CHAOS!

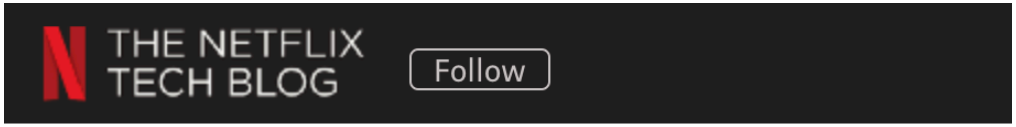
Breaking your systems to make them unbreakable

"You can't let your failures define you.
You have to let your failures teach you."

–Barack Obama

"Hope is not a strategy."

– Corey Bertram



3. The best way to avoid failure is to fail constantly.

http://bit.ly/netflix-5-things

Chaos Monkey

JASON YEE

Sr. Technical Evangelist
Travel Hacker
Whiskey Hunter
Pokemon Trainer
Conference Organizer

Tw: @gitbisect
Em: jyee@datadoghq.com

DATADOG

SaaS-based observability platform:

- Metrics
- Traces (APM)
- Logs
- Synthetics & RUM
- and more!

Tw: @datadoghq

We're hiring:
jobs.datadoghq.com

"By running Chaos Monkey in the middle of a business day, in a carefully monitored environment with engineers standing by to address any problems, we can still learn the lessons about the weaknesses of our system, and build automatic recovery mechanisms to deal with them.

So next time an instance fails at 3 am on a Sunday, we won't even notice."

–Netflix Technology Blog, 2011
http://bit.ly/netflix-chaos

"By running Chaos Monkey in the *middle of a business day*, in a carefully monitored environment with engineers standing by to address any problems, we can still learn the lessons about the weaknesses of our system, and build automatic recovery mechanisms to deal with them.

So next time an instance fails at 3 am on a Sunday, we won't even notice."

–Netflix Technology Blog, 2011
http://bit.ly/netflix-chaos

"By running Chaos Monkey in the *middle of a business day*, in a *carefully monitored environment* with engineers standing by to address any problems, we can still learn the lessons about the weaknesses of our system, and build automatic recovery mechanisms to deal with them.

So next time an instance fails at 3 am on a Sunday, we won't even notice."

–Netflix Technology Blog, 2011
http://bit.ly/netflix-chaos

"By running Chaos Monkey in the *middle of a business day*, in a *carefully monitored environment* with *engineers standing by* to address any problems, we can still learn the lessons about the weaknesses of our system, and build automatic recovery mechanisms to deal with them.

So next time an instance fails at 3 am on a Sunday, we won't even notice."

–Netflix Technology Blog, 2011
http://bit.ly/netflix-chaos

"By running Chaos Monkey in the *middle of a business day*, in a *carefully monitored environment* with *engineers standing by* to address any problems, we can still *learn the lessons* about the weaknesses of our system, and build automatic recovery mechanisms to deal with them.

So next time an instance fails at 3 am on a Sunday, we won't even notice."

–Netflix Technology Blog, 2011
http://bit.ly/netflix-chaos

Don’t be a jerk!

Game Days

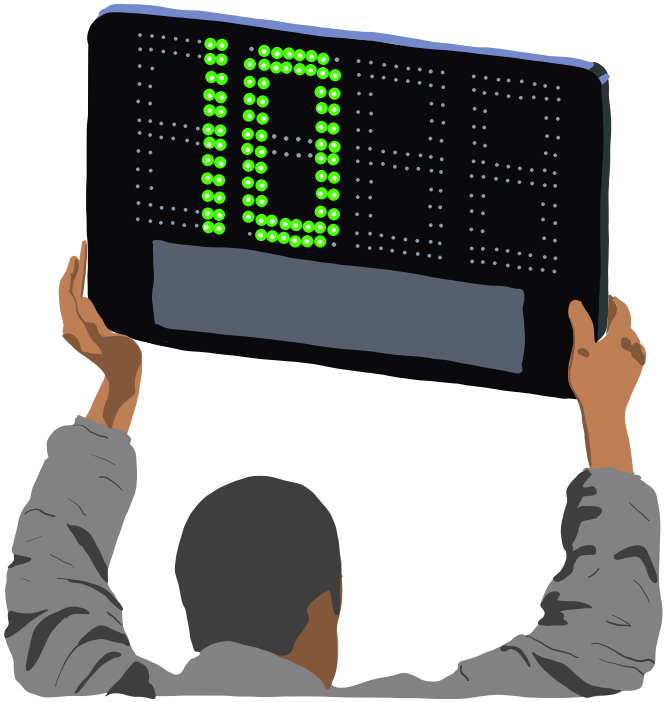
90 Minutes

90 Minutes

30 minutes planning.

