Linux&git_보고서_인턴_20기_2025402040_컴퓨터정보공학부_노기문

목차

1. 리눅스&우분투, Git&GitHub란?

-리눅스(Linux)

리눅스(Linux)는 리누스 토르발스가 창시한 Unix 계열운영체제제품군으로 다중 사용자, 다중 작업, 이식성, 보안성 등을 주요 특징으로 합니다. 운영체제 커널 및 해당 커널을 탑재한 다양한 배포판을 아우른다. 오픈 소스를 표방하는 대표적인 운영체제입니다.

리눅스는 크게 커널과 배포판(Distribution) 개념으로 나뉩니다. 커널은 하드웨어 제어와 시스템 자원 관리를 담당하며, 배포판은 커널에 더해 패키지 관리 시스템, 데스크톱 환경, 각종 유틸리티를 포함하여 사용자가 편리하게 사용할 수 있도록 구성한 형태입니다. 대표적인 배포판에는 Ubuntu, Debian, Fedora, CentOS, Arch Linux 등이 있습니다.

-우분투(Ubuntu)

우분투(Ubuntu)는 가장 널리 사용되는 리눅스 배포판 중 하나로, 영국의 소프트웨어 회사 Canonical Ltd.에서 관리하며 데비안*데비안이란 커뮤니티인 데비안 프로젝트에서 개발하고 있는 리눅스 배포판. 을 기반으로 합니다. 사용자 친화성과 안정성 덕분에 교육, 연구, 기업 환경에서 폭넓게 활용됩니다.

주요 특징:

쉬운 설치와 업데이트

그래픽 기반 설치 프로그램을 제공하여 초보자도 쉽게 설치할 수 있으며, apt 패키지 관리자를 통해 간단히 프로그램 설치 및 업데이트가 가능합니다.

LTS (Long Term Support)

장기 지원(LTS) 버전은 5년간 보안 업데이트와 유지보수를 제공하여 서버 및 기업 환경에서 널리 사용됩니다.

데스크톱 환경

기본적으로 GNOME 데스크톱 환경을 제공하며, KDE Plasma, Xfce, LXQt 등 다양한 환경으로 변경할 수 있습니다.

소프트웨어 생태계

무료 오픈소스 소프트웨어를 풍부하게 제공하며, Snap 스토어를 통해 손쉽게 애플리케이션을 설치할 수 있습니다.

보안과 안정성

기본적으로 방화벽과 사용자 권한 관리가 잘 갖추어져 있으며, 오픈소스 커뮤니티와 Canonical에서 보안 패치를 신속히 제공합니다.

서버 환경 활용

Ubuntu Server는 GUI 없이 동작하며 클라우드, 데이터베이스, 웹 서버 운영에 적합합니다. AWS, Azure, GCP 같은 클라우드 서비스에서도 표준 이미지로 제공됩니다.

-깃(Git) 과 깃허브(GitHub)

깃(Git)

Git은 2005년 리누스 토르발스 씨가 리눅스 커널 개발을 위해 만든 분산형 버전 관리 시스템(Distributed Version Control System, DVCS) 입니다. 소스 코드의 변경 이력을 기록하고 협업을 용이하게 하여 현대 소프트웨어 개발에서 사실상 표준 도구로 자리 잡았습니다.

Git의 특징:

분산형 구조

모든 사용자가 전체 저장소의 복사본을 로컬에 보관하므로 중앙 서버가 없어도 작업할 수 있습니다.

빠른 성능

브랜치 생성, 병합, 커밋 같은 작업이 매우 빠르게 처리됩니다.

브랜치 관리

독립적인 개발 흐름을 만들 수 있어 기능 개발, 버그 수정, 실험을 안전하게 수행할 수 있습니다.

데이터 무결성

Git은 SHA-1 해시 알고리즘을 사용하여 데이터 위변조를 방지합니다.

협업 효율성

GitHub, GitLab, Bitbucket 같은 플랫폼과 결합하면 이슈 관리, 코드 리뷰, CI/CD까지 지원되어 협업 환경에 최적화됩니다.

