**8**

**Creating a Foundation for Data Platforms**

If there’s one big advantage of the public cloud, it’s that platforms that utilize it can hold massive amounts of data. That is one of the main reasons that companies use cloud technology. In the cloud, they can collect data from a variety of sources and start analyzing this data to improve their business strategy, products, and services. The first question that companies will ask themselves is: where do we start with building data platforms?

In this chapter, we will discuss the basic architecture of data lakes and consider the various solutions that the major cloud providers offer. We will also look at the challenges that come with collecting and analyzing vast amounts of data, including the phenomenon that is called data gravity, since it’s hard to transport these large amounts of data across different platforms. A solution to overcome this challenge is data mesh.

We will cover the following topics in this chapter:

* Choosing the right platform for data
* Building and sizing a data platform
* Designing for portability and interoperability
* Overcoming the challenges of data gravity
* Managing the foundation for data lakes

**Choosing the right platform for data**

It is a cliché, but nonetheless, it’s very true: data is the new gold. It is with good reason that in enterprise architecture frameworks, data is named as the first thing that a business must consider, analyzing what data it should use and how to gain optimal benefits from that data. No business can operate without data; it needs data to gain insights into markets and the demands of its customers. It needs data to drive the business.

You will find the term **data-driven** in almost every cloud assessment study. What does data-driven mean? A company makes decisions based on the analysis of data. Intuition or decision based on previous experience is ruled out. Every action is supported by the analysis of data.

To enable a data-driven business, we need one thing: the data itself, typically in vast amounts and preferably in (near) real time. The collection of data is prerequisite number one. Prerequisite number two is that this data must be accessible. So, we need accurate, relevant data that is available for data analytics. These are the key requirements for data and to become a data-driven organization.

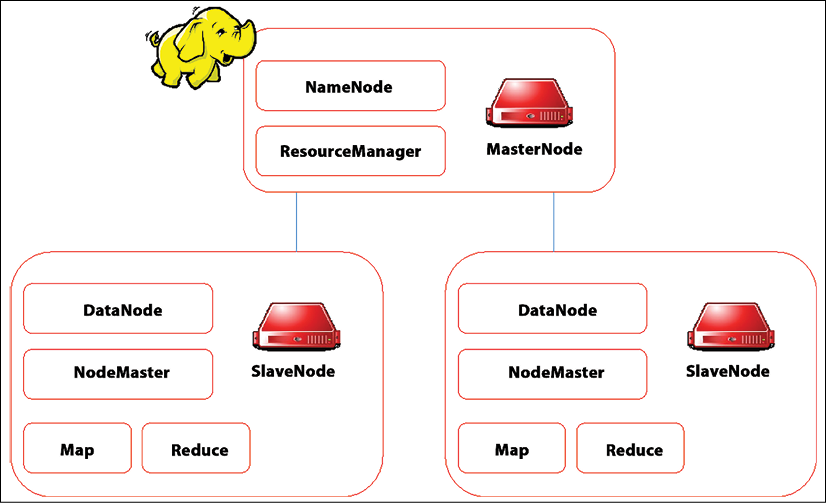
To enable capturing data, storing it, cleansing the data, and preparing it for analysis through data analytics, companies need data platforms. Unfortunately, there’s no magic formula or golden nugget that will get an instant data platform. Let’s first define what a platform is.

A data platform is first of all a central repository where a company captures data. Data that is scattered among a variety of databases and other sources is very hard to analyze. Hence, we try to collect all the relevant data in one repository. From that single repository, data analysts can start processing the data. In this process, logical collections are transformed into datasets, including the cleansing of data; next, the algorithms are defined to mine the data and produce valuable outcomes for the business.

Defining and designing a data platform that can do this must take architectural layers into account, regardless of the specific cloud platform that is used. These layers are:

* **Discovery**: Where are the data sources and is the data accessible?
* **Observability**: This is mainly about the quality of the data. Is the data recent and accurate? Has all the relevant data been collected?
* **Ingestion**: Moving data from one place to the other, for instance, raw data from the data lake to the data warehouse for further processing.
* **Storage**: The “physical” place where data is stored. Think of AWS S3 or Azure blob storage.
* **Modeling**: Building the data models.
* **Analytics**: The usage of cleaned data to run metrics against, aiming to get meaningful results out of data.

