SSE2033

SUNG KYUN KWAN UNIVERSITY

Makefile, Git

Prof. Jinkyu Jeong (jinkyu@skku.edu)
TA – Gyusun Lee (gyusun.lee@csi.skku.edu)
TA – Jiwon Woo (jiwon.woo@csi.skku.edu)
Computer Systems and Intelligence Laboratory (http://csi.skku.edu)
Sung Kyun Kwan University

Makefile

Why makefile?

- Simplify compiling source codes
- Describe the relationships among files
- Provide commands for updating each file
- Recompile each changed file

```
gcc -o test main.c test.c hello.c
```

VS

```
make
```

Rule

```
target ... : prerequisites ...
    recipe
    ...
    ...
```

- A **rule** explains how and when to remake certain files, or to carry out an action
- A **target** is the name of a file that is generated by a program, or the name of an action to carry out (ex. clean, install)
- A **prerequisite** is a file that is used as input to create the target
- A **recipe** is an action that 'make' carries out

Example

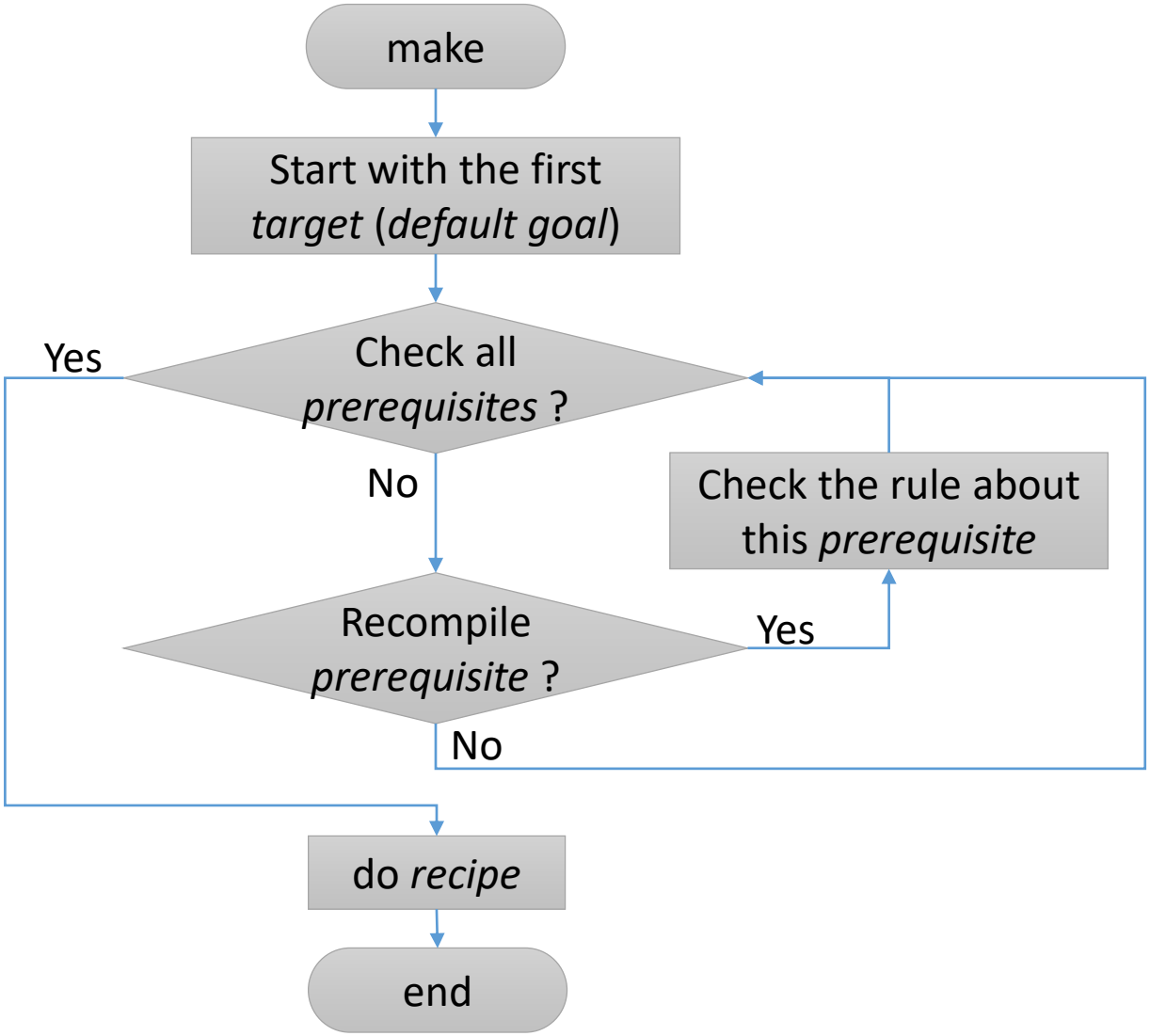
Target 'hello' depends on 'main.o' and 'hello.o'

```
# Makefile

hello: main.o hello.o
    gcc -W2 -o hello main.o hello.o
main.o: main.c
    gcc -W2 -c main.c
hello.o: hello.c
    gcc -W2 -c hello.c
clean:
    rm *.o hello
```

Target 'clean' is not a file, but it is the name of an action

```
$ make
$ ls
$ hello hello.c hello.o main.c main.o
```

