Laboratorio: Compilar y testear una aplicación con Jenkins

Parte 1: Confirmar la aplicación a Git

Parte 2: Descargar y ejecutar la imagen de Jenkins Docker

Parte 3: Configurar Jenkins

Parte 4: Usar Jenkins para ejecutar una compilación de su aplicación

Parte 5: Usar Jenkins para probar una compilación

Aspectos básicos/Situación

En este laboratorio confirmará el código de la aplicación de en un repositorio de GitHub, modificará el código localmente y luego confirmará los cambios. Asimismo instalará un contenedor Docker que incluye la última versión de Jenkins. Configure Jenkins y luego lo utilizará para descargar y ejecutar su programa de aplicación de muestra. Además, se modificará el trabajo creado para verificar que el programa de muestra se ejecuta correctamente cada vez que lo compila.

Recursos necesarios

1. Computadora con so linux preferentemente ububtu.
2. VS Code u otro editor de su agrado
3. Docker instalado

Instrucciones

Parte 1 - Confirmar la aplicación a Git

En esta parte, creará un repositorio en GitHub para confirmar los archivos de aplicación de muestra que creó en un laboratorio anterior. Se creó una cuenta de GitHub en un laboratorio anterior. Si aún no lo ha hecho, visite github.com ahora y cree una cuenta.

Paso 1: Iniciar sesión en GitHub y crear un nuevo repositorio.

a. Iniciar sesión en **https://github.com/** con sus credenciales.
b. Seleccionar el botón "**New**" o haga click en el ícono "**+**" en la esquina superior derecha y seleccione "**New repository**".
c. Crear un respositorio “**Private**” utilizando la siguiente información:
d. Nombre del repositorio : **calculadora**
e. Descripción: **TP SdeC, explorando CI/CD con GitHub y Jenkins**.

Paso 2: Configurar sus credenciales de Git localmente (en su computadora).

Suponiendo que está ejecutando **VS Code** y abrió la carpeta donde tiene su proyecto, abrir una ventana de terminal con **VS Code**. Luego, utilizar su nombre en lugar de "**Usuario de ejemplo**" para el nombre entre comillas dobles. Utilizar **sample@example.com** para su dirección de correo electrónico.

```
javier@javier-ThinkPad-T450: ~$ git config –global user.name «Usuario de ejemplo»

javier@javier-ThinkPad-T450: ~$ git config –global user.email sample@example.com
```

Laboratorio: Compilar y testear una aplicación con Jenkins

Paso 3: Inicializar un directorio como repositorio local de Git.

Descargar los archivos de esta url https://codeload.github.com/javierajorge/calculator/zip/master
Extraer y copiar estos archivos en el directorio ~**/jenkins/calculadora**. Navegar al directorio **jenkins/calculadora** e inicialice como un repositorio de Git.

```
javier@javier-ThinkPad-T450: ~$ cd jenkins/calculadora
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$ git init
```

Paso 4: Apuntar el repositorio local de Git al repositorio en GitHub.

Utilizar el comando **git remote add** para agregar una URL de Git con un alias remoto de «origen» y apunte al repositorio recién creado a GitHub. Usando la URL del repositorio de Git que creó en el paso 1, solo debe reemplazar el **github-username** en el siguiente comando con su nombre de usuario de GitHub.

Nota: Su nombre de usuario de GitHub distingue entre mayúsculas y minúsculas.

```
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$ git remote add origin https://github.com/github-username/calculadora.git
```

Paso 5: Almacenar, confirmar e insertar los archivos de aplicación de muestra en el repositorio de GitHub.

a. Use el comando **git add** para almacenar los archivos en el directorio **jenkins/calculadora**. Utilice el argumento asterisco (*) para almacenar todos los archivos en el directorio actual.

```
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$ git add *
```

b. Utilizar el comando **git status** para ver los archivos y directorios que están almacenados y listos para ser confirmados en su repositorio de GitHub.

```
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$ git status
```

c. Utilizar el comando **git commit** para confirmar los archivos almacenados y comenzar a realizar un seguimiento de los cambios. Agregar un mensaje de su elección o utilizar el que se proporciona aquí.

```
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$ git commit -m "Confirmando archivos de aplicación de ejemplo."
```

d. Usar el comando **git push** para enviar sus archivos locales de aplicación de muestra a su repositorio de GitHub.

```
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$ git push origin master
Nombre de usuario para 'https://github.com': username
Contraseña para 'https://AllJohns@github.com': contraseña
Enumeración de ...
```

