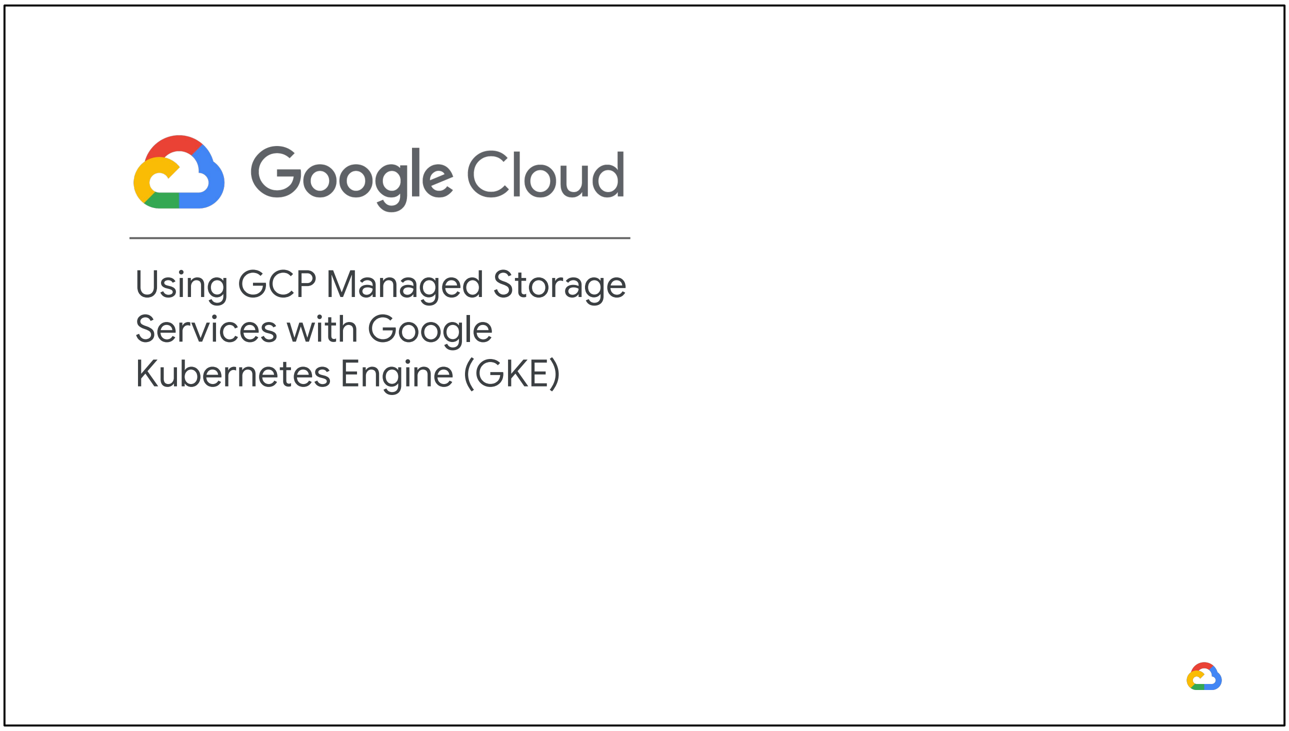
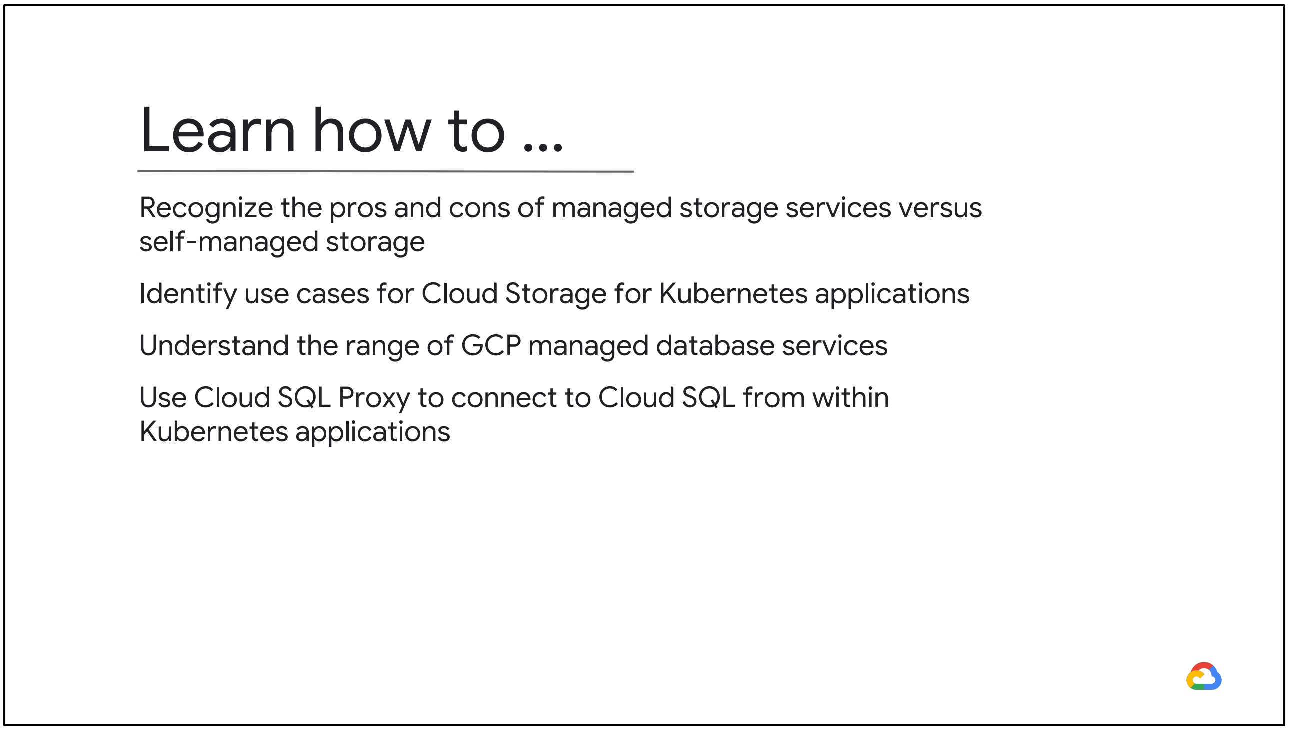


Google Cloud

Using GCP Managed Storage Services with Google Kubernetes Engine (GKE)

Welcome to the “Using GCP Managed Storage Services with GKE” module. When you implement solutions using GKE, you can choose to build your own storage solutions by attaching volumes to containers. But these require you to manage and protect your data yourself. You can also choose to use GCP managed storage services, so it becomes Google’s job to manage the availability and performance of your data. This is the theme for this module.



Learn how to ...

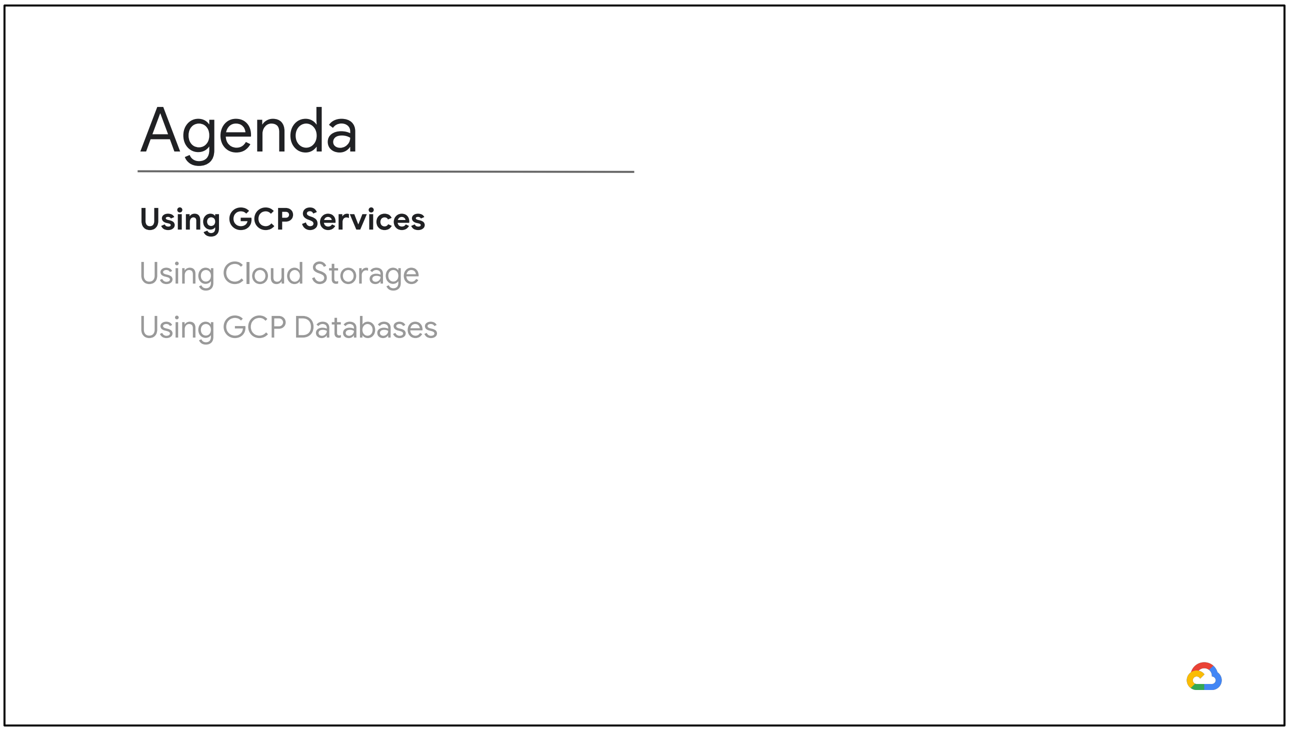
Recognize the pros and cons of managed storage services versus self-managed storage

Identify use cases for Cloud Storage for Kubernetes applications

Understand the range of GCP managed database services

Use Cloud SQL Proxy to connect to Cloud SQL from within Kubernetes applications

In this module, you'll learn how to identify use cases for Cloud Storage for applications running in a Kubernetes cluster; recognize the pros and cons of managed storage services versus self-managed containerized storage; identify use cases for Cloud Storage for applications running in a Kubernetes cluster; understand the range of GCP managed database services; and how to simplify the task of connecting from within Kubernetes applications to Cloud SQL, one of those services, using the Cloud SQL Proxy.



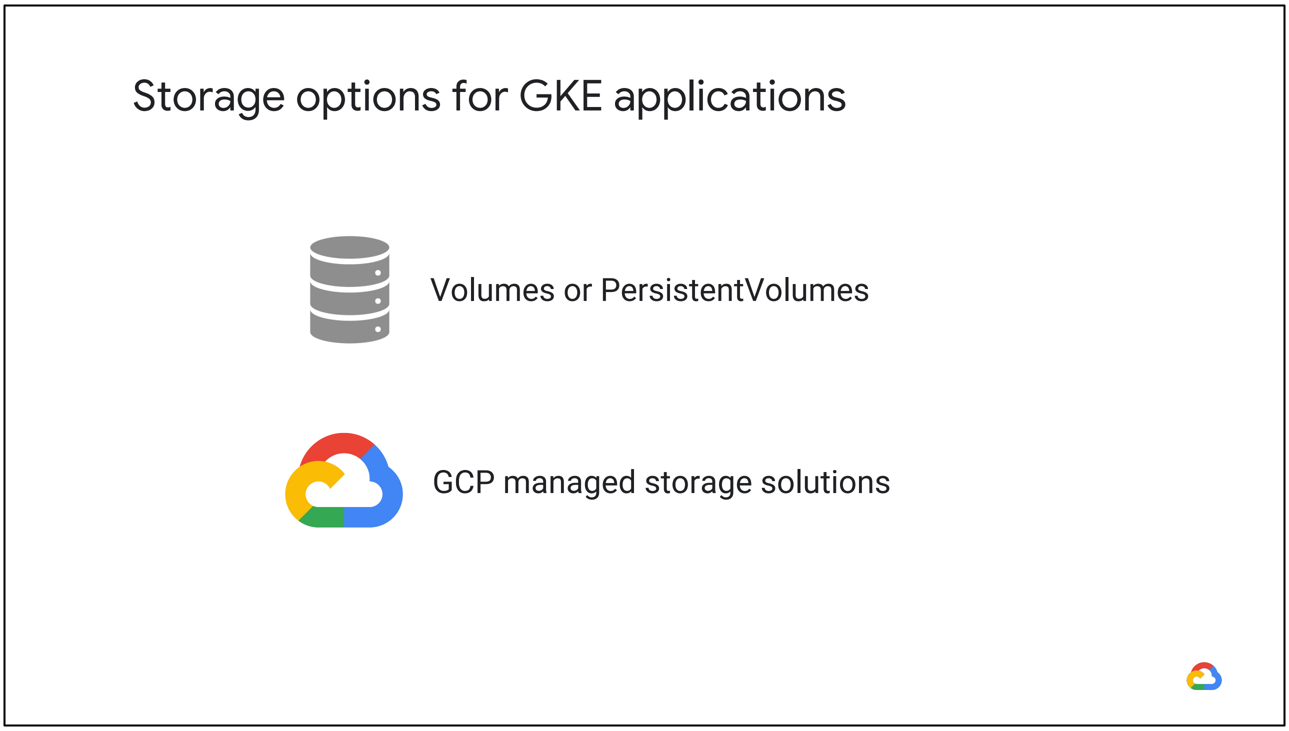
Agenda

Using GCP Services

Using Cloud Storage

Using GCP Databases

Let’s start by learning how to use GCP Services. In this first lesson, you will learn about the pros and cons of using GCP managed storage services versus self-managed containerized storage.



