KubeCon | CloudNativeCon

Europe 2022

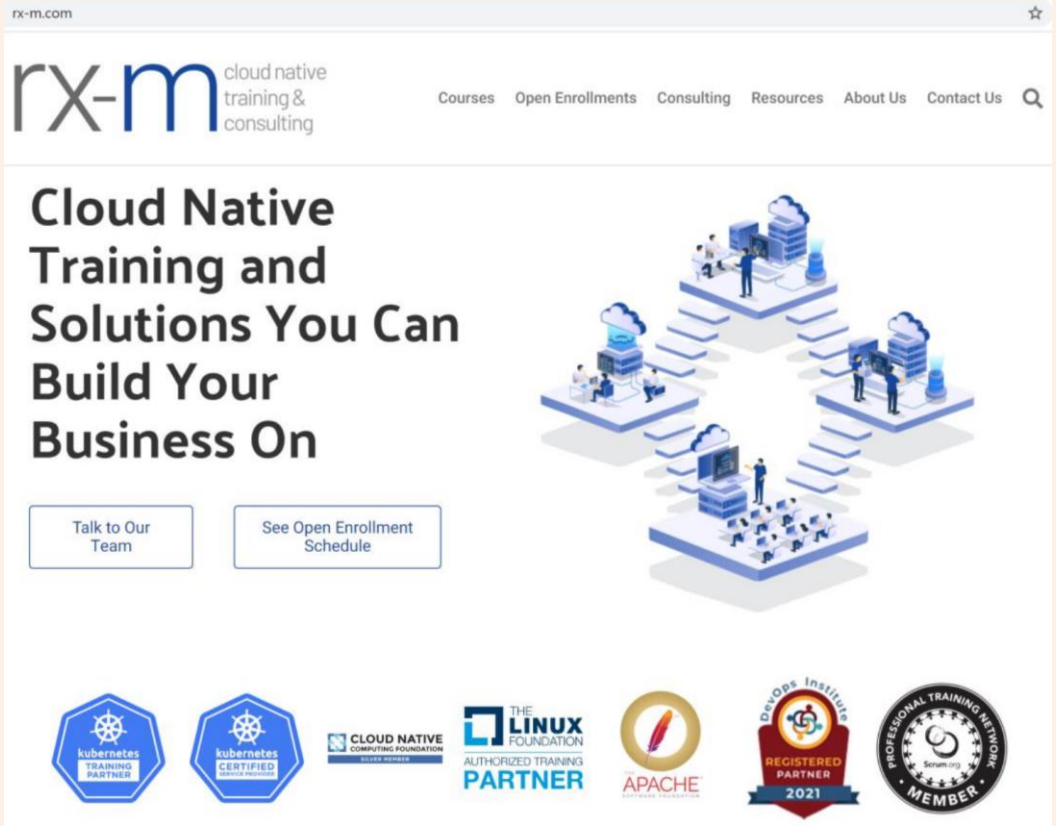
WELCOME TO VALENCIA!!

Kubernetes Networking 101

Randy Abernethy
Managing Partner, RX-M LLC

Presenter and Dutch Uncles

- @RandyAbernethy
  - Managing Partner at RX-M
  - CNCF Ambassador
- Cloud Native Uncles
  - Chris
    - “kubectl said what?” at 16:00 today!!
  - Iliyan
  - Valentin
- Tutorial Lab Doc:
  - https://github.com/RX-M/kubecon-eu-2022/net101.md



@rxmllc
Linkedin.com/company/rx-m-llc
https://www.youtube.com/channel/UCyFZuVfrRposGJ86mkWcF_Q

Kubernetes Networking 101

- **Goal:**
  - Introduce the world of Kubernetes network communications
  - … in a high level but practical way

- Concepts and Projects Explored:
  1. Container Networking
     - CNI and Cilium
  2. Kubernetes Services
     - Kubernetes
  3. Kubernetes DNS
     - CoreDNS
  4. Outside Access
     - Emissary, Envoy and MetalLB
  5. Service Mesh
     - Linkerd

