Grafana Labs

Embracing Disruption

Adding a Bit of Chaos to Help You Grow!

Paul Balogh
Developer Advocate, Grafana Labs
@javaducky

Overview

1. **Why are we here?**
2. A brief history of how we test
3. How fault injection can help us
4. Where do we go from here?

Application reliability is **hard**

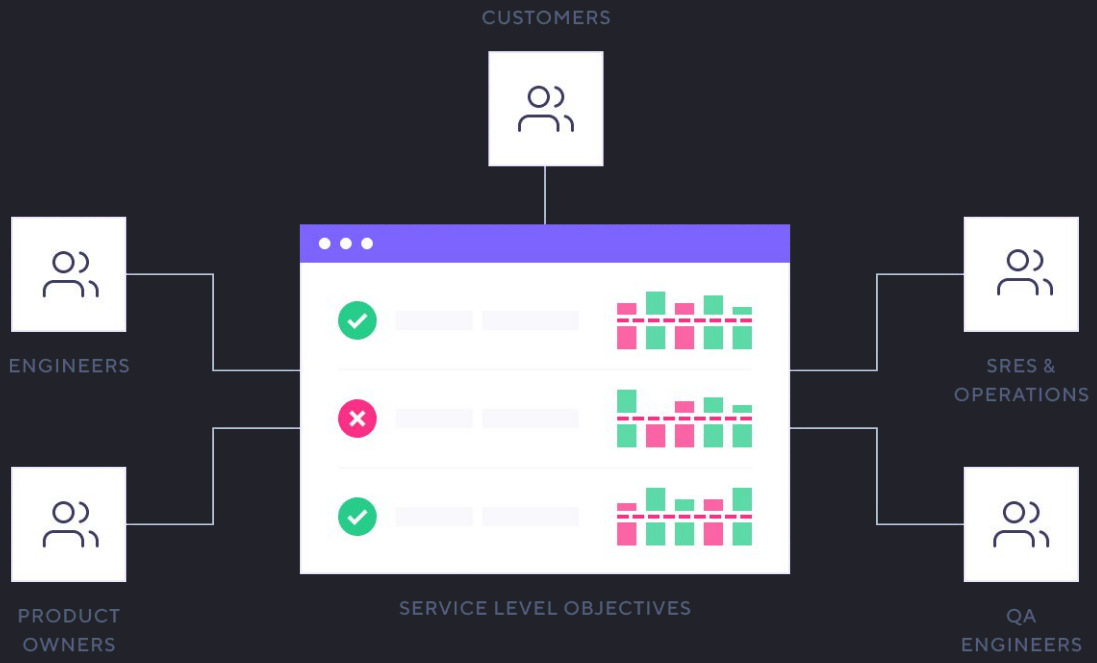
Complex architecture and infrastructure

Many potential points of failure

Inadequate tooling and practices

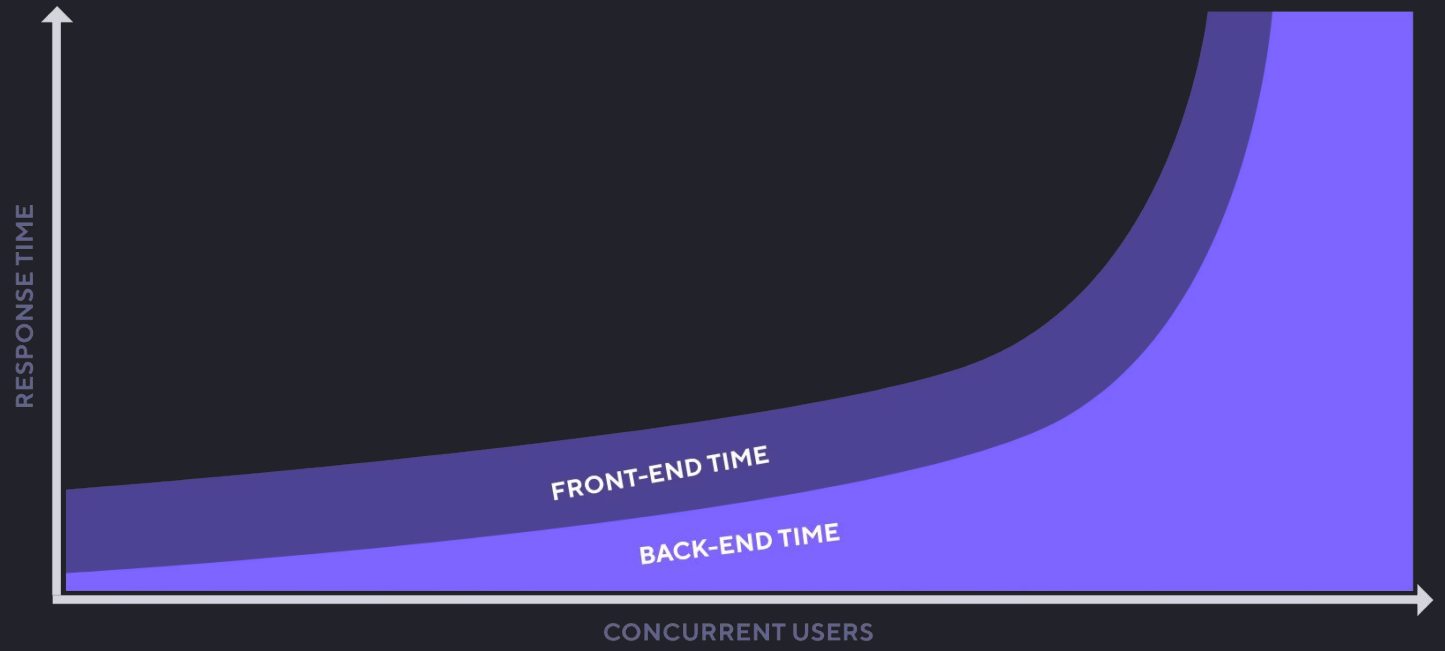
High demands on **availability**

SLOs



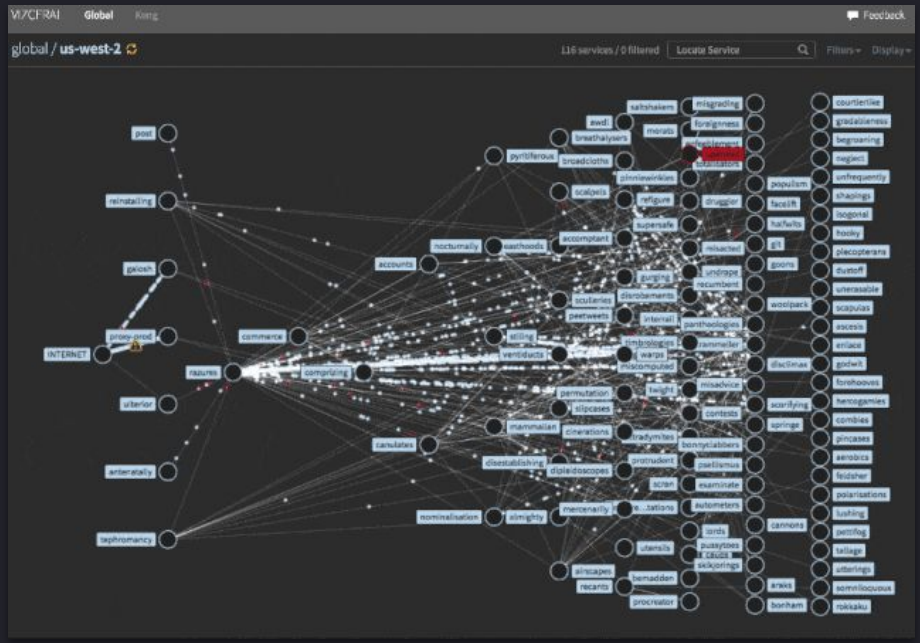
High demands on **usability**

