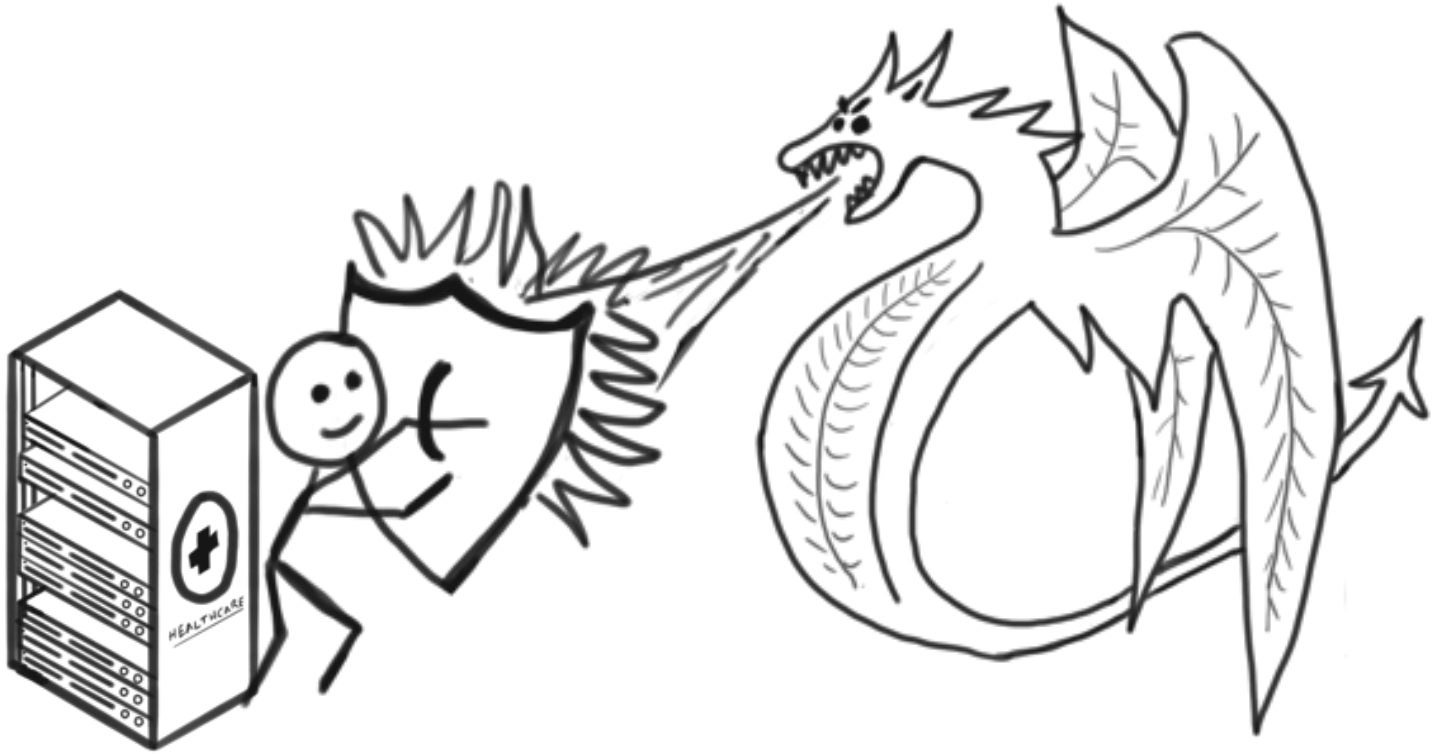
Building Confidence in Healthcare Systems Through Chaos Engineering



