everis
an NTT DATA Company

Automatización Vol.2 Continuous Testing

Francisco Aznar Ferrer

HPC Alicante 2018

Acerca de mí

everis
an NTT DATA Company

A.K.A - Francisco Aznar Ferrer

5 años de experiencia como

Automation Test Engineer

&

Software Quality Assurance

- Microservices, DevOps culture, QA Engineering -

Línea del curso

Automatización Vol 2 Continuous Testing

Objetivo del curso: Introducción al mundo de la automatización de pruebas.

Duración (h): 30h
Formato: 6h/día en seis días consecutivos

Perfiles
All

Formadores:

- Francisco José Aznar Ferrer

Conocimientos previos requeridos:

- Experiencia previa en el mundo del desarrollo de Software.
- Conocimientos básicos de programación (Java).

Contenidos del curso:

- Capitulo 1 – Descubriendo **Docker**.
- Capitulo 2 – Descubriendo el **Continuous Testing.**
- Capitulo 3 – **Jenkins** como sistema CI**.**
- Capitulo 4 – **Selenium Grid** con **Docker Compose.**

Índice

Contenido del Curso

- **Descubriendo *Docker*.**
  - Que es Docker?
  - Docker vs Máquinas Virtuales.
  - Imágenes y Contenedores.
  - ***P1 – Descargar Imágenes y levantar Contenedores.***
  - Construir Imágenes con Dockerfile.
  - ***P2 – Construir nuestra imagen.***
- **Descubriendo el Continuous Testing.**
  - Entendamos CI, CT & CD.
  - Release Pipeline.
- ***Jenkins como sistema de CI.***
  - Conozcamos *Jenkins CI*.
  - ***P3 – Instalar y Configurar Jenkins.***
  - ***P4 – Solucionar problemas en el entorno.***
  - ***P5 – Integrar Cucumber Report.***
- ***Selenium Grid* con *Docker Compose*.**
  - Que es *Selenium Grid?*
  - ***P6 – Montar Selenium Grid con Docker.***
  - *Docker Compose* como Aliado.
  - ***P7 – Combinar Jenkins + Selenium Grid con Docker Compose.***

everis
an NTT DATA Company

Capitulo 1

Descubriendo Docker

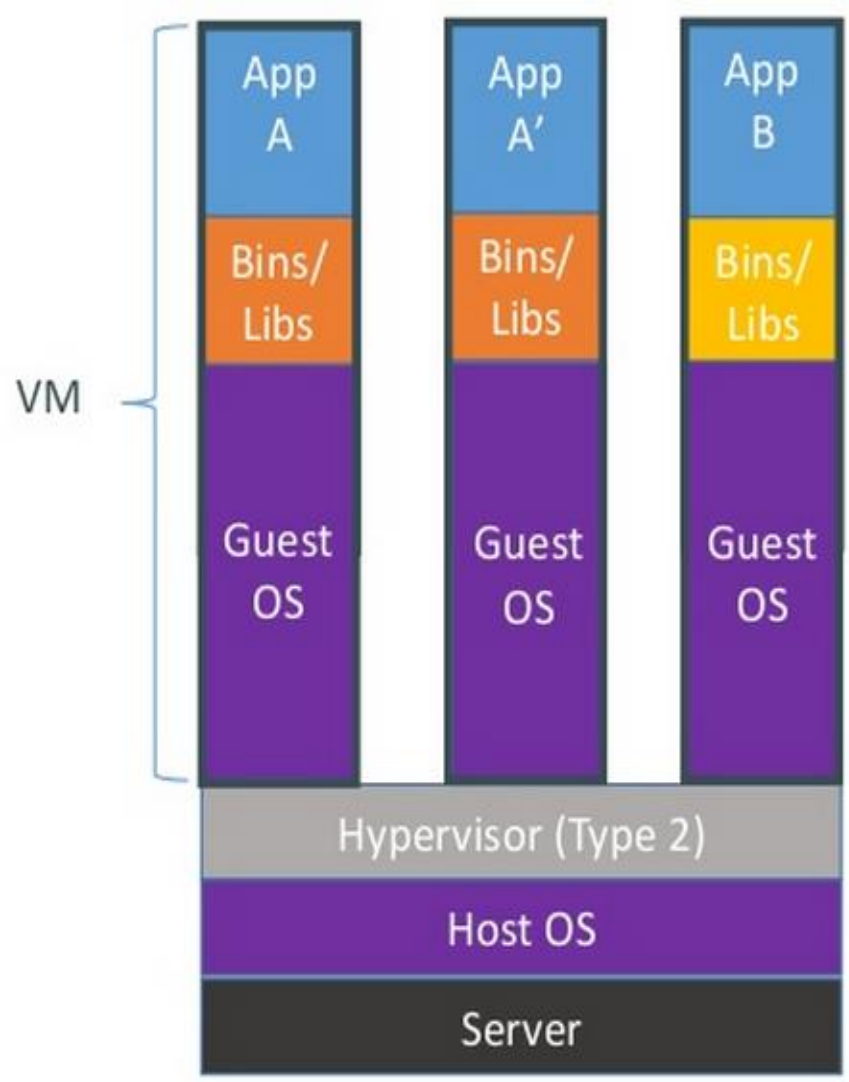
Docker es una herramienta open-source que nos permite realizar una 'virtualización ligera', con la que poder empaquetar entornos y aplicaciones que posteriormente podremos desplegar en cualquier sistema que disponga de esta tecnología.

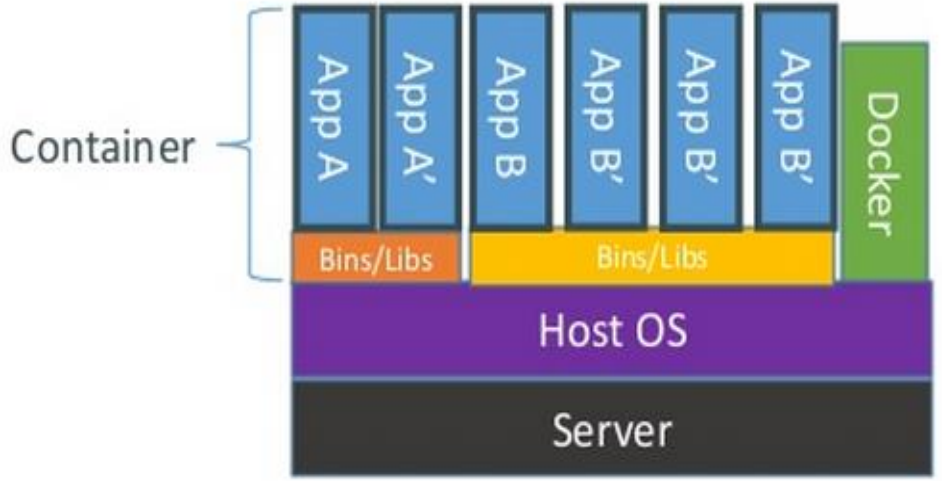
Para ello Docker extiende LXC (LinuX Containers), que es un sistema de virtualización que permite crear múltiples sistemas totalmente aislados entre sí, sobre la misma máquina o sistema anfitrión.

Docker vs Máquinas Virtuales

La gran diferencia es que una máquina virtual necesita contener todo el sistema operativo mientras que un contenedor Docker aprovecha el sistema operativo sobre el cual se ejecuta, comparte el Kernel del sistema operativo anfitrión e incluso parte de sus bibliotecas.

Docker vs Máquinas Virtuales





everis
an NTT DATA Company

Pros

- Las instancias se arrancan en pocos segundos.
- Es fácil de automatizar e implantar en entornos de integración continua.
- Existen multitud de imágenes que pueden descargarse y modificarse libremente.

Contras

- Sólo puede usarse de forma nativa en entornos Unix aunque se puede virtualizar gracias a boot2docker tanto en OSX como en Windows.
- Las imágenes sólo pueden estar basadas en versiones de Linux modernas (kernel 3.8 mínimo).
- Como es relativamente nuevo, puede haber errores de código entre versiones.

Los términos que hay que manejar con **Docker** son principalmente 2, las **imágenes** y **contenedores**.

- Las imágenes en **Docker** no son mas que un sistema operativo base, con un conjunto de aplicaciones empaquetadas.

- Un contenedor en **Docker** es la instanciación o ejecución de una imagen.

"Haciendo una analogía con la POO una imagen es una ***clase*** y un contenedor es la instanciación de una clase, es decir un ***objeto***"

Practica 1

Descargar Imágenes y levantar Contenedores

Practica 1

Descargar Imágenes y levantar Contenedores

1 – Registrarse en ***Docker Hub***.

- ***https://hub.docker.com/***

2 – Bajarnos una Imagen de Docker ya definida.

- ***docker pull sonarqube***
- ***docker images***

3 – Levantar nuestro primer contenedor.

- ***docker run -d --name sonarqube -p 9000:9000 sonarqube***
- ***docker ps***

4 – Detener el contenedor.

- **docker stop sonarqube**
- **docker ps –a**
- **docker rm sonarqube**

Practica 2

Construir nuestras imágenes

- **FROM :** Indica la imagen que tomamos como base, en este caso la imagen oficial de Ubuntu
- **MAINTAINER:** Especifica el autor de la imagen.

- **ENV:** Definimos una variables de entorno en la imagen base.
  - **http_proxy** http://user:pass@proxy/ (Definimos la variable http_proxy)
  - **https_proxy** http://user:pass@proxy/ (Definimos la variable https_proxy)

- **RUN:** Ejecuta una sentencia sobre la imagen base
  - **apt-get update** (actualiza los repositorios de Ubuntu)
  - **apt-get install apache2 -y** (Instala el apache)
  - **echo "<h1>Apache & Docker by Automation Vol 2 Training</h1>" > /var/www/html/index.html** (crea un fichero index.html)

- **EXPOSE:** Exponemos el puerto 80 del contenedor para que pueda ser mapeado por la máquina anfitrión.
- **ENTRYPOINT:** Indicamos que se ejecute apache2ctl -D FOREGROUND cada vez que arranquemos el contenedor.

Practica 2

Construir nuestras imágenes

1 – Crear un fichero ***Dockerfile***.

```
FROM ubuntu
MAINTAINER Francisco Aznar
ENV http_proxy http://user:pass@proxy/
ENV https_proxy http://user:pass@proxy/
RUN apt-get update
RUN apt-get install apache2 -y
RUN echo "<h1>Apache + Docker Automation vol 2</h1>" > /var/www/html/index.html
EXPOSE 80
ENTRYPOINT apache2ctl -D FOREGROUND
```

2 – Construir nuestra primera imagen.

- docker build -t {dockerhub.user}/apache .

3 – Subir nuestra imagen a Docker Hub (Docker Registry).

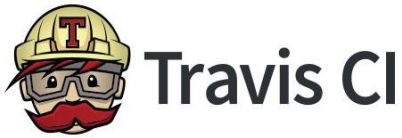
- docker push {dockerhub.user}/apache

Capitulo 2

Descubriendo el CT

Entendamos CI, CT & CD

- **Continuous Integration o Integración Continua (CI):** Entendemos como ***Integración Continua CI*** el proceso de automatización de la compilación y la prueba del código cada vez que la aplicación sufre algún cambio.



- **Continuous Testing o Calidad Continua (CT):** Definimos ***Calidad Continua CT*** como la practica de integrar y automatizar la actividad de pruebas en cada "*commit*", CT ayuda el equipo a optimizar mejor sus tiempos de manera mas eficiente.

- **Continuous Delivery o Entrega Continua (CD):** Podemos decir que la ***Entrega Continua CD*** es la practica de racionalizar y automatizar todos los procesos que conducen a la implementación de nuestra aplicación en cada cambio, desde probar la calidad de la compilación en entornos dedicados, hasta desplegar el nuevo software automáticamente en sus respectivos entornos.

