AWS re:Invent

DOP309-R

Improving resiliency with chaos engineering

Adrian Hornsby
Principal Evangelist
Amazon Web Services

Olga Hall
Senior Manager, Resilience Engineering
Amazon.com

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

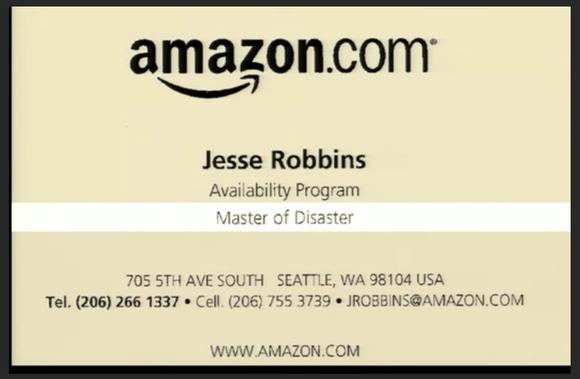
aws


E 4

GameDay at Amazon



Jesse Robbins, “Master of Disaster”

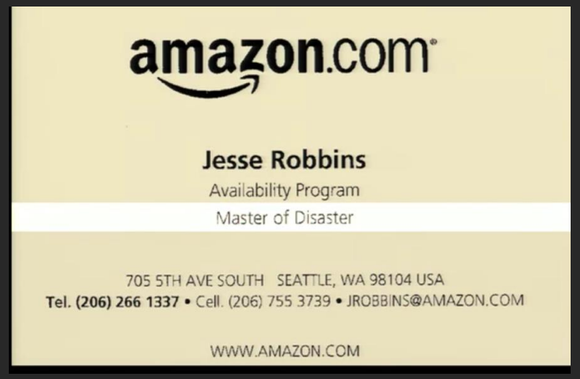


- A volunteer **firefighter**
- Created **GameDay in 2004** to purposefully create regular major failures
- Founded **Chef** and **Velocity**, the web performance & operations conference

GameDay at Amazon



Jesse Robbins, “Master of Disaster”



- Test, train and prepare Amazon systems, software, and people to respond to a disaster
- Increase Amazon retail website resiliency by purposely injecting failures into critical systems

Rise of the monkeys

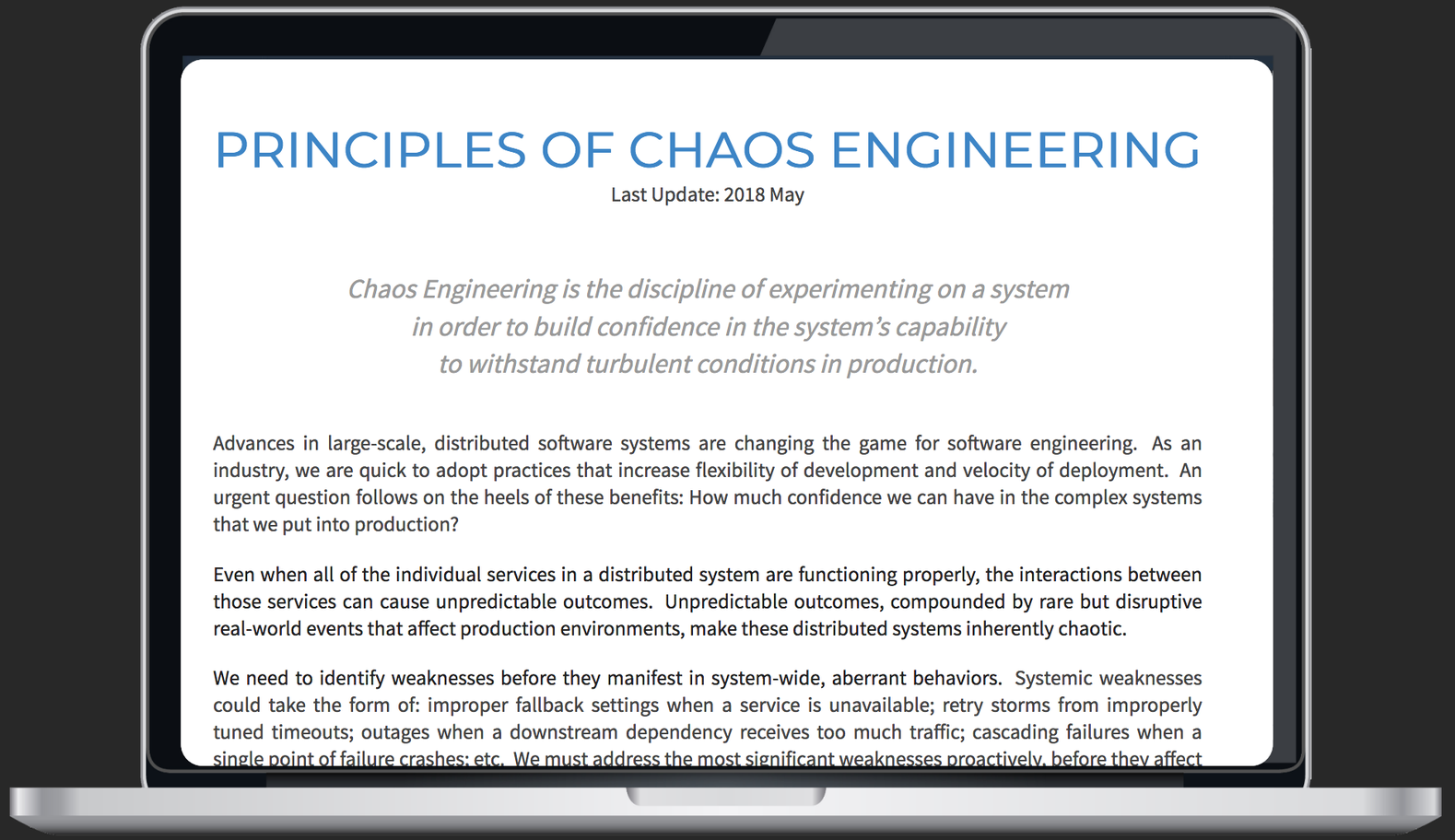
“Simian Army to keep our cloud safe, secure, and highly available.”

- 2011 Netflix blog

Set of scheduled agents:

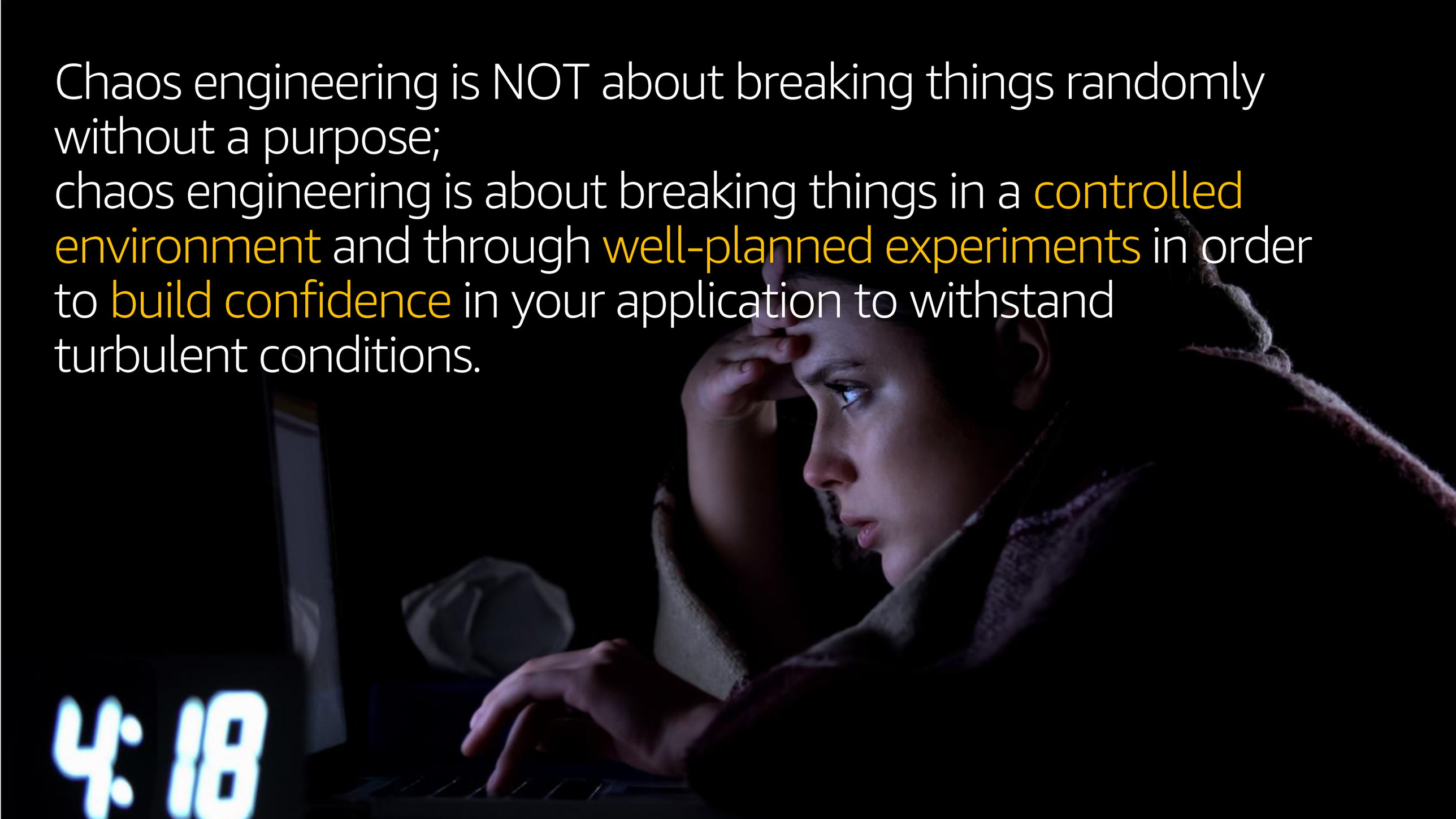
- Shut down services randomly
- Slow down performances
- Check conformity
- Break an entire region
- Integrate with spinnaker (CI/CD)

Chaos engineering formalized by Netflix (mid-2015)



principlesofchaos.org

Chaos engineering is NOT about breaking things randomly without a purpose;
chaos engineering is about breaking things in a controlled environment and through well-planned experiments in order to build confidence in your application to withstand turbulent conditions.



