

Cloud Computing for Data Science (315327)

Model Question Paper — Set A · Model Answers

Q.1

a) Define cloud computing with examples.

2 M

Cloud computing (NIST): a model for enabling convenient, **on-demand network access to a shared pool of configurable computing resources** (networks, servers, storage, applications and services) that can be **rapidly provisioned and released** with minimal management effort or service-provider interaction. User pays only for what is used.

Examples: Gmail / Google Drive (SaaS), Google Colab, Amazon EC2 and Amazon S3 (IaaS), Google App Engine (PaaS).

(Definition 1 M, two examples 1 M)

b) Four characteristics of cloud computing.

2 M

1. **On-demand self-service** — user gets servers / storage automatically, without human interaction with the provider.
2. **Broad network access** — services available over the network from any device (mobile, laptop, PC).
3. **Resource pooling** — provider's resources are pooled and shared by many users (multi-tenancy).
4. **Rapid elasticity** — resources scale out / in quickly as demand changes.
5. **Measured service** — usage is metered; pay-as-you-go billing.

(Any four, ½ M each)

c) Virtualization and its types.

2 M

Virtualization is a technique of creating a **virtual (software-based) version** of a computing resource — server, operating system, storage device, network or data — so that one physical resource can be shared as many isolated virtual resources (e.g. many virtual machines on one server, managed by a hypervisor).

Types: server, network, storage, desktop, application and data virtualization.

(Definition 1 M, any two types 1 M)

d) Block storage vs object storage.

2 M

Point	Block storage	Object storage
Unit of data	Fixed-size blocks with addresses	Objects = data + metadata + unique ID
Access	Mounted as a disk volume by the OS (iSCSI / FC)	HTTP REST API (PUT / GET) by key
Metadata	No metadata with the block	Rich, user-defined metadata

Best for	Databases, VM boot disks (low latency)	Images, videos, backups, data lakes (huge scale)
Example	Amazon EBS	Amazon S3

(Any two points, 1 M each)

e) Features of key-value databases.

2 M

1. **Simple data model** — data stored as (key, value) pairs; value can be any type (string, JSON, binary).
2. **Very fast access** — GET / PUT by key in near constant time; often kept in memory (Redis).
3. **Schema-less** — no fixed table structure; each value can differ.
4. **Horizontal scalability** — keys are partitioned (sharded) across many servers.
5. **High availability** — data replicated across nodes. Examples: Amazon DynamoDB, Redis, Riak.

(Any four, ½ M each)

f) Features of Azure Machine Learning Studio.

2 M

1. **Designer** — drag-and-drop, low-code canvas to build ML pipelines.
2. **Automated ML (AutoML)** — automatically tries algorithms and hyper-parameters and picks the best model.
3. **Notebooks** — Jupyter notebooks on managed compute instances for code-first work.
4. **Managed compute, model registry and endpoints** — train on clusters and deploy models as real-time / batch endpoints.

(Any two, 1 M each)

g) Components of Amazon SageMaker.

2 M

1. **SageMaker Studio / Notebook instances** — web IDE with Jupyter notebooks.
2. **Ground Truth** — data labelling; **Data Wrangler** — data preparation.
3. **Training jobs** with built-in algorithms and **Automatic Model Tuning** (hyper-parameter tuning).
4. **Endpoints** (real-time inference) and **Batch Transform**.
5. **Model Monitor** — watches deployed models for drift; **Autopilot** — AutoML.

(Any four, ½ M each)

Q.2

a) Cloud computing architecture.

4 M

Cloud computing architecture is divided into two parts — **front end** (client side) and **back end** (provider side) — connected through a **network (Internet)**.

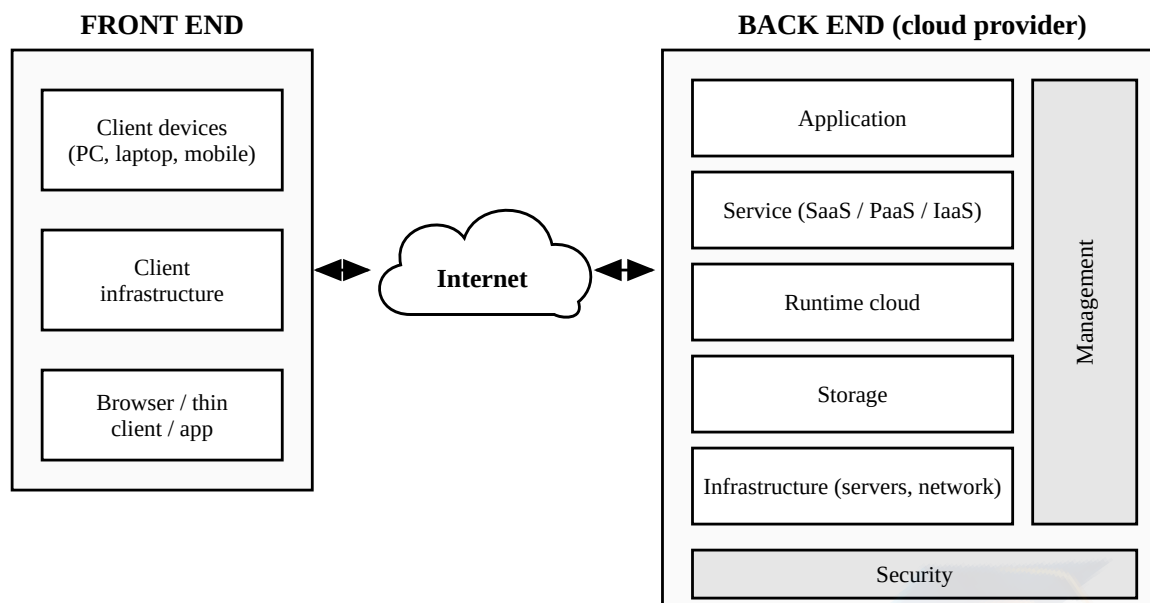


Fig.: Architecture of cloud computing (front end and back end)

- 1. Front end:** the client part used by the user — client devices (PC, mobile) and interfaces such as a web browser, thin client or mobile app through which cloud services are accessed.
- 2. Back end:** managed by the cloud provider — *application* (the service the user uses), *service* layer (SaaS / PaaS / IaaS), *runtime cloud* (execution environment, VMs), *storage* and *infrastructure* (servers, network devices, virtualization software).
- 3. Management:** allocates resources, monitors, schedules and balances load in the back end.
- 4. Security:** protects data and services (authentication, encryption, firewalls); **Network / Internet** connects both ends and gives cloud-based delivery of services.

