https://doi.org/10.31891/csit-2023-2-9
UDC 004.89

Markiyan PYTS, Ivanna DRONYUK
Lviv Polytechnic National University

THEORETICAL BACKGROUND FOR CREATING REAL WORLD DATA LAKE ARCHITECTURE

Data Lakes are the methods for storing and managing large quantities of unstructured data. Modern enterprises and small businesses, regardless of their size, can use this data to derive valuable insights about their business, such as process improvements or product usage. Although this approach to extracting insights is powerful, only some studies describe the actual implementation architectures of data lakes and warehouses.

There are a lot of high-level studies of data lakes, their use cases, and approaches to data analysis. Still, we did not discover a practical guide on how to appropriately unite the available tools to set up a complete data lake capable also of consuming live event stream data, e.g. ingesting data about a user visiting a product website or interacting with some product feature or consuming IoT device event.

The main goal of this article is to describe the architecture using AWS to create a robust, cheap-to-maintain, and scalable solution for Data Lake and Data Warehouse. It can be used by small Software as a Service (SAAS) companies, big enterprises, or individual researchers to build the base of such solutions with a clear guideline of all moving pieces and an understanding of how they are connected.

The article provides a broad overview of setting up a data lake on AWS (Amazon Web Services). It covers setting up an Application Programming Interface (API) to consume data, store data, visualize data, and the ability to create data lakes across multiple AWS accounts quickly with a single Command-line Interface (CLI) command.

This is useful for creating a scalable data lake or data warehouse setup that doesn't require much manual work. We describe how such design can be done using infrastructure as a code approach to achieve this and propose AWS architecture for solving the task of compelling data storage. The article provides a diagram of the proposed architecture accompanied by a high-level description and theoretical background.

Keywords: business intelligence, data science, infrastructure as a code, data warehouse, data lake, Data lake architecture, Amazon Web Services.

Маркіян ПИЦ, Іванна ДРОНЮК
Національний Університет «Львівська Політехніка»

ТЕОРЕТИЧНА ОСНОВА ДЛЯ СТВОРЕННЯ АРХІТЕКТУРИ REAL WORLD DATA LAKE

Data Lake — це методи зберігання та керування великою кількістю неструктурованих даних. Сучасні середні та малі підприємства, незалежно від їх розміру, можуть використовувати ці дані, щоб отримати важливу інформацію про свій бізнес, наприклад, удосконалення процесів або використання продукту.

Існує багато досліджень високого рівня щодо озер даних, їх використання та підходів до аналізу даних. Проте ми не знайшли практичного посібника про те, як належним чином об'єднати наявні інструменти для створення повноцінного озера даних, здатного також споживати дані потоку подій у реальному часі, наприклад, поглинати дані про відвідування користувачем веб-сайту продукту або взаємодію з певною функцією продукту, або споживати події з пристроїв Інтернету речей.

Основна мета цієї статті - описати архітектуру з використанням AWS для створення надійного, дешевого в обслуговуванні та масштабованого рішення для озера даних та сховища даних. Вона може бути використана невеликими компаніями, що надають програмне забезпечення як послугу (SAAS), великими підприємствами або окремими дослідниками для побудови бази таких рішень з чітким описом всіх рухомих частин і розумінням того, як вони пов'язані між собою.

У статті представлено широкий огляд налаштування Data Lake на AWS (веб-сервіси Amazon). Він охоплює налаштування інтерфейсу прикладного програмування (API) для споживання даних, зберігання даних, візуалізації даних і можливості швидкого створення Data Lake для кількох облікових записів AWS за допомогою однієї команди інтерфейсу командного рядка (CLI).

Запропонований підхід корисний для створення масштабованого Data Lake або налаштування сховища даних, яке не вимагає багато ручної роботи. Для досягнення цього використано підхід до інфраструктури як коду. Запропонована AWS архітектура для вирішення задачі ефективності зберігання даних. Стаття демонструє діаграму запропонованої архітектури та її високорівневий опис з теоритичним підгрунтям.

Ключові слова: бізнес-аналітика, наука про дані, інфраструктура як код, сховище даних, Data Lake архітектура, Data Lake, веб-сервіси Amazon.

