatically adjusts temperature.
- Sends a mobile notification to the user.

## 4. Benefits:

- **Remote Monitoring** and Control.
- **Automation** based on real-time data.

- **Energy Efficiency** and cost savings.
- **Improved Security** (e.g., CCTV storage in the cloud).

Q8

**b) Differentiate Distributed Cloud Computing vs Edge Computing [9 Marks]**

Both **Distributed Cloud Computing** and **Edge Computing** aim to bring computation closer to data sources, but they differ in architecture, control, and use cases.

**Definitions:**

- **Distributed Cloud Computing:**  
The public cloud infrastructure is extended to multiple physical locations (data centers), but operations, updates, and services are centrally managed by the cloud provider.
- **Edge Computing:**  
Computation and data processing occur at the **edge** of the network — closer to the data source (e.g., IoT devices) — reducing latency and bandwidth usage.

**Differences Between Distributed Cloud and Edge Computing:**

Feature	Distributed Cloud Computing	Edge Computing
Location of Processing	Multiple cloud regions or zones	Near or at the data source (edge devices)
Control	Managed centrally by cloud provider	Often managed locally or on-site
Latency	Moderate (depends on cloud region proximity)	Very low (real-time responses possible)
Use Case Examples	Multi-region app deployment, compliance, DR	Smart cameras, autonomous vehicles, IoT sensors
Hardware Dependency	Relies on cloud data centers	Utilizes edge gateways/devices
Scalability	Highly scalable via cloud infrastructure	Limited scalability at each edge location

<b>Data Storage</b>	Centralized or regionally distributed in the cloud	Local storage on edge devices
<b>Bandwidth Usage</b>	Higher due to data sent to cloud	Lower due to local processing
<b>Security</b>	Centralized security controls	Localized security per device

➤ NOV / DEC 2022

Q7)

a) What do you mean by IoT Cloud ?And how IoT cloud can be used in home automation? [9]

**IoTCloud** refers to the integration of **Internet of Things (IoT)** devices with **Cloud Computing** platforms to enable data storage, processing, and remote control. It allows smart devices to connect, communicate, and be managed via the cloud.

Definition

Rest question repeated !!

b) Explain Architecture and Working of Kubernetes. [9 Marks]

**Kubernetes** is an open-source container orchestration platform used to **automate deployment, scaling, and management of containerized applications** (e.g., using Docker).

**Kubernetes Architecture:**

Kubernetes follows a **Master-Worker** architecture, as shown in the diagram.

**Master Node (Control Plane):**

Controls and manages the entire cluster.

- **Kubectcl**: CLI tool to interact with the Kubernetes cluster.
- **API Server**: Central hub that exposes the Kubernetes API; all communications happen through it.
- **Scheduler**: Assigns workloads (pods) to available worker nodes based on resource availability.
- **Controller-Manager**: Handles routine tasks like replication, endpoint updates, and failure handling.
- **etcd**: Key-value store used to persist cluster state/configuration.

**Worker Node:**

Executes the workloads (Pods).

- **Kubelet:** Agent that ensures containers are running in Pods as expected.
- **Kube-Proxy:** Manages network communication within the cluster and routes traffic.
- **Docker** (or other container runtime): Runs the actual containers within Pods.

#### Working of Kubernetes:

1. **User Deployment:** User submits deployment YAML via kubectl.
2. **API Server Receives Request:** It validates and passes the request to the Scheduler.
3. **Scheduler:** Decides which node will run the Pod.
4. **Kubelet on Worker Node:** Instructs Docker to run the container.
5. **Kube-Proxy:** Handles networking and exposes services.
6. **etcd:** Continuously updates and stores cluster state (nodes, pods, configs).

Kubernetes provides a robust, automated, and scalable system for running containerized applications, with clear separation of control (master) and execution (worker) responsibilities.

#### Q8

##### a) What are the future trends in cloud computing? Explain in brief. [9 Marks]

Cloud computing is constantly evolving with advancements in technologies, leading to newer trends that enhance efficiency, scalability, and security.

##### Future Trends in Cloud Computing:

###### 1. Serverless Computing:

- Developers can deploy code without managing servers.
- Resources are dynamically allocated.
- Example: AWS Lambda, Azure Functions.

###### 2. Edge Computing Integration:

- Data is processed closer to the source (e.g., IoT devices) to reduce latency.
- Enhances real-time decision-making.

###### 3. Multi-Cloud and Hybrid Cloud Strategies:

- Organizations use multiple cloud providers or a mix of private and public clouds.
- Increases flexibility, avoids vendor lock-in.

###### 4. AI and Machine Learning as a Service:

- Cloud platforms offer tools to build and deploy ML models.

- Example: Google Cloud AI, AWS SageMaker.

