

Cloud Computing for Data Science (315327)

Model Question Paper — Set C · Model Answers

Q.1

a) Resource pooling and measured service.

2 M

- **Resource pooling:** the provider's computing resources (CPU, memory, storage, network) are **pooled and shared among many consumers** using a multi-tenant model; resources are dynamically assigned and reassigned as per demand. The user generally does not know the exact location of the resources (location independence).
- **Measured service:** resource usage is **automatically monitored, metered and reported** (e.g. CPU-hours, GB stored, API calls), giving transparency for both provider and user; user is billed on a pay-per-use basis.

(Each 1 M)

b) Examples and users of IaaS, PaaS, SaaS.

2 M

Model	Example	Mainly used by
IaaS	Amazon EC2 / Google Compute Engine	System / network administrators
PaaS	Google App Engine / Heroku / Azure App Service	Application developers
SaaS	Gmail / Microsoft 365 / Salesforce	End users

(Examples 1 M, users 1 M)

c) Private cloud vs community cloud.

2 M

Point	Private cloud	Community cloud
Users	Only one organisation	Several organisations with common concerns
Cost	Borne by one organisation (high)	Shared among members
Control	Full control by the organisation	Shared governance among members
Example	A bank's own cloud	Cloud shared by hospitals / govt. departments

(Any two points, 1 M each)

d) How cloud computing helps a data scientist.

2 M

1. Gives **on-demand GPUs / large VMs** for training models without buying hardware; pay only for hours used.
2. Provides **ready tools** — hosted Jupyter notebooks, AutoML, managed ML platforms (SageMaker, Vertex AI) and big-data warehouses (BigQuery, Redshift).
3. Easy **storage and sharing** of large data sets and notebooks with the team, and quick deployment of models as APIs.

(Any two, 1 M each)

e) Use cases of cloud storage.

2 M

1. Backup of files, databases and VMs.
2. Archiving of rarely-used data (e.g. Amazon S3 Glacier).
3. Disaster recovery (replicated copies in another region).
4. Data processing / big-data analytics and ML data lakes.
5. Content delivery (videos, images through CDN).

(Any four, ½ M each)

f) Limitations of ML in the cloud.

2 M

1. Data privacy, security and regulatory-compliance concerns.
2. Cost can grow quickly (GPU hours, storage, data transfer).
3. Vendor lock-in to one provider's tools.
4. Dependence on Internet; latency in uploading large data; less control over hardware.

(Any four, ½ M each)

g) Pre-tuned AI services.

2 M

Pre-tuned AI services are cloud AI services whose models are **already trained and tuned by the provider** on large data sets and offered as ready-to-use APIs; users get predictions without training any model.

Examples: Amazon Rekognition (image analysis), Amazon Comprehend (sentiment / NLP), Google Cloud Vision API, Azure AI Speech.

(Definition 1 M, two examples 1 M)

Q.2

a) PaaS and continuous delivery.

4 M

Platform as a Service (PaaS) provides a ready **platform — runtime, middleware, OS, databases and development tools** — on which developers build, test and deploy applications without managing servers, storage or networking. The developer manages only the application code and data.

- **Features:** language runtimes (Java, Python, PHP, Node.js), auto-scaling and load balancing, built-in databases and services, multi-tenant, pay per use.
- **Examples:** Google App Engine, Heroku, Microsoft Azure App Service, AWS Elastic Beanstalk, Red Hat OpenShift.

Continuous delivery using PaaS: continuous delivery (CD) keeps code always in a deployable state and releases it automatically and frequently.

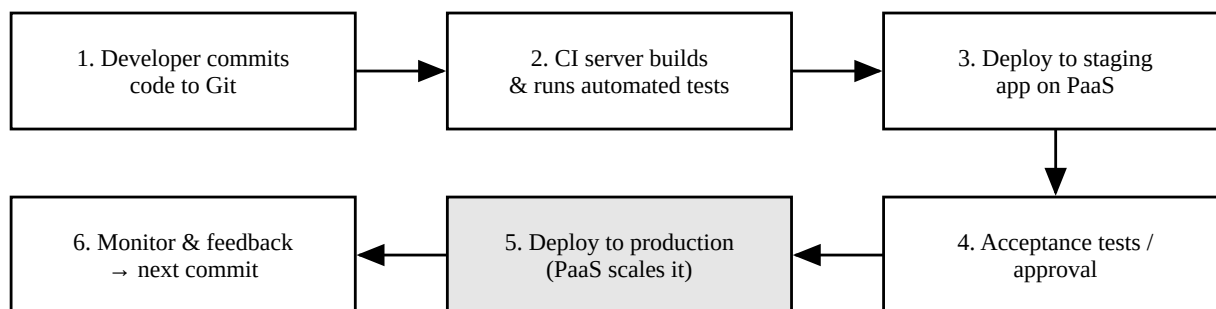


Fig.: Continuous delivery pipeline using PaaS

- Each code commit triggers an automatic **build and test** (CI tools such as GitHub Actions, Jenkins, AWS CodePipeline, Azure DevOps).
- PaaS gives identical **staging and production environments** and one-command / API deployment, so tested builds are pushed automatically.
- Features like deployment slots, rolling updates and easy rollback make releases safe; auto-scaling and monitoring give quick feedback.

(PaaS 2 M, continuous delivery 2 M)

b) Virtualization and its benefits.

4 M

Virtualization is the technique of creating virtual versions of physical resources (servers, storage, networks, desktops, data) using software such as a hypervisor, so that one physical resource can run many isolated virtual resources. It is the foundation of cloud computing.

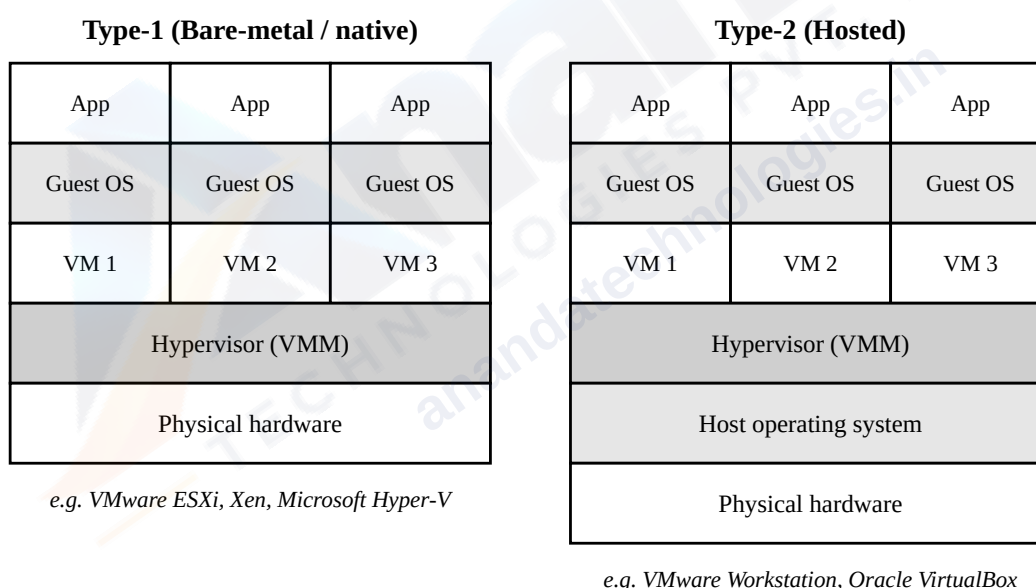


Fig.: Type-1 (bare-metal) and Type-2 (hosted) hypervisor

1. **Better resource utilisation and lower cost** — server consolidation reduces hardware, power, cooling and space.
2. **Scalability and fast provisioning** — VMs created / cloned from templates in minutes; supports elasticity of cloud.
3. **Isolation and security** — failure or attack in one VM does not affect others; multi-tenancy is possible.

