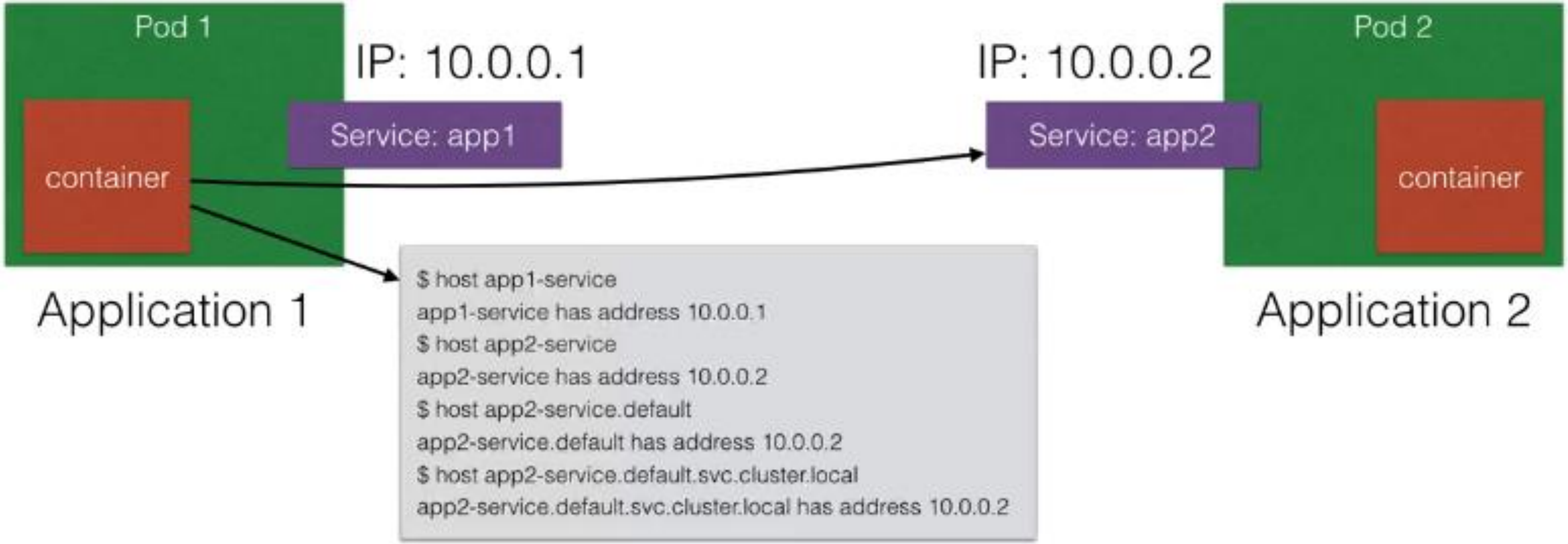
DNS

- As of Kubernetes 1.10, DNS is a **built-in** service launched automatically using addon manger
- The DNS Service can be used within pods to find other service running on the same cluster
- Multiple containers **within 1 pod** don’t need this service, as they can contact each other **directly**
- A container in the same pod can **connect / communicate** to other containers directly using **localhost:port**

