Data Pipelines: Ways Data Collection and Analysis Pipelines Can Be Built Through Cloud Services

A Technical Report submitted to the Department of Computer Science

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

Richard Jacques, Department of Computer Science

ABSTRACT

A Quality Assurance team at Amazon needed a service that could simplify and automate test data log collection, test result analysis, and visualize the analysis at scale. My solution involved leveraging Amazon Web Services (AWS) to build out an automated data pipeline capable of running, collecting, and analyzing hundreds of thousands of test logs per day, and connecting it to a dashboard web-app which displays charts and graphs of the resulting analysis. The AWS cloud services used to architect the data pipeline consisted of Code Pipeline, CodeBuild, EventBridge, SES, and Aurora Serverless. I wrote the dashboard using React Framework and ChartJS, hosted on AWS Amplify and connected to the data pipeline through AWS Lambda and AWS API Gateway. The final outcome was a success as it helped fill a vital need within the team for fast and reliable test data analysis. The complexity of data collection is abstracted away allowing engineers to better focus on the tests. The dashboard allows managers and engineers to quickly gain an understanding of the test result data without having to do complex analysis. This solution can be expanded in the future to handle teams with different data, allowing more teams to use the service. The web-app dashboard can also be improved upon by implementing more user customization in the graph and chart visualizations.

1. INTRODUCTION

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. A common problem many companies face is how best to conduct useful data collection and analysis while also investing a minimal amount of capital and time. Companies like Palantir provide data analysis services using Machine Learning and Artificial Intelligence. Companies purchase their services and give up their data in return for valuable insight generated from their 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.

2. RELATED WORKS

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, 2019). 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 find it easier to use AWS services instead of creating their own.

Some companies such as Palantir choose to develop their own data analysis 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 (Mezzalira, et al, 2022).

AWS also provides many official guides on how best to leverage their services for a variety of projects. These projects range from large scale applications deployed by big corporations to small services developed by university students. The low-cost of AWS helps to lower the entrance bar for many users to get started. Also there is very extensive documentation as well as a very active community that can help new users to familiarize themselves with the services and to more easily find solutions to their wide-ranging problems (Kava and Gong, 2020).

Cloud services provided by AWS, Google, and Salesforce are universally recognized and considered to be reliable by many in the industry (Mesbahi, et al, 2018). The learning curve for these services are low compared to actually building the services, and the complexities and costs of maintaining these systems are abstracted away.

3. SYSTEM DESIGN

When used properly cloud services can be integrated together to build powerful data processing pipelines. The main challenge is in choosing the correct services to fit the needs of the project without incurring too much cost or complexity.

3.1. PALANTIR ARCHITECTURE

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 business, is show in Figure 1 (Mezzalira, et al, 2022):

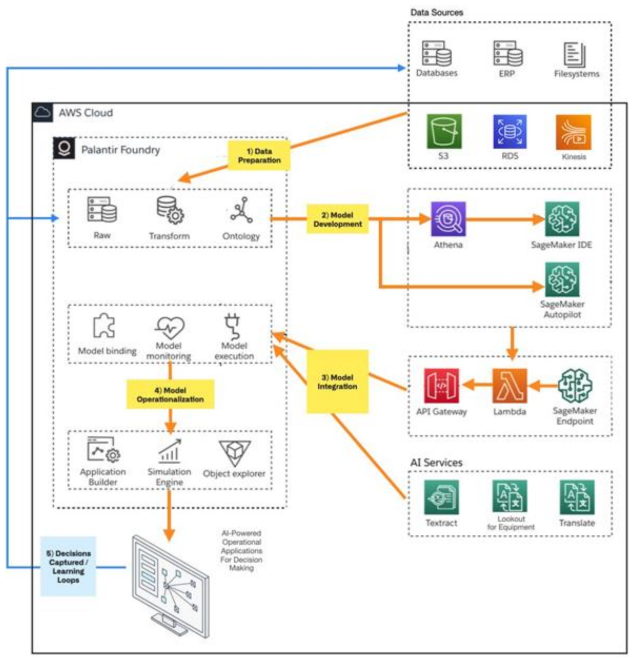

Figure 1: Palantir Foundry Architecture

Complex Machine Learning models are applied within Palantir's Foundry platform. The data used by the models is 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. 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.

3.2. FULLY AWS ARCHITECTURE

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):

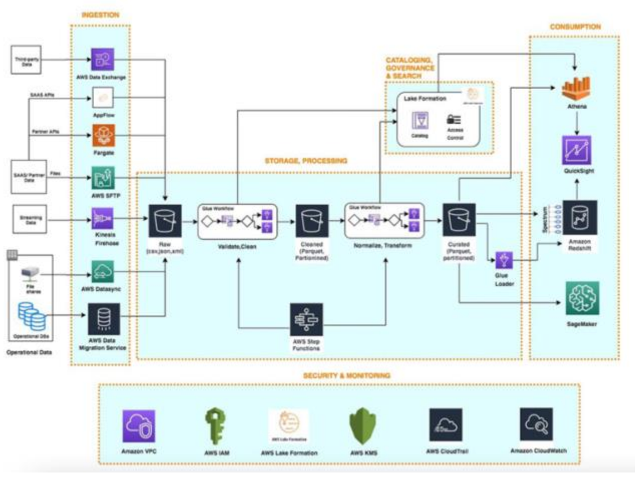

Figure 2: Serverless Data Lake Centric Analytics Architecture

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.

3.3. SYSTEM REQUIREMENTS

My project 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 in 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.