90 Minutes

30 minutes planning.
50 minutes playing.



90 Minutes

30 minutes planning.

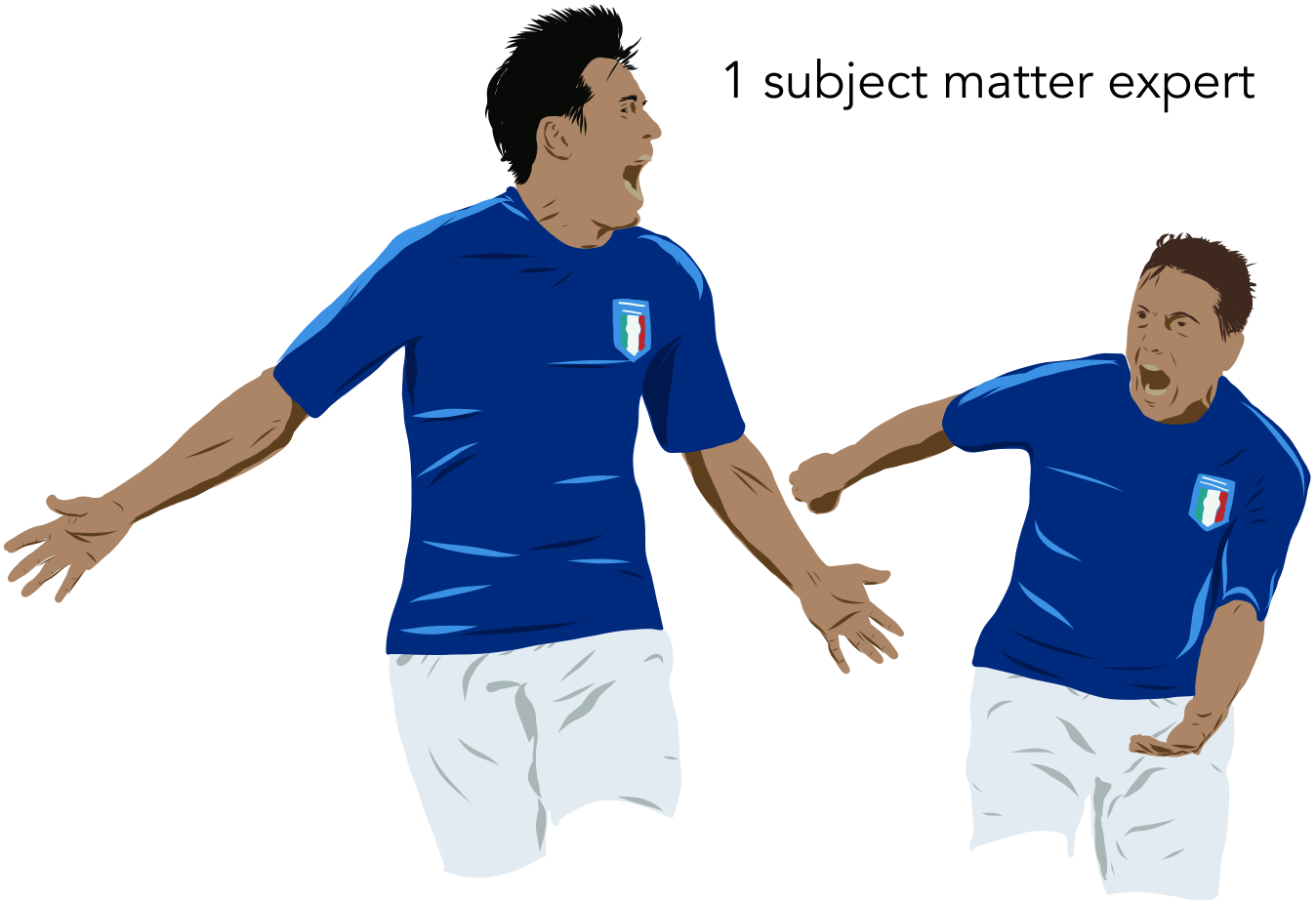
50 minutes playing.

10 minutes reporting.

