LOCAL KUBERNETES FOR DUMMIES (AKA ME)

So you want to move up from docker-compose?

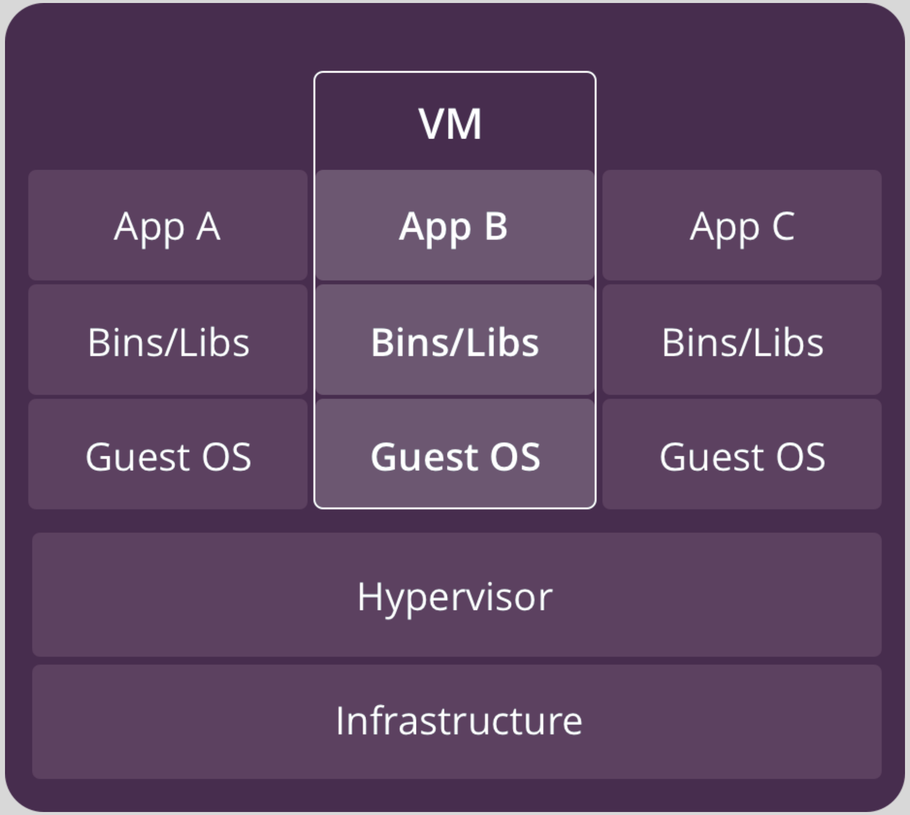
Andrew Denner
STLLUG
March, 2021

About me—Andrew Denner

- Senior Scientific Software developer for Large Agricultural Company
- President of Central Iowa Linux Users Group (CIALUG)
- I will share slides after talk, and at http://denner.co
- Twitter: @adenner

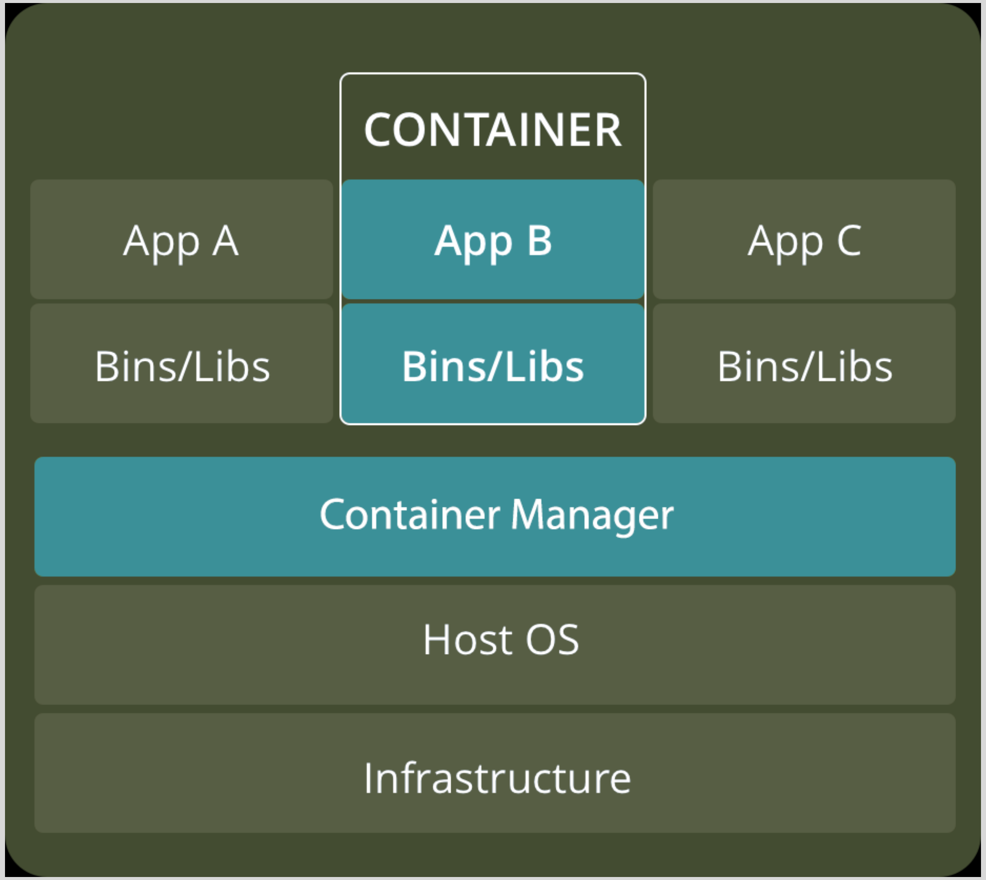
Brief Intro and level setting

- Why do we care?
  - Bare metal servers are big, best running many things
- VM
  - Types of VM:
    - Vmware vSphere
    - Virtual Box
    - Xen
    - Hyper-V
    - KVM
  - Heavy Weight (time cpu memory)
  - Guest runs overtop of host os/hypervisor
  - Can be mixed, i.e. windows and Linux together
  - Less sharing
  - Full isolation



