Módulo 5: Arquitectura y escalabilidad

Orquestación de contenedores con Kubernetes

aspasia
LA FORMACIÓN DE TU FUTURO

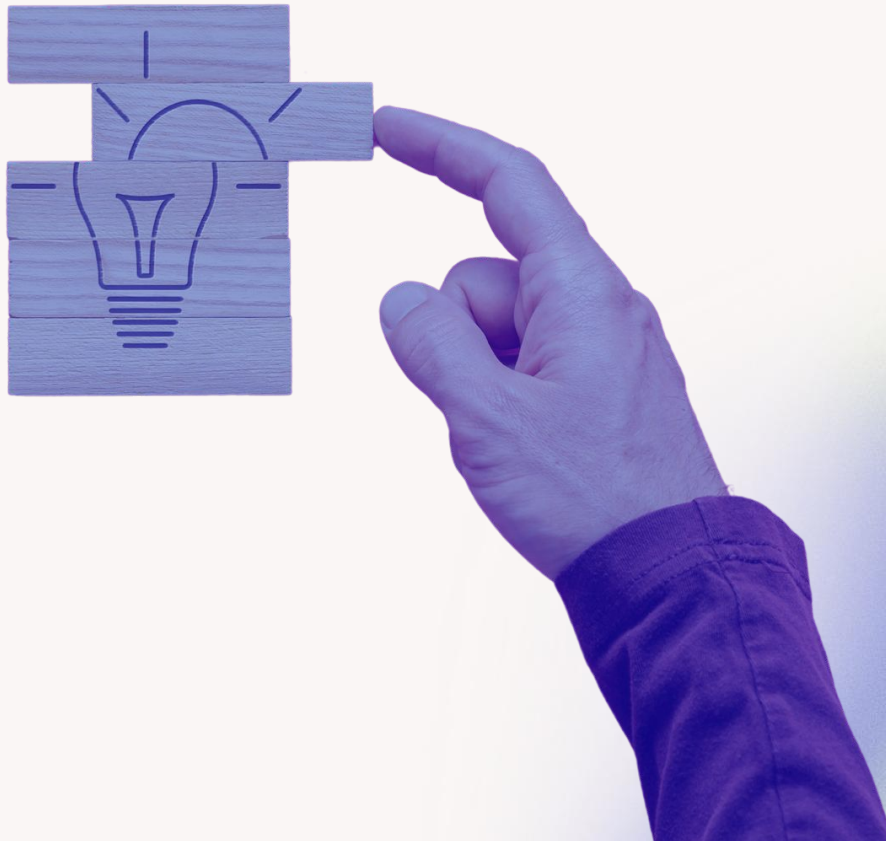
MÓDULO 5

- Principios fundamentales de diseño de una arquitectura
- Arquitecturas monolíticas y microservicios
- Fundamentos de tecnologías de contenedores
- ***Orquestación de contenedores con Kubernetes***
- Registro de contenedores
- Platform as a Service, Backend as a Service y Frontend as a Service

aspasia
LA FORMACIÓN DE TU FUTURO

Objetivos

Aprender los fundamentos de Kubernetes como orquestador de contenedores, su arquitectura, comandos clave y buenas prácticas para desplegar y escalar aplicaciones en entornos productivos.



aspasia
LA FORMACIÓN DE TU FUTURO

¿Por qué es necesario utilizar herramientas de orquestación de contenedores como Kubernetes?

