



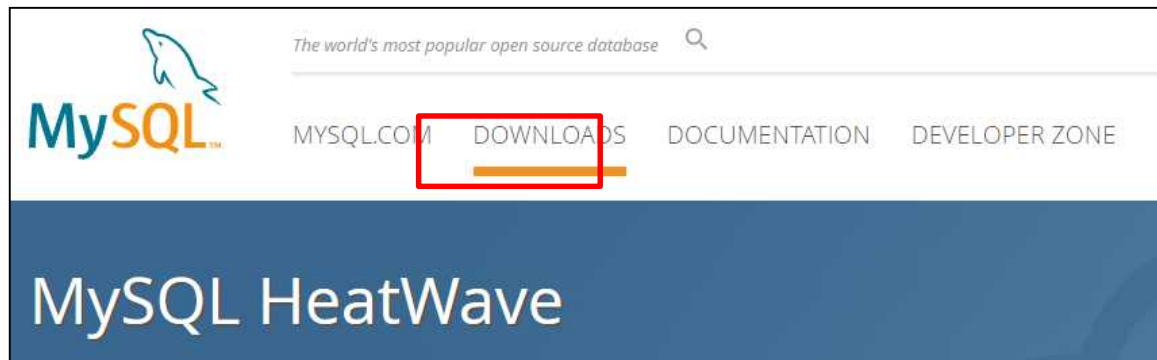
TABA_DB/SQL실습1

What is MySQL?



- MySQL은 가장 많이 사용되고 있는 관계형 데이터베이스 관리 시스템(RDBMS)
- MySQL은 오픈소스이며, 다중 사용자 및 다중 스레드 지원
- C, JAVA, PHP, R 등 다양한 프로그래밍 언어를 위한 API를 제공
- MySQL은 리눅스, 윈도우 등 다양한 운영체제에서 사용 가능하며, PHP와 웹 개발에 자주 사용(Apache, PHP, MySQL)

Install MySQL



Install MySQL

MySQL Community Downloads

- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository
- MySQL Community Server
- MySQL Cluster
- MySQL Router
- MySQL Shell
- MySQL Operator
- MySQL Workbench
- MySQL Installer for Windows
- MySQL for Visual Studio
- C API (libmysqlclient)
- Connector/C++
- Connector/J
- Connector/NET
- Connector/Node.js
- Connector/ODBC
- Connector/Python
- MySQL Native Driver for PHP
- MySQL Benchmark Tool
- Time zone description tables
- Download Archives

Install MySQL

Repository Setup Packages

Red Hat Enterprise Linux 9 / Oracle Linux 9 (Architecture Independent), RPM Package (mysql80-community-release-el9-1.noarch.rpm)	10.3K	Download
Red Hat Enterprise Linux 8 / Oracle Linux 8 (Architecture Independent), RPM Package (mysql80-community-release-el8-4.noarch.rpm)	14.1K	Download
Red Hat Enterprise Linux 7 / Oracle Linux 7 (Architecture Independent), RPM Package (mysql80-community-release-el7-7.noarch.rpm)	10.9K	Download
Red Hat Enterprise Linux 6 / Oracle Linux 6 (Architecture Independent), RPM Package (mysql80-community-release-el6-7.noarch.rpm)	10.9K	Download

MD5: d07a0c6a95783c43d0c520c245cf18e0

MD5: 72a4647a99c7ac1e3a8efb874b1d4af4

MD5: 659400f9842fffb8d64ae0b650f081b9

MD5: 31233d8fbbcdh2700a1723736b21ec67

Install MySQL



[No thanks, just start my download.](#)

마우스 우측 클릭 - 링크 주소 복사

Install MySQL

- **MySQL repository 설치**
- `sudo yum install -y https://dev.mysql.com/get/mysql84-community-release-el9-1.noarch.rpm`

```
[root@localhost shjeon]# yum install -y https://dev.mysql.com/get/mysql80-community-release-el8-4.noarch.rpm
CentOS Stream 9 - BaseOS                               112 kB/s | 5.9 MB   00:53
CentOS Stream 9 - AppStream                             5.7 MB/s | 15 MB   00:02
CentOS Stream 9 - Extras packages                       6.5 kB/s | 8.5 kB   00:01
Last metadata expiration check: 0:00:01 ago on Wed 07 Sep 2022 11:04:58 AM KST.
mysql80-community-release-el8-4.noarch.rpm             33 kB/s | 14 kB    00:00
Dependencies resolved.

=====
Package                                Architecture    Version          Repository        Size
=====
Installing:
mysql80-community-release              noarch          el8-4            @commandline      14 k
Transaction Summary
=====
Install 1 Package
```

Install MySQL

- MySQL repository 확인
- `yum repolist enabled | grep "mysql.*"`

```
[ec2-user@ip-172-31-6-212 ~]$ yum repolist enabled | grep "mysql.*"  
mysql-8.4-lts-community           MySQL 8.4 LTS Community Server  
mysql-connectors-community        MySQL Connectors Community  
mysql-tools-8.4-lts-community     MySQL Tools 8.4 LTS Community
```


Install MySQL

- MySQL 설치
- `sudo yum install -y mysql-server`

```
[ec2-user@ip-172-31-6-212 ~]$ sudo yum install -y mysql-server
MySQL 8.4 LTS Community Server                                1.4 MB/s
MySQL Connectors Community                                   304 kB/s
MySQL Tools 8.4 LTS Community                                760 kB/s
Dependencies resolved.
=====
Package                                Architecture          Version                Repository
=====
Installing:
mysql-community-server                  x86_64                 8.4.2-1.el9            mysql-8.4-lts-community
Installing dependencies:
libaio                                  x86_64                 0.3.111-13.el9         baseos
libtirpc                                x86_64                 1.3.3-9.el9            baseos
```

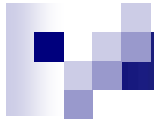
- MySQL 설치 확인
- `mysqld -V`

```
[ec2-user@ip-172-31-6-212 ~]$ mysqld -V
/usr/sbin/mysqld  Ver 8.4.2 for Linux on x86_64 (MySQL Community Server - GPL)
```

Install MySQL

- MySQL 서버 실행 및 상태 확인
- `sudo systemctl enable mysqld && sudo systemctl start mysqld && sudo systemctl status mysqld`

```
[ec2-user@ip-172-31-6-212 ~]$ sudo systemctl status mysqld
• mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service; enabled; preset: disabled)
   Active: active (running) since Thu 2024-09-12 02:40:31 UTC; 24s ago
     Docs: man:mysqld(8)
           http://dev.mysql.com/doc/refman/en/using-systemd.html
   Process: 68400 ExecStartPre=/usr/bin/mysqld_pre_systemd (code=exited, status=0/SUCCESS)
  Main PID: 68465 (mysqld)
    Status: "Server is operational"
     Tasks: 35 (limit: 22622)
    Memory: 442.0M
       CPU: 2.545s
    CGroup: /system.slice/mysqld.service
            └─68465 /usr/sbin/mysqld
```



MySQL 기본

- MySQL 8.0 버전은 서버 설치과정에서 임시 비밀번호 생성
- 아래 명령어로 임시 비밀번호 확인
- `sudo grep 'temporary password' /var/log/mysqld.log`

```
d' /var/log/mysqld.log  
A temporary password is generated for root@localhost: HAcEFTGLm8>1
```

MySQL 기본

- MySQL 접속
- `mysql -u root -p`
- 임시 비밀번호가 없는 경우 그냥 **mysql** 입력하면 됨.

```
[root@localhost shjeon]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.30

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

MySQL 기본

- Root 유저의 비밀번호 변경
- 비밀번호에는 대문자, 소문자, 숫자, 기호 모두 조합
- ALTER USER 'root'@'localhost' IDENTIFIED BY 'Tibero1!';

```
[root@centos8 vagrant]# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.26 Source distribution