DNS

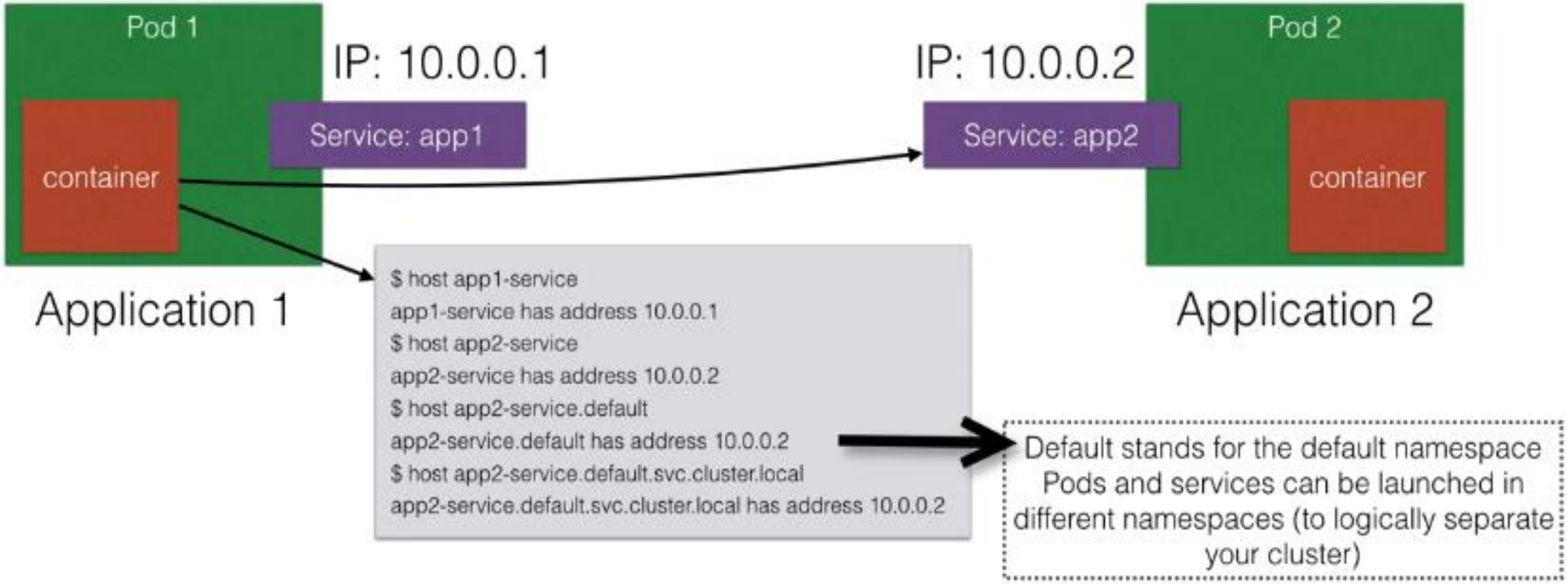
- An example of how app 1 could reach app 2 using DNS:



```
$ host app1-service
app1-service has address 10.0.0.1
$ host app2-service
app2-service has address 10.0.0.2
$ host app2-service.default
app2-service.default has address 10.0.0.2
$ host app2-service.default.svc.cluster.local
app2-service.default.svc.cluster.local has address 10.0.0.2
```

