l-m.dev / talk

>> Destroying My Homelab With Kubernetes

Friday, July 17th | Colombo Theatre B | 2 PM - 4 PM

2. About me

- Liam, l-m, github.com/l1mey112 (GH)
- https://l-m.dev/, (email) l-m at l-m.dev

- 2nd Year at **UNSW**, Adv Math/CS
- Exec at **Linux Society** and runs **Live @ LNSC**.

Live @ LNSC

3. This Talk?

This wouldn’t be that long I think

// TODO: (1) Kubernetes

// TODO: (2) Self hosting and my setup

>> (1) Kubernetes

5. (1.H) Kubernetes

Kubernetes (K8s) is a container orchestrator.

Released by Google in 2014, a rewrite of their internal cluster manager, *Borg*.

> Kubernetes, also known as K8s, is an open source system for automating deployment, scaling, and management of containerized applications.
>
> — kubernetes.io

https://github.com/kubernetes/kubernetes

6. Kubernetes?

kubernetes = container orchestrator over lots of servers (called nodes)

- orchestration = wrangle your hundreds of servers to deploy your infrastructure
  (run N web servers, expose this port, connect two containers)

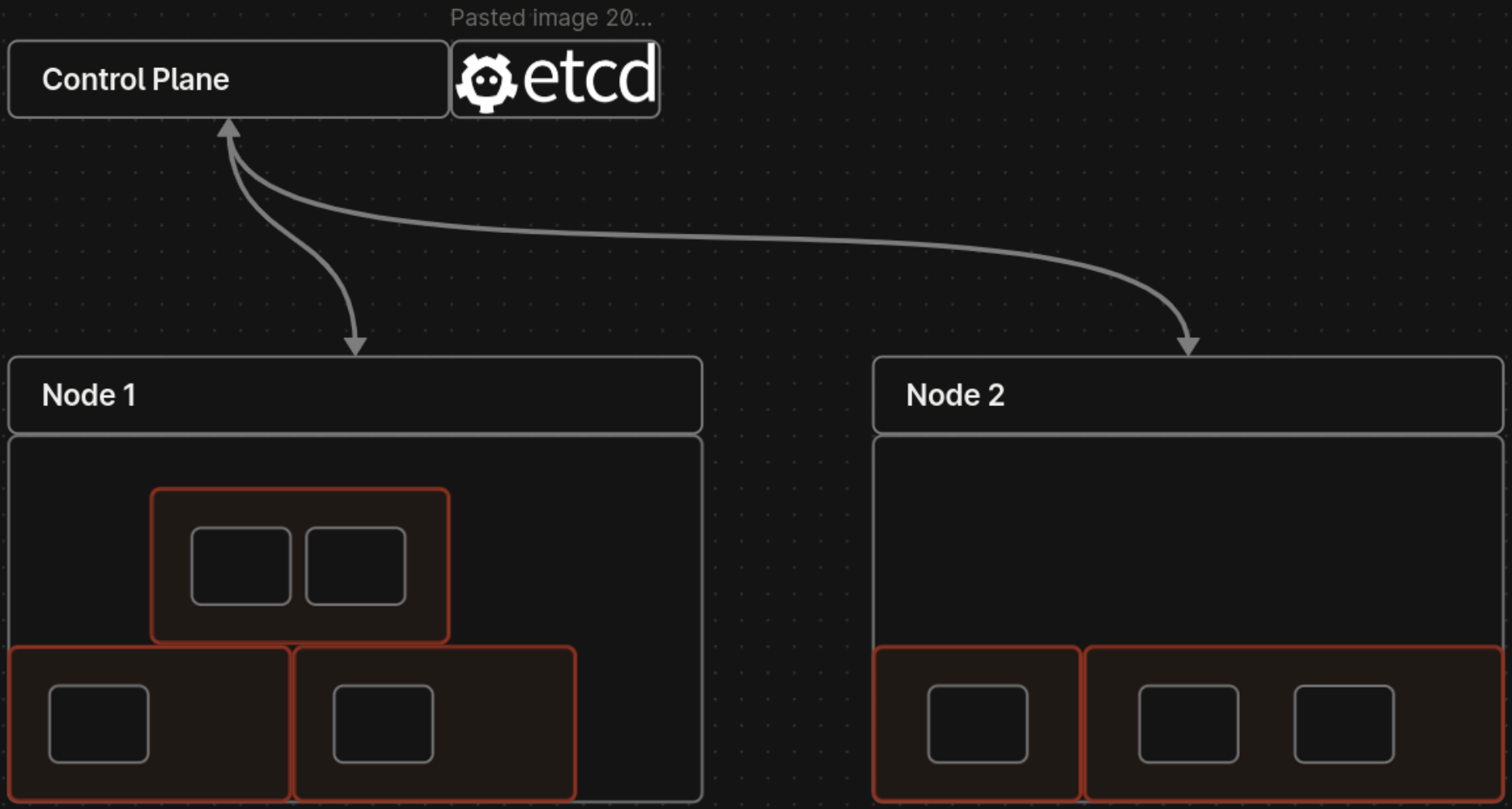
- declarative = the desired state of your infra is written down and deployed
  (describe the entirety of your infrastructure in 1000 yaml files)

