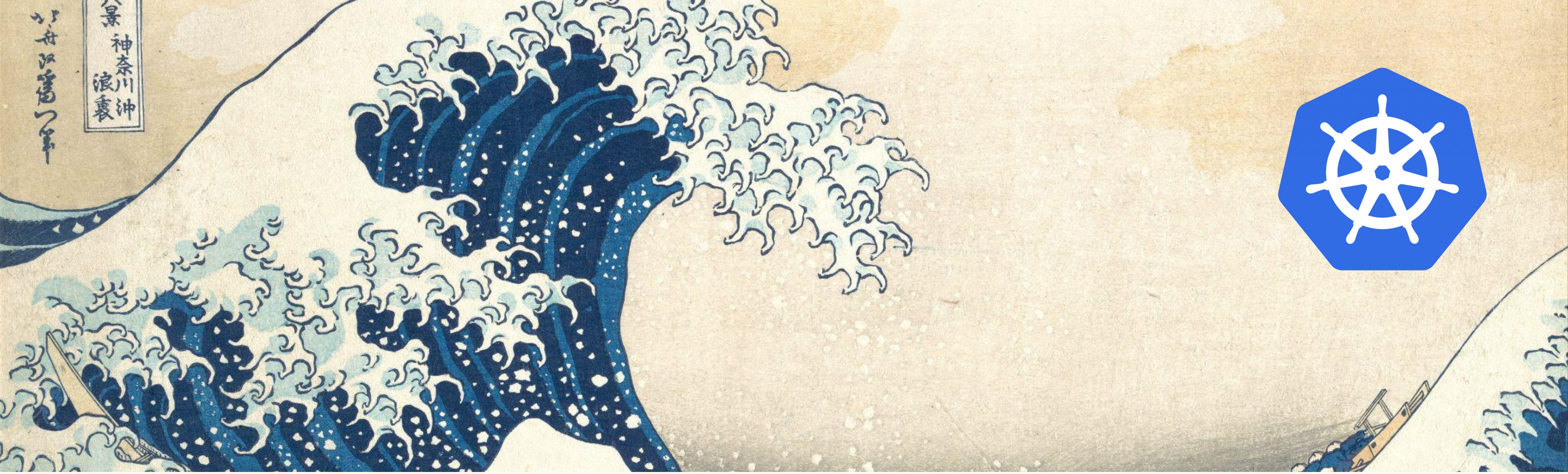


What *is* a Kubernetes?

Nikhil Jha | Spring 2024
The Open Computing Facility at UC Berkeley

Why is Kubernetes?

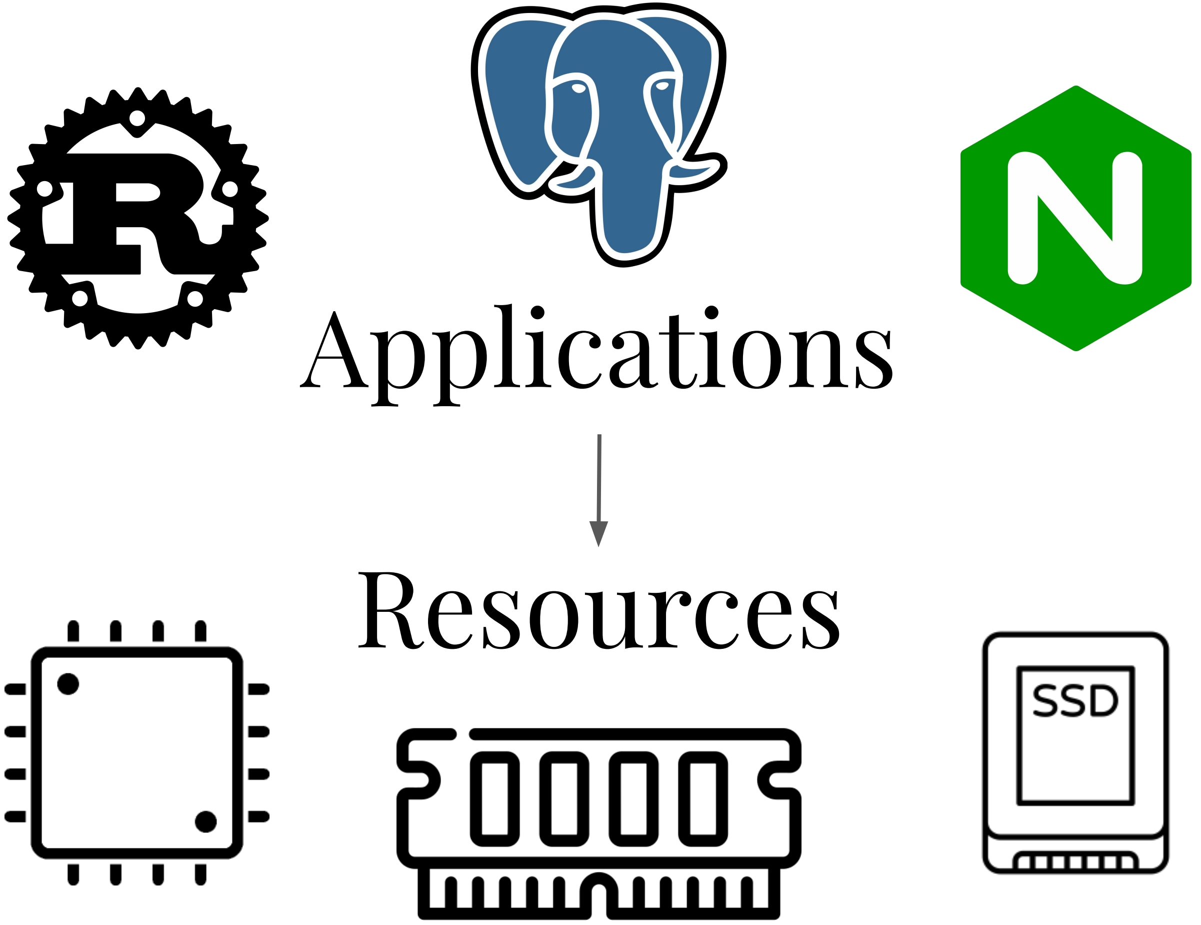
1

A Fundamental Problem in Computer Science:

How do we *write* software?