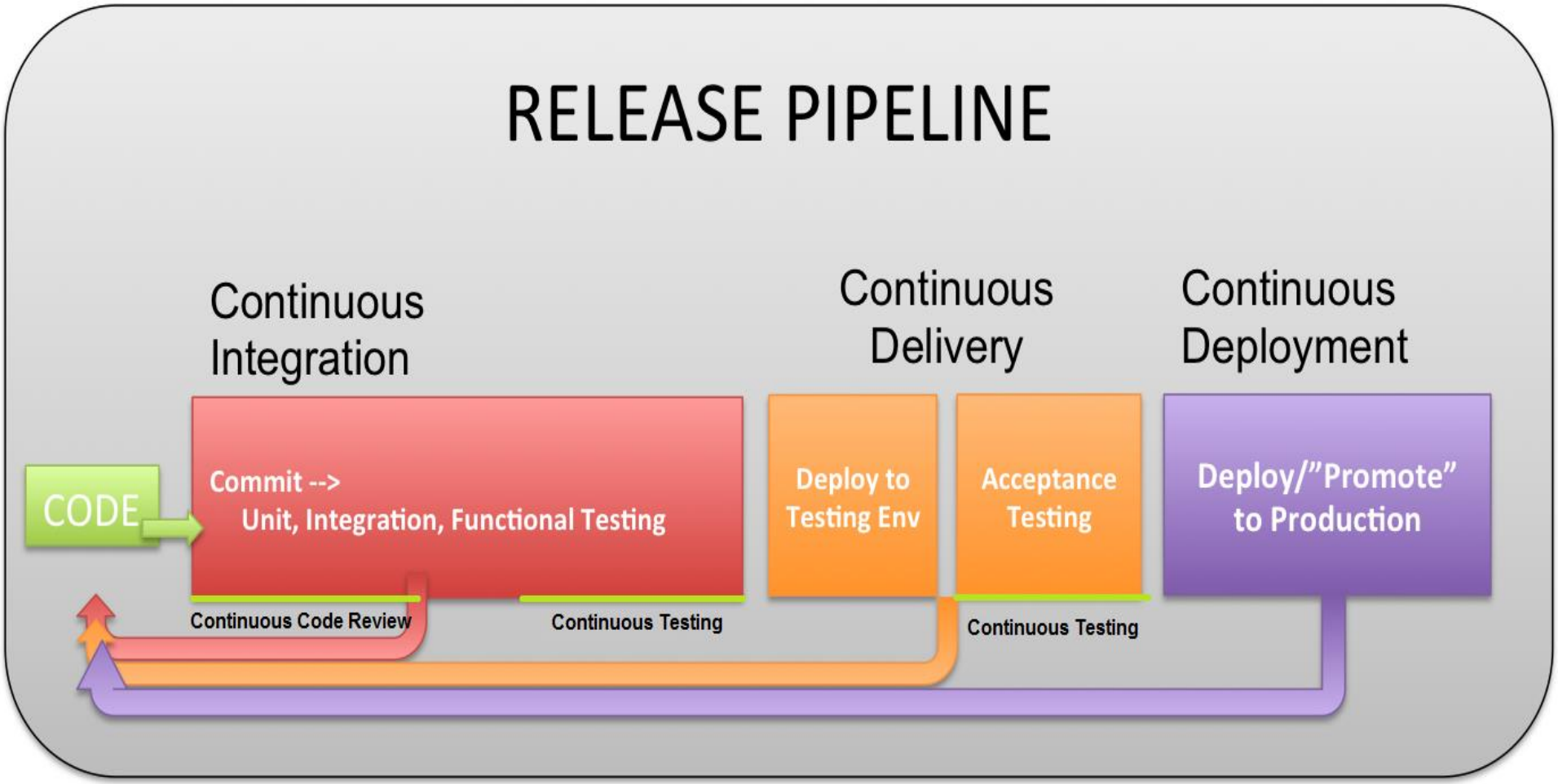
everis
an NTT DATA Company

Continuous Delivery Tool Chain

PERFORCE
Version everything.

Version Control	Build Automation	Infrastructure Automation	Test Automation	Scripting
PERFORCE	Jenkins	puppet labs	Cucumber	Perl
git	Electric Cloud	docker	Se	Ruby
mercurial	Travis CI	vmware	Katalon Intelligent Test Automation	RUNDECK
	TFS	VAGRANT		

everis
an NTT DATA Company



everis
an NTT DATA Company

Capitulo 3

Jenkins como sistema CI

everis
an NTT DATA Company

Que es Jenkins CI?

Jenkins es un servidor de integración continua, gratuito, ***open-source*** y actualmente uno de los más populares para esta función.

*"Esta herramienta, proviene de otra similar llamada Hudson, ideada por **Kohsuke Kawaguchi**, que trabajaba en Sun. Unos años después de que Oracle comprara Sun, la comunidad de Hudson decidió renombrar el proyecto a Jenkins, migrar el código a **GitHub** y continuar el trabajo desde ahí. No obstante, Oracle ha seguido desde entonces manteniendo y trabajando en **Hudson**."*

everis
an NTT DATA Company

Que papel juega Jenkins dentro del proceso CI?

La base de Jenkins son las tareas, donde indicamos qué es lo que hay que hacer en un ***build***. Por ejemplo, podríamos programar una tarea en la que se compruebe el repositorio de control de versiones cada cierto tiempo, y cuando un desarrollador quiera subir su código al control de versiones, este se compile y se ejecuten las pruebas.

Si el resultado no es el esperado o hay algún error, Jenkins notificará al desarrollador, al equipo de QA, por email o cualquier otro medio, para que lo solucione. Si el ***build*** es correcto, podremos indicar a Jenkins que intente integrar el código y subirlo al repositorio de control de versiones.

everis
an NTT DATA Company

Practica 3

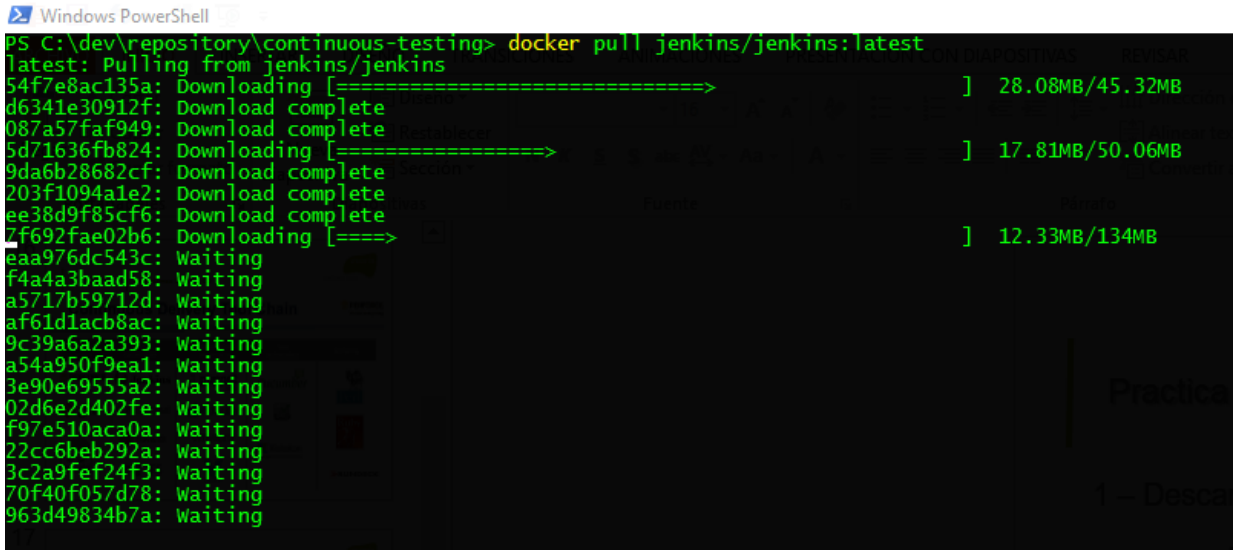
Instalar y configurar Jenkins

Capitulo 3

Practica 3 - Instalar y Configurar Jenkins

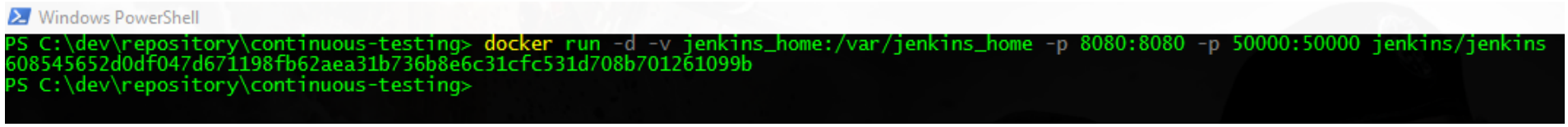
1 – Descargarse la imagen de Jenkins CI en el Docker Hub.

- docker pull jenkins/jenkins:latest



2 – Arrancar el contenedor y crear el volumen para persistir nuestra configuración.

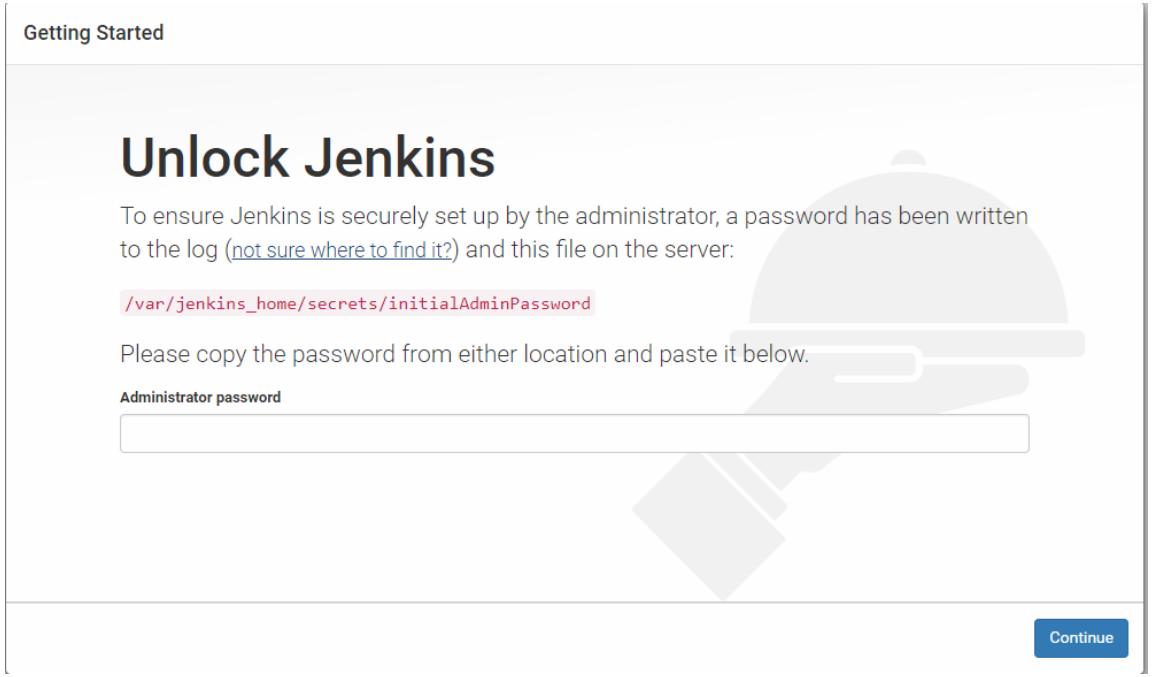
- docker run -d -v jenkins_home:/var/jenkins_home -p 8080:8080 -p 50000:50000 jenkins/jenkins



Practica 3 - Instalar y Configurar Jenkins

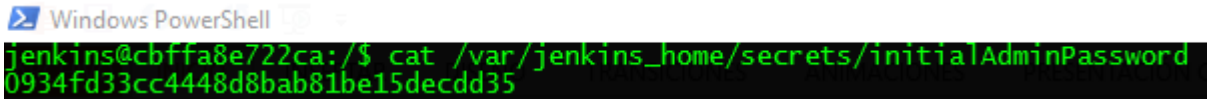
3 – Comenzar con Jenkins.

