Planning For Failure

using chaos engineering

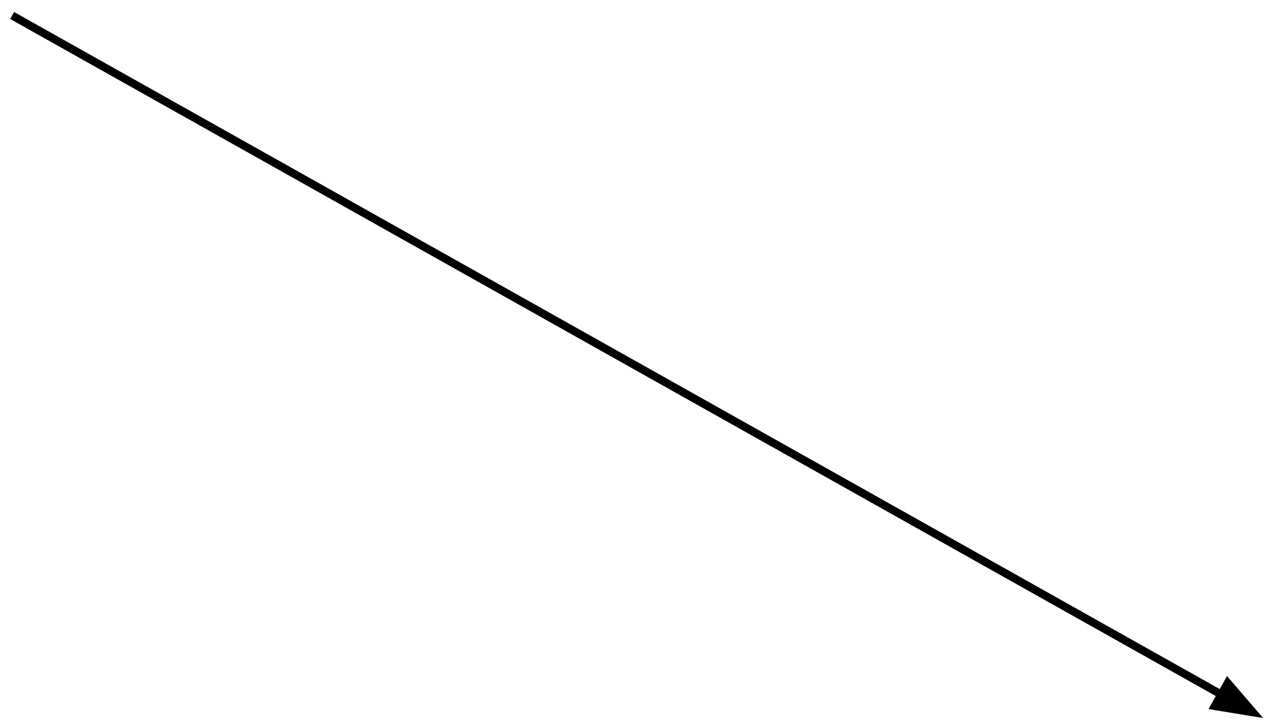
Amit Saha
Software Engineer @ Atlassian, Sydney.

Why am I qualified to speak on this topic?

Engineer who recently applied Chaos Engineering practices to help teams assess production readiness of their services

Hypothesis

Chaos engineering techniques can help continuously assess your production readiness



