

Proyecto Pascal:

Mi proyecto se va a basaren la optimización de rutas interplanetarias y gestión de transporte espacial:

El proyecto se centra en la optimización de rutas y la gestión de transporte interplanetario de personas y mercancías, similar al funcionamiento actual de las aerolíneas pero a escala espacial. El objetivo es analizar rutas más eficientes, minimizar tiempos de recorrido y maximizar la utilización de recursos espaciales (combustible, capacidad de carga, tiempo de viaje, etc.).

Herramientas y tecnologías seleccionadas:

Apache Kafka: Para la ingesta y transmisión de flujos de datos en tiempo real provenientes de sensores de naves, centros de control planetarios y estaciones espaciales.

HDFS (Hadoop Distributed File System): Para el almacenamiento a gran escala de datos estructurados (horarios, rutas, cargas) y no estructurados (registros de sensores, logs de sistemas de navegación).

Data Lake: Para organizar y centralizar datos estructurados, semiestructurados y no estructurados, permitiendo flexibilidad en la exploración y análisis.

Apache Hive: Para realizar consultas SQL sobre grandes volúmenes de datos almacenados en HDFS.

Apache Spark: Para el procesamiento de datos tanto en batch como en streaming.

YARN: Como sistema de gestión de recursos dentro del ecosistema Hadoop.

Arquitectura Lambda: Para combinar el análisis de datos históricos y en tiempo real, permitiendo una visión completa de las operaciones espaciales.

Diseño de arquitectura distribuida:

Se propone una arquitectura Lambda que combina procesamiento batch (para datos históricos y análisis profundo) y streaming (para monitorización y respuestas en tiempo real). La arquitectura está compuesta por las siguientes capas:

Capa de ingesta de datos:

Apache Kafka recoge datos en tiempo real desde naves espaciales, sensores y centros de monitoreo.

Capa de almacenamiento:

HDFS almacena datos históricos en crudo.

El Data Lake organiza datos de diferentes fuentes: rutas, pasajeros, tiempos, consumo energético, etc.

Capa de procesamiento:

Spark Streaming procesa los datos en tiempo real desde Kafka.

Spark Batch y Hive procesan datos históricos para análisis detallado de patrones de rutas, congestión interplanetaria, etc.

Capa de consulta y análisis:

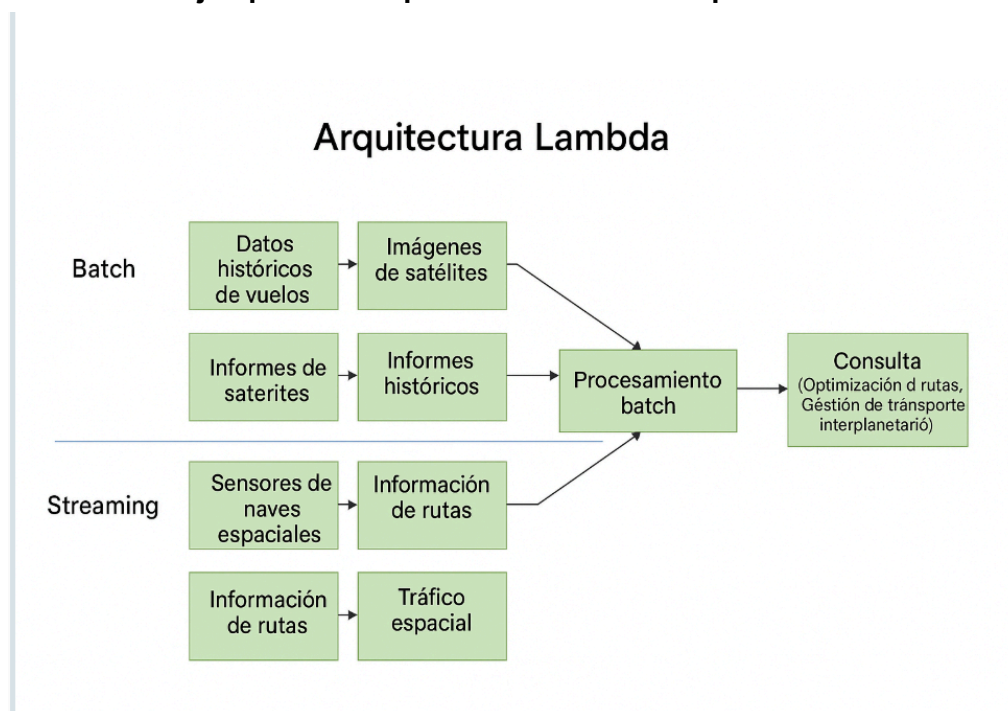
Hive permite ejecutar consultas SQL para extraer información útil.

