LitmusChaos Maintainer Session

Uma Mukkara
Project Maintainer
LitmusChaos

Speaker Name
Project Maintainer
LitmusChaos

Udit Gaurav
Project Maintainer
LitmusChaos

Saiyam Pathak
CNCF Ambassador
CIVO

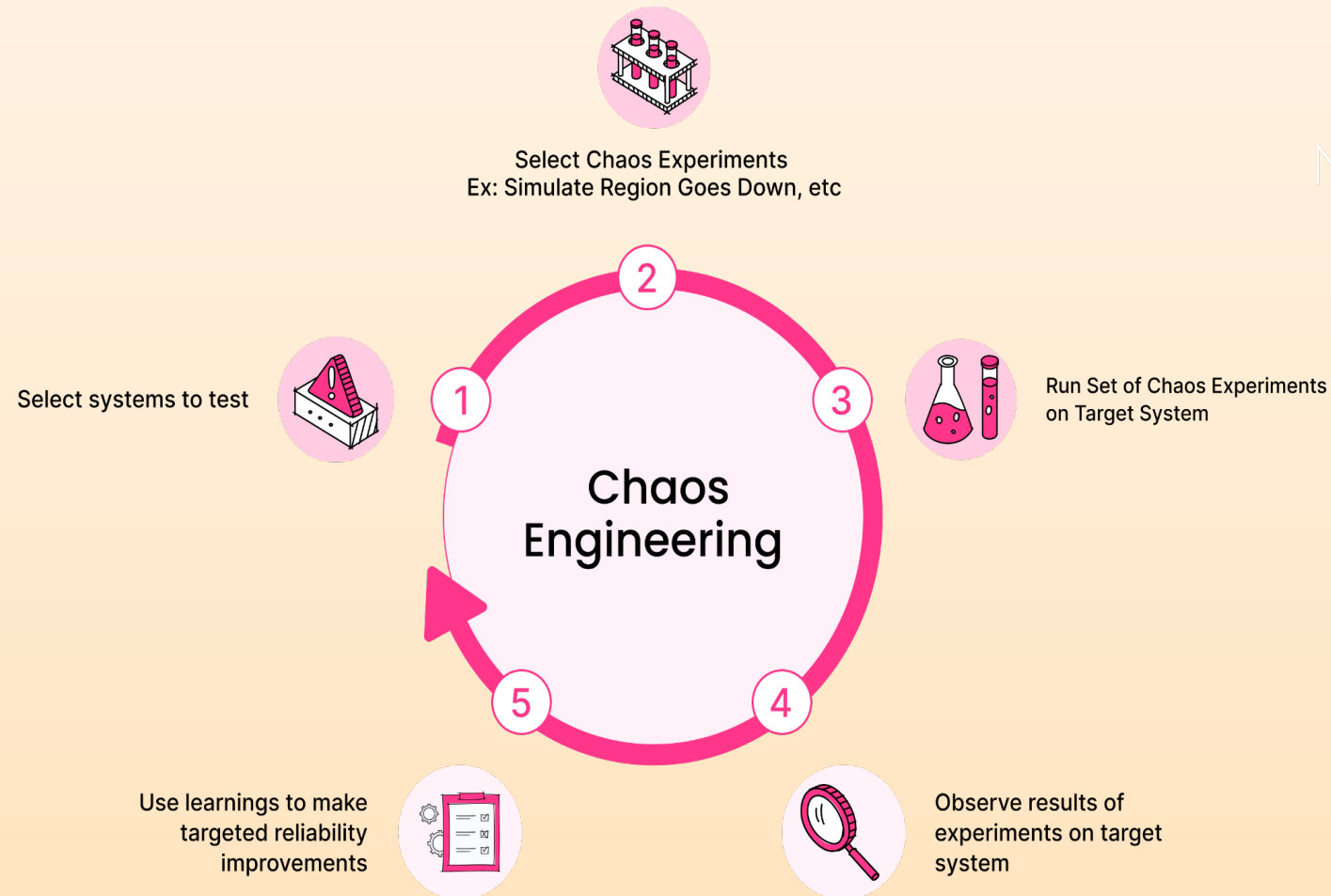# Cloud Native Chaos Engineering

- What

- Why

- How

What is Chaos Engineering

Select Chaos Experiments
Ex: Simulate Region Goes Down, etc

Chaos Engineering

Select systems to test

Run Set of Chaos Experiments on Target System

Use learnings to make targeted reliability improvements

Observe results of experiments on target system

Why…

Application Failures

- Excessive Logging to debug
- Too many retries
- Service Timeout

Infrastructure Failures

- Device failures
- Network failures
- Region not available

Operational Failures

- Capacity issues
- Incident management
- Monitoring dashboards not available

Reputational Impact

We've resolved the issue, but please note some features may take a bit longer for the fix to take effect. You may need to reload Slack (Cmd/Ctrl + Shift + R) to see the fix on your end. Apologies for the disruption!