7. Simple architecture

1. You package the things you want to run into **containers**
2. Group these containers together into **pods**
   (collection of containers that live & die together)
3. Tell Kubernetes to schedule these pods onto **nodes**
   (your physical servers)


```
ls *.yaml | xargs -n1 kubectl apply -f
```

9. Simple architecture

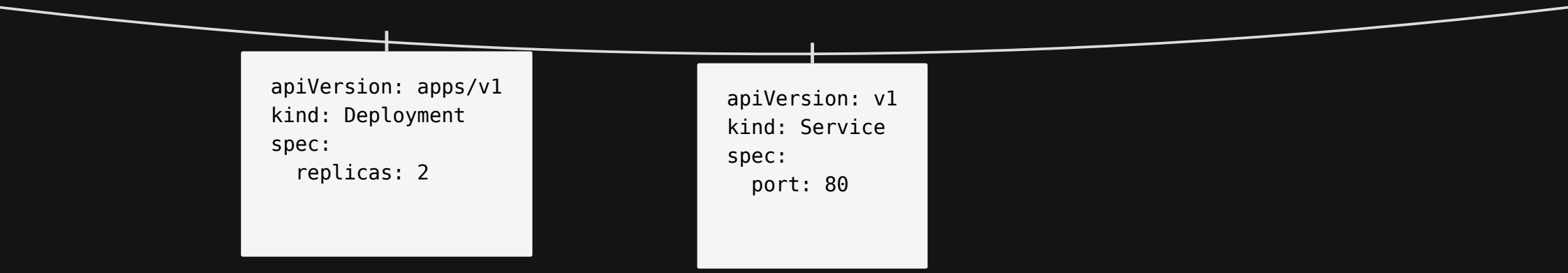


10. etcd

kubernetes (or, etcd) is a like clothesline, sticky notes, …

10. etcd

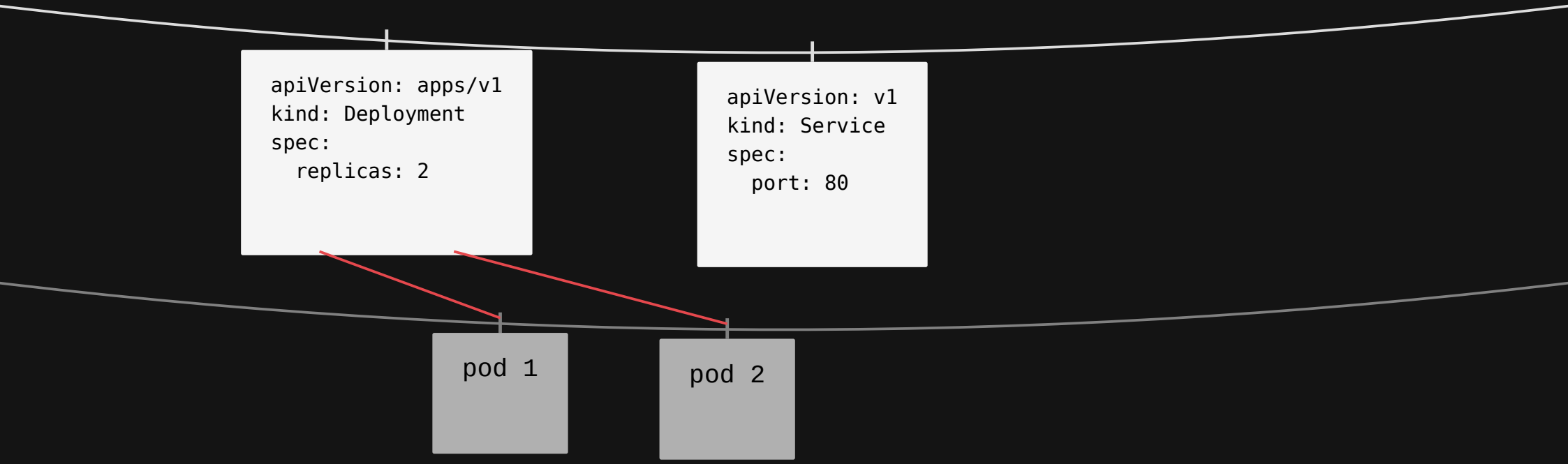
kubernetes (or, etcd) is a like clothesline, sticky notes, …



