EBook Gratis

APRENDIZAJE mercurial

Free unaffiliated eBook created from
Stack Overflow contributors.

#mercurial

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: mercurial

It is an unofficial and free mercurial 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 mercurial.

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 mercurial

Observaciones

Mercurial es un moderno sistema de control de versiones distribuido y de código abierto. Es posible que hayas oído hablar de git, que es algo más popular hoy en día; git y Mercurial son muy comparables y ofrecen una funcionalidad similar.

Los desarrolladores utilizan Mercurial para realizar un seguimiento de los cambios en el código fuente de sus aplicaciones. Puede usarse para realizar un seguimiento de los cambios en cualquier directorio de archivos, aunque, como la mayoría de los sistemas de control de origen, funciona mejor cuando esos archivos son archivos de texto (en lugar de archivos binarios). Mercurial también permite que varios desarrolladores trabajen en el mismo directorio de código fuente simultáneamente, y administra el proceso de seguimiento de los cambios de cada desarrollador y luego fusiona esos cambios.

Versiones

Mercurial sigue un plan de lanzamiento basado en el tiempo y publica un lanzamiento importante cada tres meses, en febrero, mayo, agosto y noviembre. Un lanzamiento menor se publica cada mes restante. Tenga en cuenta que Mercurial no utiliza el control de versiones semático, por lo tanto, no hay una diferencia significativa entre 2.9.2 (la última versión de las 2.9 versiones) y 3.0.

La compatibilidad con versiones anteriores es un requisito crítico para Mercurial, por lo que generalmente es seguro simplemente actualizar sus instalaciones de Mercurial según sea necesario. Sin embargo, es una buena idea hacerlo al menos una vez cada tres meses, ya que algunas características pueden hacer que los clientes antiguos no puedan operar en repositorios más nuevos.

Las notas de la versión del proyecto detallan lo que está cambiando entre las versiones, mientras que las notas de actualización ofrecen una guía clara sobre lo que los usuarios deben tener en cuenta al actualizar.

Una selección de versiones notables:

Versión	Notas	Fecha
3.5	Soporte de gotas para Python 2.4 y 2.5	2015-07-31
2.1	Introduce "fases" de conjunto de cambios que permiten la modificación segura del historial	2012-02-01
1.9	Presenta una API de Command Server para admitir una mejor integración de aplicaciones	2011-07-01

Versión	Notas	Fecha
1.7	El nuevo formato de repositorio admite nombres de archivos poco comunes	2010-11-01

Examples

Instalación y configuración

Puede descargar Mercurial desde el sitio web del proyecto, y hay utilidades gráficas para Windows, Linux y OSX si lo prefiere a una interfaz de línea de comandos. La mayoría de los gestores de paquetes Unix incluyen Mercurial, por ejemplo en Debian / Ubuntu:

```
$ apt-get install mercurial
```

Puede verificar que Mercurial está instalado ejecutando:

```
$ hg --version
```

Preparar

Mercurial trabaja fuera de la caja, pero es probable que desee configurar Mercurial para saber quién es usted antes de continuar. Para asociar un nombre de usuario con su edición de confirmaciones `~/.hgrc` (o `mercurial.ini` en su directorio de inicio en Windows) y agregue las siguientes líneas:

```
[ui]
username = Your Name <your@email.address>
```

Si no desea hacer esto, siempre puede especificar un nombre de usuario cuando se comprometa con el indicador `-u` , por ejemplo:

```
$ hg commit -u "Your Name <your@email.address>"
```

Empezando

Ver también el Tutorial de Mercurial.

Creando un repositorio mercurial

Un repositorio de Mercurial es simplemente un directorio (denominado "directorio de trabajo") que contiene un directorio `.hg` con metadatos sobre el contenido del repositorio. Esto hace que Mercurial sea muy ligero y fácil de usar. Para crear un nuevo repositorio simplemente ejecute:

```
$ hg init project
```

Donde `project` es el nombre del directorio que te gustaría crear. Esto crea un directorio de `project` junto con un directorio de `project/.hg` contiene el propio repositorio.

```
$ cd project
$ echo Hello World > hello.txt
$ hg stat
? hello.txt
```

Acabamos de crear un archivo `hello.txt` en el repositorio y `hello.txt` `hg status` (o `stat` para abreviar) para ver el estado actual de nuestro repositorio. Como puedes ver, `hello.txt` está anotado con un `?` , lo que significa que Mercurial todavía no es consciente de ello. El comando `add` registra este nuevo archivo con Mercurial, por lo que se incluirá en la próxima confirmación.

```
$ hg add hello.txt
```

