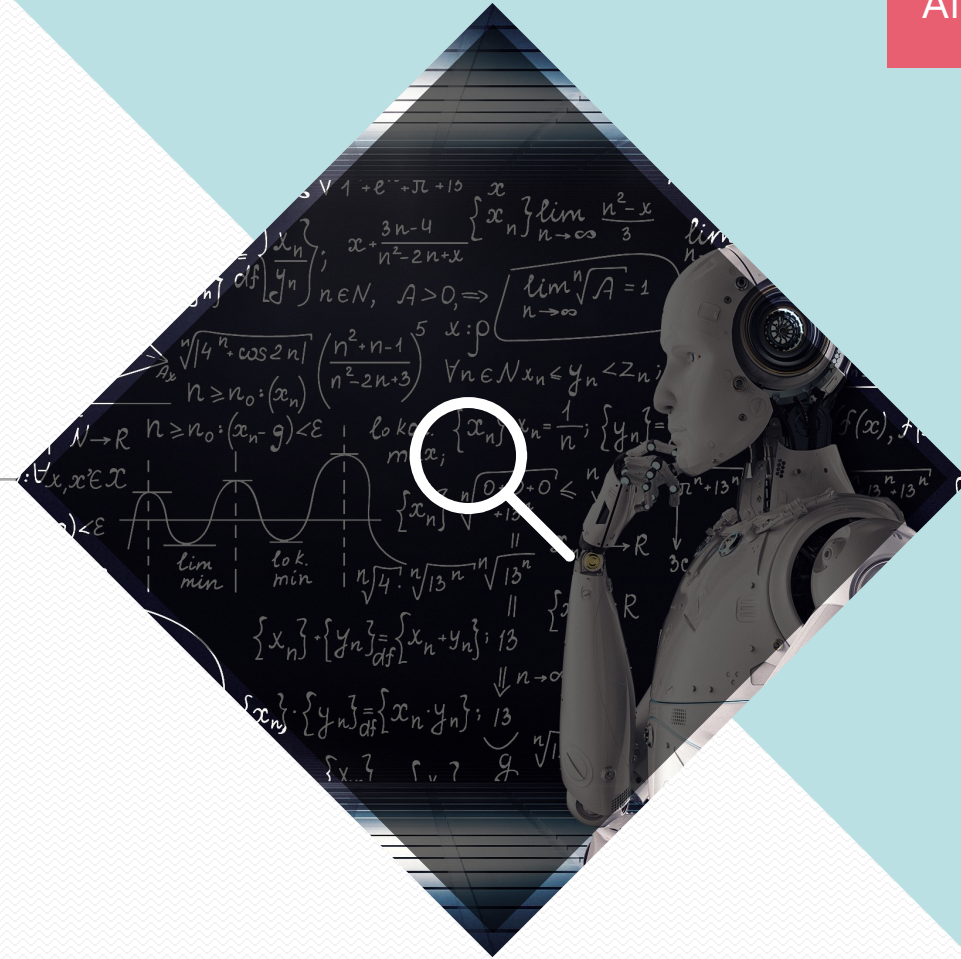


# GitHub



*“As more and more artificial intelligence is entering into the world”*

# CONTENTS

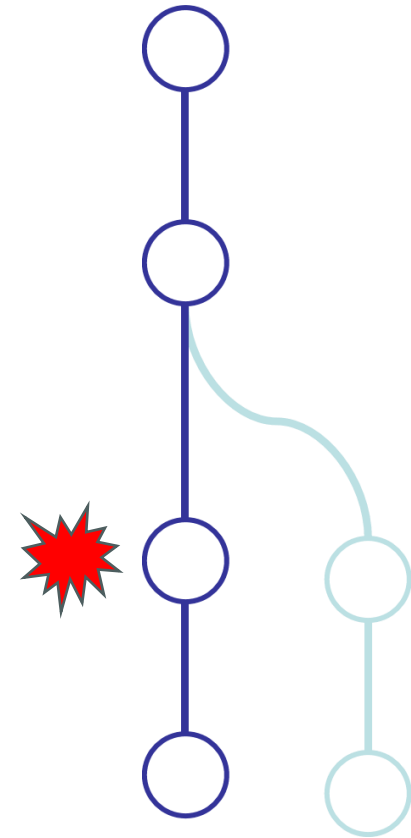
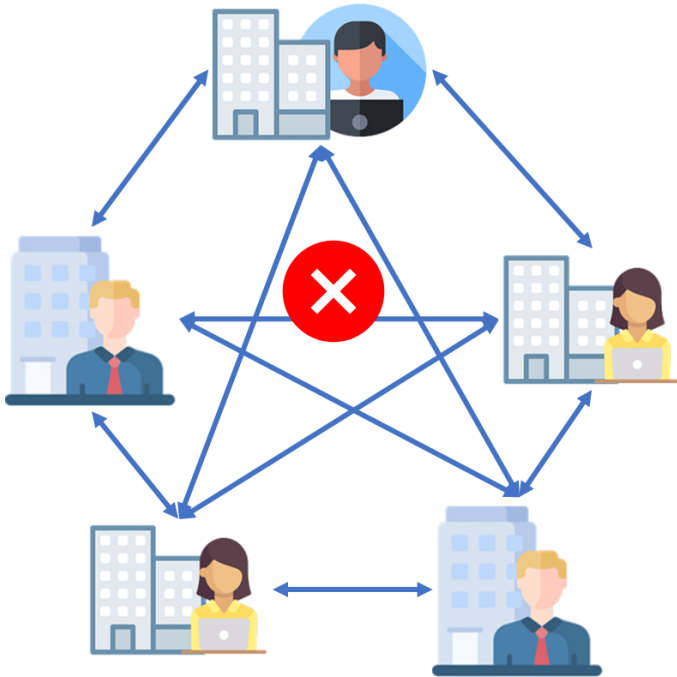
- *Before Start*
  - Why GitHub?
- *GitHub*
  - What is GitHub?
  - Getting Started
  - Setting Up a GitHub Repository
  - Creating Issue
  - Creating Pull Request
  - Ground Rule for Intel Curriculum
  - Homework Workflow



# Why GitHub?

# Why is Software Management needed?

- Collaboration
- Storing/Restoring
- Figure out what happened



# Git vs GitHub : What's the difference?

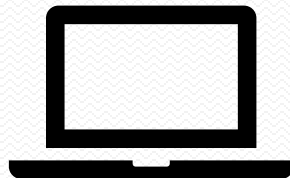


**Git**

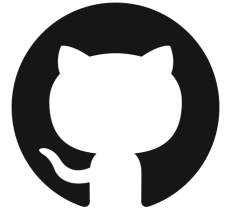
Used for  
version  
control

Tracks  
changes  
made to a  
file

Installed  
locally on  
computer



**GitHub**



Cloud-  
based



Used for  
hosting Git  
repository

Provides a  
web  
interface to  
view file  
changes

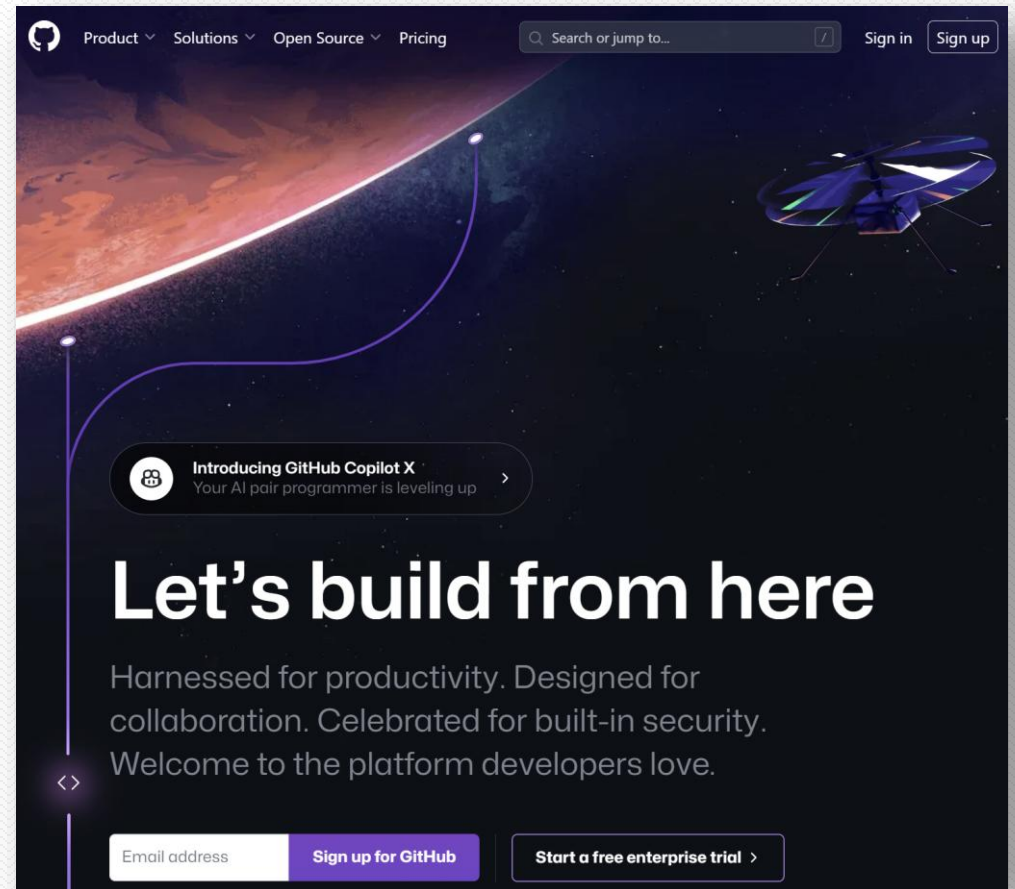


# GitHub

# What is GitHub?

- A platform and cloud-based service for software development and version control **using Git**, allowing developers to store and manage code.
- It is commonly used to host open source software development projects.
- Link : <https://github.com/>

- Without GitHub? You can setup a remote Git repository in your local server for collaboration.



# Getting Started

- GitHub Desktop

- An application that enables to interact with GitHub using a GUI instead of the command line or a web browser.
- Use GitHub Desktop to complete most Git commands from local machines with visual confirmation of changes.
- Supported OS
  - Windows
  - macOS
- Link : <https://desktop.github.com/>

- GitHub CLI

- An open source tool for using GitHub from your computer's command line.
- Supported OS
  - Windows
  - macOS
  - Linux
- Link : <https://github.com/cli/cli>
- Command to authenticate

```
$ gh auth login
```



# Getting Started

- Ensure the email address that will be associated with the commits you push from the command line as well as web-based Git operations you make

1 Settings

서울기술교육센터 (kccistc)  
Your personal account

Go to your personal profile

Public profile  
Account  
Appearance  
Accessibility  
Notifications

Access

Billing and plans  
Emails  
Password and authentication  
Sessions  
SSH and GPG keys  
Organizations  
Enterprises  
Moderation

## Emails

**wckang@korcham.net** – Primary

This email will be used for account-related notifications and can also be used for password resets.

- Not visible in emails  
This email will not be used as the 'from' address for web-based Git operations, e.g., edits and merges. We will instead use 148169295+kccistc@users.noreply.github.com.
- Receives notifications  
This email address is the default used for GitHub notifications, i.e., replies to issues, pull requests, etc.

