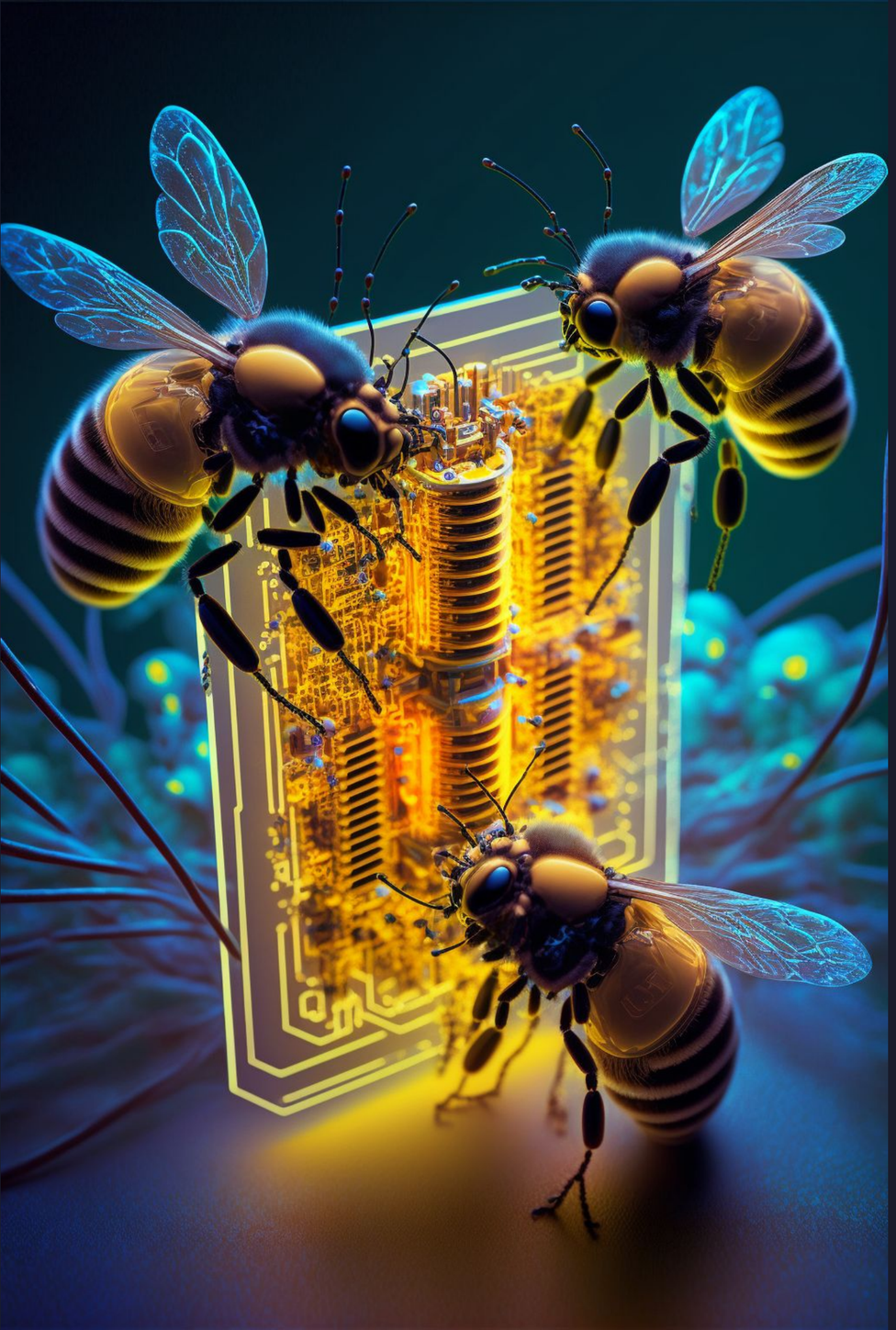
ISOVALENT

Turn Your Kubernetes Cluster Into a Castle Using Advanced Cilium Network Policies

Speaker: **Philip Schmid**



Agenda

- Introduction
- Securing the Nodes
- Protecting Infra Components
- Default Rule Set for User Workloads

