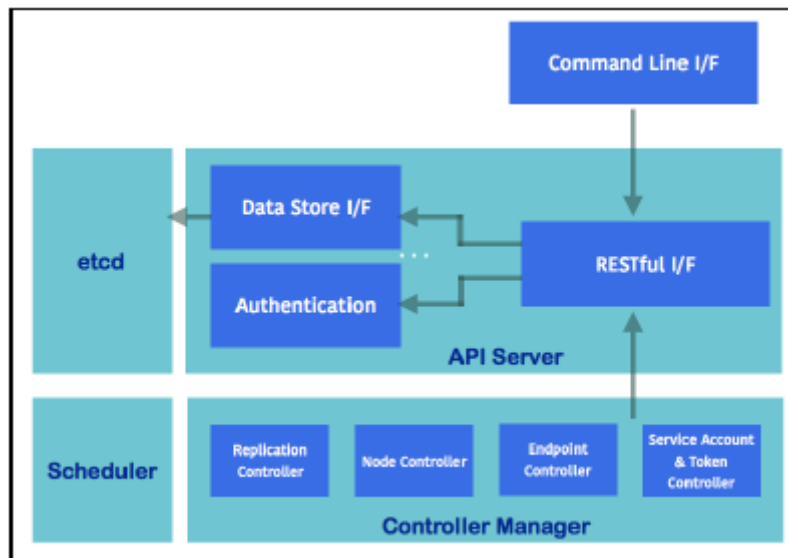


Getting started with Kubernetes

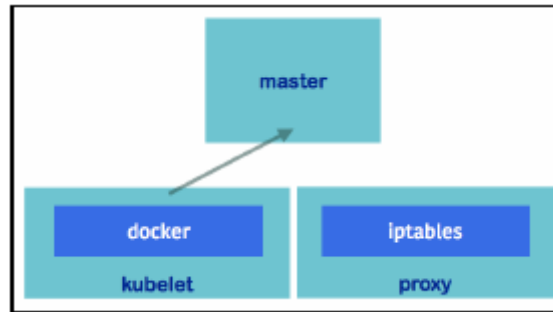
components

master

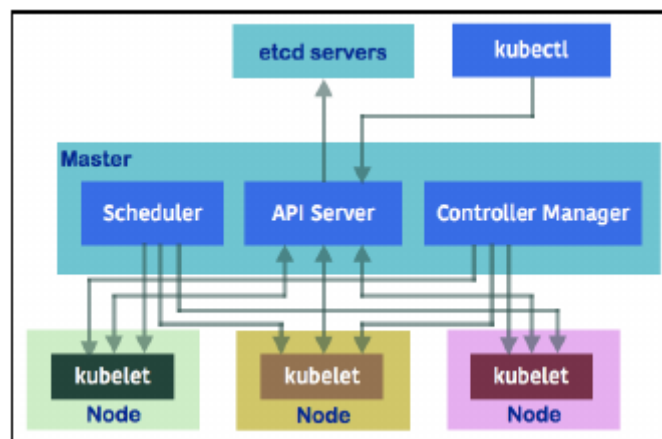


- api Server
 - kube-apiserver
 - RESTful API
- Controller Manager
 - kube-controller-manager
 - observa los cambios en la API y mantiene el cluster en el estado deseado
 - deployment controller: se asegura que el deployment se ejecuta en el número establecido de contenedores
 - node controller: responde y desaloja el pod cuando los nodos se caen
 - endpoint controller: relaciones entre pods y servicios
 - service account & token controller: crear cuenta por defecto y tokens de acceso
- Scheduler
 - kube-scheduler
 - determina que nodos son los mejores candidatos para ejecutar los pods
 - no solo se basa en el uso de los recursos, (más adelante)
- etcd:
 - base de datos distribuida key-value
 - todos los objetos de la RESTful API se guardan aquí
 - etcd se encarga de guardar y replicar los datos

nodes



- kubelet
 - es el proceso principal
 - reporta la actividad del nodo a **kube-apiserver** periodicamente
- proxy
 - kube-proxy
 - enruta a través del pod balanceador y los pods
 - enruta desde internet a los servicios
 - 3 modos:
 - userspace:
 - iptables:
 - ipvs:
- docker



getting started

kubectl

- `kubectl version`
- `kubectl logs --help`

objects

- Namespaces
 - permite implementar el aislamiento entre múltiples clusters virtuales

