

Git

Presenter: Haotao (Eric) Lai

Contact: haotao.lai@gmail.com

Acknowledge

- images with white background is from the following link:
<http://marklodato.github.io/visual-git-guide/index-en.html>
- images with transparent background is from a book named “*Pro Git*”:
<https://git-scm.com/book/en/v2>
- images in the workflow section is from the atlassian git tutorial:
<https://www.atlassian.com/git/tutorials>

History

Git ([/git/](#)^[7]) is a **version control system** for tracking changes in **computer files** and coordinating work on those files among multiple people. It is primarily used for source code management in **software development**,^[8] but it can be used to keep track of changes in any set of files. As a **distributed revision control** system it is aimed at speed,^[9] data integrity,^[10] and support for distributed, non-linear workflows.^[11]

Git was created by **Linus Torvalds** in 2005 for development of the **Linux kernel** with other kernel developers contributing to its initial development.^[12] Its current maintainer since 2005 is **Junio Hamano**.

As with most other distributed version control systems, and unlike most **client–server** systems, every Git **directory** on every **computer** is a full-fledged **repository** with complete history and full version tracking abilities, independent of network access or a central server.^[13]

Git is **free software** distributed under the terms of the **GNU General Public License** version 2.

----- from wiki

Setting up a repository

git init, git clone, git config

Preparation

- I assume you already know how to install it.
- First time you use it, need to do some configuration job.
- config (in a Mac):
 - /etc/gitconfig, --system
 - ~/.gitconfig, --global
 - specific_git_repo/.git/config
- `$ git config --global user.name "Eric Lai"`
- `$ git config --global user.email haotao.lai@gmail.com`