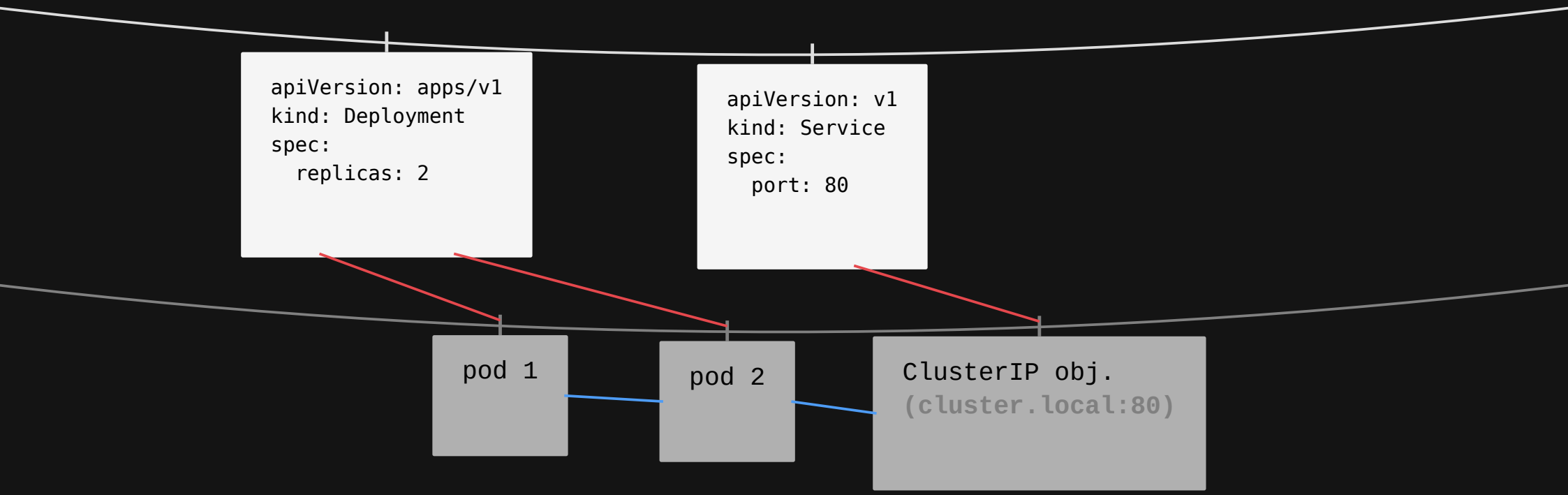
11. etcd


apiVersion: apps/v1
kind: Deployment
spec:
replicas: 2
apiVersion: v1
kind: Service
spec:
port: 80

11. etcd


apiVersion: apps/v1
kind: Deployment
spec:
replicas: 2
apiVersion: v1
kind: Service
spec:
port: 80
pod 1
pod 2

11. etcd


apiVersion: apps/v1
kind: Deployment
spec:
replicas: 2
apiVersion: v1
kind: Service
spec:
port: 80
pod 1
pod 2
ClusterIP obj.
(cluster.local:80)

12. Lets run through an example

- We want two replicas of an nginx pod hosted at port :80 (we are web scale)
- We want a unique cluster internal IP address for the pods.
- We want the ClusterIP (^^^) to be exposed under cube.l-m.dev.

---

1. `kind: Deployment` with the pod description written there
2. `kind: Service` pointing at the pods
3. `kind: Ingress` pointing at the service

Deployment (makes pods) -> Service (nginx-pods.cluster.local) -> Ingress (routes cube.l-m.dev)

13. Lets run through an example

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx-deployment
  namespace: nginx-test
spec:
  replicas: 2
  selector:
    matchLabels:
      component: deployment    # -to find
  template:                    #  the pods
    metadata:                  #  |
      labels:                  #  |
        component: deployment  # -/
    spec:                      # pod spec
      containers:
        - name: nginx
          image: nginx:alpine
          ports:
            - containerPort: 80
```

- the Deployment contains a "pod spec" defining a single pod, which contains N = 1 num. of containers

```yaml
apiVersion: v1
kind: Service
metadata:
  name: service
  namespace: nginx-test
spec:
  ports:
    - port: 80
      targetPort: 80
  selector:
    component: deployment # selects the
# <--------------------/ pods as before
```

14. Lets run through an example