- Obtener la contraseña de arranque de Jenkins.

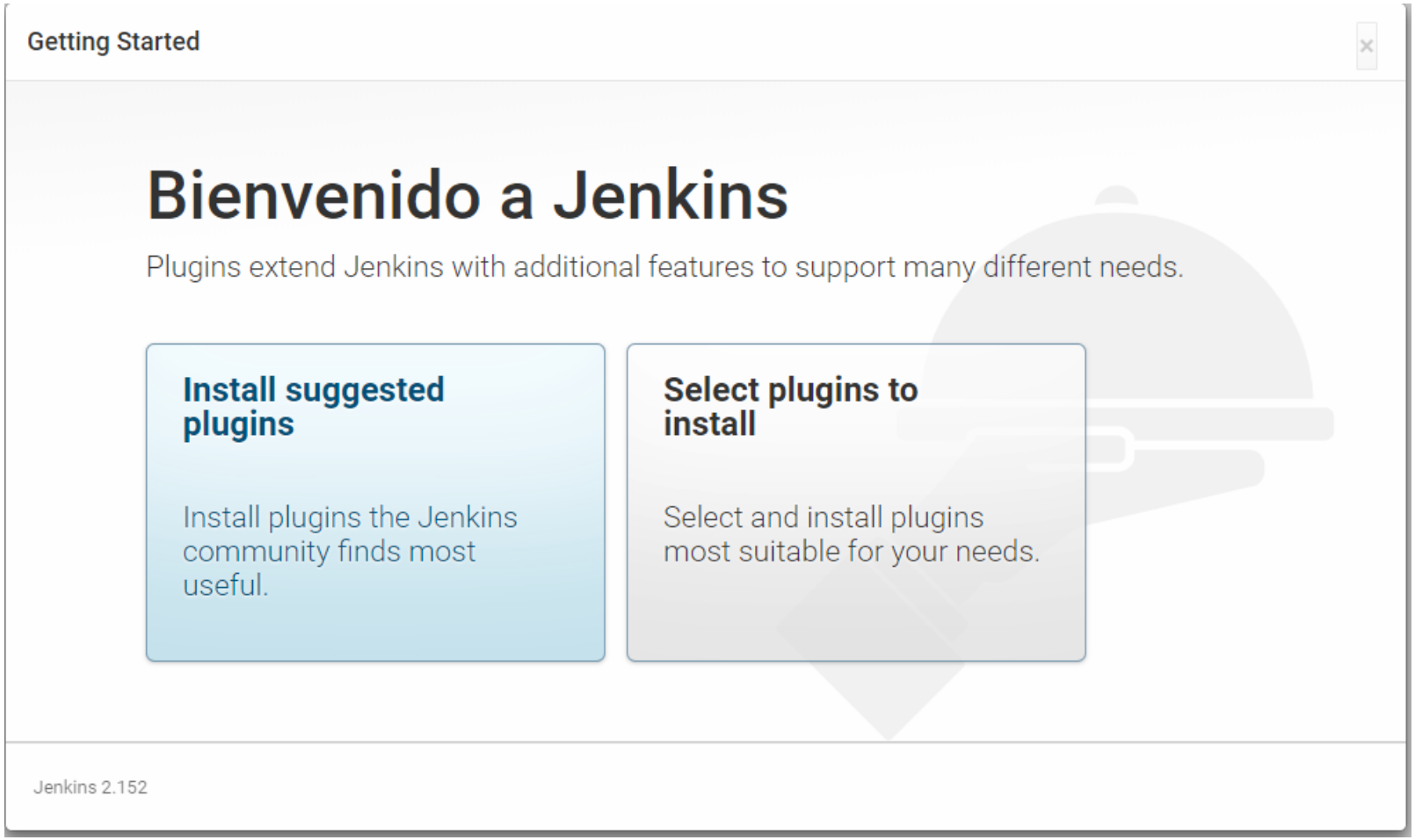


- docker exec -it {ID_Contenedor} bash

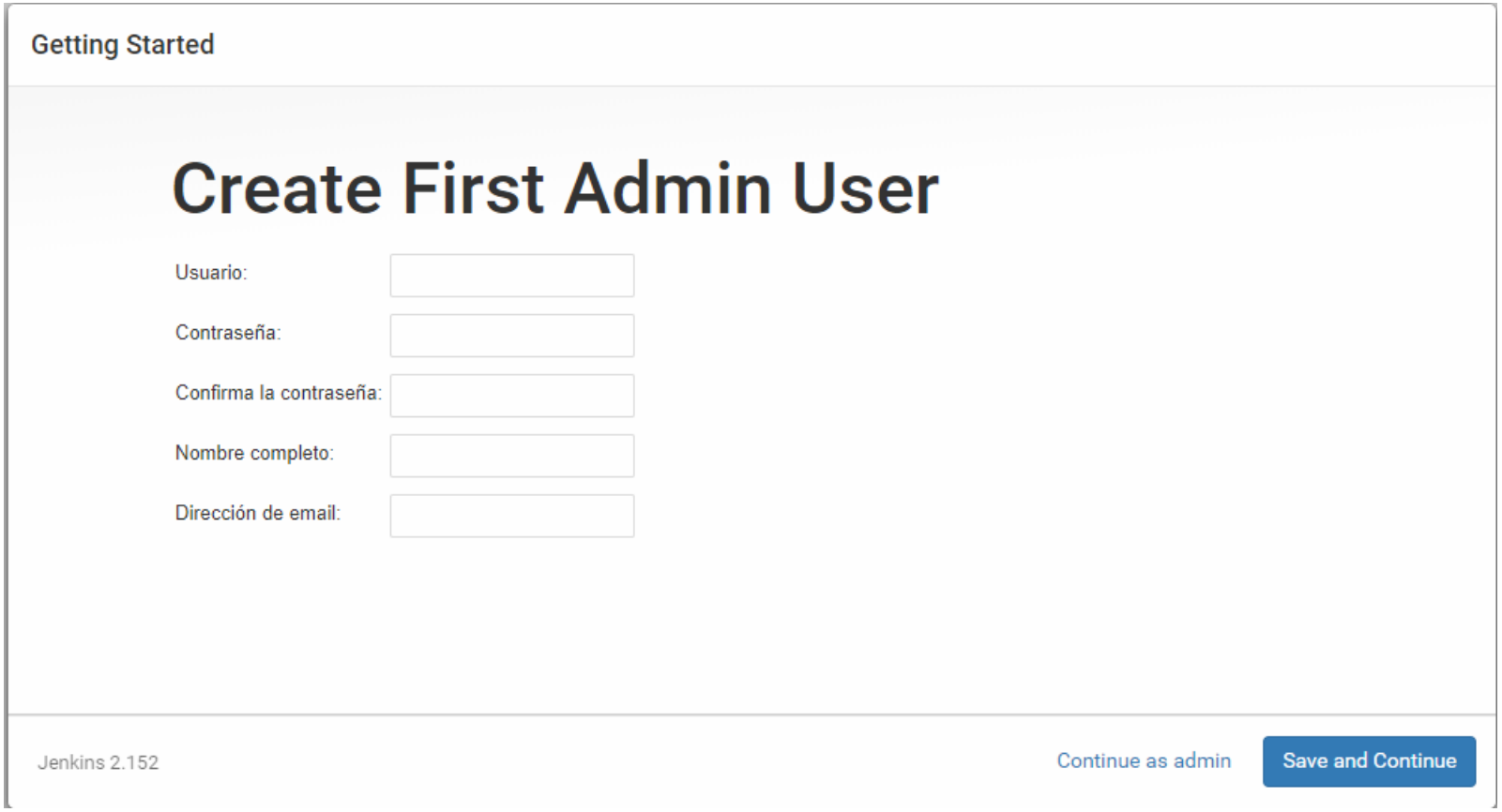
- cat /var/Jenkins_home/secrets/initialAdminPassword



4 – Instalar Plugins.



5 – Crear el usuario Admin.



Practica 3 - Instalar y Configurar Jenkins

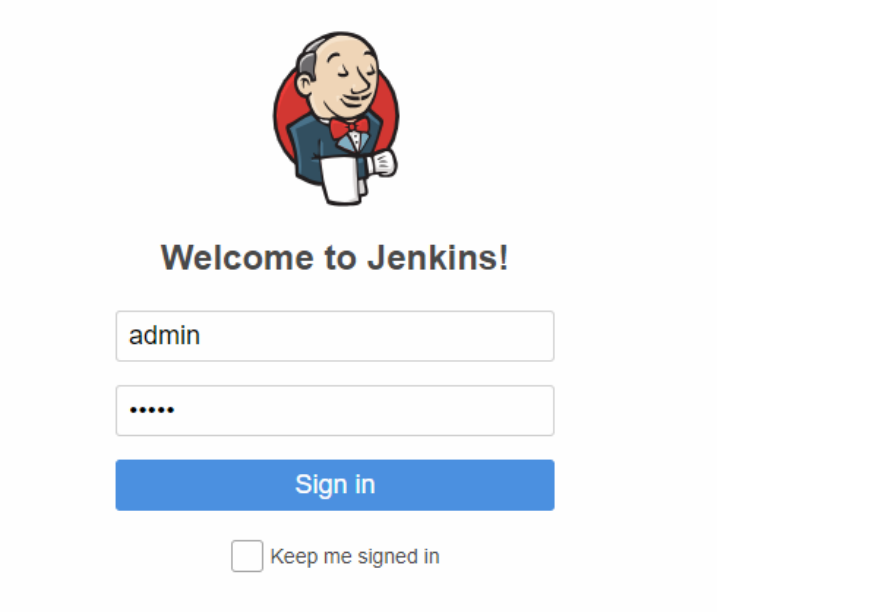
6 – Reiniciar nuestro Jenkins CI para aplicar cambios.