Failure Readiness

Agenda

Introduction

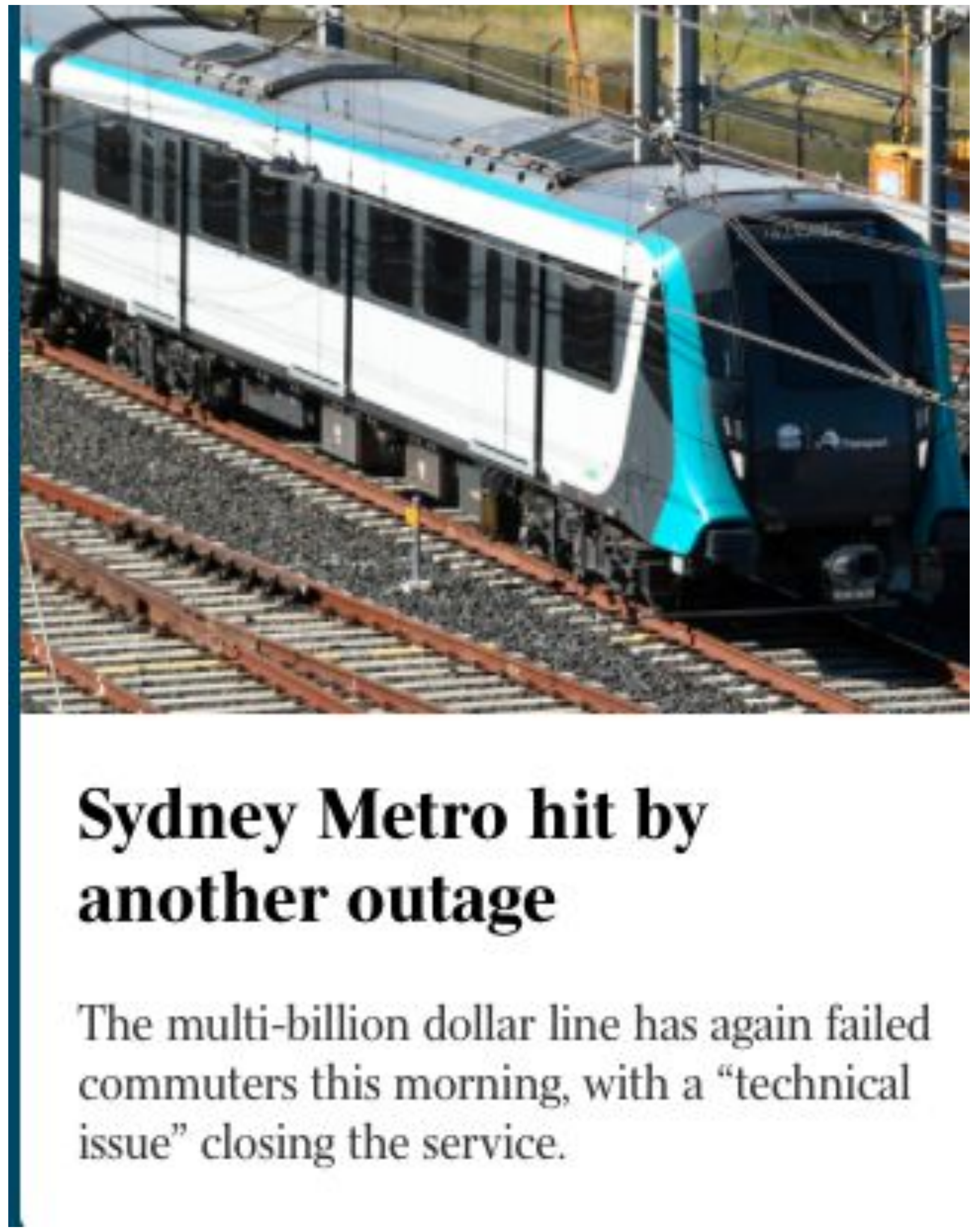
Organizing Experiments

People Systems

Chaos Engineering Tools

Summary

Introduction



Sydney Metro hit by another outage

The multi-billion dollar line has again failed commuters this morning, with a "technical issue" closing the service.

Systems Fail

Hardware - “mercurial cores - cores that don’t count”

Finite resources - memory, disk space, cpu cycles, network capacity

Check out “Silent Data Corruption” a paper by Facebook and related work by Google where they found that a task only failed on a specific *core*

Systems Fail

Software - “Bugs, Real world failure breaking underlying assumptions in software”

Systems Fail

People - “During an incident, on-call person finds they don’t have the necessary access to deactivate a malicious account”

Hence..

We put in circuit breakers, rate limiters, exponential backoffs, etc

We put in redundancy - Run X copies of my application

Hence..

We put in Disaster Recovery plans

We create Runbooks for people to follow

Hence..

We don’t aim for 100% reliability, we throw money at the problem by signing SLAs

And yet…

The next incident happens and catches us by surprise

-> Alerts didn’t fire?

