

# Uber

## Testing @ Uber

*How we changed Uber's engineering culture*

# Our Team



**Quess Liu**

Sr Staff Engineer



**Daniel Tsui**

Sr Engineering Manager

**1**

**Problem Statement**

**2**

**Humble Beginnings**

**3**

**What did we  
innovate on?**

**4**

**Changing the  
Engineering Culture**

**5**

**Key Strategies**

**6**

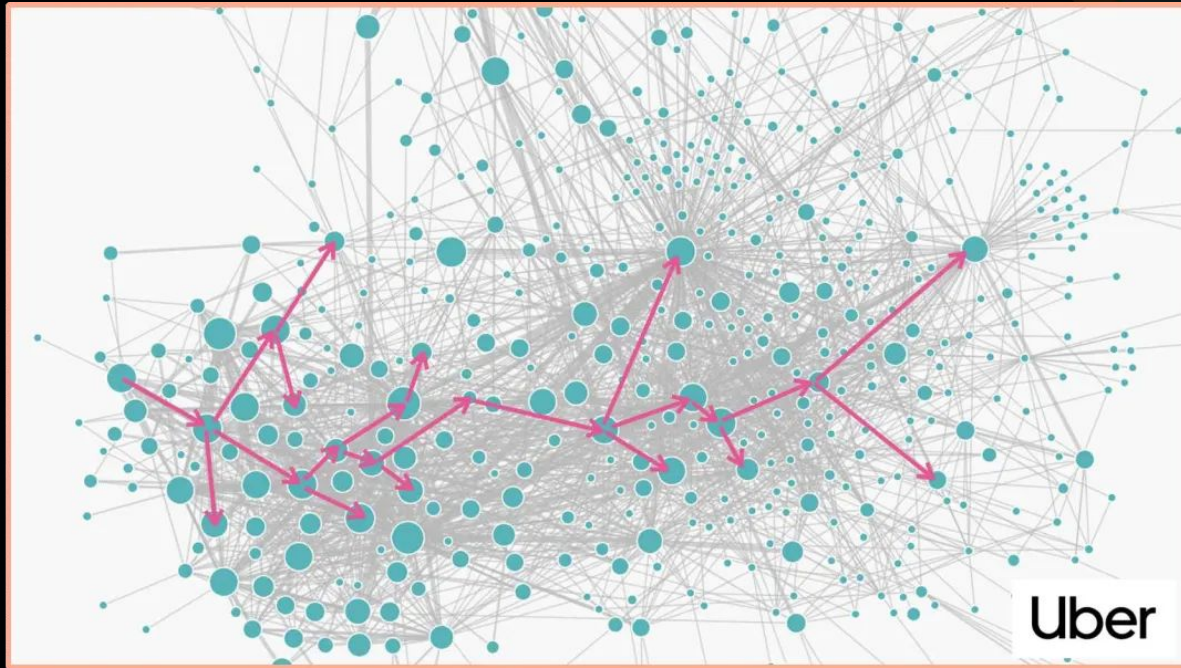
**What's next?**

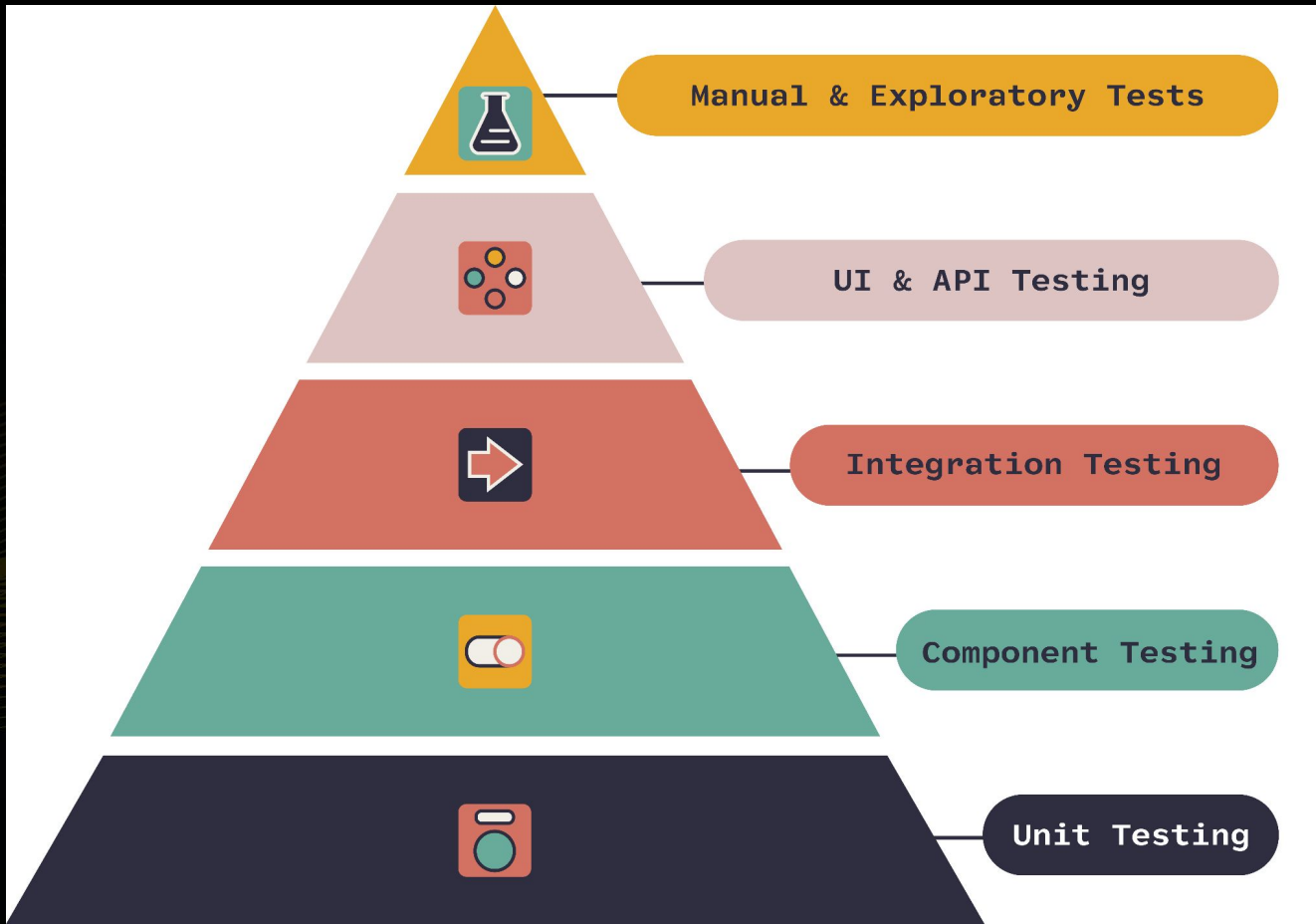