- docker ps

- docker stop {ID_Contenedor} y docker start {ID_Contenedor}

```
PS C:\dev\repository\continuous-testing> docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                                              NAMES
cbffa8e722ca        jenkins/jenkins     "/sbin/tini -- /usr/…"   20 minutes ago      Up 20 minutes       0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp   mystifying_ardinghelli
PS C:\dev\repository\continuous-testing> docker stop cbffa8e722ca
cbffa8e722ca
PS C:\dev\repository\continuous-testing> docker start cbffa8e722ca
cbffa8e722ca
```

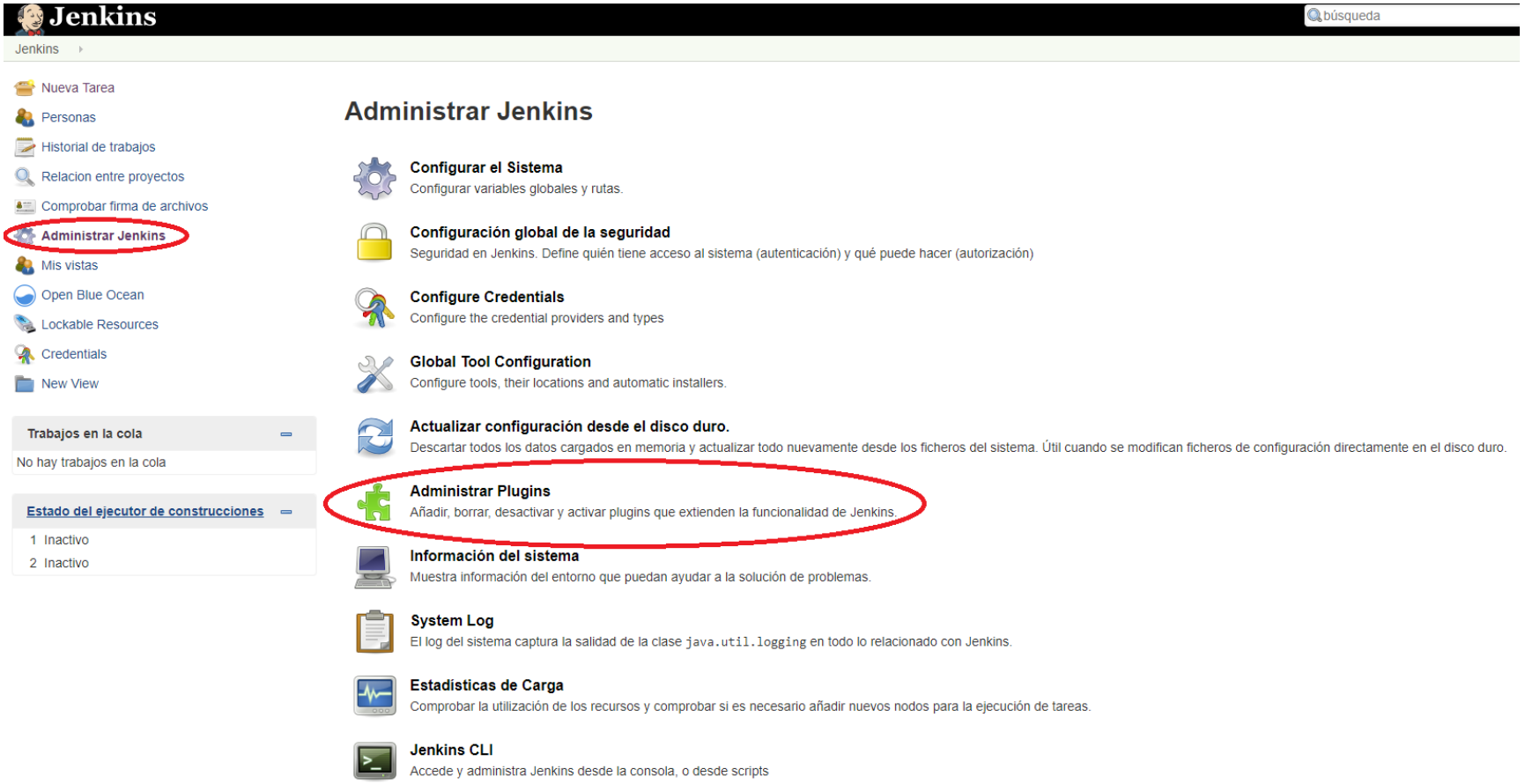
- Iniciamos sesión en Jenkins CI -> ***http://localhost:8080/login***



Practica 3 - Instalar y Configurar Jenkins

7 – Instalar y configurar el plugin para el versionado de código Mercurial.

- Acceder al menú ***Administrar Jenkins y*** al panel de ***Administrar Plugins.***

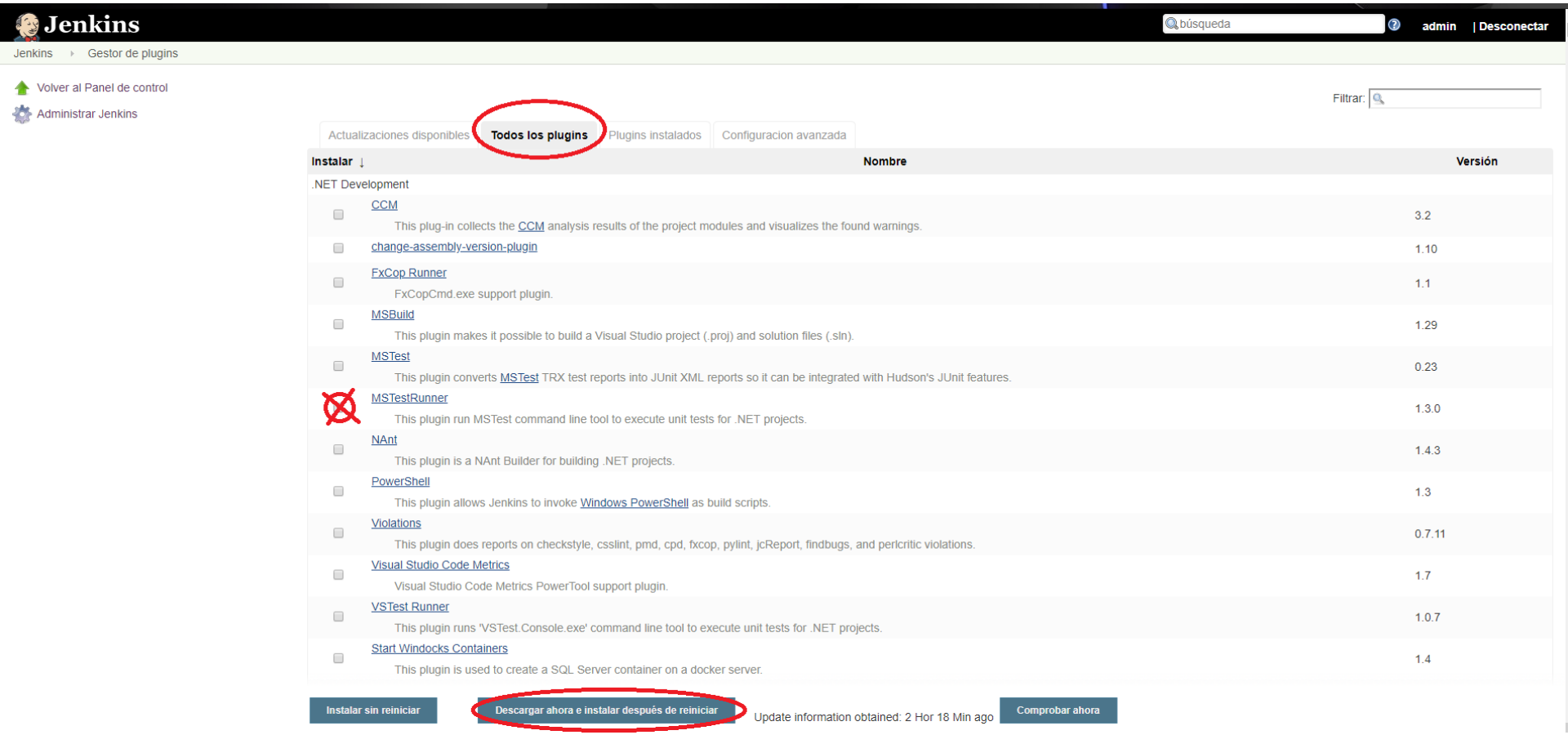


Capitulo 3

Practica 3 - Instalar y Configurar Jenkins

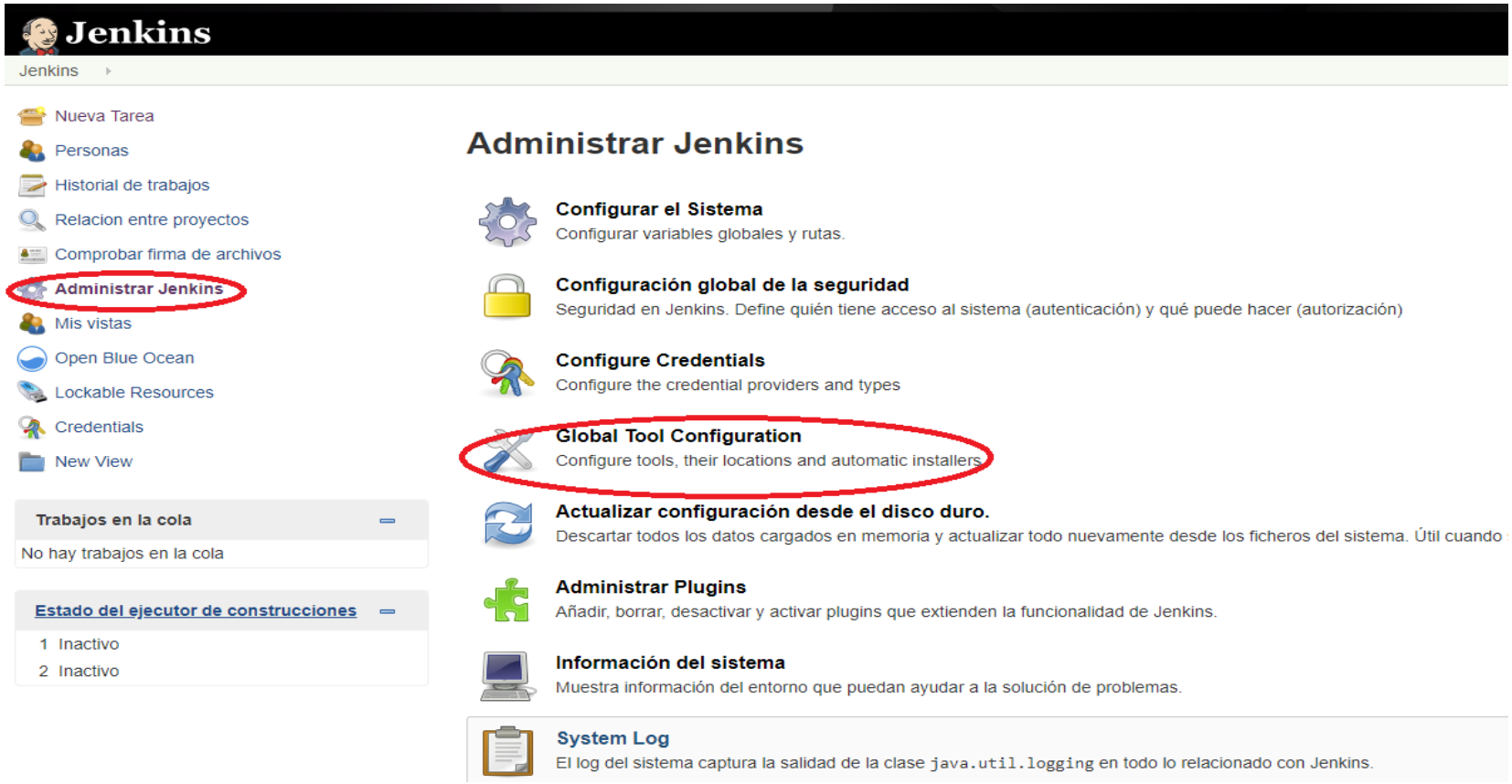
7.1 – Instalar y configurar el plugin para el versionado de código Mercurial.

- Elegir la pestaña de ***Todos los Plugins***, buscar y seleccionar el plugin de ***Mercurial.***
- Seleccionar la opción de ***Instalar ahora y reiniciar después de instalar.***



7.2 – Instalar y configurar el plugin para el versionado de código Mercurial.

