Introduction to Kubernetes

Content

- Kubernetes architecture
- Main objects in Kubernetes
- How to install Kubernetes

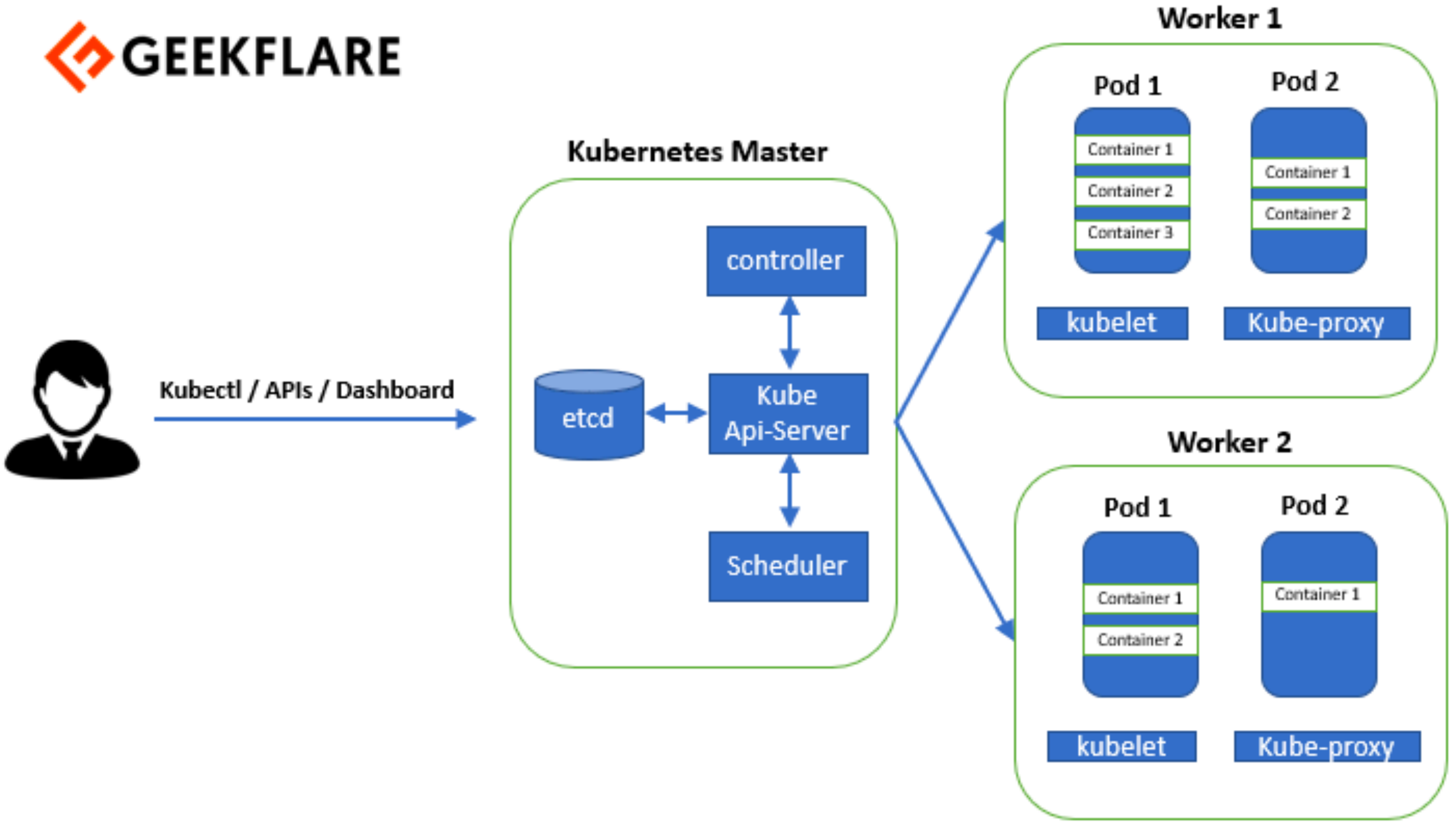
Kubernetes architecture

Kubernetes is a modern platform used to orchestrate containerized applications.

Advantages:

- Repeatable configuration
- Simple scaling (horizontal and vertical)
- Automatic worker node scaling (adjust resources)
- Automatic application recovery

GEEKFLARE


Worker 1
Pod 1
Pod 2
Container 1
Container 2
Container 3
Container 1
Container 2
kubelet
Kube-proxy
Kubernetes Master
controller
etcd
Kube
Api-Server
Scheduler
Kubectl / APIs / Dashboard
Worker 2
Pod 1
Pod 2
Container 1
Container 2
Container 1
kubelet
Kube-proxy

Master nodes

- master is a virtual machine that contains system components (control-plane)
- **api-server** is to validate and to config data for different objects.
- **controller** is to keep cluster state and to transfer current state to desired one
- **sheduler** is to schedule workload across the worker nodes checking current resource consumption and using certain rules to find “the best” node.
- etcd is a high-available key-value DB written on Raft used to keep all of the data from Kubernetes cluster.