Nota: Si en lugar de una solicitud de su nombre de usuario, recibe un mensaje de VS Code con el mensaje, La extensión 'Git' quiere iniciar sesión usando GitHub, entonces configuró mal sus credenciales de GitHub en el paso 2 y/o el URL de GitHub en el paso 4. La dirección URL debe tener el nombre de usuario correcto que distingue entre mayúsculas y minúsculas y el nombre del repositorio que creó en el paso 1. Para revertir su comando anterior git add, use el comando **git remote rm origin**. A continuación, vuelva al paso 2 asegurándose de introducir las credenciales correctas y, en el paso 4, introduzca la dirección URL correcta.

Laboratorio: Compilar y testear una aplicación con Jenkins

> **Note**: Si después de introducir el nombre de usuario y la contraseña, se obtiene un mensaje de error fatal (fatal error) indicando "repositorio no encontrado (repository not found)", probablemente una URL incorrecta fue enviada. Necesitará revertir su comando **git add** con el comando **git remote rm origin.**

Parte 2: Descargar y ejecutar la imagen de Jenkins en Docker

En esta parte, descargará una imagen de Jenkins en Docker. A continuación, inicie una instancia de la imagen y verifique que el servidor Jenkins se esté ejecutando.

Paso 1: Descargar la imagen de Jenkins Docker

Nota: Si aún no tiene instalado docker, puede instalarlo con el siguiente comando:

```
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$ docker apt intall
docker.io
```

Puede encontrar la imagen de Jenkins Docker aquí: **https://hub.docker.com/r/jenkins/jenkins**. En el momento de escribir este laboratorio, ese sitio especifica usar el comando **docker pull jenkins/jenkins** para descargar el último contenedor Jenkins. Se debería obtener una salida similar a la siguiente:

```
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$ docker pull
jenkins/jenkins:lts
Está extrayendo de jenkins/jenkins
3192219afd04: Extrayendo de la capa fs
17c160265e75: Extrayendo de la capa fs
cc4fe40d0e61: Extrayendo de la capa fs
9d647f502a07: Extrayendo de la capa fs
d108b8c498aa: Extrayendo de la capa fs
1bfe918b8aa5: Extracción completa
dafa1a7c0751:Extracción completa
…
Estado: Imagen más nueva descargada para jenkins/jenkins:lts
docker.io/jenkins/jenkins:lts
javier@javier-ThinkPad-T450: ~/jenkins/sample-app$
```

Paso 2: Arrancar el contenedor de Jenkins Docker.

Introduzca el siguiente comando en **una línea**. Es posible que tenga que copiarlo en un editor de texto si está viendo una versión PDF de este laboratorio para evitar saltos de línea. Este comando iniciará el contenedor de Docker con Jenkins y luego permitirá que los comandos de Docker se ejecuten dentro de su servidor Jenkins.

```
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$ docker run --rm -u root -p
8080:8080 -v jenkins-data:/var/jenkins_home -v $(which docker):/usr/bin/docker -v
/var/run/docker.sock:/var/run/docker.sock -v "$HOME":/home --name jenkins_server
jenkins/jenkins:lts
```

Las opciones utilizadas en este comando de **docker run** son las siguientes:

--rm : esta opción elimina automáticamente el contenedor Docker cuando se detiene.

-u : Esta opción especifica el usuario. Desea que este contenedor Docker se ejecute como raíz (root) para que se permitan todos los comandos Docker introducidos dentro del servidor Jenkins.

-p : Esta opción especifica el puerto en el que se ejecutará el servidor Jenkins localmente.

Laboratorio: Compilar y testear una aplicación con Jenkins

-v : Estas opciones unen los volúmenes de montaje necesarios para Jenkins y Docker. El primer **-v** especifica dónde se almacenarán los datos de Jenkins. El segundo **-v** especifica dónde ubicar Docker para que pueda ejecutarse dentro del contenedor Docker que está ejecutando el servidor Jenkins. El tercer **-v** especifica la variable PATH para el directorio principal.

Paso 3: Verificar que el servidor Jenkins se esté ejecutando.

El servidor Jenkins debería estar ahora en ejecución. Copiar la contraseña de administrador que aparece en la salida, como se muestra en la siguiente ventana.