깃허브(GitHub)

GitHub는 Git을 기반으로 한 분산 버전 관리 및 협업 플랫폼입니다. 2008년에 설립되었으며, 현재는 Microsoft가 소유하고 있습니다. 전 세계 개발자들이 소스 코드를 공유하고 협업하는 가장 대표적인 서비스로 자리잡았습니다. GitHub는 단순히 Git 저장소를 원격으로 관리하는 것뿐 아니라, 이슈 관리, 코드 리뷰, 프로젝트 관리, CI/CD(지속적 통합 및 배포) 기능까지 지원하는 통합 개발 플랫폼입니다.

GitHub의 특징:

원격 저장소 제공

개발자는 로컬 저장소에서 작업한 내용을 GitHub 원격 저장소에 업로드(Push)하여 안전하게 보관할 수 있습니다.

협업 기능

여러 명이 동시에 프로젝트에 참여할 수 있도록 Pull Request, 코드 리뷰, 병합(Merge) 기능을 제공합니다.

프로젝트 관리

GitHub Issues, Projects(칸반 보드 형식), Wiki 기능을 통해 버그 추적, 할 일 관리, 문서화를 체계적으로 할 수 있습니다.

오픈소스 생태계

수많은 오픈소스 프로젝트가 GitHub에서 관리되며, 개발자들은 누구나 해당 코드를 포크(Fork)하거나 기여(Contribute)할 수 있습니다.

자동화와 배포

GitHub Actions 기능을 통해 빌드, 테스트, 배포 과정을 자동화할 수 있습니다. CI/CD 파이프라인을 무료로 구성할 수 있다는 점에서 많은 프로젝트가 활용합니다.

보안 기능

코드 보안 분석, 의존성 취약점 경고, 비밀키 관리(Secrets) 등 안전한 개발 환경을 제공합니다.

GitHub 기본 개념:

저장소(Repository)

프로젝트 단위의 작업 공간으로, 소스 코드와 이력(commit history)을 포함합니다. 공개 저장소(Public)와 비공개 저장소(Private)로 구분됩니다.

포크(Fork)

다른 사람의 저장소를 복제하여 내 계정에서 독립적으로 개발할 수 있도록 하는 기능입니다. 오픈소스 기여 과정에서 자주 사용됩니다.

풀 리퀘스트(Pull Request)

포크나 별도 브랜치에서 수정한 내용을 원본 저장소에 반영해 달라고 요청하는 절차입니다. 협업 시 코드 리뷰와 토론의 중심이 됩니다.

이슈(Issue)

버그 보고, 기능 제안, 질문 등을 기록하는 기능입니다. 라벨(Label), 마일스톤(Milestone), 담당자(Assignee)를 지정할 수 있어 체계적인 프로젝트 관리가 가능합니다.

액션(Actions)

워크플로우 자동화를 지원하는 기능으로, 빌드, 테스트, 배포를 자동으로 실행할 수 있습니다.

GitHub와 Git의 관계:

Git은 버전 관리 도구입니다. (로컬에서 코드 변경 이력을 관리)

GitHub는 Git 저장소를 클라우드에 올려 협업과 공유를 가능하게 하는 플랫폼입니다.

따라서 Git은 도구, GitHub는 서비스라고 이해하시면 됩니다.

2. 우분투 및 Git 설치

2-1. 우분투 설치:

(1) 파티션 분할

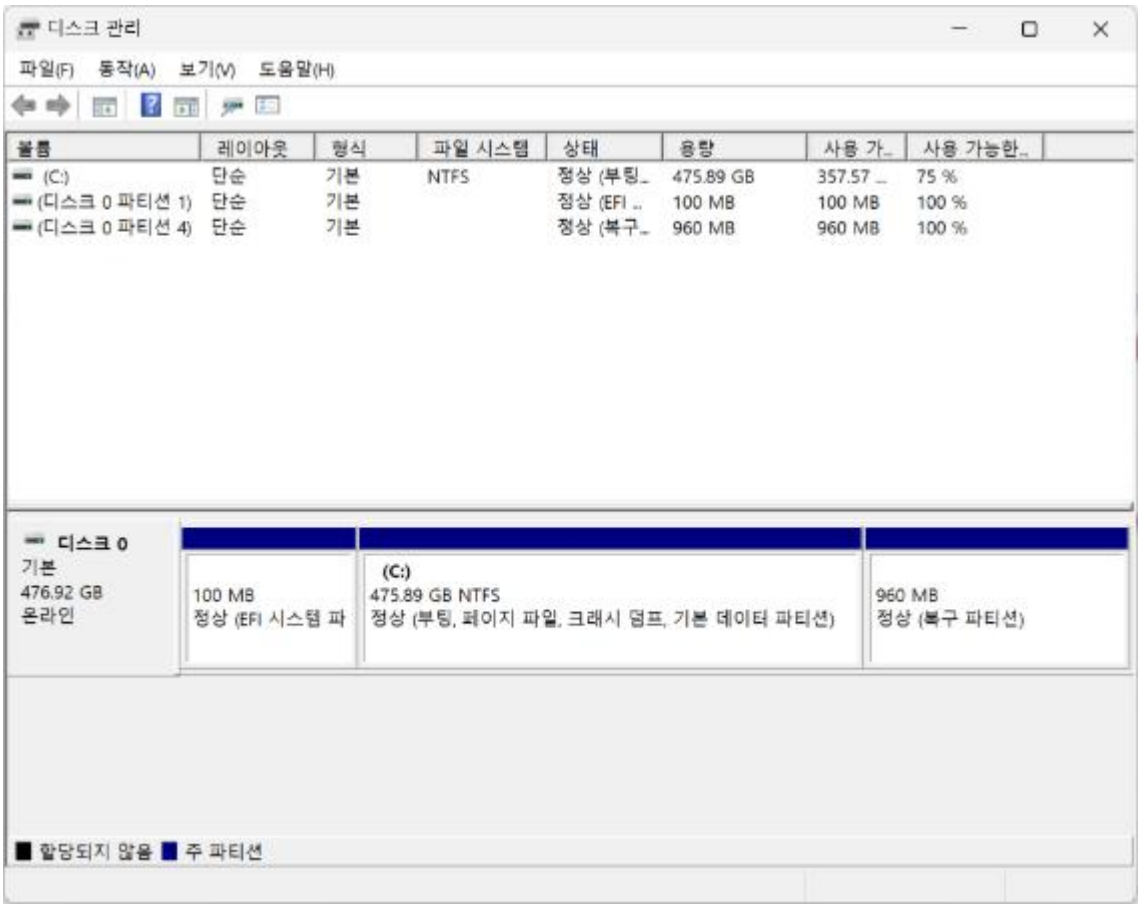
우분투를 설치하기 위해서는 설치 공간을 미리 확보해야 합니다. 방법은 크게 두 가지입니다.

1. 기존에 설치된 OS(예: Windows)를 완전히 지우고 우분투를 단독으로 설치하는 방법

2. 기존 OS는 그대로 두고 디스크를 분할하여 우분투를 함께 설치하는 방법(듀얼 부팅)

1번 방식은 인터넷에 자료가 많으므로 참고하시면 되고, 여기서는 2번 방법(듀얼 부팅)을 중심으로 설명드리겠습니다.

먼저 Windows 환경에서 디스크 관리에 들어갑니다.



여기서 원하는 디스크(예: 475GB 파티션)를 선택하여 우클릭 → 볼륨 축소를 클릭합니다. 이후 우분투 설치를 위한 여유 공간을 분할합니다. 최소 50GB 이상을 권장하지만, 실습과 다양한 프로그램 설치를 고려하면 약 200GB 정도를 권장합니다.

(2) 우분투 다운로드

Ubuntu 공식 웹사이트에서 최신 LTS 버전인 Ubuntu 22.04.5 LTS (Jammy Jellyfish) 를 다운로드합니다.

LTS(Long Term Support) 버전은 5년간 보안 업데이트와 유지보수가 제공되므로 안정적으로 사용하실 수 있습니다.

(3) 부팅 USB 제작

우분투를 설치하려면 부팅 가능한 USB가 필요합니다.

Rufus 공식 사이트에서 프로그램을 다운로드하여 실행합니다.

장치 선택: 설치할 USB 선택

부트 선택: 앞서 다운로드한 Ubuntu ISO 이미지 지정

파티션 방식: GPT(UEFI 기반) 또는 MBR(BIOS 기반) 선택

설정을 마친 후 “시작”을 누르면 부팅 가능한 USB가 제작됩니다.

(4) BIOS 설정

제작된 USB를 컴퓨터에 꽂은 후, 전원을 켜자마자 BIOS(또는 UEFI) 에 진입합니다.