UX



Ever increasing **complexity**

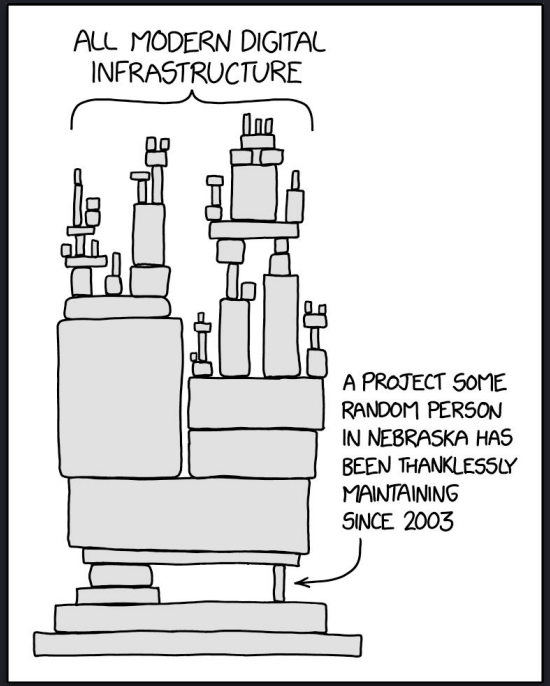
Distributed



(Example of Netflix services)

Potentially fateful **interdependency**

Fragility



https://xkcd.com/2347/

You are not alone

Overview

1. Why are we here?
2. **A brief history of how we test**
3. How fault injection can help us
4. Where do we go from here?

The way we test

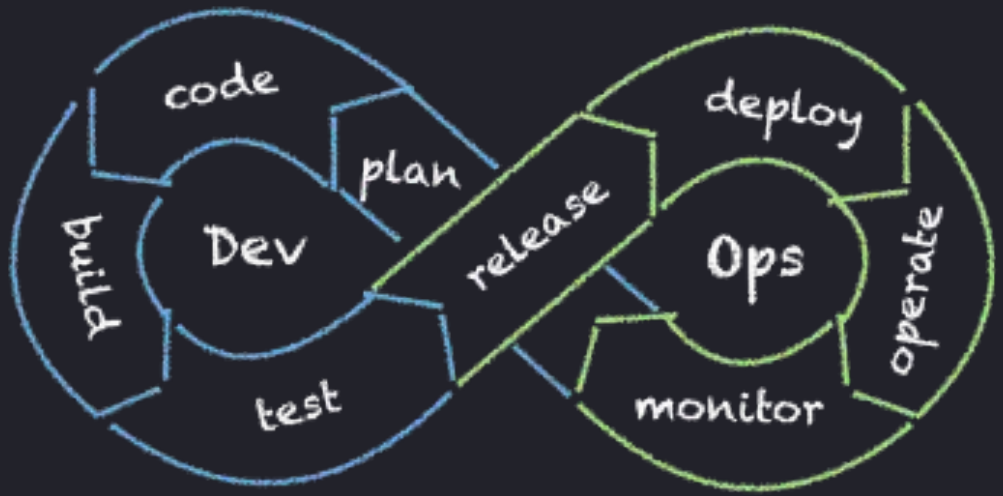
	Checklist / OLD WAY
Release frequency	Quarterly or biannually
Testing frequency	Before releases
How it's initiated	Manually
Testing environment	Test and Production

