원격 접속

원격 접속

- Server/Client
  - ✓ 개요
    - 많은 사람이 네이버(www.naver.com)를 사용하는데 이것 역시 서버/클라이언트의 개념
    - 네이버라는 웹 서버가 작동하고 있고 사람들은 웹 브라우저(인터넷 익스플로러 크롬, 사파리 파이어폭스 등)라는 웹 클라이언트 프로그램을 이용해서 웹 서버에 접속
    - 서버 프로그램이 작동할 때 서버를 사용하려면 클라이언트 프로그램이 필요
    - 텔넷 서버도 마찬가지로 텔넷 서버 프로그램이 작동할 때 텔넷 서버에 접속하려면 텔넷 클라이언트 프로그램이 있어야 함
    - 대부분의 다른 서버 프로그램도 각각의 클라이언트 프로그램이 있어야 서버에 접속해서 사용할 수 있는 것과 같음

원격 접속

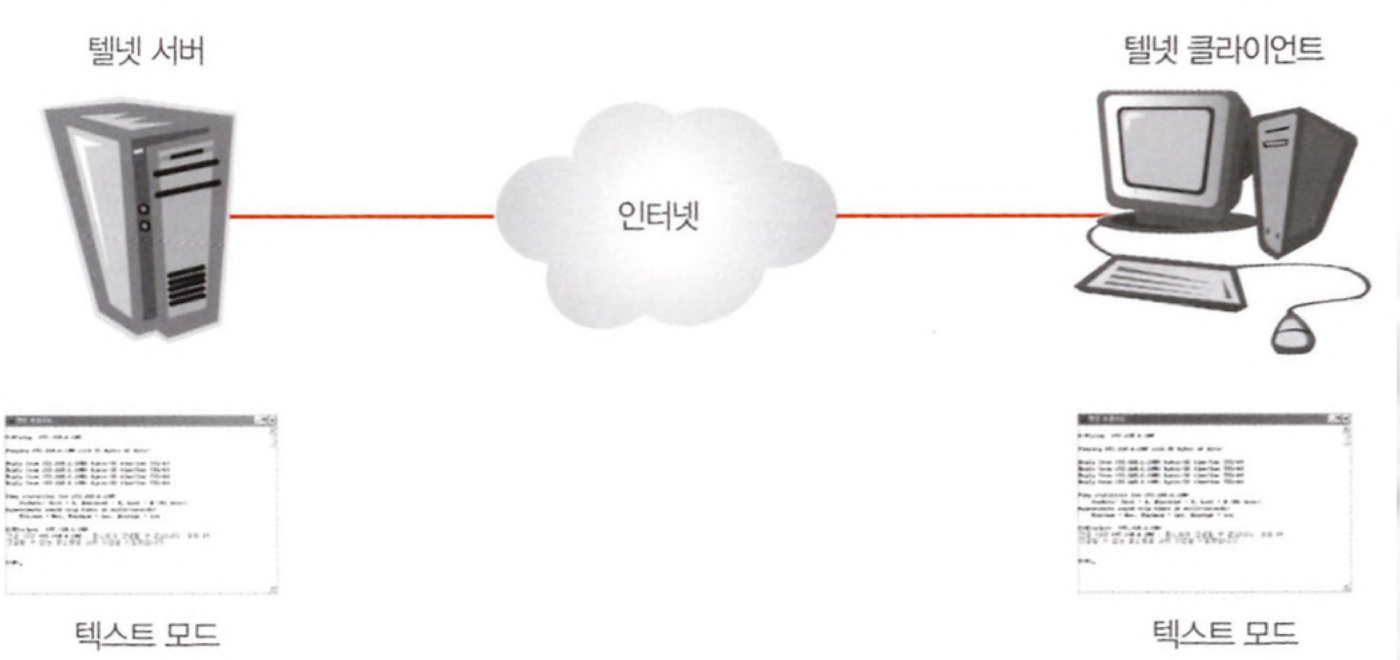
- Server/Client
  - ✓ 특징
    - 서버에 접속하려면 반드시 클라이언트 프로그램이 필요한데 서버가 리눅스라고 해서 클라이언트도 리눅스일 필요는 없음
    - 서버의 OS와 클라이언트의 OS가 같아야 하는 것은 아님
    - 각각의 서버 프로그램은 자신에게 맞는 별도의 클라이언트 프로그램이 필요웹 서버 (아파치 또는 IIS)
      - 웹 서버 => 웹 클라이언트(인터넷 익스플로러, 크롬, 사파리, 파이어폭스 등)
      - 텔넷 서버 => 텔넷 클라이언트(telnet, PuTTY 등)
      - FTP 서버 => FTP 클라이언트(알FTP, wsFTP, ftp, gftp 등)
      - XRDP 서버 => XRDP 클라이언트(원격 데스크톱)
      - SSHD 서버 => SSH 클라이언트(ssh, PuTTY 등)
      - 오라클 서버 => 오라클 클라이언트(sqlplus)