#### 5. Quantum Computing in Cloud:

- Future cloud services will provide access to quantum computing.
- Useful for complex simulations and cryptography.

#### 6. Cloud Security Enhancements:

- Focus on zero-trust security models, data encryption, and compliance tools.
- Security-as-a-service is gaining importance.

#### 7. Sustainable and Green Cloud:

- Cloud providers are shifting to energy-efficient data centers.
- Use of renewable energy and carbon-neutral targets.

#### 8. Cloud-Native Technologies:

- Rise in usage of microservices, containers (Docker), orchestration (Kubernetes).
- Ensures scalability and faster deployment.

#### 9. IoT and Cloud Convergence:

- Seamless integration of IoT devices with cloud platforms.
- Enables remote monitoring and control.

#### b) Differentiate between Distributed Cloud Computing Vs Edge Computing?[9]

➔ Repeated Question !!

---

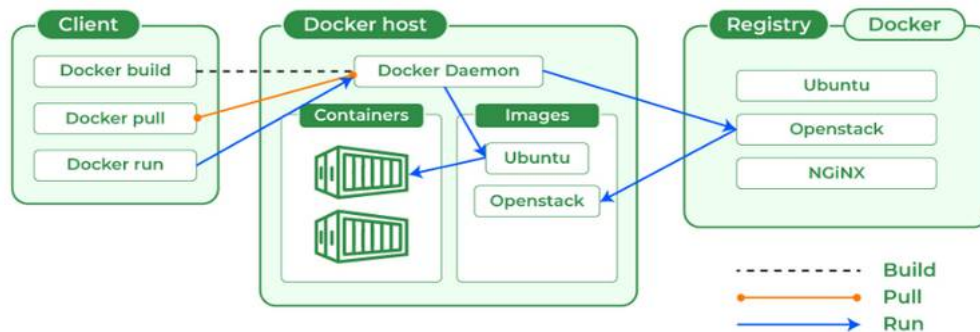
➤ MAY / JUN 2023

Q7)

#### a) Describe Client-Server Architecture of Docker [9 Marks]

Docker follows a **client-server architecture** where the **Docker client** communicates with the **Docker daemon (server)** to build, run, and manage containers.





## Components of Docker Architecture

1. **Docker Client**
  - Acts as the primary interface for users.
  - Sends commands like `docker run`, `docker build`, etc.
  - Communicates with the Docker daemon using REST API over UNIX socket or network.
2. **Docker Daemon (dockerd)**
  - Runs on the host machine.
  - Listens for Docker API requests.
  - Manages Docker objects like containers, images, volumes, and networks.
3. **Docker Images**
  - Read-only templates used to create containers.
  - Built using Dockerfile.
4. **Docker Containers**
  - Lightweight, executable instances of Docker images.
  - Isolated but share the host OS kernel.
5. **Docker Registry**
  - Storage and distribution system for Docker images (e.g., Docker Hub).
  - Clients can push or pull images from a registry.

## Client-Server Interaction Flow

1. User runs `docker` command from CLI (Client).
2. Command sent to Docker Daemon.
3. Daemon performs requested task (e.g., pull image, start container).
4. Results returned to client for display.

## Benefits of Docker's Architecture

- Decouples user interface from execution engine.
- Supports remote container management.
- Scalable and lightweight deployment.

**b) Explain Mobile Cloud in detail? [9]**

**Mobile Cloud Computing (MCC)** is a model that combines **cloud computing**, **mobile computing**, and **wireless networks** to bring rich computational resources to mobile users, network operators, and cloud providers.

---

**Definition:**

Mobile Cloud Computing refers to **running mobile applications on cloud servers** rather than locally on mobile devices. It allows mobile devices to **offload data storage, computing, and processing** to the cloud, improving performance and battery efficiency.

---

**Architecture:**

1. **Mobile Device (Client):**
    - User interacts with the app.
    - Sends request (like upload, compute, sync) to cloud servers.
  2. **Network (Internet/Wireless):**
    - Acts as a communication bridge between mobile and cloud.
  3. **Cloud Server:**
    - Processes the request.
    - Stores and manages data.
    - Sends results/output back to the mobile client.
- 

**Key Features:**

- **Resource Optimization:** Offloads heavy processing to the cloud.
  - **Platform Independence:** Apps can run across devices with different platforms (iOS/Android).
  - **Centralized Data Storage:** Data is stored and accessed from the cloud.
- 

**Advantages:**

- **Improved Performance:** Reduces load on mobile CPU and RAM.
- **Battery Efficiency:** Saves power by shifting intensive tasks to cloud.
- **Scalability:** Easily handles increased users or workloads.