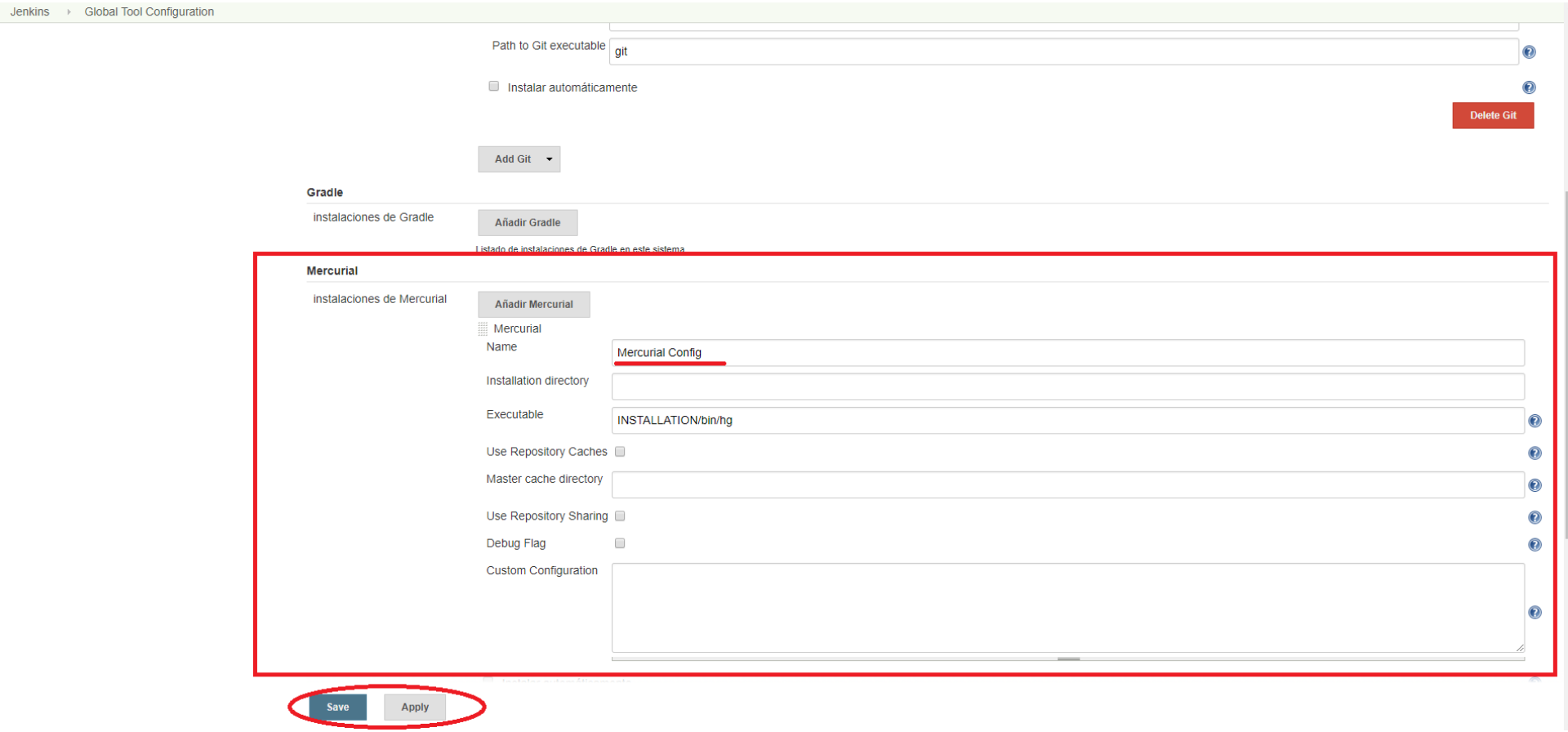
- Esperar al reinicio de Jenkins CI y ***Acceder a Administrar*** Jenkins de nuevo.
- Acceder al panel de ***Global Tool Configuration.***



Practica 3 - Instalar y Configurar Jenkins

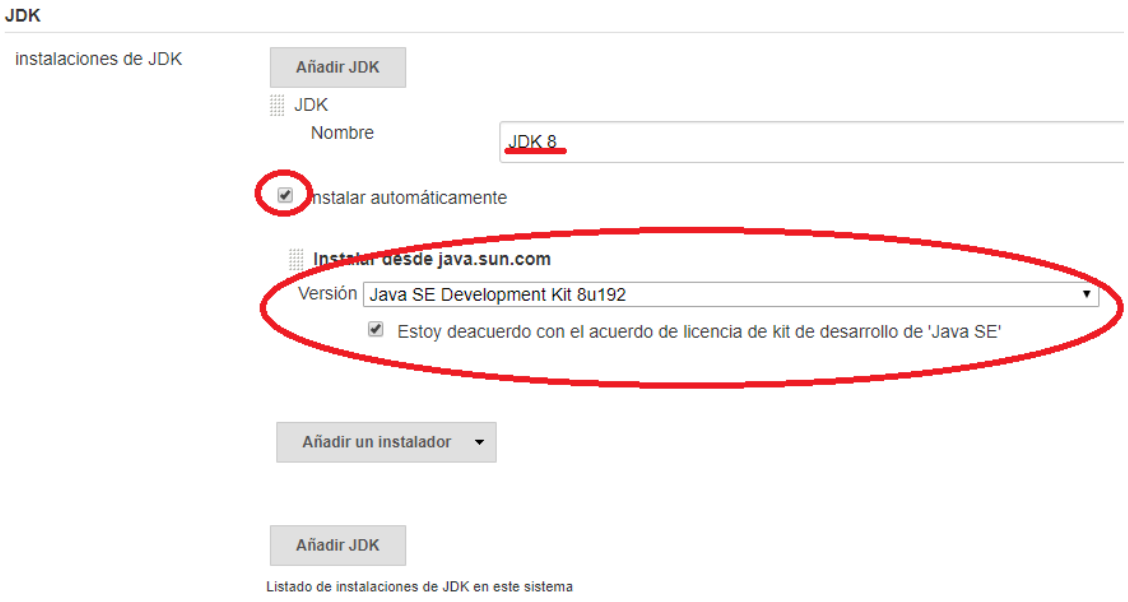
7.2 – Instalar y configurar el plugin para el versionado de código Mercurial.

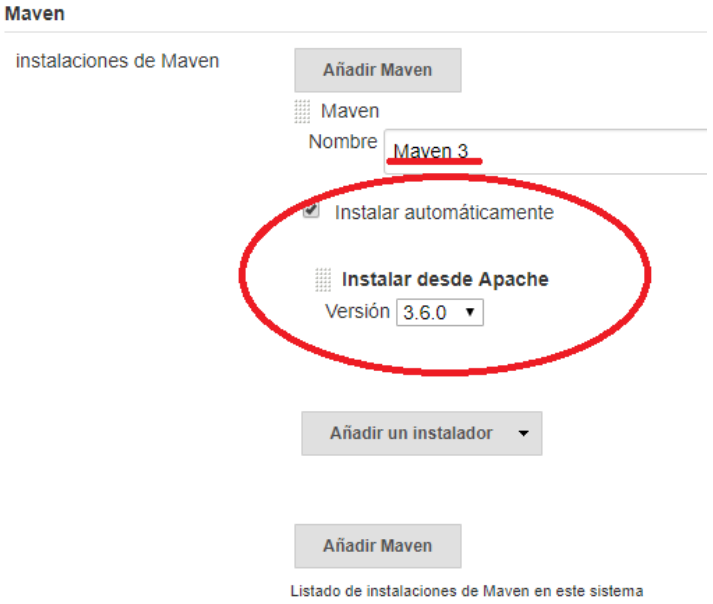
- Abrir la configuración de Mercurial desde ***Instalaciones de Mercurial.***
- Definir el nombre de la configuración y ***Apply*** and ***Save.***



8 – Configurar la instalación del gestor de dependencias ***Maven*** y ***JDK***.

- Acceder al panel de ***Global Tool Configuration.***
- Abrir la configuración de ***Maven*** y ***JDK*** desde ***Instalaciones de Maven*** e ***Instalaciones de JDK.***



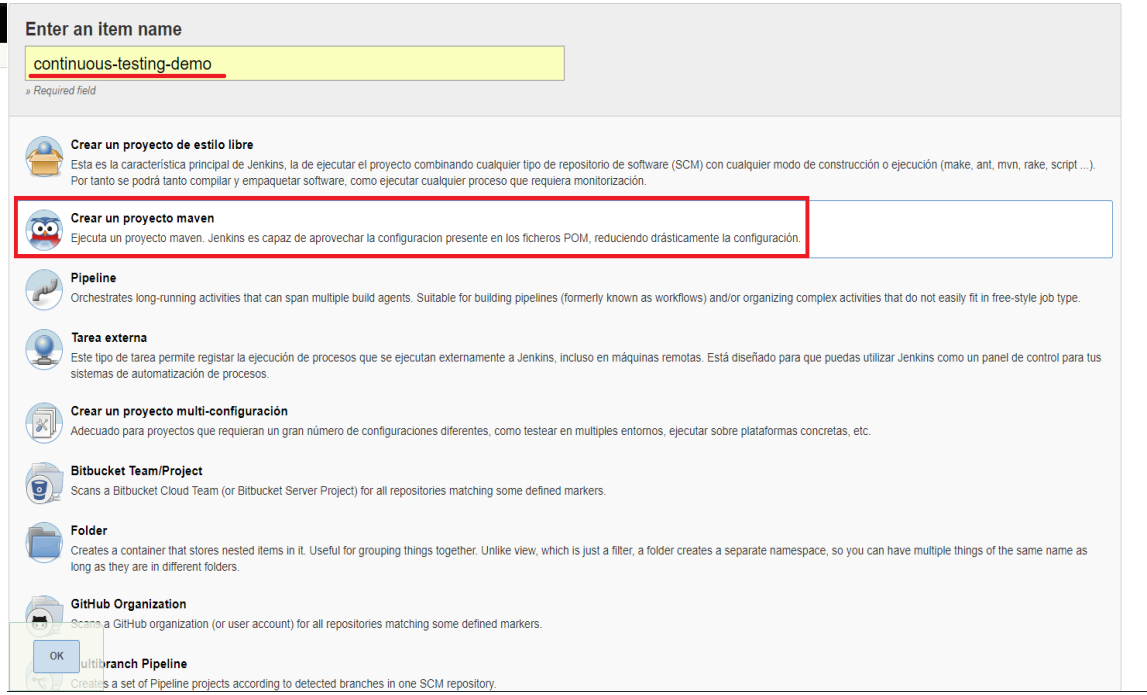


Capitulo 3

Practica 3 - Instalar y Configurar Jenkins

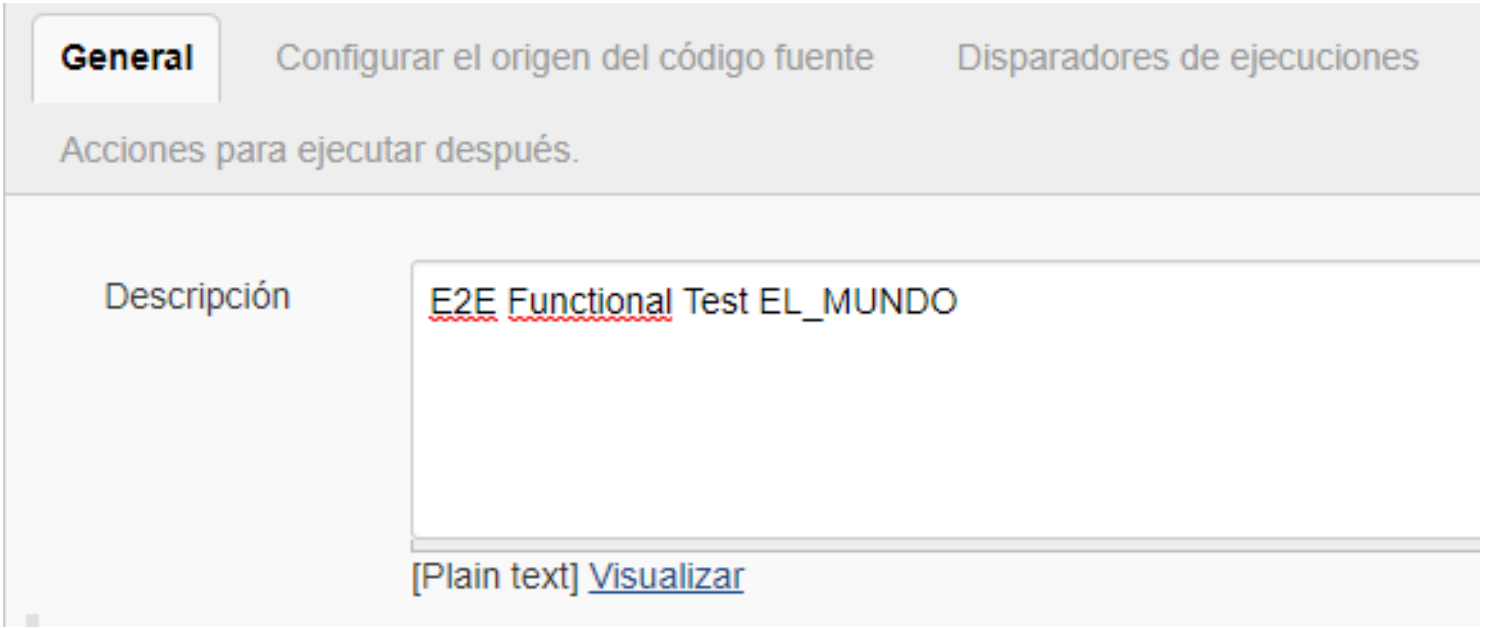
9 – Crear y Configurar un nuevo Job para nuestro proyecto de pruebas funcionales.

