DDL : Data Definition Language

데이터 정의어(Data Definition Language)

데이터 정의어(이하 DDL)는 데이터 간에 관계를 정의하여 데이터베이스 구조를 설정하는 SQL 문장

구분	명령어	설명
데이터베이스	CREATE DATABASE	데이터베이스를 생성한다.
	ALTER DATABASE	데이터베이스를 변경한다.
테이블	CREATE TABLE	테이블을 생성한다.
	ALTER TABLE	테이블을 변경한다.
	DROP TABLE	테이블을 제거한다.
테이블 스페이스	CREATE TABLESPACE	테이블 스페이스를 생성한다.
	ALTER TABLESPACE	테이블 스페이스를 변경한다.
	DROP TABLESPACE	테이블 스페이스를 제거한다.

데이터 정의어(Data Definition Language)

구분	명령어	설명
인덱스	CREATE INDEX	인덱스를 생성한다.
	ALTER INDEX	인덱스를 변경한다.
	DROP INDEX	인덱스를 제거한다.
뷰	CREATE VIEW	뷰를 생성한다.
	ALTER VIEW	뷰를 변경한다.
	DROP VIEW	뷰를 제거한다.
동의어	CREATE SYNOMYM	동의어를 생성한다.
	DROP SYNOMYM	동의어를 제거한다.
사용자	CREATE USER	사용자를 생성한다.
	ALTER USER	사용자를 변경한다.
	DROP USER	사용자를 제거한다.

데이터 정의어(Data Definition Language)

구분	명령어	설명
함수	CREATE FUNCTION	함수를 생성한다.
	ALTER FUNCTION	함수를 변경한다.
	DROP FUNCTION	함수를 제거한다.
프러시저	CREATE PROCEDURE	프러시저를 생성한다.
	ALTER PROCEDURE	프러시저를 변경한다.
	DROP PROCEDURE	프러시저를 제거한다.
타입	CREATE TYPE	타입을 생성한다.
	ALTER TYPE	타입을 변경한다.
	DROP TYPE	타입을 제거한다.

데이터 정의어(Data Definition Language)

구분	명령어	설명
권한	GRANT	사용자에게 특권을 부여한다.
	REVOKE	사용자에게 특권을 회수한다.
역할	CREATE ROLE	역할을 생성한다.
	ALTER ROLE	역할을 변경한다.
	DROP ROLE	역할을 제거한다.
객체	RENAME	테이블, 뷰, 동의어, 시퀀스 등의 스키마 객체의 이름을 변경한다.

Create Tables to Store Data

테이블 인스턴스 –테이블의 구조와 칼럼의 특성을 요약

Symbols	Explanation
PK	기본 키 열
FK	외래 키 열
FK1, FK2	동일한 테이블에 있는 두 개의 외부 키
NN	NOT NULL 열
U	고유 열

Create Tables to Store Data

Table Instance Chart

Table Name: E_EMP

Column Name	ID	LAST_NAME	FIRST_NAME	START_DATE	SALARY	MANAGER_ID	DEPT_ID
Key Type	PK					FK1	FK2
Nulls / Unique	NN, U					NN	
FK Ref Table						S_EMP	S_DEPT
FK Ref Column						ID	ID
Data Type	NUMBER	CHAR	CHAR	DATE	NUMBER	NUMBER	NUMBER
Maximum Length	7	25	25		11	7	7
Sample Data	1	Alexander	Lee	2022-09-01	1500		10
	2	Jonathan	Hong	2022-10-05	1000	1	20

Create Tables to Store Data

Syntax

```
CREATE TABLE [user.] table_name
        ({column_name datatype | table_constraint
          ,column_name datatype | table_constraint});
```

where	*table_name*	테이블 이름
	column_name	열 이름
	datatype	데이터 타입
	table_constraint	제약 조건

Create Tables to Store Data

스키마 객체 이름

- 이름 부여 시 따옴표(“ ”) 없는 식별자는 모두 대문자로 간주
- 따옴표 있는 식별자는 대소문자를 구분

모두 다른 식별자	모두 같은 식별자
department **“department”** **“Department”**	**department** **DEPARTMENT** **“DEPARTMENT”**

식별자를 기술 할 때 규칙

- 길이가 **30bytes**를 넘으면 안된다.
- 따옴표 없는 식별자는 알파벳, 한글, 숫자, 언더바(_), $, #만 사용. 단, 숫자, ‘$’, ‘#’는 첫 글자로 올 수 없다.
- 따옴표 있는 식별자는 공백을 포함한 어떤 문자 사용 가능. 다만, 큰따옴표(“ ”)는 사용 불가능.
- 하나의 네임스페이스 안에 서로 다른 두 객체가 동일한 이름을 사용할 수 없다.

Create Tables to Store Data

Tibero에서 제공하는 데이터 타입

구분	데이터 타입
문자형	CHAR, VARCHAR, VARCHAR2, NCHAR, NVARCHAR, NVARCHAR2, RAW, LONG, LONG RAW
숫자형	NUMBER, INTEGER, FLOAT
날짜형	DATE, TIME, TIMESTAMP, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE
간격형	INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND
대용량 객체형	CLOB, BLOB, XMLTYPE
내재형	ROWID

Create Tables to Store Data

Datatype	Description
CHAR(s)	고정된 문자열 길이, 최대 2,000자까지 선언, 문자열의 길이가 0인 값은 NULL로 인식
	CHAR(size[BYTE | CHAR]) -> EXAM CHAR(10)
VARCHAR2(s)	가변 문자열 길이, 최대 4,000자까지 선언, 문자열의 길이가 0인 값은 NULL로 인식
	VARCHAR2(size[BYTE | CHAR]) -> EXAM VARCHAR2(10)
VARCHAR(s)	VARCHAR2 타입과 동일
LONG	VARCHAR2와 비슷하지만, 최대 2GB까지 선언
DATE	연도는 BC 9,999 ~ AD 9,999까지 표현,
NUMBER(p,s)	정수 또는 실수를 저장, 음양으로 절댓값이 1.0×10^{-130}보다 크거나 같고, 1.0×10^{126}보다 작은 38자리의 수를 표현할 수 있으며 0과 ±무한대를 포함한다.

Create Tables to Store Data

데이터 무결성 제약 조건

Constraint	Description
NOT NULL	NULL값을 허용하지 않고, 반드시 데이터를 입력. 제약조건이 NULL이면 해당 컬럼은 NULL값을 허용.
UNIQUE	해당 칼럼이 중복되는 데이터가 존재할 수 없는 유일성을 보장하는 제약조건
PRIMARY KEY	NOT NULL, UNIQUE 제약 조건의 결합과 같다. 테이블 또는 뷰는 단 한 개의 PRIMARY KEY 제약조건을 가질 수 있다.
FOREIGN KEY	같은 테이블 또는 서로 다른 두 개 테이블의 키 컬럼 사이의 관계
CHECK	expr로 표현한 조건이 항상 참이 되도록 유지. 특정 조건을 평가 후 만족하지 못하면 에러 발생

Create Tables to Store Data

EXAMPLE

Table Name: S_REGION

Column Name	ID	NAME
Key Type	PK	
Nulls/Unique	NN, U	NN, U
Sample data	1	North America
	2	South America
	3	Africa/Middle East
	4	Asia
	5	Europe

```
SQL> CREATE TABLE s_region
  2     ( id                          NUMBER(7),
  3      name                         VARCHAR2(50)
  4               CONSTRAINT s_region_name_nn NOT NULL,
  5               CONSTRAINT s_region_id_pk PRIMARY KEY (id),
  6               CONSTRAINT s_region_name_uk UNIQUE (name));
Table 'S_REGION' created.
```

Create Tables to Store Data

EXAMPLE

Table Name: S_DEPT

Column Name	deptno	dname	loc
Key Type	PK		
Nulls/Unique			
Datatype	NUMBER	VARCHAR2	VARCHAR2
Maximum Length	2	14	13
Sample data	10	ACCOUNTING	NEW YORK
	20	RESEARCH	DALLAS
	30	SALES	CHICAGO
	40	OPERATIONS	BOSTON

```
SQL> CREATE TABLE s_dept
  2     ( deptno                    NUMBER(2),
  3       dname                     VARCHAR2(14),
  4       loc                       VARCHAR2(13),
  5               CONSTRAINT s_dept_pk PRIMARY KEY(deptno));
Table 'S_DEPT' created.
```

Create Tables to Store Data

EXAMPLE

Table Name: S_DEPT

```
INSERT INTO S_DEPT VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO S_DEPT VALUES (20, 'RESEARCH',   'DALLAS');
INSERT INTO S_DEPT VALUES (30, 'SALES',      'CHICAGO');
INSERT INTO S_DEPT VALUES (40, 'OPERATIONS', 'BOSTON');
```

Create Tables to Store Data

EXAMPLE

Table Name: S_EMP

Column Name	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
Key Type	PK			FK1				FK2
Nulls/Unique	NN, U	NN						
FK Ref Table				S_EMP				S_DEPT
FK Ref Column				EMPNO				DEPTNO
Datatype	NUMBER	VARCHAR2	VARCHAR2	NUMBER	DATE	NUMBER	NUMBER	NUMBER
MaxLength	4	10	9	4		7	7	2
Sample data	7839	KING	PRESIDENT	null	1981-11-17	5000	null	10
	7566	JONES	MANAGER	7839	1981-02-04	2975	null	20
	7902	FORD	ANALYST	7566	1981-03-12	3000	null	20

Create Tables to Store Data

```
SQL> create table s_emp
( empno NUMBER (7) CONSTRAINT s_emp_empno_nn NOT NULL,
  ename VARCHAR2 (10) CONSTRAINT s_emp_ename_nn NOT NULL,
  job VARCHAR2 (9),
  mgr NUMBER (4),
  hiredate DATE,
  sal NUMBER (7),
  comm NUMBER (7),
  deptno NUMBER (2),
  constraint s_emp_id_pk PRIMARY KEY (empno),
  constraint s_emp_mgr_fk FOREIGN KEY(mgr) REFERENCES s_emp(empno),
  constraint s_emp_deptno_fk FOREIGN KEY(deptno) REFERENCES s_dept(deptno));
```

Create Tables to Store Data

EXAMPLE

Table Name: S_EMP

```
INSERT INTO S_EMP VALUES (7839,'KING','PRESIDENT',NULL,'81-11-17',5000,NULL,10);
INSERT INTO S_EMP VALUES (7698,'BLAKE','MANAGER',7839,'81-05-01',2850,NULL,30);
INSERT INTO S_EMP VALUES (7782,'CLARK','MANAGER',7839,'81-05-09',2450,NULL,10);
INSERT INTO S_EMP VALUES (7566,'JONES','MANAGER',7839,'81-04-01',2975,NULL,20);
INSERT INTO S_EMP VALUES (7654,'MARTIN','SALESMAN',7698,'81-09-10',1250,1400,30);
INSERT INTO S_EMP VALUES (7499,'ALLEN','SALESMAN',7698,'81-02-11',1600,300,30);
INSERT INTO S_EMP VALUES (7844,'TURNER','SALESMAN',7698,'81-08-21',1500,0,30);
INSERT INTO S_EMP VALUES (7900,'JAMES','CLERK',7698,'81-12-11',950,NULL,30);
INSERT INTO S_EMP VALUES (7521,'WARD','SALESMAN',7698,'81-02-23',1250,500,30);
INSERT INTO S_EMP VALUES (7902,'FORD','ANALYST',7566,'81-12-11',3000,NULL,20);
INSERT INTO S_EMP VALUES (7369,'SMITH','CLERK',7902,'80-12-09',800,NULL,20);
INSERT INTO S_EMP VALUES (7788,'SCOTT','ANALYST',7566,'82-12-22',3000,NULL,20);
INSERT INTO S_EMP VALUES (7876,'ADAMS','CLERK',7788,'83-01-15',1100,NULL,20);
INSERT INTO S_EMP VALUES (7934,'MILLER','CLERK',7782,'82-01-11',1300,NULL,10);
```

CONFIRM THE STRUCTURE OF A TABLE

테이블 구조 확인

Syntax

DESCRIBE *table_name*

Example

```
SQL> DESCRIBE s_emp;

COLUMN_NAME                    TYPE           CONSTRAINT
-------------------------------------- ----------------- --------------------
EMPNO                          NUMBER(7)      PRIMARY KEY
                                               NOT NULL
ENAME                          VARCHAR(10)    NOT NULL
JOB                            VARCHAR(9)
MGR                            NUMBER(4)      REFERENTIAL
HIREDATE                       DATE
SAL                            NUMBER(7)
COMM                           NUMBER(7)
DEPTNO                         NUMBER(2)      REFERENTIAL

INDEX_NAME                     TYPE           COLUMN_NAME
------------------------------ ---------------------- --------------------
S_EMP_ID_PK                    NORMAL         EMPNO
```

ADD A COLUMN TO A TABLE

테이블에 칼럼 추가하기

Syntax

```
ALTER TABLE table_name
ADD ( column_name datatype
[, column_name datatype]…)
```

Example

```
SQL> ALTER TABLE s_region
  2  ADD (comments VARCHAR2 (255));
Table 'S_REGION' altered.
```

MODIFY A COLUMN

테이블에 존재하는 칼럼 속성 변경
데이터 타입, 기본값, 제약조건 변경

Syntax

```
ALTER TABLE table_name
MODIFY ( column_name datatype
[, column_name datatype]…)
```

Example

S_EMP 테이블의 ENAME 칼럼의 길이 20으로 변경 및 SAL 값은 NULL이 될 수 없음

```
SQL> ALTER TABLE s_emp
    2    MODIFY (ename VARCHAR2 (20));
Table 'S_EMP' altered.

SQL> ALTER TABLE s_emp
    2    MODIFY (SAL NOT NULL);
Table 'S_EMP' altered.
```

ADD AND REMOVE DATA CONSTRAINTS

Example

S_EMP 테이블의 외래키 삭제 및 추가

```
SQL> ALTER TABLE s_emp
  2  DROP CONSTRAINT s_emp_mgr_fk;
Table 'S_EMP' altered.

SQL> ALTER TABLE s_emp
  2  ADD CONSTRAINT s_emp_mgr_fk
  3  FOREIGN KEY (mgr) REFERENCES s_emp(empno);
Table 'S_EMP' altered.
```

DROP A TABLE

DROP TABLE 명령어를 사용해 데이터베이스에서 테이블 제거.
DROP TABLE 명령어는 한번 실행되면 취소할 수 없다.

Syntax

```
DROP TABLE table_name
```

Example

S_REGION 테이블 삭제

```
SQL> DROP TABLE s_region;

Table 'S_REGION' dropped.
```

Simplify Data Access with Views

- 테이블 뷰를 만들어 논리적 하위 집합 또는 데이터 조합을 표시한다.
- 뷰는 실제 데이터가 포함되지 않는다.
- 뷰 이름은 테이블과 같은 네임스페이스를 사용하므로 스키마 내 다른 이름과 중복되면 안된다.

- 장점
  - 접근 제어로 보안 제공
  - 데이터 관리가 편리

- 단점
  - 삽입, 삭제, 갱신, 연산에 제약이 있다.

CREATE VIEWS

Syntax

```
CREATE VIEW view_name [ (alias, [alias]…)
AS query
WHERE
WITH CHECK OPTION
```

where	*view_name*	뷰 이름
	alias	별칭
	query	SELECT 문
	WITH CHECK OPTION	해당 옵션을 사용하면 INSERT/UPDATE 가능
	WITH READ ONLY	읽기 전용 뷰
	constraint	CHECK OPTION에 할당된 제약조건

CREATE VIEWS

Example

부서 번호가 10인 직원 번호, 이름, 직무가 포함된 뷰를 생성

```
SQL> CREATE VIEW empvu10
  2 AS SELECT empno, ename, job
  3 FROM s_emp
  4 WHERE deptno = 10;

View 'EMPVU10' created.

SQL> SELECT *
  2 FROM empvu10;

    EMPNO ENAME              JOB
---------- -------------------- ---------
     7782 CLARK              MANAGER
     7839 KING             PRESIDENT
     7934 MILLER             CLERK

3 rows selected.
```

CREATE VIEWS

Example

부서 번호가 20인 뷰를 생성. 직원 번호는 ID , 이름은 EMPLOYEE, 직무는 TITLE로 표현.

```
SQL> CREATE VIEW empvu20 (id, employee, title)
  2 AS SELECT empno, ename, job
  3 FROM s_emp
  4 WHERE deptno = 20;

View 'EMPVU20' created.

SQL> SELECT *
  2 FROM empvu20;

        ID EMPLOYEE           TITLE
---------- -------------------- ---------
      7369 SMITH              CLERK
      7566 JONES              MANAGER
      7788 SCOTT              ANALYST
      7876 ADAMS              CLERK
      7902 FORD               ANALYST

5 rows selected.
```

CREATE VIEWS

Example

월급이 1500 이상인 뷰를 생성. 직원 번호는 ID , 이름은 NAME, 월급은 MONTHLY_SALARY로 표현.

```
SQL> CREATE VIEW salvu1500
   AS SELECT empno ID, ename NAME, sal MONTHLY_SALARY
   FROM s_emp
   WHERE sal >= 1500;

View 'SALVU1500' created.

SQL> SELECT *
  2 FROM salvu1500;

        ID NAME       MONTHLY_SALARY
---------- ---------- --------------
      7499 ALLEN              1600
      7566 JONES              2975
      7698 BLAKE              2850
      7782 CLARK              2450
      7788 SCOTT              3000
      7839 KING               5000
      7844 TURNER             1500
      7902 FORD               3000

8 rows selected.
```

CREATE VIEWS

Example

S_EMP 테이블에서 부서 번호가 30인 뷰를 'WITH CHECK OPTION'을 이용하여 생성

```
SQL> CREATE VIEW empvu30
  2 AS SELECT *
  3 FROM s_emp
  4 WHERE deptno = 30
  5 WITH CHECK OPTION;
```

부서 번호를 20으로 업데이트

```
UPDATE empvu30
  SET deptno=20
  WHERE deptno=30;
TBR-10010: Statement does not satisfy the WHERE clause of the view.
```

부서 번호가 30인 새로운 사원 정보 입력

```
SQL> INSERT INTO empvu30
   VALUES (9999, 'TABA', 'STUDENT', NULL, '22-10-05', 1500, NULL, 30);

1 row inserted.
```

CONFIRM VIEW NAMES AND STRUCTURES

USER_VIEWS에서 현재 사용자에 속한 뷰의 정보를 조회할 수 있다.

Example

```
SQL> DESCRIBE user_views;

COLUMN_NAME                    TYPE           CONSTRAINT
------------------------------ -------------- ----------
VIEW_NAME                      VARCHAR(128)
TEXT                           LONG
```

Example

```
SQL> SELECT * FROM user_views;

VIEW_NAME
------------------------------------------------------------------------------
TEXT
------------------------------------------------------------------------------
SALVU1500
SELECT empno ID, ename NAME, sal MONTHLY_SALARY
   FROM s_emp
   WHERE sal >=
```

DROP A VIEW

Syntax

```
DROP VIEW view_name
```

Example

EMPVU10 뷰 삭제

```
SQL> DROP VIEW empvu10;

View 'EMPVU10' dropped.
```

Generate Primary Key Value

- 유일한 연속적 값을 생성할 수 있는 스키마 객체
- 기본 키 또는 유일 키에 값을 넣을 때 사용

Syntax

```
CREATE SEQUENCE sequence_name
   [ INCREMENT by n ]
   [ START WITH n ]
   [ {MAXVALUE n | NOMAXVALUE } ]
   [ {CYCLE | NOCYCLE } ]
   [ {CACHE n | NOCACHE } ]
   [ {ORDER | NOORDER } ]
```

where	*table_name*	테이블 이름

Generate Primary Key Value

where	*table_name*	테이블 이름
	INCREMENT BY	시퀀스 간격(default : 1)
	START WITH	시퀀스 시작 값
	MAXVALUE	시퀀스 최댓값
	NOMAXVALUE	최댓값 지정 x
	CYCLE	최댓값 도달 시 재시작
	CACHE	캐시를 사용해서 미리 할당 (default : 20)
	ORDER	시퀀스 값 순서 유지

Generate Primary Key Value

Example

50부터 시작하고 10씩 증가하는 시퀀스 생성

```
SQL> CREATE SEQUENCE s_dept_id
  2 MINVALUE 1
  3 MAXVALUE 99999
  4 INCREMENT BY 10
  5 START WITH 50
  6 NOCACHE
  7 NOORDER
  8 NOCYCLE;
```

CONFIRM SEQUENCES

USER_SEQUENCES Columns

Column	Description
SEQUENCE_NAME	시퀀스 이름
MIN_VALUE	최솟값
MAX_VALUE	최댓값
INCREMENT_BY	증가 값
CYCLE_FLAG	반복 유무
ORDER_FLAG	순서 유무
CACHE_SIZE	캐시할 시퀀스 번호 수
LAST_NUMBER	디스크에 기록된 마지막 시퀀스 번호

CONFIRM SEQUENCES

Example

```
SQL> DESC user_sequences;

COLUMN_NAME                    TYPE           CONSTRAINT
------------------------------ -------------- ----------
SEQUENCE_NAME                  VARCHAR(128)
MIN_VALUE                      NUMBER
MAX_VALUE                      NUMBER
INCREMENT_BY                   NUMBER
CYCLE_FLAG                     VARCHAR(1)
ORDER_FLAG                     VARCHAR(1)
IF_AVAIL                       VARCHAR(1)
CACHE_SIZE                     NUMBER
LAST_NUMBER                    NUMBER
```

CONFIRM SEQUENCES

Example

```
SQL> SELECT sequence_name, min_value, max_value, increment_by, cycle_flag
  2 FROM user_sequences;

SEQUENCE_NAME MIN_VALUE  MAX_VALUE INCREMENT_BY CYCLE_FLAG
------------------------------------------------------------------------------------------------------------
 S_EMP_ID                       1           99999                    10                N


1 row selected.
```

REFERENCE PRIMARY KEY VALUES

INSERT 명령에서 시퀀스를 참조하여 값을 자동으로 생성

Expression	Description
sequence_name.NEXTVAL	시퀀스의 다음 값을 반환
sequence_name.CURRVAL	시퀀스의 마지막 값은 반환

Example

새로운 부서 생성

```
SQL> INSERT INTO s_dept
  2  VALUES (s_dept_id.NEXTVAL, 'HR', 'SEOUL');

SQL> INSERT INTO s_dept
   2 VALUES (s_dept_id.NEXTVAL, 'FINANCE', 'MILPITAS');

SQL> SELECT * FROM s_dept;

    DEPTNO DNAME         LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
        50 HR             SEOUL
        60  FINANCE       MILPITAS
```

DROP A SEQUENCE

Syntax

```
DROP SEQUENCE sequence_name
```

Example

S_DEPT_ID 시퀀스 삭제

```
SQL> DROP SEQUENCE s_dept_id;

Sequence 'S_DEPT_ID' dropped.
```

Improve Query Performance

- 데이터베이스 테이블에 하나 이상의 인덱스를 작성하여 일부 쿼리 성능을 향상 시킬 수 있다.
- 자주 사용되는 **WHERE** 조건이나 **JOIN**
- 많은 양의 데이터 값을 가진 열
- 테이블 전체 데이터 중 **10-15%** 데이터를 처리하는 경우 효과적
- 테이블이 작거나 자주 업데이트 되는 경우 인덱스는 비효율적.

데이터베이스에서 테이블 데이터가 액세스 되는 방법

Access Method	Description
by ROWID	데이터의 정확한 위치를 나타내는 행 주소를 사용한 방법
FULL-TABLE SCAN	테이블의 모든 행을 순차적으로 검색하는 방법
by INDEX	열 값의 정렬된 트리 구조를 사용한 이진 검색

Improve Query Performance

Example

ROWID로 테이블 데이터 액세스

```
SQL> SELECT ename
  2 FROM s_emp
  3 WHERE rowid = 'AAAArBAACAAAABFAAK';
```

FULL-TABLE SCAN 으로 테이블 데이터 액세스

```
SQL> SELECT ename
  2 FROM s_emp;
```

CREATE INDEXS

UNIQUE 인덱스를 만들어 칼럼에 중복될 수 없는 유일 값 보장

Syntax

CREATE INDEX *index_name*
ON *table_name (column_name [, column_name] ...)*

where	*index_name*	인덱스 이름
	table_name	테이블 이름
	column_name	열 이름

Example

직원이름 열에 인덱스 생성

```
SQL> CREATE INDEX s_emp_ename_i
  2 ON s_emp(ename);

Index 'S_EMP_ENAME_I' created.
```

CONFIRM THE EXISTENCE OF INDEXES

USER_INDEX에서 인덱스 정보 확인

USER_INDEX 테이블

Column	Description
INDEX_NAME	인덱스 이름
TABLE_OWNER	인덱스 소유자
TABLE_NAME	인덱싱된 개체 이름
TABLE_TYPE	인덱싱된 개체 유형
UNIQUESNESS	인덱스의 고유성: UNIQUE or NONUNIQUE

CONFIRM THE EXISTENCE OF INDEXES

USER_INDEX에서 인덱스 정보 확인

```
SQL> DIESCRIBE user_indexes;

COLUMN_NAME                    TYPE           CONSTRAINT
------------------------------ -------------- ----------
INDEX_NAME                     VARCHAR(128)
INDEX_TYPE                     VARCHAR(26)
TABLE_OWNER                    VARCHAR(128)
TABLE_NAME                     VARCHAR(128)
TABLE_TYPE                     VARCHAR(9)
UNIQUENESS                     VARCHAR(9)
```

CONFIRM THE EXISTENCE OF INDEXES

Example

생성한 인덱스 표시

```
SQL> SELECT index_name, uniqueness
  2 FROM user_indexes
  3 WHERE table_name = 'S_EMP';

INDEX_NAME                                    UNIQUENESS
--------------------------------------------------------------------------------
S_EMP_ID_PK                                   UNIQUE
S_EMP_ENAME_I                                 NONUNIQUE
```

	차이점	공통점
PK(Primary Key)	NOT NULL	중복될 수 없는 유일값
	하나의 PK	
	OBJECT – CONSTRAINT	
UNIQUE INDEX	NULL	
	여러 개 생성 가능	
	OBJECT – INDEX	

DROP A INDEX

Syntax

```
DROP INDEX index_name
```

Example

S_DEPT_ID 시퀀스 삭제

```
SQL> DROP INDEX s_emp_ename_i;

Index 'S_EMP_ENAME_I' dropped.
```

Control User Access: Overview

- 데이터베이스 관리자는 사용자에게 **SQL** 보안 명령을 사용해 테이블에 대한 액세스 권한을 제공

- Control User Access
  - 데이터베이스에 대한 권한 제공
  - 테이블 및 시퀀스와 같은 사용자 개체에 대한 액세스를 제공하고 제거
  - 데이터 사전에서 주어진 권한 및 받은 권한 확인
  - 데이터베이스 개체에 대한 동의어 또는 대체 이름 작성

SYSTEM PRIVILEGES: OVERVIEW

- 데이터베이스 관리자는 사용자에게 시스템 권한을 부여하여 사용자는 특정 작업을 수행할 수 있다.
- 시스템 권한은 명령을 실행할 수 있는 권한이다.

Type of System Privileges

System Privilege	Description
In One's Own Schema	자신의 스키마에 테이블 및 시퀀스를 생성할 수 있는 권한
On all Objects of a Specified Type	모든 스키마에서 테이블 생성 및 테이블 또는 뷰를 업데이트 할 수 있는 권한
On the System or a User	사용자를 생성할 수 있는 권한

SYSTEM PRIVILEGES: OVERVIEW

- **60**개가 넘는 고유한 시스템 권한이 있다.
- 각 시스템 권한을 통해 사용자는 특정 작업을 수행할 수 있다.

Type of System Privileges

Class	System Privilege	Operations Permitted
SESSION	CREATE SESSION	데이터베이스 연결 허용
TABLE	CREATE TABLE	테이블 및 인덱스 생성
		CONNECT, DML, DROP, ALTER, TRUNCATE 가능
TABLE	SELECT ANY TABLE	모든 스키마에 모든 테이블, 뷰 쿼리 사용 가능

GRANT SYSTEM PRIVILEGES

- 데이터베이스 관리자는 **GRANT SQL** 명령을 사용하여 사용자 및 역할 권한을 부여하고 취소할 수 있다.

Syntax

GRANT *system_priv* TO [*user, role, PUBLIC*] [*WITH ADMIN OPTION*]

where	*system_priv*	부여되는 시스템 권한
	TO	권한을 부여 받는 대상
	user	일반 사용자
	role	권한 받을 역할
	PUBLIC	권한 받을 공유 사용자
	WITH ADMIN OPTION	사용할 수 있는 특권, 남에게 부여할 수 있는 관리 권한

GRANT SYSTEM PRIVILEGES

Example

실습을 위한 새로운 사용자 생성

```
SQL>  CREATE USER scott
   2 IDENTIFIED by tibero;
```

GRANT SQL 명령을 사용해 scott 사용자의 스키마에 테이블을 생성할 수 있는 권한 부여

```
SQL>  GRANT CREATE SESSION, CREATE TABLE TO scott;

Granted.
```

scott 사용자에게 스키마의 테이블을 변경할 수 있는 권한 부여

```
SQL>  GRANT ALTER ANY TABLE TO scott;

Granted.
```

CONFIRM SYSTEM PRIVILEGES GRANTED

DBA_SYS_PRIVS에서 시스템 권한 확인

```
SQL> DIESCRIBE dba_sys_privs;

COLUMN_NAME                    TYPE          CONSTRAINT
-------------------------------- ----------------- --------------------
GRANTEE                        VARCHAR(128)
PRIVILEGE                      VARCHAR(40)
ADMIN_OPTION                   VARCHAR(3)
```

scott 사용자에게 부여된 권한 조회

```
SQL>  SELECT grantee, privilege FROM dba_sys_privs
    2 WHERE grantee='SCOTT';
GRANTEE                                  PRIVILEGE
--------------------------------------------------------------------------------
SCOTT                                    CREATE SESSION
SCOTT                                    CREATE TABLE
SCOTT                                    ALTER ANY TABLE

3 rows selected.
```

OBJECT PRIVILEGES

데이터베이스 관리자가 사용자에게 부여할 수 있는 객체 권한으로는 테이블, 뷰, 시퀀스, 프로시저가 있다.

