Data Store Services

Memi Lavi
www.memilavi.com

Data Store Services

- GCP offers various types of data store services
- Offer various data store types, distribution and speed
  - ie. SQL, NoSQL, Document, In-memory and more
- Extreme pricing variance
- Make sure to select the right data store service for your needs

Data Store Services Types

- Cloud SQL
- Spanner
- AlloyDB
- BigTable
- BigQuery
- Firestore
- Cloud Storage
- Memorystore

Data Store Services Types

- We'll go through these services in this section
- At the end of it we'll learn how to select the best data service for your needs

Data Store Services Types

- Factors to consider when comparing databases:
  - SQL vs NoSQL
  - SLA
  - Backups
  - Security
  - Distribution
  - Cost

Database on VM

- Databases can be installed on a VM instance
- In addition to the managed data stores
- ie. No managed Oracle in GCP, must be installed on VM
- We won't discuss this option in this section

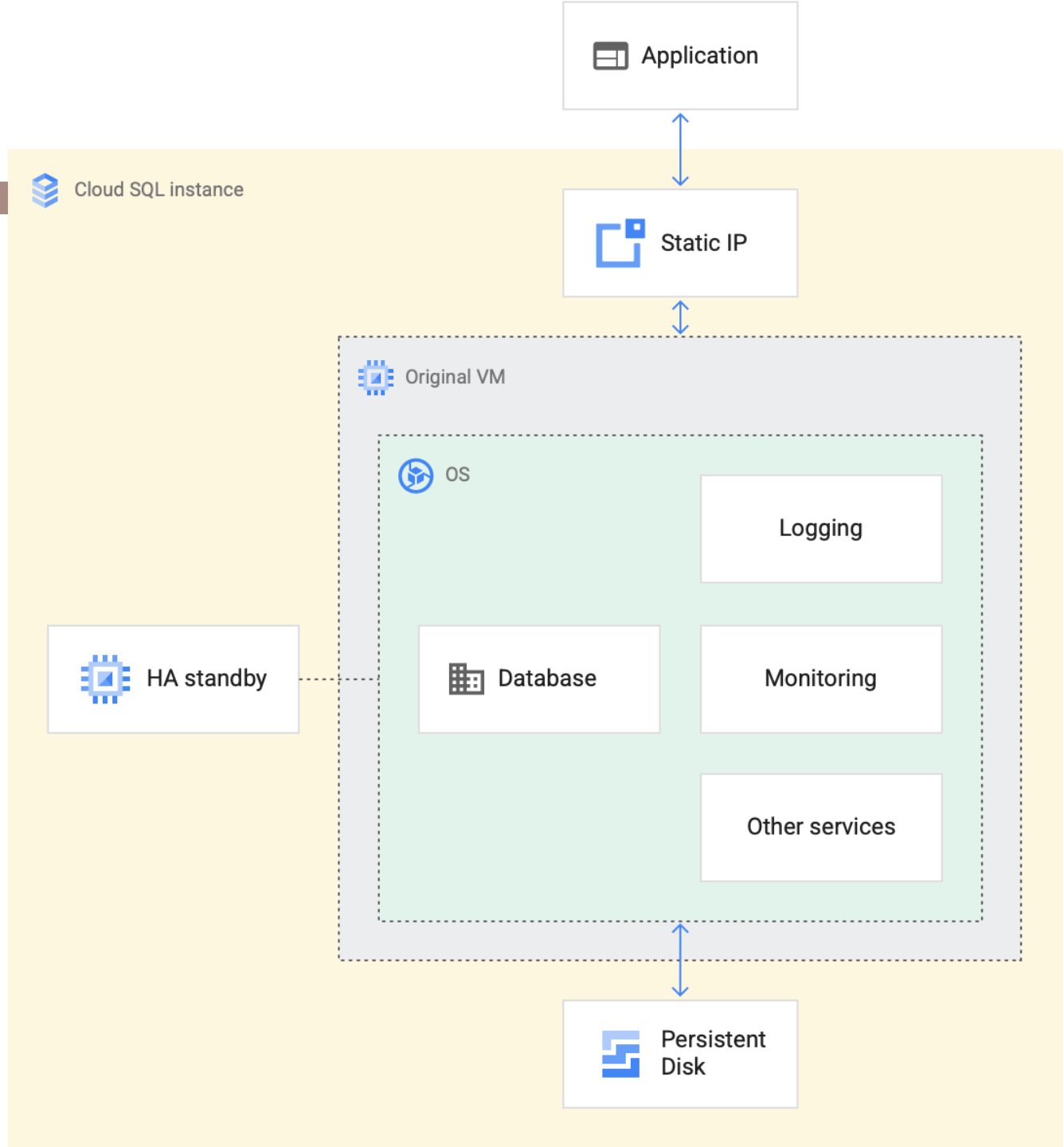
Cloud SQL

- Fully managed relational database
- Regional
- SQL Server, MySQL and PostgreSQL flavors
- Cost effective
- Highly available

Cloud SQL

- Every instance of CloudSQL is built on VM instance with the selected database installed
- If HA (high availability) is configured, another instance is created in the same region, different zone

Cloud SQL


Application
Cloud SQL instance
Static IP
Original VM
OS
Logging
HA standby
Database
Monitoring
Other services
Persistent Disk

Cloud SQL

- Database are managed using the regular database management tools
- Examples:
  - MySQL Workbench for MySQL
  - Toad for PostgreSQL
  - SQL Server Management Studio for SQL Server

Cloud SQL Backup

- Database is backed up daily
- Begins in a configurable 4-hour window
- Backup is stored in a region close the instance region

Cloud SQL Encryption

- Data is automatically encrypted using the AES-256 algorithm
- Encryption keys are automatically managed by Google
- Customers can bring their own keys

Cloud SQL Editions

Enterprise

- All the core capabilities of Cloud SQL
- SLA: 99.95%

Enterprise Plus

- All the core capabilities of Cloud SQL
- Better performance
- Better availability
- Costlier
- SLA: 99.99%
- Not available for SQL Server

Cloud SQL Pricing

- Based on:
  - Edition
  - vCPU used
  - Memory
  - Storage

Cloud SQL Pricing

Cloud SQL for MySQL

DB-STANDARD-1 HA

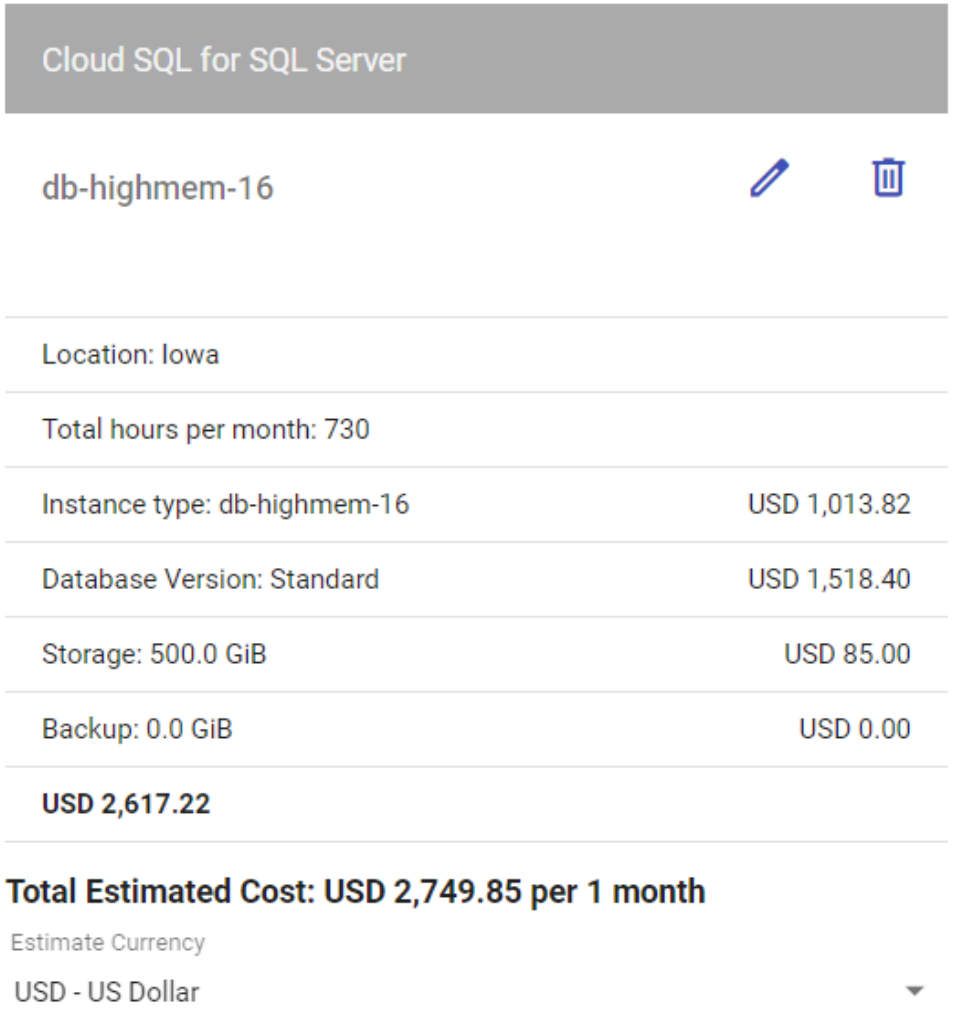
Edition: Enterprise	
Number of instances: 1	
Location: Iowa	
Total hours per month: 730.0	
Instance type: db-standard-1	USD 98.62
SSD Storage: 100.0 GiB	USD 34.00
Backup: 0.0 GiB	USD 0.00
USD 132.62	

