

1) 진행 상황 정리

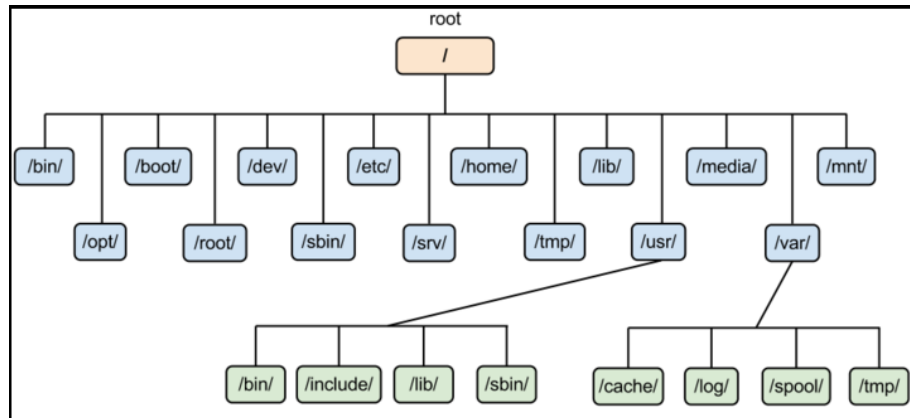
- 현재는 host 를 % 로 전체 허용, db의 보안을 위해
 - 인바운드 규칙이 우선 적용
 - 우분투의 방화벽이 두번째로 적용
 - db 의 계정 권한 설정 / db의 IP 설정
 - putty 는 공개키/비밀키 가지고 있기에 사실상 예외(root급 권한)

2) 시스템 설치

- IntelliJ : [IntelliJ IDEA Community Edition](#)
 - 스크롤 조금 내려야 함, 커뮤니티버전은 jsp 생성 불가
 - 돋보기>시스템 환경 변수 검색 > 환경 변수>새로 만들기> JAVA_HOME, C:\Program File\Java\jdk-11 > 확인
 - CLASSPATH, %JAVA_HOME%\lib
 - %JAVA_HOME%\bin
 - 와일드카드문자 : ? ' * % (%사이의 전부다~~하라)
 - 아래가 자동으로 세팅됨
 - Java : open jdk or Oracle java(보편적)
 - Oracle java : LT 관리 잘됨, 버전이 다른 것 동시 설치시 충돌, 변경시 이전 것 삭제
 - Java17 - Spring boot
 - Java11 - 학습용에 적합
 - Oracle : [JDK 11 Documentation - Home \(oracle.com\)](#)
 - https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe (sha256)
 - Java 는 설치해도 아이콘이 제공되지 않음
 - [환경변수 설정](#) : 시스템 환경 변수 검색 > 고급 > 환경 변수
 - 새로만들기 > CLASSPATH %JAVA_HOME%\lib
 - 새로만들기 > JAVAHOME C:\Program Files\Java\jdk-17
 - 시스템변수 > Path (더블클릭) > 새로만들기 > %JAVA_HOME%\bin > 위로이동 > 확인 (다른버전 자바가 있을 경우 우선순위 설정)
 - CMD 에서 확인 : java -version & javac -version
- ```
C:\Users\student00>java -version
java version "17.0.9" 2023-10-17 LTS
Java(TM) SE Runtime Environment (bui
Java HotSpot(TM) 64-Bit Server VM (b

C:\Users\student00>javac -version
javac 17.0.9
```
- 안나오면 환경재설정 필요
- java : .class 파일을 실행 역할
  - javac : 작성한 텍스트를 .class(bytecode) 파일로 컴파일(c) 역할
  - 환경변수 필요성 : 운영체제 어디든 자바를 인식(사용)할 수 있도록 경로지정
- Apache Tomcat
  - 다운로드 경로 : apache tomcat 검색 > [download](#) > [Quick Navigation 9.0.83](#) > Core > [64-bit Windows zip \(pgp, sha512\)](#)

