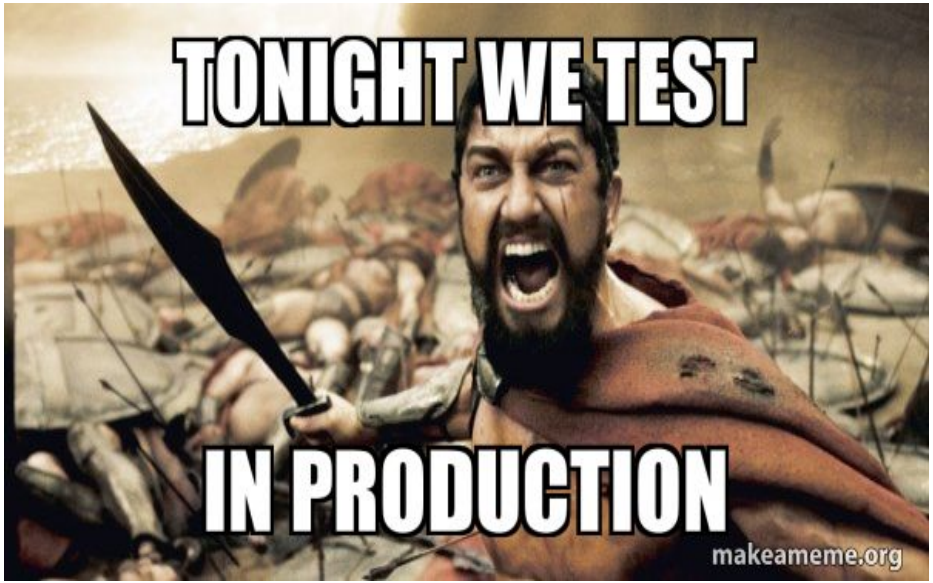
Testing in Production

Diego Pacheco



About me...

- Cat's Father
- Head of Software Architect
- Agile Coach
- SOA/Microservices Expert
- DevOps Practitioner
- Speaker
- Author

