



Tutorial de extremo a extremo para la integración y entrega continuas por Dockerizing un Gasoducto de Jenkins

por Hüseyin Akdoğan · 30 y 17 de noviembre · DevOps Zone

Descargue el plan que puede llevar a una compañía de cualquier nivel de madurez hasta la entrega continua a escala empresarial utilizando una combinación de Atomic Release Automation, los más de 20 años de experiencia en automatización de negocios de Atomic y las herramientas y prácticas comprobadas que la compañía ya está aprovechando .

La integración continua y la entrega continua son temas fundamentales en la industria del software, especialmente con las tecnologías de nube y contenedores. Las tecnologías de contenedores como Docker , Kubernetes y OpenShift y los servidores de automatización como Jenkins facilitan la administración de los flujos de trabajo de nuestros proyectos.

El repositorio que he creado responde a la pregunta de cómo administrar automáticamente el proceso de construcción, prueba con la cobertura más alta y las fases de implementación.

Nuestro objetivo es garantizar que nuestro oleoducto funcione bien después de cada código. Los procesos que queremos administrar automáticamente:

- Código de pago
- Ejecutar pruebas
- Compila el código
- Ejecute el análisis de Sonarqube en el código
- Crear imagen Docker
- Empuje la imagen al Docker Hub
- Tira y ejecuta la imagen

1) Ejecucion de los servicios

Como uno de los objetivos es obtener el informe Sonarqube de nuestro proyecto, deberíamos poder acceder a Sonarqube desde el servicio de Jenkins. Docker compose es la mejor opción para ejecutar servicios trabajando juntos. Configuramos nuestros servicios de aplicaciones en un archivo yaml, como se muestra a continuación.

docker-compose.yml

```

1  versión : '3.2'
2  servicios :
3    sonarqube :
4      construir :
5        contexto : sonarqube /
6      puertos :
7        - 9000: 9000
8        - 9092: 9092
9      container_name : sonarqube
10   Jenkins :
11     construir :
12       contexto : jenkins /
13     privilegiado : verdadero
14     usuario : root
15     puertos :
16       - 8080: 8080
17       - 50000: 50000
18     container_name : jenkins
19     volúmenes :
20       - / tmp / jenkins: / var / jenkins_home #Recuerde que el directorio tmp está diseñad
21       - /var/run/docker.sock:/var/run/docker.sock
22     depends_on :
23       - sonarqube

```

Las rutas de los archivos Docker de los contenedores se especifican en el atributo de contexto en el archivo docker-compose. El contenido de estos archivos es el siguiente:

sonarqube/Dockerfile

```

1  DESDE sonarqube: 6.7-alpine

```

jenkins/Dockerfile

```

1  DE jenkins: 2.60.3

```

Si ejecutamos el siguiente comando en el mismo directorio que el docker-compose.yml archivo, los contenedores Sonarqube y Jenkins girarán y se ejecutarán.

```

1  docker-compose -f docker-compose.yml up --build
2  docker ps
3  MANDO DE IMAGEN ID DE CONTENEDOR CREADO NOMBRES DE PUERTOS DE ESTADO
   8710f422d65f nicolas_jenkins "/bin/bash" /var/run/docker.sock

```

```
8/1034320655 pipeline_jenkins / bin / run.sh - / usr ... Hace aproximadamente un m
```

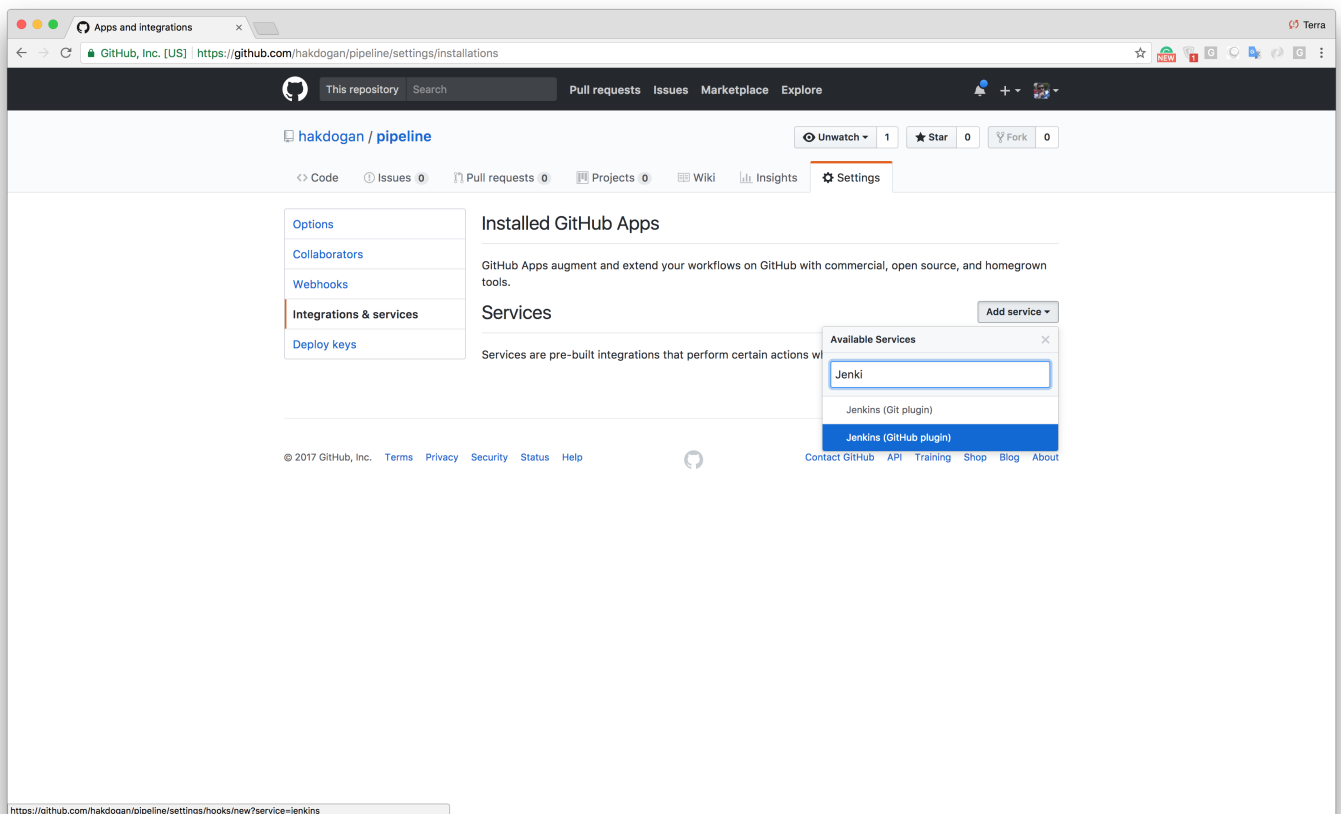
4

```
f5bed5ba3266 pipeline_sonarqube ".bin/run.sh" Hace aproximadamente un minut
```