```
l-m@debian ~> kubectl -n nginx-test get pods -o wide
NAME                          READY   STATUS    RESTARTS   AGE   IP             NODE
deployment-58b95f56fc-46hhp   1/1     Running   0          98s   10.244.1.220   kube-01
deployment-58b95f56fc-hxw9j   1/1     Running   0          9d    10.244.1.10    kube-01
l-m@debian ~> kubectl -n nginx-test get svc service -o jsonpath='{.spec.clusterIP}'
10.96.239.36⏎
l-m@debian ~>
```

URL: service.nginx-test.svc.cluster.local (k8s runs an in cluster DNS resolver)

15. Lets run through an example

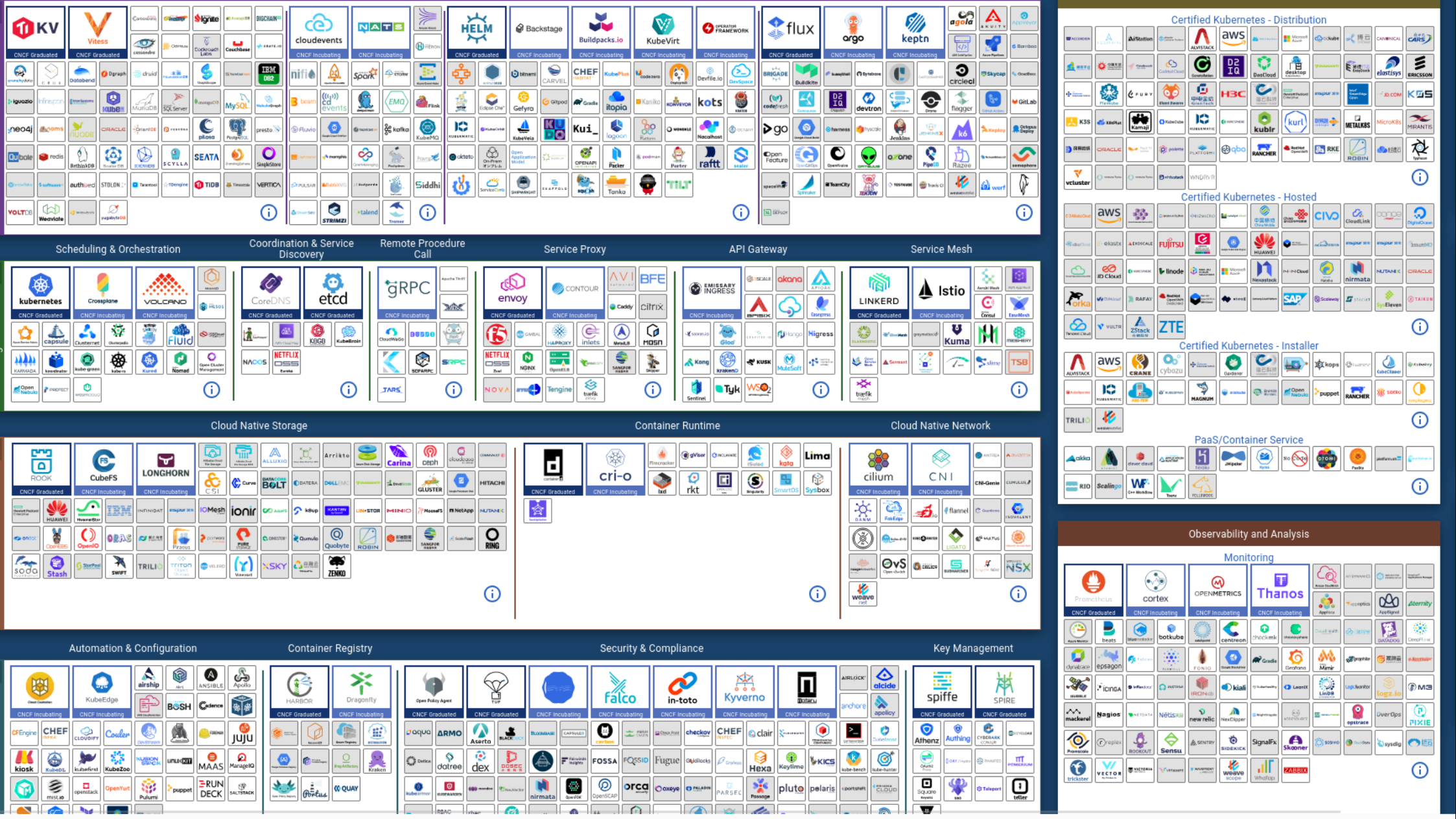
```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress
  namespace: nginx-test
spec:
  ingressClassName: traefik
  rules:
    - host: cube.l-m.dev
      http:
        paths:
          - backend:
              service:
                name: service
                port:
                  number: 80
            path: /
            pathType: Prefix
```

- the Ingress obj. needs an **Ingress Controller**

think: dynamic reverse proxy

- traefik
- nginx-ingress

personal anecdote: a concrete example was 1000x better than the "docs" out there


Scheduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Proxy
API Gateway
Service Mesh
Cloud Native Storage
Container Runtime
Cloud Native Network
Automation & Configuration
Container Registry
Security & Compliance
Key Management
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Certified Kubernetes - Installer
PaaS/Container Service
Observability and Analysis
Monitoring

thanks to the CNCF

everything is YAML

deployments > ! traefik.yaml

