The Myth of the Mono-Cluster

> whoami

- pm for cdn @ google.
- open-source maintainer (github.com/gorilla)
- still unsure whether writing conf talks is a good idea.

> this talk?

- reducing the blast radius of your k8s clusters
- understanding the failure modes
- considerations as you scale k8s at an org-level

> the myths

- **one large cluster makes sense**
- isn't this what google does w/ borg?
- i have to manage more masters == bad
- centralized control
- keep scaling - it's kubernetes!
- don't want teams running their own stuff
- easier to maximise resource usage

Big 👏 Cluster 👏 Energy

> the problem

- building a single, company-wide ~~cluster~~ deployment platform *sounds* like a good idea, but it isn't.
- teams work at different paces
- **they all have different risk tolerances**

> challenges

- avoiding following the org chart
- how big is too big?
- how small is too small?
- multi-cluster orchestration

💥 Common Failure Modes 💥

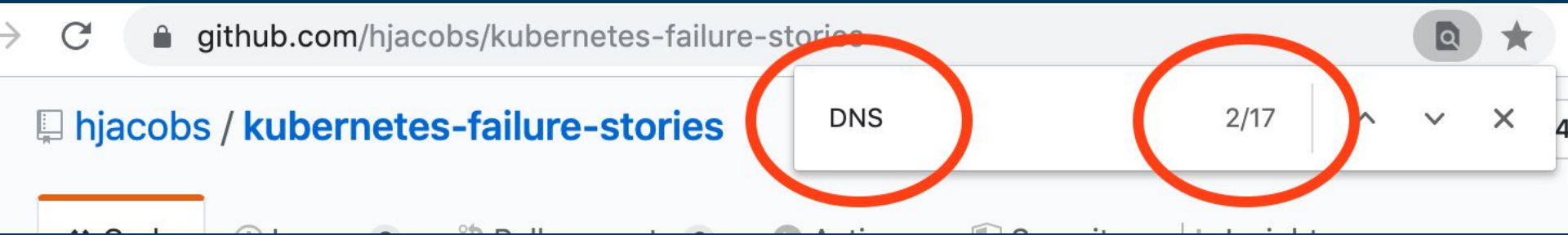
> fail

- **what components failure counts scale proportionally (or worse) with cluster size/shape?**
- what is the impact on my services?
- can we avoid it?

> dns

- can often become the bottleneck
- more-so if you’re relying on syncing to DNS infrastructure outside of your cluster (likely)
- DNS performance degradation severely impacts cluster-wide performance

> dns


github.com/hjacobs/kubernetes-failure-stories
hjacobs / kubernetes-failure-stories
DNS
2/17

> apiserver & etcd

- your control plane
- scaling your masters vertically only gets you so far
- can be really hard to fix problems (rollback, scale out) if the apiserver is unavailable or unresponsive

