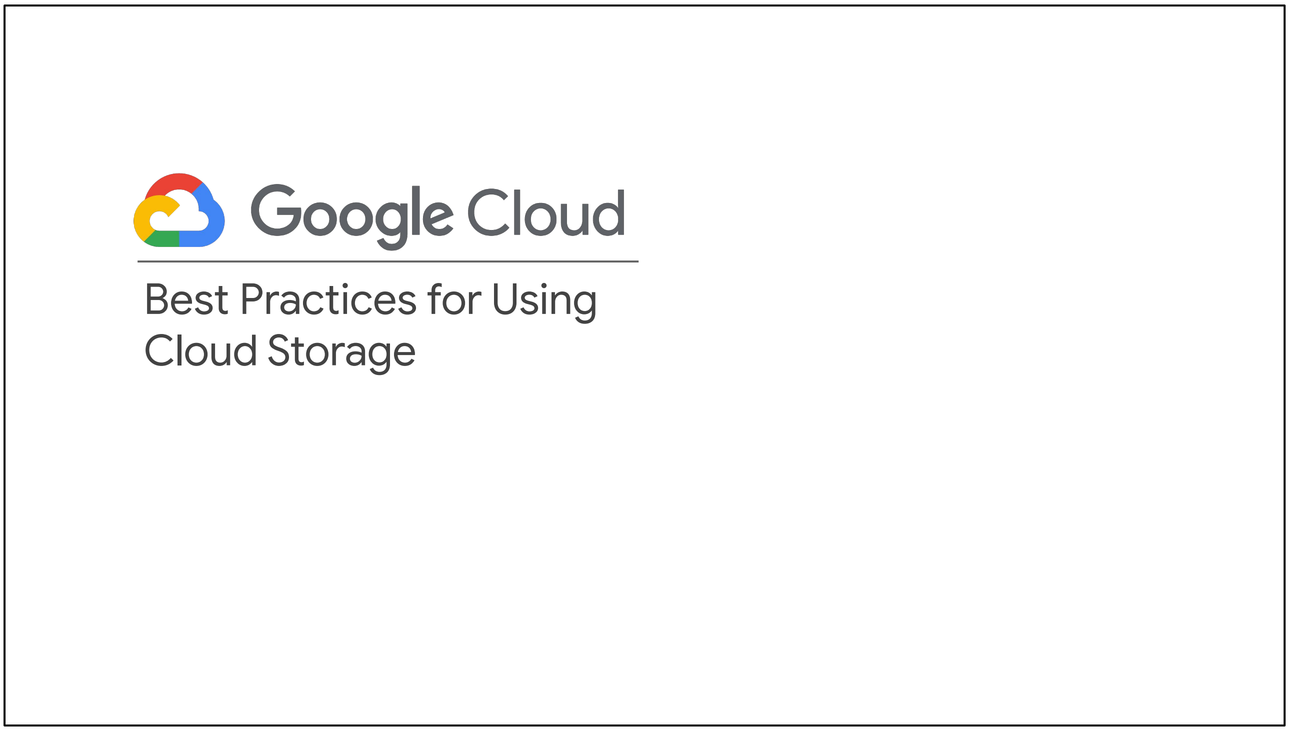


Google Cloud

Best Practices for Using Cloud Storage

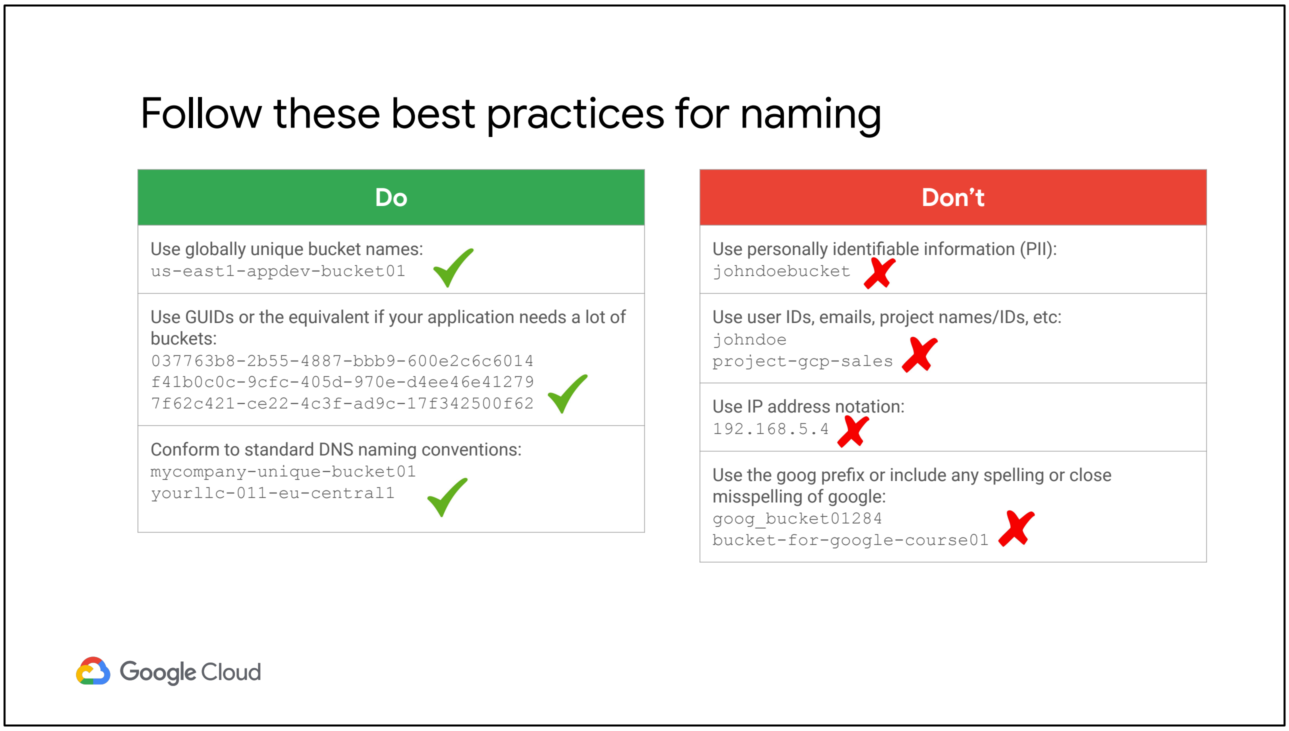
Course name: Developing Applications with Google Cloud Platform
Module name: Best Practices for Using Cloud Storage