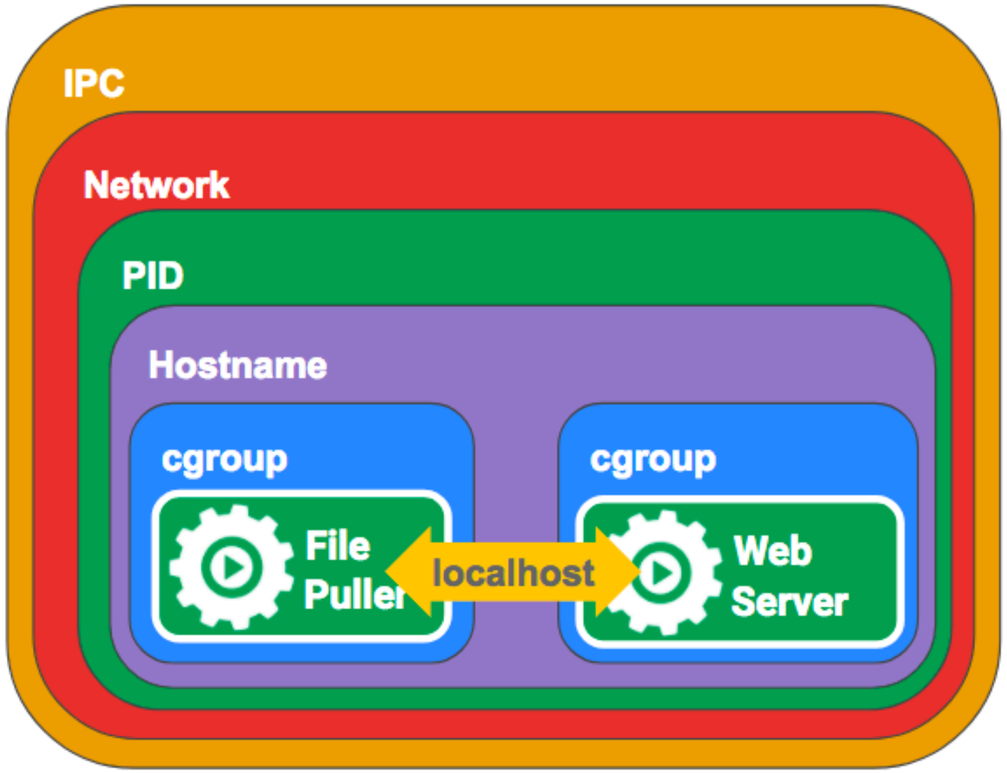
Worker nodes

- **kubelet** is to control containers lifecycle: prepare environment to launch, container creation and its termination
- **kube-proxy** is responsible for network interaction inside cluster
- **container-runtime**: Docker, CRI-O, rkt and etc.

