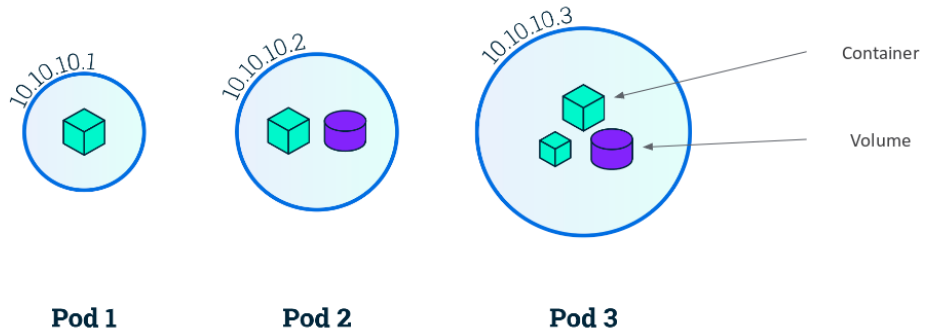
MÓDULO 5

Conceptos de Kubernetes

CONCEPTOS DE KUBERNETES: POD

QUÉ SON PODS

- Los pods son los objetos más pequeños y básicos que se pueden implementar en Kubernetes.
- Un pod representa una instancia única de un proceso en ejecución en el clúster.
- Los pods comprenden uno o más contenedores, como los de Docker.
- Cuando un pod ejecuta varios contenedores, estos se administran como una sola entidad y comparten los recursos del pod.
- En general, ejecutar varios contenedores en un solo pod representa un caso práctico avanzado.



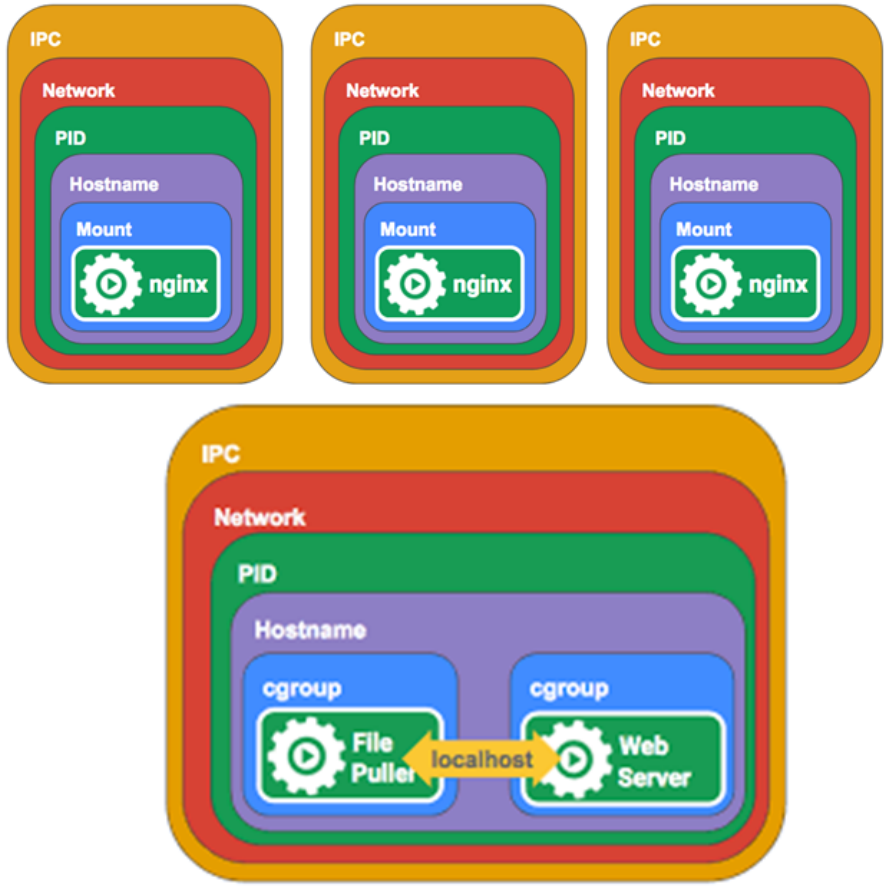
RECURSOS DE UN POD

Los pods también poseen recursos compartidos de red y almacenamiento para sus contenedores:

- Red: A los pods se les asignan direcciones IP únicas de manera automática. Los contenedores del pod comparten el mismo espacio de nombres de red, incluida la dirección IP y los puertos de red. Los contenedores en un pod se comunican entre sí dentro del pod en localhost.

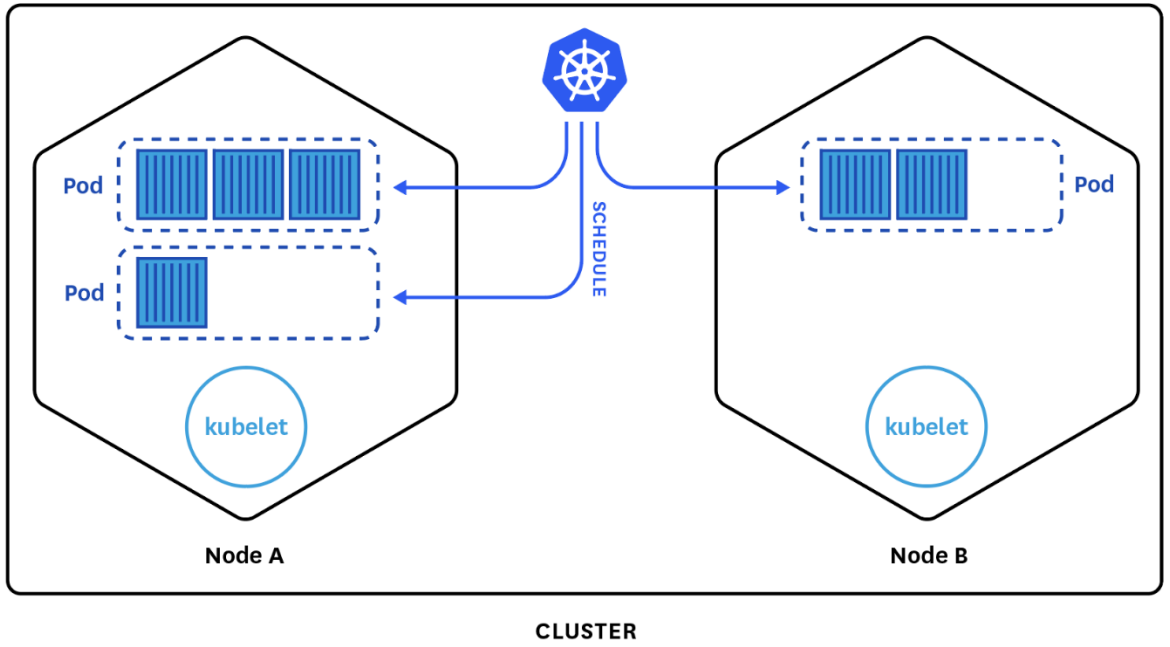
- IPC: Los contenedores van a poder ver los procesos de un contenedor con otro contenedor .

- UTS: UTS (Unix timesharing system) es el hostname y como estamos compartiendo la misma IP seria razonable compartir el mismo hostname en los tres contenedores.



OBJETIVO DE UN POD

- El objetivo de un pod es ejecutar una única instancia de tu aplicación en el clúster.
- Los pods se ejecutan en los nodos del clúster.
- Una vez creado, un pod permanece en su nodo hasta que se completa su proceso; entonces, el pod se borra y se expulsa del nodo debido a falta de recursos o hasta que el nodo falle.
- Si un nodo falla, se programa automáticamente la eliminación de sus pods.



RECURSOS DE KUBERNETES

- Podemos ver los recursos de nuestro clúster a través de la API con el comando :