Featured products: Cloud Storage

Qwiklabs: Storing image and video files in Cloud Storage

10 minutes

For more information, see: https://cloud.google.com/storage/docs/best-practices



Follow these best practices for naming

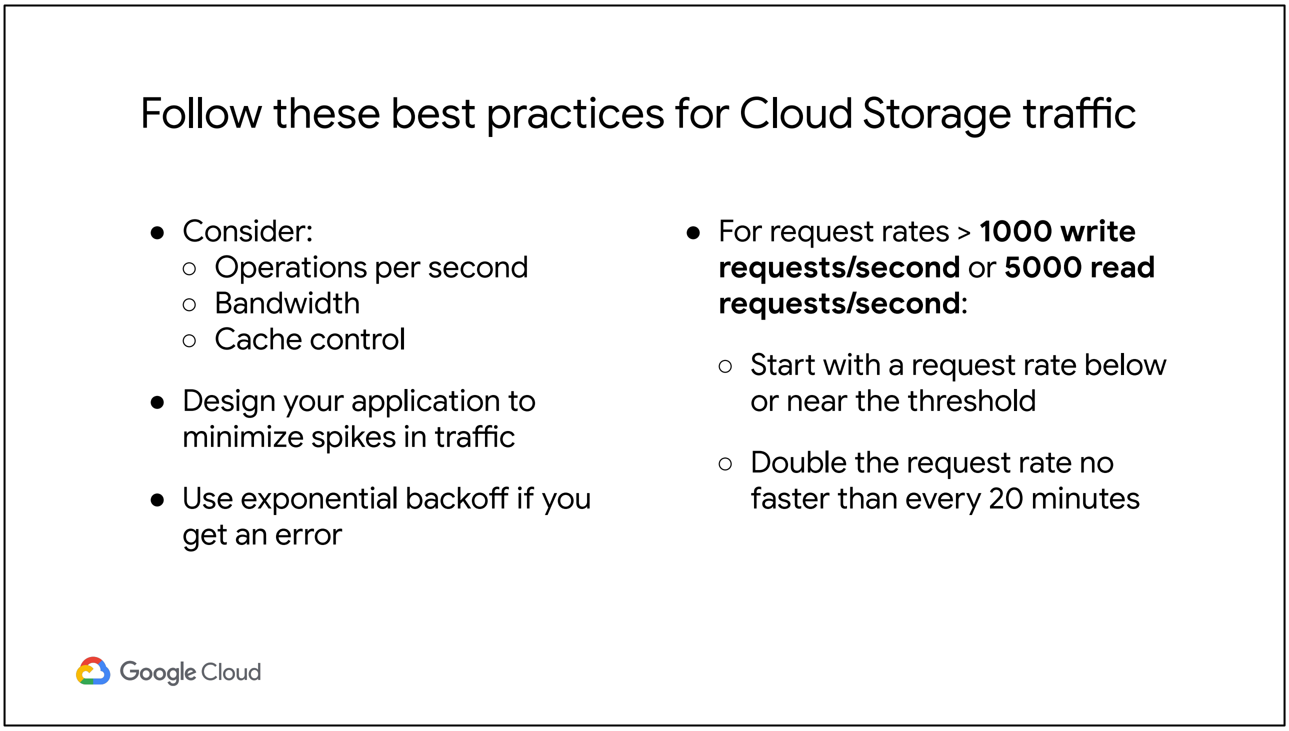
Do	Don't
Use globally unique bucket names: `us-east1-appdev-bucket01` ✓	Use personally identifiable information (PII): `johndoebucket` ✗
Use GUIDs or the equivalent if your application needs a lot of buckets: `037763b8-2b55-4887-bbb9-600e2c6c6014` `f41b0c0c-9cfc-405d-970e-d4ee46e41279` `7f62c421-ce22-4c3f-ad9c-17f342500f62` ✓	Use user IDs, emails, project names/IDs, etc: `johndoe` `project-gcp-sales` ✗
Conform to standard DNS naming conventions: `mycompany-unique-bucket01` `yourllc-011-eu-central1` ✓	Use IP address notation: `192.168.5.4` ✗
	Use the goog prefix or include any spelling or close misspelling of google: `goog_bucket01284` `bucket-for-google-course01` ✗

Google Cloud

Google Cloud Storage bucket names are global and publicly visible and must be unique across the entire Google Cloud Storage. If your applications require many buckets, use GUIDs or the equivalent for bucket names. Your application should have retry logic in place to handle name collisions. Consider keeping a list to cross-reference your buckets.

Avoid using any information in bucket names that can be used to probe for the existence of other resources. Use caution if putting personally identifiable information in object or bucket names, because they appear in URLs. Bucket names should conform to standard DNS naming conventions, because the bucket name can appear in a DNS record as part of a CNAME redirect.

If you use part of your company domain name in a bucket name (forward or reverse), Google will try to verify that you own the domain. Avoid using your company domain name in a bucket unless your DNS admin can verify ownership.
For more information, see:
https://cloud.google.com/storage/docs/bucket-naming#requirements



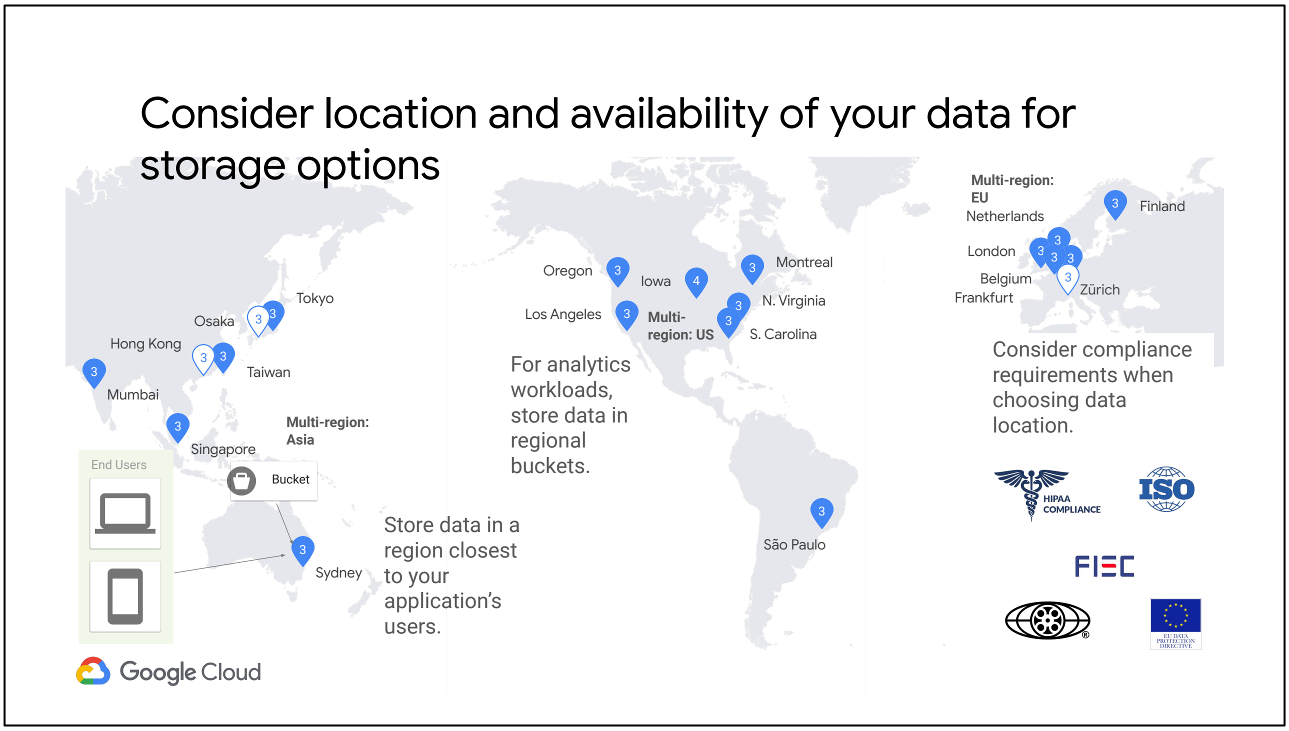
Follow these best practices for Cloud Storage traffic

- Consider:
  - Operations per second
  - Bandwidth
  - Cache control
- Design your application to minimize spikes in traffic
- Use exponential backoff if you get an error
- For request rates > **1000 write requests/second** or **5000 read requests/second**:
  - Start with a request rate below or near the threshold
  - Double the request rate no faster than every 20 minutes

Google Cloud

Consider operations per second, bandwidth (how much data will be sent over what time frame), and cache control when designing your application with Cloud Storage. If you specify the Cache-Control metadata on objects, read latency will be lowered on hot or frequently accessed objects. Use exponential backoff if you get an error. For more information on setting object metadata, see: https://cloud.google.com/storage/docs/viewing-editing-metadata#edit

Design your application so that it minimizes spikes in traffic; spread updates out throughout the day. Google Storage has no upper bound on request rate, but for best performance when scaling to high request rates, follow the request rate and access distribution guidelines: If your request rate is less than 1000 write requests per second or 5000 read requests per second, then no ramp-up is needed. If your request rate is expected to go over these thresholds, you should start with a request rate below or near the thresholds and then double the request rate no faster than every 20 minutes. For more information, see: https://cloud.google.com/storage/docs/request-rate



Consider location and availability of your data for storage options

For analytics workloads, store data in regional buckets.