Total Estimated Cost: USD 132.62 per 1 month

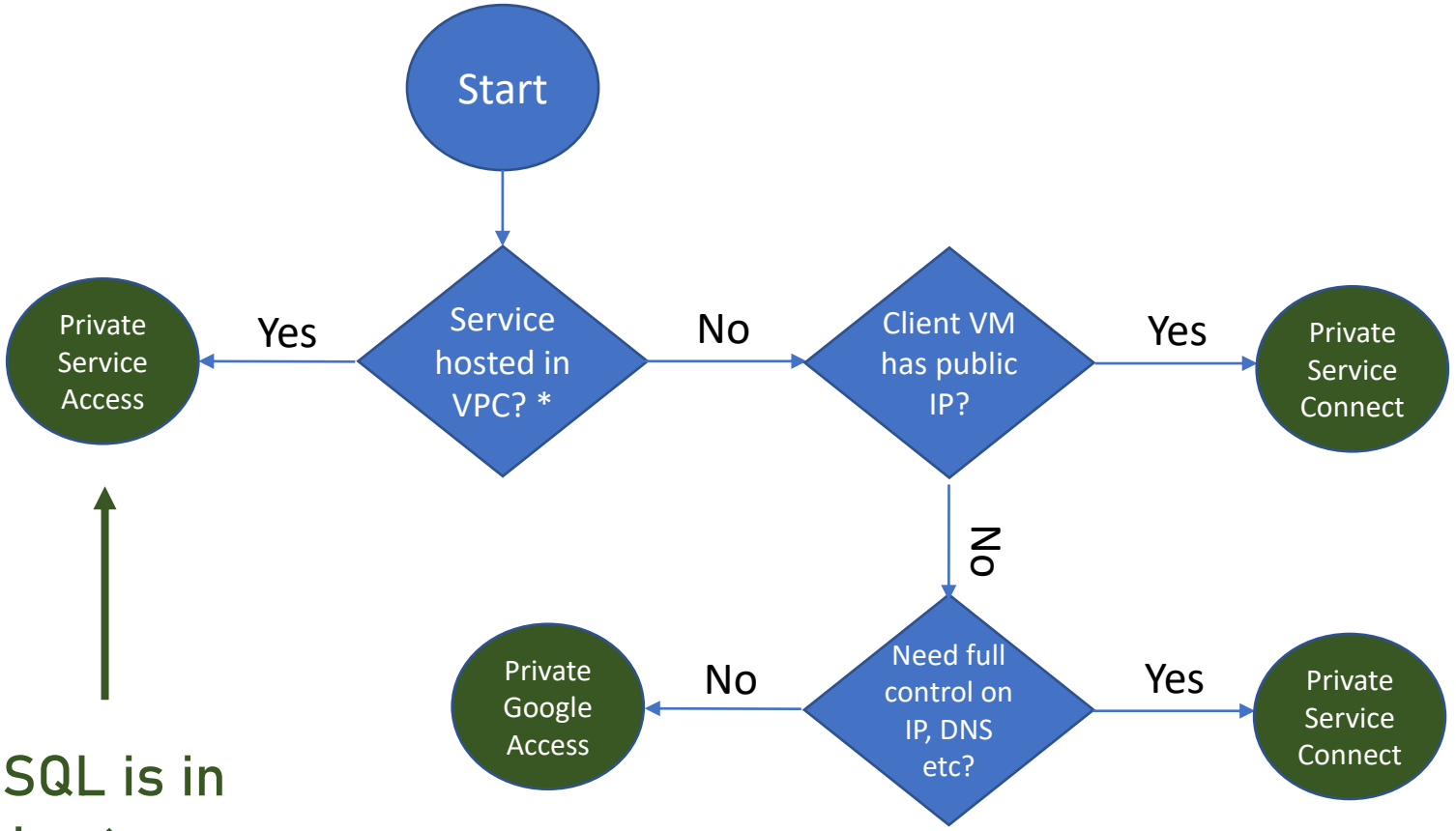
Estimate Currency

USD - US Dollar

Cloud SQL Pricing



Choosing Private Access Implementation



Cloud SQL is in this list

* See updated list here: https://cloud.google.com/vpc/docs/private-services-access#private-services-supported-services

Connecting App Engine to Cloud SQL

- Two options:

Public IP

- Usually not the most secure option
- No single outbound IP for App Engine
  - As opposed to VM

Private IP

- The most secure way to connect
- Implementation differs between App Engine type

Connecting App Engine to Cloud SQL

Standard

- Use Serverless VPC Access to connect to the VPC with the database's private IP
- Connect to the database's Private IP

Flexible

- Service must be in the same VPC of the Cloud SQL private IP
- By default deployed in the `default` VPC
- Use the database's private IP to connect

Connecting App Engine to Cloud SQL

- Basically – just change the connection string
- We can do that by redeploying the code
- Later we'll learn about better places to store connection strings

Connecting App Engine to Cloud SQL

- Basically – just change the connection string
- We can do that by redeploying the code
- Later we'll learn about better places to store connection strings

AlloyDB

- Fully managed PostgreSQL-compatible database
- Extremely fast – up to 4X faster than PostgreSQL
- Highly available, SLA 99.99%
- Regional
- Built-in generative AI
- More expensive than CloudSQL

AlloyDB High Availability

- AlloyDB is deployed in a cluster
- Contains all instances, databases, logs and other metadata
- Deployed in a single VPC

AlloyDB High Availability

- Two types of instances in a cluster

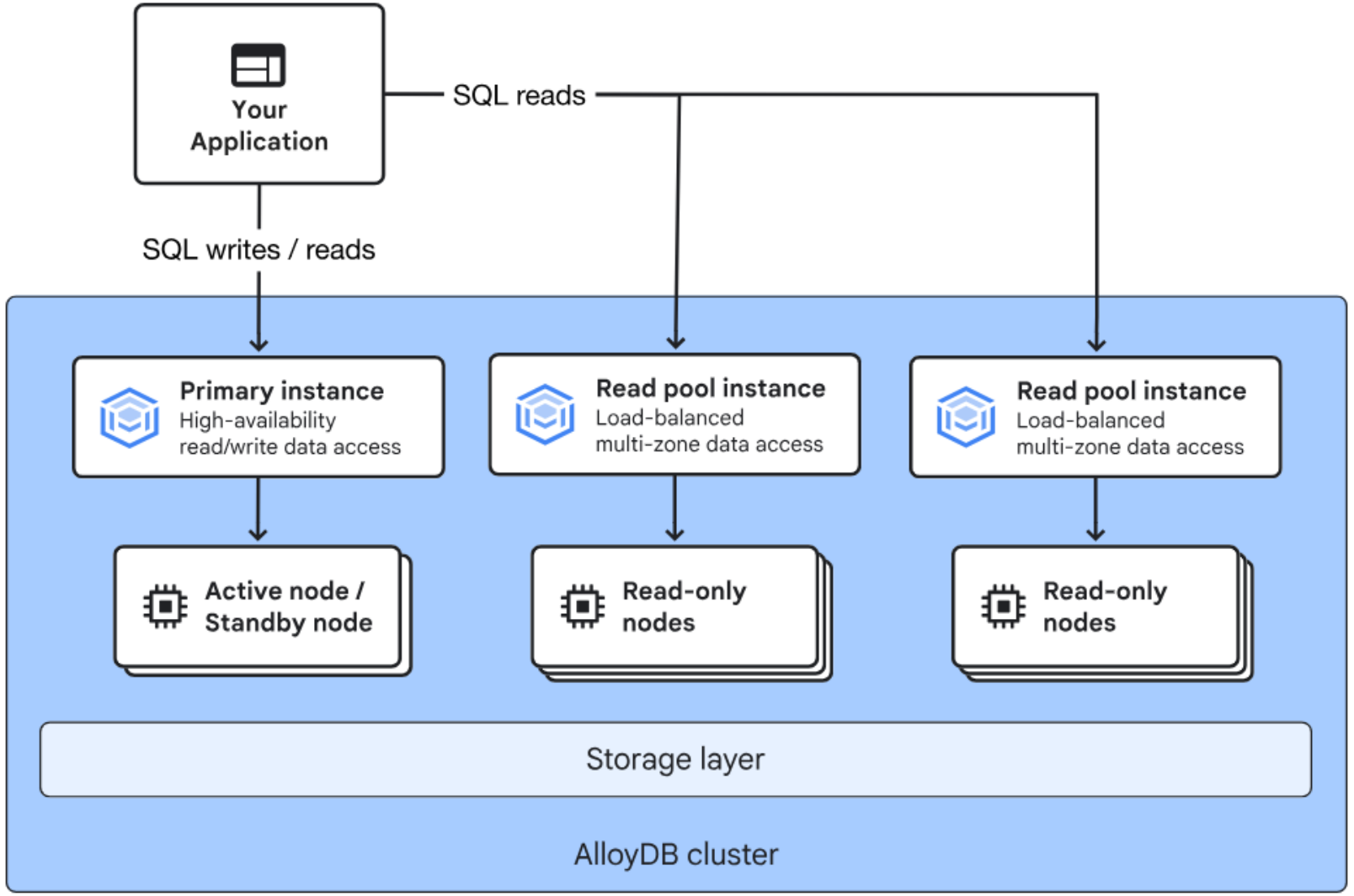
Primary

- Provides read/write access to your data
- Can have a standby node
- Automatically receives requests based on availability of the primary instance

Read pool

- Provides read access to the data
- Up to 20 read instances per cluster

AlloyDB Cluster



AlloyDB High Availability

- AlloyDB cluster can be replicated cross-region
  - To a secondary cluster
- Create cross-region disaster recovery support