A data platform is often referred to as a data lake, which is nothing other than a massive storage location that holds raw data from multiple sources. This can be all sorts of data, from files to streaming data. Typically, these data lakes use object storage such as blob storage in Azure or S3 in AWS. Data lakes can be built on-premises, but more common is the use of public cloud providers where data lakes are configured in storage clusters. To distribute large datasets across these clusters, technology such as Apache Hadoop is used. A basic architecture of Hadoop, using HDFS, or the Hadoop Distributed File System, is shown in *Figure 8.1*. MapReduce is a technology that is used for applications to process vast amounts of data in parallel across nodes in the cluster.



*Figure 8.1: High-level architecture of Hadoop cluster*

What’s the difference between data lakes and data warehouses? A data lake holds raw, unstructured data, whereas warehouses offer structured data. Typically, datasets that are extracted from data lakes are inserted into warehouses for querying and analytics.

A combination of a data lake and a data warehouse is a **lakehouse**. With a lakehouse, additional, formatted structures are implemented on top of the data lake. This was initially pushed by Databricks, but since then, more technologies that provide lakehouse solutions have entered the market. Examples are Delta Lake and Apache Iceberg.

Now, let’s explore the various solutions that cloud providers offer for creating data platforms.

**Azure Data Lake and Data Factory**

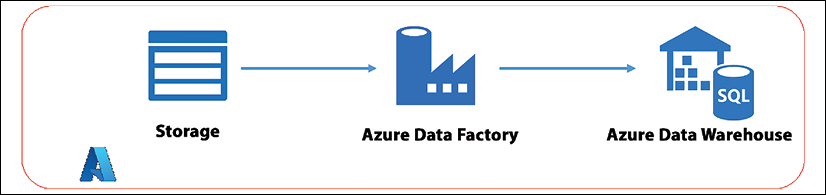
The full name for the service is Azure Data Lake Storage Gen2. The reason to have this very specific name is that the service is nothing less than a solution built on top of Azure storage. So, you can use the blob API or **Azure Data Lake Storage (ADLS)**. Keep in mind that not all features of blob storage are available in ADLS at this time.

Data Lake Gen2 adds file system semantics to blob storage so that petabytes of data can be organized in objects and files with a hierarchical structure of directories. This enables easier access to various types of data. But: it’s still raw data.

Azure also provides solutions to start assembling and analyzing datasets. This solution is **Data Factory**. With Data Factory, analysts can process data using **extract-transform-load** (**ETL**) or **extract-load-transform** (**ELT**) as a code-free service running in Azure. It allows for building data pipelines that cover the different stages of data processing:

* **Ingest**: Collecting datasets
* **Control and data flow**: Designing data workflows in pipelines
* **Schedule**: To run data processes at specified times or as a trigger when new data is ingested
* **Monitor**: Tracking the activities in the pipelines

A high-level overview of the Data Factory architecture is presented in *Figure 8.2*.



*Figure 8.2: High-level architecture of Azure Data Factory*

To explain it in very simple terms: Data Factory allows for connecting to data sources, collecting data, processing that data, and preparing it for analytics. The outcomes of analysis might be presented to tools such as Microsoft Power BI. **Azure Data Factory (ADF)** starts with ingesting, then preparing/transforming/analyzing, and next publishing the results to data stores, ready for other tools (such as Power BI) to consume them. ADF supports dozens of data stores (see <https://learn.microsoft.com/en-us/azure/data-factory/connector-overview>).

At Ignite 2022, Microsoft introduced Azure Data Explorer: a fully managed big data analytics platform, supporting the analysis of massive amounts of data in near real time. It also offers ingestion support for various data sources, including AWS S3 and OpenTelemetry Metrics, Logs, and Traces. It’s one development that shows how important the market for big data is to cloud providers.

