Safe and Sound

Adrian Cockcroft

@adrianco

AWS VP Cloud Architecture Strategy

Availability, Safety and Security have similar characteristics

Hard to measure near misses

Hard to model complex dependencies

Catastrophic failure modes

Availability, Safety and Security have similar mitigations

Layered defense in depth

Bulkheads to contain blast radius

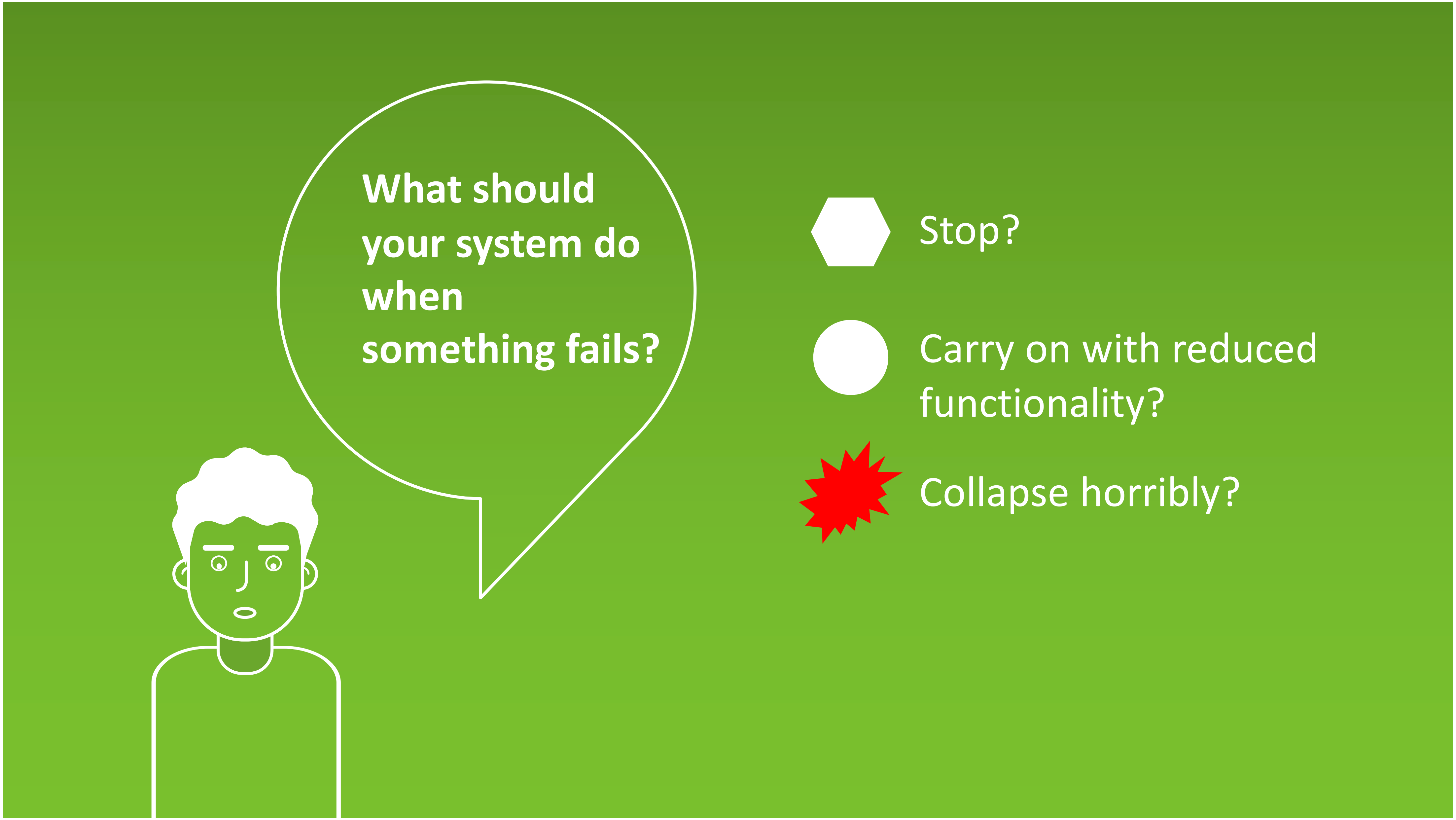
Minimize dependencies/privilege

Availability, Safety and Security Break Each Other

Security breaks availability

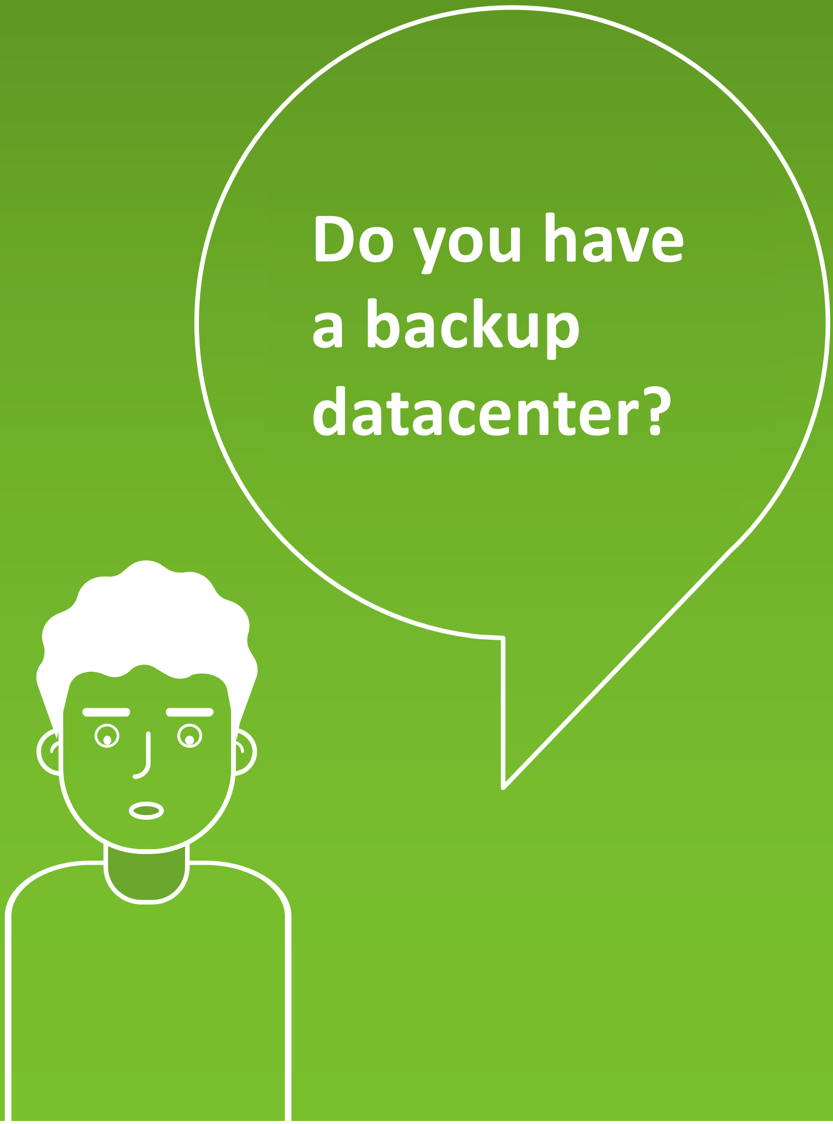
Availability breaks safety

Etc.


What should your system do when something fails?
Stop?
Carry on with reduced functionality?