How *make* processes a makefile



Example

- If ‘main.c’ is modified and enter *make* command

```
# Makefile
①Find default goal
hello: main.o hello.o
    gcc -W2 -o hello main.o hello.o ⑤Do recipe
main.o: main.c    ②Check
    gcc -W2 -c main.c    ③Do recipe
hello.o: hello.c    ④Check
    gcc -W2 -c hello.c
clean:
    rm *.o hello
```

Variable

- **Be defined once and substituted in multiple places**
- **Substitute the variable's value by writing $(*variable*)**

```
# Makefile
TARGET=hello
CC=gcc
CFLAGS=-W2
OBJECTS=main.o hello.o

$(TARGET): $(OBJECTS)
    $(CC) $(CFLAGS) -o $(TARGET) $(OBJECTS)
main.o: main.c
    $(CC) $(CFLAGS) -c main.c
hello.o: hello.c
    $(CC) $(CFLAGS) -c hello.c
clean:
    rm $(OBJECTS) $(TARGET)
```

Automatic variables

- $@ : the file name of the target of the rule
- $^ : the names of all the prerequisite
- $? : the names of all the prerequisites that are newer than the target
- $< : the name of first prerequisite

```
$(TARGET): $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^
main.o: main.c
    $(CC) $(CFLAGS) -c $^
hello.o: hello.c
    $(CC) $(CFLAGS) -c $^
```

Special built-in targets

- **.PHONY**
  - This target is not really the name of a file
  - Two reasons to use a phony target
    - avoid conflict with a file of a same name
    - improve performance
- **Others**
  - .SUFFIXES, .DEFAULT, .POSIX, etc.

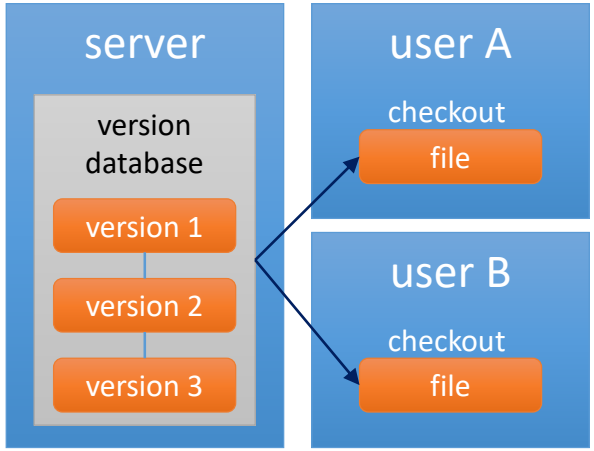
```
.PHONY: clean
clean:
    rm *.o hello
```

Git

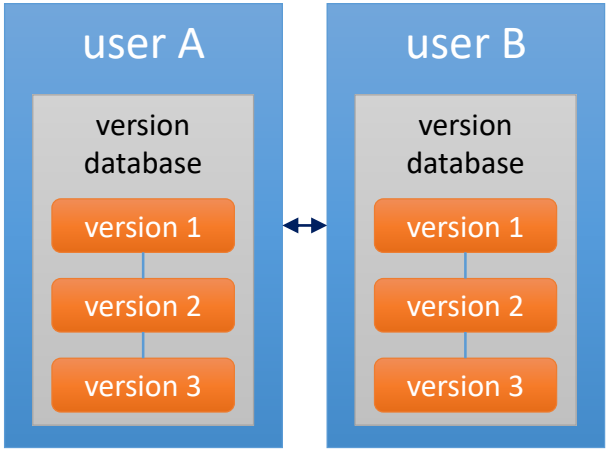
Version Control System (VCS)

- **Manage changes of documents, computer programs, and other collections of information**

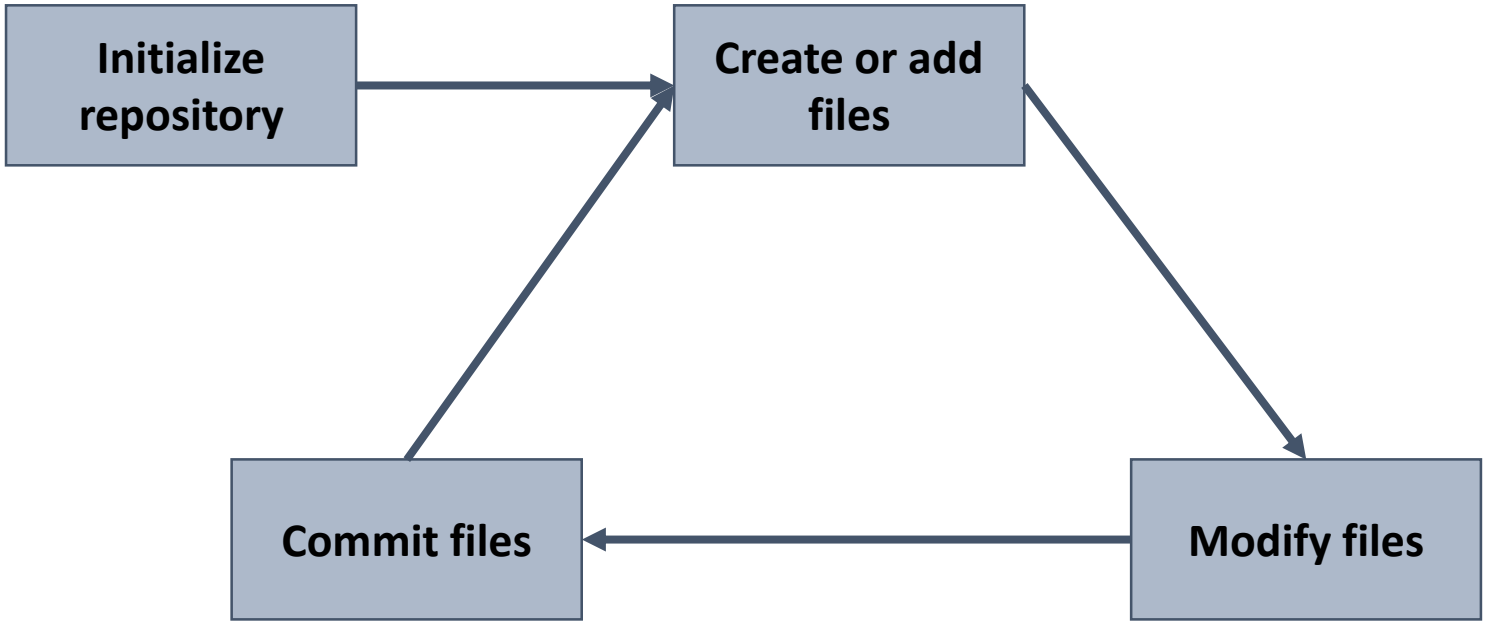
Centralized revision control system



Distributed revision control system



Work flow on local repository



Install & Setup

- **Install**
  - Linux : sudo apt-get install git
  - Windows, Mac : download at http://git-scm.com/
- **User setup**
  - git config --global user.name “name”
  - git config --global user.email “e-mail”

```
commit 242d0e6edd9d6b110fe877b65e7f46b913d0c1ee
Author: Donghyun Kim <wadong100@gmail.com>
Date:   Mon Oct 15 01:22:26 2018 -0700

    remote repository add a README.md
```

Git basic command

- **git init**
  - Create an empty Git repository or reinitialize an existing one
- **git add “filename”**
  - Add file contents to the index
- **git rm “filename”**
  - Remove files from the working tree and from the index
- **git commit**
  - Record changes to the repository
  - options
    - -a : Tell the command to automatically stage files that have been modified
    - -m “msg” : Use the given “msg” as the commit message
- **git status**
  - Show the working tree status

Example

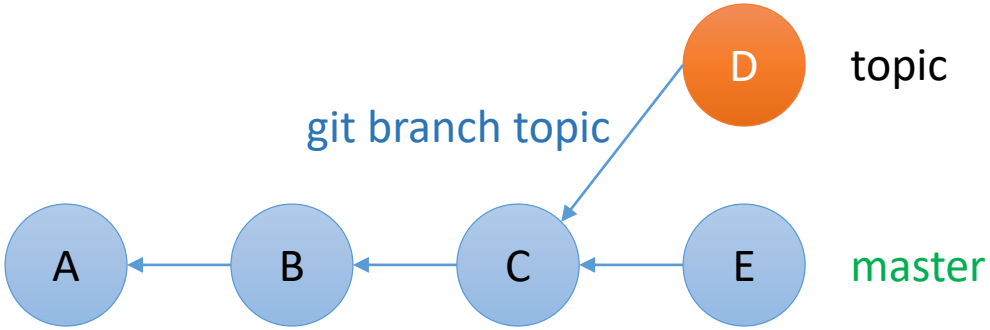
```
$mkdir git_tutorial && cd git_tutorial
~/git_tutorial git init
~/git_tutorial vi hello.c
~/git_tutorial git add hello.c
~/git_tutorial git status
~/git_tutorial git commit
~/git_tutorial vi hello.c
~/git_tutorial git commit –m “Modify hello.c file”
```

```
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   hello.c
```

```
Create hello.c file
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   new file:   hello.c
#
```

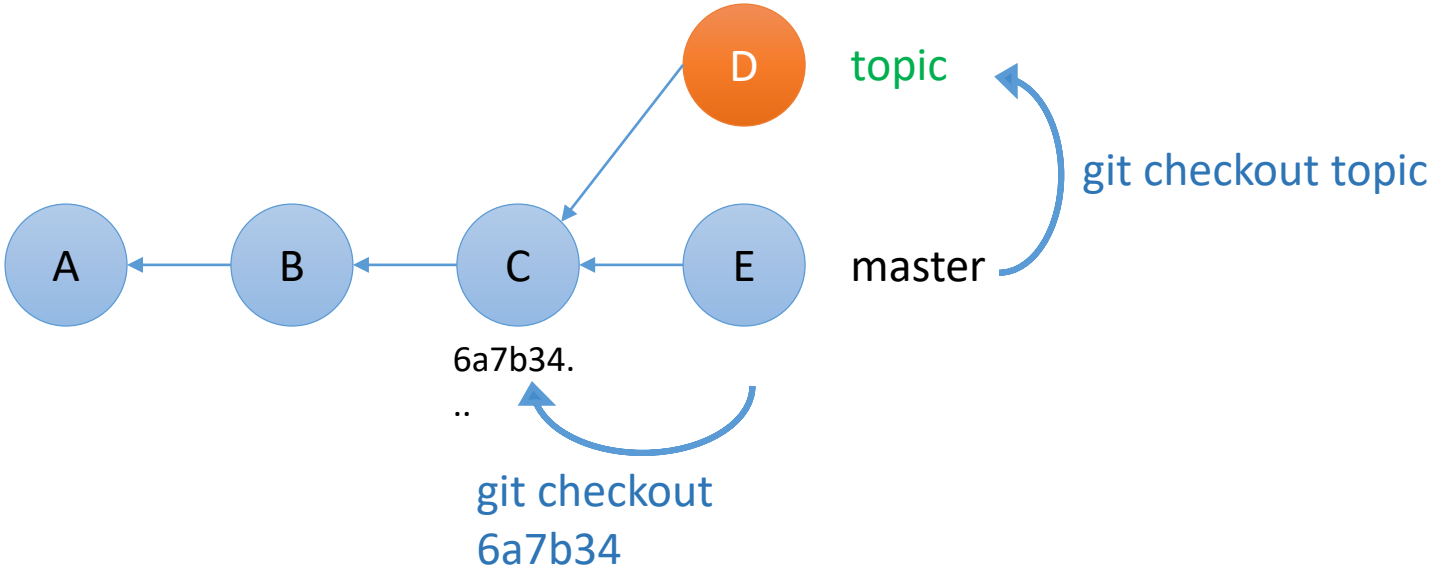
Git branch command

- **git branch**
  - List, create, or delete branches
  - options
    - [-d] “branchname” : The name of the branch to create or delete



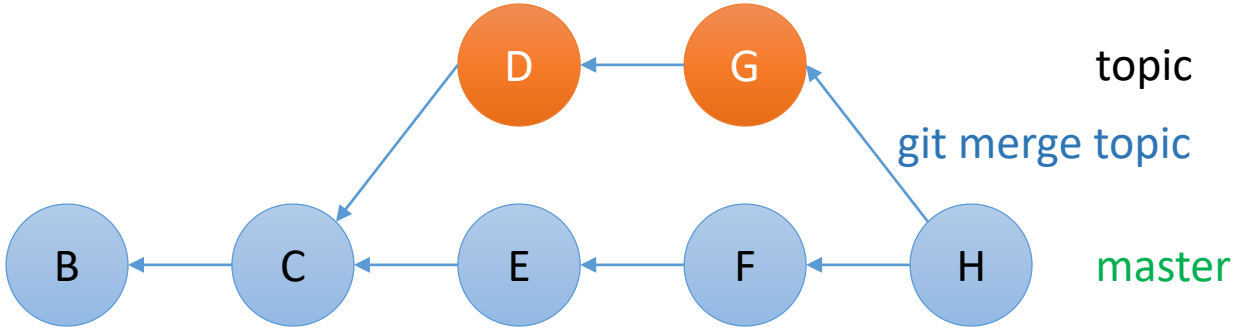
Git branch command

- **git checkout**
  - Checkout a branch or paths to the working tree
  - options
    - “branchname” : switch to “branchname”
    - -b “newbranch” : create “newbranch” and switch



Git branch command

- **git merge**
  - Join two or more development histories together
  - options
    - “branchname” : Reply the changes of “branchname” on top of current branch



Conflict

master

```
#include <stdio.h>

int main(void) {
    printf("Hello!\n");
    printf("Master!\n");
}
```

topic

```
#include <stdio.h>

int main(void) {
    printf("Hello!\n");
    printf("Topic!\n");
}
```

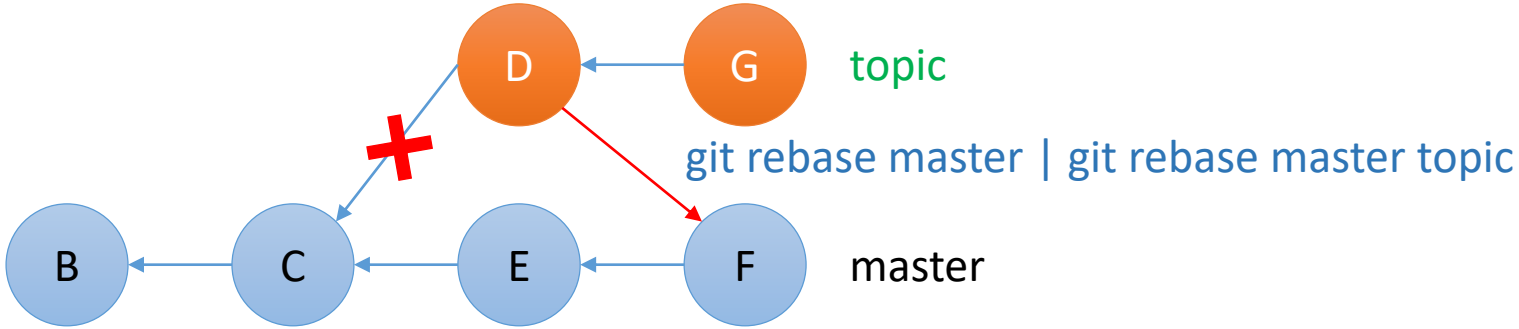
git merge topic

```
#include <stdio.h>

int main(void) {
    printf("Hello!\n");
<<<<<<< HEAD
    printf("Master!\n");
=======
    printf("Topic!\n");
>>>>>>> topic
}
```

