Do as I say, not as I do

How Atlassian built and run our own Kubernetes clusters, and why we think you shouldn't do the same

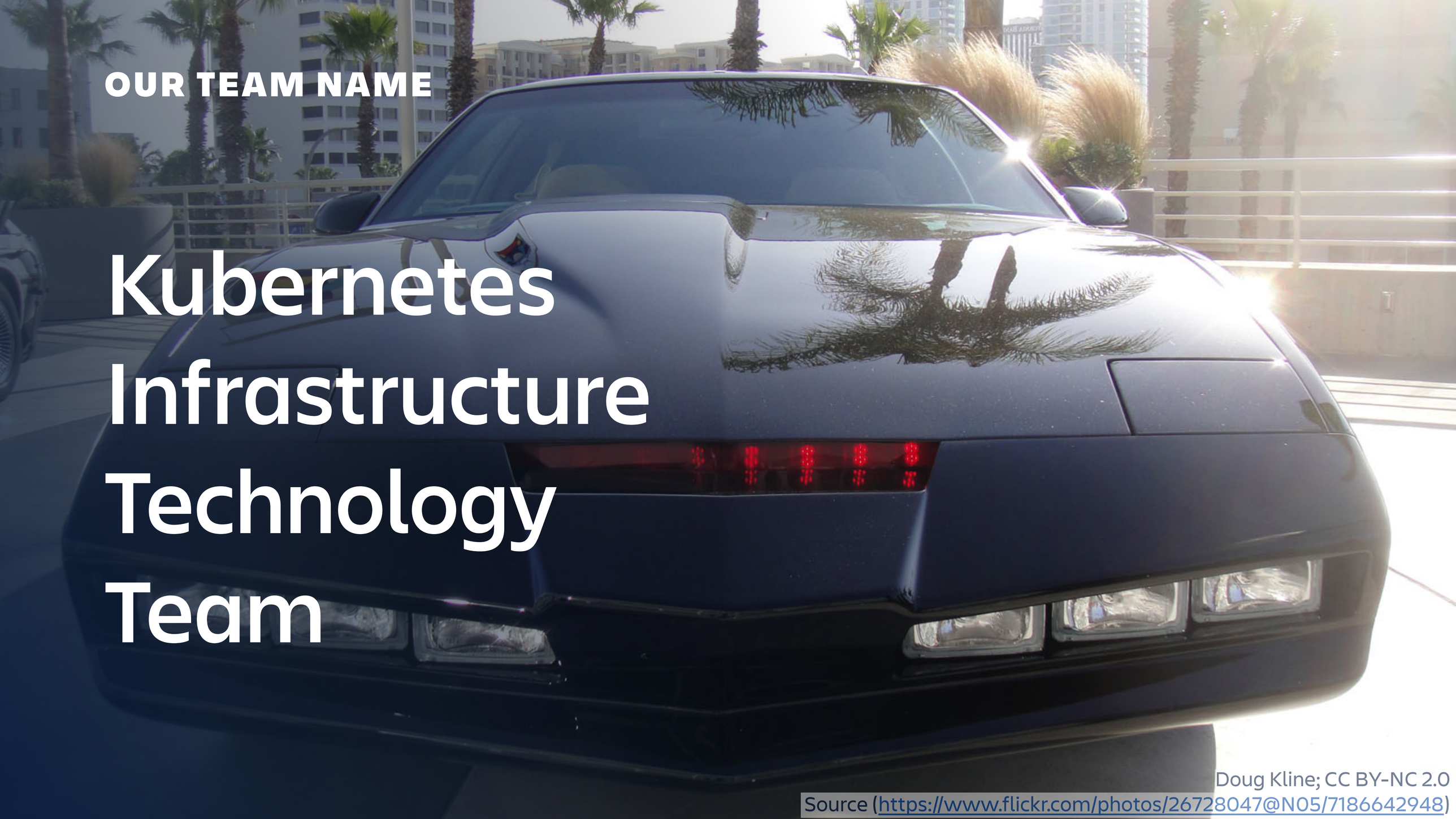
NICK YOUNG | PRINCIPAL ENGINEER | @YOUNGNICK

OUR TEAM NAME

Kubernetes Infrastructure Technology Team

OUR TEAM NAME

Kubernetes Infrastructure Technology Team



What I’d like you to walk away with

What were we trying for

What we did

How it went

Why you shouldn’t

WHAT WERE WE BUILDING?

