



Comience con Jenkins 2.0 con Docker

por Hannah Inman MVB · Mayo. 12, 16 · DevOps Zone

Descargue su copia de Entrega continua con Jenkins y descubra cómo puede entregar mejor el software más rápido. Presentado en asociación con CloudBees .

Este es un blog invitado de Shashikant Jagtap, usuario de Jenkins desde hace mucho tiempo y líder del Jenkins Area Meetup con sede en Londres (JAM).

Siendo un gran admirador de Jenkins, recientemente encontré la versión beta de Jenkins 2.0. El proyecto Jenkins lanzó una versión beta para que los usuarios jugaran con las nuevas características de Jenkins 2.0. ¡La nueva versión de Jenkins, también conocida como Jenkins 2.0 , ya está disponible! Jenkins 2.0 nos ofrece algunas características nuevas e increíbles:

- Pipeline-as-code
- Interfaz de usuario mejorada y experiencia del usuario
- Mejoras de seguridad y complementos
- Nuevo sitio web de Jenkins como una ventanilla única para una guía de inicio y otra documentación

La información detallada sobre la nueva versión se puede encontrar [aquí](#).

En esta publicación, probemos la nueva versión de Jenkins con su propia imagen Docker .

Jenkins Inside Docker

- Datos almacenados para que el reinicio del contenedor no pierda trabajos y datos de complementos

Vamos a crear un Dockerfile para construir nuestro maestro Jenkins:

```
1 $ mkdir jenkins2-docker
2 $ cd jenkins2-docker
3 $ vim Dockerfile
```

Ahora inserte lo siguiente en el archivo Docker:

```
1 DE jenkinsci / jenkins: 2.0-beta-1
2 Raíz del usuario
3 EJECUTAR mkdir / var / log / jenkins
4 EJECUTAR mkdir / var / cache / jenkins
5 RUN chown -R jenkins: jenkins / var / log / jenkins
6 RUN chown -R jenkins: jenkins / var / cache / jenkins
7 USUARIO jenkins
8 ENV JAVA_OPTS = "- Xmx8192m"
```

En este archivo Docker, estamos creando otra imagen sobre la imagen Jenkins Docker con algún directorio de registro de Jenkins y opciones de Java.

Ahora, crearemos otro archivo Docker llamado 'Dockerfile-data' para que podamos montar la imagen / contenedores al ejecutar jenkins-master:

```
1 $ vim Dockerfile-data
```

Agregue el siguiente contenido al archivo Docker:

```
1 DESDE debian: jessie
2
```

Ahora que tenemos un Dockerfile para el maestro Jenkins y los datos de Jenkins, podemos construir esas imágenes:

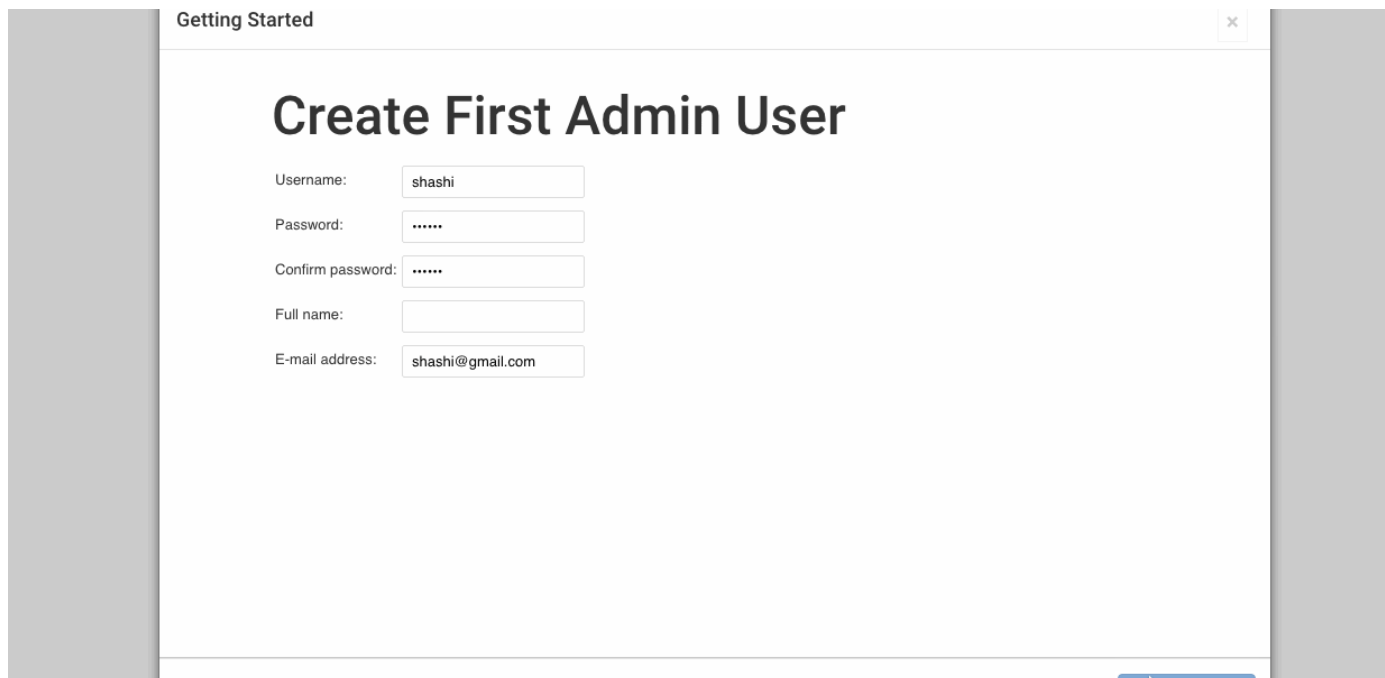
```
1 $ docker build -t jenkins-data -f Dockerfile-data.  
1 $ docker build -t jenkins2.
```

Podemos lanzar los contenedores para cada imagen:

```
1 $ docker run --name = jenkins-data jenkins-data  
$ Estibador plazo -p 8080 : 8080 -p 50000 : 50000 --name = Jenkins-master --volumes -de:  
2
```

Ahora podemos ver que Jenkins 2.0 se iniciará en la IP del host de Docker y en el puerto 8080, por ejemplo: <http://192.168.99.100:8080/>

```
##( 04/04/16@12:56am )( shashi@Shashikants-MacBook-Pro ):/open-source/jenkins2-docker@master✓  
docker run -p 8080:8080 -p 50000:50000 --name=jenkins-master --volumes-from=jenkins-data -d jenkins2  
d7c53da0eb61fb2db9fe630ce91f9b56d3ca8f97e3fe0e3a11a47d18da474413  
##( 04/04/16@12:56am )( shashi@Shashikants-MacBook-Pro ):/open-source/jenkins2-docker@master✓  
docker ps -a
```



The screenshot shows a web browser window titled "Getting Started" with a close button in the top right corner. The main heading is "Create First Admin User". Below the heading are five form fields:

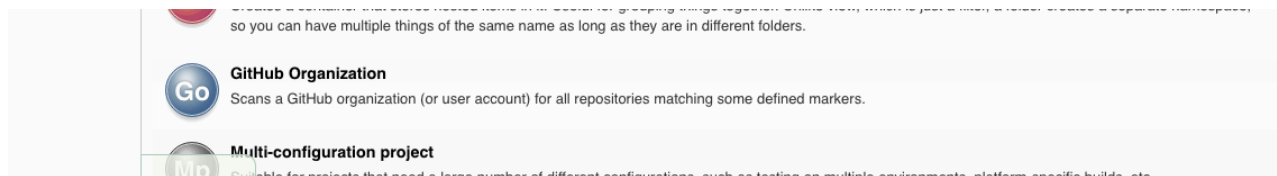
- Username: shashi
- Password: (masked)
- Confirm password: (masked)
- Full name: (empty)
- E-mail address: shashi@gmail.com