kubectl api-resources

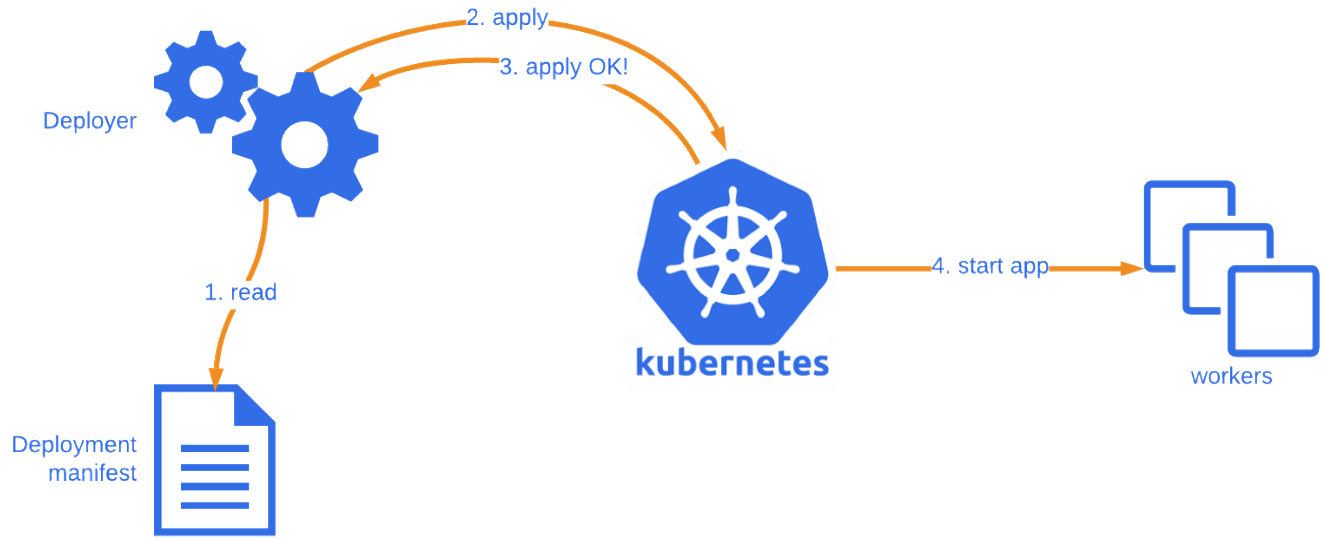
```
NAME                              SHORTNAMES   APIGROUP                       NAMESPACED   KIND
bindings                                                                      true         Binding
componentstatuses                 cs                                          false        ComponentStatus
configmaps                        cm                                          true         ConfigMap
endpoints                         ep                                          true         Endpoints
events                            ev                                          true         Event
limitranges                       limits                                      true         LimitRange
namespaces                        ns                                          false        Namespace
nodes                             no                                          false        Node
persistentvolumeclaims            pvc                                         true         PersistentVolumeClaim
persistentvolumes                 pv                                          false        PersistentVolume
pods                              po                                          true         Pod
podtemplates                                                                  true         PodTemplate
replicationcontrollers            rc                                          true         ReplicationController
resourcequotas                    quota                                       true         ResourceQuota
secrets                                                                       true         Secret
serviceaccounts                   sa                                          true         ServiceAccount
services                          svc                                         true         Service
mutatingwebhookconfigurations                  admissionregistration.k8s.io   false        MutatingWebhookConfiguration v
idatingwebhookconfigurations                admissionregistration.k8s.io   false        ValidatingWebhookConfiguration
```

CONCEPTOS DE KUBERNETES: MANIFEST

MANIFEST

- Un “Deployment Manifest" es un archivo en formato yaml .

- Que define el recurso que queremos crear o actualizar en kubernetes .




```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: myapp
  name: myapp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - image: polarsquad/hello-world-app:master
        name: hello-world
        ports:
        - containerPort: 3000
```

CAMPOS FICHERO MANIFEST DE KUBERNETES

- En el archivo .yaml del objeto de Kubernetes que quieras crear, obligatoriamente tendrás que indicar los valores de los siguientes campos (como mínimo):
  - **apiVersion** - Qué versión de la API de Kubernetes estás usando para crear este objeto.
  - **kind** - Qué clase de objeto quieres crear.
  - **metadata** - Datos que permiten identificar unívocamente al objeto, incluyendo una cadena de texto para el name, UID, y opcionalmente el namespace.
  - **spec** - El formato del campo spec es diferente según el tipo de objeto de Kubernetes, y contiene campos anidados específicos de cada objeto.

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: podtest2
5  spec:
6    containers:
7    - name: cont1
8      image: nginx:alpine
9  ---
10 apiVersion: v1
11 kind: Pod
12 metadata:
13   name: podtest3
14 spec:
15   containers:
16   - name: cont1
17     image: nginx:alpine
```

PODS CON MÁS DE UN CONTENEDOR

Al agregar otro elemento "***name***" estamos creando un contendor más en el mismo pod, con los componentes que tendrá ese pod:

```
1   apiVersion: v1
2   kind: Pod
3   metadata:
4     name: doscont
5   spec:
6     containers:
7     - name: cont1
8       image: python:3.6-alpine
9       command: ['sh', '-c', 'echo cont1 > index.html && python -m http.server 8082']
10    - name: cont2
11      image: python:3.6-alpine
12      command: ['sh', '-c', 'echo cont2 > index.html && python -m http.server 8083']
```

LABELS

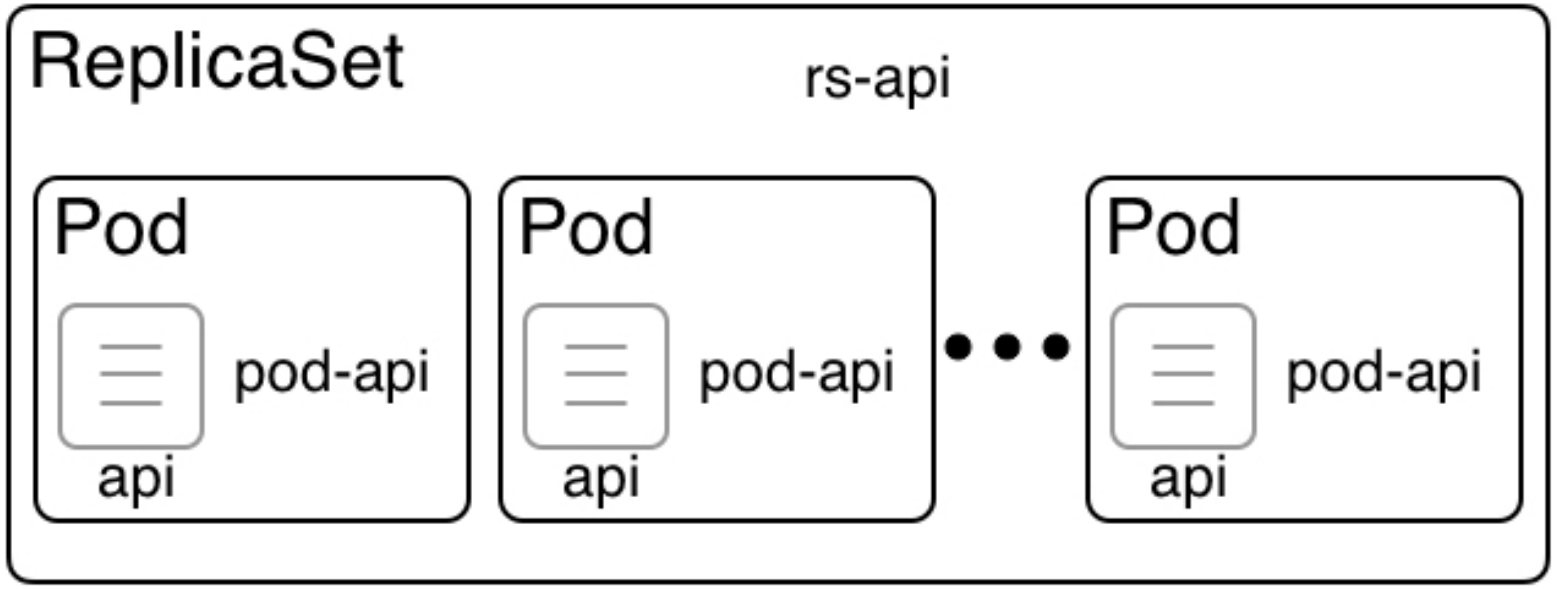
- Labels son metadata que aplicamos a un pod para poder distinguirlo.
- Los Labels van dentro del apartado metadata y justo debajo.
- Los Labels son arbitrarios, podemos definir los que necesitemos.

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: podtest2
5    labels:
6      app: front
7      env: dev
8  spec:
9    containers:
10   - name: cont1
11     image: nginx:alpine
12 ---
13 apiVersion: v1
14 kind: Pod
15 metadata:
16   name: podtest3
17   labels:
18     app: backend
19     env: dev
20 spec:
21   containers:
22   - name: cont1
23     image: nginx:alpine
```

