Kubernetes from Basic to Advanced

kubernetes

Session #04
Networking & Services

kubernetes

Session Contents

- Networking
- Services
- ClusterIP
- NodePort
- LoadBalancer
- Ingress

kubernetes

Networking

Kubernetes Network Model

- Every Pod in a cluster gets its own unique cluster-wide IP address
  - Each container inside the pod shares their network namespace
  - Means all containers shares the same ports
- Do not need to explicitly create links between Pods
- Almost never need to deal with mapping container ports to host ports
- This creates a clean, backwards-compatible model where Pods can be treated much like VMs or physical hosts from the perspectives of port allocation, naming, service discovery, load balancing, application configuration, and migration

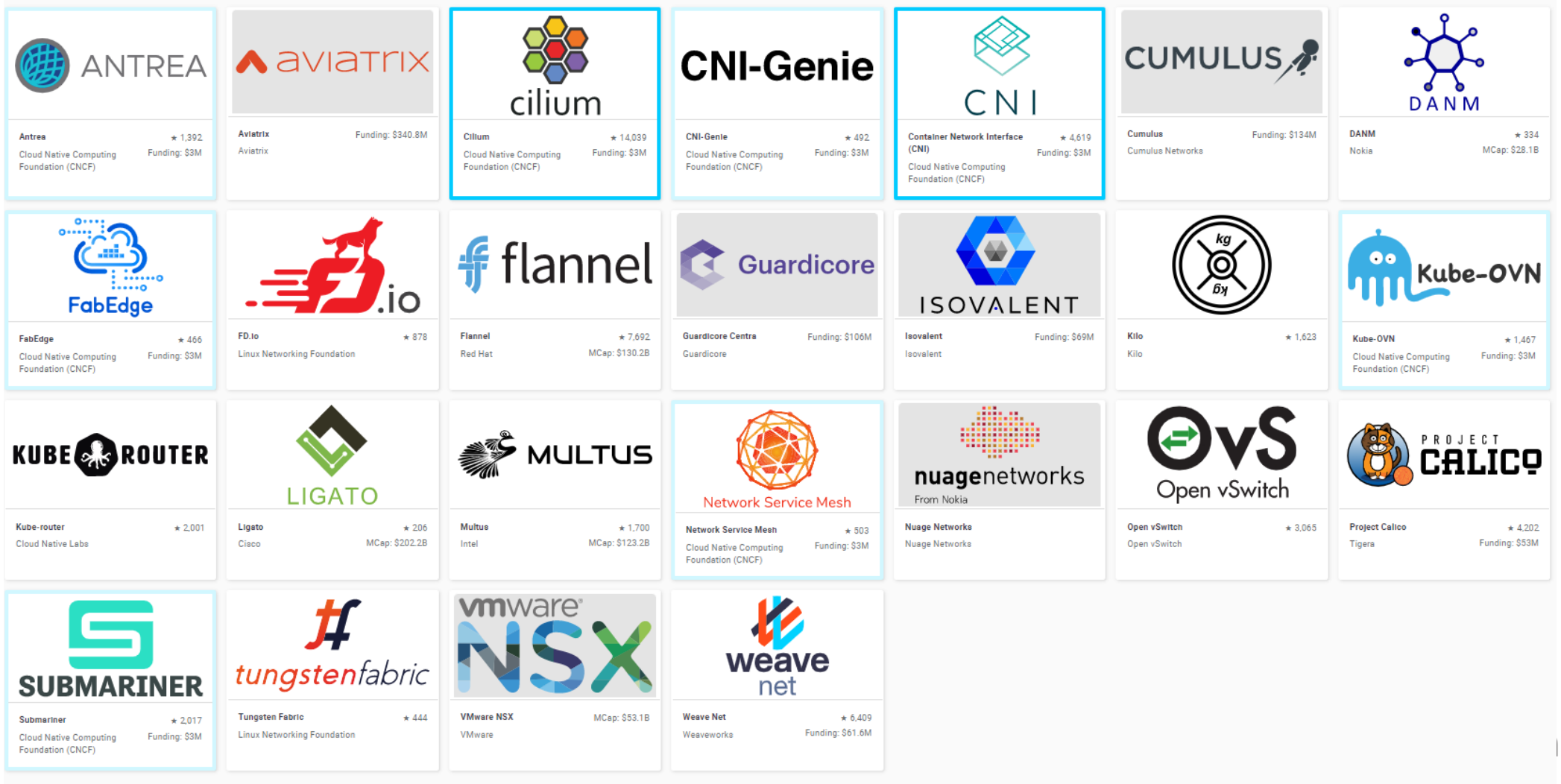
kubernetes

Kubernetes Network Providers

- Kubernetes uses the concept of network plugin that can be implemented using different providers
- A network implementation needs to follow these two requirements
  - Pods can communicate with all other pods on any other node without NAT
  - Agents on a node (e.g. system daemons, kubelet) can communicate with all pods on that node