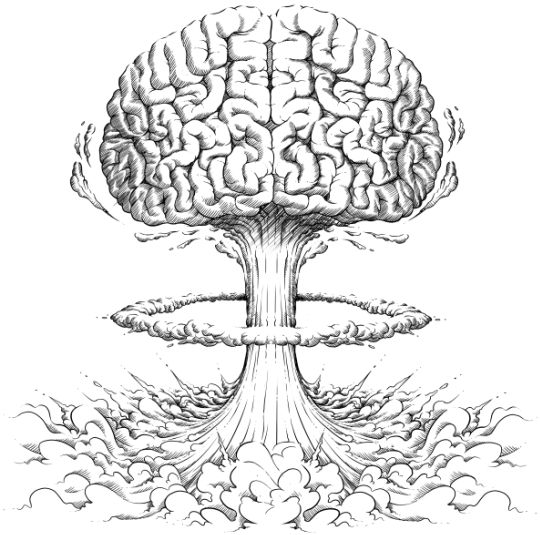
diegopacheco

@diego_pacheco

http://diego-pacheco.blogspot.com.br/

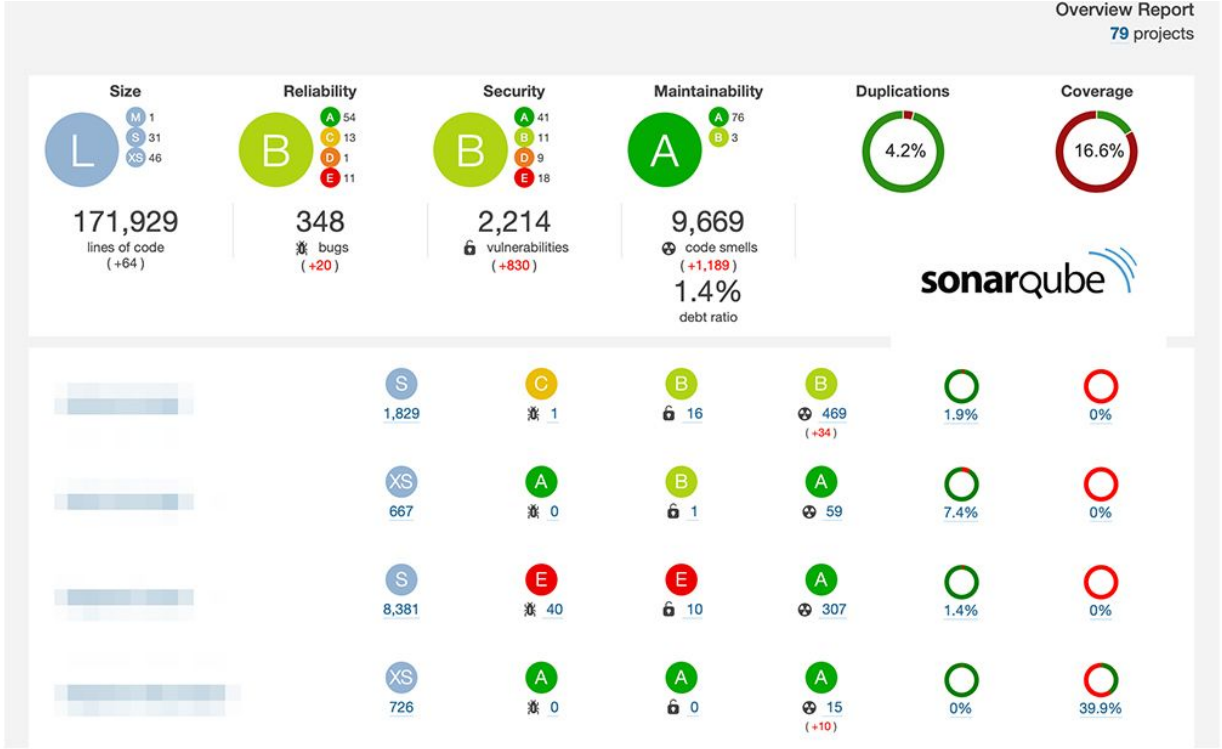
https://diegopacheco.github.io/

Did you lost your mind?



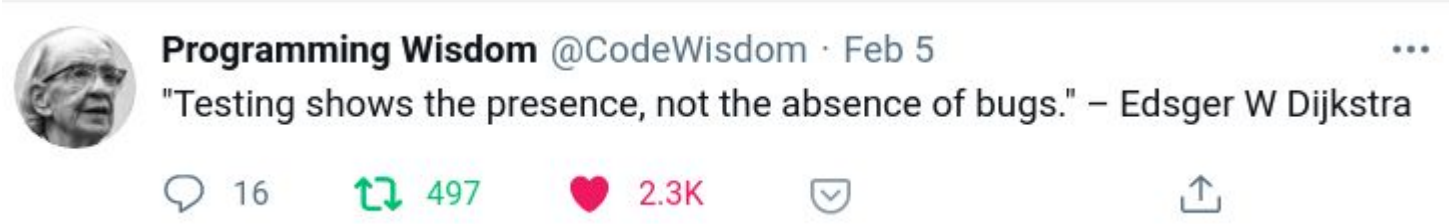
The issues with pre-production testing - Prediction

100% test coverage is a flawed idea



100% test coverage is a flawed idea

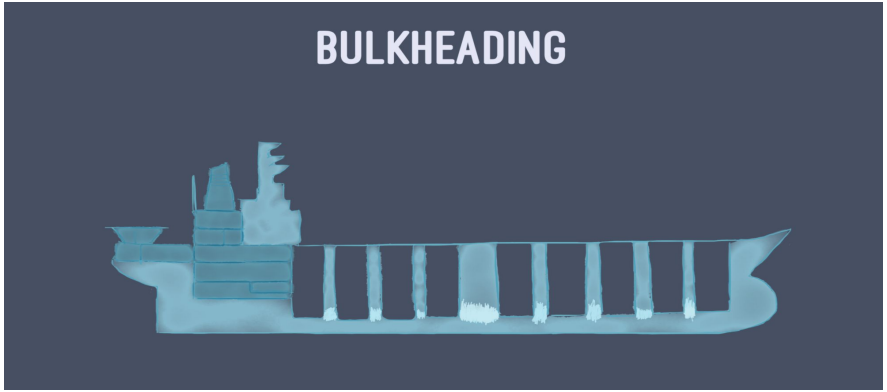
- ❏ Coverage tools just capture (unit and sometimes integration tests) what about the rest?
- ❏ Unit Tests have cost(Mostly Mocks)
- ❏ There is waste in tests. Numbers != Quality.
- ❏ There is no clear and generic/global number about distribution of tests (%)
- ❏ People want signals but there is no such metric can work(plus you will fight configs).