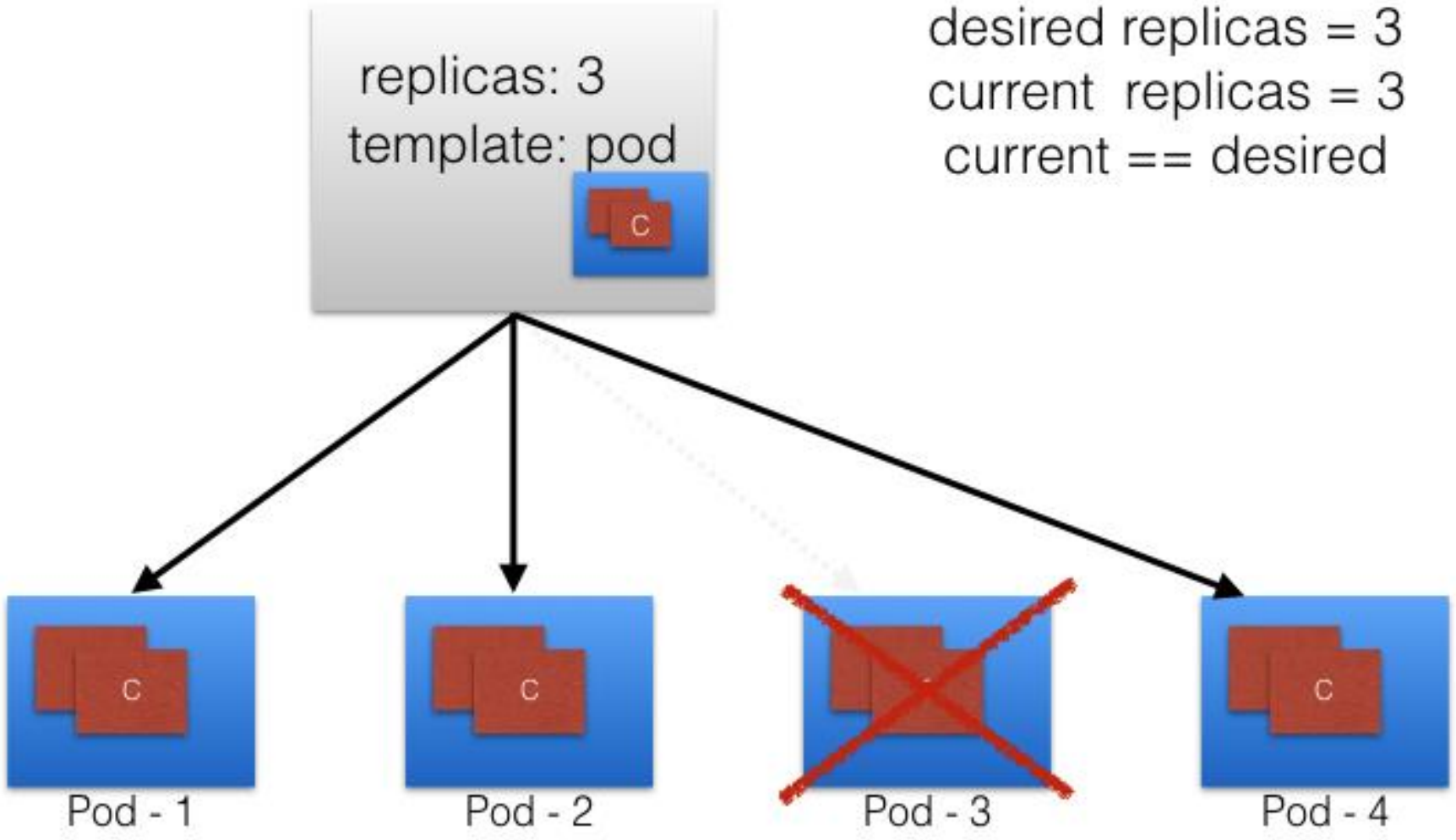
CONCEPTOS DE REPLICASET

QUÉ ES UN REPLICASET

- Un replicaset es un objeto separado del Pod.
- Un Replicaset es el propietario de los Pods.
- Un Replicaset está a un nivel más alto que el Pod



REPLICASET


replicas: 3
template: pod
C
desired replicas = 3
current replicas = 3
current == desired
C
C
C
C
Pod - 1
Pod - 2
Pod - 3
Pod - 4

CONFIGURACIÓN DE REPLICASET

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-test
  labels:
    app: rs-test
spec:
  replicas: 3
  selector:
    matchLabels:
      app: pod-label
  template:
    metadata:
      labels:
        app: pod-label
    spec:
      containers:
      - name: cont1
        image: python:3.7-alpine
        command: ['sh', '-c', 'echo cont1 > index.html && python -m http.server 8082']
```

DEFINICIÓN YAML DE UN REPLICASET

- **replicas**: Indicamos el número de pods que siempre se van a estar ejecutando.

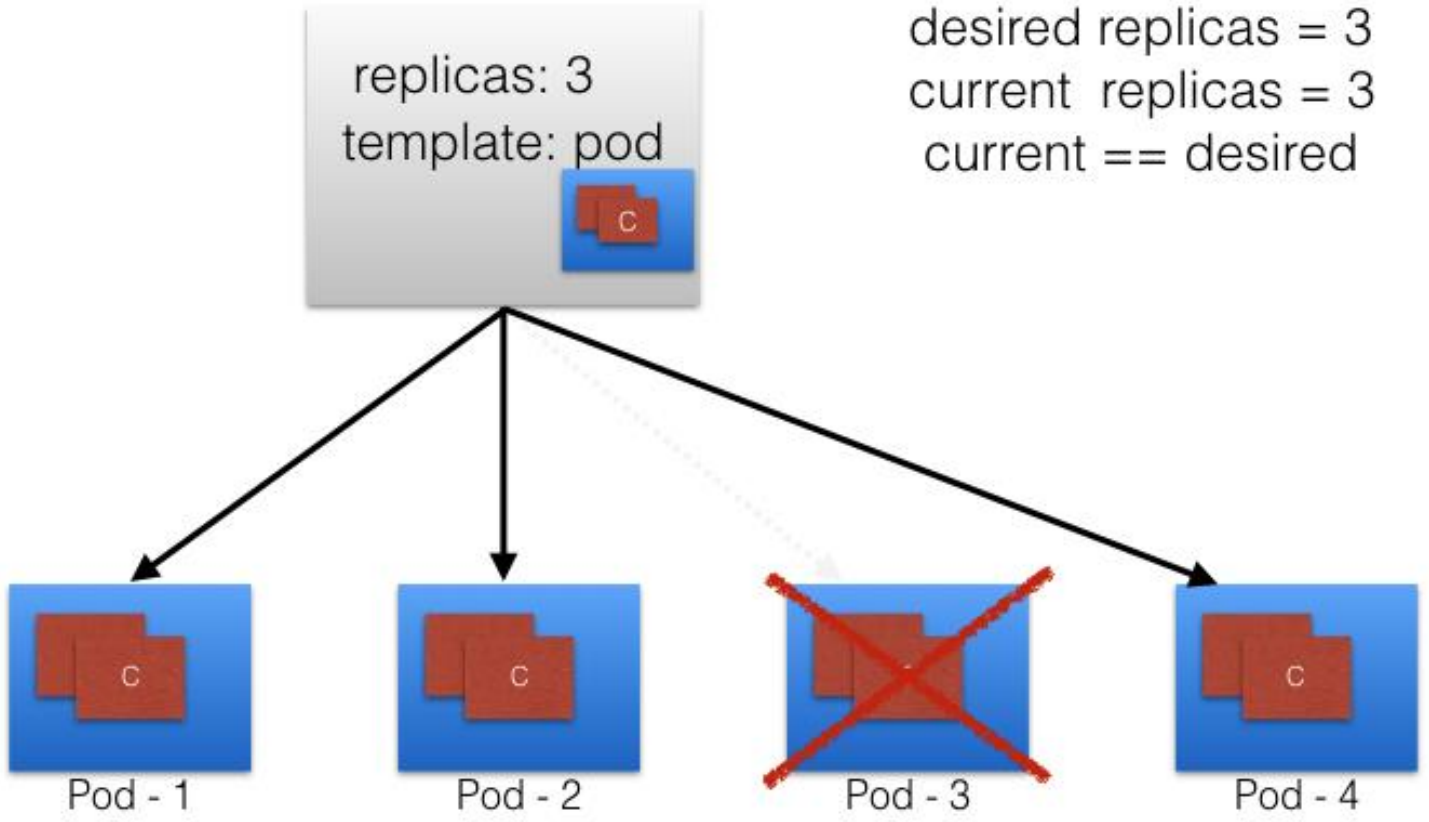
- **selector**: Indicamos los pods que vamos a replicar y vamos a controlar con el ReplicaSet. En este caso va a controlar pods que tengan un label app cuyo valor sea nginx. Si no se indica el campo selector se seleccionan por defecto los pods cuyos labels sean iguales a los que hemos declarado en la sección siguiente.

- **template**: El recurso ReplicaSet contiene la definición de un pod.

```
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
  name: nginx
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - image:  nginx
          name:  nginx
```

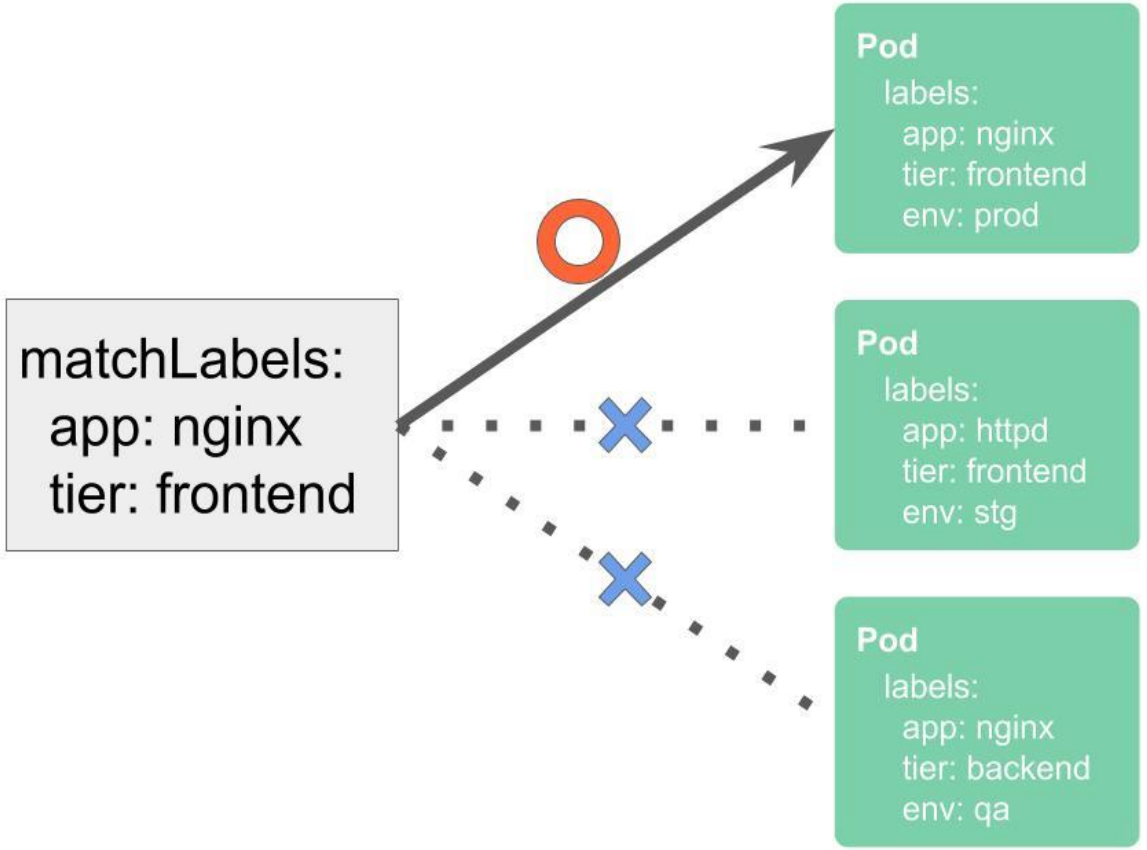
CONFIGURACIÓN DE REPLICASET

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-test
  labels:
    app: rs-test
spec:
  replicas: 3
  selector:
    matchLabels:
      app: pod-label
  template:
    metadata:
      labels:
        app: pod-label
    spec:
      containers:
      - name: cont1
        image: python:3.7-alpine
        command: ['sh', '-c', 'echo cont1 > index.html && python -m http.server 8082']
```