원격 접속

❖ Telnet

✓ 개요

- Telnet은 Telecommunication Network의 약자로 인터넷이나 로컬 영역 네트워크(LAN) 연결에 사용되는 네트워크 프로토콜
- 1969년에 처음 개발된 아주 오래된 표준 중 하나로 멀리 떨어진 컴퓨터에 가상으로 접속하여 마치 그 컴퓨터 앞에 앉아 있는 것처럼 명령어를 실행할 수 있게 해줌



원격 접속

- Telnet
  - ✓ 특징
    - 원격 접속: 텍스트 기반의 인터페이스(CLI)를 통해 원격 호스트를 제어
    - TCP 기반: 기본적으로 TCP 23번 포트를 사용하여 통신
    - 플랫폼 독립성: 윈도우, 리눅스, 맥 등 서로 다른 OS 간에도 표준화된 방식으로 통신이 가능
    - NVT(Network Virtual Terminal): 서로 다른 시스템 환경에서도 명령어를 인식할 수 있도록 가상의 단말기 개념을 사용하여 데이터를 주고 받음
  - ✓ 장점
    - 설치가 간편하고 사용법이 매우 쉬움
    - 시스템 리소스를 적게 차지하며 오래된 장비(레거시 시스템)와도 호스트 호환성이 뛰어남
  - ✓ 단점
    - 데이터가 암호화되지 않은 평문(Plaintext)으로 전송되어 패킷 스니핑 등을 통해 정보가 유출될 위험이 큼

원격 접속

- Telnet
  - ✓ 보안 문제 때문에 실제 원격 제어용으로는 SSH(Secure Shell)가 텔넷을 완전히 대체했지만 오늘날에도 엔지니어들은 다음과 같은 용도로 텔넷을 자주 사용
    - 포트 개방 확인 (네트워크 진단): 특정 서버의 특정 포트가 열려 있는지 확인할 때 유용 - telnet 1.2.3.4 80 (해당 IP의 80번 포트가 열려 있는지 테스트)
    - 애플리케이션 테스트: HTTP, SMTP 등 텍스트 기반 프로토콜의 응답을 직접 확인하기 위해 수동으로 명령을 보낼 때 사용
  - ✓ 설치 확인
    - dpkg -l | grep telnet

      ii inetutils-telnet 2:2.5-3ubuntu4 arm64 telnet client

      ii telnet 0.17+2.5-3ubuntu4 all transitional dummy package for inetutils-telnet default switch
      - ◆ 클라이언트만 설치되어 있음

원격 접속

❖ SSH

✓ 개요

- SSH(Secure Shell)는 네트워크 상의 다른 컴퓨터에 로그인하거나 원격 시스템에서 명령을 실행할 수 있게 해주는 보안 네트워크 프로토콜
- 과거에 사용하던 Telnet이나 FTP 등은 데이터를 Plaintext으로 전송하여 해킹에 매우 취약했지만 SSH는 모든 통신 내용을 암호화하여 보안성을 획기적으로 높임

✓ 특징

- 강력한 암호화: 패스워드뿐만 아니라 전송되는 데이터 전체를 암호화
- 인증(Authentication): 접속하려는 사용자가 올바른 사용자인지 확인(비밀번호 방식, 공개키 방식 등)
- 무결성(Integrity): 전송된 데이터가 중간에 위변조되지 않았음을 보장
- 압축: 데이터를 압축하여 전송하므로 네트워크 효율이 좋음

원격 접속

- SSH
  - ✓ SSH의 동작 원리
    - SSH는 기본적으로 클라이언트-서버 모델로 동작하며 표준 포트는 22번
    - 가장 권장되는 인증 방식은 비밀번호가 아닌 SSH Key(공개키/개인키)를 사용하는 방식
      - 공개키(Public Key): 서버에 저장되는데 누구나 봐도 상관없으며 데이터를 암호화하는 데 사용
      - 개인키(Private Key): 클라이언트(사용자 PC)에 보관되는데 절대로 유출되면 안 되며 암호화된 데이터를 푸는(복호화) 데 사용
  - ✓ SSH의 주요 활용 사례
    - 원격 터미널 접속: 리눅스 서버 등에 접속하여 명령어를 입력하고 관리
    - SFTP / SCPSSH: 보안 채널을 이용한 안전한 파일 전송
    - SSH Tunneling: 보안이 취약한 다른 프로토콜을 SSH 안에 넣어 안전하게 전달
    - Git 원격 저장소: GitHub, GitLab 등에서 코드를 푸시(Push)할 때 인증 수단으로 사용