4. **High availability and disaster recovery** — VMs can be migrated live, snapshotted and restored on another host.
5. **Run multiple OS** on one machine — useful for testing and legacy applications.

(Definition 1 M (diagram optional), any four benefits 3 M)

c) Key-value databases.

4 M

A **key-value database** is a type of NoSQL database that stores data as a collection of **(key, value) pairs**. The **key** is a unique identifier; the **value** can be any data — a string, number, JSON document or binary object — and is treated as opaque by the database. Data is read and written only through the key.

Key-value store (e.g. Amazon DynamoDB, Redis)	
KEY	VALUE (opaque)
user:101	{"name": "Riya", "city": "Pune"}
cart:101	["pen", "notebook"]
session:9f2a	logged-in, expires 10:30
img:55	<binary image bytes>

Operations: PUT(key, value) · GET(key) · DELETE(key) — lookup by key only

Fig.: Key-value database — each key maps to a value

Features:

- Very simple model and API: PUT, GET, DELETE by key.
- Extremely fast reads / writes (often in-memory); low latency.
- Schema-less — values need not follow a fixed structure.
- Horizontally scalable — data partitioned across nodes using the key (consistent hashing); replication gives high availability.

Examples / uses: Amazon DynamoDB, Redis, Riak, Memcached — used for session storage, shopping carts, caching, user profiles, leaderboards and storing ML feature values for fast online prediction.

(Concept 1 M, diagram 1 M, features 1 M, example 1 M)

d) Key features of Amazon SageMaker.

4 M

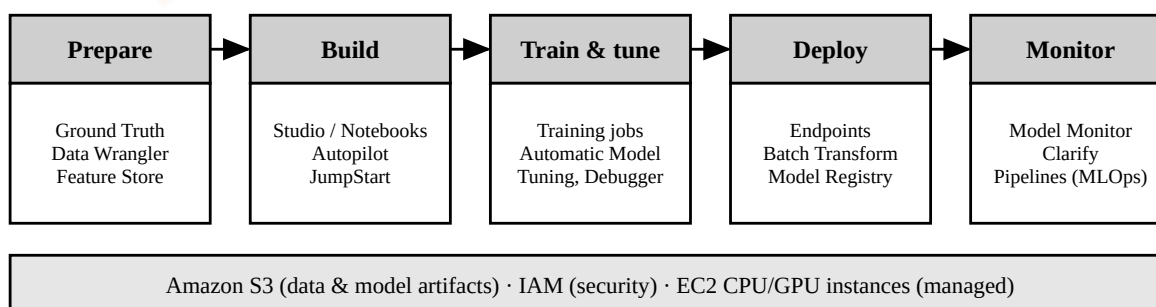


Fig.: Amazon SageMaker — components across the ML workflow

1. **SageMaker Studio** — fully managed web IDE with Jupyter notebooks for building models.

2. **Managed training** — built-in algorithms or own TensorFlow / PyTorch / scikit-learn code trained on chosen CPU / GPU instances; distributed and spot training.
3. **Automatic Model Tuning and Autopilot (AutoML)** — finds best hyper-parameters / models automatically.
4. **One-click deployment** — real-time endpoints with auto-scaling, Batch Transform; **Model Monitor** for drift; Ground Truth for labelling; pay-as-you-go.

(Any four features, 1 M each)

Q.3

a) Web services (SOAP and REST).

4 M

A **web service** is a software component that is available over the network and allows **machine-to-machine communication using standard web protocols** (HTTP) and formats (XML, JSON), independent of language or platform. Cloud services (e.g. S3, EC2, AI APIs) are offered as web services, so web services and SOA are the foundation of cloud computing.

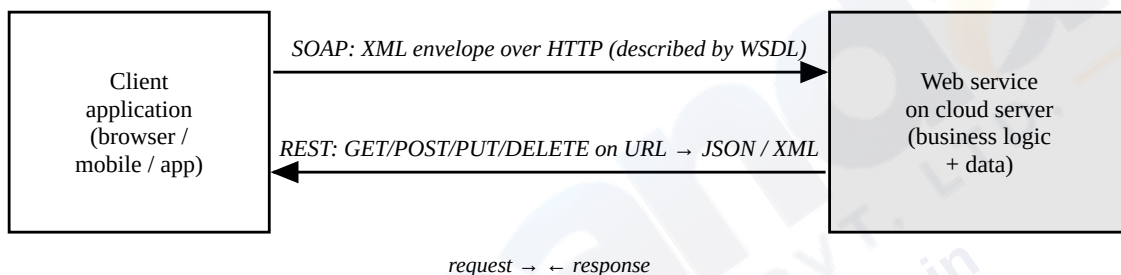


Fig.: Web service communication (SOAP and REST)

SOAP-based web services:

- SOAP (Simple Object Access Protocol) — messages are XML envelopes sent over HTTP / SMTP.
- **WSDL** describes the service interface; **UDDI** registry is used to publish and find services.
- Strict standards, built-in WS-Security; heavier.

RESTful web services:

- REST (Representational State Transfer) — every resource has a URL; standard HTTP methods GET, POST, PUT, DELETE act on it.
- Stateless, lightweight, usually JSON; easy to cache and scale — most cloud APIs (AWS, Google, Azure) are REST APIs.

(Concept 1 M, SOAP 1½ M, REST 1½ M)

b) Virtual machines vs containers.

4 M

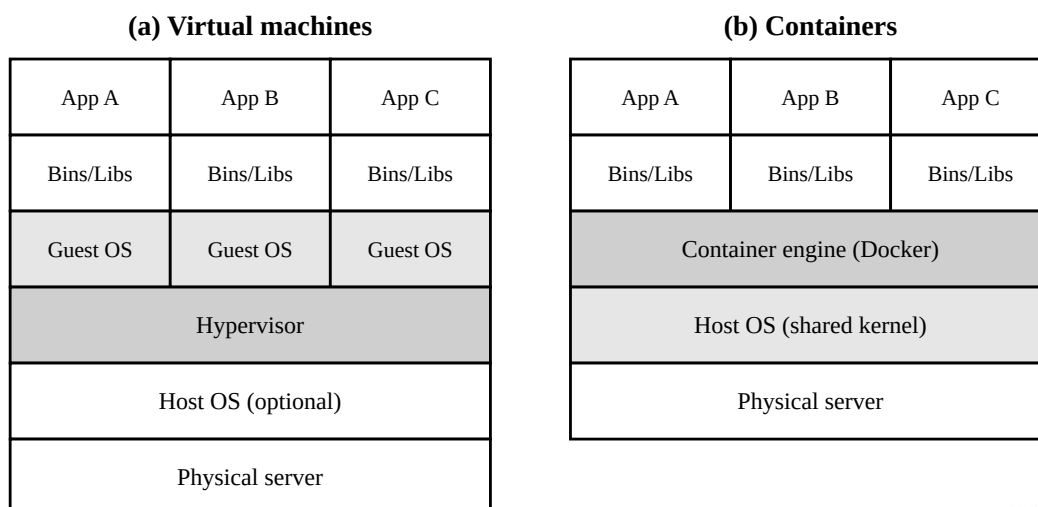


Fig.: Virtual machines vs containers

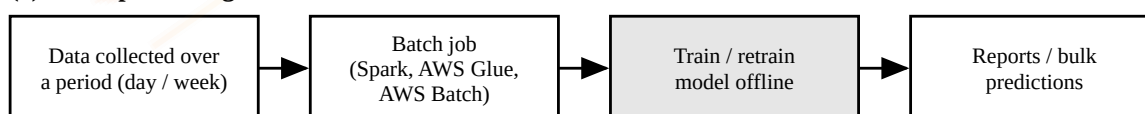
Point	Virtual machine (virtualization)	Container (containerization)
Virtualizes	Hardware — via hypervisor	Operating system — via container engine
OS	Each VM has its own full guest OS	All containers share the host OS kernel
Size	Heavy — gigabytes	Lightweight — megabytes
Start-up time	Minutes	Seconds or less
Isolation	Strong (separate kernel)	Process-level; weaker than VM
Density / overhead	Fewer VMs per host, more overhead	Many containers per host, low overhead
Portability	VM images are large, hypervisor-specific	"Build once, run anywhere" container images
Examples	VMware ESXi, Hyper-V, KVM, VirtualBox	Docker, containerd; orchestrated by Kubernetes

