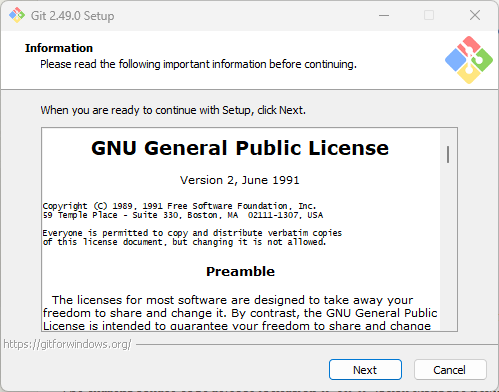
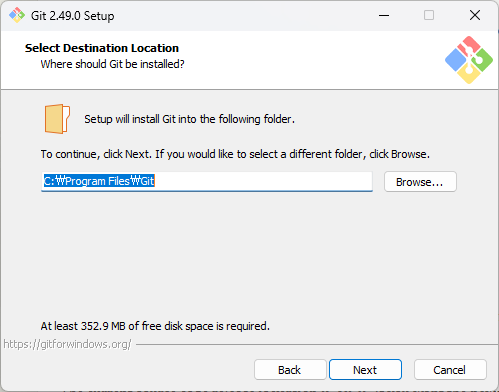
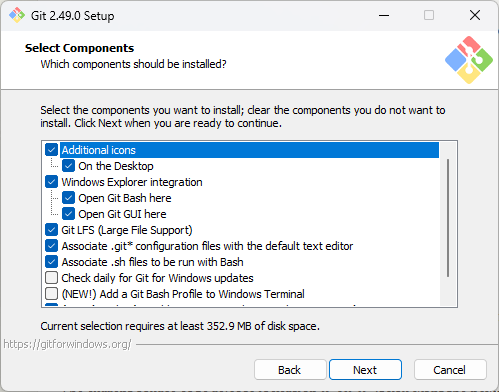
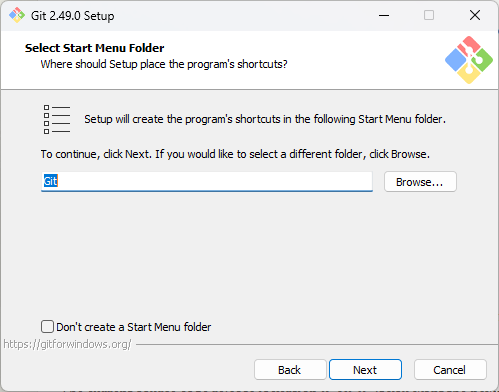
Git 설치

