# Automating Resiliency via Chaos Engineering

June 20, 2019

Balaji Arunachalam, Director of Engineering

Shan Anwar, Staff Software Engineer

# Agenda

Introduction

Outages and their cost

Case for change at Intuit

Resiliency testing journey at Intuit

CloudRaider deep dive

Demo

Q/A

# Intuit Inc.


intuit
powering
prosperity
intuit turbotax.
intuit quickbooks.
intuit proconnect.
intuit mint.

# Balaji Arunachalam

B.E, Computer Science

M.S, Computer Science

Software Developer

Software Developer

**Director, Engineering**

QUALITY ENGINEERING

Developer Productivity



SRE

# Shan Anwar

B.S Computer Science

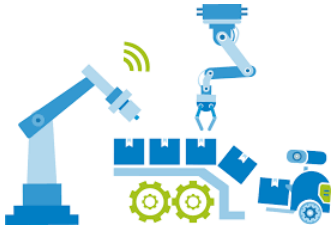
Software Engineer

**Staff Software Engineer**

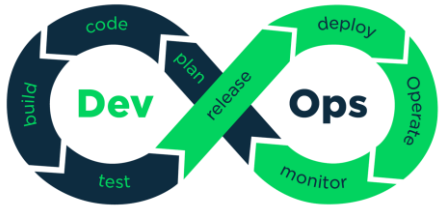
Java

QUALITY ENGINEERING

Developer Productivity



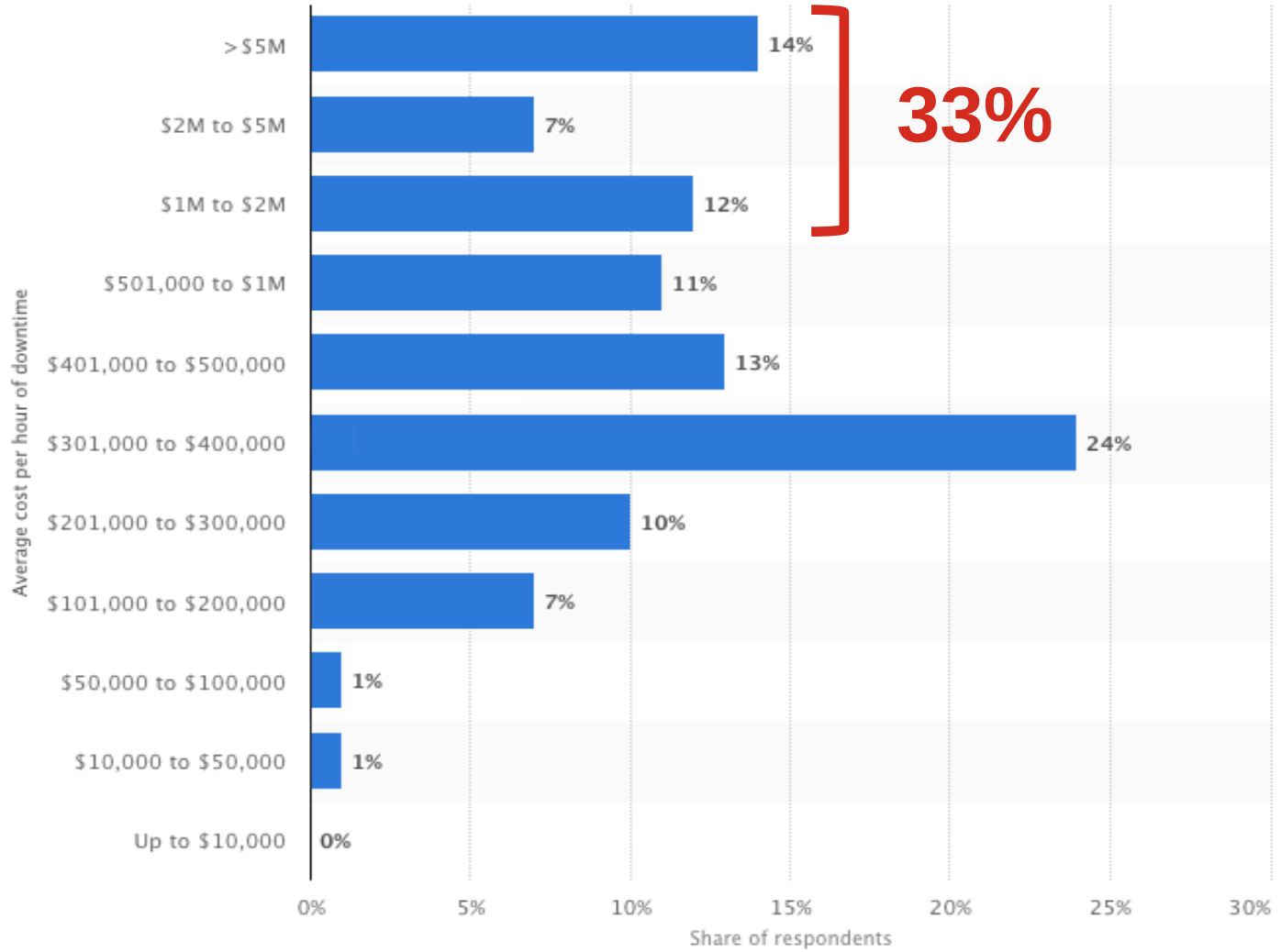
Tools Automation




Site Reliability Engineering

# Hourly cost of an outage

33% of companies pay +$1M per hour of outage




Source: Statisa

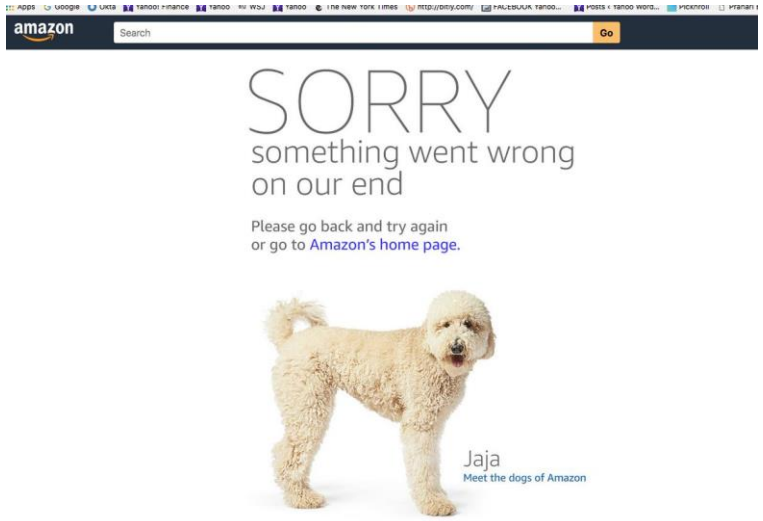
**2017 Gartner Report**

# Chaos around the world

**IRS**
Delayed tax returns process

**IRS provides additional day to file and pay for taxpayers through Wednesday, April 18; IRS processing systems back online**

