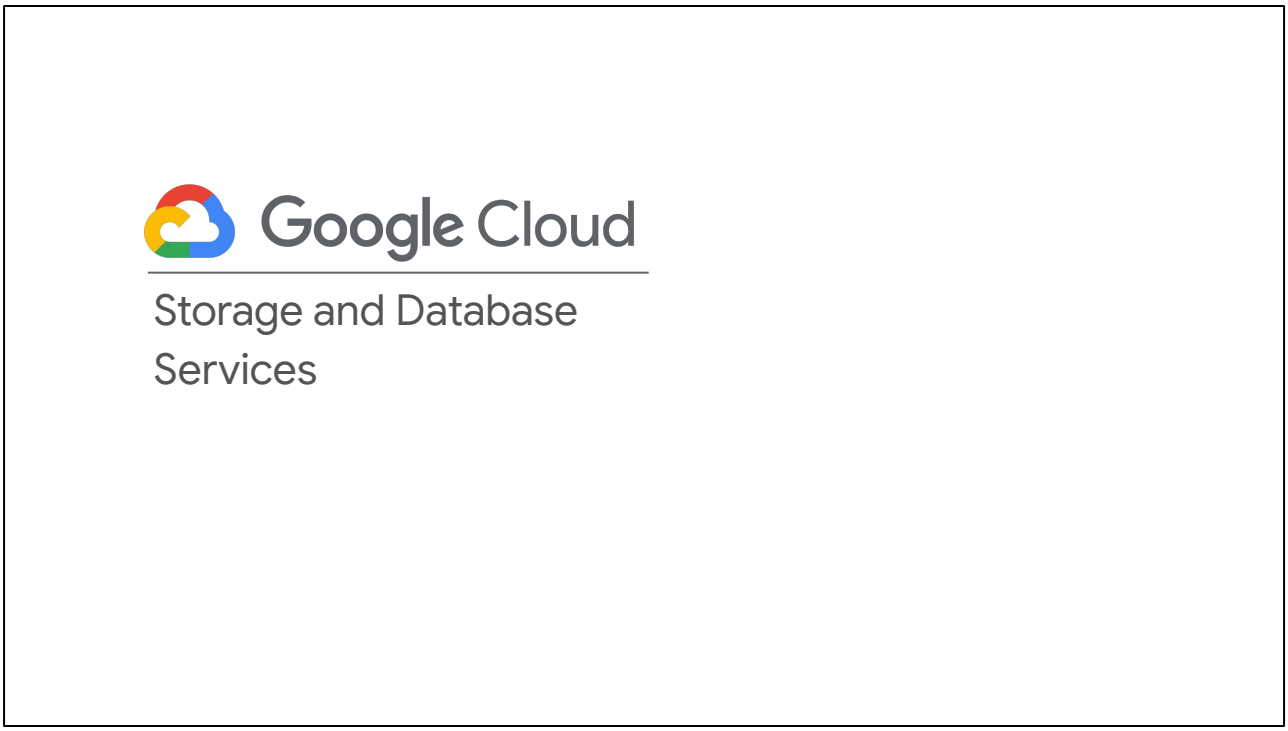
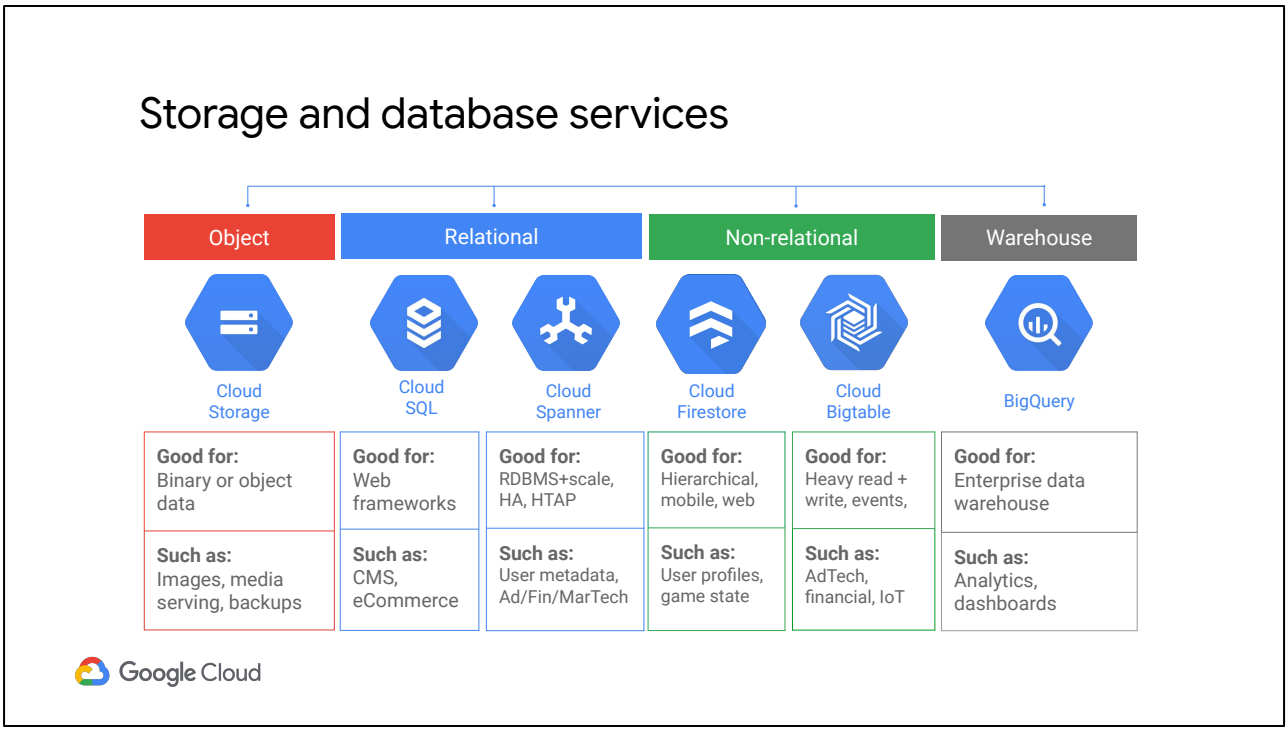


In this module, we cover storage and database services in GCP. Every application needs to store data, whether it's business data, media to be streamed, or sensor data from devices.

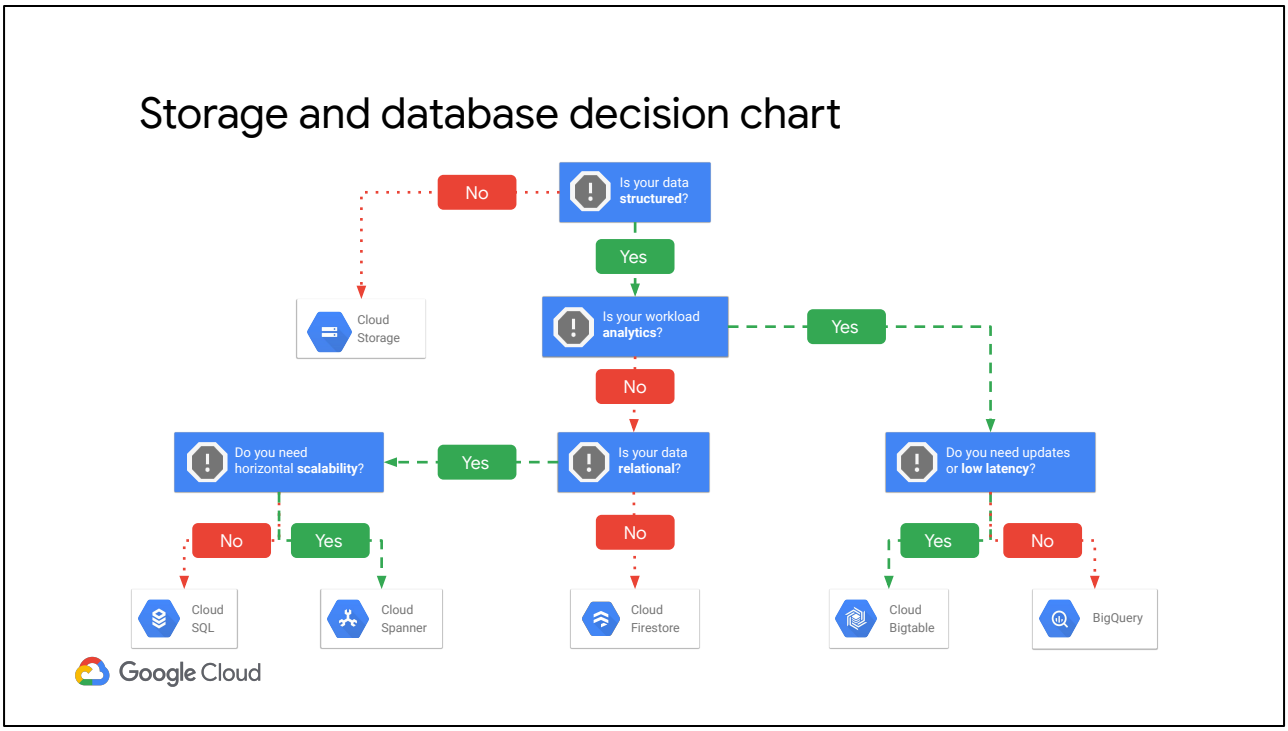
From an application-centered perspective, the technology stores and retrieves the data. Whether it's a database or an object store is less important than whether that service supports the application's requirements for efficiently storing and retrieving the data, given its characteristics.

Google offers several data storage services to choose from. In this module, we will cover Cloud Storage, Cloud SQL, Cloud Spanner, Cloud Firestore, and Cloud Bigtable. Let me start by giving you a high-level overview of these different services.



This table shows the storage and database services and highlights the storage service type (relational, non-relational, object and data warehouse), what each service is good for, and intended use.

BigQuery is also listed on the right. I'm mentioning this service because it sits on the edge between data storage and data processing. You *can* store data in BigQuery, but the intended use for BigQuery is big data analysis and interactive querying. For this reason, BigQuery is covered later in the series.

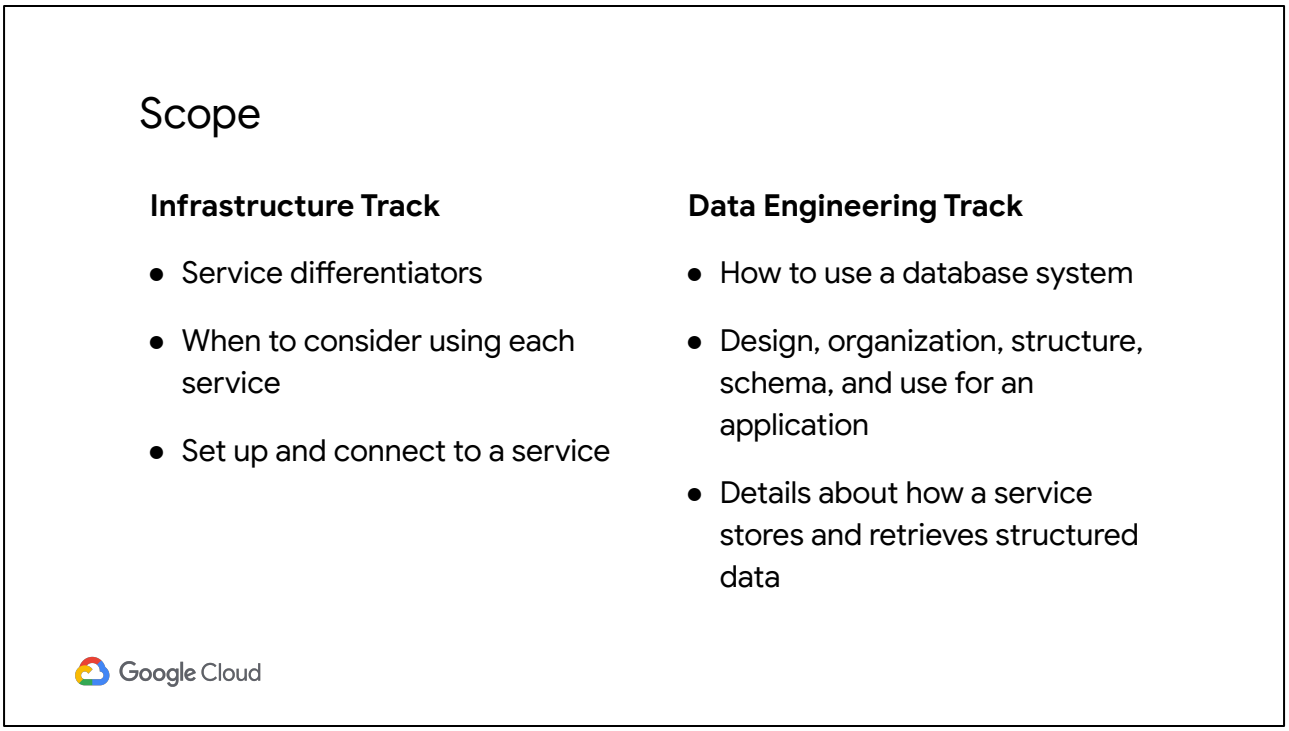


If tables aren't your preference, I also added this decision tree to help you identify the solution that best fits your application.