The issues with pre-production testing - Replication

Failure Cannot be legislated/regulated!

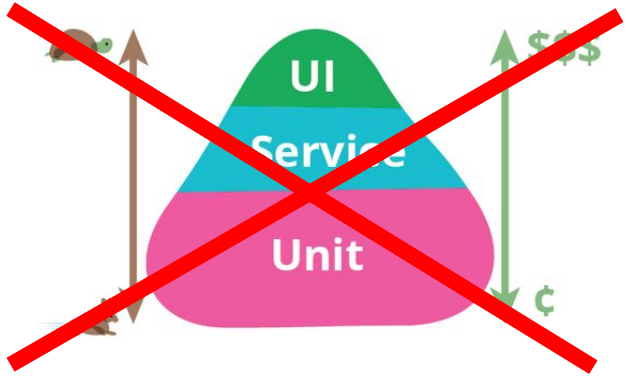
- We need a new approach
- We need a new mentality
- Instead of trying to predict everything(which is impossible)
- We need to have better understand on what's going on and be able to isolate failure and test in production.



Testing Diversity

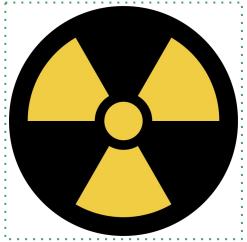
Testing in production

Pre-production	Deploy	Release	Post-release
- Unit tests - Functional tests - Component tests - Stress tests - Fuzz tests - Static analysis - Property based tests - Coverage tests - Benchmark tests - Regression tests - Contract tests - Lint tests - Acceptance tests - Mutation tests - Smoke tests - UI/UX tests - Usability tests - Penetration tests - Threat modeling	- Integration tests - Tap compare - Load tests - Shadowing - Config tests - Soak tests	- Canarying - Monitoring - Traffic shaping - Feature flagging - Exception tracking	- Teeing - Profiling - Logs/events - Chaos testing - Monitoring - A/B tests - Tracing - Dynamic exploration - Real user monitoring - Auditing - On-call experience