Ahora que Mercurial está al tanto de un archivo modificado, puede ejecutar `diff` para ver exactamente qué ha cambiado desde la última confirmación. En este caso, estamos agregando el contenido completo de `hello.txt` :

```
$ hg diff
diff -r 000000000000 hello.txt
--- /dev/null   Thu Jan 01 00:00:00 1970 +0000
+++ b/hello.txt Sat Jul 23 01:38:44 2016 -0400
@@ -0,0 +1,1 @@
+Hello
```

Y una vez que estemos contentos con ellos y listos para revisar nuestros cambios, podemos ejecutar el `commit` :

```
$ hg commit -m "Created a hello world file."
```

Tenga en cuenta que incluimos un mensaje de confirmación con `-m` : si no especifica `-m` Mercurial abrirá un editor de texto en el que puede ingresar un mensaje de confirmación. Esto es útil si desea proporcionar un mensaje de varias líneas más largo.

Una vez que haya confirmado los cambios, ya no se mostrarán si ejecuta `hg stat` ya que el repositorio ahora está sincronizado con el contenido del directorio de trabajo. Puede ejecutar el `log` para ver una lista de confirmaciones, y `-v` incluye detalles adicionales como los archivos que cada confirmación tocó:

```
$ hg log -v
changeset:   0:b4c06cc77a42
tag:         tip
user:        Michael Diamond@Aodh <dimo414@gmail.com>
date:        Sat Jul 23 01:44:23 2016 -0400
files:       hello.txt
description:
Created a hello world file.
```

Empujando y tirando

Mercurial hace que sea fácil compartir su trabajo y obtener contribuciones de otros desarrolladores. Esto implica tres pasos clave; clonando , tirando , y empujando .

Clon

Para copiar un repositorio remoto a su disco local, debe "clonarlo". Para hacerlo, simplemente pase la URL remota desde la que desea clonar. Para clonar el código fuente de Mercurial simplemente ejecute:

```
$ hg clone https://selenic.com/hg
```

Esto crea un directorio `hg` local que contiene una copia del repositorio de Mercurial que puede compilar, editar y comprometer (aunque no puede publicar sus confirmaciones en el repositorio principal).

Halar

Una vez que tenga un repositorio desprotegido, querrá mantenerlo sincronizado ya que otros publican cambios en él. Puedes desplegar nuevos cambios simplemente ejecutando:

```
$ hg pull
```

Esto introduce nuevas confirmaciones pero no actualiza su directorio de trabajo, por lo que no verá ningún cambio de inmediato. Para actualizar el contenido del directorio de trabajo ejecute:

```
$ hg up
```

Que actualiza su directorio de trabajo a la revisión de punta (más reciente) en el repositorio.

También puede ejecutar:

```
$ hg pull -u
```

Para introducir nuevos cambios y actualizar el directorio de trabajo en un solo paso.

empujar

Suponiendo que tenga acceso de escritura al repositorio remoto, puede publicar cualquier confirmación que haya hecho localmente en el repositorio remoto con la misma facilidad con:

```
$ hg push
```

Esto carga sus cambios siempre y cuando no haya habido otros confirmaciones desde la última vez que realizó la extracción. Si su `push` es rechazado porque "crearía cabezas adicionales", eso significa que necesita incorporar esos nuevos cambios y fusionarlos con los suyos.

```
$ hg pull
$ hg merge  # this creates a new changeset merging your changes with the remote changes
$ hg commit -m "Merged in remote changes"
$ hg push
```

En la mayoría de los casos, esto es todo lo que tendrá que hacer ya que Mercurial se encarga de fusionar sus cambios automáticamente, sin embargo, a veces tendrá que resolver los conflictos de combinación manualmente (consulte el tema de fusión). Si lo necesita, siempre puede cancelar una combinación y volver a un directorio de trabajo limpio con:

```
$ hg up -c
```

Pero recuerda que esto es una operación destructiva; Cualquier cambio en el directorio de trabajo será borrado.

Derivación

Cuando iniciamos nuestro trabajo por primera vez, debemos decidir si se trata de un área de trabajo separada en la que estamos trabajando o si es parte de una línea de trabajo existente. Si existe, podemos trabajar fuera de esa rama. Si es nuevo, comenzaremos una nueva sucursal.

