리눅스 MariaDB 소스 컴파일 설치 과정

■ MariaDB 설치 과정 설명

1. MariaDB에서 사용할 포트 번호에 대한 방화벽 개방

MariaDB 데이터베이스 서버를 구축하는 과정에서 데이터베이스 서버에서 사용하는 포트 번호를 방화벽 설정을 통해 개방해주지 않으면 사용자는 MariaDB 데이터베이스 서버에 접근할 수가 없게 된다. 따라서 MariaDB 서버에 필요한 포트를 개방하는 작업을 수행해야 한다. 이 과정에서 작성되는 코드는 다음과 같으며, 각 코드는 지정한 포트에 대해 영구적으로 개방한다는 의미를 지니고 있다. 또한, MariaDB에 대한 포트 번호로 63306번 포트를 사용할 것이므로 add-port 속성에 63306을 적힌 것을 확인할 수 있다.

```
==========================================================================
firewall-cmd --zone=public --permanent --add-port=63306/tcp
==========================================================================
```

2. 방화벽 리로드 및 설정된 포트 리스트 확인

방화벽에 대해 설정한 사항들을 적용하고 지정한 포트들이 잘 개방되었는지 확인하는 단계이다. 방화벽 리로드는 firewall-cmd —reload를 통해 수행되며 해당 코드를 작성하지 않을 경우, 방화벽의 설정이 갱신되지 않기에 반드시 수행해야 한다. 방화벽에 대한 리로드가 완료되었으면 firewall-cmd —list-ports를 통해 현재까지 개방된 포트 번호 리스트를 출력시킨다. 만약 포트가 제대로 개방되었다면 데이터베이스 서버에서 사용하기 위해 설정한 포트 번호(66306)가 잘 출력되는 것을 확인할 수 있을 것이다.

3. 소스 컴파일 방식의 설치를 위한 의존성 패키지 설치

```
==========================================================================
```
=> 의존성 패키지의 경우 양이 방대하기에 YUM을 사용하여 설치하였다.

=> MariaDB 자체에 대한 설치는 소스 컴파일 방식으로 설치를 진행하였다.

```
==========================================================================
```

우리는 MariaDB 데이터베이스를 설치하기 위해 소스 컴파일 방식을 사용할 것이다. 하지만 소스 컴파일을 하기 위해서는 이를 위한 의존성 패키지들을 설치해주어야 한다. 의존성 패키지란 특정 프로그램을 설치하는 데 필요한 라이브러리 패키지를 의미한다. 예를 들어 우리는 MariaDB를 소스 컴파일 방식으로 설치하기 위해 cmake나 make 등의 패키지를 이용하게 되는데 해당 패키지가 없으면 MariaDB를 설치하지 못할 수 있다. 즉, cmake나 make 등의 패키지가 MariaDB의 설치를 위한 의존성 패키지가 되는 것이다. MariaDB를 소스 컴파일 방식으로 설치하는 데 필요한 의존성 패키지는 매우 많다. 여기서 우리는 의존성 패키지를 설치해주어야 하므로 해당 과정에서만 YUM을 사용하여 의존성 패키지에 대한 설치 작업을 진행하였다(MariaDB 설치 자체는 소스 컴파일 방식으로 진행). 아래는 의존성 패키지를 설치하기 위해 사용한 코드들의 의미와 각 코드에 있는 의존성 패키지들에 대한 설명을 작성한 것이다.

[첫 번째 입력 코드]

코드 내용	
yum -y install cmake make gcc gcc-c++ ncurses-devel libevent	
코드 의미	
cmake make gcc gcc-c++ ncurses-devel libevent 등의 패키지들을 yum으로 설치한다. (-y 옵션을 통해 설치 과정에서 묻는 것에 대해 모두 yes를 입력하도록 설정)	
패키지	**설명**
cmake	cmake란 Build Process를 작성한 것으로, CMakeLists.txt에 기술된 내용을 바탕으로 각 확경에 맞는 makefile을 생성해 주는 빌드 프로그램이다. 일반적으로 cmake는 makefile을 만들 때 추가적인 파일이 추가되지 않는 한 makefile을 수정하지 않고 자동으로 생성해 준다는 장점이 존재한다. 그러나 단순히 makefile만을 생성해

	주는 것이기 때문에 Make를 추가로 해야한다.
make	make 명령어를 사용하기 위해 존재하는 패키지이다. make 명령어는 makefile이라는 파일을 참조하여 컴파일러에 명령을 전달하며 makefile에 기술된 Shell 명령어들이 순차적으로 실행될 수 하는 역할을 담당한다.
gcc	GNU 프로젝트의 오픈 소스 컴파일러 컬렉션으로 유닉스/리눅스 계열 플랫폼의 표준 컴파일러다. 리처드 스톨먼이 1987년에 만들었으며 'GNU C Compiler'의 약어로써 c언어 컴파일러를 지칭한다. 이외에도 gcc는 빌드 과정에서 여러 개의 코드를 작성해야 하는 단점이 존재한다. 따라서 과정의 번거로움을 해결하고 빌드 과정을 자동화하기 위해 make 패키지를 활용하게 된다.
gcc-c++	gcc와 마찬가지로 GNU 프로젝트의 오픈 소스 컴파일러 컬렉션으로 유닉스/리눅스 계열 플랫폼의 표준 컴파일러다. 단, gcc는 c 언어로 된 파일을 컴파일 하기 위한 목적이라면 gcc-c++는 c++ 언어로 된 파일을 컴파일하기 위한 목적에서 차이점을 보인다고 할 수 있다.
ncurses-devel	ncurses는 UNIX System V에서 사용하던 curses 라이브러리를 재작성한 라이브러리다. 해당 라이브러리가 하는 일은 터미널 화면에서 사용자 친화적인 환경을 만드는 것이다. 즉 콘솔 화면에서 메뉴바를 제어하게 한다든지, 키보드제어를 돕는 등의 구실을 하게 된다. 그러나 주목할 점은 ncurses 뒤에 devel이 붙어 있다는 것이다. 일반적으로 devel 파일은 다른 프로그램들을 위한 라이브러리 역할과 소스 코드가 컴파일되는 것을 지원하는 역할을 한다. 즉, ncurses-devel은 우리가 ncurses 라이브러리를 사용한다기보다는 소스 코드 컴파일을 통한 설치 과정에서 리눅스 서버가 컴파일 과정을 수행하기 위해 사용하는 것이라고 봐야 할 것이다.
libevent	libevent는 특정 파일 디스크럽터에 지정된 이벤트가 발생하거나, 지정된 시간이 지나서 타임아웃이 되었을 때, 미리 설정해둔 함수를 실행해 주는 메커니즘을 수행하는 패키지이다.