Some prerequisites to chaos engineering

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Software

- Certificate expiration
- Memory leaks
- Licenses
- Versioning

Infrastructure

- Redundancy (multi-AZ)
- Self-healing
- Isolation (bulkheads)
- Infrastructure as code

Application

- Timeouts
- Retries w/backoff
- Exception handling
- Circuit breakers
- Load shedding

Operations

- Human operators
- Monitoring and observability
- Incident response
- Measure, measure, measure

Software

- Certificate expiration
- Memory leaks
- Licenses
- Versioning

Infrastructure

- Redundancy (multi-AZ)
- Self-healing
- Isolation (bulkheads)
- Infrastructure as code



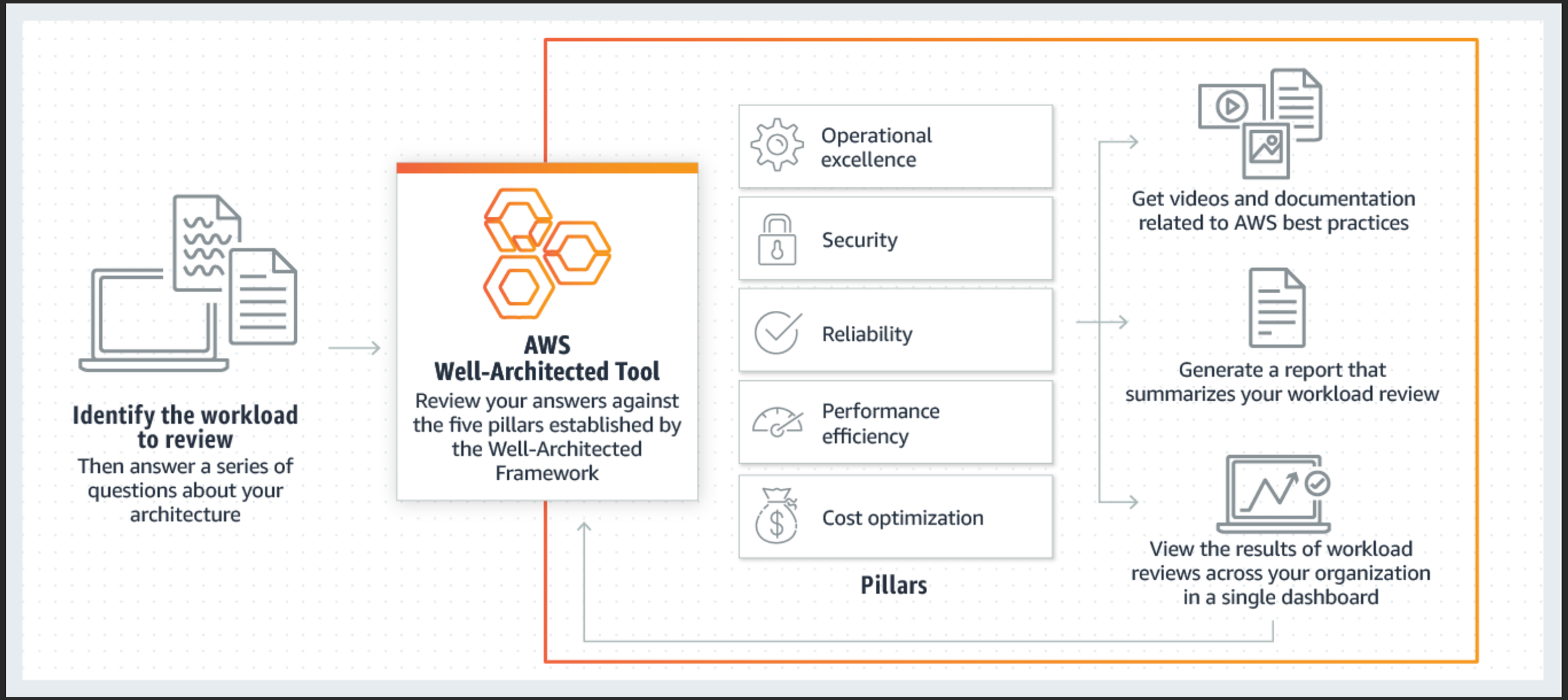
Application

- Timeouts
- Retries w/backoff
- Exception handling
- Circuit breakers
- Load shedding

Operations

- Monitoring and observability
- Incident response
- Measure, measure, measure

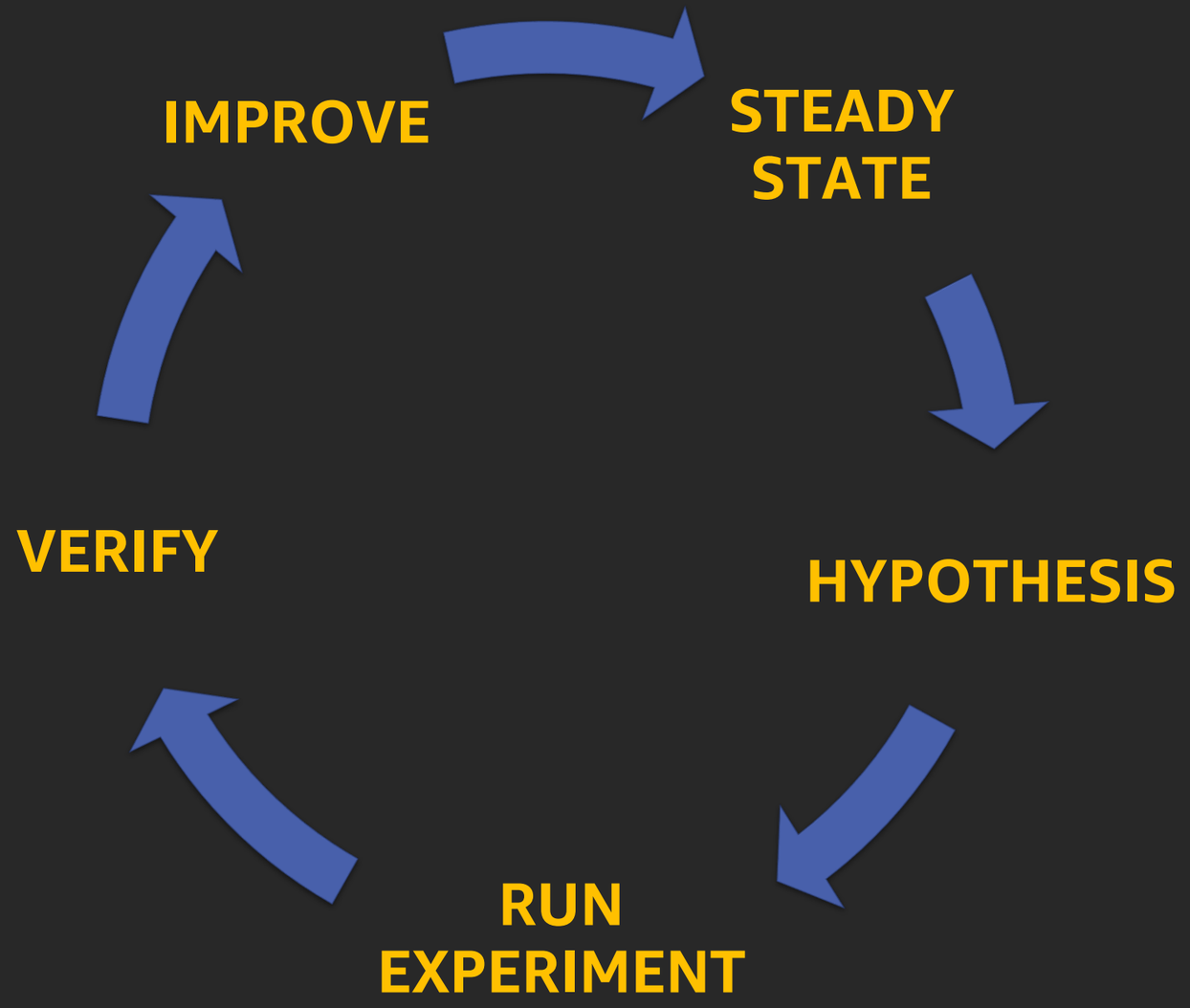
https://aws.amazon.com/wellarchitected



Phases of chaos engineering

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Phases of chaos engineering

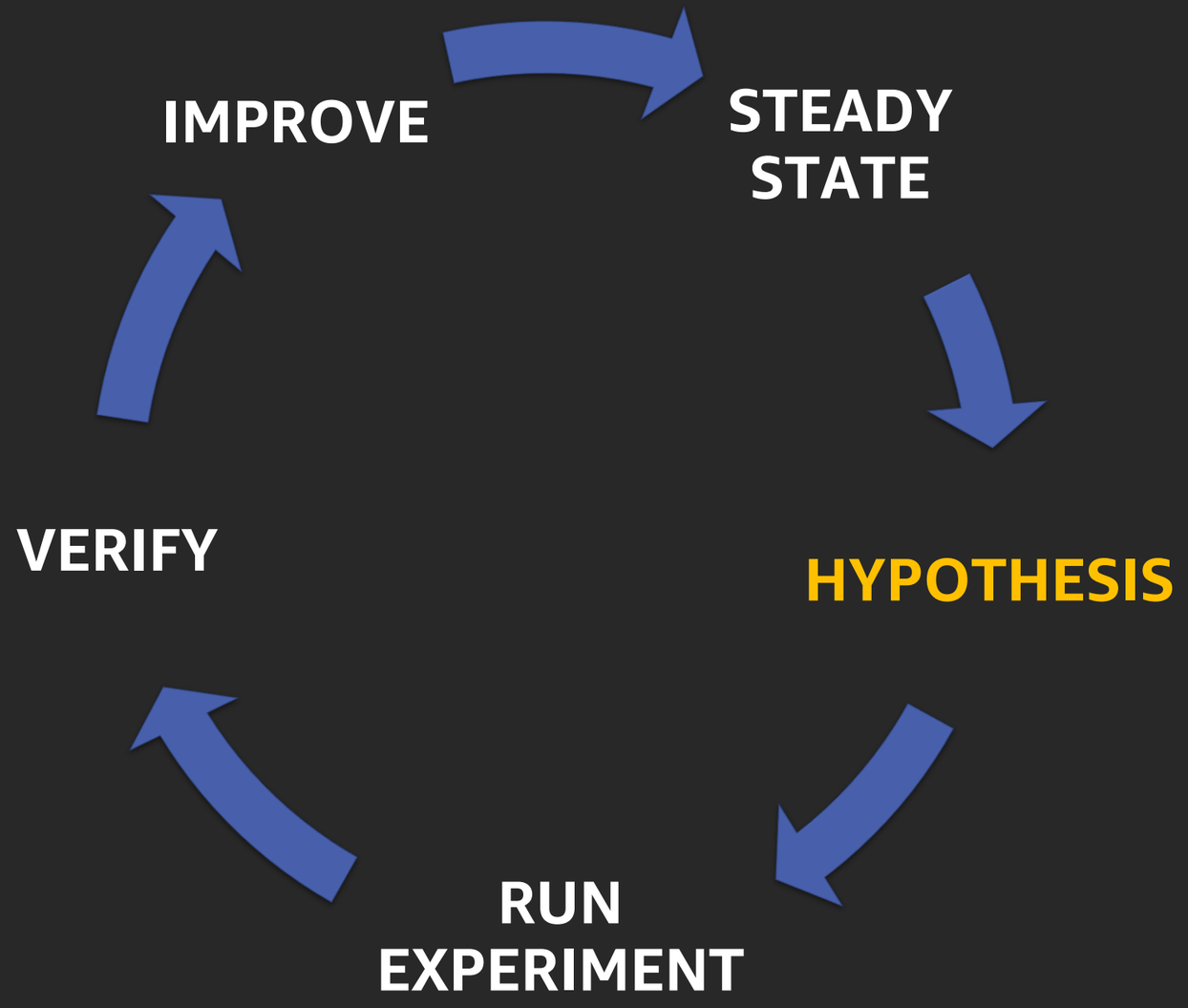


What is steady state?

