



EBook Gratis

APRENDIZAJE jenkins

Free unaffiliated eBook created from
Stack Overflow contributors.

#jenkins

Tabla de contenido

Acerca de.....	1
Capítulo 1: Empezando con Jenkins.....	2
Observaciones.....	2
Versiones.....	2
Jenkins.....	2
Jenkins 1.x vs Jenkins 2.x.....	2
Examples.....	5
Instalación.....	5
Actualización de jenkins (instalaciones RPM).....	6
Configurando el Proxy Nginx.....	6
Instalar el complemento desde una fuente externa.....	7
Mueve a Jenkins de una PC a otra.....	7
Configurar un proyecto en Jenkins.....	7
Introducción completa de Jenkins en un solo lugar.....	8
Configure un proyecto de compilación simple con Jenkins 2 pipeline script.....	14
Capítulo 2: Complemento de estrategia de rol.....	16
Examples.....	16
Configuración.....	16
Administrar roles.....	16
Asignar roles.....	17
Capítulo 3: Configuración de Build Automation para iOS utilizando Shenzhen.....	20
Examples.....	20
Configuración de automatización de compilación de iOS utilizando Shenzhen.....	20
Capítulo 4: Configuración de Jenkins para la automatización de la compilación de iOS.....	21
Introducción.....	21
Parámetros.....	21
Observaciones.....	21
Examples.....	21
Ejemplo de tabla de tiempo.....	21
Capítulo 5: Configurar Auto Git Push en la construcción exitosa en Jenkins.....	23

Introducción.....	23
Examples.....	23
Configurando el Auto Push Job.....	23
Capítulo 6: Instale Jenkins en Windows con soporte SSH para repositorios privados de GitHub..	32
Examples.....	32
Las solicitudes de extracción de GitHub fallan.....	32
PSEXEC.exe PS Tool de Microsoft.....	32
Genere una nueva clave SSH solo para Jenkins usando PSEXEC o PSEXEC64.....	32
Crear las credenciales de Jenkins.....	33
Ejecute una solicitud de extracción de prueba para verificar, y listo.....	35
Capítulo 7: Jenkins Groovy Scripting	37
Examples.....	37
Crear usuario predeterminado.....	37
Deshabilitar el asistente de configuración.....	37
Cómo obtener información sobre la instancia de Jenkins.....	38
Cómo obtener información sobre un trabajo de Jenkins.....	38
Creditos	39

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [jenkins](#)

It is an unofficial and free jenkins ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official jenkins.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con Jenkins

Observaciones

[Jenkins](#) es una herramienta de integración continua de código abierto escrita en Java. El proyecto fue bifurcado de [Hudson](#) después de una disputa con [Oracle](#) .

Jenkins proporciona servicios de integración continua para el desarrollo de software. Es un sistema basado en servidor que se ejecuta en un contenedor de servlets como Apache Tomcat. Es compatible con herramientas SCM, incluidas AccuRev, CVS, Subversion, Git, Mercurial, Perforce, Clearcase y RTC, y puede ejecutar proyectos basados en Apache Ant y Apache Maven, así como scripts de shell arbitrarios y comandos por lotes de Windows. El principal desarrollador de Jenkins es [Kohsuke Kawaguchi](#) . Lanzado bajo la licencia MIT, Jenkins es software libre.

Las compilaciones pueden iniciarse por diversos medios, incluido el desencadenamiento por cometer en un sistema de control de versiones, programando a través de un mecanismo similar a cron, construyendo cuando otras compilaciones se han completado y solicitando una URL de compilación específica.

Versiones

Jenkins

Versión	Fecha de lanzamiento
1.656	2016-04-03
2.0	2016-04-20

Jenkins 1.x vs Jenkins 2.x

Jenkins es (y sigue siendo) un sistema de integración continua (CI) que permite la automatización del proceso de desarrollo de software, como el código de construcción en los activadores de confirmación de SCM. Sin embargo, la creciente necesidad de entrega continua (CD) ha solicitado que Jenkins evolucione para un sistema CI puro a una mezcla de CI y CD. Además, la necesidad de no materializar los trabajos de Jenkins ha ido creciendo y los trabajos clásicos de Jenkins 1.x `Freestyle/Maven jobs` comenzaron a ser demasiado limitados para ciertas necesidades.

Bajo Jenkins, el complemento 1.xa llamado `workflow-plugin` apareció para permitir a los desarrolladores escribir código para describir trabajos. Jenkins 2 va más allá al agregar soporte incorporado para `Pipeline as Code` . El principal beneficio es que las tuberías, al ser archivos de

scripts Groovy, pueden ser más complejas que los trabajos de estilo libre configurados por UI y pueden ser controladas por versión. Jenkins 2 también agrega una nueva interfaz que facilita la visualización de diferentes "etapas" definidas en una tubería y sigue el progreso de toda la tubería, como a continuación:

Stage View

Average stage times:
(Average full run time: ~27y
220d)

Build the sudo
images for
installation

19s

#34

Mar 03

18:56

81
commits

1 min 34s

master

failed

#33

Dec 22

13:00

3
commits



17s

master

#32

Dec 22

12:33

20
commits



15s

master

#31

Dec 22

12:16

No
Changes



16s

master

#30

Dec 22

No



Descripción general de Jenkins 2 .

Además, el [registro de cambios completo](#) está disponible en el sitio web de Jenkins.

Examples

Instalación

Para sistemas basados en apt-get como Ubuntu

Agrega el repositorio de Jenkins:

```
wget -q -O - https://jenkins-ci.org/debian/ Jenkins-ci.org.key | sudo apt-key
```

```
sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian-stable binary/ >  
/etc/apt/sources.list.d/jenkins.list'
```

Actualice las fuentes e instale Jenkins:

```
sudo apt-get update
```

```
sudo apt-get install jenkins
```

Ahora se crea un usuario jenkins y, de forma predeterminada, Jenkins se ejecutará en el puerto 8080.

Para distribuciones basadas en RPM como Red Hat Enterprise Linux (RHEL), CentOS, Fedora o Scientific Linux

Para descargar el archivo de repositorio para la versión estable:

```
sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo
```

O si quieres los últimos lanzamientos semanales:

```
sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat/jenkins.repo
```

Importar la clave pública:

```
sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key
```

Instale Jenkins usando yum:

```
sudo yum install jenkins
```

Jenkins requiere java para ejecutarse, para instalarlo:

```
sudo yum install java
```

Para iniciar / detener / reiniciar el uso de jenkins:

```
sudo service jenkins start/stop/restart
```


Actualización de jenkins (instalaciones RPM)

1. Directorio de inicio de jenkins de copia de seguridad
2. Reemplace jenkins.war en la siguiente ubicación con un nuevo archivo WAR. / usr / lib / jenkins / jenkins.war`
3. Reiniciar Jenkins
4. Verifique los complementos fijados y, si es necesario, descuelgue
5. Reload Configuration from Disk

nota: para las actualizaciones de Jenkins 2 para el servidor de aplicaciones jetty agrupadas, desactive el puerto AJP (configure `JENKINS_AJP_PORT="-1"`) en `/etc/sysconfig/jenkins` .

Configurando el Proxy Nginx

De forma nativa, Jenkins se ejecuta en el puerto 8080. Podemos establecer un proxy desde el puerto 80 -> 8080 para que se pueda acceder a Jenkins a través de:

```
http://<url>.com
```

en lugar de la predeterminada

```
http://<url>.com:8080
```

Comience por instalar Nginx.

```
sudo aptitude -y install nginx
```

Eliminar la configuración predeterminada para Nginx

```
cd /etc/nginx/sites-available
```

```
sudo rm default ../sites-enabled/default
```

Crear el nuevo archivo de configuración.

```
sudo touch jenkins
```

Copie el siguiente código en el archivo `jenkins` recién creado.

```
upstream app_server {
    server 127.0.0.1:8080 fail_timeout=0;
}

server {
    listen 80;
    listen [::]:80 default ipv6only=on;
    server_name ;

    location / {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_redirect off;
```

```
if (!-f $request_filename) {  
    proxy_pass http://app_server;  
    break;  
}  
}  
}
```

Cree un enlace simbólico entre sitios disponibles y sitios habilitados:

```
sudo ln -s /etc/nginx/sites-available/jenkins /etc/nginx/sites-enabled/
```

Reinicie el servicio de proxy Nginx

```
sudo service nginx restart
```

Jenkins ahora será accesible desde el puerto 80.

Instalar el complemento desde una fuente externa

```
java -jar [Path to client JAR] -s [Server address] install-plugin [Plugin ID]
```

El cliente JAR debe ser el archivo CLI JAR, no el mismo JAR / WAR que ejecuta el propio Jenkins. Las ID únicas se pueden encontrar en una página respectiva de los complementos en la wiki de la CLI de Jenkins (<https://wiki.jenkins-ci.org/display/JENKINS/Plugins>)

Mueve a Jenkins de una PC a otra

Esto me sirvió para pasar de Ubuntu 12.04 (versión Jenkins 1.628) a Ubuntu 16.04 (versión Jenkins 1.651.2). Primero [instalé Jenkins desde los repositorios](#) .