### 3) 파일 시스템 구조<sup>1</sup>



- **bin** : user binary
  - `/usr/bin` 디렉토리에 대한 기호 링크
- **dev** : device
  - 로컬 장치를 위한 특수 파일용 장치 노드를 포함합니다. `/dev` 디렉토리는 테이프 장치, 프린터, 디스크 파티션 및 터미널에 대한 특수 파일을 포함합니다.
- **etc** : 환경설정(config), 각 장치에 대한 다양한 구성 파일을 포함
  - `/etc/hosts` `/etc/passwd`
- **home** :
  - ubuntu 설치됨
  - 사용자 홈 디렉토리를 포함하는 파일 시스템에 대한 마운트 위치 역할을 합니다. `/home` 파일 시스템은 각각의 사용자의 파일 및 디렉토리를 포함합니다.
  - 독립형 시스템에서 별도의 로컬 파일 시스템은 `/home` 디렉토리 위에 마운트됩니다. 네트워크에서, 서버는 몇 개의 시스템에서 액세스되어야 하는 사용자 파일을 포함할 수도 있습니다. 이런 경우, 서버의 `/home` 디렉토리 사본은 `/home` 파일 시스템에 원격적으로 마운트됩니다.
- **mnt** : 가상화
- **lib** : 라이브러리
  - `lib*.a`인 구조 독립형 라이브러리가 있는 `/usr/lib` 디렉토리에 대한 기호 링크
- **usr** : 사용자 정보 기록
  - 변경되지 않고 시스템에서 공유될 수 있는 파일(실행 가능한 프로그램 및 ASCII 문서)을 포함하는 파일 시스템을 위한 마운트 위치의 역할을 실행합니다.
- **var** : db caching, webserver
  - 각각의 기계를 연결 변환하는 다양한 파일에 대한 마운트 위치 역할을 합니다. 이 파일은 포함하고 있는 파일이 커지기 쉬운 파일이므로, `/var` 파일 시스템은 하나의 파일 시스템으로 구성됩니다. 예를 들어, 임시 작업 파일을 포함하는 `/usr/tmp` 디렉토리에 대한 기호 링크입니다.
- 비우선 순위
  - `/export` 원격 클라이언트에 대한 서버의 디렉토리 및 파일을 포함합니다.
  - `/sbin` 시스템을 부트하고 `/usr` 파일 시스템을 마운트할 때 필요한 파일이 있습니다. 부팅 중에 사용되는 대부분의 명령은 부트 이미지 RAM 디스크 파일 시스템이기 때문에 `/sbin` 디렉토리에 상주하는 명령은 극소수입니다.
  - `/tmp` 시스템 작성 임시 파일을 포함하는 파일 시스템에 대한 마운트 위치 역할을 합니다.
  - `/u` `/home` 디렉토리에 대한 기호 링크.

<sup>1</sup> 파일및디렉토리 - 생활코딩

- sudo apt-get install tree

- 리눅스 접속

```
ubuntu@ip-172-31-33-154:~$
```

- ubuntu : 현재의 root (ID)
- 172-31-33-154 : aws 가 접속자를 구분하기 위한 private IP Address
- ~ : 특정 디렉토리(현재 home)의 하위 디렉토리인 상태를 의미

- home directory

```
ubuntu@ip-172-31-33-154:~$ cd .
ubuntu@ip-172-31-33-154:~$ cd ..
ubuntu@ip-172-31-33-154:/home$ tree
.
└── ubuntu

1 directory, 0 files
ubuntu@ip-172-31-33-154:/home$
```

- cd . : 내자신
- cd .. : 상위 디렉토리 이동
  - home 으로 변경
- cd / : root 로 이동
- cd /home/ubuntu : 초기 디렉토리 위치로 이동
  - cd ~ 도 가능, 우분투의 default 위치이므로
- tree : 현재 디렉토리 트리 확인

- cd http
  - cd httpd
  - cd ppp : 현재 아파치 미설치로 없음

- sudo vi ufw.conf

```
Set to yes to
to allow your
ENABLED=no

Please use the
See 'man ufw'
LOGLEVEL=low
~
~
```

- enable=no : 방화벽 전체오픈
- ~ : 더이상 내용 없음

- cd .ssh > ls -al > cat authorized\_keys | grep 'g'