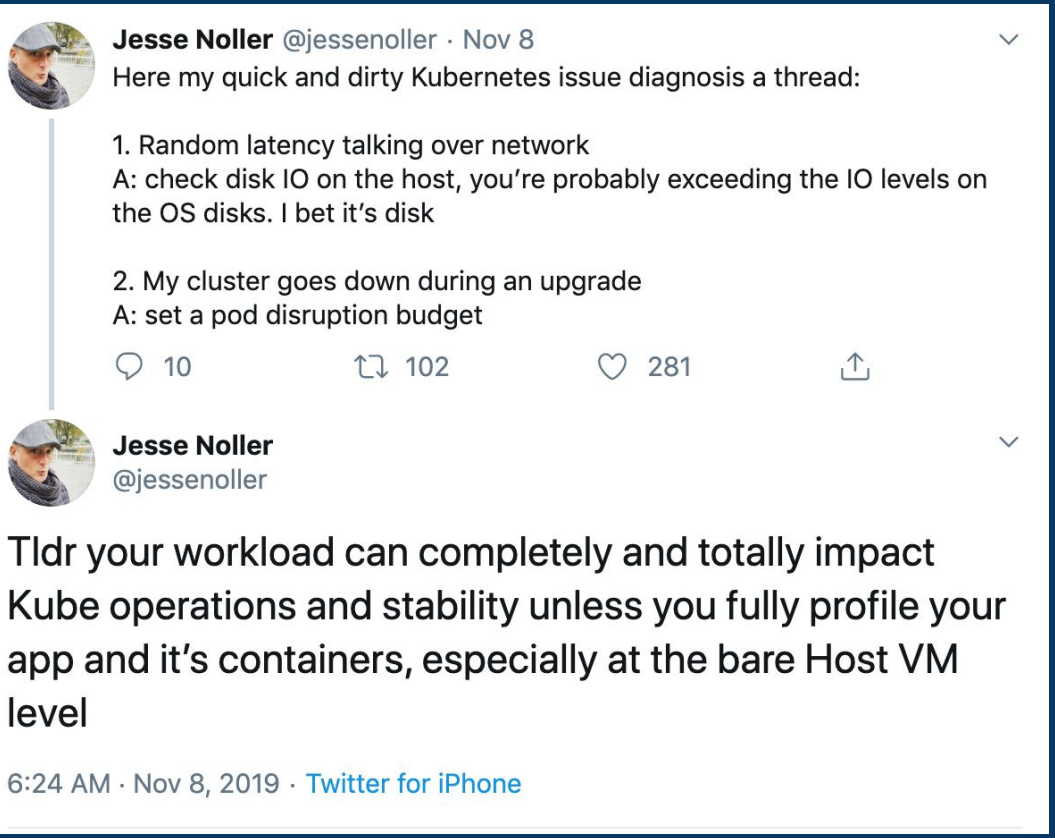
Kelsey Hightower ✓
@kelseyhightower

Control planes should be scoped to the smallest failure domain you can afford.

A single Kubernetes control plane (think multi regional) is one bad configuration or upgrade away from a regional outage.

Multiple zonal clusters, and canary rollouts, is one way to mitigate this.

7:19 AM · Jun 12, 2019 · Twitter for iPhone


Jesse Noller @jessenoller · Nov 8
Here my quick and dirty Kubernetes issue diagnosis a thread:
1. Random latency talking over network
A: check disk IO on the host, you're probably exceeding the IO levels on the OS disks. I bet it's disk
2. My cluster goes down during an upgrade
A: set a pod disruption budget
10
102
281
Jesse Noller
@jessenoller
Tldr your workload can completely and totally impact Kube operations and stability unless you fully profile your app and it's containers, especially at the bare Host VM level
6:24 AM · Nov 8, 2019 · Twitter for iPhone

Multi-tenancy has multiple dimensions

Josh Rosso
@joshrosso

What was your team's biggest "misstep" when architecting, deploying, and/or operating @kubernetes for the first time?