Carl Chesser

@che55er | che55er.io

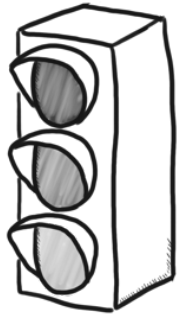
Cerner

JOE'S KANSAS CITY BAR-B-QUE

About me

Our Story

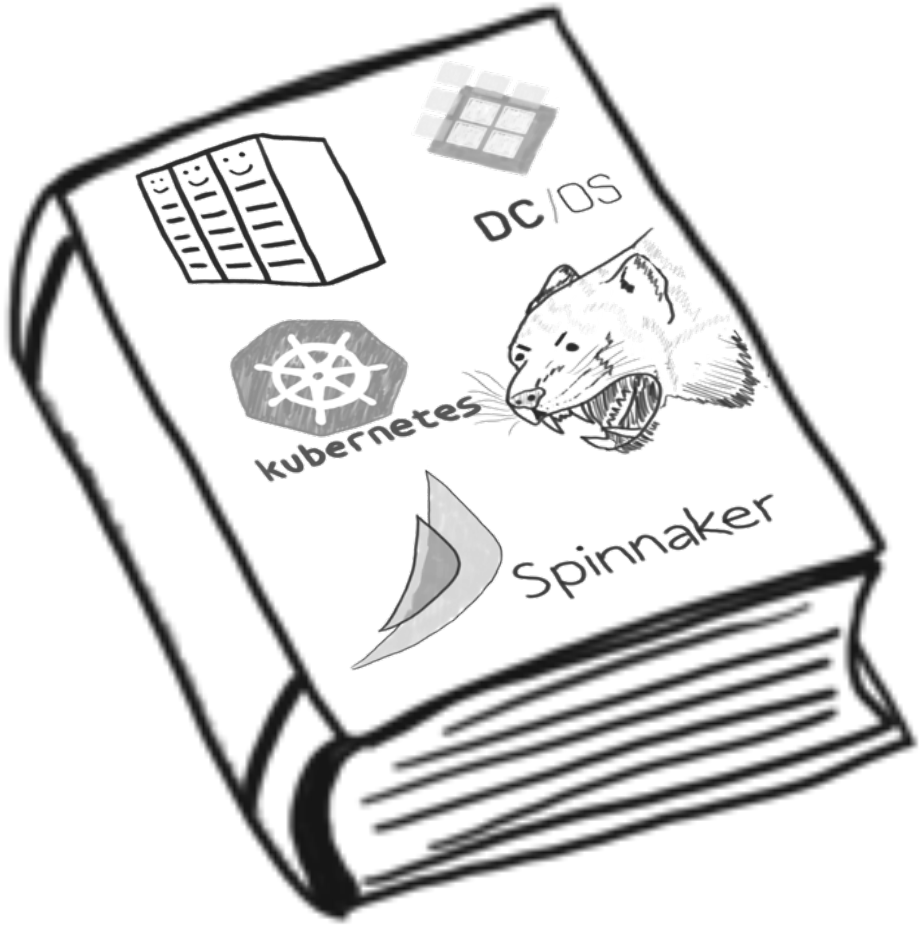
Traffic Management Patterns



Introducing Chaos Experiments

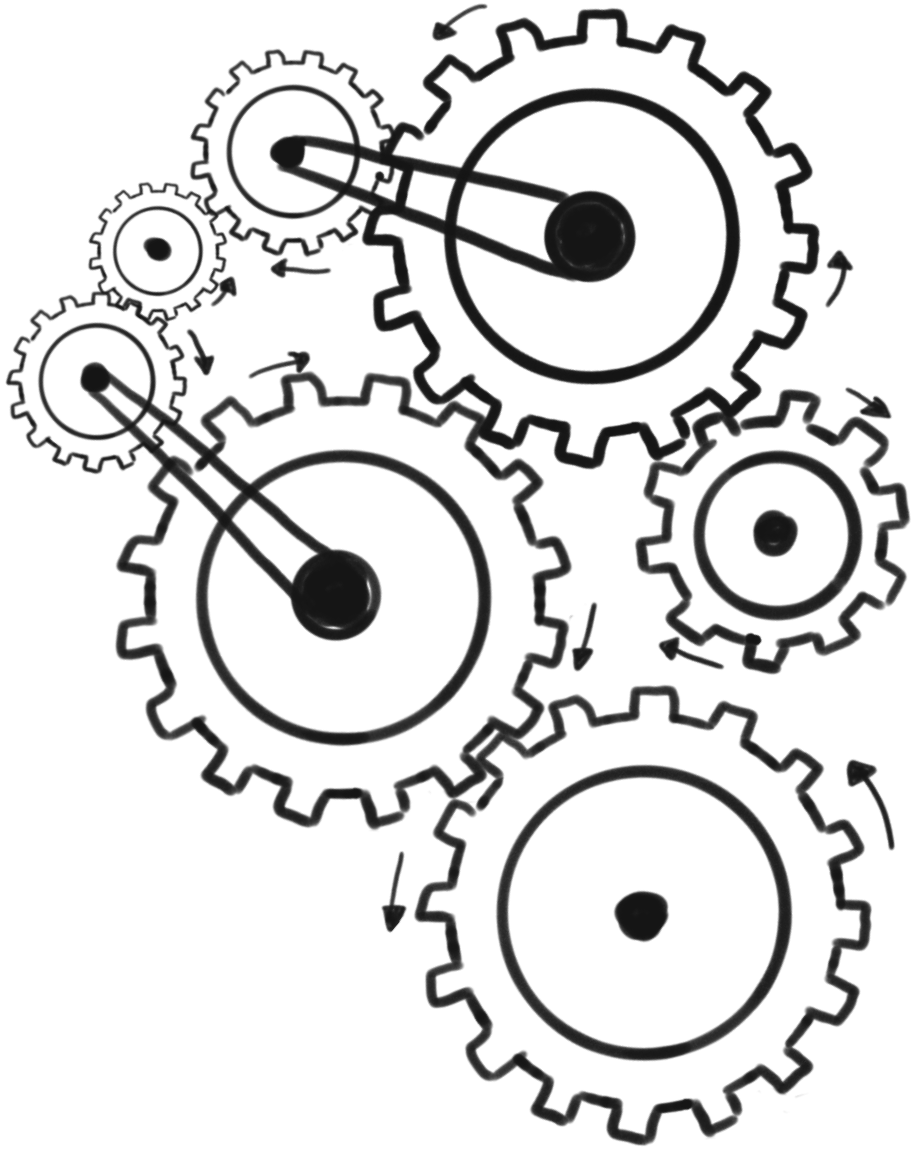
Summary

Our Story



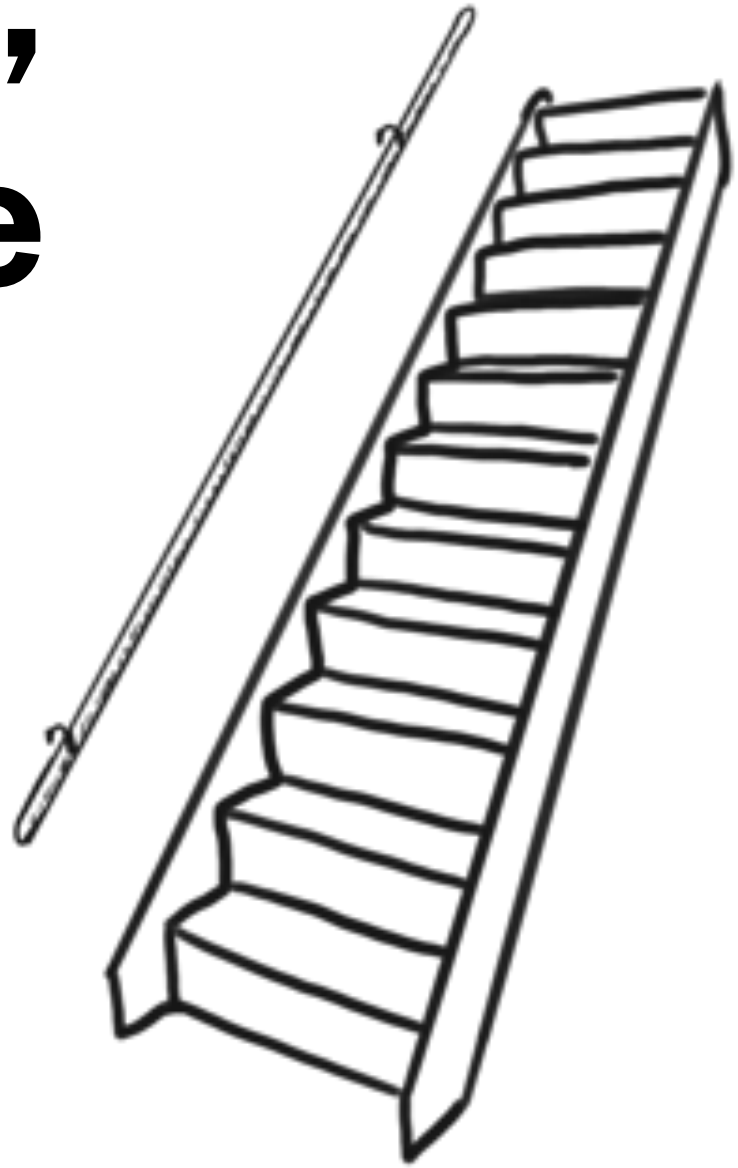
The Challenge

Changing existing service deployments in a complex deployment environment supporting critical workloads.



Incrementally Build, Allowing for Change

We wanted to control our deployments through declarative configuration, that would allow us to continue to change.

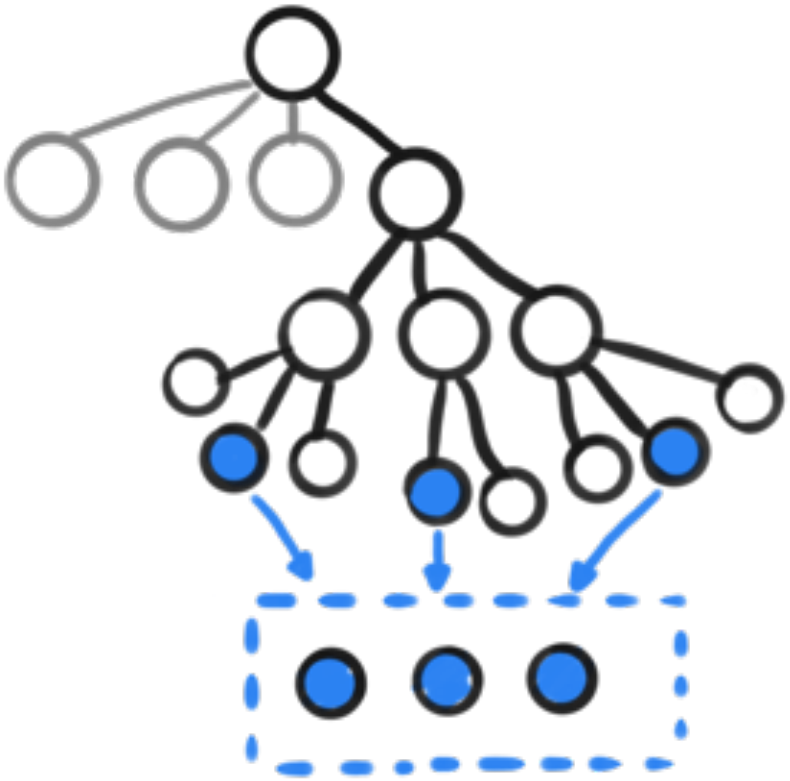


Complexity was Growing within our Systems

As we pursued more ways of increasing availability and infrastructure features, our systems grew in size and complexity.

