# High‑Level Design: Amazon SageMaker, AWS Bedrock, and Amazon S3 (ML)

**Target readers:** Cloud architects, ML engineers, platform engineers, security & compliance teams, DevOps engineers, and stakeholders involved in ML platform decisions.

## 1. Purpose

This document presents a high‑level design for an integrated Machine Learning platform using Amazon SageMaker, AWS Bedrock, and Amazon S3. It explains how these services interconnect to support the full ML lifecycle: data ingestion and storage, experimentation and development, model training, model management and serving, and generative AI capabilities. It includes architecture guidance, components per AWS region, observability and backup/restore recommendations, and common platform services.

## 2. Audience

* Cloud and platform architects designing ML platforms
* ML engineers and data scientists who will use the platform
* DevOps and SRE teams responsible for deployment and operations
* Security, compliance, and governance stakeholders
* Project managers and technical product owners

## 3. Introduction

Modern ML systems require scalable, secure storage of data and artifacts, flexible training and serving infrastructure, and the ability to incorporate foundation models or generative AI. This design uses Amazon S3 as the canonical data and artifact store, Amazon SageMaker for experimentation, training, model registry and serving, and AWS Bedrock where foundation-model capabilities (LLMs and multimodal models) are needed. The three services are integrated to deliver reproducible pipelines, secure data handling, and observable operations.

Key objectives: - Single source of truth for data and artifacts (S3) - Repeatable and auditable training pipelines (SageMaker Pipelines / SageMaker Projects) - Access to managed foundation models for generative tasks (Bedrock) - Secure, least‑privilege access across services - Observability, backup and restore, and cost controls

## 4. Architecture and Design

**1. Data Layer (Amazon S3)**

* Raw data ingested from multiple sources (CSV, JSON, Parquet, Images, etc.).
* Organized into prefixes:
  + /raw/ → incoming datasets
  + /curated/ → cleaned, pre-processed datasets
  + /splits/ → training, validation, test datasets
  + /artifacts/ → model outputs, logs, metrics

**2. Experimentation & Development (SageMaker Studio)**

* Data scientists launch **Studio notebooks**.
* Read/write directly from **S3 buckets**.
* Perform **feature engineering**, **EDA**, and small-scale prototyping.
* Git integration ensures reproducibility and version control.

**3. Training & Pipelines (SageMaker Pipelines + Training Jobs)**

* Pipelines orchestrate:
  + Data preprocessing (SageMaker Processing jobs).
  + Model training (managed Training jobs on Spot/On-Demand instances).
  + Model evaluation.
  + Registration into the **Model Registry**.
* Outputs (trained models, logs, metrics) are persisted back into **S3**.

**4. Model Registry & Serving (SageMaker)**

* Trained models go into **Model Registry**.
* Promotion lifecycle: *Staging → Approved → Production*.
* Deployment options:
  + **Real-time inference** (SageMaker Endpoints).
  + **Batch inference** (Batch Transform).
  + **Async inference** (asynchronous endpoints).

**5. Foundation Models (Bedrock Integration)**

* For tasks requiring **LLMs / Foundation Models** (e.g., summarization, text generation, semantic search):
  + Applications call **Bedrock APIs** directly.
  + Custom fine-tuned models from SageMaker may integrate with Bedrock workflows.
* Traffic to Bedrock flows securely via **VPC endpoints + API Gateway**.

**6. Feature Store & Online Serving**

* Optional **SageMaker Feature Store**:
  + **Offline store** → backed by **S3** for training.
  + **Online store** → backed by **DynamoDB** for low-latency inference features.

**7. Orchestration & CI/CD**

* Infra-as-Code: **Terraform / CloudFormation**.
* CI/CD pipelines via **CodePipeline + CodeBuild** (or Jenkins).
* Automates deployment of pipelines, models, and infrastructure.

**8. Security & Governance**

* **IAM roles** per service (least privilege).
* **KMS** for encrypting S3 buckets, EBS volumes, and model artifacts.
* **VPC isolation** for training and inference jobs.
* **CloudTrail + AWS Config** for governance and auditing.

**9. Observability**

* **CloudWatch** for logs, metrics, alarms (pipelines, training jobs, inference).
* **S3 lifecycle policies** for data archival and backup.
* **EventBridge** triggers for retraining or monitoring data drift.