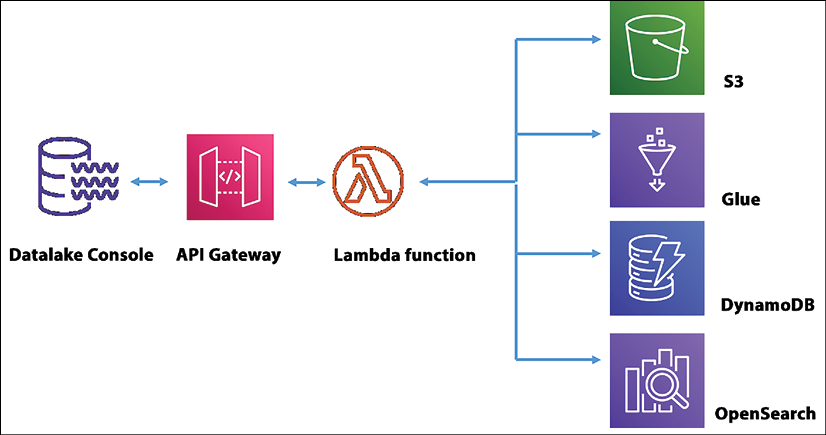
One other service that must be mentioned here is Azure Synapse, which connects enterprise data warehousing with big data analytics. Azure Synapse brings together SQL technologies used in enterprise data warehousing, Spark technologies used for big data, Data Explorer for log and time series analytics, pipelines for data integration and ETL/ELT, and deep integration with other Azure services such as Power BI, Cosmos DB, and Azure **ML (Machine Learning)**. For data governance, Microsoft offers Microsoft Purview, which provides a unified data governance service that helps you manage your on-premises, multi-cloud, and **software-as-a-service (SaaS)** data.

**AWS Data Lake and Redshift**

The AWS Data Lake solution consists of various components. The raw data is stored on top of S3 storage. Part of the solution is Amazon DynamoDB, a NoSQL database service that is fully managed by AWS and offers continuous backups and automated replication across regions to make the service resilient. Import and export tools are provided too with DynamoDB.

One final element in the AWS Data Lake proposition is AWS Glue. The name of the service has been chosen accurately since it really glues together the various components. AWS Glue performs the ETL process: discovering and preparing the datasets—or data catalog—into DynamoDB where the data can be analyzed. Both DynamoDB and Glue are serverless services. The analytics service is provided through OpenSearch, previously Elasticsearch. The Lambda serverless function is used as a message trigger to start the process.

*Figure 8.3* shows a high-level architecture.



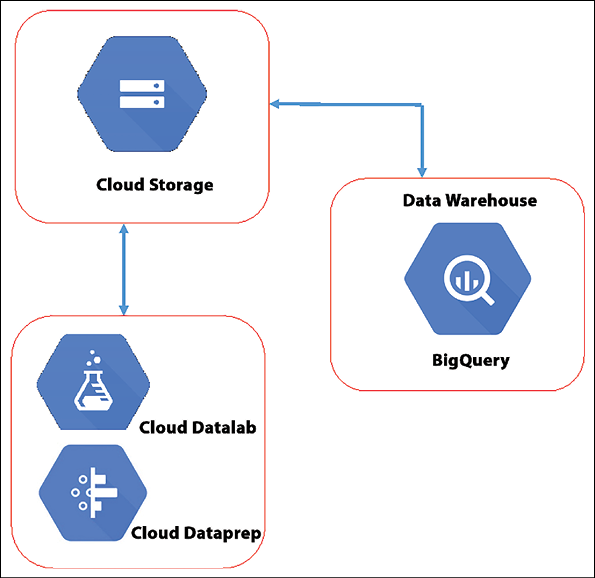
*Figure 8.3: High-level architecture of AWS Data Lake*

In the context of big data, a different AWS service is often mentioned: Amazon Redshift. This is a data warehouse based on PostgreSQL. Redshift allows for 16 petabytes of—structured—data on one cluster.