- ”Normal” behavior of your system
- Not the internal attributes of the system (CPU, memory, etc.)
- Operational metrics tied with customer experience yields best results

The steady state varies when an unmitigated failure triggers an unexpected problem, and it should cause the chaos experiment to be aborted.

Phases of chaos engineering



What if…?

“What if this load balancer breaks?”

“What if Redis becomes slow?”

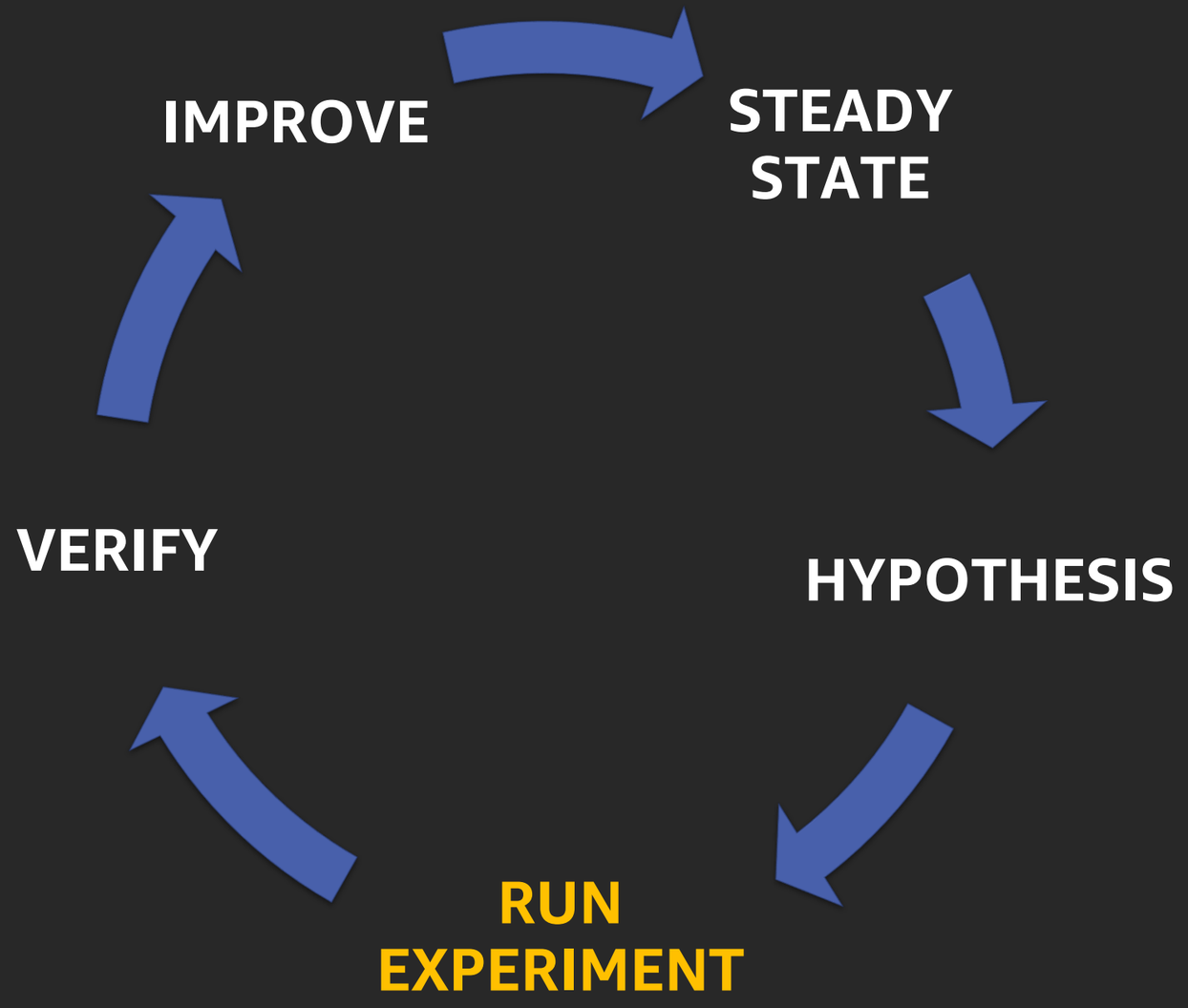
“What if a host on Cassandra goes away?”

”What if latency increases by 300 ms?”

”What if the database stops?”

Make it everyone’s problem!

Phases of chaos engineering



Failure injection

- Start small & build confidence
  - Application level (exceptions, errors, etc.)
  - Host level (services, processes, etc.)
  - Resource attacks (CPU, memory, IO, etc.)
  - Network attacks (dependencies, latency, packet loss, etc.)
  - AZ attack
  - Region attack
  - People attack

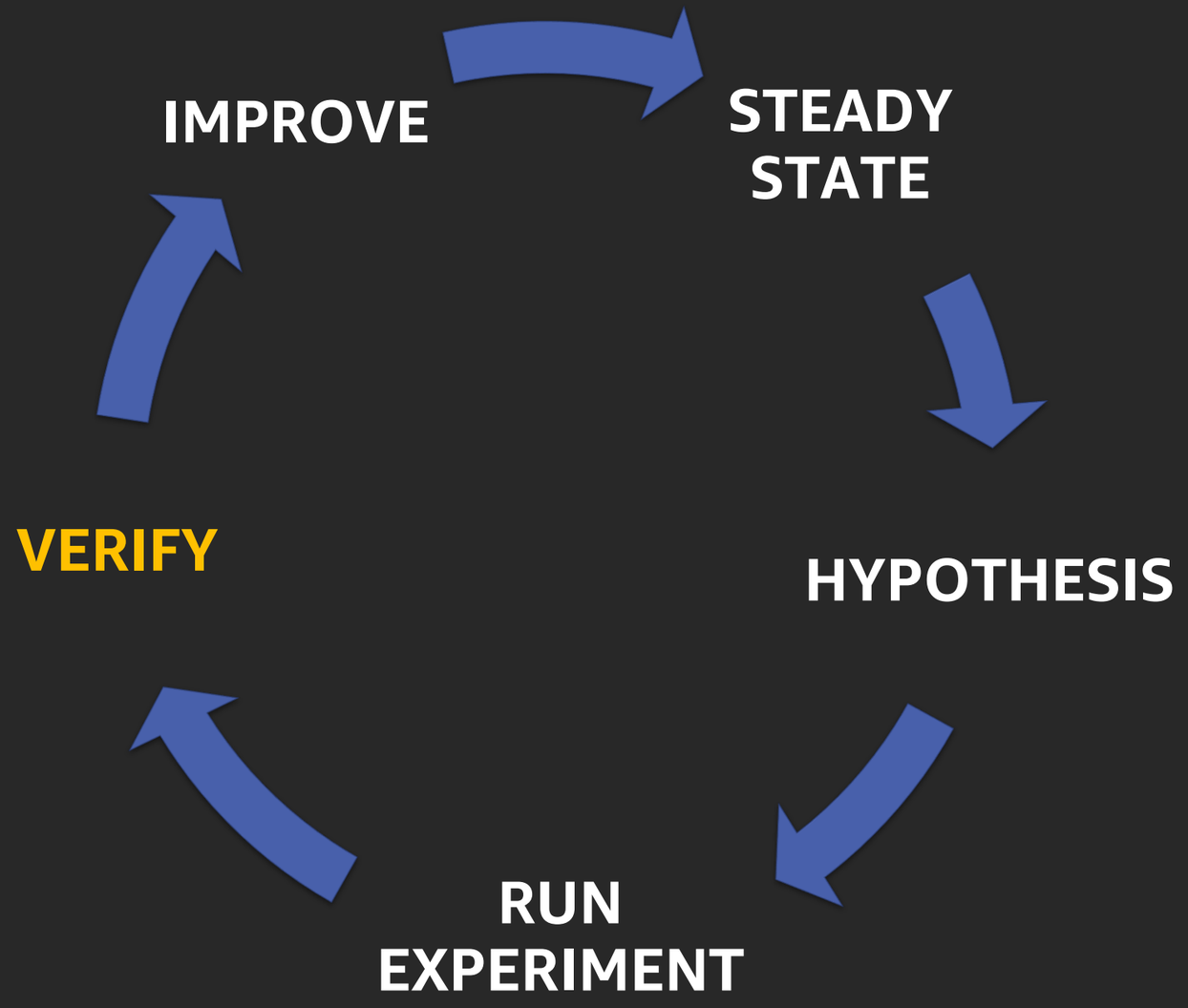
Rule of thumb

Have an emergency STOP or a good ROLLBACK strategy!

Careful with state that can’t be rolled back

(corrupt or incorrect data)

Phases of chaos engineering



Quantifying the result of the experiment

- Time to detect?
- Time for notification? And escalation?
- Time to public notification?
- Time for graceful degradation to kick in?
- Time for self-healing to happen?
- Time to recovery—partial and full?
- Time to all-clear and stable?

Postmortems —COE (correction of errors)

- What happened?
- What was the impact on customers and your business?
- What were the causes? The 5 whys—not linear
- What data do you have to support this?
  - Especially metrics and graphs
- What lessons did you learn?
- What corrective actions are you taking?
  - Actions items
  - Related items (trouble tickets, etc.)

Blast radius

- How many customers?
- What functionality?
- How many locations?

