Chapter 4
Designing a Data Processing Solution

Google Cloud Professional Data Engineer Exam objectives covered in this chapter include the following:

1. **Designing data processing systems**

1. **✓ 1.3 Designing a data processing solution. Considerations include:**

1. Choice of infrastructure
2. System availability and fault tolerance
3. Use of distributed systems
4. Capacity planning
5. Hybrid cloud and edge computing
6. Architecture options (e.g., message brokers, message queues, middleware, service-oriented architecture, serverless functions)
7. At least once, in-order, and exactly once, etc., event processing

2. **✓ 1.4 Migrating data warehousing and data processing. Considerations include:**

1. Awareness of current state and how to migrate a design to a future state
2. Migrating from on-premises to cloud (Data Transfer Service, Transfer Appliance, Cloud Networking)
3. Validating a migration

Data engineers need to understand how to design processing solutions that start with data collection and end with data exploration and visualization. In this chapter, you will learn about designing infrastructure for data engineering and machine learning, including how to do the following:

- Choose an appropriate compute service for your use case
- Design for scalability, reliability, availability, and maintainability
- Use hybrid and edge computing architecture patterns
- Design distributed processing systems and use appropriate event processing models
- Migrate a data warehouse from on-premises data centers to GCP

The discussion in this chapter further highlights the need to employ multiple GCP services together to build a comprehensive solution for common enterprise use cases.

Designing Infrastructure

Data engineers are expected to understand how to choose infrastructure appropriate for a use case; how to design for scalability, reliability, availability, and maintainability; and how to incorporate hybrid and edge computing capabilities into a design.

CHOOSING INFRASTRUCTURE

GCP provides a range of compute infrastructure options. The best choice for your data engineering needs may depend on several factors. The four key compute options with which you should be familiar are as follows:

- Compute Engine

- Kubernetes Engine
- App Engine
- Cloud Functions

Newer services, such as Cloud Run and Anthos, are also available for use, but they are currently not included in the Professional Data Engineer exam and so will not be covered here.

Compute Engine

Compute Engine is GCP's infrastructure-as-a-service (IaaS) product. With Compute Engine, you have the greatest amount of control over your infrastructure relative to the other GCP compute services.

Compute Engine provides virtual machine (VM) instances, and users have full access to the VM's operating system. Users can choose from a large number of operating systems that are available on GCP. Once an instance is created, users are free to install and configure additional software to meet their needs.

Users also configure the machine type either by choosing a predefined machine type or by configuring a custom machine type. Machine types vary by the number of vCPUs and the amount of memory provided. Instances can be configured with more security features, such as Shielded VMs and accelerators, such as GPUs, which are often used with machine learning and other compute-intensive applications.

In addition to specifying the machine type, operating system, and optional features, you will specify a region and zone when creating a VM.

VMs can be grouped together into clusters for high availability and scalability. A managed instance group is a set of VMs with identical configurations that are managed as a single unit. Managed instance groups are configured with a minimum and a maximum number of instances. The number of instances in the group will vary to scale up or down with workload.

Compute Engine is a good option when you need maximum control over the configuration of VMs and are willing to manage instances.

Kubernetes Engine

Kubernetes is a container orchestration system, and *Kubernetes Engine* is a managed Kubernetes service. Kubernetes is an open source platform developed by Google and now widely used for deploying containerized applications. Kubernetes is deployed on a cluster of servers. You could run your own Kubernetes cluster by deploying it to VMs in Compute Engine, but then you would have to take on the responsibility for operating the cluster. With Kubernetes Engine, Google maintains the cluster and assumes responsibility for installing and configuring the Kubernetes platform on the cluster. Kubernetes Engine deploys Kubernetes on managed instance groups.

One of the advantages of Kubernetes is that users can precisely tune the allocation of cluster resources to each container. This is especially useful when applications are designed as a set of microservices. For example, a microservice might need only a small fraction of a CPU. In that case, a container running that microservice can be allocated only the amount of CPU needed. This allows for more efficient use of compute resources.

Alternatively, if the microservice were deployed on a VM with a full vCPU, the vCPU would be underutilized. This leaves developers with the choice of either tolerating inefficient utilization or running additional microservices on the same virtual machine. This approach has its own disadvantages. Unless all microservices have the same lifecycle, you may find that performing maintenance on one microservice can disrupt the operations of the other microservices.

By separating application components into microservices and running those microservices in their own containers, you can improve resource utilization and possibly make the application easier to maintain and update.

Another advantage of Kubernetes is that it can run in multiple environments, including other cloud providers and in on-premises data centers. Kubernetes can detect when containers within pods are not functioning correctly and replace them. Kubernetes can also scale up and down the number of pods to accommodate changing workloads.

Perhaps the most significant constraint of Kubernetes is that applications must be containerized to run on Kubernetes. If existing applications running on premises in VMs are migrating to the cloud, a lift-and-shift migration, in which the system architecture is not changed, to Compute Engine is likely the fastest way to get to the cloud. Once there, you can containerize your application.

> NOTE Anthos Migrate is a GCP service for migrating workloads configured to run on physical machines and virtual machines to containers. The service is currently in beta. For more information, see `https://cloud.google.com/migrate/anthos/`.