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> alter user 'root'@'localhost' IDENTIFIED BY 'TABa1';
Query OK, 0 rows affected (0.01 sec)
```

MySQL 기본

- 사용자 정보 확인
- `select user, host, authentication_string FROM mysql.user;`

```
mysql> select user, host, authentication_string from mysql.user;
```

user	host	authentication_string
mysql.infoschema	localhost	\$A\$005\$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED
mysql.session	localhost	\$A\$005\$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED
mysql.sys	localhost	\$A\$005\$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED
root	localhost	*9058A56D277B8F43DADC2E73AB1BEE8CDE7F9F37

```
4 rows in set (0.00 sec)
```

MySQL 기본

- MySQL 내 데이터베이스 조회하기
- `show databases;`
- 데이터베이스 사용하기
- `use database명`

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.01 sec)

mysql> use mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
```

MySQL 기본

- Database 내 테이블 확인하기
- `show tables;`

```
mysql> show tables;
```

```
+-----+  
| Tables_in_mysql |  
+-----+  
| columns_priv    |  
| component       |  
| db              |  
| default_roles   |  
| engine_cost     |  
| func            |  
| general_log     |  
| global_grants   |  
| gtid_executed   |  
| help_category   |  
| help_keyword    |  
| help_relation   |  
| help_topic      |
```


MySQL 기본

- Database 만들기
- create database 데이터베이스명;
- show databases;

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| testdb |
+-----+
5 rows in set (0.01 sec)
```

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.

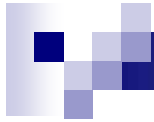
데이터 정의어(Data Definition Language)

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 SYNONYM	동의어를 생성한다.
	DROP SYNONYM	동의어를 제거한다.
사용자	CREATE USER	사용자를 생성한다.
	ALTER USER	사용자를 변경한다.
	DROP USER	사용자를 제거한다.



데이터 정의어(Data Definition Language)

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)

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

MySQL에서 제공하는 데이터 타입

구분	데이터 타입
문자형	CHAR, VARCHAR, TEXT, MEDIUMTEXT, LONGTEXT, JASON
숫자형	TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT, FLOAT, DECIMAL, DOUBLE
날짜형	DATE, TIME, DATETIME, TIMESTAMP, YEAR
이진 데이터 타입	BINARY, VARBINARY, TINYBLOB, BLOB,

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

Create Tables to Store Data

Table Instance Chart

Table Name: EMPLOYEE

Column Name	Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Key Type				PK					FK	
Nulls / Unique	NN		NN	NN						NN
FK Ref Table									EMPLOYEE	
FK Ref Column									Ssn	
Data Type	VARCHAR	CHAR	VARCHAR	CHAR	DATE	VARCHAR	CHAR	DECIMAL	CHAR	INT
Maximum Length	15		15	9		30		10,2	9	
Sample Data	John	B	Smith	123456789	1965-01-09	731 Foundren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5

Create Tables to Store Data

Syntax

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

where	<i>table_name</i>	테이블 이름
	<i>column_name</i>	열 이름
	<i>datatype</i>	데이터 타입
	<i>table_constraint</i>	제약 조건

Create Tables to Store Data

```
CREATE TABLE EMPLOYEE
( Fname      VARCHAR(10) NOT NULL,
  Minit      CHAR,
  Lname      VARCHAR(20)  NOT NULL,
  Ssn        CHAR(9)     NOT NULL,
  Bdate      DATE,
  Address    VARCHAR(30),
  Sex        CHAR(1),
  Salary     DECIMAL(5),
  Super_ssn  CHAR(9),
  Dno        INT          NOT NULL,
  PRIMARY KEY (Ssn),
  FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE (ssn));
```

Create Tables to Store Data

```
INSERT INTO EMPLOYEE
```

```
VALUES ('James','E','Borg',888665555,'1937-11-10','450 Stone, Houston TX','M',55000,null,1),  
('Franklin','T','Wong',333445555,'1965-12-08','638 Voss, Houston TX','M',40000,888665555,5),  
('John','B','Smith',123456789,'1965-01-09','731 Fondren, Houston TX','M',30000,333445555,5),  
('Ramesh','K','Narayan',666884444,'1962-09-15','975 Fire Oak, Humble TX','M',38000,333445555,5),  
('Jennifer','S','Wallace',987654321,'1941-06-20','291 Berry, Bellaire TX','F',43000,888665555,4),  
('Joyce','A','English',453453453,'1972-07-31','5631 Rice, Houston TX','F',25000,333445555,5),  
('Alicia','J','Zelaya',999887777,'1968-01-19','3321 Castle, Spring TX','F',25000,987654321,4),  
('Ahmad','V','Jabbar',987987987,'1969-03-29','980 Dallas, Houston TX','M',25000,987654321,4);
```

Create Tables to Store Data

EXAMPLE

Table Name: DEPARTMENT

Column Name	Dname	Dnumber	Mgr_ssn	Mgr_start_date
Key Type		PK	FK	
Nulls/Unique	NN,U	NN	NN	
FK Ref Table			EMPLOYEE	
Data Type	VARCHAR	INT	CHAR	DATE
Maximum Length	15		9	
Sample data	Research	5	333445555	1988-05-22

```
CREATE TABLE DEPARTMENT
( Dname      VARCHAR(15)  NOT NULL,
  Dnumber    INT          NOT NULL,
  Mgr_ssn    CHAR(9)      NOT NULL,
  Mgr_start_date DATE,
  PRIMARY KEY (Dnumber),
  UNIQUE     (Dname),
  FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
```

Create Tables to Store Data

```
INSERT INTO DEPARTMENT  
VALUES ('Research',5,333445555,'1988-05-22'),  
       ('Administration',4,987654321,'1995-01-01'),  
       ('Headquarters',1,888665555,'1981-06-19');
```

Create Tables to Store Data

EXAMPLE

Table Name: DEPT_LOCATIONS

Column Name	Dnumber	Dlocation
Key Type	PK	PK, FK
Nulls/Unique	NN,U	NN
FK Ref Table		DEPARTMENT
FK Ref Column		Dnumber
Data Type	INT	VARCHAR
Maximum Length		15
Sample data	1	Houston

```
CREATE TABLE DEPT_LOCATIONS
( Dnumber      INT          NOT NULL,
  Dlocation    VARCHAR(15)  NOT NULL,
  PRIMARY KEY (Dnumber, Dlocation),
  FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
```

Create Tables to Store Data

```
INSERT INTO DEPT_LOCATIONS  
VALUES (1,'Houston'),  
       (4,'Stafford'),  
       (5,'Bellaire'),  
       (5,'Sugarland'),  
       (5,'Houston');
```


Create Tables to Store Data

EXAMPLE

Table Name: PROJECT

Column Name	Pname	Pnumber	Plocation	Dnum
Key Type		PK		FK
Nulls/Unique	NN	NN		NN
FK Ref Table				DEPARTMENT
FK Ref Column				Dnumber
Data Type	VARCHAR	INT	VARCHAR	INT
Maximum Length	15		15	
Sample data	ProductX	1	Bellaire	5

```
CREATE TABLE PROJECT
( Pname    VARCHAR(15)  NOT NULL,
  Pnumber  INT          NOT NULL,
  Plocation VARCHAR(15),
  Dnum     INT          NOT NULL,
  PRIMARY KEY (Pnumber),
  UNIQUE    (Pname),
  FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );
```

Create Tables to Store Data

```
INSERT INTO PROJECT
VALUES ('ProductX',1,'Bellaire',5),
      ('ProductY',2,'Sugarland',5),
      ('ProductZ',3,'Houston',5),
      ('Computerization',10,'Stafford',4),
      ('Reorganization',20,'Houston',1),
      ('Newbenefits',30,'Stafford',4);
```

Create Tables to Store Data

EXAMPLE

Table Name: **WORKS_ON**

Column Name	Essn	Pno	Hours
Key Type	PK,FK	PK,FK	
Nulls/Unique	NN	NN	NN
FK Ref Table			
FK Ref Column			
Data Type	CHAR	INT	DECIMAL
Maximum Length	9		3,1
Sample data	123456789	1	32.5