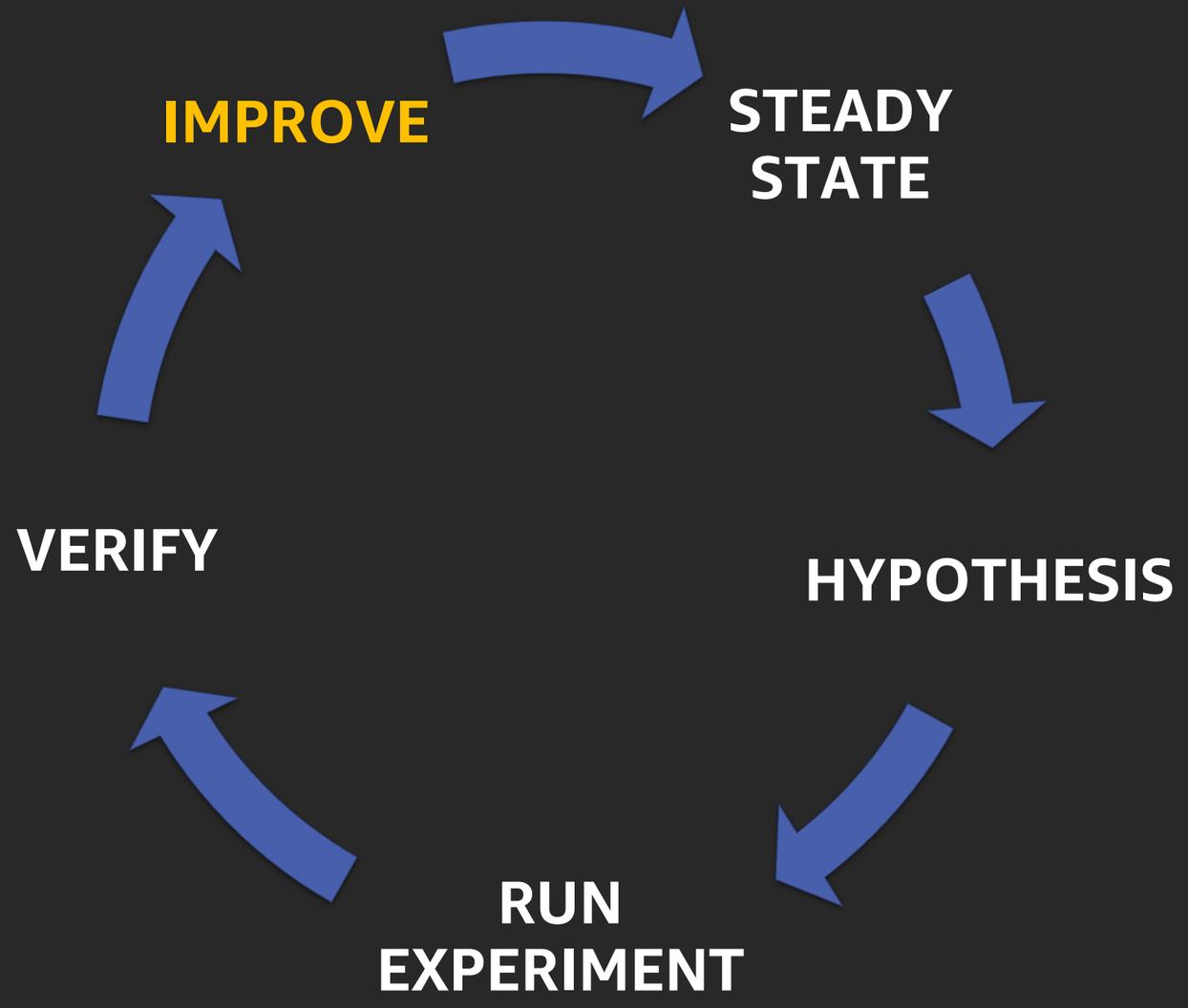


Two rules to remember

Don’t blame people for mistakes

There is no isolated ”cause” of an accident

Phases of chaos engineering



Audit & inspection

Weekly operational metrics review

- Continuous inspection mechanism
- Maintains focus on operations
- Foundation of a healthy operations program

Typical agenda

- Success stories
- Action items follow-up
- Review COEs
- Review key service metrics

Big challenges to chaos engineering

- Chaos engineering won’t make your system more robust;
  people will
- Chaos engineering won’t replace __all__ the rest (test, quality, etc.)
- Rollbacks are HARD because of state
- Your systems will continue to fail—sorry
- Starting is perceived as hard

Big challenges to chaos engineering

Mostly cultural

- No time or flexibility to simulate disasters
- Team is already spending all of its time fixing things
- Can be very political
- Might force deep conversations
- Deeply invested in a specific technical roadmap that chaos engineering tests show is not as resilient to failures as originally predicted

From chaos to resilience

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

prime

Included with Prime membership













Rent or buy new releases













Subscribe to channels



MLB.TV



CBS ALL ACCESS

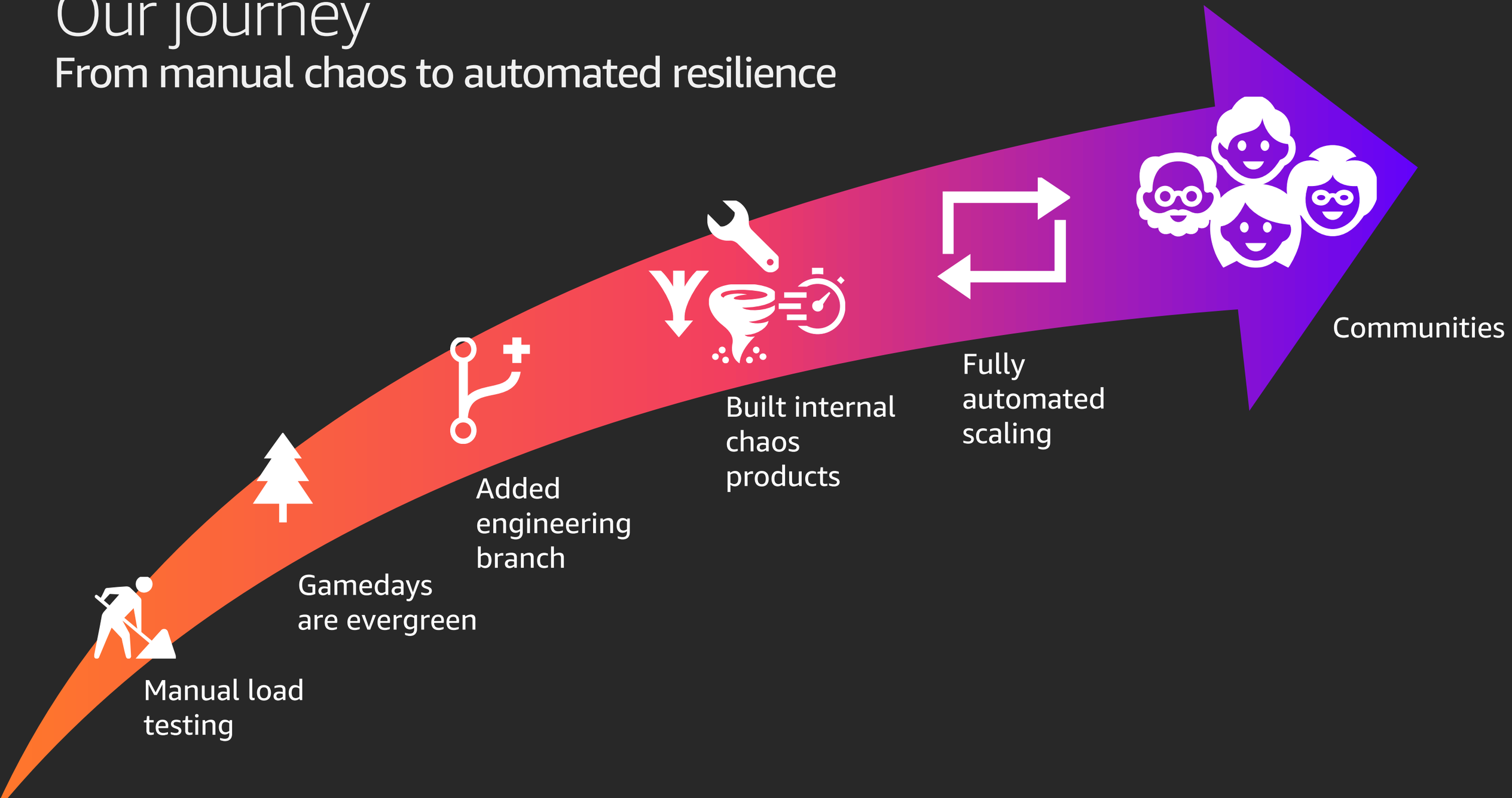
CINEMAX

OUTLANDER

STARZ

Our journey

From manual chaos to automated resilience



Our journey

From manual chaos to automated resilience

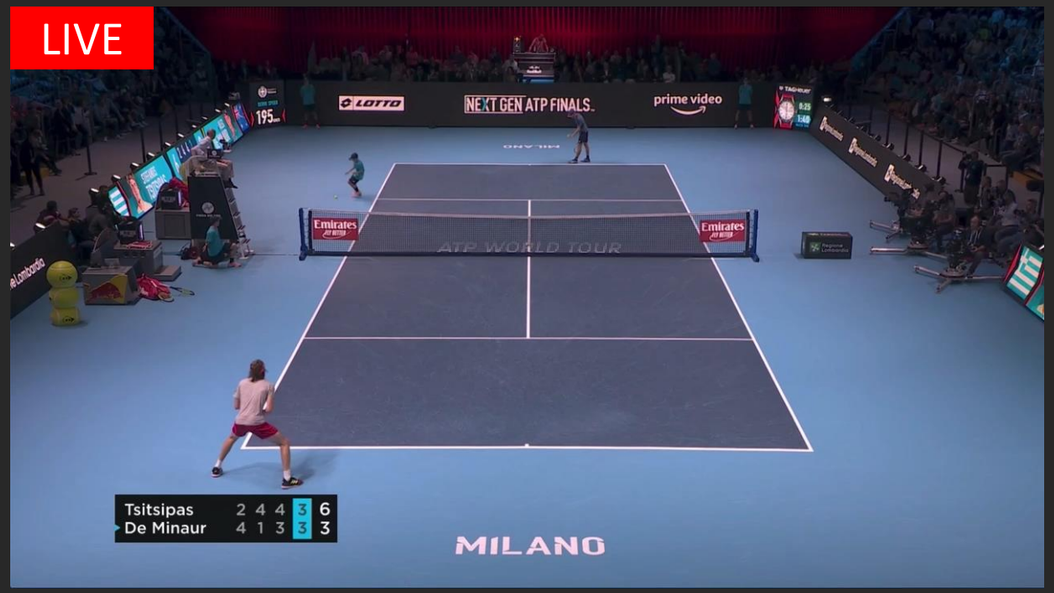
Evergreen programs

- Availability
- Scaling
- Resilience
- High-velocity events Readiness
- Performance

Program structure

- Central engineer team
- Infrastructure TPMs team
- Infrastructure efficiency and optimization
- Opt-in SPOCs in each team

