Kubernetes

Tomáš Kukrál

Second Foundation Tech

linkedin.com/in/tomaskukral

Agenda

Kubernetes 101

- Container vs pod
- Immutable pods
- Networking
- Configuration files
- Exposing services
- Other workload

Demo: Babysitting the app

- Probes
- Resource management
- Crashes
- Malware and network security
- Hardware failures
- Scaling

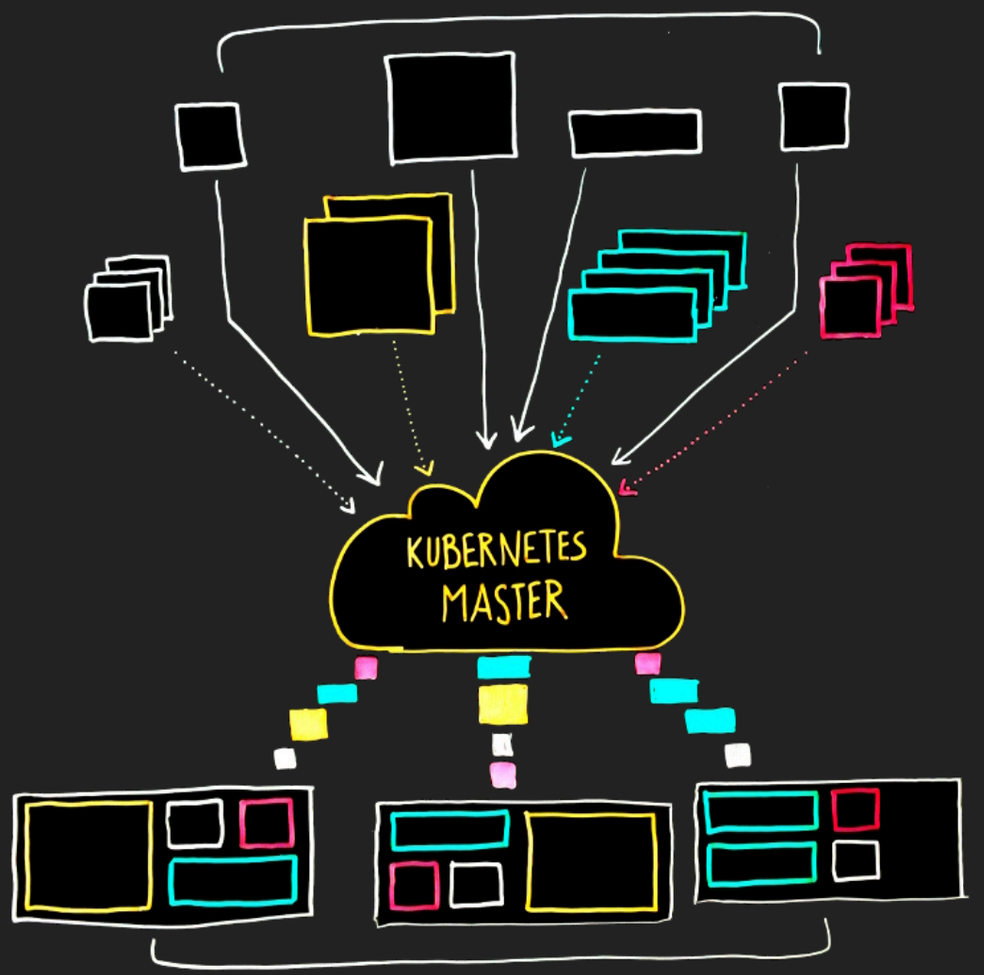
Orchestration

Allow moving from **host-centric** to **container-centric** infrastructure.

Orchestration is complex task of managing the container **lifecycle**.