```
spec:
  template:
    spec:
      containers:
        - name: cloudflared-linuxunsw
          - /etc/cloudflared/config/config.yaml
          - run
          volumeMounts:
            - name: cloudflared-linuxunsw-config
              mountPath: /etc/cloudflared/config
              readOnly: true
            - name: cloudflared-linuxunsw-credentials
              mountPath: /etc/cloudflared/creds
              readOnly: true
          livenessProbe:
            httpGet: { path: /ready, port: 2001 }
            initialDelaySeconds: 10
            periodSeconds: 10
            failureThreshold: 1
          resources:
            requests:
              cpu: 50m
              memory: 64Mi
            limits:
              memory: 128Mi
      volumes:
        - name: cloudflared-config
          configMap:
            name: cloudflared-config
        - name: cloudflared-credentials
          secret:
            secretName: traefik-cf-tunnel-config
        - name: cloudflared-linuxunsw-config
          configMap:
            name: cloudflared-linuxunsw-config
        - name: cloudflared-linuxunsw-credentials
          secret:
            secretName: traefik-cf-tunnel-linuxunsw
---
```

deployments > ! website.yaml

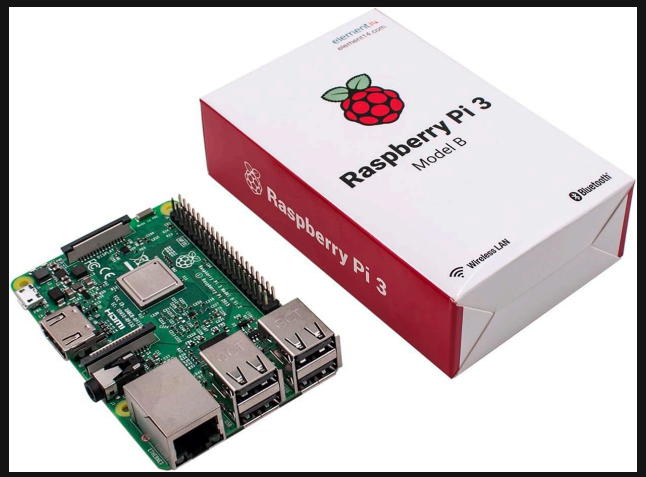
```
spec:
  template:
    spec:
      containers:
        - name: nginx
          volumeMounts:
            - { name: html, mountPath: /var/www/html }
            - { name: conf, mountPath: /etc/nginx/conf.d/default.con
          ports:
            - containerPort: 80
      volumes:
        - name: html
          persistentVolumeClaim: { claimName: public }