CNI

cilium

kubernetes

envoy

CoreDNS

EMISSARY INGRESS

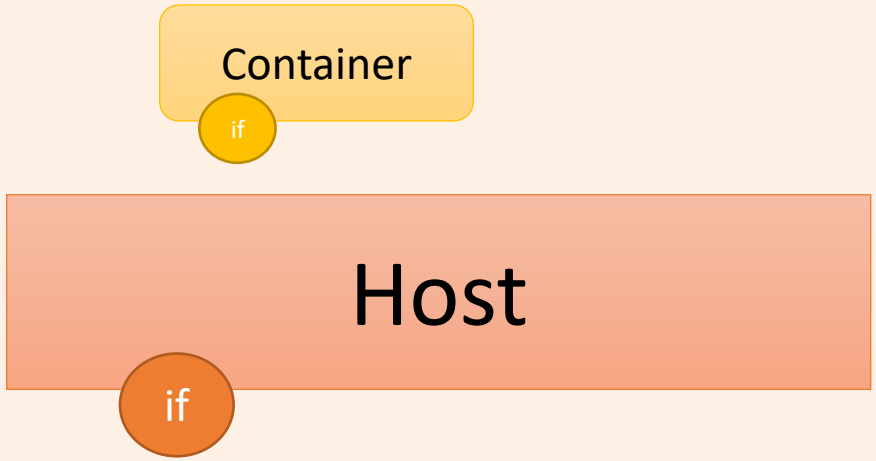
LINKERD

METALLB

1. Container Networking Concepts

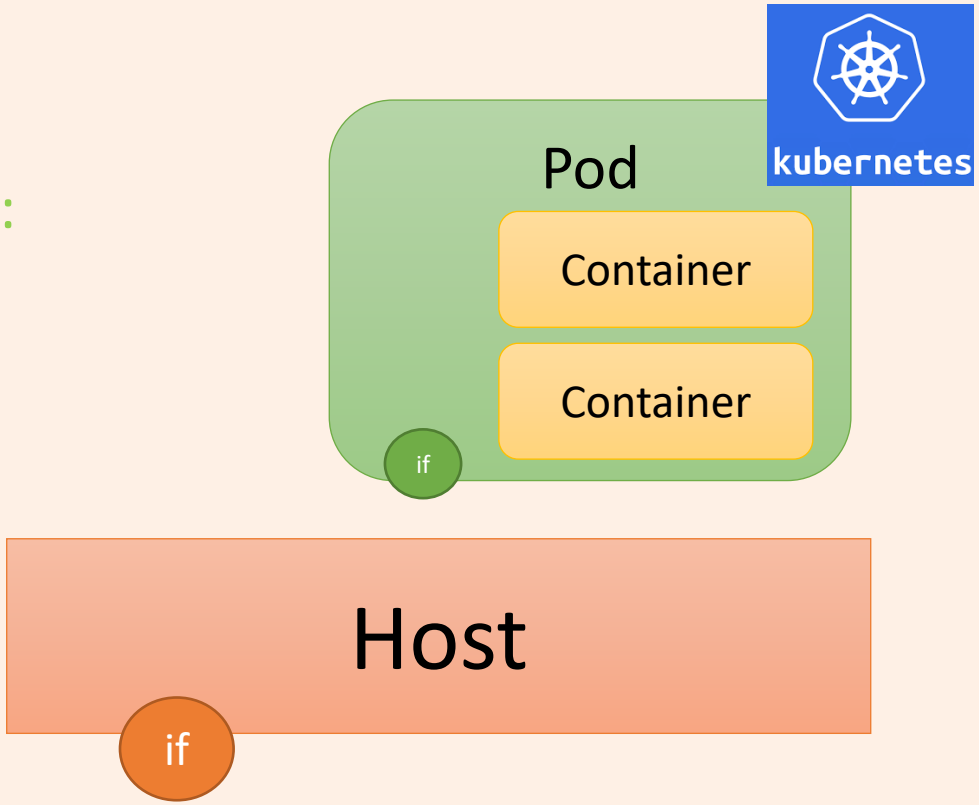
Container Networking

- There’s no such thing as a Container
  - There are just processes running on Linux
  - However, we can isolate a process using Linux namespaces
  - A network namespace gives a process and its children a virtual ip stack
- Network namespaces include private copies of:
  - Interfaces
    - loopback
    - eth0
  - Routes
  - IP Tables
  - And so on



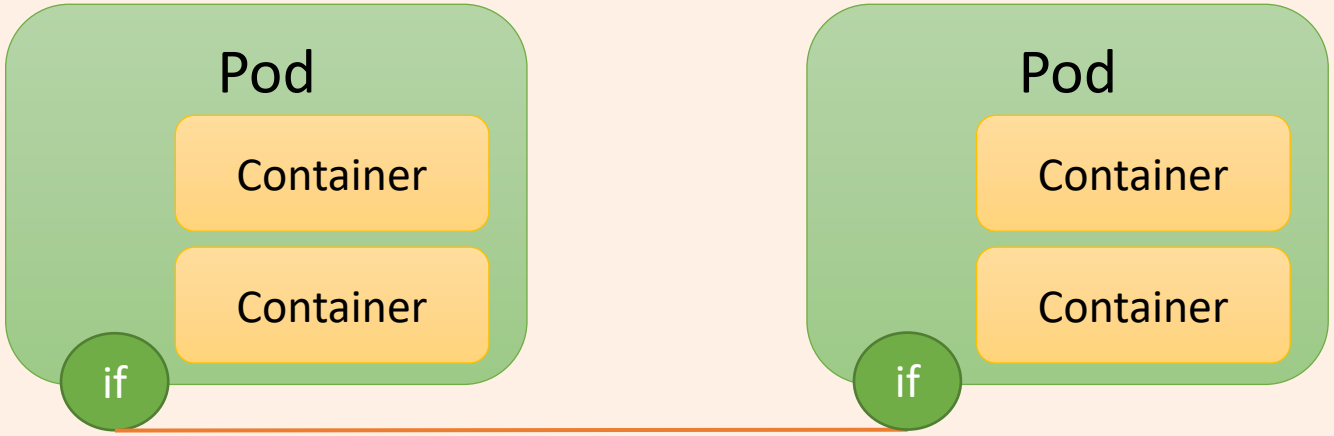
Pod Architecture

- Running a container under Docker places that container in its own private network namespace by default
- Kubernetes collects sets of containers together in Pods
  - Pods are atomic in Kubernetes:
    - Scheduled as a unit
    - Scaled as a unit
    - Terminated as a unit
- All the containers in a Pod share the same network namespace



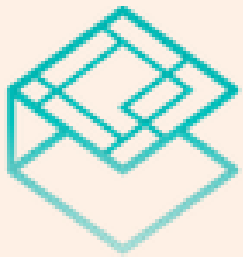
Pod Networking

- Components of distributed applications need to communicate
- When we deploy services in pods, the pods need a network
- We don’t really care if the pods are on the same machine or different machines, we just care that they can talk