Slack's Outages

Financial Impact

We want our customers to know that this is a contained issue affecting one of our facilities, and not due to any cybersecurity event. We apologize for the inconvenience caused by these system issues, and any Wells Fargo fees incurred as a result of these issues will be reversed.

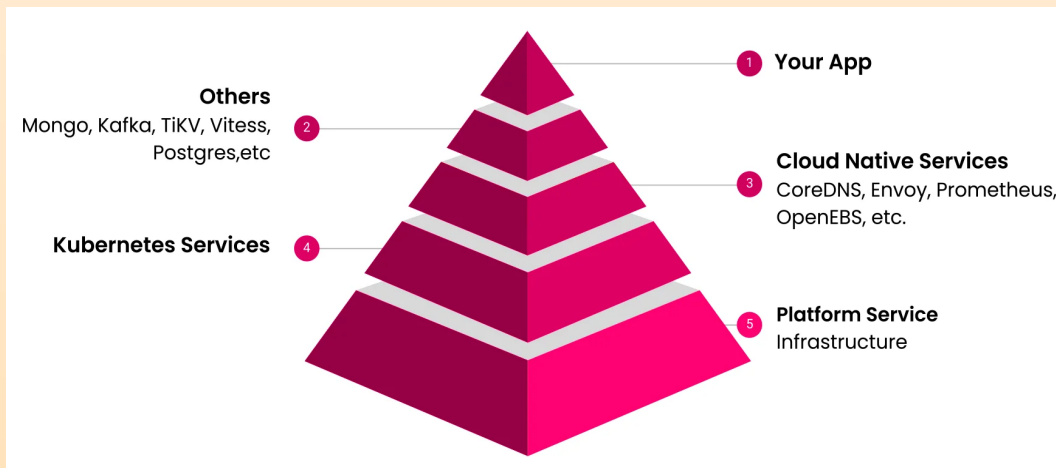Est. >$55M in losses to WF

Poor User Experience

I'm afraid we're currently experiencing some system issues at the airport this morning, Pip. We're doing all we can to resolve this and 1/2

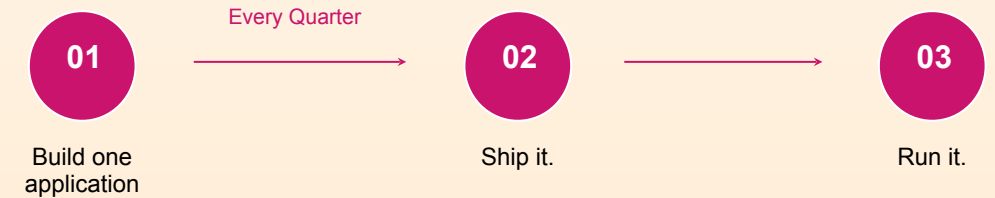75,000+ passengers travel plans impacted

Why...

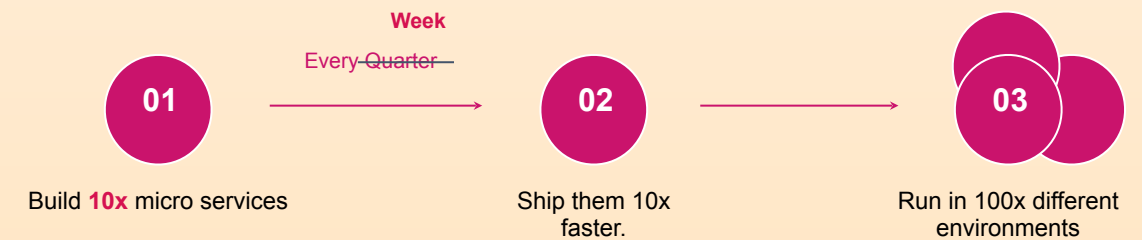## Proliferation of applications into micro services leads to a RELIABILITY challenge

