GitHub Cheat Sheet

Ignore Whitespace

Adding `?w=1` to any diff URL

```
-    below branch is the
+below branch is the
+
+## Whitespace
```

```
below branch is the
+
+## Whitespace
```

Adjust Tab Space

Adding `?ts=4` to a diff or file URL

```
if install_ingress:
    self.kubernetes
    self.linux
    self.windows
    self.solaris
    self.centos
```

```
if install_ingress:
    self.kubernetes
    self.linux
    self.windows
    self.solaris
    self.centos
```

Lorem ipsum dolor sit

Commit History by Author

Adding `?author={user}` to the URL

Cloning a Repository

When cloning a repository the `.git` can be left off the end.

```
$ git clone https://github.com/gluu/example
```

Branch

Compare all branches to another branch

```
https://github.com/{user}/{repo}/branches
```

If you go to the repo's branches page, next to the commits button:

Gists

Work with small codes without create fully fledged repo

Add `.pibib` to the end of Gist URL to get the HTML-only version

Gists can be treated as a repository so they can be cloned like any other:

```
$ git clone https://githuts.com/GuuuFederation/private/Issues/302
```

Keyboard Shortcuts

Allow you to navigate repository easily

T Bring up a file explorer

W Bring up the branch selector

S Focus the search field for the current repository. Pressing ↓ to select the "All GitHub" option changes the field to search all of GitHub.

L Edit labels on existing issues.

Y When looking at a file will change your URL to one which, in effect, freezes the page you are looking at. If code changes, you will still be able to see what you saw at that current time.

Line Highlighting in Repositories

adding #L(line number) or #L(line number)-(line number) to the end of URL will highlight that line number/s

Example: github/workflows/label_pr.yml#L9-L12

Closing Issues via Commit Messages

Keywords fix/fixes/fixed, close/closes/closed, resolve/resolves/resolved, followed by the issue number, will close the issue once it is committed to the repository's default branch.

Cross-Link Issues

to link to another issue in the same repository, type:
#issue number
and it will be auto-linked, to link to an issue in another repository type:
{user}/{repo}#ISSUE_NUMBER

Compare Branches across Forked Repositories

```
https://github.com/{user}/{repo}/compare/{foreign-user}:{branch}...{own-branch}
```

Compare Range

```
https://github.com/{user}/{repo}/compare/{range}
```
Dates are in the format YYYY-MM-DD

EXAMPLE
```
https://github.com/{user}/{repo}/compare/master@{1.day.ago}...master
https://github.com/{user}/{repo}/compare/master@{2014-10-04}...master
```

Locking Conversations

Pull Requests and Issues can now be locked by owners or collaborators of the repo.

users who are not collaborators on the project will no longer be able to comment.

This conversation has been locked and limited to collaborators.

CI Status on Pull Requests

Travis CI build received Pull Request like it would every time you make a new commit

Filters

issues and pull requests allow filtering in the user interface.

For the Rails repo: github.com/rails/rails/issues, the following filter is built by selecting the label "activerecord":

```
is:issue label:activerecord
```

But, you can also find all issues that are NOT labeled activerecord:

```
is:issue -label:activerecord
```

Additionally, this also works for pull requests:

```
is:pr -label:activerecord
```

Github has tabs for displaying open or closed issues and pull requests but you can also see merged pull requests. Just put the following in the filter:

```
is:merged
```

Syntax Highlighting in Markdown Files

```ruby
require 'blockquote'
table = Blockquote.new('Name', 'Email')
table.add_row('ma don', 'mark@gmail.com')
puts table.to_s
```

```ruby
require 'blockquote'
table = Blockquote.new('Name', 'Email')
table.add_row('ma don', 'mark@gmail.com')
puts table.to_s
```

Emojis

Emojis can be added to Pull Requests, Issues, commit messages, repository descriptions, etc. using :name_of_emoji:.