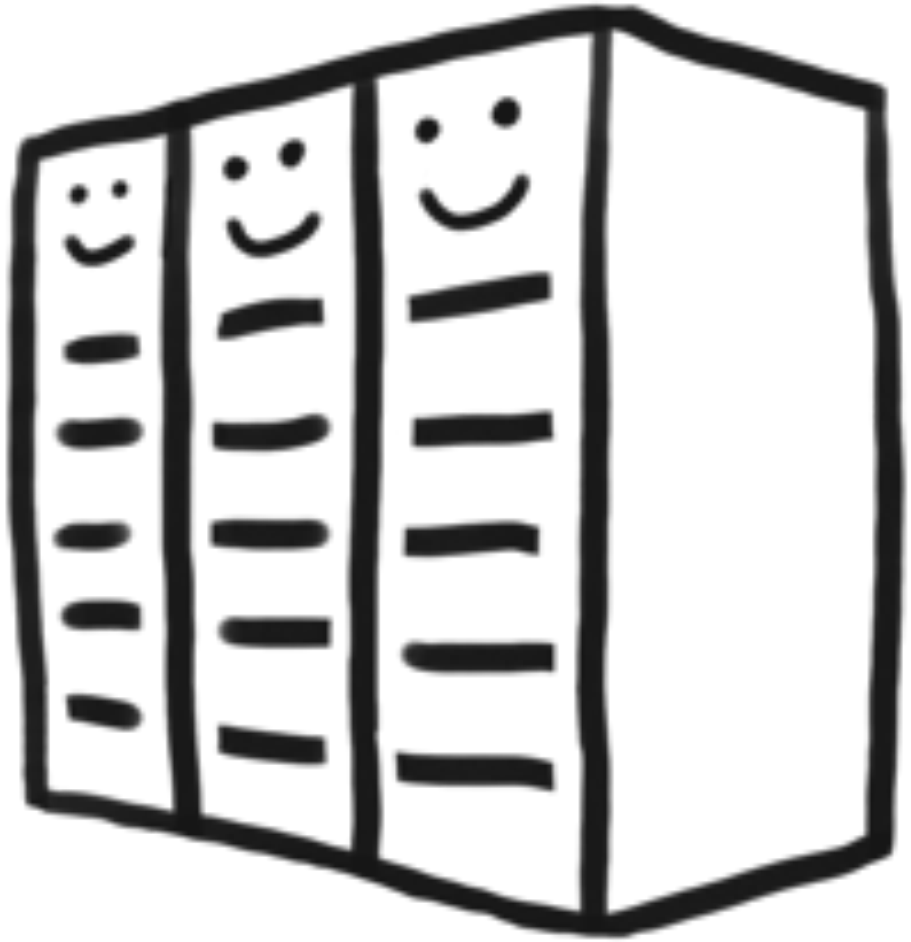
Cross Functional Team Alignment for Experiments

As we built and tested our infrastructure, we wanted cross team alignment when evaluating the layers of the infrastructure.



Introducing OpenStack

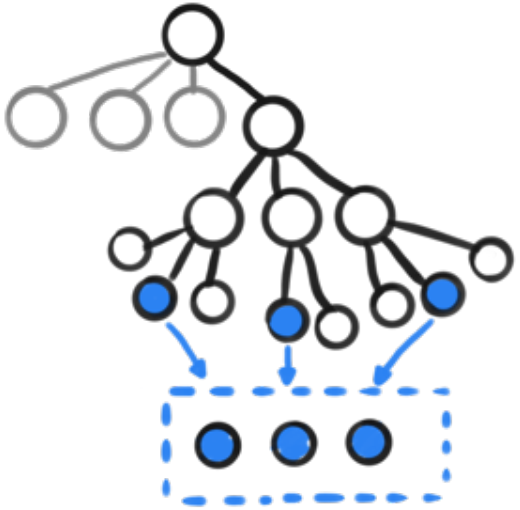
We wanted to have clean availability zone separation, and therefore wanted to ensure we didn't have shared resources.



Introduce the Tiger Team

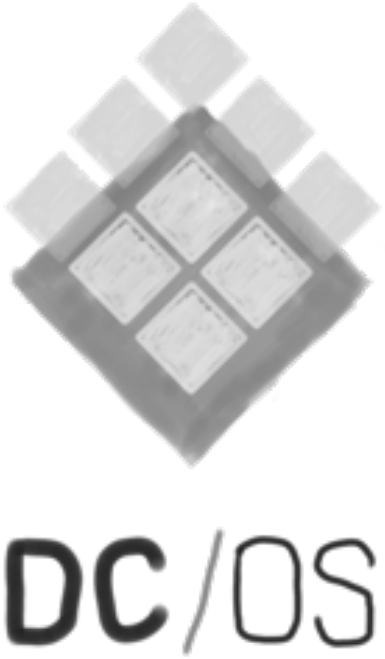
We had three different organizations, but wanted one cross functional team.

Platform and Operations already were located together, and needed to get our Infrastructure team located together as well.



Starting with DC/OS

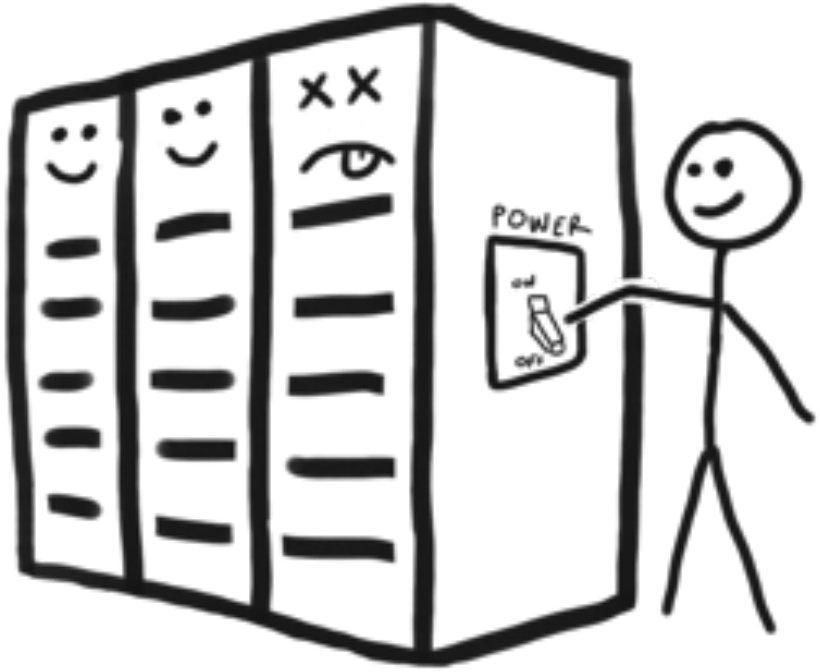
When we began our journey, we were leveraging DC/OS to manage our workloads (via marathon).



With our usage of DC/OS and OpenStack, we were needing to better understand the reactions of these systems in common failure modes.