(Diagram 2 M, explanation 2 M)

b) Batch data vs streaming data in ML.

4 M

Point	Batch data	Streaming data
Meaning	Data collected over a period and processed together as one large batch	Data generated continuously and processed record-by-record as it arrives
Latency	High — results after minutes / hours	Low — results in milliseconds / seconds
Data size	Large, bounded data set	Unbounded, never-ending flow of small events
ML use	Offline model training, periodic retraining, bulk predictions	Real-time predictions, online learning, live dashboards
Tools	Hadoop MapReduce, Apache Spark, AWS Batch, AWS Glue	Apache Kafka, Amazon Kinesis, Spark Streaming, Apache Flink, Google Pub/Sub
Example	Retrain a sales-forecast model every night	Credit-card fraud detection while the payment happens

(Any four points, 1 M each)

c) Need and applications of virtualization.

4 M

Need for virtualization:

1. **Better hardware utilisation** — physical servers are often only 10–20 % used; many VMs on one server use it fully.
2. **Cost saving** — fewer physical servers → less power, cooling, space and maintenance.
3. **Isolation and security** — each VM is isolated; a crash in one does not affect others.
4. **Fast provisioning and flexibility** — new VMs from templates in minutes; easy backup, migration and disaster recovery; foundation of cloud computing (IaaS).

Applications:

- **Server consolidation** in data centres and cloud IaaS (Amazon EC2 VMs).
- **Running multiple OS** (Windows + Linux) on one machine for development and testing.
- **Virtual desktops (VDI)** for colleges / offices; **virtual labs** for practicals.

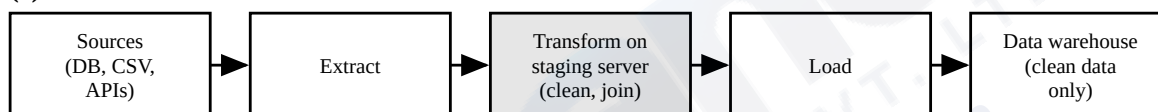
(Need 2 M, any two applications 2 M)

d) Support for ETL / ELT pipelines while selecting a cloud ML platform.

4 M

An ML model is only as good as its data. Data comes from many sources (databases, logs, CSV, APIs) and must be **extracted, cleaned / transformed and loaded** before training. So a good cloud ML platform should provide built-in **ETL / ELT pipeline support**.

(a) ETL — Extract → Transform → Load



(b) ELT — Extract → Load → Transform

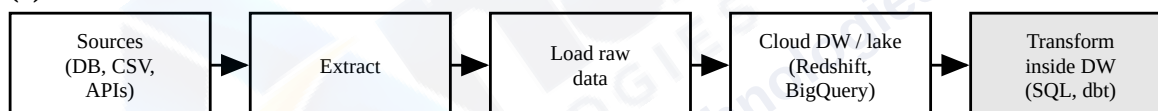


Fig.: ETL vs ELT pipeline

1. **Connectors to data sources** — easy extraction from RDBMS, object storage, streaming sources (e.g. AWS Glue, Azure Data Factory, Google Cloud Dataflow).
2. **Transformation tools** — cleaning, joining, feature engineering at scale (Spark, SQL in Redshift / BigQuery, SageMaker Data Wrangler).
3. **ETL or ELT choice** — ETL transforms before loading (good for structured, sensitive data); ELT loads raw data into a cloud warehouse / data lake and transforms there using its compute (faster for big data).
4. **Automation and scheduling** — pipelines run on schedule or on new data, so models are always trained on fresh, consistent data; integration with feature store and training jobs.

(Explanation 2 M, any two support points / diagram 2 M)

Q.3

a) Software as a Service (SaaS).

4 M

SaaS is a cloud service model in which a complete **application is delivered over the Internet** on a subscription / pay-per-use basis. The provider manages everything — application, data storage, runtime, OS, servers and network; the user only uses the software through a browser or app.

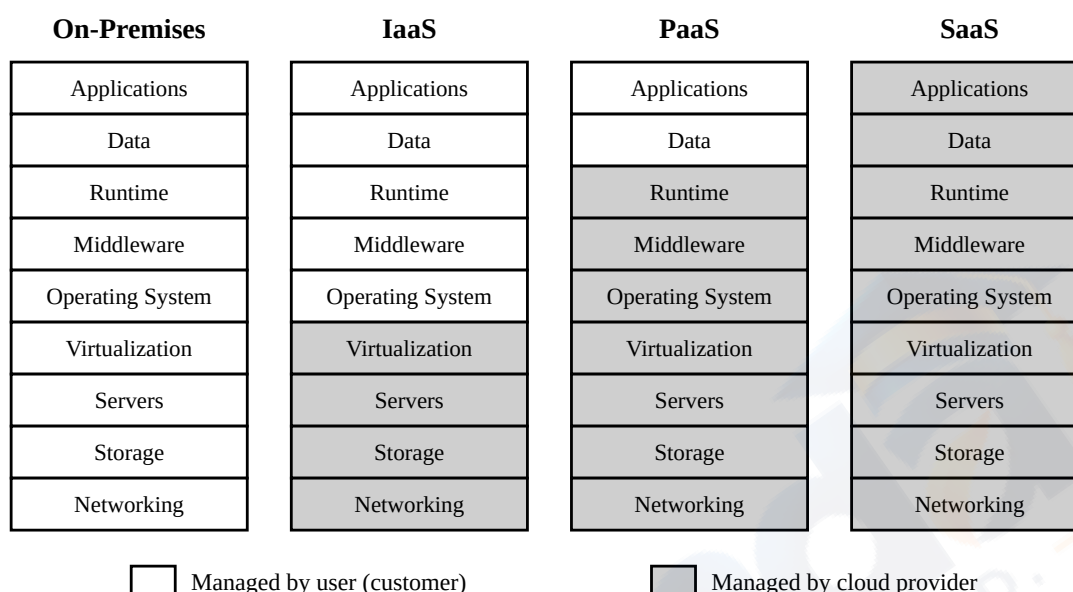


Fig.: Division of management responsibility in IaaS, PaaS and SaaS

Characteristics:

- Accessed through a web browser from any device; no installation.
- Multi-tenant — one application instance serves many customers.
- Provider handles updates, patches and backup automatically.
- Subscription (monthly / yearly) or free with ads.

