9. Integración y despliegue continuo con Jenkins y docker

Para acabar el curso vamos a salirnos un poco del tema pero sin salirnos del todo. Pretendo que ,usando contenedores, podamos montar un "pseudo" flujo de despliegue continuo mediante un servidor Jenkins que lanzaremos desde un contenedor. De esta manera mostraremos otro entorno donde los contenedores son útiles y abriremos el campo de conocimientos demostrando la utilidad de este tipo de conceptos dentro del mundo del desarrollo de software actual.

¿Qué es Jenkins?

Jenkins es un **servidor de automatización** opensource que me va a permitir automatizar ciertas partes del proceso de desarrollo de software. Tiene un sistemas de plugins muy completo que permite integrar en cada una de estas fases herramientas para el control de versiones, construcción de software, testing etc...

https://commons.wikimedia.org/wiki/File:Jenkins_logo_with_title.svg. *Jenkins, servidor de automatización* (CC BY-SA)

¿Pero qué vamos a automatizar?

Una vez hayamos levantado un servicio de Jenkins definiremos un flujo automático (PIPELINE) con las siguientes fases:

1. **PRIMER PASO:** Descarga de un código desde un repositorio de GitHub y construcción (build)
2. **SEGUNDO PASO:** Simulación de Testing. No entraremos más en esta fase porque no es materia para este curso.
3. **TERCER PASO:** Generación de una nueva imagen docker que contenga ese código a través de un fichero Dockerfile que esté conectado con DockerHub.
4. **CUARTO PASO:** Despliegue del código en un servidor que serán también un contenedor.

Este flujo es un flujo ad-hoc para este curso que no responde exactamente a las definiciones generales que vamos a presentar en el próximo apartado, pero nos va a servir para repasar conceptos que hemos visto a la vez que nos introducimos en el mundo de la automatización.

9.1 Conceptos importantes

CONTINUOS INTEGRATION (CI)

Es una **práctica en el desarrollo** de software que consiste en hacer **de manera automática integraciones** dentro de un proyecto. Este concepto de integración se refiere **a la compilación y a la realización de las pruebas** necesarias para comprobar la corrección de software. Es automático y se inicia tras realizar cambios en el sistema de control de versiones (Git,Svn, Mercurial etc..).

Se hace especial hincapié en la automatización de los test para comprobar que no hay errores tras añadir cambios. Dentro de esas pruebas puede haber pruebas unitarias (U), de integración (I) , de aceptación (A) etc...

CONTINUOS DELIVERY (CD)