Introduction

Data lakes help enterprises store and manage large amounts of structured and unstructured data in a cost-effective, secure, and easily accessible way. It is used as a source of data for Business intelligence (BI) [1][10] tools.

Business Intelligence is a crucial process for organizations [12] that involves transforming raw data into meaningful information for decision-making. BI[10] helps companies identify potential opportunities, and threats, and make better decisions.

The setup of a data lake is usually a long and complicated process that requires a lot of expertise in the area and appropriate tools to assemble an initial solution to start ingesting data from various sources. In addition to this,

there are several other important considerations, such as security, cost efficiency [1], access to modern data transformation [16], and visualization tools [5]. Ensuring that a data lake's IT infrastructure can be managed easily is also crucial to prevent a single manual configuration error from causing the entire system to break down.

Companies and individuals who want to start implementing a data lake require a clear guide on the topic, to know which services to use, what each service in the data lake is doing, and how they are connected.

There are a lot of high-level studies [2] of data lakes, their use cases, and approaches to data analysis. Still, we did not discover a practical guide on how to appropriately unite the available tools to set up a complete data lake capable also of consuming live event stream data, e.g. ingesting data about a user visiting a product website or interacting with some product feature or consuming IoT [13] device event.

The main goal of this article is to describe the architecture using AWS [6] to create a robust, cheap-to-maintain, and scalable solution for Data Lake and Data Warehouse. It can be used by small Software as a Service (SAAS) companies, big enterprises, or individual researchers to build the base of such solutions with a clear guideline of all moving pieces and an understanding of how they are connected.

Theoretical background for creating architecture

Regarding storing data, there are two popular options: a Data Lake [15] or a Data Warehouse [2]. While both serve the same purpose of storing data, there are some critical differences between them. A data lake is a repository that allows for storing raw, unstructured data in its native format. In contrast, a data warehouse is designed to store structured data that has already been processed and organized.

Data lakes are ideal for storing large volumes of data that may be unstructured or require further processing. This makes them particularly useful for data scientists and analysts who need to analyze data in its raw form. In contrast, data warehouses are ideal for storing structured data ready for analysis. They are well-suited for business intelligence and reporting applications that require fast access to data.

With AWS services [8] available, choosing between a data lake or a warehouse is unnecessary. We can use tools such as AWS Glue Studio or AWS Glue DataBrew to prepare the data for further analysis. Learning Data Science [3,7] basics are also beneficial to get the basic knowledge needed to use data transformation [16] and visualization tools.

Having the difference defined, we should also understand that data from the Data Lake can be post-processed and used in the data warehouse. These approaches usually coexist in modern setups as different stages of data flow.

Data Lake setup needs many different functionalities and tools glued to assemble the working solution. We recommend AWS [8] for data lake infrastructure based on the [11] course and the following factors:

- Many modern SAAS Businesses and companies are using AWS for infrastructure already because AWS is the biggest cloud provider in the world at the time of this article's creation. So it will be convenient to use familiar services for businesses that use AWS already.
- AWS exists in the cloud. Therefore data that is being stored can be shared with personnel all over the world.
- AWS follows deny access by default principle which helps ensure that only authorized people will access the data.

AWS Lambda is a highly scalable, event-driven computing service that allows developers to run code without provisioning or managing servers. It supports various programming languages, including Node.js, Python, and Java, and can be used to build a wide range of applications, including web applications, mobile backends, and IoT [13] applications. Various events and API Gateway ones can trigger Lambda functions. Lambda is an essential building block for many data lake architectures, as it allows for data processing in real time. Access if this data format is what we expect to consume and use AWS Software Development Kit (SDK) to push data farther to the Kinesis Firehose delivery stream, which will push it into the Data Lake.

AWS Kinesis Firehose is an essential building block for many data lake architectures, as it allows for data processing in real time. Other AWS services, such as data transformation and visualization, can consume this data stream for further processing. It can continuously collect and load data from various sources, such as IoT [13] devices, clickstreams, social media feeds, and logs. Kinesis Firehose takes care of all the underlying infrastructure and scaling and focuses on analyzing data. Kinesis Firehose also provides built-in support for data delivery to Amazon S3, which makes it easy to store and cost-effectively manage large volumes of data. Overall, AWS Kinesis Firehose is a powerful and flexible tool for collecting, processing and delivering real-time data streams to other AWS services.