Examples: Gmail, Google Docs, Microsoft 365, Salesforce CRM, Zoom, Dropbox.

Advantages: low cost, no maintenance, accessible anywhere, quick to start. **Disadvantages:** data stored with provider (privacy), needs Internet, limited customisation.

(Concept 1 M, characteristics 1 M, examples 1 M, advantages / disadvantages 1 M)

b) Advantages of cloud computing for machine learning.

4 M

1. **On-demand, scalable compute** — powerful CPUs, GPUs and TPUs can be rented for training and released after use; scale up for big data sets.
2. **Pay-as-you-go cost** — no need to buy expensive GPU servers; pay only for training hours and storage used.
3. **Managed ML services** — platforms like Amazon SageMaker, Azure ML and Google Vertex AI give notebooks, AutoML, training, tuning and deployment in one place.
4. **Large, cheap storage close to compute** — data lakes (Amazon S3, Google Cloud Storage) and warehouses (Redshift, BigQuery) hold huge data sets for training.

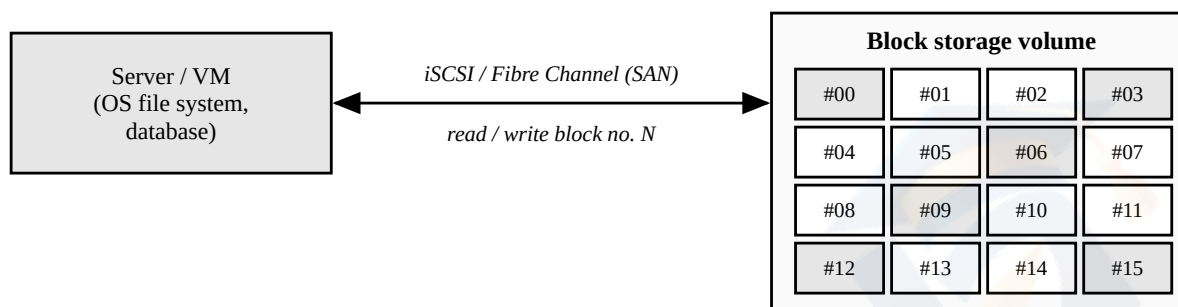
5. **Easy deployment and collaboration** — models deployed as API endpoints in minutes; teams share notebooks and data from anywhere.
6. **Pre-trained AI services** — vision, speech and NLP APIs can be used without building models.

(Any four advantages, 1 M each)

c) Block-based cloud storage system.

4 M

In **block-based storage**, data is split into **fixed-size chunks called blocks**. Each block gets a **unique address** and is stored separately; the storage system keeps no extra metadata. When data is read, the blocks are put together again using their addresses. The volume appears to the server as a **raw disk**, which the OS formats with its own file system.



Data split into fixed-size blocks, each with a unique address; no metadata stored with the block
e.g. Amazon EBS, Azure Managed Disks, Google Persistent Disk

Fig.: Block-based storage system

- Storage is delivered over a **Storage Area Network (SAN)** using iSCSI or Fibre Channel, or as a cloud block volume attached to a VM.
- **Low latency, high IOPS** — any block can be read / changed directly, so it is ideal for databases, transactional systems and VM boot disks.
- A volume is normally attached to **one server / VM** at a time; snapshots give backup.
- **Examples:** Amazon EBS, Azure Managed Disks, Google Persistent Disk.
- **Limitations:** costlier than object storage, limited metadata, not ideal for sharing files among many users.

(Explanation 2 M, diagram 2 M)

d) Jupyter Notebook and its workflow.

4 M

Jupyter Notebook is an open-source, browser-based tool to create documents (**.ipynb**) that combine **live code, its output, charts, equations and markdown text**. It is the most used tool in data science and is offered on the cloud as Google Colab, SageMaker Studio notebooks and Azure ML notebooks.

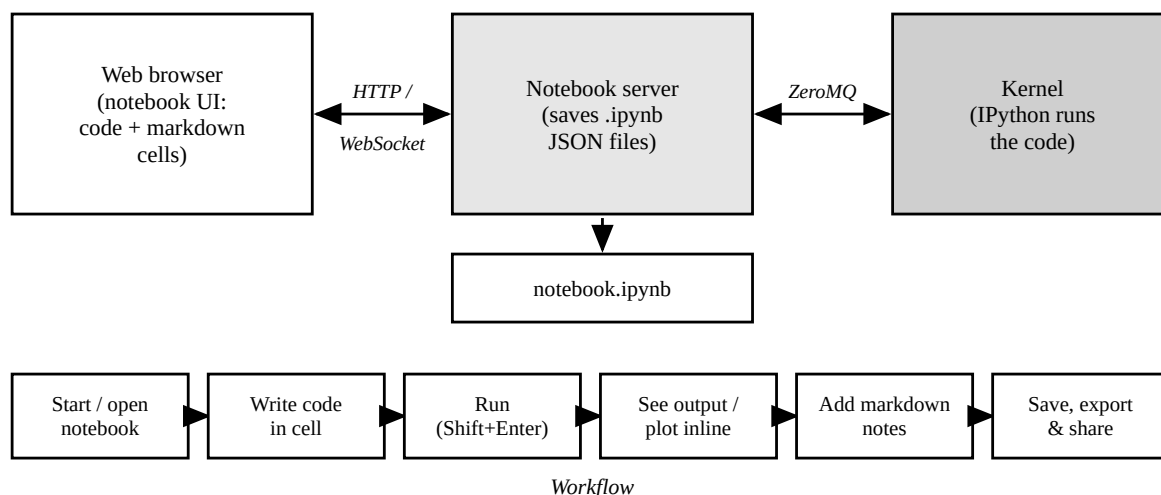


Fig.: Jupyter Notebook architecture and workflow

Components: browser UI with *cells* (code, markdown, raw), notebook server (stores the .ipynb JSON file) and *kernel* (e.g. IPython) that actually runs the code.

Workflow:

1. Start the server (`jupyter notebook`) or open a cloud notebook; create a new notebook and select a kernel (Python 3).
2. Write code in a code cell (load data with pandas, train a model) and run it with **Shift + Enter**; output, tables and plots appear just below.
3. Add markdown cells to explain steps; re-run / edit cells to experiment (variables stay in kernel memory).
4. Save (auto-checkpoints), then export as HTML / PDF / .py or share through GitHub / cloud storage.