스키마 객체 특권	테이블	뷰	시퀀스	PSM 프로그램 (프러시저, 함수 등)	디렉터리
SELECT	O	O	O		
INSERT	O	O			
ALTER	O		O		
UPDATE	O	O			
DELETE	O	O			
TRUNCATE	O				
EXECUTE				O	
INDEX	O				
REFERENCES	O	O			
READ					O
WRITE					O

GRANT OBJECT PRIVILEGES

- 데이터베이스 관리자는 **GRANT** 명령을 사용하여 사용자 및 역할 권한을 부여하고 취소할 수 있다.

Syntax

GRANT *object_priv* ON [*column*] OBJECT TO [*user, role, PUBLIC*] [*WITH GRANT OPTION*]

where	*object_priv*	부여되는 객체 권한
	column	특정 객체 일부 칼럼
	(username.)OBJECT	스키마 객체 권한 대상이 되는 객체
	TO	객체 권한을 부여 받는 사용자
	WITH GRANT OPTION	부여 받은 권한을 다른 사용자에게 부여할 수 있는 권한

GRANT OBJECT PRIVILEGES

Example

scott에게 S_EMP 테이블을 조회 할 수 있는 권한 부여

```
SQL> GRANT SELECT ON s_emp TO scott;
Granted.
```

scott에게 S_EMP 테이블에 직원 번호, 직원 이름, 부서 번호를 삽입할 수 있는 권한과 월급을 수정할 수 있는 권한 부여

```
SQL> GRANT INSERT(empno, ename, deptno),
  2 UPDATE(sal)
  3 ON s_emp TO scott;

Granted.
```

GRANT OBJECT PRIVILEGES

Example

scott에게 S_DEPT 테이블을 조회 할 수 있는 권한과 다른 사람에게 동일한 권한을 부여할 수 있는 권한 부여

```
GRANT SELECT, INSERT
ON s_dept
TO scott
WITH GRANT OPTION;
```

scott이 시스템의 모든 사용자에게 sys의 S_DEPT 테이블을 조회할 수 있는 권한 부여

```
SQL> conn scott
Enter Password:

SQL> GRANT SELECT
  2 ON sys.s_dept
  3 TO PUBLIC;

Granted.
```

CONFIRM OBJECT PRIVILEGES GRANTED

USER_TAB_PRIVS_MADE Columns

Column	Description
GRANTEE	권한이 부여된 사용자 이름
TABLE_NAME	객체 이름
GRANTOR	권한을 부여한 사용자 이름
PRIVILEGE	부여된 권한
GRANTABLE	WITH GRANT OPTION 부여 유무

```
SQL> DESCRIBE user_tab_privs_made

COLUMN_NAME                    TYPE           CONSTRAINT
-------------------------------------- ----------------- --------------------
GRANTEE                        VARCHAR(128)
TABLE_NAME                       VARCHAR(128)
GRANTOR                         VARCHAR(128)
PRIVILEGE                      VARCHAR(40)
GRANTABLE                        VARCHAR(3)
```

CONFIRM OBJECT PRIVILEGES GRANTED

Example

현재 사용자가 scott에게 부여된 객체 조회

```
SQL> SELECT *
  2 FROM user_tab_privs_made
  3 WHERE GRANTEE = 'SCOTT';

GRANTEE                        TABLE_NAME  GRANTOR  PRIVILEGE     GRANTABLE
------------------------------ ---------- -------- ---------- ------------
SCOTT                          S_DEPT     SYS      INSERT     YES
SCOTT                          S_DEPT     SYS      SELECT     YES
SCOTT                          S_EMP      SYS      SELECT     NO

3 rows selected.
```

CONFIRM OBJECT PRIVILEGES GRANTED

Example

사용자 scott이 자신에 부여된 객체 조회

```
SQL> SELECT *
  2 FROM user_tab_privs_recd;

OWNER                TABLE_NAME          GRANTOR             PRIVILEGE         GRANTABLE
-------------------------------------------------------------------------------------------------------------------
SYS                  S_EMP               SYS                 SELECT            NO
SYS                  S_DEPT              SYS                 SELECT            YES
SYS                  S_DEPT              SYS                 INSERT            YES

3 rows selected.
```

REMOVE OBJECT PRIVILEGES

SQL 명령어 REVOKE를 사용해 다른 사용자에게 부여된 권한을 제거

Syntax

```
REVOKE privilege, privilege…
ON object_name
FROM [user1_name, user2_name … | PUBLIC | role]
[CASCADE CONSTRAINTS]
```

where	CASCADE CONSTRAINTS	REFERENCE 스키마 객체 권한을 회수하는 경우
		참조 무결성 제약조건을 지운 뒤 권한 회수 (CASCADE CONSTRAINT 을 사용하지 않고, 회수하면 에러 발생)

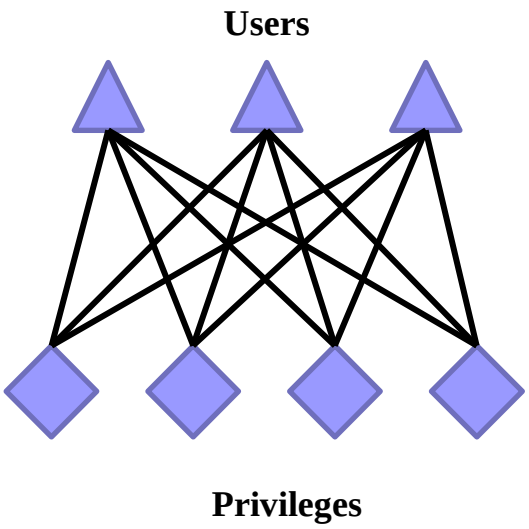
Example

```
SQL> REVOKE SELECT, INSERT
  2 ON s_dept
  3 FROM scott;

Revoked.
```

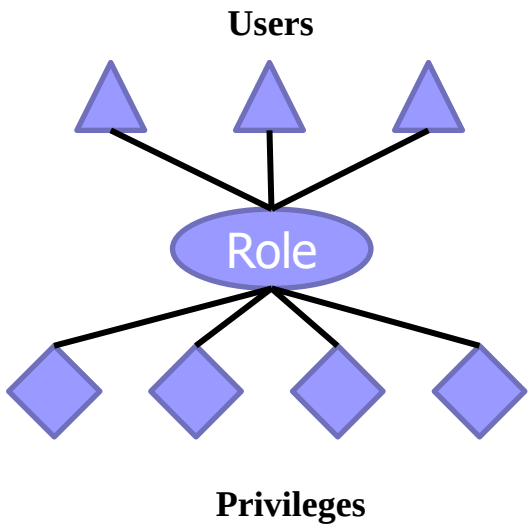
PRIVILEGES GROUPED BY ROLE

- 역할 사용을 통해 권한 관리를 단순화 시킴
- 시스템 권한과 객체 권한으로 구성 가능
- 역할은 사용자가 소유하지 않고, 스키마에도 존재하지 않음.

Grant Privileges Without Roles



Grant Privileges With Roles



CREATE A ROLE

Syntax

CREATE ROLE *role_name* [*NOT IDENTIFIED*] [*IDENTIFIED BY password*]

where	*role_name*	역할 이름
	NOT IDENTIFIED	패스워드 사용하지 않는다.(Default 값)
	IDENTIFIED BY	역할에 패스워드 설정
	password	패스워드

CREATE A ROLE

Example

역할 이름을 acct_rec으로 생성

```
SQL> CREATE ROLE acct_rec;
Role 'ACCT_REC' created.
```

패스워드가 bicentennial이고, 역할 이름을 acct_pay으로 생성

```
SQL> CREATE ROLE acct_pay
  2 IDENTIFIED BY bicentennial;
```

동의어와 테이블 생성 권한이 있는manager 역할 생성

```
SQL> GRANT CREATE TABLE, CREATE SYNONYM
  2 TO manager;
Granted.
```

신규 사용자 kevin에게 manager 권한 부여

```
SQL> GRANT manager TO kevin;

Granted.
```

DML : Data Manipulation Language

데이터 조작어(Data Manipulation Language)

데이터 조작어(이하 DML)는 데이터베이스에 저장된 데이터에 대한 질의, 삽입, 갱신, 삭제를 수행하기 위한 SQL 문장

명령어	설명
SELECT	데이터를 조회한다.
INSERT	데이터를 삽입한다.
UPDATE	데이터를 변경한다.
DELETE	데이터를 삭제한다.

DISPLAY TABLE STRUCTURE

칼럼의 이름과 데이터 타입을 포함한 테이블 구조 확인

Syntax

DESC[RIBE] *tablename*

Example

```
SQL> DESCRIBE s_emp;
```

```
COLUMN_NAME                                TYPE           CONSTRAINT
---------------------------------------------------------------- ---------------- ----------------------------------
EMPNO                                       NUMBER(4)       PRIMARY KEY
ENAME                                       VARCHAR(10)
JOB                                        VARCHAR(9)
MGR                                         NUMBER(4)
HIREDATE                                      DATE
SAL                                        NUMBER(7,2)
COMM                                        NUMBER(7,2)
DEPTNO                                      NUMBER(2)       REFERENTIAL


INDEX_NAME                          TYPE              COLUMN_NAME
------------------------------------------- ---------------------- -------------------------------------------
PK_EMP                             NORMAL             EMPNO
```

Display Data with the SELECT Statement

SQL SELECT 문을 사용하여 데이터베이스 테이블의 데이터를 조회할 수 있다.

Syntax

SELECT *column_informantion*
FROM *table(s)*
WHERE *condition*
ORDER BY *expression or keyword*

where	SELECT	검색할 열, 식 또는 상수를 지정
	FROM	데이터를 가지고 올 테이블 지정
	WHERE	특정 행을 검색할 기준(선택사항)
	ORDER BY	조회된 데이터를 정렬

DISPLAY ALL DATA IN A TABLE

테이블의 모든 칼럼을 출력하려면 **SELECT** 키워드에 '*'를 입력

Syntax

```
SELECT *
FROM table_name
```

Example

```
SQL> SELECT *
  2  FROM s_dept;
```

```
    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

4 rows selected.
```

DISPLAY ALL DATA IN A TABLE

조회하고 싶은 칼럼 이름과 해당 칼럼이 정의된 테이블을 입력하여 조회

Syntax

```
SELECT [DISTINCT] column_name [, column_name]
FROM table_name
```

Example

```
SQL> SELECT dname
  2     FROM s_dept;
```

```
DNAME
--------------
ACCOUNTING
RESEARCH
SALES
OPERATIONS

4 rows selected.
```

DISPLAY ALL DATA IN A TABLE

S_EMP 테이블의 칼럼과 구조를 조회하고, 사원 이름, 급여를 출력

Example

```
SQL> SELECT dname
  2      FROM s_dept;
```

```
SQL> DESC s_emp;

SQL> SELECT ename, sal
  2 FROM s_emp;

ENAME           SAL
---------- ----------
SMITH           800
ALLEN          1600
WARD           1250
JONES          2975
MARTIN         1250
BLAKE          2850
CLARK          2450
SCOTT          3000
KING           5000
TURNER         1500
ADAMS          1100
JAMES           950
FORD           3000
```

DISPLAY UNIQUE COLUMNS OF DATA

DISTINCT 절을 사용하여 고유한 데이터 행을 조회한다. **DISTINCT** 절은 **SELECT** 문 결과가 반환되기 전 중복된 행을 제거한다.

Syntax

SELECT [DISTINCT] *column_name* [, *column_name*]
FROM *table_name*

Example

```
SQL> SELECT job
  2 FROM s_emp;
JOB
---------
CLERK
SALESMAN
SALESMAN
MANAGER
SALESMAN
MANAGER
MANAGER
ANALYST
PRESIDENT
SALESMAN
CLERK
CLERK
ANALYST
STUDENT
```

```
SQL> SELECT DISTINCT job
  2 FROM s_emp;

JOB
---------
CLERK
ANALYST
STUDENT
PRESIDENT
SALESMAN
MANAGER
```

DISPLAY SPECIFIC COLUMN NAMES

별칭 - **SELECT** 문에서 칼럼 이름을 대체로 정의할 수 있다.
별칭에 공백, 특수문자가 포함되어 있거나, 대소문자 구분이 필요하면 별칭을 큰 따음표로 묶는다.

Example

```
SQL> SELECT DISTINCT job "Title"
  2 FROM s_emp;

Title
---------
CLERK
ANALYST
STUDENT
PRESIDENT
SALESMAN
MANAGER
```

```
SQL> SELECT ename EMPLOYEES
  2 FROM s_emp;

EMPLOYEES
----------
SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAMS
JAMES
FORD
```

DISPLAY SPECIFIC ROWS OF DATA

WHERE절을 사용해 조건에 맞는 행을 조회한다.

Syntax

```
SELECT { * |  column_name [, column_name…] }
FROM table_name
WHERE condition
```

Example

```
SQL> SELECT ename
  2     FROM s_emp
  3     WHERE sal > 2000;
```

```
ENAME
----------
KING
BLAKE
CLARK
JONES
FORD
SCOTT

6 rows selected.
```

DISPLAY SPECIFIC ROWS OF DATA

Comparison Operators Overview

=	Equal to
<>	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

DISPLAY SPECIFIC ROWS OF DATA

Comparison Operators Overview

BETWEEN…AND…	between two values
NOT BETWEEN… AND…	Not between two values
IN (list)	Equal to any member of the following list
NOT IN (list)	Not Equal to any member of the following list
LIKE	Match a character pattern using wildcard characters
IS NULL	Is a null
IN NOT NULL	Is not a null

DISPLAY SPECIFIC COLUMN NAMES

=

부서 번호가 **20**인 직원의 부서 번호, 사원 이름, 직무를 조회

Example

```
SQL> SELECT deptno, ename, job
  2     FROM s_emp
  3     WHERE deptno = 20;

DEPTNO ENAME      JOB
---------- ---------- ---------
        20 JONES      MANAGER
        20 FORD       ANALYST
        20 SMITH      CLERK
        20 SCOTT      ANALYST
        20 ADAMS      CLERK
```

DISPLAY SPECIFIC COLUMN NAMES

=

BLAKE의 사원 번호, 이름, 직무, 급여를 조회

Example

```
SQL> SELECT empno, ename, job, sal
  2 FROM s_emp
  3 WHERE ename = 'Blake';

0 row selected.

SQL> SELECT empno, ename, job, sal
  2 FROM s_emp
  3 WHERE ename = 'BLAKE';

    EMPNO ENAME      JOB              SAL
---------- ---------- --------- ----------
     7698 BLAKE      MANAGER         2850

1 row selected.
```

DISPLAY SPECIFIC COLUMN NAMES

=

입사 날짜가 **82/12/22**일인 사원의 이름, 부서 번호, 입사 날짜 조회

Example

```
SQL> SELECT ename, deptno, hiredate
  2  FROM s_emp
  3  WHERE hiredate = '82/12/22;

ENAME      DEPTNO HIREDATE
---------- ---------- ---------------------------------------------------------
SCOTT          20 0082/12/22
```

DISPLAY SPECIFIC COLUMN NAMES

< >

직무가 **MANAGER**가 아닌 사원의 이름, 직무, 급여 조회

Example

```
SQL> SELECT ename, job, sal
  2     FROM s_emp
  3     WHERE job <> 'MANAGER';
```

```
ENAME      JOB            SAL
---------- --------- ----------
KING       PRESIDENT      5000
MARTIN     SALESMAN       1250
ALLEN      SALESMAN       1600
TURNER     SALESMAN       1500
JAMES      CLERK           950
WARD       SALESMAN       1250
FORD       ANALYST        3000
SMITH      CLERK           800
SCOTT      ANALYST        3000
ADAMS      CLERK          1100
MILLER     CLERK          1300

11 rows selected.
```

DISPLAY SPECIFIC COLUMN NAMES

>

입사 날짜가 **82/11/01**보다 큰 사원의 이름, 입사 날짜 조회

Example

```
SQL> SELECT ename, hiredate
  2     FROM s_emp
  3     WHERE hiredate > '82/11/01';
ENAME      HIREDATE
---------- ---------------------------------------------------------------
SCOTT      82/12/22
ADAMS      83/01/15
```

급여가 **2,000**이상인 사원의 이름, 급여 조회

```
SQL> SELECT ename, sal
  2     FROM s_emp
  3     WHERE sal > 2000;
ENAME           SAL
---------- ----------
KING            5000
BLAKE           2850
CLARK           2450
JONES           2975
FORD            3000
SCOTT           3000
```

DISPLAY SPECIFIC COLUMN NAMES

>=

부서 번호가 **30**이상인 사원의 부서 번호, 이름 조회

Example

```
SQL> SELECT deptno, ename
  2 FROM s_emp
  3 WHERE deptno >= 30;

DEPTNO ENAME
---------- ----------
       30 BLAKE
       30 MARTIN
       30 ALLEN
       30 TURNER
       30 JAMES
       30 WARD
```

이름이 **SCOTT** 이상인 사원의 이름과 직무 조회

```
SQL> SELECT ename, job
  2 FROM s_emp
  3 WHERE ename >= 'SCOTT';

ENAME      JOB
---------- ---------
TURNER     SALESMAN
WARD       SALESMAN
SMITH      CLERK
SCOTT      ANALYST
```

DISPLAY SPECIFIC COLUMN NAMES

<

입사 날짜가 **81/05/01** 보다 작은 사원의 이름, 부서 번호, 급여, 입사 날짜 조회

Example

```
SQL> SELECT deptno, ename, sal, hiredate
  2 FROM s_emp
  3 WHERE hiredate < '81/05/01';
```

```
 DEPTNO ENAME          SAL HIREDATE
---------- ---------- ---------- ---------------------------------------------------------
        20 JONES            2975 81/04/01
        30 ALLEN             1600 81/02/11
        30 WARD              1250 81/02/23
        20 SMITH              800 80/12/09

4 rows selected.
```

DISPLAY SPECIFIC COLUMN NAMES

<=

부서 번호가 **10**이하인 사원의 이름, 부서 번호, 직무 조회

Example

```
SQL> SELECT deptno, ename, job
  2 FROM emp
  3 WHERE deptno <= 10;
```

```
 DEPTNO ENAME      JOB
---------- ---------- ---------
        10 KING       PRESIDENT
        10 CLARK      MANAGER
        10 MILLER     CLERK

3 rows selected.
```

DISPLAY SPECIFIC COLUMN NAMES

BETWEEN

입사 날짜가 **80/12/01**과 **81/03/01**사이에 있는 사원의 이름과 입사 날짜 조회

Example

```
SQL> SELECT ename, hiredate
  2 FROM s_emp
  3 WHERE hiredate BETWEEN '80/12/01' AND '81/03/01';
ENAME      HIREDATE
---------- ---------------------------------------------------------------
ALLEN      81/02/11
WARD       81/02/23
SMITH      80/12/09
```

급여가 **1,500**과 **2,000** 사이인 사원의 이름과 급여 조회

```
SQL> SELECT ename, sal
  2 FROM s_emp
  3 WHERE sal BETWEEN 1500 AND 2000;

ENAME           SAL
---------- ----------
ALLEN           1600
TURNER          1500

2 rows selected.
```

DISPLAY SPECIFIC COLUMN NAMES

NOT BETWEEN

부서 번호가 **10**과 **20** 사이에 있지 않은 부서 번호 조회

Example

```
SQL> SELECT *
  2 FROM s_dept
  3 WHERE deptno NOT BETWEEN 10 AND 20;

 DEPTNO DNAME          LOC
---------- -------------- -------------
       30 SALES          CHICAGO
       40 OPERATIONS     BOSTON
```

사원 이름이 **ALLEN**과 **SMITH** 사이에 있지 않은 사원 이름과 부서 번호 조회

```
SQL> SELECT ename, deptno
  2 FROM s_emp
  3 WHERE ename NOT BETWEEN 'ALLEN' AND 'SMITH';

ENAME         DEPTNO
---------- ----------
TURNER            30
WARD              30
ADAMS             20
```

DISPLAY SPECIFIC COLUMN NAMES

IN

직무가 **MANAGER, SALESMAN**인 사원 이름, 직무, 부서 번호 조회

Example

```
SQL> SELECT ename, job, deptno
  2 FROM s_emp
  3 WHERE job IN ('MANAGER', 'SALESMAN');
```

```
ENAME      JOB          DEPTNO
---------- --------- ----------
BLAKE      MANAGER          30
CLARK      MANAGER          10
JONES      MANAGER          20
MARTIN     SALESMAN         30
ALLEN      SALESMAN         30
TURNER     SALESMAN         30
WARD       SALESMAN         30

7 rows selected.
```

DISPLAY SPECIFIC COLUMN NAMES

NOT IN

부서 번호가 **10, 20**이 아닌 사원의 사원 번호, 이름, 부서 번호 조회

Example

```
SQL> SELECT empno, ename, deptno
  2 FROM s_emp
  3 WHERE deptno NOT IN (10,20);

 EMPNO ENAME          DEPTNO
---------- ---------- ----------
      7698 BLAKE            30
      7654 MARTIN           30
      7499 ALLEN            30
      7844 TURNER           30
      7900 JAMES            30
      7521 WARD             30

6 rows selected.
```

DISPLAY SPECIFIC COLUMN NAMES

LIKE

사원 이름이 '**M**'으로 시작하는 사원 이름 조회

Example

```
SQL> SELECT ename
  2 FROM s_emp
  3 WHERE ename LIKE 'M%';

ENAME
----------
MARTIN
MILLER
```

입사 날짜가 '**01**'로 끝나는 사원 이름, 입사 날짜 조회

```
SQL> SELECT ename, hiredate
  2 FROM s_emp
  3 WHERE hiredate LIKE '%01';

ENAME      HIREDATE
---------- ---------------------------------------------------------------
BLAKE      81/05/01
JONES      81/04/01
```

DISPLAY SPECIFIC COLUMN NAMES

LIKE

사원 이름에 '**A**'를 포함하는 사원 조회

Example

```
SQL> SELECT ename
  2 FROM s_emp
  3 WHERE ename LIKE '%A%';

ENAME
----------
BLAKE
CLARK
MARTIN
ALLEN
JAMES
WARD
ADAMS

7 rows selected.
```

DISPLAY SPECIFIC COLUMN NAMES

LIKE

사원 이름이 '**B**'로 시작해서 '**F**'끝나는 다섯 글자의 사원 이름 조회

Example

```
SQL> SELECT ename
  2 FROM s_emp
  3 WHERE ename LIKE 'B___E';
ENAME
----------
BLAKE
```

두 번째 문자가 '**L**'인 사원의 이름 조회

```
SQL> SELECT ename
  2 FROM s_emp
  3 WHERE ename LIKE '_L%';

ENAME
----------
BLAKE
CLARK
ALLEN

3 rows selected.
```

DISPLAY SPECIFIC COLUMN NAMES

IS NULL

커미션이 **NULL** 값인 사원의 이름, 직무, 급여 조회

Example

```
SQL> SELECT ename, job, sal
  2 FROM s_emp
  3 WHERE comm IS NULL;

ENAME      JOB            SAL
---------- --------- ----------
KING       PRESIDENT      5000
BLAKE      MANAGER        2850
CLARK      MANAGER        2450
JONES      MANAGER        2975
JAMES      CLERK           950
FORD       ANALYST        3000
SMITH      CLERK           800
SCOTT      ANALYST        3000
ADAMS      CLERK          1100
MILLER     CLERK          1300

10 rows selected.
```

DISPLAY SPECIFIC COLUMN NAMES

IS NOT

NULL

커미션이 **NULL** 값인 아닌 사원의 이름, 직무, 커미션 조회

Example

```
SQL> SELECT ename, job, comm
  2 FROM s_emp
  3 WHERE comm IS NOT NULL;
```

```
ENAME      JOB           COMM
---------- --------- ----------
MARTIN     SALESMAN        1400
ALLEN      SALESMAN         300
TURNER     SALESMAN           0
WARD       SALESMAN         500

4 rows selected.
```

DISPLAY ROWS WITH COMPLEX CONDITIONS

AND, OR을 이용해 **WHERE**절에서 여러 조건을 결합하여 테이블 조회

Sysntx

```
SELECT { * |  column_name [, column_name…] }
FROM table_name
WHERE condition {AND | OR} condition
```

DISPLAY ROWS WITH COMPLEX CONDITIONS

AND

부서 번호가 **20**이고, 급여가 **2,000** 보다 큰 사원의 이름, 급여, 부서 번호 조회

Example

```
SQL> SELECT ename, sal, deptno
  2 FROM s_emp
  3 WHERE deptno = 20
  4 AND sal > 2000;

ENAME           SAL     DEPTNO
---------- ---------- ----------
JONES          2975         20
FORD           3000         20
SCOTT           3000         20
```

부서 번호가 **10**이고, 직무가 **MANAGER**인 사원의 이름, 급여, 부서 번호, 직무 조회

```
SQL> SELECT ename, sal, deptno, job
  2 FROM s_emp
  3 WHERE deptno = 10
  4 AND job = 'MANAGER';

ENAME           SAL     DEPTNO JOB
---------- ---------- ---------- ---------
CLARK          2450         10 MANAGER
```

DISPLAY ROWS WITH COMPLEX CONDITIONS

OR

부서 번호가 **20**이거나, 직무가 **SALESMAN**인 사원의 이름, 급여, 부서 번호, 직무 조회

Example

```
SQL> SELECT ename, sal, deptno, job
  2 FROM s_emp
  3 WHERE deptno = 20
  4 OR job = 'SALESMAN';

ENAME           SAL     DEPTNO JOB
---------- ---------- ---------- ---------
JONES          2975         20 MANAGER
MARTIN         1250         30 SALESMAN
ALLEN          1600         30 SALESMAN
TURNER         1500         30 SALESMAN
WARD           1250         30 SALESMAN
FORD           3000         20 ANALYST
SMITH           800         20 CLERK
SCOTT          3000         20 ANALYST
ADAMS          1100         20 CLERK

9 rows selected.
```

DISPLAY ROWS WITH COMPLEX CONDITIONS

우선 순위는 모든 비교 연산자, **AND** 그리고 **OR** 순서다.

Rules of Precedence

ORDER Evaluated	Operator
1	All Comparison Operators (=, <>, >, >=, <, <=, IN, LIKE, IN NULL, BETWEEN…AND)
2	AND
3	OR

DISPLAY ROWS WITH COMPLEX CONDITIONS

급여가 **1,000** 이상이고, 직무 번호가 **10** 또는 **20**인 사원의 이름, 급여, 직무 번호 조회

Example

```
SQL> SELECT ename, sal, deptno
   FROM s_emp
   WHERE sal >= 1000 AND deptno = 10 OR deptno = 20;

ENAME           SAL     DEPTNO
---------- ---------- ----------
SMITH           800        20
JONES          2975        20
CLARK           2450        10
SCOTT          3000        20
KING           5000        10
ADAMS           1100        20
FORD           3000        20

7 rows selected.
```

ORDER THE ROWS DISPLAYED

특정 칼럼을 기준으로 정렬이 가능하다.

Sysntx

```
SELECT { * | column_name [, column_name…] }
FROM table_name
WHERE condition
ORDER BY   column_name  {ASC | DESC} [, column_name [ASC | DESC] …]
```

where	column_name	칼럼 이름
	table_name	테이블 이름
	ASC	행을 오름차순으로 정렬(Default)
	DESC	행을 내림차순으로 정렬

ORDER THE ROWS DISPLAYED

급여를 기준으로 오름차순 정렬

Example

```
SQL> SELECT ename, sal
  2 FROM s_emp
  3 WHERE deptno = 30
  4 ORDER BY sal;
```

```
ENAME           SAL
---------- ----------
JAMES           950
MARTIN          1250
WARD            1250
TURNER          1500
ALLEN           1600
BLAKE           2850

6 rows selected.
```

ORDER THE ROWS DISPLAYED

급여를 기준으로 내림차순 정렬

Example

```
SQL> SELECT ename, sal
  2 FROM s_emp
  3 WHERE deptno = 30
  4 ORDER BY sal DESC;
```

```
ENAME           SAL
---------- ----------
BLAKE          2850
ALLEN          1600
TURNER          1500
MARTIN          1250
WARD           1250
JAMES          950

6 rows selected.
```

ORDER THE ROWS DISPLAYED

사원 번호, 사원 이름, 부서 번호를 사원 번호 및 부서 번호로 오름차순 정렬

Example

```
SQL> SELECT empno, ename, deptno
  2 FROM s_emp
  3 ORDER BY empno, deptno;
```

```
 EMPNO ENAME         DEPTNO
---------- ---------- ----------
      7369 SMITH            20
      7499 ALLEN            30
      7521 WARD             30
      7566 JONES            20
      7654 MARTIN           30
      7698 BLAKE            30
      7782 CLARK            10
      7788 SCOTT            20
      7839 KING             10
      7844 TURNER           30
      7876 ADAMS            20
      7900 JAMES            30
      7902 FORD             20
      7934 MILLER           10

14 rows selected.
```

ORDER THE ROWS DISPLAYED

급여가 **1,500**이상인 사원의 이름, 부서 번호, 급여를 조회하고, 부서 번호로 오름차순, 급여로 내림차순하여 정렬

Example

```
SQL> SELECT ename, deptno, sal
  2 FROM s_emp
  3 WHERE sal >= 1500
  4 ORDER BY deptno, sal DESC;
```

```
ENAME        DEPTNO       SAL
---------- ---------- ----------
KING             10       5000
CLARK             10       2450
FORD             20       3000
SCOTT             20       3000
JONES             20       2975
BLAKE             30       2850
ALLEN             30       1600
TURNER            30       1500

8 rows selected.
```

INSERT DATA

INSERT 문을 이용해 테이블에 새로운 행을 추가할 수 있다.