Validate Early Concerns

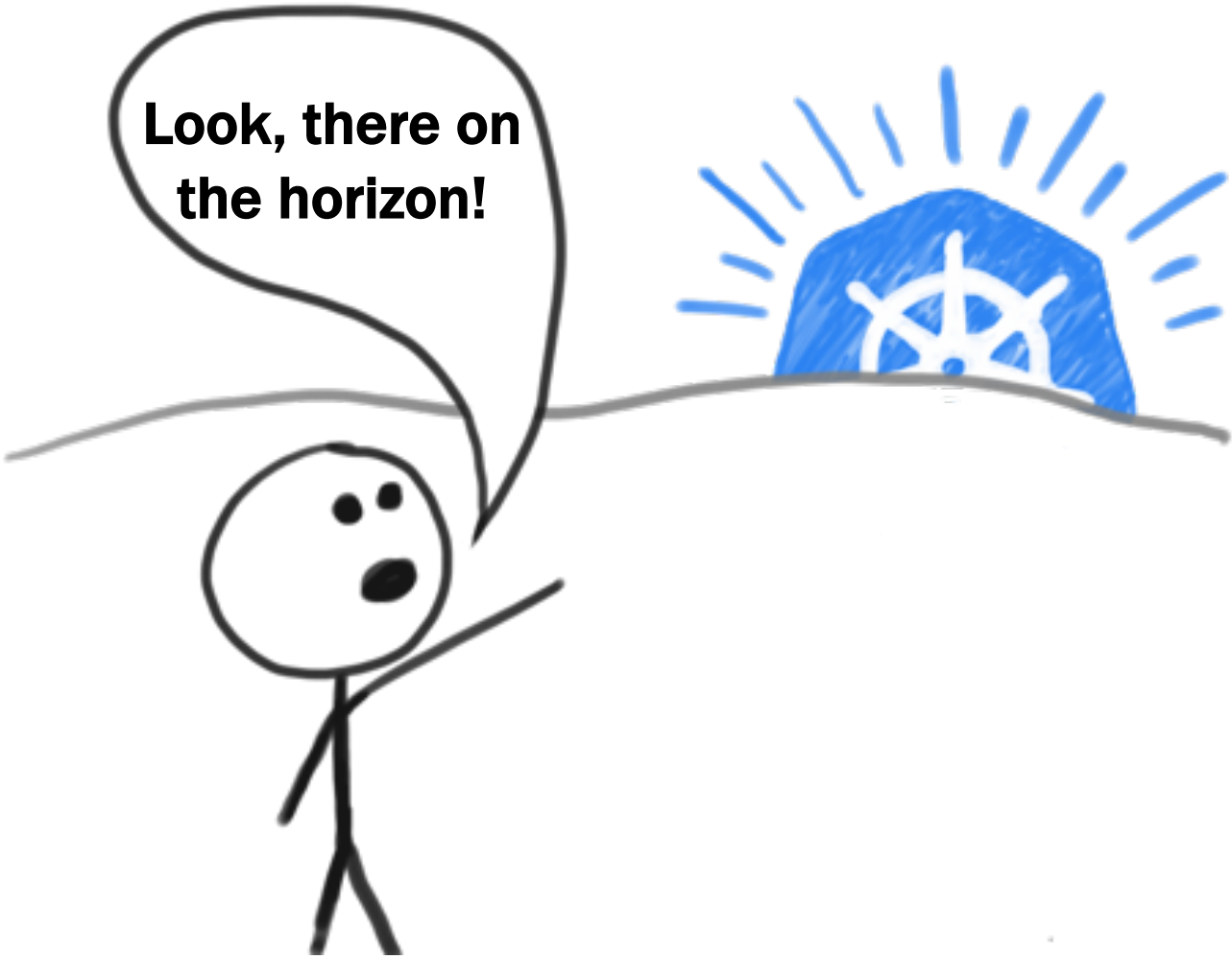
We introduced gamedays to start validating concerns of the whole system.



Simulating traffic through the system while killing VMs, powering off hypervisor, stopping availability zones, and shared infrastructure in DC/OS.

Evolving to Kubernetes

As we lived with our current system, we knew we would need to evolve it to Kubernetes.



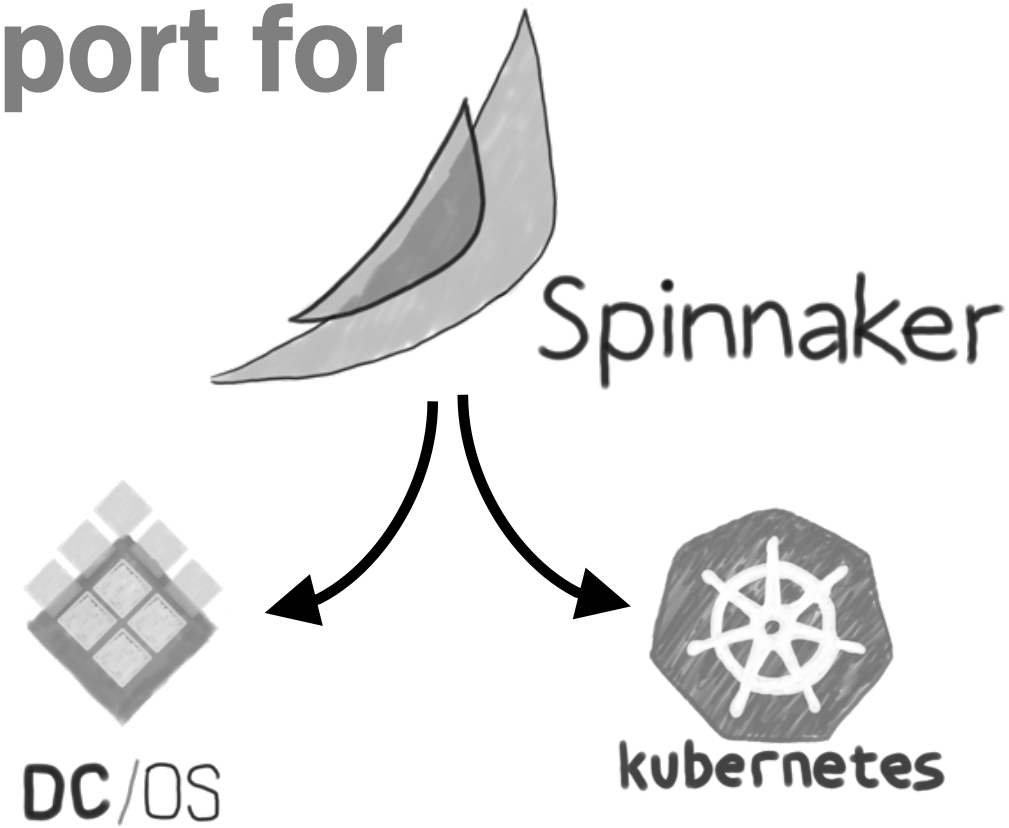
Competing Time in Growing Both Systems

As we were evolving our system, we wanted to collapse the amount of effort and time to start comparing effects of production workloads.

DC/OS

Leveraging Spinnaker

When we built our deployments for DC/OS, we added support for DC/OS to Spinnaker.



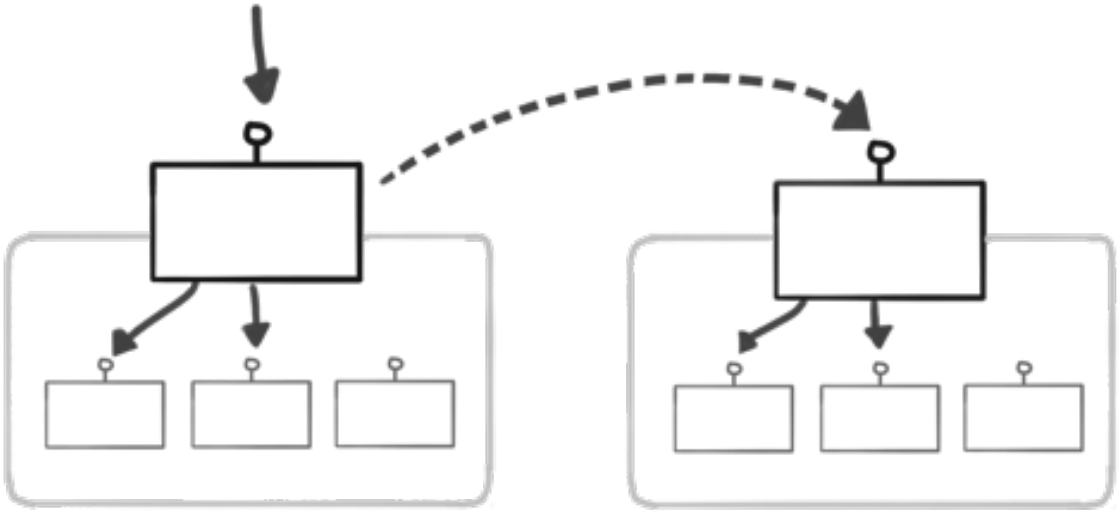
We then leveraged it to deploy to both systems as we compared the behavior in Kubernetes.