- QA bottleneck
- Lower coverage
- Late in process

The way we test

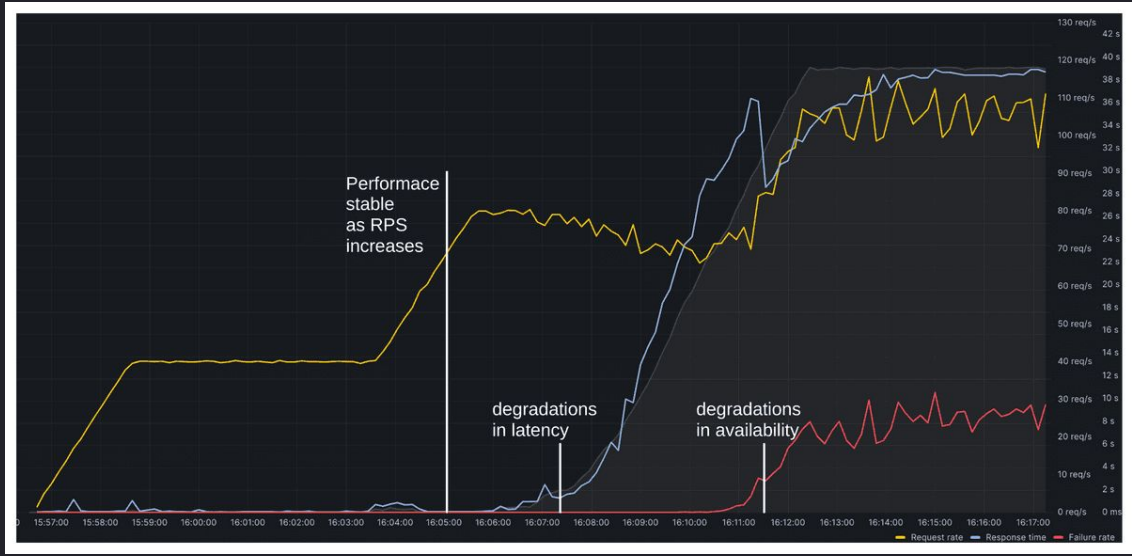
	Checklist / OLD WAY	DevOps / MODERN WAY
Release frequency	Quarterly or biannually	Weekly, Daily, As needed
Testing frequency	Before releases	Nightly, feature branches, continuous with synthetic monitoring
How it's initiated	Manually	Scheduled. Automatically as part of CI/CD
Testing environment	Test and Production	Staging (Long-lived) and ephemeral environments (Short-lived)

The way we test



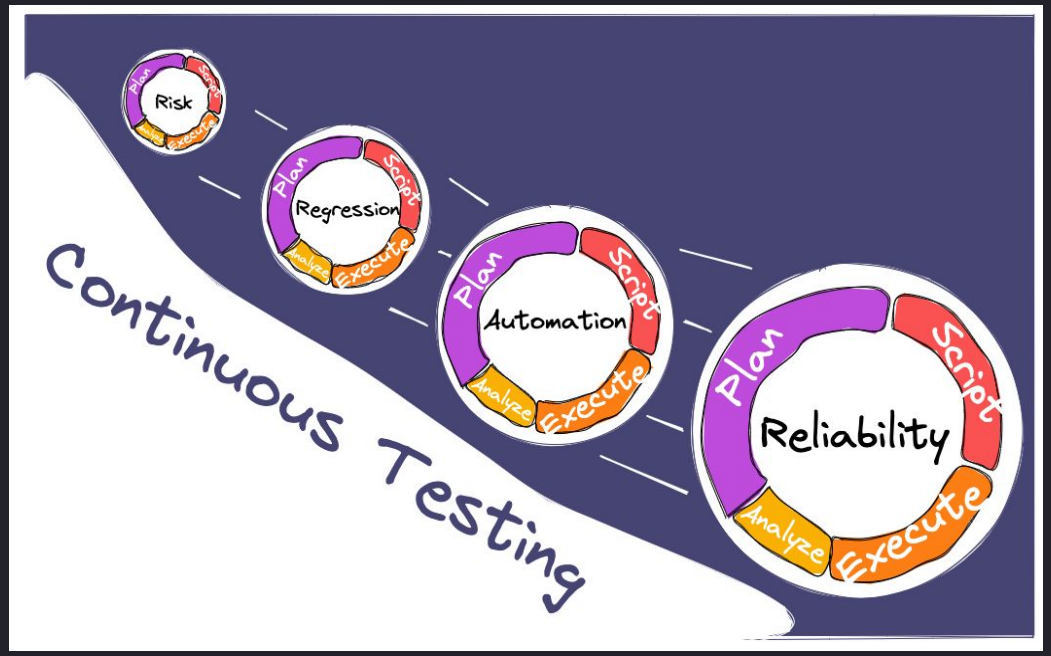
- Unit testing
- Integration testing
- Contract testing
- Functional testing
- E2E testing
- Load testing

