#ASLI ENGINEERING

Canary Deployments

SWIPE

BY

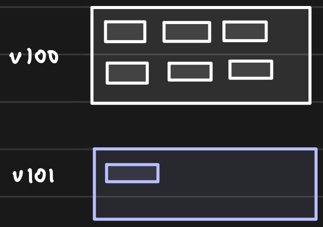
ARPIT BHAYANI

Canary Deployments

Canary deployment is a pattern that allows us to rollout code, features, changes to an initial subset of users/servers before we take it to 100%..

roll out to canary

- if all good, roll out to 100% of servers
- if something is wrong, rollback



Canary deployment thus acts as an Early Warning Indicator so that we could avoid a massive outage

The flow would be:

1. deploy the new version to one server
2. monitor the vitals: CPU, RAM, error rates
3. test the changes expicitly on canary
4. if all good, rollout to remaing servers
5. if something is wrong, rollback

Why the name "canary deployment"?

In 1920s during coal mining, the miners used to carry caged canaries into the coal mines.

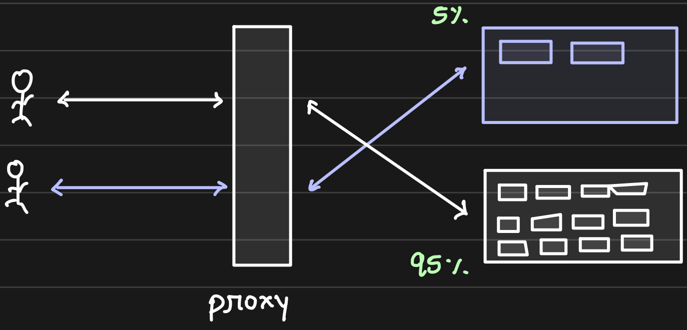
if there was a high level of toxic gas, the canary would die, alerting the miners to evacuate.

In our canary deployment the few servers/users we roll out to our canaries for us that

- provides early warnings if something goes wrong

How canary deployments are implemented?

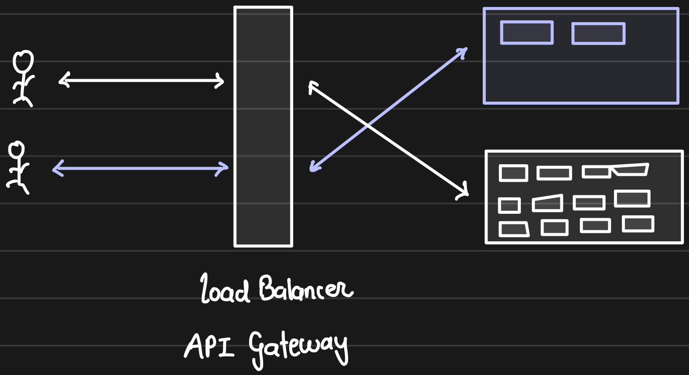
We create a small parallel infrastructure and put a proxy like load Balancer or API Gateway in front



ARPIT BHAYANI

Pros of doing a canary deployment

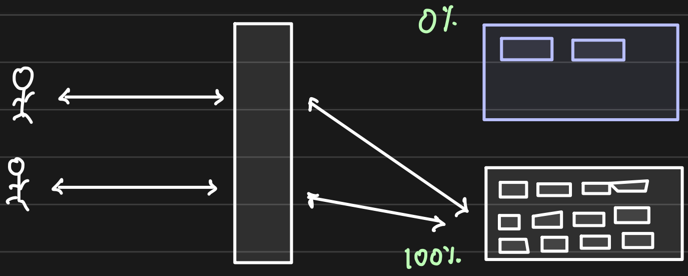
- allows us to test the changes in production with real users



The fleet can be logically separate or physically separate

- rollbacks are much faster

Because the new version of code is deployed on only a few machines, rolling back our changes requires us to rollback only those few machines



- minimal blast radius

  if the new version of code has "bugs"
  with canary deployment it would affect a
  very limited set of requests

- zero downtime deployments

  incremental rollout : 1%, 5%, 10%, 20%, 50%, 100%

- we could deploy even when we are unsure about the new release

- We can use canary setup to do A/B Testing

  eg: Search V1 and search V2

  monitor CTR side by side

Note: Selection of servers / users can be more sophisticated

- geographical
- user cohorts
- random selection
- beta users

- sticky + random selection
- internal employees

Selection criteria is usecase specific

Cons of doing a canary deployment

- engineers will be habituated to test in production

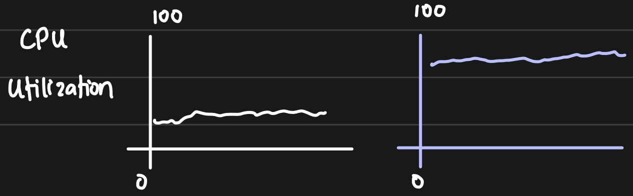
  Testing should never happen in production, but if we have a canary deployment we are encouraging engineers to test in the production environment, and slowly we may find most testing happening in production

- architecting a canary deployment is complex

  Extra infra components: LB, API Gateway, separate scaling policy, instance/container launch config etc.

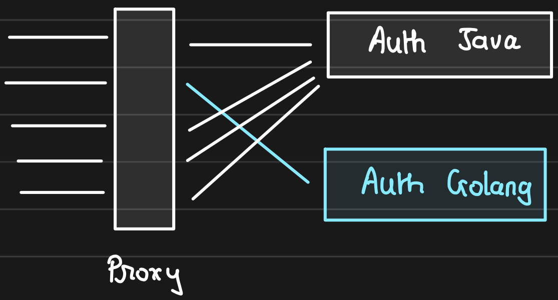
- parallel monitoring setup

  Observability is super important in canary setup
  Side by side comparison of vitals - CPU, RAM, response times, exceptions, etc.



You absolutely need canary ↓

When you re-wrote your service



You would not do a 0 to 1 swith from one service to another.

This has to be done incrementally

100% of traffic will be moved to new Auth Service written in Golang only after you have 100% confidence about the correctness of implementation

ARPIT BHAYANI