- container parameters
- number of replicas
- (anti)affinities
- networking
- services
- resource allocation
- configuration management
- monitoring and healing
- load balancing
- updates
- daily tasks (backups)
- scaling - automated and manual
- network, storage

Kubernetes



Resources

- **Node** is machine running containers
- **Pod** is group of containers withing the sandbox
- **Deployment**, **ReplicaSet**

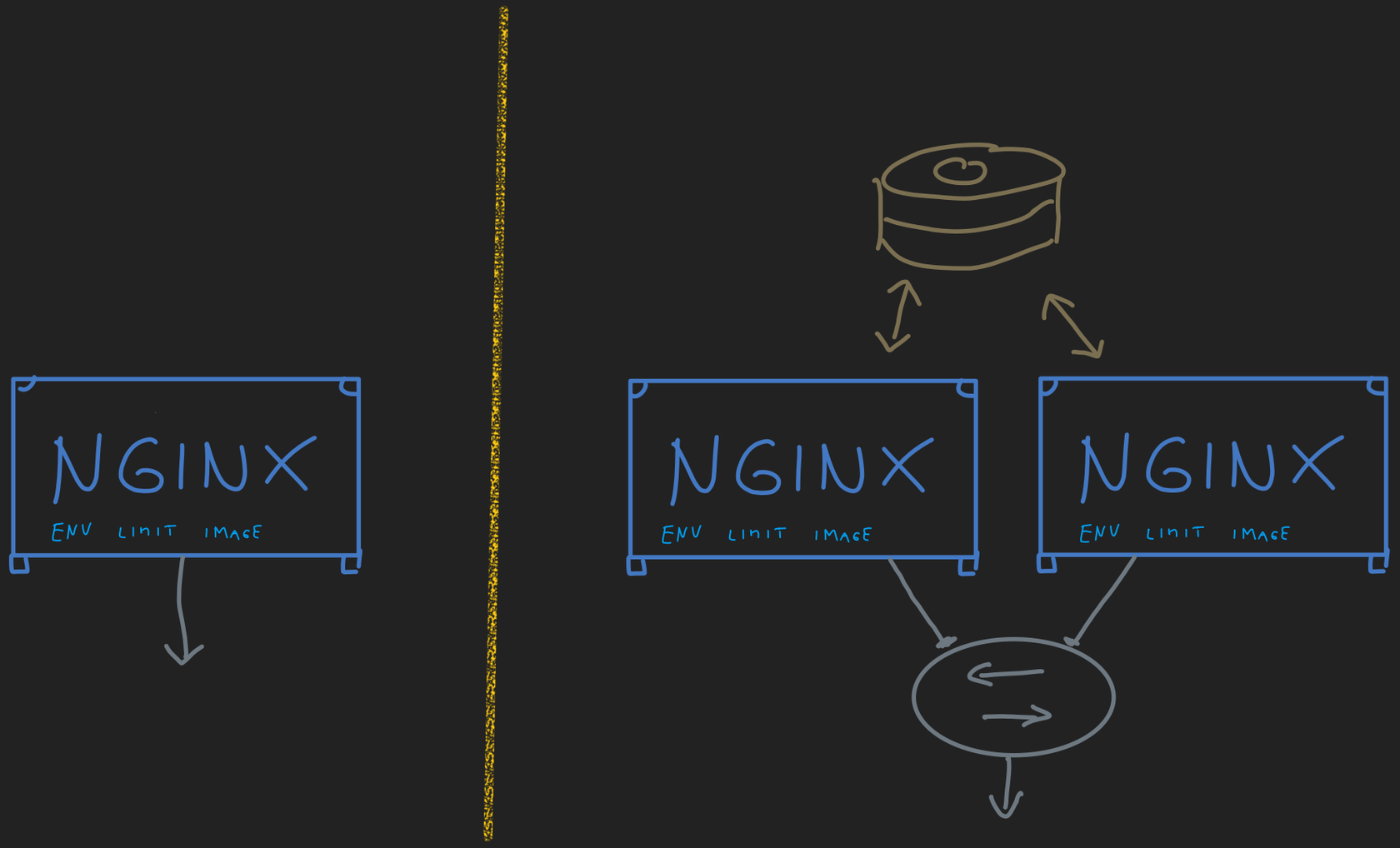
  Defines how containers (pods) should be running and how many of them should be active.
- **Service**

  Routes traffic from to containers and provide fixed endpoint and address.
- **Endpoint**

  Socket (IP and port) of available service, usually pod IP
- **ConfigMap**, **Secret**

  Provide standardized configuration and password management.