ADOPCIÓN DE PODS DESDE REPLICASET

- Un ReplicaSet puede heredar pods que n haya creado pero que coincidan con el selector que nosotros hemos definido.
- No es conveniente crear pods planos.
- Los pods siempre deben ser creados por objetos de mayor nivel.



RESUMEN REPLICASET

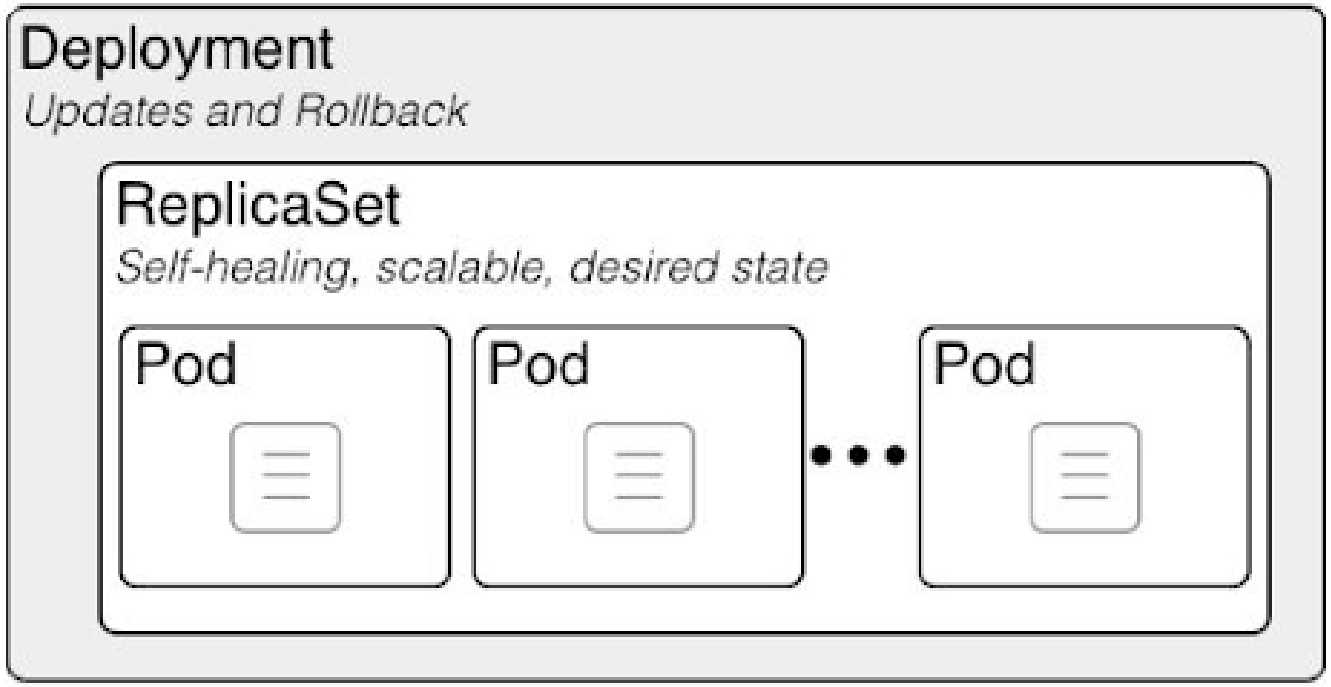
- El objetivo de un ReplicaSet es que debe mantener un numero “n” de réplicas del mismo pod en todo momento.
- Si algún Pod se cae, ReplicaSet es el encargado de crear uno nuevo según el manifest.
- Los pods siempre deben ser creados por objetos de mayor nivel.
- Un ReplicaSet no puede actualizar los pods por ejemplo cambiar la imagen, configuraciones….

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-test
  labels:
    app: rs-test
spec:
  replicas: 3
  selector:
    matchLabels:
      app: pod-label
  template:
    metadata:
      labels:
        app: pod-label
    spec:
      containers:
      - name: cont1
        image: python:3.7-alpine
        command: ['sh', '-c', 'echo cont1 > index.html && python -m http.server 8082']
```

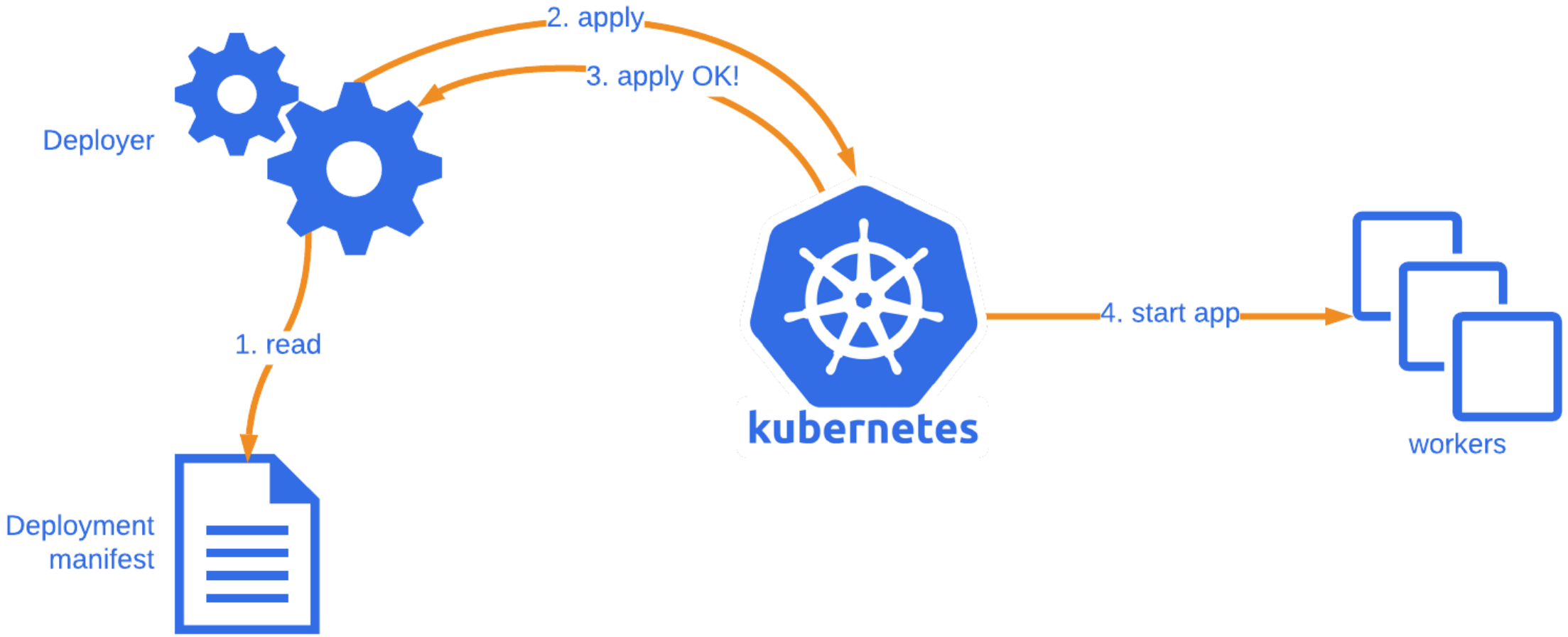
CONCEPTOS KUBERNETES DEPLOYMENT

QUÉ ES UN DEPLOYMENT

- Deployment es el propietario de los Replicaset.
- Cuando se crea un Deployment tengo que especificar un template para mi replicaset .
- Hay dos parámetros importantes en el Deployment MaxUnnavaible y Maxshort
- Un Deployment puede tener máximo 10 replicaset.



FLOW DE UN DEPLOYMENT



DEFINICIÓN YAML DE UN DEPLOYMENT

- El despliegue de un Deployment crea un ReplicaSet y los Pods correspondientes.
- Por lo tanto en la definición de un Deployment se define también el replicaSet asociado. En la práctica siempre vamos a trabajar con Deployment.
- Los atributos relacionados con el Deployment que hemos indicado en la definición son:
  - **revisionHistoryLimit**: Indicamos cuántos ReplicaSets antiguos deseamos conservar, para poder realizar rollback a estados anteriores. Por defecto, es 10.
  - **strategy:** Indica el modo en que se realiza una actualización del Deployment.
  - **recreate**: elimina los Pods antiguos y crea los nuevos.
  - **RollingUpdate**: actualiza los Pods a la nueva versión.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
spec:
  revisionHistoryLimit: 2
  strategy:
    type: RollingUpdate
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        name: nginx
        ports:
        - name: http
          containerPort: 80
```