CNI

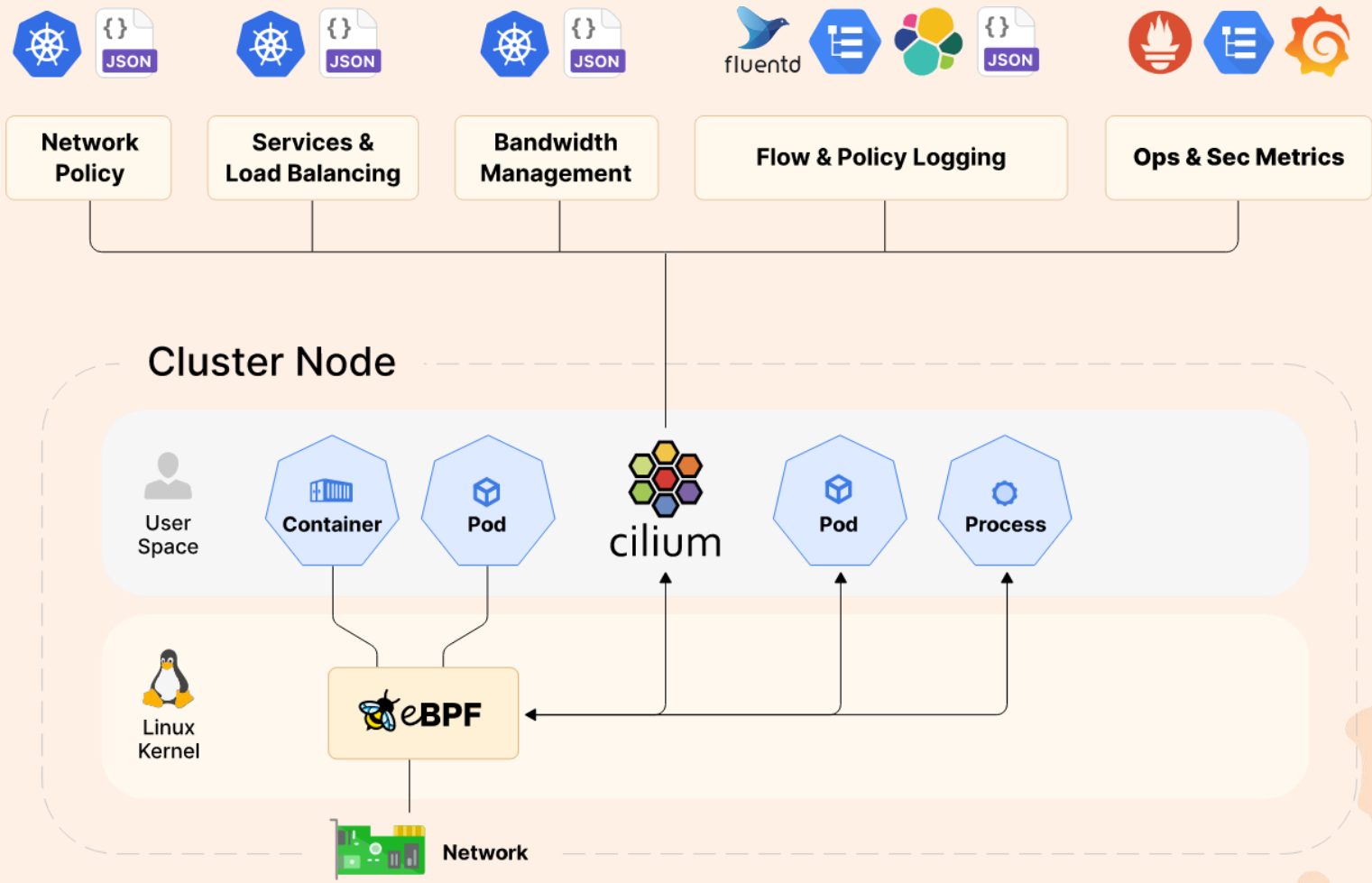
- Kubernetes relies on three key plugins for core functionality:
  - CRI – Container runtime interface
  - CNI – Container networking interface
  - CSI – Container storage interface
- CNI is a specification that allows systems like Kubernetes to integrate with software defined networking (SDN) solutions
- CNI plugins provide:
  - Pod network wiring
  - Pod IP addresses
  - Network policy implementation
  - Potentially much more