- objetos de diferentes namespaces no se ven entre ellos
- ciertos recursos genéricos no pertenecen a ningún namespace
- por defecto, existen 3 namespaces:
 - default
 - kube-system: almacenar objetos creados por el sistema kubernetes
 - kube-public: visible por todos los usuarios

```
apiVersion: v1
kind: Namespace
metadata:
name: project1
```

```
kubectl create -f 3-2-1_ns.yaml
```

```
kubectl get [namespaces|ns]
```

```
kubectl run nginx --image=nginx:1.12.0 --replicas=2 --port=80 --
namespace=project1
```

```
kubectl get pods --namespace=project1
```

- Name

- único en cada namespace
- usa este nombre como parte de la URL de acceso al recurso
- menor a 254c, letras minúsculas, números, guiones y puntos
- kubernetes también le asigna un UID

- Labels y selector

- asigna etiquetas arbitrarias a los objetos

```
labels:
  $key1: $value1
  $key2: $value2
```

- se pueden filtrar las etiquetas con selectores

```
selector:
  matchLabels:
    $key1: $value1
  matchExpressions:
    - {key: $key2, operator: In, values: [$value1, $value2]}
```

- Annotations

- al igual que los tags, usado para especificar metadatos no identificables
- almacenar configuración

- Pods

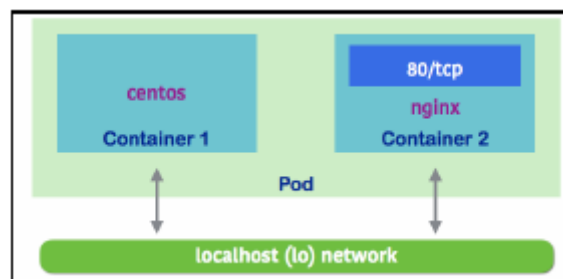
- mínima unidad deployable
- puede contener uno o varios contenedores (sidecar containers, como por ejemplo el proxy de istio)
- los contenedores de un mismo pod comparten contexto (en el mismo nodo), y por lo tanto red y volúmenes compartidos

```
kubectl explain pods
```

- ejemplo, sin uso práctico, de 2 contenedores en 1 pod:

```
// an example for creating co-located and co-scheduled container by pod
```

```
apiVersion: v1
kind: Pod
metadata:
  name: example
spec:
  containers:
  - name: web
    image: nginx
  - name: centos
    image: centos
    command: ["/bin/sh", "-c", "while : ;do curl http://localhost:80/;
sleep 10; done"]
```



- `kubectl create -f 3-2-1_pod.yaml`
- `kubectl logs example -c centos`
- `kubectl describe pods example`
- un pod está asociado con un **service account** que provee una identidad para los procesos que ejecutan el pod. Lo controla el service account y el token controllers en el API Server
- montará un volumen de solo lectura para cada contenedor en la ruta `/var/run/secrets/kubernetes.io/serviceaccount` con el contenido del token al API access
- listar los service account:

```
kubectl get serviceaccounts
```

- ReplicaSet:
- vigila que el número de pods (replica pods) están siempre bien y en ejecución en el cluster
- ejemplo RS:

3-2-2_rs.yml

```
// an example for RS spec
# cat 3-2-2_rs.yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx
spec:
  replicas: 2
  selector:
```

```
matchLabels:
  project: chapter3
matchExpressions:
- {key: version, operator: In, values: ["0.1", "0.2"]}
template:
  metadata:
    name: nginx
    labels:
      project: chapter3
      service: web
      version: "0.1"
  spec:
    containers:
    - name: nginx
      image: nginx
      ports:
      - containerPort: 80
```

- `kubectl create -f 3-2-2_rs.yml`

- `kubectl get rs`

- `kubectl get pods`

- si creásemos un pod a mano que cumpliera con los criterios del **replicaSet** (label version:0.1), detectará que hay uno de más y lo eliminará:

- [3-2-2_rs_self_created_pod.yaml](#)

```
apiVersion: v1
kind: Pod
metadata:
  name: our-nginx
  labels:
    project: chapter3
    service: web
    version: "0.1"
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

- `kubectl create -f 3-2-2_rs_self_created_pod.yaml`

- podemos modificar el número de réplicas con:

```
kubectl edit rs nginx
```

Last update: 13/04/2020 13:11 info:libros:devops-kubernetes:cap3 <https://miguelangel.torresegea.es/wiki/info:libros:devops-kubernetes:cap3?rev=1586808678>

- `kubectl describe rs nginx`

- `kubectl delete rs nginx`

Deployments

From: <https://miguelangel.torresegea.es/wiki/> - **miguel angel torres egea**

Permanent link: <https://miguelangel.torresegea.es/wiki/info:libros:devops-kubernetes:cap3?rev=1586808678>

Last update: **13/04/2020 13:11**