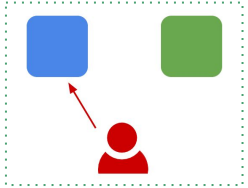


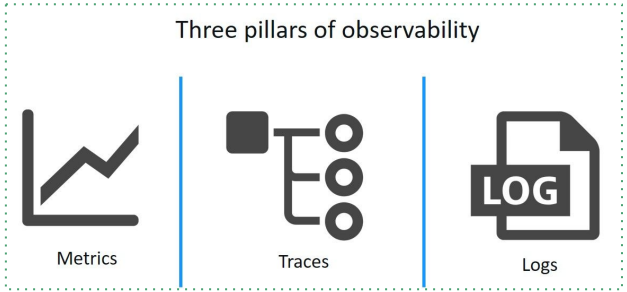
Pattern Landscape

Nuclear Options

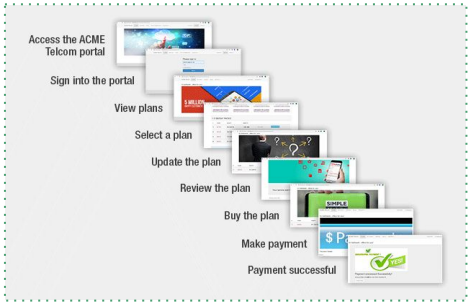


Blue/Green Deployment

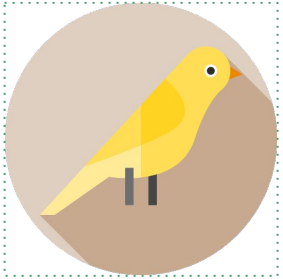




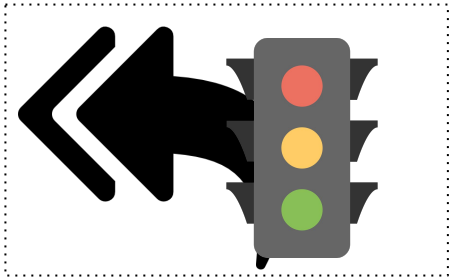
Synthetic monitoring



Automated Canary / Split Traffic

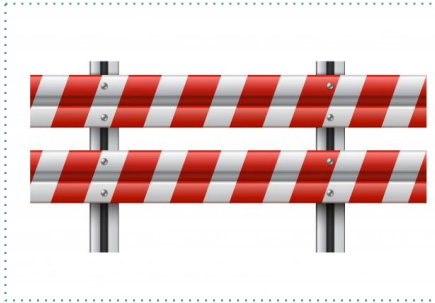


Reply Traffic

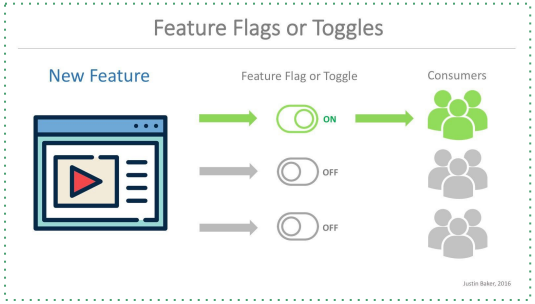


Dark Canary / Dark Launch

Safe Guards

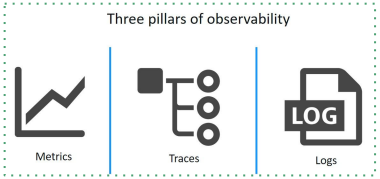


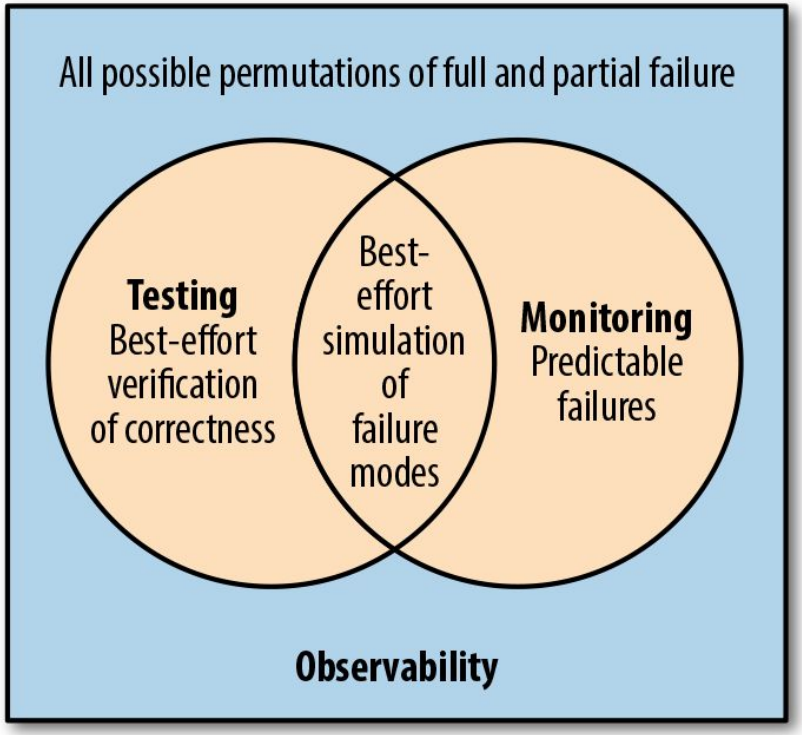
Pattern Landscape - Toggles



- Common in dramatic new versions (e.g. New News Feed)
- In a nutshell is Properties + IFs in the Code
- Can be Role based, Strategy Based. E.g all types of user, from a country, with same age or with x,y,z prop.
- Require Code Changes (Better than branching)
- Use the prefix “TEMP” in front of the flag to express it will be clean up soon. E.g “TEMP-feature-flag-01-login “
- And if you use branches name the branch. E.g “cleanup/TEMP-feature-flag-01-login“
- Some people love it other hate(abuses and tech debt)
- Flags should be used with proper Observability.
- Flag Status - e.g is the flag is not used for more than 7 days is safe to remove.
- Tech vs Non-Tech Flags (Business Feature vs Migration)

