

¿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)

E-mail: Contraseña:

Deseo registrarme

He olvidado mis datos de acceso

[Inicio](#) [Quiénes somos](#) [Tutoriales](#) [Formación](#) [Comparador de salarios](#) [Nuestro libro](#)

[Charlas](#) [Más](#)

✂ Estás en:

[Inicio](#) [Tutoriales](#) [Publicar un repositorio Mercurial con Apache](#)



DESARROLLADO POR:

Ángel García Jerez



Consultor tecnológico de desarrollo de proyectos informáticos. Co-autor del libro "Actualización y mantenimiento del PC (Edición de 2010) publicado por Anaya Multimedia

Ingeniero Técnico en Informática de Sistemas e Ingeniero en Informática (premio al mejor expediente de su promoción)

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

Somos expertos en Java/J2EE

Catálogo de servicios Autentia



Últimas Noticias

Mi primer coderetreat, Chispas!!!

Entregamos nuestro primer diploma ...

Comic Flash de Head Hunting

XI Charla Autentia - Mule - Recordatorio

Comparte el Conocimiento en Adictos

[Anuncios Google](#)

[Apache](#)

[Mercurial](#)

[Apache Tomcat Host](#)

[Instal](#)

Fecha de publicación del tutorial: 2009-02-26



Share |

[Regístrate para votar](#)

PUBLICAR UN REPOSITORIO MERCURIAL CON APACHE

1. Introducción
2. Entorno
3. Instalación de Mercurial
4. Instalación del módulo modwsgi para Apache
5. Configurando Apache 2.x con el módulo modwsgi
6. Retocando el script hgwebdir.wsgi.



7. Configurando Mercurial.
8. Publicando un repositorio .
9. Restringiendo el acceso al repositorio.
10. Activando el push sobre el repositorio..
11. Conclusiones.

[Histórico de NOTICIAS](#)

Últimos Tutoriales

1. Introducción

En el primer [tutorial](#) sobre Mercurial os enseñamos las características y pasos básicos para empezar a trabajar con él. Ahora os vamos a enseñar los pasos necesarios para que los repositorios sean accesibles a través de Apache.

Mercurial soporta diferentes protocolos para acceder a un repositorio (http, https, ssh, file y local), pero cuando queremos publicar un repositorio para que sea accesible remotamente, se reducen las posibilidades. De todas ellas, https es el más utilizado y el que recomienda Mercurial en su documentación.

En la página oficial existe una [tabla](#) en la que se describen todas las posibles soluciones para publicar un repositorio. En nuestro caso, para el protocolo https, podremos utilizar:

- **hg server bajo un proxy Nginx:** esta solución utiliza el servidor que viene integrado con Mercurial. Como éste sólo soporta el protocolo http, será necesario utilizar un proxy para utilizar https. Con esta solución vamos a poder publicar varios repositorios. Por defecto estará habilitada la posibilidad de hacer push, permite autenticación y no utiliza CGI.
- **hgweb:** es una solución que se integra con un servidor web existente utilizando python, CGI o WSGI. Se integra con la autenticación del servidor, pero sólo será capaz de publicar un único repositorio y por defecto no estará habilitado el uso de push.
- **hgwebdir:** se trata de la misma solución que hgweb pero con la única diferencia de que vamos a poder publicar más de un repositorio.






De todas las soluciones, la más adecuada es hgwebdir ya que nos da la posibilidad de publicar varios repositorios y su ejecución estará bajo un servidor web como va a ser Apache. La solución hg server sólo la deberíamos utilizar para situaciones temporales y no como un servidor permanente.

Como único requisito de este tutorial daremos por hecho que disponemos de un servidor web Apache con el protocolo https configurado.

2. Entorno

Entorno utilizado para escribir este tutorial:

- **Hardware:** VirtualBox 3.2.8 corriendo sobre un Mac Book Pro (Core 2 Duo 2,8 Ghz, 4 GB RAM, 500 GB)

-  [Liferay IDE](#)
-  [Rendimiento en espacio y transferencia de un servidor Subversion](#)
-  [Liquibase- Incorporación del histórico de cambios en una BBDD existente](#)
-  [Cómo subir tutoriales a Adictos](#)
-  [Spring + REST + JSON = SOAUI](#)

Últimos Tutoriales del Autor

-  [Mercurial, un sistema de control de versiones distribuido](#)
-  [MediaWiki - NamespacePermissions](#)
-  [Facebook Social Plugins](#)
-  [Desplegando en Tomcat un proyecto web con Hudson.](#)
-  [DBUnit y aplicaciones JDBC](#)