The Team

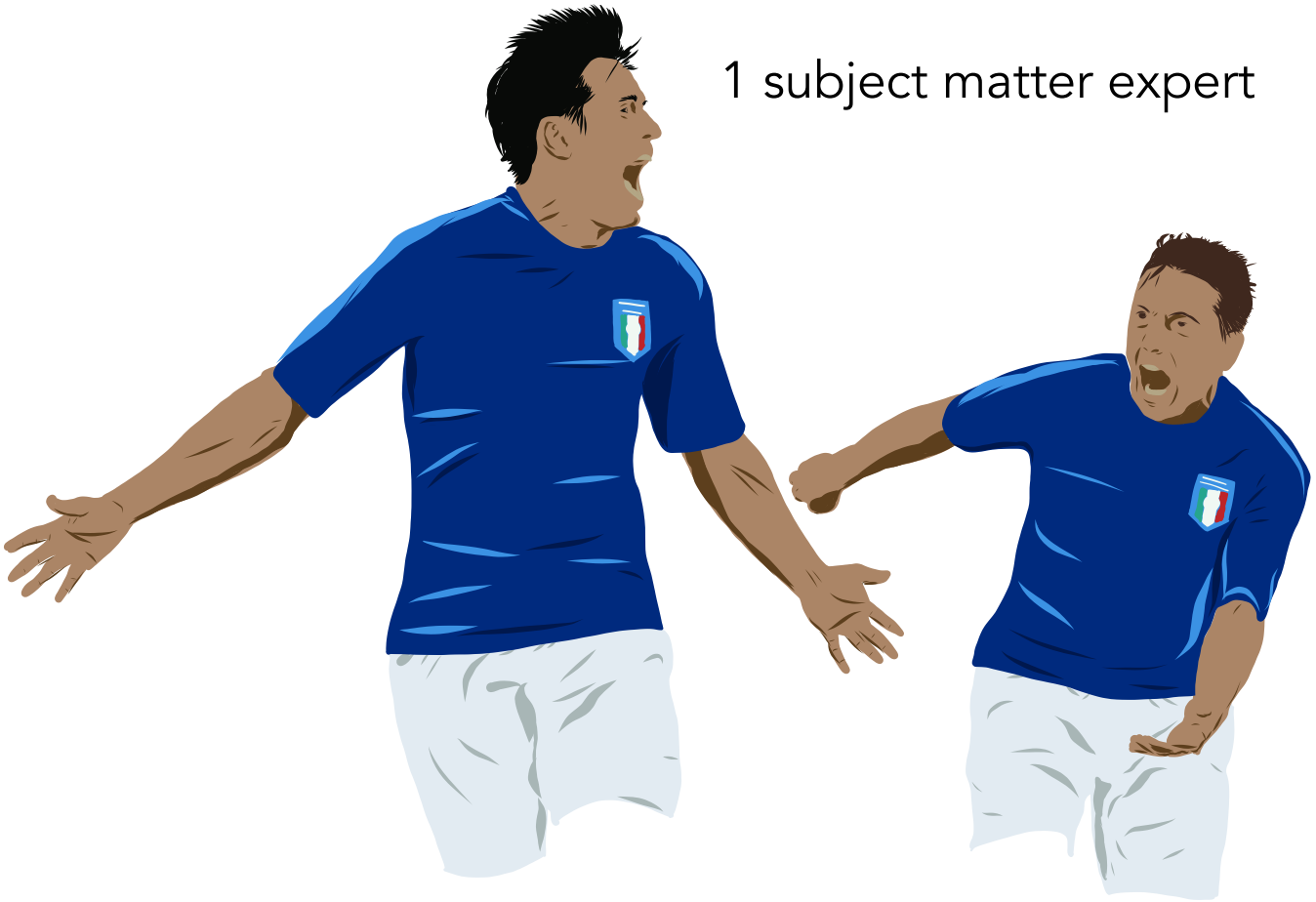
The Team

1 subject matter expert



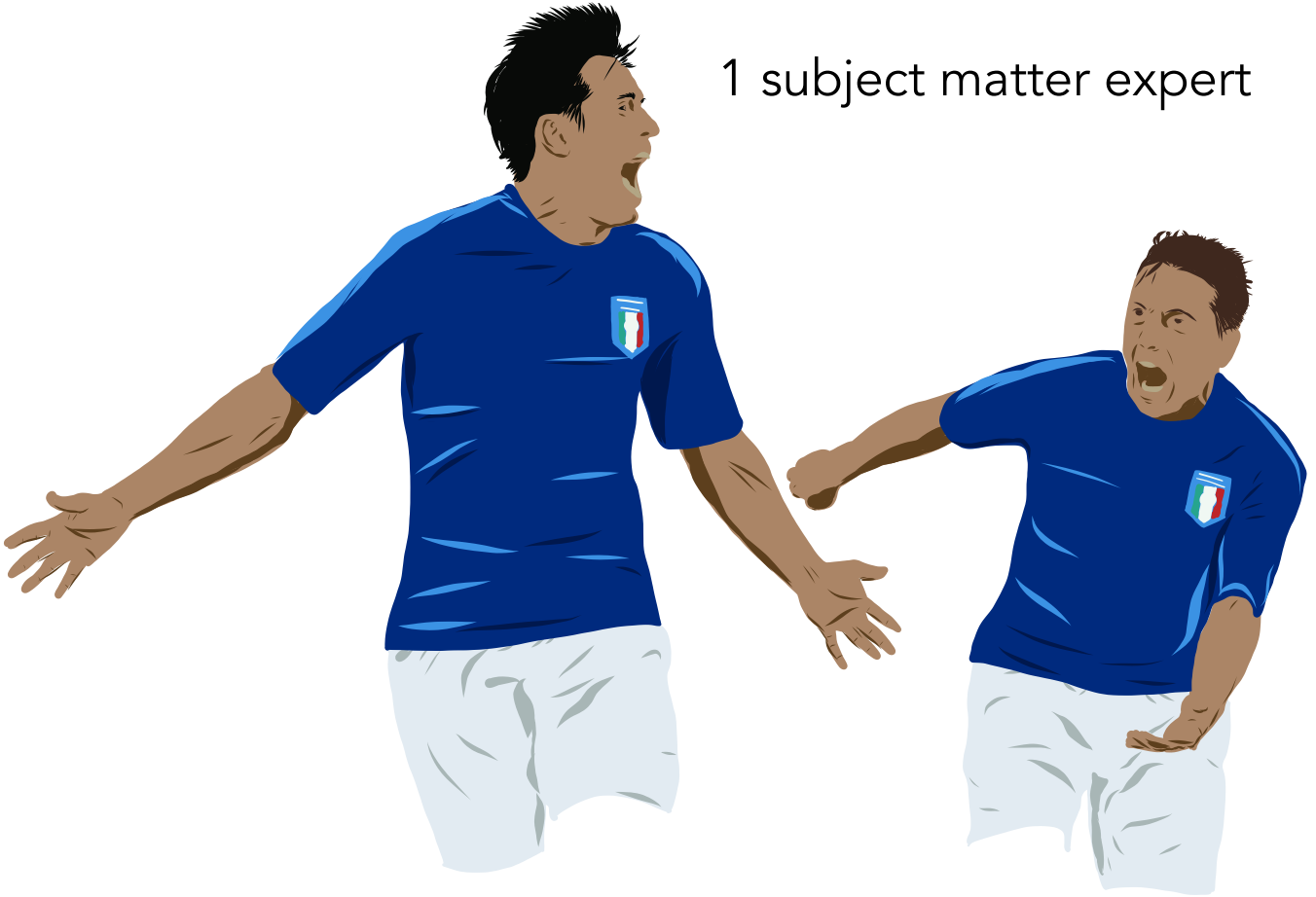
The Team

1 subject matter expert



1 SRE

The Team