DNS

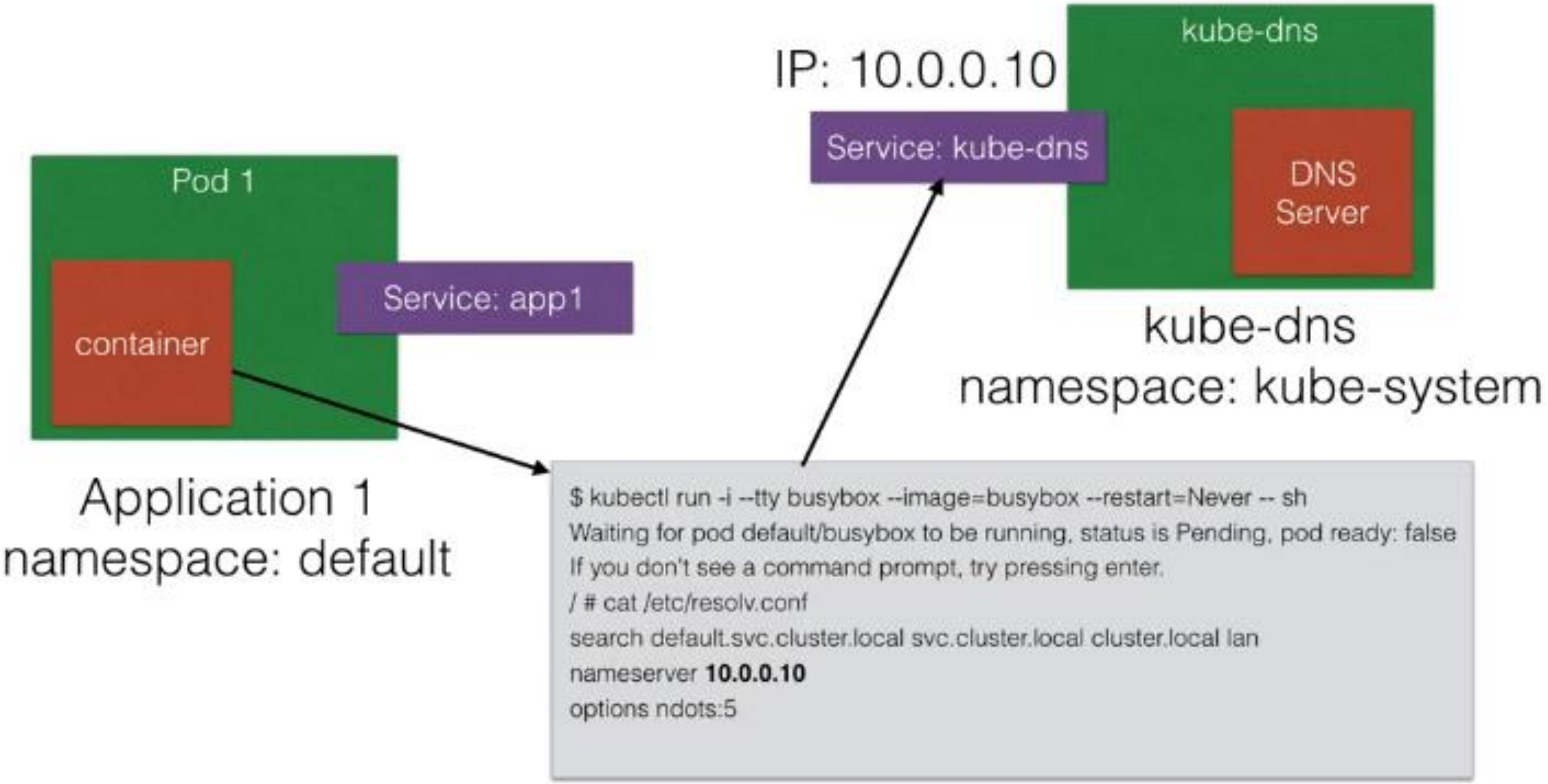
- An example of how app 1 could reach app 2 using DNS:



```
$ host app1-service
app1-service has address 10.0.0.1
$ host app2-service
app2-service has address 10.0.0.2
$ host app2-service.default
app2-service.default has address 10.0.0.2
$ host app2-service.default.svc.cluster.local
app2-service.default.svc.cluster.local has address 10.0.0.2
```

