SFL Scientific Client Presentation

What is a Data Lake?

Data Lakes fill a crucial role to any data driven organization and are paramount to the success of ensuring productionalized data assets are supported so that research and development can flourish.

Data Lakes serve as the central data repository for all incoming data assets to your organization. Data Lakes provide a persistent storage solution for raw data in all forms: unstructured, semi-structured, and structured. This means that data from video or audio files, XML or JSON, or relational data can all be stored in a central location that your Data Engineers, Data Scientists, and Developers can access. This functionality is made possible because Data Lakes do not enforce any schema on read.

In addition to handling a large variety of data, Data Lakes are also optimal for handling a large volume of data. Data Lake solution providers make certain your Data Lake can scale as your data needs evolve. The benefit of this is that you will not need to continually migrate data as your business demands increase. Furthermore, most Data Lake providers help ensure your data scales horizontally to provide redundancy of your data assets. In turn, local failures or outages will not wipe away your data and will keep data access latencies low as data storage can be provisioned all over the world, keeping your data close to its users.

Lastly, Data Lakes provide a cost effective storage solution since data is stored raw. Storing data in its raw form is often ideal because no additional processing or metadata will be associated with it.

Now that we have a firm understanding of what a Data Lake is, we can compare them to Data Warehouses to better understand how both serve separate functions to build a single data solution to power your organization.

While Data Lakes store data in its raw form, Data Warehouses store processed data. Typically, ETL (Extract, Transform, Load) pipelines will read raw data from a Data Lake, process the data, and then store that data in a Data Warehouse. The data stored in a Data Warehouse typically has a defined business need. As a result, data stored in a Data Warehouse is structured and has a defined schema on write. Data Warehouses are associated with more traditional relational data stores like SQL based stores. You can think of this like a large, well-defined Excel sheet. As a result of this structured data storage and additional data processing, data stored in Data Warehouses often incurs a cost penalty compared to data stored in a Data Lake. However, this cost serves a purpose, as the data in a Data Warehouse is optimized for query performance and ensures the data is accessible to more business functions such as Business Analysts, Business Intelligence Engineers, Marketers, Finance, etc.

Explain Serverless Architecture along with its pros and cons:

Serverless architecture is a bit of a misnomer. Serverless architecture does not mean compute without servers. That would be impossible. Rather, serverless architecture refers to an infrastructure paradigm that removes the need to consider hardware and network resources while developing applications and ML products. This paradigm shift allows Software Engineers and Data Scientists to focus more on code, data models, and the user experience (UX) and less on the underlying infrastructure.

Serverless architectures are a core component of cloud computing. By taking resource and infrastructure management out of the hands of developers, serverless architectures enable resources to scale along with business needs and user volume. This offers a great benefit because it assures resources are efficiently adapted to changes and costs are managed.

Serverless architectures are largely event driven. For instance, changes such as file loads into your Data Lake, changes to your database, or HTTP requests will trigger actions, such as data workflows, messages, notifications, or database updates.

Serverless architectures offer a lot of promise and opportunity. They enable systems and applications to scale, allowing your products to consistently meet the demands of your customers. Scaling is typically conducted horizontally, which offers protection from environmental impacts and lower latency to consumers by providing global redundancies.

The efficient resource utilization of serverless architectures also provide cost-effective solutions–you only pay for what you need. Taking this one step further, there is less overhead and technical debt as hardware and infrastructure management are bundled and managed by a third party.

Lastly, the lower overhead and infrastructure abstraction provided by serverless architectures enable quick development and production deployments. This offers quick development iterations, allowing your organization to come to market quickly with new products and features.

While serverless architectures have many advantages, they are not a silver bullet. One major drawback of serverless architectures are their dependency and reliance on third parties to manage your infrastructure, and more importantly, your data assets. There are serious privacy and regulatory concerns that need to be considered before adopting a serverless architecture. An organization must be due diligent so any personally identifiable information (PII) is safeguarded while that data is off-premises.

This third party dependency can also lead to Vendor Lock-In, as vendors typically want organizations to sign multi-year contracts for their services and support. Additionally, while there are many solution providers that offer serverless architectures, there are benefits to staying with one provider for all services. For example, utilizing a single provider may enable seamless integration between services and employees only having to learn a single suite of tools. This may not be ideal as one team may prefer, or even require, a solution provided by a vendor that your organization does not do business with.