#### 4) apache 설치

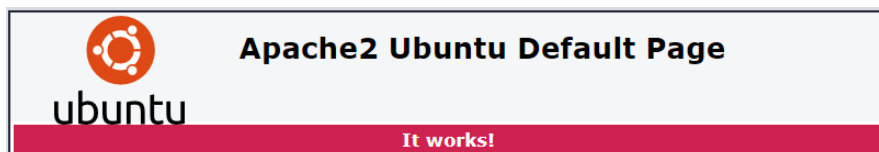
- 설치전 : `sudo apt-get update / sudo apt-get upgrade`
- 설치 : `sudo apt-get install apache2`
  - 기본 환경 설정 파일은 `/etc/apache2/apache2.conf`
- 버전 확인 : `apache2 -v`

```
Server version: Apache/2.4.41 (Ubuntu)
Server built: 2023-10-26T13:54:09
```

- 넷툴 설치 : `sudo apt-get install net-tools`
  - `cd /bin > ls -al > nettools`, `netstart` 확인 안됨
  - 넷툴 : 네트워크에 연결된 Address들을 확인
- 넷툴로 확인 : `netstat -ntlp`

| Proto | Recv-Q | Send-Q | Local Address | Foreign Address | State  | PID/Program name |
|-------|--------|--------|---------------|-----------------|--------|------------------|
| tcp   | 0      | 0      | 0.0.0.0:22    | 0.0.0.0:*       | LISTEN | -                |
| tcp   | 0      | 0      | 127.0.0.53:53 | 0.0.0.0:*       | LISTEN | -                |
| tcp   | 0      | 0      | 0.0.0.0:3306  | 0.0.0.0:*       | LISTEN | -                |
| tcp6  | 0      | 0      | :::22         | :::*            | LISTEN | -                |
| tcp6  | 0      | 0      | :::80         | :::*            | LISTEN | -                |

- 아파치가 제대로 작동되지 않는다? 무슨 차이로 확인?
- AWS에서 막힌 상태
- 약간의 변동이 있으면 재시작 권장
  - 아파치 재시작 : `sudo systemctl restart apache2`
  - 아파치 상태 확인 : `sudo systemctl status apache2`
  - `sudo service apache2 stop` 도 가능
- 아파치 웹서버 설치 완료시 : 브라우저에서 탄력적 IP 입력



- <http://3.35.213.179/>
- `cd /var/www/html > sudo vi index.html` 에서 소스코드 확인

- 권한 상태

```
-rw-r--r--
```

- 사용자 권한 / 그룹 / 게스트 순서
- 읽기, 쓰기 / 읽기 / 읽기

```
drwxr-xr-x 3 root root 4096
```

## 5) java 설치이후

### 5-1) AWS EC2 에서의 Java 설치(ubuntu)

- JDK 설치 : `sudo apt-get install openjdk-11-jdk`
- JRE 설치 : `sudo apt-get install openjdk-11-jre`
  - `jre` : java의 구문이 구동되도록, 서버에는 `jre`만 설치가 권장
- 리눅스에서는 설치 프로그램을 다운하지 않음, 명령어만 입력함, 필수 링크가 연동되어 있음
- 윈도우에서는 `java-17`, 리눅스에서는 `java-11`, (classic) `java-8`
  - 버전이 낮을수록 대중적, 안정적인, 대형시스템에서 채택
  - 우리가 만들었던 것은 둘다 상관 없음
  - 차후 `spring boot(Framework)` 를 위해 17 미리 설치
  - `spring(JSP,Servlet)` 은 8 이후 버전으로 다 상관 없음
- 경로확인 : `cd /usr/lib/jvm > ls -al`

```
lrwxrwxrwx 1 root root 21 Oct 19 16:55 java-1.11.0-openjdk-amd64
-> java-11-openjdk-amd64
```