Fix conflict and commit

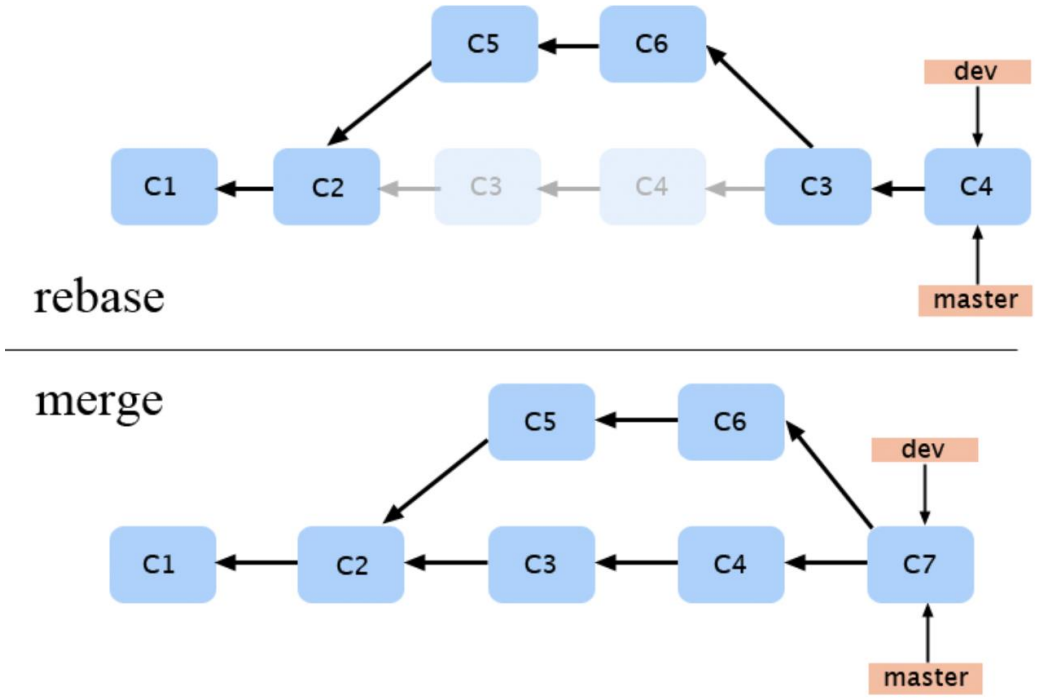
Git branch command

- **git rebase**
  - Forward-port local commits to the updated upstream head
  - options
    - git rebase “upstream”
    - git rebase “upstream” “branch”
    - git rebase –onto “newbase” “upstream” “branch”