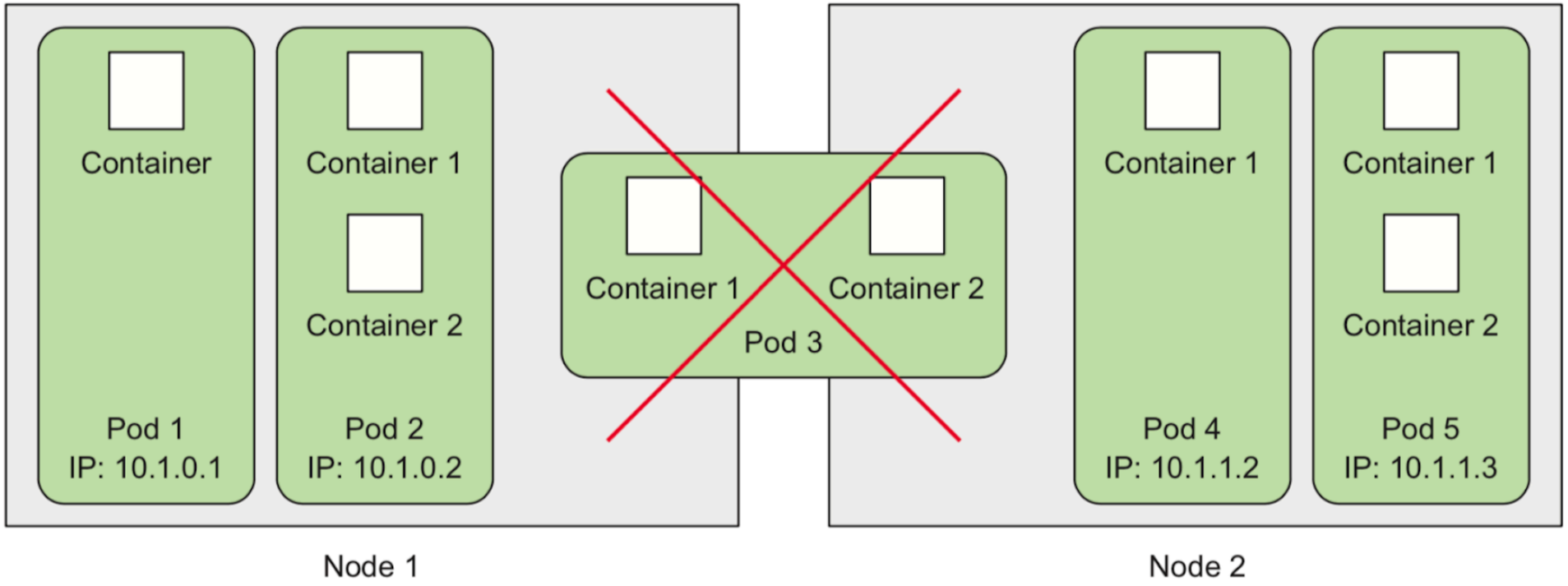
Objects in Kubernetes

Pod

is a logically connected group of containers (one or more) which have common virtual environment (like network namespace, PID and etc).



Containers in pod can’t be located on different nodes. they schedule and scale together.



To understand how to create pods, try to answer several questions:

- Should applications scale together or separately?
- Should services run on one node or they can run on different nodes?
- Are components bounded or independent?

Example of independent services: frontend and backend

Example of bounded services: application itself and container for getting metrics and expose it for monitoring system

Description of Pod object in YaML

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
```

apiVersion - api version in Kubernetes

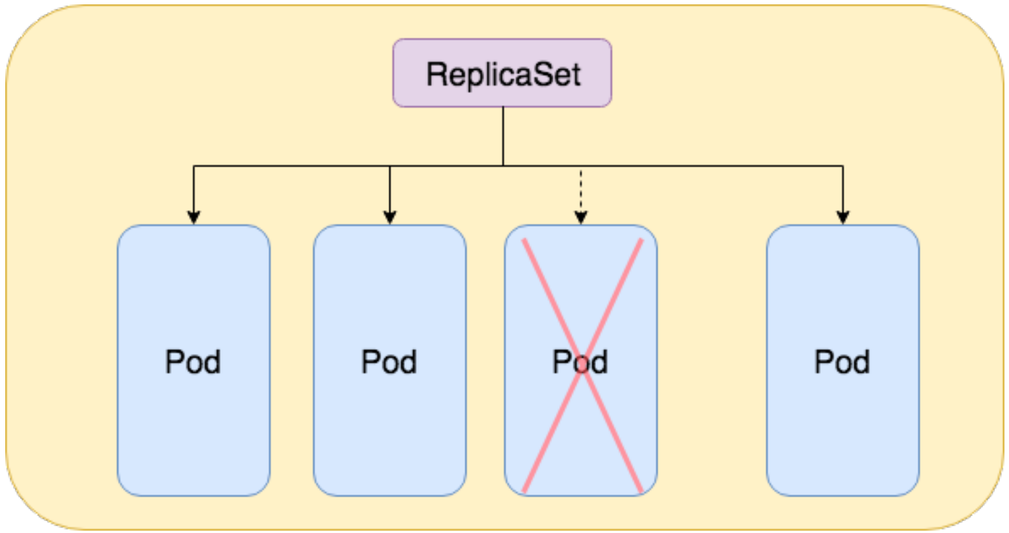
kind - object type, which we want to create

metadata - contains name, labels and annotations (service information)

spec - contains definition of object itself

ReplicaSet

is a Kubernetes controller which used to control number of application replicas with required specification. ReplicaSet object match pods by labels.



Description of ReplicaSet object in YaML

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
name: nginx-rs
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      name: nginx
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        name: nginx
```