The way we test



- Applications instrumented
- Observability platform available

The way we test



- Start simple
- Test frequently
- Continually expand
- Evolve over time

Yet we learn from failure

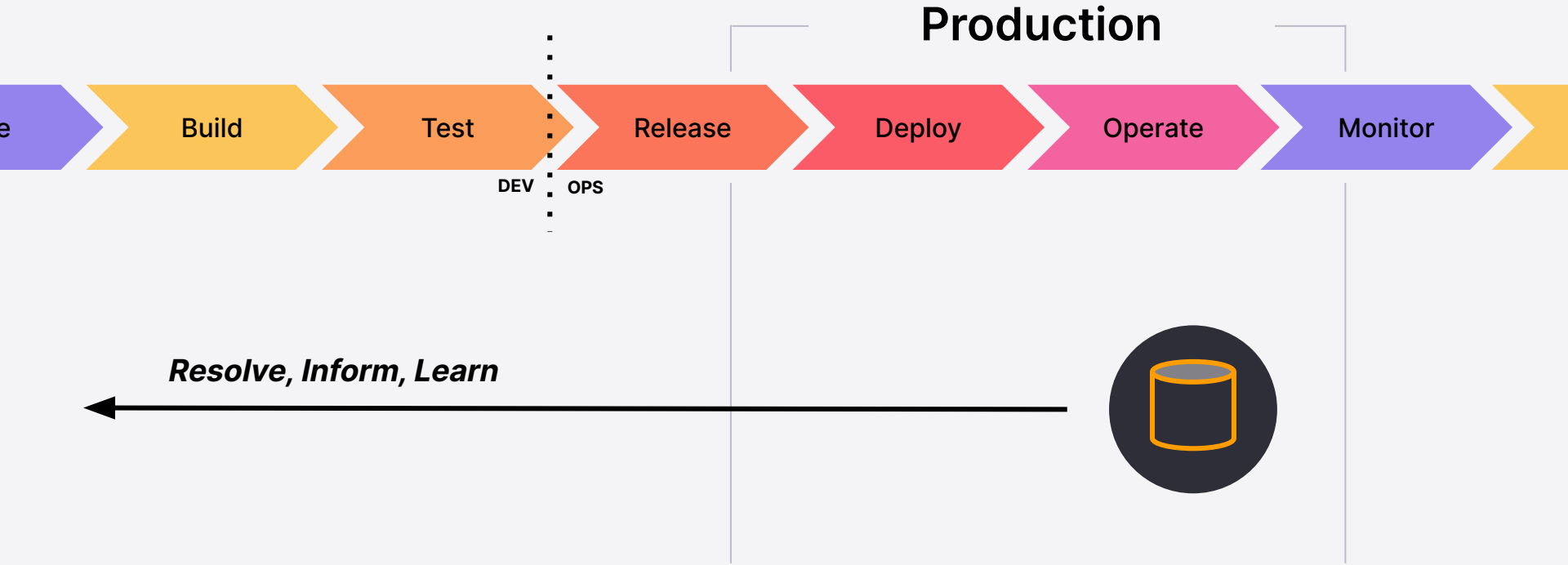
Overview

1. Why are we here?
2. A brief history of how we test
3. **How fault injection can help us**
4. Where do we go from here?