In cloud native, your code depends on hundreds of other microservices and runs on many platforms. The potential of being subjected to a dependent component failure is huge.
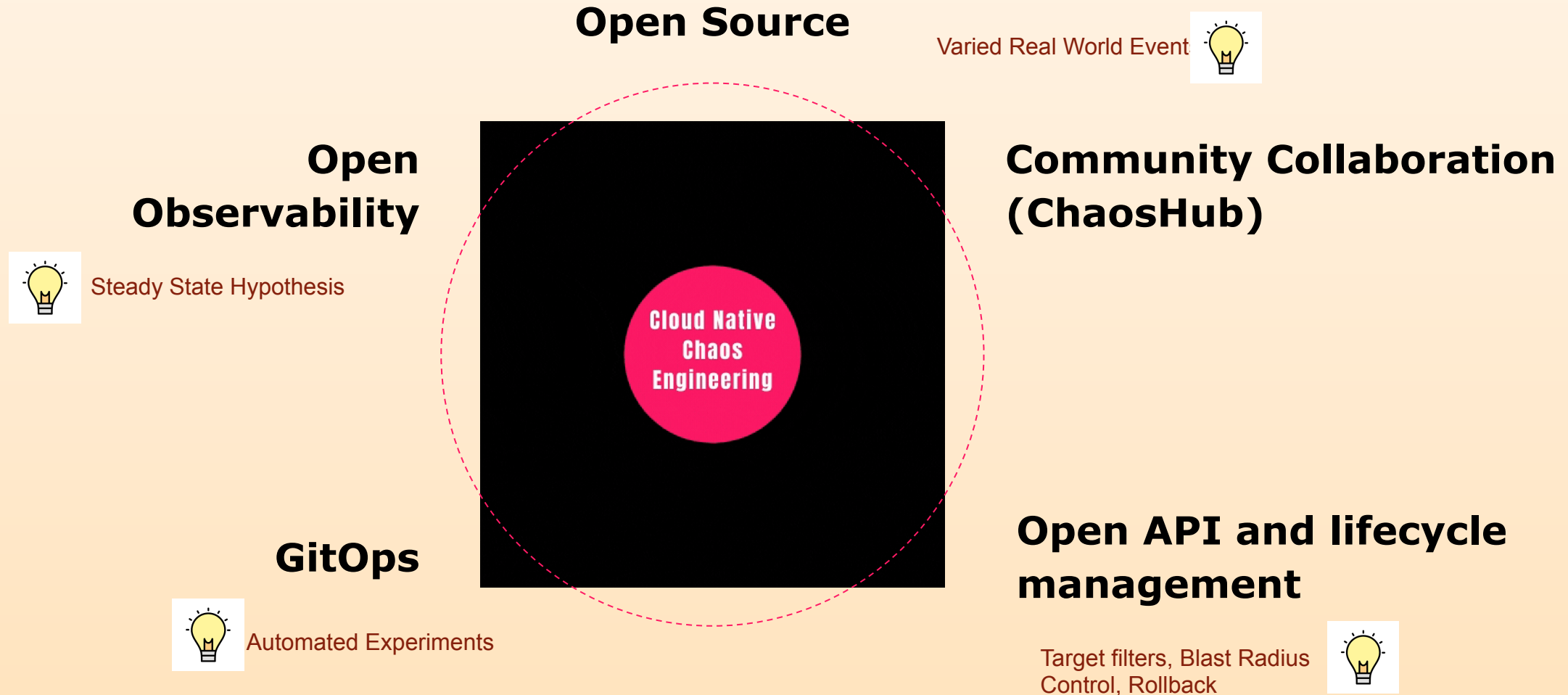
Legacy DevOps

01 — Every Quarter → **02** → **03**

Build one application   Ship it.   Run it.

Cloud native DevOps

01 — Week / Every Quarter → **02** → **03**

Build **10x** micro services   Ship them 10x faster.   Run in 100x different environments

Too many fault scenarios. Significant increase in service down potential because of a failure of a dependent service

1 **Your App**

2 **Others**
Mongo, Kafka, TiKV, Vitess, Postgres,etc

3 **Cloud Native Services**
CoreDNS, Envoy, Prometheus, OpenEBS, etc.

4 **Kubernetes Services**

5 **Platform Service**
Infrastructure

How...