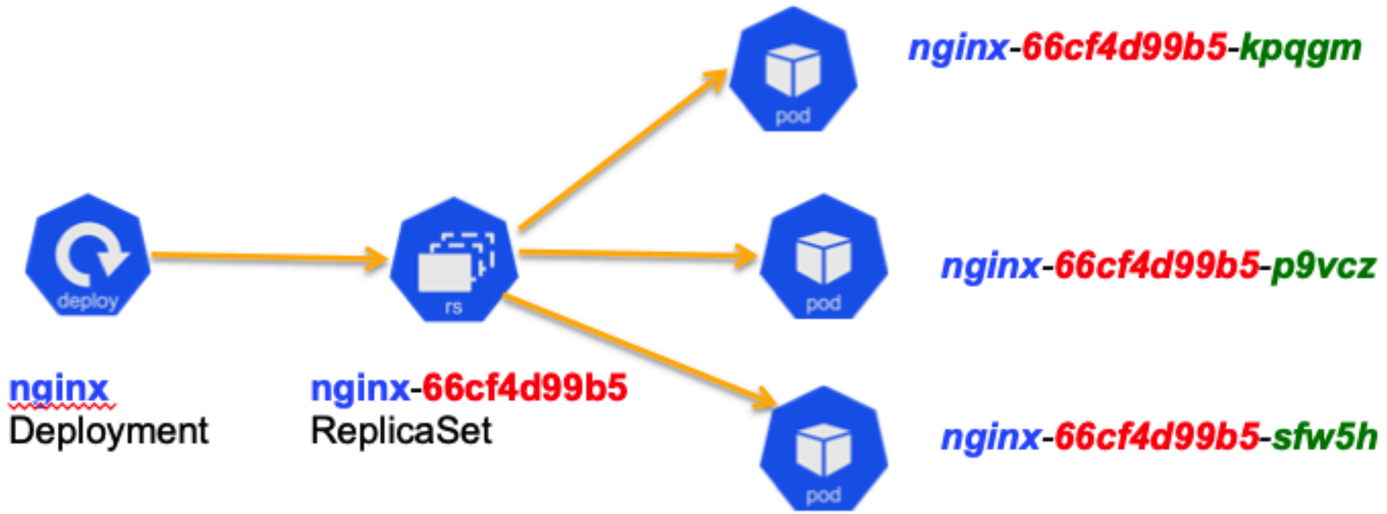
replicas - desired count of replicas

selector - labels to find target pods

template - template for pod (the same content as YAML for Pod)

Deployment

is an object to provide update of ReplicaSets.