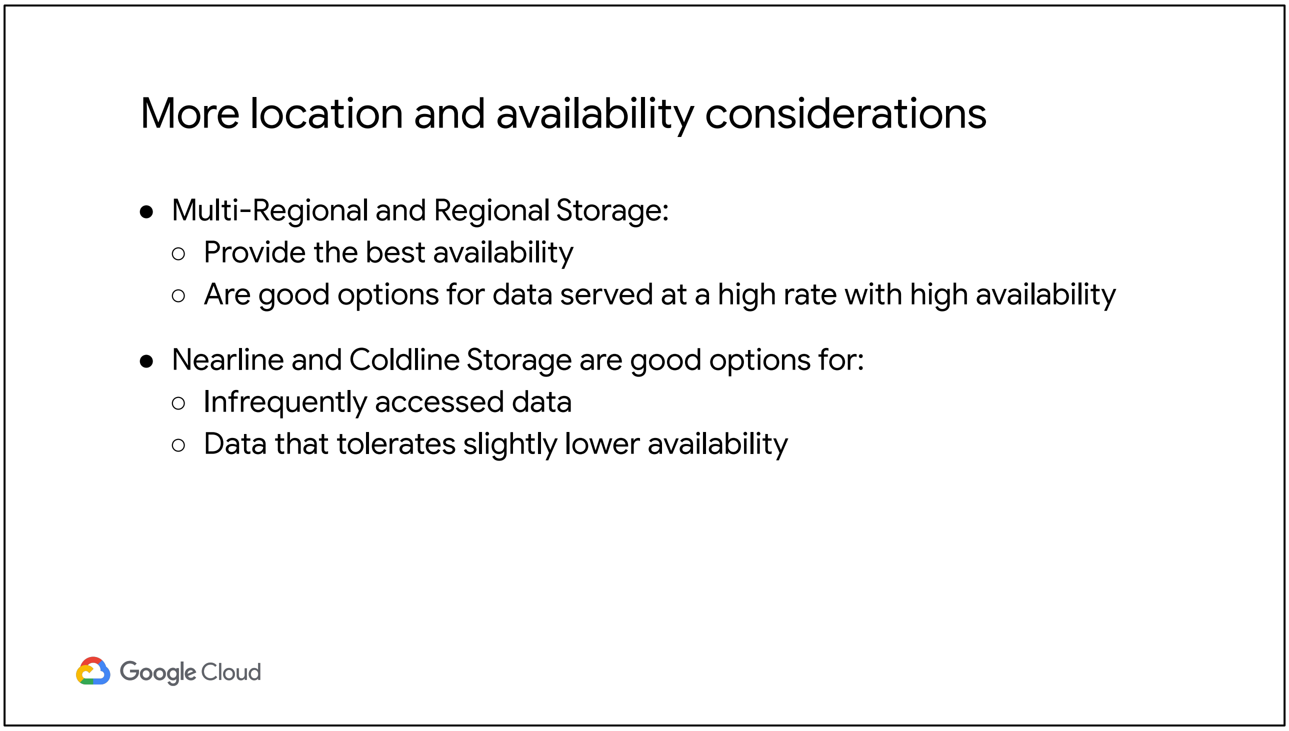
Store data in a region closest to your application's users.

Consider compliance requirements when choosing data location.

Consider the location and required availability of your data when choosing your storage options.

Store data in a region closest to your application's users, and consider region-specific compliance requirements when choosing data location. For analytics workloads, store your data in regional buckets to reduce network charges and for better performance (compared to multi-regional).

For more information, see: https://cloud.google.com/security/compliance



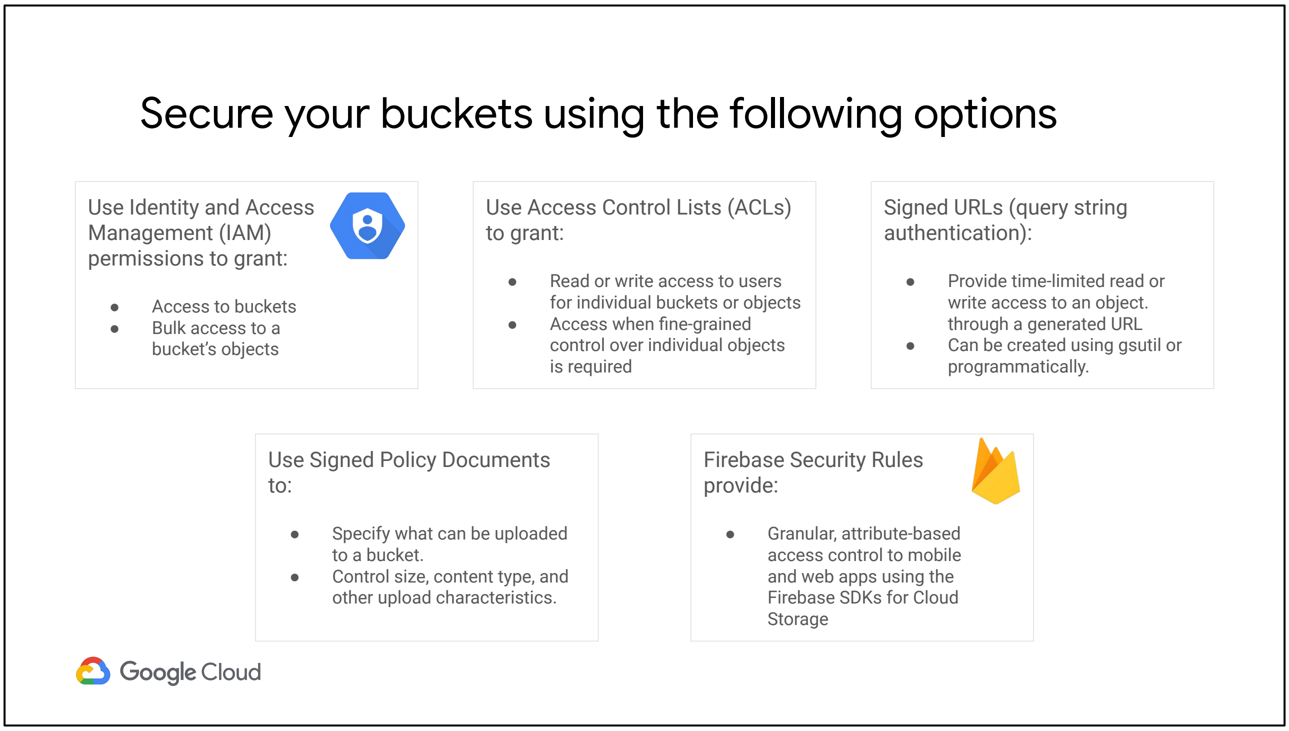
More location and availability considerations

- Multi-Regional and Regional Storage:
  - Provide the best availability
  - Are good options for data served at a high rate with high availability
- Nearline and Coldline Storage are good options for:
  - Infrequently accessed data
  - Data that tolerates slightly lower availability

Google Cloud

Multi-Regional Storage and Regional Storage provide the best availability, with the trade-off of a higher price, and are good options for data that is served at a high rate with high availability.

Nearline Storage and Coldline Storage are good options for infrequently accessed data and for data that tolerates slightly lower availability.



Secure your buckets using the following options

Use Identity and Access Management (IAM) permissions to grant:

- Access to buckets
- Bulk access to a bucket's objects

Use Access Control Lists (ACLs) to grant:

- Read or write access to users for individual buckets or objects
- Access when fine-grained control over individual objects is required

Signed URLs (query string authentication):

- Provide time-limited read or write access to an object. through a generated URL
- Can be created using gsutil or programmatically.

Use Signed Policy Documents to:

- Specify what can be uploaded to a bucket.
- Control size, content type, and other upload characteristics.

Firebase Security Rules provide:

- Granular, attribute-based access control to mobile and web apps using the Firebase SDKs for Cloud Storage

Google Cloud

You can control access to your Cloud Storage buckets and objects using these options.

