

Cloud Computing for Data Science (315327)

Model Question Paper — Set B · Model Answers

Q.1

a) Grid computing.

2 M

Grid computing is a distributed computing model in which **geographically dispersed, heterogeneous computers** belonging to different organisations are connected through middleware and work together as one virtual super-computer to solve a large task. The task is split into small jobs that run in parallel on the grid nodes.

Example: Worldwide LHC Computing Grid (CERN) that processes particle-physics data on computers in many countries.

(Definition 1 M, example 1 M)

b) On-demand self-service and rapid elasticity.

2 M

- **On-demand self-service:** a user can provision computing resources such as servers, storage or a notebook instance **automatically through a web portal / API**, whenever needed, without human interaction with the provider. e.g. launching an EC2 instance in minutes.
- **Rapid elasticity:** resources can be **scaled out or in quickly** (often automatically) as demand changes, so they appear unlimited to the user. e.g. auto-scaling web servers during exam-result day.

(Each 1 M)

c) Virtualization and its need.

2 M

Virtualization is the creation of a virtual (software) version of a physical resource — server, storage, network, OS — so that one physical resource is shared by many isolated users / virtual machines through a hypervisor.

Need: (1) better utilisation of hardware and lower cost (server consolidation); (2) isolation, fast provisioning and easy migration / backup — the base of cloud IaaS.

(Definition 1 M, two needs 1 M)

d) Four types of cloud storage.

2 M

1. **Public cloud storage** — provider-owned, shared, pay-per-use (Amazon S3, Google Drive).
2. **Private cloud storage** — dedicated to one organisation, behind its firewall.
3. **Hybrid cloud storage** — mix of private (sensitive data) and public (backup, extra capacity).
4. **Community cloud storage** — shared by organisations with common needs (e.g. hospitals).

(By storage format, cloud storage is also classified as block, file and object storage.)

(Four types, ½ M each)

e) Limitations of key-value databases.

2 M

1. **Lookup by key only** — cannot search or filter on values; no secondary queries without extra indexes.
2. **No joins / relationships** between data; not suitable for complex relational data.
3. **No SQL / complex queries and analytics** — reporting and aggregation are difficult.
4. **Limited transactions and consistency** — many stores give only eventual consistency; value is opaque, so partial updates are hard.

(Any four, ½ M each)

f) Cloud machine learning.

2 M

Cloud machine learning means building, training, deploying and managing ML models using **cloud-provided compute, storage and managed ML services** on a pay-per-use basis, instead of local hardware.

Platforms: Amazon SageMaker, Azure Machine Learning Studio, Google Cloud Vertex AI / AutoML, IBM Watson Studio.

(Definition 1 M, two platforms 1 M)

g) GPUaaS providers.

2 M

1. **Amazon Web Services** — EC2 P- and G-series GPU instances.
2. **Google Cloud** — Compute Engine GPUs (NVIDIA A100 / H100 / L4), also TPUs.
3. **Microsoft Azure** — NC / ND / NV-series GPU virtual machines.
4. **Oracle Cloud / IBM Cloud / NVIDIA DGX Cloud / Lambda / Paperspace.**

(Any four, ½ M each)

Q.2

a) Infrastructure as a Service (IaaS).

4 M

IaaS is the cloud service model that provides basic **computing infrastructure — virtual machines, storage, networks and load balancers** — **on demand** over the Internet. The provider manages the physical data centre, servers, storage, networking and virtualization; the user installs and manages the OS, middleware, runtime, applications and data.

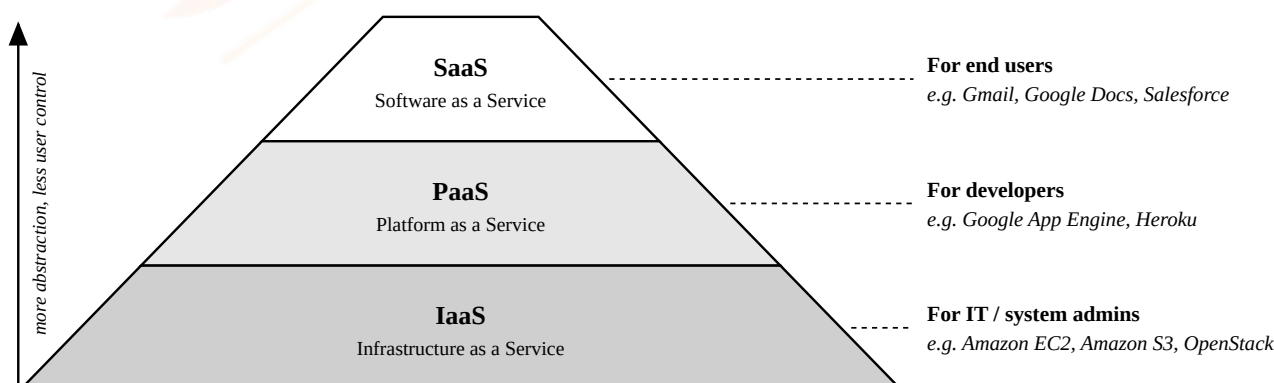


Fig.: Cloud service models (SPI model) with examples

- **Characteristics:** resources delivered as VMs; dynamic scaling; pay per hour / GB; user gets full control (root access) of the VM; multi-tenant hardware.
- **Examples:** Amazon EC2, Amazon S3 and EBS, Google Compute Engine, Microsoft Azure Virtual Machines, OpenStack.
- **Advantages:** no capital cost on hardware; highly scalable and flexible; full control over software stack; fast setup of test labs and GPU servers for ML.
- **Disadvantages:** user is responsible for OS patching, security and backups of VMs; needs skilled administrators; cost may rise if VMs are left running.

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

b) Community cloud.

4 M

A **community cloud** is a cloud infrastructure **shared by several organisations that have common concerns** — same mission, security requirements, policy or compliance needs. It may be managed by the member organisations or by a third party, and hosted on- or off-premises.

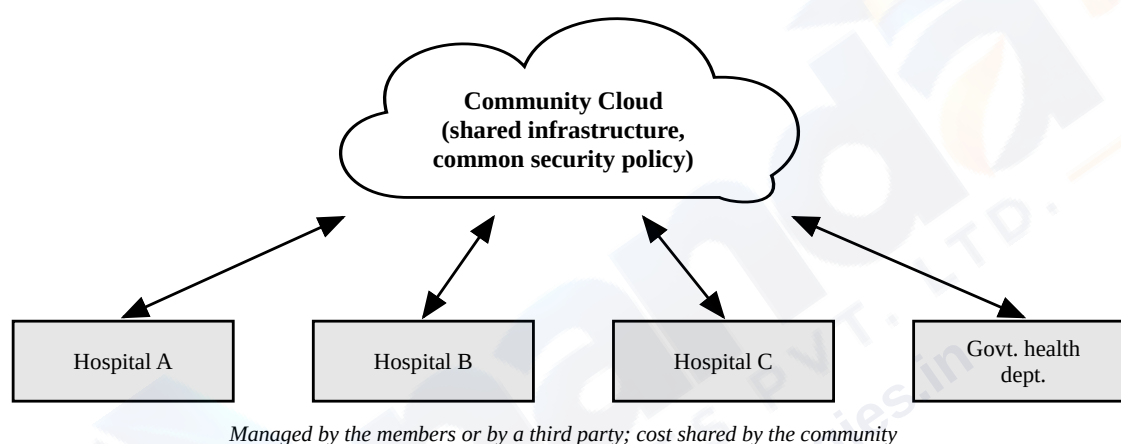


Fig.: Community cloud shared by organisations with common concerns

Examples: cloud shared by government departments, by hospitals for health records, or by universities for research.