1 subject matter expert

1 SRE

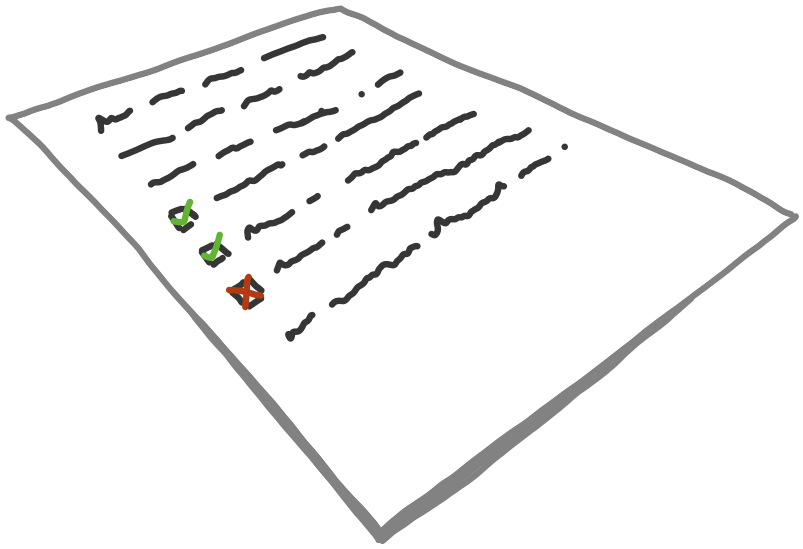
Sometimes, 1 junior engineer/developer

Before

- Schedule it.

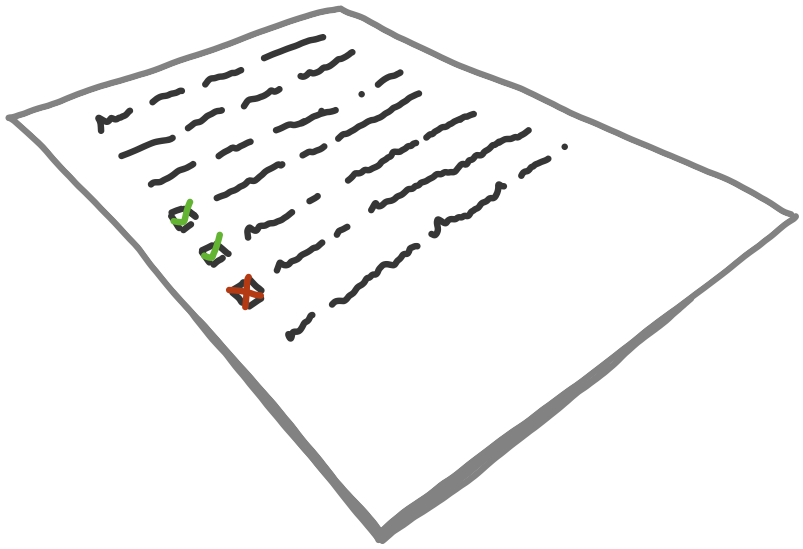