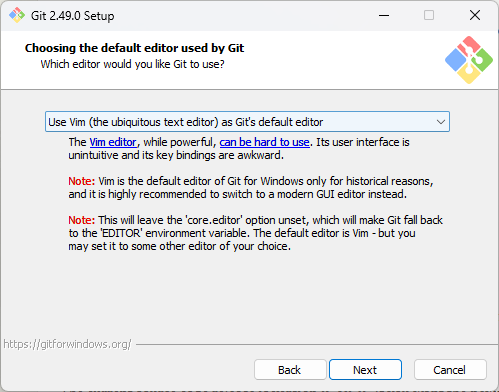
**Git 설치**

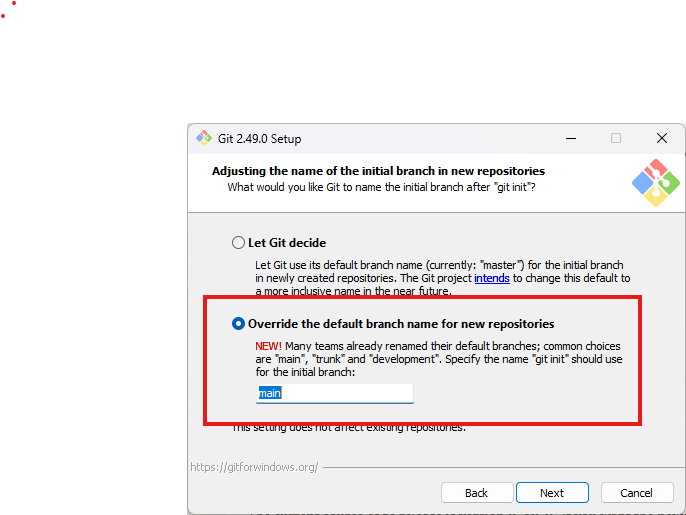


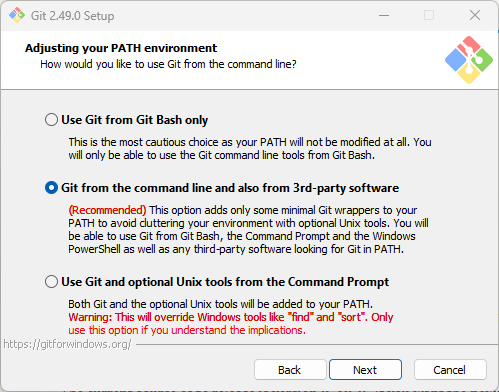


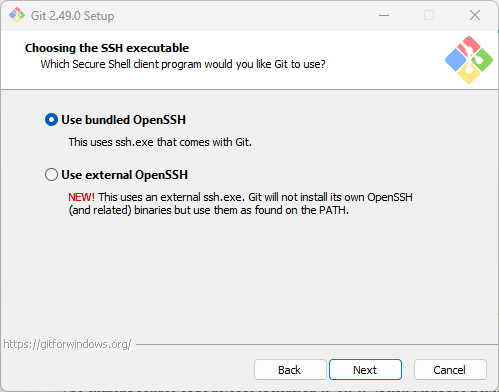


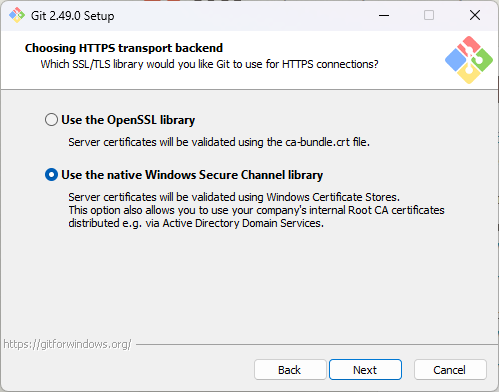


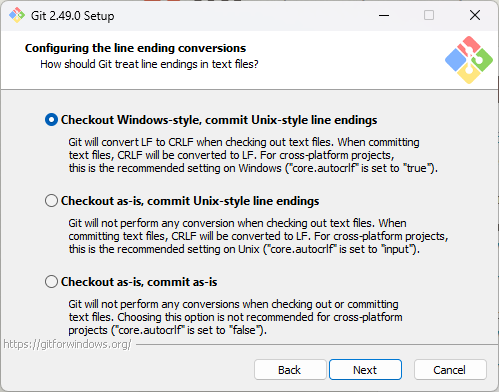


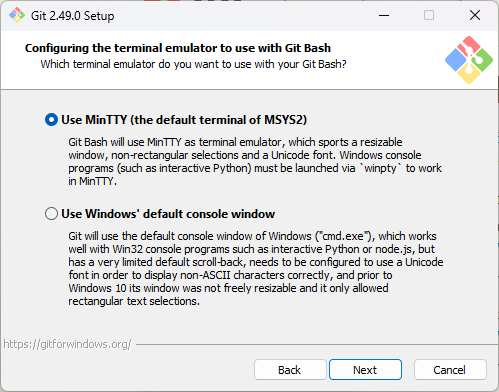


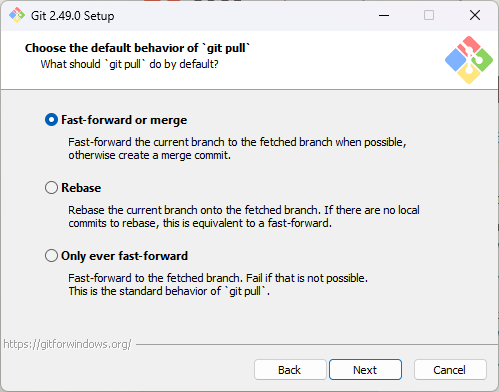


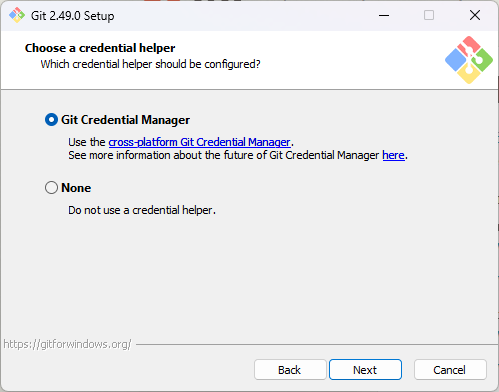


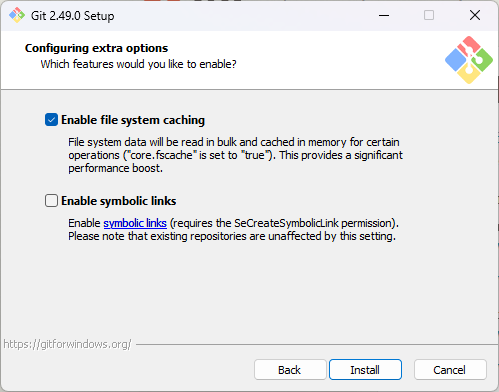


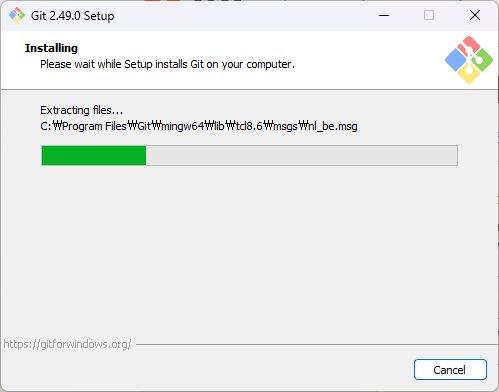


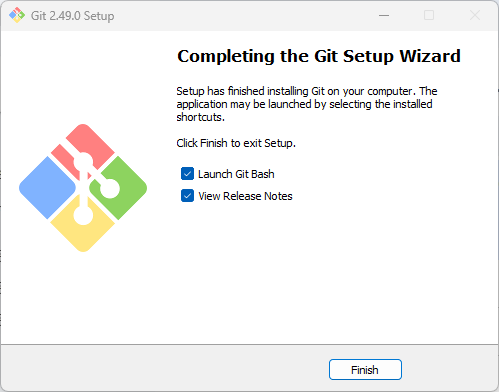












탭 2

Linux 기본 명령어

**Linux 기본 명령어**

Git 기본 명령어 (Linux 명령어 숙지)

1. pwd : 현재 내가 위치한 경로를 확인

2. ~ : 프롬프트, 현제 계정의 홈폴더

3. whoami : 로그온 한 계정명

4. cd <이동할폴더명> : 경로[폴더] 이동 하는 명령어

. : 현재 폴더

.. : 상위 폴더

/ : 절대 경로의 최상위 폴더 , (루트폴더)

5. clear : 화면을 깨끗하게 지움 (Windows 명령 프롬프트 : cls)

6 명령어 호출 : 윗쪽 화살표, 아랫쪽 화살표

7. ls : 폴더의 파일이나 폴더를 출력

-a : 숨어있는 폴더 ( .폴더명), 숨어있는 파일 (.파일명) 을 출력

-l : 폴더나 파일의 자세한 정보를 출력

ls -la

8. mkdir : 폴더 생성 하는 명령어

9. touch : 파일 생성하는 명령어

10. alias : 명령어의 별칭을 사용해서 실제 명령어로 작동

alias la = 'ls -la'