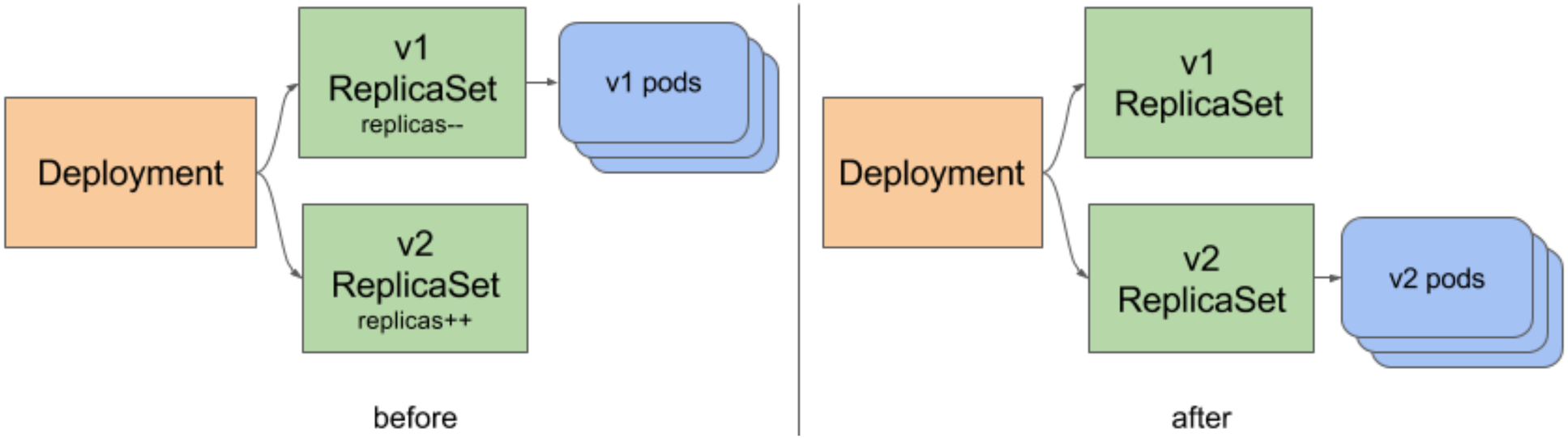
Description of Deployment in YaML

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      name: nginx
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        name: nginx
```

same as in YAML of ReplicaSet but deployment allows to define deployment strategy, perform rollback, control ReplicaSets.

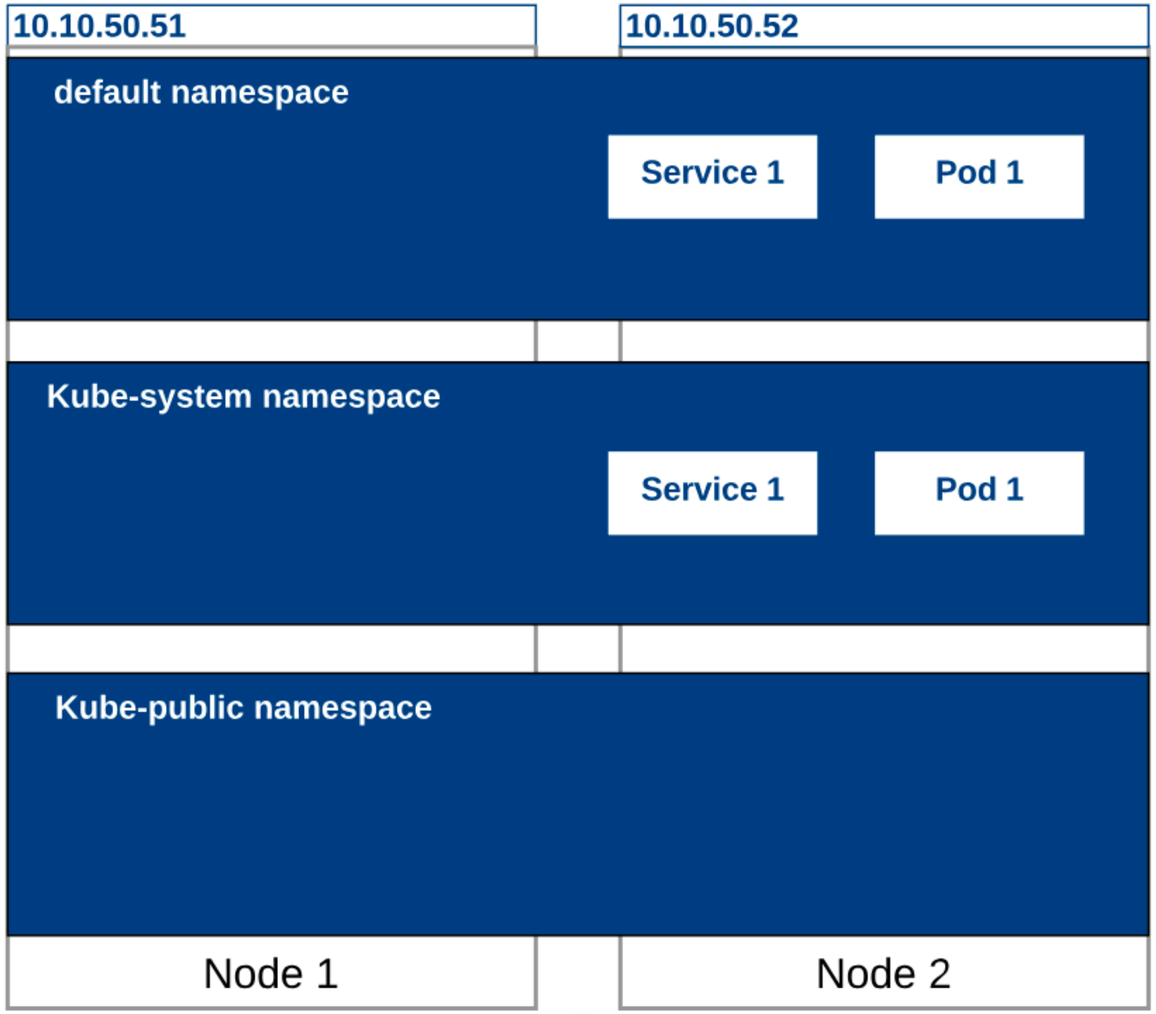
1. Create yaml for deployment object, apply it
2. Deployment creates ReplicaSet object
3. ReplicaSet checks count of pods using selector (labels) and create new or delete old pods if desired != current

Demo time!


Deployment
v1
ReplicaSet
replicas--
v1 pods
v2
ReplicaSet
replicas++
before
Deployment
v1
ReplicaSet
v2
ReplicaSet
v2 pods
after

Namespace

provides the scope for Pods, Services, and Deployments in the cluster.



Namespace allows to:

- Isolate environments and workload
- Control access rights (allow / prohibit)
- Limit consuming resources per namespace

Important to know:

- During a namespace deletion all of the objects connected to the namespace will be deleted too.
- You can’t rename a namespace

Description of Namespace in YaML