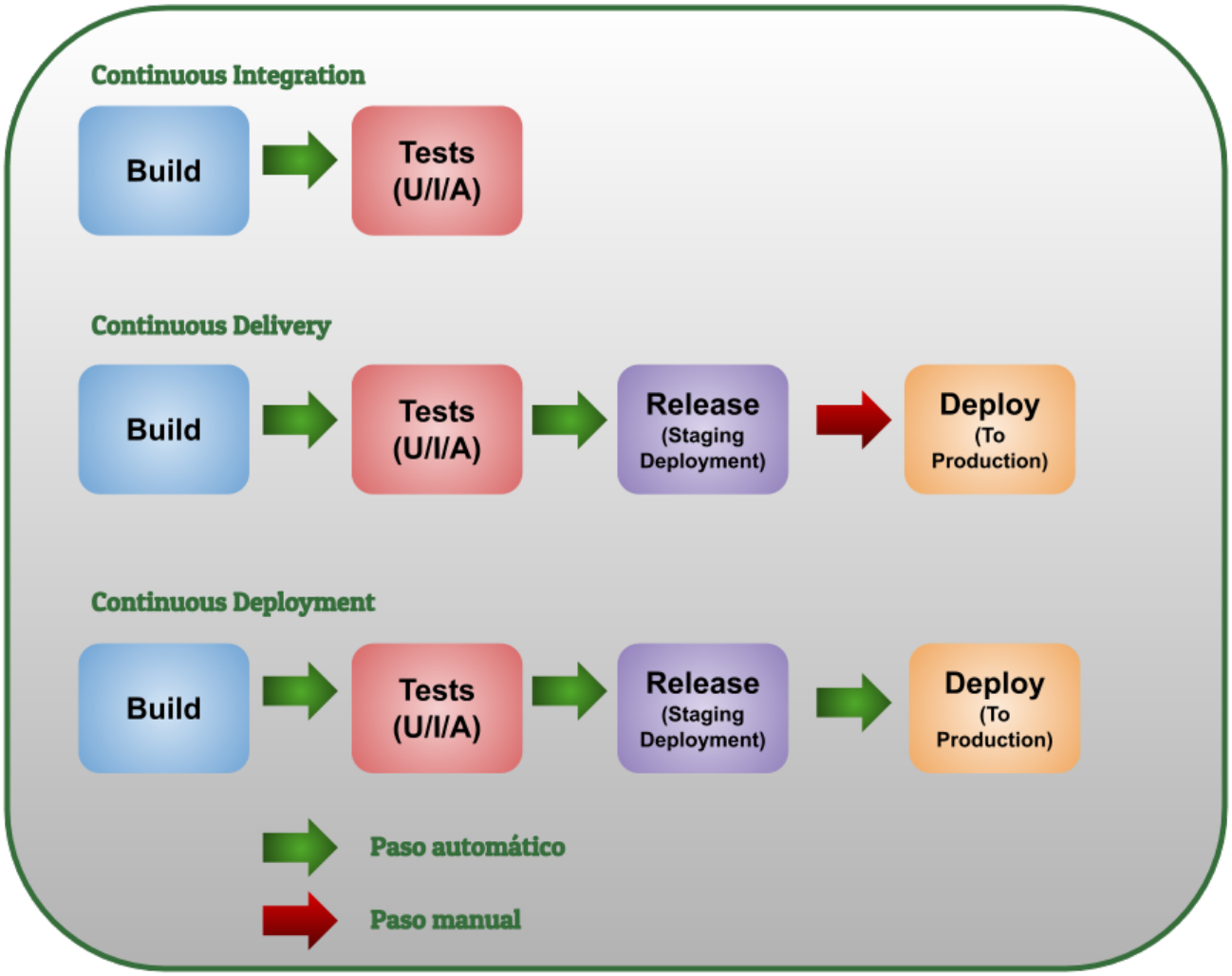
Es una aproximación al proceso de desarrollo de software que pretende que se puedan **generar versiones de nuestro sofware en cualquier momento y de una manera segura o confiable.**

Es por lo tanto una **evolución del proceso de Continuous Integration (CI)** a la que se añade la generación del release del software para su **entrega manual (Staging Deployment).** Esto significa que tras tras el proceso de build, testing y generación de la versión hay que **"hacer click" en un botón para desplegar la aplicación.** No hay despliegue sin intervención humana pero está todo preparado para hacerse.

CONTINUOS DEPLOYMENT (CDep)

Va **un paso más allá del Continuos Delivery (CD), se despliega el código a producción de manera automática** y todo empieza tras validar los cambios en el código en el sistema de control de versiones. **No hay intervención humana**. Me permite realizar despliegues en cualquier momento y acelera todo el proceso.

Podemos ver la diferencia entre esos conceptos de manera gráfica.