Amazon Simple Storage Service (S3) is a highly scalable and reliable object storage service offered by AWS. It enables developers and businesses to store and retrieve data anywhere on the web. One of the critical benefits of S3 is its simplicity. S3 provides a simple web services interface for storing and retrieving data from any location on the internet.

The S3 provides several storage classes for different data access patterns and cost requirements. The most commonly used storage classes are Standard, Standard Infrequent Access (Standard-IA), and Glacier. The Standard storage class is suitable for frequently accessed data and provides low latency and high throughput performance. The Standard storage class is suitable for infrequently accessed data but offers low latency and high throughput performance. The Glacier storage class is designed for long-term data archiving and offers the lowest storage costs but higher retrieval times.

Overall, Amazon S3 is a highly scalable, reliable, and secure object storage service that provides simple and cost-effective storage for all data types. With its numerous features and integration with other AWS services, S3 has become an essential tool for many businesses and developers looking to store and manage their data in the cloud. The interesting fact is that even though S3 is file storage, augmented with AWS Athena Capability, It can store data in a database-like format called Parquet. This is a much cheaper storage option than traditional databases, yet Athena can query this data with SQL like a usual database.

AWS Lake Formation [11] is a tool that simplifies the setup and management of data lakes on AWS. It provides various features, such as data ingestion, data cataloging, and access control, which can help organizations get up and running with their data lake quickly and easily. One of the key benefits of AWS Lake Formation is its ease of use. We use AWS Lake Formation to automate everyday tasks, such as data ingestion, cataloging, and access control.

In summary, AWS Lake Formation is a robust and comprehensive service that helps organizations build, secure, and manage data lakes quickly and easily. Its advanced security features, ease of use, and integration with other AWS services make it a popular choice for organizations looking to store and analyze large amounts of data in the cloud.

Features for creating a proposed Architecture

In the proposed architecture, Lake Formation monitors AWS Glue databases and tables that our infrastructure as a code solution will define and deploy, which can be referenced in our Kinesis Firehouse delivery stream described above.

AWS Glue is a fully managed extract, transform, and load (ETL) service offered by Amazon Web Services (AWS). AWS Glue makes it easy to automate the process of discovering, cataloging, and cleaning data, which saves time and resources.

One of the key features of AWS Glue is its ability to automatically discover and catalog data from various sources, such as databases, file systems, and streaming services. AWS Glue crawls data sources, extracts metadata, and creates a centralized metadata repository called the AWS Glue Data Catalog. This makes it easy to search, discover, and query data across multiple sources.

AWS Glue supports various data formats, including CSV, JSON, Parquet, and ORC. It also provides various connectivity options, such as JDBC, ODBC, and AWS services like Amazon S3 and Redshift. This makes it easy to load data from various sources and transform it into the desired format for analysis.

In the proposed architecture, we use the AWS glue database and table (defined in our infrastructure as a code solution) to define the data schema of events which is expected to come into the data lake. The kinesis delivery stream, which consumes events from the outside world, is associated with the appropriate glue table. It validates data that comes with the event against glue table schema and if valid, automatically puts this data into the S3 bucket as a Parquet file which AWS Athena can read.

AWS Athena is a powerful and fully managed serverless query service offered by Amazon Web Services (AWS). It allows querying data in Amazon S3 using standard SQL syntax without the need for complex data processing infrastructure. With AWS Athena, we analyze data stored in Amazon S3 and extract insights quickly and easily.

One of the key benefits of AWS Athena is its serverless architecture. There are no servers to manage or provision capacity. AWS Athena automatically scales queries to match the data's size and complexity, making it ideal for ad-hoc queries and exploratory analysis. AWS Athena provides a simple and intuitive interface for querying data.

Amazon QuickSight is a cloud-based business intelligence [1] service that provides easy-to-use and interactive dashboards and visualizations [5]. It allows users to access and analyze data from various sources, including AWS data services, databases, and third-party applications. With Amazon QuickSight, businesses can gain valuable insights into their data and make informed decisions faster. One of the key benefits of Amazon QuickSight is its ease of use. It provides a simple and intuitive user interface that allows users to create dashboards and visualizations without the need for extensive technical knowledge. Users can quickly and easily connect to their data sources, choose from various visualization options, and create interactive dashboards with drag-and-drop functionality.

