Hg Mercurial Cheat Sheet

Serge Y. Stroobandt

> #This page is work in progress! Much of the explanatory text still needs to be written. Nonetheless, the basic outline of this page may already be useful and this is why I am sharing it. In the mean time, please, bare with me and check back for updates.

Distributed revision control

Why I went with Mercurial

- Python, Mozilla, Java, Vim
- Mercurial has been better supported under Windows.
- Mercurial also offers named branches

Emil Sit:

- August 2008: Mercurial offers a comfortable command-line experience, learning Git can be a bit daunting
- December 2011: Git has three "philosophical" distinctions in its favour, as well as more attention to detail

Lowest common denominator It is more important that people start using distributed revision control instead of nothing at all.

The Pro Git book is available online.

Collaboration styles

- Mercurial working practices
- Collaborating with other people

Use SSH

shorthand

Installation

```
$ sudo apt-get update
$ sudo apt-get install mercurial mercurial-git meld
```

Configuration

Local system-wide configuration

```
$ nano .bashrc

    export NAME="John Doe"
    export EMAIL="john.doe@domain.url"

$ source .bashrc
```

~/.hgrc on a client

```
user@client $ nano ~/.hgrc

    [ui]
    username = user@client
    editor = nano
    merge = meld
    ssh = ssh -C

    [extensions]
    convert =
    graphlog =
    mq =
    progress =
    strip =
```

~/.hgrc on the server

```
user@server $ nano ~/.hgrc

    [ui]
    username = user@server
    editor = nano
    merge = meld
    ssh = ssh -C

    [extensions]
    convert =
    graphlog =
    mq =
    progress =
    strip =

    [hooks]
    changegroup = hg update >&2
```

Initiating

One starts with initiate a new repository.

```
$ hg init directory
```

Status

```
$ hg st
```

Ignoring

Ignore files

The following `.hgignore` file will ignore using `glob` syntax:

- all intermediary files in every directory,
- all JabRef `.sav` files in every directory,
- all LibreOffice `.~lock.` files in every directory,
- all Vim `.swp` files in every directory,
- all files in the `not/` directory,

...and using regular expression syntax:

- all `.pdf` files only in the top level base directory.

```
syntax:glob
*.aux
*.bbl
*.blg
*.~lock.*
*.log
not/*
*.sav
*.swp

syntax:re
^(?!.*/).*\.pdf$
```

List ignored files

This lists the ignored files without the status prefix:

```
hg st -ni
```

Remove ignored files locally

The following command will delete all ignored files in the local folder. Use with caution.

```
hg st -ni0 |xargs -0 rm
```

Adding and removing files

To schedule all new files to be added to the repository for version control and to record the deletion of all missing files from the repository:

```
$ hg addr
```

The files will be added or removed at the next commit. To undo before that, see `hg forget`.

Committing

Forget changes before commit

To undo a file add

```
$ hg forget filepath
```

Revert local changes

To revert a locally modified file back to the latest local repository version, without creating a backup:

```
$ hg rev -C filepath
```

To do the same for all files in the repository:

```
$ hg rev -C --all
```

Commit changes

> Some advice: “Commit early and commit often!”

```
$ hg com -m 'commit message'
```

If the message consists out of a single word, the quotes can be omitted.

```
$ hg com -m one-word-message
```

Change the last commit message

```
$ hg com --amend -m 'new commit message'
```

Undo the last commit

This command should be used with care. There is no way to undo a rollback. Moreover, this command is not intended for use on public repositories. Please, consult the help information. Use `hg commit --amend` instead of rollback to correct mistakes in the last commit.

```
$ hg rollback
repository tip rolled back to revision x (undo commit)
working directory now based on revision x
```

Above command does not alter the working directory. To entirely clean up the working directory and revert to the previous commit status, use the following commands.

```
$ hg rev -C -a
reverting ...

$ hg st -nu0 |xargs -0 rm
```

Wipe files from history

Matters are a bit more complicated than represented here, so be sure to first read the following excellent article. This is especially the case when working in a group of collaborators. So here is the short story. First, create a `map.txt` file:

```
# This filemap is used to exclude specific files
exclude subdir/filename1.ext
exclude subdir/filename2.ext
exclude subdir2
```

Then, run these commands:

```
hg convert --filemap map.txt ~/oldrepo ~/newrepo
cd ~/newrepo
hg update
```

Attention:

> **All instances of the old repository,** including those on remote clients and servers, **need to be removed** with `rm -R`. Then, clone again from the new repository.

