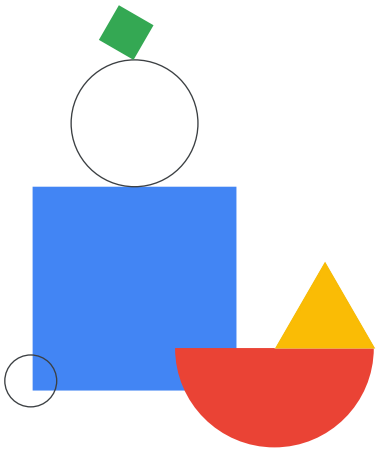
Google Cloud

Storage and Database Services

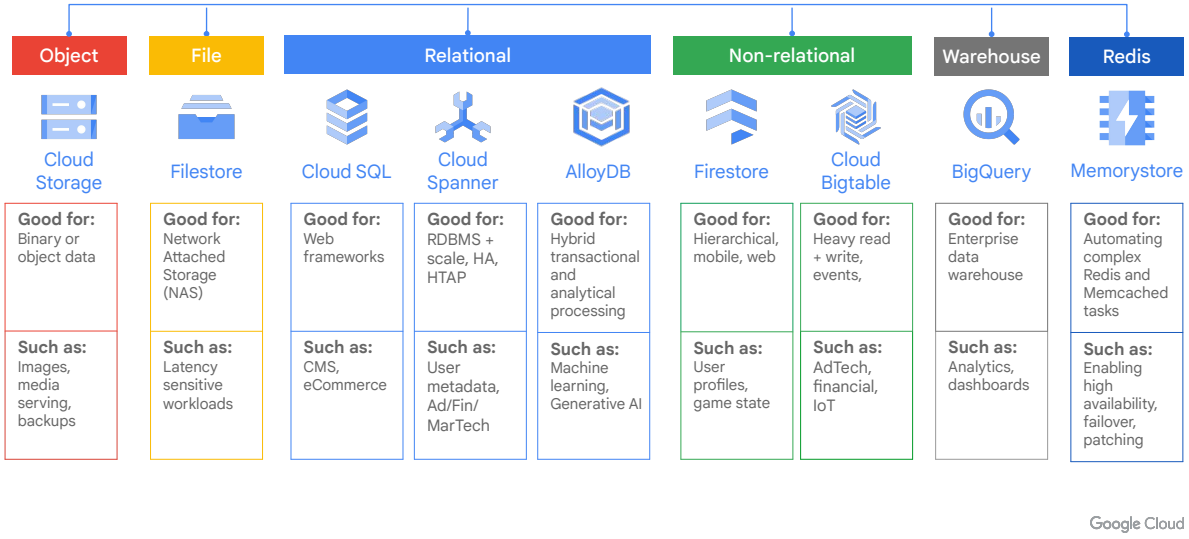


In this module, we cover storage and database services in Google Cloud. Every application needs to store data, whether it's business data, media to be streamed, or sensor data from devices.

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.

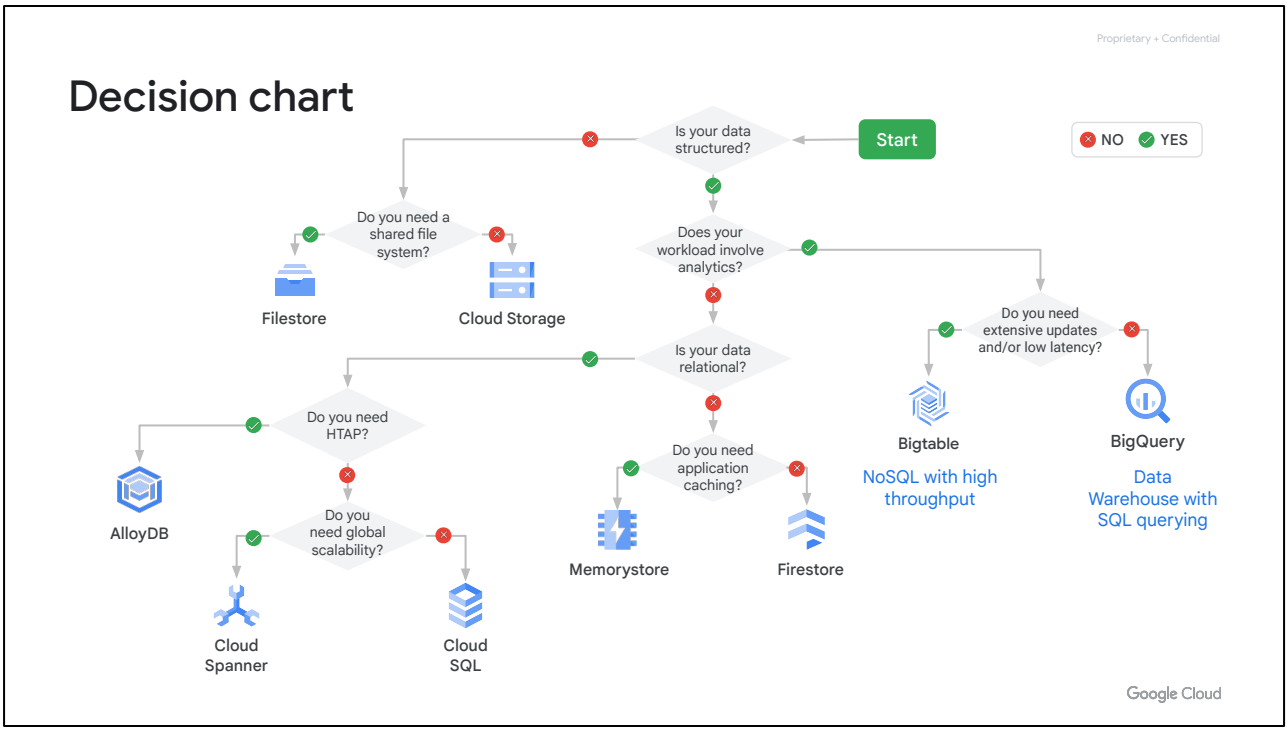
Google offers several data storage services to choose from. In this module, we will cover Cloud Storage, Filestore, Cloud SQL, Cloud Spanner, AlloyDB, Firestore, Cloud Bigtable, and Memorystore. Let me start by giving you a high-level overview of these different services.

Storage and database services



This table shows the storage and database services and highlights the storage service type, 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 course.



If tables aren't your preference, here's a 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 its not, then ask yourself if you need a shared file system. If you do, then choose Filestore.
- If you don't, then choose Cloud Storage.
- If your data is structured, does your workload focus on analytics? If it does, you will want to choose Bigtable or BigQuery, depending on your latency and update needs.
    - BigQuery is recommended as a data warehouse, is the default storage for tabular data, and is optimized for large-scale, ad-hoc SQL-based analysis and reporting. While BigQuery data manipulation language (DML) enables you to update, insert, and delete data from your BigQuery tables, because it has a built-in cache BigQuery works really well in cases where the data does not change often.
    - Bigtable is a NoSQL wide-column database. It's optimized for low latency, large numbers of reads and writes, and maintaining performance at scale.
    - In addition to analytics, Bigtable is also suited as a 'fast lookup' non-relational database for datasets too large to store in memory, with use cases in areas such as IoT, AdTech and FinTec.
- If your workload doesn't involve analytics, check whether your data is relational. If it's not relational, do you need application caching?
    - If caching is a requirement, choose Memorystore, an in-memory

    - database.
    - Otherwise choose Firestore, a document database.
- If your data is relational and you need Hybrid transaction/analytical processing, also known as HTAP, choose AlloyDB.
    - If you don't need HTAP and don't need global scalability, choose Cloud SQL.
    - If you don't need HTAP and need global scalability, choose Spanner.

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, please refer to the following two resources:

https://cloud.google.com/storage-options/
https://cloud.google.com/products/databases/

Scope

Infrastructure Track

- Service differentiators
- When to consider using each service
- Set up and connect to a service

Data Engineering Track

- How to use a database system
- Design, organization, structure, schema, and use for an application
- Details about how a service stores and retrieves structured data

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

01	Cloud Storage and Filestore Lab: Cloud Storage
02	Cloud SQL Lab: Implementing Cloud SQL
03	Cloud Spanner
04	AlloyDB
05	Firestore
06	Cloud Bigtable
07	Memorystore

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

We'll also provide a quick overview of Memorystore, which is Google Cloud's fully managed Redis service.

Let's get started by diving into Cloud Storage and Filestore!

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

Key features:

- Scalable to exabytes
- Time to first byte in milliseconds
- Very high availability across all storage classes
- Single API across storage classes

Cloud Storage is Google Cloud'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 key features

Use cases:

- Website content
- Storing data for archiving and disaster recovery
- Distributing large data objects to users via direct download

Key features:

- Scalable to exabytes
- Time to first byte in milliseconds
- Very high availability across all storage classes
- Single API across storage classes

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

	Standard	Nearline	Coldline	Archive
Use case	"Hot" data and/or stored for only brief periods of time like data-intensive computations	Infrequently accessed data like data backup, long-tail multimedia content, and data archiving	Infrequently accessed data that you read or modify at most once a quarter	Data archiving, online backup, and disaster recovery
Minimum storage duration	None	30 days	90 days	365 days
Retrieval cost	None	$0.01 per GB	$0.02 per GB	$0.05 per GB
Availability SLA	99.95% (multi/dual) 99.90% (region)	99.90% (multi/dual) 99.00% (region)		99.90% (multi/dual) 99.00% (region)
Durability	99.999999999%			

Cloud Storage has four storage classes: Standard, Nearline, Coldline and Archive and each of those storage classes provide 3 location types:

- There's a multi-region is a large geographic area, such as the United States, that contains two or more geographic places.
- Dual-region is a specific pair of regions, such as Finland and the Netherlands.
- A region is a specific geographic place, such as London.

Overview of storage classes

	Standard	Nearline	Coldline	Archive
Use case	"Hot" data and/or stored for only brief periods of time like data-intensive computations	Infrequently accessed data like data backup, long-tail multimedia content, and data archiving	Infrequently accessed data that you read or modify at most once a quarter	Data archiving, online backup, and disaster recovery
Minimum storage duration	None	30 days	90 days	365 days
Retrieval cost	None	$0.01 per GB	$0.02 per GB	$0.05 per GB
Availability SLA	99.95% (multi/dual) 99.90% (region)	99.90% (multi/dual) 99.00% (region)		99.90% (multi/dual) 99.00% (region)
Durability	99.999999999%			

Objects stored in a multi-region or dual-region are geo-redundant. Now, let's go over each of the storage classes:

Standard Storage is best for data that is frequently accessed (think of "hot" data) and/or stored for only brief periods of time. This is the most expensive storage class but it has no minimum storage duration and no retrieval cost.