Advantages:

- Cost is shared among members — cheaper than each building a private cloud.
- Better security, privacy and compliance than a public cloud.
- Easy collaboration and data sharing between member organisations.
- Common standards and policies for all members.

Disadvantages:

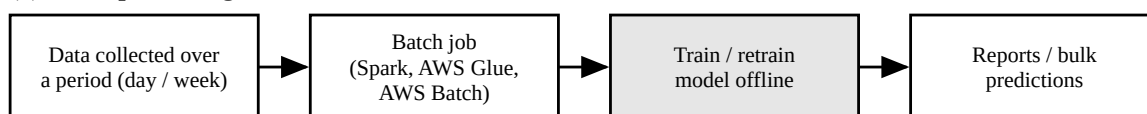
- Fixed storage and bandwidth shared among members; less scalable than public cloud.
- Costlier than public cloud.
- Deciding responsibility, governance and security among members is difficult.
- Not suitable for every organisation; data may still be visible to other members if policies are weak.

(Concept 1 M, advantages 1½ M, disadvantages 1½ M)

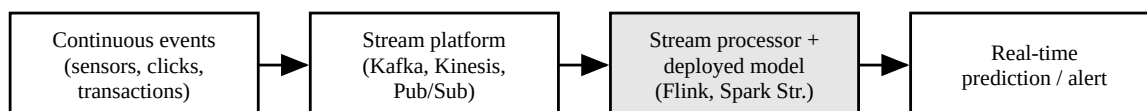
c) Batch and streaming data processing in ML.

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

Batch data processing: data is collected over a period (hour, day, week) and processed together as a large batch by scheduled jobs (Apache Spark, Hadoop, AWS Glue, AWS Batch).

- Used for **training and periodic re-training** of models and for bulk (batch) predictions, e.g. monthly churn scores for all customers.
- High throughput, simple, cheaper; but results are delayed (high latency).

Streaming data processing: data is produced continuously (sensor readings, clicks, payments) and processed **as soon as it arrives** through streaming platforms (Apache Kafka, Amazon Kinesis, Google Pub/Sub) and stream processors (Spark Streaming, Apache Flink).

- Used for **real-time inference** and online learning, e.g. fraud detection, live recommendations, predictive maintenance alerts.
- Low latency, but more complex and needs always-on resources.

(Batch 1½ M, streaming 1½ M, diagram 1 M)

d) Factors for selection of a cloud ML platform.

4 M

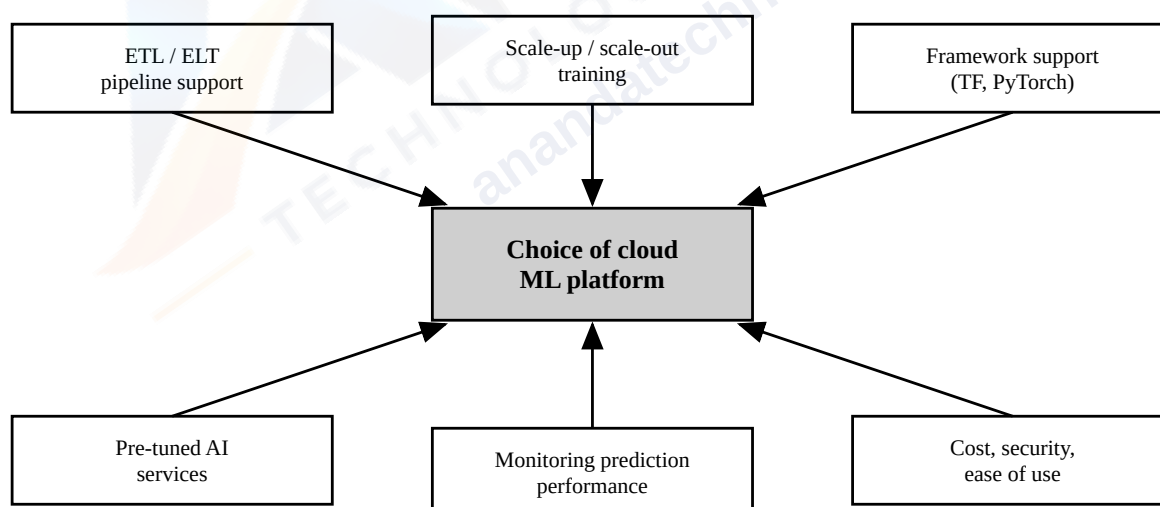


Fig.: Factors for selecting a cloud ML platform

1. **Support for ETL / ELT pipelines:** built-in tools to extract, clean and load data from many sources (AWS Glue, Azure Data Factory, Dataflow).
2. **Support for scale-up and scale-out training:** large GPU instances and distributed training across many nodes for big data and deep learning.

3. **Support for ML frameworks:** TensorFlow, PyTorch, scikit-learn, XGBoost, plus custom containers and notebooks.
4. **Pre-tuned AI services:** ready APIs (vision, speech, NLP) to save development time.
5. **Monitoring prediction performance:** tools to detect drift, track accuracy and trigger retraining.
6. **Other factors:** cost / pricing model, security and compliance, ease of use (AutoML, low-code), integration with existing data and cloud.

(Any four factors, 1 M each)

Q.3

a) Components of cloud computing architecture.