You can use Identity and access Management (IAM) permissions to grant access to buckets and to provide bulk access to a bucket's objects. IAM permissions do not give you fine-grained control over individual objects. For more information, see: https://cloud.google.com/storage/docs/access-control/using-iam-permissions

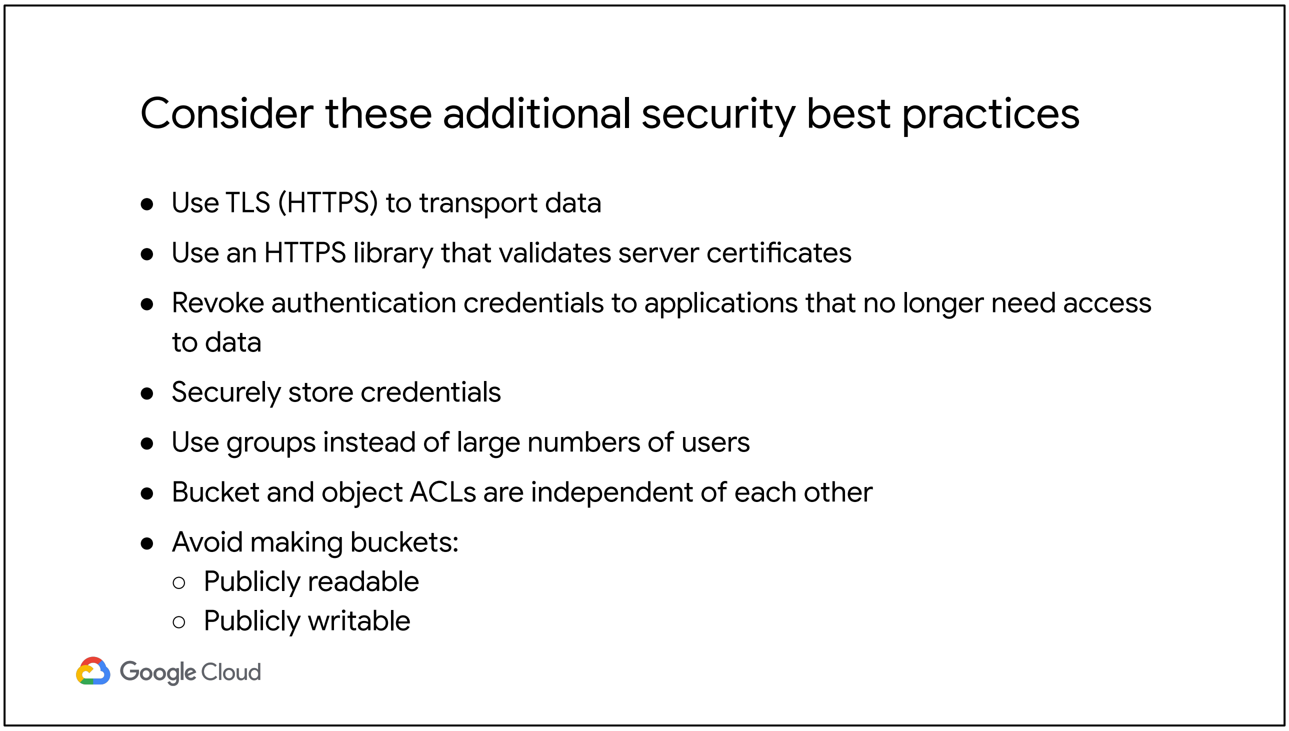
You can use Access Control Lists (ACLs) to grant read or write access to users for individual buckets or objects. It is recommended that you only use ACLs when you need fine-grained control over individual objects. For more information, see: https://cloud.google.com/storage/docs/access-control/create-manage-lists

Use signed URLs (query string authentication) to provide time-limited read or write access to an object through a URL you generate. The shared URL provides access to anyone it's shared with for the duration specified. You can create signed URLs using gsutil or programmatically with your application. For more information, see:
Create a signed URL using gsutil:
https://cloud.google.com/storage/docs/access-control/create-signed-urls-gsutil
Create a signed URL programatically:
https://cloud.google.com/storage/docs/access-control/create-signed-urls-program

Signed policy documents allow you to specify what can be uploaded to a bucket. They allow greater control over size, content type, and other upload characteristics

than using signed URLs. They are for website owners to allow visitors to upload files to Google Cloud Storage. Signed policy documents only work with form posts. For more information on signed policy documents, see:
https://cloud.google.com/storage/docs/xml-api/post-object#policydocument

Firebase security rules provide granular, attribute-based access control to mobile and web applications using the Firebase SDKs for Cloud Storage. For more information, see: https://firebase.google.com/docs/storage/security



Consider these additional security best practices

- Use TLS (HTTPS) to transport data
- Use an HTTPS library that validates server certificates
- Revoke authentication credentials to applications that no longer need access to data
- Securely store credentials
- Use groups instead of large numbers of users
- Bucket and object ACLs are independent of each other
- Avoid making buckets:
  - Publicly readable
  - Publicly writable

Google Cloud

Always use TLS (HTTPS) to transport your data when you can. This ensures that both your credentials and your data are protected as you transport data over the network.

Make sure that you use an HTTPS library that validates server certificates. A lack of server certificate validation makes your application vulnerable to man-in-the-middle attacks or other attacks. Be aware that HTTPS libraries shipped with certain commonly used implementation languages do not, by default, verify server certificates.

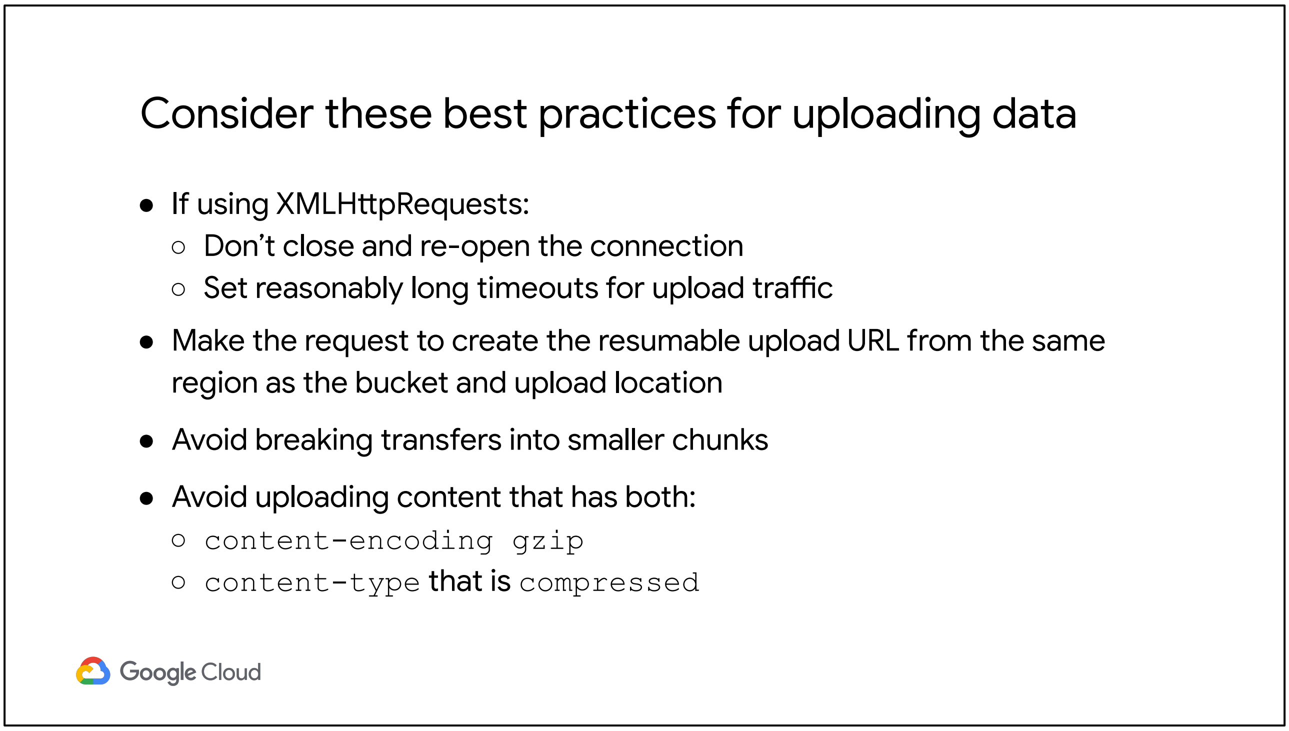
When applications no longer need access to your data, you should revoke their authentication credentials. To do this for Google services and APIs, sign in to your Google Account and click Authorizing applications and sites. On the next page, you can revoke access for applications by clicking Revoke Access next to the application.