9:05 AM · Jun 22, 2019 · Twitter Web Client

19 Retweets **45** Likes

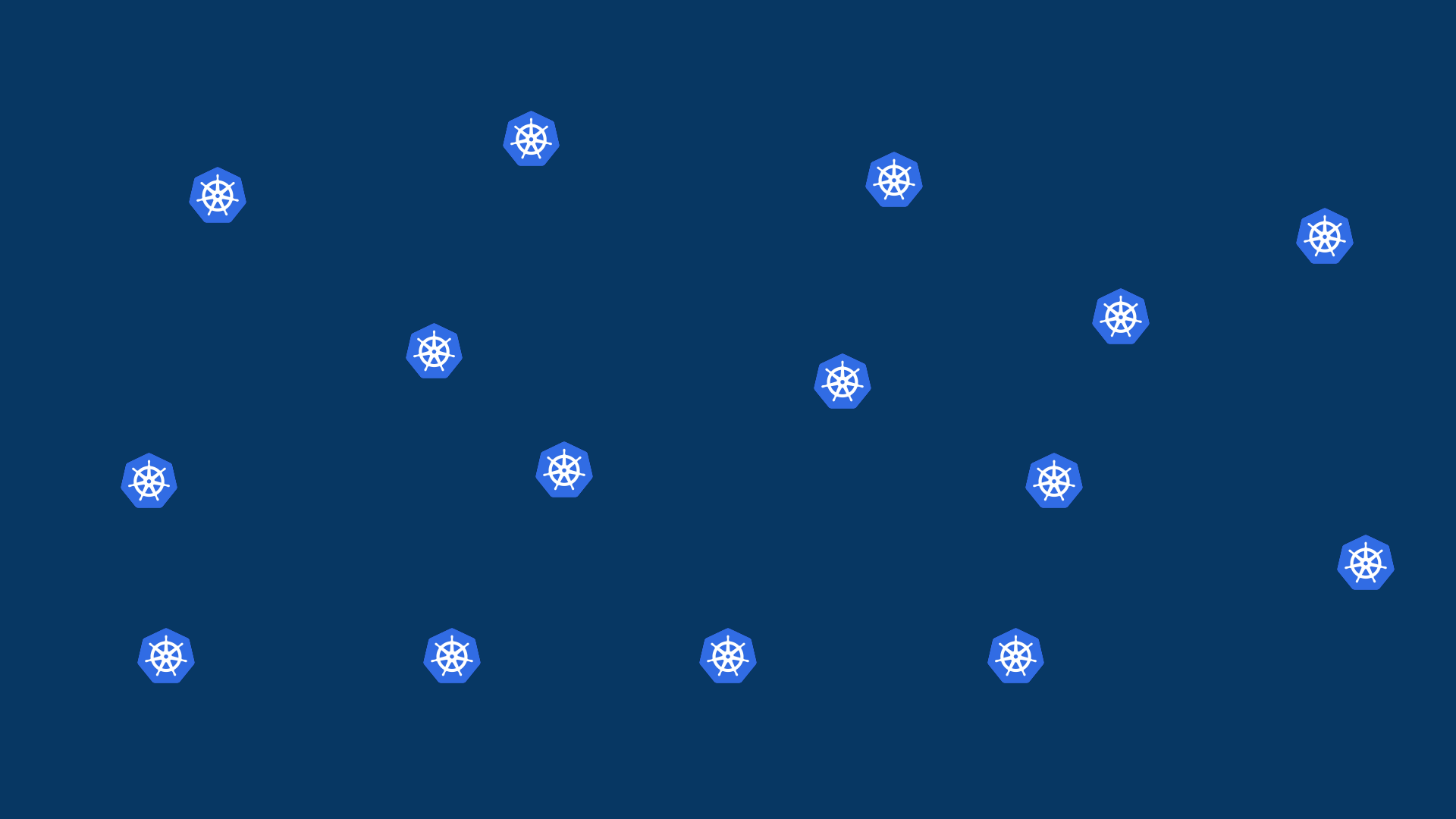
Josh Rosso @joshrosso · Jun 22
Replying to @joshrosso
Mine was trying to make clusters too "general-purpose". IMO, we wasted too much time making a single cluster satisfy all workload requirements. And we missed out on the opportunity to gain k8s operational knowledge early by running smaller clusters and learning how they worked.