AlloyDB Backup

- Continuous backup
- Microsecond granularity
- Restore to any point in time up to 14 days in the past
  - Can be extended to 35 days
- Automated daily files backup

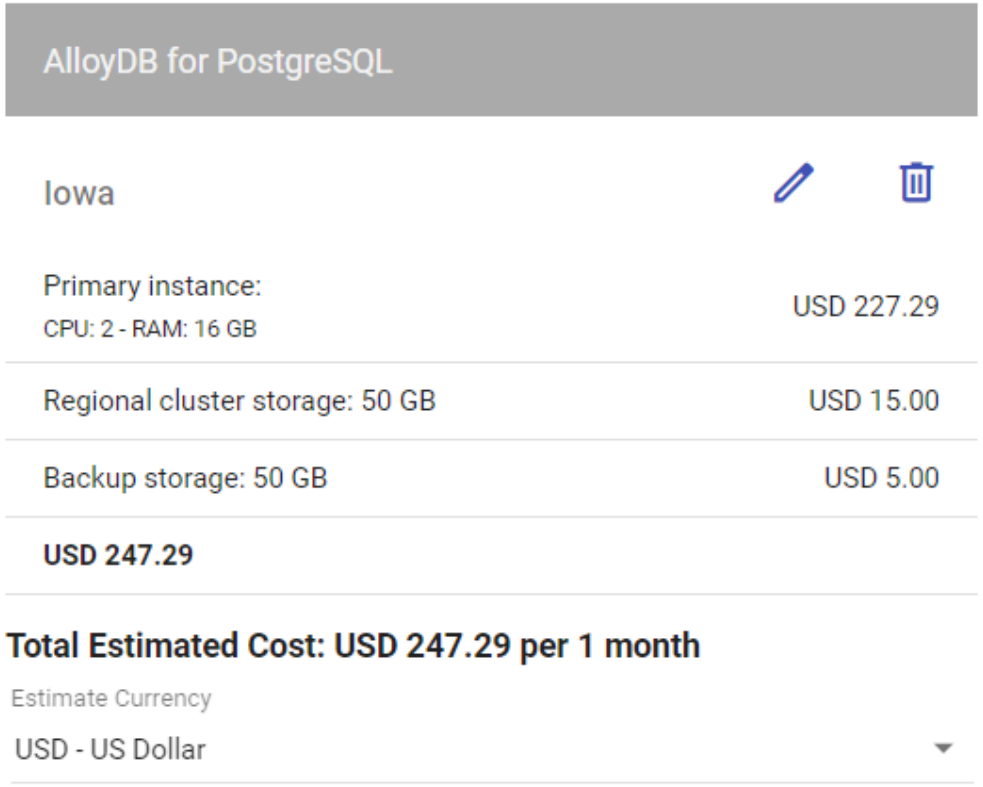
AlloyDB Omni

- Downloadable edition of AlloyDB
- Runs in container
- On Linux OS
- Fully compatible with PostgreSQL

AlloyDB Pricing

- Based on:
  - vCPU used
  - Memory
  - Storage

AlloyDB Pricing



Spanner

- Fully managed relational scalable database
- Distributed across regions
- Highly available
- SLA: 99.999%
- Multi dialect

Spanner Dialects

- Can choose between:
  - GoogleSQL (used also by BigQuery)
  - PostgreSQL

Spanner Capacity

- Capacity is specified in either:

Processing Units

- Minimum is 100 PU
- 1000 PU = 1 node
- Storage is allocated based on PU
- 100 PU =~400GB

Nodes

- Represents server task
- The more nodes = more distribution

Spanner Regional vs Multi-regional

- When deploying Spanner we choose type of deployment

Regional

- Three read-write replicas in the region
- Each in different zone
- Resilient for zone failures

Multi-regional

- Two regions are read-write
- Additional read-only regions
- Better reliability and availability
- Higher latency

Spanner Pricing

- Based on:
  - Compute capacity (measured in Processing Unit)
  - Storage
  - Backup storage

Spanner Pricing

Cloud Spanner	
Spanner nodes: 4	USD 14,600.00
Storage: 1,500 GiB per month	USD 1,050.00
Region: North America (Iowa/South Carolina/Oregon/Los Angeles/Oklahoma)	
USD 15,650.00	

BigTable

- Fully managed scalable NoSQL database
- Key-value store
- Distributed across regions (up to 8)
- HBase compatible
- Single digit millisecond latency
- SLA: 99.999%

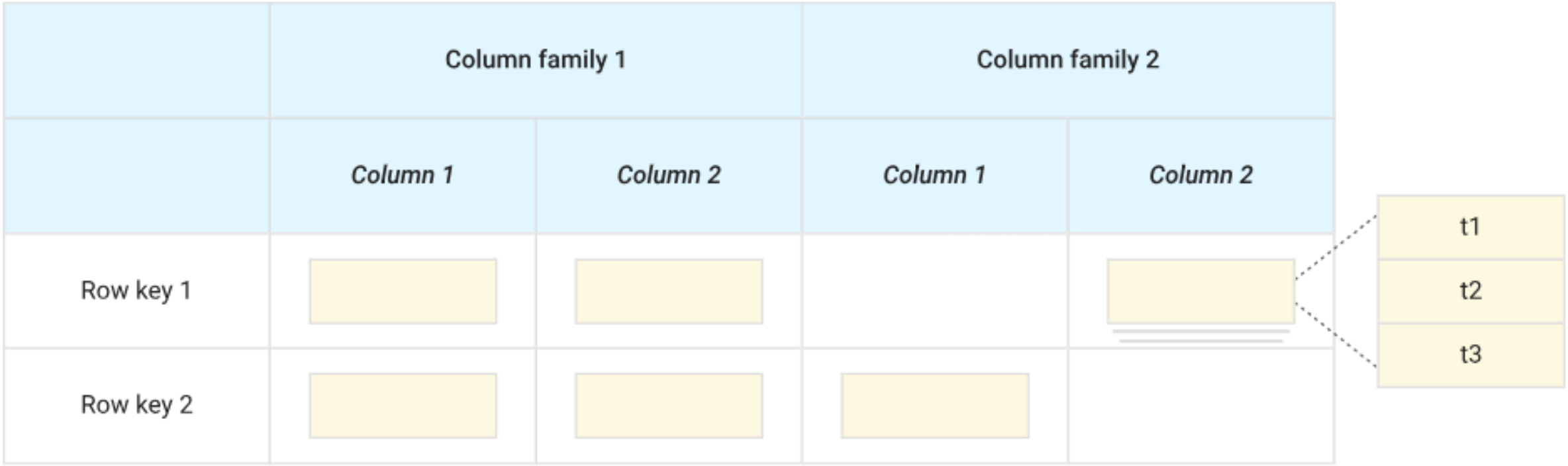
BigTable

- Can scale to billions of rows and thousands of columns
- Can store petabytes of data
- Simple to manage
  - Automatic upgrades
  - Automatic restarts
  - Automatic data maintenance
  - Automatic replication across clusters

BigTable

- Stores data in tables
- Each table has rows and family of columns
- Each column in the family is identified by column family and column identifier
- Columns can be unused in a row
- Each cell has a unique timestamp

BigTable


Column family 1
Column family 2
Column 1
Column 2
Column 1
Column 2
Row key 1
Row key 2
t1
t2
t3