**1**

# **Problem Statement**

# Testing Microservices is Hard





# Going against conventional wisdom

- Testing Pyramid does not match our architecture
  - Complex business workflows span across many services
  - Isolated unit and component tests:
    - Prone to overmocking
    - Maintenance is hard - mocks and stubs fall out of sync with prod
    - Present limited value in preventing incidents





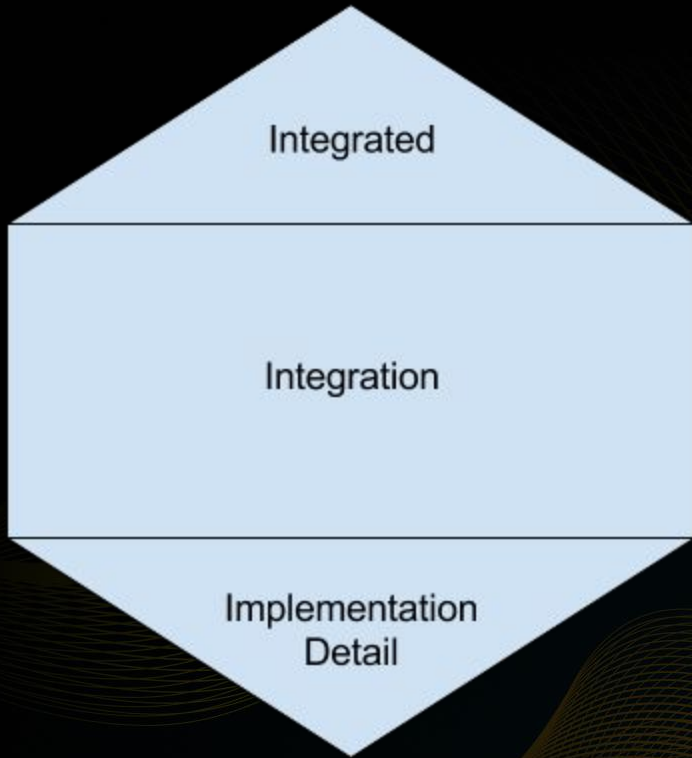
# Don't go too crazy with microservices!


If it's too late...

Up to 50% of our top incidents can be prevented with a basic backend integration test.