Storage options for GKE applications

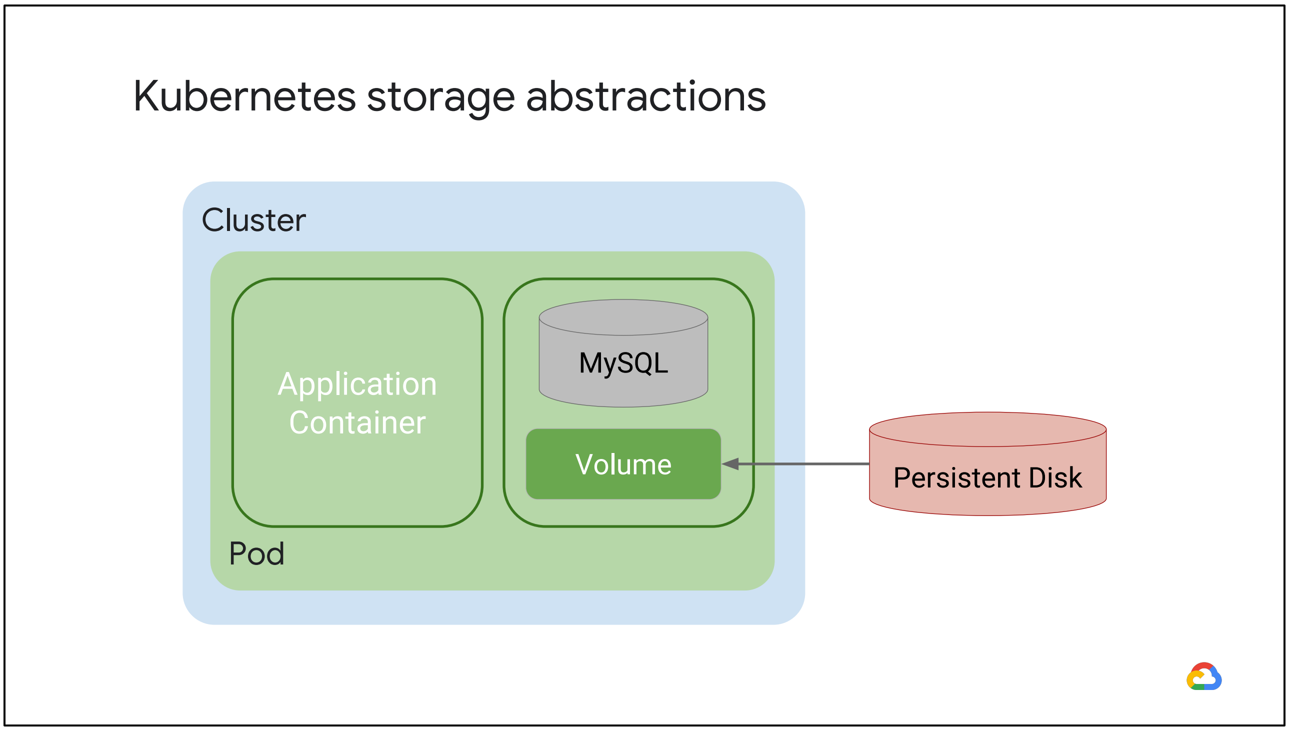
Volumes or PersistentVolumes

GCP managed storage solutions

There are several storage options for applications running on Google Kubernetes Engine. They make different tradeoffs, and you can choose whichever is right for your applications.

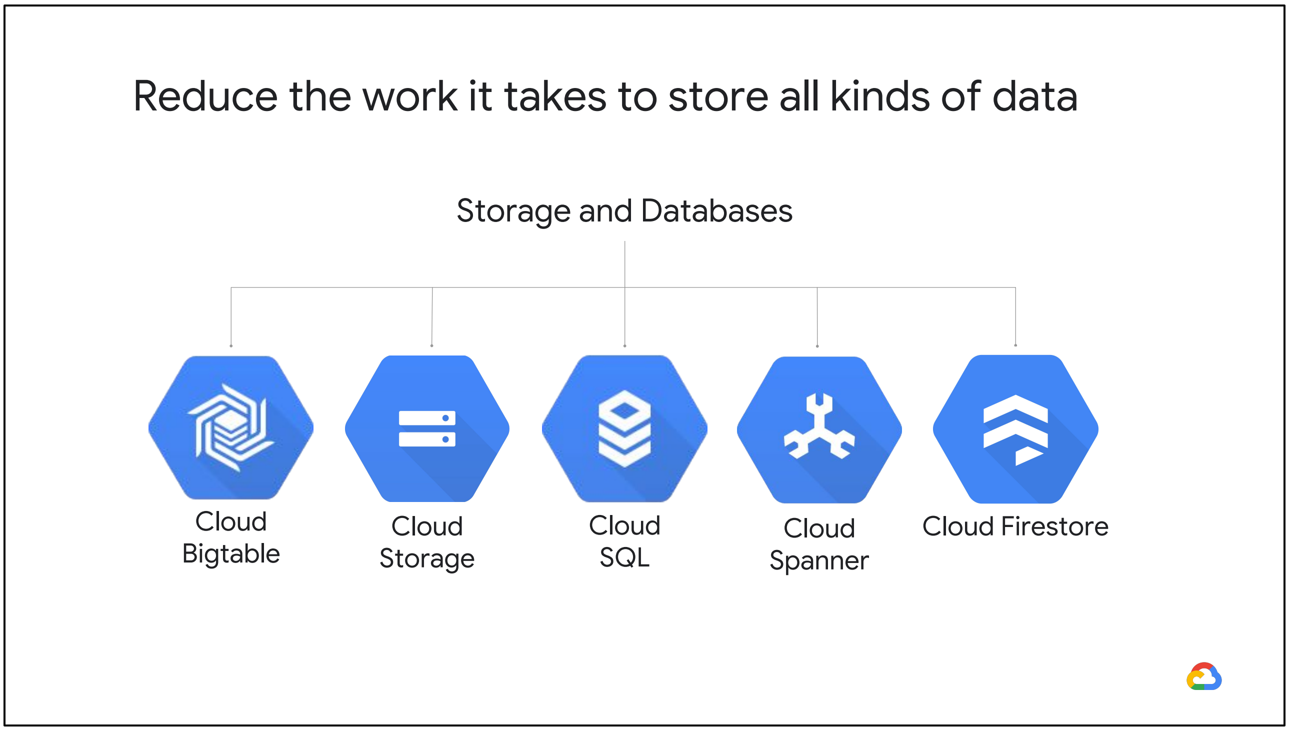
Kubernetes provides storage abstractions—Volumes and PersistentVolumes—which you can use to offer storage to applications in your cluster. You learned about these in another module in this specialization. In this module, you'll learn about the alternative: GCP managed storage solutions.

Remember that Volumes and PersistentVolumes provide file system capacity accessible directly to your applications. You can use these for durable file storage, or as a backing store for databases or other storage services that you deploy and manage yourself. In GKE, these are typically backed by Compute Engine Persistent Disks. However, if you use these Kubernetes storage abstractions, you’ll have to manage the storage services you build using them.



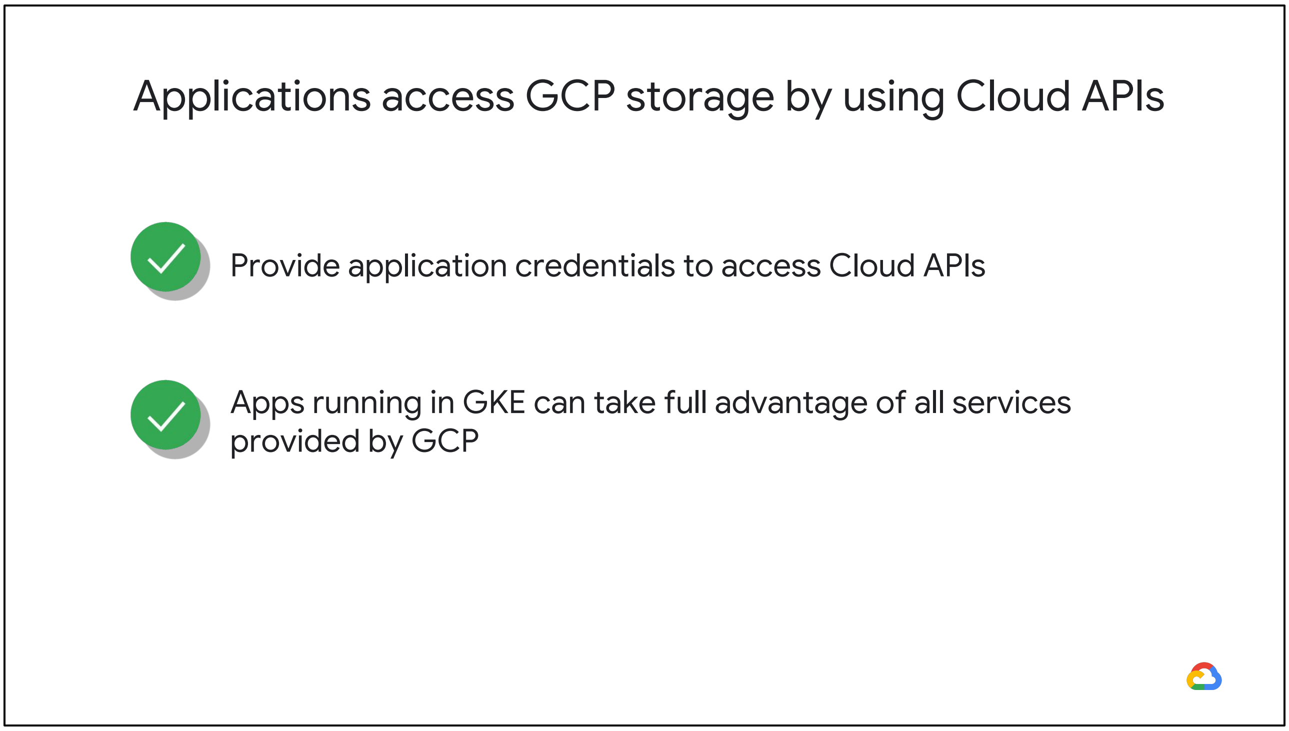
Kubernetes storage abstractions

For example, you can build and deploy a MySQL server as a container yourself using a Kubernetes Volume based on Persistent Disks to store the database files. You will, however, have to manage the server application lifecycle yourself. You will be entirely responsible for building a resilient and reliable service.



Reduce the work it takes to store all kinds of data

You may want to focus on your application rather than maintaining a storage service for it. Google Cloud's fully-managed database and storage services reduce the work it takes to store all kinds of data. The relational and non-relational database and object storage services of GCP can help remove operational management burden. You will learn more about each of these services later in the module.



Applications access GCP storage by using Cloud APIs

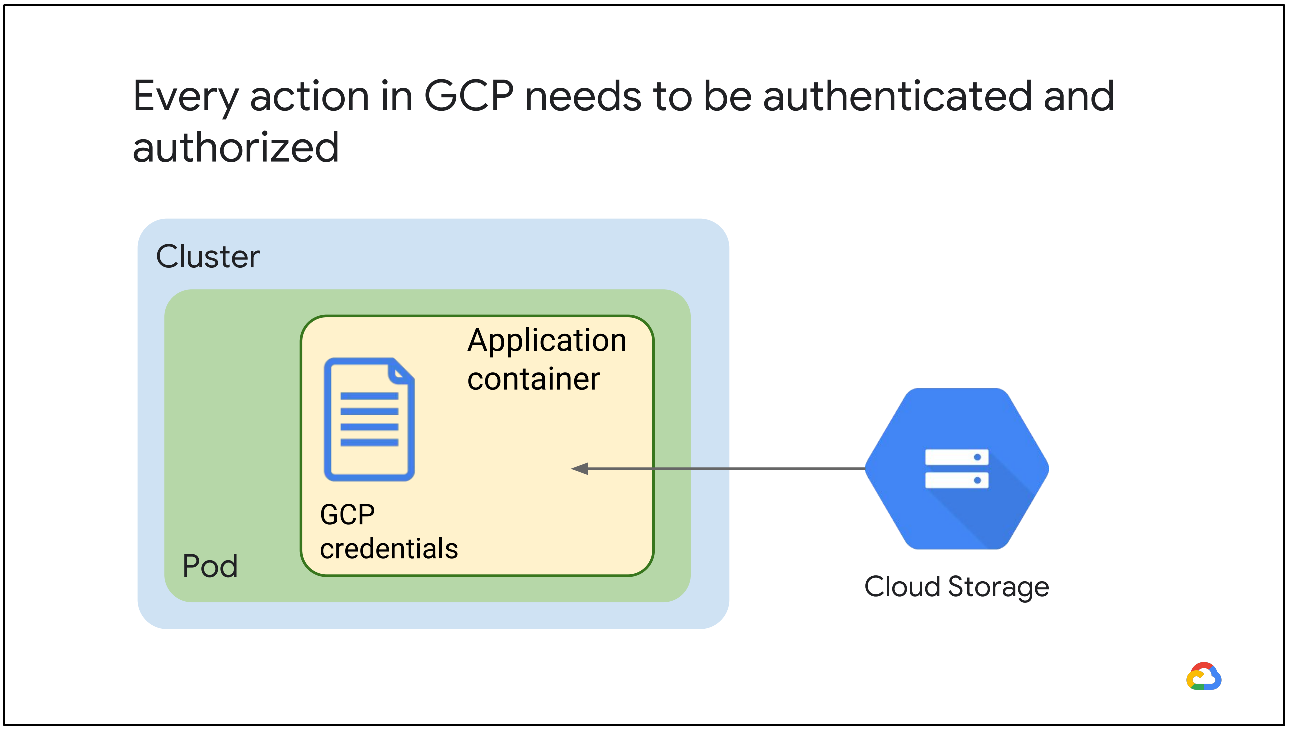
✓ Provide application credentials to access Cloud APIs

✓ Apps running in GKE can take full advantage of all services provided by GCP

To consume these GCP managed storage services within a GKE-managed Kubernetes cluster, your containerized application must be able to use the Google Cloud Platform APIs.