aspasia
LA FORMACIÓN DE TU FUTURO

Introducción a Kubernetes

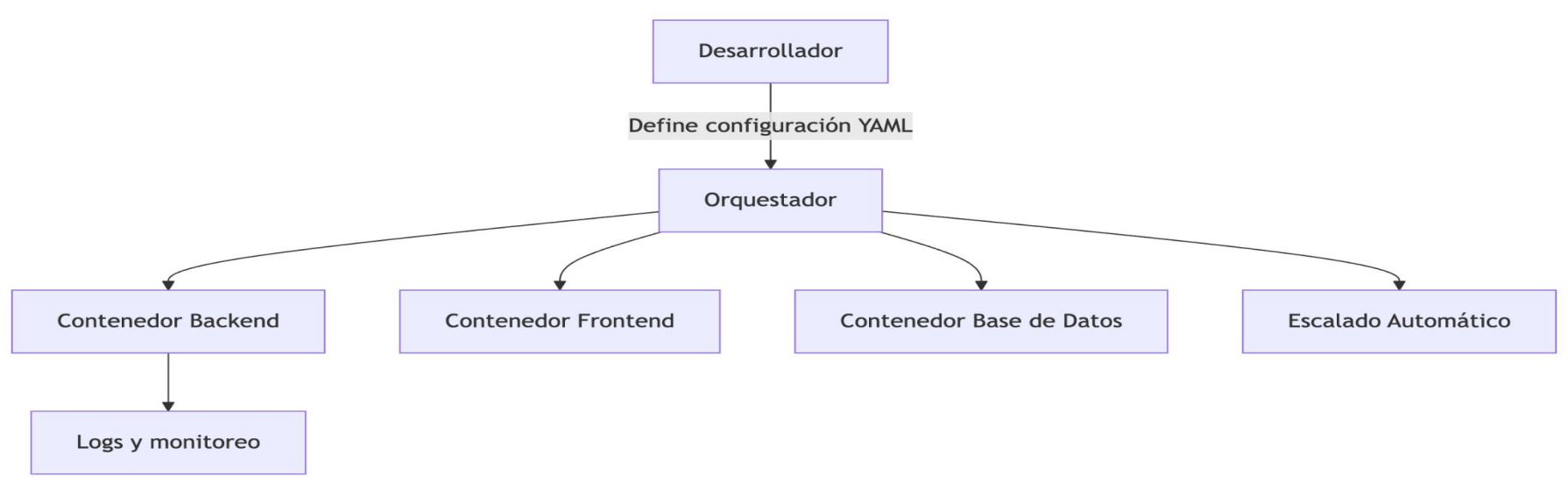
Introducción a la Orquestación de Contenedores

aspasia
LA FORMACIÓN DE TU FUTURO

📌 La orquestación de contenedores es el proceso de automatizar el despliegue, la gestión, el escalado y la red de aplicaciones en contenedores. A medida que las arquitecturas basadas en microservicios crecen en complejidad, surge la necesidad de coordinar múltiples contenedores distribuidos en distintos entornos.

Introducción a la Orquestación de Contenedores

aspasia
LA FORMACIÓN DE TU FUTURO



Introducción a la Orquestación de Contenedores

aspasia
LA FORMACIÓN DE TU FUTURO

✅ **¿Por qué se necesita un orquestador?**

- Gestionar decenas o cientos de contenedores automáticamente.
- Asegurar alta disponibilidad y recuperación ante fallos.
- Escalar aplicaciones bajo demanda.
- Desplegar versiones actualizadas sin tiempo de inactividad.

📖 **Ejemplo simple**:Si una aplicación consta de varios microservicios (backend, frontend, base de datos), el orquestador se encarga de:

- Iniciar los contenedores en el orden correcto.
- Monitorizarlos.
- Reemplazar los fallidos.
- Balancear la carga entre instancias.

¿Qué es Kubernetes y para qué se utiliza?

aspasia
LA FORMACIÓN DE TU FUTURO

📌 **Kubernetes** (también conocido como *K8s*) es un sistema de orquestación de contenedores **de código abierto** desarrollado originalmente por Google y ahora gestionado por la Cloud Native Computing Foundation (CNCF). Automatiza el despliegue, escalado y operación de aplicaciones en contenedores.

¿Qué es Kubernetes y para qué se utiliza?

aspasia
LA FORMACIÓN DE TU FUTURO

✅ **Funcionalidades clave de Kubernetes:**

- **Gestión de clústeres** de contenedores distribuidos.
- **Autoescalado** de cargas según demanda.
- **Reinicio automático** de contenedores fallidos.
- **Rolling updates** sin tiempo de inactividad.
- **Balanceo de carga y descubrimiento de servicios**.

📖 **Ejemplo práctico:**

Un equipo DevOps despliega una aplicación en un clúster Kubernetes con 3 nodos. Si uno falla, Kubernetes reprograma automáticamente los pods en los nodos disponibles.