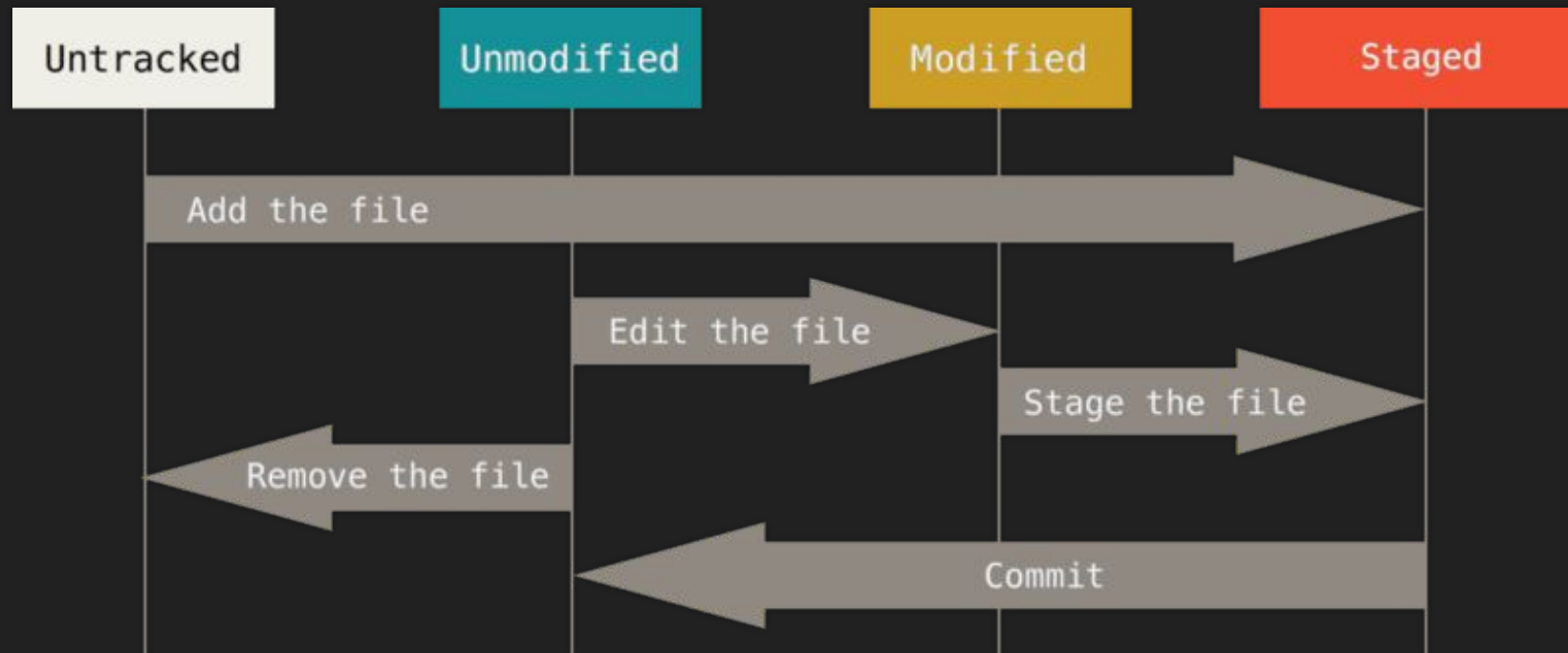
Get / Create a Git Repository

- get a repo from a remote server like GitHub or Bitbucket
 - you need to use "**git clone [remote-url] [local-path]**" command
- create a local Git repo
 - you need to use "**git init**" command in the working directory
- you can find a directory in any Git repo named ".git/", that's the place all the magic happened