Let's walk through this together:

- First, ask yourself: Is your data structured? If it's not, choose Cloud Storage.
- If your data is structured, does your workload focus on analytics? If it does, you will want to choose Cloud Bigtable or BigQuery, depending on your latency and update needs.
- Otherwise, check whether your data is relational. If it's not relational, choose Cloud Firestore.
- If it is relational, you will want to choose Cloud SQL or Cloud Spanner, depending on your need for horizontal scalability.

Depending on your application, you might use one or several of these services to get the job done. For more information on how to choose between these different services, see the links section of this video [https://cloud.google.com/storage-options/, https://cloud.google.com/products/databases/]



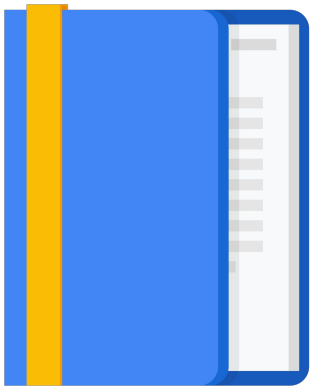
Before we dive into each of the data storage services, let's define the scope of this module.

The purpose of this module is to explain which services are available and when to consider using them from an infrastructure perspective. I want you to be able to set up and connect to a service without detailed knowledge of how to use a database system.

If you want a deeper dive into the design, organizations, structures, schemas and details on how data can be optimized, served and stored properly within those different services, I recommend Google Cloud's Data Engineering courses.

Agenda

Cloud Storage
Lab
Cloud SQL
Lab
Cloud Spanner
Cloud Firestore
Cloud Bigtable
Cloud Memorystore



Google Cloud

Let’s look at the agenda. This module covers all of the services we have mentioned so far. To become more comfortable with these services, you will apply them in two labs.

I’ll also provide a quick overview of Cloud Memorystore, which is GCP’s fully managed Redis service.

Let’s get started by diving into Cloud Storage!

Cloud Storage is an object storage service

- Use cases:
  - Website content
  - Storing data for archiving and disaster recovery
    - Distributing large data objects to users via direct download
- Scalable to exabytes
- Time to first byte in milliseconds
- Very high availability across all storage classes
- Single API across storage classes

Google Cloud

Cloud Storage is GCP's object storage service, and it allows world-wide storage and retrieval of any amount of data at any time. You can use Cloud Storage for a range of scenarios including serving website content, storing data for archival and disaster recovery, or distributing large data objects to users via direct download. Cloud Storage has a couple of key features:

- It's scalable to exabytes of data
- The time to first byte is in milliseconds
- It has very high availability across all storage classes
- And It has a single API across those storage classes

Some like to think of Cloud Storage as files in a file system but it's not really a file system. Instead, Cloud Storage is a collection of buckets that you place objects into. You can create directories, so to speak, but really a directory is just another object that points to different objects in the bucket. You're not going to easily be able to index all of these files like you would in a file system. You just have a specific URL to access objects.

Overview of storage classes

	Regional	Multi-Regional	Nearline	Coldline
Design patterns	Data that is **used in one region** or needs to remain in region	Data that is **used globally** and has no regional restrictions	Data that is accessed **no more than once a month**	Data that is accessed **no more than once a year**
Use case	Data-intensive computations, data governance	Website content, interactive workloads	Backup, long-tail multimedia content	Archiving or disaster recovery
Availability SLA	99.9%	99.95%	99.0%	99.0%
Durability	99.999999999%	99.999999999%	99.999999999%	99.999999999%
Duration	Hot data	Hot data	30-day minimum	90-day minimum
Retrieval cost	none	none	$	$$

Google Cloud

Cloud Storage has four storage classes: Regional, Multi-regional, Nearline, and Coldline.

- **Regional** Storage enables you to store data at lower cost, with the trade-off of data being stored in a specific regional location, instead of having redundancy distributed over a large geographic area. I recommend using regional storage when storing frequently accessed data in the same region as your Compute Engine instances. This provides you with better performance for data-intensive computations. You should also choose regional storage for data governance reasons - like if your data needs to remain in a specific region.
- **Multi-Regional** Storage, on the other hand, is geo-redundant, which means Cloud Storage stores your data redundantly in at least two geographic locations separated by at least 100 miles within the multi-regional location of the bucket. Multi-Regional Storage can be placed only in multi-regional locations, such as the United States, the European Union, or Asia. Multi-Regional Storage is appropriate for storing data that is frequently accessed, such as serving website content, interactive workloads, or data supporting mobile and gaming applications.
- **Nearline** storage is a low-cost, highly durable storage service for storing infrequently accessed data. This storage class is a good choice when you plan to read or modify your data less than once a month because the storage cost

- is low, but there is an associated retrieval cost. For example, if you want to continuously add files to Cloud Storage and plan to access those files once a month for analysis, Nearline Storage is a great choice. I also recommend using Nearline Storage for backups and serving long-tail multimedia content.
- **Coldline** Storage is a very-low-cost, highly durable storage service for data archival, online backup, and disaster recovery. However, unlike other "cold" storage services, your data is available within milliseconds, not hours or days. Coldline Storage is the best choice for data that you plan to access at most once a year, due to its lower storage cost but higher retrieval cost. You might use Coldline Storage if you want to archive data or have access in the event of a disaster recovery event.

Let’s focus on durability and availability. All of these storage classes have 11 nines of durability, but what does that mean? Does that mean you have access to your files at all times? No, what that means is you won't lose data. You may not be able to access the data which is like going to your bank and saying well my money is in there, it's 11 nines durable. But when the bank is closed I don't have access to it which is the availability that differs between the storage classes.

Cloud Storage overview

Buckets

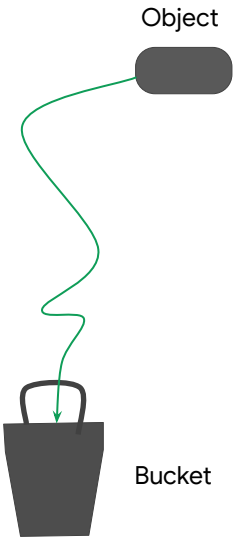
- Naming requirements
- Cannot be nested

Objects

- Inherit storage class of bucket when created
- No minimum size; unlimited storage

Access

- `gsutil` command
- (RESTful) JSON API or XML API

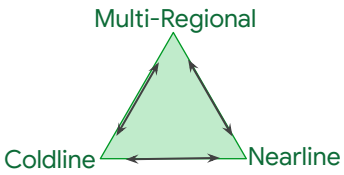


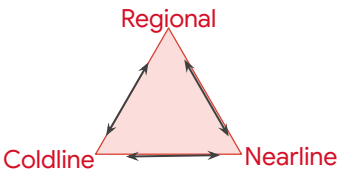
Google Cloud

Cloud Storage is broken down into a couple of different items here.

- First of all, there are buckets which are required to have a globally unique name and cannot be nested.
- The data that you put into those buckets are objects that inherit the storage class of the bucket and those objects could be text files, doc files, video files, etc. There is no minimum size to those objects and you can scale this as much as you want as long as your quota allows it.
- To access the data, you can use the gsutil command, or either the JSON or XML APIs.

Changing default storage classes





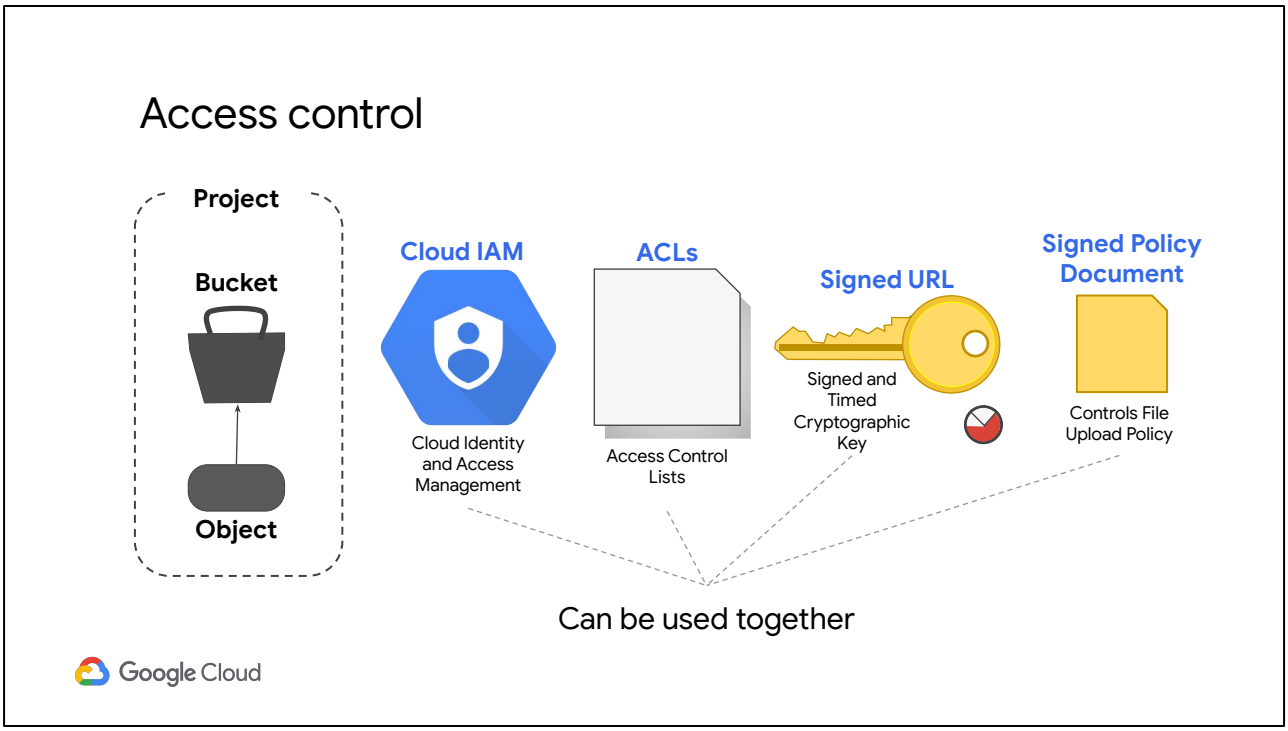
- Default class is applied to new objects
- Regional bucket can never be changed to Multi-Regional
- Multi-Regional bucket can never be changed to Regional
- Objects can be moved from bucket to bucket
- Object Lifecycle Management can manage the classes of objects

Google Cloud

When you upload an object to the bucket, if you don't specify a storage class for the object, the object is assigned the bucket's storage class. You can change the default storage class of a bucket but you can't change a regional bucket to a multi-regional, and vice versa. As the slides illustrates, both multi-regional buckets and regional buckets can be changed to coldline or nearline.

When you upload an object, you can specify a storage class for it. You can also change the storage class of an object that already exists in your bucket without moving the object to a different bucket or changing the URL to the object. Setting a per-object storage class is useful, for example, if you have objects in your bucket that you want to keep, but that you don't expect to access frequently. In this case, you can minimize costs by changing the storage class of those specific objects to Nearline Storage or Coldline Storage.

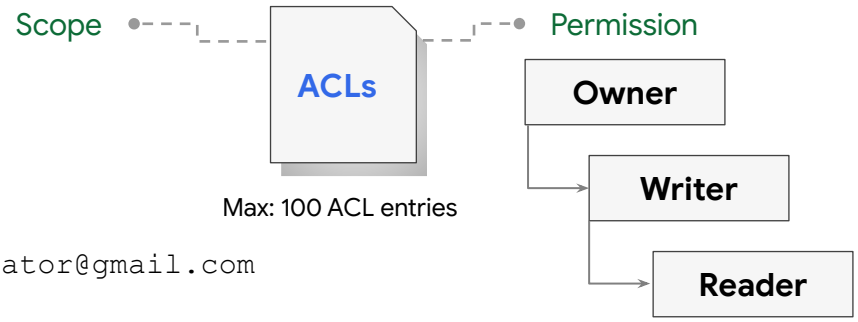
In order to help manage the classes of objects in your bucket, Cloud Storage offers Object Lifecycle Management. More on that later.



Let's look at access control for your objects and buckets that are part of a project.

- We can use IAM for the project to control which individual user or service account can see the bucket, list the objects in the bucket, view the names of the objects in the bucket, or create new buckets. For most purposes, Cloud IAM is sufficient, and roles are inherited from project to bucket to object.
- Access control lists or ACLs offer finer control.
- For even more detailed control, signed URLs provide a cryptographic key that gives time-limited access to a bucket or object.
- Finally, a signed policy document further refines the control by determining what kind of file can be uploaded by someone with a signed URL. Let's take a closer look at ACLs and signed URLs.

Access control lists (ACLs)



Examples:

- `collaborator@gmail.com`
- `allUsers`
- `allAuthenticatedUsers`

Google Cloud

An ACL is a mechanism you can use to define who has access to your buckets and objects, as well as what level of access they have. The maximum number of ACL entries you can create for a bucket or object is 100.