```
CREATE TABLE WORKS_ON
( Essn      CHAR(9)      NOT NULL,
  Pno       INT          NOT NULL,
  Hours     DECIMAL(3,1) NOT NULL,
  PRIMARY KEY (Essn, Pno),
  FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
  FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );
```

Create Tables to Store Data

```
INSERT INTO WORKS_ON  
VALUES (123456789,1,32.5),  
      (123456789,2,7.5),  
      (666884444,3,40.0),  
      (453453453,1,20.0),  
      (453453453,2,20.0),  
      (333445555,2,10.0),  
      (333445555,3,10.0),  
      (333445555,10,10.0),  
      (333445555,20,10.0),  
      (999887777,30,30.0),  
      (999887777,10,10.0),  
      (987987987,10,35.0),  
      (987987987,30,5.0),  
      (987654321,30,20.0),  
      (987654321,20,15.0),  
      (888665555,20,16.0);
```

Create Tables to Store Data

EXAMPLE

Table Name: **DEPENDENT**

Column Name	Essn	Dependent_name	Sex	Bdate	Relationship
Key Type	PK,FK	PK			
Nulls/Unique	NN	NN			
FK Ref Table	EMPLOYEE				
FK Ref Column	Ssn				
Data Type	CHAR	VARCHAR	CHAR	DATE	VARCHAR
Maximum Length	9	15			8
Sample data	3334455555	Alice	F	1986-04-05	Daughter

```
CREATE TABLE DEPENDENT
( Essn      CHAR(9)      NOT NULL,
  Dependent_name VARCHAR(15) NOT NULL,
  Sex       CHAR,
  Bdate     DATE,
  Relationship VARCHAR(8),
  PRIMARY KEY (Essn, Dependent_name),
  FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn) );
```

Create Tables to Store Data

```
INSERT INTO DEPENDENT
VALUES (333445555,'Alice','F','1986-04-04','Daughter'),
       (333445555,'Theodore','M','1983-10-25','Son'),
       (333445555,'Joy','F','1958-05-03','Spouse'),
       (987654321,'Abner','M','1942-02-28','Spouse'),
       (123456789,'Michael','M','1988-01-04','Son'),
       (123456789,'Alice','F','1988-12-30','Daughter'),
       (123456789,'Elizabeth','F','1967-05-05','Spouse');
```

CONFIRM THE STRUCTURE OF A TABLE

테이블 구조 확인

Syntax

```
DESCRIBE table_name
```

Example

```
mysql> DESCRIBE EMPLOYEE;
```

```
mysql> describe EMPLOYEE;
```

Field	Type	Null	Key	Default	Extra
Fname	varchar(10)	NO		NULL	
Minit	char(1)	YES		NULL	
Lname	varchar(20)	NO		NULL	
Ssn	char(9)	NO	PRI	NULL	
Bdate	date	YES		NULL	
Address	varchar(30)	YES		NULL	
Sex	char(1)	YES		NULL	
Salary	decimal(5,0)	YES		NULL	
Super_ssn	char(9)	YES	MUL	NULL	
Dno	int	NO		NULL	

10 rows in set (0.00 sec)

ADD A COLUMN TO A TABLE

테이블에 칼럼 추가하기

Syntax

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

Example

```
ALTER TABLE DEPT_LOCATIONS  
ADD COLUMN (comments VARCHAR (255));  
  
DESCRIBE DEPT_LOCATIONS;
```


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

EMPLOYEE 테이블의 Fname 칼럼의 길이 20으로 변경 및 Bdate 값은 NULL이 될 수 없음

```
ALTER TABLE EMPLOYEE  
MODIFY Fname varchar(20);  
  
SQL> ALTER TABLE EMPLOYEE  
MODIFY Bdate date NOT NULL;
```

ADD AND REMOVE DATA CONSTRAINTS

Example

s_emp 테이블의 제약조건 조회

```
select * from information_schema.table_constraints where table_name='EMPLOYEE';
```

```
mysql> select * from information_schema.table_constraints where table_name='EMPLOYEE';
```

CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE	ENFORCED
def	taba	PRIMARY	taba	EMPLOYEE	PRIMARY KEY	YES
def	taba	EMPLOYEE_ibfk_1	taba	EMPLOYEE	FOREIGN KEY	YES

```
2 rows in set (0.00 sec)
```

DROP A TABLE

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

Syntax

```
DROP TABLE table_name
```

Example

EX 예제 테이블 삭제

```
CREATE TABLE EX(ID INT);  
  
DROP TABLE EX;
```

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	<i>view_name</i>	뷰 이름
	<i>alias</i>	별칭
	<i>query</i>	SELECT 문
	<i>WITH CHECK OPTION</i>	해당 옵션을 사용하면 INSERT/UPDATE 가능
	<i>WITH READ ONLY</i>	읽기 전용 뷰
	<i>constraint</i>	CHECK OPTION에 할당 된 제약조건

CREATE VIEWS

Example

부서 번호가 5인 직원의 SSN, 이름이 포함된 뷰를 생성

```
CREATE VIEW EMP5
AS SELECT Ssn, Fname, Lname
FROM EMPLOYEE
WHERE Dno = 5;
```

```
select * from EMP5;
```

Ssn	Fname	Lname
123456789	John	Smith
333445555	Franklin	Wong
453453453	Joyce	English
666884444	Ramesh	Narayan

4 rows in set (0.01 sec)

CREATE VIEWS

Example

부서 번호가 4인 뷰를 생성. Ssn은 ID , Fname은 Employee로 표현.

```
CREATE VIEW EMP4 (ID, Employee)
AS SELECT Ssn, Fname
FROM EMPLOYEE
WHERE Dno = 4;
```

```
mysql> select * from EMP4;
```

```
+-----+-----+
| ID    | Employee |
+-----+-----+
| 987654321 | Jennifer |
| 987987987 | Ahmad   |
| 999887777 | Alicia  |
+-----+-----+
```

```
3 rows in set (0.00 sec)
```

```
5 rows selected.
```

CREATE VIEWS

Example

월급이 30,000 이상인 뷰를 생성. 모든 정보를 포함

```
CREATE VIEW SAL30000
AS SELECT *
FROM EMPLOYEE
WHERE Salary >= 30000;
```

```
SQL> SELECT *
FROM SAL30000;
```

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1965-12-08	638 Voss, Houston TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble TX	M	38000	333445555	5
James	E	Borg	888665555	1937-11-10	450 Stone, Houston TX	M	55000	NULL	1
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire TX	F	43000	888665555	4

DROP A VIEW

Syntax

```
DROP VIEW view_name
```

Example

EMP5 뷰 삭제

```
DROP VIEW EMP5;
```

Control User Access: Overview

- 데이터베이스 관리자는 사용자에게 SQL 보안 명령을 사용해 테이블에 대한 액세스 권한을 제공
- Control User Access
 - 데이터베이스에 대한 권한 제공
 - 테이블같은 사용자 개체에 대한 액세스를 제공하고 제거
 - 데이터 사전에서 주어진 권한 및 받은 권한 확인

SYSTEM PRIVILEGES: OVERVIEW

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

시스템 권한 유형

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 업데이트할 수 있는 권한
On the System or a User	사용자를 생성할 수 있는 권한

SYSTEM PRIVILEGES: OVERVIEW

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

시스템 권한 유형

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 부여할 수 있다.

Syntax