        - name: conf
          configMap: { name: nginx }
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: public
  namespace: website
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: democratic-csi
  resources:
    requests:
      storage: 1Gi
---
apiVersion: v1
kind: Service
metadata:
  name: service
  namespace: website
spec:
  ports:
    - port: 80
      targetPort: 80
  selector:
```

>> (2) How I’ve hosted for a while

19. Lay of the land

I’ve been selfhosting for a while now

Raspberry Pi 3



(near infant)

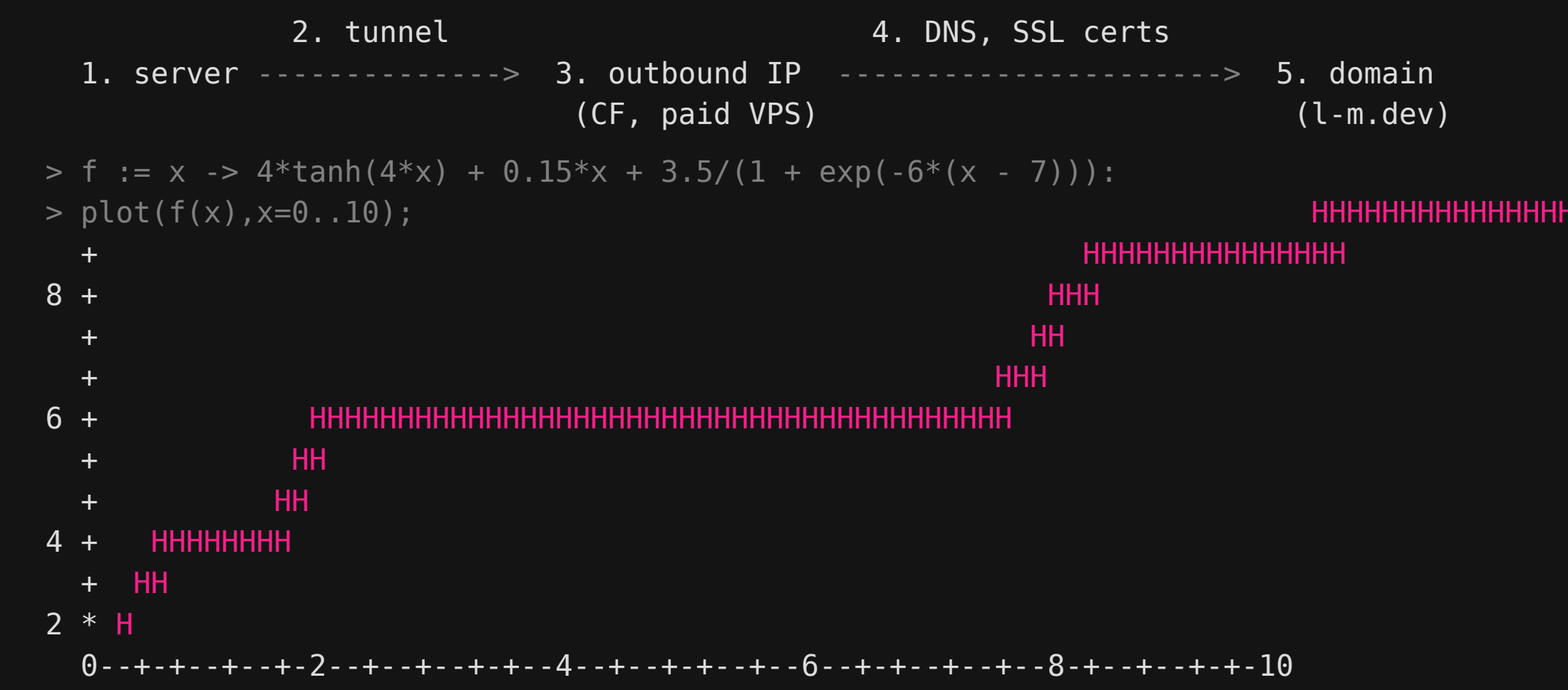
Intel Nuc

(most of highschool)

Rack mount (UCS C220 M3S)

(present day present time)

20. Learning this stuff



21. Two areas of concern (difficulty curve)

Catch-all rungs of getting shit running (1)

1. apt install package && systemctl enable --now package
2. docker run -d –restart=unless-stopped package
3. docker compose up -d
4. Dockge / Portainer
5. Proxmox or TrueNAS one click thing
6. assert(0 && "unreachable");

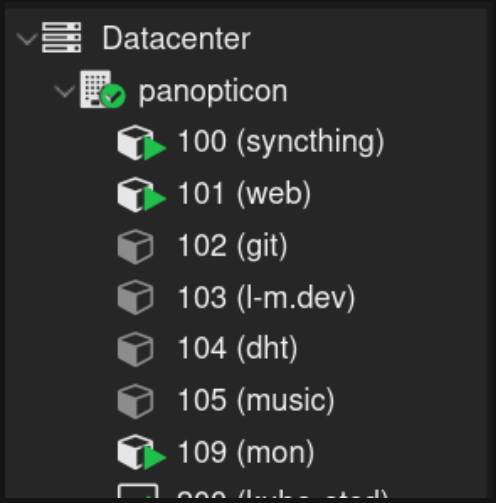
S**t to worry about to get it CONNECTED **(2)** ———>

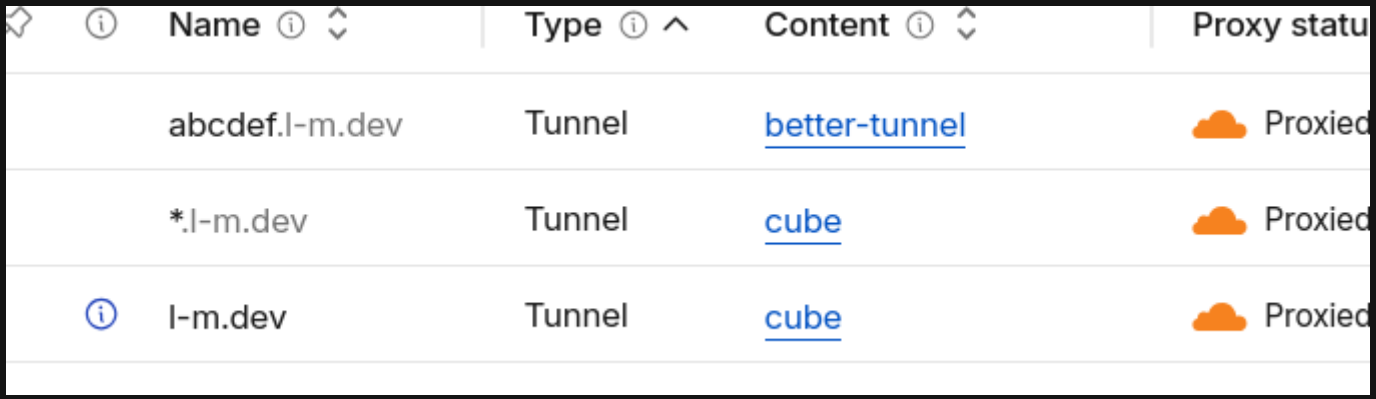
1. cloudflared
2. WireGuard
3. tailscale
4. ngrok
5. port forwarding
6. CGNAT
7. rathole
8. autossh -R
9. netbird
10. VPS
11. frp
12. ddclient
13. duckdns
14. iptables DNAT
15. fail2ban
16. Let's Encrypt
17. certbot
18. acme.sh
19. DNS-01 challenges

22. My homelab

For me right now, dead simple:

- Proxmox on a server with ZFS root
- LXC containers running Debian or Alpine
- Assign predictable IP addresses, ssh in and install my stuff
- Use cloudflare tunnels





this is a PITA

i hate installing stuff

setup