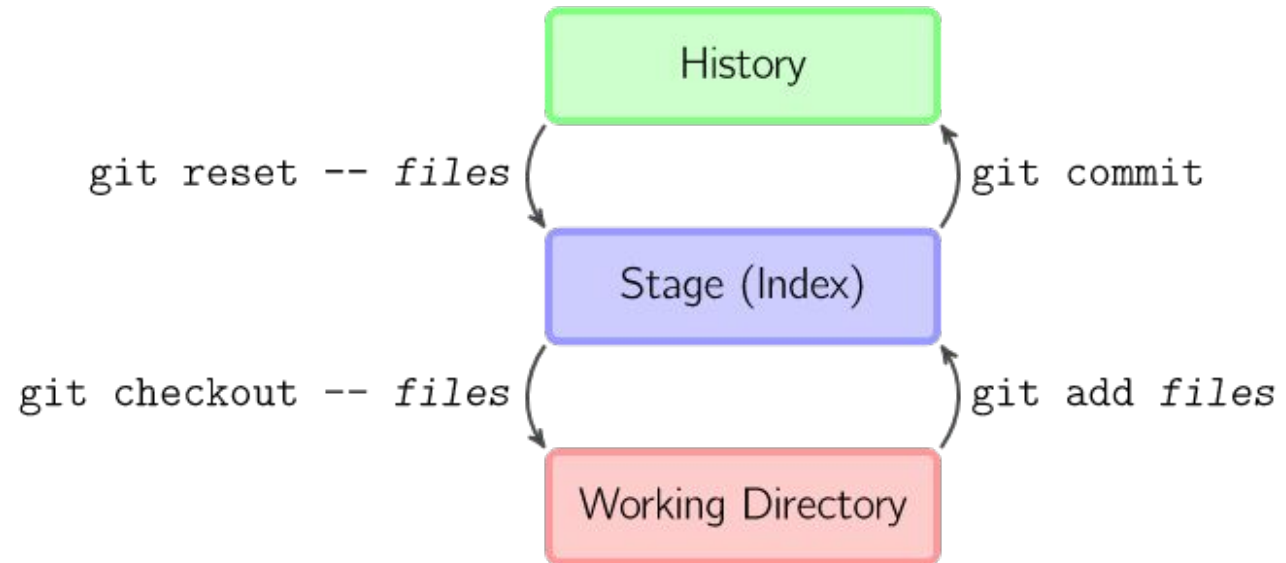
Saving changes

git add, git commit, git stash, .gitignore

File Status Lifecycle



Three Areas



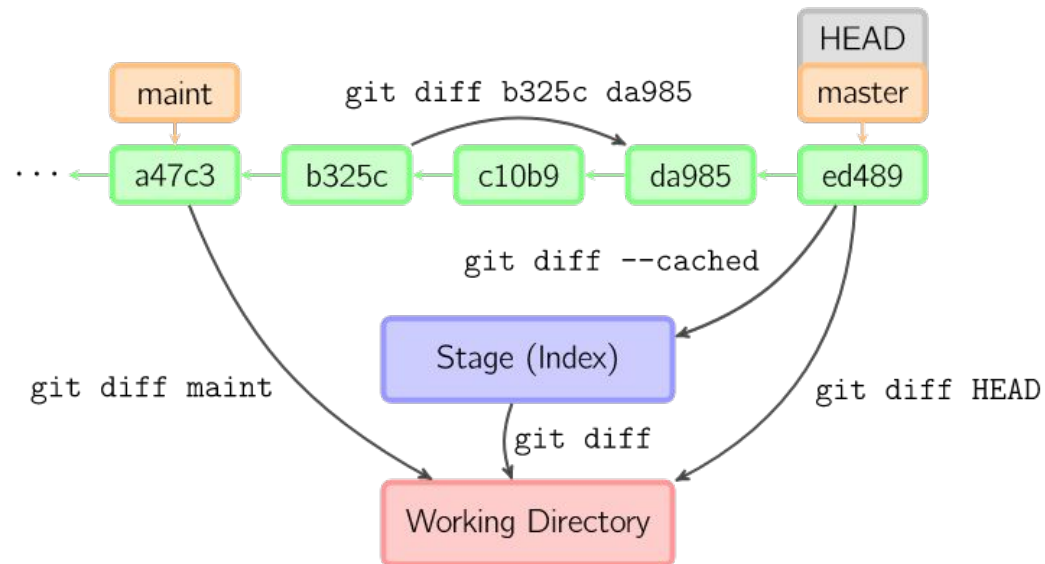
Git Stage Area

- add file into stage area
 - use “**git add file_name**” command

Remove File

- remove file from working directory
 - use “**git rm file_name**” command
- remove file only from version control system (don't trace them anymore)
 - use “**git rm --cached file_name**” command

Check Difference



Take a Snapshot (1)

- save a version of your work
 - use “**git commit**” command
 - **git commit -m “commit message”** (put commit message in a line)
 - **git commit** (vim will be opened to ask you input commit message)
- every time you commit a “version”, you have a chance to come back

Take a Snapshot (2)