1. [Detener ambos servidores Jenkins](#)
2. Copie `JENKINS_HOME` (por ejemplo, `/var/lib/jenkins`) desde el servidor antiguo al nuevo. Desde una consola en el nuevo servidor:

```
rsync -av username@old-server-IP:/var/lib/jenkins/ /var/lib/jenkins/
```

3. [Comience su nuevo servidor Jenkins](#)

Puede que no necesites esto, pero tuve que

- Manage Jenkins y Reload Configuration from Disk .
- Desconecta y vuelve a conectar todos los esclavos.
- Verifique que en `Configure System > Jenkins Location` , la Jenkins URL esté asignada correctamente al nuevo servidor de Jenkins.

Configurar un proyecto en Jenkins.

Aquí revisaremos la última copia del código de nuestro proyecto, ejecutaremos las pruebas y haremos la aplicación en vivo. Para lograrlo, siga los pasos a continuación:

1. Abre Jenkins en el navegador.
2. Haga clic en el enlace **Nuevo trabajo**.
3. Ingrese el nombre del proyecto y seleccione el enlace **Crear un proyecto de software de estilo libre**.
4. Haga clic en el botón **Aceptar**.
5. En la **sección Administración del código fuente**, seleccione la casilla de radio junto a la herramienta de administración del código fuente. En mi caso he seleccionado **Git**.

Proporcionar url de git repo como `git://github.com/example/example.git`

6. Debajo de los **desencadenantes de compilación**, seleccione el cuadro de radio junto a **Encuesta SCM**.
7. Proporcionar ********* en el cuadro de **horario**. Este cuadro es responsable de activar la compilación a intervalos regulares. ********* especifica que, el trabajo se activará cada minuto para los cambios en git repo.
8. En la sección **Generar**, haga clic en el botón **Agregar paso de compilación** y luego seleccione la opción con la que desea compilar el proyecto. He seleccionado **Ejecutar Shell**. En el cuadro de comandos, escriba el comando para compilar, ejecute las pruebas y desplácelo a prod.
9. Desplácese hacia abajo y **Guardar**.

Así que arriba, hemos configurado un proyecto básico en Jenkins que activará la compilación a cada minuto para el cambio en su repositorio git. Nota: para configurar el proyecto complejo, es posible que tenga que instalar algunos complementos en Jenkins.

Introducción completa de Jenkins en un solo lugar.

1. Jenkins:

Jenkins es una herramienta de integración continua de código abierto escrita en Java. El proyecto fue bifurcado de Hudson después de una disputa con Oracle.

En pocas palabras, Jenkins es el principal servidor de automatización de código abierto. Construido con Java, proporciona cientos de complementos para permitir la creación, prueba, implementación y automatización de prácticamente cualquier proyecto.

Características: Jenkins ofrece las siguientes funciones principales listas para usar, y muchas más se pueden agregar a través de complementos:

Instalación sencilla: simplemente ejecute `java -jar jenkins.war`, desplácelo en un contenedor de servlets. Sin instalación adicional, sin base de datos. ¿Prefieres un instalador o paquete nativo? Tenemos esos también. Configuración sencilla: Jenkins se puede configurar completamente desde su interfaz gráfica de usuario amigable con extensas verificaciones de errores sobre la marcha y ayuda en línea. Ecosistema de complementos enriquecidos: Jenkins se integra con prácticamente todos los SCM o herramientas de construcción existentes. Ver complementos. Extensibilidad: la mayoría de las partes de Jenkins se pueden ampliar y modificar, y es fácil crear nuevos complementos de Jenkins. Esto le permite personalizar Jenkins a sus necesidades. Compilaciones distribuidas: Jenkins puede distribuir cargas de compilación / prueba a varias

computadoras con diferentes sistemas operativos. ¿Construyendo software para OS X, Linux y Windows? No hay problema.

Instalación:

```
$ wget -q -O - https://jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-key add -

$ sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian binary/ >
/etc/apt/sources.list.d/jenkins.list'
$ sudo apt-get update
$ sudo apt-get install jenkins
to do more refer link :
```

Ref: <https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Ubuntu>

Ref: <http://www.vogella.com/tutorials/Jenkins/article.html>

Ref: <https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins>

Directorio **JENKINS_HOME** Jenkins necesita algo de espacio en el disco para realizar compilaciones y mantener archivos. Puede verificar esta ubicación desde la pantalla de configuración de Jenkins. De forma predeterminada, se establece en `~/.jenkins`, pero puede cambiarlo de una de las siguientes maneras: Establezca la variable de entorno "JENKINS_HOME" en el nuevo directorio de inicio antes de iniciar el contenedor de servlet. Establezca la propiedad del sistema "JENKINS_HOME" en el contenedor de servlets. Establezca la entrada de entorno JNDI "JENKINS_HOME" en el nuevo directorio. Consulte la colección de documentación específica del contenedor para obtener más información sobre cómo hacer esto para su contenedor. También puedes cambiar esta ubicación después de haber usado Jenkins por un tiempo. Para hacer esto, detenga completamente a Jenkins, mueva los contenidos del antiguo JENKINS_HOME al nuevo hogar, establezca el nuevo JENKINS_HOME y reinicie Jenkins. JENKINS_HOME tiene una estructura de directorio bastante obvia que se parece a la siguiente:

JENKINS_HOME

```
+-- config.xml      (jenkins root configuration)
+-- *.xml           (other site-wide configuration files)
+-- userContent     (files in this directory will be served under your
http://server/userContent/)
+-- fingerprints    (stores fingerprint records)
+-- plugins         (stores plugins)
+-- workspace       (working directory for the version control system)
    +- [JOBNAME]    (sub directory for each job)
+-- jobs
    +- [JOBNAME]    (sub directory for each job)
        +- config.xml (job configuration file)
        +- latest     (symbolic link to the last successful build)
        +- builds
            +- [BUILD_ID] (for each build)
                +- build.xml (build result summary)
                +- log       (log file)
                +- changelog.xml (change log)
```

Jenkins Build Jobs:

Crear un nuevo trabajo de construcción en Jenkins es simple: simplemente haga clic en el elemento del menú "Nuevo trabajo" en el panel de Jenkins. Jenkins admite varios tipos diferentes de trabajos de creación, que se le presentan cuando elige crear un nuevo trabajo

Proyecto de software freestyle

Los trabajos de compilación de estilo libre son trabajos de compilación de propósito general, que proporcionan una flexibilidad máxima.

Proyecto Maven El "proyecto maven2 / 3" es un trabajo de construcción especialmente adaptado a los proyectos Maven. Jenkins entiende los archivos de pom Maven y las estructuras del proyecto, y puede usar la información obtenida del archivo pom para reducir el trabajo que necesita hacer para configurar su proyecto.

Flujo de trabajo

Organiza actividades de larga duración que pueden abarcar varios esclavos de compilación. Adecuado para construir tuberías y / u organizar actividades complejas que no encajan fácilmente en el tipo de trabajo de estilo libre.

Supervisar un trabajo externo El trabajo de compilación "Supervisar un trabajo externo" le permite vigilar los procesos no interactivos, como los trabajos cron.

Trabajo de configuración múltiple El "proyecto de configuración múltiple" (también conocido como "proyecto de matriz") le permite ejecutar el mismo trabajo de creación en muchas configuraciones diferentes. Esta potente función puede ser útil para probar una aplicación en muchos entornos diferentes, con diferentes bases de datos o incluso en diferentes máquinas de compilación.

1. Construyendo un proyecto de software (estilo libre)

Jenkins se puede utilizar para realizar el trabajo típico del servidor de compilación, como hacer compilaciones continuas / oficiales / nocturnas, ejecutar pruebas o realizar algunas tareas repetitivas por lotes. Esto se llama "proyecto de software de estilo libre" en Jenkins. Configuración del proyecto Vaya a la página principal de Jenkins, seleccione "Nuevo trabajo", luego elija "Crear un proyecto de software de estilo libre". Este tipo de trabajo consta de los siguientes elementos: SCM opcional, como CVS o Subversion, donde reside su código fuente. Desencadenadores opcionales para controlar cuándo Jenkins realizará compilaciones. algún tipo de script de compilación que realiza la compilación (ant, maven, shell script, archivo por lotes, etc.) donde el trabajo real ocurre pasos opcionales para recopilar información de la compilación, como archivar los artefactos y / o grabar javadoc y probar resultados pasos opcionales para notificar a otras personas / sistemas con el resultado de la compilación, como enviar correos electrónicos, mensajes instantáneos, actualizar el rastreador de problemas, etc.

Compilaciones para proyectos de control que no son de origen A veces es necesario compilar un proyecto simplemente con fines de demostración o el acceso a un repositorio SVN / CVS no está disponible. Al elegir configurar el proyecto como "Ninguno" en "Administración del código fuente", tendrá que:

1. Genere el proyecto al menos una vez (fallará), pero Jenkins creará la estructura jenkins / workspace / PROJECTNAME /
2. Copie los archivos del proyecto a jenkins / workspace / PROJECTNAME /
3. Construir de nuevo y configurar adecuadamente.

