KubeCon | CloudNativeCon

North America 2023

KubeCon | CloudNativeCon

North America 2023

Unveiling the Impact of Chaos Litmus Probes on SLOs: Enhancing Resilience in Kubernetes Environments

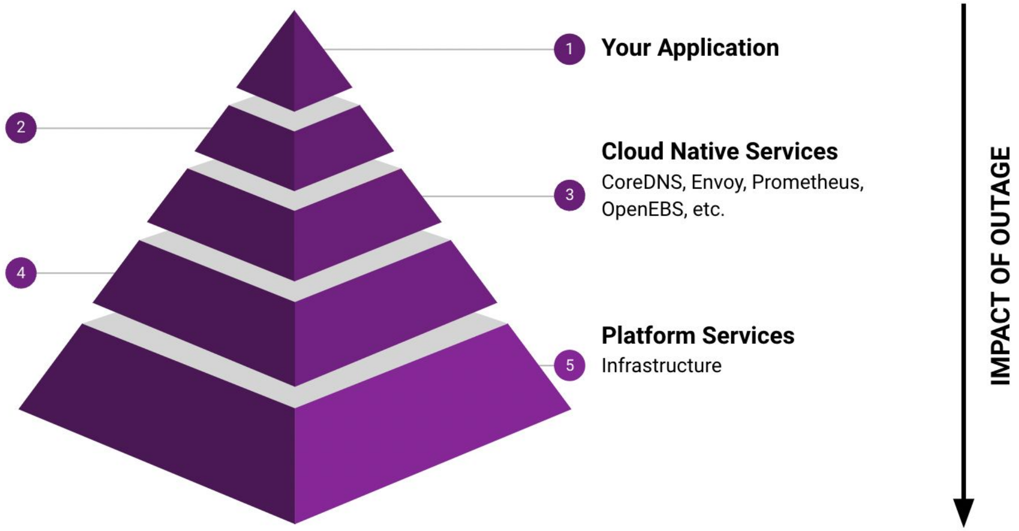
Amit Kumar Das
Senior Software Engineer, Harness
Maintainer, LitmusChaos

Saranya Jena
Senior Software Engineer, Harness
Maintainer, LitmusChaos

What is Chaos Engineering?

Chaos Engineering is the discipline to identify weakness in a system through controlled experiments that introduce random and unpredictable behavior.

The main concept behind chaos engineering is to break a system on purpose to collect information that will help improve the system's resiliency. Chaos engineering is an approach to software testing and quality assurance. It is well suited to modern distributed systems and processes.



1 Your Application
2 Application Dependencies
MongoDB, Kafka, TiKV, Vitess, Postgres, etc.
3 Cloud Native Services
CoreDNS, Envoy, Prometheus, OpenEBS, etc.
4 Kubernetes Services
5 Platform Services
Infrastructure
IMPACT OF OUTAGE