The top 5 used Emojis on GitHub are:

1. :shipit:
2. :sparkles:
3. :+1:
4. :+1:
5. :clap:

Images/GIFs

Images and GIFs can be added to comments

`![Alt Text](http://www.URL.com/image.gif)`

Raw images from the repo can be used by calling them directly

`![Alt Text](https://github.com/[user]/[repo]/path/image.gif)`

All images are cached on GitHub, so if your host goes down the image will remain available.

Embedding Images in GitHub Wiki

This syntax allow you to specify the height or width of the image

`http://www.URL.com/image.gif | height = 100px`

Pasting Clipboard Image to Comments

(Tested on Chrome and Firefox browsers)

After taking a screenshot and adding it to the clipboard (mac: cmd-ctrl-shift-4), you can simply paste (cmd-v / ctrl-v) the image into the comment section and it will be auto-uploaded to github.

Quick Licensing

When creating a repository, GitHub gives you the option of adding in a pre-made license

Licenses can be added to an existing repositories by creating a new file, type LICENSE to get an option to use a template:
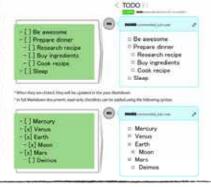
Rendering PDF

Rendering PDF

Simply browse to a PDF document and it will be rendered in the browser like any other file.

Task Lists

In Issues and Pull requests check boxes can be added with the following syntax (notice the space):

* When they are clicked, they will be updated in the pure Markdown.
* In full Markdown documents, read-only checklists can be added using the following syntax.

Relative Links

Relative links are recommended in your Markdown files when linking to internal content.

```
[Link to a header](#awesome-section)
[Link to a file](docs/readme)
```

* Absolute links have to be updated whenever the URL changes.
* Using relative links makes your documentation neatly stand on its own.

Metadata and Plugin Support for GitHub Pages

Within Jekyll pages and posts, repository information is available within the site.github namespace, and can be displayed, for example, using [[site.github.project_title]].

The Jemoji and jekyll-mentions plugins enable emoji and @mentions in your Jekyll posts and pages to work just like you'd expect when interacting with a repository on GitHub.com.

Viewing YAML Metadata in your Documents

Blogging websites, like Jekyll with GitHub Pages, depend on YAML-formatted metadata at the beginning of the post. GitHub will render this metadata as a horizontal table for easier reading.

Rendering Tabular Data

GitHub supports rendering tabular data in the form of `.csv` (comma-separated) and `.tsv` (tab-separated) files.

Title	Release Year	Locations	Fun Facts	Production Company
Extreme Flexibility	2008	London		Glou Cloud
Scales in Billions	2008	Cairo		Glou Cloud
Cloud Native	2003	Paris		Glou Cloud
Open Source	2010	Kyoto		Glou Cloud
Digital Identity	2016	Sao Paulo		Glou Cloud
OpenID Connect	2002	Madenah		Open ID
FIDO	2007	Kansas		FIDO Ini
User Access	2004	New York		Glou Cloud

Revert a Pull Request

After a pull request is merged, you can revert the process by clicking the Revert button to create a pull request with reverted changes to this specific pull request.

Diffs

Rendered Prose Diffs

Commits and pull requests, including rendered documents supported by GitHub, feature source and rendered views.

Click the "rendered" button to see the changes as they'll appear in the rendered document. Rendered prose view is handy when you're adding, removing, and editing text.

GitHub Cheat Sheet

Diffable Maps

Any time you view a commit or pull request on GitHub that includes geodata, GitHub will render a visual representation of what was changed.

Expanding Context in Diffs

Using the unfold button in the gutter of a diff, reveals additional lines of context with a click. clicking unfolds will reveal the whole file, this feature is available anywhere GitHub renders diffs.

Diff or Patch of Pull Request

Get the diff of a Pull Request by adding a `.diff` or `.patch` extension to the end of the URL. For example:

```
https://github.com/cheat-sheet/pull/15
https://github.com/cheat-sheet/pull/15.diff
https://github.com/cheat-sheet/pull/15.patch
```

The `.diff` extension would give you this in plain text:

Rendering and diffing images

GitHub can display several common image formats, including PNG, JPG, GIF, and PSD. In addition, there are several ways to compare differences between versions of those image formats.