Live streaming = higher bar



Customers are happy when

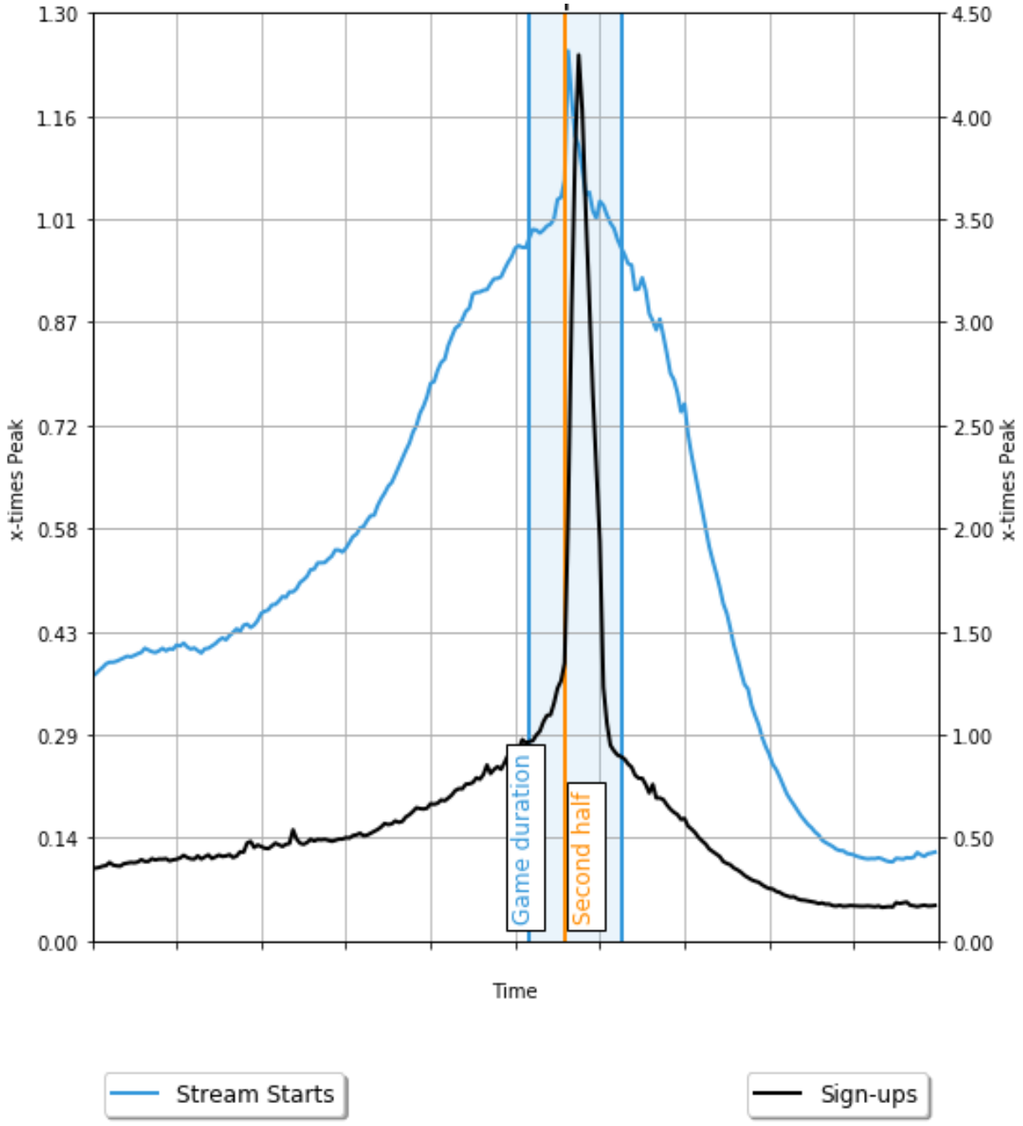
- Streams do not fail (100% Uptime, ~~99.99~~%)
- Performance on par and better than broadcast
- Video quality is enjoyable (UHD, HDR, HFR)
- Customer experience is intuitive

Readiness to handle failure (or the unknown) is feature zero.

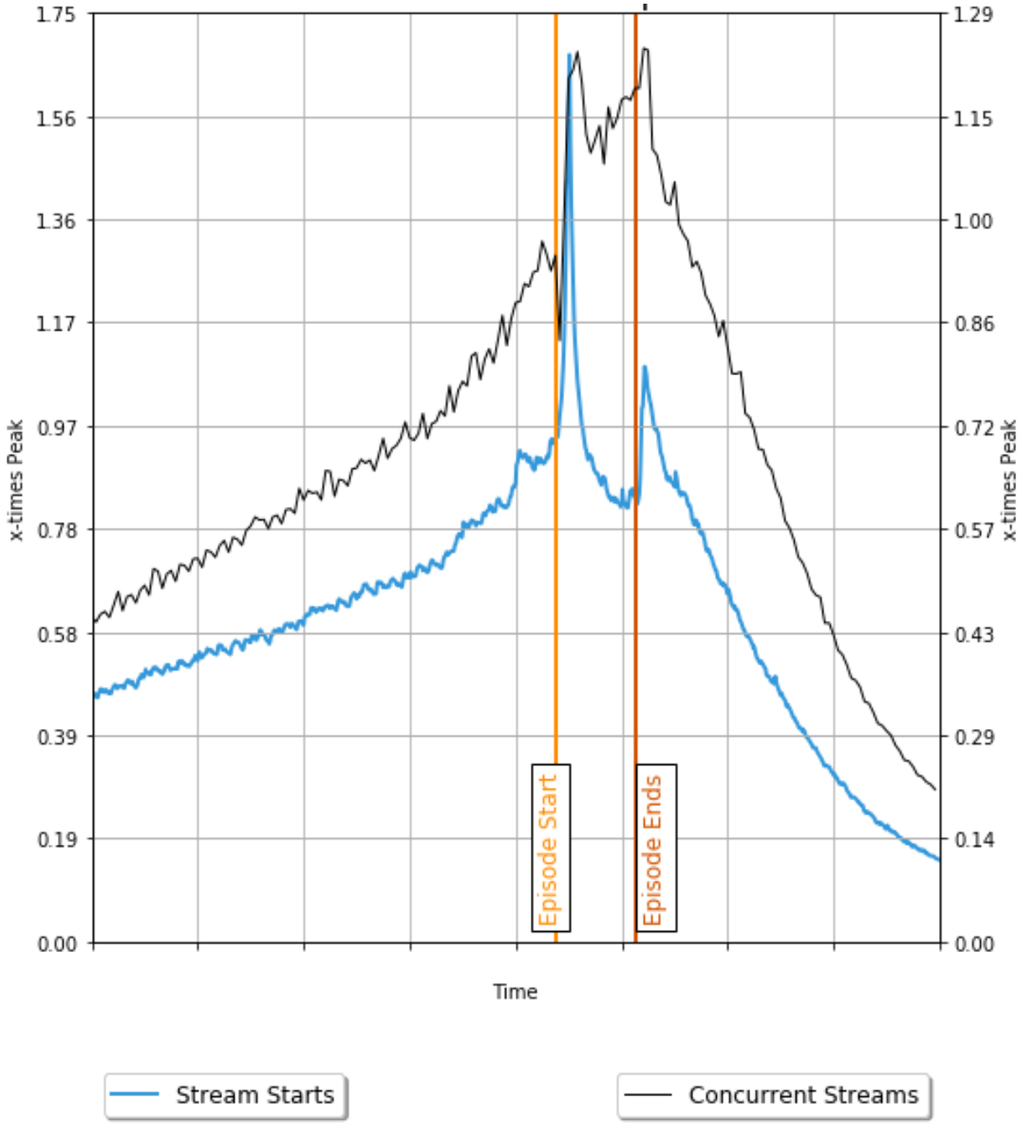
© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How would you predict this pattern?

The match gets interesting—subscribing in the second half

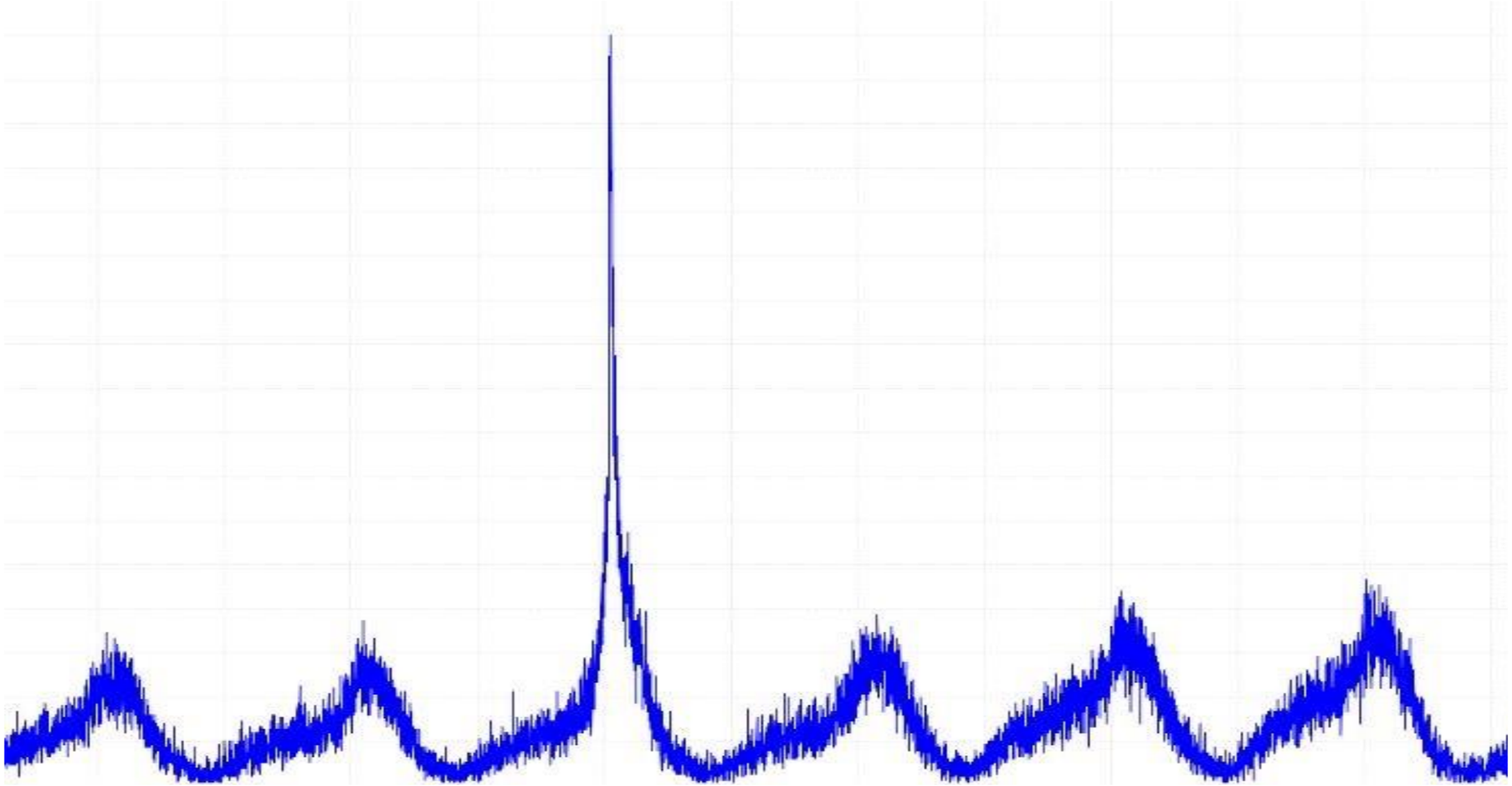


Customers re-watching a newly released same episode



How about this one?

When last minute gets real. Customers subscribing a minute before the show starts.