Container vs pod



Kubernetes operation

- API calls
- `kubectl` CLI, `OpenLens`, `client-go` library, `k9s`

```
kubectl get nodes
kubectl get po -n kad
kubectl logs kad-5dbdbb5cdb-brj7g
kubectl exec -it kad-5dbdbb5cdb-brj7g -- /bin/bash
kubectl delete deploy kad
kubectl apply -f kad.yml
kubectl describe svc kad
kubectl delete namespace kad
```

```
 kubectl get po -n default -v7
Config loaded from file:  /home/tom/.kube/config
"Request" verb="GET" url="https://api.c0.s8k.cz:6443/api/v1/namespaces/default/pods?limit=500" headers=<
    Accept: application/json;as=Table;v=v1;g=meta.k8s.io,application/json;as=Table;v=v1beta1
    User-Agent: kubectl/v1.35.0 (linux/amd64) kubernetes/6645204
"Response" status="200 OK" milliseconds=208
NAME                     READY   STATUS    RESTARTS   AGE
nginx-5f4dd4fb5b-7bjsl   1/1     Running   0          11h
nginx-5f4dd4fb5b-mr5jk   1/1     Running   0          11h
nginx-5f4dd4fb5b-sj29k   1/1     Running   0          11h
```

```
 curl localhost:8001/version
{
  "major": "1",
  "minor": "35",
  "emulationMajor": "1",
  "emulationMinor": "35",
  "minCompatibilityMajor": "1",
  "minCompatibilityMinor": "34",
  "gitVersion": "v1.35.0",
  "gitCommit": "66452049f3d692768c39c797b21b793dce80314e",
  "gitTreeState": "clean",
  "buildDate": "2025-12-17T12:32:07Z",
  "goVersion": "go1.25.5",
  "compiler": "gc",
  "platform": "linux/amd64"
}
```

First application

Elementary application consists of *deployment* and *service*. Common (not best) practise is to store resources locally in YAML files(s) and enforce them in Kubernetes.

- Create YAML files (e.g. `deploy.yml`) on your machine. Example is in **nginx/** directory.
- Enforce resource (create or update) it

```
kubectl apply -f deploy.yml
deployment.apps/nginx created
```

- Verify resource exists

```
kubectl get deployment
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   2/2     2            2           23s
```

- Resources can be deleted by name

```
kubectl delete deployment nginx
deployment.extensions "nginx" deleted
```

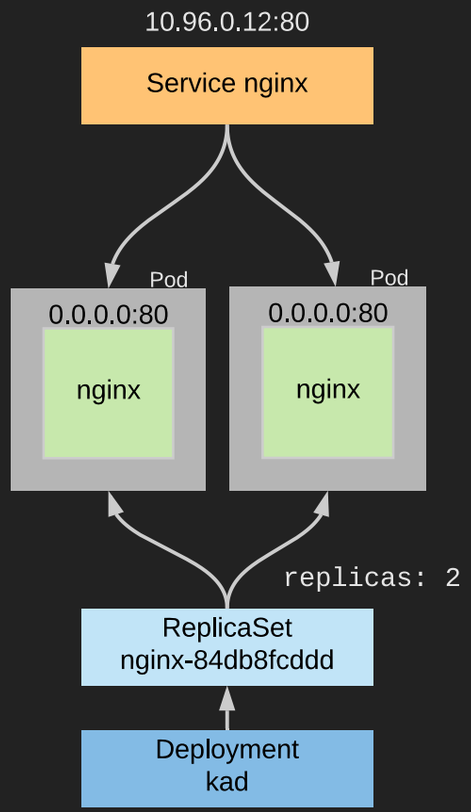
Deployment + service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: kad
spec:
  selector:
    app: kad
  ports:
  - protocol: TCP
    targetPort: 5000
    port: 80
```

- **gitlab.com/6shore.net/kubernetes-examples**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kad
spec:
  replicas: 2
  selector:
    matchLabels:
      app: kad
  template:
    metadata:
      labels:
        app: kad
    spec:
      containers:
      - name: app
        image: docker.io/tomkukral/kad
```

