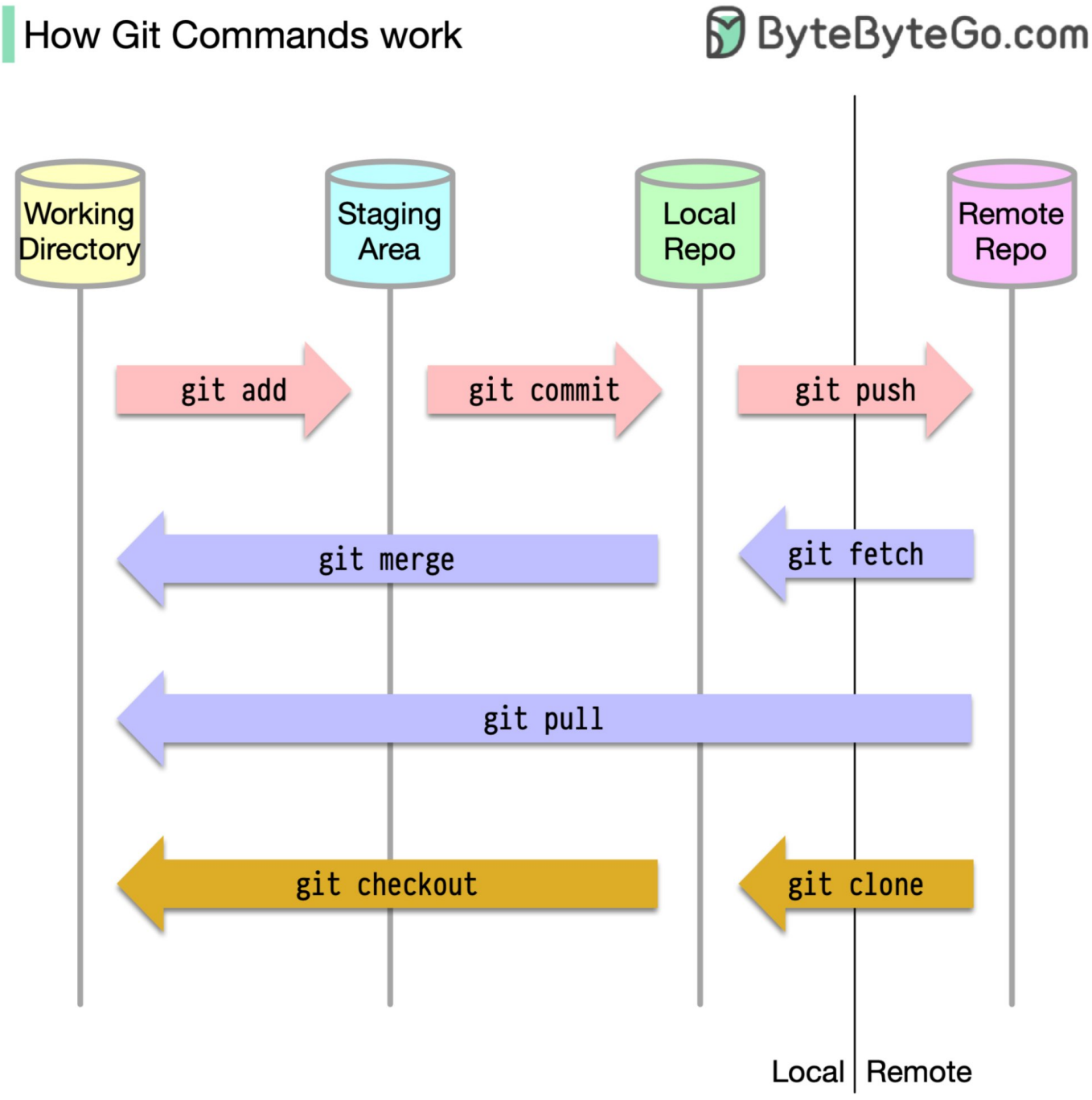
Git

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

- Learning Git
- Git Installation
- Git Tips
- FAQ
- Git ????
- Contribute to Github

Learning Git

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

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 init
git config --global user.name "<user-name>"
git config --global user.email "<your-email-addr>"
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 ???

```
git mv <old-folder> <new-folder>
```

Commit ??

??? commit ??

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

? add ?? commit (?? edited & deleted)

```
git commit -am "<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

?? Comments

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

?????? Comment

```
git commit -v --amend
```

Clone for specified branch version

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

SSH Key Authentication

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

# Gitlab 網站匯入 public ssh-key
# 測試 ssh key authentication
# 如果輸出 Welcome to GitLab, @alang! 表示連線認證成功
ssh -T git@gitlab.shurafom.eu

# 下載專案
# NOTE: 位址必須是 git@XXX.xxx.xxx 開頭。
# 如果使用 https:// 則必須改用 Personal Token 認證方式
git clone git@gitlab.shurafom.eu:myproject/myprog.git
```

diff

```
# 還沒執行 git add 前, 比對目前檔案與最近一次 commit 版本的內容差異
git diff file
```

```
# 比對兩個檔案的內容差異
git diff --word-diff file1 file2

# 檢視兩個 Commit 版本的內容差
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

Local unstaged changes *(??? add)*

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

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
```

Git Rebase

- ? PR ???? Git rebase ????? commit ??

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
    # 重新修改最後一筆 commit 的 comment
    onemore = commit -a --amend --no-edit
    # 刪除最後一筆 commit, 保留文件的修改
    undo = reset --soft HEAD^
    # 刪除最後一筆 commit, 不保留文件
    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
```

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
# 判斷是否在專案目錄 <git-repo>/.git 底下
git rev-parse --is-inside-git-dir
# 判斷是否在專案目錄 <git-repo> 底下 (不包含 .git 目錄下)
git rev-parse --is-inside-work-tree
```

??????

- How to Fix Merge Conflicts in Git

Contribute to Github

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

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 foo-the-bars 3.5.
```

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 HEAD
```

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.