How managed is your managed platform?



sinjo.dev



“Managed”

is a

broad term



kubernetes

Hi



sinjo.dev



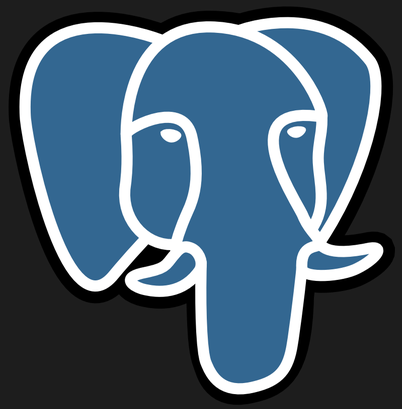




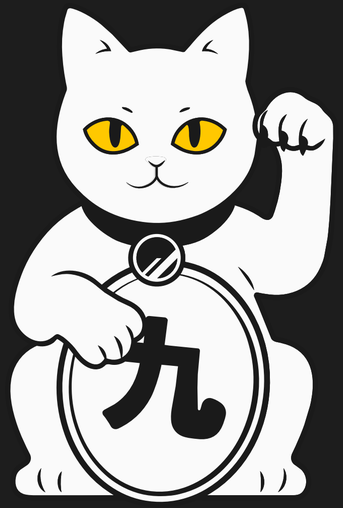
sinjo.dev

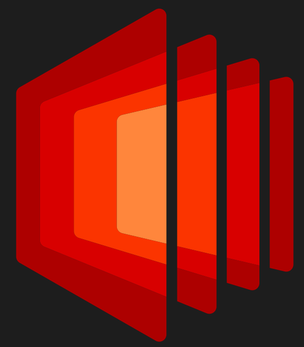


PlanetScale



PostgreSQL





Vitess



kubernetes

How managed?



Lightly **Strongly**

(made up scale, don't worry about it)

Case studies

Case studies

- **AWS:** Elastic Kubernetes Service

Case studies

- **AWS:** Elastic Kubernetes Service
- **GCP:** Google Kubernetes Engine

Case studies

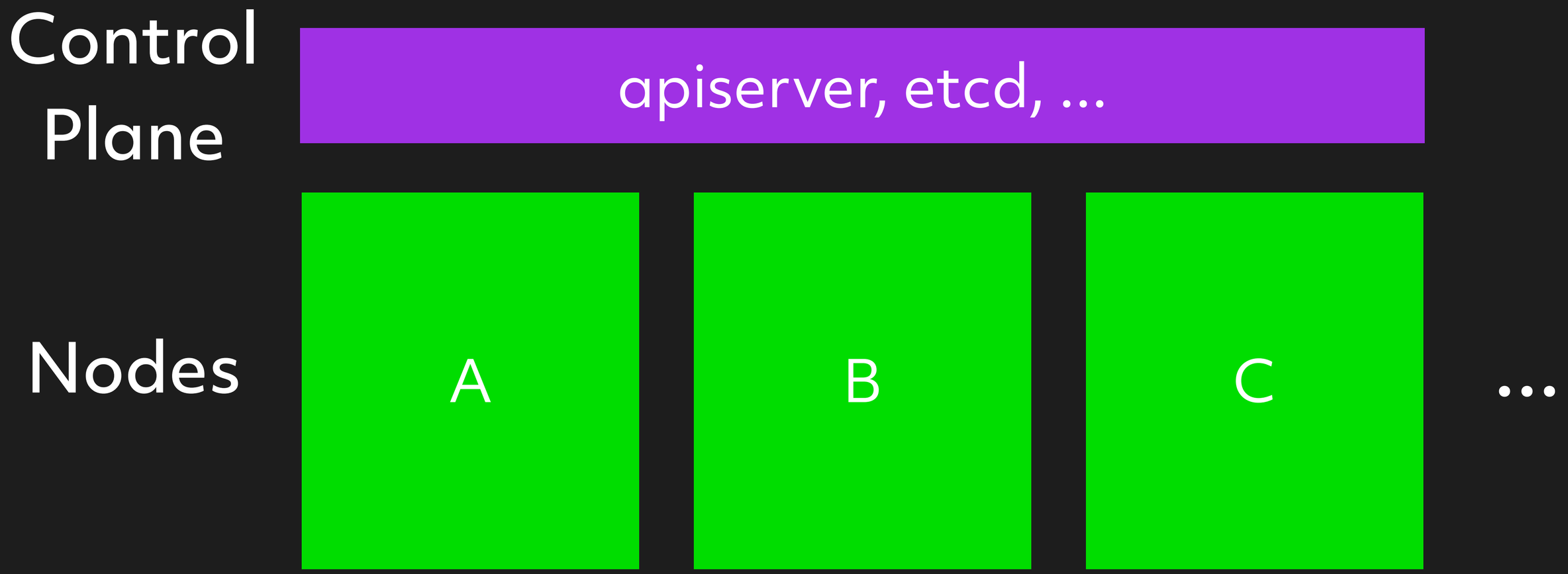
- **AWS:** Elastic Kubernetes Service
- **GCP:** Google Kubernetes Engine
- **AWS:** Fargate for EKS


