Kubernetes: Kernels & Distros

KubeCon 2017, Austin
December 7, 2017

Tim Hockin <thockin@google.com> @thockin
Michael Rubin <mrubin@google.com> @matchstick

Where is Kubernetes today?

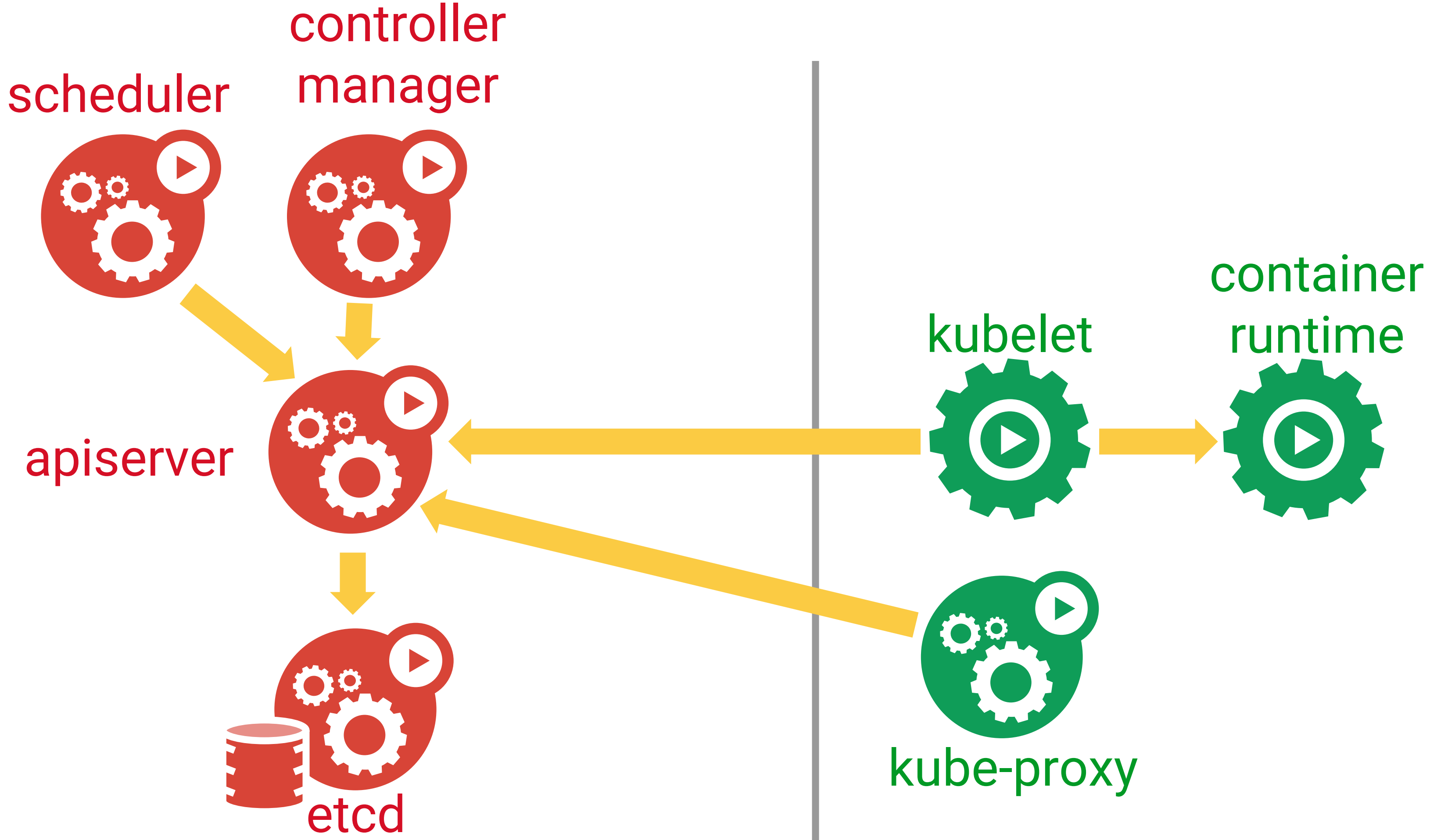
Kubernetes is ...