Immutable pods



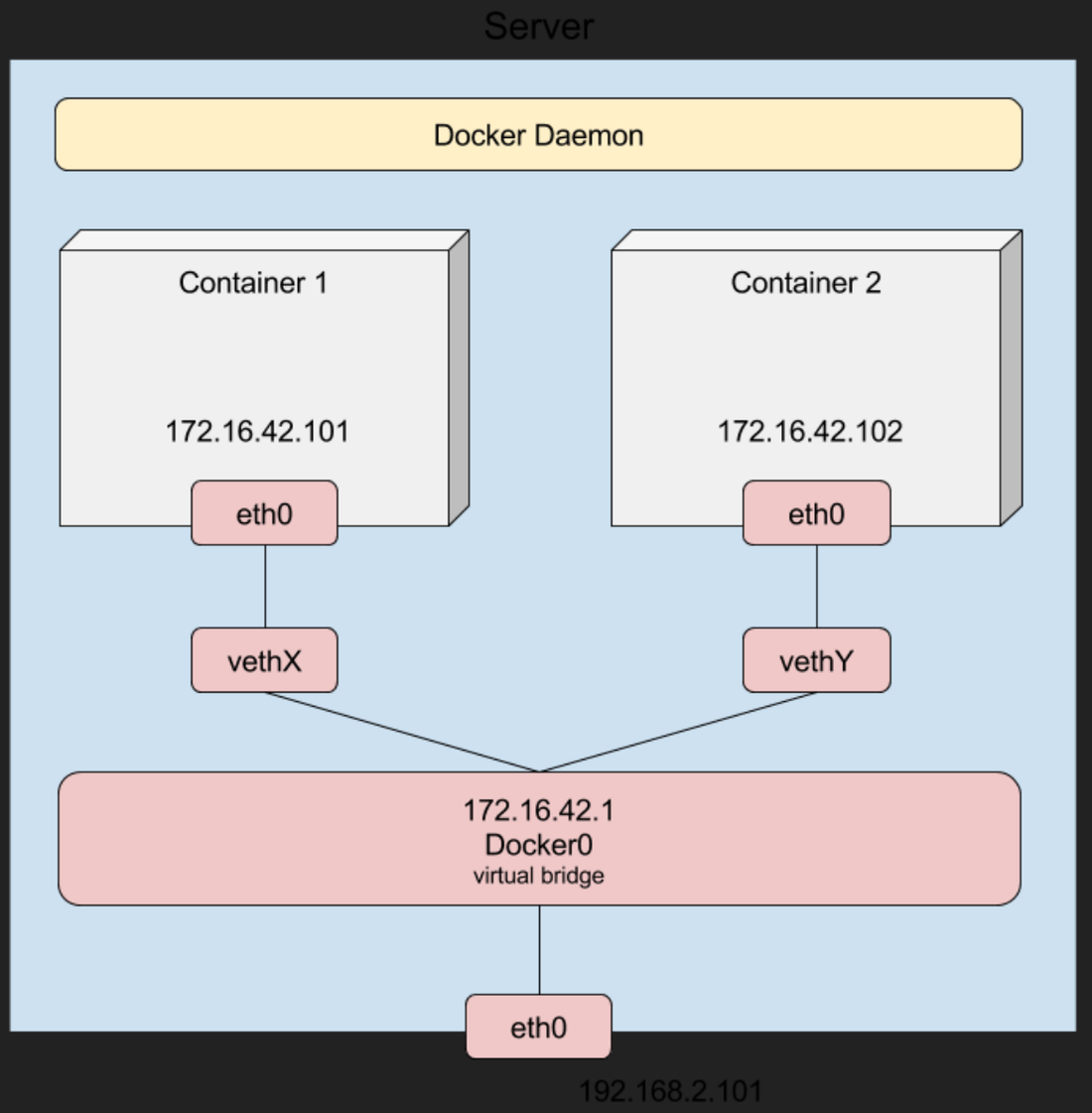

```
kubectl get all

NAME                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx    2/2     2            2           114m

NAME                                DESIRED   CURRENT   READY
replicaset.apps/nginx-84db8fcddd    2         2         2

NAME                            READY   STATUS    RESTARTS   AGE
pod/nginx-84db8fcddd-lxr98      1/1     Running   1          114m
pod/nginx-84db8fcddd-w8hqs      1/1     Running   1          114m

NAME                 TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)
service/nginx        ClusterIP   10.96.0.12    <none>        80/TCP
```