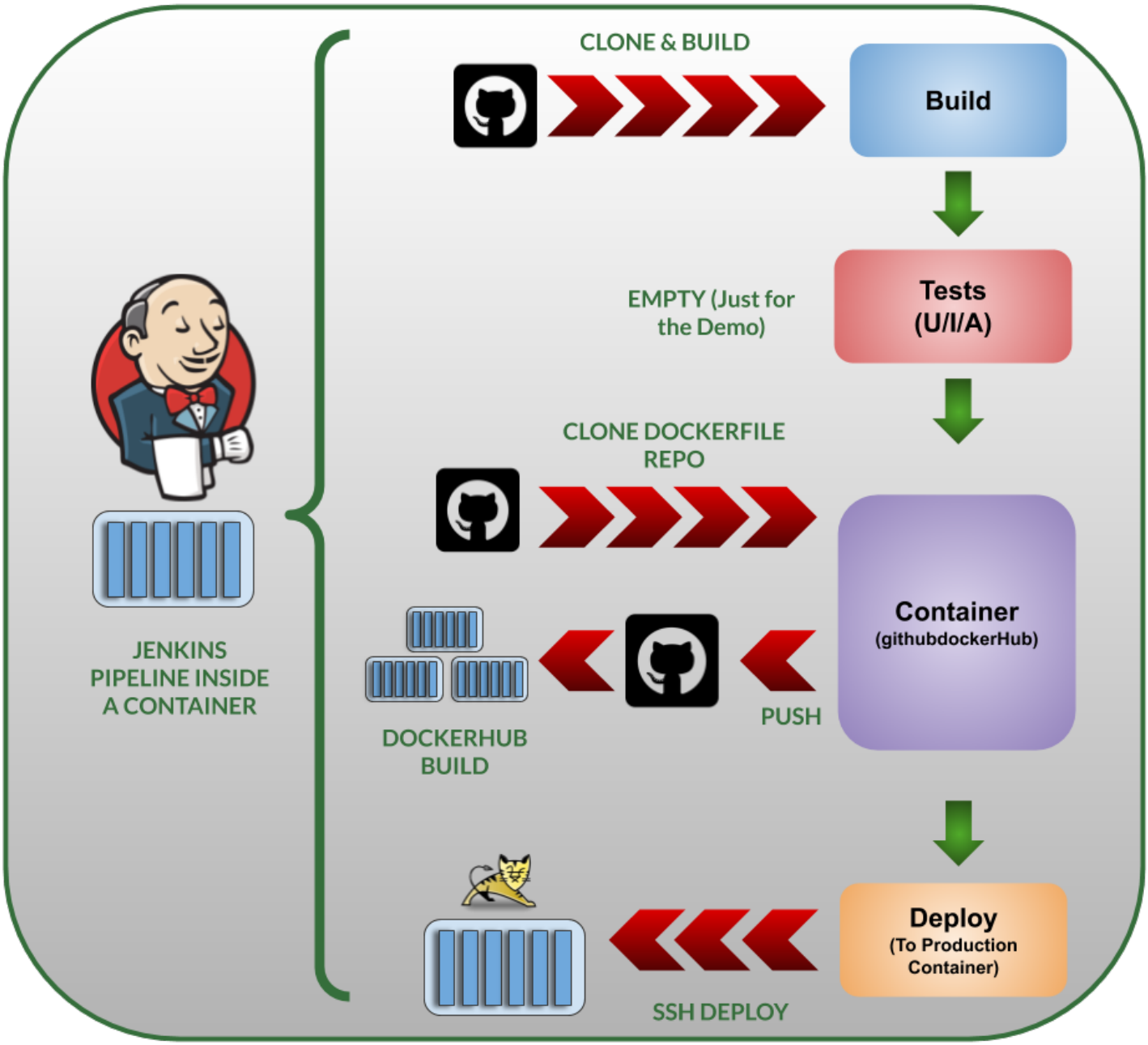


Juan Diego Pérez Jiménez. *CI / CD / CDep* (Dominio público)

9.2 Ejemplo práctico

Para finalizar, tal y como habíamos anunciado en el primer apartado de este tema, vamos a realizar una práctica que cubra muchos de los aspectos que hemos visto a lo largo del curso y que introduzca ideas y conceptos relacionados con la unión de los contenedores y de los procesos de integración (CI) , entrega (CD) y despliegue continuo (CDep). El proceso no es exactamente de ese tipo de procesos, es un proyecto "ad-hoc" para mis clases pero nos va a ayudar a pensar en automatización de procesos.