(Any four points, 1 M each)

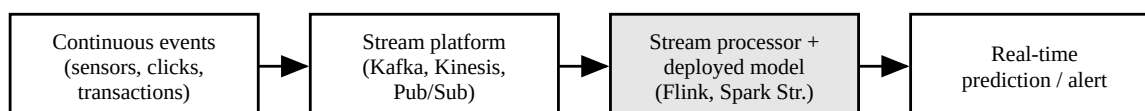
c) Bank fraud scenario — streaming vs batch.

4 M

(a) Batch processing



(b) Stream processing



Batch: high latency (hours), high throughput · Streaming: low latency (milliseconds–seconds)

Fig.: Batch processing vs stream processing in ML

(i) Real-time fraud detection → stream (streaming data) processing.

- Every card swipe is an event sent instantly to a streaming platform (Amazon Kinesis / Apache Kafka).
- A stream processor (Apache Flink, Spark Streaming, Kinesis Data Analytics) enriches the event with features (amount, location, last transactions) and calls the deployed fraud model endpoint.
- Decision (approve / block / OTP) is made in milliseconds, before the payment completes — so low latency is essential.

(ii) Nightly retraining → batch processing.

- All of the day's transactions with confirmed fraud labels are stored in a data lake / warehouse (S3, Redshift).
- A scheduled batch job (Spark / AWS Glue + SageMaker training job) runs at night on the full data set to clean data, build features and retrain the model.
- The new model is validated and deployed to the real-time endpoint for the next day; high throughput is more important than latency.

(Streaming case 2 M, batch case 2 M)

d) Google Cloud AutoML.

4 M

Google Cloud AutoML, now offered through **Vertex AI**, is a suite of ML products that lets developers with little ML expertise train **custom, high-quality models on their own data** using a graphical interface; Google automates model selection, transfer learning, architecture search and hyper-parameter tuning.

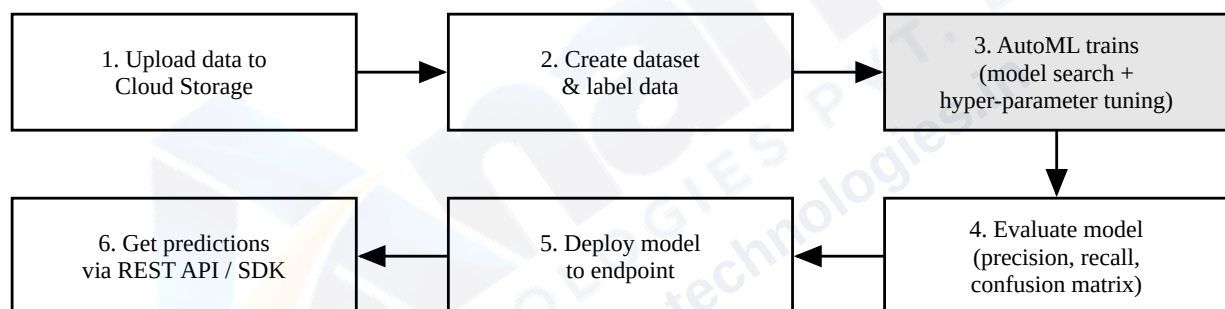


Fig.: Google Cloud AutoML (Vertex AI) workflow

Functionalities:

- **AutoML Vision / Image:** image classification and object detection.
- **AutoML Video:** video classification, object tracking, action recognition.
- **AutoML Natural Language / Text:** text classification, entity extraction, sentiment analysis; **AutoML Translation** for custom translation models.
- **AutoML Tables / Tabular:** classification, regression and forecasting on structured data.
- Built-in evaluation (precision, recall, confusion matrix), one-click deployment to endpoints, REST API / SDK access, export for edge devices.

(Concept 1 M, workflow / diagram 1 M, functionalities 2 M)

Q.4

a) Front end and back end of cloud architecture.

4 M

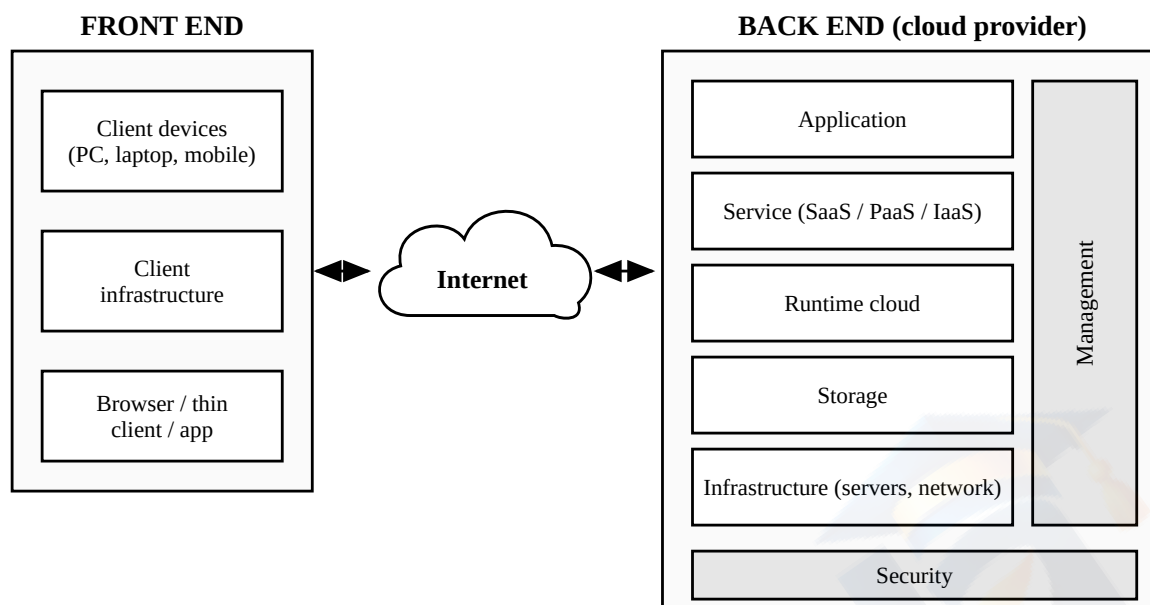


Fig.: Architecture of cloud computing (front end and back end)

Front end (client side):

- **Client infrastructure / devices** — PCs, laptops, mobiles, thin clients.
- **User interface** — web browser, mobile app or client software (e.g. Google Drive app) through which the user requests services.