Otras alternativas a Kubernetes

aspasia
LA FORMACIÓN DE TU FUTURO

📌 Aunque Kubernetes es el líder del mercado, existen otros orquestadores que pueden ser considerados:

Herramienta	Facilidad	Escalabilidad
Kubernetes	Media	Alta
Docker Swarm	Alta	Media
Nomad	Alta	Alta
OpenShift	Media	Alta

✅ **Alternativas relevantes**:

- **Docker Swarm**: Solución integrada de Docker, más simple pero con menor robustez.
- **Nomad (HashiCorp)**: Ligero, flexible, permite ejecutar no solo contenedores sino también binarios y máquinas virtuales.
- **Apache Mesos/Marathon**: Potente pero con mayor complejidad de configuración.
- **OpenShift**: Plataforma empresarial basada en Kubernetes, con extras de seguridad y gestión.

aspasia
LA FORMACIÓN DE TU FUTURO

¿Cuáles son las ventajas que ofrece Kubernetes frente a otros orquestadores de contenedores?

aspasia
LA FORMACIÓN DE TU FUTURO

Arquitectura y elementos de Kubernetes

Arquitectura de Kubernetes: Nodos Master y Worker

aspasia
LA FORMACIÓN DE TU FUTURO

📌 Kubernetes se organiza como un clúster compuesto por múltiples nodos. Hay **dos tipos de nodos** principales:

✅ **Nodo Master (Control Plane)**:

- Gestiona el estado deseado del clúster.
- Planea, programa y supervisa los contenedores.

✅ **Componentes del Master:**

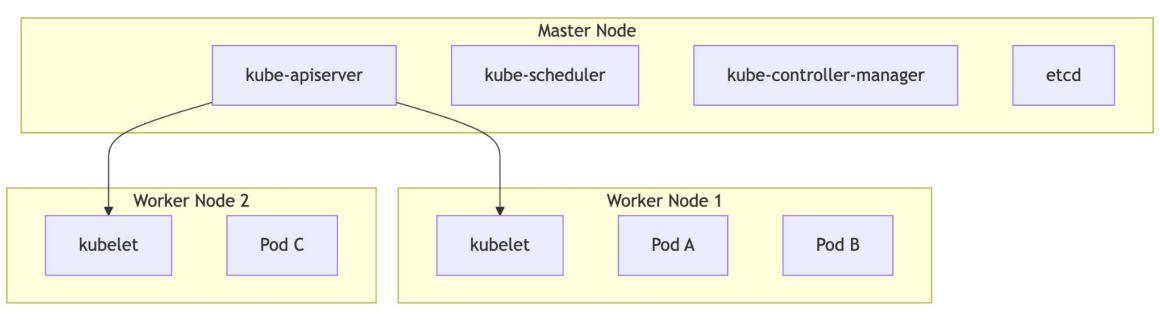
- kube-apiserver: interfaz de comunicación con Kubernetes.
- kube-scheduler: decide en qué nodo se ejecutan los pods.
- kube-controller-manager: aplica decisiones del scheduler.
- etcd: base de datos distribuida para el estado del clúster.

Arquitectura de Kubernetes: Nodos Master y Worker

aspasia
LA FORMACIÓN DE TU FUTURO

✅ **Nodos Worker**:

- Ejecutan las aplicaciones (pods).
- Cada nodo worker incluye:
  - kubelet: agente que corre en cada nodo y reporta al master.
  - kube-proxy: gestiona la red.
  - **Runtime de contenedores**: como Docker o containerd.



Pod y ReplicaSet

aspasia
LA FORMACIÓN DE TU FUTURO

📌 **Pod**: Unidad mínima de ejecución en Kubernetes, puede contener uno o más contenedores que comparten red y almacenamiento.

📌 **ReplicaSet**: Asegura que haya una cantidad específica de **réplicas activas** de un pod.

✅ **Ventajas:**

- Alta disponibilidad automática.
- Tolerancia a fallos.
- Autorreemplazo de pods caídos.

