Network security in Kubernetes

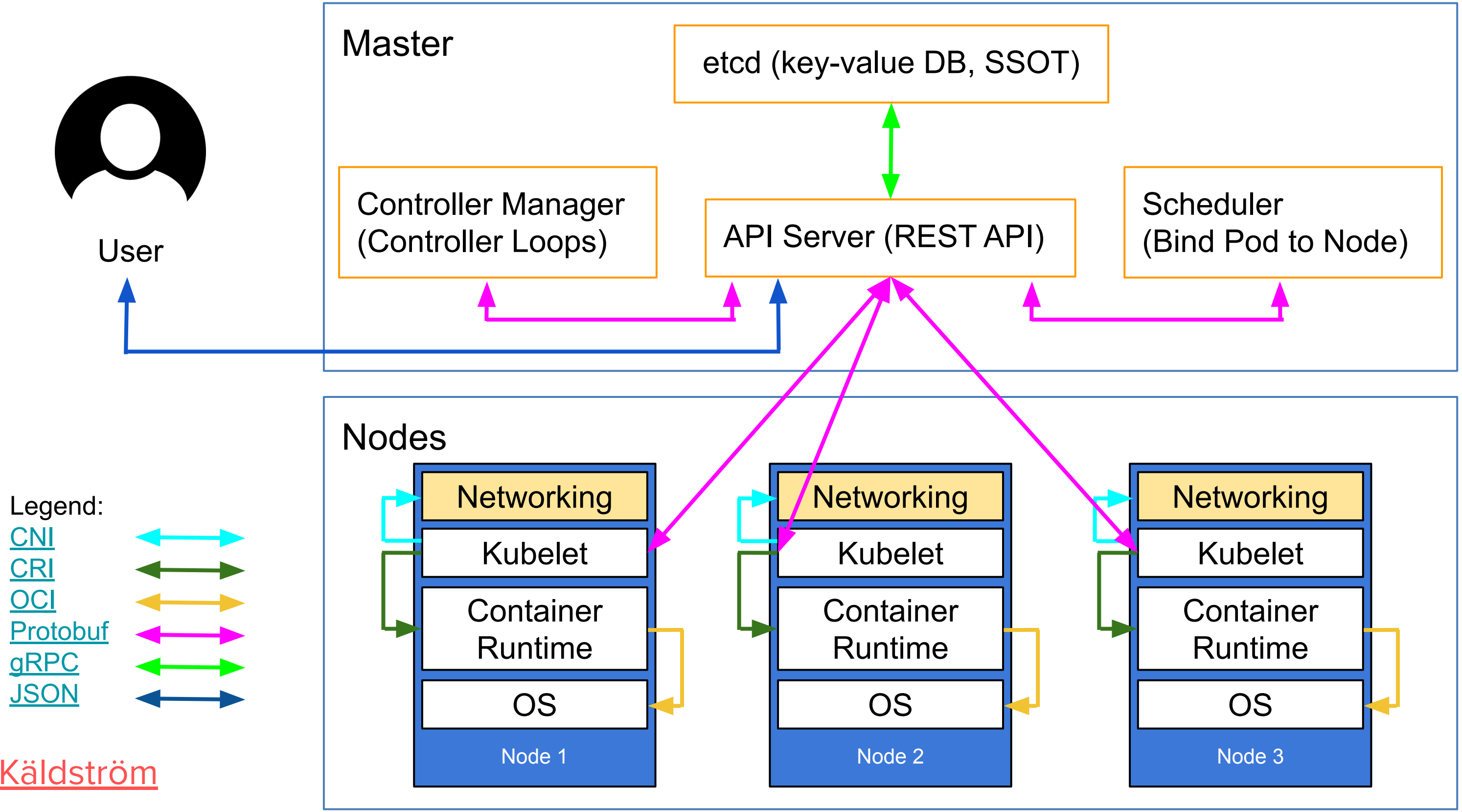
whoami

- Senior Security Engineer at Adevinta
- Member of CNCF Security SIG
- Current focus:
    - Containers’ security
    - Kubernetes
    - Machine Learning platforms
- Hobbies:
    - SciFi
    - Skiing
    - Hiking

Agenda

- Kubernetes High level Architecture
- CNI
- Network Policies
- Service Meshes
- Testing
- Questions

Kubernetes’ high-level component architecture



User

Master

etcd (key-value DB, SSOT)

Controller Manager (Controller Loops)

API Server (REST API)

Scheduler (Bind Pod to Node)