Default stands for the default namespace
Pods and services can be launched in different namespaces (to logically separate your cluster)

DNS – How does it work ?

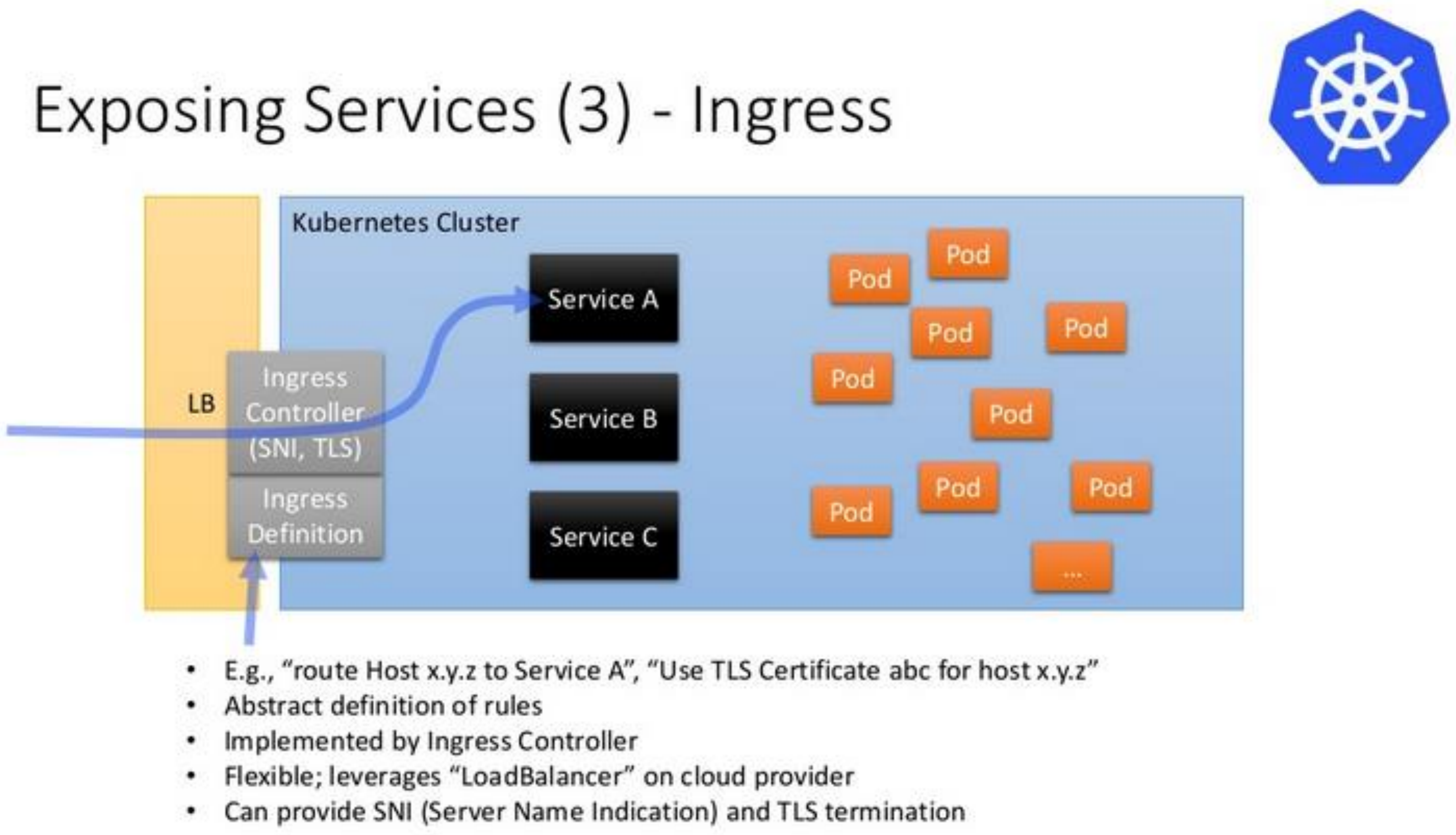


```
$ kubectl run -i --tty busybox --image=busybox --restart=Never -- sh
Waiting for pod default/busybox to be running, status is Pending, pod ready: false
If you don't see a command prompt, try pressing enter.
/ # cat /etc/resolv.conf
search default.svc.cluster.local svc.cluster.local cluster.local lan
nameserver 10.0.0.10
options ndots:5
```

Ingress

- Ingress is a solution available since kubernetes 1.1 that allows **inbound connection** to the cluster
- It's an alternative to the external LoadBalancer & NodePorts
  - Ingress allows you to **easily expose services** that need to be accessible from **outside** to the **cluster**
- With ingress you can run your own **ingress controller** (basically a LoadBalancer) within the kubernetes cluster
- There are a default ingress controllers available, or you can **write your own** ingress controller