Pattern Landscape - Observability





- Basic Requirement for everything
- Whats Going On right now?
- Is the services working properly?
- Running your tests against prod is a form of Observability
- Maturity Levels
  - 5 - Self-Healing / Self-Operating Systems
  - 4- Alerts
  - 3- Dashboards
  - 2 - Logs
  - 1 - Praying that all is good
- Levels
  - Business - Domain Observability e.g Sales, Logins
  - Infrastructure e.g k8s pods, Network Traffic
  - Application e.g RPS, Errors, Tail Latency, queue len.
  - OS/Machine e.g CPU, Memory, Disk, Network

Pattern Landscape - Synthetic Monitoring

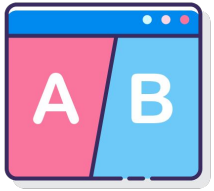
- A.K.A Semantic Monitoring or simply Business Monitoring
- Replicates the user experience (Great to measure user Latency)
- Answer basic and yet important question - is the business working fine? Can the users use the product?
- Similar to E2E Test or UI Testing but Running in prod and generating metrics/dashboards/alerts.
- Might Require Instrumentation in other services (e.g make some records invisible to services).
- Can be done with same E2E/UI Tools such as Selenium or Cypress.
- Ideally should run outside of your cloud (if you want know the user latency).
- Like any good test need to be self-contained and have proper isolation.

pingdom

Selenium

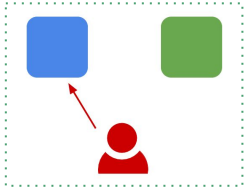
cypress.io

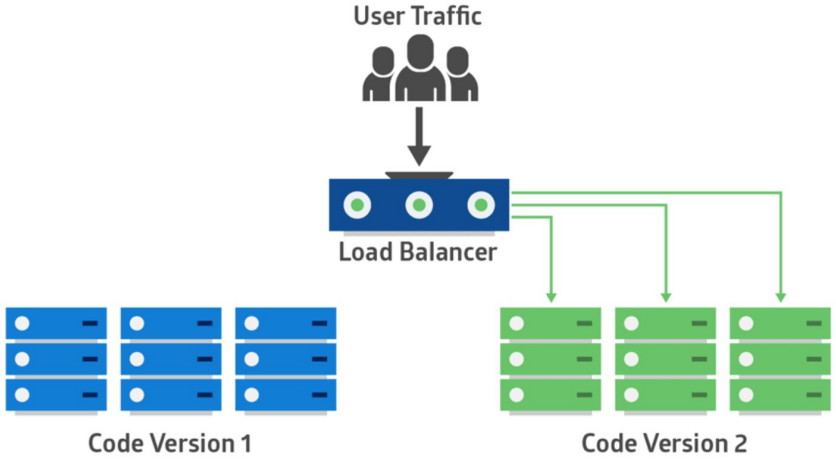
A/B/n Testing



- Split Testing
- Testing in Production for Business
- Used by all Big Tech companies in SV
- Huge source of Revenue tuning
- From simple colors and wording to business behaviors.
- Can be done with simple duplication on the UI. e.g page1 and page2.
- Easily will require feature toggles and coordination with backend services
- Microsoft Bing has 30B possible experiments (is very likely 2 users never get the same site).