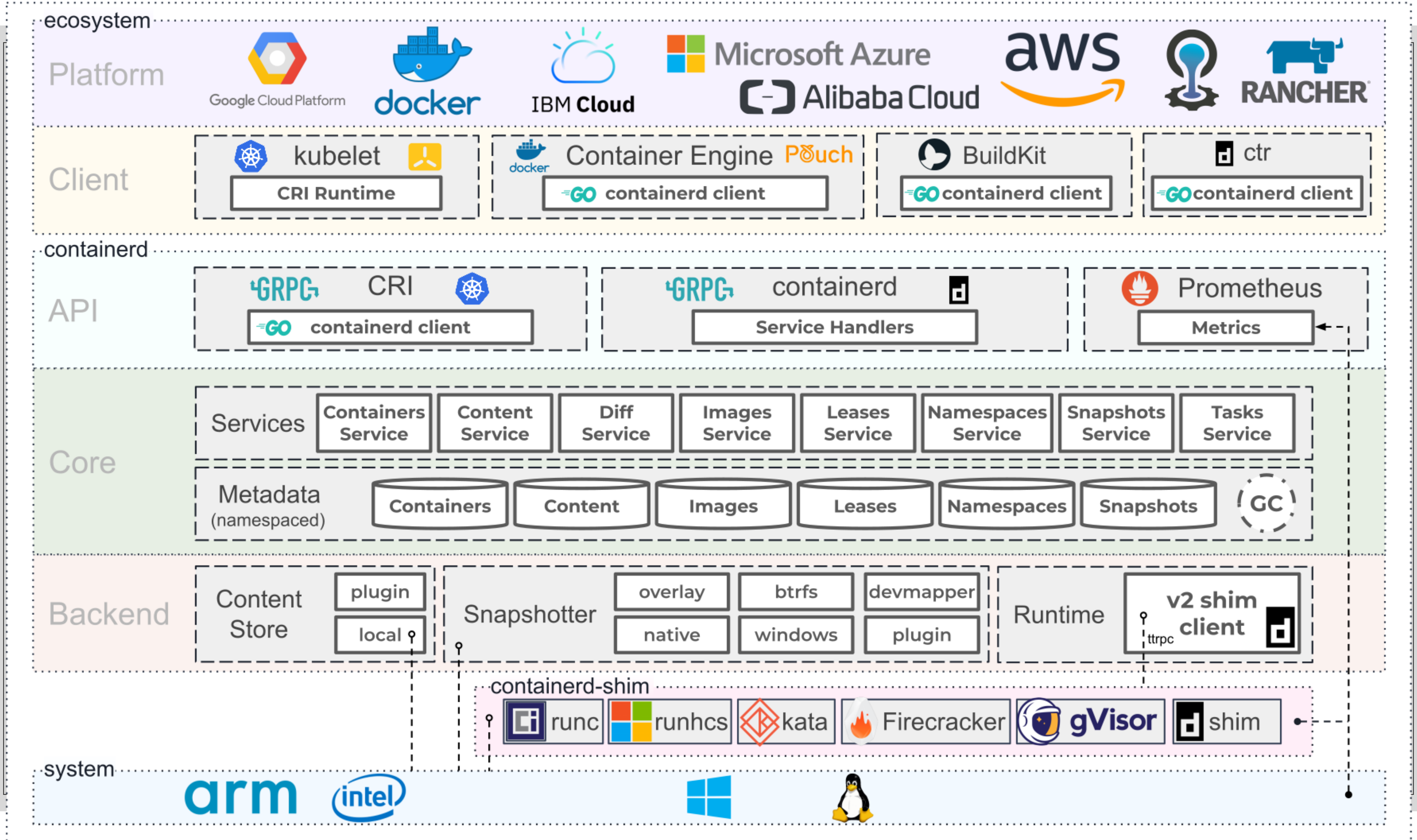
Brief Intro and level setting (Cont.)

- Containers
  - More Shared—Host Kernel
  - Namespaces et.al. to be "vm like"
  - Types:
    - LXC
    - Docker
    - Podman
    - Containerd (CNCF)
  - Benefits:
    - Light and fast (memory, startup, size)
    - Native performance
    - Process level isolation
    - By default dockerd is running as root (rootless)
    - Container repository



Amazon says…

“Docker is a software platform that allows you to build, test, and deploy applications quickly. Docker packages software into standardized units called containers that have everything the software needs to run including libraries, system tools, code, and runtime. Using Docker, you can quickly deploy and scale applications into any environment and know your code will run.”



ecosystem

Platform

Google Cloud Platform · docker · IBM Cloud · Microsoft Azure · Alibaba Cloud · aws · RANCHER

Client

kubelet — CRI Runtime

Container Engine (docker, Pouch) — containerd client

BuildKit — containerd client

ctr — containerd client

containerd

API

GRPC CRI — containerd client

GRPC containerd — Service Handlers

Prometheus — Metrics

Core

Services: Containers Service, Content Service, Diff Service, Images Service, Leases Service, Namespaces Service, Snapshots Service, Tasks Service

Metadata (namespaced): Containers, Content, Images, Leases, Namespaces, Snapshots, GC