(Explanation 2 M, workflow 2 M)

Q.4

a) Roles in Service Oriented Architecture (SOA).

4 M

SOA is a design style in which an application is built from loosely coupled, reusable **services** that communicate over a network through standard interfaces (web services — SOAP / WSDL / UDDI, or REST). Cloud services are built on SOA.

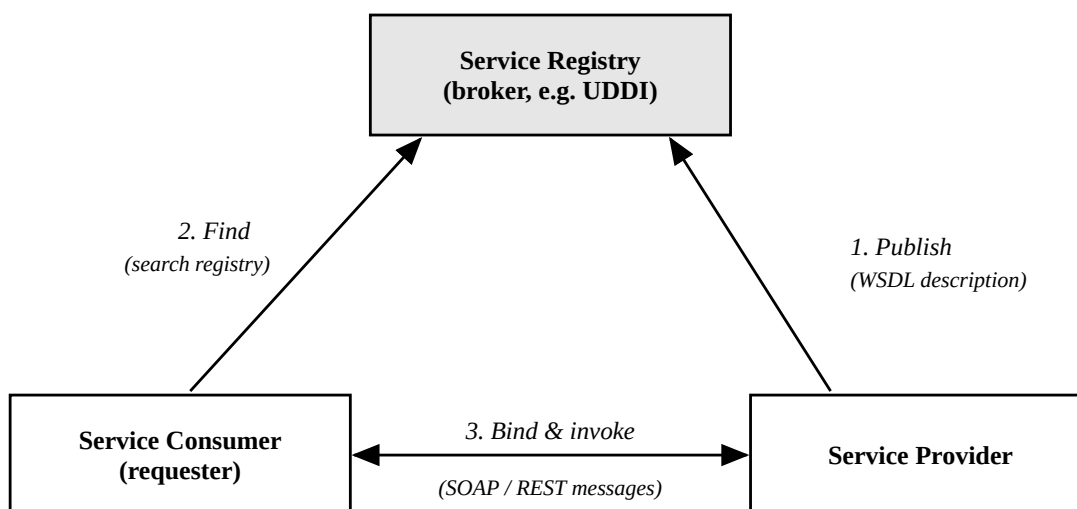


Fig.: Roles in Service Oriented Architecture (publish – find – bind)

1. **Service provider:** creates the service, hosts it on its server, and **publishes** its description (WSDL — what it does, inputs / outputs, address) in the service registry. It is responsible for availability and security of the service.
2. **Service consumer (requester):** the client application that needs the service. It **finds** the service in the registry, then **binds** to the provider and invokes it by sending request messages and receiving responses.
3. **Service registry (broker):** a directory (e.g. UDDI) where providers publish and consumers search for services.
4. Thus SOA works on the **publish – find – bind** cycle; e.g. a travel website (consumer) uses a payment-gateway service (provider).

(Diagram 1 M, roles 3 M)

b) Public vs private vs hybrid cloud.

4 M

Point	Public cloud	Private cloud	Hybrid cloud
Ownership	Owned and run by a third-party provider	Used by one organisation (on-premises or hosted)	Combination of private and public
Users	General public, many tenants	Single organisation only	One organisation using both
Cost	Low, pay-as-you-go, no capital cost	High — own hardware and staff	Moderate
Security / control	Less control; shared infrastructure	Highest control and security	Sensitive data private, rest public
Scalability	Very high (virtually unlimited)	Limited to own capacity	High — cloud bursting to public
Examples	AWS, Microsoft Azure, Google Cloud	VMware / OpenStack private cloud of a bank	Bank: core data private, website on AWS

(Any four points, 1 M each)

c) Working of Amazon Redshift.

4 M

Amazon Redshift is a fully managed, petabyte-scale **cloud data warehouse** on AWS. It uses **massively parallel processing (MPP)** and **columnar storage** to run SQL analytics fast.

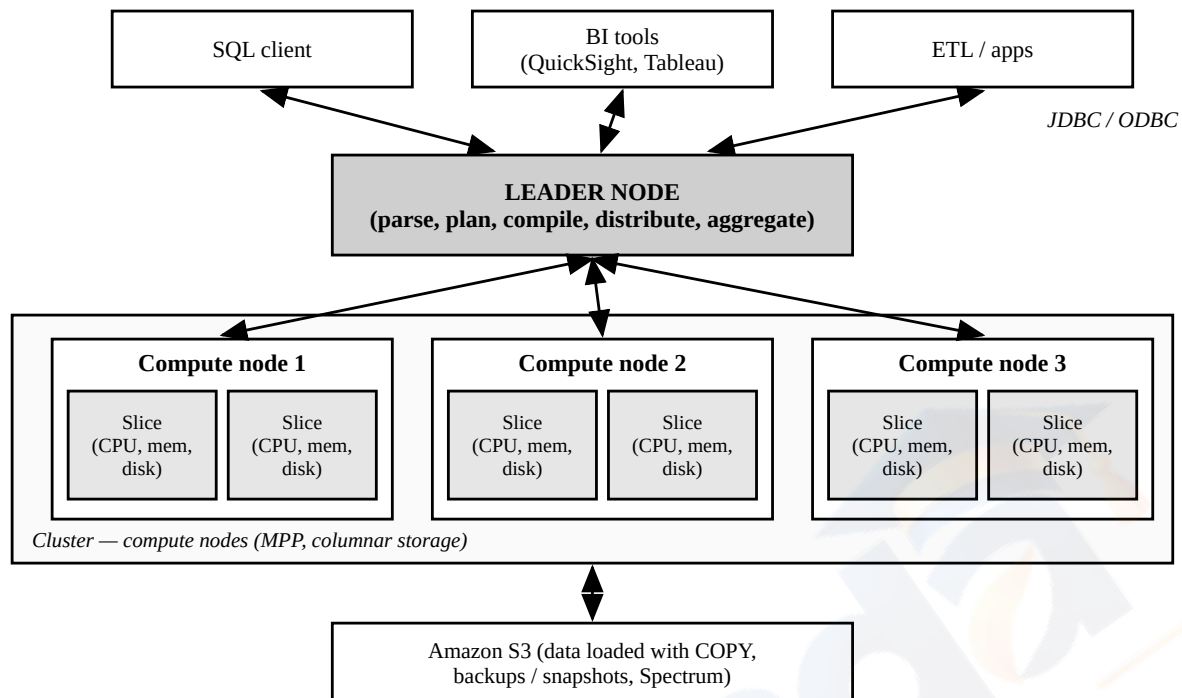


Fig.: Amazon Redshift architecture

1. Client tools (SQL client, BI tools) connect through **JDBC / ODBC** and send SQL queries to the **leader node**.
2. The **leader node** parses the query, makes an execution plan, compiles code and distributes it to the compute nodes.
3. Each **compute node** is divided into **slices** (own CPU, memory and disk); slices work in parallel on their part of the data, stored column-wise and compressed.
4. Compute nodes return intermediate results; the leader node aggregates them and returns the final result to the client. Data is loaded from Amazon S3 using the **COPY** command.

