**Qué es Cloud Storage?**  
Es un servicio de almacenamiento de objetos, un objeto es una pieza de datos inmutable identificada por un nombre de objeto. Un bucket es un contenedor de objetos.

* Principales características
  + Experiencia unificada
  + Listado fuertemente consistente
  + Geo - redundante
  + Escalable a exabytes

**Usos Cloud Storage**

* Almacenamiento y entrega de contenido: Imágenes o videos.
* Cómputo, analítica y ML: Grandes volúmenes de información y usar [Cloud Dataproc](https://cloud.google.com/dataproc) para analizar esa información.
* Respaldos y archivado: Respaldos de bases de datos, máquinas virtuales, sistemas de información, etc.

**Estructura típica**

Los objectos están asociados a *Buckets*. Los *Buckets* están relacionados con proyectos y los proyectos se asocian a una Organización.

**Seguridad**

* Podemos **gestionar el acceso** a los objetos dentro de Cloud Storage.
* El 100% de los objetos en Cloud Storage están **cifrados** todo el tiempo; tanto en reposo como en movimiento.
* Capacidad de retención; la cantidad de tiempo que garantizamos el almacenar de un objeto.
* Control de versiones, como git, podemos “reescribir” los objetos con la nueva versión de los mismos y conservar el historial de cada cambio.

**Cifrado**

Control flexible a la hora de cifrar los datos.

* **Default:** Google gestiona las claves de forma transparente.
* **Customer-managed (CMEK) :** Gestionas tus llaves en Google Cloud KMS.
* **Customer-supplied (CSEK):** Almacena tus llaves fuera de Google Cloud.

**Usar Cloud Storage**

Podemos desarrollar nuestras herramientas para interactuar con Cloud Storage.

* Desde nuestros programas.
* Construir nuestras librerías.
* Rehusar las librerías que Google nos ofrece.

Herramientas que ya existentes para usar Cloud Storage:

* Cloud Console
* Gsutil

### gsutil

Se sugiere ponerle al Bucket que este asociado con el proyecto, para que sea fácil de ubicar. El nombre del Bucket debe ser único.

Crear un Bucket (make Bucket)

gsutil mb gs://mi-proyecto-bucket-01

Ver mis Buckets

gsutil ls

Ver los objetos en un Bucket

gsutil ls gs://mi-proyecto-bucket-01

Copiar objetos a un Bucket

gsutil cp ./archivo.txt gs://mi-proyecto-bucket-01

Copiar objetos de un Bucket a otro lado

gsutil cp gs://mi-proyecto-bucket-01/archivo.txt .

Activar el versionamiento en los objetos de un Bucket

gsutil versioning set on gs://mi-proyecto-bucket-01

Desactivar el versionamiento en los objetos de un Bucket

gsutil versioning set off gs://mi-proyecto-bucket-01

**Breve Resumen: Clases de almacenamiento de Cloud Storage**

Al crear un Bucket debemos definir la ubicación y clase.

**Ubicación**

Define donde residirán nuestros datos.

**Tipos de ubicación**

* **Regional**: Los datos se **replican en las zonas** de una sola región. Para datos a los que se accede con frecuencia dentro una región.
* **Dual-regional**: Los datos se replican en **regiones cercanas**. Para disponibilidad de datos accedidos frecuentemente en un área específica.
* **Multi-regional**: Los datos se replican en **diferentes regiones** (continentes). Para la más alta disponibilidad de datos frecuentemente accedidos.