A Fundamental Problem in Computer Infrastructure:

How do we *run* software?



Non-logo images: noun project

Problem:

What is the best* way to organize or spend our resources to run this software?

Kubernetes Subproblem:

Given we have a bunch of computers that are already running, how do we best* organize them to run our software?

“best” = ?

“Efficiently”, “at scale”, minimal downtime, in a way that adapts to your org structure (layer 8), etcetc…

“best” = ?

Means different things to different people!

What is Kubernetes?

2

1. A Database

image: https://pixabay.com/vectors/database-storage-data-storage-152091/

1. A Database

- Distributed
- Key / Value
- Typed

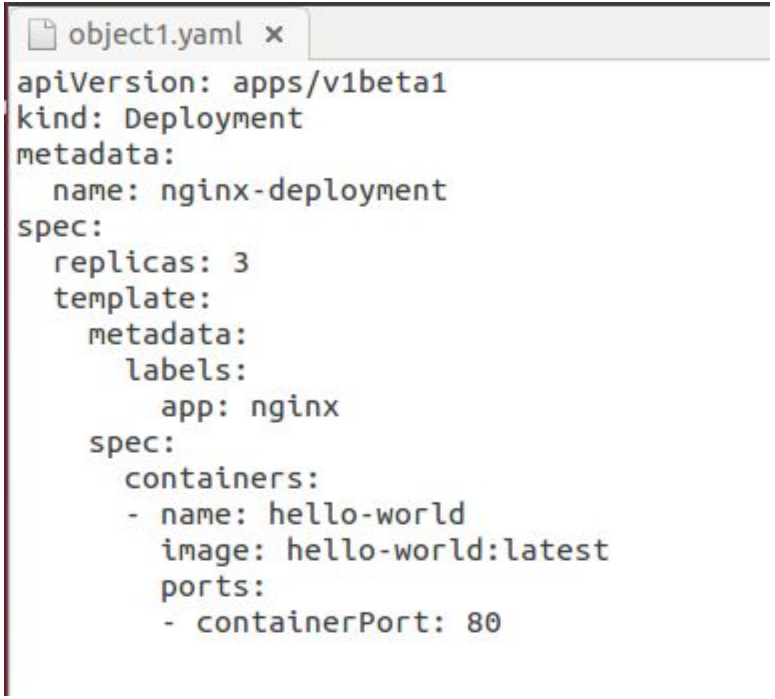
Examples of Objects

- "Run 5 replicas of X software on unique machines."
- "Make X software available at web address hello.example.com."
- "Make sure each copy of X has 16 GiB storage."

Crazier Objects

- “Create a Postgres database + account for X.”
- “Run a Minecraft server.”
- “Make me a coffee at 8:45 AM every morning.”

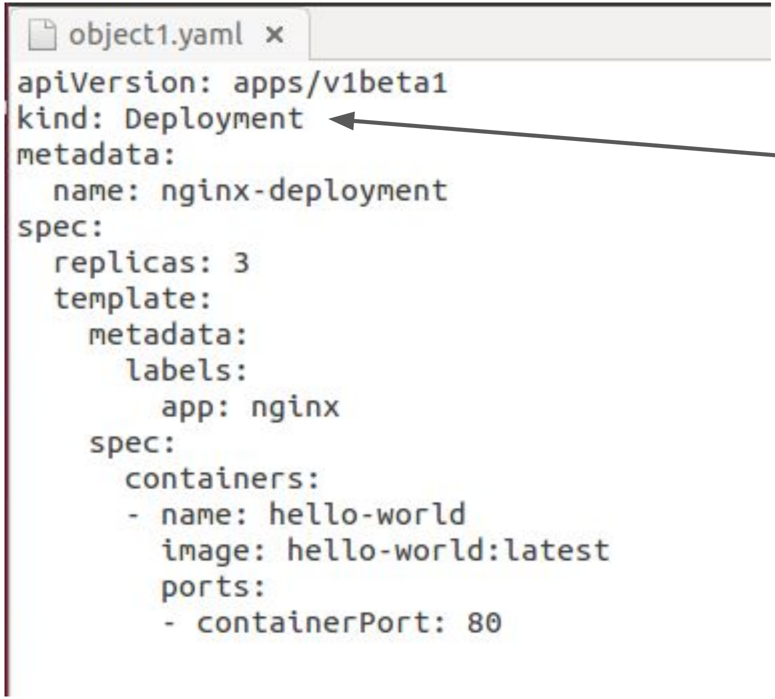
An example of something in the database…



```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: hello-world
        image: hello-world:latest
        ports:
        - containerPort: 80
```

image: https://stackoverflow.com/questions/46640049

An example of something in the database…

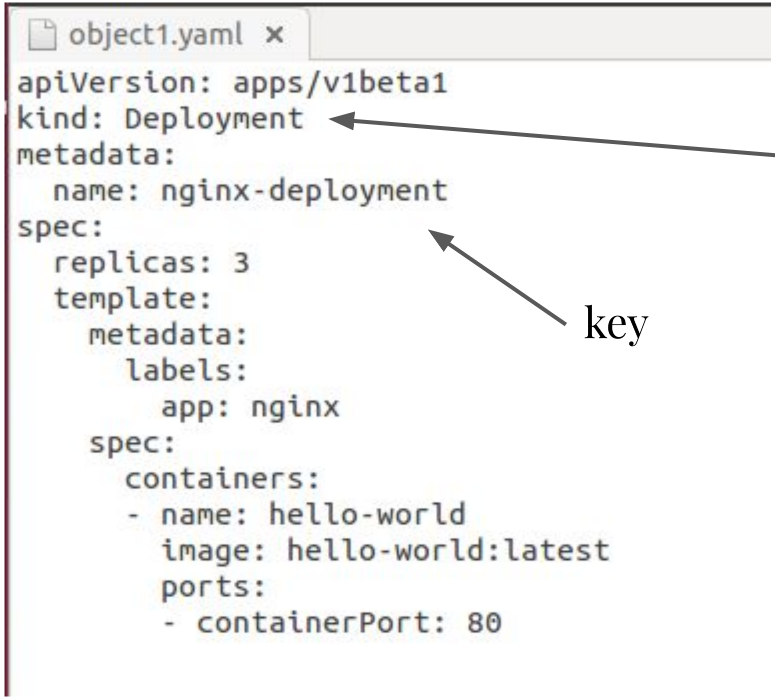


```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: hello-world
        image: hello-world:latest
        ports:
        - containerPort: 80
```

