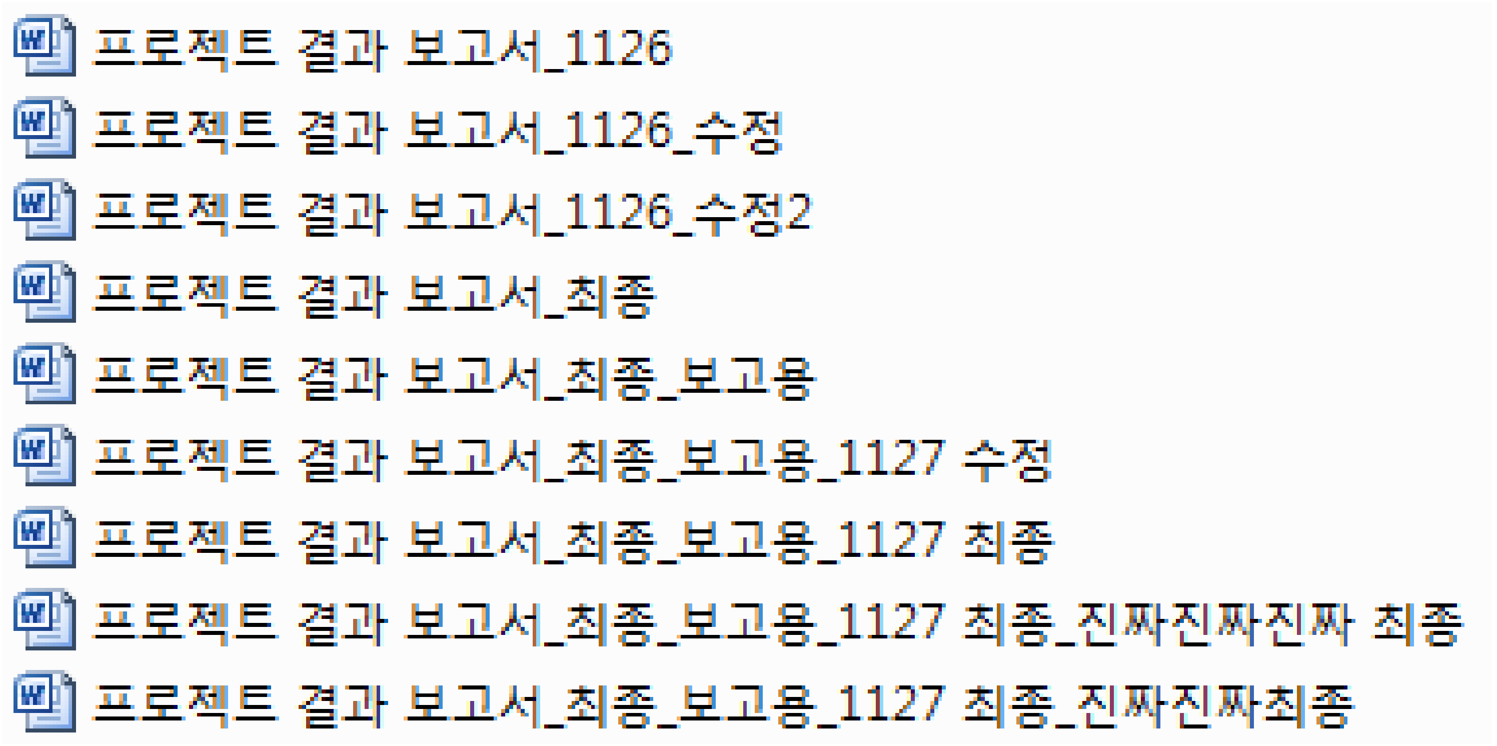
5118008 English for Software Developer

Introduction to Git and GitHub

8 Apr 2024

Shin Hong



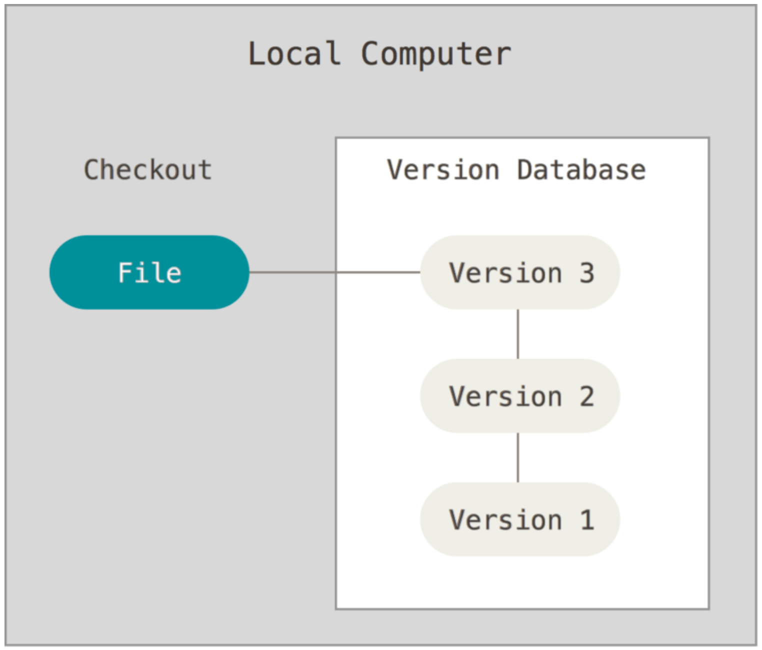
프로젝트 결과 보고서_1126
프로젝트 결과 보고서_1126_수정
프로젝트 결과 보고서_1126_수정2
프로젝트 결과 보고서_최종
프로젝트 결과 보고서_최종_보고용
프로젝트 결과 보고서_최종_보고용_1127 수정
프로젝트 결과 보고서_최종_보고용_1127 최종
프로젝트 결과 보고서_최종_보고용_1127 최종_진짜진짜진짜 최종
프로젝트 결과 보고서_최종_보고용_1127 최종_진짜진짜최종

Version Control System (VCS)

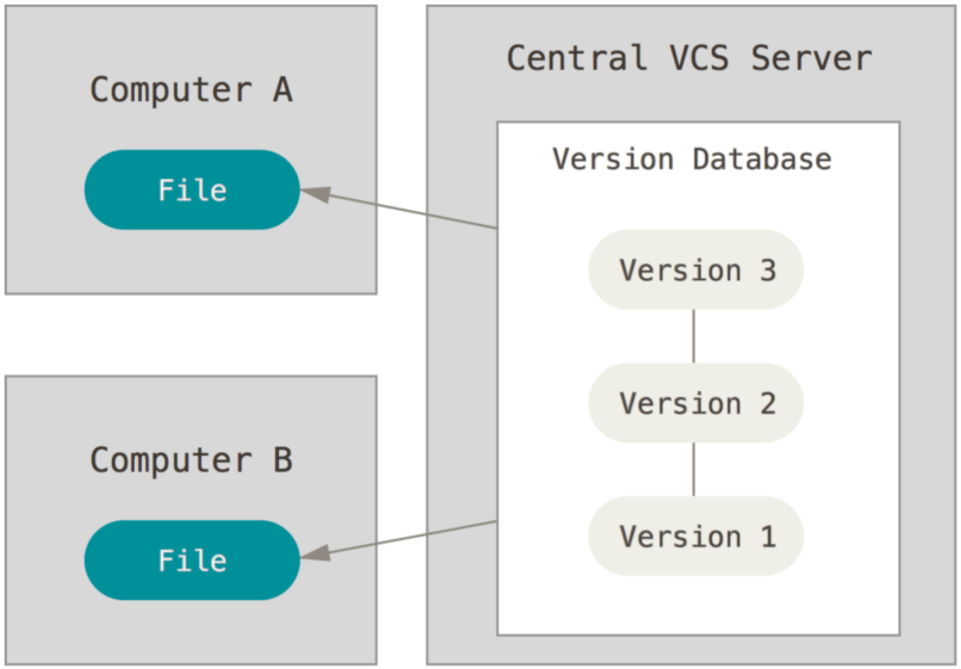
- VCS is a system that records changes to files over time
  - aims at supporting
    - incremental development
    - divergence
    - collaboration

- VCS basically provides
  - systematic back-up
  - time travelling
  - variant management
  - correct & convenient synch. with collaborators

Three Types of VCS (1/2)



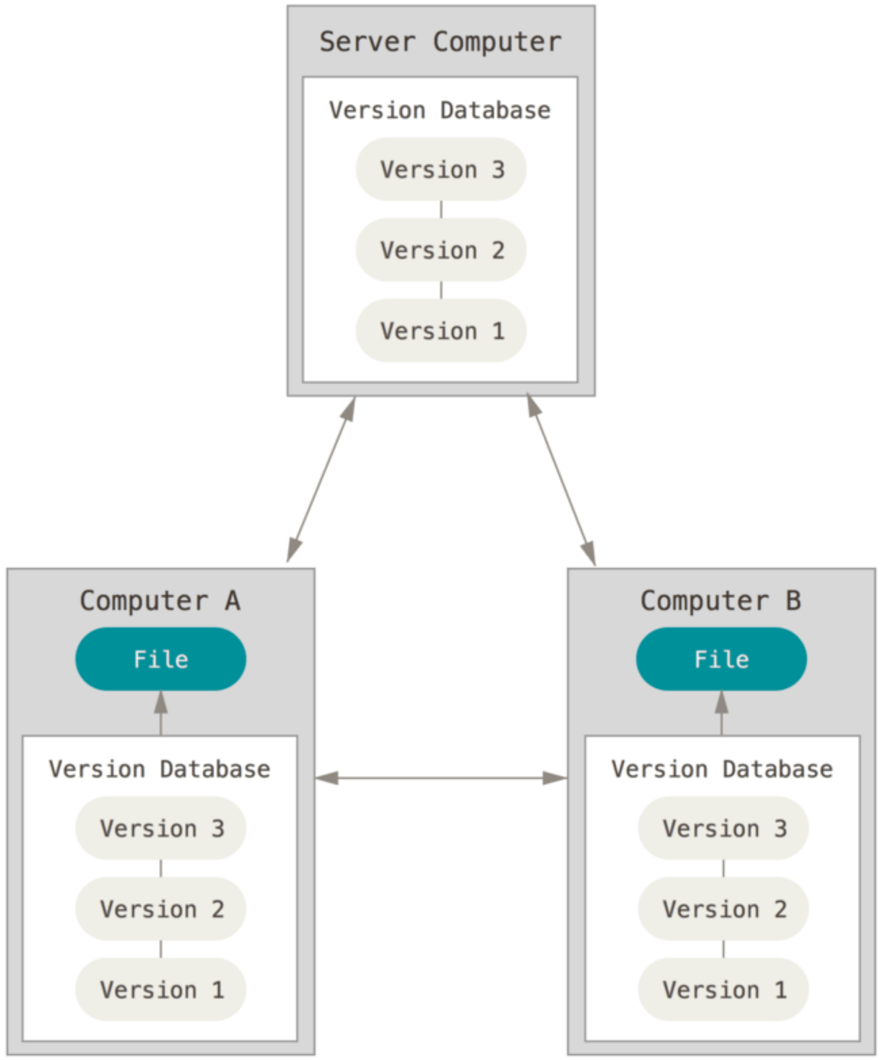
<Local VCS>



<Centralized VCS>

rcs

Three Types of VCS (2/2)



mercurial

<Distributed VCS>

Git

- Born in 2005 to support version control of Linux kernel
- Free and open sourced
- Fully distributed VCS
- Strong support for managing a large number of variants
- Efficient at handling large projects like Linux kernel

Github

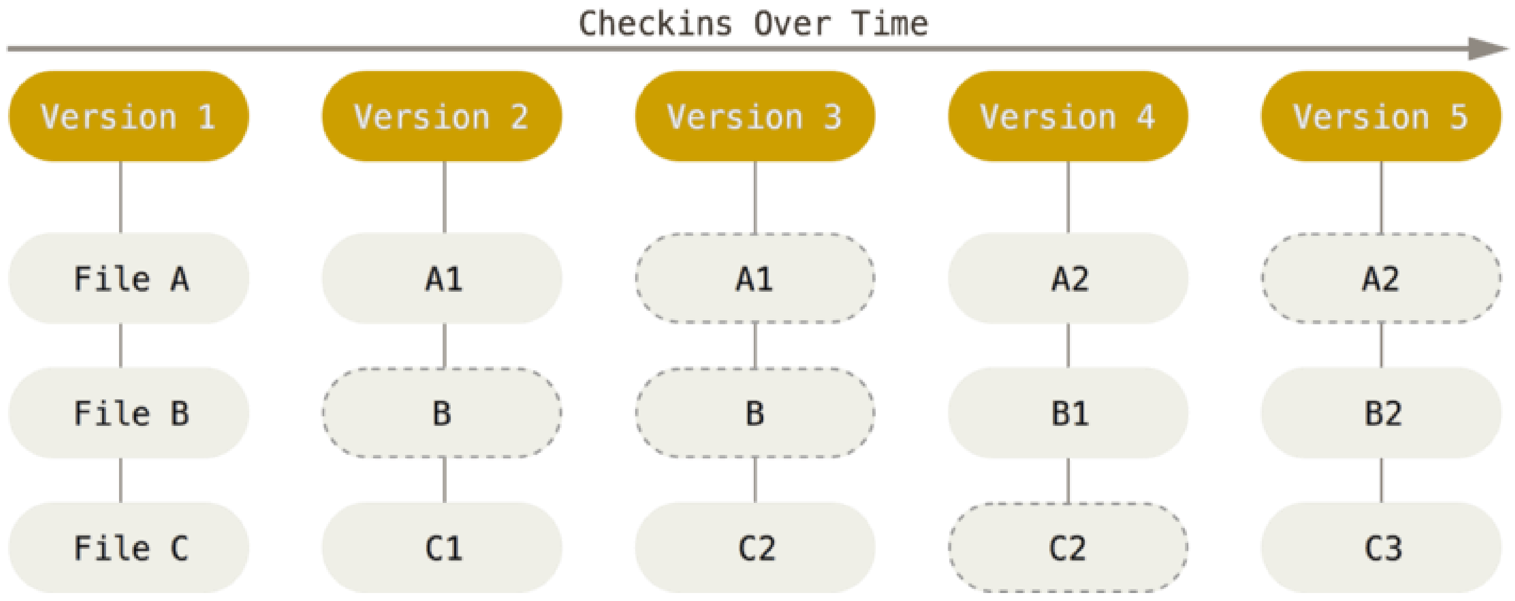
- Github
  https://github.com

- Get the Student Developer Pack from Github Education
  https://education.github.com/

Git in a Nutshell

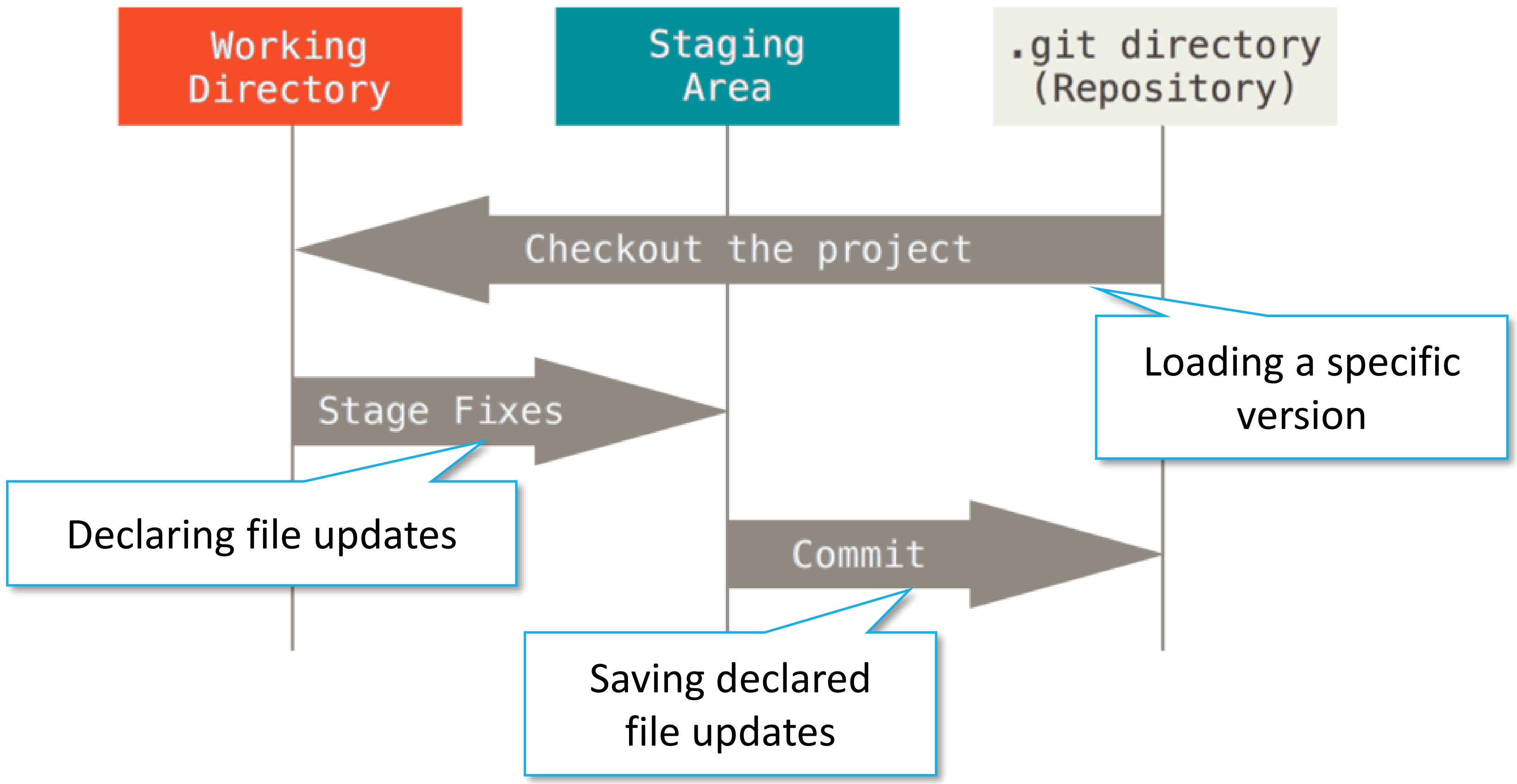
- Git works as if it is an application-level file system
- A user can declare a directory as a git repository
    - The directory becomes the working directory
    - The git directory (`.git`) is created to save the database
- A user can edit files in the working directory as usual
- A user can call git when it needs version control
    - store the history of each file in the directory
    - store the snapshot of the directory
    - switch to a specific version of the directory
    - synchronize the current version with another version

Repository Version as a Snapshot of Files



- Git saves a file at every update declaration and stores the file images in sequences
- Git represents a version of a repository as a map from file names to their specific versions of the files
- Git calls a version of a repository by the checksum of the file contents and the metadata generated by SHA-1
  - 40 characters string of hexadecimal characters
  - E.g., "24b9da6552252987aa493b52f8696cd6d3b00373"

Stage, Commit and Checkout



Git Commands

- Basic commands
  - no variation, no collaboration
- Commands to work with remote
  - collaboration
- Commands to manage branches
  - variations

Configuration

- Three configuration files
  - /etc/gitconfig: setting for every user on the system
  - ~/.gitconfig : setting for a specific user.
  - .git/config: setting for a particular project

- Important config values

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

```
$ git config --global core.editor "'C:/Program
Files/Notepad++/notepad++.exe' -multiInst -nosession"
```

Initializing Git Repository

- Start to track an existing directory in git
- git init

```
$ ls
README.md

$ git init

$ ls
.git  README.md
```

Checking Repository Status

- Query the current status of the working directory and the git repository
- git status

```
$ git status

On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be
committed)

    README.md

nothing added to commit but untracked files present
(use "git add" to track)
```

Tracking a New File

- Start to track a file in Git
- git add <file name>

```
$ git add README.md

$ git status

On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:

    new file:   README.md
```

Staging a File Update

- Stage a modified file (declare a file update for next commit)
- git add <file name>

```
$ vim README.md
$ git status

On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:

    modified:   README.md

$ git add README.md
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:

    modified:   README.md
```

Committing Staged Changes

- Save the staged files to the git directory
- Take a new snapshot of the working directory and save the snapshot as a commit (version)
- Every commit has pointers to its parent commits
- git commit

```
$ git commit

[master
 1 file
 create
```

```
# Please enter the commit message for your changes. Lines
starting
# with '#' will be ignored, and an empty message aborts the
commit.
# On branch master
# Your branch is up-to-date with 'origin/master'.
#
# Changes to be committed:
#       new file:   README.md
#       modified:   README.md
~
~
".git/COMMIT_EDITMSG" 9L, 283C
```

- git comm

Removing a File from Tracking

- Erase a file and remove the file from the working directory

```
$ rm REAMDE.md
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:

    deleted:  REAMDE.md
```

- Remove a file from tracking and keep the file in the directory

```
$ git add README.html
$ git rm --cached README.html
$ ls
.git  README.md  README.html
```

Renaming a Tracked File

- Rename a tracked file with a new name

```
$ ls
.git  README.md README.html
$ git mv README.md README
$ ls
.git   README   README.html
```

Checking Commit History

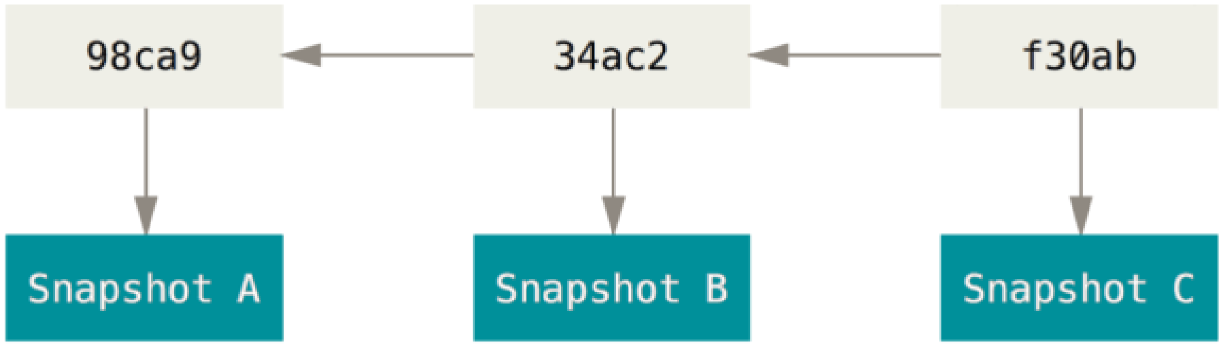
- Show the commit history at the working directory
- git log

```
$ git log

commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the version number

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 16:40:33 2008 -0700

    removed unnecessary test
…
```

Going Back to a Commit

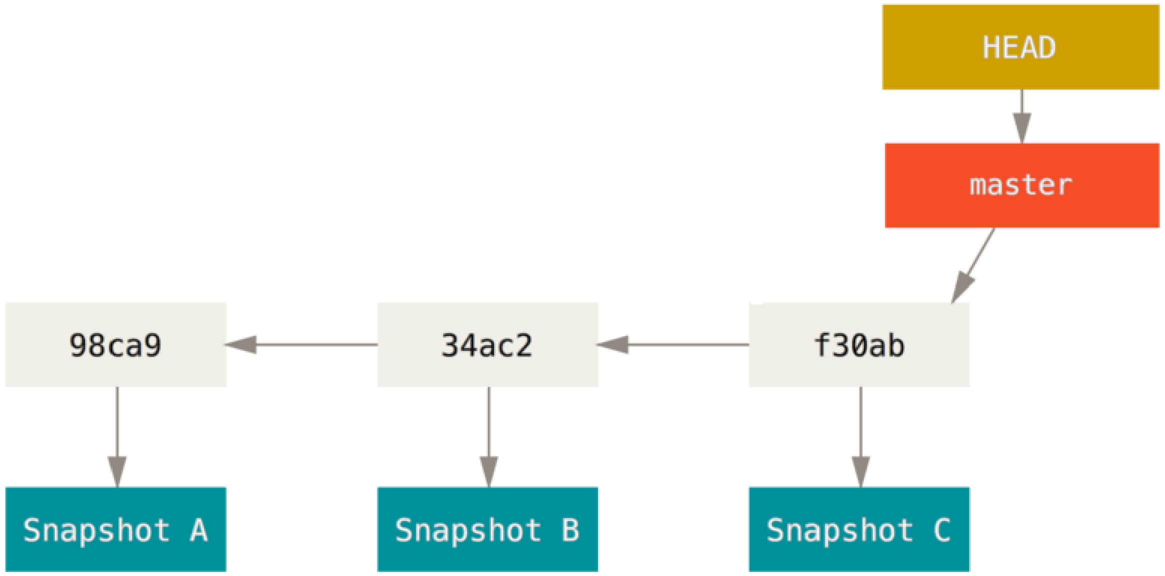
- Put the files at a specific commit to the working directory
- Must commit the staged files or stash them before a checkout
- git checkout

```
$ git checkout 34ac26dff817ec66f44342007202690a93763949
```



Branch

- A branch is a movable pointer to one of commits.
  - Define a single stream of commits
- The default branch name is `master`.
  - For each commit, the master branch moves forward automatically.
- HEAD is a pointer to a commit (and its representing branch) that the current working directory is in

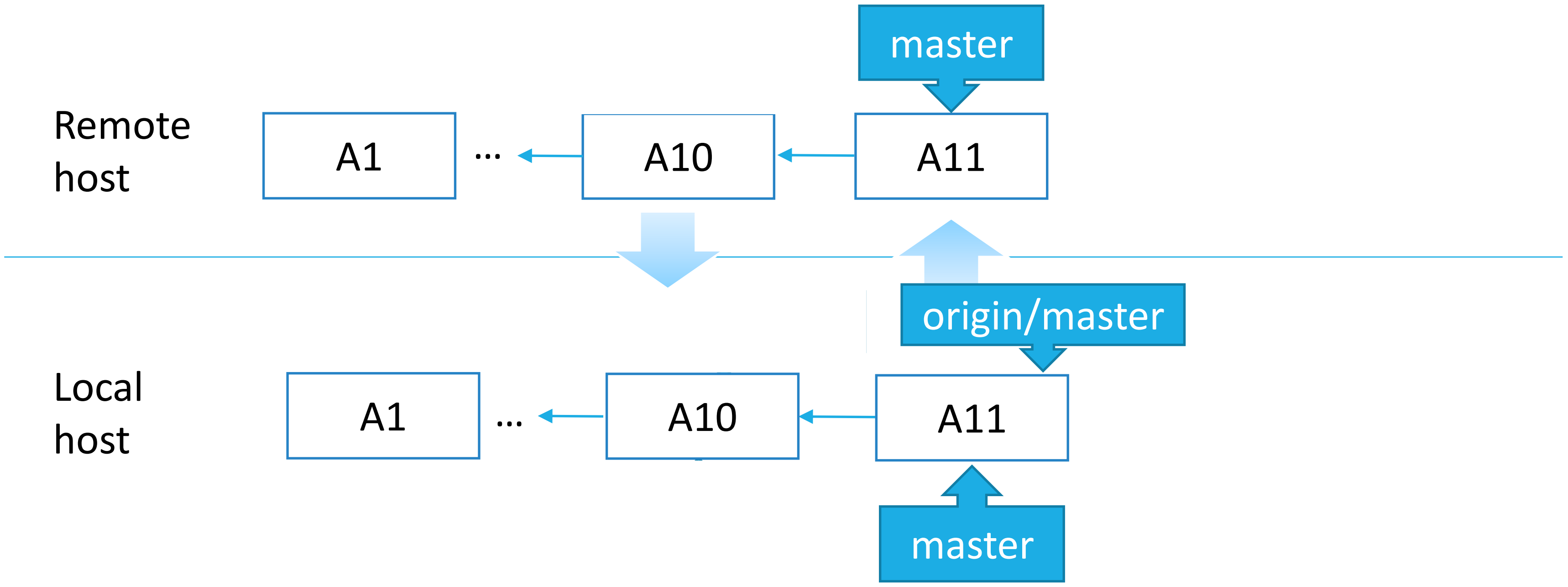


Cloning an Existing Repository

- Get a copy of an existing repository from a remote site
  - Receive not only the up-to-date snapshot but also a full copy of all data of the repository
  - Name the cloned repository as `origin` (by default)
- git clone

```
$ git clone https://github.com/libgit2/libgit2

$ ls
.git   libgit2

$ git remote
origin
```

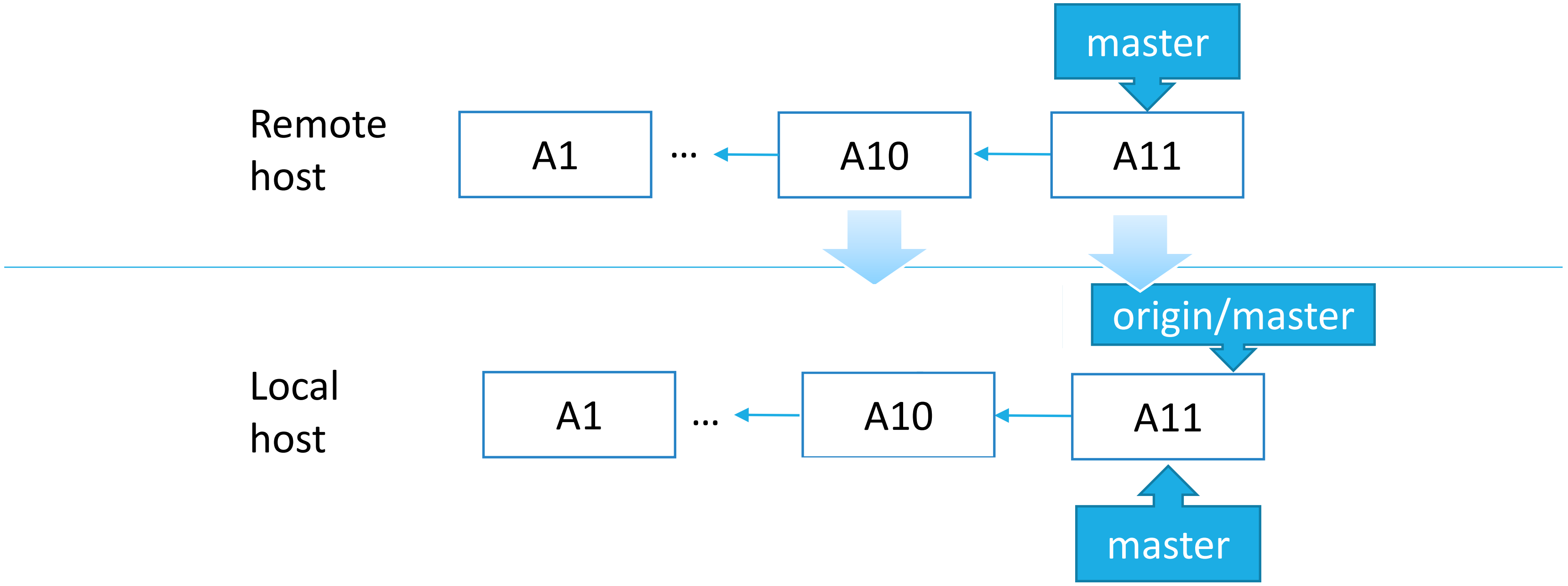
Pushing to Remote



- Push the master branch to the origin server
- git push

```
$ vim libgit.c
$ git commit
$ git push origin master
```

Fetching from a Remote Repository



- Update the recent changes of the remote repository
- git fetch origin

```
$ git fetch origin master
```

Merging

```
$ git merge origin/master
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit
the result.
$ vim index.html



$ git
$ git
```

```
…
<<<<<<< HEAD:index.html
<div id="footer">contact :
email.support@github.com</div>
=======
<div id="footer">
 please contact us at support@github.com
</div>
>>>>>>> origin/master:index.html
…
```