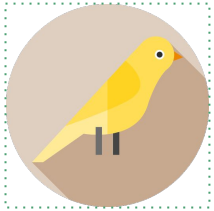
Blue / Green Deployment

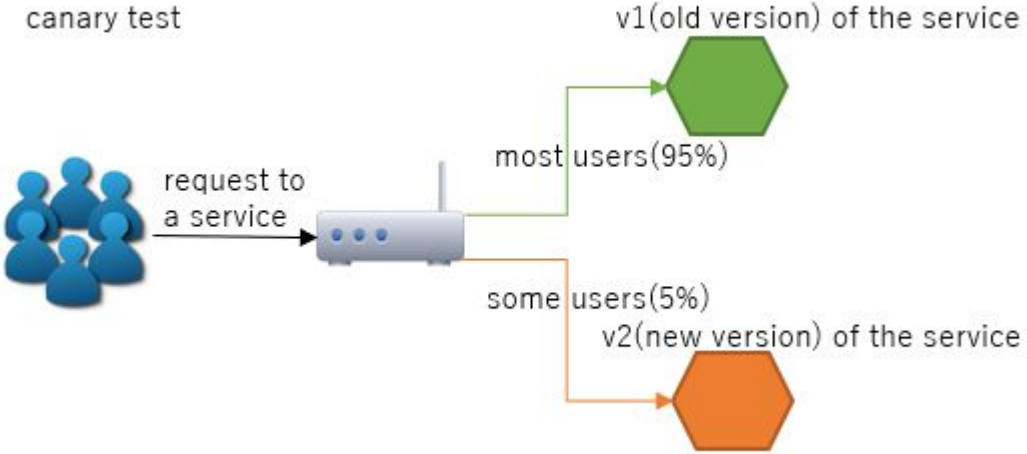




- It’s like a switch
- It’s about fast recovery
- Also reduce the blast radius
- Unfortunately is all or nothing
- Also can be used to test things in prod before affecting real users.

Pattern Landscape - Split Traffic / Canary



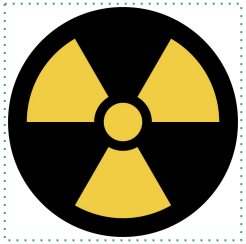


- IMHO Best form of Testing
- As real as you can get.
- It's all about reducing blast radius
- Works with progressive split traffic 1%, 5%, 10%, 25%, 50%, 75%, 100% of users
- Can be used with Green/Blue as well
- Require tooling:
  - Asgard/Spinnaker
  - Harness
  - Istio/Flagger
  - Argo

Pattern Landscape - Chaos Engineering

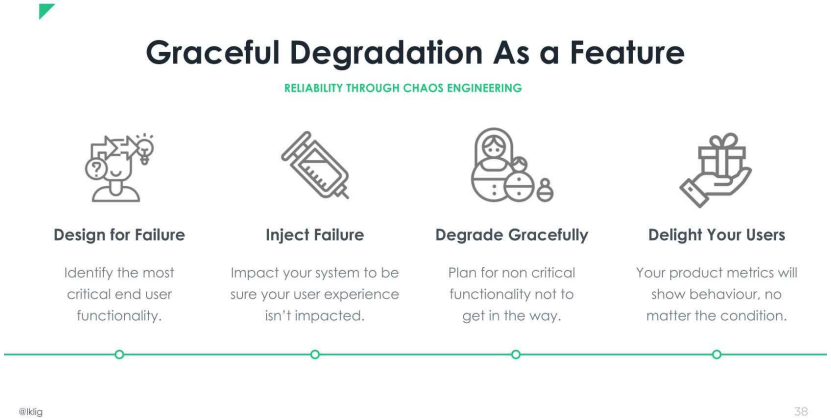
- You cannot legislate failure.
- Form of testing that make sure your infrastructure in antifragile
- Based on Experiments and Expectations
- Netflix runs in prod for years without issues
- Anti-Fragility mindset - Make sure you can recover rather than “prevent failure”
- Induce failure:
  - Tear down one machine
  - Burn CPU, Disk, IO, Network, Memory
  - Never Return a service call
  - Return all nulls
  - Return String bombs
  - Do GC Bombs
  - Introduce Lagging, Lose packages
  - Tear down a AZ, Tear down a Region

Facebook “Nuclear Options”