-> Auto scaling didn’t happen?

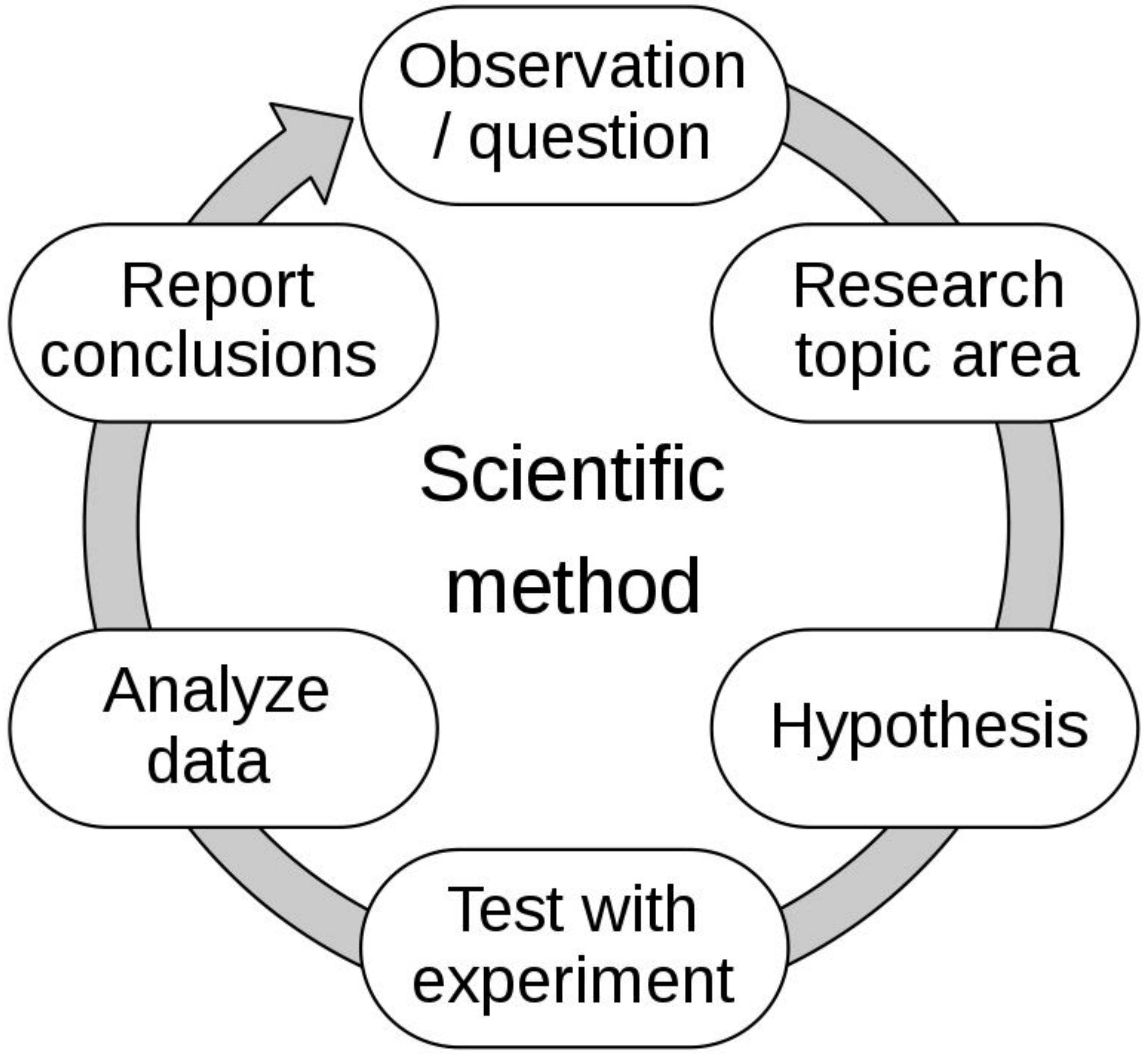
😱 DevOops

-> Oh? I thought we had circuit breakers..

-> Oh, **SLO BREACH!!!!!**

What can we do?