To allow your applications to use a GCP API, you need to enable the relevant API, and you will need to give the proper credentials to your application. Using these credentials, the application will authenticate itself to the service and get authorization to perform tasks.

When the appropriate credentials have been created and made available to your applications in GKE, they can take full of advantage of cloud services.



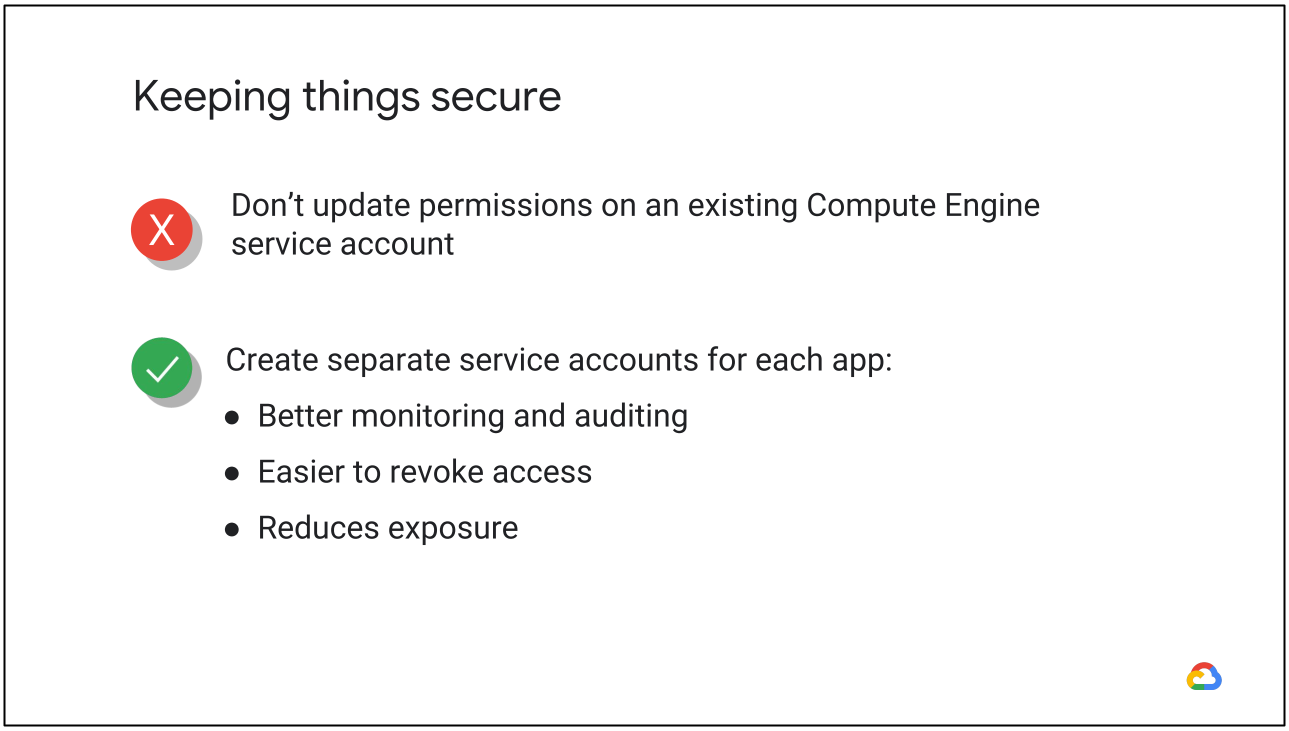
Every action in GCP needs to be authenticated and authorized

Cluster
Pod
Application container
GCP credentials
Cloud Storage

Every action in GCP needs to be authenticated and authorized. Your application must be able to send credentials for the Service Accounts it uses directly to the API for the GCP API it uses. For example, in this diagram, a containerized application is using resources in Google Cloud Storage by presenting credentials.

Your applications should use Cloud IAM Service Accounts for GCP API authentication and authorization. (Remember, Kubernetes has service accounts too, but they serve a different purpose. Cloud IAM service accounts are defined at the GCP level, not inside Kubernetes.)

In GKE, you usually provide the authentication credentials for Cloud IAM Service Accounts to applications using Kubernetes Secrets, which you learned about in an earlier module in this specialization. These secrets contain the credentials in either JSON or P12 format.



Keeping things secure

✗ Don’t update permissions on an existing Compute Engine service account

✓ Create separate service accounts for each app:

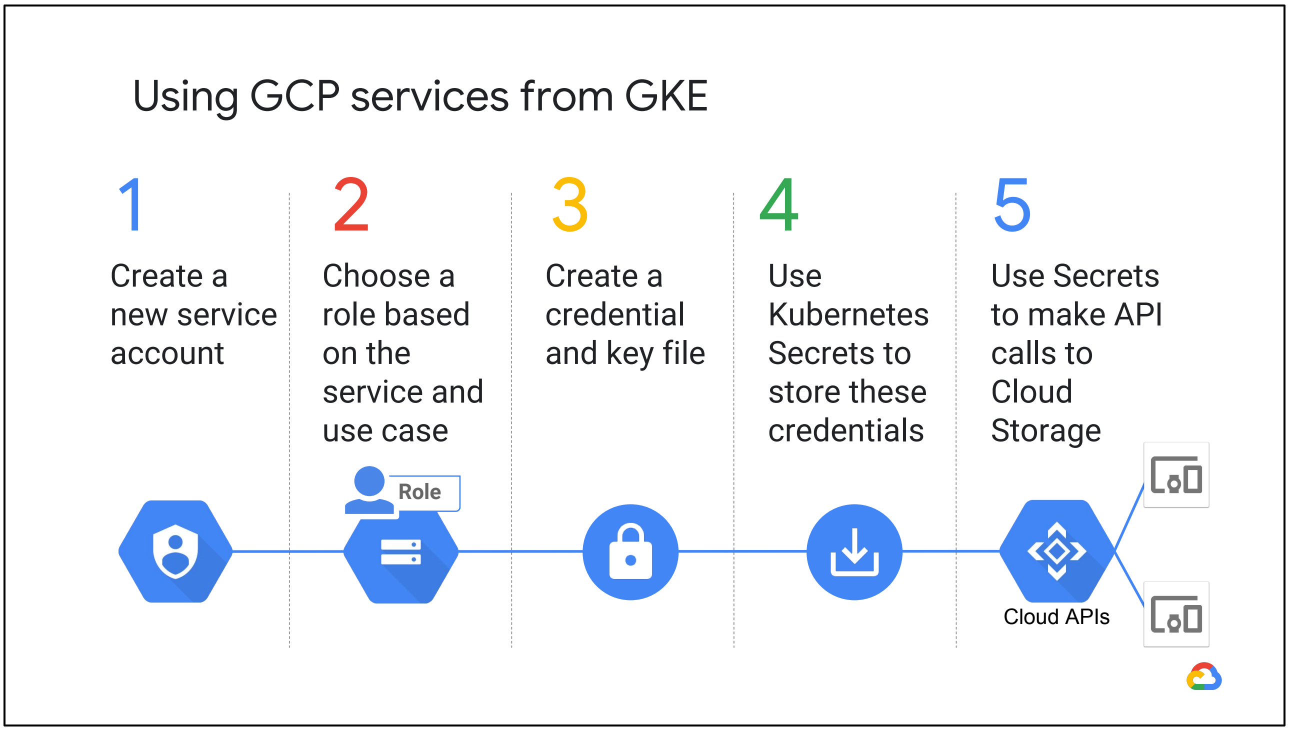
- Better monitoring and auditing
- Easier to revoke access
- Reduces exposure

The service account must be assigned an IAM role that has the permissions your application requires. Don’t simply update the permissions for the existing Compute Engine default service account.

Instead, create a new service account for each application that needs to make requests to GCP services.
Using separate service accounts for different applications lets you minimize the privileges associated with each account because having separate accounts makes it easier to enable monitoring and auditing of API requests at the application level.

It’s also easier to revoke access from specific applications by deleting the service accounts associated with those applications, instead of revoking access for a shared service account. That would affect multiple applications at the same time, which could be a management problem.

In the event of a security breach, having separate accounts can reduce your exposure. Having many service accounts lowers each account’s value to an attacker.



Using GCP services from GKE

1 Create a new service account

2 Choose a role based on the service and use case

3 Create a credential and key file

4 Use Kubernetes Secrets to store these credentials

5 Use Secrets to make API calls to Cloud Storage

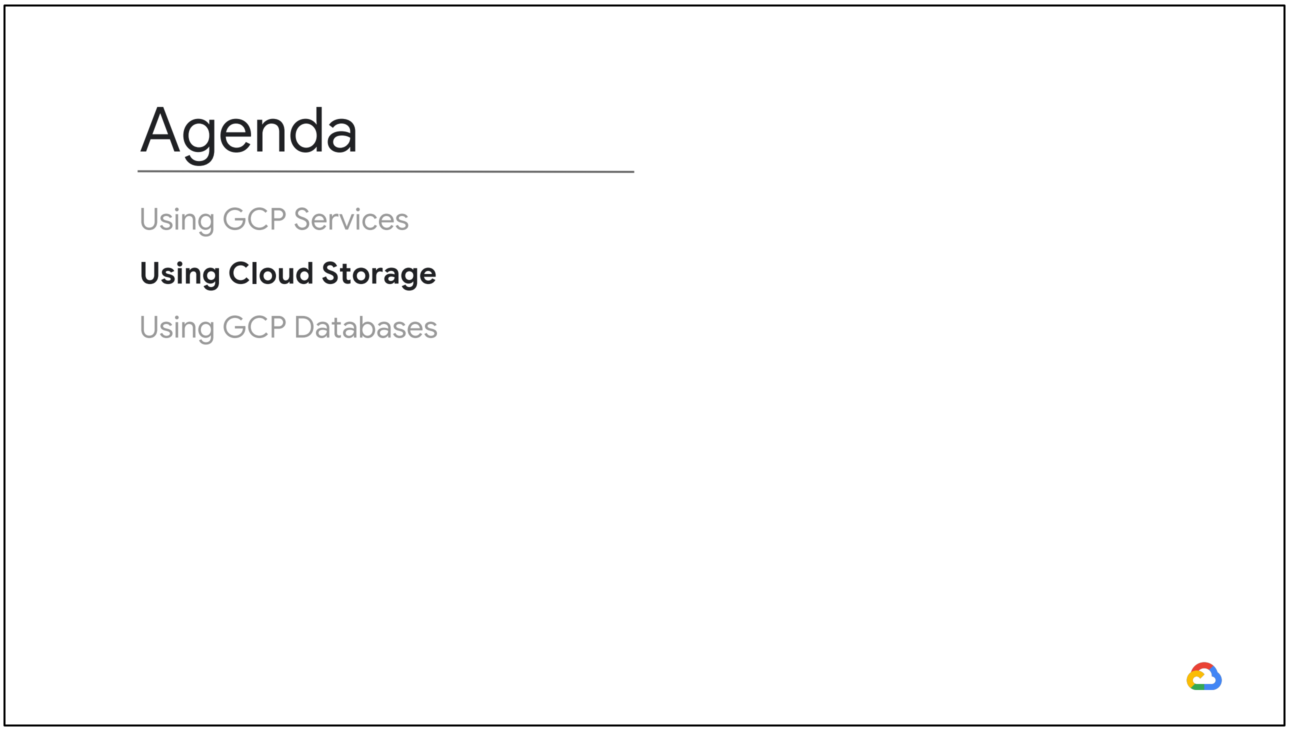
The first step in the process is to create a new Cloud IAM service account.