📖 **Ejemplo YAML para 3 réplicas:**

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: ejemplo-rs

spec:
  replicas: 3
  selector:
    matchLabels:
      app: demo
  template:
    metadata:
      labels:
        app: demo
    spec:
      containers:
        - name: nginx
          image: nginx
```

Deployment

aspasia
LA FORMACIÓN DE TU FUTURO

📌 Un **Deployment** es un recurso que gestiona automáticamente los ReplicaSets y permite actualizaciones declarativas (rolling updates).

✅ **Beneficios:**

- Control de versiones.
- Rollbacks automáticos.
- Cambios sin downtime.

📖 **Ejemplo:**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deploy

spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.21
```

Service

aspasia
LA FORMACIÓN DE TU FUTURO

📌 Un **Service** en Kubernetes expone los pods a través de una dirección IP estable o DNS interno.

✅ **Tipos de Service:**

- **ClusterIP**: Solo accesible dentro del clúster.
- **NodePort**: Abre un puerto en cada nodo.
- **LoadBalancer**: Usa balanceadores externos (en la nube).
- **ExternalName**: Redirige a un nombre DNS externo.

📖 **Ejemplo:**

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-svc

spec:
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 80
  type: NodePort
```

DaemonSet

aspasia
LA FORMACIÓN DE TU FUTURO

📌 Un **DaemonSet** asegura que una copia de un pod se ejecute en **todos los nodos** (o un subconjunto).

✅ **Casos de uso comunes:**

- Logs.
- Monitoreo (como Prometheus Node Exporter).
- Configuración de red.

Ejemplo de DaemonSet:

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: node-logger
```

```yaml
spec:
  selector:
    matchLabels:
      app: logger
  template:
    metadata:
      labels:
        app: logger
    spec:
      containers:
        - name: fluentd
          image: fluent/fluentd
```

Jobs

aspasia
LA FORMACIÓN DE TU FUTURO

📌 Un **Job** ejecuta **una tarea finita** (por ejemplo, migración de base de datos o limpieza de archivos). Asegura que el proceso se complete exitosamente al menos una vez.

✅ **Variaciones:**

- **Jobs simples**: ejecución única.
- **Parallel Jobs**: múltiples ejecuciones simultáneas.
- **CronJobs**: Jobs programados (como tareas CRON).

Ejemplo de Job:

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: hello-job
spec:
  template:
    spec:
      containers:
        - name: hello
          image: busybox
          command: ["echo", "Hola desde Kubernetes"]
      restartPolicy: Never
```

aspasia
LA FORMACIÓN DE TU FUTURO

¿Cómo se relacionan los componentes principales de la arquitectura de Kubernetes?

aspasia
LA FORMACIÓN DE TU FUTURO

Instalación y configuración de un clúster

Instalación local con Minikube

aspasia
LA FORMACIÓN DE TU FUTURO

📌 **Minikube** es una herramienta que permite crear un clúster Kubernetes de un solo nodo de forma local.

✅ **Requisitos previos**:

- VirtualBox o Docker
- kubectl
- minikube instalado

🛠️ **Pasos de instalación** (Linux/macOS/Windows con Docker):

```
# Instalar Minikube
curl -LO
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64
/usr/local/bin/minikube

# Iniciar el clúster usando Docker como driver
minikube start --driver=docker

# Verificar que el clúster funciona
kubectl get nodes
```

Instalación con KIND (Kubernetes IN Docker)

aspasia
LA FORMACIÓN DE TU FUTURO

📌 **KIND** permite ejecutar clústeres Kubernetes dentro de contenedores Docker.

✅ **Ventajas**:

- Ligero y rápido.
- Ideal para testing de pipelines CI/CD.

🛠️ **Pasos básicos:**

