Wk-10

L1 Deployment Envt. & Strategies

- Envt. → sys. that execute a software applicⁿ.
  - types → dev., testing, staging & producⁿ
- dev. Envt. → local env. of a SDE. Contains IDE & other tools. dev. Code base (working copy) with local DB commits to shared code base (mainline / master)
- Staging envt. → exactly resembles the producⁿ envt., runs on a remote machine.
  (→ deploying has many activities & config., preserve new features, performance testing
- Blue / Green Dep. → staged deployment, create a new sep. producⁿ envt. for the new version w/o affecting the current one. Pros → rollback is easy; eg. cycling btw. real & peer. versions.
- Canary Dep. → phased / incremental rollout, slowly rollout to small subset of users. Cons → manage multiple instances at once.
- Versioned Dep. → allow users to choose versions, keep all versions alive, if user updates route them to new version. Cons → maintain many version

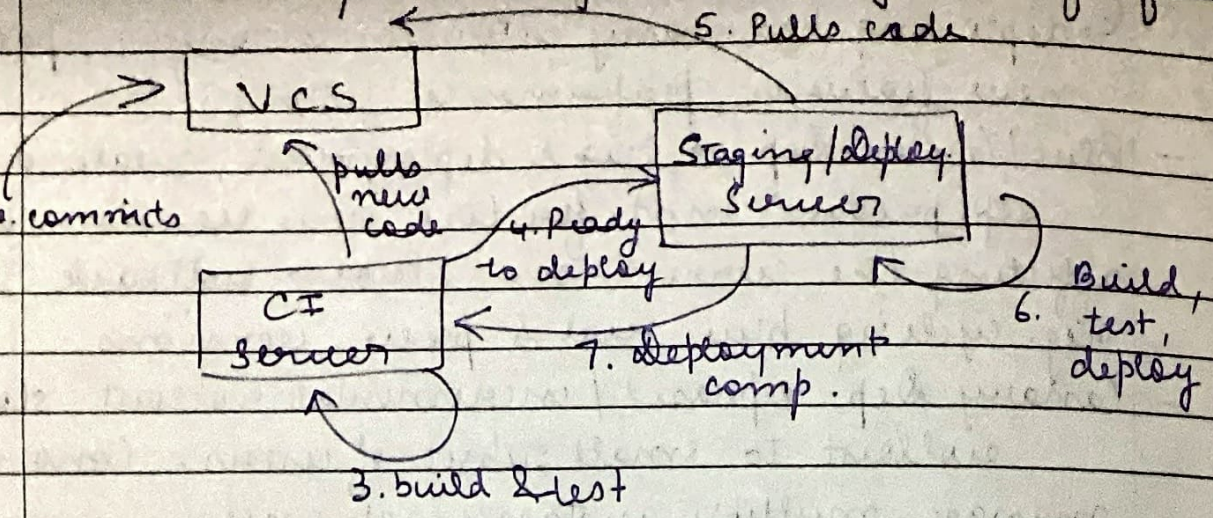
L2 Deployment Hosting

- Hosting – Infrastruc. req. to host applicⁿˢ, usually hosted on server sys., no unplanned downtime.
- Bare Metal Server → purchase actual server hardware, Pros → highest performance, Cons → most expensive, time & effort needed.
- IAAS → (Hardware + OS → IAAS) Eg. digitalocean, AWS
  Pros → Cheaper, no maintenance overhead
  Cons → own set of config; shared by others, performance ↓

- PAAS → (IAAS + Web server) Eg. Heroku, google engine
  Pros → very easy to deploy
  Cons → lack of control.

L5 Conti. Integra^n

↳ automating the integra^n & deployment of software



- Best practices → maintain a single source repo, automate build (Ant, Gradle, Builder), make build self-testing, commit to main branch regularly, every commit should build the main branch on an CI server. Eg. Jenkins, Cruise Control, fix broken builds immediately, automate deployment.
- benefits → reduces deployment time, avoid last-minute confusion, beneficial to users & developers.
- cons → initial effort req. to setup processes.

L6 Performance & Monitoring

- Improves performance of web app →
1. Caching → store results of common opera^n in

memory, fetch from memory. Eg: Redis, Memcached. Different levels → web browser, web server, DB (results of recent query).

2. Task Queues → app. have resource & time-consuming tasks, use asyn. work queue (executed outside the HTTP req.-response cycle)

3. Load & Render Pgs → quick loading is imp., it affects busines of your prod., minify code (digital Ocean), compress static pages (Brotli, Gzip)

- Monitoring — issues can still arise in the live env.
  ↓ tools

1. Generate reports — Analyze page resources, check SEO & accesibility metrics, eg. Lighthouse.
2. Understand Customer behaviour
   → Clickstreams → which seq. of pages do your users go.
   → Think/Dwell time ⇒ user stays on your page.
   → Abandonment
3. Analytics tools → embed JS in your site.
4. For A/B-testing — Version A + B ⇒ use canary deployments, some users get A and some B. Now check which is better.