Nuestro flujo de trabajo es:

- `hg branch MyNewFeature`
- trabajo Trabajo trabajo
- `hg commit -m "committing my changes"`
- trabajo Trabajo trabajo
- `hg commit -m "more changes"`

En este punto, queremos impulsar nuestro trabajo al servidor remoto. Pero antes de empujar los cambios (ignore esto si es una rama nueva que no ha empujado antes), debemos verificar si hay algún cambio entrante en esta rama. Podemos verificar esto con:

```
hg incoming -b .
```

Si hay conjuntos de cambios entrantes en nuestra sucursal, debemos hacer un tirón y volver a ajustar nuestros cambios al principio de la lista de cambios.

```
hg pull -b . --rebase
```

Una vez hecho esto o si no hay conjuntos de cambios entrantes, podemos continuar con el Push.

Solo queremos impulsar nuestro trabajo actual, no todo lo que hemos hecho. Realmente nunca presiono todo mi repositorio, sino mi línea de trabajo actual. El razonamiento es que al empujar todo el repositorio se supone que estoy integrando múltiples líneas de trabajo. Pero solo quiero integrar mi línea de trabajo actual, y solo quiero trabajar en una línea a la vez.

Si esta es la primera vez que estoy empujando esta rama:

```
hg push -b . --new-branch
```

Si ya he empujado esta rama:

```
hg push -b .
```

El comando "-b." Significa simplemente empujar la rama actual, y no cualquier otra cosa.

Para cambiar entre las ramas de trabajo :

```
hg update myBranchName
```

Lea Empezando con mercurial en línea:
https://riptutorial.com/es/mercurial/topic/2075/empezando-con-mercurial

Capítulo 2: Colas mercuriales

Sintaxis

- hg qnew -m "Mi mensaje de confirmación" myPatch
- hg qpop
- hg qpush
- hg qrefresh -m "Mi nuevo mensaje de confirmación"
- hg qapplied
- hg qseries
- hg qfinish
- hg qdelete myPatch
- hg qfold myPatch
- hg qqueue --list
- hg qqueue --crear myNewQueue
- hg qqueue --delete myNewQueue

Examples

Habilitar extensión

Edite **Mercurial.ini** (Windows) o **.hgrc** (Linux / OSX):

```
[extenstions]
mq =
```

Crear y actualizar parches

Cree nuevos parches con `hg qnew patch-name` y luego actualícelos con nuevos cambios con `hg qrefresh` :

```
hg qnew myFirstPatch  // Creates a new patch called "myFirstPatch"
...                   // Edit some files
hg qrefresh           // Updates "myFirstPatch" with changes
hg qnew mySecondPatch // Creates a new patch called "mySecondPatch" on top of "myFirstPatch"
...                   // Edid some files
hg qrefresh           // Updates "mySecondPatch" with changes
```

Crear nuevo parche con mensaje de confirmación:

```
hg qnew -m "My first patch" myFirstPatch
```

Actualice el mensaje de confirmación del parche actual:

```
hg qrefresh -m "My new message"
```

Actualice el mensaje de confirmación (multilínea) del parche actual:

```
hg qrefresh -e
```

Parches Push y Pop

Al presionar un parche, se aplica el parche al repositorio mientras se abre un parche para que no se aplique el parche desde el repositorio.

`hg qpop` el parche actual de la cola con `hg qpop` y vuelva a colocarlo en la cola con `hg qpush` :

```
hg qpop
hg qpush
```

Pop todos los parches:

```
hg qpop -a
```

PUsh todos los parches:

```
hg qpush -a
```

Parches de acabado

Los parches de acabado los convierten en cambios permanentes.

Finalizar el **primer** parche aplicado en la cola:

```
hg qfinish
```

Terminar **todos los** parches aplicados en la cola:

```
hg qfinish -a
```

Rebase

Para reajustar con los últimos conjuntos de cambios en el repositorio ascendente:

```
hg qpop -a    // Pop all patches
hg pull -u    // Pull in changesets from upstream repo and update
hg qpush -a   // Push all patches on top of new changesets
```

Si hay algún conflicto, se verá obligado a combinar sus parches.

Parches Doblar / Combinar

Para plegar (combinar) dos parches:

```
hg qnew firstPatch   // Create first patch
hg qnew secondPatch  // Create second patch
hg qpop              // Pop secondPatch
hg qfold secondPatch // Fold secondPatch with firstPatch
```

Múltiples colas

Se puede crear más de una cola. Cada cola puede considerarse como una rama separada.

```
hg qqueue --create foo // Create a new queue called "foo"
hg qqueue --list       // List all queues
hg qqueue --active     // Print name of active queue
hg qqueue --rename bar // Rename active queue "foo" to "bar"
hg qqueue --delete bar // delete queue "bar"
```

Crea dos colas y cambia entre ellas:

```
hg qqueue --create foo // Create a new queue called "foo" and make it active
hg qqueue --create bar // Create a new queue called "bar" and make it active
hg qqueue foo          // Switch back to queue "foo"
```