App Engine

App Engine is GCP's original platform-as-a-service (PaaS) offering. App Engine is designed to allow developers to focus on application development while minimizing their need to support the infrastructure that runs their applications. App Engine has two versions: App Engine Standard and App Engine Flexible.

App Engine Standard is a platform for running applications in a language-specific serverless environment. Currently, App Engine Standard supports Go, Java, PHP, Node.js, and Python, but the set of supported languages may have changed by the time you read this. App Engine Standard is available in two forms: first generation and second generation. Second-generation services offer more memory and more runtime options.

Each application running in App Engine Standard has an instance class that determines the amount of available CPU and memory. Resources in instance classes currently run from 256 MB of memory and a 600 MHz CPU up to 2048 MB of memory and a 4.8 GHz CPU.

App Engine Flexible runs Docker containers. This allows developers to customize their runtime environments in ways not available in App Engine Standard. For example, developers can start with custom Docker images, install additional libraries and packages, and then deploy container instances based on that image. App Engine Flexible also provides health checks and automatically patches the underlying operating system.

App Engine Standard is a good option when your application is developed in one of the supported languages and needs to scale up and down based on workload. App Engine Flexible is a good option when you want to have the advantages of a PaaS as well as the ability to customize your runtime environment.

Cloud Functions

Cloud Functions is a serverless, managed compute service for running code in response to events that occur in the cloud. Events such as writing a message to a Cloud Pub/Sub topic or uploading a file to Cloud Storage can trigger the execution of a Cloud Function. Cloud Functions also respond to events in HTTP, Firebase, and Stackdriver Logging.

For each supported service, Cloud Functions respond to specific events. In Cloud Storage, a script can execute in response to uploading, deleting, and archiving files. HTTP events include `GET`, `POST`, `PUT`, `DELETE`, and `OPTIONS` operations.

Cloud Functions is written using JavaScript, Python 3, and Go.

Cloud Functions is a good option when you need to execute code in response to events that can occur at any time, such as uploading a file or calling a webhook. They are also useful when ingesting data using Cloud Pub/Sub, such as in an IoT ingestion pipeline.

The four main compute infrastructure options are summarized in Figure 4.1.

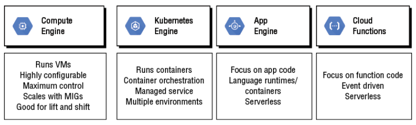

Figure 4.1 Summary of compute option features

AVAILABILITY, RELIABILITY, AND SCALABILITY OF INFRASTRUCTURE

When we design systems, we need to consider three nonfunctional requirements: availability, reliability, and scalability.

Availability is defined as the ability of a user to access a resource at a specific time. Availability is usually measured as the percentage of time that a system is operational.

Availability is a function of *reliability*, which is defined as the probability that a system will meet service-level objectives for some duration of time. Reliability is often measured as the mean time between failures.

Scalability is the ability of a system to handle increases in workload by adding resources to the system as needed. This implies that as workload increases, resources will be available to process that workload. It also implies that as workload decreases, the amount of allocated resources will decrease to a level sufficient to meet the workload plus some marginal extra capacity.

> Monitoring and logging are also essential practices to ensure infrastructure availability and reliability. Those topics will be discussed in detail in Chapter 5, "Building and Operationalizing Processing Infrastructure."

Making Compute Resources Available, Reliable, and Scalable

Highly available and scalable compute resources typically employ clusters of machines or virtual machines with load balancers and autoscalers to distribute workload and adjust the size of the cluster to meet demand.

Compute Engine

When using Compute Engine, you are responsible for ensuring high availability and scalability. This is done with managed instance groups (MIGs). MIGs are defined using a template. Templates include specifications for the machine type, boot disk image or container image, labels, and other instance properties. All members of a MIG are identical.

When an instance in a MIG fails, it is replaced with an identically configured VM. In addition, there may be times when an instance is running but the application is not functioning correctly—for example, if a fatal error occurs. In those cases, application-specific health checks can be used to detect the problem and replace the instance.

Load balancers direct traffic only to responsive instances, and they use health checks to determine which instances are available to accept traffic. Load balancers are either global or regional, and some are designed for internal traffic only whereas others work with external traffic as well.

The global load balancers are HTTP(S) Load Balancing, SSL Proxy, and TCP Proxy. The regional load balancers are Network TCP/UDP, Internal TCP/UDP, and Internal HTTP(S). Global load balancers can be used to distribute workloads across regions, further increasing the availability and reliability of applications.

Instance groups can be either zonal or regional. In zonal instance groups, all instances are created in the same zone. Regional instance groups place instances in multiple zones in the same region. The latter provides higher availability because the MIG could withstand a zone-level failure and the applications would continue to be available.

Autoscalers add and remove instances according to workload. When using autoscaling, you create a policy that specifies the criteria for adjusting the size of the group. Criteria include CPU utilization and other metrics collected by Stackdriver, load-balancing capacity, and the number of messages in a queue.

Note that when you use Compute Engine, you have the greatest level of control over your instances, but you are also responsible for configuring managed instance groups, load balancers, and autoscalers.