amazon
Google

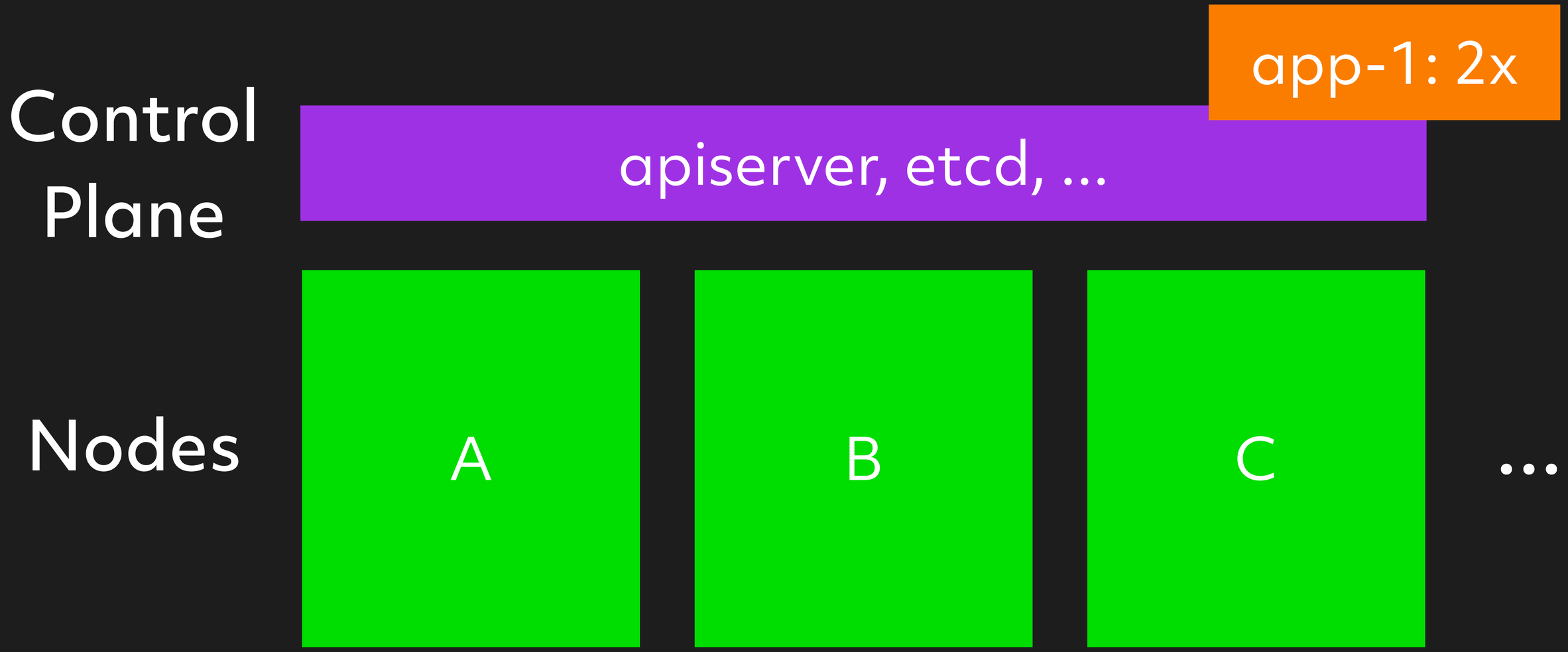
The “same” service...

