Problems of Relying on Data as Basis for Important Decision Making and How Best to Mitigate them Through Proper Data Collection and Analysis Design.

A Research Paper submitted to the Department of Engineering and Society

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Ruohan Ding

Spring 2023

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Advisor

Pedro A. P. Francisco, Department of Engineering and Society

Introduction

Data can be a wonderful tool that provides previously hidden and valuable insight, but it can also be misleading and discriminative. This is an issue that impacts almost everyone in the world. Flaws in data collection, mistakes in the data analysis algorithms, or even misleading visualization of data can misdirect and give wrong information. Companies that rely on this data to make decisions will ultimately make flawed decisions that can have lasting impacts (Redman, 2016). There can never be a 100% guarantee that the data being used isn't flawed and thus that presents the question of how reliable it is to use data for decision making? Furthermore, how can we design our data collection and analysis pipelines to prevent these flaws from appearing in our data?

Companies like Palantir provide data analysis services using Machine Learning and Artificial Intelligence (Palantir, 2023). Customers then purchase these services and give up their data in return for valuable insight generated from the models.

However, relying on third-party companies does present potential problems. For one there are security concerns with and possibly regulations against sharing sensitive data with an outside company. Some companies prefer to develop their own data-processing pipelines, as this allows for complete control over the entire process. Unfortunately, the services and tools necessary to successfully develop these pipelines require significant time and capital investments that most companies cannot afford. Therefore, using existing cloud services designed specifically to build data-processing pipelines is a good midpoint between relying on a third-party company and creating proprietary software.

Any company that can successfully leverage the wealth of data available to them will be able to generate greater revenue, make better informed decisions, and improve their overall efficiency (Gavin et al., 2019). There are many existing cloud architectures for the design of data processing pipelines.

Background and Significance / Motivation

Data is the defining feature of the 21st century. There is an incredible amount of data in the world but most of it is unprocessed and unused. Data processing is a huge problem that almost every industry experiences. Small startups and big corporations alike need a system to properly collect, analyze, and visualize their data to provide them with valuable insight. Therefore many companies use existing cloud services like Amazon Web Services (AWS) that are specifically designed to resolve this problem (Mesbahi, Rahmani, Hosseinzadeh, et al., 2018). There are many existing cloud architectures for the design of data processing pipelines. One popular technique is to combine together many different AWS services. Even some existing companies that specialize in data analysis, such as the aforementioned Palantir, find it easier to use AWS services instead of creating their own.

The prevalence of data also leads to a prevalence of mistakes. Relying on flawed data to make important decisions often leads to disaster. Flawed data consists of inaccurate, biased, misleading, outdated, and incomplete data. Catastrophic mistakes can be made when they are used as the basis for complex algorithms and decision-making. Data is seen and used as a critical necessity in many industries in today's world.

In the medical field, inaccurate or incomplete medical histories, test results, and patient data can lead to a misdiagnosis and prescription of wrong medication. All of this can lead to real and serious harm to the patient. In the finance field, companies that rely on incomplete or inaccurate financial data to make decisions can end up miscalculating risks and making bad business deals (Redman, 2016). Resulting in steep losses for the business. Politicians who rely on bad data can make wrong campaign decisions and cost them an election. Governments that use flawed data could end up implementing policies that harm their citizens more than help. These are just a few examples of the cascading effects that relying on wrong data can cause.

Building out sophisticated pipelines that are made specifically for data processing and that are designed to prevent flawed data is a big step in the right direction. These pipelines can not only be the first line of defense in identifying bad data but also be able to remove or clean the data. Proper pipelines

should be able to flag data that it deems to be false or misleading and filter them out from the rest of the process.

Methodology

This paper will use the Social Construction of Technology (SCOT) theoretical framework to see how data analysis tools are shaped by the values and practices of the engineers that design them. As well as explore how the resulting data from these analytical pipelines can be used to reinforce the existing power relations in the society that developed them. We will first identify existing problems that data has already caused and how they fit in the SCOT theory. Then we will do case studies on current companies and analyze how their tools are shaped by their engineers and the society in which they were built.