Syntax

INSERT INTO *table* [*column*] VALUES (*value, value*…)

where	*table*	테이블 이름
	column	칼럼 이름
	value	열에 해당하는 값

INSERT DATA

INSERT 절에 칼럼을 나열하지 않고 테이블의 모든 칼럼에 대한 값을 삽입

Example

```
SQL> DESC s_dept;

COLUMN_NAME          TYPE            CONSTRAINT
-------------------------------------------------------------------------------
DEPTNO               NUMBER(2)       PRIMARY KEY
DNAME                VARCHAR(14)
LOC                  VARCHAR(13)

INDEX_NAME           TYPE            COLUMN_NAME
-------------------------------------------------------------------------------
PK_DEPT              NORMAL          DEPTNO
```

S_DEPT 테이블에 **'HR'** 부서 정보 입력

```
SQL> INSERT INTO s_dept
  2 VALUES (50, 'HR', 'SEOUL');

 DEPTNO DNAME         LOC
---------- -------------- -------------
       10 ACCOUNTING     NEW YORK
       20 RESEARCH       DALLAS
       30 SALES          CHICAGO
       40 OPERATIONS     BOSTON
       50 HR             SEOUL
```

INSERT DATA

INSERT 절에 칼럼을 나열하고 테이블의 모든 칼럼에 대한 값을 삽입

Example

```
SQL> DESCRIBE s_emp;

COLUMN_NAME                    TYPE           CONSTRAINT
-------------------------------------- ----------------- --------------------
EMPNO                          NUMBER(7)      PRIMARY KEY
                                              NOT NULL
ENAME                          VARCHAR(10)    NOT NULL
JOB                            VARCHAR(9)
MGR                            NUMBER(4)      REFERENTIAL
HIREDATE                       DATE
SAL                            NUMBER(7)
COMM                           NUMBER(7)
DEPTNO                         NUMBER(2)      REFERENTIAL

INDEX_NAME                     TYPE           COLUMN_NAME
------------------------------ ---------------------- --------------------
S_EMP_ID_PK                    NORMAL         EMPNO
```

INSERT DATA

S_EMP 테이블에 새로운 직원 정보 입력

Example

```
SQL> INSERT INTO s_emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
  VALUES (1234,'ALEX', 'DESIGNER', 7839, SYSDATE, 3000, NULL, 20);

1 row inserted.

SQL> SELECT *
  2 FROM s_emp
  3 WHERE ename = 'ALEX';

    EMPNO    ENAME      JOB           MGR    HIREDATE                                  SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ---------------------------------------------------------- ---------- ---------- ----------
     1234       ALEX      DESIGNER       7839    2022/10/07                                 3000                  20

1 row selected.
```

INSERT SPECIAL VALUES

새로운 사원 **CHRIS**에 대한 정보를 입력하되, 직무, 담당 매니저, 급여 정보를 제공하지 않는다. 빈 문자열로 **NULL**을 명시적으로 삽입

Example

```
SQL> INSERT INTO emp
  2 VALUES (7633, 'CHRIS', '', '', 2500, '', 50);

1 row inserted.
```

새로운 사원 **HENRY**에 대한 정보를 입력하되, 직무, 담당 매니저, 급여 정보를 제공하지 않는다. **NULL**을 암시적으로 삽입

```
SQL>  INSERT INTO s_emp(empno, ename, sal, deptno)
  2 VALUES (7634, 'HENRY', 1300, 50);

1 row inserted.
```

```
SQL> SELECT *
  2 FROM s_emp
  3 WHERE empno IN(7633, 7634);

    EMPNO ENAME      JOB           MGR HIREDATE                                 SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ------------------------------------------------------------ ---------- ---------- ----------
     7634 HENRY                                                                1300                          50
     7633 CHRIS                                                                2500                          50

2 rows selected.
```

INSERT SPECIAL VALUES

데이터베이스 사용자의 이름으로 **S_EMP** 테이블에 정보 입력하기

Example

```
SQL> INSERT INTO s_emp (empno, ename, hiredate, sal, deptno)
  2 VALUES (7636, USER, SYSDATE, 2000, 10);

1 row inserted.
```

```
SQL> SELECT *
  2 FROM s_emp
  3 WHERE empno = 7636;

    EMPNO ENAME      JOB           MGR HIREDATE                                                                  SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ---------------------------------------------------------------------- ---------- ---------- ----------
     7636 SYS                    2022/10/09                                                                    2000                  10

1 row selected.
```

UPDATE DATA

UPDATE 문으로 기존 행 수정

Syntax

UPDATE *table*
SET VALUES (*column* = *value*)
[WHERE *condition*]

where	*table*	업데이트 할 테이블 이름
	column	업데이트 할 칼럼 이름
	value	새로운 값
	condition	조건에 맞는 행을 업데이트

UPDATE DATA

사원번호 **7636**의 부서 번호와 **ALEX**의 부서 번호와 급여 변경

Example

```
SQL> UPDATE s_emp
  2 SET deptno = 20
  3 WHERE empno = 7636;

1 row updated.

SQL> UPDATE s_emp
  2 SET deptno = 20, sal = 4000
  3 WHERE ename = 'ALEX';

1 row updated.
```

```
SQL> SELECT empno, ename, sal, deptno
  2 FROM s_emp
  3 WHERE empno =7636 OR ename = 'ALEX';

    EMPNO ENAME           SAL     DEPTNO
---------- ---------- ---------- ----------
     1234 ALEX            4000         20
     7636 SYS             2000         20

2 rows selected.
```

UPDATE DATA

회사에서 부서 번호 **10**인 사원들에게 커미션을 **1,000**씩 지급한다.

Example

```
SQL> UPDATE s_emp
  2 SET comm = 1000
  3 WHERE deptno = 10;

3 rows updated.

SQL> SELECT ename, comm, deptno
  2 FROM s_emp
  3 WHERE deptno= 10;

ENAME          COMM     DEPTNO
---------- ---------- ----------
CHRIS          1000         10
CLARK           1000         10
KING           1000         10

3 rows selected.
```

DELETE DATA

DELETE 문으로 기존 행 삭제

Syntax

```
DELETE FROM table
[WHERE condition]
```

where	*table*	테이블 이름
	condition	조건에 맞는 행을 삭제

DELETE DATA

S_EMP 테이블에서 ALEX 사원 삭제

Example

```
SQL> DELETE FROM s_emp
  2 WHERE ename = 'ALEX';

1 row deleted.
```

```
SQL> SELECT ename
  2 FROM s_emp
  3 WHERE ename = 'ALEX';

0 row selected.
```

DELETE DATA

S_EMP 테이블에서 부서 번호 **50**인 사원들 삭제

Example

```
SQL> DELETE FROM s_emp
  2 WHERE deptno = 50;

2 row deleted.
```

```
SQL> SELECT ename
  2 FROM s_emp
  3 WHERE deptno = 50;

0 row selected.
```

조건이 없이 테이블 이름만 입력한 경우, 테이블 내 모든 데이터가 삭제된다. 전체 테이블을 삭제하는 경우가 아니면 **WHERE** 절을 생략하면 안된다.

CONTROL TRANSACTIONS

- 트랜잭션(TRANSACTION) - INSERT, UPDATE, DELETE
- 데이터 조작 작업은 데이터베이스 버퍼에 영향을 준다.
- 현재 사용자는 SELECT 문으로 데이터 조작 작업의 결과를 검토할 수 있다.
- 다른 사용자는 현재 사용자에 대한 데이터 조작 작업의 결과를 볼 수 없다.
- 영향을 받은 행은 LOCK이 걸리게 되고, 다른 사용자는 행을 변경할 수 없다.

Control Transaction Logic

Statement	Description
COMMIT	현재 트랜잭션을 종료하고 트랜잭션의 갱신된 내용을 데이터베이스에 반영
ROLLBACK	현재 트랜잭션을 종료하고 트랜잭션에서 갱신된 내용 모두를 취소

CONTROL TRANSACTIONS

State of the Data After COMMIT

- **COMMIT** 문을 사용하여 보류중인 모든 변경 내용**(INSERT, UPDATE, DELETE)**을 영구적으로 만든다.
- **COMMIT** 후 데이터 변경 사항이 데이터베이스 파일에 기록된다.
- 영향을 받은 행은 **LOCK**이 해제되고, 다른 사용자가 행을 변경할 수 있다.

S_DEPT 테이블에 새로운 부서 추가하고, **COMMIT**하기

Example

```
SQL> INSERT INTO dept (deptno, dname, loc)
  2 VALUES (60, 'LAW', 'LA');

SQL> SELECT *
  2 FROM dept
  3 WHERE dname = 'LAW';

    DEPTNO DNAME         LOC
---------- -------------- -------------
        60 LAW           LA

SQL> COMMIT;

Commit completed.
```

CONTROL TRANSACTIONS

State of the Data After ROLLBACK

- 데이터의 변경이 취소되고, 데이터의 이전 상태가 복원
- 영향을 받은 행은 **LOCK**이 해제되고, 다른 사용자가 행을 변경할 수 있다.

CONTROL TRANSACTIONS

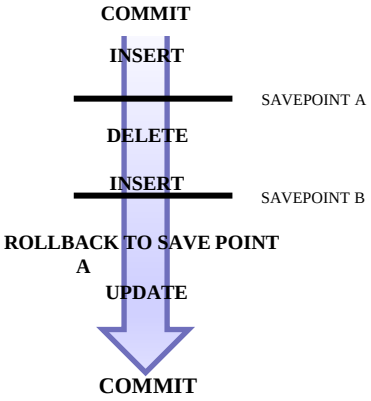
S_EMP 테이블에 부서 번호 20인 사원들의 행을 실수로 삭제했다. ROLLBACK을 이용해 복원한다.

Example

```
SQL> DELETE FROM s_emp
  2 WHERE deptno = 20;

5 rows deleted.

SQL> SELECT *
  2 FROM s_emp
  3 WHERE deptno = 20;

0 row selected.

SQL> ROLLBACK;

Rollback completed.

SQL> SELECT *
  2 FROM s_emp
  3 WHERE deptno = 20;

    EMPNO ENAME      JOB           MGR HIREDATE                                    SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ------------------------------------------------------------ ---------- ---------- ----------
     7369 SMITH      CLERK          7902 0080/12/09                                800                   20
     7566 JONES      MANAGER        7839 0081/04/01                               2975                   20
     7788 SCOTT      ANALYST        7566 0082/12/22                               3000                   20
     7876 ADAMS      CLERK          7788 0083/01/15                               1100                   20
     7902 FORD       ANALYST        7566 0081/12/11                               3000                   20
```

CONTROL TRANSACTIONS

- **SAVEPOINT** 문을 이용하여 현재 트랜잭션에 저장 지점을 만든다.
- 부분 롤백을 수행하기 위해선 저장 지점을 미리 만들어야 한다.
- 동일한 이름의 저장 지점을 설정하면 이전의 저장 지점은 삭제된다.



Alter Transaction Logic

Statement	Description
SAVEPOINT	현재 트랜잭션 내에서 저장 지점 표시
ROLLBACK TO SAVEPOINT	저장 지점이 표시된 후 보류 중인 변경 내용 삭제

CONTROL TRANSACTIONS

S_EMP 테이블에 새로운 사원의 정보를 추가하고 SAVEPOINT a, b 만들기

Example

```
SQL> INSERT INTO s_emp(empno, ename, hiredate, sal)
  2 VALUES (3790, 'GOODMAN', SYSDATE, 2000);

1 row inserted.

SQL> SAVEPOINT a;

Savepoint created.

SQL> INSERT INTO s_emp(empno, ename, hiredate, sal)
  2 VALUES (3791, 'BADMAN', SYSDATE, 1000);

1 row inserted.

SQL> SAVEPOINT b;

Savepoint created.

SQL> INSERT INTO s_emp(empno, ename, hiredate, sal)
  2 VALUES (3792, 'YESMAN', SYSDATE, 3000);
1 row inserted.
```

```
SQL> SELECT empno, ename
  2 FROM s_emp;

     EMPNO ENAME
---------- ----------
      3790 GOODMAN
      3791 BADMAN
      3792 YESMAN
      7369 SMITH
      7499 ALLEN
      7521 WARD
      7566 JONES
      7654 MARTIN
      7698 BLAKE
      7782 CLARK
      7788 SCOTT
      7839 KING
      7844 TURNER
      7876 ADAMS
      7900 JAMES
      7902 FORD
```

CONTROL TRANSACTIONS

마지막 두 사원을 실행 취소하되, 첫 번째 사원은 그대로 둔다. **SAVEPOINT**를 이용해 영구적으로 변경한다.

Example

```
SQL> ROLLBACK TO SAVEPOINT a;

Rollback completed.

SQL> COMMIT;

Commit completed.
```

```
SQL> SELECT empno, ename
  2 FROM s_emp;

     EMPNO ENAME
---------- ----------
      3790 GOODMAN
      7369 SMITH
      7499 ALLEN
      7521 WARD
      7566 JONES
      7654 MARTIN
      7698 BLAKE
      7782 CLARK
      7788 SCOTT
      7839 KING
      7844 TURNER
      7876 ADAMS
      7900 JAMES
      7902 FORD
```

CONTROL TRANSACTIONS

- **COMMIT 또는 ROLLBACK을 암묵적으로 수행하는 상황에 주의해야 한다.**

Implicit Transaction Processing

Circumstance	Result
CREATE TABLE과 같은 DDL 명령어 실행	자동 COMMIT
명시적 COMMIT 또는 ROLLBACK을 하지 않고 데이터베이스 정상 종료	자동 COMMIT
비정상적인 종료 또는 시스템 오류	자동 ROLLBACK

PERFORM COMPUTATIONS WITH DATA

PERFORM COMPUTATIONS WITH DATA

산술 및 SQL 함수를 사용하여 다양한 방법으로 데이터를 수정하고 조회한다.

- 숫자 및 날짜 값을 사용하여 계산 수행
- NULL 값을 포함하는 계산 처리
- 숫자, 날짜 및 문자 값을 수정
- 날짜 값을 다양한 방식으로 표시
- 행 그룹에 대한 계산 조회 및 수행

Arithmetic Operators	
+	더하기
-	빼기
*	곱하기
/	나누기

PERFORM COMPUTATIONS WITH NUMBERS

직무가 '**MANAGER**'인 사원에 대해 **$500**의 급여 인상을 계산 후 사원 이름, 급여, 인상된 급여**(NEW SALARY)**를 출력하라
.

Example

```
SQL> SELECT ename, sal, sal + 500 "NEW SALARY"
   FROM s_emp
   WHERE job = 'MANAGER'
   ORDER BY  sal + 500;
```

```
ENAME           SAL NEW SALARY
---------- ---------- ----------
CLARK          2450       2950
BLAKE          2850       3350
JONES          2975       3475

3 rows selected.
```

PERFORM COMPUTATIONS WITH NUMBERS

부서 번호가 **20**인 사원에 대해 급여의 **10%**를 보너스를 지급 후, 사원 이름, 급여, 보너스**(BONUS)**를 출력하라.

Example

```
SQL> SELECT ename, sal, sal * 0.1 BONUS
  2 FROM s_emp
  3 WHERE deptno = 20
  4 ORDER BY sal * 0.1;
```

```
ENAME           SAL      BONUS
---------- ---------- ----------
SMITH           800         80
ADAMS          1100        110
JONES          2975      297.5
SCOTT          3000        300
FORD           3000        300

5 rows selected.
```

PERFORM COMPUTATIONS WITH NUMBERS

Rules of Precedence

산술 연산자 우선 순위는 다음과 같다.

1. 곱셈과 나누기 **(*, /)**

2. 덧셈과 빼기 **(+, -)**

Example

부서 번호 **10**인 사원에 대해 연간 보상을 지급한다. 이름, 급여, 연간 보상**(ANNUAL COMPENSATION)**을 출력하라. 연간 보상은 급여에 **$100**를 상여 후 **12**를 곱한다.

```
SQL> SELECT ename, sal, (sal + 100) * 12 "ANNUAL COMPENSATION"
  2 FROM s_emp
  3 WHERE deptno = 10;
```

```
ENAME          SAL ANNUAL COMPENSATION
---------- ---------- ------------------
CLARK          2450              30600
KING           5000              61200

2 rows selected.
```

PERFORM COMPUTATIONS WITH NUMBERS

ROUND

ROUND(*num1*, *num2*) – *num1*을 소수점 아래 *num2* 위치에서 반올림한 값을 반환

Example

직무가 **CLERK**인 사원에게 급여를 근무 일수로 나눈 성과급을 지급한다. 이름, 급여, 성과급**(PERFORMANCE PAY)**을 출력하라. 근무 일수는 **22**일, 성과급은 소수점 둘째 자리에서 반올림 한다.

```
SQL> SELECT ename, sal, ROUND(sal/22, 2)
  2 FROM s_emp
  3 WHERE job = 'CLERK';
```

```
ENAME          SAL ROUND(SAL/22,0)
---------- ---------- ---------------
SMITH          800          36.36
ADAMS          1100           50
JAMES          950          43.18
```

PERFORM COMPUTATIONS WITH NUMBERS

TRUNC

TRUNC(*num1*, *num2*) – *num1*을 소수점 아래 *num2* 위치에서 버림한 값을 반환

Example

부서 번호가 **20**인 사원에게 급여를 근무 일수로 나눈 성과급을 지급한다. 이름, 급여, 성과급**(PERFORMANCE PAY)**을 출력하라. 근무 일수는 **30**일, 성과급은 소수점 둘째 자리에서 버림 한다.

```
SQL> SELECT ename, sal, TRUNC(sal /30, 3)
  2 FROM s_emp
  3 WHERE deptno = 20;
```

```
ENAME          SAL TRUNC(SAL/30,2)
---------- ---------- ---------------
SMITH           800          26.66
JONES          2975          99.16
SCOTT          3000           100
ADAMS          1100          36.66
FORD           3000           100
```

PERFORM COMPUTATIONS WITH NUMBERS

MOD

MOD(*num1, num2*) – *num1*을 *num2*로 나눈 나머지를 반환하는 함수

Example

사원 번호가 짝수인 사원의 정보를 모두 출력하라.

```
SQL> SELECT *
  2 FROM s_emp
  3 WHERE MOD(empno, 2) = 0;
```

```
 EMPNO ENAME      JOB            MGR HIREDATE                                                    SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ------------------------------------------------------------------------ ---------- ---------- ----------
      3790 GOODMAN                    2022/10/09                                                2000
      7566 JONES      MANAGER         7839 1981/04/01                                           2975                    20
      7654 MARTIN     SALESMAN        7698 1981/09/10                                           1250       1400         30
      7698 BLAKE      MANAGER         7839 1981/05/01                                           2850                    30
      7782 CLARK      MANAGER         7839 1981/05/09                                           2450                    10
      7788 SCOTT      ANALYST         7566 1982/12/22                                           3000                    20
      7844 TURNER     SALESMAN        7698 1981/08/21                                           1500          0         30
      7876 ADAMS      CLERK           7788 1983/01/15                                           1100                    20
      7900 JAMES      CLERK           7698 1981/12/11                                           950                     30
      7902 FORD       ANALYST         7566 1981/12/11                                           3000                    20

10 rows selected.
```

PERFORM COMPUTATIONS WITH NUMBERS

NVL

NVL(*expr1, expr2*) – *expr1*이 NULL이 아니면 *expr1*을 반환하고, *expr1*이 NULL이면 *expr2*를 반환하는 함수

Example

부서 번호가 30인 사원에게 급여와 커미션을 더한 금액을 지급한다. 사원 이름, 급여, 커미션, 추가 금액을 출력하라. 단, 커미션이 없으면 0으로 출력하라

```
SQL> SELECT ename, sal, comm, NVL(sal + comm, 0)
  2 FROM s_emp
  3 WHERE deptno = 30;
```

```
ENAME          SAL       COMM NVL(SAL+COMM,0)
---------- ---------- ---------- ---------------
ALLEN          1600        300          1900
WARD           1250        500          1750
MARTIN         1250       1400          2650
BLAKE          2850                        0
TURNER         1500          0          1500
JAMES           950                       0
```

PERFORM COMPUTATIONS WITH DATES

산술 연산자를 이용해 날짜를 계산할 수 있다.

Date Expression
date + number
date – number
date - date

Example

급여가 **2,000** 이상인 사원의 견습 기간 종료일을 확인한다. 이름, 입사 날짜, 견습 기간 종료일을 출력하라. 견습 기간은 입사 후 **90**일까지이다.

```
SQL> SELECT ename, hiredate, hiredate + 90
  2 FROM s_emp
  3 WHERE sal >= 2000;
```

```
ENAME      HIREDATE                                    HIREDATE+90
---------- ---------------------------------------------------------- ----------------------------------------------------------
JONES      1981/04/01                                  1981/06/30
BLAKE      1981/05/01                                  1981/07/30
CLARK      1981/05/09                                  1981/08/07
SCOTT      1982/12/22                                  1983/03/22
KING       1981/11/17                                  1982/02/15
FORD       1981/12/11                                  1982/03/11
```

PERFORM COMPUTATIONS WITH DATES

SYSDAT E

SYSDATE 함수를 이용해 오늘 날짜를 출력할 수 있다.

Example

부서 번호 20인 사원의 이름과 근무 기간을 일, 월로 출력한다. 단, 월은 30일로 계산한다

```
SQL> SELECT ename, SYSDATE - hiredate DAYS, (SYSDATE - hiredate) / 30 MONTHS
  2 FROM s_emp
  3 WHERE deptno = 20;
```

```
ENAME          DAYS     MONTHS
---------- ---------- ----------
SMITH      15280.3261 509.344204
JONES      15167.3261 505.577537
SCOTT      14537.3261 484.577537
ADAMS      14513.3261 483.777537
FORD       14913.3261 497.110871

5 rows selected.
```

PERFORM COMPUTATIONS WITH DATES

ADD_MONTHS

ADD_MONTHS(*date, integer*) – date에 integer만큼의 달을 더한 결과를 구하는 함수

Example

직무가 MANAGER인 사원의 이름과 입사 날짜에 6개월을 더한 값을 출력한다.

```
SQL> SELECT ename, ADD_MONTHS(hiredate, 6)
  2 FROM s_emp
  3 WHERE job = 'MANAGER';
```

```
ENAME      ADD_MONTHS(HIREDATE,6)
---------- ------------------------------------------------------------
JONES      1981/10/01
BLAKE      1981/11/01
CLARK      1981/11/09

3 rows selected.
```

PERFORM COMPUTATIONS WITH DATES

LAST_DAY

LAST_DAY(*date*) - 특정 일자에 해당하는 월의 마지막 일자를 표시하는 함수

Example

현재 달의 마지막 일자를 출력하라

```
SQL> SELECT LAST_DAY(SYSDATE)
  2 FROM dual;
```

```
LAST_DAY(SYSDATE)
---------------------------------------------------------------
2022/10/31

1 row selected.
```

NEXT_DAY

NEXT_DAY(*date, str*) – date와 가장 가까운 다음 요일 **str**의 날짜를 반환하는 함수

PERFORM COMPUTATIONS WITH DATES

MONTS_BETWEEN

MONTHS_BETWEEN(*date1*, *date2*) – date1과 date2의 개월 차를 구하는 함수. 두 날짜 사이의 일수를 31로 나눈 값을 반환

Example

직무가 **SALESMAN**인 사원의 이름, 입사일, 근무 개월을 출력하라

```
SQL> SELECT LAST_DAY(SYSDATE)
  2 FROM dual;
```

```
ENAME      HIREDATE          MONTHS_BETWEEN(SYSDATE,HIREDATE)
---------- ----------------------------------------------------------------------------------------

ALLEN      1981/02/11          499.979055

WARD       1981/02/23          499.591958

MARTIN     1981/09/10           493

TURNER     1981/08/21           493.656474


4 rows selected.
```

REFORMAT DATES

날짜 형식

Element	Description	Element	Description
DD	1개월 중 몇 번째 날인지 출력(1- 31)	YYYY	연도를 네 자리로 출력 (2022)
DY	축약 표기한 요일을 출력 (ex : THU)	HH:MI:SS	시간, 분, 초 (10:00:00)
DAY	요일을 출력 (ex : MONDAY)	Q	분기를 출력 (1-4)
MM	달을 출력 (1-12)	HH24	시간을 출력 (0 – 23)
MON	축약된 달 이름을 출력 (ex : DEC)	RM	달을 로마 숫자로 출력 (I-XII)
MONTH	달을 출력한다 (ex : NOVEMBER)	AM or PM	오전 또는 오후
YY	연도를 두 자리로 출력 (22)	TZH or THM	시간대에서 시간 or 분 출력

REFORMAT DATES

TO_CHAR

TO_CHAR(*date_value, format_mask*) – date_value를 format_mask 형식에 따라 문자열로 변환 하는 함수

Example

직무가 MANAGER인 사원의 이름, 입사일을 YY년 MM월 DD일 형식으로 출력하라

```
SQL> SELECT ename, TO_CHAR(hiredate, 'MM/YY')
   FROM s_emp
   WHERE job =  'MANAGER';
```

```
ENAME      HIREDATE            TO_CHAR(HIREDATE,'YY"년"MM"월"DD"일"')
---------- ------------------- ----------------------------------------
JONES      1981/04/01          81 년 04 월 01 ▒
BLAKE      1981/05/01          81 년 05 월 01 ▒
CLARK      1981/05/09          81 년 05 월 09 ▒

3 rows selected.
```

REFORMAT DATES

TO_DATE

TO_CHAR(*character_value, format_mask*) – character_value를 format_mask 형식에 따라 Date로 변환 하는 함수

Example

S_EMP 테이블에 신입사원을 추가한다. 입사일은 900708이다

```
SQL> INSERT INTO s_emp(empno, ename, hiredate, deptno)
  2 VALUES (8371, 'MARU', TO_DATE('19900708', 'YYYYMMDD'), 40);

1 row inserted.
```

```
 EMPNO ENAME      JOB           MGR HIREDATE                SAL    COMM     DEPTNO
---------- ---------- --------- ---------- -------------------- ---------- ---------- ----------
8371          MARU                                1990/07/08                                40

1 row selected.
```

MANIPULATE CHARACTER STRINGS

||

접합 연산자로 문자열을 결합 시킨다.

Example

부서 번호 **20**인 사원의 사원 번호와 이름을 합쳐 출력한다.

```
SQL> SELECT empno || ' ' || ename "ID AND EMPLOYEE"
  2 FROM s_emp
  3 WHERE deptno = 20;
```

```
ID AND EMPLOYEE
--------------------------------------------------------------------------------
7369 SMITH
7566 JONES
7788 SCOTT
7876 ADAMS
7902 FORD

5 rows selected.
```

MANIPULATE CHARACTER STRINGS

문자 함수

NAME : Delhi Sports

Function	EXAMPLE	RESULT
INITCAP	INITCAP (NAME)	Delhi Sports
UPPER	UPPER (NAME)	DELHI SPORTS
LOWER	LOWER (NAME)	delhi sports
SUBSTR	SUBSTR (NAME, 1, 4)	Delh
LENGTH	LENGTH (NAME)	12

MANIPULATE CHARACTER STRINGS

INITCAP

INITCAP – 첫 글자만 대문자로 변환한다.

Example

직무가 **CLERK**인 사원의 이름, 직무를 출력한다. 이름은 첫 글자는 대문자, 나머지는 소문자로 출력한다.

```
SQL> SELECT INITCAP(ename) NAME, job
  2 FROM s_emp
  3 WHERE job = 'CLERK';
```

```
NAME       JOB
---------- ---------
Smith      CLERK
Adams      CLERK
James      CLERK

3 rows selected.
```

MANIPULATE CHARACTER STRINGS

UPPER

UPPER - 모든 문자를 대문자로 변환한다.

Example

INITCAP 함수를 이용해 출력한 직원 이름을 UPPER 함수를 이용해 대문자로 변환한다.

```
SQL> SELECT UPPER(INITCAP(ename)) NAME, job
  2 FROM s_emp
  3 WHERE job = 'CLERK';
```

```
NAME       JOB
---------- ---------
SMITH      CLERK
ADAMS      CLERK
JAMES      CLERK

3 rows selected.
```

MANIPULATE CHARACTER STRINGS

LOWER

LOWER - 모든 문자를 소문자로 변환한다.

Example

부서 번호 **30**인 직원 이름, 직무를 출력한다. **LOWER** 함수를 이용해 모두 소문자로 변환한다.

```
SQL> SELECT LOWER(ename), LOWER(job)
  2 FROM s_emp
  3 WHERE deptno = 30;
```

```
LOWER(ENAME) LOWER(JOB)
------------ ----------
allen        salesman
ward         salesman
martin       salesman
blake        manager
turner       salesman
james        clerk

6 rows selected.
```

MANIPULATE CHARACTER STRINGS

SUBSTR

SUBSTR(*char, m* [*, n*] – char 내 m 번째 위치로부터 n 길이의 문자열을 추출하는 함수. n이 지정되지 않으면 마지막까지 추출

Example

부서 번호 10인 직원 이름을 뒤에서 첫 번째에서 2개까지 출력한다.

```
SQL>  SELECT ename, SUBSTR(ename, 1, 2)
  2  FROM s_emp
  3 WHERE deptno = 10;
```

```
ENAME      SUBSTR(ENAME,1,2)
---------- -----------------
CLARK      CL
KING       KI

2 rows selected.
```

MANIPULATE CHARACTER STRINGS

LENGTH

LENGTH – 문자열의 길이를 반환하는 함수

Example

부서 번호 **30**인 직원 이름과 이름의 길이를 출력하라.

```
SQL> SELECT ename, LENGTH(ename)
  2 FROM s_emp
  3 WHERE deptno = 30;
```

```
ENAME      LENGTH(ENAME)
---------- -------------
ALLEN                 5
WARD                  4
MARTIN                6
BLAKE                 5
TURNER                6
JAMES                 5
```

이름 길이가 **6**이상인 직원의 이름을 출력하라

```
SQL> SELECT ename
  2 FROM s_emp
  3 WHERE LENGTH(ename) >= 6;
```

```
ENAME
----------
MARTIN
TURNER
```

PERFORM SUMMARY COMPUTATIONS

숫자 함수

Function	Description
AVG	평균값
MAX	최댓값
MIN	최솟값
SUM	합
COUNT	로우의 개수를 세는 함수

MANIPULATE CHARACTER STRINGS

AVG SUM
MIN MAX

Example

직무가 SALESMAN 인 사원들의 급여 평균(AVERAGE), 최댓값(MAXIMUM), 최솟값(MINIMUM), 합(SUM)을 출력하라

```
SQL> COLUMN average FORMAT $99,999.99
SQL> COLUMN maximum FORMAT $99,999.99
SQL> COLUMN minimum FORMAT $99,999.99
SQL> COLUMN sum FORMAT $99,999.99
SQL> SELECT AVG(sal) average, MAX(sal) maximum, MIN(sal) minimum, SUM(sal) sum
  2 FROM s_emp
  3 WHERE job = 'SALESMAN';
```

```
 AVERAGE     MAXIMUM     MINIMUM         SUM
----------- ----------- ----------- -----------
  $1,400.00   $1,600.00   $1,250.00   $5,600.00

1 row selected.
```

MANIPULATE CHARACTER STRINGS

COUNT

Example

S_EMP 테이블에 있는 총 사원의 수를 구하라

```
SQL> SELECT COUNT(*)
  2 FROM s_emp;
```

```
 COUNT(*)
----------
        14

1 row selected.
```

커미션을 받는 사원의 수를 구하라

```
SQL> SELECT COUNT(comm) "Employees with Comm"
  2 FROM s_emp;
```

```
Employees with Comm
-------------------
                  4

1 row selected.
```

GROUP ROWS TOGETHER

부서 번호가 30인 사원은 6명으로 6번 표시가 된다. GROUP BY 절을 사용하면 각 부서에 대해 한 줄로 출력할 수 있다.

```
SQL> SELECT empno, ename, deptno
  2 FROM s_emp
  3 WHERE deptno = 30;

 EMPNO ENAME          DEPTNO
---------- ---------- ----------
      7499 ALLEN              30
      7521 WARD               30
      7654 MARTIN             30
      7698 BLAKE              30
      7844 TURNER             30
      7900 JAMES              30

6 rows selected.
```

```
SQL> SELECT deptno, COUNT(*) NUBMER
    FROM s_emp
    WHERE deptno = 30
    GROUP BY deptno;

 DEPTNO     NUBMER
---------- ----------
        30          6

1 row selected.
```

GROUP ROWS TOGETHER

GROUP BY 및 HAVING 절이 있는 행 그룹에 대해 요약 결과를 출력한다.

