Chaos Engineering

Niclas Gothberg

Why?

- Distributed systems are fragile and have become much more complex
- Disturbances have become harder to predict
- We all depend on uptime of systems to a greater extent
- Financial impact of system downtown
- Failures are given

Availability (SLA)	Downtime per year
95%	18 days 6 hours
99%	3 days 15 hours
99.9%	8 hours 45 minutes

Resilience* is engineered

* “The ability of a workload to handle and recover from unexpected conditions. . .”

- Infrastructure
- Network
- Data
- Application
- People & Processes

"Chaos Engineering is the discipline of experimenting on a system in order to build confidence in the system's capability to withstand turbulent conditions in production."

Principles of Chaos Engineering
http://principlesofchaos.org

REL 12. How do you test reliability? Info

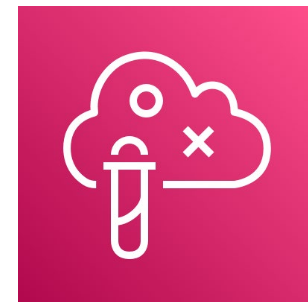
After you have designed your workload to be resilient to the stresses of production, testing is the only way to ensure that it will operate as designed, and deliver the resiliency you expect.

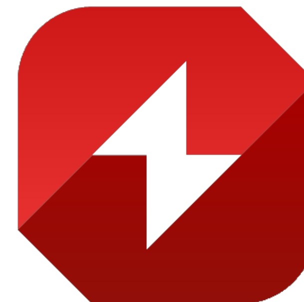
Question does not apply to this workload Info

Select from the following

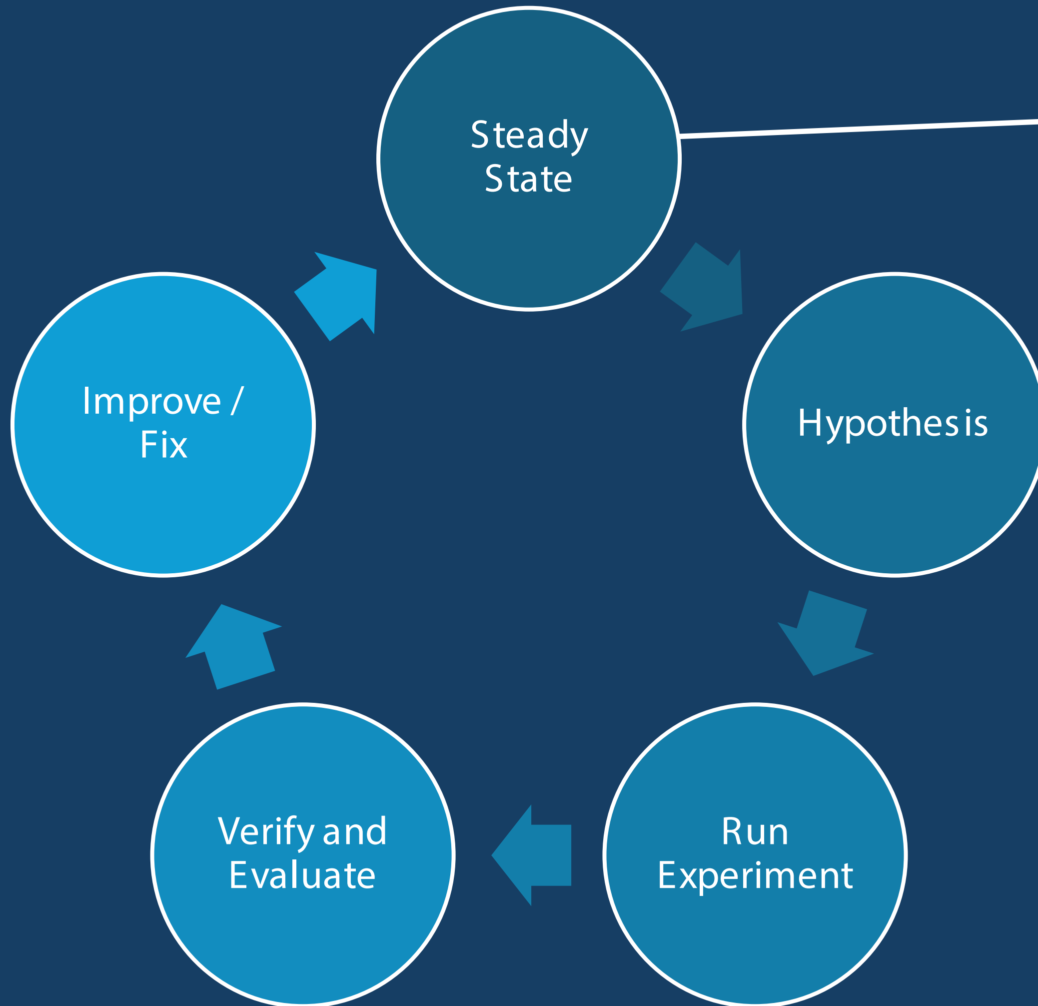
- [ ] Use playbooks to investigate failures Info
- [ ] Perform post-incident analysis Info
- [ ] Test functional requirements Info
- [ ] Test scaling and performance requirements Info
- [ ] Test resiliency using chaos engineering Info
- [ ] Conduct game days regularly Info

And Others….





