



# Using Github

---

# What Github Does

---

- Online project hosting site.
- "Remote" git repository with access control
- Issue Tracking
- Documentation and web pages ([github.io](https://github.io))
- Integrates with other services
  - Continuous Integration, e.g. CircleCI

# What to do

---

## 1. Create a Github Account

- Put your **REAL NAME** in your profile
- Add a **PHOTO** that **clearly** shows **your face**
- Include a public **Email**. Prefer KU-Gmail
- Write a **short profile** about yourself

2. Sign-up form: **`https://goo.gl/cwrBbW`**

3. Receive an e-mail invitation to join OOP2018

4. Click to join Github Organization.

# Github Profile

Students in last year's OOP course.

1. Real name

2. Photo

3. Email

4. Description of you



**Jirayu  
Laungwilawan**  
Jirayul

Faculty of Engineering , Major -  
Software and Knowledge  
Engineering.

Follow

Block or report user

📍 Thailand

✉ [jirayu.l@ku.th](mailto:jirayu.l@ku.th)

🔗 <https://github.com/Jirayul>



**Kongpon  
Charanwattanakit**  
kykungz

Software Developer, Undergraduate  
Software and Knowledge  
Engineering Student

Follow

Block or report user

👤 Kasetsart University

📍 Bangkok, Thailand

✉ [jackykongpon@gmail.com](mailto:jackykongpon@gmail.com)

🔗 <https://kykungz.github.io/>



# How to Use Github

---

## 2 Situations + 1 Special Case

Case 1: *You already have project code on your local computer. You want to copy to Github.*

Case 2: *Project already exists on Github. You want to copy it to your computer.*

Special Case:

Case 3: *A new project (no files yet).*

# Case 1: Starting from Local Project

*You already have a project on your computer*

1. Create a **local** "git" repository.

```
cmd> git init
```

```
cmd> git add .gitignore src
```


```
cmd> git commit -m "initial code checkin"
```

2. Create an **empty** repo on Github.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

 fatalaijon ▾

Repository name

/ demo ✓

Great repository names are short and memorable. Need inspiration? How about **symmetrical**

Description (optional)

Demonstration project

# Case 1: adding Github as remote

3. Copy the URL of new Github repository (https or ssh).

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

or



We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

4. In your local project directory, add Github as a remote repository named "origin":

```
cmd> git remote add origin  
      https://github.com/fatalaijon/demo.git
```

5. Copy the local repository to Github

```
cmd> git push -u origin master
```

You only need "-u origin master" the first time you push to Github. Next time, just type "git push".



## Case 2: Starting from Github

*A project already exists on Github.*

*You want to "clone" it your local computer & do work.*

1. Get the Github project URL

`https://github.com/fatalaijon/demo.git`

or: go to project on Github and click on



Clone or download ▾

then copy the URL.

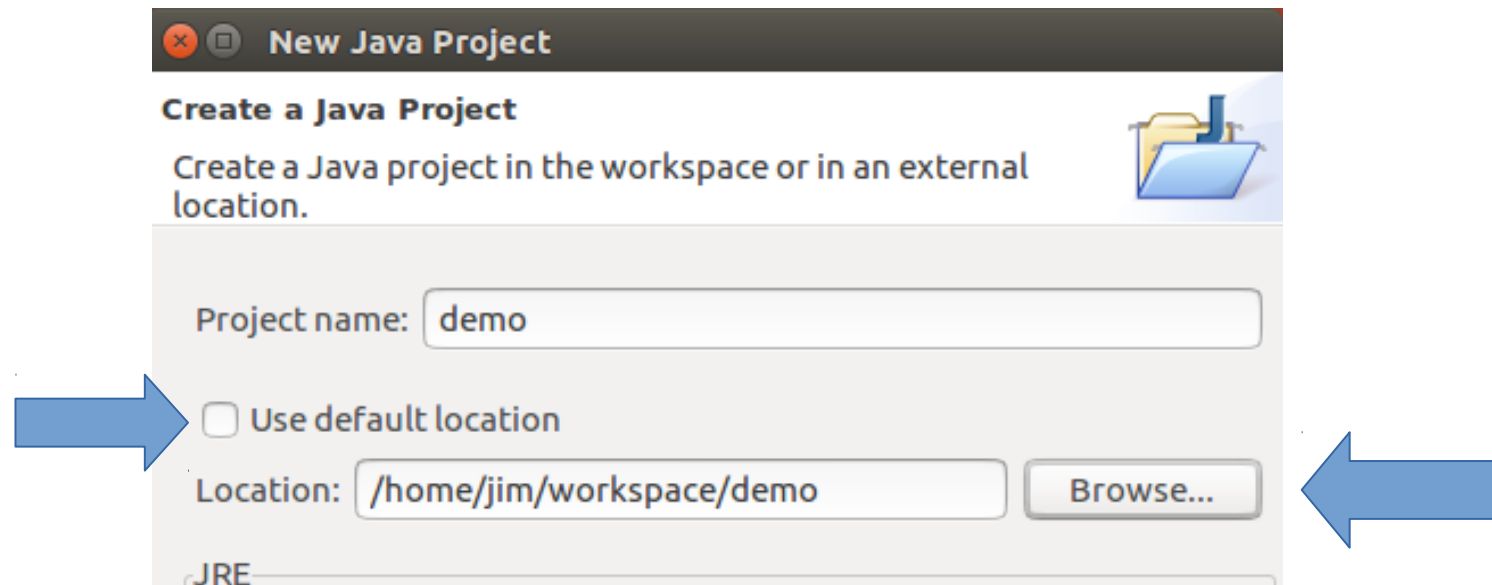
2. In your workspace, type:

`cmd> git clone https://github.com/...`

**NOTE:** "git clone" creates a new directory (named demo) inside your current directory. If the directory already exists, clone **won't work**.

## Case 2: Create an IDE project

3. Start your IDE and create a new project using the code in the directory you just cloned.



*That's it!*

Github is automatically the remote "origin".  
Just `git push` your committed work to github.

# *You can use a different project name*

The name of your **local directory** (cloned from Github) can be **different** from the Github repository name.

1) Rename the directory yourself.

= or =

2) Specify directory name when you "clone":

# Clone "**demo**" into local directory "**mydemo**"

```
cmd> git clone https://github.com/fatalai  
jon/demo.git    mydemo
```

# Comparison of 2 Cases

---

# Git Workflow

*After you have a repository + Github, what do you do?*

*Git workflow for an **individual** project:*

1) Check status of your working copy:

```
cmd> git status
```

2) Commit changes or update your working copy.

```
"git commit ..."
```

3) Do some work:

Code, test. Code, test. ... Review.

Now what?

# Git Workflow (cont'd)

## 4) Check status again:

```
cmd> git status
```

```
Changes not staged for commit:
```

```
    modified:   src/Problem2.java
```

```
Untracked files:
```

```
    src/Problem3.java
```

## 5) **Add** and **commit** your work to the local repository

```
cmd> git add src/Problem2.java src/Problem3.java
```

```
cmd> git commit -m "Solved problems 2 and 3"
```

```
[master 29abae0] Solved problem 2 and 3
```

```
2 files changed, 44 insertions(+), 5 deletions
```

# Git Workflow (with Github)

## 6) Push the changes to Github

```
cmd> git push
```

```
Compressing objects: 100% (12/12), done.
```

```
Writing objects: 100% (12/12), 3.60 KiB,  
done.
```

```
Total 12 (delta 9), reused 0 (delta 0)
```

```
remote: Resolving deltas: 100% (9/9), ...
```

```
To https://github.com/fatailaijon/demo.git
```

```
468abdf..29abae0  master -> master
```

That's it!

Repeat the cycle (1 - 6) as you work.

# Git Workflow for Team Projects

On a **team project**, other people will be committing files to the **same** Github repository.

For team projects, you should update your local repository from Github before trying to "push" your work to Github.

If Github updates your local repository, then you should merge changes into your working copy and test again, before trying to push to Github.

(illustration in class)



# Assignment

---

<https://cpske.github.io/programming1/git/git-assignment>