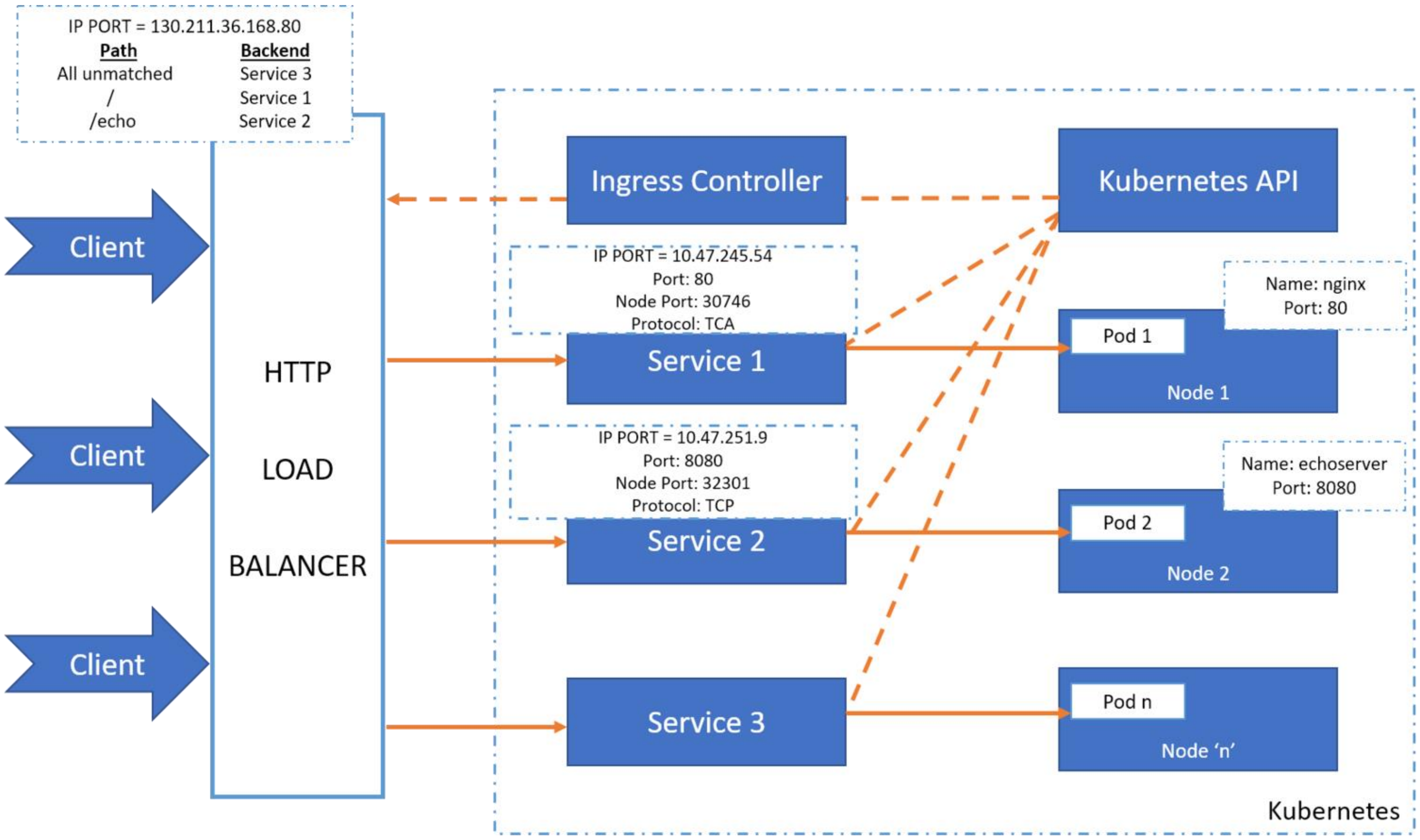
Ingress



Exposing Services (3) - Ingress

- E.g., "route Host x.y.z to Service A", "Use TLS Certificate abc for host x.y.z"
- Abstract definition of rules
- Implemented by Ingress Controller
- Flexible; leverages "LoadBalancer" on cloud provider
- Can provide SNI (Server Name Indication) and TLS termination