Backend

Content Store: plugin, local

Snapshotter: overlay, btrfs, devmapper, native, windows, plugin

Runtime: v2 shim client, ttrpc

containerd-shim

runc · runhcs · kata · Firecracker · gVisor · shim

system

arm · intel

DRAWBACKS TO DOCKER

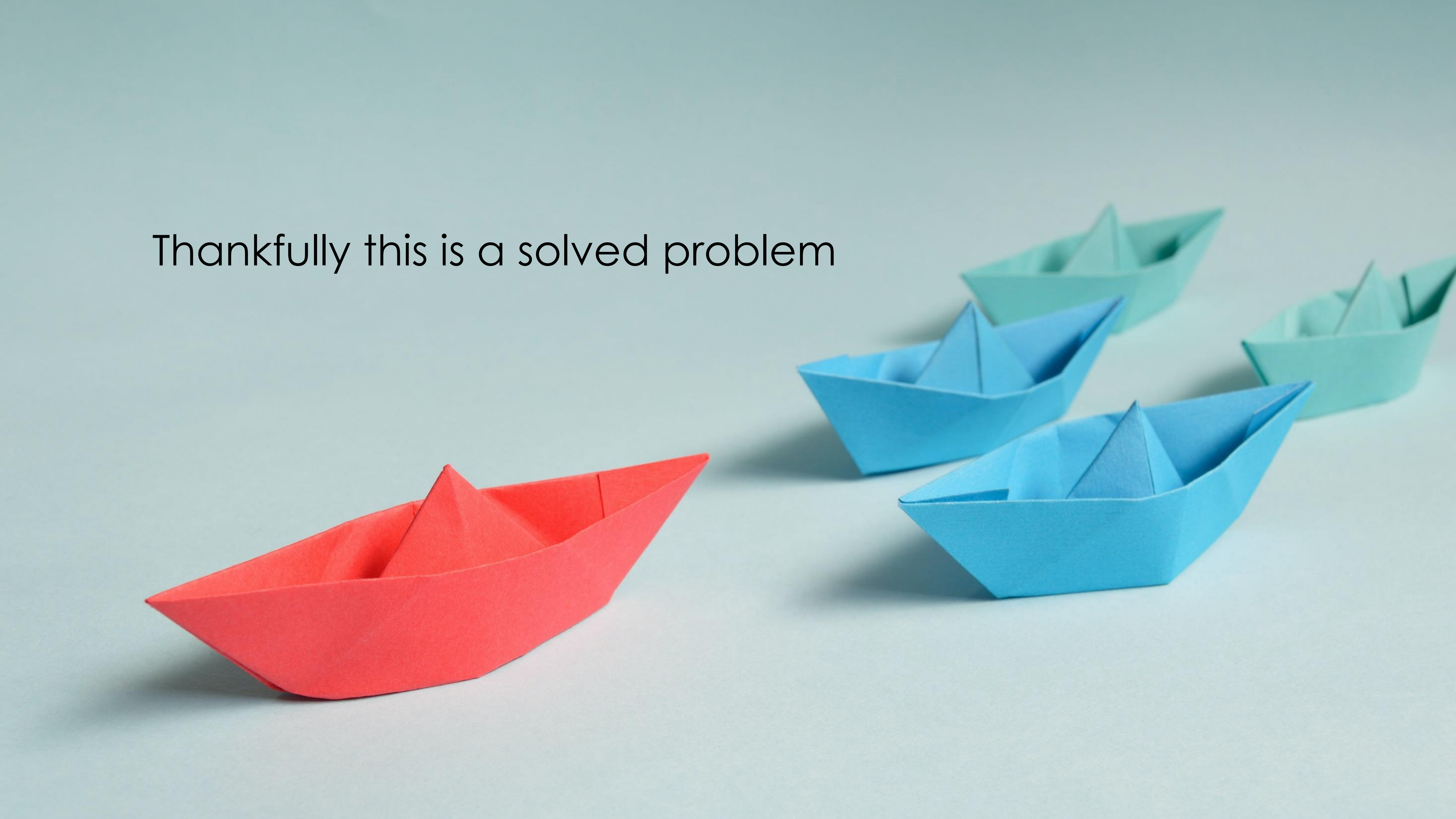
Orchestration of multiple containers

Networking headaches

Still just one machine—shifting of the snowflake

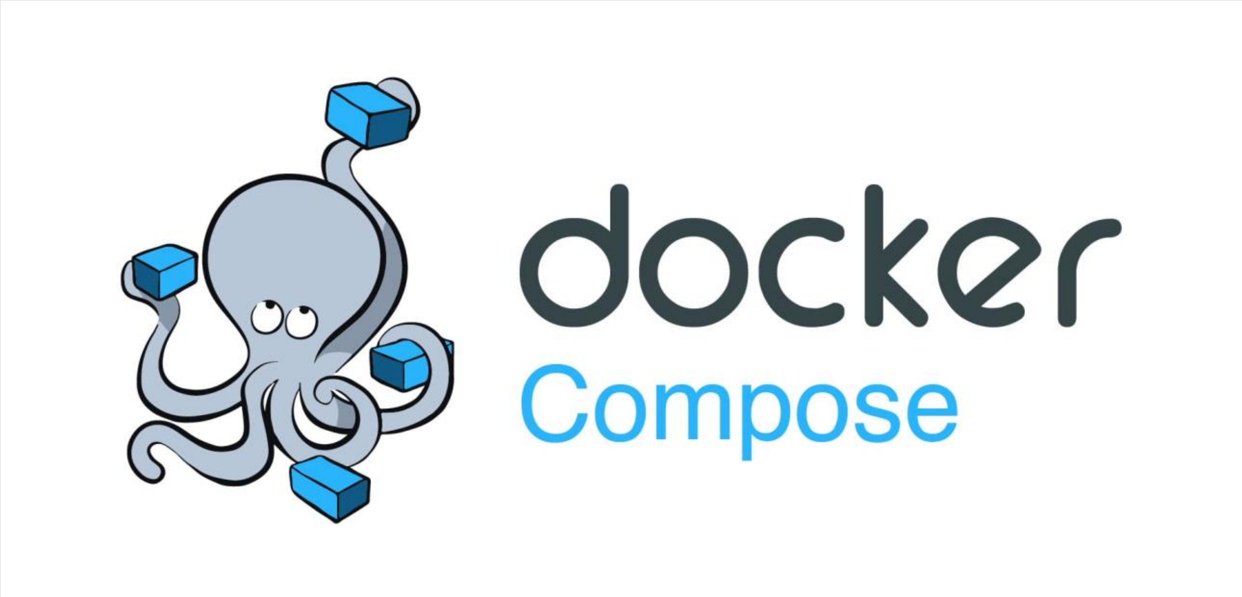
You're gonna need a bigger boat.