- **`git pull`** execuets fetch and merge at once

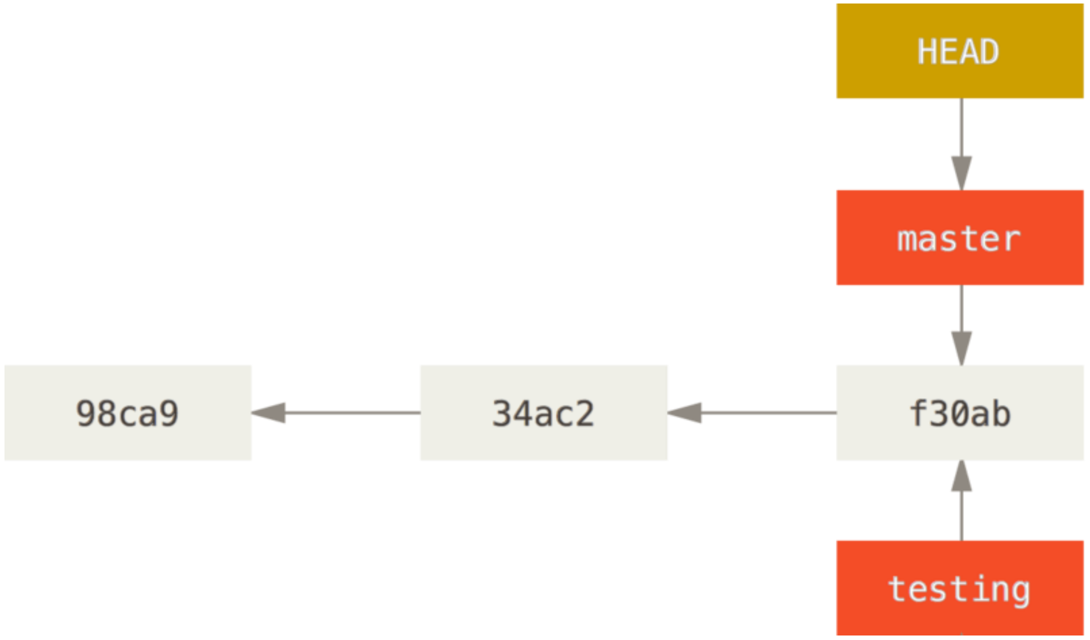
Git Commands

- Basic commands
  - no variation, no collaboration
- Commands to work with remote
  - collaboration
- Commands to manage branches
  - variations

Creating a New Branch

- Create a new pointer to the same commit the working directory is on.
- git branch