11. history : 지금까지 작업했던 명령어를 출력

12. > : 리다이렉션, 출력 방향을 바꿈 , 덮어쓰기

>> : 이어쓰기

13. cat : 파일의 내용을 출력

-n : 라인 번호를 넣어서 출력

14. 경로

- 상대 경로 : 현재 내가 위치한 폴더를 기준

- 절대 경로 : /

15. cp : 복사

- 파일 복사

cp <복사할원본경로> <복사할대상>

cp /c/temp/aaa.txt .

- 폴더 복사 : 폴더 하위의 파일까지 복사

cp -r <복사할원본경로> <복사할대상>

cp -r /C/WorkSpace/ .

16. 폴더 용량 보기

du -sh /C/WorkSpace

17. 삭제 : <== 주의 <복사한것을 삭제>

-- 파일 삭제

rm <삭제할원본파일명>

-- 폴더 삭제 <== <주의>

rm -rf <삭제할원본폴더명>

18. 특정 폴더의 모든 파일을 복사

cp -r <원본폴더>/\* .

19. 특정 폴더의 모든 내용을 삭제

rm -rf ./\*

20.: : 파일 , 폴더 이동, 폴더, 파일의 이름변경

<== 대상 폴더가 존재하면 이동, 그렇지 않으면 이름변경

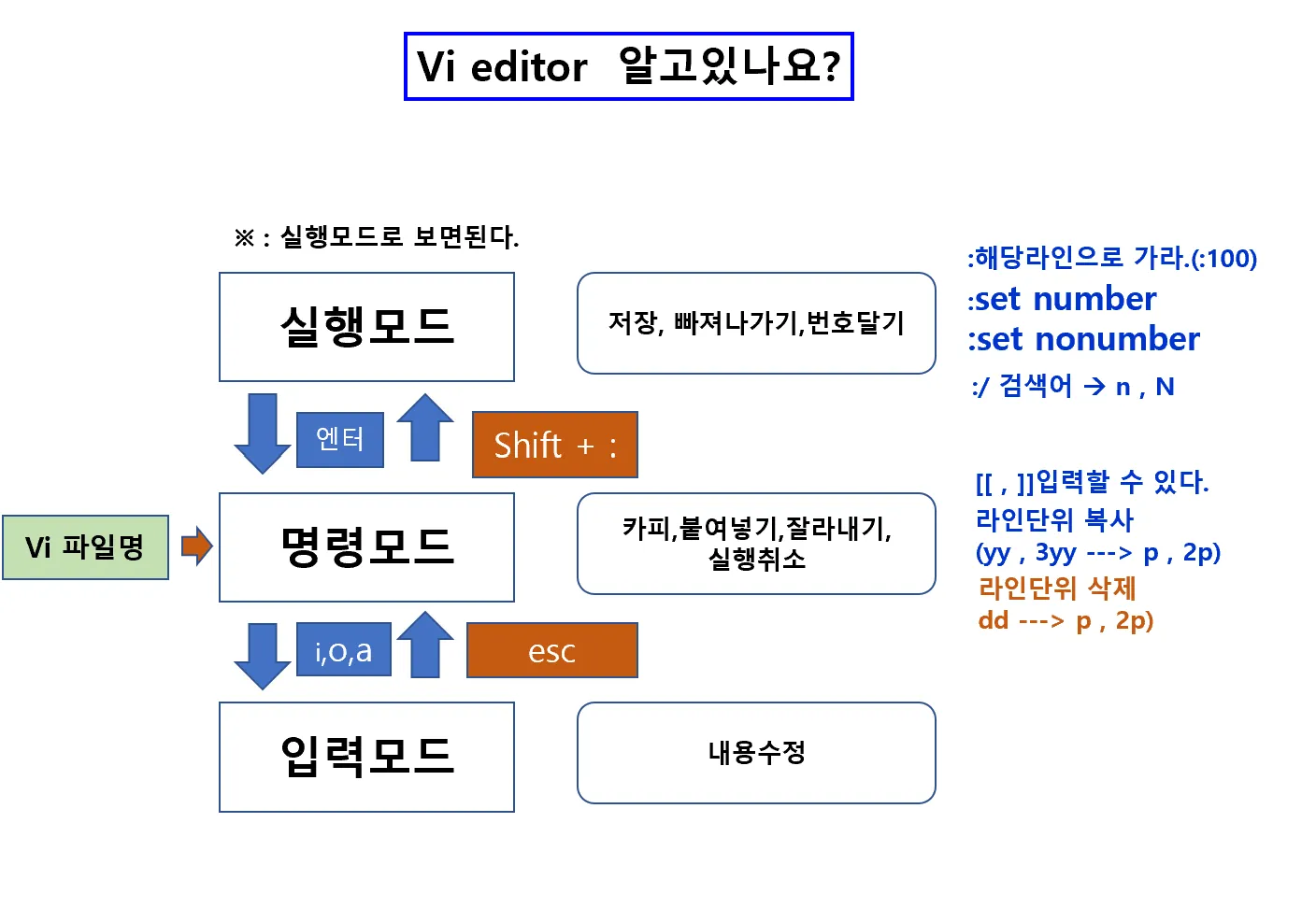
mv <원본> <대상>