kubernetes

Kubernetes Network Providers



etes

Service

Services: Motivation

- Kubernetes Pods are created and destroyed to match the state of your cluster making them ephemeral
- Each Pod gets its own IP address, however in a Deployment, the set of Pods running in one moment in time could be different from the set of Pods running that application a moment later
- If some set of Pods provides functionality to other Pods inside your cluster, how do they find out and keep track of which IP address to connect to?

kubernetes

What is a Service?

- An abstraction that defines a logical set of loosely-coupled pods and a policy by which to access them as a network service.
- Use selectors to define which pods to include.
- Every type of service (unless ExternalName) load balances traffic to Pods (Layer 4)
- Preferrable way to expose Pods to other Pods within the cluster
- Maps an external and global port to target (container) ports

kubernetes

What is a Service?

- Kubernetes automatically updates which Pods are available to which service by creating Endpoints objects
- Exists 3 types of Services
  - ClusterIP
  - NodePort
  - LoadBalancer
- Exists one additional named ExternalName that to create an internal service to redirect to another service
  - Mostly used to create a proxy server to an external service that may be changes between environments

kubernetes

Service Manifest

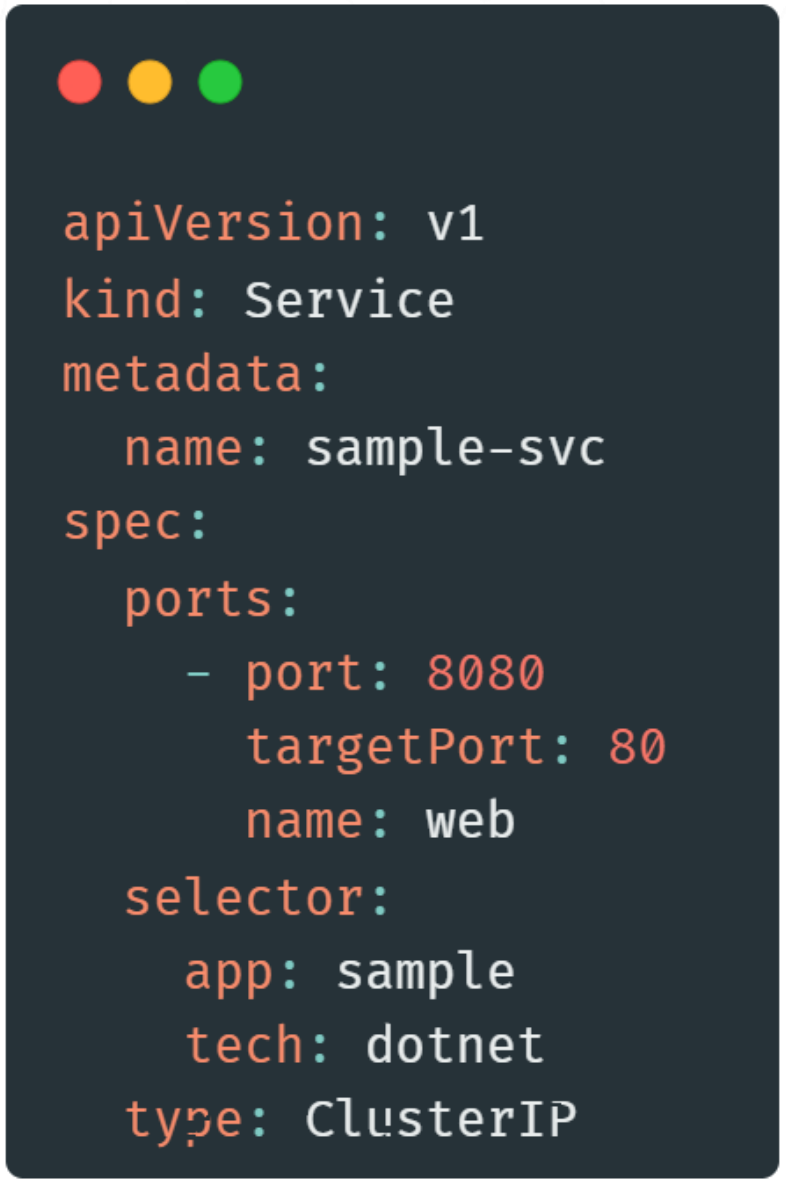
Service properties

port sets service port

targetPort maps pod port