Fear of Running Experiments on Live Traffic

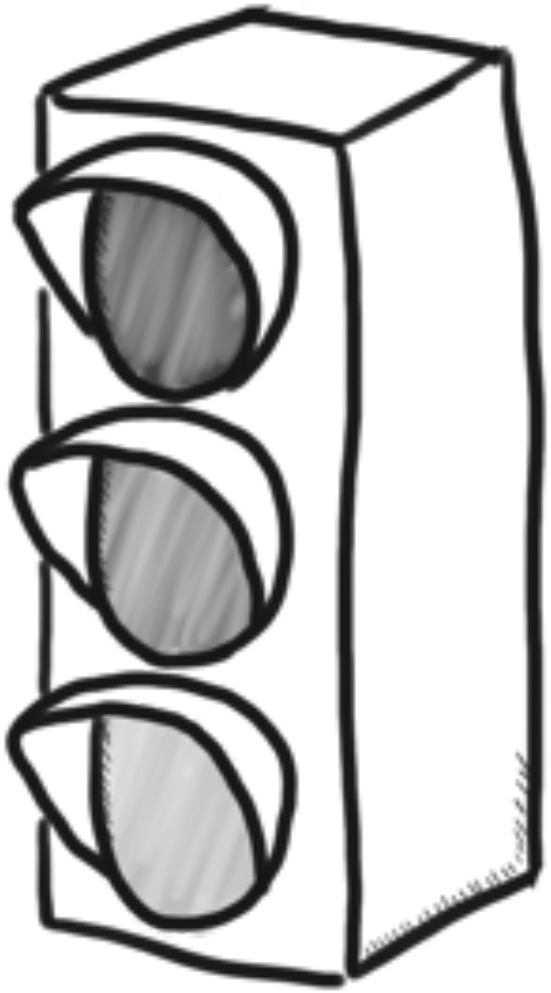
The introduction of chaos experiments on live production systems, even for a small percentage of traffic, can seem too risky.

Introduced Shadow Traffic

Rather than delaying when we could start evaluating our newer system, we could leverage a replay of production traffic.



Traffic Management Patterns



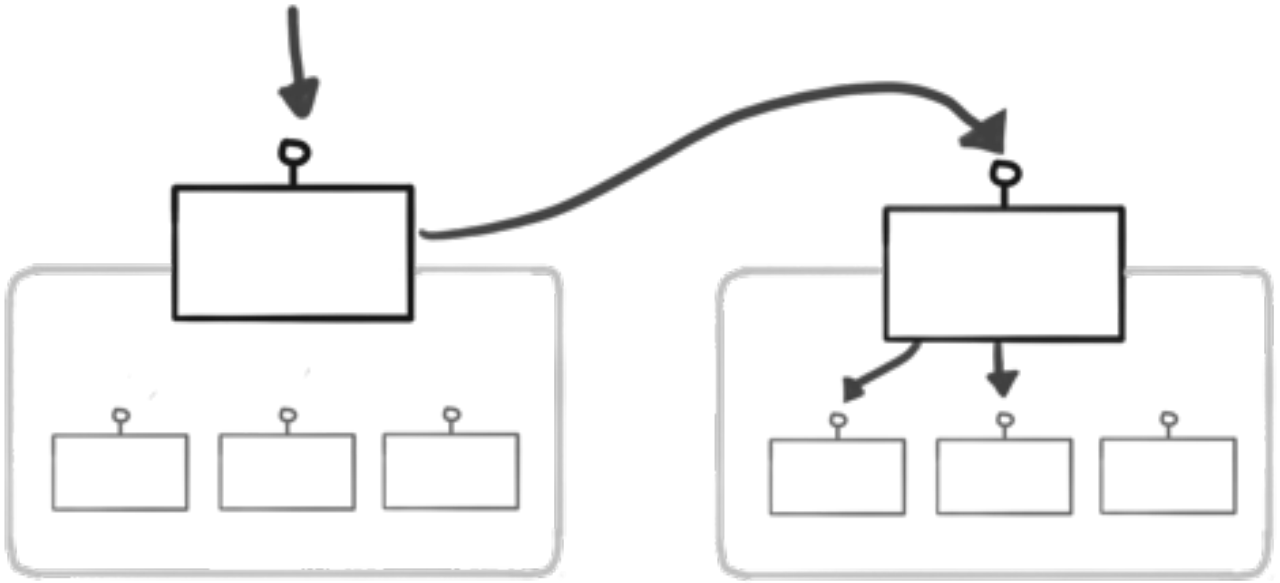
API Gateway to Facilitate Change

We evolved our systems many times by leveraging a control gate into our system.

Used as an abstraction of the backing system.

Chaining Traffic

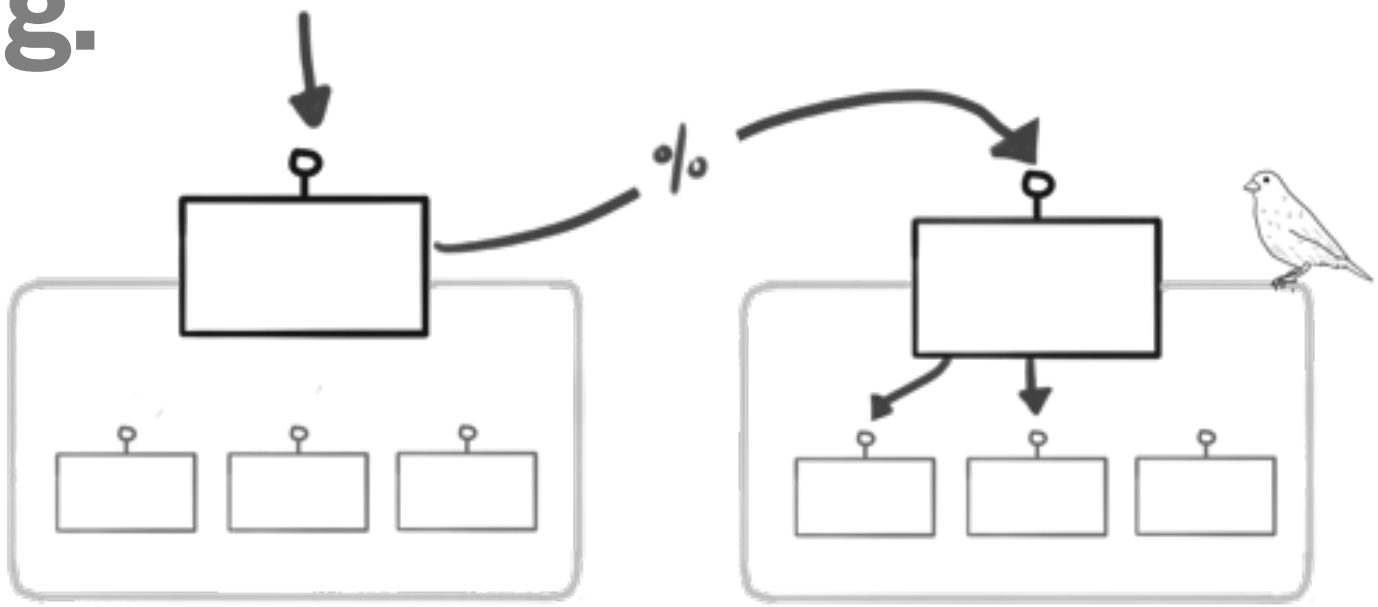
Supports an API gateway to simply call another gateway, versus the backing set of services.