...but different

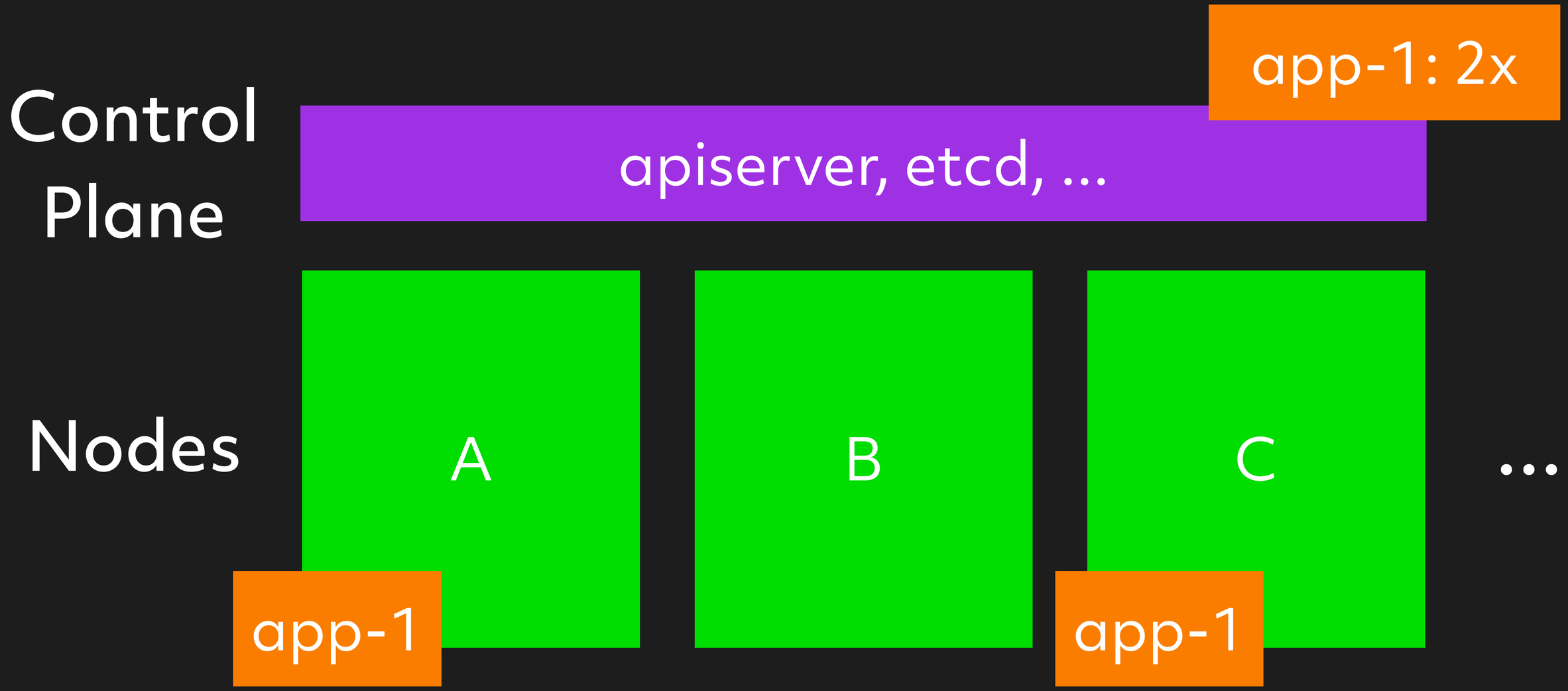
Kube in a nutshell



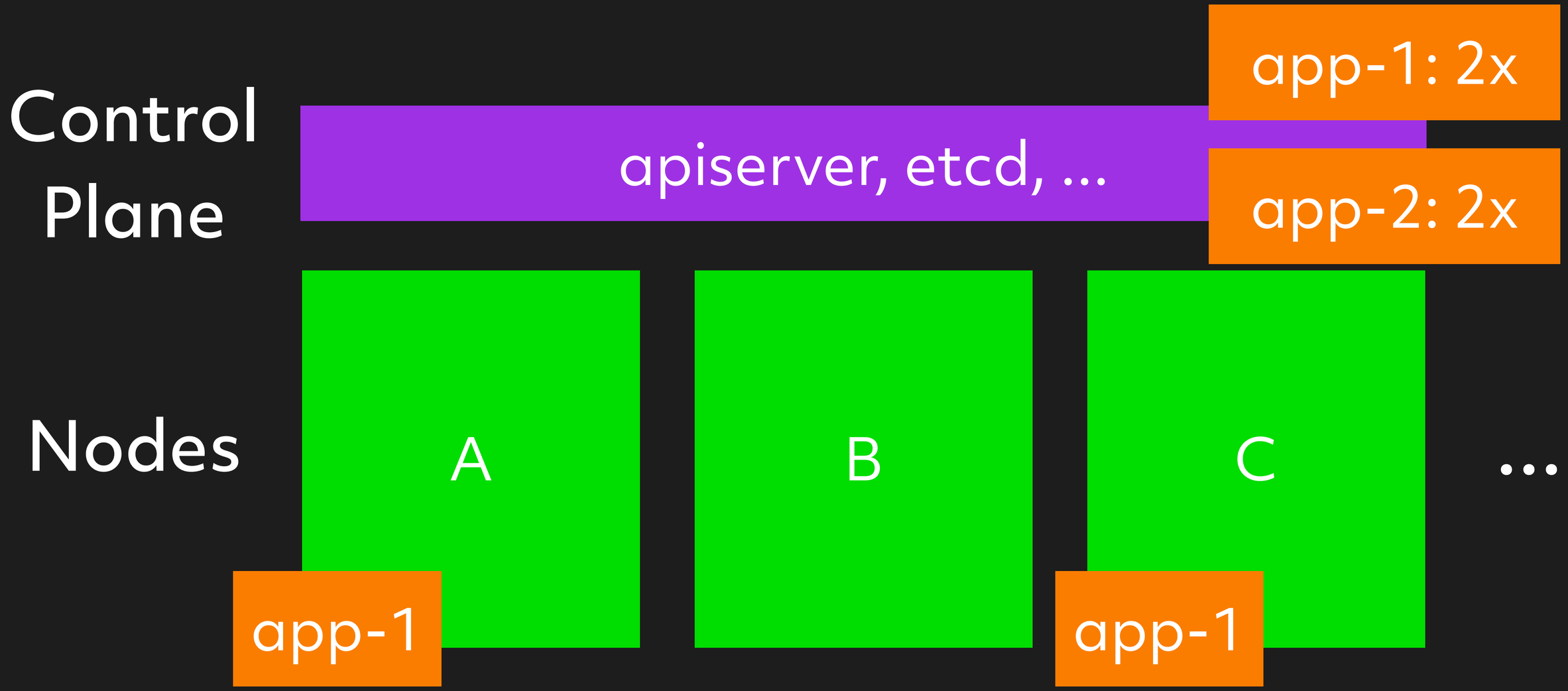
Kube in a nutshell



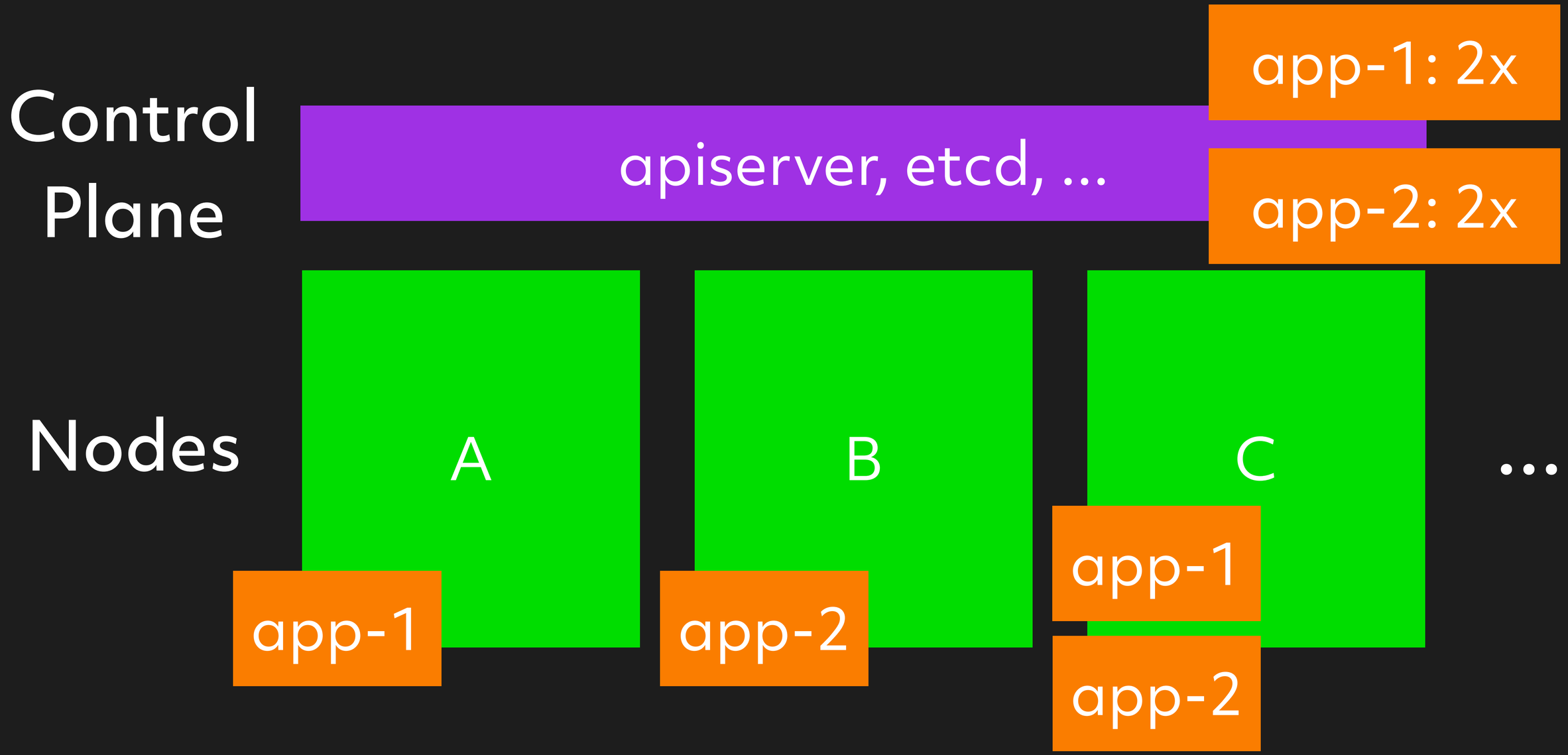
Kube in a nutshell



Kube in a nutshell



Kube in a nutshell



Points of comparison

Points of comparison

- Upgrades: API and Nodes

Points of comparison

- Upgrades: API and Nodes
- System components

Points of comparison

- Upgrades: API and Nodes
- System components
- Node OS choice

Case 1

AWS Elastic Kubernetes Service

How managed?



