🚢 Despliegue y Monitoreo Continuo para la Plataforma PortTrack

Nombre: Luis Montero
Archivo: Despliegue_M7_DevOps_LuisMontero.pdf
Módulo: 7 - Despliegue y Monitoreo Continuo

🎯 Objetivo

Diseñar e implementar una estrategia de despliegue continuo y monitoreo para **PortTrack**, una plataforma portuaria que gestiona barcos, rutas, carga y personal, asegurando **automatización**, **observabilidad**, **seguridad** y **resiliencia**.

🚢 1. Estrategia de Despliegue Continuo

✅ Tipo de despliegue elegido: Canary Release

- Permite liberar nuevas versiones a un porcentaje limitado de tráfico.
- Reduce riesgos: monitoreo inicial antes de liberar al 100%.
- Fácil rollback si se detectan errores.

⚙️ Herramientas seleccionadas:

- **GitHub Actions** para CI/CD por su integración directa con Git.
- **Jenkins** para tareas personalizadas y pipelines complejos.

♻️ Estrategia de rollback:

- Versionado de contenedores (Docker).
- Uso de etiquetas (v1, v1.1-canary, etc).
- Implementación de Helm rollback en Kubernetes o kubectl rollout undo.

🔐 2. Configuración de Entornos y Seguridad

🌍 Entornos diferenciados:

- dev → desarrollo local.
- staging → pruebas funcionales.
- test → pruebas automatizadas.
- prod → entorno en vivo.

🔑 Gestión de secretos:

- Uso de **AWS Secrets Manager** o **Vault**.
- En GitHub Actions: variables encriptadas (secrets.DB_PASSWORD).

🔐 **Seguridad en despliegue:**

- Autenticación con **AWS Cognito**.
- Revisión de imágenes con Trivy.
- Comunicación HTTPS entre microservicios.

📈 **3. Implementación de Monitoreo Continuo**

📊 **Herramientas utilizadas:**

- **Prometheus + Grafana** → monitoreo de métricas (CPU, RAM, errores).
- **Stack ELK (Elasticsearch + Logstash + Kibana)** → análisis de logs.
- **Fluentd** como alternativa ligera para recolección.

📋 **Logs y métricas monitoreadas:**

- Solicitudes HTTP (latencia, códigos de error).
- Uso de CPU por microservicio.
- Trazabilidad de rutas marítimas.

🚨 **Configuración de alertas:**

- Alertas por Grafana hacia Slack: errores 5xx, caída de pods, alta latencia.
- Dashboards por microservicio: barcos, carga, rutas, seguridad.

💬 **4. Automatización y ChatOps**

🧠 **Herramientas:**

- **Slack + Hubot**
- Notificaciones integradas desde GitHub Actions y Grafana.

🤖 **Flujos ChatOps:**

- @hubot despliegue prod → inicia pipeline de producción.
- @hubot alerta → responde con el último incidente registrado.
- Slack recibe:

- Fallos en tests.
- Alertas de CPU/memoria.
- Cambios en infraestructura.

🗺️ 5. Diagrama de Arquitectura

Incluye:

- Microservicios (barcos, carga, rutas).
- Gateway API.
- Seguridad con Cognito.
- Stack de monitoreo (Prometheus, ELK).
- Orquestador: Kubernetes.
- CI/CD: Jenkins y GitHub Actions.