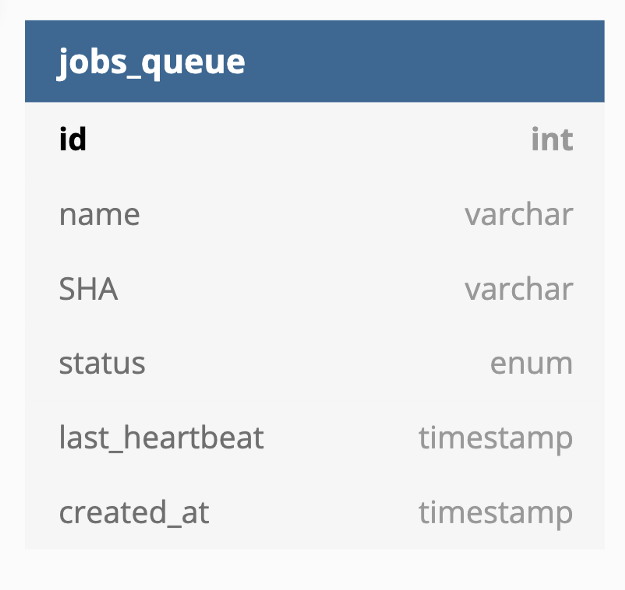
**Q.1. Describe a high level design for a Distributed Code Deployment System.**

## Part 1: Building

Since we want to persist the history of jobs. We can think of a SQL table for storing the job entries (this can represent our queue).  
We define our status enum as { QUEUED, RUNNING, FAILED, COMPLETED, CANCELLED}.

How to work this as a queue ?  
We can select the jobs where status = QUEUED and has the oldest created\_at timestamp.  
Hence the index on status and created\_at will improve performance.



BEGIN:  
 SELECT \* from jobs\_queue where status = "QUEUED"   
 ORDER BY created\_at ASC LIMIT 1  
 if(id == nil) ROLLBACK;  
 UPDATE jobs\_queue SET status = "RUNNING" WHERE id = x;  
COMMIT;

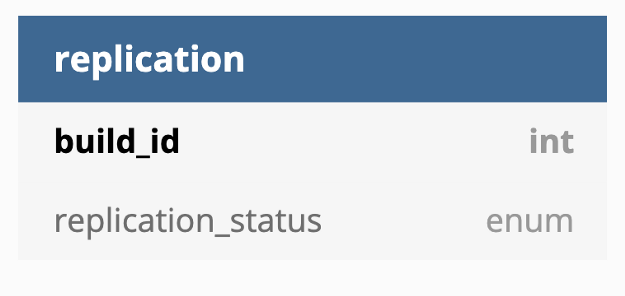
**What if our worker crashes while building code ?**

* We can tackle it by adding a column last\_heartbeat in our jobs\_queue table. Whenever a worker picks up a job and is running it. It will continuously send heartbeat signals to the job table. Let’s say a build takes around 15 mins, the workers will send this heartbeat every 3 mins.
* Now we introduce a new monitoring service that will monitor the jobs\_queue table for all the “RUNNING” jobs. If we find a job’s heartbeat not updated for 2\*heartbeat duration i.e. 6 mins in our case, then we rollback that job’s status to “QUEUED”.

**What if we want the developers to be able to deploy the build only if the binary is replicated to all the regions ?**

To tackle this requirement we can have a really simple service which polls the master blob store for any new binary and tracks the status of that binary by polling to all the regional blob stores.

The build is only deployable if the replication\_status in the table is “COMPLETED”.



## Part 2: Deploying

We assumed earlier that the build takes 15 mins and let’s say replication takes another 5 mins. So for us to meet the 30 min deadline for entire deployment, we are left with 10 mins. For 100K machines to download a 10 GB file from the blob store over network seems unreasonable. Hence we can create a Peer-to-Peer network. All the machines in a region are part of a Peer to Peer network, this will enable them to download multiple such binaries really fast.

**What exactly happens when a developer presses the “Deploy” build “Build\_1” ?**

We can take advantage of our clustered architecture to introduce a Key-value store (like zoo-keeper, consul) at every regional cluster. And a master Key-value store which is actually updated when the developers presses the button. The idea is to have a config like -

{  
 build\_number: "build\_0"  
}

Whenever the build number in the KVS is updated the machines downloads the new binary and deploys it.  
The regional KVS polls the master KVS for any change in the build\_number and whenever the application servers are in steady state they poll their regional KVS for any change in the build\_number.