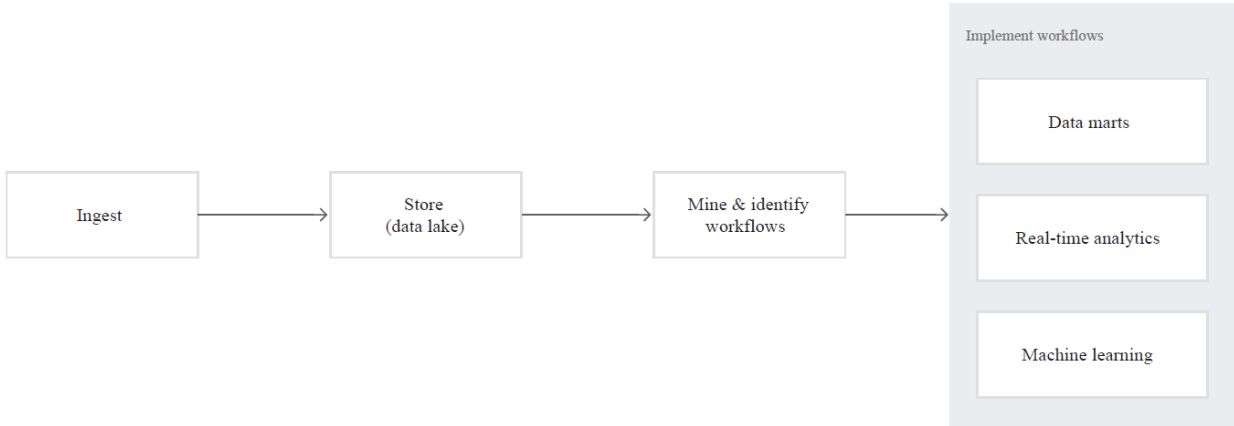
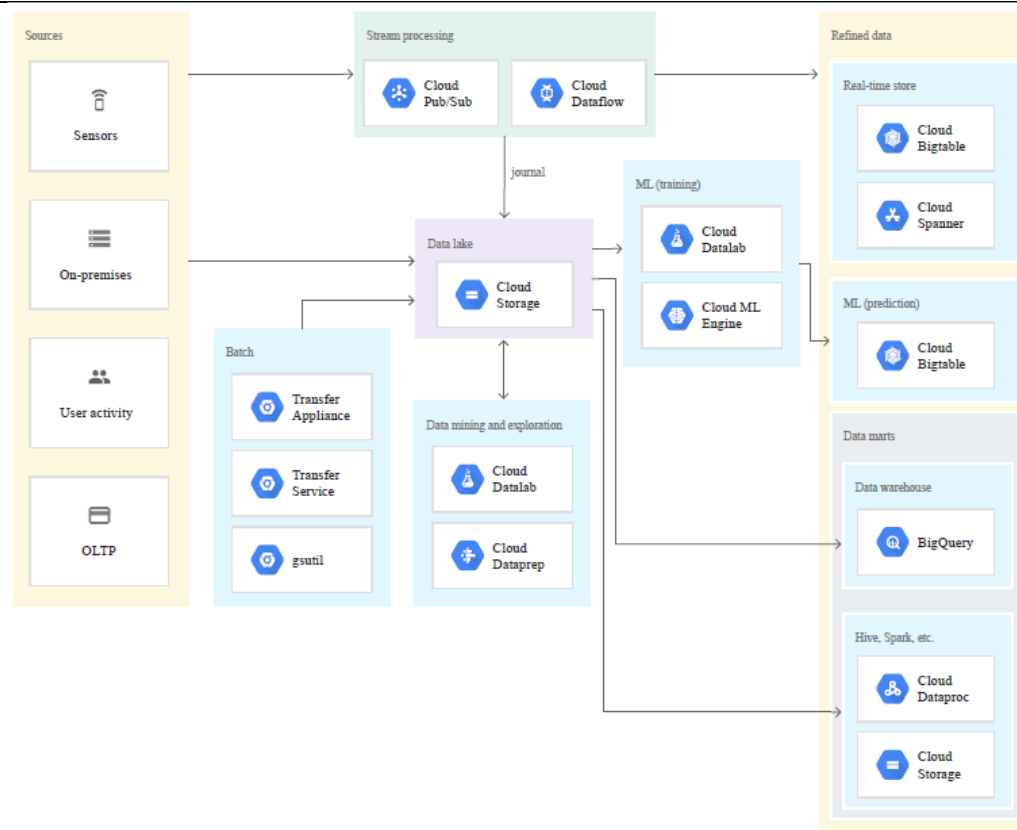


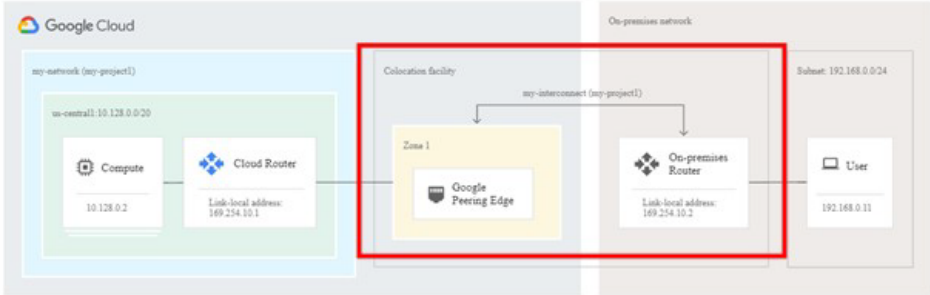
Infringement Chart for US 10,154,092 vs. Google

Claim Language	Google Evidence
1. An apparatus comprising:	Cloud Storage is an apparatus With these considerations in mind, here's how you can build a data lake on Google Cloud. The following diagram depicts the key stages in a data lake solution.  <pre> graph LR Ingest[Ingest] --> Store[Store (data lake)] Store --> Mine[Mine & identify workflows] Mine --> Implement[Implement workflows] subgraph Implement [Implement workflows] DM[Data marts] RTA[Real-time analytics] ML[Machine learning] end </pre> Cloud Storage as a data lake Cloud Architecture Center Google Cloud Source: https://cloud.google.com/architecture/build-a-data-lake-on-gcp
a first interface configured to receive input/output (I/O)	Cloud Storage has a first interface configured to receive input/output (I/O) traffic from a first host device via a dedicated I/O channel (traffic from a host device (on-premises sources), the I/O traffic comprising a read command

traffic from a first host device via a dedicated I/O channel, the I/O traffic comprising a read command;



Cloud Storage as a data lake | Cloud Architecture Center | Google Cloud
Source: <https://cloud.google.com/architecture/build-a-data-lake-on-gcp>

	How does Dedicated Interconnect work? For Dedicated Interconnect, you provision a cross connect between the Google network and your own router in a common location. The following example shows a single Dedicated Interconnect connection between a GCP VPC network and on-premises network:  The diagram illustrates a Dedicated Interconnect setup. On the left, the Google Cloud environment includes a VPC network named 'my-network (my-project1)' with a subnet 'us-central1 10.128.0.0/20'. Inside this VPC, there is a 'Compute' instance with IP '10.128.0.2' and a 'Cloud Router' with a 'Link-local address' of '169.254.10.1'. A red box highlights the 'Colocation facility' area, which contains a 'Google Peering Edge' router in 'Zone 1' and an 'On-premises Router' with a 'Link-local address' of '169.254.10.2'. These two routers are connected via a 'my-interconnect (my-project1)' link. To the right of the colocation facility is the 'On-premises network' with a 'Subnet: 192.168.0.0/24' containing a 'User' with IP '192.168.0.11'. Dedicated Interconnect overview Source: https://cloud.google.com/storage-transfer/docs/on-prem-overview
a second interface configured to receive first data via a network;	Google Storage Transfer Service receive first data (data from online sources) via a network

Complete large-scale data transfers fast

Transfer petabytes of data from on-premises sources or other clouds over online networks— billions of files and 10s of Gbps. Optimize your network bandwidth and accelerate transfers with scale-out performance.

KEY FEATURES

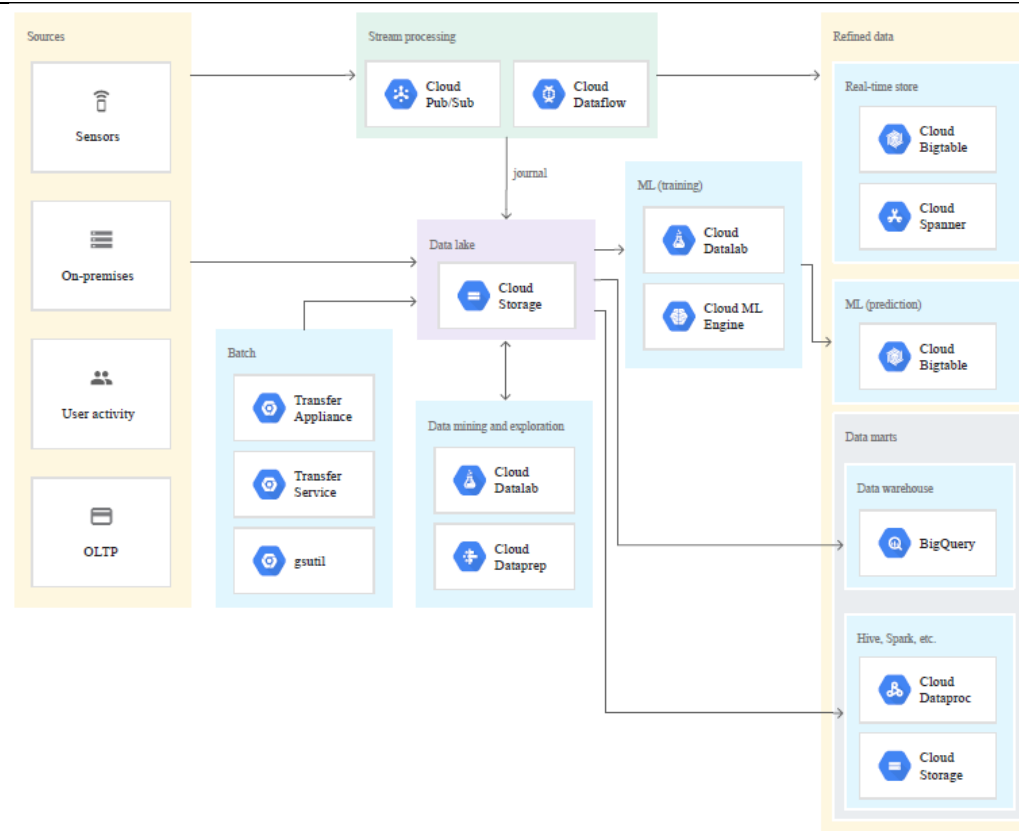
Reliable and secure data transfer services

Storage Transfer Service for cloud data

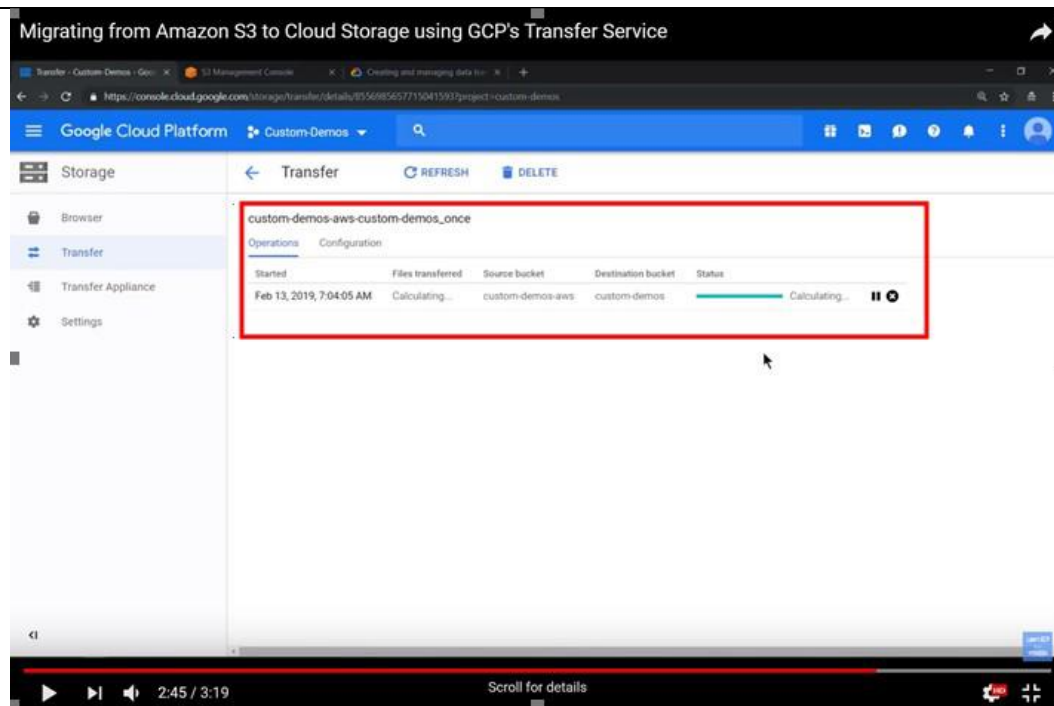
Transfer data from online sources like AWS S3 and Azure Blob Storage to Cloud Storage (/storage) in one simple process. Perform transfers as needed to enable a multicloud strategy, or periodically move data as part of a data-processing pipeline or analytical workflow.