selector defines pods selector

type defines service type



Endpoint Resource

Endpoint properties

Each block defines pod settings

Each block defines pod settings

Each block defines pod settings

`ports` defines service port

```yaml
apiVersion: v1
kind: Endpoints
metadata:
  name: sample-svc
subsets:
- addresses:
  - ip: 10.1.0.177
    nodeName: docker-desktop
    targetRef:
      kind: Pod
      name: sample-dep-8966bc4c5-67ngv
  - ip: 10.1.0.178
    nodeName: docker-desktop
    targetRef:
      kind: Pod
      name: sample-dep-8966bc4c5-nrfpp
  - ip: 10.1.0.179
    nodeName: docker-desktop
    targetRef:
      kind: Pod
      name: sample-dep-8966bc4c5-92ts6
  ports:
  - name: web
    port: 80
    protocol: TCP
```

Load Balancing

- Depends on kube-proxy configuration mode
- User space proxy mode
  - Uses round-robin algorithm to select pods
- Iptables proxy mode (default)
  - Uses random selection of pods
- IPVS proxy mode
  - rr: round-robin
  - lc: least connection (smallest number of open connections)
  - dh: destination hashing
  - sh: source hashing

kubernetes

Service Discovery

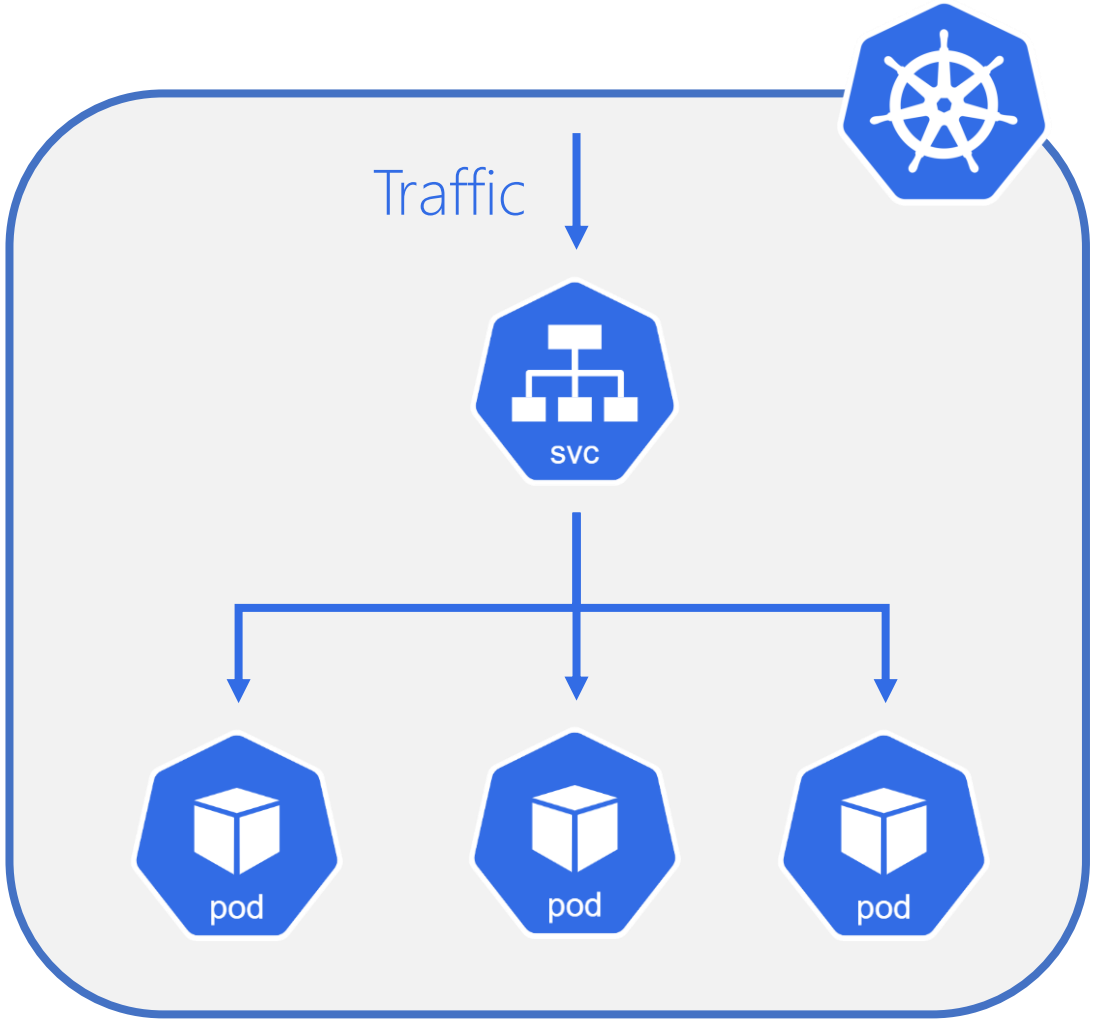
- Any cluster have an internal service to allow service discovery
- This service is implemented using a plugin strategy
- Most used is CoreDNS, an opensource DNS server
- Every cluster have a global ClusterIP service named **`kube-dns`** with endpoints to plugin pods
- Every pod is configured with **`kube-dns`** service IP as nameserver to name resolution