CNI

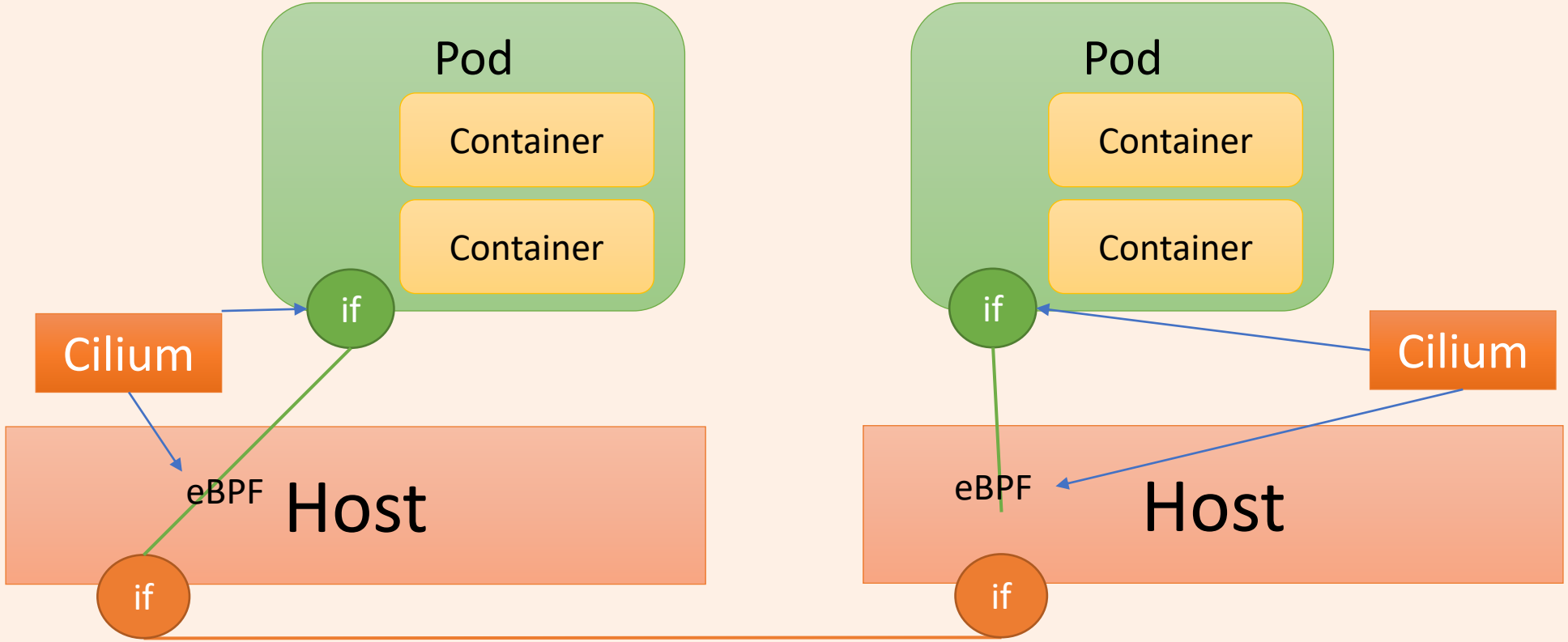
Cilium

- Cilium is an incubating CNCF project
- CNI compliant
- Provides Pod Networking features with the help of Linux eBPF



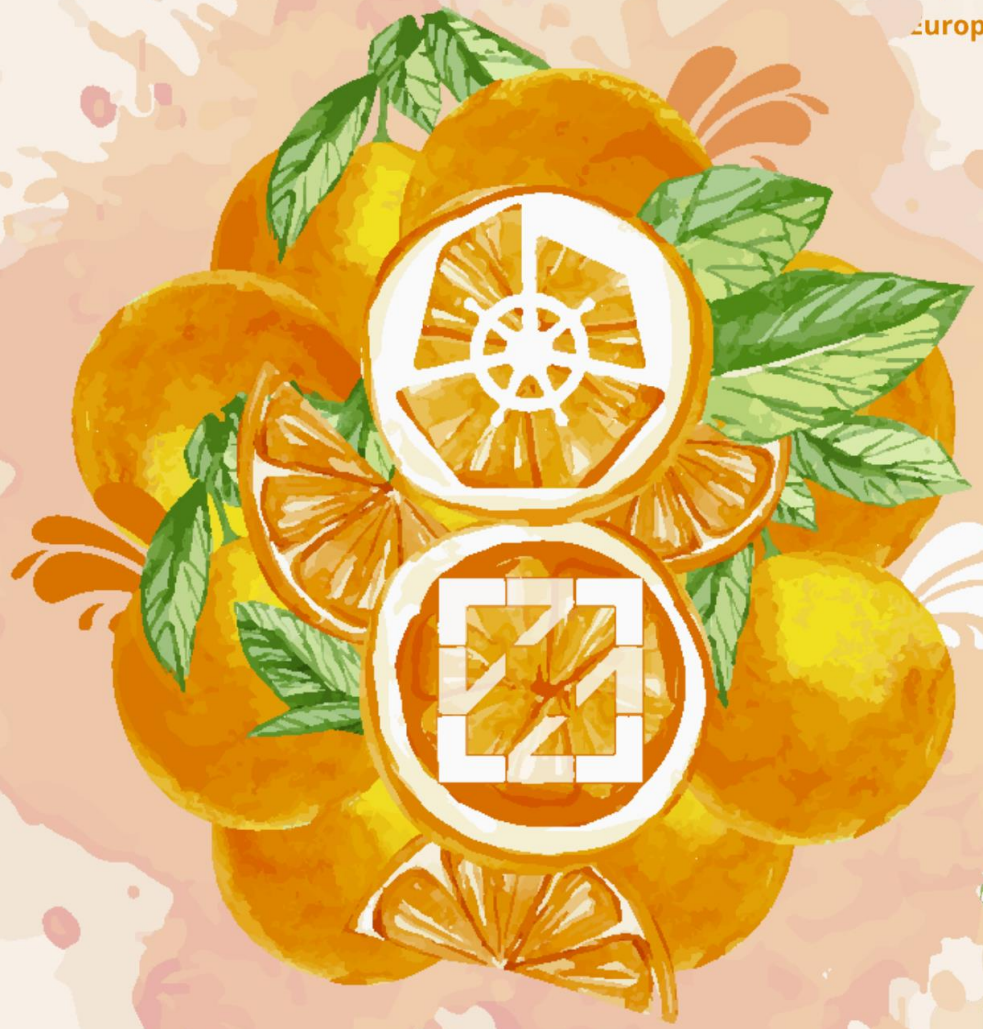
Cilium CNI

- CNI plugins are responsible for configuring a pod's network
  - Interfaces, ip addresses, routes and so on
  - Configuration external to the pod necessary to support intrapod networking