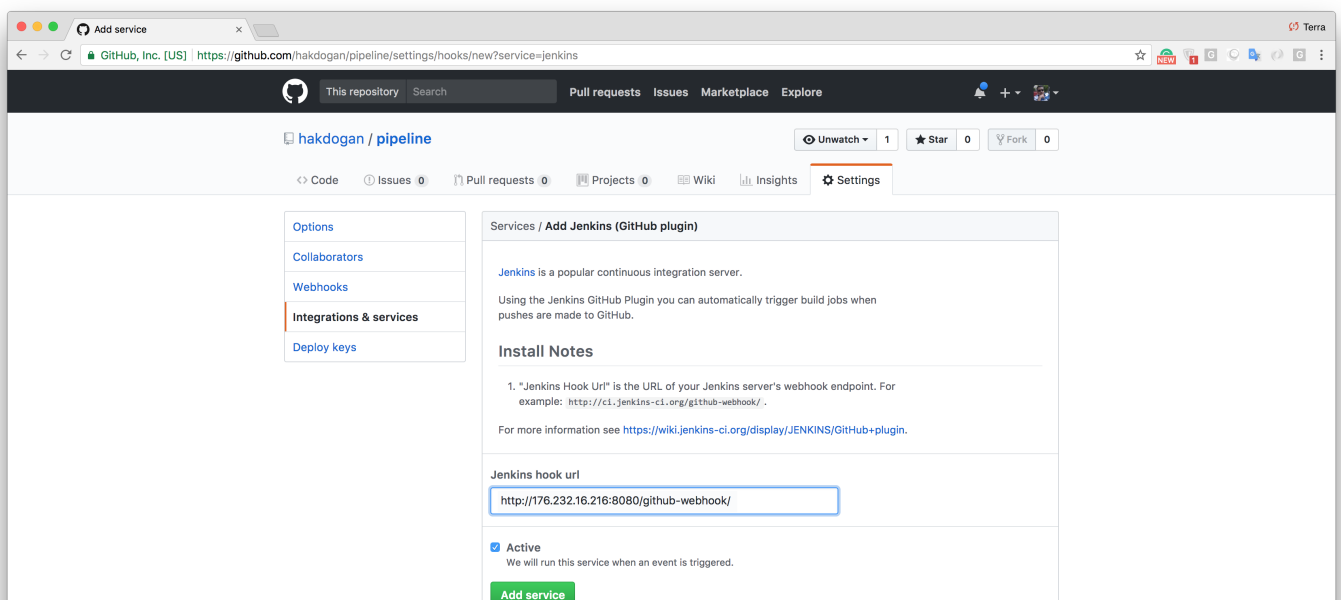
5

Configuración de GitHub

Definiremos un servicio en GitHub para llamar al Jenkins Github webhook porque queremos activar la canalización. Para hacer esto, vaya a *Configuración -> Integraciones y servicios*. El Jenkins Github plugin que desea mostrar en la lista de servicios disponibles, como a continuación.



Después de esto, deberíamos agregar un nuevo servicio escribiendo la URL del contenedor doblado de Jenkins junto con la `/github-webhook/` ruta.



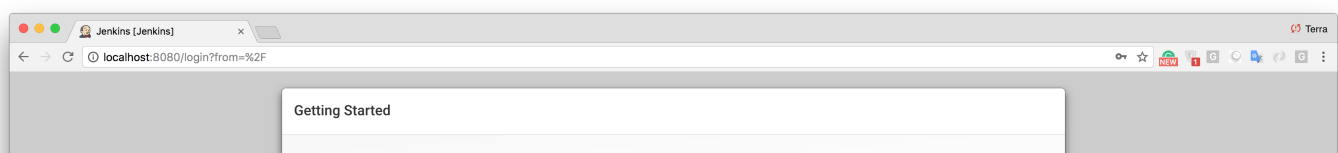
El siguiente paso es crear un `SSH key` usuario de Jenkins y definirlo como `Deploy keys` nuestro repositorio de GitHub.

