Continuous Resilience

Uma Mukkara

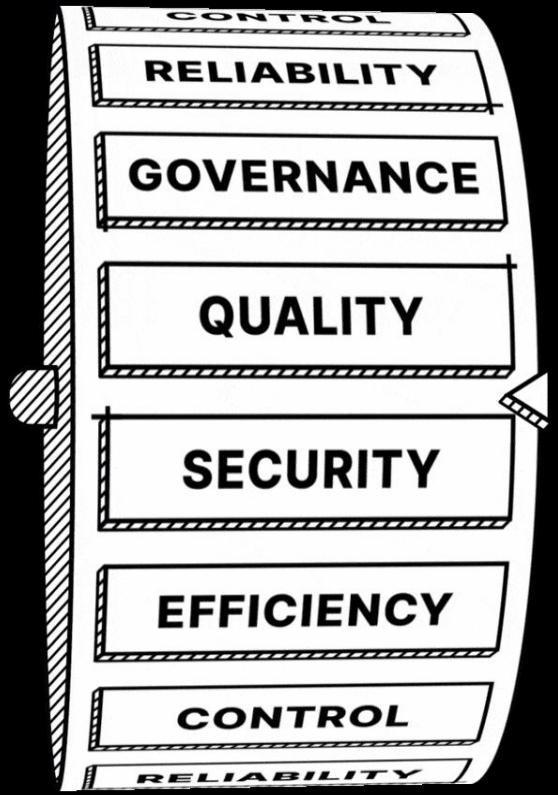
Head of Chaos Engineering at Harness
Co-founder of LitmusChaos and Maintainer