No introduzca ningún comando en esta ventana del servidor. Si detiene accidentalmente el servidor Jenkins, deberá volver a ingresar el comando **docker run** del paso 2 anterior. Después de la instalación inicial, la contraseña de administrador se muestra como a continuación.

```
…

*************************************************************
*************************************************************
*************************************************************


Se requiere la configuración inicial de Jenkins. Se ha creado un usuario
administrador y se ha generado una contraseña.

Utilice la siguiente contraseña para proceder a la instalación:


77dc402e31324c1b917f230af7bfebf2 <—Su contraseña será diferente


Esto también se puede encontrar en: /var/jenkins_home/secrets/initialAdminPassword

*************************************************************
*************************************************************
*************************************************************
<output omitted>
2020-05-12 16:34:29 .608+0000 [id=19] INFO Hudson.webappMain$3 #run: Jenkins is
fully up and running
```

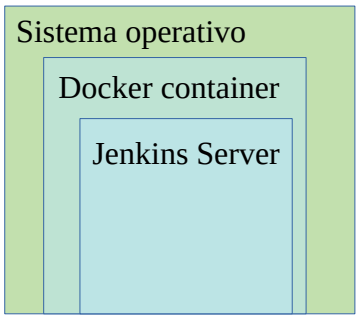
Nota: Si pierde la contraseña, no se muestra como se muestra arriba, o necesita reiniciar el servidor Jenkins, siempre puede recuperar la contraseña accediendo a la línea de comandos del contenedor Jenkins Docker. Cree una segunda ventana de terminal en VS Code e ingrese los siguientes comandos para que no detenga el servidor Jenkins:

```
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$ docker exec -it
jenkins_server /bin/bash
root@19d2a847a54e: /# cat /var/jenkins_home/secrets/initialAdminPassword
77dc402e31324c1b917f230af7bfebf2
root @19d2a847a54e: /# exit
exit
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$
```

Nota: El ID del contenedor (19d2a847a54e resaltado anteriormente) y la contraseña serán diferentes.

Laboratorio: Compilar y testear una aplicación con Jenkins

Paso 4: Investigar los niveles de abstracción que se están ejecutando actualmente en el equipo.



Parte 3: Configurar Jenkins

En esta parte, se completará la configuración inicial del servidor Jenkins.

Paso 1: Abrir una pestaña del navegador.

Vaya a http://localhost:8080/ e inicie sesión con su contraseña de administrador copiada.

Paso 2: Instalar los complementos de Jenkins recomendados.

Haga clic en **Install suggested plugins** y espere a que Jenkins descargue e instale los complementos. En la ventana del terminal, verá mensajes de registro a medida que la instalación continúe. Asegúrese de no cerrar esta ventana de terminal. Puede abrir otra ventana de terminal para acceder a la línea de comandos.

Paso 3: Omitir la creación de un nuevo usuario administrador.

Una vez finalizada la instalación, se le mostrará la ventana **Create First Admin User**. Por ahora, haga click en **Skip and continue as admin** en la parte inferior.

Paso 4: Omitir la creación de una instancia.

En la ventana **Instance Configuration**, no cambie nada. Haga clic en **Save and Finish** en la parte inferior.

Paso 5: Empezar a usar Jenkins.

En la siguiente ventana, haga clic en **Start using Jenkins**. Ahora debería estar en el panel principal con un mensaje de **Welcome to Jenkins!** .

Parte 4: Usar Jenkins para ejecutar una compilación de la aplicación

La unidad fundamental de Jenkins es el **job** (también conocido como un **project**). Puede crear **jobs** que realicen una variedad de tareas, entre las que se incluyen las siguientes:

- Recuperar código de un repositorio de administración de código fuente como GitHub.
- Compilar una aplicación utilizando una secuencia de comandos (**script**) o una herramienta de compilación.
- Empaquetar una aplicación y ejecutarla en un servidor

En esta parte, creará un **job** de Jenkins simple que recupera la última versión de su aplicación de muestra de GitHub y ejecuta el script de compilación. En Jenkins, puede probar su aplicación (**Parte 5**) y agregarla a una **pipeline** de desarrollo (Parte 8).

Laboratorio: Compilar y testear una aplicación con Jenkins

Paso 1: Crear un nuevo job.

a. Haga click en el enlace **Create a job** directamente debajo del mensaje: **¡Welcome to Jenkins!** También puede hacer clic en **New Item** en el menú de la izquierda.
b. En el campo **Enter an Item Name** ponga como nombre **BuildAppJob**.
c. Haga click en **Freestyle Project** como tipo de trabajo. En la descripción, la abreviatura de ACS significa administración de configuración de software (Software Configuration Management, SCM), que es una clasificación de software responsable del seguimiento y control de los cambios en el software.
d. Arrástrelo hasta la parte inferior de la página y haga click en **OK**.