Jenkins establece variables de entorno

Cuando se ejecuta un trabajo de Jenkins, establece algunas variables de entorno que puede usar en su script de shell, comando por lotes, script de Ant o POM de Maven. Vea la lista de variables haciendo clic en ENVIRONMENT_VARIABLE

Configurando compilaciones automáticas

Las compilaciones en Jenkins se pueden activar periódicamente (según una programación, especificada en la configuración), o cuando se han detectado cambios en la fuente del proyecto, o se pueden activar automáticamente solicitando la URL:

<http://YOURHOST/jenkins/job/PROJECTNAME/build>

Esto le permite enganchar las construcciones de Jenkins en una variedad de configuraciones. Para obtener más información (en particular, hacer esto con la seguridad habilitada), consulte API de acceso remoto.

Construye por fuente de cambios

Puede hacer que Jenkins evalúe su sistema de control de revisión para detectar cambios. Puede especificar con qué frecuencia Jenkins sondea su sistema de control de revisión utilizando la misma sintaxis que crontab en Unix / Linux. Sin embargo, si su período de sondeo es más corto de lo necesario para sondear su sistema de control de revisión, puede terminar con múltiples compilaciones para cada cambio. Debe ajustar su período de sondeo para que sea más largo que el tiempo necesario para sondear su sistema de control de revisión o utilizar un activador posterior a la confirmación. Puede examinar el registro de sondeo de cada compilación para ver cuánto tiempo llevó encuestar su sistema.

Alternativamente, en lugar de sondear en un intervalo fijo, puede usar un activador de URL (descrito anteriormente), pero con / polling en lugar de / build al final de la URL. Esto hace que Jenkins encueste al SCM en busca de cambios en lugar de construirlo inmediatamente. Esto evita que Jenkins ejecute una compilación sin cambios relevantes para las confirmaciones que afectan a los módulos o ramas que no están relacionados con el trabajo. Cuando se utiliza / sondeo, el trabajo debe configurarse para sondeo, pero la programación puede estar vacía.

Construye por correo electrónico (sendmail)

Si tiene la cuenta raíz de su sistema y está usando sendmail, encontré la forma más fácil de modificar / etc / aliases y agregar la siguiente entrada: jenkins-foo: "| / bin / wget -o / dev / null

<http://YOURHOST/jenkins/job/PROJECTNAME/build> "

y luego ejecute el comando "newaliases" para informar a sendmail sobre el cambio. Cada vez que

alguien envíe un correo electrónico a "jenkins-foo @ yoursystem", esto activará una nueva compilación. Vea esto para más detalles sobre la configuración de sendmail. Construye por correo electrónico (qmail) Con qmail, puede escribir /var/qmail/alias/.qmail-jenkins de la siguiente manera: | / bin / wget -o / dev / null [http: // YOURHOST / jenkins / job / PROJECTNAME / construir](http://YOURHOST/jenkins/job/PROJECTNAME/construir)

2. Construyendo un proyecto Maven

Jenkins proporciona un tipo de trabajo dedicado a Maven 2/3. Este tipo de trabajo integra a Jenkins profundamente con Maven 2/3 y proporciona los siguientes beneficios en comparación con el proyecto de software de estilo libre más genérico.

Jenkins analiza los POM de Maven para obtener gran parte de la información necesaria para hacer su trabajo. Como resultado, la cantidad de configuración se reduce drásticamente.

Jenkins escucha la ejecución de Maven y descubre qué se debe hacer cuando está solo. Por ejemplo, registrará automáticamente el informe JUnit cuando Maven ejecute la fase de prueba. O si ejecuta el objetivo javadoc, Jenkins grabará automáticamente javadoc.

Jenkins crea automáticamente dependencias de proyecto entre proyectos que declaran dependencias de SNAPSHOT entre sí. Vea abajo. Por lo tanto, en su mayoría solo necesita configurar la información de SCM y los objetivos que le gustaría ejecutar, y Jenkins descubrirá todo lo demás.

Este tipo de proyecto puede proporcionar automáticamente las siguientes características:

Archivar artefactos producidos por una construcción.

Publicar resultados de prueba

Desencadenar trabajos para proyectos que son dependencias posteriores

Despliega tus artefactos en un repositorio de Maven

Resultados de la prueba de ruptura por módulo

Opcionalmente, reconstruya solo los módulos modificados, acelerando sus construcciones

Encadenamiento automático a partir de dependencias de módulos.

Jenkins lee las dependencias de su proyecto desde su POM, y si también se basan en Jenkins, los desencadenantes se configuran de tal manera que una nueva compilación en una de esas dependencias iniciará automáticamente una nueva compilación de su proyecto. Jenkins entiende todo tipo de dependencias en POM. A saber, padre POM

```
<dependencies> section of your project
<plugins> section of your project
<extensions> section of your project
<reporting> section of your project
```

Este proceso tiene en cuenta las versiones, por lo que puede tener varias versiones / ramas de su

proyecto en el mismo Jenkins y determinará correctamente las dependencias. **Tenga en cuenta** que los rangos de versiones de dependencia no son compatibles, consulte [<https://issues.jenkins-ci.org/browse/JENKINS-2787> ♦♦1] para conocer el motivo.

Esta función se puede deshabilitar a petición - vea la opción de configuración Construir cada vez que se genere una dependencia SNAPSHOT

Instalación:

1. entra en Manage Jenkins >> configura el sistema
2. en la pestaña maven "Haga clic en la instalación maven

Puede hacer que Jenkins instale automáticamente una versión específica de Maven, o proporcionar una ruta a una instalación local de Maven (puede configurar tantas versiones de Maven para sus proyectos de compilación como desee, y usar diferentes versiones de Maven para diferentes proyectos. Si marca la casilla Instalar automáticamente, Jenkins descargará e instalará la versión solicitada de Maven por usted y la instalará en el directorio de herramientas en el directorio de inicio de Jenkins.

Cómo usarlo

Primero, debe configurar una instalación de Maven (este paso puede omitirse si está utilizando DEV @ cloud). Esto se puede hacer yendo a la pantalla de configuración del sistema (Administrar Jenkins-> Configurar sistema). En la sección "Instalaciones de Maven", 1) haga clic en el botón Agregar, 2) asígnele un nombre como "Maven 3.0.3" y luego 3) elija la versión del menú desplegable.

Ahora, Jenkins instalará automáticamente esta versión cada vez que sea necesaria (por ejemplo, en cualquier nueva máquina de compilación) descargándola de Apache y descomprimiéndola.

Crear un nuevo trabajo de Maven:

1. Al hacer clic en "Nuevo trabajo / Nuevo elemento" en la mano izquierda
2. Dale un nombre
3. Elija el "Construir un proyecto de Maven 2/3"
4. Salva tu trabajo

Ahora necesitas configurar tu trabajo.

1. Elija el SCM que desea usar (ej. Usar git)
2. elegir objetivo de Maven para llamar
3. Añadir URL y credencial del repositorio.
4. comprobar usuario privado maven repo:

También puede definir la ruta de acceso para el mismo.

5. Proyecto de construcción

Cree su proyecto haciendo clic en **compilar ahora** y haga clic en la barra de progreso en la parte izquierda **"Crear estado del ejecutor"** para ver cómo Jenkins instala Maven, verifica su proyecto y construye usando maven.

Explotación florestal:

<https://wiki.jenkins-ci.org/display/JENKINS/Logging>

Consola de Script:

Jenkins proporciona una consola de scripts que le brinda acceso a todas las funciones internas de Jenkins. Estos scripts están escritos en Groovy y encontrará algunos ejemplos de ellos en esta [página](#).


Configure un proyecto de compilación simple con Jenkins 2 pipeline script

Aquí crearemos un canal de Groovy en Jenkins 2 para realizar los siguientes pasos:

- Verifique cada 5 minutos si se ha enviado un nuevo código a nuestro proyecto.
- Código de salida de repo SCM
- Maven compila nuestro código de Java.
- Ejecuta nuestras pruebas de integración y publica los resultados.

Aquí están los pasos que haremos:

1. Asegúrese de que tengamos al menos una versión 2.0 de Jenkins (puede verificarlo en la esquina inferior derecha de su página) como:

 Jenkins ver. 2.6

2. En la página de inicio de Jenkins, haga clic en **Nuevo artículo**
3. Ingrese el nombre del proyecto y seleccione **Pipeline**
4. En la sección **Disparadores de compilación**, seleccione la opción **Poll SCM** y agregue el siguiente horario CRON de 5 minutos: `* / 5 * * * *`
5. En la sección **Pipeline**, elija **Pipeline Script** o **Pipeline Script desde SCM**
6. Si seleccionó **Pipeline Script de SCM** en el paso anterior, ahora necesita especificar su URL de repositorio de SCM (Git, Mercurial, Subversion) en la **URL del repositorio**, como `http://github.com/example/example.git`. También debe especificar la **ruta de la secuencia de comandos** de su archivo de secuencias de comandos Groovy en su repositorio `example.git`, por ejemplo, `pipelines/example.groovy`
7. Copie el siguiente código Groovy, ya sea directamente en la ventana del script Groovy si previamente hizo clic en **Pipeline Script** o en su `example.groovy` si eligió **Pipeline Script de SCM**

```

node('remote') {
    // Note : this step is only needed if you're using direct Groovy scripting
    stage 'Checkout Git project'
    git url: 'https://github.com/jglick/simple-maven-project-with-tests.git'
    def appVersion = version()
    if (appVersion) {
        echo "Building version ${appVersion}"
    }

    stage 'Build Maven project'
    def mvnHome = tool 'M3'
    sh "${mvnHome}/bin/mvn -B -Dmaven.test.failure.ignore verify"
    step([$class: 'JUnitResultArchiver', testResults: '**/target/surefire-reports/TEST-
*.xml'])
}
def version() {
    def matcher = readFile('pom.xml') =~ '<version>(.)</version>'
    matcher ? matcher[0][1] : null
}

```

Aquí tienes, ahora deberías poder compilar y probar tu primer proyecto de Jenkins utilizando la canalización Jenkins 2 Groovy.