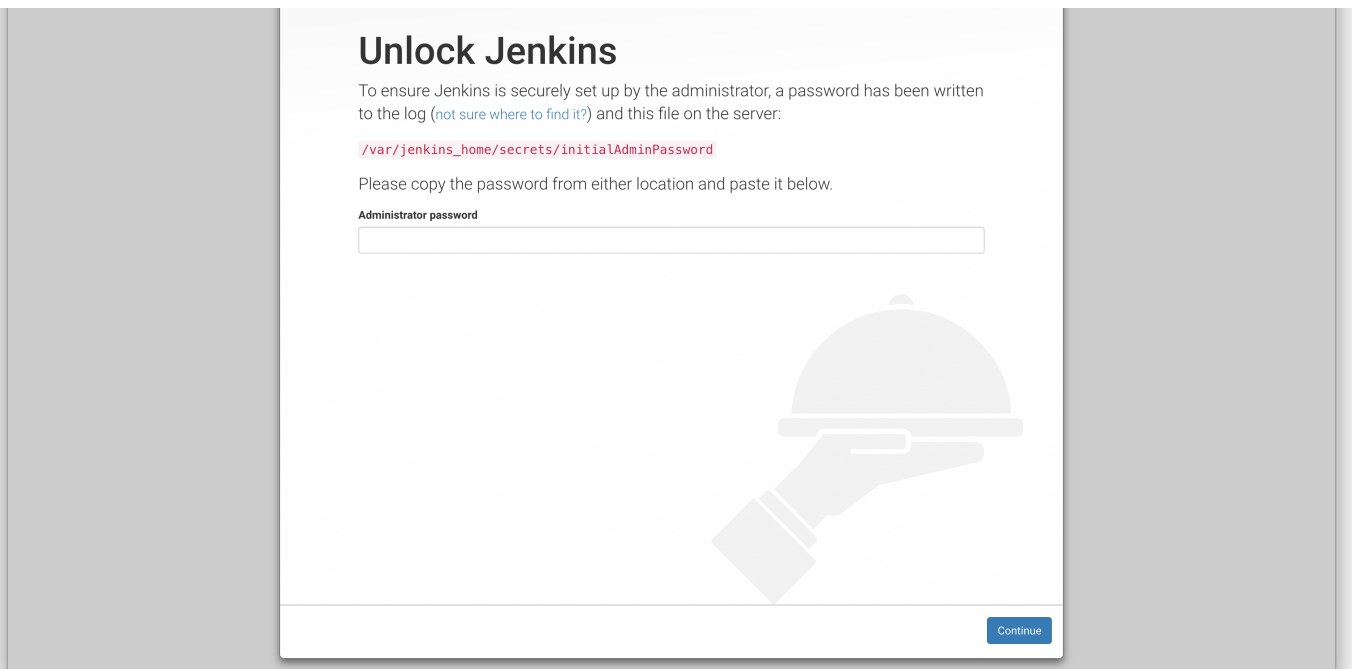


```
1 ssh git @ github.com
2 La solicitud de asignación de PTY falló en el canal 0
3 Hola <tu nombre de usuario github> / <nombre del repositorio>! Se ha autenticado con éxito
4 La conexión a github.com se cerró.
```

Configuración Jenkins

Hemos configurado Jenkins en el archivo de redacción de la ventana acoplable para que se ejecute en el puerto 8080, por lo tanto, si visitamos `http://localhost:8080`, seremos recibidos con una pantalla como esta.





Necesitamos la contraseña de administrador para proceder a la instalación. Se almacena en el `/var/jenkins_home/secrets/initialAdminPassword` directorio y también se escribe como salida en la consola cuando comienza Jenkins.

```

1 Jenkins | *****
2 Jenkins |
3 Jenkins | Se requiere la configuración inicial de Jenkins. Se ha creado un usuario admini:
4 Jenkins | Por favor use la siguiente contraseña para proceder con la instalación:
5 Jenkins |
6 Jenkins | 45638c79cecd4f43962da2933980197e
7 Jenkins |
8 Jenkins | Esto también se puede encontrar en: / var / jenkins_home / secrets / initialAdmi
9 Jenkins |
10 Jenkins | *****

```

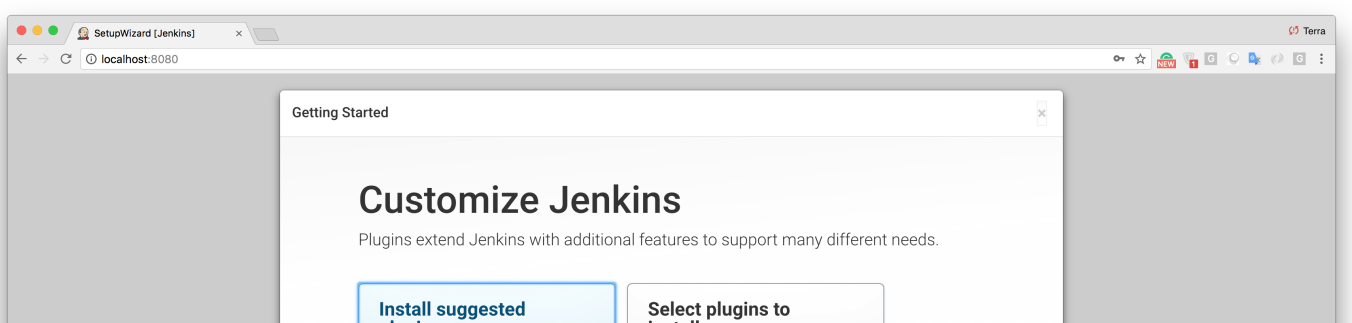
Para acceder a la contraseña desde el contenedor.