**Google’s data lake and BigLake**

The foundation of Google’s data lake is Google Cloud’s Cloud Storage, comparable to Azure blob and AWS S3. Cloud Storage holds objects and files up to 5 TB per item. Also comparable to other cloud providers is the fact that Cloud Storage comes in different classes with various service levels in terms of availability and performance, which is reflected in the pricing. Exploring raw data in Cloud Storage can be done with Cloud Datalab and Cloud Dataprep. These tools will allow you to view data and determine if data is relevant, for instance.

The offloading of data into a data warehouse is done through BigQuery, a fully managed serverless service that even includes built-in machine learning capabilities to perform real-time and predictive analysis. *Figure 8.4* shows a simple architecture for GCP’s data lake solution.



*Figure 8.4: High-level architecture of Google’s data lake solution*

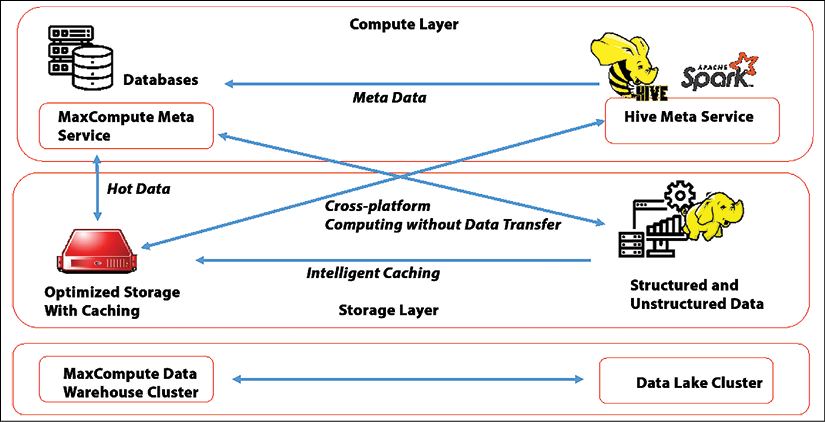
BigQuery Omni is a multi-cloud variant of this solution that also allows for data analytics using data that is stored in AWS and Azure.

BigLake, which was introduced in 2022, takes the idea of multi-cloud a bit further. BigLake provides a single-pane-of-glass view and a storage engine across various platforms, meaning that it “combines” data sitting in Cloud Storage, AWS S3, and Azure Data Lake Storage Gen2 as if it were one big lake. Data engineers would only have to work in one console. Next, BigLake offers fine-grained security controls.

**Alibaba Cloud Lakehouse**

A common way to start data engineering and data analytics is to offload data from a data lake to a data warehouse. The warehouse is separated from the lake, and tools such as Apache Hive are used to request and transport data between sources and processing platforms.

Alibaba Cloud Lakehouse takes a different approach, by integrating the warehouse with the lake, allowing data to flow between the two platforms. The data in the lake and the warehouse are seamlessly integrated, including the metadata. The data lake uses Alibaba’s cloud storage **Object Storage Service** (**OSS**), comparable to S3 or blob, but the warehouse is built on a solution called MaxCompute. This solution implements a unified storage access layer that supports HDFS, the file system of Hadoop, and OSS, all in read/write modus. *Figure 8.5* is a high-level presentation of the architecture.



*Figure 8.5: High-level architecture of Alibaba Cloud Lakehouse*

A good blog post on Alibaba’s solution can be found at <https://alibabatech.medium.com/data-lakes-or-data-warehouses-cd5122ba7634>.

MaxCompute still uses Apache Hive and Spark to map databases in the data lake to projects in MaxCompute, the data warehouse. So, although it’s presented as one system, the architecture still resembles the common architecture to process flows between lakes and warehouses.

**Oracle Big Data Service**

There’s one more solution that deserves mentioning. As a company that finds its origin in processing data with database solutions, it makes sense that Oracle has developed a solution for big data, accurately called **Oracle Big Data Service**.

The solution is built on top of **Oracle Cloud Infrastructure** (**OCI**) and contains the deployment of fully managed Hadoop clusters to form a data lake. This is Oracle’s own version of Hadoop, called **Oracle Distribution of Apache Hadoop** (**ODH**).