📌 **End-to-End Flow (Summary):**  
**Raw Data (S3) → SageMaker Studio (EDA/Dev) → Pipelines (Training/Processing) → Model Registry → Endpoints / Batch Inference → Bedrock (LLM augmentation) → Observability (CloudWatch, Config, CloudTrail).**

### 4.3 Network & VPC considerations

* Use VPC endpoints for S3, SageMaker, and Bedrock (where supported) to avoid public internet egress.
* Deploy SageMaker training and endpoints into private subnets when needed for data residency.
* For Bedrock, if using private network options, route calls through a hardened bastion or API Gateway with private integration—ensure encryption in transit.

### 4.4 Security & IAM

* Principle of least privilege: create narrowly scoped IAM roles for SageMaker execution, Bedrock access, and pipeline runners.
* Use KMS CMKs for S3 encryption, and enable object-level logging via CloudTrail Data Events.
* Implement attribute-based access control (ABAC) or tag-based policies to separate environments (dev/staging/prod).
* Use AWS Secrets Manager or Parameter Store for credentials and model keys.

### 4.5 Scalability & Cost controls

* Use managed spot instances for training when appropriate to reduce cost.
* Configure S3 lifecycle (transition to infrequent access/Glacier) for older artifacts.
* Use SageMaker Serverless Endpoints or multi‑model endpoints for infrequently used models to lower serving costs.
* Monitor cost using AWS Cost Explorer and set up budgets and alerts.

### 4.6 Resilience & Recovery

* Keep multiple copies of critical artifacts across regions if required for DR (cross‑region replication for S3).
* Versioned artifacts and model registry enable rollback to previous model versions.

## 5. List of Components & Services (example per AWS region)

The list below is an example mapping; exact services used may vary by org and compliance requirements. Replace <region> with the AWS region (e.g., us-east-1, eu-west-1, ap-south-1).

**Core ML components** - Amazon S3 (s3): canonical data & artifacts store (buckets: <org>-ml-raw-<region>, <org>-ml-curated-<region>, <org>-ml-artifacts-<region>) - Amazon SageMaker (sagemaker): Studio, TrainingJobs, ProcessingJobs, Model Registry, Endpoints, Feature Store - AWS Bedrock (bedrock): foundation model access (LLMs)

**Supporting AWS services** - Amazon VPC: private networking and endpoints - AWS IAM & AWS KMS: security and encryption - AWS Lambda / Step Functions: orchestration for ingestion - AWS CodePipeline / CodeBuild / CodeDeploy or GitHub Actions: CI/CD - Amazon CloudWatch: logs & metrics - AWS CloudTrail: audit & API logging (enable Data Events for S3) - AWS Config: governance - Amazon ECR: container images for training and inference - Amazon EFS / FSx (optional): for shared file system support in distributed training - Amazon DynamoDB (optional): backing store for online feature store - Amazon OpenSearch / Amazon Managed Service for Grafana: log analytics and dashboards - AWS SNS / SQS: notification & queuing

**Optional / advanced** - Amazon SageMaker Neo: optimized model compilation - Amazon SageMaker Edge Manager: edge deployment - AWS Glue / AWS DataBrew: data catalog and transformations - AWS Backup: centralized backup of EFS and other supported services

## 6. Screenshots (recommended list and placement)

*This section contains placeholders for recommended screenshots. For a production‑ready document, capture these from your AWS Console or ask for realistic mock console screenshots to be generated and embedded.*

Suggested screenshots (one image per bullet): - S3 bucket structure showing prefixes for raw/, curated/, artifacts/. - SageMaker Studio main UI (notebook/workspace view). - Example SageMaker Pipeline run (visualization of pipeline steps and status). - SageMaker Model Registry entry (model versions and approvals). - SageMaker Endpoint configuration (instance type, autoscaling panel). - Bedrock console or Bedrock API request example (if available in region). - CloudWatch dashboard showing training metrics, endpoint latency, and errors. - IAM role policy snippet for SageMaker execution role (redacted / masked).

*Screenshot guidance:* - Redact any PII, account IDs, or secrets. For presentation use realistic mock screenshots showing the same UI layout but with fictional data. - Capture console pages with clear labels and add callouts: dataset, model version, metrics, and alert thresholds.

