Github

Git Cheat Sheet

REBELLABS by ZEROTURNAROUND

Create a Repository

From scratch -- Create a new local repository
```
$ git init [project name]
```

Download from an existing repository
```
$ git clone my_url
```

Observe your Repository

List new or modified files not yet committed
```
$ git status
```

Show the changes to files not yet staged
```
$ git diff
```

Show the changes to staged files
```
$ git diff --cached
```

Show all staged and unstaged file changes
```
$ git diff HEAD
```

Show the changes between two commit ids
```
$ git diff commit1 commit2
```

List the change dates and authors for a file
```
$ git blame [file]
```

Show the file changes for a commit id and/or file
```
$ git show [commit]:[file]
```

Show full change history
```
$ git log
```

Show change history for file/directory including diffs
```
$ git log -p [file/directory]
```

Working with Branches

List all local branches
```
$ git branch
```

List all branches, local and remote
```
$ git branch -av
```

Switch to a branch, my_branch, and update working directory
```
$ git checkout my_branch
```

Create a new branch called new_branch
```
$ git branch new_branch
```

Delete the branch called my_branch
```
$ git branch -d my_branch
```

Merge branch_a into branch_b
```
$ git checkout branch_b
$ git merge branch_a
```

Tag the current commit
```
$ git tag my_tag
```

Make a change

Stages the file, ready for commit
```
$ git add [file]
```

Stage all changed files, ready for commit
```
$ git add .
```

Commit all staged files to versioned history
```
$ git commit -m "commit message"
```

Commit all your tracked files to versioned history
```
$git commit -am "commit message"
```

Unstages file, keeping the file changes
```
$ git reset [file]
```

Revert everything to the last commit
```
$ git reset --hard
```

Synchronize

Get the latest changes from origin (no merge)
```
$ git fetch
```

Fetch the latest changes from origin and merge
```
$ git pull
```

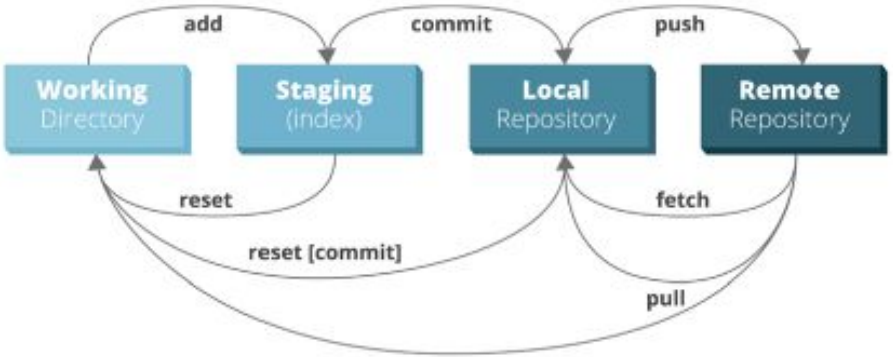
Fetch the latest changes from origin and rebase
```
$ git pull --rebase
```

Push local changes to the origin
```
$ git push
```

Finally!

When in doubt, use git help
```
$ git command --help
```

Or visit https://training.github.com/ for official GitHub training.



What is Git ?

The word ”git” refers to the version control system and is a tool that allows developers to keep track of constant revisions in their code.

What is Github?

The "Hub" is a community of like-minded people, whose collective efforts revise, improve and derive ideas on the code loaded.

git clone

```
$ git init myProject
$ git clone
https://github.com/zeroturnaround/ziprebe
l.git
$git add .
$ git commit -m "Added unzip capability"
$git push -u origin main
```

What is Branch?

The use of branches in Git is not optional, even if you are not aware of it, you will always work on a single active branch while working on your project.

When you first create your project in Git, Git creates a branch for you called master by default
and you start working on that branch.

```
$git checkout branchname
```

```
$git branch
```

```
$git status
```

What is Pull Request ?