4 M

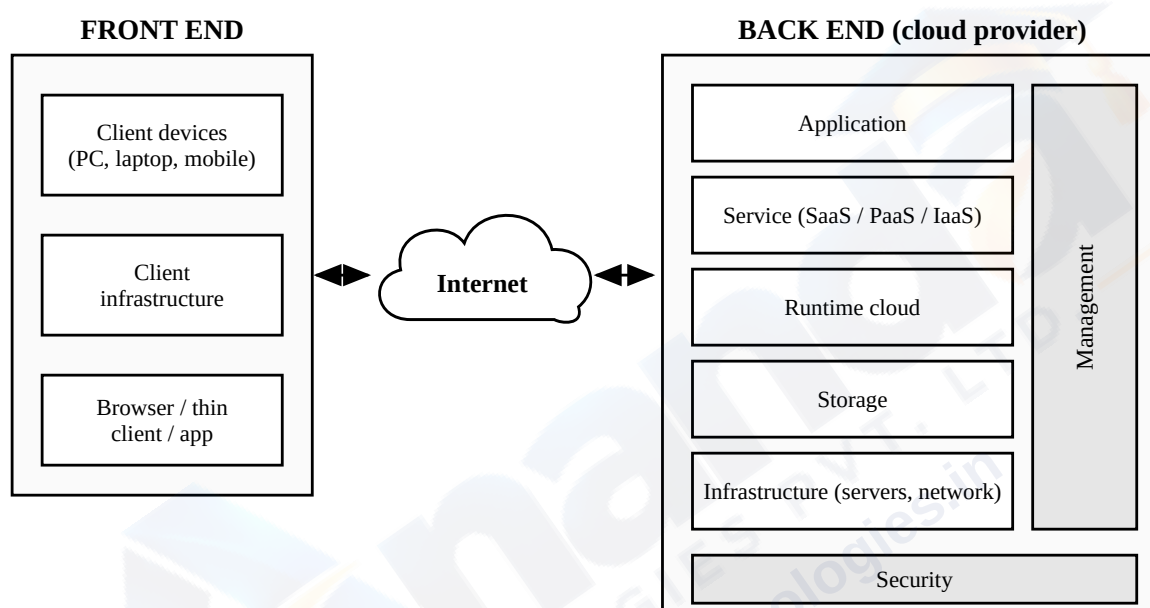


Fig.: Architecture of cloud computing (front end and back end)

1. **Front end (client platform):** the user side — client devices (desktop, laptop, mobile, thin client) and the interfaces / applications (web browser, mobile app, CLI) used to access cloud services.
2. **Back end (cloud platform):** owned by the provider — application, service (SaaS / PaaS / IaaS), runtime cloud, storage, infrastructure (servers, virtualization), management and security components that actually deliver the service.
3. **Network:** the Internet / intranet / WAN that connects front end and back end; includes load balancers, DNS and secure connections (HTTPS, VPN).
4. **Cloud-based delivery:** the services are delivered over the network as on-demand services — software (SaaS), platform (PaaS), infrastructure (IaaS) — through public, private or hybrid clouds.

(Diagram 1 M, four components 3 M)

b) Architecture of network virtualization.

4 M

Network virtualization combines physical network resources and network functions into software-based **virtual networks**. Many isolated virtual networks can run on one shared physical network.

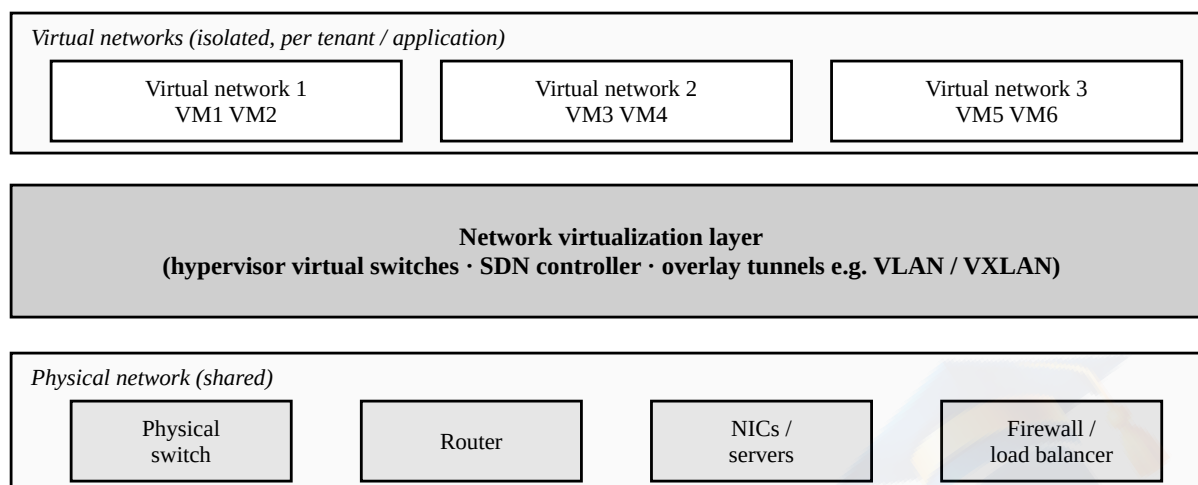


Fig.: Architecture of network virtualization

- Physical network layer:** real switches, routers, NICs, cables, firewalls — the shared underlay.
- Network virtualization layer:** software that abstracts the hardware — virtual switches in the hypervisor (vSwitch), an **SDN controller** that centrally controls traffic, and overlay protocols such as **VLAN / VXLAN** that create logical networks.
- Virtual networks:** each tenant / application gets its own isolated network with virtual switches, routers, firewalls and load balancers, connecting its VMs or containers.
- Types:** *external* (many physical networks combined or one LAN divided into VLANs) and *internal* (a network inside one server between VMs). Benefits: isolation, fast setup, better utilisation, central management (e.g. VMware NSX, AWS VPC).

(Diagram 2 M, explanation 2 M)

c) Block vs file vs object storage.

4 M

Block storage	File storage	Object storage
Fixed-size blocks raw volume, OS formats it SAN (iSCSI / FC)	Folders & files hierarchical paths NAS (NFS / SMB)	Object = data + metadata + unique ID REST API (HTTP)
e.g. Amazon EBS	e.g. Amazon EFS	e.g. Amazon S3

Fig.: Block, file and object storage

Point	Block-based	File-based	Object-based
Data unit	Fixed-size blocks	Files in folders	Objects (data + metadata + ID)
Structure	No structure; OS formats volume	Hierarchical directory tree	Flat namespace (buckets)

Access	iSCSI / Fibre Channel (SAN)	NFS / SMB (NAS)	HTTP REST API
Metadata	None	Limited (name, date, size)	Rich, custom metadata
Performance	Lowest latency	Medium	Higher latency, huge throughput
Scalability	Limited by volume size	Moderate	Practically unlimited
Use	Databases, VM disks	Shared files, home folders	Backups, media, data lakes, ML data sets
Example	Amazon EBS	Amazon EFS, Azure Files	Amazon S3, Azure Blob

(Any four points, 1 M each)

d) Functionality of Azure Machine Learning Studio.

4 M

Azure Machine Learning Studio is the web portal (ml.azure.com) of the Azure Machine Learning service, used by data scientists to manage the complete ML life cycle inside an **Azure ML workspace**.