Amazon QuickSight supports various data sources, including Amazon S3 via Athena. It provides a range of visualization options, including charts, graphs, and maps, which allows users to create compelling and interactive dashboards. Users can customize their dashboards with branding and layout options, as well as add filters and parameters to enable interactive data exploration. Amazon QuickSight also provides AI-powered features such as auto-narratives that provide contextual explanations to data in the dashboard, anomaly detection, and forecasting.

Infrastructure as code (IaC) [4] is a practice of managing and provisioning technology infrastructure through machine-readable definition files, rather than manually configuring hardware devices and software systems. In IaC, infrastructure is defined using code, which can be version-controlled, tested, and deployed in the same way as software code. The code is written in a declarative format, which describes the desired state of the infrastructure. IaC tools, such as CloudFormation, interpret these definitions and automatically provision and manage the infrastructure accordingly. IaC offers several benefits over traditional infrastructure management approaches.

There are such stages:

- Firstly, it allows for creating of reproducible infrastructure, which can be easily shared and reused across different teams and projects.
- Secondly, it enables the automation of infrastructure provisioning and management, reducing the need for manual intervention and ensuring consistency and reliability.
- Thirdly, it provides a clear audit trail of changes to the infrastructure, which can be tracked and audited easily.

AWS CloudFormation is a powerful and fully managed service to create and manage resources in a repeatable and automated way. One of the key benefits of AWS CloudFormation is its ability to automate infrastructure provisioning and management. By defining the desired state of infrastructure in a template, we can create and manage resources in a repeatable and consistent way. AWS CloudFormation automatically handles the complexity of resource dependencies and orchestrates the deployment of resources in the correct order.

For the proposed architecture, we used AWS CloudFormation to provide all infrastructure required for our data lake. Without it, it would be very hard to maintain all the infrastructure pieces manually in the AWS management console. It would be much more complicated if we needed to deploy similar data lakes in different AWS accounts.

Another significant benefit is that the CloudFormation code can be stored on Git code version control [14]. This provides the advantage of knowing how our infrastructure looked in the past and how it looks now and grants the ability to modify and augment it in the future. Also, we can roll back to the latest working copy if some wrong modification got introduced by mistake.

CloudFormation is an excellent tool on its own, but it's rarely used as is. There are popular tools and frameworks which allow working with it at a higher level, increasing the speed of implementation. One such tool is Serverless Framework [9]. With CloudFormation, we could define infrastructure in YAML format once and use Serverless Framework CLI to deploy or destroy the infrastructure if it's not needed anymore.

The Serverless Framework [9] is an open-source framework that enables developers to build, deploy, and manage serverless applications easily. It abstracts the underlying infrastructure details and allows developers to focus on writing code, reducing the time and complexity required to develop and deploy serverless applications.

The Proposed Architecture Overview

In this section, we present the proposed architecture of a data lake. Fig. 1 represents the architecture that is powering the solution. It fits both Data Lakes and Data Warehouses.

