Tecnicatura Universitaria en Programación Informática

Jenkins

Integración continua

Laboratorio de Redes y Sistemas Operativos

ALUMNOS:

- ❏ Avantaggiato, Mateo Andres.
- ❏ Castro, Alejandro David.
- ❏ Loiacono, Pablo.

PROFESOR: Di Biase, José Luis**.**

FECHA: 2013 segundo cuatrimestre.

Jenkins en Ubuntu 12.04

Jenkins es una aplicación para realizar integración continua de proyectos de software. Derivado del desarrollo original Hudson. El mismo está implementado en Java por lo que se necesita una máquina virtual en el sistema para poder utilizarlo. Soporta herramientas de control de versiones como CVS, Subversion, Git, Mercurial, Perforce y Clearcase y puede ejecutar proyectos basados en Apache Ant y Apache Maven.

Principalmente es utilizado para proyectos escritos en Java, pero permite también otros lenguajes como Python o Ruby, simplemente con la instalación de sus respectivos plugins.

A continuación se explicara como instalar y configurar el servidor para administrar un proyecto Java. Se procura hacer todo mediante consola para emular un servidor.

Instalación

Para poder desplegar la aplicación se requiere Java 7 (se recomienda openjdk-7-jre and openjdk-7-jdk)

```
sudo apt-get install openjdk-7-jre
sudo apt-get install openjdk-7-jdk
```

En la página de Jenkins podemos observar cómo agregar el repositorio e instalar la aplicación
https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Ubuntu

```
wget -q -O - http://pkg.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
```

Durante la instalación se produjo un error relacionado con permisos del sistema ya contemplado en el sitio de Jenkins.
https://issues.jenkins-ci.org/browse/JENKINS-20407

```
sudo mkdir /var/run/jenkins
sudo apt-get install jenkins
```

Para configurarlo debemos editar el archivo “/etc/default/jenkins”

Por defecto funciona en el puerto 8080.

La URL http://localhost:8080/ nos permite utilizar la vista web de lla aplicación.
El servicio correspondiente a Jenkins se encuentra en “/etc/init.d/”.
Para resetear el servicio debemos ejecutar.

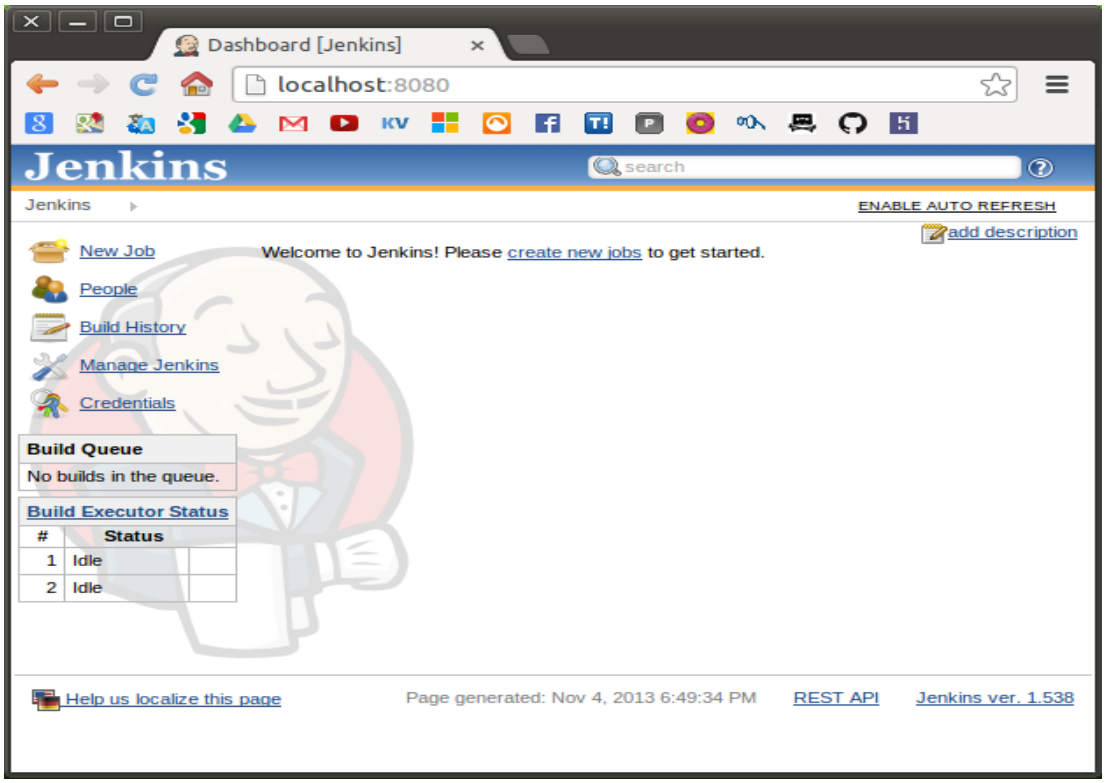
```
sudo /etc/init.d/jenkins restart
```

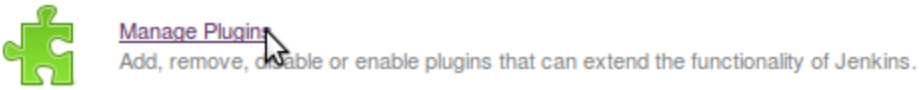
Estando la aplicación alojada en un repositorio local procederemos a instalar los correspondientes plugins de git.