ISOVALENT

About Me

Philip Schmid

Solutions Architect @ Isovalent

CK(AD|A|S) | CCNP

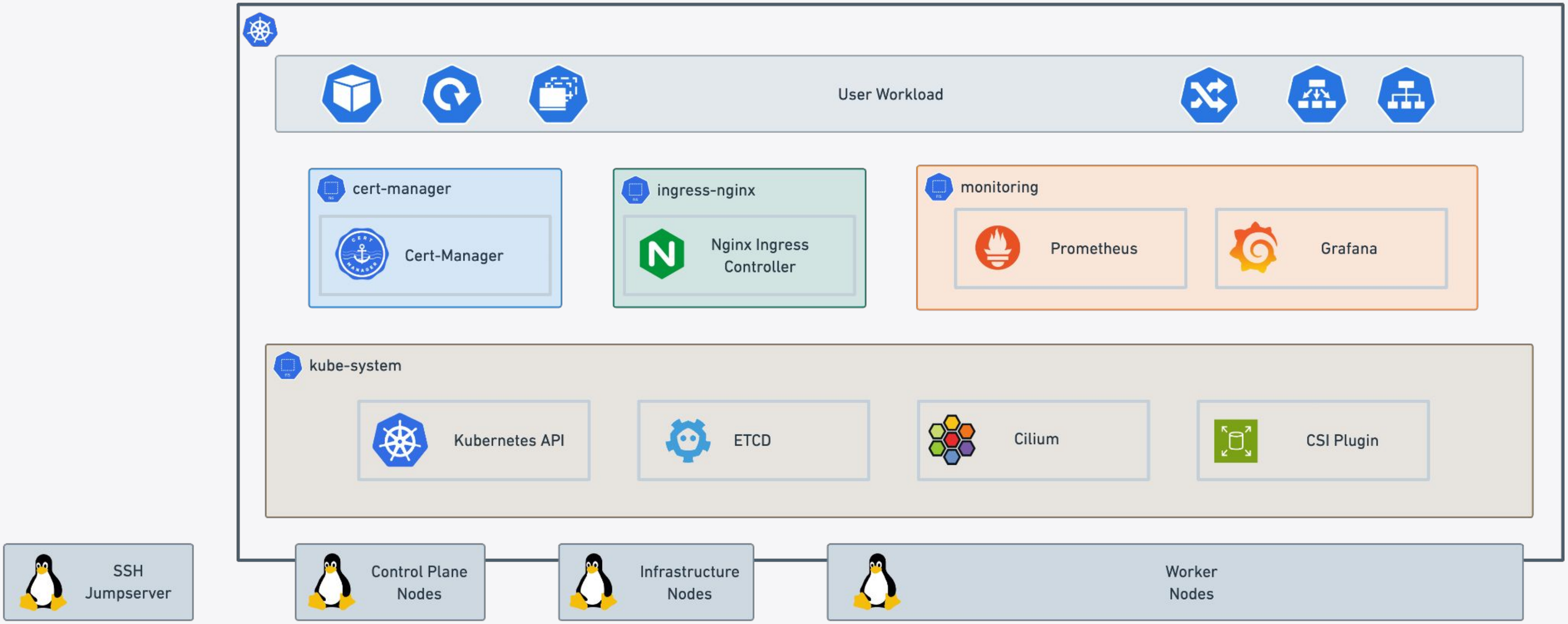
Introduction

Goal

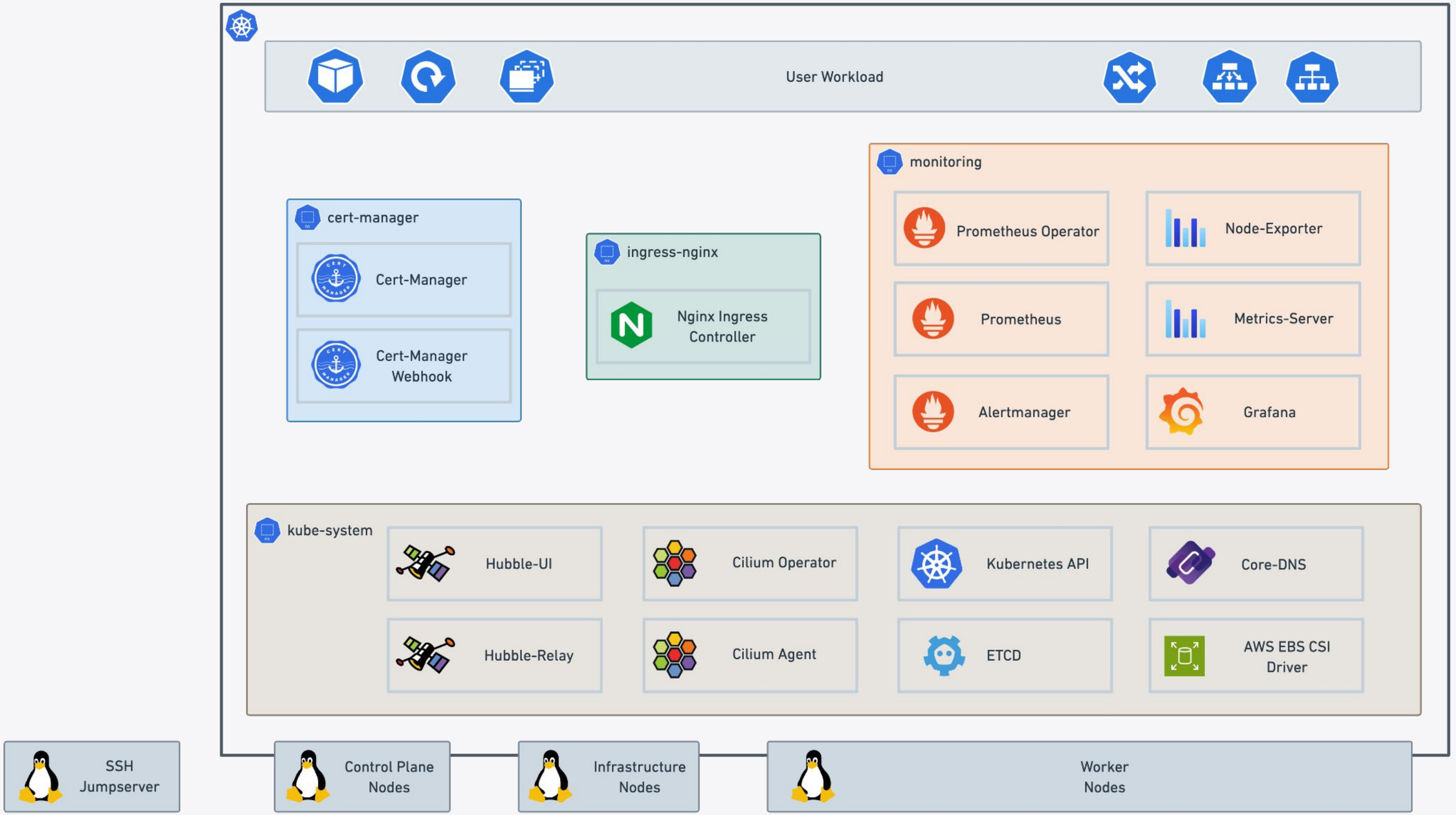
- This isn’t a ...
  - ... Network Policy 101 talk
  - ... abstract Network Policy design pattern talk