3.4. ARCHITECTURE OVERVIEW

The architecture I chose to build in order to meet the requirements set out in section 3.3. 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 gives 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 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.

3.5. KEY COMPONENTS

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 compute and storage which allows it to replicate data seamlessly and increase the availability of the service, this functionality is depicted in Figure 3.

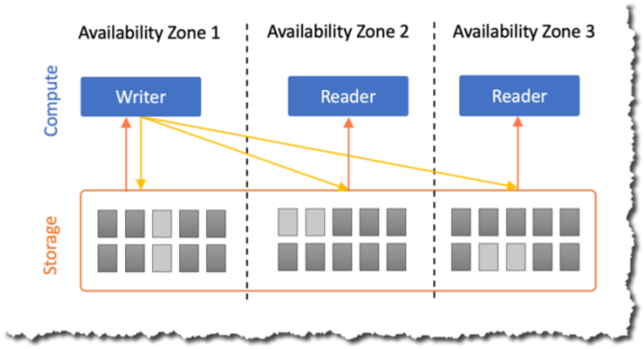

Figure 3: Aurora Separation of Storage and Computation

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).

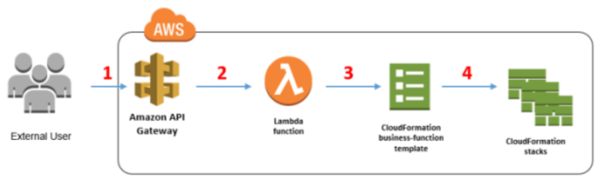

Figure 4: Primary CloudFormation Template

The pipeline streamlines the entire data generation to 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 to identifying the root cause of the errors.

4. RESULTS

The dashboard is being actively used by the quality assurance team on a daily basis to log the manual tests that they do. It is a massive improvement from having to log their tests in an Excel sheet that is then shared among the team. The dashboard gives them a centralized place to view, store, and analyze their test results. The main page of the dashboard which contains informative charts and graphs based on the data is put on three big monitors in the office. This allows engineers and managers to quickly glance and see trends in the data or spikes of failures. This, in turn, allows them to more easily notice problems and helps to raise awareness for commonly failing tests.

The pipeline is an enhancement to an existing system that served the same purpose. The main improvement the pipeline offers is that it is done completely over the cloud instead of locally. This limits the variables that could causes failures during the running of the tests, like Wifi outage or computer running out of battery. It also streamlines the entire process as all the necessary scripts are ran back-to-back in order.

5. CONCLUSION

Data is one of the most plentiful resources of the modern age. Hidden within it is a wealth of knowledge available only to those that can successfully collect and analyze it. Cloud services serve as a flexible, low-cost, and straightforward way to build out powerful data processing pipelines. Engineers can use cloud services for a variety of use cases ranging from small projects to large-scale corporate productions. Most of the complexities associated with developing and maintaining the tools necessary to host data pipelines are completely abstracted away with cloud services. The pipelines and associated dashboard I created for the Quality Assurance team at Amazon were developed completely on AWS. For the time being, cloud services are the best way to build, host, and deploy data pipelines.

6. FUTURE WORK

The natural next-step of this project would be to make it support more types of data and to conduct more powerful analysis on the data. The dashboard can be expanded upon by introducing more analysis functionality. Machine learning algorithms can be applied to the underlying data in order to create more advanced analysis, and the results can be displayed on the dashboard.

The pipeline can be expanded to include more types of media. Currently my data pipeline can only support digital-text based data. Audio, visual, and written-text data can be supported by including the relevant services offered by AWS or other cloud providers.

REFERENCES

[1] Gavin, M., 2019. Business Analytics: What It Is & Why It's Important | HBS Online. [online] Harvard Business School. Available at: <https://online.hbs.edu/blog/post/importance-of-business-analytics> [Accessed 23 September 2022].

[2] Mesbahi, M., Rahmani, A. and Hosseinzadeh, M., 2018. Reliability and high availability in cloud computing environments: a reference roadmap. Human-centric Computing and Information Sciences, [online] 8(1). Available at: <https://hcis-journal.springeropen.com/articles/10.1186/s13673-018-0143-8> [Accessed 23 September 2022].

[3] Mezzalira, L., Hyatt, L., Denti, V. and Jaupaj, Z., 2022. Let's Architect! Tools for Cloud Architects | Amazon Web Services. [online] Amazon Web Services. Available at: <https://aws.amazon.com/blogs/architecture/lets-architect-tools-for-cloud-architects/> [Accessed 23 September 2022].

[4] Kava, P. and Gong, C., 2020. AWS serverless data analytics pipeline reference architecture | Amazon Web Services. [online] Amazon Web Services. Available at: <https://aws.amazon.com/blogs/big-data/aws-serverless-data-analytics-pipeline-reference-architecture/> [Accessed 23 September 2022].

[5] Villalba, M., 2022. Amazon Aurora Serverless V2 is generally available: Instant scaling for ..., AWS. [online] AWS News Blog. Available at: https://aws.amazon.com/blogs/aws/amazon-aurora-serverless-v2-is-generally-available-instant-scaling-for-demanding-workloads/ [Accessed: November 30, 2022].

[6] Eichorst, B. and Megler, V., 2017. Blogs, Amazon. [online] AWS Compute Blog. Available at: https://aws.amazon.com/blogs/compute/how-to-provision-complex-on-demand-infrastructures-by-using-amazon-api-gateway-and-aws-lambda/ [Accessed: November 29, 2022].