Linux & Git 보고서

20기 인턴 우정윤

목차

1. 서론
2. Linux 개요
3. Linux 기본 사용법
4. Linux 고급 기능
5. Git 개요
6. Git 기본 사용법
7. Git 고급 기능
8. Linux와 Git의 연계
9. 사례연구
10. 결론

1. 서론

본 보고서에서는 Linux와 Git에 대해서 알아보고 이들에 대한 개념과 중요성 및 서로 어떻게 연계되는지 알아보고자 한다.

2. Linux 개요

2.1 Linux의 역사

1991년 핀란드 헬싱키 대학의 대학원생 리누스 토르발스가 개인 프로젝트로 개발한 운영체제가 바로 Linux이다. 그는 당시 상용 Unix 시스템이 너무 비싸고 접근이 어렵다고 생각했고 무료로 누구나 사용할 수 있는 유사 운영체제를 만들고자 하였다. 그리고 이 운영체제에 리차드 스톨만이 주도한 자유 소프트웨어 운동과 결합하여 GNU/Linux 형태로 발전하였다.

2.2 Linux의 특징

특징으로는 소스코드가 공개되어 있어서 누구나 자유롭게 사용, 수정, 배포할 수 있다. 작은 임베디드 장치부터 대규모 슈퍼컴퓨터까지 다양한 환경에서 동작하며 서버 운영체제로 안정성이 높아서 기업에서 선호하는 편이다. 여러 사용자가 동시에 로그인 가능하며 여러 프로세스를 동시에 실행 할 수 있다.

2.3 Linux의 구조

리눅스는 크게 커널, 쉘과 사용자 영역으로 나뉜다. 커널은 운영 체제의 핵심부분으로서 하드웨어와 직접 상호작용하며 자원(CPU, 메모리, 디스크, 네트워크 등)을 관리한다. 드라이버, 프로세스 스케쥴러, 메모리 관리자 등을 포함한다. 쉘은 일종의 명령어 해석기로 사용자가 명령어를 입력하면 커널에 전달하는 인터페이스로 bash, zsh 등 다양한 종류가 존재한다. 파일 시스템의 경우 응용프로그램으로 리눅스 상에서 사용자가 사용할 수 있는 일반 프로그램을 말한다.

2.4 활용 분야

리눅스는 다양한 분야에서 활용되는데 서버 시장에서 웹 서버, 데이터베이스, 서버등에서 사용되고 안드로이드 기본 모바일 OS에서 사용되며 AWS, GCP, Azure 대부분의 서버가 Linux기반으로 이루어져 있다. 라즈베리파이, IOT 기기, 네트워크 장비 등에 임베디드 시스템에서 많이 사용되며 대부분의 슈퍼컴퓨터가 Linux 기반이다.

3. Linux 기본 사용법

3-1. 기본 명령어

Linux는 Cli 기반으로 사용되면 대표적인 명령어는 다음과 같다.

- 파일/디렉토리 관련

  Pwd: 현재 디렉토리 경로 출력

  Ls: 파일 목록 보기

  Cd: 디렉토리 이동

  Mkdir: 새 디렉토리 생성

  Rm: 파일 삭제, 디렉토리 삭제

- 파일 내용 확인

  Cat: 파일 내용 출력

  Less, more: 긴 파일 페이지 단위 출력

  Head, tail: 파일 앞/뒤 일부 출력

- 권한 관련

  Chmod: 권한 변경

  Chown: 소유자 변경

- 프로세스 관리

  Ps: 현재 실행 중인 프로세스 목록

  Top: 실시간 CPU/메모리 사용량 확인

  Kill: 프로세스 종료

3-2. 사용자와 그룹 관리

- 사용자 추가: adduser student
- 비밀번호 변경: passwd student
- 그룹 관리: groupadd dev, usermod –aG dev student

3-3. 패키지 관리

- Apt(UBUNTU 계열)

  Sudo apt update: 패키지 목록 갱신

  Sudo apt install vim: 패키지 설치

3-4. 쉘과 스크립트

- 쉘 종류: bash, zsh, ksk 등

스크립트를 활용해 반복작업 자동화, 서버 관리, 빌드 자동화 등에 활용 할 수 있다.

