

Full Stack Web Development

# Intro to Git, Github and Exercise

# Outline

- Introduction Git
  - What is git ?
  - Why we need git ?
  - How to configure git ?
  - Git command list
  - Conventional Git Commit Message
- Exercise



# What is Git ?

Git is a tool used to manage the versions of the programs we create. Easily, we can separate which programs are being released and which are still being developed.

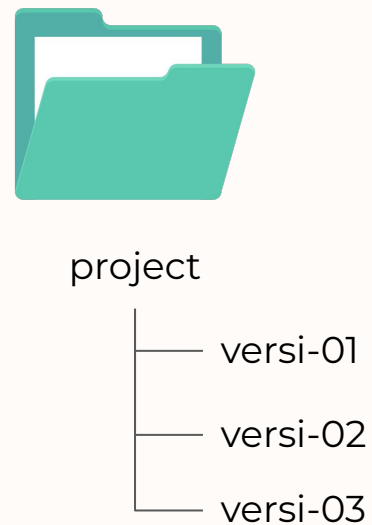


# Illustration

## *Without Git*



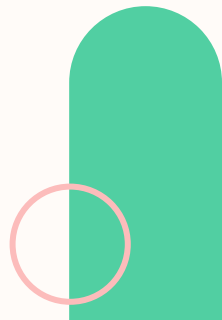
## *With Git*



# Why we need git ?

With git it will be easier to manage projects, where every change made can be documented in history.

Apart from that, we can also save our project online on web-based hosting repository such as **GitHub**, **GitLab**, **Bitbucket** etc., and make the collaboration process easier.



# Illustration

## *From Local*



project

— versi-01

— versi-02

— versi-03



# What is Github ?

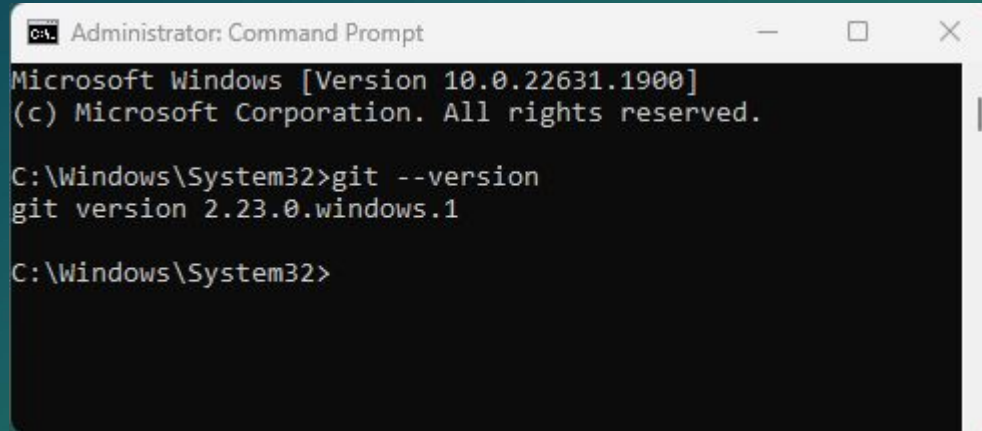
[GitHub](#) is a web-based platform that incorporates git version control features so they can be used collaboratively. It also includes project and team management features, as well as opportunities for networking and social coding.



# Illustration

---

Make sure that git is already installed on your laptop.



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.22631.1900]
(c) Microsoft Corporation. All rights reserved.

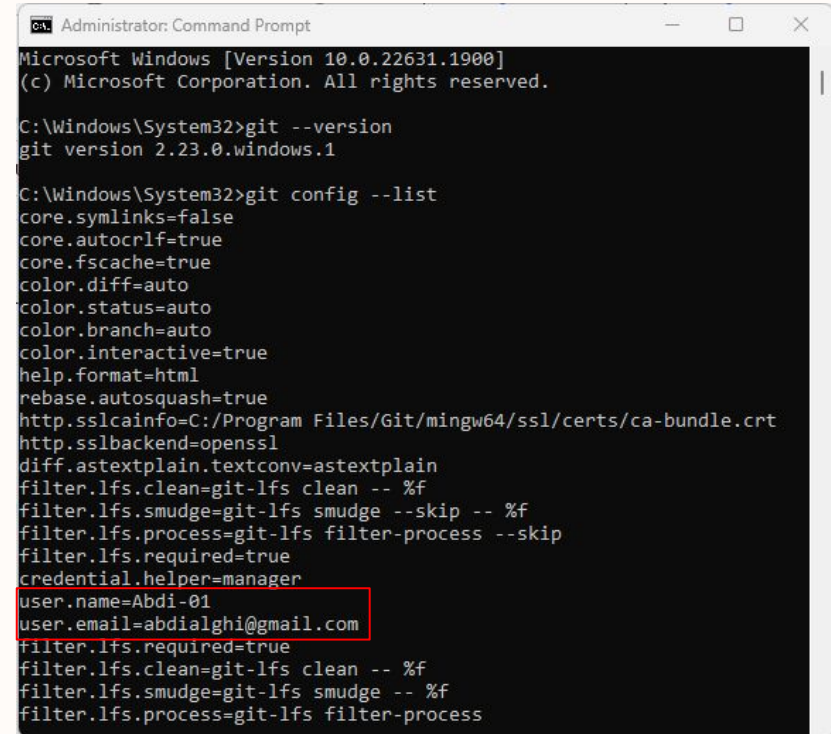
C:\Windows\System32>git --version
git version 2.23.0.windows.1

C:\Windows\System32>
```



# Check config list for connect to github

- Use `git config --list`
  - Please make sure user.name matches the GitHub account.
  - Please make sure user.email matches the GitHub account.
- If it doesn't exist yet, execute the following example command :
  - `git config --global user.name "student-01"`
  - `git config --global user.email "student@purwadhika.com"`



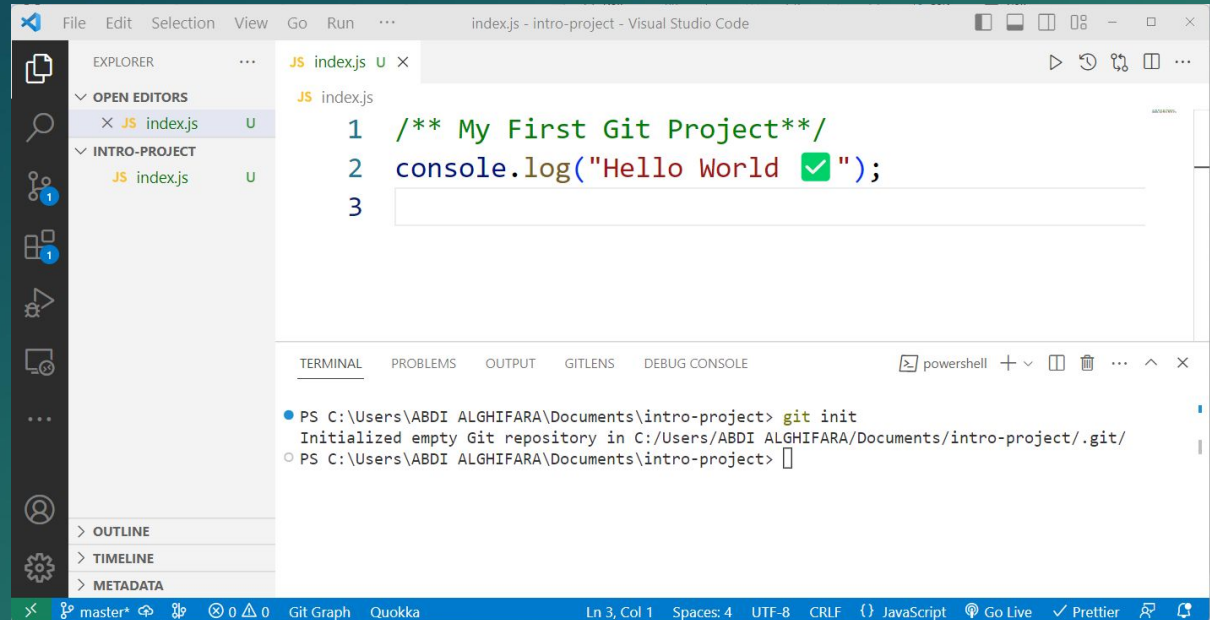
```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.22631.1900]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>git --version
git version 2.23.0.windows.1

C:\Windows\System32>git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
rebase.autosquash=true
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
http.sslbackend=openssl
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge --skip -- %f
filter.lfs.process=git-lfs filter-process --skip
filter.lfs.required=true
credential.helper=manager
user.name=Abdi-01
user.email=abdialghi@gmail.com
filter.lfs.required=true
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
```

# Create project directory

- Create a folder with a file named index.js.
- Try adding some programs to it.
- Then, open the terminal and execute the command "git init."



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a project directory named 'INTRO-PROJECT' containing a file 'index.js'. The editor window displays the content of 'index.js' with the following code:

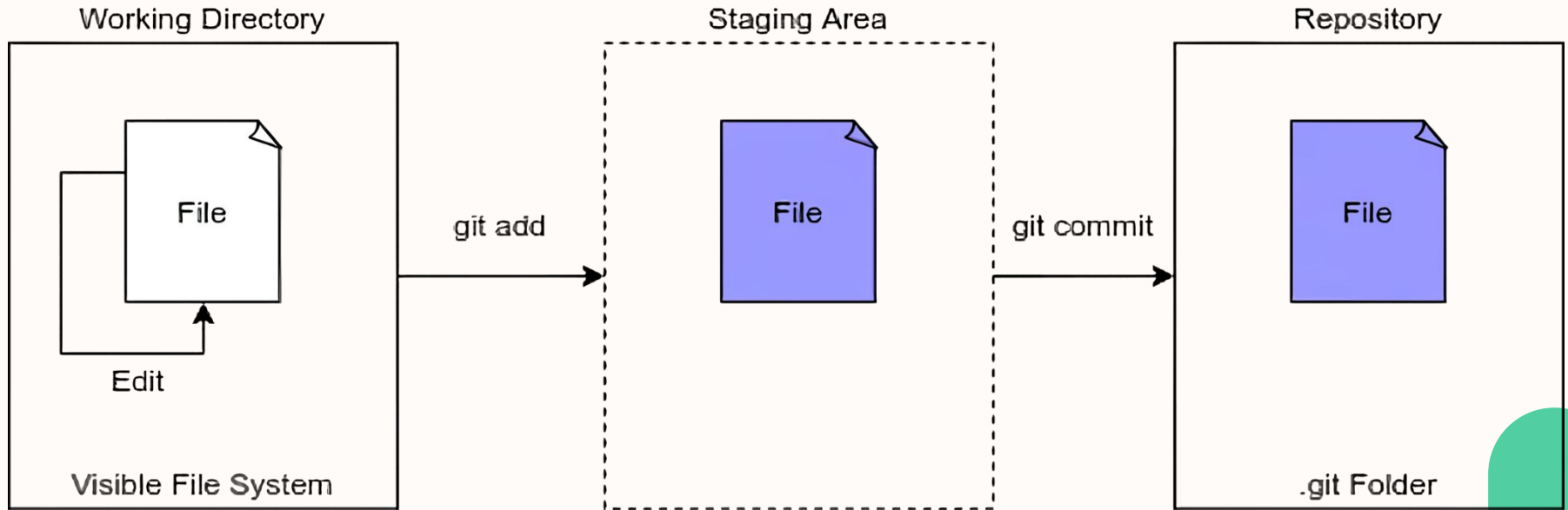
```
1 /** My First Git Project**/  
2 console.log("Hello World ✓");  
3
```

The terminal window at the bottom shows the execution of the 'git init' command in a PowerShell session:

```
PS C:\Users\ABDI ALGHIFARA\Documents\intro-project> git init  
Initialized empty Git repository in C:/Users/ABDI ALGHIFARA/Documents/intro-project/.git/  
PS C:\Users\ABDI ALGHIFARA\Documents\intro-project>
```

*git init* digunakan untuk melakukan inisialisasi pada directory agar dapat menggunakan fitur *git*

## Git record area



# Git status file

As you can see, after running the command "git init," the file we created earlier has the status U, which means untracked, indicating that the file is new. Some other statuses are:

**UNTRACKED**

indicating that the file is not being tracked by Git.

**ADDED**

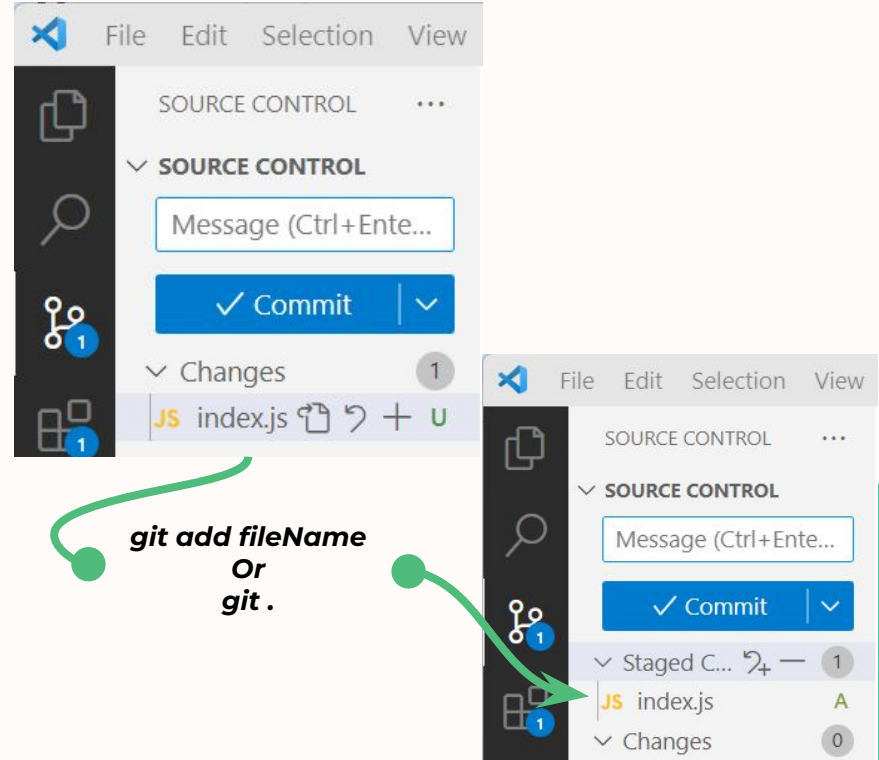
indicating that the file has been added to the repository.

**MODIFIED**

indicating that the file has been modified.

**DELETED**

indicating that the file has been deleted.



# Git command list

- When you want to save your work into Git:
  - *git add . or git add fileName: Add all changes in the current directory or specific file to the staging area.*
  - *git commit -m "feat: create file index.js": Commit the changes with a descriptive message.*
- When creating a new branch:
  - *git branch newBranch: Create a new branch.*
  - *Or git checkout -b branchName: Create a new branch and switch to it.*
- Ketika berpindah branch
  - *git checkout branchName*
- Ketika merubah nama branch saat ini
  - *git branch -m newBranchName*
- Ketika ingin melihat history perubahan yang sudah dicommit
  - *git log*
- Ketika ingin melihat history perubahan file tertentu
  - *git log fileName*
- Ketika ingin menghubungkan git local dengan github
  - *git remote add labelName urlRepoGithub*
    - Example : *git remote add origin https://github.com/student-01/intro-project.git*
- Ketika ingin melihat list remote github
  - *git remote -v*
- Ketika ingin menghapus remote github
  - *git remote remove labelName*
- Ketika meng-upload ke github
  - *git add . or git add fileName // jika belum*
  - *git commit -m "message" // jika belum*
  - *git push labelRemote branchName*
    - Example : *git push origin develop*

# Conventional Git Commit Message

When creating a commit message, make sure it represents what was done, using a formula :

`<type>(<scope>) : <subject>`

Type list :

- **build**: Build related changes (eg: npm related/ adding external dependencies)
- **chore**: A code change that external user won't see (eg: change to .gitignore file or .prettierrc file)
- **feat**: A new feature
- **fix**: A bug fix
- **docs**: Documentation related changes
- **refactor**: A code that neither fix bug nor adds a feature. (eg: You can use this when there is semantic changes like renaming a variable/ function name)
- **perf**: A code that improves performance
- **style**: A code that is related to styling
- **test**: Adding new test or making changes to existing test

*Example : feat (register) : start to create register page and register API*



# Exercise - Example in Pseudocode

Take a look at this example:

- Write a code to display the multiplication table of a given integer.
  - Example : Number → 9
  - Output :
    - 9 x 1
    - 9 x 2
    - ...
    - 9 x 10

Lets find out how to solve this problem through pseudocode!



Problem:

Write a code to display the multiplication table of a given integer.  
example given integer is 9

Hint:

1. Find out what is multiplication table
2. Multiplication table formula with example input 9
  - 9 x 1 = 9
  - 9 x 2 = 18
  - 9 x n = 9n
  - 9 x 10 = 90
3. Convert 9 as a constant variables, and set 10 as a limit of multiplication
  - const input = 9
  - const limit = 10
4. Define the loops rule
  - for(let i = 1; i <= limit; i++)

Solving:

1. define and assign variable for input and limit
  - const input = 9
  - const limit = 10
2. since we would like to create multiplication from 1 until 10 and multiply by input, lets create the loop starting with 1 and stop on 10. since we already define the limit as 10, lets use that variables
  - FOR(let i = 1; i <= limit; i++)
3. lets print out the multiplication based on input with index of loop starting with 1
  - OUTPUT input x i
4. close the looping statement after showing the output
  - END FOR

# Exercise - Debugging the Code

Try this: <https://pythontutor.com/visualize.html#mode=edit>

Lets write the code and see how it works. Take a look at the picture below. Click next button to see the process step by step. Left side screen shows our code, the right one show the process and output.

The screenshot displays the Python Tutor JavaScript ES6 visualization tool. The left pane shows the code being executed:

```
JavaScript ES6
known limitations

1 → const input = 9;
2   const limit = 10;
3
4   for( let i = 1; i <= limit; i++) {
5     console.log(`${input} x ${i} = ${input * i}`)
6   }
```

Below the code, a legend indicates that a green arrow points to the line just executed and a red arrow points to the next line to execute. A progress bar at the bottom shows the current step is 1 of 34. Navigation buttons include '<< First', '< Prev', 'Next >', and 'Last >>'.

The right pane is titled 'Print output (drag lower right corner to resize)' and contains a large empty box for the output. Below this box are two tabs: 'Frames' and 'Objects'.



# Exercise - Debugging the Code

Green arrow shows current code that executed, while the red one is the next code to be execute.

In this step, it would execute line 1 & 2 which is define the variables and assign the values.

JavaScript ES6  
[known limitations](#)

```
1 const input = 9;
2 → const limit = 10;
3
4 → for( let i = 1; i <= limit; i++) {
5   console.log(`${input} x ${i} = ${input * i}`)
6 }
```

[Edit this code](#)

→ line that just executed  
→ next line to execute

<< First < Prev Next > Last >>

Print output (drag lower right corner to resize)

Frames

Objects

Global frame

input	9
limit	10
i (block 1)	undefined

Step 3 of 34

# Exercise - Debugging the Code

i (block 1) define the starting looping position. i (block 2) define the current index looping positions.

JavaScript ES6  
[known limitations](#)

```
1 const input = 9;
2 const limit = 10;
3
4 → for( let i = 1; i <= limit; i++) {
5   console.log(`${input} x ${i} = ${input * i}`)
6 }
```

[Edit this code](#)

→ line that just executed  
→ next line to execute

Step 4 of 34

Print output (drag lower right corner to resize)

Frames	Objects
Global frame	
input	9
limit	10
i (block 1)	1
i (block 2)	1

# Exercise - Debugging the Code

This picture shows that line of code 5 is being executed. And generate an output in the right side panel. Every output would be written in the right panels.

JavaScript ES6  
[known limitations](#)

```
1 const input = 9;
2 const limit = 10;
3
4 for( let i = 1; i <= limit; i++) {
5   console.log(`${input} x ${i} = ${input * i}`)
6 }
```

[Edit this code](#)

→ line that just executed  
→ next line to execute

<< First < Prev Next > Last >>

Step 6 of 34

Print output (drag lower right corner to resize)

9 x 1 = 9

Frames

Objects

Global frame

Input	9
limit	10
i (block 1)	1
i (block 2)	1

# Exercise - Debugging the Code

Keep pressing next button to see the looping process. If looping conditional is no longer valid, it would go to line of code 6 which mean that is the end of looping progress.

JavaScript ES6  
[known limitations](#)

```
1  const input = 9;
2  const limit = 10;
3
4  → for( let i = 1; i <= limit; i++) {
5      console.log(`${input} x ${i} = ${input * i}`)
6  → }
```

[Edit this code](#)

→ line that just executed  
→ next line to execute

<< First < Prev Next > Last >>

Done running (34 steps)

Print output (drag lower right corner to resize)

9 x 6 = 54  
9 x 7 = 63  
9 x 8 = 72  
9 x 9 = 81  
9 x 10 = 90

Frames

Objects

Global frame

input

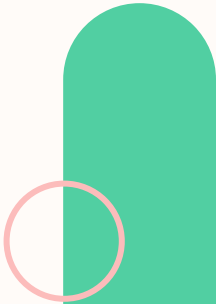
9

limit

10

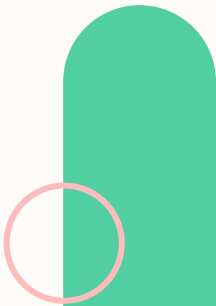
# Exercise

- Write a code to display the multiplication table of a given integer.
  - Example : Number  $\rightarrow 9$
  - Output :
    - $9 \times 1$
    - $9 \times 2$
    - ...
    - $9 \times 10$
- Write a code to check whether a string is a palindrome or not.
  - Example : 'madam'  $\rightarrow$  palindrome
- Write a code to convert centimeter to kilometer.
  - Example : 100000  $\rightarrow$  "1 km"
- Write a code to format number as currency (IDR)
  - Example : 1000  $\rightarrow$  "Rp. 1.000,00"
- Write a code to remove the first occurrence of a given "search string" from a string
  - Example : string = "Hello world", search string = "ell"  $\rightarrow$  "Ho world"
- Write a code to capitalize the first letter of each word in a string
  - Example : "hello world"  $\rightarrow$  "Hello World"



# Exercise

- Write a code to swap the case of each character from string
  - Example : 'The QuiCk BrOwN Fox' -> ' tHE qUlCk bRoWn fOX'
- Write a code to find the largest of two given integers
  - Example : num1 = 42, num2 = 27 → 42
- Write a conditional statement to sort three numbers
  - Example : num1 = 42, num2 = 27, num3 = 18 → 18, 27, 42
- Write a code that shows 1 if the input is a string, 2 if the input is a number, and 3 for others data type.
  - Example : "hello" → 1
- Write a code to change every letter a into \* from a string of input
  - Example : 'An apple a day keeps the doctor away' -> `\*n \*pple \* d\*y keeps the doctor \*w\*y`



# Thank You!