We need to identify where our blind spots are before we can find ways to fix them. However, it seems that these blind spots exist in almost every aspect of our data pipelines and in every type of data we collect. For example, Black Americans are about four times as likely to develop kidney failure when compared to White Americans, but the algorithm that controls transplant placement orders often places Black Americans lower on lists than White Americans (Christensen, Manley, Resendez et al., 2021). The flaw in these algorithms can be hard to resolve and they most aptly can be attributed to flawed data. But it could also be a reflection of the oppression that Black Americans face in America's society. These analytical tools were developed in societies that unintentionally or otherwise tend to favor White people. It is interesting to see the bias of the society reflected in its tools. These types of bias can be found everywhere and in all types of industries. Data often gives a comprehensive snapshot of the population, and therefore it is good at representing the average person under average circumstances. That means disabled people, racial minorities, and people in poverty all suffer from biases in these data algorithms, and the effects of these biases can have devastating impacts on these people's lives.

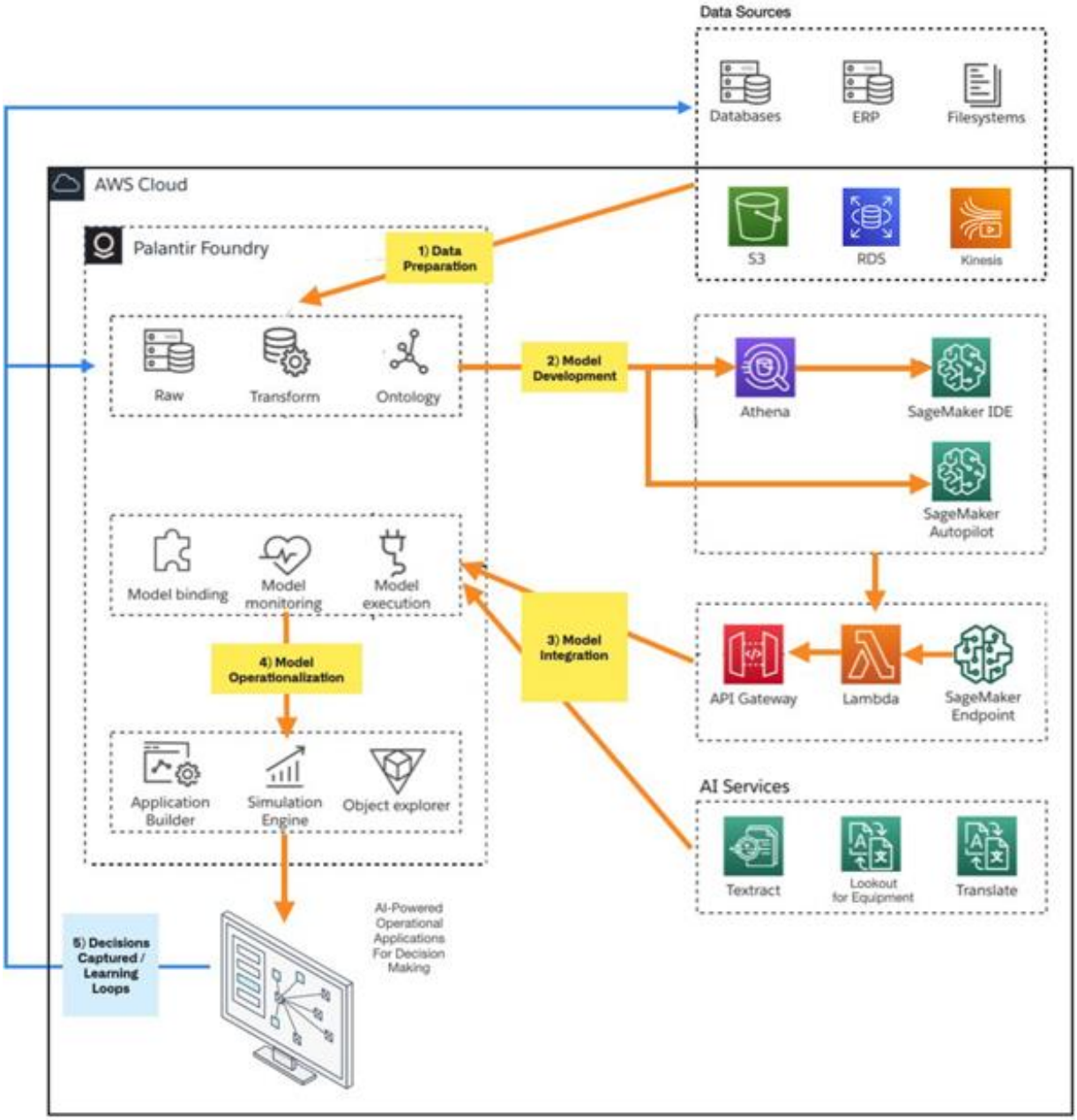