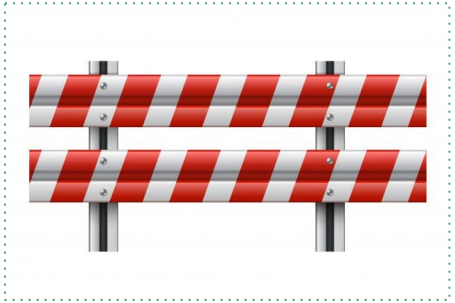
- ❏ Testing in Production and Chaos Engineering meet UX
- ❏ Facebook - Change Users Story - e.g Disable Recommendations, chat, Simple feed.
- ❏ UX Degradation -- Netflix
- ❏ Resilience Matrix

Resilience Matrix



Pattern Landscape - Safeguards

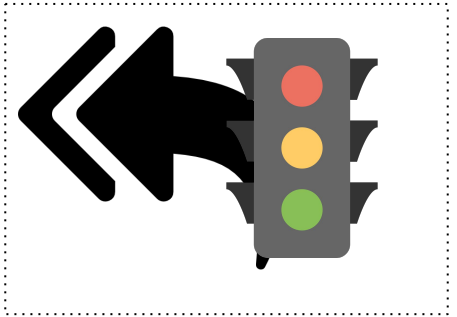
Safe Guards



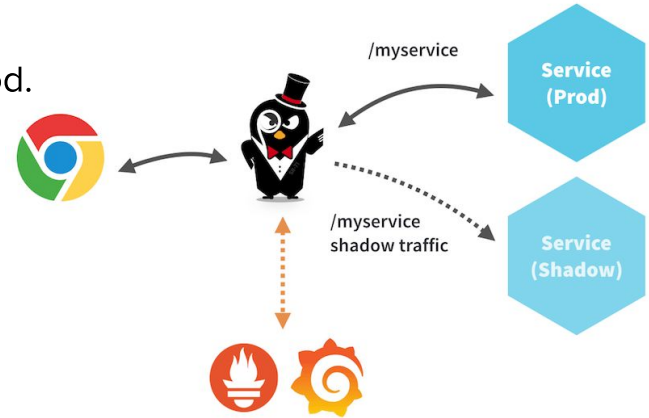
- ❏ Have threshold to STOP.
- ❏ Synthetical monitoring and alerts
- ❏ Make sure Tests are self-contained (Running tests in prod)
- ❏ Observability is the key for everything
- ❏ Dark Pool either in NO-Persist mode or persist in different DB or other env. (Code Instrumentation)
- ❏ Beta Users Program is a great form of isolation and will work well with split traffic (is you have a lot)
- ❏ Traffic split to 1 user first (your internal users in prod maybe?)
- ❏ Use Feature toggles in order to easily and fast switch things off or back to normal.

Pattern Landscape - Reply Traffic

Reply Traffic

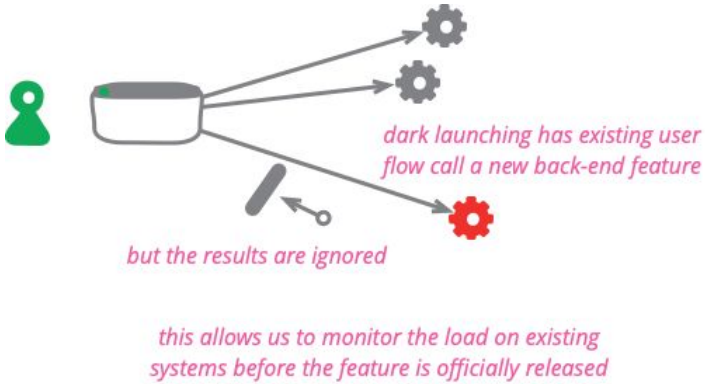


- ❏ Used in conjunction with Dark Canary
- ❏ Also known as shadowing or mirroring or dark traffic testing
- ❏ Capture production traffic
- ❏ It’s about isolation
- ❏ Should be Zero impact on real users (dark pool)
- ❏ Either by logs or distributed log solution like kafka
- ❏ Reply the traffic in prod(dark canary) or even in non-prod env.
- ❏ Might require some sudo anonymization as move outside of prod.
- ❏ Can get complicated and might require lots of infrastructure