Before

- Schedule it.
- Pick tests. Start easy.



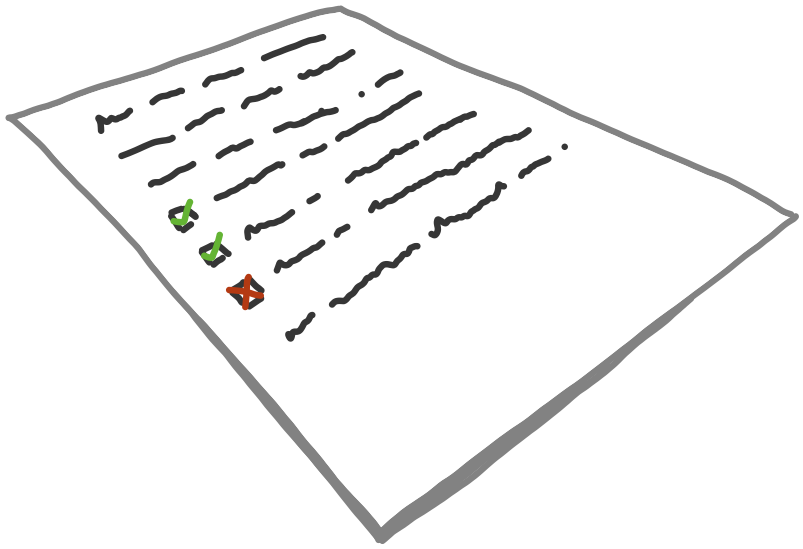
Before

- Schedule it.
- Pick tests. Start easy.
- Write down what you expect to happen.