[두 번째 입력 코드]

코드 내용
yum -y install openssl openssl-devel gnutls-devel libxml2 libxml2-devel wget
코드 의미
openssl openssl-devel gnutls-devel libxml2 libxml2-devel wget 등의 패키지들을 yum으로 설치한다. (-y 옵션을 통해 설치 과정에서 묻는 것에 대해 모두 yes를 입력하도록 설정)

패키지	설명
openssl	OpenSSL은 네트워크를 통한 데이터 통신에 쓰이는 프로토콜인 TLS와 SSL의 오픈 소스 구현 패키지이다. C 언어로 작성된 중심 라이브러리 안에는, 기본적인 암호화 기능 및 여러 유틸리티 함수들이 구현되어 있다.
openssl-devel	소스 코드 컴파일을 통한 설치 과정에서 리눅스 서버가 컴파일을 수행할 때 필요한 openssl의 기능들을 제공하는 패키지이다.
gnutls-devel	GnuTLS는 TLS, SSL 및 DTLS 프로토콜를 구현하여 무료로 제공하는 패키지이다. openssl-devel과 마찬가지로 사용자가 gnutls 패키지를 직접 사용하기보다는 소스 코드 컴파일을 통한 설치 과정에서 리눅스 서버가 컴파일 과정을 수행하기 위해 사용하는 것이라고 봐야 할 것이다.
libxml2	Libxml2는 기존의 XML 파서들보다 다양한 인터페이스를 가지고 다양한 언어그룹에서 사용할 수 있는 기능을 포함하고 있는 XML 파서에 대한 패키지 라이브러리다.
libxml2-devel	소스 코드 컴파일을 통한 설치 과정에서 리눅스 서버가 컴파일을 수행할 때 필요한 libxml2의 기능을 제공하는 패키지이다.
wget	**'Web Get'의 약어로 웹상의 파일을 내려받을 때 패키지이다. 일반적으로 wget 패키지에서 제공하는 명령어인 wget은 비 상호작용 네트워크 다운로더의 매커니즘을 가지고 있다. 즉, 네트워크상에서 데이터를 내려받는 기능을 수행하며 HTTP, HTTPS, FTP 프로토콜을 지원한다. 이외에도 wget은 느리거나 불안정한 네트워크 환경에서도 매우 잘 작동하는 견고한 프로그램이다. 네트워크 환경이 불안해서 도중에 연결이 끊겼다면, 연결이 끊긴 시점부터 내려받는 기능도 가지고 있다.**

[세 번째 입력 코드]

코드 내용	
yum -y install git curl libcurl-devel libssl-dev libcurl4-openssl-devel	
코드 의미	
git curl libcurl-devel libssl-dev libcurl4-openssl-devel 등의 패키지들을 yum으로 설치한다. (-y 옵션을 통해 설치 과정에서 묻는 것에 대해 모두 yes를 입력하도록 설정)	
패키지	**설명**
git	git은 여러 명의 사용자가 작업을 조율하기 위해 생겨난 분산 버전 관리 시스템을 의미한다. 즉, git 패키지를 설치하면 버전 관리 시스템과 관련된 명령어를 사용할 수 있게 된다.
curl	서버와 통신할 수 있는 커맨드 명령어 툴을 제공하는 패키지이다. 일반적으로 웹 개발에 매우 많이 사용되고 있다. 이러한 curl은 수많은 프로토콜을 지원한다는 장점을 보유하고 있다. 대표적으로 HTTP, FTP, LDAP, TELNET, HTTPS, DICT와 같은 프로토콜을 지원한다. curl은 대부분의 리눅스 배포판에 기본으로 제공하고 있으며 리눅스 커뮤니티가 관리하는 패키지 관리 시스템을 이용해 간단히 설치할 수 있다.
libcurl-devel	liburl 패키지는 HTTP, FTP, LDAP, TELNET, HTTPS, DICT와 같은 프로토콜을 지원하는 클라이언트 제작을 도와주기 위해 존재하는 패키지이다. 여기서 패키지 명에 devel가 붙어 있으므로 컴파일 설치 과정에서 리눅스 서버가 컴파일을 수행할 때 필요한 libcurl의 기능을 제공하는 패키지라고 볼 수 있다.
libssl-dev	**일반적으로 OpenSSL은 libcrypto와 libssl의 두 개의 라이브러리로 구성하고 있다. 여기서 libcrypto는 혼자 사용할 수 있는 범용 암호화 라이브러리이며 libssl은 libcrypto에 의존하는 TLS 라이브러리라고 볼 수 있다.** **즉, 앞선 코드에서 openSSL을 설치하였기 때문에 libssl에 대한 라이브러리 패키지는 사용할 수 있는 상태지만, 소스 코드 컴파일 과정에서 필요한 libssl 기능들을 쉽게 사용할 수 있도록 지원하기 위해서 설치하는 패키지라고 생각하면 될 것이다.**
libcurl4-openssl-devel	libcurl4와 openssl의 기능을 제공하는 패키지이다. libcurl4은 기존의 libcurl을 담당하며 openssl은 SSL 지원을 담당하게 된다. 또한, devel로 작성되었기에 libcurl4-openssl 패키지를 직접 사용하기보다는 소스 코드 컴파일을 통한 설치 과정에서 리눅스 서버가 컴파일 과정을 수행하기 위해 사용하는 것이라고 봐야 할 것이다.

[네 번째 입력 코드]

코드 내용
yum -y install epel-release
코드 의미
epel-release 패키지를 yum으로 설치한다. (-y 옵션을 통해 설치 과정에서 묻는 것에 대해 모두 yes를 입력하도록 설정) EPEL은 Extra Packages of Enterprise Linux의 줄임말이다. 말 그대로 리눅스의 추가 패키지를 의미하며 설치 과정에서 빠트릴 수 있는 패키지를 설치하는 구실을 한다.

[다섯 번째 입력 코드]

코드 내용
yum -y install varnish
코드 의미
varnish의 패키지를 yum으로 설치한다. (-y 옵션을 통해 설치 과정에서 묻는 것에 대해 모두 yes를 입력하도록 설정) varnish는 HTTP 요청에 신속한 응답을 제공하기 위해 결과 데이터를 캐싱(Caching)하는 동작을 수행한다. 즉, 같은 요청이 들어오면 캐시 된 데이터를 반환하여 응답 시간을 줄이는 리버스 프록시(Reverse Proxy)라고 볼 수 있다. 이러한 varnish는 흔히 웹 가속기라고도 불리기도 한다.