Scientific Method



Organizing Experiments

https://www.brooklynprinthouse.com/products/the-science-hat

Intentionally break our systems under controlled conditions

We break the carefully orchestrated mix of sync and async workflows, and verify the guard rails

What do we break?

- Finite resources - CPU/Memory/File descriptors
- Network
- Dependencies - libraries/services/datastores
- Any resource your software depends on
  - logging
  - monitoring
  - APM agents

Fault Injection Points

Example experiments

Async workflows, introduce timeouts longer than your polling duration..does your app keep polling forever?

Example experiments

Webhooks - receive the webhook correctly, but drop the payload without processing it.

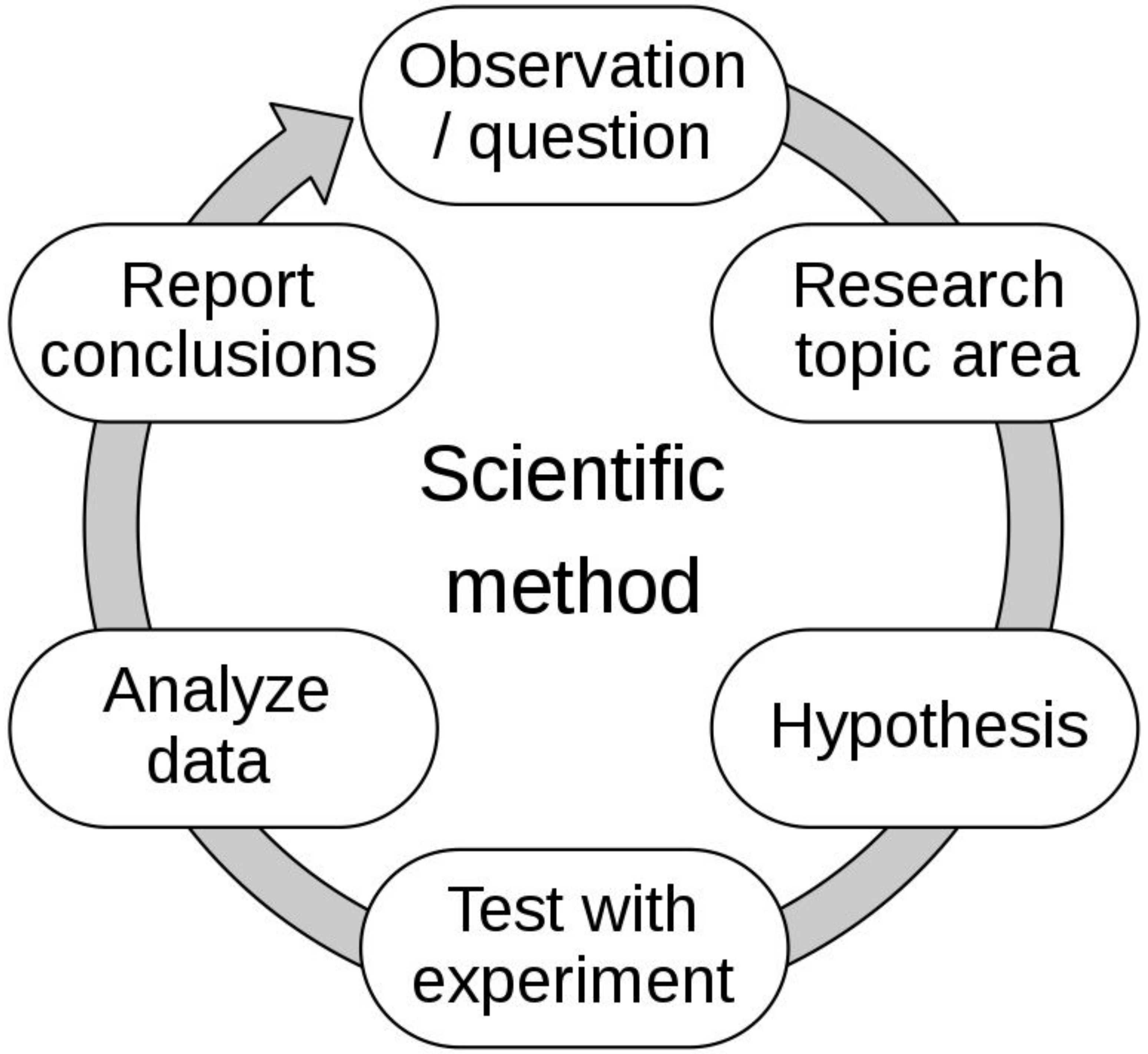
Example experiments

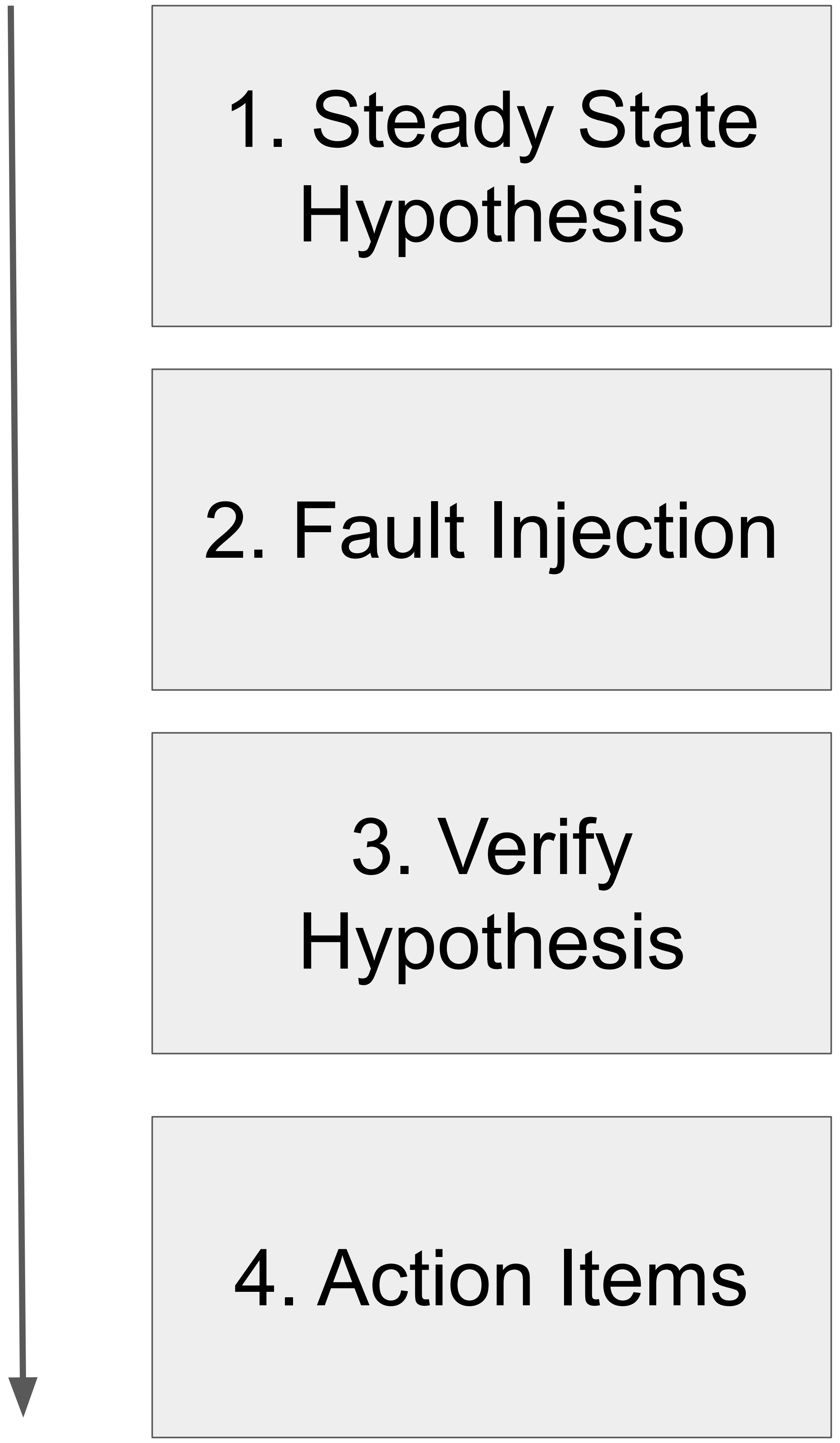
Say, your WSGI runtime worker processes are memory intensive and you enable recycling behavior and you were also careful to setup a jitter so that your workers don’t all restart at the same time.