Back end (cloud provider side):

- **Application** — the software or platform delivered to the user.
- **Service** — SaaS / PaaS / IaaS; **runtime cloud** — execution environment / VMs.
- **Storage** — databases and object / block storage; **infrastructure** — servers, network devices, hypervisors.
- **Management** (resource allocation, monitoring) and **security** (authentication, encryption, firewalls).

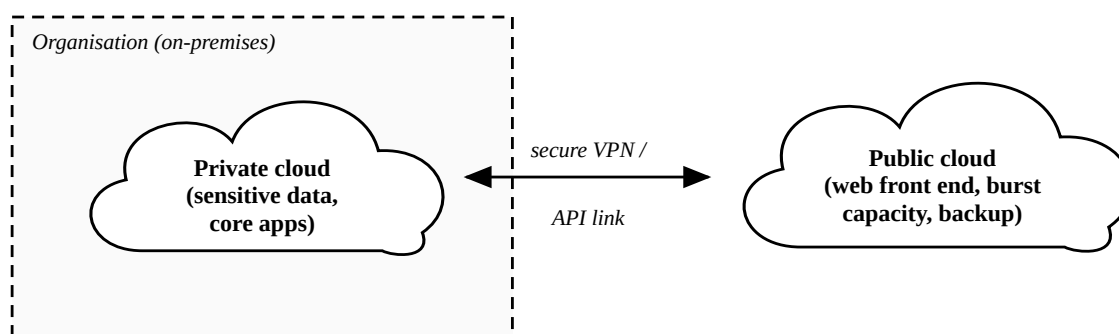
The two ends communicate through the **Internet / network**.

(Diagram 2 M, explanation 2 M)

b) Hybrid cloud.

4 M

A **hybrid cloud** is a combination of **two or more distinct clouds** — usually a **private cloud** and a **public cloud** — that remain separate but are connected by technology that allows data and application portability. Sensitive workloads run in the private cloud, while the public cloud is used for less-critical work or extra capacity.



Cloud bursting: extra load moves to the public cloud when private capacity is full

Fig.: Hybrid cloud — private and public clouds connected

Advantages:

- Flexibility — choose the best place for each workload.
- Cost-effective — pay public cloud only for extra / peak load (cloud bursting).
- Security and compliance — confidential data stays private.
- Scalability and business continuity (backup / DR in public cloud).

Disadvantages:

- Complex to design, integrate and manage two environments.
- Security risk at the connection points; data transfer cost and latency.
- Needs compatible tools and skilled staff.

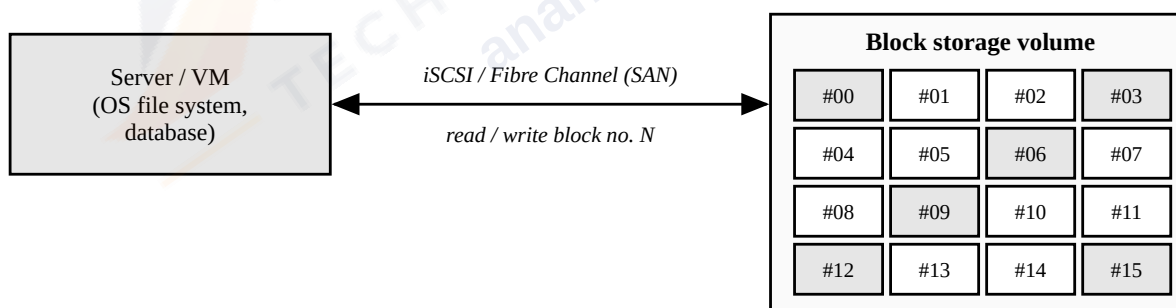
Example: a bank keeps customer accounts in its private cloud and runs its public website and ML experiments on AWS.

(Concept with diagram 2 M, advantages 1 M, disadvantages 1 M)

c) Block-based and file-based storage.

4 M

Block-based storage: data is divided into fixed-size blocks; each block has a unique address and is stored independently without metadata. The volume is attached to a server as a raw disk (through SAN — iSCSI / Fibre Channel); the OS formats it with a 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

- Very low latency and high IOPS — suited to databases, transactional apps and VM boot disks.
- Examples: Amazon EBS, Azure Managed Disks, Google Persistent Disk.

File-based storage: data is stored as **files inside a hierarchy of folders**; each file is located by its path (e.g. /projects/data.csv). It is provided by **Network Attached Storage (NAS)** and shared over the network using NFS (Linux) or SMB / CIFS (Windows).

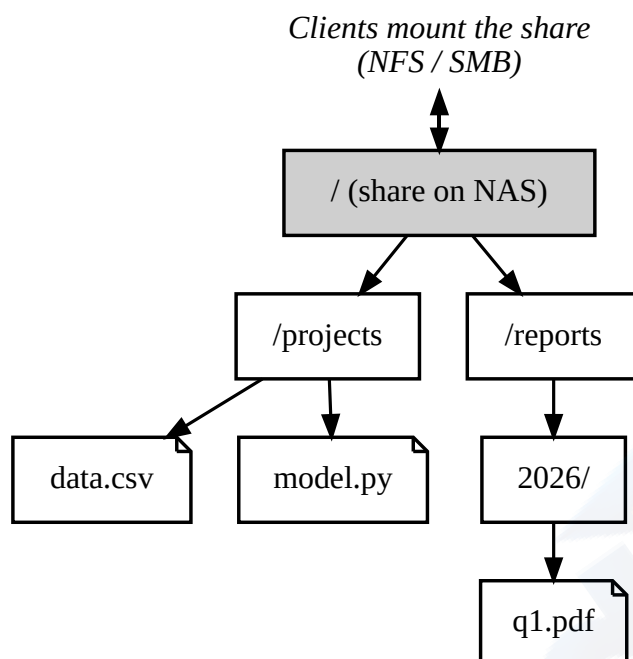


Fig.: File-based storage — hierarchy of folders and files (NAS)

- Easy to use and share among many users / servers at the same time; keeps basic metadata (name, size, date, permissions).
- Performance drops with very large numbers of files; scaling is limited. Examples: Amazon EFS, Azure Files, Google Filestore.

(Block storage 2 M, file storage 2 M)

d) Overview of an ML system and benefits of managed ML platforms.

4 M

An **ML system** is not only the model; it is the complete pipeline that takes raw data to useful predictions and keeps them accurate.

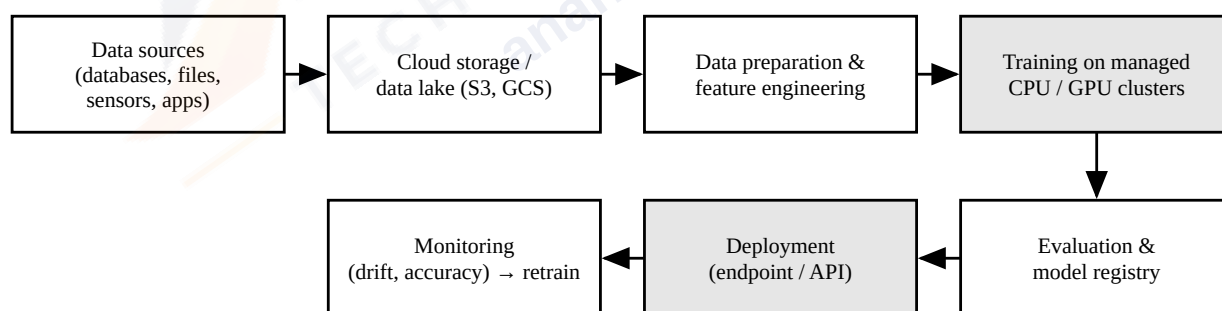


Fig.: Machine learning in the cloud — overview of an ML system

Parts: data sources and storage → data preparation and feature engineering → model training → evaluation and registry → deployment (serving) → monitoring and retraining.