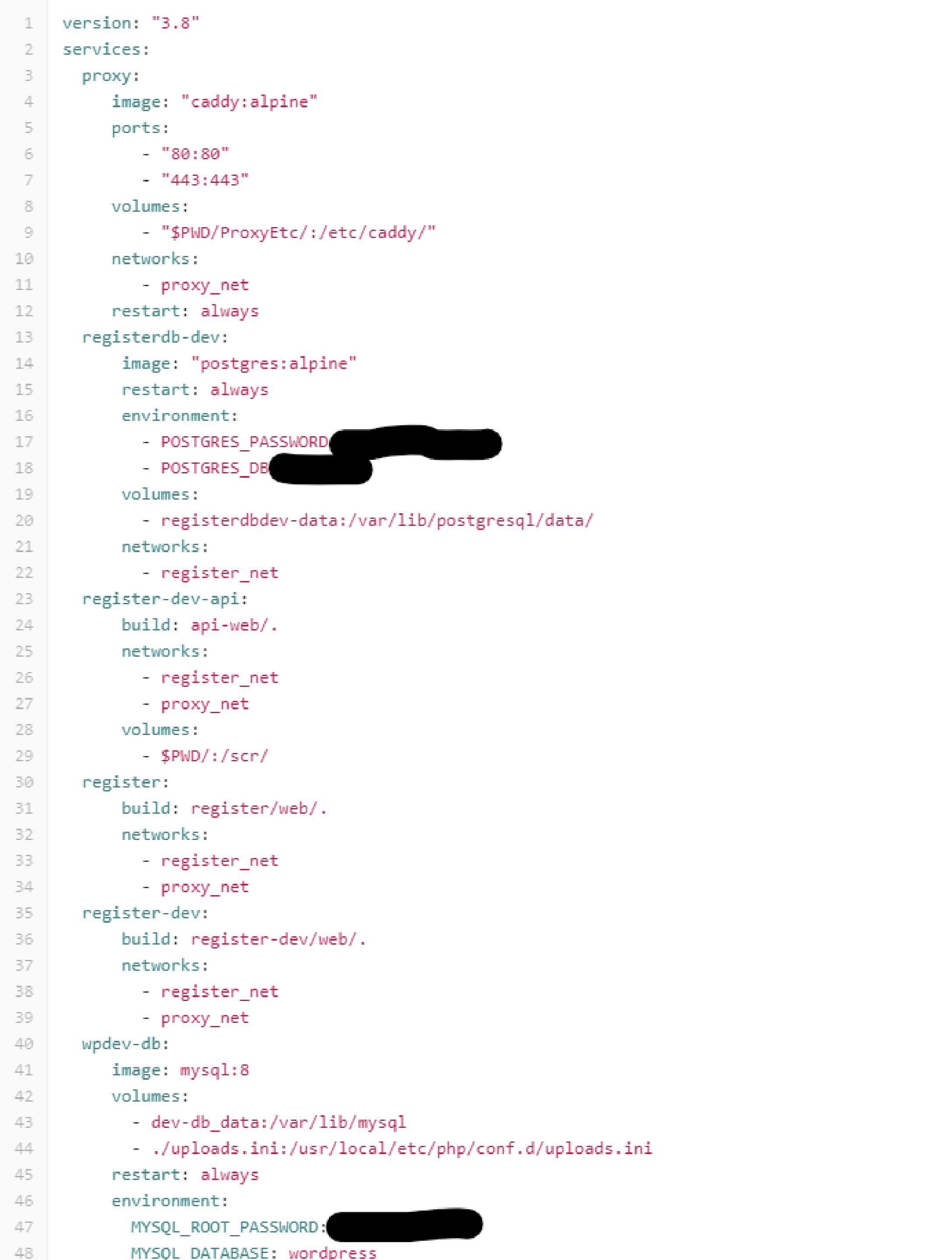
Thankfully this is a solved problem



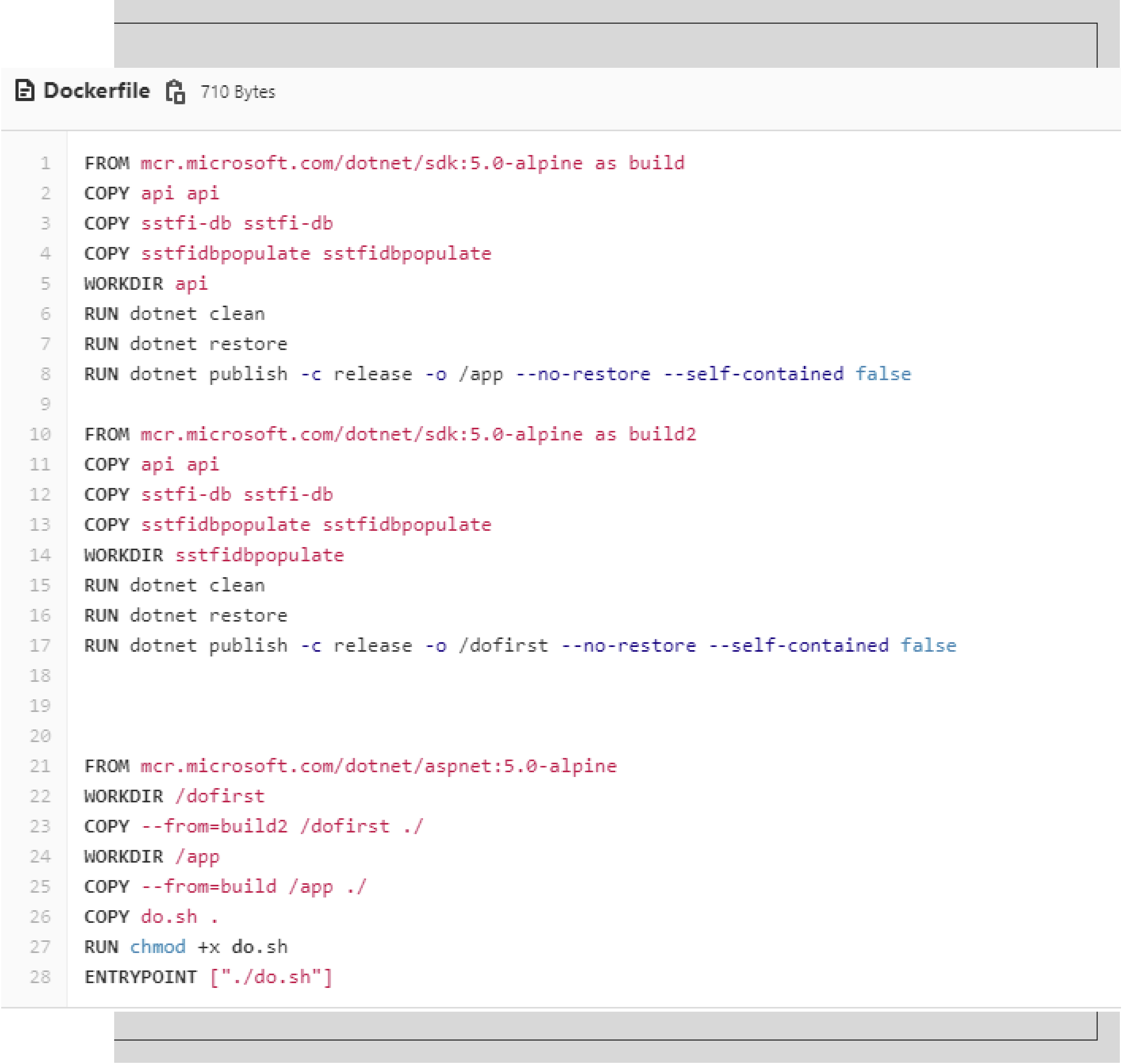
Docker-compose

- Can create docker-compose.yaml files recipes for a full set of images
- Cures orchestration challenge but you have to roll your own yaml
- Easy—Yaml files
- By default still on one machine
- Apparently may be able to hit k8s as well
  (I haven't tried this) (link)





```
version: "3.8"
services:
  proxy:
     image: "caddy:alpine"
     ports:
        - "80:80"
        - "443:443"
     volumes:
        - "$PWD/ProxyEtc/:/etc/caddy/"
     networks:
        - proxy_net
     restart: always
  registerdb-dev:
      image: "postgres:alpine"
      restart: always
      environment:
        - POSTGRES_PASSWORD[redacted]
        - POSTGRES_DB[redacted]
      volumes:
        - registerdbdev-data:/var/lib/postgresql/data/
      networks:
        - register_net
  register-dev-api:
      build: api-web/.
      networks:
        - register_net
        - proxy_net
      volumes:
        - $PWD/:/scr/
  register:
      build: register/web/.
      networks:
        - register_net
        - proxy_net
  register-dev:
      build: register-dev/web/.
      networks:
        - register_net
        - proxy_net
  wpdev-db:
     image: mysql:8
     volumes:
       - dev-db_data:/var/lib/mysql
       - ./uploads.ini:/usr/local/etc/php/conf.d/uploads.ini
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD:[redacted]
       MYSQL_DATABASE: wordpress
```