For our first case study let's look at Palantir, a contractor for the United States Department of Defense (DOD). This company is a leader in machine learning data analysis. The DOD employs them to conduct complex military analysis on their behalf, and they rely on this analysis to determine the next best steps for a particular military situation (Palantir, 2023). Sometimes these situations are international conflicts and are extremely sensitive. One wrong move could lead to a drastic escalation and the possibility of an all-out war. Therefore Palantir needs to make sure that the data they are relying on must be 100% accurate in order to mitigate the risk of making any mistakes. The values and goals of the engineers in this particular company are to create pipelines that are extremely unbiased in order to generate the best possible analysis. Therefore Palantir is successfully able to write good data analysis pipelines. They are a great example of SCOT because the context in which the technology was developed in is one that emphasizes correctness and tries to eliminate unintentional or intentional bias. Palantir is able to achieve this in part by cleaning and checking their data through complex algorithms designed specifically to identify flaws, and also by constructing data collection and analysis pipelines that are properly maintained, easily manageable, and intuitive. They accomplished this by developing their own data processing systems in conjunction with existing AWS Cloud services. Their complex Artificial-Intelligence powered analysis is hosted on AWS servers and much of their data deployment and service hosting is also handled by AWS. This shows that these cloud services can be highly flexible. Companies can use as little or as much of these services as fits their needs.