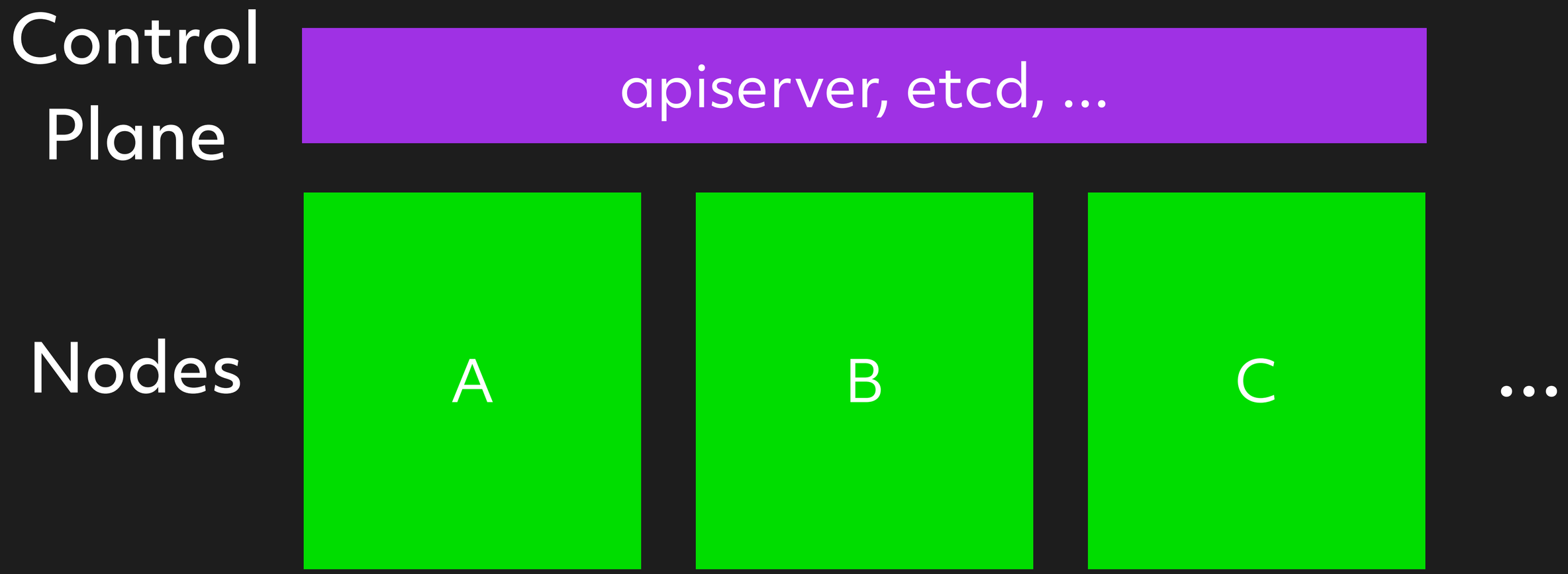


Lightly

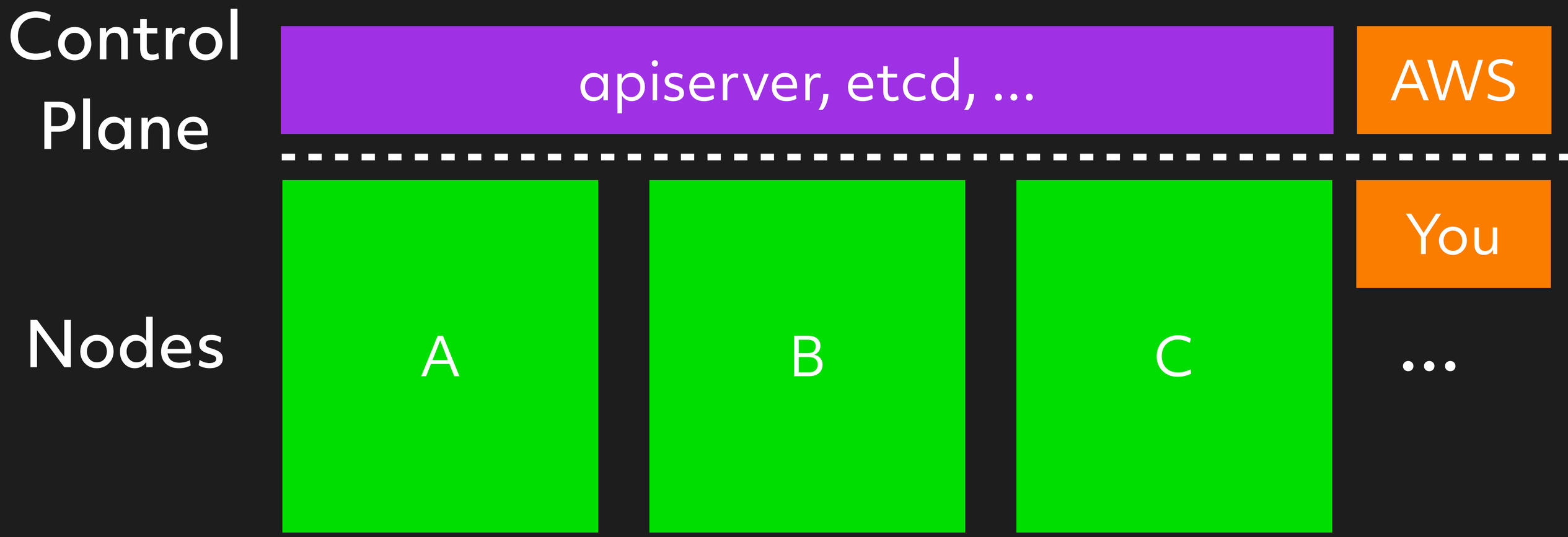
Strongly

(made up scale, don't worry about it)

EKS Responsibilities



EKS Responsibilities



Points of comparison

Applying upgrades

- Control Plane (API)
  - AWS applies
  - You choose timing (with limit)
- Nodes
  - You apply

Points of comparison

Applying upgrades

- Control Plane (API)
  - AWS applies
  - You choose timing (with limit)
- Nodes
  - You apply

System components

Basics set up for you

You can replace/ reconfigure

Points of comparison

Applying upgrades

- Control Plane (API)
  - AWS applies
  - You choose timing (with limit)
- Nodes
  - You apply

System components

Basics set up for you

You can replace/ reconfigure

Operating system choice

Whatever you like!

Documented contract to join cluster

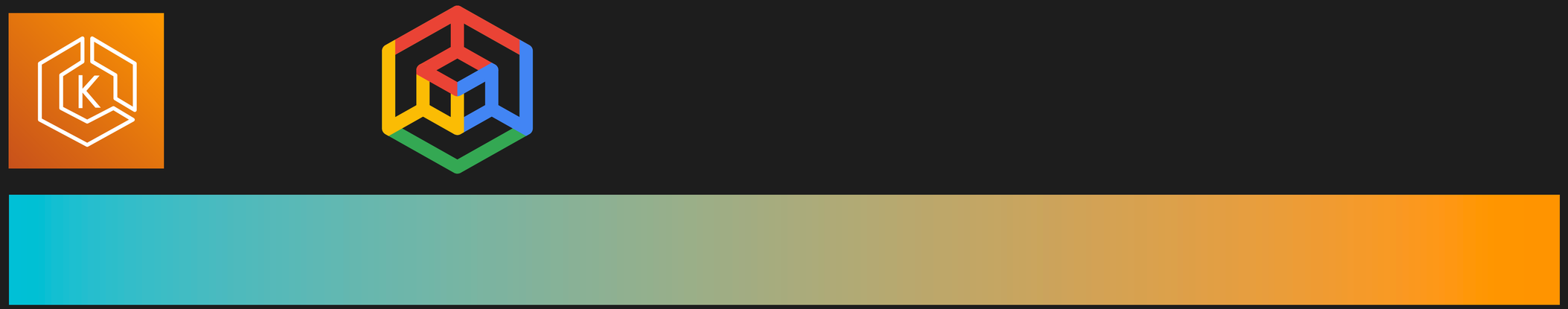
Why choose ***this*** platform?

