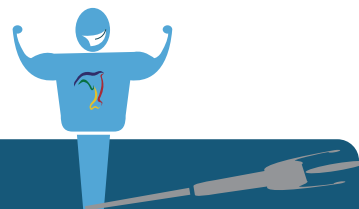


¿Qué ofrece Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

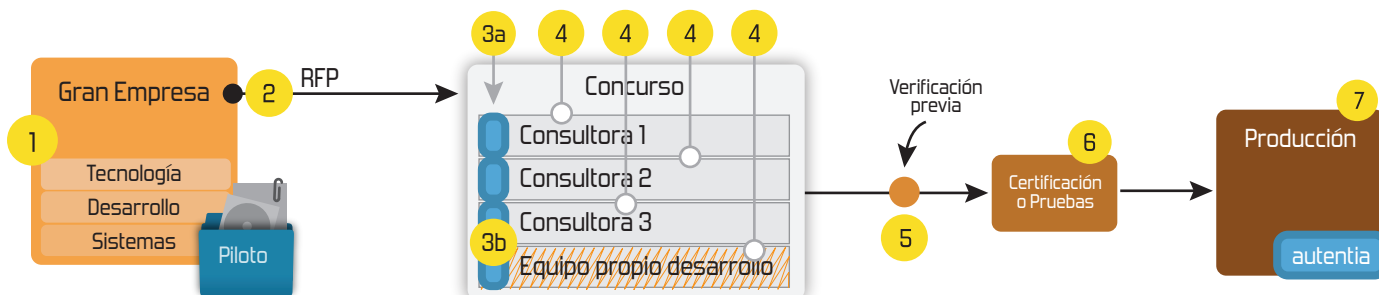
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces,
HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

Control de autenticación y
acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)



Entra en Adictos a través de

E-mail

Contraseña

[Entrar](#) [Registrarme](#) [Olvidé mi contraseña](#)

[Inicio](#) [Quiénes somos](#) [Formación](#) [Comparador de salarios](#) [Nuestros libros](#) [Más](#)

» Estás en: [Inicio](#) [Tutoriales](#) [Crear servidor propio de Git en CentOS 6.5](#)



Rubén Aguilera Díaz-Heredero

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero en Informática, especialidad en Ingeniería del Software

Puedes encontrarme en [Autentia](#): Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE



[Ver todos los tutoriales del autor](#)

Catálogo de servicios Autentia



Fecha de publicación del tutorial: 2014-07-11

Tutorial visitado 1 veces [Descargar en PDF](#)

Crear servidor propio de Git en CentOS 6.5

0. Índice de contenidos.

- 1. Entorno
- 2. Introducción
- 3. Vamos al lío
- 4. Conclusiones

1. Entorno

Este tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil Mac Book Pro 17" (2,6 Ghz Intel Core i7, 8 GB DDR3)
- Sistema Operativo: Mac OS X Snow Leopard 10.6.4
- Máquina virtual con CentOS 6.5

2. Introducción

Git es un sistema de control de versiones distribuido del que ya hemos hablado en distintas ocasiones en el portal, por lo que si no lo conoces te recomiendo las siguientes lecturas:

- [Git y cómo trabajar con un repositorio de código distribuido](#)
- [Primeros pasos con github: subir un proyecto al repositorio](#)
- [Trabajando con GIT, introducción al uso de los branch y git-completion.bash](#)

Lo más habitual cuando se trabaja con Git es crearnos una cuenta en algún servicio como Github o BitBucket que nos proporcionan un servidor de Git con el que podemos trabajar y subir nuestros proyectos personales o privados.

Pero a veces nos encontramos con clientes que por motivos de seguridad no quieren tener el código de sus proyectos en la "nube" ni aún pagando un servicio privado.

Es por ese motivo que os voy a enseñar como se crea un servidor propio de Git con la desventaja de que el backup de los datos y la gestión de los usuarios tendremos que hacerla nosotros.

3. Vamos al lío

Lo primero que necesitamos es una instancia de Linux, en nuestro caso lo vamos a hacer en CentOS aunque en otras distribuciones como Ubuntu los pasos son muy similares.

Una vez tenemos acceso a la máquina Linux bien sea por SSH o en el propio terminal de la máquina, con un usuario con permisos, tenemos que instalar una instancia de Git.

Para hacerlo en CentOS previamente tenemos que instalar una serie de dependencias para lo que ejecutamos:

```
01. $ yum -y install zlib-devel openssl-devel cpio expat-devel gettext-devel gcc perl-ExtUtils-MakeMaker
```

Seguidamente instalamos el paquete de Git de este modo:

```
01. $ yum install git
```

Para toda la gestión con Git vamos a crear un usuario específico al que vamos a llamar "git", en un alarde de originalidad sin precedentes :-)

Para ello desde un terminal ejecutamos:



Síguenos a través de:



Últimas Noticias

» [Screencasts de programación narrados en Español](#)

» [Sorteo de entradas para APIdays Mediterranea](#)

» [Concurso del Día de la Madre:](#)

» [Aprende gratis ReactiveCocoa](#)

» [Checklist de Scrum de Autentia](#)

[Histórico de noticias](#)

Últimos Tutoriales

» [Primeros pasos con Neo4j](#)

» [Introducción a WSO2 API Manager](#)

» [Introducción a Groovy y Grails con Maven: el patrón CRUD](#)

» [Menu.bat Una forma cómoda de ejecutar comandos y aplicaciones, por ejemplo para Maven](#)

