# Object Storage – Part 2

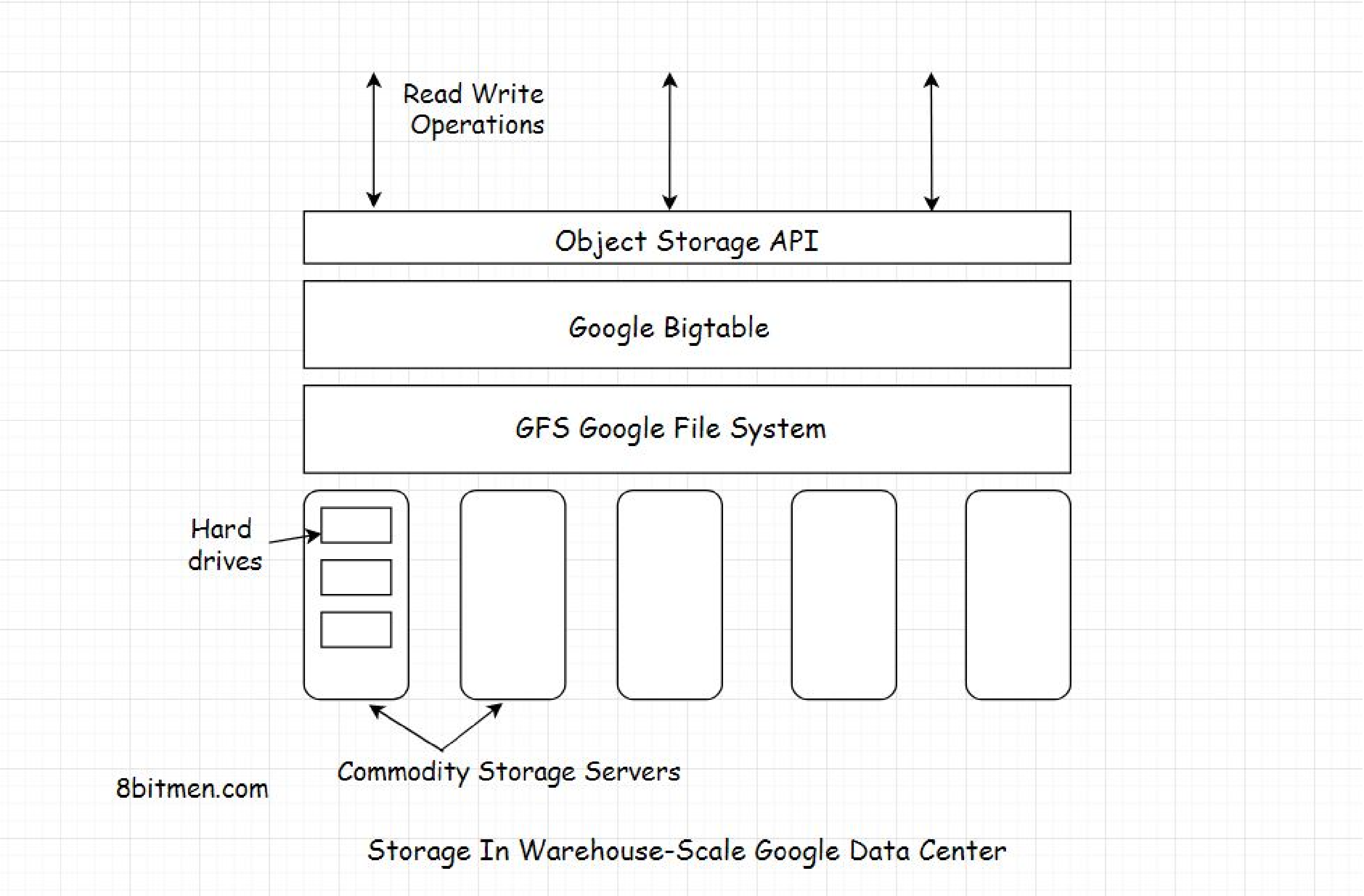
This lesson continues the discussion on object storage.

**We'll cover the following**

- Overview
- Object storage in the cloud

## Overview #

The data stored in the cloud object store is accessed and managed via *REST APIs*. Cloud providers use the commodity server approach to implement their object stores. We already know what commodity servers are, so let's take a look at the cloud object store infrastructure.



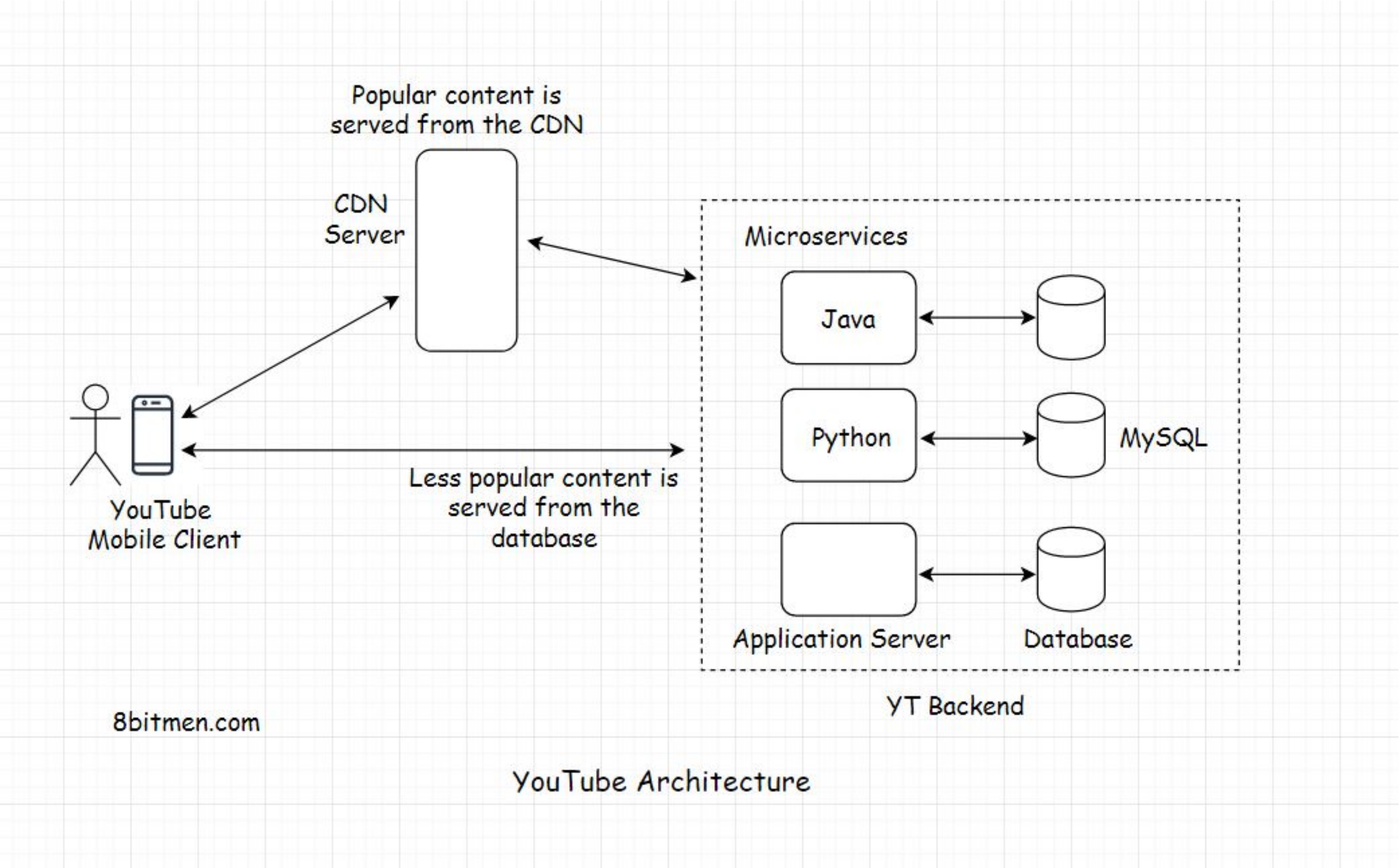
I wrote a post on my blog a while back about how *YouTube* manages petabyte-scale data without running out of storage space. Here is an excerpt from the post:

All the data is stored in the hard drives in massive warehouse-scale *Google* datacenters. The data that is stored in the physical storage is managed by the *Google File System* and *Bigtable*.

