
Facultad de ciencias – Universidad Nacional de Ingeniería

# Desarrollo de software

# Actividad 2

Aarón Flores Alberca
Leonardo Alexander Chacón Roque

# TAREA TEÓRICA 1

Investigar una herramienta de IaC (p. ej. Terraform) y describir cómo organiza sus módulos.

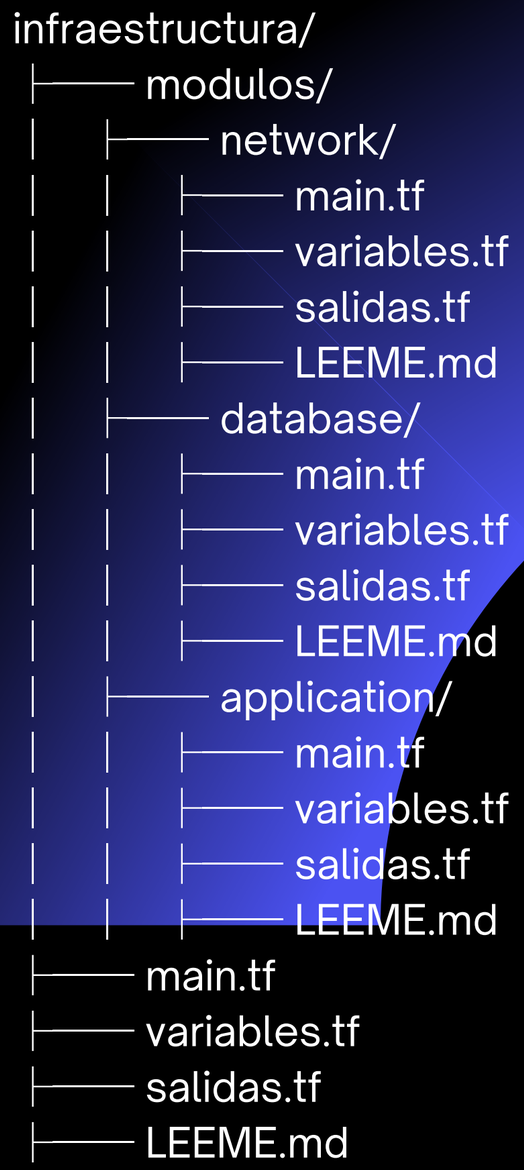
- Terraform organiza sus módulos de la siguiente forma: módulo raíz o carpeta principal donde guardamos los archivos de la configuración (main.tf, variables.tf, outputs.tf); módulos reutilizables donde están los subdirectorios dentro del módulo raíz o se almacenan en repositorios externos.

# TAREA TEÓRICA 1

Proponer la estructura de archivos y directorios para un proyecto hipotético que incluya tres módulos: network, database y application. Justificar la jerarquía elegida.

Razones de la estructura:

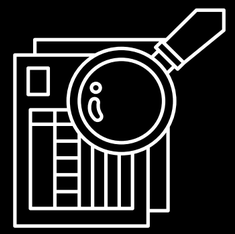
- Cada módulo es reutilizable.
- Cada módulo tiene una definición independiente de variables y salidas.



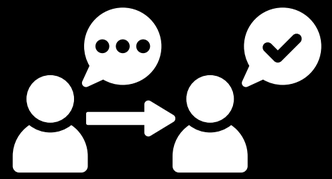
# TAREA TEÓRICA 2

Describir un flujo simple de despliegue donde un desarrollador hace un cambio en el código, se construye una nueva imagen Docker y se actualiza un Deployment de Kubernetes.

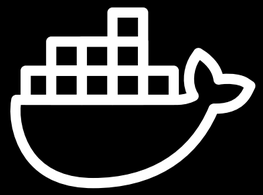
Un desarrollador modifica o mantiene una sección del código fuente de un proyecto a gran escala



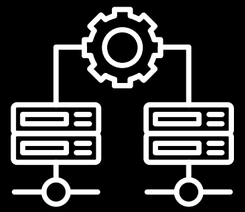
Después de la modificación, crea una rama al repositorio principal en la que se ejecutan las pruebas unitarias para verificar su funcionamiento.