- **Cost Reduction:** Less need for high-end hardware.
  - **Data Accessibility:** Access your data anytime, anywhere.
  - **Easy Updates:** Apps can be updated directly on the cloud, not every device.
- 

**Use Cases:**

- **Cloud Storage Apps:** Google Drive, Dropbox, iCloud
- **Mobile Cloud Gaming:** NVIDIA GeForce Now, Xbox Cloud
- **Mobile Healthcare Apps:** Real-time analysis using cloud AI
- **Mobile CRM/ERP Tools:** Salesforce on mobile

**Q8)**

**a) Differentiate between Distributed Cloud Computing Vs Edge Computing?**

→ already done !

**b) Explain the Concept of DevOps in Detail [9 Marks]**

**Definition:**

**DevOps** is a combination of **Development (Dev)** and **Operations (Ops)**. It is a **set of practices, tools, and cultural philosophies** that aims to **bridge the gap between software development and IT operations** to deliver applications and services faster, more efficiently, and with better quality.

**Core Objectives:**

- Automate and integrate the processes between software development and IT teams.
- Enable **continuous integration, continuous testing, and continuous delivery (CI/CD)**.
- Enhance **collaboration, speed, and reliability** in delivering software.

**Key Principles of DevOps:**

1. **Collaboration & Communication:** Development and operations teams work together throughout the lifecycle.
2. **Automation:** From code integration to deployment and testing.
3. **Continuous Integration (CI):** Developers merge code frequently.
4. **Continuous Delivery (CD):** Code changes are automatically tested and deployed.

5. **Monitoring & Feedback:** Real-time performance monitoring helps in quick fixes and updates.

#### **DevOps Lifecycle Phases:**

1. **Plan** – Define goals, requirements, and tasks.
2. **Develop** – Write code and review collaboratively.
3. **Build** – Compile and build the code.
4. **Test** – Automated testing ensures code quality.
5. **Release** – Deploy code to production environments.
6. **Deploy** – Deliver to users frequently.
7. **Operate** – Manage infrastructure and applications.
8. **Monitor** – Track performance and user feedback.

#### **Tools in DevOps:**

- **CI/CD:** Jenkins, GitLab CI, Travis CI
- **Version Control:** Git, GitHub
- **Configuration Management:** Ansible, Chef, Puppet
- **Containerization:** Docker, Kubernetes
- **Monitoring:** Prometheus, Nagios, Grafana

#### **Advantages of DevOps:**

- Faster software releases and updates
  - Improved collaboration between teams
  - High software quality and reliability
  - Reduced time to recover from failures
  - Better customer satisfaction
-

➤ NOV / DEC 2023

Q7)

a) Draw & Explain client - server architecture of docker?

**Docker** employs a **client-server architecture** that facilitates the development, deployment, and management of containerized applications. This architecture comprises several key components:

### 1. Docker Client

- **Role:** Acts as the primary user interface.
- **Functionality:** Sends commands (e.g., docker build, docker run) to the Docker daemon via a REST API over UNIX socket or network interface.
- **Flexibility:** Can communicate with daemons on the same host or remote systems.

### 2. Docker Daemon (dockerd)

- **Role:** Serves as the core engine of Docker.
- **Responsibilities:**
  - Builds, runs, and manages containers.
  - Handles Docker objects like images, networks, and volumes.
  - Listens for API requests from the Docker client and processes them accordingly.

### 3. Docker Registry

- **Purpose:** Stores Docker images.
- **Types:**
  - **Public Registries:** Such as Docker Hub, accessible to all users.
  - **Private Registries:** Hosted within an organization for internal use.
- **Operations:** Users can push images to or pull images from the registry.

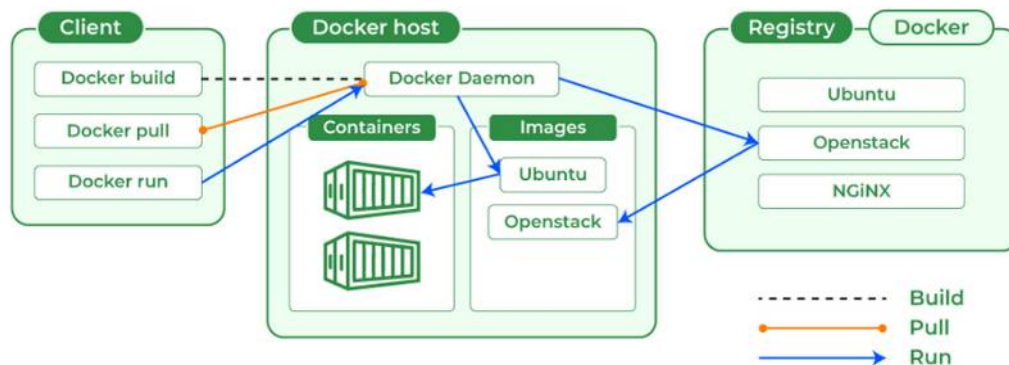
### 4. Docker Objects

- **Images:** Read-only templates used to create containers.