harness
The Platform for Software Delivery

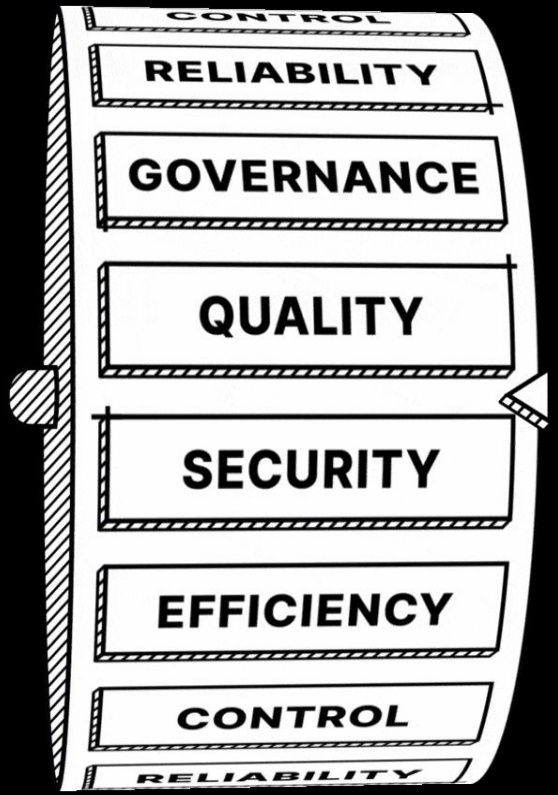
Harness
Chaos Engineering

Litmus

Innovation in software is a continuous process



Lets talk innovation in achieving Reliability or Resilience



The Cost of Software Development

27M

Software developers globally

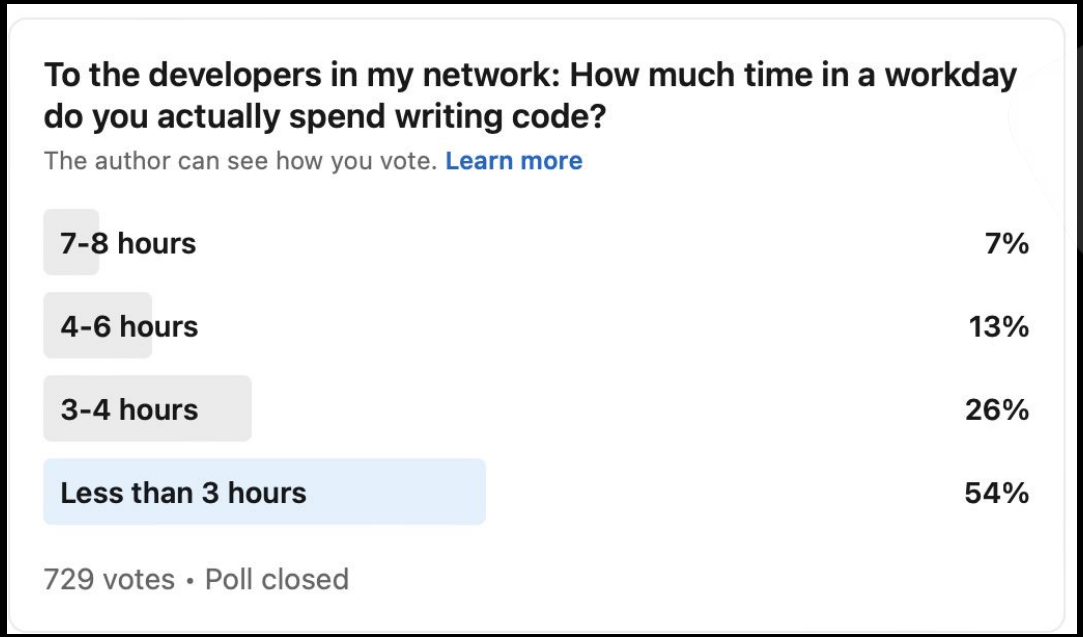
$100K

Average salary

$2.7T

Annual payroll

Developer Time Spent Coding



The Opportunity

$2.7T

Annual payroll

↓ 50%

Developer toil

$675B

Redirected to development

↑ 25%

Developer budget

The Opportunity

Innovate to Increase Developer Productivity and Save Costs

Where can you increase developer productivity?

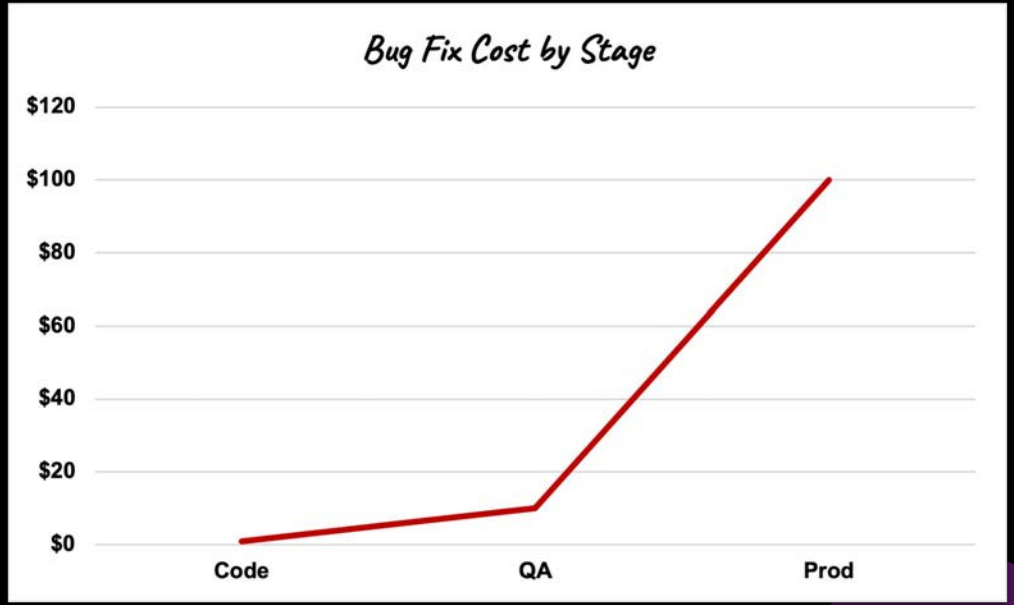
- Reduce Software Build Time
- Reduce Software Deployment Time
- **Reduce Software Debug Time**

Why do Developers spend more time in Debugging?