- Well-run, off-the-shelf control plane
- Deep control of everything else

Case 2

Google Kubernetes Engine

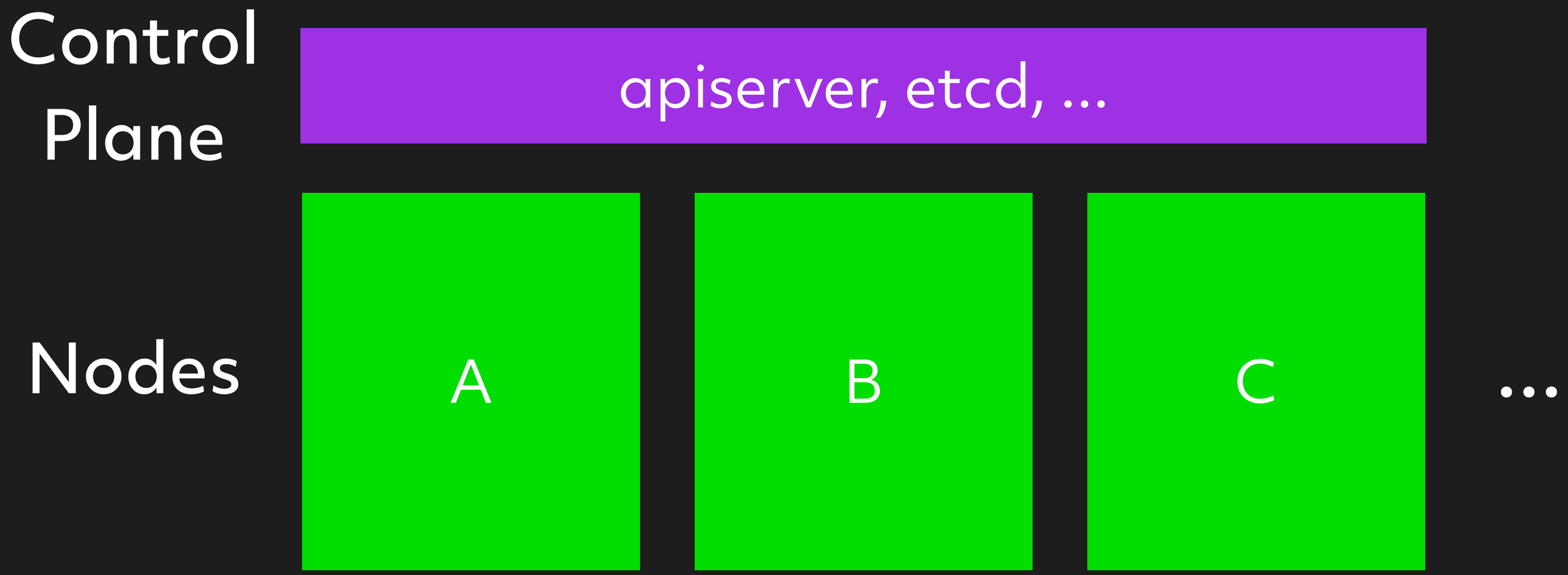
How managed?



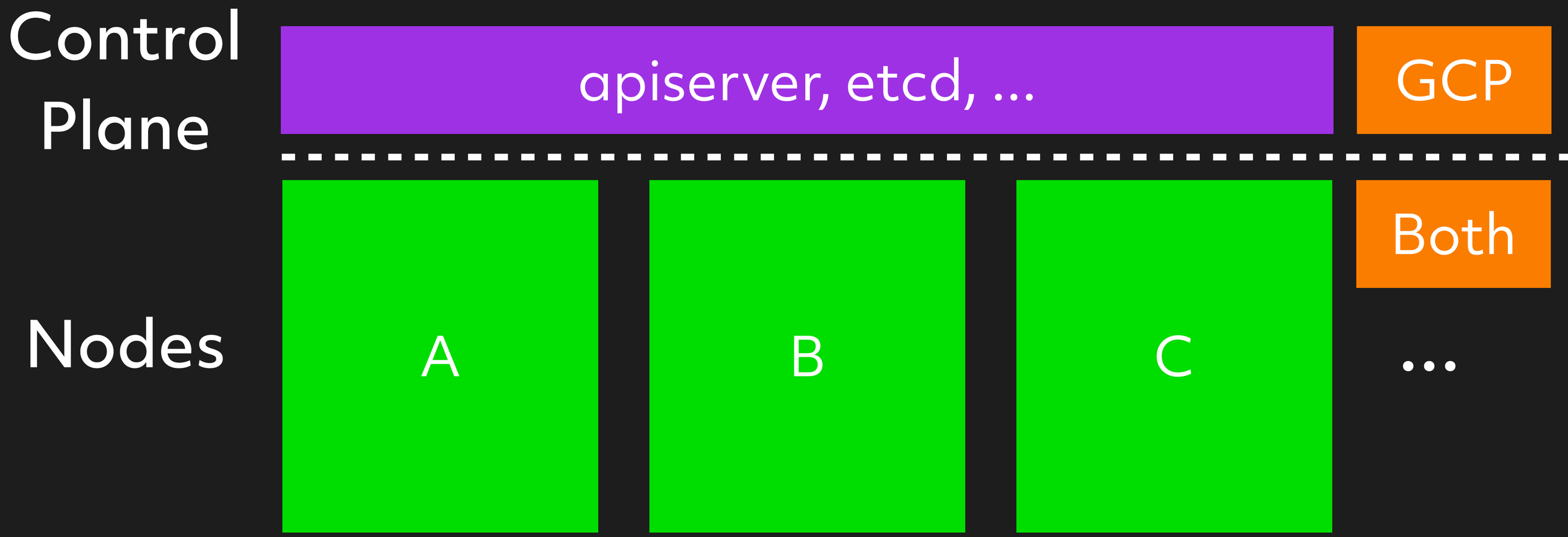
Lightly **Strongly**

(made up scale, don't worry about it)