When used in a region, Standard Storage is appropriate for storing data in the same location as Google Kubernetes Engine clusters or Compute Engine instances that use the data. Co-locating your resources maximizes the performance for data-intensive computations and can reduce network charges.

When used in a dual-region, you still get optimized performance when accessing Google Cloud products that are located in one of the associated regions, but you also get improved availability that comes from storing data in geographically separate locations.

When used in multi-region, Standard Storage is appropriate for storing data that is accessed around the world, such as serving website content, streaming videos, executing interactive workloads, or serving data supporting mobile and gaming applications.

Overview of storage classes

	Standard	Nearline	Coldline	Archive
Use case	"Hot" data and/or stored for only brief periods of time like data-intensive computations	Infrequently accessed data like data backup, long-tail multimedia content, and data archiving	Infrequently accessed data that you read or modify at most once a quarter	Data archiving, online backup, and disaster recovery
Minimum storage duration	None	30 days	90 days	365 days
Retrieval cost	None	$0.01 per GB	$0.02 per GB	$0.05 per GB
Availability SLA	99.95% (multi/dual) 99.90% (region)	99.90% (multi/dual) 99.00% (region)		99.90% (multi/dual) 99.00% (region)
Durability	99.999999999%			

Nearline Storage is a low-cost, highly durable storage service for storing infrequently accessed data like data backup, long-tail multimedia content, and data archiving. Nearline Storage is a better choice than Standard Storage in scenarios where slightly lower availability, a 30-day minimum storage duration, and costs for data access are acceptable trade-offs for lowered at-rest storage costs.

Overview of storage classes

	Standard	Nearline	Coldline	Archive
Use case	"Hot" data and/or stored for only brief periods of time like data-intensive computations	Infrequently accessed data like data backup, long-tail multimedia content, and data archiving	Infrequently accessed data that you read or modify at most once a quarter	Data archiving, online backup, and disaster recovery
Minimum storage duration	None	30 days	90 days	365 days
Retrieval cost	None	$0.01 per GB	$0.02 per GB	$0.05 per GB
Availability SLA	99.95% (multi/dual) 99.90% (region)	99.90% (multi/dual) 99.00% (region)		99.90% (multi/dual) 99.00% (region)
Durability	99.999999999%			

Coldline Storage is a very-low-cost, highly durable storage service for storing infrequently accessed data. Coldline Storage is a better choice than Standard Storage or Nearline Storage in scenarios where slightly lower availability, a 90-day minimum storage duration, and higher costs for data access are acceptable trade-offs for lowered at-rest storage costs.

Overview of storage classes

	Standard	Nearline	Coldline	Archive
Use case	"Hot" data and/or stored for only brief periods of time like data-intensive computations	Infrequently accessed data like data backup, long-tail multimedia content, and data archiving	Infrequently accessed data that you read or modify at most once a quarter	Data archiving, online backup, and disaster recovery
Minimum storage duration	None	30 days	90 days	365 days
Retrieval cost	None	$0.01 per GB	$0.02 per GB	$0.05 per GB
Availability SLA	99.95% (multi/dual) 99.90% (region)	99.90% (multi/dual) 99.00% (region)		99.90% (multi/dual) 99.00% (region)
Durability	99.999999999%			

Archive Storage is the lowest-cost, highly durable storage service for data archiving, online backup, and disaster recovery. Unlike the "coldest" storage services offered by other Cloud providers, your data is available within milliseconds, not hours or days. Archive Storage also has higher costs for data access and operations, as well as a 365-day minimum storage duration. Archive Storage is the best choice for data that you plan to access less than once a year.

Overview of storage classes

	Standard	Nearline	Coldline	Archive
Use case	"Hot" data and/or stored for only brief periods of time like data-intensive computations	Infrequently accessed data like data backup, long-tail multimedia content, and data archiving	Infrequently accessed data that you read or modify at most once a quarter	Data archiving, online backup, and disaster recovery
Minimum storage duration	None	30 days	90 days	365 days
Retrieval cost	None	$0.01 per GB	$0.02 per GB	$0.05 per GB
Availability SLA	99.95% (multi/dual) 99.90% (region)	99.90% (multi/dual) 99.00% (region)		99.90% (multi/dual) 99.00% (region)
Durability	99.999999999%			

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 we don't have access to it which is the availability that differs between storage classes and the location type.

Cloud Storage overview

Buckets

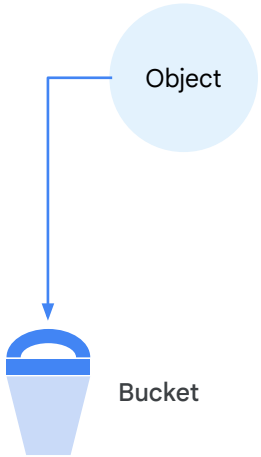
- Naming requirements
- Cannot be nested

Objects

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

Access

- `gcloud storage` command
- (RESTful) JSON API or XML API



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 gcloud storage command, or either the JSON or XML APIs.

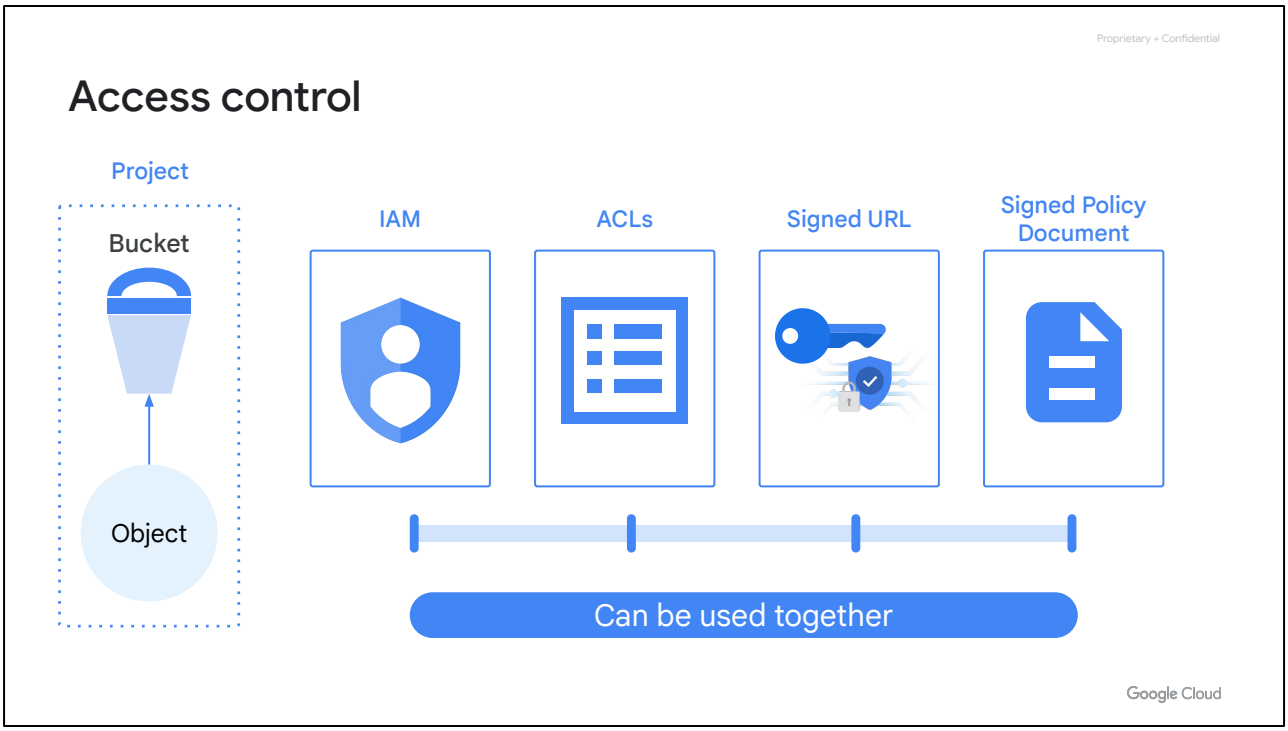
Changing default storage classes

- Default class is applied to new objects
- Regional bucket can never be changed to Multi-Region/Dual-Region
- 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

When you upload an object to a bucket, the object is assigned the bucket's storage class, unless you specify a storage class for the object. You can change the default storage class of a bucket but you can't change the location type from regional to multi-region/dual-region or vice versa.

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, Coldline or Archive Storage.

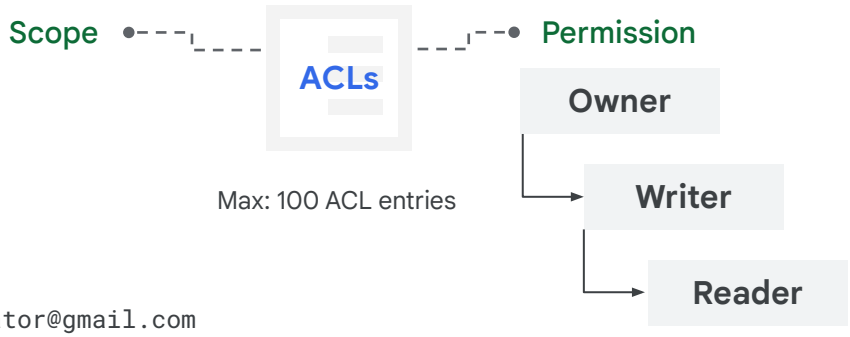
In order to help manage the classes of objects in your bucket, Cloud Storage offers Object Lifecycle Management. More on that later.



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, 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`

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 gcloud storage:

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


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.

Cloud Storage features

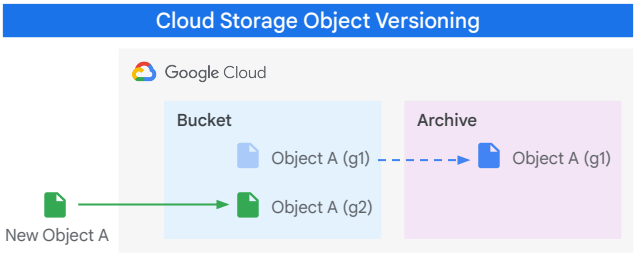
- Customer-supplied encryption key (CSEK)
  - Use your own key instead of Google-managed keys
- Object Lifecycle Management
  - Automatically delete or archive objects
- Object Versioning
  - Maintain multiple versions of objects
- Directory synchronization
  - Synchronizes a VM directory with a bucket
- Object change notifications using Pub/Sub
- Autoclass

There are also several features that come with Cloud Storage. We 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 versions 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.
- **Object change notifications** can be configured for Cloud Storage using Pub/Sub.
- When enabled, Autoclass manages all aspects of storage classes for a bucket. We will discuss this later.