- Oversight
- Dependencies have not been tested
- Lack of understanding of the product architecture
- Code RAN in a new environment

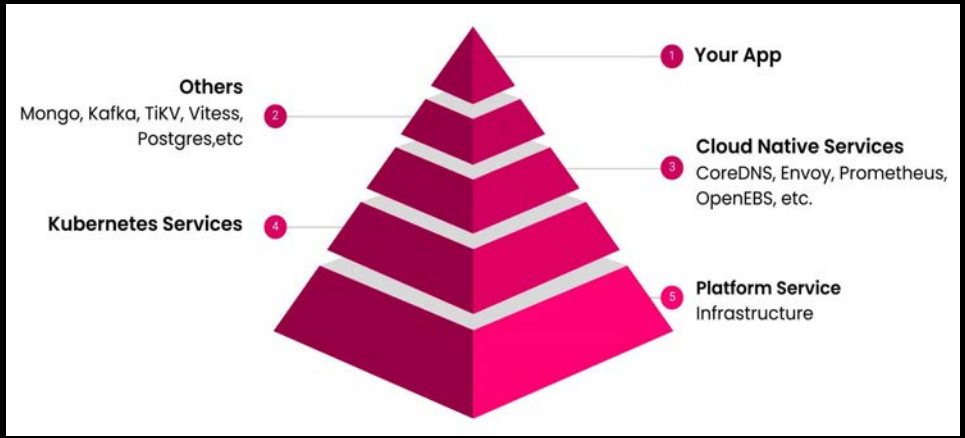
Cost of Debugging

Dev productivity is still okay if the debugging is happening in QA and Pre-Prod



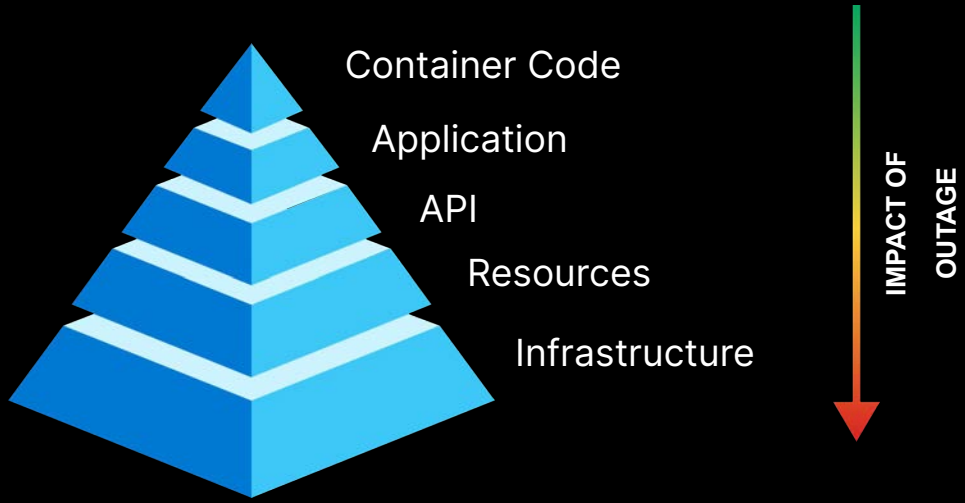
Revisit cloud native developers

Developers focus within the containers and it's APIs

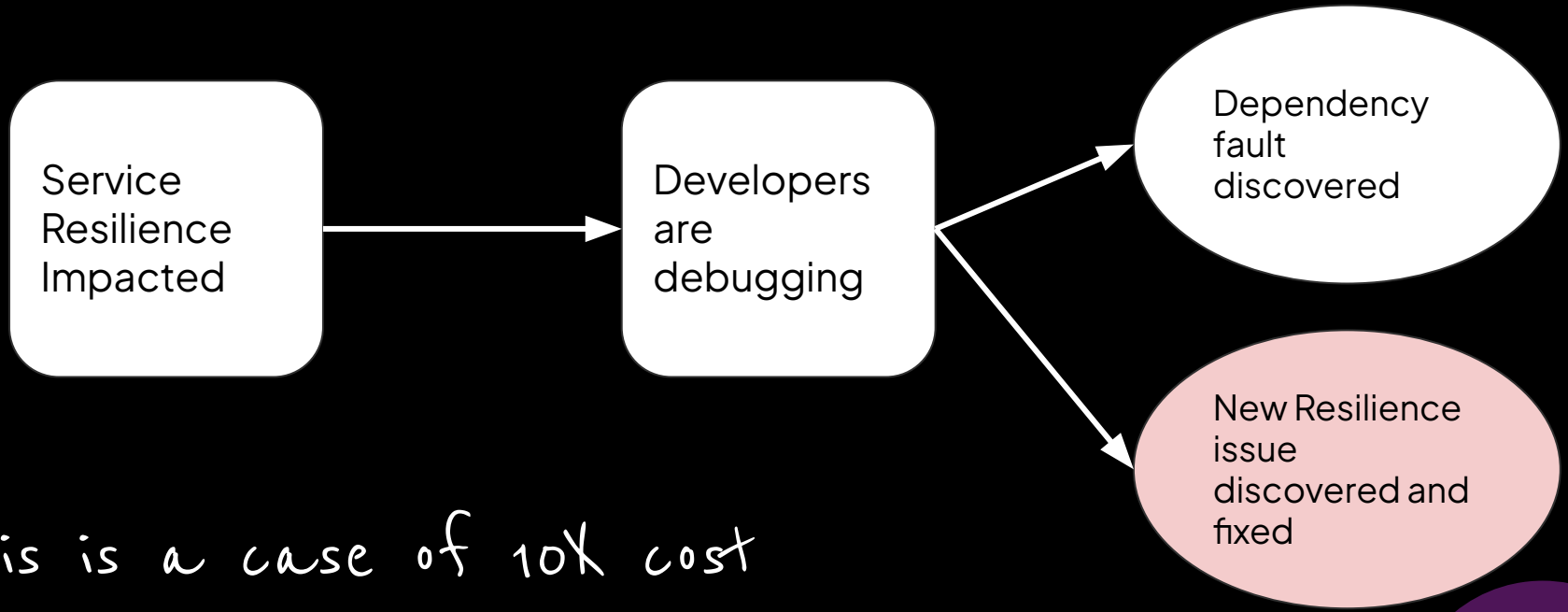


Impact of outage

Containers are tested for functionality. What if there are faults occurring in deep dependencies?



Faults in deep dependencies can cause debug time

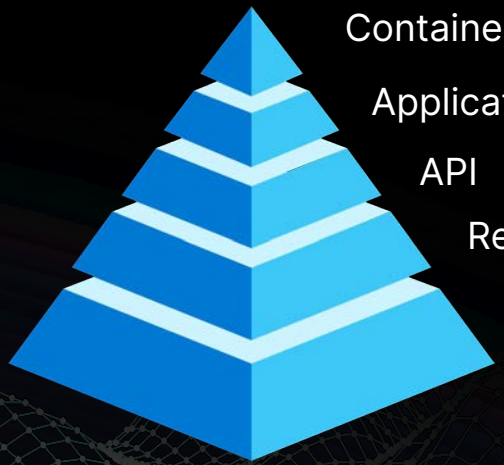


This is a case of 10X cost

Dependent fault testing is a need in cloud native

Test your code for faults happening in

- Within the code
- It's API consumption
- It's external resources
- It's dependent Infrastructure



This means

Cloud Native Developers

Need to do

Chaos Testing

Revisit the famous use case of Chaos Engineering

Introduce controlled faults to reduce expensive outages

Revisit the famous use case of Chaos Engineering

Introduce controlled faults to reduce expensive outages

- Recommends Production Chaos Testing
- Very high barrier
- Game Day Model

Traditional Chaos Engineering has been

- A reactive Approach
- (Or) Driven by regulations, e.g: Banking

New patterns of adoption of Chaos Engineering is driven by

- The need to increase developer productivity
- The need to increase quality in cloud native environments
- The need to guarantee Reliability in the move to cloud native