Kubernetes Engine

Kubernetes is designed to support high availability, reliability, and scalability for containers and applications running in a cluster. Kubernetes deploys containers in an abstraction known as a *pod*. When pods fail, they are replaced much like failed instances in a managed instance group. Nodes in Kubernetes Engine belong to a pool, and with the autorepair feature turned on, failed nodes will be reprovisioned automatically.

There are other ways Kubernetes Engine ensures high availability and reliability. When using Kubernetes Engine, you can specify whether the endpoint for accessing the cluster is zonal or regional. In the latter case, you can access the cluster even if there is a failure in a zone. Also, you can specify a high availability cluster configuration that replicates master and worker nodes across multiple zones.

App Engine and Cloud Functions

A key advantage of serverless, managed services like App Engine and Cloud Functions is that they are designed to be highly available, scalable, and reliable.

In App Engine, when the scheduler has a request, it can send it to an existing instance, add it to a queue, or start another instance, depending on how you have configured App Engine. With App Engine, you do have the option of configuring policies for scaling. This is done by specifying values for target CPU utilization, target throughput utilization, and maximum concurrent requests.

Cloud Functions are designed so that each instance of a cloud function handles one request at a time. If there is a spike in workload, additional function instances can be created. You have some control over this with the ability to set a maximum number of concurrently running instances.

Making Storage Resources Available, Reliable, and Scalable

GCP provides a range of storage systems, from in-memory caches to archival storage. Here are some examples.

Memorystore is an in-memory Redis cache. Standard Tier is automatically configured to maintain a replica in a different zone. The replica is used only for high availability, not scalability. The replica is used only when Redis detects a failure and triggers a failover to the replica.

Persistent disks are used with Compute Engine and Kubernetes Engine to provide network-based disk storage to VMs and containers. Persistent disks have built-in redundancy for high availability and reliability. Also, users can create snapshots of disks and store them in Cloud Storage for additional risk mitigation.

Cloud SQL is a managed relational database that can operate in high-availability mode by maintaining a primary instance in one zone and a standby instance in another zone within the same region. Synchronous replication keeps the data up to date in both instances. If you require multi-regional redundancy in your relational database, you should consider Cloud Spanner.

Cloud Storage stores replicas of objects within a region when using standard storage and across regions when using multi-regional storage.

Making Network Resources Available, Reliable, and Scalable

Networking resources requires advanced planning for availability, reliability, and scalability.

You have the option of using Standard Tier or Premium Tier networking. Standard Tier uses the public Internet network to transfer data between Google data centers, whereas Premium Tier routes traffic only over Google's global network. When using the Standard Tier, your data is subject to the reliability of the public Internet.

Network interconnects between on-premises data centers and Google Cloud are not rapidly scaled up or down. At the low end of the bandwidth spectrum, VPNs are used when up to 3 Gbps is sufficient. It is common practice to use two VPNs to connect an enterprise data center to the GCP for redundancy. HA VPN is an option for high-availability VPNs that uses two IP addresses and provides a 99.99 percent service availability, in contrast to the standard VPN, which has a 99.9 percent service level agreement.

For high-throughput use cases, enterprises can use Cloud Interconnect. *Cloud Interconnect* is available as a dedicated interconnect in which an enterprise directly connects to a Google endpoint and traffic flows directly between the two networks. The other option is to use a partner interconnect, in which case data flows through a third-party network but not over the Internet. Architects may choose Cloud Interconnect for better security, higher speed, and entry into protected networks. In this case, availability, reliability, and scalability are all addressed by redundancy in network infrastructure.

Another infrastructure consideration for data engineers is hybrid cloud computing.

HYBRID CLOUD AND EDGE COMPUTING

There are some enterprise data engineering use cases that leverage both cloud and non-cloud compute and storage resources. The analytics hybrid cloud combines on-premises transaction processing systems with cloud-based analytics platforms. In situations where connectivity may not be reliable or sufficient, a combination of cloud and edge-based computing may be used.

Analytics Hybrid Cloud

The *analytics hybrid cloud* is used when transaction processing systems continue to run on premises and data is extracted and transferred to the cloud for analytic processing. This division of computational labor works well because transaction processing systems often have predictable workloads with little variance. Analytic workloads are predictable but can be highly variable, with little or no load at given times and bursts of high demand at other times. The latter is well suited to the cloud's ability to deliver on-demand, scalable compute resources.

GCP has services that support the full lifecycle of analytics processing, including the following:

- Cloud Storage for batch storage of extracted data
- Cloud Pub/Sub for streaming data ingestion
- Cloud Dataflow and Cloud Dataproc for transformation and some analysis
- Cloud BigQuery for SQL querying and analytic operations
- Cloud Bigtable for storage of large volumes of data used for machine learning and other analytic processing
- Cloud Datalab and Cloud Data Studio for interactive analysis and visualization

Cloud Dataproc is particularly useful when migrating Hadoop and Spark jobs from an on-premises cluster to GCP. When migrating data that had been stored in an HDFS filesystem on premises, it is best to move that data to Cloud Storage. From there, Cloud Dataproc can access the data as if it were in an HDFS filesystem.

