UCLM UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior Ingeniería Informática

DSI DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

XUNTA DE GALICIA | CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE

CFR FERROL

Tema 3. Implantación de sistemas seguros de despliegue de software

- DevOps

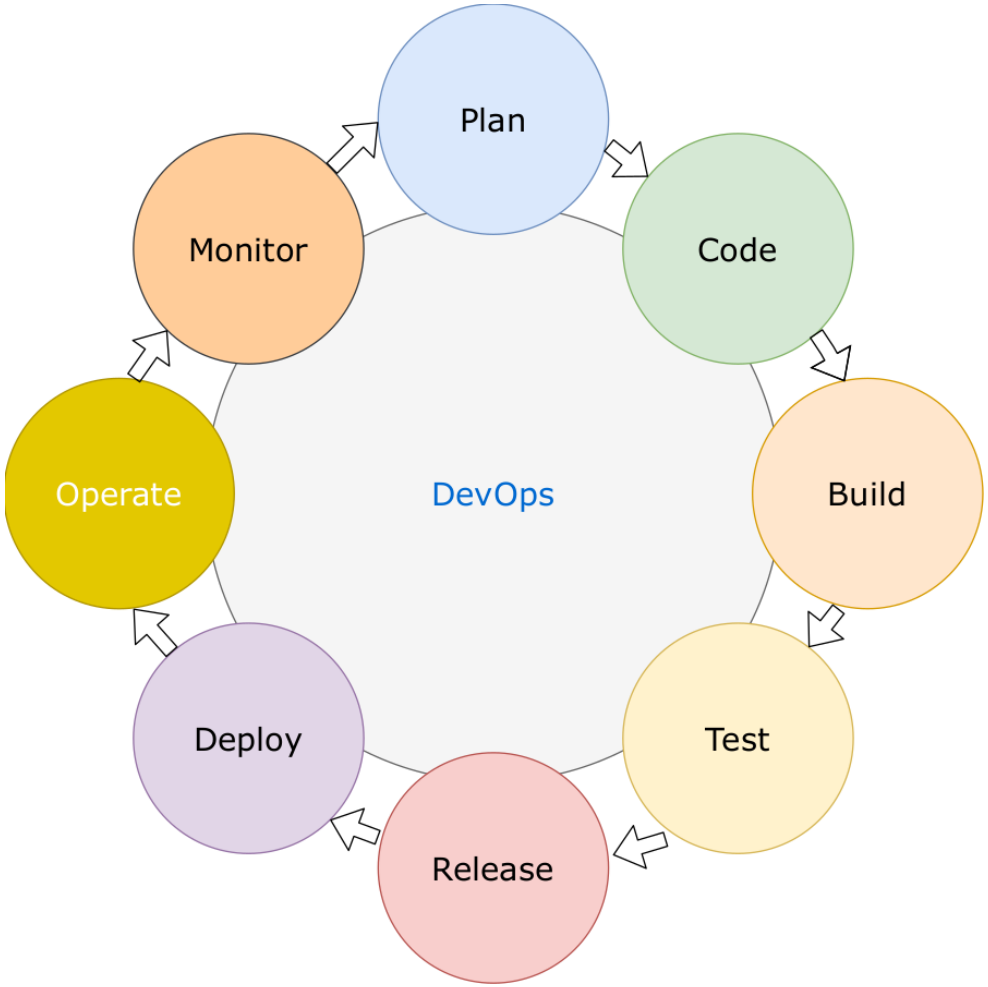
UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro

GOBIERNO DE ESPAÑA | MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

DevOps

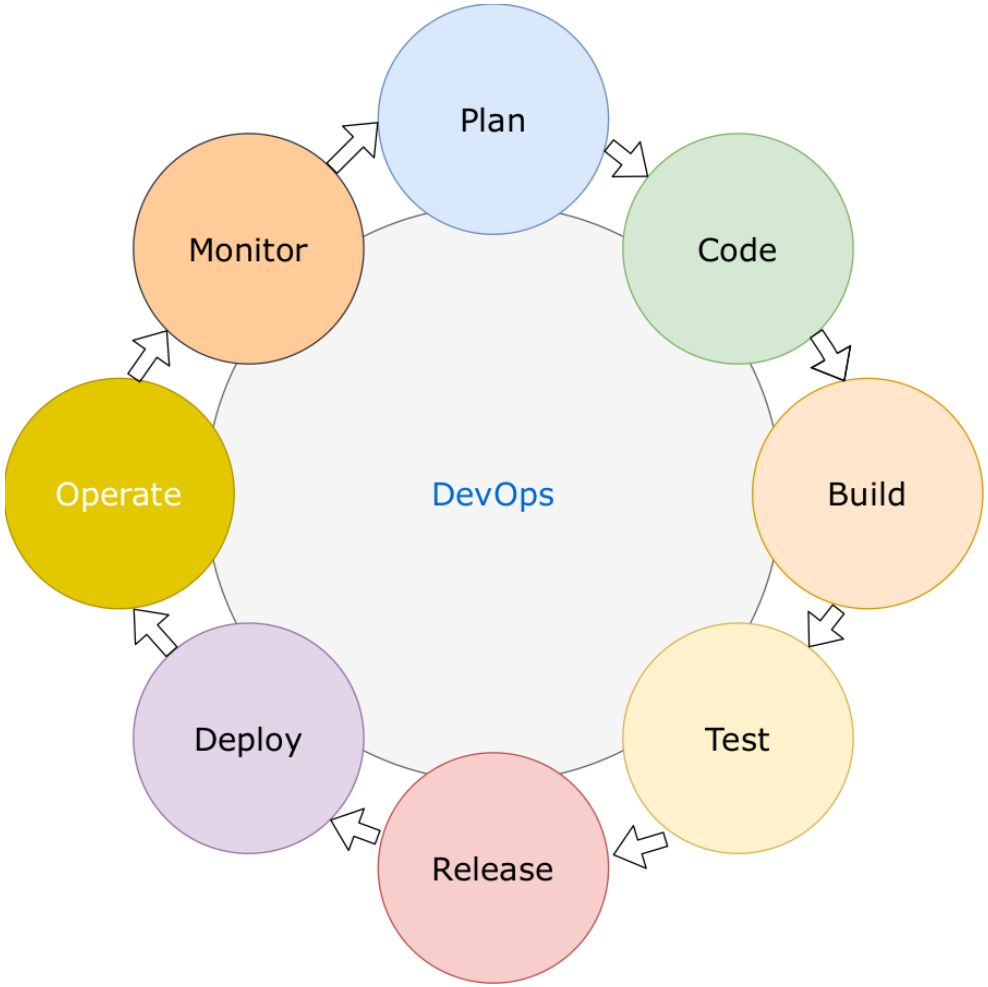
- Es un cambio de filosofía en la forma de trabajar entre los equipos de desarrollo y operaciones.
- La idea subyacente es la comunicación, colaboración e integración entre equipos de desarrollo y equipos de operaciones TI.
- El objetivo principal es reducir costes y tiempo mejorando la calidad del software desarrollado (agilizando, automatizando y monitorizando).
- Proceso asociado a metodologías de **integración continua** o **CI** (compilación y pruebas), **entrega continua** o **CD** (liberación de versiones) y **despliegue continuo** (en entornos reales/producción).
  - Procesos automáticos, repetibles y llevados a cabo lo más a menudo posible.

DevOps
Ciclo de vida (Dev)



- **Plan**: definición de requisitos del producto, así como la definición de roles y asignación de tareas a los integrantes del equipo.
- **Code**: fase de desarrollo e implementación del código del producto.
- **Build**: construcción de software (compilación) de forma automatizada y favoreciendo la creación de entornos replicables.
- **Test**: realización de pruebas de software automatizadas con el objetivo de encontrar y corregir errores.

DevOps
Ciclo de vida (Ops)



- **Release**: en esta fase se libera una nueva versión del producto (p.e. con una funcionalidad nueva o con errores solventados).
- **Deploy**: proceso automatizado en el cual se despliega la nueva versión en un entorno de producción. Este proceso no debe afectar al funcionamiento de la versión anterior ya desplegada.
- **Operate**: fase de configuración y corrección de errores que se encuentran en el entorno de producción al actualizar.
- **Monitor**: adopción de las medidas y métricas necesarias para controlar la salud, el comportamiento y el rendimiento del producto.

DevOps
SecDevOps

- Dentro del ciclo de vida de DevOps, el equipo de seguridad empieza a trabajar en las últimas fases.
  - Deploy, Operate y Monitor.

- Siguiendo la filosofía **SecDevOps**, la seguridad se aplica desde el inicio del desarrollo y es un proceso constante de manera que permite encontrar y subsanar problemas de seguridad desde el inicio y en todas las fases.

- Comunicación, colaboración e integración del equipo de seguridad, de desarrolladores y de TI desde el inicio del proyecto.

DevOps
Integración continua (CI)

- La **integración continua o CI** tiene como objetivo automatizar la integración de código de los desarrolladores.
  - Build, Test (local) y parte de Release.

- Facilita a los desarrolladores la unión (merge) de su código → permite analizar el producto global y encontrar posibles problemas de manera más sencilla.
  - Los desarrolladores envían sus cambios de forma periódica a un repositorio central que ejecuta un sistema de control de versiones (Git).