**Amazon Prime Day**
Lost $90M in 1 day

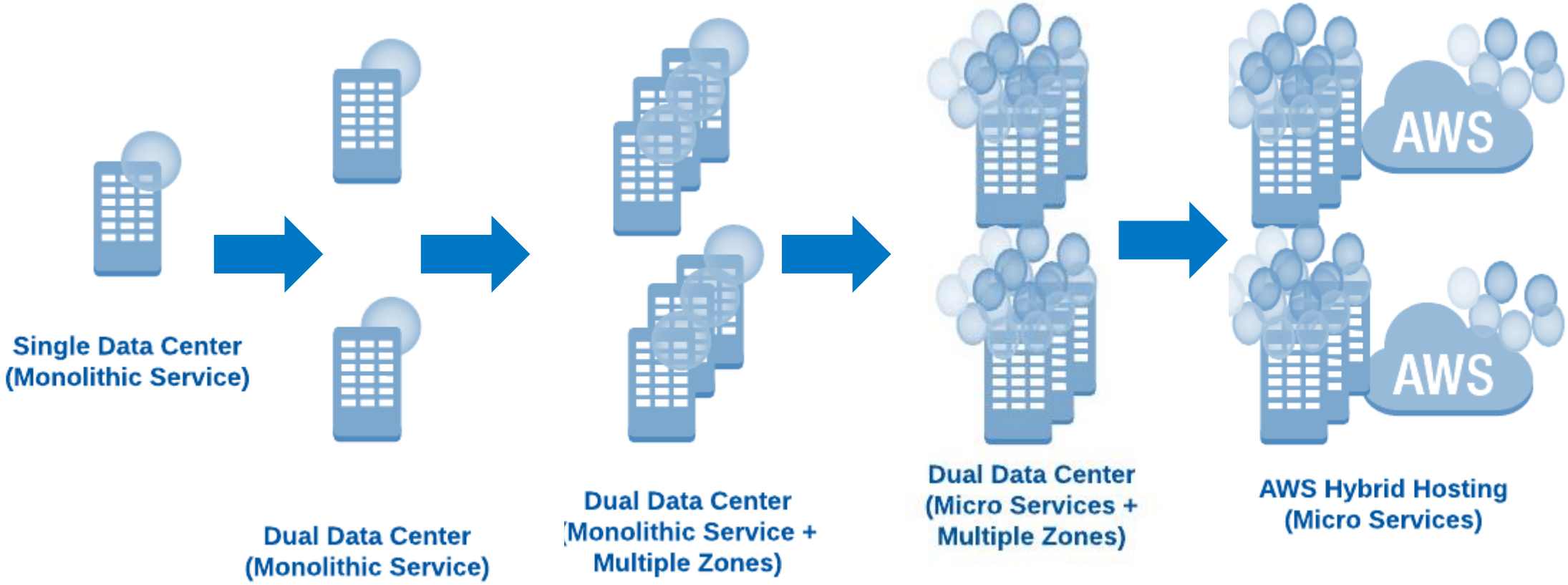



**Sabre Outage**
Shut down 100 airports & 300 airlines for 4 hours


**Sutter Health Outage**
Prevented access to patient health records for 1 day

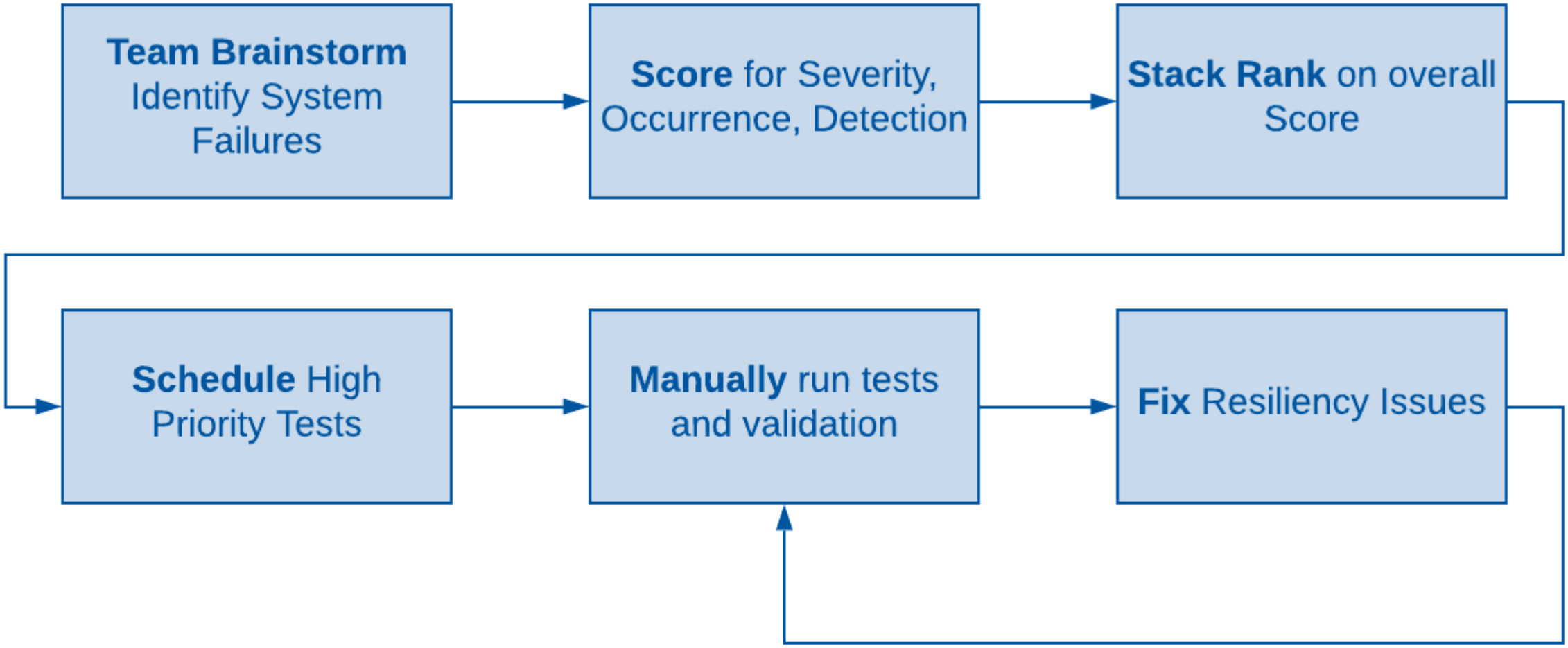
# Case of Change at Intuit

Resiliency test complexity increases as system complexity increases…



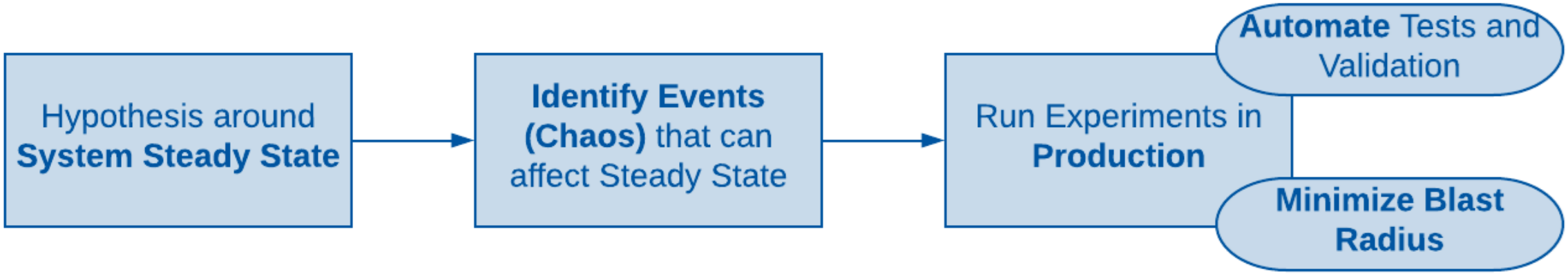
# Before chaos engineering

FMEA (Failure Mode Effect Analysis), since 1950s



# Chaos engineering

Very similar but not the same…



# What was missing in resiliency testing?

## FMEA testing helped but it is…

- Manual and laborious
- Too expensive
- After-thought
- Too specialized