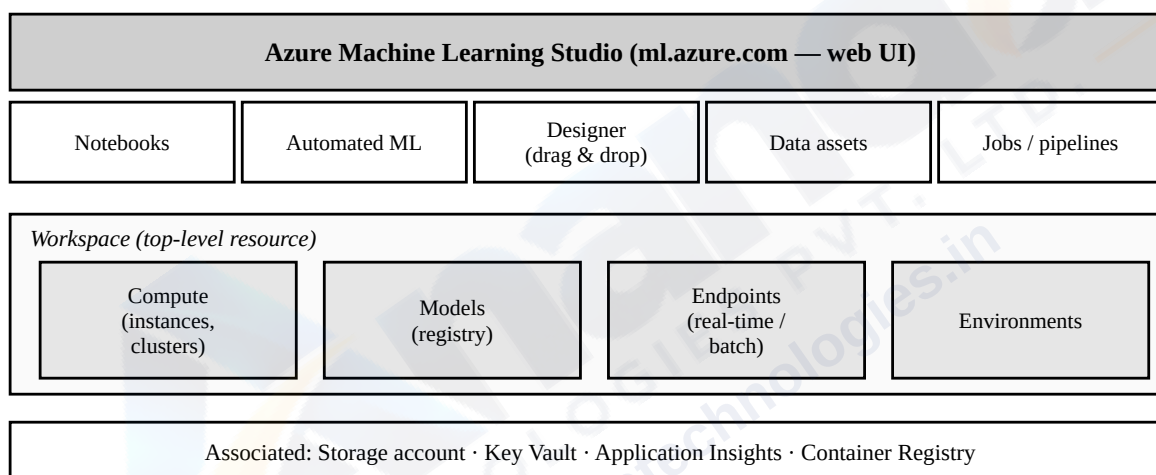


Fig.: Azure Machine Learning workspace and Studio

1. **Authoring:** *Notebooks* (Jupyter, code-first), *Designer* (drag-and-drop pipelines, low-code) and *Automated ML* (no-code: tries many algorithms and hyper-parameters automatically).
2. **Data and compute management:** register *data assets* / datastores; create *compute instances* (personal dev VM) and *compute clusters* (auto-scaling CPU / GPU nodes for training).
3. **Training and tracking:** run *jobs* and *pipelines*; experiments record metrics, logs and outputs (MLflow-compatible).
4. **Deployment and MLOps:** register models in the *model registry*, deploy to *real-time or batch endpoints*, monitor models, Responsible-AI dashboard, role-based access.

(Introduction 1 M, functionality 3 M)

Q.4

a) Evolution from mainframes to cloud; utility computing.

4 M

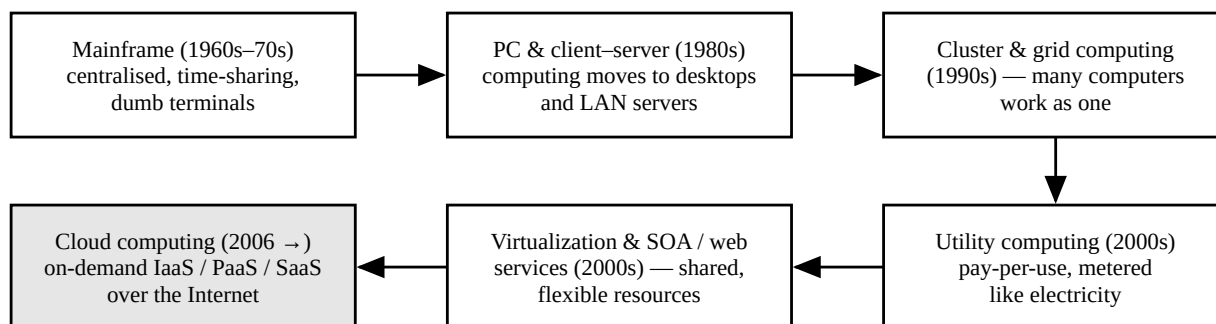


Fig.: Evolution of computing from mainframes to cloud

- 1. Mainframe era:** one large central computer shared by many users through dumb terminals (time-sharing); costly and centralised.
- 2. PC and client-server era:** processing moved to personal computers and LAN servers; each organisation bought its own hardware.
- 3. Cluster and grid computing:** many computers linked to act as one powerful system; grid connects resources across organisations.
- 4. Utility computing, virtualization and SOA:** resources offered as metered services; virtualization made sharing of hardware easy; web services made software reusable.
- 5. Cloud computing:** combines all the above — on-demand, self-service, elastic IaaS / PaaS / SaaS over the Internet.

Utility computing: a model in which computing resources (processing, storage) are provided as a metered service, like electricity or water — the user pays only for the amount used.

(Evolution 3 M, utility computing 1 M)

b) Role of cloud computing in data science.

4 M

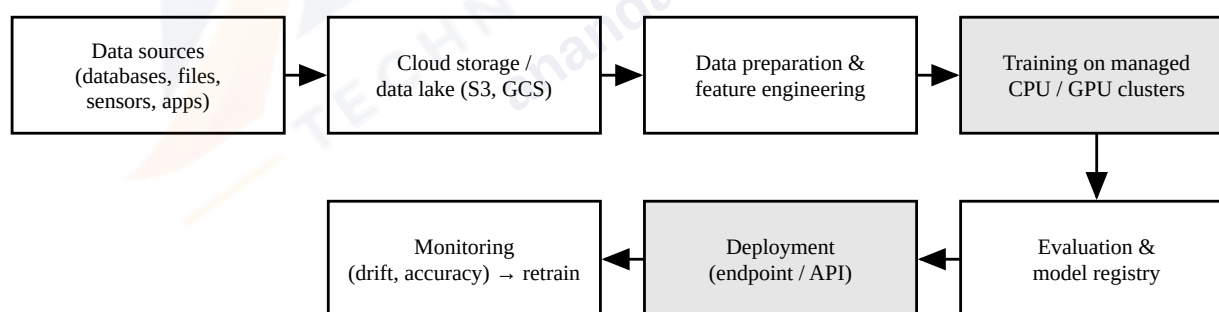


Fig.: Machine learning in the cloud — overview of an ML system

- 1. Data storage and management:** cheap, unlimited object storage, data lakes and warehouses (S3, BigQuery, Redshift) hold structured and unstructured big data.
- 2. Data processing:** managed Spark / Hadoop (EMR, Dataproc), ETL tools and serverless SQL clean and transform data at scale.
- 3. Compute for ML:** on-demand CPUs, GPUs and TPUs for training complex models; pay only while used.

4. **Tools and collaboration:** hosted Jupyter notebooks (Colab, SageMaker Studio, Azure ML) shared by the whole team; version control of data and models.
5. **Deployment and monitoring:** models served as APIs, scaled automatically and monitored; results visualised in BI dashboards.

(Any four roles, 1 M each)

c) Google BigQuery for ML.

4 M

Google BigQuery is a fully managed, **serverless, petabyte-scale data warehouse** on Google Cloud Platform used to store and analyse huge data sets with standard SQL. It is a key cloud data-science tool.