Then you must choose an appropriate Cloud IAM role during the creation of the service account. Roles grant permissions to the accounts bearing them. Take Cloud Storage as an example. Say your application needs to read from Cloud Storage buckets. Giving its service account the Storage Object Viewer role grants the application permission to read data from Cloud Storage but not change it.

After you select the required roles, specify whether you want to use JSON or P12 format for the credential file. Then a new key is created for you and saved in your chosen format. It is your responsibility to manage the rotation of these credentials over time.

In order to use the service account inside GKE, you should create a Kubernetes Secret resource type to store the service account's credential file. Google Cloud Key Management Service, or “KMS,” can be used to manage and rotate secrets, including Cloud IAM authentication credentials. That provides further protections for Secret resources in GKE. Cloud KMS is out of our scope in this specialization, but there is a link to learn more about it associated with this module.

You then mount the Secret as a Volume as part of the Pod or Deployment definition. Application containers within the Pods can now access the credential as a file and use it to make API calls through client libraries. These credentials authenticate your application as the service account you created. Assuming you gave that service account a role with the right permissions, your application will be able to use the Cloud Storage service and the objects it manages for your application.



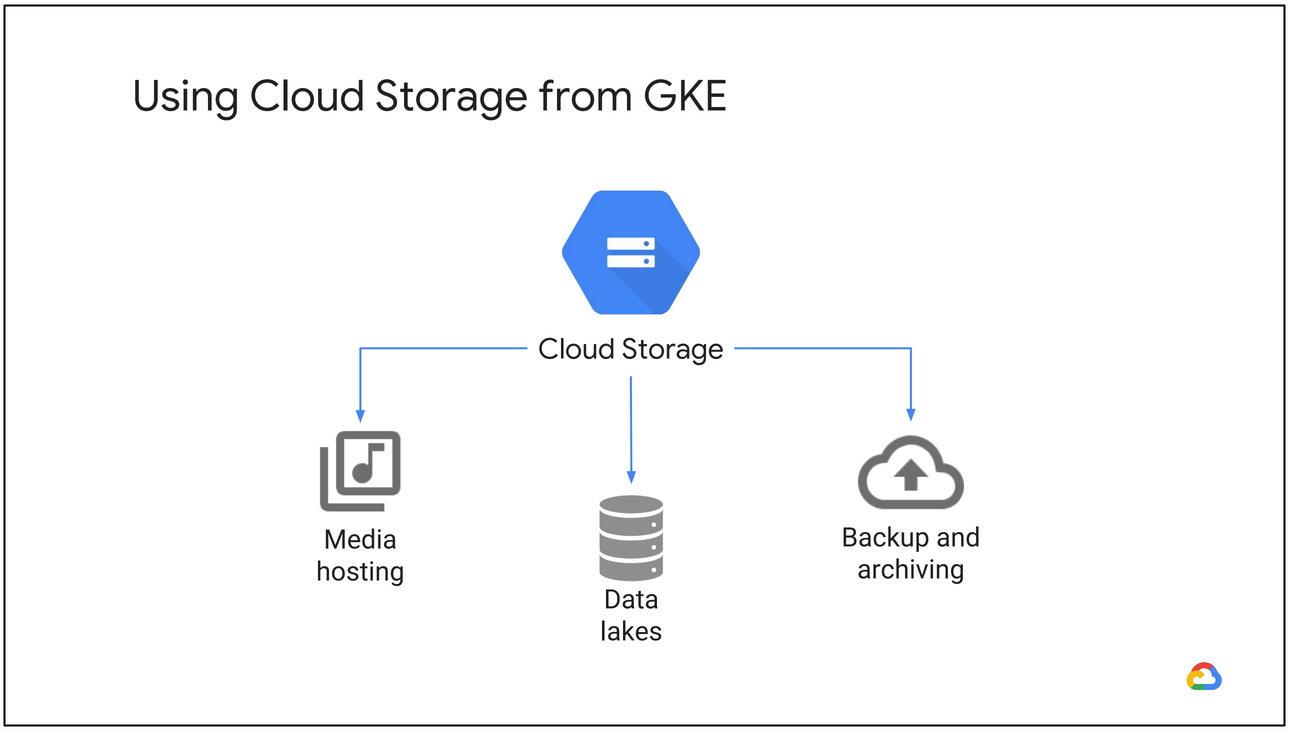
Agenda

Using GCP Services

Using Cloud Storage

Using GCP Databases

During the previous lesson, I used Cloud Storage in my example of how to consume GCP services. In this lesson, you'll learn about how to consume Cloud Storage solutions within GKE.



Using Cloud Storage from GKE

Cloud Storage is an object storage service with a wide variety of uses. “Object storage” simply means the storage of ordered groups of bytes. The storage service doesn’t know or care about the structure and semantics of those bytes. Your application determines that.

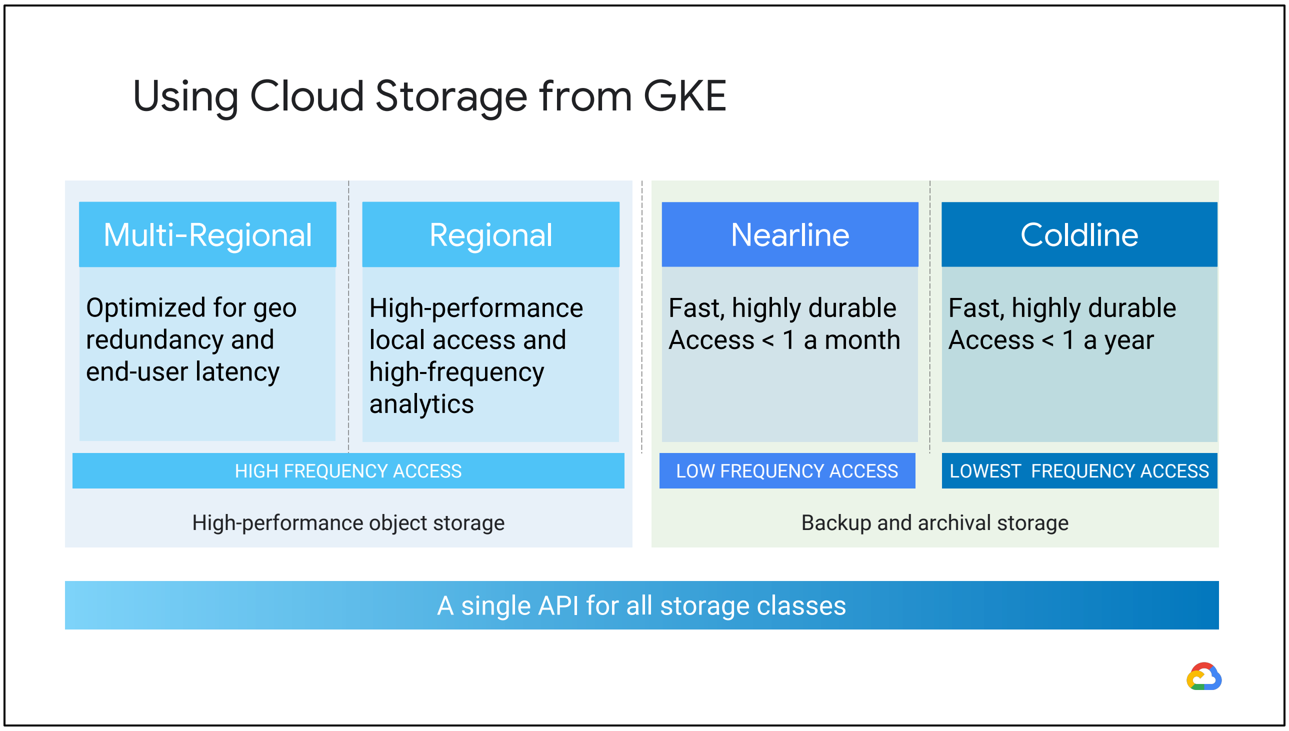
Cloud Storage is commonly used for media hosting, such as serving images for a website, or streaming music and videos.

Cloud Storage can also be used as a data lake for analytics and machine learning workloads such as genomics and data analytics.

You can also use Cloud Storage for backups and archives, such as tape migrations or disaster recovery.

Any of these kinds of applications, and more, that require object storage can be deployed to GKE. For example, your mobile app backend might be deployed on GKE, and it would probably need to access images from Cloud Storage buckets.

To sum up Cloud Storage: it provides object storage for unstructured files, sometimes called “blob storage.” You can also use it to store structured data, but the service isn’t optimized for query capabilities. If you need to query data, other services are ideal for that, and you will learn about them soon. Cloud Storage is not designed for use as a file system, so it doesn’t take the place of Persistent Volumes.



Using Cloud Storage from GKE

Multi-Regional	Regional	Nearline	Coldline
Optimized for geo redundancy and end-user latency	High-performance local access and high-frequency analytics	Fast, highly durable Access < 1 a month	Fast, highly durable Access < 1 a year
HIGH FREQUENCY ACCESS		LOW FREQUENCY ACCESS	LOWEST FREQUENCY ACCESS
High-performance object storage		Backup and archival storage	

A single API for all storage classes

Cloud Storage offers four storage classes. Careful. Kubernetes also has a concept called “storage classes,” but they are unrelated. Kubernetes storage classes are profiles for how Persistent Volumes are provisioned. In contrast, Cloud Storage’s storage classes offer you different tradeoffs of price and geographic redundancy.

Multi-Regional Storage is appropriate for storing data that is frequently accessed, such as website content, interactive workloads, or data that supports mobile and gaming applications.

Regional Storage is appropriate for storing data in the same regional location as the Compute Engine instances or Google Kubernetes Engine clusters. That delivers better performance for data-intensive computations, as opposed to storing data in a multi-regional location. You pay for regional and multi-regional storage based on how much you consume.

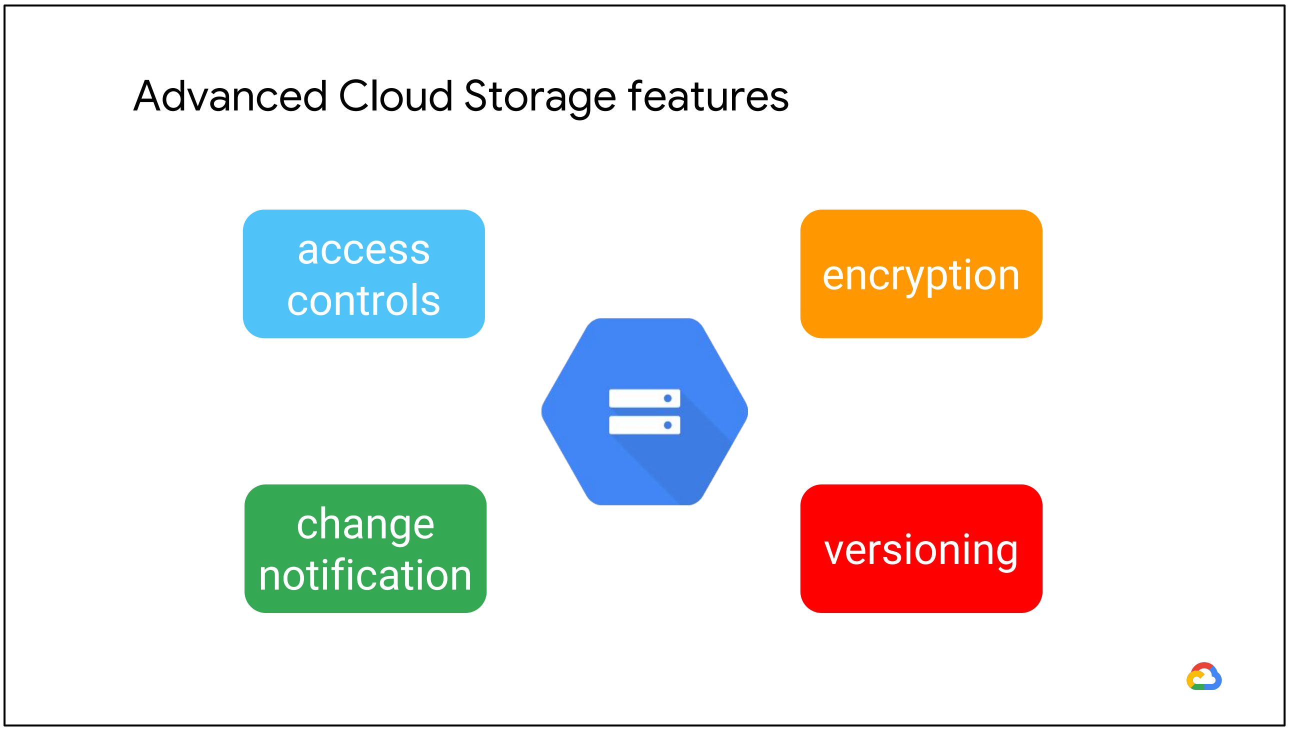
Nearline Storage is ideal for data that is read or modified on average less than once a month; for example, if you want to continuously add files to Cloud Storage and plan to access those files quarterly for analysis. Nearline Storage is also appropriate for data backup, disaster recovery, and archival storage.