```
# Instalar KIND
GO111MODULE="on" go install sigs.k8s.io/kind@v0.20.0

# Crear un clúster básico
kind create cluster

# Verificar los nodos
kubectl get nodes
```

📖 **Configuración avanzada**:

```
# kind-config.yaml
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4

nodes:
  - role: control-plane
  - role: worker

kind create cluster --config kind-config.yaml
```

Plataformas gestionadas: EKS, GKE y AKS

aspasia
LA FORMACIÓN DE TU FUTURO

📌 Estas plataformas ofrecen **clústeres gestionados en la nube**, eliminando la necesidad de configurar infraestructura compleja.

Plataforma	Proveedor	Características
EKS	AWS	Alta integración con servicios de AWS
GKE	Google	Escalabilidad automática y seguridad integrada
AKS	Azure	Fácil integración con Azure DevOps y AD

🛠️ Ejemplo: creación de clúster con GKE

```
# Autenticarse
gcloud auth login

# Crear clúster
gcloud container clusters create my-cluster \
  --num-nodes=3 \
  --zone=us-central1-a

# Obtener credenciales para kubectl
gcloud container clusters get-credentials my-cluster \
  --zone=us-central1-a
```

Uso de kubectl y consola Kubernetes

aspasia
LA FORMACIÓN DE TU FUTURO

📌 kubectl es la **herramienta de línea de comandos oficial** para gestionar recursos en Kubernetes.

✅ **Comandos básicos**:

```
# Ver clúster actual
kubectl config current-context

# Listar pods, servicios y despliegues
kubectl get pods
kubectl get svc
kubectl get deployments

# Aplicar un manifiesto YAML
kubectl apply -f mi-app.yaml

# Obtener detalles de un recurso
kubectl describe pod <nombre>
```

✅ **Consolas alternativas**:

- **Lens**: UI para múltiples clústeres.
- **Octant**: dashboard local para desarrolladores.
- **Kubernetes Dashboard**: UI web oficial.

aspasia
LA FORMACIÓN DE TU FUTURO

¿Cuáles son las ventajas y desventajas de utilizar soluciones locales como Minikube frente a plataformas gestionadas?

aspasia
LA FORMACIÓN DE TU FUTURO

Despliegue de aplicaciones en Kubernetes

Creación y gestión de Pods (manifiesto)

aspasia
LA FORMACIÓN DE TU FUTURO

📌 En Kubernetes, el **Pod** es la unidad mínima de despliegue. Un Pod contiene uno o más contenedores que comparten red y almacenamiento.

✅ **Estructura de un manifiesto básico de Pod**:

```yaml
# pod-nginx.yaml

apiVersion: v1
kind: Pod

metadata:
  name: nginx-pod
  labels:
    app: nginx

spec:
  containers:
    - name: nginx-container
      image: nginx:latest
      ports:
        - containerPort: 80
```

🛠️ **Comandos para aplicar y gestionar el Pod**:

```
kubectl apply -f pod-nginx.yaml
kubectl get pods
kubectl describe pod nginx-pod
kubectl delete pod nginx-pod
```

Replicación y escalado

aspasia
LA FORMACIÓN DE TU FUTURO

📌 Para lograr **alta disponibilidad y escalabilidad**, se usa un **ReplicaSet** o un **Deployment**.

✅ Ejemplo de Deployment con 3 réplicas:

```yaml
# deployment-nginx.yaml
apiVersion: apps/v1
kind: Deployment

metadata:
  name: nginx-deployment

spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.25
          ports:
            - containerPort: 80
```

🛠️ Comandos útiles:

```
kubectl apply -f deployment-nginx.yaml
kubectl get deployments
kubectl get rs
kubectl scale deployment nginx-deployment --replicas=5
kubectl rollout restart deployment nginx-deployment
```

Exposición de aplicaciones

aspasia
LA FORMACIÓN DE TU FUTURO

📌 Para acceder a un Pod desde fuera del clúster, se utiliza un **Service**.

✅ **Tipos de Service**:

- ClusterIP: acceso interno.
- NodePort: acceso externo a través de un puerto del nodo.
- LoadBalancer: acceso público (usado en la nube).

✅ **Manifiesto de un Service tipo NodePort**:

```yaml
# service-nginx.yaml
apiVersion: v1
kind: Service

metadata:
  name: nginx-service

spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30036
```

🛠️ Aplicar y probar el acceso:

```
kubectl apply -f service-nginx.yaml
minikube service nginx-service
```

aspasia
LA FORMACIÓN DE TU FUTURO

¿Cómo se gestionan los procesos de replicación y escalado en Kubernetes y qué papel juegan en la alta disponibilidad?

aspasia
LA FORMACIÓN DE TU FUTURO

Gestión del ciclo de vida de las aplicaciones

Actualización de aplicaciones

aspasia
LA FORMACIÓN DE TU FUTURO

📌 Kubernetes permite actualizaciones sin downtime mediante el **rolling update**, gracias a los Deployments.

✅ Modificamos el Deployment existente para actualizar la imagen de Nginx:

```yaml
# deployment-nginx-v2.yaml
apiVersion: apps/v1
kind: Deployment

metadata:
  name: nginx-deployment

spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.26
          ports:
            - containerPort: 80
```

🛠️ Aplicación de la actualización

```
kubectl apply -f deployment-nginx-v2.yaml
kubectl rollout status deployment
nginx-deployment
kubectl rollout history deployment
nginx-deployment
```

Autoescalado

aspasia
LA FORMACIÓN DE TU FUTURO

📌 Kubernetes permite escalar automáticamente según la **carga del CPU o memoria** mediante el **Horizontal Pod Autoscaler (HPA)**.

✅ **Creamos un HPA para escalar entre 2 y 5 réplicas**:

```
kubectl autoscale deployment nginx-deployment --cpu-percent=50 --min=2 --max=5
kubectl get hpa
```

🛠️ **Simulación de carga para observar el escalado (opcional)**:

```
kubectl run -i --tty load-generator --image=busybox /bin/sh
while true; do wget -q -O- http://nginx-service; done
```

Gestión de almacenamiento

aspasia
LA FORMACIÓN DE TU FUTURO

📌 Kubernetes permite acoplar almacenamiento persistente a los Pods mediante **Persistent Volumes (PV)** y **Persistent Volume Claims (PVC)**.

✅ Creamos un volumen local simple (en Minikube):

```yaml
# pvc-demo.yaml
apiVersion: v1
kind:
PersistentVolumeClaim

metadata:
  name: demo-pvc
```

```yaml
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
```

✅ Lo conectamos a un contenedor:

```yaml
# pod-pvc.yaml
apiVersion: v1
kind: Pod

metadata:
  name: nginx-pvc

spec:
  containers:
    - name: nginx
      image: nginx
      volumeMounts:
        - mountPath:
"/usr/share/nginx/html"
          name: html-storage
```

```yaml
  volumes:
    - name: html-storage
      persistentVolumeClaim:
        claimName: demo-pvc
```

🛠️ Aplicación de los manifiestos:

```
kubectl apply -f pvc-demo.yaml
kubectl apply -f pod-pvc.yaml
```

aspasia
LA FORMACIÓN DE TU FUTURO

¿Cómo se gestionan los procesos de replicación y escalado en Kubernetes y qué papel juegan en la alta disponibilidad?

aspasia
LA FORMACIÓN DE TU FUTURO

Configuraciones y secrets

ConfigMaps

aspasia
LA FORMACIÓN DE TU FUTURO

📌 **ConfigMaps** permiten separar la configuración del contenedor de la imagen, permitiendo una mayor flexibilidad y portabilidad.

✅ Ejemplo de un ConfigMap que contiene parámetros de configuración:

```yaml
# configmap-demo.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config

data:
  APP_MODE: "production"
  APP_DEBUG: "false"
```

🛠️ Aplicación y verificación:

```
kubectl apply -f configmap-demo.yaml
kubectl apply -f
pod-with-configmap.yaml
kubectl logs configmap-pod
```

✅ Uso del ConfigMap dentro de un Pod:

```yaml
# pod-with-configmap.yaml
apiVersion: v1
kind: Pod

metadata:
  name: configmap-pod

spec:
  containers:
    - name: app
      image: busybox
      command: ["sh", "-c", "echo
Mode=$APP_MODE Debug=$APP_DEBUG &&
sleep 3600"]
```