Síguenos a través de:

- **Sistema Operativo:** Ubuntu 64 bits 10.04 LTS
- **Mercurial:** 1.6.3
- **Python:** 2.6.1
- **Wsgi:** 2.8.2



Últimas ofertas de empleo

3. Instalación de Mercurial

Lo primero que tenemos que hacer es instalar Mercurial y la librería de python. Para ello ejecutaremos el siguiente comando:

```
01. | apt-get install mercurial python python-dev
```

A continuación debemos crear el directorio donde vamos a almacenar los repositorios y demás ficheros de configuración:

```
01. | mkdir -p /var/hg/repositories
02. | mkdir -p /var/hg/cgi-bin
03. | touch /var/hg/hgweb.config
```

Ahora copiamos el script de hgwebdir que posteriormente utilizaremos en Apache. Este fichero se encuentra en uno de estos dos path: `/usr/share/doc/mercurial-common/examples` o `/usr/share/doc/mercurial/examples/`. En nuestro caso ejecutaríamos el siguiente comando

```
01. | cp /usr/share/doc/mercurial-
common/examples/hgwebdir.wsgi /var/hg/cgi-bin
```

De momento no vamos hacer ningún cambio en él, lo haremos más adelante. Lo último que nos quedaría es dar permisos para que estos recursos puedan ser accedidos por el usuario con el que se ejecuta Apache.

```
01. | chown -R www-data:www-data /var/hg
```

4. Instalación del módulo modwsgi para Apache

Como hemos comentado en la introducción, vamos a utilizar la solución hgwebdir. Esto implica que tengamos que decidir que opción utilizar para servir los repositorios (módulo de python, módulo de wsgi o CGI). En nuestro caso vamos a instalar el módulo modwsgi porque es la mejor solución por su facilidad de configuración y su mejor rendimiento respecto a las otras alternativas.

Existen varias formas de instalar el módulo en nuestro Apache: de forma manual, compilando los fuentes de `http://code.google.com/p/modwsgi/`, o instalando el módulo ya precompilado de los repositorios de Debian. Por comodidad utilizaremos la segunda opción, así que ejecutamos el siguiente comando:

```
01. | apt-get install libapache2-mod-wsgi
```

2010-08-30

Otras - Electricidad - BARCELONA.

2010-08-24

Otras Sin catalogar - LUGO.

2010-06-25

T. Información - Analista / Programador - BARCELONA.

Una vez finalizada la instalación del módulo, tenemos que comprobar que en el directorio `/etc/apache2/mods-enabled/` existen los siguientes ficheros `wsgi.conf` y `wsgi.load`. Si no existiesen deberíamos crear enlaces a esos ficheros desde `/etc/apache2/mods-available/`.

```
01. | ls -l /etc/apache2/mods-enabled/wsgi.*
```

Recordad que todos los paths que estamos utilizando en este tutorial son los estándar y por tanto deberán ser modificados si tu entorno utiliza otros distintos.

5 Configurando Apache 2.x con el módulo modwsgi

El siguiente paso que debemos seguir es configurar nuestro servidor web. Editamos el fichero `default-ssl` situado en el path `/etc/apache2/sites-available`, y justo antes de la línea `SSLEngine On`, añadimos:

```
01. | WSGIScriptAlias    /hg /var/hg/cgi-bin/hgwebdir.wsgi
02.
03. | <Directory /var/hg/cgi-bin>
04. |     Options ExecCGI
05. |     AddHandler wsgi-script .wsgi
06. |
07. |     AllowOverride None
08. |     Order allow,deny
09. |     Allow from all
10. | </Directory>
11. | # SSL Engine Switch:
12. | # Enable/Disable SSL for this virtual host.
13. | SSLEngine on
```

Con estas líneas lo que hacemos es que cuando se acceda a la url `https://nuestroservidor/hg`, se lance el script `/var/hg/cgi-bin/hgwebdir.wsgi` y nos de acceso a los repositorios que tengamos configurados.

6 Retocando el script hgwebdir.wsgi

Un vez que hayamos configurado Apache tendremos que modificar el fichero `hgwebdir.wsgi` que copiamos al principio. Este fichero será el encargado de proporcionar el acceso a los repositorios que configuremos. En él se define la ruta del fichero de configuración de Mercurial. Por defecto el script intentará buscar un fichero llamado `hgweb.config` dentro del directorio `cgi-bin`. En nuestro caso esta ruta no es válida y la tendremos que sustituir por `/var/hg/hgweb.config`, así que editamos el fichero `/var/hg/cgi-bin/hgwebdir.wsgi`, nos dirigimos a la última línea y donde pone:

```
01. | application = hgwebdir('hgweb.config')
```

lo sustituimos por:

```
01. | config = 0/var/hg/hgweb.config0
02. | application = hgwebdir(config)
```

Ahora sólo nos quedaría añadir la configuración necesaria al fichero hgweb.config.