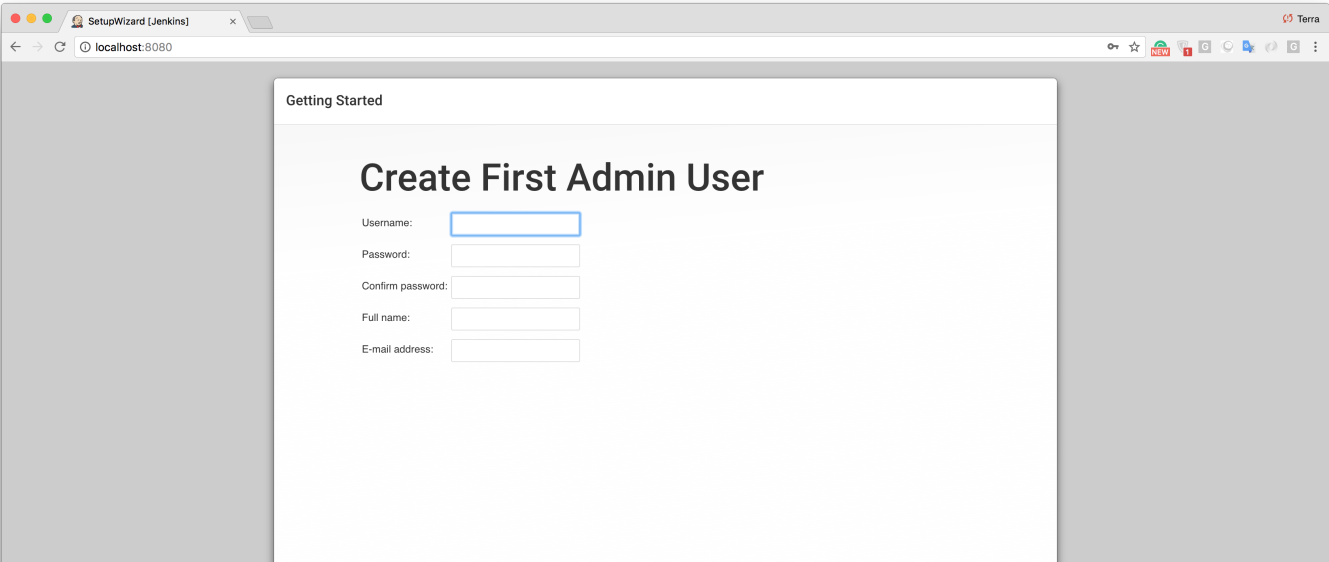
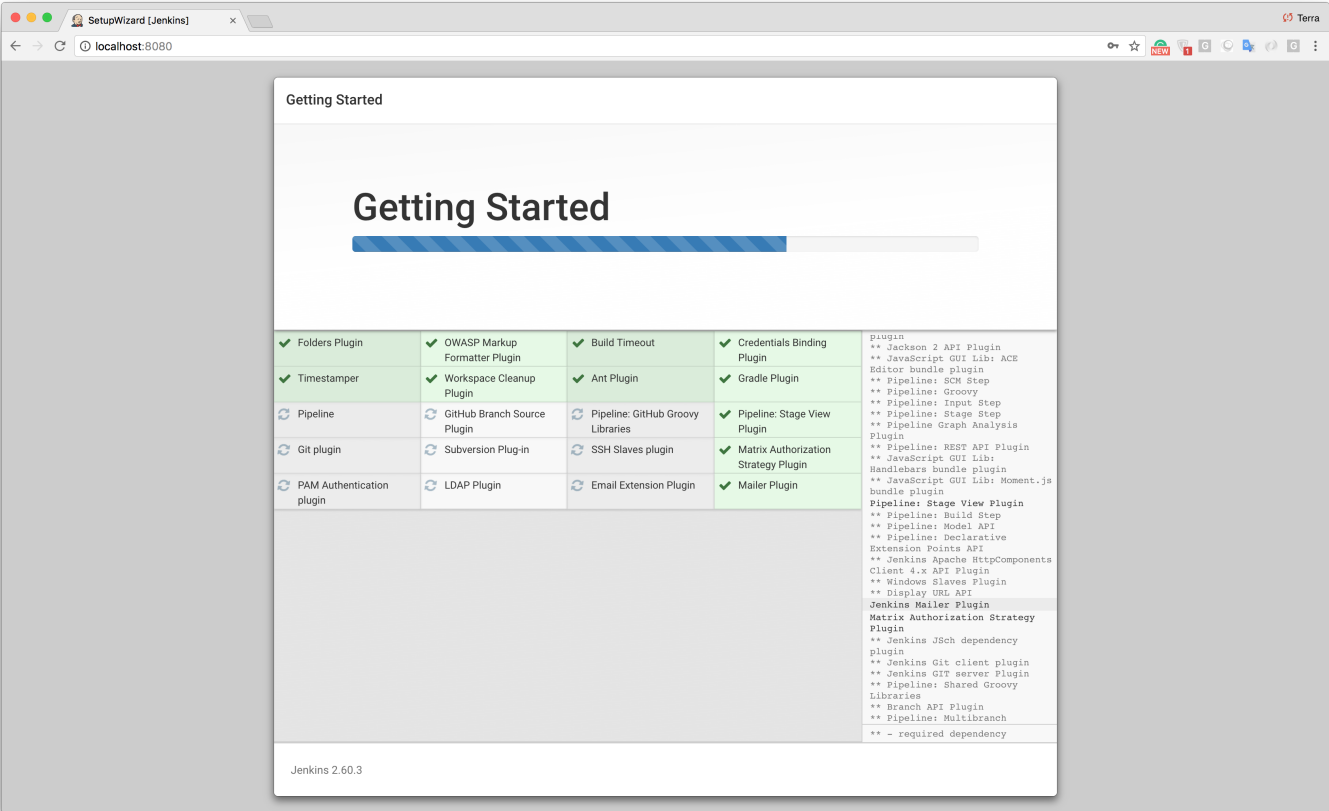
```

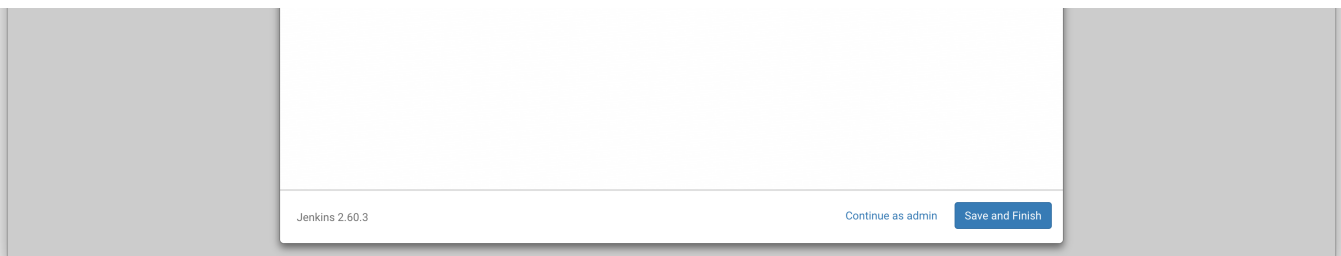
1 docker, ejecutivo , jenkins, sh
2 / $ cat / var / jenkins_home / secrets / initialAdminPassword