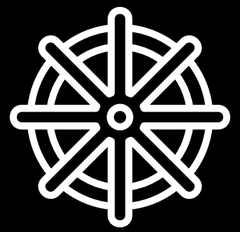
Una vez haya pasado satisfactoriamente las pruebas, se solicita un pull request para ser mergeado a la rama principal.
Usualmente validado por un supervisor senior



Cuando dicho pull request es aceptado y validado, el sistema dispara un trigger por dicho evento y crea una imagen con identificadores únicos de una nueva versión



Nuevamente se expone a dicha imagen a rigurosos escaneos de seguridad ante vulnerabilidades y que cumpla estándares preestablecidos



Finalmente una vez habiendo aprobado dichas últimas pruebas es transferido a su fase de despliegue en donde es orquestrado gracias a Kubernetes el cual crea diferentes pods con dichas imagenes y actualiza el sistema con alguna estrategia de despliegue.

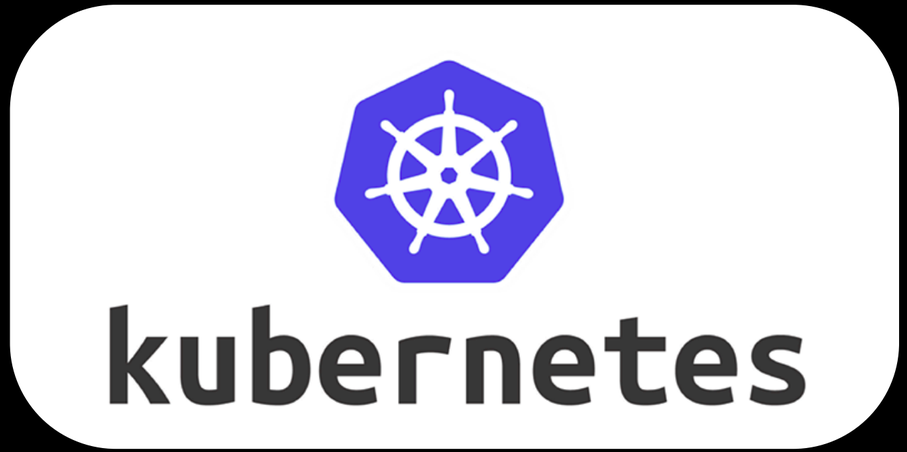
# Ventajas del uso de Kubernetes en eventos de alto tráfico

TAREA TEÓRICA 2

Kubernetes es una plataforma portable de código abierto utilizado para la orquestación de contenedores que se encarga de automatizar y organizar el despliegue, mantenimiento y la escalabilidad de aplicaciones contenerizadas, gracias a un balanceador de cargas y una abstracción del manejo de múltiples contenedores.

Este cambio de paradigma en el mundo del desarrollo de software lanzado en 2014 representó grandes ventajas en cuanto al manejo de recursos, entre ellas tenemos las siguientes:

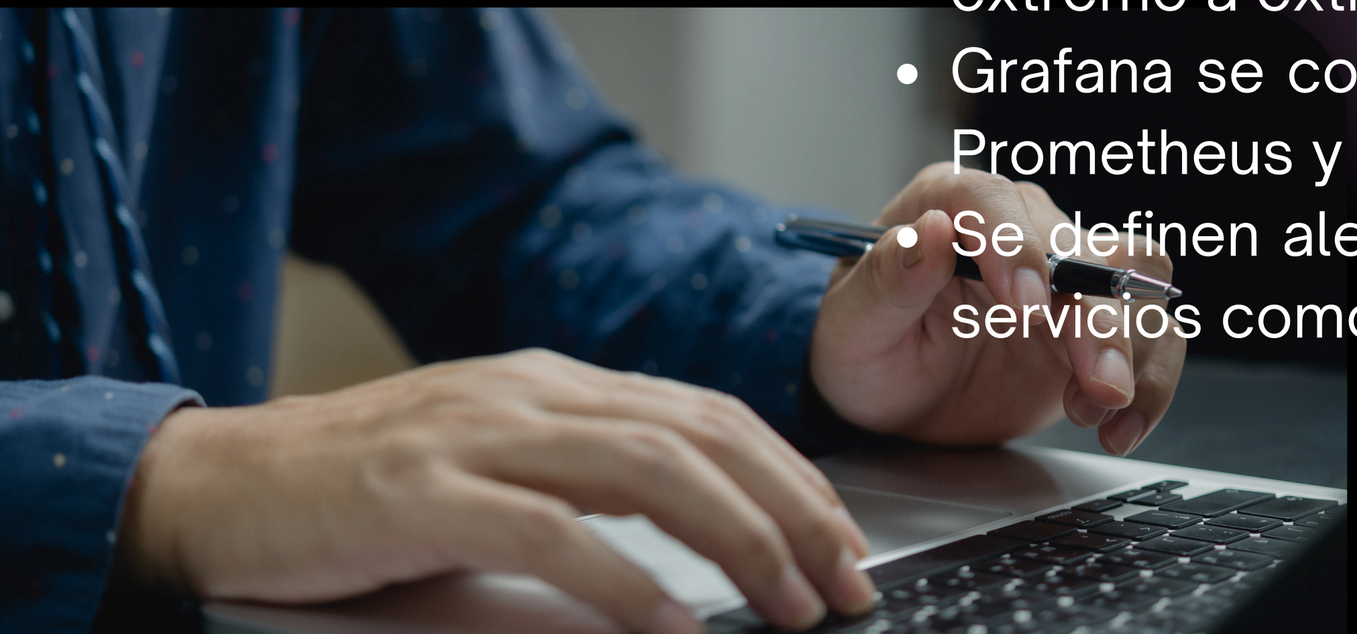
- Redistribución del flujo de trabajo en entornos de trabajos multinube
- Propiedades de autorecuperación que permiten detectar contenedores o pods con malfuncionamientos para intentar reactivarlos.
- Escalado automático, permitiendo otorgar recursos de manera dinámica a pods que los requieran.



# TAREA TEÓRICA 3

Investigar y describir cómo Prometheus y Grafana se integran con Kubernetes para monitorear los contenedores y el cluster.

- Prometheus se despliega como un servicio en el cluster y recolecta métricas de los pods de extremo a extremo.
- Grafana se configura para obtener las métricas de Prometheus y crear dashboards visuales.
- Se definen alertas en Prometheus que se envían a servicios como Slack o PagerDuty.



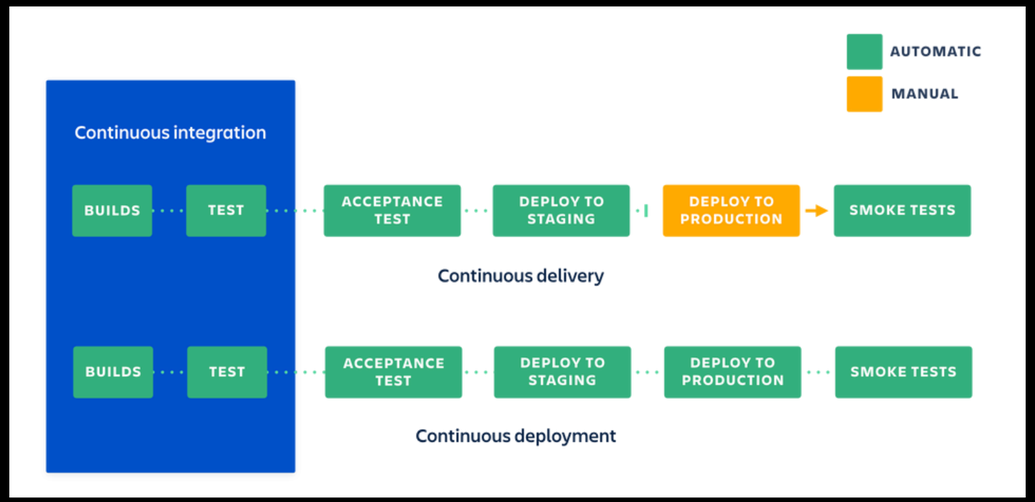
# TAREA TEÓRICA 3

Proponer un set de métricas y alertas mínimas para una aplicación web (por ejemplo, latencia de peticiones, uso de CPU/memoria, tasa de errores).

- Alertar si supera 500ms de latencia.
- Alertar si supera el 80% de la memoria asignada.
- Alertar si supera el 5% de las tasas de errores en un periodo.

