



# Kubernetes

Intro a la orquestación de contenedores!

**Catherin Cruz**

@UserTwoGG



# Monolitos vs Microservicios

---



**Monolitos** son aplicaciones enfocadas en generar un único entregable, la mayoría de productos inician con este enfoque, pero al momento de adicionar funcionalidades...

- Presentan problemas de desarrollo y despliegue
- Demasiadas interdependencias entre funcionalidades
- Fuertemente dependientes de la red y los recursos

La solución actual, es dividir por **Micrsoservicios** enfocados al mismo dominio del negocio, donde sea más fácil modificar unidades de código, y desplegarlas independientemente.



# Bases del DevOps

---



## Contenedores

Permite especificar con exactitud qué partes de tu entorno necesitas y cuales quieres ejecutar. Son fáciles de empaquetar y mover, usualmente usan los recursos del computador de una forma eficiente. Desacoplan una aplicación de sus dependencias del sistema operativo.



## Docker

Virtualización a nivel de sistema operativo, es primordialmente desarrollado para ambientes Linux donde usa la aislación de recursos propias del Kernel, lo que le permite la ejecución independiente de contenedores en una sola instancia de Linux , evitando la sobrecarga de una máquina virtual.

# Google y Kubernetes



Sistema de orquestación de código abierto, para contenedores Docker creado por **Google**. Inició su desarrollo desde el 2003 como Omega, luego Borg.

Maneja el despliegue de estos contenedores en un sistema Cluster, controlando los recursos y la red.

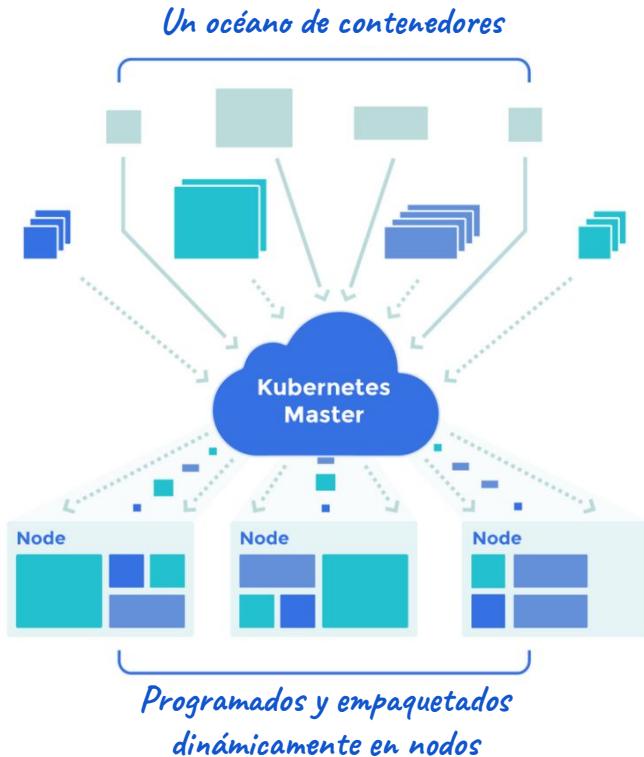
Simplifica las tareas de seguimiento, despliegue, escalabilidad, configuración, y versionamiento. Usa etiquetas para identificar objetos DevOps.

Ahora manejado por CNCF Cloud Native Computing Foundation. Soportado por AWS, Docker y Azure.

# Beneficios de la plataforma



- Agrupación de contenedores
- Auto-recuperación
- Auto-escalabilidad
- Administración DNS
- Balanceo de cargas
- Ejecución de actualizaciones
- Monitoreo de recursos y logs



# Nodo Maestro

---



Panel de control del sistema, encargado de monitorear, hacer cambios, programar las tareas, y responder a los eventos. Compuesto por cuatro funcionalidades:

**API Server:** Único componente del nodo maestro que permite interacción del usuario, mediante la exposición de servicios REST y consumo de archivos JSON.

