

Connecting Github with Jenkins and Dockers

Erick Montes Bedolla

Docker y Jenkins ▾

Docker es una plataforma que facilita la creación y diseño de contenedores, envío de imágenes y creación de versión de la imagen.

Los contenedores incluyen todo lo necesario para que un software se ejecute. Se usan como máquinas virtuales pero suelen utilizarse para levantar máquinas independientes con sistemas operativos muy ligeros.

Los contenedores virtualizan el sistema operativo de un servidor. Docker se instala en cada servidor y proporciona comandos sencillos que puede utilizar para crear, iniciar o detener contenedores.

Jenkins por su parte es un servidor cuyo uso es compilar y probar proyectos de software de una forma continua, facilitando el integrar cambios de un proyecto e ir entregando al usuario nuevas versiones. Útil en el proceso de Continuous Integration.

Proceso de integración con Github ▾

Después de instalar tanto Dockers como Jenkins en la computadora, se elaboró un programa simple que se subió a Github en un repositorio --> <https://github.com/ErickMontesDK/jenkins-docker>.

Crear contenedor ▾

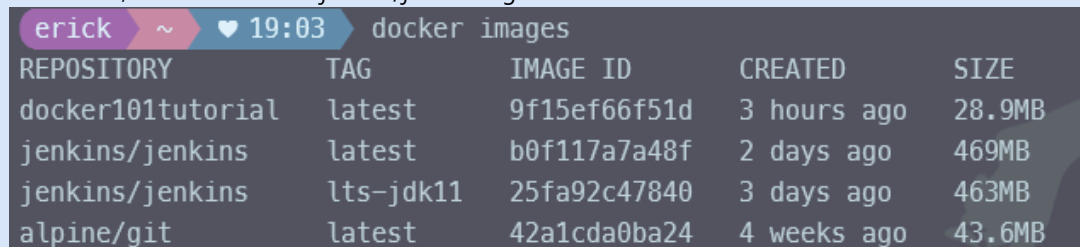
Para conectar Jenkins a docker, instalamos la imagen de jenkins en docker con el cmd

```
docker pull jenkins/jenkins
```

Una vez instalada, podemos ver nombre del contenedor con el comando

```
docker images
```

En mi caso, es con el nombre jenkins/jenkins tag=latest



REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker101tutorial	latest	9f15ef66f51d	3 hours ago	28.9MB
jenkins/jenkins	latest	b0f117a7a48f	2 days ago	469MB
jenkins/jenkins	lts-jdk11	25fa92c47840	3 days ago	463MB
alpine/git	latest	42a1cda0ba24	4 weeks ago	43.6MB

Para ejecutar el contenedor, se escribe el comando:

```
docker run -p 8080:8080 -p 5000:5000 -d -v jenkins_home:/var/jenkins_home jenkins/jenkins:latest'
```

Si se ejecutó con éxito, nos podemos conectar al localhost de Jenkins :8080, donde pide una contraseña para ingresar. La contraseña se obtiene con el siguiente código

Obtiene el id de nuestro contenedor

```
docker ps
```

Ingresamos al docker shell. Solo me funcionó el windows powershell, no trabajaba en git bash

```
docker exec -i -t <container_id> /bin/bash
```

Nos da la clave de ingreso

```
jenkins@<container_id>:/$ cat /var/jenkins_home/secrets/initialAdminPassword
```

```
PS C:\Users\erick> docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED        STATUS        PORTS
e9a634d5b4ea   jenkins/jenkins:latest    "/usr/bin/tini -- /u..." 18 minutes ago Up 18 minutes 0.0.0.0:5000->5000/tcp, 0.0.0.0:8080->8080/tcp, 50000/tcp
serene_panini
PS C:\Users\erick> docker exec -i -t e9a634d5b4ea /bin/bash
jenkins@e9a634d5b4ea:/$ cat /var/jenkins_home/secrets/initialAdminPassword
ba16f003b05246628f25826cf7e94b5d
```

 Conectar con el repositorio 

Ya dentro de localhost:8080 en la interfaz de Jenkins, creamos nueva tarea, donde asignamos un nombre y que el proyecto sea de **Estilo libre**

Enter an item name

» Required field



Crear un proyecto de estilo libre

Esta es la característica principal de Jenkins, la de ejecutar el proyecto combinando cualquier tipo de repositorio de software (SCM) con cualquier modo de construcción o ejecución (make, ant, mvn, rake, script ...). Por tanto se podrá tanto compilar y empaquetar software, como ejecutar cualquier proceso que requiera monitorización.



Pipeline

En la configuración general, se selecciona la opción de Github y escribimos el url del repositorio a conectar. Igualmente en la sección de configurar el origen del código fuente, se selecciona Git, se escribe el url del repositorio y se selecciona la branch con la que se va

a trabajar.

Panel de Control > test-docker-github > Configuration

Configure

General

⚙️ General

🔑 Configurar el origen del código fuente

🕒 Disparadores de ejecuciones

🌐 Entorno de ejecución

☰ Build Steps

📦 Acciones para ejecutar después.

Descripción

[Plain text] [Visualizar](#)

☐ Desechar ejecuciones antiguas ?

☐ Esta ejecución debe parametrizarse ?

☒ GitHub project

Project url ?

<https://github.com/ErickMontesDK/jenkins-docker/>

Avanzado...

Configurar el origen del código fuente

☐ Ninguno

☒ Git ?

Repositories ?

Repository URL ?

<https://github.com/ErickMontesDK/jenkins-docker>

Credentials ?

- none -

De esta forma ya esta conectado

☰ Configurar ejecución ▾

Ya que queremos que cada que hagamos un cambio al repositorio de Github ejecute el código para verificar si el código esta bien, otra vez en la sección de configuracion hay que ir a la sección de Ejecutar.

Aquí configuramos los comandos que queremos que Jenkins haga en nuestro proyecto, en mi caso como es un archivo sh selecciono la opción de Ejecutar linea de comando y escribo el código que escribiría en el cmd

```
chmod +x helloWorld.sh
./helloWorld.sh
```

Build Steps

☰ Ejecutar línea de comandos (shell) ?

Comando

Visualizar [la lista de variables de entorno disponibles](#)

```
chmod +x helloWorld.sh
./helloWorld.sh
```

☰ Revisión de resultados ▾

Una vez configurado, seleccionamos **Construir ahora**, haciendo que el programa se ejecute. Dentro podemos ver el console.ouput, que nos muestra lo que nos generó nuestro código y vemos que se ejecutó sin problema

Estatus

</> Cambios

Console Output

View as plain text

Editar información de la ejecución

Delete build '#3'

Git Build Data

Ejecución previa

✓ Salida de consola

Started by user [Erick Montes](#)
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/test-docker-github
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/jenkins_home/workspace/test-docker-github/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url <https://github.com/ErickMontesDK/jenkins-docker> # timeout=10
Fetching upstream changes from <https://github.com/ErickMontesDK/jenkins-docker>
> git --version # timeout=10
> git --version # 'git version 2.30.2'
> git fetch --tags --force --progress -- <https://github.com/ErickMontesDK/jenkins-docker> +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 49cf723daeea3e365045a7ec5b0b3012e7619e73 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 49cf723daeea3e365045a7ec5b0b3012e7619e73 # timeout=10
Commit message: "Add files via upload"
First time build. Skipping changelog.
[test-docker-github] \$ /bin/sh -xe /tmp/jenkins10872381558986699875.sh
+ chmod +x helloWorld.sh
+ ./helloWorld.sh
Hello World
Finished: SUCCESS