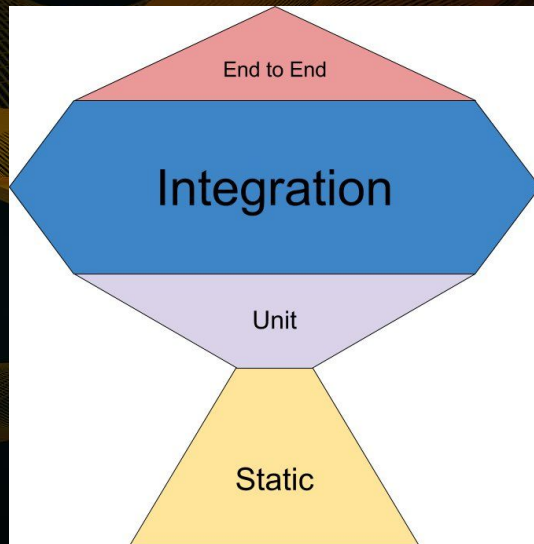




Kent C. Dodds   
@kentcdodds

"The Testing Trophy" 🏆

A general guide for the **\*\*return on investment\*\*** 🤑 of the different forms of testing with regards to testing JavaScript applications.



<https://engineering.atspotify.com/2018/01/testing-of-microservices/>



# 2

## Humble Beginnings

How did our journey start?

# 2020: Almost Quit

- COVID
- 25% layoffs
- 2 people left on the team supporting almost a dozen testing systems

# Stacked Odds

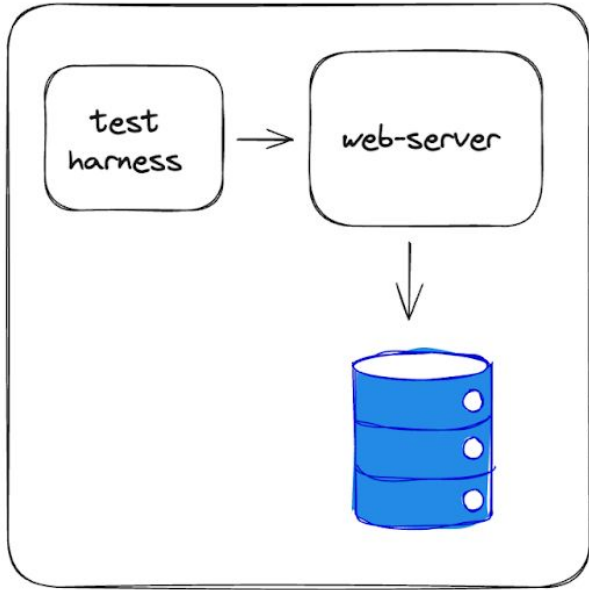
“I’m not sure if anyone in the industry has made integration testing easy... maybe it’s just a hard problem that can’t be solved”

- Uber Distinguished Engineer

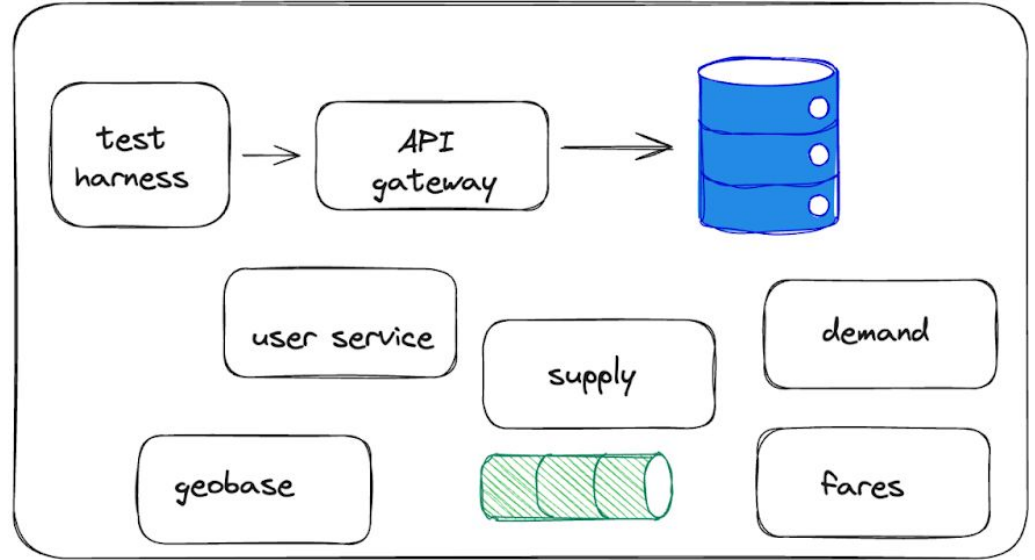
# Incremental value without boiling the ocean