- This talk shows a **practical example** of how **you could** secure ...
  - ... **your** Kubernetes nodes and clusters in general
  - ... **your** Kubernetes infrastructure components and user workloads
- ... by using advanced Cilium Network Policies

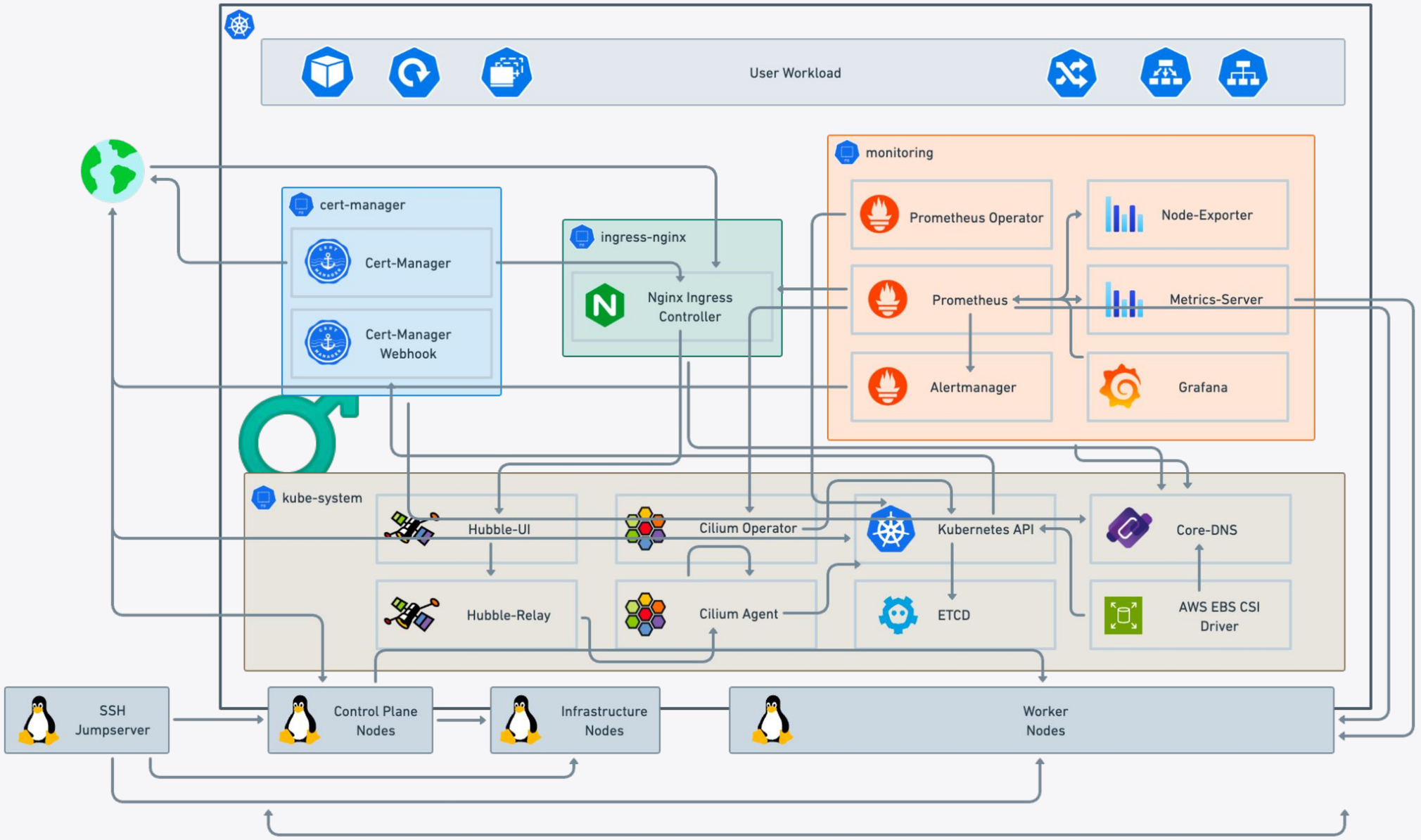
Demo Use Case



Demo Use Case in Detail



Demo Use Case in Detail with Connections



Structured Way to Succeed with Network Policies

1. General recommendation: Take care of Network Policies as early as possible
2. Start bottom up:
   a. Nodes
   b. Kubernetes
   c. Cilium
   d. Infrastructure components
   e. User workloads
3. Always keep an eye on DROPPED connections:

```
hubble observe -t policy-verdict -f --verdict AUDIT --verdict DROPPED
```

4. Re-iterate over policies until there are no false-positive left

Securing the Nodes

Motivation for Securing the Nodes

- Security team: *“Is the Linux host firewall enabled?”*
- Industry standards like
  - PCI DSS (v4.0): *“1.2 Network security controls (NSCs) are configured and maintained”*
  - ISO 27001:2013: A.13.1.1 Network Controls
- K8s Ops team: Improve own security posture