Hub

Hub is a command line Git wrapper that gives extra features and commands that make working with GitHub easier.
This allows you to do things like:

```
$ hub clone GluuFederation/oxd
```

Contribution Guidelines

GitHub supports adding three different files which help users contribute to your project. These files can either be placed in the root of your repository or a `.github` directory under the root.

CONTRIBUTING File

Adding a `CONTRIBUTING` or `CONTRIBUTING.md` file to either the root of your repository or a `.github` directory will add a link to your file when a contributor creates an Issue or opens a Pull Request.

SSH keys

You can get a list of public ssh keys in plain text format by visiting:

```
https://github.com/{user}.keys
```

e.g. https://github.com/tiimgreen.keys

Profile Image

You can get a user's profile image by visiting:

```
https://github.com/{user}.png
```

e.g. https://github.com/moabu.png

ISSUE_TEMPLATE file

A template can be defined for all new issues opened in a project. The content of this file will pre-populate the new issue box when users create new issues.
Add an `ISSUE_TEMPLATE` or `ISSUE_TEMPLATE.md` file to either the root of repository or a `.github` directory.

PULL_REQUEST_TEMPLATE file

You can define a template for all new pull requests opened in your project. The content of this file will pre-populate the text area when users create pull requests. Add a PULL_REQUEST_TEMPLATE or PULL_REQUEST_TEMPLATE.md file to either the root of your repository or a `.github` directory.

Octicons

GitHubs icons (Octicons) have now been open sourced.

GitHub Student Developer Pack

If you are a student you will be eligible for the GitHub Student Developer Pack. This gives you free credit, free trials and early access to software that will help you when developing.

GitHub Resources

Title	Link
GitHub Explore	https://github.com/explore
GitHub Blog	https://github.com/blog
GitHub Help	https://help.github.com/
GitHub Training	https://training.github.com/
GitHub Developer	https://developer.github.com/
Github Education	https://education.github.com/
GitHub Best Practices	https://www.datree.io/resources/github-best-practices

Repository Templates

You can enable templating on your repository which allows anyone to copy the directory structure and files, allowing them to instantly use the files (e.g. for a tutorial or if writing boilerplate code).
This can be enabled in the settings of your repository.

Changing to a template repository will give a new URL endpoint which can be shared and instantly allows users to use your repository as a template. Alternatively, they can go to your repository and click the 'Use as template' button.

Git

Remove All Deleted Files from the Working Tree

When you delete a lot of files using /bin/rm you can use the following command to remove them from the working tree and from the index, eliminating the need to remove each one individually:

```
$ git rm $(git ls-files -d)
```

For example:

```
$ git status
On branch master
Changes not staged for commit:
        deleted:    a
        deleted:    c

$ git rm $(git ls-files -d)
rm 'a'
rm 'c'

$ git status
On branch master
Changes to be committed:
        deleted:    a
        deleted:    c
```

Previous Branch

To move to the previous branch in Git:

```
$ git checkout -
# Switched to branch 'master'

$ git checkout -
# Switched to branch 'next'

$ git checkout -
# Switched to branch 'master2'
```

Stripspace

Git Stripspace:

- Strips trailing whitespace
- Collapses newlines
- Adds newline to end of file

A file must be passed when calling the command, e.g.:

```
$ git stripspace < README.md
```

Fixup and Autosquash

If there is something wrong with a previous commit (can be one or more from HEAD), for example abcde, run the following command after you've amended the problem:

```
$ git commit --fixup=abcde
$ git rebase abcde~ --autosquash -i
```

Checking out Pull Requests

Pull Requests are special branches on the GitHub repository which can be retrieved locally in several ways:
Retrieve a specific Pull Request and store it temporarily in FETCH_HEAD for quickly diff-ing or merge-ing:

```
$ git fetch origin refs/pull/[PR-Number]/head
```

Acquire all Pull Request branches as local remote branches by refspec:

```
$ git fetch origin '+refs/pull/*/head:refs/remotes/origin/pr/*'
```