It’s a complete service to get customers up and running fast, including a toolbox with various ETL and analytics tools such as the analytics engine Spark, the non-relational database HBase, the data warehouse software Hive, and the configuration tool ZooKeeper.

**Building and sizing a data platform**

As with every service that we deploy in the cloud, we need a foundation to build the platform on. Hence, building a landing zone that can hold raw data is the first step. This landing zone should be an environment that serves only one purpose: to capture raw data. It’s recommended to build this landing zone separate from core IT systems. It should be scalable but low cost, since it will hold a lot of data. The issue with keeping data is that it might increase the cloud bill exponentially. Data storage comes at a very low price per unit of data, but the catch is that we need a lot of these small units.

It is important to implement governance from the start. This includes defining and implementing guardrails for the classification of data and tagging.

Once the landing zone has been established, data analysts can start using the data lake as a sandbox environment. This is the second stage. Analysts can start building prototypes of data models and work with the raw data that is collected in the data lake. They can also test various tools to find what will work best and give the most benefit to the business.

There will be a moment when datasets have been defined and tools selected. This is the time to start integrating the datasets with other business data. This is the process of **ETL** or **ELT**: **extract-transform-load** or **extract-load-transform**. From the raw data in the data lake, the required datasets are collected, extracted, and loaded in the enterprise data warehouses. Since only relevant data is extracted, the sizing of the data warehouse doesn’t need to be increased. A lot of data will remain in the data lake, built on cheaper storage. Tools such as the Data Factory and OpenSearch will help in optimizing queries.

Now, the data lake is a core component of the IT infrastructure and the business of an enterprise. Strong governance is an absolute requirement in operating the data lake and the data flows that connect the data lake with data warehouses and various data models. These models will generate output to applications and enable detailed insights into the business itself, the efficiency and performance of business-supporting operations, and the markets the business operates in.

We have now defined the four stages of building and sizing the data lake. It’s summarized in *Figure 8.6*.



*Figure 8.6: Four stages of building and managing data platforms*

There are some best practices to keep in mind in building and sizing a data platform. Number one to keep in mind is that you have to understand the business in order to collect the right, relevant data. There must be a clear objective for implementing a data lake: what business problem are we solving by querying vast amounts of data? And what sort of data would we really need as a business?

Second: know what the data is about. Data must be recognized and that can be achieved by tagging the data. Hence, metadata is crucial. There’s always the risk of a data swamp, that is, companies and their employees drowning in data. One way to solve this is data cleansing, which is as important a process as ingesting, scheduling, and monitoring. Depending on the data classification, data must be cleaned. This is also important in terms of controlling costs.

Lastly: security must be priority number one. Keep data safe by implementing authentication, authorization, and encryption, both for data in transit and for data at rest. Remember what we said earlier in this chapter: data is the new gold.

**Designing for interoperability and portability**

Portability and interoperability should be driven by use and business cases—not purely for the sake of portability or interoperability. In IT systems, there are four levels that define the portability of systems: data, applications, platforms, and infrastructure, following the **Architecture Development Method** (**ADM**) of TOGAF.

* Data represents information in such a form that it can be processed by computers. Data is stored in storage that is accessible to computers.
* An application is software that performs actions that are triggered by business requests.
* Platforms support applications.
* Infrastructure is a collection of computation, storage, and network resources. Computation can also refer to cloud computing including VMs, containers, and serverless functions.

One important note that we have to make at this point is that cloud computing causes a “blurring” effect in the demarcation of infrastructure, platforms, and applications. Think of PaaS and SaaS, where PaaS includes the infrastructure layer and SaaS is a fully integrated stack of application software, the platform, and the underlying infrastructure. In PaaS and SaaS, the demarcation of the various architecture layers is not as defined anymore as in traditional IT architecture.