Discussion / Results

The company Palantir has its own models for data analysis and collection but they also leverage AWS to develop and integrate the models and prepare data. The architecture of their Foundry software, a data analytics tool for large businesses, is shown in Figure 1 (Mezzalira, et al, 2022):

Figure 1

Palantir Foundry Architecture



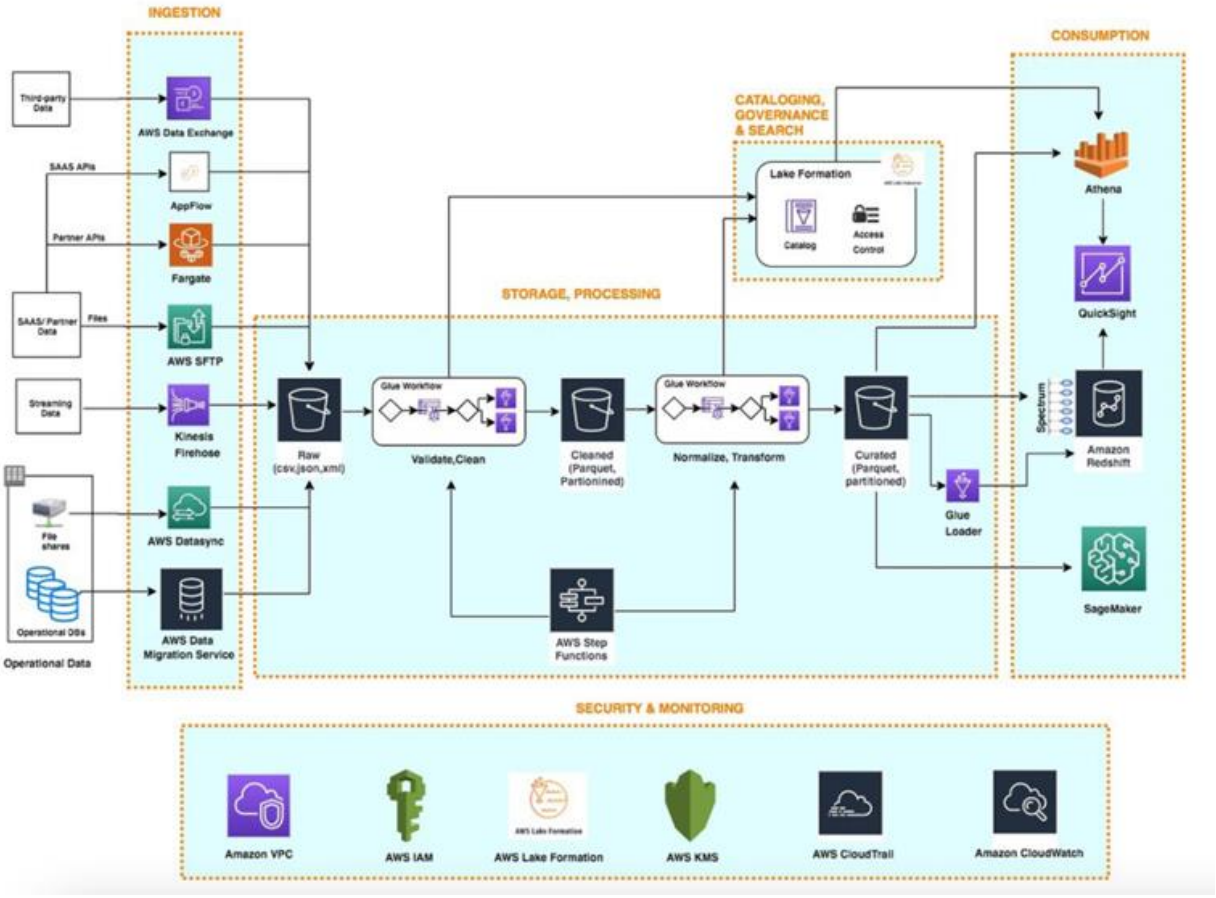
Complex Machine Learning models are applied within Palantir's Foundry platform. The data used by the models are stored and prepared from S3, RDS, and Kinesis which are all hosted on AWS. The

models are deployed through Athena and SageMaker on AWS, which in turn is connected to Lambda and API Gateway. Foundry then hits the API Gateway to access these models and use them to conduct data analysis. That analysis is forwarded to its client-facing front-end. A new company can supply Palantir with their data and see its analysis after that data is processed through these pipelines. Palantir is a great example of when SCOT has a positive effect. It designs its tools with the specific goal of keeping bias out. They instill a value of correctness in their engineers which in turn is reflected heavily in the tools that they create. Companies who recognize that the bias of their engineers and the society in which they are in influences the data analytical pipelines they develop can reach out to Palatnir for its services. Palantir has used SCOT to its advantage and built a proper environment for their engineers to design good, unbiased products. Despite Palantir being a well-established company, it still utilizes AWS to manage the bulk of its data storage and model deployment. This showcases the flexibility of cloud and how it can remain useful from small to large-scale projects.

Many users may not need or want the ultra-sophisticated data analysis that Palantir provides and instead need simple, frictionless analysis done on their data. AWS has many guides for building out an appropriate data analytics pipeline architecture for these use-cases. A suggested possible design built fully with AWS services is shown below in Figure 2 (Kava and Gong, 2020):

Figure 2

Serverless Data Lake Centric Analytics Architecture



This architecture comprehensively covers everything from data ingestion to security and monitoring. Third-party data can be ingested by AWS Data Exchange, Datasync, or Kinesis Firehose and fed into storage. The storage can take the form of relational databases such as Aurora Serverless, block storage such as S3, or NoSQL databases such as DynamoDB. It all depends on the needs of the user.

