**ARCHITECTURE**:

A computer screen capture

Description automatically generated with medium confidence

Figure 1 – Folder Structure

* 1 TCP/IP server built using python3 which interacts with clients and performs the operations
* 1 Database, which is a collection of files (1 file per key) stored in the ‘data’ folder by the server for native storage
* Client process that initiates set and get requests to the server
* Everything.sh is the script file to run the entire workflow.
* prashasti-karlekar-fall2022-firebase.json contains public/private credentials to connect to the Firestore for my project used for this assignment
* bucket\_data.json is a dummy starter JSON file used for bucket storage on google cloud

**OPERTIONS PERFORMED**:

1. The project “prashasti-karlekar-fall2022” was created with region set to northamerica-northeast1 and zone set to northamerica-northeast1-a and SSH keys were generated using ssh-keygen command, followed by gcloud init to save the changes performed.

Text

Description automatically generated

1. Default network was created, followed by creating specific firewall rules.

Text

Description automatically generated

1. Instances : client-instance and server-instance were created and the required files were transferred onto the VMs.

Text

Description automatically generated

1. To run the VMs select 4. This installs the packages required to run the programs and also runs the server and client programs. The output of this is a client menu which waits for user request.

Text

Description automatically generated

Text

Description automatically generated

1. For the native storage, the set operation writes to a newly created file with filename of the key in the data directory and the get operation reads the requested key’s value from the data directory
2. For Firestore, the cloud firestore “Prashasti-karlekar-fall2022” was created. The credentials file for this store was then downloaded and is present in the project folder. The database contains a collection “KVStore” which then contains a document called “test”. This document is written to/read from the server program to utilize firestore as a database for key-value store.
3. For Google Cloud bucket storage, the commands in script file in create\_bucket() was performed.

gcloud storage buckets create gs://prashasti\_kvstore --location=US-EAST1 --uniform-bucket-level-access

This creates a bucket in Google cloud with name prashasti\_kvstore.

Also, an empty JSON file was copied onto the bucket for its storage utilization.

1. The client feeds a command with a specific storage option from {1,2,3} where 1 is native storage, 2 is Firestore and 3 is Google Cloud Bucket. Accordingly the server processes the client request and sends the output to the client.
2. To stop the VMs, enter option 5. This stops the client-instance and server-instance.

Text

Description automatically generated

1. To delete the storage bucket, enter 6.
2. To delete the server-instance and client-instance, enter 7.

Text

Description automatically generated

1. To delete default network and set firewall rules, enter 8.

Text

Description automatically generated

IMPLEMENTATION:

1. TCP- socket server:

The server was implemented using the socket library offered by python. Upon creation of the server socket, it is associated with a specific network interface of server-instance running on Google cloud and port number. Once it accepts a connection from a client, it returns a new socket object representing the connection and a tuple holding the address of the client i.e. (host, port). This socket is then used to communicate with the client. Based on the client request, the set or get operation is performed for the specified storage option and a response for the request is sent back to the client.

* SET COMMAND:

This stores the value for later retrieval.

set <key> <value-size-bytes> <storage\_option> \r\n

<value> \r\n

The set command is whitespace delimited. It takes the key to be stored, the length of the value associated with the key, the storage option where this key needs to be stored and the value itself as the arguments. The server responds with either "STORED\r\n", or "NOT-STORED\r\n" depending on whether the key was stored or not.

Text

Description automatically generated

SET operation for Native Storage

Text

Description automatically generated

SET operation for Firestore

Text

Description automatically generated

SET operation for Google Cloud Bucket Storage

* GET COMMAND:

This will fetch the data corresponding to the key and return it to the client

get <key> <storage\_option>\r\n

The get command gets the value associated with the given key from selected storage. It accepts the key name and storage option as the argument.

Text

Description automatically generated

GET operation for Native Storage

Text

Description automatically generated

GET operation for Firestore

Text

Description automatically generated

GET operation for Google Cloud Bucket Storage

1. Client:

The client program connects to the server and makes get or set requests. It sends a command line/data block and receives a response from the server which indicates the success or failure of request made.

**STORAGE BACKENDS - WHAT & WHY:**

**Native Storage**:

When a large number of written and read from the data store, affects the set operation in terms of space requirements. But the get operation is fast. With the increase in the number of keys, the space requirements increase and hence does not make this model very scalable. Few limitations and improvements of native storage:

1. The current implementation of the server has a single process concurrency which allows easy caching and database access but has potential disadvantages like blockage of services when busy.
2. In-memory data structure stores like Redis can be implemented for better performance and improve the scalability.
3. The response time of server is merely 1/100th of a second for few requests, but the same can be tested for 1000s of client requests.
4. There are size limits associated with the value provided and this can be enhanced by accepting input files from the system.

**Cloud Firestore**:

Why I chose this:

This NoSQL data model stores data as collections of documents which is easier to compose as a key-value store. The documents store key-value pairs in dictionary form.

Advantages:

* Ideal for key-value pairs since the structure is similar to JSON format
* Performs better when scaled largely as it used subcollections within documents
* Since the queries are indexed, the performance of the query does not depend on the data set but instead on the result set

Total costs:

Pay for what you use. According to the official site, it is $11.10/month for read/write costs + $1.04/month for storage/networking costs

Performance:

* Firestore is highly scalable and supports parallel reads.
* Retries the failed transactions to deal with transient errors
* Updating to a single document more than once per second leads to high latency and timeouts.

**Google Cloud Bucket:**

Why I chose this:

Buckets are the basic containers that holds the data. Google Cloud provides advantages like easy identity management, object versioning and authentication.

Advantages:

* With multiple storage bucket locations, it allows multiple redundancy options.
* Provides a low cost for data backup and archival
* Provides uniformly control access to Cloud storage resources
* Enable Pub/Sub notifications for Cloud Storage