Don't rely on aligning everyone to do the right thing

Start everything using docker-compose



microservice proliferation

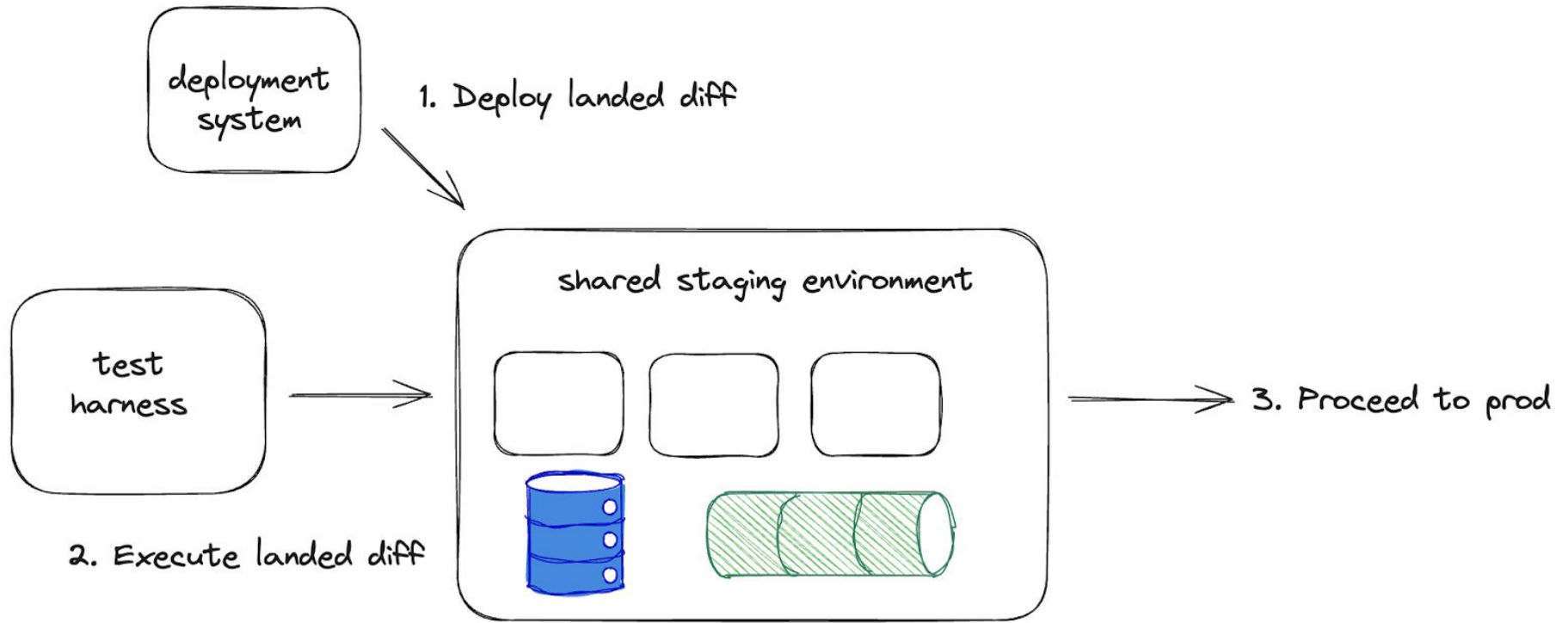


after a certain point, doesn't scale

**Every service must maintain a docker-compose config**







**Every service must setup and maintaining staging**



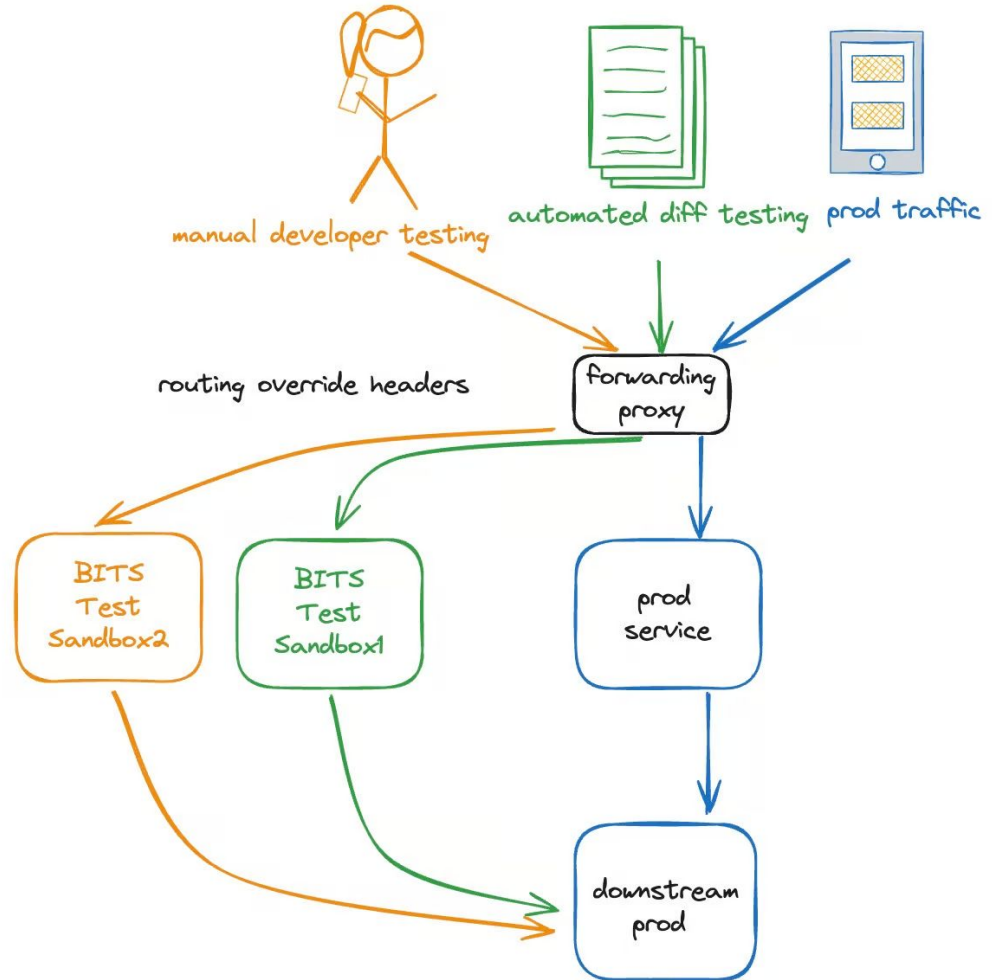