Canary Traffic

Supports gradually transitioning a subset of traffic to a different target by leveraging chaining.

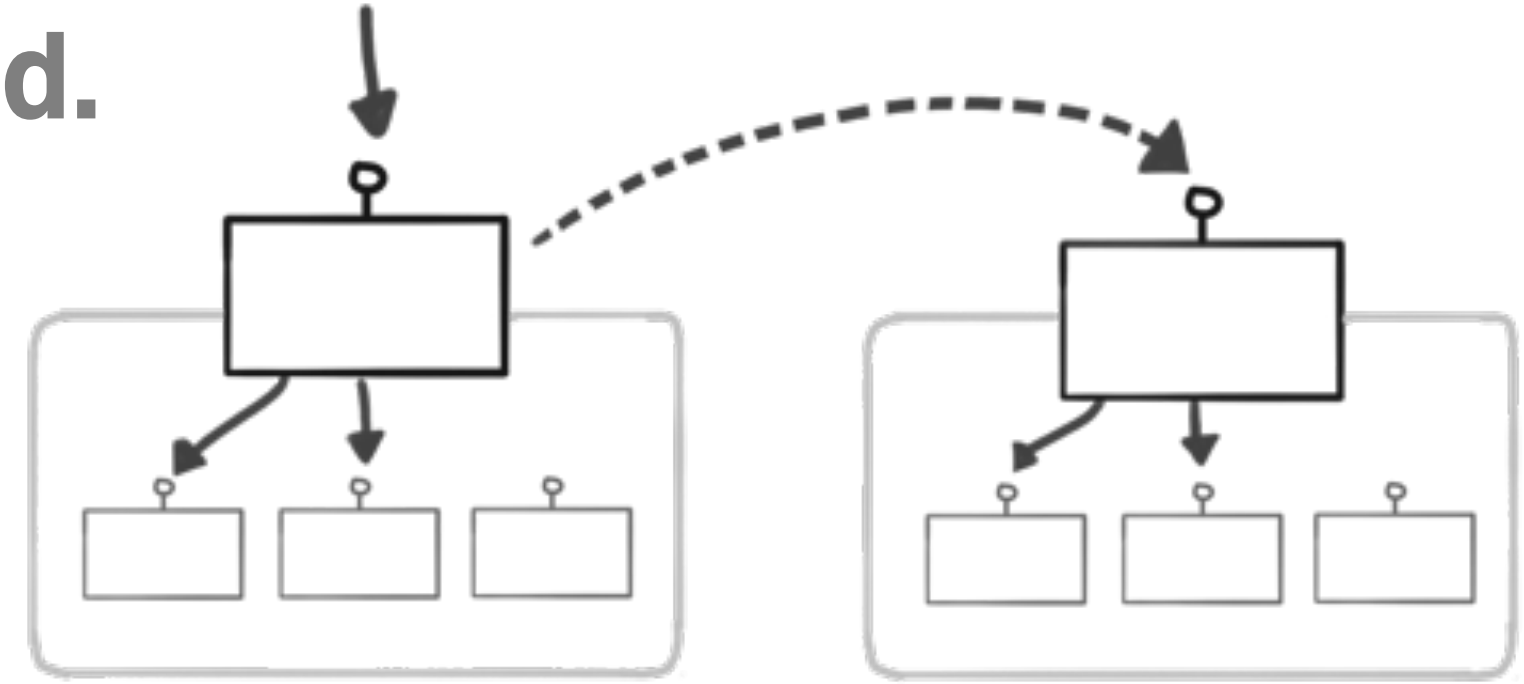
Avoid the Big Bang.



Shadowing Traffic

Replays a percentage of traffic to another backend.

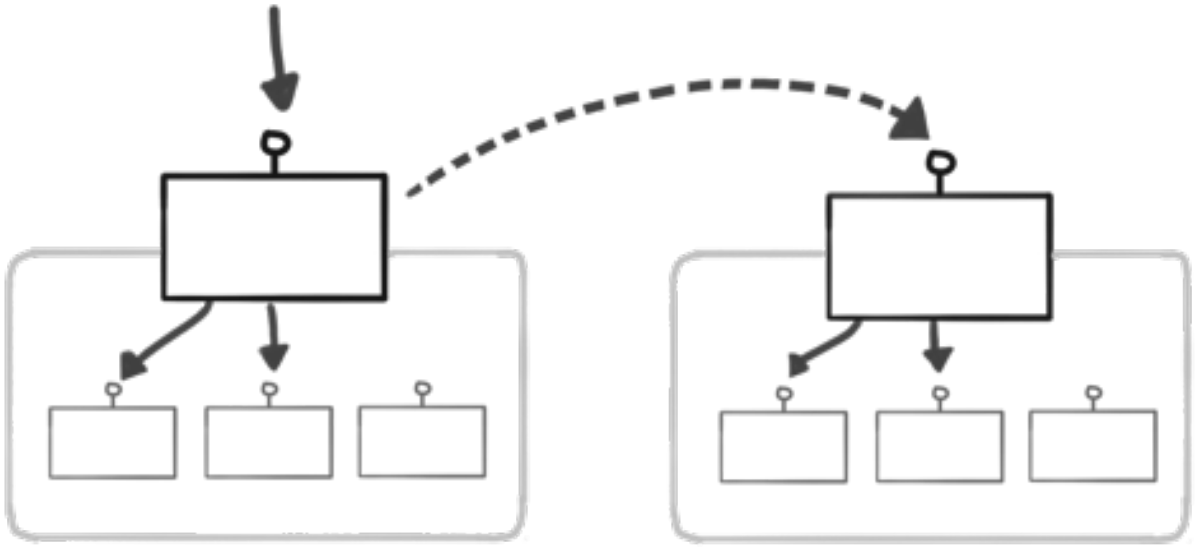
Background replay of safe requests. (read-only, HTTP GET)



Build in a bulkhead for your resource pool supporting the replay of traffic to avoid unnecessary stress on your service at bursts of traffic.

Shadow Allows Early Testing

Rather than imposing a canary early with experiments, where a small percentage of failure still introduces undesirable risk, look to leverage a shadow of traffic.



Learning from Production as we built the New

We were able to further compare and evaluate the system as we expanded the new and applied gameday exercises.