진입 키: 제조사별로 F2, F10, F12, DEL 등이 있으며 부팅 시 화면 하단에 표시됩니다.

Boot 메뉴에서 USB를 부팅 우선순위 1번으로 설정합니다.

(5) 우분투 설치 진행

부팅 후 "Ubuntu" 옵션이 보이면 선택하고 ENTER를 누릅니다.

만약 검은 화면에서 멈춘다면 “Ubuntu (safe graphics)”를 선택합니다.

설치 화면에서 다음과 같이 진행합니다.

Install Ubuntu 선택 => 언어: 기본 영어 유지 후 Continue => 와이파이 연결 선택 후 Continue => 설치 유형: Something else 선택 => 디스크 파티션 설정: Free space 선택 후 + 버튼 클릭 => Size: 분할한 공간 입력 => Type: Logical, Use as: Ext4 journaling file system, Mount point: / 입력 후 OK => 시간대 선택: Seoul => 사용자 계정 설정; 이름: 간단하게 (예: ngm), PC 이름: 간단하게 (예: ngm) 비밀번호: 간단히 설정 (예: 1) => “Log in automatically” 체크

설치를 마치고 재부팅 메시지가 뜨면, USB를 제거한 뒤 ENTER를 누릅니다.

이후 정상적으로 Ubuntu 바탕화면이 보이면 설치가 완료된 것입니다.

만약 설치 중 예상치 못한 오류 화면이 나오면 노트북을 강제로 종료한 뒤 다시 처음부터 재시도하셔도 됩니다.

2-2. Git 설

Ubuntu가 정상 설치되었다면, 다음은 개발에 필수적인 Git을 설치합니다. Git은 소스 코드 버전 관리와 협업에 반드시 필요한 도구입니다.

(1) Git 설치

터미널을 열고 다음 명령어를 순서대로 입력합니다.

sudo apt-get update : 패키지 목록을 최신 상태로 갱신

sudo apt-get upgrade: 설치된 패키지를 최신 버전으로 업데이트

sudo apt-get install git: Git 설치

(2) 사용자 정보 설정

Git을 사용하려면 사용자 이름과 이메일을 등록해야 합니다.

git config --global user.name "ngm3315"

git config --global user.email "ngm3315@naver.com"

이 정보는 커밋 기록에 남게 되므로, 본인 식별이 가능한 이름과 이메일을 입력하는 것이 좋습니다.

(3) 리포지토리 클론(복제)

GitHub와 연동된 개인 저장소를 불러오려면 git clone 명령어를 사용합니다.

git clone https://github.com/ngm3315/robit_intern_NGM.git

이 명령어를 실행하면 원격 저장소에 있는 파일과 이력들이 로컬 PC로 다운로드됩니다.

정리

Ubuntu 설치는 파티션 분할 → ISO 다운로드 → 부팅 USB 제작 → BIOS 설정 → 설치 진행 순으로 진행되며, 중간에 생길 수 있는 오류는 대부분 안전 모드 설치(safe graphics)나 강제 재부팅으로 해결할 수 있습니다.

Git은 apt-get을 통해 간단히 설치할 수 있으며, 사용자 정보 설정과 원격 저장소 클론 과정을 통해 실제 협업 환경에 활용할 수 있습니다. 이 두 과정을 마치면, Ubuntu 환경에서 본격적으로 소프트웨어 개발 및 프로젝트 관리가 가능합니다.

3. 우분투 - Git - GitHub | 연동 방법

Ubuntu 환경에서 Git과 GitHub를 연동하는 과정은 개발자가 협업과 버전 관리를 효율적으로 수행하기 위해 반드시 거쳐야 하는 단계입니다. Git은 로컬에서 코드의 변경 이력을 관리하는 도구이고, GitHub는 그 이력을 원격 저장소에 보관하며 협업 기능을 제공하는 플랫폼입니다. 따라서 두 시스템을 연동하면, 사용자는 로컬에서 작업한 내용을 원격 저장소에 반영하고, 다른 사람의 변경 사항을 받아와 함께 프로젝트를 진행할 수 있습니다.