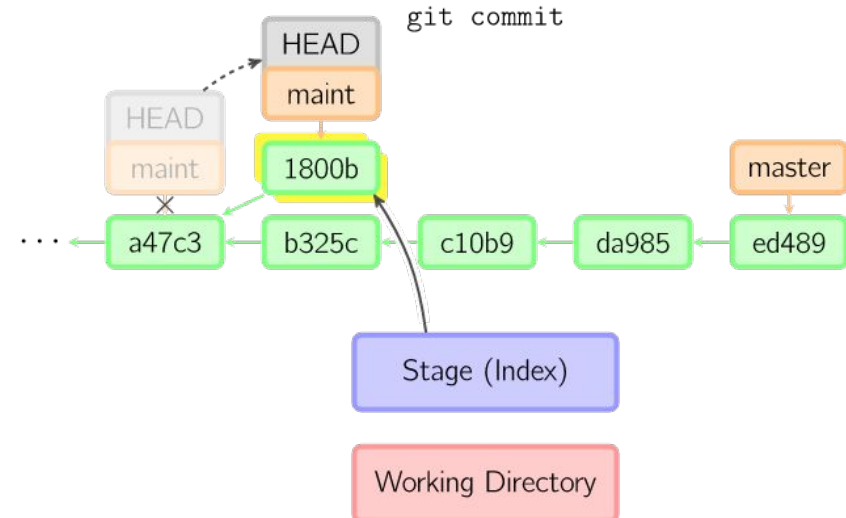
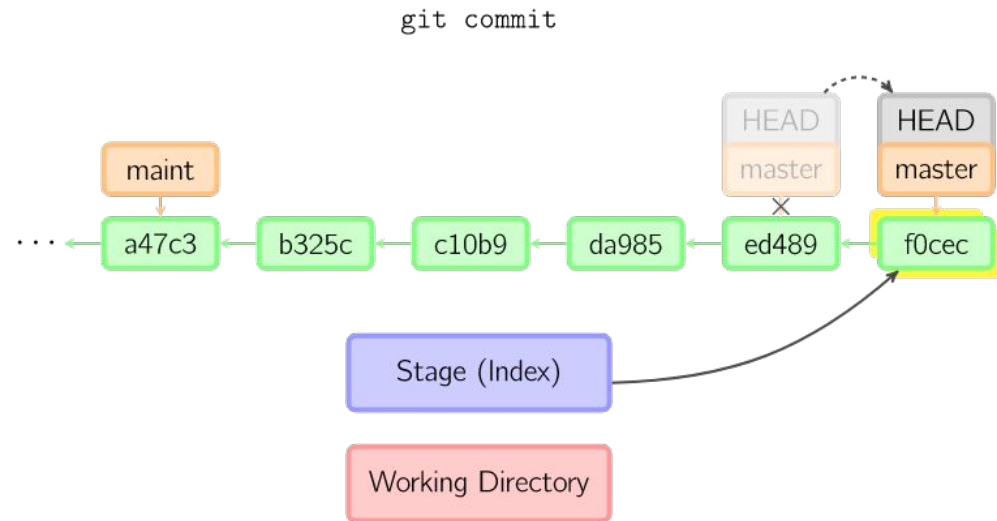


image from: <http://marklodato.github.io/visual-git-guide/index-zh-cn.html>

Temporary save your work

- Need to jump to other emergency work immediately, but haven't finish your current work
 - use "**git stash**" command
- Get my work back
 - use "**git stash pop**" or "**git stash apply**"

But, by default Git will NOT stash the untracked files or ignore files.

Inspecting a repository

git status, git log

Undo changes

git checkout, git revert, git reset, git clean

Checkout

- **git checkout <commit>**
- **git checkout <commit> <file>**
- **git checkout <branch>**
 - checkout file from history or stage area
 - switch between branch

Revert vs. Reset

revert: using a new commit to undo the previous commit

- **git revert <commit>**

reset: when you undo with git reset (and the commits are no longer referenced by any ref or the reflog), there is no way to retrieve the original copy—it is a permanent undo.

- **git reset <commit>**

Clean

The git clean command removes untracked files (DANGEROUS !!!) from your working directory.

Collaborating

Remote Repository (1)

- `git remote add <remote_repo_name> <URL>`
- `git remote -v`
- `git remote rm <remote_repo_name>`
- `git remote rename <old_name> <new_name>`

Remote Repository (2)

- **git push [remote_repo_name] [branch_name]**
 - push your local data to the remote repo
 - something different when you first push, follows the tips provided by Git
- **git pull**
 - something like “**git fetch && git merge**”
- **git fetch [remote_repo_name]**
 - get all data from the remote repo
 - will NOT do any merge for you (unlike git pull)

Tag (1)

- when you have an important version of data (like a release version), you would like to give a tag for that version, so that you can find it easily with using the SHA number
- **git tag** (list all tags)
- **git tag -a v1.4 -m 'my version 1.4'** (an annotated tag example)
- **git tag <tag_name>** (a lightweight version tag)
- **git tag -a v1.2 9fceb02** (add a tag to a previous commit)

Tag (2)