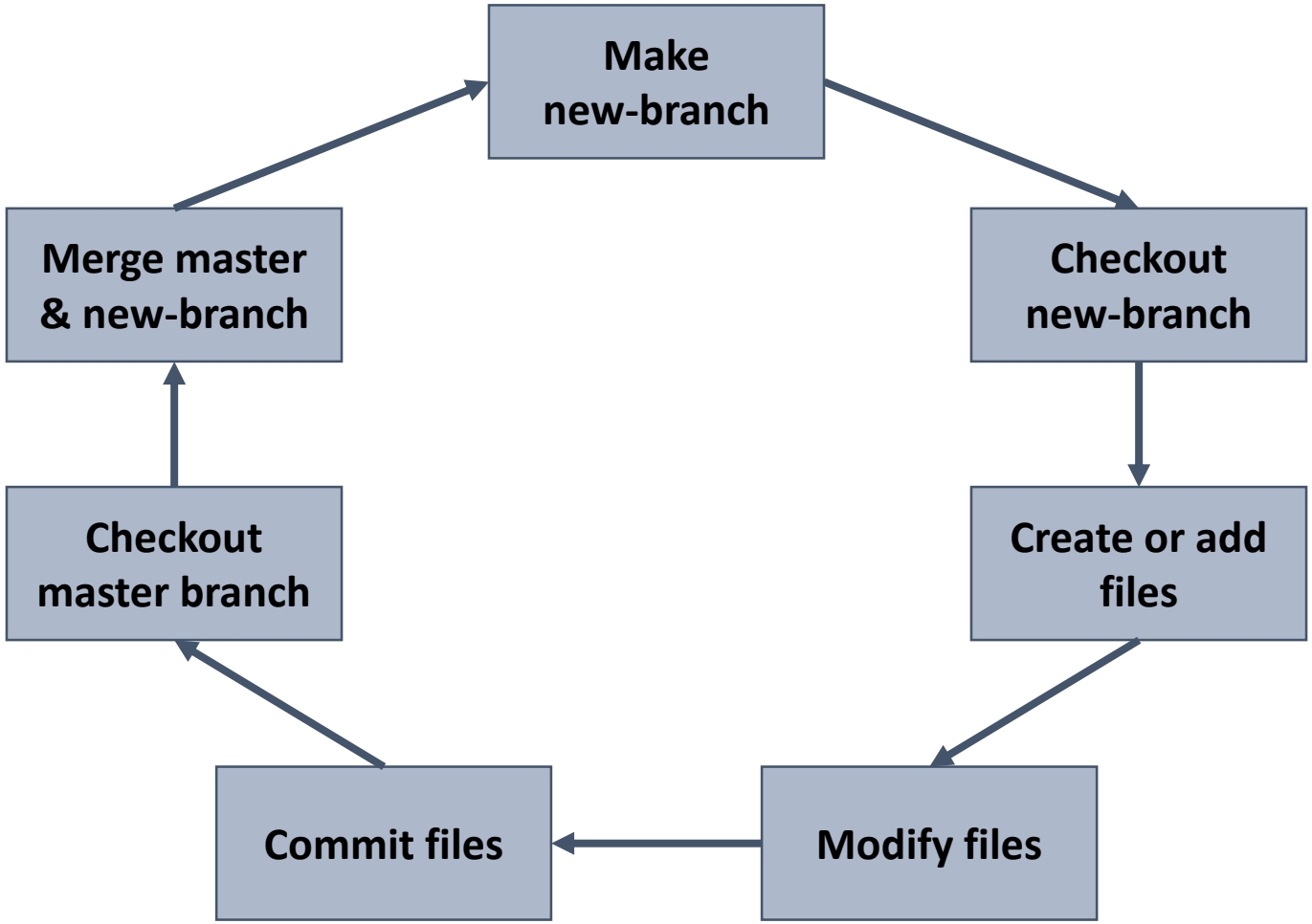


Git branch command

- **git merge vs git rebase**

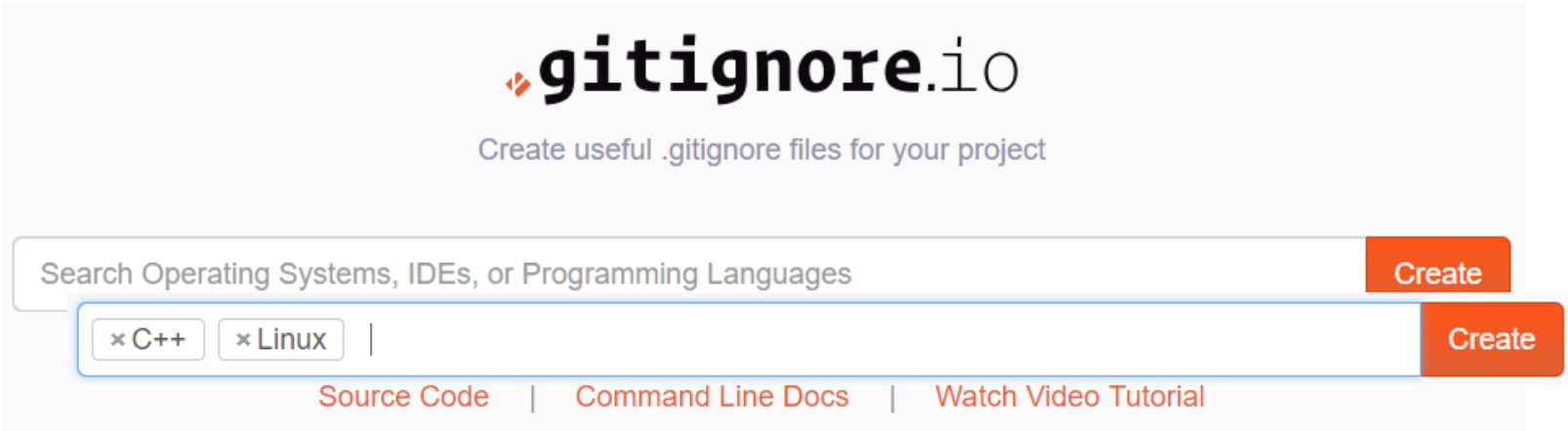


Work flow on local repository



.gitignore

- Ignore auxiliary files such as logs, input/out data, etc
- Generate automatically at https://www.gitignore.io/




```
~/git_tutorial git add .gitignore
~/git_tutorial git commit –m “added ‘.gitignore’ file”
```

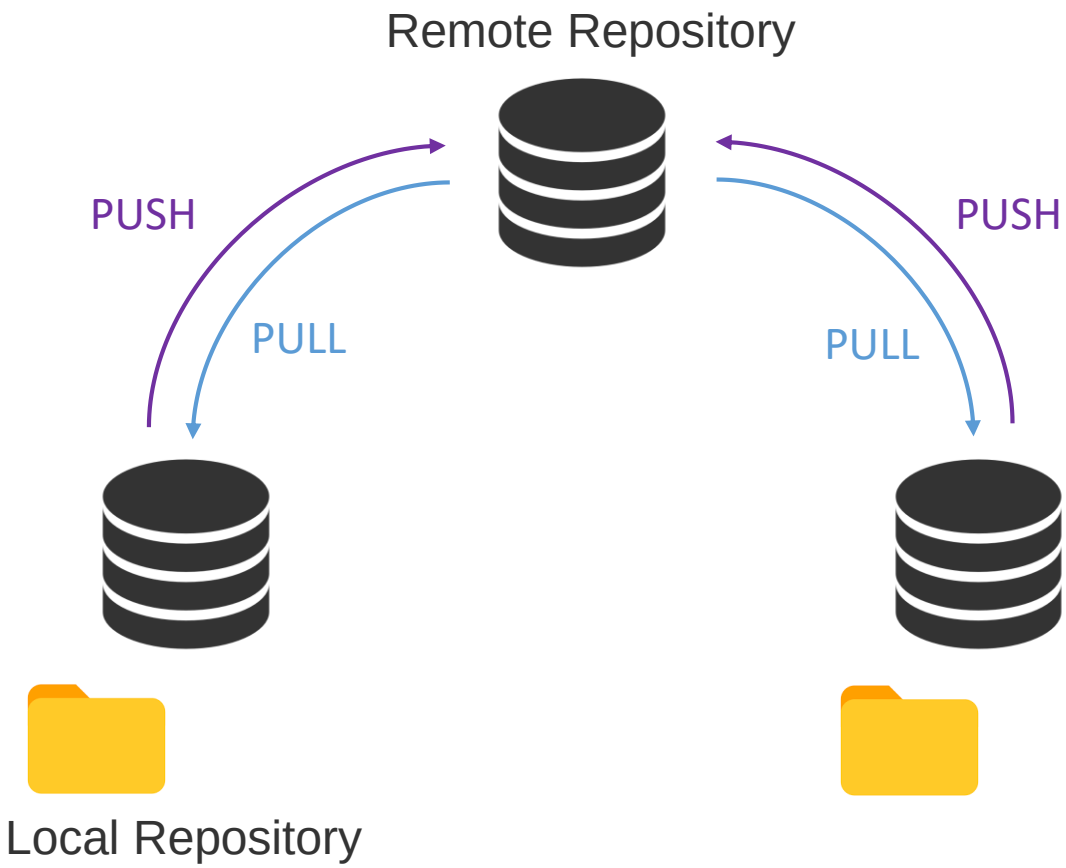
Git log command

- **git log**
  - Show commit logs
  - options
    - -p : Show all changes at each commit
    - --stat : Show statistics about modified files at each commit
    - --name-only : Show only modified file name at each commit
    - --relative-date : Show commit log with relative date
    - --graph : Draw a text-based graphical representation of the commit history