Add email address

Email address Add

### 2 Primary email address

Because you have email privacy enabled, wckang@korcham.net will be used for account-related notifications as well as password resets. 148169295+kccistc@users.noreply.github.com will be used for web-based Git operations, e.g., edits and merges.

wckang@korcham.net Save

# Getting Started

- The first thing you should do when you install Git is to set your user name and email address.

```
$ git config --global user.name "FirstName LastName"
$ git config --global user.email "email@example.com"
```

- Check git setting

```
$ git config --list

user.name=FirstName LastName
user.email=email@example.com
color.status=auto
color.branch=auto
...
```

- Another way to check git setting
  - Open **.gitconfig**
  - Windows
    - C:\Users\<user name>\.gitconfig
  - Ubuntu
    - /home/<user name>\.gitconfig

# Practice : Getting Started

- Goal
  - Create GitHub account
  - Install GitHub CLI : <https://github.com/cli/cli>

```
$ gh auth login
```

```
? What account do you want to log into? GitHub.com
```

```
? What is your preferred protocol for Git operations? HTTPS
```

```
? Authenticate Git with your GitHub credentials? Yes
```

```
? How would you like to authenticate GitHub CLI? Login with a web browser
```

```
! First copy your one-time code: D8FF-6166
```

```
Press Enter to open github.com in your browser...
```

```
✓ Authentication complete.
```

```
- gh config set -h github.com git_protocol https
```

```
✓ Configured git protocol
```

```
! Authentication credentials saved in plain text
```

```
✓ Logged in as mokiya
```

# Practice : Getting Started

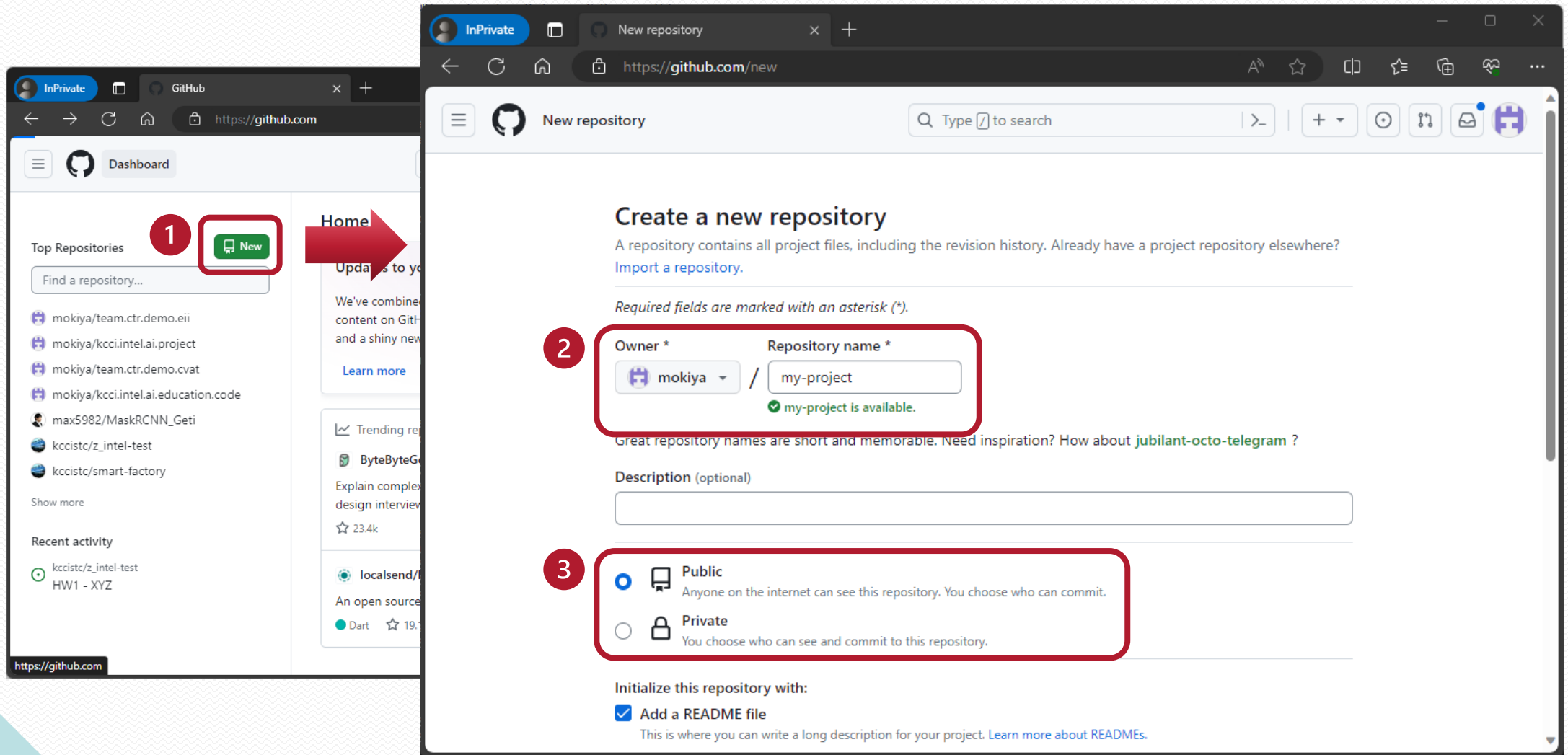
- Goal
  - Set primary email in GitHub
  - Set name & email in local machine

```
$ sudo apt-get update
$ sudo apt-get install git

$ git config --global user.name "FirstName LastName"
$ git config --global user.email "email@example.com"
```