Desde la web de jenkins.
Nos dirigimos al link [Administrar Jenkins] y [Administrar Plugins]

Y seleccionamos los siguientes plugins para instalar.

GitHub API
Git Client
Git Plugin
Git Server
SSH credentials





Configurar Jenkins con Git

Instalamos git y creamos un proyecto

```
sudo apt-get install git
```

Sobre la carpeta del proyecto

```
git init
```

Podemos agregar un documento con el nombre ".gitignore" para ignorar archivos durante la revisión. Dentro podemos encontrar las siguientes líneas.

```
.metadata/*
RemoteSystemsTempFiles/*
*~
projectname/.settings/*
projectname/target/*
projectname/.classpath
projectname/.project
```

Instalamos y configuramos las claves de ssh con especial cuidado de no olvidar la clave.

```
sudo apt-get install openssh-server
ssh-keygen -t rsa -b 2048 -f ~/.ssh/id_rsa
pass: ****
```

Una vez instalado esto se podrá acceder al repositorio por medio de ssh.

```
git ls-remote -h git@url:project-path
```

Pero Jenkins aún no puede conectarse al repositorio porque no posee las credenciales ssh de ese usuario.
Para solucionarlo utilizaremos el plugin SSH de jenkins para otorgarle acceso permanente.

Se deben crear claves SSH para el usuario de Jenkins.

```
sudo su - jenkins
ssh-keygen -t rsa -b 2048
```

Para poder acceder a los repositorio de forma consistente creamos un usuario con ese nombre.

```
sudo adduser git #seleccionar una contraseña
su git
mkdir .ssh
```

Agregamos la clave pública de jenkins al usuario git

```
sudo su - jenkins
scp .ssh/id_rsa.pub git@url:.ssh/authorized_keys
```

SCP es un comando de ssh que facilita el clonado de claves.
Cada vez que agreguemos una nueva clave deberemos seleccionar un nombre que no exista aún para no sobreescribir otra clase.
Para agregar a Jenkins la dirección del servidor como host conocido basta con consultar los remotos de algún repositorio

```
sudo su - jenkins
git ls-remote -h git@127.0.0.1:/home/usuario/workspaces/labo/
```

En este punto se puede crear una nuevo proyecto en Jenkins, seleccionar git como origen de código (no github) y asignar la ruta al repositorio

```
git@127.0.0.1:/home/usuario/workspaces/project-folder/
```

Configurar un proyecto Maven

Por ser un proyecto Java que utiliza Maven, se debe contar con la aplicación Maven en el sistema operativo donde se instaló Jenkins y agregar la ruta del ejecutable dentro de su configuración.

Posiblemente a esta altura se esté mostrando un error, al momento de construir el proyecto, relacionado con la falta de un paquete de Java llamado “tools.jar”. Esto se debe a que debemos indicarle a Jenkins donde esta ubicado la JDK que va a utilizar, y lo mismo para la versión de Maven.

Para resolver esto vamos a :
Administrar jenkins -> configurar sistema -> Maven -> instalación de maven.

Aquí vamos a indicar la ruta donde se encuentra la carpeta con el ejecutable de Maven a utilizar (en este caso 2.2.1)

```
/usr/share/maven2/
```

Especificar la versión de Java para compilar:

En caso de ser necesario especificar la versión Java utilizada para construir el proyecto podemos agregar el siguiente contenido al pom de maven

```
<build>
  [...]
  <plugins>
     <plugin>
       <groupId>org.apache.maven.plugins</groupId>
       <artifactId>maven-compiler-plugin</artifactId>
       <version>3.1</version>
       <configuration>
         <source>1.4</source>
         <target>1.4</target>
       </configuration>
     </plugin>
  </plugins>
  [...]
</build>
```

Instalar y configurar plugins

A continuación detallaremos cómo administrar las opciones de un plugin para un determinado proyecto, para esto utilizamos Jenkins-Cobertura-Plugin

Se deben llevar a cabo 3 acciones en el caso de Cobertura.

1) Instalar el plugin desde la página de Jenkins como se indicó con los anteriores.

2) Incluir en el archivo pom.xml el siguiente código para especificarle a maven que genere los correspondientes reportes.

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>cobertura-maven-plugin</artifactId>
  <version>2.5.1</version>
  <configuration>
    <formats>
      <format>xml</format>
    </formats>
  </configuration>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>cobertura</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

3) Ir a la configuración del proyecto, y dentro de la pestaña “Acciones para ejecutar después” seleccionar “añadir una acción” y dentro del menú desplegable la opción “Publicar informes de Cobertura”. Dentro del formulario que se despliega agregar la ruta de los informes generados para que Jenkins pueda mostrarlos.

```
**/target/site/cobertura/coverage.xml
```

Una vez realizados todos estos pasos se podrá observar el resultado de los informes en la pantalla de cada build del proyecto realizados posteriormente.