*GFS Google File System* is a distributed file system developed by *Google* to manage large scale data in a distributed environment.

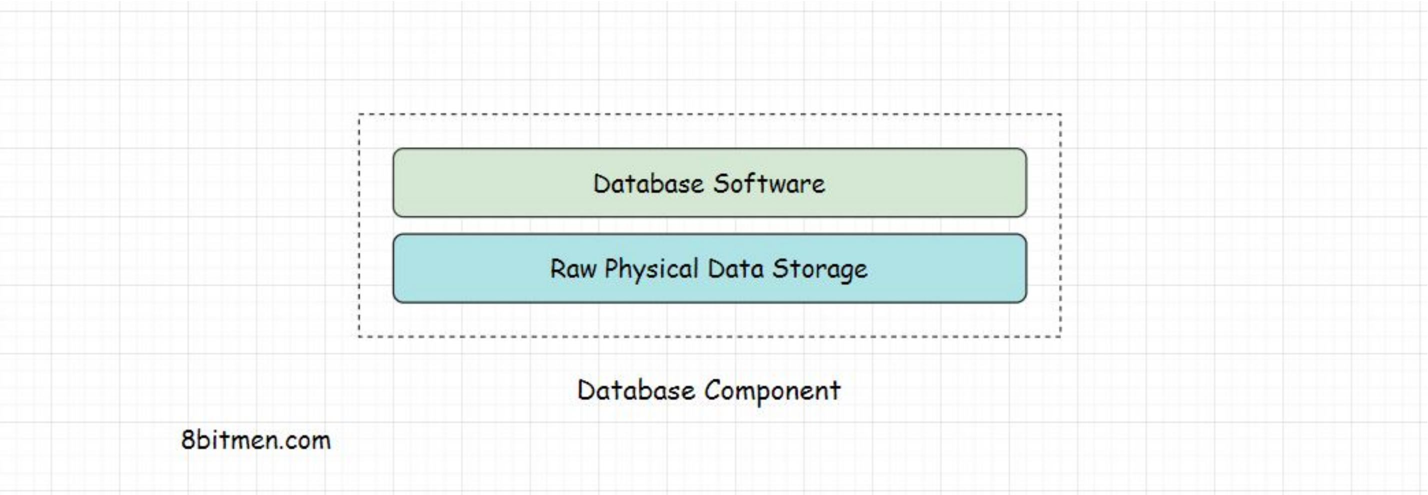
*BigTable* is a low-latency distributed data storage system built on *Google File System* to deal with *petabyte-scale* data spread over thousands of machines. It's used by over *sixty* Google products.

The raw video data goes into hard drives. All the *relationship data, metadata, user preferences, profile information, account settings, contextual data required to fetch the video from the storage*, etc. goes into *MySQL*.