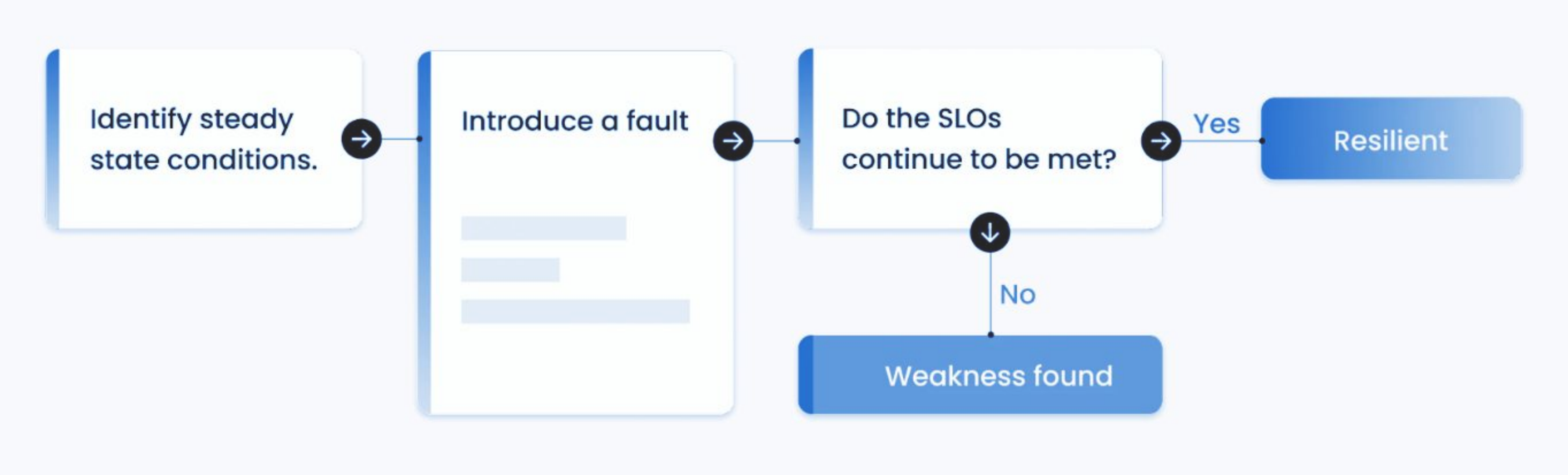
> 90% of Resilience of your application depends on-

Other Cloud-Native components and the infrastructure

Steady State Hypothesis and Resilience

To specifically address the uncertainty of distributed systems at scale, Chaos Engineering can be thought of as the facilitation of experiments to uncover systemic weaknesses. These experiments follow four steps:

- Start by defining ‘steady state’ as some measurable output of a system that indicates normal behavior.
- Hypothesize that this steady state will continue in both the control group and the experimental group.
- Introduce variables that reflect real world events like servers that crash, hard drives that malfunction, network connections that are severed, etc.
- Try to disprove the hypothesis by looking for a difference in steady state between the control group and the experimental group.



Identify steady state conditions.