[여섯 번째 입력 코드]

코드 내용
yum -y install jemalloc-devel
코드 의미
jemalloc-devel의 패키지를 yum으로 설치한다. (-y 옵션을 통해 설치 과정에서 묻는 것에 대해 모두 yes를 입력하도록 설정) jemalloc은 할당을 뜻하는 malloc에 제작자인 Jason Evans의 앞글자를 따서 지어진 이름이다. 현재 페이스북, 파이어폭스에서 사용하고 있는 메모리 할당 라이브러리이다. jemalloc은 메모리 단편화를 최소화하는 데 집중되어 설계되었으며 병렬화 지원을 위한 기능을 제공하고 있다.

4. cmake 버전 최신화

우리는 의존성 패키지 설치를 위해 yum을 통해 cmake를 설치하였다. 그러나 간혹 cmake가 최신 버전이 설치되지 않는 경우가 발생한다. 이럴 때는 yum remove cmake로 cmake를 삭제해주고 소스를 다운받아 설치하는 과정을 거치게 된다. 이를 위해 수행되는 각 명령어에 대한 설명은 다음과 같다.

[첫 번째 입력 코드]

코드 내용
cd /usr/local/src
코드 의미
/usr/local/src 디렉토리로 이동한다. (소스를 다운 받아 설치하는 경우 /usr/local/src 디렉토리에 다운 받는 것을 원칙으로 하고 있다.)

[두 번째 입력 코드]

코드 내용
wget https://cmake.org/files/v3.7/cmake-3.7.2.tar.Z
코드 의미
wget 명령어로 https://cmake.org/files/v3.7/cmake-3.7.2.tar.Z 경로에 있는 cmake-3.7.2의 압축 파일을 다운받는다.

[세 번째 입력 코드]

코드 내용
mv cmake-3.7.2.tar.Z cmake-3.7.2.tar.gz
코드 의미
내려받은 cmake-3.7.2의 압축 파일의 확장자를 tar.Z에서 tar.gz로 변경한다.

[네 번째 입력 코드]

코드 내용
tar -zxvf cmake-3.7.2.tar.gz
코드 의미
내려받은 cmake-3.7.2에 대한 압축 파일을 -zxvf 옵션으로 압축 해제한다. 이때 사용하는 zxvf 옵션에 대한 설명은 다음과 같다. -z 압축하거나 해제함(gz 확장자) / -x tar 압축 해제 / -v 압축하거나 해제할 때의 과정을 출력 / -f 파일의 이름 지정

[다섯 번째 입력 코드]

코드 내용
cd cmake-3.7.2
코드 의미
압축이 해제되어 생성된 cmake-3.7.2 디렉터리에 접근한다.

[여섯 번째 입력 코드]

코드 내용
./bootstrap --prefix=/usr/bin/cmake
코드 의미
/usr/bin/cmake로 설치하겠다는 의미로써 결과물로 makefile이 생성되게 된다.

[여섯 번째 입력 코드]

코드 내용
make
코드 의미
현재 디렉토리의 makefile에 대한 컴파일을 수행하며 결과물로 실행 가능한 설치 파일을 생성한다.

[일곱 번째 입력 코드]

코드 내용
make install
코드 의미
make를 통해 생성한 설치 파일을 실행하여 설치를 진행한다.

5. /etc/profile 파일에 환경변수 추가

cmake를 실행하기 위한 환경변수를 지정하는 단계이다. 우리가 환경변수를 설정하는 것은 운영체제가 컴퓨터의 어떤 경로에서든지 대상 파일을 인식하도록 하기 위해서다. 즉, cmake 실행 경로를 환경변수로 등록하는 것은 어느 경로에 있든지 운영체제가 cmake의 존재를 인식하도록 하기 위함이라고 볼 수 있다. 그러므로 환경변수 등록이 완료되면 불필요하게 cmake의 실행을 위한 모든 경로를 적어주지 않아도 쉽게 cmake 명령어만으로도 사용이 가능한 것을 확인할 수 있을 것이다. 결론적으로 우리는 환경변수를 사용하기 위해서 /etc/profile 파일에 echo 명령어로 환경변수 코드를 추가해 주었으며 코드에 관한 내용은 다음과 같다.

```
================================================================================
  echo "
  PATH=$PATH:$HOME/bin:/usr/local/apache/bin:/usr/bin/cmake/bin:
  " >> /etc/profile
================================================================================
```

6. 수정한 설정 파일(/etc/profile) 적용

리눅스 환경에서는 설정 파일을 수정하는 것만으로 내용이 적용되지는 않는다. 즉, 설정이 적용되기 위해서는 재부팅을 통한 과정을 수행하여야 하는데 설정하는 것만으로 재부팅을 하기에는 여러모로 비효율적일 수 있다. 따라서 리눅스 계열의 운영체제에서는 수정된 새로운 환경 설정 내용을 즉시 적용하기 위한 목적으로 source 명령어를 제공하고 있다. source 명령어는 source [디렉터리 경로] 형식으로 작성하며, 이에 따라 /etc/profile의 내용을 적용하기 위해 아래와 같은 코드로 source 명령어를 작성해주었다.

```
================================================================================
  source /etc/profile
================================================================================
```

7. cmake 설치 및 환경 변수 설정이 잘 적용되었는지 확인

cmake의 설치 및 환경변수 설정이 잘 되었는지 확인하는 단계이다. 이를 위해 cmake —version을 입력하여 cmake가 설치된 버전을 열람하도록 한다. cmake의 설치 및 환경변수가 잘 적용되었을 경우, 우리가 설치한 버전인 3.7.2가 출력되게 될 것이다

8. mysql 유저와 그룹 생성 및 MariaDB 데이터 디렉터리 생성

MariaDB는 보안을 위해 특정 계정 및 그룹에서 관리하도록 하고 있다. 이때 사용되는 계정과 그룹은 mysql이라는 이름으로 사용하게 된다. 즉, mysql의 이름을 갖는 계정과 그룹을 생성해 주어야 한다. 우선 mysql의 이름을 갖는 그룹을 생성해 주기 위해 'groupadd -g 406 mysql'이라는 코드를 작성해주었다. 해당 코드는 406이라는 그룹 번호를 가지는 mysql 그룹을 생성하겠다는 의미를 지니고 있다. 이후에는 'useradd -M -r -u406 -g406 -s /sbin/nologin mysql' 코드를 입력하여 mysql이라는 이름을 갖는 계정을 생성했다. 이때 사용되는 옵션에 대한 설명은 다음과 같다.