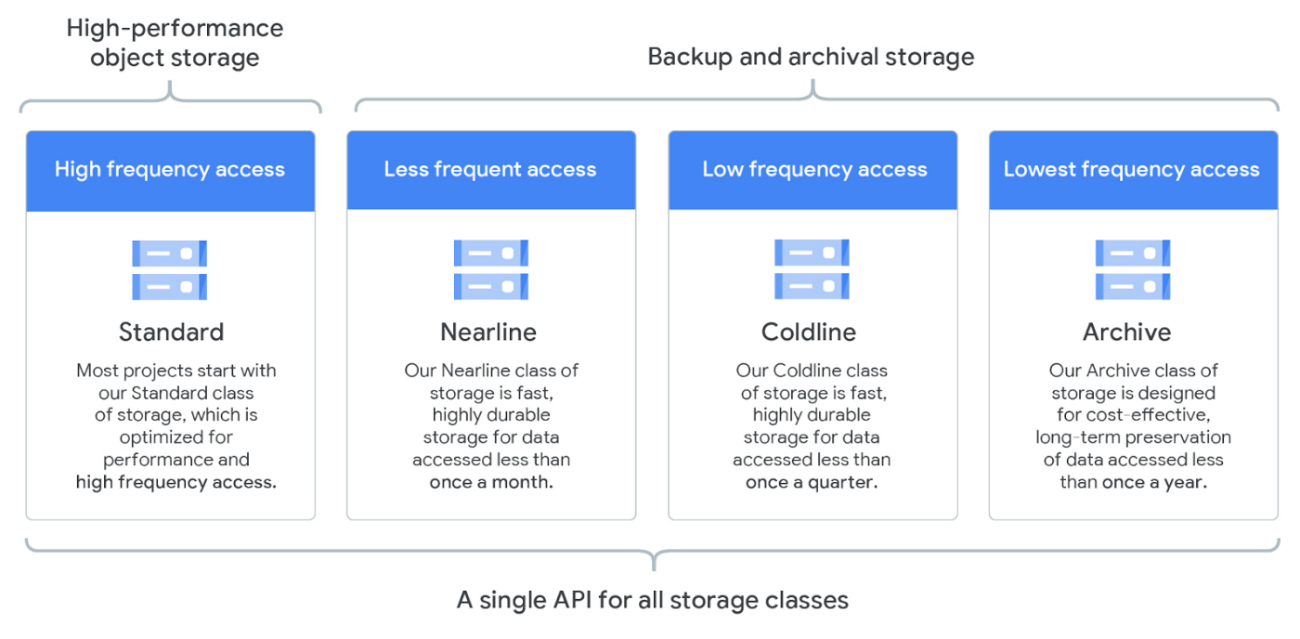
**Clases**

Se refieren a la frecuencia con la que accedemos y disponibilidad que deseamos sobre los objetos dentro del bucket. Además del precio.

* **Standars**: Objetos a los que accedemos frecuentemente (más caro).
* **Nearline**: Acceso a los objetos una vez al mes.
* **Coldline**: Acceso a los objetos una vez al trimestre.
* **Archive**: Acceso a los objetos una vez al año.

**Clases vs. ubicación**

Podemos combinar las clases con diferentes ubicaciones, de acuerdo a nuestras necesidades.



**CREACION DE BUCKET**

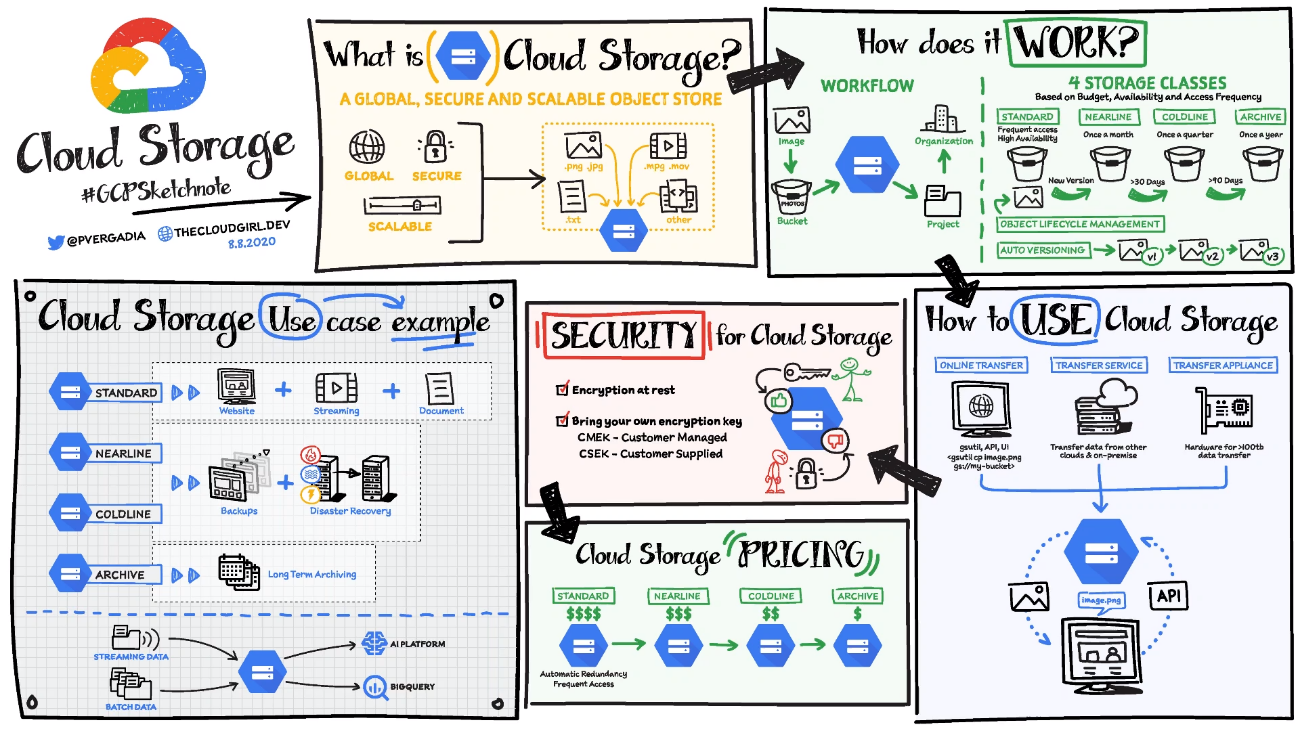
Se crea un Bucket Standar Multi-region en US. Se previo el acceso al público y se eligió el acceso uniforme. Finalmente seleccionamos una retención de 10 años. Esto en gsutil se vería así:

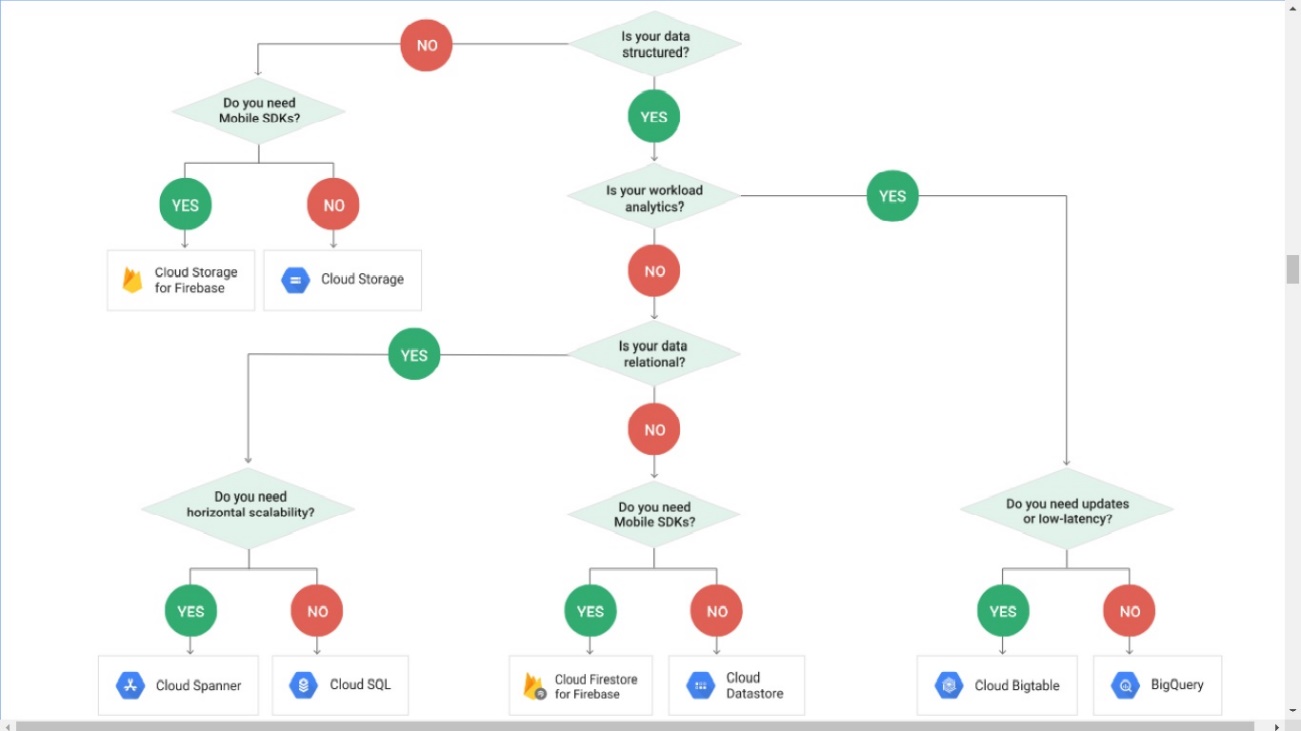
gsutil mb -c standard -l us --pap enforced -b on --retention 10y gs://gemb-platzi-storage-bucket-01

**Donde**:  
-c es la clase  
-l es el tipo de locación que deseamos  
--pap especifica el acceso público  
-b es activar el acceso uniforme  
**Avanzado**:  
--retention es el periodo de retención del recurso.