1 4 31

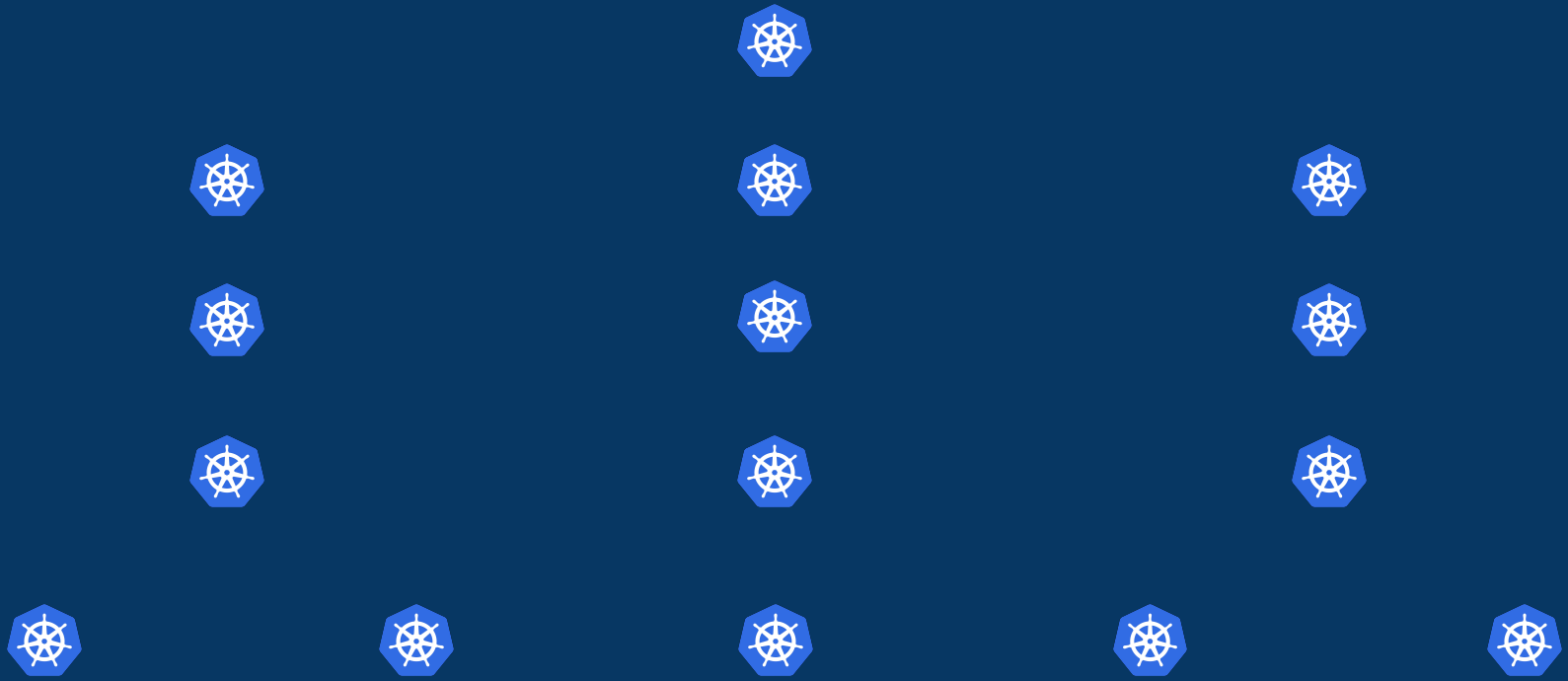
> so?