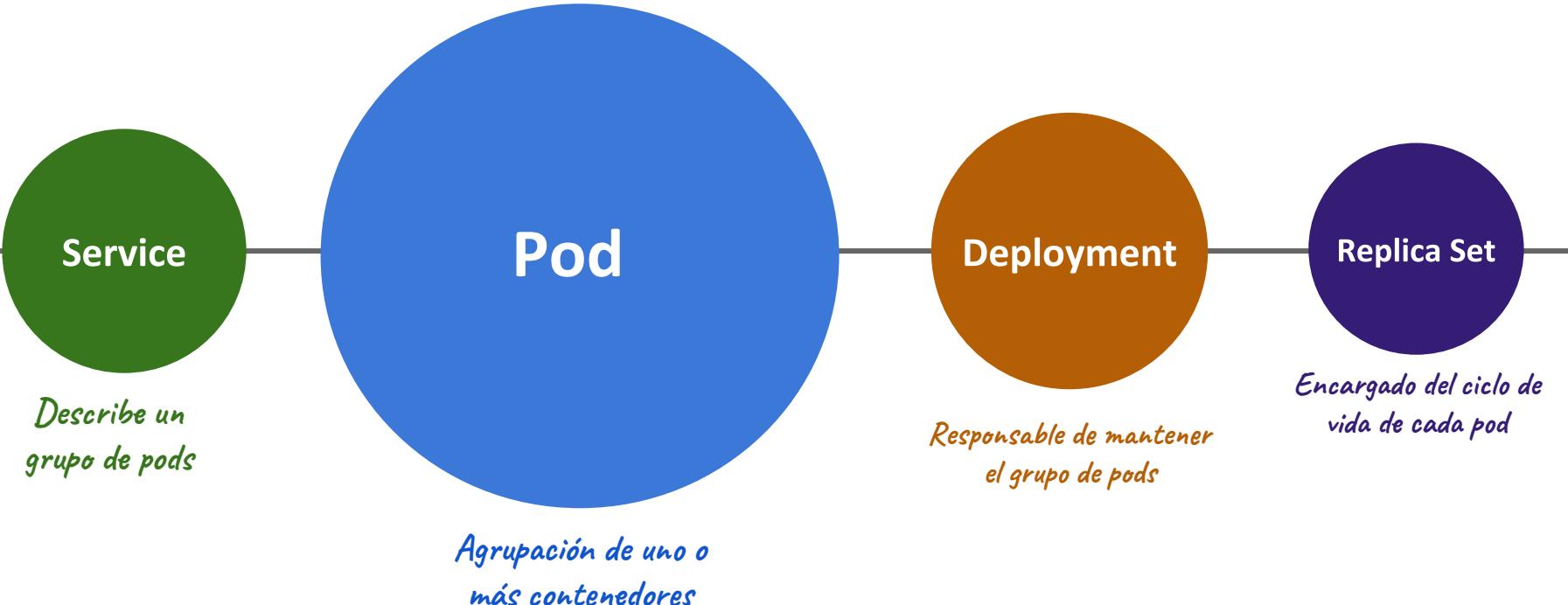
**Data Store:** Base de datos de tipo llave valor, tipo “etcd”, es robusta, consistente y brinda alta-disponibilidad.

**Controller Manager:** Administra las tareas del clúster, incluyendo los controladores de: nodos, replicación, endpoints, cuentas, y tokens.

**Scheduler:** Realiza el control y seguimiento de los nuevos pods (ya sea un uno o un grupo), y les asigna nodos.

# Los objetos de Kubernetes

---





# Objetos: Pod

---

Unidad básica de construcción de k8s, desplegados como una unidad en un nodo del cluster.

Agrupación de uno o más contenedores, los cuales comparten almacenamiento y servicios de red, como dirección IP y puerto.

Los pod son objetos con un **ciclo de vida definido**, luego de presentar una falla o ser programados para redespliegue dejan de existir.





# Objetos: Replica Set



Los pods no se despliegan, y es trabajo del replica set asegurarse que todos los pods requeridos están en ejecución.

Pueden manejar unidades o grandes grupos de pods. Es recomendado **asignar etiquetas** de identificación por grupos y unidades.

Todo este proceso de control y despliegue está **automatizado**, y no es necesaria intervención de usuario más que para la configuración, lo cual facilita el manejo de la capacidad de escala.

# Objetos: Service

---



Abstracción que **define un grupo lógico de pods** y las reglas de acceso a los mismos.

Se encarga de ocultar múltiples pods detrás de una dirección de red.

El encargarse de la identificación de los pods, le permite realizar el balanceo de cargas.

Métodos de identificación de servicios:

- Variables de entorno (kubelet node agent)
- Servicio propio de DNS (cluster add-in)



# Objetos: Deployment



LoadingArtist.com



Controlador que **describe y mantiene el estado deseado** del grupo de pods y las réplicas, mediante el uso de archivos yaml.

Los archivos yaml puede ser creados manualmente o de forma automática al ejecutar comandos de despliegue.