typed

image: https://stackoverflow.com/questions/46640049

An example of something in the database…



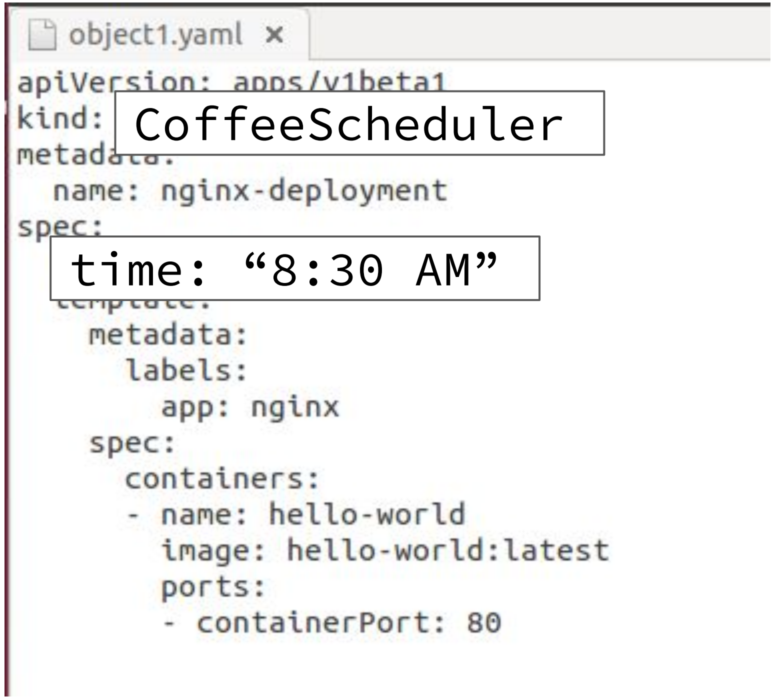
```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: hello-world
        image: hello-world:latest
        ports:
        - containerPort: 80
```

typed

key

image: https://stackoverflow.com/questions/46640049

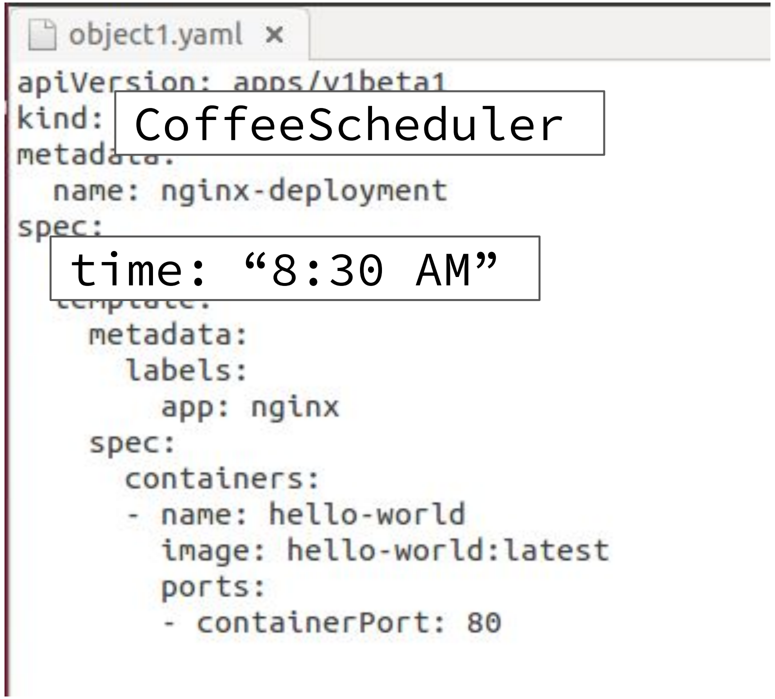
Not in Kubernetes by default, but you can make this!



```
apiVersion: apps/v1beta1
kind: CoffeeScheduler
metadata:
  name: nginx-deployment
spec:
  time: “8:30 AM”
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: hello-world
        image: hello-world:latest
        ports:
        - containerPort: 80
```

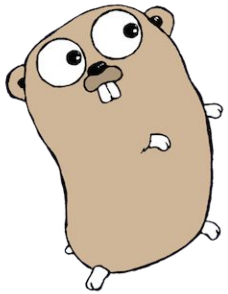
image: https://stackoverflow.com/questions/46640049

Not in Kubernetes by default, but you can make this!