Es una práctica voluntaria, puede resultar compleja, pero el objetivo es vislumbrar el mundo de posibilidades que nos abre el uso de las tecnologías vistas. Todos los archivos necesarios serán proporcionados. Nuestro objetivo va a ser un automatizar el siguiente proceso:



Juan Diego Pérez Jiménez. *Pipeline de Jenkins - Pseudo CDep* (Dominio público)

PASOS DEL PROCESO

Los pasos del proceso son los pasos del Pipeline de Jenkins y serán:

1. **BUILD:** Descargar el código del ejemplo y compilarlo.
2. **TESTS:** Realizar los Test. En este ejemplo se deja vacío para no complicar más las cosas. Si nuestro proyecto tuviera pruebas sería el lugar para ejecutarlas.
3. **CONTAINER:** Clonaremos el repositorio de Github donde tengo el Dockerfile y tras añadir el fichero resultante de la compilación se hará un push. Este repositorio estará conectado a su vez con un repositorio de DockerHub que será el que haga el build de manera automática.
4. **DEPLOY:** Pondremos el archivo en un contenedor Tomcat al que accederemos por SSH.

RECURSOS NECESARIOS

Vamos a necesitar los siguiente recursos:

- Un contenedor de la imagen jenkins/jenkins:lts que lanzaremos con el siguiente comando docker. En ese contenedor instalaremos la herramienta Maven y los plugins Maven Integration y SSH Pipeline Steps.

```
docker run -d --name serverJenkins -p 9393:8080 -p 50001:50000 jenkins/jenkins:lts
```

- Un contenedor de la imagen jperjim398/tomcatcursocep que contiene un servidor Tomcat y un servidor SSH que usaremos para conectarnos y desplegar la aplicación en el servidor Tomcat. Lanzaremos este contenedor con la siguiente orden docker.

```
docker run -d -it --name cep -p 9292:8080 -p 2222:22 jperjim398/tomcatcursocep
```

- El código de la aplicación del ejemplo que se puede encontrar aquí: https://github.com/jleetutorial/maven-project
- Un repositorio en github que tenga inicialmente el siguiente Dockerfile.

```
FROM tomcat:9.0.39-jdk11

COPY *.war /user/local/tomcat/webapps
```

- Un repositorio en DockerHub conectado con el repositorio anterior.
- El fichero Jenkinsfile que define el Pipeline.

```
def produccion = [:]
        produccion.name = 'curso'
        produccion.host = '172.17.0.2'
        produccion.user = 'root'
        produccion.password = 'root'
        produccion.allowAnyHosts = true
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {

                //Creo el directorio del código. Si no existe se crea

                dir('codigo') {
                    // Get some code from a GitHub repository
                    git 'https://github.com/jleetutorial/maven-project.git'

                    // Run Maven on a Unix agent.
                    sh "mvn clean package -DskipTests"
```

```
            // To run Maven on a Windows agent, use
            // bat "mvn -Dmaven.test.failure.ignore=true clean package"
        }

    }

    post {
        // If Maven was able to run the tests, even if some of the test
        // failed, record the test results and archive the jar file.
        success {
            archiveArtifacts 'codigo/webapp/target/*.war'
        }
    }
}
stage('Test') {
    steps {
        sh "echo 'Realización de algunos Test'"
    }

}
stage('Container') {
    steps {
        sh "echo 'Creo el contenedor'"
        dir('contenedor') {
            withCredentials([usernamePassword(credentialsId: 'gitprueba', passwordVariable: 'password',
usernameVariable: 'usuario')]) {
                git 'https://github.com/pruebainf/dockerimage-from-jenkins-pipelin.git'
                sh('''
                    cp ../codigo/webapp/target/webapp.war .
                    git add .
                    git config --global user.email 'testif@iesalixar.org'
                    git config --global user.name  'testif@iesalixar.org'
                    git commit -m 'Desde Jenkinsfile'
                    git config --local credential.helper "!f() { echo username=\\$usuario; echo password=\\$password; };
f"
                    git push origin master
```

```
                ''')
            }
        }

    }
    stage('Deploy') {
        steps {
            sh "echo 'Desplegando'"
            sshPut remote: produccion, from: 'codigo/webapp/target/webapp.war' , into: '/usr/local/tomcat/webapps/'
        }

    }

  }
}
```