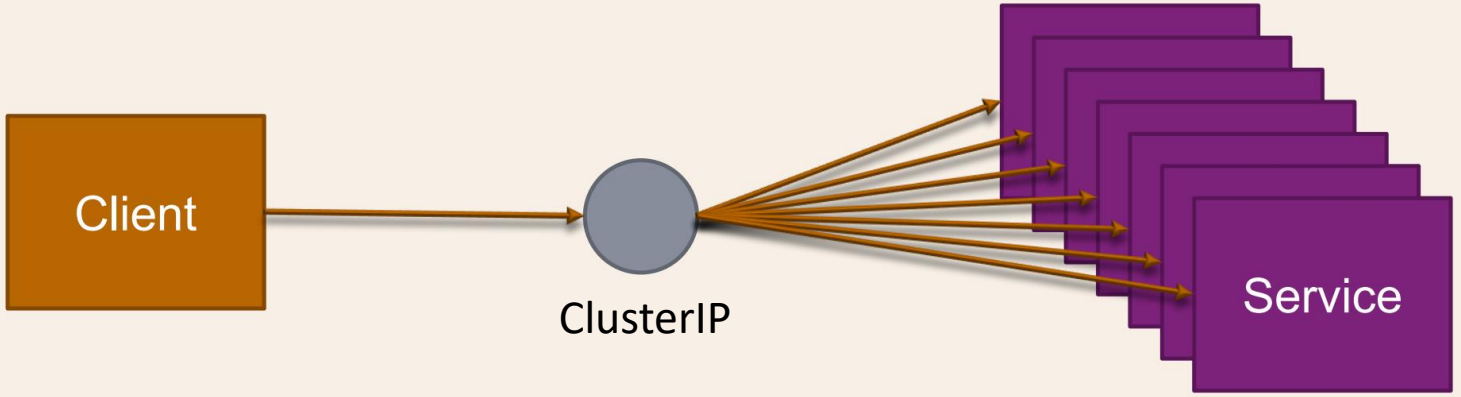
Lab Step 1 – Pod Networking

ssh to your lab system
Install a Kubernetes cluster
Install the Cilium CNI plugin
Explore the Pod network



2. Kubernetes Service Communications

Simple Kubernetes service communications



Cluster IP Implementation

- kube-proxy
  - User mode
  - IPTables
  - IPVS
- CNI Plugin
  - User mode
  - eBPF
  - Other approaches and combinations

Independent Address Spaces

- A Kubernetes solution requires several non overlapping address spaces
  - **HostIP** (node) Range – managed by IT or your Cloud
  - **ClusterIP** Range – configured when installing K8s
  - **PodIP** Range – configured when installing your CNI plugin

IPv4 and IPv6 Support

- The CNI controls Pod addressing (can be IPv4/6/dual/etc)
- Kubernetes Controls ClusterIP addressing:
  - IPv4-only GA in K8s v1.0
    - Only IPv4 Services
  - IPv6-only GA in K8s v1.18
    - If enabled, only IPv6 Services
  - Either IPv4/IPv6 in K8s 1.20
    - Either IPv4 or IPv6 services in the same cluster
  - Dual-stack IPv4/IPv6 GA in K8s 1.23 [two ranges must be supplied]
    - Services ipFamilyPolicy can be set to:
      - SingleStack – uses the first configured service cluster IP range
      - PreferDualStack – allows you to optionally define a single ClusterIP
      - RequireDualStack – requires you to define IPv4 & 6 if you define ClusterIP

Specifying a Service

- Services are a resource Kind
  - In the **Core** group at maturity **v1**
- Like all resources they require a name in the metadata
- Services have:
  - **Ports** to forward
  - **EndPoints** to forward to
- Selectors identify pods to generate endpoints from

```yaml
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  ports:
  - port: 80
    name: http
  selector:
    app: web
```

Endpoints

- Selectors automatically generate EndPoints for services
- EndPoints can be created manually as well

```yaml
apiVersion: v1
kind: Endpoints
metadata:
  name: website
subsets:
- addresses:
  - ip: "10.10.10.10"
  ports:
  - port: 80
    name: web
```

```
ubuntu@ip-172-31-24-84:~$ kubectl get endpoints website
NAME      ENDPOINTS          AGE
website   10.10.10.10:80     10h
ubuntu@ip-172-31-24-84:~$
```

Types of Services

- **Headless** *(special case of ClusterIP)*
  - Supports name resolution but not forwarding
- **ClusterIP**
  - For load balanced intra-cluster service communications
- **NodePort**
  - For external service access via a universal port
- **LoadBalancer**
  - Uses a plugin to enable an external load balancer
- **ExternalName**
  - Aliases this service to the specified externalName