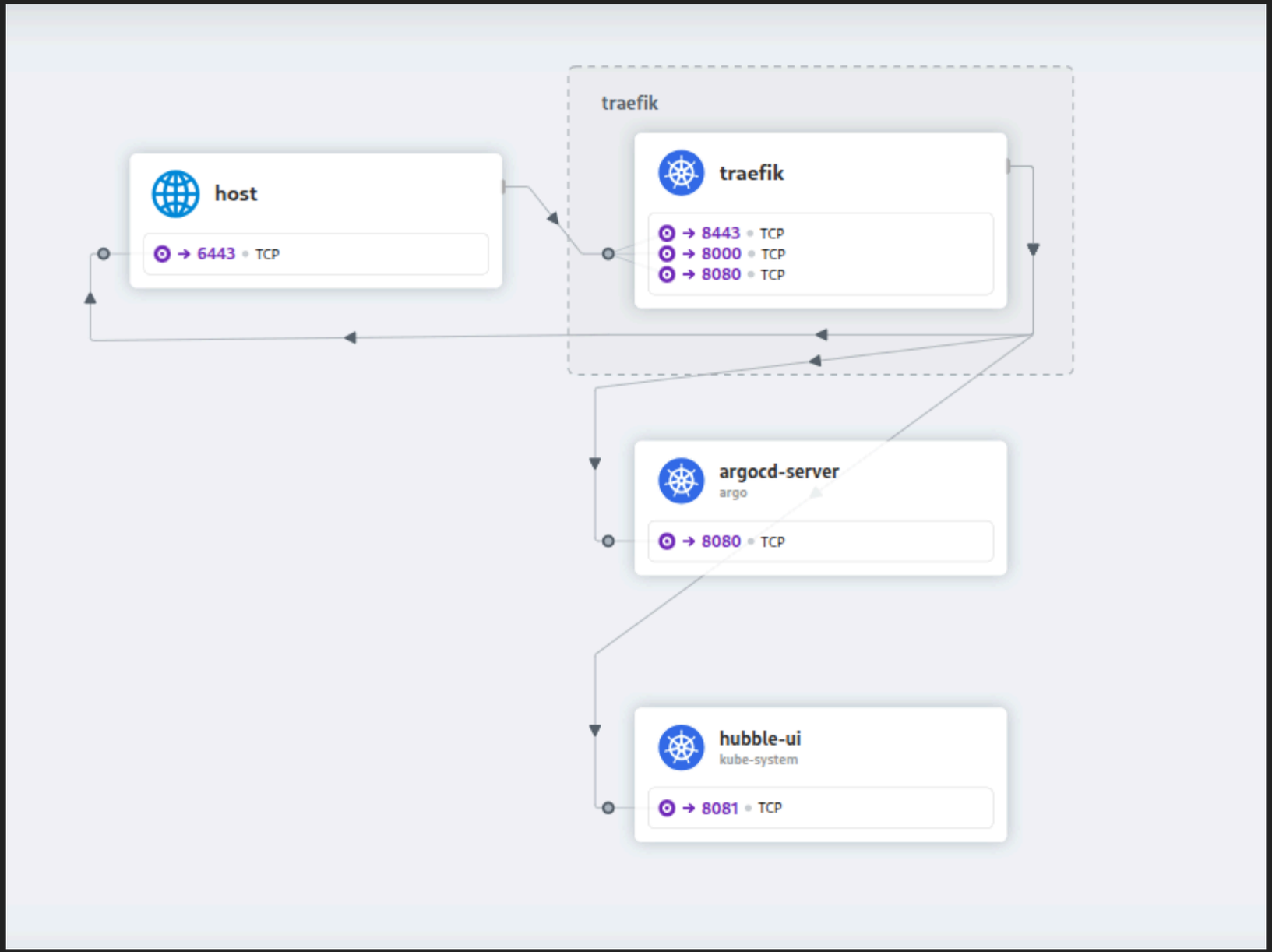
Pod is replaced on any "template" change.