```

Después de ingresar la contraseña, descargaremos los complementos recomendados y definiremos un admin user .



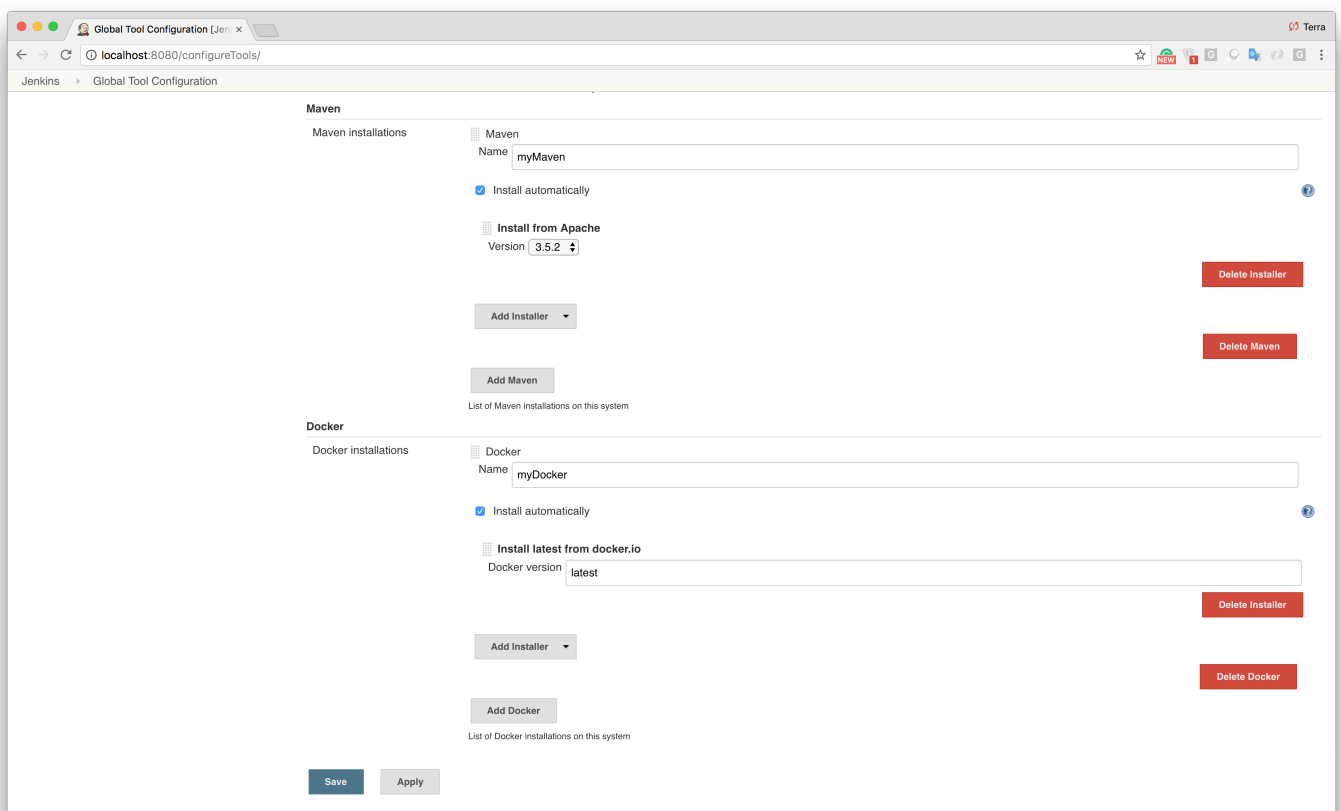




Después de hacer clic en Guardar y Terminar y comenzar a usar los botones de Jenkins, deberíamos ver la página de inicio de Jenkins. Uno de los siete objetivos enumerados anteriormente es que debemos tener la capacidad de construir una imagen en el Jenkins dockerized. Eche un vistazo a las definiciones de volumen del servicio de Jenkins en el archivo de redacción.

```
1 - /var/run/docker.sock:/var/run/docker.sock
```

El propósito es comunicarse entre el Docker Daemon y Docker Client (lo instalaremos en Jenkins) sobre el zócalo. Al igual que el cliente de Docker, también necesitamos Maven compilar la aplicación. Para la instalación de estas herramientas, que necesitamos para llevar a cabo la Maven y Docker client configuraciones bajo *Manejo de Jenkins -> Configuración Global de Herramientas* del menú.