**3**

**Innovation**

# Test Against Production

- It's reliable.
- It already exists.
- BITS allows for concurrent testing.



# BITS - Test Container Orchestration

- Provision isolated test sandboxes
- Inject routing baggage

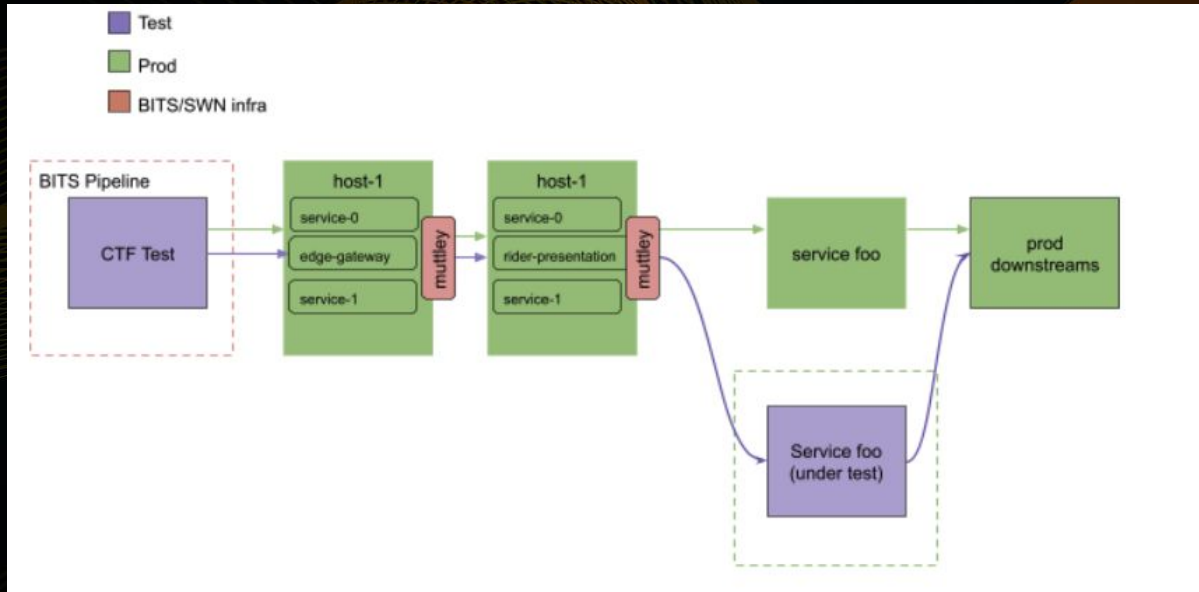
Baggage key: "routing-override"