```yaml
      env:
        - name: APP_MODE
          valueFrom:
            configMapKeyRef:
              name: app-config
              key: APP_MODE
        - name: APP_DEBUG
          valueFrom:
            configMapKeyRef:
              name: app-config
              key: APP_DEBUG
```

Secrets

aspasia
LA FORMACIÓN DE TU FUTURO

📌 Los **Secrets** permiten almacenar datos sensibles como contraseñas, tokens o claves de forma segura y encriptada.

✅ Creación de un Secret desde archivo o línea de comando:

```
kubectl create secret generic db-secret \
  --from-literal=username=admin \
  --from-literal=password=supersecure
```

🛠️ Aplicación y prueba:

```
kubectl apply -f pod-with-secret.yaml
kubectl logs secret-pod
```

✅ Uso del Secret como variables de entorno:

```
# pod-with-secret.yaml
apiVersion: v1
kind: Pod
metadata:
  name: secret-pod

spec:
  containers:
    - name: db-client
      image: busybox
      command: ["sh", "-c",
"echo USER=$DB_USER
PASS=$DB_PASS && sleep 3600"]
```

```
      env:
        - name: DB_USER
          valueFrom:
            secretKeyRef:
              name: db-secret
              key: username
        - name: DB_PASS
          valueFrom:
            secretKeyRef:
              name: db-secret
              key: password
```

Buenas prácticas y recomendaciones

aspasia
LA FORMACIÓN DE TU FUTURO

✅ **Uso correcto de ConfigMaps y Secrets:**

- Separar configuración del código y no hardcodear variables.
- Evitar incluir secrets en imágenes o repositorios.
- Usar RBAC para limitar el acceso a los secretos.
- Cifrar los secrets en reposo usando herramientas como **Sealed Secrets o Vault.**

✅ **Otras recomendaciones:**

- Versionar ConfigMaps/Secrets si los actualizas con frecuencia.
- Revisar el uso de envFrom para importar múltiples claves si es seguro.
- No usar ConfigMaps para datos sensibles.

Ejemplo final: aplicación web usando ConfigMap + Secret

aspasia
LA FORMACIÓN DE TU FUTURO

✅ Este patrón permite que la app acceda a su configuración **no sensible** desde el ConfigMap y a las **credenciales** desde el Secret.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp

spec:
  replicas: 1
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
        - name: web-container
          image: myorg/webapp:latest
          envFrom:
            - configMapRef:
                name: app-config
            - secretRef:
                name: db-secret
```

aspasia
LA FORMACIÓN DE TU FUTURO

¿Cómo contribuyen los ConfigMaps y Secrets a una gestión segura y flexible de la configuración en Kubernetes?

aspasia
LA FORMACIÓN DE TU FUTURO

Ejercicio Guiado: Despliegue de una Aplicación con Kubernetes

aspasia
LA FORMACIÓN DE TU FUTURO

Ejercicio Guiado: Despliegue de una Aplicación con Kubernetes

Kubernetes es la herramienta más utilizada hoy para **orquestar contenedores** en producción. En este ejercicio, crearás un **clúster local con Minikube**, desplegarás una aplicación web simple, la escalarás, la expondrás al mundo, y gestionarás su configuración y secretos de forma segura.

aspasia
LA FORMACIÓN DE TU FUTURO

Objetivos

- Instalar y ejecutar un clúster local con Minikube.
- Comprender los elementos clave de Kubernetes (Pod, Deployment, Service, ConfigMap, Secret).
- Desplegar una aplicación web contenedorizada en Kubernetes.
- Escalar, exponer y configurar la aplicación.
- Familiarizarse con comandos básicos de kubectl.

🛠️ Paso 1: Instalar Minikube y Kubect

aspasia
LA FORMACIÓN DE TU FUTURO

📌 **¿Qué haremos?**

- Prepararemos el entorno para usar Kubernetes localmente.

Instrucciones:

- Instalar kubectl:
  - https://kubernetes.io/docs/tasks/tools/
- Instalar Minikube:

```
curl -LO
https://storage.googleapis.com/minikube/releases/lat
est/minikube-linux-amd64
sudo install minikube-linux-amd64
/usr/local/bin/minikube
```

- Iniciar el clúster:

```
minikube start --driver=docker
```

- Verificar:

```
kubectl get nodes
```

🛠️ Paso 1: Instalar Minikube y Kubect

aspasia
LA FORMACIÓN DE TU FUTURO

3. Iniciar el clúster:

```
minikube start --driver=docker
```

4. Verificar:

```
kubectl get nodes
```

📦 Paso 2: Crear un Deployment y un Service

aspasia
LA FORMACIÓN DE TU FUTURO

📌 ¿Qué haremos?