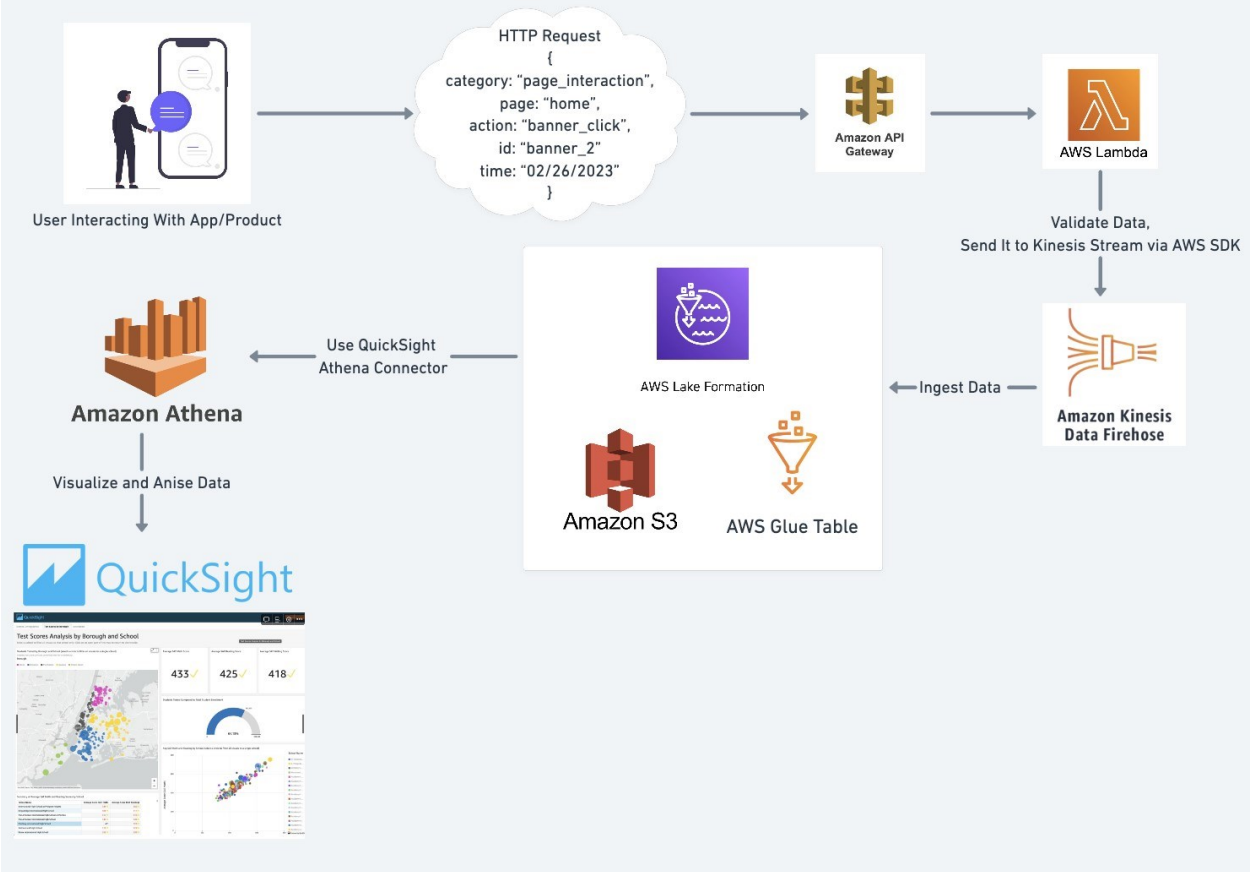

Fig. 1. Data Lake Architecture

Fig. 1 describes a situation when we want more governance over the data to be ingested and stored in the system, which is closer to the Data Warehouse scenario, though can also be used as Data Lake later, since data is in S3 while being originally formatted to serve the user interactions analytics purposes (e.g. tracking user clicks on product webpage), can be post-processed later using many data processing tools. AWS provides to actually transform this data and use it in different sorts of analysis and use cases.

Conclusion

We proposed to create an AWS Architecture that allows storing large amounts of data in the data lake with the optional ability to filter what data format is being stored in the data lake and what data is discarded by the system, to achieve this, we have AWS lambda function exposed to the outside world via AWS API Gateway that acts like a filtering gateway between the data lake and events data that is being ingested into it from external systems and apps. This way, the solution is flexible and can be used to create a Data Warehouse if this approach is preferred.

The lambda function sends data to the Kinesis Firehose delivery stream, which can handle close to the infinite scale of data being sent to it, so we don't have to worry about the system being overwhelmed by external events' quantity and frequency. Valid data is being ingested into the S3 bucket, which is associated with the AWS Glue database table.

S3 storage which is used in the proposed architecture, is much cheaper than regular databases, it does not limit the amount of data that can be stored. This way, we can be sure that as more data comes into the data lake, we will not run out of storage space.

With Infrastructure As A Code proposed in the architecture, we can easily maintain the infrastructure of the data lake, and extend it with new tables, data storage types, and schemas. Also, the requirement is to move the infrastructure into a different AWS account and patch or remove some of its pieces. In that case, it is safe and easy since CloudFormation remembers all the parts of the infrastructure and the connections between them for us. This is especially useful if we have an AWS account that hosts more things than our data lake, which is common among organizations.