From here the data can be cleaned and processed through Lambda functions or passed through code hosted on CodeBuild. After the data is properly attained and formatted it can be fed directly to its cataloging or consumption services such as Athena or SageMaker. This entire pipeline is secured and

monitored by AWS's own IAM and VPC services which allow users to restrict access to and limit behaviors of services within the pipeline. This guide demonstrates how flexible AWS cloud services are because there exists a service for every use case. The best part is that since all these services are developed by AWS, they communicate very effectively with one another.

As can be seen from the architecture, data cleaning is inherently built into the design of the data processing pipelines. The collection and analysis portions of the architecture are two distinctly different things. They are connected through the cleaning process. Without first properly cleaning the data, it isn't allowed to move from one portion to the next. This goes a long way in securing the validity of the data. Forcing data to pass tests and checks increases the reliability of the data and increases the confidence we can have in it. Furthermore, since this is a distinct service from the rest of the pipeline we can easily modify and update its functionality later on. If new algorithms and methods of catching flawed data arise they can easily be implemented into this system without the need to take apart the entire thing.

AWS built its services with the goal of creating easy, frictionless tools that other engineers can leverage for their personal use. It is then up to the individual engineers who use these tools to build pipelines. The resulting pipelines are shaped much more heavily by the values and beliefs of these individual engineers than the AWS engineers. Therefore I want to analyze examples of how to build proper pipelines using AWS services with SCOT in mind.

The data processing project that I decided to build can be split up into two main parts. First, a cloud system processes, stores, and analyzes large amounts of data in a reasonable amount of time. The system receives thousands of data logs every day and that data without crashing or taking too long. The data then must be formatted and inserted into a long-term storage database to be used by the next phase of this project. The second phase consists of building out a web-app dashboard that is accessible and easy-to-use. Users should be able to access the dashboard and create an account for themselves. This account must be secure and have multi-factor verification enabled. Users then should be able to add data through this dashboard as well as generate graphs and tables from existing data.

Basic requirements such as the security of the data, scalability of the database with respect to the amount of data, and the ability to handle spikes of users are inherently addressed by the cloud services. AWS services have advanced built-in load balancing, data scaling, and data protection. This makes implementing any new project a lot easier as these simple but highly important aspects of the project are essentially abstracted away.

The architecture I chose to build in order to meet the requirements set out above was inspired by many related works found through online research and by official guides provided by Amazon. Many times tradeoffs had to be made when I chose between two different implementation routes.

The frontend web-app dashboard was mainly written in React. The ChartJS framework was used extensively in order to generate detailed charts and graphs of the data. The dashboard was hosted on AWS Amplify which worked with Cognito in order to provide two-factor authentication to the web-app. Amplify was also configured with IAM roles that give it permissions to communicate with the backend through Lambda functions. The backend consists of the AWS Aurora Serverless database. The database was connected to API Gateway which allowed outside cloud functions to access and parse it. Cloud Lambda functions were written and connected to API Gateway in order to gain access to the database. These cloud functions are called by the frontend in order to insert, delete, modify, and retrieve data from the database.