Lea Empezando con Jenkins en línea: <https://riptutorial.com/es/jenkins/topic/919/empezando-con-jenkins>

Capítulo 2: Complemento de estrategia de rol

Examples

Configuración

Administrar roles

Roles globales : cree roles con el conjunto seleccionado de características de Jenkins, por ejemplo, generalmente para un proyecto de desarrollo, se pueden crear 2 roles.

1. Desarrollador: el rol global solo se puede configurar como **general** : leer
2. ProjectOwner- El rol global se puede establecer en **General** : Leer

Esto restringe al desarrollador y al propietario del proyecto para leer el acceso a todas las características de Jenkins.



Manage and Assign Roles

Global roles

Role	Overall					Credentials				
	Administer	Configure	UpdateCenter	Read	RunScripts	UploadPlugins	Create	Delete	ManageDomains	Update
Developer	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
ProjectOwner	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Role to add



Add

Funciones de proyecto : cree funciones restringiendo el acceso de los usuarios a las respectivas funciones de trabajo y credenciales de jenkins mediante expresiones regulares.

Por ejemplo, para un proyecto de desarrollo 'MyProjectA'; los propietarios de proyectos deben tener permisos completos para los trabajos y los desarrolladores necesitan acceso de compilación a los trabajos de Jenkins. Así creamos los siguientes roles:

- **ProjectA_admin** : marque todas las opciones en Job viz. *Construir, cancelar, configurar, crear, eliminar, descubrir, mover, leer, área de trabajo*
- **ProjectA_dev** - verifique las opciones Crear, Cancelar, Leer, Área de trabajo en Trabajo

Project roles

	Role	Pattern	Credentials					Job						
			Create	Delete	ManageDomains	Update	View	Build	Cancel	Configure	Create	Delete	Discover	Move
	ProjectA_admin	MyProjectA.*.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	ProjectA_dev	MyProjectA.*.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Role to add

Pattern

Add

Para restringir los proyectos anteriores a los propietarios y desarrolladores de proyectos respectivos, todos los trabajos deben seguir un patrón predefinido.

Supongamos que 'MyProjectA' necesita 3 trabajos de compilación de jenkins:
MyProjectA_Dev_Build, *MyProjectA_QA_Build*, *MyProjectA_Nightly_Sonar_Analysis*

Para restringir el propietario del proyecto y los desarrolladores del proyecto 'MyProjectA' a trabajos de compilación anteriores, proporcione ' *Patrón* ' como **MyProjectA. ***.

Asignar roles

Ayuda a asignar usuarios o grupos de proyectos a los respectivos roles de Global o Project. Por ejemplo, para asignar un desarrollador 'Gautam' al rol global de desarrollador, proporcione el nombre de usuario 'Gautam', haga clic en *Agregar* y seleccione la casilla de verificación junto a 'Gautam' y debajo del rol global de desarrollador.



Assign Roles

Global roles

User/group	Developer	ProjectOwner	admin
admin	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
gautam	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

User/group to add

Add

Project roles

User/group	ProjectA_admin	ProjectA_dev
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>
gautam	<input type="checkbox"/>	<input checked="" type="checkbox"/>

User/group to add

Add

De manera similar, agregue el usuario bajo roles de proyecto y seleccione los roles de proyecto respectivos para asignar los roles de proyecto requeridos.

Si observa que debajo de las capturas de pantalla puede ver que el usuario 'gautam' tiene acceso solo a los proyectos que comienzan con MyProjectA. Además, el acceso del usuario está restringido para construir y configurar falta.

All		MyProjectA		
S	W	Name	Last Success	Last Failure
		MyProjectA Dev Build	N/A	N/A
		MyProjectA Nightly Sonar Analysis	N/A	N/A
		MyProjectA QA Build	N/A	N/A

 [Back to Dashboard](#)

 [Status](#)

 [Changes](#)

 [Workspace](#)

 [Build Now](#)

Project MyProjectA_Dev_Build

 [Workspace](#)

 [Recent Changes](#)

 **Build History** [trend](#) 

 [RSS for all](#)  [RSS for failures](#)

Permalinks

Lea Complemento de estrategia de rol en línea:

<https://riptutorial.com/es/jenkins/topic/5741/complemento-de-estrategia-de-rol>

Capítulo 3: Configuración de Build Automation para iOS utilizando Shenzhen

Examples

Configuración de automatización de compilación de iOS utilizando Shenzhen

Parte I: Configurar la máquina Mac para usar Shenzhen

Ir a la terminal

Instalar Shenzhen

```
sudo gema instalar shenzhen
```

```
sudo gema instalar nomad-cli
```

Descargar la utilidad de línea de comandos XCode

```
xcode-select --install
```

Aparece una ventana emergente con el siguiente texto

El comando xcode-select requiere las herramientas de desarrollo de la línea de comandos. ¿Te gustaría instalar las herramientas ahora?

Click - Instalar

Crear directorio de proyectos

Gitclone tu proyecto

```
git clone https://akshat@bitbucket.org/company/projectrepo.git
```

Proyecto de compilación utilizando el siguiente comando

```
ipa build --verbose
```

PD: Si ve algún error, seleccione el Perfil de aprovisionamiento activo y confirme con los archivos del proyecto. y ejecuta `ipa build --verbose` nuevamente.

Lea Configuración de Build Automation para iOS utilizando Shenzhen en línea:

<https://riptutorial.com/es/jenkins/topic/8002/configuracion-de-build-automation-para-ios-utilizando-shenzhen>

Capítulo 4: Configuración de Jenkins para la automatización de la compilación de iOS.

Introducción

Ahora puede definir el proceso de Integración continua y Entrega continua (**CI / CD**) como código con Jenkins 2.0 para sus proyectos en iOS 10. Las actividades como compilación, prueba, cobertura de código, estilo de verificación, informes y notificaciones se pueden describir en una sola. expediente.

Para leer el artículo completo, vaya a [Pipeline en Jenkins 2.0 como Código para iOS 10 y XCode 8](#)

Parámetros

Parámetro	Detalles
nodo ('nodo de iOS')	Jenkins Node con Mac OS. Si Jenkins está instalado en Mac OS, use el <code>node {....}</code>

Observaciones

El artículo está escrito en ambos idiomas: inglés y español.

Examples

Ejemplo de tabla de tiempo

El código fuente se puede [clonar o descargar de GitHub](#) para probarlo.

```
node('iOS Node') {  
  
    stage('Checkout/Build/Test') {  
  
        // Checkout files.  
        checkout([  
            $class: 'GitSCM',  
            branches: [[name: 'master']],  
            doGenerateSubmoduleConfigurations: false,  
            extensions: [], submoduleCfg: [],  
            userRemoteConfigs: [[  
                name: 'github',  
                url: 'https://github.com/mmorejón/time-table.git'  
            ]]  
        ])  
    }  
}
```



```

        // Build and Test
        sh 'xcodebuild -scheme "TimeTable" -configuration "Debug" build test -destination
"platform=iOS Simulator,name=iPhone 6,OS=10.1" -enableCodeCoverage YES |
/usr/local/bin/xcpretty -r junit'

        // Publish test results.
        step([$class: 'JUnitResultArchiver', allowEmptyResults: true, testResults:
'build/reports/junit.xml'])
    }

    stage('Analytics') {

        parallel Coverage: {
            // Generate Code Coverage report
            sh '/usr/local/bin/slather coverage --jenkins --html --scheme TimeTable
TimeTable.xcodeproj/'

            // Publish coverage results
            publishHTML([allowMissing: false, alwaysLinkToLastBuild: false, keepAll: false,
reportDir: 'html', reportFiles: 'index.html', reportName: 'Coverage Report'])

        }, Checkstyle: {

            // Generate Checkstyle report
            sh '/usr/local/bin/swiftlint lint --reporter checkstyle > checkstyle.xml || true'

            // Publish checkstyle result
            step([$class: 'CheckStylePublisher', canComputeNew: false, defaultEncoding: '',
healthy: '', pattern: 'checkstyle.xml', unhealthy: ''])
        }, failFast: true|false
    }

    stage ('Notify') {
        // Send slack notification
        slackSend channel: '#my-team', message: 'Time Table - Successfully', teamDomain: 'my-
team', token: 'my-token'
    }
}

```

Lea Configuración de Jenkins para la automatización de la compilación de iOS. en línea:

<https://riptutorial.com/es/jenkins/topic/8868/configuracion-de-jenkins-para-la-automatizacion-de-la-compilacion-de-ios->

Capítulo 5: Configurar Auto Git Push en la construcción exitosa en Jenkins

Introducción

Este documento lo guiará a través de los pasos para configurar un trabajo de Jenkins que le permita al usuario configurar el empuje automático en una construcción exitosa. La operación de empuje puede ser controlada por el usuario. El usuario puede elegir si desea realizar la operación de empuje automático en una compilación exitosa o no.