21. | : 파이프 , 명령어와 명령어를 연결해서 출력

netstat -na

netstat -na | grep 80

22. echo : 내용을 화면에 출력 사용



| **🧑‍🎓 제가 진짜 사용하는 vi 명령어들 알짜 모음> 명령모드 (esc 눌렀을때, vi 처음 들어갔을때)**   1. **파일의 끝으로 이동할때는 : 1G** 2. **한줄 잘라내기 : 1dd** 3. **세줄 잘라내기 : 3dd** 4. **한줄복사 : 1yy** 5. **두줄복사 : 2yy** 6. **붙여넣기 : p** 7. **한글자 삭제 : x** 8. **단어 삭제 : dw** 9. **실행취소 : u** 10. **줄의 맨 앞 : o** 11. **줄의 맨 뒤 : shift+ $ - 문장의 맨뒤로 가라, shift + G - 파일의 맨밑에 줄로가라 (이거 개꿀) 마지막행 모드 (esc -> : 눌렀을때)(명령어 뒤에 !를 붙이면 강제로 수행하니까, vi가 궁시렁 궁시렁 거리면" ! "붙여서 실행 하면 됩니다.)** 12. **[[ : 맨 앞으로 , ]] : 맨 뒤로** 13. **저장만 : w** 14. **종료만 : q** 15. **저장 후 종료 : wq** 16. **강제 저장 : wq!** 17. **라인 번호좀 보자 : set nu** |
| --- |