Collapse horribly?

Permissive failure, what's the real cost of continuing?

See ***Memories, Guesses, and Apologies*** by Pat Helland



How often do you failover apps to it?

How often do you failover the **whole datacenter** at once?

“Availability Theater”

A fairy tale…

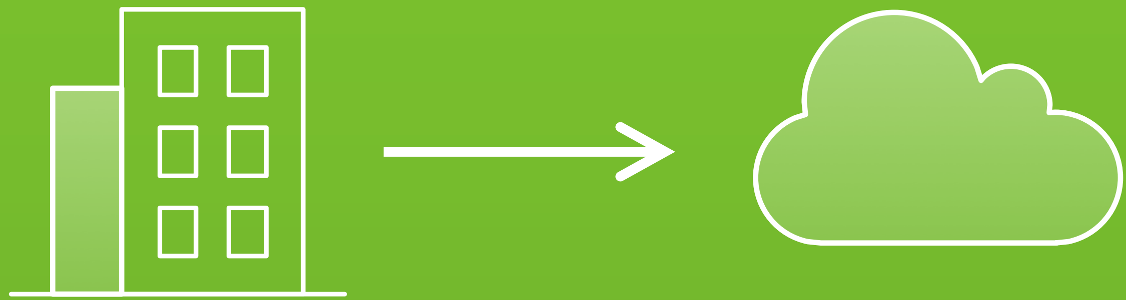
Once upon a time, in theory, if everything works perfectly, we have a plan to survive the disasters we thought of in advance.

How did that work out?

Forgot to renew domain name…	SaaS vendor
Didn’t update security certificate and it expired…	Entertainment site
Datacenter flooded in hurricane Sandy…	Finance company, Jersey City
Whoops!	**YOU, tomorrow**