Explorando las características de Jenkins 2.0

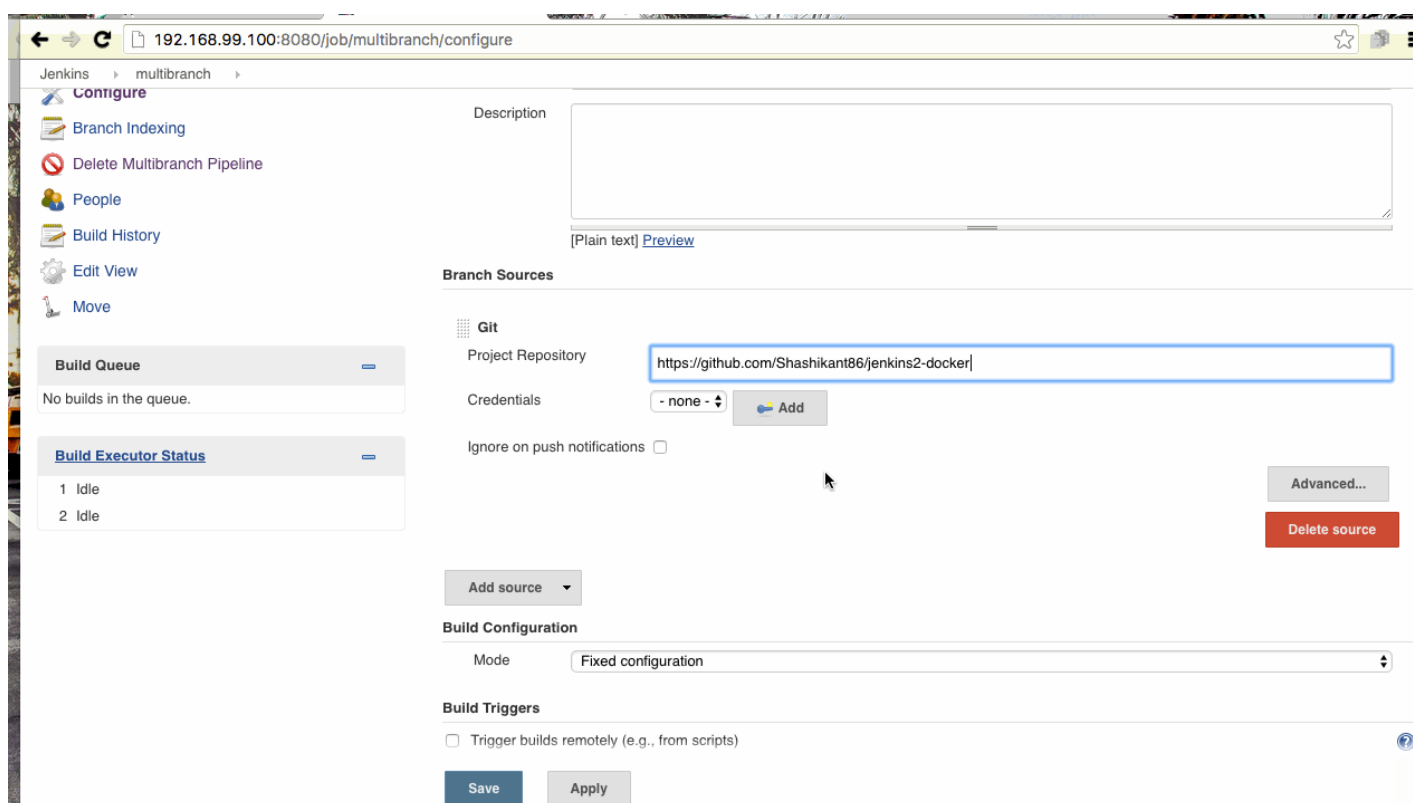
Las características de Jenkins se explican en el informe anterior y demostraremos dos características poderosas:

Esto estaba disponible en la versión anterior de Jenkins, pero ahora es más fácil que nunca. Pipeline-as-code tiene los siguientes beneficios:

- Defina fácilmente las tuberías simples y complejas a través del DSL en un Jenkinsfile.
- Pipeline-as-code proporciona un lenguaje común para ayudar a los equipos (por ejemplo, Dev y Ops) a trabajar juntos.
- Comparta fácilmente las interconexiones entre equipos al almacenar "pasos" comunes en




Ahora podemos crear diferentes tuberías según las ramas de GitHub. He creado un repositorio de Github 'jenkins2-docker' con múltiples ramas y he configurado Jenkins en consecuencia.



Hay algunas otras características de Jenkins 2.0 que se lanzarán y estoy muy emocionado al respecto.