Storage Transfer Service | Google Cloud

Source: <https://cloud.google.com/storage-transfer-service#storage-transfer-service>



Cloud Storage as a data lake | Cloud Architecture Center | Google Cloud
 Source: <https://cloud.google.com/architecture/build-a-data-lake-on-gcp>

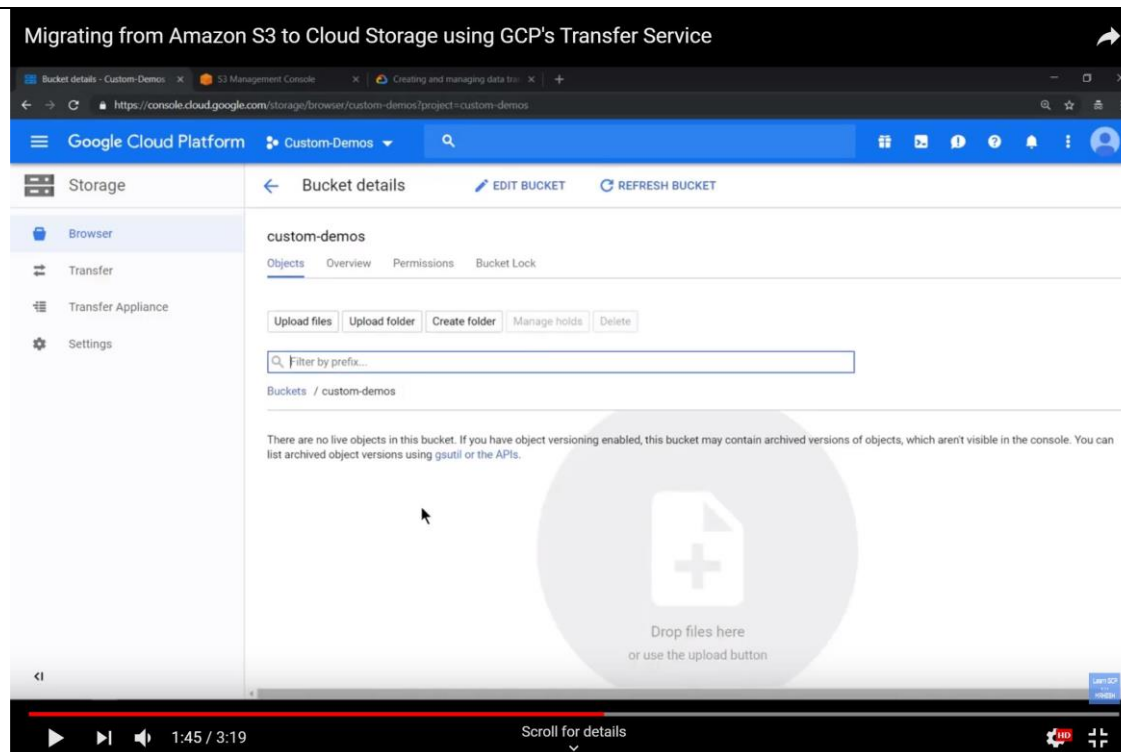


Source: <https://youtu.be/sBNtvg5D2CM?t=154>

a cache memory configured to store second data;

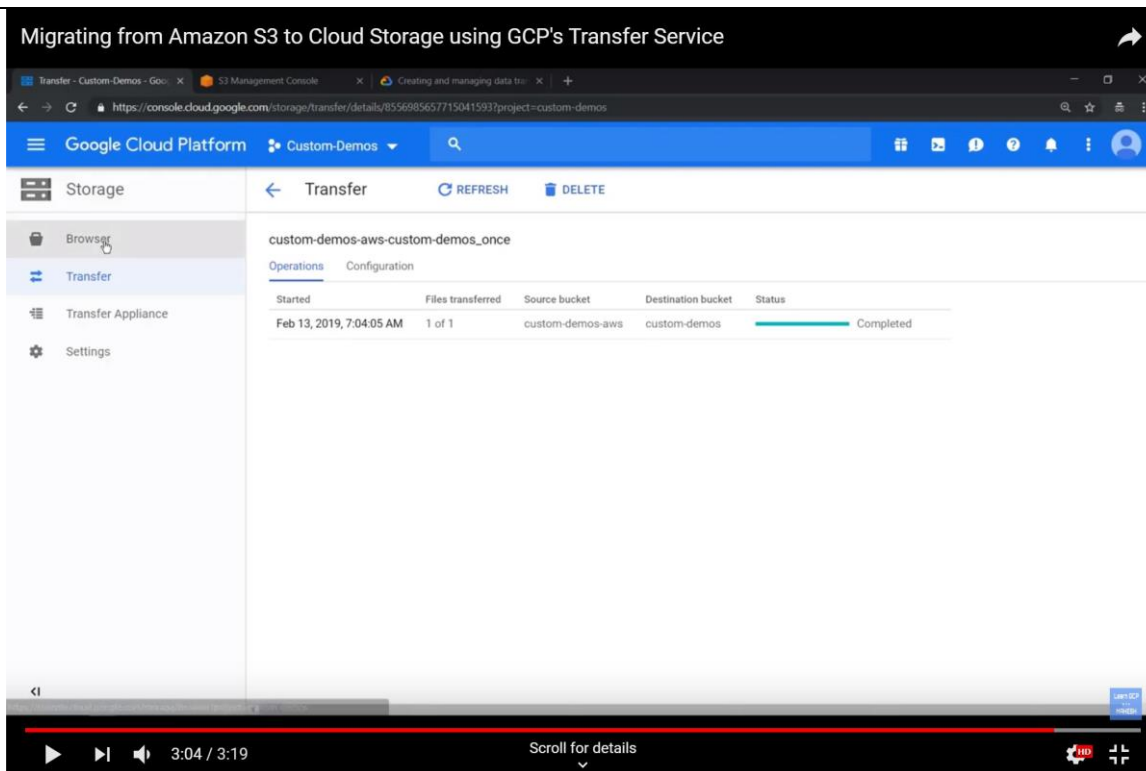
Cloud Storage comprises a cache memory configured to store second data

	<pre> def list_gcs_objects(google_access_key_id, google_access_key_secret, bucket_name): """Lists GCS objects using boto3 SDK""" # Create a new client and do the following: # 1. Change the endpoint URL to use the # Google Cloud Storage XML API endpoint. # 2. Use Cloud Storage HMAC Credentials. client = boto3.client("s3", region_name="auto", endpoint_url="https://storage.googleapis.com", aws_access_key_id=google_access_key_id, aws_secret_access_key=google_access_key_secret,) # Call GCS to list objects in bucket_name response = client.list_objects(Bucket=bucket_name) # Print object names print("Objects:") for blob in response["Contents"]: print(blob["Key"]) </pre> Source: https://cloud.google.com/storage/docs/migrating
a storage device configured to store third data; and	that a storage bucket is empty.



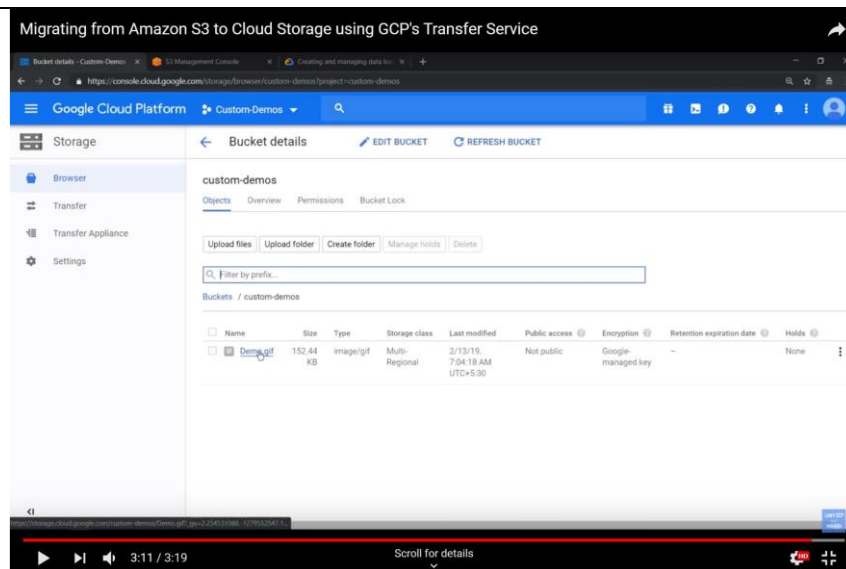
Source: <https://youtu.be/sBNtvG5D2CM?t=105>

the transfer process is complete.



Source: <https://youtu.be/sBNtvG5D2CM?t=105>

the data is stored in the storage bucket after completion of transfer process, which appears to relate to “storing third data at a storage device” as claimed.



Source: <https://youtu.be/sBNtvG5D2CM?t=191>

a processor coupled to the cache memory, the processor coupled to the storage device via a communication path that is distinct from the dedicated I/O channel, the processor configured to access the cache memory during

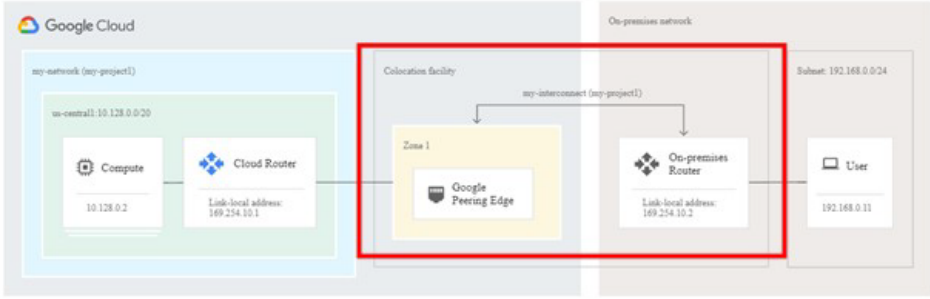
Cloud Storage comprises a processor(Console is operated bt a processor) coupled to the cache memory

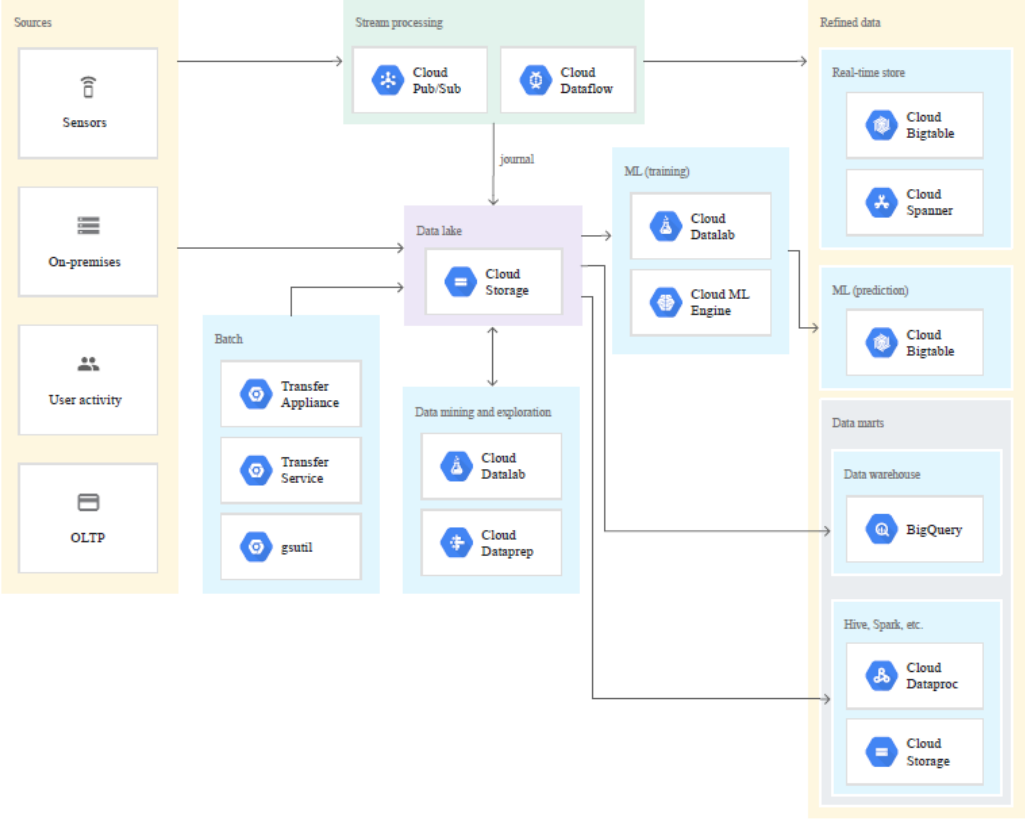
Basic tools for Cloud Storage

Here are some basic ways you can interact with Cloud Storage:

- **Console:** The [Google Cloud console](#) provides a visual interface for you to manage your data in a browser.
- **gsutil:** [gsutil](#) is a command-line tool that allows you to interact with Cloud Storage through a terminal. If you use other Google Cloud services, you can [download the Google Cloud CLI](#), which includes gsutil along with the gcloud tool for other services.
- **Client libraries:** The Cloud Storage [client libraries](#) allow you to manage your data using one of your preferred languages, including C++, C#, Go, Java, Node.js, PHP, Python, and Ruby.
- **REST APIs:** Manage your data using the [JSON](#) or [XML](#) API.

processing of the I/O traffic, the processor further configured to perform an access operation at the storage device based on the I/O traffic.	Source :https://cloud.google.com/storage/docs/introduction The request to transfer data is received via a dedicated channel and actual data from S3 is received at the google cloud storage via internet. Further the data received from S3 is loaded into cache memory and then written into cloud storage bucket. Therefore, it is evident that the path used for writing data into storage bucket is different (the Processor uses fast memory (i.e. cache memory) for processing the various functions which are used to complete the process of transfer of data from S3 to GCS) <pre>def list_gcs_objects(google_access_key_id, google_access_key_secret, bucket_name): """Lists GCS objects using boto3 SDK""" # Create a new client and do the following: # 1. Change the endpoint URL to use the # Google Cloud Storage XML API endpoint. # 2. Use Cloud Storage HMAC Credentials. client = boto3.client("s3", region_name="auto", endpoint_url="https://storage.googleapis.com", aws_access_key_id=google_access_key_id, aws_secret_access_key=google_access_key_secret,) # Call GCS to list objects in bucket_name response = client.list_objects(Bucket=bucket_name) # Print object names print("Objects:") for blob in response["Contents"]: print(blob["Key"])</pre> Source: https://cloud.google.com/storage/docs/migrating
2. The apparatus of claim 1, wherein the dedicated I/O	The dedicated I/O channel is associated with dedicated throughput that corresponds to the I/O traffic, for example, the channel from application server to client is the dedicated I/O channel associated with dedicated throughput corresponds to the I/O traffic.

channel is associated with dedicated throughput that corresponds to the I/O traffic.	How does Dedicated Interconnect work? For Dedicated Interconnect, you provision a cross connect between the Google network and your own router in a common location. The following example shows a single Dedicated Interconnect connection between a GCP VPC network and on-premises network:  The diagram illustrates the Dedicated Interconnect architecture. On the left, the Google Cloud environment shows a VPC network (my-network) with a subnet (us-central1) containing a Compute instance and a Cloud Router. The Cloud Router is connected to a Google Peering Edge in a colocation facility. This facility also houses an On-premises Router. The On-premises Router is connected to an on-premises network containing a Subnet and a User. A red box highlights the colocation facility and the interconnect connection between the Google Peering Edge and the On-premises Router. Dedicated Interconnect overview Source: https://cloud.google.com/storage-transfer/docs/on-prem-overview
3. The apparatus of claim 1, wherein the I/O traffic is distinct from the first data.	Cloud Storage comprises a second interface configured to receive first data via a network. Since receive first data via a network, not between computers and storage just as the I/O traffic, the I/O traffic is distinct from the first data.