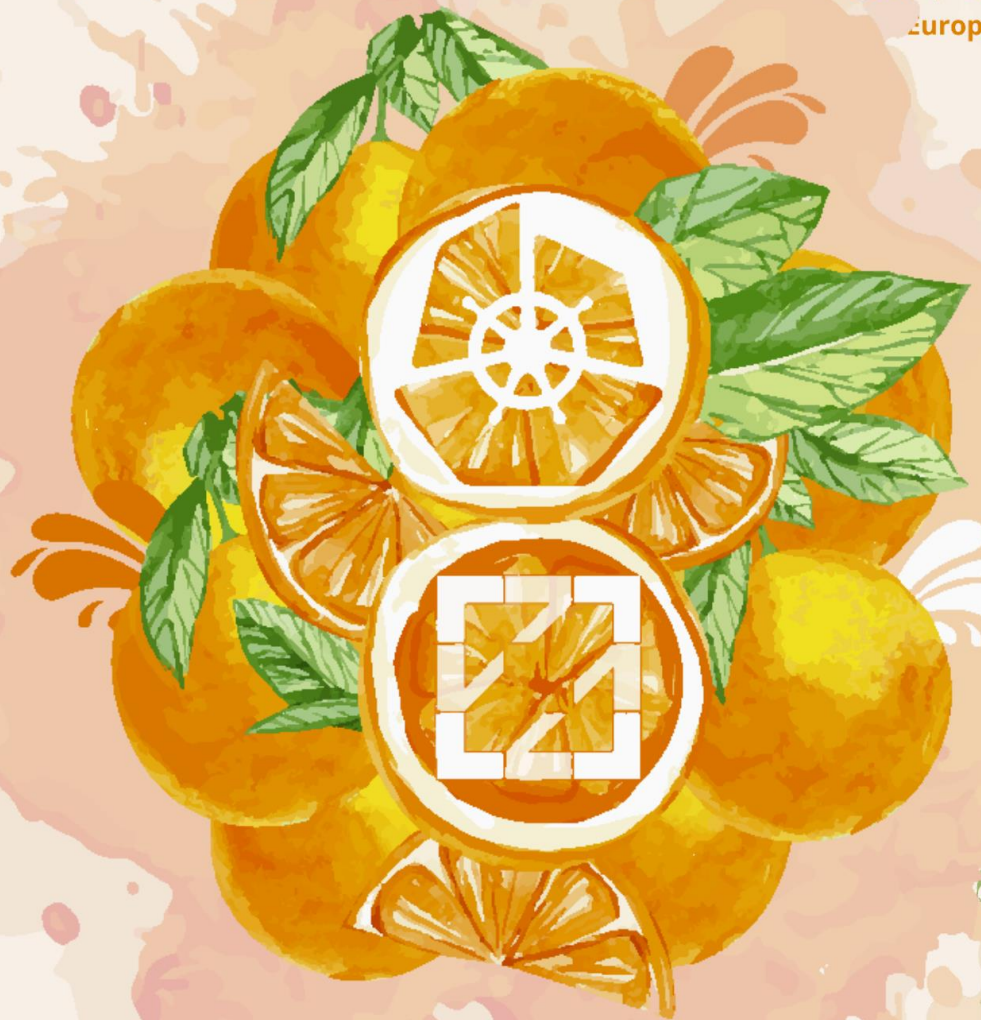
Lab Step 2 - Services

Create a Deployment

Create a ClusterIP Service

Figure out how it works

3. Kubernetes DNS



DNS in Kubernetes

- Pod Container Filesystem hacks:
  - /etc/resolv.conf
  - /etc/hostname
  - /etc/hosts
- CoreDNS
  - Deployment (ReplicaSet, 2 Pods)
    - AutoScaler in some distros
  - Service
    - 10.96.0.10

```
ubuntu@ip-172-31-24-84:~$ kubectl get all -n kube-system | grep dns
pod/coredns-6d4b75cb6d-fzsbw                 1/1     Running   0          45h
pod/coredns-6d4b75cb6d-wvnvv                 1/1     Running   0          45h
service/kube-dns   ClusterIP   10.96.0.10   <none>        53/UDP,53/TCP,9153/TCP   47h
deployment.apps/coredns            2/2     2            2           47h
replicaset.apps/coredns-6d4b75cb6d          2         2         2       47h
ubuntu@ip-172-31-24-84:~$
```

Service Name Resolution

- myservice.mynamespace.svc.cluster.local
  - **myservice** – the service name
  - **mynamespace** – the namespace
  - **svc** – the directory for services
  - **cluster.local** – the cluster suffix set when installing the cluster
- Realistic example:
  - web.production.svc.k8s54.rx-m.com
- Resolves to ClusterIP

Headless Services

- Used for StatefulSets where loadbalancing does not make sense
  - You need to talk to who you need to talk to, pods are not replicas
- Services with no ClusterIP are Headless
  - Resolving the Service name produces the list of endpoints for all pods in the service
  - You can also resolve a specific pod by ordinal
    - redis-0.redis.datans.svc.cluster.local
    - redis-1.redis.datans.svc.cluster.local
    - redis-2.redis.datans.svc.cluster.local
    - …

```yaml
apiVersion: v1
kind: Service
metadata:
  name: headlesswebsite
spec:
  ports:
  - port: 80
    name: http
  selector:
    app: website
  clusterIP: None
```

Overriding Kubelet DNS config

- Default resolv.conf settings can be overridden in Pod specs
  - Nameservers
  - Searches
  - Options

```yaml
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: dns-example
spec:
  containers:
    - name: test
      image: nginx
  dnsPolicy: "None"
  dnsConfig:
    nameservers:
      - 1.2.3.4
    searches:
      - ns1.svc.cluster-domain.example
      - my.dns.search.suffix
    options:
      - name: ndots
        value: "2"
      - name: edns0
```