Syntax

```
SELECT column_name
FROM table_name
WHERE condition
GROUP BY group_by_expression
```

where	*group_by_expression*	그룹화가 되는 기준의 열을 지정

Example

S_EMP 테이블에 있는 직무를 출력한다.

```
SQL> SELECT job, COUNT(*) "Number"
  2 FROM s_emp
  3 GROUP BY job;
```

```
JOB         Number
--------- ----------
CLERK            3
ANALYST          2
PRESIDENT        1
SALESMAN         4
MANAGER          3
```

GROUP ROWS TOGETHER

Example

부서 번호에 따른 직원 수 출력

```
SQL> SELECT deptno, COUNT(*) "Head Count"
  2 FROM s_emp
  3 GROUP BY deptno;
```

```
 DEPTNO Head Count
---------- ----------
        10          2
        40          1
        20          5
        30          6
```

GROUP BY 절 없이 정규 열과 그룹 함수를 함께 사용하면 안된다.

```
SQL> SELECT deptno, COUNT(*) "Employees Within Titles"
  2  FROM s_emp;
TBR-8038: Expression is not in a GROUP BY clause.
```

GROUP ROWS TOGETHER

둘 이상의 **GROUP BY** 칼럼을 나열하여 그룹 및 하위 그룹에 대한 결과를 출력한다.

Example

```
SQL> SELECT deptno, job, COUNT(*) "Employees Within Titles"
  2 FROM s_emp
  3 WHERE deptno = 30
  4 GROUP BY deptno, job;
```

```
 DEPTNO JOB       Employees Within Titles
---------- --------- ----------------------
       30 CLERK                     1
       30 SALESMAN                  4
       30 MANAGER                   1
```

```
SQL> SELECT  job, deptno, COUNT(*) "Employees Within Titles"
   2 FROM s_emp
   3 WHERE deptno = 30
   4 GROUP BY job, deptno ;
```

```
JOB          DEPTNO Employees Within Titles
--------- ---------- ----------------------
CLERK            30                  1
SALESMAN         30                  4
MANAGER          30                  1
```

DISPLAY SPECIFIC GROUPS

특정 행 또는 특정 그룹 출력

Syntax

```
SELECT column_name [,column_name]
FROM table_name
WHERE condition
GROUP BY group_by_expression
HAVING condition
```

where	*condition*	지정된 조건이 참(TRUE)인 그룹만 반환

DISPLAY SPECIFIC GROUPS

직원이 세 명 이상인 부서의 급여 평균과 직원 수를 출력하라

Example

```
SQL> SELECT deptno, AVG(sal) average, COUNT(*) "Number of Employees"
  2 FROM s_emp
  3 GROUP BY deptno
  4 HAVING COUNT(*) >= 3;
```

```
 DEPTNO     AVERAGE Number of Employees
---------- ----------- -------------------
       20   $2,175.00                   5
       30   $1,566.67                   6

2 rows selected.
```

DISPLAY SPECIFIC GROUPS

각 직무 별 급여의 합이 **5,000**보다 큰 직무와 급여 합을 출력하라. 급여의 합은 내림차순으로 정렬

Example

```
SQL> SELECT job, SUM(sal) sum
  2 FROM s_emp
  3 GROUP BY job
  4 HAVING SUM(sal) > 5000
  5 ORDER BY SUM(sal) DESC;
```

```
JOB             SUM
--------- -----------
MANAGER     $8,275.00
ANALYST     $6,000.00
SALESMAN    $5,600.00

3 rows selected.
```

DISPLAY DATA FROM MULTIPLE TABLES

DISPLAY DATA FROM MULTIPLE TABLES

- 조인을 작성하여 단일 **SELECT** 문 안에 있는 관련 테이블의 데이터를 표시한다.
- 두 개 이상의 테이블에 유사한 정보가 들어 있는 경우 동일한 테이블에 모두 저장된 것처럼 정보를 제공한다.

Desired Display
직원 이름과 해당 부서 ①
부서명과 해당 지역 ②

ID	LAST_NAME	FIRST_NAME	USERID	START_DATE	COMMENTS	MANAGER_ID	TITLE	① DEPT_ID	SALARY	COMMISSION_PCT
1	Velasquez	Carmen	cvelasqu	1990-03-03			President	50	2500	
2	Ngao	LaDoris	lngao	1990-03-08		1	VP, Operations	41	1450	
3	Nagayama	Midori	mnagayam	1991-06-17		1	VP, Sales	31	1400	
4	Quick-To-See	Mark	mquickto	1990-04-07		1	VP, Finance	10	1450	
5	Ropeburn	Audry	aropebur	1990-03-04		1	VP, Administration	50	1550	
6	Urguhart	Molly	murguhar	1991-01-18		2	Warehouse Manager	41	1200	
7	Menchu	Roberta	rmenchu	1990-05-14		2	Warehouse Manager	42	1250	
8	Biri	Ben	bbiri	1990-04-07		2	Warehouse Manager	43	1100	
9	Catchpole	Antoinette	acatchpo	1992-02-09		2	Warehouse Manager	44	1300	
10	Havel	Marta	mhavel	1991-02-27		2	Warehouse Manager	45	1307	
11	Magee	Colin	cmagee	1990-05-14		3	Sales Representative	31	1400	10
12	Giljum	Henry	hgiljum	1992-01-18		3	Sales Representative	32	1490	12,5
13	Sedeghi	Yasmin	ysedeghi	1991-02-18		3	Sales Representative	33	1515	10
14	Nguyen	Mai	mnguyen	1992-01-22		3	Sales Representative	34	1525	15
15	Dumas	Andre	adumas	1991-10-09		3	Sales Representative	35	1450	17,5
16	Maduro	Elena	emaduro	1992-02-07		6	Stock Clerk	41	1400	
17	Smith	George	gsmith	1990-03-08		6	Stock Clerk	41	940	
18	Nozaki	Akira	anozaki	1991-02-09		7	Stock Clerk	42	1200	
19	Patel	Vikram	vpatel	1991-08-06		7	Stock Clerk	42	795	
20	Newman	Chad	cnewman	1991-07-21		8	Stock Clerk	43	750	
21	Markarian	Alexander	amarkari	1991-05-26		8	Stock Clerk	43	850	
22	Chang	Eddie	echang	1990-11-30		9	Stock Clerk	44	800	
23	Patel	Radha	rpatel	1990-10-17		9	Stock Clerk	34	795	
24	Dancs	Bela	bdancs	1991-03-17		10	Stock Clerk	45	860	
25	Schwartz	Sylvie	sschwart	1991-05-09		10	Stock Clerk	45	1100	

② REGION

ID	NAME
1	North America
2	South America
3	Africa / Middle East
4	Asia
5	Europe

EMP ① — DEPT ②

ID	NAME	REGION_ID
10	Finance	1
31	Sales	1
32	Sales	2
33	Sales	3
34	Sales	4
35	Sales	5
41	Operations	1
42	Operations	2
43	Operations	3
44	Operations	4
45	Operations	5
50	Administration	1

DISPLAY DATA FROM MULTIPLE TABLES

- **View**를 생성해서 여러 테이블을 단일 테이블인 것처럼 참조할 수 있다.
- **WHERE** 절에 간단한 조인 조건을 작성하여 둘 이상의 관련 테이블의 행을 조회한다.

Syntax

```
SELECT table.column, table.column…
FROM table1, table2…
WHERE table1.column = table2.column
```

where	*table.column*	데이터를 검색하는 테이블 및 칼럼
	table1.column = table2.column	테이블을 결합 하는 조건, 칼럼 이름 앞에 테이블 명을 표시

DISPLAY DATA FROM RELATED TABLES

모든 직원의 성, 부서 **ID** 그리고 부서 이름을 출력하라

Example

```
SQL> SELECT e_emp.last_name, e_emp.dept_id, e_dept.name
  2 FROM e_emp, e_dept
  3 WHERE e_emp.dept_id = e_dept.id;
```

```
LAST_NAME                 DEPT_ID NAME
------------------------- ---------- -------------------------
Velasquez                         50 Administration
Ngao                              41 Operations
Nagayama                          31 Sales
Quick-To-See                      10 Finance
Ropeburn                          50 Administration
Urguhart                          41 Operations
…
25 rows selected.
```

DISPLAY DATA FROM RELATED TABLES

모든 부서의 부서 **ID,** 지역 **ID** 및 지역 이름을 출력하라

Example

```
SQL> SELECT e_dept.id "Department ID",
  2 e_region.id "Region ID",
  3 e_region.name "Region Name"
  4 FROM e_dept, e_region
  5 WHERE e_dept.region_id = e_region.id;
```

```
Department ID  Region ID Region Name
-------------- ---------- -------------------------------------------------
           50          1 North America
           10          1 North America
           41          1 North America
           42          2 South America
…
12 rows selected.
```

DISPLAY DATA FROM RELATED TABLES

북미(North America) 지역의 모든 부서의 지역 **ID,** 지역 이름**,** 부서 **ID** 및 부서 이름을 출력하라

Example

```
SQL> SELECT e_region.id "Region ID",
   e_region.name "Region Name",
   e_dept.id "Dept ID",
   e_dept.name "Dept Name"
   FROM e_dept, e_region
   WHERE e_region.id = e_dept.region_id
   AND e_region.name = 'North America';
```

```
 Region ID Region Name          Dept ID Dept Name
---------- -------------------- ---------- ------------------------
         1 North America                50 Administration
         1 North America                10 Finance
         1 North America                41 Operations
         1 North America                31 Sales
```

DISPLAY DATA WITHOUT A DIRECT MATCH

- 외부 조인이 있는 다른 테이블의 행과 직접 일치하지 않는 행을 한 테이블에서 출력할 수 있다.

Syntax

```
SELECT table.column, table.column…
FROM table1, table2…
WHERE table1.column = table2.column (+)
```

```
SELECT table.column, table.column…
FROM table1, table2…
WHERE table1.column(+) = table2.column
```

where	*table1.column = table2.column*	테이블을 함께 결합하는 조건.
	(+)	외부 결합 기호. WHERE 절 조건 아무쪽에 배치할 수 있지만, 동시에는 안된다.

DISPLAY DATA WITHOUT A DIRECT MATCH

모든 고객의 영업 담당자 이름과 **ID** 및 고객 이름을 출력한다. 고객에게 영업 담당자가 할당되지 않는 경우에도 고객 이름을 포함한다.

Example

```
SQL> COLUMN last_name HEADING 'Sales Rep Name' FORMAT A15
SQL> COLUMN id HEADING 'Sales Rep ID'
SQL> COLUMN name HEADING 'Customer Name' FORMAT A30
SQL> SELECT e_emp.last_name, e_emp.id, e_customer.name
  2 FROM e_emp, e_customer
  3 WHERE e_emp.id(+) = e_customer.sales_rep_id
  4 ORDER BY e_emp.id;
```

DISPLAY DATA WITHOUT A DIRECT MATCH

```
Sales Rep Name  Sales Rep ID Customer Name
--------------- ------------ ----------------------------
Magee                    11 Womansport
Magee                    11 Beisbol Si!
Magee                    11 Big John's Sports Emporium
Magee                    11 Ojibway Retail
Giljum                   12 Unisports
Giljum                   12 Futbol Sonora
Sedeghi                  13 Hamada Sport
Nguyen                   14 OJ Atheletics
Nguyen                   14 Delhi Sports
Dumas                    15 Kam's Sporting Goods
Dumas                    15 Sportique
Dumas                    15 Muench Sports
Dumas                    15 Kuhn's Sports
Dumas                    15 Sporta Russia
                        Sweet Rock Sports

15 rows selected.
```

DISPLAY DATA WITHOUT A DIRECT MATCH

모든 고객과 해당 주문의 고객 **ID,** 고객 이름 및 주문 **ID**를 출력하라. 주문하지 않은 고객 **ID**와 이름도 출력하라.

Example

```
SQL> SELECT e_customer.id "Customer ID",
  2 e_customer.name "Customer Name",
  3 e_ord.id "Order ID"
  4 FROM e_customer, e_ord
  5 WHERE e_customer.id = e_ord.customer_id(+);
```

DISPLAY DATA WITHOUT A DIRECT MATCH

```
Customer ID Customer Name                                   Order ID
----------- ---------------------------------------------- ----------
        201 Unisports                                      97
        202 OJ Atheletics                                  98
        203 Delhi Sports                                   99
        204 Womansport                                     100
        205 Kam's Sporting Goods                                101
        206 Sportique                                      102
        208 Muench Sports                                  103
        208 Muench Sports                                  104
        209 Beisbol Si!                                    105
        210 Futbol Sonora                                  106
        211 Kuhn's Sports                                  107
        212 Hamada Sport                                   108
        213 Big John's Sports Emporium                          109
        214 Ojibway Retail                                 110
        204 Womansport                                     111
        210 Futbol Sonora                                  112
        207 Sweet Rock Sports
        215 Sporta Russia

18 rows selected.
```

CREATE A VIEW OF MULTIPLE TABLES

CREATE VIEW 문에 여러 테이블 쿼리를 포함시켜 여러 테이블의 **VIEW**를 생성한다.

모든 직원의 성, 부서 **ID** 및 부서 이름을 포함하는 **VIEW**를 생성하라

Example

```
SQL> CREATE VIEW empdeptvu AS
  2 SELECT e_emp.last_name,
  3        e_emp.dept_id,
  4 e_dept.name
  5 FROM e_emp, e_dept
  6 WHERE e_emp.dept_id = e_dept.id;

View 'EMPDEPTVU' created.
```

CREATE A VIEW OF MULTIPLE TABLES

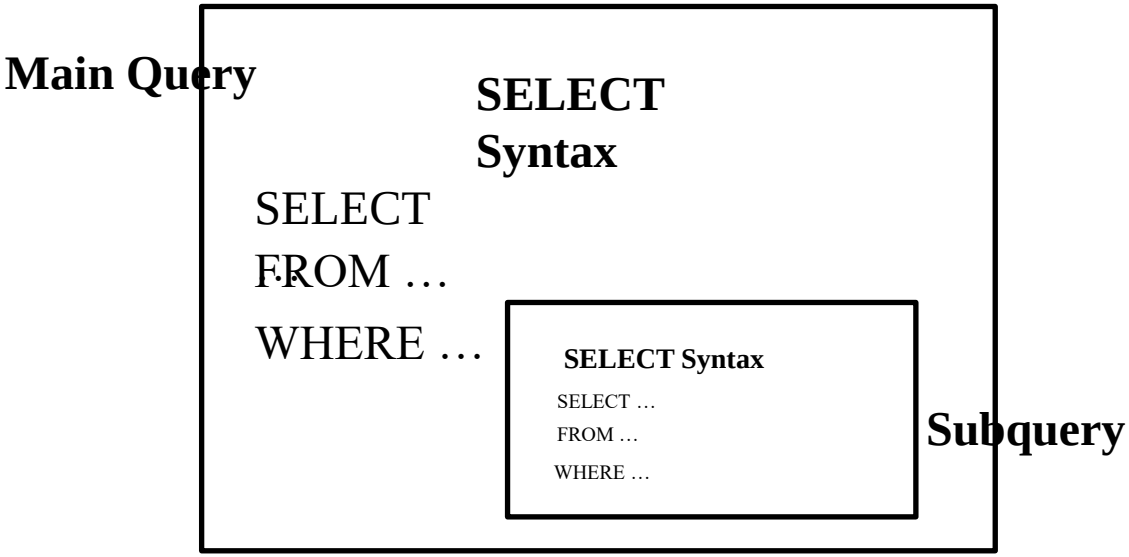
Example

```
Sales Rep Name     DEPT_ID Customer Name
-------------- ---------- ------------------------------
Velasquez              50 Administration
Ngao                   41 Operations
Nagayama               31 Sales
Quick-To-See           10 Finance
Ropeburn               50 Administration
Urguhart               41 Operations
Menchu                 42 Operations
Biri                   43 Operations
Catchpole              44 Operations
Havel                  45 Operations
…
25 rows selected.
```

PASS VALUES BETWEEN QUERIES

PASS VALUES BETWEEN QUERIES

- 하위 쿼리를 작성하여 다른 쿼리의 결과에 따라 달라지는 쿼리와 함께 데이터를 출력한다.



NEST QUERIES THE RETURN ONE ROW

- **SELECT** 문을 중첩하여 하위 쿼리에서 반환된 단일 행에 따라 기본 쿼리로 표시된 행을 제한한다.
- 예를 들어 **Smith**와 동일한 직무를 가진 직원을 **Smith**가 어떤 직무를 가지고 있는지 모른채로 출력한다.

Syntax

```
SELECT column_name, [, column_name]
FROM table1_name
WHERE expression operator
        (SELECT column_name, [, column_name]
         FROM table1_name)
```

where	*operator*	>, =, <, IN 과 같은 비교 연산자

NEST QUERIES THE RETURN ONE ROW

Smith와 동일한 직무를 가진 직원의 성을 출력하라.

Example

```
SQL>  SELECT last_name, title
   FROM e_emp
   WHERE title =
        (SELECT title
         FROM e_emp
         WHERE last_name = 'Smith');
```

```
Sales Rep Name  TITLE
--------------- --------------------------
Maduro          Stock Clerk
Smith           Stock Clerk
Nozaki          Stock Clerk
Patel           Stock Clerk
Newman          Stock Clerk
Makarian        Stock Clerk
Chang           Stock Clerk
Patel           Stock Clerk
Dancs           Stock Clerk
Schwartz        Stock Clerk
```

NEST QUERIES THE RETURN ONE ROW

Biri와 같은 부서에 있는 모든 직원의 성, 직무, 부서 ID를 출력하라.

Example

```
SQL> SELECT last_name, title, dept_id
  2 FROM e_emp
  3 WHERE dept_id = (SELECT dept_id
  4                  FROM e_emp
  5                  WHERE UPPER(last_name) = 'BIRI');
```

```
Sales Rep Name  TITLE                    DEPT_ID
--------------- ------------------------ ----------
Biri            Warehouse Manager            43
Newman          Stock Clerk                  43
Makarian        Stock Clerk                  43

3 rows selected.
```

NEST QUERIES THE RETURN MULTIPLE ROW

지역 2에 있는 부서에서 일하는 직원의 직원 **ID**, 성, 급여, 부서 **ID**를 출력하라

Example

```
SQL> SELECT id, last_name salary, dept_id
  2 FROM e_emp
  3 WHERE dept_id IN
  4           (SELECT id
  5           FROM e_dept
  6           WHERE region_id = 2);
```

```
Sales Rep ID SALARY                   DEPT_ID
------------ ------------------------ ----------
           7 Menchu                         42
          12 Giljum                         32
          18 Nozaki                         42
          19 Patel                        42

4 rows selected.
```

NEST QUERIES THE RETURN MULTIPLE ROW

다중 행 하위 쿼리는 **IN**과 같은 다중 행 연산자를 사용해야 한다.

Patel이라는 직원과 함께 부서에서 일하는 모든 직원의 직원 **ID**, 성, 부서 **ID**를 출력하라

Example

```
SQL> SELECT id, last_name, dept_id
  2 FROM e_emp
  3 WHERE dept_id = (SELECT dept_id
  4                  FROM e_emp
  5                   WHERE UPPER(last_name) = 'PATEL');
```

```
TBR-11002: Subquery returned multiple rows where a scalar value was expected.
```

NEST MULTIPLE QUERIES

영업부(**Sales**) 또는 지역 **2**에 있는 모든 직원의 직원 **ID,** 성**,** 급여 및 부서 **ID** 를 출력하라

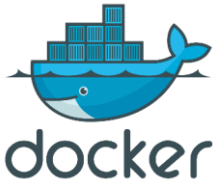
Example

```
SQL> SELECT id, last_name, dept_id
  2 FROM e_emp
  3 WHERE dept_id IN (SELECT id
  4                   FROM e_dept
  5                   WHERE name = 'Sales')
  6 OR dept_id IN (SELECT id
  7                FROM e_dept
  8                WHERE region_id = 2);
```

```
Sales Rep ID Sales Rep Name     DEPT_ID
------------ --------------- ----------
           3 Nagayama                31
           7 Menchu                  42
          11 Magee                   31
          12 Giljum                  32
          13 Sedeghi                 33
          14 Nguyen                  34
          15 Dumas                   35
          18 Nozaki                  42
          19 Patel                   42
          23 Patel                   34

10 rows selected.
```

빅데이터처리 플랫폼

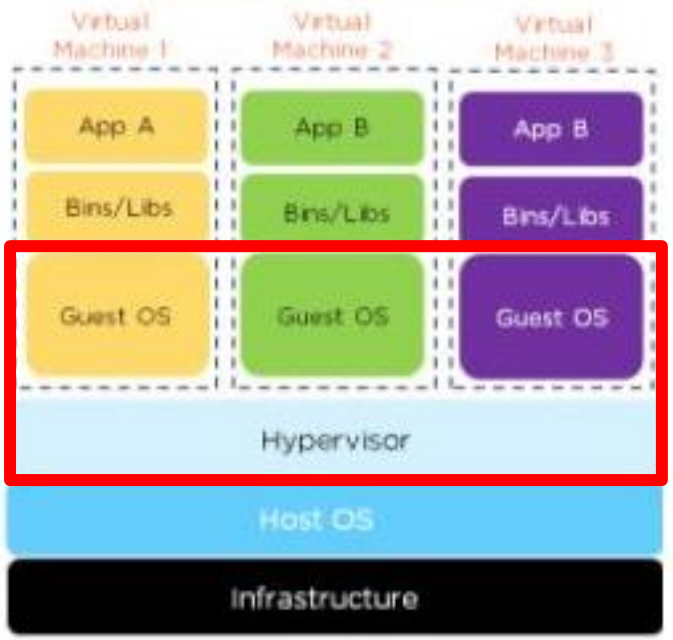
- What is Containers?
  - 리눅스 컨테이너의 여러 기능과 합쳐져 솔루션을 제공
  - 컨테이너는 호스트 OS의 자원 공유하여 효율적으로 사용할 수 있게 해준다.
  - 애플리케이션을 컨테이너로 좀 더 쉽게 사용할 수 있게 만든 오픈소스
  - 격리된 가상공간을 만들긴 하지만 실제 수행을 호스트 운영 체제에서 수행
  - 가상머신과 달리 성능 손실이 거의 없는 차세대 클라우드 솔루션으로 주목



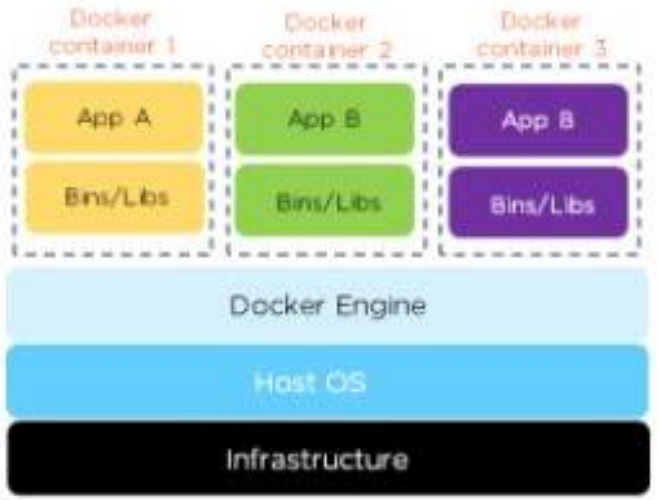
- Docker Project
  - Docker Compose, Docker Machine, Registry 등 다양한 프로젝트 존재
  - 일반적 도커 : Docker Engine을 의미
  - Docker Project는 Docker Engine을 효율적으로 사용하기 위한 도구.

- VM(Virtual Machine) VS Docker
  - VM
    - 구동중인 운영체제와 각종 라이브러리까지 전체 영역을 독립적으로 구분
    - 하이퍼바이저(Hypervisor)를 통한 가상화로 성능 손실이 발생

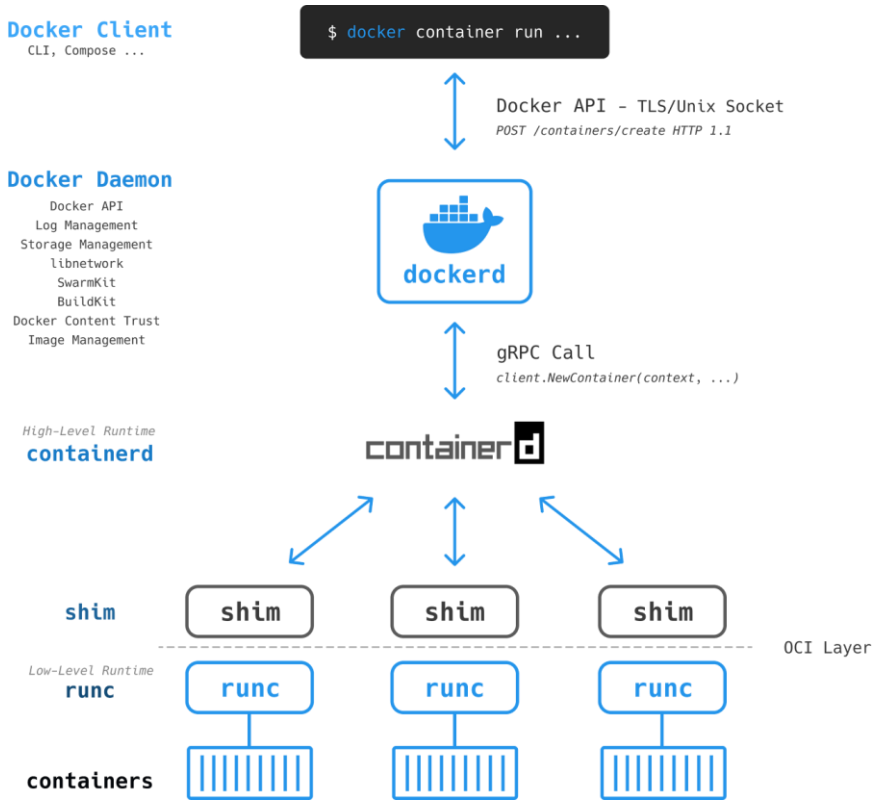
      * 하이퍼바이저 : 가상 머신 모니터(Virtual Machine Monitor, VMM)라고 불리며, 운영체제와 가상 머신의 리소스를 분리해 VM의 생성 및 관리 지원
    - 실제 앱이 구동되기까지 시간이 오래 걸림.



- VM(Virtual Machine) VS Docker
  - Docker
    - 가상머신과 비교하여 Guest OS를 포함하지 않기 때문에 훨씬 가볍고 실행 속도가 빠름.
    - 클라우드 환경에서 쉽게 배포 가능
    - 새로운 기능을 테스트할 때 사용하면 테스트 효율성 증가
    - Docker Hub를 통해 다양한 오픈소스 이미지 사용가능(MySQL, Python, Nginx 등)
    - 필요에 따라 기존 이미지를 기반으로 커스텀 이미지를 생성할 수 있음



- 도커 구조
  - Docker Client
    - docker run과 같은 명령어를 API로 변환해 dockerd에 요청
  - Docker Daemon(dockerd)
    - 도커 이미지 관리, 이미지 빌드 등 다양한 작업 수행
    - containerd 호출
  - containerd
    - docker에서 분리되어 오픈소스로 운영되고 있는 컨테이너 런타임
    - Container Runtime Interface(CRI)
  - runc
    - 실제 컨테이너를 생성하는 곳
    - Low-level Conatiner Runtime



```
# yum-utiles 패키지 설치
sudo yum -y update
sudo yum install -y yum-utils

# repository 설정
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

# 최신 버전의 Docker Engine 설치
# sudo yum install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

# 특정 버전의 Docker Engine 설치
version='23.0.1'
sudo yum install -y docker-ce-${version} docker-ce-cli-${version} containerd.io docker-buildx-plugin docker-compose-plugin

sudo systemctl start docker
sudo systemctl enable docker
```

#설치 확인

sudo systemctl status docker.service

```
ubuntu@ip-172-31-45-237:~$ sudo systemctl status docker.service
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2024-03-10 11:40:05 UTC; 4min 25s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 10040 (dockerd)
      Tasks: 10
     Memory: 110.3M
        CPU: 593ms
     CGroup: /system.slice/docker.service
             └─10040 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

sudo docker run hello-world

```
ubuntu@ip-172-31-45-237:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:d000bc569937abbe195e20322a0bde6b2922d805332fd6d8a68b19f524b7d21d
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

sudo systemctl start servicename

- 서비스 시작

sudo systemctl restart servicename

- 서비스 재시작

sudo systemctl stop servicename

- 서비스 중지

sudo systemctl status servicename

- 서비스 상태 확인

- 도커 그룹에 유저 추가하기

  sudo groupadd docker #(docker group 생성)

  sudo usermod -aG docker $USER #(docker group 해당 유저 추가)

  newgrp docker #(적용하기)

- 도커 버전 확인
  - docker -v

```
For more help on how to use Docker, head to https://docs.docker.
ubuntu@ip-172-31-45-237:~$ docker -v
Docker version 25.0.4, build 1a576c5
```

- 이미지 실행하기

docker run debian echo "hello World"

```
ubuntu@ip-172-31-45-237:~$ docker run debian echo "hello World"
Unable to find image 'debian:latest' locally
latest: Pulling from library/debian
7bb465c29149: Pull complete
Digest: sha256:4482958b4461ff7d9fabc24b3a9ab1e9a2c85ece07b2db1840c7cbc01d053e90
Status: Downloaded newer image for debian:latest
hello World
```

- 같은 명령어를 실행하게 되면 이미지를 다운받지 않고 바로 실행.

```
ubuntu@ip-172-31-45-237:~$ docker run debian echo "hello World"
hello World
```

1. debian은 사용하고자 하는 이미지 이름
2. 이미지가 없는 것을 확인하고 온라인으로 도커 허브에서 최신 버전의 이미지 다운
3. 이미지를 실행 상태의 컨테이너로 전환
4. echo "hello World" 라고 작성한 명령어 실행
5. 컨테이너 종료

- 컨테이너 내부에서 쉘(shell) 사용하기

  docker run -i -t debian /bin/bash

```
ubuntu@ip-172-31-45-237:~$ docker run -i -t debian /bin/bash
root@7cb9a56cbcf5:/# echo "hello docker"
hello docker
root@7cb9a56cbcf5:/# exit
exit
ubuntu@ip-172-31-45-237:~$
```

run : 컨테이너 실행

-i : 컨테이너와 상호 입출력 가능 옵션

-t : 셀을 사용할 수 있는 tty 활성

/bin/bash : bash 쉘을 반환

- 도커 기본 명령어

  docker run -h new_host -i -t debian /bin/bash

```
ubuntu@ip-172-31-45-237:~$ docker run -h new_host -i -t debian /bin/bash
root@new_host:/#
```

  docker ps #실행중인 컨테이너 상세 정보 반환

```
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS          PORTS     NAMES
e4e9ccc99036   debian    "/bin/bash"   4 seconds ago    Up 4 seconds              gifted_sutherland
ubuntu@ip-172-31-45-237:~$
```

-h :컨테이너 호스트 이름 부여

- IMAGE : 컨테이너 생성 시 사용된 이미지 이름
- COMMAND : 컨테이너 시작 시 실행될 명령어
- CREATED : 컨테이너 생성된 이후 시간
- STATUS : 컨테이너 상태(UP : 실행중, Exited: 중지, Pause : 일시중지)
- PORTS : 컨테이너가 오픈한 포트와 호스트에 연결 상태
- NAMES : 컨테이너 고유 이름, 중복 불가능, 변경가능

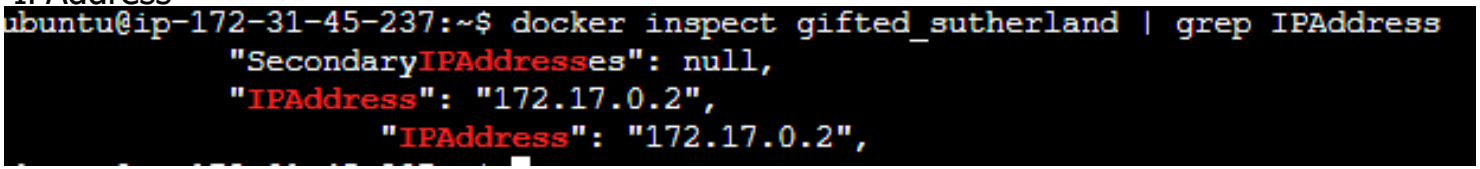
- 도커 기본 명령어

  docker insepct 컨테이너 명

  컨테이너의 많은 정보 출력

  grep 명령어로 필요한 정보를 찾을 수 있음.

  docker inspect gifted_sutherland | grep IPAddress



  exit

- 도커 기본 명령어

  docker docker ps -a

  모든 상태를 포함한 컨테이너 목록을 확인

```
ubuntu@ip-172-31-45-237:~$ docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS                      PORTS     NAMES
e4e9ccc99036   debian        "/bin/bash"              10 minutes ago   Exited (0) 2 minutes ago              gifted_sutherland
2e8f3b602ab0   debian        "/bin/bash"              11 minutes ago   Exited (0) 10 minutes ago             blissful_liskov
7cb9a56cbcf5   debian        "/bin/bash"              18 minutes ago   Exited (0) 18 minutes ago             keen_stonebraker
8ed084e7a91c   debian        "echo 'hello World'"     25 minutes ago   Exited (0) 25 minutes ago             optimistic_hugle
cf31553748db   debian        "echo 'hello World'"     26 minutes ago   Exited (0) 26 minutes ago             stupefied_blackburn
8bd81a1b3d45   hello-world   "/hello"                 45 minutes ago   Exited (0) 45 minutes ago             inspiring_wilson
```

  docker start 컨테이너명 (컨테이너 시작)

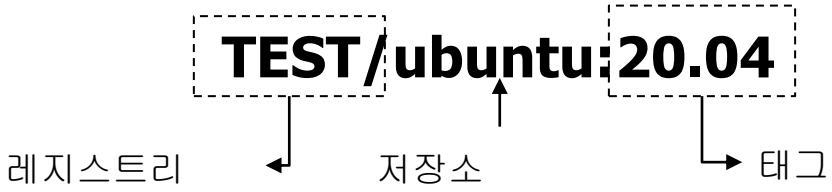
  docker rm 컨테이너명 (컨테이너 삭제)

  docker container prune (중지된 모든 컨테이너 삭제)

  docker rm -v $(docker ps -aq) (모든 컨테이너 삭제)

- 도커 이미지
  - 도커 이미지는 도커 허브(Docker Hub)라는 중앙 이미지 저장소에서 다운로드함.
  - docker create, run, pull 명령어로 이미지 다운로드 가능
  - 도커 계정 생성 후 이미지 업로드 및 다운로드 가능
  - 도커 허브 비공개 저장소는 요금을 지불해야 사용 가능
  - 이미지 저장소를 직접 구축해 사용 가능

- 도커 이미지
  - 가상머신 생성 시 사용하는 ISO와 비슷한 개념
  - 여러 개의 층으로 된 바이너리 파일로 존재
  - 컨테이너 생성 시 읽기 전용으로 사용됨
  - 도커 명령어로 레지스트리(Docker hub)로부터 다운 가능



  - 레지스트리(Registry) 이름: 이미지를 운영하고 배포하는 역할 (생략 시 Docker hub)
  - 저장소(Repository) 이름: 관련된 이미지의 집합 (생략 불가능)
  - 태그(Tag) : 저장소에 있는 이미지에 붙여진 알파벳과 숫자로 된 구분자 (생략 시 latest)

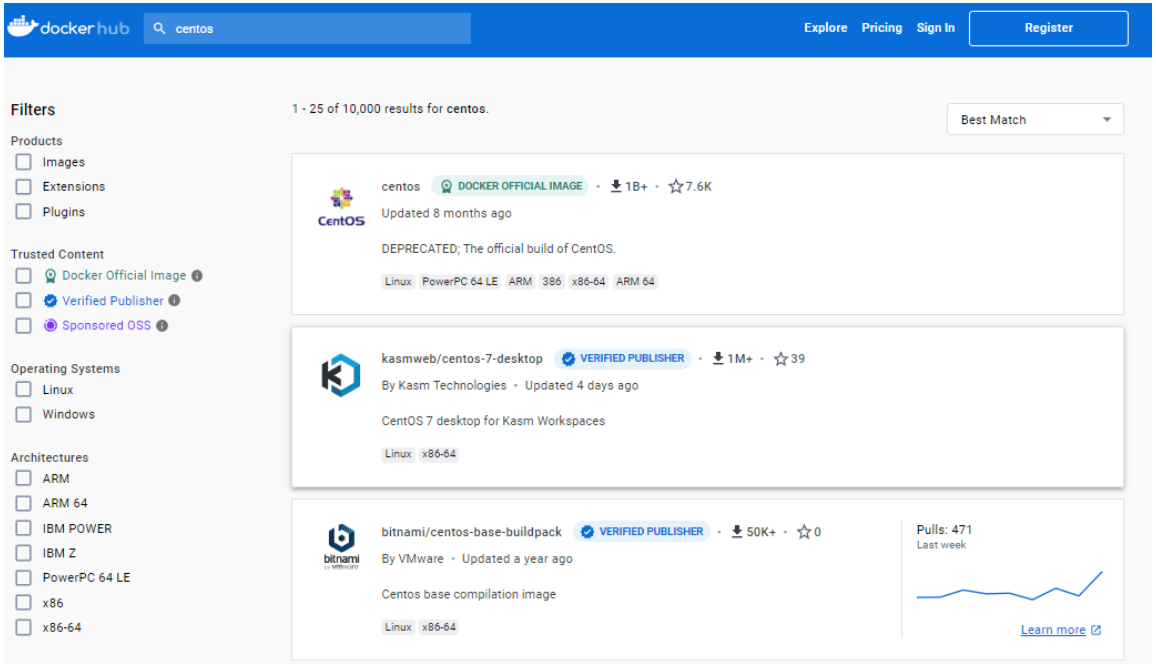
- 도커 이미지 검색하기
  - docker search 저장소명

docker search centos

```
ubuntu@ip-172-31-45-237:~$ docker search centos
NAME                              DESCRIPTION                                     STARS     OFFICIAL
centos                            DEPRECATED; The official build of CentOS.       7713      [OK]
kasmweb/centos-7-desktop          CentOS 7 desktop for Kasm Workspaces            43
bitnami/centos-base-buildpack     Centos base compilation image                   0
dokken/centos-7                   CentOS 7 image for kitchen-dokken               6
dokken/centos-8                   CentOS 8 image for kitchen-dokken               6
spack/centos7                     CentOS 7 with Spack preinstalled                2
dokken/centos-6                   EOL: CentOS 6 image for kitchen-dokken          0
atlas/centos7-atlasos             ATLAS CentOS 7 Software Development OS          2
spack/centos6                     CentOS 6 with Spack preinstalled                1
ustclug/centos                    Official CentOS Image with USTC Mirror          0
dokken/centos-stream-8                                                            5
eclipse/centos_jdk8               CentOS, JDK8, Maven 3, git, curl, nmap, mc, …   5
dokken/centos-stream-9                                                            9
adoptopenjdk/centos7_build_image                                                  2
corpusops/centos-bare             https://github.com/corpusops/docker-images/     0
```

  - STARS : 도커 사용자로부터 즐겨찾기 숫자
  - OFFICAL : 공식 사이트에서 제공하는 이미지

- 도커 이미지 검색하기
  - hub.docker.com



- 도커 생성하기
  - 이미지 생성을 위한 컨테이너 생성

    docker create -i -t --name image_test ubuntu:20.04

  - docker commit [option] CONTAINER [저장소명[:TAG]]

    docker commit -a "Dankook" -m "My first commit" image_test my_ubuntu:1.0
  - -a : author (이미지 작성자)
  - -m : 커밋 메시지

```
ubuntu@ip-172-31-45-237:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED              SIZE
my_ubuntu     1.0       655ec43dc0f5   3 seconds ago        72.8MB
ubuntu        1         6cd3f4434199   34 seconds ago       72.8MB
ubuntu        20.04     3cff1c6ff37e   3 weeks ago          72.8MB
```

- 도커 이미지 확인
  - docker insepct [저장소명]

    docker inspect my_ubuntu:1.0

```
ubuntu@ip-172-31-45-237:~$ docker inspect my_ubuntu:1.0
[
    {
        "Id": "sha256:655ec43dc0f5bb6f947abd43a0997239aa5d44537bc7105ce1d82b0241219981",
        "RepoTags": [
            "my_ubuntu:1.0"
        ],
        "RepoDigests": [],
        "Parent": "sha256:3cff1c6ff37e0101a08119d0743ba95c7f505d00298f5a169129e4e9086cde9e",
        "Comment": "My first commit",
        "Created": "2024-03-10T14:07:04.490160208Z",
        "Container": "ceb1fabfa776adf9f30d22b9e592c26d02e9d8e662c49bc698690f731bcc3859",
        "ContainerConfig": {
            "Hostname": "ceb1fabfa776",
            "Domainname": "",
            "User": "",
```

- 도커 이미지 히스토리 확인
  - docker history [이미지명]

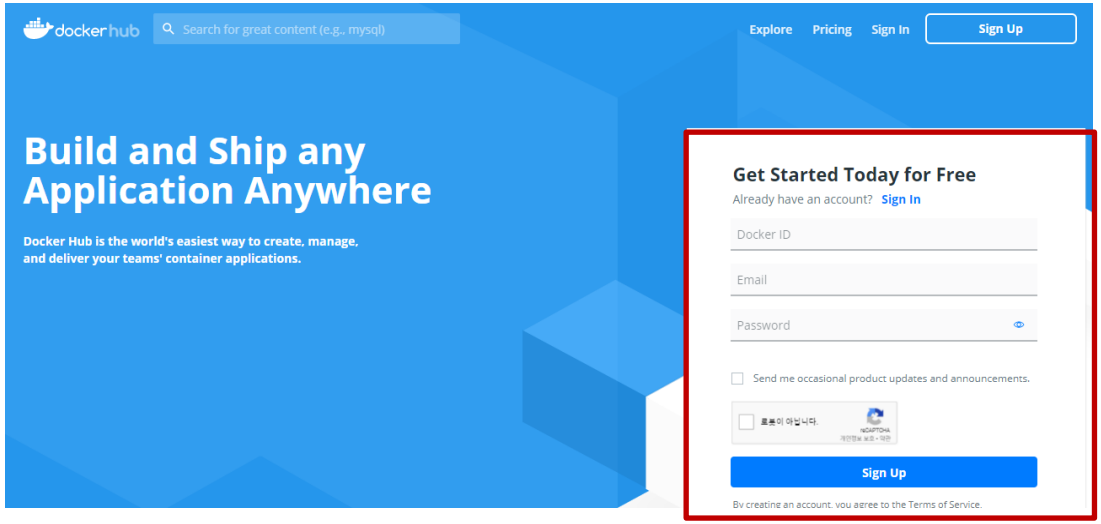
    docker history my_ubuntu:1.0

```
ubuntu@ip-172-31-45-237:~$ docker history my_ubuntu:1.0
IMAGE          CREATED         CREATED BY                                      SIZE      COMMENT
655ec43dc0f5   2 minutes ago   /bin/bash                                       0B        My first commit
3cff1c6ff37e   3 weeks ago     /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>      3 weeks ago     /bin/sh -c #(nop) ADD file:a25798f31219000d6…   72.8MB
<missing>      3 weeks ago     /bin/sh -c #(nop)  LABEL org.opencontainers.…   0B
<missing>      3 weeks ago     /bin/sh -c #(nop)  LABEL org.opencontainers.…   0B
<missing>      3 weeks ago     /bin/sh -c #(nop)  ARG LAUNCHPAD_BUILD_ARCH     0B
<missing>      3 weeks ago     /bin/sh -c #(nop)  ARG RELEASE                  0B
```

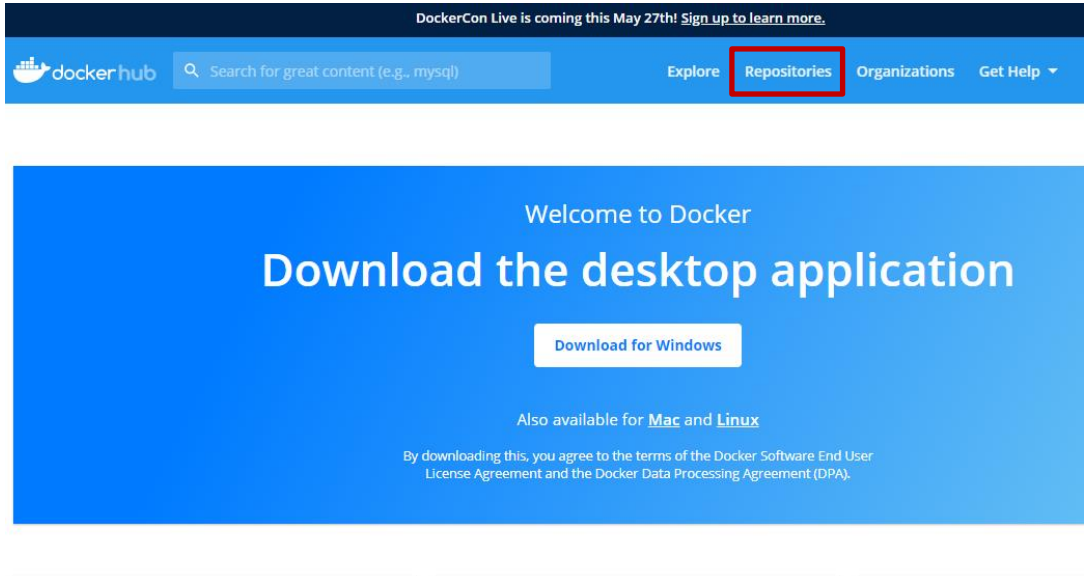
- 도커 이미지 삭제
  - docker rmi [이미지명]
    - 이미지를 사용중인 컨테이너 존재할 경우 삭제할 수 없음
    - docker rm -r 로 삭제 가능

- 도커 이미지 배포
  - 파일 배포
    - 추출한 이미지 파일을 복사 후 저장
  - 도커 허브
    - 이미지 클라우드 저장소
    - Public 무료저장소와 Private 유료 저장소 사용가능
    - 무료 플랜에서는 공개 저장소는 무제한, 사설은 최대 1개 가능
  - 사설 레지스트리
    - 사용자가 직접 도커 이미지 저장소를 직접 구성
    - 저장소 서버, 저장공간을 사용자가 직접 관리
    - 회사 사내망 환경에서 이미지 배포 시 좋은 방법

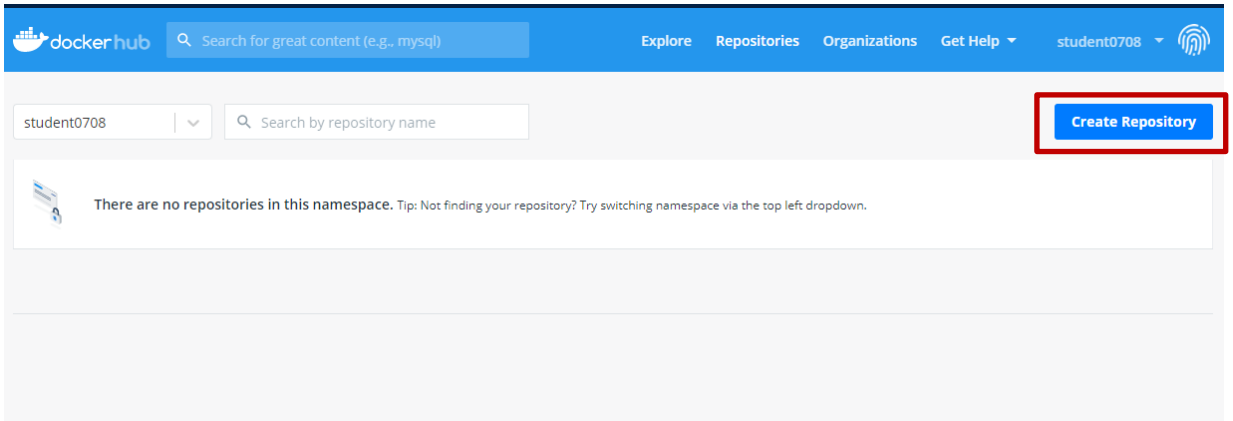
- 이미지를 공유하기 위해서는 **Docker hub**에 회원 가입 및 로그인
  - hub.docker.com



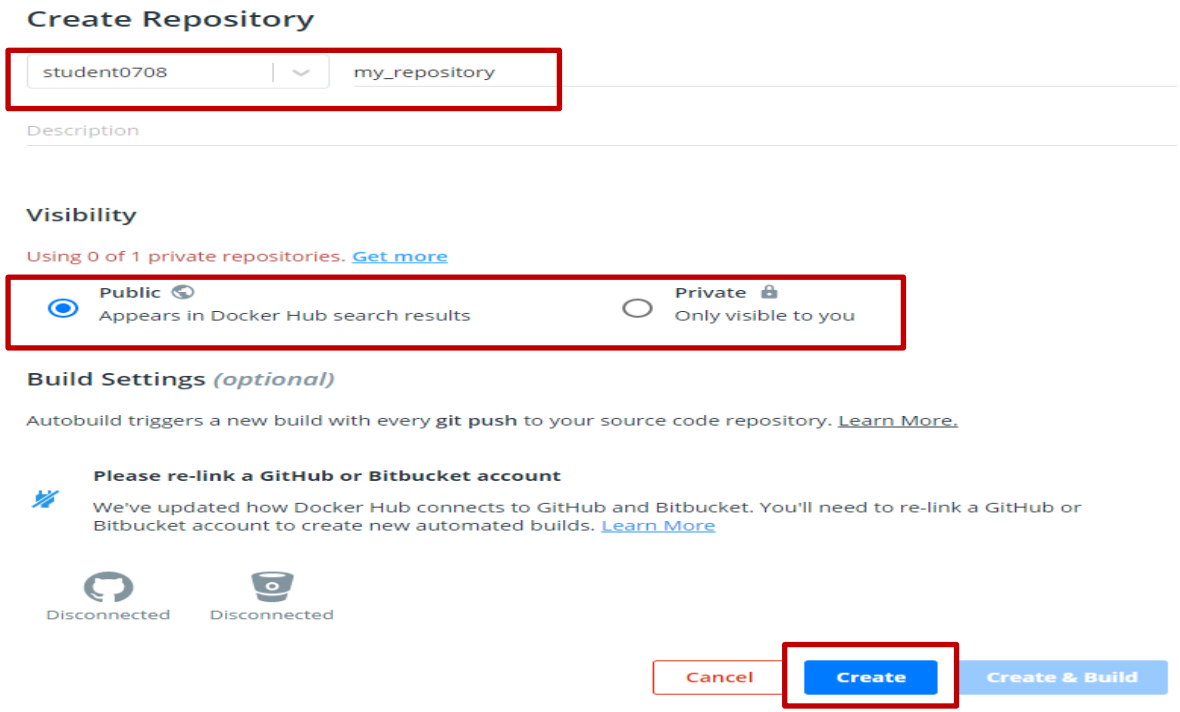
- 로그인 후 상단 [Repositories] 메뉴 클릭



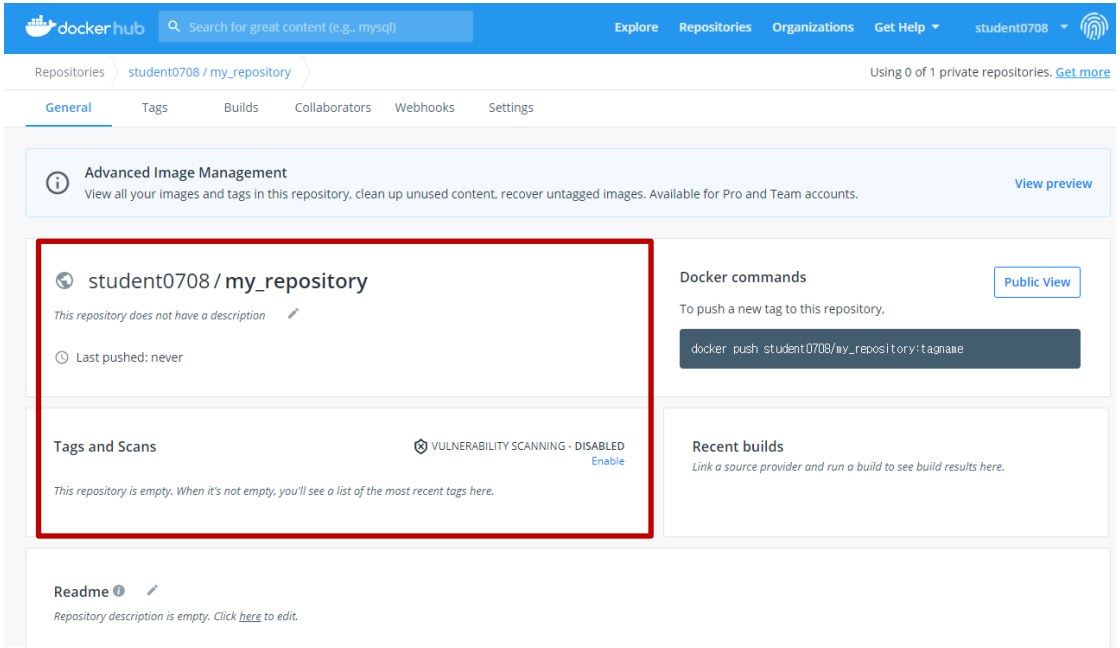
- [Create Repository] 버튼을 클릭해 저장소 생성 화면으로 이동



- 저장소 이름 지정
- 공개 혹은 개인으로 저장소 생성



- 생성된 저장소를 확인


docker hub
Search for great content (e.g., mysql)
Explore
Repositories
Organizations
Get Help
student0708
Repositories
student0708 / my_repository
Using 0 of 1 private repositories. Get more
General
Tags
Builds
Collaborators
Webhooks
Settings
Advanced Image Management
View all your images and tags in this repository, clean up unused content, recover untagged images. Available for Pro and Team accounts.
View preview
student0708 / my_repository
This repository does not have a description
Last pushed: never
Docker commands
Public View
To push a new tag to this repository,
docker push student0708/my_repository:tagname
Tags and Scans
VULNERABILITY SCANNING - DISABLED
Enable
This repository is empty. When it's not empty, you'll see a list of the most recent tags here.
Recent builds
Link a source provider and run a build to see build results here.
Readme
Repository description is empty. Click here to edit.

- 콘솔로 돌아가 login
- 로그인하기 위해서는 docker login 명령어를 사용
- ID와 패스워드는 Docker 허브에 가입한 계정 정보 입력

```
ubuntu@ip-172-31-45-237:~$ docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you do
cker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT
zations using SSO. Learn more at https://docs.docker.com/go/access-tokens/

Username: shjeon0617
Password:
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

- 저장소 이름과 동일하게 이미지 새롭게 생성

  docker commit -a "Dankook" -m "my seccond commit" image_test shjeon0617/my_ubuntu:1.0

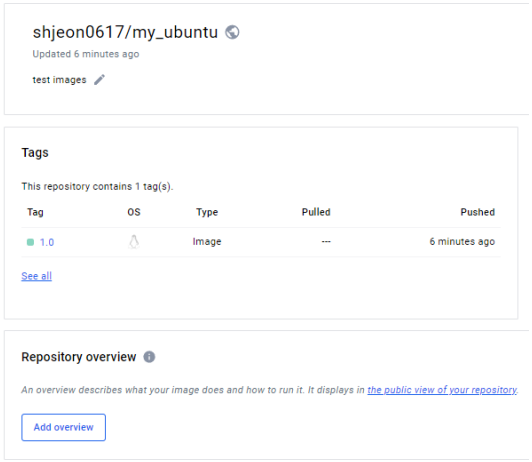
- 이미지 업로드하기

  docker push shjeon0617/my_ubuntu:1.0

  docker push [option] 레지스트리/저장소명:태그명

```
ubuntu@ip-172-31-45-237:~$ docker commit -a "Dankook" -m "my seccond commit" image_test shjeon0617/my_ubuntu:1.0
sha256:2544e603efc02e21f0550f27dcaff9db5282233aa79d9eeb795b00e303c9cb3c
ubuntu@ip-172-31-45-237:~$ docker push shjeon0617/my_ubuntu:1.0
The push refers to repository [docker.io/shjeon0617/my_ubuntu]
5faf9c0a9efe: Mounted from shjeon0617/ubuntu
1.0: digest: sha256:169bda65ff3a218dcabe581829963b0456e9394f3bb89b91e3bc8c849f2109b0 size: 529
```

- 이미지 업로드하기
  - 업로드 후 **Docker hub**에서 업로드 된 이미지 확인

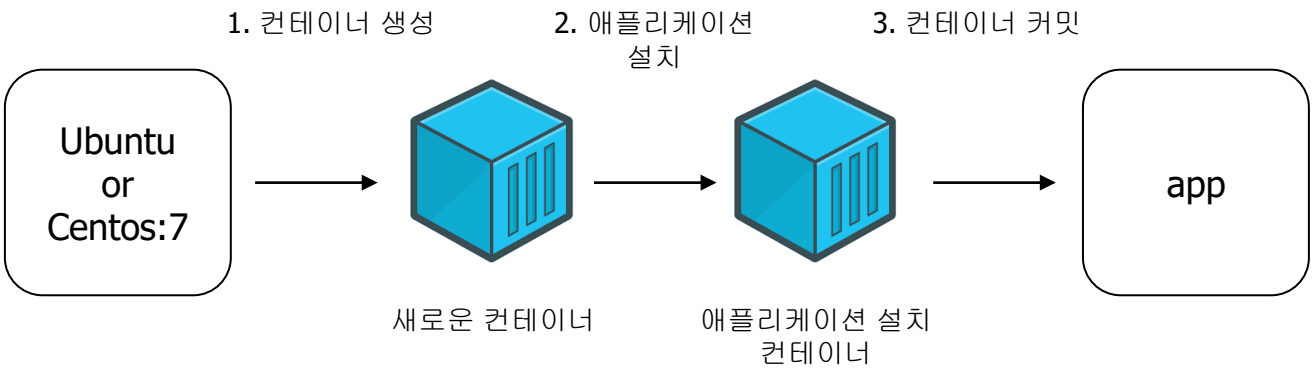


  - 콘솔에서 이미지 검색

```
ubuntu@ip-172-31-45-237:~$ docker search shjeon0617
NAME                    DESCRIPTION   STARS     OFFICIAL
shjeon0617/my_ubuntu                  0
```

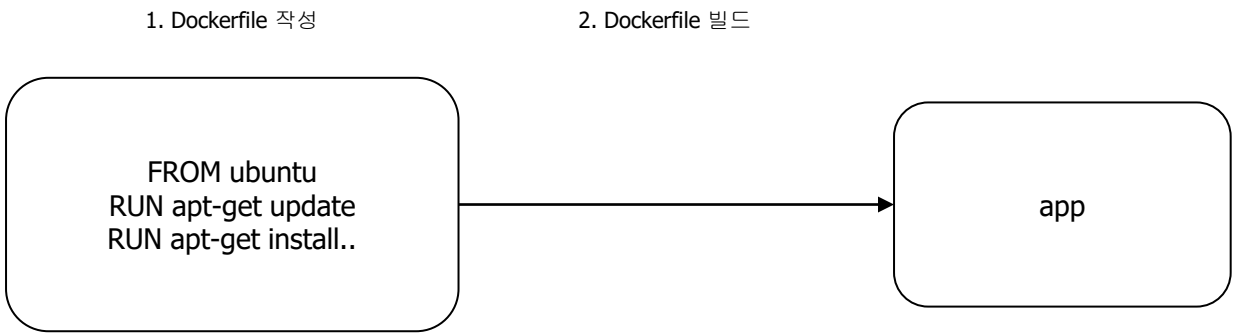
- 컨테이너로 이미지 생성

1. 이미지로 컨테이너 생성
2. 애플리케이션 설치 및 환경설정, 소스코드 복제
3. 컨테이너 이미지 커밋(commit)



- Dockerfile로 이미지 생성

1. Dockerfile 작성
2. 빌드 명령어로 Dockerfile을 통해 이미지 생성



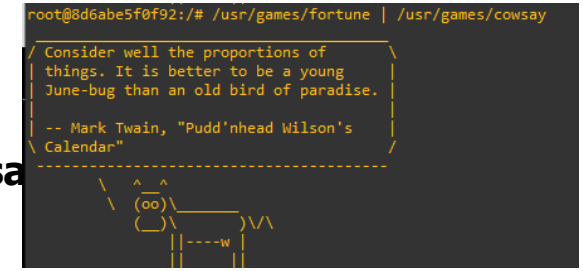
- 컨테이너로 이미지 생성

  docker run -it --name cowsay --hostname cowsay ubuntu:14.04 bash

  root@game:/# **apt-get update**

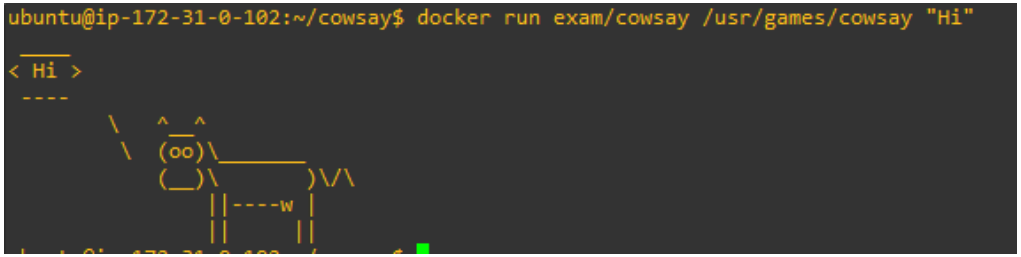
  root@game:/# **apt-get install -y cowsay fortune**

  root@game:/# **/usr/games/fortune | /usr/games/cowsa**



  docker commit cowsay exam/cowsay

  docker run exam/cowsay /usr/games/cowsay "Hi"



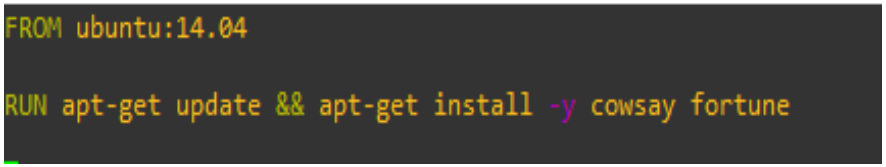
- Dockerfile

  mkdir cowsay

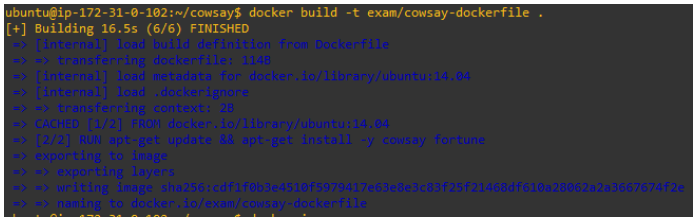
  cd cowsay

  vi Dockerfile

```
FROM ubuntu:14.04

RUN apt-get update && apt-get install -y cowsay fortune
```



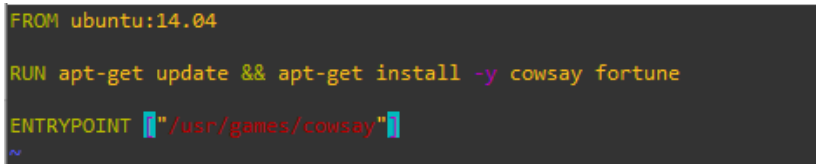
  docker build -t exam/cowsay-dockerfile .



  docker run exam/cowsay-dockerfile /usr/games/cowsay "Hi"

- Dockerfile
- ENTRYPOINT
  - ENTRYPOINT를 이용하면 도커파일을 더 활용할 수 있다.

```
FROM ubuntu:14.04

RUN apt-get update && apt-get install -y cowsay fortune

ENTRYPOINT ["/usr/games/cowsay"]
```




```
docker build -t exam/cowsay-dockerfile .
```

```
docker run exam/cowsay-dockerfile "Hi"
```

ENTRYPOINT - Docker 컨테이너가 실행될 때 기본적으로 실행되는 명령어를 정의하
컨테이너가 시작되면 ENTRYPOINT에 설정된 명령어가 실행

- 호스트 공유 볼륨 공유
  - MySQL 컨테이너 생성

docker run -d --name database -e MYSQL_ROOT_PASSWORD=test1 -e MYSQL_DATABASE=mydb -v /home/mydb:/var/lib/mysql mysql:5.7

- -d : 백그라운드(Detached)모드 실행, 컨테이너가 백그라운드에서 계속 실행되도록 함.
- -v : 호스트 디렉토리 및 파일을 컨테이너의 디렉토리 및 파일과 공유.

  ex) 호스트 : /home/mydb | 컨테이너 : /var/lib/mysql

- 호스트 /home/mydb 폴더

ls /home/mydb

```
ubuntu@ip-172-31-36-152:/home$ ls /home/mydb/
auto.cnf    client-cert.pem  ib_logfile0  ibtmp1  mysql.sock          public_key.pem   sys
ca-key.pem  client-key.pem   ib_logfile1  mydb    performance_schema  server-cert.pem
ca.pem      ib_buffer_pool   ibdata1      mysql   private_key.pem     server-key.pem
```

- database 컨테이너 폴더 확인

docker exec database ls /var/lib/mysql

```
ubuntu@ip-172-31-36-152:/home$ docker exec database ls /var/lib/mysql
auto.cnf
ca-key.pem
ca.pem
client-cert.pem
client-key.pem
ib_buffer_pool
ib_logfile0
ib_logfile1
ibdata1
```

- 호스트 공유 볼륨 공유

  - 옵션으로 볼륨을 사용하는 컨테이너를 다른 컨테이너와 공유
  - 새로운 컨테이너 생성

docker run -it --name exam -h exam --volumes-from database ubuntu

```
ubuntu@ip-172-31-36-152:/home$ docker run -it --name exam -h exam --volumes-from database ubuntu
root@exam:/# ls /var/lib/mysql/
auto.cnf    client-cert.pem  ib_logfile0  ibtmp1  mysql.sock          public_key.pem   sys
ca-key.pem  client-key.pem   ib_logfile1  mydb    performance_schema  server-cert.pem
ca.pem      ib_buffer_pool   ibdata1      mysql   private_key.pem     server-key.pem
root@exam:/#
```

□ --volumes-from 컨테이너명: 다른 컨테이너와 볼륨 공유

- 도커 볼륨 기능
  - 도커 볼륨 생성

docker volume create --name myvol

  - 생성된 볼륨 확인

docker volume ls

```
ubuntu@ip-172-31-36-152:/home$ docker volume ls
DRIVER    VOLUME NAME
local     db_data
local     ebe80c1fce3751e648dc41bb1cc6dc02f7a14a19b0d2472fd3adda091dbaca43
local     myvol
```

- 도커 볼륨 기능
  - voltest1 컨테이너 생성 후 파일 생성

docker run -it --name voltest1 -v myvol:/root/ ubuntu

echo hello >> /root/volume

exit (컨테이너 나오기)

docker volume ls

  - voltest2 컨테이너 생성 후 파일 확인

docker run -it --name voltest2 -v myvol:/root/ ubuntu

cat /root/volume

```
root@f0cc108db8fc:/# cat /root/volume
hello
```

- 도커 볼륨 정보
  - 도커 엔진에서 볼륨은 디렉토리에 상응하는 단위
  - 볼륨은 다양한 스토리지 백엔드 플러그인 드라이버 사용가능
  - 기본적으로 제공하는 드라이버는 local
  - 볼륨 정보 확인

    docker inspect myvol

```
ubuntu@ip-172-31-36-152:/home$ docker inspect myvol
[
    {
        "CreatedAt": "2024-03-16T09:01:11Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/myvol/_data",
        "Name": "myvol",
        "Options": null,
        "Scope": "local"
    }
]
```

- 도커 볼륨 삭제
  - 볼륨은 컨테이너를 삭제해도 자동으로 삭제되지 않음.
  - 사용하지 않는 볼륨 삭제

docker volume rm 볼륨명

  - 사용하지 않은 볼륨 한번에 삭제

docker volume prune

```
ubuntu@ip-172-31-36-152:/home$ docker volume prune
WARNING! This will remove anonymous local volumes not used by at least one container.
Are you sure you want to continue? [y/N] y
Deleted Volumes:
ebe80c1fce375le648dc41bb1cc6dc02f7a14a19b0d2472fd3adda091dbaca43

Total reclaimed space: 206.9MB
```

- 도커 run 명령어로 MySQL 컨테이너 만들기

  docker run -d --name wpdb -v db_data:/var/lib/mysql --restart=always -e MYSQL_ROOT_PASSWORD=somewordpress -e MYSQL_DATABASE=wordpress -e MYSQL_USER=wordpress -e MYSQL_PASSWORD=wordpress mysql:5.7

```
ubuntu@ip-172-31-36-152:/home$ docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS                 NAMES
ee2cd221c51e   mysql:5.7   "docker-entrypoint.s…"   41 seconds ago   Up 38 seconds   3306/tcp, 33060/tcp   wpdb
```

  --restart=always : 컨테이너 항상 재시작

  -d : 백그라운드 생성

  -e : 컨테이너 환경변수

- 도커 run 명령어로 워드프레스 컨테이너 만들기

  docker run -d --name=wp -v wordpress_data:/var/www/html --restart=always -p 8000:80 -e WORDPRESS_DB_PASSWORD=wordpress -e WORDPRESS_DB_NAME=wordpress -e WORDPRESS_DB_USER=wordpress -e WORDPRESS_DB_HOST=db --link wpdb:db wordpress

```
ubuntu@ip-172-31-36-152:/home$ docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED         STATUS         PORTS                                   NAMES
d97156c1e01b   wordpress   "docker-entrypoint.s…"   8 seconds ago   Up 6 seconds   0.0.0.0:8000->80/tcp, :::8000->80/tcp   wp
ee2cd221c51e   mysql:5.7   "docker-entrypoint.s…"   2 minutes ago   Up 2 minutes   3306/tcp, 33060/tcp                     wpdb
```

  --link : 컨테이너 간 접근 시 별명으로 접근 가능

  -> wpdb 컨테이너를 db 별명으로 접근 가능

  ➔--link에 입력된 컨테이너가 중지 혹은 존재하지 않을 경우 실행 불가

  -p : 호스트의 특정 포트와 컨테이너의 포트 연결

  -> 호스트의 8000포트와 컨테이너의 80포트 연결

- Docker Compose를 이용해 컨테이너 만들기

- Docker Compose(도커 컴포즈)
  - 단일 서버에서 여러 개의 컨테이너를 하나의 서비스로 정의해 컨테이너를 묶어 관리할 수 있는 작업환경을 제공

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose  #docker-compose 설치
sudo chmod +x /usr/local/bin/docker-compose #권한 설정
```

- Docker Compose를 이용해 컨테이너 만들기

- docker stop 및 docker rm 명령어로 컨테이너 삭제
- 프로젝트를 위한 디렉토리 생성 및 yml파일 생성

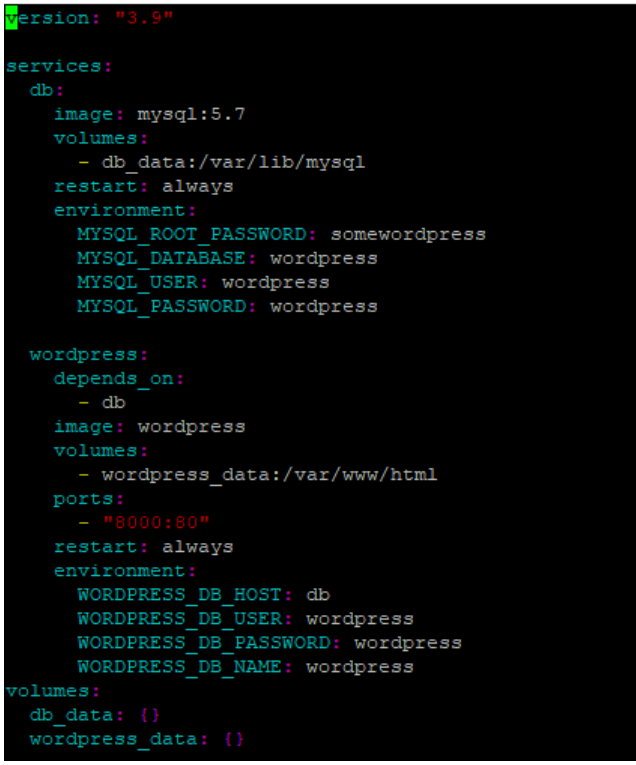
  mkdir project

  cd project

  vi compose.yml

  -> 파일명은 docker-compose.yml, docker-compose.yaml, compose.yml, compose.yaml 로 해야 함

```
version: "3.9"

services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress
    volumes:
      - wordpress_data:/var/www/html
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
volumes:
  db_data: {}
  wordpress_data: {}
```



```yaml
version: "3.9"

services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress
    volumes:
      - wordpress_data:/var/www/html
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
volumes:
  db_data: {}
  wordpress_data: {}
```

db : 컨테이너 이름 (폴더명_서비스명_n)

image : 사용할 이미지

volumes : 볼륨 설정

environment : 내부 컨테이너 환경 변수

depends_on 컨테이너 연결

ports : 포트 연결

- Docker Compose 명령어

up : 컨테이너 생성/시작

-d : 백그라운드에서 실행 / --build : 이미지 빌드 / --scale SERVICE=N

down : 리소스 삭제

--rmi all : 모든 이미지 삭제 / -v : Compose 정의된 볼륨 삭제

-f : compose.yml 현재 디렉토리에 파일이 없거나 이름이 다르면 -f 옵션으로 경로 설정 가능

ex) docker-compose -f /home/ubuntu/project.yml up

- Docker Compose 명령어

docker-compose -h

```
ubuntu@ip-172-31-0-102:~/project$ docker-compose -h
Define and run multi-container applications with Docker.

Usage:
  docker-compose [-f <arg>...] [--profile <name>...] [options] [--] [COMMAND] [ARGS...]
  docker-compose -h|--help

Options:
  -f, --file FILE             Specify an alternate compose file
                              (default: docker-compose.yml)
  -p, --project-name NAME     Specify an alternate project name
                              (default: directory name)
  --profile NAME              Specify a profile to enable
  -c, --context NAME          Specify a context name
  --verbose                   Show more output
  --log-level LEVEL           Set log level (DEBUG, INFO, WARNING, ERROR, CRITICAL)
  --ansi (never|always|auto)  Control when to print ANSI control characters
  --no-ansi                   Do not print ANSI control characters (DEPRECATED)
  -v, --version               Print version and exit
  -H, --host HOST             Daemon socket to connect to

  --tls                       Use TLS; implied by --tlsverify
  --tlscacert CA_PATH         Trust certs signed only by this CA
  --tlscert CLIENT_CERT_PATH  Path to TLS certificate file
  --tlskey TLS_KEY_PATH       Path to TLS key file
  --tlsverify                 Use TLS and verify the remote
  --skip-hostname-check       Don't check the daemon's hostname against the
                              name specified in the client certificate
  --project-directory PATH    Specify an alternate working directory
                              (default: the path of the Compose file)
  --compatibility             If set, Compose will attempt to convert keys
                              in v3 files to their non-Swarm equivalent (DEPRECATED)
  --env-file PATH             Specify an alternate environment file

Commands:
  build              Build or rebuild services
  config             Validate and view the Compose file
  create             Create services
  down               Stop and remove resources
  events             Receive real time events from containers
  exec               Execute a command in a running container
  help               Get help on a command
  images             List images
  kill               Kill containers
  logs               View output from containers
  pause              Pause services
  port               Print the public port for a port binding
  ps                 List containers
  pull               Pull service images
  push               Push service images
  restart            Restart services
  rm                 Remove stopped containers
  run                Run a one-off command
  scale              Set number of containers for a service
  start              Start services
  stop               Stop services
  top                Display the running processes
  unpause            Unpause services
  up                 Create and start containers
  version            Show version information and quit
```

- Docker Compose를 이용해 컨테이너 생성

docker-compose up -d

```
ubuntu@ip-172-31-0-102:~/project$ docker-compose up -d
Creating network "project_default" with the default driver
Creating project_db_1 ... done
Creating project_wordpress_1 ... done
```

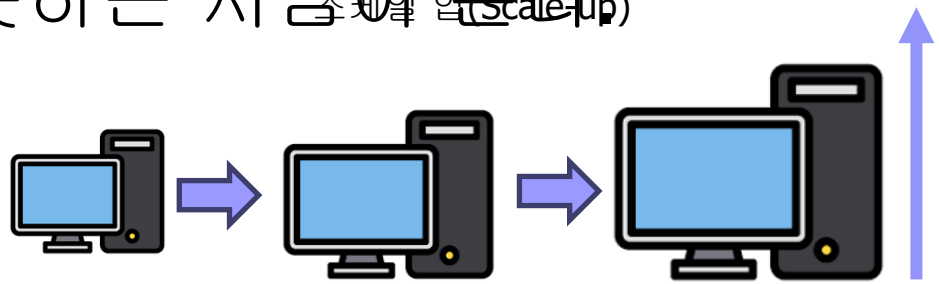
docker ps

```
ubuntu@ip-172-31-0-102:~/project$ docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS                                   NAMES
b5c3cae8603b   wordpress   "docker-entrypoint.s…"   53 seconds ago   Up 52 seconds   0.0.0.0:8000->80/tcp, :::8000->80/tcp   project_wordpress_1
3b7e9c47e3ab   mysql:5.7   "docker-entrypoint.s…"   54 seconds ago   Up 53 seconds   3306/tcp, 33060/tcp                     project_db_1
```

- 전통적인 데이터 처리 시스템
  - 과거에는 데이터 처리를 위한 시스템 확장이 어려웠음
  - 전통적인 방식에서 한 대의 컴퓨터 처리 능력에 제한

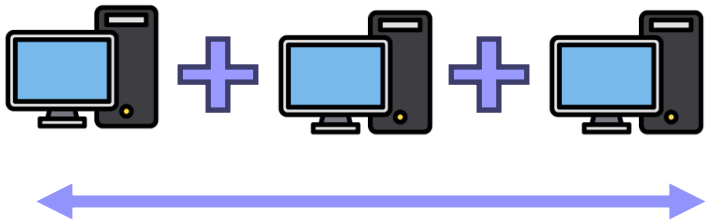
- 스케일 업
  - 시스템을 확장할 때 구조를 바꿀 필요가 없다.
  - 소프트웨어를 고사양 서버로 단순히 이관하는 방식으로 확장석을 높임 (생각보다 어려움)
  - 언젠가는 스케일 업 방식으로 더 이상 확장하지 못하는 시점이 옴

스케일 업(Scale-up)



- 스케일 아웃
  - 여러 대의 장비에 처리를 분산 시키는 방식
  - 스케일 업 방식에 비해 비용이 저렴
  - 서버들에 데이터 처리 방법을 개발해야 했음.
  - 스케쥴링, 장애처리 고려 등

스케일 아웃(Scale-out)



- 한계점
  - **스케일 업 : 비용이 비쌌음.**
  - **스케일 아웃 : 시스템 개발 및 관리가 어려움**
  - 큰 기업, 정부 기관, 학계를 제외한 곳에서 전통적인 스케일 업 및 스케일 아웃 방식을 그다지 사용하지 않았음.
  - 여러 대의 호스트나 여러 개의 CPU 성능을 효과적으로 활용하기 어려움.
  - 원하는 만큼 작업량을 효율적 처리하기 위한 전략은 엄청난 노력을 요함.

- 한계점
  - 데이터양을 늘어나지만 하드웨어는 한계가 있음
  - 고사양 서버를 한 대가 아닌 두 대, 세 대 ?
  - 하이브리드 아키텍처는 **하드웨어 구입 비용** 및 **클러스터 관리**를 위한 로직 개발 **두 가지 모두 필요**.
  - 빅데이터 처리 업계에서는 **스케일 아웃 방식이 사실상 표준**

- 솔루션
  - 많은 작업을 병렬 처리하는 하드웨어를 유연하게 사용하기 위해서는 **소프트웨어를 똑똑하고, 하드웨어는 단순하게**
  - 하드웨어는 **리소스 셋**으로만 이용
  - 소프트웨어가 처리 작업들에 하드웨어를 할당하는 역할
  - 서버가 늘어나도 각 서버의 장애나 문제점은 영향을 주지 않게 해야함.
  - 각각의 장비는 주기적으로 **장애가 발생**할 수 있다는 점을 고려

- 하둡
  - 2003년, 2004년 구글 내부의 기술을 설명하는
  - 구글 파일시스템(GFS)과 맵리듀스(MapReduce)
  - 더그 커팅은 구글의 GFS와 맵리듀스 논문에서 영감을 받아 시스템을 구현
  - 하둡은 아파치 오픈소스 재단의 최상위 레벨 프로젝트

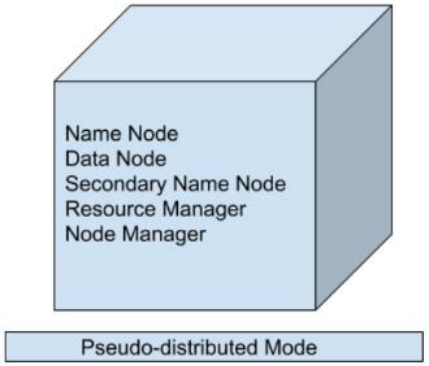
출처 : https://post.naver.com/viewer/postView.nhn?volumeNo=28925185&memberNo=50533718

- 하둡 구성요소
  - 하둡 분산 파일 시스템(HDFS, Hadoop Distributed File System)
    - 클러스터에 스케일 아웃 방식으로 대용량 데이터 셋을 저장하는 파일시스템
    - 지연율보다 처리율에 최적화 / 이중화가 아닌 데이터 복제를 통해 고가용성을 얻음
  - 맵리듀스(MapReduce)
    - 대용량 데이터를 병렬 처리 하기 위해 개발된 프로그래밍 모델
    - 입력 데이터를 분산 처리하는 맵(Map) 함수 단계와 다시 하나의 결과물로 합치는 리듀스(Reduce) 함수

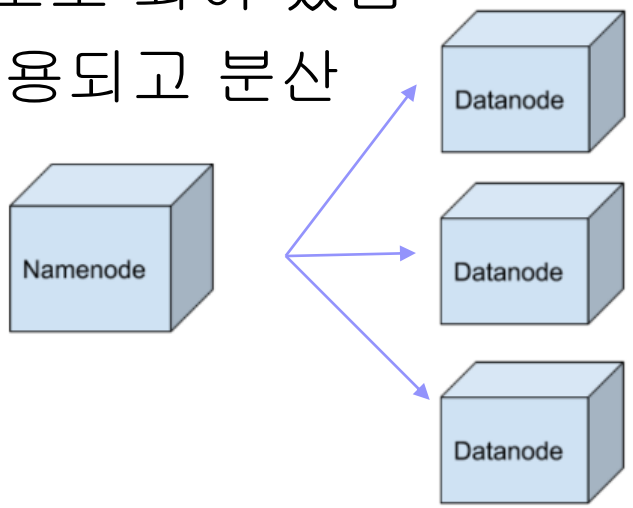
- 공통 구성 요소
  - HDFS와 맵리듀스는 아래와 같은 원칙을 지킨다.
    - 저가 서버들로 구성한 클러스터에서 구동하도록 설계
    - 서버를 추가함으로써 용량 및 성능을 확장하는 방식
    - 장애 탐지 및 대응 메커니즘
    - 사용자는 해결할 문제 자체만 집중하도록 시스템의 많은 부분을 드러내지 않는다.
    - 물리적 시스템을 제어하는 소프트웨어 클러스터를 구성하는 아키텍처

- 하둡의 세 가지 구동방식
  - 로컬 자립형 방식(Standalone Mode)
    - Hadoop이 실행되는 기본 모드
    - HDFS를 사용하지 않음
    - mapred-site.xml, core-site.xml, hdfs-site.xml 등 구성 파일을 수정할 필요 없음
    - 디버깅 목적으로 사용

- 하둡의 세 가지 구동방식
  - 가분산 방식(Pseudo-distributed Mode)
    - Namenode와 Datanode가 모두 동일한 시스템에 있음
    - 최적화된 미니 클러스터를 효과적
    - 구성 파일 변경이 필요(hdfs, core ..



- 하둡의 세 가지 구동방식
  - 완전분산 방식(Fully-Distributed Mode)
    - masternode와 datanode가 별도로 되어 있음
    - 데이터가 여러 노드에 걸쳐 사용되고 분산
    - 구성 파일 변경이 필요



- 실습 폴더 생성

  mkdir hadoop

  cd hadoop

- Dockerfile 이용해 Hadoop 이미지 생성

  core-site.xml, mapred-site.xml, hdfs-site.xml 작성

- core-site.xml 작성
  - 하둡 핵심 property 정의

vi core-site.xml

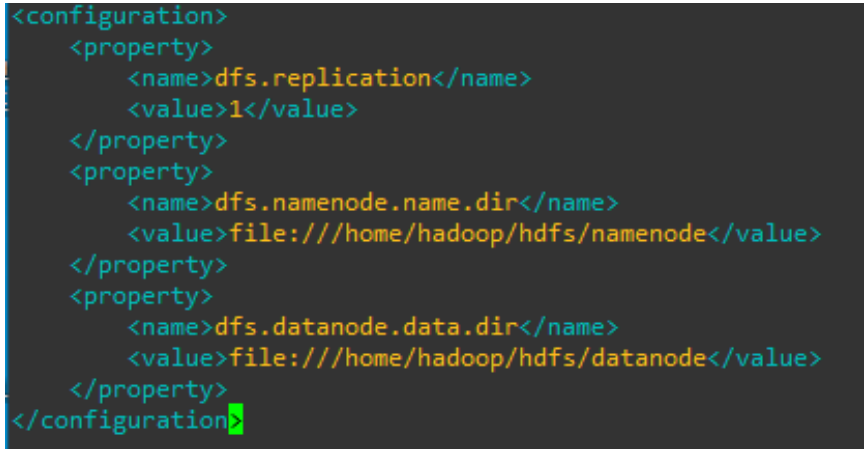
```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://namenode:9000</value>
    </property>
</configuration>
```


<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://namenode:9000</value>
</property>
</configuration>

hdfs-site.xml 작성

- namenode와 datanode 저장소 디렉토리 설정

```
<configuration>
   <property>
      <name>dfs.replication</name>
      <value>1</value>
   </property>
   <property>
      <name>dfs.namenode.name.dir</name>
      <value>file:///home/hadoop/hdfs/namenode</value>
   </property>
   <property>
      <name>dfs.datanode.data.dir</name>
      <value>file:///home/hadoop/hdfs/datanode</value>
   </property>
</configuration>
```

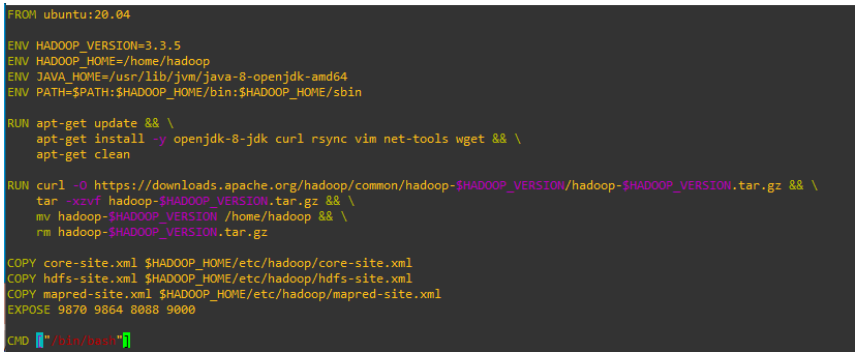


- mapred-site.xml 작성
  - 맵리듀스 파일 값을 정의

```
<configuration>
   <property>
     <name>mapreduce.framework.name</name>
     <value>yarn</value>
   </property>
</configuration>
```

Dockerfile 작성

```
FROM ubuntu:20.04

ENV HADOOP_VERSION=3.3.5
ENV HADOOP_HOME=/home/hadoop
ENV JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
ENV PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

RUN apt-get update && ☐
    apt-get install -y openjdk-8-jdk curl rsync vim net-tools wget && ☐
    apt-get clean

RUN curl -O https://downloads.apache.org/hadoop/common/hadoop-$HADOOP_VERSION/hadoop-$HADOOP_VERSION.tar.gz && ☐
    tar -xzvf hadoop-$HADOOP_VERSION.tar.gz && ☐
    mv hadoop-$HADOOP_VERSION /home/hadoop && ☐
    rm hadoop-$HADOOP_VERSION.tar.gz

COPY core-site.xml $HADOOP_HOME/etc/hadoop/core-site.xml
COPY hdfs-site.xml $HADOOP_HOME/etc/hadoop/hdfs-site.xml
COPY mapred-site.xml $HADOOP_HOME/etc/hadoop/mapred-site.xml
EXPOSE 9870 9864 8088 9000

CMD ["/bin/bash"]
```



ENV : 컨테이너 내부에 사용되는 환경 변수 설정
COPY : 호스트 머신의 파일을 컨테이너로 복제
EXPOSE : 포트 노출
CMD : 컨테이너 실행 시 기본으로 실행되는 명령

- docker build

  docker build -t test/hadoop .

```
spark@ubuntu-VirtualBox:~/hadoop$ docker build -t test/hadoop .
[+] Building 236.9s (11/11) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 801B
 => [internal] load metadata for docker.io/library/ubuntu:20.04
 => [internal] load .dockerignore
 => => transferring context: 2B
 => CACHED [1/6] FROM docker.io/library/ubuntu:20.04
 => [internal] load build context
 => => transferring context: 103B
 => [2/6] RUN apt-get update &&     apt-get install -y openjdk-8-jdk curl rsync vim net-tools &&     apt-
 => [3/6] RUN curl -O https://downloads.apache.org/hadoop/common/hadoop-3.3.5/hadoop-3.3.5.tar.gz &&
 => [4/6] COPY core-site.xml /usr/local/hadoop/etc/hadoop/core-site.xml
 => [5/6] COPY hdfs-site.xml /usr/local/hadoop/etc/hadoop/hdfs-site.xml
 => [6/6] COPY mapred-site.xml /usr/local/hadoop/etc/hadoop/mapred-site.xml
 => exporting to image
 => => exporting layers
 => => writing image sha256:95eea1b6b553ce5066899ca48bd493a87e1d35400aed8af25ee1d2da379bd460
 => => naming to docker.io/test/hadoop
```

docker-compose.yml 작성

```
version: '3'
services:
  namenode:
    image: test/hadoop
    platform: linux/amd64
    container_name: namenode
    hostname: namenode
    ports:
      - "9870:9870"
      - "9000:9000"
      - "8032:8032"
    command: "/bin/bash -c 'hdfs namenode -format && hdfs namenode'"
  datanode-1:
    image: test/hadoop
    platform: linux/amd64
    container_name: datanode-1
    hostname: datanode-1
    ports:
      - "9861:9864"
      - "9001:9000"
    command: ["hdfs", "datanode"]
  datanode-2:
    image: test/hadoop
    platform: linux/amd64
    container_name: datanode-2
    hostname: datanode-2
    ports:
      - "9862:9864"
      - "9002:9000"
    command: ["hdfs", "datanode"]
  datanode-3:
    image: test/hadoop
    platform: linux/amd64
    container_name: datanode-3
    hostname: datanode-3
    ports:
      - "9863:9864"
      - "9003:9000"
    command: ["hdfs", "datanode"]
```

- docker-compose

  docker-compose up -d

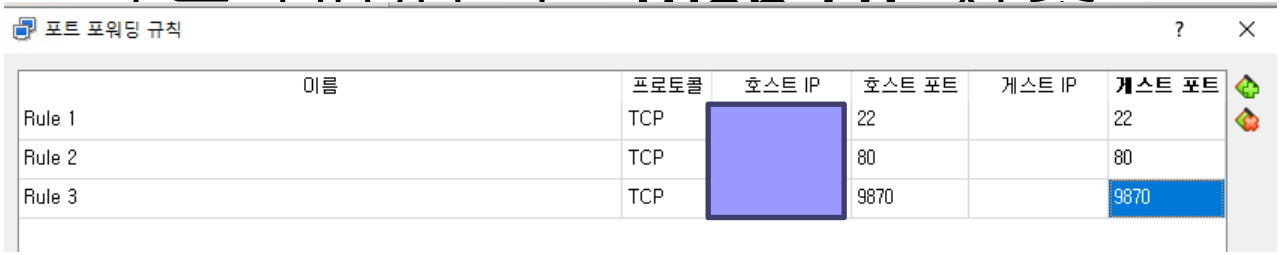
```
spark@ubuntu-VirtualBox:~/hadoop$ docker-compose up -d
WARN[0000] /home/spark/hadoop/compose.yml: the attribute `versio
[+] Running 4/4
 ✔ Container datanode-3  Started
 ✔ Container namenode    Started
 ✔ Container datanode-2  Started
 ✔ Container datanode-1  Started
```

```
spark@ubuntu-VirtualBox:~/hadoop$ docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS          PORTS
66b9b072fd45   test/hadoop     "hdfs datanode"          10 minutes ago   Up 10 minutes   8088/tcp, 9870/tcp, 0.0.0.0:9
d7b6295dc4dc   test/hadoop     "hdfs datanode"          10 minutes ago   Up 10 minutes   8088/tcp, 9870/tcp, 0.0.0.0:9
5370e7b532c9   test/hadoop     "/bin/bash -c 'hdfs …"   10 minutes ago   Up 10 minutes   0.0.0.0:8032->8032/tcp, :::80
c5db83689b3b   test/hadoop     "hdfs datanode"          10 minutes ago   Up 10 minutes   8088/tcp, 9870/tcp, 0.0.0.0:9
```

- 하둡 컨테이너 접속

  docker exec -it namenode /bin/bash

  하둡 네임노드 Web UI 접속



  - VirtualBox-네트워크-포트 포워딩- 호스트 포트와 게스트 포트 입력 (9870)

하둡 Web UI

Summary

Security is off.