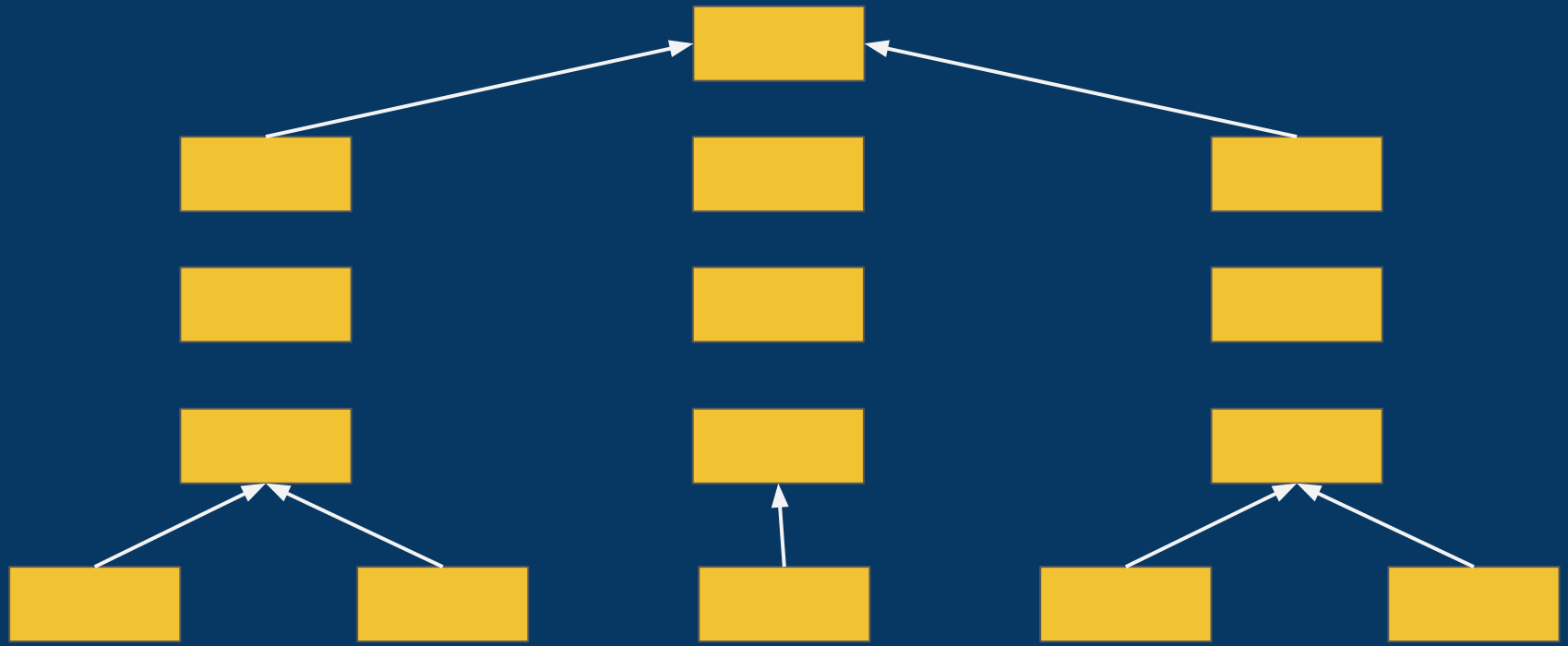
- so we should avoid a single prod cluster
- that makes sense!
- but what do we do next?

Knee-jerk reaction?




EVERYBODY
GETS A CLUSTER!
imgflip.com





organizations which design systems ... are constrained to produce designs which are ***copies of the communication structures of these organizations.***