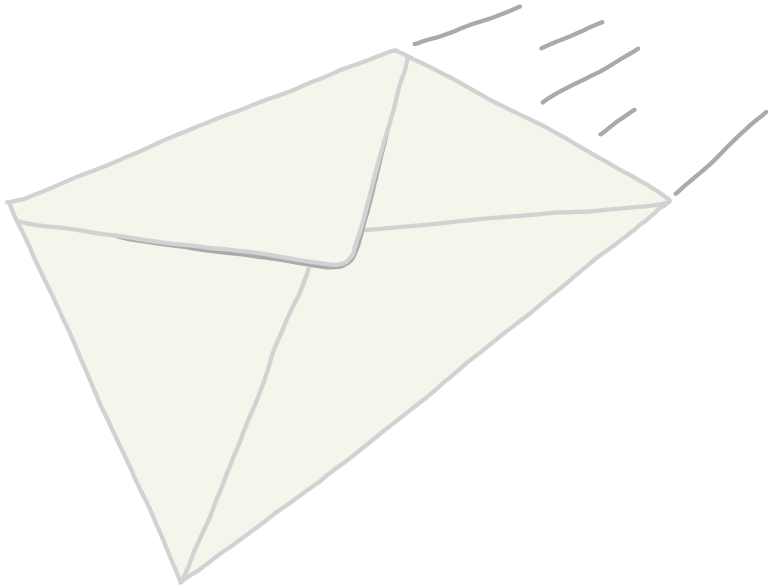
Before

- Schedule it.
- Pick tests. Start easy.
- Write down what you expect to happen.
- Write down your plan if things go wrong.



Before

- Schedule it.
- Pick tests. Start easy.
- Write down what you expect to happen.
- Write down your plan if things go wrong.
- Share your document!



During

- Staging -> Production (off-peak) -> Production (primetime)

During

- Staging -> Production (off-peak) -> Production (primetime)
- Announce start in group chat.

During

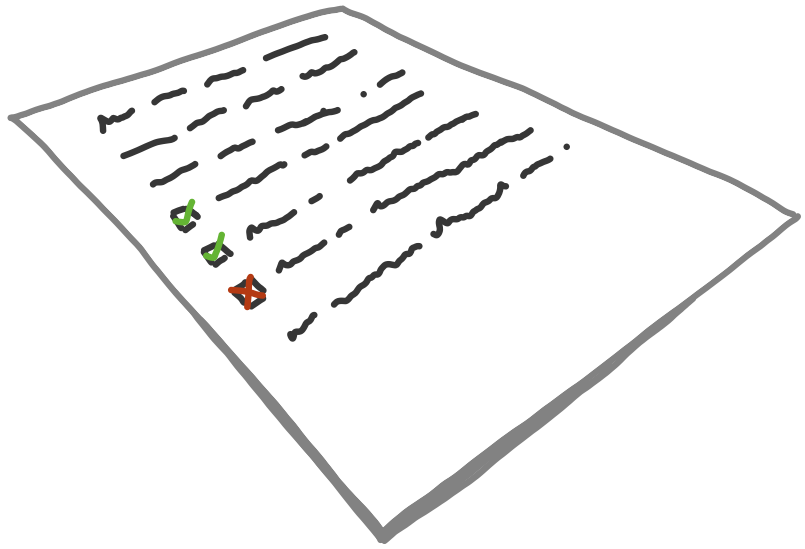
- Staging -> Production (off-peak) -> Production (primetime)
- Announce start in group chat.
- Maintain discussion in group chat.

During

- Staging -> Production (off-peak) -> Production (primetime)
- Announce start in group chat.
- Maintain discussion in group chat.
- Monitor for outages.

During

- Staging -> Production (off-peak) -> Production (primetime)
- Announce start in group chat.
- Maintain discussion in group chat.
- Monitor for outages.
- Run your test *and take notes!*

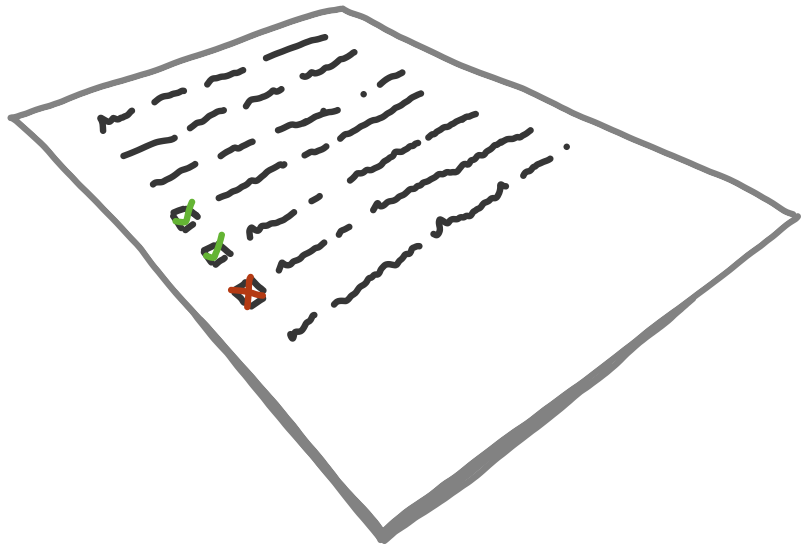


After

- Create cards/tickets to track issues that need work.