Hemos agregado los instaladores de Maven y Docker y hemos marcado la `Install automatically` casilla de verificación. Jenkins instala estas herramientas cuando nuestra secuencia de comandos (*archivo Jenkins*) se ejecuta por primera vez. Damos `myMaven` y `myDocker` nombramos a las herramientas. Accederemos a estas herramientas con estos nombres en el archivo de script.

Dado que realizaremos algunas operaciones, como la `checkout` base de código y `pushing an image to Docker Hub` , necesitamos definir el `Docker Hub Credentials` . Tenga en cuenta que si estamos utilizando un repositorio privado, debemos definirlo `Github credentials` . Estas definiciones se realizan en la

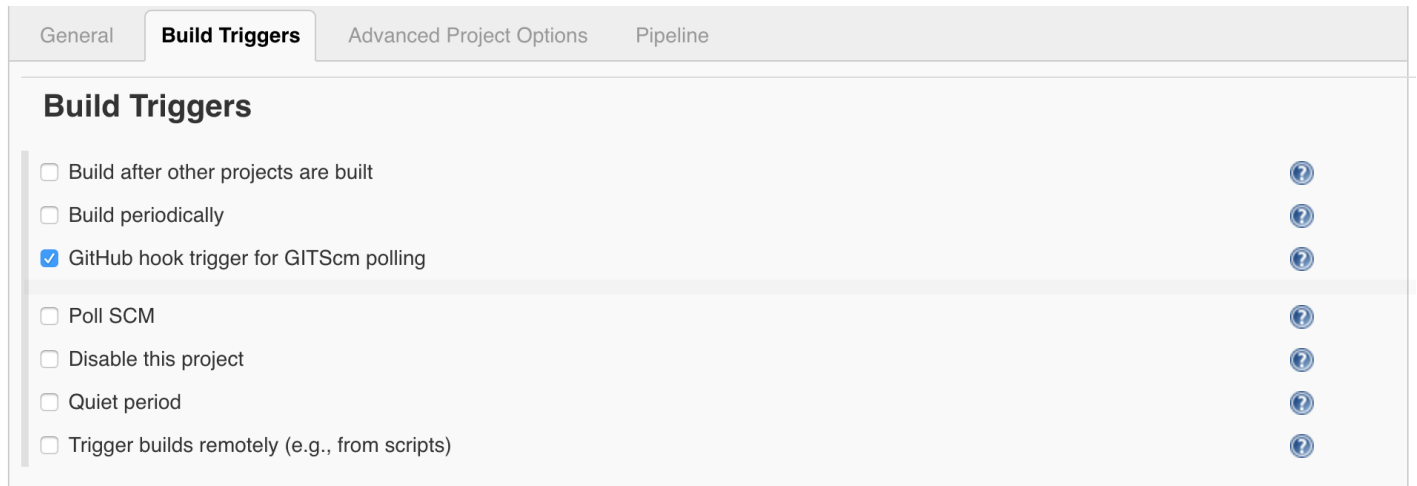
Página de inicio de Jenkins -> Credenciales -> Credenciales globales (sin restricciones) -> Agregar menú de credenciales .

The screenshot shows the 'New Credentials' page in Jenkins. The breadcrumb navigation is 'Jenkins > Credentials > System > Global credentials (unrestricted)'. The form is titled 'New Credentials [Jenkins]'. It has a 'Kind' dropdown set to 'Username with password'. The 'Scope' is 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Username' field is filled with 'hakdogan'. The 'Password' field is masked with dots. The 'ID' field is filled with 'dockerHubAccount'. The 'Description' field is filled with 'Docker Hub Account'. There is an 'OK' button at the bottom left. The footer indicates 'Page generated: 23-Nov-2017 11:23:16 UTC' and 'Jenkins ver. 2.60.3'.

Usamos el valor que ingresamos en el `id` campo para Docker Login en el archivo de script. Ahora, definimos pipeline en *Jenkins Home Page -> New Item* menu.

The screenshot shows the 'New Item' page in Jenkins. The breadcrumb navigation is 'Jenkins > All'. The form is titled 'New Item [Jenkins]'. It has a text input field for 'Enter an item name' with the value 'hakdogan-pipeline-tutorial'. Below this, there are several options for creating a new item: 'Freestyle project', 'Pipeline' (which is highlighted with a blue border), 'Multi-configuration project', 'Folder', 'GitHub Organization', and 'Multibranch Pipeline'. At the bottom, there is a section 'if you want to create a new item from other existing, you can use this option:' with a 'Copy from' dropdown and a text input field. There is an 'OK' button at the bottom left.

En este paso, seleccionamos `GitHub hook trigger for GITScm polling` opciones para una ejecución automática de la canalización por `github hook` llamada.