Coldline Storage is a very low-cost, highly durable storage service for data archiving, online backup, and disaster recovery. Unlike with other "cold" storage services, data is available within milliseconds, not hours or days. The pricing structure for Coldline storage is ideal for data that is retrieved less than once a year.

Nearline and Coldline offer a trade-off. They cost less per gigabyte stored than do Multi-regional and Regional storage. But they assess additional charges for retrieval. So, the less frequently you anticipate accessing data, the better a candidate it is for Nearline and Coldline. As you would guess, Coldline has the lowest storage price and a higher retrieval price than Nearline.

All storage classes offer you low latency and high durability. You only need a single API for all storage classes. Your application can use several storage classes for different kinds of data.

Automatic rules can be configured that will move files that haven’t been accessed in a specified period to coldline storage. From a user perspective nothing changes with these files, they remain accessible using the same applications and interfaces, but the files are now stored more cost effectively.



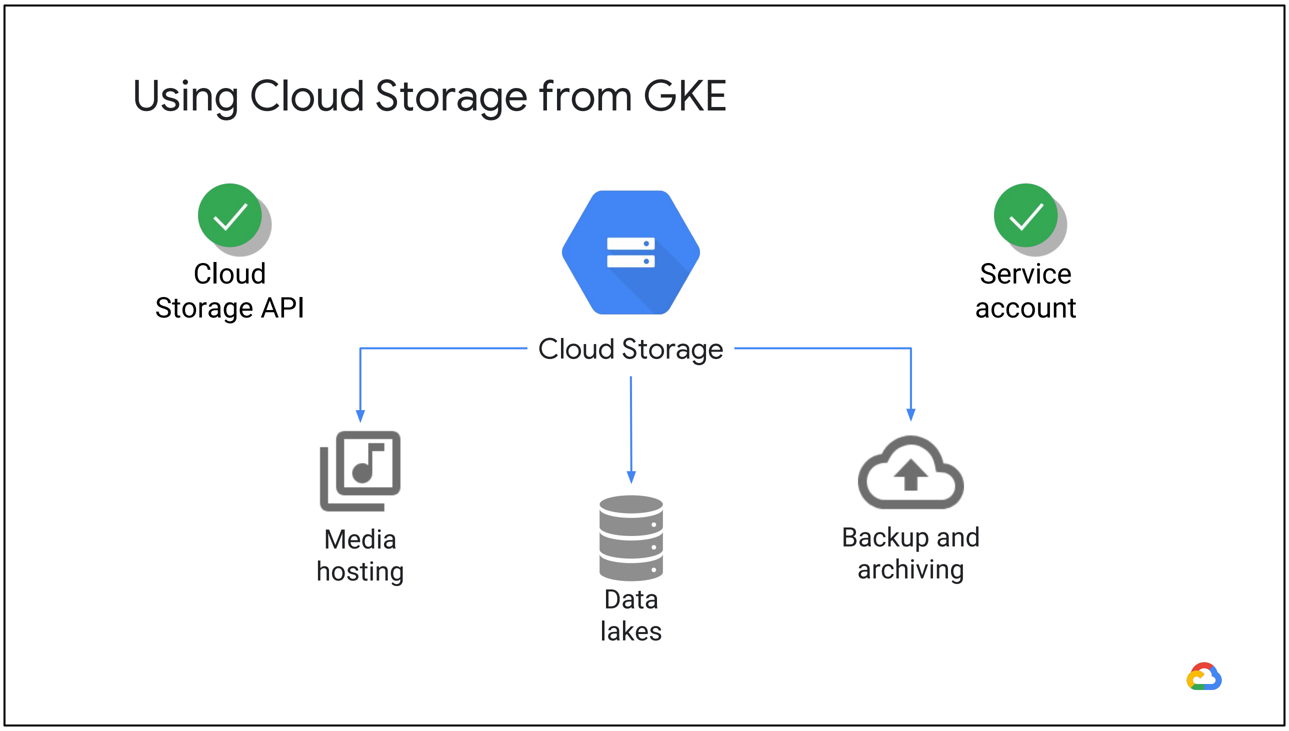
Advanced Cloud Storage features

Cloud Storage supports comprehensive, granular, access control features. You use these features to ensure that all your objects stored in Cloud Storage are protected appropriately.

Cloud Storage always encrypts your data on the server side, before it is written to disk. Cloud Storage also supports server-side encryption using either customer-supplied or customer-managed encryption keys. This encryption occurs after Cloud Storage receives your data, but before the data is written to disk and stored. You should consider these options if you face regulatory requirements that mean the default encryption isn’t enough.

The Cloud Storage service offers object change notifications, so that an application gets notified through a call to a URL, whenever changes are made to a Cloud Storage bucket. You can also achieve the same effect in a more scalable and manageable way using Cloud Storage together with Cloud Pub/Sub, which is GCP’s fully-managed real-time messaging service.

Cloud Storage supports versioning, but you must enable it before it can be used. After you’ve enabled versioning, Cloud Storage creates an archived version of an object each time the live version of the object is overwritten or deleted. The archived version keeps the name of the object but is uniquely identified by a generation number.

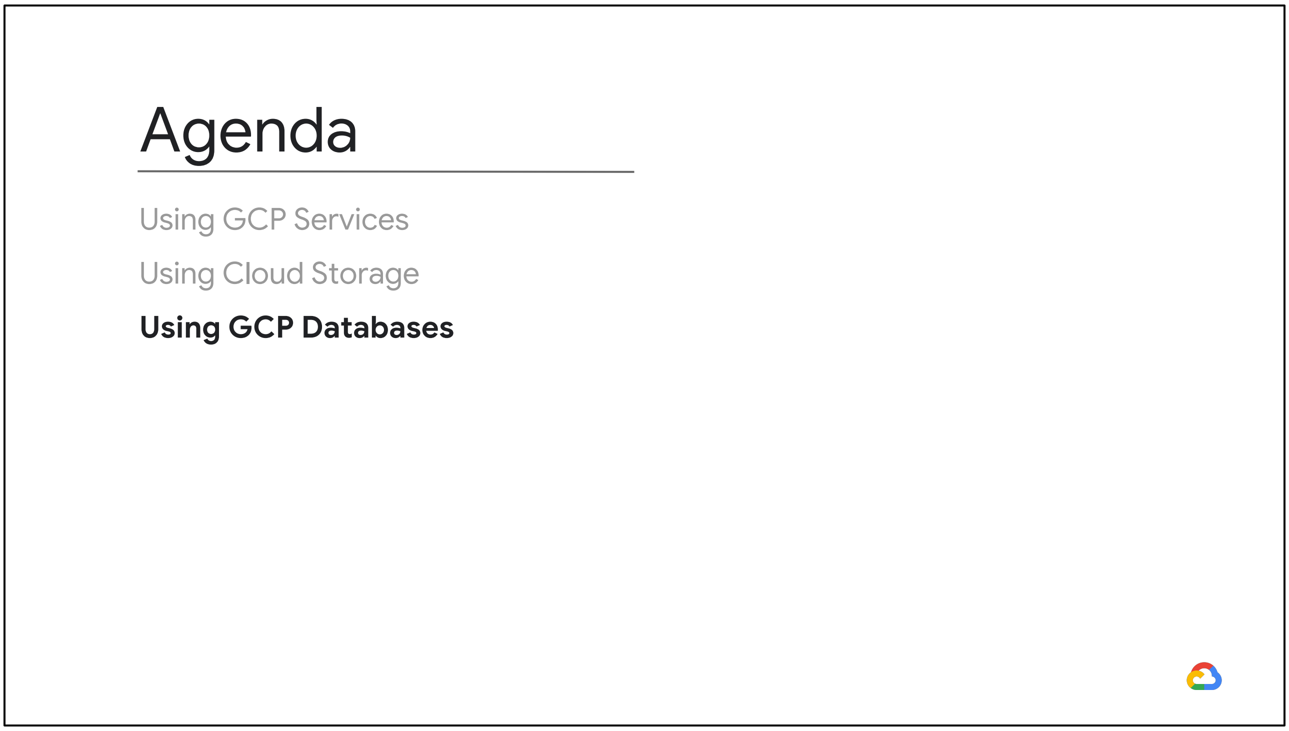


Using Cloud Storage from GKE

Applications within your GKE cluster can access Cloud Storage using the Cloud Storage APIs. As with other Google Cloud APIs, you must enable the Cloud Storage API before your applications can use it.

Don’t forget: After you enable the relevant API, your application will need to use a service account to access the service you want to use.

Google Storage is strongly consistent and offers read after write guarantees. This is very useful for Kubernetes applications where loosely coupled Pods might need to write out and read back in data independently.



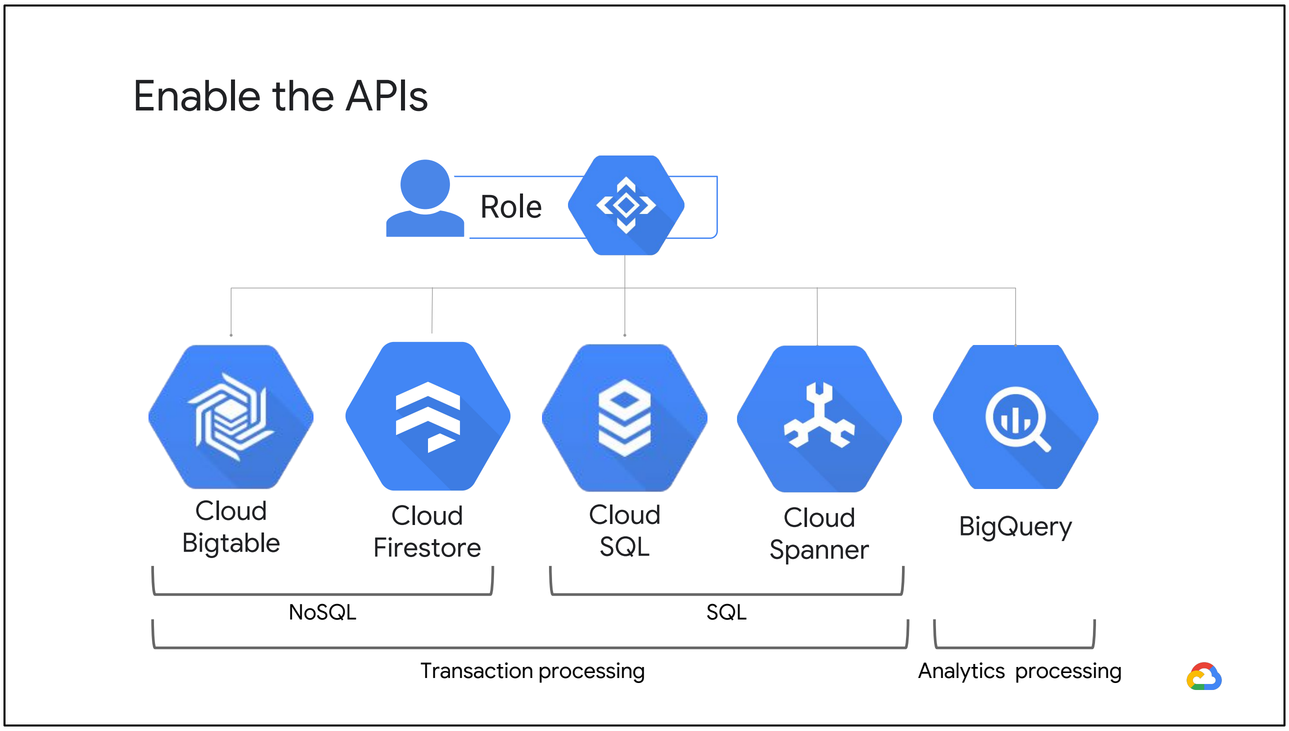
Agenda

Using GCP Services

Using Cloud Storage

Using GCP Databases

Now let’s look at the types of managed database services provided by GCP. In this lesson we will look at the key features of Cloud Bigtable, BigQuery, Cloud Firestore, Cloud SQL and Cloud Spanner. You will also learn how to use Cloud SQL Proxy to simplify the process of connecting GKE applications to Cloud SQL instances.