GitHub

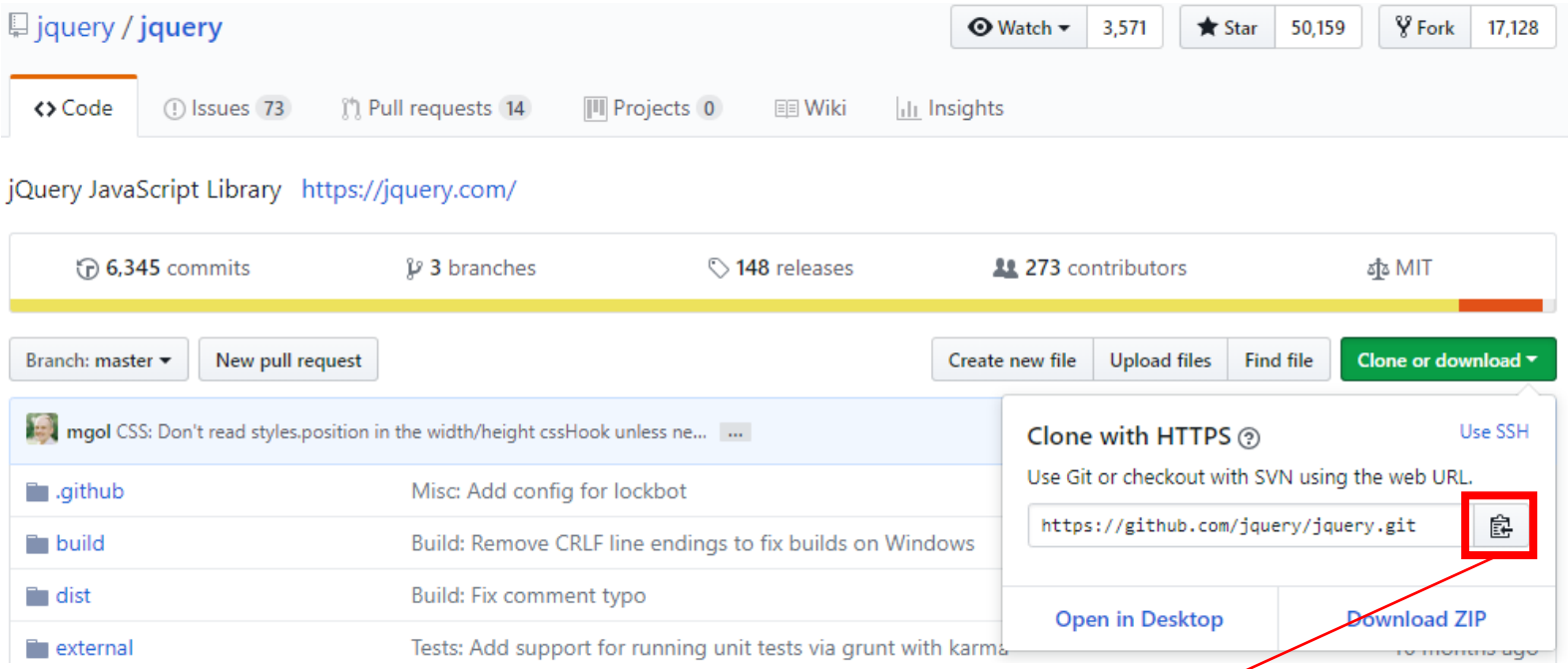
- **Remote repository (place of co-work)**
- **Sign up for GitHub**
  - https://github.com
- **Public repository for free user**
- **Functions**
  - Fork : Copy other user's repository
  - Pull requests : Communication with users
  - Issues : Discuss issues between users in repository
  - Wiki : Create a structured record of repository