Little's Law

Number of Customers In System = Arrival Rate x Mean Time In System

$$L = \lambda . W$$

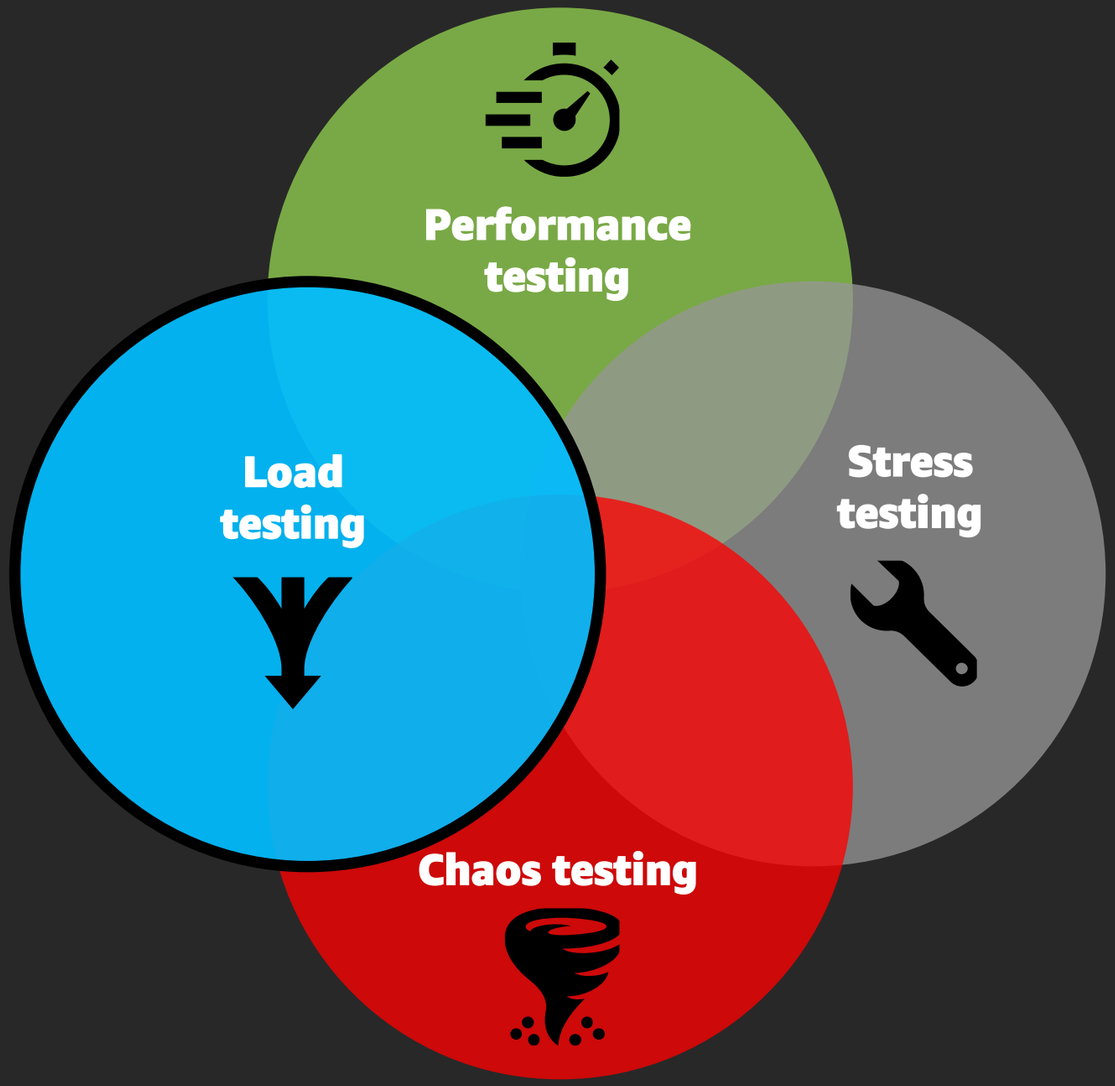
Understand customers first.
Create your chaos tool kit after.

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Where do you start?

Ask questions.

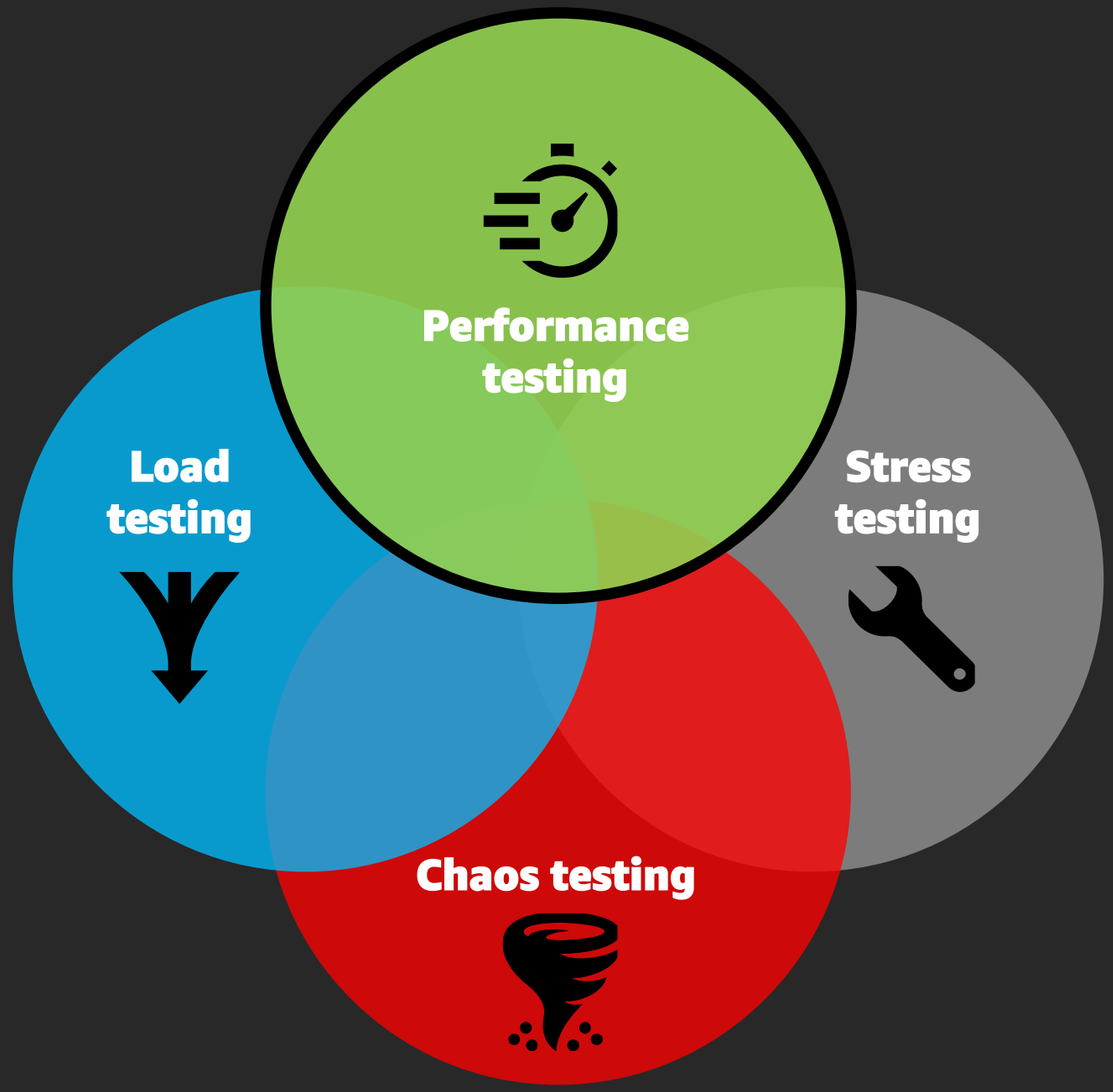
Can our services handle an expected realistic load?



Resilience framework

- ✓ Mostly positive testing
- ✓ Emphasis on prod behavior

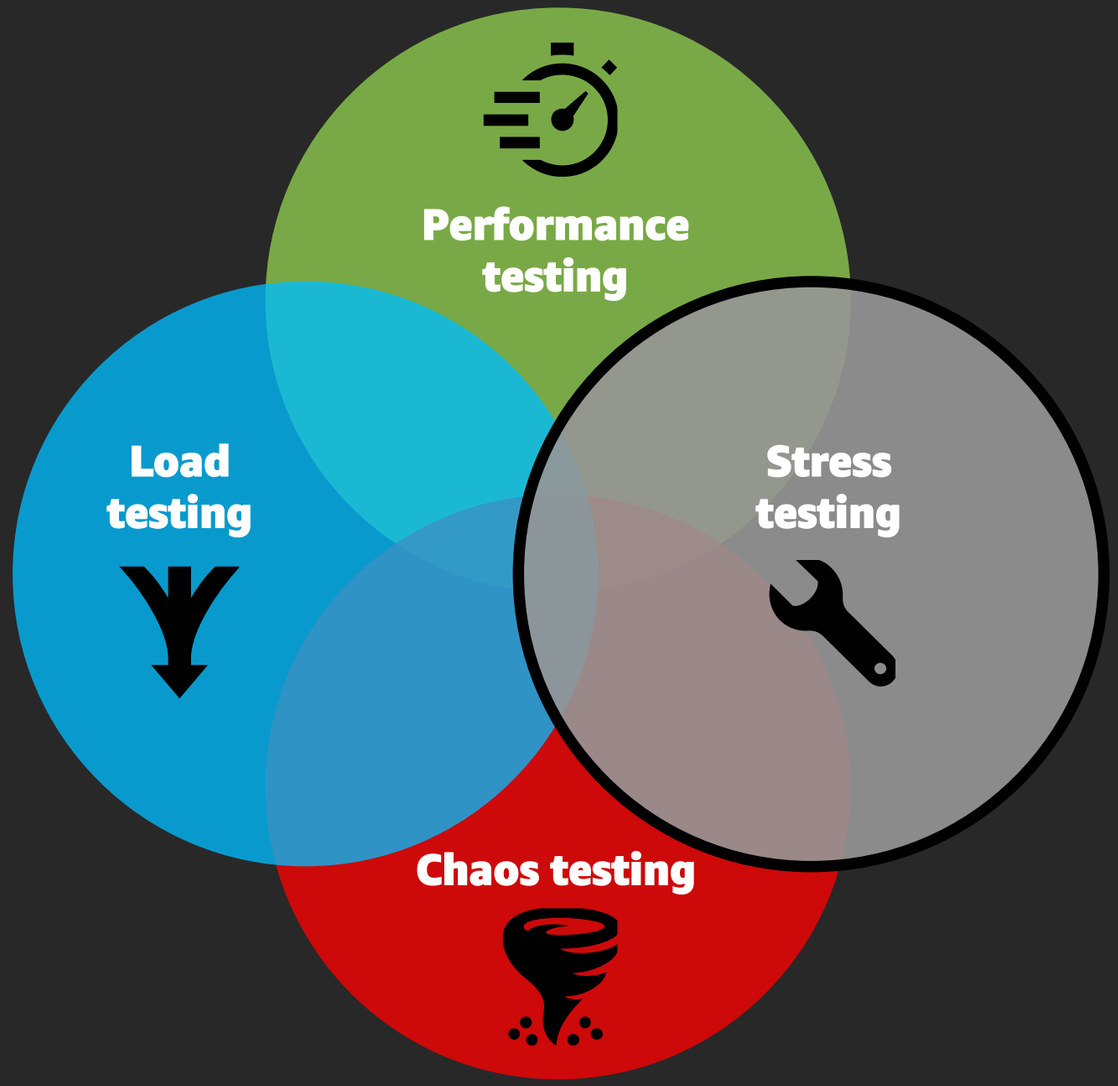
How does load change performance?



