Trabajo final
Arquitectura de Software

Diseñar e implementar el despliegue completo de una aplicación compuesta por base de datos, backend, usando Helm para empaquetar y ArgoCD para gestionar la entrega continua.

1. Requisitos de la aplicación

La aplicación será un sistema de gestión de pedidos con:

1. Base de datos: PostgreSQL.
2. Backend: API en Java, Python o NodeJS .

2. Lo que deben entregar

Cada grupo debe implementar:

a. Chart de Helm

- Un único chart de Helm con subcharts para:
    - db → PostgreSQL (usar chart oficial de Bitnami como dependencia).
    - backend → Deployment + Service + ConfigMap/Secret para conexión DB.
- Variables (`values.yaml`) que permitan definir:
    - Imagen y tag de cada componente.
    - Credenciales de base de datos.
    - Réplicas de cada componente.
    - Configuración de recursos (CPU/memoria).

b. Recursos Kubernetes obligatorios

- Deployment → backend.
- Service → ClusterIP para backend.
- Ingress →
    - `/api/*` → backend
- PersistentVolumeClaim para datos de PostgreSQL.

- ConfigMap → configuración no sensible del backend (por ejemplo, URL de la DB).
- Secret → credenciales de DB.

c. Integración con ArgoCD
Un repositorio Git con:

```
charts/
  pedido-app/
    Chart.yaml
    values.yaml
environments/
   prod/
    application.yaml  (Definición de ArgoCD Application)
```

- Un entorno gestionado por ArgoCD:
    - `my-tech`
- Cada vez que cambie el `values.yaml` en Git a través de Jenkins, ArgoCD debe sincronizar el clúster automáticamente.

3. Criterios de evaluación

1. Funcionamiento end-to-end – La aplicación debe ser accesible vía Ingress Implementar un *Horizontal Pod Autoscaler* para el backend.
2. Persistencia de datos – Los pedidos deben mantenerse aunque el pod de PostgreSQL se reinicie.
3. Uso de buenas prácticas:
    - Límite y solicitud de recursos (`resources.limits` / `resources.requests`).
    - No exponer credenciales en texto plano.
4. CI completo con jenkins al detectar un cambio (o lanzarlo manualmente) en la API genere una versión nueva de la imagen de la aplicaci´øn
5. Automatización GitOps – Cambios en Git reflejados automáticamente en Kubernetes vía ArgoCD.

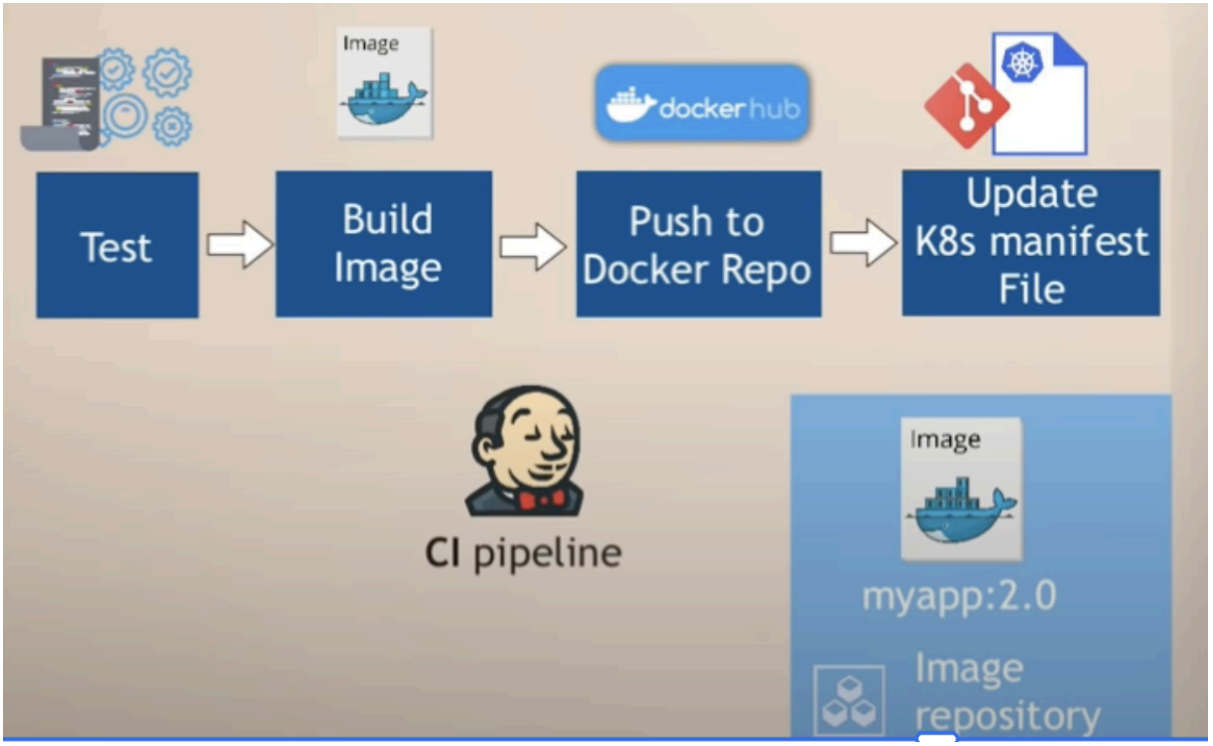
Trabajo final
Arquitectura de Software

4. Entregables

- Repositorio Git con el chart de Helm y definiciones de ArgoCD.
- Documento (README) explicando:
  - Cómo instalar el chart manualmente con Helm.
  - Cómo está configurado ArgoCD para sincronizar.
  - Endpoints de acceso (backend).

- Demostración en vivo:

  - Al crear un nuevo endpoint o eliminar uno y realizar un commit/push sobre el repositorio se debe lanzar a través de jenkins una tarea que se encargue de ir hasta la actualización de la imagen/tag en `values.yaml` y mostrar cómo ArgoCD actualiza el clúster sin comandos manuales.



Trabajo final
Arquitectura de Software

Rúbrica

Categoría	Descripción	Peso	Escala de Evaluación
1. Demostración en vivo	Proceso de CI/CD completo	20 pts	**0 pts:** No hay CI/CD **10 pts:** Solo existe CI. **15 pts:** Está el CI y CD sin argo. **20 pts:** todo el proceso de CI/CD completo.
2. Despliegue de recursos obligatorios	Existencia y correcta configuración de Deployment, Service, Ingress, PVC, ConfigMap, Secret.	25 pts	**0 pts:** Faltan más de la mitad de los recursos. **15 pts:** Todos los recursos presentes pero con errores funcionales. **20 pts:** Recursos completos y funcionales, pero sin buenas prácticas. **25 pts:** Recursos completos, funcionales y siguiendo buenas prácticas (nombres, labels, etc.).
3. Configuración de ArgoCD	Definición de `Application` por ambiente, apuntando al repositorio correcto y sincronización automática.	20 pts	**0 pts:** No hay configuración de ArgoCD. **10 pts:** Configuración manual y no funcional. **15 pts:** Configuración funcional pero requiere sincronización manual. **20 pts:** Configuración funcional, automática y documentada.

Trabajo final
Arquitectura de Software

4. Funcionamiento end-to-end	El Backend carga y este se conecta a la base de datos con persistencia.	20 pts	**0 pts:** No funciona o no hay integración. **10 pts:** Funciona parcialmente (backend aislados). **15 pts:** Funciona todo, pero sin persistencia de datos. **20 pts:** Funciona todo, con persistencia y rutas correctas en el Ingress.
5. Buenas prácticas y calidad	Uso de `resources`, seguridad en credenciales, etiquetas (`labels`), separación de ambientes, limpieza del repo.	10 pts	**0 pts:** No aplica buenas prácticas. **5 pts:** Aplica algunas pero de forma inconsistente. **8 pts:** Aplica casi todas. **10 pts:** Aplica todas las buenas prácticas de forma consistente.
6. Documentación y entrega	README claro, instrucciones para instalar manualmente con Helm y descripción de cómo ArgoCD sincroniza.	5 pts	**0 pts:** No hay README. **3 pts:** README incompleto o confuso. **5 pts:** README claro, con pasos reproducibles y diagramas si aplica.