Remote repository



GitHub basic command

- **git clone**
  - Copy remote repository to local repository



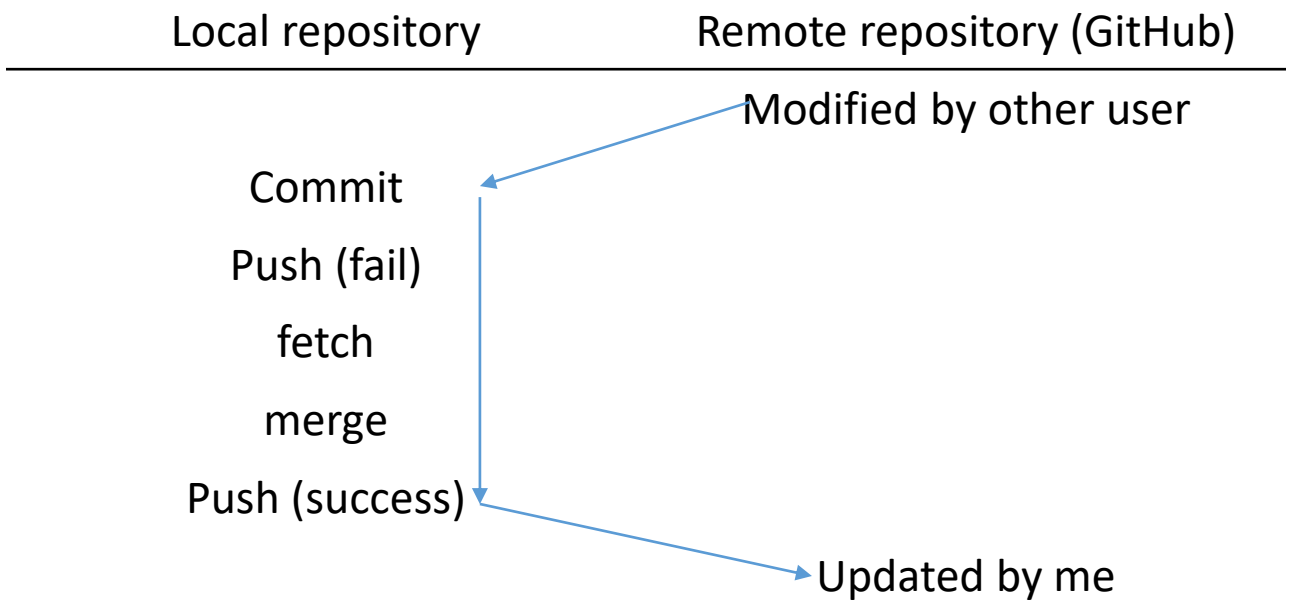

```
~/git_tutorial git clone https://github.com/jquery/jquery.git
```

GitHub basic command

- **git remote**
  - Link local repository and remote repository
  - options
    - -v : Check the connection with local and remote repository
    - add “name” “url” : Add a remote named “name” for the repository at “url”
- **git push**
  - Push local repository contests to remote repository
  - options
    - “repository” : destination (name or url)
    - --all : push all modified contents in local repository
- **git diff**
  - Show changes between local and remote

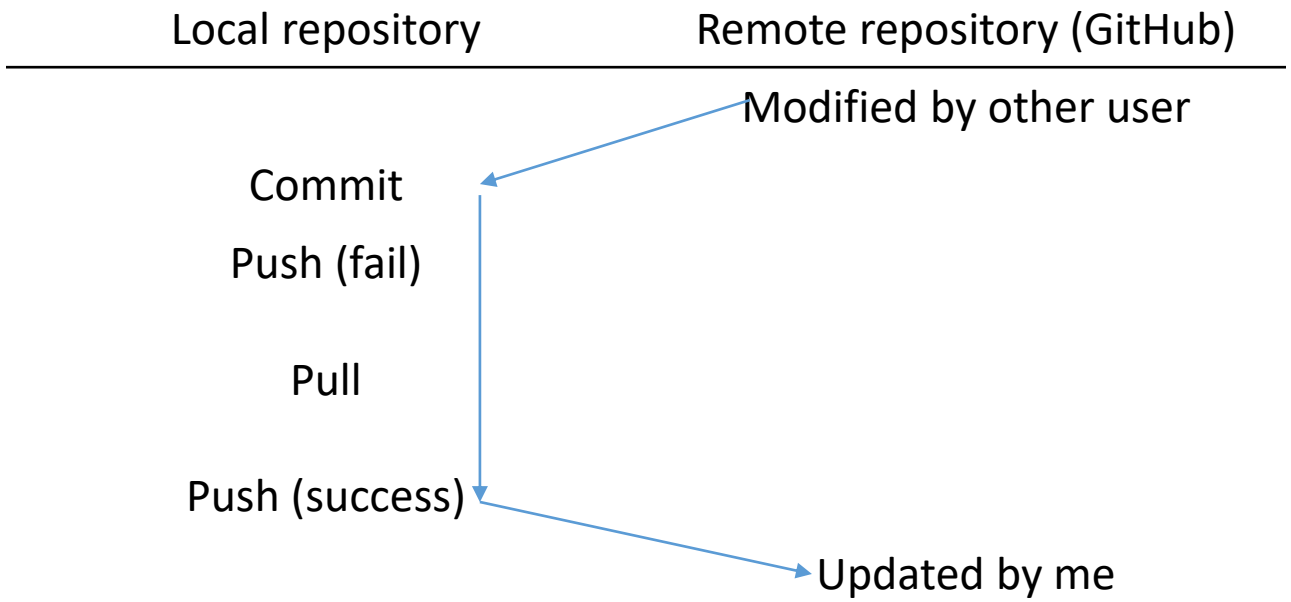
GitHub basic command

- **git fetch**
  - Fetch contents from remote repository
  - options
    - “repository” : name or url of remote repository
    - --all : Fetch all contents from remote repository



GitHub basic command

- **git pull**
  - Fetch and integrate contents from remote repository
  - options
    - “repository” : name or url of remote repository
    - --all : Fetch all contents from remote repository



How to write a Git commit message

1. Separate subject from body with a blank line
2. Limit the subject line to 50 characters
3. Capitalize the subject line
4. Do not end the subject line with a period
5. Use the imperative mood in the subject line
6. Wrap the body at 72 characters
7. Use the body to explain what and *why* vs. *how*

Reference

- **Pro Git 2nd Edition**
  - https://git-scm.com/book/en/v2
- **How to write a git commit message**
  - https://chris.beams.io/posts/git-commit/
- **Command "git –help"**
  - ex) git checkout --help