Consider how to populate a data warehouse or data lake initially in GCP. If you are migrating a large on-premises data warehouse or data lake, you may need to use the Cloud Transfer Appliance; smaller data warehouses and data marts can be migrated over the network if there is sufficient bandwidth. For additional details on how long it takes to transfer different volumes of data, see Google's helpful matrix on transfer volumes, network throughputs, and time required to transfer data at `https://cloud.google.com/products/data-transfer/`.

Edge Cloud

A variation of hybrid clouds is an *edge cloud*, which uses local computation resources in addition to cloud platforms (see Figure 4.2). This architecture pattern is used when a network may not be reliable or have sufficient bandwidth to transfer data to the cloud. It is also used when low-latency processing is required.

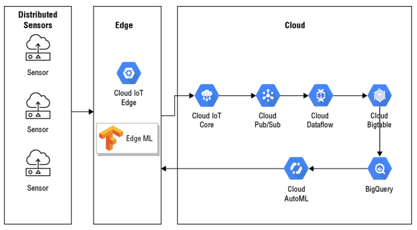

Figure 4.2 Edge computing brings some computation outside the cloud and closer to where the results of computation are applied.

For example, an IoT system monitoring a manufacturing device may employ machine learning models to detect anomalies in the device's functioning. If an anomaly is detected, the IoT sensor must decide quickly if the machine should be shut down. Rather than sending data to the machine learning model running in the cloud, the model could be run near the monitored machine using an Edge TPU, which is a tensor processing unit (TPU) designed for inference at the edge. (For details, see `https://cloud.google.com/edge-tpu/`.)

When using an edge cloud, you will need to have a continuous integration/continuous deployment (CI/CD) process to ensure consistency across edge devices. When full applications are run at the edge, consider using containers. This approach will help with consistency as well.

Designing infrastructure for data engineering services requires that you understand compute service options in GCP; know how to design for availability, reliability, and scalability; and know how to apply hybrid cloud design patterns as needed.

Designing for Distributed Processing

Distributed processing presents challenges not found when processing is performed on a single server. For starters, you need mechanisms for sharing data across servers. These include message brokers and message queues, collectively known as *middleware*. There is more than one way to do distributed processing. Some common architecture patterns are service-oriented architectures, microservices, and serverless functions. Distributed systems also have to contend with the possibility of duplicated processing and data arriving out of order. Depending on requirements, distributed processing can use different event processing models for handling duplicated and out-of-order processing.

DISTRIBUTED PROCESSING: MESSAGING

One way to categorize components of a distributed processing system is as components that move data from one process to another and components that transform or store data. Message brokers and message queues fall into the former category; middleware is the generic name for the latter category.

Message Brokers

Message brokers are services that provide three kinds of functionality: message validation, message transformation, and routing.

Message validation is the process of ensuring that messages received are correctly formatted. For example, a message may be specified in a Thrift or Protobuf format. Both Thrift and Protobuf, which is short for Protocol Buffers, are designed to make it easy to share data across applications and languages. For example, Java might store structured data types one way, whereas Python would store the same logical structure in a different way. Instead of sharing data using language-specific structures, software developers can map their data to a common format, a process known as *serialization*. A serialized message can then be placed on a message broker and routed to another service that can read the message without having to have information about the language or data structure used in the source system.

Message transformation is the process of mapping data to structures that can be used by other services. This is especially important when source and consumer services can change independently. For example, an accounting system may change the definition of a sales order. Other systems, like data warehouses, which use sales order data, would need to update ETL processes each time the source system changes unless the message broker between the accounting system and data warehouse implemented necessary transformations. The advantage of applying these transformations in the message broker is that other systems in addition to the data warehouse can use the transformed data without having to implement their own transformation.

Message brokers can receive a message and use data in the message to determine where the message should be sent. *Routing* is used when hub-and-spoke message brokers are used. With a *hub-and-spoke model*, messages are sent to a central processing node and from there routed to the correct destination node. See Figure 4.3 for an example of a hub-and-spoke model.

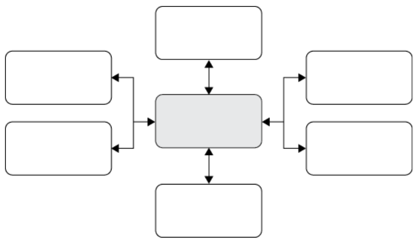

Figure 4.3 A hub-and-spoke message broker pattern

Message Queues

Message queues are used to enable asynchronous communications between processes. In the simplest case, one process writes a message to a message queue, and another process reads that message and removes it from the queue. In some cases, a message may be read by multiple processes. This kind of processing model is well suited to the publish and subscribe model.

In the *publish and subscribe model*, one or more processes write messages to a queue. For example, an IoT sensor could write a message to a message queue in the cloud. In addition, other sensors of the same type can write to the same queue. The data in the message queue may be consumed by one or more processes. For example, Sensor 1 may write to a message queue called Monitoring Queue, and Instance A of the processing service reads the message and processes the data. Next, Sensor 2 writes a message to Monitoring Queue, and that message is read by Instance B of the consuming process. See Figure 4.4 for an example of asynchronous message processing.