Networking

- Provide:
  - ingress, egress, port mapping
  - communication between containers
  - single pod
  - single node
  - different node
  - **different cluster**

- Default network is bridge
- `docker0` bridge will choose unused subnet - risk of IP collision
- Uses iptables with NAT (MASQUERADE)


Server
Docker Daemon
Container 1
172.16.42.101
eth0
Container 2
172.16.42.102
eth0
vethX
vethY
172.16.42.1
Docker0
virtual bridge
eth0
192.168.2.101


traefik
host
→ 6443 • TCP
traefik
→ 8443 • TCP
→ 8000 • TCP
→ 8080 • TCP
argocd-server
argo
→ 8080 • TCP
hubble-ui
kube-system
→ 8081 • TCP

Configuration files

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: kad
  namespace: kad
data:
  filecontent: |
    Lorem ipsum dolor sit amet,
    consectetur adipiscing elit.
    Phasellus sodales sem ...
  color: red
```

Mount file containing `filecontent` to `/etc/kad/config.yml`

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kad
spec:
  ...
    spec:
      containers:
      - name: kad
        image: docker.io/tomkukral/kad
        volumeMounts:
        - name: config
          mountPath: /etc/kad/
      volumes:
      - name: config
        configMap:
          name: kad
          items:
          - key: filecontent
            path: config.yml
```

Exposing services

- **ClusterIP** is internal IP, for internal services
  - `ClusterIP ⇝ Pod IP`
- **NodePort** opens port on **all** nodes in cluster
  - `NodePort ⇝ Cluster IP ⇝ Pod IP`
- **LoadBalancer** service uses external load balancer
  - `external LB ⇝ NodePort ⇝ Cluster IP ⇝ Pod IP`
- **Ingress** controller (aka reverse proxy) provides external access to services in cluster, usually via HTTP
  - `Ingress controller ⇝ Service Endpoints ⇝ Pod IP`
- **HTTPRoute**

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: kad
spec:
  rules:
  - host: "kad.secfo.tech"
    http:
      paths:
      - pathType: Prefix
        path: /
        backend:
          service:
            name: kad
            port:
              number: 80
```

Other workload

- **DaemonSet**

  Is special type of pod. DaemonSet is running on every available node.

- **Job**, **CronJob**

  Make job (by running pod), ensure it will successfully terminate and save output.

- **StatefulSet**

  Provide pods with unique identity, sequential scaling and auto-allocated fixed storage.

Babysitting the app

Crashes

- Kubernetes (container runtime) will restart the container on any crash or regular exit code.

```
NAME                   READY   STATUS             RESTARTS      AGE
kad-5dbdbb5cdb-brj7g   0/1     Running            1 (13s ago)   4m23s
kad-5dbdbb5cdb-hv25b   0/1     Running            2 (12s ago)   4m24s
kad-5dbdbb5cdb-r8vtv   0/1     CrashLoopBackOff   1 (12s ago)   4m21s
redis-0                1/1     Running            0             21m
```

Resource management

- Memory leak, CPU spike, storage exhaustion, ...