```
GRANT system_priv ON Database.table TO [user, role] @'localhost'
```

where	<i>system_priv</i>	부여되는 시스템 권한
	<i>Database</i>	데이터베이스 명
	<i>table</i>	테이블 명
	<i>TO</i>	권한을 부여 받는 대상
	<i>user</i>	일반 사용자
	<i>role</i>	권한 받을 역할

GRANT SYSTEM PRIVILEGES

Example

MySQL에 등록된 사용자 확인

```
use mysql;
```

```
select user, host from user;
```

```
+-----+-----+
| user      | host    |
+-----+-----+
| mysql.infoschema | localhost |
| mysql.session   | localhost |
| mysql.sys       | localhost |
| root           | localhost |
+-----+-----+
```

GRANT SYSTEM PRIVILEGES

Example

실습을 위해 사용자 생성

```
CREATE USER 'USER명'@'localhost' IDENTIFIED BY 'Password'; (password 대/소문자, 숫자, 기호 포함)
```

```
CREATE USER 'scott'@'localhost' IDENTIFIED BY 'Tibero1!';
```

GRANT SQL 명령을 사용해 **scott** 사용자가 **taba** 데이터베이스에서 **CREATE**를 할 수 있는 권한 부여

```
GRANT CREATE ON taba.* to 'scott'@'localhost';
```

scott 사용자가 **taba** 데이터베이스에서 **SELECT, INSERT**를 할 수 있는 권한 부여

```
GRANT SELECT, INSERT ON taba.* to 'scott'@'localhost';
```

GRANT OBJECT PRIVILEGES

Example

scott에게 EMPLOYEE 테이블을 조회 할 수 있는 권한 부여

```
GRANT SELECT ON taba.EMPLOYEE TO 'scott'@'localhost';
```

scott에게 EMPLOYEE 테이블의 급여를 수정할 수 있는 권한 부여

```
GRANT UPDATE(Salary)  
ON taba.EMPLOYEE  
TO 'scott'@'localhost';
```


GRANT OBJECT PRIVILEGES

Example

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

```
GRANT SELECT
ON taba.DEPARTMENT
TO 'scott'@'localhost'
WITH GRANT OPTION;
```

현재 사용자가 scott에게 부여된 객체 조회

```
+-----+
| Grants for scott@localhost |
+-----+
| GRANT USAGE ON *.* TO `scott`@`localhost` |
| GRANT SELECT, INSERT, UPDATE, CREATE ON `taba`.* TO `scott`@`localhost` |
| GRANT SELECT ON `taba`.`DEPARTMENT` TO `scott`@`localhost` WITH GRANT OPTION |
| GRANT SELECT, UPDATE (`Salary`) ON `taba`.`EMPLOYEE` TO `scott`@`localhost` |
+-----+
```

REMOVE OBJECT PRIVILEGES

- 데이터베이스 관리자는 **REVOKE SQL** 명령을 사용하여 사용자에게 부여된 권한을 제거

Syntax

```
REVOKE system_priv ON Database.table FROM [user, role] @'localhost'
```

where	<i>system_priv</i>	부여되는 시스템 권한
	<i>Database</i>	데이터베이스 명
	<i>table</i>	테이블 명
	<i>FROM</i>	권한을 제거할 대상
	<i>user</i>	일반 사용자
	<i>role</i>	권한 받을 역할

REMOVE OBJECT PRIVILEGES

scott에게 DEPARTMENT 테이블을 조회할 수 있는 권한 제거

Example

```
REVOKE SELECT
ON taba.DEPARTMENT
FROM 'scott'@'localhost';
```

scott의 모든 권한을 제거

Example

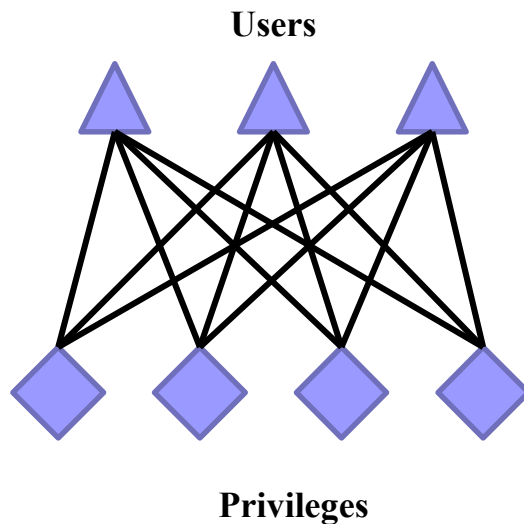
```
REVOKE ALL
ON *.*
FROM 'scott'@'localhost';

SHOW GRANTS FOR 'scott'@'localhost';
```

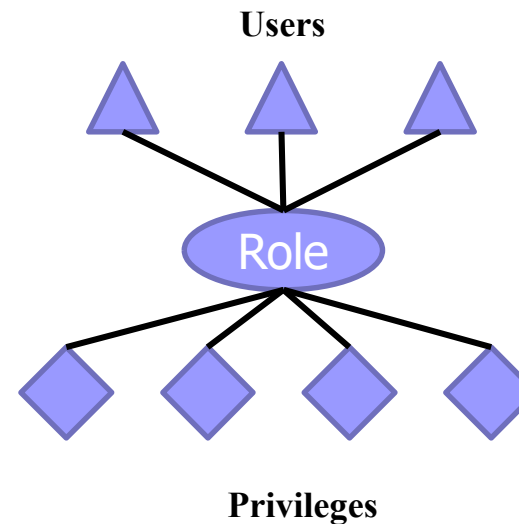
PRIVILEGES GROUPED BY ROLE

- 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;
```

Example

manager 역할 생성

```
CREATE ROLE manager;
```

manager 역할에 EMPLOYEE 테이블의 Salary 칼럼 수정, 데이터값 삽입이 가능하도록 권한 부여

```
GRANT UPDATE(Salary), INSERT  
ON taba.EMPLOYEE  
TO manager;
```

scott에게 manager 권한 부여

```
GRANT manager  
TO 'scott'@'localhost';
```



TABA_DB/SQL실습1-2

데이터 조작용어(Data Manipulation Language)

데이터 조작용어(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 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

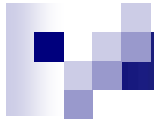
Syntax

```
SELECT *
FROM table_name
```

Example

```
SELECT *
FROM DEPARTMENT;
```

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Headquarters	1	888665555	1981-06-19
Administration	4	987654321	1995-01-01
Research	5	333445555	1988-05-22



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

```
SELECT FNAME  
FROM EMPLOYEE;
```

```
+-----+  
| FNAME |  
+-----+  
| John  |  
| Franklin |  
| Joyce |  
| Ramesh |  
| James |  
| Jennifer |  
| Ahmad |  
| Alicia |  
+-----+
```

DISPLAY ALL DATA IN A TABLE

S_EMP 테이블의 칼럼과 구조를 조회하고, 사원 이름, 급여를 출력

Example

```
SELECT Fname, Lname, Salary
FROM EMPLOYEE;
```

```
+-----+-----+-----+
| Fname | Lname | Salary |
+-----+-----+-----+
| John  | Smith | 30000  |
| Franklin | Wong | 40000  |
| Joyce | English | 25000  |
| Ramesh | Narayan | 38000  |
| James | Borg   | 55000  |
| Jennifer | Wallace | 43000  |
| Ahmad  | Jabbar | 25000  |
| Alicia | Zelaya | 25000  |
+-----+-----+-----+
```

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

```
mysql> SELECT Dno  
-> FROM EMPLOYEE;
```

```
+-----+  
| Dno |  
+-----+  
| 5 |  
| 5 |  
| 5 |  
| 5 |  
| 1 |  
| 4 |  
| 4 |  
| 4 |  
+-----+
```

```
mysql> SELECT DISTINCT Dno  
-> FROM EMPLOYEE;
```

```
+-----+  
| Dno |  
+-----+  
| 5 |  
| 1 |  
| 4 |  
+-----+
```

DISPLAY SPECIFIC COLUMN NAMES

별칭 - SELECT 문에서 칼럼 이름을 대체로 정의할 수 있다.