Or setup the remote to fetch Pull Requests automatically by adding these corresponding lines in your repository's .git/config:

```
[remote "origin"]
    fetch = +refs/heads/*:refs/remotes/origin/*
    url = git@github.com:GluuFederation/oxAuth.git
```

```
[remote "origin"]
    fetch = +refs/heads/*:refs/remotes/origin/*
    url = git@github.com:GluuFederation/oxAuth.git
    fetch = +refs/pull/*/head:refs/remotes/origin/pr/*
```

For Fork-based Pull Request contributions, it's useful to checkout a remote branch representing the Pull Request and create a local branch from it:

```
$ git checkout pr/42 pr-42
```

Or should you work on more repositories, you can globally configure fetching pull requests in the global git config instead.

```
git config --global --add remote.origin.fetch "+refs/pull/*/head:refs/remotes/origin/pr/*"
```

This way, you can use the following short commands in all your repositories:

```
git fetch origin
```
```
git checkout pr/42
```

Merged Branches

Running:
```
$ git branch --merged
```

will give you a list of all branches that have been merged into your current branch.

Conversely:
```
$ git branch --no-merged
```

will give you a list of branches that have not been merged into your current branch.

Empty Commits

Commits can be pushed with no code changes by adding --allow-empty:

```
$ git commit -m "Great-morals commit" --allow-empty
```

Some use-cases for this include:
- Annotating the start of a new bulk of work or a new feature.
- Documenting when you make changes to the project that aren't code related.
- Communicating with people using your repository.
- The first commit of a repository: git commit -m "Initial commit" --allow-empty.

Styled Git Status

Running:

```
$ git status
```


By adding -sb:

```
$ git status -sb
```


Styled Git Log

Running:

```
$ git log --all --graph --pretty=format:"%Cred%h%Creset -%C(auto)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset" --abbrev-commit --date=relative
```


Git Query

A Git query allows you to search all your previous commit messages and find the most recent one matching the query.

```
$ git show :/query
```

where query (case-sensitive) is the term you want to search, this then finds the last one and gives details on the lines that were changed.

```
$ git show :/typo
```


Press q to quit.

Git Grep

Git Grep will return a list of lines matching a pattern.
Running:

```
$ git grep aliases
```

will show all the files containing the string aliases.

Press q to quit.

You can also use multiple flags for more advanced search. For example:

- -e The next parameter is the pattern (e.g., regex)
- --and, --or and --not Combine multiple patterns.

Use it like this:

```
$ git grep -e pattern --and -e anotherpattern
```

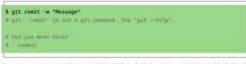## Web Server for Browsing Local Repositories

Use the Git `instaweb` command to instantly browse your working repository in `gitweb`. This command is a simple script to set up `gitweb` and a web server for browsing the local repository.

```
$ git instaweb
```

Lorem ipsum

Auto-Correct

Git gives suggestions for misspelled commands and if auto-correct is enabled the command can be fixed and executed automatically.
Auto-correct is enabled by specifying an integer which is the delay in tenths of a second before git will run the corrected command. Zero is the default value where no correcting will take place, and a negative value will run the corrected command with no delay.

For example, if you type `git comit` you will get this:

```
$ git comit -m "Message"
# git: 'comit' is not a git command. See 'git --help'.

# Did you mean this?
#    commit
```

Auto-correct can be enabled like this (with a 1.5 second delay):

```
$ git config --global help.autocorrect 15
```

So now the command `git comit` will be auto-corrected to `git commit` like

```
$ git comit -m "Message"
# WARNING: You called a Git command named 'comit', which does not exist.
# Continuing under the assumption that you meant 'commit'
# in 1.5 seconds automatically...
```

The delay before git will rerun the command is so the user has time to abort.

Aliases

Aliases are helpers that let you define your own git calls. For example you could set `git a` to run `git add --all`.