## Buenas prácticas:

- Agregar chequeo de estados
- imagen:versión por contenedor
- Asignar etiquetas descriptivas
- Agregar validaciones de inicio
- Usar canales de comunicación

# Katacoda y Minikube



**Katacoda**

More Information about Katacoda   Search

Launch Single Node Kubernetes Cluster

Step 1 of 4

## Step 1 - Start Minikube

Minikube has been installed and configured in the environment. Check that it is properly installed, by running the *minikube version* command:

```
minikube version ↵
```

Start the cluster, by running the *minikube start* command:

```
minikube start ↵
```

Great! You now have a running Kubernetes cluster in your online terminal. Minikube started a virtual machine for you, and a Kubernetes cluster is now running in that VM.

CONTINUE

Terminal   Dashboard   +

Your Interactive Bash Terminal.

```
$  
$ minikube version  
minikube version: v0.25.0  
$ minikube start  
There is a newer version of minikube available (v0.25.2). Download it here:  
https://github.com/kubernetes/minikube/releases/tag/v0.25.2  
  
To disable this notification, run the following:  
minikube config set WantUpdateNotification false  
Starting local Kubernetes v1.9.0 cluster...  
Starting VM...  
Getting VM IP address...  
Moving files into cluster...  
Setting up certs...  
Connecting to cluster...  
Setting up kubeconfig...  
Starting cluster components...  
Kubectl is now configured to use the cluster.  
Loading cached images from config file.  
$ [ ]
```

<https://www.katacoda.com/courses/kubernetes/launch-single-node-cluster>



# Comandos básicos

---

Minikube facilita el despliegue de K8s localmente, ejecuta un cluster de un nodo en una MV.

```
$ minikube start
```

```
$ minikube dashboard
```

Kubectl interface de línea de comandos ejecutables dentro del cluster de k8s.

```
$ kubectl cluster-info
```

```
$ kubectl get nodes --output wide --watch
```

```
$ kubectl get all
```



# Despliegues

---

```
$ kubectl run k8s --image=twogghub/k8s-workshop:1.1-rolling
```

```
$ kubectl get deployment k8s --output wide
```

```
$ kubectl expose deployment k8s --port=8080  
--external-ip=$(minikube ip) --type=LoadBalancer
```

En Katacoda click en + y “Select port to view on Host1”, para ver en un nuevo tab el puerto **8080**

```
$ kubectl describe service k8s
```

```
$ kubectl get pods,services,deployments --output wide
```

# Interacción con el Pod

---



```
$ kubectl run ghost --image=ghost:0.9
```

```
$ kubectl expose deployment ghost --port=2368 --labels="version=bash"  
--external-ip=$(minikube ip) --type=LoadBalancer
```

```
$ kubectl label pod <pod-name> dev=twogg
```

```
$ kubectl logs <pod-name>
```

```
$ kubectl get pod <pod-name> --output=json
```

```
$ kubectl exec -ti <pod-namename> /bin/bash
```



# Actualizar y Escalar

```
$ kubectl set image deployment k8s k8s=twogithub/k8s-workshop:1.2-yaml
```

```
$ kubectl scale --replicas=3 deployment k8s
```

La bandera **--watch** permite el seguimiento continuo de los eventos de un grupo de pods.

```
$ kubectl get pods --output wide --watch
```

```
kubectl get pods --output wide
```

```
$ kubectl delete pod <pod-name>
```



# Validación de estados

---

```
$ kubectl create -f https://url_repo/despliegues/yamls/k8sdp.yaml
```

```
$ kubectl scale --replicas=3 deployment k8sdp
```

La bandera **--watch** permite el seguimiento continuo de los eventos de un grupo de pods.

```
$ kubectl get pods --output wide --watch --selector="env=k8sdp"
```

```
$ kubectl set image deployment k8sdp k8sdp=twogithub/k8s-workshop:1.2-yaml
```

```
$ kubectl rollout undo deployment/k8sprod
```



# Limpieza del Cluster

---

```
$ kubectl delete pod <pod-name>
```

Cada pod es generado con base al archivo de despliegue, cada vez que se elimina un pod, k8s va a desplegar un reemplazo, siempre busca mantener la cantidad de réplicas definidas deseadas. Para eliminar un pod permanente primero se debe modificar o eliminar el despliegue.

```
$ kubectl delete deployment k8s
```

```
$ kubectl delete services,pods,deployments --all
```

# Links y Referencias

---



- Introducción comandos Kubectl  
<https://github.com/twogg-git/k8s-intro>
- Documentación oficial  
<https://kubernetes.io/docs/home/>
- Kubernetes con ejemplos  
<http://kubernetesbyexample.com/>
- Cursos Katacoda  
<https://www.katacoda.com/learn>
- Taller comandos Kubectl  
<https://github.com/twogg-git/k8s-workshop>



Catherin Cruz  
@UserTwoGG