kubernetes

ClusterIP

ClusterIP

- Exposes the Service on a cluster-internal IP
- Choosing this value makes the Service only reachable from within the cluster
- Default service type
- Service name can be used as DNS
  - http://svc-name
  - http://svc-name.ns.svc.cluster.local



ClusterIP Manifest

Service properties

port sets service port

targetPort maps pod port

selector defines pods selector

type defines service type

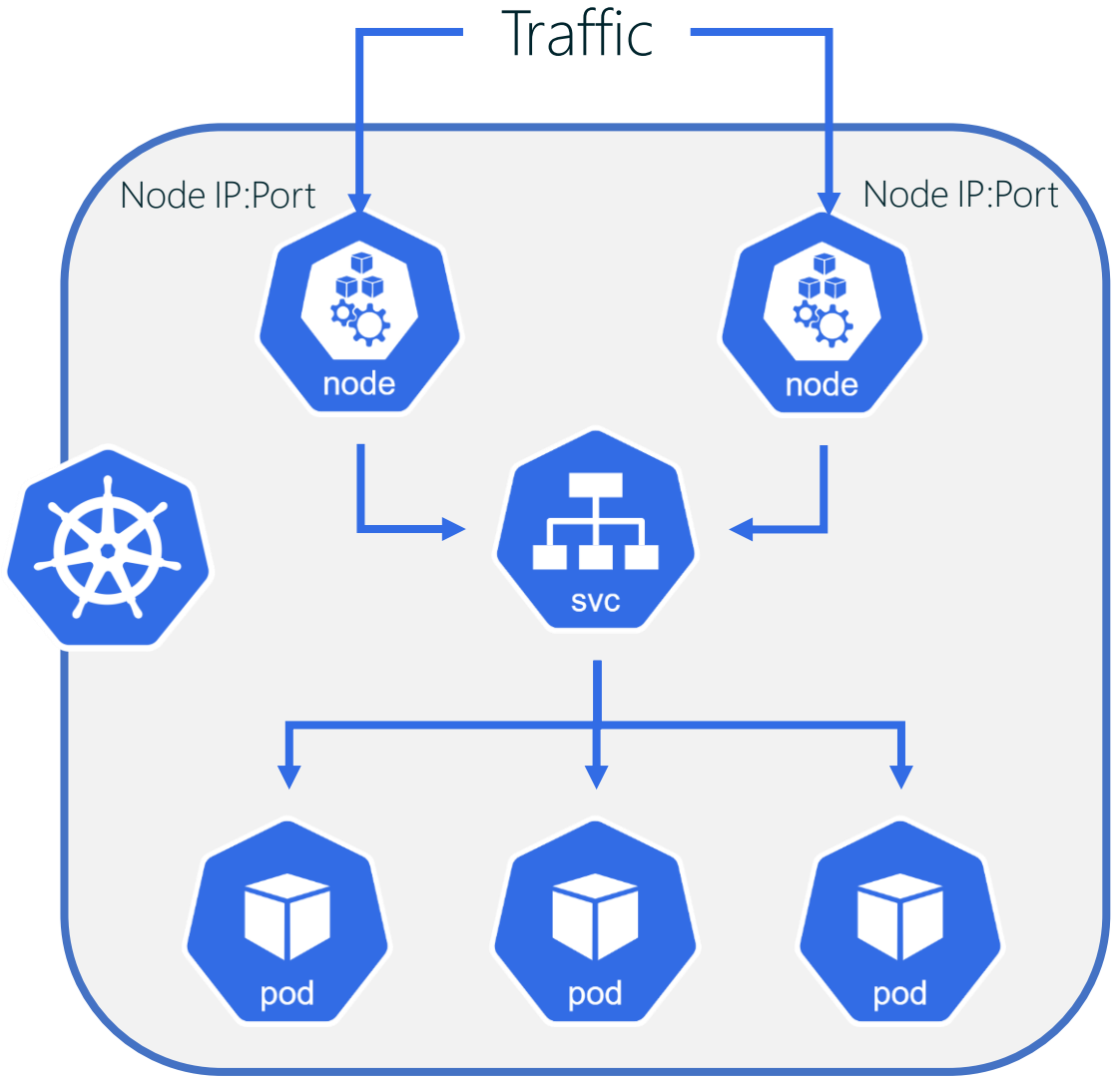
```yaml
apiVersion: v1
kind: Service
metadata:
  name: sample-svc
spec:
  ports:
    - port: 8080
      targetPort: 80
      name: web
  selector:
    app: sample
    tech: dotnet
  type: ClusterIP
```