3-5. 파일 시스템 구조

- 리눅스 파일 시스템 특징: 루트 디렉토리를 기준으로 모든 것이 파일로 존재하며 드라이브 문자가 없는 계층적 구조가 특징이다.
- 주요 디렉토리

  /bin: 기본 실행 파일

  /etc: 설정 파일

  /home: 사용자 홈 디렉토리

  /var: 로그, 캐시 등 가변 데이터

  /dev: 장치 파일

  /proc: 현재 실행 중인 프로세스 정보

4. Linux 고급 기능

4-1. 사용자와 그룹 관리

리눅스는 다중 사용자 환경을 지원하므로, 접근 권한과 보안 유지를 위해 사용자와 그룹 개념이 필수적이다. 주요 명령어로는 사용자를 추가하는 sudo adduser alice와 사용자를 삭제하는 sudo deluseralice, 그룹 추가에 쓰이는 sudo groupadd dev와 그룹에 사용자를 추가하는 sudo usermod –aG dev alice와 현재 로그인 사용자를 확인하는 who 와 w가 있다. Root 계정은 시스템의 모든 권한을 가진 관리자 계정이다.

4-2. 파일 권한과 보안

리눅스 권한 체계는 사용자, 그룹, 기타로 나뉜다. 읽기에는 r 쓰기에는 w 실행에는 x가 사용된다.

4-3. 프로세스와 서비스 관리

실행 중인 프로세를 확인 할때는 ps aux, top, htop 명령어가 사용된다. 또한 특정 프로세스를 종료해야 할때는 kill-9 Pidrk 사용된다. 서비스 관리의 경우 서비스 시작은 sudo systemctl start apache2, 서비스 중지는 sudo systemctl stop apache2, 서비스 자동 실행 등록은 sudo systemctl enable apache2가 쓰인다.

4-4. 네트워크 관리

- ifconfig (ip addr): 네트워크 인터페이스 확인
- ping google.com: 네트워크 연결 확인
- netstat -tulnp: 열려 있는 포트 확인
- scp file user@host:/path: 파일 전송

4-5. 시스템 로그와 모니터링

- var/log/syslog: 시스템 로그
- /var/log/auth.log: 인증 관련 로그
- /var/log/dmesg: 부팅 로그
- df -h: 디스크 사용량 확인

- du -sh dir/: 특정 디렉토리 크기 확인
- free -m: 메모리 사용량 확인

5. Git 개요

5-1. 버전 관리 시스템의 필요성

여러 사람이 같은 프로젝트 파일을 동시에 수정할 때 충돌이 발생하게 된다. 파일 버전을 수동으로 관리하면 혼란이 벌어지는데 실수로 내용을 덮어쓰거나 삭제했을 때 복구가 어렵다. 따라서 이를 해결하기 위해 파일의 변경 이력을 저장하고 과거 버전으로 되돌릴 수 있게 해주는 것이 VCS이다. 여러 사람이 동시에 작업 가능하게 해주며 변경 사항 추적 및 기록이 가능하다.

5-2. Git의 역사

2005년, 리누스 토르발스(Linus Torvalds)가 리눅스 커널 개발을 위해 Git을 개발하였다. 기존 상용 VCS(BitKeeper)를 대체하기 위해 무료/오픈소스 버전 관리 도구 필요하여 만들어진 오늘날 가장 널리 사용되는 분산형 버전 관리 시스템(DVCS)이다.

5-3. Git의 특징

- 분산형: 모든 개발자가 전체 저장소를 로컬에 보관하고 중앙 서버가 없어도 로컬에서 독립적으로 작업이 가능하다.
- 빠른 속도와 효율성: 변경 내용을 스냅샷 방식으로 저장하고 해시를 이용한 데이터 무결성을 보장한다.
- 브랜치 기능: 독립된 작업 공간을 쉽게 만들 수 있다. 병합 기능으로 협업 효율이 증가한다.
- 오픈소스와 협업 친화적: GitHub, GitLab등 다양한 호스팅 서비스와 연계되며 전 ㅅ계 개발자들이 Git을 통해 오픈소스 프로젝트에 기여한다.

5-4. Git의 활용분야

- 소프트웨어 개발 협업
- 오픈소스 프로젝트
- 개인학습 및 포트폴리오 관리

6. Git 기본 사용법

6-1. 설치

sudo apt update, sudo apt install git를 이용하여 설치한다.

6-2. 로컬 저장소 생성

mkdir myproject

cd myproject

git init

를 입력하면 .git 폴더가 생성되며 Git저장소로 초기화 된다.

6-3. Git 기본 명령어

- Git init: 현재 디렉토리를 Git 저장소로 초기화
- Git clone URl: 원격 저장소 복제
- Git status: 현재 작업 상태 확인
- Git add: 변경된 파일 스테이징
- Git commin –m: 스냅샷 저장
- Git log: 커밋 히스토리 확인
- Git diff: 변경 사항 확인

6-4. Git 기본 워크플로우

- 파일 생성/수정: git add
- 파일 저장: git commit
- 원격 저장소 업로드: git push
- 협업 시 최신 코드 반영: git pull

7. Git 고급 기능

7-1. 원격 저장소

원격 저장소 조회 및 변경/삭제가 가능하며 하나의 프로젝트에 GitHub, GitLab등 복수의 저장소를 연결할 수 있다.