가장 먼저 해야 할 일은 Ubuntu 환경에 Git을 설치하고 사용자 정보를 등록하는 것입니다. 앞서 설명드린 것처럼 sudo apt-get install git으로 Git을 설치한 뒤, git config --global user.name과 git config --global user.email 명령어로 사용자 이름과 이메일을 설정합니다. 이 정보는 GitHub에 올린 커밋 기록에도 반영되므로 본인을 명확히 구분할 수 있는 정보를 입력하는 것이 바람직합니다.

다음 단계는 GitHub 계정을 준비하는 것입니다. GitHub에 접속해 회원 가입을 한 뒤, 새로운 저장소(Repository)를 생성합니다. 저장소를 만들 때는 이름, 공개 여부(Public/Private), README 파일 생성 여부 등을 설정할 수 있습니다. 이후 해당 저장소의 HTTPS 주소나 SSH 주소를 확인해 두는 것이 중요합니다.

Ubuntu 환경에서 GitHub와 원격 저장소를 연동하려면 두 가지 방식이 있습니다. 첫째는 HTTPS 방식이고, 둘째는 SSH 키를 사용하는 방식입니다. HTTPS는 간단하지만 매번 사용자 이름과 비밀번호(또는 토큰)를 입력해야 하는 불편함이 있습니다. 반면 SSH 방식은 한 번 키를 등록해 두면 추가 인증 과정 없이 안전하게 접속할 수 있기 때문에 장기적으로 더 편리합니다.

SSH 방식을 사용하는 경우, 먼저 Ubuntu에서 ssh-keygen -t rsa -b 4096 -C "사용자이메일" 명령어를 실행해 SSH 키를 생성합니다. 생성된 공개키는 ~/.ssh/id_rsa.pub 파일에 저장되며, 이 파일의 내용을 GitHub 계정 설정(Settings → SSH and GPG keys)에 등록합니다. 등록을 완료하면, ssh -T git@github.com 명령어로 정상적으로 연결되는지 확인할 수 있습니다.

이제 로컬 저장소와 GitHub 저장소를 연결할 수 있습니다. 새로운 프로젝트를 로컬에서 시작한다면 git init 명령어로 초기화한 뒤, git remote add origin <저장소주소>를 입력해 GitHub와 연동합니다. 이미 GitHub에 저장소가 있다면 git clone <저장소주소>를 사용하여 해당 저장소를 내려받아 작업하면 됩니다. 작업한 후에는 git add .과 git commit -m "메시지"로 변경 사항을 기록하고, git push origin main 명령어를 통해 GitHub에 반영합니다. 다른 사람의 변경 사항을 가져올 때는 git pull origin main 명령어를 사용합니다.

정리하면, Ubuntu에서 Git을 설치하고 사용자 정보를 설정한 뒤, GitHub 계정을 생성하여 원격 저장소와 연동하는 과정이 기본 흐름입니다. 이 과정을 마치면 개인 프로젝트 관리뿐만 아니라 팀 프로젝트에서도 효율적으로 협업할 수 있으며, 원격 저장소를 통해 언제 어디서든 동일한 코드 베이스에 접근할 수 있습니다.

4. 우분투와 Git 주요 명령어 설명

Ubuntu 환경에서 개발을 하다 보면 기본적으로 알아두어야 할 명령어들이 있습니다. 우분투 자체는 리눅스 배포판이므로, 리눅스 공통으로 사용되는 터미널 명령어를 그대로 활용합니다. 또한 Git은 버전 관리 도구로, 협업 과정에서 반드시 익혀야 할 명령어들이 존재합니다. 여기서는 우분투 명령어와 Git 명령어를 줄글 형식으로 정리하여 최대한 상세히 설명드리겠습니다.

1. 우분투 명령어:

우분투에서 가장 많이 사용하는 기본 명령어는 파일과 디렉터리를 다루는 것입니다. ls 명령어는 현재 디렉터리 내의 파일과 폴더 목록을 보여주며, ls -l을 입력하면 권한, 소유자, 크기, 수정 시간까지 상세히 확인할 수 있습니다. 숨김 파일까지 확인하려면 ls -a를 사용합니다. 디렉터리를 이동할 때는 cd를 사용하며, cd ~는 홈 디렉터리로, cd ..는 상위 디렉터리로 이동합니다. 현재 위치를 확인할 때는 pwd 명령어를 사용합니다.

