David Dunlap

SFL Scientific – Written Exercise

1. What is a Data Lake? Explain its benefits, how it differs from a data warehouse, and how it might benefit a client.

The terms “Data Lake” and “Data Warehouse” give us clues to help demonstrate what they are and what they can be used for. A warehouse implies inherent structure. Everything in the warehouse has its proper place and is organized. On the other hand, a lake contains many different types of organisms and can have many different streams feeding into it. A data warehouse is a storage solution where all the data is structured and has already been processed. A data lake is a storage repository that can contain many different types of data, both structured and unstructured. It holds large amounts of raw data in its original format. This means that it is very quick and easy to update, add to, and change the data within the data lake. It is a much more flexible solution than a data warehouse where the data must be extracted, cleaned, and transformed before being loaded. Data lakes are most beneficial for performing deep analysis on data. Since there is a wide variety of types of data contained in the data lake, patterns can be found across and within many unique pieces of data.

2. Explain serverless architecture. What are its pros and cons?

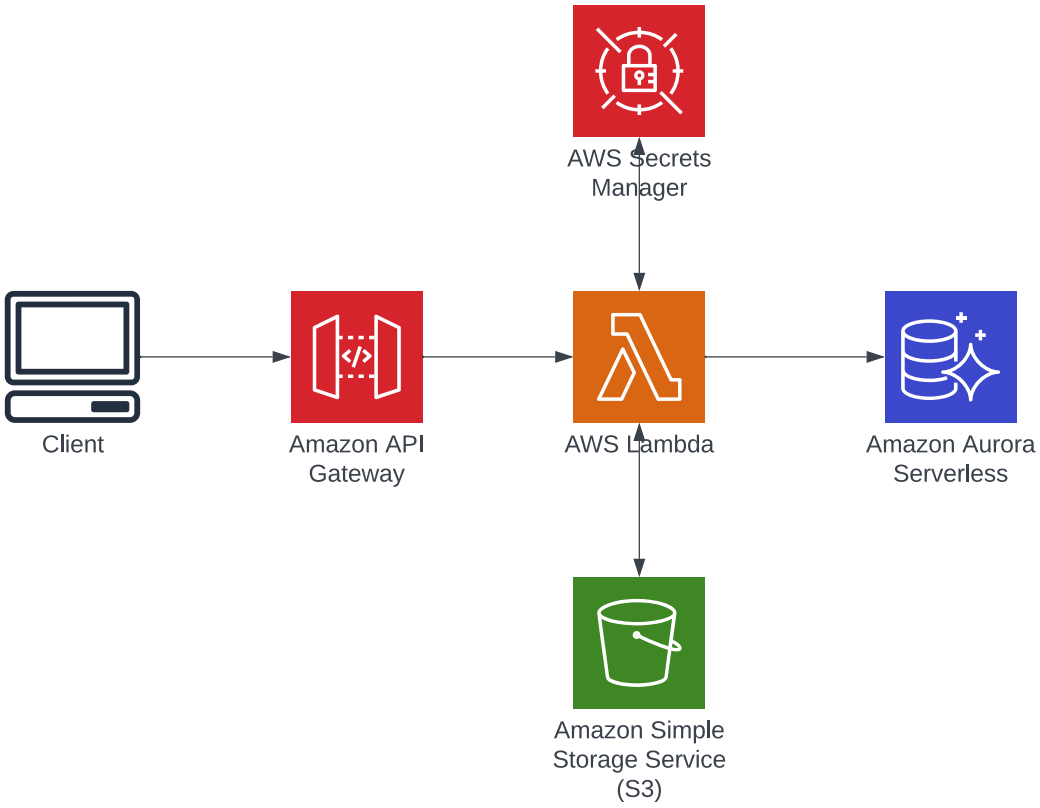
Serverless architecture is when a codebase is not hosted on a singular monolith server, but instead broken up into smaller chunks. Each piece of the program is separated into its own functional piece that can be independently scaled depending on demand. While it is called “serverless” it is still hosted on a server. The difference is that this server is handled by some vendor (such as AWS) instead of the developers. A serverless architecture works best for applications that are lightweight and flexible. It is less ideal for applications that are large and have a consistent load.