Make sure that you securely store your credentials. This can be done differently depending on your environment and where you store your credentials.

Use groups in preference to explicitly listing large numbers of users. Not only does it scale better, it also provides a very efficient way to update the access control for a large number of objects all at once. It's cheaper because you don't need to make a request per-object to change the ACLs.
Before adding objects to a bucket, check that the default object ACLs are set to your requirements first. This could save you a lot of time updating ACLs for individual objects.

Bucket and object ACLs are independent of each other, which means that the ACLs on a bucket do not affect the ACLs on objects inside that bucket. It is possible for a user without permissions for a bucket to have permissions for an object inside the bucket.

The Google Cloud Storage access control system includes the ability to specify that objects are publicly readable. After a bucket has been made publicly readable, data on the internet can be copied to many places. It's effectively impossible to regain read control over an object written with this permission. The Google Cloud Storage access control system also includes the ability to specify that buckets are publicly writable. Although configuring a bucket this way can be convenient for various purposes, we recommend against using this permission: it can be abused for distributing illegal content, viruses, and other malware, and the bucket owner is legally and financially responsible for the content stored in their buckets.



Consider these best practices for uploading data

- If using XMLHttpRequests:
  - Don’t close and re-open the connection
  - Set reasonably long timeouts for upload traffic
- Make the request to create the resumable upload URL from the same region as the bucket and upload location
- Avoid breaking transfers into smaller chunks
- Avoid uploading content that has both:
  - `content-encoding gzip`
  - `content-type` that is `compressed`

Google Cloud

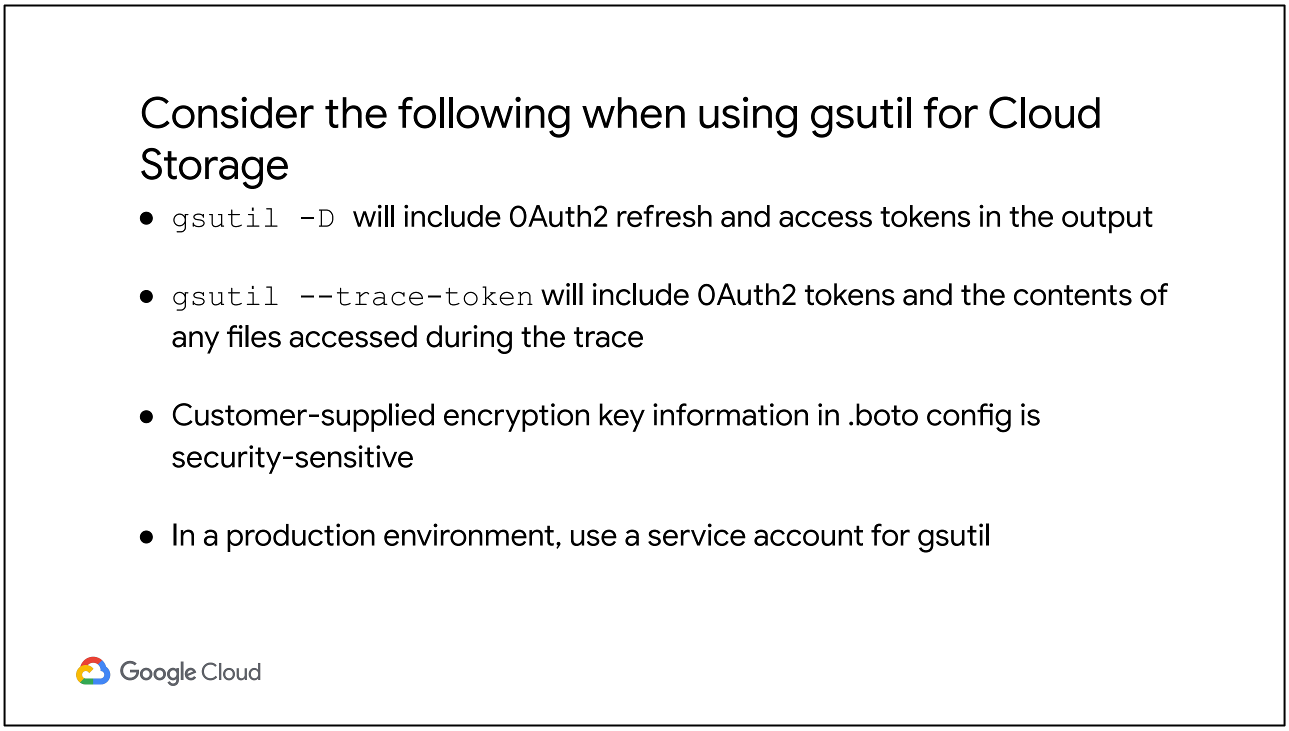
If you use XMLHttpRequest (XHR) callbacks to get progress updates and detect that progress has stalled, do not close and re-open the connection. Doing so creates a bad positive feedback loop during times of network congestion. When the network is congested, XHR callbacks can get backlogged behind the acknowledgement (ACK/NACK) activity from the upload stream, and closing and reopening the connection when this happens uses more network capacity at exactly the time when you can least afford it.

For upload traffic, set reasonably long timeouts. For a good end-user experience, you can set a client-side timer that updates the client status window with a message (e.g., "network congestion") when your application hasn't received an XHR callback for a long time. Don't just close the connection and try again when this happens.

If you use Google Compute Engine instances with processes that POST to Cloud Storage to initiate a resumable upload, you should use Compute Engine instances in the same locations as your Cloud Storage buckets. You can then use a geo IP service to pick the Compute Engine region to which you route customer requests, which will help keep traffic localized to a geo-region. For resumable uploads, the resumable session should stay in the region in which it was created. Doing so reduces cross-region traffic that arises when reading and writing the session state, which improves resumable upload performance.

Avoid breaking a transfer into smaller chunks if possible, and instead upload the entire content in a single chunk. Avoiding chunking removes fixed latency costs, improves throughput, and reduces QPS against Google Cloud Storage.

If possible, avoid uploading content that has both content-encoding: gzip and a content-type that is compressed, because this may lead to unexpected behavior. You can also use decompressive transcoding (automatically decompressing a file for the requestor) by storing the file in gzip format in Cloud Storage and setting the associated metadata to Content-Encoding: gzip. For more information, see: https://cloud.google.com/storage/docs/transcoding#decompressive_transcoding



Consider the following when using gsutil for Cloud Storage

- `gsutil -D` will include OAuth2 refresh and access tokens in the output
- `gsutil --trace-token` will include OAuth2 tokens and the contents of any files accessed during the trace
- Customer-supplied encryption key information in .boto config is security-sensitive
- In a production environment, use a service account for gsutil

