# Services and MitM

Intercepting Traffic in a Kubernetes Cluster

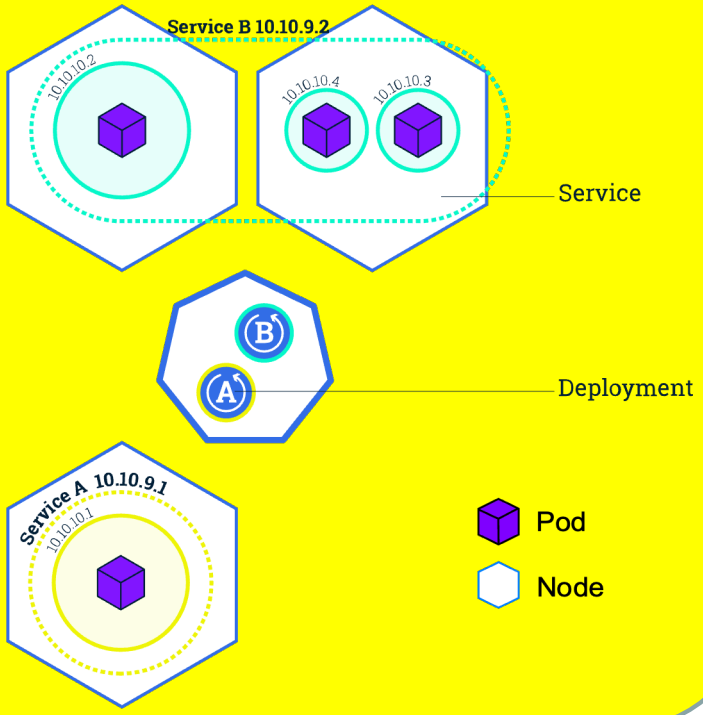
# Deeper Dive on Services

**Service: a load balancer**

**A service creates:**

- **a DNS name**
- **IP address**
- **port**

**These redirect traffic they receive to the pods that match the labels specified by the Service's description.**



# Service Manifest

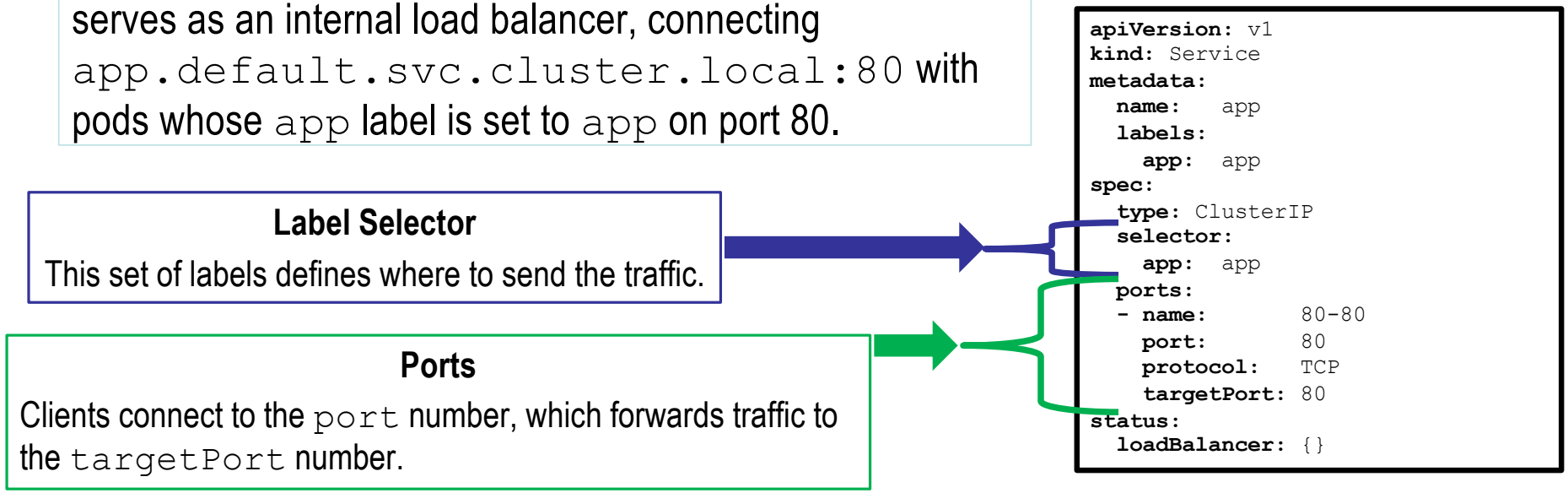
This manifest defines a `ClusterIP`-type service, which serves as an internal load balancer, connecting `app.default.svc.cluster.local:80` with pods whose `app` label is set to `app` on port 80.