Each ACL consists of one or more entries, and these entries consist of two pieces of information:

- A scope, which defines who can perform the specified actions (for example, a specific user or group of users).
- And a permission, which defines what actions can be performed (for example, read or write).

The allUsers identifier listed on this slide represents anyone who is on the internet, with or without a Google account. The allAuthenticatedUsers identifier, in contrast, represents anyone who is authenticated with a Google account.

For more information on ACLs, refer to the links section of this video [https://cloud.google.com/storage/docs/access-control/lists]

Signed URLs

- "Valet key" access to buckets and objects via ticket:
  - Ticket is a cryptographically signed URL
  - Time-limited
  - Operations specified in ticket: HTTP GET, PUT, DELETE (not POST)
  - Any user with URL can invoke permitted operations
- Example using private account key and gsutil:

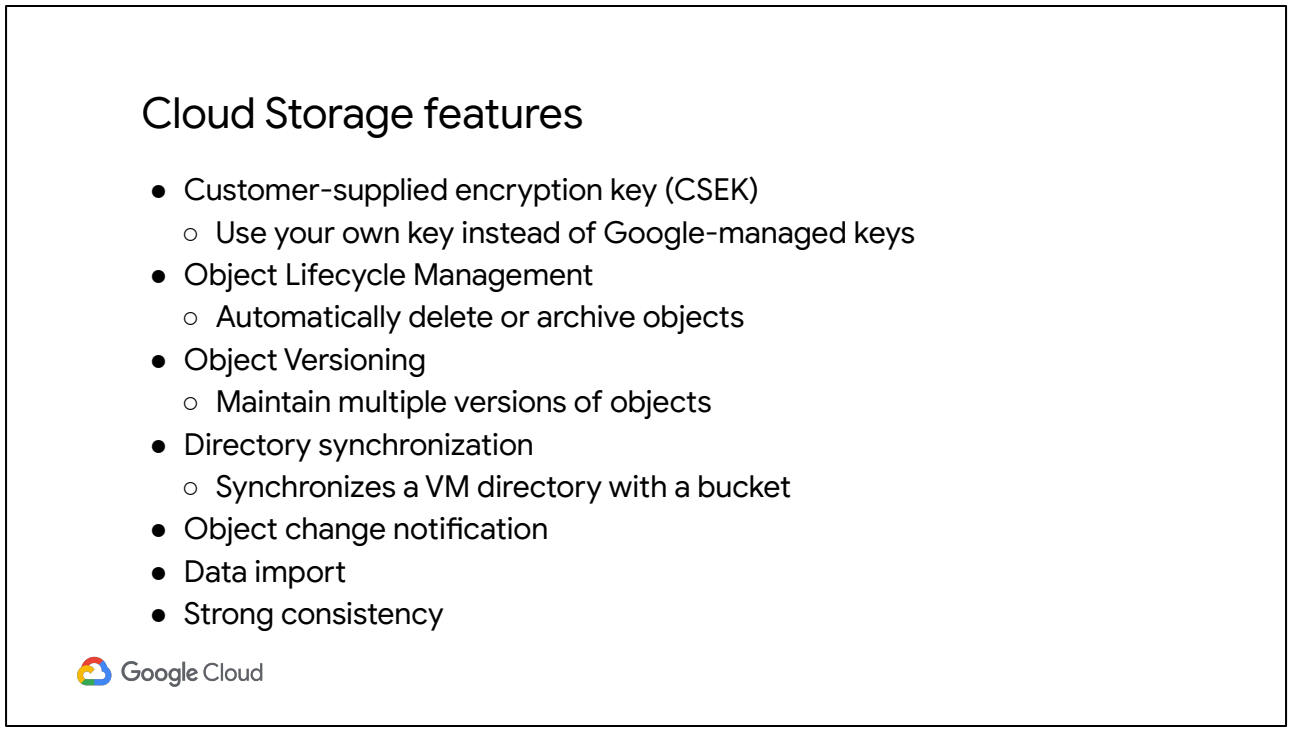
```
gsutil signurl -d 10m path/to/privatekey.p12
gs://bucket/object
```

Google Cloud

For some applications, it is easier and more efficient to grant limited-time access tokens that can be used by any user, instead of using account-based authentication for controlling resource access. (For example, when you don't want to require users to have Google accounts).

Signed URLs allow you to do this for Cloud Storage. You create a URL that grants read or write access to a specific Cloud Storage resource and specifies when the access expires. That URL is signed using a private key associated with a service account. When the request is received, Cloud Storage can verify that the access-granting URL was issued on behalf of a trusted security principal, in this case the service account, and delegates its trust of that account to the holder of the URL.

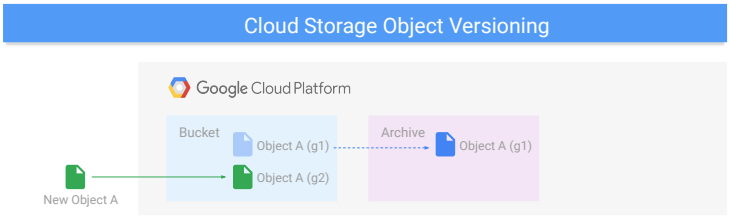
After you give out the signed URL, it is out of your control. So you want the signed URL to expire after some reasonable amount of time.



There are also several features that come with Cloud Storage. I will cover these at a high-level for now because we will soon dive deeper into some of them.

- Earlier in the course series, we already talked a little about Customer-supplied encryption keys when attaching persistent disks to virtual machines. This allows you to supply your own encryption keys instead of the Google-managed keys, which is also available for Cloud Storage.
- Cloud Storage also provides Object Lifecycle Management which lets you automatically delete or archive objects.
- Another feature is object versioning which allows you to maintain multiple version of objects in your bucket. You are charged for the versions as if they were multiple files, which is something to keep in mind.
- Cloud Storage also offers directory synchronization so that you can sync a VM directory with a bucket.
- We will discuss Object Change Notification, data import and strong consistency in more detail after going into object versioning and object lifecycle management.

Object Versioning supports the retrieval of objects that are deleted or overwritten



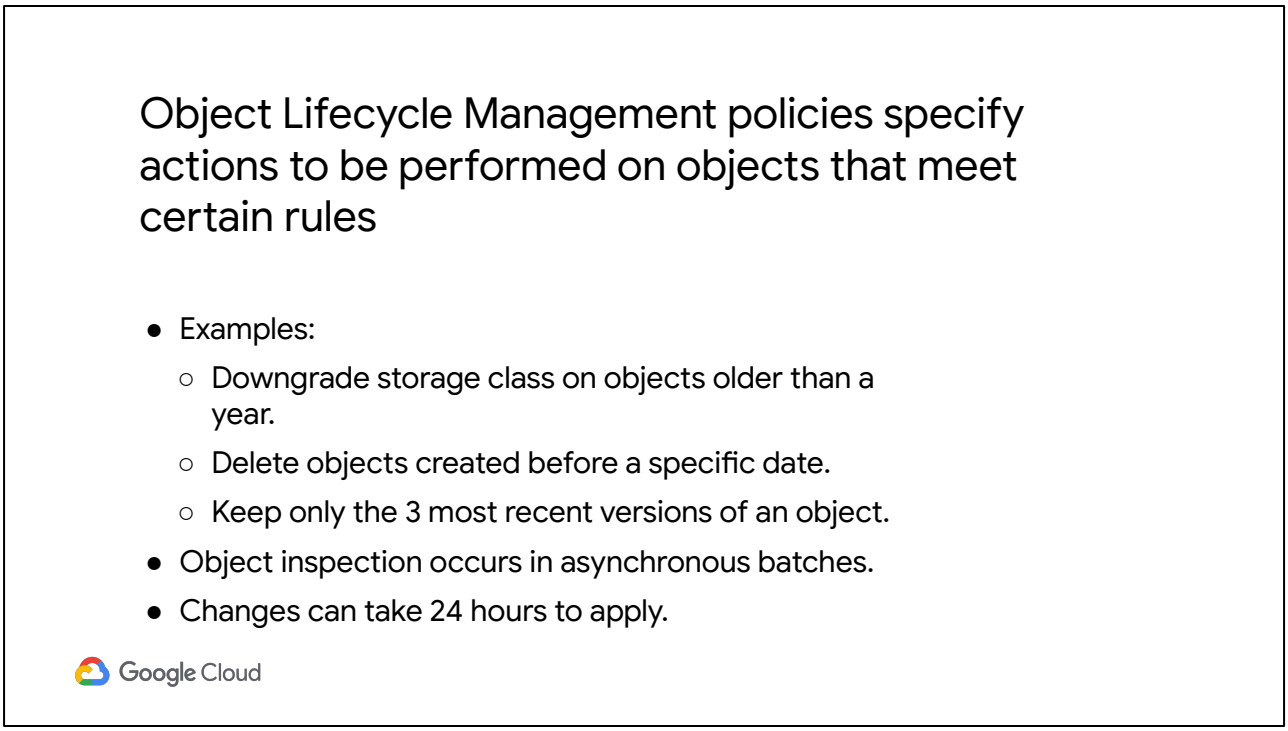
- Objects are immutable.
- Object Versioning:
  - Maintain a history of modifications of objects.
  - List archived versions of an object, restore an object to an older state, or delete a version.

Google Cloud

In Cloud Storage, objects are immutable, which means that an uploaded object cannot change throughout its storage lifetime. To support the retrieval of objects that are deleted or overwritten, Cloud Storage offers the Object Versioning feature. Object Versioning can be enabled for a bucket. Once enabled, Cloud Storage creates an archived version of an object each time the live version of the object is overwritten or deleted. The archived version retains the name of the object but is uniquely identified by a generation number as illustrated on this slide by g1.

When Object Versioning is enabled, you can list archived versions of an object, restore the live version of an object to an older state, or permanently delete an archived version, as needed. You can turn versioning on or off for a bucket at any time. Turning versioning off leaves existing object versions in place and causes the bucket to stop accumulating new archived object versions.

For more information on Object Versioning, refer to the links section of this video: https://cloud.google.com/storage/docs/object-versioning



To support common use cases like setting a Time to Live for objects, archiving older versions of objects, or "downgrading" storage classes of objects to help manage costs, Cloud Storage offers Object Lifecycle Management.

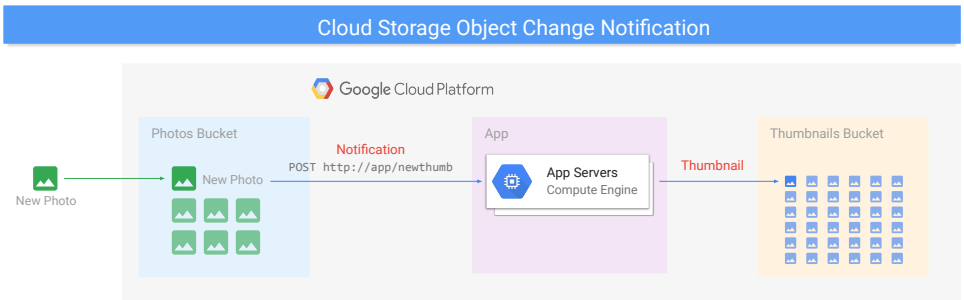
You can assign a lifecycle management configuration to a bucket. The configuration is a set of rules that apply to all the objects in the bucket. So when an object meets the criteria of one of the rules, Cloud Storage automatically performs a specified action on the object. Here are some example use cases:

- Downgrade the storage class of objects older than a year to Coldline Storage.
- Delete objects created before a specific date. For example, January 1, 2017.
- Keep only the 3 most recent versions of each object in a bucket with versioning enabled.

Object inspection occurs in asynchronous batches, so rules may not be applied immediately. Also, updates to your lifecycle configuration may take up to 24 hours to go into effect. This means that when you change your lifecycle configuration, Object Lifecycle Management may still perform actions based on the old configuration for up to 24 hours. So keep that in mind.

For more information on Object Lifecycle Management, refer to the links section of this video: https://cloud.google.com/storage/docs/lifecycle

Object change notification can be used to notify an application when an object is updated or added to a bucket



Recommended: Cloud Pub/Sub Notifications for Cloud Storage

Google Cloud

Object Change Notification can be used to notify an application when an object is updated or added to a bucket through a watch request.

Completing a watch request creates a new notification channel. The notification channel is the means by which a notification message is sent to an application watching a bucket. As of this recording, the only type of notification channel supported is a webhook.

After a notification channel is initiated, Cloud Storage notifies the application any time an object is added, updated, or removed from the bucket. For example, as shown here, when you add a new picture to a bucket, an application can be notified to create a thumbnail.

However, Cloud Pub/Sub notifications are the recommended way to track changes to objects in your Cloud Storage buckets because they're faster, more flexible, easier to set up, and more cost-effective. Cloud Pub/Sub is Google's distributed real-time messaging service, which is covered in the Developing Applications track.