	 Cloud Storage as a data lake Cloud Architecture Center Google Cloud Source: https://cloud.google.com/architecture/build-a-data-lake-on-gcp
5. The apparatus of claim 1, wherein	The second interface comprises an Ethernet interface

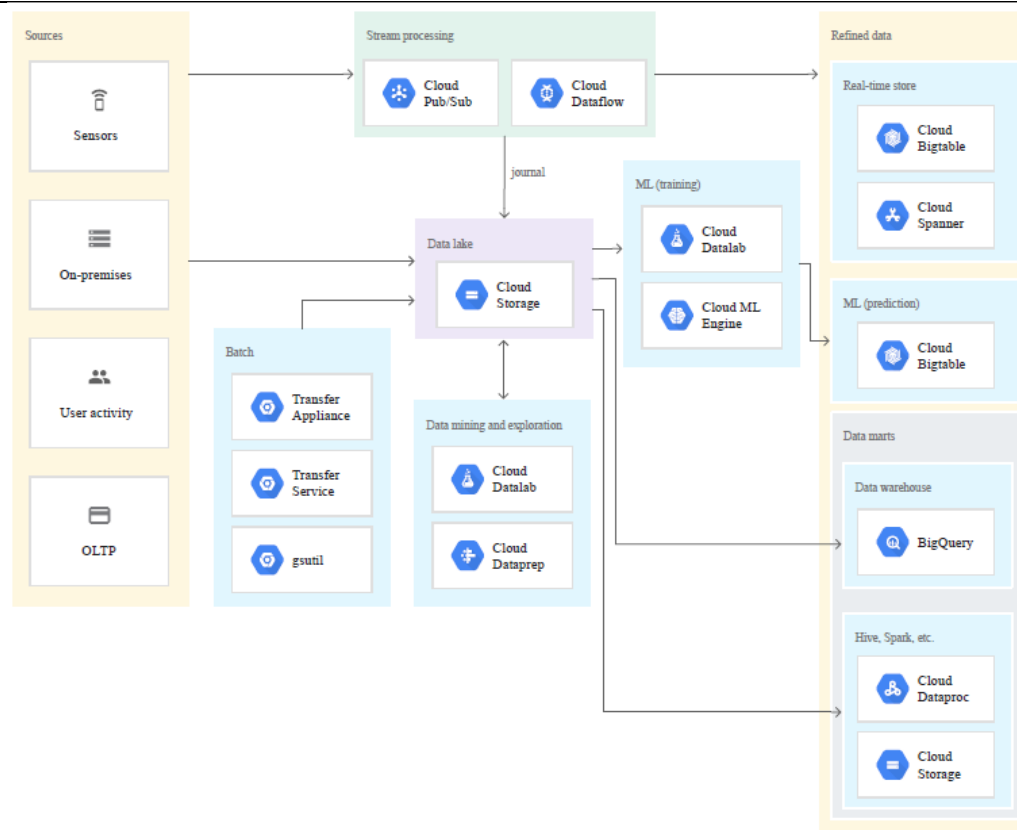
the second interface comprises an Ethernet interface or an asynchronous transfer mode (ATM) interface.	Complete large-scale data transfers fast Transfer petabytes of data from on-premises sources or other clouds over online networks— billions of files and 10s of Gbps. Optimize your network bandwidth and accelerate transfers with scale-out performance. KEY FEATURES Reliable and secure data transfer services Storage Transfer Service for cloud data Transfer data from online sources like AWS S3 and Azure Blob Storage to Cloud Storage (/storage) in one simple process. Perform transfers as needed to enable a multicloud strategy, or periodically move data as part of a data-processing pipeline or analytical workflow. Storage Transfer Service Google Cloud Source: https://cloud.google.com/storage-transfer-service#storage-transfer-service
6. The apparatus of claim 1, wherein the I/O traffic further comprises a write command.	The I/O traffic comprising a write command (access data include read and write)

	Writing to Cloud Storage To write a file to Cloud Storage: docs/appengine/storage/app.go View on GitHub <pre data-bbox="651 418 1791 1117"> // createFile creates a file in Google Cloud Storage. func (d *demo) createFile(fileName string) { fmt.Fprintf(d.w, "Creating file /%v/%v\n", d.bucketName, fileName) wc := d.bucket.Object(fileName).NewWriter(d.ctx) wc.ContentType = "text/plain" wc.Metadata = map[string]string{ "x-goog-meta-foo": "foo", "x-goog-meta-bar": "bar", } d.cleanUp = append(d.cleanUp, fileName) if _, err := wc.Write([]byte("abcde\n")); err != nil { d.errorf("createFile: unable to write data to bucket %q, file %q: %v", d.bucketName, fileName, err) return } if _, err := wc.Write([]byte(strings.Repeat("f", 1024*4) + "\n")); err != nil { d.errorf("createFile: unable to write data to bucket %q, file %q: %v", d.bucketName, fileName, err) return } if err := wc.Close(); err != nil { d.errorf("createFile: unable to close bucket %q, file %q: %v", d.bucketName, fileName, err) return } } </pre> Source: https://cloud.google.com/appengine/docs/standard/go111/googlecloudstorageclient/read-write-to-cloud-storage
7. The apparatus of claim 1, wherein the first data comprises a	the first data comprises a second read request, a write request, or a combination thereof

second read request, a write request, or a combination thereof.	Types of writes and when to use them ★ Note: For simplicity, write request types are identified here by their API names. The names are different in each client library. All write requests include the following basic components: • The name of the table to write to. • An app profile ID, which tells Bigtable how to route the traffic. • One or more mutations. A mutation consists of four elements: • Column family name • Column qualifier • Timestamp • Value that you are writing to the table Source: https://cloud.google.com/bigtable/docs/writes
8. The apparatus of claim 1, wherein the network comprises a plurality of interconnected computing devices.	The network comprises a plurality of interconnected computing devices

	Cloud Storage as a data lake Cloud Architecture Center Google Cloud Source: https://cloud.google.com/architecture/build-a-data-lake-on-gcp
9. The apparatus of claim 1, wherein the storage device and the dedicated	the storage device (from host to storage area) and the dedicated I/O channel (from Appliance server to client) are independently accessible

I/O channel are independently accessible, and wherein the processor is configured to read at least a portion of the second data from the cache memory based on the read command.



Cloud Storage as a data lake | Cloud Architecture Center | Google Cloud
Source: <https://cloud.google.com/architecture/build-a-data-lake-on-gcp>

	the processor is configured to read at least a portion of the second data from the cache memory based on the read command <pre> def list_gcs_objects(google_access_key_id, google_access_key_secret, bucket_name): """Lists GCS objects using boto3 SDK""" # Create a new client and do the following: # 1. Change the endpoint URL to use the # Google Cloud Storage XML API endpoint. # 2. Use Cloud Storage HMAC Credentials. client = boto3.client("s3", region_name="auto", endpoint_url="https://storage.googleapis.com", aws_access_key_id=google_access_key_id, aws_secret_access_key=google_access_key_secret,) # Call GCS to list objects in bucket_name response = client.list_objects(Bucket=bucket_name) # Print object names print("Objects:") for blob in response["Contents"]: print(blob["Key"]) </pre> Source: https://cloud.google.com/storage/docs/migrating
12. The apparatus of claim 1, wherein the host device comprises a server.	The host device comprises a server

	Easily transfer data to Cloud Storage Storage Transfer Service offers a highly performant, online pathway to Cloud Storage –both with the scalability and speed you need to simplify the data transfer process. For offline data transfer our Transfer Appliance is a shippable storage server that sits in your datacenter and then ships to an ingest location where the data is uploaded to Cloud Storage. Source: https://cloud.google.com/storage
13. A method comprising:	Google Storage Transfer Service is a service(method) for transferring data from cloud or on-premises sources What is Storage Transfer Service? Storage Transfer Service is a product that enables you to: • Move or backup data to a Cloud Storage bucket either from other cloud storage providers or from your <u>on-premises</u> (/storage-transfer/docs/key-terms#on-prem) storage. • Move data from one Cloud Storage bucket to another, so that it is available to different groups of users or applications. • Periodically move data as part of a data processing pipeline or analytical workflow. Overview Cloud Storage Transfer Service Documentation Google Cloud Source: https://cloud.google.com/storage-transfer/docs/overview

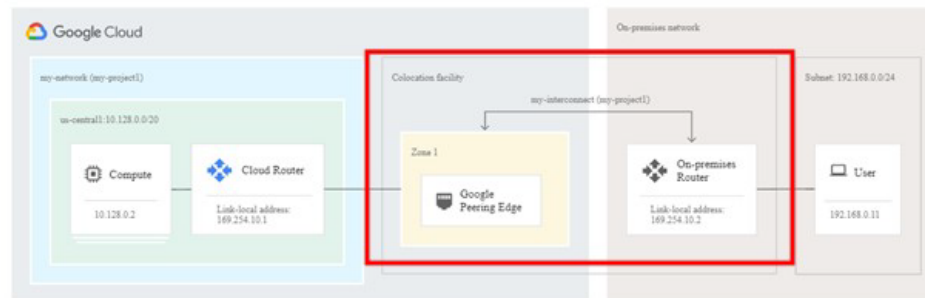
receiving input/output (I/O) traffic from a host device via a dedicated I/O channel at a first interface, the I/O traffic comprising a write command;	Google Storage Transfer Service input/output (I/O) traffic from a host device (on-premises sources) The diagram illustrates a comprehensive Google Cloud architecture centered around Cloud Storage as a data lake. On the left, 'Sources' include Sensors, On-premises systems, User activity, and OLTP. Data from these sources is processed through two main paths: 'Stream processing' (using Cloud Pub/Sub and Cloud Dataflow) and 'Batch' processing (using Transfer Appliance, Transfer Service, and gsutil). Both paths feed into the central 'Data lake' (Cloud Storage). From the Data lake, data is distributed to various other services: 'ML (training)' using Cloud Datalab and Cloud ML Engine; 'ML (prediction)' using Cloud Bigtable; 'Refined data' including a 'Real-time store' (Cloud Bigtable, Cloud Spanner) and 'Data marts' (Data warehouse with BigQuery, and other tools like Hive, Spark, Cloud Dataproc, and Cloud Storage). A 'journal' connection also links the Data lake to the ML (training) and ML (prediction) components. Cloud Storage as a data lake Cloud Architecture Center Google Cloud

Source: <https://cloud.google.com/architecture/build-a-data-lake-on-gcp>

traffic from a host device (on-premises sources) via a dedicated I/O channel (Dedicated Interconnect Work) at a first interface.

How does Dedicated Interconnect work?

For Dedicated Interconnect, you provision a cross connect between the Google network and your own router in a common location. The following example shows a single Dedicated Interconnect connection between a GCP VPC network and on-premises network:

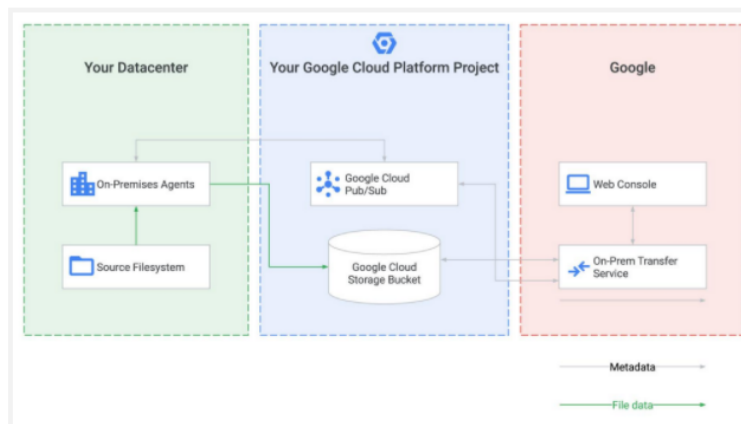


Dedicated Interconnect overview

Source: <https://cloud.google.com/storage-transfer/docs/on-prem-overview>

Getting started with Transfer Service for on-premises data

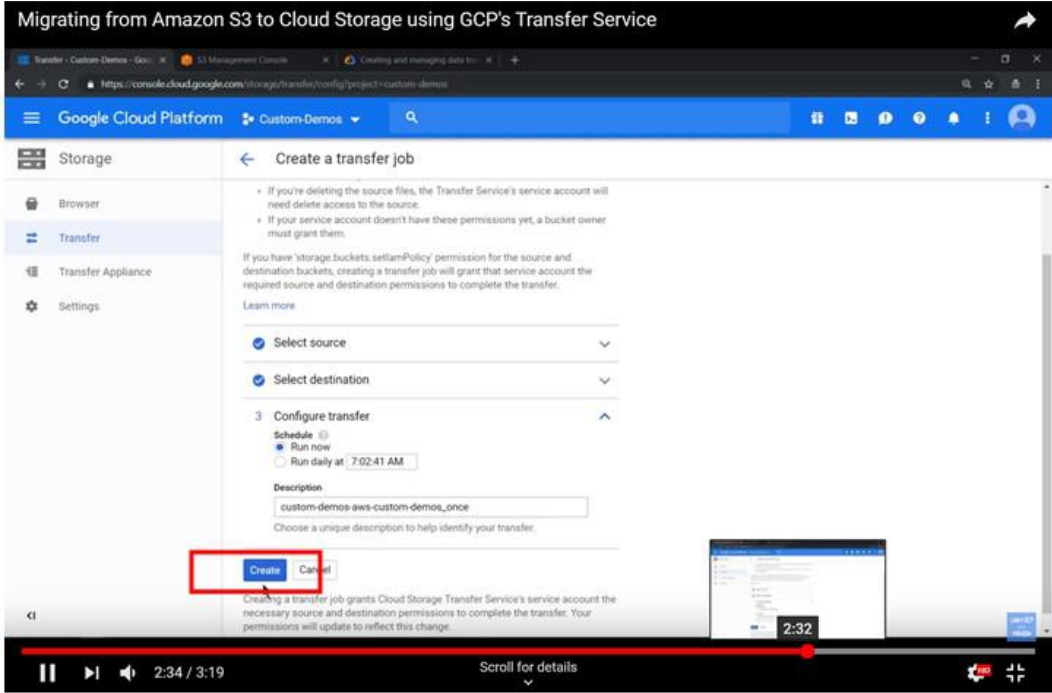
Here's how it works. First, install and start the on-premises software (the agent), then go to the Cloud Console and submit directories to transfer to Cloud Storage. When transferring data, the service will parallelize your transfer across many agents, and then coordinate these agents to transfer your data over a secure internet connection to Cloud Storage. Transfer Service for on-premises data also features a fully self-service GUI with detailed transfer logs so that you can create, monitor, and manage transfer jobs with confidence.