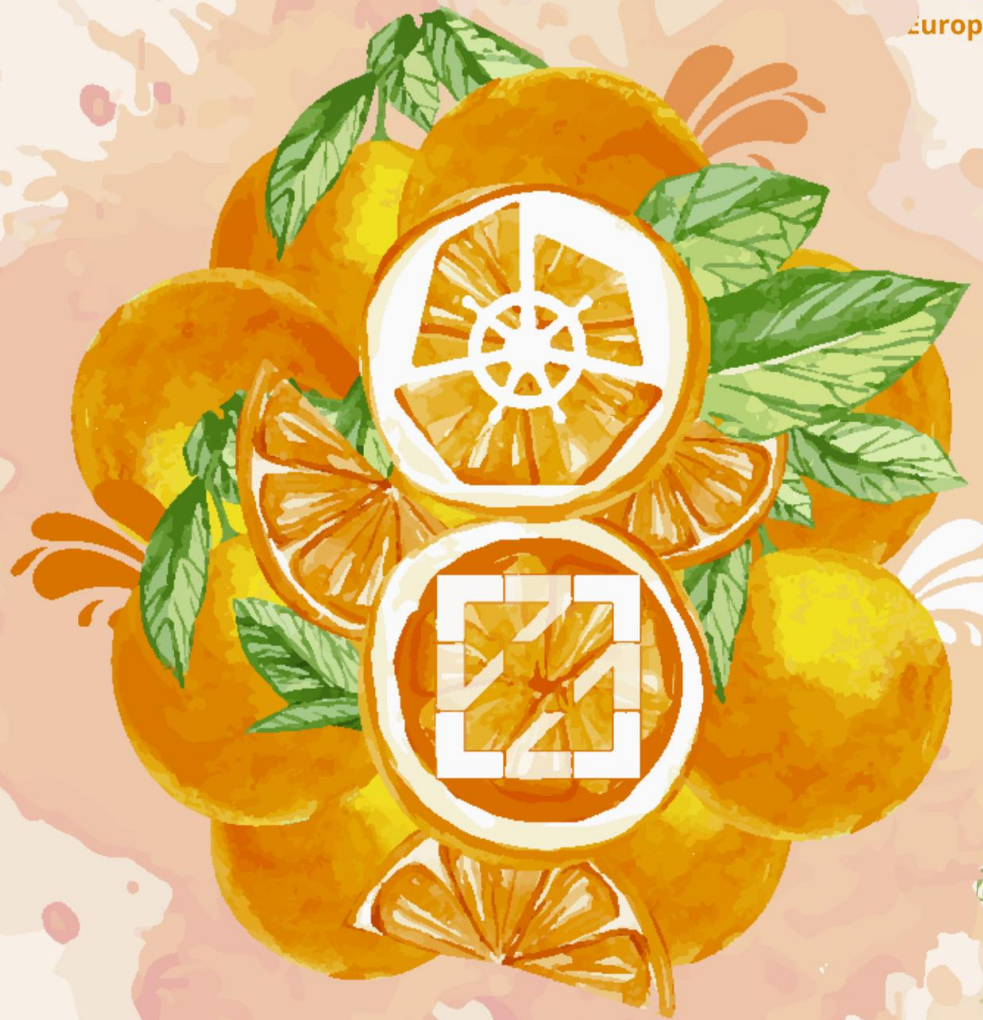
Lab Step 3 - DNS

Work with DNS

Create a Headless Service

Use Headless DNS

4. Accessing services from the Outside



Outside Access

- How can we reach services inside the cluster from outside the cluster?
  - HostPort – Pod feature that forwards a port from the host
  - NodePort – Service type that adds a NodePort forwarded on every node
  - LoadBalancer – Service type, calls a plugin to create an external load balancer
  - Ingress – Kubernetes framework for HTTP/HTTPS proxying
  - Gateway – Like ingress but more sophisticated
- Almost all schemes for inbound cluster access depend on either host ports or node ports

HostPort

- Pods can define host ports
- Maps a port on the worker node interface to the container interface port
- Useful for cluster admins
  - Typically in combination with DaemonSets
- Not good for applications
  - Deployments create pods that are scheduled, how do we know the port will be open on a given machine?
  - You have to know where the pod lands to reach it

```
apiVersion: v1
kind: Pod
metadata:
  name: hostportpod
spec:
  containers:
  - name: hp
    image: nginx
    ports:
    - containerPort: 80
      hostPort: 8080
```

NodePort

- Similar to host port in that it maps a port on the host
  - Unlike a host port, the service is assigned a unique high number port from an admin defined range
- The NodePort is forwarded on every node in the cluster to the service
- NodePort services also have all of the features of a ClusterIP service

```
apiVersion: v1
kind: Service
metadata:
  name: website
spec:
  type: NodePort
  ports:
  - port: 80
    nodePort: 31100
    name: http
  selector:
    app: website
```

LoadBalancer

- LoadBalancer services provision an external load balancer through a cluster plugin
  - Cloud Solutions
    - AWS Elastic Load Balancer
    - Azure Load Balancer
    - GCP Network Load Balancer
  - On Prem Solutions
    - MetalLB
    - Netris
    - KubeVIP
    - Avi (VMware)
    - F5
    - Citrix ADC
- LoadBalancer services also have all of the features of a NodePort service

```
apiVersion: v1
kind: Service
metadata:
  name: emissary-ingress
spec:
  ports:
  - name: http
    nodePort: 31211
    port: 80
    protocol: TCP
    targetPort: 8080
  - name: https
    nodePort: 31361
    port: 443
    protocol: TCP
    targetPort: 8443
  selector:
    app.kubernetes.io/name: emissary-ingress
  type: LoadBalancer
```

Ingress

- A Kubernetes framework
- Kubernetes defines the Ingress resource type
  - HTTP/HTTPS only
- An ingress controller must be installed to implement Ingress resource functionality

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: ambassador
  name: web-ingress
spec:
  ingressClassName: ambassador
  rules:
  - http:
      paths:
      - path: /engine
        pathType: Prefix
        backend:
          service:
            name: engine
            port:
              number: 80
```

Gateway

- Not a Kubernetes thing
  - Often an Ingress Controller on steroids
- Common features:
  - Advanced Security and Auth features
  - Sophisticated routing and load balancing
  - Support for other protocols
    - gRPC
    - SCTP
    - UDP
    - TCP
    - Apache Thrift
  - Protocol translation and upgrade
  - Uses CRDs for config