Data import services

- **Transfer Appliance:** Rack, capture and then ship your data to GCP.
- **Storage Transfer Service:** Import online data (another bucket, an S3 bucket, or web source).
- **Offline Media Import:** Third-party provider uploads the data from physical media.

Google Cloud

The GCP Console allows you to upload individual files to your bucket. What if you have to upload terabytes or even petabytes of data? There are three services that address this: Transfer Appliance, Storage Transfer Service, and Offline Media Import.

- Transfer Appliance is a hardware appliance you can use to securely migrate large volumes of data (from hundreds of terabytes up to 1 petabyte) to GCP without disrupting business operations. The images on this slide are transfer appliances.
- The Storage Transfer Service enables high-performance imports of online data. That data source can be another Cloud Storage bucket, an Amazon S3 bucket, or an HTTP/HTTPS location.
- Finally, Offline Media Import is a third party service where physical media (such as storage arrays, hard disk drives (HDDs), tapes, and USB flash drives) is sent to a provider who uploads the data.

For more information on these three services, refer to the links section of this video:
https://cloud.google.com/transfer-appliance/
https://cloud.google.com/storage-transfer/docs/
https://cloud.google.com/storage/docs/offline-media-import-export

Cloud Storage provides strong global consistency

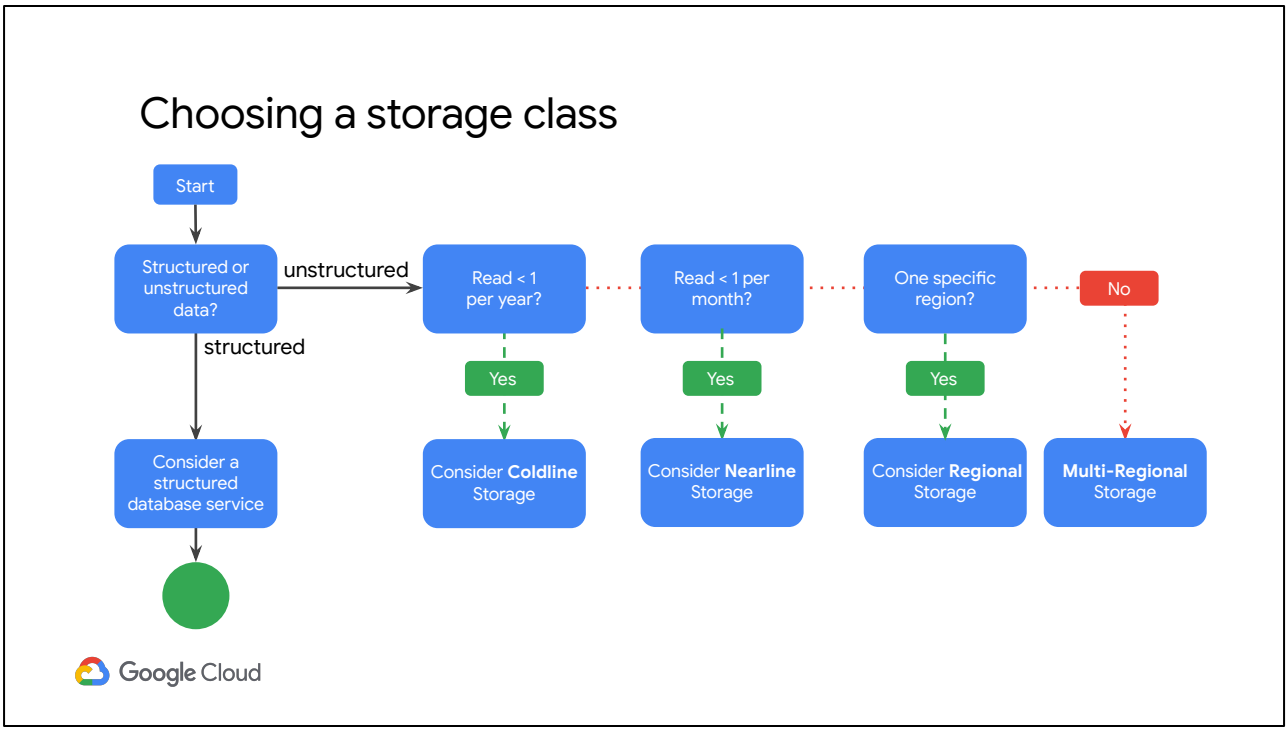
- Read-after-write
- Read-after-metadata-update
- Read-after-delete
- Bucket listing
- Object listing
- Granting access to resources

Google Cloud

When you upload an object to Cloud Storage and you receive a success response, the object is immediately available for download and metadata operations from any location where Google offers service. This is true whether you create a new object or overwrite an existing object. Because uploads are strongly consistent, you will never receive a 404 Not Found response or stale data for a read-after-write or read-after-metadata-update operation.

Strong global consistency also extends to deletion operations on objects. If a deletion request succeeds, an immediate attempt to download the object or its metadata will result in a 404 Not Found status code. You get the 404 error because the object no longer exists after the delete operation succeeds.

Bucket listing is strongly consistent. For example, if you create a bucket, then immediately perform a list buckets operation, the new bucket appears in the returned list of buckets.
Finally, object listing is also strongly consistent. For example, if you upload an object to a bucket and then immediately perform a list objects operation, the new object appears in the returned list of objects.



Let's explore the decision tree to help you find the appropriate storage class in Cloud Storage.

- If you will read your data less than once a year, you should consider using **Coldline storage.**
- If you will read your data less than once a month, you should consider using **Nearline storage**.
- And if you will be doing reads and writes more often than that, you should consider choosing **Multi-Regional or Regiona**l, depending on your locality needs.

So far we have only considered unstructured data. Structured data services are covered next.

Lab

Cloud Storage

Google Cloud

Let’s take some of the Cloud Storage concepts that we just discussed and apply them in a lab.

In this lab, you'll create buckets and perform many of the advanced options available in Cloud Storage. You'll set access control lists to limit who can have access to your data and what they're allowed to do with it. You'll use the ability to supply and manage your own encryption keys for additional security. You'll enable object versioning to track changes in the data, and you'll configure lifecycle management so that objects are automatically archived or deleted after a specified period. Finally, you’ll use the directory synchronization feature that I mentioned and share your buckets across projects using Cloud IAM.

Lab review

Cloud Storage

Google Cloud

In this lab, you learned to create and work with buckets and objects, and applied the following Cloud Storage features:

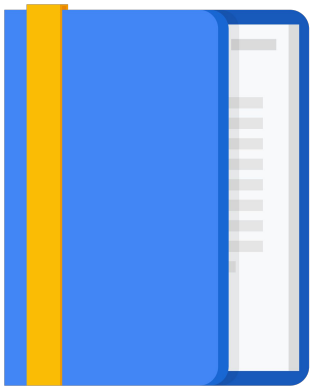
- Customer-supplied encryption keys
- Access control lists
- Lifecycle management
- Object versioning
- Directory synchronization
- And cross-project resource sharing using IAM

Now that you're familiar with many of the advanced features of Cloud Storage, you might consider using them in a variety of applications that you might not have previously considered. A common, quick, and easy way to start using GCP, is to use Cloud Storage as a backup service.

You can stay for a lab walkthrough, but remember that GCP's user interface can change, so your environment might look slightly different.

Agenda

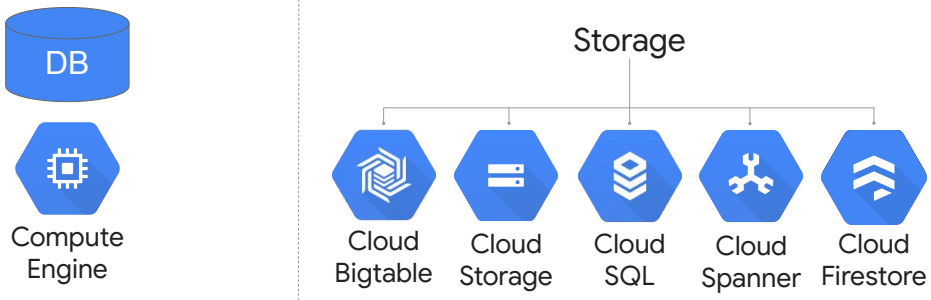
- Cloud Storage
- Lab
- **Cloud SQL**
- Lab
- Cloud Spanner
- Cloud Firestore
- Cloud Bigtable
- Cloud Memorystore



Google Cloud

Let’s dive into the structured or relational database services. First up is Cloud SQL.

Build your own database solution or use a managed service



Google Cloud

Why would you use a GCP service for SQL, when you can install a SQL Server application image on a VM using Compute Engine?

The question really is, should you build your own database solution or use a managed service? There are benefits to using a managed service, so let's learn about why you'd use Cloud SQL as a managed service inside of GCP.

Cloud SQL is a fully managed database service (MySQL or PostgreSQL)

- Patches and updates automatically applied
- You administer MySQL users
- Cloud SQL supports many clients
  - `gcloud sql`
  - App Engine, G Suite scripts
  - Applications and tools
    - SQL Workbench, Toad
    - External applications using standard MySQL drivers

Cloud SQL

Google Cloud

Cloud SQL is a fully managed service of either MySQL or PostgreSQL databases. This means that patches and updates are automatically applied but you still have to administer MySQL users with the native authentication tools that come with these databases.

Cloud SQL supports many clients, such as Cloud Shell, App Engine and G Suite scripts. It also supports other applications and tools that you might be used to like SQL Workbench, Toad and other external applications using standard MySQL drivers.

Cloud SQL instance

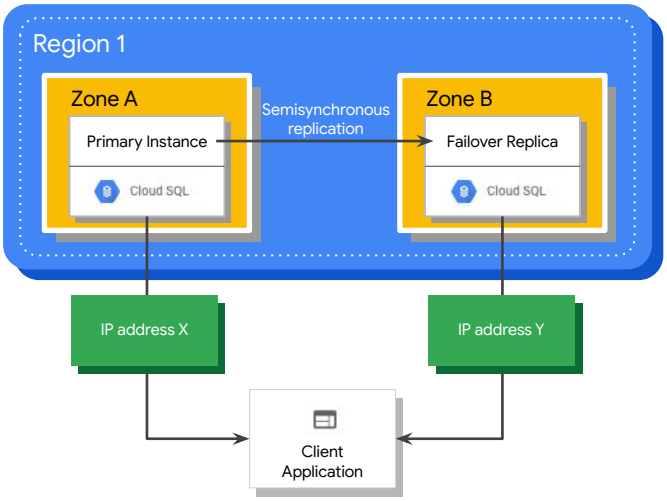
- Performance:
  - 30 TB of storage
  - 40,000 IOPS
  - 416 GB of RAM
  - Scale out with read replicas
- Choice:
  - MySQL 5.6 or 5.7
  - PostgreSQL 9.6 or 11.1

Google Cloud

Cloud SQL delivers high performance and scalability with up to 30 TB of storage capacity, 40,000 IOPS, and 416 GB of RAM per instance. You can easily scale up to 64 processor cores and scale out with read replicas.
You can use Cloud SQL with either MySQL 5.6 or 5.7 or PostgreSQL 6.9 or 11.1 as of this recording.

Cloud SQL services

- Replica services
- Backup service
- Import/export
- Scaling
  - Up: Machine capacity
  - Out: Replicas

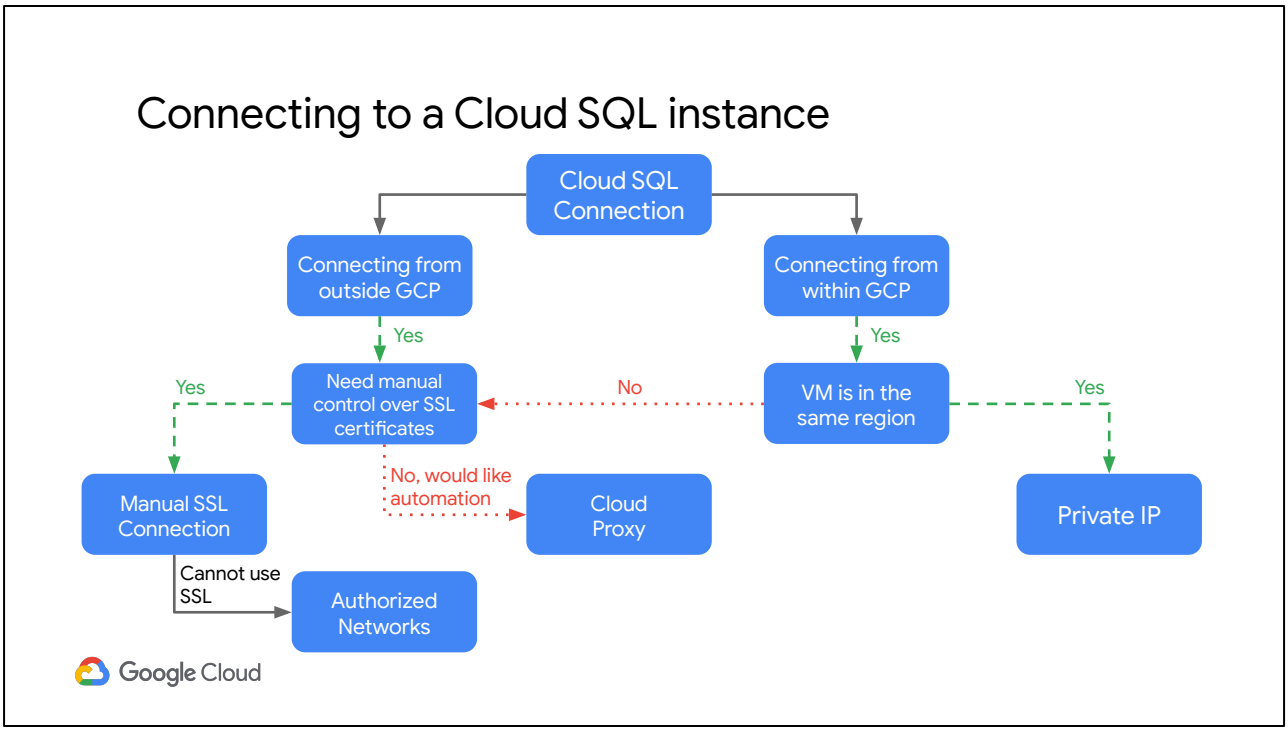


Google Cloud

Let's focus on some other services provided by Cloud SQL:

- There is replica service that can replicate data between multiple zones as shown on the right. This is useful for automatic failover if an outage occurs.
- Cloud SQL also provides automated and on-demand backups with point-in-time recovery.
- You can import and export databases using mysqldump, or import and export CSV files.
- Cloud SQL can also scale up, which does require a machine restart or scale out using read replicas.