A set of clusters that could run 95% or more of compute workloads in Atlassian

Why did we build our own?

Tools weren't ready

Three years ago, the 'stand me up a cluster' tooling was immature.

Compliance

We wanted to build our platform from day one to match our compliance regime.

Knowledge gain

We wanted to make sure we fully understood the thing we were running.

What we did

NICK'S RULE #0 OF DESIGNING SCALABLE STUFF

Design out the biggest problems you know about, so you can find new and interesting ones later.

The problems we decided to solve

Manage blast radius

We built a layer cake with strong isolation between layers, and clearly defined what a cluster meant to us.

Manage dependencies

Eventually, lots of things would be running on us - we could only depend on AWS things.

Cattle, not pets

We embraced immutable infrastructure as much as possible.

The layer cake

Goliath

All configuration that runs inside Kubernetes.
Importantly, includes RBAC, PSPs, etc.

KARR

All the compute, control plane and etcd pieces.
Stands up an apiserver endpoint, nothing else.

FLAG

Base AWS configuration, including VPCs,
subnets, VGWs, security groups, and so on.

Cattle, not pets

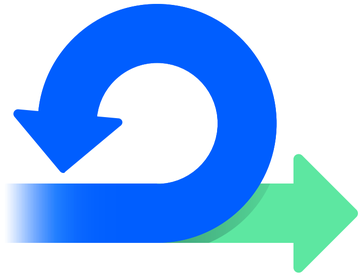
Controllers and nodes

Created in ASGs,
cycled automatically
or scaled by autoscaler

etcd servers

Like milk cows you
know the name of.

(image from https://github.com/etcd-io/etcd/blob/master/logos/etcd-glyph-color.png)



Rebuilding

We can burn a cluster
down to the FLAG and
rebuild in <30min.

Managing Dependencies

Secrets

Wherever possible, secrets are stored in private S3 buckets only accessible to the nodes.

Image storage - ECR

We can’t depend on any other container registry being up.

How we did

So how did we do?

Clusters scale pretty well

Biggest size so far is over 14000 vCPUs and 50TB of RAM, and we currently have ~20.

Mainly batch (for now)

Batch workloads are the easiest to get working on Kubernetes. We currently run about 177k Pods per day (that's 2 per sec!).

Service workloads coming

Ingress and service discovery are the keys here. Services meshes will help but are still under very active development.

Some stuff was not great

Managing etcd is hard

We had lots of problems with running etcd while we learned about its sharp edges.

Goliath layer needed work

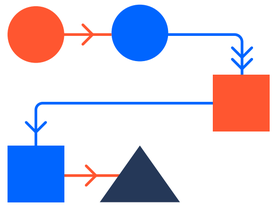
Ansible is great for early prototyping, the deployment model we used didn't scale. Mostly fixed now!

Security is a challenge

This isn't really tied to the design, but it's worth noting!

Tips

Design notes



Manage dependencies

Be careful about dependency cycles.

Cattle, not pets

Immutable infrastructure practices will make your life easier in the long run.

Design notes

Use layers

The apiserver is a
natural layer boundary.

Get involved

The kubernetes
community is great!

(image from https://github.com/kubernetes/kubernetes/blob/master/logo/logo.svg)

Do as I say, not as I do

Use boring tools

Managed K8s: pretty great

It's already at 80-90% of use cases,
and increasing.

You don't want to run etcd

Running your own etcd is akin to running
your own database 30 years ago.

Use the existing tooling

If you must build, kops, kubicorn,
and kubeadm are completely usable now.

Thanks!

NICK YOUNG | PRINCIPAL ENGINEER | @YOUNGNICK