Nota : El código fuente para esta demostración está disponible en Github Repo llamado 'Jenkins2-Docker'. Este artículo fue escrito originalmente por Shashikant Jagtap, <http://shashikantjagtap.net>

Temas: DOCKER, JENKINS CI

Publicado en DZone con el permiso de Hannah Inman , DZone MVB . [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

Recursos para socios de DevOps

Continúe su transformación digital con un Blueprint para entrega continua.

Automic



Automatice la seguridad en su oleoducto DevOps

Sonatype



4 formas de mejorar tus pruebas de DevOps

xMatters



Jenkins, Docker y DevOps: los catalizadores de innovación

CloudBees

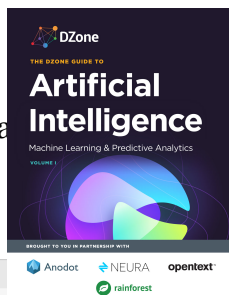


How to Handle File Uploads Using

Let's work on the case in which we need to upload a file and validate whether the file is uploaded.

Steps:

- Launch the URL of the application
- Maximize the window
- Use the file upload widget to upload a



Manual Mode:

La Guía de Inteligencia Artificial 2017: Aprendizaje automático y análisis predictivo

- Descubra patrones en Analytics usando el poder del aprendizaje automático
- Aprenda sobre las redes neuronales usando las bibliotecas de Java
- Vea cómo el proyecto de código abierto de Google puede enriquecer las aplicaciones empresariales

1 - Open Browser
2 - Maximize Window
3 - Upload File
4 - Get Attribute
5 - Verify Match

Upload File
Upload File
Upload File
Upload File
Upload File

"C:\\Users\\Public\\Pictures\\Sample Pictures\\Desert.jpg"
"C:\\Users\\Public\\Pictures\\Sample Pictures\\Desert.jpg"
"C:\\Users\\Public\\Pictures\\Sample Pictures\\Desert.jpg"
"C:\\Users\\Public\\Pictures\\Sample Pictures\\Desert.jpg"
"C:\\Users\\Public\\Pictures\\Sample Pictures\\Desert.jpg"

Open browser and navigate to given URL
Maximize the window
Passing the path of the file
Capturing the file name after upload and storing it in a variable
Verifying the Actual path and Expected path of the

We can also use the **script mode**. The script below is the code to upload a file and validate the uploaded file.

Script Mode:

Descargar My Free PDF

```

1 'Open browser and navigate to given URL'
WebUI.openBrowser('C:\\\\Users\\\\User\\\\Desktop\\\\Katalon Articles\\\\File Upload\\\\Up
2
3 'Maximize the window\r\n'
WebUI.maximizeWindow()
4
5 'Passing the path of the file'
WebUI.uploadFile(findTestObject('Upload File'), 'C:\\\\Users\\\\Public\\\\Pictures\\\\Sam
6
7
8 'Capturing the file name after upload and storing it in a variable'
```

Manual Mode:

Upload file by Sendkeys

+ Add - Delete + Move up - Move down

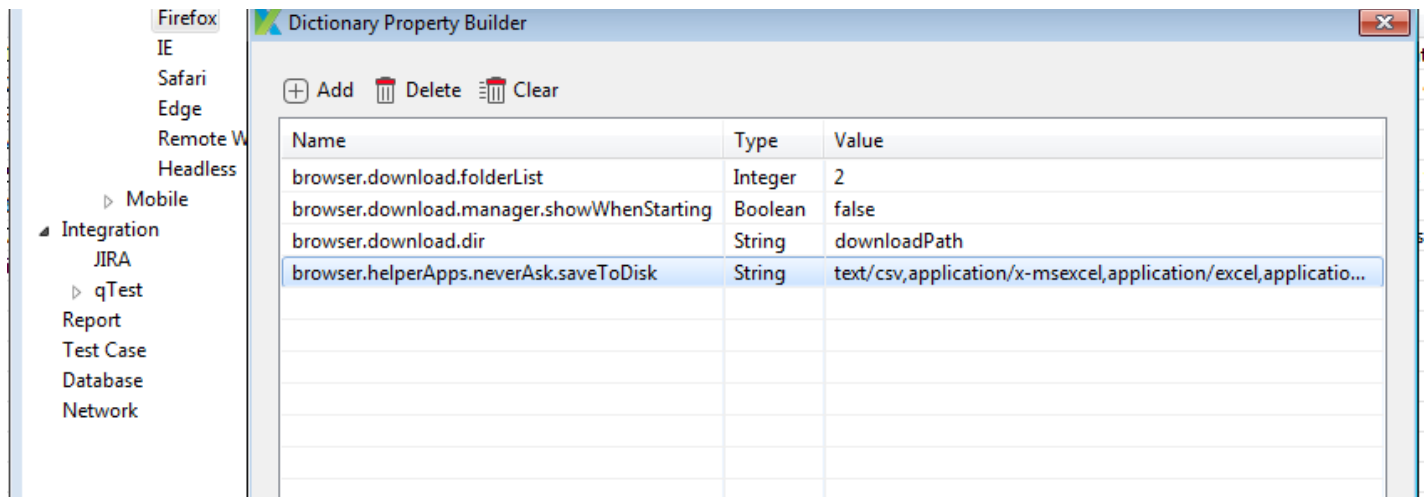
Item	Object	Input	Output	Description
1 - Open Browser		"C:\\Users\\User\\Desktop\\Katalon Articles\\File Upload\\UploadFile.html"		Open browser and navigate to given URL
2 - Maximize Window				Maximize the window
3 - Send Keys	Upload File	"C:\\Users\\Public\\Pictures\\Sample Pictures\\Desert.jpg"		
4 - Get Attribute	Upload File	"value"	FilePath	
5 - Verify Match		FilePath; "C:\\fakepath\\Desert.jpg"; false		Verifying the Actual path and Expected path of file

Script Mode:

```

1  'Open browser and navigate to given URL'
2
3  WebUI.openBrowser('C:\\\\Users\\\\User\\\\Desktop\\\\Katalon Articles\\\\File Upload\\\\Up
4  <
5  'Maximize the window\r\n'
6
7  WebUI.maximizeWindow()
8
9  'Uploading the File using Send Keys method by passing the File path'
10
11 WebUI.sendKeys(findTestObject('Upload File'), 'C:\\\\Users\\\\Public\\\\Pictures\\\\Sample
12 <
13 'Capturing the file name after upload and storing it in a variable'
14
15 FilePath = WebUI.getAttribute(findTestObject('Upload File'), 'value')
16
17 'Verifying the Actual path and Expected path of file'

```

Script Mode:

```

1  import org.openqa.selenium.By as By
2  import org.openqa.selenium.WebDriver as WebDriver
3  import org.testng.Assert as Assert
4  import com.kms.katalon.core.webui.driver.DriverFactory as DriverFactory
5  import com.kms.katalon.core.webui.keyword.WebUiBuiltInKeywords as WebUI
6  import internal.GlobalVariable as GlobalVariable
7
8  'Define Custom Path where file needs to be downloaded'
9  String downloadPath = 'D:\\FileDownloadChecking'
10
11 'Launch a browser and Navigate to URL'
12 WebUI.openBrowser(GlobalVariable.FileDownloadCheckingURL)
13
14 WebDriver driver = DriverFactory.getWebDriver()
15
16 'Clicking on a Link text to download a file'

```

```
32 for (int i = 0; i < dir_contents.length; i++) {  
33     println('File Name at 0 is : ' + dir_contents[i].getName())  
34     'Verifying the file name is available in the folder '  
35     if (dir_contents[i].getName().equals(fileName)) {  
36         'If the file is found then it will return a value as true'  
37         return flag = true  
38     }  
39 }  
40 'If the file is found then it will return a value as false'  
41 return flag  
42 }
```

We have just learned how to handle file uploads and verify downloaded files using Katalon Studio. You can download the source code [here](#).

For further instructions and help, please refer to the [Upload File guidelines](#).

The DevOps Zone is brought to you in partnership with Sonatype Nexus. See how the Nexus platform infuses precise open source component intelligence into the DevOps pipeline early, everywhere, and at scale. Read how in this [ebook](#).

Like This Article? Read More From DZone



**Securing Your Inline SQL
Statements From SQL Injection**



**Crime Analysis Using H2O
Autoencoders (Part 1)**



**The 12 Days of Content (Day 12):
Developer Resolutions**



**Free DZone Refcard
Getting Started With Docker**