Safe mode is ON. Resources are low on NN. Please add or free up more resourcesthen turn off safe mode manually. NOTE: If you turn off safe mode before adding resources, the NN will immediately return to safe mode. Use "hdfs dfsadmin -safemode leave" to turn safe mode off.

1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).

Heap Memory used 137.52 MB of 317 MB Heap Memory. Max Heap Memory is 1.72 GB.

Non Heap Memory used 52.23 MB of 53.31 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	145.26 GB
Configured Remote Capacity:	0 B
DFS Used:	84 KB (0%)
Non DFS Used:	141.86 GB
DFS Remaining:	0 B (0%)
Block Pool Used:	84 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	3 (Decommissioned: 0, In Maintenance: 0)

하둡 예제 (Wordcount)

- 하둡 명령어

hdfs dfs [GENERIC_OPTIONS] [COMMAND_OPTIONS]

- cat -파일 내용 출력

hdfs dfs -cat

- cp -hdfs 내부에서 파일을 복사

hdfs dfs -cp

- mkdir -특정 path에 폴더 생성

hdfs dfs -mkdir

- mv -hdfs 내부에서 파일 옮기기

hdfs dfs -mv

- put -local에서 파일을 hdfs에 저장

hdfs dfs -input

- copyToLocal -hdfs에 있는 파일을 local에 다운

hdfs dfs -copyToLocal

- du -hdfs 내부 특정 file이나 디렉토리의 사이즈를 보여줌

hdfs dfs -du

- ls -특정 디렉토리의 파일 혹은 디렉토리 출력

hdfs dfs -ls

- rm - hdfs에서 폴더 혹은 파일 삭제

hdfs dfs -rm

하둡 예제 (Wordcount)

- Wordcount에 사용될 예제 파일 다운
  - wget http://www.gutenberg.org/cache/epub/1661/pg1661.txt
- hdfs에 폴더 생성하기
  - hdfs dfs -mkdir -p /sample/input
- hdfs에 예제파일 이동하기
  - hdfs dfs -put pg1661.txt /sample/input

하둡 예제 (Wordcount)

- 워드카운드 실행하기

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.4.jar wordcount /sample/input /sample/output

- 결과물 확인

hdfs dfs -cat /sample/output/part*

```
sundial 1
sundial,        1
sundial,'       1
sundial.'"      1
sundial?'       1
sundials        1
sunk    8
sunk,   1
sunlight;       1
sunset, 1
sunshine.       1
superb  1
superior        1
superior,       1
superscribed    1
superscription  1
supper  4
supper, 2
supper."        1
supplementing   1
```

하둡 예제 (Wordcount)

- Teragen / terasort
  - 하둡 성능을 측정하기 위해 사용
  - 일정 데이터를 생성하고 데이터를 정렬할 때 속도를 측정

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.4.jar teragen ☐

-Ddfs.replication=1 ☐

-Dmapred.map.tasks=36 ☐

-Dmapred.reduce.tasks=16 ☐

5000 /teragen/teragen.data

복사본 1개 맵 테스트 36 / 리듀스 테스크 16 / 생성 용량 5,000/ 생성된 데이터 저장할 곳

하둡 예제 (Wordcount)

- Teragen

```
2024-03-24 18:07:40,871 INFO mapred.Task: Final Counters for attempt_local1133262668_0001_m_000000_0: Counters: 22
        File System Counters
                FILE: Number of bytes read=316597
                FILE: Number of bytes written=866543
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=0
                HDFS: Number of bytes written=10737418200
                HDFS: Number of read operations=5
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=3
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=107374182
                Map output records=107374182
                Input split bytes=83
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=173
                Total committed heap usage (bytes)=297271296
        org.apache.hadoop.examples.terasort.TeraGen$Counters
                CHECKSUM=230593859918397906
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=10737418200
2024-03-24 18:07:40,871 INFO mapred.LocalJobRunner: Finishing task: attempt_local1133262668_0001_m_000000_0
2024-03-24 18:07:40,871 INFO mapred.LocalJobRunner: map task executor complete.
2024-03-24 18:07:40,935 INFO mapreduce.Job:  map 100% reduce 0%
2024-03-24 18:07:40,935 INFO mapreduce.Job: Job job_local1133262668_0001 completed successfully
2024-03-24 18:07:40,940 INFO mapreduce.Job: Counters: 22
        File System Counters
                FILE: Number of bytes read=316597
                FILE: Number of bytes written=866543
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=0
                HDFS: Number of bytes written=10737418200
                HDFS: Number of read operations=5
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=3
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=107374182
                Map output records=107374182
                Input split bytes=83
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=173
                Total committed heap usage (bytes)=297271296
        org.apache.hadoop.examples.terasort.TeraGen$Counters
                CHECKSUM=230593859918397906
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=10737418200
```

- 생성된 파일 확인

```
oem@ubuntu-server:~$ hdfs dfs -ls /teragen
Found 2 items
-rw-r--r--   1 oem supergroup          0 2024-03-24 18:07 /teragen/_SUCCESS
-rw-r--r--   1 oem supergroup 10737418200 2024-03-24 18:07 /teragen/part-m-00000
```

하둡 예제 (Wordcount)

- terasort

  hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.4.jar terasort /teragen/teragen.data /teragen/terasort.data

하둡 예제 (Wordcount)

```
2024-03-24 18:26:34,215 INFO mapreduce.Job: Counters: 36
        File System Counters
                FILE: Number of bytes read=485799799745
                FILE: Number of bytes written=938092152454
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=446412866700
                HDFS: Number of bytes written=10737418200
                HDFS: Number of read operations=8670
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=164
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=107374182
                Map output records=107374182
                Map output bytes=10952166564
                Map output materialized bytes=11166915408
                Input split bytes=9600
                Combine input records=0
                Combine output records=0
                Reduce input groups=107374182
                Reduce shuffle bytes=11166915408
                Reduce input records=107374182
                Reduce output records=107374182
                Spilled Records=322122546
                Shuffled Maps =80
                Failed Shuffles=0
                Merged Map outputs=80
                GC time elapsed (ms)=2993
                Total committed heap usage (bytes)=147702939648
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=10737418200
        File Output Format Counters
                Bytes Written=10737418200
2024-03-24 18:26:34,215 INFO terasort.TeraSort: done

real    5m30.607s
user    5m15.090s
sys     0m44.482s
```

DKU

DB Lab
Dankook University
데이타베이스 연구실

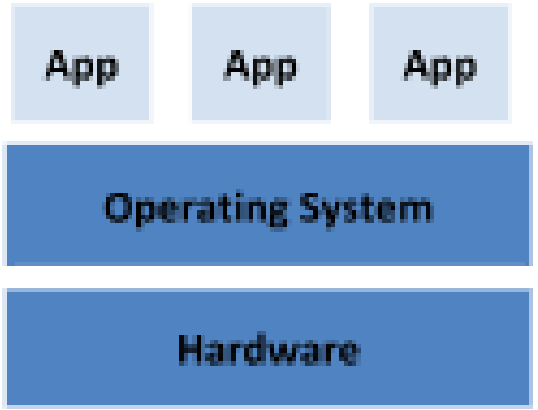
쿠버네티스란?

- 컨테이너를 쉽고 빠르게 배포, 확장 및 관리를 자동화해주는 오픈소스 플랫폼
- 명칭은 키잡이(helmsman)나 파일럿을 뜻하는 그리스어에서 유래
- K8s라는 표기는 "K"와 "s" 사이 8글자를 나타내는 약식 표기
- 구글이 2014년 쿠버네티스 프로젝트를 오픈소스화 함
- 관리자가 서버를 배포할 때 원하는 상태를 선언하는 방식 사용(Desired State)

쿠버네티스란?

kubernetes

- 전통적인 배포 시대(Traditional Deployment)
  - 초기 조직은 애플리케이션을 물리 서버에서 실행
  - 리소스 할당 문제 발생
  - 여러 물리 서버에서 각 애플리케이션을 실행
  - 리소스가 충분히 활용되지 않음.
  - 물리 서버를 유지하는 데에 높은 비용

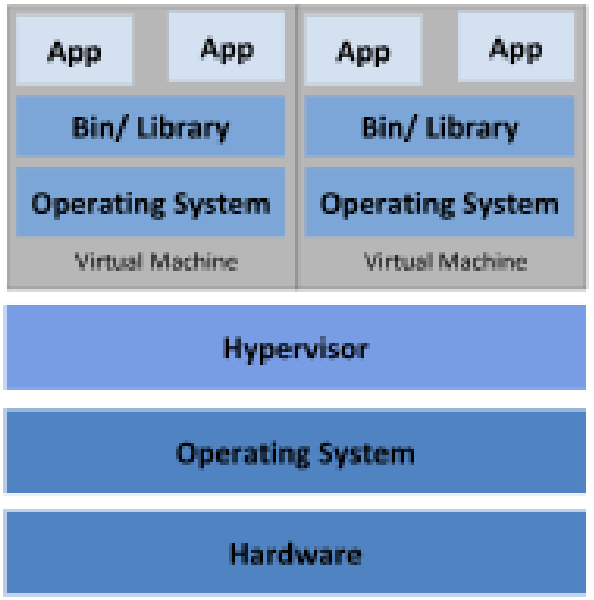


Traditional Deployment

쿠버네티스란?

kubernetes

- 가상화된 배포 시대(Virtualized Deployment)
  - 전통적인 배포 시대의 해결책으로 가상화 도입
  - 단일 물리서버의 CPU에서 여러 가상 머신 실행
  - 가상화를 사용하면 VM간에 격리로 보안 제공
  - 물리서버에서 리소스를 효율적으로 사용

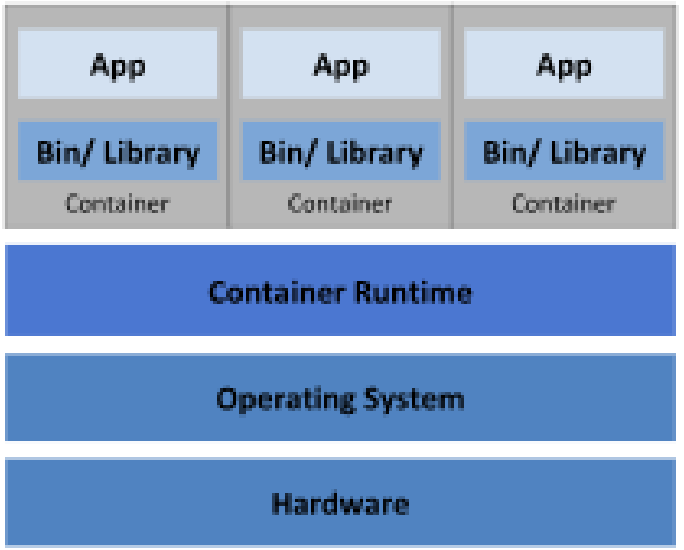


Virtualized Deployment

쿠버네티스란?

kubernetes

- 컨테이너 개발 시대(Container Deployment)
  - VM과 유사하지만 격리 속성 완화
  - 애플리케이션 간 OS 공유
  - VM이미지를 사용하는 것보다 컨테이너 이미지 생성이 쉽고 효율적
  - 클라우드 및 OS간 이식성(Ubuntu, RHEL, 퍼블릭 클라우드)



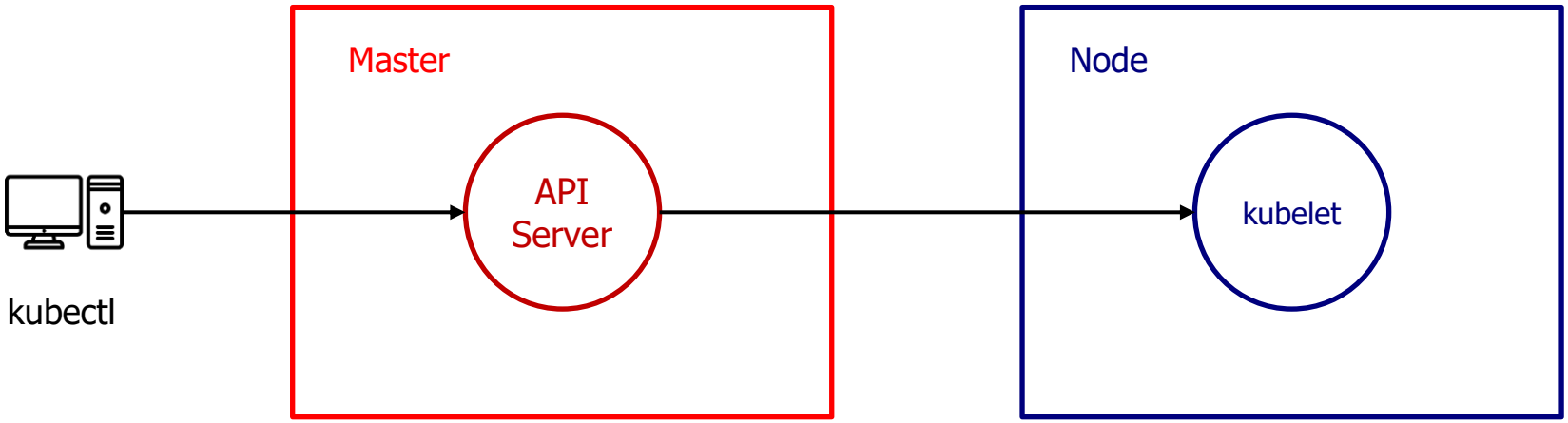
Container Deployment

쿠버네티스란?

- 컨테이너 오케스트레이션
  - 컨테이너 기반 애플리케이션 배포 관리, 제어 및 모니터링, 스케일링, 네트워킹 관리 도구
- 컨테이너 오케스트레이션 종류
  - 도커 스웜(Docker Swarm), 아파치 메소스(Apache Mesos), 노마드(Normad)
  - 쿠버네티스가 컨테이너 기반 인프라 시장에서 사실상 표준

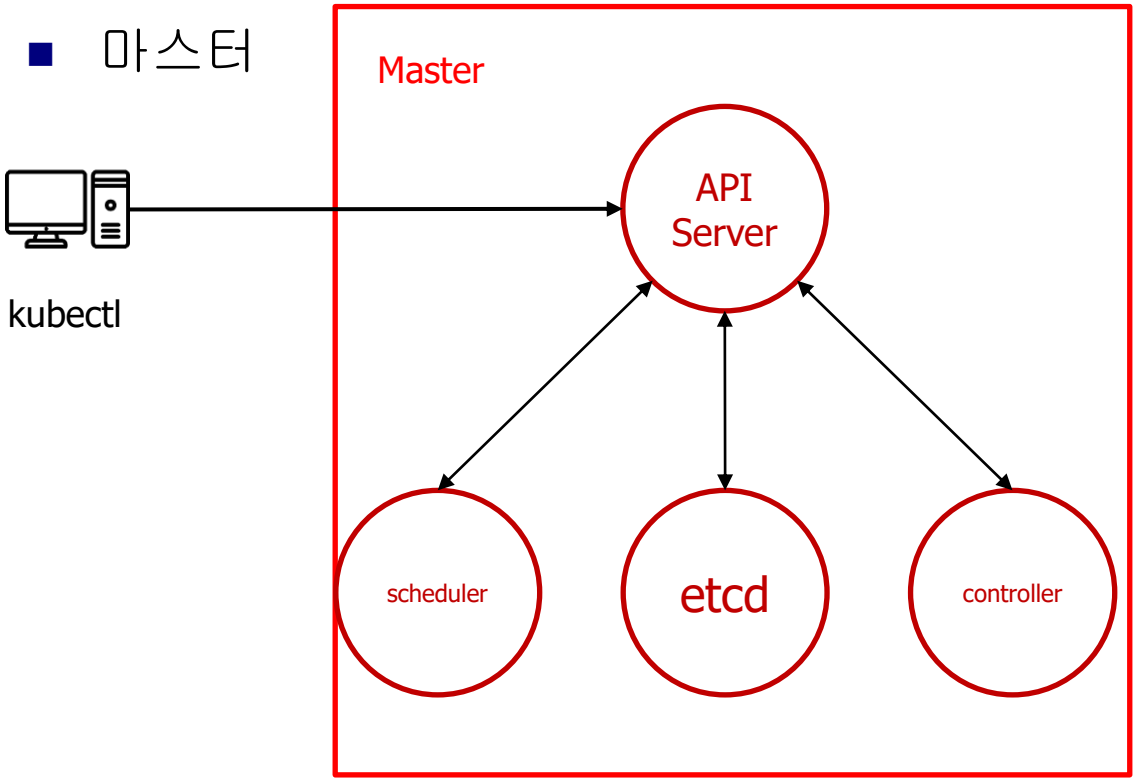
쿠버네티스 구조

- 마스터 - 노드 구조



쿠버네티스 구조

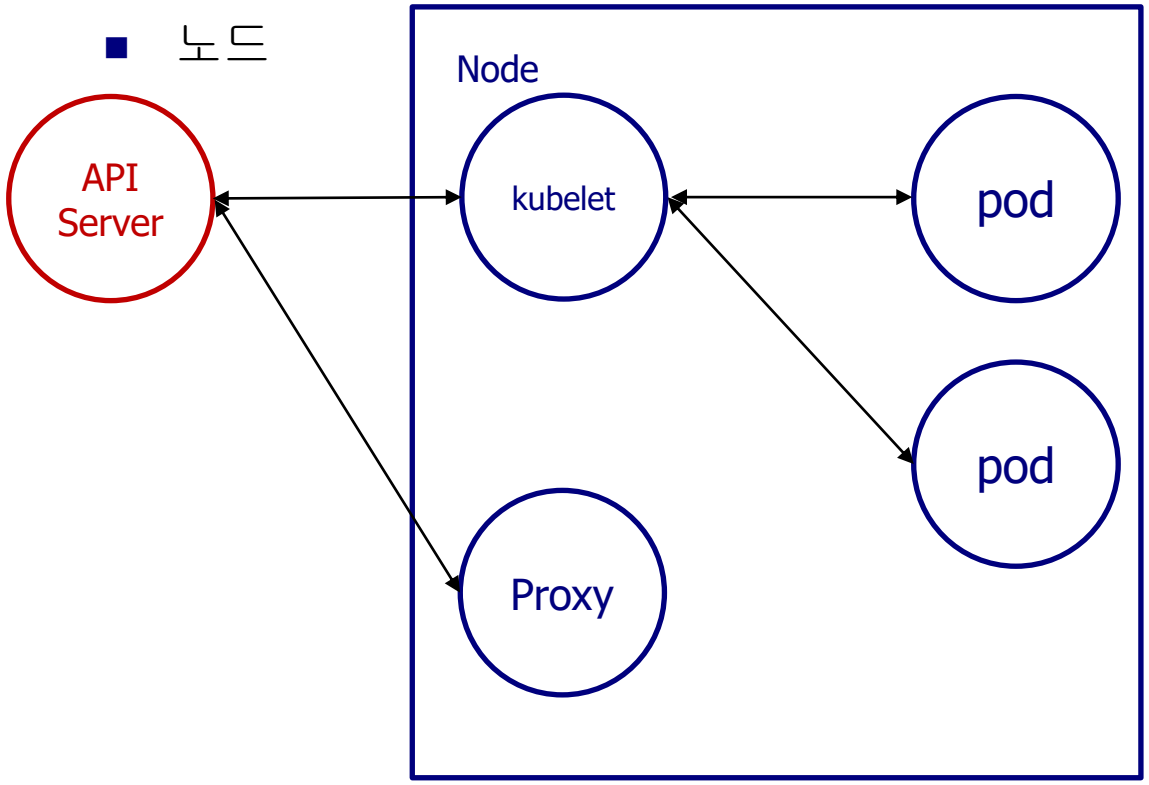
- 마스터



- 핵심 컴포넌트
- Kubectl
  - 마스터의 API Server 통신
- API Server
  - 관리자 요청 및 내부 모듈과 통신
  - etcd와 유일하게 통신
- etcd
  - 모든 상태와 데이터를 저장
  - Key-Value 형태
- Scheduler
  - 새로 생성된 pod을 감지
- controller
  - 다양한 컨트롤러 존재
  - 복제, 노드, 엔드포인트 등

쿠버네티스 구조

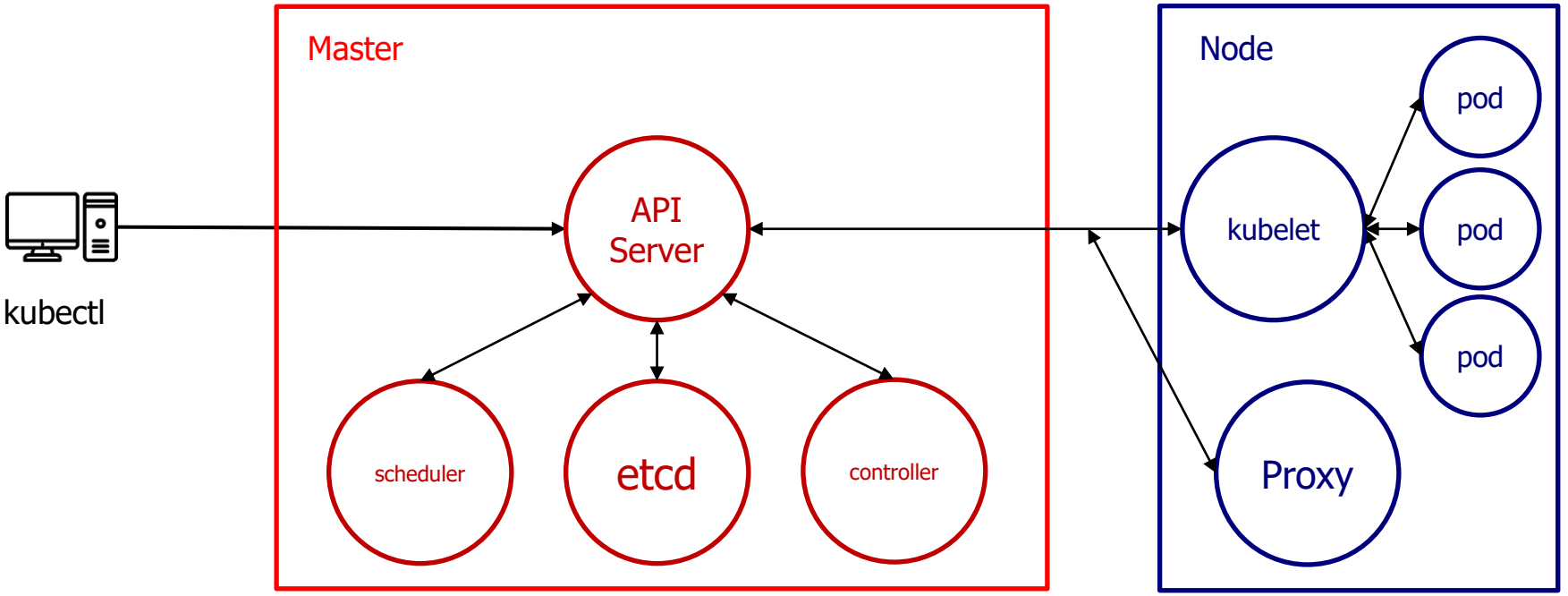
- 노드



- 핵심 컴포넌트
- kubelet
  - 각 노드에서 실행
  - pod 실행, 중지 및 상태 체크
  - CRI(Container Runtime Interface)
  - - Docker, Containerd, CRI-O…
- proxy
  - 네트워크 프록시와 부하 분산 역할
  - 내/외부 통신 설정 등

쿠버네티스 구조

- 쿠버네티스 프로세스



minikube Install

- minikube
  - 쿠버네티스 클러스터를 설치하기 위해선 3대의 마스터 서버와 n개의 노드 서버가 필요.
  - mac, Linux, Windows에서 K8s 클러스터를 빠르게 설정해주는 도구
  - minikube를 이용하면 로컬에서 K8s 클러스터를 만들 수 있다.

minikube Install

- minikube

  curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

  sudo install minikube-linux-amd64 /usr/local/bin/minikube

  sudo rm minikube-linux-amd64

minikube Install

- minikube

  minikube start

```
spark@ubuntu-VirtualBox:~$ minikube start
* minikube v1.33.0 on Ubuntu 22.04 (vbox/amd64)
* Automatically selected the docker driver. Other choices: ssh, none
* Using Docker driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.43 ...
* Downloading Kubernetes v1.30.0 preload ...
```

```
spark@ubuntu-VirtualBox:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

minikube Install

- kubectl

curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl

```
spark@ubuntu-VirtualBox:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.i
nux/amd64/kubectl"

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    523      0 --:--:-- --:--:-- --:--:--   524
100 49.0M  100 49.0M    0     0  10.3M      0  0:00:04  0:00:04 --:--:-- 11.2M
```

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
echo "$(cat kubectl.sha256) kubectl" | sha256sum --check

```
spark@ubuntu-VirtualBox:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.i
nux/amd64/kubectl.sha256"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    560      0 --:--:-- --:--:-- --:--:--   560
100    64  100    64    0     0    202      0 --:--:-- --:--:-- --:--:--   202
spark@ubuntu-VirtualBox:~$ echo "$(cat kubectl.sha256)  kubectl" | sha256sum --check
kubectl: OK
```

minikube Install

- kubectl

  sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

  chmod +x kubectl
  mkdir -p ~/.local/bin
  mv ./kubectl ~/.local/bin/kubectl

  kubectl version --client

```
spark@ubuntu-VirtualBox:~$ kubectl version --client
Client Version: v1.30.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
```

minikube Install

- minikube 및 kubectl 명령어

minikube start : cluster 실행
minikube delete : cluster 실행
minikube stop / pause : 정지 / 일시정지
minikube ip : 노드의 ip 확인
minikube ssh : node 접속

kubectl cluster-info : cluster 설정확인
kubectl delete pod pod명 --grace-period=0 --force :pod강제 삭제
kubectl get events : event 모니터링
kubectl describe pods/{pod명} or nodes/{node명} : 상세 보기

쿠버네티스 오브젝트

- 파드(Pod)
  - 쿠버네티스에서 생성하고 관리할 수 있는 배포 가능한 가장 작은 컴퓨팅 단위
  - 하나 이상의 컨테이너 그룹
  - Pod는 스토리지 및 네트워크를 공유하고, 구동하는 방식에 대한 명세를 갖는다.
  - 컨테이너와 비슷한 개념이지만 완전히 같은 개념은 아님

쿠버네티스 오브젝트

- 파드 생성
- vim 편집기로 yaml작성

vi simple-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

kubectl run
- CLI로 pod를 생성

kubectl create
- yaml으로 pod 생성
- 동일한 pod가 있을 경우 에러 발생

kubectl apply
- yaml으로 pod 생성
- 동일한 pod가 없으면 새로운 pod 생성
- 동일한 pod가 있으면 기존 config와 비교해서 수정된 부분 업데이트

kubectl apply -f simple-pod.yaml #파드 생성
kubectl get po #생성된 파드 확인
kubectl delete -f simple-pod.yaml