Phases of Chaos Engineering



Set of measurements that indicates "normal" behavior of a system from a business perspective, and within a given set of tolerances

Chaos Experiment - Example

What if we had slow response times in our Analytics cluster

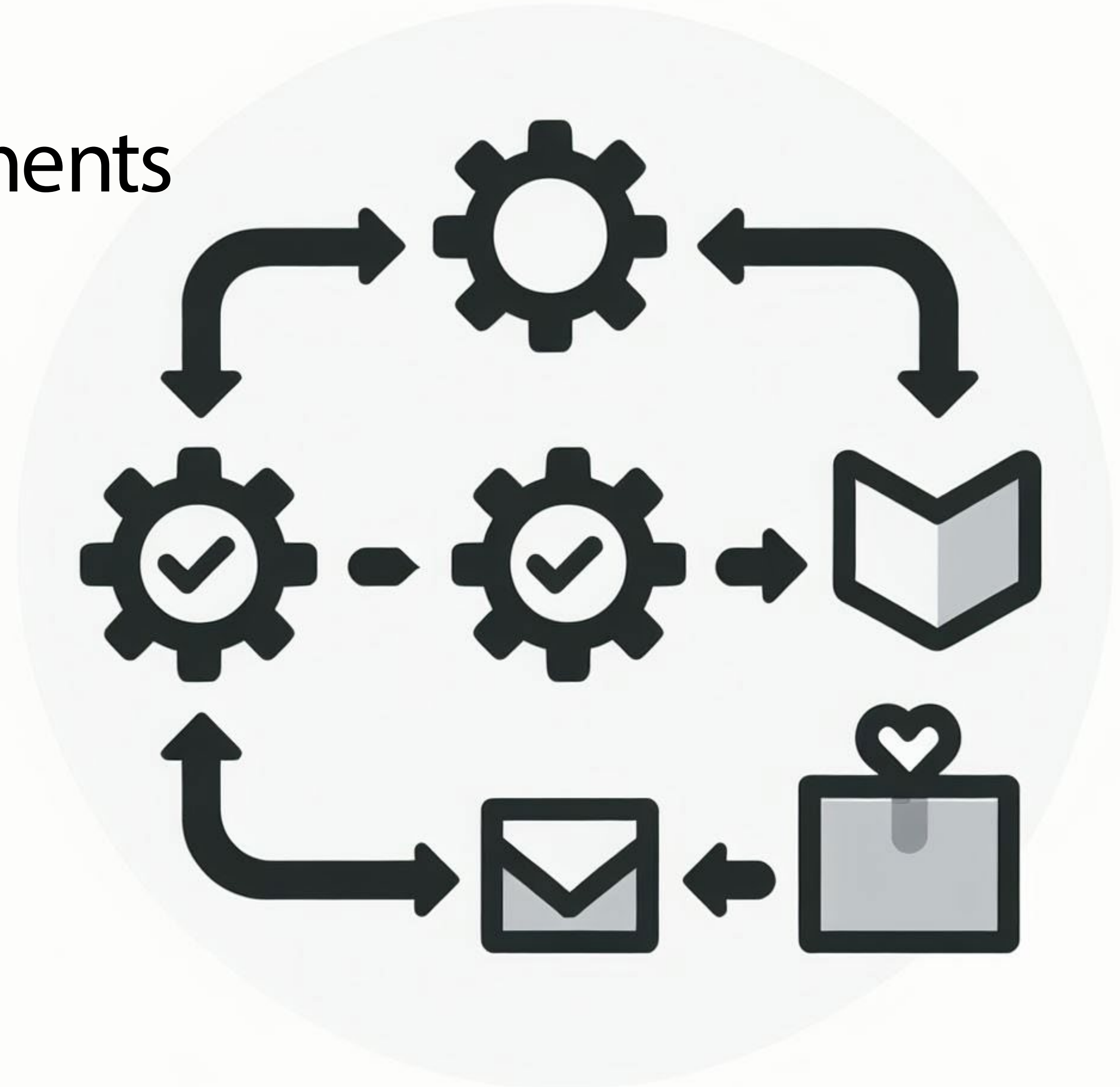
Hypothesis	No customer impact expected
Conditions	Duration: 10 min Failure Injection: Latency 400-600ms Targets: 70% of all request
Result	10% increase in timeout errors 2s latency increase for user wait time Alert threshold set to high