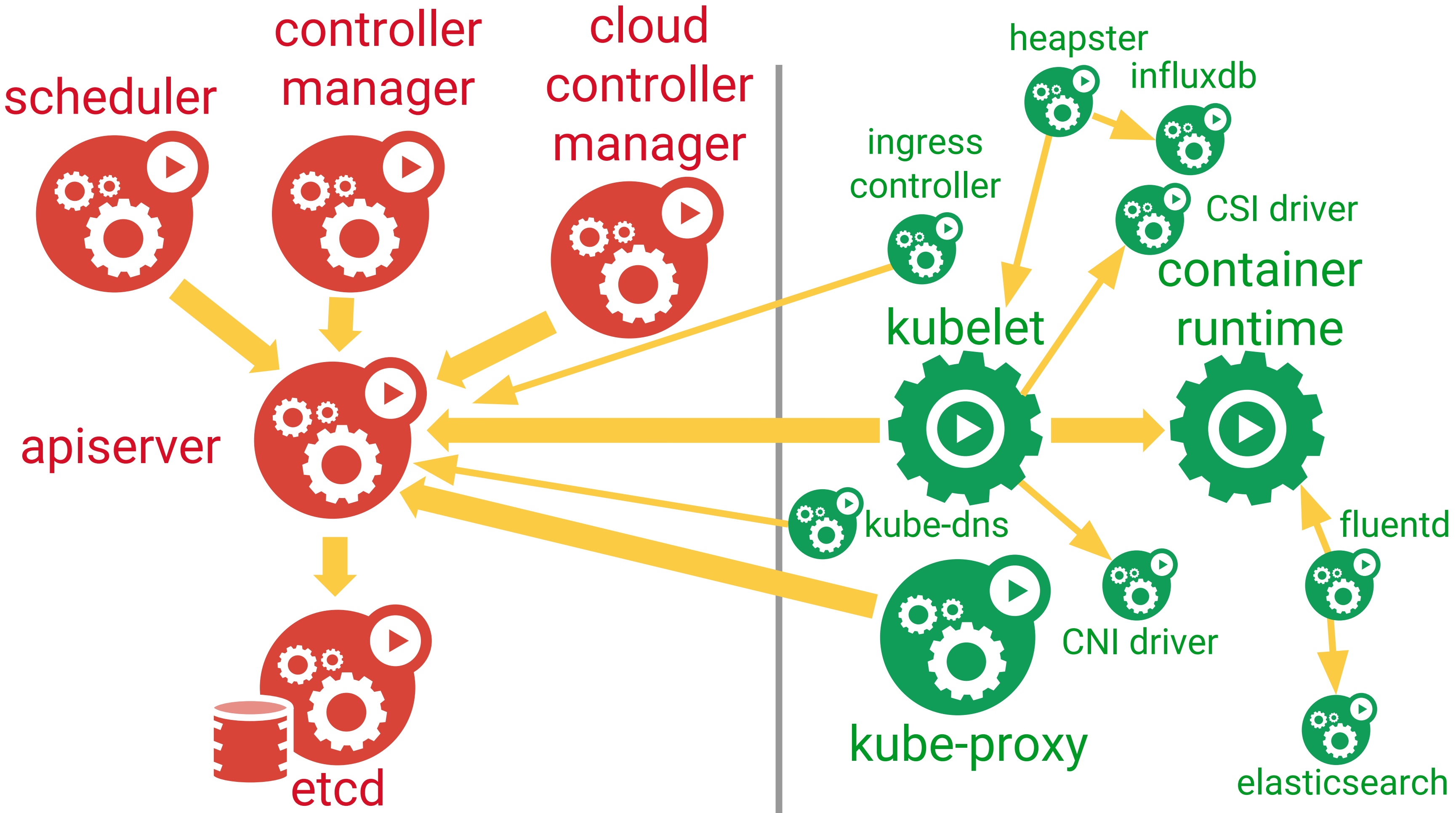
A fairly large project (~3M LOC)

Still growing

Not a monolithic program

A set of cooperating microservices


controller manager
scheduler
apiserver
etcd
kubelet
container runtime
kube-proxy


scheduler
controller manager
cloud controller manager
apiserver
etcd
heapster
influxdb
ingress controller
CSI driver
container runtime
kubelet
kube-dns
fluentd
CNI driver
kube-proxy
elasticsearch

Extensibility - Enabling out-of-tree

Drivers

- Network (CNI)
- Storage (flex, CSI)
- Device (e.g. GPU)
- Cloud providers

Container runtimes

Operators / controllers

Add-ons

- e.g. logging, monitoring

Some assembly required

Must Find Components!

Some assembly required

Some assembly required

Some assembly required

Must Find Components!

Download Components!

Version Skew!

Some assembly required

Must Find Components!

Download Components!

Unusable to most users

Version Skew!

Test Hell!