**Label Selector**

This set of labels defines where to send the traffic.

**Ports**

Clients connect to the `port` number, which forwards traffic to the `targetPort` number.



```
apiVersion: v1
kind: Service
metadata:
  name:    app
  labels:
    app:   app
spec:
  type: ClusterIP
  selector:
    app:   app
  ports:
  - name:        80-80
    port:        80
    protocol:    TCP
    targetPort: 80
status:
  loadBalancer: {}
```

# Services: DNS Names

Services create a DNS name (A record) in DNS:

**app.default.**svc.cluster.local

Services also create SVC records for the named port:

_80-80._tcp.**app.default.**svc.cluster.local

```
apiVersion: v1
kind: Service
metadata:
  name:       app
  namespace: default
  labels:
    app:  app
spec:
  type: ClusterIP
  selector:
    app:  app
  ports:
  - name:        80-80
    port:        80
    protocol:    TCP
    targetPort: 80
status:
  loadBalancer: {}
```

# Services: Environment Variables

Services also create environment variables in all pods created in the same namespace after the service is instantiated.

```
APP_SERVICE_HOST=10.101.28.52
APP_SERVICE_PORT=80
APP_PORT=tcp://10.101.28.52:80
APP_PORT_80_TCP_ADDR=10.101.28.52
APP_SERVICE_PORT_80_80=80
APP_PORT_80_TCP_PORT=80
```

# Services: Types (ToC)

Services have four types:

- clusterIP: internal load balancer
- LoadBalancer: external load balancer (requires cloud-provided load balancers)
- NodePort: an open port on every node's external network interface
- ExternalName: a DNS mapping from an internal service to an external DNS entry
- ExternalIP: a redirection of an external IP to internal pods

# Services: ClusterIP

A `ClusterIP` connects cluster internal traffic.

DNS for `app.default.svc.cluster.local` will map to a virtual IP created for the service.

Traffic to that virtual IP, on the named port, will be sent to pods whose labels match the label selector.

Traffic will be sent to the named targetport.

```
apiVersion: v1
kind: Service
metadata:
  name:      app
  namespace: default
  labels:
    app:  app
spec:
  type: ClusterIP
  selector:
    app:  app
  ports:
  - name:       80-80
    port:       80
    protocol:   TCP
    targetPort: 80
status:
  loadBalancer: {}
```

# Services: Load Balancer

A Load Balancer works like a ClusterIP, but also allocates an external IP via a cloud provider load balancer.

```
kubectl create service loadbalancer lb \
--tcp=8000:80
```

```
apiVersion: v1
kind: Service
metadata:
  name: lb
  labels:
    app: lb
spec:
  type: LoadBalancer
  selector:
    app: lb
  ports:
  - name:       8000-80
    port:       8000
    protocol:   TCP
    targetPort: 80
```

# Services: NodePort

A NodePort works like a ClusterIP, but also allocates a port on every node's external IP address and forwards traffic from that IP:port pair to the virtual IP:port pair.

The external ports are automatically allocated from a specific range, which defaults to 30000-32767.

```
kubectl create service nodeport np \
--tcp=80:80
```

```
apiVersion: v1
kind: Service
metadata:
  name: np
  labels:
    app: np
spec:
  type: NodePort
  selector:
    app: np
  ports:
  - name:       80-80
    port:       80
    protocol:   TCP
    targetPort: 80
```

# Services: ExternalName

An ExternalName simply maps a service name (with its svc.cluster.local DNS name) to an external DNS name.

```
kubectl create service externalname cnn \
--external-name www.cnn.com
```

```
apiVersion: v1
kind: Service
metadata:
  name:  cnn
  labels:
    app: cnn
spec:
  type:  ExternalName
  selector:
    app:          cnn
  externalName: www.cnn.com
status:
  loadBalancer: {}
```

# Services: ExternalIP

An ExternalIP service instructs all nodes (via kube-proxy) to redirect traffic that is both:

- destined to an IP address in the `externalIPs` list
- destined to a protocol-port pair in the `ports` list

The traffic is directed to pods specified by the `selector`.

```
apiVersion: v1
kind: Service
metadata:
  name:          replacement-dns
spec:
  type:          ExternalIP
  selector:
    app:         mirror-site
  externalIPs:
               - 8.8.4.4
               - 8.8.8.8
  ports:
  - name:        dns
    protocol:    UDP
    port:        53
    targetPort:  53
```

# Exercise: DEF CON 29 Capture the Flag

Please, open the Firefox browser on the class machine to:

http://localhost:10000/exercises/kubernetes-ctf-dc2021