

# Git

git

??2005??GPL????????????????????Linux????????????????????GNU  
Interactive Tools??? git????????????????BitKeeper?Monotone?

- [Learning Git](#)
- [Git Installation](#)
- [Git Tips](#)
- [FAQ](#)
- [Git ????](#)
- [GitHub](#)
- [Branch](#)
- [Remote repository](#)
- [Merge](#)
- [????](#)
- [Pull request](#)
- [Code review](#)
- [Cheat Sheets](#)

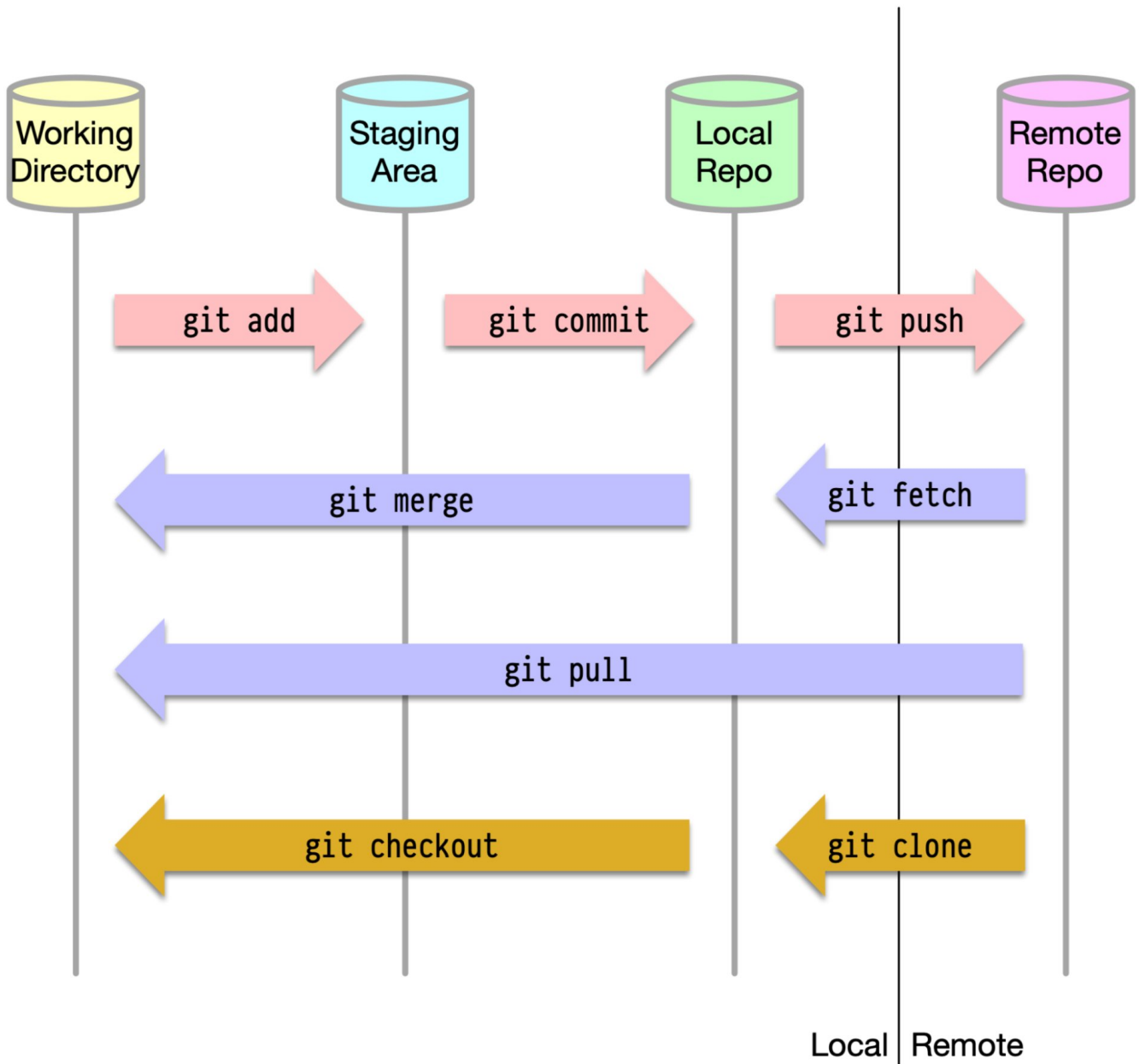
# Learning Git

## Git ????

?? 2 ? 3 ?????? Git ???????

1. Modified/Untracked: ?????????? `git add`
2. Staged: ??? `git add` ????? `git commit`
3. Committed: ??? `git commit`

## Git Commands Work



??

- [Git - Book](#)
- [Git ?? - Git???? & Github??](#)
- [Git ?????? - ?????? | My.APOLLO \(myapollo.com.tw\)](#)
- [?? Git ?? \(Branching\)](#)
- [Learn Git Branching \(??\)](#)

??

- [Getting started with GIT on Linux](#)
- [git - the simple guide - no deep shit! \(up1.github.io\)](#)
- [10 things to love about Git](#)
- [Getting started with Git - GitHub Docs](#)
- [Oh Shit, Git!?!](#)
- [How to undo \(almost\) anything with Git](#)
- [10 Git tutorials to level up your open source skills in 2022](#)
- [Make your own Git subcommands](#)
- [My guide to using the Git push command safely](#)
- [7 Lesser-Known Git Commands and Tricks](#)
- [How to create a pull request in GitHub](#)
- [My favorite Git tools | Opensource.com](#)
- [50+ Useful Git Commands for Everyone](#)
- [Video] [Git Tutorial for Absolute Beginners](#)

## Bitbucket

- [Learn Git with Bitbucket Cloud | Atlassian Git Tutorial](#)

## Git Tools

- [LazyGit](#) - simple terminal UI for git commands
  - [Video] [LazyGit: A Powerful Way to Use Git](#)

## Git Server

- [Gitea](#)
  - YT: [How to install Gitea, a self hosted git server. - YouTube](#)
  - YT: [Private Self-Hosted GitHub Server | Gitea Complete Setup Guide - YouTube](#)
- [Gitlab](#)

## CI/CD

- [Introduction to GitHub Actions for Python Projects - PyImageSearch](#)
- [Drone CI](#) is a self-service Continuous Integration platform for busy development teams.
  - iT+: [????????? Container ?????? CI/CD ???](#)
- [Jenkins](#) - A common open source CI system

- [\[Day 15\] Jenkins ?? - iT ????:??????????? IT ????](#)

# Git Installation

## Git Client

```
# CentOS/RedHat 5/6
# Install from source
# Get the required version of GIT from https://www.kernel.org/pub/software/scm/git/
yum install zlib-devel openssl-devel cpio expat-devel gettext-devel
wget https://mirrors.edge.kernel.org/pub/software/scm/git/git-2.0.5.tar.gz
tar xzf git-2.0.5.tar.gz
cd git-2.0.5
./configure --prefix=/opt/git-2.0.5
make
make install
```

# Git Tips

## Tutorials

- [50+ Useful Git Commands for Everyone](#)

?????

```
# Using git to edit the configuration
git config global --edit

# List the global configurations
git config --global --list

# Using vi/cat to edit the configuration
vi ~/.gitconfig

# Set the author's email address and name
git config --global user.email "alang.hsu@gmail.com"
git config --global user.name "Alang Hsu"

# Set default editor
git config --global core.editor "vi"

# Set default branch name
git config --global init.defaultBranch "main"
```

.gitconfig

```
[user]
email = alang.hsu@gmail.com
name = Alang Hsu
[core]
editor = vi
[init]
defaultBranch = main
```

?????

```
mkdir test-git-push
cd test-git-push
git config --global user.name "<user-name>"
git config --global user.email "<your-email-addr>"
git init
echo "Test Git Push only" > README.md
git add .
git commit -m "Initial commit"
git remote add origin https://<user-name>@github.com/a-lang/test-git-push.git
git remote -v
git push -u origin master
```

## Git commit

??? commit ??

```
git commit -m "Fixed a typo in somewhere"
```

? add ?? commit (-a)

- ?? edited & deleted
- ?????? commit ??????????????
- ????????????

```
git commit -a -m "<commit-message>"
```

????????

```
# [] Vim []
git config --global core.editor "vim"

# [] -m []
git commit
```

Commit ??????

1. ??????????????
2. ?????? 50 ??????????????
3. ????????
4. ??????????

5. ????????
6. ??????? 72 ??????????????????
7. ????? what ?? why vs. how

??(??)???? Commit `--amend`

```
git commit --amend

# □ add □□□ commit
git commit -a --amend
```

`--amend` ?????commit ????? commit ????

???????????????????????????????????? commit ??????? commit ????????????

????? push ? commit ????? --amend????????????????????

## ??? HEAD

HEAD ?????? checked-out ????????????????????

## Git diff

```
# □□□ git add □□□□□□□□□□ commit □□□□□□
git diff file

# □□□□□□□□□□
git diff --word-diff file1 file2

# □□□ Commit □□□□□
git log --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold
blue)<%an>%Creset' --abbrev-commit --date=relative
git diff <old-commit-id> <new-commit-id>
```

## Undo in Git

????????????????

“ git restore -p : ??????????????????????????????

## Local unstaged changes (??? add)

```
git status  
git restore <filename>
```

????? staging ?????? checkout ???

checkout ????????? committed ? staged ???

```
git checkout <file-name>
```

## Local staged changes (?? add ??? commit)

```
git status  
git restore --staged <filename>  
git restore <filename>
```

## Local committed changes (?? commit ??? push)

```
git status  
git log  
git reset --soft HEAD~  
git log
```

“ **NOTE:** ?????????? commit????????? commit?????? git reset <commit-id> ??  
commit-id ??? git log --oneline ???

????? commit ?? push ?????????? git revert ?

## Public committed changes (?? push)

```
git log --oneline  
git revert <last-commit-id> --no-edit  
git push  
git log
```

NOTE: ??? `git revert` ??log ???? Revert "XXXX" ? commit ?????? commit  
???????

## Git Prompt with bash

.bashrc:

```
# Kali-like Prompt with Git status

git_stats() {
    local STATUS=$(git status -s 2> /dev/null)
    local UNTRACKED=$(echo "$STATUS" | grep '^??' | wc -l)
    local STAGED=$((($(echo "$STATUS" | grep '^M ' | wc -l) + $(echo "$STATUS" | grep '^D ' | wc -l) + $(echo "$STATUS" | grep '^R ' | wc -l) + $(echo "$STATUS" | grep '^C ' | wc -l) + $(echo "$STATUS" | grep '^A ' | wc -l)))
    local DRC=$((($(echo "$STATUS" | grep '^ D' | wc -l) + $(echo "$STATUS" | grep '^ R' | wc -l) + $(echo "$STATUS" | grep '^ C' | wc -l)))
    local MODIFIED=$(echo "$STATUS" | grep '^ M' | wc -l)
    local STATS=""
    if [ $UNTRACKED != 0 ]; then
        STATS="\e[43m untr: $UNTRACKED "
    fi
    if [ $MODIFIED != 0 ]; then
        STATS="$STATS\e[43m mod: $MODIFIED "
    fi
    if [ $DRC != 0 ]; then
        STATS="$STATS\e[43m drc: $DRC "
    fi
    if [ $STAGED != 0 ]; then
        STATS="$STATS \e[42m staged: $STAGED "
    fi
    if [ ! -z "$STATS" ]; then
        echo -e "\e[30m $STATS\e[0m"
    fi
}

function origin_dist {
    local STATUS=$(git status 2> /dev/null)
    local DIST_STRING=""
    local IS_AHEAD=$(echo -n "$STATUS" | grep "ahead")
}
```

```

local IS_BEHIND=$(echo -n "$STATUS" | grep "behind")
if [ ! -z "$IS_AHEAD" ]; then
    local DIST_VAL=$(echo "$IS_AHEAD" | sed 's/^[^0-9]*//g')
    DIST_STRING="$DIST_VAL AHEAD"
elif [ ! -z "$IS_BEHIND" ]; then
    local DIST_VAL=$(echo "$IS_BEHIND" | sed 's/^[^0-9]*//g')
    DIST_STRING="BEHIND $DIST_VAL"
fi
if [ ! -z "$DIST_STRING" ]; then
    echo -en "\e[97;45m $DIST_STRING"
fi
}

__PS1_GIT_BRANCH='`__git_ps1` '
__PS1_GIT_DIST='`origin_dist` '
__PS1_GIT_STATS='`git_stats` '

if $__git_ps1 2>/dev/null;then

PS1="\[\033[38;5;209m\] ──[\[\033[38;5;141m\]\u[\033[38;5;209m\]@[\033[38;5;105m\]\h[\033[38;5;231m\]:\w[\033[38;5;209m\]]\[\033[33m\]${__PS1_GIT_BRANCH}${__PS1_GIT_DIST}${__PS1_GIT_STATS}\[\033[00m\]\n\[\033[38;5;209m\] └─\[\033[38;5;209m\]\\$[\033[37m\] "
else
    source /usr/share/git-core/contrib/completion/git-prompt.sh

PS1="\[\033[38;5;209m\] ──[\[\033[38;5;141m\]\u[\033[38;5;209m\]@[\033[38;5;105m\]\h[\033[38;5;231m\]:\w[\033[38;5;209m\]]\[\033[33m\]${__PS1_GIT_BRANCH}${__PS1_GIT_DIST}${__PS1_GIT_STATS}\[\033[00m\]\n\[\033[38;5;209m\] └─\[\033[38;5;209m\]\\$[\033[37m\] "
fi

```

## Rename & Delete files

```

# Deleting
git rm my.sh

# Renaming
git mv old.sh new.sh

```

# Git Alias

- [10 levels of Git aliases: Beginner to intermediate concepts](#)

```
git config --global alias.st status
git config --global alias.c commit
git config --global alias.br branch
```

```
~/gitconfig
```

```
[alias]
  st = status
  c = commit
  loo = log --oneline
  # [commit] commit [comment]
  onemore = commit -a --amend --no-edit
  # [commit] commit, [commit]
  undo = reset --soft HEAD^
  # [commit] commit, [commit]
  cancel = reset --hard HEAD^
```

# Gitignore

Git ???

????????????????:

0. ?????????? .o .so ?
1. ??? Node.js ?????? node\_modules ???
2. ? Log ?????????????????? .log ?
3. ?????????????? .env ?
4. ??? (IDE) ??????????

```
.gitignore
```

```
# ignore all directories with the name test
test/

.env
log/
node_modules/
```

```
make/*.o
make/*.so
```

```
# ignores all .md files
.md
```

```
# does not ignore the README.md file
!README.md
```

## Git log

- ?? commit-id ??????? 4 - 8 ???????

```
# [4] commit [2]
```

```
git log
```

```
git log --oneline
```

```
# -p: patch, [4] commit [4]
```

```
git log -p
```

```
git log -p -2 # [4] commit [4]
```

```
# [4] commit [4]
```

```
git show <commit-id>
```

```
# [2] commit [2], [8]
```

```
git --stat
```

```
#
```

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

```
* 957ae62 - (HEAD -> master, origin/master, origin/HEAD) Fixed the bug if the file config doesn't exist (4 7 [4]) <A-Lang>
```

```
* 2b64ce6 - Just fixed some no-wrap (4 8 [4]) <A-Lang>
```

```
* 1eab2be - Added the title in the window (4 8 [4]) <A-Lang>
```

```
* 5f139b3 - Added the function contents_edit (4 8 [4]) <A-Lang>
```

```
* 727a72b - Removed the directory sshto-1.0 (4 8 [4]) <A-Lang>
```

```
* 764c2a4 - fix download with changed user (4 9 [4]) <Ivan Marov>
```

```
* b8b718f - make pause if error occurred in go_to_target (4 11 [4]) <Vaniac>
```

```
* cf20d21 - new screenshot and readme (5 [4]) <Vaniac>
```

```
* 1bcbb6a - ref (5) <Vaniac>
* 48df1bd - new screenshot and readme (5) <Vaniac>
```

## Git revert (Rollback)

?? `git revert` ????? rollback ??? commit ?????????????????? Revert ? commit ??????  
commit ?????????? (?? `git reset` ??)

Rollback ??? commit ?????

```
# Rollback the latest commit
git revert HEAD
```

Rollback ?? commit ?????

```
“ ???? revert ? commit ?????????????????????????? git merge
????????????????? revert????? git revert --abort
```

```
git revert <commit-id>
```

# FAQ

## [GitHub] ?? git push

????

```

“ remote: Support for password authentication was removed on August 13,
2021.
remote: Please see https://docs.github.com/en/get-started/getting-started-
with-git/about-remote-repositories#cloning-with-https-urls for information on
currently recommended modes of authentication.
fatal: 'https://github.com/a-lang/benchy.git/' ??????

```

?????? 2021/8/13 ??GitHub ? push ?????????????????????????????? personal token?

?? GitHub ?????? personal token

GitHub > Setting > Developer Settings > Personal Access Token > Tokens (classic)

?? token ??????git push ?????????? token?

???????? token???? token ??????

```
git config --global credential.helper cache
```

?? token ?????? token

```
git config --global --unset credential.helper
git config --system --unset credential.helper
```

## git clone ????

```
“(gnome-ssh-askpass:23713): Gtk-WARNING **: cannot open display
```

????:

```
unset SSH_ASKPASS
```

```
????: ?? ~/.bash_profile
```

```
# Fixed for the error with the git  
export GIT_ASKPASS=
```

# Git ????

## ?? rev-parse

```
# Getting the top-level directory
git rev-parse --show-toplevel

# Find your way home
git rev-parse --show-cdup

## Current location
# [REDACTED] <git-repo>/ .git [REDACTED]
git rev-parse --is-inside-git-dir
# [REDACTED] <git-repo> [REDACTED] ([REDACTED] .git [REDACTED])
git rev-parse --is-inside-work-tree
```

## ??????

- [How to Fix Merge Conflicts in Git](#)

## Git credential cache

This command caches credentials for use by future Git programs. The stored credentials are kept in memory of the cache-daemon process (instead of being written to a file) and are forgotten after a configurable timeout.

```
git config credential.helper cache
git config credential.helper 'cache --timeout=3600'
```

# GitHub

## Contribute to GitHub

Steps to contribute your changes / patches in open source repository.

1. Fork the repository
2. Create a new branch ( `git checkout -b feature-branch` )
3. Make your changes
4. Commit ( `git commit -m 'Add new feature'` )
5. Push to the branch ( `git push origin feature-branch` )
6. Open a Pull Request

## Preparing your Fork

1. Hit 'fork' on Github, creating e.g. yourname/theproject
2. Clone your project:

```
git clone git@github.com:yourname/theproject
```

3. Create a branch:

```
cd theproject  
git checkout -b feature-branch
```

## Making your Changes

1. Add changelog entry crediting yourself.
2. Write tests expecting the correct/fixed functionality; make sure they fail.
3. Hack, hack, hack.
4. Run tests again, making sure they pass.
5. Commit your changes:

```
git commit -m "Foo the bars"
```

# Creating Pull Requests

1. Push your commit to get it back up to your fork:

```
git push origin feature-branch
```

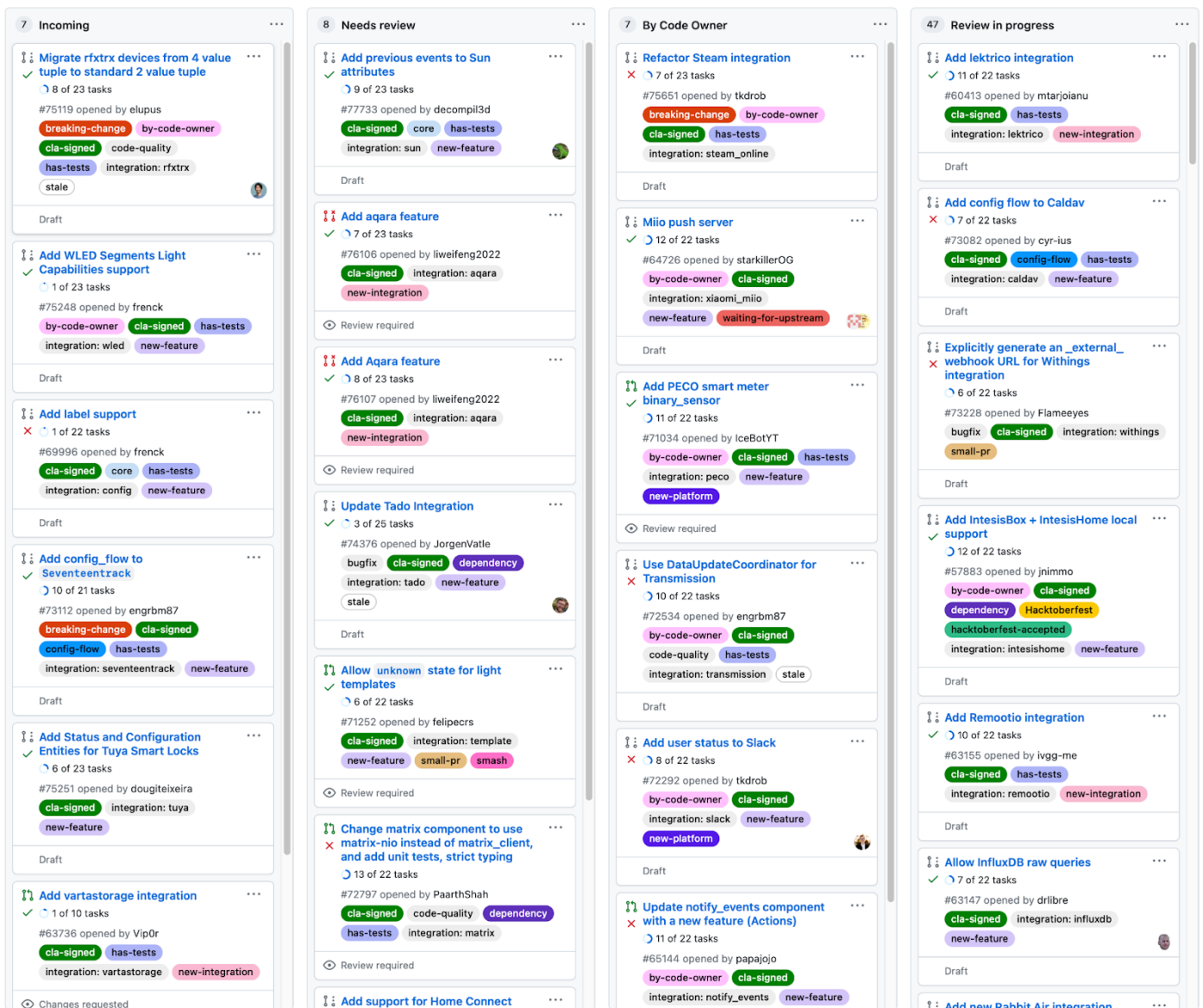
2. Visit Github, click handy “Pull request” button that it will make upon noticing your new branch.

3. In the description field, write down issue number (if submitting code fixing an existing issue) or describe the issue + your fix (if submitting a wholly new bugfix).

4. Hit ‘submit’! And please be patient - the maintainers will get to you when they can.

## GitHub Project

GitHub offers multiple tools to manage and plan your work. For example, GitHub Projects is a flexible tool for tracking and managing work on GitHub. You can use Projects to create an adaptable spreadsheet, task-board, and road map which integrates with your issues and pull requests. With GitHub projects, you can filter, sort, and group your issues and pull requests and customize to fit your team’s workflow. Projects can be created in a repository, and then issues can be added to them.



# GitHub Issues

GitHub Issues is a part of GitHub Projects, and it provides a way to track tasks that you need to complete. An issue can be a bug, a feature request, or a housekeeping task (like upgrading a library to the latest version). Issues can have extensive text and descriptions attached to them, including screenshots and snippets of code. Issues can be discussed, commented on, assigned to people, and tagged.