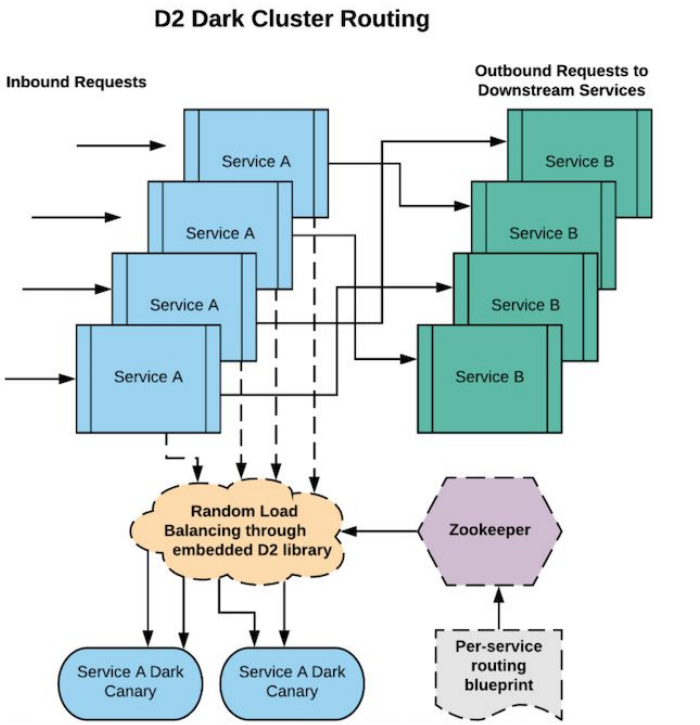


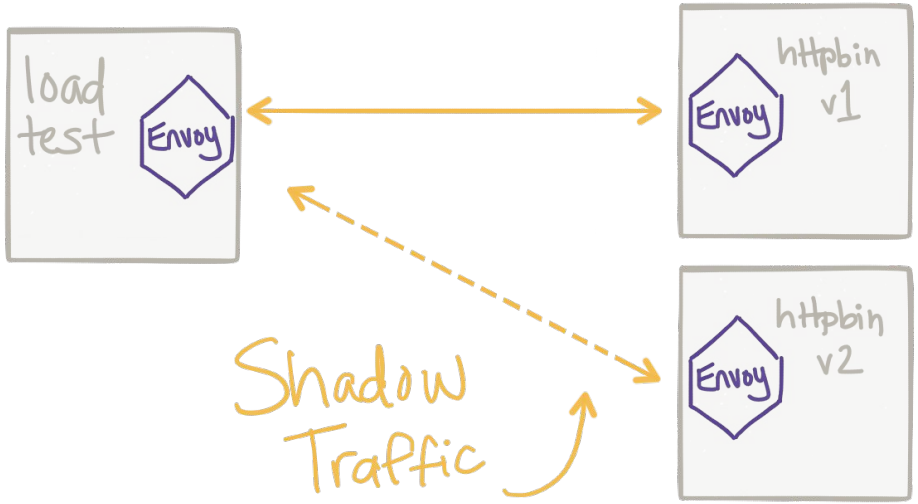
Pattern Landscape - Dark Canary

- It’s not a DC Hero :-)
- Can be used with or without Reply Traffic
- It’s a set of pools - similar to Blue/Green but results are discarded at end
- Used to see production load in services before they get exposed.
- Awesome way to catch bugs and scenarios that just happens in prod.



Pattern Landscape - Dark Canary





LinkedIN Dark Canary Formula

$$\text{Ratio of DarkRequests} = \frac{\text{(\# DarkServiceACluster instances)}}{\text{(\# ServiceACluster instances)}} * \text{multiplier}$$

$$\text{Outbound DarkRequest QPS} = (\text{Ratio of DarkRequests}) * \text{inbound QPS}$$

$$2\ \text{QPS} = \frac{2\ \text{instances}}{100\ \text{instances}} * 1 * 100\ \text{QPS}$$

https://engineering.linkedin.com/blog/2020/production-testing-with-dark-canaries

Testing in Production

Diego Pacheco