To add an alias, either navigate to `~/.gitconfig` and fill it out in the following format:

```
[alias]
  co = checkout
  ci = commit
  p = push
  # Show verbose output about tags, branches or remotes
  tags = tag -l
  branches = branch -a
  remotes = remote -v
```

...or type in the command-line:

```
$ git config --global alias.new_alias git_function
```

For example:

```
$ git config --global alias.ci commit
```

For an alias with multiple functions use quotes:

```
$ git config --global alias.ci commit
```

Some useful aliases include:

Alias	Command	Type
git ci	git commit	git config --global alias.ci commit
git co	git checkout	git config --global alias.co checkout
git ac	git add -A git commit	git config --global alias.ac '!git add -A && git commit'
git st	git status -sb	git config --global alias.st 'status -sb'
git tags	git tag -l	git config --global alias.tags 'tag -l'
git branches	git branch -a	git config --global alias.branches 'branch -a'
git cleanup	git branch --merged | grep -v '*' | xargs git branch -d	git config --global alias.cleanup "!git branch --merged | grep -v '*' | xargs git branch -d"
git remotes	git remote -v	git config --global alias.remotes 'remote -v'
git lg	git remote -v git remote -v git remote -v git remote -v git remote -v	git config --global alias.lg "log --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"

Git Configurations

Your `.gitconfig` file contains all your Git configurations.

Color

To add more color to your Git output:

```
$ git config --global color.ui 1
```

Git Resources

Title	Link
Official Git Site	http://git-scm.com
Official Git Video Tutorials	http://git-scm.com/videos
Code School Try Git	http://try.github.com/
Introductory Reference & Tutorial for Git	http://gitref.org/
Official Git Tutorial	http://git-scm.com/docs/gittutorial
Everyday Git	http://git-scm.com/docs/everyday
Git Immersion	http://gitimmersion.com/
Git Fast	https://github.com/gorangajic/git-fast
Git for Computer Scientists	http://eagain.net/articles/git-for-computer-scientists/
Git Magic	http://www-cs-students.stanford.edu/~blynn/gitmagic/
Git Visualization Playground	http://onlywei.github.io/explain-git-with-d3/#freeplay
Learn Git Branching	http://pcottle.github.io/learnGitBranching/
A collection of useful .gitignore templates	https://github.com/github/gitignore
Unixorn's git-extra-commands collection of git scripts	https://github.com/unixorn/git-extra-commands
GitHub repo cheat sheet	https://github.com/tiimgreen/github-cheat-sheet

Git Books

Title	Link
Pragmatic Version Control Using Git	https://pragprog.com/titles/tsgit/pragmatic-version-control-using-git
Pro Git	http://git-scm.com/book
Git Internals PluralSight	https://github.com/pluralsight/git-internals-pdf
Git in the Trenches	http://cbx33.github.io/gitt/
Version Control with Git	http://www.amazon.com/Version-Control-Git-collaborative-development/dp/1449316387
Pragmatic Guide to Git	https://pragprog.com/titles/pg_git/pragmatic-guide-to-git
Git: Version Control for Everyone	https://www.packtpub.com/application-development/git-version-control-everyone

Git Videos

Title	Link
Linus Torvalds on Git	https://www.youtube.com/watch?v=4XpnKHJAok8
Introduction to Git with Scott Chacon	https://www.youtube.com/watch?v=ZDR433b0HJY
Git From the Bits Up	https://www.youtube.com/watch?v=MYP56QJpDr4
Graphs, Hashes, and Compression, Oh My!	https://www.youtube.com/watch?v=ig5E8CcdM9g
GitHub Training & Guides	https://www.youtube.com/watch?v=PLg7s8cbHAD1SC8Iteya0YDuXZ5Xy,kgufJ-6v-FyfaLX4HAeM

Git Articles

Title	Link
GitHub Flow	http://scottchacon.com/2011/08/31/github-flow.html
Migrating to Git Large File Storage (Git LFS)	http://yobari.com/en/hpc-articles-geek-stuff/migrating-to-git-lfs-for-developing-deep-learning-applications-with-large-files/

THE END