Benefits of managed ML platforms (Amazon SageMaker, Azure ML, Google Vertex AI):

1. **No infrastructure management** — provider sets up, scales and patches servers, GPUs and clusters.

2. **End-to-end tools in one place** — notebooks, labelling, AutoML, training, tuning, deployment and monitoring.
3. **Scalability and cost control** — resources start only when needed and stop after the job; spot / serverless options.
4. **Faster time to production and MLOps** — one-click endpoints, pipelines, model registry, versioning, built-in security and compliance.

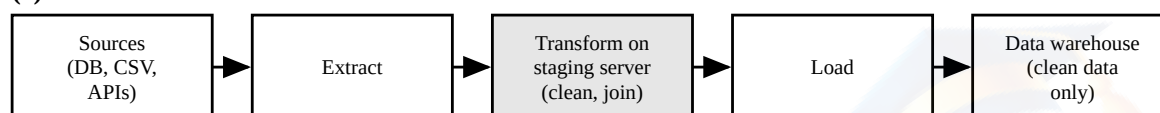
(ML system overview 2 M, any four benefits 2 M)

e) ETL pipeline with sales-data example.

4 M

ETL (Extract – Transform – Load) is a data pipeline that collects data from sources, converts it into a clean, consistent form and loads it into a data warehouse for analytics and ML.

(a) ETL — Extract → Transform → Load



(b) ELT — Extract → Load → Transform

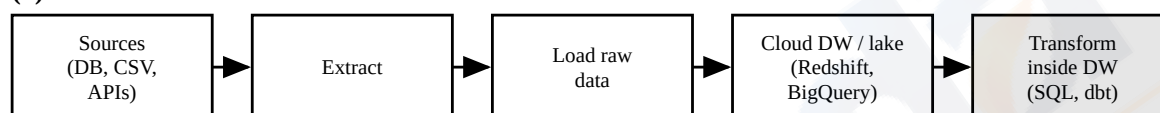


Fig.: ETL vs ELT pipeline

1. **Extract:** read raw data from sources — e.g. daily sales CSV files from shops, an orders database, a payment API.
2. **Transform:** remove duplicate orders and rows with missing values, correct formats (city names, dates), convert data types, compute new columns (amount = qty × price), aggregate.
3. **Load:** insert the cleaned rows into the warehouse table (Redshift / BigQuery), usually on a schedule (AWS Glue, Azure Data Factory).

Small working example (Python; an in-memory SQLite table stands in for the warehouse):

```

# Mini ETL: Extract (CSV) -> Transform (clean) -> Load (SQL table)
import csv, io, sqlite3

raw = """order_id,city,product,qty,price
101,Pune,Pen,10,5
102,nanded ,Notebook,4,40
103,Mumbai,Pen,,5
104,Pune,Bag,1,650
102,nanded ,Notebook,4,40
"""

# 1. EXTRACT
rows = list(csv.DictReader(io.StringIO(raw)))
print("Extracted rows:", len(rows))

# 2. TRANSFORM: remove duplicates and missing qty, clean city, add amount
seen, clean = set(), []
for r in rows:
    if r["order_id"] in seen or not r["qty"]:
        continue
    seen.add(r["order_id"])
  
```

```

    qty, price = int(r["qty"]), float(r["price"])
    clean.append((int(r["order_id"]), r["city"].strip().title(), r["product"],
    qty, qty * price))
print("Rows after transform:", len(clean))

# 3. LOAD into the warehouse table
db = sqlite3.connect(":memory:")
db.execute("CREATE TABLE sales(order_id INT, city TEXT, product TEXT, qty INT,
amount REAL)")
db.executemany("INSERT INTO sales VALUES (?, ?, ?, ?, ?)", clean)

for city, total in db.execute("SELECT city, SUM(amount) FROM sales GROUP BY city
ORDER BY city"):
    print(f"{city:<8}{total:>8.2f}")

```

Output:

```

Extracted rows: 5
Rows after transform: 3
Nanded      160.00
Pune        700.00

```

Of 5 extracted rows, one duplicate (order 102) and one row with missing quantity (order 103) were removed in the transform step, so 3 clean rows were loaded.

(Explanation of E, T, L 3 M, example 1 M)

Q.5

a) Server, storage and application virtualization.

6 M

(i) **Server virtualization:** a physical server is partitioned into many isolated **virtual machines** by a hypervisor; each VM has its own OS and applications and believes it owns the hardware.

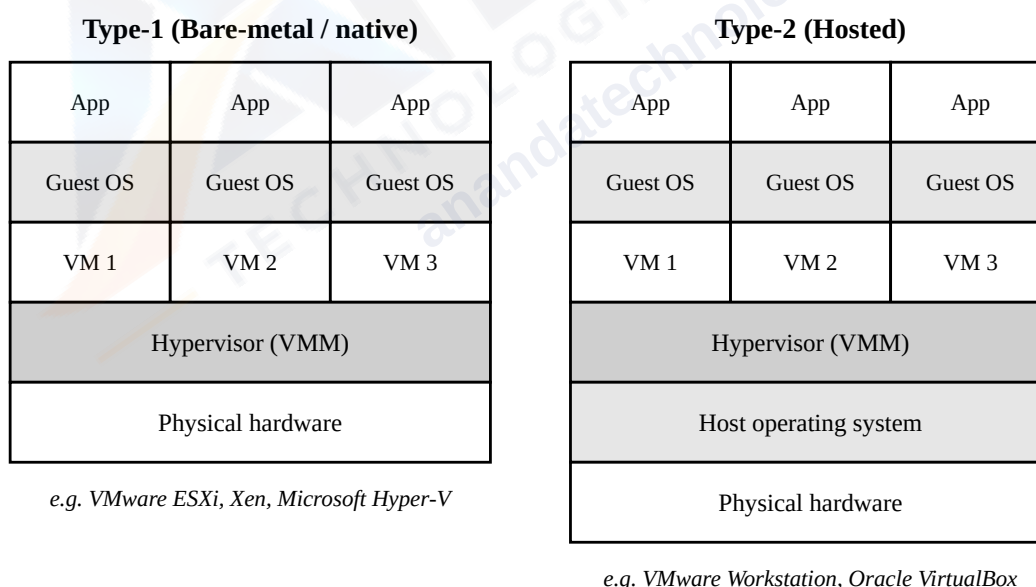


Fig.: Type-1 (bare-metal) and Type-2 (hosted) hypervisor

- Type-1 (bare-metal)** hypervisor runs directly on hardware (VMware ESXi, Xen, Hyper-V, KVM); **Type-2 (hosted)** runs on a host OS (VirtualBox, VMware Workstation).

- Approaches: full virtualization, para-virtualization, hardware-assisted virtualization (Intel VT-x, AMD-V).
- Benefits: server consolidation, live migration, high availability — basis of IaaS (EC2).

(ii) Storage virtualization: physical storage from many disks / arrays is **pooled** and presented as a single logical storage; a virtualization layer maps logical volumes to physical locations.