Resilience framework

- ✓ Latency
- ✓ Errors
- ✓ Throughput

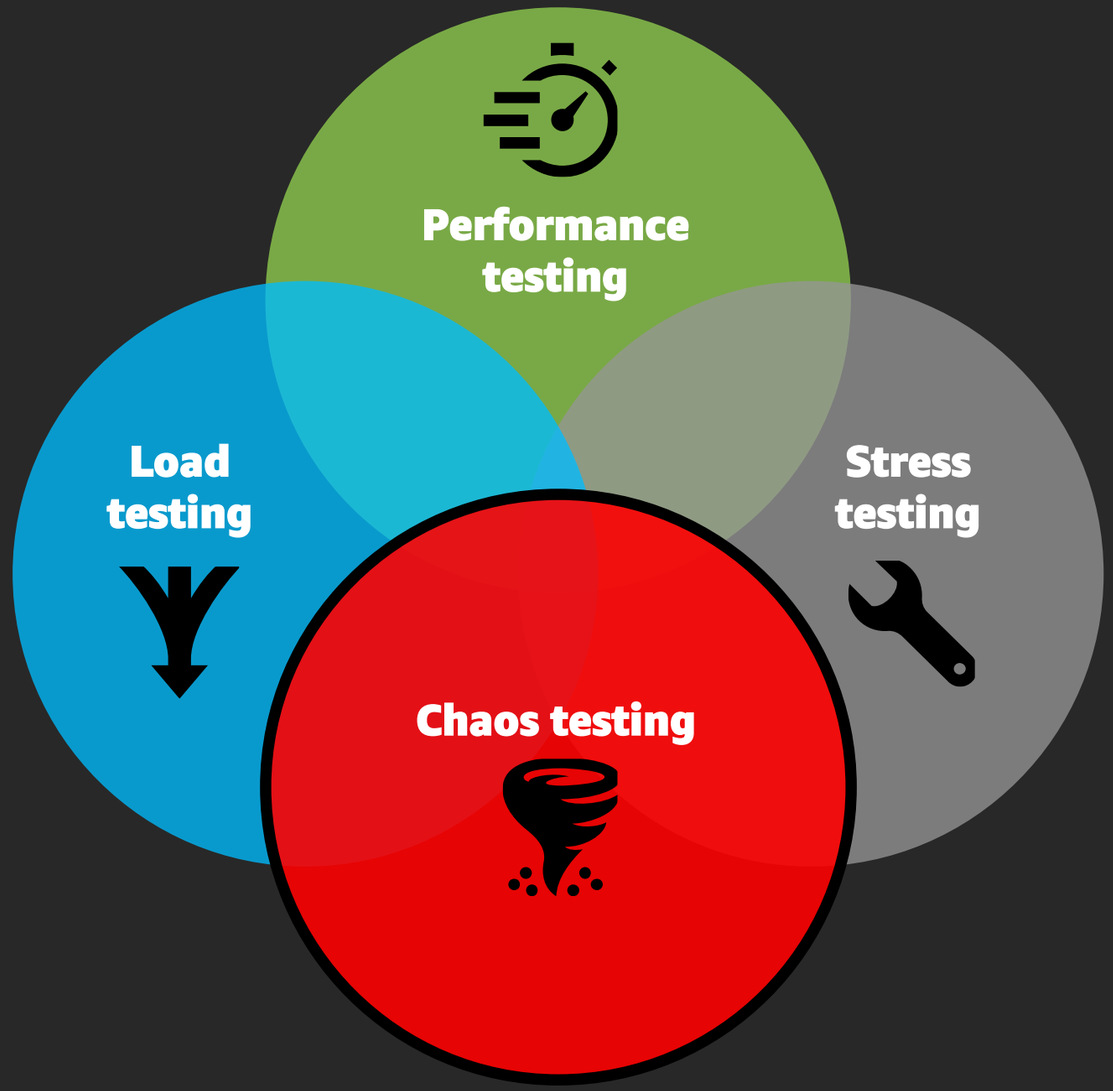
When and where is a breaking point visible?



Resilience framework

- ✓ Emphasis on breaking the system
- ✓ Not on prod-like behavior

How will services handle underlying failures?



Resilience framework

- ✓ Data center down
- ✓ Dependencies failing
- ✓ Slow network
- ✓ CPU or memory maxed out

What answers did we get?

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Arrival rate matters for live and/or exclusive video



Customer arrival rate

λ

Session starts

Discover

Select

TRANSACT

Concurrency

Playback

L W

Concurrent sessions

Our toolkit

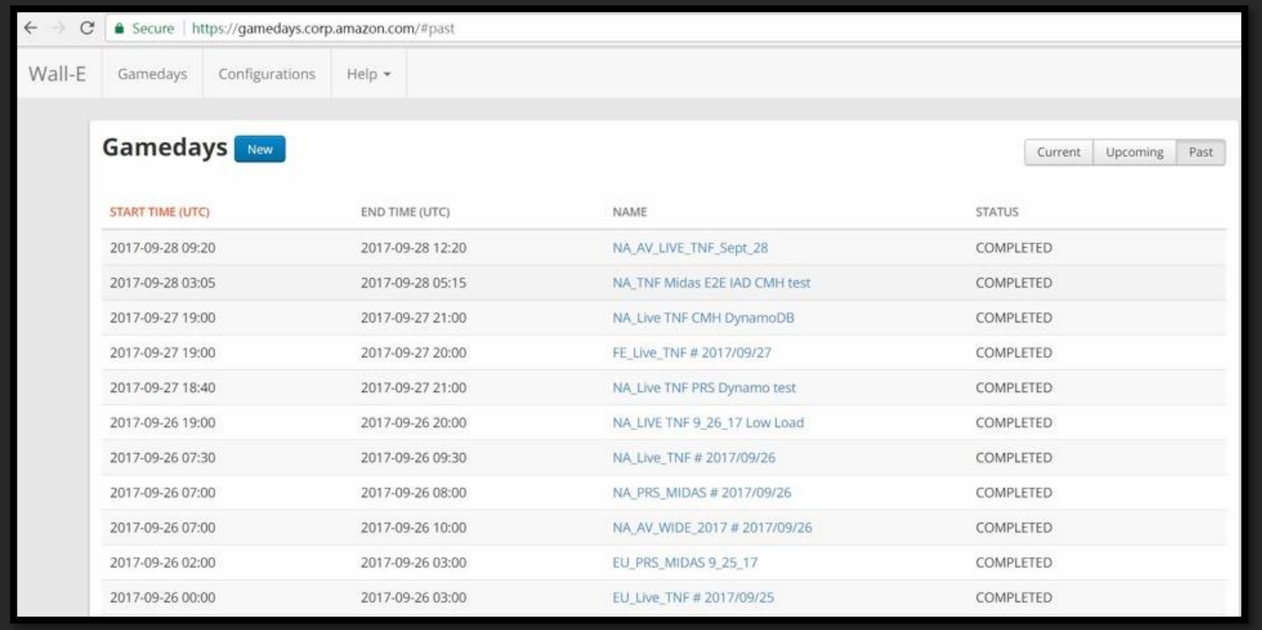
© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Platform-wide automated Gamedays

- Test ability to sustain a given target
- Replay a traffic segment that occurred during an outage
- Intentionally cause failures and determine if our systems behaved as expected
- Verify your alarms and configurations
- Validate a new service or a new feature—operational readiness review

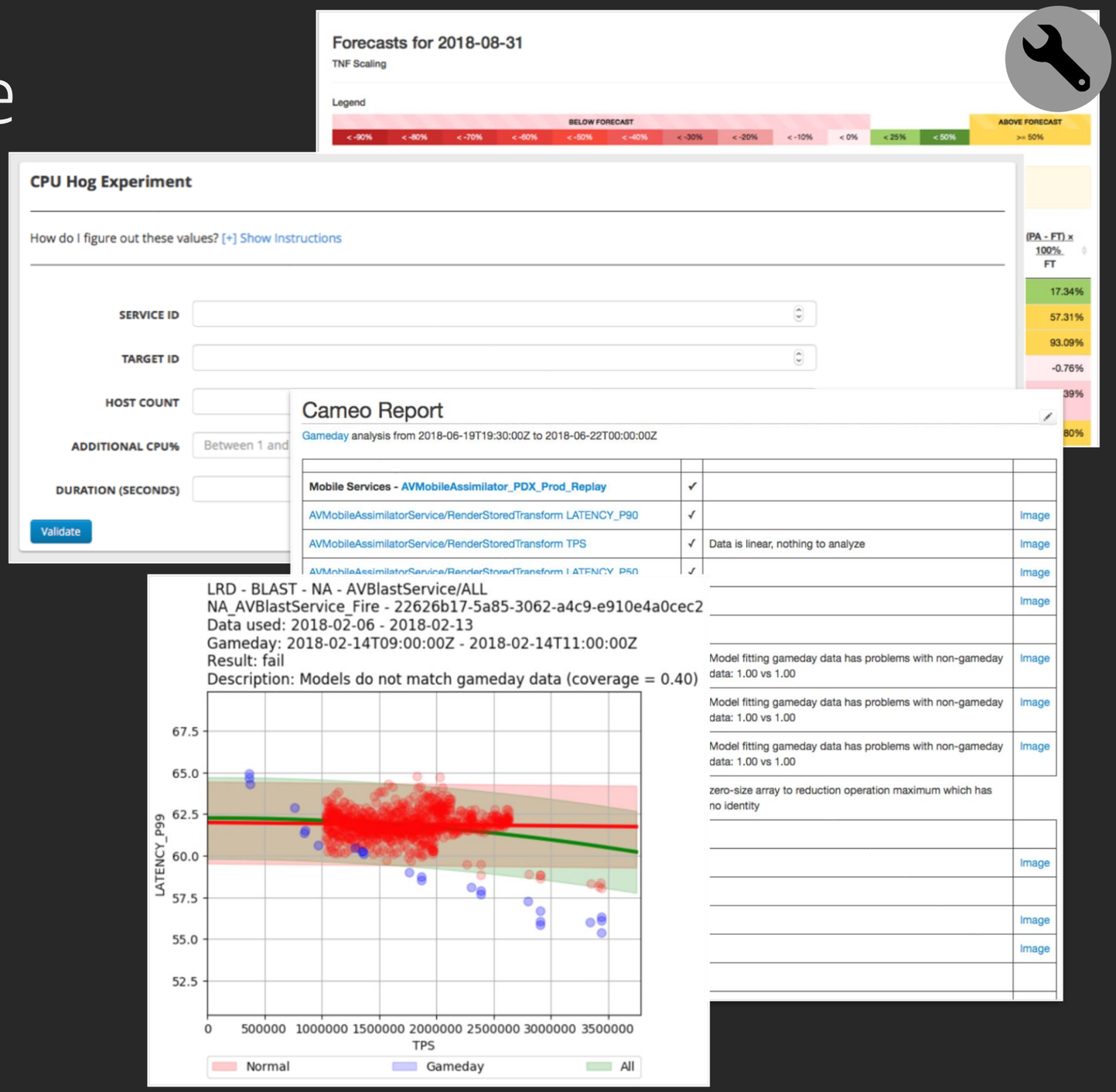
Tools for measuring performance

- Control plane for running Gamedays
- Has the ability to control any number of load generators
- Smart scheduling and hands off execution