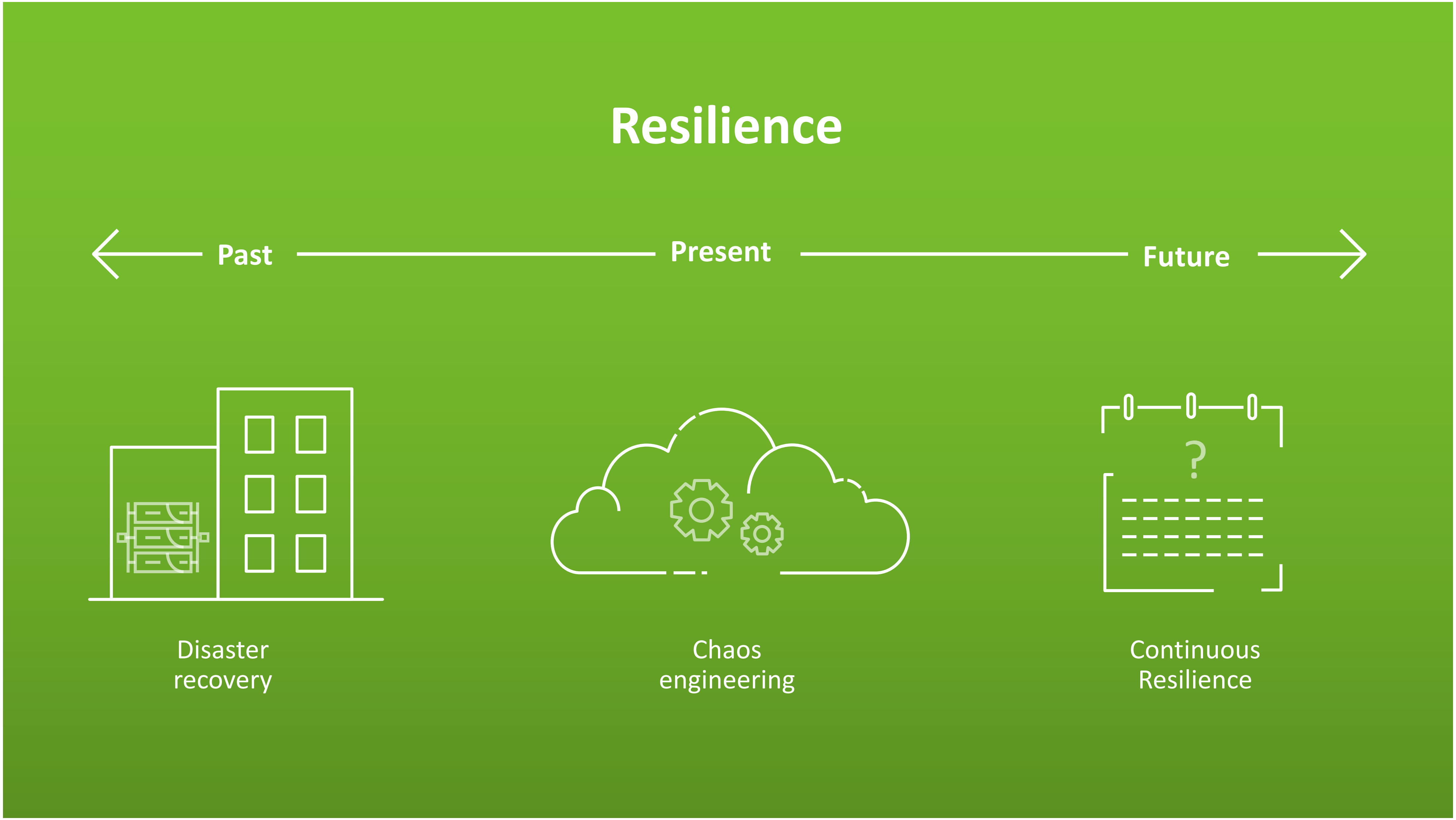
“You can’t legislate against failure, focus on fast detection and response.”

—Chris Pinkham



Datacenter to cloud migrations are under-way for the most business and safety critical workloads

AWS and our partners are developing patterns, solutions and services for customers in all industries including travel, finance, healthcare, manufacturing…


Resilience
Past
Present
Future
Disaster recovery
Chaos engineering
Continuous Resilience

You can only be as strong as your
weakest link

Dedicated teams are needed to find weaknesses before they take you out!

Defense In Depth

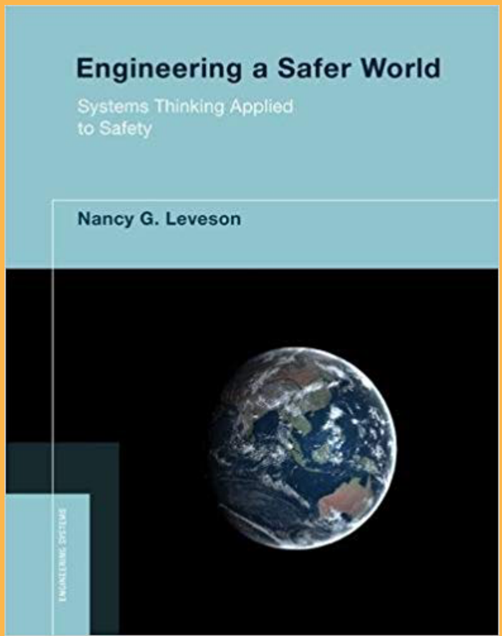
Experienced staff

Robust applications

Dependable switching fabric

Redundant service foundation

"If we change the name from chaos engineering to continuous resilience, will you let us do it all the time in production?"