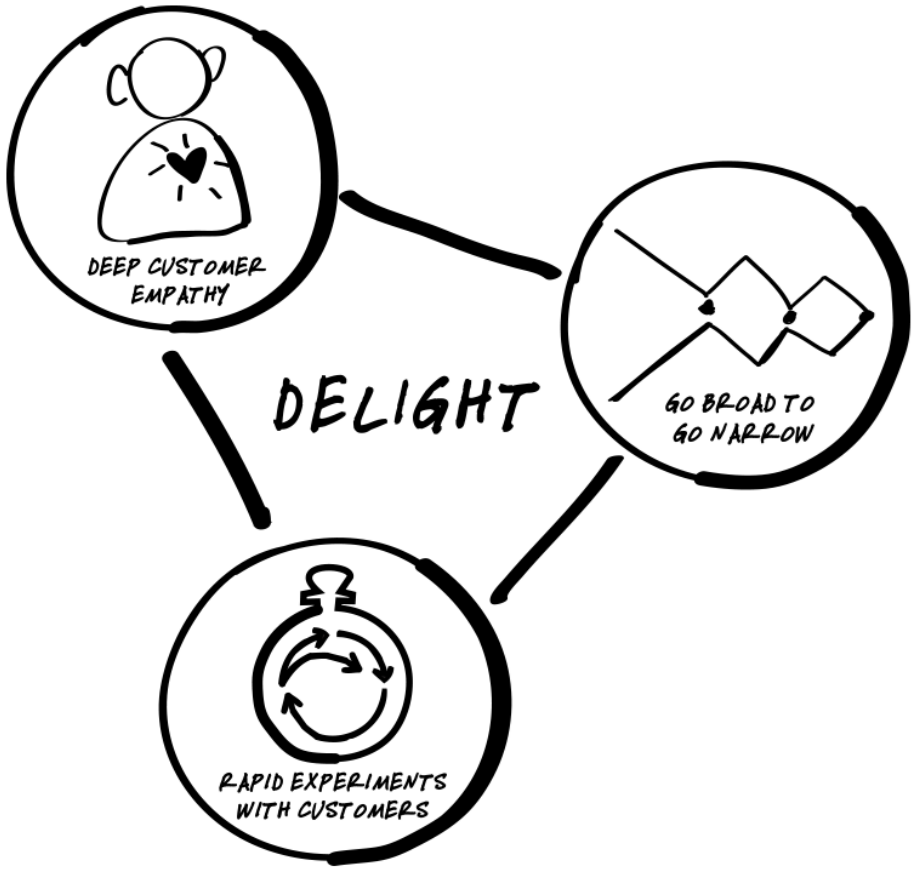
## Chaos testing helped but it is…

- Unstructured and ad hoc
- Happens later
- Prevents resiliency culture
- Uncomfortable for teams
- No regression testing

# Requirements to find an optimal solution…

| | |
|---|---|
| **Earlier in SDLC** | Resiliency testing to start as part of system design |
| **Shift left testing** | Shift left resiliency testing to developers to scale and enable test driven development |
| **Automation is a must** | Tests to be automated as part of release pipeline |
| **Natural language** | Test cases to be used as the test plan |
| **Serves as a prerequisite for chaos engineering** | 100% regression test pass is required before chaos testing in production |