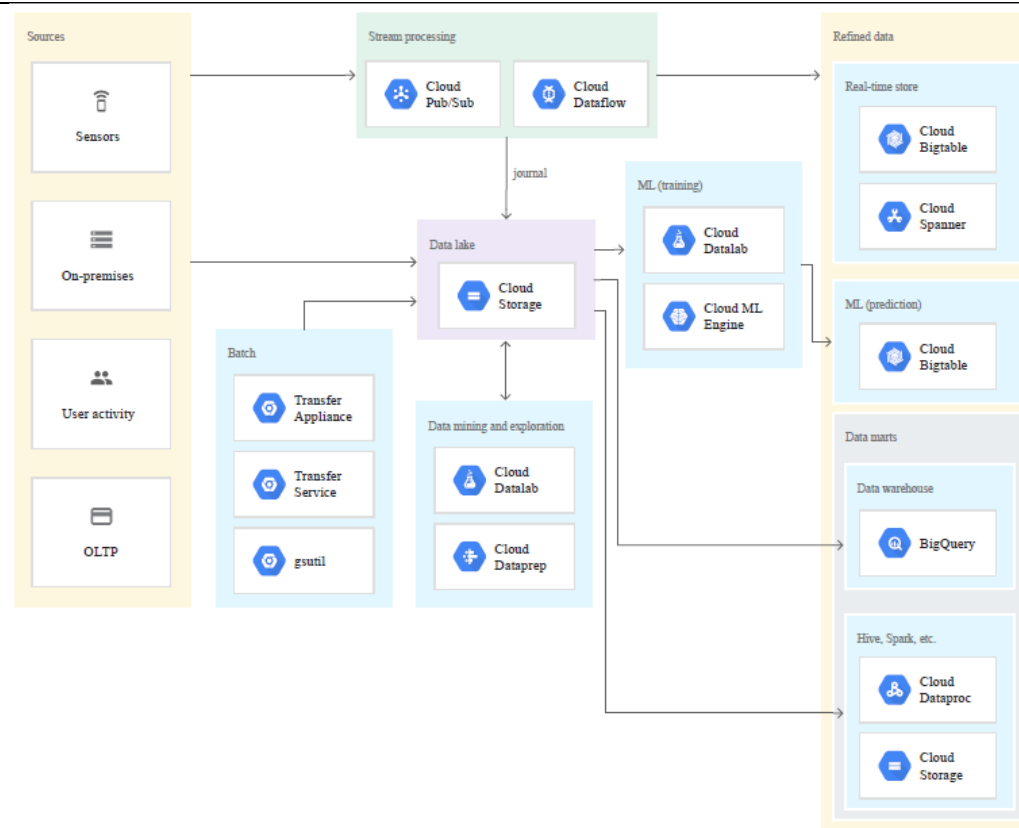
Introducing Storage Transfer Service for on-premises data

Source: <https://cloud.google.com/blog/products/storage-data-transfer/introducing-storage-transfer-service-for-on-premises-data>

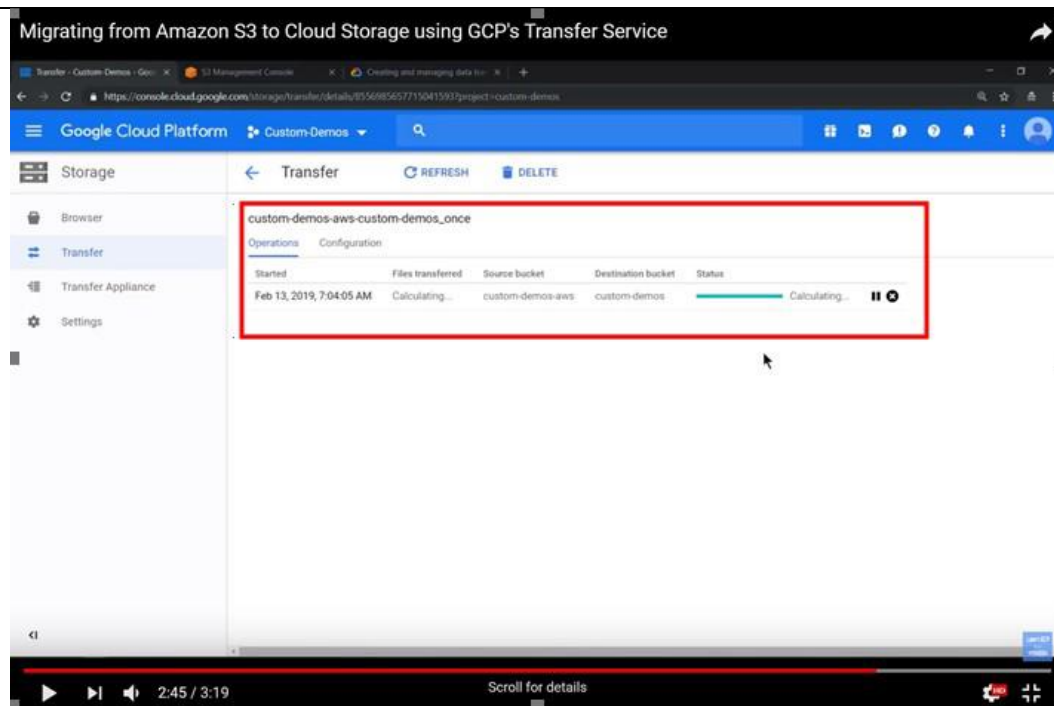
When user clicks on the create button highlighted with red rectangle in the figure 2. An I/O command is received by the processor of allocated cloud machine to transfer data from any other cloud to a bucket of google platform cloud, which appears to relate to “receiving input/output (I/O)” the purpose of the create command is to load files into google cloud storage bucket, which appears to relate to write command.

	 Source: https://youtu.be/sBNtvG5D2CM?t=154
receiving first data via a network at a second interface;	Google Storage Transfer Service receive first data (data from online sources) via a network

	Complete large-scale data transfers fast Transfer petabytes of data from on-premises sources or other clouds over online networks— billions of files and 10s of Gbps. Optimize your network bandwidth and accelerate transfers with scale-out performance. KEY FEATURES Reliable and secure data transfer services Storage Transfer Service for cloud data Transfer data from online sources like AWS S3 and Azure Blob Storage to Cloud Storage (/storage) in one simple process. Perform transfers as needed to enable a multicloud strategy, or periodically move data as part of a data-processing pipeline or analytical workflow. Storage Transfer Service Google Cloud Source: https://cloud.google.com/storage-transfer-service#storage-transfer-service
--	---



Cloud Storage as a data lake | Cloud Architecture Center | Google Cloud
 Source: <https://cloud.google.com/architecture/build-a-data-lake-on-gcp>



Source: <https://youtu.be/sBNtvg5D2CM?t=154>

storing second data at a cache memory;

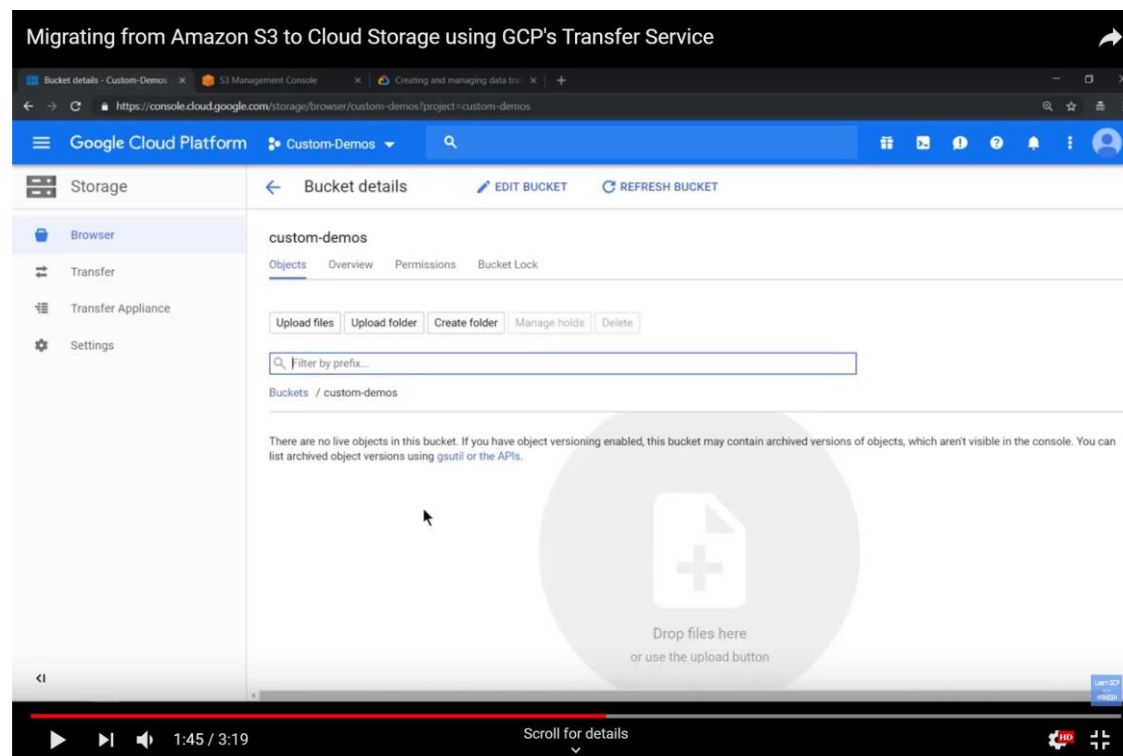
During the execution of this service all data and commands received on google cloud platform are processed only after loading into fast memory (like cache) as evident from code of a Function Shown below, Hence, getting the files from Amazon S3 and printing file name in Google Cloud Platform bucket appears to relate to “second data” as claimed.

```
def list_gcs_objects(google_access_key_id, google_access_key_secret, bucket_name):  
    """Lists GCS objects using boto3 SDK"""  
    # Create a new client and do the following:  
    # 1. Change the endpoint URL to use the  
    #    Google Cloud Storage XML API endpoint.  
    # 2. Use Cloud Storage HMAC Credentials.  
  
    client = boto3.client(  
        "s3",  
        region_name="auto",  
        endpoint_url="https://storage.googleapis.com",  
        aws_access_key_id=google_access_key_id,  
        aws_secret_access_key=google_access_key_secret,  
    )  
  
    # Call GCS to list objects in bucket_name  
    response = client.list_objects(Bucket=bucket_name)  
  
    # Print object names  
    print("Objects:")  
    for blob in response["Contents"]:  
        print(blob["Key"])
```

Source: <https://cloud.google.com/storage/docs/migrating>

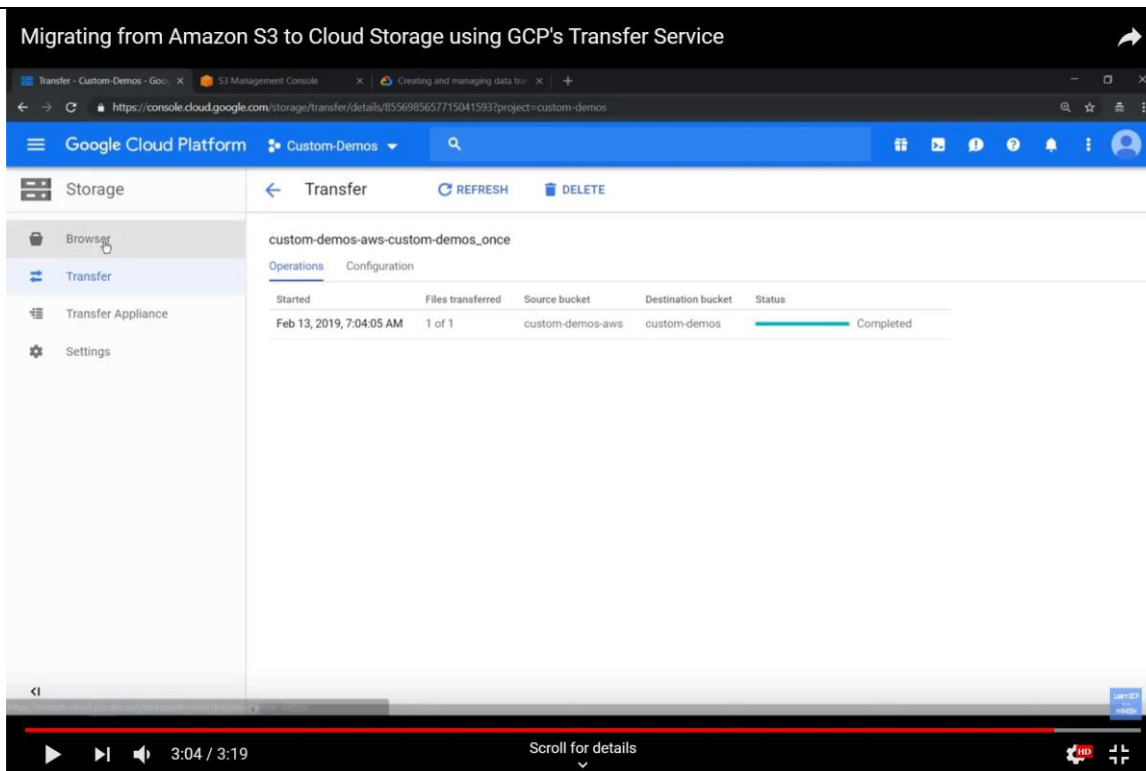
storing third data
at a storage
device;

that a storage bucket is empty.



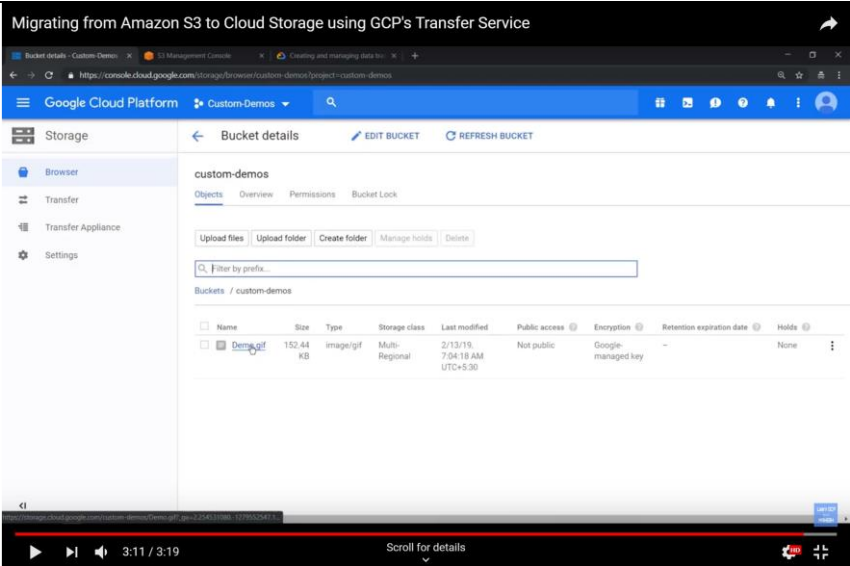
Source: <https://youtu.be/sBNtvG5D2CM?t=105>

the transfer process is complete.