**명령 모드에서의 더 많은 명령어**

| **1. 모드 설명**  명령 모드 (Command Mode): vi 에디터에서 기본적으로 활성화되는 모드입니다. 이 모드에서는 텍스트를 수정하거나 파일을 저장, 종료하는 명령을 실행할 수 있습니다.  입력 모드 (Insert Mode): 텍스트를 입력하는 모드입니다. 이 모드에서는 키보드 입력이 실제 텍스트로 기록됩니다.  **2. 기본적인 모드 전환**  입력 모드로 전환:  i: 커서 앞에 텍스트 삽입  I: 현재 줄의 맨 앞에 텍스트 삽입  a: 커서 뒤에 텍스트 삽입  A: 현재 줄의 맨 끝에 텍스트 삽입  o: 커서 아래에 새 줄 삽입  O: 커서 위에 새 줄 삽입  명령 모드로 전환:  Esc: 입력 모드에서 명령 모드로 돌아가기  **3. 텍스트 이동**  한 글자씩 이동:  h: 왼쪽으로 한 글자 이동  j: 아래로 한 줄 이동  k: 위로 한 줄 이동  l: 오른쪽으로 한 글자 이동  단어 단위로 이동:  w: 다음 단어의 시작으로 이동  e: 현재 단어의 끝으로 이동  b: 이전 단어의 시작으로 이동  줄 이동:  0: 현재 줄의 맨 앞  ^: 현재 줄의 첫 번째 비어 있지 않은 문자로 이동  $: 현재 줄의 맨 끝  파일 내에서 이동:  gg: 파일의 첫 번째 줄로 이동  G: 파일의 마지막 줄로 이동  nG: n번째 줄로 이동  **4. 텍스트 수정**  삭제:  x: 커서 위치의 한 글자 삭제  dw: 커서 위치에서부터 다음 단어까지 삭제  dd: 현재 줄 삭제  D: 커서부터 현재 줄의 끝까지 삭제  d$: 커서부터 줄 끝까지 삭제  복사 & 붙여넣기:  yy: 현재 줄을 복사  p: 복사한 내용을 커서 뒤에 붙여넣기  P: 복사한 내용을 커서 앞에 붙여넣기  수정:  r: 커서 위치의 문자 하나를 바꿈  R: 커서 위치에서 여러 문자를 덮어씀  cw: 커서부터 단어의 끝까지 변경 (입력 모드로 전환됨)  cc: 현재 줄을 변경 (입력 모드로 전환됨)  **5. 검색 및 치환**  검색:  /문자열: 문자열을 검색 (다음 항목으로 이동)  ?문자열: 문자열을 역방향으로 검색  n: 다음 검색 결과로 이동  N: 이전 검색 결과로 이동  치환:  :%s/old/new/g: 전체 파일에서 old를 new로 치환  :s/old/new/g: 현재 줄에서 old를 new로 치환  :s/old/new/gc: 치환하기 전에 확인  **6. 파일 저장 및 종료**  저장:  :w: 파일 저장  :w 파일명: 다른 이름으로 저장  종료:  :q: 종료  :q!: 저장하지 않고 종료  :wq: 저장하고 종료  ZZ: 저장하고 종료 (입력 모드에서)  **7. 기타 유용한 명령어**  파일 정보:  :e 파일명: 다른 파일 열기  :w!: 파일 덮어쓰기  :set nu: 줄 번호 표시  :set nonu: 줄 번호 숨기기  :syntax on: 구문 강조 표시  :syntax off: 구문 강조 표시 끄기  Undo / Redo:  u: 마지막 변경 취소 (Undo)  Ctrl + r: Undo를 취소하고 다시 실행 (Redo)  **8. 기타**  복잡한 명령어 연습: vi는 여러 가지 복잡한 명령어를 조합할 수 있는 강력한 텍스트 편집기입니다. 예를 들어, d3w는 커서부터 3개의 단어를 삭제하는 명령입니다.  위 명령어들은 vi를 사용할 때 매우 기본적인 것들입니다. 사용해보면서 익숙해지면 더욱 다양한 고급 기능들을 활용할 수 있습니다! |
| --- |

GitHub 회원 가입

GitHub 회원 가입

생성 날짜 : 2025.04.07

생성자 : 박장우

| 1. **https://github.com/** 사이트 접속 및 **Sign up** [클릭]  텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진  AI가 생성한 콘텐츠는 부정확할 수 있습니다. |
| --- |

| **2. GitHub에 회원 가입**  텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  AI가 생성한 콘텐츠는 부정확할 수 있습니다.  - Email  자신의 메일 주소를 넣습니다. 회원 가입 후 메일로 인증 메시지가 도착하고 **메일 인증 후 사용 가능**합니다. 정확한 정보를 입력하세요.  - Password  암호를 복잡성을 요구 합니다. 복잡한 암호를 넣으세요. **대문자+소문자+특수문자+숫자** 4 항목 중 3가지 이상이 조합 되어야 합니다. **최소 8자 이상** 넣어야 합니다.  - Username  GitHub에서 사용한 자신의 ID 입니다.  - Your country  **Korea, South** |
| --- |