Baggage value: unpadded base64url-encoded JSON object

```
{
  "services" : [
    {
      "service_name": <service_name>,
      "host": "<host>",
      "pool": "<used for cohorting metrics, required>"
      "ports": {
        "tchannel": "<tchannel port>",
        "http": "<http port>",
        "http2": "<http2 port>"
      }
    }
  ]
}
```

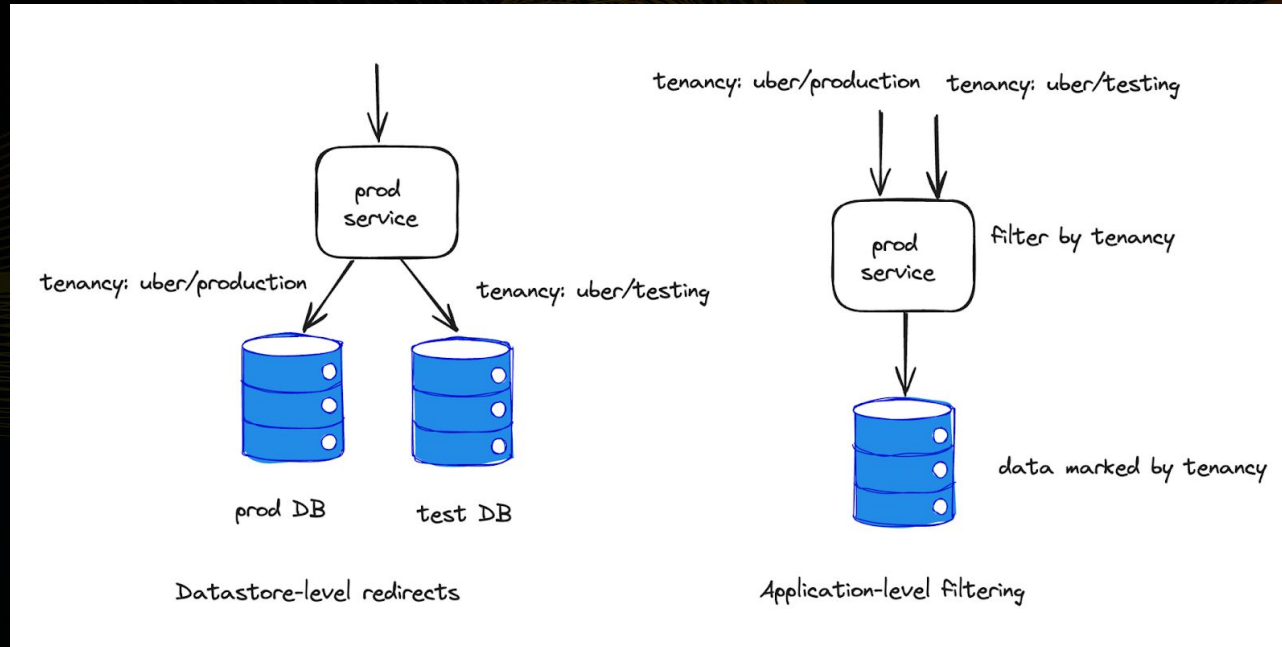
# Isolation

- Test through public edge
- APIs are entity-scoped (i.e. to a test account)