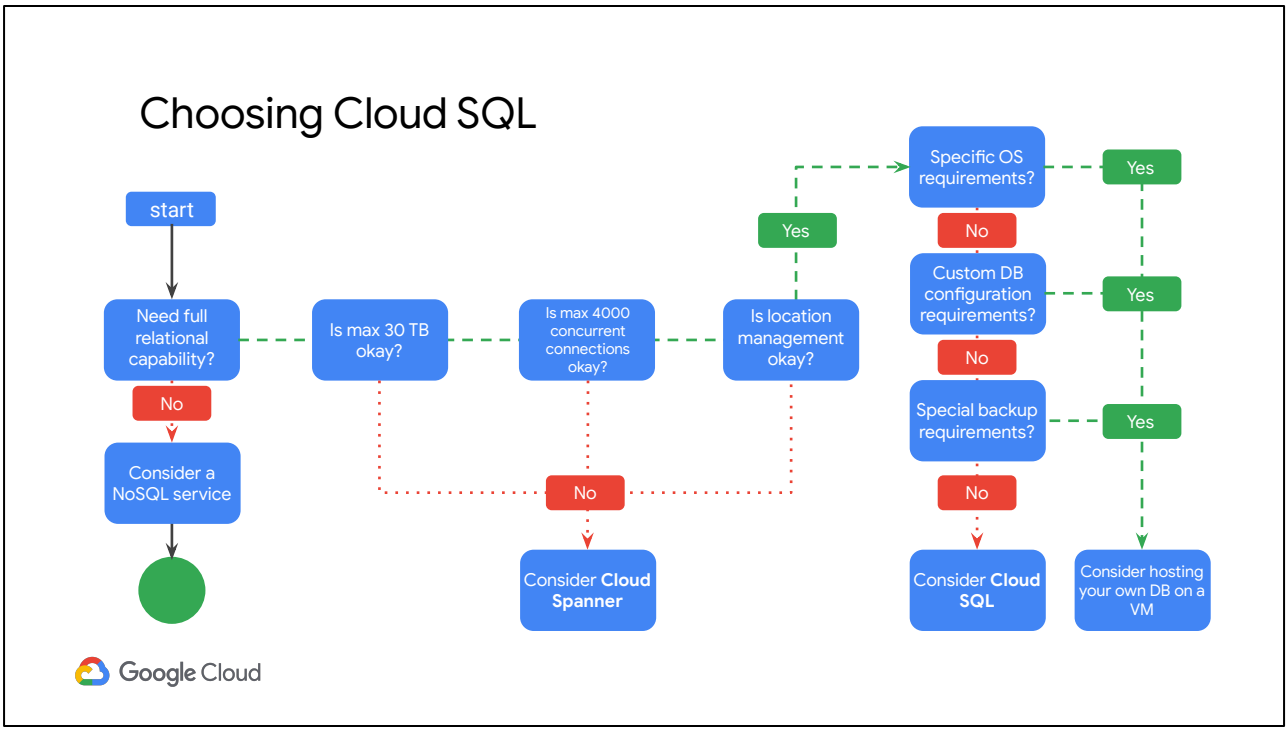
That being said, if you are concerned about horizontal scalability, you'll want to consider Cloud Spanner which we'll cover later in this module.



Choosing a connection type to your Cloud SQL instance will influence how secure, performant, and automated it will be. If your are connecting an application that is hosted within the same GCP project as your Cloud SQL instance, and it is collocated in the same region, choosing the Private IP connection will provide you with the most performant and secure connection using private connectivity. In other words, traffic is never exposed to the public internet.

If the application is hosted in another region or project, or if you are trying to connect to your Cloud SQL instance from outside of GCP, you have 3 options. In this case, I recommend using Cloud Proxy, which handles authentication, encryption, and key rotation for you. If you need manual control over the SSL connection, you can generate and periodically rotate the certificates yourself. Otherwise, you can use an unencrypted connection by authorizing a specific IP address to connect to your SQL server over its external IP address.

You will explore these options in the upcoming lab, but if you want to learn more about Private IP, see the links section of this video [https://cloud.google.com/sql/docs/mysql/private-ip].



To summarize, let's explore this decision tree to help you find the right data storage service with full relational capability.

If you need more than 30 TB of storage space or over ,4000 concurrent connections to your database, or if you want your application design to be responsible for scaling, availability, and location management when scaling up globally, then consider using Cloud Spanner, which we will cover later in this module.

If you have no concerns with these constraints, ask yourself whether you have specific OS requirements, custom database configuration requirements, or special backup requirements. If you do, consider hosting your own database on a VM using Compute Engine. Otherwise, I strongly recommend using Cloud SQL as a fully managed service for your relational databases.

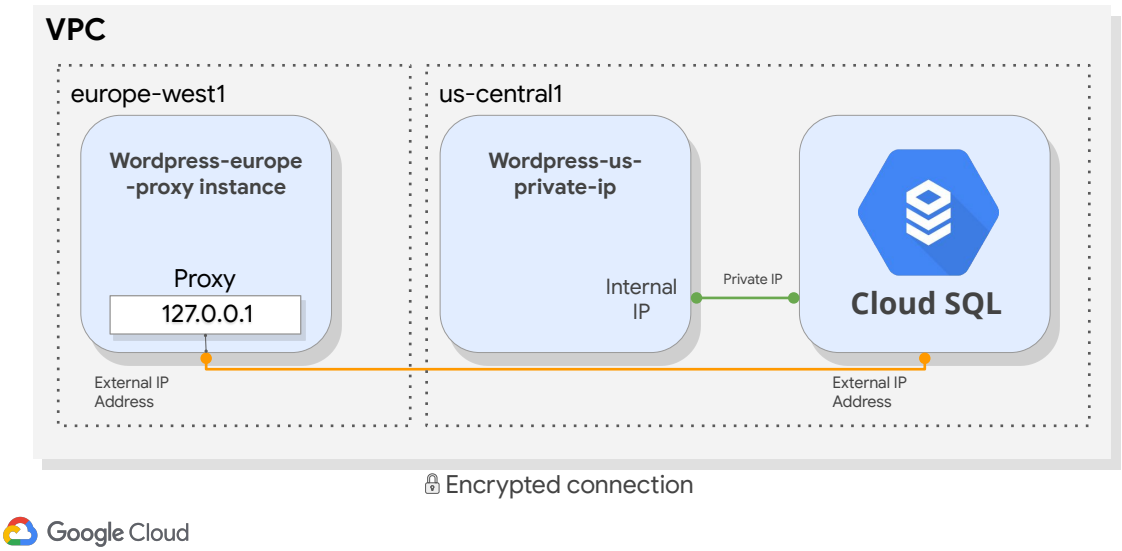
If you're now convinced that using Cloud SQL as a managed service is better than using or re-implementing your existing MySQL solution, see the links section for a solution on how to migrate from MySQL to Cloud SQL [https://cloud.google.com/solutions/migrating-mysql-to-cloudsql-concept]

Lab

Implementing Cloud SQL

Google Cloud

Let’s take some of the Cloud SQL concepts that we just discussed and apply them in a lab.



In this lab, you configure a Cloud SQL server and learn how to connect an application to it via a proxy over an external connection. You also configure a connection over a Private IP link that offers performance and security benefits. The app we chose to demonstrate in this lab is Wordpress, but the information and best practices are applicable to any application that needs a SQL Server.

By the end of this lab, you will have 2 working instances of a Wordpress frontend connected over 2 different connection types to its SQL instance backend, as shown in this diagram.

Lab review

Implementing Cloud SQL

Google Cloud

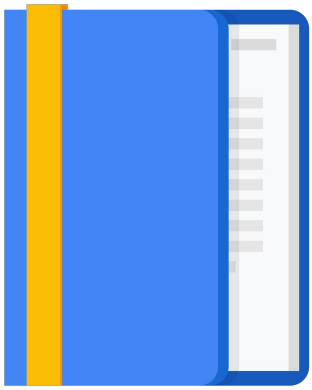
In this lab, you created a Cloud SQL database and configured it to use both an external connection over a secure proxy and a Private IP address, which is more secure and performant.

Remember, you can only connect via Private IP if the application and the Cloud SQL server are collocated in the same region and are part of the same VPC network. If your application is hosted in another region, VPC or even project, use a proxy to secure its connection over the external connection.

You can stay for a lab walkthrough, but remember that GCP's user interface can change, so your environment might look slightly different.

Agenda

- Cloud Storage
- Lab
- Cloud SQL
- Lab
- **Cloud Spanner**
- Cloud Firestore
- Cloud Bigtable
- Cloud Memorystore



Google Cloud

If Cloud SQL does not fit your requirements because you need horizontal scalability, consider using Cloud Spanner.

Cloud Spanner combines the benefits of relational database structure with non-relational horizontal scale

Cloud Spanner

- Scale to petabytes
- Strong consistency
- High availability
- Used for financial and inventory applications
- Monthly uptime
  - Multi-regional: 99.999%
  - Regional: 99.99%

Google Cloud

Cloud Spanner is a service built for the cloud specifically to combine the benefits of relational database structure with non-relational horizontal scale.
This service can provide petabytes of capacity and offers transactional consistency at global scale, schemas, SQL, and automatic, synchronous replication for high availability. Use cases include financial applications and inventory applications traditionally served by relational database technology.

Depending on whether you create a multi-regional or regional instance, you'll have different monthly uptime SLAs as shown on this slide. However, for up-to-date numbers, you should always refer to the documentation, which you'll find in the links section of this video: [https://cloud.google.com/spanner/sla]

Characteristics

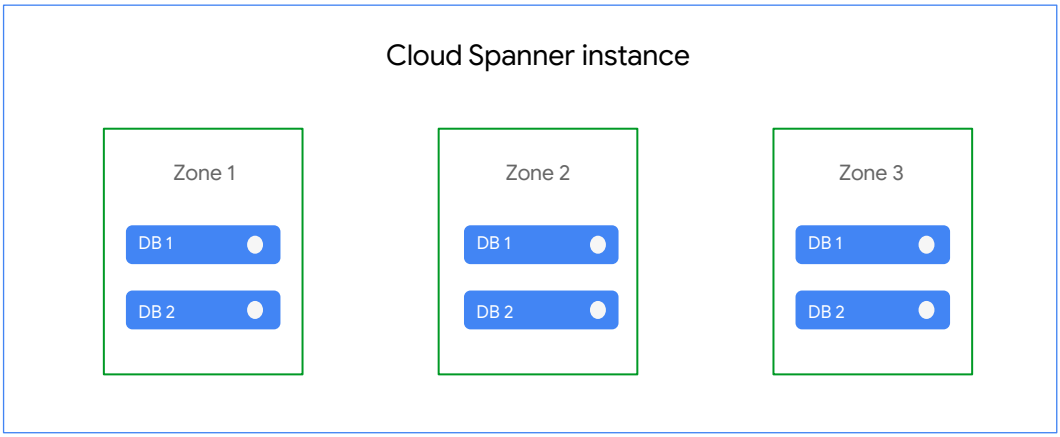
	Cloud Spanner	Relational DB	Non-Relational DB
Schema	✓ Yes	✓ Yes	✗ No
SQL	✓ Yes	✓ Yes	✗ No
Consistency	✓ Strong	✓ Strong	✗ Eventual
Availability	✓ High	✗ Failover	✓ High
Scalability	✓ Horizontal	✗ Vertical	✓ Horizontal
Replication	✓ Automatic	↻ Configurable	↻ Configurable

Google Cloud

Let's compare Cloud Spanner with both relational and non-relational databases. Like a relational database, Cloud Spanner has schema, SQL, and strong consistency. Also, like a non-relational database, Cloud Spanner offers high availability, horizontal scalability, and configurable replication.