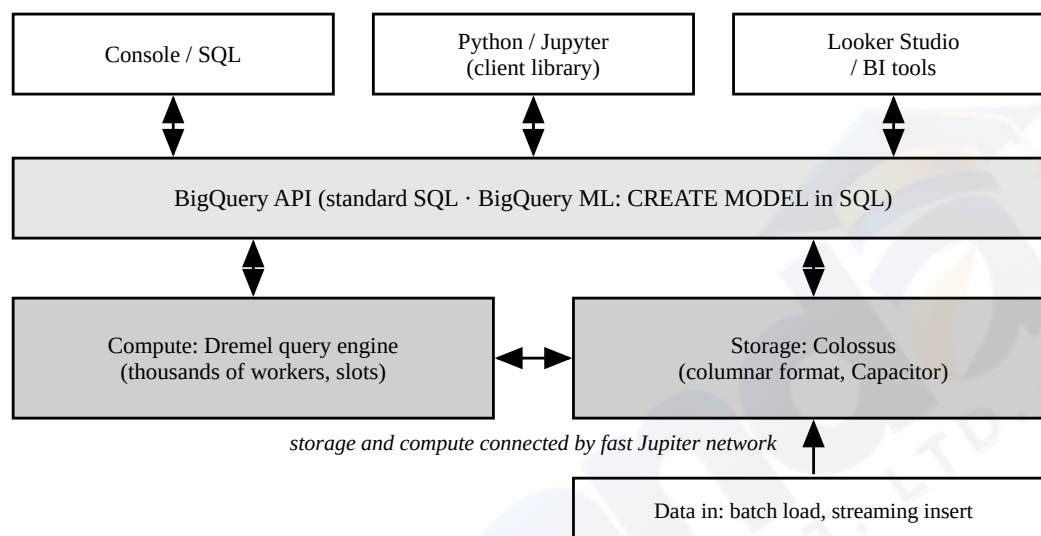


Fig.: Google BigQuery — serverless data warehouse (storage and compute separated)

1. **Serverless:** no servers or clusters to manage; Google allocates compute (slots) automatically.
2. **Separation of storage and compute:** data stored in columnar format on Colossus; queries run by the Dremel engine on thousands of workers — fast scans of terabytes in seconds.
3. **BigQuery ML:** create and run ML models (linear / logistic regression, k-means, time-series ARIMA_PLUS, boosted trees, imported TensorFlow models) using SQL — e.g. `CREATE MODEL ... OPTIONS(model_type='logistic_reg')`, then `ML.PREDICT`.
4. **Integration:** streaming inserts for real-time data, Python client library / Jupyter notebooks, Looker Studio dashboards, Vertex AI; pricing per bytes scanned (on-demand) or reserved capacity.

(Introduction 1 M, diagram 1 M, features / ML use 2 M)

d) Services provided by AIaaS.

4 M

AI as a Service (AIaaS) means ready-to-use artificial-intelligence capabilities provided by cloud vendors through APIs and managed platforms, on a pay-per-use basis — so companies can add AI without building everything themselves.

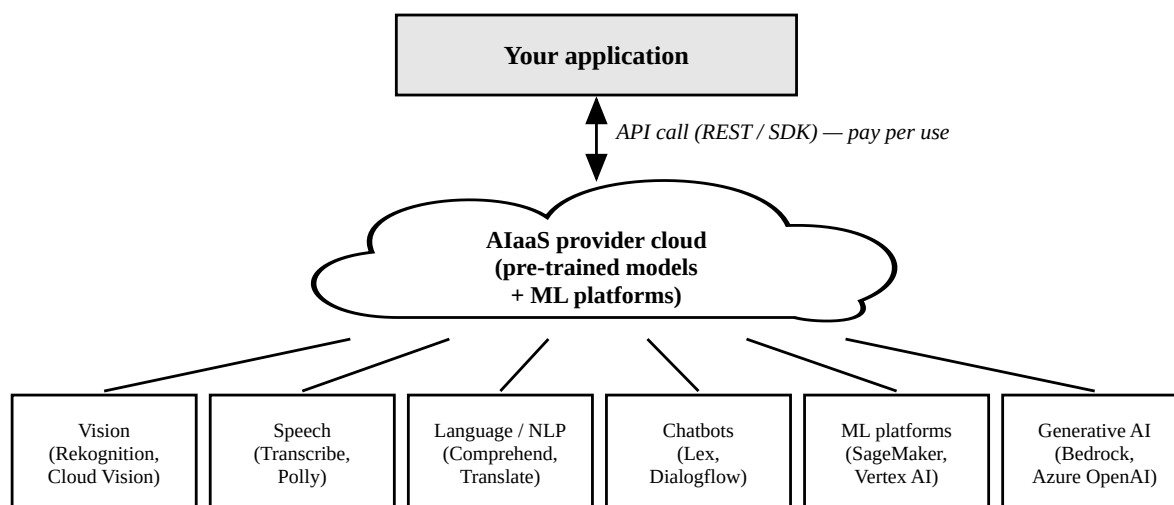


Fig.: Artificial Intelligence as a Service (AIaaS)

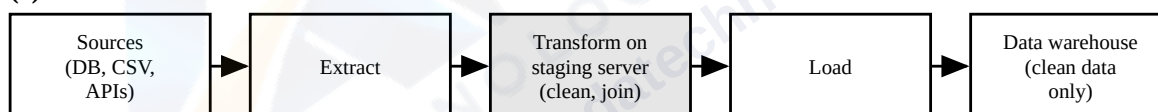
1. **Computer vision:** image labelling, face and object detection, OCR (Amazon Rekognition, Google Cloud Vision, Azure AI Vision).
2. **Speech and language:** speech-to-text, text-to-speech, translation, sentiment and entity analysis (Amazon Transcribe / Polly / Comprehend / Translate, Google Speech-to-Text, Natural Language API).
3. **Conversational AI (chatbots):** Amazon Lex, Google Dialogflow, Azure Bot Service.
4. **Machine learning platforms and AutoML:** SageMaker, Azure ML, Vertex AI to build custom models; **generative AI** foundation models (Amazon Bedrock, Azure OpenAI Service, Vertex AI Gemini); also recommendation and forecasting services.

(Concept 1 M, any three services 3 M)

e) ETL vs ELT.