(Diagram 2 M, working 2 M)

d) Benefits and limitations of ML in the cloud.

4 M

Benefits:

1. Scalable compute (GPU / TPU) on demand for training large models.
2. Pay-per-use; no capital investment in hardware.
3. Managed tools — notebooks, AutoML, training, deployment, monitoring in one platform.
4. Easy deployment as APIs, collaboration and access from anywhere; pre-trained AI services available.

Limitations:

1. Data privacy, security and compliance concerns when sensitive data leaves the organisation.
2. Running cost can grow quickly (GPU hours, storage, data-transfer / egress charges).
3. Vendor lock-in — pipelines tied to one provider's services are hard to move.

4. Needs reliable high-speed Internet; latency for very large data uploads; less low-level control.

(Benefits 2 M, limitations 2 M)

e) Scale-up and scale-out training.

4 M

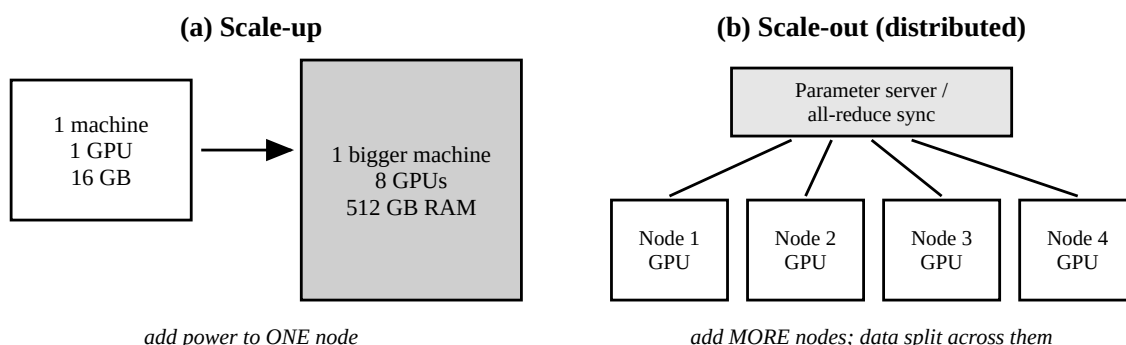


Fig.: Scale-up (vertical) vs scale-out (horizontal) training

Scale-up (vertical scaling): training is moved to a **bigger single machine** — more GPUs, more GPU memory, more RAM / CPU (e.g. from 1 GPU to an 8-GPU instance). Simple — little code change; but limited by the largest machine available and a single point of failure.

Scale-out (horizontal scaling / distributed training): training is spread over **many machines (nodes)** working in parallel.

- **Data parallelism** — each node gets a different part of the data and a copy of the model; gradients are synchronised (parameter server or all-reduce).
- **Model parallelism** — a very large model is split across nodes / GPUs.
- Supported by SageMaker distributed training, Horovod, PyTorch DDP, TensorFlow distribution strategies.
- Almost unlimited scaling and fault tolerance, but needs fast networking and more complex setup.

A cloud ML platform that supports both lets the team start small and grow as data and models grow.

(Scale-up 1½ M, scale-out 1½ M, diagram 1 M)

Q.5

a) Types of virtualization.

6 M

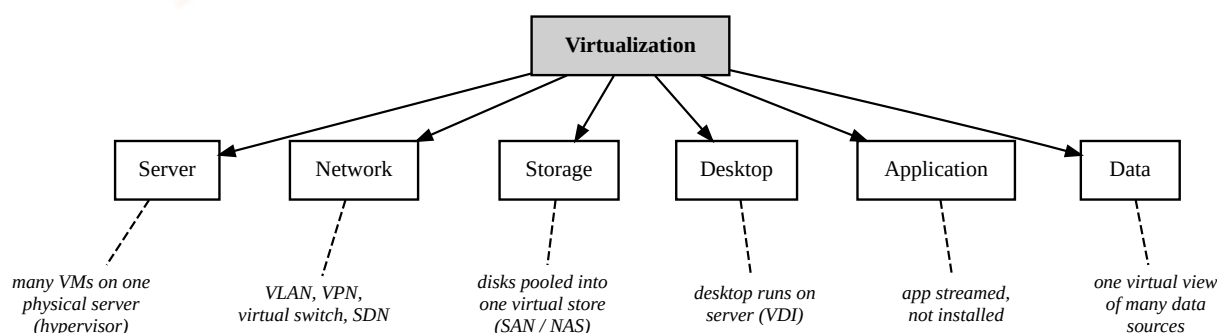


Fig.: Types of virtualization

1. **Server virtualization:** one physical server is divided into many isolated **virtual machines** by a hypervisor; each VM runs its own OS. Types: full, para- and hardware-assisted virtualization. e.g. VMware ESXi, Hyper-V, KVM.
2. **Network virtualization:** physical network resources (switches, routers, links) are combined / divided into independent **virtual networks** using VLAN, virtual switches, SDN and overlays (VXLAN).
3. **Storage virtualization:** many physical disks / storage arrays are **pooled** and presented as one logical storage (SAN, NAS); simplifies backup and allocation.
4. **Desktop virtualization (VDI):** user's desktop OS runs on a central server; users access it remotely from any device (e.g. Citrix, VMware Horizon, Amazon WorkSpaces).
5. **Application virtualization:** application runs in an isolated environment / is streamed from a server without being installed on the client (e.g. Microsoft App-V, VMware ThinApp).
6. **Data virtualization:** data from many different sources is shown as **one virtual, unified view** without copying it (e.g. Denodo, TIBCO Data Virtualization).