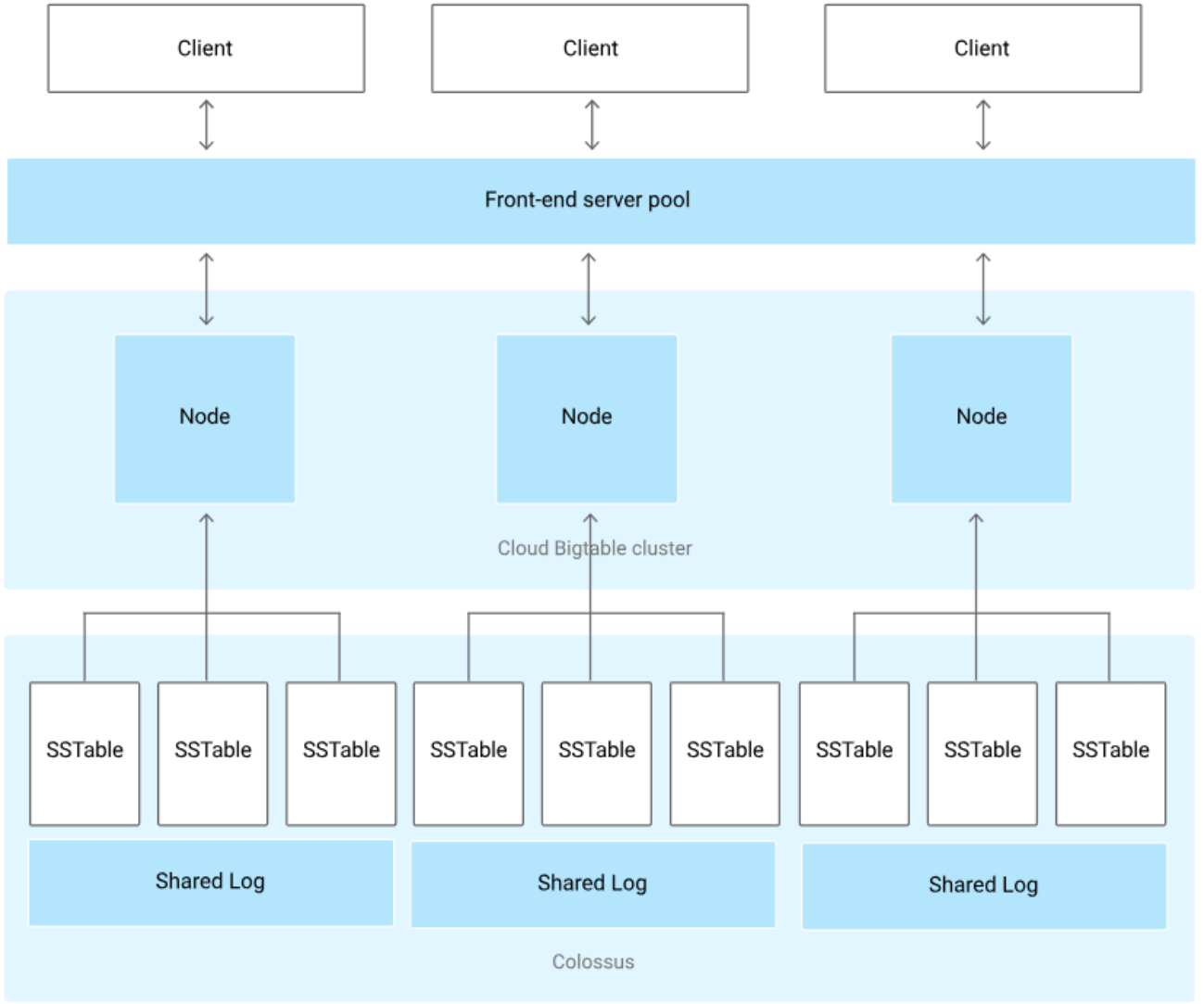
BigTable

- Data is stored in Colussus, Google's internal durable file system
- Backups is handled automatically by BigTable
- Expires after 90 days

BigTable Architecture

- *Instance* is the top level component
- Contains one or more *Clusters,* each in a different zone
- Each cluster has at least 1 *node*
- The node is the actual compute resource that processes the data
- A table belongs to an instance, not a cluster

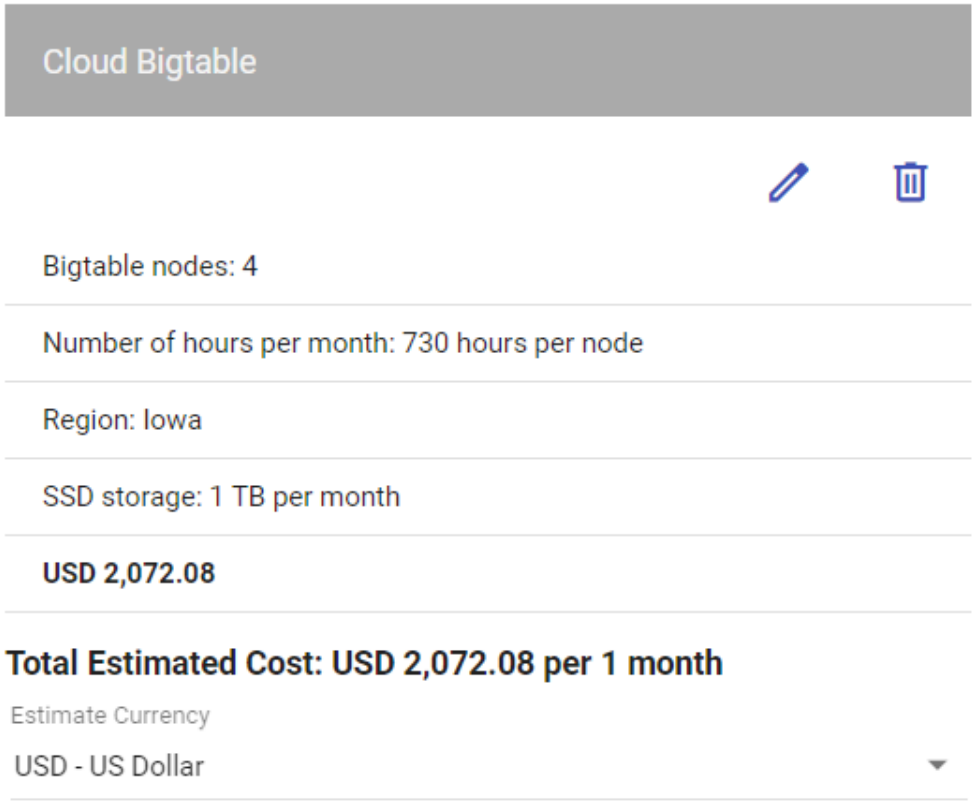
BigTable Architecture



BigTable Pricing

- Based on:
  - Type of instance and number of nodes
  - Storage

BigTable Pricing



Data Store

Data Store

SQL

NoSQL

SQL Database

- Stores data in tables
- Tables have concrete set of columns

Column Name	Type	Nullable?
OrderId	Numeric	No
OrderDate	DateTime	No
CustomerId	Numeric	No
DeliveryAddress	String	No

SQL Database - Relationships

Column Name	Type	Nullable?
OrderId	Numeric	No
OrderDate	DateTime	No
CustomerId	Numeric	No
DeliveryAddress	String	No

Column Name	Type	Nullable?
OrderItemId	Numeric	No
OrderId	Numeric	No
ItemName	String	No
Quantity	Numeric	No

SQL Database - Transactions

- Atomic set of actions
- ACID:
  - Atomicity
  - Consistency
  - Isolation
  - Durability

SQL Database - Querying

- Using SQL
- Structured Query Language
- Very mature

```
Select OrderID, OrderDate, CustomerId, DeliveryAddress
From Orders
Where OrderDate >'01/01/2018'
```

NoSQL

- Emphasis on scale and performance
- Schema-less
- Data usually stored in JSON format

NoSQL - Transactions

- Eventual Consistency
- Data can be temporarily inconsistent

NoSQL – Querying

- No standard
- Can be frustrating…

Data Store - Summary

SQL Databases	NoSQL Databases
Not Huge	Huge
Structured Data	Un- or Semi- Structured

Designing BigTable Schema

- Before working with data in BigTable we need to design its schema
- Different from relational database schema
- Focus on reads
  - How to ensure reads are as fast as possible

Designing BigTable Schema

- Think of the following elements:

Table

Column families

Columns

Rows

Cells

Row keys

Designing BigTable Schema

- Table:
  - Related data should be in the same table
  - Avoid distributing related data in multiple tables
    - As you'll do with relational database
  - Many tables cause high latency

Designing BigTable Schema

- Column families:
  - Related column should be in the same family
  - Example: All columns related to the order metadata (date, user, store etc.)
  - Up to 100 column families per table
  - Use short name for column families
    - You'll use it in your data operations

Designing BigTable Schema

- Columns:
  - Treat column qualifier as data
    - Saves space
    - Unintuitive…

Designing BigTable Schema

- Columns:
  - Instead of:

OrderId	OrderItem	ItemName
17	86	Harry Potter
17	13	Rama
17	101	Exhalation

Designing BigTable Schema

- Columns:
  - Use:

OrderId:17	Item86:Harry-Potter	Item13:Rama	Item101:Exhalation

Designing BigTable Schema

- Rows:
  - All data of entity should be in a single row
  - No more than 100MB in a single row

Designing BigTable Schema

- Cells:
  - No more than 10MB in a single cell

Designing BigTable Schema

- Row keys:
  - Perhaps the most important element of the schema
  - The only indexed element in the row
  - Should query based on the row key
  - Other types of queries trigger a full table scan

Designing BigTable Schema

- Row keys:
  - Keep it short (no more than 4KB, preferably much shorter)
  - Store multiple delimited values in the key
    - These values will be used by queries
  - Example: `readit#order#17`

Working with BigTable Data

- Not using SQL
- Write and Read requests
- Usually using client libraries instead of calling the API directly
- Need to specify the instance and table IDs
- Various types of requests, we'll focus on simple write requests

Firestore

- Fully managed NoSQL database
- Document store
- Can be distributed across regions
- Built-in synchronization and offline mode
- SLA: 99.999% (with multi-region replication)
- Cost effective

Firestore

- Great as a backend for mobile and web apps
- REST API-based backend
  - Client code can directly call the database using REST API
- Synchronize with offline clients
- Multi-region deployment - can serve clients everywhere
- Client libraries for popular languages and platforms

Firestore Modes

- Firestore can be deployed in two modes:

Native

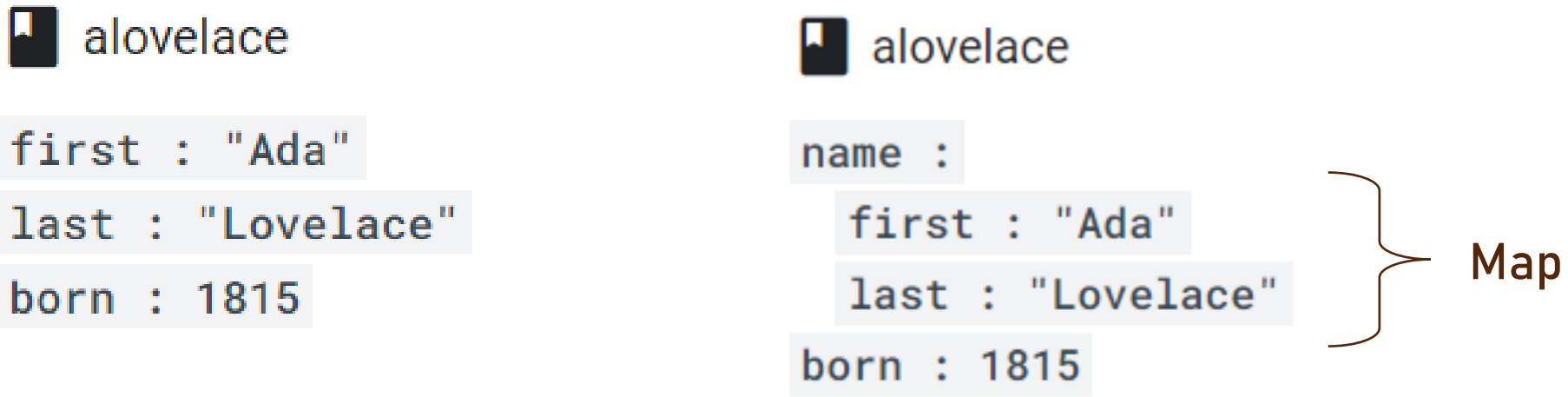
- The new mode
- Document database
- New API
- Offline support
- Rich client libraries