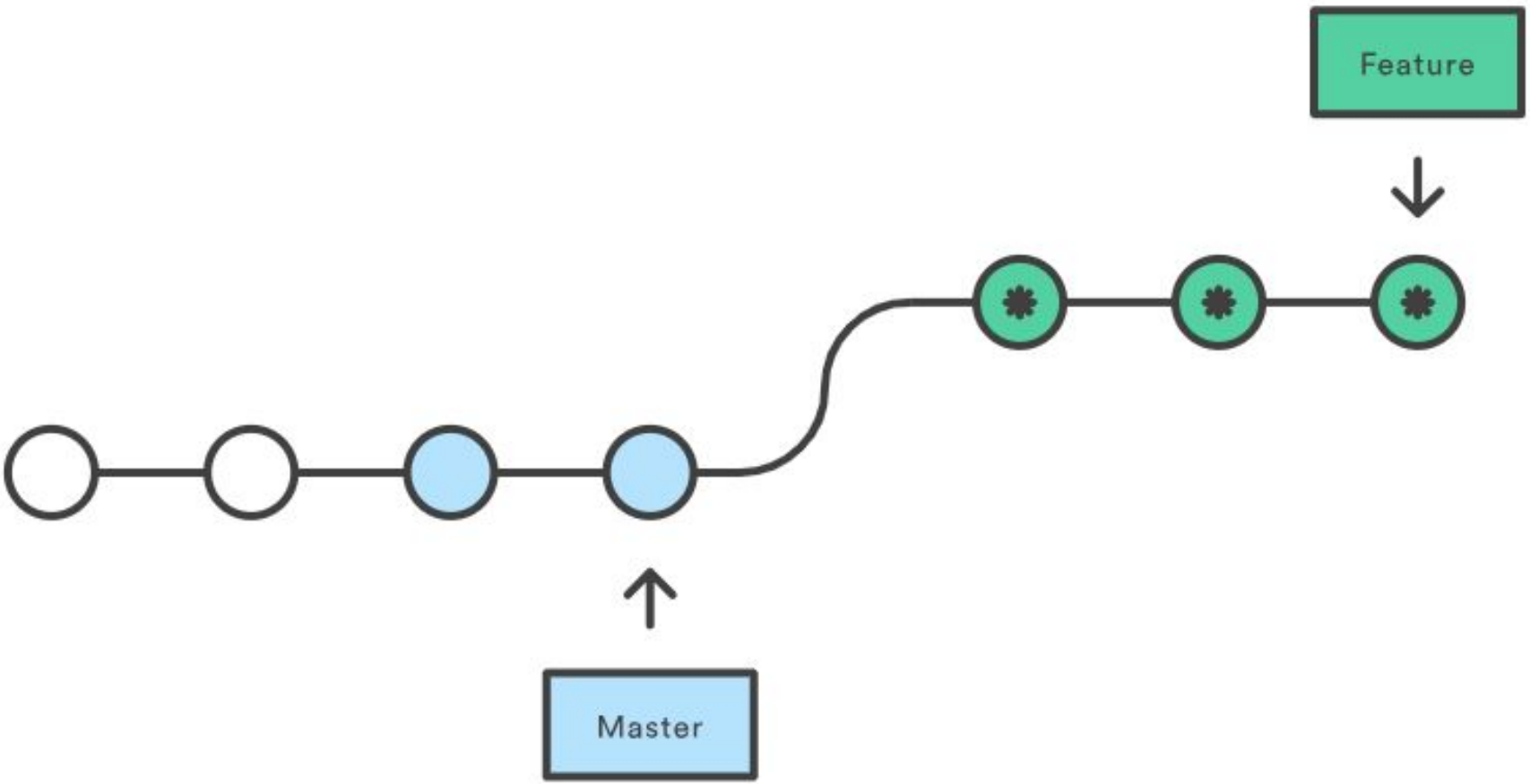
In their simplest form,
pull requests are a mechanism for a developer to notify team members that they have completed a feature.