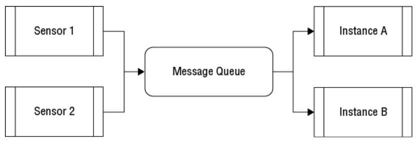

Figure 4.4 Simple asynchronous message processing

When the receiving processes are stateless—that is, they do not need to track information over time and across messages—then messages can be consumed by an instance of the receiving process. If the consumers need to maintain state information, such as maintaining the average value of the last 10 readings from a sensor, then all the data from a sensor must be processed by the same instance. The need to maintain state is one of the key considerations when you are designing message processing services.

Event Processing Models

Message queues can provide different guarantees with regard to how messages are provided to consuming processes. It is assumed that message queues are reliable and that a message that is written to a queue will be available to processes reading from the queue. Distributed systems also have to function under less than ideal conditions. For example, some messages may be delayed and arrive out of order. In other cases, a producing service may not receive an acknowledgment that a message was received, so that producing service sends the message again. If the acknowledgment was sent by a consuming service but not received by the producer, the message will be duplicated in the queue.

Here are some of the guarantees that message queues can provide:

At Least Once Delivery In this case, all messages are delivered at least once to a consuming service, though messages may be delivered more than once either to the same consuming service or to different consuming services.

At Least Once, in Order Delivery This is like at least once delivery, but it also adds a guarantee that messages will be ordered by time. This kind of guarantee can lead to some delay in processing because the message queue may buffer messages while waiting for late-arriving data to appear so that it can be sent in the correct order.

Exactly Once This model guarantees that a message is processed exactly once. This is an important guarantee if operations on messages are not idempotent. For example, if a message includes the number of products sold in an order and that number is used to decrement inventory levels, then duplicate messages can leave the inventory database in an incorrect state since the number of products sold will be decremented more than once.

Message queues and consuming processes must deal with late and missing data. In the case of late data, a message queue could be configured to drop data that is later than some threshold period. This is known as a *watermark*. Watermarks are widely used in stream processing to help bound the amount of time processes wait before performing operations, such as aggregating data for a particular time period.

DISTRIBUTED PROCESSING: SERVICES

Messaging components, such as message queues, are responsible for moving data between services. The services that operate on that data may be organized around one of a few service models. Three such models are service-oriented architectures (SOAs), microservices, and serverless functions.

Service-Oriented Architectures

A *service-oriented architecture (SOA)* is a distributed architecture that is driven by business operations and delivering business value. Typically, an SOA system serves a discrete business activity. SOAs are self-contained sets of services. Services are independent and do not need to know anything about other services except enough to pass messages. Some other characteristics of SOA systems are components are loosely coupled, long-lived, and autonomous. SOA systems have been designed using the Simple Object Access Protocol (SOAP) and Common Object Request Broker Architecture (CORBA), but Representational State Transfer (REST) is more widely used today.

Microservices

Microservices are a distributed systems architecture. Like other distributed systems, microservice architectures use multiple, independent components and common communication protocols to provide higher-level business services. Microservices are often used with Agile software development practices, including frequent deployments. Microservices are designed to implement a single function. This allows services to be updated independently of one another. Microservices can be readily deployed in containers as well. The combination of independencies and container-based deployments makes microservices well suited to teams and organizations that practice Agile development.

Serverless Functions

Serverless functions extend the principles of microservices by removing concerns for containers and managing runtime environments. With serverless functions, code that implements a single function can be deployed on a PaaS, such as Google Cloud Functions, without having to configure containers. Serverless functions can still use message queues and common messaging protocols so that they provide more support for DevOps teams who want to focus on coding application functionality more than systems administration.

Migrating a Data Warehouse

In the strictest sense, *data warehouses* are repositories of enterprise data organized for business intelligence reporting and analysis. For the purpose of our discussions, data warehouses include extraction, transformation, and load scripts; views and embedded user-defined functions; as well as reporting and visualization tools. Identity management information and access control policies used to protect the confidentiality, integrity, and availability of a data warehouse are also included for our purposes.

At a high level, the process of migrating a data warehouse involves four stages:

- Assessing the current state of the data warehouse
- Designing the future state
- Migrating data, jobs, and access controls to the cloud
- Validating the cloud data warehouse

All data warehouse migrations should follow this basic pattern, but it is important to note that there are different kinds of migrations.

An *off-loading data warehouse migration* involves copying data and schema from the on-premises data warehouse to the cloud to get a solution up and running in the cloud as fast as possible. This is a reasonable choice when the business needs the extra storage and compute capacity of the cloud, or if you want end users to have access to cloud-based reporting and analysis tools as fast as possible.

A *full data warehouse migration* includes all the steps in the scope of an off-loading migration plus moving data pipelines so that data can be loaded directly into the data warehouse from source systems. This approach allows you to take advantage of cloud tools for extraction, transformation, and load.

Both off-loading and full migration should follow the four-stage process outlined here.