- Crear un nuevo Job como ***Proyecto Maven***.



9.1 – Crear y Configurar un nuevo ***Job*** para nuestro proyecto de pruebas funcionales.

- Configurar nuestro ***Job*** añadiendo una descripción.

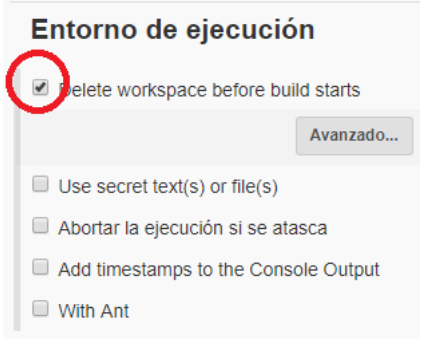


9.2 – Crear y Configurar un nuevo ***Job*** para nuestro proyecto de pruebas funcionales.

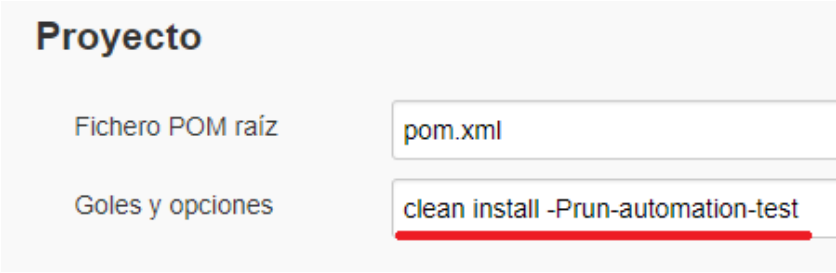
- Configurar el origen del código fuente y añadimos los credenciales de nuestro repositorio.

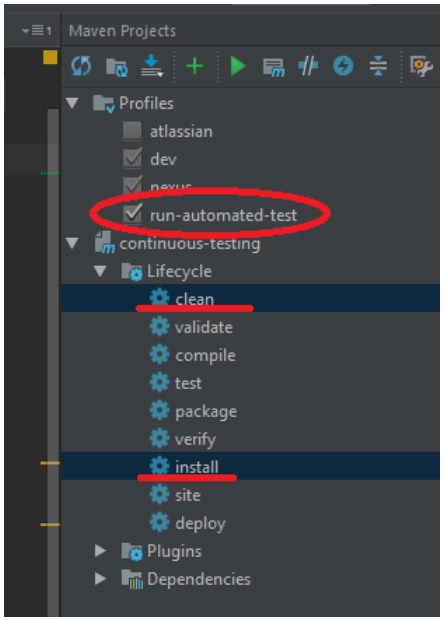
9.3 – Crear y Configurar un nuevo ***Job*** para nuestro proyecto de pruebas funcionales.

- Habilitar la opción en ***Entorno de Ejecución*** para limpiar el ***Workspace*** antes de cada ejecución.

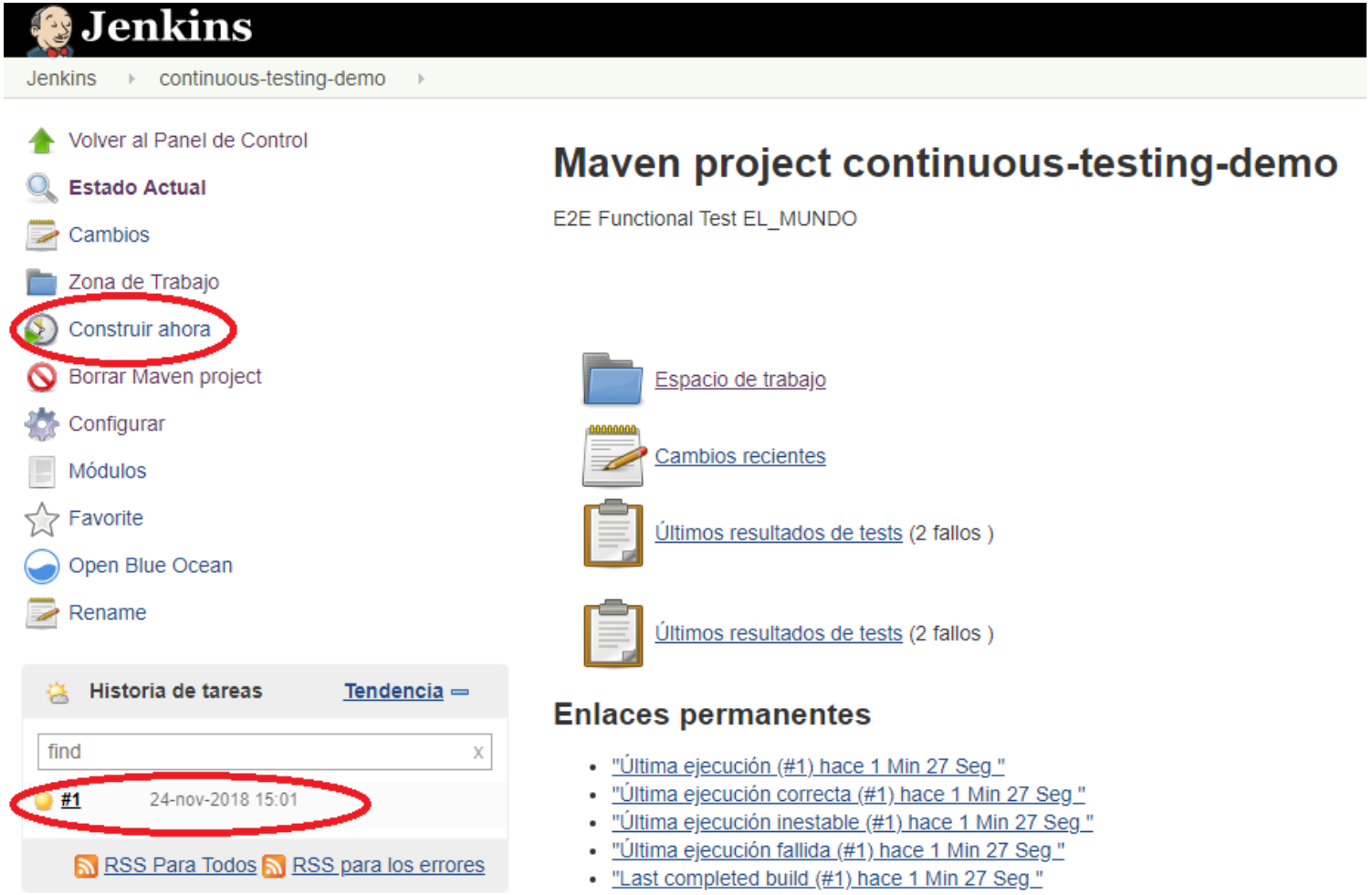


- Definir los ***Goles y Opciones*** del proyecto.





10 – Guardar la nueva configuración y ejecutar la construcción del proyecto a través de nuestro ***Job***.



Practica 4

Solucionar problemas en el entorno

Practica 4 - Solucionar problemas en el entorno

1 – Nuestra batería de pruebas necesita un Chrome previamente instalado en el entorno para poder ejecutar las pruebas.

- Sobrescribir la imagen base de ***jenkins/jenkins*** mediante un ***Dockerfile*** para crear una con un Chrome instalado.

```
FROM jenkins/Jenkins

USER root

RUN cd /tmp
RUN apt-get update
RUN apt-get install -y libxss1 libappindicator1 libindicator7
RUN wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
RUN apt install -y ./google-chrome*.deb
```

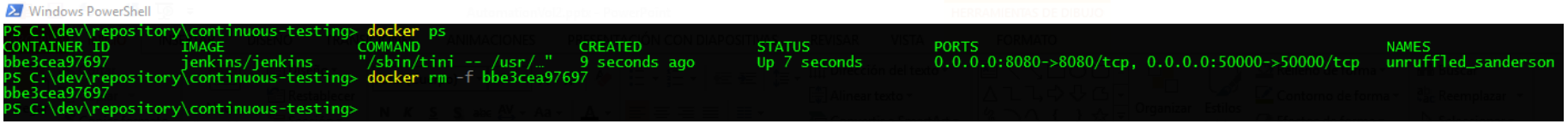
- Construir nuestra versión de ***Jenkins CI*** modificada.
  - ***docker build –t {usuario_docker_hub}/jenkins .*** *(ejem: franciscoaznar/jenkins)*
  - ***docker push {usuario_docker_hub}/jenkins:latest***

Practica 4 - Solucionar problemas en el entorno

2 – Eliminar el contenedor de Jenkins CI con la versión sin modificar.

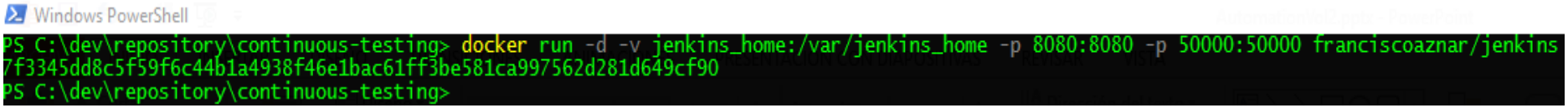
- Borrar el contenedor activo de Jenkins CI.

- docker rm –f {ID_Contenedor}



3 – Levantar de nuevo el contenedor con nuestra imagen modificada de Jenkins CI.

