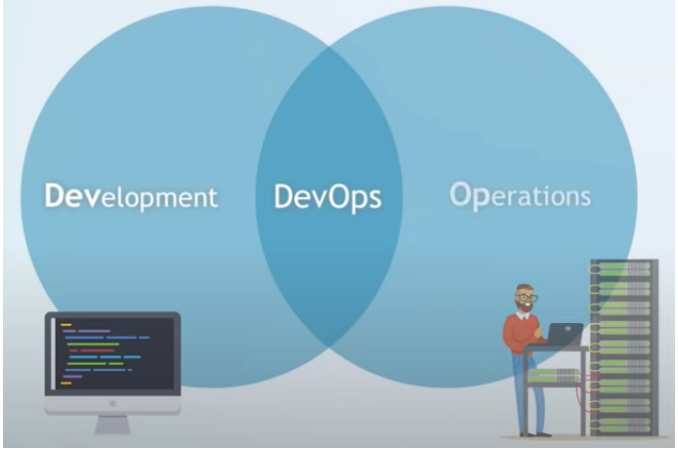
Examen 4. Martha Paola Peña Sotelo

Qué es DevOps y en qué consiste, etapas de devops

Dev Ops = Es una intersección entre desarrollo y operación, que permite el automatismo y gestión eficiente de los recursos en IT. En otros términos, implica la colaboración entre desarrolladores, quienes desarrollan el código y operadores de sistemas, quienes construyen la infraestructura en la cual el código va a correr



DevOps es un conjunto de cultura, procesos y herramientas que permiten entregar actualizaciones de software a alta velocidad, garantizando la calidad del código, la estabilidad de la aplicación y evitando el aislamiento de los equipos.

Objetivos de DevOps

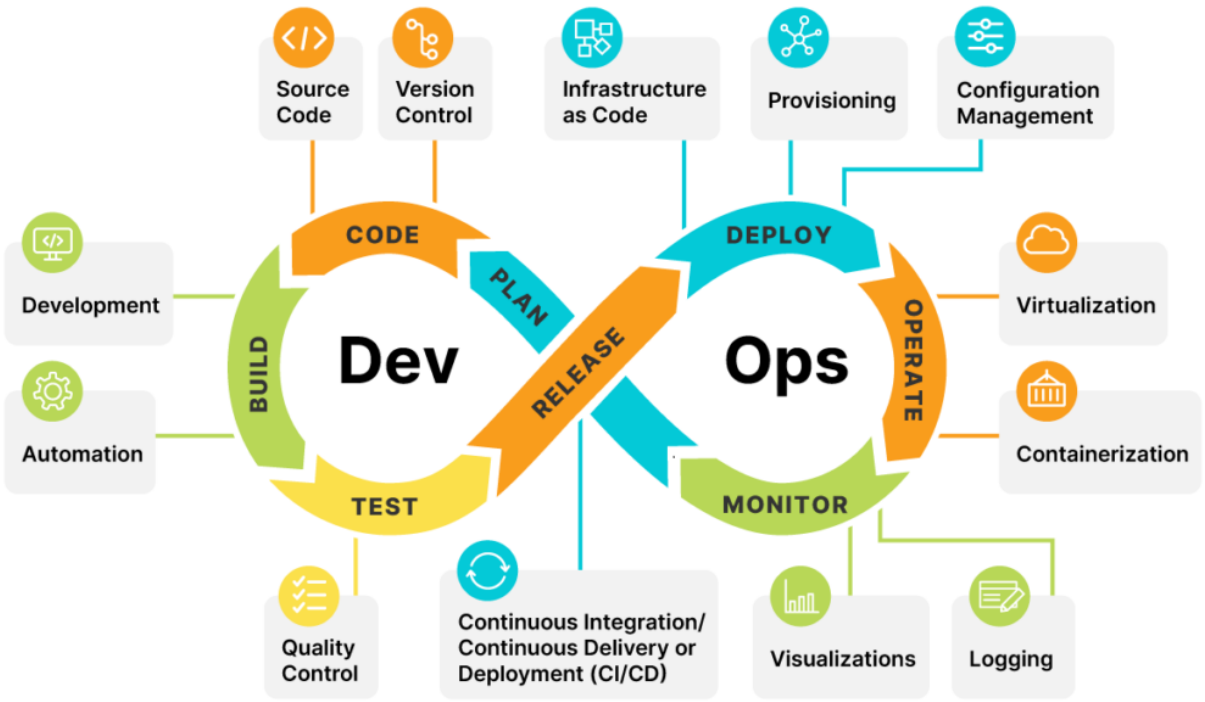
- Mejorar la frecuencia y la calidad del despliegue ⇒ **Entrega y despliegue continuo** de actualizaciones.
- Mejorar el *time to market* o tiempo de salida al mercado
- Permite capturar y corregir rápidamente los errores