CONFIGURACIÓN DE UN DEPLOYMENT

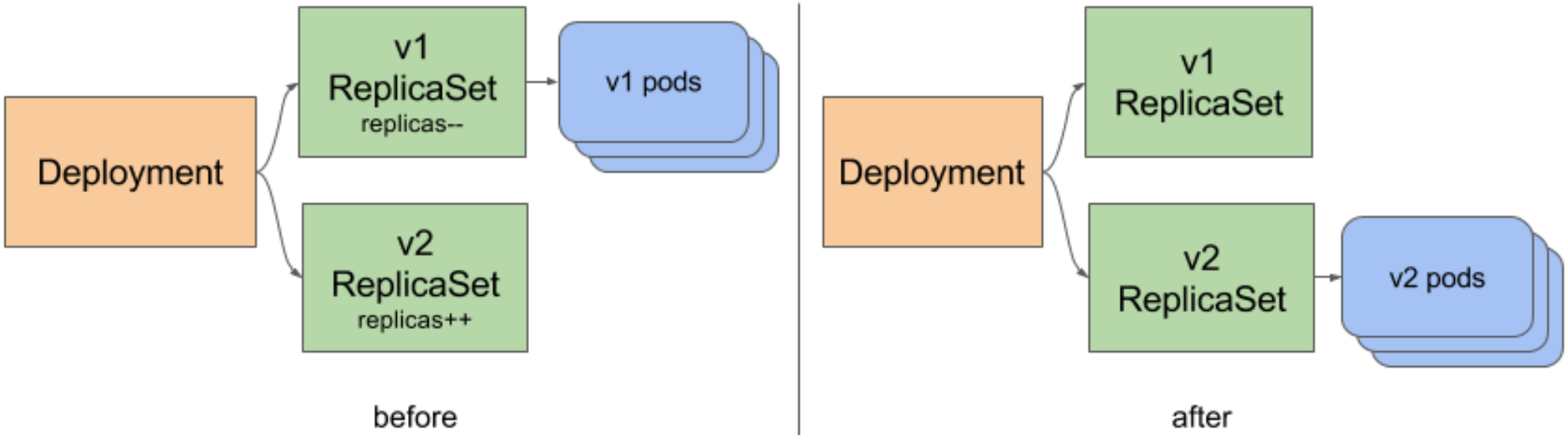
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deployment-test
  labels:
    app: front
spec:
  replicas: 3
  selector:
    matchLabels:
      app: front
  template:
    metadata:
      labels:
        app: front
    spec:
      containers:
      - name: nginx
        image: nginx:alpine

```

ROLLING UPDATES (ACTUALIZA LA VERSIÓN DE TU APLICACIÓN)

- El Rolling Updates nos sirve para poder actualizar un deployment para que consecuentemente se actualicen nuestros pods.

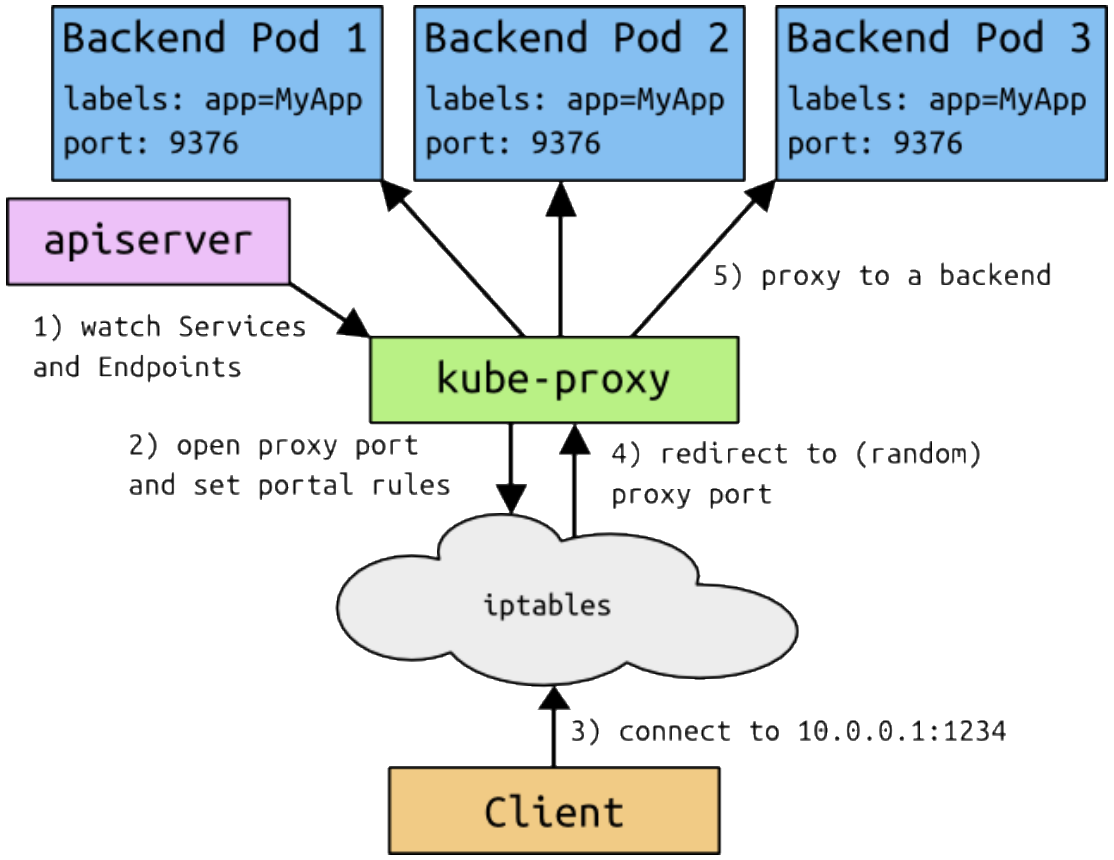
- Al modificar algo del pod como puede ser la versión de imagen de la aplicativo tendremos que decirle al deployment que actualice.



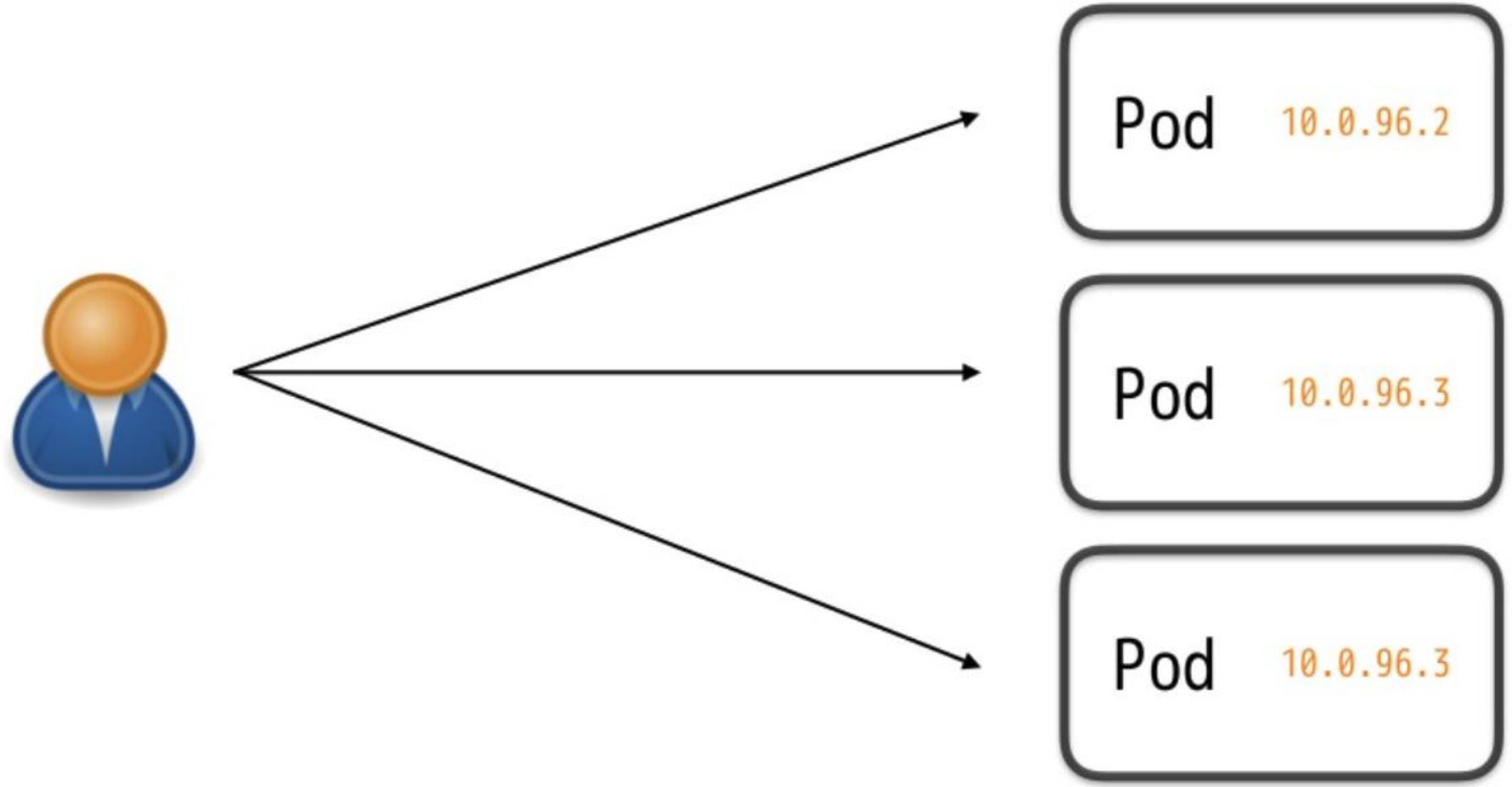
CONCEPTOS KUBERNETES: SERVICIOS

SERVICIOS EN KUBERNETES

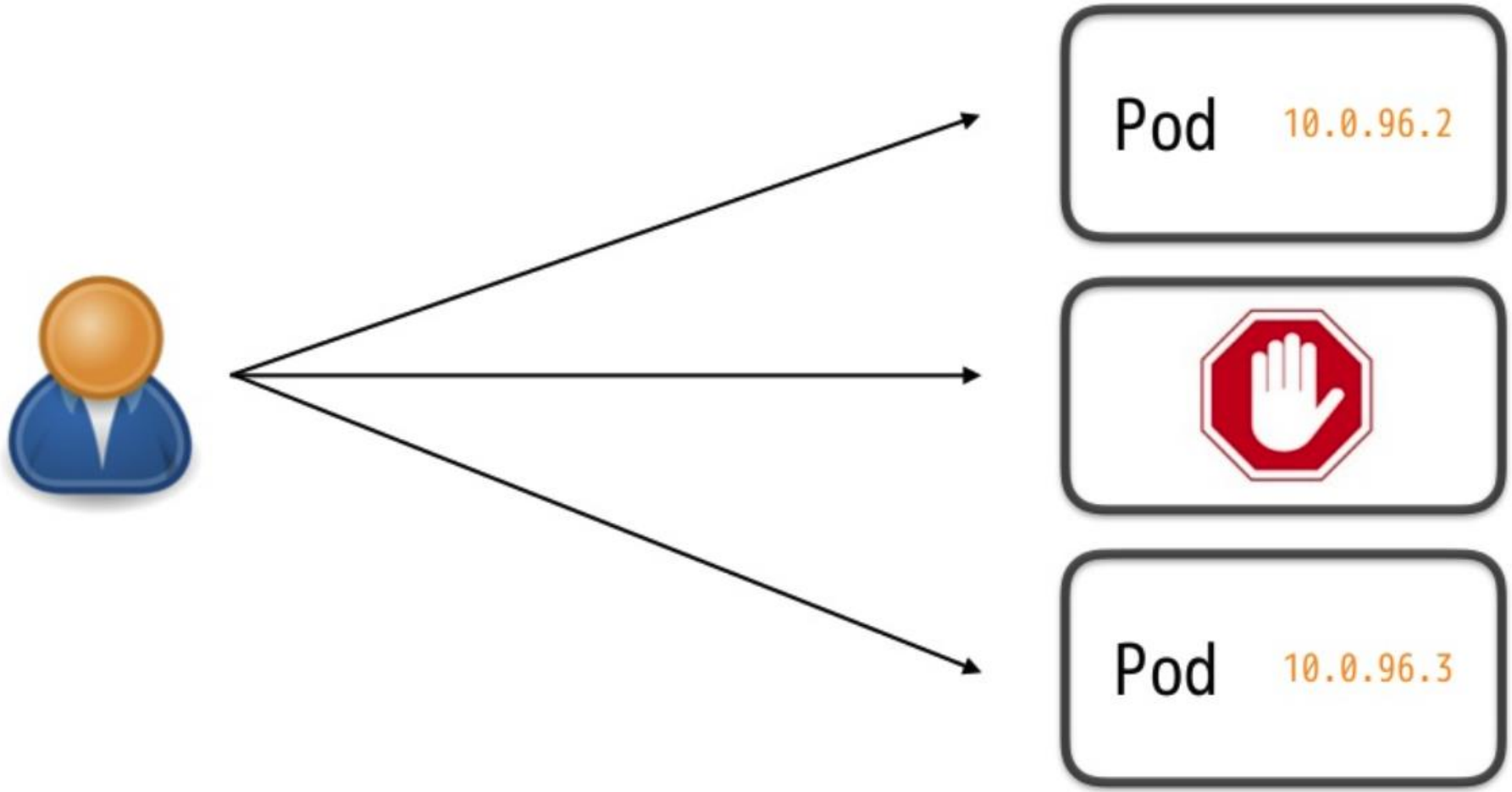
- Los servicios (services) nos permiten acceder a nuestras aplicaciones.
- Un servicio es una abstracción que define un conjunto de pods que implementan un micro-servicio. (Por ejemplo el servicio frontend).