```
apiVersion: v1
kind: Namespace
metadata:
  name: custom-namespace
  labels:
    team: devops
    env: staging
```

Service

There are several types of services:

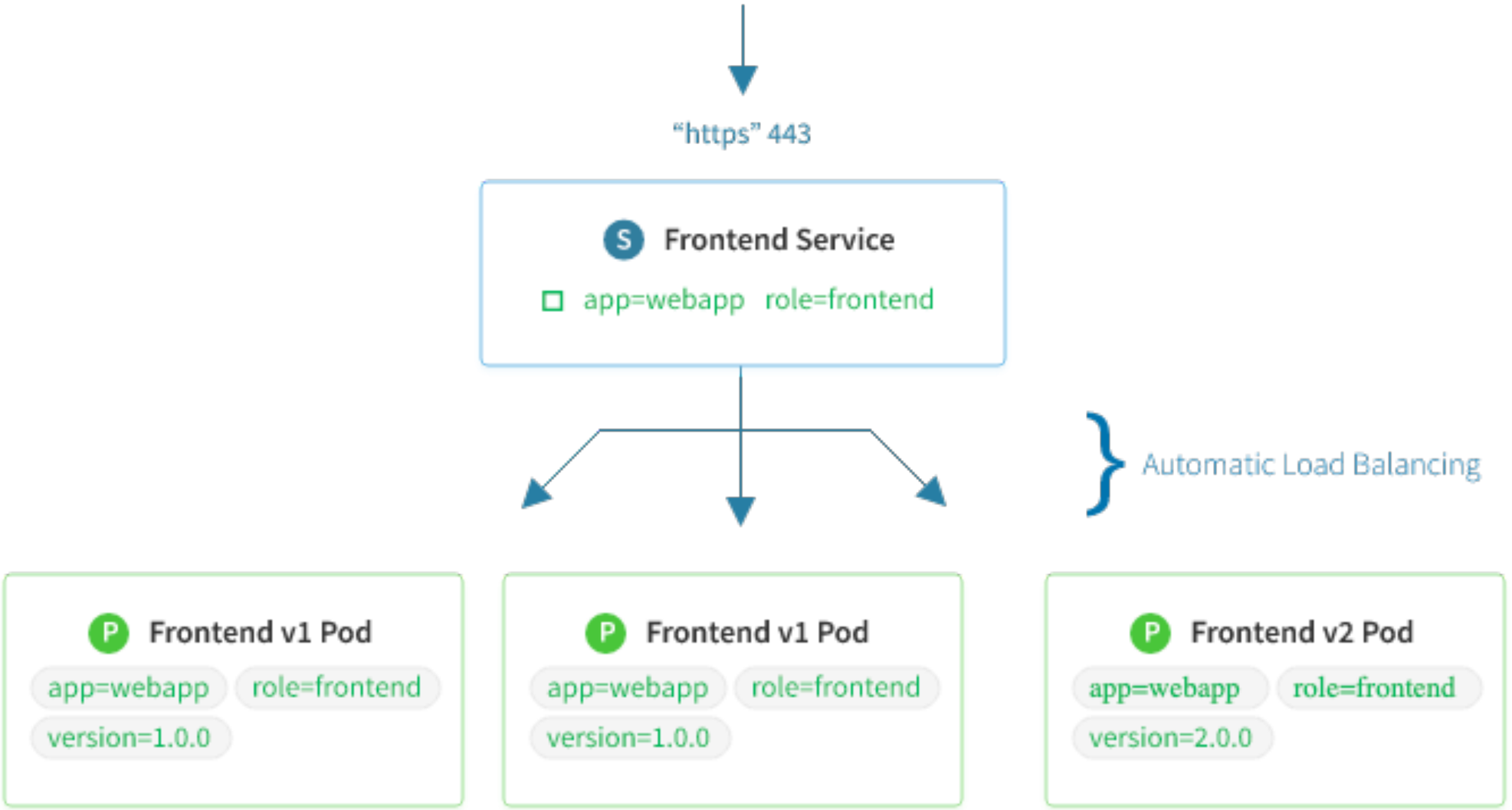
- **ClusterIP** - accessible inside the cluster, type by default
  - Access from same namespace: <service_name>: ***web***
  - Access from another namespace: <service_name>.<namespace>: ***web.nginx***
  - Fullname: <service_name>.<namespace>.svc.cluster.local: ***web.default.svc.cluster.local***
- **LoadBalancer** - accessible outside the cluster by creating LoadBalancer via Public Cloud (automatically, using annotations). NodePort / ClusterIP are created automatically for this service.
- **NodePort** - expose service using static port number in each worker node. <NodeIP>:<NodePort>
- **ExternalAddress** - type of service which allows to create a CNAME for some external service. Mongodb.production -> mongo1-c01-sdfg.

Description of Service in YaML

