THREEPOINTS
THE SCHOOL FOR DIGITAL BUSINESS

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

# Master en FullStack Developer

## CI/CD y DevOps
## Practica 1 - Construcción de un Pipeline

MIA 2011

## Objetivo

El objetivo de esta práctica es familiarizarse con el entorno de Jenkins, y construir un primer pipeline gráfico para entender lo que son los pipelines de CI y CD, y ver las capacidades de Jenkins.

Para esta práctica, se ha montado una máquina virtual para el curso, la cual dispone de los siguientes servicios:

- Gitlab: http://localhost
  - Usuario: threepoints
  - Password: threepoints
- Jenkins: http://localhost:8080
  - Usuario:threepoints
  - Password: threepoints
- SonarQube: http://localhost:8090
  - Usuario: admin
  - Password: threepoints
  - (Está apagada por defecto, hay que seguir las instrucciones del tema 3 para desplegarlo).

Para entrar a esta máquina virtual, se usarán las credenciales:
- Usuario: threepoints
- Password: threepoints

## 1. Configuración de los Plugins (2 puntos)

En esta primera pregunta, hace falta encontrar y mostrar capturas de pantalla de los siguientes elementos:
1. Configuración de las credenciales de Gitlab.
2. Configuración del plugin de SonarQube.
3. Configuración del plugin de Gitlab.

## 2. Creación de un Pipeline básico de CI (6 puntos).

Para esta primera práctica, se ha subido al gitlab un proyecto, que consiste en una aplicación web dockerizada: http://localhost/threepoints/docker-vulnerable-dvwa/

Para esta asignatura, habrá que crear un nuevo Job, que consista en un pipeline básico que realice ciertas comprobaciones antes de construir y lanzar el Docker con el proyecto que tenemos.

El pipeline debe de tener las siguientes fases:

1. **Descarga de código**. Debe de descargarse el código del gitlab para su integración.
2. **Ejecución de pruebas de calidad de código**. Para esto, se usará el plugin de SonarQube en la instancia que tenemos creada.
3. **Construcción del container de Docker**. Para ejecutar esto, hace falta ejecutar el comando '**`docker build -t dvwa .`**'

El Job tiene que llamarse "**practica_1_2022_*NOMBRE_ALUMNO***" y se debe de entregar:

- Captura de pantalla del Job ejecutado exitoso.
- Capturas de cada paso de ejecución comentado.
- Captura de pantalla con los resultados de SonarQube.

## 3. Hacer que el Pipeline se ejecute mediante Push a Git (2 puntos).

El siguiente punto consiste en hacer que el pipeline reciba peticiones de Gitlab cada vez que se hace un push a Git. Esto es fácil de realizar mediante la funcioanldiad de Webhooks de Gitlab, que permite cada vez que se haga una acción en un repositorio, poder enviar una petición HTTP PUSH a cualquier servicio que se defina.

Jenkins, a su vez, gracias al plugin de integración con Gitlab, permite crear un servicio que reciba ese Push, y ejecute la tarea. A la hora de activar este plugin, nos dará la opción de generar una clave secreta, que habrá que pasarle a Gitlab a la hora de establecer el webhook, para permitir identificar unívocamente esta llamada.

Durante este ejercicio, hay que adaptar el Job creado en el ejercicio 2 para que se ejecute cada vez que hay un push en el repositorio. Como entregable de este ejercicio, será necesario:

- Captura de pantalla de la configuración del proyecto de Gitlab.
- Captura de pantalla de la configuración del Job de Jenkins.

## 4. Opcional: Circuito con CD (+2 puntos adicionales).

Al Job mencionado en el ejercicio anterior, haría falta hacer una nueva tarea que:

1. Pare la ejecución del Docker, si estaba siendo ejecutado.
2. Lo ejecute (Docker run) mapeando el puerto 80 del Docker con el 81 de la máquina.

Para este ejercicio, hará falta entregar:

- Captura de pantalla con los pasos de job de Jenkins necesarios.
- Captura de pantalla con la ejecución exitosa.

- Captura de pantalla de lo que se ve al meterse en http://localhost:81 tras lanzar el job de despliegue.

Este ejercicio es opcional, y se puede lograr la máxima nota de la práctica sin él.