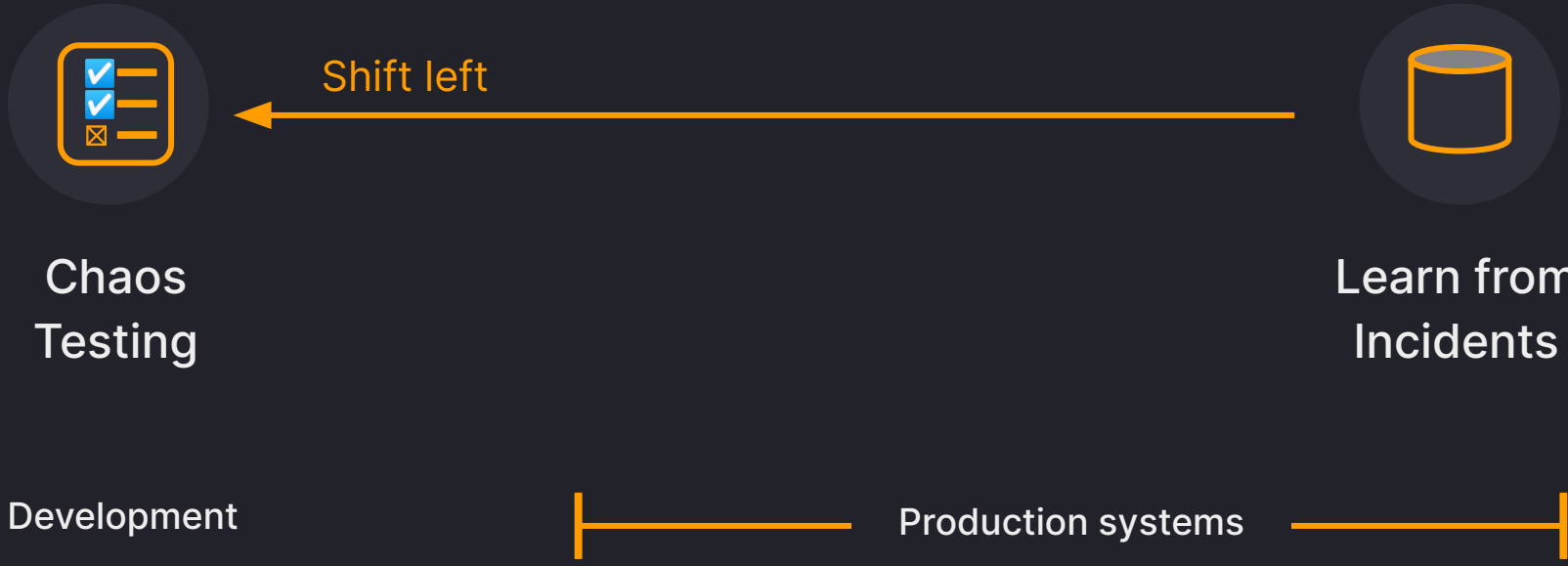
Fault Injection

A software testing technique which **introduces errors** to a system to **ensure** it can **withstand** and **recover** from those conditions.

Failure happens



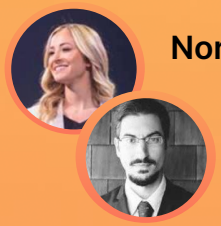
Build **more** confidence to withstand failures?



“”

From the distributed system perspective, almost all interesting availability experiments can be driven by affecting latency or response type.

- Chaos Engineering, O’Reilly



Nora Jones

Casey Rosenthal

Introducing **k6** and **xk6-disruptor**

k6, a *reliability testing* tool

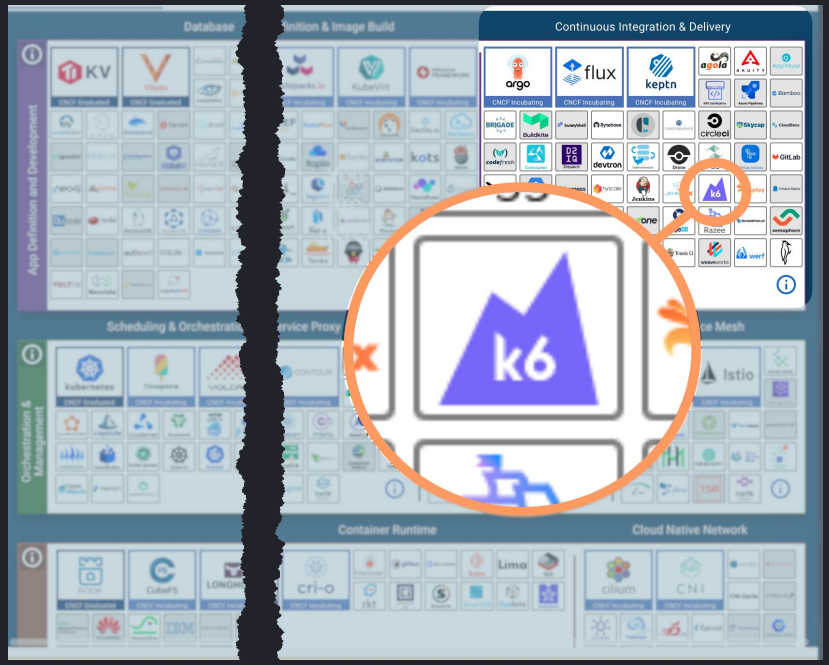
- Formerly known as *Load Impact*
- Open Source since 2016
- ~22.4k GitHub stars *(as of January 2024)*
- Promotes “shift-left” testing
- Acquired by Grafana Labs in 2021

github.com/grafana/k6

xk6-disruptor for *fault injection*

- Becomes a project in August 2022, evolved from previous experiments

github.com/grafana/xk6-disruptor



k6: a reliability testing tool

OpenSource

OSS is at the heart of **what we do** and helps leave the world a little better than we found it

Scriptable

CLI and API **designed for automating** your tests with pass/fail criteria using JavaScript syntax

Performant

A k6 engine **written in Go** making it one of the the best performing tools available

Extensible

Use Go(lang) code to **add support** for new outputs, protocols, and products from within your test scripts

Testing with errors

How effective is testing known errors?

92% of the catastrophic system failures are the result of incorrect handling of non-fatal errors

In 58% of the cases the resulting faults could have been detected through simple testing of error handling code

"Simple Testing Can Prevent Most Critical Failures: An Analysis of Production Failures in Distributed Data-Intensive Systems"
Yuan et al. USENIX OSDI 2014

How to improve error handling?