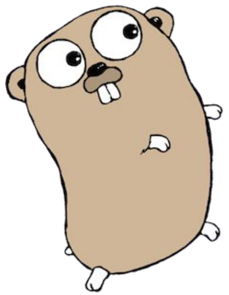


```yaml
apiVersion: apps/v1beta1
kind: CoffeeScheduler
metadata:
  name: nginx-deployment
spec:
  time: “8:30 AM”
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: hello-world
        image: hello-world:latest
        ports:
        - containerPort: 80
```

Java logo lol



2. Controller Software



2. Controller Software

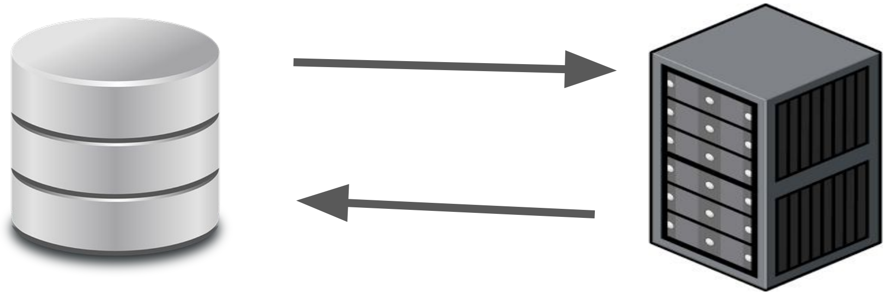


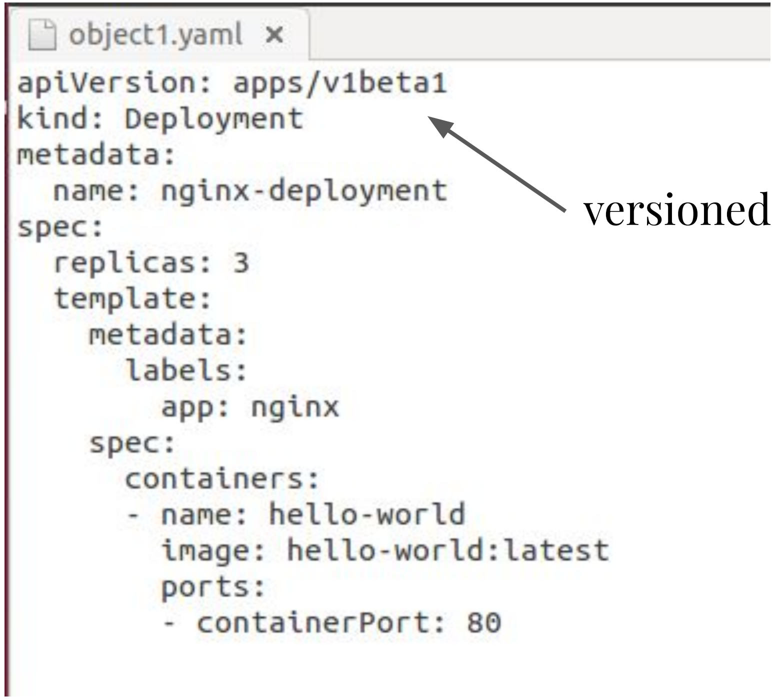
image: https://publicdomainvectors.org/en/free-clipart/Isometric-server-cabinet-vector-graphics/13444.html

3. Standardized APIs

3. Standardized APIs

- Versioned
- Stable
- Universal

An example of something in the database…



```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: hello-world
        image: hello-world:latest
        ports:
        - containerPort: 80
```

versioned

image: https://stackoverflow.com/questions/46640049

3. Standardized APIs

- Versioned
- Stable
- Universal

The other details…

3

=

“a container orchestration system”

=

“a container orchestration system”

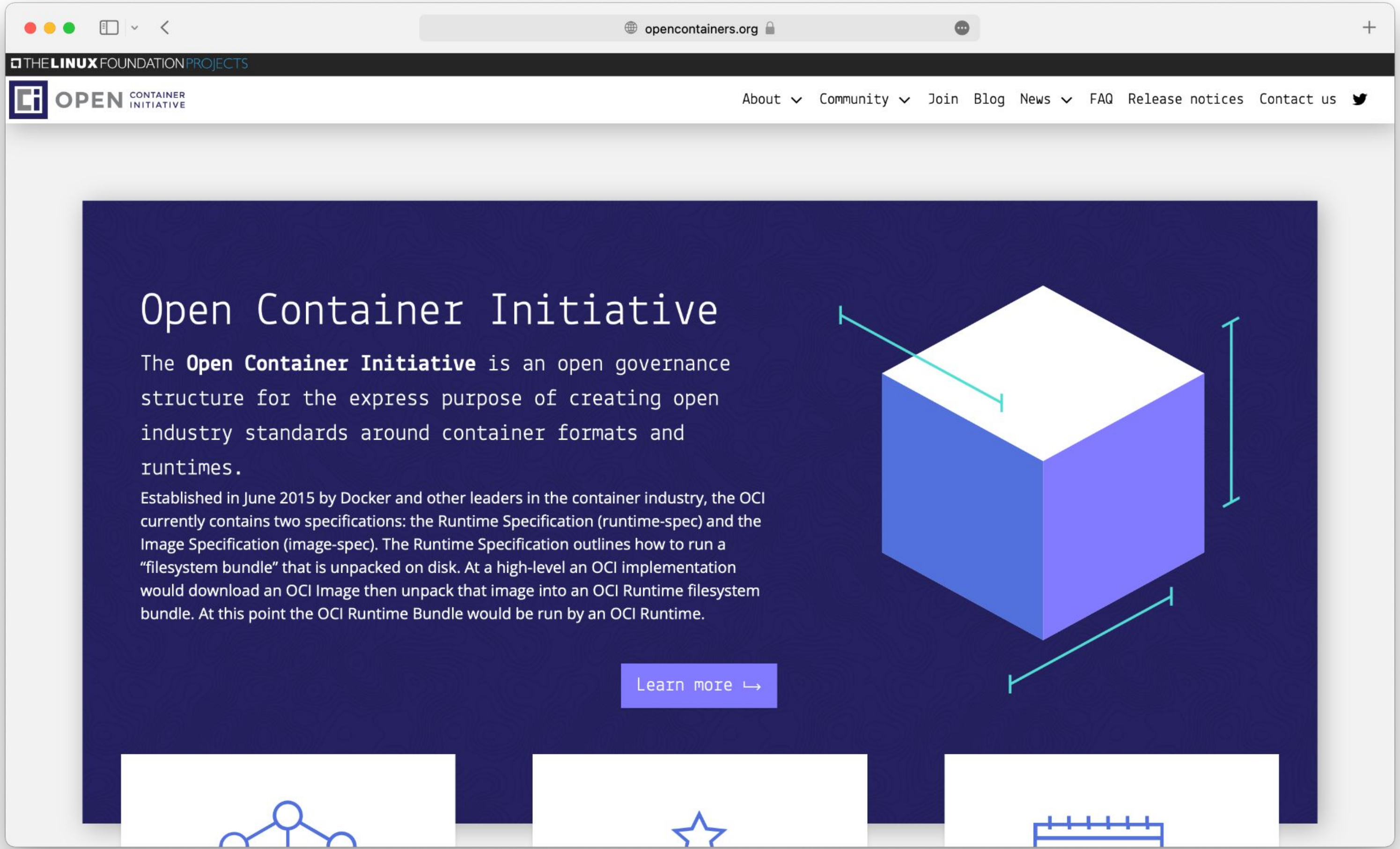
yet I haven’t even talked about containers lol… but they’re important!