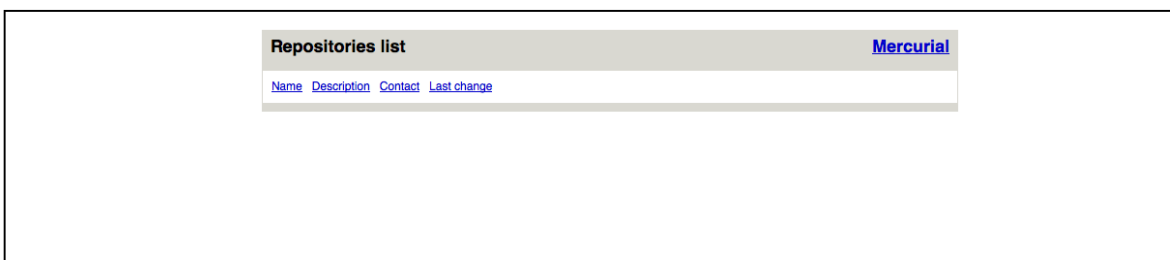
7 Configurando Mercurial

Prácticamente hemos finalizado la configuración básica para poder publicar un repositorio a través de Apache. Ahora nos queda añadir la configuración necesaria al fichero /var/hg/hgweb.config. Realmente se podría considerar como cualquier otro fichero de configuración hgrc que nos podríamos encontrar en el directorio .hg de un repositorio. La única diferencia es que en él vamos a configurar todo lo necesario para el acceso a los repositorios vía https.

Como ya explicamos en el anterior tutorial sobre Mercurial, este fichero está dividido en secciones donde cada una tendrá un conjunto de propiedades que configura una parte de Mercurial. De momento vamos a añadir a este fichero la url base que utilizamos para acceder a los repositorios y el estilo con el que se presentará la información vía web. Para ello añadimos las siguientes líneas:

```
01. [web]
02. #Indicamos que el estilo del html sea con la plantilla gitw
03. style=gitweb
04. #Como al script hgwebdir.wsgi se accede con el prefijo hg s
05. baseurl=/hg
```

Si ahora abrimos un navegador y tecleamos https://localhost/hg debe mostrarse algo como esto:



Como podéis comprobar no nos aparece casi nada porque todavía no hemos configurado ningún repositorio para que se pueda acceder a través del Apache.

8 Publicando un repositorio

Como parece evidente, antes de publicar un repositorio deberemos crearlo. Por tanto manos a la obra. Para crear un repositorio Mercurial debemos utilizar el comando "hg init". En nuestro caso haremos lo siguiente:

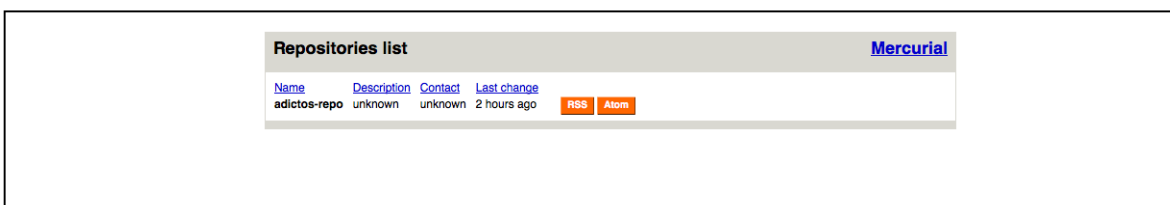
```
01. cd /var/hg/repositories/
02. mkdir adictos-repo
03. cd adictos-repo
04. hg init
05. cd ..
06. chown -R www-data:www-data adictos-repo
```

Una vez creado debemos añadirlo a la configuración para que sea accesible. Existen dos formas de publicar un repositorio o conjunto de repositorios. Lo podemos hacer usando la sección paths o collections. Collections era la forma antigua que se utilizaba en las versiones anteriores de Mercurial 1.1 y se mantiene para mantener compatibilidad, pero lo correcto sería usar la sección paths. En ella vamos a definir todos los repositorios que queremos que sean accesibles a través de nuestro servidor web. Básicamente se trata de indicar el alias que se utilizará para acceder a él y la ruta física donde se encuentra el repositorio en nuestro sistema. En nuestro caso la sección paths quedaría así:

```
01. [paths]
02. #Añadimos el repositorio creado para que se pueda acceder v
    -repo.
03. /adictos-repo = /var/hg/repositories/adictos-repo
```

Con esta configuración, cuando accedamos a <https://miservidor/hg/adictos-repo>, estaremos accediendo al repositorio situado en `/var/hg/repositories/adictos-repo`.