# Automation journey via Intuit D4D (Design 4 Delight)

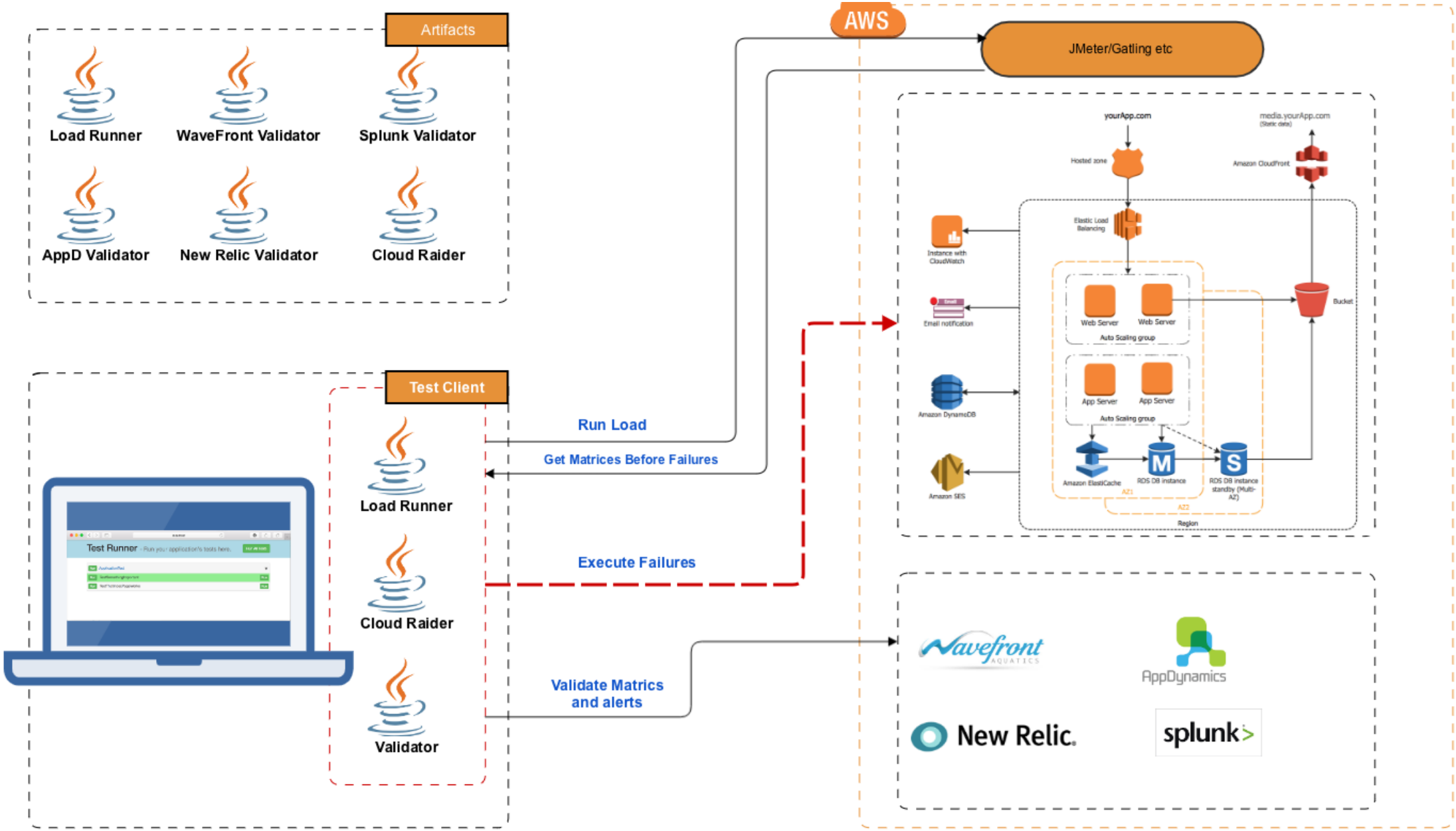



cloud raider

# CloudRaider key benefits for Intuit

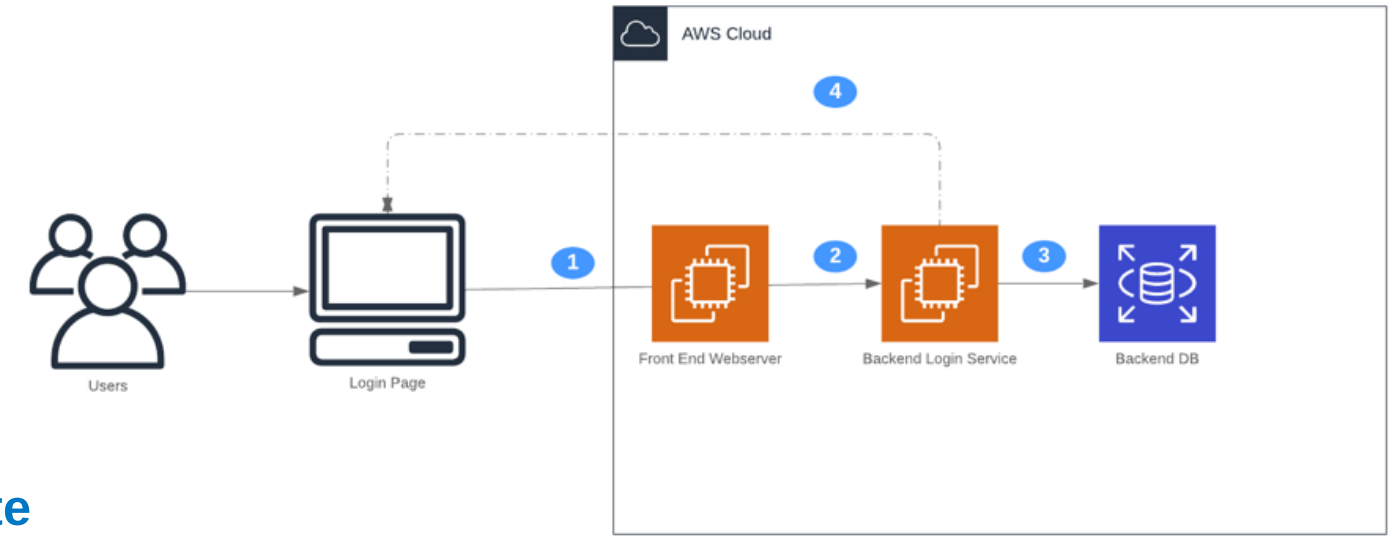
- Natural language construct
- Controlled failures in AWS (EC2, ALB, Route53, S3, RDS, DynamoDB, Elasticache etc.)
- Reduced execution time from days to hours
- No human interaction
- Early bugs/failures detection
- Extensible
- Open-source Java/Cucumber library https://github.com/intuit/CloudRaider/

# FMEA workflow




intuit

# Simple login service walkthrough




## FMEA template

| What is the Process? | What is the Potential Failure Mode? | What are the Potential Effects of the Failure Mode | Severity | What are the Possible Causes of the Failure? | Occurrence | How Do We Currently Prevent Each Listed Cause of Failure from Happening? | Detection | S | O | D | RPN |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Customer Sign in to website | Site Not Reachable | Customers unable to login | 8 | Webserver is unavailable | 4 | Alerts & Monitors | 3 | 8 | 4 | 3 | 96 |