containers =

“Why now and not 30 years ago?”

$$\text{a } \overbrace{\text{packaged}}^{\text{container}} \text{ app}$$

standard (OCI) $\overbrace{\text{sandboxed}}^{\text{container}}$ process



Open Container Initiative

The **Open Container Initiative** is an open governance structure for the express purpose of creating open industry standards around container formats and runtimes.

Established in June 2015 by Docker and other leaders in the container industry, the OCI currently contains two specifications: the Runtime Specification (runtime-spec) and the Image Specification (image-spec). The Runtime Specification outlines how to run a “filesystem bundle” that is unpacked on disk. At a high-level an OCI implementation would download an OCI Image then unpack that image into an OCI Runtime filesystem bundle. At this point the OCI Runtime Bundle would be run by an OCI Runtime.

container

standard (OCI) sandboxed process

typically implemented as set of isolated processes

container

standard (OCI) sandboxed process

typically implemented as set of isolated processes

namespaces (e.x. netns), cgroup, fs

Why is this “better”?

1. works on multiple machines

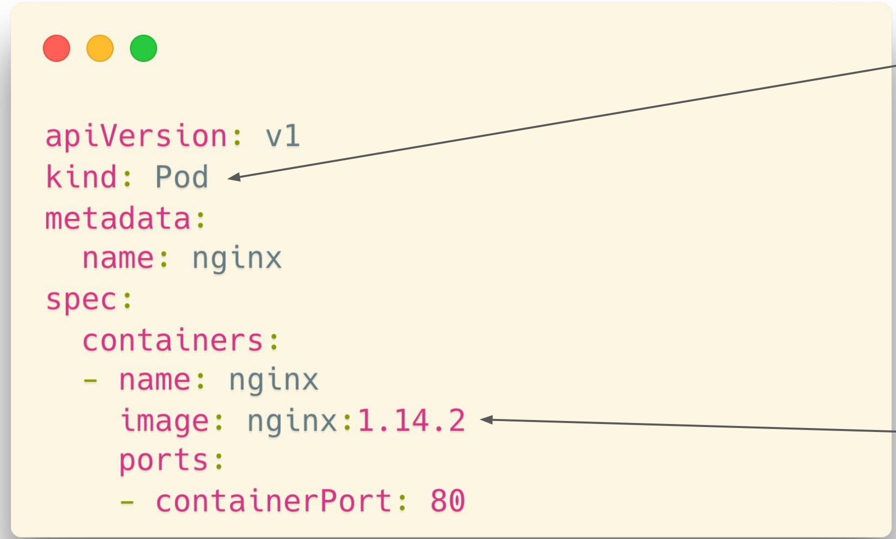
2. declarative infrastructure

SQL : C

Kubernetes : Your OS

Appendix: An Example

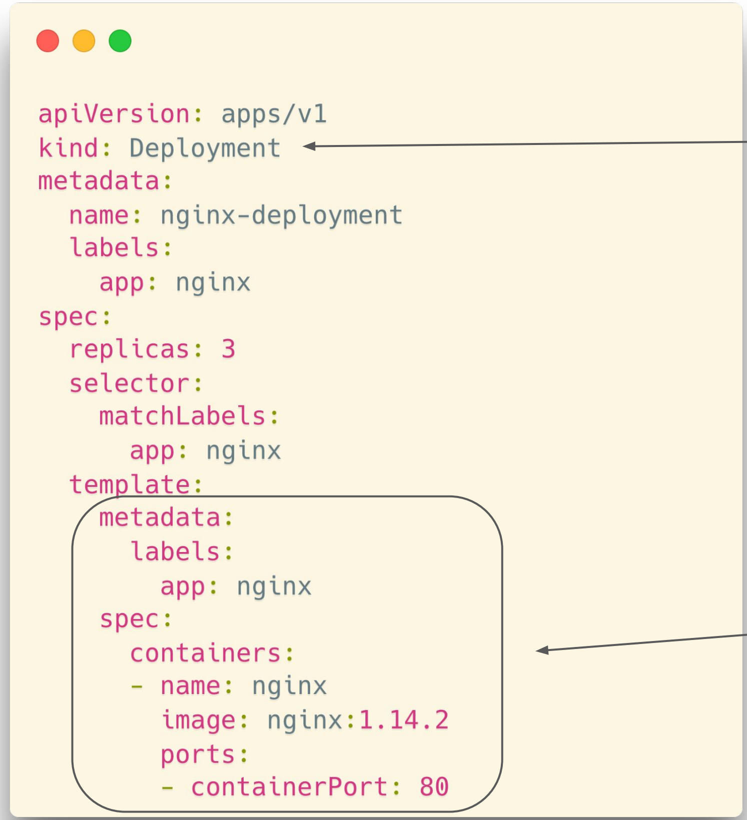
5



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

“Pod” represents the existence of a container

It contains information about the image to run, and the container port

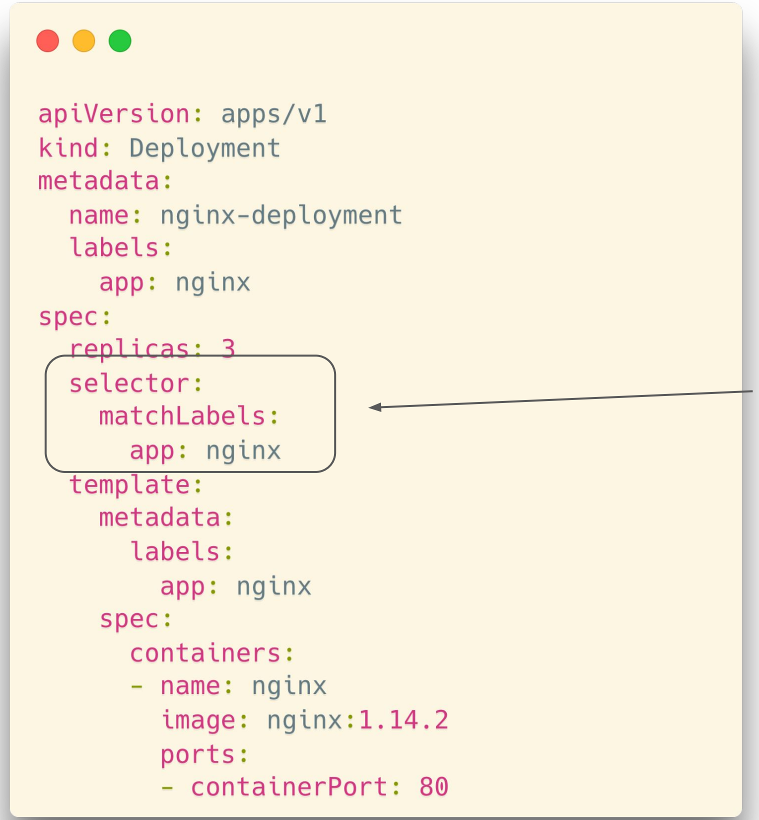


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

"Deployment" represents a set of fungible containers

notice that it contains a "template" for what Pod it should create

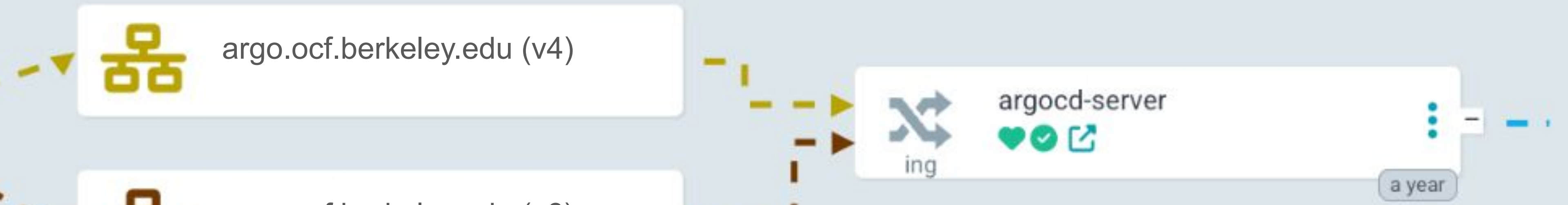
* Note: ReplicaSet is a thing that exists, so I may slightly lie when presenting this slide.



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

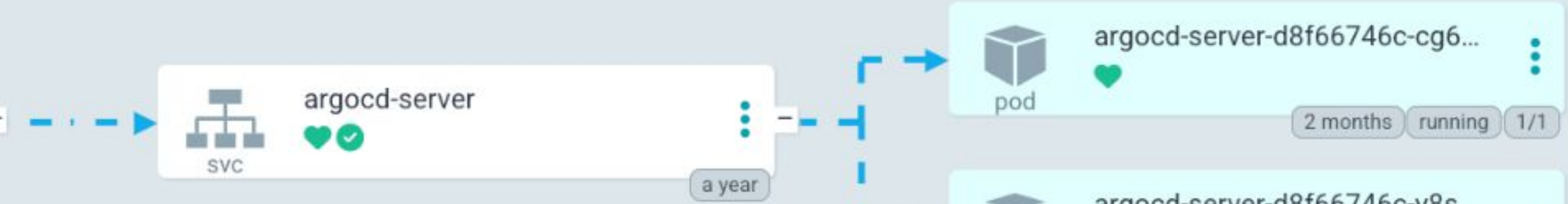
Label, so the controller software for the Deployment can tell which Pod objects it owns

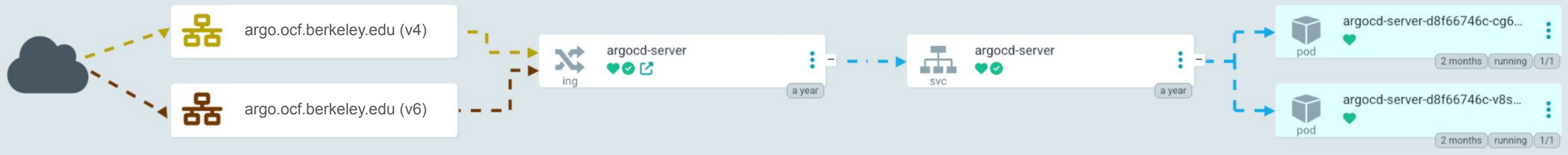
* Note: ReplicaSet is a thing that exists, so I may slightly lie when presenting this slide.