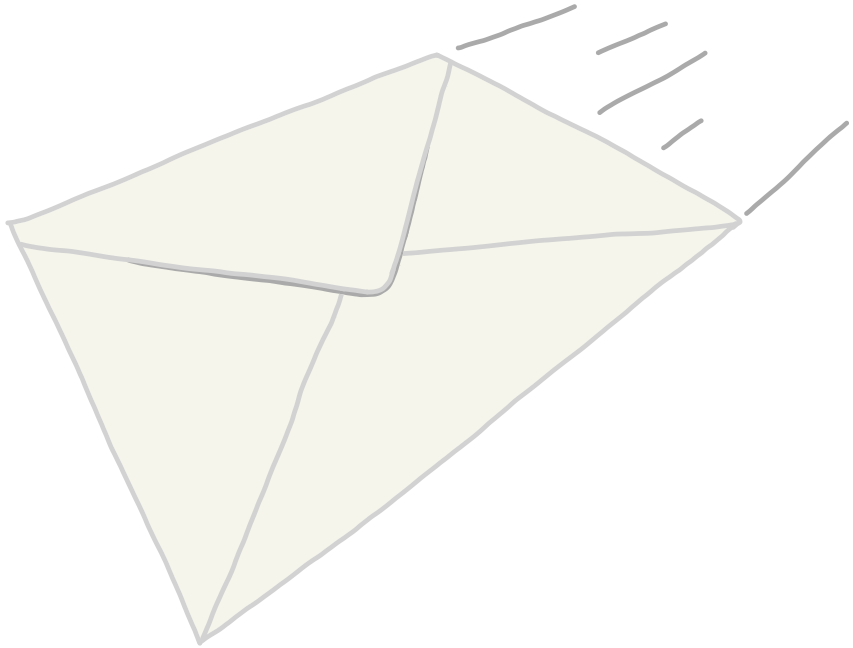
After

- Create cards/tickets to track issues that need work.
- Write a summary & key lessons.



After

- Create cards/tickets to track issues that need work.
- Write a summary & key lessons.
- Email to all of engineering.



After

- Create cards/tickets to track issues that need work.
- Write a summary & key lessons.
- Email to all of engineering.
- Celebrate!

Game Levels

0. Terminate the service. Block access to 1 dependency.

Game Levels

0. Terminate the service. Block access to 1 dependency.

1. Block access to all dependencies.

Game Levels

0. Terminate the service. Block access to 1 dependency.

1. Block access to all dependencies.

2. Terminate the host.

Game Levels

0. Terminate the service. Block access to 1 dependency.

1. Block access to all dependencies.

2. Terminate the host.

3. Degrade the environment.
   Monitor & alert on Latency, Errors, Traffic, & Saturation

Game Levels

0. Terminate the service. Block access to 1 dependency.

1. Block access to all dependencies.

2. Terminate the host.

3. Degrade the environment.

4. Spike traffic.

Game Levels

0. Terminate the service. Block access to 1 dependency.

1. Block access to all dependencies.

2. Terminate the host.

3. Degrade the environment.

4. Spike traffic.

5. Terminate the region/cloud.

Experiment time!

Review

Share!

Plan!

Have fun!

Additional Resources

- systemctl, kill, iptables, stress-ng

Additional Resources

- systemctl, kill, iptables, stress-ng
- Comcast - https://github.com/tylertreat/comcast

Additional Resources

- systemctl, kill, iptables, stress-ng
- Comcast - https://github.com/tylertreat/comcast
- Vegeta - https://github.com/tsenart/vegeta

Additional Resources

- systemctl, kill, iptables, stress-ng
- Comcast - https://github.com/tylertreat/comcast
- Vegeta - https://github.com/tsenart/vegeta
- Kubernetes & Istio - https://istio.io/docs/tasks/traffic-management/fault-injection/

Additional Resources

- Gremlin - https://www.gremlin.com/
- ChaosToolkit - https://chaostoolkit.org/
- Verica? - https://verica.io