Pushing

Push to the main repository

```
$ nano .hg/hgrc
    [paths]
    default = user@server.url:port/relative_path
```

```
$ hg out
```

```
$ hg push
```

Auto-update the main repository

`.hg/hgrc`

```
[hooks]
changegroup = hg update
```

Push to an additional server

`.hg/hgrc`

```
[paths]
webserver = ssh://user@webserver.url/public_html

[hooks]
changegroup = hg update >&2
changegroup.webserver = hg push -f webserver
```

This goes in `.hg/hgrc` on the remote repository. The output in this case needs to be redirected to `stderr (&2)`, because `stdout (&1)` is used for the actual data stream. By the way, `stdin` is `&0`. Instead of a web server, a web-based repository hosting service like Bitbucket could be used. The path would then be something along the line of: `ssh://hg@bitbucket.org/user_name/repository_name`.

Push aborts

Pushes to the main repository may be aborted with one of the following messages:

```
remote: abort: outstanding merge conflicts
```

```
remote: abort: untracked files in working directory differ from files in
requested revision
```

This happens when changes have been made directly to the main repository without committing. Performing a clean update to revision `tip` on the main repository server resolves this issue. Doing so, deletes the uncommitted changes on the main repository server.

```
$ hg update -C -r tip
```

Cloning

Clone from a remote repository

Hg Mercurial can be used with the secure shell protocol:

```
$ hg clone ssh://user@server.url:port/relative_remote_path
$ hg clone ssh://user@server.url:port//absolute_remote_path
```

Clone to a remote repository

```
$ hg clone localrepo ssh://user@server.url:port/relative_remote_path
$ hg clone localrepo ssh://user@server.url:port//absolute_remote_path
```

Pulling

Pull updates

```
$ hg inc
```

```
$ hg pull -u
```

Viewing

View repository heads

```
$ hg heads
```

View the log

For a semi-graphical log in ASCII art, add the `graphlog=` extension to the `~/.hgrc` configuration file as follows:

```
[extensions]
graphlog =
```

To view the graphical log:

```
$ hg log -G
```

TortoiseHg

TortoiseHg is an application with a graphical user interface to view and manipulate `hg-mercurial` repositories.

```
$ sudo apt install tortoisehg
```

Merging

`~/.hgrc`

```
[ui]
merge = meld
```

```
$ hg merge
```

```
$ hg com -m merge
$ hg push
```

Reverting

Revert to a previous version

To revert the repository to a previous version, without creating a backup:

```
$ hg rev -C -r revision_number -a
```

Work from there and change something.

```
$ hg com -m reversion
$ hg push
```

Branching

There are four types of branching using Hg Mercurial: clones, bookmarks, named branches and anonymous. These are conceptually different from Git branches, with bookmarks being the closest to Git branches. All this is explained beautifully with nice drawings in Steve Losh's *A Guide to Branching in Mercurial*.

Create a new branch

```
$ hg branch experiment
```

View the name of the current branch

```
$ hg branch
```

List all branches

```
$ hg branches
```

Push a new branch

Use `--new-branch` to instruct `push` to create a new named branch that is currently not present at the destination. This allows you to only create a new branch without forcing any other changes.

```
$ hg com -m message
$ hg pus --new-branch
```

Change back to the default branch

Use the `hg update` command to switch to an existing branch; for example the `default` branch.

```
$ hg update default
```

Merge a branch with the default branch

By default, `push` will not permit the creation of new heads at the destination, since multiple heads would make it unclear which head to use. In this situation, it is recommended to pull and merge before pushing.

```
$ hg update default
$ hg merge experiment
$ hg com -m "branch merge"
```

Close a branch

Use `hg commit --close-branch` to mark the current branch head as closed. When all heads of a branch are closed, the branch will be considered closed. Closing branches prevents the list of branches from growing too large.

```
$ hg update experiment
$ hg com --close-branch
```

Merge & close an anonymous branch

Occasionally, an anonymous branch may appear in the commit log tree, caused by an error. Ordinary branch merging will not work and rather raise the following error:

```
abort: merging with a working directory ancestor has no effect
```

Here is how to go about merging and closing such an anonymous branch:

```
$ hg update -r oldest_anonymous_branch
$ hg merge
$ hg com -m "merged branches"
$ hg branches
$ hg update -r newest_anonymous_branch
$ hg com --close-branch
```

Prune a dead branch

Here is how to strip a revision and all later revisions on the same branch. Please note, this is a history altering operation. Be careful no one else pulled this repository. Moreover, for this to work, the MqExtension needs to be loaded as `mq =` in `~/.hgrc`.

```
hg strip --no-backup -r revisionnumber
```

Help

Here is how to get help with a specific command:

```
$ hg help command
$ hg -v help command
```

Resolve local status corruption

```
$ hg update -C
$ hg debugrebuildstate
```

This work is published at https://hamwaves.com/hg-mercurial/en/.

Last update: Monday, March 1, 2021.