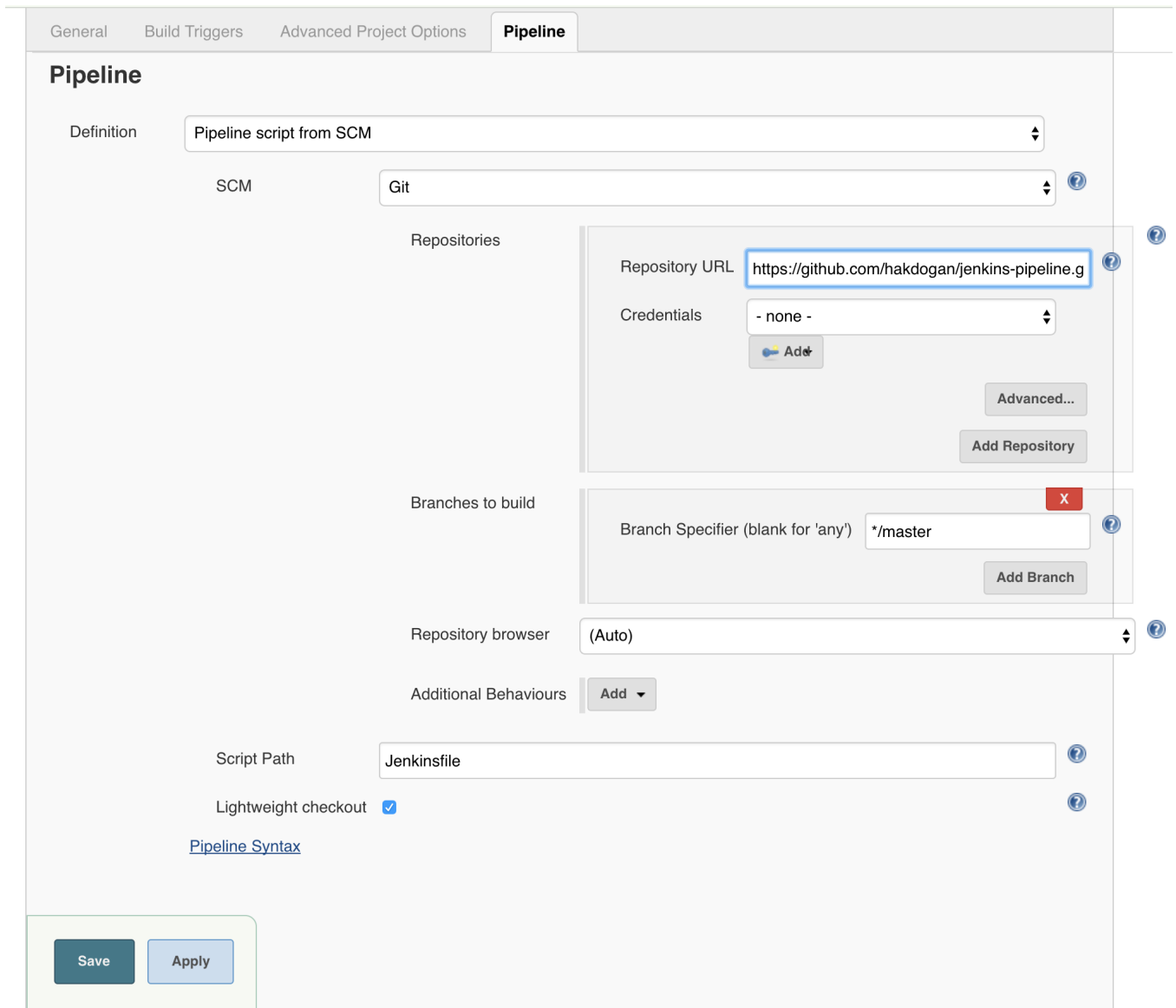


General **Build Triggers** Advanced Project Options Pipeline

Build Triggers

- ☐ Build after other projects are built
- ☐ Build periodically
- ☒ GitHub hook trigger for GITScm polling
- ☐ Poll SCM
- ☐ Disable this project
- ☐ Quiet period
- ☐ Trigger builds remotely (e.g., from scripts)

También en la sección `Pipeline`, seleccionamos `Pipeline script from SCM` como Definición, definimos el repositorio `GitHub` y el nombre de la rama, y especificamos la ubicación del script (archivo `Jenkins`).



General Build Triggers Advanced Project Options **Pipeline**

Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories:

- Repository URL: `https://github.com/hakdogan/jenkins-pipeline.g`
- Credentials: - none -
- Advanced...
- Add Repository

Branches to build:

- Branch Specifier (blank for 'any'): `*/master`
- Add Branch

Repository browser: (Auto)