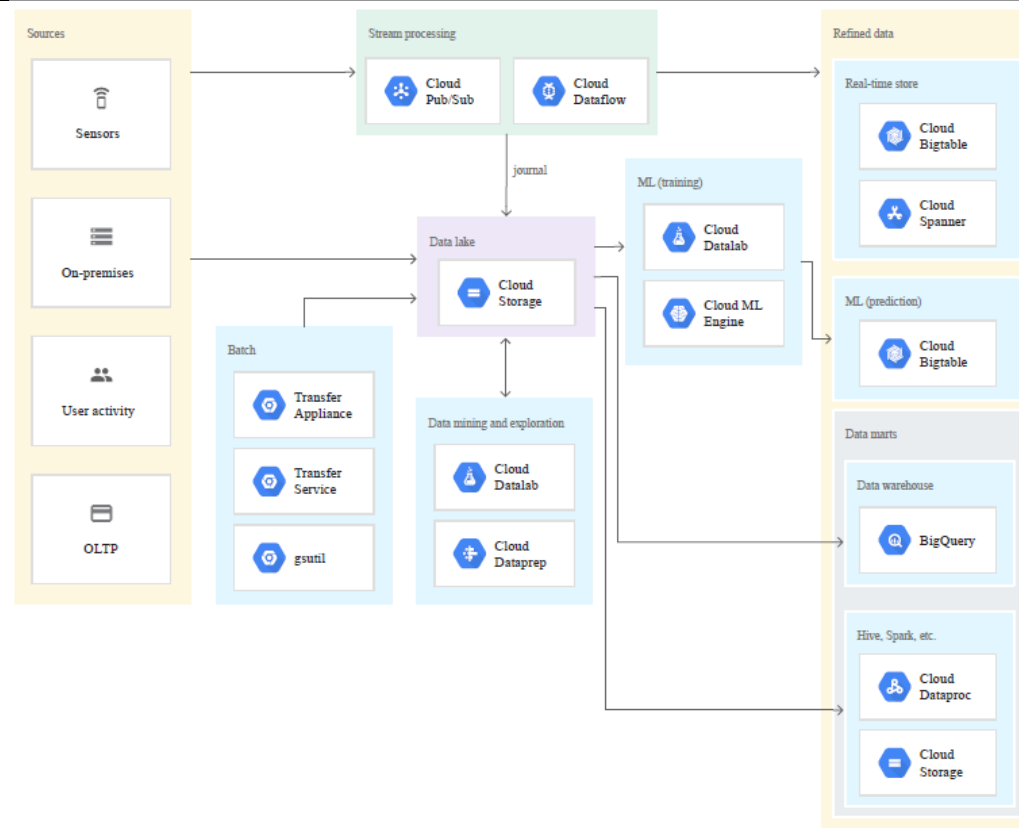
Source: <https://youtu.be/sBNtvG5D2CM?t=105>

the data is stored in the storage bucket after completion of transfer process, which appears to relate to “storing third data at a storage device” as claimed.

	 Source: https://youtu.be/sBNtvG5D2CM?t=191
accessing the cache memory during processing of the I/O traffic; and	During the execution of this service all data and commands received on google cloud platform are processed only after loading into fast memory (like cache) as evident from codes I/O function in the follow evidence

	<pre> def list_gcs_objects(google_access_key_id, google_access_key_secret, bucket_name): """Lists GCS objects using boto3 SDK""" # Create a new client and do the following: # 1. Change the endpoint URL to use the # Google Cloud Storage XML API endpoint. # 2. Use Cloud Storage HMAC Credentials. client = boto3.client("s3", region_name="auto", endpoint_url="https://storage.googleapis.com", aws_access_key_id=google_access_key_id, aws_secret_access_key=google_access_key_secret,) # Call GCS to list objects in bucket_name response = client.list_objects(Bucket=bucket_name) # Print object names print("Objects:") for blob in response["Contents"]: print(blob["Key"]) </pre> Source: https://cloud.google.com/storage/docs/migrating
performing one or more access operations at the storage device based on the I/O traffic, the one or	The request to transfer data from amazon's S3 is received via a dedicated channel and actual data from S3 is received at the google cloud storage via internet. Further the data received from S3 is loaded into cache memory and then written into cloud storage bucket. Therefore, it is evident that the path used for writing data into storage bucket is different (the Processor uses fast memory (i.e. cache memory) for processing the various functions which are used to complete the process of transfer of data from S3 to GCS)

more access operations utilizing a communication path between a processor and the storage device, the communication path distinct from the dedicated I/O channel.	<pre>def list_gcs_objects(google_access_key_id, google_access_key_secret, bucket_name): """Lists GCS objects using boto3 SDK""" # Create a new client and do the following: # 1. Change the endpoint URL to use the # Google Cloud Storage XML API endpoint. # 2. Use Cloud Storage HMAC Credentials. client = boto3.client("s3", region_name="auto", endpoint_url="https://storage.googleapis.com", aws_access_key_id=google_access_key_id, aws_secret_access_key=google_access_key_secret,) # Call GCS to list objects in bucket_name response = client.list_objects(Bucket=bucket_name) # Print object names print("Objects:") for blob in response["Contents"]: print(blob["Key"])</pre> Source: https://cloud.google.com/storage/docs/migrating
14. The method of claim 13, wherein the I/O traffic is distinct from the first data.	Cloud Storage comprises a second interface configured to receive first data via a network. Since receive first data via a network, not between computers and storage just as the I/O traffic, the I/O traffic is distinct from the first data.

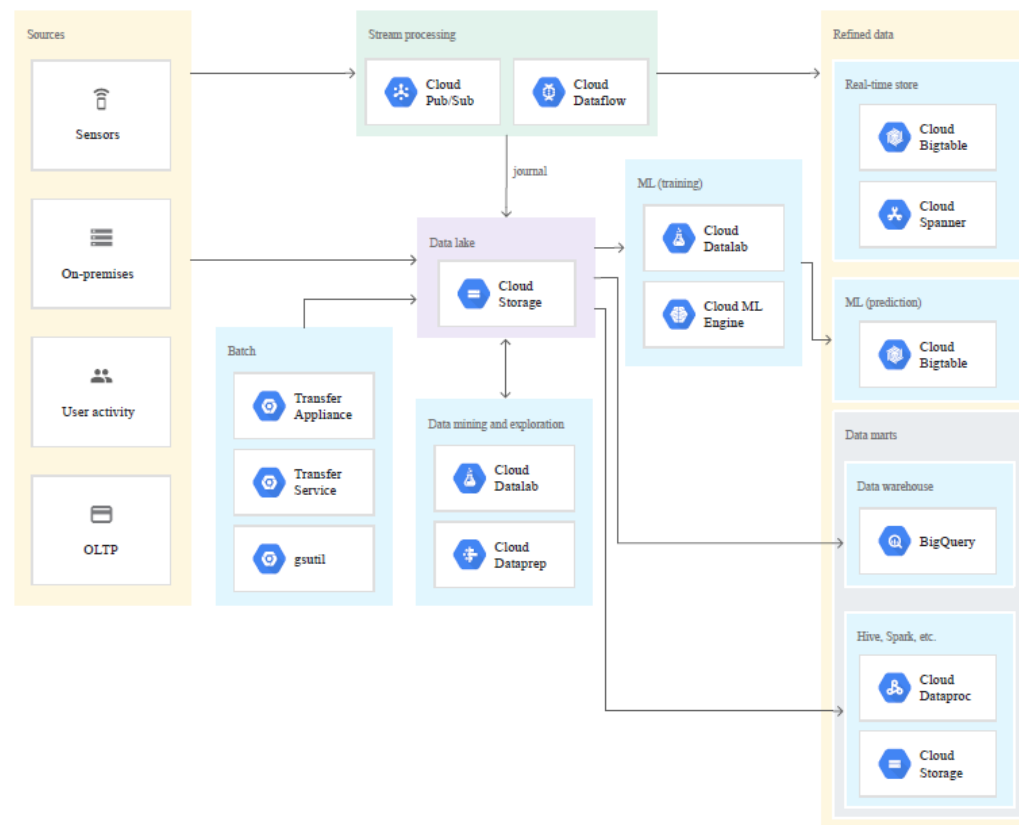


Cloud Storage as a data lake | Cloud Architecture Center | Google Cloud
Source: <https://cloud.google.com/architecture/build-a-data-lake-on-gcp>

15. The method of claim 13, further

The storage device (from host to storage area) and the dedicated I/O channel (from Appliance server to client) are independently accessible

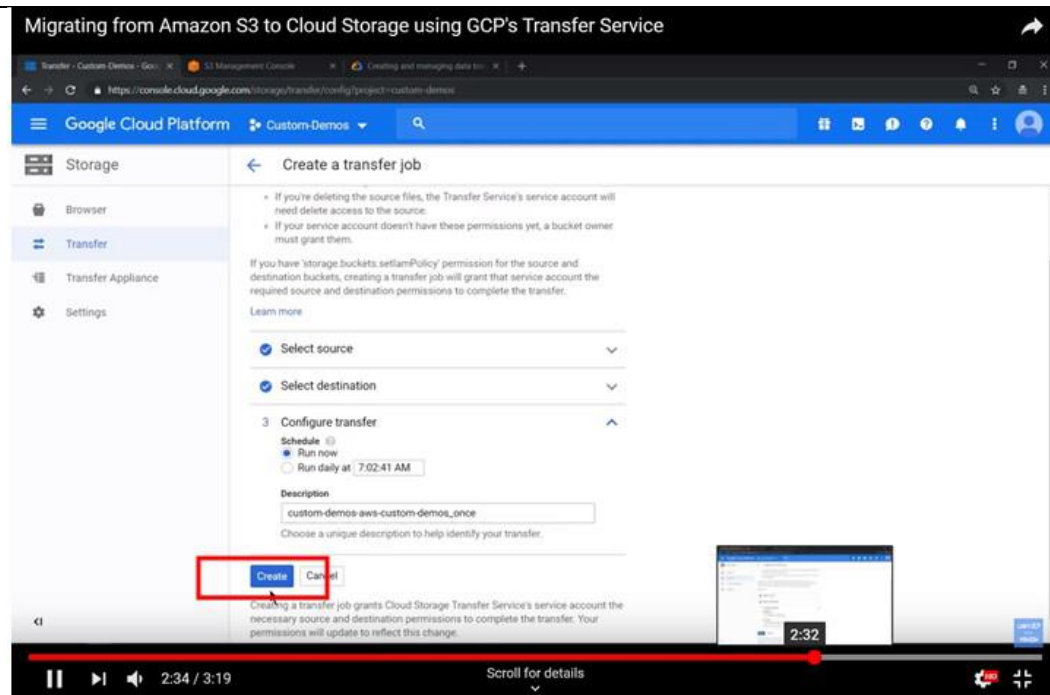
comprising performing a first access operation at the storage device independently of the dedicated I/O channel.



Cloud Storage as a data lake | Cloud Architecture Center | Google Cloud
Source: <https://cloud.google.com/architecture/build-a-data-lake-on-gcp>

	the processor is configured to read at least a portion of the second data from the cache memory based on the read command <pre> def list_gcs_objects(google_access_key_id, google_access_key_secret, bucket_name): """Lists GCS objects using boto3 SDK""" # Create a new client and do the following: # 1. Change the endpoint URL to use the # Google Cloud Storage XML API endpoint. # 2. Use Cloud Storage HMAC Credentials. client = boto3.client("s3", region_name="auto", endpoint_url="https://storage.googleapis.com", aws_access_key_id=google_access_key_id, aws_secret_access_key=google_access_key_secret,) # Call GCS to list objects in bucket_name response = client.list_objects(Bucket=bucket_name) # Print object names print("Objects:") for blob in response["Contents"]: print(blob["Key"]) </pre> Source: https://cloud.google.com/storage/docs/migrating
16. The method of claim 13, wherein the second data is written at the	the second data is written at the cache memory in response to the write command.

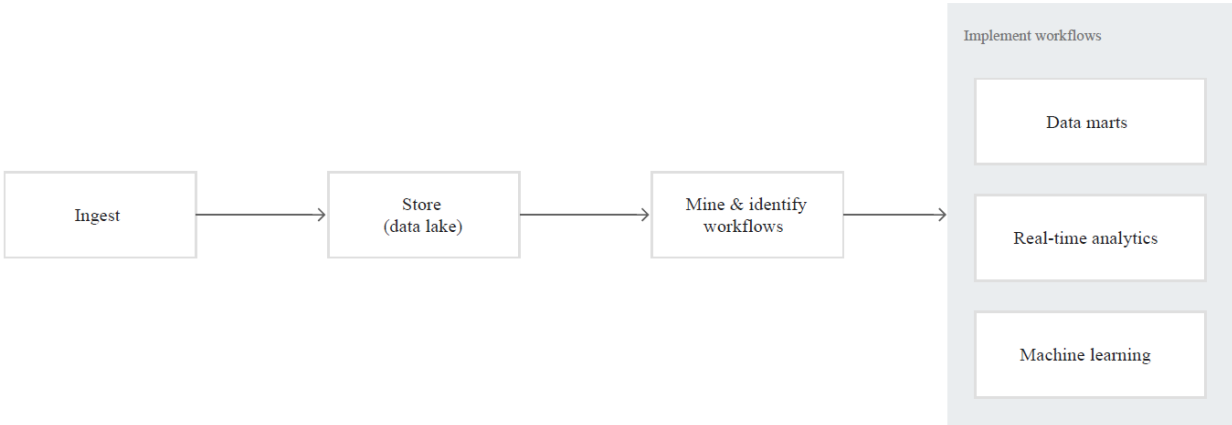
cache memory in response to the write command.



