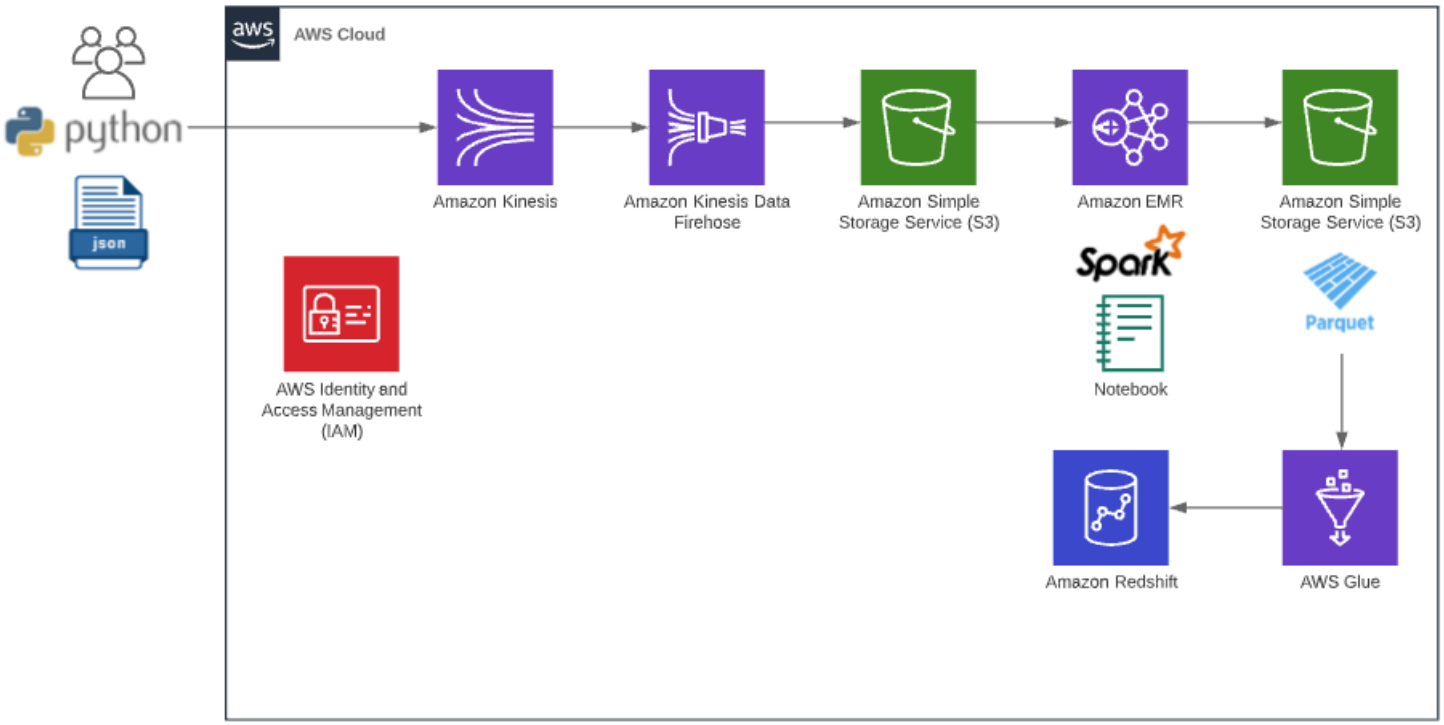
I spent a few days to familiarize myself a bit with AWS. I have been doing a lot of reading lately about it, but had only created a couple AWS VMs. I'm much more familiar with Azure, having worked with it for the past year or so as part of the UW Big Data Technology certificate program.

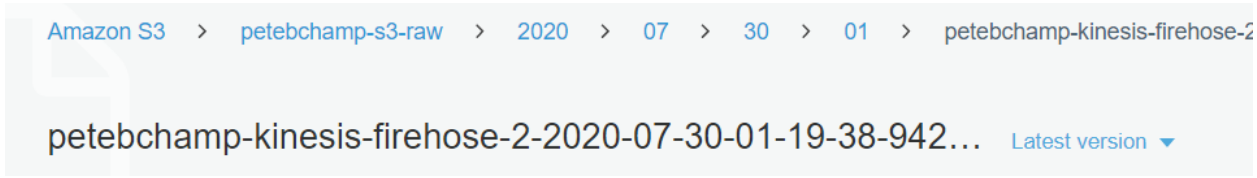
Here's the overall architecture I came up with, which I'll describe in more detail below:

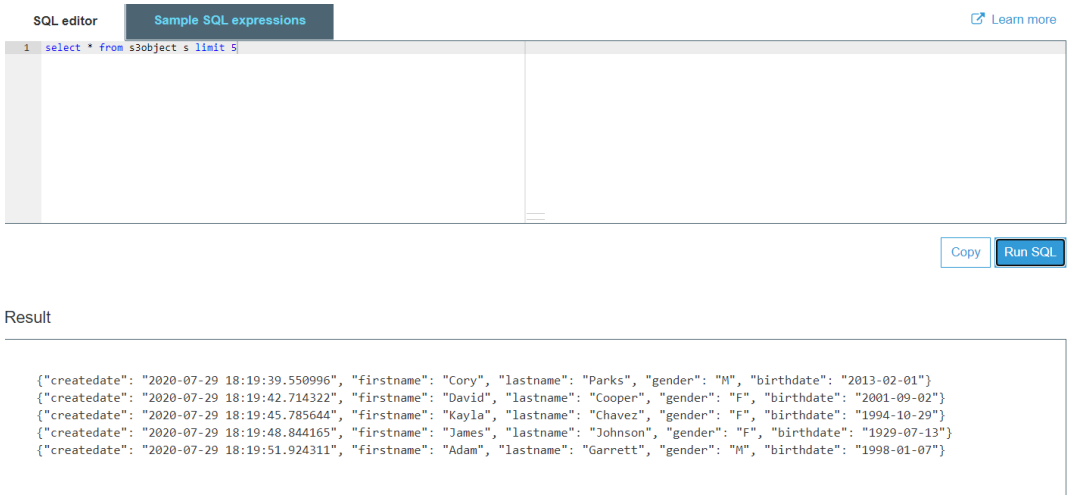


I was especially interested in working with some streaming data. As with most cloud platforms there are multiple ways in AWS to ingest data streams, including Kinesis, Amazon Managed Streaming for Apache Kafka (Amazon MSK), or Kafka on an AWS VM (for this POC). I opted for Kinesis just to get experience with it. Perhaps I'll try MSK next, as I do have some experience using Kafka.

I first set up a Kinesis Data stream. Then I wrote small Python scripts to act as producer and consumer for Kinesis, generating some random data (see the notebooks). I was able to see the streaming data in the consumer script.

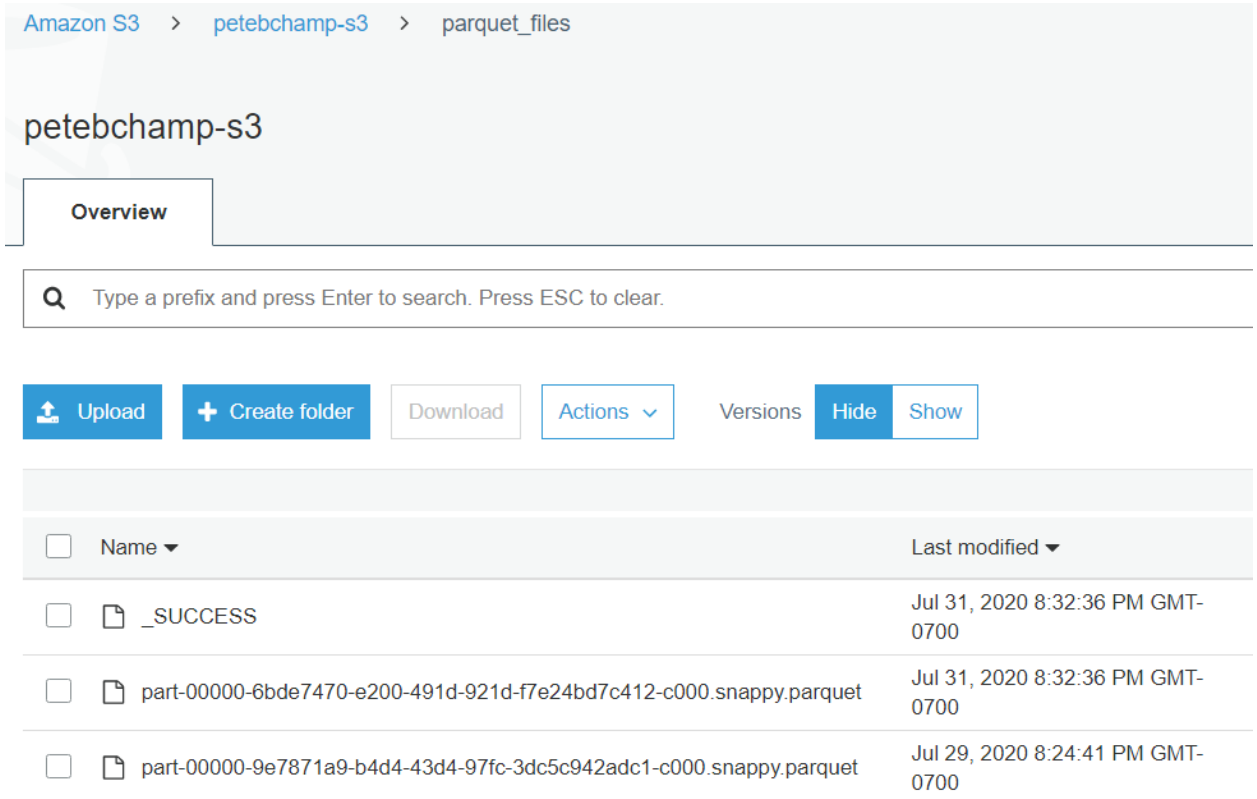
I then set up a Kinesis Firehose delivery stream to move the data from the data stream to an S3 bucket, where I was able to see the records:





I wanted to do some manipulation of the data, and looked into using AWS Lambda, but opted at this time to use Spark in AWS EMR. I created two Jupyter Spark notebooks, both of which check for and eliminate duplicate records and then write the data back to S3 as parquet files. The first one just deals with a static/batch set of data, as I was working through the Spark code. I then modified that one to read and write streaming data. Both notebooks are included here. I created several iterations of EMR clusters just to go through all the options, and needed to include Hive to be able to use Spark SQL code, after having that error the first time.

Here are the parquet files in S3:



I wanted to write the data directly from Spark to Redshift, and looked into installing the Redshift JDBC, but it seemed very involved, with installations of the driver and the correct version of Java components. Instead, I just decided to use AWS Glue to copy the data to a Redshift table I created.

```
create table person (
        create_date TIMESTAMP ,
        first_name varchar(20),
        last_name varchar(50),
        gender char(1),
        birth_date date
        );
```

I did have to set up a new IAM role and associate it to the Glue for the job to complete successfully. Also, I first created the table schema in Glue manually, and then tried having a Crawler discover the schema based on the parquet files, and it came up with the same schema I had created.

If I were doing anything more elaborate with Redshift I would definitely want to use a client tool like SQL Workbench or pgAdmin, although it is convenient to be able to query right from the console.