4 M

(a) ETL — Extract → Transform → Load



(b) ELT — Extract → Load → Transform

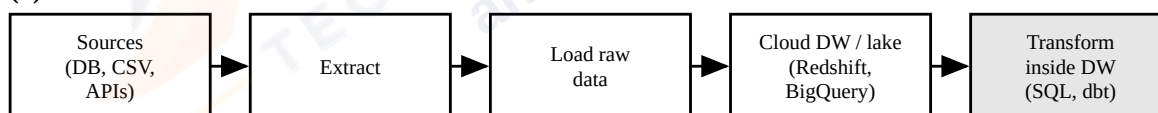


Fig.: ETL vs ELT pipeline

Point	ETL (Extract–Transform–Load)	ELT (Extract–Load–Transform)
Order	Transform before loading	Load raw data first, transform later
Where transform happens	Separate staging / ETL server	Inside the target cloud warehouse / data lake
Data loaded	Only cleaned, structured data	All raw data (structured + unstructured)
Speed for big data	Slower; staging server is a bottleneck	Faster; uses scalable warehouse compute

Flexibility	New transformation needs re-extraction	Raw data kept, can re-transform any time
Best suited	On-premises DW, sensitive data to be masked before load	Cloud DW — Redshift, BigQuery, Snowflake
Tools	Informatica, Talend, SSIS, AWS Glue	dbt, SQL in BigQuery / Redshift, Azure Synapse

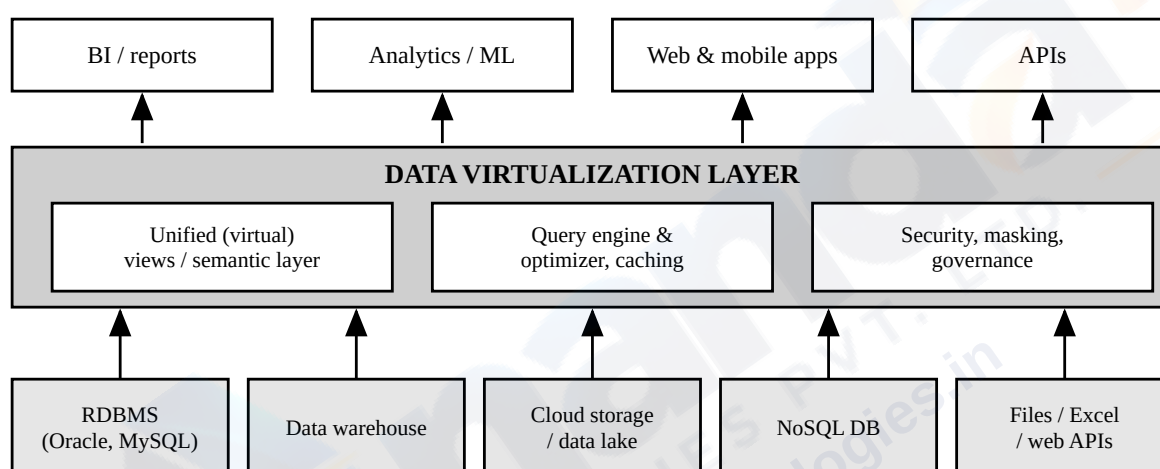
(Any four points, 1 M each)

Q.5

a) Data virtualization.

6 M

Data virtualization is a data-management approach that creates a **single, virtual (logical) layer** over many different data sources so that users and applications can query all data **as if it were in one place — without physically copying or moving it**.



Data stays in its source; connectors fetch it in real time when queried

Fig.: Data virtualization — one virtual data layer over many sources

Working / components:

1. **Connectors** link to sources — RDBMS, data warehouses, cloud storage, NoSQL, files, web APIs.
2. **Abstraction (semantic) layer** builds unified virtual views / tables combining data from these sources.
3. **Query engine** receives a query on the virtual view, splits it into sub-queries for each source, optimises and pushes them down, then combines results; **caching** improves speed.
4. **Security and governance** — single point for access control, data masking and lineage.

Benefits: real-time access to current data; no duplicate copies and less ETL; faster integration and lower cost; one consistent view for BI and ML; central security. Tools: Denodo, TIBCO Data Virtualization, IBM Cloud Pak for Data.

Real-world application (e-commerce company): customer data is in a CRM, orders in MySQL, website clicks in a cloud data lake and delivery status in a partner's API. A data-virtualization layer gives one virtual *Customer 360* view; the marketing team's dashboard and the recommendation ML

model query this view directly, always getting up-to-date data without building a separate data warehouse.

(Concept 1 M, diagram 1 M, working 2 M, benefits 1 M, application 1 M)

b) Amazon Redshift architecture, launching a cluster and loading from S3.

6 M

A **cloud data warehouse** is a managed, scalable database in the cloud optimised for analytical (OLAP) queries on large historical data. **Amazon Redshift** is AWS's cloud data warehouse.

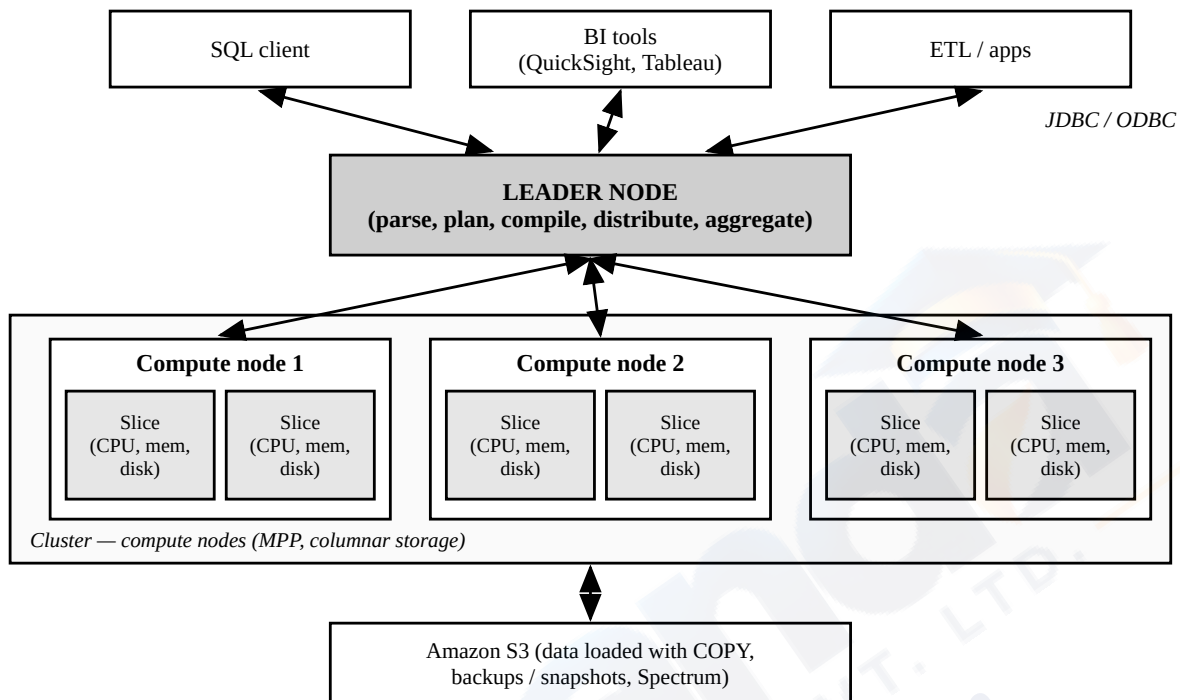


Fig.: Amazon Redshift architecture

Architecture:

1. **Client applications** — SQL clients, BI tools and ETL tools connect using JDBC / ODBC drivers.
2. **Leader node** — receives queries, parses them, prepares the execution plan, compiles code, distributes work to compute nodes and aggregates the final result. Stores metadata.
3. **Compute nodes** — execute the compiled code in parallel (MPP) and return intermediate results; data stored in **columnar**, compressed form.
4. **Node slices** — each compute node is partitioned into slices, each with its own share of CPU, memory and disk; table rows are distributed across slices (distribution key) so all slices work in parallel.