# Multi-tenancy

- Implement multi-tenancy for cross-entity interactions



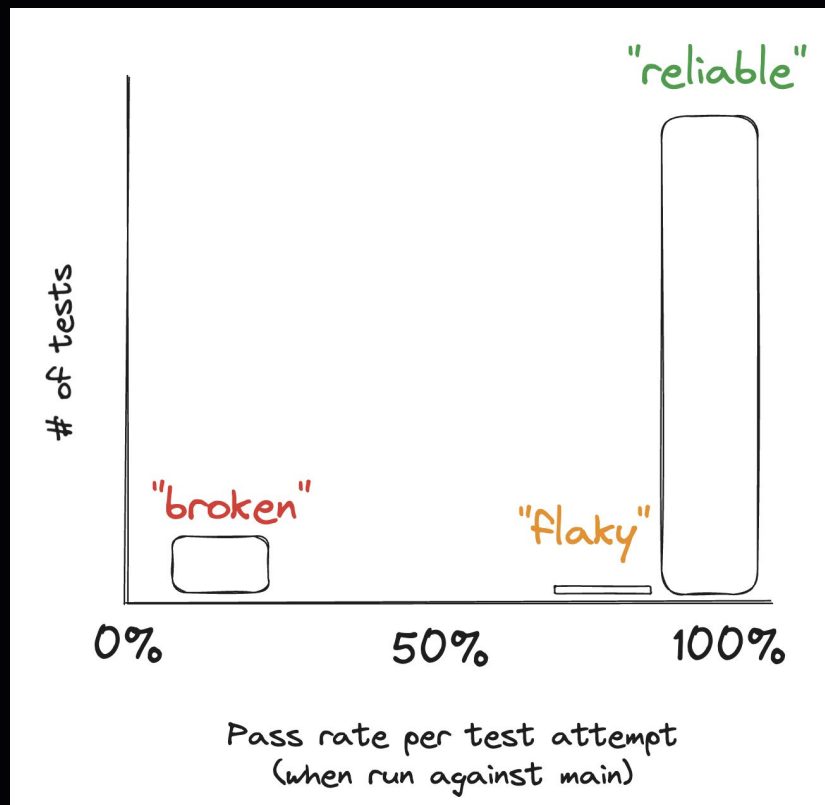


# 4

## Changing the Engineering Culture

# Broken, not Flaky

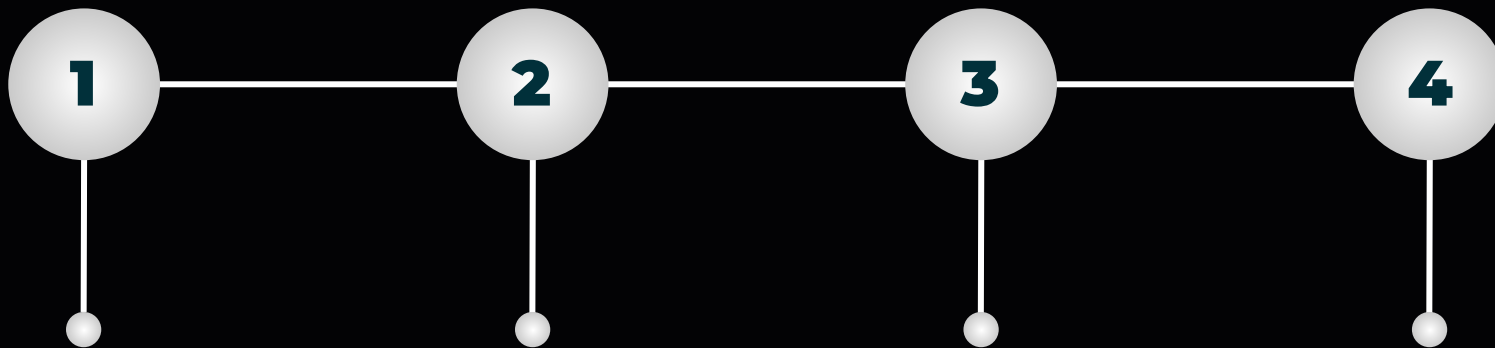
- Retries resolve any non-determinism
- Quarantine the broken tests unless they are critical



# Actionability: Placebo Executions

Primary Run Outcome	Placebo Run Outcome	Signal	Action
✓	✗	<ul style="list-style-type: none"><li>The diff or commit being tested fixes an issue on production (or an issue with a broken test)</li></ul>	<ul style="list-style-type: none"><li>✓ Diff/commit considered safe to land</li></ul>
✗	✗	<ul style="list-style-type: none"><li>There is an issue with production or the test is broken</li><li>Diff or commit being tested does not fix issue</li></ul>	<ul style="list-style-type: none"><li>✗ Diff/commit considered unsafe to land</li><li>Attention should be given to fix prod or the test</li></ul>
✗	✓	<ul style="list-style-type: none"><li>Diff or commit introduces a breaking change</li></ul>	<ul style="list-style-type: none"><li>✗ Diff/commit considered unsafe to land</li><li>Fix your code</li></ul>
✓	✓	<ul style="list-style-type: none"><li>No issues introduced with diff/commit being tested</li></ul>	<ul style="list-style-type: none"><li>✓ Diff/commit considered safe to land</li></ul>

# Our Evolution



**2021**

Organic Adoption

**2022**

Significant drop in incidents/1000 diffs

**2023**

Leadership convinced. Top-down program

**2024**

Half of diffs at Uber tested this way.  
71% reduction in incidents/1000 diffs



**5**

## **Key Strategies**

# Testing = Architecture

- Define your systems' core flows and make sure they are e2e tested
  - At Uber, this exercise is done by Sr Staff and Principal Engineers
- Our reliability is measured by your core flow availability, rather than individual API uptime





# Dogfood

- We defined our own core flow, wrote an E2E test for it, and it gates all our changes
- We also get paged if it breaks
  - We feel any flakiness in the system



# Don't rely on 100%

- What happens if context is dropped?
- What if your tests aren't 100% reliable?