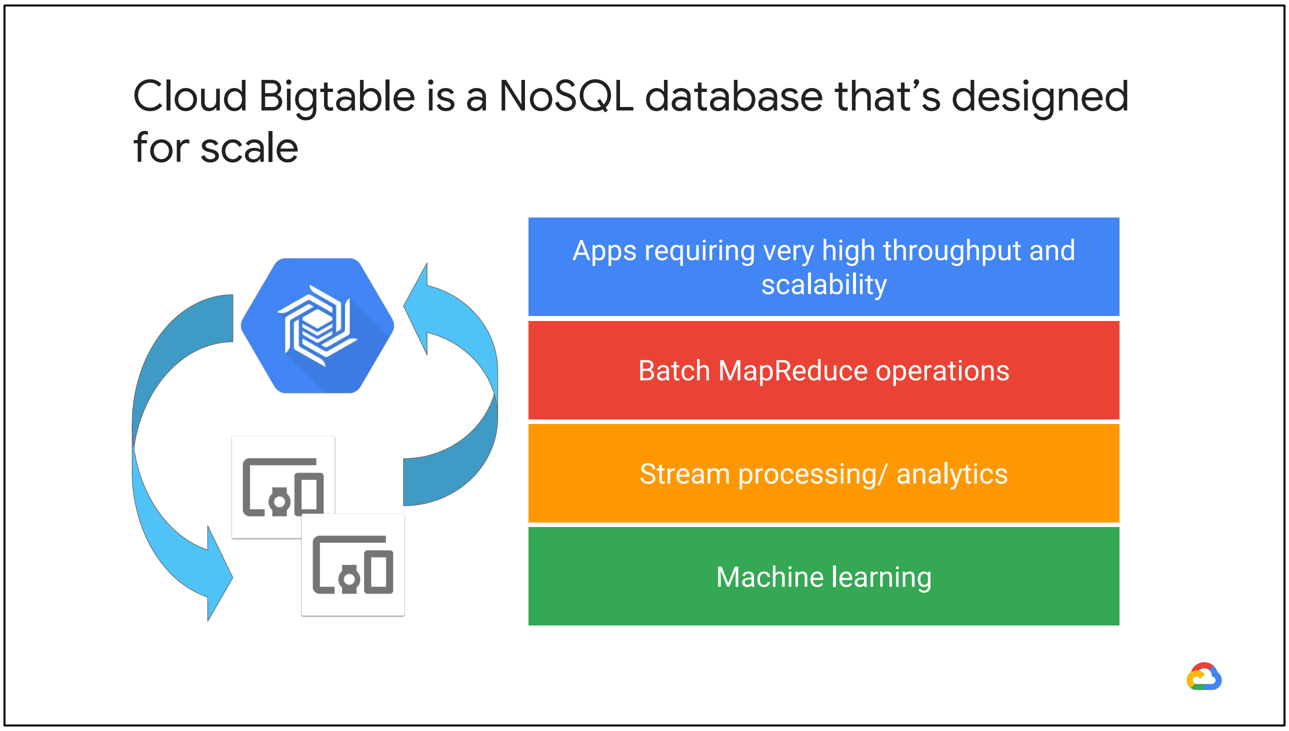
You can use the same approach that you use to access Cloud Storage for accessing other GCP services, such as Cloud Bigtable, BigQuery, Cloud Firestore, Cloud SQL, and Cloud Spanner.

GCP's database services serve two overall purposes. Many are suitable for use to support an application's online data use, such as its storage and retrieval of data. On the other hand, GCP also offers an analytics database service that's optimized for data mining and discovery of patterns and trends.

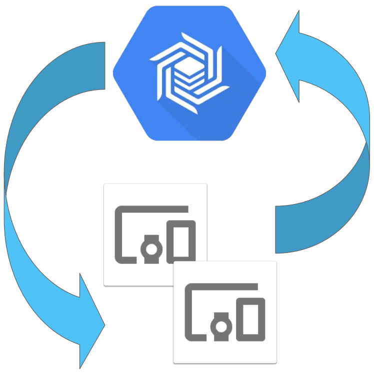
In turn, the database services that back online applications fall into the categories of SQL and NoSQL. SQL database services are for relational data, and you choose them when you want the database engine to help you enforce the semantics of the database. NoSQL database services are for flexibly structured data, and it's up to your application to maintain your data's integrity.

You’ll learn how to choose among these database services later in this lesson. But, whichever you choose, make sure that you enable the API of the service and give your applications appropriate Cloud IAM service accounts. You will do this in the lab for Cloud SQL.

Now I’ll introduce you to each of the GCP database services. You can use each of them with Kubernetes applications running in GKE, and each of them is suited for particular use cases.



Cloud Bigtable is a NoSQL database that's designed for scale

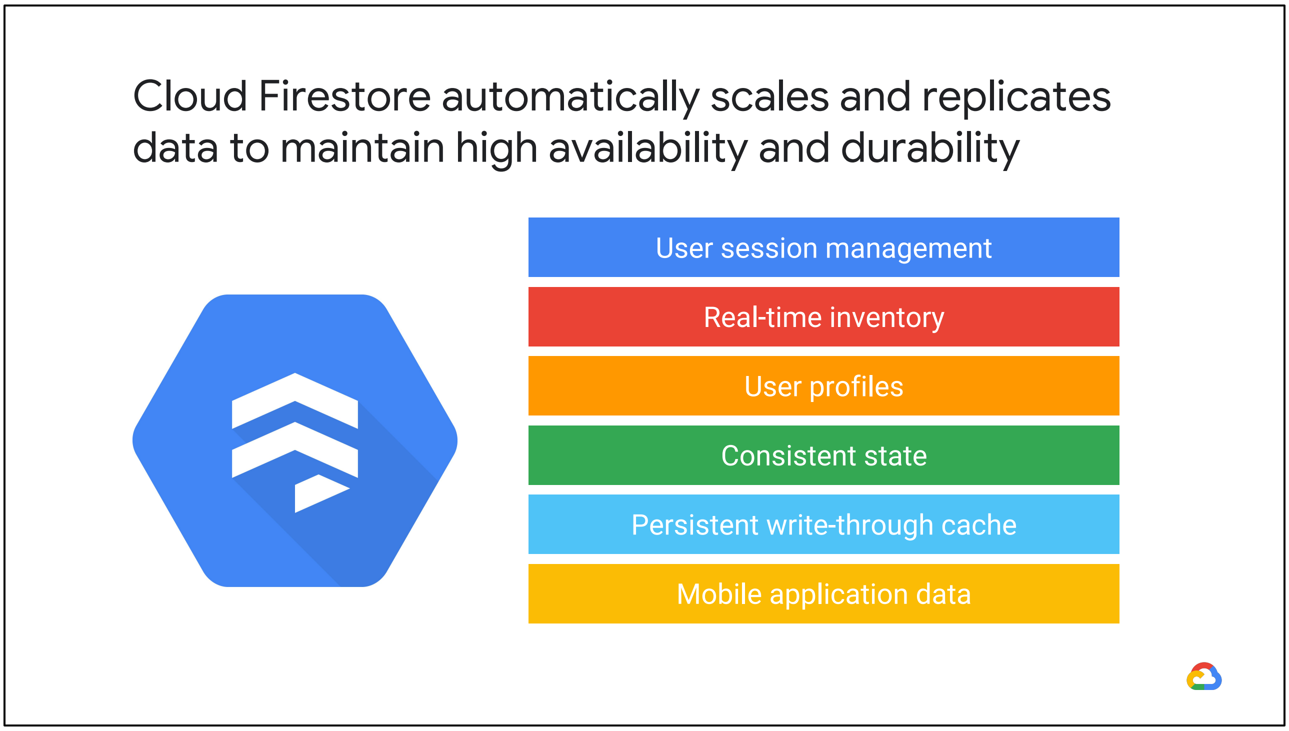


- Apps requiring very high throughput and scalability
- Batch MapReduce operations
- Stream processing/ analytics
- Machine learning

Bigtable stores data in massively scalable tables, each of which is a map from a single key to a wide column. Choose Bigtable when you have flat data that fits in one row per key, and when you need access latencies for your data to be in the millisecond range. Also, choose Bigtable when you need a database that scales linearly as you add more nodes to it. Cloud Bigtable can scale up to petabytes of data and smoothly handle millions of operations per second with latency of less than 10 milliseconds.

This makes it suitable for applications that require very high throughput and scalability for non-structured key-value data.
It is is an excellent storage engine solution for MapReduce operations, because it uses the same API for database access as Apache HBase.
It's also commonly used for stream processing and analytics and machine-learning applications.

You can use Cloud Bigtable for applications that store and query time-series data, marketing data, financial data, IoT data, and more.



Cloud Firestore automatically scales and replicates data to maintain high availability and durability

- User session management
- Real-time inventory
- User profiles
- Consistent state
- Persistent write-through cache
- Mobile application data

Like Cloud Bigtable, Cloud Firestore is a NoSQL database service. But it has some important differences. Cloud Firestore automatically scales and replicates your data to maintain high availability and durability. You can choose whether your data is replicated regionally or multi-regionally. It offers strong consistency guarantees for transactions. Because every field is automatically indexed, you can do fast queries.

You can use Cloud Firestore to set up real-time sessions for shopping carts and booking events, so you can provide the transactional compliance these applications require.

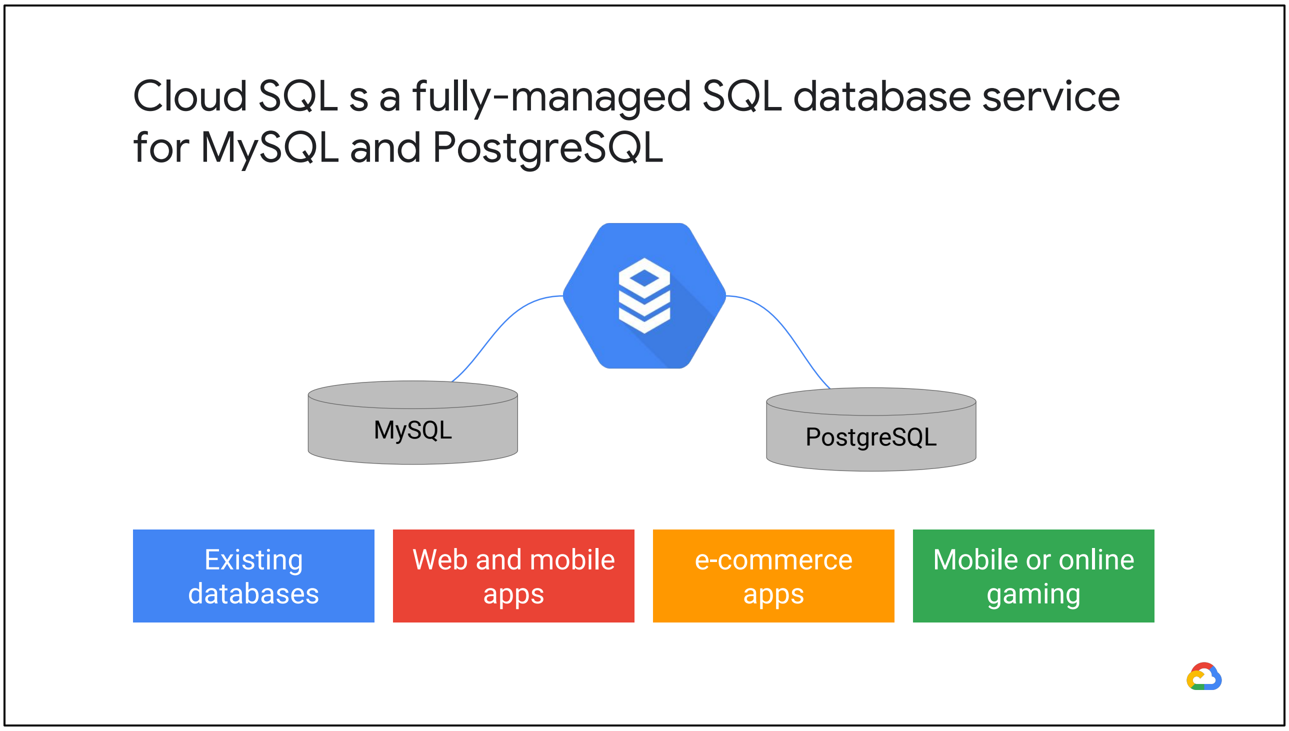
Cloud Firestore's data model is a document store. To understand what the word "Document" means in this context, think of a snippet of JSON data, containing fields and values, and fields can be nested. These fields don't always have to be the same for everything you store in the database. You can use Cloud Firestore to store non-homogeneous data without the need to limit yourself to a rigid schema. For instance, this makes it ideal if you’re storing diverse inventory data.

Suppose your application maintains profiles for each user. What if, six months after your application launches, you realize you need to start storing a new field for users? Cloud Firestore’s flexible schema lets you evolve the structure of the user profiles in your application as you add new features.

Cloud Firestore offers what are called “ACID transactions.” In essence, ACID compliance means that the data in your database stays internally consistent, because transactions either execute in their entirety or not at all. Because of Cloud Firestore’s ACID compliance, your application can maintain a consistent state even in large implementations. For example, suppose you have a large online gaming application. Cloud Firestore is ideal for ensuring that a consistent global game state is maintained for a massive number of concurrent player sessions.

You can use Cloud Firestore as a persistent write-through cache, providing a simple key-value store that prevents data loss and provides persistent state if your application fails.

Finally, Cloud Firestore is also a part of Firebase, Google’s mobile app platform with integrated, unified client libraries in various mobile programming language. Mobile app developers can use Firestore to store and synchronize app data at a global scale.