- Realizar correcciones en corto tiempo: esto garantiza la estabilidad de la aplicación, se puede hacer monitoreo de la aplicación para conocer los errores y corregirlos con la mayor velocidad.
- Evitar el aislamiento de los equipos, implementando herramientas que permitan que todo el ciclo de desarrollo de sorfware en lugar de ser dos etapas separadas sean una sola donde ambos equipos colaboren.
- Garantizar la calidad del código: con las herramientas adecuadas se pueden hacer test que prueben el código para evitar que entre en conflicto con el de otro desarrollador

> 💡 Nota: no existe como tal un rol de DevOps.

El profesional que se encarga de diseñar los procesos e implementar herramientas es el Site Reliability Engineer(SRE), el ingeniero de confiabilidad del sitio.

Ciclo de vida DevOps



El ciclo DevOps es un ciclo que no se detiene cuya finalidad es garantizar que todos los objetivos se cumplan. Está compuesto de las siguientes etapas:

▼ 1. Plan

Se analizan los requerimientos del proyecto o de la funcionalidad que se está requiriendo, se planifica cómo desarrollarla, según la metodología que se esté usando.

> 💡 Herramientas de planeación como **Jira, Asana, Trello o Notion**

▼ 2. Code

Los programadores tienen asignadas sus tareas o sus historias de usuario y empiezan a escribir el código según lo que se les haya asignado

> 💡 Herramientas de programación: **Repositorios(GitHub, GitLab)**

▼ 3. Build

Todos los programadores juntan su código, este código se integra en master, luego se hace una compilación que se entrega a un paquete ejecutable después de pasar los test de integración.

> 💡 Herramientas de compilación: **Apache Maven, Gradle.**

▼ 4. Test

Se prueba el software para garantizar que funcione según los requerimientos que se pidieron en el paso inicial

> 💡 Herramientas de testing: **Selenum, Gremlin, JUnit,**

▼ 5. Release

Luego de pasar los test de la aplicacion se crea una imagen, artefacto o ejectuable, dependiendo de la aplicación para pasar a producción

> 💡 Herramientas: **Jenkins, CircleCi**

▼ 6. Deploy

Enviar el paquete o ejecutable a los servidores de producción para que lo usen los usuarios finales.

> 💡 Herramientas: **GitHub Actions, GitLab, Docker**

▼ 7. Operaciones

En realidad esto está ocurriendo todo el tiempo, aquí se encuentran las tareas de configuración de los sistemas, de optimización , implementación de infraestructura, servidores, replicas, caché.

> 💡 Herramientas: **Chef, Ansible, Kubernates, Docker**

▼ 8. Monitor

Etapa de monitoreo. Se utilizan herramientas que están mirando todo el tiempo la aplicación para saber si ésta se encuentra en peligro de caerse por alguna falla, o si falló saber exactamente qué la hizo fallar y corregir lo más pronto posible el error.

> 💡 Herramientas de monitoreo: **New Relic, Amazon cloudwatch, Grafana, Prometeus**

Lo ideal es pasar de manera automática por las etapas del ciclo, gracias a técnicas como CI/CD:

La **integración continua** consiste en crear pruebas automatizadas, de tal forma que cada vez que un programador envíe su código, esas pruebas corran de manera

automática, revisen si el código cuenta con ciertos estándares y políticas que se han definido previamente, y si no entra en conflicto con otro código que ya existe dicho código pasa a la siguiente etapa.

Continuous delivery, se toma el código que ya pasó los test de integración para crear un ejectuable listo para producción.

Continuous deployment, es hacer el deploy.