- tag will NOT be pushed to remote repo
 - **git push <remote_repo_name> <tag_name>** (push a tag)
 - **git push <remote_repo_name> --tag** (push all tags)
- you can't move the HEAD between tags, if you want to switch back to a tag, you can create a branch to hold the data in that tag
 - **git checkout -b [branch_name] [tag_name]**
 - be careful if you want to change the content of that tag

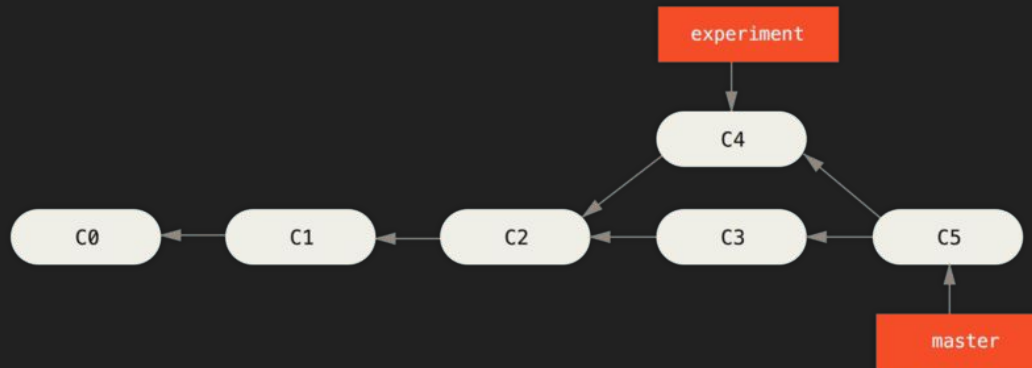
Branch --- that's what make Git powerful

- list all branches:
 - **git branch**
- create a new branch:
 - **git branch <branch_name>**
- create and switch that branch
 - **git checkout -b <branch_name>**
- switch between two existed branches:
 - **git checkout <branch_name>**
- merge a target branch into current branch:
 - **git merge <target_branch_name>**

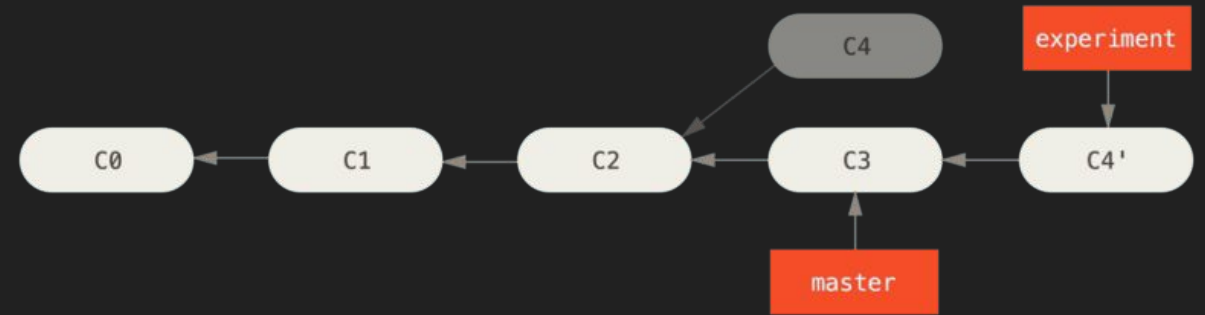
Branch Management

- list all merged branch (usually they can be deleted)
 - **git branch --merged** (same story with --no-merged)
 - **git branch -d <branch_name>** (only work for merged branch)
- delete anyway (force delete)
 - **git branch -D <branch_name>**