파일과 디렉터리를 생성하거나 삭제할 때는 각각 touch와 rm, mkdir와 rmdir을 사용합니다. 예를 들어 touch test.txt는 빈 파일을 생성하고, mkdir project는 project라는 이름의 폴더를 생성합니다. 파일을 삭제할 때는 rm test.txt, 폴더를 삭제할 때는 rm -r project처럼 -r 옵션을 붙여야 합니다.

파일 내용을 확인하는 방법도 다양합니다. cat은 파일 전체 내용을 한 번에 출력하고, less는 페이지 단위로 나누어 확인할 수 있습니다. 긴 파일을 분석할 때는 head와 tail을 통해 앞부분이나 뒷부분 일부만 볼 수도 있습니다.

프로그램 설치와 관리와 관련된 명령어로는 apt-get이 가장 중요합니다. 패키지 목록을 갱신할 때는 sudo apt-get update, 설치된 패키지를 최신으로 업그레이드할 때는 sudo apt-get upgrade를 입력합니다. 새로운 프로그램을 설치하려면 sudo apt-get install <패키지명>을 사용하고, 필요 없어진 패키지를 제거할 때는 sudo apt-get remove <패키지명>을 사용합니다. 시스템 전체의 불필요한 패키지를 정리하려면 sudo apt-get autoremove를 활용합니다.

시스템 모니터링과 관련해서는 top 명령어가 자주 쓰입니다. top을 입력하면 현재 실행 중인 프로세스와 CPU, 메모리 사용량을 실시간으로 볼 수 있습니다. 보다 향상된 인터페이스를 원하면 htop을 설치해 사용할 수도 있습니다. 디스크 용량을 확인하려면 df -h, 특정 폴더의 크기를 확인하려면 du -sh <폴더명>을 입력합니다. 네트워크 관련해서는 ping으로 연결을 확인하고, ifconfig나 ip addr로 네트워크 인터페이스 상태를 확인할 수 있습니다.

권한 관리도 중요한 부분입니다. chmod는 파일이나 디렉터리의 권한을 수정하는 명령어이며, 예를 들어 chmod 755 실행파일은 실행 권한을 부여합니다. chown은 소유자를 변경하는 명령어로, sudo chown 사용자명 파일명 형식으로 사용합니다.

마지막으로 자주 활용하는 명령어로는 clear가 있습니다. 터미널 화면을 깔끔히 지워주며, 긴 작업 후 화면을 초기화할 때 편리합니다. 종료 시에는 exit를 입력해 세션을 마칠 수 있습니다.

우분투 기본 명령어 표

명령어	설명	사용 예시
ls	현재 디렉터리의 파일/폴더 목록 출력	ls -al(숨김 파일까지 상세 출력)
cd	디렉터리 이동	cd /home/user
pwd	현재 작업 중인 디렉터리 경로 출력	pwd
touch	새 파일 생성	touch test.txt
mkdir	새 디렉터리 생성	mkdir project
rm	파일 삭제	rm test.txt
rm -r	디렉터리 및 내부 파일 삭제	rm -r project
cat	파일 내용 전체 출력	cat test.txt
less	파일 내용을 페이지 단위로 출력	less longfile.txt
head	파일의 앞부분 출력	head -n 20 test.txt
tail	파일의 뒷부분 출력	tail -f log.txt(실시간 로그 확인)
sudo apt-get update	패키지 목록 갱신	sudo apt-get update
sudo apt-get upgrade	설치된 패키지 업그레이드	sudo apt-get upgrade
sudo apt-get install	새 패키지 설치	sudo apt-get install git
sudo apt-get remove	패키지 제거	sudo apt-get remove git
df -h	디스크 사용량 확인	df -h
du -sh	특정 디렉터리 크기 확인	du -sh /home/user
top	실행 중인 프로세스 확인	top
chmod	파일 권한 변경	chmod 755 run.sh
chown	파일 소유자 변경	sudo chown user:user test.txt
clear	터미널 화면 초기화	clear

2. Git 명령어:

Git은 소스 코드의 버전 관리를 위한 도구로, 명령어를 통해 프로젝트를 초기화하고, 변경 사항을 기록하며, 원격 저장소와 동기화합니다.