GKE Responsibilities



GKE Responsibilities



Points of comparison

Applying upgrades

- Control Plane (API)
  - GCP applies
  - You can delay (with limit)
- Nodes
  - GCP applies
  - You can delay (with limit)

Points of comparison

Applying upgrades

- Control Plane (API)
  - GCP applies
  - You can delay (with limit)
- Nodes
  - GCP applies
  - You can delay (with limit)

System components

More set up out of the box

But less flexible on configuration

Points of comparison

Applying upgrades

- Control Plane (API)
  - GCP applies
  - You can delay (with limit)
- Nodes
  - GCP applies
  - You can delay (with limit)

System components

More set up out of the box

But less flexible on configuration

Operating system choice

Ubuntu or Container-Optimized OS

No endorsed path to your own images

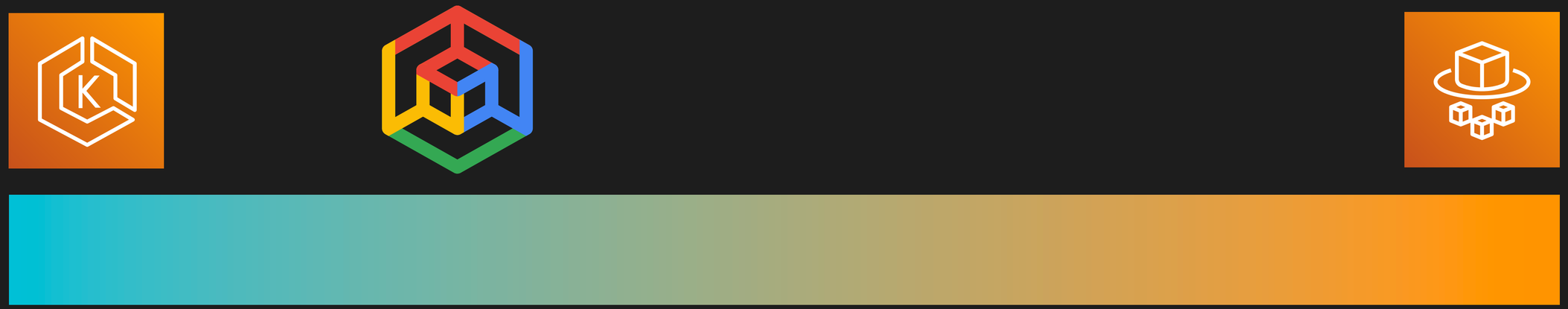
Why choose ***this*** platform?

- You want a mostly vanilla Kubernetes
- You want someone else to maintain it
- You're willing to sacrifice some flexibility

Case 3

AWS Fargate for EKS

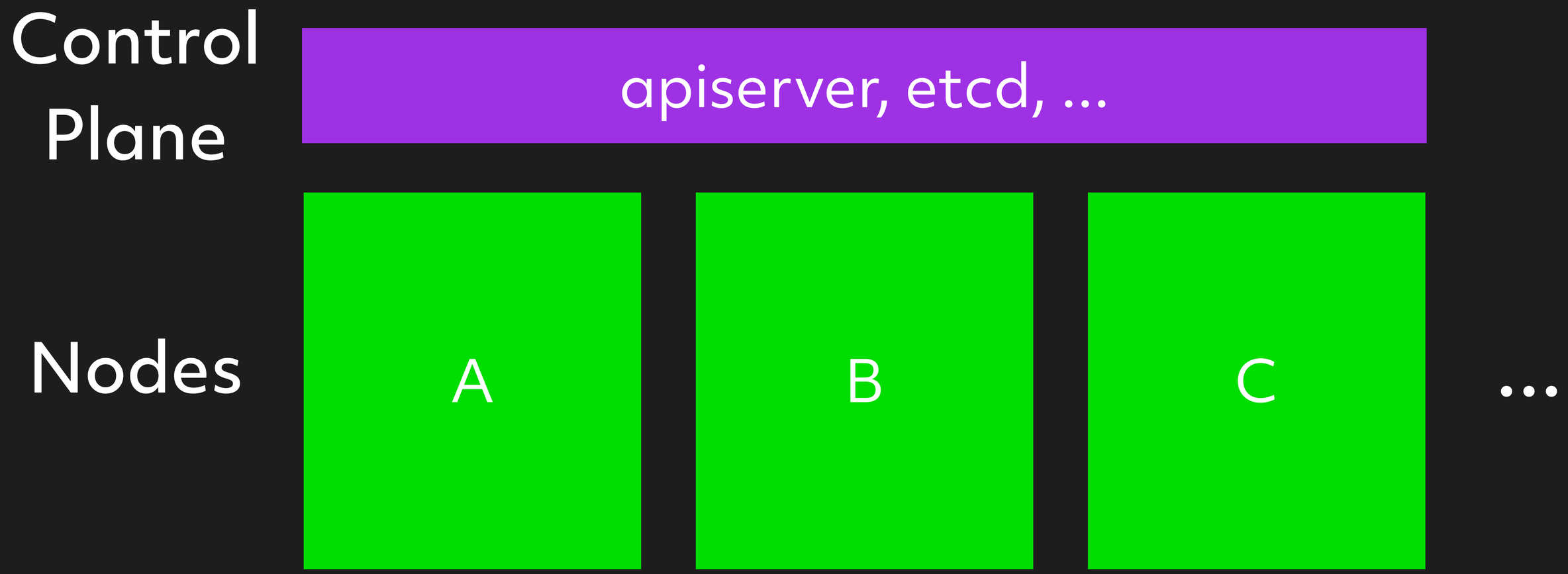
How managed?