Paso 2: Configurar el Jenkins BuildAppJob.

Ahora se encuentra en la ventana de configuración donde se pueden introducir detalles sobre su trabajo. Las pestañas en la parte superior son accesos directos a las secciones siguientes. Hacer clic en las pestañas para explorar las opciones que se pueden configurar. Para este trabajo, sólo se necesita agregar algunos detalles de configuración.

a. Hacer clic en la pestaña **General** y agregar una descripción para su trabajo. Por ejemplo, "**Mi primer trabajo en Jenkins.**"
b. Hacer clic en la pestaña **Administración de código fuente (Source Code Management)** y elegir el botón de opción **Git**. En el campo URL del repositorio, agregue el enlace del repositorio de GitHub para la aplicación de muestra, teniendo en cuenta que su nombre de usuario distingue mayúsculas de minúsculas. Asegúrese de agregar la extensión .git al final de su URL.

   Por ejemplo:

   `https://github.com/`*`github-username`*`/calculadora.git`

c. Para **Credenciales (Credentials)**, haga clic en el botón **Agregar (Add)** y elija **Jenkins**.
d. En el cuadro de diálogo **Agregar credenciales (Add Credentials)**, rellene con su nombre de usuario y contraseña de GitHub y, a continuación, haga clic en **Agregar (Add)**.

   Nota: Recibirá un mensaje de error que indica que la conexión ha fallado. Esto se debe a que aún no ha seleccionado las credenciales.

e. En el menú desplegable de **Credenciales (Credentials)** donde actualmente dice **Ninguno (None)**, elija las credenciales que acaba de configurar.
f. Después de agregar la URL y las credenciales correctas, Jenkins prueba el acceso al repositorio. No debería haber mensajes de error. Si los hay, verificar la URL y las credenciales. Tendrá que **añadirlos** de nuevo, ya que en este momento no hay forma de eliminar los que se han introducido anteriormente.
g. En la parte superior de la ventana de configuración de **BuildAppJob**, haga clic en la pestaña **Build**.
h. Abra menú desplegable **Add build step** y elija **Execute shell**.
i. En el campo **Command**, escriba el comando que utiliza para ejecutar la compilación make.

   `make`

j. Haga clic en el botón **Save**. Será regresado al panel de Jenkins con **BuildAppJob** seleccionado.

Nota: Como el contenedor no tiene acceso a las aplicaciones del host, es necesario actualizar la lista de paquetes disponibles y luego instalar make y gcc. La forma mas sensilla en sistemas debian es instalar el pakete build-essential.

k. Primero debe acceder al docker mediante una consola para poder instalar los paketes.

Laboratorio: Compilar y testear una aplicación con Jenkins

```
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$ docker exec -it
jenkins_server /bin/bash
```

l. Luego instalar los paquetes necesarios

```
root@f2c9059c2cea:/# apt update && apt install build-essential
root@f2c9059c2cea:/# apt clean
root@f2c9059c2cea:/# rm -rf /var/lib/apt/lists/*
```

m. Puede salir con **exit**

n. Para que la proxima vez que intente levantar la instancia estos cambios sean permantentes se puede crear una nueva instancia de este docker con los cambios realizados usando el comando commit.

```
javier@javier-ThinkPad-T450:~$ docker ps -l
CONTAINER ID        IMAGE                 COMMAND
CREATED             STATUS              PORTS
NAMES
918dcdd85e8c        jenkins/jenkins:lts   "/sbin/tini -- /usr/…"   9
← javier@javier-ThinkPad-T450:~$ docker commit 918dcdd85e8c jenkins-build-
essentials
```

p. Detener la imagen original

```
← javier@javier-ThinkPad-T450:~$ docker container stop jenkins_server
```

r. La proxima vez debera correr la nueva imagen...

```
docker run --rm -u root -p 8080:8080 -v
jenkins-data:/var/jenkins_home -v $(which docker):/usr/bin/docker -v
/varun/docker.sock:/var/run/docker.sock -v "$HOME":/home --name
jenkins_server jenkins-build-essentials
```

Paso 3: Utilizar Jenkins para compilar una aplicación.