Demo | ClusterIP

NodePort

NodePort

- Exposes the Service on each Node's IP at a static port (the NodePort)
- A ClusterIP Service, to which the NodePort Service routes, is automatically created.
- You'll be able to contact the NodePort Service, from outside the cluster, by requesting **`<NodeIP>:<NodePort>`**
- If **`nodePort`** property is not specified, is automatically set a port from the range 30000-32767



NodePort Manifest

Service properties

port sets service port

targetPort maps pod port

nodePort maps node port

selector defines pods selector

type defines service type

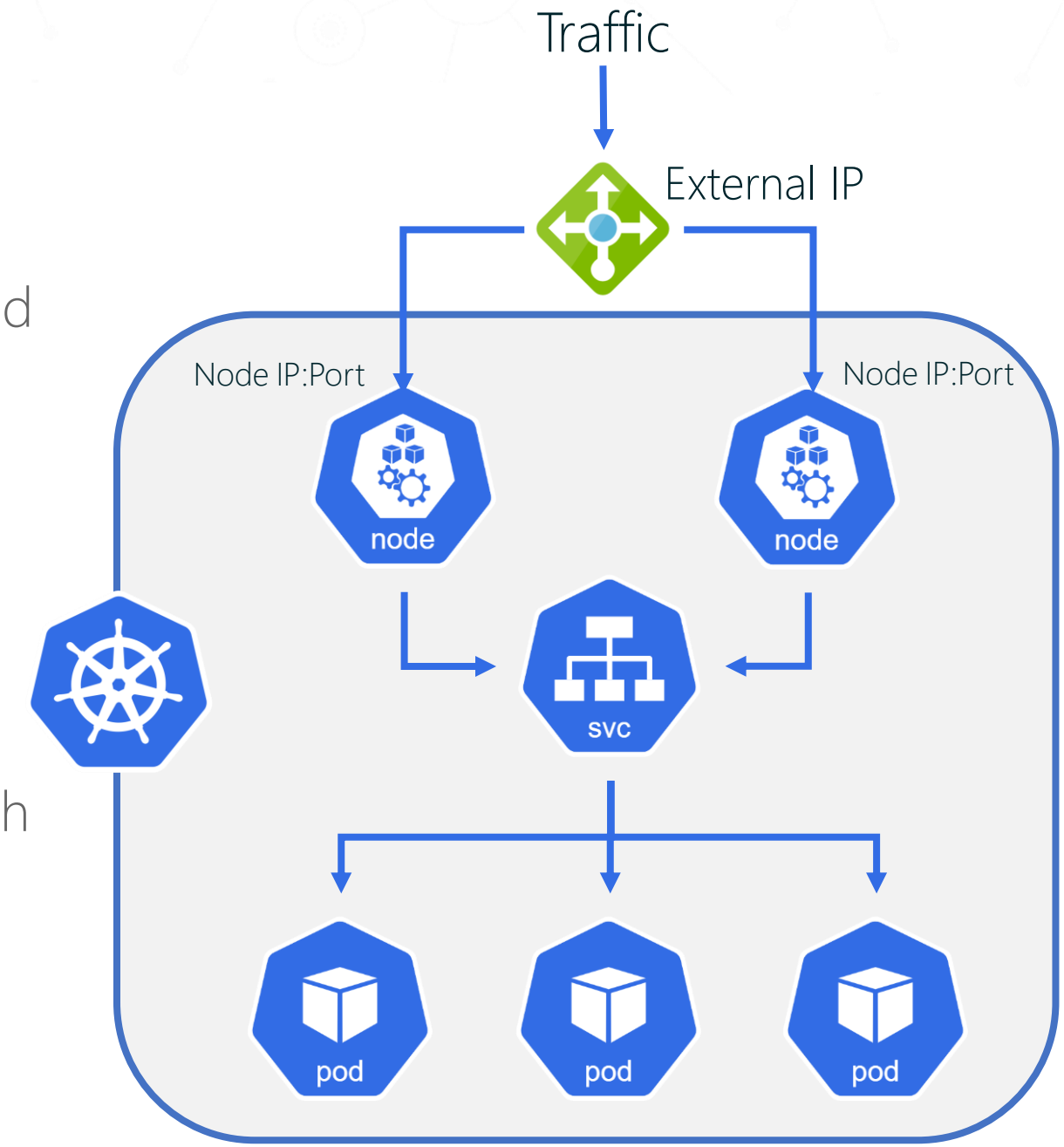
```yaml
apiVersion: v1
kind: Service
metadata:
  name: sample-node-svc
spec:
  ports:
    - port: 9100
      targetPort: 80
      nodePort: 31000
      name: web
  selector:
    app: sample
    tech: dotnet
  type: NcdePort
```