# TAREA TEÓRICA 4

## Explicar la diferencia entre entrega continua (continuous delivery) y despliegue continuo (continuous deployment).




Dentro del contexto DevOps, tanto la entrega continua (continuous delivery) y el despliegue continuo (continuous deployment) son conceptos clave que ocupan la etiqueta de CD dentro del pipeline CI/CD de alguna empresa u organización dependiendo de los requerimientos de su negocio.

Ambos son considerados prácticas de despliege de código hacia un entorno de testing o producción luego de ser expuestos a rigurosas pruebas de seguridad.

La principal diferencia entre ambas es que en el despliegue continuo, los cambios realizados son trasladados automáticamente a producción después de aprobar las pruebas de seguridad, mientras que en la entrega continua una vez terminadas las revisiones se acorda manualmente una fecha de lanzamiento a producción mediante alguna estrategia.

Se puede ver al despliegue continuo como un caso especial de entrega continua en el que todo ya se encuentra automatizado; sin embargo, es por eso mismo que el segundo resulta ser mucho más complejo de mantener que el primero.

Además, los casos de uso de estas prácticas dependerán de los requerimientos de cada negocio, como ya se habia mencionado

# TAREA TEÓRICA 4

## Describir la relevancia de implementar pruebas automáticas (unitarias, de integración, de seguridad) dentro del pipeline.

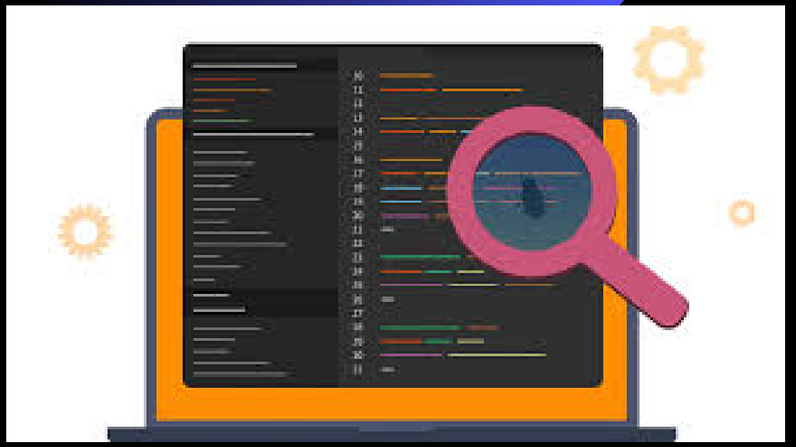
La adición de pruebas dentro del ciclo de vida del desarrollo del software se ha vuelto un estándar en la industria con el pasar de los años y con la mejoría de las herramientas necesarias para este.

Entre los muchísimos beneficios que posee, resaltamos:

- Detección de errores en fases tempranas
- Mejoría notoria en la calidad de código
- Puede servir de documentación para entender lo que realiza cada módulo

Así mismo, también es de vital importancia resaltar la automatización de dichas pruebas, con las que haciendo uso de soluciones como GitHub Actions, Jenkins o GitLab CI se puede reducir los problemas de implementarlas manualmente mediante un solo script; además, proporciona reproducibilidad lo que permite que diferentes miembros del equipo puedan ejecutarlas en sus entornos de trabajo.

Estas facilidades junto con las ventajas proporcionadas por las pruebas garantizan acelerar el ciclo de desarrollo de software a la vez que generan productos de muchísima más alta calidad.



# EVALUACIÓN DE LA TEORÍA

Importancia de:
**IaC**: Automatización y consistencia, escalabilidad, erazabilidad y versionado.
**Contenedores**: Portabilidad, ejecución ligera, escalabilidad flexible.
**Kubernetes**: Automatización de despligue, balanceo de carga y autorecuperación.
**Observabilidad**: Detección temprana de fallas y optimización del rendimiento.
**CI/CD**: Entrega rápida y frecuente de nuevas funcionabilidades, reducción de errores en producción y mayor colaboración y eficiencia.

# EVALUACIÓN DE LA TEORÍA

Riesgos y Desafíos:

- Sobrecarga cognitiva
- Necesidad de capacitación
- Configuración de seguridad
- Complejidad de la observabilidad

# Gracias

-