ubuntu

The Evolution of the Kubernetes Model

Do you even kubernetes?

Matthew.Bruzek@ubuntu.com

Agenda

Our Kubernetes journey was not easy,
we learned some tips along the way

- Who, what, why
- Kubernetes
- Many models
- Where we are today
- Where we are going

Who?

What?

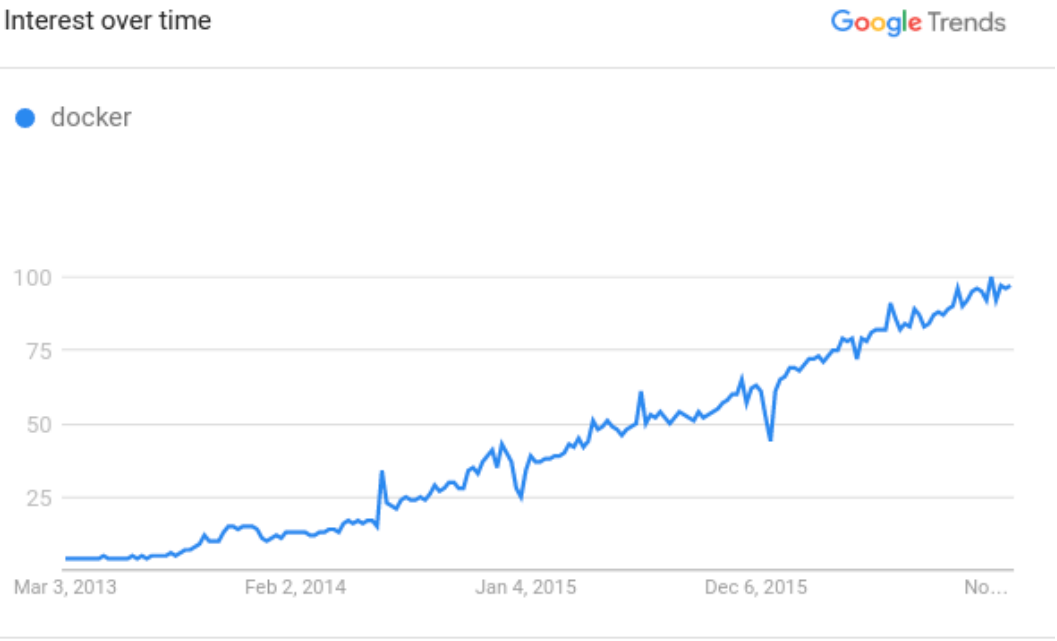
- **Kubernetes** is an orchestration platform for application containers
- From the ancient Greek word for “helmsman”
- Based on Google’s “Borg” project to manage and orchestrate containers: https://research.google.com/pubs/pub44843.html
- Announced in **March 2014**

What?

What are containers anyway?

- A method of isolating an application from the host system
- A lightweight alternative to full virtualization
  - By running through the host kernel
- Application containers == Docker
- Machine containers == LXC -> LXD

Why?



The explosive popularity of application containers means many more people are creating them

You need software tools to orchestrate and run this many containers

How?

Managing Kubernetes is not easy!

Take advantage of the solutions in the repository (getting started)

How?

- We use **Juju** to model the independent services and applications
  - Re-use common operation code
  - Best practices like transport layer security (TLS)
  - Separation of concerns, each charm is its own project
  - Independently scale the cluster parts to create a specific cluster
  - Deploy to any public cloud, bare metal and even a laptop *

* using LXD containers

- Juju has proved useful with the Kubernetes project which is moving so fast (breaking changes on almost every point release!)

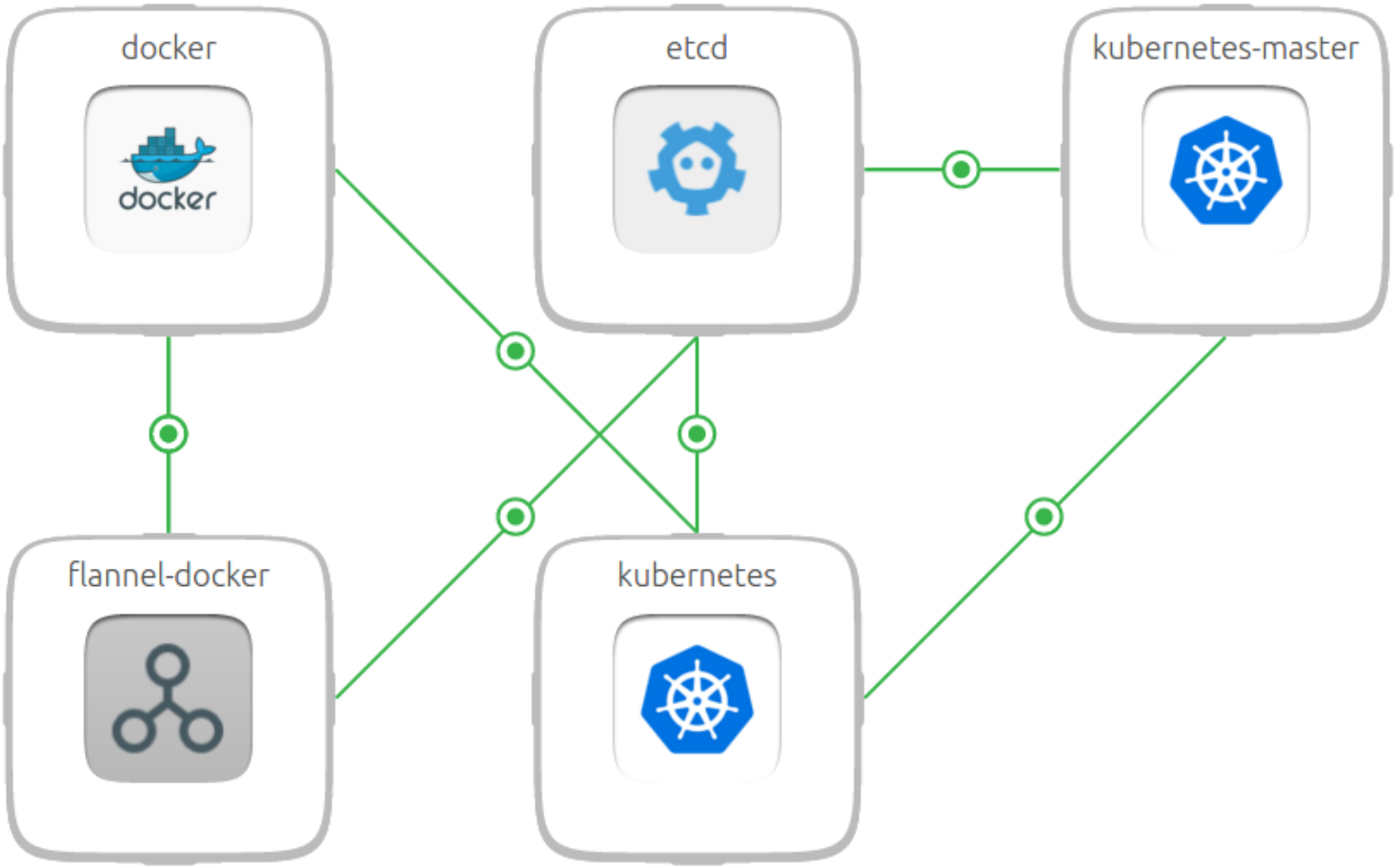
Kubernetes explained

- Kubernetes is declarative:
  - Operators declare what they want the cluster to look like
- Uses **etcd** as the distributed key/value store
- Smallest unit is a **pod**:
  - Can contain multiple application containers
  - Shared namespace
  - Each pod has an IP address
  - Applications have access to shared volumes
- Kubernetes supports container runtimes other than Docker