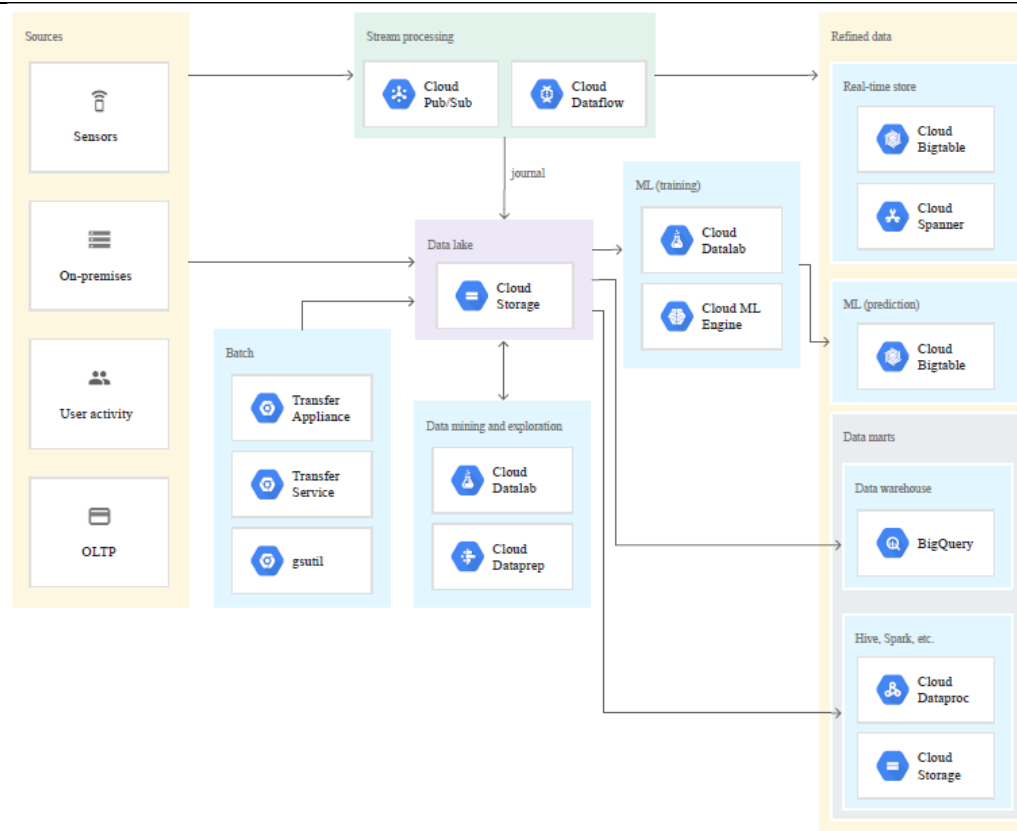
Source: <https://youtu.be/sBNtvG5D2CM?t=154>

19. An apparatus comprising:

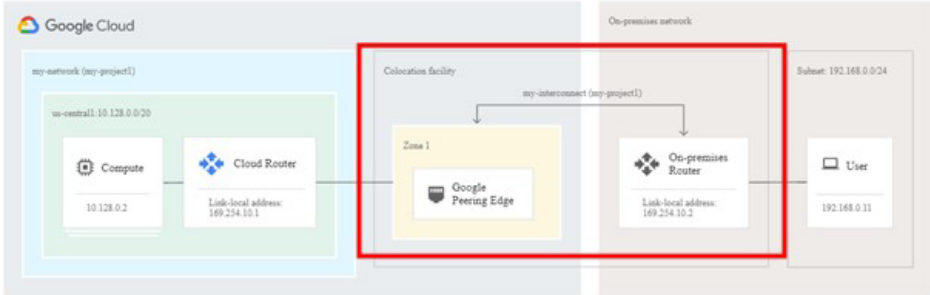
Cloud Storage is an apparatus

	With these considerations in mind, here's how you can build a data lake on Google Cloud. The following diagram depicts the key stages in a data lake solution.  <pre> graph LR Ingest[Ingest] --> Store[Store (data lake)] Store --> Mine[Mine & identify workflows] Mine --> Implement[Implement workflows] subgraph Implement [Implement workflows] Data[Data marts] Analytics[Real-time analytics] ML[Machine learning] end </pre> Cloud Storage as a data lake Cloud Architecture Center Google Cloud Source: https://cloud.google.com/architecture/build-a-data-lake-on-gcp
a first interface configured to receive input/output (I/O) traffic from a first host device via a dedicated I/O channel, the I/O traffic comprising one or more read	Cloud Storage has a first interface configured to receive input/output (I/O) traffic from a first host device via a dedicated I/O channel (traffic from a host device (on-premises sources), the I/O traffic comprising a read command

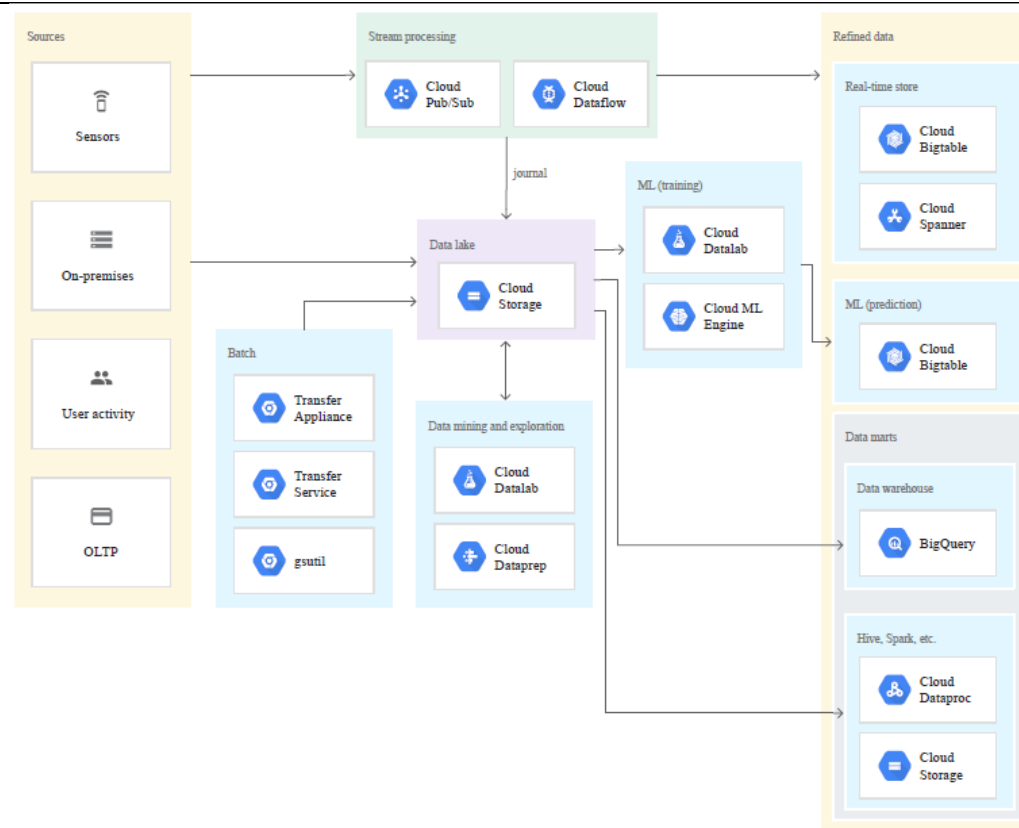
commands, one or more write commands, or a combination thereof;



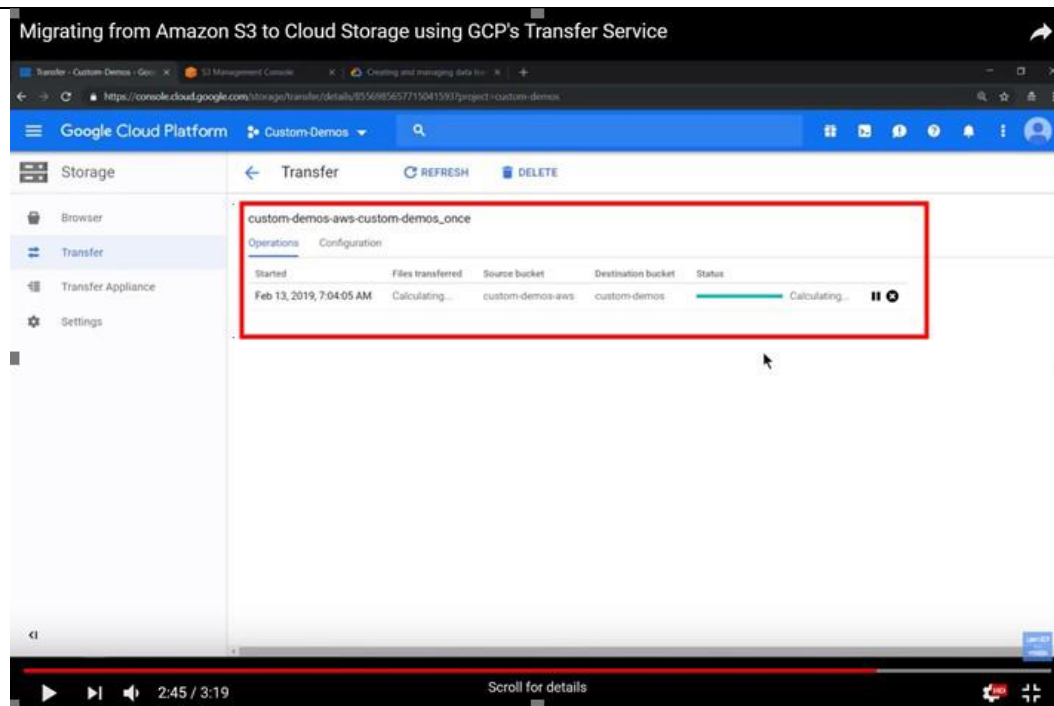
Cloud Storage as a data lake | Cloud Architecture Center | Google Cloud
Source: <https://cloud.google.com/architecture/build-a-data-lake-on-gcp>

	How does Dedicated Interconnect work? For Dedicated Interconnect, you provision a cross connect between the Google network and your own router in a common location. The following example shows a single Dedicated Interconnect connection between a GCP VPC network and on-premises network:  The diagram illustrates a Dedicated Interconnect setup. On the left, the Google Cloud network (my-network) contains a VPC (us-central1: 10.128.0.0/20) with a Compute instance (10.128.0.2) and a Cloud Router (Link-local address: 169.254.10.1). This Cloud Router is connected to a Google Peering Edge in a Colocation facility (Zone 1). The Peering Edge is connected to an On-premises Router (Link-local address: 169.254.10.2) in the On-premises network. The On-premises network also includes a Subnet (192.168.0.0/24) with a User (192.168.0.11). A red box highlights the Colocation facility area containing the Peering Edge and the On-premises Router. Dedicated Interconnect overview Source: https://cloud.google.com/storage-transfer/docs/on-prem-overview
a second interface configured to receive first data	Google Storage Transfer Service receive first data (data from online sources) via a network

	Complete large-scale data transfers fast Transfer petabytes of data from on-premises sources or other clouds over online networks— billions of files and 10s of Gbps. Optimize your network bandwidth and accelerate transfers with scale-out performance. KEY FEATURES Reliable and secure data transfer services Storage Transfer Service for cloud data Transfer data from online sources like AWS S3 and Azure Blob Storage to Cloud Storage (/storage) in one simple process. Perform transfers as needed to enable a multicloud strategy, or periodically move data as part of a data-processing pipeline or analytical workflow. Storage Transfer Service Google Cloud Source: https://cloud.google.com/storage-transfer-service#storage-transfer-service
--	---



Cloud Storage as a data lake | Cloud Architecture Center | Google Cloud
 Source: <https://cloud.google.com/architecture/build-a-data-lake-on-gcp>

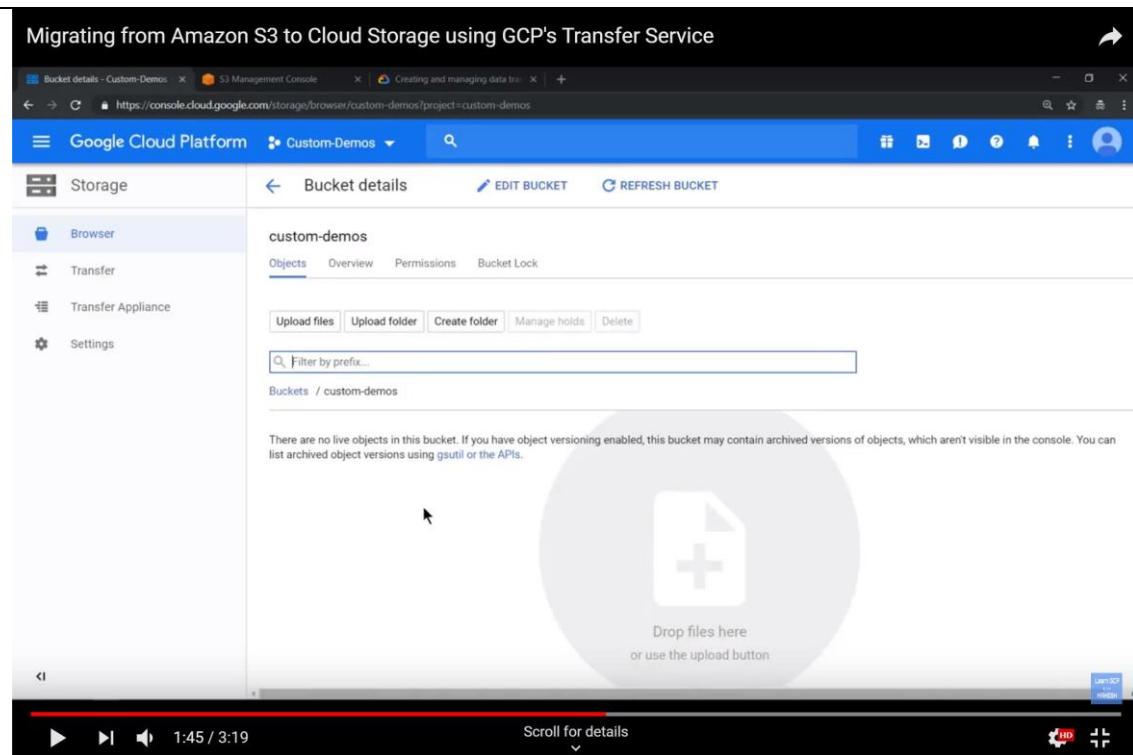


Source: <https://youtu.be/sBNtvg5D2CM?t=154>

a cache memory;

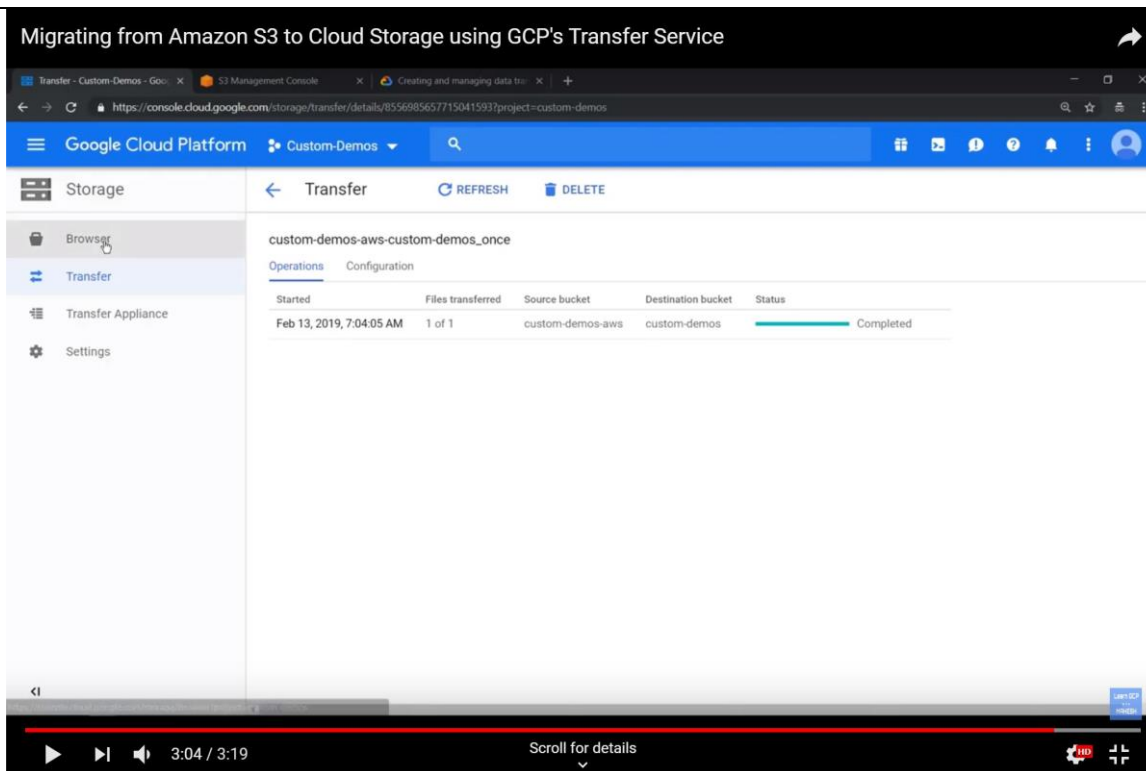
Cloud Storage comprises a cache memory

	<pre> def list_gcs_objects(google_access_key_id, google_access_key_secret, bucket_name): """Lists GCS objects using boto3 SDK""" # Create a new client and do the following: # 1. Change the endpoint URL to use the # Google Cloud Storage XML API endpoint. # 2. Use Cloud Storage HMAC Credentials. client = boto3.client("s3", region_name="auto", endpoint_url="https://storage.googleapis.com", aws_access_key_id=google_access_key_id, aws_secret_access_key=google_access_key_secret,) # Call GCS to list objects in bucket_name response = client.list_objects(Bucket=bucket_name) # Print object names print("Objects:") for blob in response["Contents"]: print(blob["Key"]) </pre> Source: https://cloud.google.com/storage/docs/migrating
a storage device; and	a storage bucket is empty.