Another drawback of serverless architectures is that they increase the difficulty of testing and debugging code. Since code is placed on servers off-premises, developers may struggle with getting system read-outs for error messages or there could be other networking or environment issues that are difficult to troubleshoot. This results in a learning curve that is a source of technical debt.

Lastly, other minor drawbacks associated with serverless architectures are delayed cold starts or launches and the dependency on an internet connection to access your resources. These items are more inconveniences, but are worth considering based on the needs of your organization and business functions.

What is MLOps and how to approach from a tool and system perspective:

Machine Learning Operations, also known as MLOps, is an overarching term to describe the systems and tools used to manage the lifecycle of your AI and ML models. The lifecycle of AI and ML models, similar to the lifecycle of any commercial product, consists of four distinct phases: (1) Development, (2) Deployment or Launch, (3) Monitoring, and (4) Management. At each of these stages, and across each stage, tools and systems are required to make certain your people and software are fast, efficient, and conform to internal procedures, voice of customer, and regulatory bodies. MLOps is the discipline that bridges the gap between different job functions and stages of the product lifecycle to guarantee you can deliver your business objectives. We will now break down each of the four stages of the product life cycle and talk about what systems and tools can be utilized to enable success at each stage.

During development, your Data Scientists, Data Engineers, and Software Engineers need managed data and development environments to build products, features, pipelines, and applications. This means that these technical functions require data in various data stores such as SQL and NoSQL that they can query and investigate. Data Lake solutions provided by AWS, Azure, GCP, or Snowflake are common examples that will help deliver these goals. Additionally, a data registry service such as Alation indexes these data assets so that they can be searched to break down silos within the organization and ensure data is known and available for all functions.

In addition to data access, these technical developers also need a development environment, such as integrated development environments (IDEs) like Visual Studio Code and Atom, or notebook environments like Jupyter or Databricks. These platforms provide areas where developers can build code, test hypotheses, and execute deliverables to bring products and features to market.

An additional set of tools that fosters company culture and dialogue are chat and messaging tools such as Slack, Google Chat, or Microsoft Teams. These tools are vital in today’s environment with remote, hybrid, and in-office work environments across the globe. These tools will ensure all teams can collaborate and communicate efficiently.

Once a new product or feature has been sufficiently researched and developed, it must be tested prior to deployment. Ideally testing will be ongoing during development, but having sufficient unit tests and test suites will make sure your software and models are well defined and robust moving forward. Once these tests are established, it is best to place them into a CI/CD build pipeline. This enables these tests to be repeatedly enforced during any changes so that the product will perform as expected into perpetuity. Version control software such as Git, in combination of CI/CD solutions such as Jenkis, GitHub Actions, or GitLab CI/CD, provide a winning combination to enable production ready versions of code are clearly tagged and tested prior to deployment.

As models and products are released into production, it is paramount that their environment is configured appropriately. This is where tools such as Docker, Terraform, and Kubernetes come into play. These tools gurantee that run environments are reproducible and managed so that software and models deliver as expected and will not be hindered by dependency or runtime disconnects.

Machine learning models are nothing without data inputs. Data pipeline orchestration tools such as Airflow or Luigi make sure models can be trained with the click of a button and that production data is delivered to your models in a structured and consistent way. Orchestration tools are a great way to manage and control all data flow within your organization.

After models have been deployed into production, continued success will rely upon constant monitoring. Monitoring your solutions will ensure both that models are performing as expected and that they are delivering on the business objectives they set out to accomplish. To monitor your systems, dashboarding tools like Tableau, PowerBI, AWS Quicksight, and Apache Superset allow all KPIs to be available to you on demand and performance can be tracked over time. This will enable your team to react immediately to any unexpected problems and measure improvements.

Work on your models, features, and products does not stop with deployment! New features and improvements are constantly being added as you respond to the voice of your customers and the market. As such, keeping track of which versions of your model are available for production and the seamless change from one model to another is a key function of an ML Registry solution such as MLFlow.

Additionally, as your user base grows or changes over time, it is vital to have an infrastructure that changes with demand. This is where cloud providers and serverless architectures can provide a key function of handling all of your resource needs in a cost- and time-efficient manner.

Last but not least, documentation tools like Confluence are a great asset for a central repository to share information about development progress, product documentation, tutorials, and more. A tool like Confluence will help to break silos and allow information to be documented and accessible to members across your organization.

As you can see, MLOps is a large topic covering ample breadth and depth. However, the tools and systems you develop by employing MLOps best practices will ensure your organization is agile, fault tolerant, informed, and successful.