Git Tips and Tricks

Git Setup

The values you set here will be used as defaults when you set up local repositories and connect to remote repositories.

```
git config --global user.name "Your Name Here"

git config --global user.email "your_email@example.com"

git config --global init.defaultBranch main

git config --list
```

Set values for a specific repository (you must be in the root directory of the repository when you issue these commands):

```
git config --local user.name "Your Name Here"

git config --local user.email "your_email@example.com"
```

Initialization

Initialize a local Git repository

```
cd <directory to add to source control>

git init
```

Initialize a local Git repository, based on a remote repository

This uses Github as an example.

1. Open terminal.
2. Create directory: mkdir ~/test-repo (Change “test-repo” to whatever you want.)
3. Change to the new directory: cd ~/test-repo
4. Initialize a local Git repository: git init
5. Point your local repository to the remote repository: git remote add origin https://github.com/yourUserNameHere/test-repo.git

Sync local repo with remote

To sync your local repository with its associated remote repository, issue this command:

```
git fetch origin
```

You may need to retrieve and approve remote changes to force them into your local copy:

```
git checkout master

git merge origin/master
```

You should also be able to use “pull” to combine the fetch and merge into a single operation:

```
git pull origin
```

Clone a copy of a remote repository

```
cd <local_directory>

git clone https://github.com/<username>/<repository_name.git>
```

(This example uses GitHub. Modify it to whatever remote hosting site you need.)

Adding new files to a repository

This should be done before committing.

Add a single file:

```
git add <filename>
```

Add all new files:

```
git add
```

Update tracking for files that changed names or were deleted:

```
git add -u
```

To do both, use this:

```
git add -A
```

Commit changed files to a repository

After files have been added, you can commit them to be saved as an intermediate version:

```
git commit -m "message"
```

…where “message” is a useful description of what you did.

This only updates your local repo. If your repo is associated with a remote repository, the remote repo is not updated in this step.

Update remote repository

If you have saved local commits and you would like to update an associated remote repository:

```
git push
```

Branching

If you are working on a project with a version being shared by many people, you may not want to edit that version.

You can create a branch (and switch to it) with this command:

```
git checkout -b branchname
```

List available branches:

```
git branch
```

List remote branches:

```
git branch -r
```

List all branches:

```
git branch -a
```

(The branch you are on is marked with an asterisk.)

Refresh local list of remote branches:

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

If you have a remote branch, and you need to create/sync with it locally, do the following.

First, list remote branches, and note the name of the branch you want to create locally:

```
git branch -a
```

If, for example, the remote branch is named ‘remotes/origin/dev’, you can then create it locally, sync it, and switch to it with the following command:

```
git checkout -t remotes/origin/dev
```

If you want control over the naming of the local tracking branch, you can specify it as follows:

```
git checkout -t -b dev remotes/origin/dev
```

If master (or main) changes while you’re making changes in a branch, you can merge them into the branch as follows:

```
git checkout main

git pull

git checkout <branch_name>

git merge main
```

To push the current branch to a remote repository, creating a new branch on the remote:

```
git push origin <remote_branch_name>
```

If you are currently in a local branch, and you know the branch also exists in a remote repository, but a git pull fails, you may need to link the local branch to the server branch. For example, if the local and server branches are both named “dev”, you’d link them as follows:

```
git branch --set-upstream-to=origin/dev dev
```

To switch back to the master branch:

```
git checkout master
```

Delete a local branch:

```
git branch -D <branch-name>
```

Delete a remote branch:

```
git branch -r -D <branch-name>
```

Security

To bypass SSL verification, add this entry to .bashrc:

```
alias git='git -c http.sslVerify=false'
```

Only use this if you *know* the network you're using can be trusted. This setting leaves you open to man-in-the-middle attacks.

Git Setup For Specific Hosting Sites

Git Setup for Bitbucket

Set up your local directory:

```
mkdir /path/to/your/project

cd /path/to/your/project

git init

git remote add origin
https://bitbucketid@bitbucket.org/bitbucketid/bitbucketproject.git
```

Create your first file, commit, and push:

```
echo "Your Name" >> contributors.txt

git add contributors.txt

git commit -m 'Initial commit with contributors'

git push -u origin master
```

Git Setup for GitHub (new)

```
mkdir -p ~/<projectname>

cd $_

echo "# <projectname>" >> README.md
```

```
git init

git add README.md

git commit -m "first commit"

git remote add origin https://github.com/<username>/<projectname>.git

git push -u origin master
```

Git Setup for GitHub (existing repository)

```
git remote add origin https://github.com/<username>/<projectname>.git

git push -u origin master
```

Automatic Authentication for GitHub, Bitbucket, and GitLab

Tired of being prompted for your userid and password every time you push local changes to your remote repository? Add the following to ~/.netrc.

For GitHub:

```
machine github.com
    login <username>
    password <password>
```

For Bitbucket:

```
machine bitbucket.org
    login <username>
    password <password>
```

For GitLab:

```
machine gitlab.com
    login <username>
    password <password>
```

(You can have all entries in the same file, if needed.)

More Git Tutorials and Reference

If you need more details, here are some comprehensive Git tutorials and references.

Official Tutorial – Hosted on the official Git site.

Pro Git – The entire book, hosted on the official Git site.

Vogella – A tutorial.

Atlassian Tutorials – A series of tutorials hosted on the Atlassian site. (These guys also host Bitbucket.)