Spark MLlib (opcional) puede aplicarse para algoritmos de predicción de demanda de rutas o fallos.

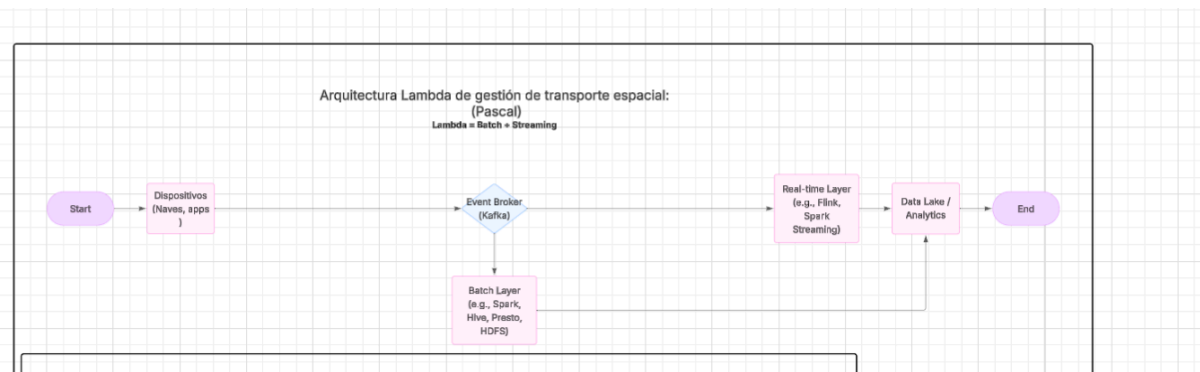
Capa de visualización

Herramientas como Power BI, Tableau o Matplotlib permiten crear dashboards que muestran métricas clave: rutas óptimas, congestión, tiempos de viaje, etc.

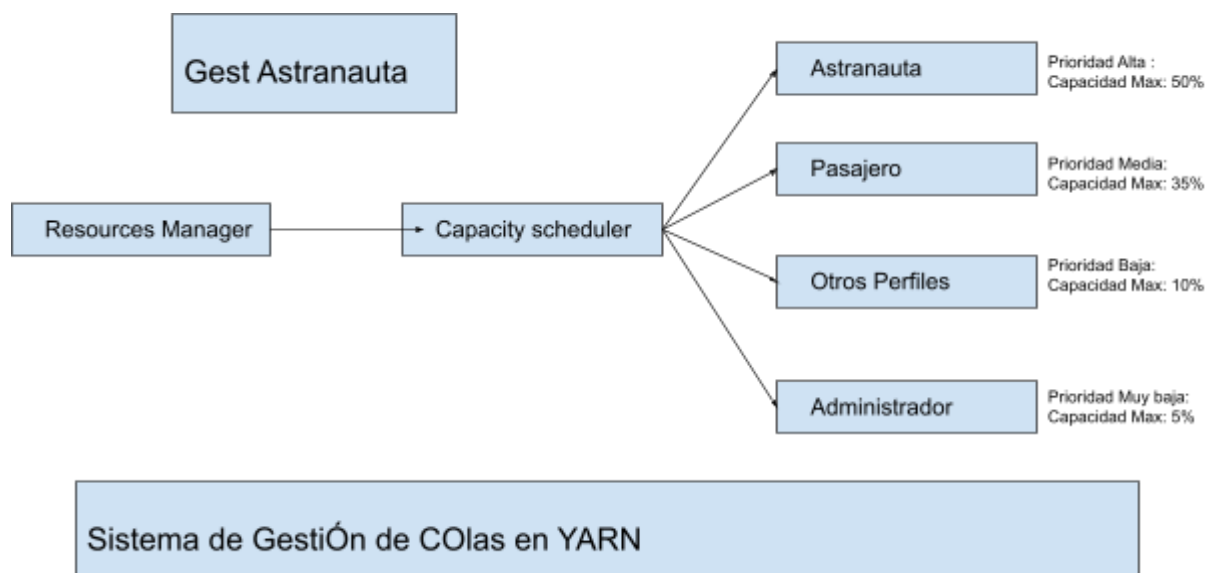
Así sería un Ejemplo de la representación de la arquitectura Lambda del sistema



Un ejemplo de la distribución de la arquitectura Lambda seria losiguiente



Y asi como seria la distribución del Sistema de Gestión de colas en YARN



Piloto de nave prioridad Alta: :

Prioridad Alta 50%, Es el perfil más crítico. La operación del sistema de transporte interplanetario depende directamente de ellos. Necesitan procesamiento en tiempo real para recibir datos de navegación, estado del entorno espacial, alertas de colisión y cambios de ruta. Sin este procesamiento prioritario, la seguridad de los viajes podría verse comprometida.

