Git Cheat Sheet

Git Overview

Git allows you to store all your files in a repository and it tracks all the changes between commits. You do not need a server to run git as it is simply a set of command line tools that can operate on your local file system.

Services such as Github provide a type of 'central' repository which you can push and pull code changes to. However, unlike alternatives such as SVN, as git is distributed you can have many 'central' repositories.

A good use case is to use Github or Bitbucket to store your repository in the cloud, clone it locally for development and push to a shared drive on your NAS for your own backups.

Repositories

Clone Repository
Make a full copy of the entire repository including branches and history.
```
git clone http://[username]@server/repo.git
```

Create a new local repository with working files
Create a new local repository with a working set of files.
```
git init
```

Create a new local repository with bare files
Create a new local repository no working files (i.e. it's just the repository).
```
git init --bare
```

User Management

Set your global user details
Set your display name and email address.
```
git config --global user.name "My Name"
git config --global user.email user@example.com
```

Reset the author of commit after updating global username and email
Do this after commiting and realising you haven't set you user name and email.
```
git commit --amend --reset-author
```

File Management

See the history of commits
```
git log
```

See the status of your files (staged/unstaged)
```
git status
```

Add a specific file to be staged in the next commit
```
git add some/path/to/file.txt
```

Add a all unstaged to be staged in the next commit
```
git add *
```

Remove a file
```
git rm some/path/to/file.txt
```

Commit staged files
```
git commit -m "Some comment"
```

Misc

Reset the working directory
Do this if you screw up your working directory and want to revert all changes.
```
git reset HEAD *
git clean -df & git checkout --
```

Revert a single file
```
git checkout path/to/file.txt
```

Terminology

Working set
These are the files on disk which you can edit using your normal software. After editing them you can choose to commit them to the repository.

Staged files
These are new or modified files that are marked to be included in the next commit. Unstaged files wont be committed.

Remotes
These are links to other repositories which you can fetch, push and pull with.

Branch
This is a fork of the repository where changes can be contained within. All repositories have a 'master' branch by default.

Remotes

Add a new remote repository
If you cloned your repository, you will have by default the 'origin' remote. But you can add any others.
```
git remote add remotename user@server/repo.git
```

Fetch changes from remote
You can fetch remote changes from all branches without merging.
```
git fetch remotename
```

Pull changes from remote
Pull performs a fetch of a single branch and merge in one operation.
```
git pull remotename branchname
```

Push your commits to the remote
You can push an individual branch to the remote.
```
git push remotename branchname
```

Branches

See what branches are available locally
You can see all the local branches.
```
git branch
```

See all the branches (including those fetched from a remote)
Remote branches use the format remotename/branchname
```
git branch -a
```

Create a new branch
This will base it on the branch you currently have checked out (master being default) and will keep your modified files for checking into the newly created branch. You will need to switch to it after creating.
```
git branch branchname
```

Switch to branch
You can switch to local branches just by using its name, but you can also switch to a remote branch by adding the remote name. You'll need to create a new local branch if you switch to a remote one (see next command).
```
git checkout [remotename/]branchname
```

Create local branch from remote branch
Do this after switching to remotename/branchname.
```
git checkout -b localbranchname
```

Merge changes from another branch into the working set
```
git merge [remotename/]branchname
```

Run a merge tool to resolve merge conflicts
```
git mergetool
```

Delete a remote branch
```
git push remotename --delete branchname
```

Is this useful? Spot an error or something missing, tweet to **@Martin_Adams**