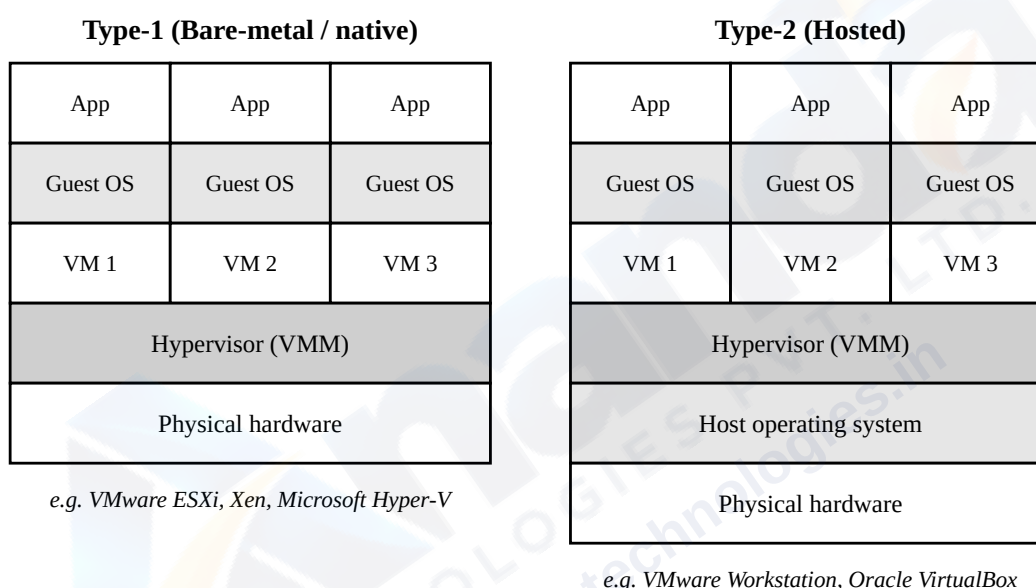


Fig.: Type-1 (bare-metal) and Type-2 (hosted) hypervisor

(Six types 1 M each (diagram included))

b) Amazon SageMaker — components and workflow.

6 M

Amazon SageMaker is a fully managed AWS service to **build, train and deploy machine learning models** at scale. It removes the work of managing servers for each step of the ML life cycle.

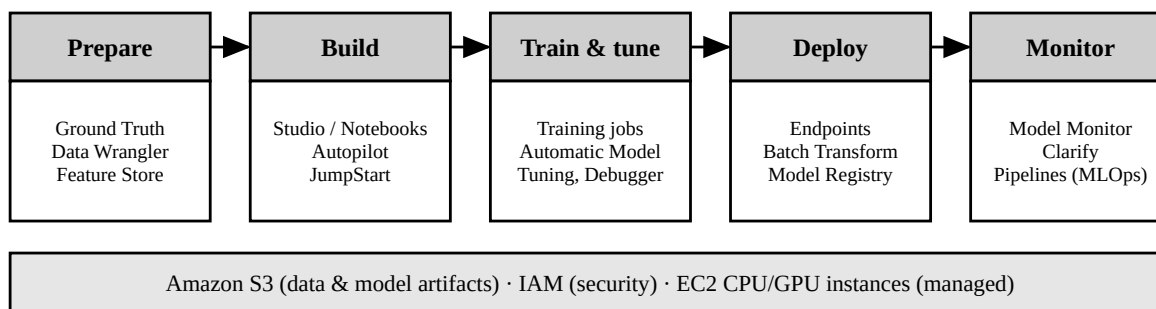


Fig.: Amazon SageMaker — components across the ML workflow

Components (by workflow stage):

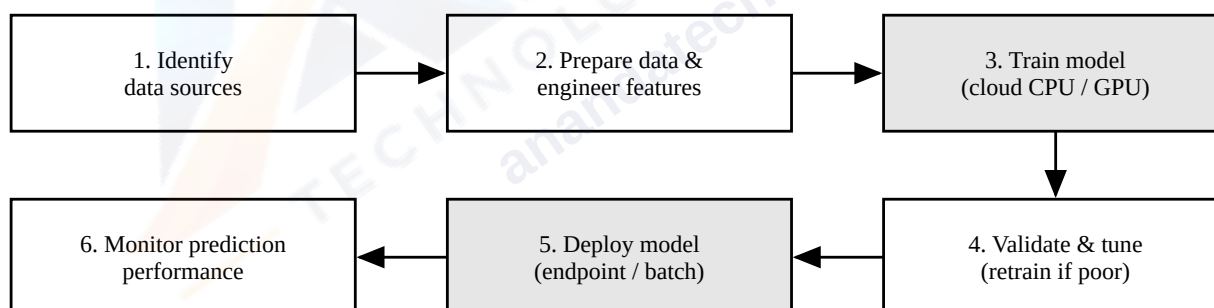
- 1. Prepare:** *Ground Truth* (data labelling with human workers / active learning), *Data Wrangler* (visual data preparation), *Feature Store* (store and share features).
- 2. Build:** *SageMaker Studio* — web IDE with Jupyter notebooks; *Autopilot* — AutoML; *JumpStart* — pre-built models and solutions.
- 3. Train and tune:** managed *training jobs* on CPU / GPU instances with built-in algorithms (XGBoost, linear learner, image classification) or own TensorFlow / PyTorch / scikit-learn code; *Automatic Model Tuning* for hyper-parameters; *Debugger* and *Experiments*.
- 4. Deploy:** one-click *real-time endpoints* with auto-scaling, *Batch Transform* for bulk predictions, *Model Registry* for versions.
- 5. Monitor:** *Model Monitor* detects data / model drift; *Clarify* checks bias and explains predictions; *Pipelines* automate MLOps.

Characteristics: fully managed, pay-per-use, scalable distributed training, data in Amazon S3, secured with IAM and VPC.

(Introduction 1 M, diagram 1 M, components / workflow 4 M)

c) Steps to train an ML project in the cloud.

6 M



If monitoring shows drift or falling accuracy, go back to step 2 and retrain

Fig.: Steps to train a machine learning project in the cloud

- 1. Identify data sources:** find where the required data is — databases, logs, IoT sensors, files, public data sets — and bring it into cloud storage (Amazon S3, Google Cloud Storage) or a warehouse (Redshift, BigQuery) using ETL / ELT pipelines.
- 2. Prepare data and engineer features:** clean missing / wrong values, remove duplicates, encode categories, scale values, create useful features; split into training, validation and test sets (tools: Data Wrangler, Dataflow, notebooks).

