# Git Stuff

From here,

*“Git is an open source, distributed version control system designed to handle everything from small to very large projects with speed and efficiency.”*

Some useful configurations before starting with git. I have this simple script that I run on a new machine I want to work on.

## Create, Clone, Add, Commit

Create new repo

```
$ mkdir /path/to/project ; cd /path/to/project ; git init
```

Clone a repo

```
$ git clone git://server/path/to/project.git
```

Add everything in current path to repo

```
$ git add .
```

Add file(s) to repo @ next commit

```
$ git add <file1> <file2>
```

Commit changes added

```
$ git commit
```

Commit ALL changes in currently tracked files

```
$ git commit -a
```

## Create/Apply Patches

Create a diff patch

```
$ git diff COMMIT_1 COMMIT_2 > PATCHFILE
```

Apply a diff patch

```
$ git apply < PATCHFILE
```

Create a formatted patch (can also use -n where n is the number of previous commits to create patches for)

```
$ git format-patch -o /path/to/patches/ -1
```

Apply a formatted patch

```
$ git am < PATCHFILE
```

Apply a formatted patch (create new date)

```
$ git am --ignore-date < PATCHFILE
```

Create formatted patches (note: sha1id1..sha1id2 is the range of commits used to create patches)

```
$ git format-patch -o /path/to/patches/ sha1id1..sha1id2
```

To rename created patch(es) with commit timestamp

```
 for a in *.patch ; do l=$(cat $a|grep 'Date: ') ; d=$(date +"%Y%m%d%H%M%S"
-d "${l#*: }") ; mv $a "$d${a:4}" ; done
```

To apply the patches (stopping on error)

```
for a in ~/temp/patches/*.patch ; do git am <$a ; [ $? -ne 0 ] && break;
done
```

# Branching/Merging

Create new local branch

```
$ git branch <local-branch>
```

Remove local branch (-D to force)

```
$ git branch -d <local-branch>
```

Create a local branch to track remote branch

```
$ git checkout --track -b <local-branch> <remote-branch>
```

Set current local branch to track a remote branch

```
$ git branch -u <remote>/<branch>
```

# Basic Management

Checkout specific commit

```
$ git checkout COMMIT
```

Force reset to a specific commit

```
$ git reset --hard [COMMIT]
```

Clean up untracked file(s)/folder(s)

```
$ git clean -f -d
```

Re-committing to add forgotten items

```
$ git reset --soft HEAD^
# <do the forgotten thingy, if necessary do a git add>
$ git commit
```

Changing First Commit Message

```
# checkout the root commit
git checkout <sha1-of-root>
# amend the commit
git commit --amend
# rebase all the other commits in master onto the amended root
git rebase --onto HEAD HEAD master
```

Manage Tag(s)

```
# git tag
# git tag <newtag>
# git tag -d <oldtag>

# git fetch --tags
# git push --tags

# (this works on github - unknown elsewhere)
# git push <remote> :<oldtag>
```

Other commands:

```
# git config --list
# git config remote.origin.url NEW_URL
# git pull ANOTHER_URL master

# git branch -vv
# git branch -u <remote>/<remote_branch> <local_branch>
```

```
# git bundle create <somefile> HEAD
# git pull <somefile> HEAD

# git archive --format=tar --prefix=<folder-prefix>/ <tag/commit> | gzip
>/path/to/tarball-release.tar.gz
```

Merge to master branch while on other branch (without checkout):

```
# git fetch . <src_branch>:master
```

Check log for master branch while on other branch (without checkout):

```
# git log master
```

Merge specific file(s) from another branch

```
# git checkout <src_branch> file1 [file2 file3...]
```

Create a GIT repository on shared hosting (with SSH access) and publish through HTTP

```
# sshfs -p <ssh-port> -o workaround=rename user@server:public_html
/local/mount/path
# cd /local/mount/path
# mkdir repo
# cd repo
# git clone --bare /local/repo <repo-name>
# cd <repo-name>
# git --bare update-server-info
# cp hooks/post-update.sample hooks/post-update
```

Running GIT Daemon:

```
git daemon --reuseaddr --base-path=/path/to/repos --export-all --verbose --
enable=receive-pack
```