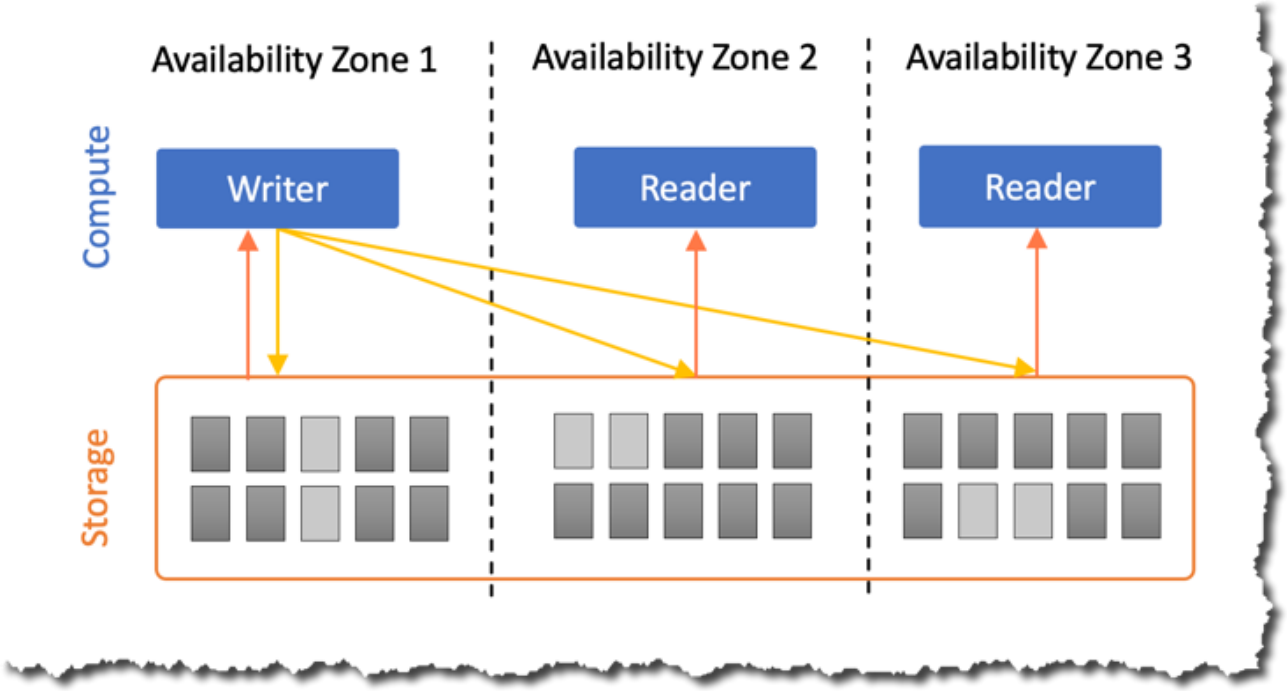
AWS CodeBuild was also set up with a repository in CodeCommit that contained the code for running nightly-tests, collecting the results, and storing the data. A pipeline was then built out in CodePipeline. This pipeline would trigger the CodeBuild functionality which in turn ran the code contained in the CodeCommit repository in its own virtual environment. CodeBuild and CodePipeline both were connected to Aurora Severless's Data API which allowed them to directly conduct Create, Read, Update and, Delete (CRUD) operations on the data without the need to use Lambda and API Gateway. After CodeCommit is finished, the pipeline would then use EventBridge and SES to send out

email notifications of success or failure to stakeholders. The pipeline is triggered at a set time every night by a Jenkins job.

Choosing the right database was possibly the most impactful part of this project. AWS offers many different types of databases ranging from bucket based storage like S3 to NoSQL databases like DynamoDB. The implementation that I chose to go with was AWS Aurora Serverless V2, which is a traditional SQL database. The reason I chose to go with this route was that the type of data I was storing fit best with the traditional SQL storage schemas and its CRUD operations. If I had additional data in the form of image or audio files then I might have gone with a bucket-based storage implementation. There is also a lot less maintenance to worry about when using Aurora Serverless V2. AWS takes care of startup, shutdown, and capacity scaling (Villalba, 2022). Aurora Serverless V2 also has a built in Data API that allows other AWS services with the same IAM roles to easily access its data without the need to write outside APIs. Aurora also decouples computing and storage which allows it to replicate data seamlessly and increase the availability of the service, this functionality is depicted in Figure 3.