Engineering a Safer World

Systems Thinking Applied to Safety

Nancy G. Leveson

STPA – Systems Theoretic Process Analysis

STAMP – Systems Theoretic Accident Model & Processes

http://psas.scripts.mit.edu for handbook and talks

Observability

Kalman, 1961 paper

On the general theory of control systems

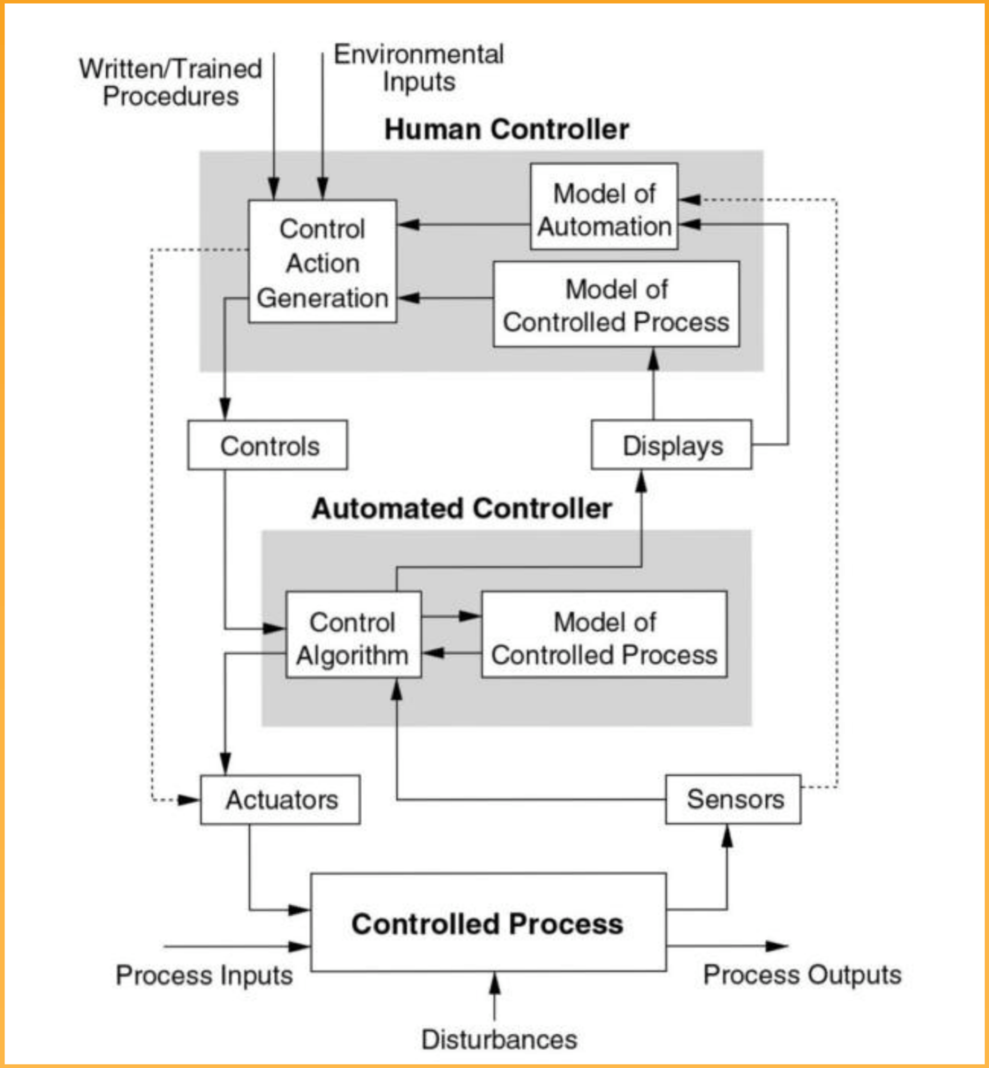
A system is observable If the behavior of the entire system can be determined by only looking at its inputs and outputs

Physical and software control systems are based on models, remember all models are wrong, but some models are useful…

Observability

STPA Model

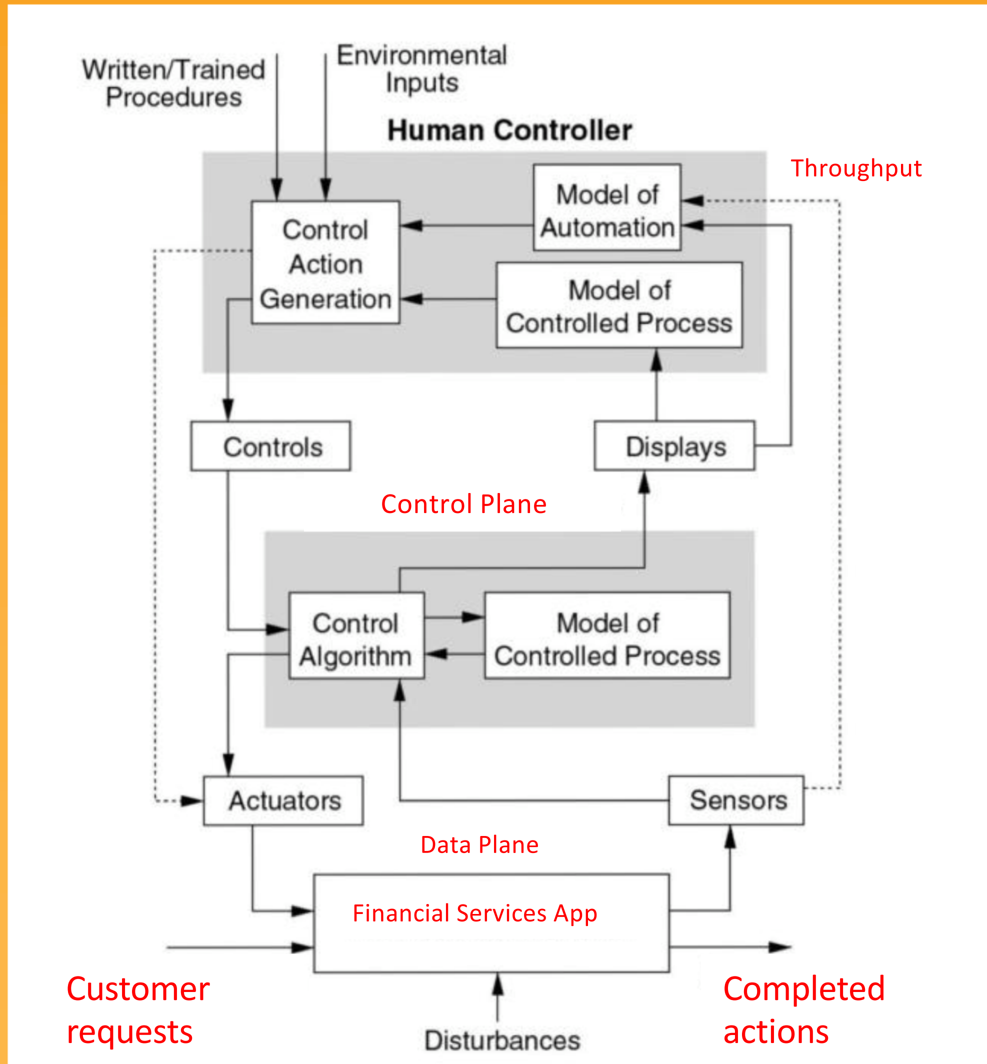
(System Theoretic Process Analysis)