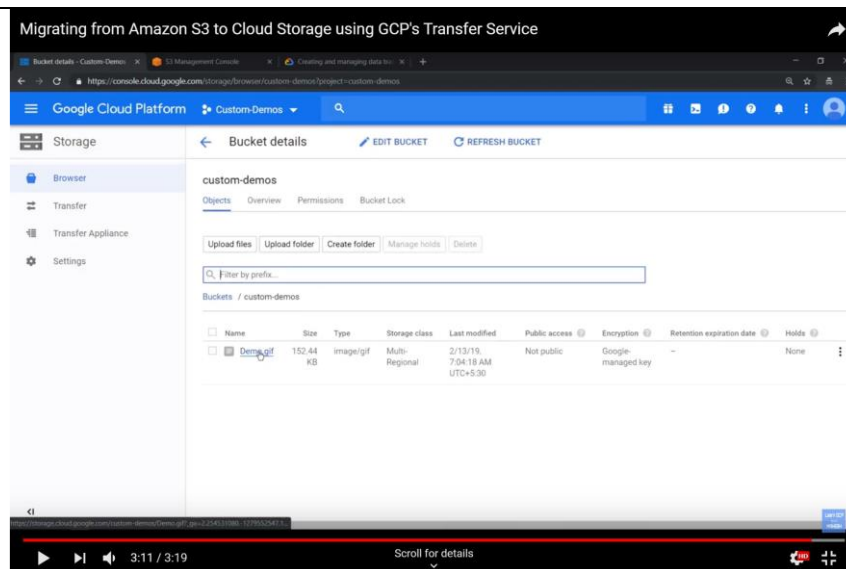
Source: <https://youtu.be/sBNtvG5D2CM?t=105>

the transfer process is complete.



Source: <https://youtu.be/sBNtvG5D2CM?t=105>

the data is stored in the storage bucket after completion of transfer process, which appears to relate to “at a storage device” as claimed.



Source: <https://youtu.be/sBNtvG5D2CM?t=191>

a processor coupled via a communication path to the storage device, the processor configured to access the cache memory during processing of the I/O traffic, the processor further configured to perform one or

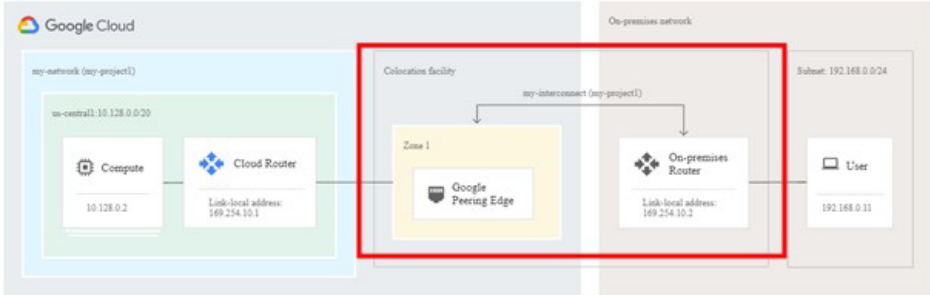
Cloud Storage comprises a processor(Console is operated bt a processor) coupled to the cache memory

Basic tools for Cloud Storage

Here are some basic ways you can interact with Cloud Storage:

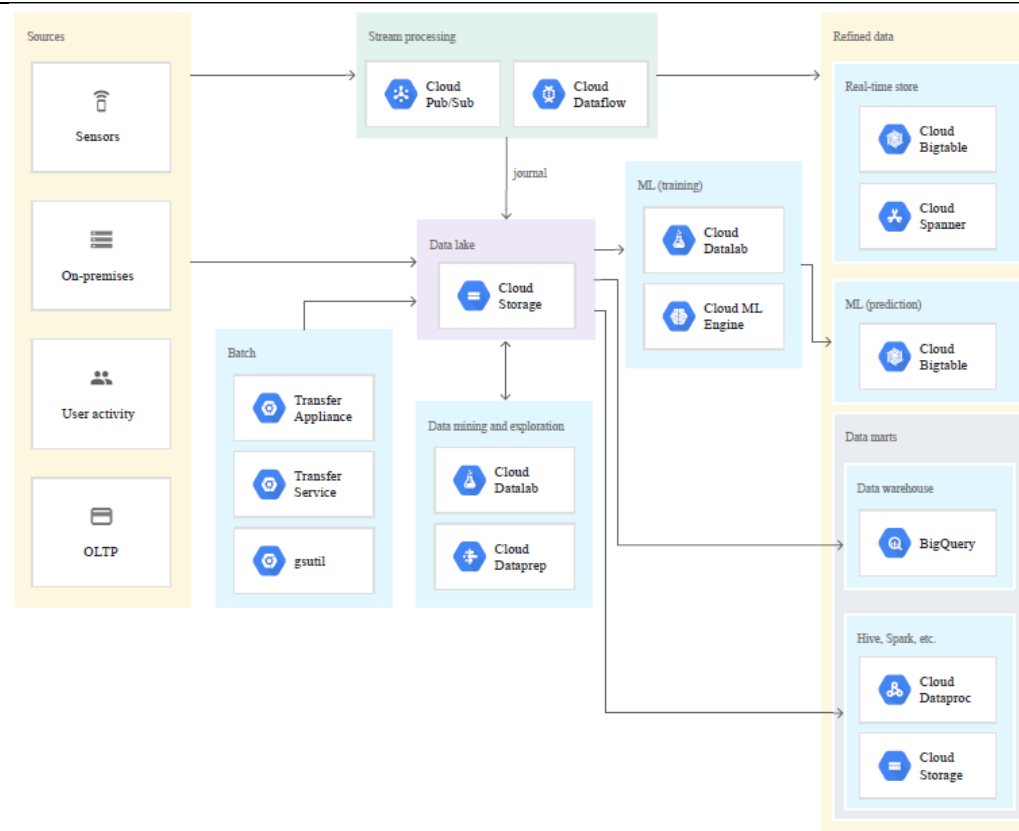
- **Console:** The [Google Cloud console](#) provides a visual interface for you to manage your data in a browser.
- **gsutil:** [gsutil](#) is a command-line tool that allows you to interact with Cloud Storage through a terminal. If you use other Google Cloud services, you can [download the Google Cloud CLI](#), which includes gsutil along with the gcloud tool for other services.
- **Client libraries:** The Cloud Storage [client libraries](#) allow you to manage your data using one of your preferred languages, including C++, C#, Go, Java, Node.js, PHP, Python, and Ruby.
- **REST APIs:** Manage your data using the [JSON](#) or [XML](#) API.

more access operations at the storage device based on the I/O traffic, wherein the communication path is distinct from the dedicated I/O channel.	Source :https://cloud.google.com/storage/docs/introduction The request to transfer data is received via a dedicated channel and actual data from S3 is received at the google cloud storage via internet. Further the data received from S3 is loaded into cache memory and then written into cloud storage bucket. Therefore, it is evident that the path used for writing data into storage bucket is different (the Processor uses fast memory (i.e. cache memory) for processing the various functions which are used to complete the process of transfer of data from S3 to GCS) <pre>def list_gcs_objects(google_access_key_id, google_access_key_secret, bucket_name): """Lists GCS objects using boto3 SDK""" # Create a new client and do the following: # 1. Change the endpoint URL to use the # Google Cloud Storage XML API endpoint. # 2. Use Cloud Storage HMAC Credentials. client = boto3.client("s3", region_name="auto", endpoint_url="https://storage.googleapis.com", aws_access_key_id=google_access_key_id, aws_secret_access_key=google_access_key_secret,) # Call GCS to list objects in bucket_name response = client.list_objects(Bucket=bucket_name) # Print object names print("Objects:") for blob in response["Contents"]: print(blob["Key"])</pre> Source: https://cloud.google.com/storage/docs/migrating
20. The apparatus of claim 19, wherein the	The dedicated I/O channel is associated with dedicated throughput that corresponds to the I/O traffic, for example, the channel from application server to client is the dedicated I/O channel associated with dedicated throughput corresponds to the I/O traffic.

dedicated I/O channel is associated with dedicated throughput that corresponds to the I/O traffic.	How does Dedicated Interconnect work? For Dedicated Interconnect, you provision a cross connect between the Google network and your own router in a common location. The following example shows a single Dedicated Interconnect connection between a GCP VPC network and on-premises network:  The diagram illustrates a Dedicated Interconnect setup. On the left, the Google Cloud environment shows a VPC network named 'my-network (my-project1)' with a subnet 'us-central1-10.128.0.0/20'. Inside this network, there is a 'Compute' instance with IP '10.128.0.2' and a 'Cloud Router' with a 'Link-local address' of '169.254.10.1'. A red box highlights the 'Colocation facility' area, which contains a 'Google Peering Edge' router in 'Zone 1' and an 'On-premises Router'. These two routers are connected via a 'my-interconnect (my-project1)' link. The 'On-premises Router' has a 'Link-local address' of '169.254.10.2'. To the right of the colocation facility is the 'On-premises network' with a 'Subnet: 192.168.0.0/24' containing a 'User' with IP '192.168.0.11'. Dedicated Interconnect overview Source: https://cloud.google.com/storage-transfer/docs/on-prem-overview
21. The apparatus of claim 19, wherein the I/O traffic is distinct from the data.	Cloud Storage comprises a second interface configured to receive first data via a network. Since receive data via a network, not between computers and storage just as the I/O traffic, the I/O traffic is distinct from the data.

	Cloud Storage as a data lake Cloud Architecture Center Google Cloud Source: https://cloud.google.com/architecture/build-a-data-lake-on-gcp
22. The apparatus of claim 19, wherein the data	The data corresponds to one or more read request, a write request, or a combination(access) thereof

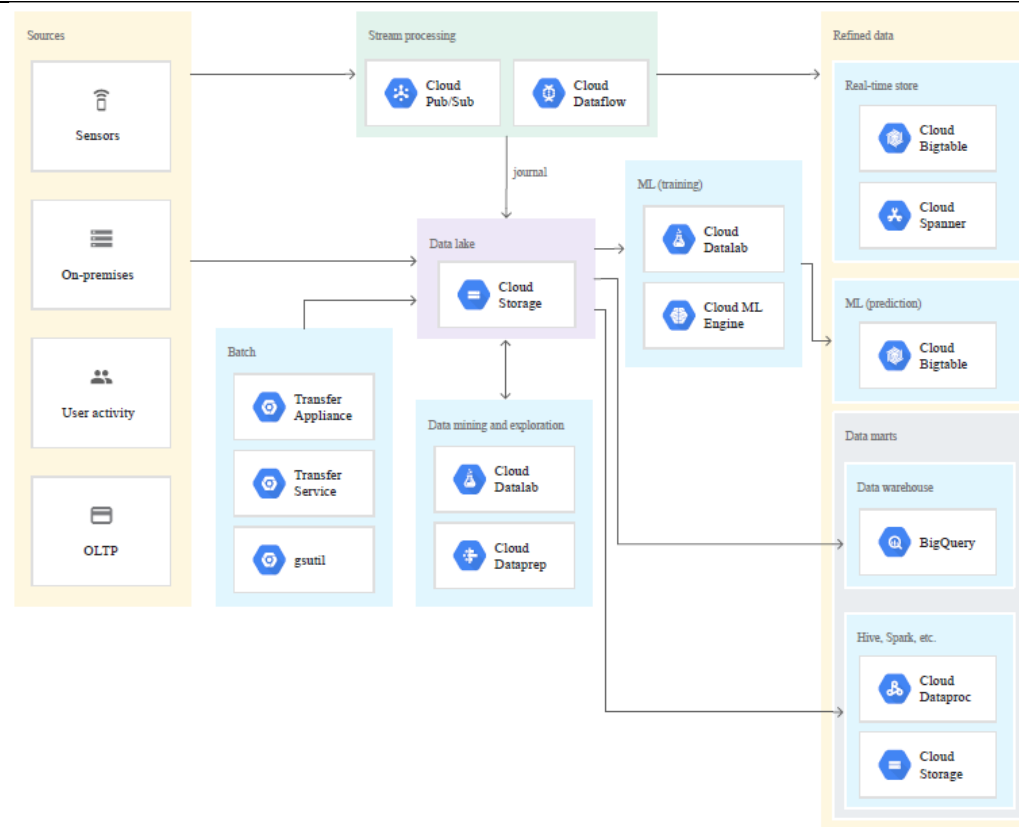
corresponds to one or more read requests, one or more write requests, or a combination thereof.	Types of writes and when to use them ★ Note: For simplicity, write request types are identified here by their API names. The names are different in each client library. All write requests include the following basic components: • The name of the table to write to. • An app profile ID, which tells Bigtable how to route the traffic. • One or more mutations. A mutation consists of four elements: • Column family name • Column qualifier • Timestamp • Value that you are writing to the table Source: https://cloud.google.com/bigtable/docs/writes
23. The apparatus of claim 19, wherein the storage device and the dedicated I/O channel are independently accessible.	The storage device (from host to storage area) and the dedicated I/O channel (from Appliance server to client) are independently accessible