Additional Behaviours: Add

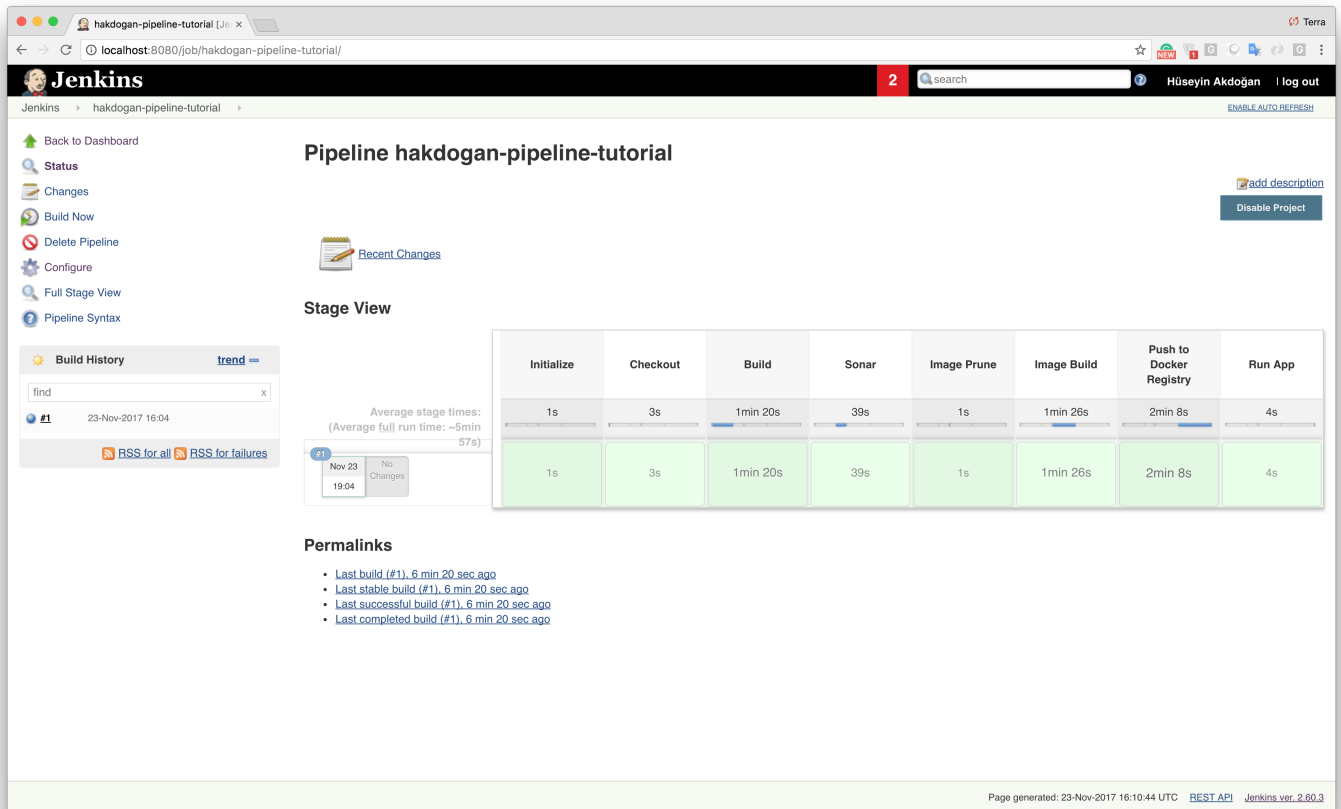
Script Path: Jenkinsfile

Lightweight checkout: ☒

[Pipeline Syntax](#)

Save Apply

Después de eso, cuando se realiza una inserción en el repositorio remoto o cuando se activa manualmente la tubería por `Build Now` opción, se ejecutarán los pasos descritos en el archivo `Jenkins`.



Revisión de puntos importantes del archivo de Jenkins

```

1 stage ( 'Initialize' ) {
2     def dockerHome = herramienta 'myDocker'
3     def mavenHome   = herramienta 'myMaven'
4     env . PATH = "$ { dockerHome } / bin: $ { mavenHome } / bin: $ { env . PATH }"
5 }

```

Las herramientas de cliente de Maven y Docker que hemos definido en Jenkins en el menú Configuración global de herramientas se agregan a la variable de entorno PATH para usar estas herramientas con un comando sh.

```

1 stage ( 'Registro de Push to Docker' ) {
2     withCredentials ([ usernamePassword ( Id de credenciales: 'dockerHubAccount' , usern
3         pushToImage ( CONTAINER_NAME , CONTAINER_TAG , USERNAME , PASSWORD )
4     }
5 }

```

withCredentials proporcionado por Jenkins Credentials Binding Plugin y unir credenciales a las variables. Pasamos el valor dockerHubAccount con el credentialsId parámetro. Recuerde que el valor dockerHubAccount es el ID de credenciales de Docker Hub que hemos definido en *Jenkins Home Page -> Credentials -> Global credentials (no restringido) -> Add Credentials* menu. De esta forma, accedemos a la información de nombre de usuario y contraseña de la cuenta para iniciar sesión.

Configuración Sonarube

Configuración Sonarqube

Para Sonarqube , hemos hecho las siguientes definiciones en el `pom.xml` archivo del proyecto.

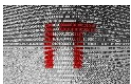
```
1 < sonar.host.url > http: // sonarqube: 9000 </ sonar.host.url >
2 ...
3 < dependencias >
4 ...
5     < dependencia >
6         < groupId > org.codehaus.mojo </ groupId >
7         < artifactId > sonar-maven-plugin </ artifactId >
8         < versión > 2.7.1 </ version >
9         < tipo > maven-plugin </ type >
10     </ dependency >
11 ...
12 </ dependencias >
```

En el archivo de redacción de la ventana acoplable, dimos el nombre del servicio Sonarqube que es sonarqube , por eso en el archivo `pom.xml`, la URL del sonar se definió como `http: // sonarqube: 9000`.

En este artículo, he intentado compartir un tutorial completo, espero que esto pueda ayudarlo.

Descargue el "Plan práctico para la entrega continua" para conocer cómo Automatic Release Automation puede ayudarlo a comenzar o continuar la transformación digital de su empresa.

Me gusta este artículo? Leer más de DZone



DevOps Intelligence cambia el juego



5 razones por las que Jenkins es tan útil y popular



Explorando DevOps: enfoque en CI / CD



Gratis DZone Refcard Comenzando con Docker

Temas: DOCKER, JENKINS, CI / CD, INTEGRACIÓN CONTINUA, ENTREGA CONTINUA, DEVOPS

Publicado en DZone con permiso de Hüseyin Akdoğan. [Vea el artículo original aquí.](#)

Las opiniones expresadas por los contribuidores de DZone son suyas.

Obtenga lo mejor de DevOps en su bandeja de entrada.

Manténgase actualizado con el boletín DevOps quincenal de DZone. [VER UN EJEMPLO](#)

SUSCRIBIR

DevOps Partner Resources

The Ultimate Monitoring Tools Landscape

Papertrail



Setting Up the Critical DevOps Role of Enterprise Release Management

Plutora



The DevOps Journey - From Waterfall to Continuous Delivery

Sauce Labs



Redefine application release processes and unify your DevOps strategy.

Automtic