| **3. Verify your account**    이 퍼즐을 풀어서 귀하가 인간이라는 것을 알 수 있게 해주십시오.  - **시각 퍼즐**  - 오디오 |
| --- |

| **4. 메일 인증**    **자신의 메일 주소 확인** |
| --- |

| **5. GitHub 로그인** |
| --- |

| **6.** |
| --- |

| **7. 로그인 완료 및 확인** |
| --- |

| **8. 로그인 정보 확인 및 Sign Out [Login Out]** |
| --- |

| **9. Sign in [Login]** |
| --- |

GitHub Repository 생성

GitHub Repository 생성

생성 날짜 : 2025.04.07

생성자 : 박장우

| **1. 로그인후 Create Repository [클릭] 또는 URL에 https://github.com/new 입력** |
| --- |

| **2.** |
| --- |

| **3.** |
| --- |

| **4.** |
| --- |

| **5.** |
| --- |

| **6.** |
| --- |

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

Git 명령어

**Git 명령어**

**0. Git 용어 정리**

* **워킹 디렉토리 (Working Directory) - Git 이 관리하는 디렉토리, (.git)** 
  + **초기화된 폴더 (.git)**
  + **git init (특정 폴더를 초기화, .git 폴더가 생성)**
* **스테이지** 
  + **commit을 하기 위한 공간**
  + **git add**
* **레파지토리 (Repository) -** 
  + **Commit 된 데이터가 저장되어 있음.**
  + **git commit**

**1. 기본 설정**

사용자 정보 설정:

git config --global user.name "사용자 이름" : 사용자 이름 설정

git config --global user.email "이메일 주소" : 이메일 주소 설정

설정 확인:

git config --list : 현재 설정된 Git 설정 목록 보기

**2. 저장소 초기화 및 클론**

로컬 저장소 초기화:

git init: 현재 디렉토리를 Git 저장소로 초기화

원격 저장소 복제:

git clone <repository URL>: 원격 저장소를 로컬로 복제

**3. 파일 상태 확인**

파일 상태 확인:

git status: 현재 작업 디렉토리와 스테이지의 상태를 확인

**4. 파일 추적 및 커밋**

파일 추적 및 스테이지:

git add <파일명>: 파일을 스테이지 영역에 추가

git add .: 현재 디렉토리 내 모든 변경된 파일을 스테이지

커밋 생성:

git commit -m "커밋 메시지": 스테이지된 파일들을 커밋 (메시지 필수)

git commit -a -m "커밋 메시지": 변경된 모든 파일을 자동으로 스테이징하고 커밋

**5. 커밋 확인**

커밋 로그 확인:

git log: 커밋 로그 출력 (기본적으로 최신 커밋부터)

git log --oneline: 간략한 형태로 커밋 로그 출력

**6. 원격 저장소와의 연동**

원격 저장소 추가:

git remote add origin <원격 저장소 URL>: 원격 저장소 추가

원격 저장소 상태 확인:

git remote -v: 설정된 원격 저장소 URL 확인

원격 저장소에서 변경 사항 가져오기:

git fetch: 원격 저장소에서 최신 변경 사항을 가져오기

git pull: 원격 저장소의 변경 사항을 가져오고 병합

원격 저장소에 변경 사항 푸시:

git push origin <브랜치명>: 로컬 브랜치를 원격 저장소에 푸시

**로컬 레파지토리를 생성하면서서 내용을 원격 레파지토리로 push**

| **git init**  **git add .**  **git commit -m “커밋 메세지”**  **git remote add origin “원격레파지토리 주소”**  **git push origin main** |
| --- |

**로컬 레파지토리가 존재할때 수정된 내용을 원격 레파지토리로 push**

| **git add .**  **git commit -m “커밋 메세지”**  **git push origin main** |
| --- |

https://www.bing.com/

[**https://www.bing.com/images/create**](https://www.bing.com/images/create) **⇐ 하루100개 이미지 생성 , Chat GPT (3개 생성)**