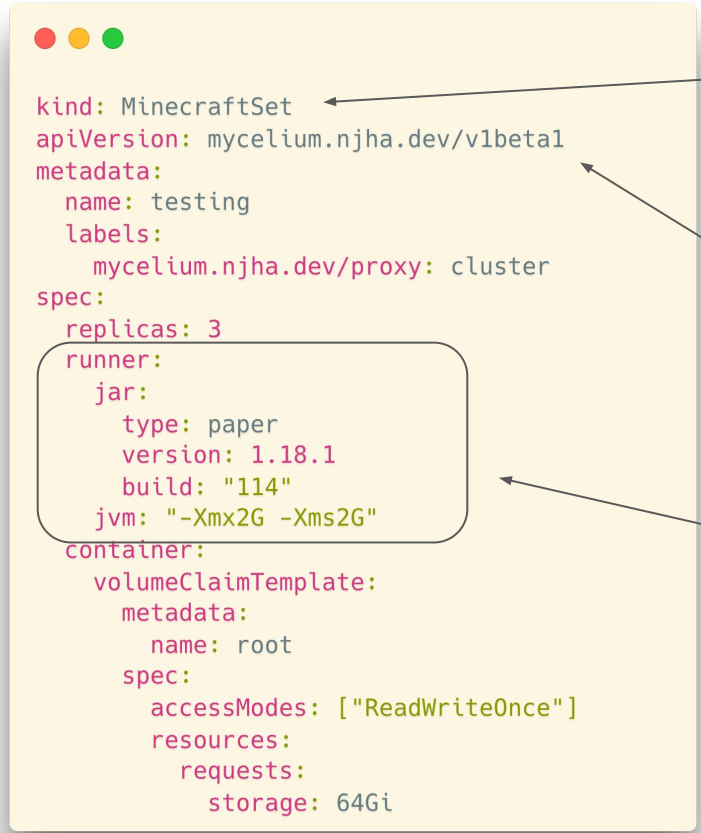


argo.ocf.berkeley.edu (v4)

argo.ocf.berkeley.edu (v6)


argocd-server
svc
a year
argocd-server-d8f66746c-cg6...
pod
2 months
running
1/1
argocd-server-d8f66746c-v8s...
pod
2 months
running
1/1


argo.ocf.berkeley.edu (v4)
argo.ocf.berkeley.edu (v6)
argocd-server
ing
a year
argocd-server
svc
a year
argocd-server-d8f66746c-cg6...
pod
2 months
running
1/1
argocd-server-d8f66746c-v8s...
pod
2 months
running
1/1

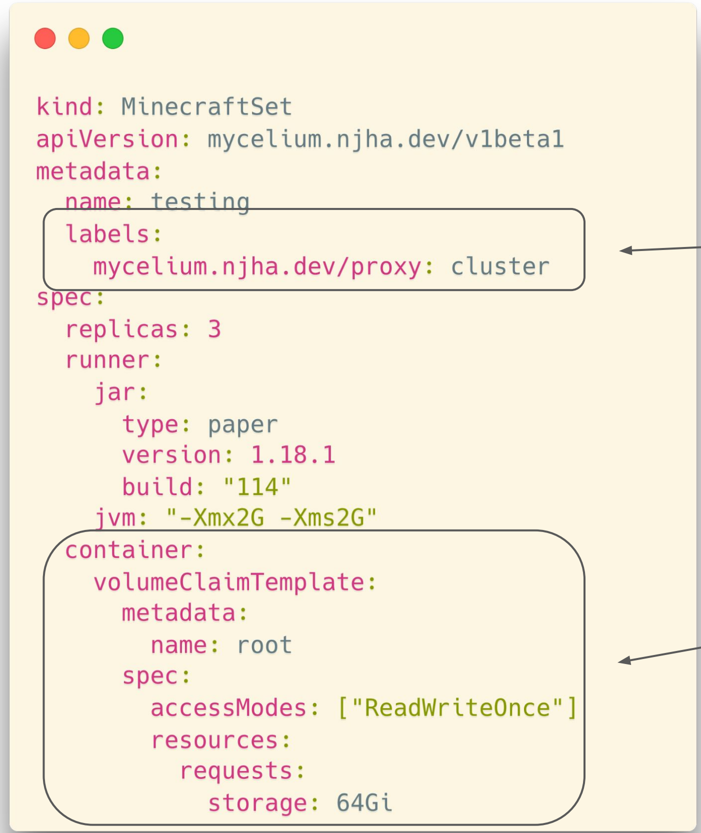


```yaml
kind: MinecraftSet
apiVersion: mycelium.njha.dev/v1beta1
metadata:
  name: testing
  labels:
    mycelium.njha.dev/proxy: cluster
spec:
  replicas: 3
  runner:
    jar:
      type: paper
      version: 1.18.1
      build: "114"
    jvm: "-Xmx2G -Xms2G"
  container:
    volumeClaimTemplate:
      metadata:
        name: root
      spec:
        accessModes: ["ReadWriteOnce"]
        resources:
          requests:
            storage: 64Gi
```