- docker run -d -v jenkins_home:/var/jenkins_home -p 8080:8080 -p 50000:50000 {usuario_docker-hub}/jenkins



everis
an NTT DATA Company

Practica 5

Integrar Cucumber Report

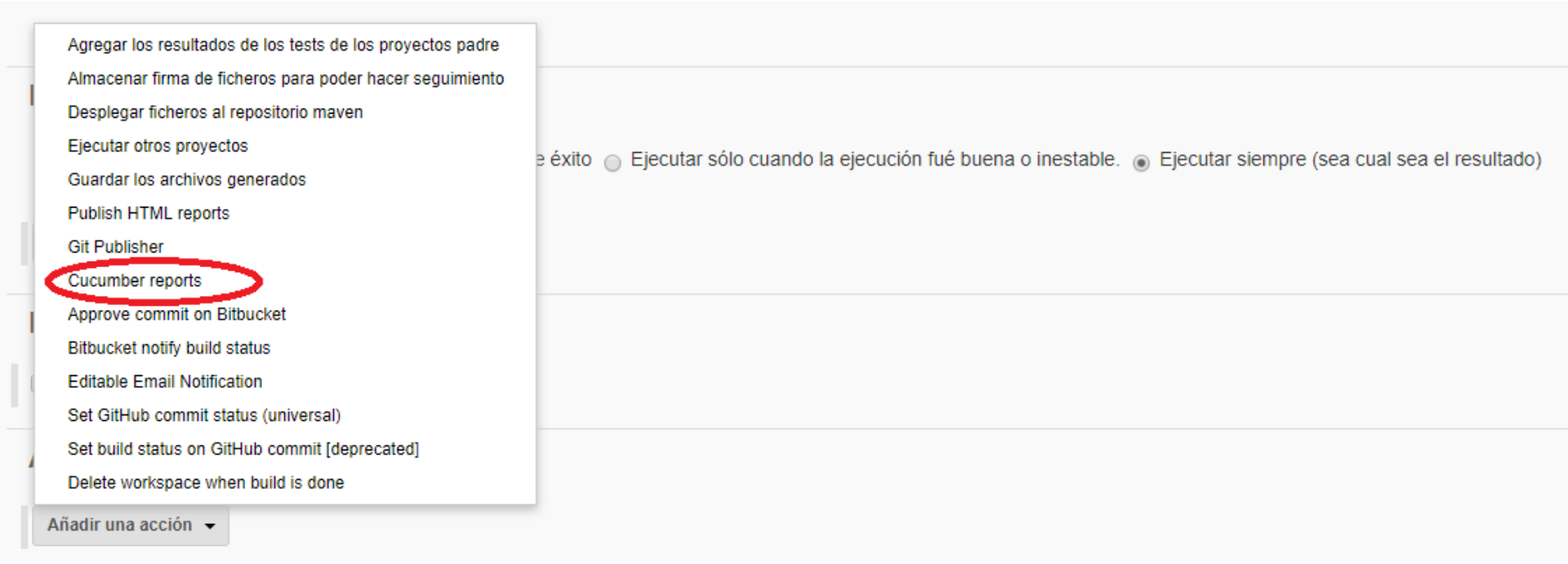
Capitulo 3

Practica 5 – Integrar Cucumber Report

1 – Descargar el plugin para el reporte de ***Cucumber*** de Jenkins CI.

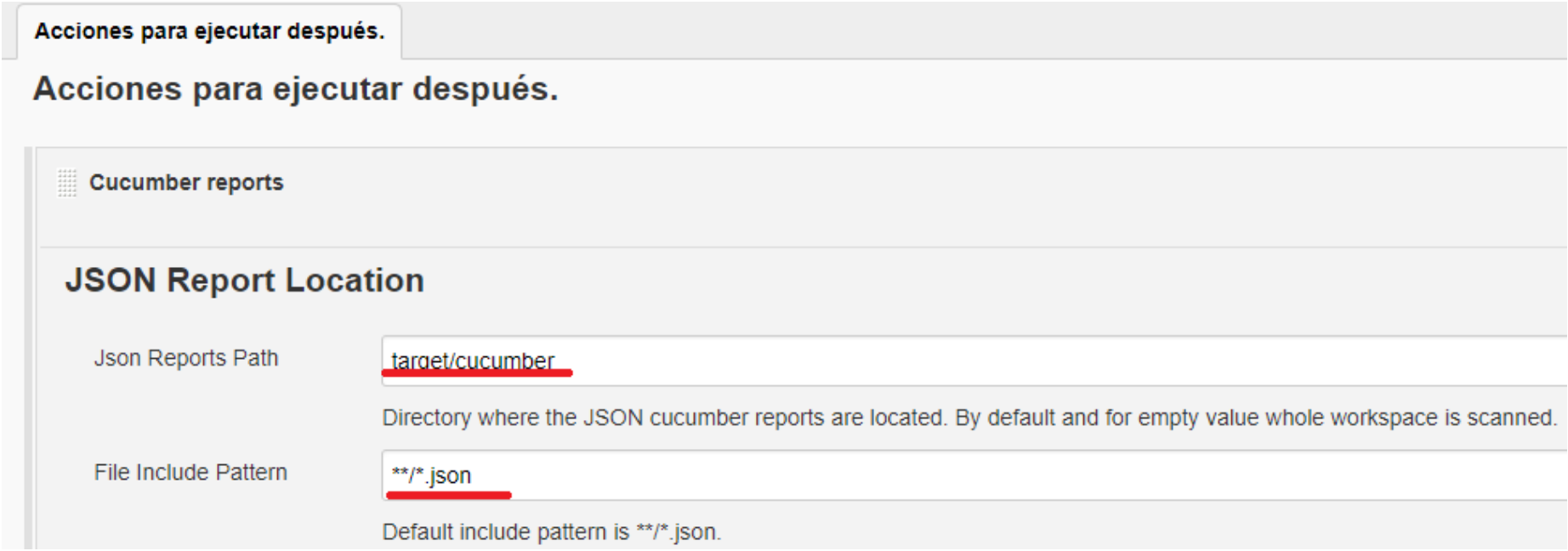
- Descargamos e instalamos el plugin de ***Cucumber Report*** como ya hemos visto previamente.

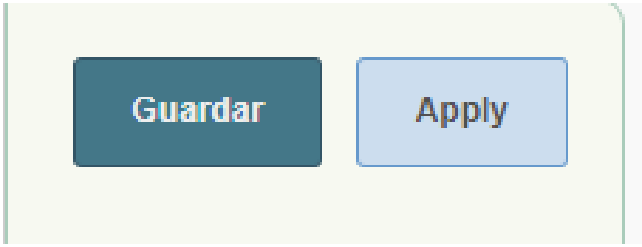
2 – Habilitamos y Configuramos desde nuestro Job las opciones de ***Cucumber Report*** en **Acciones para ejecutar después.**



Practica 5 – Integrar Cucumber Report

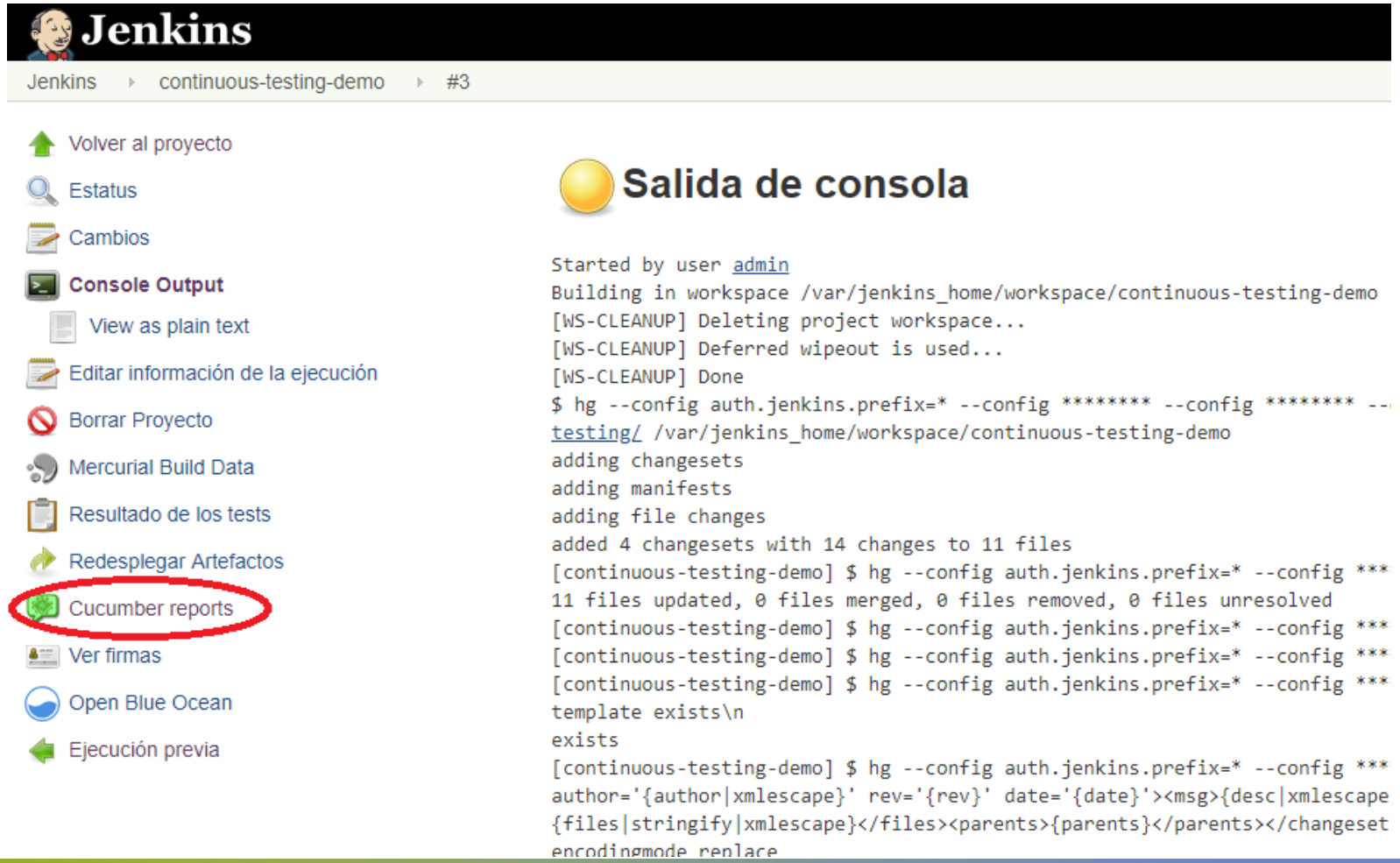
3 – Configurar plugin en base al reporte que generara nuestro proyecto de pruebas funcionales y aplicamos cambios y guardamos la configuración.





Practica 5 – Integrar Cucumber Report

4 – Ejecutamos de nuevo el ***Job*** y entramos en el panel de construcción, si todo ha ido bien, veremos el icono de ***Cucumber Report.***



Practica 5 – Integrar Cucumber Report

5 – Al terminar la ejecución del ***Job***, ***Cucumber Report*** nos mostrara el reporte de las pruebas funcionales.

