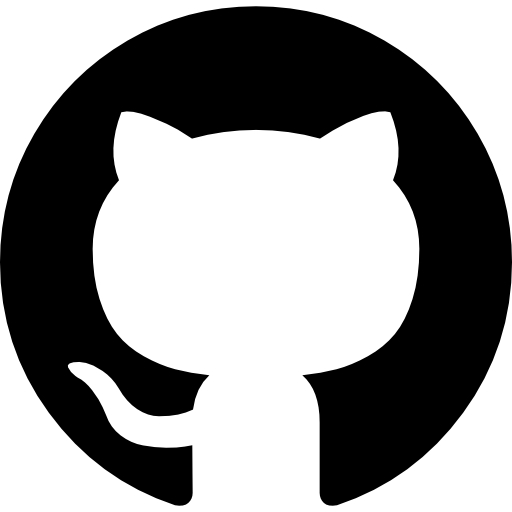
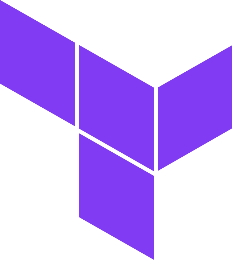
Este proyecto aborda la implementación de una infraestructura en la nube mediante AWS y Kubernetes para desplegar una intranet basada en WordPress. Se utilizarán herramientas de infraestructura como código con Terraform y un sistema de integración y despliegue continuo con ArgoCD.

**Nombre del tutor** JUAN IGNACIO PULIDO TRULLEN

Adrian Saenz de Magarola

KUBERNETIZACIÓN DE UNA INTRANET

# 1. Introducción

## 1.1. Descripción del problema

En muchas pequeñas y medianas empresas, la gestión de servicios internos como intranets, paneles administrativos o sistemas de comunicación se sigue realizando sobre servidores físicos o máquinas virtuales sin mecanismos modernos de automatización, monitorización o escalabilidad. Esto conlleva problemas como:  
- Elevados costes de mantenimiento.  
- Dificultades en el despliegue de actualizaciones.  
- Falta de resiliencia ante fallos.  
- Escalabilidad limitada y procesos manuales repetitivos.  
Este proyecto nace como una respuesta a esa problemática general: la necesidad de modernizar y automatizar la infraestructura que soporta estos servicios, aprovechando las ventajas de la computación en la nube, la contenerización y las herramientas de infraestructura como código.

## 1.2. Objetivo general del proyecto

El objetivo principal es diseñar e implementar una infraestructura automatizada y escalable en la nube para alojar servicios internos, utilizando tecnologías modernas como Kubernetes, Terraform, ArgoCD y herramientas de monitorización.

Como caso práctico, se desplegará una intranet basada en WordPress, que servirá como entorno de prueba para validar la infraestructura.

## 1.3. Enfoque y tecnologías

Para alcanzar estos objetivos, se usará AWS como proveedor cloud, instancias EC2 como nodos para un clúster Kubernetes, Terraform para el aprovisionamiento automático de la infraestructura, ArgoCD como sistema GitOps de despliegue continuo, y Prometheus y Grafana para la observabilidad.

# 2. Análisis del contexto y justificación

## 2.1. Contexto tecnológico actual

La industria IT está experimentando una transformación acelerada hacia modelos cloud-native. Según el informe anual de la CNCF (Cloud Native Computing Foundation), más del 80 % de las empresas ya usan contenedores en producción, y Kubernetes se ha consolidado como el orquestador estándar. Al mismo tiempo, prácticas como la automatización de despliegues (CI/CD), la infraestructura como código (IaC), y la monitorización continua han dejado de ser opcionales para convertirse en requisitos fundamentales.

Las empresas que no adaptan su infraestructura tecnológica corren el riesgo de quedar atrás en eficiencia, seguridad y competitividad.