Filters

is:pr is:open

Labels 1,989

Milestones 1

New pull request

528 Open

56,180 Closed

Author

Label

Projects

Milestones

Reviews

Assignee

Sort

Set up shopping list during onboarding

cla-signed

core

has-tests

integration: onboarding

integration: shopping\_list

new-feature

Quality Scale: internal

small-pr

#97974 opened 14 minutes ago by frenck · Review required

9 of 20 tasks

1

Modernize aemet weather

cla-signed

deprecation

has-tests

integration: aemet

Quality Scale: No score

#97969 opened 2 hours ago by emontnemery · Review required

20 tasks

1

Use datetime.now and datetime.fromtimestamp

cla-signed

code-quality

core

has-tests

#97964 opened 4 hours ago by cdce8p · Draft

1 of 20 tasks

1

Fix docstrings in mobile app device tracker

cla-signed

code-quality

core

integration: mobile\_app

Quality Scale: internal

small-pr

#97963 opened 4 hours ago by joostlek · Review required

7 of 20 tasks

1

Modbus: set state\_class etc in slaves

bugfix

by-code-owner

cla-signed

integration: modbus

Quality Scale: gold

small-pr

#97961 opened 4 hours ago by janiversen · Review required

8 of 20 tasks

1

modbus: Repair swap for slaves

bugfix

by-code-owner

cla-signed

has-tests

integration: modbus

Quality Scale: gold

small-pr

#97960 opened 4 hours ago by janiversen · Review required

8 of 20 tasks

1

Add unique\_id to media\_player entity of frontier\_silicon

bugfix

by-code-owner

cla-signed

integration: frontier\_silicon

Quality Scale: No score

small-pr

#97955 opened 7 hours ago by wlcrs · Review required

7 of 20 tasks

1

Restore bthome state at start when device is in range or sleepy

cla-signed

has-tests

integration: bthome

new-feature

Quality Scale: No score

small-pr

#97949 opened 9 hours ago by bdraco · Approved

3 of 20 tasks

1

3

Add initial sensors for Enphase Encharge batteries

cla-signed

integration: enphase\_envoy

new-feature

Quality Scale: No score

#97946 opened 13 hours ago by cgarwood · Draft

6 of 20 tasks

8

Experiment to add support for Yale Doorman to august

by-code-owner

cla-signed

integration: august

Quality Scale: No score

#97940 opened 18 hours ago by bdraco · Draft

20 tasks

“ Tip: ????? issue ?????????? git commit ?????????? #+Issue NO. ???  
Fixed this bug #156

## Resources

- [A Quick Guide to Using GitHub for Project Management](#)
  - This article provides a brief overview of project management tools on GitHub.
- [GitHub for project management](#)
  - This lesson offers detailed descriptions of GitHub’s project management tools.
- [Using GitHub as your Project Management Tool](#)
  - This video provides examples on GitHub project management tools.
- [GitHub Issues: Project Planning for Developers](#)
  - This GitHub page shows the many project management tools available for developers.

# Branch

## Tips

- Branch : ??? commit ??????????????
- HEAD: ?????
- main (master): ??????????????
- branch-name ?????????? test/feature1, dev/feature1
- ??? branch (HEAD) ??? : `git branch` ? `git log -1`
- ?? branch ?????????? commit ?????

## Git branch

# List all the branches of local repo.

```
git branch
```

# List the branches of remote repo.

```
git branch -r
```

# List all branches of local and remote repos.

```
git branch -a
```

# Create new branch

```
git branch <branch-name>
```

```
git checkout <branch-name>
```

# Alternatively, using the following one-liner command

```
# <origin-name> [HEAD] [origin/main]
```

```
git checkout -b <branch-name> <origin-name>
```

# Remove a branch

```
# NOTE: [branch] [HEAD], [origin/main]
```

```
git branch -d <branch-name>
```

## ????

# Clone [url]

```
git clone http://your.company.com/yourname/my_proj.git
```

```
cd my_proj
```

```
# git checkout origin/main -- test/util-cmd (checkout)
```

```
git checkout -b test/util-cmd origin/main
```

```
git branch -a
```

```
# Change your codes
```

```
# git add .
```

```
git add .
```

```
git commit -m "Added a new branch test/util-cmd"
```

```
# git push --set-upstream origin test/util-cmd
```

```
git push --set-upstream origin test/util-cmd
```

# Remote repository

Git repository ?????????????? repository ?????? GitHub?Gitlab?Bitbucket??????? Git Server?

?????????Remote repository ??????????????????????????

## Clone from remote repo.

```
# Specified version
```

```
git clone -b 8.3.0 https://github.com/OpenSIPS/opensips-cp.git /var/www/opensips-cp
```

## Git remote

??

- branch (??)??? local ? remote?? remote branch ??? origin ???
- `git remote show` ?????????????? `git remote update` ?????????????? branch ?  
commit ?????? `git fetch` ???

```
# Show the configuration of the remote repository
```

```
git remote -v
```

```
origin https://github.com/a-lang/sshto.git (fetch)
```

```
origin https://github.com/a-lang/sshto.git (push)
```

```
# Get more information
```

```
git remote show origin
```

```
* origin
```

```
fetch https://github.com/a-lang/sshto.git
```

```
push https://github.com/a-lang/sshto.git
```

```
HEAD master
```

```
remote
```

```
master
```

```
'git pull' https://github.com/a-lang/sshto.git
```

```
master master
```

```
'git push' https://github.com/a-lang/sshto.git
```

```
master master (push)
```

```
# Update the contents of remote branch
git remote update
```

## Remote branches

?????? branch (??) ??

```
# Way 1
git branch -r
```

```
# Way 2
git remote show origin
```

## Push

????????????

```
# [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] refactor
git push origin refactor
git push -u origin refactor # [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] refactor
```

```
# [ ] [ ] [ ] [ ] [ ] refactor
git push --delete origin refactor
```

```
# [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]  
# [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] commit [ ] [ ] [ ] pull request [ ] [ ]  
git push -f
```

## Pull

???????????????????? branch ??????

```
cd my_proj
git pull
```

## Pull vs. Fetch

- Pull: ?????????????????? branch
- Fetch: ?????????????????? branch

???????????? 15 ??

## SSH Key Authentication

```
# [] ssh-key
# [ ]: ~/.ssh/id_rsa (private key) , ~/.ssh/id_rsa.pub (public-key)
ssh-keygen -t rsa -b 4096 -C "alang@my-linux-desktop"

# Gitlab [ ] public ssh-key
# [ ] ssh key authentication
# [ ] Welcome to GitLab, @alang! [ ]
ssh -T git@gitlab.shuraфом.eu

# [ ]
# NOTE: [ ] git@XXX.xxx.xxx [ ]
# [ ] https:// [ ] Personal Token [ ]
git clone git@gitlab.shuraфом.eu:myproject/myprog.git
```

## Fetch

[illegible]

1. ?????????????

```
git remote show origin
```

```
* remote origin
Fetch URL: https://github.com/redquinoa/health-checks.git
Push URL: https://github.com/redquinoa/health-checks.git
HEAD branch: master
Remote branch:
  master tracked
Local branch configured for 'git pull':
  master merges with remote master
```

Local ref configured for 'git push':  
master pushes to master (local out of date)

?? master pushes to master (local out of date) ????????????

