**Explanation for Question 1:**

**Pitfalls in provided setup**

* Need cache server to reduce network bandwidth and better app response
* I don’t see **Firewall** here, it’s not advisable to expose our system directly over net.
* User **cannot use API via LB** as API server is not LB which means that we may have to provide API server link to API consumers which may expose our API servers and we may face security and performance issues on our API server.
* In current setup no DB replication/clustering is available, we should have DB clustering, master-master replication in place for business continuity plan.
* I would recommend to setup a DB master-slave setup, where we can configure all **read request** would be serve by slave node only, this will reduce additional load from the master DB server.

**Common monitoring parameters**

* **On server resources:**
  + CPU usage
  + RAM usage
  + Disk space
  + Zombie and high memory consuming process\*
* ***On Applications front:***
  + JAVA heap memory
  + API performance
  + Web-app performance
  + DB connectivity
  + Logging requesters/consumers userid and node name where his request was served
  + Failed requests logging and monitoring for API and web app
* ***Networking:***
  + Network traffic (loss of packets.. generally it is handled by network team)
* ***Server Certificates:***
  + Tracking certificate expiry

**Immediate action when some problem with DB**

* Check if DB server is up and running
* If DB is up, then check if we see for current active connections on DB.
* Check DB connectivity from local machine as well as from the server.
* If DB is down, involve DBA ASAP.

**Questions for system admin**

* What is current server topology?
* How are we managing multi geographical sites, what are the implementations in place for better performance?
* Do we have any Infrastructure and application monitoring solutions in place?
* Self-healing systems in place?
* Do we have Staging, UAT, Pre-Prod environments in place? If yes then how are we managing to keep these environments as much as close to production?
* User management
* How are we handling product releases, Fixpacks, CSD?
* What are the measures in place for Performance, data integrity and data security?
* How are we managing server certificates?
* How are we managing server patching?

**Explanation for Question 2:**

**Common Problems during migration**

* *Downtime required for Migration*.

**Mitigation action:** We shouldbreak down this activity based on below pointers

* + - **DB migration**: We can add newly created Cloud instance in our DB cluster which will make sure that the newly added Cloud node is in sync with all the other nodes and this can be easily done without downtime
    - **Web-App and API:** I would prefer to containerize them, which will rule out the environmental related Issues and migration would be seamless
* *Storage migration:*

Data migration to cloud storage can be done at run time during quite period via automated scripts/jobs.

* *Network and security* 
  + Configuration of Cloud firewall.
  + Setting up VPN.
  + Configurations of DNS entries same as existing.
  + **I am not an expert in this area** but this is one of the important aspect to be implemented and tested before migration itself to avoid surprises.

**Change of Architecture for cloud hosting**

* Implement cloud load balance for better performance.
* Would create containers for my App and API and will host App and API in same container
* Host multiple containers on same host, which will share requests on each host.
* Add Web-app and API nodes directly under Cloud LB.
* **Read requests** would be only via DB slave to reduce load on DB

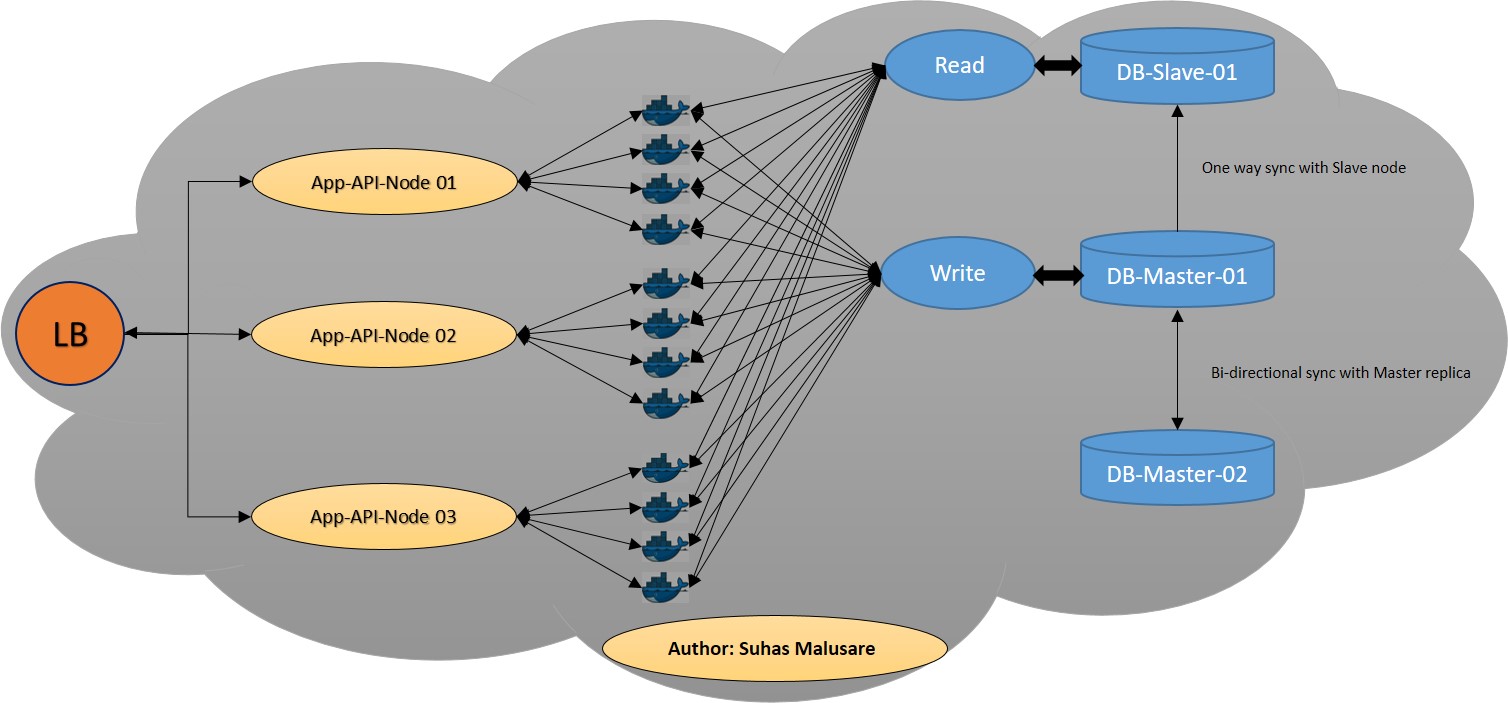


Fig.1: Proposed architecture (based on my current non-cloud implementation)

**Explanation for Question 3:**

I have not worked on any automated architecture provisioning but developed an [application](https://github.com/SAAMalusare/GoGCE) written in Go-Lang to create VM on Google cloud compute, it also returns the health-check and status of an existing cloud machines.