**CLOUD DEPLO3YMENT**

A satellite collecting real-time data

A backend (Flask/Python) that stores/serves this data (via API)

A frontend (website/dashboard) that displays it to users

A cloud platform where everything is hosted (like AWS, GCP, or Azure)

Multiple user laptops accessing different dashboards from the same source

Cloud deployment is the process of hosting your applications,services, and data on cloud infrastructure so they are accessible anytime, anywhere over the internet.

Instead of running software on a personal computer or local server, we deploy it on cloud platforms like AWS, Google Cloud, or Azure, which manage the servers, networking, and scalability.

**ARCHITECTURE OVERVIEW**:

Satellite -> Ground Station -> Backend ->Cloud -> Frontend User System

**From backend to cloud:**

backend sends data to the cloud via these APIs, and stores it in cloud databases.

These APIs are hosted in the cloud (e.g., AWS, GCP)

Your backend stores the data in cloud-hosted databases like:

AWS RDS / DynamoDB,GCP Cloud SQL / Firestore

For real-time streaming telemetry, you can use:

AWS KinesisGoogle Pub/Sub.

These tools help push high-frequency satellite data into the cloud for processing and analytics.

**From cloud to Website:**

Website is hosted on the cloud (e.g., AWS S3, GCP Firebase Hosting).

Frontend fetches data from cloud-based backend APIs.

Real-time data is displayed using cloud resources.

The cloud platforms:

* AWS EC2 / Elastic Beanstalk
* Google Cloud Run / App Engine
* Microsoft Azure App Services
* Heroku (simpler PaaS)
* Render / Railway / Vercel (for full-stack)

Google Cloud Platform

It updated the Real time datas.

GCP handles real-time telemetry, edge data processing, and ML model deployment very well.

Amazon Web Services:

It heavy-duty real-time streaming ,industry-level deployment

Serverless backend

Microsoft Azure:

Strong with identity management

Good for .NET/Windows users

**Steps to deploy backend to cloud and cloud to webdashboard:**

* + Google Cloud (Pub/Sub + Cloud Run/App Engine), AWS (Kinesis + Lambda/ECS), or Azure (Event Hubs + Stream Analytics).
  + A managed streaming service to ingest and process real-time data (e.g., Google Pub/Sub, AWS Kinesis, Azure Event Hubs)
  + Configure your neuromorphic satellite system or backend to publish data events continuously to the streaming service.
  + Processed results can be stored in real-time databases (Cloud Bigtable, DynamoDB) or cached storage for quick retrieval.
  + Alternatively, forward data directly to your backend APIs that feed your website
  + +.
  + Update your website frontend code to call these APIs periodically or use WebSocket connections to receive push updates.
  + Use JavaScript fetch for polling or libraries like Socket.io for push-based live data.
  + Verify data flows from source → streaming service → processing → backend API → frontend dashboard in near real time.
  + Monitor latency, throughput, and errors with cloud monitoring tools.

**Challenges**

Not allowing HTTP/HTTPS or database ports

Using localhost instead of the actual cloud URL in the frontend.

**Fix:** Use the deployed API URL (e.g., <https://yourapi.cloud.com>).

Browser blocks the frontend from accessing the backend if CORS is not set.

**Fix:** Add CORS settings in your backend (e.g., Flask-CORS or Express-CORS).

App doesn’t run on the right port in the cloud environment.

**Fix:** Use environment variables like PORT in your backend.

Backend can't reach your cloud database.

**Fix:** Check your database credentials and whitelist IPs if needed.

Hard to find what's wrong when the app breaks.

**Fix:** Enable cloud logging tools (e.g., GCP Logs, AWS CloudWatch).