```
apiVersion: v1
kind: Service
metadata:
  name: frontend
  labels:
    app: webapp
    role: frontend
spec:
  # type is optional, if ClusterIP desired
  type: ClusterIP
  ports:
  - port: 443
    name: https
    targetPort: https
    selector:
      app: webapp
      role: frontend
```

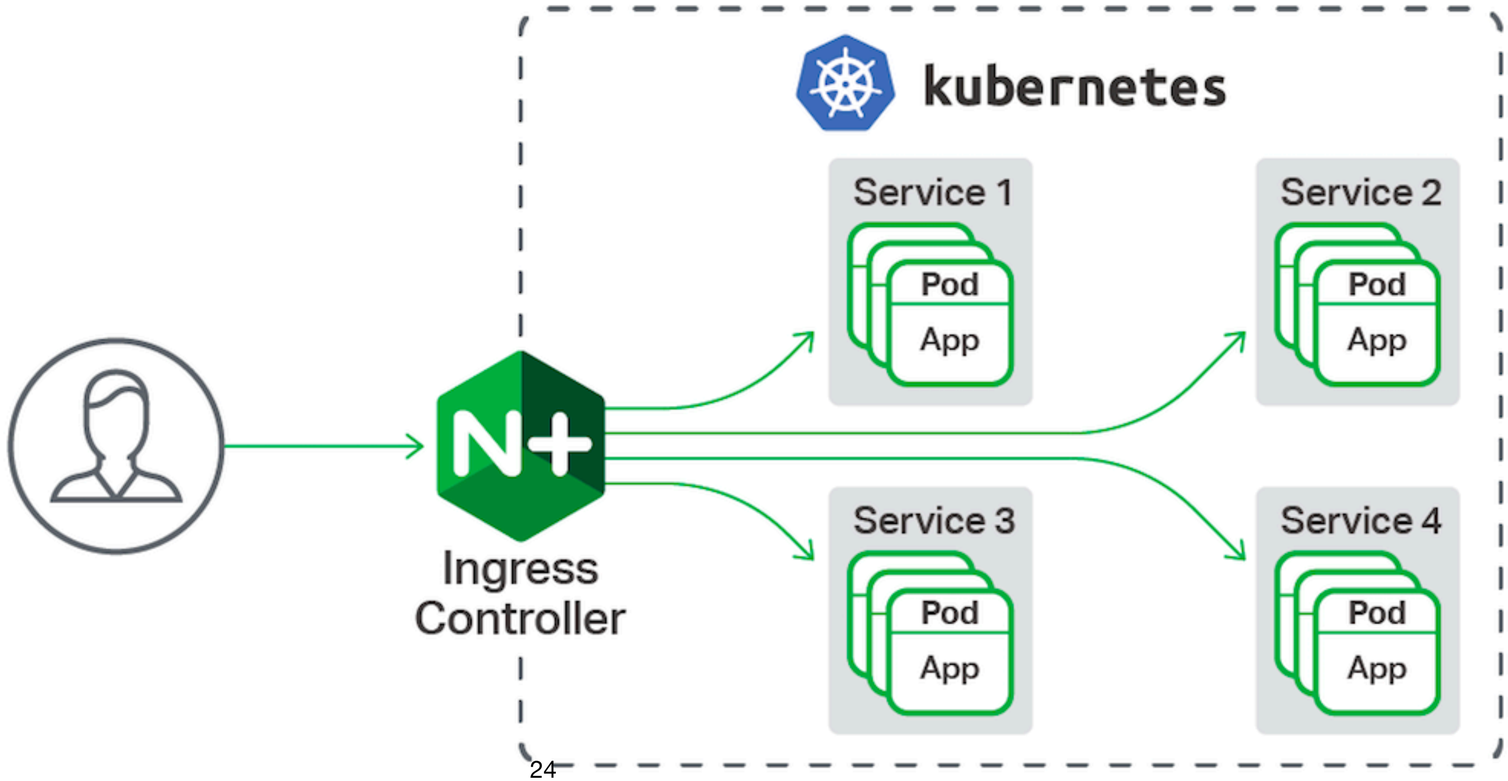
ports map defines following:

- the **port** should be opened in service
- the **targetPort** the container accepts traffic on
- names of source / target ports
- the selector defines which pods should receive traffic.


"https" 443
Frontend Service
app=webapp role=frontend
Automatic Load Balancing
Frontend v1 Pod
app=webapp
role=frontend
version=1.0.0
Frontend v1 Pod
app=webapp
role=frontend
version=1.0.0
Frontend v2 Pod
app=webapp
role=frontend
version=2.0.0

Ingress

is an object, which controls an access from outside to services inside cluster



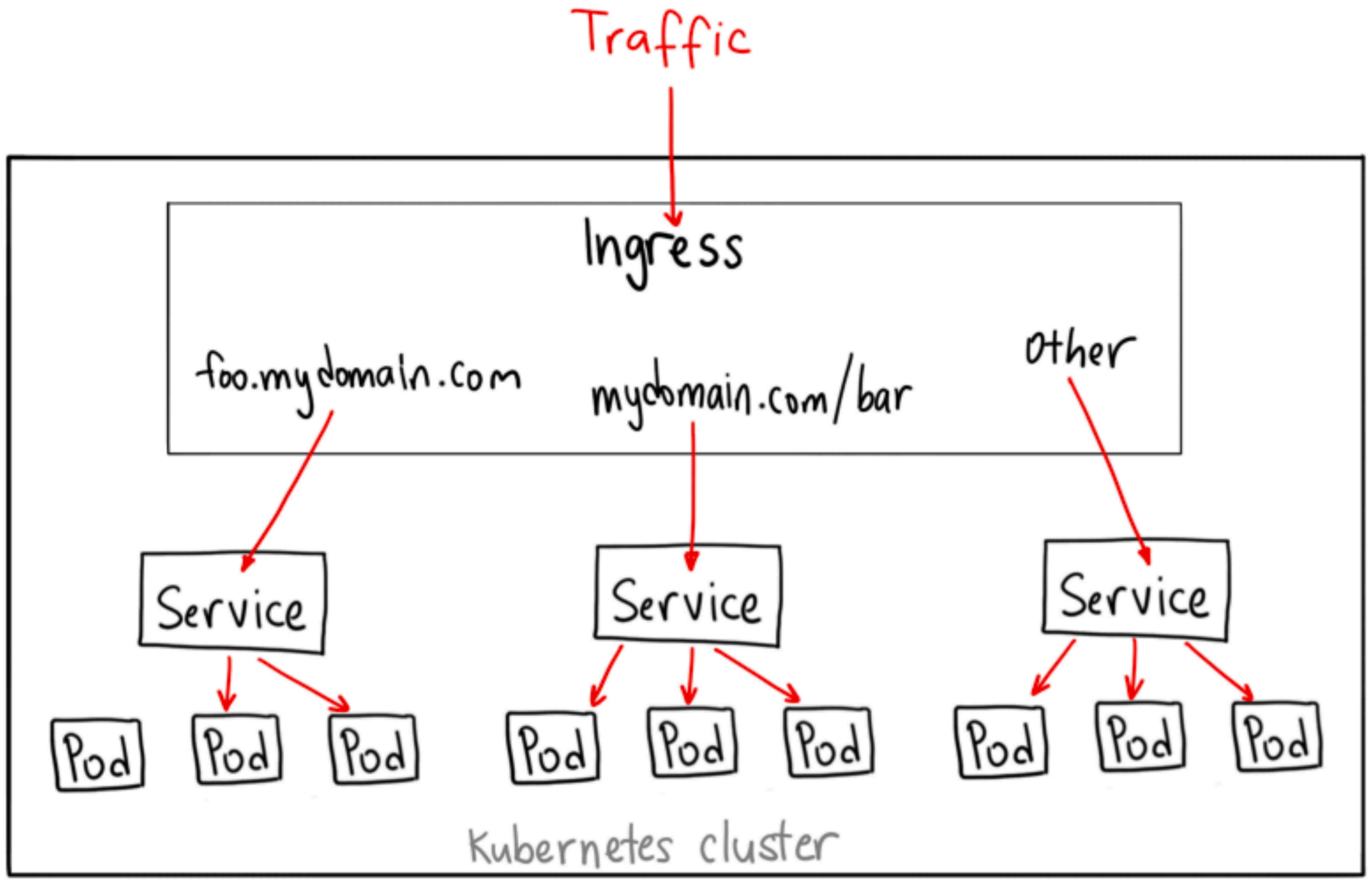
- If we just apply the ingress object definition, nothing will happen
- We need some service to keep an eye on it
- That means, that there is one more controller: Ingress controller
- There are many of it, but here the list of the most popular:
  - Nginx ingress controller
  - Istio ingress controller
  - Traefik controller
  - HA proxy ingress controller

Description of Ingress in YaML

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ui
spec:
rules:
  - http:
    paths:
    - path: /ui
      backend:
        serviceName: nginx
        servicePort: http
```

In the paths: describe paths, which we want to route and specify the target backend for it (service name).

Demo time!


Traffic
Ingress
foo.mydomain.com
mydomain.com/bar
other
Service
Service
Service
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Kubernetes cluster

Last but not least:

- **Daemonsets** is the same as deployment but containers from the object specification will be deployed to all the nodes in cluster (used for nodes monitoring, log shippers and etc.)
- **StatefulSets** is almost the same as deployment, but it has defined names of pods: nginx-0, nginx-1 and etc. Also it has an additional list of features for stateful apps.
- **Configmap** is an object to keep configuration: variables, files.
- **Secrets** is the same as Configmap, but values are decoded using base64.
- **Job/CronJob** - using these type of objects we can run some non-regular/ regular workload. We can use it for creating backup, prepare report, cleanup space and etc.

Ways to install Kubernetes

- Simple: minikube, kind, docker desktop in Mac / Windows
- Medium: cloud-managed installations - AWS EKS, GCP GKE, Azure AKS и др.
- Expert: training kubernetes-the-hardway, kubeadm, kops, kubespray

What’s next?

- Books: Kubernetes in Action, Kubernetes for Fullstack developers
- Learning by doing: Katacoda
- Online course: Introduction to Kubernetes
- Article with resources for learning K8S