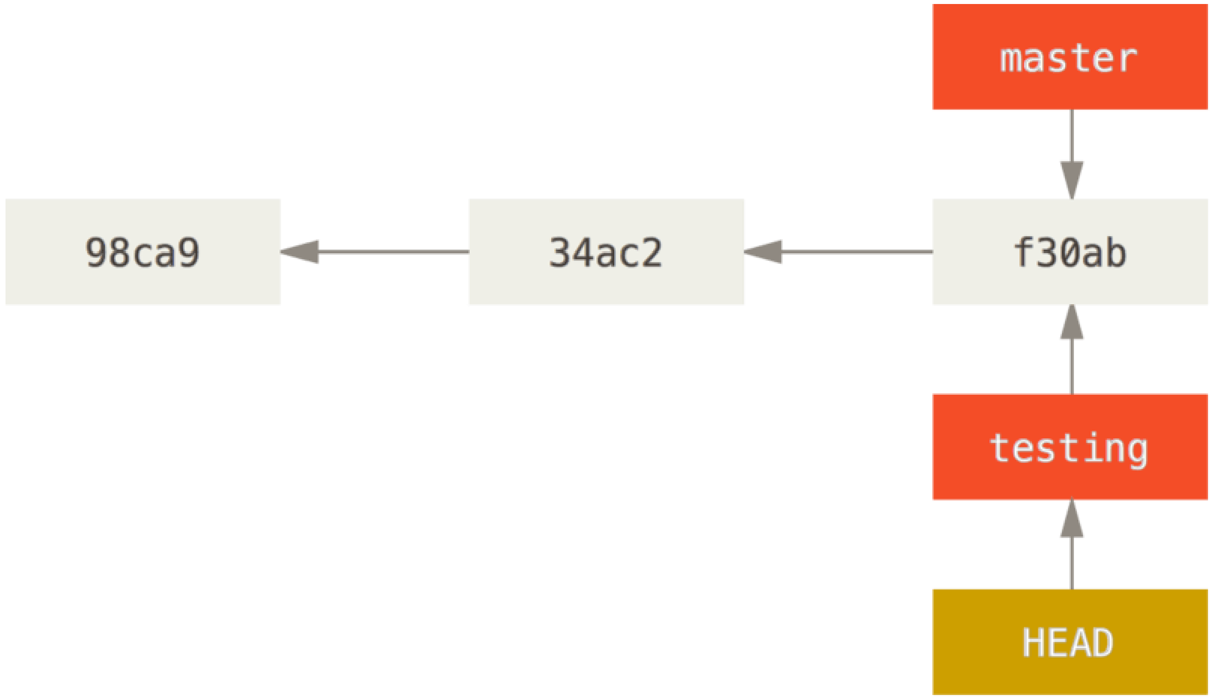
```
$ git branch testing
```



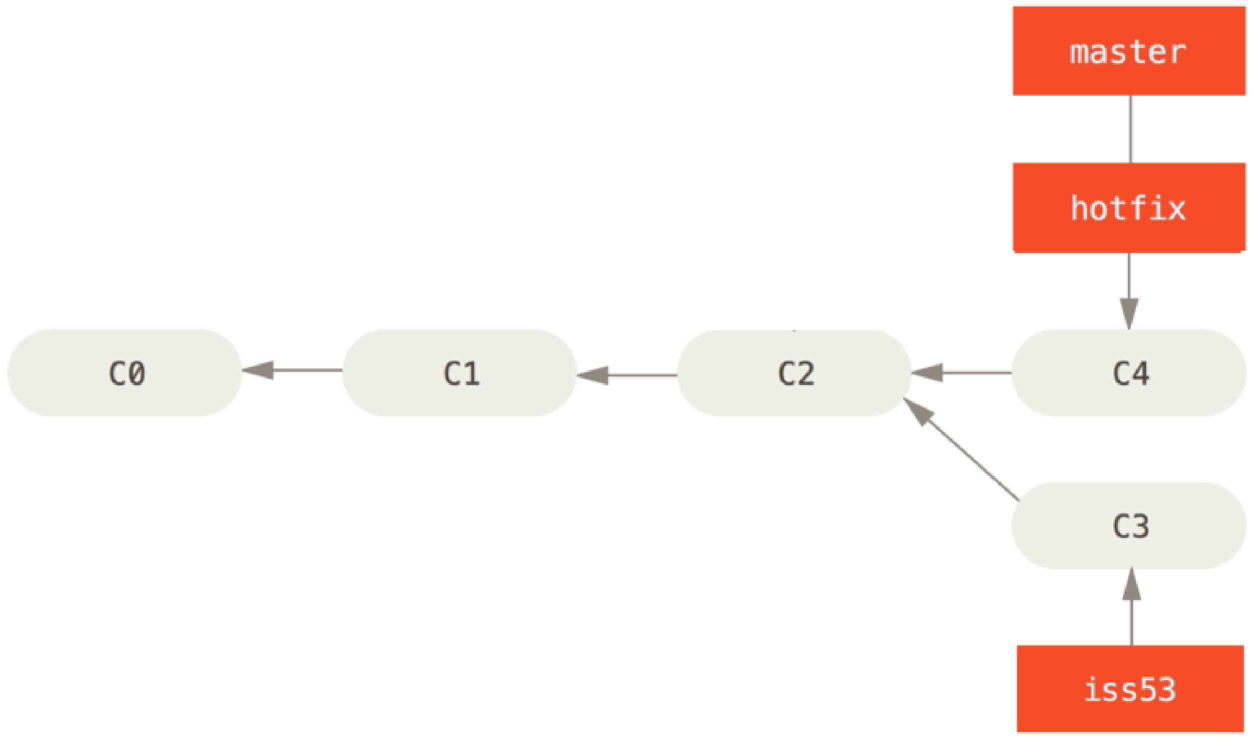
Switching Branches

- Switch to a branch
- git checkout

```
$ git checkout testing
```



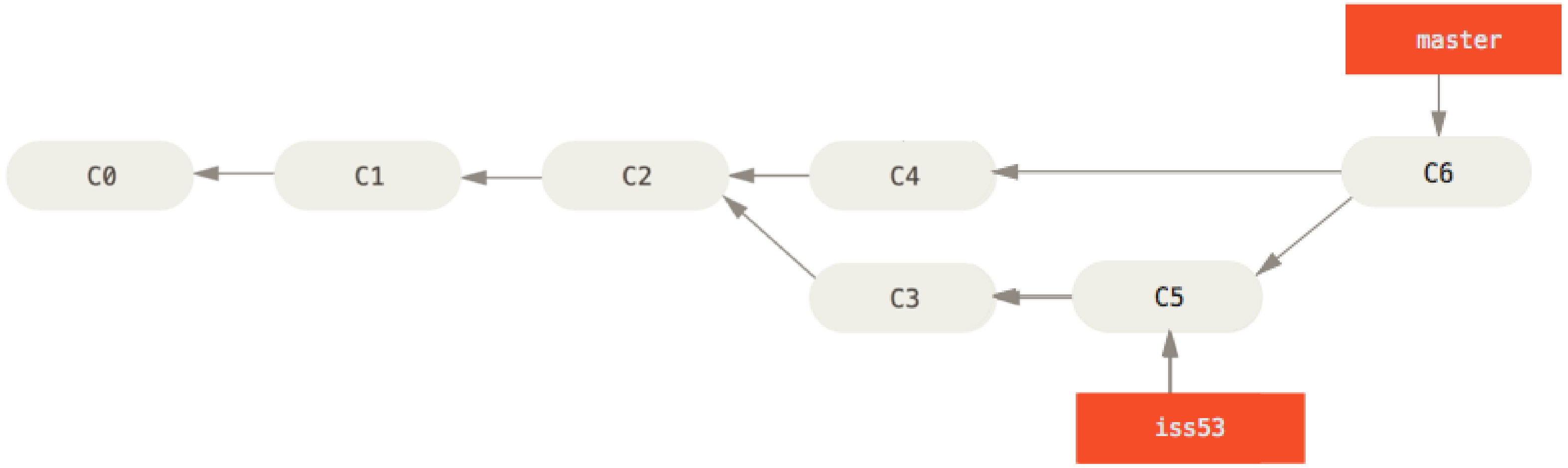
Merging with a Branch (1/3)



- Merge the HEAD branch with a specific branch
- git merge

```
$ git checkout master
$ git merge hotfix
```

Merging with a Branch (2/3)



```
$ git merge iss53
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit
the result.
```

Merging with a Branch (3/3)

```
$ git merge iss53
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit
the result.
$ vim index.html



$ gi
$ gi
```

```
…
<<<<<<< HEAD:index.html
<div id="footer">contact :
email.support@github.com</div>
=======
<div id="footer">
 please contact us at support@github.com
</div>
>>>>>>> iss53:index.html
…
```

Fork and Pull Request

- Fork is to clone a github repo at your github account
  - Need to clone a forked repo to work on codebase
  - Register the original repo as a remote to pull the original directly

```
git remote add upstream https://URL
```

- Pull request is to ask a github repository maintainer to merge the master with a given commit