Kubernetes scheduler

```
while True:
    delta = diff(desired_state, current_state)
    schedule(delta)
```

First iteration



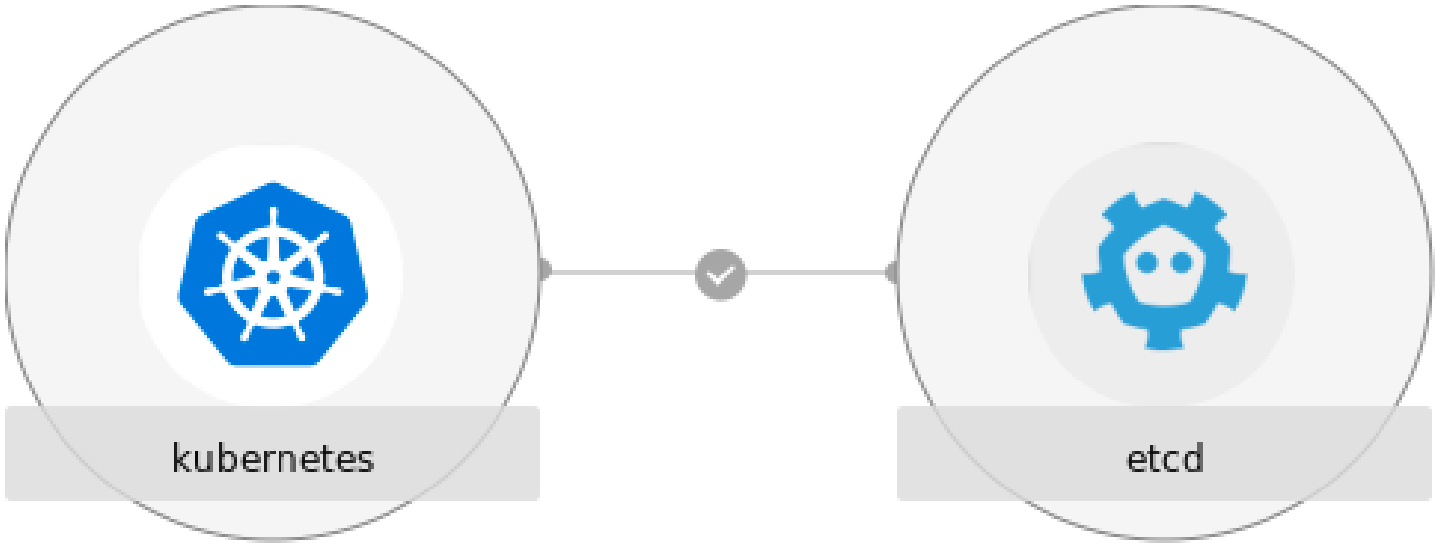
Minions

Second iteration

Hyperkube - the all-in-one binary for the Kubernetes server components

- Every node in the cluster is both a Kubernetes master and a minion
- All services ran in containers:
  - apiserver
  - controller-manager
  - scheduler
  - kubelet
  - proxy
- Etcd runs on separate hosts
- Flannel ran on the host VM

Second iteration



Running all-in-one did not work for us!