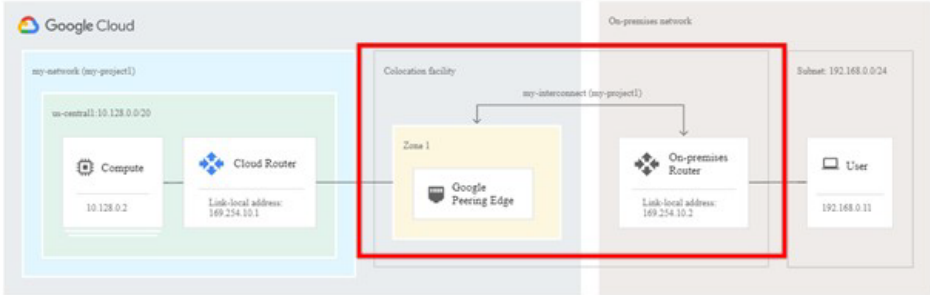
Cloud Storage as a data lake | Cloud Architecture Center | Google Cloud
 Source: <https://cloud.google.com/architecture/build-a-data-lake-on-gcp>

24. An apparatus comprising:	Cloud Storage is an apparatus With these considerations in mind, here's how you can build a data lake on Google Cloud. The following diagram depicts the key stages in a data lake solution.  <pre> graph LR Ingest[Ingest] --> Store[Store (data lake)] Store --> Mine[Mine & identify workflows] Mine --> Implement[Implement workflows] subgraph Implement [Implement workflows] Data[Data marts] Analytics[Real-time analytics] ML[Machine learning] end </pre> Cloud Storage as a data lake Cloud Architecture Center Google Cloud Source: https://cloud.google.com/architecture/build-a-data-lake-on-gcp
means for receiving input/output (I/O) traffic from a first host device via a dedicated I/O channel, the I/O	Cloud Storage has a first interface configured to receive input/output (I/O) traffic from a first host device via a dedicated I/O channel (traffic from a host device (on-premises sources), the I/O traffic comprising a read command

traffic comprising a write command;

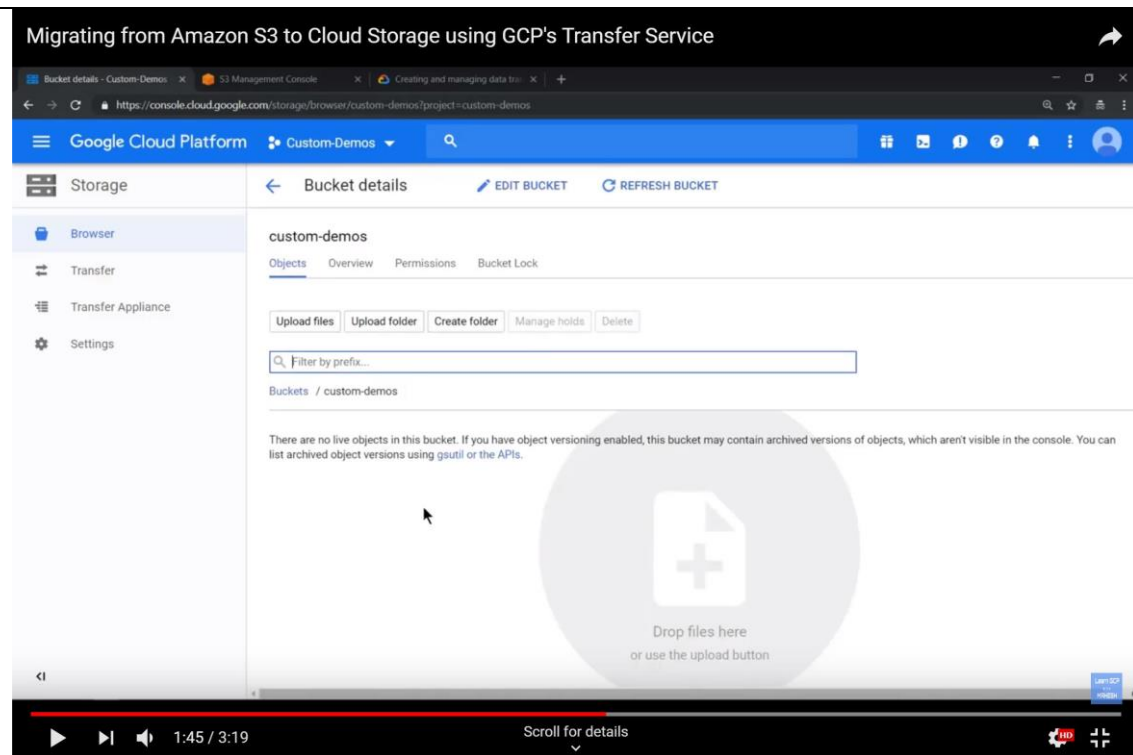


Cloud Storage as a data lake | Cloud Architecture Center | Google Cloud
Source: <https://cloud.google.com/architecture/build-a-data-lake-on-gcp>

	How does Dedicated Interconnect work? For Dedicated Interconnect, you provision a cross connect between the Google network and your own router in a common location. The following example shows a single Dedicated Interconnect connection between a GCP VPC network and on-premises network:  The diagram illustrates a Dedicated Interconnect setup. On the left, the Google Cloud environment shows a VPC network 'my-network (my-project1)' with a subnet 'us-central1 10.128.0.0/20'. Inside this VPC, there is a 'Compute' instance with IP '10.128.0.2' and a 'Cloud Router' with a 'Link-local address' of '169.254.10.1'. A red box highlights the 'Colocation facility' area, which contains a 'Google Peering Edge' router in 'Zone 1' with a 'Link-local address' of '169.254.10.1'. On the right, the 'On-premises network' contains an 'On-premises Router' with a 'Link-local address' of '169.254.10.2' and a 'User' with IP '192.168.0.11'. A 'Subnet: 192.168.0.0/24' is also shown. Arrows indicate the network connections between the Google Cloud components, the colocation facility, and the on-premises network. Dedicated Interconnect overview Source: https://cloud.google.com/storage-transfer/docs/on-prem-overview
means for receiving data via a network;	Google Storage Transfer Service receive data (data from online sources) via a network

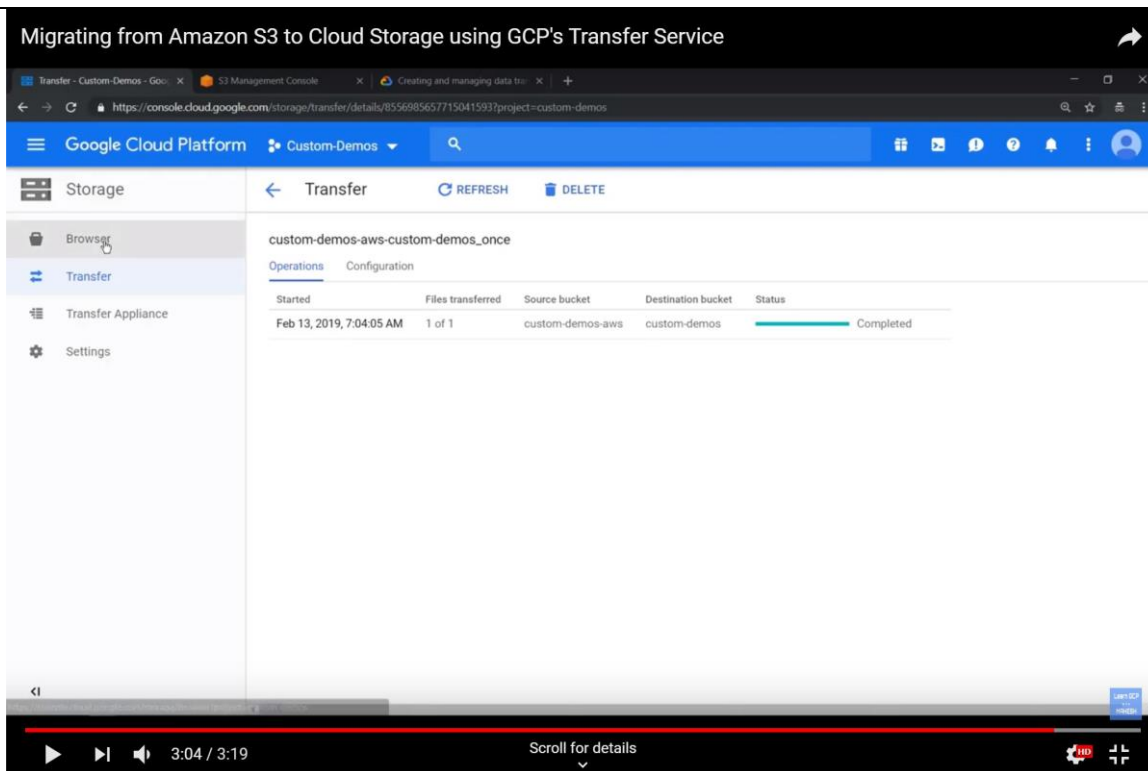
	Complete large-scale data transfers fast Transfer petabytes of data from on-premises sources or other clouds over online networks— billions of files and 10s of Gbps. Optimize your network bandwidth and accelerate transfers with scale-out performance. KEY FEATURES Reliable and secure data transfer services Storage Transfer Service for cloud data Transfer data from online sources like AWS S3 and Azure Blob Storage to <u>Cloud Storage</u> (/storage) in one simple process. Perform transfers as needed to enable a multicloud strategy, or periodically move data as part of a data-processing pipeline or analytical workflow. Storage Transfer Service Google Cloud Source: https://cloud.google.com/storage-transfer-service#storage-transfer-service
means for short-term data storage;	Cloud Storage comprises a cache memory (means for short-term data storage)

	<pre> def list_gcs_objects(google_access_key_id, google_access_key_secret, bucket_name): """Lists GCS objects using boto3 SDK""" # Create a new client and do the following: # 1. Change the endpoint URL to use the # Google Cloud Storage XML API endpoint. # 2. Use Cloud Storage HMAC Credentials. client = boto3.client("s3", region_name="auto", endpoint_url="https://storage.googleapis.com", aws_access_key_id=google_access_key_id, aws_secret_access_key=google_access_key_secret,) # Call GCS to list objects in bucket_name response = client.list_objects(Bucket=bucket_name) # Print object names print("Objects:") for blob in response["Contents"]: print(blob["Key"]) </pre> Source: https://cloud.google.com/storage/docs/migrating
means for long-term data storage; and	means for long-term data storage a storage bucket is empty.



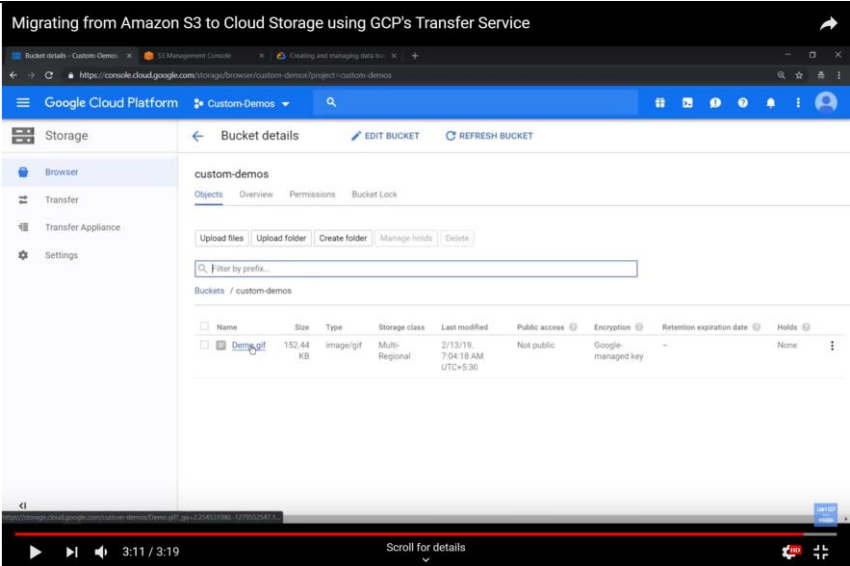
Source: <https://youtu.be/sBNtvG5D2CM?t=105>

the transfer process is complete.



Source: <https://youtu.be/sBNtvG5D2CM?t=105>

the data is stored in the storage bucket after completion of transfer process, which appears to relate to “means for long-term data storage ” as claimed.

	 Source: https://youtu.be/sBNtvG5D2CM?t=191
means for performing one or more access operations at the means for short-term data storage during processing of the I/O traffic and for performing one or more access operations at the means for long-term data	Means (console) for performing one or more access operations at the means for short-term data storage during processing of the I/O traffic and for performing one or more access operations at the means for long-term data storage based on the I/O traffic, the means for performing distinct from the dedicated I/O channel.

storage based on the I/O traffic, the means for performing distinct from the dedicated I/O channel.

Basic tools for Cloud Storage

Here are some basic ways you can interact with Cloud Storage:

- **Console:** The [Google Cloud console](#) provides a visual interface for you to manage your data in a browser.
- **gsutil:** [gsutil](#) is a command-line tool that allows you to interact with Cloud Storage through a terminal. If you use other Google Cloud services, you can [download the Google Cloud CLI](#), which includes gsutil along with the gcloud tool for other services.
- **Client libraries:** The Cloud Storage [client libraries](#) allow you to manage your data using one of your preferred languages, including C++, C#, Go, Java, Node.js, PHP, Python, and Ruby.
- **REST APIs:** Manage your data using the [JSON](#) or [XML](#) API.

Source :<https://cloud.google.com/storage/docs/introduction>

The request to transfer data is received via a dedicated channel and actual data from S3 is received at the google cloud storage via internet. Further the data received from S3 is loaded into cache memory and then written into cloud storage bucket. Therefore, it is evident that the path used for writing data into storage bucket is different (the Processor uses fast memory (i.e. cache memory) for processing the various functions which are used to complete the process of transfer of data from S3 to GCS)

```
def list_gcs_objects(google_access_key_id, google_access_key_secret, bucket_name):
    """Lists GCS objects using boto3 SDK"""
    # Create a new client and do the following:
    # 1. Change the endpoint URL to use the
    #    Google Cloud Storage XML API endpoint.
    # 2. Use Cloud Storage HMAC Credentials.

    client = boto3.client(
        "s3",
        region_name="auto",
        endpoint_url="https://storage.googleapis.com",
        aws_access_key_id=google_access_key_id,
        aws_secret_access_key=google_access_key_secret,
    )

    # Call GCS to list objects in bucket_name
    response = client.list_objects(Bucket=bucket_name)

    # Print object names
    print("Objects:")
    for blob in response["Contents"]:
        print(blob["Key"])
```

Source: <https://cloud.google.com/storage/docs/migrating>