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.



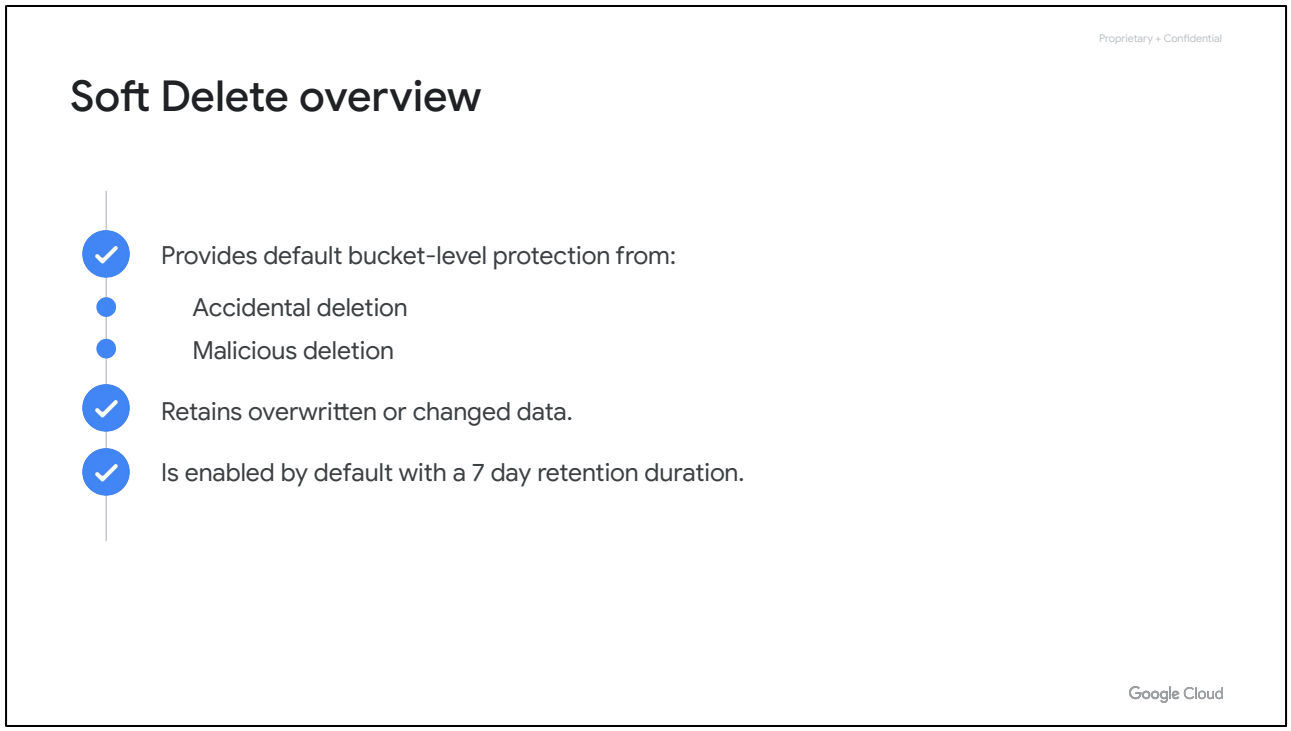
In Cloud Storage, **objects are immutable**, which means that an uploaded object cannot change throughout its storage lifetime. To support the retrieval of objects that are deleted or overwritten, Cloud Storage offers the Object Versioning feature.

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.

Google recommends that you use Soft Delete instead of Object Versioning to protect against permanent data loss from accidental or malicious deletions.

For more information on Object Versioning, refer to the documentation.

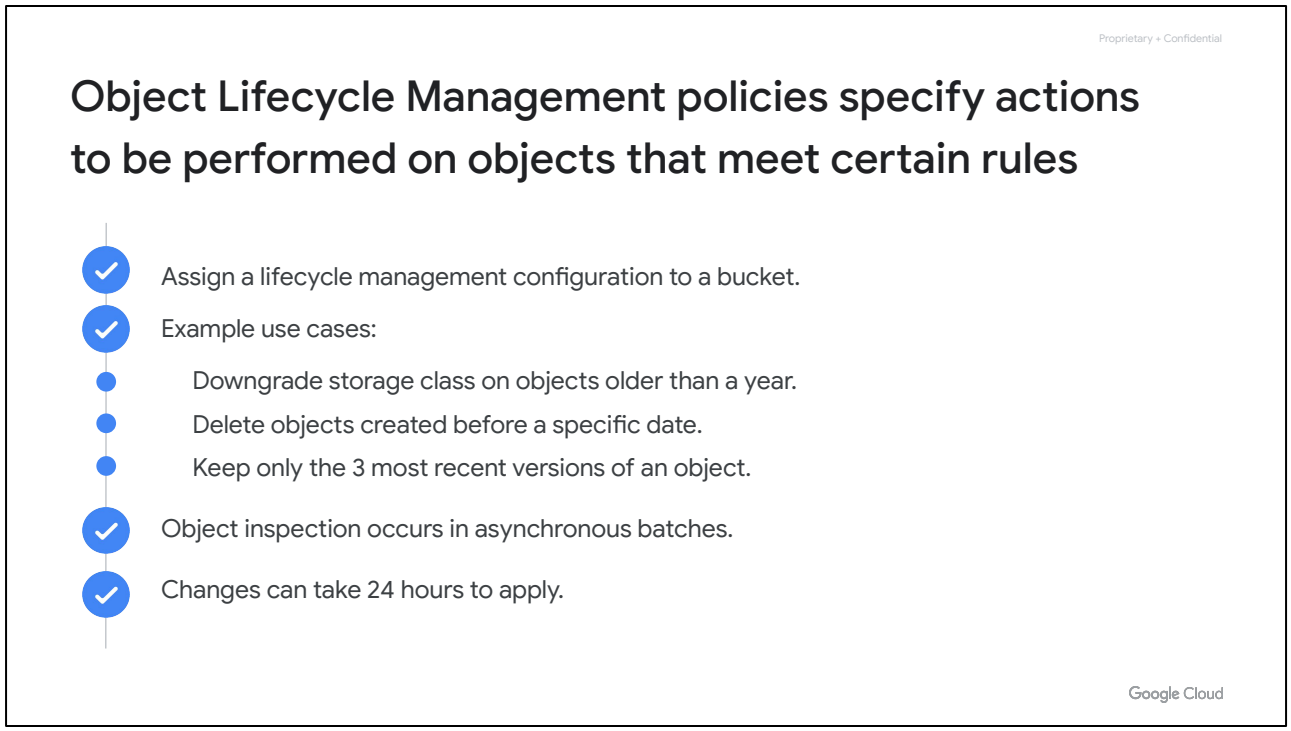


Soft Delete provides default bucket-level protection for your data from accidental or malicious deletion by preserving all recently deleted objects for a specified period of time.

The objects stored in Cloud Storage buckets are immutable. If you overwrite or change the data of an object, Cloud Storage deletes its earlier version and replaces it with a new one. Soft Delete retains all these deleted objects, whether from a delete command or because of an overwrite, essentially capturing all changes made to bucket data for the configured retention duration.

When you create a Cloud Storage bucket, the Soft Delete feature is enabled by default with a retention duration of seven days. During the retention duration, you can restore deleted objects, but after the duration ends, Cloud Storage permanently deletes the objects. By updating the bucket's configuration, you can increase the retention duration to 90 days or disable it by setting the retention duration to 0.

For more information on Soft Delete, refer to the documentation.



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.

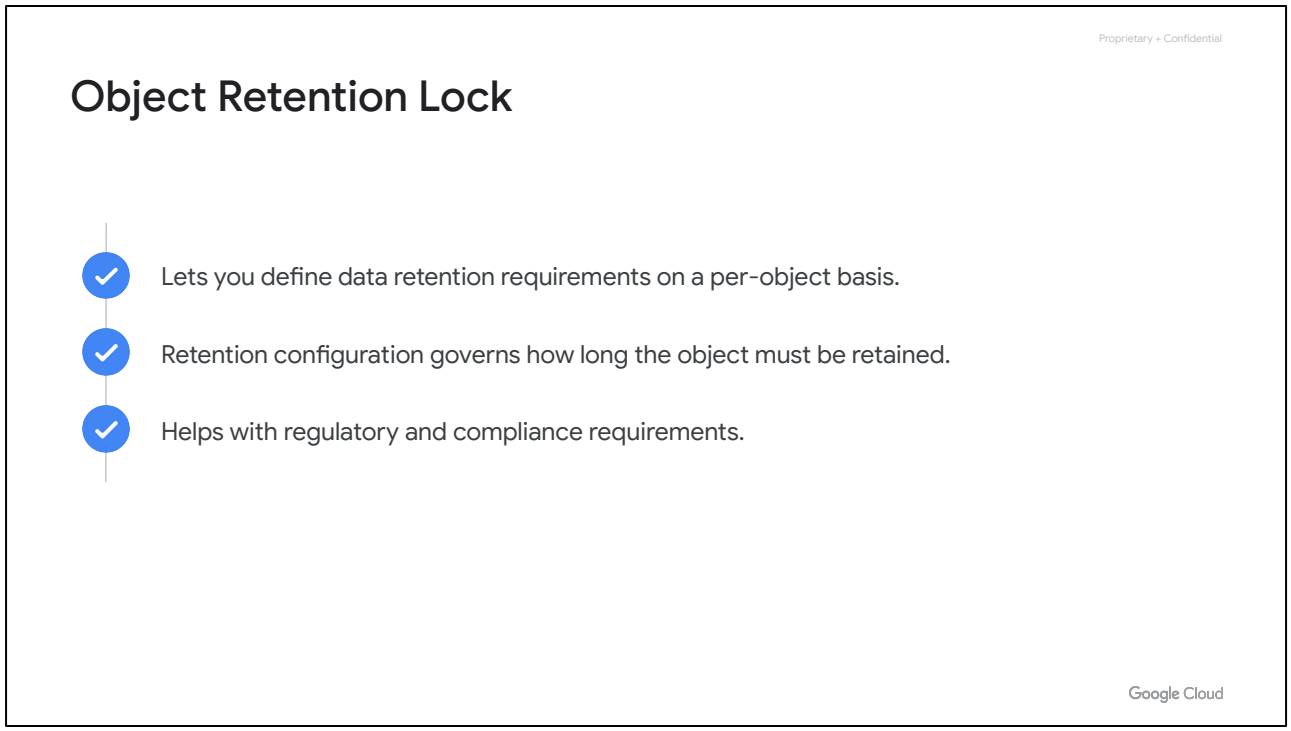
You can assign a lifecycle management configuration to a bucket. The configuration is a set of rules that apply to all the objects in the bucket. So when an object meets the criteria of one of the rules, Cloud Storage automatically performs a specified action on the object.