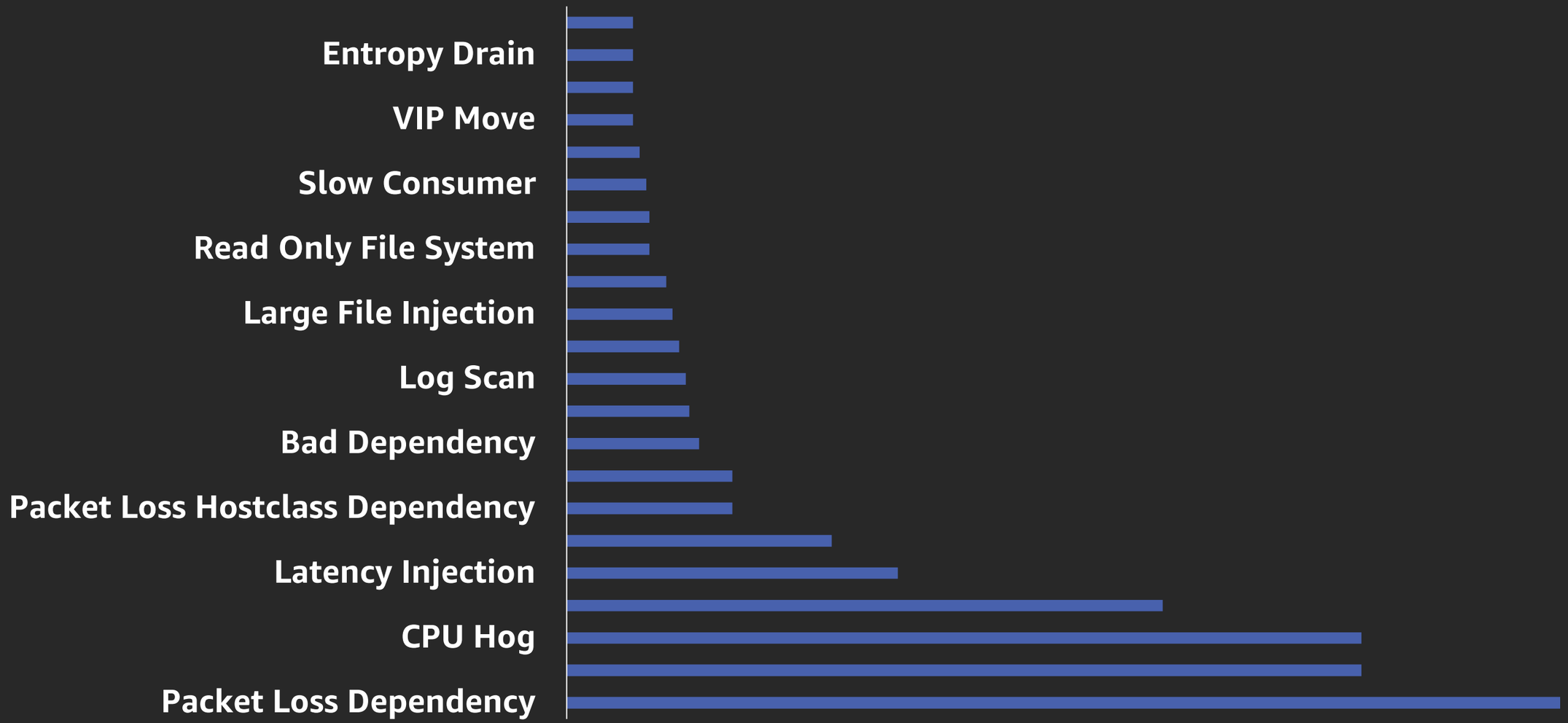


Resilience product suite

- Forecasting
- Resilience Experiments
- Reporting and Analysis
- Actual traffic vs. Gameday
- Load representation



Failure injection test (FIT) use case



Failure injection test (FIT) use case

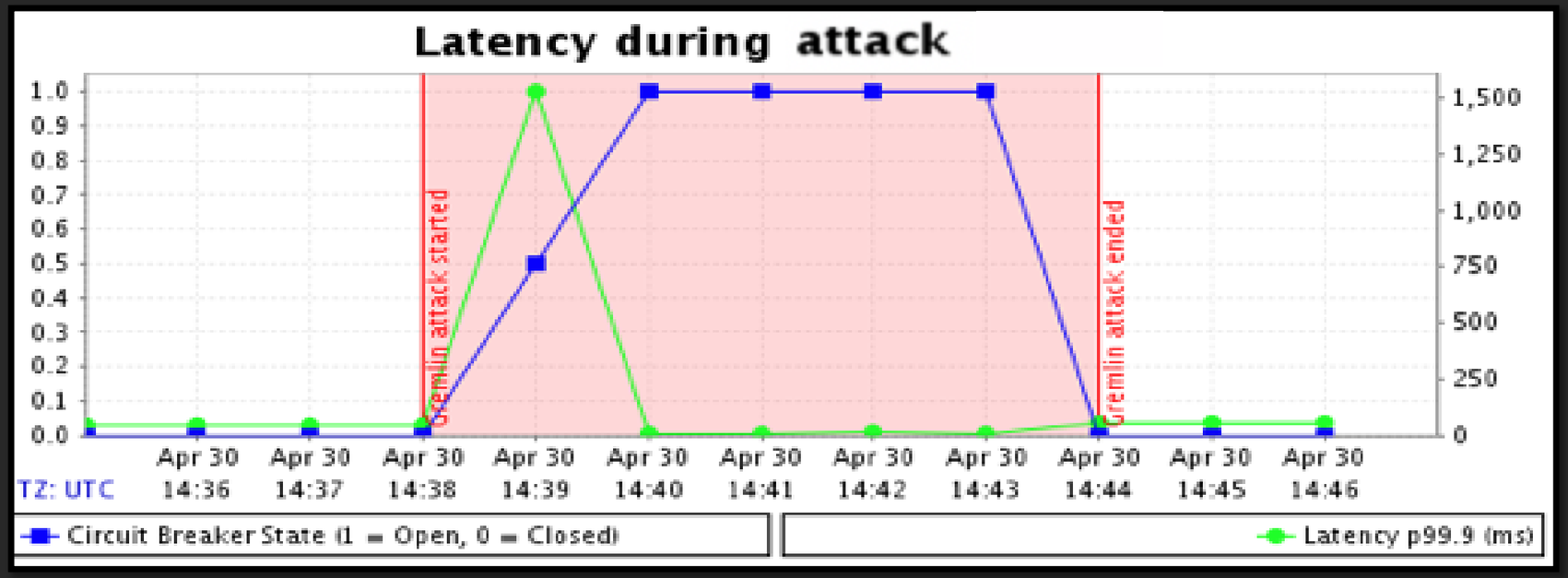


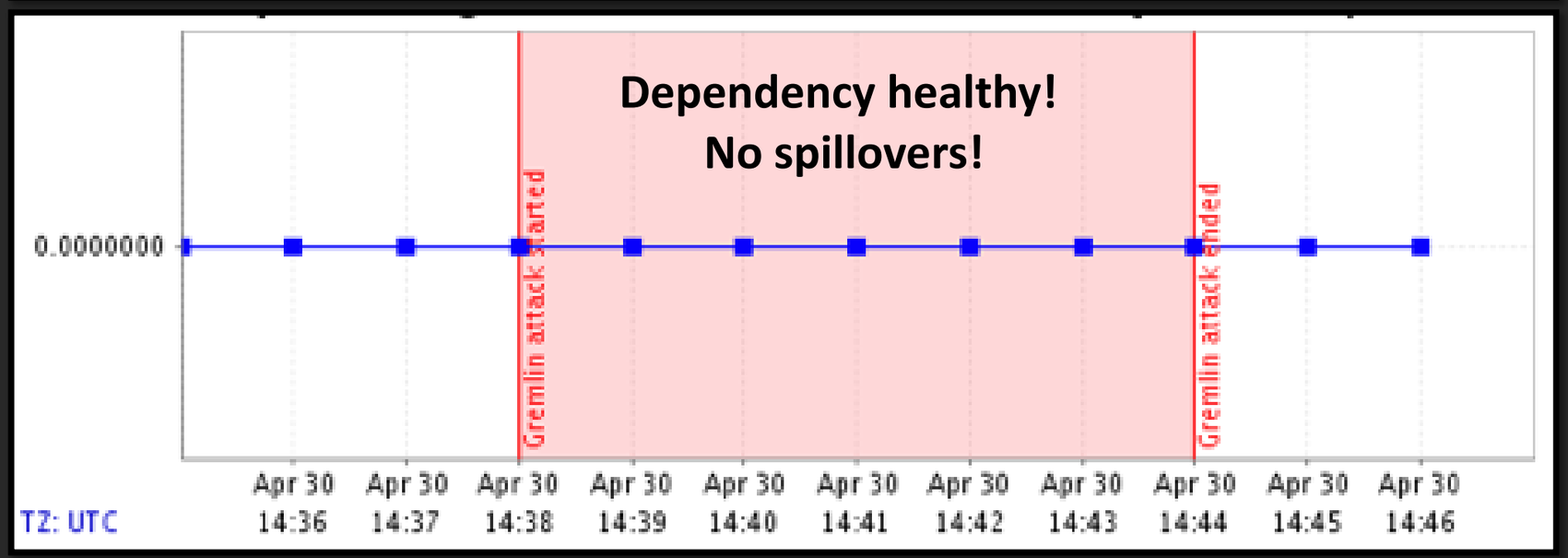
Problem: Downstream dependency throttling caused availability drop

Solution: Implemented fast-fail circuit-breaker

- FIT Test: Use a FIT to simulate dependency throttling and tune limits
- Outcome: Measured circuit-breaker performance, retries and latency impact

Failure injection test (FIT) use case





Insights

1. Need a tool set, not a single platform.
2. Opt for the best fit for your use cases.

Automation portfolio:

- Forecasting
- Failure injection
- Reporting & Learning
- Experiments Repository
- Services Discovery / Repository

Key takeaways



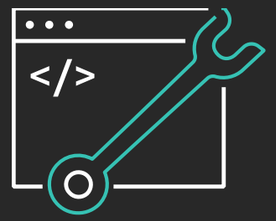
1. Start with a specific goal.
2. The first step is the most important. Keep a programmatic focus.
3. Resilience is bigger than chaos, it prepares for unknowns.
4. Keep learning about customers, create a feedback loop between your learnings and customers.
5. Chaos is fun! Start your journey today.

Meet chaos where you are.
Take it on your own journey.

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Learn DevOps with AWS Training and Certification

Resources created by the experts at AWS to propel your organization and career forward



Take free digital training to learn best practices for developing, deploying, and maintaining applications



Classroom offerings, like DevOps Engineering on AWS, feature AWS expert instructors and hands-on activities



Validate expertise with the **AWS Certified DevOps Engineer - Professional** or **AWS Certified Developer - Associate** exams

Visit aws.amazon.com/training/path-developing/

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws training and certification

Thank you!

Adrian Hornsby

adhorn@amazon.com

Twitter: @adhorn

Olga Hall

ohall@amazon.com

Twitter: @**ovhall**

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws



Please complete the session survey in the mobile app.

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.