Observability

STPA Model

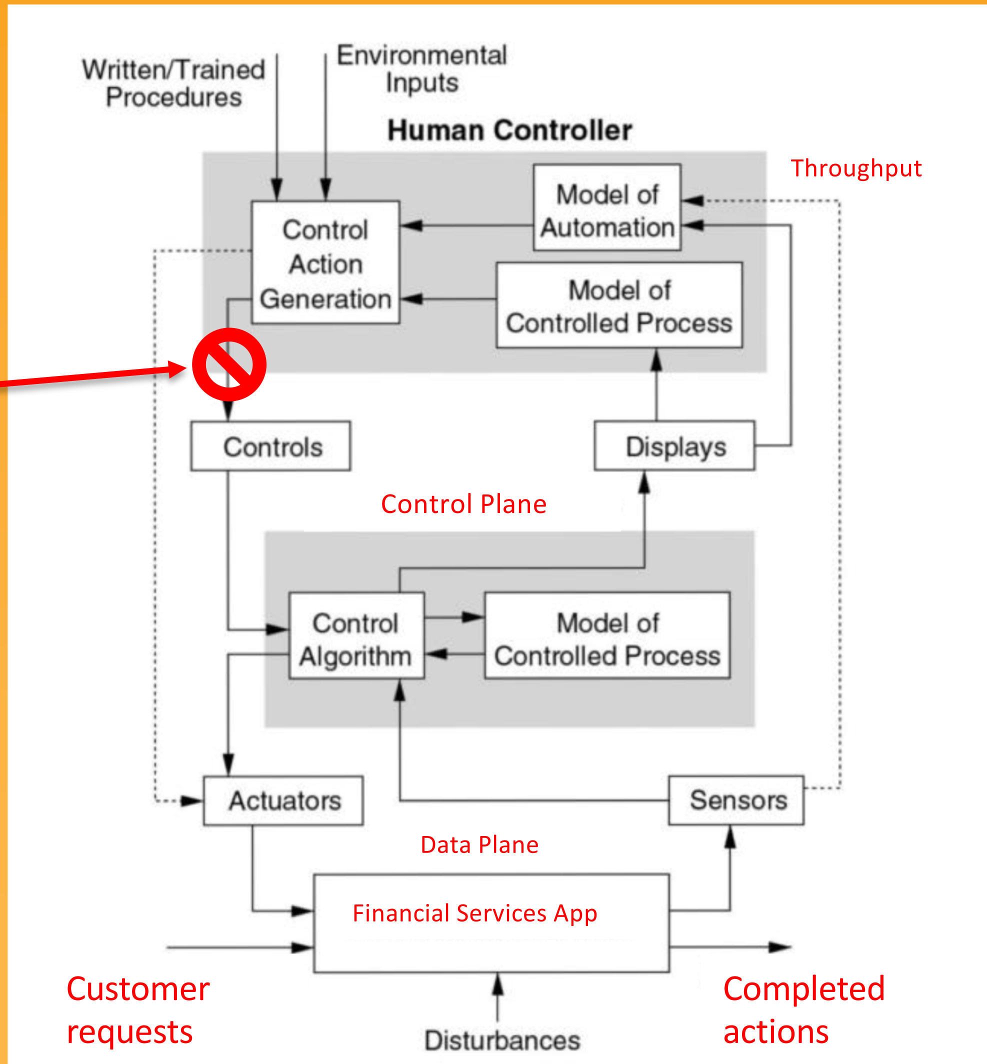
Understand Hazards that could disrupt successful application processing



STPA Hazards

Human Control Action:

Not provided
Unsafe action
Safe but too early
Safe but too late
Wrong sequence
Stopped too soon
Applied too long
Conflicts
Coordination problems
Degradation over time



STPA Hazards

Sensor Metrics:

Missing updates

Zeroed

Overflowed

Corrupted

Out of order

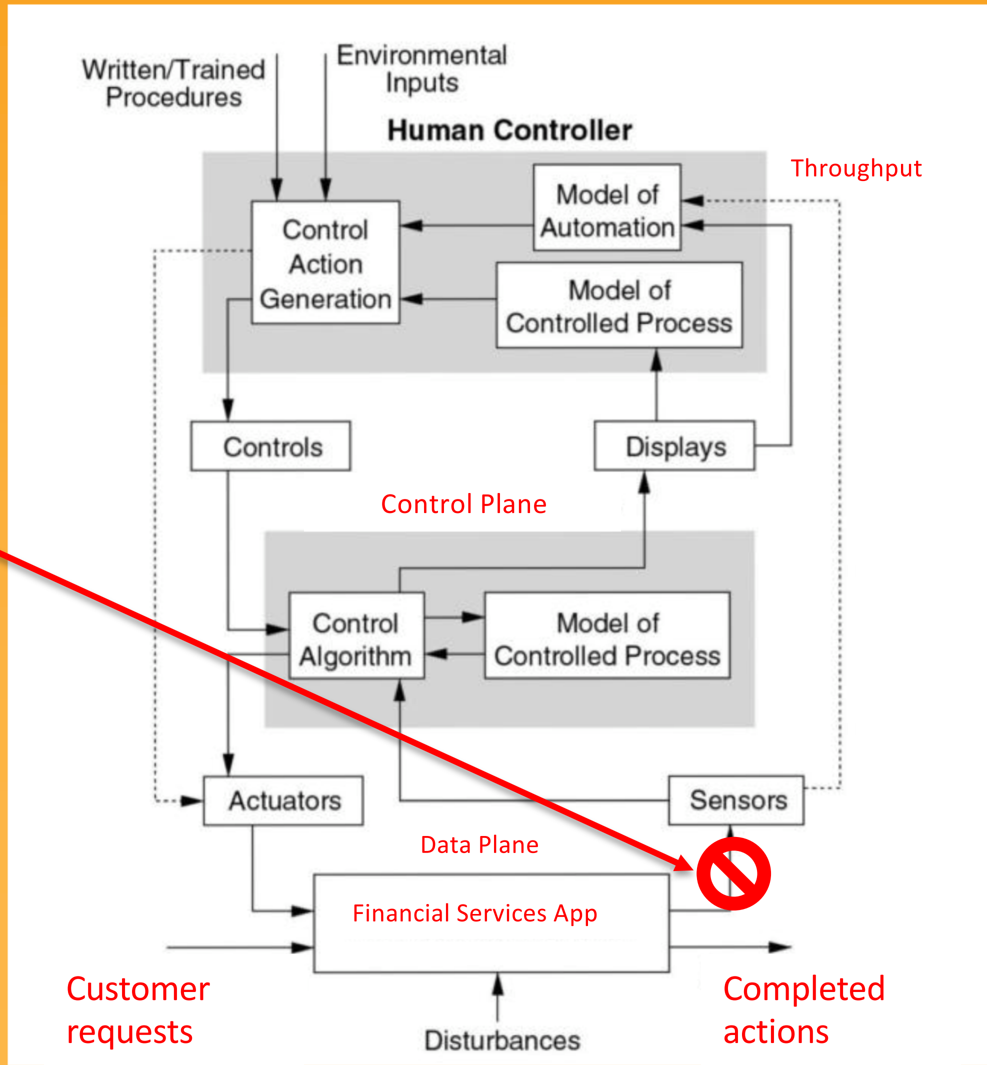
Updates too rapid

Updates infrequent

Updates delayed

Coordination problems

Degradation over time



How do we usually calculate risk?

Severity * Probability = Risk

Assumes that we can determine severity and probability

Assumes we always detect the failure when it occurs

Basic model for financial and economic risk analysis

Failure Modes and Effects Analysis (FMEA)

Engineering oriented risk analysis

Severity * Probability * **Detectability** = Risk

Add observability to mitigate silent failures

Discuss and record component level failure modes

Prioritize mitigation work where it will do most good

FMEA for Web Services - Layered Responsibility

Product Managers and Developers – unique business logic

Software Platform Team – standard components and services

Infrastructure Platform Team – resources, regions and networks

Resilience Engineering – observability and incident management

FMEA Severity Mapped to Infrastructure

Effect	SEVERITY of Effect	Ranking
Hazardous without warning	Earthquake or meteorite destroys datacenter building, no warning, people injured	10
Hazardous with warning	Hurricane or tornado destroys datacenter building, several days warning, people injured	9
Very High	Datacenter flooded, compute and storage systems destroyed, building ok	8
High	Fire in datacenter, suppression system saves building, partial permanent compute and storage loss	7
Moderate	Hardware failure, CPU, disk, or power supply needs replacement. Often occurs after power or cooling failures.	6
Low	Power cut, cooling failure or network partition. Compute and storage returns when power, cooling and network are restored	5
Very Low	System operable with significant degradation of performance	4
Minor	System operable with some degradation of performance	3
Very Minor	System operable with minimal interference	2
None	No effect	1

FMEA Probability Per Service Request

Guess to start with, then measure in production

PROBABILITY of Failure	Failure Prob	Ranking
Very High: Failure is almost inevitable	>1 in 2	**10**
	1 in 3	**9**
High: Repeated failures	1 in 8	**8**
	1 in 20	**7**
Moderate: Occasional failures	1 in 80	**6**
	1 in 400	**5**
	1 in 2,000	**4**
Low: Relatively few failures	1 in 15,000	**3**
	1 in 150,000	**2**
Remote: Failure is unlikely	<1 in 1,500,000	**1**