Dockerfile 710 Bytes

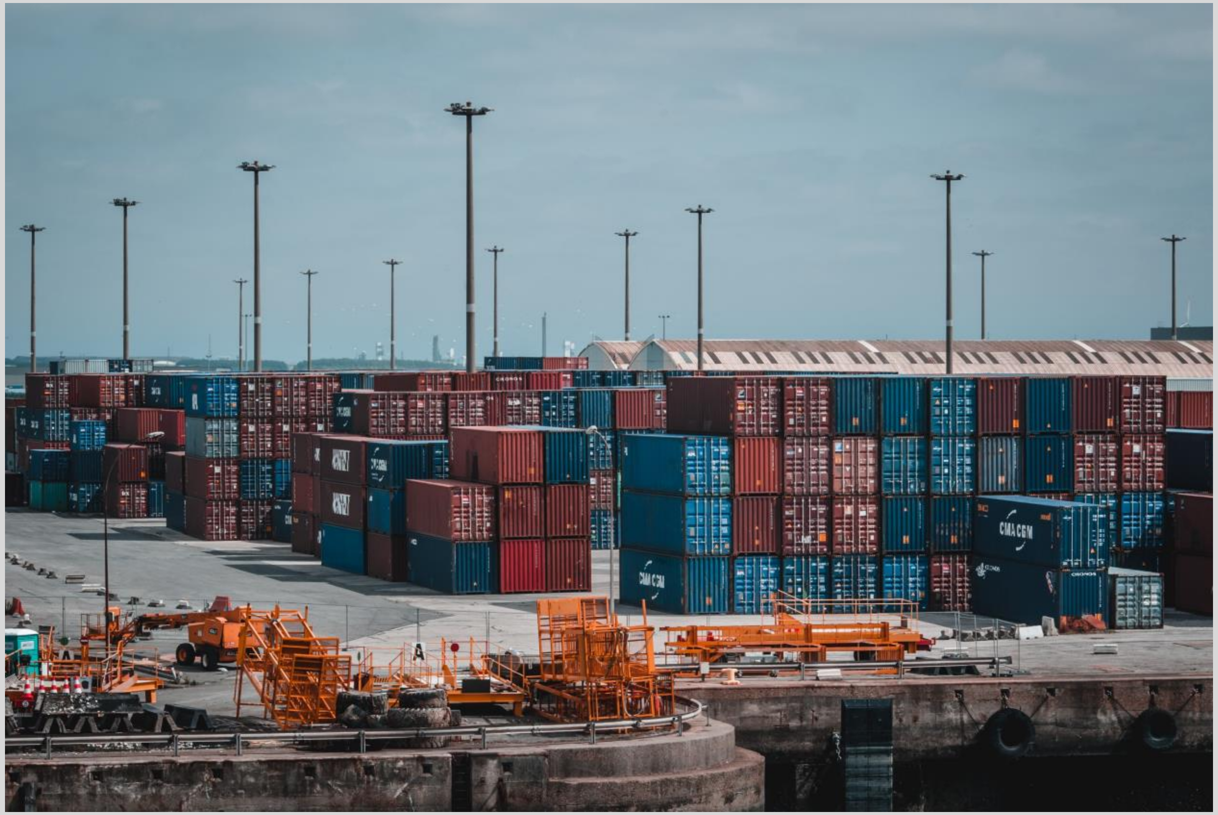
```
FROM mcr.microsoft.com/dotnet/sdk:5.0-alpine as build
COPY api api
COPY sstfi-db sstfi-db
COPY sstfidbpopulate sstfidbpopulate
WORKDIR api
RUN dotnet clean
RUN dotnet restore
RUN dotnet publish -c release -o /app --no-restore --self-contained false

FROM mcr.microsoft.com/dotnet/sdk:5.0-alpine as build2
COPY api api
COPY sstfi-db sstfi-db
COPY sstfidbpopulate sstfidbpopulate
WORKDIR sstfidbpopulate
RUN dotnet clean
RUN dotnet restore
RUN dotnet publish -c release -o /dofirst --no-restore --self-contained false



FROM mcr.microsoft.com/dotnet/aspnet:5.0-alpine
WORKDIR /dofirst
COPY --from=build2 /dofirst ./
WORKDIR /app
COPY --from=build /app ./
COPY do.sh .
RUN chmod +x do.sh
ENTRYPOINT ["./do.sh"]
```

Docker Swarm

- Extended mode of Docker
- “Swarm” of docker hosts
- Simple to setup, but less flexible
- Shares docker command structure
- You will eventually hit the wall, In my case eventually code was going to be deployed to k8s anyway so why not start there
- Really not the direction that industry is going

Kubernetes

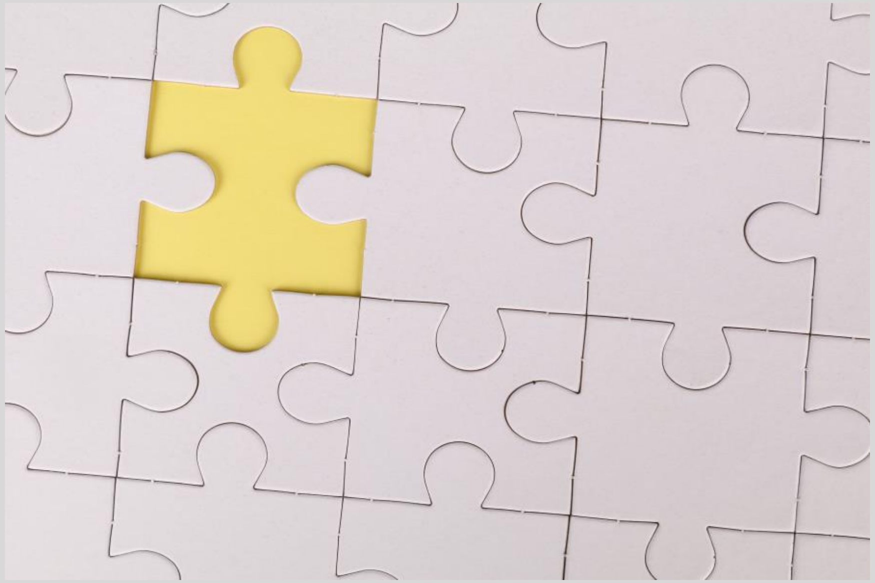
- Abbreviated K8s (8 letters between k and s)
- Greek for helmsman
- Originally from Google in 2014—Planet Scale
- Think docker compose writ large
- What it provides:
  - Service Discovery
  - Storage Orchestration
  - Bin Packing
  - Self Healing
  - Secret Management
  - Industry Standards

