DigitalHouse>

Certified Tech Developer
The Ultimate Degree

Infraestructura II

Actividad obligatoria e individual
Dificultad: ¡Alta!

Continuous Integration con Jenkinsfile Parte 5

Deploy

¡Seguimos con la práctica! En esta clase vamos a configurar la última etapa de nuestro pipeline, vamos a desplegar nuestra aplicación en un servidor. ¡Y dejarla lista para ser usada!

Nuestro es desafío es:

- Tener arriba Jenkins configurado con Docker.
- Tener arriba Nexus 3.
- Agregar nuestro template para el servidor. En este punto debemos utilizar el contenedor **server** que se ofreció en la práctica de clase 16 (link).
- Agregar el paso de deploy en el Jenkinsfile y levantar nuestro contenedor con la app.

En la siguiente página se encuentra la resolución. Continúa únicamente para realizar una autoevaluación.

Resolución

- **Tener arriba Jenkins configurado con Docker.**

Este paso fue explicado por el profesor en la primera sección de la clase.

» Se sigue la siguiente guía: link.

- **Tener arriba Nexus 3.**

Este paso se realizó en la práctica de la clase anterior.

» Se puede revisar aquí: link.

- **Agregar nuestro template para nuestro servidor.** En este punto debemos utilizar el contenedor **server** que se ofreció en la práctica de la clase 16: link.

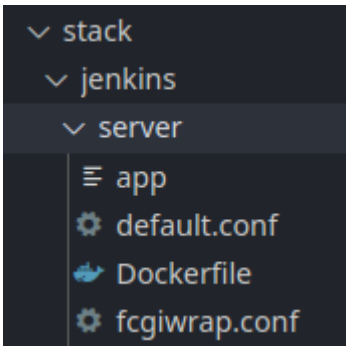
La aplicación de Java que venimos utilizando escribe en consola (salida estándar) un mensaje.

Para efectos didácticos queremos desplegarla en un servidor web como NGINX, donde pueda ser consumida de forma directa por los usuarios. NGINX trabaja con otro servicio FastCGI (php-fpm) que se encarga de ejecutar PHP y devolver la salida a NGINX para que la retorne al cliente (explorador web).

En nuestro caso, como no vamos a ejecutar PHP, sino nuestra aplicación Java, debemos utilizar el contenedor de NGINX citado que posee las modificaciones necesarias para correr nuestra app.

Descargamos el ZIP de la práctica, lo descomprimimos y copiamos la carpeta **server** con su contenido a nuestro repositorio —donde tenemos nuestro Jenkinsfile y el código de nuestra aplicación—.

En la siguiente imagen vemos la carpeta server y sus archivos dentro de la carpeta stack del ZIP descomprimido:



Si observamos el archivo ***Dockerfile***, vemos que contiene una línea que hace referencia a una aplicación ***.jar***:

```
COPY holamundo.jar /app/holamundo.jar
```

Con nuestro pipeline vamos a poner la aplicación en su lugar para poder desplegarla dentro del contenedor y lanzarla.

- **Agregar el paso de deploy en el Jenkinsfile y levantar nuestro contenedor con la app.**

En nuestro Jenkinsfile hacemos la etapa de deploy como se observa:

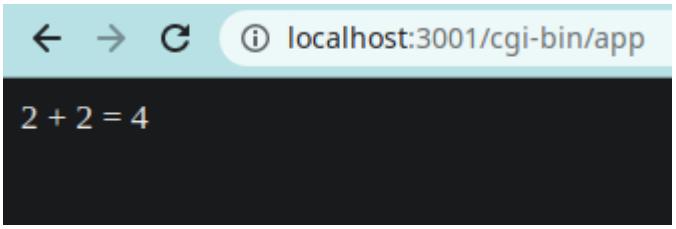
```
stage('Deploy') {
    steps {
        dir('maven-adderapp') {
            script{
                pom = readMavenPom file: "pom.xml";
                files = findFiles(glob: "target/*.${pom.packaging}");
                env.file = files[0].path;
            }
        }

        sh "cp maven-adderapp/$file server/holamundo.jar"
```

```
                dir ('server') {
                    sh """
                        docker rm -f server || true
                        docker build . -t server
                       docker container run -d --name server --publish 3001:80
server
                    """
                }
            }
        }
    }
```

¡Vamos a revisarlo! Podemos distinguir tres partes de código:

- El primer '*dir('maven-adderapp')*' obtendrá el nombre del archivo ejecutable de nuestra aplicación.
- Luego, con el comando '*cp*' lo copiaremos para que quede dentro del directorio '*server*' donde tenemos toda la definición de nuestro contenedor, quien hará el rol del ambiente donde desplegamos nuestra aplicación.
- Para terminar en '*dir('server')*' construimos la imagen y corremos el contenedor.

Una vez terminado el pipeline tendremos un contenedor corriendo **server** con nuestra aplicación en http://localhost:3001/cgi-bin/app. Como mi aplicación es el adderapp y su ejecución es sumar "2+2" visualizo lo siguiente:



Este es el repositorio rama 'deploy' del código usado para este ejercicio.

Conclusión

Con esta práctica cerramos el trabajo que venimos realizando en las últimas clases. Pudimos hacer un **proceso completo de CI/CD** y exploramos el uso de Jenkins y Nexus —además de otras herramientas— para poder **compilar, testear, desplegar y archivar el release de nuestra aplicación**.

Aprendimos que el armado de todo este proceso es complejo y que hacerlo de manera manual no sería eficiente —sin contar con los errores humanos que podríamos cometer por fatiga en la repetición—. Nuestro pipeline nos ahorra tiempo y hace que el proceso corra siempre de la misma forma y sin errores en los distintos pasos.