## 2.2. Justificación del proyecto

Este proyecto busca simular un entorno real de migración de servicios internos hacia una infraestructura moderna. Se justifica por:  
- La necesidad de minimizar los errores manuales y los tiempos de despliegue.  
- El deseo de garantizar escalabilidad y disponibilidad ante cambios o picos de tráfico.  
- La importancia de contar con herramientas que faciliten la observación y el mantenimiento de la infraestructura.  
- La intención de formar una base sólida para futuras migraciones de servicios desde servidores físicos a la nube.

# 3. Estado del arte

El enfoque principal del proyecto se centra en la infraestructura y automatización. Por ello, el análisis comparativo se organiza en cinco áreas clave:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Categoría | Tecnologías Analizadas | Comparativa breve | Tecnología Elegida | Justificación |
| Contenerización | Docker, Podman, containerd | Docker es el estándar de facto, con amplia comunidad y documentación. | Docker | Mayor compatibilidad, soporte y madurez en producción. |
| Orquestación | Kubernetes, Docker Swarm, AWS ECS | Kubernetes ofrece mayor escalabilidad y soporte, aunque más complejo. | Kubernetes | Es el estándar actual en cloud-native. |
| Infraestructura como Código | Terraform, Pulumi, AWS CloudFormation | Terraform tiene una curva de aprendizaje moderada y es más portátil. | Terraform | Es más extendido y agnóstico de proveedor. |
| CI/CD & GitOps | ArgoCD, FluxCD, Jenkins | ArgoCD tiene mejor integración con Git y soporte para Kubernetes. | ArgoCD | Simplicidad, visualización clara y soporte activo. |
| Monitorización | Prometheus, Datadog, ELK | Prometheus es open source y se integra muy bien con Kubernetes. | Prometheus + Grafana | Ligero, gratuito y altamente integrable. |

Referencias:  
- CNCF Annual Report 2023  
- Docker Official Docs: https://docs.docker.com/  
- Kubernetes Documentation: https://kubernetes.io/docs/  
- Terraform Docs: https://developer.hashicorp.com/terraform  
- ArgoCD Docs: https://argo-cd.readthedocs.io  
- Prometheus Docs: https://prometheus.io/docs/

# 4. Requisitos del proyecto

## 4.1. Requisitos funcionales

- Despliegue de la infraestructura mediante Terraform.  
- Creación de un clúster Kubernetes con nodos EC2 en AWS.  
- Despliegue de WordPress contenerizado dentro del clúster.  
- Automatización del despliegue continuo con ArgoCD.  
- Monitorización de servicios con Prometheus y visualización en Grafana.  
- Autenticación de acceso a WordPress mediante OAuth con GitHub.  
- Control de versiones del código e infraestructura mediante GitHub.