```
apiVersion: getambassador.io/v3alpha1
kind:  Mapping
metadata:
  name:  quote-backend
spec:
  prefix: /backend/
  service: quote
  circuit_breakers:
  - max_connections: 2048
    max_pending_requests: 2048
  add_request_headers:
    x-test-proto: "%PROTOCOL%"
    x-test-ip: "%DOWNSTREAM_REMOTE_ADDRESS_WITHOUT_PORT%"
    x-test-static: This is a test header
    x-test-static-2:
      value: The test header
    x-test-object:
      value: This the value
      append: False
```

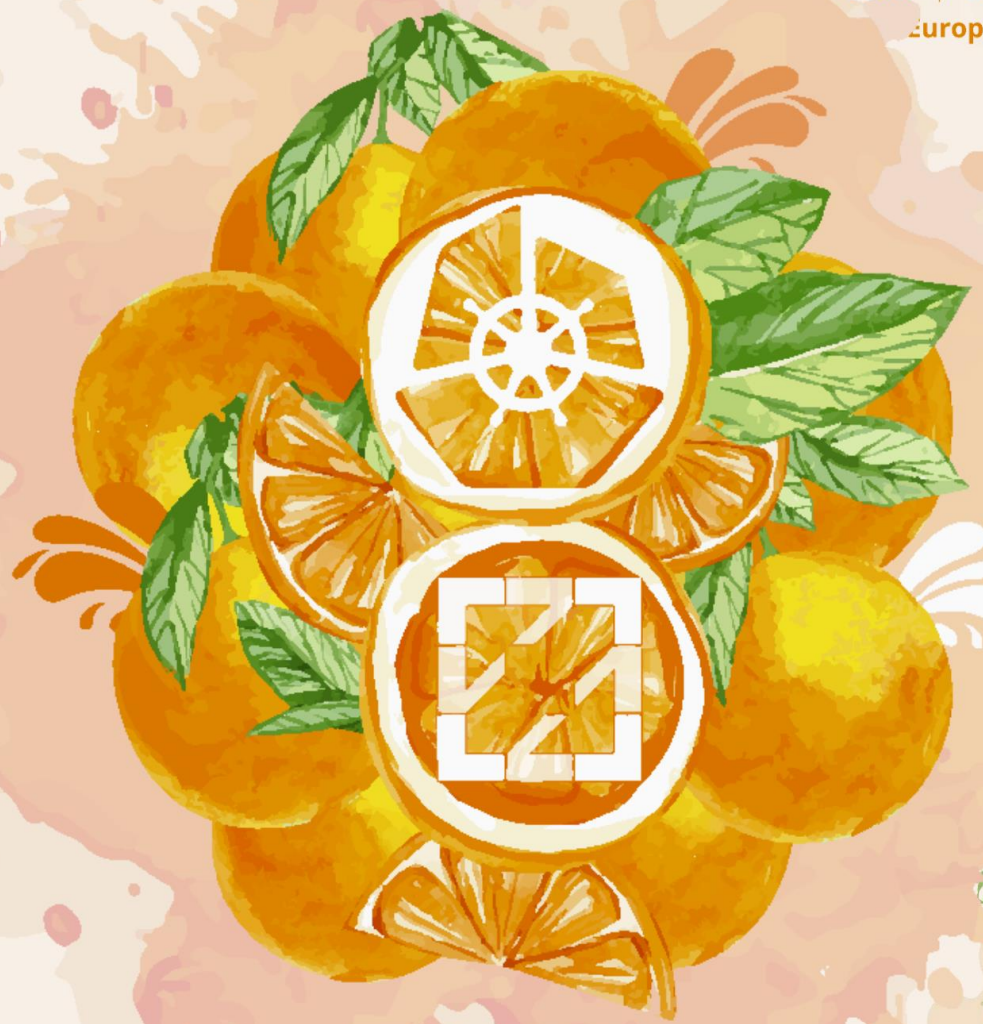
Lab Step 4

NodePort Services

Ingress

Gateways

5. Service Mesh

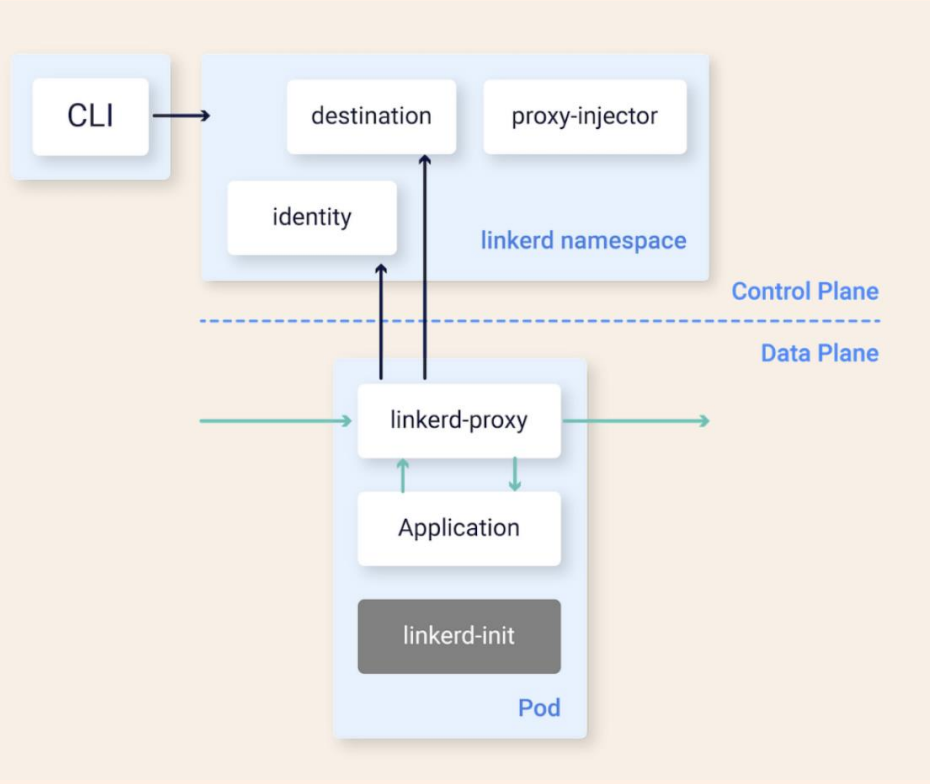


Service Mesh Functionality

- A Service Mesh can implement cross cutting concerns that are desired by all of your service
  - mTLS
  - Communications Metrics
  - Communications Policy
  - Traces
  - Fault Injection/Chaos support
  - Advanced Traffic Management
- A Service Mesh can also simplify cross cluster communications

Types of Service Mesh

- Proxy Based
  - Implemented using the ambassador pattern
  - Most common and most tested
- eBPF Based
  - Bleeding edge
  - Promises performance benefits but includes some downsides
- Library Based
  - Implemented in the app process by a library
    - e.g. gRPC
  - New, fast but requires all apps to use “the library”



The End

Many thanks for attending!

rx-m cloud native training & consulting

@RandyAbernethy
rx-m.com