## 7. Observability (Monitoring, Logging) & Backup/Restore

### 7.1 Monitoring & Logging

* **CloudWatch Metrics & Logs:** configure SageMaker to push training job logs and endpoint logs to CloudWatch. Create dashboards for training loss/accuracy curves, endpoint latency, error rates, and instance health.
* **Model Telemetry:** emit custom metrics from training/inference code (through CloudWatch PutMetric) for feature drift, data quality, and prediction distributions.
* **Structured Logs & Tracing:** use structured JSON logs (request\_id, model\_version, input\_hash, latency\_ms) and enable X‑Ray tracing for request flows where appropriate.
* **Alerting:** set alarms on metric thresholds (endpoint 5xx errors, sustained high latency, drift alerts) and integrate alerts with SNS, PagerDuty, or Slack.
* **Audit & Lineage:** enable CloudTrail with S3 data events and SageMaker API logging to capture who ran which pipeline, which artifacts were created, and when model promotions occurred.

### 7.2 Backup & Restore

* **S3 best practices:** enable S3 versioning for artifact buckets and configure cross‑region replication for critical buckets when regional DR is required.
* **Automated snapshots:** for EFS or FSx used in training, schedule AWS Backup snapshots.
* **Model registry backup:** export model artifacts (tarball) and metadata snapshots to a dedicated backup bucket. Regularly export model metadata (approvals, tags) via automation to store outside the primary account if required.
* **Restore procedures:** maintain runbooks that describe steps to restore datasets (from versioned objects), re-register models, and re-deploy endpoints using IaC templates.
* **Validation:** after restore, run smoke tests (sample inference, metrics smoke) to confirm integrity.

## 8. Common Platform Services & Patterns

* **Identity & Access Management:** central IAM policies and role‑assumption model for cross‑account access. Use AWS Organizations SCPs for guardrails.
* **Infrastructure as Code:** manage resources using Terraform modules (S3, SageMaker roles, VPC endpoints) and store code in version control with PR reviews.
* **CI/CD for Models:** automate model build → test → deploy pipelines. Integrate model testing steps: unit tests, integration tests, performance tests, and canary deployments.
* **Data Catalog & Governance:** maintain a data catalog (Glue Data Catalog) and classify sensitive datasets. Integrate with Lake Formation if fine‑grained access control is needed.
* **Secrets Management:** use Secrets Manager for third‑party API keys and Parameter Store for less sensitive configuration.
* **Cost Governance:** tagging strategy for cost allocation, budgets, and automated spot/instance recommendations.
* **Standardized Notebook Environments:** prebuilt container images in ECR with common ML libraries, security scanners, and baseline tooling.
* **Model Explainability & Compliance:** integrate SHAP/LIME jobs into pipelines and store explainability artifacts in S3 for auditability.

## Appendix

### Example S3 prefixes (recommended)

s3://<org>-ml-raw-<region>/dataset-name/YYYY-MM-DD/  
s3://<org>-ml-curated-<region>/dataset-name/version=  
s3://<org>-ml-artifacts-<region>/models/<project>/<model-name>/v1/

### Minimal IAM role policy snippets (conceptual)

* SageMaker execution role allowing S3 read/write to artifact buckets and access to ECR and KMS (redact ARNs in real policies).

### Recommended runbooks

* Training job failure troubleshooting
* Model comparison and rollback
* Restoring S3 bucket from versioned objects

## Next steps & deliverables

1. Capture real AWS Console screenshots (or request realistic mock screenshots) and attach them to this document.
2. Review and map required compliance controls (HIPAA, PCI, etc.) if your workloads process regulated data.
3. Convert this into a formal Word document or slide deck for stakeholder review — I can export this into a .docx on request and embed screenshots.