원격 접속

- SSH
  - ✓ 설치 및 동작 및 접속
    - 설치: sudo apt -y install openssh-server
    - 서비스 구동 및 확인
      - 서비스 시작: sudo systemctl start ssh
      - 부팅과 동시에 서비스 시작: sudo systemctl enable ssh
      - 서비스 가동 여부 확인: sudo systemctl status ssh

        ssh.service - OpenBSD Secure Shell server

        Loaded: loaded (/usr/lib/systemd/system/ssh.service; enabled; preset: enabled)

        Active: active (running) since Wed 2026-01-07 22:55:59 UTC; 5h 52min ago

        TriggeredBy: ● ssh.socket

        Docs: man:sshd(8)

        man:sshd_config(5)
    - 방화벽에서 22번 포트 해제: sudo ufw allow 22/tcp

원격 접속

- ❖ SSH
  - ✓ 설치 및 동작 및 접속
    - ● 다른 컴퓨터에서 접속: ssh 사용자이름@IP주소 -p 포트번호
      - ◆ 22번 포트의 경우 -p 옵션 생략 가능
    - ● Virtual Machine의 경우 Port Forwarding을 해야 할 수 있음

원격 접속

- SSH
  - ✓ 설치 및 동작 및 접속
    - 처음 접속하면 아래와 같은 메시지가 출력됨

      The authenticity of host '[192.168.202.104]:20001 ([192.168.202.104]:20001)' can't be established.

      ED25519 key fingerprint is SHA256:mmPS+2yG9bAMYYYXmk2HEt873Pf1AaJQf7tfbFXiSKI.

      This host key is known by the following other names/addresses:

      ~/.ssh/known_hosts:18: [192.168.202.104]:10001

      Are you sure you want to continue connecting (yes/no/[fingerprint])?

      - ◆ 이 메시지는 에러가 아니라 SSH가 처음 접속하거나 변경된 서버를 확인할 때 띄우는 보안 확인 절차로 본인이 관리하는 서버가 확실하다면 yes를 입력하고 진행
      - ◆ The authenticity... can't be established: "이 서버의 신원을 증명할 수 없습니다." (이전에 이 포트로 접속한 기록이 없기 때문)
      - ◆ ED25519 key fingerprint...: 서버가 내민 '지문(Fingerprint) ' 으로 서버마다 고유한 값을 가짐
      - ◆ Are you sure you want to continue connecting?: "이 서버를 믿고 계속 연결할까요?"라고 묻는 것

원격 접속

- SSH
  - ✓ SSH 키 방식 접속 설정
    - 비밀번호보다 훨씬 안전하며 매번 비밀번호를 입력할 필요가 없어 매우 편리
    - SSH 키 쌍(Key Pair) 생성
      - 접속하고자 하는 클라이언트 컴퓨터의 터미널(Mac/Linux) 또는 PowerShell(Windows)에서 수행

        ssh-keygen -t ed25519
      - Enter를 세 번(저장 위치와 비밀번호 설정을 기본값으로 두는 과정)
      - 완료되면 ~/.ssh/ 폴더에 id_rsa(개인키)와 id_rsa.pub(공개키) 파일이 생성됨

원격 접속

- SSH
  - ✓ SSH 키 방식 접속 설정
    - ● 서버에 공개키 등록
      - ◆ 접속하고자 하는 컴퓨터에서 ssh-copy-id 명령어를 사용해서 접속하려는 서버에 키를 등록

        ssh-copy-id -p 포트번호 [사용자명]@[서버주소]
      - ◆ 수동으로 등록
        - ▪ 접속하고자 하는 컴퓨터에서 공개키 내용을 직접복사
          - cat ~/.ssh/id_ed25519.pub | ssh [사용자명]@[서버주소] "mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys"

원격 접속

- ❖ SSH
  - ✓ SSH 키 방식 접속 설정
    - ● 서버 설정
      - ◆ 권한 설정
        - ▪ chmod 700 ~/.ssh
        - ▪ chmod 600 ~/.ssh/authorized_keys
      - ◆ SSH 설정 확인
        - ▪ sudo nano /etc/ssh/sshd_config

          PubkeyAuthentication yes

          AuthorizedKeysFile .ssh/authorized_keys

          PasswordAuthentication yes # 테스트 후 no로 바꿔도 됨
      - ◆ SSH 재시작
        - ▪ sudo systemctl restart ssh

원격 접속

- SSH
  - ✓ SSH 키 방식 접속 설정
    - 키를 지정해서 접속: ssh -i ~/.ssh/id_ed25519 [사용자명]@[서버주소]
    - 접속 간편화 – 클라이언트에서 설정
      - nano ~/.ssh/config

        Host myserver

        HostName [서버주소]

        User [사용자명]

        IdentityFile ~/.ssh/id_ed25519
      - ssh myserver