"MinecraftSet" represents the existence of a group of Minecraft servers

It's managed by controller software I wrote, hence njha.dev

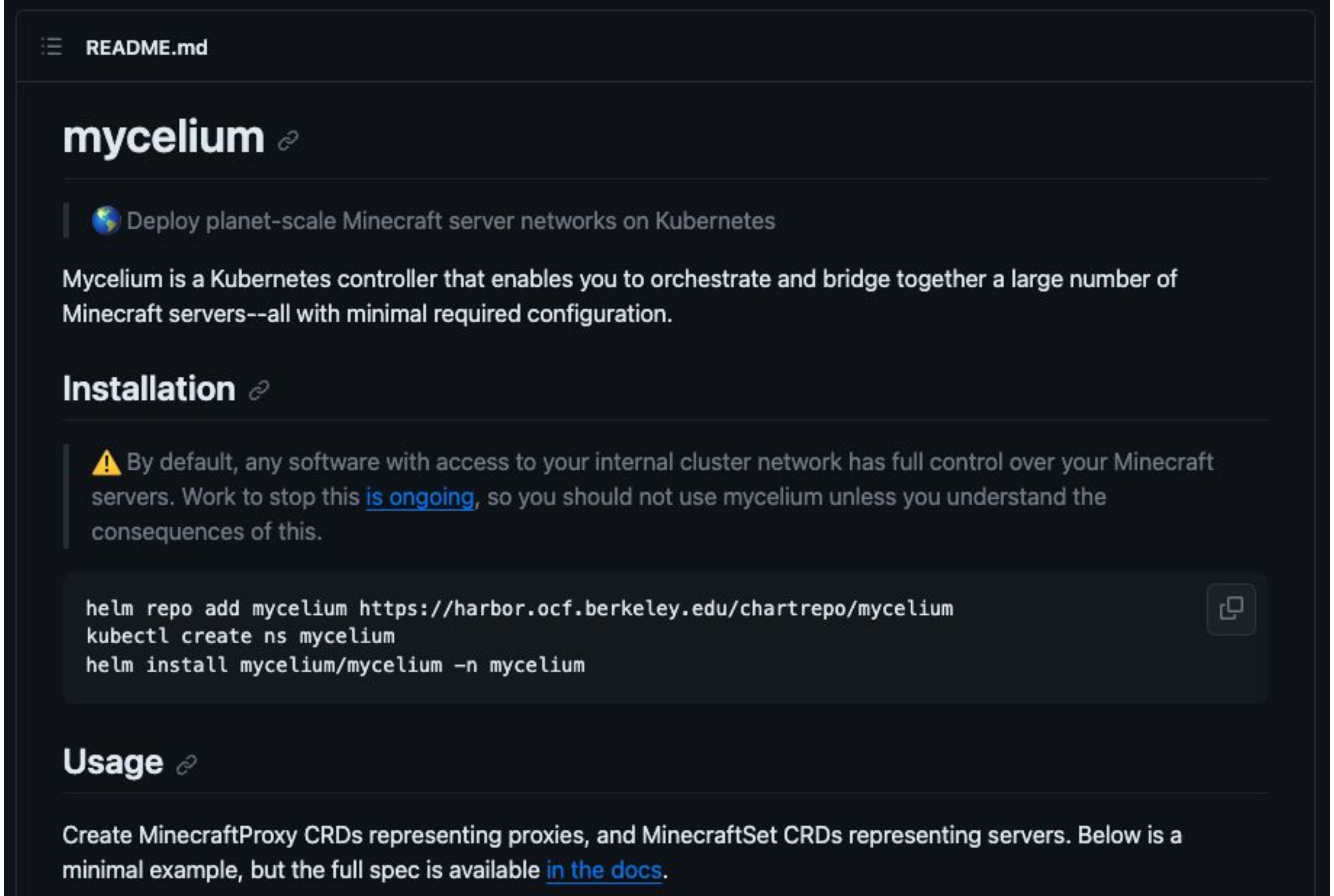
The configuration options are specific to a Minecraft server (!!)



```yaml
kind: MinecraftSet
apiVersion: mycelium.njha.dev/v1beta1
metadata:
  name: testing
  labels:
    mycelium.njha.dev/proxy: cluster
spec:
  replicas: 3
  runner:
    jar:
      type: paper
      version: 1.18.1
      build: "114"
    jvm: "-Xmx2G -Xms2G"
  container:
    volumeClaimTemplate:
      metadata:
        name: root
      spec:
        accessModes: ["ReadWriteOnce"]
        resources:
          requests:
            storage: 64Gi
```

Labels are used to organize resources

You can still override the template options for the Pod



README.md

mycelium

> 🌎 Deploy planet-scale Minecraft server networks on Kubernetes

Mycelium is a Kubernetes controller that enables you to orchestrate and bridge together a large number of Minecraft servers--all with minimal required configuration.

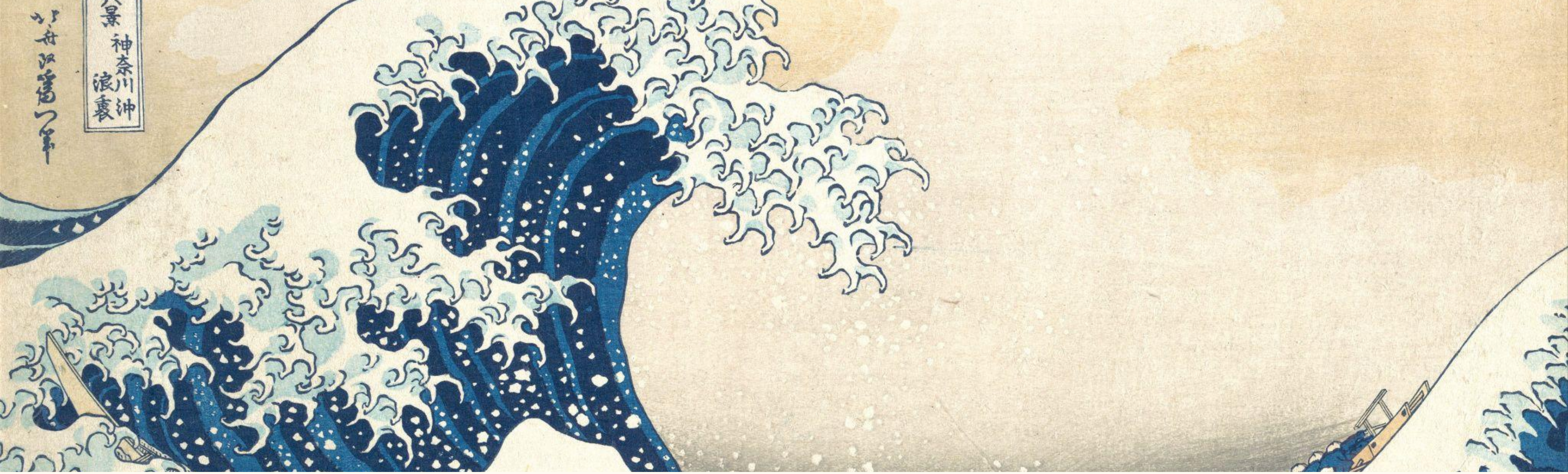
Installation

> ⚠️ By default, any software with access to your internal cluster network has full control over your Minecraft servers. Work to stop this is ongoing, so you should not use mycelium unless you understand the consequences of this.

```
helm repo add mycelium https://harbor.ocf.berkeley.edu/chartrepo/mycelium
kubectl create ns mycelium
helm install mycelium/mycelium -n mycelium
```

Usage

Create MinecraftProxy CRDs representing proxies, and MinecraftSet CRDs representing servers. Below is a minimal example, but the full spec is available in the docs.



process::exit(0); // ty