FMEA Detectability

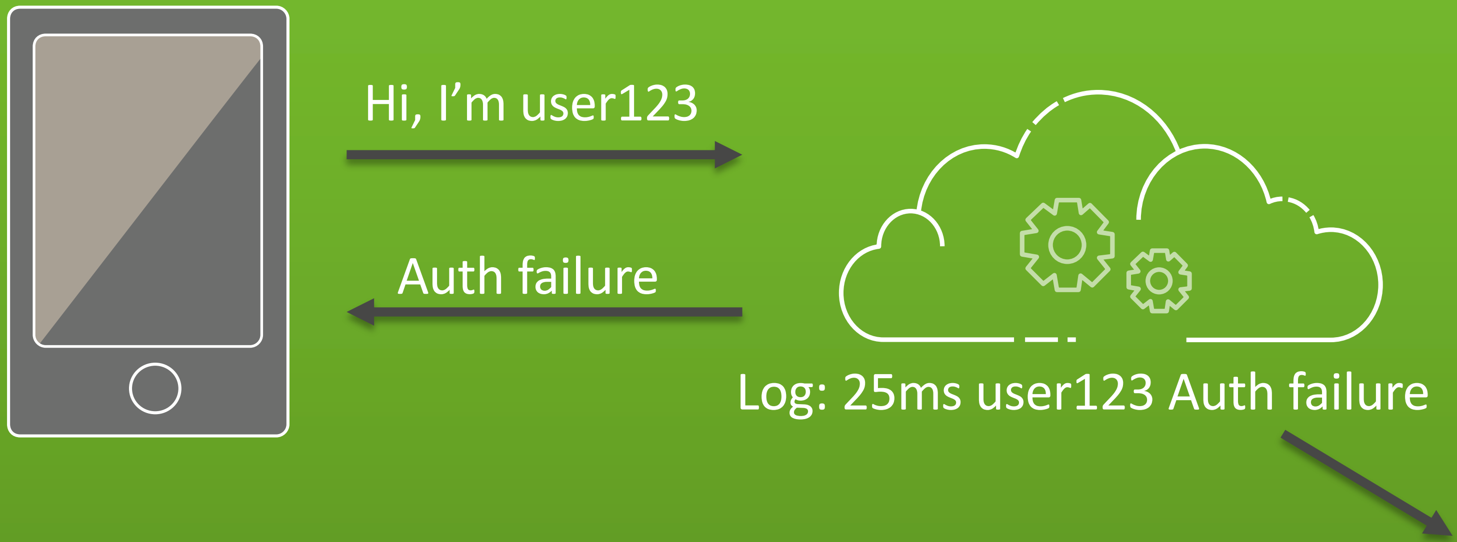
Needs an observable monitoring alert to detect a failure

Detection	Likelihood of DETECTION by Design Control	Ranking
Absolute Uncertainty	Design control cannot detect potential cause/mechanism and subsequent failure mode	**10**
Very Remote	**Very remote chance the design control will detect potential cause/mechanism and subsequent failure mode**	**9**
Remote	**Remote chance the design control will detect potential cause/mechanism and subsequent failure mode**	**8**
Very Low	**Very low chance the design control will detect potential cause/mechanism and subsequent failure mode**	**7**
Low	**Low chance the design control will detect potential cause/mechanism and subsequent failure mode**	**6**
Moderate	**Moderate chance the design control will detect potential cause/mechanism and subsequent failure mode**	**5**
Moderately High	**Moderately High chance the design control will detect potential cause/mechanism and subsequent failure mode**	**4**
High	**High chance the design control will detect potential cause/mechanism and subsequent failure mode**	**3**
Very High	**Very high chance the design control will detect potential cause/mechanism and subsequent failure mode**	**2**
Almost Certain	Design control will detect potential cause/mechanism and subsequent failure mode	**1**

FMEA Example

Customer is trying to make a request to a service

what could go wrong?



FMEA Example

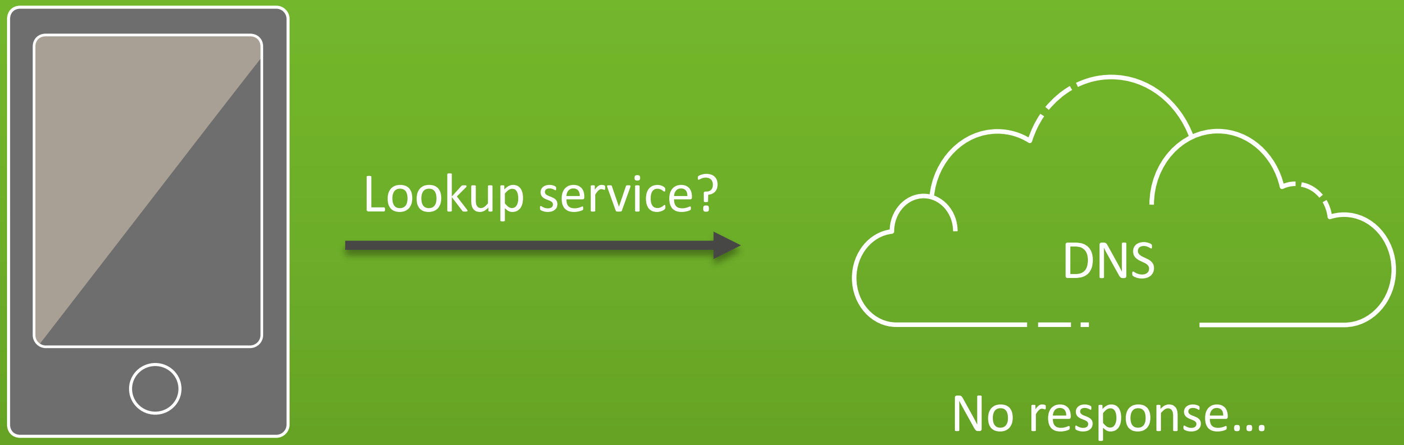
Authentication Failures

Item / Function	Potential Failure Mode(s)	Potential Effect(s) of Failure	Sev	Potential Cause(s)/ Mechanism(s) of Failure	Prob	Current Design Controls	Det	RPN	Recommended Action(s)
Authentication	Client can't authenticate	Can't connect application	5	Certificate timeout, version mismatch, account not setup, credential changed	3	Log and alert on authentication failures	3	**45**	
	Slow or unreliable authentication	Slow start for application	4	Auth service overloaded, high error and retry rate	3	Log and alert on high authentication latency and errors	4	**48**	

FMEA Example

Customer is trying to obtain an IP address for a service

what could go wrong?