*Document created: High‑Level Design for SageMaker + Bedrock + S3 (ML). Replace placeholders with environment‑specific values before publishing.*

# High‑Level Diagram — Flow, Components & Use Case

Below is a high‑level diagram (Mermaid) that shows the core components, data flow, network boundaries, and how SageMaker, Bedrock, and S3 interconnect. You can copy the Mermaid block into any Mermaid renderer (or a draw.io/diagrams.net import) to generate a visual. If you want, I can also export this diagram as an image and embed realistic mock console screenshots into the Word doc.

flowchart LR  
 subgraph OnPrem/External  
 A[Data Producers / Ingest] -->|ingest events/files| B[Ingestion API / Kinesis / Direct S3 Put]  
 end  
  
 subgraph AWS\_VPC\_Private  
 direction TB  
 B --> C[S3 Landing Bucket   
(raw) (s3://<org>-ml-raw-<region>)]  
 C --> D[Data Processing   
(Lambda / StepFunctions / Glue)]  
 D --> E[S3 Curated Bucket   
(curated, feature store snapshots)]  
  
 E --> F[SageMaker Studio / Notebooks]  
 F --> G[SageMaker Pipelines   
(Processing -> Training -> Evaluation)]  
 G --> H[SageMaker Model Registry]  
 H --> I[SageMaker Endpoint(s)   
(Real-time / Async / Batch)]  
  
 %% Feature store and online store  
 E --> J[SageMaker FeatureStore   
(Offline -> Online)]  
 J --> I  
  
 %% Bedrock integration  
 subgraph Bedrock\_Access  
 direction LR  
 K[AWS Bedrock (LLMs / FMs)]  
 end  
 F -->|calls / experiments| K  
 G -->|optional: use FM in pipeline| K  
 K -->|outputs stored| C  
  
 %% Observability and governance  
 I --> L[CloudWatch Metrics & Logs]  
 G --> L  
 H --> M[Model Artifacts (S3 artifacts bucket)]  
 M -->|versioning / replication| C  
  
 %% Security and infra  
 N[KMS / IAM / VPC Endpoints]  
 C --> N  
 G --> N  
 I --> N  
 end  
  
 style C fill:#f9f,stroke:#333,stroke-width:1px  
 style K fill:#efe,stroke:#333,stroke-width:1px  
 style I fill:#ff9,stroke:#333,stroke-width:1px

## Diagram notes (textual)

* **Ingestion**: Data producers push raw data to the S3 landing bucket via APIs (Kinesis, API Gateway) or direct S3 uploads. S3 events or Step Functions kick off processing jobs.
* **Processing**: Transform/clean jobs (Glue, Processing Jobs) write curated datasets to the curated S3 prefix.
* **Experimentation**: Data scientists use SageMaker Studio to iterate; notebooks read curated data from S3 and may call Bedrock for foundation-model experiments.
* **Pipelines & Training**: SageMaker Pipelines orchestrates preprocessing, training (spot/on‑demand), evaluation, and registers artifacts in the Model Registry.
* **Model Serving**: Approved models are deployed to SageMaker Endpoints (real‑time, async, batch). Endpoints query Feature Store for online features as needed.
* **Bedrock**: Used for LLM/MultiModal tasks; Bedrock responses can be stored back to S3 for lineage and retraining.
* **Observability**: CloudWatch, CloudTrail (S3 data events), and OpenSearch/Grafana handle logging, metrics, traces, and dashboards.
* **Security**: VPC endpoints for S3 and SageMaker, KMS for encryption, and least‑privilege IAM roles.

## Example overall usage case (Healthcare: Clinical NLP + Predictive Model)

1. **Use case**: Extract clinical insights from unstructured notes, augment structured EHR data, and build a model predicting 30‑day readmission risk.
2. **Flow**:
   * Clinical notes are ingested into S3 (raw). S3 event triggers an ETL job that extracts text and stores cleaned records in curated S3.
   * A SageMaker Pipeline preprocesses features (text embedding via Bedrock LLM or SageMaker embeddings), joins structured EHR features, and trains a predictive model.
   * The model is registered in SageMaker Model Registry, approved after evaluation and explainability checks (SHAP), then deployed to a secure SageMaker Endpoint.
   * Real‑time inference happens via the endpoint, which also logs predictions and model metadata to CloudWatch and S3 for lineage.
   * Bedrock is used selectively for de‑identification, summarization, or semantic embedding tasks where a foundation model offers better performance.
3. **Compliance notes**: Ensure PHI handling controls: VPC isolation, encryption, audit trails, and HIPAA BAAs where required.

If you’d like, I can now: - Export an updated **.docx** that includes this diagram as an embedded image and the updated sections, or - Generate realistic mock AWS Console screenshots (SageMaker Studio, S3 layout, Pipeline run, Bedrock request) and embed them into the Word document.

Tell me which of the two you want and I will produce the Word file with the diagram and/or screenshots embedded.