ASSESSING THE CURRENT STATE OF A DATA WAREHOUSE

The initial assessment stage includes work to identify both technical requirements of existing use cases as well as detailing the business benefits of migrating from an on-premises solution to a cloud data warehouse.

Technical Requirements

One of the first things to do in a data warehouse migration is to identify existing use cases, including information about the data needed, such as its source, frequency of updating, ETL jobs used with the data, and access controls applied in the use case. Gathering technical requirements should include the following:

- A *list of data sources*, including metadata about data sources, such as the frequency at which the data source is updated
- A *data catalog*, which includes both attributes collected from source systems and derived attributes, such as summaries by time and location

- A *data model*, including schemas, tables, views, indexes, and stored procedures
- A *list of ETL scripts and their purposes*
- A *list of reports and visualization* generated from the data in the data warehouse
- A *list of roles and associated access controls*, including administrators, developers, and end users

These items should be fairly well defined. You should also include some less well-defined details about the existing data warehouse, such as limitations in the data or reporting tools that limit analysis; constraints, such as when data is available from sources and when it has to be available for querying from the data warehouse; and any unmet technical requirements, like sufficiently fine-grained adequate access controls.

Business Benefits

The business benefits of a data warehouse migration should be assessed in the early stages of a migration project. Business value can be derived from cost savings, reduction in backlog of ETL and reporting work, and increased agility and ability to deliver insights under changing conditions.

The total cost of ownership should be considered when assessing a data warehouse migration. This includes the cost of software licensing, hardware infrastructure, developer and system administrator's time, along with any third-party support or consulting costs.

If legacy software and limitations of an existing data warehouse implementation are leading to a backlog of reporting and data ETL work in an enterprise, that enterprise is incurring an opportunity cost. The cost of insights that are not available to business decision-makers are not readily quantifiable, but it is still a factor that should be considered along with other business benefit topics.

DESIGNING THE FUTURE STATE OF A DATA WAREHOUSE

Once it is determined that it makes business sense to migrate a data warehouse to GCP, it is time to plan the future state of the data warehouse. The use cases identified in the initial stages should provide the broad boundaries of what will be deployed in the cloud. You will also need to define the key performance indicators (KPIs) that are used to measure how well the migration process is meeting objectives. KPIs may include the amount of data migrated, the number of reports now available in the cloud warehouse, and the number of workloads completely migrated.

As part of designing for the future state, you should consider how you can take advantage of BigQuery features for data warehousing, including the following:

- There is no need to plan for storage and query resources since BigQuery is a serverless, managed service.
- BigQuery uses the columnar format, which supports nested and repeated fields, which can help denormalize data models, thus reducing the need for joins.
- Data is stored on the Colossus filesystem, which stores redundant blocks of data on multiple physical disks so that you do not have to manage redundant copies of data to ensure availability and durability.
- Federated storage is used to query data stored in Cloud Storage, Bigtable, or Google Drive.
- BigQuery maintains a seven-day history of changes so that you can query a point-in-time snapshot of data.

BigQuery has some differences from other data warehouse databases. In BigQuery, tables are grouped into datasets. Identity and access management (IAM)–based access controls are applied at the dataset level, not the table level. For this reason, if you need to grant someone permission to view data in one table in a dataset, they will have permission to view data in any table in that dataset. You can restrict access, however, by creating views over the data to which you would like users to have access. Those views can be placed in another dataset, and users can be granted roles that allow them to access data through the view. Users do not need access to the datasets that provide the underlying data to the views.

MIGRATING DATA, JOBS, AND ACCESS CONTROLS

After assessing and planning, data and job migration begins. There is no one best way to migrate data and jobs. A few ways to prioritize are as follows:

- Exploiting current opportunities
- Migrating analytical workloads first
- Focusing on the user experience first
- Prioritizing low-risk use cases first

Exploiting current opportunities focuses on use cases that demonstrate clear business value. This could be a long backlog request that can be delivered now that you have access to the improved query-processing capabilities of BigQuery.

Online transaction processing systems are typically the source of data for data warehouses; however, data warehouses can also be the source of information for transaction-processing systems. For example, sales KPIs can be calculated in the data warehouse and sent back to a sales management system, where the data is used to calculate sales bonuses. In cases like this, the transaction-processing system may have more dependencies and may not be easily moved to the cloud. In that case, the analytical workload can be migrated to the cloud while the online transaction processing systems stay in on-premises systems.

Another option is to deliver data and reporting tools that can be leveraged by analysts or other users who can start generating insights immediately. Also, customers may find it useful to have aggregated data about their accounts. For example, small to midsize businesses may want to know how much they are spending on different categories of goods, and a weekly email with summary data can help address that need.

When a team of data warehouse developers is new to GCP and BigQuery, it may be best to start with low-risk use cases. These would be use cases that do not have other systems depending on them, and they are not needed for high-priority business processes that have to be available and reliable.

Regardless of which approach you use, you should approach migrations as an iterative process. This strategy allows you to learn about your requirements, data characteristics, and GCP functional characteristics in one iteration and apply that knowledge in later iterations.

