## Key-value server on Google cloud

Gowtham Chowta

**Setup:**Two VM’s are created. Server is hosted on one VM and the client connects to this server from other VM. I am using a E2-medium machine (2 vCPU, 4 GB Memory) for the server machine and E2-micro machine (2 vCPU, 1 GB Memory)

**Approach:**

I was using the gcloud API and the python client libraries to handle the infrastructure as code. We need the application credentials to access the gcloud apis and ssh keys to connect to the virtual machine created.

1. Application credentials setup. (This step has to be done manually).
2. Authentication. Generate the public and private key file and transfer the public key to the specific GCP project.
3. Networking. Setup a default network and firewall rules.
4. Instantiate the server machine and install all the dependencies needed.
5. Instantiate the client machine and install all the dependencies needed.
6. Start the server.
7. Create a client request to the server and validate the request-response format in the console or in appropriate storage services.
8. Cleanup. Delete all the VM’s, all the storage buckets.

**Operation:**

Client can perform either SET or GET requests on to the server. Client will be transparent about the underlying storage server is using.

**Storage:**

*Default*:

This is the default storage. On every set request the data is stored on the server local file system. If the server is restarted/corrupted the data is lost.

*Firestore:*

This is the Google’s document database. Again, on every set request the data is stored directly on to the google firestore database. And every get request calls the firestore service to retrieve the value.

*Cloud storage:*

This is the Google’s blob storage. This is the simple storage usually used to store large files/images. Again, every set request a blob/file is created and saved. And every get request checks the existing bucket if there exists a file with that name.

**Latency:**

Using the firestore the average request time is 40-50ms.

Cloud storage set request was taking up to 300ms and the get request was taking 40ms.

While the default storage as expected was the least among the three. It is taking less than 2ms to process any request as it is directly storing in the servers local storage.

I chose firestore as it was the Google’s No-SQL database which is idea to store the key-value stores. And I went with the cloud storage which is much simpler, cheaper and slower. As expected, the firestore was taking much lesser time when compared to the cloud storage. Usually, cloud storage is used to store large files/images, which is why was not an ideal choice for key-value server. It also has slower write speeds, which is again not acceptable if we want to build a key-value server.

Things I could have improved in my implementation:

* I wanted to try the Google’s in memory key-value store which is the Google’s version of Redis. I was not able to make it work and did not get enough time to spend on it.
* The implementation I had has a mix of Google’s client libraries and gcloud CLI. I wish I could have written everything in one version to keep it cleaner.