Send traffic to your service at an expected requests per second. How does your worker recycling behavior manifest? Are they configuration giving you the expected behavior?

Kill a worker periodically - what happens?

A formal chaos engineering experiment





If my dependency, **service2** does not respond within 300 millisecond, **service1** will return an error within ~ 300 milliseconds

Add a latency of 5 seconds between application and database

Did the application return an error after ~300 millisecond or did it wait longer?

If the application waited longer, check timeout configurations, repeat experiment

Logistics - Before

System Design and Architecture Knowledge essential to come up with interesting experiment ideas

1. Brainstorm experiment and hypothesis i.e. step 1

2. Choose your tools for failure injection i.e step 2

3. Schedule a meeting with the key people - people who would be on-call, team leads, architects. Block time (~1-1.5 hours) for steps 3 and 4.

Brainstorming an experiment!

https://en.wikipedia.org/wiki/Miniature_wargaming

Logistics - Before

Wargame Champion - a person who coordinates the various logistics, schedules the experiment and then orchestrates the experiment along with other team members

Ensure your failure injection tools/scripts work before the real chaos engineering experiment!

Logistics - Before/During

An ability to generate load/traffic - locust - a python community favorite for HTTP services

> If you have integration/synthetic tests, they are useful too. Combine them with Bash scripts

Logistics - During

Get everyone in a virtual room

Share dashboards/notification channels/logs

Blind failure injection versus pre-defined fault injection

Staging or Production?

One of those questions

Start *gradually* with with Staging and when you are confident, go for Production

Perhaps - Opt-out for Staging and Opt-in for Production

Pre-requisites

An ability to run arbitrary commands or having admin style privileges in your cloud/on-prem account

Instrumentation - metrics/dashboards/logs/data

People Systems

https://www.seekpng.com/ima/u2q8w7e6t4w7e6i1/

A “hypothetical” scenario

- Pager beeps - we have spam
- Cool, let’s start deactivating the accounts creating those spam requests
- Ah! I don’t have the right permissions, let me get Jane. Oops.
- Jane - let me get Jill
- …. spam continues ….
- Jill - Okay, I have started to deactivate the accounts

People are systems too!

- Incident Management
- Training Day
- Team building exercise!
- Playbooks
- Bus Factor - zombie apocalypse (Release it! Chapter 17)

- "Chaos Engineering for People Systems", Dave Rensin - Google
  - Companies are distributed systems
  - Stay-cation