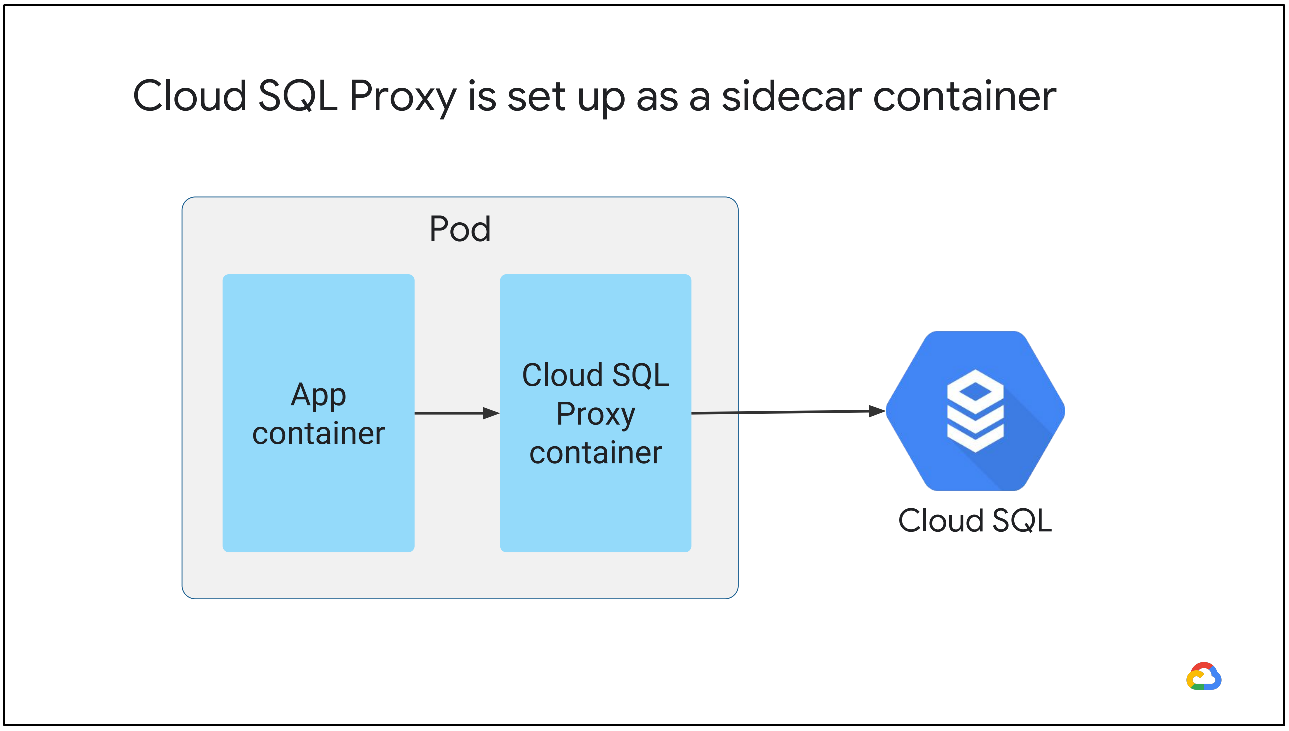
Cloud SQL s a fully-managed SQL database service for MySQL and PostgreSQL

Now I will turn away from NoSQL database services to SQL database services. For these services, you define a schema for your database, and the database engine enforces it to help you maintain your database.

Cloud SQL is a fully-managed SQL database service for MySQL and PostgreSQL. You could run these database services yourself in Compute Engine or Kubernetes Engine, but Cloud SQL is intended to save you work. GCP maintains, patches, and updates the database software. It manages data replication, and it performs backups for you. Cloud SQL also supports automatic failovers for high availability. Each of these capabilities are something you would otherwise have to do yourself.

You can use Cloud SQL as a direct replacement for existing MySQL or PostgreSQL database servers used by your applications. After you create a Cloud SQL instance, you can import a database dump from your existing server and simply re-configure your application to point to the Cloud SQL instance.

You can also use Cloud SQL in any application with relational database requirements, such as web and mobile applications e-commerce applications and mobile or online gaming.



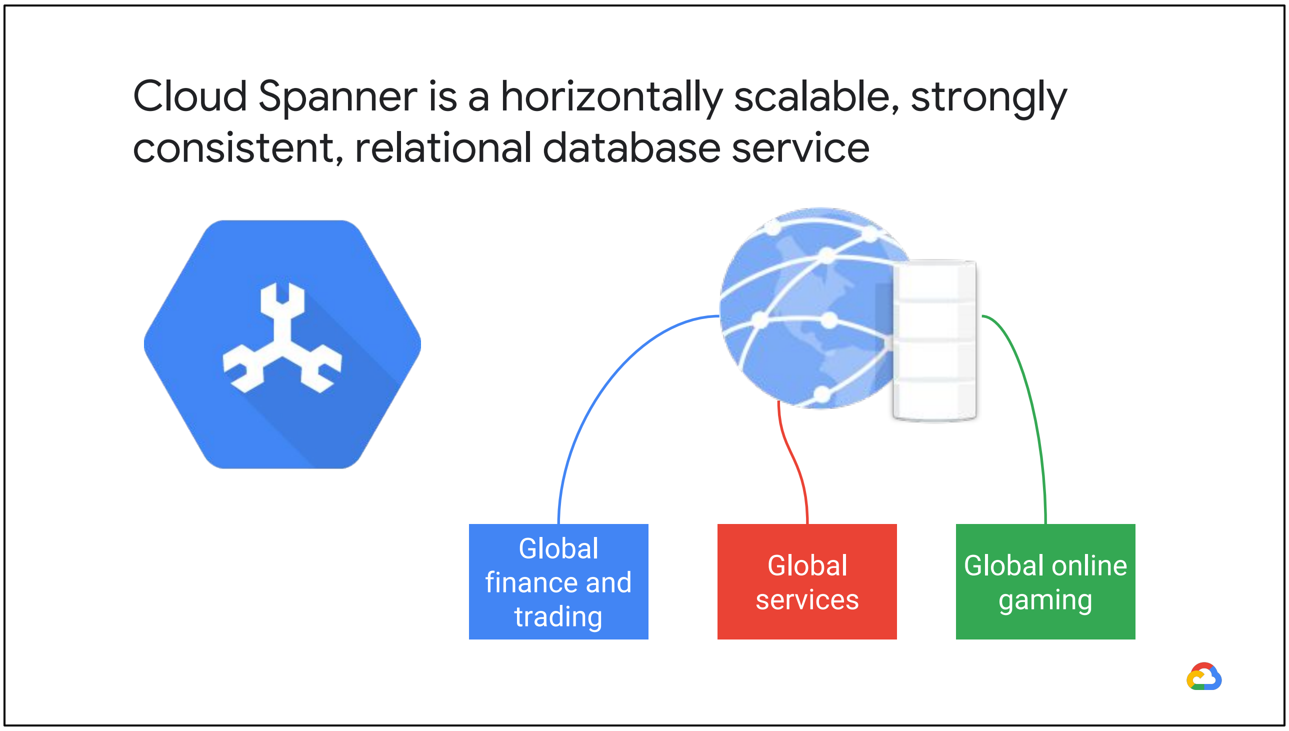
Cloud SQL Proxy is set up as a sidecar container

Your applications running in GKE access Cloud SQL using the Cloud SQL Proxy. That's a piece of software you add to your application to provide secure access to Cloud SQL database instances, with no need to configure SSL or whitelist IP addresses.

Cloud SQL Proxy automatically encrypts traffic in-transit and handles authentication with SQL. So even though the Pods that make up your application might be dynamic, they can all reliably access the database.

In GKE, Cloud SQL Proxy is set up as a "sidecar" container in the same Pod that contains your application. Your application communicates with the SQL Proxy container using the localhost network address.

The rest of the setup is similar to what you've seen before: enable the API, set up a Cloud IAM service account, and use Secrets to give credentials Pods. With Cloud SQL Proxy in place, your application can connect to Cloud SQL just like any other external application would.



Cloud Spanner is a horizontally scalable, strongly consistent, relational database service

Global finance and trading
Global services
Global online gaming

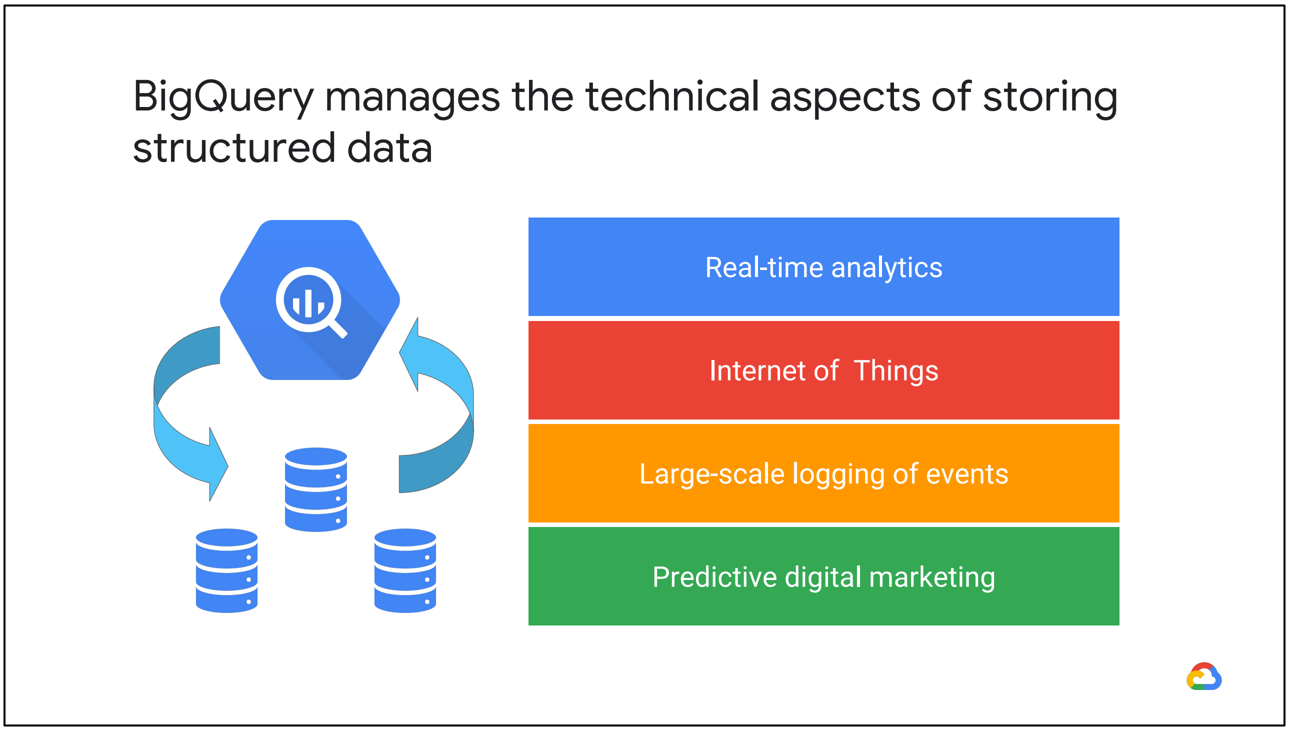
The last transactional GCP database service that we will discuss is Cloud Spanner. It’s like Cloud SQL in a number of ways, but it also has some important differences.

Like Cloud SQL, Cloud Spanner is a relational database service that offers ACID consistency guarantees for transactions. Unlike Cloud SQL, Cloud Spanner is globally distributed. Even if you choose to host a Cloud Spanner database across multiple GCP regions, it offers strong consistency guarantees for transactions. Cloud Spanner is semantically indistinguishable from a single-machine database. That property makes it ideal for applications with a wide geographic reach. Also unlike Cloud SQL, Cloud Spanner is horizontally scalable. If you need more storage or transactions-per-second capacity, you can simply add Cloud Spanner nodes to your existing instance. At the time this course was developed, the maximum capacity of a Cloud SQL database was a little more than 10 terabytes. But Cloud Spanner scales into the petabyte range.

In global financial trading applications, traditional database solutions rely on very complex infrastructure to ensure that the transactional integrity of trading is maintained at a global scale. But Cloud Spanner’s globally consistent data removes much of the complexity needed for these solutions.

Similarly, insurance, call center, supply-chain management, telecoms, logistics, and e-commerce businesses that operate applications or web services on a global scale are are a good match for Cloud Spanner.

Using Cloud Spanner, you can build highly interactive online gaming solutions on a global scale. Traditional database solutions often require significant trade-offs in order to ensure consistent state for players, which can prevent games from scaling beyond single servers or limited geographic regions.



BigQuery manages the technical aspects of storing structured data

- Real-time analytics
- Internet of Things
- Large-scale logging of events
- Predictive digital marketing

You’ve learned about GCP’s database services that are suitable for backing online applications. Now I’ll tell you about GCP’s data warehousing service.

Even though BigQuery supports SQL queries, it’s different from Google Cloud’s other relational database services in that it’s a columnar store. That means it’s optimized for analyzing data sets rather than for making row-by-row queries and changes. Running a data warehouse can be a lot of work. But BigQuery manages the technical aspects of storing structured data, including compression, encryption, replication, performance tuning, and scaling. Also, a traditional data warehouse can be expensive even when you’re not using it. But a key advantage of BigQuery is that your data storage is billed separately from any compute charges, and you only incur compute charges when you’re running a query.