| | | | | Network time outs | | Auto scale | | | | | |
| | | | | High CPU on servers | | Page on call | | | | | |
| | | | | Low Memory on Servers | | | | | | | |
| | | | | Disk Full on servers | | | | | | | |
| | | | | Application stopped | | | | | | | |

# CloudRaider code (instance failure)

**Feature:** Instance Termination

@terminationInjection
**Scenario Outline:** Simple Login Service Unreachable due to server failures.
Validate that alarms were triggered before recovery.

**Resources** →
**Given** EC2 < *ec2Name* >
**And** CloudWatch Alarm < *alarmName* >

**Injecting Failure** →
**When** terminate all instances
**And** wait for < *wait* > minute

**Validations** →
**And** assertCW alarm = <*state1*>
**And** assertEC2 healthy host count = < *expected-count* >
**And** assertCW alarm = <*state2*>

@dev
**Examples:**

**Data Driven** →

| *ec2Name* | *alarmName* | *wait* | *expected-count* | *state1* | *state2* |
|---|---|---|---|---|---|
| "login-frontend" | "login-frontend-UnHealthyHosts" | 1 | 1 | "ALARM" | "OK" |
| "login-backend" | "login-backend-UnHealthyHosts" | 1 | 1 | "ALARM" | "OK" |