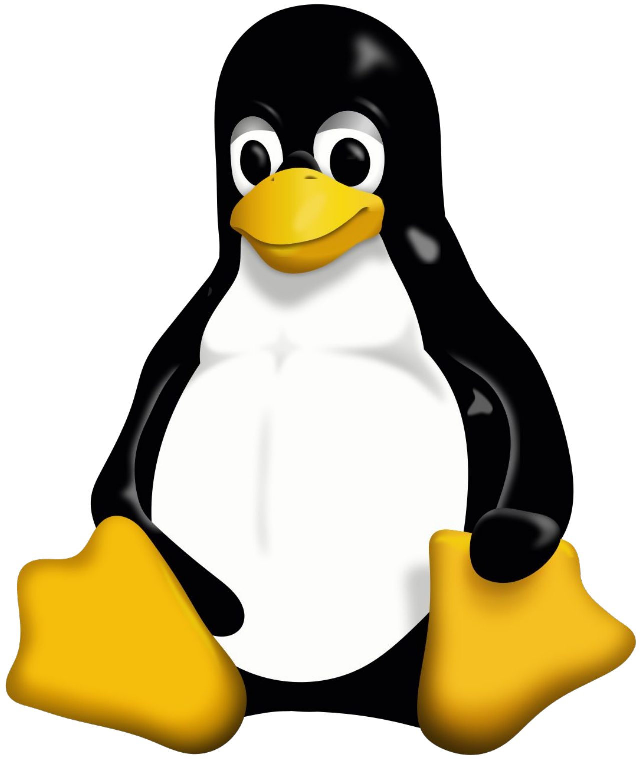
Comparative analysis

Linux is many projects

Decentralized & layered:

- Installers
- Kernel
- Bootup
- Shells
- Tools
- Programming Languages
- GUIs

Roughly “kernel” & “distro”



The kernel - Not useful alone

One big program

Lives in its own git repo
- No tools, add-ons, etc.

Releases fairly frequently
- 8-10 weeks

Has only X.Y versions
- Bugs in X.Y get fixed in X.Y+1
- X.Y.Z patch releases managed by community

Distributions

Everything else is developed separately

DIY systems are not tenable

ca. 1992, the concept of "Linux distros" emerged

EVERYONE uses a distro

Distributions

Distros serve different needs

Most users don't care about kernel version, just distro version

Generally release slowly

- Quarters to years

Emphasize their differences

- Technical & opinions

Distributions

Distro Value Add

- Platform support (drivers & extensions)
- Packaging & component lifecycle
- Support, security, & testing
- Simple installation

Kubernetes: Kernel or Distribution?

Is Kubernetes a kernel or a distro?

Releases quarterly

Includes enough to run *most* clouds

- But not all of them
- May not be true in the future

Contains a bunch of drivers

- But not all of them
- May not be true in the future

Is Kubernetes a kernel or a distro?

Integrates some 3rd party add-ons

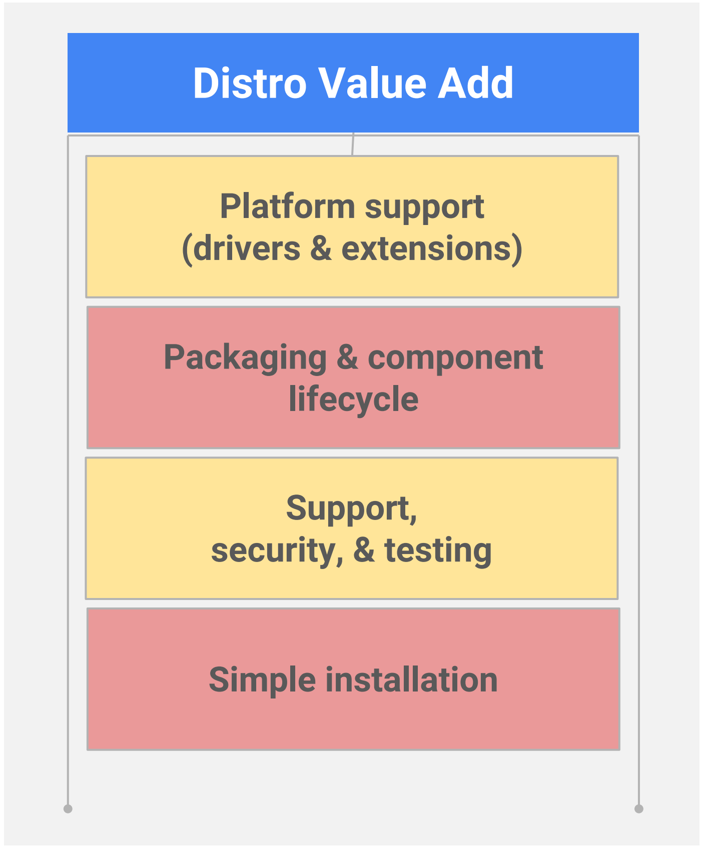
- But not many
- Does not usually carry patches