[useradd 명령어에서 사용된 옵션들]

코드	설명
-M	홈 디렉터리를 만들지 않겠다는 의미
-r	시스템 계정으로 생성하겠다는 의미

-u	계정의 UID 번호를 지정
-g	소속되는 그룹 번호를 지정
-s	로그인 시에 사용할 쉘의 경로를 지정

위에서 설명한 옵션대로 계정을 생성하게 되면 오류 없이 mysql 계정이 잘 생성될 것이다. 중요한 점은 mysql 계정은 로그인할 수 없다는 것이다. 이는 -s를 통해 로그인 시에 사용할 쉘의 경로를 /sbin/nologin로 지정하였기 때문이다. 그렇다면 /sbin/nologin이 무엇이기에 로그인할 수 없다는 것일까? 일반적으로 nologin은 shell과 ssh의 접근을 허용하지 않으며 홈디렉토리를 제공하지 않는다. 즉, 보안상 로그인할 필요가 없지만 중요한 기능을 수행하는 계정을 만들었을 때 nologin의 경로를 지정하게 된다. 그러므로 /sbin/nologin으로 -s를 지정하여 생성한 계정은 로그인할 수 없게 되는 것이다. 결론적으로 우리는 mysql의 디렉터리를 관리하는 계정을 생성해 주는 것을 목적으로 하고 있다. 따라서 타 사용자가 mysql 계정에 임의로 접근하는 것을 차단할 필요가 있으므로 해당 옵션과 경로를 지정해준 것이고 보면 될 것이다.

계정과 그룹에 대한 생성이 완료되었으면 추후에 mysql_install_db 명령어를 수행할 때 생성되는 시스템 테이블 등의 데이터를 저장하는 디렉터리를 생성한다. 여기서 시스템 테이블은 권한, 역할, 플러그인 등을 관리하는 테이블을 의미한다. 이를 위해 우리는 mkdir /mariadb-data 명령어를 통해서 루트 디렉터리 아래에 mariadb-data 디렉터리를 생성해 주었다.

9. MariaDB 소스 코드 압축 파일 다운로드 및 build 디렉터리 생성

MariaDB를 설치하기 위해 MariaDB의 소스 코드 압축 파일을 내려받는 과정이다. 이를 위해 파일을 내려받을 때 사용하기로 한 디렉터리의 경로인 cd /usr/local/src로 이동하였으며 wget 명령어를 사용하여 아래와 같은 URL에 있는 mariadb-10.4.18.tar.gz를 내려받아 주었다.

==========

https://downloads.mariadb.com/MariaDB/mariadb-10.4.18/source/mariadb-10.4.18.tar.gz

==========

해당 URL을 통해 압축 파일을 내려받았으면 tar -zxvf mariadb-10.4.18.tar.gz를 입력하여 압축을 해제하는 과정을 수행한다. 이후에는 mkdir /usr/local/src/mariadb-10.4.18/build라는 코드를 통해 생성된 mariadb-10.4.18 디렉터리 내에 build 디렉터리를 생성주고 cd 명령어로 생성한 build 디렉터리로 이동해준다.

10. cmake를 통한 컴파일 수행 및 makefile 생성

build 디렉터리에 접근을 완료했으면 cmake ../ -LH라는 코드를 입력하여 mariadb-10.4.18 디렉터리에 지정된 컴파일 옵션을 확인하는 과정을 거친다. 만약 옵션을 변경하고 싶다면 컴파일 옵션에 -D를 붙여서 갑을 주게 된다. mariadb-10.4.18 디렉터리에 지정된 컴파일 옵션을 수정하기 위해 작성한 코드와 그에 대한 설명은 다음과 같다.

[컴파일 옵션 변경 코드]

코드 내용
cmake ../ ₩ -DCMAKE_INSTALL_PREFIX=/usr/local/mysql ₩ -DMYSQL_DATADIR=/mariadb-data ₩ -DMYSQL_UNIZ_ADDR=/usr/local/mysql/tmp/mysql.sock ₩ -DINSTALL_SYSCONFDIR=/etc ₩ -DINSTALL_SYSCONF2DIR=/etc/my.cnf.d ₩ -DMYSQL_TCP_PORT=63306 ₩ -DDEFAULT_CHARSET=utf8 ₩ -DDEFAULT_COLLATION=utf8_general_ci ₩ -DWITH_INNOBASE_STORAGE_ENGINE=1 ₩

-DWITH_ARCHIVE_STORAGE_ENGINE=1 ₩
-DWITH_ARIA_STORAGE_ENGINE=1 ₩
-DWITH_BLACKHOE_STORAGE_ENGINE=1 ₩
-DWITH_FEDERATEDX_STORAGE_ENGINE=1 ₩
-DWITH_PARTITION_STORAGE_ENGINE=1 ₩
-DWITH_PERFSCHEMA_STORAGE_ENGINE=1 ₩
-DWITH_XTRADB_STORAGE_ENGINE=1 ₩
-DWITH_ZLIB=system ₩
-DWITH_READLINE=1 ₩
-DWITH_SSL=system ₩
-DWITH_MYISAM_STORAGE_ENGINE=1

라인 별 코드	설명
cmake ../ ₩	build 디렉터리의 상위 디렉터리에 설정된 설치 옵션을 컴파일하겠다는 의미이다. 이외에 ₩는 다음 중에 계속된다는 의미이며 뒤에 나오는 -D로 시작되는 내용은 설정된 설치 옵션에서 몇몇 옵션을 지정한 값으로 변경하겠다는 의미를 지니고 있다.
-DCMAKE_INSTALL_PREFIX=/usr/local/mysql ₩	make install을 통해 설치할 때 생성되는 파일과 디렉토리가 있을 설치 디렉토리를 지정한다는 의미를 지니고 있다. 해당 속성의 값으로 /usr/local/mysql의 경로로 지정하였기 때문에 MariaDB의 파일과 디렉터리는 /usr/local/mysql 경로로 설치되게 된다.
-DMYSQL_DATADIR=/mariadb-data ₩	시스템 테이블을 비롯한 Mariadb의 데이터들이 저장될 디렉터리를 지정하는 옵션이다. 해당 옵션의 값으로 이전 단계에서 생성한 mariadb-data로 지정해주었다.
-DMYSQL_UNIZ_ADDR=/usr/local/mysql/tmp/mysql.sock ₩	MariaDB의 소켓 디렉터리를 지정하는 옵션이다. 해당 옵션의 값으로 /usr/local/mysql/tmp/mysql.sock를 지정해주었으며 mysql.sock은 MariaDB에 대한 소켓 디렉터리는 의미한다. 주목할 점은 MariaDB의 소켓 디렉터리의 이름을 mysql.sock으로 쓰고 있다는 것이다. 이는 MariaDB의 개발진들이 MySQL의 소스 코드를 기반으로 MariaDB를 제작하였기 때문이다.
-DINSTALL_SYSCONFDIR=/etc ₩	MariaDB 설치 과정에서 생성할 시스템 환경 설정 디렉터리의 경로를 지정하는 옵션이다. 해당 옵션의 값으로는 /etc를 기재해주었다.
-DINSTALL_SYSCONF2DIR=/etc/my.cnf.d ₩	MariaDB 설치 과정에서 생성할 두 번째 시스템 환경 설정 디렉터리의 경로를 지정하는 옵션이다. 해당 옵션의 값으로는 /etc/my.cnf.d를 기재해주었다.