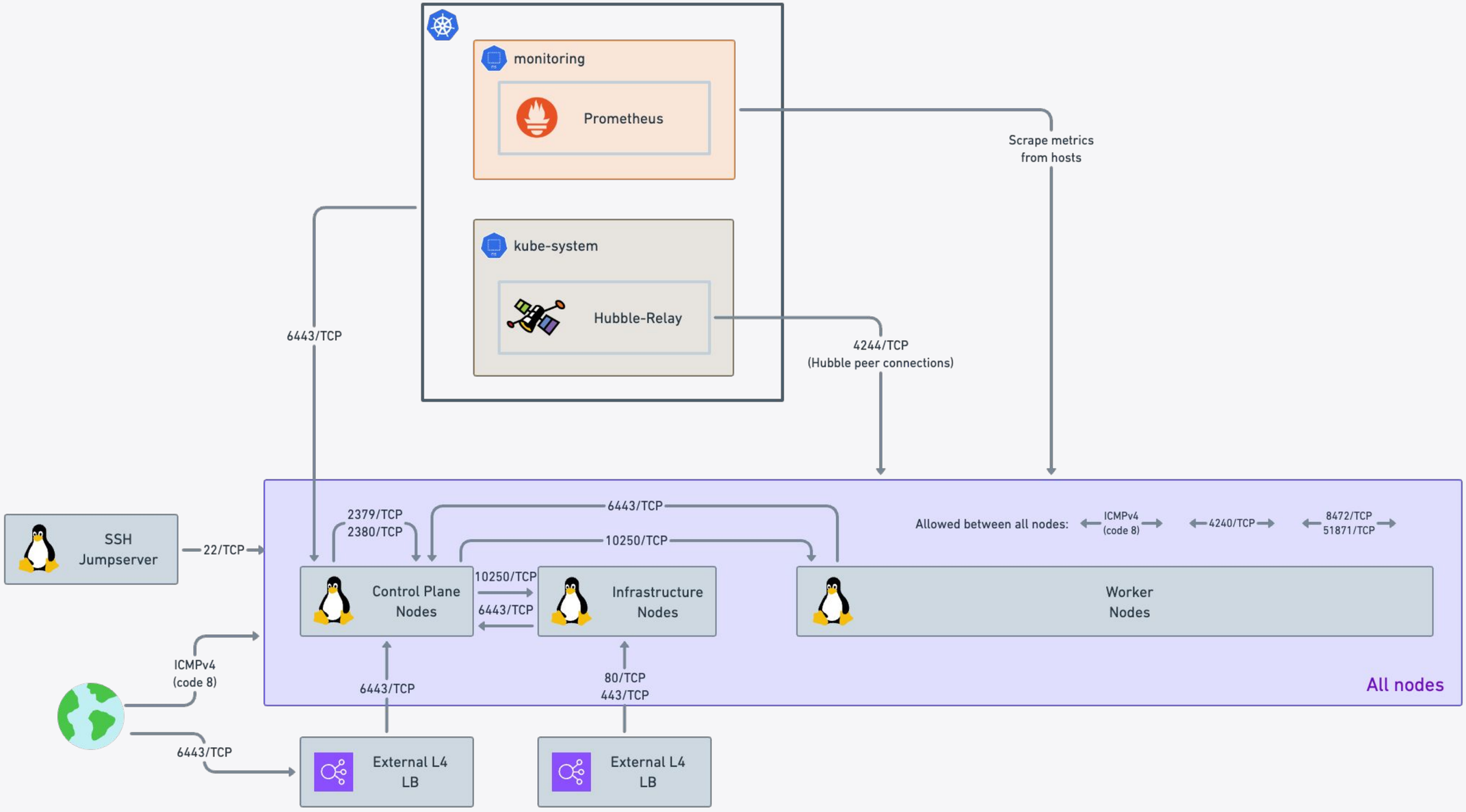
- **→ Use Cilium Host Firewall / Policies, especially for high security environments**

Securing the Nodes using Cilium Host Firewall

- Implement host firewalling solely using eBPF (no netfilter!)
- Explicitly enable it: `hostFirewall.enabled=true`
- Rules are fully managed via Kubernetes
  - `CiliumClusterwideNetworkPolicy` with `spec.nodeSelector` (instead of `spec.endpointSelector`)
- **Important:** After enabling Cilium Host Firewall, **enable Policy Audit mode** for the Host CiliumEndpoints before applying any host policies (see doc)!

```yaml
apiVersion: "cilium.io/v2"
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: host-cp
spec:
  description: Cilium host policy set for CP nodes
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/control-plane: ""
  ingress:
  # Kubernetes control plane connectivity from everywhere
  - toPorts:
    - ports:
        # Kubernetes API
      - port: "6443"
        protocol: TCP
  # ETCD control plane connectivity from nodes
  - fromEntities:
    - kube-apiserver
    toPorts:
      - ports:
          # etcd client port
        - port: "2379"
          protocol: TCP
          # etcd peer communication port
        - port: "2380"
          protocol: TCP
```

Securing the Nodes using Cilium Host Firewall



Securing the Nodes using Cilium Host Firewall

Control Plane nodes

```yaml
apiVersion: "cilium.io/v2"
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: host-cp
spec:
  description: Cilium host policy set for CP nodes
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/control-plane: ""
  ingress:
  # Kubernetes control plane connectivity from everywhere
  - toPorts:
    - ports:
        # Kubernetes API
      - port: "6443"
        protocol: TCP
  # ETCD control plane connectivity from nodes
  - fromEntities:
    - kube-apiserver
    toPorts:
      - ports:
          # etcd client port
        - port: "2379"
          protocol: TCP
          # etcd peer communication port
        - port: "2380"
          protocol: TCP
```

Securing the Nodes using Cilium Host Firewall

Infra nodes

```yaml
apiVersion: "cilium.io/v2"
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: host-infra
spec:
  description: Cilium host policy set for infra nodes
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/infra: ""
  ingress:
  # Data plane connectivity from everywhere to Nginx
  - toPorts:
    - ports:
        # Ingress HTTP
      - port: "80"
        protocol: TCP
        # Ingress HTTPS
      - port: "443"
        protocol: TCP
```