- **Containers:** Executable instances of images, encapsulating application code and dependencies.
- **Networks:** Facilitate communication between containers.
- **Volumes:** Persist data generated by and used by Docker containers.

**Interaction Flow:**

1. **Command Initiation:** The user issues a command via the Docker client.
2. **Request Transmission:** The client sends this request to the Docker daemon.
3. **Processing:** The daemon processes the request, interacting with the necessary Docker objects.
4. **Execution:** The daemon executes the required operations, such as running a container or pulling an image.
5. **Response:** The daemon sends feedback to the client, completing the interaction cycle



**b) Explain Multimedia Cloud in detail? [9 Marks]**

**Multimedia Cloud** refers to the delivery, storage, processing, and management of multimedia content—such as images, audio, video, and animations—using cloud computing technologies. It enables users and applications to access and manipulate multimedia data over the internet without relying on local resources.

**Key Features of Multimedia Cloud:**

1. **On-Demand Access:**  
Users can upload, store, and access multimedia content anytime and anywhere through the internet.
2. **Scalability:**  
Multimedia Cloud can handle large volumes of data and many users simultaneously by dynamically allocating resources.
3. **Cost-Effective:**  
It eliminates the need for expensive hardware and maintenance since cloud providers manage infrastructure.
4. **Flexibility:**  
Supports diverse multimedia applications such as streaming services, video conferencing, virtual reality, and online gaming.
5. **High Availability and Reliability:**  
Cloud providers ensure that multimedia content is always accessible with backup and failover systems.

**How Multimedia Cloud Works:**

- **Content Storage:** Multimedia files are stored on distributed servers in the cloud.
- **Content Delivery:** Uses Content Delivery Networks (CDNs) to stream or deliver content efficiently.
- **Processing:** Cloud resources handle complex multimedia processing tasks like encoding, rendering, or real-time analytics.
- **User Interaction:** Users access multimedia through apps or browsers, receiving fast and smooth experiences.

**Applications of Multimedia Cloud:**

- Video streaming platforms (e.g., Netflix, YouTube)
- Online gaming
- Video conferencing tools (e.g., Zoom, Microsoft Teams)
- Digital media archives
- Virtual and augmented reality experiences

Multimedia Cloud combines the power of cloud computing with multimedia management to provide scalable, cost-effective, and high-quality multimedia services accessible worldwide. It plays a vital role in modern communication, entertainment, and collaboration

Q8)

- a) **Differentiate between Distributed Cloud Computing Vs Edge Computing? [9]**  
Already done !
- b) **Describe the concept of DevOps in detail?**  
Already done !

**MAY / JUN 2024**

**Q7)**

**a) Explain the mobile cloud computing?**

**ALREADY DONE !**

**b) Explain docker with its Architecture ? [6]**

**already done !!!**

**c) Explain the application of IOT and cloud in your home? [5]**

**already done !!!**

**Q8)**

**a) What is Energy aware cloud computing? Explain in details?**

**→ Already done (repeated )**

**b) Explain Container & Kubernetes in detail? [6]**

**Container:**

- A container is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, libraries, and settings.
- Containers provide process and filesystem isolation using OS-level virtualization, allowing multiple containers to run on the same host without interfering with each other.
- Unlike virtual machines, containers share the host OS kernel, making them more efficient and faster to start.
- Popular container platforms include Docker and Podman.

**Kubernetes:**

- Kubernetes is an open-source container orchestration platform developed by Google to automate the deployment, scaling, and management of containerized applications.
- It manages clusters of containers, handling tasks like load balancing, automatic scaling, rolling updates, and self-healing (restarting failed containers).
- Key components include:
  - **Pods** (smallest deployable units containing one or more containers),
  - **Nodes** (machines running containers),
  - **Master node** (controls the cluster with API server, scheduler, controller manager),



- **Services** (to expose applications).
- Kubernetes enables running applications reliably at scale across many hosts in a distributed environment.

**c) Explain Distributed Cloud Computing? [5]**

- Distributed cloud computing refers to a cloud computing model where cloud services are distributed across multiple geographic locations, including different data centers or edge locations.
- Instead of all resources being centralized in a single cloud region, workloads and services are run closer to end-users or specific locations to reduce latency, improve performance, and meet regulatory or data sovereignty requirements.
- It integrates public cloud, private cloud, and edge computing resources, providing a seamless experience while maintaining centralized management.
- Benefits include improved fault tolerance, faster data processing near the source, and better compliance with data laws.
- Examples include content delivery networks (CDNs), edge computing platforms, and hybrid cloud architectures.

---

➤ **NOV / DEC 2024**

**FULLY REPEATED QUESTIONS !!!!**