How to create a pull request in GitHub

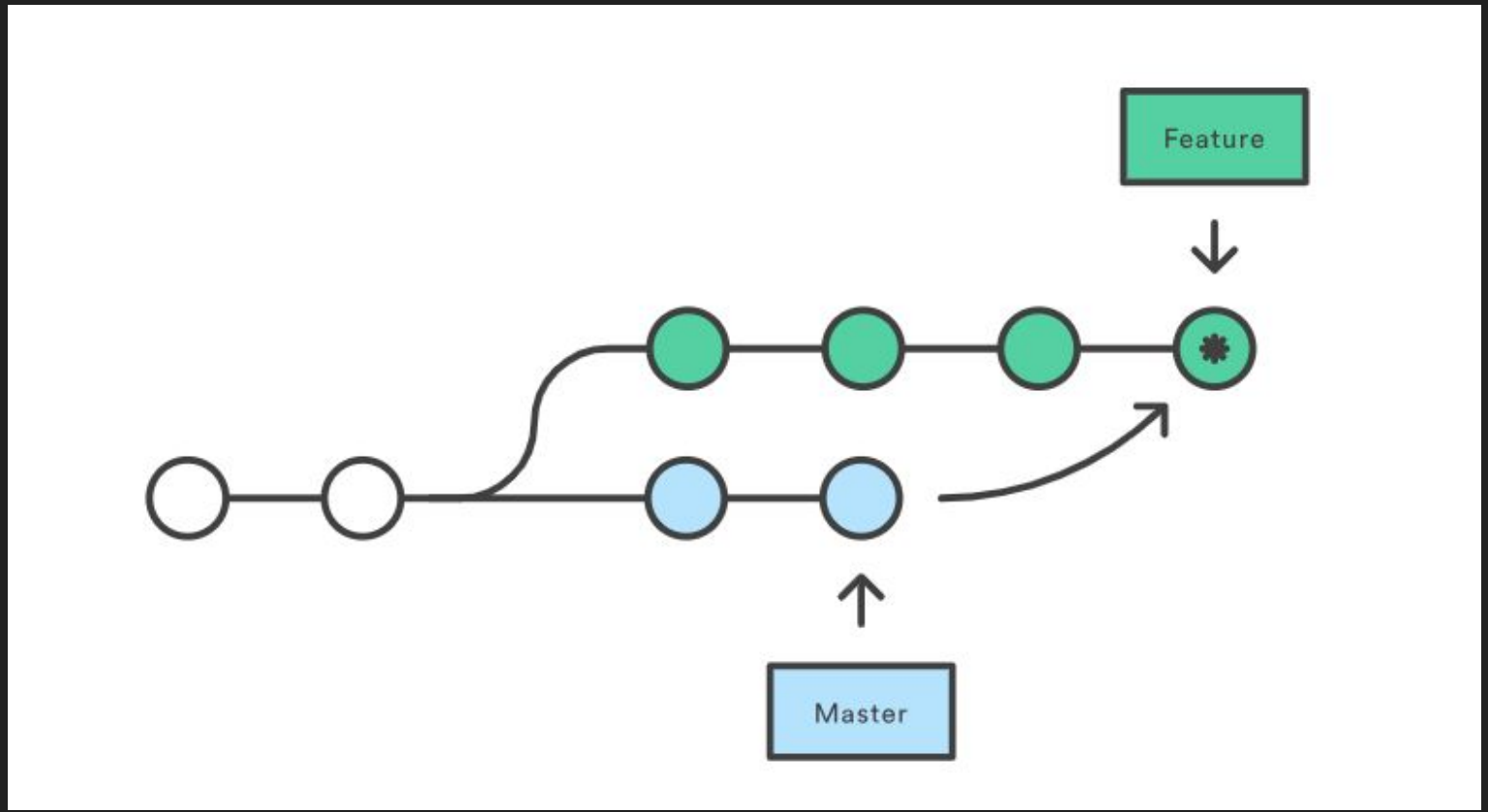
1. Find a project you want to contribute to
2. Fork it
3. Clone it to your local system
4. Make a new branch
5. Make your changes
6. Push it back to your repo
7. Click the Compare pull request button
8. Click Create pull request to open a new pull request

What is Rebase?