Examples

Configurando el Auto Push Job

Crear un trabajo de construcción (de acuerdo a sus requerimientos). Para este ejemplo, he creado un trabajo de estilo libre (AutoPush) para realizar la compilación ANT.

Vamos a crear dos variables, PUSH (parámetro de elección) y TAG_NUMBER (parámetro de cadena).

Podemos elegir el valor SÍ o NO para PUSH, esto decidirá si empujar el código a una etiqueta o no en una compilación exitosa.

Podemos especificar un nombre de etiqueta (ej. 1.0.1) para TAG_NUMBER para crear una nueva etiqueta (ej. 1.0.1) en el repositorio remoto con el mismo nombre o especificar un nombre de etiqueta existente para actualizar una etiqueta existente.

Project AutoPush

This build requires parameters:

PUSH

YES ▼

Controls whether to push the code to a new release tag or not.

TAG_NUMBER

1.0.1

Enter a new tag number to create a new tag or enter an existing tag number to update an existing tag.

Build

Ahora vamos a pasar a la configuración del trabajo.

1. Marque la casilla de verificación "Este proyecto está parametrizado" y cree un Parámetro de elección llamado "PUSH" y proporcione SÍ y NO como las opciones. Este parámetro decidirá si desea enviar el código a una etiqueta / versión específica o no.

☒ This project is parameterized

The screenshot shows the 'Choice Parameter' configuration window. It has a title bar with a list icon and the text 'Choice Parameter'. Below the title bar, there are three main sections: 'Name' with a text input field containing 'PUSH'; 'Choices' with a text area containing 'YES' and 'NO'; and 'Description' with a text area containing 'Controls whether to push the code to a new release tag or not.' At the bottom of the description area, there is a link '[Plain text] Preview'.

2. Luego cree un Parámetro de Cadena llamado "TAG_NUMBER", utilizando este parámetro podemos especificar un nuevo número de etiqueta para crear una nueva etiqueta o especificar un número de etiqueta existente para actualizar una etiqueta existente.

The screenshot shows the 'String Parameter' configuration window. It has a title bar with a list icon and the text 'String Parameter'. Below the title bar, there are three main sections: 'Name' with a text input field containing 'TAG_NUMBER'; 'Default Value' with an empty text input field; and 'Description' with a text area containing 'Enter a new tag number to create a new tag or enter an existing tag number to update an existing tag.' At the bottom of the description area, there is a link '[Plain text] Preview'. Below the description area, there is a button 'Add Parameter' with a dropdown arrow.

3. En la sección Administración del código fuente, seleccione Git y proporcione la URL del repositorio. Este repositorio contiene el código fuente que va a compilar y después de una compilación exitosa, se creará una etiqueta de lanzamiento en el mismo repositorio.


Source Code Management

☐ None

☒ Git

Repositories

Repository URL

Credentials 



Advanced

Add Repository

Branches to build

Branch Specifier (blank for 'any')

Add Branch

4. Después de agregar los detalles del repositorio, haga clic en Avanzado y proporcione un nombre a su repositorio que luego será referido en el complemento de Git Publisher para identificar el repositorio.


Source Code Management

☐ None

☒ Git

Repositories

Repository URL

Credentials 



Advanced

Add Repository

Branches to build

Branch Specifier (blank for 'any')

Add Branch

Source Code Management

☐ None

☒ Git

Repositories

Repository URL

Credentials

Name

Refspec

Branches to build

Branch Specifier (blank for 'any')

5. A continuación, agregue el paso de compilación. En este ejemplo estoy construyendo un proyecto ANT.

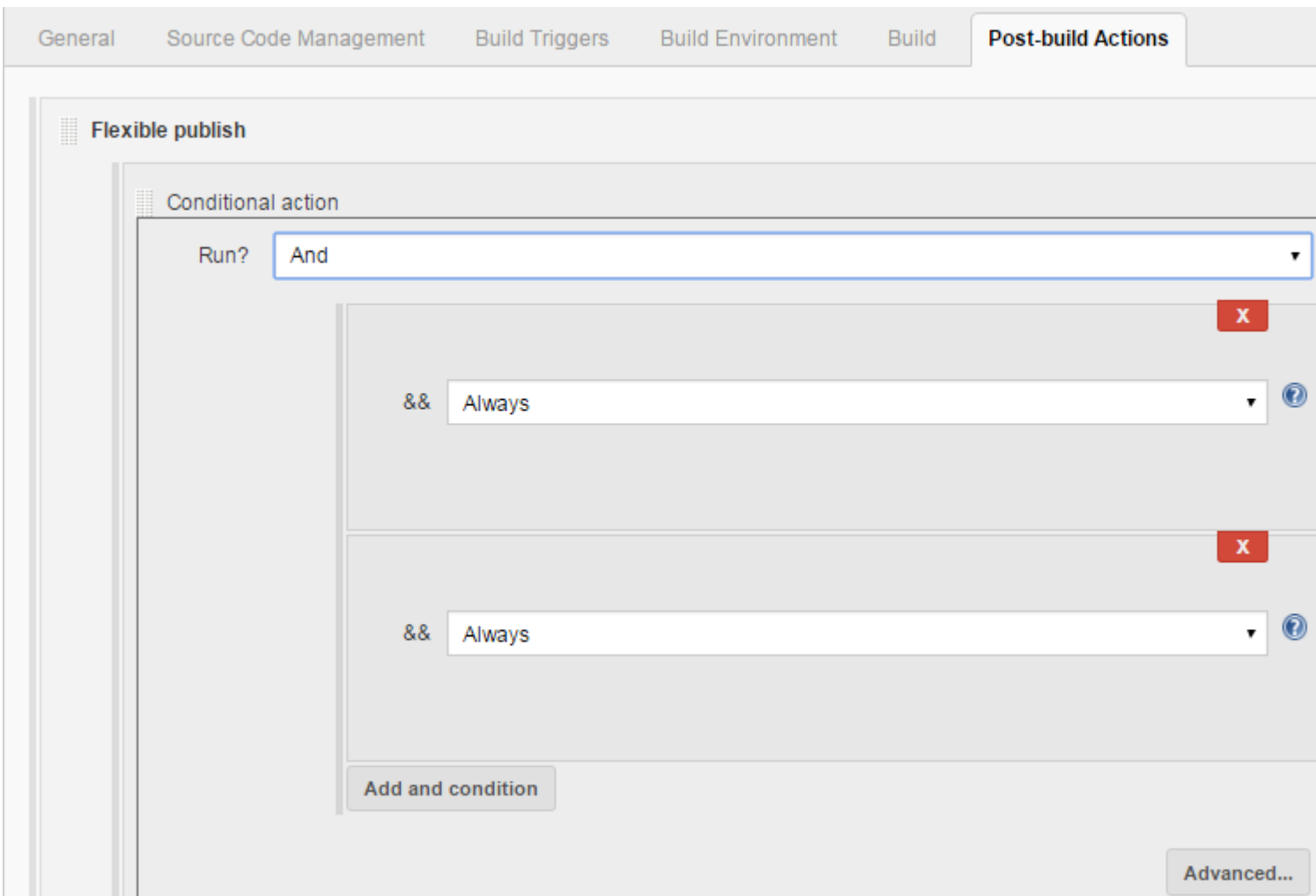
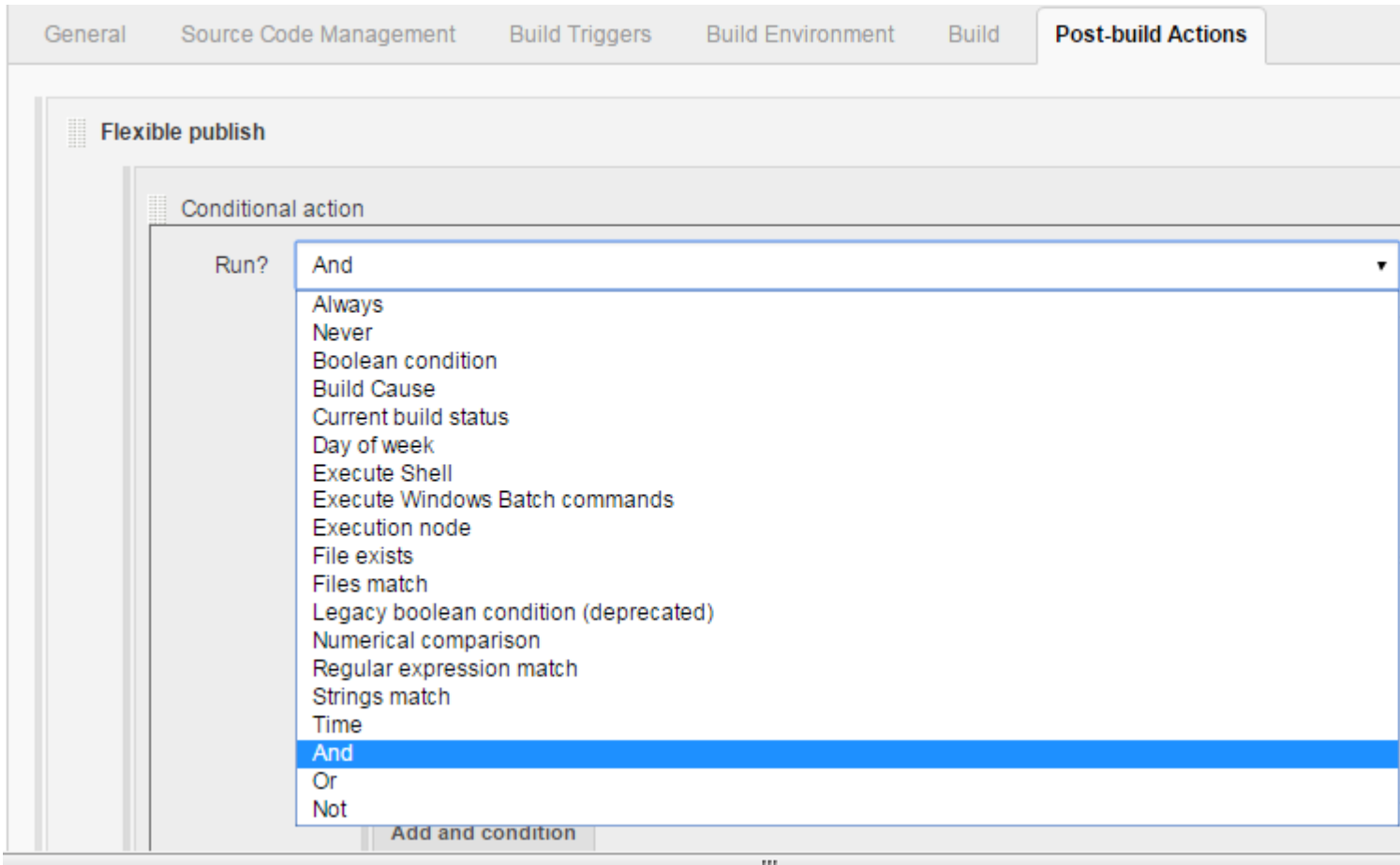
Build