-DMYSQL_TCP_PORT=63306 ₩	MariaDB의 포트 번호를 지정하는 옵션이다. 해당 옵션에는 우리가 사용하기로 한 포트 번호인 63306을 기재해주었다.
-DDEFAULT_CHARSET=utf8 ₩	기본 문자셋을 설정하는 옵션이다. 해당 옵션의 값으로 utf8을 지정해주었다.
-DDEFAULT_COLLATION=utf8_general_ci ₩	기본 콜레이션을 설정하는 옵션이다. 여기서 콜레이션은 특정 문자 셋에 의해 데이터베이스에 저장된 값들을 비교 검색하거나 정렬 등의 작업을 위해 문자들을 서로 비교할 때 사용하는 규칙들의 집합을 의미한다. 해당 옵션에 대한 값으로는 utf8_general_ci를 작성하였다. 일반적으로 기본 문자셋이 utf8로 설정된 경우 이에 대한 콜레이션 값으로 utf8_general_ci를 자주 사용하게 된다.
-DWITH_INNOBASE_STORAGE_ENGINE=1 ₩	InnoDB를 함께 설치할지를 결정해주는 옵션이다. 여기서 옵션의 값으로 1을 지정한 것을 볼 수 있는데 이는 해당 옵션의 값을 논리형으로 받기 때문이다. 즉, 1이 적혀있으므로 InnoDB를 함께 설치하겠다는 의미를 내포하고 있다고 볼 수 있다.
-DWITH_ARCHIVE_STORAGE_ENGINE=1 ₩	아카이브 스토리지 엔진을 함께 설치할지를 결정해주는 옵션이다. 여기서 아카이브는 MySQL 관계형 데이터베이스 관리 시스템을 위한 스토리지 엔진을 의미한다. 직전 코드와 마찬가지로 MariaDB와 함께 설치하겠다는 의미인 1을 기재해주었다.
-DWITH_ARIA_STORAGE_ENGINE=1 ₩	ARIA 스토리지 엔진을 함께 설치할지를 결정해주는 옵션이다. 여기서 ARIA 스토리지 엔진은 MyISAM의 단점을 보완하기 위해서 만들어진 스토리지 엔진을 의미한다. 이를 위해 MyISAM Storage Engine의 소스 코드를 기반으로 만들어진 스토리지 엔진이라는 특징이 있다. 직전 코드와 마찬가지로 MariaDB와 함께 설치하겠다는 의미인 1을 기재해주었다.
-DWITH_BLACKHOLE_STORAGE_ENGINE=1 ₩	BLACKHOLE 스토리지 엔진을 함께 설치할지를 결정해주는 옵션이다. 주로 장비 교체 등의 작업에서 마스터에 데이터를 저장하지 않고 슬레이브에 데이터를 전달하고자 하는 경우 사용되는 스토리지 엔진이다. 직전 코드와 마찬가지로 MariaDB와 함께 설치하겠다는 의미인 1을 기재해주었다.
-DWITH_FEDERATEDX_STORAGE_ENGINE=1 ₩	FEDERATEDX 스토리지 엔진을 함께 설치할지를 결정해주는 옵션이다. FEDERATEDX 스토리지 엔진은 로컬 서버에서 원격 서버의 테이블을 연결해 마치 자신의 테이블인 것처럼 조회하고 입력하고 삭제하는 기능을 제공한다. 즉, 로컬에서 작업하지만 실제로는 원격의 서버 테이블에 실제 작업이 수행되게 되는 것이다. 직전 코드와 마찬가지로 MariaDB와 함께 설치하겠다는 의미인 1을 기재해주었다.

-DWITH_PARTITION_STORAGE_ENGINE=1 ₩	PARTITION 스토리지 엔진을 함께 설치할지를 결정해주는 옵션이다. 직전 코드와 마찬가지로 MariaDB와 함께 설치하겠다는 의미인 1을 기재해주었다.
-DWITH_PERFSCHEMA_STORAGE_ENGINE=1 ₩	MySQL 성능 스키마와 관련된 스토리지 엔진을 함께 설치할지를 결정해주는 옵션이다. 일반적으로 MySQL 성능 스키마는 낮은 수준에서 MySQL 서버 실행을 모니터링하는 기능을 수행한다. 직전 코드와 마찬가지로 MariaDB와 함께 설치하겠다는 의미인 1을 기재해주었다.
-DWITH_XTRADB_STORAGE_ENGINE=1 ₩	XtraDB 스토리지 엔진을 함께 설치할지를 결정해주는 옵션이다. XtraDB 스토리지 엔진은 Percona에서 InnoDB 소스 코드를 보완해서 만든 스토리지 엔진이다. XtraDB 스토리지 엔진은 MySQL에는 제공되지 않으며, 오직 Percona Server와 MariaDB에서만 사용할 수 있는 특징이 있다. InnoDB의 소스 코드를 변환한 것이기 때문에 InnoDB의 데이터 파일을 그대로 사용할 수 있을 정도로 호환성이 보장하고 있다. 직전 코드와 마찬가지로 MariaDB와 함께 설치하겠다는 의미인 1을 기재해주었다.
-DWITH_ZLIB=system ₩	zlib의 지원 유형을 설정해주는 옵션이다. 여기서 zlib는 Deflate 압축 알고리즘을 C 언어로 구현한 라이브러리를 의미한다. 이외에도 해당 옵션에서는 system과 bunbled를 제공하고 있는데 각각 '로컬 환경에서 zlib를 지원하는 방식을 사용하겠다'와 '설치 파일에 설정된 지원 방식으로 zlib를 사용하겠다'라는 의미를 내포하고 있다. 해당 옵션의 값으로는 로컬 환경에서 zlib를 사용할 것이기에 system을 기재해주었다.
-DWITH_READLINE=1 ₩	MariaDB에서 한글 입력을 지원하도록 설정하는 옵션으로 논리형 값을 받기 때문에 0이나 1을 기재해주게 된다. 우리는 한글 입력을 지원하도록 설정할 것이기에 값으로 1을 기재해주었다.
-DWITH_SSL=system ₩	설치 과정에서 어느 쪽의 SSL 라이브러리를 이용할 것인지 지정해주는 옵션이다. system과 bundled라는 2개의 값을 입력할 수 있으며 각각 '로컬의 라이브러리를 이용하겠다.'와 '설치 파일에 포함된 라이브러리를 이용하겠다.'라는 의미를 지니고 있다. 여기서 값을 system으로 지정할 경우 시스템에 설치된 SSL library를 이용하게 되기 때문에 관련 library가 설치되어 있어야 한다. 해당 옵션의 값으로는 로컬에 있는 라이브러리를 이용할 것이기에 system을 기재해주었다.
-DWITH_MYISAM_STORAGE_ENGINE=1 ₩	MyISAM 스토리지 엔진을 함께 설치할지를 결정해주는 옵션이다. MyISAM 스토리지 엔진은 ISAM 스토리지 엔진의 단점을 보완하기 위해 나온 업그레이드 버전으로 비-트랜잭션-세이프(non-transactional-safe) 테이블을 관리한다. 전체적으로 속도가 InnoDB보다 빠르며, 그중에서도 Select 작업 속도가 빠르므로 읽기 작업에 적합하다는 특징이 있다. 앞선 스토리지 엔진들의 옵션들과 마찬가지로 MariaDB와 함께 설치하겠다는 의미인 1을 기재해주었다.

위의 과정을 정상적으로 수행하면 MakeFile이 생성되게 된다. 그러나 오류가 발생하여 컴파일에 실패했을 경우가 가끔 발생한다. 이때에는 rm -rf /usr/local/src/mariadb-10.4.18/CMakeCache.txt라는 코드를 입력하여 CMakeCache.txt 파일을 삭제하고 다시 컴파일 하는 과정을 거쳐야 한다.

11. make 및 make install로 MariaDB 빌드 및 설치

cmake를 통해 몇몇 소스 코드 옵션을 변경하는 작업 및 컴파일까지 완료가 되었으면 현재 디렉터리의 경로인 build 디렉터리에 MakeFile이 생성되게 된다. ls 명령어를 통해 MakeFile이 잘 생성된 것을 확인하였으면 make를 입력하여 빌드를 한다. 그 뒤에는 make install를 입력하여 MariaDB 설치 작업을 수행하게 된다.

11-1. MariaDB를 설치할 때 메모리가 부족하여 오류가 나는 경우 대처 방법

해당 단계는 make install을 통해 MariaDB를 설치하는 과정에서 메모리가 부족의 문제로 에러가 발생하여 설치가 중단되는 경우에 대한 대처 방법을 서술하고 있다. 따라서 11번 단계에서 정상적으로 설치가 되었다면 해당 단계를 넘어가도 무방하다. 일반적으로 메모리가 부족한 현상을 해결하기 위해서는 메모리 swap 관련 명령어를 사용하게 된다. 여기서 메모리 swap이란 하드디스크의 일부를 메모리처럼 사용할 수 있게 해주는 기술을 의미한다. 다음은 메모리 swap을 위해 작성한 코드와 그에 대한 설명을 서술한 것이다.

[메모리 swap에 대한 코드 및 설명]

코드	설명
fallocate -l 2048M /swapfile 혹은 dd if = dev/zero of=/swapfile count=2048 bs=1M	루트 경로 밑에 swapfile을 생성하겠다는 의미이다. 이때 fallocate와 dd 명령어 중에 하나를 사용하게 된다. 두 명령어 모두 지정한 크기의 특정 파일을 생성할 때 사용한다 우선 fallocate 명령어의 경우 -l 옵션을 통해 2048M(2GB) 크기의 swap 파일을 생성하고 있다. 즉, -l 옵션은 생성할 파일의 크기를 의미한다. 다음으로 dd 명령어의 경우 1메가바이트씩 2048번을 /dev/zero에서 읽어서 /swapfile에 기록하는 것을 의미한다. 즉, 1메가바이트씩 2048번을 읽어서 swapfile 생성하고 기록하는 것이므로 2GB 크기의 swapfile이 생기게 된다.
chmod 0600 /swapfile	chmod 명령어로 swapfile에 대하여 해당 파일의 소유자(root)만 읽고 쓸 수 있도록 권한을 변경하는 코드이다. 따라서 그룹과 기타 사용자에 대한 접근 권한이 모두 막히게 된다.
mkswap /swapfile	mkswap 명령어로 생성한 2048M 크기의 swapfile를 지정하여 2048M 크기의 swap 영역을 구성한다.
swapon /swapfile	swapon 명령어를 통해 지정한 swap 영역 파일인 swapfile을 대상으로 swap 영역을 활성화한다.
free -m	메모리 영역에 등록된 정보들을 열람하는 코드이다. 여기서 -m은 total, used, free 카테고리의 출력 단위를 바이트 단위가 아닌 메가바이트 단위로 지정할 때 사용하는 옵션이다. swapon을 통해 swap 영역이 정상적으로 활성화 되었을 경우 메모리를 뜻하는 mem 이외에 swap이라는 이름의 메모리가 메모리 영역 올라간 것을 확인할 수 있다.

blkid /swapfile	swapfile에 대한 UUID를 출력해주는 코드이다. /etc/fstab에 swapfile의 정보를 기재해주고 swapon -a 코드만으로 swapfile에 대한 swap 메모리를 활성화할 수 있도록 하기 위해 사용하였다.
vi /etc/fstab	vi 편집기를 통해 /etc/fstab 파일을 열고 swapfile에 대한 정보를 기재하기 위해 사용하는 코드이다. 해당 파일에는 swapfile에 대한 UUID와 파일의 유형(swap), 백업 수행 여부(0 또는 1), 파일 체크 옵션(0, 1, 2)을 작성하게 된다.
swapoff /swapfile	swapoff 명령어를 통해 활성화된 swapfile에 대한 swap 메모리를 비활성화한다. 이는 swapon -a 명령어가 제대로 동작하는지 확인하기 위함이다.
swapon -a	/etc/fstab에 파일 시스템 유형이 swap으로 등록된 swap 메모리를 자동으로 실행하는 코드이다. 즉, /etc/fstab 파일에 swapfile에 대한 내용을 기재해주었을 경우 swapon /swapfile이 아닌 swapon -a를 입력해주어도 swapfile에 대한 스왑 메모리가 활성화되게 된다.
swapon -s	현재 활성화된 스왑 메모리의 정보를 출력하는 코드이다. 만약 스왑 메모리가 활성화되어 있지 않을 경우 아무런 결과가 출력되게 된다.