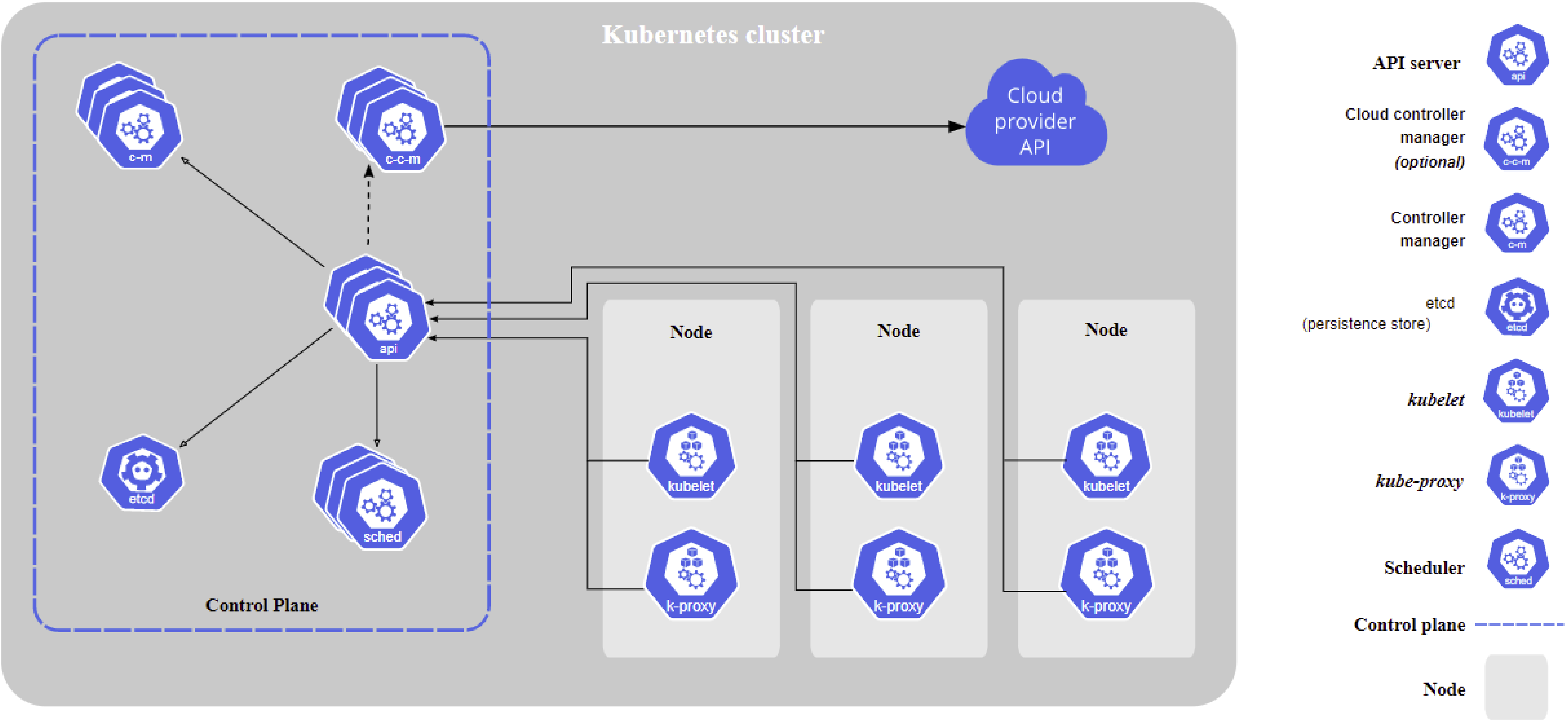


*Batteries not included

What it isn’t

- Not a drop-in replacement for docker
- No limits to what can run—Long running services, short batches
- Does not auto deploy source code—CICD not included
- No middleware, message bus et.al. but can run as yet another container
- Really hard to setup by default (think linux from scratch)




Kubernetes cluster
c-m
c-c-m
Cloud provider API
api
etcd
sched
Control Plane
Node
kubelet
k-proxy
Node
kubelet
k-proxy
Node
kubelet
k-proxy
API server
api
Cloud controller manager (optional)
c-c-m
Controller manager
c-m
etcd (persistence store)
etcd
kubelet
kubelet
kube-proxy
k-proxy
Scheduler
sched
Control plane
Node

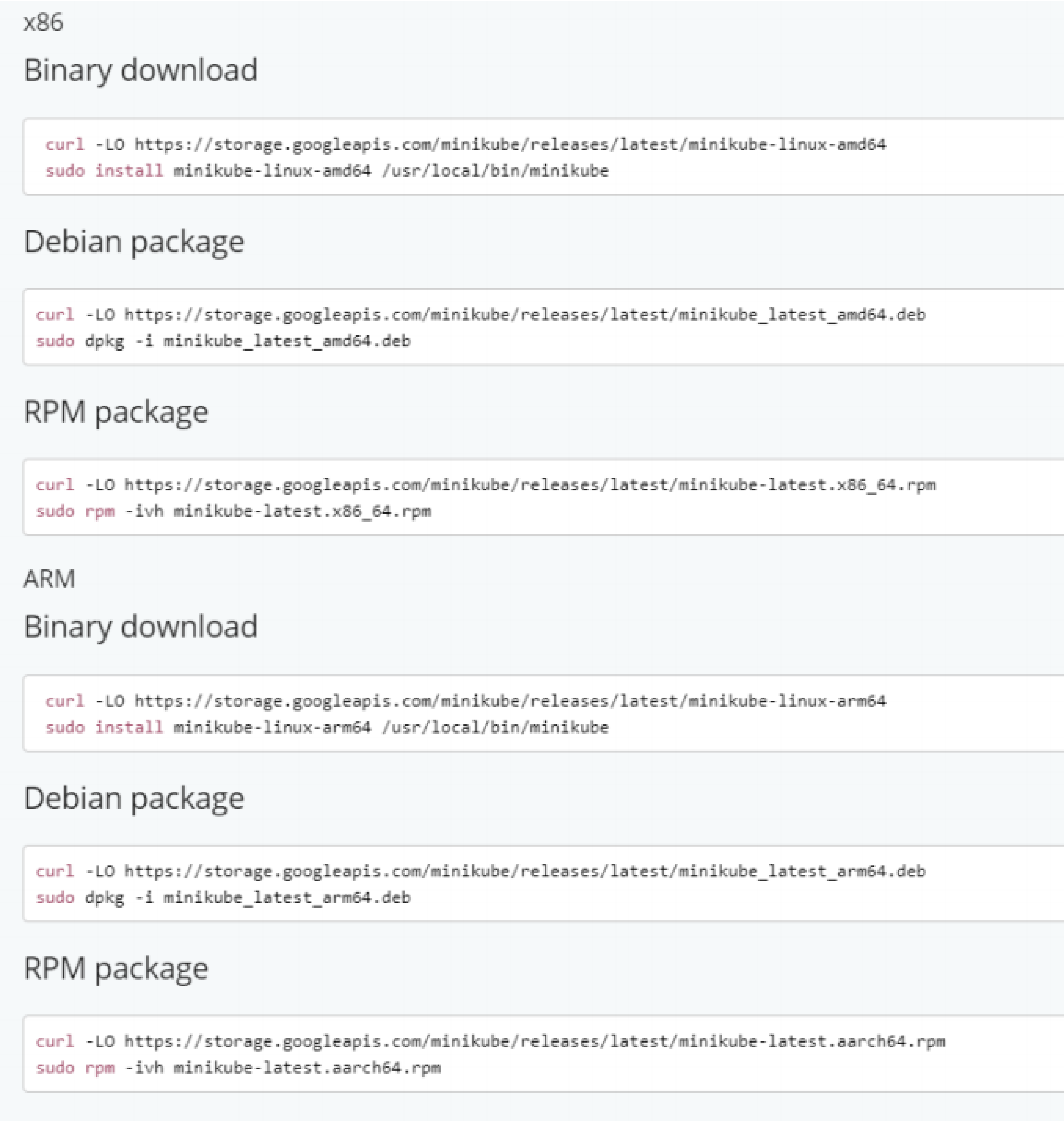
K8s terms

- Cluster—Nodes that containerized apps run on
- Controller—Manage state (located in the control plane) deployment controller daemonset namespace controller and persistent volume controller
- Manifest—JSON or YAML that specifies desired state of K8s object, create modify delete things like pods deployments and services
- Pod—Base k8s object, group (or one) of containers running on cluster
- Volume—Directory with data accessible by containers in pod
- Workload—Application running on k8s (Deployments, statefulsets, daemonsets, jobs, cronjobs)
- Kubectl—cmd line config tool create, inspect, update, delete

Confused yet?