```
spark@ubuntu-VirtualBox:~$ kubectl apply -f simple-pod.yaml
pod/nginx created
spark@ubuntu-VirtualBox:~$ kubectl get po nginx
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          14s
spark@ubuntu-VirtualBox:~$ kubectl delete -f simple-pod.yaml
pod "nginx" deleted
```

쿠버네티스 오브젝트

- 네임스페이스(Namespace)
  - 쿠버네티스에서 사용되는 리소스들을 구분해 관리하는 그룹
  - Pod, Service 등은 네임스페이스별로 생성 및 관리 가능, 접근 권한도 다르게 설정 가능
  - 네임스페이스는 여러 개의 팀이나 프로젝트에 걸쳐 많은 사용자가 있는 환경에서 사용하도록 만들어짐
  - 쿠버네티스는 defalut, kube-node-lease, kube-public, kube-system이라는 네 개의 네임스페이스를 갖고 있다.

쿠버네티스 오브젝트

- 네임스페이스 조회
  kubectl get namespace or ns

```
spark@ubuntu-VirtualBox:~$ kubectl get ns
NAME              STATUS   AGE
default           Active   15h
kube-node-lease   Active   15h
kube-public       Active   15h
kube-system       Active   15h
```

- 새 네임스페이스 생성1

vi my-namespace.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: newns1
```

- kubectl create -f my-namespace.yaml

- 아래 명령어로 네임스페이스 생성 가능
  kubectl create namespace | ns newns2

  kubectl get ns

```
spark@ubuntu-VirtualBox:~$ kubectl create -f my-namespace.yaml
namespace/newns1 created
spark@ubuntu-VirtualBox:~$ kubectl create namespace newns2
namespace/newns2 created
spark@ubuntu-VirtualBox:~$ kubectl get ns
NAME              STATUS   AGE
default           Active   15h
kube-node-lease   Active   15h
kube-public       Active   15h
kube-system       Active   15h
newns1            Active   18s
newns2            Active   5s
```

kubectl delete ns newns2 #네임스페이스 삭제

쿠버네티스 오브젝트

- 서비스(Service)
  - 파드는 언제든 다른 노드로 옮겨지거나 삭제될 수 있음.
  - 생성될 때마다 새로운 IP를 받게 되는데, 내/외부 통신을 유지하기 어려움
  - 쿠버네티스에서 실행되고 있는 파드를 네트워크에 노출시키는 가상의 컴포넌트
  - 내/외부의 애플리케이션과 연결 혹은 사용자와 연결될 수 있도록 고정 IP를 갖는 서비스를 이용해 통신 가능

쿠버네티스 오브젝트

- 서비스 타입
  - Cluster IP
    - 가장 기본이 되는 Service 타입
    - 클러스터 내부 통신만 가능, 외부 트래픽 불가능
  - NodePort
    - 클러스터 내/외부 통신이 가능
    - 외부 트래픽을 전달받을 수 있음
    - 노드의 포트를 사용
  - LoadBlancer
    - 기본적으로 외부에 존재하며, 클라우드 프로바이더와 함께 사용되어 외부 트래픽을 받음
  - ExternalName
    - 위 3가지와 전혀 다른 서비스 타입
    - 외부로 나가는 트래픽을 변환하기 위한 용도
    - 도메인 이름을 변환하여 연결해주는 역할

쿠버네티스 오브젝트

- 볼륨(Volume)
  - 쿠버네티스에서 파드를 생성하면 디렉토리를 임시로 사용함
  - 컨테이너가 삭제되면 파일 손실되는 문제 발생
  - 파드가 사라지더라도 사용할 수 있는 디렉터리는 볼륨 오브젝트를 이용해 생성
  - Docker의 볼륨과 비슷한 개념
  - 쿠버네티스 버전에 따라 사용 가능한 볼륨이 있음
  - 볼륨 종류
    - emptyDir : 일시적 데이터 저장, 파드가 삭제되면 스토리지도 삭제
    - localPath: 노드의 파일시스템을 파드로 마운트
    - nfs
    - awsEBS, gcePersistentDisk, azureDisk : 클라우드 전용 스토리지
    - PV(Persistent Volumes) : 영구볼륨, 포드가 종료되어도 데이터는 삭제되지 않음.
    - PVC(Persistent Volumes Claim) : 생성된 PV를 사용

쿠버네티스 오브젝트

- PV yaml 작성

demoPV.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: demo-pv
spec:
  capacity:
    storage: 100Mi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/pv/log"
  persistentVolumeReclaimPolicy: Retain
```

- capacity : 볼륨크기
- accessModes
  - ReadWriteOnce : 하나의 노드에서 RW 가능
  - ReadOnlyMany : 여러노드에서 R 가능
  - ReadWriteMany : 여러 노드에서 RW가능
- persistentVolumeReclaimPolicy
  - PV 종료 시 볼륨에 저장된 데이터 삭제 옵션
  - Retain : PVC가 삭제되어도 PV의 데이터 보존
  - Delete : PVC가 삭제되면 데이터 및 PV도 삭제
  - Recycle : PVC가 삭제되면 데이터만 삭제

쿠버네티스 오브젝트

- PVC yaml 작성

  demoPVC.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: demo-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 50Mi
```

- capacity : 볼륨크기
- accessModes
  - 사용하는 PV와 동일한 옵션을 사용해야함
- requests
  - 사용을 원하는 스토리지 요구 명시
  - storage : 사용하고자 하는 최소한의 크기

쿠버네티스 오브젝트

- PV, PVC 생성 및 조회

  kubectl create -f demoPV.yaml

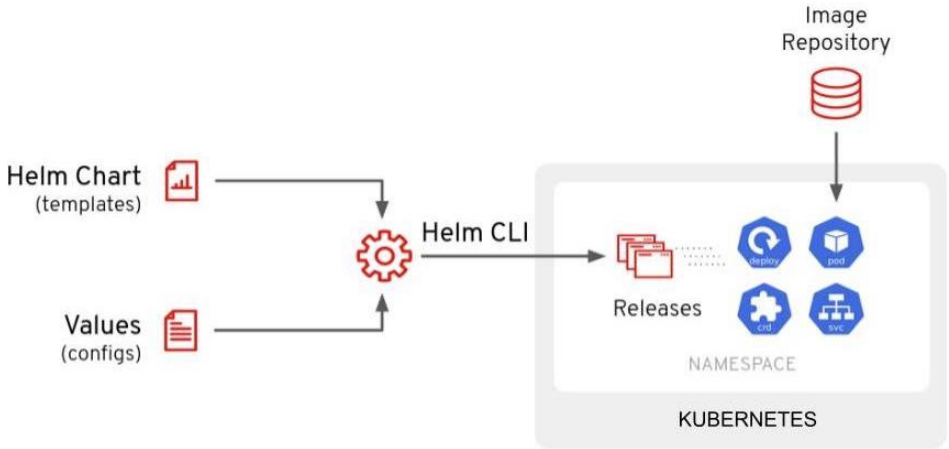
  kubectl create -f dempPVC.yaml

  kubectl get persistentvolume #pv

```
spark@ubuntu-VirtualBox:~$ kubectl create -f dempPV.yaml
persistentvolume/demo-pv created
spark@ubuntu-VirtualBox:~$ kubectl create -f dempPVC.yaml
persistentvolumeclaim/demo-pvc created
spark@ubuntu-VirtualBox:~$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
demo-pv                                    100Mi      RWX            Retain           Available
pvc-32e18ed2-a636-49e8-8a99-6c930e2a8257   2Gi        RWO            Delete           Bound       default/storage-prometheus-alertmanager-0
pvc-8cdb1b2e-13e0-4367-9f2d-03824f77c5eb   50Mi       RWX            Delete           Bound       default/demo-pvc
```

Helm

- Helm
  - K8s 애플리케이션을 위한 오픈소스 패키지 매니저
  - Helm 구성 요소
    - Chart : 애플리케이션을 배포하는데 사용되는 관련 Kubernetes YAML파일
    - Repository : 차트를 저장, 공유, 배포할 수 있는 곳
    - Release : Kubernetes 클러스터에 배포된 차트 특정 인스턴스



Helm

- Helm 설치

curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
chmod 700 get_helm.sh
./get_helm.sh

- 설치 확인

helm version

```
spark@ubuntu-VirtualBox:~$ helm version
version.BuildInfo{Version:"v3.14.4", GitCommit:"81c902a123462fd4052bc5e9aa9c513c4c8fc142", GitTreeState:"clean", GoVersion:"go1.21.9"}
```

Helm

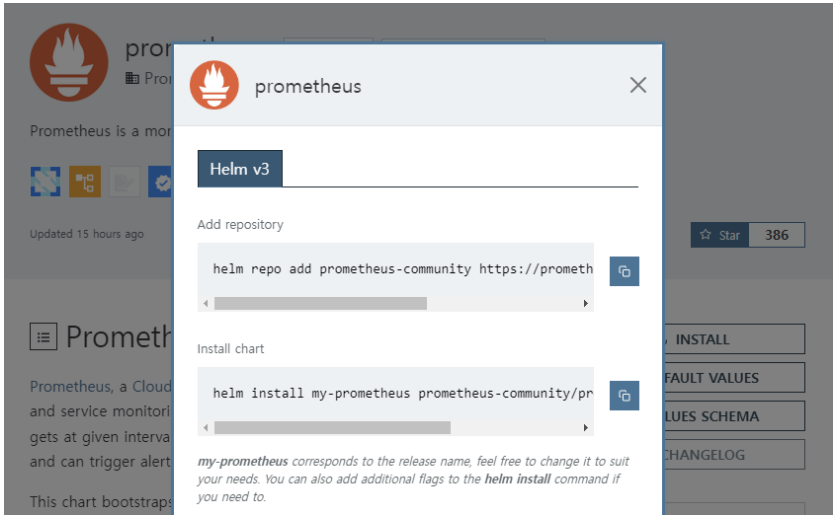
- Helm 명령어
  - helm search hub : 저장소에 있는 helm차트를 helm hub에서 검색
  - helm install chart명 : 특정 chart패키지 설치
  - helm show values chart명 : chart 옵션 설정가능한 values 조회
  - helm uninstall release명 : 릴리즈 삭제
  - helm repo list : 저장된 저장소 목록 조회
  - helm repo add {name} {url} : 저장소 저장
  - helm repo remove {name} : 저장소 삭제

Helm을 이용한 모니터링 시스템 만들기

- 프로메테우스(Prometheus)
  - SoundCloud에서 만든 오픈소스 모니터링 툴
  - Kubernetes환경에서 모니터링하기 원하는 리소스로부터 metric을 수집하고 해당 metric을 이용해 모니터링
- 그라파나(Grafana)
  - 오픈소스 시각화 분석 도구
  - 여러 데이터 소스에 대한 대시보드 템플릿 제공
  - 프로메테우스도 UI를 제공하지만, 기능이 빈약해 그라파나와 연동 사용

Helm을 이용한 모니터링 시스템 만들기

- 프로메테우스(Prometheus)
- https://artifacthub.io/ 에서 Prometheus 검색 - Install - Add repo
  helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
  helm repo update



Helm을 이용한 모니터링 시스템 만들기

- 프로메테우스(Prometheus)
- helm install prometheus prometheus-community/prometheus
- kubectl get all

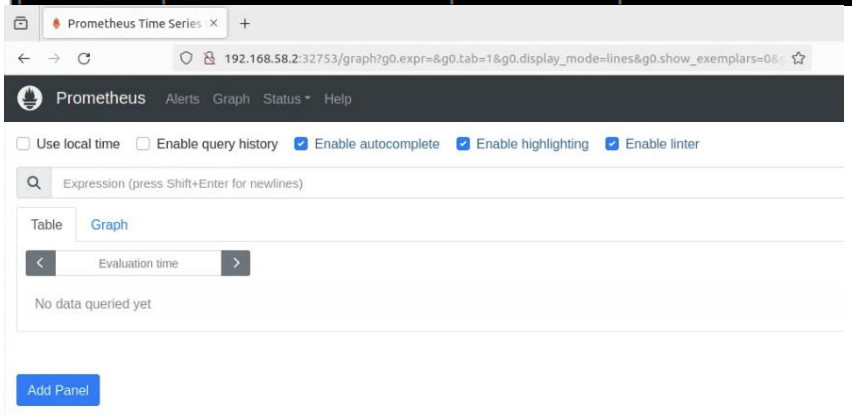
```
spark@ubuntu-VirtualBox:~/.cache/helm/repository$ kubectl get all
NAME                                                READY   STATUS    RESTARTS   AGE
pod/prometheus-alertmanager-0                       1/1     Running   0          5m54s
pod/prometheus-kube-state-metrics-76fc9c6f55-znbn5  1/1     Running   0          5m54s
pod/prometheus-prometheus-node-exporter-xmmwj       1/1     Running   0          5m54s
pod/prometheus-prometheus-pushgateway-7c758897fd-cttlb   1/1     Running   0          5m54s
pod/prometheus-server-55768b86b9-5rjqj              2/2     Running   0          5m54s

NAME                                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
service/kubernetes                            ClusterIP   10.96.0.1        <none>        443/TCP        83m
service/prometheus-alertmanager               ClusterIP   10.104.30.167    <none>        9093/TCP       5m54s
service/prometheus-alertmanager-headless      ClusterIP   None             <none>        9093/TCP       5m54s
service/prometheus-kube-state-metrics         ClusterIP   10.96.185.187    <none>        8080/TCP       5m54s
service/prometheus-prometheus-node-exporter   ClusterIP   10.106.132.146   <none>        9100/TCP       5m54s
service/prometheus-prometheus-pushgateway     ClusterIP   10.103.200.90    <none>        9091/TCP       5m54s
service/prometheus-server                     ClusterIP   10.109.183.118   <none>        80/TCP         5m54s
service/prometheus-server-ext                 NodePort    10.111.104.25    <none>        80:32753/TCP   101s

NAME                                                DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/prometheus-prometheus-node-exporter  1         1         1       1            1           kubernetes.io/os=linux   5m54s

NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/prometheus-kube-state-metrics       1/1     1            1           5m54s
deployment.apps/prometheus-prometheus-pushgateway   1/1     1            1           5m54s
deployment.apps/prometheus-server                   1/1     1            1           5m54s

NAME                                                DESIRED   CURRENT   READY   AGE
replicaset.apps/prometheus-kube-state-metrics-76fc9c6f55       1         1         1       5m54s
replicaset.apps/prometheus-prometheus-pushgateway-7c758897fd   1         1         1       5m54s
replicaset.apps/prometheus-server-55768b86b9                   1         1         1       5m54s
```

Helm을 이용한 모니터링 시스템 만들기

- 프로메테우스(Prometheus)
- 외부에서 접속(VM에서 접속)

kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-ext
minikube service prometheus-server-ext

```
spark@ubuntu-VirtualBox:~/.cache/helm/repository$ minikube service prometheus-server-ext
|-----------|-----------------------|-------------|---------------------------|
| NAMESPACE |         NAME          | TARGET PORT |            URL            |
|-----------|-----------------------|-------------|---------------------------|
| default   | prometheus-server-ext |          80 | http://192.168.58.2:32753 |
|-----------|-----------------------|-------------|---------------------------|
```

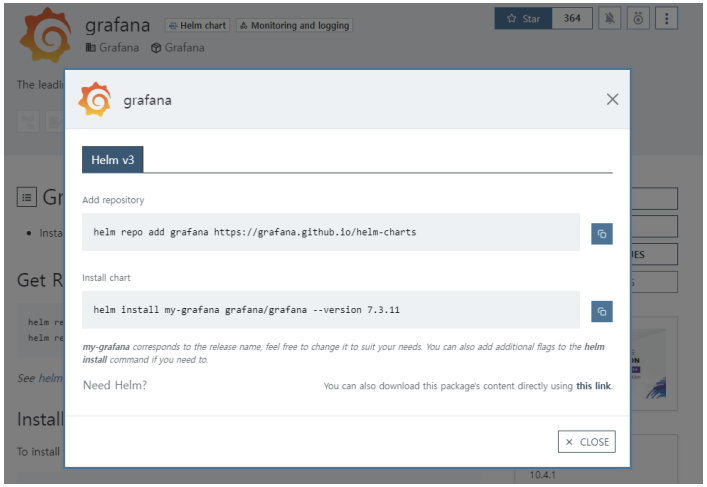


Helm을 이용한 모니터링 시스템 만들기

- 그라파나(Grafana)
- https://artifacthub.io/ 에서 Grafana 검색 - Install - Add repo

  helm repo add grafana https://grafana.github.io/helm-charts

  helm repo update



Helm을 이용한 모니터링 시스템 만들기

- 그라파나(Grafana)

helm install grafana grafana/grafana

kubectl get all

```
spark@ubuntu-VirtualBox:~/.cache/helm/repository$ kubectl get all
NAME                                                READY   STATUS    RESTARTS   AGE
pod/grafana-5ccd97668d-cxl9x                        1/1     Running   0          3m14s
pod/prometheus-alertmanager-0                       1/1     Running   0          17m
pod/prometheus-kube-state-metrics-76fc9c6f55-znbn5  1/1     Running   0          17m
pod/prometheus-prometheus-node-exporter-xmmwj       1/1     Running   0          17m
pod/prometheus-prometheus-pushgateway-7c758897fd-cttlb  1/1     Running   0          17m
pod/prometheus-server-55768b86b9-5rjqj              2/2     Running   0          17m

NAME                                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
service/grafana                               ClusterIP   10.106.237.143   <none>        80/TCP         3m14s
service/grafana-ext                           NodePort    10.98.8.196      <none>        80:31012/TCP   3m3s
service/kubernetes                            ClusterIP   10.96.0.1        <none>        443/TCP        94m
service/prometheus-alertmanager               ClusterIP   10.104.30.167    <none>        9093/TCP       17m
service/prometheus-alertmanager-headless      ClusterIP   None             <none>        9093/TCP       17m
service/prometheus-kube-state-metrics         ClusterIP   10.96.185.187    <none>        8080/TCP       17m
service/prometheus-prometheus-node-exporter   ClusterIP   10.106.132.146   <none>        9100/TCP       17m
service/prometheus-prometheus-pushgateway     ClusterIP   10.103.200.90    <none>        9091/TCP       17m
service/prometheus-server                     ClusterIP   10.109.183.118   <none>        80/TCP         17m
service/prometheus-server-ext                 NodePort    10.111.104.25    <none>        80:32753/TCP   12m

NAME                                                 DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/prometheus-prometheus-node-exporter   1         1         1       1            1           kubernetes.io/os=linux   17m

NAME                                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/grafana                        1/1     1            1           3m14s
deployment.apps/prometheus-kube-state-metrics  1/1     1            1           17m
deployment.apps/prometheus-prometheus-pushgateway  1/1     1            1           17m
deployment.apps/prometheus-server              1/1     1            1           17m
```

Helm을 이용한 모니터링 시스템 만들기

- 그라파나(Grafana)

kubectl get po

```
spark@ubuntu-VirtualBox:~/.cache/helm/repository$ kubectl get po
NAME                                             READY   STATUS    RESTARTS   AGE
grafana-5ccd97668d-cxl9x                         1/1     Running   0          23m
prometheus-alertmanager-0                        1/1     Running   0          37m
prometheus-kube-state-metrics-76fc9c6f55-znbn5   1/1     Running   0          37m
prometheus-prometheus-node-exporter-xmmwj        1/1     Running   0          37m
```

pod명 확인 후 패스워드 변경
kubectl exec --namespace default -it pod명 -- /bin/bash #pod 접속

grafana-cli admin reset-admin-password Tibero1! #Tibero1!는 패스워드

Helm을 이용한 모니터링 시스템 만들기

- 그라파나(Grafana)

kubectl expose service grafana --type=NodePort --target-port=3000 --name=grafana-ext

minikube service grafana-ext

```
spark@ubuntu-VirtualBox:~/.cache/helm/repository$ minikube service grafana-ext
|-----------|-------------|-------------|---------------------------|
| NAMESPACE |    NAME     | TARGET PORT |            URL            |
|-----------|-------------|-------------|---------------------------|
| default   | grafana-ext |          80 | http://192.168.58.2:31012 |
|-----------|-------------|-------------|---------------------------|
```

DKU

DB Lab Dankook University 데이타베이스 연구실

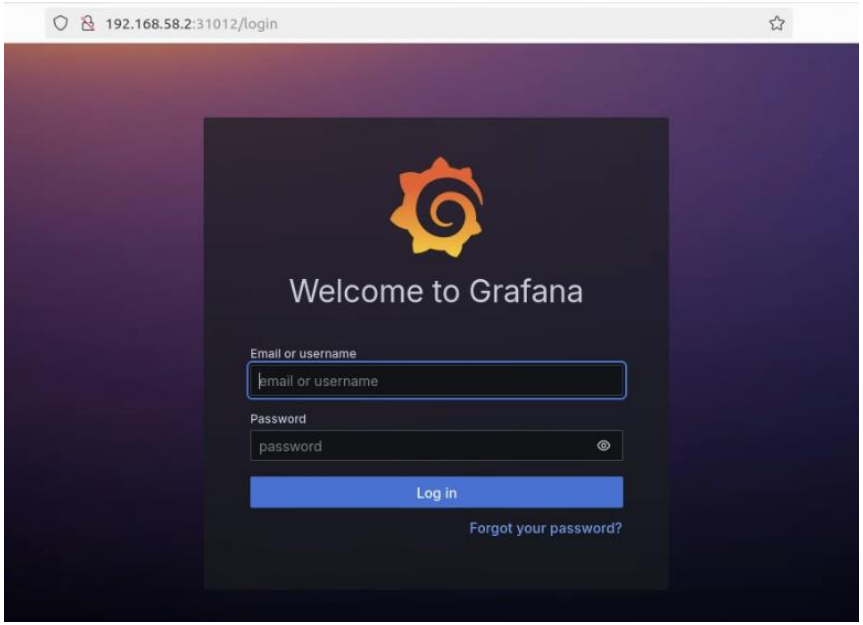
Helm을 이용한 모니터링 시스템 만들기

- 그라파나(Grafana)

192.168.58.2:31012

ID : admin

PW : 변경한 패스워드



Wordpress 만들기

- helm chart 구조
- <chart name> /

  Chart.yaml : 차트의 메타 데이터

  values.yaml : 패키지 사용자화

  templates/ : YAML 오브젝트 파일

- artifacthub.io 에서 wordpress 검색 후 repo 추가

  helm repo add bitnami https://charts.bitnami.com/bitnami

- 내부 저장소에서 검색

  helm search repo wordpress

```
ubuntu@ubuntu-VirtualBox:~$ helm search repo wordpress
NAME                    CHART VERSION   APP VERSION     DESCRIPTION
bitnami/wordpress       22.2.7          6.5.3           WordPress is the world's most popular blogging ...
bitnami/wordpress-intel 2.1.31          6.1.1           DEPRECATED WordPress for Intel is the most popu...
```

Wordpress 만들기

- wordpress chart 정보 확인
  - helm inspect values bitnami/wordpress

```
ubuntu@ubuntu-VirtualBox:~$ helm inspect values bitnami/wordpress
# Copyright Broadcom, Inc. All Rights Reserved.
# SPDX-License-Identifier: APACHE-2.0

## @section Global parameters
## Global Docker image parameters
## Please, note that this will override the image parameters, including dependencies, configured to use the global value
## Current available global Docker image parameters: imageRegistry, imagePullSecrets and storageClass
##

## @param global.imageRegistry Global Docker image registry
## @param global.imagePullSecrets Global Docker registry secret names as an array
## @param global.storageClass Global StorageClass for Persistent Volume(s)
##
global:
  imageRegistry: ""
  ## E.g.
  ## imagePullSecrets:
  ##   - myRegistryKeySecretName
  ##
  imagePullSecrets: []
  storageClass: ""
  ## Compatibility adaptations for Kubernetes platforms
  ##
  compatibility:
    ## Compatibility adaptations for Openshift
    ##
    openshift:
      ## @param global.compatibility.openshift.adaptSecurityContext Adapt the securityContext sections of the deployment to
ove runAsUser, runAsGroup and fsGroup and let the platform use their allowed default IDs. Possible values: auto (apply if th
he adaptation always), disabled (do not perform adaptation)
      ##
      adaptSecurityContext: auto
```

Wordpress 만들기

- wordpress 설치

  helm install my-wordpress bitnami/wordpress

  kubectl get all

```
ubuntu@ubuntu-VirtualBox:~$ kubectl get po
NAME                            READY   STATUS    RESTARTS   AGE
my-wordpress-cf9758478-wtvqf    1/1     Running   0          3m44s
my-wordpress-mariadb-0          1/1     Running   0          3m44s
ubuntu@ubuntu-VirtualBox:~$ kubectl get po
NAME                            READY   STATUS    RESTARTS   AGE
my-wordpress-cf9758478-wtvqf    1/1     Running   0          3m46s
my-wordpress-mariadb-0          1/1     Running   0          3m46s
ubuntu@ubuntu-VirtualBox:~$ kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/my-wordpress-cf9758478-wtvqf    1/1     Running   0          3m49s
pod/my-wordpress-mariadb-0          1/1     Running   0          3m49s

NAME                           TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
service/kubernetes             ClusterIP      10.96.0.1        <none>        443/TCP
service/my-wordpress           LoadBalancer   10.106.216.205   <pending>     80:30589/TCP,443:3273
service/my-wordpress-mariadb   ClusterIP      10.96.207.31     <none>        3306/TCP

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-wordpress   1/1     1            1           3m49s

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/my-wordpress-cf9758478    1         1         1       3m49s

NAME                                  READY   AGE
statefulset.apps/my-wordpress-mariadb   1/1     3m49s
```

Wordpress 만들기

- Pod 정보 확인
  - kubectl describe po my-wordpress-cf9758478-wtvqf #개인마다 Pod의 이름은 다름

```
Environment:
  BITNAMI_DEBUG:                  false
  ALLOW_EMPTY_PASSWORD:           yes
  WORDPRESS_SKIP_BOOTSTRAP:       no
  MARIADB_HOST:                   my-wordpress-mariadb
  MARIADB_PORT_NUMBER:            3306
  WORDPRESS_DATABASE_NAME:        bitnami_wordpress
  WORDPRESS_DATABASE_USER:        bn_wordpress
  WORDPRESS_DATABASE_PASSWORD:    <set to the key 'mariadb-password' in secret 'my-wordpres
  WORDPRESS_USERNAME:             user
  WORDPRESS_PASSWORD:             <set to the key 'wordpress-password' in secret 'my-wordpr
  WORDPRESS_EMAIL:                user@example.com
  WORDPRESS_FIRST_NAME:           FirstName
  WORDPRESS_LAST_NAME:            LastName
  WORDPRESS_HTACCESS_OVERRIDE_NONE:       no
  WORDPRESS_ENABLE_HTACCESS_PERSISTENCE:  no
  WORDPRESS_BLOG_NAME:            User's Blog!
  WORDPRESS_TABLE_PREFIX:         wp_
  WORDPRESS_SCHEME:               http
  WORDPRESS_EXTRA_WP_CONFIG_CONTENT:
```

Wordpress 만들기

- wordpress 삭제

  helm uninstall my-wordpress bitnami/wordpress

  kubectl get all

```
ubuntu@ubuntu-VirtualBox:~$ kubectl get po
NAME                            READY   STATUS    RESTARTS   AGE
my-wordpress-cf9758478-wtvqf    1/1     Running   0          3m44s
my-wordpress-mariadb-0          1/1     Running   0          3m44s
ubuntu@ubuntu-VirtualBox:~$ kubectl get po
NAME                            READY   STATUS    RESTARTS   AGE
my-wordpress-cf9758478-wtvqf    1/1     Running   0          3m46s
my-wordpress-mariadb-0          1/1     Running   0          3m46s
ubuntu@ubuntu-VirtualBox:~$ kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/my-wordpress-cf9758478-wtvqf    1/1     Running   0          3m49s
pod/my-wordpress-mariadb-0          1/1     Running   0          3m49s

NAME                            TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
service/kubernetes              ClusterIP      10.96.0.1        <none>        443/TCP
service/my-wordpress            LoadBalancer   10.106.216.205   <pending>     80:30589/TCP,443:3273
service/my-wordpress-mariadb    ClusterIP      10.96.207.31     <none>        3306/TCP

NAME                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-wordpress    1/1     1            1           3m49s

NAME                                       DESIRED   CURRENT   READY   AGE
replicaset.apps/my-wordpress-cf9758478     1         1         1       3m49s

NAME                                  READY   AGE
statefulset.apps/my-wordpress-mariadb   1/1     3m49s
```

Wordpress 만들기

- wordpress 차트 커스터마이징

helm install my-wordpress bitnami/wordpress --set mariadb.auth.username=hello_wordpress

kubectl get po

```
ubuntu@ubuntu-VirtualBox:~$ kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
custmz-wordpress-7c675b5474-kwz9r   1/1     Running   0          92s
custmz-wordpress-mariadb-0          1/1     Running   0          92s
```

kubectl describe po custmz-wordpress-7c675b5474-kwz9r

```
Environment:
  BITNAMI_DEBUG:                      false
  ALLOW_EMPTY_PASSWORD:               yes
  WORDPRESS_SKIP_BOOTSTRAP:           no
  MARIADB_HOST:                       custmz-wordpress-mariadb
  MARIADB_PORT_NUMBER:                3306
  WORDPRESS_DATABASE_NAME:            bitnami_wordpress
  WORDPRESS_DATABASE_USER:            hello_wordpress
  WORDPRESS_DATABASE_PASSWORD:        <set to the key 'mariadb-password' in secret 'custmz-wordpress-mariadb'>  Option
  WORDPRESS_USERNAME:                 user
  WORDPRESS_PASSWORD:                 <set to the key 'wordpress-password' in secret 'custmz-wordpress'>  Optional: fa
  WORDPRESS_EMAIL:                    user@example.com
  WORDPRESS_FIRST_NAME:               FirstName
  WORDPRESS_LAST_NAME:                LastName
  WORDPRESS_HTACCESS_OVERRIDE_NONE:   no
  WORDPRESS_ENABLE_HTACCESS_PERSISTENCE:  no
```

Wordpress 만들기

- wordpress 차트 커스터마이징

helm install custmz-wordpress bitnami/wordpress -f valuesF.yaml

kubectl describe po custmz-wordpress-6fc7c49494-bbv56

```
BITNAMI_DEBUG:                  false
ALLOW_EMPTY_PASSWORD:           yes
WORDPRESS_SKIP_BOOTSTRAP:       no
MARIADB_HOST:                   custmz-wordpress-mariadb
MARIADB_PORT_NUMBER:            3306
WORDPRESS_DATABASE_NAME:        hello_wordpress
WORDPRESS_DATABASE_USER:        bn_wordpress
WORDPRESS_DATABASE_PASSWORD:    <set to the key 'mariadb-p
```

vi valuesF.yaml

```
mariadb:
  auth:
    database: hello_wordpress
```