3. **Train the model:** choose the algorithm / framework (scikit-learn, TensorFlow, PyTorch) and launch a managed training job on suitable CPU / GPU instances; use scale-out for big data.
4. **Validate and tune:** evaluate on validation data (accuracy, precision, recall, RMSE); tune hyper-parameters (automatic model tuning); repeat until results are acceptable; register the best model.
5. **Deploy:** host the model as a real-time endpoint / API or run batch predictions; enable auto-scaling.
6. **Monitor:** track prediction quality, data drift, latency and cost; retrain with new data when performance drops.

(Diagram 1 M, steps 5 M)

Q.6

a) GPU as a Service (GPUaaS).

6 M

GPUaaS is a cloud service in which **GPU-equipped servers are rented on demand** over the Internet, so users can run GPU-heavy work (deep learning, rendering) without buying GPUs.

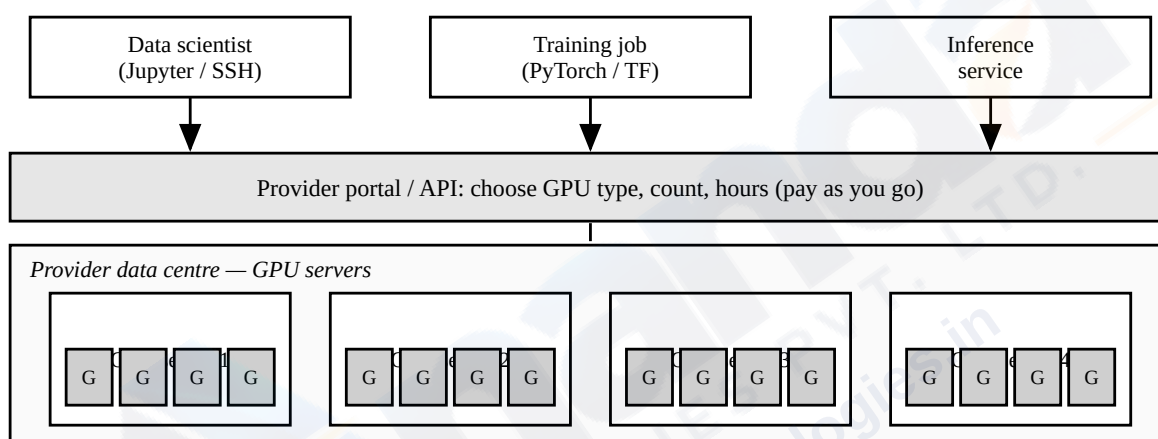


Fig.: GPU as a Service (GPUaaS)

(i) Types of GPUaaS models:

1. **On-demand (pay-as-you-go)** — GPU instances billed per second / hour; start and stop any time.
2. **Reserved / dedicated** — GPUs reserved for 1–3 years or a dedicated bare-metal server at lower hourly rate; for steady workloads.
3. **Spot / preemptible** — spare GPU capacity at large discount, but can be taken back by the provider; good for fault-tolerant training.
4. **Serverless / managed GPU** — user only submits a job or model; the platform allocates GPUs (e.g. managed training jobs, serverless inference). GPUs may be full physical GPUs or shared virtual GPUs (vGPU).

(ii) Use cases: training deep-learning models (CNN, transformers, LLMs); real-time inference; image / video processing and computer vision; scientific simulation and big-data analytics; 3-D rendering and gaming; virtual desktops for designers.

(iii) Providers: Amazon Web Services (EC2 P- and G-series instances), Google Cloud (A100 / H100 GPUs, also TPUs), Microsoft Azure (NC / ND-series VMs), Oracle Cloud, IBM Cloud,

NVIDIA DGX Cloud, Lambda, Paperspace.

(Models 2 M, use cases 2 M, four providers 2 M)

b) Pre-tuned AI services and monitoring prediction performance.

6 M

Pre-tuned AI services are ready-made AI models that the cloud provider has **already trained and tuned on very large data sets**, offered through simple APIs. The developer sends data and gets the result — no data collection, training or ML expertise needed.

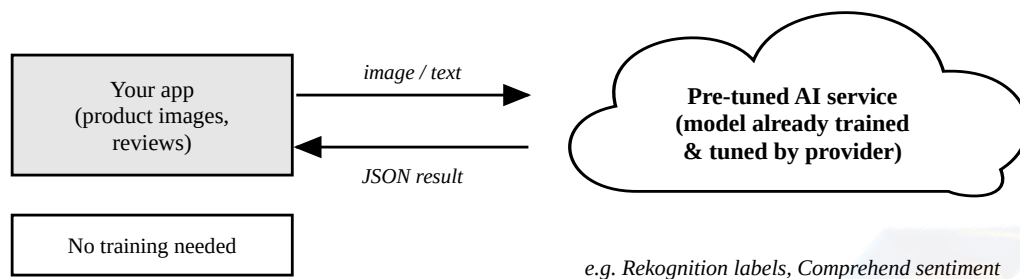


Fig.: Using a pre-tuned (pre-trained) AI service

Examples:

- **Vision:** Amazon Rekognition, Google Cloud Vision API, Azure AI Vision — object / face / text detection in images.
- **Language:** Amazon Comprehend, Google Natural Language API — sentiment, entities; Amazon Translate / Google Translation.
- **Speech:** Amazon Transcribe (speech-to-text), Amazon Polly (text-to-speech), Azure AI Speech.
- **Others:** Amazon Lex / Dialogflow chatbots, Amazon Textract (documents), Amazon Personalize (recommendations).

Features: fast to integrate, pay per API call, auto-scaling, continuously improved by provider; some allow small customisation (e.g. custom labels).

Monitoring prediction performance:

1. **Capture** inputs and predictions of the deployed model / service (logging, data capture).
2. **Track metrics** — accuracy, precision / recall against ground truth when actual results arrive; latency, error rate and throughput.
3. **Detect drift** — compare live data with training baseline (data drift) and check if the input–output relationship has changed (concept drift).
4. **Alerts and action** — dashboards and alarms (CloudWatch, SageMaker Model Monitor, Azure ML data-drift monitor, Vertex AI Model Monitoring); retrain or roll back when quality falls.