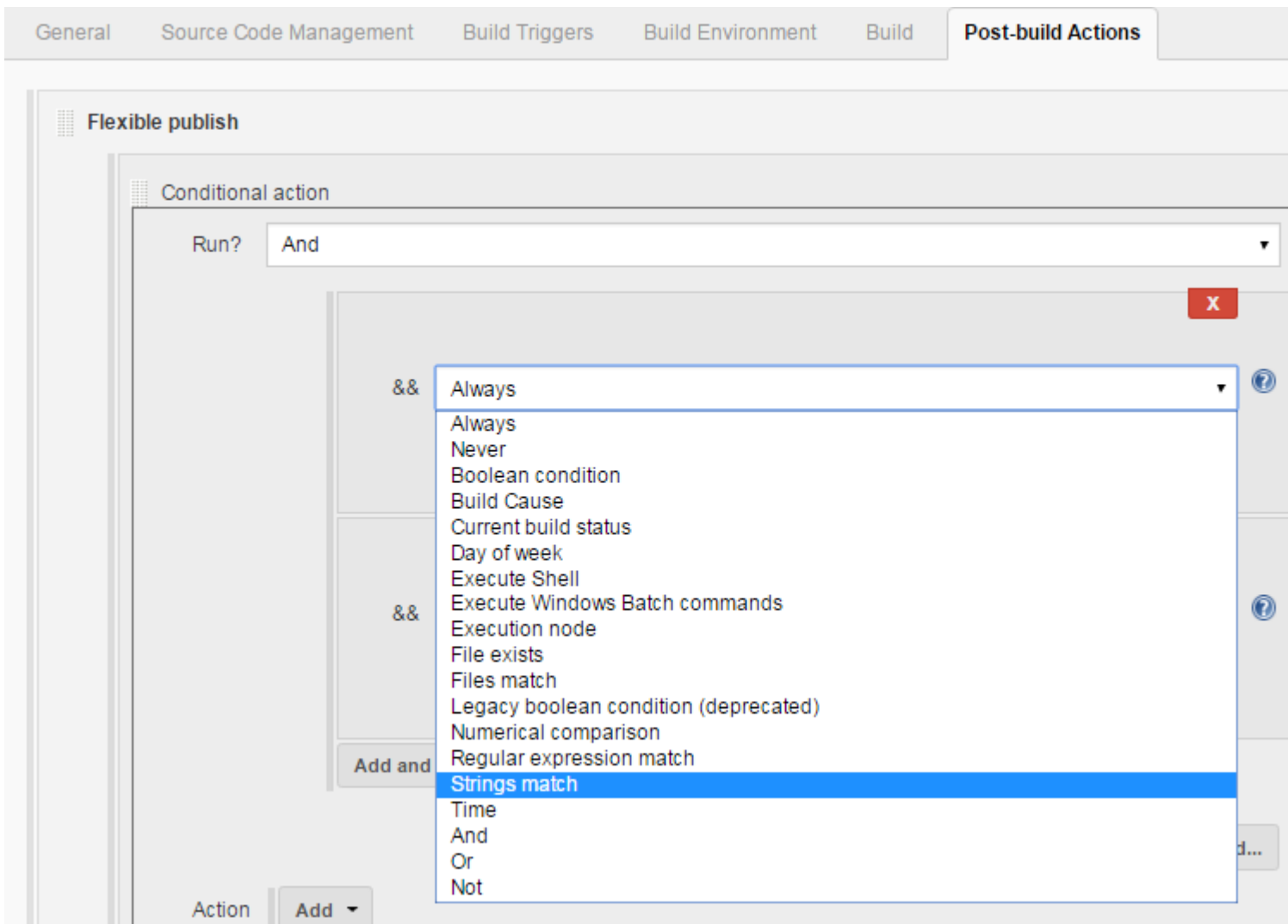
Execute shell

Command

See [the list of available environment variables](#)

6. Ahora, en la sección "Acciones posteriores a la creación", seleccione el complemento "Publicación flexible". Seleccione el valor "Y" en el menú desplegable para Acción condicional (¿Ejecutar?). Luego seleccione "Coincidencia de cadenas" en el menú desplegable para la Condición de ejecución (&&).





7. Después de seleccionar la coincidencia de cadena, especifique \$ PUSH como valor de Cadena 1 y SÍ como valor de Cadena 2. Entonces, cuando ejecute la compilación si elige el valor de PUSH como SÍ, comparará la Cadena 1 (= \$ PUSH) y la Cadena 2 (= SÍ) y activará la operación de empuje de Git y si elige NO, no lo hará. activar la operación de empuje Git.

```
Choose the value of PUSH -> YES OR NO -> Chosen value "YES"
then, $PUSH = YES
AS String 1 = $PUSH => String 1 = YES
Again, String 2 = YES, hence String 2 == String 1 (String match)
Then, trigger the Git push action.
```

General Source Code Management Build Triggers Build Environment Build **Post-build Actions**