In 35% of the cases, error handling code falls into one of three patterns:

Overreactive. Aborts the system under non-fatal errors

Low Context. Was empty or only contained a log printing statement

Incomplete. Related comments like “FIXME” or “TODO”

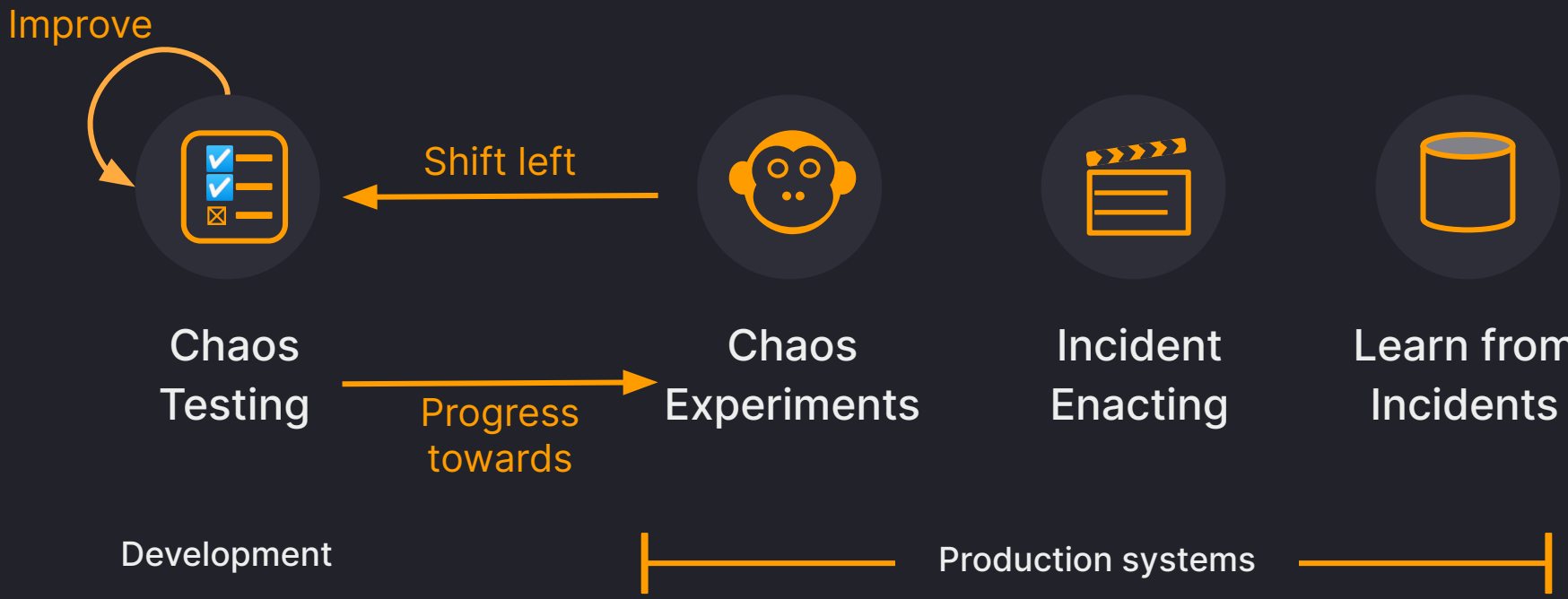
Introduce chaos testing

Incorporate chaos engineering principles early in the development process

Emphasize verification over experimentation

Change focus from uncovering unknown faults to ensuring proper handling of known faults