Here are some example use cases:

- First, downgrade the storage class of objects older than a year to Coldline Storage.
- Second, delete objects created before a specific date. For example, January 1, 2017.
- Or third, 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, refer to the Object Lifecycle Management documentation.



The Object Retention Lock feature lets you set retention configuration on objects within Cloud Storage buckets that have enabled the feature. A retention configuration governs how long the object must be retained and has the option to permanently prevent the retention time from being reduced or removed. This helps you meet data retention regulatory and compliance requirements, such as those associated with FINRA, SEC, and CFTC. This also helps provide Google Cloud immutable storage solutions with leading enterprise backup software vendor partners.

For more information, refer to the Object Retention Lock documentation.

Data import services

- **Transfer Appliance:** Rack, capture and then ship your data to Google Cloud.
- **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.

The Cloud Console allows you to upload individual files to your bucket. But 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 Google Cloud 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, tapes, and USB flash drives) is sent to a provider who uploads the data.

For more information on these three services, refer to the following resources:
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

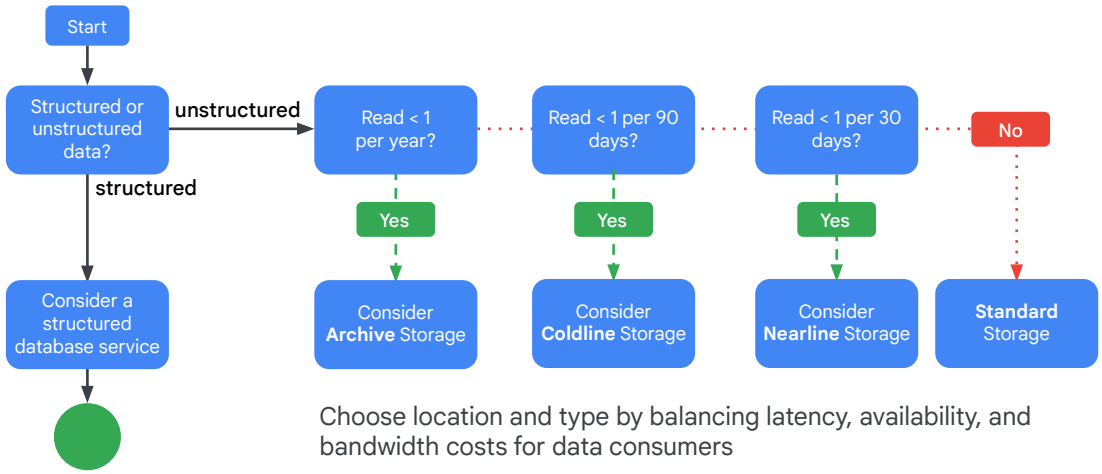
When you upload an object to Cloud Storage and you receive a success response, the object is immediately available for download and metadata operations from any location where Google offers service. This is true whether you create a new object or overwrite an existing object. Because uploads are strongly 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.

Choosing a storage class



Choose location and type by balancing latency, availability, and bandwidth costs for data consumers

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 **Archive storage.**
- If you will read your data less than once per 90 days, you should consider using **Coldline storage**.
- If you will read your data less than once per 30 days, you should consider using **Nearline storage**.
- And if you will be doing reads and writes more often than that, you should consider using **Standard storage**.

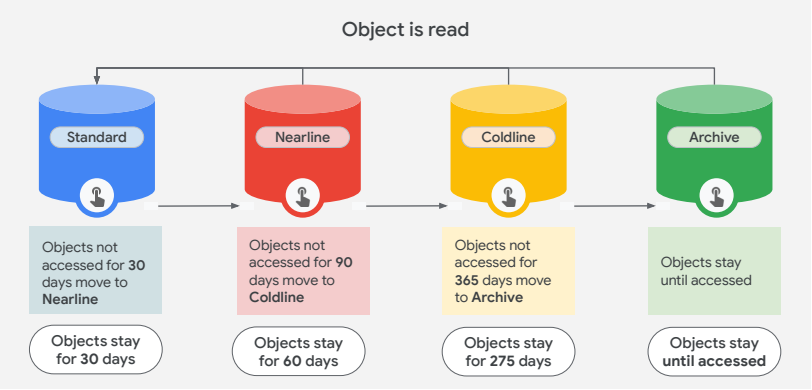
You also want to take into account the location type:

- Use a **region** to help optimize latency and network bandwidth for data consumers, such as analytics pipelines, that are grouped in the same region.
- Use a **dual-region** when you want similar performance advantages as regions, but also want the higher availability that comes with being geo-redundant.
- Use a **multi-region** when you want to serve content to data consumers that are outside of the Google network and distributed across large geographic areas, or when you want the higher data availability that comes with being geo-redundant.

If your data has a variety of access frequencies, or the access patterns for your data are unknown or unpredictable, you should consider Autoclass.

Autoclass storage in Google Cloud

Autoclass transitions objects in your bucket to appropriate storage classes based on the access pattern of each object.



The Autoclass feature automatically transitions objects in your bucket to appropriate storage classes based on the access pattern of each object. Even if a different storage class is specified in the request, all objects added to the bucket begin in Standard storage. The feature moves data that is not accessed to colder storage classes to reduce storage cost. Data that is accessed is also moved to Standard storage to optimize future accesses.

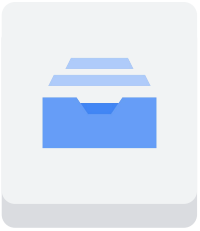
When object data is read, the object transitions to Standard storage if it's not already stored in Standard storage. Autoclass simplifies and automates cost saving for your Cloud Storage data. When enabled on a bucket, there are no early deletion charges, no retrieval charges, and no charges for storage class transitions.

For more information, view the storage classes documentation.

So far we have only considered unstructured data. Before we look at unstructured data, let's explore a high-performance, fully managed file storage offering; Filestore.

Filestore is a managed file storage service for applications

- Fully managed network attached storage (NAS) for Compute Engine and GKE instances.
- Predictable performance.
- Full NFSv3 support.
- Scales to 100s of TBs for high-performance workloads.



Filestore

Filestore is a managed file storage service for applications that require a filesystem interface and a shared filesystem for data.

Filestore gives users a simple, native experience for standing up managed Network Attached Storage (NAS) with their Compute Engine and Google Kubernetes Engine instances.

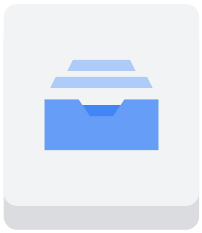
The ability to fine-tune Filestore's performance and capacity independently leads to predictably fast performance for your file-based workloads.

Filestore offers native compatibility with existing enterprise applications and supports any NFSv3-compatible client.

Applications gain the benefit of features such as scaleout performance, 100s of TBs of capacity, and file locking, without the need to install or maintain any specialized plugins or client side software.

Filestore has many use cases

- Application migration
- Media rendering
- Electronic Design Automation (EDA)
- Data analytics
- Genomics processing
- Web content management



Filestore

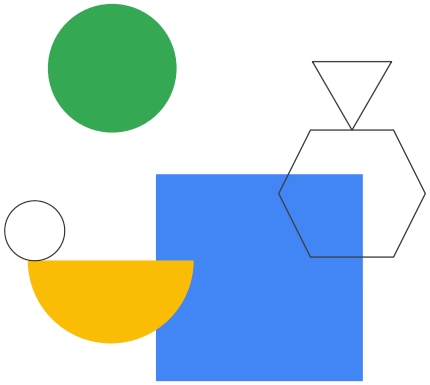
Filestore has many use cases.

- Using Filestore, you can expedite migration of enterprise applications. Many on-premises applications require a filesystem interface to data. As these applications continue to migrate to the cloud, Filestore can support a broad range of enterprise applications that need a shared filesystem.
- For media rendering, you can easily mount Filestore file shares on Compute Engine instances, enabling visual effects artists to collaborate on the same file share. As rendering workflows typically run across fleets (“render farms”) of compute machines, all of which mount a shared filesystem, Filestore and Compute Engine can scale to meet your job’s rendering needs.
- Electronic Design Automation (EDA) is all about data management. It requires the ability to batch workloads across thousands of cores and has large memory needs. Filestore offers the necessary capacity and scale to meet the needs of manufacturing customers doing intensive EDA and also makes sure files are universally accessible.
- Data analytics workloads include compute complex financial models or analysis of environmental data. These workloads are latency sensitive. Filestore offers low latency for file operations and, as capacity or performance needs change, you can easily grow or shrink your instances as needed. As a persistent and shareable storage layer, Filestore enables immediate access to data for high-performance, smart analytics without the need to lose valuable time on loading and off-loading data to clients’ drives.
- Genome sequencing requires an incredible amount of raw data, on the order

- of billions of data points per person. This type of analysis requires speed, scalability, and security. Filestore meets the needs of companies and research institutions performing scientific research, while also offering predictable prices for the performance.
- Web developers and large hosting providers also rely on Filestore to manage and serve web content, including needs such as WordPress hosting.

Lab Intro

Cloud Storage



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

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

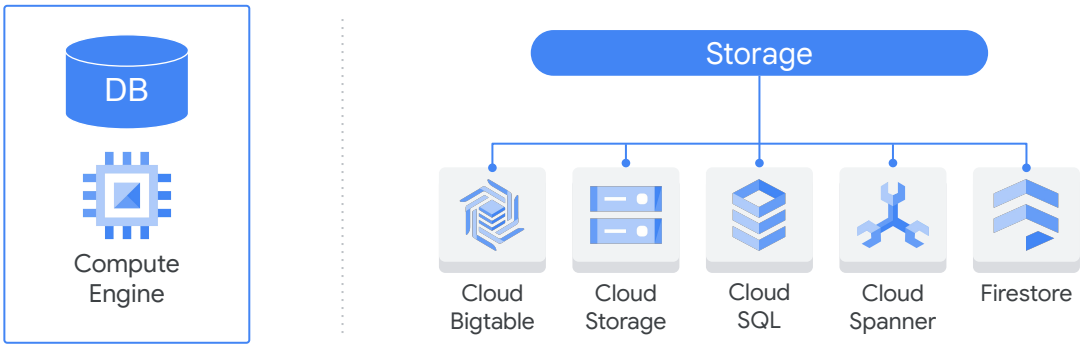
Now that you're familiar with many of the advanced features of Cloud Storage, you might consider using them in a variety of applications that you might not have previously considered. A common, quick, and easy way to start using GCP, is to use Cloud Storage as a backup service.

