Git links

# <https://guides.github.com/activities/hello-world/>

## create a new repository

create a new directory, open it and perform a   
**git init**  
to create a new git repository.

## checkout a repository

create a working copy of a local repository by running the command  
**git clone /path/to/repository**  
when using a remote server, your command will be  
**git clone username@host:/path/to/repository**

## workflow

your local repository consists of three "trees" maintained by git. the first one is your Working Directory which holds the actual files. the second one is the Index which acts as a staging area and finally the HEAD which points to the last commit you've made.



## add & commit

You can propose changes (add it to the **Index**) using  
**git add <filename>  
git add \***  
This is the first step in the basic git workflow. To actually commit these changes use  
**git commit -m "Commit message"**  
Now the file is committed to the **HEAD**, but not in your remote repository yet.

## pushing changes

Your changes are now in the **HEAD** of your local working copy. To send those changes to your remote repository, execute   
git push origin master  
Change *master* to whatever branch you want to push your changes to.   
  
If you have not cloned an existing repository and want to connect your repository to a remote server, you need to add it with  
**git remote add origin <server>**  
Now you are able to push your changes to the selected remote server

## branching

Branches are used to develop features isolated from each other. The *master* branch is the "default" branch when you create a repository. Use other branches for development and merge them back to the master branch upon completion.



create a new branch named "feature\_x" and switch to it using  
**git checkout -b feature\_x**  
switch back to master  
**git checkout master**  
and delete the branch again  
**git branch -d feature\_x**  
a branch is *not available to others* unless you push the branch to your remote repository  
**git push origin <branch>**

## update & merge

to update your local repository to the newest commit, execute   
**git pull**  
in your working directory to *fetch* and *merge* remote changes.  
to merge another branch into your active branch (e.g. master), use  
**git merge <branch>**  
in both cases git tries to auto-merge changes. Unfortunately, this is not always possible and results in *conflicts*. You are responsible to merge those *conflicts* manually by editing the files shown by git. After changing, you need to mark them as merged with  
**git add <filename>**  
before merging changes, you can also preview them by using  
**git diff <source\_branch> <target\_branch>**

## tagging

it's recommended to create tags for software releases. this is a known concept, which also exists in SVN. You can create a new tag named *1.0.0* by executing  
git tag 1.0.0 1b2e1d63ff  
the *1b2e1d63ff* stands for the first 10 characters of the commit id you want to reference with your tag. You can get the commit id by looking at the...

## log

in its simplest form, you can study repository history using.. git log  
You can add a lot of parameters to make the log look like what you want. To see only the commits of a certain author:  
**git log --author=bob**  
To see a very compressed log where each commit is one line:  
**git log --pretty=oneline**  
Or maybe you want to see an ASCII art tree of all the branches, decorated with the names of tags and branches:   
**git log --graph --oneline --decorate --all**  
See only which files have changed:   
**git log --name-status**  
These are just a few of the possible parameters you can use. For more, see git log --help

## replace local changes

In case you did something wrong, which for sure never happens ;), you can replace local changes using the command  
**git checkout -- <filename>**  
this replaces the changes in your working tree with the last content in HEAD. Changes already added to the index, as well as new files, will be kept.

If you instead want to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it like this  
**git fetch origin**  
**git reset --hard origin/master**

## useful hints

built-in git GUI  
gitk  
use colorful git output  
git config color.ui true  
show log on just one line per commit  
git config format.pretty oneline  
use interactive adding  
git add -i

# Basic Git commands

Here is a list of some basic Git commands to get you going with Git.

 for a visual introduction to Git commands and workflows, including examples.

|  |  |  |
| --- | --- | --- |
| **Git task** | **Notes** | **Git commands** |
| [Tell Git who you are](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-config) | Configure the author name and email address to be used with your commits.  Note that Git [strips some characters](http://stackoverflow.com/questions/26159274/is-it-possible-to-have-a-trailing-period-in-user-name-in-git/26219423" \l "26219423) (for example trailing periods) from user.name. | git config --global user.name "Sam Smith"  git config --global user.email sam@example.com |
| [Create a new local repository](http://atlassian.com/git/tutorial/git-basics" \l "!init) |  | git init |
| [Check out a repository](http://atlassian.com/git/tutorial/git-basics" \l "!clone) | Create a working copy of a local repository: | git clone /path/to/repository |
| For a remote server, use: | git clone username@host:/path/to/repository |
| [Add files](http://atlassian.com/git/tutorial/git-basics" \l "!add) | Add one or more files to staging (index): | git add <filename>  git add \* |
| [Commit](http://atlassian.com/git/tutorial/git-basics" \l "!commit) | Commit changes to head (but not yet to the remote repository): | git commit -m "Commit message" |
| Commit any files you've added with git add, and also commit any files you've changed since then: | git commit -a |
| [Push](http://atlassian.com/git/tutorial/remote-repositories" \l "!push) | Send changes to the master branch of your remote repository: | git push origin master |
| [Status](http://atlassian.com/git/tutorial/git-basics" \l "!status) | List the files you've changed and those you still need to add or commit: | git status |
| [Connect to a remote repository](http://atlassian.com/git/tutorial/remote-repositories" \l "!remote) | If you haven't connected your local repository to a remote server, add the server to be able to push to it: | git remote add origin <server> |
| List all currently configured remote repositories: | git remote -v |
| [Branches](http://atlassian.com/git/tutorial/git-branches) | Create a new branch and switch to it: | git checkout -b <branchname> |
| Switch from one branch to another: | git checkout <branchname> |
| List all the branches in your repo, and also tell you what branch you're currently in: | git branch |
| Delete the feature branch: | git branch -d <branchname> |
| Push the branch to your remote repository, so others can use it: | git push origin <branchname> |
| Push all branches to your remote repository: | git push --all origin |
| Delete a branch on your remote repository: | git push origin :<branchname> |
| [Update from the remote repository](http://atlassian.com/git/tutorial/remote-repositories) | Fetch and merge changes on the remote server to your working directory: | git pull |
| To merge a different branch into your active branch: | git merge <branchname> |
| View all the merge conflicts:  View the conflicts against the base file:  Preview changes, before merging: | git diff  git diff --base <filename>  git diff <sourcebranch> <targetbranch> |
| After you have manually resolved any conflicts, you mark the changed file: | git add <filename> |
| **Tags** | You can use tagging to mark a significant changeset, such as a release: | git tag 1.0.0 <commitID> |
| CommitId is the leading characters of the changeset ID, up to 10, but must be unique. Get the ID using: | git log |
| Push all tags to remote repository: | git push --tags origin |
| [Undo local changes](http://atlassian.com/git/tutorial/undoing-changes) | If you mess up, you can replace the changes in your working tree with the last content in head:  Changes already added to the index, as well as new files, will be kept. | git checkout -- <filename> |
| Instead, to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it, do this: | git fetch origin  git reset --hard origin/master |
| **Search** | Search the working directory for foo(): | git grep "foo()" |