Continually **improve** reliability



Four **tenets** of Chaos Testing

Incremental adoption

Application Centric

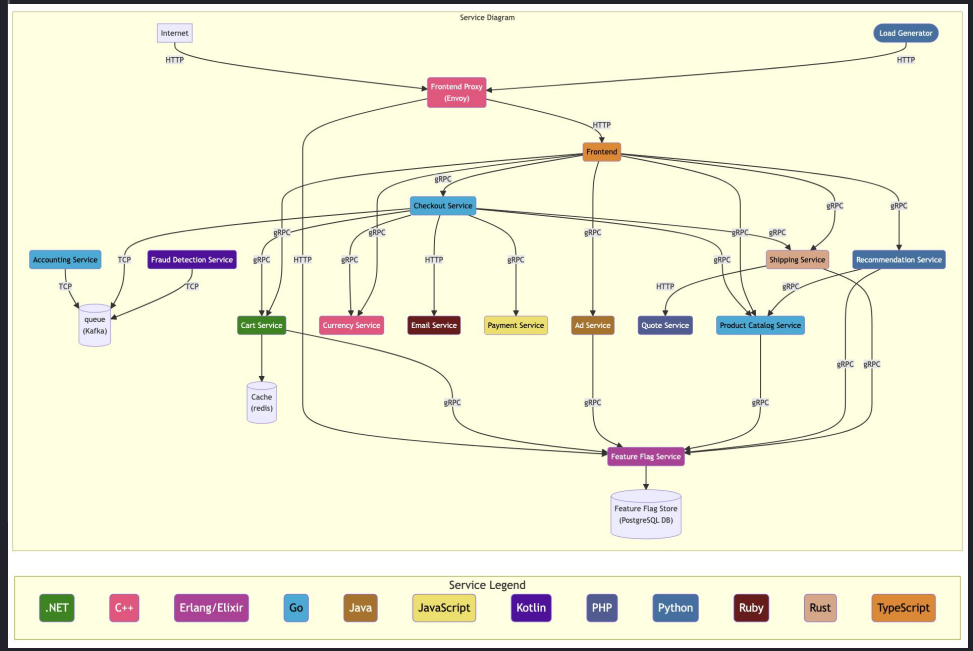
Chaos as Code

Controlled Chaos

Chaos Testing in action

OpenTelemetry Demo - Astronomy Shop

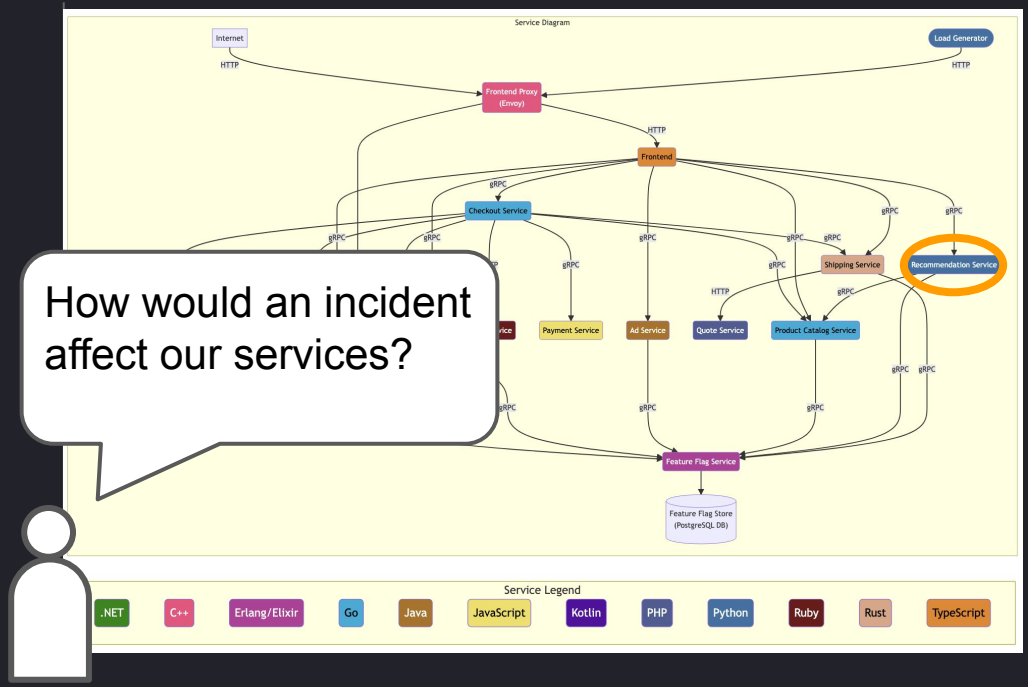
- Microservices architecture
- HTTP, gRPC, Kafka between services
- Polyglot (Go, Java, JS, ...)
- Kubernetes-ready



https://github.com/open-telemetry/opentelemetry-demo

OpenTelemetry Demo - Astronomy Shop

- Microservices architecture
- HTTP, gRPC, Kafka between services
- Polyglot (Go, Java, JS, ...)
- Kubernetes-ready



https://github.com/open-telemetry/opentelemetry-demo

Demo!

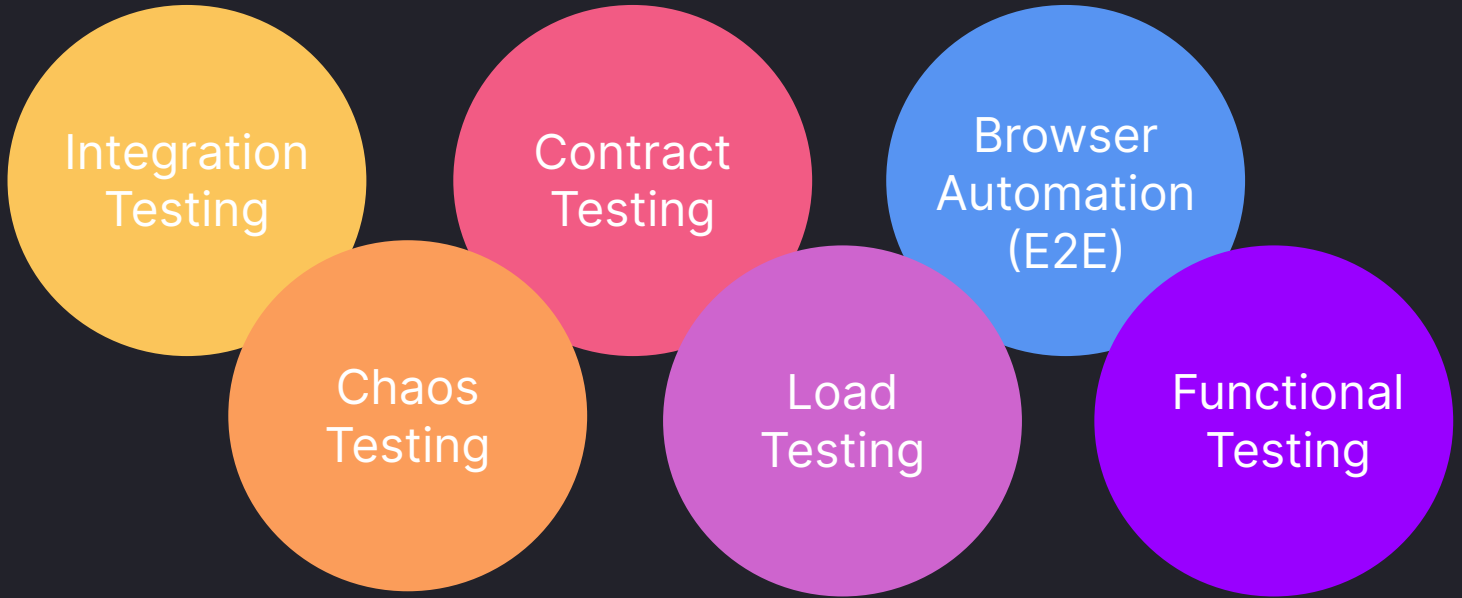
Chaos testing **principles** in action

- Tests can be **reused** to validate the system under turbulent conditions
- Conditions are defined in familiar terms: **latency** and **error rate**
- Tests have a **controlled effect** on the target service
- Tests are **repeatable** with results that are **predictable**
- Fault injection is coordinated from the **test code**
- Fault injection **should not** add any operational complexity