Demo | NodePort

LoadBalancer

LoadBalancer

- Exposes the Service externally using a cloud provider's load balancer
- NodePort and ClusterIP Services, to which the external load balancer routes, are automatically created
- On-prem needs manual configuration
- ExternalIP means an IP External related with cluster. No needs to be an Internet public IP



LoadBalancer Manifest

Service properties

port sets service port

targetPort maps pod port

selector defines pods selector

type defines service type

```yaml
apiVersion: v1
kind: Service
metadata:
  name: sample-lb-svc
spec:
  ports:
    - port: 8000
      targetPort: 80
      name: web
  selector:
    app: sample
    tech: dotnet
  type: LoadBalancer
```

Demo | LoadBalancer

Ingress

Motivation

- When need to expose HTTP/HTTPS endpoint outside of the cluster, you can use LoadBalancer services
- That may overload the use of public (external IPs) and need several additional components to implement this service
- Using services, you are only exposing on Layer 4 and when using HTTP/HTTPS is more functional to work on Layer 7
- Security and configuration wise is important to have only one place to access externally the cluster (DNS configuration, etc.)

kubernetes

What is Ingress?

- Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster
- Traffic routing is controlled by rules defined on the Ingress resource
- An Ingress may be configured to give Services externally-reachable URLs, load balance traffic, terminate SSL / TLS, and offer name-based virtual hosting
- An Ingress does not expose arbitrary ports or protocols. To expose services other than HTTP and HTTPS to the internet you need to use NodePort or LoadBalancer services