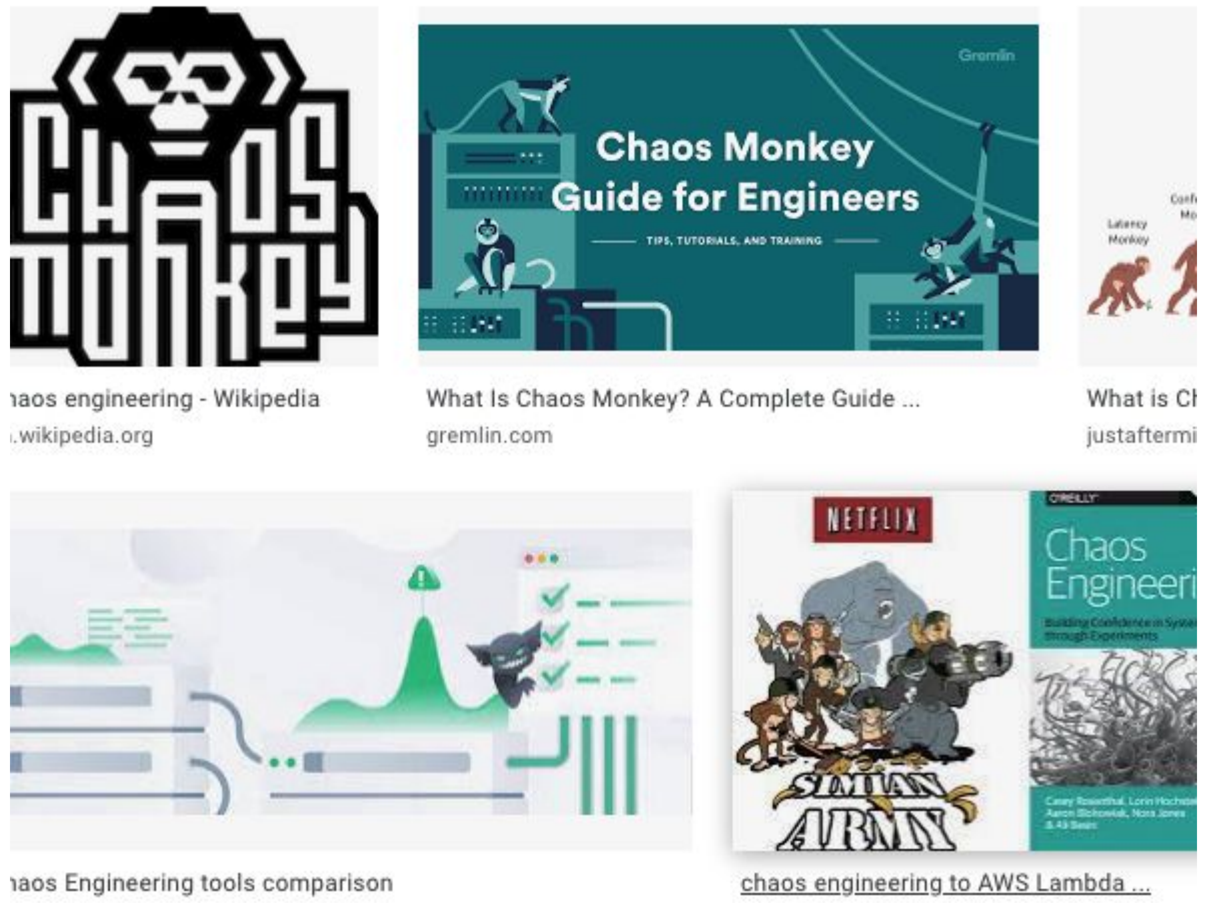
Key points to remember

1. Let people know that you are running a chaos engineering experiment

2. Be mindful of timezones of the key on-call people

WarGames (1983)