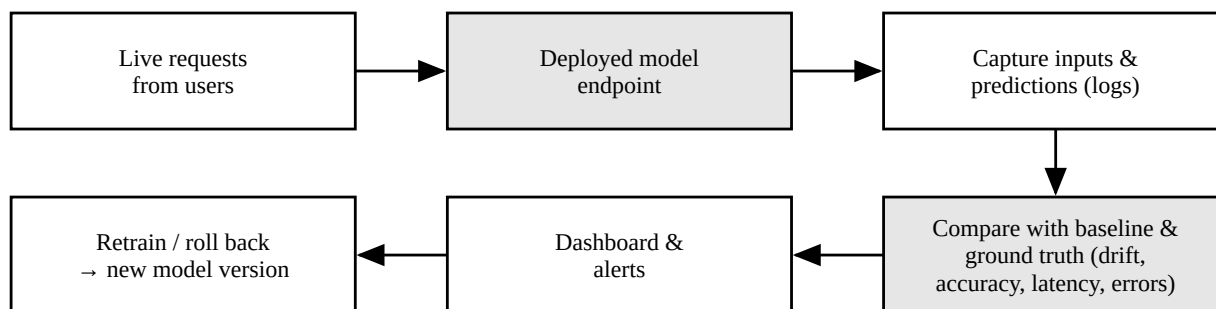


Fig.: Monitoring prediction performance of a deployed model

(Pre-tuned services with examples 3 M, monitoring 3 M)

c) Use cases of cloud storage.

6 M

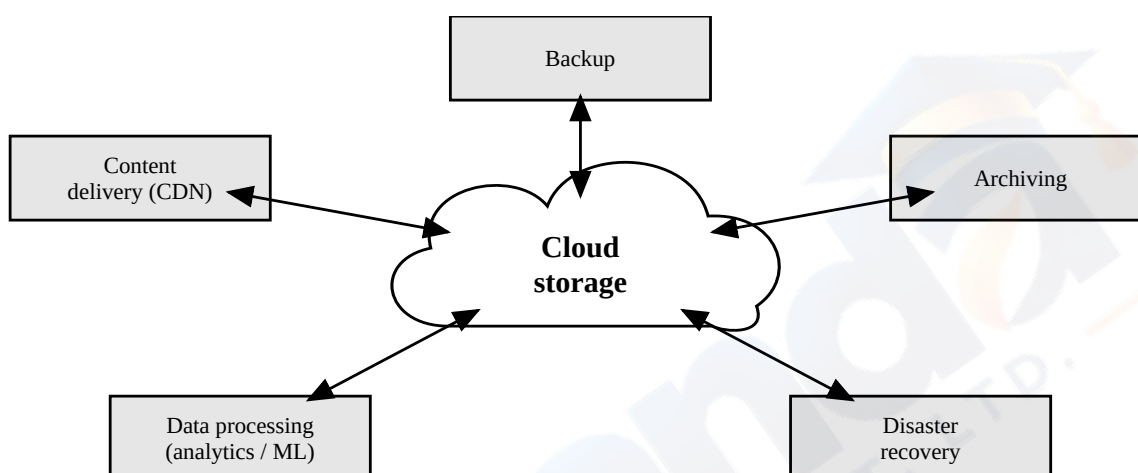
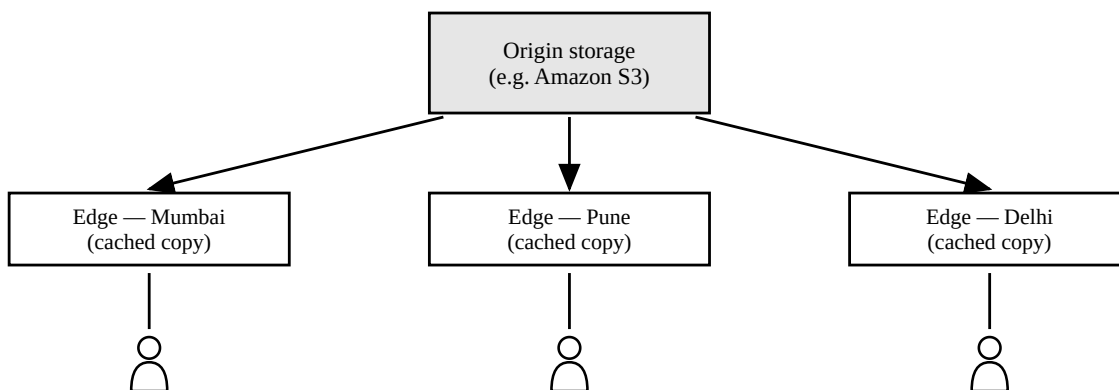


Fig.: Use cases of cloud storage

1. **Backup:** copies of files, databases and VM images are stored automatically in the cloud (e.g. AWS Backup, Azure Backup). Data is safe even if local disks fail; versioning allows restoring old copies; pay only for space used.
2. **Archiving:** old data that is rarely accessed but must be kept for years (records, logs, compliance data) is moved to very low-cost archive tiers such as **Amazon S3 Glacier**, Azure Archive Storage, Google Archive Storage. Retrieval may take minutes to hours; lifecycle rules move data automatically.
3. **Disaster recovery (DR):** data and server images are replicated to another region; if the main site fails (fire, flood, ransomware), systems are restored from the cloud quickly, without owning a second data centre.
4. **Data processing:** large data sets kept in object storage / data lakes are processed directly by cloud analytics and ML services (EMR / Spark, Redshift Spectrum, BigQuery, SageMaker) — storage and compute both scale.
5. **Content delivery:** static content (videos, images, software downloads, web pages) stored in cloud storage is cached at **edge locations** of a CDN (Amazon CloudFront, Azure CDN, Google Cloud CDN) near users, giving low latency and less load on the origin server.



Users get videos / files from the nearest edge → low latency

Fig.: Content delivery through edge locations (CDN)

(Five use cases — about 1 M each, diagram 1 M)

Note: This is a **model question paper prepared by Ananda Technologies** only for practice and exam preparation, based on the pattern of previous MSBTE question papers and the K-scheme syllabus. It is **not an official MSBTE paper**, and it does **not** mean that these questions will come in the exam. Answers follow standard textbook points; marks shown are indicative. Free for students — anandatechnologies.in