[**https://claude.ai/**](https://claude.ai/) **⇐ 코딩 AI**

[**https://gamma.app/ko**](https://gamma.app/ko) **⇐ PPT 생성 AI , 10 페이지 무료**

GitHub 의 레파지토리의 내용을 모두 집[Home] 컴퓨터로 가지고 오기

git clone “원격레파지토리주소”

Branch (가지) : 테스트, 협업

* 기본 브렌치 : main ⇐ master (옛날)
* 테스트 : 특정 시점의 커밋된 정보를 확인

git branch dev1 ⇐ 기존의 브레치의 내용을 복사해서 dev1 브렌치 생성

git branch ⇐ 브렌치 정보를 출력, 내가 지금 사용하는 브렌치를 확인

git checkout dev1 ⇐ dev1 브렌치로 이동

git checkout <특정커밋의해시코드> ⇐ 특정 커밋 시점으로 이동 , 해드가 망가짐. 테스트할 브랜치를 생성후 실습함.

git branch -D dev1 ⇐ dev1 브렌치 삭제 , 해당 브랜치에 접속해 있으면 삭제 안됨

README.md

* 해당 프로잭트를 설명하는 파일
* 프로잭트 Root 폴더에 존재해야함.
* 파일명이 README.md
* MarkDown 문법으로 작성을 해야함.

gitignore.io ⇐ github에 전송할 필요가 없는 파일을 전송하지 않도록 설정하는 파일내용을 자동으로 만듬

운영체제 : windows, mac, linux

IDE : vscode, eclipse, intellij

개발언어 : java, c, c++,

.gitignore

| # Created by https://www.toptal.com/developers/gitignore/api/windows,eclipse,java  # Edit at https://www.toptal.com/developers/gitignore?templates=windows,eclipse,java  ### Eclipse ###  .metadata  bin/  tmp/  \*.tmp  \*.bak  \*.swp  \*~.nib  local.properties  .settings/  .loadpath  .recommenders  # External tool builders  .externalToolBuilders/  # Locally stored "Eclipse launch configurations"  \*.launch  # PyDev specific (Python IDE for Eclipse)  \*.pydevproject  # CDT-specific (C/C++ Development Tooling)  .cproject  # CDT- autotools  .autotools  # Java annotation processor (APT)  .factorypath  # PDT-specific (PHP Development Tools)  .buildpath  # sbteclipse plugin  .target  # Tern plugin  .tern-project  # TeXlipse plugin  .texlipse  # STS (Spring Tool Suite)  .springBeans  # Code Recommenders  .recommenders/  # Annotation Processing  .apt\_generated/  .apt\_generated\_test/  # Scala IDE specific (Scala & Java development for Eclipse)  .cache-main  .scala\_dependencies  .worksheet  # Uncomment this line if you wish to ignore the project description file.  # Typically, this file would be tracked if it contains build/dependency configurations:  #.project  ### Eclipse Patch ###  # Spring Boot Tooling  .sts4-cache/  ### Java ###  # Compiled class file  \*.class  # Log file  \*.log  # BlueJ files  \*.ctxt  # Mobile Tools for Java (J2ME)  .mtj.tmp/  # Package Files #  \*.jar  \*.war  \*.nar  \*.ear  \*.zip  \*.tar.gz  \*.rar  # virtual machine crash logs, see http://www.java.com/en/download/help/error\_hotspot.xml  hs\_err\_pid\*  replay\_pid\*  ### Windows ###  # Windows thumbnail cache files  Thumbs.db  Thumbs.db:encryptable  ehthumbs.db  ehthumbs\_vista.db  # Dump file  \*.stackdump  # Folder config file  [Dd]esktop.ini  # Recycle Bin used on file shares  $RECYCLE.BIN/  # Windows Installer files  \*.cab  \*.msi  \*.msix  \*.msm  \*.msp  # Windows shortcuts  \*.lnk  # End of https://www.toptal.com/developers/gitignore/api/windows,eclipse,java |
| --- |