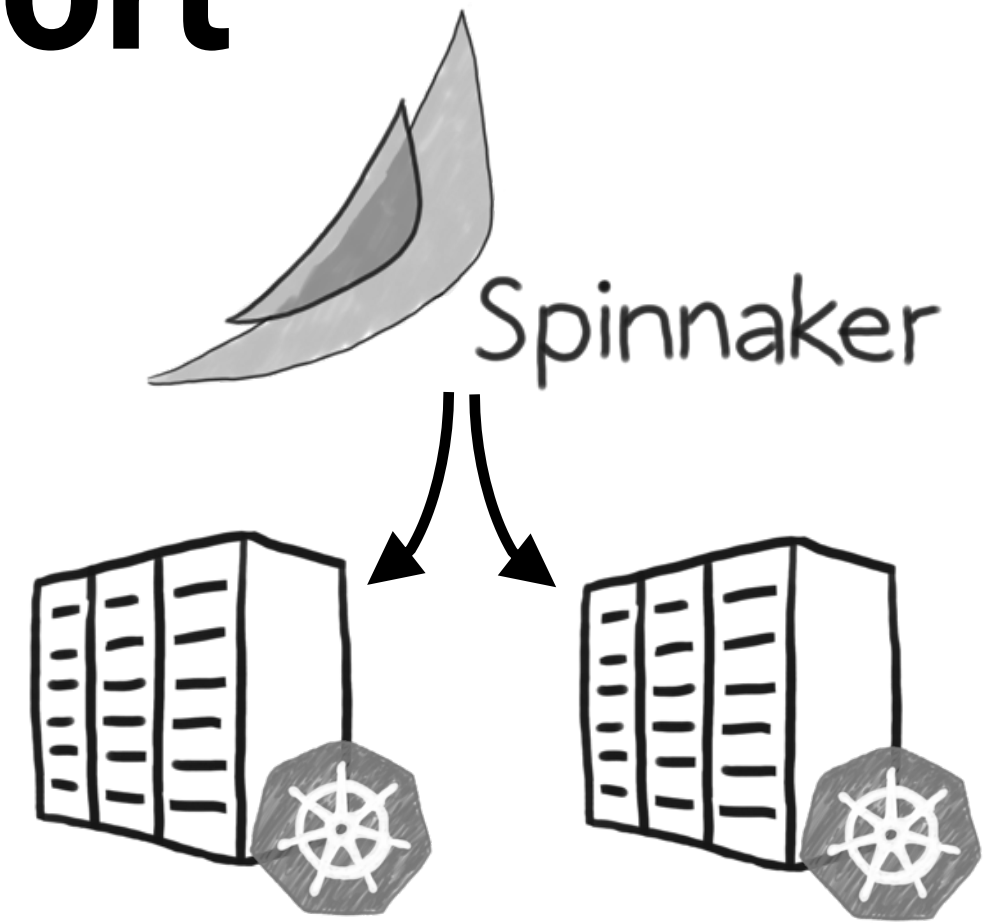
Transitioning to Kubernetes became Simple*

We identified an issue in our existing system, and through our continual assessment of the new system and practicing traffic management, it became a simple* choice.

** Simple by it being well understood, practiced, and supported by data.*

Applied in our Cross Site Kubernetes Support

Deploy services across data center sites, and we were able to leverage traffic across sites for a site incident.



Introducing Chaos Experiments

(gamedays)

Align the Introduction of Chaos with Organized Experiments

Optimize engineering focus on the introduction of chaos as planned experiments.

Minimize the opportunity for chaos to become a scape goat for mysterious issues.

Prepare for the Experiment

Describe the scenario, what is expected to occur, how it will be measured, who is needed.

Identify prerequisites that are needed to be completed

(ex. improved telemetry on connection refresh of data store)

Observability is Critical

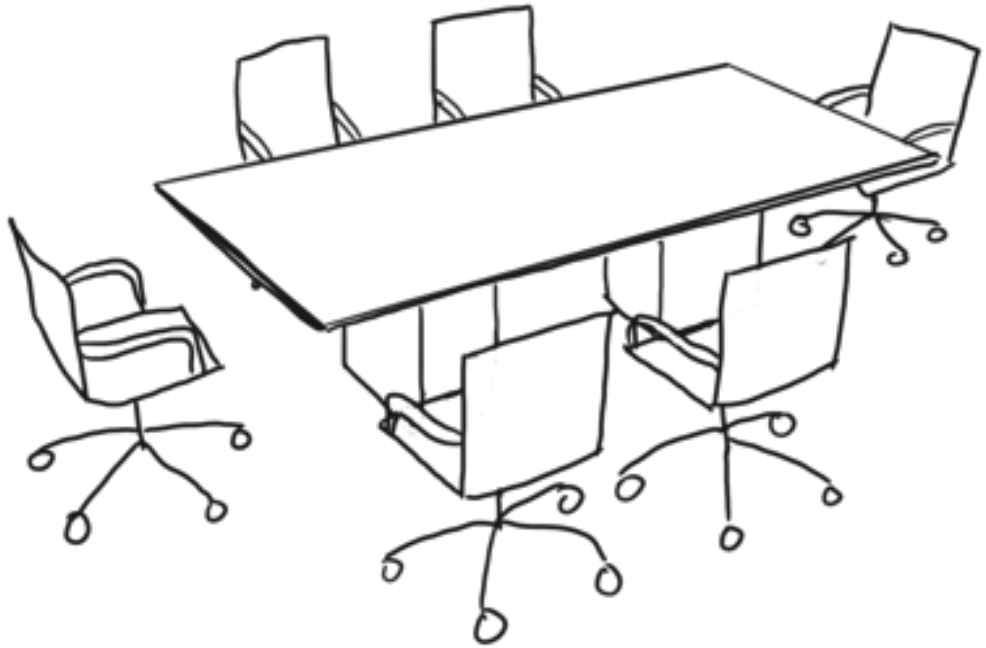
You need easy access to essential telemetry data of all the parts of the system.

You want to be able to ask different and new questions of your system without having to change the system.

When you discover a gap in visibility, focus on how to make it easy to rebuild your system with the improvement through low coordination.

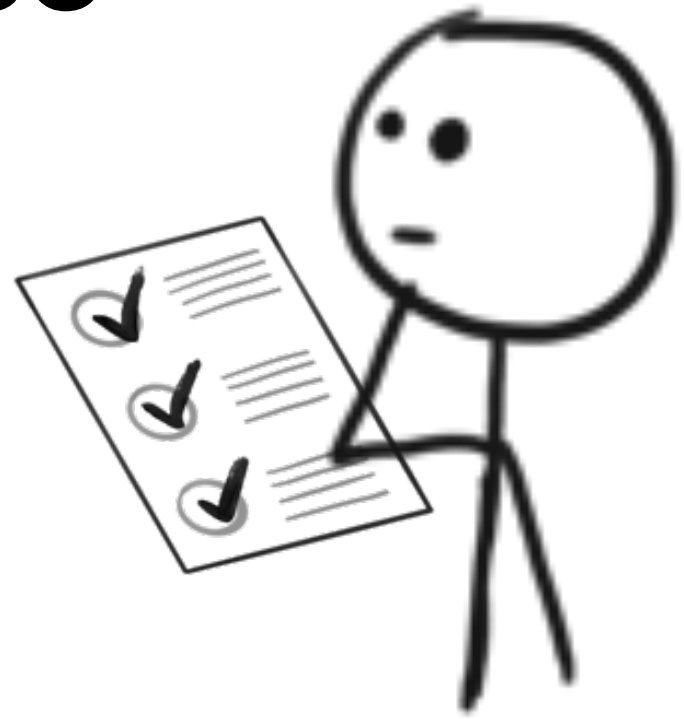
Utilize a Dedicated Space

Have a common space (physical/virtual) where everyone attends during the experiment.



You want to optimize communication when assessing the experiment. Schedule adequate amount of time for multiple iterations (ex. whole afternoon).

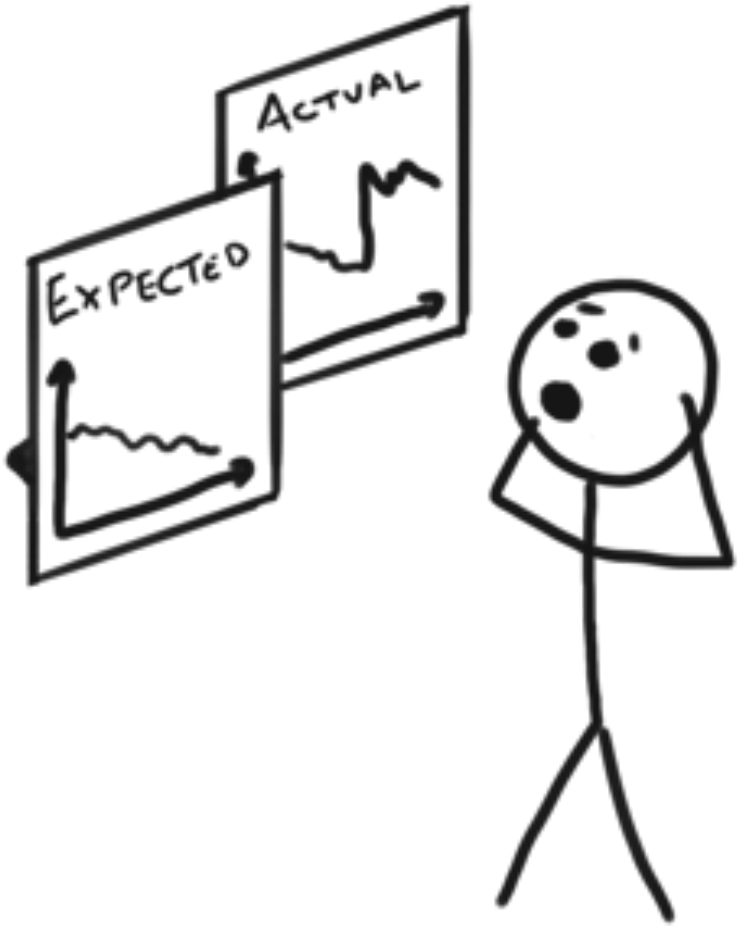
Understand and Embrace needed Compliance



Production systems will bear more compliance and controls.