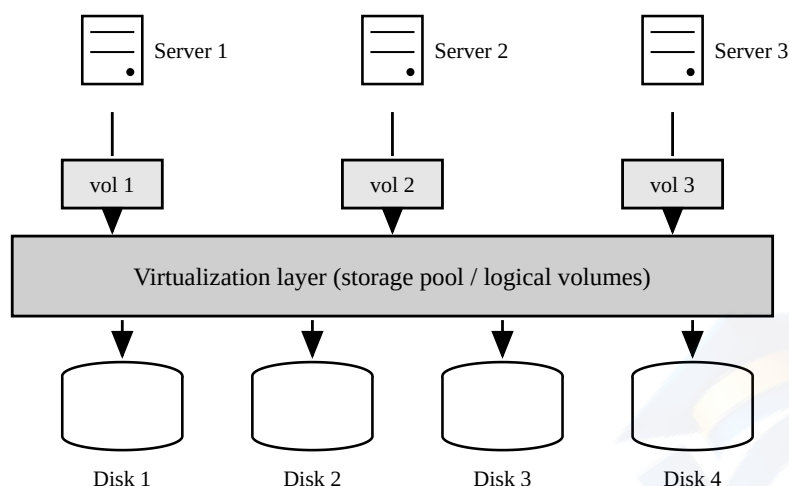


Fig.: Storage virtualization — physical disks pooled into virtual volumes

- Implemented in host, array or network (SAN / NAS) level; supports thin provisioning, snapshots, replication.
- Benefits: better utilisation, easy management and backup, non-disruptive data migration.

(iii) Application virtualization: applications are packaged and run in an isolated environment or **streamed from a server** to the user's device without normal installation; the app does not conflict with other apps or the OS.

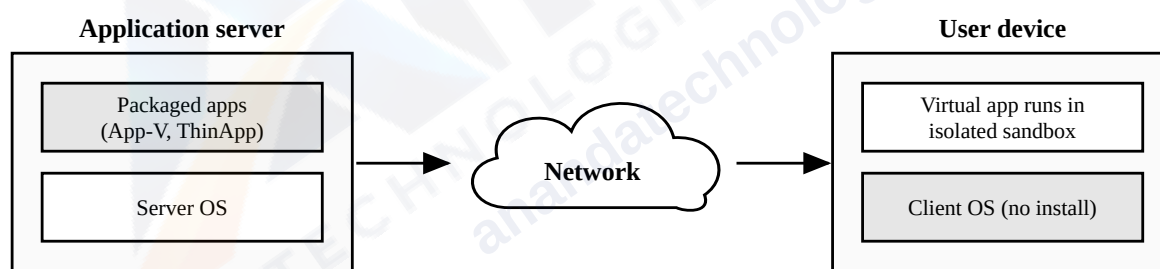


Fig.: Application virtualization (application streamed from server)

- Examples: Microsoft App-V, VMware ThinApp, Citrix Virtual Apps.
- Benefits: central updates and licensing, runs on many devices, fewer compatibility problems.

(Each type with diagram 2 M)

b) Amazon Redshift — architecture, working and loading from S3.

6 M

Amazon Redshift is a fully managed, petabyte-scale **cloud data warehouse** service in AWS used to run complex SQL analytics on large data sets. It is based on a cluster of nodes using **massively parallel processing (MPP)**, **columnar storage** and data compression.

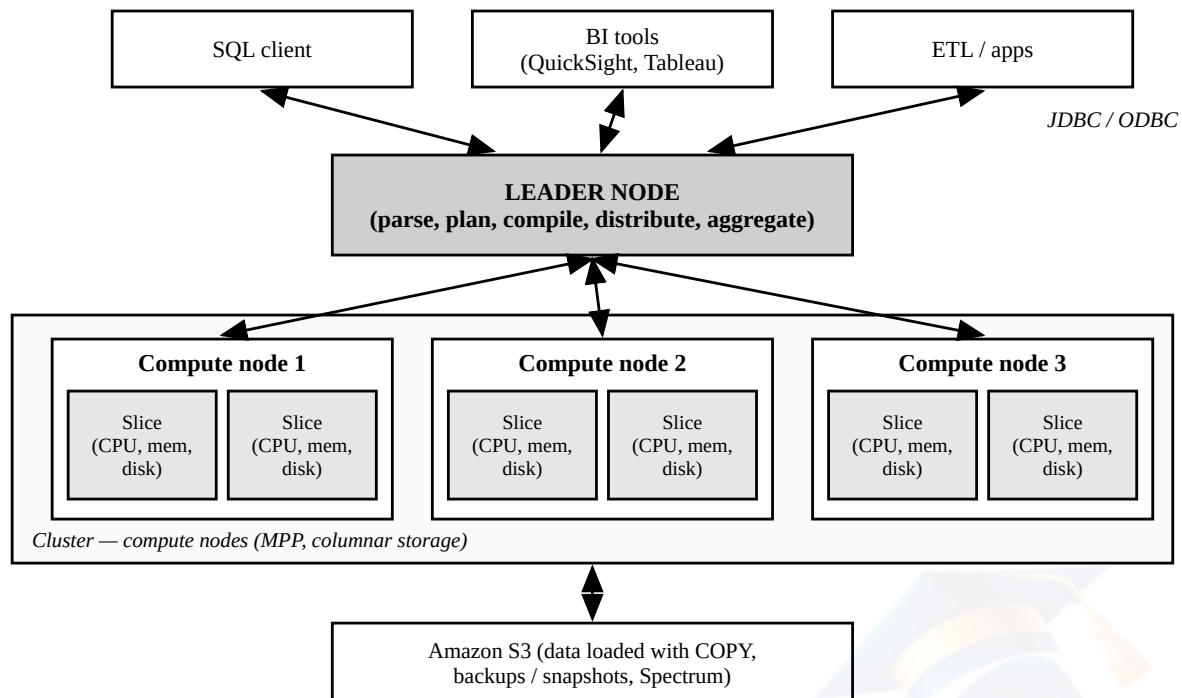


Fig.: Amazon Redshift architecture

Architecture and working:

1. **Client applications** (SQL editors, BI tools such as QuickSight / Tableau, ETL tools) connect to the cluster through JDBC / ODBC.
2. The **leader node** manages client connections, parses the query, builds an optimised execution plan, compiles code and sends it to compute nodes.
3. **Compute nodes** run the compiled code in parallel on their part of the data and send intermediate results back; the leader node aggregates and returns the final result.
4. Each compute node has **node slices**; table rows are distributed across slices (by distribution key) and stored column by column — queries read only needed columns, so I/O is reduced; zone maps skip unneeded blocks.
5. Features: scale by adding nodes / RA3 managed storage, automatic backups (snapshots) to S3, Redshift Spectrum to query S3 data directly, Redshift ML for training models with SQL.

Loading data from Amazon S3:

1. Upload data files (CSV, JSON, Parquet) to an S3 bucket — ideally split into multiple files so every slice loads in parallel.
2. Create an IAM role that allows Redshift to read the bucket and associate it with the cluster.
3. Create the target table in Redshift.
4. Run the COPY command, e.g. `COPY sales FROM 's3://my-bucket/sales/' IAM_ROLE '<role-arn>' FORMAT AS CSV IGNOREHEADER 1;` — data is loaded in parallel by all slices; check errors in `STL_LOAD_ERRORS`; run `ANALYZE` and query.

(Architecture diagram 2 M, working 2 M, loading from S3 2 M)

c) Azure ML Studio components; creating a workspace and compute cluster.