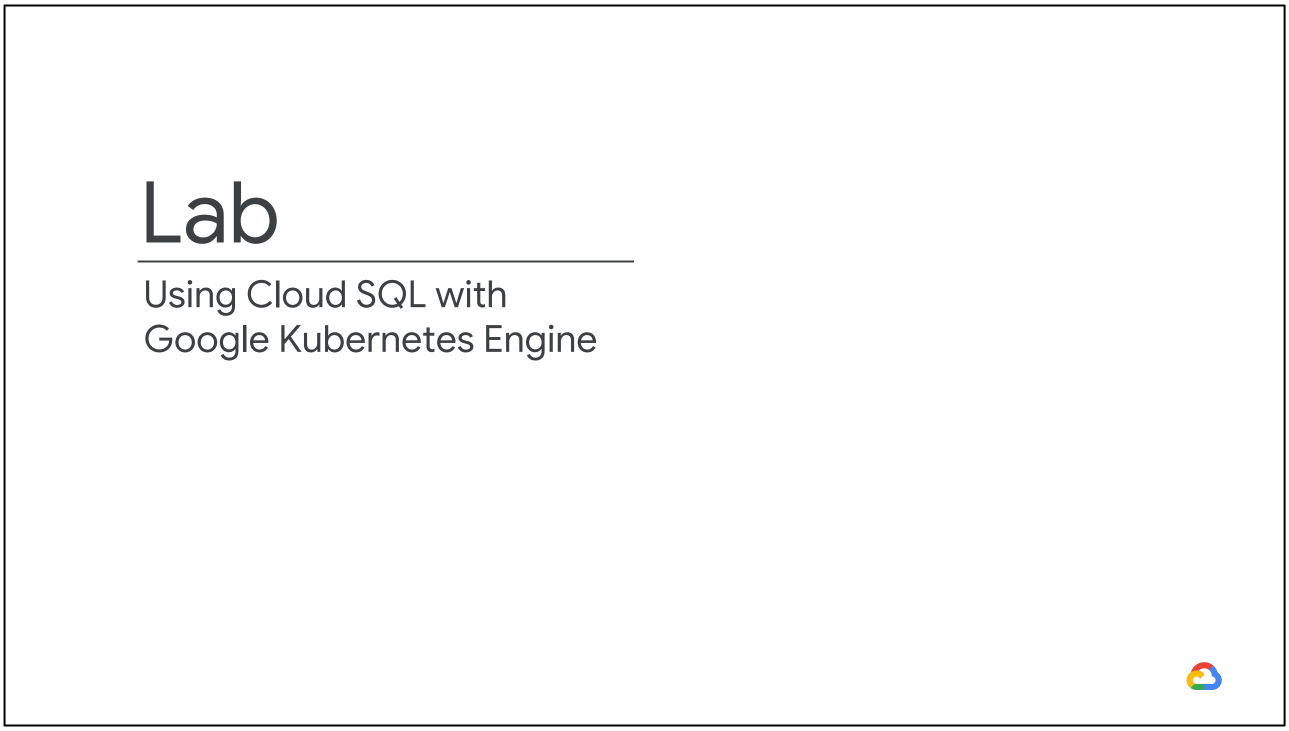
BigQuery provides a data warehousing backend for modern business intelligence solutions. It enables data integration, transformation, analysis, visualization, and reporting, using tools from Google and third parties.

A lot of people use BigQuery as part of real-time inventory, logistics, and supply chain applications. You can load incoming inventory and event stream data into BigQuery to provide the data warehousing functions and analytics.

BigQuery is ideal for storing the vast quantities of event and sensor data that IoT (Internet of Things) applications generate. This type of data tends to have the “write-once, read-many” access profile that BigQuery is optimized to handle. BigQuery can then perform ad-hoc analysis, run advanced analytics, or create additional derived data for reporting and analytics.

You can use BigQuery for applications that must consume and analyze extremely large volumes of historical or live streaming event data. BigQuery is also suitable if you have applications that must support logging of event data on a large scale. Its default quota for incoming streaming data is 100,000 rows per second. If that limit is not sufficient for your needs, just ask Google Cloud Support to increase your quota.

You can also use BigQuery to store and analyze user profile and activity data on a large scale for predictive digital-marketing applications.

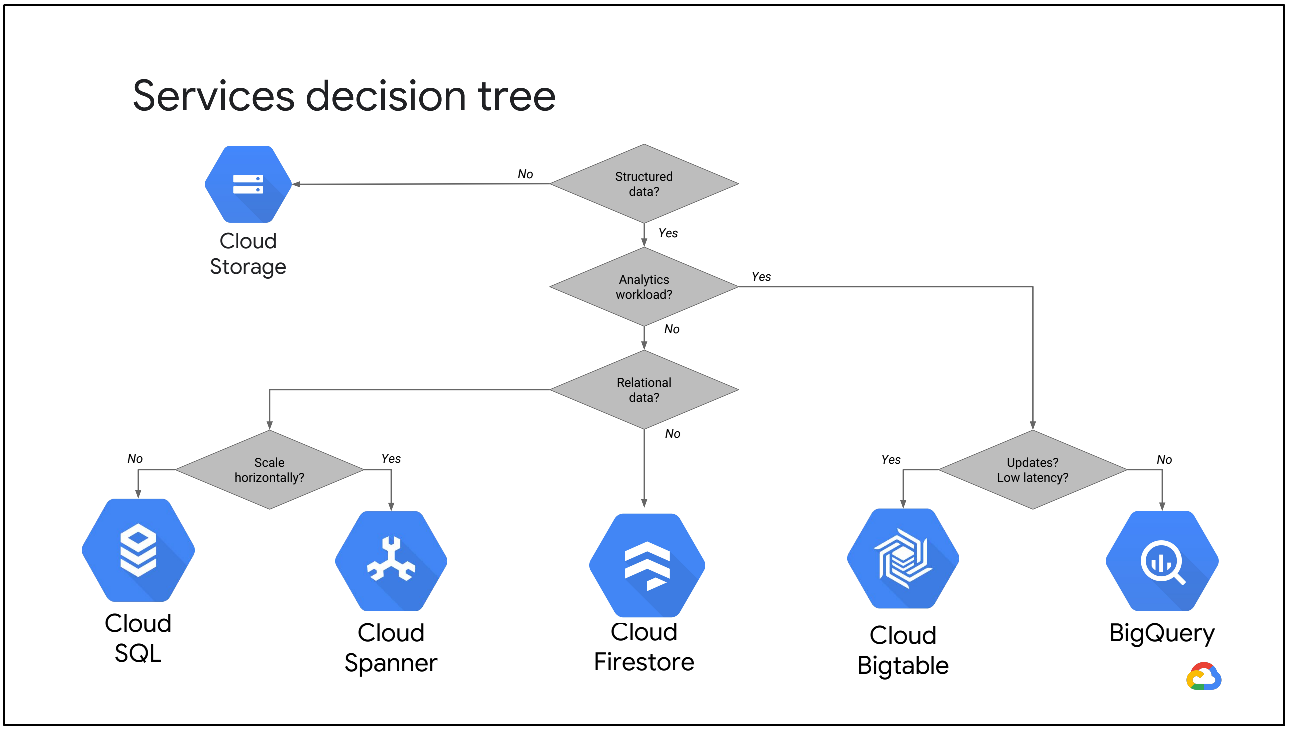


Lab

Using Cloud SQL with
Google Kubernetes Engine

In this lab, you'll set up a Kubernetes Deployment of WordPress connected to Cloud SQL via the SQL Proxy. The SQL Proxy lets you interact with a Cloud SQL instance as if it were installed locally (localhost:3306), and even though you're on an unsecured port locally, the SQL Proxy makes sure you're secure over the wire to your Cloud SQL Instance.

During this lab you'll create a GKE cluster, then you'll create a Cloud SQL Instance to connect to, followed by a Service Account to provide permission for your Pods to access the Cloud SQL Instance. Finally, you'll deploy WordPress on your GKE cluster, with the SQL Proxy as a Sidecar, connected to your Cloud SQL Instance.



You learned about many different storage and database services in this module. Maybe it seemed like a lot to remember. To help wrap up the topic, I'm going to share a simple decision tree that helps you decide among the services. Start by asking yourself about the characteristics of the data you want to store.

Is the data structured, or is it simply blobs?

If it's unstructured data, the object storage provided by Cloud Storage is probably the best choice.

But what if it's structured? In that case, ask yourself about the nature of the workload. Is it analytics-oriented? In other words, is it primary related to gathering insights into data, rather than to operations?

For now, let's assume your workload is not related to analytics. (I'll return to the case where it is in a minute.) Now ask yourself whether your data is relationally structured. In other words, is it naturally represented by one or more tables of columns and rows, with a unique key identifying each row?

If not, Cloud Firestore is a good choice. Its NoSQL orientation gives you flexibility to represent your data in the way that’s best for your application.

On the other hand, if your data is relational, now ask yourself whether you need to maintain the ability to scale horizontally in the future.

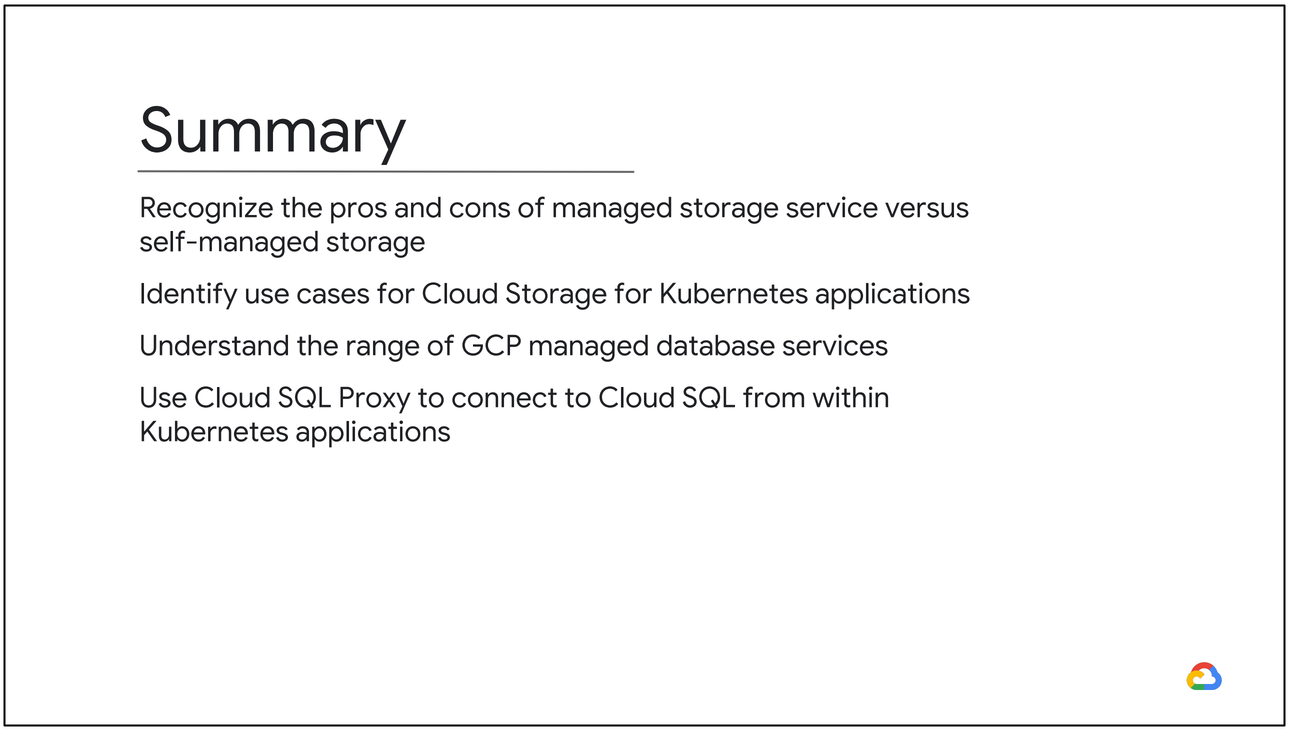
Horizontal scaling and global transaction consistency are the key benefit of Cloud Spanner.

If you don’t anticipate your database growing beyond the terabyte range, and if classic scaling strategies for relational databases like read replicas should be enough to help you meet your performance goals, Cloud SQL is a good choice.

Now let’s return to a decision point we left behind. What if your workload is related to analytics? Ask yourself whether you need your database service to offer SQL-based updates and low-latency reads and writes of individual data items.

If it does, Cloud Bigtable is a good choice.

Otherwise, assuming your analytics workloads are more oriented towards drawing conclusions based on streamed data, BigQuery is a good fit.



Summary

Recognize the pros and cons of managed storage service versus self-managed storage

Identify use cases for Cloud Storage for Kubernetes applications

Understand the range of GCP managed database services

Use Cloud SQL Proxy to connect to Cloud SQL from within Kubernetes applications

That concludes the “Using GCP Managed Storage Services with Google Kubernetes Engine” module. In this module, you learned how to recognize the pros and cons of managed storage services versus self-managed containerized storage, identify use cases for Cloud Storage for applications running in a Kubernetes cluster, identify the range of GCP managed database services, and how to use Cloud SQL proxy to simplify the task of connecting to Cloud SQL from within Kubernetes applications.

cloud.google.com