12. MariaDB에 대한 보안 설정

우리는 cmake와 make, 그리고 make install을 통해 소스 컴파일 과정으로 MariaDB 설치를 완료하였다. 하지만 아직 MariaDB를 구축하기 위한 모든 과정이 끝난 것은 아니다. 이는 MariaDB에 대한 보안 설정이 남아있기 때문이다. MariaDB의 보안 설정 작업을 쉽게 설명하기 위해 진행 순서에 따른 코드와 설명을 아래의 표에 작성해주었다.

[MariaDB의 보안 설정 코드 및 설명]

코드	설명
chown mysql:mysql -R /mariadb-data	시스템 테이블을 비롯한 mariadb의 데이터가 들어간 mariadb-data 디렉터리의 소유자 및 그룹에 대한 소유권을 각각 mysql로 설정한다. 이는 root 계정이 mariadb-data를 관리할 경우 보안상의 문제에 노출되기 쉽기 때문이다. 이외에도 -R 옵션을 사용하여 mariadb-data 디렉터리의 안에 있는 디렉터리와 파일들에 대해서도 소유권을 동일하게 설정해주었다.
chown mysql:mysql -R /usr/local/mysql	/usr/local/mysql 경로의 디렉터리의 소유자 및 그룹에 대한 소유권을 각각 mysql로 설정한다. 또한, -R 옵션을 사용하여 mysql 디렉터리의 안에 있는 디렉터리와 파일들에 대해서도 소유권을 동일하게 설정해주었다.

chmod 755 /usr/local/mysql -R	/usr/local/mysql 경로의 디렉터리의 권한 설정을 755(rwx-r-xr-x)로 설정해주었다. 또한, 직전 코드들과 마찬가지로 -R 옵션을 사용하여 mysql 디렉터리의 안에 있는 디렉터리와 파일들에 대해서도 권한 설정을 동일하게 부여해주었다.
cd /usr/local/mysql/scripts	/usr/local/mysql/scripts로 경로를 이동한다.
./mysql_install_db --user=mysql --basedir=/usr/local/mysql --datadir=/mariadb-data	scripts 디렉터리 내부에 있는 mysql_install_db를 실행시킨다. 이때 옵션으로 user와 basedir, datadir의 값을 지정하고 있다. 여기서 user는 mysqld에 대한 소유자를 지정하는 옵션이다. 즉, mysqld에 대한 소유권을 mysql이라는 사용자에게 위임하겠다는 의미이다. 다음으로 basedir는 mysql가 설치된 경로를 지정하기 위해 사용하는 옵션이다. 마지막으로 datadir은 MariaDB에서 사용하는 데이터들을 저장하는 디렉터리를 지정하는 옵션으로 여기서는 mariadb_data를 지정하였다. 일반적으로 mysql_install_db 파일은 MySQL 데이터 디렉토리를 초기화 시키고, 시스템 테이블이 존재하지 않는 경우에는 그것을 생성하는 역할을 수행한다.
mkdir -p /usr/local/mysql/innoDB/redoLogs	usr 디렉터리부터 redoLogs까지 차례차례 접근하면서 중간에 존재하지 않는 디렉터리가 있는 경우 자동으로 생성하도록 하는 코드이다. 이때 디렉터리를 자동으로 생성하기 위해 -p 옵션을 사용해주었다.
mkdir -p /usr/local/mysql/innoDB/undoLogs	usr 디렉터리부터 undoLogs까지 차례차례 접근하면서 중간에 존재하지 않는 디렉터리가 있는 경우 자동으로 생성하도록 하는 코드이다. 직전 코드와 마찬가지로 디렉터리를 자동으로 생성하기 위해 -p 옵션을 사용해주었다.
chown mysql:mysql -R /usr/local/mysql/innoDB	innoDB에 대한 개인 및 그룹을 각각 mysql로 지정하여 소유권을 위임해주는 코드이다. 따라서 해당 디렉터리는 mysql 계정 및 mysql 그룹에서 관리하게 된다. 이외에도 -R 옵션을 부여하여 innoDB 디렉터리 안에 있는 디렉터리와 파일들에 대해서도 동일하게 소유권을 변경해주었다.
mkdir /usr/local/mysql/logs	mysql 설치 디렉터리에 logs라는 로그 디렉터리를 생성한다.
mkdir /usr/local/mysql/tmp	mysql 설치 디렉터리에 tmp라는 임시 디렉터리를 생성한다.

/usr/local/mysql/support-files/mysql.server start	MariaDB 서버를 구동하기 위한 코드이다.
/usr/local/mysql/bin/mysqladmin -u root password 'css2021'	root 계정의 비밀번호를 설정한다. (해당 코드에서는 'css2021'로 설정하고 있다)
/usr/local/mysql/bin/mysql -u root -p	mysql을 실행하여 root 계정에 접속한다. (비밀번호를 입력하여 로그인이 잘 되는지 확인한 뒤 종료.)
/usr/local/mysql/bin/mysql_secure_installation	mysql_secure_installation를 실행하여 보안 설정을 진행한다. 일반적으로 외부에서 원격접속을 하는 서버를 제공하는 경우라면 mysql_secure_installation 스크립트를 통해 보안을 설정하는 것을 권장하고 있다. 실행 과정에서 나오는 질문에 대한 설명은 다음과 같다. - unix socket 인증 방식으로 변경할 것인가? [Y/n] - root 비밀번호를 설정할 것인가? [Y/n] - root를 제외한 다른 유저를 삭제할 것인가? [Y/n] - root의 원격 접속을 차단할 것인가? [Y/n] - test 데이터베이스를 삭제할 것인가? [Y/n] 제대로 된 보안 설정을 위해 우리 조는 모든 질문을 대상으로 y를 입력하여 설정을 진행해 주었다. 여기서 두 번째 질문에 대한 의문을 가질 수 있다. 우리는 사전에 root 계정에 대한 비밀번호를 설정해주었기 때문이다. 하지만 해당 비밀번호는 mysql이 제대로 구동되는지에 대한 확인을 위한 용도로 지정해준 것이다. 따라서 보안상의 취약점이 있을 수 있다. 즉, mysql_secure_installation을 통해 공식적으로 사용할 root 비밀번호를 지정하는 것이 보안상 더욱 안전하므로 해당 과정에서 비밀번호를 설정해주는 과정을 거치는 것이다.