6 M

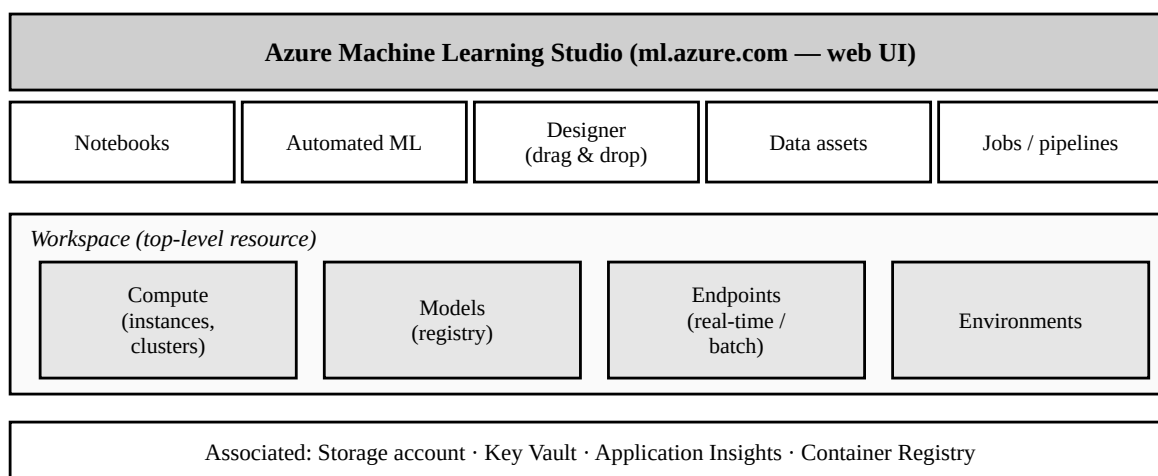


Fig.: Azure Machine Learning workspace and Studio

Components of Azure Machine Learning Studio:

1. **Designer:** drag-and-drop canvas to build ML pipelines from ready modules (data input, clean missing data, split data, train model, score, evaluate) — low-code.
2. **Automated ML:** give data and target column; Azure automatically tries algorithms, feature processing and hyper-parameters and ranks the models; best model can be deployed.
3. **Notebooks:** Jupyter notebooks inside the studio running on a compute instance, with the Azure ML Python SDK for code-first work.
4. **Compute:** *compute instance* (single managed VM for development), *compute cluster* (auto-scaling group of CPU / GPU VMs for training jobs), *Kubernetes / inference compute* and *serverless compute* for deployment.

Steps to create a workspace:

1. Sign in to the **Azure portal** with an Azure subscription; select **Create a resource** → search **Azure Machine Learning** → **Create**.
2. Choose subscription and **resource group**, enter **workspace name** and **region**; associated resources (storage account, key vault, Application Insights, container registry) are created / selected.
3. Click **Review + create** → **Create**; after deployment click **Launch studio** (or open ml.azure.com).

Steps to create a compute cluster:

1. In the studio select **Compute** → **Compute clusters** → + **New**.
2. Select location, **VM tier** (dedicated or low-priority), **VM type** (CPU / GPU) and **VM size**.
3. Enter compute name, **minimum nodes** (0 so that it scales down to zero cost) and **maximum nodes**, and idle seconds before scale-down; click **Create**. The cluster is then selected as the target when submitting training jobs.

(Components 2 M, workspace steps 2 M, compute steps 2 M)

Q.6

a) AIaaS and GPUaaS.

6 M

Cloud-based ML services are mainly of two types — AI as a Service and GPU as a Service.

(i) Artificial Intelligence as a Service (AIaaS): AI capabilities offered by cloud providers through APIs and managed platforms on a pay-per-use basis.

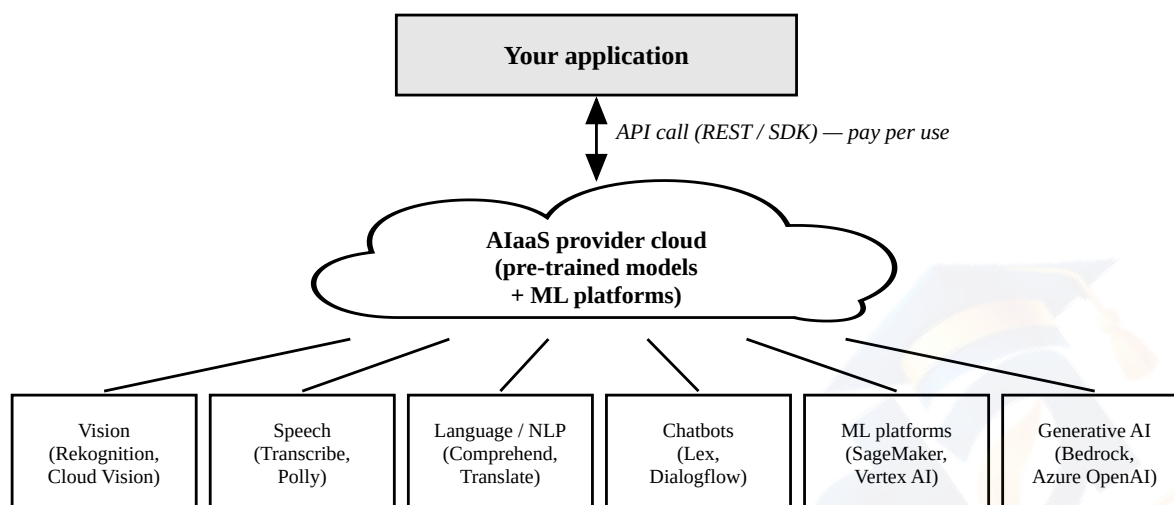


Fig.: Artificial Intelligence as a Service (AIaaS)

- **Pre-trained APIs:** vision (Amazon Rekognition, Google Cloud Vision), speech (Amazon Transcribe / Polly), language (Amazon Comprehend, Translate), chatbots (Amazon Lex, Dialogflow).
- **ML platforms / AutoML:** Amazon SageMaker, Azure ML, Google Vertex AI.
- **Generative AI:** Amazon Bedrock, Azure OpenAI Service, Vertex AI Gemini.
- **Benefits:** no need for ML experts or infrastructure, fast integration, scalable, low starting cost.
- **Concerns:** data privacy, cost at high volume, lock-in.

(ii) GPU as a Service (GPUaaS): renting GPU-powered servers from the cloud on demand for compute-heavy work.

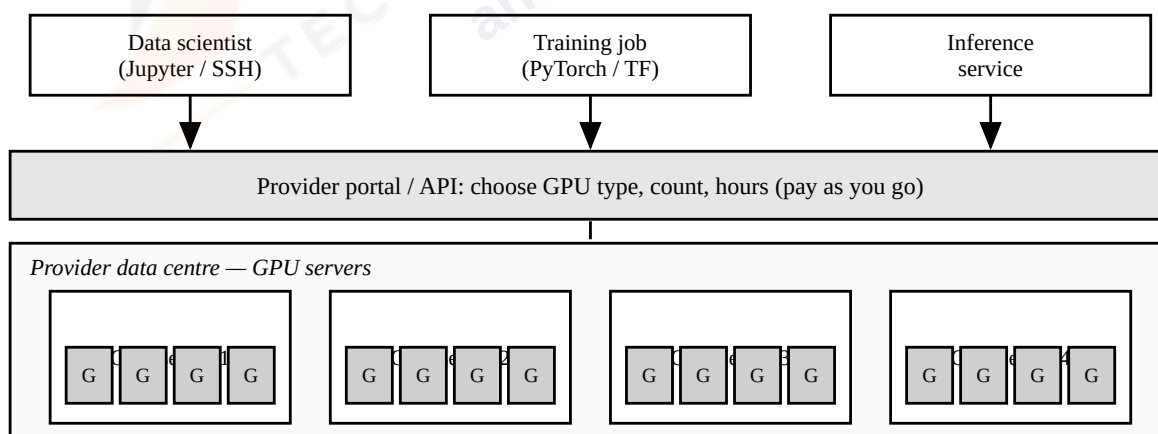


Fig.: GPU as a Service (GPUaaS)

- **Models:** on-demand, reserved / dedicated, spot / preemptible, serverless GPU; full or shared virtual GPUs.

- **Use cases:** deep-learning training and inference, computer vision, LLM fine-tuning, scientific simulation, video rendering.
- **Providers:** AWS (EC2 P / G instances), Google Cloud (A100 / H100 / L4, TPUs), Microsoft Azure (NC / ND series), Oracle, IBM, NVIDIA DGX Cloud, Lambda.
- **Benefits:** no capital cost for expensive GPUs, latest hardware, scale to many GPUs, pay per second / hour.

(AIaaS 3 M, GPUaaS 3 M)

b) Stages of a cloud ML project.

6 M

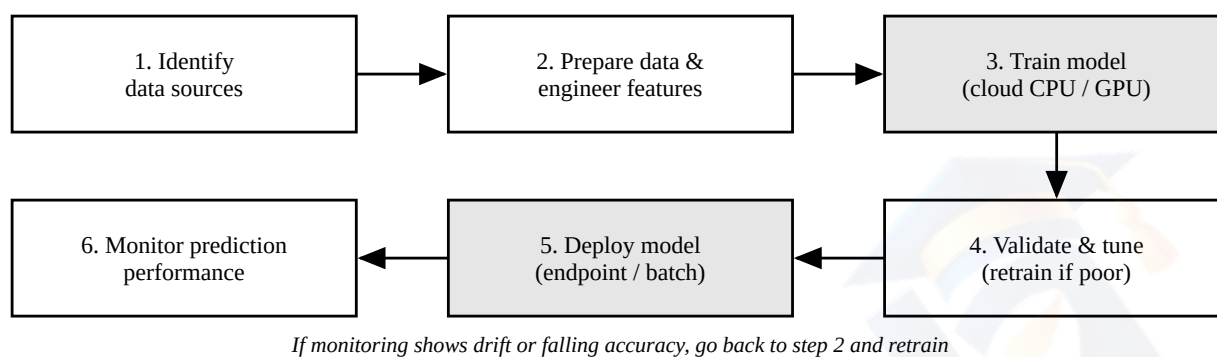


Fig.: Steps to train a machine learning project in the cloud

- 1. Identify data sources:** decide what data the problem needs and where it is — transactional databases, CSV / Excel files, logs, IoT streams, third-party APIs, public data sets. Bring it into cloud storage (S3, GCS, Azure Blob) or a warehouse (Redshift, BigQuery) through ETL / ELT or streaming pipelines; check quantity, quality and permissions.
- 2. Feature engineering (data preparation):** clean data (missing values, outliers, duplicates), transform (encoding categorical values, scaling, date parts), create new informative features, select important ones; split data into training / validation / test sets. Tools: SageMaker Data Wrangler, Dataflow, Databricks, notebooks; Feature Store for reuse.
- 3. Training:** choose algorithm and framework; launch managed training jobs on suitable CPU / GPU instances; use distributed (scale-out) training for big data; track experiments.
- 4. Validation:** evaluate on validation / test data with suitable metrics (accuracy, precision, recall, F1, AUC, RMSE); tune hyper-parameters (automatic model tuning); check bias and over-fitting; register the best model.
- 5. Deployment:** deploy as a real-time REST endpoint, serverless / batch prediction job or on edge devices; enable auto-scaling, security and versioning (A/B or canary rollout).
- 6. Monitoring:** track prediction quality, data / concept drift, latency, errors and cost; set alerts; retrain automatically with new data through MLOps pipelines.

(Diagram 1 M, five stages 5 M)

c) E-commerce scenario — pre-tuned AI services and monitoring.

6 M

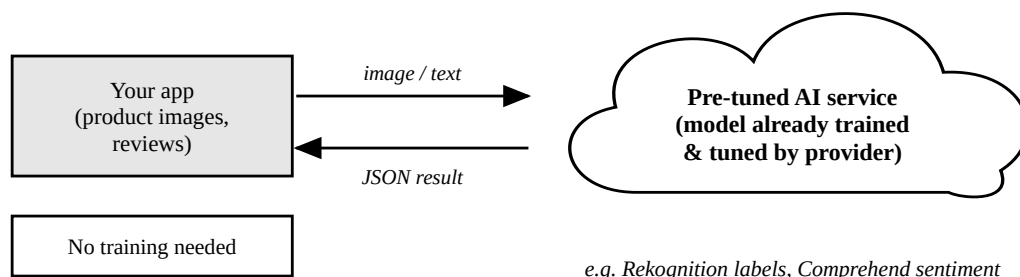


Fig.: Using a pre-tuned (pre-trained) AI service

Suitable pre-tuned AI services:

- 1. Automatic product-image tagging:** an image-analysis API such as **Amazon Rekognition** (DetectLabels), **Google Cloud Vision API** (label detection) or **Azure AI Vision** returns labels like 'shoe', 'red', 'leather' with confidence scores; tags are stored with the product for search and filters. Rekognition *Custom Labels* / AutoML can be used if company-specific categories are needed.
- 2. Detecting negative reviews:** a text-analytics API such as **Amazon Comprehend** (DetectSentiment), **Google Natural Language API** or **Azure AI Language** returns sentiment (positive / negative / neutral / mixed) with scores; negative reviews are flagged to the support team. Amazon Translate can first convert regional-language reviews to English.
- 3. Why pre-tuned:** no need to collect training data or train models; quick integration via SDK; pay per image / per unit of text; auto-scaling during sales.

Monitoring prediction performance after deployment:

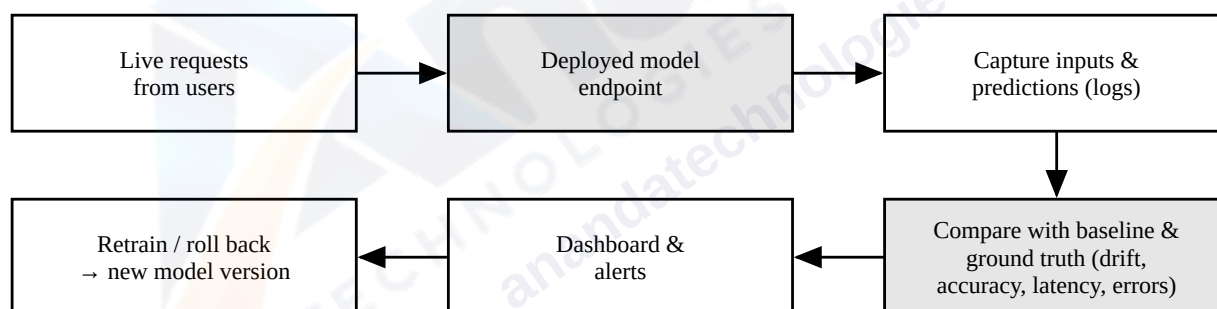


Fig.: Monitoring prediction performance of a deployed model

- 1. Log** each request — image / review, returned tags or sentiment and confidence score.
- 2. Human review sample:** regularly have staff check a random sample (e.g. 1–2 % of items) to create ground truth; calculate precision / recall of tags and accuracy of sentiment.
- 3. Watch confidence and drift:** track average confidence, share of low-confidence results and changes in input type (new product categories, new slang or languages) — signs of drift.
- 4. Operational metrics:** API latency, error rate, throttling and monthly cost in CloudWatch / Azure Monitor dashboards with alarms.
- 5. Action:** send low-confidence cases to human review (e.g. Amazon Augmented AI), add custom labels / custom classifier trained on company data if accuracy falls, and keep feedback loop from customers.

(Services for both tasks 3 M, monitoring plan 3 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