Securing the Nodes using Cilium Host Firewall

All nodes – part 1

```yaml
apiVersion: "cilium.io/v2"
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: host-all
spec:
  description: Cilium host policy for all nodes
  nodeSelector: {}
  ingress:
  # System management: ALlow SSH access from jumphost
  - fromCIDR:
    - 10.1.101.110/32
    toPorts:
    - ports:
      - port: "22"
        protocol: TCP
  # System monitoring: Allow ping to hosts
  - icmps:
    - fields:
      - type: 8
        family: IPv4
  # KAPI Control plane connectivity for Kubelet Pod log streaming
  - fromEntities:
    - kube-apiserver
    toPorts:
    - ports:
      - port: "10250"
        protocol: TCP
```

Securing the Nodes using Cilium Host Firewall

All nodes – part 2

```
28  # Cilium control plane agent HTTP health checks
29  - fromEntities:
30    - remote-node
31    toPorts:
32    - ports:
33      - port: "4240"
34        protocol: TCP
35  # Data plane inter-node connectivity
36  - fromEntities:
37    - host
38    - remote-node
39    toPorts:
40    - ports:
41        # Cilium VXLAN
42      - port: "8472"
43        protocol: UDP
44        # Cilium WireGuard
45      - port: "51871"
46        protocol: UDP
47  # Cilium Hubble relay access to Cilium agent Hubble peers
48  - fromEndpoints:
49    - matchLabels:
50        k8s:io.kubernetes.pod.namespace: kube-system
51        k8s:app.kubernetes.io/name: hubble-relay
52    toPorts:
53    - ports:
54      - port: "4244"
55        protocol: TCP
56  # Allow Metrics-Server to reach Kubelet metrics
57  - fromEndpoints:
58    - matchLabels:
59        k8s:io.kubernetes.pod.namespace: monitoring
60        k8s:app.kubernetes.io/name: metrics-server
61    toPorts:
62    - ports:
63      - port: "10250"
64        protocol: TCP
```

Protecting Infra Components

Goals for Protecting Infra Components

1. **Allow all** ingress and egress connections **within an infra namespace**
2. Try to **only allow specific** ingress and egress **connections from and to the infra namespace**
3. Implement a **restrictive, static rule set** that doesn’t need to be updated when a new user workload NS is created AND doesn’t require the user workload NS to allow something from/to infra components (Core-DNS, Nginx, Prometheus)

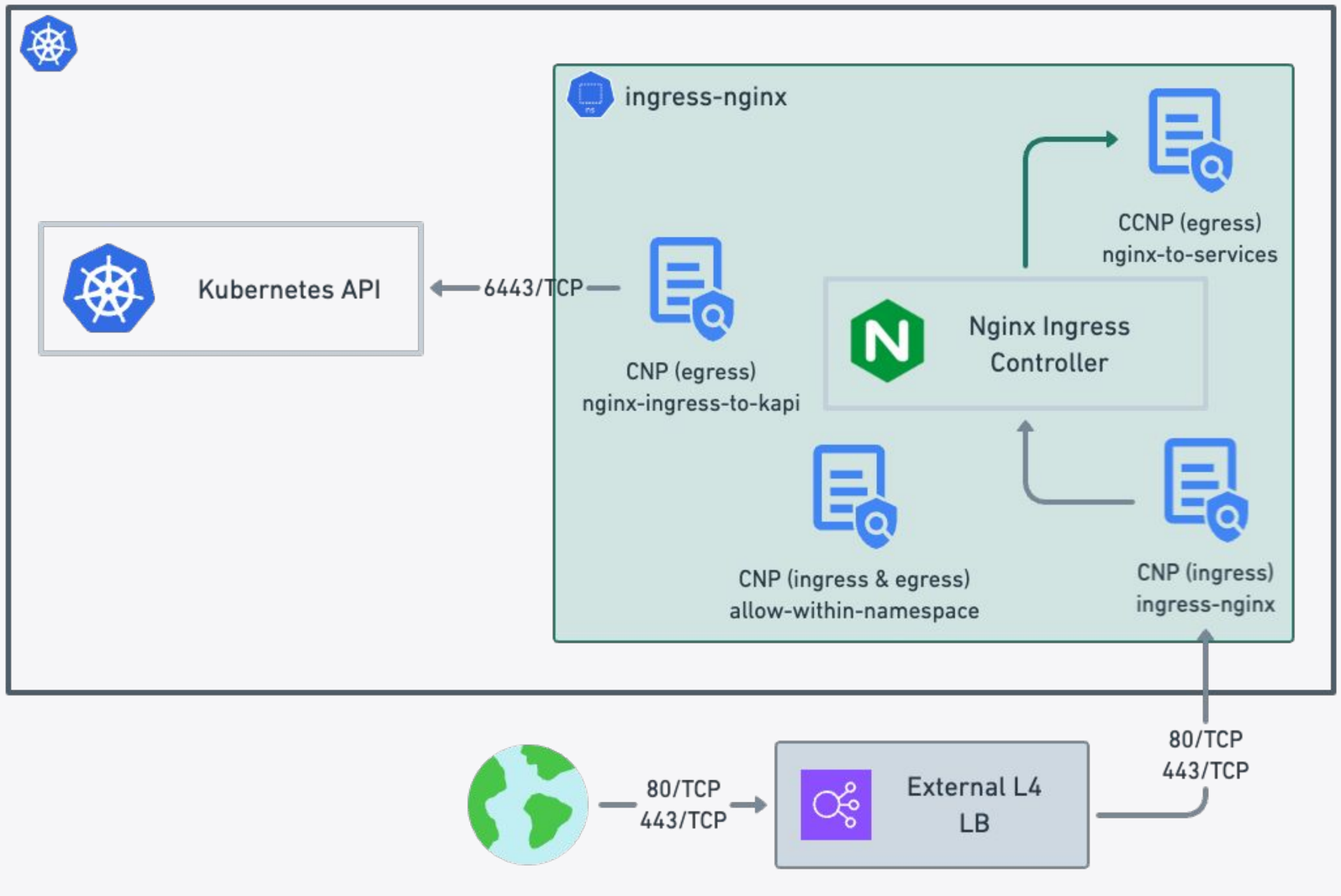
Protecting Infra Components

1. **Allow all** ingress and egress connections **within an infra namespace** ✅

```yaml
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: allow-within-namespace
  namespace: infra-ns-name
spec:
  endpointSelector: {}
  ingress:
    - fromEndpoints:
        - {}
  egress:
    - toEndpoints:
        - {}
```