Nodes

Node 1	Node 2	Node 3
Networking	Networking	Networking
Kubelet	Kubelet	Kubelet
Container Runtime	Container Runtime	Container Runtime
OS	OS	OS

Legend:

- CNI
- CRI
- OCI
- Protobuf
- gRPC
- JSON

By Lucas Käldström

Kubernetes’ high-level component architecture

Primitives:

- Pod - a deployment unit, can contain multiple containers
- Label - a logical grouping
- Namespace (not Linux namespace) - a resource grouping

Container Network Interface

Some of the popular ones:

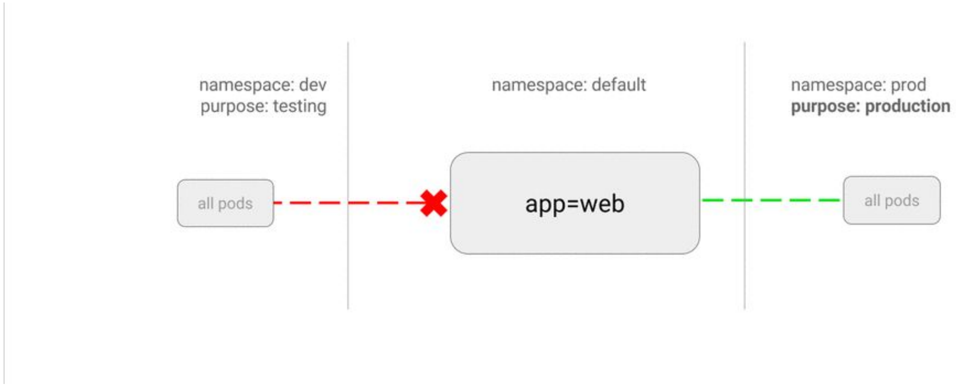
- Flannel
- Calico
- Cilium
- Amazon VPC CNI

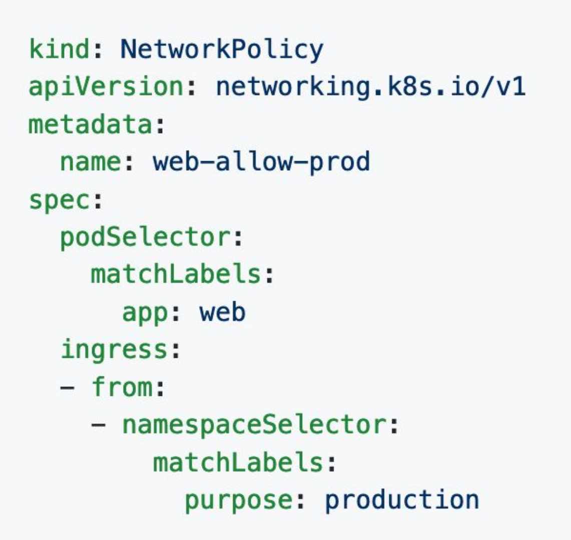
Network Policy

Covers:

- Protocols
    - TCP
    - UDP
    - SCTP (k8s 1.12+)
- CIDRs
- K8S Objects
    - Pods
    - Namespaces

Network Policies

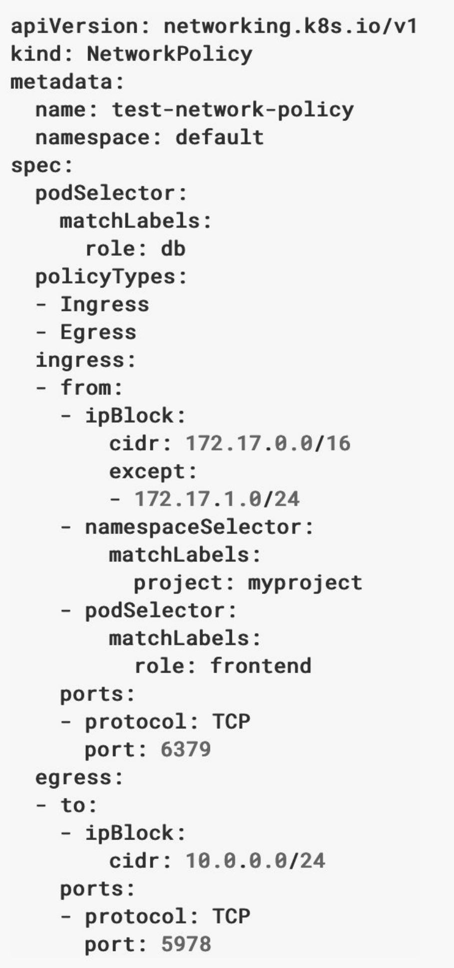




```yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: web-allow-prod
spec:
  podSelector:
    matchLabels:
      app: web
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          purpose: production
```