Ingress



IP PORT = 130.211.36.168.80

Path	**Backend**
All unmatched	Service 3
/	Service 1
/echo	Service 2

Client

Client

Client

HTTP

LOAD

BALANCER

Ingress Controller

Kubernetes API

IP PORT = 10.47.245.54
Port: 80
Node Port: 30746
Protocol: TCA

Service 1

IP PORT = 10.47.251.9
Port: 8080
Node Port: 32301
Protocol: TCP

Service 2

Service 3

Name: nginx
Port: 80

Pod 1

Node 1

Name: echoserver
Port: 8080

Pod 2

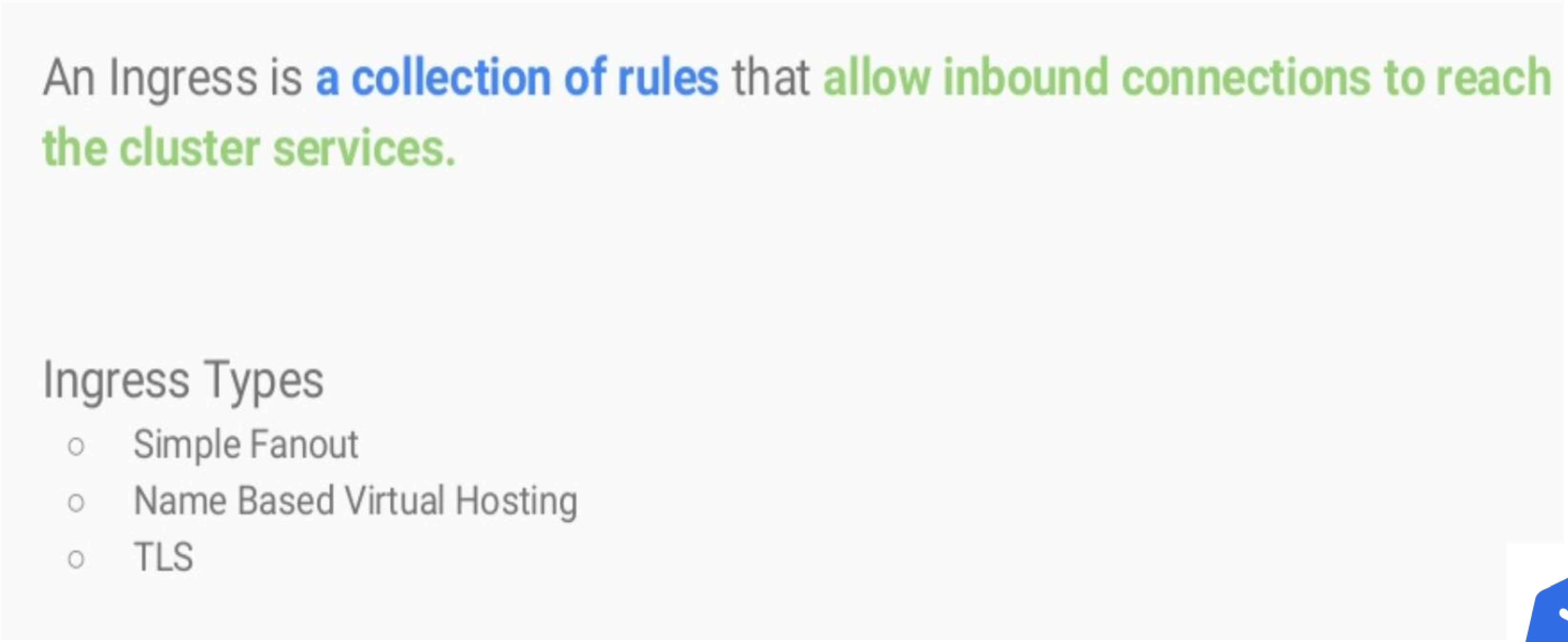
Node 2

Pod n

Node 'n'