- Desplegaremos una app Node.js de ejemplo desde Docker Hub.

Archivo: deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp-deployment

spec:
  replicas: 2
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
        - name: webapp
          image:
docker.io/nginxdemos/hello
          ports:
            - containerPort: 80
```

Archivo: service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: webapp-service

spec:
  type: NodePort
  selector:
    app: webapp
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30001
```

Comandos:

```
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
kubectl get all
```

✅ Verifica en el navegador con:

```
minikube service webapp-service
--url
```

📈 Paso 3: Escalar la aplicación

aspasia
LA FORMACIÓN DE TU FUTURO

📌 **¿Qué haremos?**

- Aumentaremos el número de réplicas.

```
kubectl scale deployment webapp-deployment --replicas=4
kubectl get pods
```

✅ Comprueba que ahora hay 4 pods activos.

🔁 Paso 4: Actualizar la aplicación

aspasia
LA FORMACIÓN DE TU FUTURO

📌 **¿Qué haremos?**

- Simularemos una actualización de versión.

```
kubectl set image deployment/webapp-deployment
webapp=nginxdemos/hello:plain-text
kubectl rollout status deployment/webapp-deployment
```

✅ Usa kubectl rollout undo si deseas revertir.

🔒 Paso 5: Configuración y secretos

aspasia
LA FORMACIÓN DE TU FUTURO

📌 **¿Qué haremos?**

- Crearemos un ConfigMap y un Secret, y los usaremos en un nuevo Pod.

```
kubectl create configmap webapp-config --from-literal=ENV=production
kubectl create secret generic webapp-secret --from-literal=PASSWORD=supersecure
```

✅ Visualiza:

```
kubectl describe configmap webapp-config
kubectl describe secret webapp-secret
```

✅ En una aplicación real se referenciarían como:

```yaml
env:
  - name: ENV
    valueFrom:
      configMapKeyRef:
        name: webapp-config
        key: ENV
  - name: PASSWORD
    valueFrom:
      secretKeyRef:
        name: webapp-secret
        key: PASSWORD
```

Preguntas finales

aspasia
LA FORMACIÓN DE TU FUTURO

- ¿Cuál es la diferencia entre un Pod y un Deployment?
- ¿Por qué es útil usar un ConfigMap o un Secret?
- ¿Qué ventajas ofrece Kubernetes frente a ejecutar contenedores manualmente con Docker?
- ¿Qué desafíos crees que enfrentaría esta app en producción?

Entregable:

aspasia
LA FORMACIÓN DE TU FUTURO

Cada equipo deberá:

- Los archivos deployment.yaml y service.yaml.
- Evidencia (capturas o comandos) de escalado, exposición y configuración.
- Diagrama de arquitectura del entorno desplegado.
- Una reflexión escrita sobre las ventajas y retos del uso de Kubernetes.

aspasia
LA FORMACIÓN DE TU FUTURO

Resumen de lo aprendido

- **Fundamentos de Kubernetes:** Plataforma de orquestación para administrar contenedores a gran escala con soporte para alta disponibilidad.
- **Elementos Clave:** Arquitectura basada en nodos, pods, deployments, services y jobs, permitiendo control y gestión del ciclo de vida de las aplicaciones.
- **Despliegue y Escalado:** Uso de manifiestos YAML, herramientas como kubectl, Minikube o KIND, y funciones como replicación y autoescalado.
- **Configuraciones y Seguridad:** Implementación de ConfigMaps y Secrets, junto con buenas prácticas para entornos gestionados como EKS, GKE y AKS.

aspasia
LA FORMACIÓN DE TU FUTURO

Próxima clase...

Registro de contenedores


dalid
aspasia
LA FORMACIÓN DE TU FUTURO