FMEA Example – see paper for more failure modes

								0	
Client Request to API Endpoint	Service unknown, address un-resolvable	Delay while discovery or DNS times out, slow fallback response	5	DNS configuration error, denial of service attack, or provider failure	1	Customer eventually complains via call center	10	**50**	Dual redundant DNS, fallback to local cache, hardcoded IP addresses. Endpoint monitoring and alerts
	Service unreachable, request undeliverable	Fast fail, no response	4	Network route down or no service instances running	1	Autoscaler maintains a number of healthy instances	1	**4**	Endpoint monitoring and alerts
	Service reachable, request undeliverable	Connect timeout, slow fail, no response	4	Service frozen/not accepting connection	1	Retry request on different instance. Healthcheck failed instances removed. Log and alert.	2	**8**	
	Request delivered, no response - stall	Application request timeout, slow fail, no response	4	Broken service code, overloaded CPU or slow dependencies	1	Retry request on different instance. Healthcheck failed instances removed. Log and alert.	2	**8**	

STPA – Top down focus on control hazards

FMEA – Bottom up focus on prioritizing failure modes

- STPA tends to have better failure coverage than FMEA
- Both are useful

Good Resilience Practices

Rule of 3 – three ways for critical operations to succeed

Synchronous data replication over three zones in a region

DR failover from primary region to either of two secondary regions

Active-Active-Active workloads across three regions

Good Resilience Practices

Fail up - DR failover between regions

- From smaller capacity region to larger capacity region
- From distant region to closer (lower latency) region

Good Resilience Practices

Chaos first

- Build your resilience environment *before* introducing apps to it
- Automated continuous zone and region failover testing
- Make it a “badge of honor” to have an app pass the chaos test

Good Resilience Practices

Continuous Resilience

Continuous Delivery needs Test Driven Development and Canaries

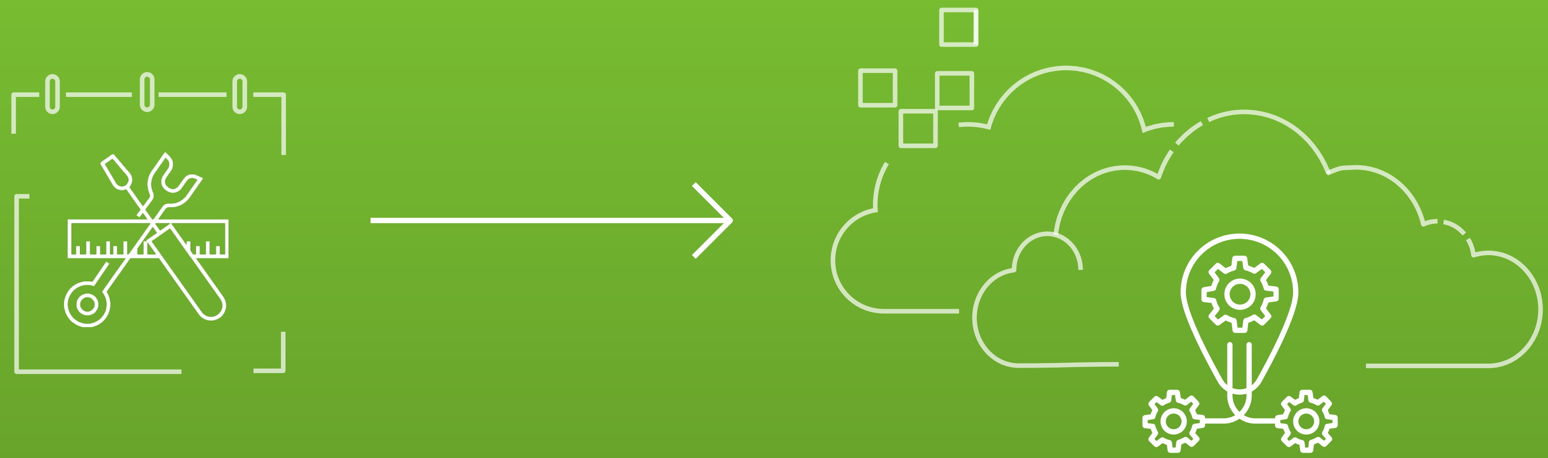
Continuous Resilience needs automation in both test and production

Make failure mitigation into a well tested code path and process

Call it Chaos Engineering if you like, it's the same thing…

Cloud provides the automation that leads to chaos engineering

As datacenters migrate to cloud, fragile and manual disaster recovery processes can be standardized and automated



Testing failure mitigation will move from a scary annual experience to automated **continuous resilience**

Safe and Sound: Continuous Resilience

Paper: Building Mission Critical Financial Services Applications on AWS
By Pawan Agnihotri with contributions by Adrian Cockcroft

Adrian Cockcroft

@adrianco

AWS VP Cloud Architecture Strategy