Title: Satellite Data Pipeline for Space Agencies

Subtitle: Scalable, Fault-Tolerant, and Secure Software Engineering Solution

Date:15/9/25

Abstract: This report presents the design and functionality of a large-scale data pipeline for space agencies, focused on processing, storing, and delivering satellite data ranging from terabytes to petabytes. The pipeline ensures scalability, high availability, fault tolerance, and secure access for mission-critical applications such as disaster monitoring, agriculture, and climate research. Real-world case studies from the European Space Agency’s Copernicus programme and the ExtremeEarth project are discussed to highlight practical implementations.

Introduction

3.1 Background

Satellites generate massive amounts of data daily (TB–PB scale), including imagery, telemetry, and sensor readings. Efficient pipelines are required to handle ingestion, transformation, storage, and secure delivery of this data.

3.2 Problem Statement

A large-scale data pipeline designed to process, store, and deliver satellite data ranging from terabytes to petabytes. It ensures scalability, high availability, and fault tolerance for mission-critical space applications.

3.3 Objectives

- Design a scalable, cloud-native pipeline for satellite data.
- Provide high availability and reliability.
- Ensure secure, role-based data access.
- Enable distributed analytics for real-time insights.

Phases of the Pipeline:

4.1 Data Ingestion Layer

- Captures raw satellite feeds in real-time.
- Sources: satellites, ground stations, IoT devices.

Tools:

• **Apache Flink (as consumer/processor)** — pairs with Kafka for low-latency stream processing (if you need stateful streaming).

• **Google Cloud Pub/Sub** — fully-managed global messaging for GCP environments (native BigQuery/Dataflow integration).

• **AWS Kinesis / Kinesis Video Streams** — AWS managed ingestion for time series / media; integrates with S3 + Lambda + SageMaker for downstream ML. Good if you run on AWS.

Pseudo code for ingesting satellite data

```
public class DataIngestion {
  public static void main(String[] args) {
    SatelliteFeed feed = new SatelliteFeed("Sentinel-2");

    while (feed.hasNextFrame()) {
      Frame data = feed.getNextFrame();

      // Push into Kafka (or any streaming queue)
      KafkaProducer producer = new KafkaProducer();
      producer.send("satellite-topic", data.toBytes());

      System.out.println("Ingested frame: " + data.getId());
    }
  }
}
```

4.2 Scalable Storage System

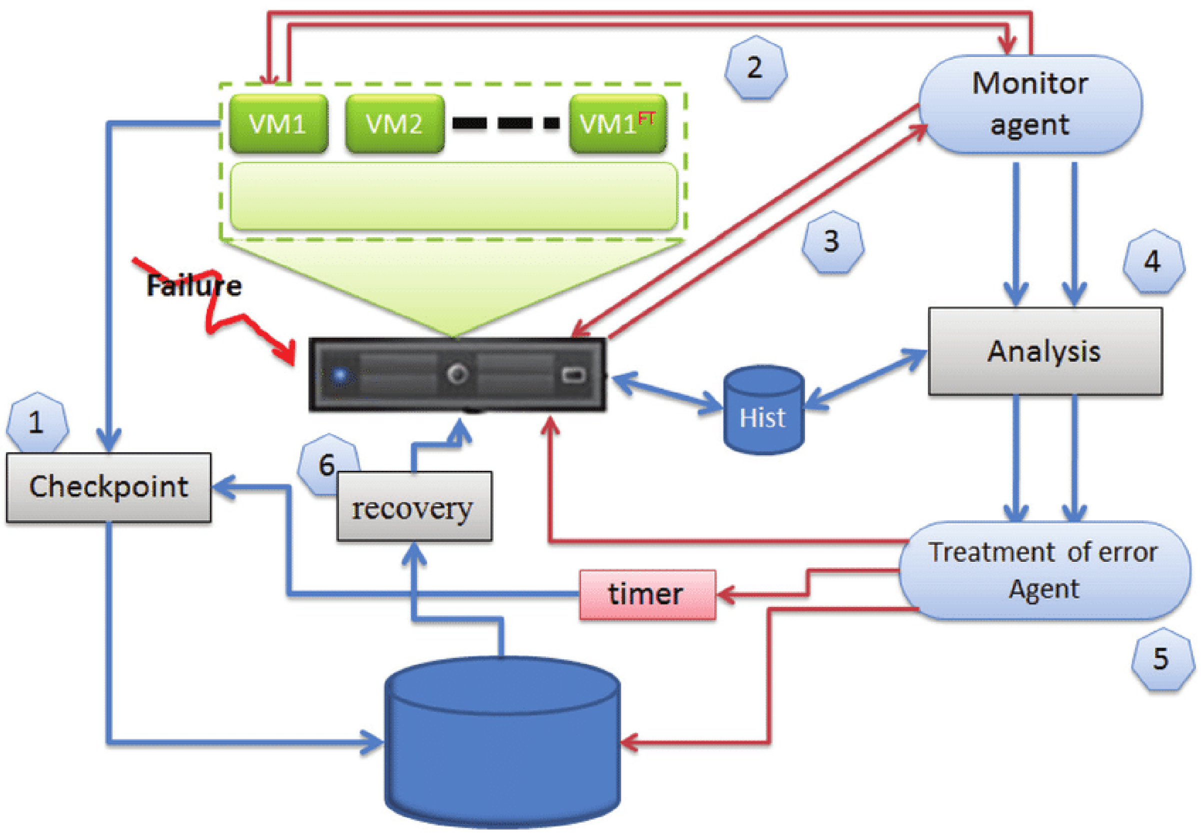
- Stores TB–PB scale EO (Earth Observation) data.
- Technologies: HDFS, Ceph, Amazon S3, CREODIAS object storage.
- Metadata indexing for efficient search.
- Hot vs. Cold storage tiers.
- Tools: **HopsFS (Hopsworks)** — HDFS-compatible next-gen FS with erasure coding, NVMe caching and tiered storage for large ML workloads (used in ExtremeEarth / Hopsworks). Good for feature stores and ML workloads