As mentioned, Cloud Spanner offers the best of the relational and non-relational worlds. These features allow for mission-critical uses cases, such as building consistent systems for transactions and inventory management in the financial services and retail industries. To better understand how all of this works, let's look at the architecture of Cloud Spanner.

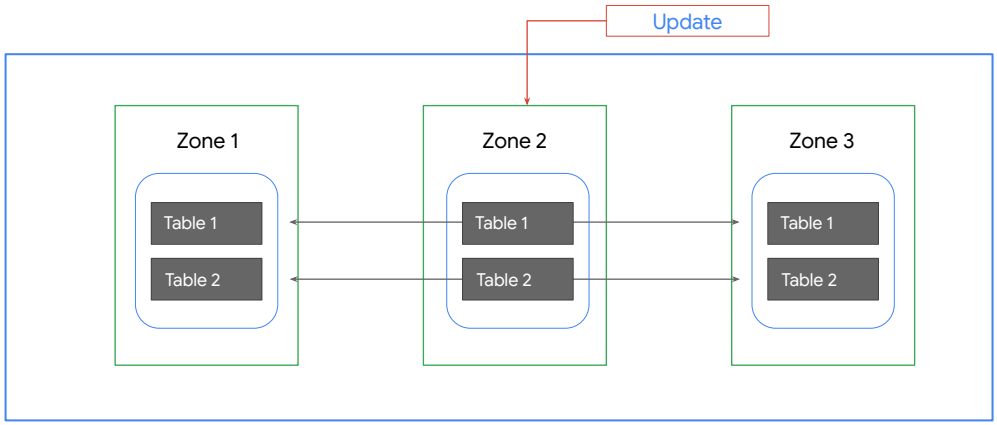
Cloud Spanner Architecture



Google Cloud

A Cloud Spanner instance replicates data in N cloud zones, which can be within one region or across several regions. The database placement is configurable, meaning you can choose which region to put your database in. This architecture allows for high availability and global placement.

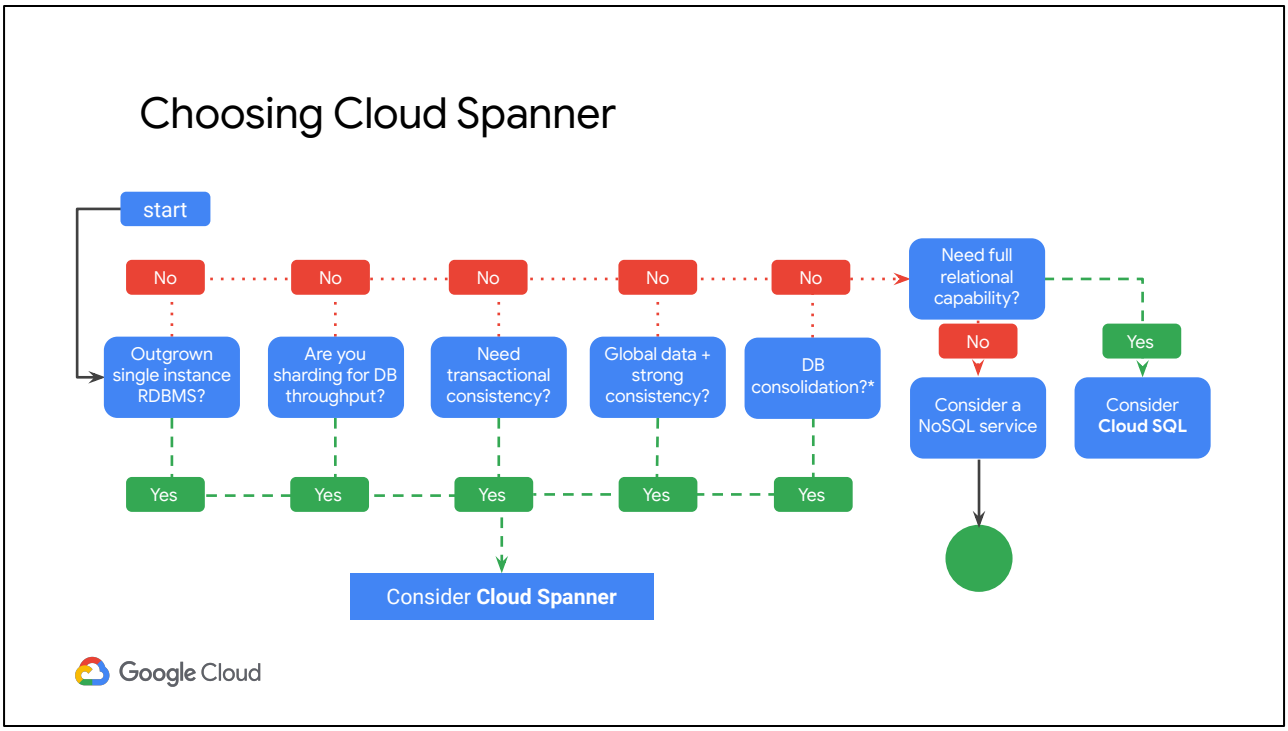
Data replication is synchronized across zones using Google's global fiber network



Google Cloud

The replication of data will be synchronized across zones using Google's global fiber network. Using atomic clocks ensures atomicity whenever you are updating your data.

That's as far as we're going to go with Cloud Spanner. Because the focus of this module is to understand the circumstances when you would use Cloud Spanner, let's look at a decision tree.



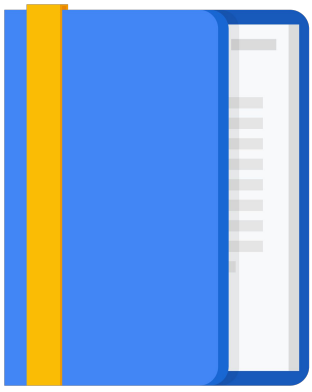
If you have outgrown any relational database, are sharding your databases for throughput high performance, need transactional consistency,
global data and strong consistency, or just want to consolidate your database, consider using Cloud Spanner.

If you don't need any of these, nor full relational capabilities, consider a NoSQL service such as Cloud Firestore, which we will cover next.

If you're now convinced that using Cloud Spanner as a managed service is better than using or re-implementing your existing MySQL solution, see the links section for a solution on how to migrate from MySQL to Cloud Spanner [https://cloud.google.com/solutions/migrating-mysql-to-spanner]

Agenda

- Cloud Storage
- Lab
- Cloud SQL
- Lab
- Cloud Spanner
- **Cloud Firestore**
- Cloud Bigtable
- Cloud Memorystore



Google Cloud

If you are looking for a highly-scalable NoSQL database for your applications, consider using Cloud Firestore.

Cloud Firestore is a NoSQL document database

- Simplifies storing, syncing, and querying data
- Mobile, web, and IoT apps at global scale
- Live synchronization and offline support
- Security features
- ACID transactions
- Multi-region replication
- Powerful query engine

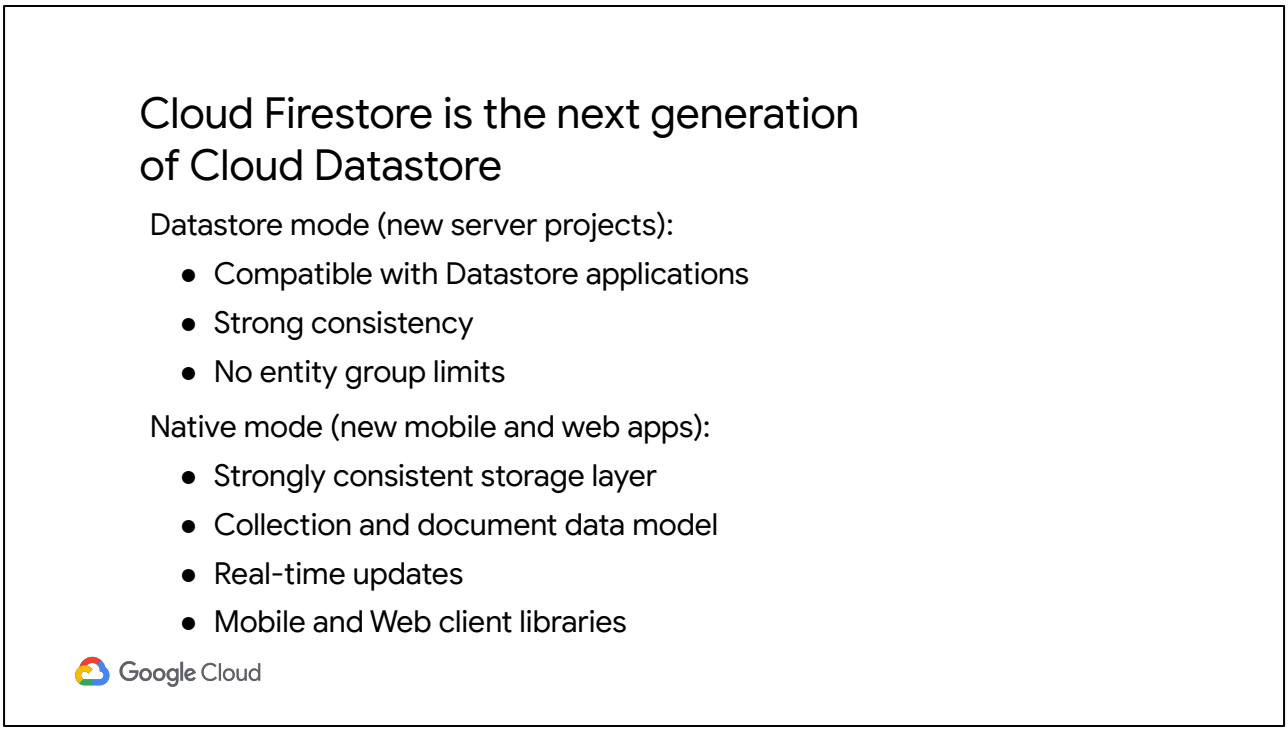
Cloud Firestore

Google Cloud

Cloud Firestore is a fast, fully managed, serverless, cloud-native NoSQL document database that simplifies storing, syncing, and querying data for your mobile, web, and IoT apps at global scale. Its client libraries provide live synchronization and offline support, and its security features and integrations with Firebase and GCP accelerate building truly serverless apps.

Cloud Firestore also supports ACID transactions, so if any of the operations in the transaction fail and cannot be retried, the whole transaction will fail.

Also, with automatic multi-region replication and strong consistency, your data is safe and available, even when disasters strike. Cloud Firestore even allows you to run sophisticated queries against your NoSQL data without any degradation in performance. This gives you more flexibility in the way you structure your data.



Cloud Firestore is actually the next generation of Cloud Datastore. Cloud Firestore can operate in Datastore mode, making it backwards- compatible with Cloud Datastore. By creating a Cloud Firestore database in Datastore mode, you can access Cloud Firestore's improved storage layer while keeping Cloud Datastore system behavior.

This removes the following Cloud Datastore limitations:

- Queries are no longer eventually consistent; instead, they are all strongly consistent.
- Transactions are no longer limited to 25 entity groups.
- Writes to an entity group are no longer limited to 1 per second.