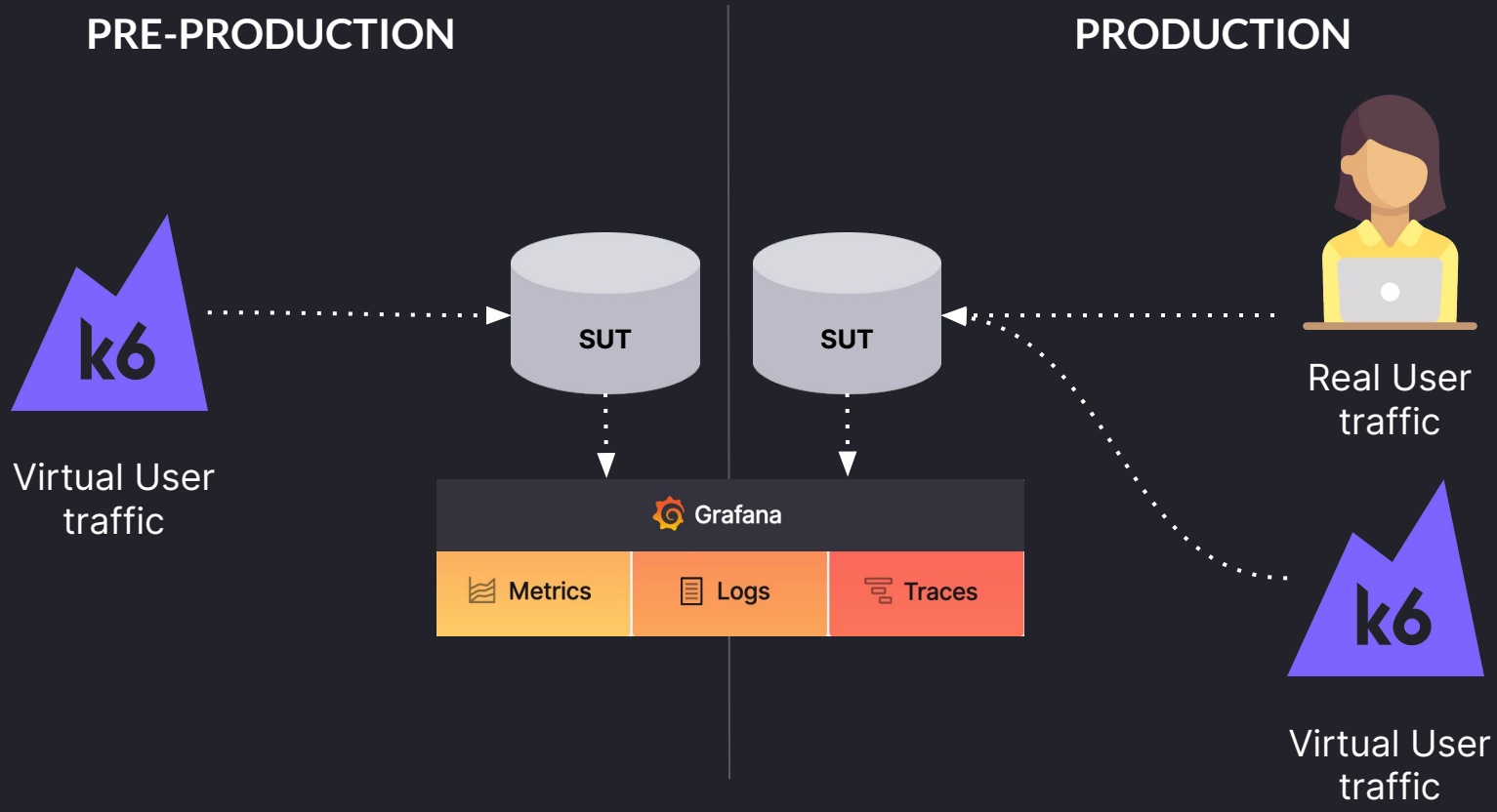
Overview

1. Why are we here?
2. A brief history of how we test
3. How fault injection can help us
4. **Where do we go from here?**

Reliability testing strategy



Proactively improve reliability



Final remarks

The ability to **operate reliably** should not be a privilege of the technology elite

Chaos Engineering can be democratized by promoting the **adoption** of Chaos Testing

To be effective, Chaos Testing **must** be compatible with the **existing testing practices** used by development teams

Our Goal

Make Chaos Engineering practices **accessible** to a broad spectrum of organizations by building a solid **foundation** from which they can progress towards more **reliable** applications

Grafana Labs

Thanks for participating!

Connect with Paul as
@javaducky or linkedin/in/pabalogh

k6.io/slack

grafana/xk6-disruptor