## 2. ????? branches ????????

- ?? branch: origin/master
- ?? branch: master
- ????? branch log ????? HEAD -> master ? origin/master ??????????????????????

```
# [ ] [ ] branches [ ] [ ] local
git fetch

# [ ] [ ] branch [ ] commit log [ ] [ ]
git branch -r      # List all remote branches
git log origin/master # Check the log for the remote branch origin/master

commit b62dc2eacfa820cd9a762adab9213305d1c8d344 (origin/master, origin/HEAD)
Author: Blue Kale <bluekale@example.com>
Date: Mon Jan 6 14:32:45 2020 -0800
    Add initial files for the checks
commit 807cb5037ccac5512ba583e782c35f4e114f8599 (HEAD -> master)
Author: My name <me@example.com>
Date: Mon Jan 6 14:09:41 2020 -800
    Add one more line to README.md
commit 3d9f86c50b8651d41adabdaebd04530f4694efb5
Author: Red Quinoa <55592533+redquinoa@users.noreply.github.com>
Date: Sat Sep 21 14:04:15 2019 -0700
    Initial commit
```

## 3. ????? branch ??????? branch???????

- git status : ?????? branch ??? branch ??????

```
# [ ] [ ]
git status

On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
(use "git pull" to update your local branch)
```

nothing to commit, working tree clean

```
# [ ] [ ] [ ] branch [ ] [ ] branch
```

```
git merge origin/master
```

Updating 807cb50..b62dc2e

Fast-forward

```
all_checks.py | 18 ++++++
```

```
disk_usage.py | 24 ++++++
```

2 files changed, 42 insertions(+)

```
create mode 100755 all_checks.py
```

```
create mode 100644 disk_usage.py
```

????????????????

```
# [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

```
git remote show origin
```

```
# [ ] [ ] [ ] [ ] [ ] [ ]
```

```
git checkout <remote-branch-name>
```

```
# [ ] [ ] [ ] [ ] [ ] [ ]
```

```
git branch
```

# Merge

Git merge ???????

1. Fast-forward merge: ?????? commit ??????
2. Three-way merge: ????? commit ??????????????????????

## Git merge ????

???????????????????????? bug???????????????????????? (main/master)?

- `git merge <branch-name>` : ??? <branch-name> ???
- `git merge --abort` : ??????

?? fix\_something ???

```
git checkout main
git merge fix_something
```

???????? Merge conflict????????(some.py)????????

????????

- <<<< HEAD : ??? (main) ??
- >>>> fix\_something: ????? (fix\_something) ??
- ===== : ?????????
- ?????????????????????

```
<<<<<< HEAD
print("Keep me!")
=====
print("No, keep me instead!")
>>>>>> fix_something
```

????????

```
git add some.py
git status
# Check if the conflict has been fixed
git commit
```

???????

```
git log --graph --oneline
```

??

```
git merge --abort
```

## Git rebase

- [? PR ???? Git rebase ????? commit ??](#)

```
git rebase <branch-name> : Move the current branch on top of the <branch-name> branch
```

???????????????????????????? three-way merges????????????????????

```
# [ ] rebase
git rebase -i master
```

## Interactive Rebasing

??

Pull Request ??

# ????

??????

??A push ??????????:

```
! [rejected]      master -> master (fetch first)
error: failed to push some refs to 'https://github.com/redquinoa/health-checks.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

?????????B ???????? (branch) ???????? (push) ???????? (commit)?

??A ????????

1. ??????????????????: `git pull`
2. ???????????? Automatic merge failed?????????????????
3. ??????????????:
  - `git log --graph --oneline --all`
  - `git log -p origin/master`
4. ??????????
  - `?????? <<<< HEAD ? >>>> something ?????`
5. ??????????
  1. `git add`
  2. `git status`
  3. `git commit`
  4. `git push`

??????

Remote branch: refactor

1. `git checkout -b refactor` : ??????????????????
2. ???????
3. `git commit -a -m "Something"` : ?? 2, 3 ?????
4. `git push -u origin refactor` : ?????????????? branch ?????? `-u origin refactor`  
?????????



# Pull request

???????????? Git ???

```

???????????????????????????? ( ? repo)? ??????????????????????????????????GitHub ??????
fork???????????????????????????? repo????????????? fork ???

```

[illegible]

- Make changes to the file.
- Change the proposal and complete a description of the change.
- Click the Proposed File Change button to create a commit in the forked repo to send the change to the owner.
- Enter comments about the change. If more context is needed about the change, use the text box.
- Click Pull Request.

## ?????Creating a pull request

???????????????? ???? ?????????????????????????????