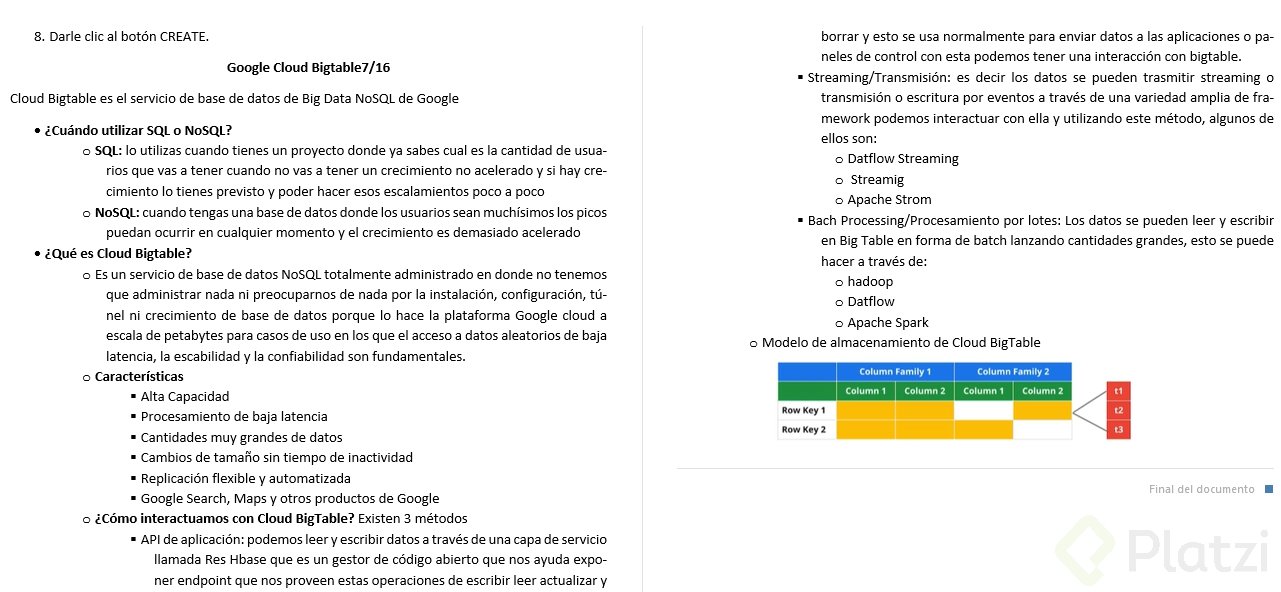
Crear tu primer bucket:

1. Seleccionar del menú tipo hamburguesa la opción Cloud Storage.
2. Darle clic a “Create Bucket”.
3. Darle un nombre único al “bucket” (puedes usar como prefijo el nombre del proyecto) y Continuar.
4. Seleccionar donde se almacenará (Multi-Región, Región Dual o Región) y Continuar.
5. Seleccionar la clase de almacenamiento (Standard, Nearline, Coldline, Archive) y Continuar.
6. Seleccionar control de acceso (Uniform, Fine-grained) y Continuar.
7. Escoger la Configuración Avanzada que necesites (Encriptación, política de retención,etc.).
8. Darle clic al botón **CREATE**.





**GOOGLE BIGTABLE**



**Google Cloud Bigtable**

Cloud Bigtable es el servicio de base de datos de Big Data NoSQL de Google, completamente administrado a escala de patabytes para casos de uso en los que el acceso de datos aleatorios de baja latencia, la escalabilidad y la confiabilidad son fundamentales

* Características
  + Alto procesamiento
  + Procesamiento de baja latencia
  + Cantidades muy grandes de datos
  + Cambios de tamaño sin tiempo de inactividad
  + Replicación flexible y automatizada
  + Google Search, Maps y otros productos de Google
* ¿Cómo interactuamos con Cloud Big Table?
  + Api de aplicación: Podemos leer y escribir datos atraves de una capa de servicio llamada rede Hbase, que es un gestor de codigo abierto que nos ayuda exponer el point que nos proveen estas operaciones de escribir leer actualizar y borrar y esto se usa normalmente para enviar datos a las aplicaciones o paneles de control
  + Streaming / Transmisión: Datflow Streaming, Saprk Streaming y Apache Storm
  + Batch Processing / Procesamiento por lotes: Los datos se pueden leer y escribir en Big Table en forma de batch (cantidades grandes) esto se puede hacer a través de hadoop, Datflow, Apache Spark.