별칭에 공백, 특수문자가 포함되어 있으면 별칭을 큰 따옴표로 묶는다.

Example

```
mysql> SELECT Lname Lastname
-> FROM EMPLOYEE;
```

```
+-----+
| Lastname |
+-----+
| Smith   |
| Wong    |
| English |
| Narayan |
| Borg     |
| Wallace |
| Jabbar   |
| Zelaya   |
+-----+
```

```
mysql> SELECT Lname "Last_name"
-> FROM EMPLOYEE;
```

```
+-----+
| Last_name |
+-----+
| Smith     |
| Wong      |
| English    |
| Narayan   |
| Borg       |
| Wallace   |
| Jabbar     |
| Zelaya     |
+-----+
```

DISPLAY SPECIFIC ROWS OF DATA

WHERE절을 사용해 조건에 맞는 행을 조회한다.

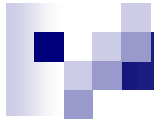
Syntax

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

Example

```
SELECT Fname, Salary, Dno
FROM EMPLOYEE
WHERE salary > 25000;
```

```
+-----+-----+-----+
| Fname | Salary | Dno |
+-----+-----+-----+
| John  | 30000  | 5   |
| Franklin | 40000  | 5   |
| Ramesh | 38000  | 5   |
| James | 55000  | 1   |
| Jennifer | 43000  | 4   |
+-----+-----+-----+
```



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
IS NOT NULL	Is not a null

DISPLAY SPECIFIC COLUMN NAMES

=

부서 번호가 4인 직원의 부서 번호, 사원 이름, SSN을 조회

Example

```
SELECT Dno, Fname, Lname, Ssn
FROM EMPLOYEE
WHERE Dno = 4;
```

Dno	Fname	Lname	Ssn
4	Jennifer	Wallace	987654321
4	Ahmad	Jabbar	987987987
4	Alicia	Zelaya	999887777

DISPLAY SPECIFIC COLUMN NAMES

=

Bdate가 1972-07-31인 사원의 이름, 주소, Bdate 조회

Example

```
SELECT Fname, Lname, Address, Bdate
FROM EMPLOYEE
WHERE Bdate = '1972-07-31';
```

Fname	Lname	Address	Bdate
Joyce	English	5631 Rice, Houston TX	1972-07-31

DISPLAY SPECIFIC COLUMN NAMES

< >

Minit가 B가 아닌 사원의 모든 정보 조회

Example

```
SELECT *
FROM EMPLOYEE
WHERE Minit <> 'B';
```

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1965-12-08	638 Voss, Houston TX	M	40000	888665555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston TX	F	25000	333445555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble TX	M	38000	333445555	5
James	E	Borg	888665555	1937-11-10	450 Stone, Houston TX	M	55000	NULL	1
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire TX	F	43000	888665555	4
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston TX	M	25000	987654321	4
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring TX	F	25000	987654321	4

DISPLAY SPECIFIC COLUMN NAMES

>

Bdate가 1962-10-10보다 큰 사원의 이름 조회

Example

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE Bdate > '1962-10-10';
```

Fname	Lname
John	Smith
Franklin	Wong
Joyce	English
Ahmad	Jabbar
Alicia	Zelaya

급여가 50,000이상인 사원의 이름, 급여 조회

```
SELECT Fname, Lname, Salary
FROM EMPLOYEE
WHERE Salary > 50000;
```

Fname	Lname	Salary
James	Borg	55000

DISPLAY SPECIFIC COLUMN NAMES

>=

부서 번호가 3이상인 사원의 부서 번호, 이름 조회

Example

```
SELECT Dno, Fname, Lname
FROM EMPLOYEE
WHERE Dno >= 3;
```

Dno	Fname	Lname
5	John	Smith
5	Franklin	Wong
5	Joyce	English
5	Ramesh	Narayan
4	Jennifer	Wallace
4	Ahmad	Jabbar
4	Alicia	Zelaya

Fname0| Franklin 이상인 사원의 모든 정보 조회

```
SELECT *
FROM EMPLOYEE
WHERE Fname >= 'Franklin';
```

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston TX	M	30000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston TX	F	25000	333445555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble TX	M	38000	333445555	5
James	E	Borg	888665555	1937-11-10	450 Stone, Houston TX	M	55000	NULL	1
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire TX	F	43000	888665555	4

DISPLAY SPECIFIC COLUMN NAMES



입사 날짜가 1972-07-31 보다 작은 사원의 이름, 급여, Bdate 조회

Example

```
SELECT Fname, Lname, Salary, Bdate
FROM EMPLOYEE
WHERE Bdate < '1972-07-31';
```

Fname	Lname	Salary	Bdate
John	Smith	30000	1965-01-09
Franklin	Wong	40000	1965-12-08
Ramesh	Narayan	38000	1962-09-15
James	Borg	55000	1937-11-10
Jennifer	Wallace	43000	1941-06-20
Ahmad	Jabbar	25000	1969-03-29
Alicia	Zelaya	25000	1968-01-19

DISPLAY SPECIFIC COLUMN NAMES



급여가 30,000이하인 사원의 이름, 급여 조회

Example

```
SELECT Fname, Lname, Salary
FROM EMPLOYEE
WHERE Salary <= 30000;
```

```
+-----+-----+-----+
| Fname | Lname | Salary |
+-----+-----+-----+
| John  | Smith | 30000  |
| Joyce | English | 25000  |
| Ahmad | Jabbar | 25000  |
| Alicia | Zelaya | 25000  |
+-----+-----+-----+
```

DISPLAY SPECIFIC COLUMN NAMES

BETWEEN

Bdate가 1941-06-20과 1972-07-31사이에 있는 사원의 이름과 **Bdate** 조회

Example

```
SELECT Fname, Lname, Bdate
FROM EMPLOYEE
WHERE Bdate BETWEEN '1941-06-20' AND '1972-07-31';
```

Fname	Lname	Bdate
John	Smith	1965-01-09
Franklin	Wong	1965-12-08
Joyce	English	1972-07-31
Ramesh	Narayan	1962-09-15
Jennifer	Wallace	1941-06-20
Ahmad	Jabbar	1969-03-29
Alicia	Zelaya	1968-01-19

DISPLAY SPECIFIC COLUMN NAMES

**NOT
BETWEEN**

급여가 27,000과 40,000 사이에 있지 않은 사원의 이름과 급여 조회

Example

```
SELECT Fname, Lname, Salary
FROM EMPLOYEE
WHERE Salary NOT BETWEEN 27000 AND 40000;
```

Fname	Lname	Salary
Joyce	English	25000
James	Borg	55000
Jennifer	Wallace	43000
Ahmad	Jabbar	25000
Alicia	Zelaya	25000

DISPLAY SPECIFIC COLUMN NAMES

IN

Minit이 B와 A 사원 이름 조회

Example

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE Minit IN('B','A');
```

```
+-----+-----+
| Fname | Lname |
+-----+-----+
| John  | Smith  |
| Joyce | English|
+-----+-----+
```

DISPLAY SPECIFIC COLUMN NAMES

LIKE

Fname이 'J'으로 시작하는 사원 이름 조회

Example

```
SELECT Fname, Lname
-> FROM EMPLOYEE
WHERE Fname LIKE 'J%';
```

Fname	Lname
John	Smith
Joyce	English
James	Borg
Jennifer	Wallace

입사 날짜가 '20'로 끝나는 사원 이름, **Bdate** 조회

```
SELECT Fname, Lname, Bdate
FROM EMPLOYEE
WHERE Bdate LIKE '%20';
```

Fname	Lname	Bdate
Jennifer	Wallace	1941-06-20

DISPLAY SPECIFIC COLUMN NAMES

LIKE

Example

사원 이름에 'A'를 포함하는 사원 조회

```
SELECT Fname  
FROM EMPLOYEE  
WHERE Fname LIKE '%A%';
```