En la parte izquierda, haga click en **Build Now** para iniciar el trabajo. Jenkins descargará su repositorio de Git y ejecutará el comando de compilación make. Su compilación debería tener éxito porque no ha cambiado nada en el código desde la Parte 3 cuando modificó el código.

Paso 4: Acceder a los detalles de la compilación.

A la izquierda, en la sección **Build History**, haga click en su número de compilación, que debería ser el **#1**, a menos que haya compilado la aplicación varias veces.

Paso 5: Ver la salida de consola.

A la izquierda, haga click en **Console Output**. Debe ver un resultado similar a lo siguiente. Observe los mensajes de éxito en la parte inferior, así como la salida del comando **docker ps -a**. Se están ejecutando al menos un contenedor docker para Jenkins en el puerto local 8080.

```
Started by user admin
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/BuildAppJob
The recommended git tool is: NONE
using credential 66b4a183-1951-49fc-a9a7-e06646aa9f26
Cloning the remote Git repository
Cloning repository https://github.com/MkSolinas/calculadora.git
 > git init /var/jenkins_home/workspace/BuildAppJob # timeout=10
Fetching upstream changes from https://github.com/MkSolinas/calculadora.git
 > git --version # timeout=10
```

Laboratorio: Compilar y testear una aplicación con Jenkins

```
 > git --version # 'git version 2.20.1'
using GIT_ASKPASS to set credentials
 > git fetch --tags --force --progress -- https://github.com/MkSolinas/calculadora.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url https://github.com/MkSolinas/calculadora.git # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision b22a57adb836500f475a3a5ccb55961b671e64c4 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f b22a57adb836500f475a3a5ccb55961b671e64c4 # timeout=10
Commit message: "Confirmando archivos de aplicación ejemplo"
First time build. Skipping changelog.
[BuildAppJob] $ /bin/sh -xe /tmp/jenkins8608235895889754566.sh
+ make
make: 'calc' is up to date.
Finished: SUCCESS
```

Laboratorio: Compilar y testear una aplicación con Jenkins

Parte 5: Usar Jenkins para testear una compilación

En esta parte agregaremos un segundo **job** para testear la aplicación y asi poder asegurarse de que funciona correctamente.

Paso 1: Editar el trabajo para probar su aplicación de muestra.

a. Vuelva a la pestaña del navegador web Jenkins y haga clic en el enlace **Jenkins** en la esquina superior izquierda para volver al panel principal.

b. Haga click en el enlace **del nombre del job** debajo de la etiqueta “name” para acceder al trabajo anterior.

c. Haga click en el enlace “configure” en el panel izquierdo.

Paso 2: Configurar Jenkins para correr una prueba.

a. Haga clic en la pestaña **Build**.

b. Haga clic en **Add build step** y elija **Execute shell**.

c. Introducir el nombre del script de pruebas ya incluido ./test.sh Este escript intenta realizar una serie de operaciones. Si las operaciones devuelven el resultado esperado, el script sale con un código 0 (cero), lo que significa que no hay errores en la compilación en el Job. Si falla algun assert, el script sale con un código de 1 (uno), lo que significa que el Job falló.

d. Guarde los cambios

Paso 4: Hacer que Jenkins ejecute el trabajo BuildAppJob de nuevo.

a. Actualice la página web con el botón de actualizar de su navegador.

b. Haga clic en el botón de compilación en el extremo derecho (un reloj con una flecha).

Paso 5: Verificar que hayan finalizado correctamente

Si todo va bien, deberíamos ver la marca de tiempo para la actualización de la columna **Último éxito** para **BuildAppJob** y **TestAppJob**. Esto significa que el código para ambos trabajos se ejecutó sin errores. Pero también podemos verificar esto por nosotros mismos.

Nota: Si las marcas de hora no se actualizan, asegúrese de activar la actualización automática haciendo clic en el vínculo situado en la esquina superior derecha.

a) Haga clic en el vínculo de la última compilación y, por consiguiente, haga clic en **Salida de consola (Console Uotput)**. Debe ver un resultado similar al siguiente:

```
Iniciado por el proyecto
...
+ exit 0 Terminado:
EXITOSO
```

Paso 6: Detener el contenedor docker

Con el siguiente comando para ver el nombre del container que está corriendo y detenerlo:

```
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$ docker container ls
```

Laboratorio: Compilar y testear una aplicación con Jenkins

```
javier@javier-ThinkPad-T450: ~/jenkins/calculadora$ docker container stop
jenkins_server
```