# Win Over Customers (devs)

- Don't be afraid to add hacks to address top needs
  - Go back and refactor/fix later

Before: No one wants to write tests...

After: Integration testing is a pillar of every org's reliability strategy at Uber





# Leadership Support

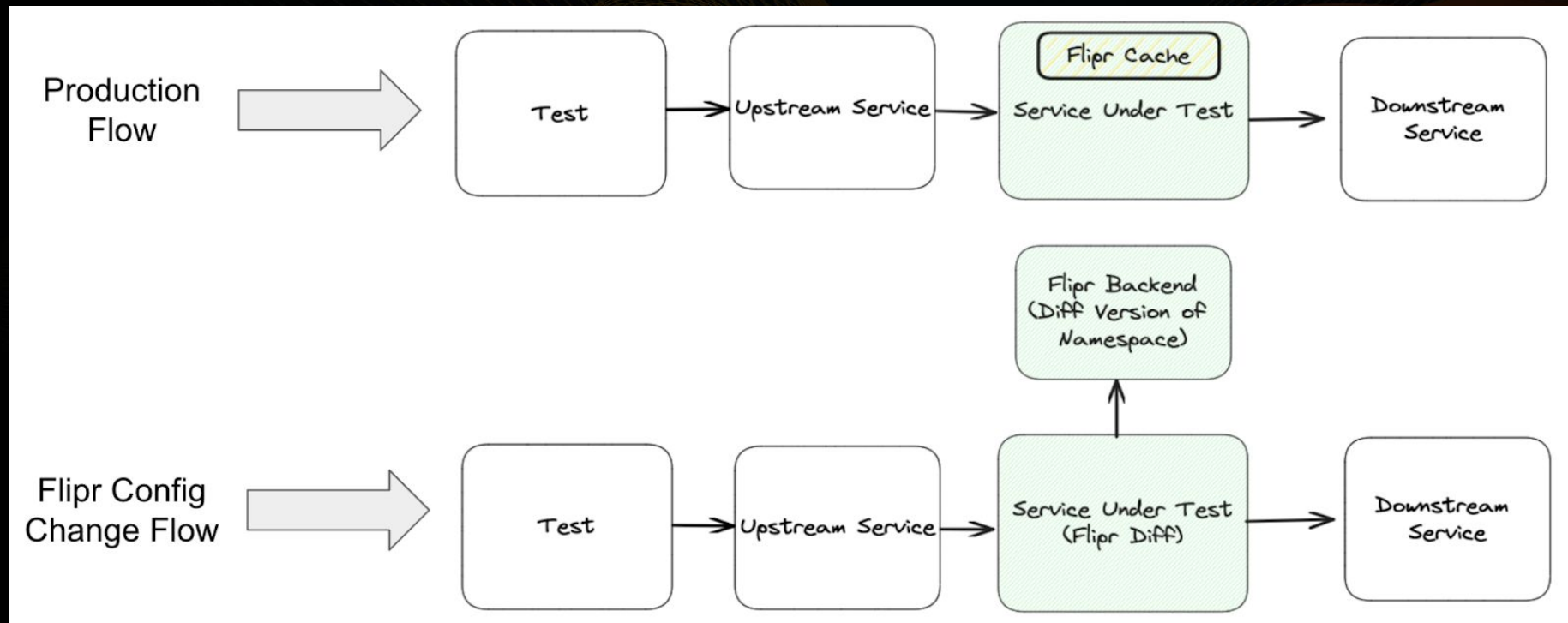
- Treat broken tests with incident-level urgency. You get paged
- Opt all tests out of being disabled

Result: Highest severity incidents in UberEats went to 0



# Design with Testability

- Many design sessions to build testing into Uber platform systems



**6**

**What's next?**



# Continue changing the culture

- Add CI/CD to all origins of code/config change
- Reduce test maintenance costs
- Move away from directly testing against production
- Context propagation



# Resources

<https://www.uber.com/blog/shifting-e2e-testing-left/>

# Analysis

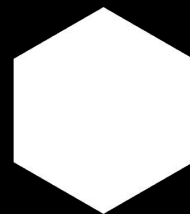
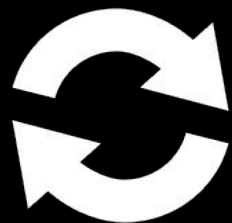

# THANKS

[name@company.com](mailto:name@company.com)

Twitter: @username

[www.company.com/careers](http://www.company.com/careers)

# Icons







# Agenda

- **Humble beginnings**
  - How did our journey start?
- **What did we innovate on?**
  - Explaining Uber's unique testing strategy
- **How did we change the engineering culture?**
  - Incorporating testing into architecture





Uber

# Uber Technologies

Description