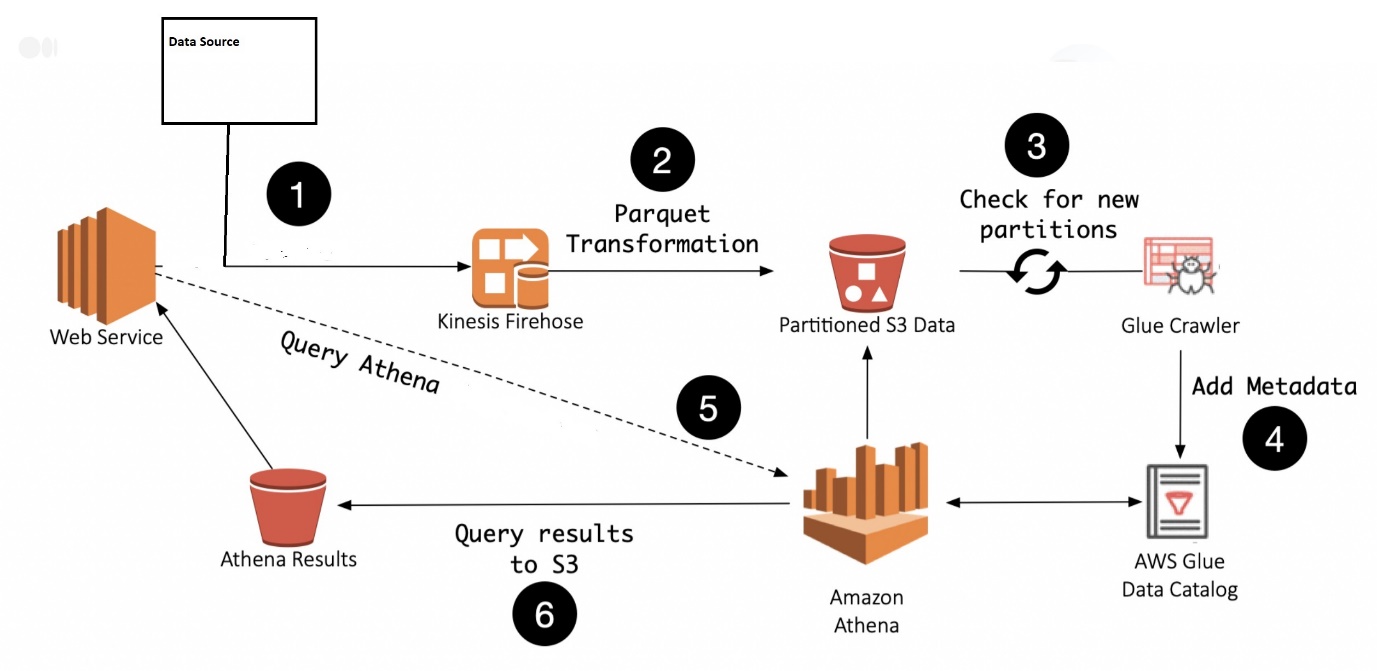
**Case: Data ingestion pipeline**

Imagine that for providing data to fuel this service, you need to receive and insert big batches of new prices, ranging within tens of thousands of items, conforming to a similar format. Each batch of items needs to be processed together, either all items go in, or none of them do.

Both the incoming data updates and requests for data can be highly sporadic - there might be large periods without much activity, followed by periods of heavy activity.

High availability is a strict requirement from the customers.

**a) How would you design the system?**



AWS Services: AWS Kinesis Firehose, S3, Glue Crawler, Glue Catalog, Athena

1) AWS Kinesis Firehose receives incoming data updates from data sources. Kinesis Firehose is a powerful and highly scalable data streaming service. It handles loading streaming data and pushing into data stores and analytic tools. It can stream large amounts of data and in batches to s3 as partitioned data. Moreover, it supports data replication across three AZ's in an AWS region, provides high availability and durability.

2) Kinesis Firehose is a fully managed service and we have an event stream with a minimal setup that outputs data to S3 in a predictable location. It supports many data formats, conversions, transformations, compressions. To keep the data file size small and compact, we can leverage Firehose’s ability to transform data from the stream to the Apache Parquet format and write into S3.

3) Glue Crawler should be configured to crawl the data in s3. The crawler traverses an S3 location, updates table schema to discover new columns as well as partitions in your data and updates the Glue Catalog. With the ability to schedule a crawler to run every few minutes, the service can have data ingested from Firehose, sent to S3, and discovered by the crawler made available as a single query able event stream.

4) Amazon Athena is a serverless service to query against S3 objects using standard SQL language. Xeneta API Service queries Athena, which gets data from Glue data catalog and return as a response. It also stores query results in s3 bucket.

**b) How would you set up monitoring to identify bottlenecks as the load grows?**

**c) How can those bottlenecks be addressed in the future?**

Storing data streams in an S3 bucket can be a bottleneck if data uploads are very large. This can be improved by using S3 multipart uploads, S3 Transfer Acceleration. We can also use S3 prefixes to improve requests per second.

As the load grows, there may be few chances of failure in the attempt to write data streams into s3 and this can be addressed. After a retry and timeout period, Kinesis firehose loads failed events into the S3 bucket. We can set up a lambda which will be triggered based on Put notification from S3, reads the failed events from the object, decodes the content, and writes the events back into S3 in the original format.

Also, we can set up monitoring on firehose using bottlenecks metrics which will report the metrics to cloud watch. Alarms can be created to monitor the streams from cloudwatch

**Additional questions**

1) The batch updates have started to become very large, but the requirements for their processing time are strict.

Kinesis Firehose is a powerful and highly scalable data streaming service and batches out data in windows of either data size or a time limit. It will take care of all of the monitoring, scaling, and data management

2) Code updates need to be pushed out frequently. This needs to be done without the risk of stopping a data update already being processed, nor a data response from being lost.

This can be achieved using CI / CD. We can set up a CI/CD pipeline that takes updated code, convert it into a docker file, build the new version of the docker image and push it to Amazon ECR. AWS EKS Deployments will update an application running in pods using rolling updates mechanism with no downtime.

3) For development and staging purposes, you need to start up a number of scaled-down versions of the system.

With Terraform, we can spin infrastructure with different parameters in different environments. This makes it possible to create scaled-down versions of the system for development or staging.

Additionally, if we want to have different environments in a single eks cluster, we can set up namespaces with different configurations.