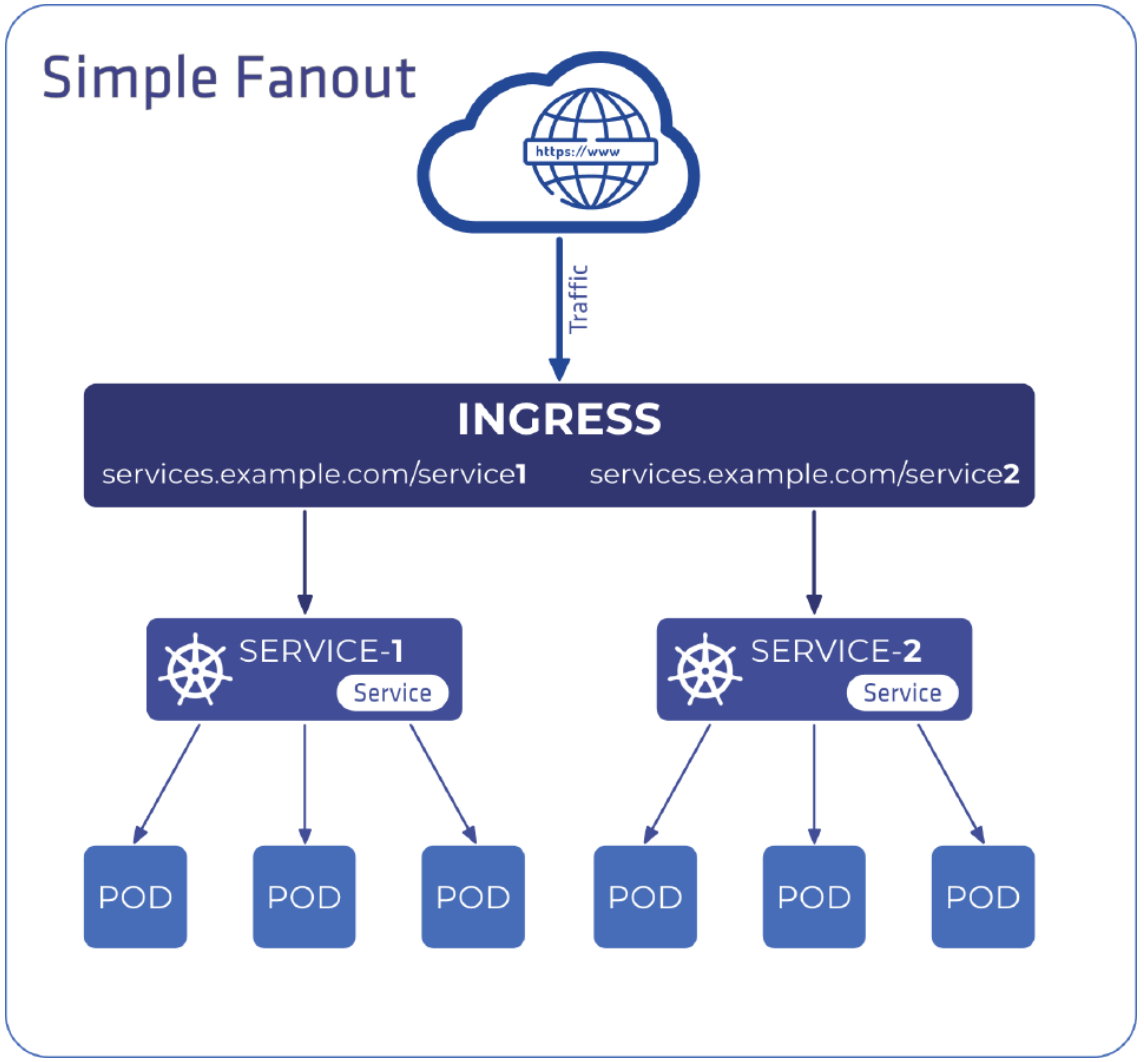
kubernetes

What is Ingress?

- Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster
- Traffic routing is controlled by rules defined on the Ingress resource
- An Ingress may be configured to give Services externally-reachable URLs, load balance traffic, terminate SSL / TLS, and offer name-based virtual hosting
- An Ingress does not expose arbitrary ports or protocols. To expose services other than HTTP and HTTPS to the internet you need to use NodePort or LoadBalancer services