# Advanced Git

Merging two different repositories with histories intact (the new one on top of the old)

- assuming repoA and repoB (repoB the same project with different code-base)
- we want to have repoB override repoA, but want to keep repoA history within that project
- so, first create a remote on repoA pointing to repoB
    - `git remote add <repoB_name> <repoB_url>`
- fetch repoB history?
    - `git fetch --all`

- here is where we overwrite repoA (but keep the history…)
    - `git merge repoB_name/master --strategy-option theirs --allow-unrelated-histories`

Rescuing from `git reset –hard`

***Note:*** *This DOES NOT work all the time, I guess… because `git reset –hard` should remove every record. But, this may… (and in my case, once, did) work.*

```
git reset HEAD@{1}
```

Remove remote branch

```
git push origin --delete {branch}
```

If the remote branch has been removed, use this to remove local reference

```
git remote update origin --prune
```

Prepare diff with binary content

```
git diff-index master --binary >bin.patch
```

List tracked files (recursively)

```
git ls-tree --name-only -r <branch>
```

List untracked files

```
git ls-tree --others
```

List ignored files

```
git ls-files --ignored --exclude-standard
```

# Local Management

Fetch updates from downstream (assume in branch staging)

```
$ for that in $(find . -not \( -path "./.git/*" -prune \) -type f) ; do git checkout staging ${that:2} ; done
```

*Dumping ground… will sort this out later…*

```
# list github remote
here=$(pwd) ; for a in $(find $here -mindepth 1 -maxdepth 1 -type d) ; do base=$(basename $a) ; pick= ; for b in $(cat ~/temp/github.txt) ; do [
```

```
"$base" = "$(basename $b)" ] && pick=$b && break ; done ; [ ! -z "$pick" ]
&& continue ; pick=$a ; cd $pick ; what=$(git remote -v | grep -e "^github"
| grep fetch) ; [ -z "$what" ] && continue ; echo "-- REMOVE!
($pick)($what)" ; done ; cd $here

# list & remove github remote
here=$(pwd) ; for a in $(find $here -mindepth 1 -maxdepth 1 -type d) ; do
base=$(basename $a) ; pick= ; for b in $(cat ~/temp/github.txt) ; do [
"$base" = "$(basename $b)" ] && pick=$b && break ; done ; [ ! -z "$pick" ]
&& continue ; pick=$a ; cd $pick ; what=$(git remote -v | grep -e "^github"
| grep fetch) ; [ -z "$what" ] && continue ; echo "-- REMOVE!
($pick)($what)" ; git remote rm github ; done ; cd $here

# list github branch
here=$(pwd) ; for a in $(find $here -mindepth 1 -maxdepth 1 -type d) ; do
base=$(basename $a) ; pick= ; for b in $(cat ~/temp/github.txt) ; do [
"$base" = "$(basename $b)" ] && pick=$b && break ; done ; [ ! -z "$pick" ]
&& continue ; pick=$a ; cd $pick ; what=$(git branch | grep github) ; [ -z
"$what" ] && continue ; that=$(git remote -v | grep -e "^github" | grep
fetch) ; [ ! -z "$that" ] && continue ;  echo "-- Branch: $pick ($what)" ;
done ; cd $here

# list & remove github branch
here=$(pwd) ; for a in $(find $here -mindepth 1 -maxdepth 1 -type d) ; do
base=$(basename $a) ; pick= ; for b in $(cat ~/temp/github.txt) ; do [
"$base" = "$(basename $b)" ] && pick=$b && break ; done ; [ ! -z "$pick" ]
&& continue ; pick=$a ; cd $pick ; what=$(git branch | grep github) ; [ -z
"$what" ] && continue ; that=$(git remote -v | grep -e "^github" | grep
fetch) ; [ ! -z "$that" ] && continue ;  echo "-- Branch: $pick ($what)" ;
git branch -D github  ; done ; cd $here
```

From:
https://azman.my1matrix.cc/ - **Azman @MY1**

Permanent link:
**https://azman.my1matrix.cc/doku.php?id=notes:git**

Last update: **2026/02/08 03:46**