- Ofrecen una dirección virtual (CLUSTER-IP) y un nombre que identifica al conjunto de pods que representa, al cual nos podemos conectar.
- La conexión al servicio se puede realizar desde otros pods o desde el exterior (mediante la generación aleatoria de un puerto).
- Los servicios se implementan con iptables.
- El componente kube-proxy de Kubernetes se comunica con el servidor de API para comprobar si se han creado nuevos servicios.
- Cuando se crea un nuevo servicio, se le asigna una nueva ip interna virtual (IP-CLUSTER) que permite conexiones desde otros pods.



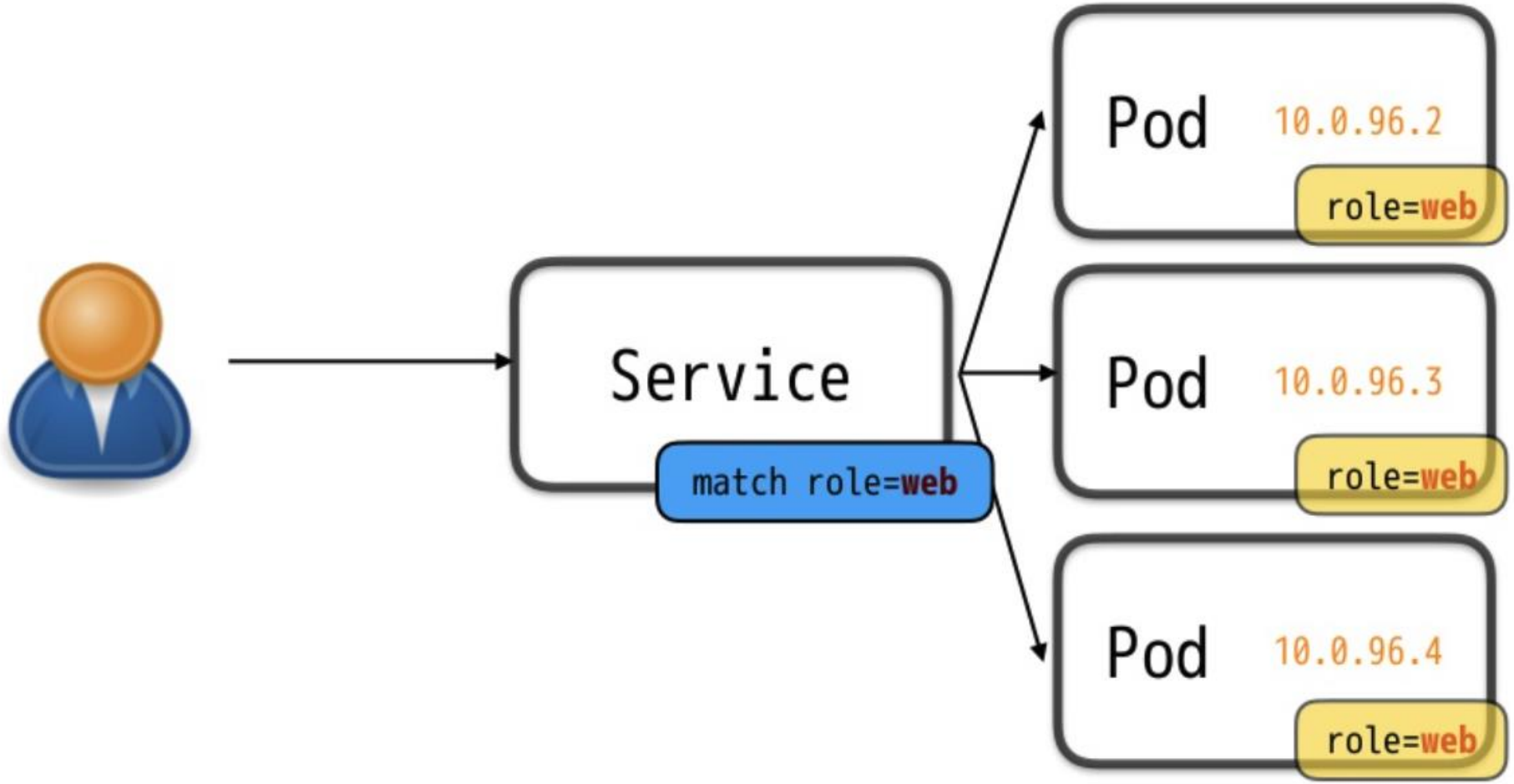