Run? And

String match

String 1 \$PUSH

String 2 YES

Case insensitive ☐

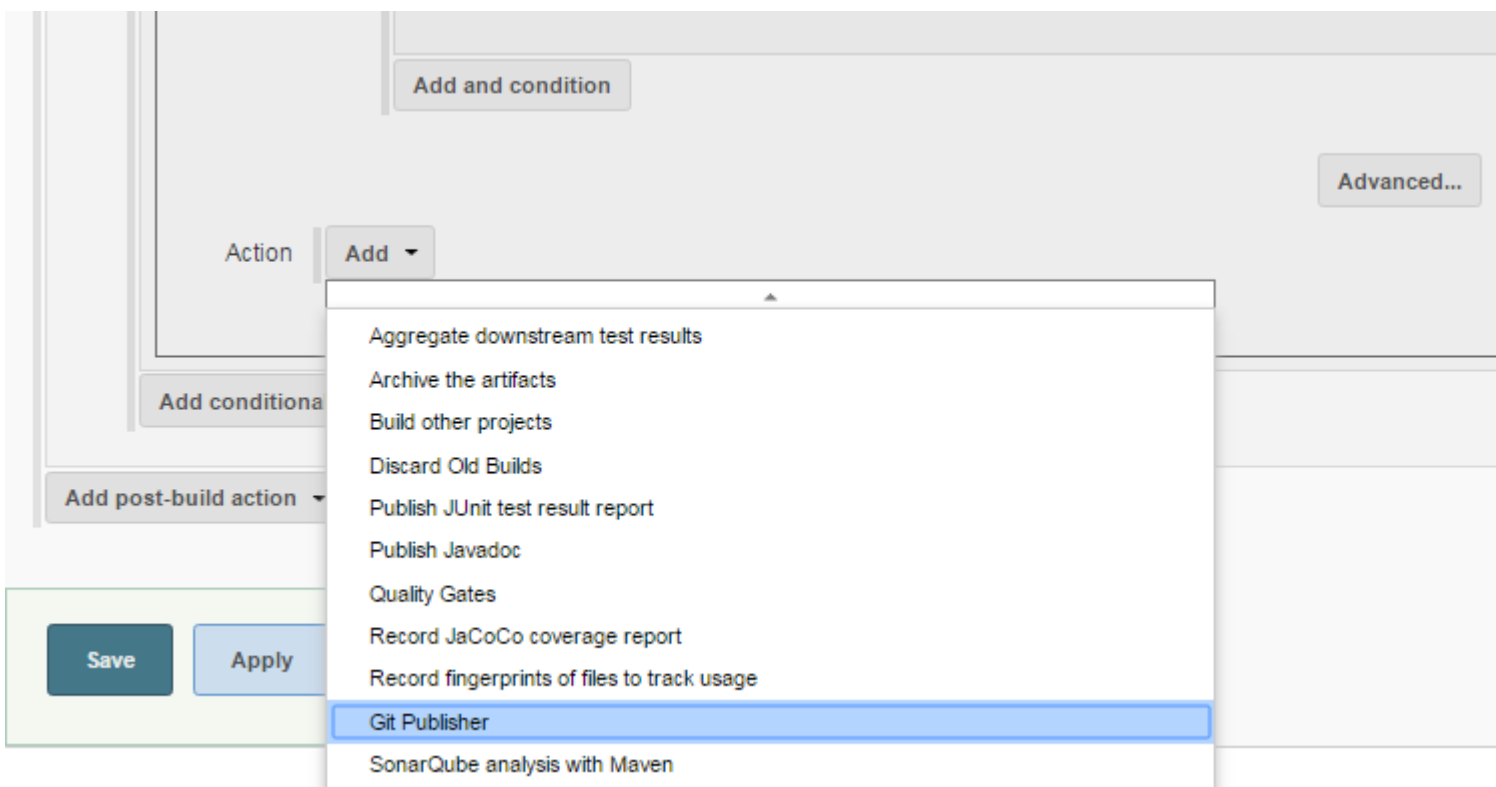
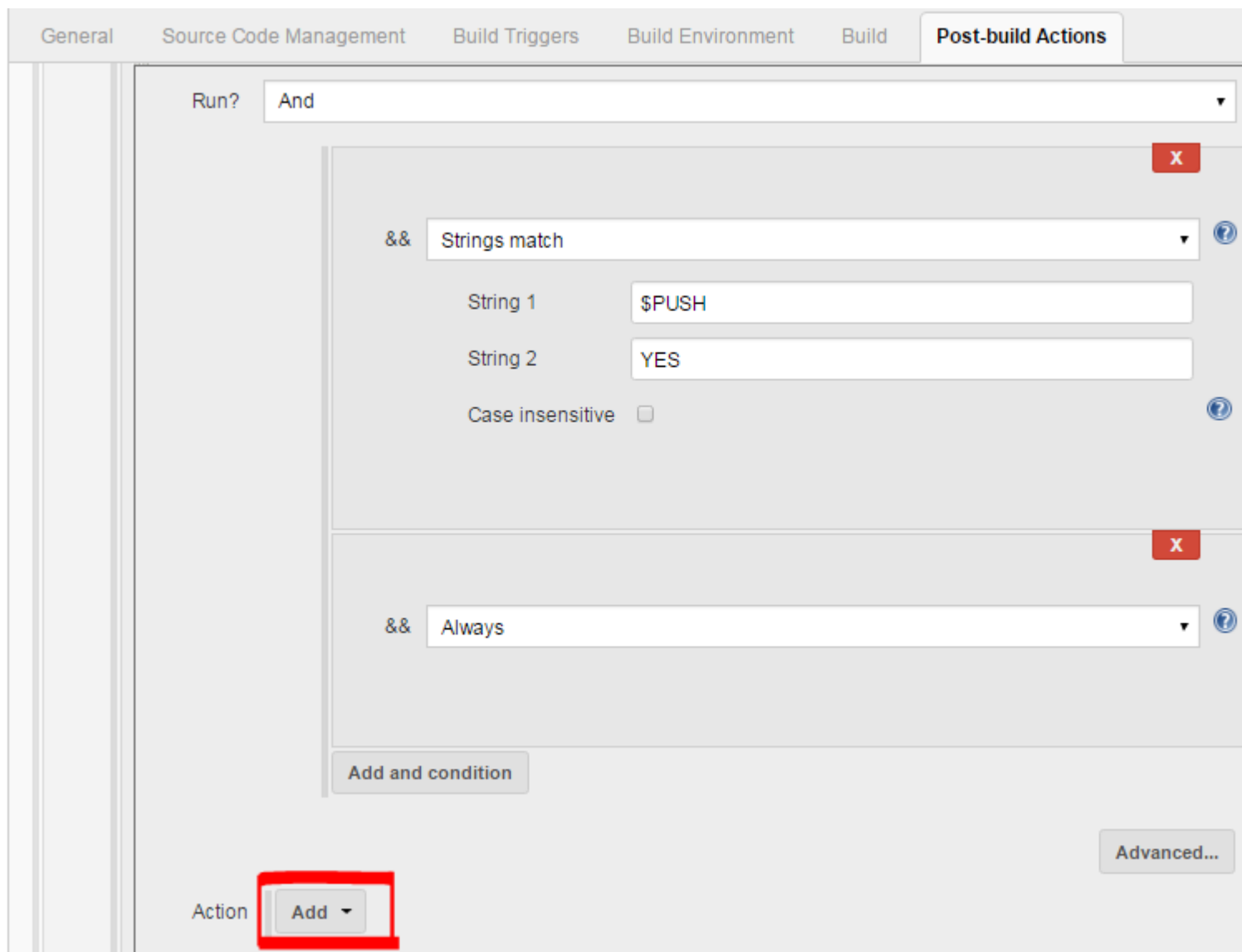
Always

Add and condition

Advanced...

Action Add

8. Ahora haga clic en Agregar opción desplegable para agregar la acción del editor Git que se activará en función de la condición de coincidencia de cadena.



9. Después de seleccionar Git Publisher, realice la configuración de la siguiente manera:

Action

- ☒ Push Only If Build Succeeds
- ☐ Merge Results
- ☒ Force Push

If pre-build merging is configured, push the result to origin

Add force option to git push

Tags

Conditional action

- Tag to push:
- Tag message:
- Create new tag: ☒
- Update new tag: ☒
- Target remote name:

Add Tag

Branches

Conditional action

- Branch to push:
- Target remote name:

Add Branch

Save Apply

Después de la configuración, guarda el trabajo y listo.

Lea [Configurar Auto Git Push en la construcción exitosa en Jenkins en línea](https://riptutorial.com/es/jenkins/topic/8972/configurar-auto-git-push-en-la-construccion-exitosa-en-jenkins):

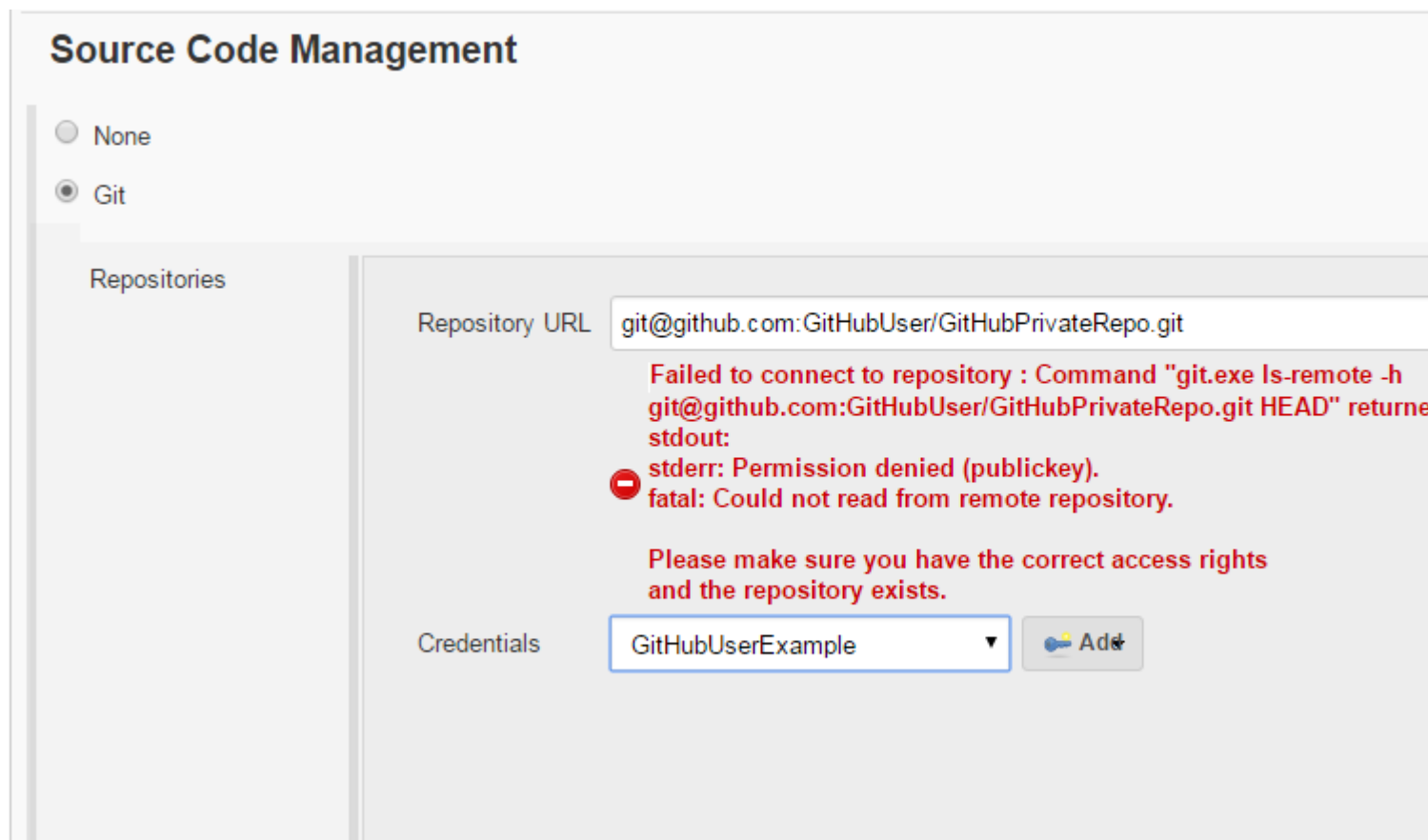
<https://riptutorial.com/es/jenkins/topic/8972/configurar-auto-git-push-en-la-construccion-exitosa-en-jenkins>

Capítulo 6: Instale Jenkins en Windows con soporte SSH para repositorios privados de GitHub

Examples

Las solicitudes de extracción de GitHub fallan

Las instalaciones fuera de la caja de Jenkins con los complementos Git y SSH no funcionarán al intentar extraer un repositorio privado de GitHub.