```
setup-alpine
```

```
adduser -g 'l-m' l-m
adduser l-m wheel
adduser l-m video
adduser l-m input
apk add doas
```

```
# vi /etc/doas.d/doas.conf
permit persist keepenv :wheel as root
```

Running stuff is a """solved""" problem

(even w/ containers it’s a lot of maintenence!)

25. You've deployed it, then what?

1. Reproducible (byte for byte equal, every deploy)
2.
3.

25. You've deployed it, then what?

1. **Reproducible** (byte for byte equal, every deploy)
2. **Declarative** (what you want done, not how to get there)
3.

25. You've deployed it, then what?

1. **Reproducible** (byte for byte equal, every deploy)
2. **Declarative** (what you want done, not how to get there)
3. **Reliable** (if it works once, it’ll probably work forever)

26. You've deployed it, then what?

1. **Reproducible** (byte for byte equal, every deploy)
2. **Declarative** (what you want done, not how to
3. **Reliable** (if it works once, it'll probabl

27. pros vs cons vs hacker

Pros

- Reproducible Declarative Reliable

  (sure bud)

HUGE PRO

- "i utilise cloud native technologies"
- kubernetes cloud ready certificate sponsored by microsoft azure
- resume gets an extra line

Cons

1. Reproducible, sure.

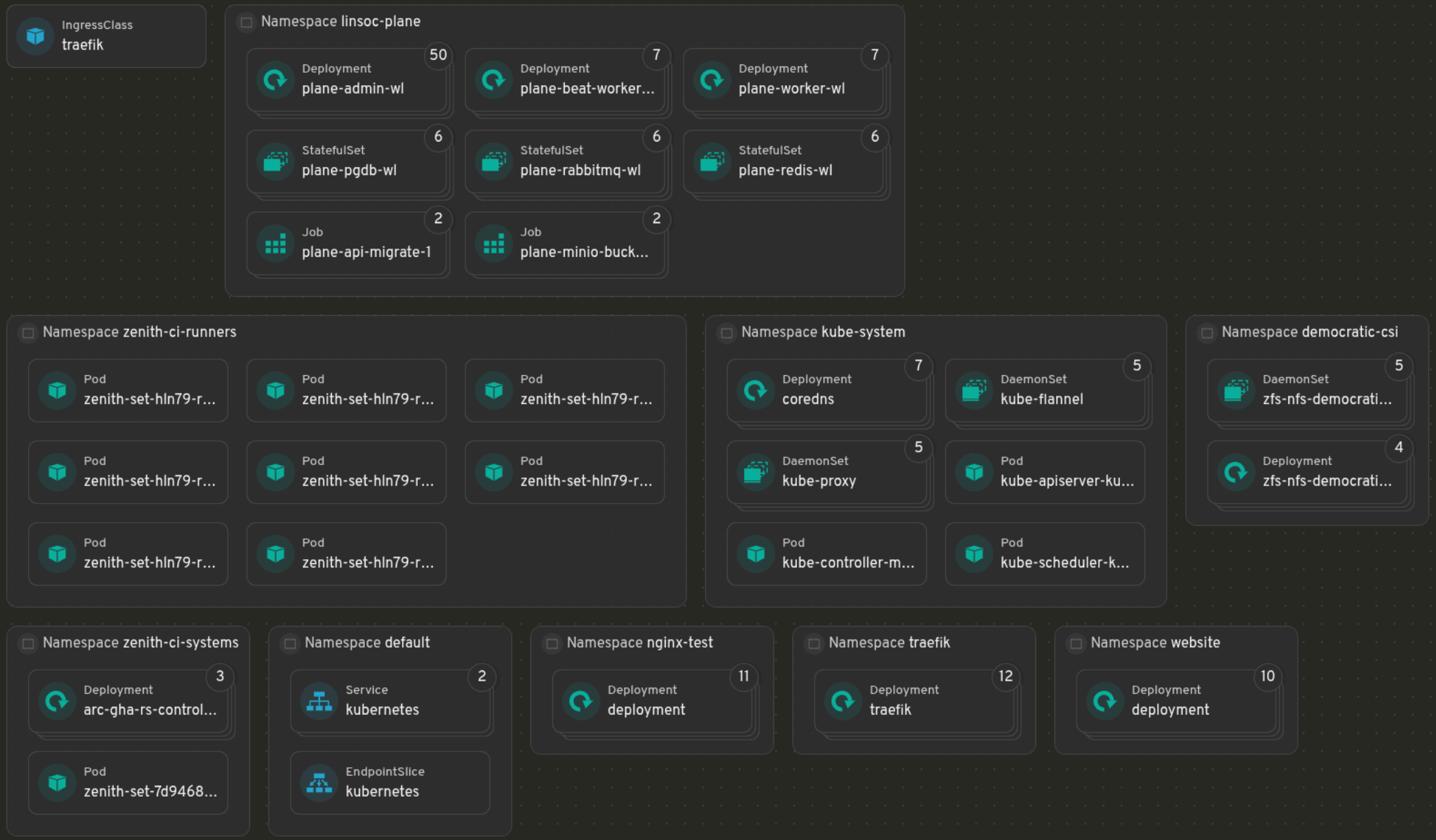
   Extreme amounts of trial and error to even Declare and have it work.

- Hundreds and hundreds of YAML files
- Helm is a disgusting abomination (pkg manager for k8s)

28. So?

I replaced my perfectly good homelab with a Kubernetes cluster and I run all my things on it (I track all the yamls in git)

- Website: see https://l-m.dev/
- These slides: https://l-m.dev/talk
- My monero node
- Syncthing
- Linux Society infrastructure
- Zenith Hosting CI
- My personal Forgejo instance


IngressClass
traefik
Namespace linsoc-plane
Deployment
plane-admin-wl
50
Deployment
plane-beat-worker...
7
Deployment
plane-worker-wl
7
StatefulSet
plane-pgdb-wl
6
StatefulSet
plane-rabbitmq-wl
6
StatefulSet
plane-redis-wl
6
Job
plane-api-migrate-1
2
Job
plane-minio-buck...
2
Namespace zenith-ci-runners
Pod
zenith-set-hln79-r...