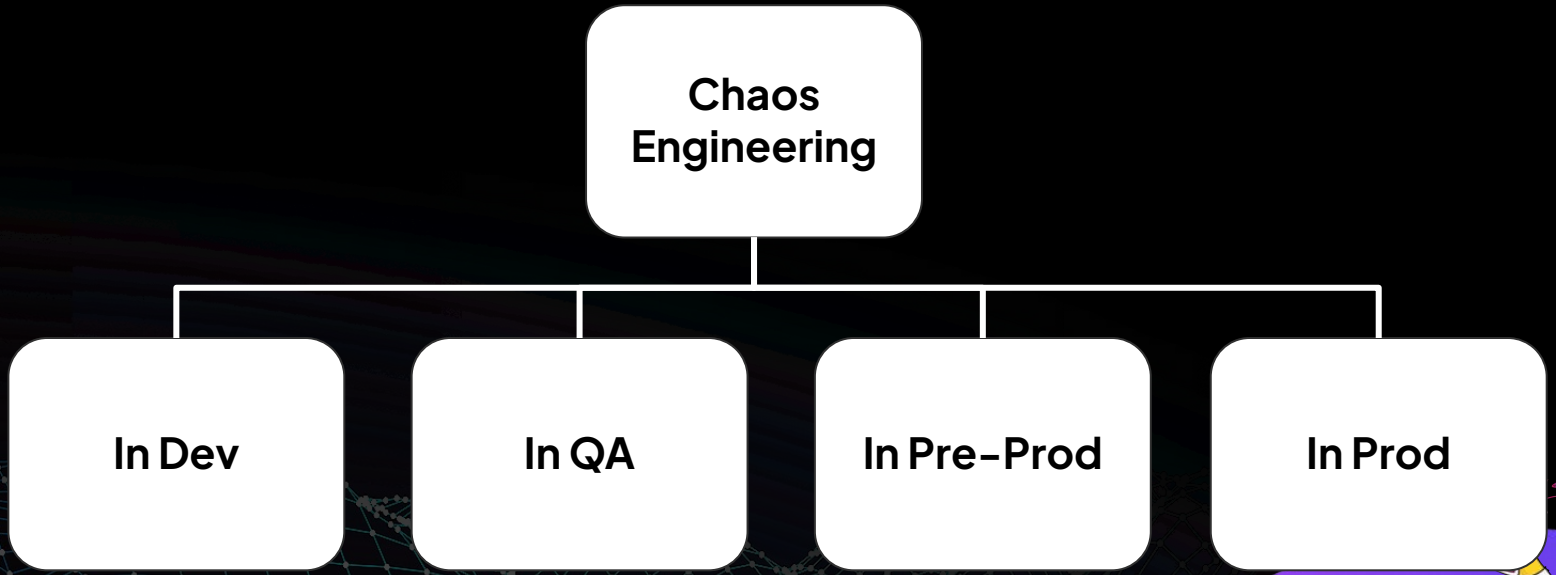
These needs lead to the emergence of the new concept

CONTINUOUS RESILIENCE

What is Continuous Resilience?

Verifying **R**esilience through
Automated **C**haos **T**esting
Continuously

Continuous Resilience is



Continuous Resilience Metrics

Resilience Score

The average success % of steady state of a given experiment or a component or a service

Resilience Coverage

$$\frac{\text{The number of chaos tests executed}}{\text{Total number of possible chaos tests}} \times 100$$

Continuous Resilience approach can also be seen as a pipeline approach

Gameday Approach Vs Pipeline Approach

Chaos via Game Days	Chaos via Pipelines
Chaos Experiments are executed on demand and with a lot of preparation	Chaos Experiments are executed continuously and without much preparation
Primarily targeted towards SRE as a persona	All personas are executing the chaos experiments
Adoption barrier is very high	Adoption barrier is much less

Traditionally … Developing Chaos Experiments

- Is a challenge → Code is always changing → Bandwidth is not budgeted
- The responsibility is typically not identified, SREs are usually pulled in into incidents and corresponding action tracking.
- Is not tracked to completion. No idea how many more to develop.

With Continuous Resilience Approach Developing Chaos Experiments is

- Is a team sport. Typically it is attributed to an extension of regular tests.
- Chaos Hubs or Experiment repositories are maintained as code in git.
- You know exactly how many more tests need to be completed, because you have the resilience coverage metric.

Continuous Resilience Demo

Summary

- Resilience is a real challenge in the modern or cloud native systems because of its nature.
- Use Chaos Experimentation to get ahead of the resilience challenge.
- Push chaos experimentation as a dev culture into the org rather than a game day culture. Dev culture approach makes it easy and scalable.

Thank You

Reach out to me at
@uma_mukkara

harness
The Platform for Software Delivery

Harness
Chaos Engineering