- En CI se **automatizan** las diferentes fases para validar que el código añadido no daña ningún componente principal del producto o aplicación.

DevOps
Integración continua (CI)

- Herramientas CI:
  - **Plan**: Jira, Trello, GitKraken Boards, etc.
  - **Code**: IDEs, Git, GitHub, Bitbucket, etc.
  - **Build**: Make, Maven, Gradle, Docker, etc.
  - **Test**: UnitTest (Python), Selenium (web), PHPUnit, etc.

  - **CI**: Jenkins, GitHub Actions, Travis, etc.
  - **DevSecOps**: OWASP ASVS, MobSF, git-secret, etc.

DevOps

Entrega continua (CD)

- Amplía la integración continua generando un paquete después de haber pasado las pruebas software funcionales.
- Su principal objetivo es tener un artefacto software preparado que pueda ser desplegado en un entorno de producción en cualquier momento.
- En CD también se **automatizan** todos los pasos necesarios para generar un paquete que pueda ser desplegado.
- Herramientas CD:
  - **Release**: Docker Hub, Nexus, Archiva, etc.

DevOps
Despliegue continuo

- Metodología/estrategia encargada de llevar a producción el resultado de las fases anteriores.

- **Automatización** de todos los pasos necesarios para poner a punto la aplicación en un entorno de producción.
  - La idea es que cualquier cambio de los desarrolladores pueda estar en producción de manera automática.

- Herramientas:
  - **Deploy**: Ansible, Chef, Puppet, etc.
  - **SecDevOps**: Nikto, NMAP, sqlmap, OWASP ZAP, etc.
  - **Simulación de fallos**: LoadRunner, JMeter, Blazemeter (apps móviles), etc.
  - **Monitorización**: Splunk, Nagios, Kibana, etc.

Control de versiones
Git

- **Git** es un sistema de control de versiones (CVS) open source.

- Su objetivo principal es el almacenamiento y administración de las diferentes versiones de los archivos que componen un producto software.

- Características:
  - Rápida recuperación de cualquiera de las versiones generadas con anterioridad → Actúan como copias de seguridad.
  - Comparación visual (diff) entre las diferentes versiones de un archivo → ¿Qué ha cambiado en la última versión? ¿Por qué?
  - Posibilidad de mantener diferentes versiones a la vez (branch) o dividir un proyecto en varias partes → Backend y frontend.

- Al igual que las imágenes de Docker en los registros, los proyectos en Git se almacenan en repositorios.

Control de versiones
GitHub

- Por el contrario, **GitHub** es una plataforma web online donde los usuarios pueden cargar y gestionar sus repositorios Git de forma remota.

- Facilita la compartición de código y la colaboración entre los integrantes de un equipo de desarrollo software.
  - Provee de repositorios públicos y privados.

- Fomenta la participación de los usuarios en proyectos open source.

GitHub

Control de versiones

Git y GitHub (PoC)

- **Instalar Git:** http://git-scm.com/downloads

- **Comprobar la instalación y obtener la versión instalada:**
  - *git version*

- **Crear un nuevo repositorio local:**
  - *git init*

- **Ver el estado del repositorio:**
  - *git status*

- **Ver la rama actual donde nos encontramos y renombrar la rama:**
  - *git branch*
  - *git branch -M main*

Control de versiones

Git y GitHub (PoC)

- **Agregar los archivos sin seguimiento al “commit”:**
  - *git add .*
- **Registrar los cambios (commit) en el repositorio:**
  - *git commit -m “1.0.0: Taller Kubernetes PPS”*
- **Nos registramos en GitHub y creamos un nuevo repositorio.**
- **Asignar el repositorio remoto de GitHub al repositorio Git local:**
  - *git remote add origin URL_REPO.git*
- **Subir commit/s con los últimos cambios a nuestro repositorio en GitHub:**
  - *git push -u origin main*

Control de versiones

Git y GitHub (PoC)

- **Crear y cambiar de rama localmente:**
  - *git branch mi-rama*
  - *git checkout mi-rama*
- **Subir la rama nueva con los cambios:**
  - *git push --set-upstream origin mi-rama*
- **Descargar un repositorio remoto:**
  - *git clone URL_REPO.git*
- **Obtener los últimos cambios de un repositorio desde GitHub:**
  - *git pull*

Automatización con Jenkins

PoC

- **Jenkins** es un software de automatización open source que implementa procesos de integración continua CI.
- Admite la integración de herramientas CVS como Git y de construcción de software como Gradle o Ant.
- Cuenta con imagen oficial en Docker Hub.

Jenkins

UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior Ingeniería Informática

DSI
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

XUNTA DE GALICIA
CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE

CFR FERROL

Puesta en producción segura

- Tema 3. Implantación de sistemas seguros de despliegue de software

UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL