Open-Source Software Practice

인공지능융합전공 여름 부트캠프

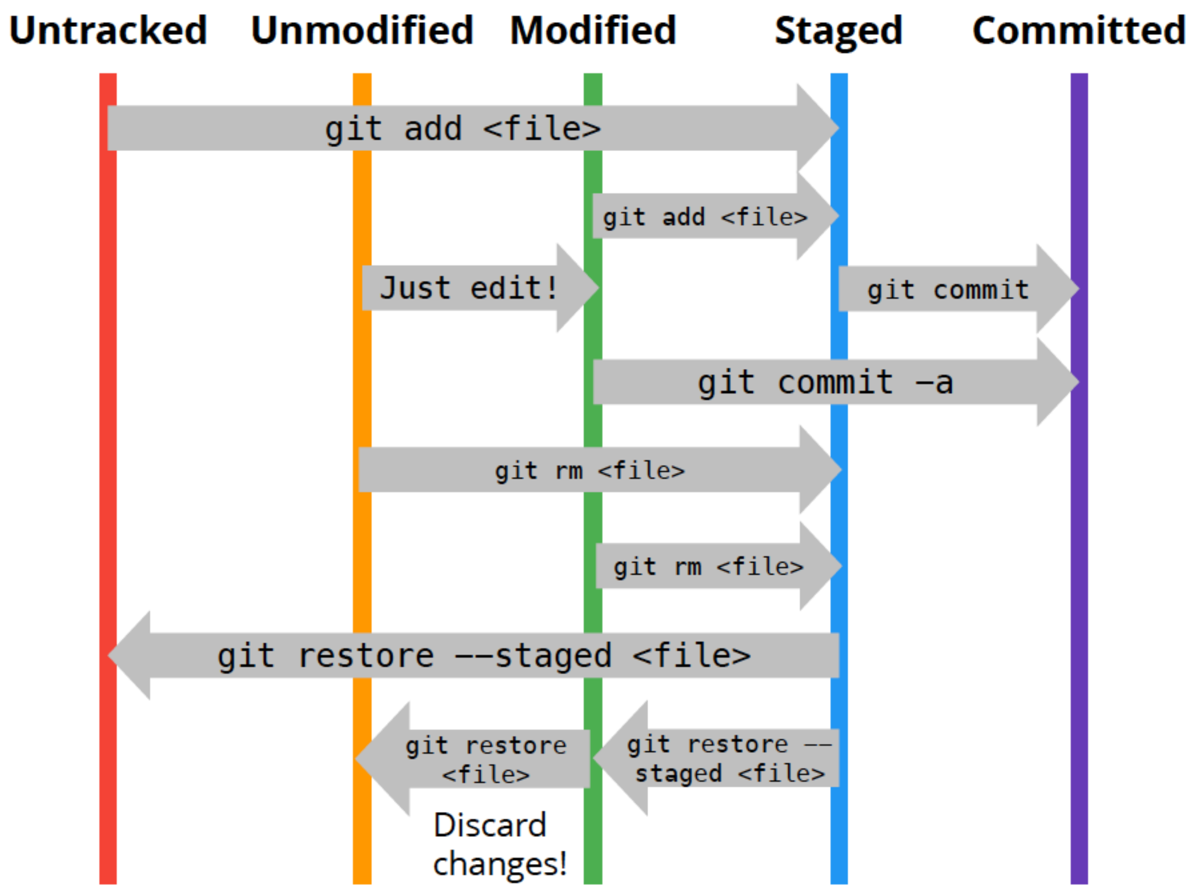
Jiwon Choi (최지원, jiwnchoi@skku.edu)
Interactive Data Computing Lab (IDCLab)

어제 복습

- **Branch**
  - Branch란?
  - Merging Branch
- **VSCode**
- **JavaScript Basics**

어제 복습

- **Git Basics**
  - git이란?
  - git과 GitHub의 차이
  - git 기본 명령어
- **HTML**
- **CSS**



Jiwon Choi, IDCLab
Sungkyunkwan University

어제 복습

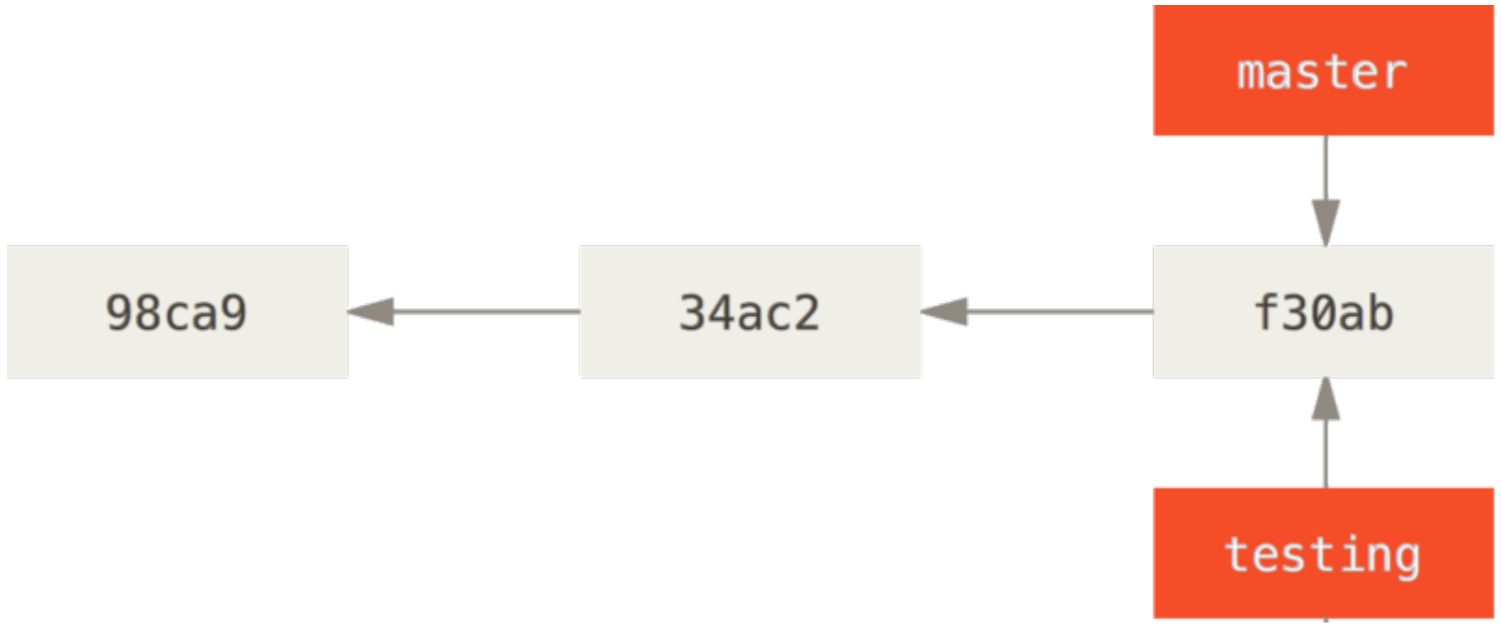
- Git 레포지토리 하나 만들기
- README.md 라는 파일을 만들기
- 커밋 한번 하기

Git Advanced

인공지능융합전공 여름 부트캠프

Jiwon Choi (최지원, jiwnchoi@skku.edu)
Interactive Data Computing Lab (IDCLab)

What is Branch?

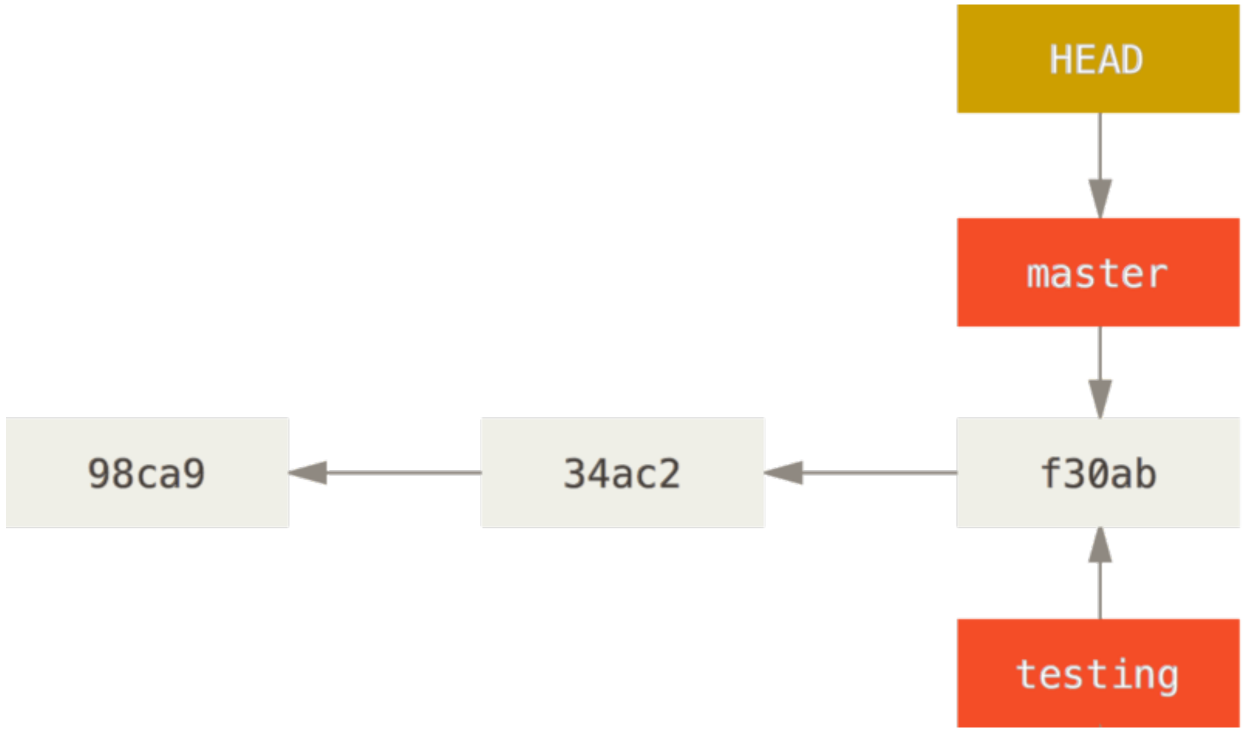


What is Branch?

- **Branching**
  - 가지치기
  - 이전 상태를 바탕으로 여러가지 상태를 가진 다른 작업을 수행하기 위해 새로운 브랜치를 만드는 것
- **Main Branch**
  - 기본 브랜치
- **Branching 예시**
  - 기능 단위, 사람 단위, 버그 픽스, ...

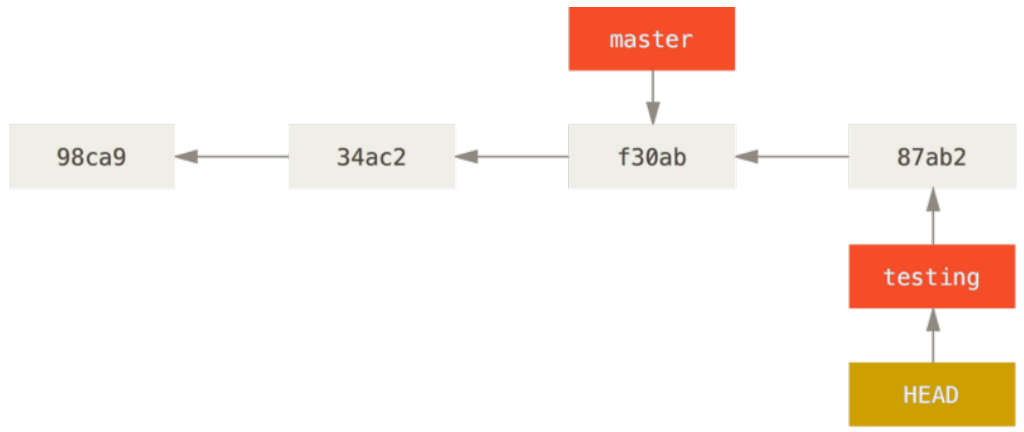
Creating Branch

- 브랜치 만들기
  - `git branch {name}`
- 브랜치 이동하기
  - `git checkout {name}`
- HEAD Pointer
  - 현재 작업중인 브랜치를 가리키는 포인터
  - Tag나 특정 Commit을 가리킬 수도 있음

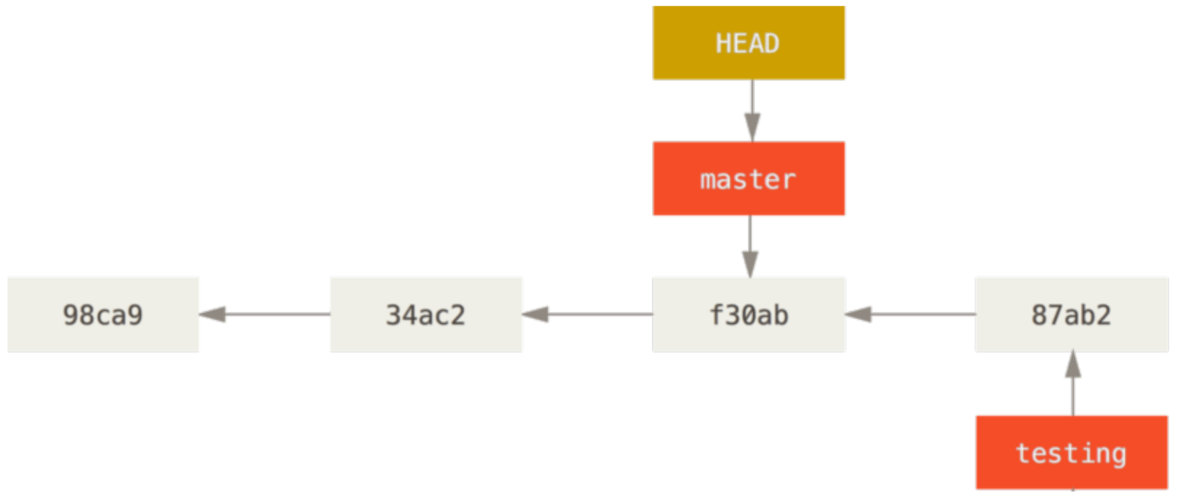


Creating Branch

- 새로운 커밋을 해보자

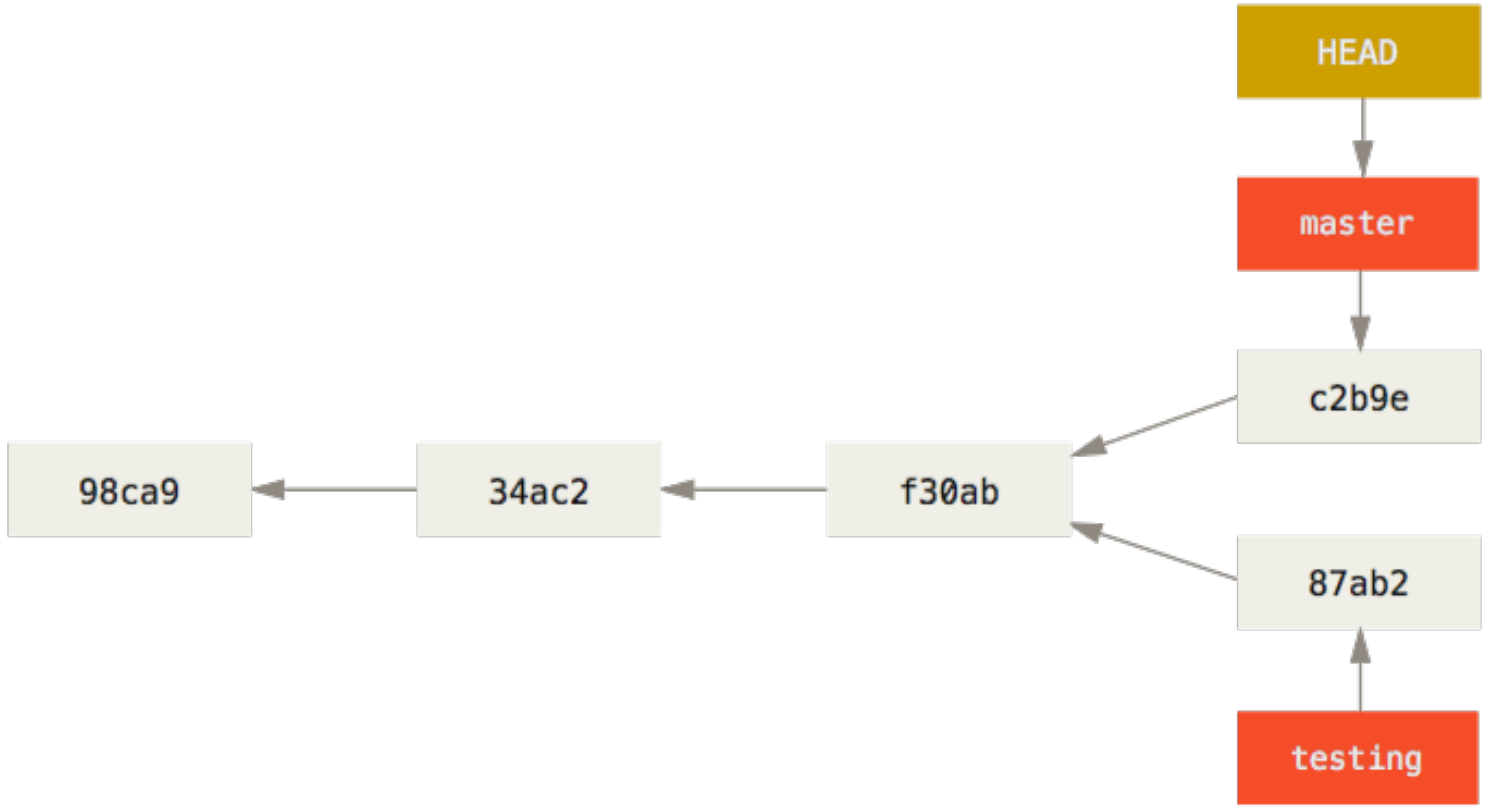


- 이전 브랜치로 돌아가보자



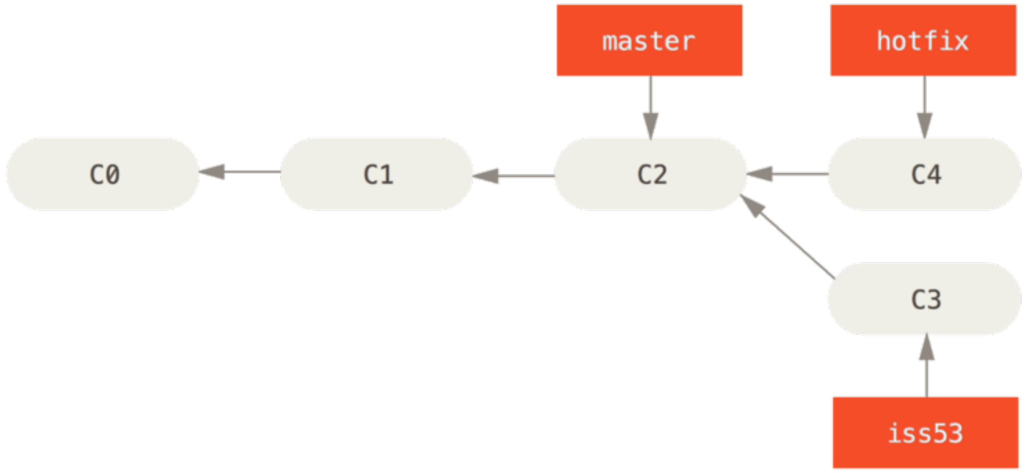
Diverging Branch

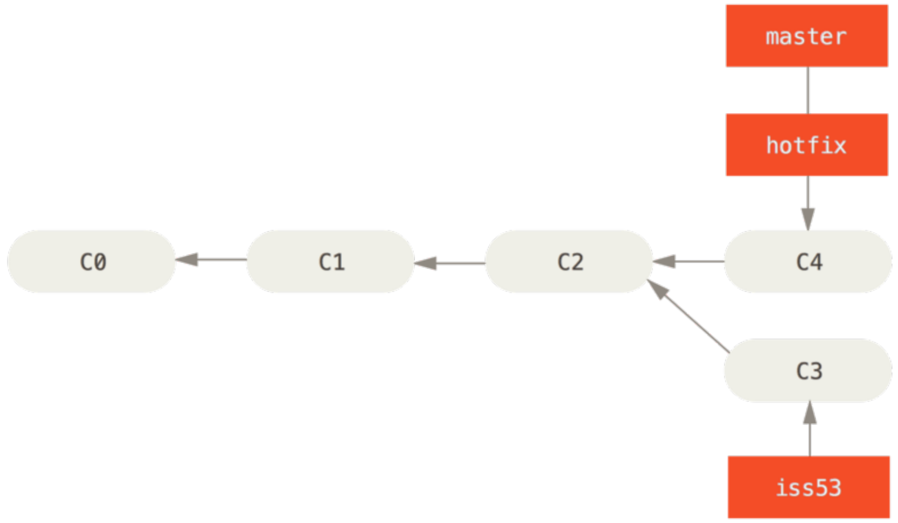
- ?



Merging Branch

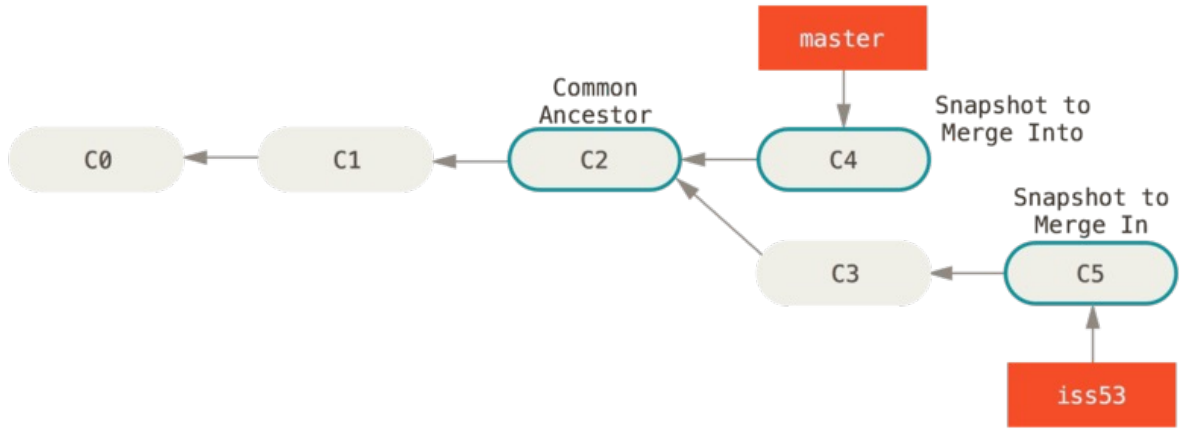
- **Fast-Forward**
  - `git checkout master`
  - `git merge hotfix`
  - `git branch –d hotfix`

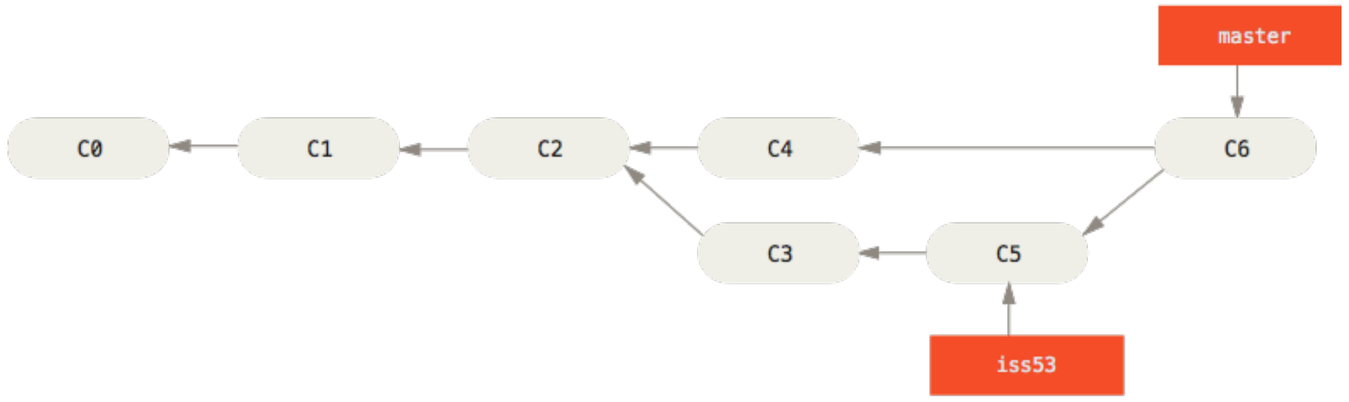




Merging Branch

- **3-way Merge**
  - `git merge iss53`
  - Merge Commit (C6)을 만들기

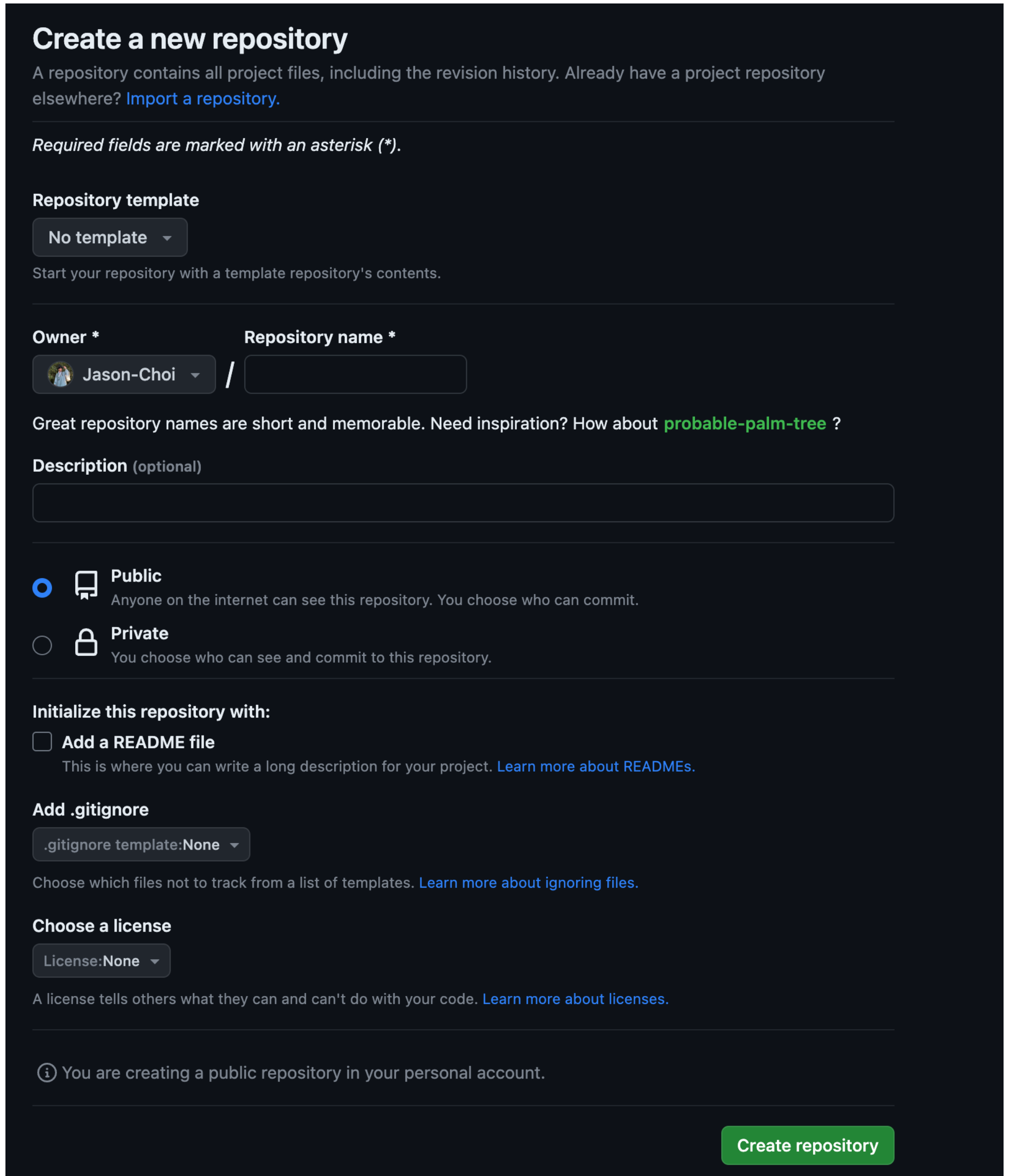




Merging Branch

- **Merge Conflict**
  - 같은 파일을 두 브랜치에서 동시에 수정하면? -> Merge Conflict 발생
- **Merge 취소**
  - git merge --abort
- **Merge Conflict를 해결하려면?**
  - 수동으로 까서 고치고 커밋

Working with Remote Repository


Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Repository template

No template

Start your repository with a template repository's contents.

Owner * Jason-Choi / **Repository name ***

Great repository names are short and memorable. Need inspiration? How about probable-palm-tree ?

Description (optional)

[✓ **Public**] Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:

☐ **Add a README file**
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

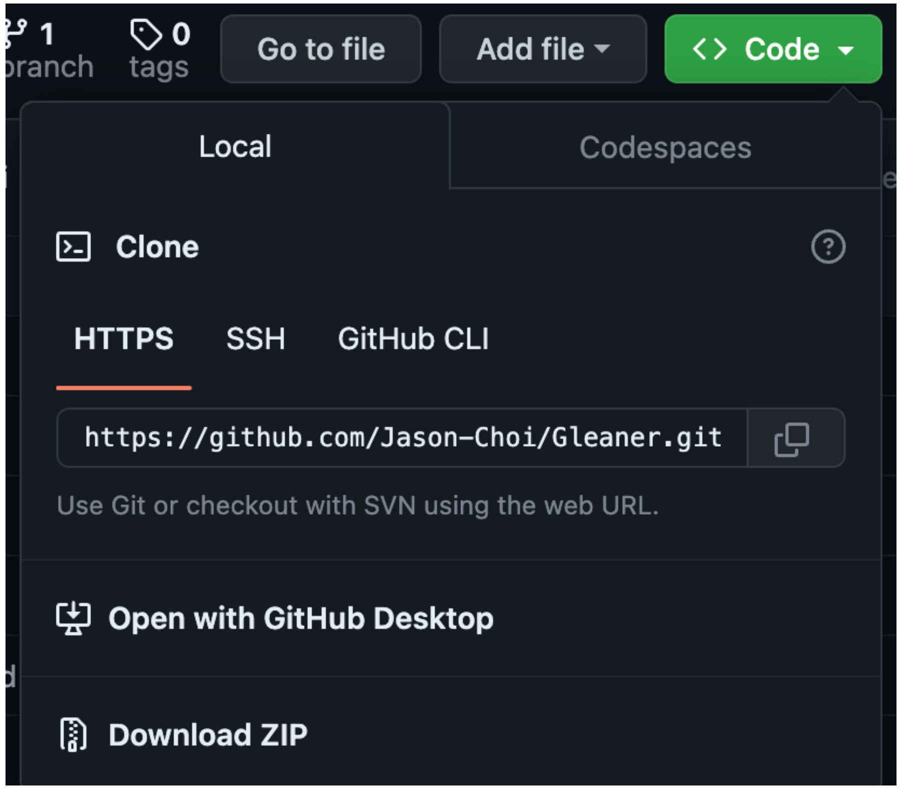
You are creating a public repository in your personal account.

Create repository

Jiwon Choi, IDCLab
Sungkyunkwan University

Working with Remote Repository

- Cloning Repository



```
https://github.com/Jason-Choi/Gleaner.git
```

Use Git or checkout with SVN using the web URL.

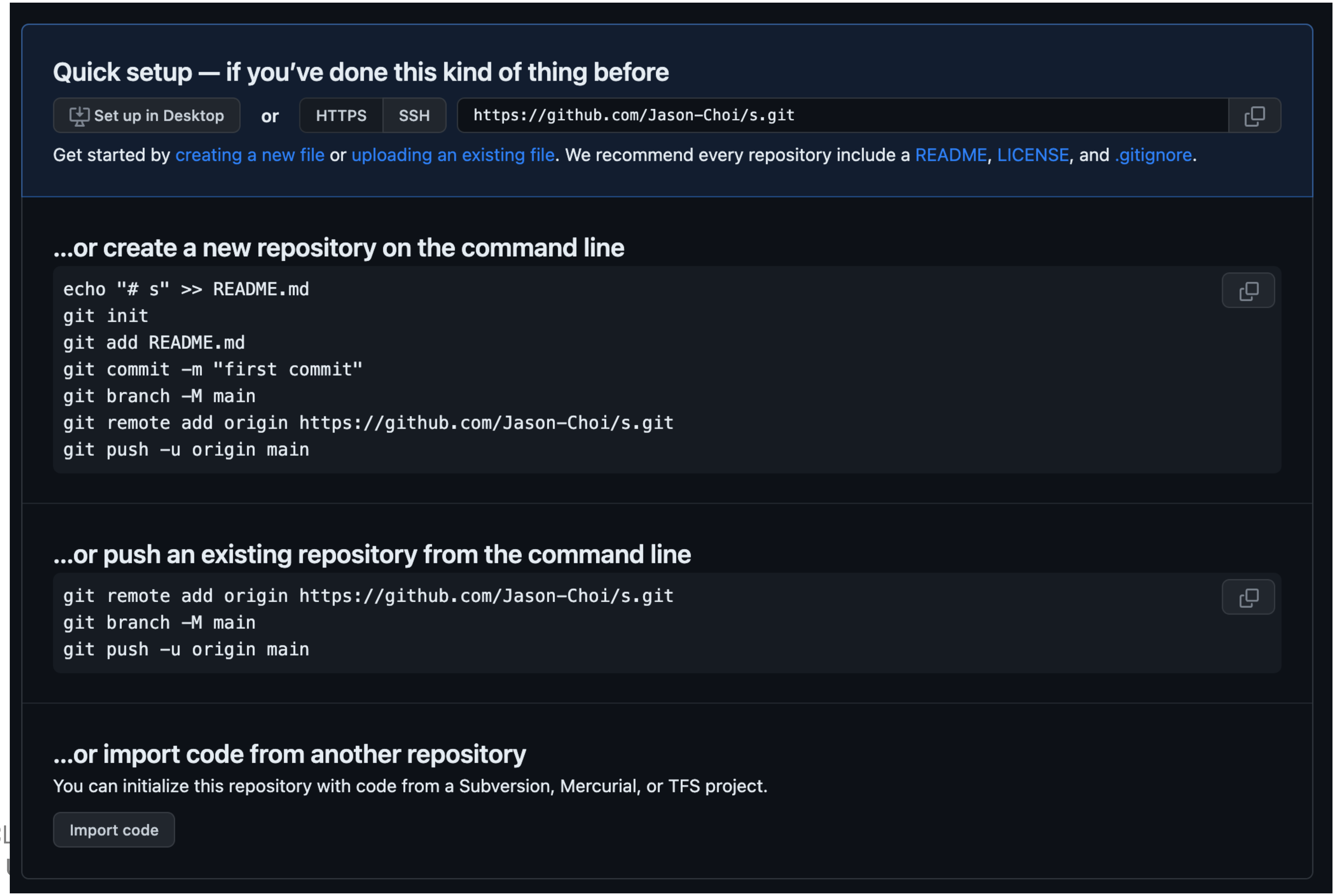
Sign-in to GitHub

- https://docs.github.com/ko/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens
- VSCode

Remote Repository

- **Showing the Remotes**
  - git remote
- **Initializing Git Repository**
  - git clone -> Automatically initialize repository!
  - git init ..

Remote Repository


Quick setup — if you've done this kind of thing before

```
https://github.com/Jason-Choi/s.git
```

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# s" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Jason-Choi/s.git
git push -u origin main
```

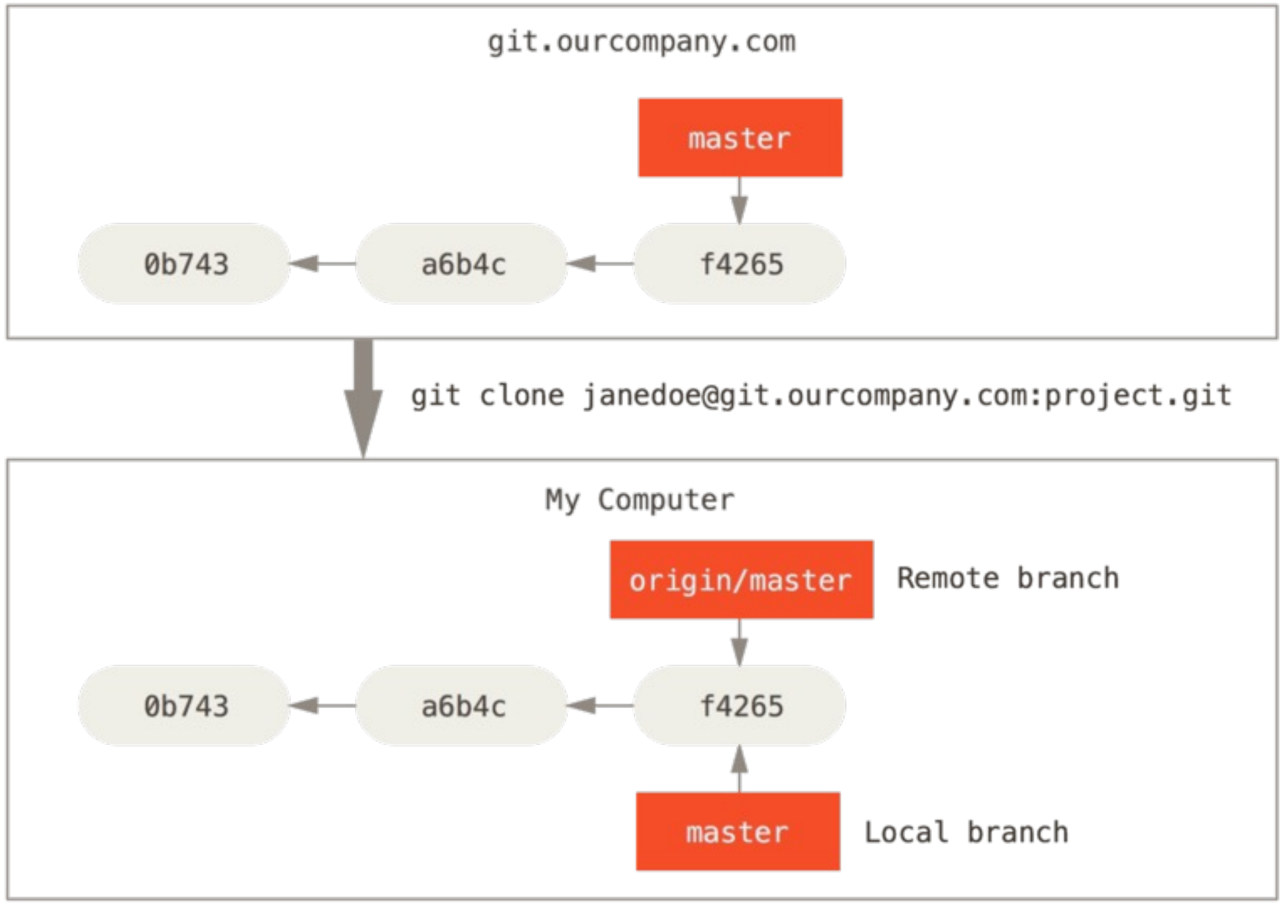
...or push an existing repository from the command line

```
git remote add origin https://github.com/Jason-Choi/s.git
git branch -M main
git push -u origin main
```

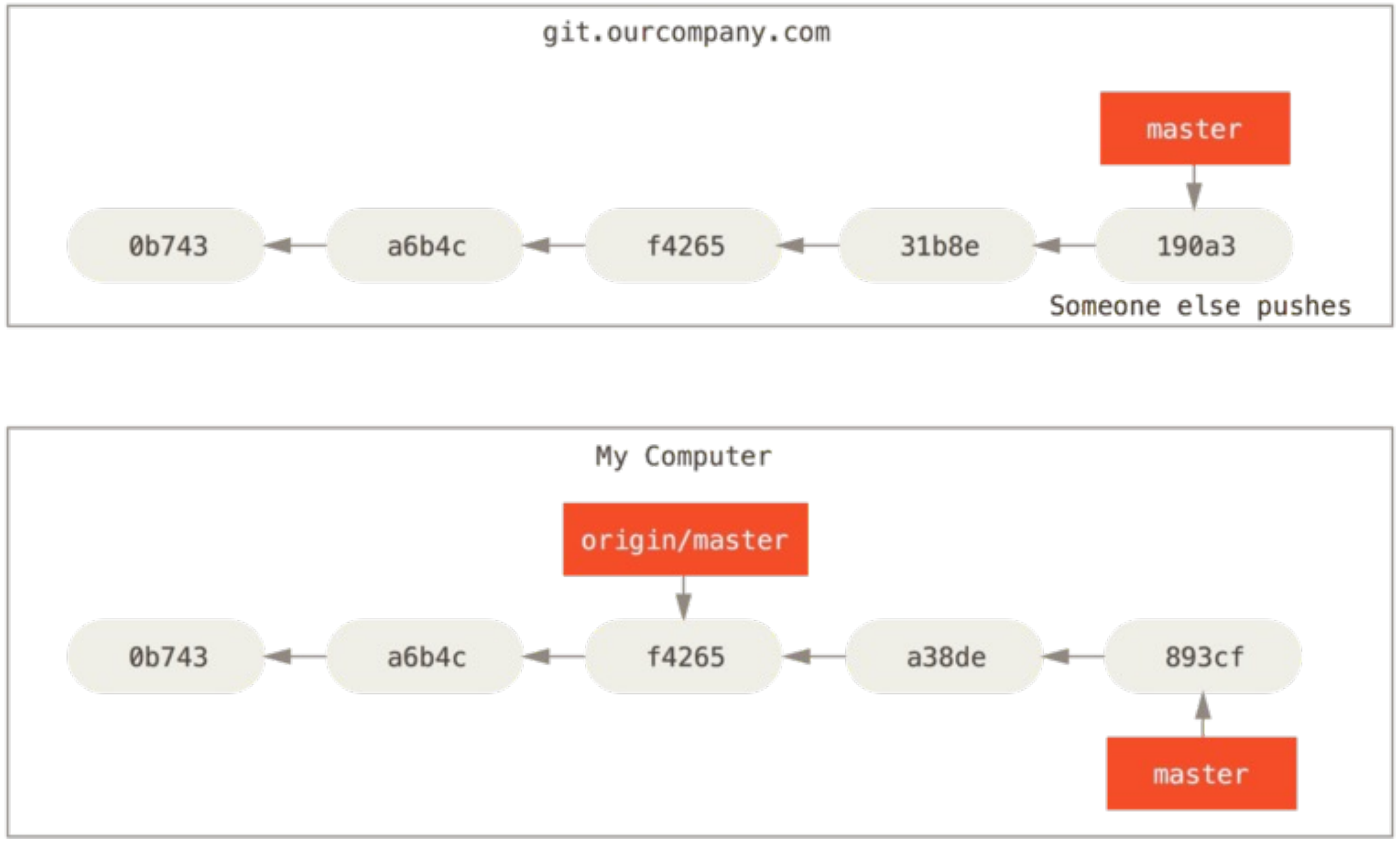
...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Remote Branch

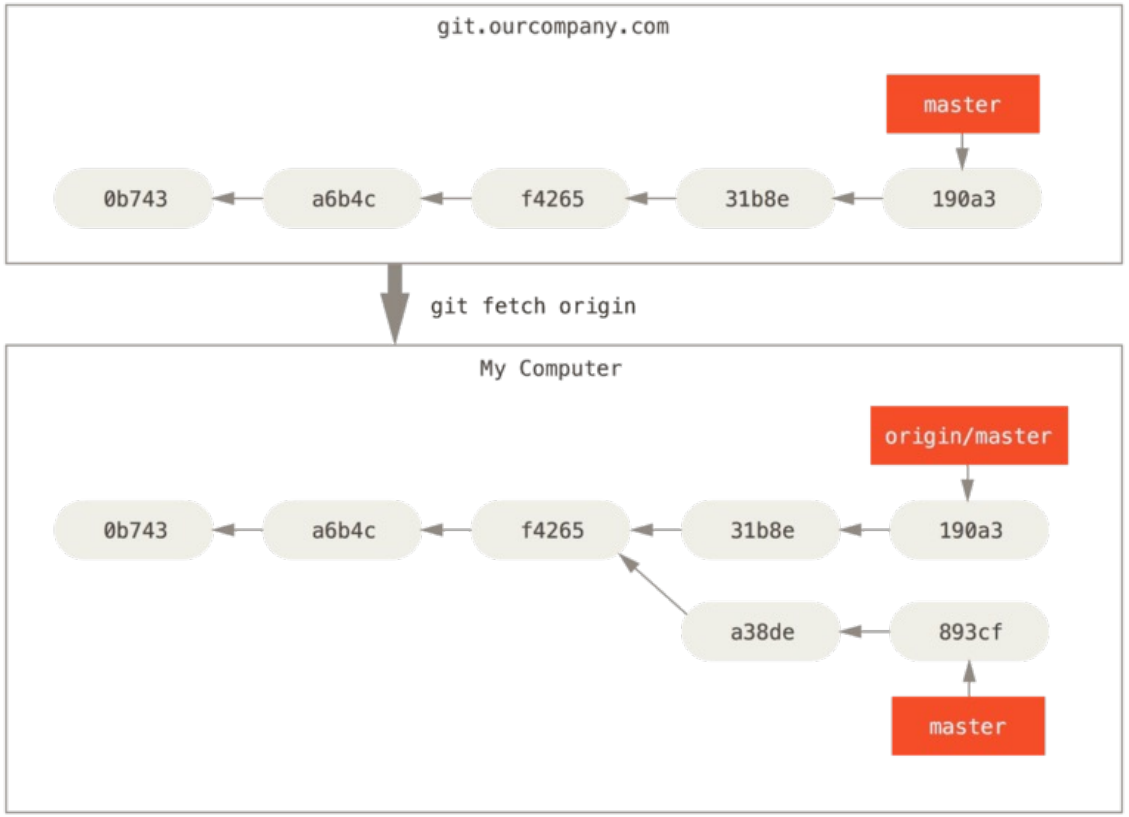


Remote Branch



Remote Branch

- main과 origin/main은 다르다
  - `git fetch`



Jiwon Choi, IDCLab
Sungkyunkwan University

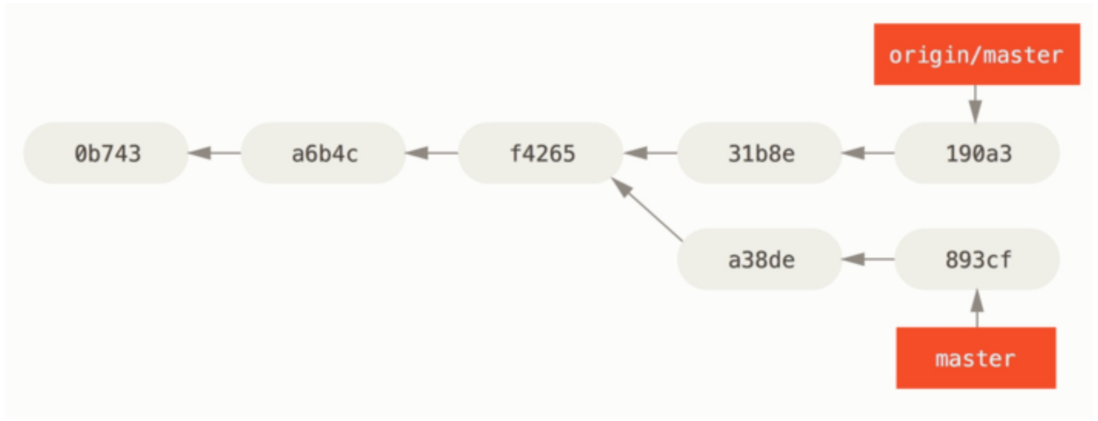
Push to / Pull from Remote Branch

- **Local Branch를 Remote Branch로 업로드하기**
    - `git push (origin)`
    - `git push origin`
    - `git push origin main`

- **Remote Branch를 Local Branch로 다운로드하기**
    - `git pull`
    - `git fetch + merge`

Merge Conflicts with Remote Branch

- Merge conflicts between local and remote



- Resolve conflicts
  - `git add .`
  - `git commit`
  - `git push`

Using Visual Studio Code

인공지능융합전공 여름 부트캠프

Jiwon Choi (최지원, jiwnchoi@skku.edu)
Interactive Data Computing Lab (IDCLab)

Why VSCode?

- **Visual Studio Code**
  - Free and Open-Source Code Editor (Not IDE!)
  - Supported by MS and GitHub
  - Web Based (by Electron, we will learn later)
  - Fast and Light-weight
  - Lots of Extension and Ecosystem
  - https://vscode.dev/

Must Have Plug-ins

- **Live Server**
- **Remote-SSH, Container**
- **GitHub Copilot**
  - Free if you are student!
  - https://github.com/features/copilot/
  - https://education.github.com/pack
- **GitHub Copilot Chat**
  - Only in VSCode-Insiders
- **EditorConfig**
- **Language Specific Extensions**

Make Your Own VSCode!

Node and JavaScript

인공지능융합전공 여름 부트캠프

Jiwon Choi (최지원, jiwnchoi@skku.edu)
Interactive Data Computing Lab (IDCLab)

Goal

- Install Node.js
- Learn JavaScript Basics

Textbook

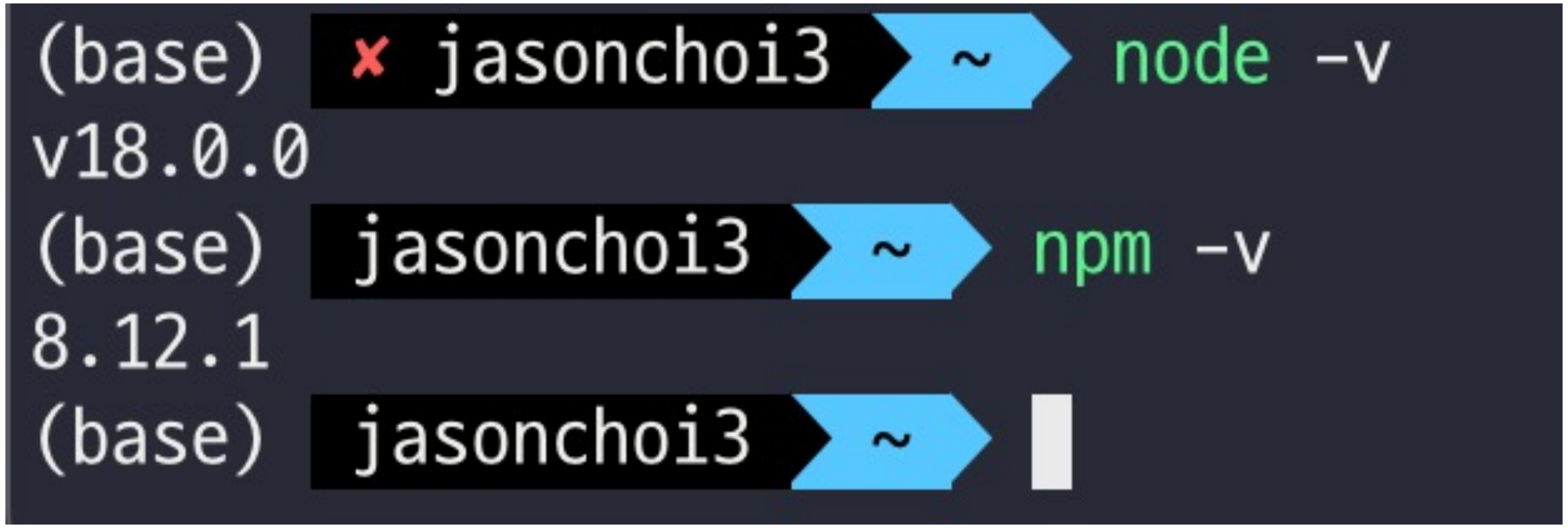
- https://javascript.info
- https://ko.javascript.info

Install Node.js

- **Install with Package (.exe, .pkg. deb ...)**
- https://nodejs.org/ko/download/

- **Install with Package Manager**
- Homebrew (Mac)
  - `brew install node`
- apt (Ubuntu, Debian)
  - `curl -fsSL https://deb.nodesource.com/setup_lts.x`
  - `sudo -E bash - sudo apt-get install -y nodejs`

Installation Check



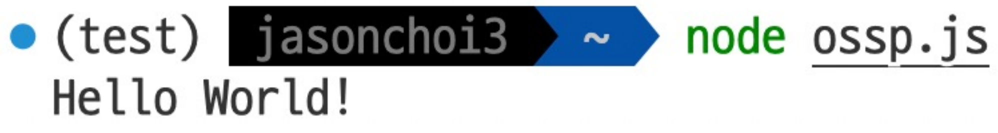
```
(base) ✘ jasonchoi3 ~ node -v
v18.0.0
(base) jasonchoi3 ~ npm -v
8.12.1
(base) jasonchoi3 ~
```

Jiwon Choi, IDCLab
Sungkyunkwan University

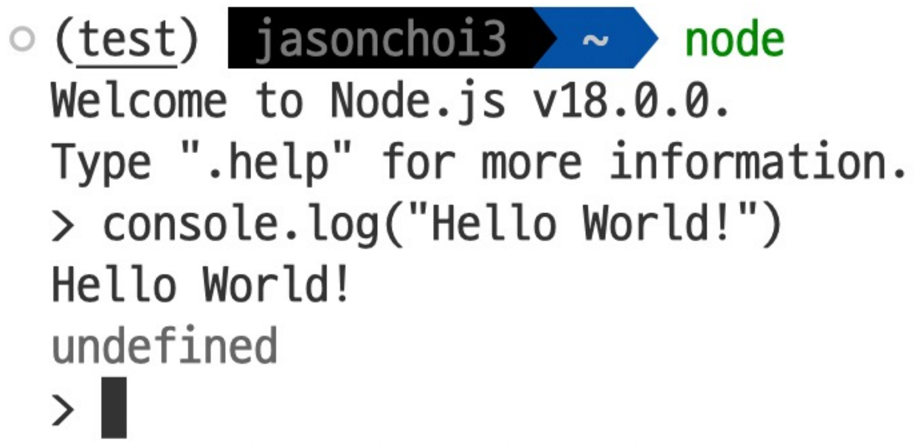
Execute JavaScript Commands & Files with Node.js

- ossp.js

```
console.log("Hello World!")
```



```
● (test)  jasonchoi3  ~  node ossp.js
Hello World!
```



```
○ (test)  jasonchoi3  ~  node
Welcome to Node.js v18.0.0.
Type ".help" for more information.
> console.log("Hello World!")
Hello World!
undefined
> 
```

Variables

- **Declare variables with** `let, const`

```
let year = 1398;
let name = "SKKU";

const department = "Computer Science and Engineering";
const ids = [20220001, 20220002, 20220003, 20220004, 20220005];
```

- `let` **vs** `const` **?**

- `var` **?**

Primitive Types

- **Number, String, Boolean, undefined, null, and more..**

- **Primitive types are immutable!**

- Immutable: cannot be altered === cannot change

```
let string = "Sungkyunkwan University"
string[0] = "B" // Error
```

Operators

- **Arithmetic:** `+, -, *, /, %, ++, --`
- **Assignment:** `=, +=, -=, *=, /=, %=`
- **Comparison:** `===`, `!==`, `>, <, >=, <=, ?`
- **Logical:** `&&, ||, !`
- **Bitwise:** `&, |, ~, ^, <<, >>`
- **Type:** `typeof`

- **Almost Same With C**

Type Conversion

- **How to Convert Type Explicitly?**

```
typeof "123" // "string"
typeof Number("123") // "number"

typeof 123 // "number"
typeof String(123) // "string"

typeof (123).toString() // "string"
```

- **Apply operators on variables of the <u>same type</u>**

if statement

- if statement of C and JavaScript are same!

```
const a = 10;
const b = 20;

if (a > b){
    console.log("a is greater than b");
}
else if (a < b){
    console.log("a is less than b");
}
else{
    console.log("a is equal to b");
}
```

```
const university = "Sungkyunkwan University";

if (university === "Sungkyunkwan University")
{
    console.log("Welcome to SKKU");
}
else {
    console.log("You are not from SKKU");
}
```

for statement

- **JavaScript has three for statement!**
- for: Default for statement (C-like)
- for..in: Iterates with indices (or key of object)
- for..of: Iterates with elements

```
const arr = ["Open", "Source", "Software"]

for (let i=0; i<3; i++) console.log(i, arr[i]) // Not int i=0;

for (let idx in arr) console.log(idx) // 1 2 3

for (let val of arr) console.log(val) // Open Source Software
```

`while` statement

- `while` statement of C and JavaScript are same!
- `do..while` also same.

```
let i = 0;
while (i < 3) {
    console.log(i);
    i++;
}
```

```
let i = 0;
do {
    console.log(i);
    i++;
} while (i < 3);
```

Array Data Structure

- **Declare Array**

```
const a = ["Open", "Source", "Software", 1398, ["Linux", "Windows", "MacOS"]];
```

- **Array Length**

```
a.length; // 5
```

- **Typeof Array**

```
typeof a // object
Array.isArray(a) // true
```

Object Data Structure

```
const IDCLab = {
    director: {
        name: "Jaemin Jo"
    },
    students: [
        { name: "John", id: 111 },
        { name: "Zoey", id: 112 },
        { name: "Chen", id: 113, graduated: true },
    ]
}

console.log(IDCLab.director)
console.log(IDCLab.director.name)

console.log(IDCLab.students)
console.log(IDCLab.students[0].name)
```

Object Data Structure

- Using Complex Data Structures

```
for (const student of IDCLab.students) {
    if (student.graduated) console.log(student.name + " graduated")
    else console.log(student.name + " is studying")
}
```

- Everything in JavaScript Except Primitive Type is Object!

```
typeof [1,2,3] // "object"
typeof {a:1, b:2} // "object"
typeof function(){} // "function"...?
```

Object Quiz

- Object with const?

```
const IDCLab = {
    director : "Jaemin Jo"
}

IDCLab.director = "Jiwon Choi" // ?!?!?!
```

- Key with Phrase?

```
const IDCLab = {
    director name : "Jaemin Jo", // ???
}
```

Function

- **Ways to Declare Functions**

```
function sum(a, b) {
    return a + b;
}

const sum = function (a, b) { return a, b };

const sum = (a, b) => { return a + b };

const sum = (a, b) => a + b;
```

Array Methods

- **Destructive Methods VS Non-Destructive Methods**

```
const arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

console.log(arr.push(11));

console.log(arr.pop());

console.log(arr.includes(2));

console.log(arr.slice(2, 5));

console.log(arr.splice(2, 5));

console.log(arr.concat([11, 12, 13]));
```

```
console.log(arr.join(' '));

console.log(arr.reverse());

console.log(arr.shift());

console.log(arr.unshift(1));

console.log(arr.sort());

console.log(arr.toString());
```

Array Methods

- **Collective Operation Methods**

```
const arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
console.log(arr.map((item) => item * 2));
console.log(arr.filter((item) => item % 2 === 0));
console.log(arr.forEach((item) => {console.log(item * 2)}));
console.log(arr.every((item) => item > 0));
console.log(arr.some((item) => item > 10));
```

Array Methods

- **Method Chaining**

```
const arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
arr.map(i => i * 2)
    .filter(i => i % 3 === 0)
    .forEach(i => console.log(i * i))
```

Summary Quiz

Write function that multiplies the number property.
(* Hint: Use for..in loop or Object.entries())

```
let menu = {
    width: 200,
    height: 300,
    title: "My menu"
};
```

→

```
multiplyNumeric(menu, 3);
```

→

```
menu = {
    width: 600,
    height: 900,
    title: "My menu"
};
```

* `multiplyNumeric` returns nothing. Just modify menu object.

Homework

- https://www.codecademy.com/learn/introduction-to-javascript