X.Y.Z releases and medium-term support

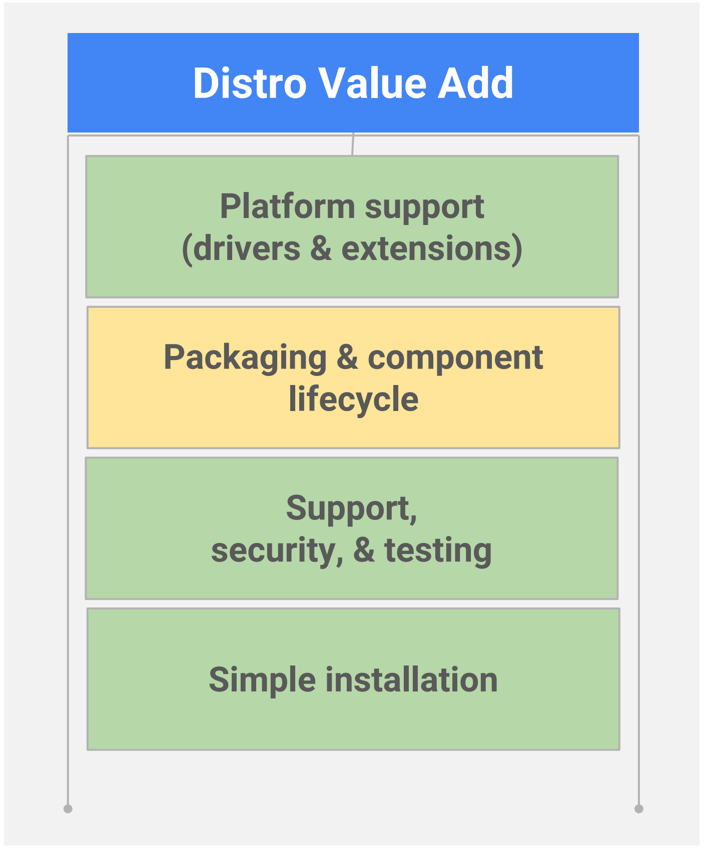
Lives in a small number of git repos

- Highly coupled components

Kubernetes upstream as a distro


Distro Value Add
Platform support (drivers & extensions)
Packaging & component lifecycle
Support, security, & testing
Simple installation

Other Kubernetes distros


Distro Value Add
Platform support (drivers & extensions)
Packaging & component lifecycle
Support, security, & testing
Simple installation

There are ALREADY more than 30 Kubernetes distributions!

- Clouds
- Enterprise vendors
- Higher level platforms
- Bespoke

Fragmentation risk: conformance

certified

kubernetes

* conformance program

Distros were inevitable.

Distros were inevitable.

How do we want to organize our project and community?

Option #1: Ignore it

Others *will* make distros

- No coordination or consistency
- We will have no say

Result:

- Fragmentation & politicization
- Many options, confusing UX
- Over time, converge on 3-4 distros?

Option #2: One distro to rule them all

Needs *huge* non-eng effort

Major distraction from k8s

Others will *still* do their own

- Opinions

Result: Probably failure, see option #1

Option 3: Find the middle-ground

Formalize what we already do

Focus on correctness and stability

Others will still do their own

- But hopefully based on ours?

Result:

- Clean up our thinking / processes
- Define tools, standards, etc.
- Derived distros benefit from staying close

Concrete Ideas

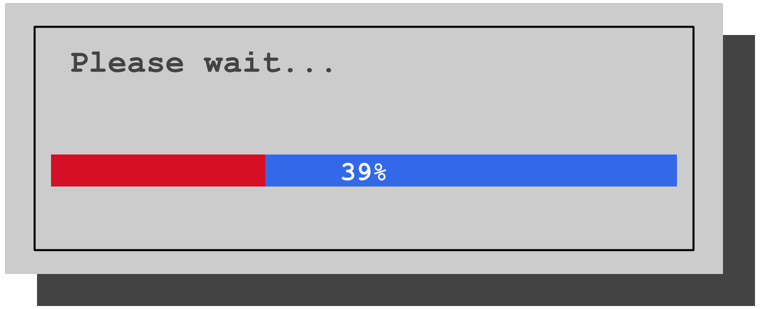
Installers

Pick ONE installer and make it great

- Which one? Contentious!