Si guardamos el fichero y volvemos a abrir un navegador accediendo a <https://localhost/hg>, veremos nuestro repositorio accesible:



9 Restringiendo el acceso al repositorio

Si queremos que el acceso a nuestro repositorio sea restringido debemos configurar Apache para que pida credenciales cuando queramos realizar una operación con el repositorio. En este ejemplo vamos a utilizar la autenticación básica, pero podremos configurar cualquier autenticación que Apache permita como Digest o Ldap.

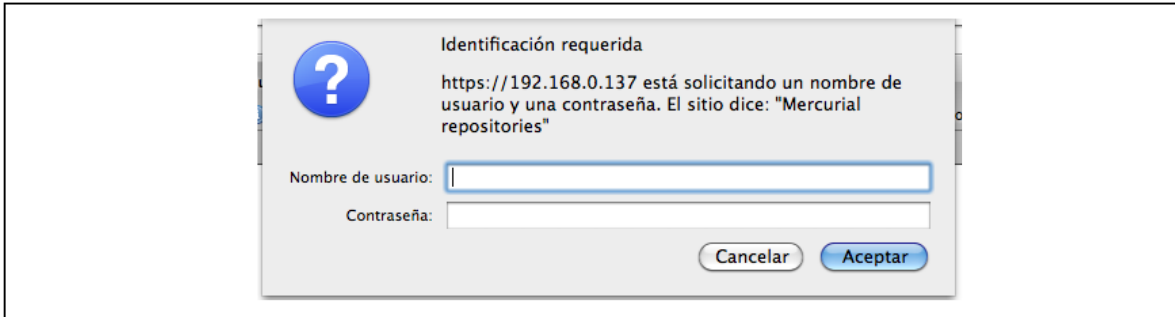
Editamos el fichero `default-ssl` y añadimos las siguientes líneas al apartado `Directory` creado anteriormente:

```
01. <Directory /var/hg/cgi-bin>
02.     Options ExecCGI
03.     AddHandler wsgi-script .wsgi
04.
05.     AllowOverride None
06.     Order allow,deny
07.     Allow from all
08.     AuthType Basic
09.     AuthName "Mercurial repositories"
10.     AuthUserFile /var/hg/hgusers
11.     Require valid-user
12. </Directory>
```


Ahora sólo nos queda crear el fichero de usuarios, y cuando terminemos reiniciaremos el apache para que coja los cambios. Para crear los usuarios hacemos lo siguiente:

```
01. cd /var/hg/hgusers
02. touch hgusers
03. htpassword /var/hg/hgusers adictos
04. htpassword /var/hg/hgusers autentia
05. chown www-data:www-data /var/hg/hgusers
```

Ahora cuando queramos acceder al repositorio nos pedirá las credenciales.



10 Activando el push sobre el repositorio.

Por defecto el push de los repositorios se encuentra desactivado. Para habilitarlo debemos editar el fichero hgrc que se encuentra dentro del directorio .hg del repositorio y añadir las siguientes líneas (en el caso de que no existiese lo deberíamos crear):

```
01. [web]
02. allow_push=*
```

Con esto permitimos la operación push de cualquier usuario sobre el repositorio. Para restringir qué usuarios pueden hacer push en vez de "*", indicaremos la lista de usuarios que podrán realizar esta operación separados por comas.

Y con este último cambio tenemos montado nuestro repositorio accesible por https. En un próximo tutorial veremos cómo añadir permisos de grano fino.

11 Conclusión

Habilitar el acceso remoto a los repositorios es algo habitual cuando trabajamos con un sistema de control de versiones. Los equipos de desarrollo lo necesitan para poder trabajar con ellos. En el caso de Mercurial sólo se podrá acceder por medio de ssh o http. Con este tutorial hemos visto la segunda opción por ser la más utilizada en este SCM.

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

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

(Sólo para usuarios registrados)

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

COMENTARIOS



SOME RIGHTS RESERVED

Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5

Copyright 2003-2010 © All Rights Reserved. Text Copyright conditions