However, the aim of TOGAF in achieving portability is to create abstract architecture layers in IT systems. Thus: data is separated from the applications and applications are separated from the technology. Portability is a requirement in itself but subsequently brings requirements along which systems should adhere to. To summarize:

* The architecture vision includes the portability of applications and interoperability.
* Requirements are collected to fulfill the vision.
* Business architecture defines the business processes where portability and interoperability are a requirement.
* Information systems architecture defines how portability and interoperability are achieved on application and data levels.
* Technology architecture defines how technology is supporting portability and interoperability.

In theory, this would allow for data and application architecture that is not dependent on specific technology layers. This is also the starting point for microservices; microservices-based applications are built as a collection of highly decoupled services that handle a single action. Each service is independently built, deployed, and monitored. The use of microservices architecture is important to create portability.

Portability is enabled through decoupling: applications are decoupled from data, data is decoupled from storage, and application code is decoupled from the underlying infrastructure including networks. Interoperability is enabled through standardization, preferably with open standards.

Portability is defined by abstraction between the basic infrastructure (often referred to as the landing zone; please refer to *Chapter 6*, *Controlling the Foundation Using Well-Architected Frameworks*), the configurations of the infrastructure components, the data, and the application. Again, we’re using the definitions by The Open Group as the industry standard. The definitions for portability are provided below:

* **Data portability** is essentially about reusing data across various applications. Data portability is likely the most difficult component in achieving portability. The structure of the data is often designed to fit a particular form of application processing, and a significant transformation is needed to produce data that can be handled by a different product. This is separate from the data carrier, the database technology, or the storage layer hosting the database. For example, PostgreSQL as a database will run on different platforms. That doesn’t mean that the data in the database can be used by different applications without transforming this data.
* Application portability is about reusing application components across various computing platforms. These can be traditional on-premises platforms, but also cloud platforms including PaaS. Examples of the latter are managed database services such as **Relational Database Service (RDS)** in AWS and Azure SQL Database. Portability requires a standard interface exposed by the supporting platform. This must enable the application to use the service discovery and communication between the platforms, as well as providing access to the platform capabilities that support the application directly. This same principle applies to interoperability.
* For platform portability, we can think of reusing platform components across clouds and on-premises infrastructure. Kubernetes as an underlying platform for containers hosting applications is an example. Kubernetes can be deployed on various platforms, including public cloud infrastructure and on-premises machines that sit in privately owned datacenters. But virtualization is also an example: virtual machines can be transferred between different platforms if the machine images are portable. Note: this is different from managing a variety of machines from one console. For example, Azure Arc allows administrators to bring non-Azure machines and Kubernetes from other clouds and on-premises under the control of Azure and manage these machines from the Azure console. The non-Azure machines, however, stay where they are; the images of these machines are not transferred to other instances. This technology leverages the possibilities for interoperability between platforms, but it doesn’t make environments portable.

It’s important to realize that cloud (native) services do not necessarily contribute to portability and interoperability.

Portability is one of an application’s non-functional requirements. An application should be as portable as possible, not tied to a specific infrastructure or platform. We can achieve this through the abstraction of layers and decoupling services. To summarize, if we approach portability from the perspective of defining, designing, and applying microservices, we can obtain the highest level of portability.

Cloud portability is the ability to move applications and data from one cloud to another with minimal disruption. Cloud portability enables the migration of cloud services from one cloud provider to another or between a public cloud and a private cloud. If we’re designing and implementing data and applications to be compliant with cloud portability, then we are really following a multi-cloud strategy. A multi-cloud strategy could be having the data lake in one cloud and having apps (with their data) in other clouds. But architects must implement the principle of decoupling into their designs from the start.

Decoupling services come with challenges, especially in refactoring existing landscapes and applications. The biggest challenge: interoperability. But when do we speak about interoperability? Products, systems, or organizations are interoperable if they can work together without restrictions. Services can discover each other, connect, and communicate in a coordinated way, without interference from the end user. If we decouple systems, applications, and data in a microservices architecture, this becomes a challenge. We will have to make sure that all services can find each other (discoverability) and know how they can communicate with each other.