Figure 3

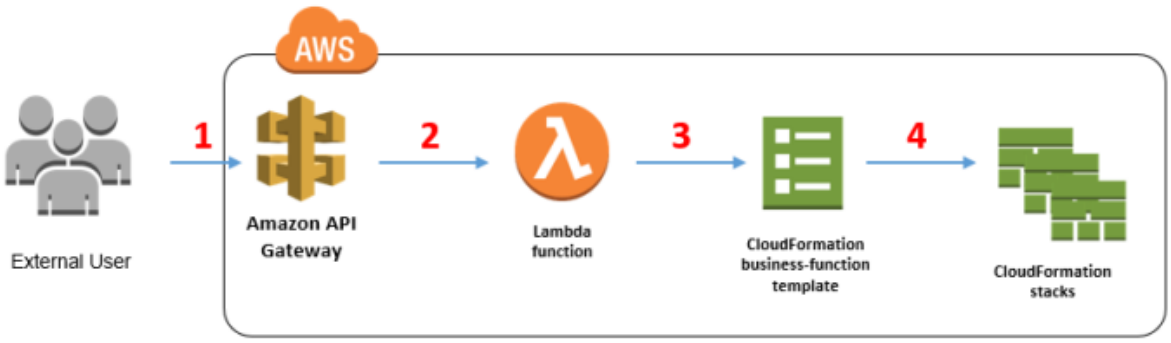
Aurora Separation of Storage and Computation



An API access point and outside cloud functions needed to be written in order for the dashboard to access Aurora Serverless. In order to accomplish this, I used a common implementation combination of Lambda and API Gateway. The cloud functions to conduct CRUD operations and more specific data processing are hosted on Lambda. API Gateway is configured with the necessary IAM roles to access the correct Aurora Serverless databases. The Lambda functions are then linked to the API Gateway and called from the frontend dashboard. This is done mainly for security so that the data stored inside the database is not widely accessible to anyone who knows where it is. This implementation is also highly streamlined due to how commonly it is used, the architecture is depicted in Figure 4 (Eichorst and Megler, 2017).

Figure 4

Primary CloudFormation Template



The pipeline streamlines the entire data generation to the visualization process. It is triggered every night by a Jenkins job and the code within the pipeline is run on a virtual environment. This allows the pipeline to easily catch any log failures that may occur. The logs are sent to AWS CloudWatch and can be instrumental in identifying the root cause of the errors.

The high availability and reliability of AWS give users security in knowing that their data stored on its servers will not be lost. The database that I chose in my architecture has multiple backups and only a near complete failure of the entire network along with the destruction of the physical AWS servers can wipe out that data. This ensures that our data will not be lost which is a big part of ensuring that our data isn't flawed. Missing or incomplete data is a massive cause of misleading data. The built in security features of AWS also prevent outside actors from intercepting the data or obtaining access to the servers and modifying that data. This added security helps to further protect the data and ensure its authenticity.

While building this pipeline I had SCOT in mind and had the express goal of eliminating personal and social bias. Therefore the tools act heavily to clean the inserted data, protect the data from outside

manipulation, and present the data in a concise and non-assumptive way that doesn't push the user to any conclusion. The data is simply presented.

Conclusion

Cloud technology is widely available and provides companies with powerful tools to develop services that can process massive amounts of data. Building these services is key to successfully using the data available to us on a mass scale. However, it is necessary to keep in mind possible biases in the data and try to build pipelines that can help to eliminate these flaws. While building out these pipelines it is important to implement an independent data cleaning service that acts as a quality tester of the data. Engineers should understand SCOT and work with it by instilling in themselves values of correctness which will then be reflected in their work. Cloud services such as AWS help to secure data and to make sure that stored data doesn't get lost. These two factors are absolute necessities in insuring the authenticity and trustworthiness of data.

More work needs to be done on this, more specifically in identifying other factors that can be enhanced or done better in order to secure clean data. Engineers in the future who collect and analyze data or who build systems to do so need to always be on high alert. They should build systems that seek to keep the integrity of data.

Data is an inescapable part of our lives. We need to learn to leverage it to achieve our own goals but we also need to be proactive in identifying and preventing its flaws.