Comandos

- **qnew** : crea un nuevo parche
- **qpop** : **saca** el parche actual de la pila
- **qpush** : empujar el siguiente parche en la pila
- **qrefresh** : actualizar el parche actual
- **qapplied** : imprime los parches ya aplicados
- **qseries** : imprime el archivo completo de la serie
- **qfinish** : mueve los parches aplicados al historial del repositorio
- **qdelete** : eliminar parches de la cola
- **qdiff** : diff del parche actual y modificaciones posteriores
- **qclone** : clonación principal y repositorio de parches al mismo tiempo
- **qfold** : dobla los parches nombrados en el parche actual
- **qgoto** : parches push o pop hasta que el parche con nombre esté en la parte superior de la pila
- **qguard** : establece o imprime guardias para un parche
- **qheader** : imprime el encabezado del parche superior o especificado
- **qimport** : importar un parche o un conjunto de cambios existente
- **qnext** : imprime el nombre del próximo parche empujable
- **qprev** : imprime el nombre del parche aplicado anterior
- **qqueue** : gestiona múltiples colas de parches
- **qrecord** : graba interactivamente un nuevo parche
- **qrename** : renombrar un parche
- **qselect** : configura o imprime parches quarded para empujar
- **qtop** : imprime el nombre del parche actual
- **qunapplied** : imprime los parches aún no aplicados

Obsoleto:

- **qinit** : **inicia** un nuevo repositorio de colas (DEPRECATED)
- **qcommit** : confirmar cambios en el repositorio de colas (DEPRECATED)
- **qsave** : guardar el estado actual de la cola (DEPRECATED)
- **qsave** : restaura el estado de la cola guardada por una revisión (DEPRECATED)

Lea Colas mercuriales en línea: https://riptutorial.com/es/mercurial/topic/5379/colas-mercuriales

Capítulo 3: Lista de comandos

Examples

Comandos para preparar confirmaciones

- **agregar** : agrega los archivos especificados en el siguiente commit
- **addremove** : agrega todos los archivos nuevos, elimina todos los archivos faltantes
- **retroceso** : efecto inverso de un conjunto de cambios anterior
- **commit, ci** : commit los archivos especificados o todos los cambios pendientes
- **copiar, cp** : marcar los archivos como copiados para la próxima confirmación
- **olvidar** : olvidar los archivos especificados en la próxima confirmación
- **fusionar** : fusionar otra revisión en el directorio de trabajo
- **remove, rm** : elimina los archivos especificados en la próxima confirmación
- **renombrar, mover, mv** : renombrar archivos; equivalente de copia + eliminar
- **resolver** : rehacer fusiones o configurar / ver el estado de combinación de archivos
- **revertir** : restaura los archivos a su estado de salida reciente

Inspeccionando la historia

- **anotar, culpar** : mostrar información del conjunto de cambios por línea para cada archivo
- **bisecta** : subdivision busqueda de conjuntos de cambios
- **cat** : muestra la revisión actual o dada de los archivos
- **diff** : repositorio de diff (o archivos seleccionados)
- **grep** : busca un patrón en archivos específicos y revisiones
- **registro, historial** : muestra el historial de revisiones de todo el repositorio o archivos

Intercambiando conjuntos de cambios con repositorios remotos

- **archivo** : crear un archivo sin versión de una revisión del repositorio
- **paquete** : crear un archivo de grupo de cambios
- **clonar** : hacer una copia de un repositorio existente
- **exportar** : volcar el encabezado y diffs para uno o más conjuntos de cambios
- **injerto** : copiar cambios de otras ramas en la rama actual
- **entrante** : muestra los nuevos conjuntos de cambios encontrados en la fuente
- **importar, parchar** : importar un conjunto ordenado de parches
- **init** : crea un nuevo repositorio en el directorio dado
- **saliente** : muestra los conjuntos de cambios que no se encuentran en el destino
- **fase** : establece o muestra el nombre de la fase actual
- **pull** : pull cambios desde la fuente especificada
- **push** : push cambia al destino especificado
- **recuperar** : deshacer una transacción interrumpida
- **deshacer** : **deshacer** la última transacción (DANGEROUS) (DEPRECATED)
- **Servir** : iniciar servidor web independiente
- **Descomprimir** : aplicar uno o más archivos de grupo de cambios.

Estado: ¿dónde estás ahora?

