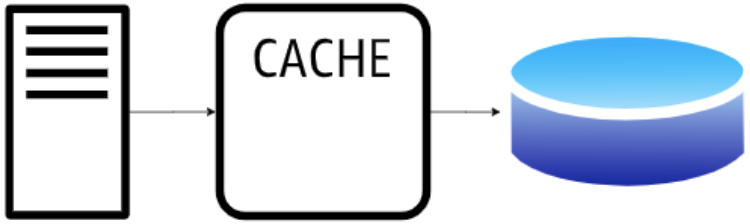
# Caching in Google Cloud Platform

# Caching

- How can **reduce the load** on
  - Your data stores
  - Your servers
- Use Caching
- **Questions to ask** when Caching?
  - **How often** does the data change?
    - Caching is amazing if the data does not change frequently!
  - Am I OK with some **stale data**?
    - What should be **TTL** (Time to Live)?
- Caching Usecases:
  - Caching **infrequently changing data** in Database
  - Caching **user sessions** from applications
  - Caching static content
  - Caching infrequently changing dynamic content



# Memorystore



Memorystore

- **In-memory datastore service**: Reduce access times
- **Fully managed** (Provisioning, Replication, Failover & Patching)
  - Highly available with 99.9% availability SLA
  - Monitoring can be easily setup using Cloud Monitoring
- Support for **Redis and Memcached**:
  - Use Memcached for Caching
    - Reference data, database query caching, session store etc
  - Use Redis for low latency access with persistence and high availability
    - Gaming Leader Boards etc
- **Can be accessed from:**
  - Compute Engine
  - App Engine flexible and standard
  - Google Kubernetes Engine
  - Cloud Functions

# App Engine memcache service

In28 Minutes

```
def get_data():
 data = memcache.get('key')
 if data is not None:

  return data
 else:

  data = query_for_data()

  memcache.add('key', data, 60)
 return data
```

- **Legacy in-memory data cache** specifically for **App Engine applications**
  - **Example Use Cases**:
    - Speed up common datastore queries
    - Caching session data and user preferences
  - Temporary storage (not backed by persistent storage)
  - **Two Service levels**:
    - **Shared memcache** (FREE): best-effort caching
    - **Dedicated memcache** ($$$$): Fixed cache capacity dedicated to your app

# Cloud CDN - Content Delivery Network




- Use **Google's global edge network** to serve global content with low latency
- Integrates with **External HTTP(S) Load Balancing**
  - LB provides frontend IP addresses and ports
- **Backends can be:**
  - Cloud Storage buckets, Instance groups, App Engine, Cloud Run, or Cloud Functions
  - Endpoints outside of Google Cloud (custom origins)
- **How Cloud CDN works?**
  - External HTTP(S) Load Balancing uses proxies - Google Front Ends (GFEs)
    - Request from user arrives at a Google Front End (GFE)
    - If URL maps to a backend with Cloud CDN configured:
      - If content is found in cache(cache hit), GFE sends cached response
      - If content is NOT found in the cache (cache miss), request is forwarded to backend (origin server)
        - Response is sent to user and cached
    - Using TTL settings to control cache duration

# Cloud CDN - Best Practices




- **Cache static content**
  - Example: Cache-Control: public, max-age=259200 (72 hours)
- Be careful with expiring **time-sensitive (or dynamic) content**
  - Smaller cache periods. Example: Cache-Control: public, max-age=300 (5 minutes)
- Using **custom cache keys** to improve cache hit ratio
  - **Default cache key** - Entire URI - ***https://yourwebsite.com/my-image/1.jpg***.
    - cache miss - ***http://yourwebsite.com/my-image/1.jpg*** (http vs https)
    - cache miss - ***http://yourwebsite.com/my-image/1.jpg?mobile=1*** (query string does not match)
  - **Customize cache key**: Any combination of protocol, host, or query string
    - `gcloud compute backend-services update BACKEND_SERVICE --enable-cdn --no-cache-key-include-protocol --no-cache-key-include-host --no-cache-key-include-query-string`
- Using **Versioned URLs** to update content
  - ***https://yourwebsite.com/my-image/1.jpg?v=1***
  - ***https://yourwebsite.com/my-image/1.jpg?v=2***