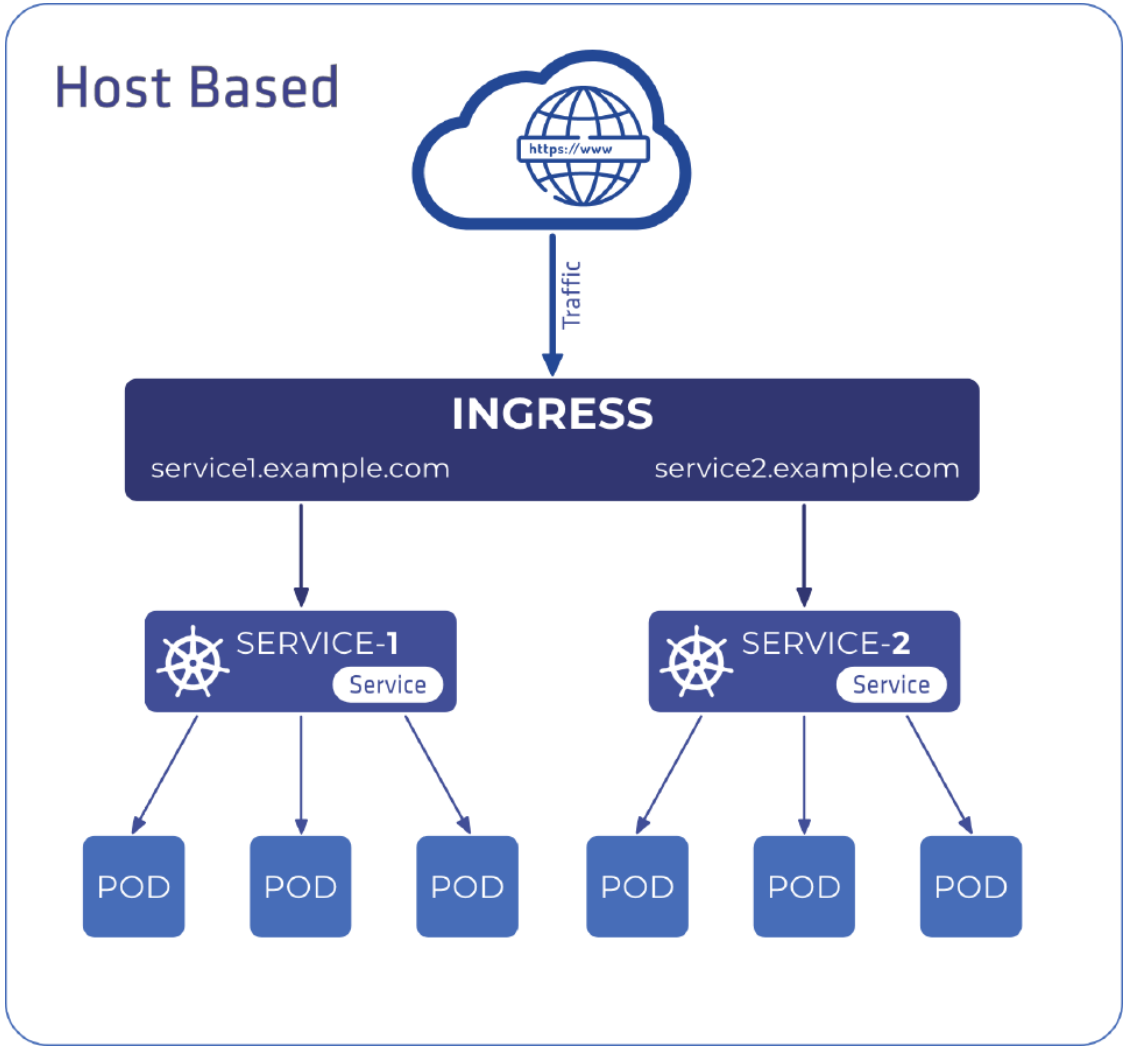
kubernetes

Ingress rule using URI



kubernetes

Ingress rule based on Host



kubernetes

Ingress Controller

- An Ingress is just a resource defining rules. This resource needs a controller to implement those rules
- Kubernetes don’t have a native Ingress Controller and must be installed by cluster administrator
- An ingress controller is mandatory to handle Ingress resource. Without it, Ingress resource doesn’t do anything

kubernetes

Ingress Controller

- Most used ingress controller
  - Nginx ingress controller
  - Contour ingress controller
  - HAProxy ingress controller
  - Traefik ingress controller
  - Kong Ingress Controller
  - ingress-gce (Google Cloud only)
  - aws-load-balancer-controller (AWS only)
  - application-gateway-kubernetes-ingress (Azure only)

kubernetes

Ingress Manifest

Ingress properties

Host sets main URL

path defines URI for matching

pathType defines how matching is performed (prefix or exact)

service block defines backend service to redirect

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: simple-fanout-example
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        pathType: Prefix
        backend:
          service:
            name: service1
            port:
              number: 4200
      - path: /bar
        pathType: Prefix
        backend:
          service:
            name: service2
            port:
              number: 8080
```

Questions?

kubernetes

Lab #03: Deployment lifecycle