Steps to launch a cluster and load data from S3:

1. Sign in to the AWS Management Console and open **Amazon Redshift**; choose **Create cluster** (or a Redshift Serverless workgroup).
2. Enter cluster identifier, choose **node type** and **number of nodes**; set admin user name and password; database name and port (5439).
3. Configure network and security — VPC, subnet group, security group (allow port 5439), public accessibility; **attach an IAM role** that allows reading from Amazon S3.

4. Click **Create cluster** and wait until status becomes *Available*; note the endpoint.
5. Connect with **Query Editor v2** or a SQL client (JDBC / ODBC endpoint) and create tables (CREATE TABLE sales (...)).
6. Upload data files to an S3 bucket and load them in parallel with COPY sales FROM 's3://bucket/sales/' IAM_ROLE '<role-arn>' FORMAT AS CSV; then run SQL queries / connect BI tools.

(Architecture with diagram 3 M, steps 3 M)

c) Google Cloud AutoML — custom image classification.

6 M

Google Cloud AutoML (now part of **Vertex AI**) lets developers with limited ML knowledge train **high-quality custom models** on their own data. AutoML automatically performs model architecture search, transfer learning and hyper-parameter tuning, and gives a model ready to deploy.

Functionalities: AutoML for images (classification, object detection), video, text (classification, entity extraction, sentiment), translation and tabular data (classification / regression / forecasting); graphical console + REST API / SDK; model evaluation reports; one-click deployment to endpoints or export to edge devices.

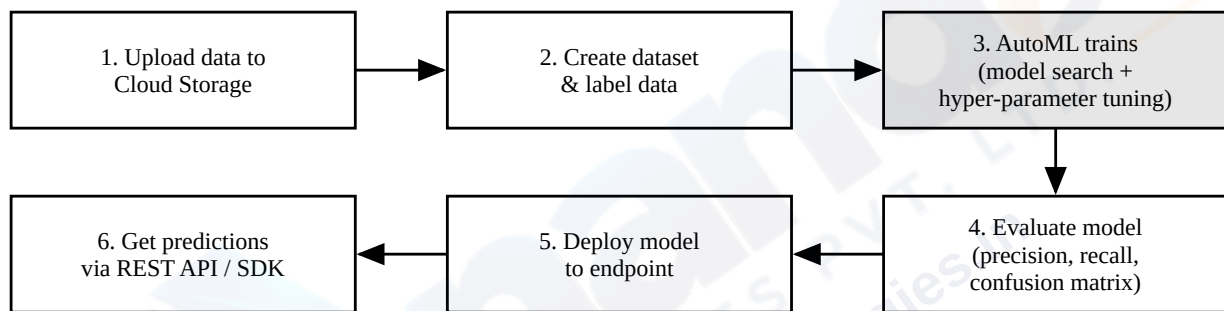


Fig.: Google Cloud AutoML (Vertex AI) workflow

Workflow to build an image classifier (e.g. 'ripe' vs 'unripe' mango):

1. **Prepare data:** collect labelled images of each class (Vertex AI needs at least 10 images per label and recommends about 1,000 per label for best results) and upload them to a **Cloud Storage** bucket; create an import CSV / JSONL file with image URI and label.
2. **Create dataset:** in Vertex AI console create an *Image* dataset, objective *single-label classification*, and import the files; check / correct labels; data is split into training, validation and test sets.
3. **Train:** choose *AutoML* training, set a node-hour budget; AutoML searches and trains the best model automatically.
4. **Evaluate:** review precision, recall, confusion matrix and per-label results; add more data and retrain if needed.
5. **Deploy:** deploy the model to an **endpoint** (or export for edge / mobile).
6. **Predict:** send new images through the console, REST API or Python SDK and receive label and confidence score; monitor and retrain with new images.

(Concept & functionalities 2 M, workflow with diagram 4 M)

Q.6

a) Key features of Amazon SageMaker.

6 M

Amazon SageMaker is AWS's fully managed machine-learning service that covers the whole ML life cycle — prepare, build, train, deploy and monitor.

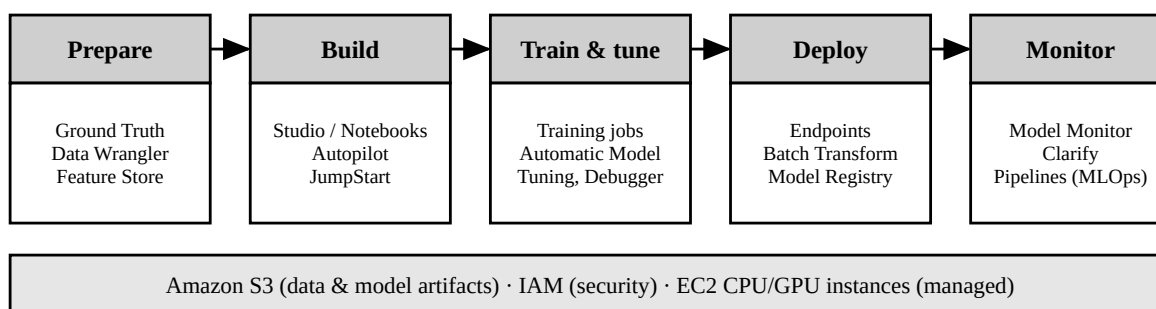


Fig.: Amazon SageMaker — components across the ML workflow

- 1. Integrated development environment:** SageMaker Studio with Jupyter notebooks, experiment tracking and visual tools.
- 2. Data preparation and labelling:** Data Wrangler (import, clean, visualise data), Ground Truth (labelling), Feature Store.
- 3. Built-in algorithms and framework support:** XGBoost, linear learner, k-means, image classification, etc.; bring-your-own TensorFlow, PyTorch, scikit-learn, Hugging Face code or containers.
- 4. Managed, scalable training:** one API call launches training on chosen CPU / GPU instances; distributed training; managed spot training to cut cost; instances stop automatically after training.
- 5. Automatic model tuning and AutoML:** hyper-parameter optimisation; Autopilot builds models automatically from tabular data; JumpStart gives pre-trained models.
- 6. Easy deployment:** real-time endpoints with auto-scaling, serverless and asynchronous inference, Batch Transform; A/B testing of model variants.
- 7. Monitoring and governance:** Model Monitor (data and quality drift), Clarify (bias, explainability), Model Registry and Pipelines for MLOps.
- 8. Security and pay-as-you-go:** IAM, VPC, encryption with KMS; pay only for resources used.

