GRIGALE

GIT Pocket guide (ver 1.0) **www.grigale.com©**

First Time GIT SETUP

git config --list – Shows current configuration settings.
git config user.name – Shows current username First and Last name configuration settings.
git config --global user.name "Haim Tzadok" – Configures GIT username First and Last name credentials.
git config --global user.email "haim.tzadok@grigale.com" – Configures GIT username email credentials.
git config --global core.editor emacs – Configures default editor.
git config --global color .ui true – Enables default terminal colors.
git config --global merge.tool vimdiff – Configures default merge tool to be vimdiff.

Getting help

git help <sub-command> – Shows available help on <sub-command>.
git <sub-command> --help – Shows available help on <sub-command>.
man git-<sub-command> – Shows available help on <sub-command>.

Creating GIT Repository

git init – Initializes a repository in the current directory.
git clone <url> – Clones a repository that resides on a url.
URL can be:
http://
ssh://
git://

Basic commands

Legend:
Untracked/Unstaged files – Files that are not reported as known to GIT.
Tracked/Staged files – Files that are known to GIT but not committed yet.
Committed files – Files that are committed and are part of the GIT repository.
git status – check status of your files.
git add file1 – Adds a file to the staging area (Stages a file that was added).
git rm file1 – removes a file from the repository and also deletes it from the staging area.
git rm --cached file1 – removes a file from the repository but does not delete it from the working area.
git mv file1 file2 – renames a file from the name file1 to be file2.
git status – View staged and unstaged changes.
git diff – Shows different between staged and unstaged areas.
git commit -m "Initial commit" – commits the staged changes with a message - "Initial commit" .
git commit -a -m "Initial commit" – stages and commits all the changes with a message - "Initial commit".
git log – shows the commit history.
git log --pretty =oneline – shows the commit history in one line.
git log --pretty =format:"%h -%an, %ar:%s" – shows the commit history in special format.
git commit --ammend – add another things that were staged to the last commit.
git reset HEAD readme – unstages the file readme from the staging area.

Remote operations

git remote – shows the available remote servers.
git remote add sc git://grigale.com/sc – adds the remote name – "sc" with the url – git://grigale.com/sc to the available remote servers.
git fetch sc – fetches changes from the remote name – "sc".
git pull sc – fetches and updates changes from the remote name – "sc" to the local branch that tracking it.
git push sc – Updates changes from local branch to the remote branch called – "sc".
git remote show sc– inspect the remote sc.
git remote rename sc origsc– renames the remote sc to be called origsc.
git remote rm origsc– deletes the remote called origsc.

TAG operations

git tag – shows available tags.
git tag v1.4– create a lightweight tag called v1.4.
git tag -a v1.4 -m "Version 1.4 "– creates an annotated tag called v1.4. (Annotated tag has description and explicit creator).
gpg --list-keys; gpg --gen-keys; git tag -s v1.4 -m "Version 1.4 "– creates a signed tag called v1.4.
git tag -v v1.4– verify a tag called v1.4.
git tag -a v1.4 -m "Version 1.4 " 9cfeb03– creates an annotated tag called v1.4 on an older commit – 9cfeb03.
git push origin v1.4 – Updates changes from local tag to the remote branch.
git push origin --tags – Updates changes from local tags to the remote branch.

Available aliases

$ git config --global alias.co checkout
$ git config --global alias.br branch
$ git config --global alias.ci commit
$ git config --global alias.st status
$ git config --global alias.unstage 'reset HEAD --'
$ git config --global alias.last 'log -1 HEAD'
$ git config --global alias.visual '!gitk'

Branches

A branch in Git is simply a lightweight movable pointer to another commit. The default branch name in Git is master. As you initially make commits, you're given a master branch that points to the last commit you made. Every time you commit, it moves forward automatically.
git branch – shows available branches. (The branch that has star before it, is the branch that we are working on.)
git branch testing – creates a branch called - testing.
git branch master – switches to a branch called - master.
git checkout -b testing – creates and switches to a branch called – testing.
git checkout master; git merge hotfix – merges hotfix branch into the master branch.
git branch -d testing – deletes the branch testing.
git branch --merged – shows which branches are already merged with the current branch.
git branch --no-merged – shows which branches are not merged with the current branch.

Stash operations

Stash enables saving your modified and staged data to a stack area, in order to allow switching between branches.

git stash – stashes the latest changes into the stack.
git stash list – shows all stashes.

git stash apply– applies the latest stash (stash@0) back to the working area.
git stash apply stash@{2}– applies a specific stash (stash@2) back to the working area.
git stash drop stash@{2}– removes the specific stash (stash@2) from the stack.
git stash branch testchanges– create a branch from the latest stash (stash@0) back to the branch.

Rewriting history

Many times, when working with Git, you may want to revise your commit history for some reason. One of the great things about Git is that it allows you to make decisions at the last possible moment.

git commit --amend – modifies the latest commit message.
git rebase -i HEAD~3 – modifies the last 3 commit messages.