Chaos Engineering Tools



haos engineering - Wikipedia
.wikipedia.org

What Is Chaos Monkey? A Complete Guide ...
gremlin.com

What is Cl
justafterm

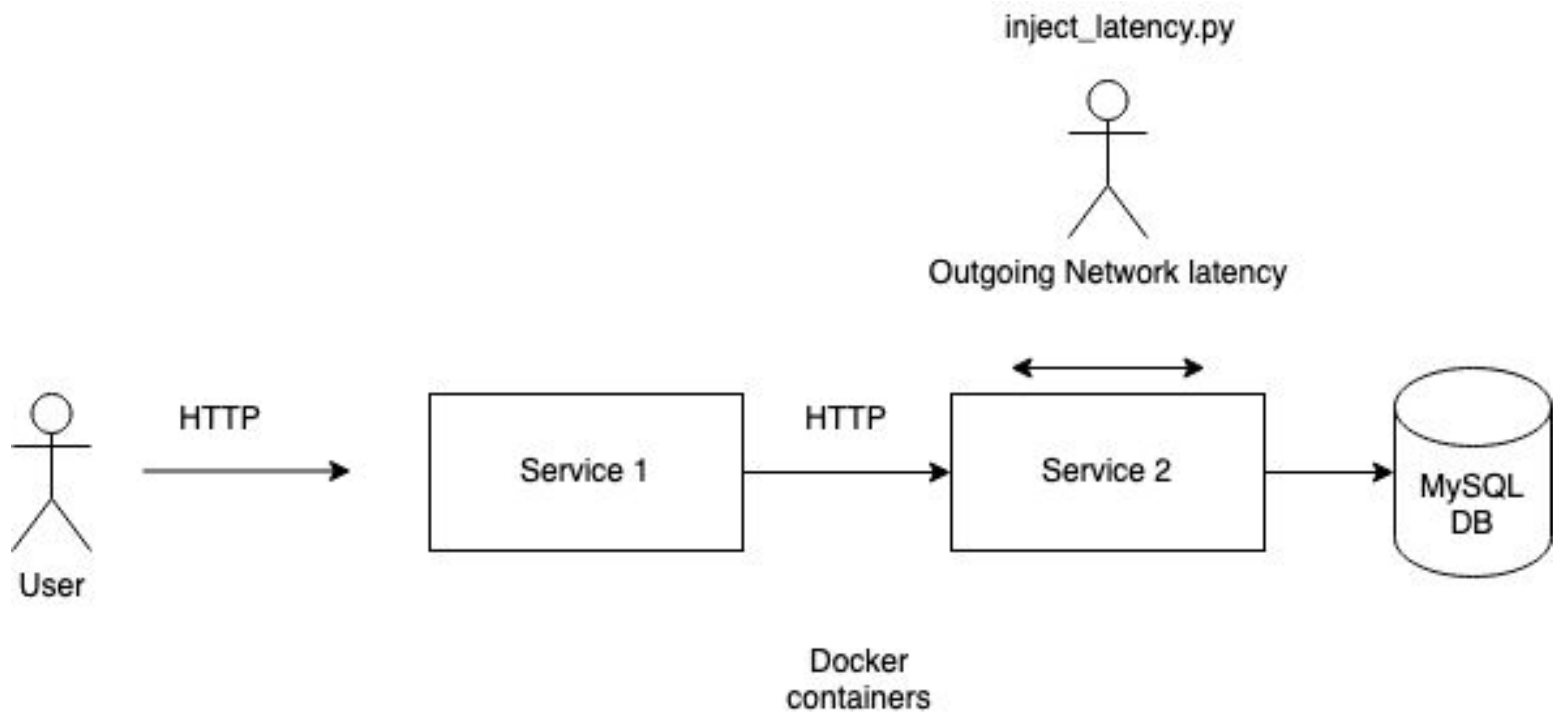
haos Engineering tools comparison

chaos engineering to AWS Lambda ...

Fundamental tools

- iptables, tcpkill - introduce DNS based sinkholes, break connections
- toxiproxy, tc - (traffic control) - add delay to network traffic
- stress-ng - simulate CPU/Memory/IO stress

Chaotic Demo (Live? May be)



https://github.com/amitsaha/pycon-au-2021

ChaosToolkit

Inject various faults using external drivers

- Cloud providers - AWS/GCP
- Kubernetes
- Toxiproxy

https://chaostoolkit.org/

Extensible using Python!

AWS Fault Injection Simulator

You can do things like do a RDS failover

You can trigger errors into the API calls which require making calls to the EC2 API
- allow you to delay the AutoScaling API calls

Summary

Chaos Engineering

- Embrace the Chaos
- It’s about assessing your readiness when your systems fail
- Known failure scenarios may come together to teach you about unknown failure scenarios - emergent properties

“Sometimes you don't get a choice;
the Chaos Monkey chooses you”

Working with Chaos Monkey
https://blog.codinghorror.com/working-with-the-chaos-monkey/

“And that's why, even though it sounds crazy, the best way to avoid failure is to fail constantly”

Working with Chaos Monkey
https://blog.codinghorror.com/working-with-the-chaos-monkey/

Key Resources

1. Principles of Chaos - https://principlesofchaos.org
2. Chapter 17, “Chaos Engineering” from the book “Release It!”, Michael Nygard
3. “Chaos Engineering, Site reliability through controlled disruption”, Mikolaj Pawlikowski, Manning Publications - great for hands on learning.

Thanks

Attendees, PyCon AU Organizers, NextDayVideo folks

I am contactable at:

- @echorand
- mail@echorand.me

Talk materials: https://github.com/amitsaha/pycon-au-2021

My website: https://echorand.me