Datastore

- Compatible with legacy Datastore database
- Entity database
- Older API
- No offline support
- Limited client libraries

Firestore Data Model

- Basic data unit is Document
- Contains fields and values

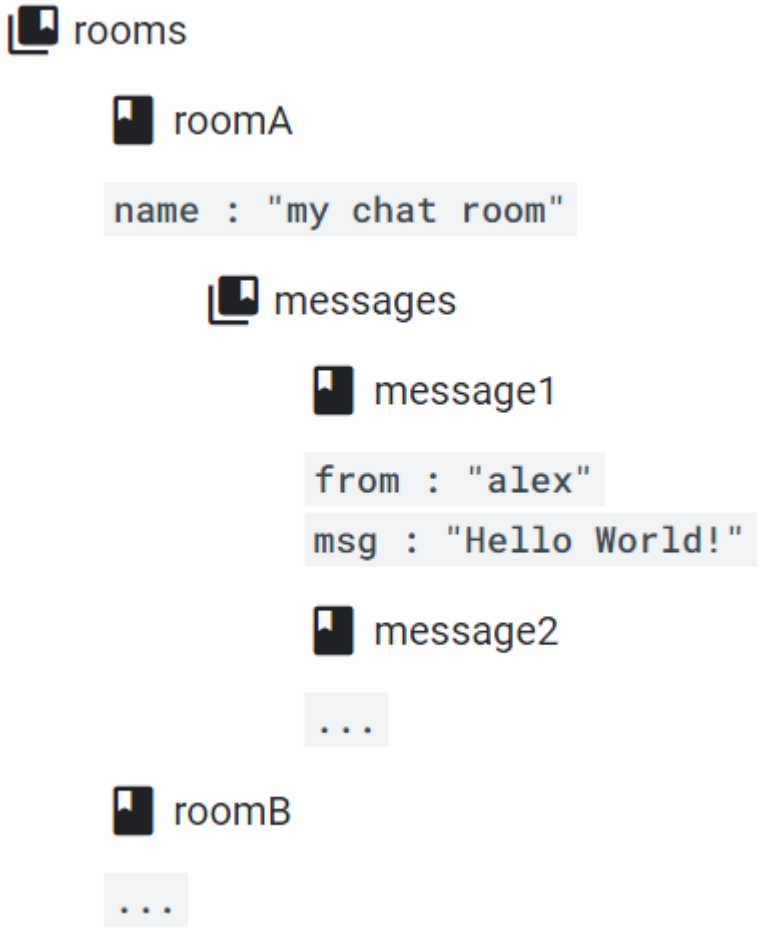


Firestore Data Model

- Documents live in Collection
- A container for documents

Firestore Data Model

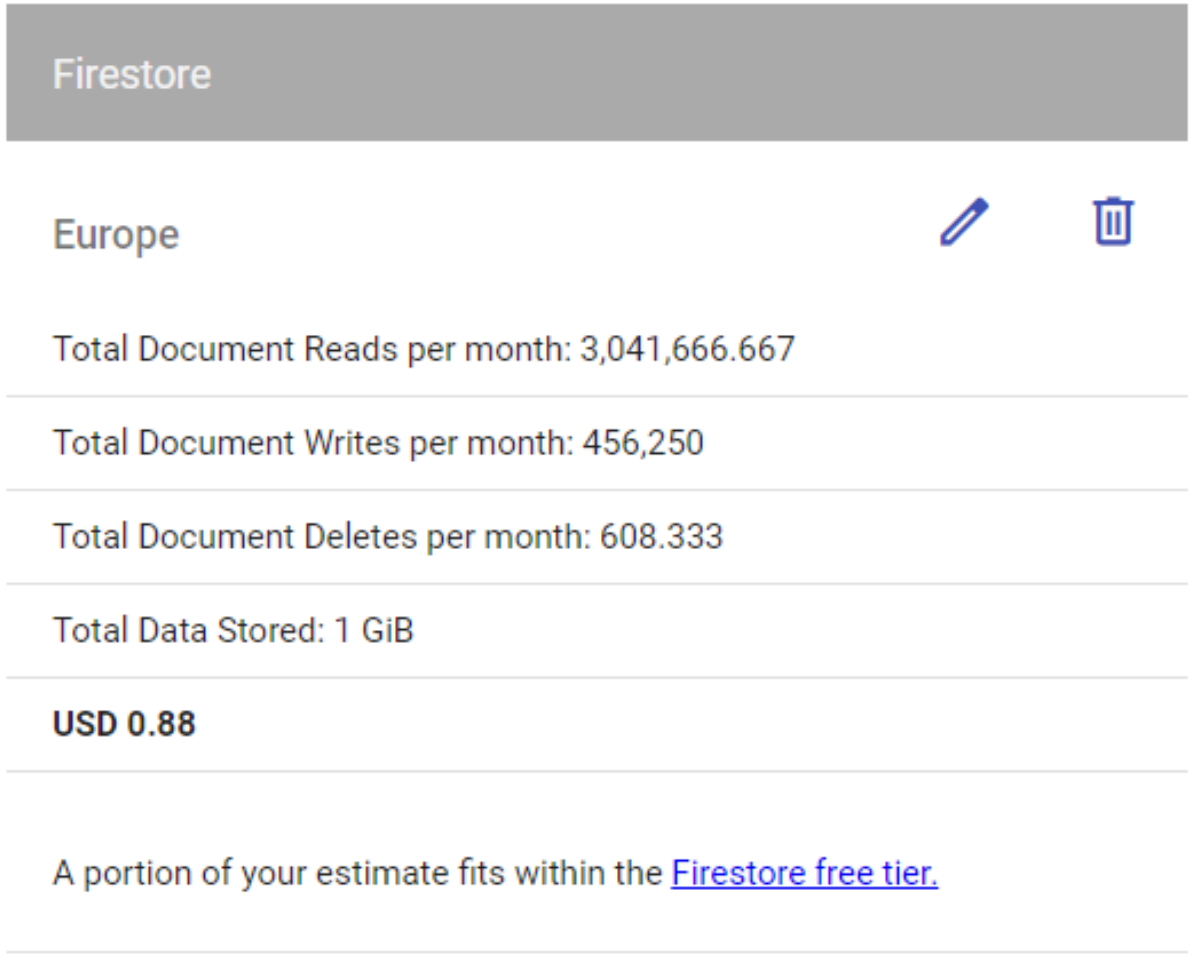
- Documents can have collection in them
- Called Subcollection



Firestore Pricing

- Based on:
  - Storage ($0.18/GB/month)
  - Writes ($0.18/100K/month)
  - Reads ($0.06/100K/month)
  - Deletes ($0.02/100K/month)

Firestore Pricing



BigQuery

- Fully managed Datawarehouse solution
- Has built-in BI and AI capabilities
- Unified experience with the BigQuery Studio
- Can work with all types of data:
  - Structured, semi-structured, unstructured
- Cost effective

BigQuery Editions

- Comes in three editions:

Standard | Enterprise | Enterprise Plus

- Differ mainly in:
  - ML capabilities
  - Performance
  - Price
  - SLA (99.9% in Standard, 99.99% the rest)

Load Data into BigQuery

Batch

Stream

Generate Data

3rd party apps

Batch Loading

- Loading a large amount of data in a single batch operation
- Source can be CSV, external database, set of log files etc.
- Can be done using:

Load jobs

SQL (LOAD DATA)

Data Transfer Service

Storage Write API

Other managed services

Streaming

- Continuously send small batches in real time
- Can be done using:

Dataflow

Datastream

Connector for SAP

Storage Write API

Pub/Sub

Generated Data

- Run SQL statement to generate data in BigQuery
- ie. `CREATE TABLE … AS`

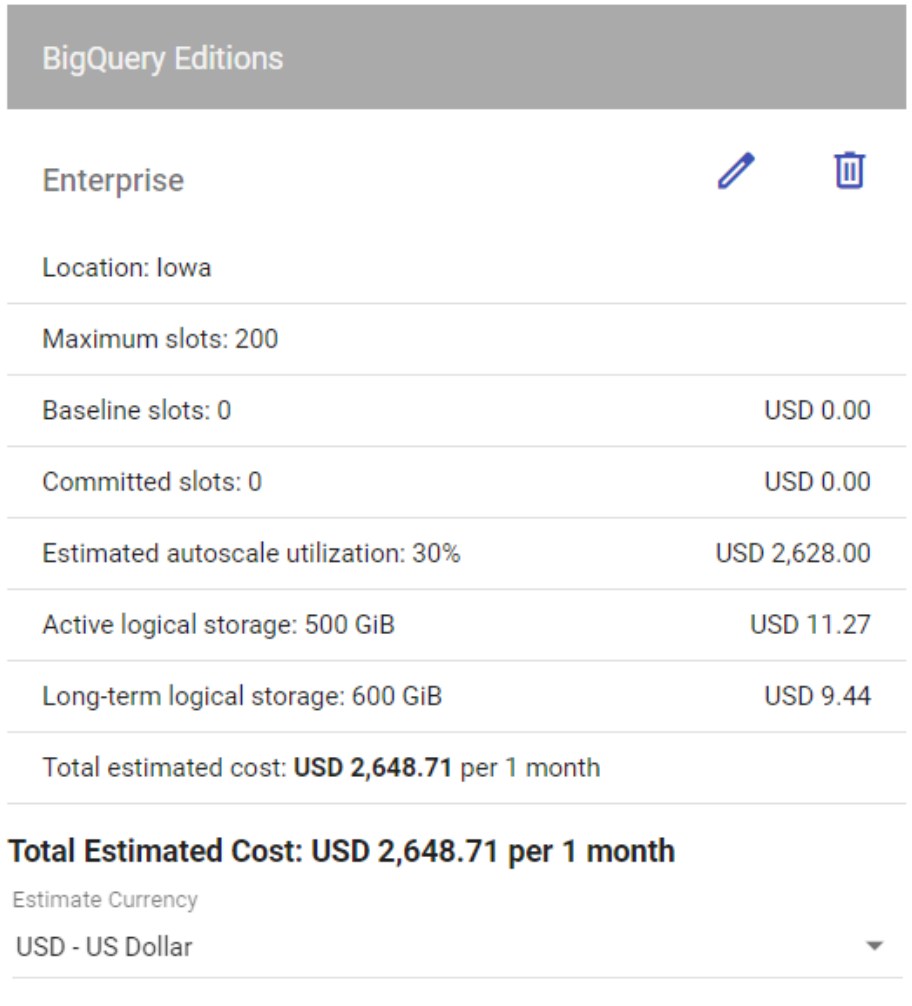
3rd Party Apps

- Load data using connectors from 3rd party apps
- ie. Informatica, Fivetran

BigQuery Pricing

- Based on:
  - Compute
  - Storage
  - Data ingestion
  - Data extraction

BigQuery Pricing



Memorystore

- Fully managed in-memory distributed cache
- Compatible with Redis and Memcached (we'll focus on Redis)
- Regional
- Autoscales to up to 250 nodes
- Microsecond latency
- SLA: 99.99%

Memorystore Redis Tiers

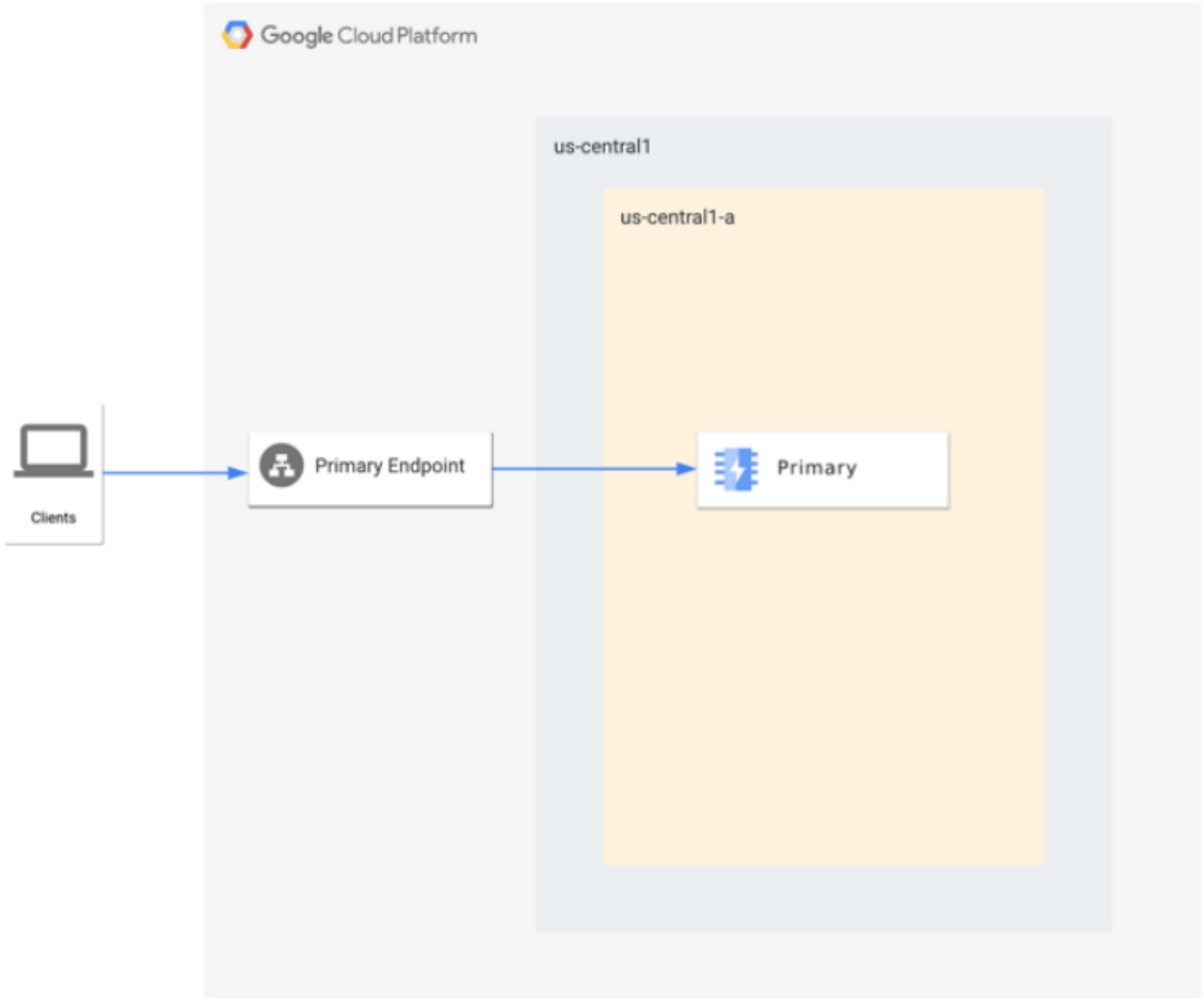
- Comes in three tiers:

Basic

Standard

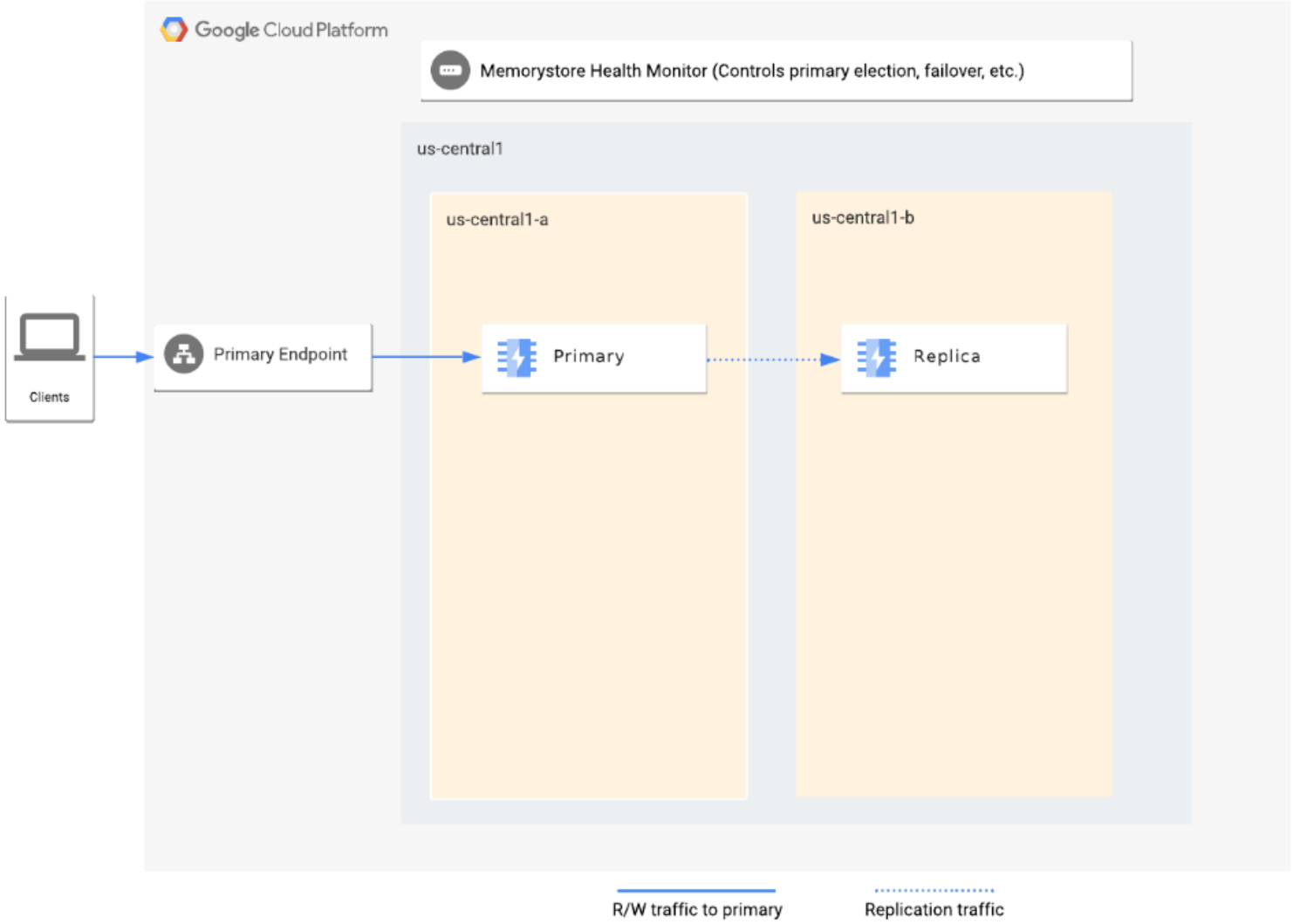
Standard with read replicas

Memorystore Redis Basic Tier



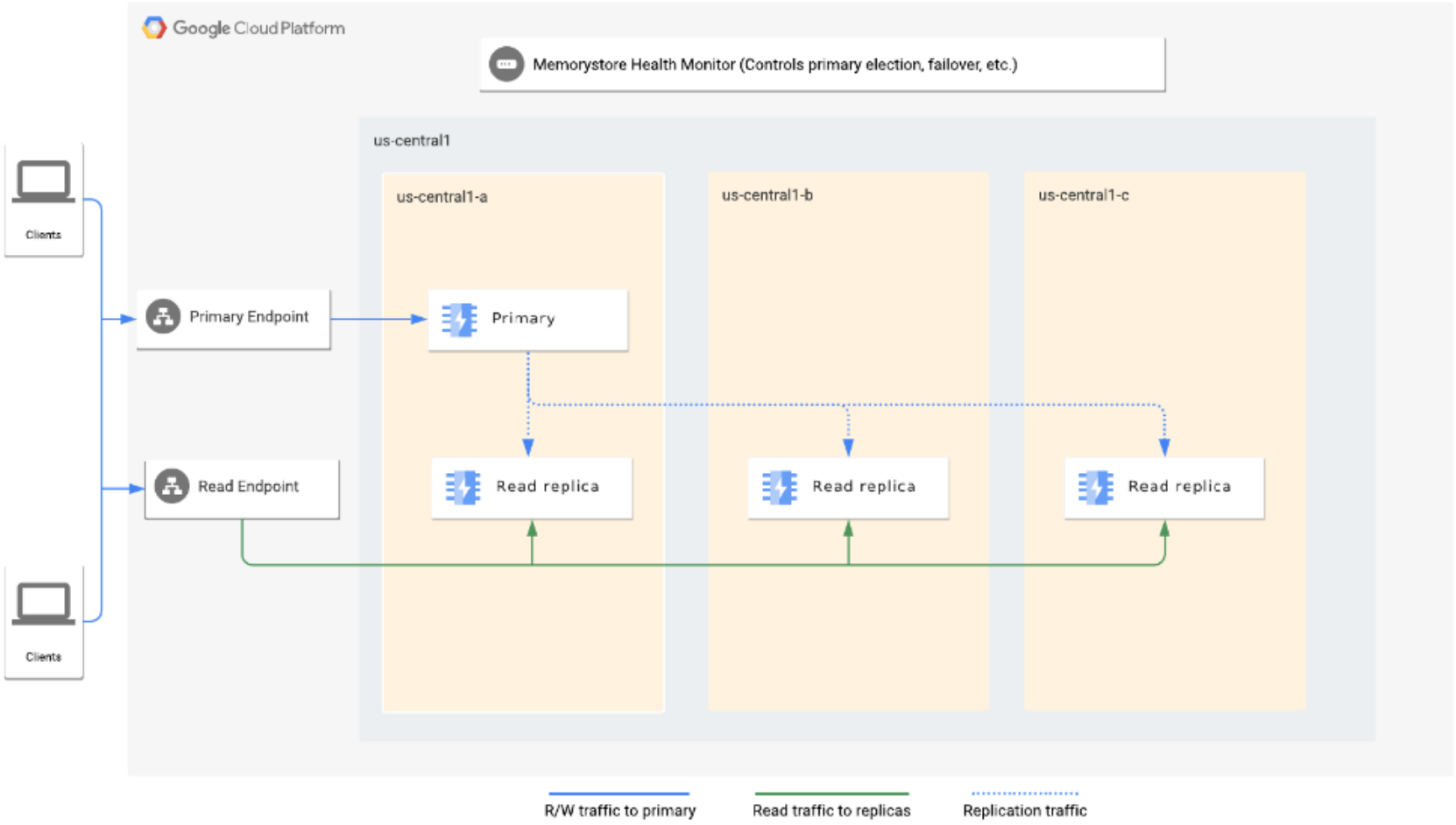
- One node

Memorystore Redis Standard Tier



- High availability with replication
- Fail over to the replica in case of primary node failure