REFERENCES

Gavin, M. (2019). Business analytics: What it is & why it's important: HBS Online. *Business Insights Blog*. https://online.hbs.edu/blog/post/importance-of-business-analytics

Mesbahi, M. R., Rahmani, A. M., & Hosseinzadeh, M. (2018). Reliability and high availability in cloud computing environments: A reference roadmap - human-centric computing and information sciences. SpringerOpen. https://hcis-journal.springeropen.com/articles/10.1186/s13673-018-0143-8

Mezzalira, L., Hyatt, L., Denti, V., & Jaupaj, Z. (2022). Let's Architect! Tools for Cloud Architects. *Amazon*. https://aws.amazon.com/blogs/architecture/lets-architect-tools-for-cloud-architects/

Kava, P., & Gong, C. (2020). AWS serverless data analytics pipeline reference architecture. *Amazon*. https://aws.amazon.com/blogs/big-data/aws-serverless-data-analytics-pipeline-reference-architecture/

Villalba, M. (2022, April 21). Amazon Aurora Serverless v2 is Generally Available: Instant Scaling for Demanding Workloads. *Amazon*. https://aws.amazon.com/blogs/aws/amazon-aurora-serverless-v2-is-generally-available-instant-scaling-for-demanding-workloads/

Eichorst, B., & Megler, V. (2017, September 12). How to Provision Complex, On-Demand Infrastructures by Using Amazon API Gateway and AWS Lambda. *Amazon*. https://aws.amazon.com/blogs/compute/how-to-provision-complex-on-demand-infrastructures-by-using-amazon-api-gateway-and-aws-lambda/

Palantir. (2022, May 19). Product design at Palantir: Q&A with the team. *Medium*. https://blog.palantir.com/product-design-at-palantir-q-a-with-the-team-e93a82e9c74

AWS. (1994). Free Data Lakes and Analytics on AWS. *Amazon*. https://aws.amazon.com/free/analytics/?trk=25daa325-0213-449e-9872-4307e16223cb&sc_channel=ps&s_kwcid=AL%214422%213%21524495195940%21p%21%21g%21%21aws%2Bdata%2Banalytics&ef_id=Cj0KCQjw2cWgBhDYARIsALggUhr3ax3IGFrJ0atYkJXJhehML1HFrpWsQU67Zi0HeQabrBTHQOvWf48aAq4PEALw_wcB%3AG%3As

Zara, M. (2020, January 15). Using artificial intelligence to detect product defects with AWS Step Functions. *Amazon*. https://aws.amazon.com/blogs/compute/using-artificial-intelligence-to-detect-product-defects-with-aws-step-functions/

AWS. (1984). AWS Global Infrastructure. *Amazon*. https://aws.amazon.com/about-aws/global-infrastructure/#:~:text=The%20AWS%20Cloud%20spans%2099,%2C%20New%20Zealand%2C%20and%20Thailand

Palantir. (2023). Offerings: Artificial intelligence & machine learning. *Palantir*. https://www.palantir.com/offerings/ai-ml/

Redman, T. (2016, September 22). Bad Data costs the U.S. $3 trillion per year. *Harvard Business Review*. https://hbr.org/2016/09/bad-data-costs-the-u-s-3-trillion-per-year

Palantir. (2023). Offerings: Defense. *Palantir*. https://www.palantir.com/offerings/defense/

AWS. (1979). Training and Certification. *Amazon*. https://aws.amazon.com/training/?trk=578b2801-ffbb-4021-a1db-01a0bafb3a4a&sc_channel=ps&ef_id=Cj0KCQjwxMmhBhDJARIsANFGOStPse8bOzSy5ECEVtATVSdA_JwrslC38cC_qgYk_t7toHen_UAwXeQaAkeBEALw_wcB%3AG%3As&s_kwcid=AL%214422%213%21507162072467%21p%21%21g%21%21amazon+web+services%2112563449882%21121199905084