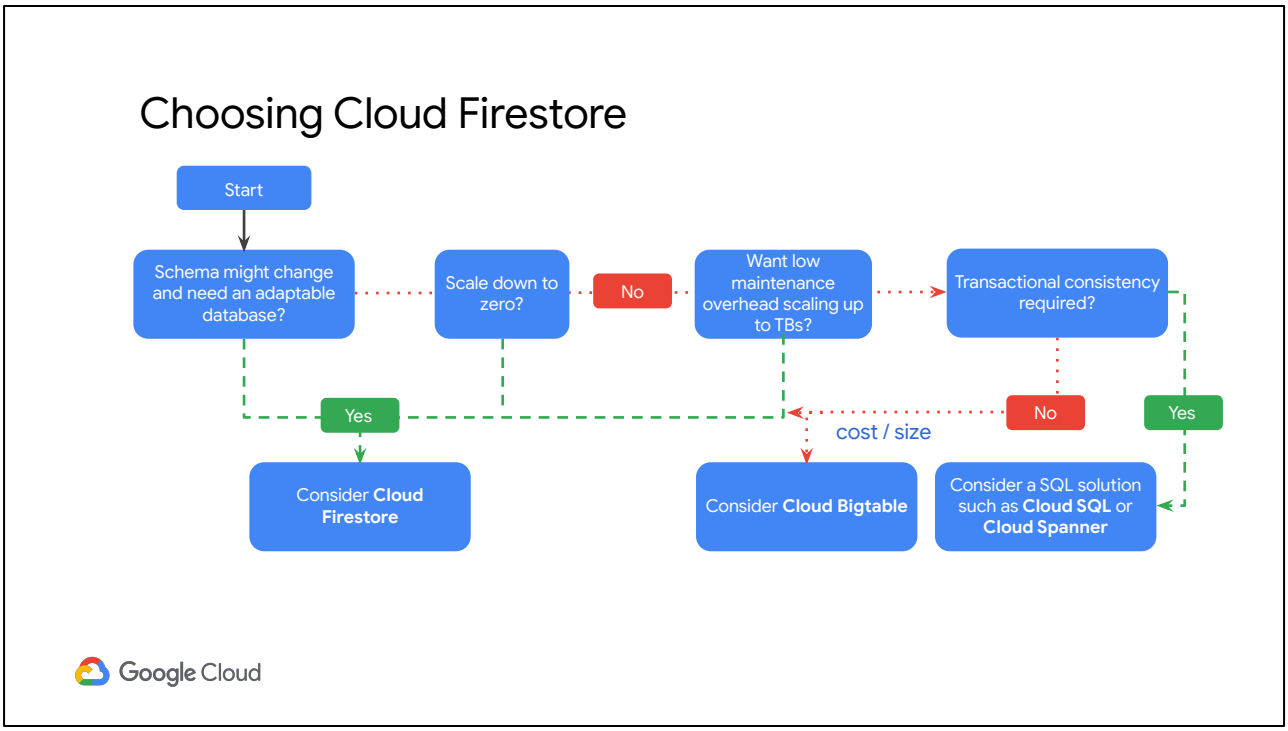
Cloud Firestore in Native mode introduces new features such as:

- A new, strongly consistent storage layer
- A collection and document data model
- Real-time updates
- Mobile and Web client libraries

Cloud Firestore is backward compatible with Cloud Datastore, but the new data model, real-time updates, and mobile and web client library features are not. To

access all of the new Cloud Firestore features, you must use Cloud Firestore in Native mode. A general guideline is to use Cloud Firestore in Datastore mode for new server projects, and Native mode for new mobile and web apps.

As the next generation of Cloud Datastore, Cloud Firestore is compatible with all Cloud Datastore APIs and client libraries. Existing Cloud Datastore users will be live-upgraded to Cloud Firestore automatically at a future date. For more information, see the links section of this video:
[https://cloud.google.com/datastore/docs/firestore-or-datastore, https://cloud.google.com/datastore/docs/upgrade-to-firestore]



To summarize, let's explore this decision tree to help you determine whether Cloud Firestore is the right storage service for your data.

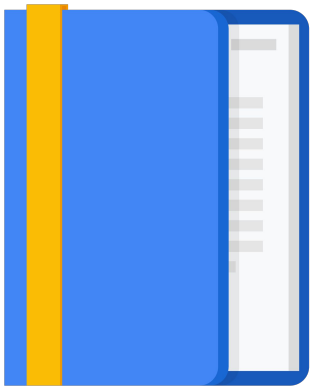
If your schema might change and you need an adaptable database, you need to scale to zero, or you want low maintenance overhead scaling up to terabytes, consider using Cloud Firestore.

Also, if you don't require transactional consistency, you might want to consider Cloud Bigtable, depending on the cost or size.

I will cover Cloud Bigtable next.

Agenda

- Cloud Storage
- Lab
- Cloud SQL
- Lab
- Cloud Spanner
- Cloud Firestore
- **Cloud Bigtable**
- Cloud Memorystore



Google Cloud

If you don't require transactional consistency, you might want to consider Cloud Bigtable.

Cloud Bigtable is a NoSQL big data database service

- Petabyte-scale
- Consistent sub-10ms latency
- Seamless scalability for throughput
- Learns and adjusts to access patterns
- Ideal for Ad Tech, Fintech, and IoT
- Storage engine for ML applications
- Easy integration with open source big data tools

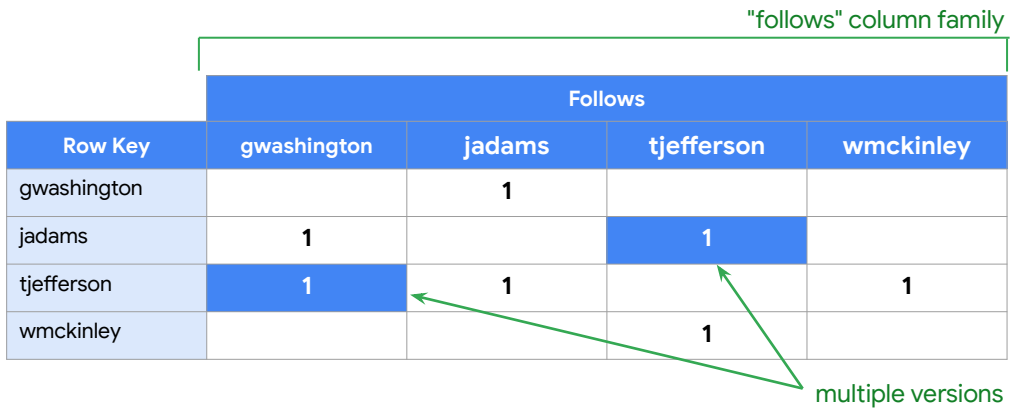
Google Cloud

Cloud Bigtable is a fully managed NoSQL database with petabyte-scale and very low latency. It seamlessly scales for throughput and it learns to adjust to specific access patterns. Cloud Bigtable is actually the same database that powers many of Google's core services, including Search, Analytics, Maps, and Gmail.

Cloud Bigtable is a great choice for both operational and analytical applications, including IoT, user analytics, and financial data analysis, because it supports high read and write throughput at low latency. It's also a great storage engine for machine learning applications.

Cloud Bigtable integrates easily with popular big data tools like Hadoop, Cloud Dataflow, and Cloud Dataproc. Plus, Cloud Bigtable supports the open source industry standard HBase API, which makes it easy for your development teams to get started. Cloud Dataflow and Cloud Dataproc are covered late in the course series. For more information on the HBase API, see the links section of this video: [https://hbase.apache.org/]

Cloud Bigtable storage model



Google Cloud

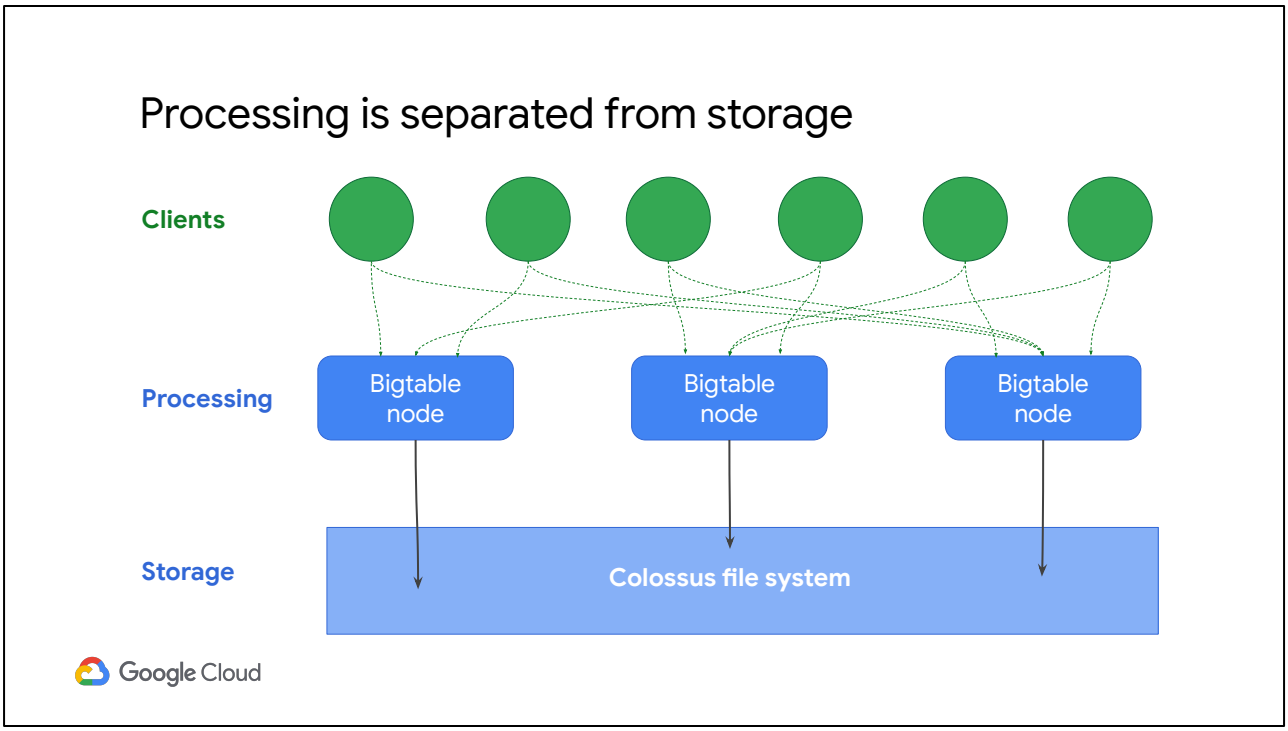
Cloud Bigtable stores data in massively scalable tables, each of which is a sorted key/value map. The table is composed of rows, each of which typically describes a single entity, and columns, which contain individual values for each row. Each row is indexed by a single row key, and columns that are related to one another are typically grouped together into a column family. Each column is identified by a combination of the column family and a column qualifier, which is a unique name within the column family.

Each row/column intersection can contain multiple cells, or versions, at different timestamps, providing a record of how the stored data has been altered over time. Cloud Bigtable tables are sparse; if a cell does not contain any data, it does not take up any space.

The example shown here is for a hypothetical social network for United States presidents, where each president can follow posts from other presidents. Let me highlight some things:

- The table contains one column family, the follows family. This family contains multiple column qualifiers.
- Column qualifiers are used as data. This design choice takes advantage of the sparseness of Cloud Bigtable tables, and the fact that new column qualifiers can be added as your data changes..

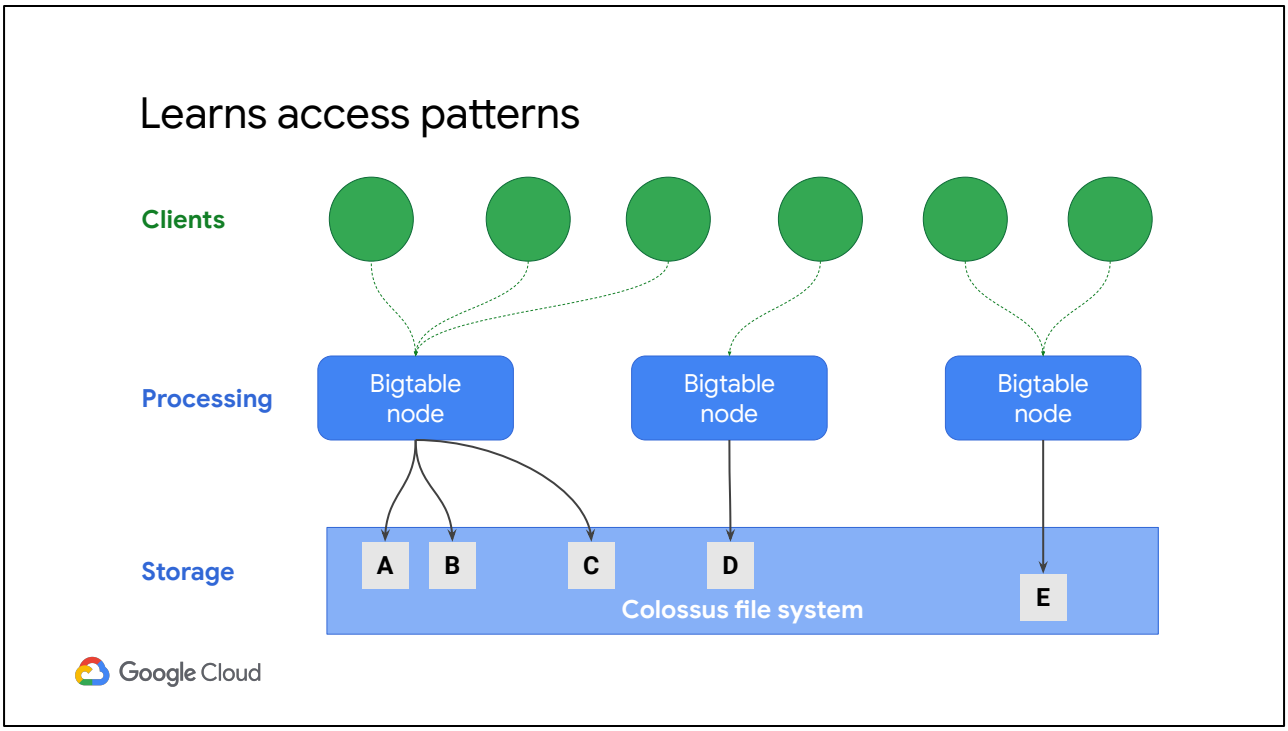
- The username is used as the row key. Assuming usernames are evenly spread across the alphabet, data access will be reasonably uniform across the entire table.