# CPU spike

**Feature:** CPU Spike

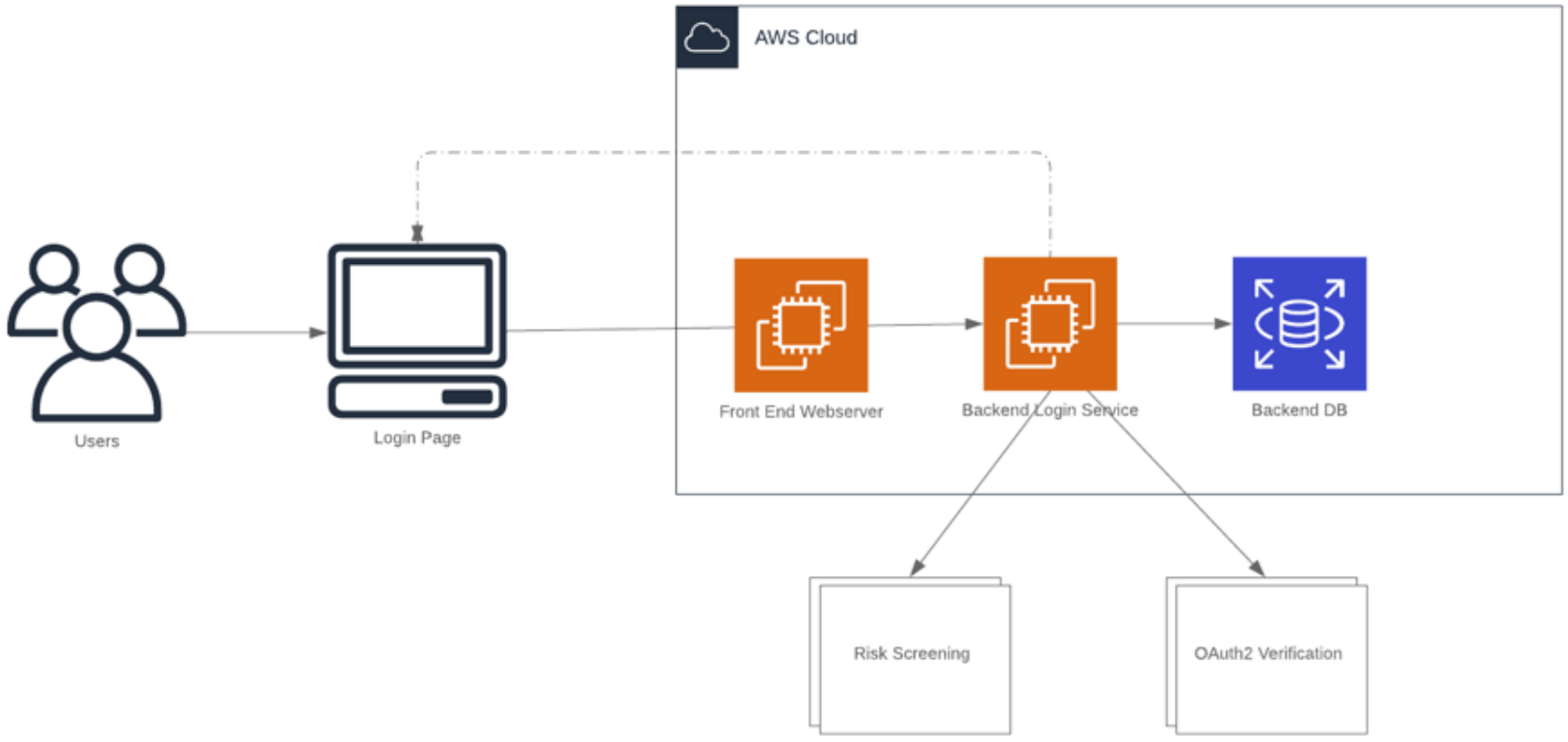
@cpuspike
**Scenario Outline:** Simple Login Service Unreachable due to server resource constraints. Validate that alarm triggers before recovery.

**Given** EC2 < *ec2Name*>
**And** CloudWatch Alarm <*alarmName*>
**When** CPU spike on <*instanceCount*> instances for <*coresCount*> cores
**And** wait for <*wait*> minute
**And** assertCW alarm = <*state1*>
**And** recover
**And** assertEC2 healthy host count = < *instanceCount*>
**And** assertCW alarm = <*state2*>

@dev
**Examples:**

| *ec2Name* | *alarmName* | *wait* | *instanceCount* | *state1* | *state2* | *coresCount* |
|---|---|---|---|---|---|---|
| "login-frontend" | "login-frontend-UnHealthyHosts". | 1 | 1 | "ALARM". | "OK" | 4 |
| "login-backend" | "login-backend-UnHealthyHosts" | 1 | 1 | "ALARM" | "OK" | 4 |