Pasajero prioridad Media::

Prioridad Media , Este perfil también requiere respuestas rápidas, como confirmaciones de abordaje, estado de viaje, y notificaciones de entrega, pero

puede tolerar pequeñas latencias. Aunque no es tan crítico como el piloto, sigue siendo clave para una experiencia fluida.

Otros perfiles , prioridad Baja:

Incluyen procesamiento de datos históricos, validaciones en segundo plano, o mantenimiento del sistema. No tienen impacto inmediato en la operación del transporte, por lo que pueden esperar sin comprometer el sistema.

Administrador del sistema, prioridad Muy Baja: :

Corresponde al equipo que realiza configuraciones, auditorías o tareas administrativas. Sus acciones no afectan la operación en tiempo real, por lo que se les asigna una capacidad mínima.

Sistema de Apache Hive:

Un Ejemplo del Dataset: Transporte Espacial Interplanetario

Columnas:

id_transaccion: ID único del viaje o evento

timestamp: Fecha y hora del evento

tipo_usuario: Piloto / Pasajero / Carga / Sensor / Administrador

origen: Planeta o estación de partida

destino: Planeta o estación de destino

estado: En curso / Finalizado / Cancelado

latencia_ms: Tiempo de respuesta registrado

prioridad: Alta / Media / Baja

tipo_evento: Navegación / Monitoreo / Comunicación / Administración / Entrega

Ejemplo de contenido:

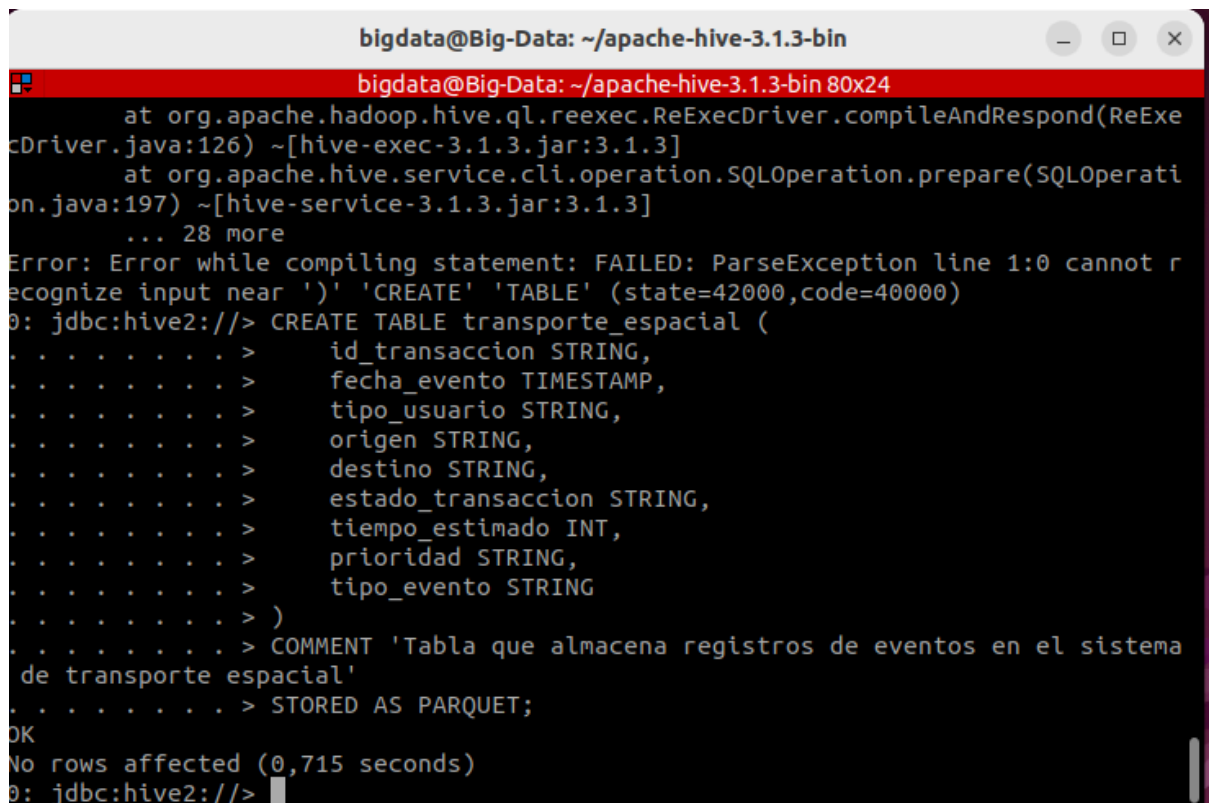
id_transaccion,timestamp,tipo_usuario,origen,destino,estado,latencia_ms,prioridad,tipo_evento
T001,2125-03-21T10:15:00Z,Piloto,Marte,Tierra,En curso,30,Alta,Navegación