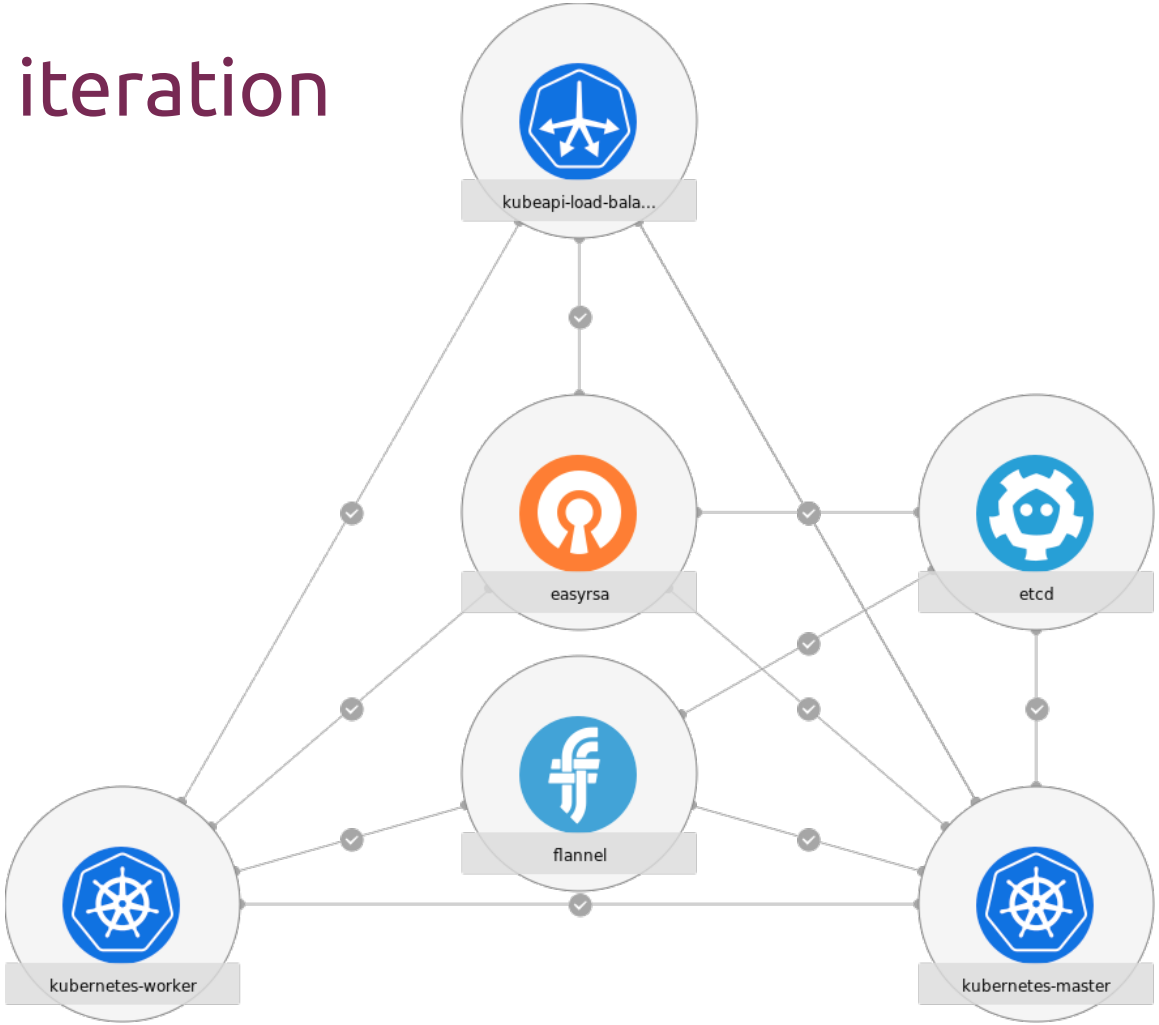
- The container runtime and hyperkube introduced two points of failure into our model
  - Docker version updates were problematic and risky
  - The version of hyperkube itself was risky to update while cluster was running

Running all-in-one was not flexible!