Branch: merge, rebase



merge



rebase

images from a book: *Git Pro*

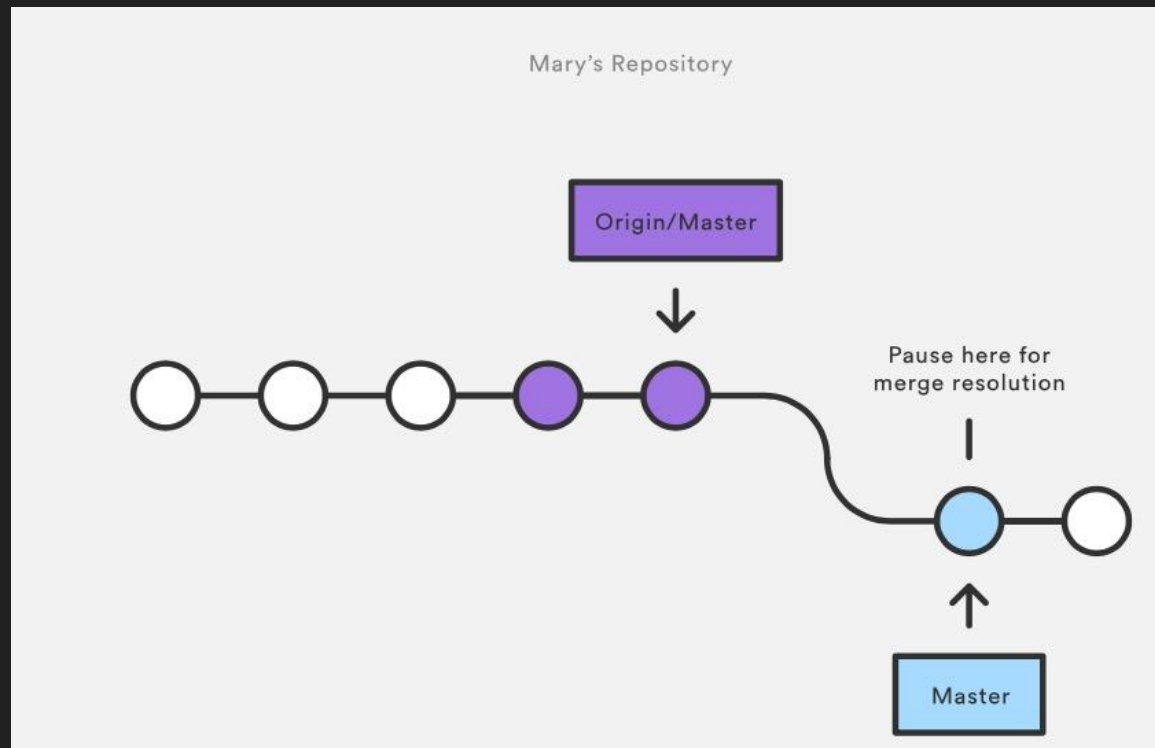
Branch Warning

- do NOT rebase any thing already in the remote repo
- only can apply rebase command to your local data

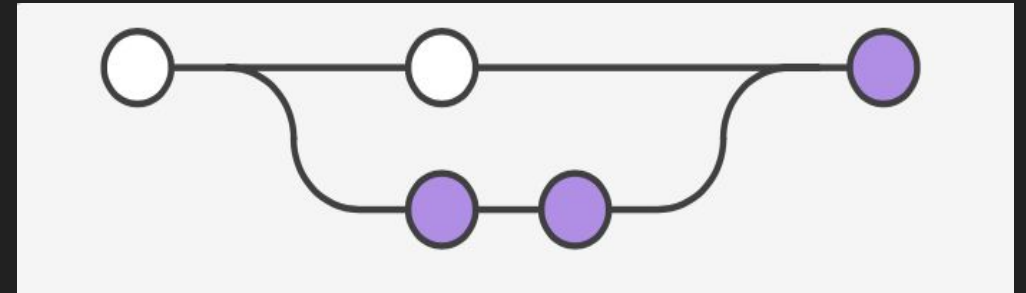
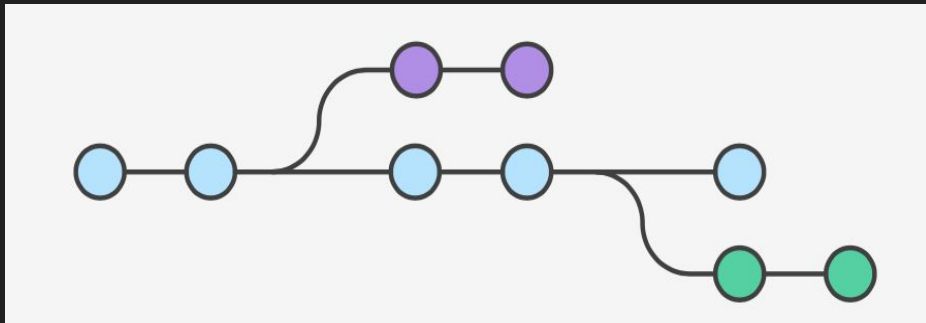
WorkFlow