Memorystore Redis Standard Tier



- High availability with replication
- Read replicas provide distributed reads

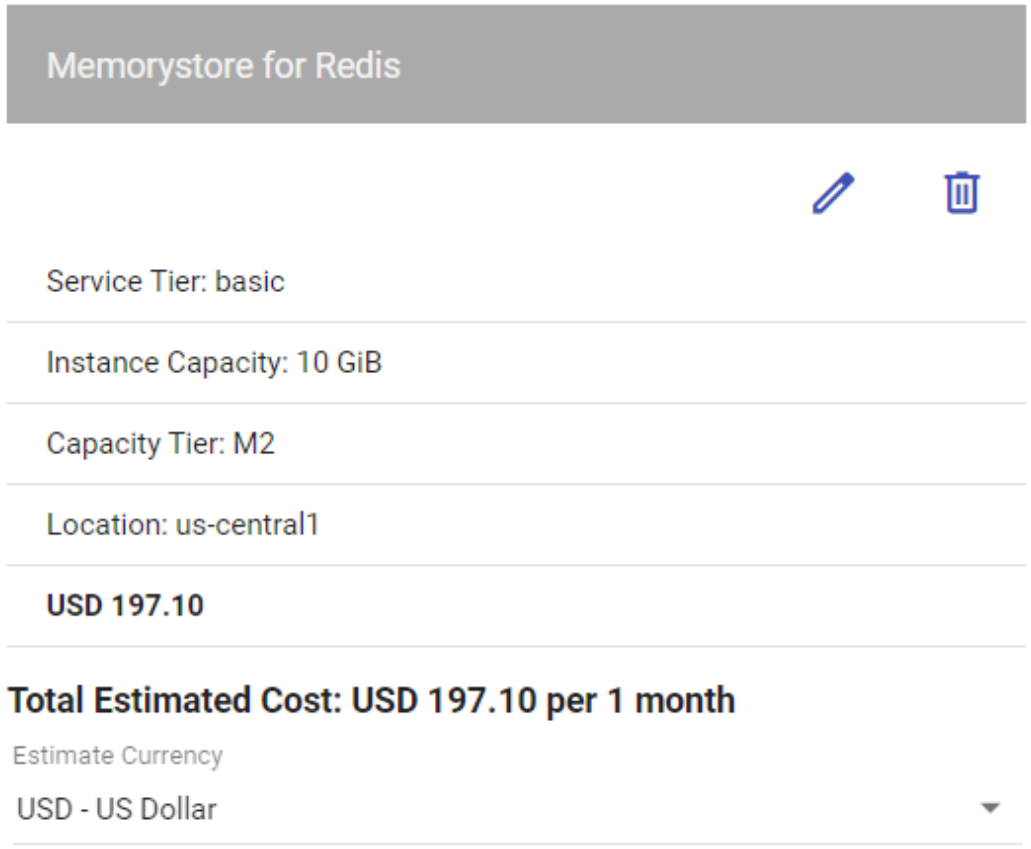
Memorystore Redis

- Max size:
  - 300GB
- Connections:
  - Only using Private IP in a VPC

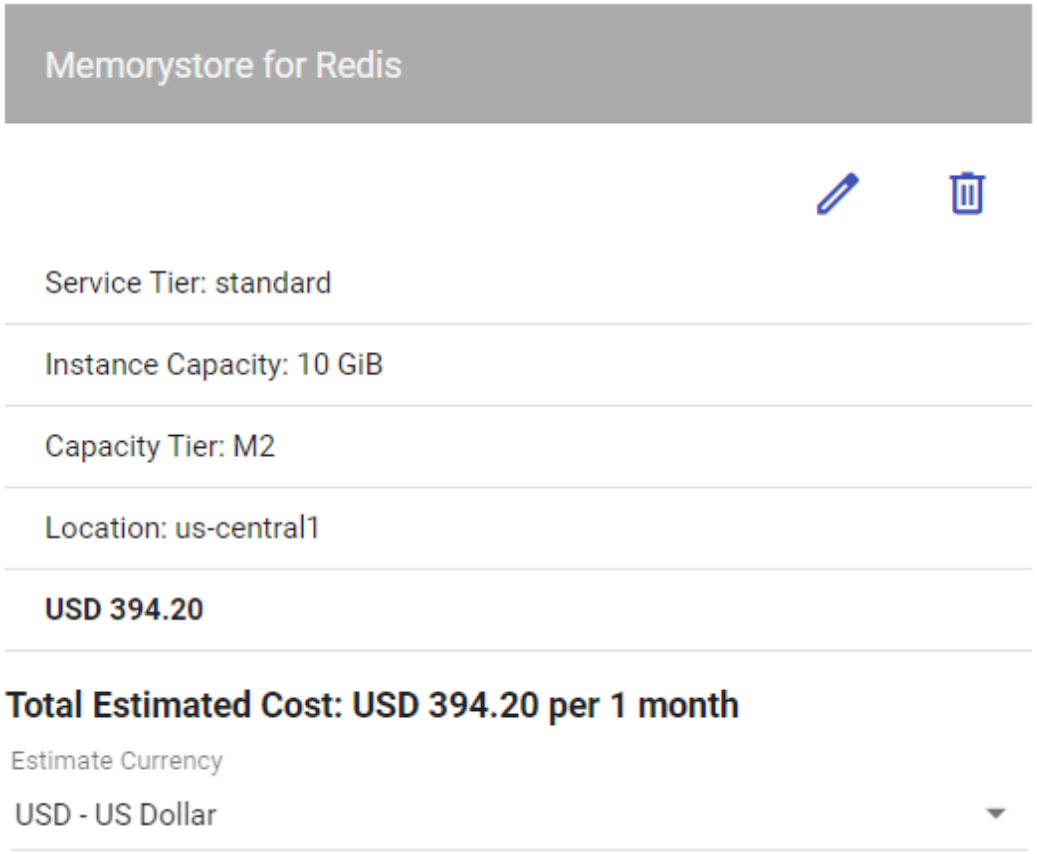
Memorystore Pricing

- Based on:
  - Tier
  - Capacity
  - Region
  - Replicas

Memorystore Pricing



Memorystore Pricing

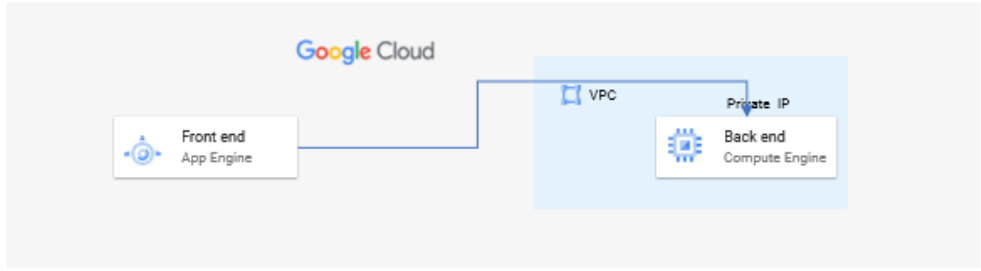


Connecting Cloud Run to Memorystore

- Cloud Run is a serverless resource
- Memorystore is in a VPC
- We need to use Serverless VPC Access to connect Cloud Run to Memorystore

Secure:

1. Traffic stays in the cloud
2. The VM instance is not open to the internet



Cloud Storage

- Fully managed object storage for unstructured data
- Great for storing files, JSON documents, etc.
- Regional or continental
- Accessible using simple-to-use API
- Cost effective
- SLA: Up to 99.95%

Cloud Storage Buckets

- Container that holds the data
- Used to organize data and control access
- No nesting
- No limit to number of buckets in a project
- Name is globally unique
- Created in a location

Cloud Storage Buckets

- Bucket location type:

Regional	Dual-region	Multi-region
• Data is stored in a single region • Redundancy across zones • Best performance • Lowest storage price • Use when need to save cost or have best performance • ie. Backup	• Data is stored in two regions * • Automatic replication • Best performance • Highest storage price • Use when need best performance and cross-region redundancy • ie. Disaster recovery	• Data is stored in multiple regions * • Automatic replication • Limited performance • Price higher than regional, lower than dual region • Use when need highest availability • ie. Content serving

* Failover is done automatically and the URL stays the same, so no action is needed

Cloud Storage Classes

- Set the object's:
  - Availability
  - Pricing
  - Minimum storage duration
- Can be set at the object level or bucket level
- Chosen based on the scenario you need to storage for

Cloud Storage Classes

Storage Class	Minimum storage duration *	Storage cost (GB/month) **	Retrieval fee (GB)	SLA (up to...)
Standard	None	~$0.020	Free	99.95%
Nearline	30 days	~$0.013	$0.01	99.9%
Coldline	90 days	~$0.06	$0.02	99.9%
Archive	365 days	~$0.0025	$0.05	99.95

* If object is removed before the min duration, it'll cost as if it was stored for the min duration

** Exact cost is also based on the region

Cloud Storage Classes

Storage Class	Use for…
Standard	Short lived or frequently accessed data
Nearline	Less accessed data, lives at least 30 days in the storage
Coldline	Data that is accessed once a quarter
Archive	Archive, backup, disaster recovery

Autoclass

- Automatically moves object between classes based on access pattern
- Optional
- Saves storage costs
- Set on the bucket level

Lifecycle Management

- Set of rules that set criteria for automatic behavior of object
- Can be used to:
  - Downgrade the storage class of an object after a specified time
  - Delete object based on various parameters
  - Keep latest versions
- Defined on the bucket level

Cloud Storage Pricing

- Based on:
  - Class
  - Storage
  - Region
  - Operations

Cloud Storage Pricing

Cloud Storage	
1x Standard Storage	
Location: Dallas	
Total Amount of Storage: 500 GiB	USD 10.00
Class A operations: 0.5 million	USD 2.50
Class B operations: 20 million	USD 8.00
Always Free usage included: No	
USD 20.50	

Signed URL

- We saw there are two ways to control access to buckets / objects

Authenticated users only

Public, unlimited access

- Sometimes we need something in between

Signed URL

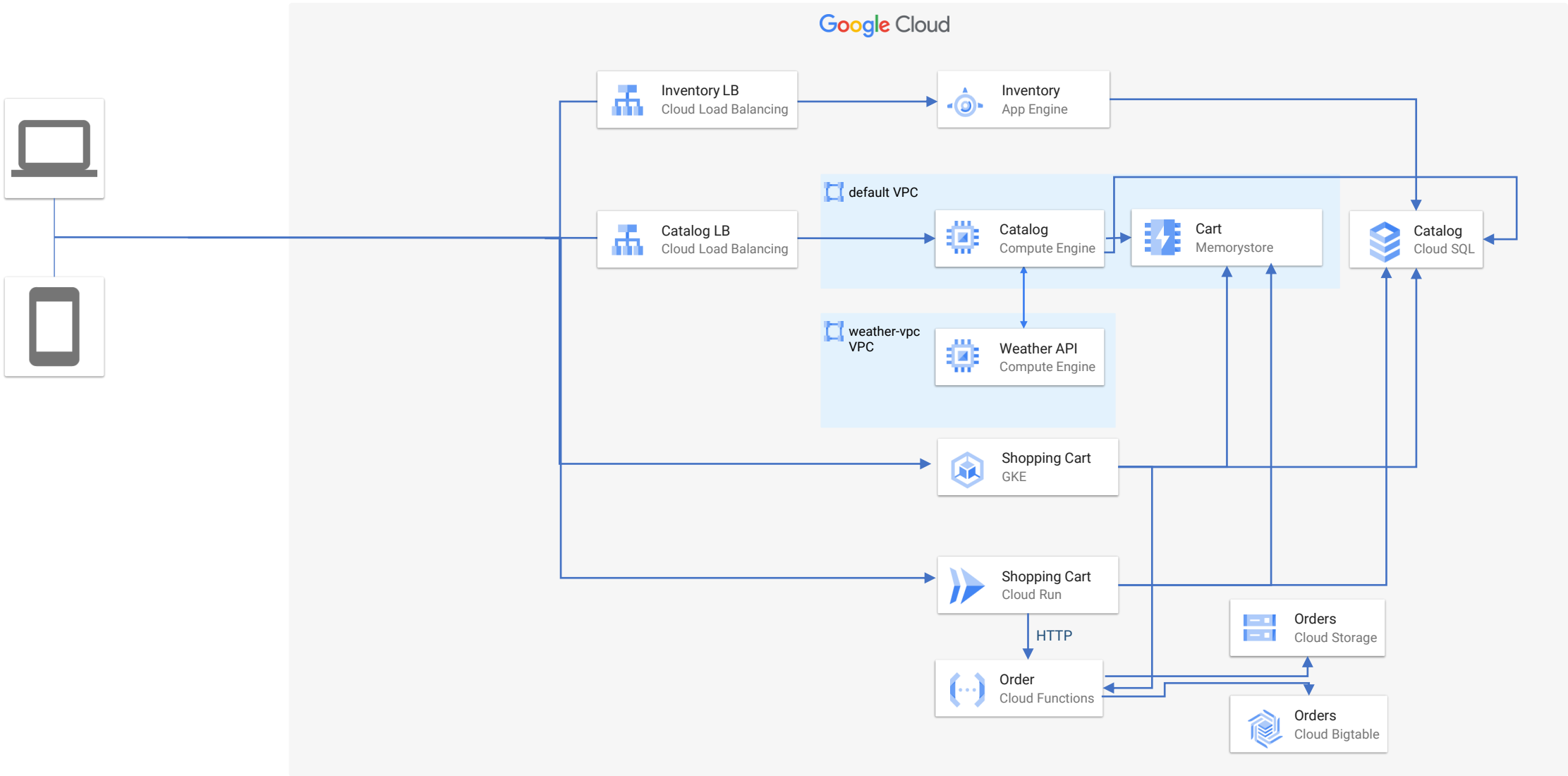
- In many apps we want to grant users:
  - Limited access to an object
  - For a designated length of time
  - Just for a specific object
  - Without having Google account
- For that we can use Signed URL

Signed URL

- A URL signed by the cloud allowing limited access to a specific object
- Can be used by any user, regardless of Google account
- Very useful if you want to allow a specific, unauthenticated user access to a specific object

ReadIt!

Architecture: ReadIt Cloud System



Data Store Services Types

- Cloud SQL
- Spanner
- AlloyDB
- BigTable
- BigQuery
- Firestore
- Cloud Storage
- Memorystore

Data Store Services Types

Service	Data type	Use for…	Cost
Cloud SQL	Relational (SQL)	Relational, transactional data	$
Spanner	Relational (SQL)	Relational, transactional data, requires distribution	$$$
AlloyDB	Relational (SQL)	Relational, transactional data, high performance, PostgreSQL compatibility	$$
BigTable	NoSQL, columnar	NoSQL data with schema, high volume	$$$
BigQuery	NoSQL	Analytics	$$
Firestore	NoSQL, document	Backend for mobile and web apps, offline and sync support required	$
Memorystore	NoSQL	Fast, distributed cache	$$
Cloud Storage	NoSQL, unstructured	Unstructured data such as files, docs etc.	$