- Breaks the separation of concerns design principle
- Not very flexible when it comes to node VM sizes
  - The worker nodes need to be large, but the master components can be relatively smaller
- Wasteful with compute and networking resources
- Unable to scale the independant parts

Outdated terminology, minions --> node

Current iteration



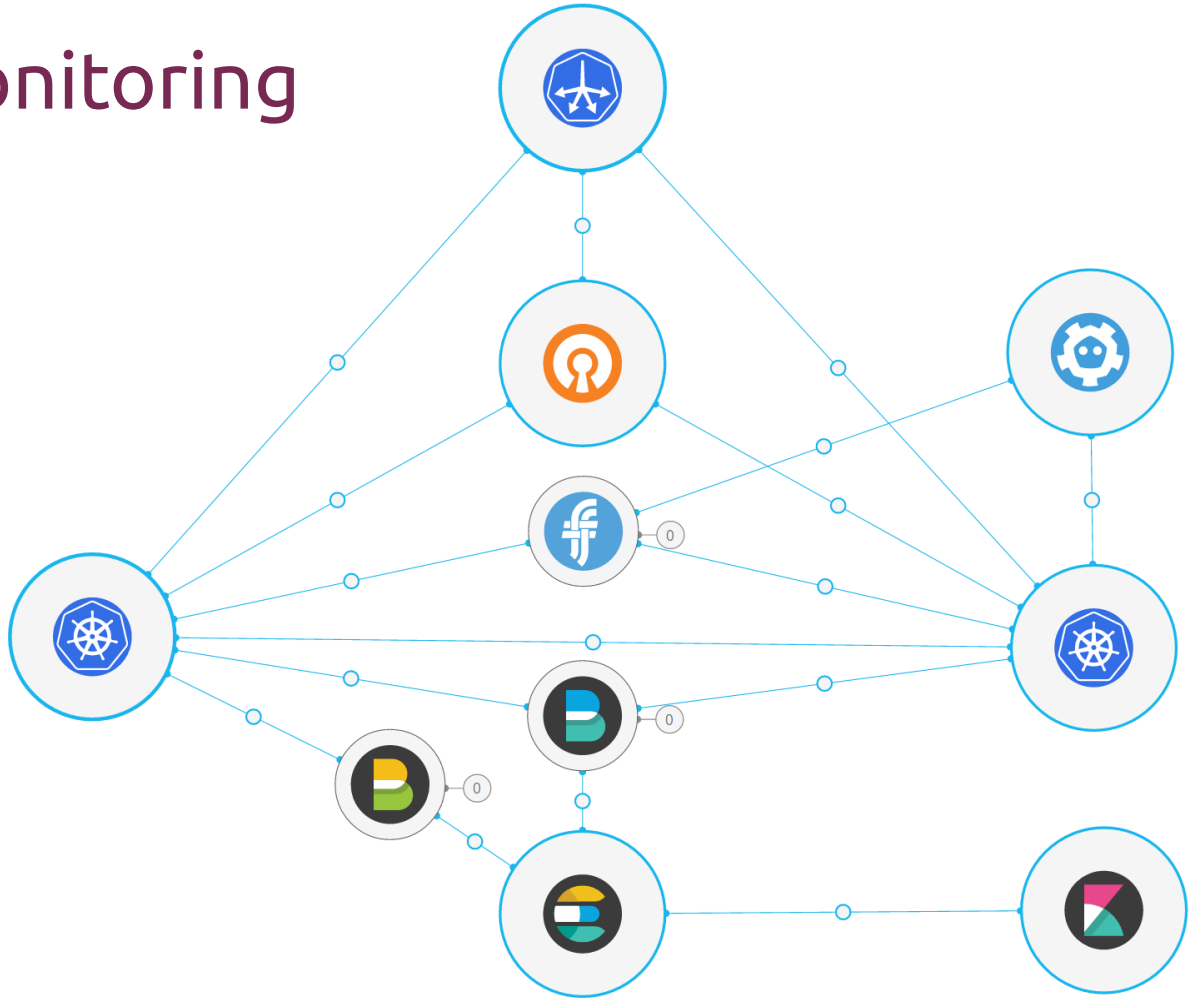
Upstream!

https://github.com/kubernetes/kubernetes.git

The Juju charms and bundles were accepted in the upstream Kubernetes project!