Best practices to achieve interoperability in multi-cloud architectures include:

* Homogenous virtualization technology, including container orchestration platforms
* Standardized protocols for authorization and authentication
* Use of standard **Application Programming Interfaces** (**APIs**)

Data synchronization is a particular issue when components in different clouds or internal resources work together, whether or not they share the same protocols or even when they are absolutely identical. Copies of the same data are kept in different systems and the challenge is to keep this data synchronized, and with that, consistent. If data is stored in different clouds, this can become a challenge. Data must first of all be discoverable and accessible, but to keep it consistently in sync, it also requires high-speed connectivity to overcome issues in latency. Items that we must consider in synchronization of data are:

* Management of the master data sources: where is the single source of truth?
* Management of data at rest and data in transit across platforms
* Data visibility
* Access management, authentication, and authorization

Full interoperability includes the continuous, dynamic discovery of infrastructure, data sources, and application components, communicating with other components, at run time.

**Overcoming the challenges of data gravity**

Applications don’t just hold data; they also produce a lot of data that they share with other applications. Data will attract new data and services in other applications. As data accumulates, more and more applications and services will use it. Data and applications are attracted to each other, as in the law of gravity. To keep it short and simple: the amounts of data will grow, either autonomously or, more likely, because data sources will be connected to other data sources.

In addition to the strategic advantage of having access to this data, this also presents a major challenge. Databases are becoming so large that it becomes almost impossible to move the data. This can lead to a situation where companies are tied to a certain location or provider to hold that data. In addition, companies that use each other’s data and services must stay close to each other in order to provide good service. By keeping data physically close together, it can be exchanged quickly without end users experiencing slow processes—the effect that we call **latency**.

Data gravity forces companies and their architects to consider a variety of topics, data privacy being one of them. This will increasingly become an issue since data will need to be accessible across different platforms and applications. At the time of writing, there’s a debate going on in several Western-European countries where universities, healthcare institutions, and governmental bodies are storing data in public clouds that are US-owned and managed. One of the main arguments: once you have the data in one of these clouds, you will never get it out of it. And: the US government might force American companies to grant insights into the data that they store, despite agreements between the US and the European Union.

The challenge of data gravity forces companies to think about where they keep their data, how they connect to other data sources, and what data they can use and/or share. Successful business processes must be able to bring data together and bring the user and the applications to the data. Since enormous amounts of data are hard to move, we will have to architect solutions where we use the data where it is, acknowledging that data is decentralized by default. If we take that as a principle, then there’s no need to collect every single piece of data in a central data store.

Again, this will inevitably lead to debates about data privacy. But if we want to create global solutions for the major challenges of our time—think of climate change and the accessibility of healthcare around the world—then we need data. We can try to transfer all this data to one repository, or design applications in such a way that we can use this data in a safe way, wherever the data is.

As a result of technology such as the cloud, IoT, and analytics, data is created everywhere. It is produced in smartphones, buildings, homes, and even across entire cities. Companies and other institutions must be prepared for this. A decentralized infrastructure can address the challenges of data gravity by providing the right coverage, capacity, and connectivity, quickly bringing users, networks, and clouds to that data. Cloud providers are already prepared for this by implementing global coverage of their services. With edge computing and 5G, we will also have the technology to access this data swiftly.

**Introducing the principles of data mesh**

One solution that we must mention here is the principle of data mesh, an architecture principle that was introduced by Zamak Deghani.

The paper about the data mesh architecture can be found at <https://martinfowler.com/articles/data-monolith-to-mesh.html>.

Data mesh is an architecture that allows applications, and with that the user, to access data without transporting this data first to a data lake. Data mesh utilizes the principles of decentralized data sources. The challenge in data mesh is still how to get the right data to the application where it can be processed, without moving that data. This can be done through data streaming.

Oracle provides a managed data mesh service called **Oracle Cloud Infrastructure** (**OCI**) GoldenGate. GoldenGate connects to data sources, replicates the data, cleans the data, and delivers this curated data through streaming data pipelines to data consumers. GoldenGate responds to triggers using Apache Kafka: as soon as events occur that call for action in the data chain, GoldenGate will start replicating and streaming relevant raw data in real time, different from traditional ETL tools that transport data in batches.