Easier button Minikube

- (https://minikube.sigs.k8s.io/)
- 2 CPUs or more
- 2GB of free memory
- 20GB of free disk space
- Internet connection



x86

Binary download

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

Debian package

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube_latest_amd64.deb
sudo dpkg -i minikube_latest_amd64.deb
```

RPM package

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-latest.x86_64.rpm
sudo rpm -ivh minikube-latest.x86_64.rpm
```

ARM

Binary download

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-arm64
sudo install minikube-linux-arm64 /usr/local/bin/minikube
```

Debian package

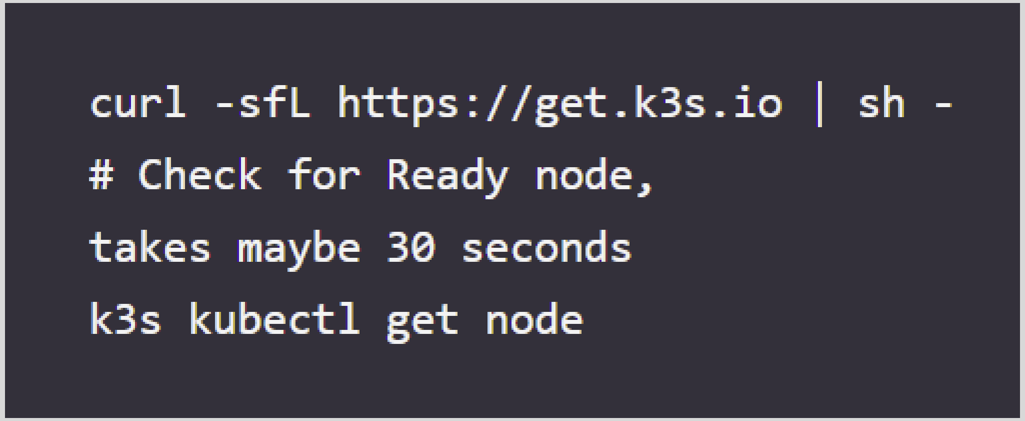
```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube_latest_arm64.deb
sudo dpkg -i minikube_latest_arm64.deb
```

RPM package

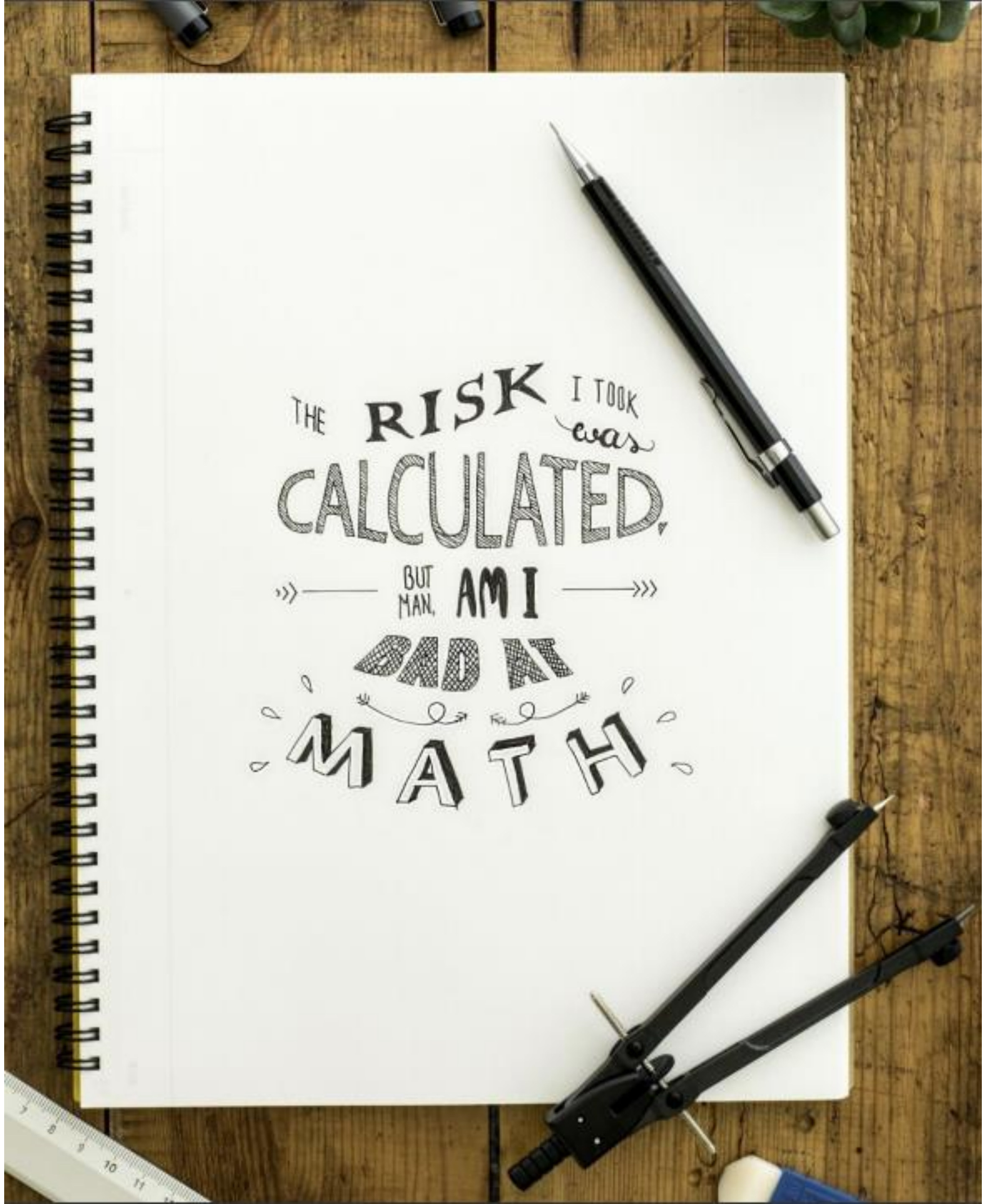
```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-latest.aarch64.rpm
sudo rpm -ivh minikube-latest.aarch64.rpm
```

Kubernetes, just the good parts (k3s)

- Kubernetes is huge and hard
- K3s stripped out the bad parts and is easier
- <40 mb binary
- Can run ARM
- Easy to set up!

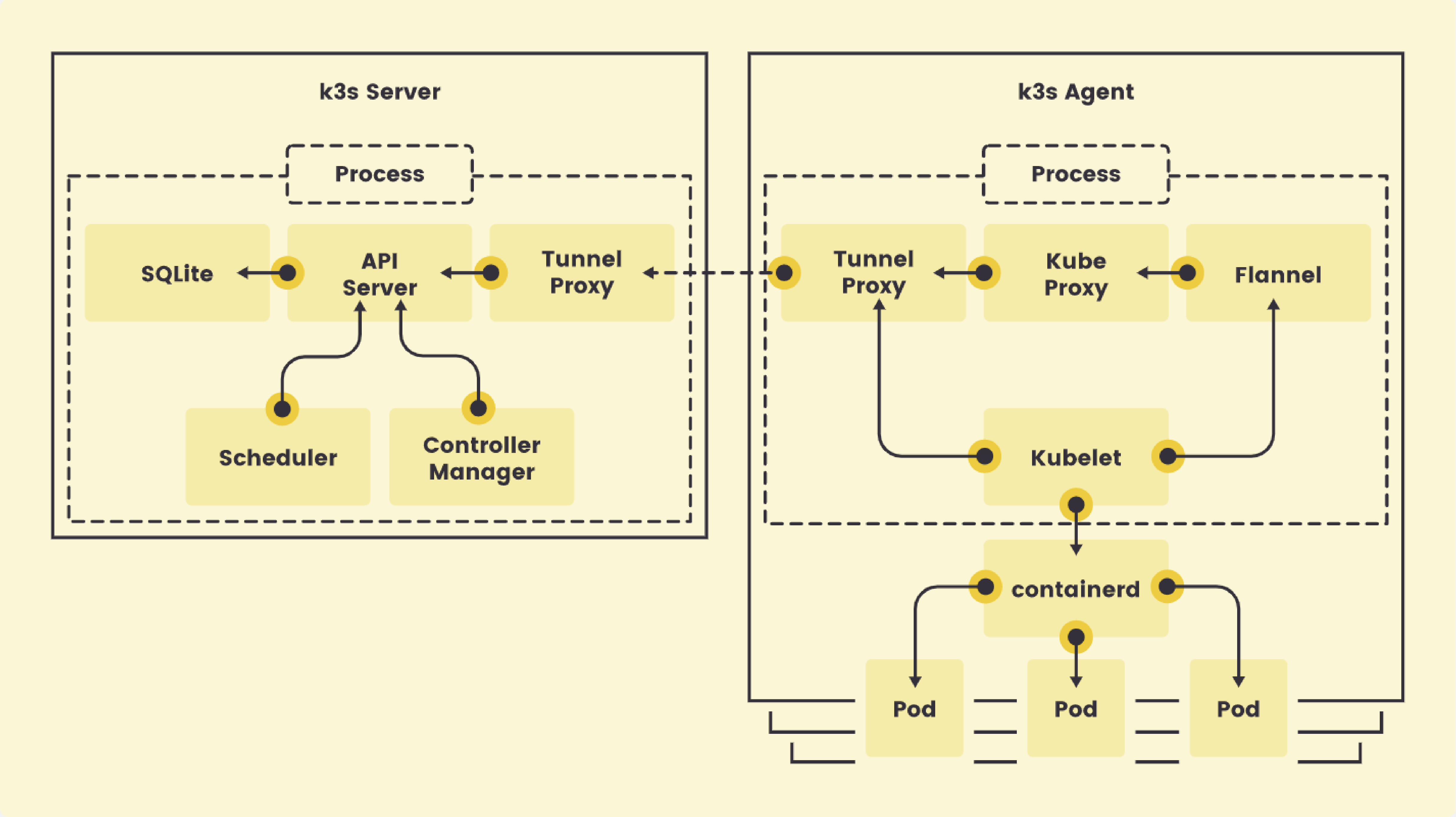


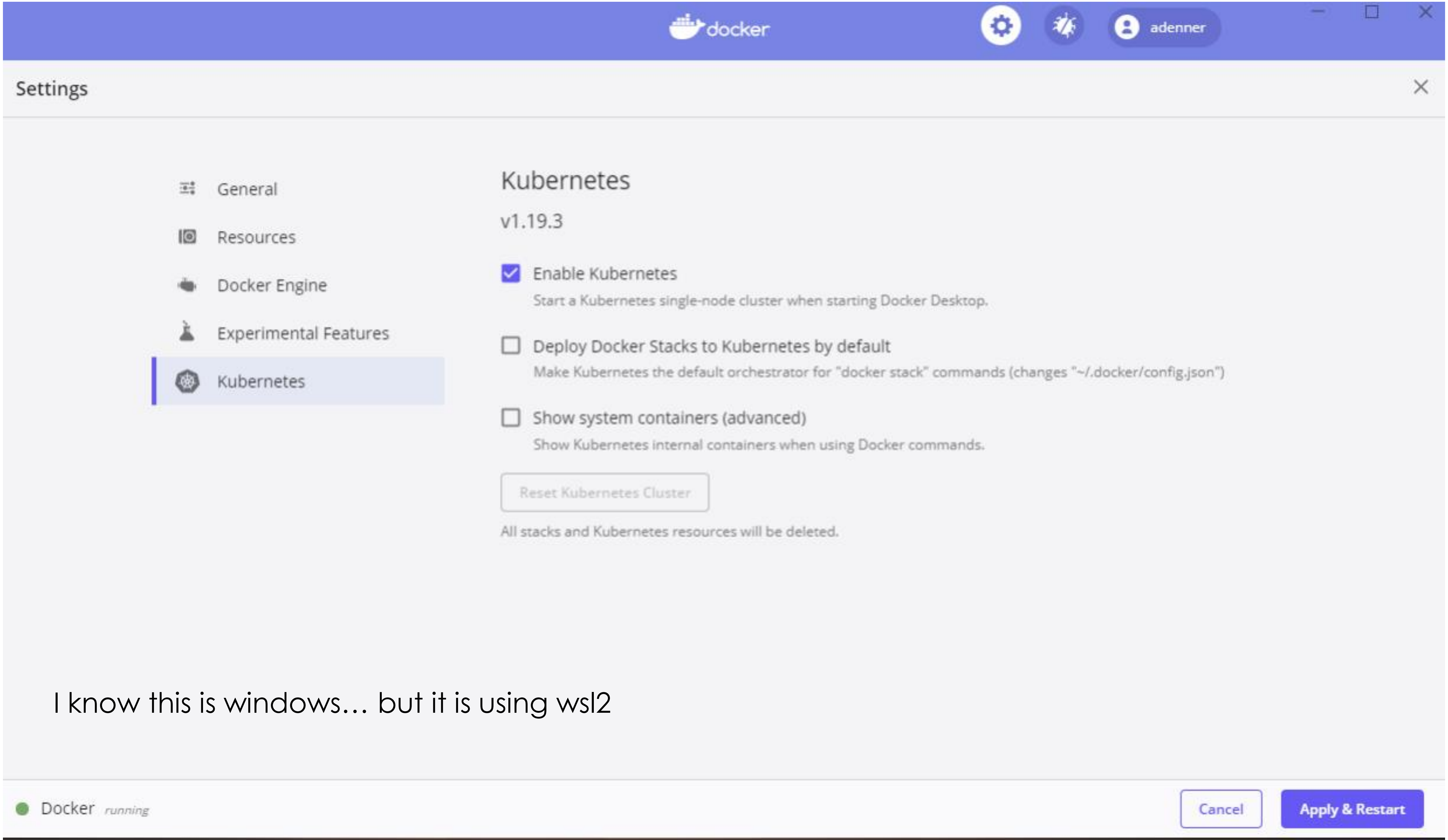
```
curl -sfL https://get.k3s.io | sh -
