Create

From existing directory
```
cd <project_directory>
git init
git add .
```

Or using single command
```
git init <project_name>
```

From existing repo in the filesystem
```
git clone local_repo new_local_repo
```
existing should be creted with `--bare` option

From remote location
```
git clone you@host.org:dir/project.git
```

Browse

State of working directory/current branch etc
```
git status
```
Or minimalistic version with -sb flag

Changes to tracked files
```
git diff [--word-diff]
```

Changes between commit1 and commit2
```
git diff <COMMIT1_HASH> <COMMIT2_HASH>
```

History of changes
```
git log [-p]
```

Who changed what and when in a file
```
git blame <file>
```

A commit identified by COMMIT ID
```
git show <HASH>
```

A specific file from a specific ID
```
git diff <HASH>:<FILE>
```

Search for patterns
```
git grep <pattern> [path]
```

Change using your editor of choice

```
git add <file>
```
Starts tracking new file or add changes to staging area

Recovery

Return to the previous commit
```
git reset [--hard] HEAD^
```
you cannot undo a hard reset (unless you know reflog ;))

Revert the last commit
```
git revert HEAD
```
Creates a new commit

Revert specific commit
```
git revert <HASH>
```
Creates a new commit

Fix the last commit
```
git commit --amend
```
Lets you add modification from staging and change msg

Checkout the particular version of a file
```
git checkout <HASH> <file>
```

Branching

List all branches
```
git branch
```

Switch to the BRANCH branch
```
git checkout <BRANCH>
```

Merge branch B1 into branch B2
```
git checkout <B2>
git merge <B1>
```

Create branch based on HEAD
```
git branch <BRANCH>
```

Create branch based on another
```
git branch <new_branch> <base>
```

Delete a branch
```
git branch -d <branch>
```

Update

Fetch latest changes from the remote repository
```
git fetch
```
this just creates remote branches in your local repository, but does not merge them back to your own branches

Pull latest changes from the remote
```
git pull
```
does a fetch followed by a merge

Apply a patch that someone sent you
```
git am -3 patch.mbox
```
In case of conflict, resolve the conflict and:
```
git am --resolve
```

Commit

Commit all local changes
```
git commit -a
```
optionally with –m"message" to skip commit editor

Publish

Push changes to origin
```
git push [origin] [branch]
```

Mark a version or a milestone
```
git tag <version_name>
```

Prepare a patch for other developers
```
git format-patch origin
```

git

Cambridge Technology Partners

/ch/open

Remotes

```
git remote add <remote> <remote_URL>
```
adds a remote repository. Can be then fetched locally.

Example:
```
git remote add coreteam
git://github.com/wycats/merb-plugins.git
git fetch coreteam
```

```
git remote -v
```
Lists current remotes

```
git remote show <remote>
```
shows information about the remote server.

```
git remote rm <remote>
```
removes selected remote

```
git push <remote> :<branch>
```
deletes a branch in a remote repository

```
git push <remote> <remote>:<remote_branch>
```
creates a branch on a remote repository

Example:
```
git push origin origin:new_feature_name
```

```
git remote prune <remote>
```
prunes deleted remote-tracking branches

Stashing

```
git stash
```

```
git stash save <optional-name>
```
saves your local modifications to a new stash

```
git stash apply
```
restores the changes recorded in the stash on top of the current working tree state

```
git stash drop [<stash-name>]
```
deletes the stash git stash clear delete all current stashes

```
git stash pop
```
restores the changes from the most recent stash, and drops it from the stack of stashed changes

```
git stash list
```
lists all current stashes

```
git stash show <stash-name> -p
```
shows the content of a stash

Useful tips

<TAB> (autocompletion) is your friend

When in doubt:
```
git help [command]
```

More details in the log
```
git log --oneline --decorate
```

Graphical log
```
git log --graph --abbrev-commit
```

Push branch to remote
```
git push <origin> <branch>
```

Delete remote branch
```
git push <origin> :<branch>
```

Staging all changes (including rm)
```
git add -A .
```

Interactive stating – walk through all changes in the files
```
git add -p
```

Configuration

```
git config [--local|--global|--system]
```
`--local`: (default) local repository config
`--global` is unique per user
`--system` is os-wide

User data
```
user.name $name
user.email $email
```
mandatory before we can start any work with git

Resolve merge conflicts

View merge conflicts
```
git diff
```
View merge conflicts against base file
```
git diff --base <FILE>
```
View merge conflicts against other changes
```
git diff --theirs <FILE>
```
View merge conflicts against your changes
```
git diff --ours <FILE>
```
After resolving conflicts, merge with
```
git add <CONFLICTING_FILE>
git rebase --continue
```