# Setting Up a GitHub Repository

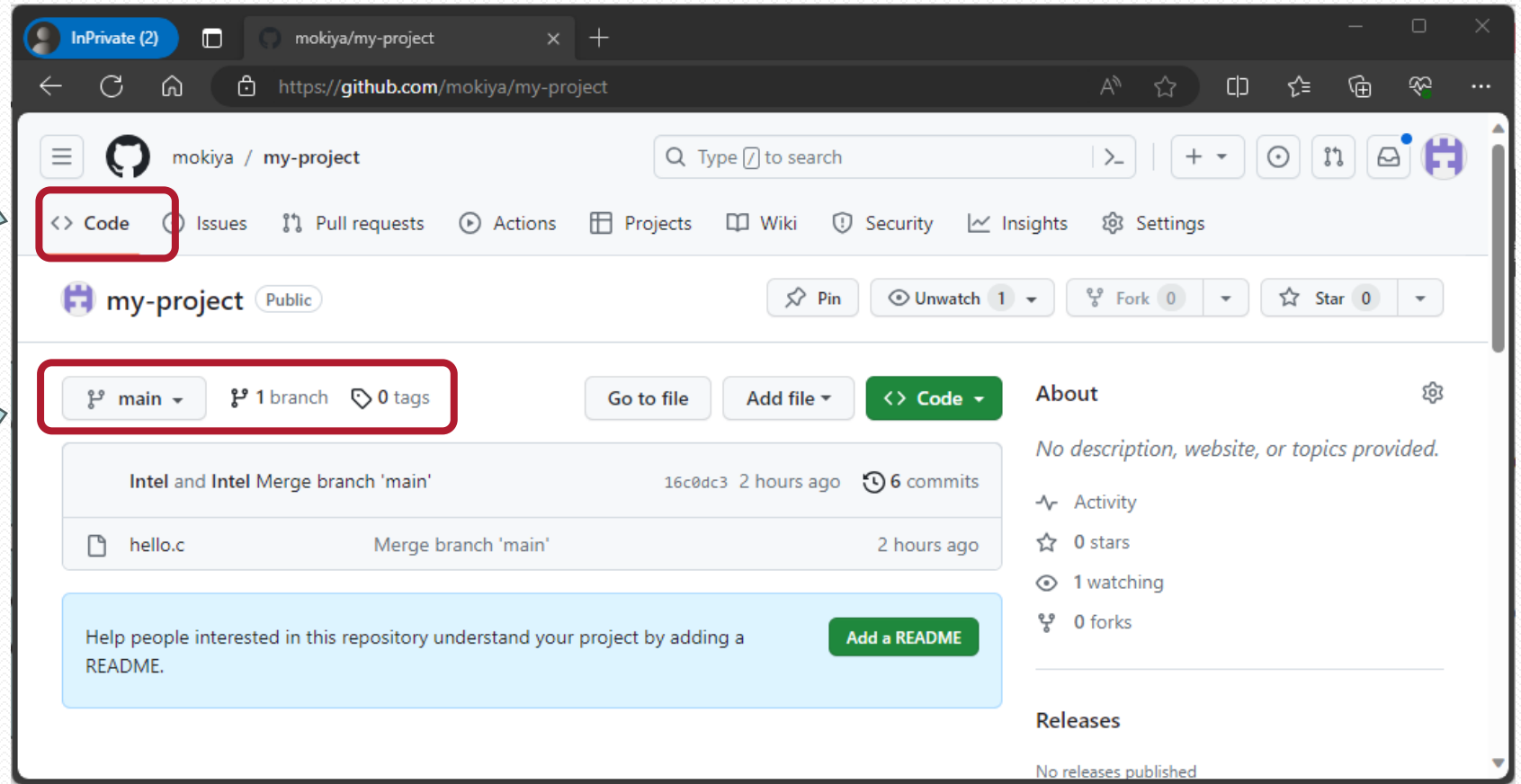
- Create new project in the GitHub



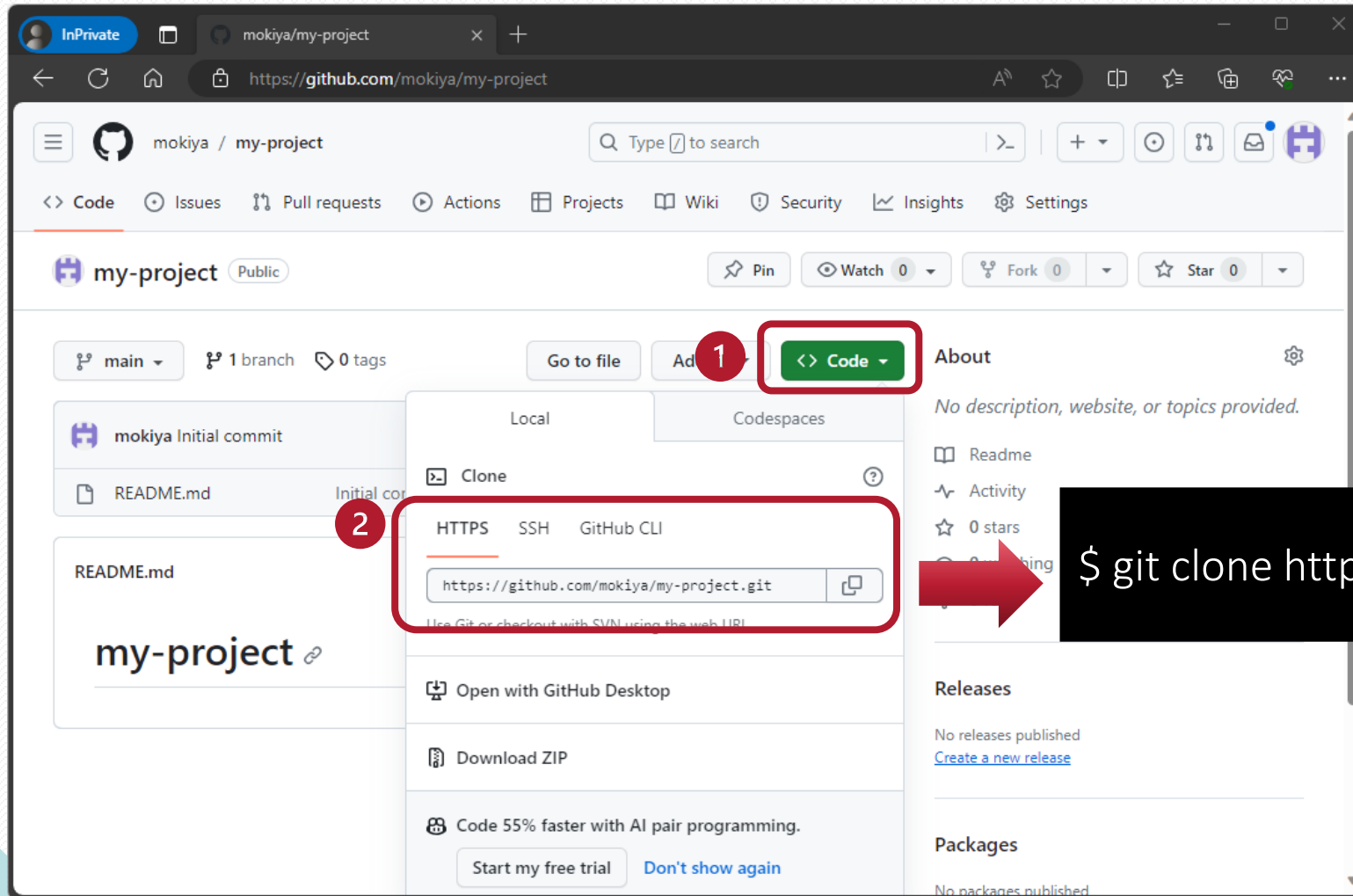
# Setting Up a GitHub Repository

Web Interface to see project source codes

Choose branch or tag what you want to a specific snapshot



# Setting Up a GitHub Repository



```
$ git clone https://github.com/mokiya/my-project.git
```

# Setting Up a GitHub Repository

- Once you successfully created GitHub repository, you can start adding commits from scratch and pushing commits to new GitHub repository.
- If you already have an existing repository, you can push an existing repository.

```
$ cd ~/existing-repo
```

```
$ git remote add github https://github.com/new-repo-url.git
```

```
$ git branch -M main
```

```
$ git push -u github main
```



# Practice : Setting Up a GitHub Repository

- Goal
  - Create new GitHub repository
  - Practice same way as we did last time, but try to practice using github.
  - Hints)
    - git clone
    - git remote
    - git push

# Practice : Setting Up a GitHub Repository

- An existing repository

```
$ cd ~/git-training  
$ git clone ~/git-training/my-project.git -b main project-z  
$ cd ./project-z  
  
$ git remote add github https://github.com/mokiyu/my-project.git  
$ git branch -M main  
$ git push -u github main
```

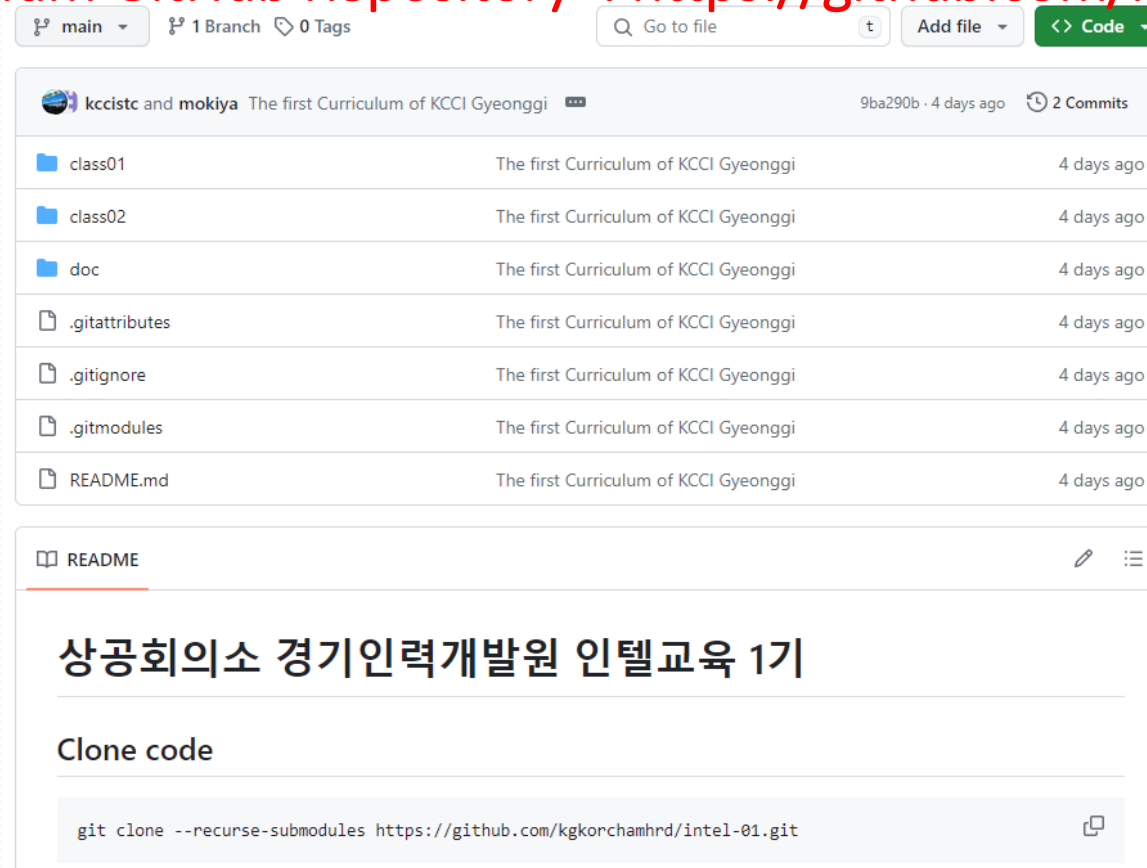
- Clone new Github repository to check, or visit GitHub web.

```
$ mkdir ~/github-training  
  
$ cd ~/github-training  
$ git clone https://github.com/mokiyu/my-project.git -b main  
$ cd ./my-project  
$ git log
```

# Intel Curriculum GitHub Repository

- Materials, sample codes, homework, team projects and so on for Intel curriculum will be managed in Intel Curriculum GitHub Repository.

Intel Curriculum GitHub Repository : <https://github.com/kgkorchamhrd/intel-04/>



The screenshot shows the GitHub repository page for 'kgkcorchamhrd/intel-04'. The repository is on the 'main' branch, has 1 branch and 0 tags. It contains 7 files: class01, class02, doc, .gitattributes, .gitignore, .gitmodules, and README.md. The README file is selected and shows the title '상공회의소 경기인력개발원 인텔교육 1기' and a 'Clone code' section with the command: `git clone --recurse-submodules https://github.com/kgkorchamhrd/intel-01.git`.

File	Description	Time
class01	The first Curriculum of KCCI Gyeonggi	4 days ago
class02	The first Curriculum of KCCI Gyeonggi	4 days ago
doc	The first Curriculum of KCCI Gyeonggi	4 days ago
.gitattributes	The first Curriculum of KCCI Gyeonggi	4 days ago
.gitignore	The first Curriculum of KCCI Gyeonggi	4 days ago
.gitmodules	The first Curriculum of KCCI Gyeonggi	4 days ago
README.md	The first Curriculum of KCCI Gyeonggi	4 days ago

**README**

## 상공회의소 경기인력개발원 인텔교육 1기

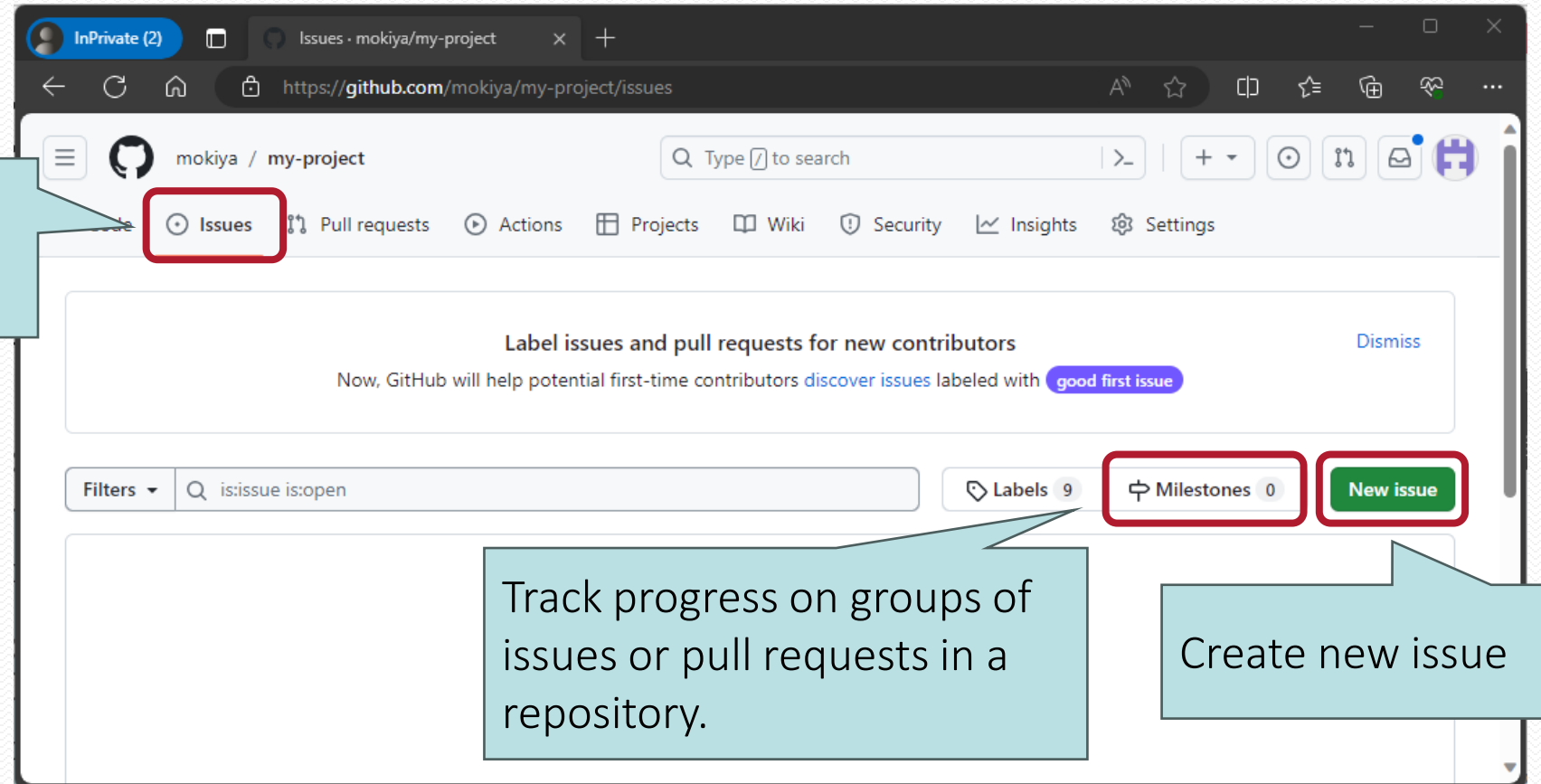
**Clone code**

```
git clone --recurse-submodules https://github.com/kgkorchamhrd/intel-01.git
```

# Creating Issue

- GitHub Issues to track ideas, feedback, tasks, or bugs for work on GitHub.

Web Interface to see project source codes



Track progress on groups of issues or pull requests in a repository.