Google Cloud

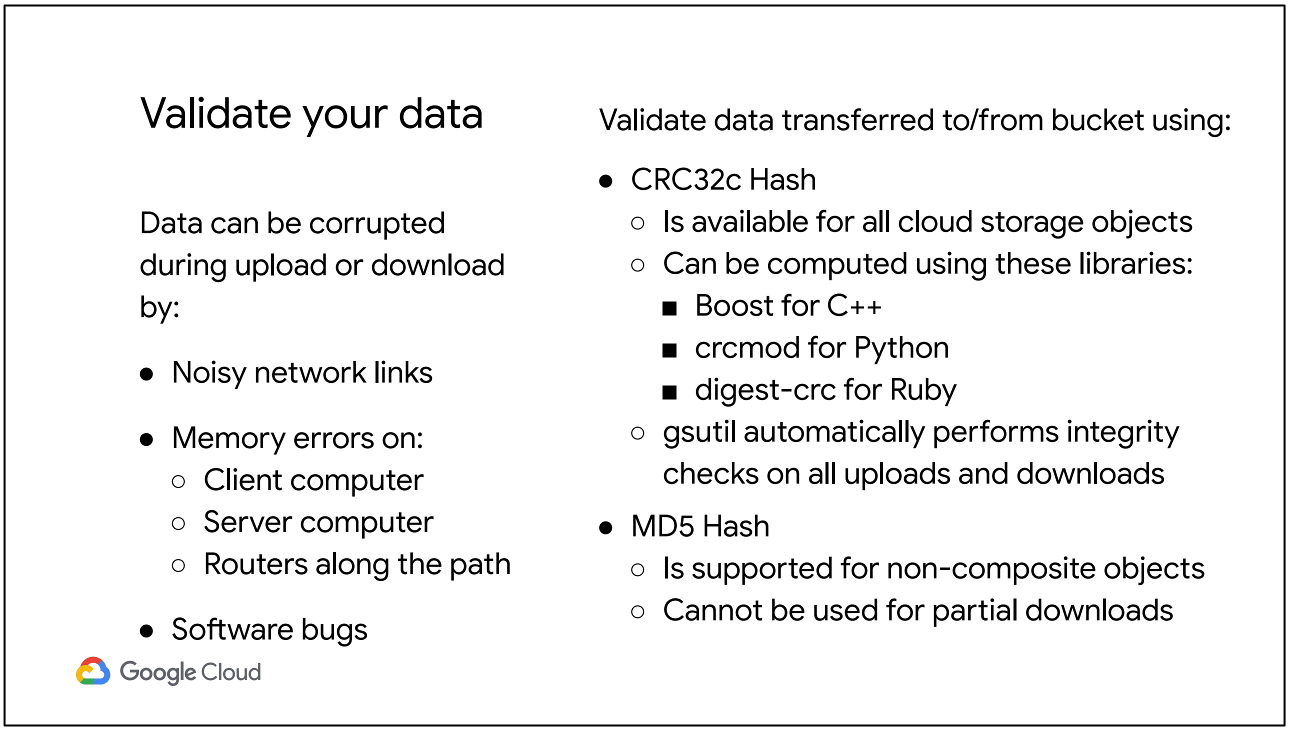
If you run gsutil -D (to generate debugging output) it will include OAuth2 refresh and access tokens in the output. Make sure to redact this information before sending this debug output to anyone during troubleshooting/tech support interactions.

If you run gsutil --trace-token (to send a trace directly to Google), sensitive information like OAuth2 tokens and the contents of any files accessed during the trace may be included in the content of the trace.

Customer-supplied encryption key information in the .boto configuration is security sensitive. The proxy configuration information in the .boto configuration is security-sensitive, especially if your proxy setup requires user and password information. Even if your proxy setup doesn't require user and password information, the host and port number for your proxy is often considered security-sensitive. Protect access to your .boto configuration file.

If you are using gsutil from a production environment, use service account credentials instead of individual user account credentials. These credentials were designed for such use and protect you from losing access when an employee leaves your company.

For more information, see:
https://cloud.google.com/storage/docs/gsutil/addlhelp/SecurityandPrivacyConsiderations#recommended-user-precautions



Validate your data

Data can be corrupted during upload or download by:

- Noisy network links
- Memory errors on:
  - Client computer
  - Server computer
  - Routers along the path
- Software bugs

Validate data transferred to/from bucket using:

- CRC32c Hash
  - Is available for all cloud storage objects
  - Can be computed using these libraries:
    - Boost for C++
    - crcmod for Python
    - digest-crc for Ruby
  - gsutil automatically performs integrity checks on all uploads and downloads
- MD5 Hash
  - Is supported for non-composite objects
  - Cannot be used for partial downloads

Google Cloud

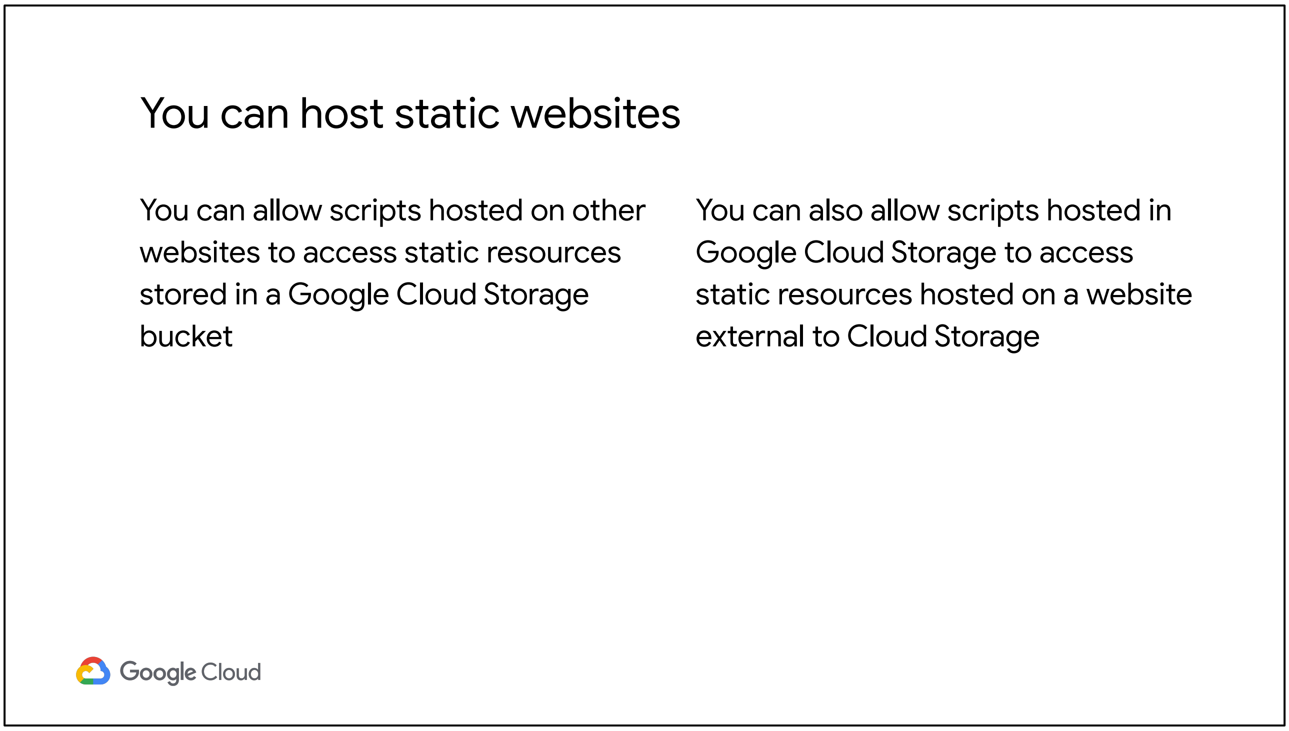
Data can be corrupted while it is uploaded to or downloaded from the cloud by noisy network links, memory errors on client or server computers or routers along the path, and software bugs. Validate the data you transfer to/from buckets using either CRC32c or MD5 checksums.

Google Cloud Storage supports server-side validation for uploads, but client-side validation is also a common approach. If your application has already computed the object's MD5 or CRC32c before starting the upload, you can supply it with the upload request, and Google Cloud Storage will create the object only if the hash you provided matches the value Google calculated. Alternatively, users can perform client-side validation by issuing a request for the new object's metadata, comparing the reported hash value, and deleting the object in case of a mismatch.

All Google Cloud Storage objects have a CRC32c hash. Libraries for computing CRC32c include Boost for C++, crcmod for Python, and digest-crc for Ruby. Java users can find an implementation of the algorithm in the GoogleCloudPlatform crc32 Java project. Gsutil automatically performs integrity checks on all uploads and downloads. Additionally, you can use the "gsutil hash" command to calculate a CRC for any local file.

MD5 hashes are supported for non-composite objects. The MD5 hash only applies to a complete object so it cannot be used to integrity check partial downloads cause by performing a range GET.