가장 먼저 새로운 프로젝트에서 Git을 시작할 때는 git init을 입력합니다. 이 명령어를 실행하면 해당 디렉터리가 Git 저장소로 초기화됩니다. 만약 GitHub에 이미 만들어둔 저장소를 내려받아 작업하고 싶다면 git clone <저장소주소>를 사용합니다. 이렇게 하면 저장소의 모든 파일과 이력이 로컬로 복제됩니다.

파일을 수정하거나 새로운 파일을 추가했을 때, Git은 자동으로 변경 사항을 추적하지 않습니다. 따라서 git add 파일명 또는 git add .을 입력해 추적 대상에 포함시켜야 합니다. 그다음 git commit -m "메시지" 명령어로 변경 내용을 로컬 저장소에 기록합니다. 커밋 메시지

는 어떤 변경을 했는지 명확히 작성하는 습관이 좋습니다.

원격 저장소와 연동하려면 먼저 git remote add origin <저장소주소>로 원격을 등록해야 합니다. 이후 git push origin main을 입력하면 로컬 저장소의 변경 사항이 GitHub 원격 저장소로 업로드됩니다. 반대로 다른 사람이 변경한 내용을 내려받을 때는 git pull origin main을 사용합니다.

여러 명이 협업할 때 중요한 기능은 브랜치입니다. 새로운 기능을 개발할 때는 git branch feature로 브랜치를 만들고, git checkout feature로 해당 브랜치로 전환합니다. 최신 Git 버전에서는 git switch -c feature를 사용해 한 번에 브랜치를 생성하고 이동할 수 있습니다. 작업을 마친 후 메인 브랜치에 반영하려면 git merge feature를 사용합니다.

작업 도중 어떤 파일이 수정되었는지 확인하려면 git status 명령어를 사용합니다. 커밋 이력을 확인하려면 git log를 입력하며, 간단하게 보고 싶다면 git log --oneline을 활용할 수 있습니다. 특정 시점으로 되돌아가고 싶을 때는 git reset이나 git checkout <커밋ID>를 사용하지만, 되돌림은 신중히 진행해야 합니다.

사용자 정보 설정도 필수입니다. git config --global user.name "사용자명"과 git config --global user.email "이메일" 명령어로 전역 설정을 등록해야 모든 커밋에 정보가 기록됩니다. 프로젝트마다 다른 정보를 쓰고 싶다면 --global 옵션을 빼고 개별 저장소에서 설정하면 됩니다.

3. 정리:

Ubuntu 명령어는 운영체제를 다루기 위한 기본 도구들이며, Git 명령어는 소스 코드 관리와 협업을 위한 핵심 도구입니다. 두 영역 모두 명령어를 자유롭게 다룰 수 있어야 개발 환경을 효율적으로 활용할 수 있습니다. Ubuntu의 명령어는 파일과 디렉터리 관리, 시스템 관리, 패키지 설치, 권한 설정 등과 직결되며, Git의 명령어는 프로젝트의 히스토리 관리와 협업 효율성을 크게 높여줍니다. Ubuntu와 Git, 그리고 GitHub까지 연동해 사용하면 개발 환경을 완전히 갖출 수 있으며, 이는 현대 소프트웨어 개발에서 필수적인 능력이라 할 수 있습니다.

Git 기본 명령어 표

명령어	설명	사용 예시
git init	새로운 Git 저장소 초기화	git init
git clone	원격 저장소 복제	git clone https://github.com/user/repo.git
git status	현재 작업 상태 확인	git status
git add	변경된 파일 스테이징	git add file.txt/ git add .
git commit	변경 사항 기록	git commit -m "메시지"
git log	커밋 이력 확인	git log --oneline
git branch	브랜치 목록 확인/생성	git branch feature
git checkout	브랜치 이동	git checkout feature

git switch	브랜치 전환 (최신 문법)	git switch -c feature
git merge	브랜치 병합	git merge feature
git remote add origin	원격 저장소 등록	git remote add origin https://github.com/user/repo.git
git push	원격 저장소로 업로드	git push origin main
git pull	원격 저장소 변경 내용 가져오기	git pull origin main
git config	사용자 정보 설정	git config --global user.name "홍길동"
git reset	커밋 되돌리기	git reset --hard HEAD~1
git diff	변경된 코드 비교	git diff

-끝-