Create new issue

# Practice : Creating Issue

- Goal
  - Submit new issue with following contents in Intel Curriculum Repository

The screenshot shows the GitHub 'New Issue' page for the repository 'kccisc/intel-02'. The page has a dark theme. The browser address bar shows the URL 'https://github.com/kccisc/intel-02/issues/new'. The repository name 'kccisc / intel-02' is displayed at the top. The page has tabs for 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', and 'Security'. The 'Issues' tab is selected. The issue title field is highlighted with a red box and a red circle with the number '1'. A callout box points to it with the text '[English Name] My practice repository'. The description field is highlighted with a red box and a red circle with the number '2'. A callout box points to it with the text '\* Name : Your English name', '\* Github account : Your GitHub account', and '\* My practice repository URL : https://github.com/abcde/my-project.git'. The 'Submit new issue' button is highlighted with a red box and a red circle with the number '3'. A callout box points to it with the text 'Submit new issue'.

1 [English Name] My practice repository

2

- \* Name : Your English name
- \* Github account : Your GitHub account
- \* My practice repository URL : https://github.com/abcde/my-project.git

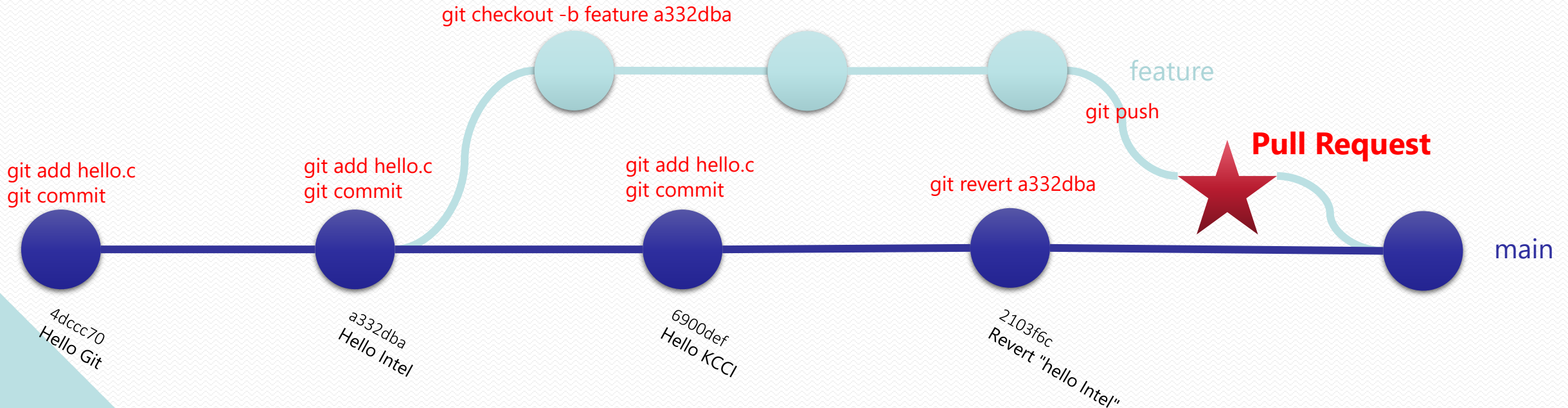
3 Submit new issue

# Practice : Creating Issue

- Goal
  - Based on your created issue, teacher will ...
    - Give Intel Curriculum Repository permission to your GitHub account
    - Visit your practice repository and review your practice.
- Please check teacher's comment in your created issue, then follow up teacher's feedback.

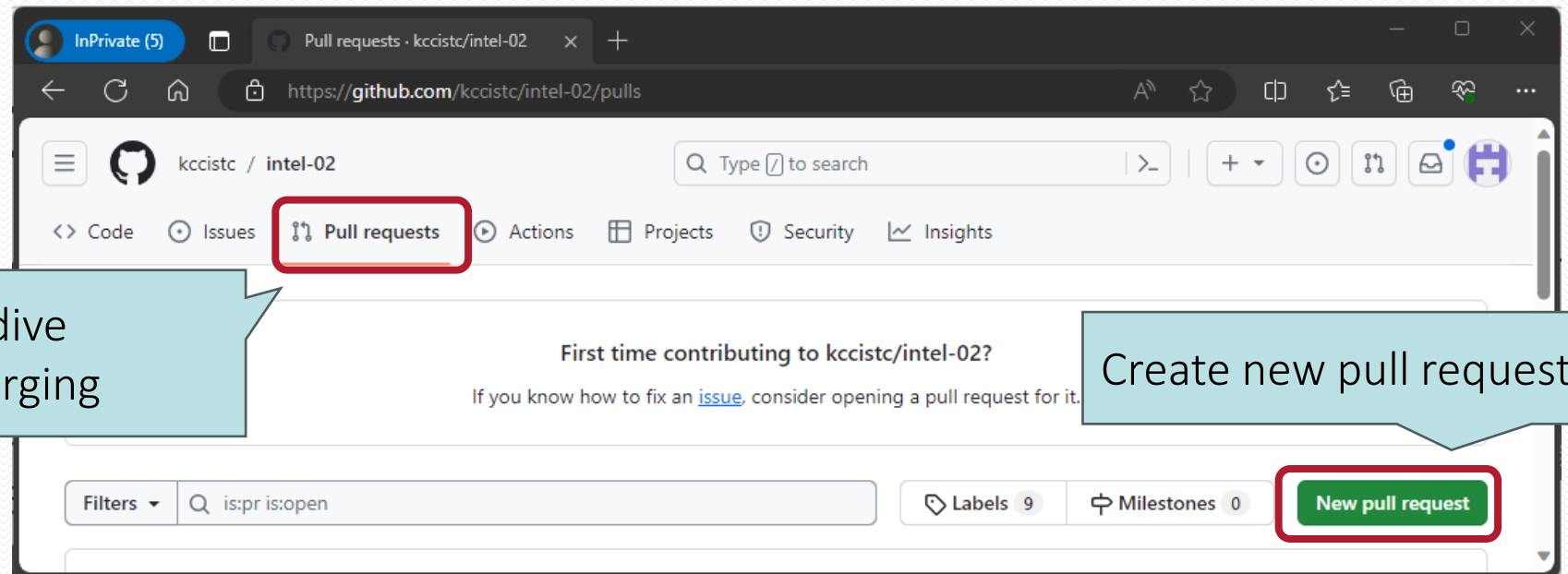
# Creating Pull Request

- GitHub is a web-based hosting service for Git repositories.
- In simple terms, you can use Git without GitHub, but you CANNOT use GitHub without Git.
- GitHub offers Pull Request as web interface to discuss changes for merging.



# Creating Pull Request

- Pull Request tells others about changes you've pushed to a branch in a repository on GitHub.
- Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.

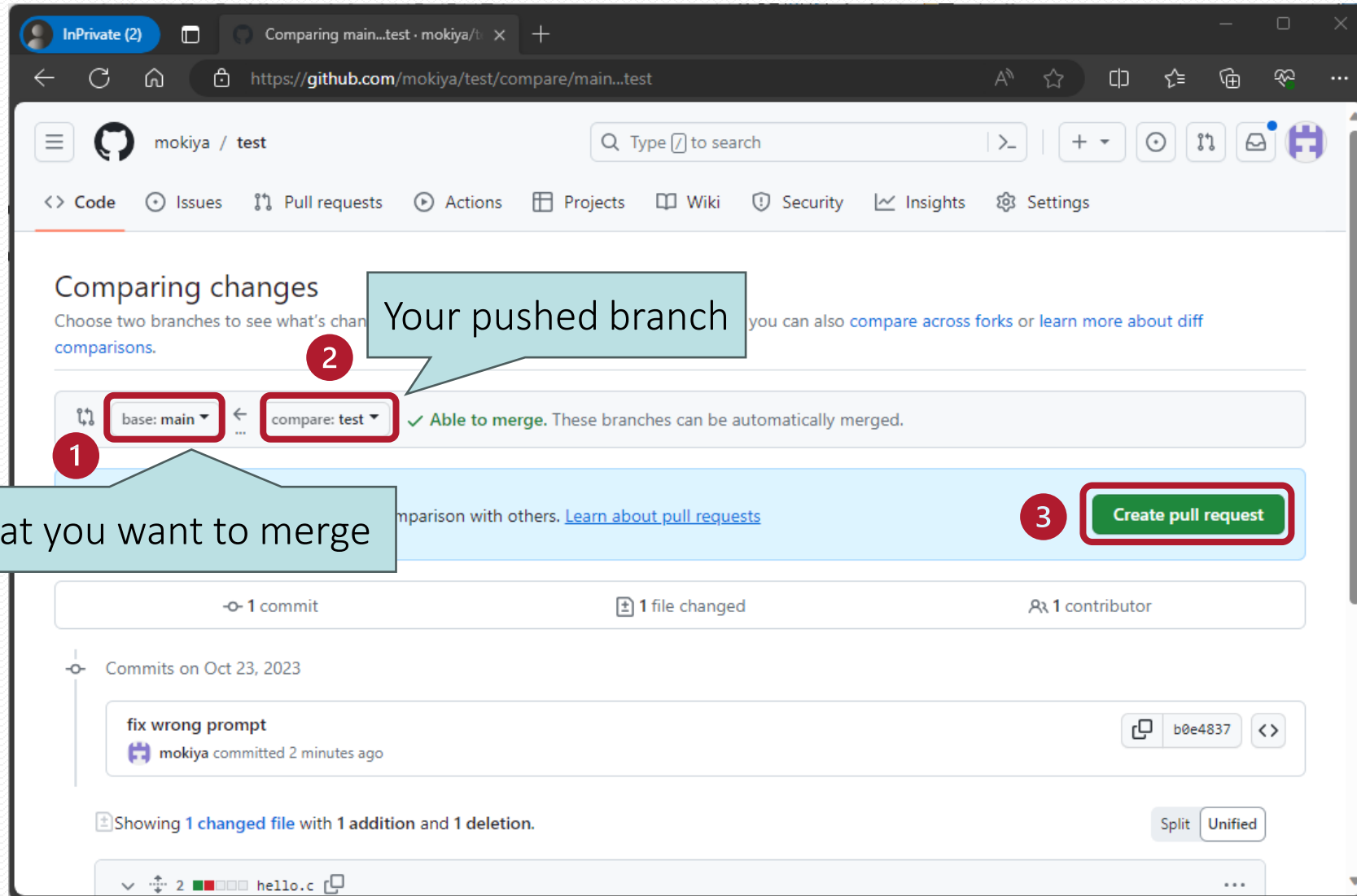


Web Interface to have deep-dive discussion on changes for merging

Create new pull request



# Creating Pull Request



# Creating Pull Request

The screenshot shows the GitHub interface for creating a pull request. The browser address bar shows the URL `https://github.com/mokiya/test/compare/main...test`. The repository name is `mokiya / test`. The page title is `Open a pull request`. Below the title, there is a description: `Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.`

The main form area has a title `fix wrong prompt`. Below the title, there are two tabs: `Write` and `Preview`. The `Write` tab is active. The text area contains the message: `I'd appreciate it, if you can review and give me any feedback.`

On the right side, there are three sections: `Reviewers` (No reviews), `Assignees` (No one—[assign yourself](#)), and `Labels` (None yet). Below these are `Projects` (None yet) and `Milestone` (No milestone).

At the bottom right, there is a green button labeled `Create pull request`.

Numbered annotations (1-4) and callouts are present:

- 1: Points to the text area containing the message.
- 2: Points to the title `fix wrong prompt`.
- 3: Points to the `Assignees` section.
- 4: Points to the `Create pull request` button.

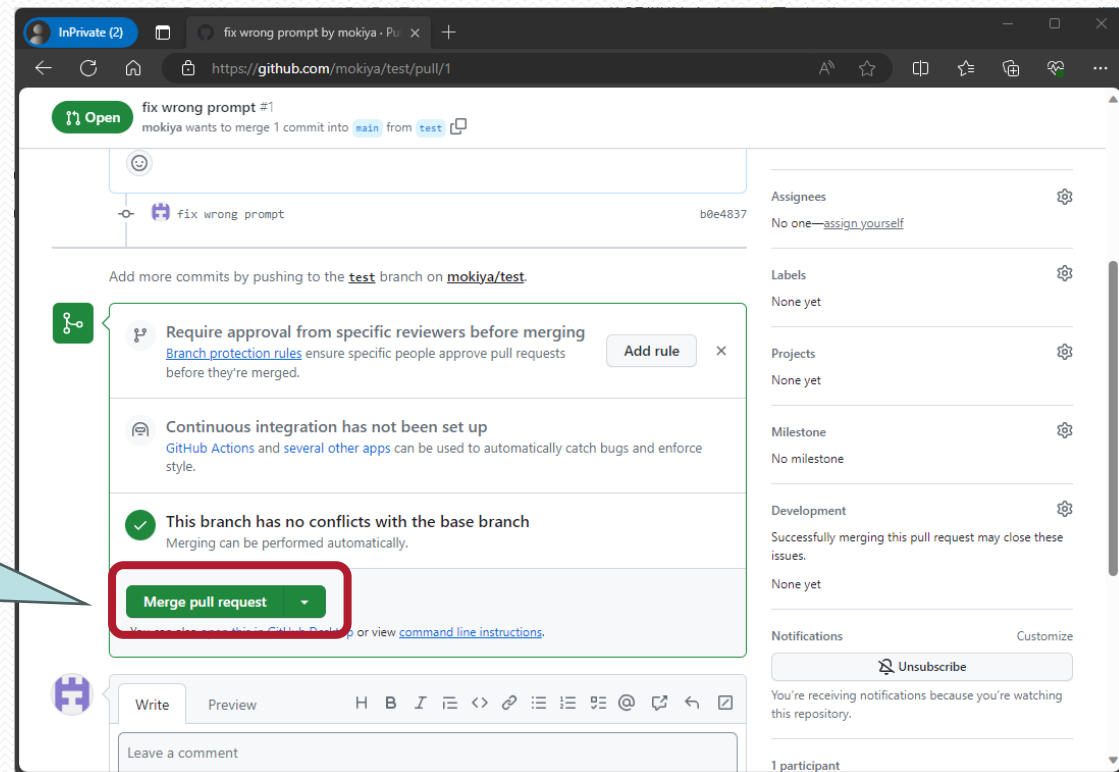
Callouts:

- `Choose reviewer`: Points to the `Reviewers` section.
- `Choose patch owner`: Points to the `Assignees` section.

# Creating Pull Request

- Once a pull request is opened, issue discussion will start.
- You might need to rework/re-upload changes based on reviewer feedback.
- You might need to rework/re-upload changes, if there are conflicts.

If reviewer approved to merge patches, patch owner can merge changes.



# Practice : Creating Pull Request

- Goal
  - Clone Intel Curriculum GitHub.
  - Create commits with following changes
    - Create directories named as your English name under `class01/homework/` and `class02/homework/`
    - Push commits to a new branch following the naming convention below
    - Naming convention : `class01-hw1-[github account]`
    - Example) if GitHub account is abcdef, branch name should be `class01-hw1-abcdef`.
  - Create Pull Request
    - Base branch should be “`main`”
    - Reviewer should be “`mokiya`”
  - Once reviewer approve your changes, merge your changes in created Pull Request

# Ground Rules for Intel Curriculum

Create an Issue with mandatory fields below using GitHub Issue in Intel Curriculum GitHub Repository

- Title / Comment
- Milestone
- Assignee
- Development

Create a Pull Request with mandatory fields in Intel Curriculum GitHub Repository

- Base branch : main
- Reviewer

Based on feedback or conflicts, might need to re-work & re-push changes.

Create an  
issue



Clone/Commit



Push



Create a Pull  
Request



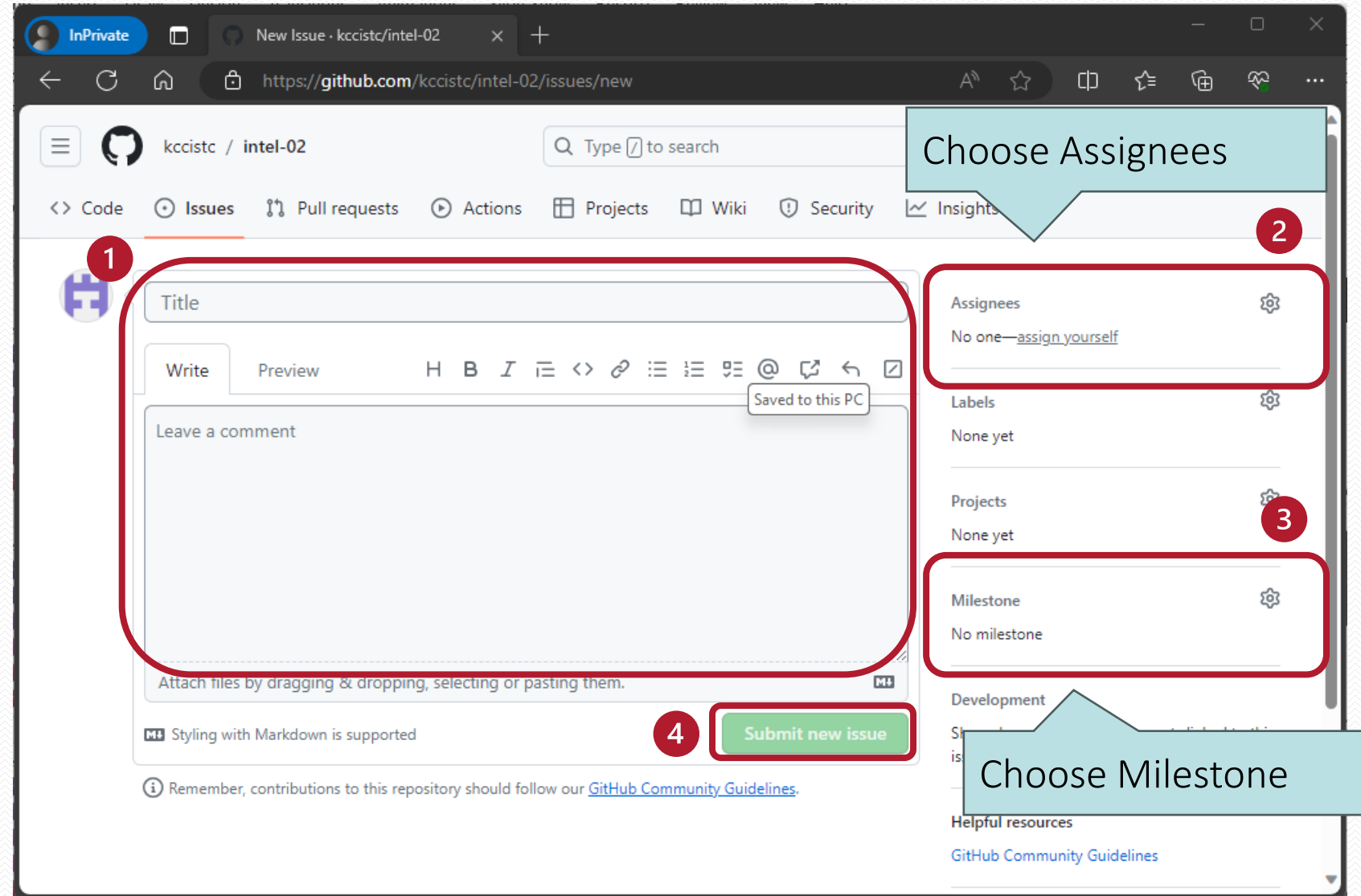
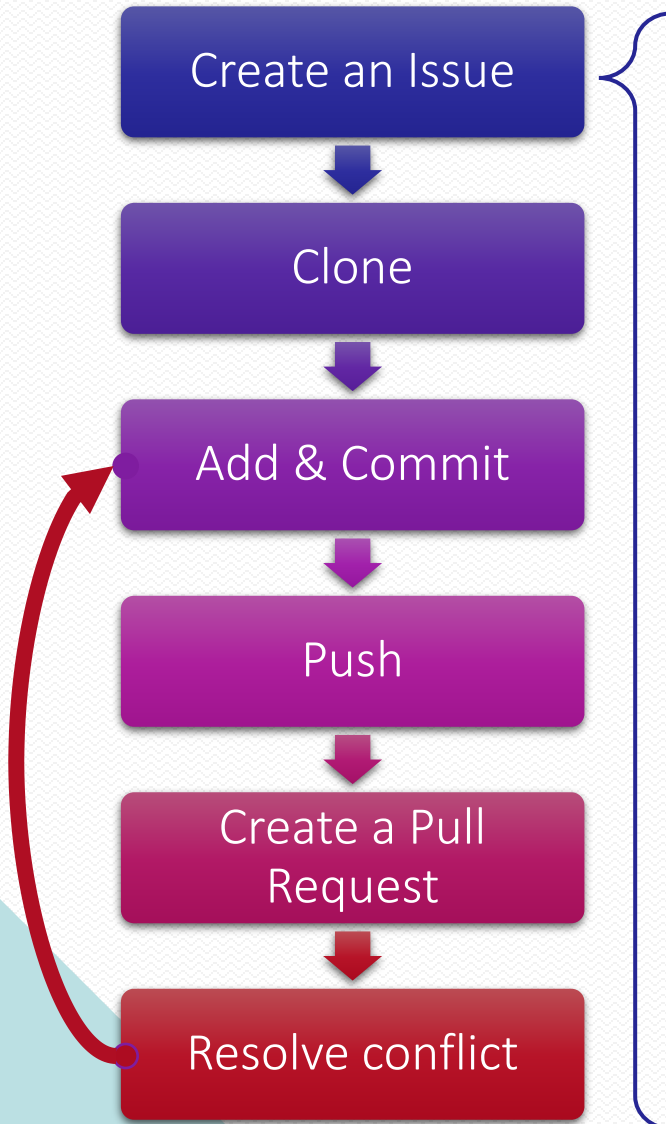
Merge

Push commits to a new branch following the naming convention below

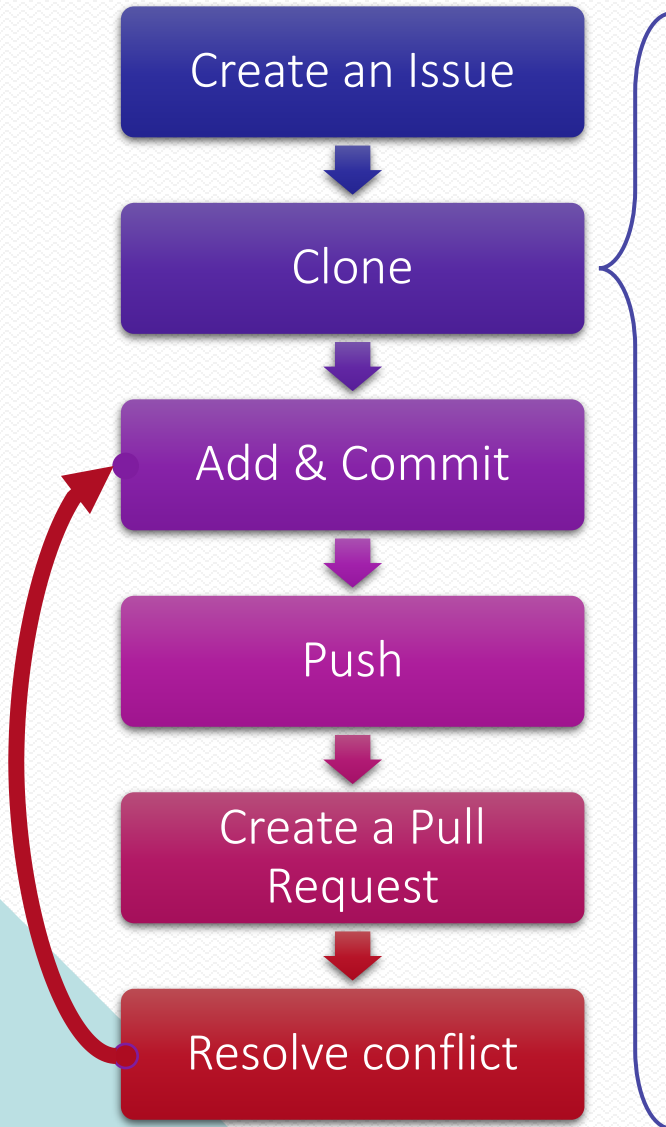
- Naming convention : class[xx]-hw[y]-[GitHub account]
- Example) GitHub account : abcdef | Course : class01 | Homework 2  
Branch name should be class01-hw2-abcdef.

Once reviewer agrees to merge,  
merge your patches in GitHub

# Homework Workflow based Ground Rules



# Homework Workflow based Ground Rules

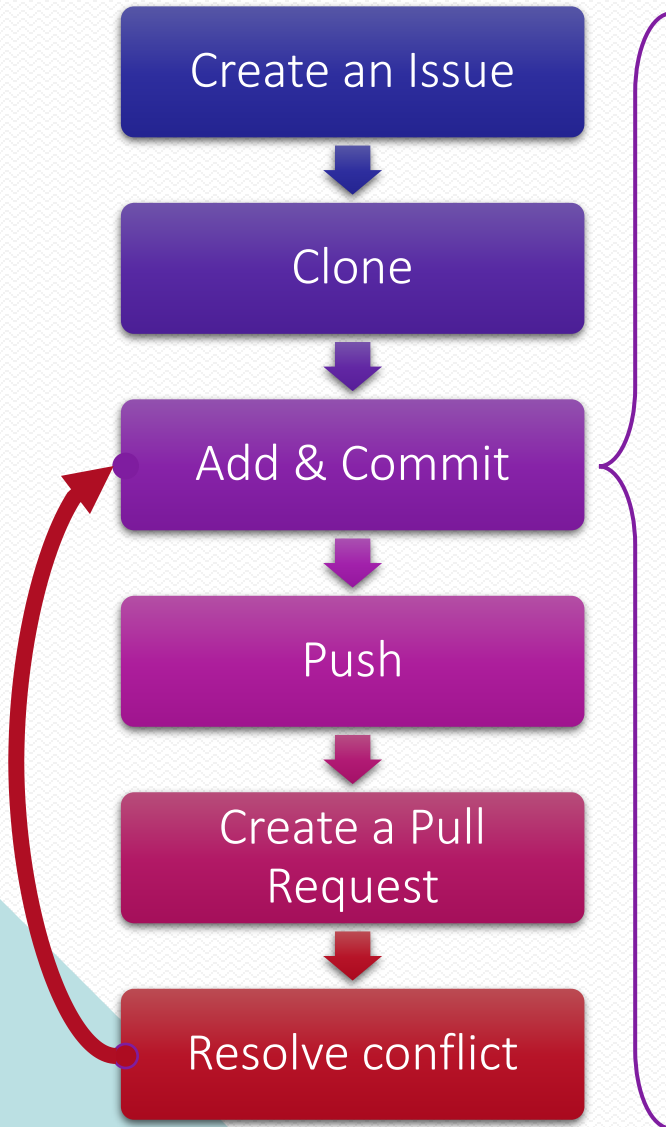


- A Git command line utility which is used to target an existing repository and create a clone, or copy of the target repository.

```
$ git clone <Repository GitHub URL> -b <Branch or Tag Name> <Directory>
```

- Clone Branch called <Branch or Tag Name> from the repository located at <Repository GitHub URL> into the folder called <Directory> on the local machine.

# Homework Workflow based Ground Rules



- A Git command adds a change in the working directory to the staging area.

```
$ git add <File name 1> <File name 2> <Directory 1>
```

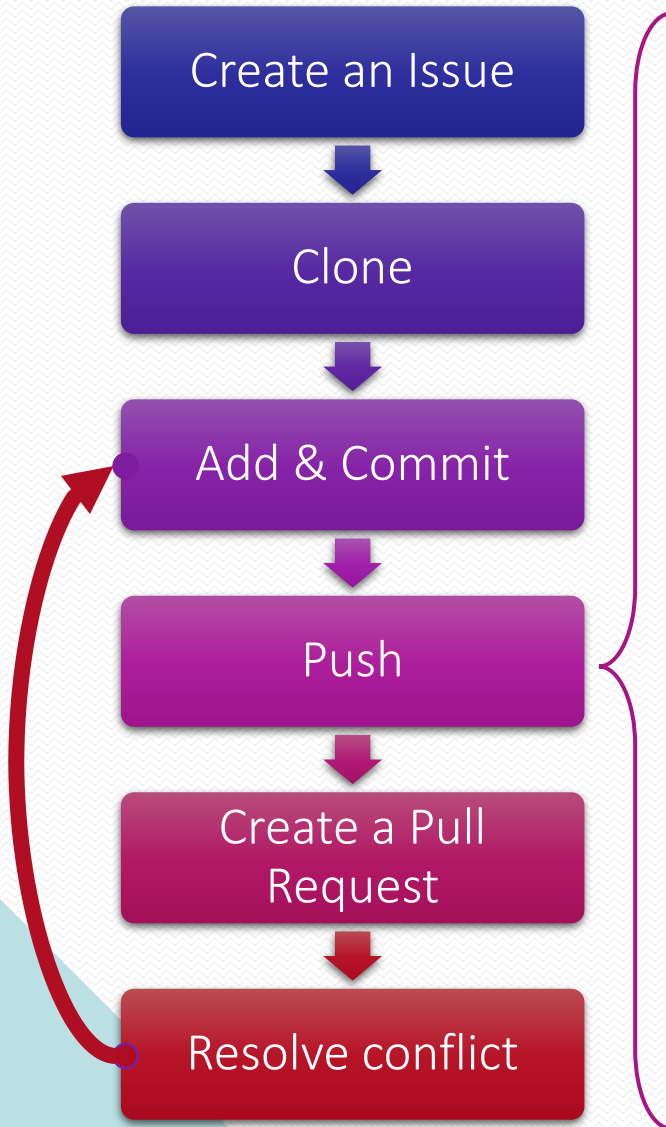
- Allow <File name 1>, <File name 2>, <Directory> to stage files to be committed.
- A Git command creates a commit, which is like a snapshot of your repository. These commits are snapshots of your entire repository at specific times.

```
$ git commit -m <Comment>
```

- Without -m option, default text editor will be opened to create the commit message.



# Homework Workflow based Ground Rules

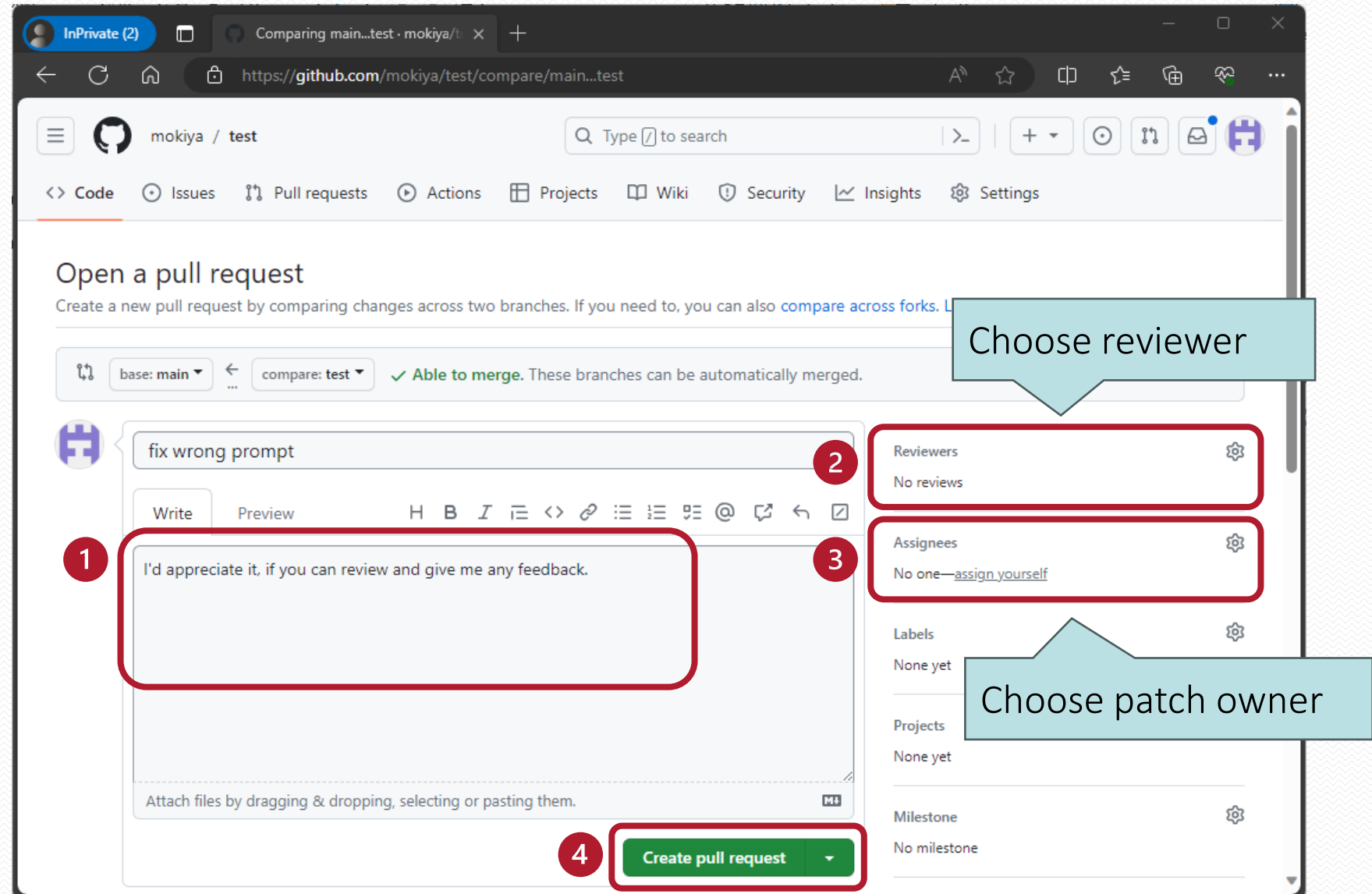
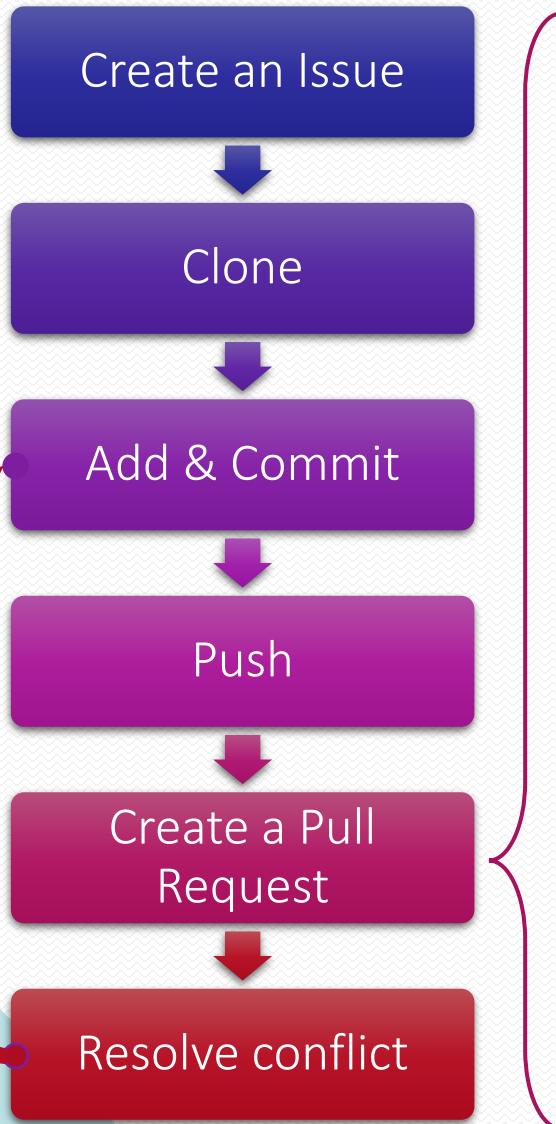


- A Git command is used to upload local repository content to a remote repository.

```
$ git push <Remote> HEAD:<Branch Name>
```

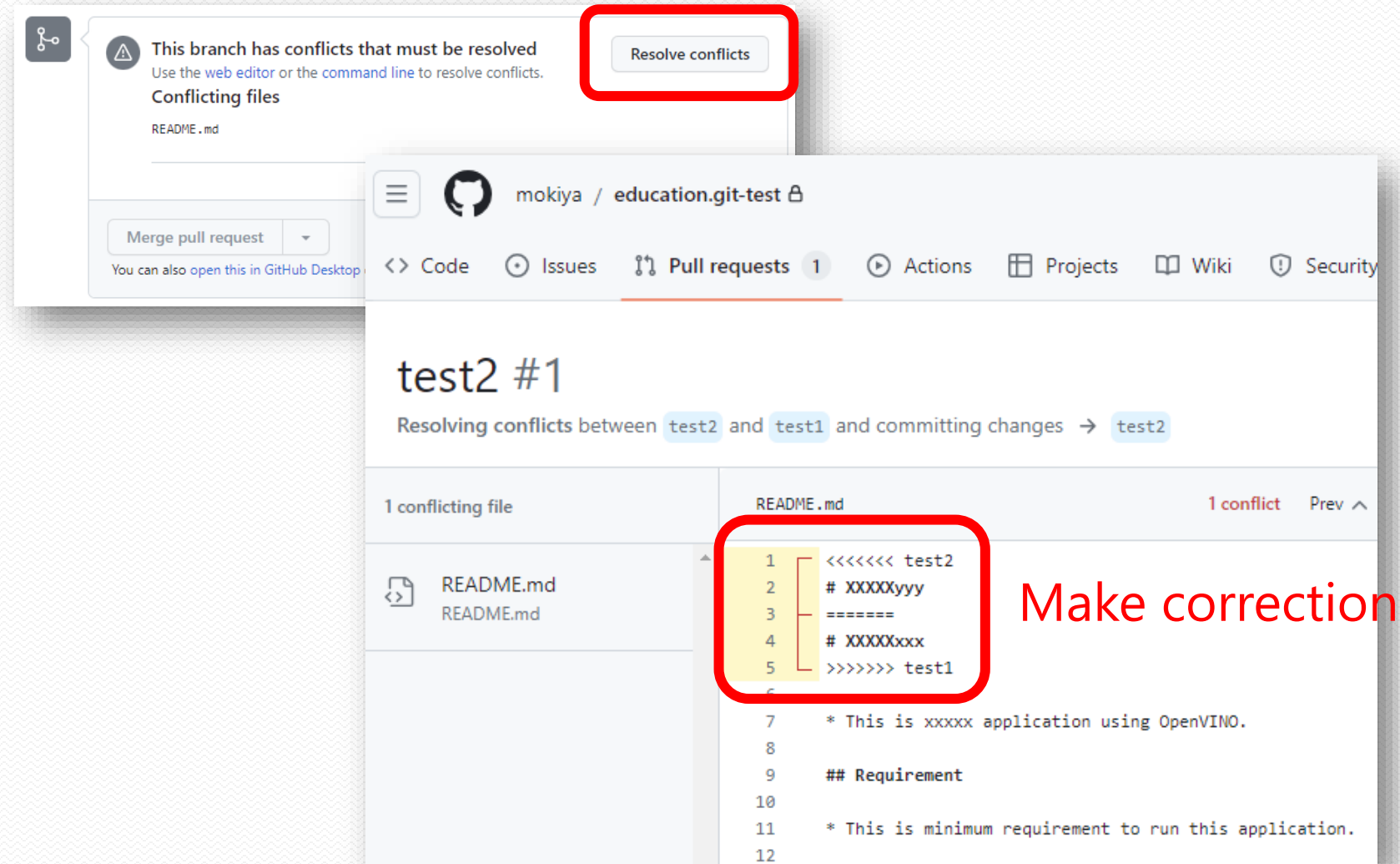
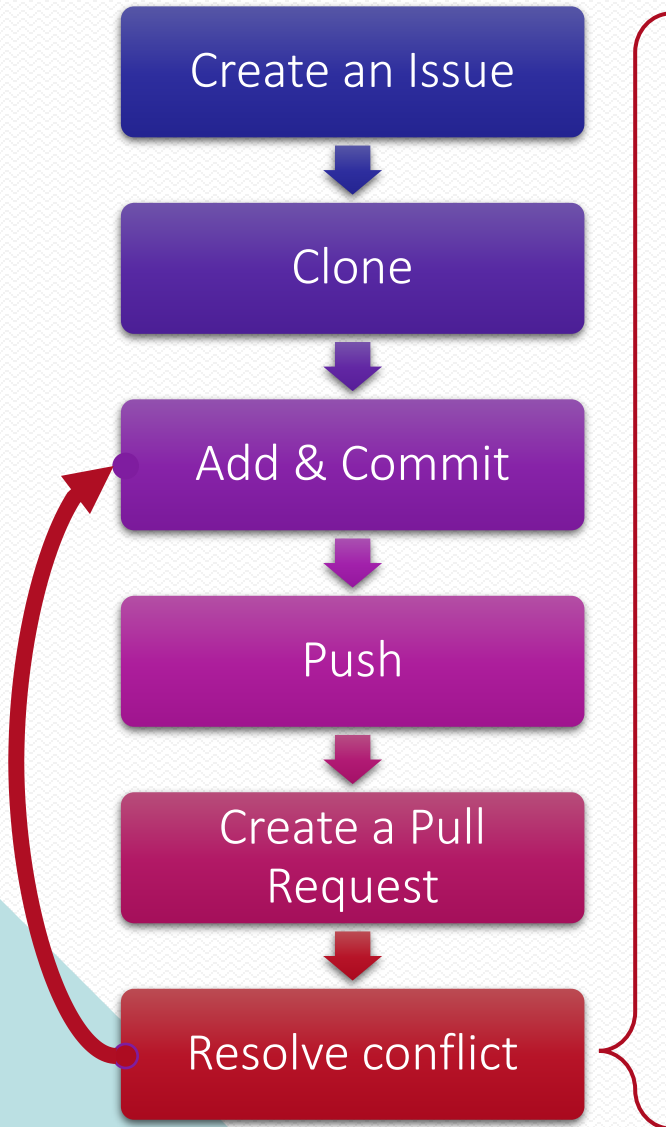
- Push your changes called **HEAD:** to specific branch (**<Branch Name>**) in the remote repository (**<remote repository>**)
- You can use commit id or local branch name instead of **HEAD:**
- Naming convention : **class[xx]-hw[y]-[GitHub account]**  
Ex) GitHub account : **abcdef**  
Course : **class01**  
Homework 2 : **hw2**  
Branch name should be **class01-hw2-abcdef**

# Homework Workflow based Ground Rules

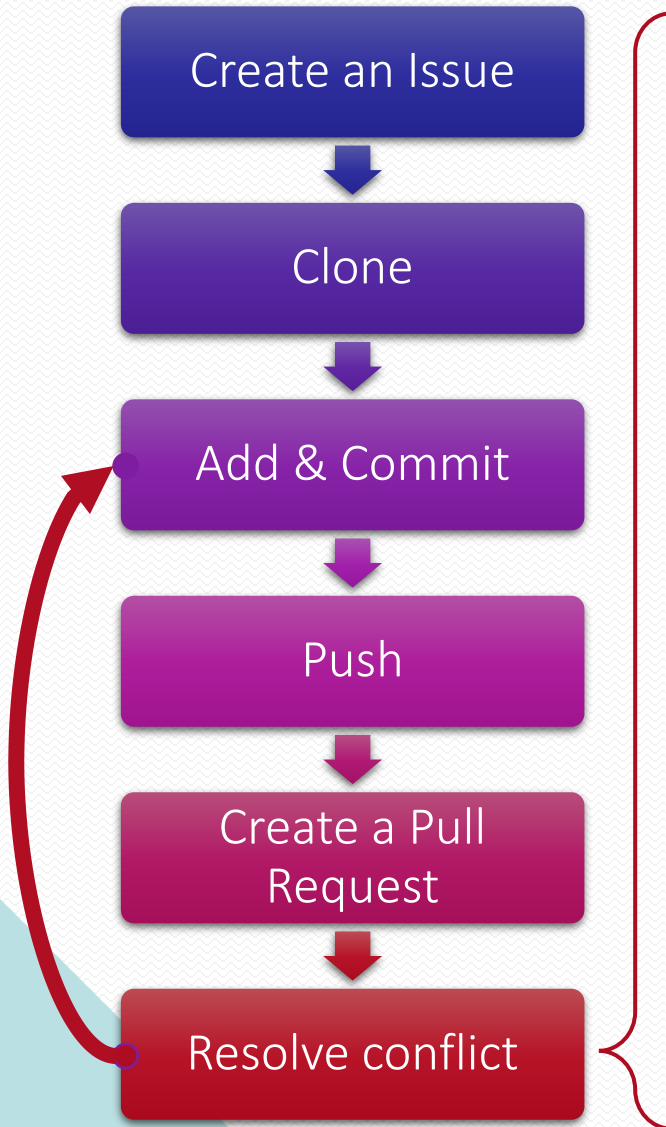


# Homework Workflow based Ground Rules

- Resolution #01 : Resolve conflicts in the GitHub



# Homework Workflow based Ground Rules



- Resolution #02 : Resolve conflicts in local machine
  - Step 1: Clone the repository or update your local repository with the latest changes.

```
$ git pull origin test1
```

- Step 2: Switch to the head branch of the pull request.

```
$ git checkout test2
```

- Step 3: Merge the base branch into the head branch.

```
$ git merge test1
```

- Step 4: Fix the conflicts and commit the result.
- Step 5: Push the changes.

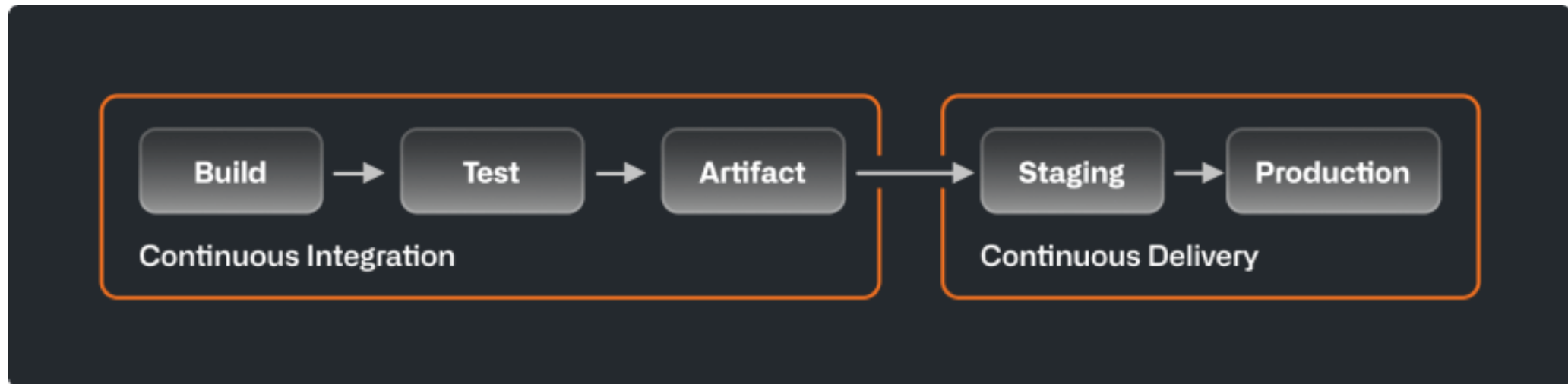
```
$ git push -u origin test2
```

# Practice : Homework Workflow

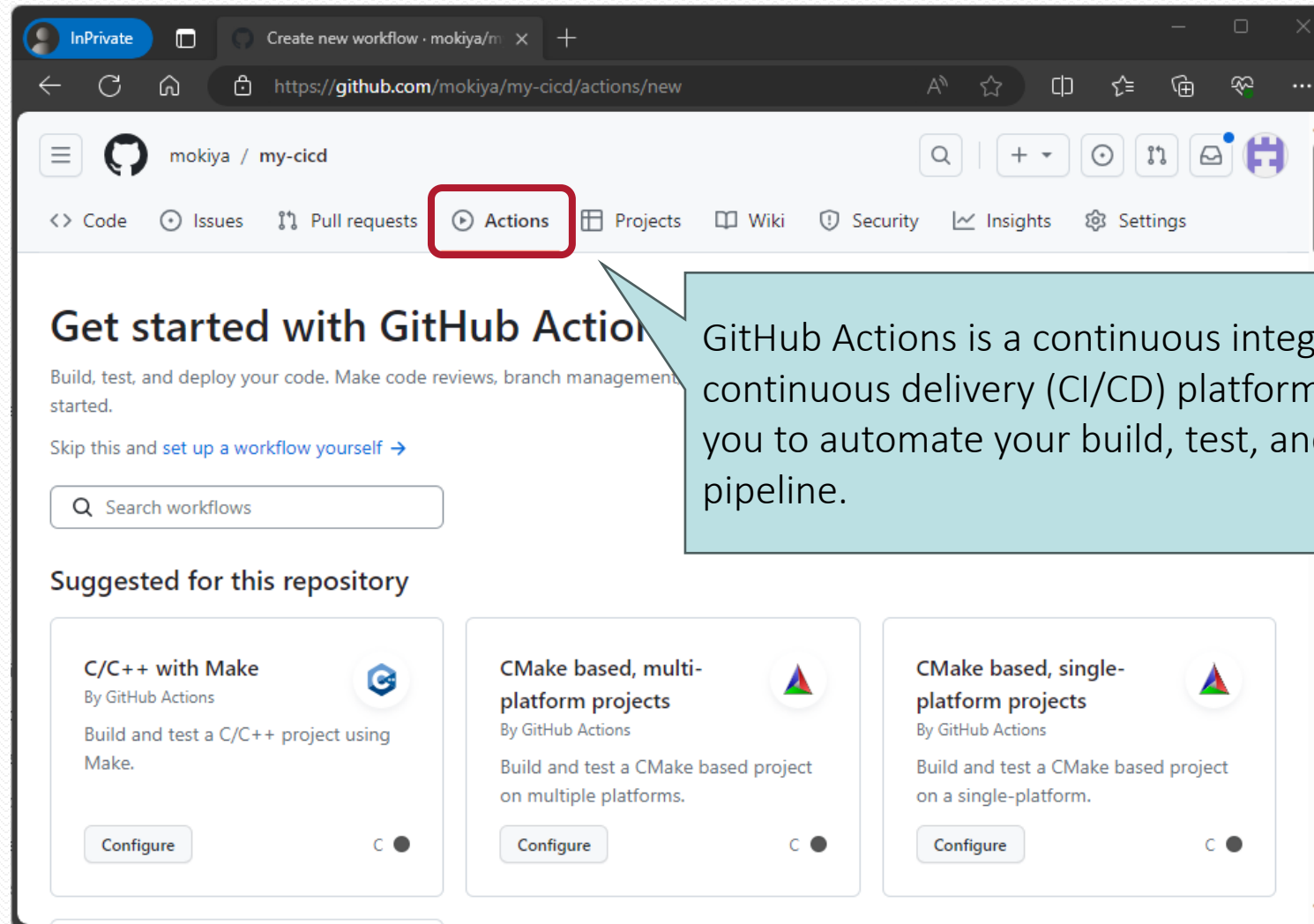
- Goal
  - Enter your name into the table in the two files below.
    - ./class01/README.md
    - ./class02/README.md
  - Be sure to work according to ground rules and create a PR.

# CI / CD (Continuous Integration & Delivery)

- CI (Continuous Integration)
  - Automatically builds, tests, and integrates code changes within a shared repository
- CD (Continuous Delivery)
  - Automatically delivers code changes to production-ready environments for approval, then deploys code changes to customers directly.



# GitHub Actions



GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment pipeline.

# Self-hosted Runner for GitHub Actions

- A self-hosted runner is a system that you deploy and manage to execute jobs from GitHub Actions on GitHub.com

The image illustrates the process of adding a self-hosted runner to a GitHub repository through three sequential screenshots, connected by red arrows.

**Screenshot 1: GitHub Runners Page**  
The first screenshot shows the GitHub interface for a repository named 'mokiya / my-cicd'. The 'Settings' tab is selected, and the 'Runners' section is visible. A red box highlights the 'New self-hosted runner' button. A red circle with the number '1' is placed over the 'Settings' tab, and a red circle with the number '2' is placed over the 'Runners' section in the left sidebar.

**Screenshot 2: Add new self-hosted runner page**  
The second screenshot shows the 'Add new self-hosted runner' page. The 'Runner image' section is visible, and the 'Linux' option is selected. A red box highlights the 'Linux' option. A red circle with the number '4' is placed over the 'Linux' option.

**Screenshot 3: GitHub Runners Page (After Addition)**  
The third screenshot shows the 'Runners' page after a new runner has been added. A red box highlights the 'New self-hosted runner' button. A red circle with the number '3' is placed over the 'New self-hosted runner' button. Below the button, a table lists the runners:

Runners	Status
gram self-hosted Linux X64	Idle



# Workflow in GitHub Actions

The image shows two overlapping browser windows from GitHub. The left window displays the repository page for 'mokiya / my-cicd' with the 'Actions' tab selected (marked with a red circle and '1'). Below the repository name, there's a section 'Suggested for this repository' (marked with a red circle and '2') which lists 'C/C++ with Make' as a suggested action. A red arrow points from this suggestion to the right window. The right window shows the workflow editor for 'my-cicd/.github/workflows/c-cpp.yml'. The workflow file content is as follows:

```
2
3 on:
4   push:
5     branches: [ "main" ]
6   pull_request:
7     branches: [ "main" ]
8
9 jobs:
10  build:
11
12    runs-on: self-hosted
13
14    steps:
15      - uses: actions/checkout@v3
16      - name: make
17        run: gcc ./hello.c -o ./hello
18      - name: test
19        run: ./hello
20      - name: deploy
21        run: echo "deploy"
22
```

The 'runs-on: self-hosted' line and the 'steps' block are highlighted with a red rounded rectangle. The right sidebar of the editor shows the 'Marketplace' tab with a search bar and a list of featured actions: 'Upload a Build Artifact' (2.5k stars), 'Setup Java JDK' (1.3k stars), and 'Close Stale Issues' (1.1k stars).

# Demo : CI / CD

The image displays two screenshots of the GitHub Actions interface, illustrating a CI/CD workflow.

**Left Screenshot (Workflow List):**

- URL: <https://github.com/mokiya/my-cicd/actions>
- Page Title: mokiya / my-cicd
- Navigation: Code, Issues, Pull requests, **Actions**, Projects, Wiki
- Message: Workflow run deleted successfully.
- Section: Actions (New workflow)
- Filter: All workflows
- Workflow: C/C++ CI
- Management: Caches, Runners (Beta)
- Section: All workflows (Showing runs from all workflows)
- Workflow run: 1 workflow run
- Workflow run details: **Update c-cpp.yml** (C/C++ CI #3: Commit 663822f pushed by mokiya)

**Right Screenshot (Workflow Run Details):**

- URL: <https://github.com/mokiya/my-cicd/actions/runs/6612651512>
- Page Title: mokiya / my-cicd
- Navigation: Code, Issues, Pull requests, **Actions**, Projects, Wiki, Security, Insights, Settings
- Section: C/C++ CI
- Workflow run: **Update c-cpp.yml #3** (Re-run all jobs)
- Summary: Triggered via push 2 minutes ago, Status: **Success**, Total duration: 25s, Artifacts: -
- Jobs: **build**
- Run details: Usage, Workflow file
- Job details: **c-cpp.yml** (on: push), **build** (15s)

# Demo : CI / CD

The image shows a GitHub Actions workflow being edited and its execution results. A red arrow points from the workflow definition to the execution logs.

**Workflow Definition (Left Panel):**

```
2
3 on:
4   push:
5     branches: [ "main" ]
6   pull_request:
7     branches: [ "main" ]
8
9 jobs:
10  build:
11    runs-on: self-hosted
12
13    steps:
14      - uses: actions/checkout@v3
15      - name: make
16        run: gcc ./hello.c -o ./hello
17      - name: test
18        run: ./hello
19      - name: deploy
20        run: echo "deploy"
21
22
```

**Execution Summary (Right Panel):**

- Summary:** build (succeeded 4 minutes ago in 15s)
- Jobs:** build (succeeded)
- Run details:** Usage, Workflow file

**Build Log (Right Panel):**

- Set up job:** 2s
- Run actions/checkout@v3:** 0s
- make:** 0s
  - 1 Run gcc ./hello.c -o ./hello
  - 2 gcc ./hello.c -o ./hello
  - 3 shell: /usr/bin/bash -e {0}
- test:** 0s
  - 1 Run ./hello
  - 4 Hello Git
  - 5 Hello KCCI
- deploy:** 0s
  - 1 Run echo "deploy"
  - 4 deploy
- Post Run actions/checkout@v3:** 0s
- Complete job:** 0s