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

Cloud SQL

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

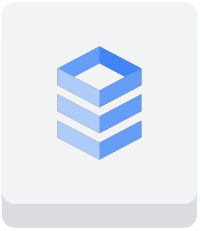


Why would you use a Google Cloud service for SQL, when you can install a SQL Server application image on a VM using Compute Engine?

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 Google Cloud.

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

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



Cloud SQL

Cloud SQL is a fully managed service of either MySQL, PostgreSQL, or Microsoft SQL Server 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 Google Workspace 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:

- 64 TB of storage
- 60,000 IOPS
- 624 GB of RAM
- Scale out with read replicas

Choice:

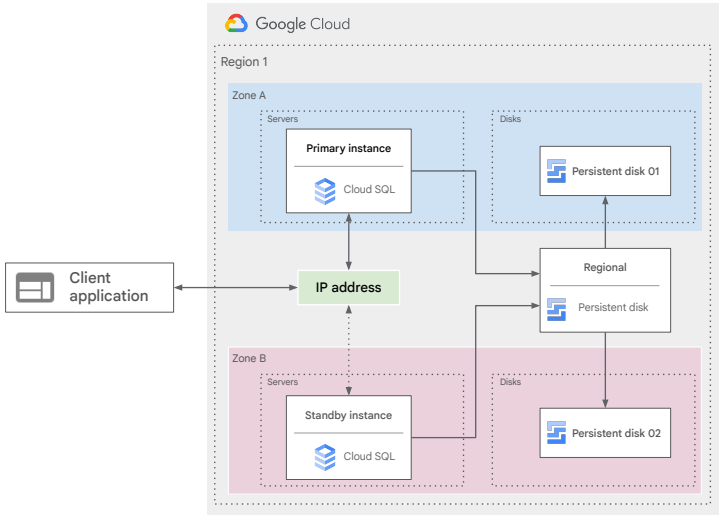
- MySQL 5.6, 5.7, or 8.0 (default)
- PostgreSQL 9.6, 10, 11, 12, 13, 14 or 15 (default)
- Microsoft SQL Server 2017 or 2019 (Standard default)

Cloud SQL delivers high performance and scalability with up to 64 TB of storage capacity, 60,000 IOPS, and 624 GB of RAM per instance. You can easily scale up to 96 processor cores and scale out with read replicas.

Currently, you can use Cloud SQL with either MySQL 5.6, 5.7, or 8.0, PostgreSQL 9.6, 10, 11, 12, 13, 14, or 15, or either of the Web, Express, Standard or Enterprise SQL Server 2017 or 2019 editions.

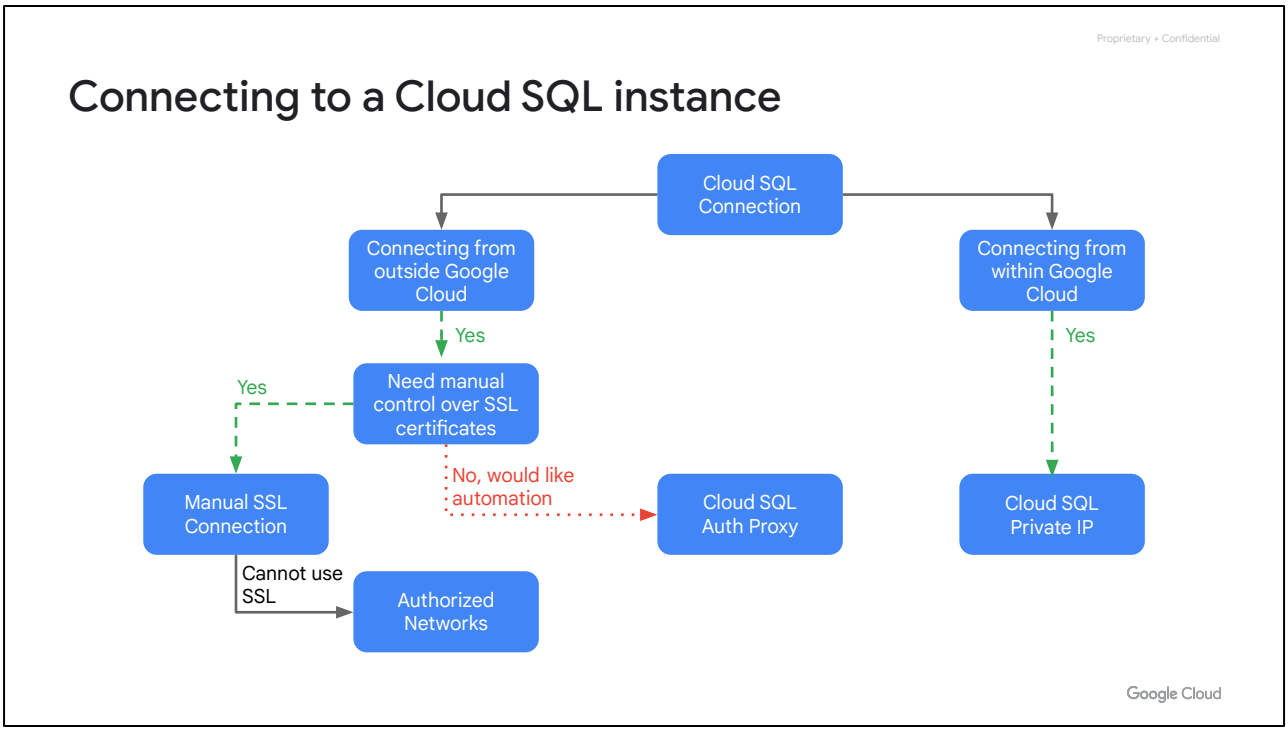
Cloud SQL services

- HA configuration
- Backup service
- Import/export
- Scaling
  - Up: Machine capacity
  - Out: Read replicas



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

- In HA configuration, within a regional instance, the configuration is made up of a *primary instance* and a *standby instance*. Through synchronous replication to each zone's persistent disk, all writes made to the primary instance are replicated to disks in both zones before a transaction is reported as committed. In the event of an instance or zone failure, the persistent disk is attached to the standby instance, and it becomes the new primary instance. Users are then rerouted to the new primary. This process is called a *failover*.
- 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.

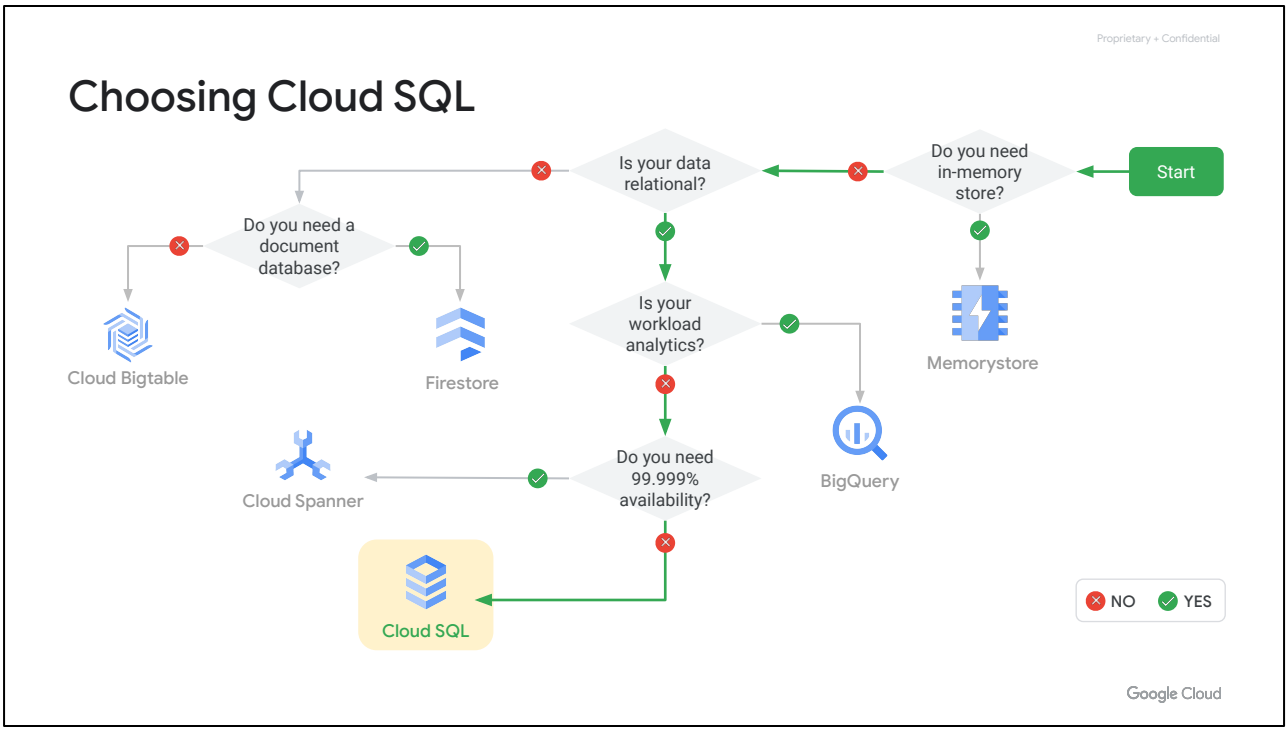


Choosing a connection type to your Cloud SQL instance will affect how secure, performant, and automated it will be. If you're connecting an application that is hosted within the same Google Cloud project as your Cloud SQL instance, and it is collocated in the same region, choosing the Private IP connection will provide you with the most performant and secure connection using private connectivity. In other words, traffic is never exposed to the public internet. Note that connecting to the Cloud SQL Private IP address from VMs in the same region is only a performance-based recommendation and not a requirement.

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 Google Cloud, you have 3 options. In this case, I recommend using the Cloud SQL Auth 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 an upcoming lab.

[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.

Memorystore provides a fully-managed in-memory data store service for workloads requiring microsecond response times, or that have large spikes in traffic, as seen in gaming environments and real-time analytics.

If you don't need an in-memory data store, but your use case is relational data used primarily for analytics, these workloads are best supported by BigQuery.