Or define a manifest format that installers consume?

- Think kernel config process and result



```
Please wait...

39%
```

Add-ons

Formalize “add-ons”

- Central repository
- Ownership
- Management mechanism
- Tooling
  - `kubectl apt-get update`?

Start with `cluster/addons/...`

- How to track upstreams?

Manage the kernel

Bound and extract "the kernel"

- Still multiple binaries
- No installer, add-ons, cloud providers

Manage it as a single component

Release it tick-tock style

- Features vs. stability

Kernel != distro

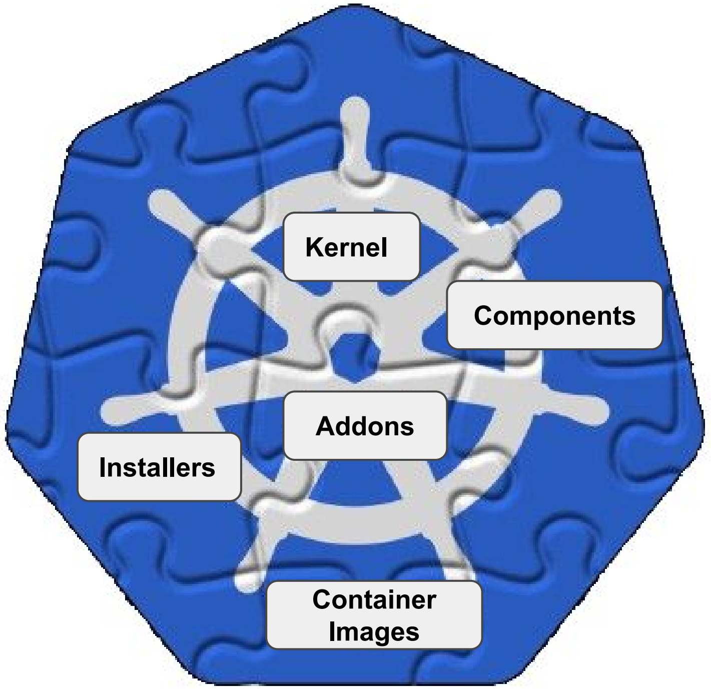
Manage the distro

Fork all code into our space

- Only ship things we control
- Carry patches IFF needed

Push everything to one repository

- No hunting all over the internet



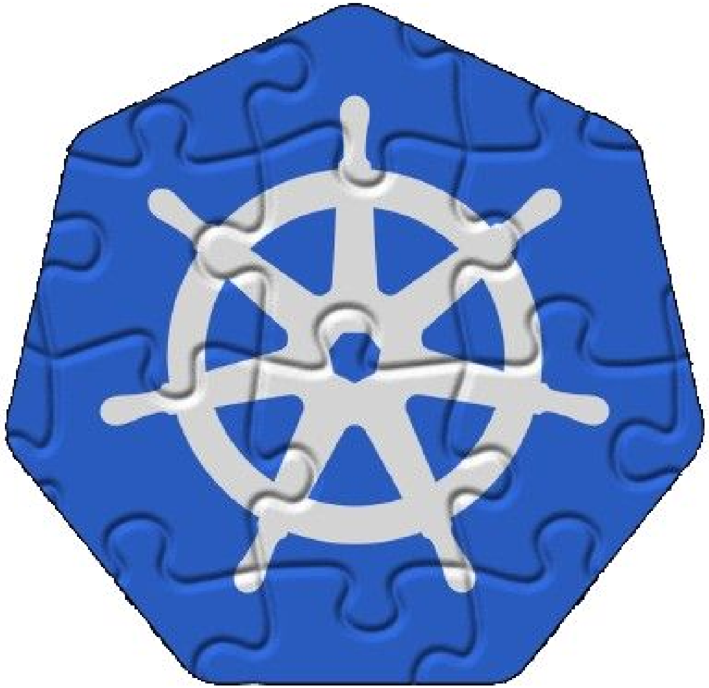
Manage the distro

Distinguish component version from package version

- e.g. 1.2.3-4 = “4th build of 1.2.3”

Release distro every 6-12 months

Base deprecation policies on distro releases



A new role - Distro Hero!

These are just some ideas

To pull this off, we need a **community**

- Different skills
- Different focus

https://commons.wikimedia.org/wiki/File:Placeholder_couple_superhero.png