This is the note that I took for watching “Google System Design Interview with Ex Google Interview”

Prompt. Design Global and Fast Code Deployment System.

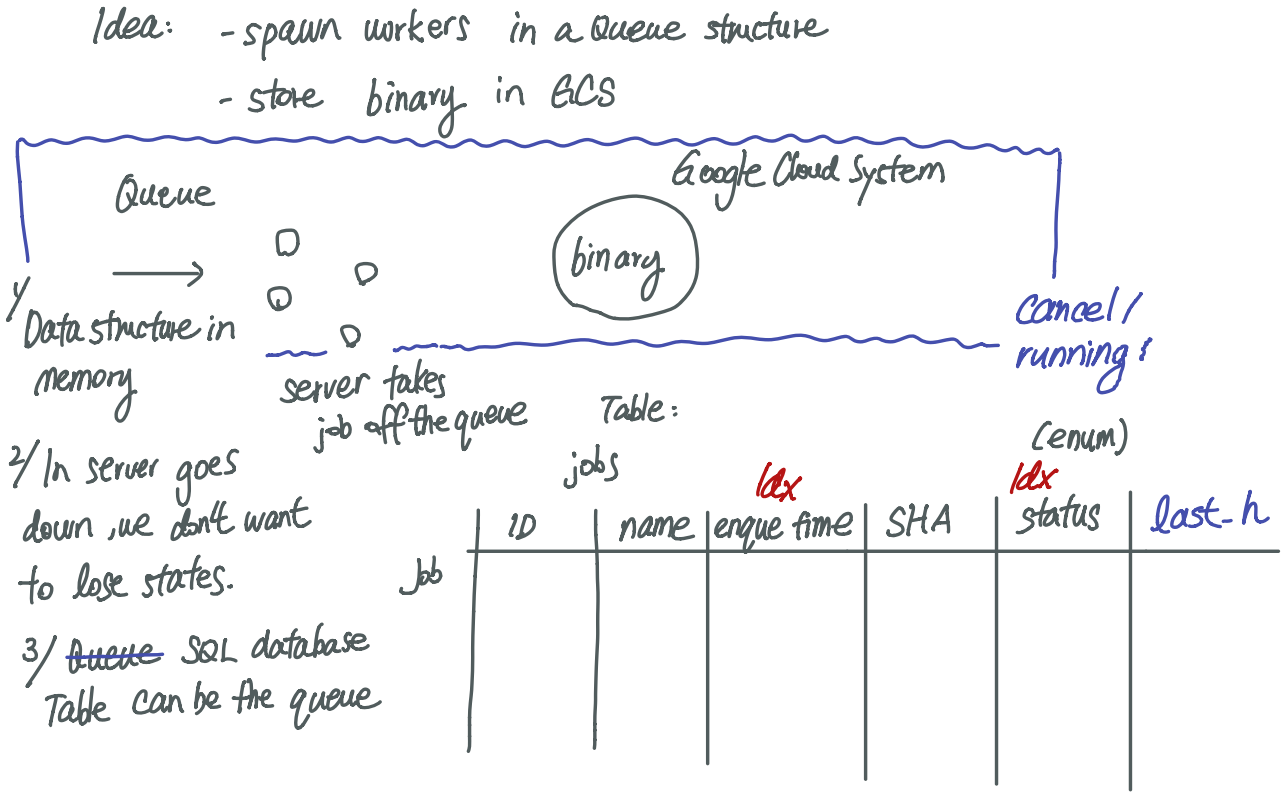
Question: What part of the code deployment system are designing?
Answer: Building and deployment.

Basically we are shipping code that is presumably shippable.

1) Building Code
2) Deployment
3) 2-3 nines of availability
4) 30 minutes
- 5-10 Regions globally
- 100+ machines
- Availability? What’s the expectation? Has to be available? The build has to reach a final state within 30 minutes. Triple 9s, half a day in a year; or are we shooting for four to five nines.
- 10 GBs of system design
- 1000 deploys per day

Solution
1. **Building Code**
2. **Deploy the code**

Communication Question: Is that a sensible way to come up with the system?



Concurrency safe: ? Since we have a 100 workers that are looking to put in database.

```
BEGIN TRANSACTION
SELECT * FROM jobs
WHERE status = "QUEUED" AND
ORDER BY created_at ASC
LIMIT 1.;
UPDATE jobs SET status = "RUNNING" where id = our.id

COMMIT

END TRANSACTION;
```

5000 / 24×4 = 96 ≈ 100 builds per work

5000

peak/slow

[?]

ROLLBACK if no id

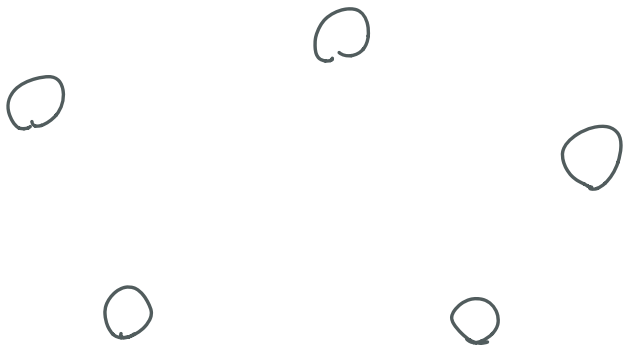
Check for heartbeat

Power shutdown?

Health check.

Question:
If we have hundreds of thousands machine across the globe, is one GCS enough?

Answer:
We can have regional clusters like regional sub system that handles the part of deployment.
For instance, we can regional GCS bucket. e.g. replicate all the region that have blobs and buckets,



Concepts that needs to review:

1. Distributed System Concepts:
    A. How to avoid race condition? (Multiple worker accessing database)
2. How to handle power outage?
3. Load-balancing
    A. You can put a load balancer to distribute traffic. The load balancer will route the request to a various of machine. (E.g. nginx, DNL load balancing, given a URL, we can resolve this URL to multiple IP address). Round robin, figuring out which one to use
4. **Caching:**
    A. cache the result of the request. Can last for 24 hours. Reddis, Cassandra. We can cache based on the location of the service.
5. Database schema design
6. **Slave-Master replications**:
    A. You have a master database
    B. It's cloned into a replicate database.
7. Database Shardin:
    A. we shard the database into smaller databases
    B. One common way to shard: horizontal shard to reduce the table size
    C. Master table will be responsible for sharping
8. No SQL: not relational, they are key value pairs
    A. Mongo DB
9. API design