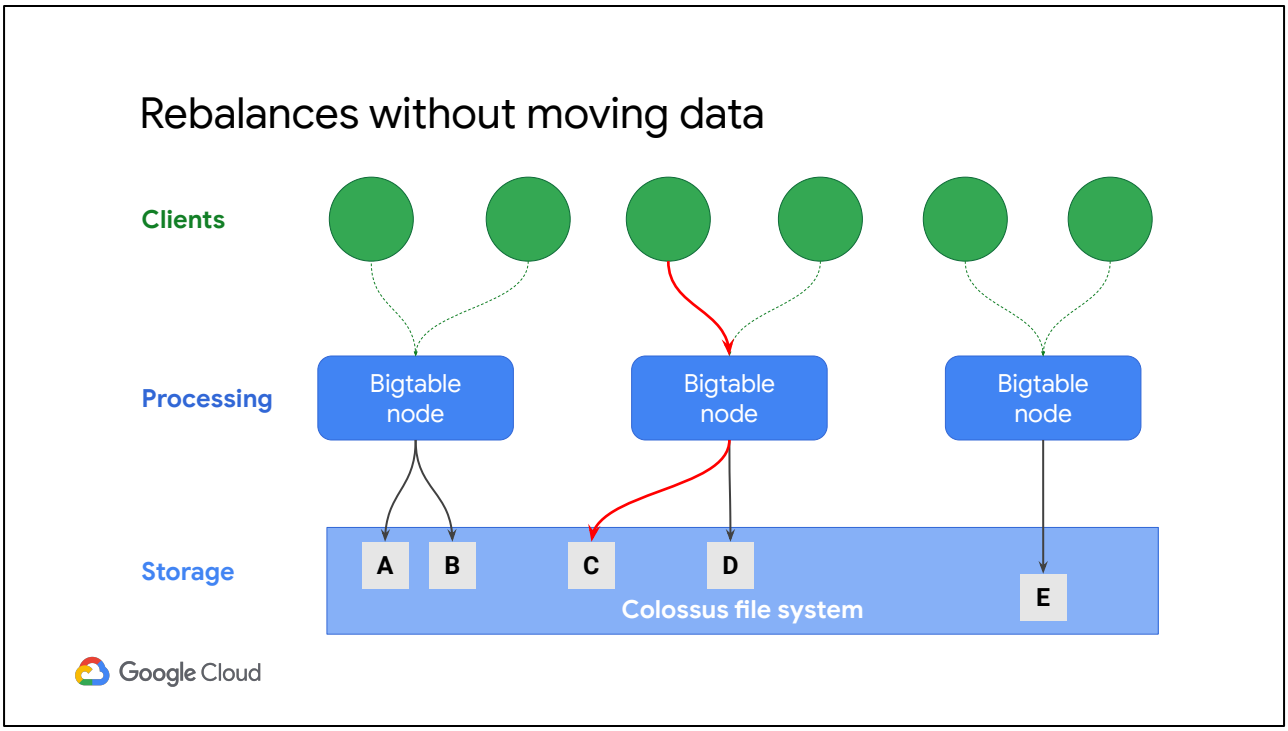
This diagram shows a simplified version of Cloud Bigtable's overall architecture. It illustrates that processing, which is done through a front-end server pool and nodes, is handled separately from the storage.

A Cloud Bigtable table is sharded into blocks of contiguous rows, called tablets, to help balance the workload of queries. Tablets are similar to HBase regions, for those of you who have used the HBase API.

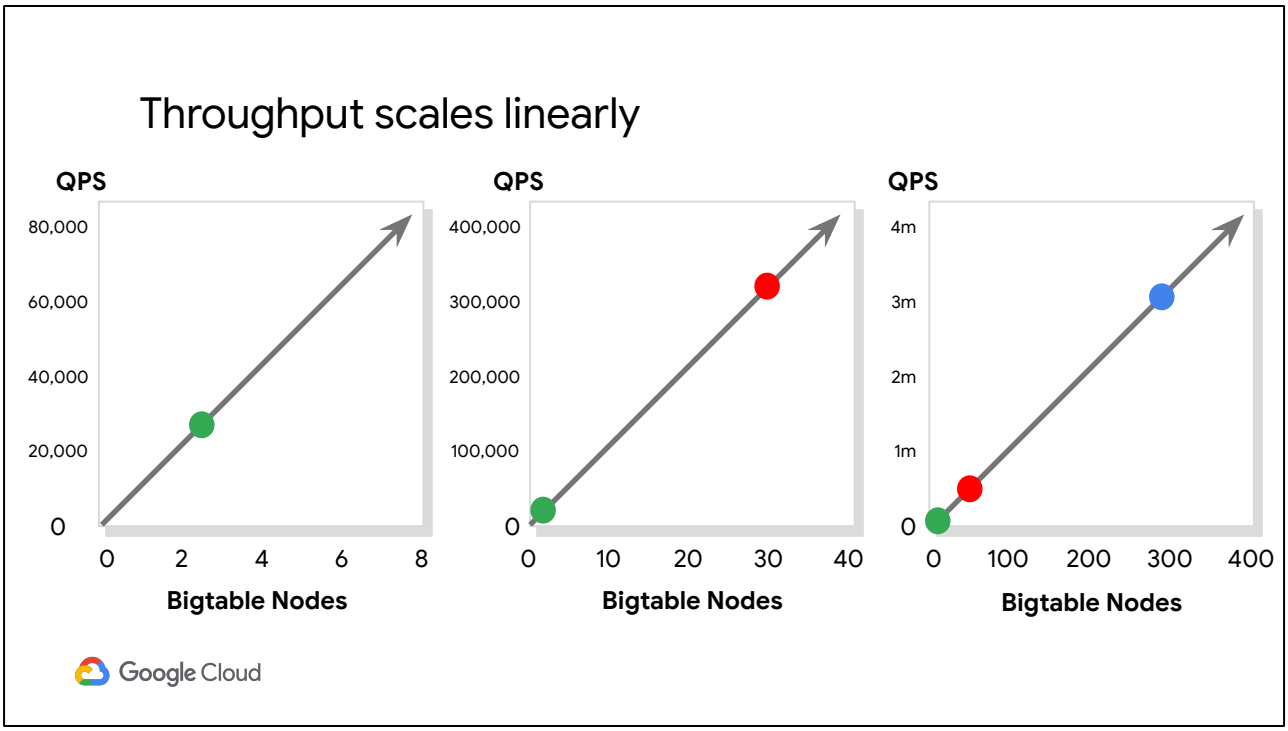
Tablets are stored on Colossus, which is Google's file system, in SSTable format. An SSTable provides a persistent, ordered immutable map from keys to values, where both keys and values are arbitrary byte strings.



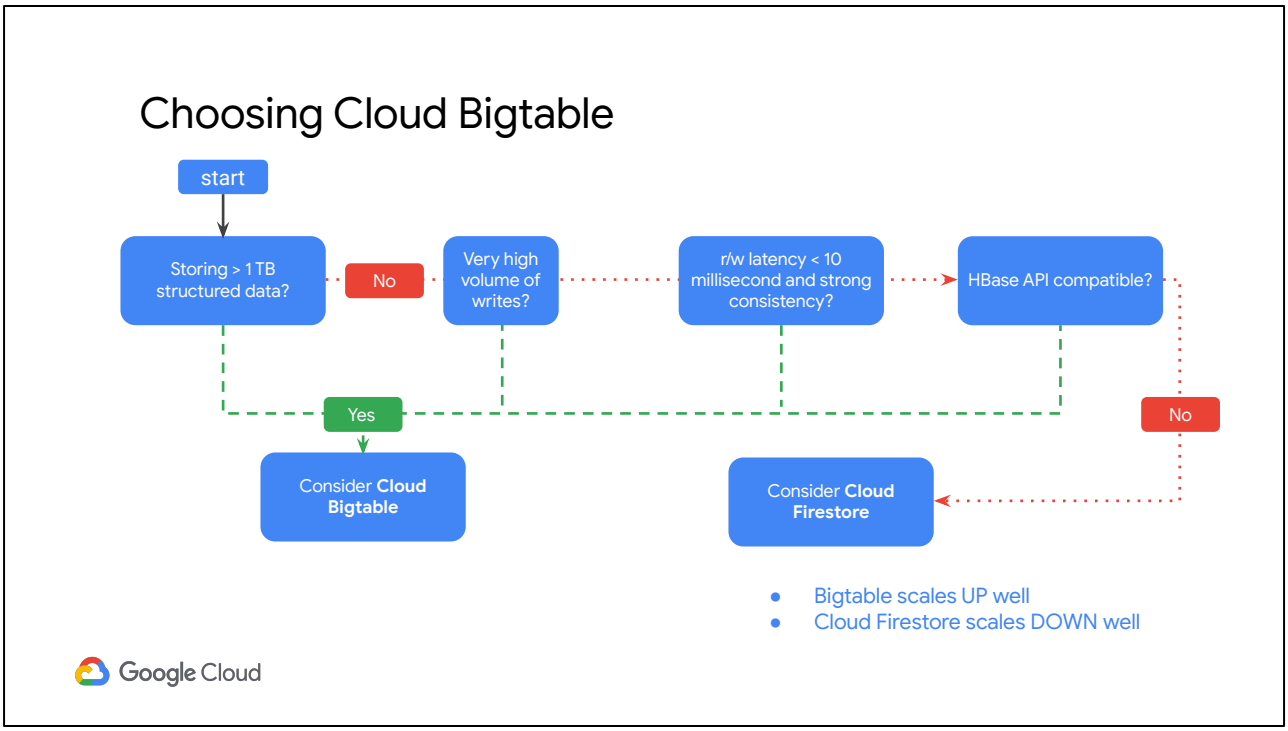
As I mentioned earlier, Cloud Bigtable learns to adjust to specific access patterns. If a certain Bigtable node is frequently accessing a certain subset of data...



… Cloud Bigtable will update the indexes so that other nodes can distribute that workload evenly, as shown here.



That throughput scales linearly, so for every single node that you do add, you're going to see a linear scale of throughput performance, up to hundreds of nodes.



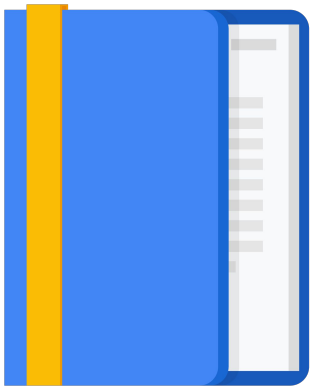
In summary, if you need to store more than 1 TB of structured data, have very high volume of writes, need read/write latency of less than 10 milliseconds along with strong consistency, or need a storage service that is compatible with the HBase API, consider using Cloud Bigtable.

If you don't need any of these and are looking for a storage service that scales down well, consider using Cloud Firestore.

Speaking of scaling, the smallest Cloud Bigtable cluster you can create has three nodes and can handle 30,000 operations per second. Remember that you pay for those nodes while they are operational, whether your application is using them or not.

Agenda

- Cloud Storage
- Lab
- Cloud SQL
- Lab
- Cloud Spanner
- Cloud Firestore
- Cloud Bigtable
- Cloud Memorystore



Google Cloud

Let me give you a quick overview of Cloud Memorystore.

Cloud Memorystore is a fully managed Redis service

- In-memory data store service
- Focus on building great apps
- High availability, failover, patching, and monitoring
- Sub-millisecond latency
- Instances up to 300 GB
- Network throughput of 12 Gbps
- Easy Lift-and-Shift

Google Cloud

Cloud Memorystore for Redis provides a fully managed in-memory data store service built on scalable, secure, and highly available infrastructure managed by Google. Applications running on GCP can achieve extreme performance by leveraging the highly scalable, available, secure Redis service without the burden of managing complex Redis deployments. This allows you to spend more time writing code so that you can focus on building great apps.

Cloud Memorystore also automates complex tasks like enabling high availability, failover, patching, and monitoring. High availability instances are replicated across two zones and provide a 99.9% availability SLA.

You can easily achieve the sub-millisecond latency and throughput your applications need. Start with the lowest tier and smallest size, and then grow your instance effortlessly with minimal impact to application availability. Cloud Memorystore can support instances up to 300 GB and network throughput of 12 Gbps.

Because Cloud Memorystore for Redis is fully compatible with the Redis protocol, you can lift and shift your applications from open source Redis to Cloud Memorystore without any code changes by using the import/export feature. There is no need to learn new tools because all existing tools and client libraries just work.

Review

Storage and Database Services

Google Cloud

In this module, we covered the different storage and database services that GCP offers. Specifically, you learned about Cloud Storage, a fully managed object store; Cloud SQL, a fully managed MySQL and PostgreSQL database service; Cloud Spanner, a relational database service with transactional consistency, global scale and high availability; Cloud Firestore, a fully managed NoSQL document database; Cloud Bigtable, a fully managed NoSQL wide-column database; and Cloud Memorystore, a fully managed in-memory data store service for Redis.

From an infrastructure perspective, the goal was to understand what services are available and how they're used in different circumstances. Defining a complete data strategy is beyond the scope of this course; however, Google offers courses on data engineering and machine learning on GCP that cover data strategy.