Git Guide

Joe Rackham

1 Getting Started

1.1 Creating a repo

- `git init *dir_name*` creates a new directory suitable to work with git
- `git init` converts an un-versioned directory into one usable with git

2 Managing Git

2.1 Configuring

- `git config` is used to set configuration variables on a global or local level
- These variables are listed in `.gitconfig` text files.
- `--local` and `--global` are used to specify the visibility level; local is default
- `git config *var*` with report the var `git config *var* *value*` will update it

user.email	User's email address
core.editor	Editor to open if input is required
colour.ui	Used to adjust terminal output colouring

2.2 Aliases

- `git config alias.*abv* *cmd*` creates an alias for a command in the git config
- Note that the alias can't include the `git` command

3 Branching & Merging

3.1 Creating a new branch

- `git branch *bname*` creates a new branch
- `git checkout *bname*` begins working on that branch
- The first push has form `git push origin *orig_branch*` to set the upstream for the branch

3.2 Fast-Forward Merging

Can ONLY be use when there is a linear path to the branch head from the parent head:

```
git checkout *parent_branch*
git merger *child_branch*
```

4 Workflow

4.1 Committing

- `git resest` will reset the changes staged for a commit
- The `--ammend` option will update the previous commit instead of creating a new one

4.2 Stashing

- `git stash` will store all changes made since the last commit for a later time
- `git stash pop` will restore these changes and remove them from the stash
- `git stash apply` will restore the changes and keep them in the stash
- Stashing ignores un-tracked and ignored files
- `git stash save *msg*` will stash changes with an annotation

4.3 Tracking Changes

- `git log` details recent changes to the repo
- `git blame` identifies the author associated with a specific committed line

4.4 Undoing Changes

Checkout:

- `git checkout *hash*` sets the working state to that of a previous commit, the head pointer isn't moved
- This puts us in a detached HEAD state where we aren't working on any branch
- `git checkout -b *new_bname*` will move any work we do to a new branch so it is not garbage collected

Revert:

- `git revert HEAD` will create a new commit with the inverse of the last commit

Reset:

- `git reset --hard` removes the previous commit from the history

4.5 Maintenance

- `git clean` will delete any un-tracked files; option `-n` can be used to do a 'dry-run'
- `git rm` removes a file from the working directory and the git index

4.6 Rewriting History

- `git rebase` facilitates changing multiple or older commits
- Public commits should never be rebased
- `git squash` uses rebasing to combine multiple commits into one for a clean