Protecting Infra Components

2. Try to **only allow specific** ingress and egress **connections from and to the infra namespace**



Protecting Infra Components

2. Try to **only allow specific** ingress and egress **connections from and to the infra namespace**

```yaml
---
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
metadata:
  name: allow-within-namespace
  namespace: ingress-nginx
spec:
  description: Allow NS internal traffic, block everything else
  endpointSelector: {}
  ingress:
  - fromEndpoints:
    - {}
  egress:
  - toEndpoints:
    - {}
```

```yaml
---
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
metadata:
  name: all-to-nginx
  namespace: ingress-nginx
spec:
  description: Allow accessing Nginx ingress controller from everywhere
  endpointSelector:
    matchLabels:
      k8s:app.kubernetes.io/name: ingress-nginx
  ingress:
  - fromEntities:
    - all
    toPorts:
    - ports:
      - port: "80"
        protocol: TCP
      - port: "443"
        protocol: TCP
```

Protecting Infra Components

2. Try to **only allow specific** ingress and egress **connections from and to the infra namespace** ✅

```yaml
---
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
metadata:
  name: nginx-ingress-to-kapi
  namespace: ingress-nginx
spec:
  description: Allow Nginx ingress controller to reach KAPI
  endpointSelector:
    matchLabels:
      k8s:app.kubernetes.io/name: ingress-nginx
  egress:
  - toEntities:
    - kube-apiserver
    toPorts:
    - ports:
      - port: "6443"
        protocol: TCP
```

Protecting Infra Components

3. Implement a **restrictive, static rule set** that doesn't need to be updated when a new user workload NS is created AND doesn't require the user workload NS to allow something from/to infra components (Core-DNS, Nginx, Prometheus) 🤔

This requires:

1. A generic egress rule which allows leaving the infra NS toward user NS (only specific ones) 🦄
2. A generic ingress rule which allows accessing user NS (only specific ones) 🪄
3. 1. & 2. but also in the other direction for passive infra components (e.g. Core-DNS)

Protecting Infra Components

Core-DNS:

```yaml
---
apiVersion: "cilium.io/v2"
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: global-infra-dns
spec:
  description: Allow coredns access from everywhere
  endpointSelector: {}
  egress:
  - toEndpoints:
    - matchLabels:
        k8s:io.kubernetes.pod.namespace: kube-system
        k8s:k8s-app: kube-dns
    toPorts:
    - ports:
      - port: "53"
        protocol: ANY
      rules:
        dns:
        - matchPattern: "*"
```

```yaml
---
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
metadata:
  name: allow-to-coredns
  namespace: kube-system
spec:
  description: Allow ingress to coredns from everywhere
  endpointSelector:
    matchLabels:
      k8s:io.kubernetes.pod.namespace: kube-system
      k8s:k8s-app: kube-dns
  ingress:
  - fromEntities:
    - all
    toPorts:
    - ports:
      - port: "53"
        protocol: ANY
```

Protecting Infra Components

- **Problem:** The egress `CiliumClusterwideNetworkPolicy` for Core-DNS was intended for all namespaces → Nginx ingress and Prometheus should only be able to access certain namespaces!

- **Idea:** Add predefined labels to namespaces to which only specific infrastructure components should have access to 🚀
- Luckily, a `CiliumClusterwideNetworkPolicy` has knowledge of labels on namespaces
  - CiliumNetworkPolicies are namespaced → no knowledge about NS labels

```
k8s:io.cilium.k8s.namespace.labels.<label-key>: <label-value>
```

Protecting Infra Components

Nginx Ingress Controller:

```yaml
---
apiVersion: "cilium.io/v2"
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: nginx-to-services
spec:
  description: Allow Nginx to access svc of NS with exposed label
  endpointSelector:
    matchLabels:
      k8s:app.kubernetes.io/name: ingress-nginx
      k8s:io.kubernetes.pod.namespace: ingress-nginx
  egress:
  - toEndpoints:
    - matchLabels:
        k8s:io.cilium.k8s.namespace.labels.exposed: "true"
```

```yaml
---
apiVersion: "cilium.io/v2"
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: global-infra-nginx
spec:
  description: Allow connections from Nginx to labeled namespaces
  endpointSelector:
    matchLabels:
      k8s:io.cilium.k8s.namespace.labels.exposed: "true"
  ingress:
  - fromEndpoints:
    - matchLabels:
        k8s:io.kubernetes.pod.namespace: ingress-nginx
        k8s:app.kubernetes.io/name: ingress-nginx
```