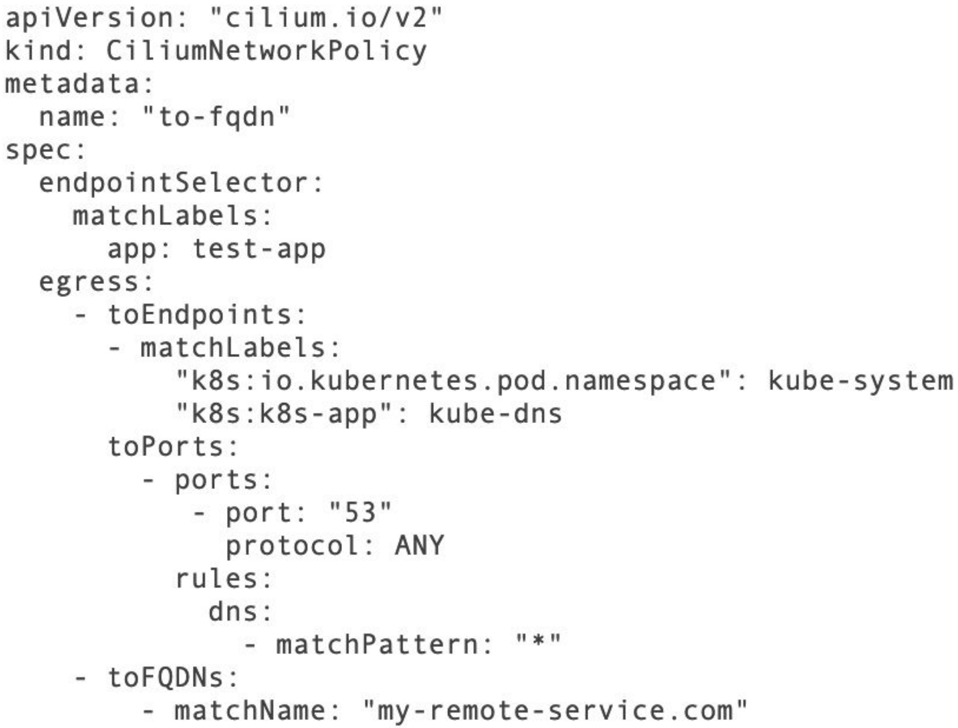
Source link

Network Policies



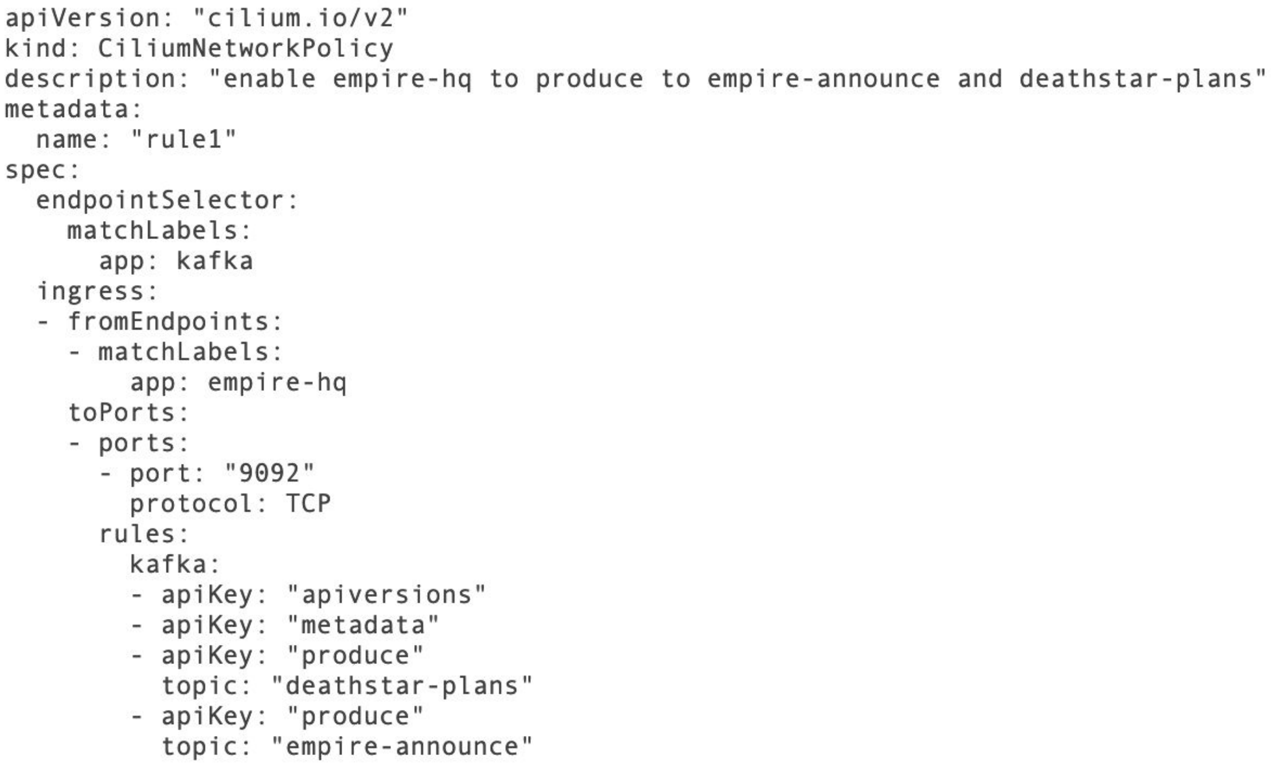
```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

Cilium Network Policies



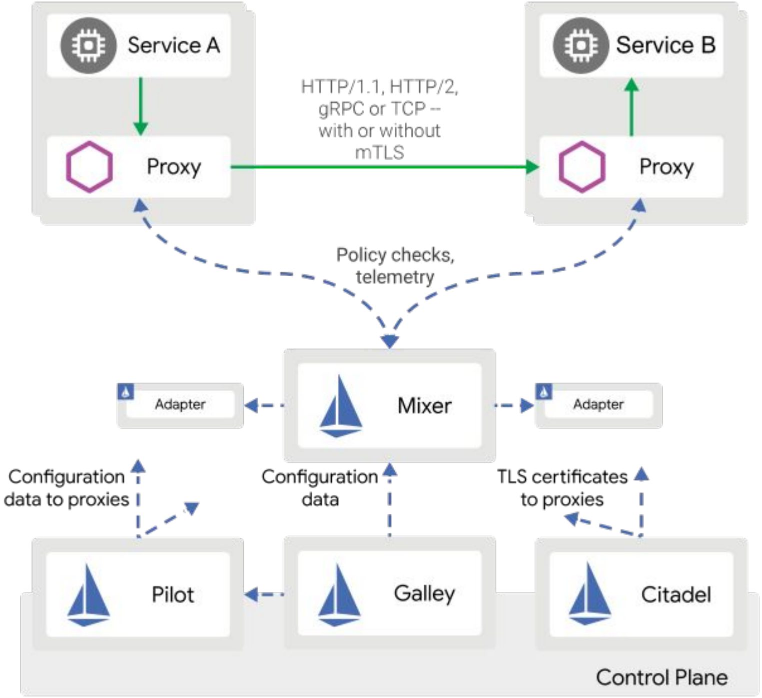
```
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
metadata:
  name: "to-fqdn"
spec:
  endpointSelector:
    matchLabels:
      app: test-app
  egress:
    - toEndpoints:
      - matchLabels:
          "k8s:io.kubernetes.pod.namespace": kube-system
          "k8s:k8s-app": kube-dns
      toPorts:
        - ports:
           - port: "53"
             protocol: ANY
          rules:
            dns:
              - matchPattern: "*"
    - toFQDNs:
        - matchName: "my-remote-service.com"
```

Cilium Network Policies



```
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
description: "enable empire-hq to produce to empire-announce and deathstar-plans"
metadata:
  name: "rule1"
spec:
  endpointSelector:
    matchLabels:
      app: kafka
  ingress:
  - fromEndpoints:
    - matchLabels:
        app: empire-hq
    toPorts:
    - ports:
      - port: "9092"
        protocol: TCP
      rules:
        kafka:
        - apiKey: "apiversions"
        - apiKey: "metadata"
        - apiKey: "produce"
          topic: "deathstar-plans"
        - apiKey: "produce"
          topic: "empire-announce"
```

Service Mesh



Testing

Tools:

- netassert - a security testing framework for fast, safe iteration on firewall, routing, and NACL rules for Kubernetes
- kube-bench - a Go application that checks whether Kubernetes is deployed securely by running the checks documented in the CIS Kubernetes Benchmark
- kube-hunter - scans for security weaknesses in Kubernetes clusters

Thank you! Any Questions?

Source