Rebase is another way to integrate changes from one branch to another. Rebase compresses all the changes into a single “patch.” Then it integrates the patch onto the target branch


Feature
Master

Rebase feature branch into master



What is Merge?

Merging takes the contents of a source branch and integrates them with a target branch. In this pro cess, only the target branch is changed. The source branch history remains the same.

git rebase

```
$git checkout experiment
$git rebase master
```

git merge

```
$git merge master experiment
```

Undo Last Git Commit with reset

```
$ git log --oneline
```

```
$ git reset --soft HEAD~1
```

git cheat sheet

Initialization

$ git init <directory>
Creates new repo in specified directory

$ git clone <url>
Copies repo from specied url

$ git config user.name <user_name>
Sets username for commits in current repo
Use --global to apply it globally

$ git config user.email <user_email>
Sets email for commit in current repo
Use --global to apply it globally

$ git config color.ui auto
Enables helpful colorisation of command line
output

$ git config --global --edit
Opens the global configuration file in text editor for manual editing

$ git remote add origin <link>
Connects your local repo to the remote one

$ vi .gitignore
Opens .gitignore file. This file is used for list of files that have to be excluded. Ensure that this file is in root of local repo. You can change **vi** into your favourite text editor

Commits

$ git add <path>
Adds path into staging. Path can be file or directory

$ git restore --staged <path>
Removes path from staging back to unstaged area

$ git rm -r <path>
Removes path and adds that change into staging

$ git commit -m <message>
Commits the stage with specified message

$ git commit --amend -m <message>
Repairs last commit with specified new message

$ git commit --amend --no-edit
Repairs last commit without editing commit message

$ git status
Lists which files are staged, unstaged, or untracked

$ git push
Uploads all commits to remote branch

$ git pull -r
Updates local branch with all new commits from remote branch with rebasing, avoiding the conflict with changes from remote

Change Review

$ git log
Lists version history for the current branch

$ git diff <commit1> <commit2>
Shows difference between two commits. It is also applied to comparing two branches

$ git diff <commit1> <commit2> --name-only
Same with above, but only show the file names only

$ git stash
Save current changes into stash stack
Usually used when current changes don't want to be commited

$ git stash pop
Applies last changes stored in stash stack onto current working HEAD

$ git stash list
Shows stash stack

$ git revert <commit>
Creates new commit that undoes all of the changes in <commit>

$ git reset <commit>
Undoes the commits after <commit>, keep the changes locally. Add --hard to discard the changes

Branch & Rebase

$ git checkout <branch>
Switches to the specified branch

$ git checkout -
Switches to the previous visited branch

$ git checkout -b <name>
Creates a new branch with specified name and switch in that branch

$ git checkout <path>
Restores changes of <path> back into latest revision

$ git branch
Lists all branchs

$ git branch -m <old> <new>
Renames branch from <old> to <new>

$ git branch -d <branch>
Deletes the specified branch

$ git rebase -i <base>
Interactively rebases the current branch onto base. It can be branch, commit, or relative reference to HEAD

$ git push -f
Uploads all commits to remote branch with force. Usually used when there are conflicts when rebasing. Do not try this unless you know what you are doing

Advanced

$ git checkout -R <old_branch> <new_branch>
$ git push origin :<old_branch> <new_branch>
Rename branch in local and remote correspondently

$ git tag <tag_name>
$ git push origin --tags
Create tag and push all created tags

$ git tag -d <tag_name>
$ git push origin --delete <tag_name>
Delete tag and push deleted tags

$ git remote set-url origin <url>
Changes remote url. Usually used after repository migration

$ git cherry-pick <commit>
Creates new commit by applying changes in <commit> into current working HEAD

$ git gc --prune=now --aggressive
Cleanups and optimizes all files in local repository

$ git bisect start
$ git bisect good
$ git bisect bad
Find the commit that contains bug with binary search

$ git blame -- <file>
Shows revision in <file> line by line