Kubernetes

Ingress

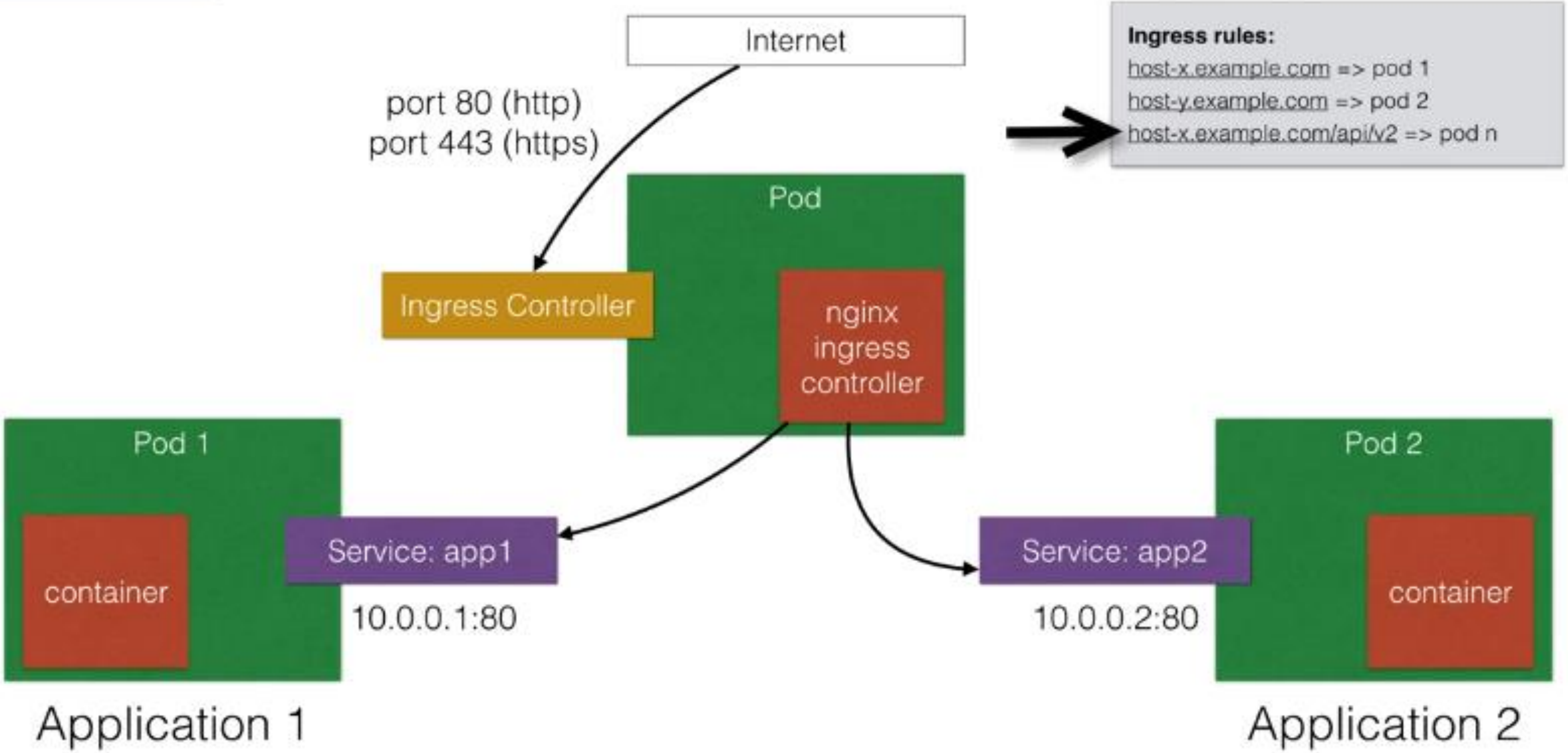


An Ingress is **a collection of rules** that **allow inbound connections to reach the cluster services.**

Ingress Types

- Simple Fanout
- Name Based Virtual Hosting
- TLS

Ingress



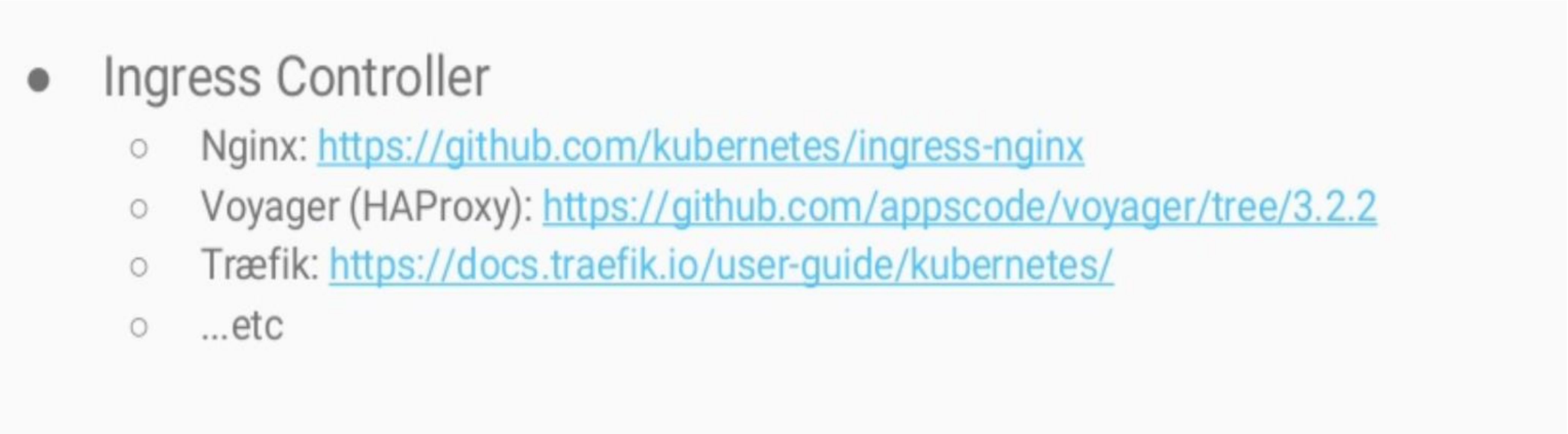
Ingress rules:

host-x.example.com => pod 1

host-y.example.com => pod 2

host-x.example.com/api/v2 => pod n

Ingress



- Ingress Controller
  - Nginx: https://github.com/kubernetes/ingress-nginx
  - Voyager (HAProxy): https://github.com/appscode/voyager/tree/3.2.2
  - Træfik: https://docs.traefik.io/user-guide/kubernetes/
  - ...etc

Demo Placeholder

- Ingress Controller

THANK YOU!