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

```
SELECT Fname  
FROM EMPLOYEE  
WHERE Fname LIKE 'B____F';
```

두 번째 문자가 'L'인 사원의 이름 조회

```
SELECT Fname  
FROM EMPLOYEE  
WHERE Fname LIKE '_L%';
```

DISPLAY SPECIFIC COLUMN NAMES

**IS NOT
NULL**

커미션이 NULL 값인 아닌 사원의 이름, 직무, 커미션 조회

Example

```
SELECT FNAME, LNAME, SUPER_SSN
FROM EMPLOYEE
WHERE SUPER_SSN IS NOT NULL;
```

```
+-----+-----+-----+
| FNAME | LNAME | SUPER_SSN |
+-----+-----+-----+
| John  | Smith | 333445555 |
| Joyce | English | 333445555 |
| Ramesh | Narayan | 333445555 |
| Franklin | Wong | 888665555 |
| Jennifer | Wallace | 888665555 |
| Ahmad | Jabbar | 987654321 |
| Alicia | Zelaya | 987654321 |
+-----+-----+-----+
```

DISPLAY ROWS WITH COMPLEX CONDITIONS

AND, OR을 이용해 WHERE절에서 여러 조건을 결합하여 테이블 조회

Syntax

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

DISPLAY ROWS WITH COMPLEX CONDITIONS

AND

부서 번호가 4이고, 급여가 40,000 보다 큰 사원의 이름, 급여, 부서 번호 조회

Example

```
SELECT Fname, Lname, Salary, Dno
FROM EMPLOYEE
WHERE Dno=4 AND Salary > 40000;
```

Fname	Lname	Salary	Dno
Jennifer	Wallace	43000	4

부서 번호가 5이고, Minit이 B인 사원의 이름, 급여, 부서 번호 조회

```
SELECT Fname, Lname, Salary, Dno
FROM EMPLOYEE
WHERE Dno = 5 AND Minit ='B';
```

Fname	Lname	Salary	Dno
John	Smith	30000	5

DISPLAY ROWS WITH COMPLEX CONDITIONS

OR

부서 번호가 4이거나, 급여가 30,000보다 큰 사원의 이름, 급여, 부서 번호 조회

Example

```
SELECT Fname, Lname, Salary, Dno
FROM EMPLOYEE
WHERE Dno =4 OR Salary > 30000;
```

Fname	Lname	Salary	Dno
Franklin	Wong	40000	5
Ramesh	Narayan	38000	5
James	Borg	55000	1
Jennifer	Wallace	43000	4
Ahmad	Jabbar	25000	4
Alicia	Zelaya	25000	4

ORDER THE ROWS DISPLAYED

특정 칼럼을 기준으로 정렬이 가능하다.

Syntax

```
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

```
SELECT Fname, Lname, Salary
FROM EMPLOYEE
ORDER BY Salary;
```

Fname	Lname	Salary
Joyce	English	25000
Ahmad	Jabbar	25000
Alicia	Zelaya	25000
John	Smith	30000
Ramesh	Narayan	38000
Franklin	Wong	40000
Jennifer	Wallace	43000
James	Borg	55000

ORDER THE ROWS DISPLAYED

급여를 기준으로 내림차순 정렬

Example

```
SELECT Fname, Lname, Salary
FROM EMPLOYEE
ORDER BY Salary DESC;
```

Fname	Lname	Salary
James	Borg	55000
Jennifer	Wallace	43000
Franklin	Wong	40000
Ramesh	Narayan	38000
John	Smith	30000
Joyce	English	25000
Ahmad	Jabbar	25000
Alicia	Zelaya	25000

ORDER THE ROWS DISPLAYED

급여가 30,000이상인 사원의 이름, 부서 번호, 급여를 조회하고, 부서 번호로 오름차순, 급여로 내림차순하여 정렬

Example

```
SELECT Fname, Lname, Dno, Salary
FROM EMPLOYEE
WHERE Salary >= 30000
ORDER BY Dno, Salary DESC;
```

Fname	Lname	Dno	Salary
James	Borg	1	55000
Jennifer	Wallace	4	43000
Franklin	Wong	5	40000
Ramesh	Narayan	5	38000
John	Smith	5	30000



TABA_DB/SQL실습2



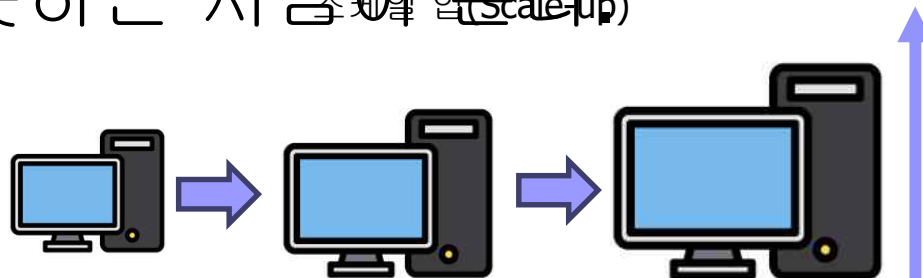
전통적 데이터 처리 시스템

- 전통적인 데이터 처리 시스템
 - 과거에는 데이터 처리를 위한 시스템 확장이 어려웠음
 - 전통적인 방식에서는 한 대의 컴퓨터의 처리 능력에 제한

전통적 데이터 처리 시스템

■ 스케일 업

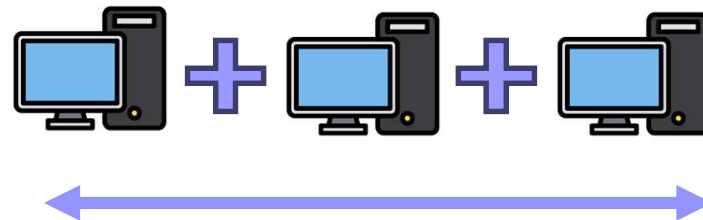
- 시스템을 확장할 때 구조를 바꿀 필요가 없다.
- 소프트웨어를 고사양 서버로 단순히 이관하는 방식으로 확장성을 높임 (생각보다 어려움)
- 언젠가는 스케일 업 방식으로 더 이상 확장하지 못하는 시점이 온다



전통적 데이터 처리 시스템

■ 스케일 아웃

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





전통적 데이터 처리 시스템

■ 한계점

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



전통적 데이터 처리 시스템

■ 스케일 업의 한계

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



전통적 데이터 처리 시스템

■ 스케일 업의 한계

- 많은 작업을 병렬 처리하는 하드웨어를 유연하게 사용하기 위해서는 소프트웨어를 똑똑하고, 하드웨어는 단순하게
- 하드웨어는 리소스 셋으로만 이용
- 소프트웨어가 처리 작업들에 하드웨어를 할당하는 역할



솔루션

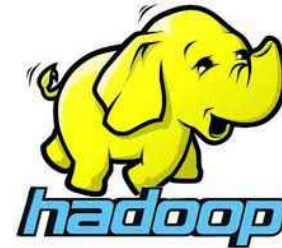
■ 장애 예측

- 서버가 늘어나도 각 서버의 장애나 문제점은 영향을 주지 않게 해야함.
- 각각의 장비는 주기적으로 장애가 발생할 수 있다는 점을 고려

하둡(Hadoop)

■ 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(Hadoop)

■ 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 단계로 나눈다.

하둡(Hadoop)

■ 공통 구성 요소

□ HDFS와 맵리듀스는 아래와 같은 원칙을 지킨다.

- 저가 서버들로 구성된 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



하둡(Hadoop)

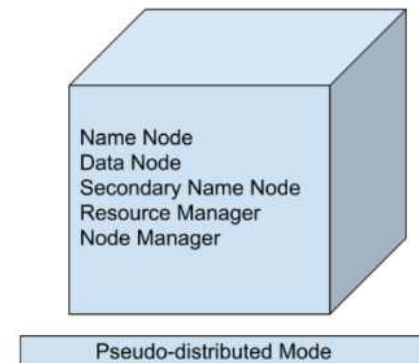
- 하둡의 세 가지 구동방식
 - 로컬 자립형 방식(Standalone Mode)
 - Hadoop이 실행되는 기본 모드
 - HDFS를 사용하지 않음
 - mapred-site.xml, core-site.xml, hdfs-site.xml 등 구성 파일을 수정할 필요 없음
 - 디버깅 목적으로 사용

하둡(Hadoop)

- 하둡의 세 가지 구동방식

- 가분산 방식(Pseudo-distributed Mode)

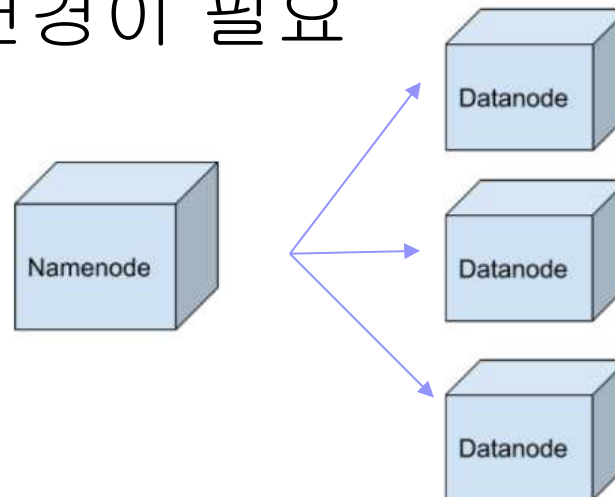
- Namenode와 Datanode가 모두 동일한 시스템에 있음
 - 최적화된 미니 클러스터를 효과적으로 생성
 - 구성 파일 변경이 필요



하둡(Hadoop)

■ 하둡의 세 가지 구동방식

- masternode와 datanode가 별도로 되어 있음
- 데이터가 여러 노드에 걸쳐 사용되고 분산
- 구성 파일 변경이 필요





하둡(Hadoop)

- JAVA 및 ssh설치

- `sudo apt install -y openjdk-8-jdk openssh-server openssh-client`

- 하둡 계정 추가

- `sudo adduser hdoop`

- `su hdoop #hdoop` 계정으로 접속



하둡(Hadoop)

- 하둡 다운로드

`wget https://downloads.apache.org/hadoop/common/hadoop-3.2.4/hadoop-3.2.4.tar.gz --no-check-certificate`

- 하둡 압축 풀기

`tar -xzvf hadoop-3.2.4.tar.gz`

하둡(Hadoop)

■ 하둡 환경변수 설정(~/.bashrc 파일 설정)

□ vi ~/.bashrc