VALIDATING THE DATA WAREHOUSE

The last step of migrating a data warehouse is to validate the migration. This includes testing and verifying that

- Schemas are defined correctly and completely
- All data expected to be in the data warehouse is actually loaded
- Transformations are applied correctly and data quality checks pass
- Queries, reports, and visualizations run as expected
- Access control policies are in place
- Other governance practices are in place

It is a good practice to validate each iterative phase of the migration and document your findings. In large enterprises, there may be multiple teams migrating different workloads, and each of those teams could learn valuable lessons from the other teams.

Exam Essentials

Know the four main compute GCP products. Compute Engine is GCP's infrastructure-as-a-service (IaaS) product.

- With Compute Engine, you have the greatest amount of control over your infrastructure relative to the other GCP compute services.
- Kubernetes is a container orchestration system, and Kubernetes Engine is a managed Kubernetes service. With Kubernetes Engine, Google maintains the cluster and assumes responsibility for installing and configuring the Kubernetes platform on the cluster. Kubernetes Engine deploys Kubernetes on managed instance groups.
- App Engine is GCP's original platform-as-a-service (PaaS) offering. App Engine is designed to allow developers to focus on application development while minimizing their need to support the infrastructure that runs their applications. App Engine has two versions: App Engine Standard and App Engine Flexible.
- Cloud Functions is a serverless, managed compute service for running code in response to events that occur in the cloud. Events are supported for Cloud Pub/Sub, Cloud Storage, HTTP events, Firebase, and Stackdriver Logging.

Understand the definitions of availability, reliability, and scalability. Availability is defined as the ability of a user to access a resource at a specific time. Availability is usually measured as the percentage of time a system is operational. Reliability is defined as the probability that a system will meet service-level objectives for some duration of time. Reliability is often measured as the mean time between failures. Scalability is the ability of a system to meet the demands of workloads as they vary over time.

Know when to use hybrid clouds and edge computing. The analytics hybrid cloud is used when transaction processing systems continue to run on premises and data is extracted and transferred to the cloud for analytic processing. A variation of hybrid clouds is an edge cloud, which uses local computation resources in addition to cloud platforms. This architecture pattern is used when a network may not be reliable or have sufficient bandwidth to transfer data to the cloud. It is also used when low-latency processing is required.

Understand messaging. Message brokers are services that provide three kinds of functionality: message validation, message transformation, and routing. Message validation is the process of ensuring that messages received are correctly formatted. Message transformation is the process of mapping data to structures that can be used by other services. Message brokers can receive a message and use data in the message to determine where the message should be sent. Routing is used when hub-and-spoke message brokers are used.

Know distributed processing architectures. SOA is a distributed architecture that is driven by business operations and delivering business value. Typically, an SOA system serves a discrete business activity. SOAs are self-contained sets of services. Microservices are a variation on SOA architecture. Like other SOA systems, microservice architectures use multiple, independent components and common communication protocols to provide higher-level business services. Serverless functions extend the principles of microservices by removing concerns for containers and managing runtime environments.

Know the steps to migrate a data warehouse. At a high level, the process of migrating a data warehouse involves four stages:

- Assessing the current state of the data warehouse
- Designing the future state
- Migrating data, jobs, and access controls to the cloud
- Validating the cloud data warehouse

Review Questions

You can find the answers in the appendix.

1. A startup is designing a data processing pipeline for its IoT platform. Data from sensors will stream into a pipeline running in GCP. As soon as data arrives, a validation process, written in Python, is run to verify data integrity. If the data passes the validation, it is ingested; otherwise, it is discarded. What services would you use to implement the validation check and ingestion?

1. Cloud Storage and Cloud Pub/Sub
2. Cloud Functions and Cloud Pub/Sub
3. Cloud Functions and BigQuery
4. Cloud Storage and BigQuery

2. Your finance department is migrating a third-party application from an on-premises physical server. The system was written in C, but only the executable binary is available. After the migration, data will be extracted from the application database, transformed, and stored in a BigQuery data warehouse. The application is no longer actively supported by the original developer, and it must run on an Ubuntu 14.04 operating system that has been configured with several required packages. Which compute platform would you use?

1. Compute Engine
2. Kubernetes Engine
3. App Engine Standard
4. Cloud Functions

3. A team of developers has been tasked with rewriting the ETL process that populates an enterprise data warehouse. They plan to use a microservices architecture. Each microservice will run in its own Docker container. The amount of data processed during a run can vary, but the ETL process must always finish within one hour of starting. You want to minimize the amount of DevOps tasks the team needs to perform, but you do not want to sacrifice efficient utilization of compute resources. What GCP compute service would you recommend?

1. Compute Engine
2. Kubernetes Engine
3. App Engine Standard
4. Cloud Functions

4. Your consulting company is contracted to help an enterprise customer negotiate a contract with a SaaS provider. Your client wants to ensure that they will have access to the SaaS service and it will be functioning correctly with only minimal downtime. What metric would you use when negotiating with the SaaS provider to ensure that your client's reliability requirements are met?

1. Average CPU utilization
2. A combination of CPU and memory utilization
3. Mean time between failure
4. Mean time to recovery

5. To ensure high availability of a mission-critical application, your team has determined that it needs to run the application in multiple regions. If the application becomes unavailable in one region, traffic from that region should be routed to another region. Since you are designing a solution for this set of requirements, what would you expect to include?