In short, GoldenGate is the bridge between data producers and consumers. A bridge enables a continuous stream of traffic, in this case, data traffic.

The major cloud providers all provide their own solutions for data mesh. Azure Synapse Analytics also utilizes Kafka or Event Hub to capture data events and next collects and publishes data to data products. A similar solution can be built with AWS Lake Formation and AWS Glue, providing solutions for data discovery, analytics, and publishing data. The data mesh solution that Google offers is Dataplex, which became generally available in early 2022.

**Managing the foundation for data lakes**

Data engineers design, build, and manage the data pipelines, but the foundation of the data lake and data warehouse is the specific landing zone for the data platform. Typically, landing zones in the cloud are operated by cloud engineers who take care of the compute, storage, and network resources.

Looking at the management of data platforms, we can distinguish various roles:

* Data architect or engineer: The architect and data engineer are often combined in one role. This role is responsible for the design, development, and deployment of the data pipelines. The engineer must have extensive knowledge of ETL or ELT principles and technologies, making sure that data from sources gets collected and transformed into usable datasets in data warehouses or other data products where the data can be further analyzed. Data also needs to be validated, which is a required skill of the engineer too. In essence, the engineer makes sure that the data that is ingested into warehouses is of good quality and usable for analytics.
* Business and data analyst: The main question that a business or data analyst must answer is, what data can be used for the business? What data provides the required insights for the business? The business ambitions, goals, and targets are translated into metrics through which data is analyzed.
* Data scientist: Where the business or data analyst defines the metrics for the data, the data scientist makes sure that the right data sources are discovered, and the appropriate data is made available to the analysts. The data scientist is crucial in finding the right data that the analysts can use to proof the metrics and provide insights that enable companies to fulfill their business strategy.
* Cloud operator: Data must be hosted somewhere, on big machines running in the cloud, for instance. These machines holding all the data must be accessible, performant, and secured from attacks. The design of the landing zone, the operations of the hosting instances, the network configuration, and security guardrails are the responsibility of cloud operators, who keep the data platform running in an optimal and highly secure way.

There’s one aspect that we have not mentioned here so far, but that does play a significant role in the management of data platforms: cost management. Big data can cost a company big money, hence implementing methods and tools to control the costs of data platforms has become an important task of the architect and the operators.

FinOps helps organizations to first understand what these costs are, how they can control these costs, and how they forecast budgets in the usage and operations of data lakes and warehouses. *Part 3*, *Controlling Costs in Multi-Cloud Using FinOps*, of this book is all about the FinOps principles.

**Summary**

In this chapter, we discussed the basic architecture principles of building and managing a data platform. We looked at data lakes that can hold vast amounts of raw data and how we can build these lakes on top of cloud storage. The next step is to fetch the right data that is usable in data models. We must extract, transfer, and load—ETL for short—the datasets into environments where data analysts can work with this data. Typically, data warehouses are used for this.

We studied the various propositions for data operations of the major cloud providers, AWS, Azure, Google Cloud, Alibaba, and Oracle. Next, we discussed the challenges that come with building and operating data platforms. There will be challenges with respect to access to data, accuracy, as well as privacy and compliance. Data gravity is another problem that we must solve. It’s not easy to move huge amounts of data across platforms, hence we must find other solutions to work with data on different platforms. Designing for interoperability and portability is therefore a key capability of architects.

There’s one question that we didn’t answer in this chapter: where does all this data come from? It’s largely coming from a growing number of devices that are connected to the internet: the **Internet of Things (IoT)**. That’s the topic of the next chapter.

**Questions**

1. What does the term ETL mean?
2. What would be the first step in building a data platform?
3. True or false: Data lakes are typically built on the common storage layers of major cloud providers such as Azure blob storage and Amazon S3.
4. What does Oracle’s GoldenGate do?

**Further reading**

* *Data Lake for Enterprises*, by Tomcy John and Pankaj Misra, Packt Publishing