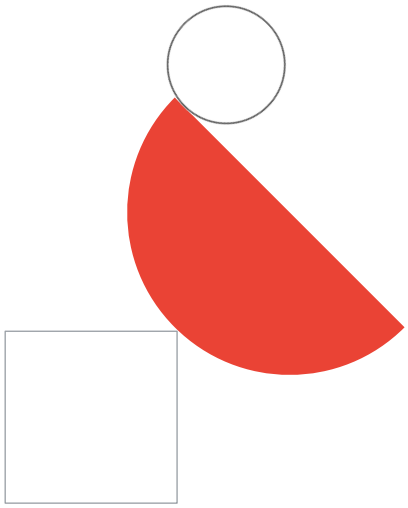
However, if your relational data workload isn't analytics the choice lies between Cloud Spanner and Cloud SQL.

If you don't need horizontal scaling or a globally available system, Cloud SQL is a cost-effective solution.

If Cloud SQL as a managed service is better than using or re-implementing your existing MySQL solution, refer to this solution on how to migrate from MySQL to Cloud SQL.

Lab Intro

Implementing Cloud SQL



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

Lab objectives

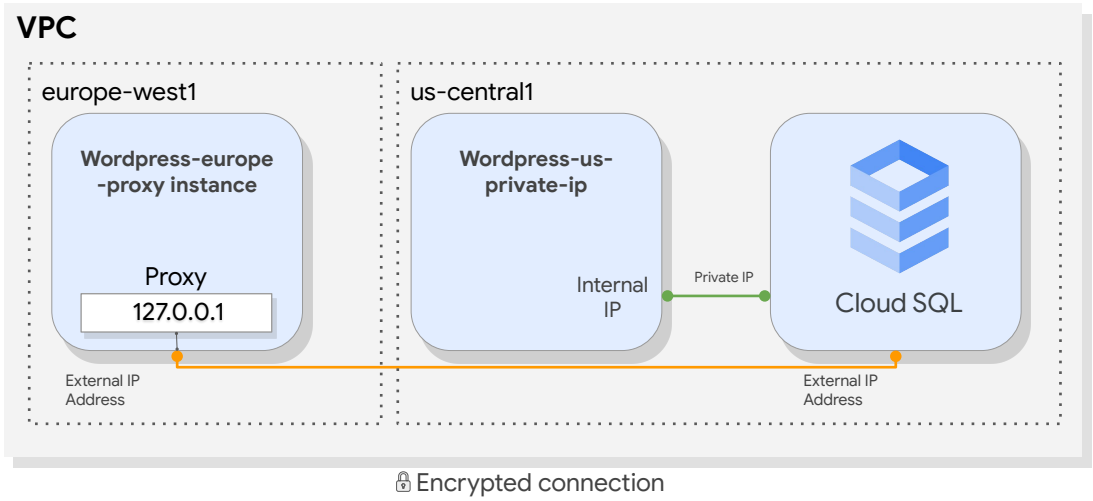
01 Create a Cloud SQL database

02 Configure a virtual machine to run a proxy

03 Create a connection between an application and Cloud SQL

04 Connect an application to Cloud SQL using Private IP address

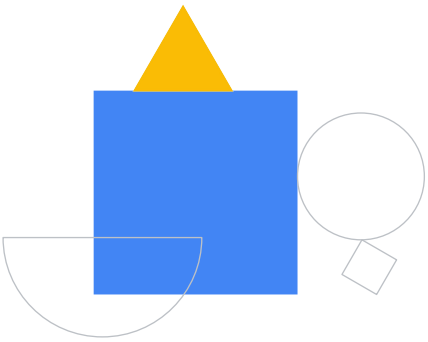
In this lab, you configure a Cloud SQL server and learn how to connect an application to it via a proxy over an external connection. You also configure a connection over a Private IP link that offers performance and security benefits. The app we chose to demonstrate in this lab is Wordpress, but the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



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.

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.

Cloud Spanner

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

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

Cloud Spanner

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.
[https://cloud.google.com/spanner/sla]

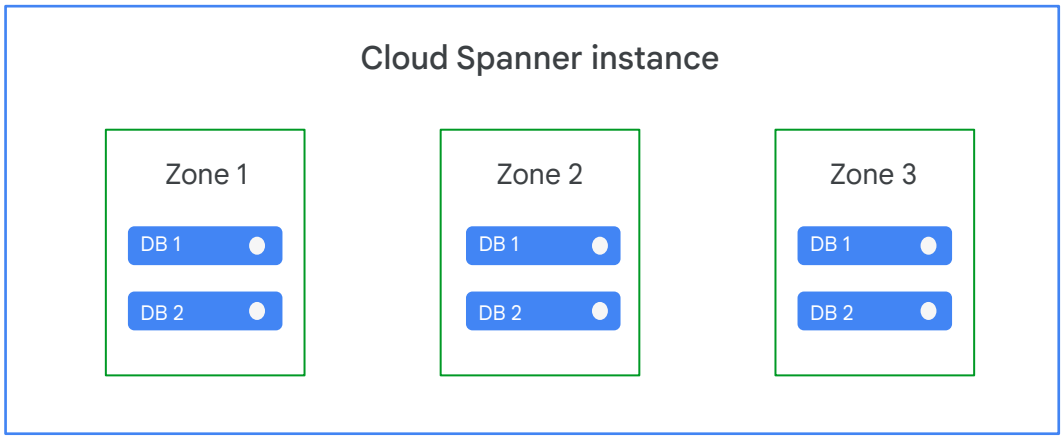
Characteristics

	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

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.

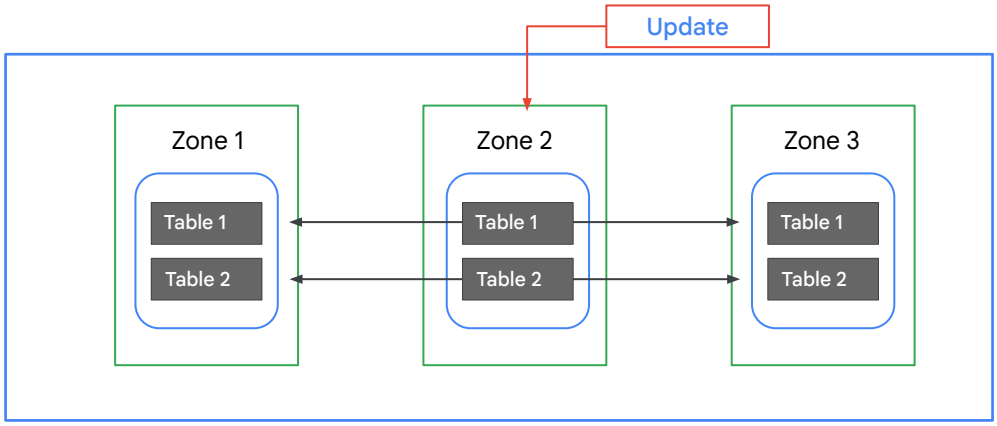
As mentioned, Cloud Spanner offers the best of the relational and non-relational worlds. These features allow for mission-critical uses cases, such as building consistent systems for transactions and inventory management in the financial services and retail industries. To better understand how all of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

Cloud Spanner architecture



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.

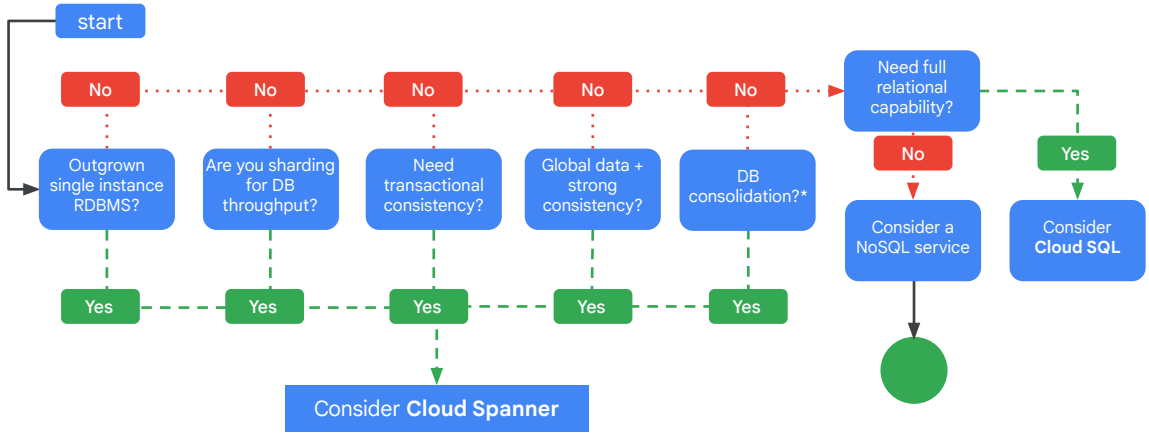
Data replication is synchronized across zones using Google's global fiber network



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.

Choosing Cloud Spanner



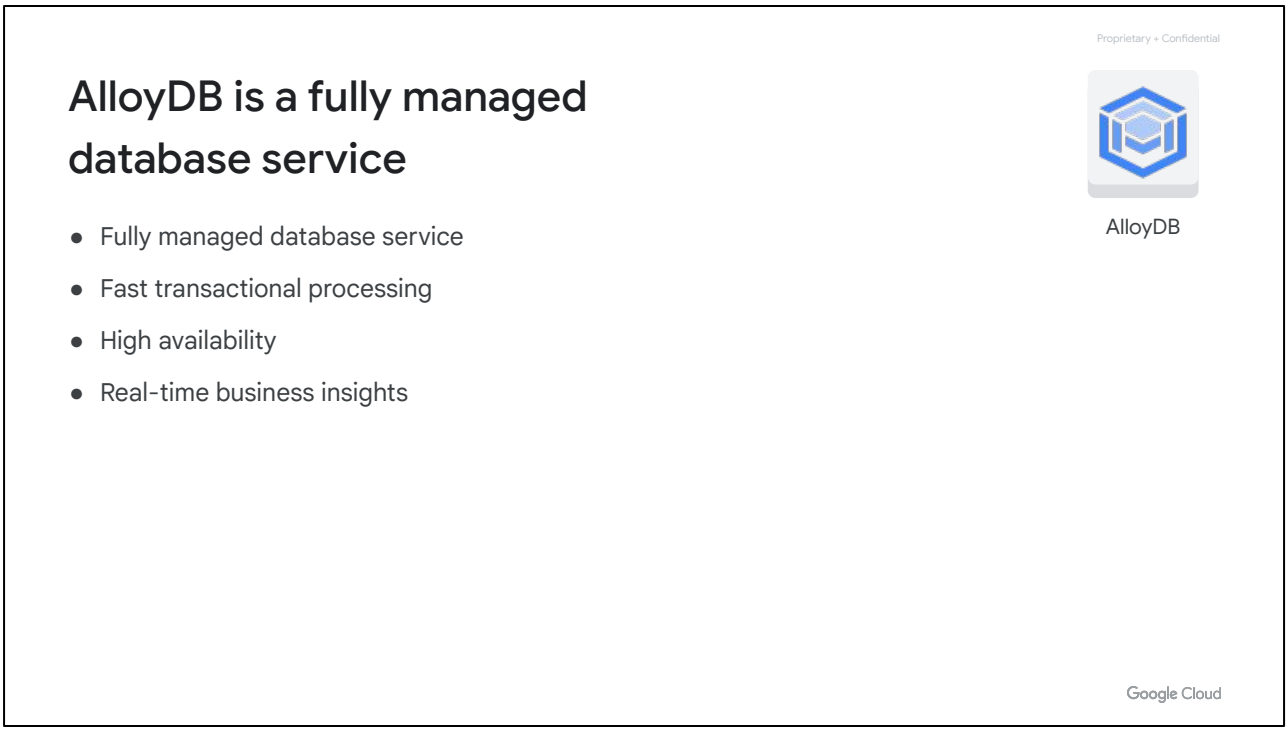
If you have outgrown any relational database, are sharding your databases for throughput high performance, need transactional consistency, global data and strong consistency, or just want to consolidate your database, consider using Cloud Spanner.