### 5-2) 리눅스에서의 JAVA 환경변수 설정

- `sudo vi /etc/profile`
  - 해당 구문을 맨 아래에 추가 > i > esc > :wq
  - `export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64`
  - `export PATH=$JAVA_HOME/bin:$PATH`
  - `export CLASS_PATH=$JAVA_HOME/lib:$CLASS_PATH`
- 변경된 내용을 저장 : `source /etc/profile`
- 우분투 리부트 : `sudo reboot now`
- 환경변수 설정 확인 : `echo $JAVA_HOME`

```
ubuntu@ip-172-31-33-154:~$ echo $JAVA_HOME
/usr/lib/jvm/java-11-openjdk-amd64
```

## 6) IntelliJ

- plugins : Jakarta EE, Spring 은 ultimate 라서 현재 불가
- plugins > korea 한글팩
- build system : 오류검사
- jdk : 환경변수 설정을 자동으로?!

## 7) New project

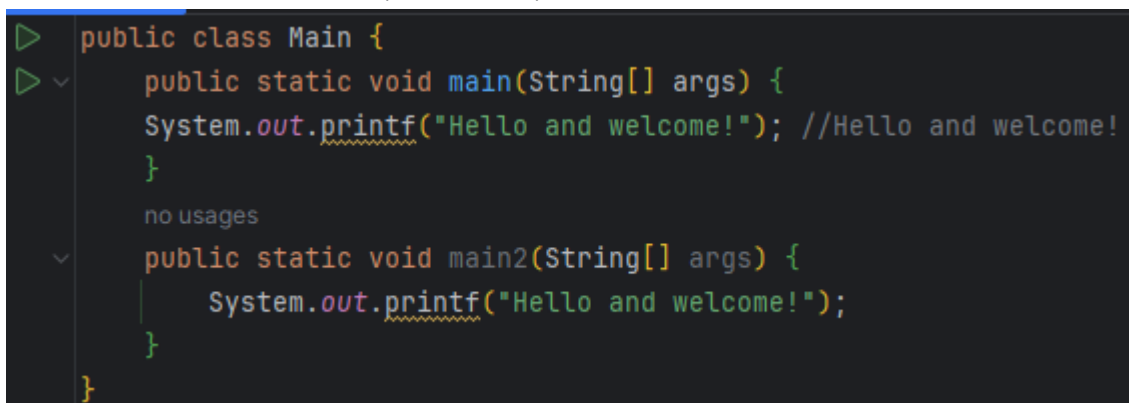
- Run : shift + F10
- 수정 제안 : alt + enter
- Break point : shift + F9
- file > settings > mouse 검색 > general > mouse control > active editor

## 8) 추천 plugin

- **Rainbow Brackets** : 코드 블록 구분
- **Nyan Progress Bar** : 로딩바가 귀여워짐
- **Atom Material Icons** : 아이콘 이빠짐
- **CodeGlance** : VS Code처럼 스크롤바 옆에 현재 파일의 코드를 작게 보여준다.
- **Codota AI Autocomplete for Java and Javascript** : AI학습 클린 코딩 자동완성
- **GitToolBox** : GIT 커밋 히스토리 보여줌 (범인찾기)
- **Key Promoter X** : 동작 하나마다 알림뜸 ( 별로 추천 안함 )
- **One Dark Theme** : 어두운 테마로 바뀜

## 9) Java

- **println()** : 기본 출력문은 **println()**은 변수의 값을 그대로 출력하므로, 값을 변환하지 않고는 다른 형식으로 출력할 수 없다.
- **printf()** : 반면에 **printf()**는 지시자를 통해 변수의 값을 여러 가지 형식으로 변환하여 출력할 수 있다.
- **Public class** Main 과 **Main.java** 은 일치해야 한다.
- **Public static void** main 은 **object** 이다.
  - 하나의 **class** 안에서는 **object** 명은 하나만 존재할 수 있다.
- 오브젝트가 두 개일 경우(**main**, **main2**)



```
public class Main {
 public static void main(String[] args) {
 System.out.printf("Hello and welcome!"); //Hello and welcome!
 }
 no usages
 public static void main2(String[] args) {
 System.out.printf("Hello and welcome!");
 }
}
```

- **main** 은 Hello and welcome! 가 출력되나, **main2** 미출력 상태
- **main2** 는 현재 no usages 상태