(Diagram 1 M, any five features 5 M)

b) Training a plant-leaf disease CNN on a cloud GPU instance.

6 M

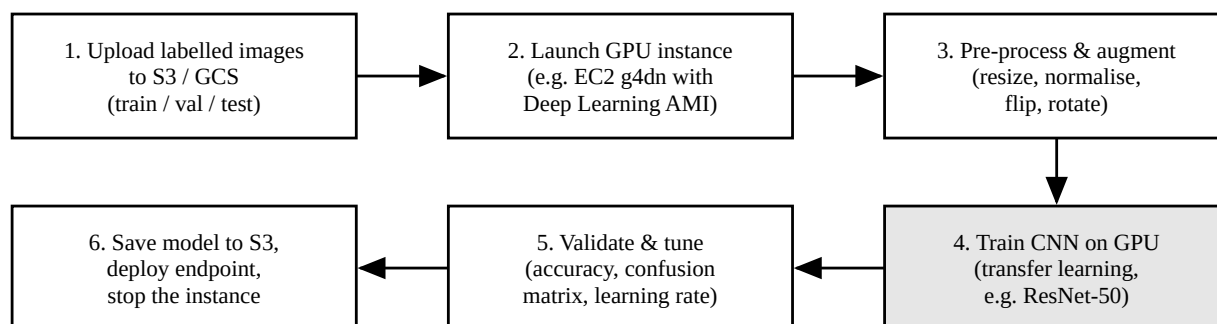


Fig.: Training an image classifier on a cloud GPU instance

- 1. Collect and upload data:** gather labelled leaf images (e.g. healthy, leaf-blight, rust) in class folders; split into train / validation / test (e.g. 70 / 15 / 15); upload to an object-storage bucket (Amazon S3 / Google Cloud Storage).
- 2. Launch a GPU instance:** choose a GPU instance type (e.g. AWS EC2 g4dn.xlarge with NVIDIA T4, or a Google Cloud VM with a T4 / L4 GPU) using a **Deep Learning AMI** / **image** that already has CUDA, cuDNN, PyTorch / TensorFlow; open SSH / Jupyter access; use spot instances to save cost.
- 3. Set up environment and data pipeline:** copy data from the bucket to the instance (or stream it); verify GPU with `nvidia-smi`; write a data loader that resizes images (e.g. 224×224), normalises them and applies augmentation (flip, rotate, zoom) to reduce over-fitting.
- 4. Build and train the model:** use **transfer learning** — a pre-trained CNN such as ResNet-50 / MobileNet with a new final layer for the disease classes; choose loss (cross-entropy), optimiser (Adam) and batch size; train for several epochs on the GPU, saving checkpoints to the bucket.
- 5. Validate and tune:** check validation accuracy / loss each epoch, plot confusion matrix; tune learning rate, epochs, unfreeze more layers; finally measure accuracy on the test set.
- 6. Save, deploy and clean up:** export the trained model to the bucket; deploy it as an API endpoint (e.g. SageMaker endpoint / Vertex AI endpoint or a Flask app) for the mobile app to call; monitor predictions; **stop / terminate the GPU instance** to avoid charges.

(Diagram 1 M, steps 5 M)

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

6 M

Pre-tuned AI services are cloud AI APIs backed by models that the provider has already trained and fine-tuned on huge data sets (e.g. Amazon Rekognition, Comprehend, Transcribe, Google Cloud Vision, Azure AI services).

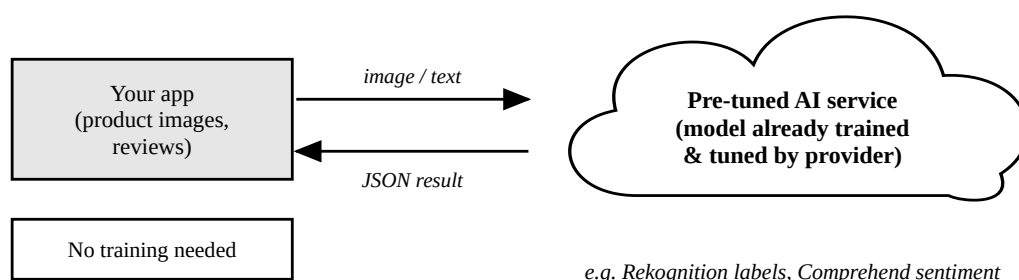


Fig.: Using a pre-tuned (pre-trained) AI service

Key features:

1. **No ML expertise or training data needed** — call an API and get results (labels, text, sentiment).
2. **Fast integration** through REST APIs and SDKs; results in milliseconds.
3. **Scalable and highly available** — provider handles infrastructure and scaling.
4. **Pay per use** (per image / per 1,000 characters / per minute of audio) and continuously improved by the provider; some allow customisation (custom labels / custom classifiers).

Monitoring prediction performance of a deployed model:

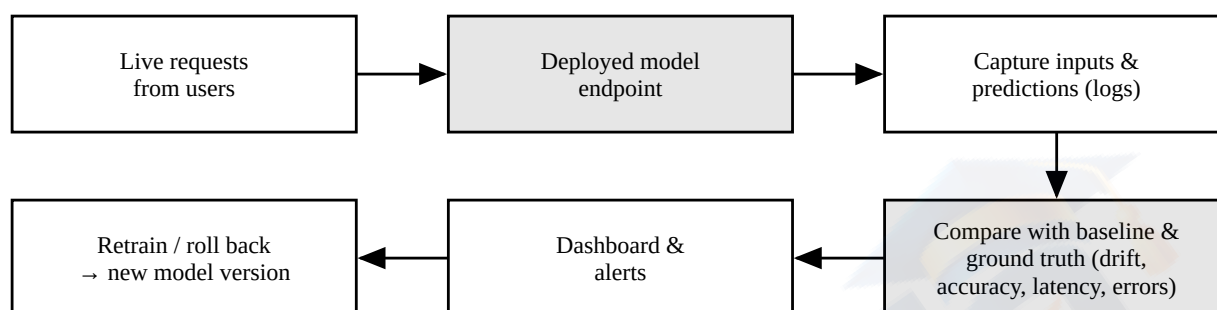


Fig.: Monitoring prediction performance of a deployed model

1. **Capture data:** log every request input, prediction and confidence (e.g. SageMaker data capture).
2. **Baseline:** compute statistics of the training data to compare against live data.
3. **Data drift and concept drift detection:** check if input distributions or the input–output relationship have changed over time.
4. **Quality metrics:** when ground truth arrives, compute accuracy, precision, recall, F1, RMSE; also latency, error rate, throughput and cost.
5. **Alerts and dashboards:** CloudWatch / Azure Monitor / Vertex AI Model Monitoring raise alarms when metrics cross thresholds.
6. **Action:** retrain with fresh data, roll back to a previous version, or A/B test a new model.

(Features 3 M, monitoring 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