» [Testing de Hadoop con MRUnit](#)

```
view plain print ?
01. $> sudo adduser git
```

Y le damos una password, por ejemplo "git", con el comando:

```
view plain print ?
01. $> sudo passwd git
```

Ahora nos logamos con el nuevo usuario "git":

```
view plain print ?
01. $> su git
```

Nos solicita la password. Se la introducimos correctamente y nos posicionamos en su home:

```
view plain print ?
01. $> cd ~
```

Dentro del home del usuario "git" vamos a crear la carpeta de proyectos "projects" donde vamos a almacenar nuestros proyectos:

```
view plain print ?
01. $> mkdir projects
```

Nos situamos en este directorio:

```
view plain print ?
01. $> cd projects
```

Ahora vamos a crear nuestro primer proyecto al que vamos a llamar "autentia-test.git". Para ello creamos el directorio

```
view plain print ?
01. $> mkdir autentia-test.git
```

Y dentro de este directorio vamos a crear el proyecto de Git con el comando:

```
view plain print ?
01. $> git --bare init
```

Si todo es correcto nos encontraremos con este mensaje: "Initialized empty Git repository in /home/git/projects/autentia-test.git"

Para probar el repositorio, desde la misma máquina o desde otra máquina cliente, podemos hacer un clone del proyecto con el comando:

```
view plain print ?
01. $> git clone git@ip_maquina_git:~/projects/autentia-test.git
02. Cloning into 'autentia-test'...
```

En este punto solicita la password del usuario git de la máquina. Una vez introducida se descarga el proyecto en nuestra máquina cliente y ya podemos trabajar como en cualquier otro repositorio de Git.

Es bastante molesto tener que introducir la password cada vez que hagamos push en el proyecto o queramos clonar otros proyectos. Para evitar esto vamos a establecer un enlace de confianza con llave pública y privada a través de SSH entre la máquina del servidor Git y nuestra máquina cliente.

Para ello desde el servidor vamos a habilitar un directorio donde los clientes nos van a dejar su clave pública.

```
view plain print ?
01. $> mkdir /home/git/clients
```

Ahora creamos el directorio .ssh dentro de la carpeta /home/git del servidor. Es muy importante que está carpeta solo tenga permisos 700 y pertenezca exclusivamente al usuario git.

```
view plain print ?
01. $> mkdir /home/git/.ssh
02. $> chmod 700 .ssh
```

Dentro de la carpeta .ssh vamos a crear el fichero authorized_keys donde vamos a almacenar todas las claves públicas de los clientes que se quieran conectar a nuestro repositorio de Git. Es muy importante que este fichero pertenezca al usuario git y que los permisos estén seteados a 600.

```
view plain print ?
01. $> cd .ssh
02. $> touch authorized_keys
03. $> chmod 600 authorized_keys
```

El cliente tiene que crear su par de clave pública y clave privada. Para ello simplemente tiene que ejecutar en un terminal:

```
view plain print ?
01. $> ssh-keygen -t rsa
```

Este comando por defecto creará dos ficheros en el directorio .ssh de la home del usuario de la máquina cliente; un fichero con la clave privada (id_rsa) y otro fichero con la clave pública (id_rsa.pub). Es muy importante que nuestra clave privada no la compartamos con nadie en ninguna circunstancia.

Últimos Tutoriales del Autor

» [Crear un plugin para Android en PhoneGap](#)

» [Intercomunicación de aplicaciones en IOS](#)

» [Crashlytics en IOS](#)

» [Acceso a la cámara con PhoneGap](#)

» [Empezando con PhoneGap](#)

El siguiente paso es transmitir a la carpeta "clients" del servidor de Git la clave pública, de esta forma:

```
view plain print ?
01. $> scp /home/username/.ssh/id_rsa.pub git@ip_maquina_git:~/clients/username_rsa.pub
```

Es una buena práctica almacenar en la carpeta "clients" el fichero con el nombre del usuario para que no se sobrescriba con alguno existente. Ahora, dentro del servidor de Git, pasamos el contenido del fichero al archivo "authorized_keys".

```
view plain print ?
01. $> cat /home/git/clients/username_rsa.pub >> .ssh/authorized_keys
```

Desde este momento cuando nuestro cliente vaya a interactuar con el servidor de Git ya no le pedirá la password.

4. Conclusiones

Como véis no es muy complicado tener nuestro propio repositorio de Git, por lo que no tenemos que depender de servicios como GitHub o BitBucket y nuestros clientes más "recelosos" ya no tienen excusa para no usar Git.

Cualquier duda o sugerencia en la zona de comentarios.

Saludos.

A continuación puedes evaluarlo:

[Regístrate para evaluarlo](#)

Por favor, vota +1 o compártelo si te pareció interesante

Share |

8+1 0

Anímate y coméntanos lo que pienses sobre este **TUTORIAL**:

» **Regístrate** y accede a esta y otras ventajas «



Esta obra está licenciada bajo licencia [Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)

IMPULSA

Impulsores

Comunidad

¿Ayuda?

sin clicks

0 personas han traído clicks a esta página

+ + + + + + + +

powered by [karmacracy](#)

Copyright 2003-2014 © All Rights Reserved | [Texto legal y condiciones de uso](#) | [Banners](#) | [Powered by Autentia](#) | [Contacto](#)

[W3C XHTML 1.0](#) [W3C CSS](#) [XML RSS](#) [XML RTOH](#)