En esta clase creamos una instancia de Bigtable con el nombre quick-start-instance, id quick-start-instance y cluster id quickstart-instance-c1. El tipo de disco fue SSD con un solo nodo. Finalmente la región fue us-east1 y la zona us-east1-c.

Desde la CLI podemos crear nuestra instancia ocupando la herramienta cbt, la el comando para crear la instancia de esta clase quedaría así:

cbt createinstance quick-start-instance "quick-start-instance" quickstart-instance-c1 us-east1-c 1 SSD

* quick-start-instance es el id
* "quick-start-instance" es el nombre
* quickstart-instance-c1 es el id del cluster
* us-east1-c es la región
* 1 es el numero de nodos
* SSD es el tipo de disco

**• Creación de BigTable en Google Cloud Plataform**  
o Seleccionamos Home  
♣ Buscar Datastore, seleccionamos Bigtable  
♣ Créate instance, llenamos todos los datos que nos pide  
♣ Seleccionamos continuar, seleccionamos disco SSD  
♣ Seleccionamos continuar, selecciomos la región, requerimos un nodo  
♣ Créate  
o Activamos cloud Shell  
♣ Primero asociamos nuestra instancia a mi perfil de cloud Shell para mandar comando y a que base de datos  
• Echo Project = ‘gcloud config get-value project’ > ~/.cbtrc  
• Echo instance = nombreinstancia >> ~/.cbtrc  
o Creamos una tabla  
• Cbt createtable my-table  
o Autorizamos la Shell para interactuar con la api  
• Cbt ls  
o Agremos un colum family o agregar columnas  
• Cbt createfamily my-table cf1 //nombre de famili1  
• Cbt ls my-table  
o Ingresamos datos en la tabla  
• cbt set my-table r1 cf1:c1=test-value  
• cbt read my-table

## Cloud SQL

Este es el servicio administrado de Bases de Datos Relacionales. Puede ser MySQL, PosgreSQL y SLQ Server.

En caso de necesitar mayor rendimiento se puede escalar de forma vertical (un máquina más potente).  
Si se requiere más disponibilidad se puede optar por una arquitectura en dos zonas.

### Características claves

* Totalmente **administrada**
* Una **solución integrada**: Se puede acceder a ellas desde cualquier lado.
* **Confiable**: ES fácil configurar las replicas, copias de seguridad y activar el proceso de Failover (reemplazar una instancia cuando esta falla).
* **Migraciones sencillas** a CloudSQL: Database Migration Service ayuda a migrar las DB on premise a la Nube facilmente.

En este tipo de Bases de datos realizamos transacciones y deben cumplir los principios ACID.

* **A**tomicity: Asegura que todas las operaciones que una transaccion se realicen, y en caso contrario que sea posible regresar al estado anterior (rollback).
* **C**onsistency: Asegura que todas las transacciones se realicen con exito, los datos deben tener sentido.
* **I**solation: Dicta que las operaciones sean aisladas y transparentes, es decir, multiples operaciones ocurren de forma independiente y sin afectarse.
* **D**urabilty: Nos asegura que el resultado de una operación permanezca incluso cuando hubo un error.

### **Cloud Spanner en acción**

#### Crear instancia desde gcloud CLI

En esta clase creamos una instancia Cloud Spanner con nombre example-db, regional y en US central. Además de que ocupamos solo 1 nodo.

Para crear dicha instancia usando gcloud usamos el siguiente comando:

gcloud spanner instances create example-db --config=regional-us-central1 --nodes=1

#### Crear base de datos desde gcloud CLI

Para crear una base de datos llamada example-db-db en nuestra instancia example-db usamos el comando:

gcloud spanner databases create example-db-db --instance=example-db

#### Crear un schema con gcloud CLI

gcloud spanner databases ddl update example-db-db \

--instance=example-db \

--ddl='CREATE TABLE Singers (

SingerId INT64 NOT NULL,

FirstName STRING(1024),

LastName STRING(1024),

SingerInfo BYTES(MAX)

) PRIMARY KEY (SingerId)'