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]

AlloyDB

Let’s now talk about AlloyDB.



AlloyDB for PostgreSQL is a fully managed, PostgreSQL-compatible database service that's designed for demanding workloads such as hybrid transactional and analytical processing. AlloyDB pairs a Google-built database engine with a cloud-based, multi-node architecture to deliver enterprise-grade performance, reliability, and availability.

AlloyDB automates administrative tasks, such as backups, replication, patching, and capacity management. AlloyDB also uses adaptive algorithms and machine learning for PostgreSQL vacuum management, storage and memory management, data tiering, and analytics acceleration.

AlloyDB provides fast transactional processing, more than 4 times faster than standard PostgreSQL for transactional workloads. It's suitable for demanding enterprise workloads, including workloads that require high transaction throughput, large data sizes, or multiple read replicas.

AlloyDB provides high-availability and an 99.99% uptime SLA, inclusive of maintenance.

AlloyDB also provides real-time business insights and is up to 100 times faster than standard PostgreSQL for analytical queries. Built-in integration with Vertex AI, Google's artificial intelligence platform, lets you call machine learning models.

Firestore

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

Firestore is a NoSQL document database

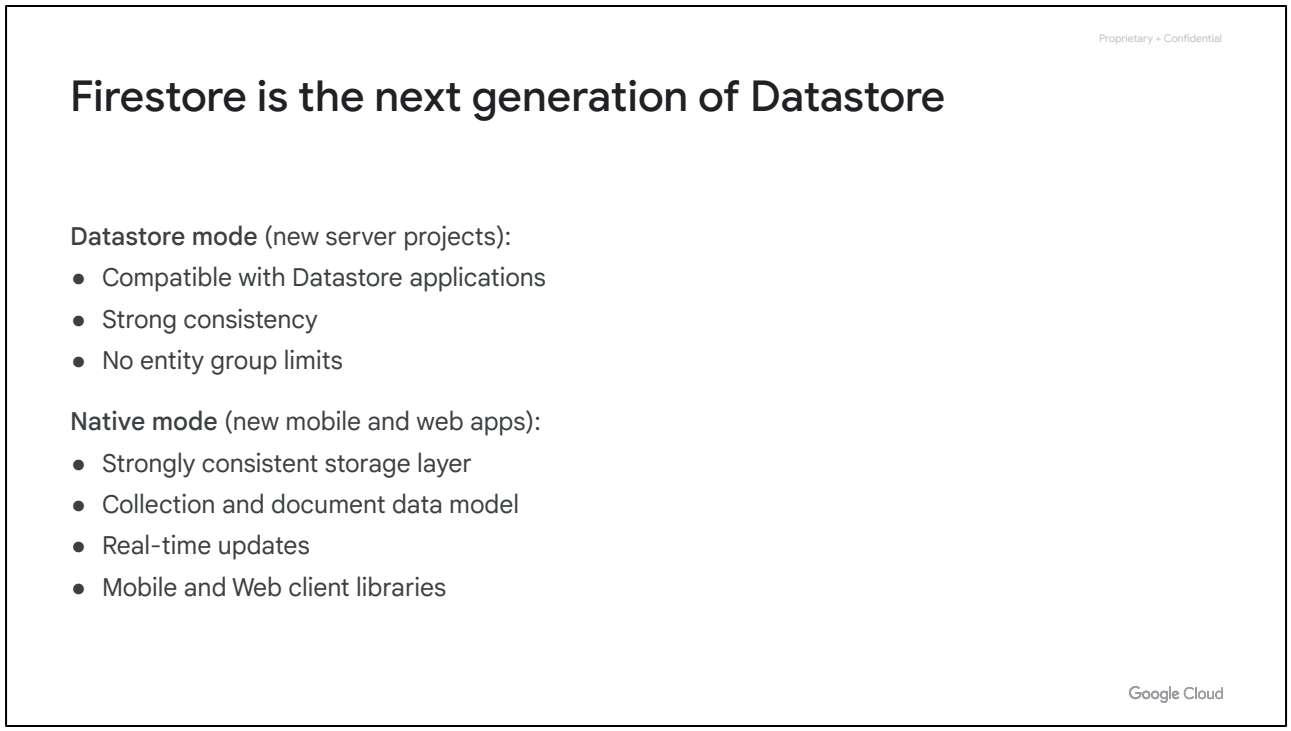
- Simplifies storing, syncing, and querying data
- Mobile, web, and IoT apps at global scale
- Live synchronization and offline support
- Security features
- ACID transactions
- Multi-region replication
- Powerful query engine

Firestore

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 Google Cloud accelerate building truly serverless apps.

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. 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.



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

This removes the following 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.

Firestore in Native mode introduces new features such as:

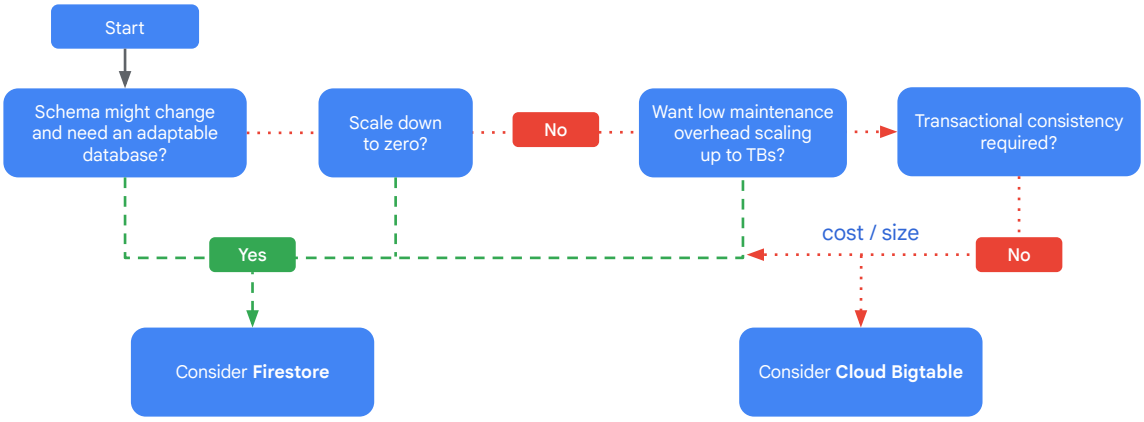
- A new, strongly consistent storage layer
- A collection and document data model
- Real-time updates
- Mobile and Web client libraries

Firestore is backward compatible with Datastore, but the new data model, real-time updates, and mobile and web client library features are not. To access all of the new Firestore features, you must use Firestore in Native mode. A general guideline is to use Firestore in Datastore mode for new server projects, and Native mode for new mobile and web apps.

As the next generation of Datastore, Firestore is compatible with all Datastore APIs and client libraries. Existing Datastore users will be live-upgraded to Firestore automatically at a future date. For more information, refer to the documentation.

https://cloud.google.com/datastore/docs/firestore-or-datastore
https://cloud.google.com/datastore/docs/upgrade-to-firestore

Choosing Firestore



To summarize, let's explore this decision tree to help you determine whether 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 Firestore.

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

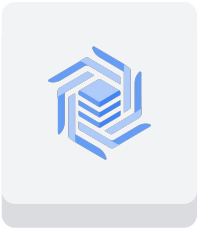
I will cover Cloud Bigtable next.

Cloud Bigtable

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



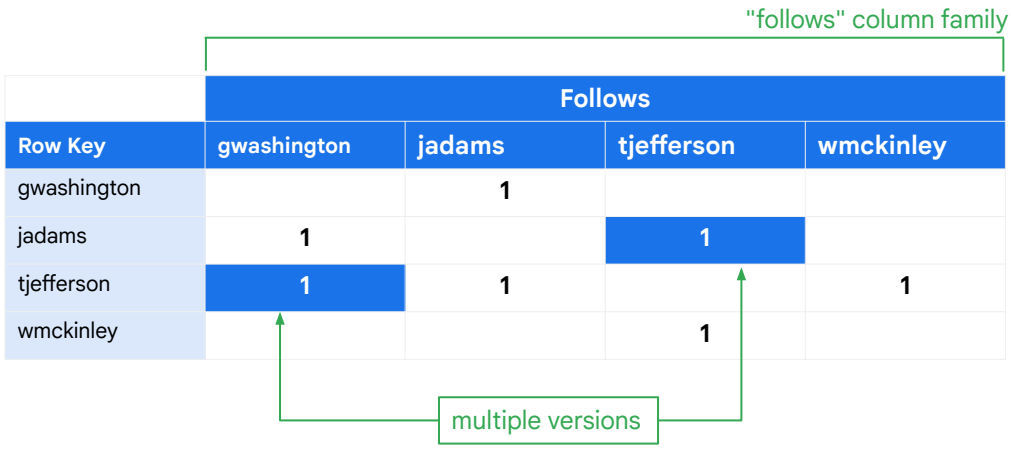
Cloud Bigtable

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



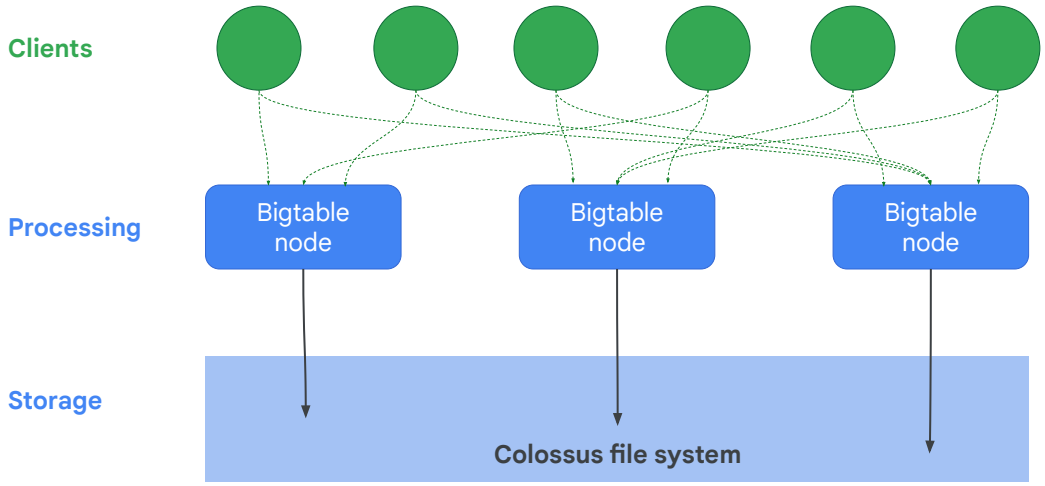
Cloud Bigtable stores data in massively scalable tables, each of which is a sorted key/value map. The table is composed of rows, each of which typically describes a single entity, and columns, which contain individual values for each row. Each row is indexed by a single row key, and columns that are 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.