For more information, see:
Hashes and e-tags: https://cloud.google.com/storage/docs/hashes-etags
CRC32C and Installing crcmod:
https://cloud.google.com/storage/docs/gsutil/addlhelp/CRC32CandInstallingcrcmod



You can host static websites

You can allow scripts hosted on other websites to access static resources stored in a Google Cloud Storage bucket

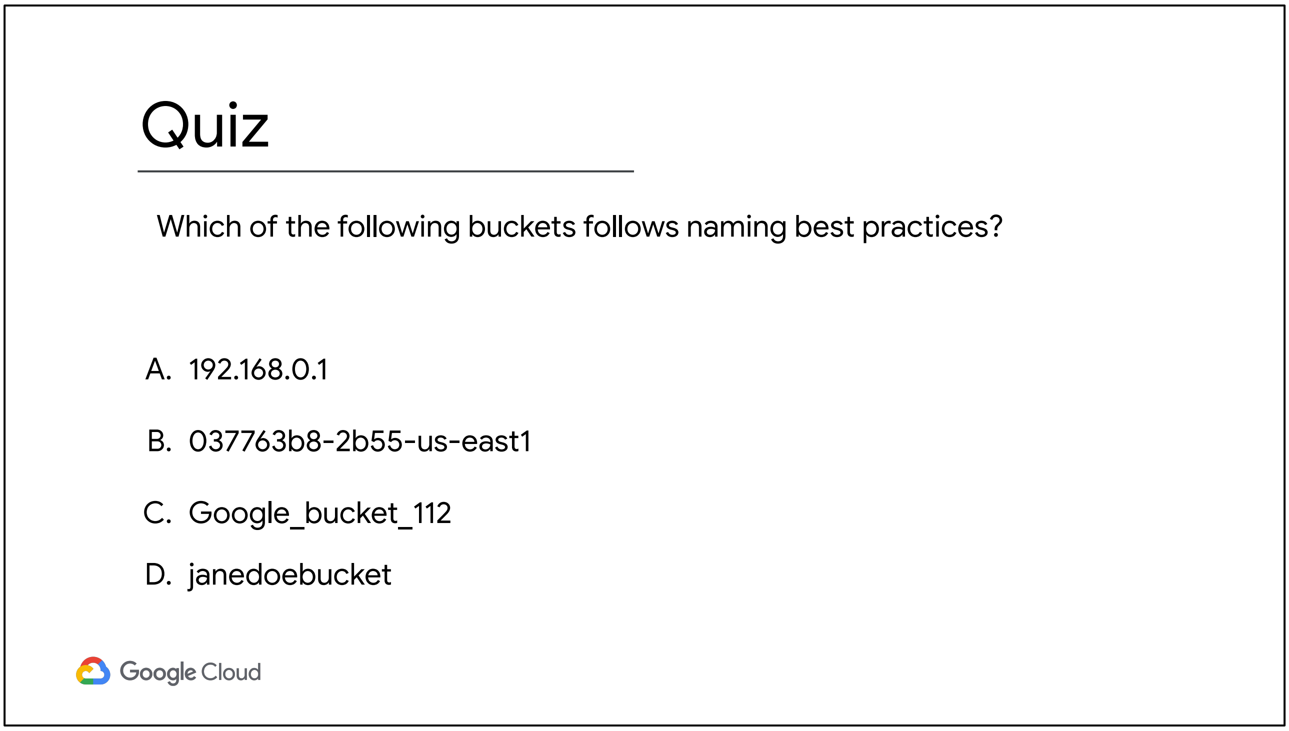
You can also allow scripts hosted in Google Cloud Storage to access static resources hosted on a website external to Cloud Storage

Google Cloud

The Cross-Origin Resource Sharing (CORS) topic describes how to allow scripts hosted on other websites to access static resources stored in a Google Cloud Storage bucket.

You can also allow scripts hosted in Google Cloud Storage to access static resources hosted on a website external to Cloud Storage. The website is serving CORS headers so that content on storage.googleapis.com is allowed access. It is recommended that you dedicate a specific bucket for this data access.

For more information, see: https://cloud.google.com/storage/docs/cross-origin



Quiz

Which of the following buckets follows naming best practices?

A. 192.168.0.1

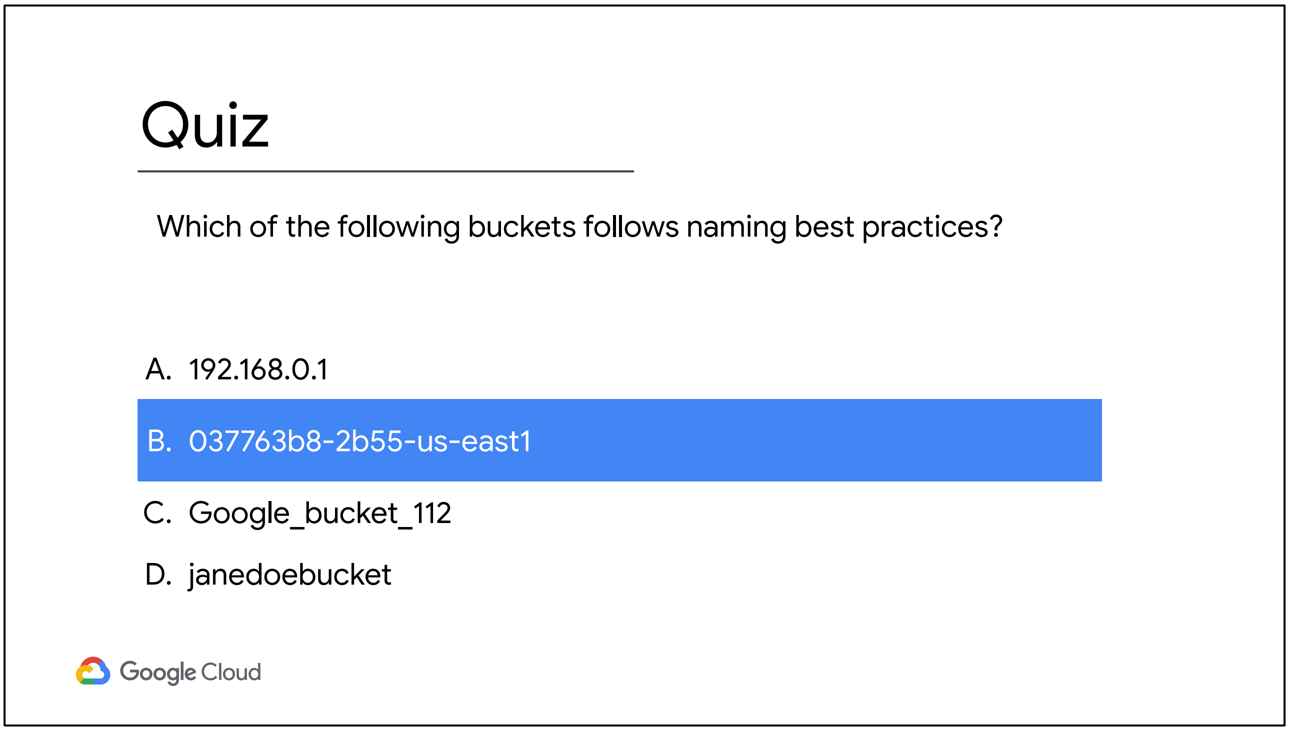
B. 037763b8-2b55-us-east1

C. Google_bucket_112

D. janedoebucket

Which of the following buckets follows naming best practices?