- **marcadores, marcadores** : crea un nuevo marcador o enumera los marcadores existentes
- **rama** : establece o muestra el nombre de la rama actual
- **ramas** : lista repositorio con nombre ramas
- **config, showconfig, debugconfig** : muestra los ajustes de configuración combinados de todos los archivos hgrc
- **archivos** : lista de archivos rastreados
- **ayuda** : muestra ayuda para un tema determinado o una descripción general de ayuda
- **identificar, id** : identificar el directorio de trabajo o la revisión especificada
- **entrante, en** : mostrar nuevos conjuntos de cambios encontrados en la fuente
- **localizar** : localizar archivos que coinciden con patrones específicos (DEPRECATED)
- **manifiesto** : muestra la revisión actual o dada del manifiesto del proyecto
- **saliente, saliente** : muestra los conjuntos de cambios que no se encontraron en el destino
- **Padres** : mostrar a los padres del directorio de trabajo o revisión (DEPRECATED)
- **rutas** : mostrar alias para repositorios remotos
- **fase** : establece o muestra el nombre de la fase actual
- **raíz** : imprime la raíz (parte superior) del directorio de trabajo actual
- **estado, st** : muestra los archivos modificados en el directorio de trabajo
- **resumen, suma** : resumen estado de directorio de trabajo
- **etiquetas** : lista de etiquetas de repositorio
- **punta** : muestra la revisión de la punta (DEPRECATED)
- **verificar** : verificar la integridad del repositorio
- **versión** : **versión de** salida e información de copyright

Flujo de trabajo: ramas, etiquetas y movimiento.

- **marcadores, marcadores** : crea un nuevo marcador o enumera los marcadores existentes
- **rama** : establece o muestra el nombre de la rama actual
- **etiqueta** : agregue una o más etiquetas para la revisión actual o dada
- **actualizar, subir, revisar, co** : actualizar el directorio de trabajo (o cambiar las revisiones)

Lea Lista de comandos en línea: https://riptutorial.com/es/mercurial/topic/4170/lista-de-comandos

Capítulo 4: Operaciones comunes

Examples

Usando el comando bisect para encontrar un error

El comando `bisect` ayuda a rastrear el conjunto de cambios que introdujo un error.

- Reinicie el estado bisect y marque la revisión actual como mala (¡contiene el error!)

```
hg bisect --reset
hg bisect --bad
```

- Vuelve a un punto donde crees que el error no está presente

```
hg update -r -200
```

- Ahora tiene que probar el software y si su suposición fue correcta (error no presente), marque la revisión como buena:

```
hg bisect --good
```

  Prueba de conjunto de cambios 800: 12ab34cd56ef (x conjuntos de cambios restantes, ~ y pruebas)

- Mercurial actualiza la revisión actual (en algún punto intermedio entre el conjunto de cambios malo y bueno)
- Vuelva a probar el software y marque adecuadamente la revisión actual. P.ej

```
hg bisect --good
```

  Pruebas changeset 900: 21ba43dc65fe (x conjuntos de cambios restantes, ~ y pruebas)

- ...
- Continuar hasta que Mercurial haya reducido la búsqueda a un solo conjunto de cambios:

```
hg bisect --bad
```

  La primera mala revisión es:

  conjunto de cambios: 987: 1234bad99889

  usuario: John Doe ____@gmail.com

fecha: 28 jul 16:00:00 2016

El comando `hg bisect` utiliza su conocimiento del historial de revisiones de su proyecto para realizar una búsqueda en el tiempo proporcional al logaritmo del número de conjuntos de cambios que se deben verificar y no tiene problemas para tratar con sucursales, fusiones o múltiples cabezas.

A veces tienes una idea de los archivos incriminados y puedes darle una pista a Mercurial:

```
hg bisect --skip "!( file('path:foo') & file('path:bar') )"
```

Esto omite todas las revisiones que no toquen los directorios `foo` o `bar` .

Uso del comando revert para descartar cambios no deseados.

El comando `revert` permite descartar cambios no deseados no deseados.

- Revertir cambios a un solo archivo.

```
hg revert example.c
```

- Revirtiendo todos los cambios.

Esto descartará **todos los** cambios, no solo el directorio actual.

```
hg revert --all
```

hg dará salida a los archivos que fueron revertidos.

revertir example.c

revertir mydir \ example.cpp

olvidando file.txt

Los archivos de copia de seguridad se producen para los cambios descartados en archivos previamente confirmados, en la forma `filename.orig`

Lea Operaciones comunes en línea: https://riptutorial.com/es/mercurial/topic/2490/operaciones-comunes

Creditos

S. No	Capítulos	Contributors
1	Empezando con mercurial	Community, dimo414, Frank Schmitt, Joel Spolsky, Tom, Vivek Vijayan
2	Colas mercuriales	jenglert, mcarlin
3	Lista de comandos	Esteis, jenglert
4	Operaciones comunes	manlio, Tom