```
## HADOOP ##
export HADOOP_HOME=/home/hadoop/hadoop-3.2.4
export HADOOP_INSTALL=$HADOOP_HOME export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

##JAVA##

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

HADOOP

export HADOOP_HOME=/home/hadoop/hadoop-3.2.4

export HADOOP_INSTALL=\$HADOOP_HOME

export HADOOP_MAPRED_HOME=\$HADOOP_HOME

export HADOOP_COMMON_HOME=\$HADOOP_HOME

export HADOOP_HDFS_HOME=\$HADOOP_HOME

export YARN_HOME=\$HADOOP_HOME

export HADOOP_COMMON_LIB_NATIVE_DIR=\$HADOOP_HOME/lib/native

export PATH=\$PATH:\$HADOOP_HOME/sbin:\$HADOOP_HOME/bin:\$JAVA_HOME/bin

export HADOOP_OPTS="-Djava.library.path=\$HADOOP_HOME/lib/native"

##JAVA_HOME 위치 찾는 법

readlink -f /usr/bin/javac

bin 전까지 나온 경로가 JAVA_HOME의 위치

/usr/lib/jvm/java-8-openjdk-amd64

```
hadoop@ip-172-31-2-77:~/hadoop-3.2.4$ readlink -f /usr/bin/javac
/usr/lib/jvm/java-8-openjdk-amd64/bin/javac
```

:wq로 저장 후

source ~/.bashrc #환경변수 적용

echo \$HADOOP_HOME #적용 확인

하둡(Hadoop)

■ hadoop-env.sh 파일 설정

- vi \$HADOOP_HOME/etc/hadoop/hadoop-env.sh
- #export JAVA_HOME 주석 제거 후 JAVA_HOME 입력

```
# Technically, the only required environment variable is JAVA_HOME.
# All others are optional. However, the defaults are probably not
# preferred. Many sites configure these options outside of Hadoop,
# such as in /etc/profile.d

# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

# Location of Hadoop. By default, Hadoop will attempt to determine
# this location based upon its execution path.
```

하둡(Hadoop)

■ core-site.xml 설정

- HDFS와 하둡 핵심 property 정의

- vi \$HADOOP_HOME/etc/hadoop/core-site.xml

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/home/hadoop/tmpdata</value>
  </property>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/home/hadoop/tmpdata</value>
  </property>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://127.0.0.1:9000</value>
  </property>
</configuration>
```

- mkdir /home/hadoop/tmpdata #디렉토리 생성

하둡(Hadoop)

■ hdfs-site.xml 설정

□ namenode와 datanode 저장소 디렉토리 설정

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

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


하둡(Hadoop)

■ mapred-site.xml 설정

□ 맵리듀스 파일 값을 정의

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

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

하둡(Hadoop)

■ yarn-site.xml 설정

□ YARN에 관련된 세팅을 정의하는 파일

```
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>0.0.0.0</value>
</property>
<property>
  <name>yarn.resourcemanager.address</name>
  <value>0.0.0.0:8032</value>
</property>
```



하둡(Hadoop)

■ yarn-site.xml 설정

□ YARN에 관련된 세팅을 정의하는 파일

```
<property>
  <name>yarn.web-proxy.address</name>
  <value>0.0.0.0:8089</value>
</property>
<property>
  <name>yarn.acl.enable</name>
  <value>0</value>
</property>
<property>
  <name>yarn.nodemanager.env-whitelist</name>
  <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREPEND_DISTCACHE,
  HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
</property>
```

하둡(Hadoop)

■ yarn-site.xml 설정

```
<configuration>
<!-- Site specific YARN configuration properties -->
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>0.0.0.0</value>
  </property>
  <property>
    <name>yarn.resourcemanager.address</name>
    <value>0.0.0.0:8032</value>
  </property>
  <property>
    <name>yarn.web-proxy.address</name>
    <value>0.0.0.0:8089</value>
  </property>
  <property>
    <name>yarn.acl.enable</name>
    <value>0</value>
  </property>
  <property>
    <name>yarn.nodemanager.env-whitelist</name>
    <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PERPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
  </property>
</configuration>
```



하둡(Hadoop)

- SSH-KEYGEN

- 하둡은 각 노드들이 통신을 해야하기 때문에 패스워드 없이 접속하도록 ssh-keygen 설정

```
ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa  
cat ~/.ssh/id_rsa.pub >>  
~/.ssh/authorized_keys  
chmod 0600 ~/.ssh/authorized_keys
```

하둡(Hadoop)

■ 네임노드 포맷

- hadoop을 가동하기 전 HDFS 파일시스템을 포맷해야함

hdfs namenode -format