Much of this is around risk, so focus on the introduction through low risk scenarios (ex. non-live systems being built).

Plan to be Surprised



We generally always learned something new about the larger system and the effects of compounding failures.

Capture what was surprising (actual results vs. what was the expected results) in an open and searchable repository.

Plan added time to digest the surprises.

Cross Functional Involvement

Helps share knowledge on how different layers of a system are viewed during the experiment.

Diverse perspectives can accelerate and improve group learning.

Prepares Your Team

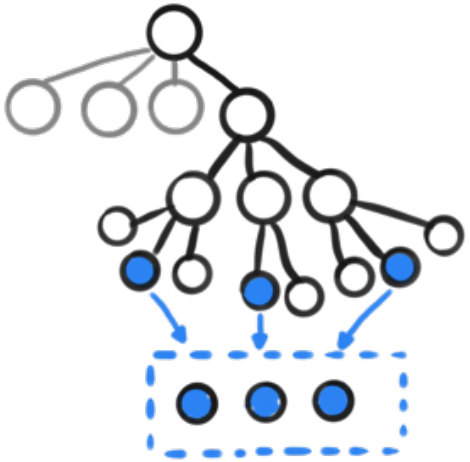
Your entire team may not be able to participate, but they should be able to learn from the findings.

Experiments help you practice how you look into the system, where signals normally arise, and identifies gaps on essential telemetry for broader insight.

Summary

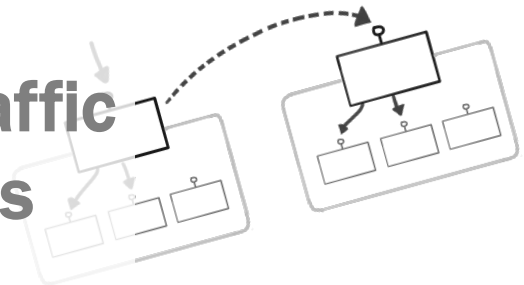
Plan for your experiments

Work to build cross functional teams to maximize learning



Identify how to make it easy to improve observability into your system

Identify how you can minimize risk through traffic management approaches



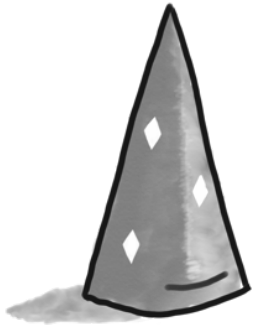
Remind your teams and leadership on measurable improvements through this practice

Technologies

https://kubernetes.io/

https://spinnaker.io/

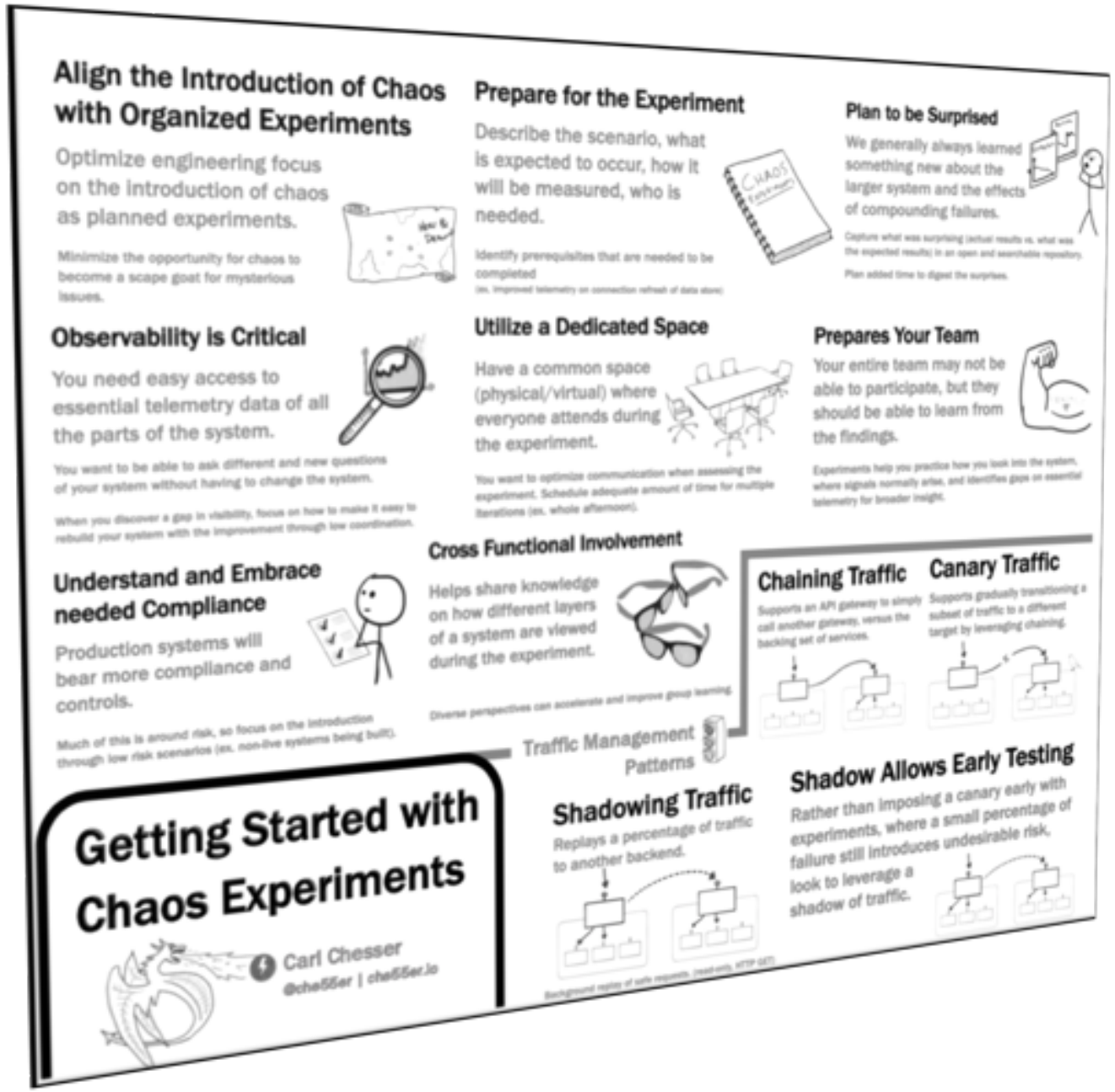
https://github.com/tsenart/vegeta



https://dropwizard.io/

https://metrics.dropwizard.io/

https://github.com/Netflix/zuul



Get your Guide!

Thank you!

Carl Chesser

@che55er | che55er.io