# Check for Ready node,
takes maybe 30 seconds
k3s kubectl get node
```



Standard Disclaimer

Running random things off the internet is inherently risky… do so at your own risk


k3s Server
Process
SQLite
API Server
Tunnel Proxy
Scheduler
Controller Manager
k3s Agent
Process
Tunnel Proxy
Kube Proxy
Flannel
Kubelet
containerd
Pod
Pod
Pod



Kubernetes

v1.19.3

☑ Enable Kubernetes

Start a Kubernetes single-node cluster when starting Docker Desktop.

□ Deploy Docker Stacks to Kubernetes by default

Make Kubernetes the default orchestrator for "docker stack" commands (changes "~/.docker/config.json")

□ Show system containers (advanced)

Show Kubernetes internal containers when using Docker commands.

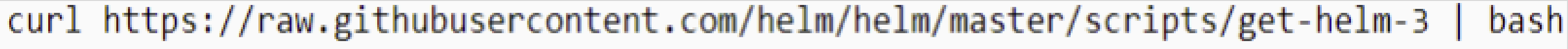
Reset Kubernetes Cluster

All stacks and Kubernetes resources will be deleted.

I know this is windows… but it is using wsl2

Helm

- Tool for managing k8s packages called charts
- Concepts:
  - Chart
  - Config
  - Release
- You can:
  - Create new charts from scratch
  - Package charts into chart archives (tgz files)
  - Interact with chart repos
  - Install/remove charts into k8s cluster
  - Manage release cycles of charts



```
curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash
```