# Cuestionario DevOps y CI/CD

## 1. ¿Qué filosofía o metodología de desarrollo de software se adapta al modo de trabajo de CI/CD?

La metodología que mejor se adapta al modo de trabajo de CI/CD es DevOps, ya que integra el desarrollo (Dev) y las operaciones (Ops) para automatizar el proceso de construcción, prueba y despliegue de software de forma continua. También se apoya en metodologías ágiles como Scrum o Kanban, que promueven entregas rápidas y mejoras constantes.

## 2. ¿Para qué sirven las siguientes tecnologías usadas en DevOps?

* Jira: herramienta de gestión de proyectos y seguimiento de incidencias.
* Git: sistema de control de versiones que permite trabajar colaborativamente en el código y mantener su historial.
* Docker: plataforma de contenedores que permite ejecutar aplicaciones de forma aislada y portátil.
* AWS (Amazon Web Services): servicio de nube que ofrece infraestructura bajo demanda.
* Chef: herramienta de automatización para configurar servidores mediante scripts declarativos.
* Ansible: herramienta de automatización y gestión de configuraciones basada en archivos YAML.
* Kubernetes: sistema de orquestación que gestiona, escala y despliega contenedores.
* Splunk: herramienta para analizar y visualizar grandes volúmenes de datos y registros del sistema.
* Nagios: herramienta de monitoreo que controla el estado de servidores, servicios y redes.
* Jenkins: servidor de automatización que permite implementar integración continua (CI) y despliegue continuo (CD).
* JUnit: framework para realizar pruebas unitarias en Java.
* Selenium (SE): herramienta para automatizar pruebas funcionales en aplicaciones web.
* Maven: herramienta de gestión de proyectos Java que automatiza compilación, empaquetado y dependencias.

## 3. ¿En qué fase de DevOps se encuadra cada tecnología?

|  |  |
| --- | --- |
| Fase DevOps | Tecnologías |
| Planificación | Jira |
| Código | Git, Maven |
| Construcción (Build) | Maven, Jenkins, Docker |
| Pruebas (Test) | JUnit, Selenium, Jenkins |
| Despliegue (Deploy) | Jenkins, Docker, Kubernetes, AWS |
| Configuración | Chef, Ansible |
| Monitoreo | Nagios, Splunk |

## 4. ¿Qué son los hooks de Git, para qué sirven y cuándo se ejecutan?

Los hooks de Git son scripts que se ejecutan automáticamente cuando ocurren ciertos eventos en el repositorio. Permiten automatizar tareas como verificar el formato del código, correr tests o validar configuraciones antes de subir cambios.

* Ejemplos de hooks comunes:
* pre-commit: se ejecuta antes de realizar el commit.
* commit-msg: se ejecuta después de escribir el mensaje de commit.
* post-commit: se ejecuta después del commit.

## 5. Hook que configura el nombre de usuario si no está definido antes del commit

Ejemplo de script (.git/hooks/pre-commit):  
  
#!/bin/bash  
USER\_NAME=$(git config user.name)  
if [ -z "$USER\_NAME" ]; then  
 echo 'No hay un nombre de usuario configurado en Git.'  
 read -p 'Por favor, introduce tu nombre de usuario: ' NEW\_NAME  
 git config --global user.name "$NEW\_NAME"  
 echo 'Nombre de usuario configurado como: $NEW\_NAME'  
else  
 echo 'Nombre de usuario detectado: $USER\_NAME'  
fi

## 6. ¿Qué diferencia hay entre Hooks y GitHub Actions?

Los hooks se ejecutan localmente dentro del repositorio, antes o después de acciones como commit o push. Sirven para automatizar tareas personales o de equipo localmente.  
  
En cambio, las GitHub Actions se ejecutan en los servidores de GitHub tras eventos remotos como push o pull request, y se usan para automatizar procesos de CI/CD en la nube (compilar, probar y desplegar).