Architecture is helpful for companies and individuals who want to set up their own BI data stores for many purposes like analysis of SAAS products usage, monitoring of IoT devices, etc. Data which is stored in the data lakes is also helpful for the creation of machine learning models.

In the future, this architecture can be used as a base of more specialized data warehousing solutions targeting specific business use cases, e.g, e-commerce, IoT devices, including medical ones, etc. It can be easily adjusted for this purpose we only require to change/add schema on AWS Glue Table and adjust filtering logic for events allowed for ingestion in the entry lambda function.

References

1. Muriithi, G. M. and J. E. Kotzé. (2013) "A conceptual framework for delivering cost-effective business intelligence solutions as a service," in Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference, p.96-100.
2. Marilex Rea Llave. (2018) "Data lakes in business intelligence: reporting from the trenches" Procedia Computer Science 138:p. 516-524.
3. Davenport, T. H. and D. J. Patil. (2012) "Data scientist: the sexiest job of the 21st century." Harvard Business Review 90 (10): p.70-79.
4. Dalla S., Palma D., Di Nucci D., Tamburri A. (2020) "AnsibleMetrics: A Python library for measuring Infrastructure-as-Code blueprints in Ansible" SoftwareX VOLUME 12, 100633
5. George L. C., Guo Ya., Stepanov D., Kumar V., Peri R., Elvitigala R. L., Spichkova M. (2020) "Usage visualization for the AWS services". Procedia Computer Science, Vol. 176, p. 3710-3717
6. Sudhanshu M., Lakhera G., Srivastava A. K., M. Kumar (2021). "Cost analysis of amazon web services – From an eye of architect and developer". Materials Today: Proceedings 46(20), p. 10757-10760
7. Eremenko K. "Data Science A-Z™: Real-Life Data Science Exercises Included by "Udemy course URL: https://www.udemy.com/course/datascience
8. Maarek S. "Ultimate AWS Certified Developer Associate" Udemy course URL: https://www.udemy.com/course/aws-certified-developer-associate-dva-c01
9. Weinberger A. "Serverless Framework Bootcamp: Node.js, AWS & Microservices" Udemy course URL: https://www.udemy.com/course/serverless-framework
10. Oyku I., Jones M. C., and Sidorova A. (2013) "Business intelligence (BI) success and the role of BI capabilities." Decision Support Systems 56 (1), p.361-370.
11. Owoyemi Y. "Build a Secure Data Lake in AWS using AWS Lake Formation" Udemy course URL: https://www.udemy.com/course/build-a-secure-data-lake-in-aws-using-aws-lake-formation
12. Davenport, T. H., and Dyché J.(2013) "Big data in big companies. "International Institute for Analytics 3. URL: https://www.iqpc.com/media/7863/11710.pdf
13. Kumar S., Tiwari P., Zymbler M. (2019) "Internet of Things is a revolutionary approach for future technology enhancement: a review" Journal of Big Data 6, 111
14. César J., Ríos C., Embury S. M., Eraslan S. (2022) "A unifying framework for the systematic analysis of Git workflows" Information and Software Technology 145: 106811
15. Larson D. and Chang V. (2016) "A review and future direction of agile, business intelligence, analytics and data science." International Journal of Information Management 36 (5): p.700-710.
16. Stein B. and Morrison A.. (2014) "The enterprise data lake: Better integration and deeper analytics." PwC Technology Forecast: Rethinking integration 1 p.1-9.

Markiyan Pyts **Маркіян Пиц**	Ph.D. student at Artificial Intelligence Department, Lviv Polytechnic National University https://orcid.org/0009-0005-1814-1586 e-mail: markiyan.pyts@gmail.com	аспірант кафедри систем штучного інтелекту Національний університет «Львівська політехніка»
Ivanna Dronyuk **Іванна Дронюк**	Ph.D., Associate Professor of Artificial Intelligence Department, Lviv Polytechnic National University https://orcid.org/ 0000-0003-1667-2584 e-mail: Ivanna.m.droniuk@lpnu.ua	доцент кафедри систем штучного інтелекту, Національний університет «Львівська політехніка»