1. Cloud Storage bucket
2. Cloud Pub/Sub topic
3. Global load balancer
4. HA VPN

6. A startup is creating a business service for the hotel industry. The service will allow hotels to sell unoccupied rooms on short notice using the startup's platform. The startup wants to make it as easy as possible for hotels to share data with the platform, so it uses a message queue to collect data about rooms that are available for rent. Hotels send a message for each room that is available and the days that it is available. Room identifier and dates are the keys that uniquely identify a listing. If a listing exists and a message is received with the same room identifier and dates, the message is discarded. What are the minimal guarantees that you would want from the message queue?

   1. Route randomly to any instance that is building a machine learning model
   2. Route based on the sensor identifier so identifiers in close proximity are used in the same model
   3. Route based on machine type so only data from one machine type is used for each model
   4. Route based on timestamp so metrics close in time to each other are used in the same model

7. Sensors on manufacturing machines send performance metrics to a cloud-based service that uses the data to build models that predict when a machine will break down. Metrics are sent in messages. Messages include a sensor identifier, a timestamp, a machine type, and a set of measurements. Different machine types have different characteristics related to failures, and machine learning engineers have determined that for highest accuracy, each machine type should have its own model. Once messages are written to a message broker, how should they be routed to instances of a machine learning service?

   1. Route randomly to any instance that is building a machine learning model
   2. Route based on the sensor identifier so that identifiers in close proximity are used in the same model
   3. Route based on machine type so that only data from one machine type is used for each model
   4. Route based on timestamp so that metrics close in time to one another are used in the same model

8. As part of a cloud migration effort, you are tasked with compiling an inventory of existing applications that will move to the cloud. One of the attributes that you need to track for each application is a description of its architecture. An application used by the finance department is written in Java, deployed on virtual machines, has several distinct services, and uses the SOAP protocol for exchanging messages. How would you categorize this architecture?

   1. Monolithic
   2. Service-oriented architecture (SOA)
   3. Microservice
   4. Serverless functions

9. As part of a migration to the cloud, your department wants to restructure a distributed application that currently runs several services on a cluster of virtual machines. Each service implements several functions, and it is difficult to update one function without disrupting operations of the others. Some of the services require third-party libraries to be installed. Your company has standardized on Docker containers for deploying new services. What kind of architecture would you recommend?

   1. Monolithic
   2. Hub-and-spoke
   3. Microservices
   4. Pipeline architecture

10. The CTO of your company is concerned about the rising costs of maintaining your company's enterprise data warehouse. Some members of your team are advocating to migrate to a cloud-based data warehouse such as BigQuery. What is the first step for migrating from the on-premises data warehouse to a cloud-based data warehouse?

   1. Assessing the current state of the data warehouse
   2. Designing the future state of the data warehouse
   3. Migrating data, jobs, and access controls to the cloud
   4. Validating the cloud data warehouse

11. When gathering requirements for a data warehouse migration, which of the following would you include in a listing of technical requirements?

   1. Data sources, data model, and ETL scripts
   2. Data sources, data model, and business sponsor roles
   3. Data sources only
   4. Data model, data catalog, ETL scripts, and business sponsor roles

12. In addition to concerns about the rising costs of maintaining an on-premises data warehouse, the CTO of your company has complained that new features and reporting are not being rolled out fast enough. The lack of adequate business intelligence has been blamed for a drop in sales in the last quarter. Your organization is incurring what kind of cost because of the backlog?

    1. Capital
    2. Operating
    3. Opportunity
    4. Fiscal

13. The data modelers who built your company's enterprise data warehouse are asking for your guidance to migrate the data warehouse to BigQuery. They understand that BigQuery is an analytical database that uses SQL as a query language. They also know that BigQuery supports joins, but reports currently run on the data warehouse are consuming significant amounts of CPU because of the number and scale of joins. What feature of BigQuery would you suggest they consider in order to reduce the number of joins required?

    1. Colossus filesystem
    2. Columnar data storage
    3. Nested and repeated fields
    4. Federated storage

14. While the CTO is interested in having your enterprise data warehouse migrated to the cloud as quickly as possible, the CTO is particularly risk averse because of errors in reporting in the past. Which prioritization strategy would you recommend?

    1. Exploiting current opportunities
    2. Migrating analytical workloads first
    3. Focusing on the user experience first
    4. Prioritizing low-risk use cases first

15. The enterprise data warehouse has been migrated to BigQuery. The CTO wants to shut down the on-premises data warehouse but first wants to verify that the new cloud-based data warehouse is functioning correctly. What should you include in the verification process?

    1. Verify that schemas are correct and that data is loaded
    2. Verify schemas, data loads, transformations, and queries
    3. Verify that schemas are correct, data is loaded, and the backlog of feature requests is prioritized
    4. Verify schemas, data loads, transformations, queries, and that the backlog of feature requests is prioritized

Support / Sign Out

PREV
Chapter 3 Designing Data Pipelines

NEXT
Chapter 5 Building and Operationalizing Processing Infrastr…