There are many pros to using a serverless architecture. One of the main benefits is that you only pay for what you use. Charges come from the time that the code is running, so for applications where processing can happen quickly, a serverless architecture can drastically reduce costs. It is also inherently scalable. If the load on the application increases, more processes can quickly be spun up. Conversely, if the load decreases, only the processes that are needed are running. Finally, since the servers are managed by the vendors instead of the development team, there is a reduced need for DevOps which further helps to reduce costs.

On the other hand, there are a few cons to using serverless architectures. Just as it can be great for applications that run quickly and have inconsistent load, it is not optimized for applications that take a long time to process and have consistent load. Because you pay for the time the code takes to run, serverless can be much less cost effective if processing isn’t happening quickly. Serverless may also affect the performance since processes need to be spun up when load increases. This is especially noticeable for functions that are called rarely and therefore need to be freshly started up nearly every time they are called. Finally, serverless may make it difficult to

switch vendors because all the processes are spread out across many of the vendor's services. For example, switching away from AWS serverless could mean needing to find alternatives to AWS Lambda, API Gateway, S3, Secrets Manager, SQS, and more.

3. Please provide a diagram of the ETL pipeline from Section 1 using serverless AWS services. Describe each component and its function within the pipeline.



The above diagram shows how the ETL pipeline from Section 1 could be created using serverless AWS services.

The first component is the API Gateway. This is the "front door" to this application and allows a client to communicate with the service through HTTP calls. Depending on the exact use cases required, this could facilitate kicking off the ETL process, sending new data to the pipeline, and/or querying data that has already been processed. API Gateway would allow for the processing of thousands of concurrent calls at once. The pipeline from Section 1 does not have an API at the front, but if moved to AWS it could allow for a much easier way to interface with the system.

The next component is AWS Lambda. This is where the meat of the processing would occur. The main code for the extraction, transformation, and loading would be hosted here. Lambda allows for running code without having to maintain a server to host it. If it is decided that the API Gateway is not needed, Lambda could also kick off the ETL process as a scheduled event so it can be triggered automatically.

The next component is AWS Secrets Manager. For this ETL pipeline, this would act in place of the .env file. It would store confidential information such as the database credentials and the IpInfo API key. The Lambda function would communicate with the Secrets Manager to get this information securely. This allows the sensitive information to be protected.

AWS Lambda also communicates with a Simple Storage Service (S3) bucket. Amazon's S3 is an object store for more permanent files. For this ETL pipeline, that would be the datacenters.csv file that contains information about the AWS region locations, as well as the DATA.csv file that contains the original data to be processed. It could also hold the init.sql that has the database schema.

The final service in the diagram is Aurora Serverless. This service allows for the creation of auto-scaling SQL databases (in this case a PostgreSQL database). Aurora allows for the ability to automatically start up, shut down, and scale the database depending on the application's needs.

4. Describe modern MLOps and how organizations should be approaching management from a tool and system perspective.

MLOps is the application of DevOps to Machine Learning where ML models are tracked through their lifecycle, especially after they are deployed into a real-world environment. MLOps covers everything from monitoring, experimenting with, adjusting, and retraining models.

There are many different branches that fall under MLOps, but there are a few key ones that organizations should make sure they are tracking. The first of these is the lifecycle of the data. This starts with data exploration which can be done with tools such as Jupyter Notebook, then data processing which can be done with tools like Hadoop and Spark, then data visualization and reporting. There is also the lifecycle of the model. This includes everything from using tools to track metadata about the model training process across the whole team, hyperparameter tuning, and continuous model retraining.

One system platform that helps with the entirety of MLOps is Amazon SageMaker. SageMaker has automated tools to detect model bias and model drift. It has tools for automated model creation, training, and tuning. Finally, it has tools for CI/CD, all from an ML-focused perspective. When organizations are approaching management of MLOps, systems like SageMaker make the entire process a lot more streamlined.