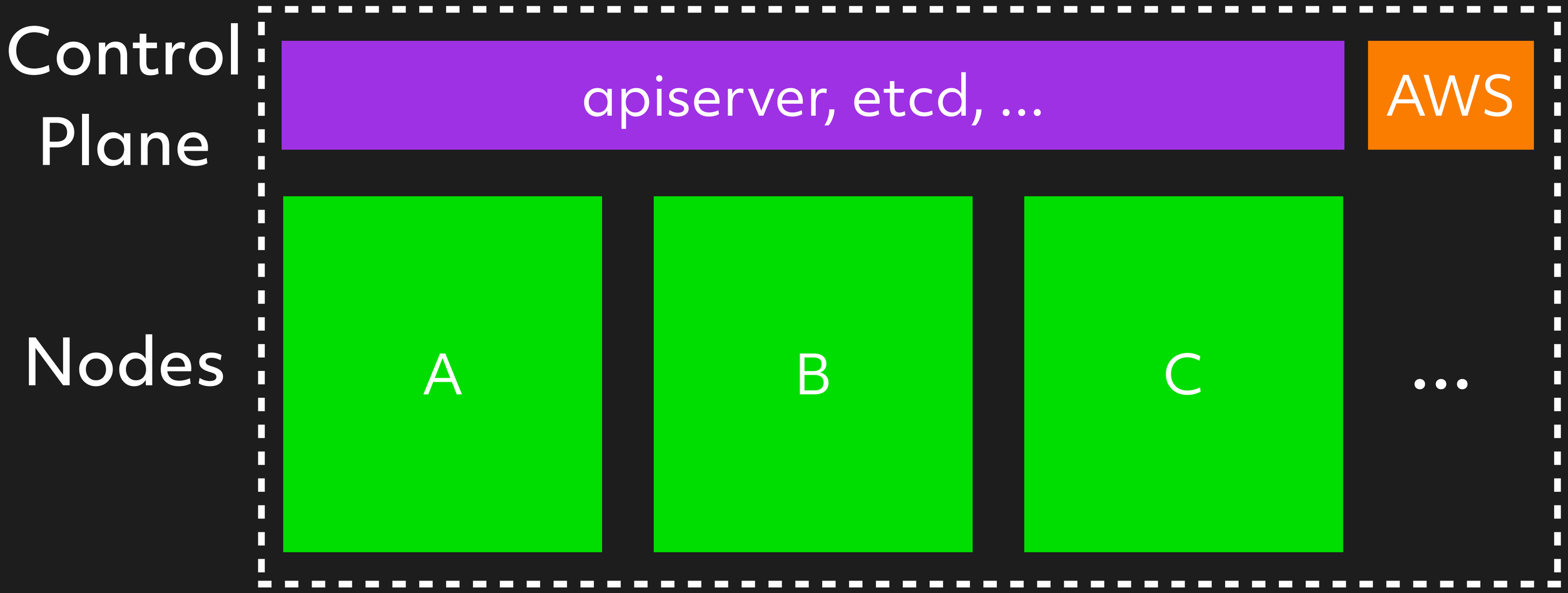
Lightly **Strongly**

(made up scale, don't worry about it)

Fargate Responsibilities



Fargate Responsibilities



Points of comparison

Applying upgrades

- Control Plane (API)
  - AWS applies
  - You choose timing (with limit)
- Nodes
  - AWS applies
  - Forced workload restarts

Points of comparison

Applying upgrades

- Control Plane (API)
  - AWS applies
  - You choose timing (with limit)
- Nodes
  - AWS applies
  - Forced workload restarts

System components

Everything below your pods is fully managed

No DaemonSets

Points of comparison

Applying upgrades

- Control Plane (API)
  - AWS applies
  - You choose timing (with limit)
- Nodes
  - AWS applies
  - Forced workload restarts

System components

Everything below your pods is fully managed

No DaemonSets

Operating system choice

Whatever AWS decides

The OS is not your concern

Why choose ***this*** platform?

- You want to schedule workloads using the Kubernetes API
- You do not want to manage the computers they run on

Before we wrap up

What drives the trade-off?



HEROKU

from Salesforce

“We’re going to build an internal ***Heroku-like*** platform”



"You can use our Terraform module"

"There's a Helm chart for that"

"We've got a Jsonnet library"

“We’re going to build an internal ***Heroku-like*** platform”

“We’re going to build an internal

platform”

It’s hard to be

all things

to

all people

Paved paths

vs

maximum flexibility

Case Studies

Case 1

AWS Elastic Kubernetes Service

Case 2

Google Kubernetes Engine

Case 3

AWS Fargate for EKS


SELF
SERVICE
TABLE
SERVICE

Choosing a platform?

Choosing a platform?

- **What do you need & what can you give up?**

Choosing a platform?

- **What do you need & what can you give up?**
- **What's your migration path out?**

Choosing a platform?

- **What do you need & what can you give up?**
- **What's your migration path out?**
- **Can you match the vendor experience?**

More flexibility means more responsibility

Building a platform?

Know who you're building for...

Know who you're building for...

...and build for them unapologetically

Thank you

✌️❤️

sinjo.dev

@PlanetScale

Image credits

- FxEmojis Cloud Emoji (colour modified to white) - CC-BY 4.0 - https://github.com/mozilla/fxemoji/blob/270af343bee346d8221f87806d2b1eee0438431a/svgs/nature/u2601-cloud.svg
- Twemoji Grinning Face with Sweat Emoji - CC-BY 4.0 - https://github.com/twitter/twemoji/blob/bad3bceeafc901ace42a3dfe0421db6388daafb9/assets/svg/1f605.svg
- Chalkboard sign - Pexels Licence - https://www.pexels.com/photo/sign-information-written-on-a-wooden-board-14027683/

Questions?

✌️❤️

sinjo.dev

@PlanetScale