— M. Conway

Conway's Law

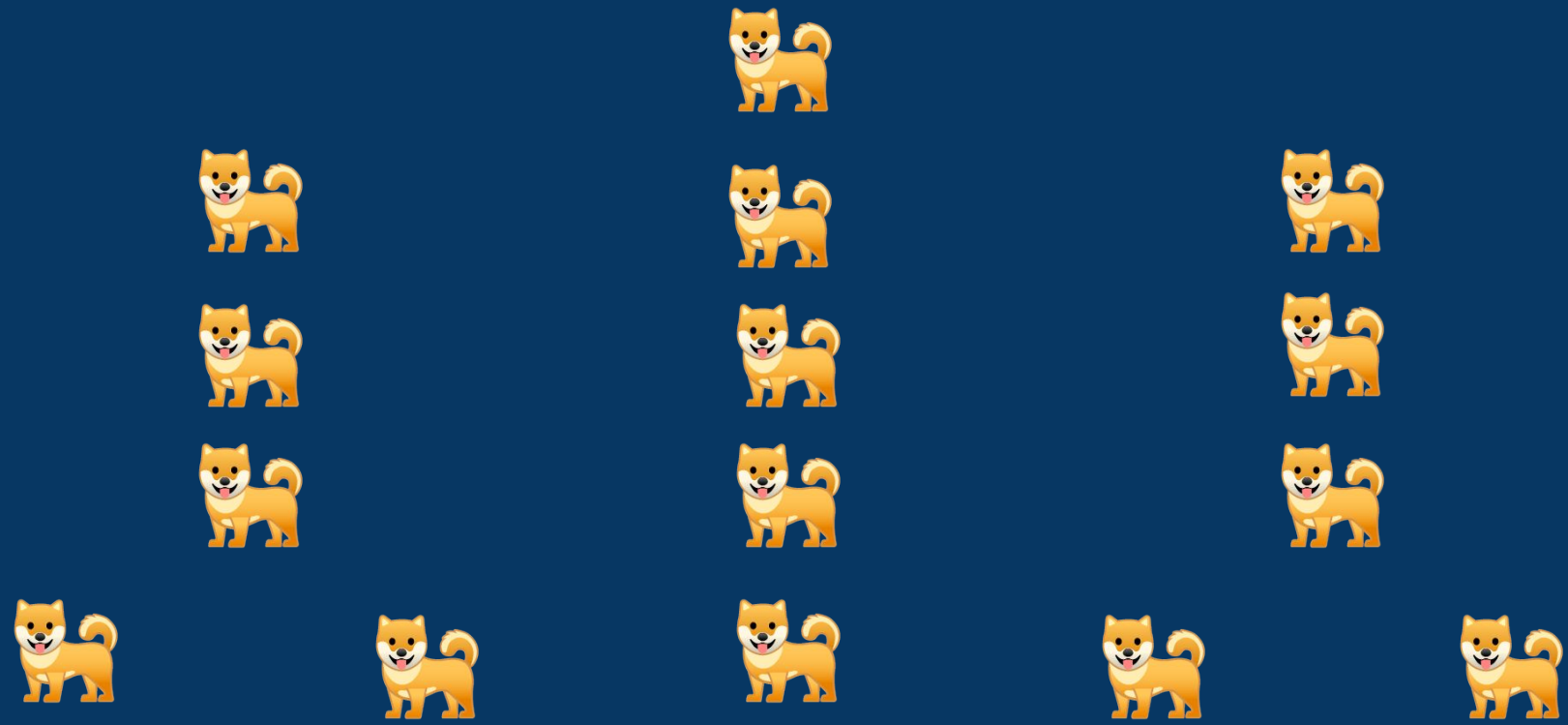
A cluster per team

> why many clusters?

- teams work at a different pace, and have different risk tolerances
- being able to manage that across a handful of clusters gives you flexibility.
- avoid moving at the pace of your most risk-averse team

> specifics

- **instead of one pet, you have lots of pets**
- (in the real world this is great, but in k8s land, not so much)
- your dev teams are now platform teams



But we also know multi-cluster is **hard**.

> hard things

- version drift (k8s, components)
- security policy enforcement
- networking
- **security patches (CVEs!)**
- deployment strategies (ci/cd)
- sre & ops vs dev teams (ownership)
- troubleshooting

OK, where do we go from here?

> no pets allowed

- use the Cluster API or your Cloud provider's toolkits to define your base cluster config
- use CI and tools like OPA/Gatekeeper to enforce consistent policies across teams & audit drift
- take a multi-cluster ingress approach to de-risk your customer-facing services

So, what is the right number of clusters?

> criteria

- map out your “risk domains”
- what services are the most likely to impact others (e.g. host level, network level)
- what is your (in real dollars) budget?

> who?

- a dedicated platform team should run your clusters
- be responsible for upgrades, security
- **avoid exposing native k8s APIs to all comers: most folks want to deliver applications and not be forced into understanding the complexities of the kubernetes scheduler**

>

Kelly Sommers
@kellabyte

I've watched org after org suffering from outages due to Kubernetes behaviours. This is not to say Kube is bad or buggy, there's just a TON of knowledge required to run Kube effectively and avoid accidental outages. It's trendy, but I think it's better off with mature ops dept.

6:36 AM · Jun 12, 2019 · TweetDeck

> takeaways

- start small
- document your shortcuts
- **understand you'll never capture all requirements up front**
- be mindful of the blast radius
- k8s is a platform: let your platform teams run it.

> thanks

- thank you!
- see my admission control micro-framework at github.com/elithrar/admission-control
- enjoy the rest of kubecon!