Práctica 3 Jenkins + DockerHub

Esta práctica consiste en una toma de contacto con el sistema de integración contínua Jenkins.

Partiremos de la imagen oficial de jenkins pero con una serie de modificaciones para poder lanzar compilaciones de proyectos c++ y de python. El dockerfile para generar la imagen está disponible en el aula virtual.

Para simplificar el despliegue de nuestra imagen de Jenkins modificada haremos uso de github y docker hub. Crearemos un repositorio en github donde subiremos el Dockerfile de nuestra imagen y lo enlazaremos con dockerhub para que genere la imagen por nosotros.

Crearemos una clase de python que realizará una serie de operaciones entre dos números enteros:

- Suma
- Resta
- Multiplicación
- División
- Raíz cuadrada

Con una serie de test para cada uno de ellos.

Una vez tenemos la imagen de docker generada y con el código de python creamos un contenedor con los siguientes criterios mínimos:

- Mapearemos el puerto local 8000 con el 8080 del contenedor
- Mapearemos la carpeta donde tengamos el código de python a la carpeta /opt/my_data del contenedor
- Mapear el contenido de la práctica 1 a /opt/practica1
- Habilitar el soporte para poder ejecutar docker dentro del contenedor

Una vez arrancado jenkins, debemos instalar un plugin necesario para crear reportes con los resultados de test:

- Xunit

En cuanto a tareas a realizar se piden crear tres jobs:

1. Job c++
   Este job deberá descargar el código del siguiente repositorio:
   https://github.com/chanfr/cppunitTest

   Además deberá generar el proyecto, compilarlo y ejecutar sus test mostrando el resultado con xunit. Para ello será necesario ejecutar las siguientes instrucciones:
   mkdir -p build
   cd build
   cmake .. -DUSEXUNIT=ON
   Make -j4

   Y para ejecutar los tests:
   ./build/test/aritTest

   Añadir el reporte que se genera en *.xml al reporte de xunit

2. Job python
   El código de python debe estar disponible en /opt/my_data asi que lo primero que haremos será copiarlo al workspace.
   El siguiente paso será ejecutar los test y añadir el resultado a xunit al igual que hemos hecho en el apartado de c++
   py.test --junitxml results.xml <tuScriptDeTest.py>

3. Shell
   Ejecutar la práctica 1 desde un job en jenkins.

Una vez configurados los jobs, crearemos vistas para agrupar los jobs, uno para los proyectos de c++, otro para los proyectos de python y otro para los de shell.

Por último se pide añadir un esclavo al master de jenkins.

La entrega de la práctica consistirá en una memoria con todos los pasos seguidos para la resolución de la misma.