Principles of Cloud Native Chaos Engineering

Open Source

Varied Real World Event

Open Observability

Steady State Hypothesis

Community Collaboration (ChaosHub)

Cloud Native Chaos Engineering

GitOps

Automated Experiments

Open API and lifecycle management

Target filters, Blast Radius Control, Rollback

How…

LitmusChaos Experiment Execution Backend: Requirements

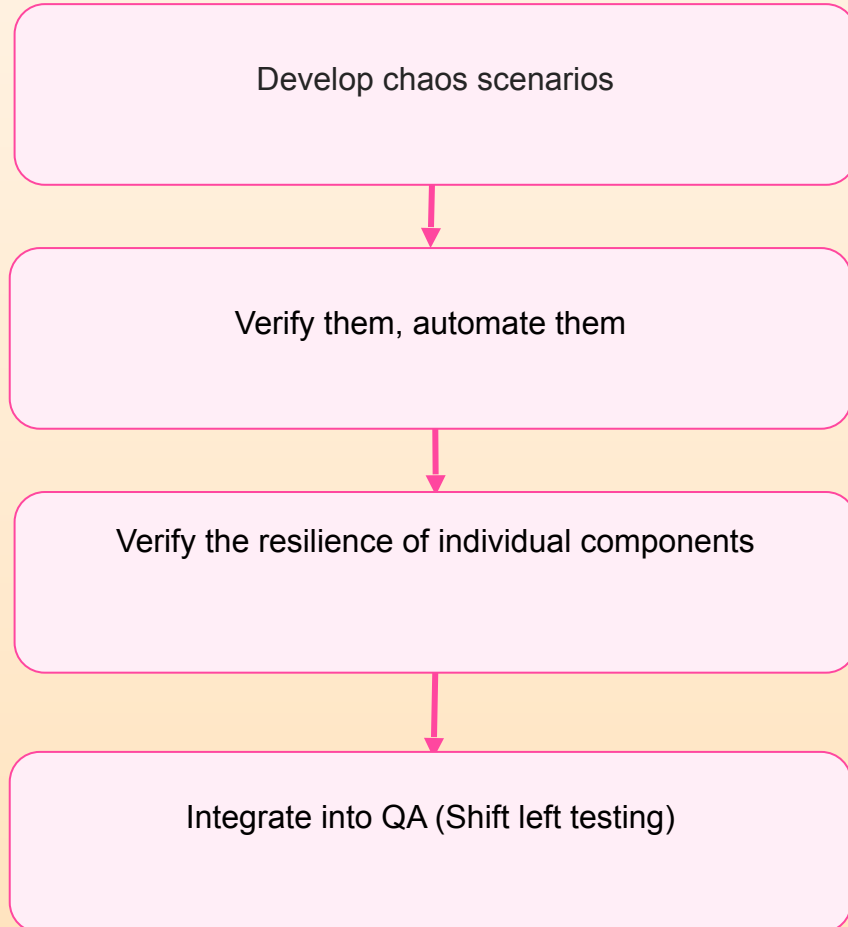| Attribute	Description	Components
Kubernetes native	Both the framework itself as well as supported targets	Chaos Operator, Chaos CRDs, Helm Chart
Declarative fault injection intent	Out-of-the-box fault types, easy-mapping to microservices	ChaosHub
Notion of an end-to-end, automated experiment	Principles of chaos: Blast radius control, varied failures, hypothesis validation, lends itself to automation	Experiment Categories, Litmus Probes Litmus Integrations
Experiment metadata as metrics	Correlate fault injection with application behavior	Litmus (Prometheus) Exporter
Extensible	Support non-native chaos libraries. Act as an orchestration engine	BYOC

How…

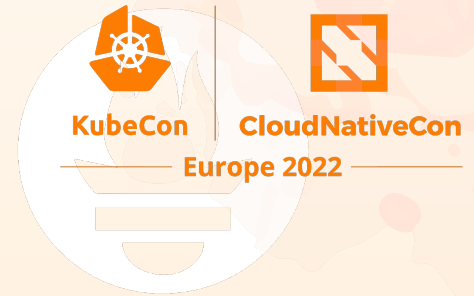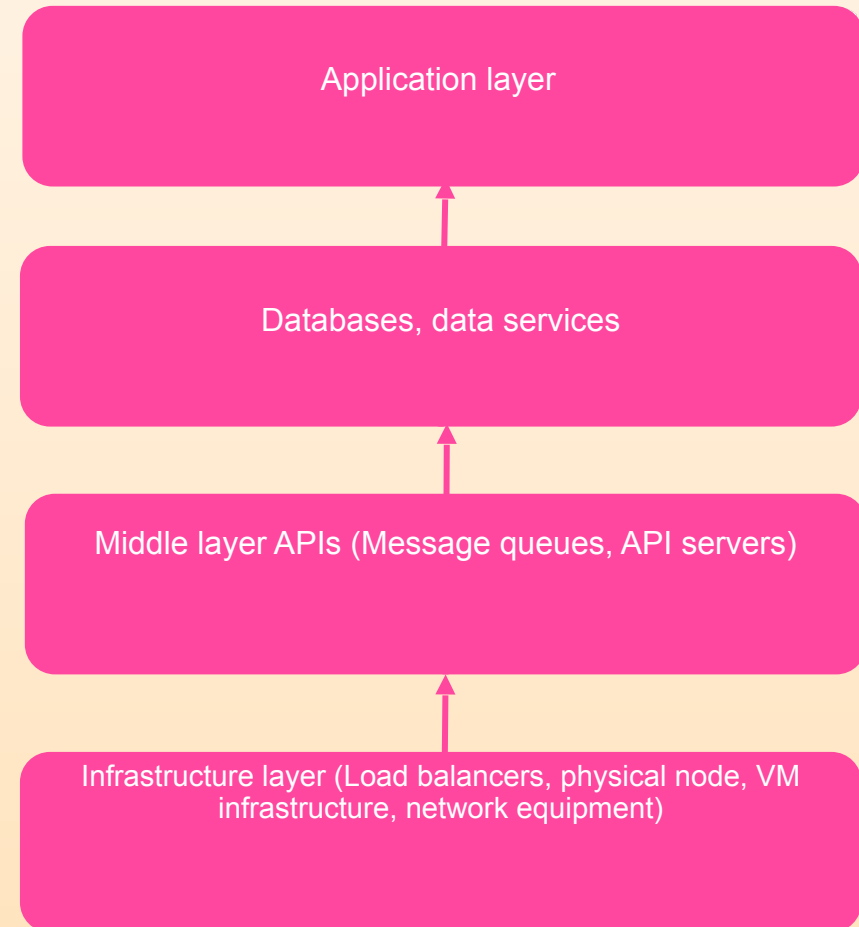## LitmusChaos Control Plane & Agents: Requirements

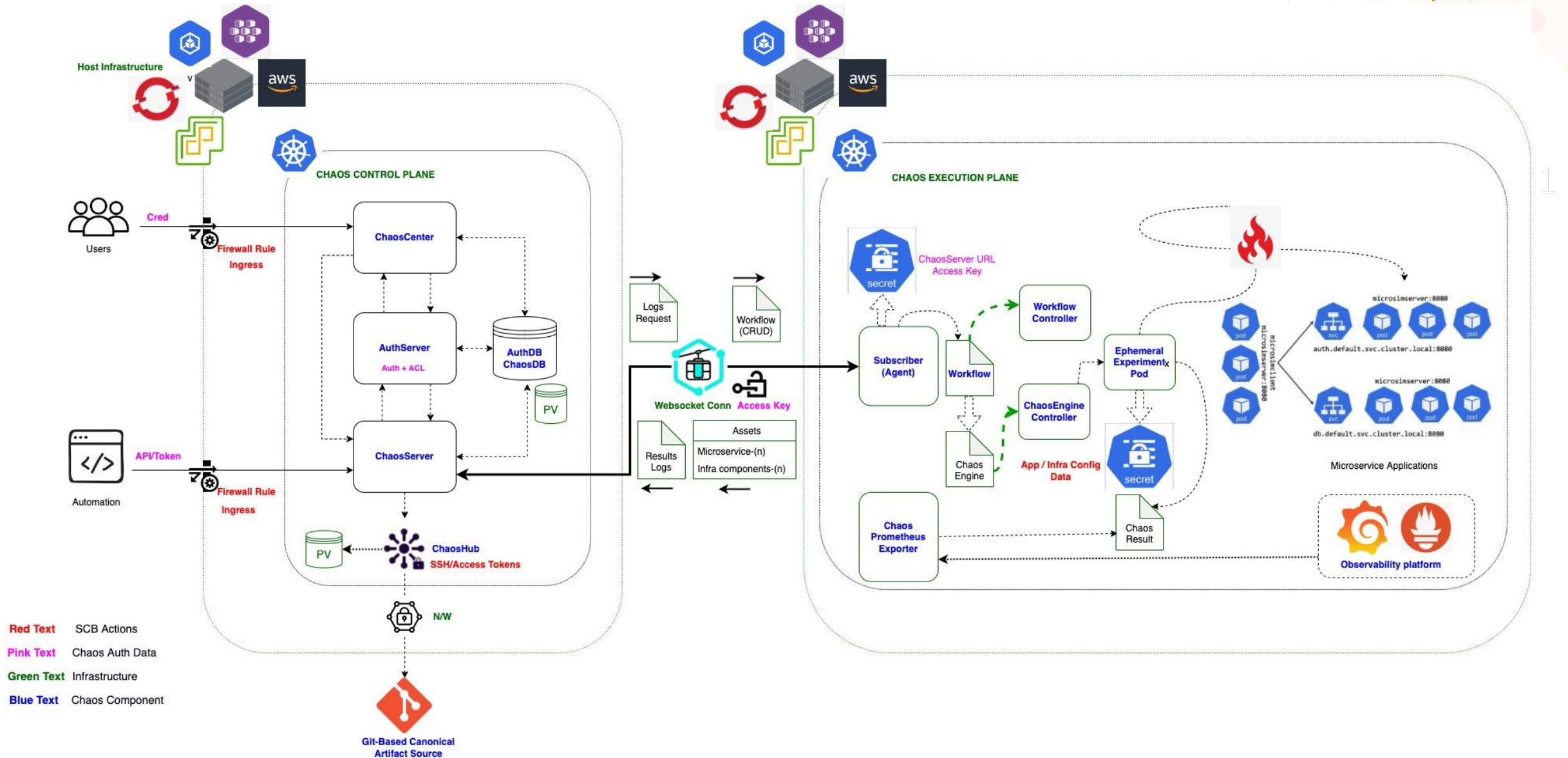| Attribute	Description	Components
Creation of Complex Scenarios	Stitch together faults in desired sequence/order	Litmus Workflows
Centralized chaos management	Chaos on a fleet of clusters	Chaos Center (server, subscriber, db)
Collaborative chaos experimentation	Users-Environment mapping & roles in chaos execution	Teaming Support
Single source of truth	Chaos artifact source, golden copies of workflows	Litmus GitOps
Chaos on multi-tenant clusters	Independent chaos instances on shared environments	Namespace Mode Support
Chaos Automation	Support for scheduled and event-triggered chaos	Cron Workflows, Subscription-based chaos

Chaos Maturity Model

HOW ?

WHERE ?

HOW ?
Develop chaos scenarios
↓
Verify them, automate them
↓
Verify the resilience of individual components
↓
Integrate into QA (Shift left testing)

WHERE ?
Application layer
↑
Databases, data services
↑
Middle layer APIs (Message queues, API servers)
↑
Infrastructure layer (Load balancers, physical node, VM infrastructure, network equipment)

# LitmusChaos Architecture

Roadmap

- Dedicated charts for agent/execution plane (auditable resource specs)
- Simplification of experiment orchestration (leaner components) (support for agent-lite)
- Namespace whitelisting (chaos targets)
- OpenTelemetry Support (metrics consumption standardization for chaos engg)
- Events dashboard & audit logs for improved observability
- Improvement of chaos-ci-lib to support 2.x (workflow APIs)
- Increased Kubernetes experiment list (IOChaos, TimeChaos)
- Improved test (unit, e2e) coverage, reporting with interoperability matrix
- Improved litmus-sdk for simpler experiment bootstrap & testing
- Better documentation (monitoring, OPA gatekeeper policies, tutorials)
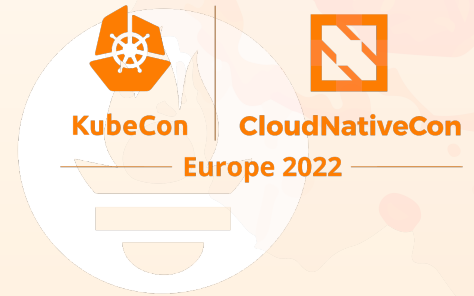
Chaos Use case in CIVO - User Story
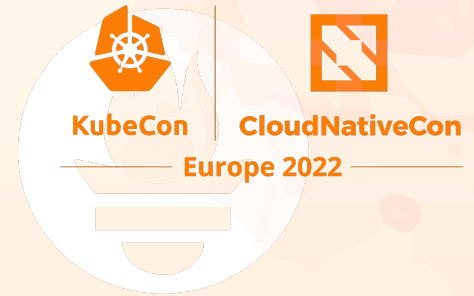
Q/A

Thank You!!!