## 4.2. Requisitos no funcionales

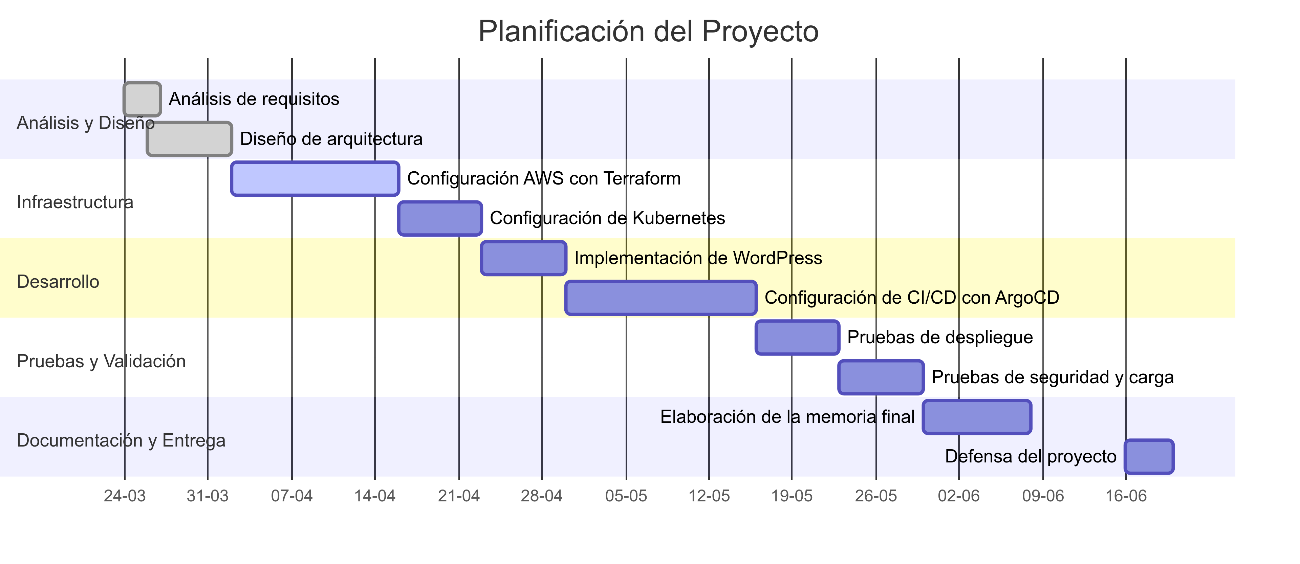
- Alta disponibilidad de los servicios (objetivo 99.9 %).  
- Escalabilidad horizontal de la aplicación.  
- Tiempos de despliegue inferiores a 5 minutos.  
- Tolerancia a fallos y capacidad de recuperación rápida.  
- Seguridad en el acceso a la infraestructura (grupos de seguridad, roles IAM).  
- Uso de soluciones dentro del free tier de AWS siempre que sea posible.

# 5. Planificación del proyecto

El proyecto se desarrollará entre marzo y junio de 2025. A continuación, se detalla la planificación de tareas por fases:

Fases:  
- Análisis de necesidades y diseño arquitectónico.  
- Configuración de infraestructura en AWS con Terraform.  
- Despliegue y gestión del clúster Kubernetes.  
- Automatización del despliegue de WordPress con ArgoCD.  
- Configuración de monitorización.  
- Validación, pruebas y documentación final.

## 5.1. Cronograma de trabajo



# 6. Desarrollo del proyecto

## 6.1. Análisis y diseño

La arquitectura propuesta sigue un enfoque cloud-native con un clúster Kubernetes sobre EC2, gestionado con Terraform. WordPress se despliega como contenedor y es gestionado por ArgoCD con flujos GitOps. Prometheus y Grafana permiten la observabilidad, y el estado se guarda en S3 con prevent\_destroy.

## 6.2. Tecnologías y herramientas empleadas

|  |  |
| --- | --- |
| Herramienta / Tecnología | Finalidad |
| Terraform | Infraestructura como código (AWS) |
| AWS EC2 + S3 | Clúster y backend de tfstate |
| Kubernetes | Orquestación de contenedores |
| Docker | Contenerización de WordPress |
| ArgoCD | GitOps y despliegue continuo |
| GitHub | Repositorio de código |
| Prometheus + Grafana | Monitorización y visualización de métricas |

## 6.3. Contrato y pliego de condiciones

|  |  |  |
| --- | --- | --- |
| Recurso | Descripción | Coste mensual aproximado |
| AWS EC2 (3x t3.medium) | Nodos para clúster K8s | $90 |
| S3 + versionado | Backend de tfstate | $0.01 |
| Prometheus + Grafana | Open source | $0 |
| DockerHub + GitHub | Plan gratuito | $0 |
| Total estimado | Durante 3 meses | ~$270 |

## 6.4. Análisis de riesgos

|  |  |  |
| --- | --- | --- |
| Riesgo | Impacto | Mitigación |
| Sobrecoste en AWS | Medio | Uso del Free Tier siempre que sea posible. |
| Fallos de configuración en Kubernetes | Alto | Validación de manifiestos y despliegues progresivos. |
| Pérdida del estado de Terraform | Crítico | S3 con versionado y prevent\_destroy. |
| Fallos en CI/CD | Medio | Uso de logs y pruebas continuas con ArgoCD. |