Pod
zenith-set-hln79-r...
Pod
zenith-set-hln79-r...
Pod
zenith-set-hln79-r...
Pod
zenith-set-hln79-r...
Pod
zenith-set-hln79-r...
Pod
zenith-set-hln79-r...
Pod
zenith-set-hln79-r...
Namespace kube-system
Deployment
coredns
7
DaemonSet
kube-flannel
5
DaemonSet
kube-proxy
5
Pod
kube-apiserver-ku...
Pod
kube-controller-m...
Pod
kube-scheduler-k...
Namespace democratic-csi
DaemonSet
zfs-nfs-democrati...
5
Deployment
zfs-nfs-democrati...
4
Namespace zenith-ci-systems
Deployment
arc-gha-rs-control...
3
Pod
zenith-set-7d9468...
Namespace default
Service
kubernetes
2
EndpointSlice
kubernetes
Namespace nginx-test
Deployment
deployment
11
Namespace traefik
Deployment
traefik
12
Namespace website
Deployment
deployment
10

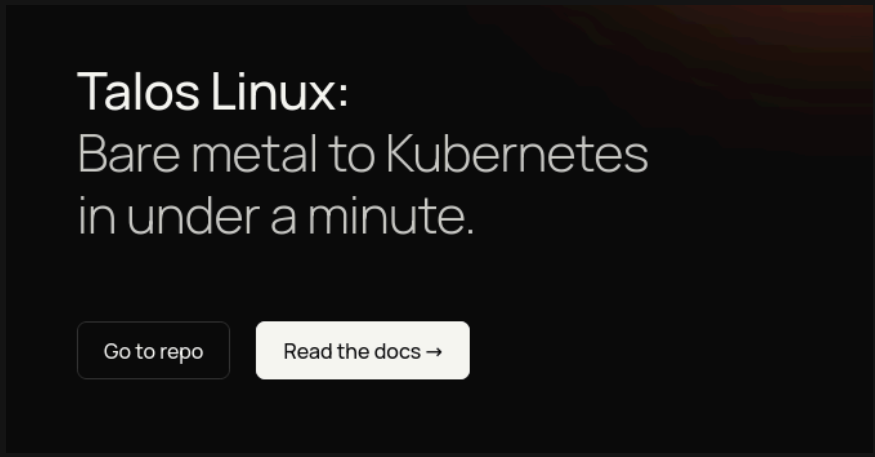
29. Personal development or?

- for me: I’ve really wanted to learn k8s for a long time
- for job: I do all the Kubernetes infrastructure at **Zenith Hosting**

30. Aside: Talos Linux

not enough time (im writing this 15 mins before LAL stars)

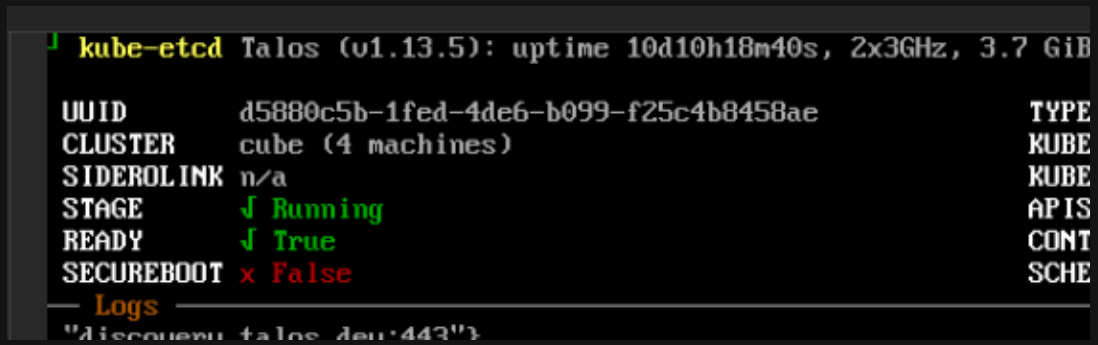
- Kubernetes deployments are described with yaml, but actually setting up the nodes (bare metal machines or VMs) is not
- Talos linux is like nixos for deploying fleets of kuberentes nodes
  - A Talos install has < 15 binaries, doesn't have ssh or a shell and configured with APIs only.



32. Aside: Talos Linux

```
# then node is completely secure after this
talosctl apply-config --insecure \
  --nodes 10.84.27.126 --file out/controlplane.yaml
```

and boom, immediate kubernetes node




```yaml
# network.yaml
machine:
  network:
    nameservers:
      - 10.84.27.1
    interfaces:
      - deviceSelector:
          physical: true
        addresses:
          - 10.84.27.120/24
        routes:
          - network: 0.0.0.0/0
            gateway: 10.84.27.1
---
apiVersion: v1alpha1
kind: HostnameConfig
hostname: kube-etcd
auto:
  $patch: delete
```

>> Finishing Up

32. Q&A - The End

Thank you everyone! It’s Q&A time.

come find me

- Liam Leadbetter
- https://l-m.dev/
- https://www.youtube.com/@l-mdotdev