PSEXec.exe PS Tool de Microsoft

El primer paso para solucionar este problema que encontré fue descargar [PSTools](#) y extraer las herramientas en una ubicación conveniente en el servidor de compilación (p. Ej., C: \ Programs \ PSTools extraí el mío).

Genere una nueva clave SSH solo para Jenkins usando PSEXec o PSEXec64

1. Primero abra el símbolo del sistema y "Ejecutar como administrador".
2. Una vez que el símbolo del sistema esté abierto, navegue al directorio de PSTools.
3. Desde el símbolo del sistema, necesitamos ejecutar git-bash utilizando PSEXec o PSEXec64

como el Servicio local, que Jenkins ejecuta en el servidor de compilación de forma predeterminada.

4. Usaremos el modificador -i para ejecutar PSEXec como interactivo y el modificador -s para ejecutar git-bash como el servicio local
5. Siga las instrucciones para crear una clave ssh en GitHub: [generar una nueva clave SSH y agregarla a ssh-agent](#)
6. Si está en un sistema Windows de 64 bits, copie la carpeta .ssh en C: \ Windows \ SysWOW64 \ config \ systemprofile.ssh (esto no era necesario en mi sistema Windows de 64 bits, pero allí había algunas instrucciones que indicaban los archivos .ssh debe guardarse allí, algo a tener en cuenta si todavía tiene problemas).
7. Agregue la clave pública SSH a sus claves github.

Your Commandline should look similar to this:

```
C:\Programs\PSTools> PSEXec.exe -i -s C:\Programs\Git\git-bash
```

Crear las credenciales de Jenkins

¡La parte difícil ya pasó! Ahora solo crea las credenciales que se utilizarán en Jenkins. Use su propio nombre de usuario y la contraseña utilizada para crear la clave SSH.



Jenkins Credentials Provider: Jenkins



Add Credentials

Domain

Kind

Scope

Username

Private Key ☐ Enter directly
☐ From a file on Jenkins master
☒ From the Jenkins master ~/.ssh

Passphrase

ID

Description

Add

Cancel

Este es el aspecto que debería tener ahora (con su propio repositorio privado de Github y nombre de usuario:

Source Code Management

☐ None

☒ Git

Repositories

Repository URL

Credentials [Add](#)

Ejecute una solicitud de extracción de prueba para verificar, y listo.

Guarde y ejecute una solicitud de extracción de prueba y ya no tendrá más problemas para que Jenkins use SSH en su máquina de compilación de Windows.



Jenkins

[Jenkins](#)

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now](#)

[Delete Project](#)

[Configure](#)

[Move](#)

[Workspace](#)

[Recent Changes](#)

Permalinks

- [Last build \(#1\). 47 min ago](#)
- [Last stable build \(#1\). 47 min ago](#)
- [Last successful build \(#1\). 47 min ago](#)
- [Last completed build \(#1\). 47 min ago](#)

Build History

[trend](#)

x

 **#1** Oct 25, 2016 10:31 AM

[RSS for all](#) [RSS for failures](#)

Lea Instale Jenkins en Windows con soporte SSH para repositorios privados de GitHub en línea:
<https://riptutorial.com/es/jenkins/topic/7626/instale-jenkins-en-windows-con-soporte-ssh-para-repositorios-privados-de-github>

Capítulo 7: Jenkins Groovy Scripting

Examples

Crear usuario predeterminado

1. Cree un archivo maravilloso por la ruta `$JENKINS_HOME/init.groovy.d/basic-security.groovy`

En el directorio de inicio de Jenkins de Ubuntu 16 lugares en `/var/lib/jenkins`

2. Colocar en el archivo siguiente código

```
#!/groovy

import jenkins.model.*
import hudson.security.*

def instance = Jenkins.getInstance()

def hudsonRealm = new HudsonPrivateSecurityRealm(false)

hudsonRealm.createAccount("admin_name", "admin_password")
instance.setSecurityRealm(hudsonRealm)
instance.save()
```

3. Reinicie el servicio Jenkins
4. Después de que Jenkins comience, debe eliminar el `$JENKINS_HOME/init.groovy.d/basic-security.groovy`

Deshabilitar el asistente de configuración

1. Abrir el archivo de configuración por defecto Jenkins y poner en `JAVA_ARGS` siguiente tecla - `Djenkins.install.runSetupWizard=false`

En Ubuntu 16, el archivo predeterminado se coloca en `/etc/default/jenkins`

2. Cree un archivo maravilloso por la ruta `$JENKINS_HOME/init.groovy.d/basic-security.groovy`

En el directorio de inicio de Jenkins de Ubuntu 16 lugares en `/var/lib/jenkins`

3. Colocar en el archivo siguiente código

```
#!/groovy

import jenkins.model.*
import hudson.util.*;
import jenkins.install.*;

def instance = Jenkins.getInstance()

instance.setInstallState(InstallState.INITIAL_SETUP_COMPLETED)
```


4. Reinicie el servicio Jenkins

5. Después de que Jenkins comience, necesita eliminar el `$JENKINS_HOME/init.groovy.d/basic-security.groovy`

Después de esto, Jenkins no le pedirá que confirme que es administrador y no verá la página de instalación de complementos.

Cómo obtener información sobre la instancia de Jenkins.

Abra su consola de script de instancia de jenkins <http://yourJenkins:port/script> siguiente es un ejemplo de cómo obtener información sobre esta instancia. Copie el código a la consola y haga clic en "Ejecutar".

```
/* This scripts shows how to get basic information about Jenkins instance */
def jenkins = Jenkins.getInstance()
println "Jenkins version: ${jenkins.getVersion()}"
println "Available JDKs: ${jenkins.getInstance().getJDKs()}"
println "Connected Nodes:"
jenkins.getNodes().each{
    println it.displayName
}
println "Configured labels: ${jenkins.getLabels()}"
```

En este ejemplo, verá información sobre la versión de Jenkins, JDK, agentes (esclavos) y etiquetas.

Cómo obtener información sobre un trabajo de Jenkins

Abra su consola de scripts de instancia de jenkins <http://yourJenkins:port/script> siguiente es un ejemplo de cómo obtener información sobre un trabajo específico. copie el código en la consola, cambie el nombre del trabajo al trabajo requerido y haga clic en "Ejecutar".

```
/*This script shows how to get basic information about a job and its builds*/
def jenkins = Jenkins.getInstance()
def jobName = "myJob"
def job = jenkins.getItem(jobName)

println "Job type: ${job.getClass()}"
println "Is building: ${job.isBuilding()}"
println "Is in queue: ${job.isInQueue()}"
println "Last successfull build: ${job.getLastSuccessfulBuild()}"
println "Last failed build: ${job.getLastFailedBuild()}"
println "Last build: ${job.getLastBuild()}"
println "All builds: ${job.getBuilds().collect{ it.getNumber()}}"
```

primero obtenemos el objeto de instancia de Jenkins, luego, utilizando esta instancia obtenemos el objeto de trabajo (elemento). desde el objeto de trabajo podemos obtener información diferente, por ejemplo: ¿se está construyendo actualmente, está en la cola, la última compilación, la última compilación por estado y mucho más?

Lea Jenkins Groovy Scripting en línea: <https://riptutorial.com/es/jenkins/topic/7562/jenkins-groovy-scripting>

Creditos

S. No	Capítulos	Contributors
1	Empezando con Jenkins	acalb , Community , Gautam Jose , Girish Kumar , Katu , Pablo , Pom12 , Priyanshu Shekhar , Rogério Peixoto , S.K. Venkat , Seb , Tyler
2	Complemento de estrategia de rol	Gautam Jose
3	Configuración de Build Automation para iOS utilizando Shenzhen	Ichthyocentaurs
4	Configuración de Jenkins para la automatización de la compilación de iOS.	Manuel Morejón
5	Configurar Auto Git Push en la construcción exitosa en Jenkins	ANIL
6	Instale Jenkins en Windows con soporte SSH para repositorios privados de GitHub	Riana
7	Jenkins Groovy Scripting	RejeeshChandran , seriezny , Tidhar Klein Orbach