4.3 Processing & Analytics Framework

- Distributed/parallel computing (Spark, Flink, Hopsworks).
- Operations:
    - Noise filtering
    - Image preprocessing
    - Data transformation
    - Machine learning (object detection, classification)

4.4 Fault-Tolerant Architecture

- **Replication** of data across nodes.
- **Checkpointing** in ML/DL pipelines.
- **Automated recovery** in case of failure.
- **Tools: Kubernetes + StatefulSets + readiness/liveness probes** — automated pod restarts and controllers enable recovery; combine with multi-AZ clusters for resilience.



4.5 High Availability Design

- **Ground-station / Telemetry Ingest**
    - Local edge collectors at each ground station receive frames/telemetry and push into a buffer/edge queue (MQTT/Kafka/AMQP). Keep a small local persisted buffer for link outages.
- **Message Bus / Ingestion Layer (durable, partitioned)**
    - Use a partitioned, durable streaming system (Apache Kafka or equivalent); run regional clusters and replicate topics for geo-resilience. This decouples producers (ground stations) from consumers and enables replay.
- **Processing Tier**
    - Real-time stream processors for low-latency products (Flink/Beam/Kafka Streams) + worker fleets for batch/nearline reprocessing. Deploy these on HA Kubernetes clusters with multiple control plane nodes and distributed etcd.
- **Long-term Storage / Archive**
    - Object storage (on-prem S3-compatible or cloud S3) for raw/level-0..N products. Maintain multi-region replication or periodic cross-region backups for disaster recovery and sovereignty constraints. Use immutable objects + versioning.
- **Metadata, Catalog & Index**
    - Centralized metadata/catalog (searchable) replicated across regions. Store product indices and provenance for reprocessing and traceability (important for science missions).
- **Delivery / APIs**
    - Frontend API layer (stateless, autoscaled) behind global load balancers and CDN. Serve files via signed/time-limited URLs or CDN signed URLs for secure, performant downloads.
- **Operations / Monitoring / Security**
    - Centralized logging, SIEM/alerting, health checks, SLOs/SLAs, and automated failover playbooks. Enforce RBAC, TLS, key management (KMS), and audit logs.

4.6 Secure Access & Retrieval

Works through:

1. **Authentication** – verify user/app identity (passwords, MFA, API keys).
2. **Authorization** – check permissions (RBAC/ABAC, who can read/write).
3. **Encryption** – protect data at rest (AES) and in transit (TLS/SSL).
4. **Controlled Retrieval** – secure APIs, signed URLs, VPN/private network.
5. **Auditing** – log and monitor all access for security and compliance.

References

1. ExtremeEarth Project: *ExtremeEarth: From Copernicus Big Data to Extreme Earth Analytics* (ESA, 2020).
2. ESA Copernicus Programme – https://www.copernicus.eu
3. NASA EarthData – https://earthdata.nasa.gov
4. Google Earth Engine – https://earthengine.google.com