- `cluster/juju` directory is the official way to deploy Kubernetes on Ubuntu
- Kubernetes is a large fast moving project with thousands of contributors
- Contributing to a large project open source
  - Life achievement unlocked, #winning!

Elastic monitoring



We learn best by breaking things

Therefore it must be cheap to try things that can break

Lessons learned

- Keep your addons manifests up-to-date
  (kube-dns, heapster, grafana, influxdb)
- Do not use caching proxies if you load balance the masters
- Transport layer security is hard, automate, and make repeatable
- **C**ontainer **N**etwork **I**nfrastructure (CNI) is ready,
  it supports different Software Defined Networks (SDN)

Lessons learned … part 2

- Make it easy to debug your cluster
- Use small containers to test networking
- Embrace change!
  - Kubernetes components flags change on minor version boundaries
- Do not reboot your systems while doing a kubernetes operation
- Do not deploy a cluster on your laptop during a conference call

kubectl is your new friend

- **kubectl** is the client command line tool for Kubernetes
- It requires kubeconfig `~/.kube/config` by default
- Production clusters need the keys in the configuration file
- The full kubeconfig is already generated within a Juju cluster

Common kubectl commands:

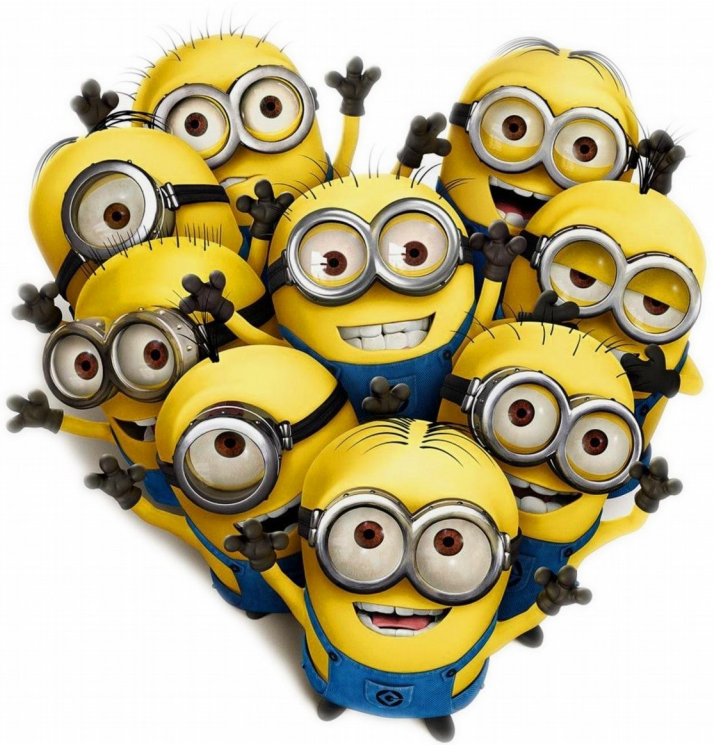
- `kubectl describe`
- `kubectl apply` rather than `kubectl create` for idempotent scheduling of manifest files
- `kubectl exec`
  - kubectl exec pod-name -i -t -- /bin/bash
- `kubectl logs`

Future items

- Integration with other monitoring and logging technologies
- More architectures, ppc64le, and s390x
- More packaging formats
- Etcd 3.0
- Different Software Defined Networks (SDN)

Join the community

- Are you a professional operator?
  - Contribute your operations code to upstream Kubernetes project
- Do you work on a component that interfaces with Kubernetes?
  - Create a charm with your component
- Update the documentation or fill in the gaps
  - https://kubernetes.io/editdocs/



The operations are in the repository

https://github.com/kubernetes/kubernetes.git

Try it out!

```
sudo snap install conjure-up --classic --beta

conjure-up kubernetes
```

Contact us

#juju in chat.freenode.net on IRC

juju@lists.ubuntu.com

Google **S**pecial **I**nterest **G**roups

sig-cluster-ops
sig-lifecycle
sig-onprem