12. MariaDB에 대한 환경 설정

보안 설정이 완료되면 마지막으로 환경 설정을 진행한다. 제대로 된 보안 설정을 위해 기존의 /etc 디렉터리에 있는 MariaDB의 설정 파일인 my.cnf 파일을 my.cnf.ori라는 이름으로 변경해주었다. 이는 my.cnf 파일을 따로 생성하여 필요한 설정 내용만 담아 가독성을 좋게 하기 위해서다. 또한, 기존의 파일을 my.cnf.ori 파일이라는 이름으로 따로 보관하여 향후 설정 과정에서 문제가 생겼을 때 사용하기 위한 백업 파일 용도로 사용하기 위한 목적도 가지고 있다. 기존의 my.cnf 파일을 my.cnf.ori라는 이름으로 변경해주었으면 /usr/local/mysql/support-files/mysql.server stop를 입력하여 MariaDB 서버를 종료한다. 이후 vi 편집기를 통해 my.cnf 파일을 생성하여 내용을 작성하고 저장하는 과정을 수행한다. 이러한 과정을 모두 수행하면 마지막으로 /usr/local/mysql/support-files/mysql.server start라는 코드를 통해 서버를 가동하는 거으로 모든 과정을 끝마치게 된다. MariaDB에 대한 설정을 위해 작성한 내용과 설명은 다음과 같다.

[my.cnf 파일에 작성된 내용]

내용
[client] port = 63306 # => 클라이언트에서 MariaDB에서 구동할 때 사용할 포트 번호 설정 (63306)

```
socket                  = /usr/local/mysql/socket/mysql.socket
# => MariaDB의 소켓 파일이 위치한 경로를 지정
# The MySQL Server
[mysqld]
port                    = 63306
# => MariaDB 서버를 구동할 때 사용할 포트 번호 설정 (63306)

socket                  = /usr/local/mysql/socket/mysql.socket
# => MariaDB의 소켓 파일이 위치한 경로를 지정

skip-external-locking
# => 외부(TCP/IP) 잠금비활성

key_buffer_size = 384M
# => 메모리에 있는 인덱스 블록에서 사용하는 버퍼의 크기를 지정
# => 해당 파일에서는 384M로 작성해주었다.

max_allowed_packet = 1M
# => mysql이 클라이언트와 통신할 때 취급 할 수 있는 데이터의 양을 의미
# => 해당 파일에서는 1M로 작성해주었다.

table_open_cache = 521
# => 서버가 하나의 테이블 캐시 인스턴스에서 유지할 수있는 최대 테이블 수를 의미
# => table_open_cache와 max_connections의 값은 반비례함

sort_buffer_size = 2M
# => MariaDB에서 인덱스 없이 정렬을 수행할 경우 디스크 버퍼에서 데이터를 정렬함
# => 일반적으로 MariaDB에서 부하가 많이 생기는 작업이 정렬
# => sort_buffer_size를 통해 정렬에 얼마만큼의 메모리 공간을 할당할지 결정하게 됨
# => 해당 파일에서는 2M로 작성해주었다.

read_buffer_size = 2M
# => MariaDB의 읽기 버퍼 사이즈를 지정함
# => 해당 파일에서는 2M로 지정해주었다.

read_rnd_buffer_size = 8M
# => 정렬된 열을 읽을 때 사용하는 버퍼의 사이즈를 지정
# => 해당 파일에서는 8M로 지정해주었다.

myisam_sort_buffer_size = 64M
# => MyISAM의 인덱스를 정렬할 때 할당할 버퍼의 크기를 지정
# => 이외에도 CREATE INDEX 또는 ALTER TABLE을 사용하여 인덱스를 지정할 때 사용되는 버퍼이기도 함
# => 해당 파일에서는 64M로 지정해주었다.

thread_cache_size = 8
```

```
# => 스레드를 재사용하기 위해 사용되는 스레드 캐시의 개수를 지정(8개)

# Try number of CPU's*2 for thread_concurrency
thread_concurrency = 8
# => 쓰레드의 개수를 지정
# => 일반적으로 CPU는 1개의 코어에 2개 쓰레드를 지니고 있음
# => 이에 따라 4개의 코어를 보유한 CPU인 경우 8을 지정하게 됨
# => 해당 파일에서는 8로 값을 설정하였다.

log-bin=mysql-bin
# => 생성할 bin 로그 파일의 이름을 지정
# => 해당 파일에서는 mysql-bin이라는 이름으로 설정하였다.

max_binlog_size = 1G
# => bin 로그 파일에 대한 용량을 지정
# => 만약 지정한 용량을 넘을 경우 앞서 저장된 로그가 지워지면서 새로운 로그가 채워지게 됨
# => 해당 파일에서는 1G로 설정하였다.

# expire_logs_days = 30
# => 로그의 보관기간

server-id  = 1
# => 마스터 서버 아이디로 고유의 값을 지정

[mysqldump]
quick
# => 버퍼를 사용하지 말고, direct로 덤프를 받음

max allowed packet = 16M
# => 허용할 최대 패킷 용량을 지정
# => 해당 파일에서는 16M로 지정하였다.

[mysql]
no-auto-rehash
# => auto-rehash 기능을 비활성화하겠다는 의미
# => auto-rehash란 테이블이나 칼럼의 이름 일부분만 작성한 뒤 TAB을 입력하여 자동 완성하는 기능을 말함

# Remove the next comment character if you are not Familiar with SQL
# safe-updates
# => 데이터의 보호를 위한 목적으로 사용하는 Safe Update Mode를 활성화

[myisamchk]  # => 여기서 설정되는 것은 myisamchk 유틸리티에 대한 설정임
key_buffer_size = 256M
# => 메모리에 있는 인덱스 블록에서 사용되는 버퍼의 크기
# => 해당 파일에서는 256M으로 지정하였다.

sort_buffer_size =256M
# => 정렬을 위해 필요한 버퍼 크기를 지정
```

```
# => 해당 파일에서는 256M으로 지정하였다.

read_buffer = 2M
# => 읽기 버퍼 사이즈를 지정함
# => 해당 파일에서는 2M로 지정해주었다.

write_buffer = 2M
# => 쓰기 버퍼 사이즈를 지정함
# => 해당 파일에서는 2M로 지정해주었다.

[mysqlhotcopy]
interactive-timeout
# => 인터렉티브 모드에서 특정 시간이 지나면 연결을 종료 하겠다는 의미
# => 인터렉티브 모드는 'mysql>'과 같은 커맨드라인을 제공하는 콘솔이나 터미널을 말함
```