#### Insertar datos a nuestra DB con gcloud CLI

Para insertan un solo registro:

gcloud spanner rows insert --database=example-db-db \

--instance=example-db \

--table=Singers \

--data=SingerId=1,FirstName=Marc,LastName=Richards

#### Actualizar datos en nuestra DB con gcloud CLI

Para actualizar un solo registro:

gcloud spanner rows update --table=Singers --database=example-db-db --instance=example-db \

--data=SingerId=1,SingerName=Will

#### Eliminar datos en nuestra DB con gcloud CLI

Para eliminar un solo registro:

gcloud spanner rows delete --table=Singers --database=example-db-db \

--instance=example-db --keys=1

#### Leer los datos mediante SQL

Para consultar nuestros registros:

gcloud spanner databases execute-sql example-db-db \

--instance=example-db \

--sql='SELECT \* FROM Singers'

Google Cloud Platform

* Ir a Spanner
* Aceptamos
* Create Instance (Creamos una instancia de Spanner)
* Indicamos un nombre (example-db)
* En configuración ponemos (Regional, us-central1)
* Aceptamos capacidad de inicio (por defecto)

Dentro de la instancia que creamos de Spanner, creamos nuestras bases de datos:

* Create Database
* example-db-db ( en este caso pone dos veces db para distinguir de la instancia de Spanner)

NO CREA UN SCHEMA, ESTO LO DEFINIMOS DESPUES\_

* Create

Una vez dentro de la base ( te deja ahí luego de hacer el paso anterior)

* Create table

CREATE TABLE Singers {

SingerId INT 64 NOT NULL,

FIrstName STRING(1024),

LastName STRING(1024),

SingerInfo BYTES(MAX),

BirthDate DATE,

} PRIMARY KEY(SingerId);

* Submit (corre la sentencia SQL para crear la tabla)  
  Automáticamente al terminar vemos la tabla y podemos acceder a ella.

Ahora cargamos datos:

* vamos a Data
* Insert

Automáticamente nos da un script, lo que haremos es editarlo.  
Se editan los que están dentro de <FIrstName> = Nombre, eliminar los <>

Luego de terminar, hacemos click en RUN y por consola nos dice que inserto un registro.

En una aplicación grande usamos un script como el de arriba a la N, es decir con la cantidad de datos que quisiéramos.

Para editar, seleccionamos el registro y luego click en **EDIT**

El proceso que se ejecutara es update sobre la tabla Singer, y por consola tendremos el resultado.

Para visualizar:  
Vamos a Query y escribimos.

SELECT

\* FROM Singers

Seleccionamos TODO lo que tenga la tabla Singers

Luego RUN y en la pestaña RESULTS nos muestra el resultado de la ejecución del query.

**CLOUD FIRESTORE**

\*\*Documento \*\*= Conjunto de datos asociados a un mismo concepto  
Ejemplo: Producto, Nota, Jugador.

**Colección**: Conjunto de documentos.  
Cloud Firestore:

* Serverless = SIN SERVIDOR. (NO ME PREOCUPO DE LA INFRAESTRUCURA
* Tiempo Real
* Soporte Offline

Características:

* Escalable
* Confiable
* Flexible
* Disponible
* Transaccional

Algunos casos de uso pueden ser:

* Gestión de perfiles de usuario
* Inventarios en tiempo real
* Cambios de estados
* Sincronización de datos

USO IDEAL

App web

APP MÓVILES

DE SERVIDOR

En consola de GCP

* 01 - Menú
* 02 - Firestore
* 03 - Elegir modo de operación (no se puede cambiar)
* 04 - En este caso NATIVO
* 05 - Ubicación (us-east4(Northrn-Virginia))
* 06 - Crear base de datos (esperar unos segundos)  
  Podemos crear manualmente o mediante carga por procesos
* 07 - Abrír la consola de GCP
* 08 - node createTestData 5  
  *En este caso el numero 5 es el equivalente a la cantidad de documentos*
* 09 - ls  
  *Listamos todo incluyendo ahora .csv*
* 10 - cat customers\_5.csv
* 11 - node importTestData customers\_5.csv
* 12 - Vamos a Firestore y actualizamos
* 13 - Tenemos la colección y los documentos

## Firestore en acción

En esta clase creamos una instancia de Firestore en Native Mode, ubicada en us-east4.

Ocupando el siguiente comando podemos crear una base de datos Google Cloud Firestore Native:

gcloud firestore databases create --region=us-east4

## Comparativa de las opciones de almacenamiento