a. 192.168.0.1
b. 037763b8-2b55-us-east-1
c. google_bucket_112
d. janedoebucket



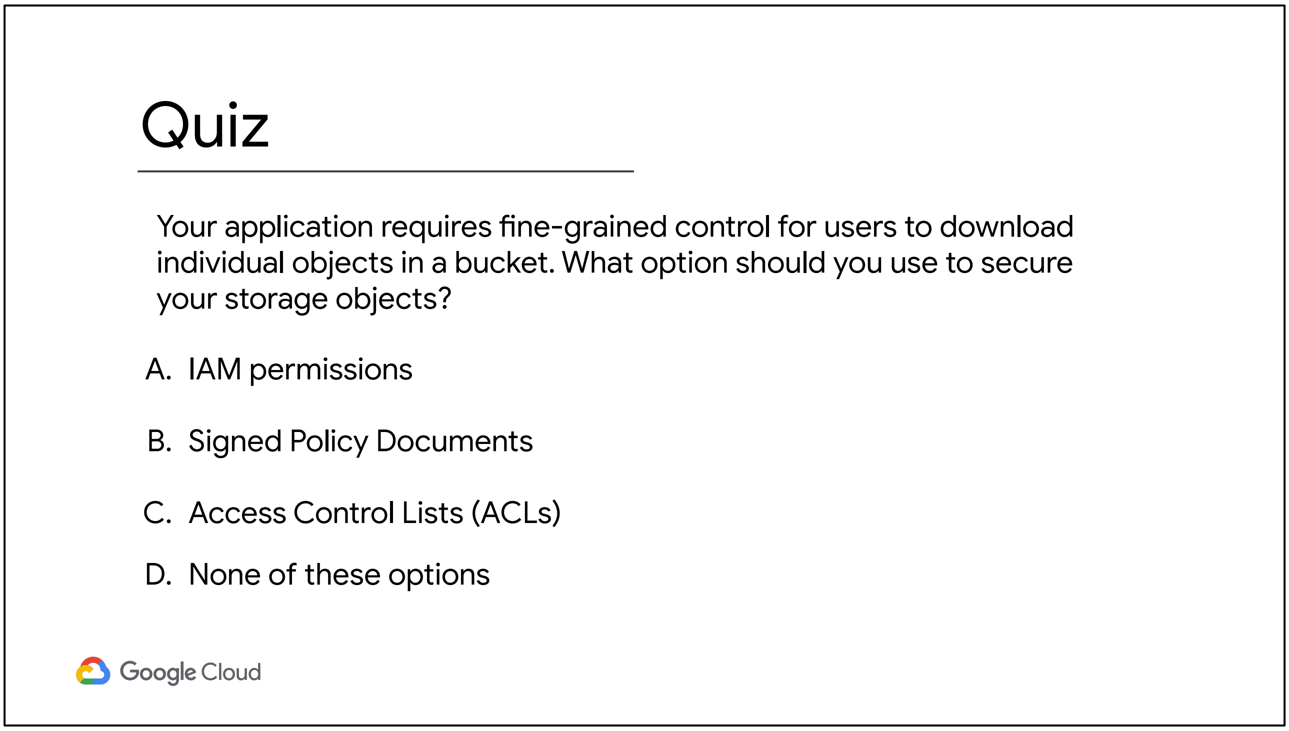
Quiz

Which of the following buckets follows naming best practices?

A. 192.168.0.1
B. 037763b8-2b55-us-east1
C. Google_bucket_112
D. janedoebucket

Which of the following buckets follows naming best practices?

a. 192.168.0.1
b. 037763b8-2b55-us-east-1*
c. google_bucket_112
d. janedoebucket



Quiz

Your application requires fine-grained control for users to download individual objects in a bucket. What option should you use to secure your storage objects?

A. IAM permissions

B. Signed Policy Documents

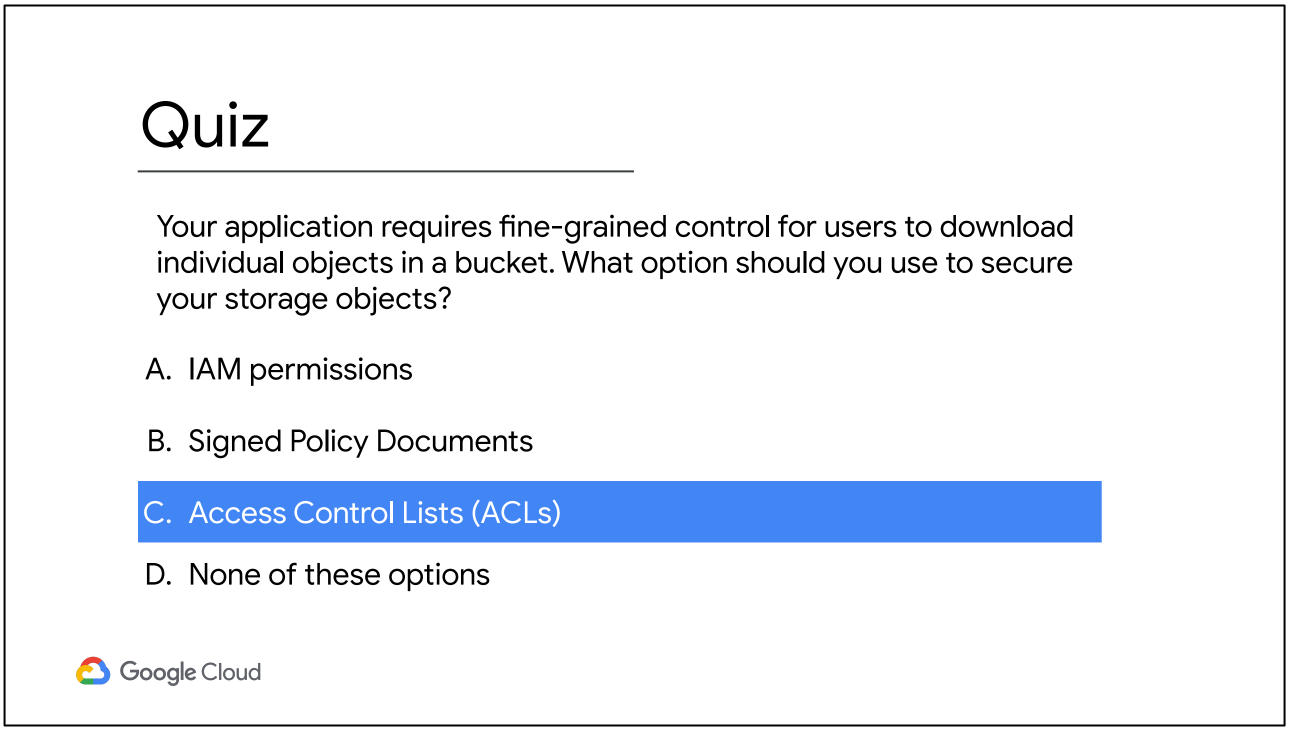
C. Access Control Lists (ACLs)

D. None of these options

Google Cloud

Your application requires fine-grained control for users to download individual objects in a bucket. What option should you use to secure your storage objects?

a. IAM permissions
b. Signed Policy Documents
c. Access Control Lists (ACLs)
d. None of these options



Quiz

Your application requires fine-grained control for users to download individual objects in a bucket. What option should you use to secure your storage objects?

A. IAM permissions

B. Signed Policy Documents

[✓ C. Access Control Lists (ACLs)]

D. None of these options

Google Cloud

Your application requires fine-grained control for users to download individual objects in a bucket. What option should you use to secure your storage objects?

a. IAM permissions
b. Signed Policy Documents
c. Access Control Lists (ACLs)*
d. None of these options

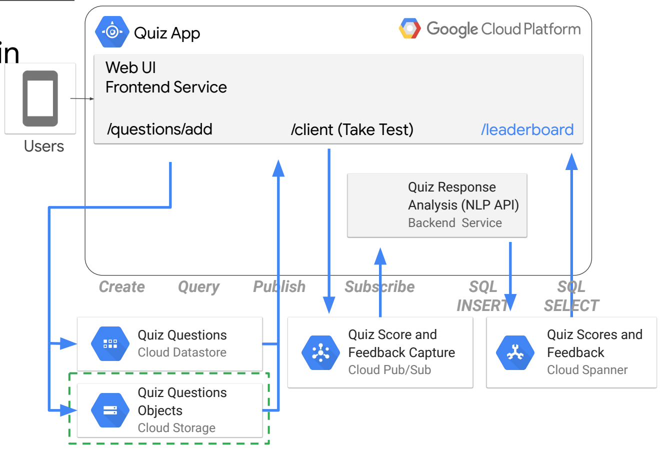
Hands-on Lab

Storing Image and Video Files in Cloud Storage

Duration: 45 minutes

Lab objectives:

- Create a Cloud Storage bucket
- Review file upload UI and code changes
- Write code to store upload file data in Cloud Storage



Google Cloud

Google Cloud