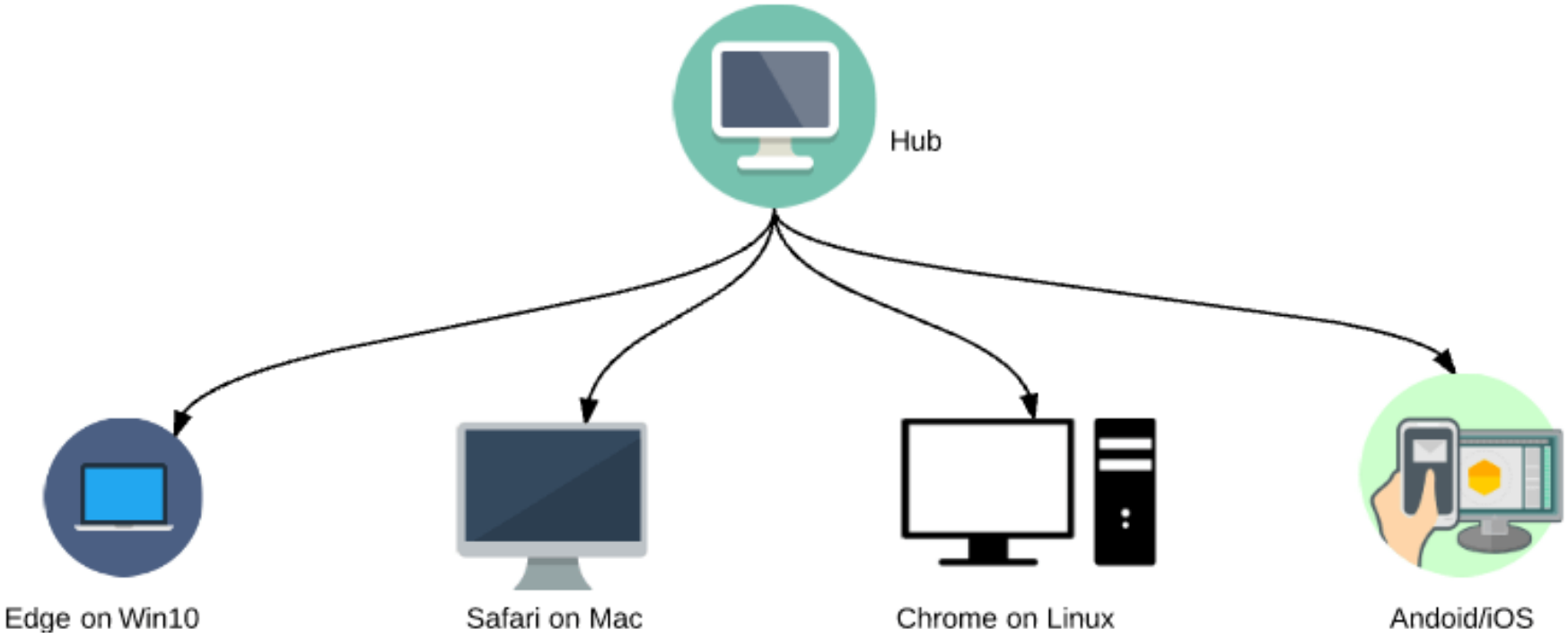
everis
an NTT DATA Company

Capitulo 4

Selenium Grid con Docker Compose

Que es Selenium Grid?

Selenium Grid es una herramienta para lanzar test de *Selenium* de forma distribuida. Utiliza internamente *Selenium Remote Control* para ejecutar los test de integración con distintos navegadores y plataformas.



everis
an NTT DATA Company

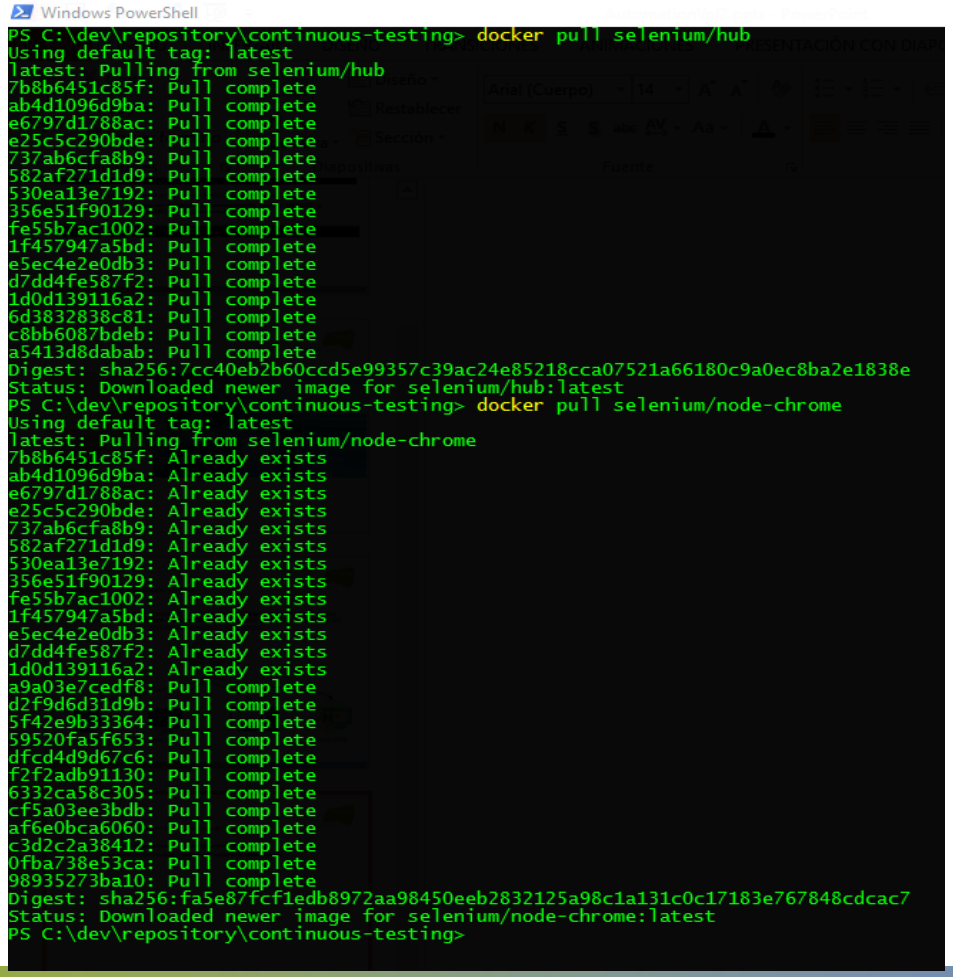
Practica 6

Montar Selenium Grid con Docker

Practica 6 – Montar Selenium Grid con Docker

1 – Descargar imágenes de ***selenium/hub*** y ***selenium/node-chrome.***

- ***docker pull selenium/hub*** y ***docker pull selenium/node-chrome.***



Practica 6 – Montar Selenium Grid con Docker

2 – Levantar los contenedores del ***Selenium Hub*** y el ***Nodo de Chrome.***

- Creamos un red lógica con el nombre ***"grid"*** para conectar los diferentes contenedores en un solo dominio.

- docker network create grid

```
Windows PowerShell
PS C:\dev\repository\continuous-testing> docker network create grid2
dbb444bcd35ac274cc8281c5c656ddcfe3431b6307ab51d7b2ed8de21c4a6784
PS C:\dev\repository\continuous-testing>
```

- Levantamos el primer contenedor que será el ***Hub***.

```
Windows PowerShell
PS C:\dev\repository\continuous-testing> docker run -d -p 4444:4444 --net grid --name hub selenium/hub:latest
82f3d14cebb5ef5ee64fa1f385b1dc8ea161988fcb21263ddc1ec17c3612a244
PS C:\dev\repository\continuous-testing>
```

- Después procedemos con levantar el prime ***Nodo de Chrome***.

- docker run -d --net grid -e HUB_HOST=hub -v /dev/shm:/dev/shm selenium/node-chrome:latest

```
Seleccionar Windows PowerShell
PS C:\dev\repository\continuous-testing> docker run -d --net grid -e HUB_HOST=hub -v /dev/shm:/dev/shm selenium/node-chrome:latest
928a3e388835b3012e31043dcf9106bb92e643fd1818c1e9560d70a96c5e76da
PS C:\dev\repository\continuous-testing>
```

Practica 6 – Montar Selenium Grid con Docker

2.1 – Levantar los contenedores del ***Selenium Hub*** y el ***Nodo de Chrome.***

- Si todo ha ido bien, podremos acceder al ***Hub -> http://localhost:4444/grid/console***

Combinar Jenkins + Selenium Grid con Docker Compose.

everis
an NTT DATA Company

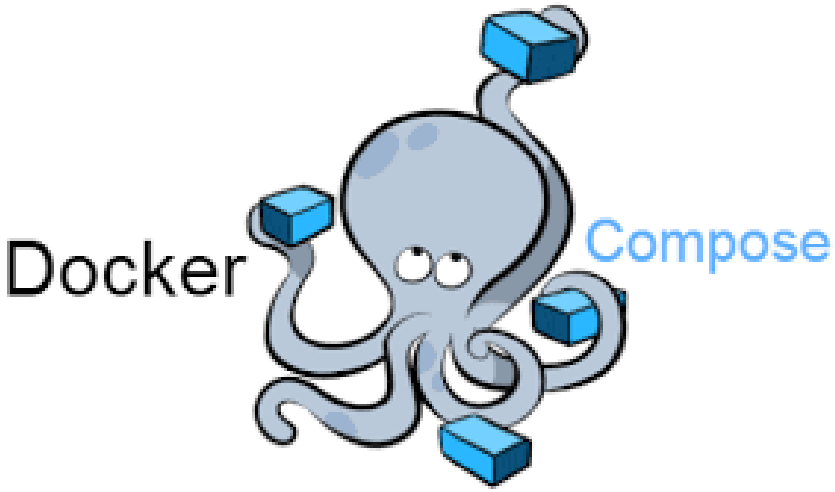
Practica 7

Jenkins + Selenium Grid con Docker Compose

everis
an NTT DATA Company

Que es Docker Compose?

Docker Compose es una herramienta creada por ***Docker*** que nos brinda la opción de levantar e intercomunicar varios contenedores al mismo tiempo, puesto que muchos servicios dependen de otros como Bases de datos, caches centralizadas etc.



Practica 7 - Docker Compose como aliado

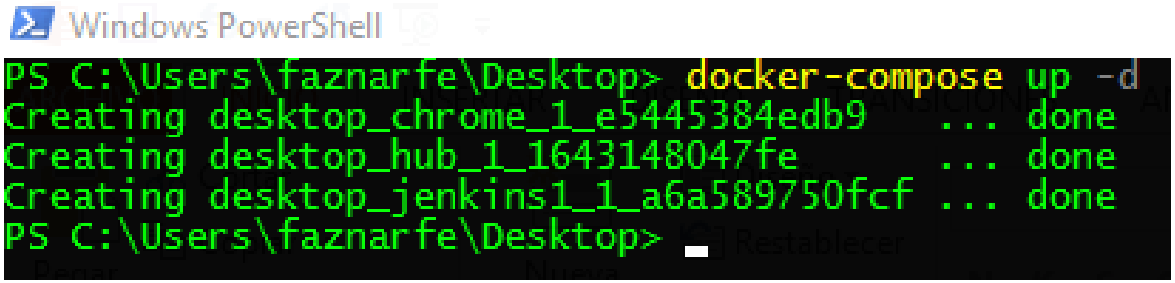
Docker Compose nos permite automatizar el levantamiento de varios contenedores mediante un fichero llamado ***docker-compose.xml***

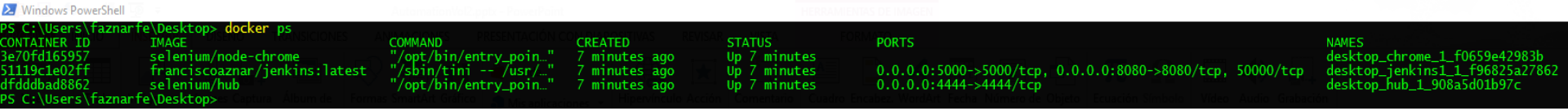
docker-compose.yml

```yaml
version: "3"

services:

  hub:
    image: selenium/hub
    ports:
      - 4444:4444

  chrome:
    image: selenium/node-chrome
    environment:
      HUB_PORT_4444_TCP_ADDR: hub
      HUB_PORT_4444_TCP_PORT: 4444

  jenkins:
    image: franciscoaznar/jenkins:latest
    ports:
      - 8080:8080
      - 5000:5000
    volumes:
      - jenkins_home:/var/jenkins_home


volumes:
    jenkins_home:

networks:
  default:
    driver: bridge
```

Practica 7 - Docker Compose como aliado

Ejecutando el comando ***docker-compose up –d*** en el directorio donde se encuentra el fichero de configuración ***docker-compose.yml,*** levantaremos nuestros ***Jenkins CI***, y el ***Selenium-Grid*** con el ***nodo de Chrome***.





Fin

Preguntas