7-2. 충돌 해결

두 개발자가 같은 파일의 같은 부분을 동시에 수정하고 병합 시 충돌이 발생하게 되는데 이를 해결하기 위해 충돌 파일에서 <<<<<<, =======,>>>>>>>부분을 확인해서 원하는 코드로 수정 후 저장하면 된다. 그 후 다시 add/commit을 실행하면 된다.

7-3. Git 내부 구조 이해

- 스냅샷 저장: Git은 파일을 전체 복사하는 대신, 변경된 부분만 기록
- Blob: 파일의 순수 데이터 자체를 저장한다. 파일의 이름이나 메타데이터는 포함하지 않고 오직 내용만을 기준으로 내용 기반의 체크섬을 생성한다.
- Tree: Git이 폴더 구조를 표현하기 위한 객체이다. 내부에 포함될 파일 식별자와 디렉토리의 목록, 그리고 각 파일이나 디렉토리의 이름을 기록한다.
- Commit: 특정 시점의 프로젝트 스냅샷을 기록하는 객체로 가장 최근의 tree 객체를 가리키고, 작성자, 커밋 메시지등에 대한 메타데이터를 포함한다.

7-4. Git과 CI/CD 연계

- Git은 단순 버전 관리 도구를 넘어 CI/CD 파이프라인의 핵심이며 GitHub Actions, GitLab Ci, Jenkins와 통합 가능하다. 따라서 코드 품질 유지, 버그 조기 발견에 효과적이며 운영 서버에 수동 개입 없이 빠르고 안정적인 배포가 가능하다.

8. Linux와 Git의 연계

Git 자체가 Linux환경에서 개발된 것으로 Git명령어는 리눅스 쉘에서 실행되는 CLI기반이다. 대부분의 서버가 리눅스 기반이므로 Git 사용과 자연스럽게 연결된다. Linux 서버에서 Git을 이용해 새로운 코드를 배포하고 리눅스의 쉘 스크립트를 활용해 이를 자동화 시킬 수 있다. 많은 개발자들이 Git을 통해 커널 소스를 관리한다.

9. 사례 연구

9-1. 리눅스 커널 개발

세계 최대 규모의 오픈소스 프로젝트 중 하나로 수천 명의 개발자가 전 세계에서 Git을 통해 협업하고 있다. 매일 수백 개의 변경 사항을 반영하며 Git을 통한 브랜치/병합 전략 때문에 안정성과 확장성을 유지한다.

9-2. 기업 및 산업 현장의 Git 활용

여러 대기업 및 소프트웨어 기업, 제조/로봇 기업들이 임베디드 시스템 개발시 Git으로 코드 관리를 하며 대부분의 클라우드 기업이 리눅스 서버 기반으로 git 기반 CI/CD로 안정적인 배포를 한다.

10. 결론

10-1. Linux와 Git의 의의

-Linux

- 오픈소스와 자유 소프트웨어 정신을 대표하는 운영체제
- 안정성, 보안성, 확장성을 바탕으로 서버, 클라우드, 임베디드, 슈퍼 컴퓨터 등 다양한 분야에서 사실상의 표준으로 자리 잡음

-Git

- 단순한 버전 관리 도구를 넘어 전 세계 개발자들이 협업할 수 있는 필수 플랫폼으로 발전
- 오픈소스 프로젝트의 성공 사례를 통해, 분산 버전 관리의 효용성을 입증
- GitHub, GitLab 등과 결합하여 DevOps 및 CD/CI 환경의 중심축 역할 수행

10-2. 미래전망

-Linux

- 클라우드, AI, IoT 서버 운영체제의 핵심으로 계속 활용
- 모바일 임베디드 장치에서도 계속 성장

- 오픈소스 커뮤니티 주도로 지속적인 진화 예상

-Git

- 원격 협업과 글로벌 소프트웨어 개발의 기본 도구로 자리매김
- CI/CD와의 결합을 통해 개발-테스트-배포 자동화의 핵심 인프라로 확장
- 오픈소스와 사용 소프트웨어 개발 모두에서 사실상 표준화

10-3. 학습 의의

이번 보고서를 통해 Linux의 구조와 철학을 이해하면서, 시스템 자원을 어떻게 관리하고 효율적으로 운영하는지를 배웠다. Git의 사용법과 고급 기능을 학습하면서, 협업 과정에서 버전 관리의 중요성과 자동화의 가치를 실감했다.