Benefits of Chaos Engineering

- Help reveal blind spots in your observability
- Promotes a “non-blaming” culture
- Inject failures to build immunity
- Unit, Integration, System testing usually don’t consider operating conditions

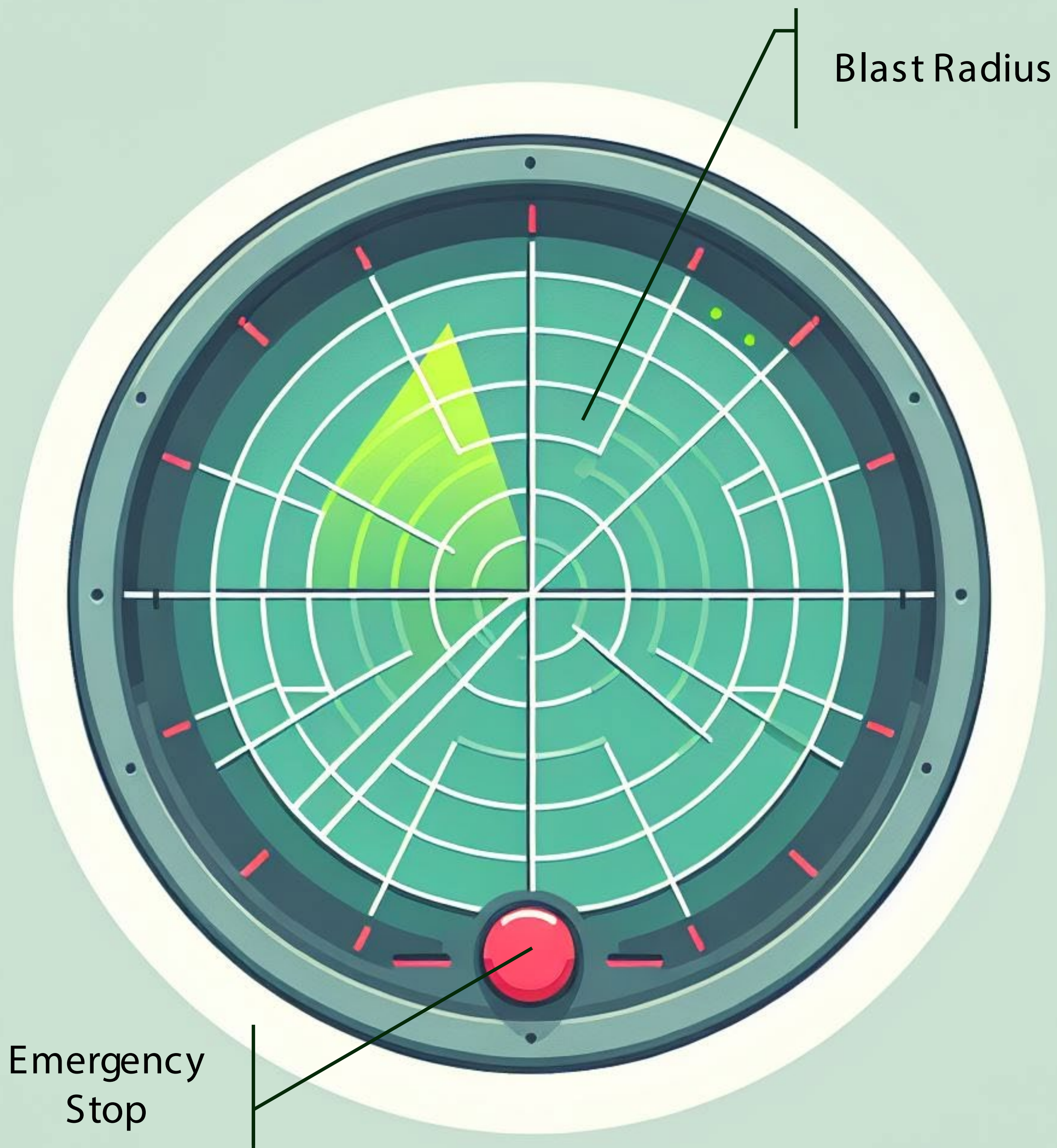
When to run your experiments

- CI/CD pipeline
- Scheduled activity
- Gamedays



Challenges

- Teams might not have the full picture
- Cultural change / internal politics
- Technical debt
- Security
- Invested in a specific technical roadmap



Some Best Practices

- Test as close to Production as possible
  - Avoid assumptions when defining operating conditions
  - Systems can behave differently depending on environment and traffic patterns
- Avoid experiments you know will break
- Limit Blast Radius
- Always make sure you have an emergency stop

Thank you!