

Overview

When you first setup Git, set up your user name and email address so your first commits record them properly.

```
git config --global user.name "My Name"
git config --global user.email "user@email.com"
```

About Git, GitHub and Heroku.

Git is a free and open source, distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

GitHub is the best way to collaborate around your code. Fork, send pull requests, and manage all your public and private git repositories.

Heroku is a cloud application platform that supports a number of different programming languages including Java, Ruby, Node.js, and Clojure—it's a new way of building and deploying web apps.

Basic Git Workflow Example

Initialize a new git repository, then stage all the files in the directory and finally commit the initial snapshot.

```
$ git init
$ git add .
$ git commit -m 'initial commit'
```

Create a new branch named featureA, then check it out so it is the active branch. then edit and stage some files and finally commit the new snapshot.

```
$ git branch featureA
$ git checkout featureA
$ (edit files)
$ git add (files)
$ git commit -m 'add feature A'
```

Amend a commit

```
$ git commit --amend
$ git push -f origin master
```

Switch back to the master branch, reverting the featureA changes you just made, then edit some files and commit your new changes directly in the master branch context.

```
$ git checkout master
$ (edit files)
$ git commit -a -m 'change files'
```

Merge the featureA changes into the master branch context, combining all your work. Finally delete the featureA branch.

```
$ git merge featureA
$ git branch -d featureA
```

Setup & Init

Git configuration, and repository initialization and cloning.

<code>git config [key] [value]</code>	set a config value in this repository
<code>git config --global [key] [value]</code>	set a config value globally for this user
<code>git init</code>	initialize an existing directory as a Git repository
<code>git clone [url]</code>	clone a Git repository from a URL
<code>git help [command]</code>	get help on any Git command

Stage & Snapshot

Working with snapshots and the Git staging area.

<code>git status</code>	show the status of what is staged for your next commit and what is modified in your working directory
<code>git add [file]</code>	add a file as it looks now to your next commit (stage)
<code>git reset [file]</code>	reset the staging area for a file so the change is not in your next commit (unstage)
<code>git diff</code>	diff of what is changed but not staged
<code>git diff --staged</code>	diff of what is staged but not yet committed
<code>git commit</code>	commit your staged content as a new commit snapshot
<code>git rm [file]</code>	remove a file from your working directory and unstage
<code>git gui</code>	tcl/tk GUI program to make all of these commands simpler

Branch & Merge

Working with Git branches and the stash.

<code>git branch</code>	list your branches. a * will appear next to the currently active branch
<code>git branch [branch-name]</code>	create a new branch at the current commit
<code>git checkout [branch]</code>	switch to another branch and check it out into your working directory
<code>git checkout -b [branch]</code>	create a branch and immediately switch to it
<code>git merge [branch]</code>	merge another branch into your currently active one and record the merge as a commit
<code>git log</code>	show commit logs
<code>git stash</code>	stash away the currently uncommitted modifications in your working directory temporarily
<code>git stash apply</code>	re-apply the last stashed changes

Share & Update

Fetching, merging and working with updates from another repository.

<code>git remote add [alias] [url]</code>	add a git URL as an alias
<code>git fetch [alias]</code>	fetch down all the branches from that Git remote
<code>git merge [alias]/[branch]</code>	merge a branch on the server into your currently active branch to bring it up to date
<code>git push [alias] [branch]</code>	push the work on your branch to update that branch on the remote git repository
<code>git pull</code>	fetch from the URL tracked by the current branch and immediately try to merge in the tracked branch

Inspect & Compare

Examining logs, diffs and object information.

