<https://dzone.com/articles/reset-checkout-and-revert> bvn god

<https://www.atlassian.com/git/tutorials/learn-undoing-changes-with-bitbucket> bvn

<https://www.atlassian.com/git/tutorials/resetting-checking-out-and-reverting>

https://www.atlassian.com/git/tutorials/advanced-overview

<https://www.atlassian.com/git/tutorials/undoing-changes>

<https://git-scm.com/book/en/v2/Git-Tools-Reset-Demystified> bvn

<https://docs.microsoft.com/en-us/vsts/git/tutorial/undo?view=vsts&tabs=visual-studio>

<https://docs.gitlab.com/ee/topics/git/numerous_undo_possibilities_in_git/> bvn

<https://sethrobertson.github.io/GitFixUm/fixup.html> bvn#

<https://blog.github.com/2015-06-08-how-to-undo-almost-anything-with-git/> bvn#

<https://git-scm.com/book/en/v2/Git-Basics-Undoing-Things> ####

Rpts:

git revertshould be used to undo changes on a public branch, and git reset should be reserved for undoing changes on a private branch.

You can also think of git revert as a[tool](https://bitbucket.org/product?utm_source=dzone&utm_medium=paid-content&utm_content=resetting-checking-out-and-reverting&utm_campaign=bitbucket_adexp-bbtofu_dzone-syn-content) for undoing committed changes, while git reset HEAD is for undoing uncommitted changes.

Like git checkout, git revert has the potential to overwrite files in the working directory, so it will ask you to commit or stash changes that would be lost during the revert operation.

Git tips commands:

**Step: Configure Git**

Once it is installed, open **terminal** (aka Bash, aka Shell, aka Prompt). You can verify that it's really there by typing:

$ git --version

This will return the version of Git that you're running and look something like this:

git version 1.9.1

(Any version 1.7.10 or higher is fine.)

Next, configure Git so it knows who to associate your changes to:

Set your name:

$ git config --global user.name "<Your Name>"

Now set your email:

$ git config --global user.email "<youremail@example.com>"

## 2)Step: Create a Repository(in pc using **command-line interface (CLI)**)

**Tips**

* **Make a new folder (aka directory)**

$ mkdir <FOLDERNAME>

* **Navigate into an existing folder (aka change directory)**

$ cd <FOLDERNAME>

* **List the items in a folder**

$ ls

* **Turn Git on for a folder**

$ git init

## 3) Commit To It

**Tips**

* **Check status of changes to a repository**
* $ git status
* **View changes to files**
* $ git diff
* **Add a file's changes to be committed**
* $ git add <FILENAME>
* **To add all files changes**
* $ git add .
* **To commit (aka save) the changes you've added with a short message describing the changes**
* $ git commit -m "<your commit message>"

## 4) GitHubbin(Create a GitHub account, add username to your Git config.)

## 5) Remote Control

**Tips**

* **Add remote connections**

$ git remote add <REMOTENAME> <URL>

* **Set a URL to a remote**

$ git remote set-url <REMOTENAME> <URL>

* **Pull in changes**

$ git pull <REMOTENAME> <BRANCHNAME>

* **View remote connections**

$ git remote -v

* **Push changes**

$ git push <REMOTENAME> <BRANCH>

**6)Forks And Clones Tips**

* **Add remote connections**
* $ git remote add <REMOTENAME> <URL>
* **View remote connections**
* $ git remote -v

## 7) Branches Aren't Just For Birds

**Tips**

* You can create and switch to a branch in one line:

$ git checkout -b <BRANCHNAME>

* Create a new branch:

$ git branch <BRANCHNAME>

* Move onto a branch:

$ git checkout <BRANCHNAME>

* List the branches:

$ git branch

* Rename a branch you're currently on:

$ git branch -m <NEWBRANCHNAME>

* Verify what branch you're working on

$ git status

## 9)Pull Never Out Of Date

**TIPS**

* **Check Git status**

$ git status

* **Pull in changes from a remote branch**

$ git pull <REMOTENAME> <REMOTEBRANCH>

* **See changes to the remote before you pull in**

$ git fetch --dry-run

## 10)Requesting You Pull Please

## 11)Merge Tada

**Tips**

* **Merge a branch into current branch**

$ git merge <BRANCHNAME>

* **Change the branch you're working on**

$ git checkout <BRANCHNAME>

* **Delete a local branch**

$ git branch -d <BRANCHNAME>

* **Delete a remote branch**

$ git push <REMOTENAME> --delete <BRANCHNAME>

* **Pull from a remote branch**

$ git pull <REMOTENAME> <BRANCHNAME>

## --

**Git Reset vs Revert vs Checkout reference**

The table below sums up the most common use cases for all of these commands. Be sure to keep this reference handy, as you’ll undoubtedly need to use at least some of them during your Git career.

| **Command** | **Scope** | **Common use cases** |
| --- | --- | --- |
| git reset | Commit-level | Discard commits in a private branch or throw away uncommited changes |
| git reset | File-level | Unstage a file |
| git checkout | Commit-level | Switch between branches or inspect old snapshots |
| git checkout | File-level | Discard changes in the working directory |
| git revert | Commit-level | Undo commits in a public branch |
| git revert | File-level | (N/A) |