Processing is separated from storage

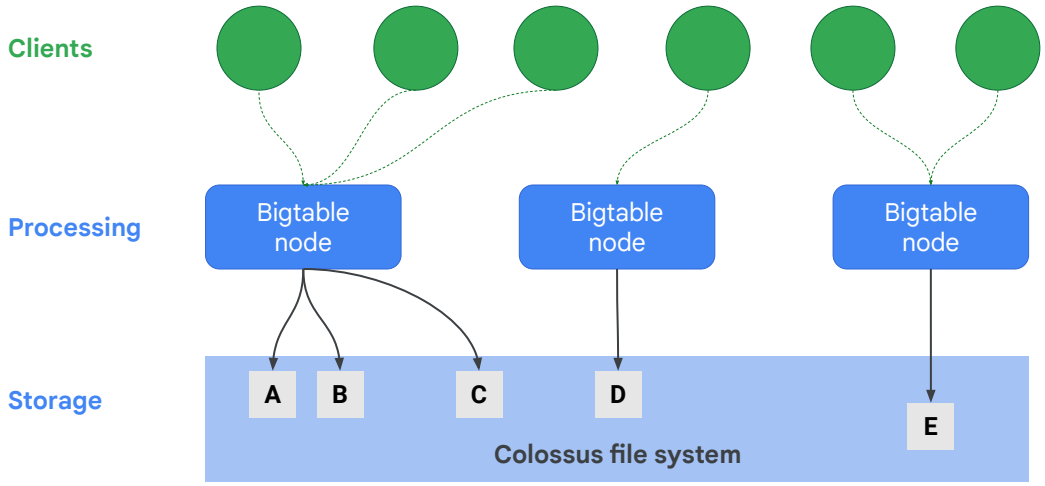


This diagram shows a simplified version of Cloud Bigtable's overall architecture. It illustrates that processing, which is done through a front-end server pool and nodes, is handled separately from the storage.

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.

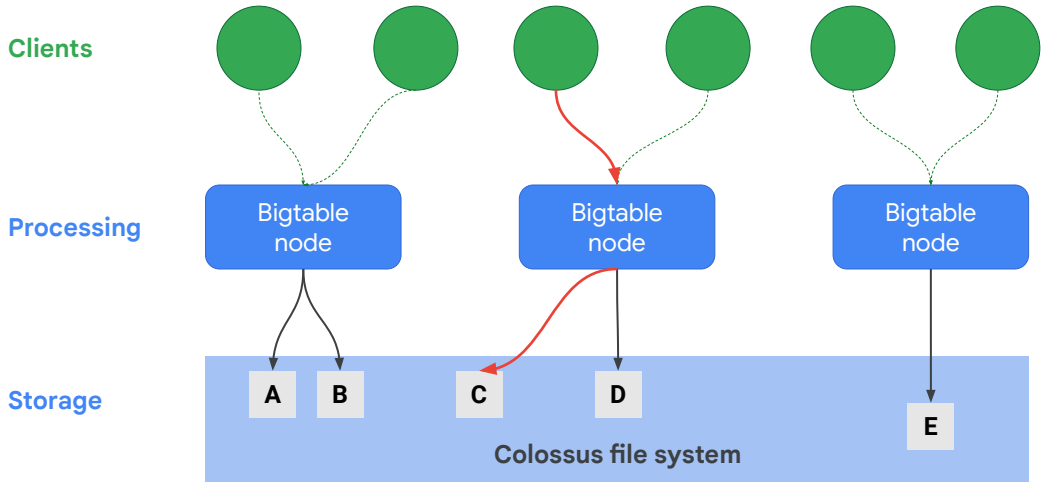
Tablets are stored on Colossus, which is Google's file system, in SSTable format. An SSTable provides a persistent, ordered immutable map from keys to values, where both keys and values are arbitrary byte strings.

Learns access patterns



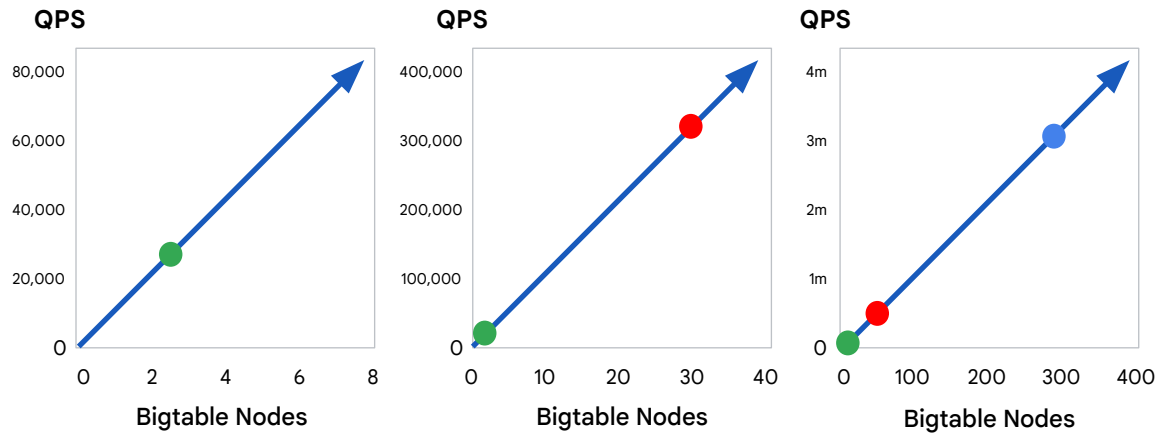
As I mentioned earlier, Cloud Bigtable learns to adjust to specific access patterns. If a certain Bigtable node is frequently accessing a certain subset of data...

Rebalances without moving data



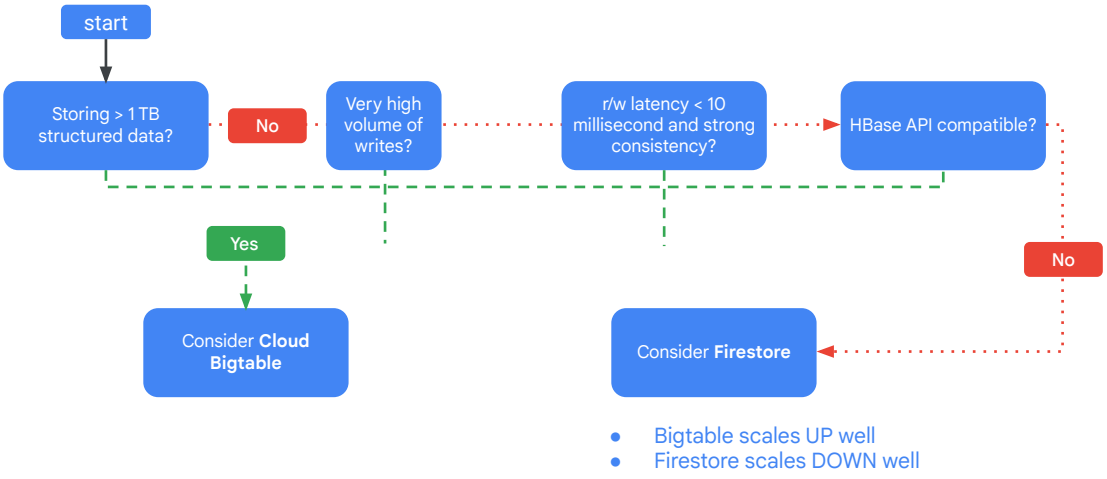
… Cloud Bigtable will update the indexes so that other nodes can distribute that workload evenly, as shown here.

Throughput scales linearly



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.

Choosing Cloud Bigtable



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 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.

Memorystore

Let me give you a quick overview of Memorystore.

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

Memorystore

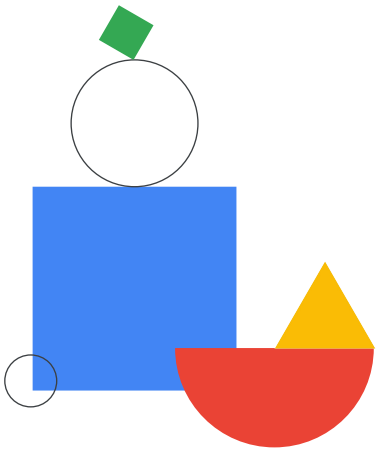
Memorystore for Redis provides a fully managed in-memory data store service built on scalable, secure, and highly available infrastructure managed by Google. Applications running on Google Cloud can achieve extreme performance by leveraging the highly scalable, available, secure Redis service without 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.

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. Memorystore can support instances up to 300 gigabytes and network throughput of 12 gigabits per second.

Because Memorystore for Redis is fully compatible with the Redis protocol, you can lift and shift your applications from open source Redis to 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



In this module, we covered the different storage and database services that Google Cloud offers. Specifically, you learned about Cloud Storage, a fully managed object store; Filestore, a fully managed file storage service; Cloud SQL, a fully managed MySQL and PostgreSQL database service; Cloud Spanner, a relational database service with transactional consistency, global scale and high availability; AlloyDB, a fully managed, PostgreSQL-compatible database service; Firestore, a fully managed NoSQL document database; Cloud Bigtable, a fully managed NoSQL wide-column database; and 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 Google Cloud that cover data strategy.