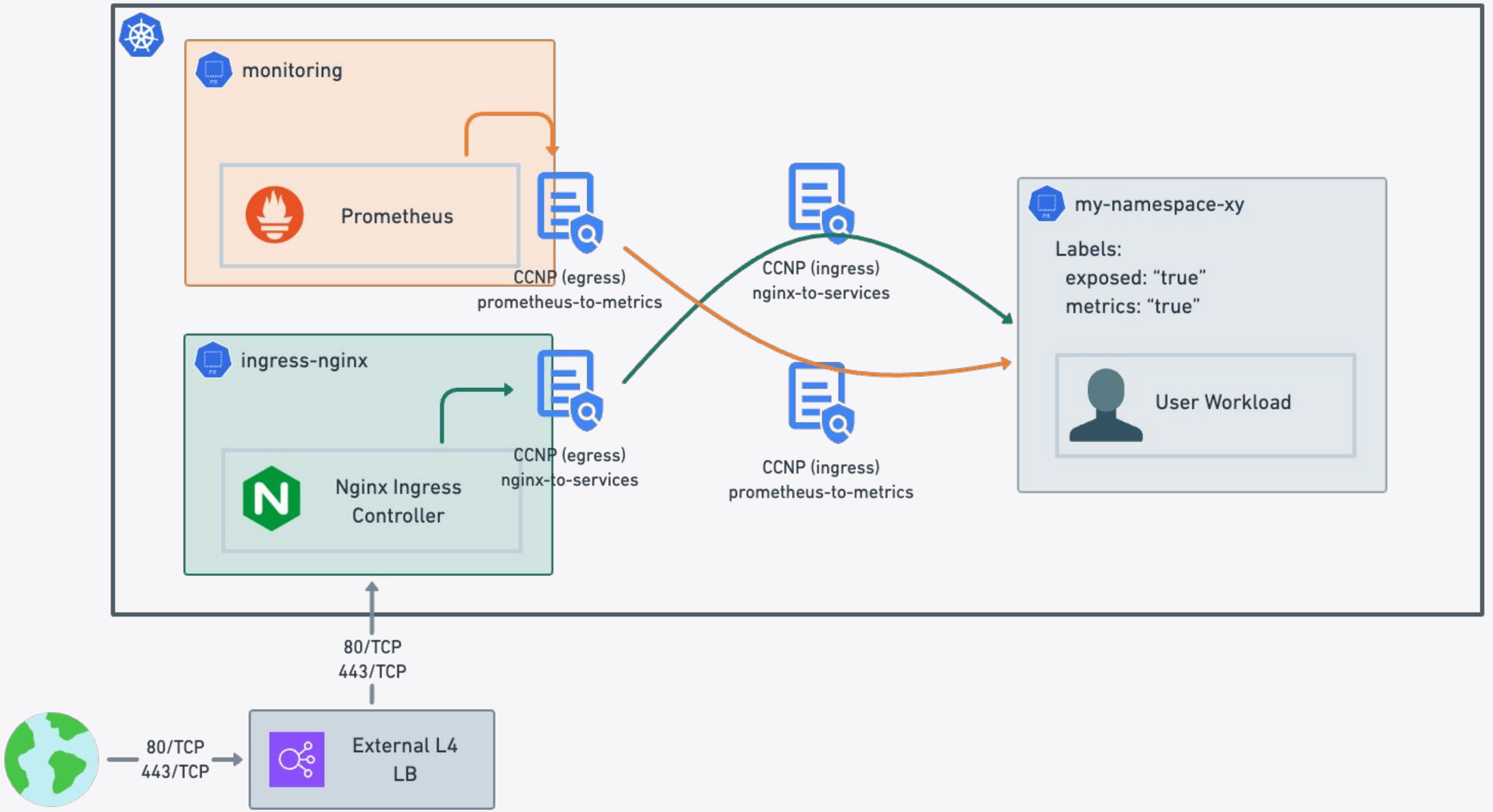
Protecting Infra Components

Prometheus:

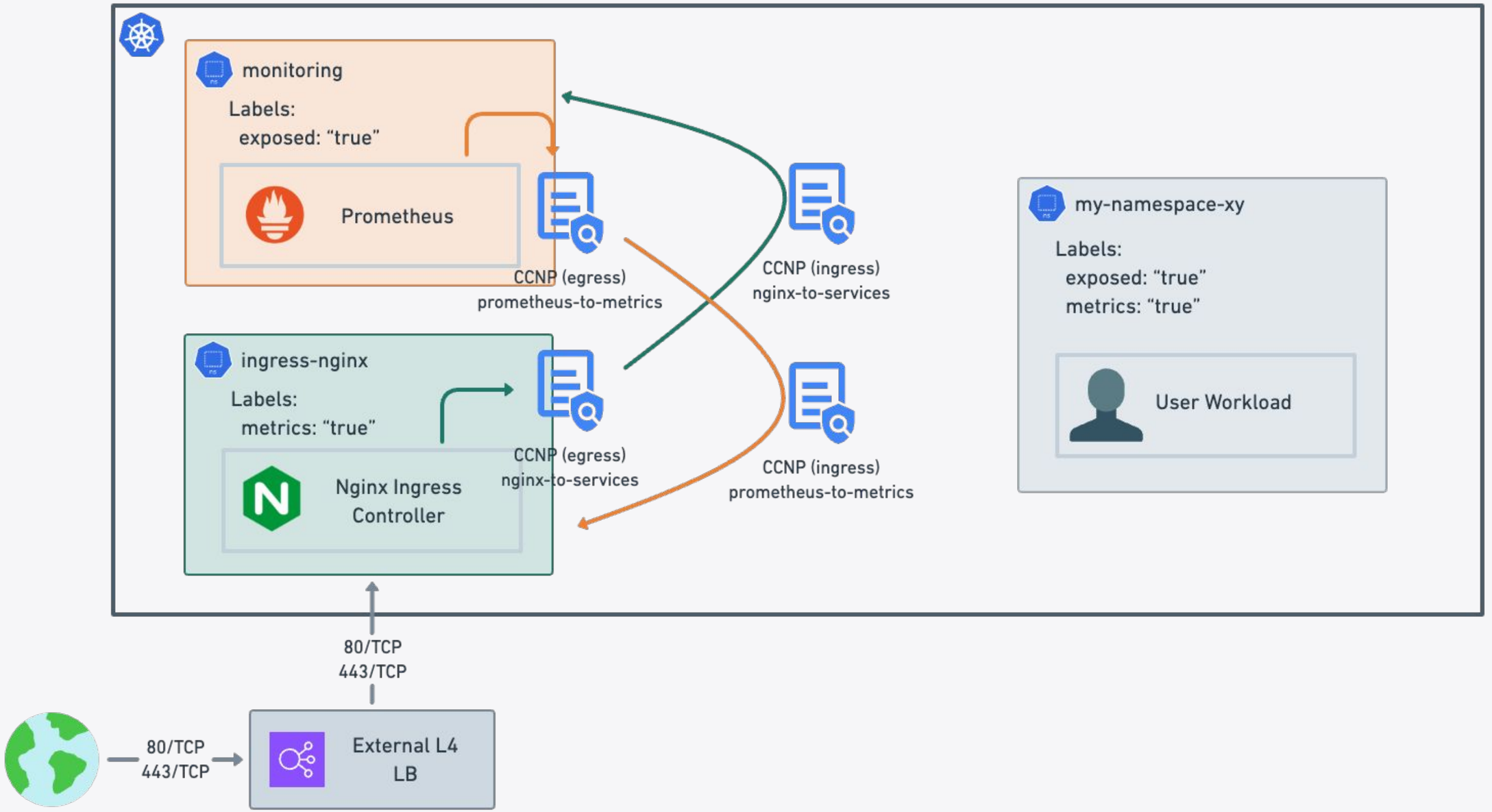
```yaml
---
apiVersion: "cilium.io/v2"
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: prometheus-to-metrics
spec:
  description: Allow Prometheus to scrape metrics in NS with metrics label
  endpointSelector:
    matchLabels:
      k8s:app.kubernetes.io/name: prometheus
      k8s:io.kubernetes.pod.namespace: monitoring
  egress:
  - toEndpoints:
    - matchLabels:
        k8s:io.cilium.k8s.namespace.labels.metrics: "true"
```

```yaml
---
apiVersion: "cilium.io/v2"
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: global-infra-prometheus
spec:
  description: Allow connections from Prometheus to labeled namespaces
  endpointSelector:
    matchLabels:
      k8s:io.cilium.k8s.namespace.labels.metrics: "true"
  ingress:
  - fromEndpoints:
    - matchLabels:
        k8s:io.kubernetes.pod.namespace: monitoring
        k8s:app.kubernetes.io/name: prometheus
```

Protecting Infra Components



Protecting Infra Components

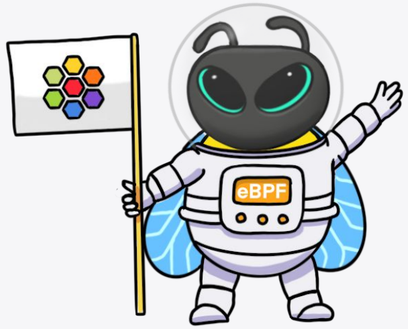


Default Rule Set for User Workloads

Default Rule Set for User Workloads

- Transition from “Coarse” (L3, L4 max.) to “Fine-Grained” (L4, DNS, L7) NS policies:
  - **Internal:** Start with allowing all ingress/egress communication within a namespace
  - **Egress:** Define to where this application needs to connect to
  - **Ingress:** Define from where something needs to connect to this application
- Use Cilium Hubble observability data to identify further communication flows

```
hubble observe -t policy-verdict -f --namespace <ns-name> [--verdict DROPPED]
```



Default Rule Set for User Workloads

- Every `CiliumClusterwideNetworkPolicy` with `spec.endpointSelector: {}` applies to all namespaces! In our case:
  - Egress to Core-DNS
  - Metric scrapes from Prometheus (with NS label `metrics: "true"`)
  - Ingress from Nginx ingress controller (with NS label `exposed: "true"`)
- Our `CiliumClusterwideNetworkPolicy` setup creates a deny-all behavior for all namespaces – even within the namespace
  - Add at least a `CiliumNetworkPolicy` to allow all namespace internal traffic

Default Rule Set for User Workloads

```yaml
# -- User workload NS creation
---
apiVersion: v1
kind: Namespace
metadata:
  name: my-namespace-xy
  # Customized labels
  # - Add exposed="true" label in case Nginx should expose something from this NS
  # - Add metrics="true" label in case Prometheus should scrape metrics from this NS
  labels:
    exposed: "true"
    metrics: "true"
```

Default Rule Set for User Workloads

```yaml
# -- Default rule
---
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
metadata:
  name: allow-within-namespace
  namespace: my-namespace-xy
spec:
  description: Allow NS internal traffic, block everything else
  endpointSelector: {}
  ingress:
  - fromEndpoints:
    - {}
  egress:
  - toEndpoints:
    - {}
```

ISOVALENT

Thank you!

Additional Resources

- Cluster setup and presented policies:
  - https://github.com/PhilipSchmid/cilium-netpol-demo
- KubeCon 22: Adopting Network Policies in Highly Secure Environments
  - https://www.youtube.com/watch?v=yikVhGM2ye8
- 10 Free hands-on labs about Cilium Network Policies:
  - https://isovalent.com/learning-tracks/#securityProfessionals
- More “Securing Network with Cilium” examples:
  - https://docs.cilium.io/en/stable/security/
- Cilium Network Policy documentation entry point:
  - https://docs.cilium.io/en/stable/security/policy/

- **The cluster setup, all shown and even more policies can be found on Github:** https://github.com/PhilipSchmid/cilium-netpol-demo