# Extending the example (more dependencies)



# Dependency failure

**Feature:** Block downstream dependencies

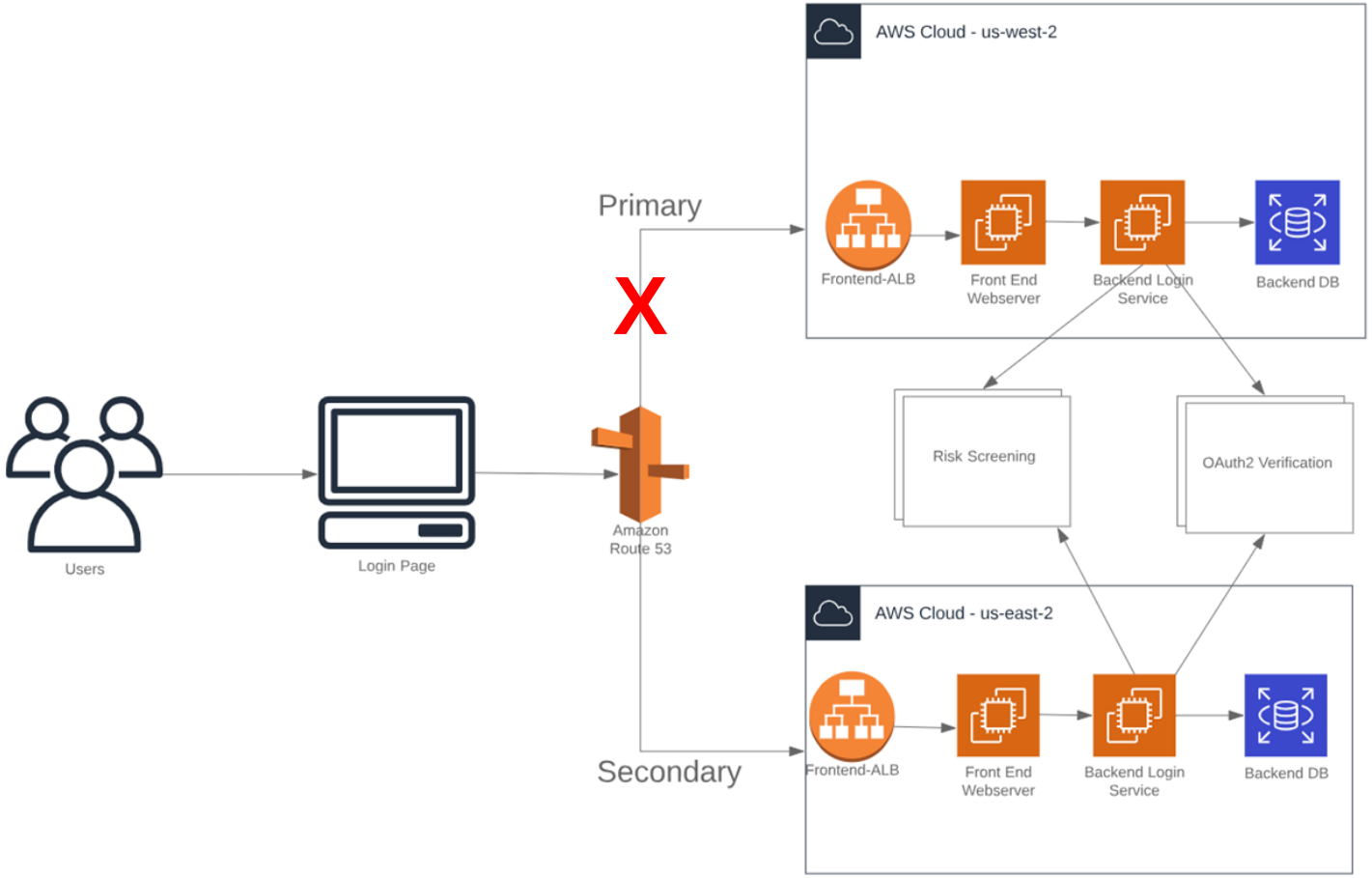
@blockdepndency
**Scenario Outline:** Given Simple Login Service, block critical dependency to validate application resiliency (Circuit Breakers etc)

**Given** EC2 <ec2Name>
**And** CloudWatch Alarm <*alarmName*>
**When** block domain <*domainName*> on <instanceCount> instances
**And** wait for <*wait*> minute
**And** assertCW alarm = <*state1*>
**And** recover
**And** assertEC2 healthy host count = <*expected-count*>
**And** assertCW alarm = <*state2*>

@dev
**Examples:**

| *ec2Name* | *alarmName* | *wait* | *instanceCount* | *state1* | *state2* | *domainName* | *expected-count* |
|---|---|---|---|---|---|---|---|
| "login-frontend" | "login-backend-UnHealthyHosts" | 1 | 1 | "ALARM" | "OK" | "riskscreening.com" | 1 |
| "login-backend" | "login-backend-UnHealthyHosts" | 1 | 1 | "ALARM" | "OK" | "oauthverifier.com" | 1 |