QUÉ ES UN SERVICIO



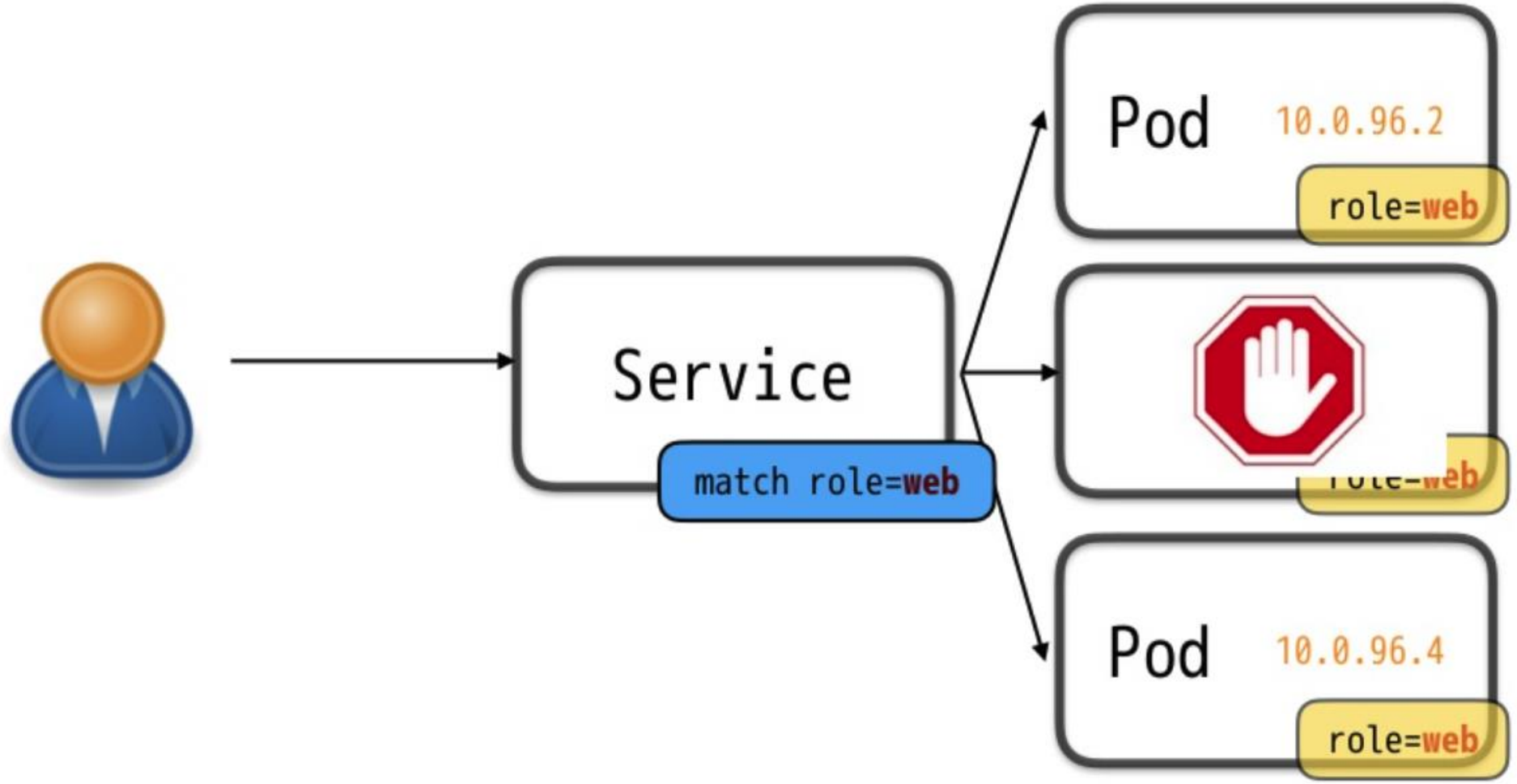
QUÉ ES UN SERVICIO



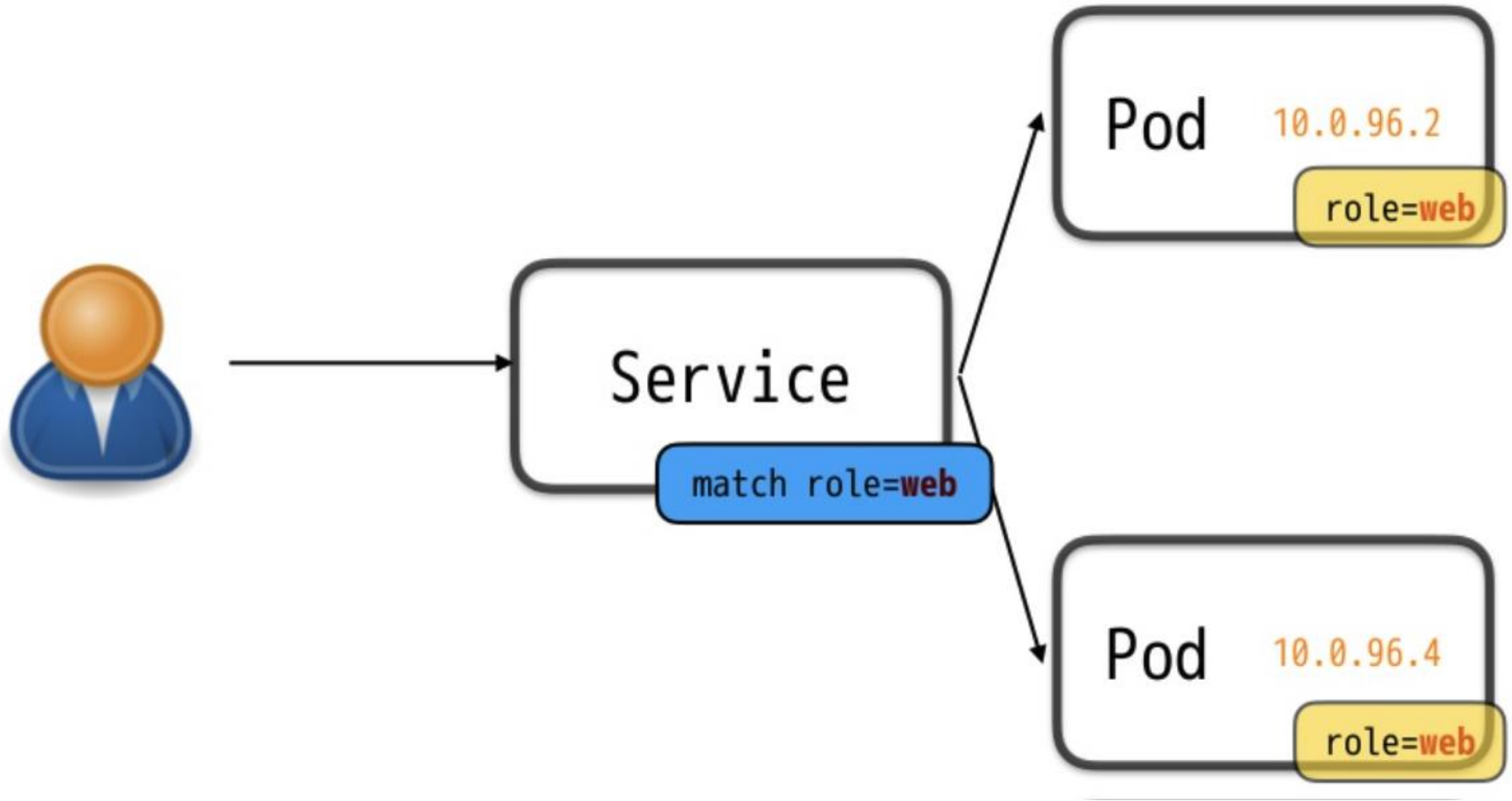
QUÉ ES UN SERVICIO



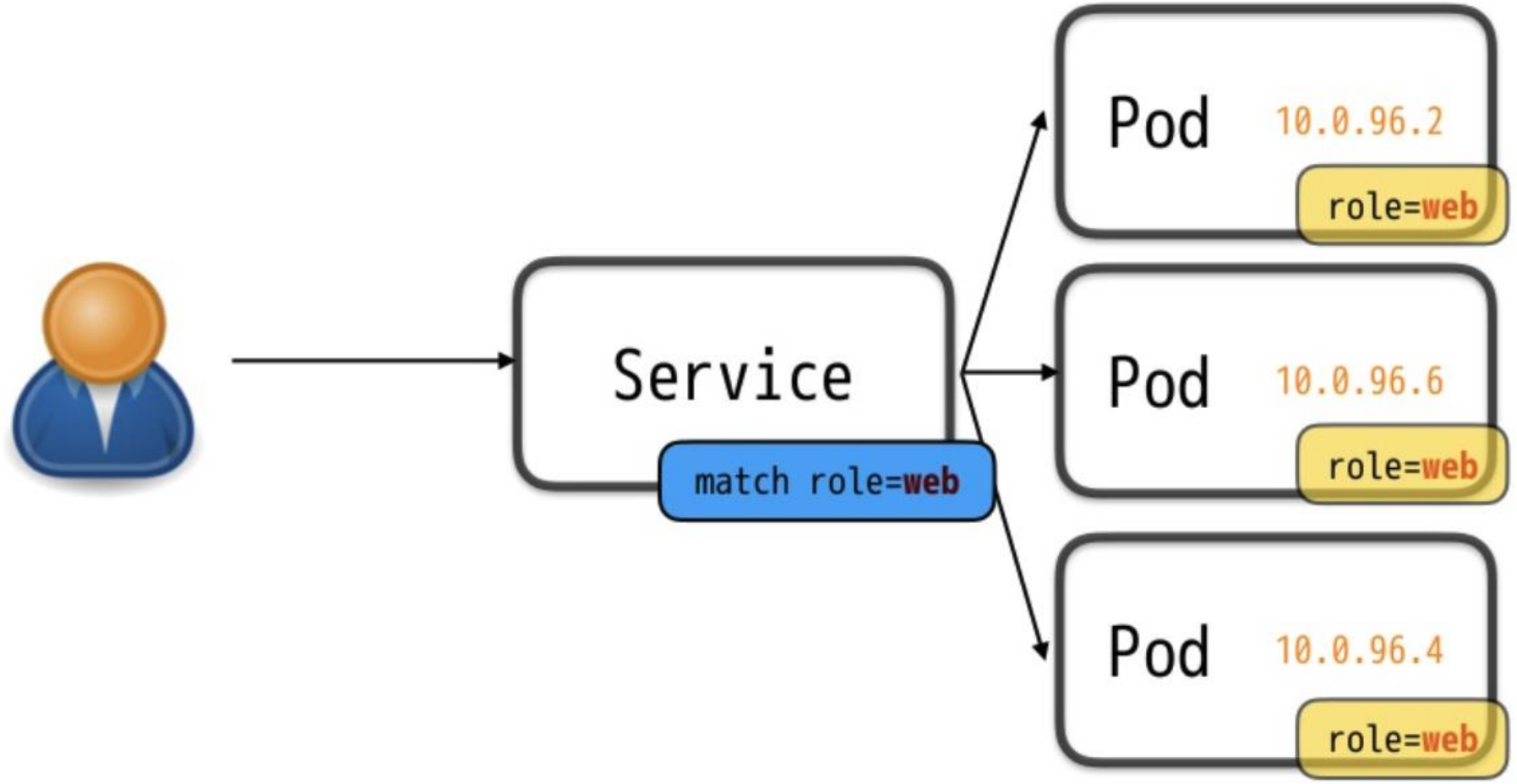
QUÉ ES UN SERVICIO



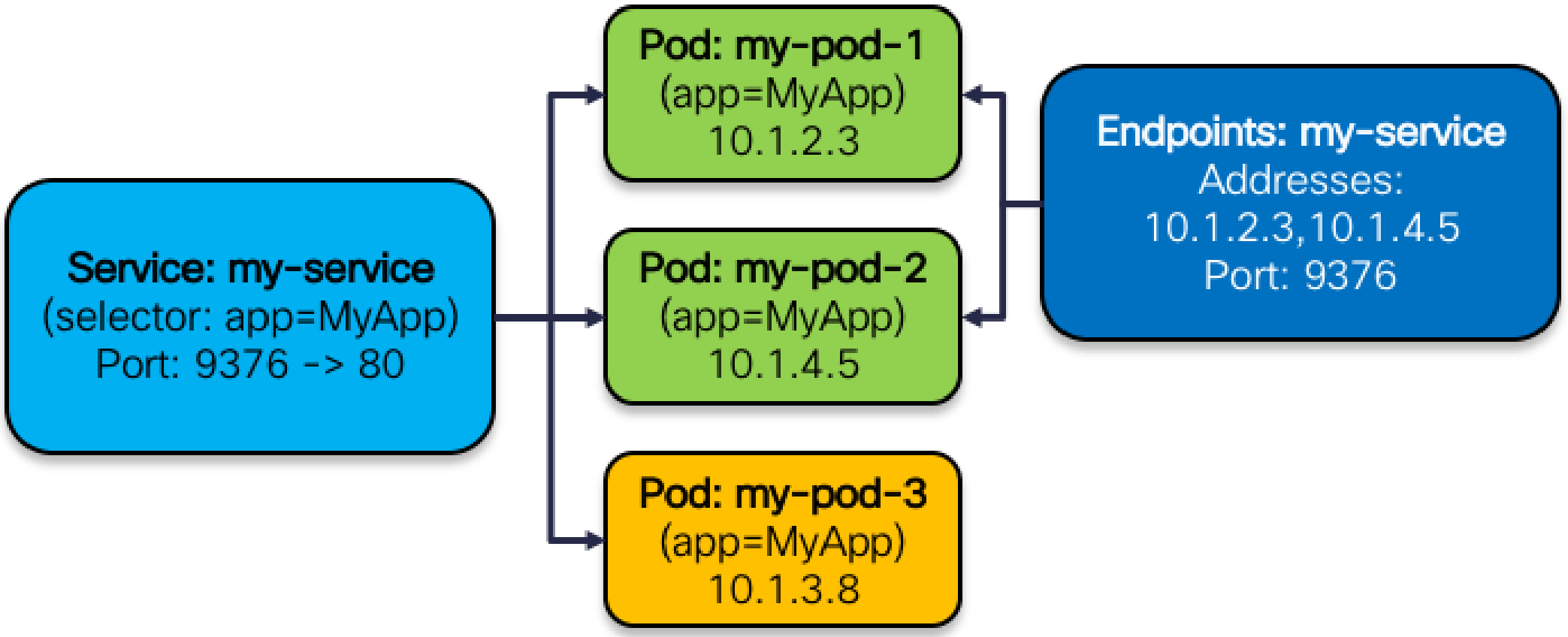
QUÉ ES UN SERVICIO



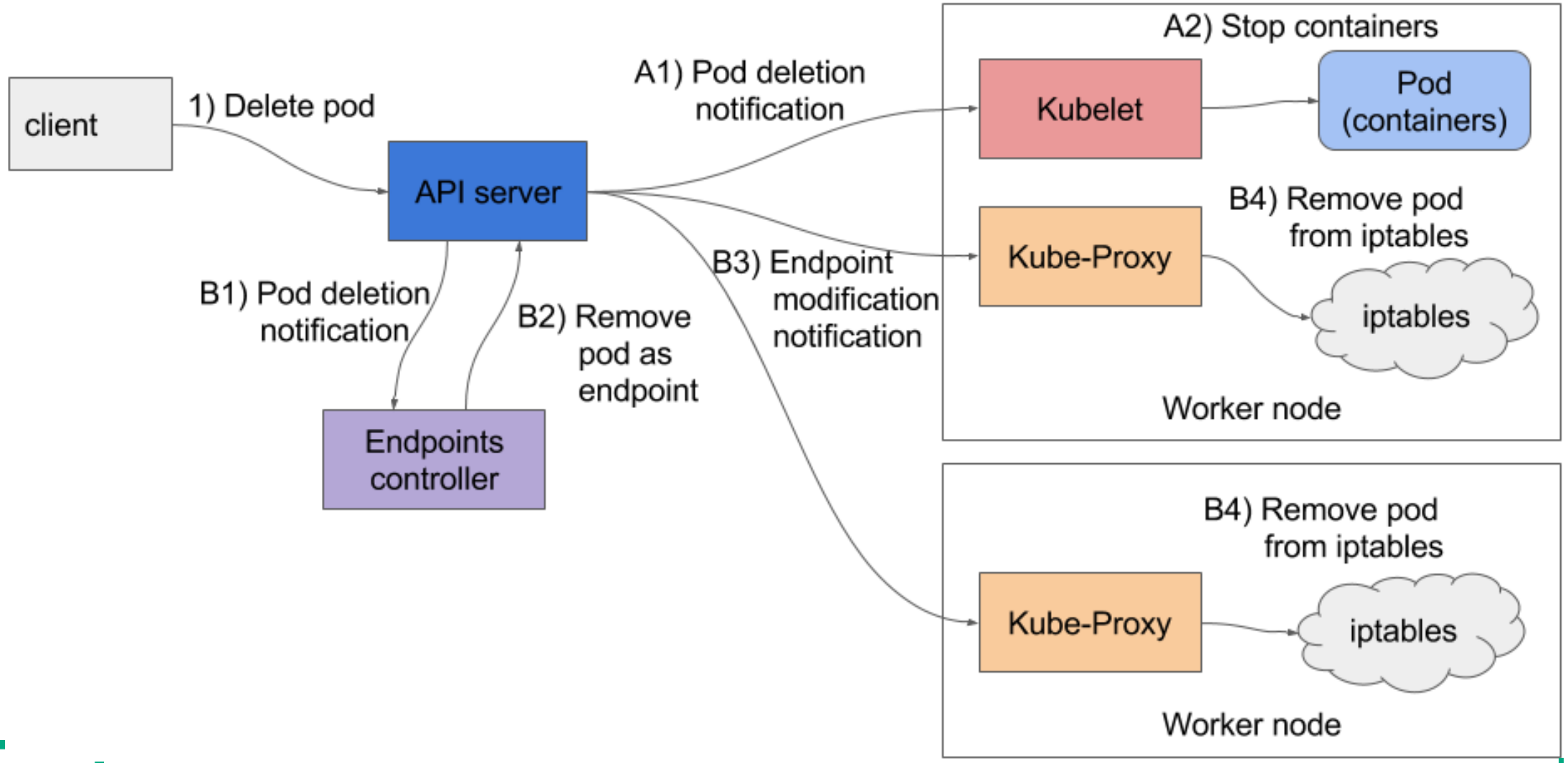
QUÉ ES UN SERVICIO



QUÉ ES UN ENDPOINT



SOLICITUDES DE CLIENTES CON KUBERNETES



CONCEPTOS DE KUBERNETES: NAMESPACE

QUÉ ES UN NAMESPACE

- Un namespace es una separación lógica que nos limita a lo que el namespace nos pueda mostrar .