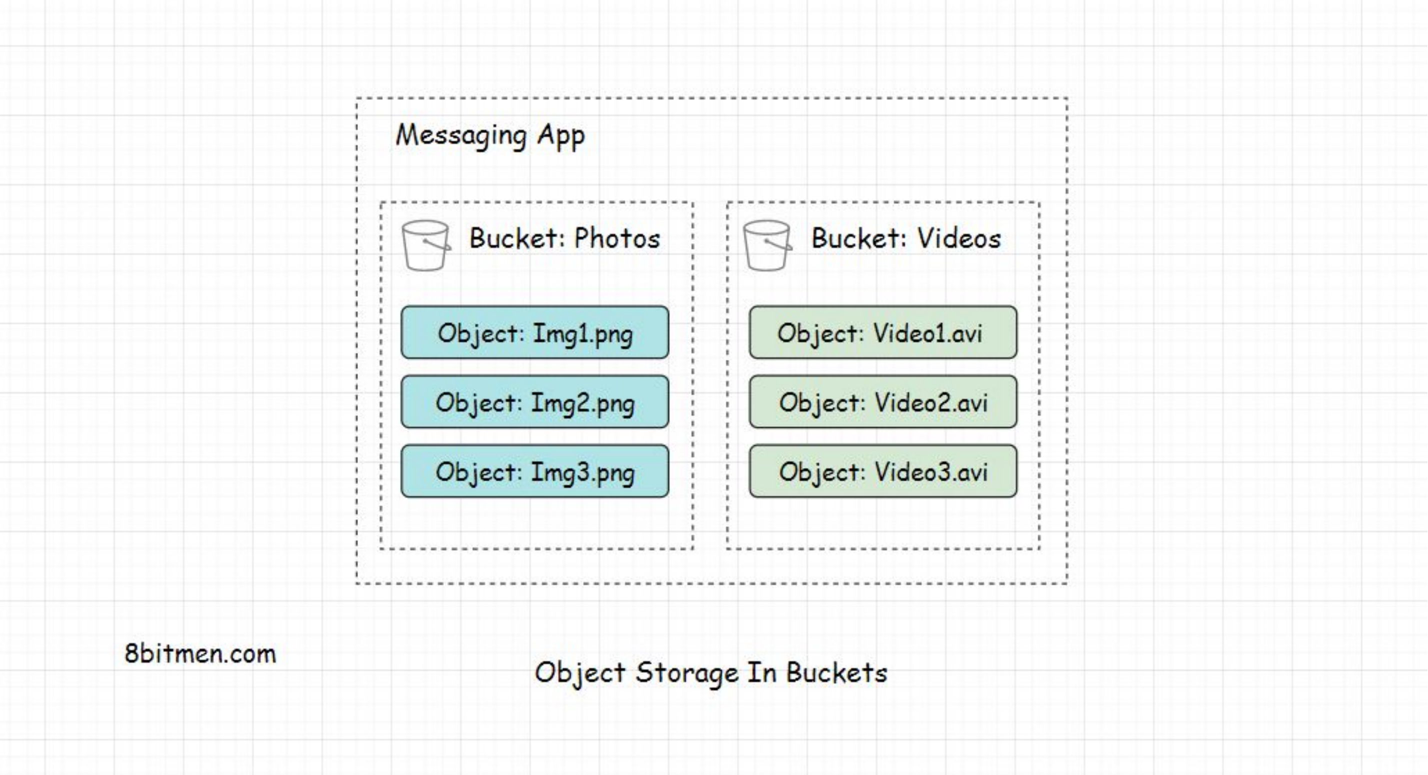
There is an *object storage API* layer that runs on top of *Google BigTable* that facilitates data read-write operations.

Just in case you are wondering what the difference between a *database* and *data storage* is, data storage entails the storage of raw data in physical storage like *HDDs* and *SSDs*, while databases contain software that manages the raw data. Both the database software and the physical data storage together make the database component in a web application.



## Object storage in the cloud #

*Object storage* is like the native storage for the cloud. The objects that can be files of any format are stored in the *buckets*. A bucket is a container for the object. A project may contain multiple buckets.



To store objects, a developer has to first create a bucket. Once the bucket is created, he can upload the objects to the bucket using the *REST API*. Any object we store in a cloud object store should be contained in a bucket, and there is a reason for that.

Just like containers isolate the workloads from the outer world, buckets isolate objects from the outer world. Buckets facilitate access management; only the entities having the right authorization can access the objects. Buckets also facilitate data encryption, object versioning, object access management, and so on.

In a project, we can create different buckets based on the type of objects being stored. For instance, videos can be stored in one particular bucket and user images can be stored in another bucket.

When creating a new bucket, we give it a unique name, geographic location (where we want our bucket to reside), and storage class. I'll talk about storage classes in the next lesson.

The unique identifier of the object, also known as the *key*, the bucket, and the object version uniquely identify an object in an object-store.

Developers can interact with the cloud object store via the *command line tools and browser-based console, using the SDKs*, *client libraries*, and also the *REST APIs*.

This discussion will be continued in the next lesson.