# Extending further (multi regional failover)



# Failover Scenario

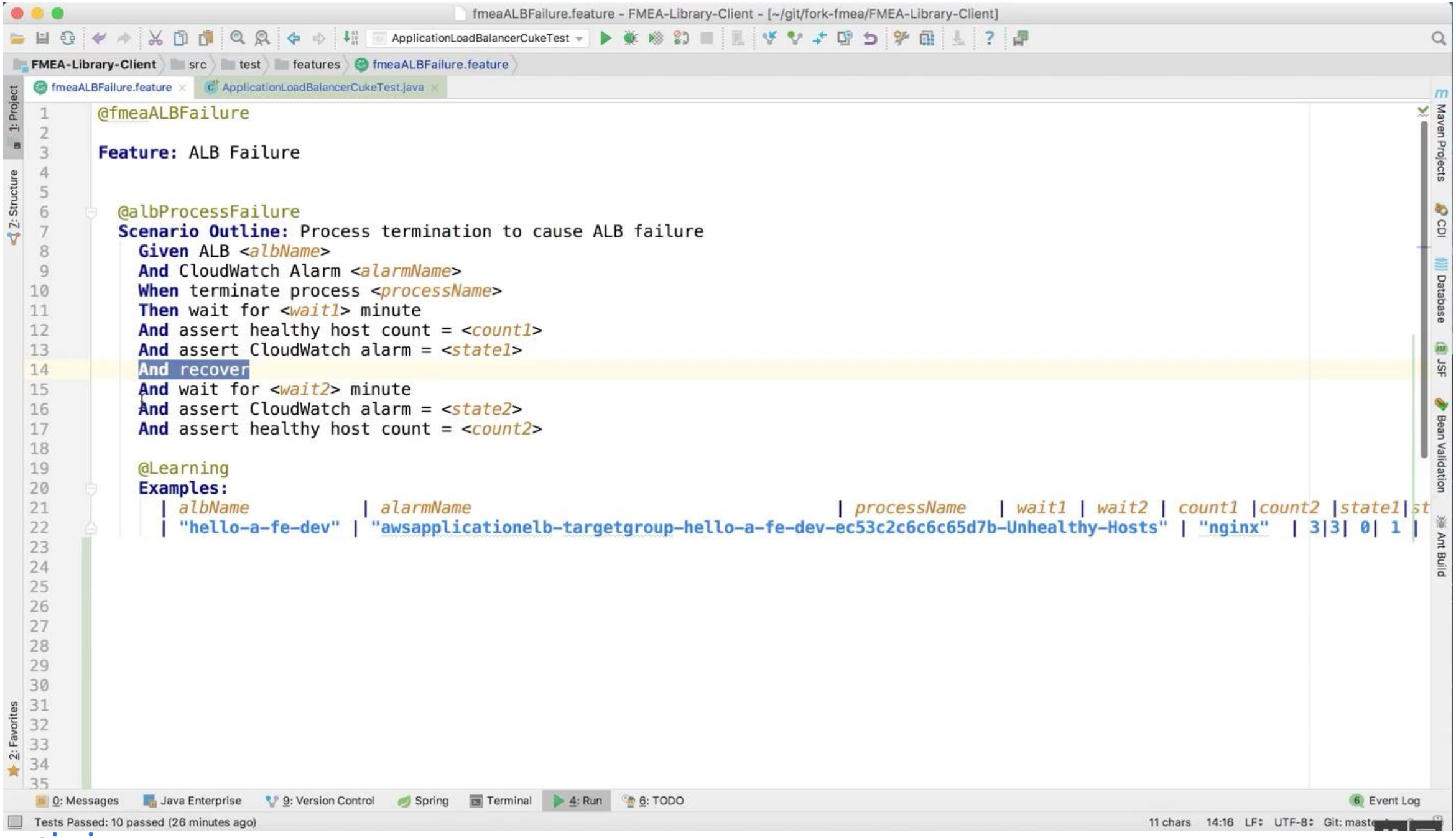
**Feature:** Route53 Failover

@route53Failover
**Scenario Outline:** Route53 failover by bringing down hosts in primary region
**Given** ALB <*albName*>
**And** CloudWatch Alarm <*alarmName*>
**And** R53 Healthcheck ID < *healthCheckId*>
**When** terminate all instances
**Then** wait for < *wait1* > minute
**And** assertCW alarm = <*state1*>
**And** assertTrue R53 failover from <primary> to <*secondary*>
**And** assertR53 HealthCheck state = <*healthCheckState1*>
**And** wait for <wait2> minute
**And** assertCW alarm = <*state2*>
**And** assertFalse R53 failover from <*primary*> to <*secondary*>
**And** assertR53 HealthCheck state = <*healthCheckState2*>

@dev
**Examples:**

| *albName* | *alarmName* | *primary* | *secondary* | *wait1* | *wait2* | *state1* | *state2* | *healthCheckId* | *healthCheckState1* | *healthCheckState2* |
|---|---|---|---|---|---|---|---|---|---|---|
| ”frontend-primary" | ”Route53-primaryFailureAlarm" | " login-primary.com " | ”login-secondary.com" | 3 | 5 | "ALARM" | "OK" | “id1234”| "FAILURE" | "SUCCESS" |

# Demo


fmeaALBFailure.feature - FMEA-Library-Client - [~/git/fork-fmea/FMEA-Library-Client]
ApplicationLoadBalancerCukeTest
FMEA-Library-Client
src
test
features
fmeaALBFailure.feature
fmeaALBFailure.feature
ApplicationLoadBalancerCukeTest.java
@fmeaALBFailure
Feature: ALB Failure
@albProcessFailure
Scenario Outline: Process termination to cause ALB failure
Given ALB <albName>
And CloudWatch Alarm <alarmName>
When terminate process <processName>
Then wait for <wait1> minute
And assert healthy host count = <count1>
And assert CloudWatch alarm = <state1>
And recover
And wait for <wait2> minute
And assert CloudWatch alarm = <state2>
And assert healthy host count = <count2>
@Learning
Examples:
| albName | alarmName | processName | wait1 | wait2 | count1 | count2 | state1 | st
| "hello-a-fe-dev" | "awsapplicationelb-targetgroup-hello-a-fe-dev-ec53c2c6c6c65d7b-Unhealthy-Hosts" | "nginx" | 3 | 3 | 0 | 1 |
1: Project
7: Structure
2: Favorites
Maven Projects
CDI
Database
JSF
Bean Validation
Ant Build
0: Messages
Java Enterprise
9: Version Control
Spring
Terminal
4: Run
6: TODO
6 Event Log
Tests Passed: 10 passed (26 minutes ago)
11 chars
14:16
LF
UTF-8
Git: maste

# Q&A