<code>git log</code>	show the commit history for the currently active branch
<code>git log branchB..branchA</code>	show the commits on branchA that are not on branchB
<code>git log --follow [file]</code>	show the commits that changed file, even across renames
<code>git diff branchB...branchA</code>	show the diff of what is in branchA that is not in branchB
<code>git show [SHA]</code>	show any object in Git in human-readable format
<code>gitx</code>	gitx is only for MacOS
<code>gitk</code>	Generic program for browsing commits
<code>git-gui</code>	Gui tool for making commits

Contributing on GitHub

To contribute to a project hosted on GitHub you can fork the project on github.com, then clone your fork locally, make a change, push back to GitHub, and then send a pull request, which will email the maintainer.

```
fork project on github
$ git clone https://github.com/my-user/project
$ cd project
$ (edit files)
$ git add (files)
$ git commit -m 'Explain what I changed'
$ git push origin master
```

go to github and click 'pull request' button

```
Get list of remotes configured
$ git remote -v
```

Working with Private Spaces

```
Create a private space
$ heroku spaces:create my-space-name --org my-org-name
```

```
Private space in a specific region
$ heroku spaces:create my-space-name --org my-org-name --region tokyo
```

```
View info about Private space
$ heroku spaces:info --space my-space-name
```

```
List of trusted IP Addresses
$ heroku trusted-ips --space acme-prod
```

```
Destroy a Private Space
$ heroku spaces:destroy --space my-space-name
```

```
Managing apps in private space
Create an app in a private space
$ heroku create my-space-app --space my-space-name
```

```
List apps in a private space
$ heroku apps --space my-space-name
```

```
Destroy an App
$ heroku apps:destroy -a my-space-app
```

For other cheatsheets: <http://developer.force.com/cheatsheets>

Deploying to Heroku with Git

```
Use the heroku command-line tool to create an application and git remote:
$ heroku login
$ heroku create
```

```
[Creating glowing-dusk-965... done, stack is bamboo-mri-1.9.2
http://glowing-dusk-965.herokuapp.com/ <http://glowing-dusk-965.herokuapp.com/> | git@heroku.com:glowing-dusk-965.git <x-msg://536/git@heroku.com:glowing-dusk-965.git> Git remote heroku added]
$ heroku create my-app Creates a heroku app with name "my-app"
```

```
Use SSH while creating a heroku app
$ heroku create --ssh-git
```

```
Use git to deploy the application.
$ git push heroku master
```

```
Create an additional Heroku app for staging, and name the git remote "staging".
$ heroku create my-staging-app --remote staging
```

```
Use git to deploy the application via the staging remote.
$ git push staging master
```

```
Rename an app
$ heroku apps:rename newname
```

```
Clone an existing heroku app
$ heroku git:clone -a myapp
```

```
Developing Apps in a Org
heroku create --org my-org
```

```
Leave an App
heroku apps:leave --app my-website
```

```
Specify Default Org
export HEROKU_ORGANIZATION=my-org
```

Heroku Postgres Commands

```
Start a database
$ heroku addons:add heroku-postgresql
$ heroku pg:promote HEROKU_POSTGRESQL_PURPLE_URL
```

```
Enable backups for PostgreSQL
$ heroku addons:add pgbackups:auto-month
```

```
See List of all Postgres DBs
$ heroku pg:info
```

```
Establish psql session
$ heroku pg:psql
```

```
The pg:ps command queries the pg_stat_statements table in Postgres to give a concise view into currently running queries
$ heroku pg:ps
```

Configure Http Proxy for CLI

```
$ export HTTP_PROXY=http://proxy.server.com:portnumber
or
$ export HTTPS_PROXY=https://proxy.server.com:portnumber
$ heroku login
```

```
Heroku Redis
$ heroku plugins:install heroku-redis
```

```
Heroku Redis attached to an application
$ heroku addons:create heroku-redis:hobby-dev -a sushi
```

```
Check REDIS_URL
$ heroku config | grep REDIS
```

```
Set Primary Instance
$ heroku redis:promote HEROKU_REDIS_JADE
```

Miscellaneous Commands

```
Transfer app to your org
heroku apps:transfer my-org --app app-one
```

```
Get list of Heroku releases
$ heroku releases
```

```
Run Terminal
$ heroku run bash
```

```
Restart Dynos
$ heroku restart
```