Centralized Workflow, Feature Branch Workflow, Gitflow Workflow, Forking Workflow

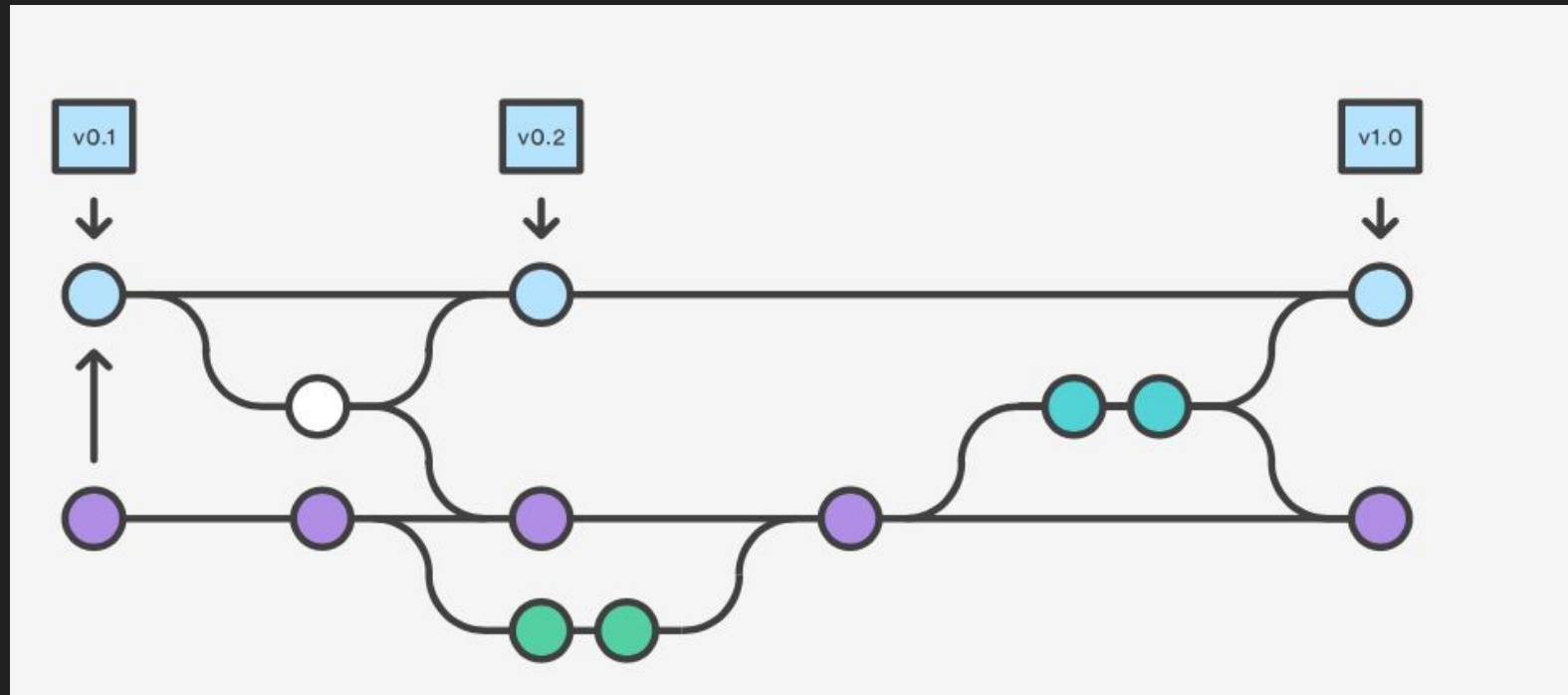
Centralized Workflow



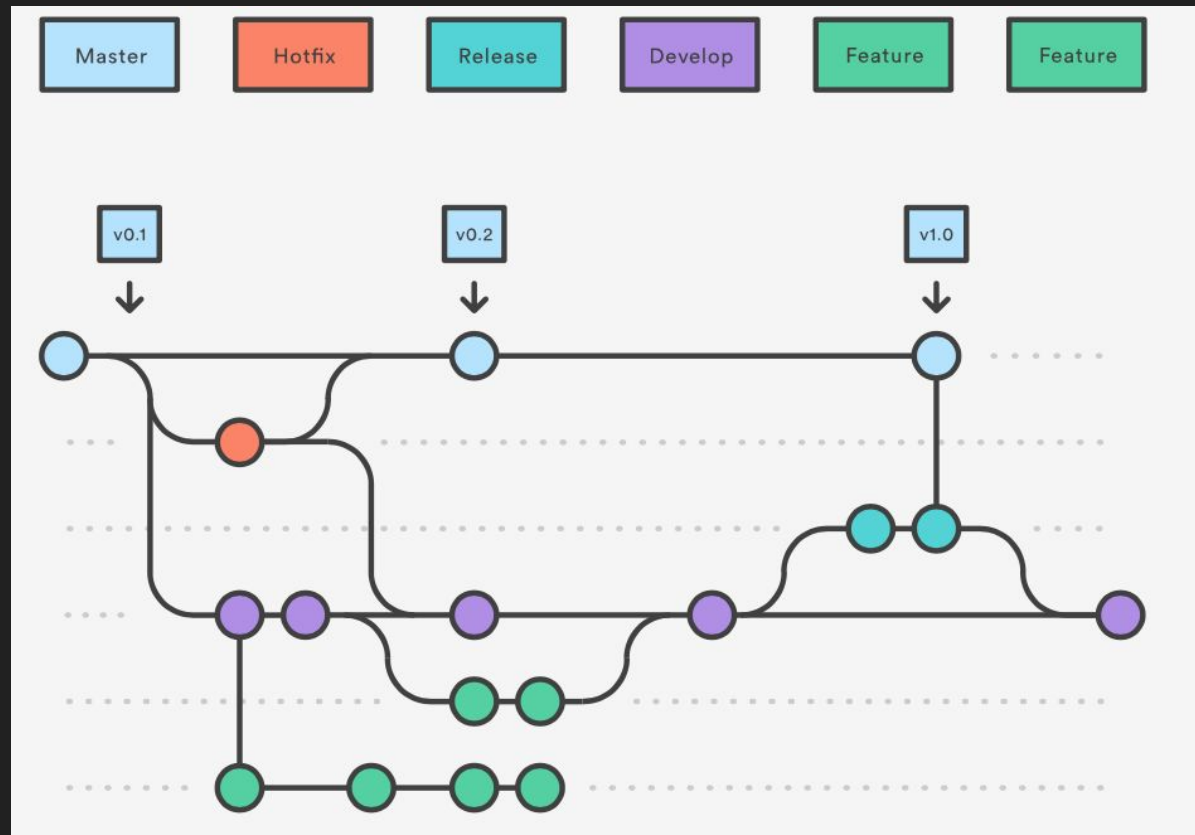
Feature Branch Workflow



Gitflow Workflow



Forking Workflow



Resources

A well maintained list of resources related to Git on GitHub, check it through the following link:

<https://github.com/dictcp/awesome-git>