- **Merge commits.** All commits from the feature branch are added to the base branch in a merge commit using the `--no-ff` option.
- **Squash and merge commits.** Multiple commits of a pull request are squashed, or combined into a single commit, using the fast-forward option. It is recommended that when merging two branches, pull requests are squashed and merged to prevent the likelihood of conflicts due to redundancy.
- **Merge message for a squash merge.** GitHub generates a default commit message, which you can edit. This message may include the pull request title, pull request description, or information about the commits.
- **Rebase and merge commits.** All commits from the topic branch are added onto the base branch individually without a merge commit.
- **Indirect merges.** GitHub can merge a pull request automatically if the head branch is directly or indirectly merged into the base branch externally.

## Practice

## Basic processes

1. Fork the project you're interested in to your repository (via web)
  - Login to your GitHub repository
  - Visit the project: <https://github.com/google/it-cert-automation-practice>

- Fork your own copy of this project via GitHub's website
- Clone the repository to local
  - `git clone https://github.com/google/it-cert-automation-practice`
  - `cd it-cert-automation-practice`
- Setup a remote for the upstream repo.
  - `git remote -v`
  - `git remote add upstream https://github.com/google/it-cert-automation-practice`
  - `git remote -v`
- Configure Git
  - `git config --global user.name "Yourname"`
  - `git config --global user.email "your@email"`
- Create new local branch for the fix
  - `git branch improve-username-behavior`
  - `git checkout improve-username-behavior`
- Fix and test the code
- Commit the changes
  - `git status`
  - `git add .`
  - `git commit`
  - `git push origin improve-username-behavior`
- Create a pull request (PR) (via web)
  1. Login to your forked repository
  2. Go to Pull requests > Create a pull request
  3. Edit the title and the description for the pull request
- You can see information about the branch's current deployment status and past deployment activity on the Conversation tab of the upstream repo. .

“Tip: ?? Commit ??????????????????(123)???????????”

Updated validations.py python script.  
Fixed the behavior of validate\_user function in validations.py.  
Fix for #123

Tip: Pull Request Conversation

# Code review

[illegible]

## Code style guides

- [Style guide for Google code](#)
- [PEP 8 – Style Guide for Python Code](#)

## Code style tools

- **Black** is a PEP 8 compliant opinionated formatter with its own style

## Five tips for pull request reviews

Some of the considerations you should have with pull request reviews are:

1. **Be selective with reviewers:** It's important to select a reasonable number of reviewers for a pull request. Adding too many reviewers can lead to inefficient use of resources, as too many people reviewing the same code may not be productive.
2. **Timely reviews:** Ideally, reviews should be completed within two hours of the pull request being submitted. Delays in reviews can lead to context switching and hinder overall productivity.
3. **Constructive feedback:** Feedback should be constructive and explain what needs to be changed and, more importantly, why those changes are suggested. Friendly and non-accusatory language fosters a positive and collaborative atmosphere.
4. **Detailed pull request description:** The pull request should include a detailed description that covers the changes made in the feature branch compared to the development branch, prerequisites, usage instructions, design changes with comparisons to mockups, and any additional notes that reviewers should be aware of. This information ensures that reviewers have a comprehensive understanding of the changes.
5. **Interactive rebasings:** Interactive Rebasing allow developers to modify individual commits without cluttering the commit history with redundant or unrelated changes. Keeping commits clean and relevant contributes to a more organized and maintainable codebase.

# Cheat Sheets

## Git Commands

### Git Commands Cheat Sheet

 ByteByteGo

#### Basics

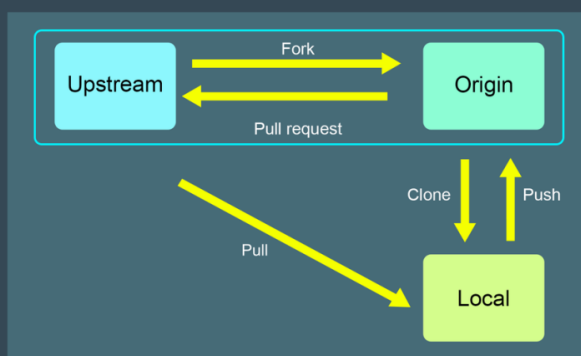
**git help [commands]**

Use this when you are stuck

#### Terminology

|        |                           |
|--------|---------------------------|
| main   | default primary branch    |
| origin | default upstream branch   |
| HEAD   | current branch            |
| HEAD^  | parent of HEAD            |
| HEAD~3 | great grandparent of HEAD |

#### Start to Work



#### git init

create a new local Git repo

#### git clone

copy a repo

#### git pull

fetch remote version and update local branch

#### git add [file]

stage changes to tracked and untracked files

#### git commit

commit previously staged changes

#### git push origin HEAD

push local changes to the origin

#### Branches

##### Git Branch WorkFlow



#### git branch --all

list all the local and remote branches

#### git checkout hotfix

change to an existing branch called hotfix

#### git checkout main

#### git merge hotfix

merge branch changes from hotfix to main

#### git log --graph --oneline

show a pretty branch history

#### Conflicts

#### git diff

see specific local changes

#### git diff --ours

compare the working tree with our branch

#### git diff --theirs

compare the working tree with their branch

#### Useful Tools

#### git archive

create release tarball

#### git cherry-pick [commit-id]

enable arbitrary Git commits to be picked by reference