```
containers:
- name: app
  image: docker.io/tomkukral/kad
  resources:
    requests:
      cpu: 50m
      memory: 128Mi
    limits:
      memory: 150Mi
      ephemeral-storage: 64Mi
```

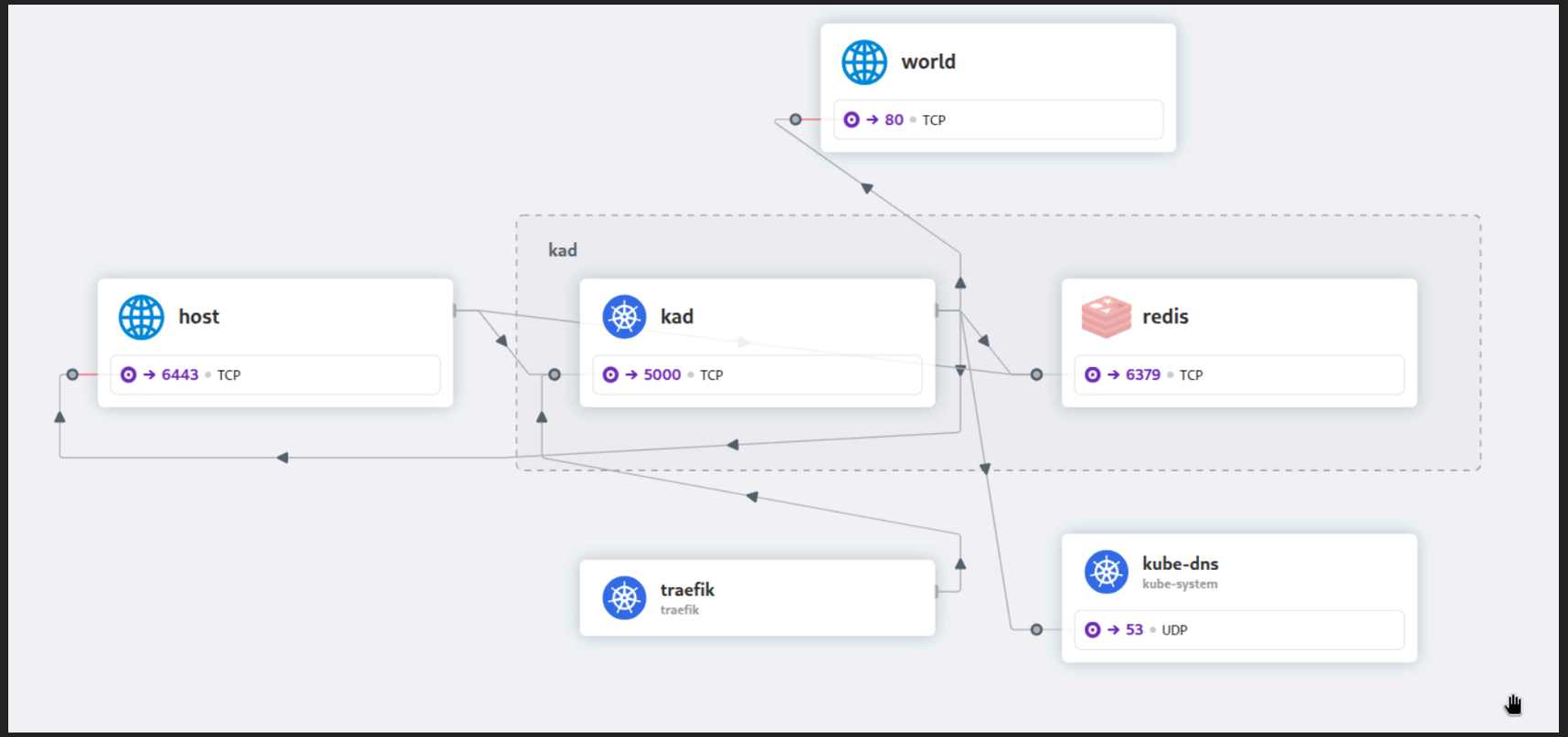
Probes

Application (container) is running but application isn't able to serve requests.

- **Liveness and readiness probes**

```
containers:
- name: app
  image: docker.io/tomkukral/kad
  livenessProbe:
    httpGet:
      path: /check/live
      port: 5001
    initialDelaySeconds: 10
    periodSeconds: 30
  readinessProbe:
    httpGet:
      path: /check/ready
      port: 5001
    initialDelaySeconds: 1
    periodSeconds: 5
```

Malware and network security



Hardware failures

Pods will be rescheduled to other nodes on hardware failure, but external dependencies can block it.

Kubernetes at SF

- AKS Kubernetes
- Onprem deployed via kubeadm

```
600-1k nodes
50-150k CPU
10T-1PB RAM
20-30k pods
```

SF

second-foundation.eu

Q&A