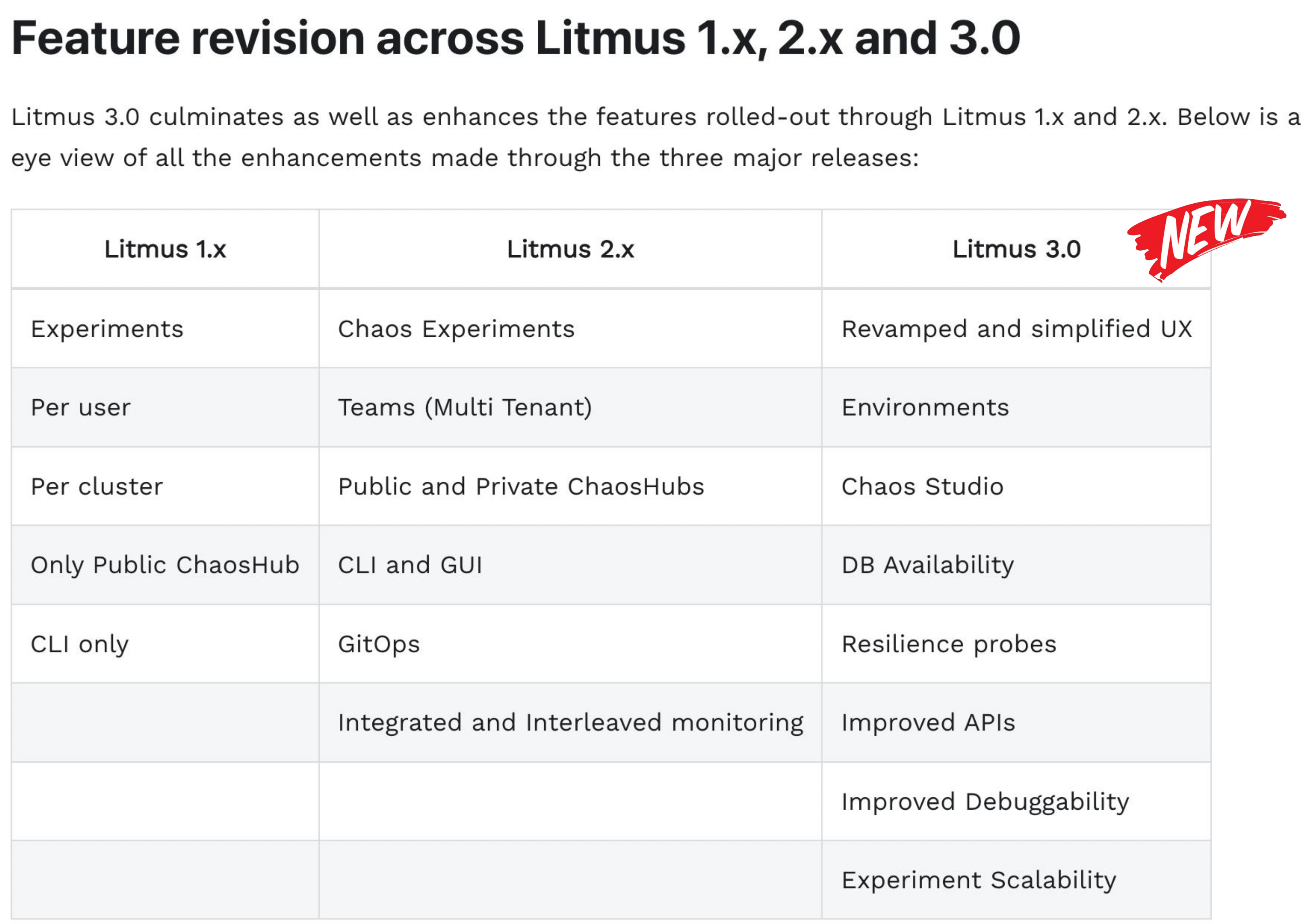
Introduce a fault

Do the SLOs continue to be met?

Achieving resiliency with Litmus 3.0

- Redefined User Experience
- Introduction to Environments
- Chaos Studio
- Resilience Probes
  - Resilience probes now support a plug-and-play architecture.
  - Users can create probes once and utilize them repeatedly across various experiments.
  - Comprehensive support for steady-state validation enhances system resilience.
  - Helps users assess the robustness of their applications more effectively.
- MongoDB High Availability Support
- Terminology Changes
- API Refactoring and Enhanced Code Architecture

Why Litmus 3.0?



Feature revision across Litmus 1.x, 2.x and 3.0

Litmus 3.0 culminates as well as enhances the features rolled-out through Litmus 1.x and 2.x. Below is a bird's eye view of all the enhancements made through the three major releases:

Litmus 1.x	Litmus 2.x	Litmus 3.0
Experiments	Chaos Experiments	Revamped and simplified UX
Per user	Teams (Multi Tenant)	Environments
Per cluster	Public and Private ChaosHubs	Chaos Studio
Only Public ChaosHub	CLI and GUI	DB Availability
CLI only	GitOps	Resilience probes
	Integrated and Interleaved monitoring	Improved APIs
		Improved Debuggability
		Experiment Scalability

Types of Litmus Probes

Litmus currently supports four types of Probes:

- **httpProbe**: To query health/downstream URIs
- **cmdProbe**: To execute any user-desired health-check function implemented as a shell command
- **k8sProbe**: To perform CRUD operations against native & custom Kubernetes resources
- **promProbe**: To execute promql queries and match prometheus metrics for specific criteria

The probes can be set up to run in different modes:

- **SoT**: Executed at the Start of Test as a pre-chaos check
- **EoT**: Executed at the End of Test as a post-chaos check
- **Edge**: Executed both, before and after the chaos
- **Continuous**: The probe is executed continuously, with a specified polling interval during the chaos injection.
- **OnChaos**: The probe is executed continuously, with a specified polling interval strictly for chaos duration of chaos

Demo