T002,2125-03-21T10:15:05Z,Pasajero,Marte,Tierra,En curso,120,Media,Comunicación
T003,2125-03-21T10:15:10Z,Sensor,Órbita Saturno,Base Alfa,Finalizado,540,Baja,Monitoreo
T004,2125-03-21T10:15:15Z,Administrador,Centro Control,Base Luna,En
curso,800,Baja,Administración
T005,2125-03-21T10:15:20Z,Carga,Estación Titán,Marte,Cancelado,150,Media,Entrega
T006,2125-03-21T10:15:25Z,Piloto,Luna,Marte,En curso,25,Alta,Navegación
T007,2125-03-21T10:15:30Z,Sensor,Marte,Estación Europa,Finalizado,620,Baja,Monitoreo

A continuación vamos a cargar el Daset en un entorno real con un sistema de Apache Hive :

En primer lugar creamos la tabla :

```
CREATE TABLE transporte_espacial (
  id_transaccion STRING,
  fecha_evento TIMESTAMP,
  tipo_usuario STRING,
  origen STRING,
  destino STRING,
  estado_transaccion STRING,
  tiempo_estimado INT,
  prioridad STRING,
  tipo_evento STRING
)
COMMENT 'Tabla que almacena registros de eventos en el sistema de transporte espacial'
STORED AS PARQUET;
```



```
bigdata@Big-Data: ~/apache-hive-3.1.3-bin
bigdata@Big-Data: ~/apache-hive-3.1.3-bin 80x24
at org.apache.hadoop.hive.ql.reexec.ReExecDriver.compileAndRespond(ReExecDriver.java:126) ~[hive-exec-3.1.3.jar:3.1.3]
at org.apache.hive.service.cli.operation.SQLOperation.prepare(SQLOperation.java:197) ~[hive-service-3.1.3.jar:3.1.3]
... 28 more
Error: Error while compiling statement: FAILED: ParseException line 1:0 cannot recognize input near ')' 'CREATE' 'TABLE' (state=42000,code=40000)
0: jdbc:hive2://> CREATE TABLE transporte_espacial (
. . . . . > id_transaccion STRING,
. . . . . > fecha_evento TIMESTAMP,
. . . . . > tipo_usuario STRING,
. . . . . > origen STRING,
. . . . . > destino STRING,
. . . . . > estado_transaccion STRING,
. . . . . > tiempo_estimado INT,
. . . . . > prioridad STRING,
. . . . . > tipo_evento STRING
. . . . . > )
. . . . . > COMMENT 'Tabla que almacena registros de eventos en el sistema de transporte espacial'
. . . . . > STORED AS PARQUET;
OK
No rows affected (0,715 seconds)
0: jdbc:hive2://>
```

Después insertamos los datos de pruebas:

```
INSERT INTO transporte_especial VALUES
('TX001', '2025-05-07 10:23:00', 'Conductor', 'Marte', 'Luna', 'en_proceso', 120, 'Alta',
'Solicitud'),
('TX002', '2025-05-07 10:25:15', 'Pasajero', 'Tierra', 'Estación Orbital A', 'confirmado', 200,
'Media', 'Asignación'),
('TX003', '2025-05-07 10:27:42', 'Administrador', 'Centro Control Tierra', 'Estación Beta',
'procesado', 350, 'Baja', 'Monitoreo'),
('TX004', '2025-05-07 10:30:11', 'Tester', 'Simulador A', 'Simulador B', 'completado', 80,
'Baja', 'Test'),
('TX005', '2025-05-07 10:32:00', 'Conductor', 'Luna', 'Marte', 'en_proceso', 95, 'Alta', 'Ruta
actualizada');
```