```
hadoop@ip-172-31-2-77:~/hadoop-3.2.4$ hdfs namenode -format
WARNING: /home/hadoop/hadoop-3.2.4/logs does not exist. Creating.
2024-03-19 02:58:58,941 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = ip-172-31-2-77/172.31.2.77
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.2.4
STARTUP_MSG:   classpath = /home/hadoop/hadoop-3.2.4/etc/hadoop:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/commons-collections-3.2.2.jar:/home/
/hadoop/hadoop-3.2.4/share/hadoop/common/lib/kerb-admin-1.0.1.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/jsr311-api-1.1.1.jar:/home/hadoop/h
hadoop/hadoop-3.2.4/share/hadoop/common/lib/commons-math3-3.1.1.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/paranamer-2.3.jar:/home/hadoop/h
hadoop-3.2.4/share/hadoop/common/lib/gson-2.9.0.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/jsr305-3.0.2.jar:/home/hadoop/hadoop-3.2.4/share
2.4/share/hadoop/common/lib/kerb-common-1.0.1.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/jetty-server-9.4.43.v20210629.jar:/home/hadoop/had
/hadoop/hadoop-3.2.4/share/hadoop/common/lib/jetty-io-9.4.43.v20210629.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/httpcore-4.4.13.jar:/home
ome/hadoop/hadoop-3.2.4/share/hadoop/common/lib/animal-sniffer-annotations-1.17.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/jersey-server-1
.5.2.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/kerb-core-1.0.1.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/jackson-databind-2.14
a-2.5.0.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/jaxb-api-2.2.11.jar:/home/hadoop/hadoop-3.2.4/share/hadoop/common/lib/jaxb-impl-2.2.3-1
```

하둡(Hadoop)

■ 하둡 실행하기

- start-dfs.sh
- start-yarn.sh
- jps

```
hadoop@ip-172-31-2-77:~/hadoop-3.2.4$ jps
5968 ResourceManager
5763 SecondaryNameNode
5459 NameNode
6262 WebAppProxyServer
6088 NodeManager
5581 DataNode
6509 Jps
```

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 사이즈
를 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)


■ 워드카운트 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 W  
-Ddfs.replication=1 W  
-Dmapred.map.tasks=36 W  
-Dmapred.reduce.tasks=16 W  
107374182 /teragen/teragen.data
```

복사본 1개 맵 테스트 36 / 리듀스 테스트 16 / 생성할 용량 10GB / 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-000000
```



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)

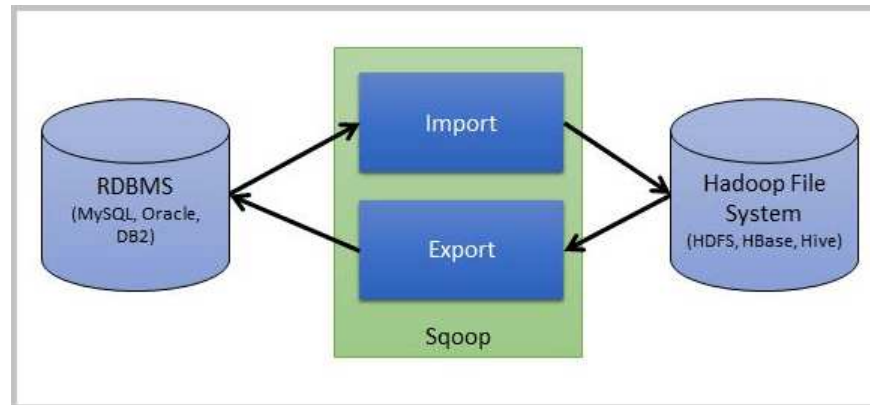
```
2024-03-24 18:26:34,215 INFO mapreduce.Job: Counters: 36
  File System Counters
    FILE: Number of bytes read=485799799745
    FILE: Number of bytes written=938092152454
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: 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=0
    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
```

Apache Sqoop

■ Apache Sqoop(SQL to Hadoop)

- Hadoop과 관계형 데이터베이스 간 데이터를 전송할 수 있도록 설계된 오픈소스 소프트웨어
- Oracle, MySQL등 RDBMS 특정 테이블, 데이터를 HDFS로 옮길 수 있음
- 반대로 HDFS에 저장된 데이터를 RDBMS로 옮길 수 있음
- Sqoop은 2009년 처음 버전이 나왔고, 현재 프로젝트는 종료됨



Apache Sqoop

■ 스쿵 설치

cd /home/hadoop

wget https://archive.apache.org/dist/sqoop/1.4.7/sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz

tar -xzf sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz #압축해제

mv sqoop-1.4.7.bin__hadoop-2.6.0 sqoop #폴더명 변경

rm sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz #tar파일 삭제

vi ~/.bashrc #환경변수 추가

export SQOOP_HOME=/home/hadoop/sqoop

```
export HADOOP_HOME=/home/hadoop/hadoop-3.0.0
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
export SQOOP_HOME=/home/hadoop/sqoop
```


Apache Sqoop

■ sqoop 환경설정

cd /home/hadoop

wget https://archive.apache.org/dist/sqoop/1.4.7/sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz

tar -xzf sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz #압축해제

mv mv sqoop-1.4.7.bin__hadoop-2.6.0 sqoop #폴더명 변경

rm sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz #tar파일 삭제

vi ~/.bashrc #환경변수 추가

export SQOOP_HOME=/home/hadoop/sqoop

PATH에 Sqoop도 추가

export PATH=\$PATH:\$HADOOP_HOME/sbin:\$HADOOP_HOME/bin:\$SQOOP_HOME/bin

source ~/.bashrc

```
## SQOOP ##
export SQOOP_HOME=/home/hadoop/sqoop

## HADOOP ##
export HADOOP_HOME=/home/hadoop/hadoop-3.1.4
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin:$SQOOP_HOME/bin
```

Apache Sqoop

■ sqoop 환경설정

```
cp sqoop/conf/sqoop-env-template.sh sqoop/conf/sqoop-env.sh
```

```
vi sqoop/conf/sqoop-env.sh
```

```
export HADOOP_COMMON_HOME=$HADOOP_HOME
```

```
export HADOOP_MAPRED_HOME=$HADOOP_HOME
```

```
#Set path to where bin/hadoop is available
export HADOOP_COMMON_HOME=$HADOOP_HOME

#Set path to where hadoop-*.core.jar is available
export HADOOP_MAPRED_HOME=$HADOOP_HOME

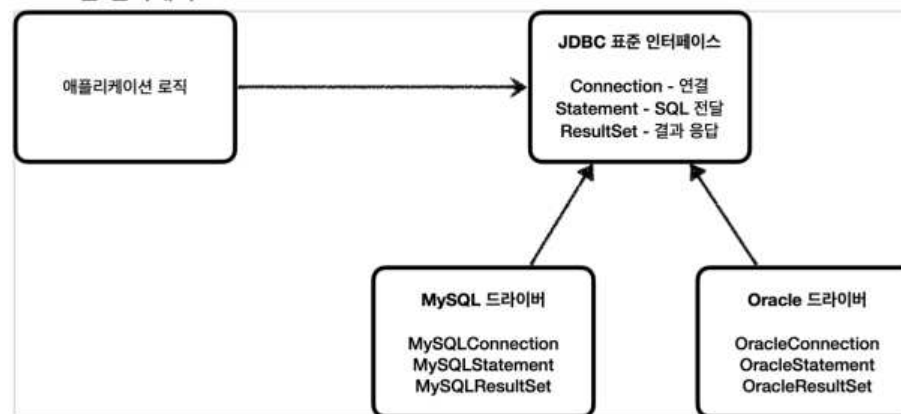
#set the path to where bin/hbase is available
#export HBASE_HOME=

#Set the path to where bin/hive is available
#export HIVE_HOME=

#Set the path for where zookeeper config dir is
#export ZOOKEEPER=
```

Sqoop 예제

- MySQL JDBC 다운
- JDBC(Java Database Connectivity) 는 Java기반 애플리케이션의 애플리케이션의 데이터를 데이터베이스에서 저장 및 업데이트하거나, 데이터베이스에 저장된 데이터를 Java에서 사용할 수 있도록 하는 자바 API





Sqoop 예제

- MySQL JDBC 다운 및 Sqoop lib 디렉토리에 옮기기

wget <https://downloads.mysql.com/archives/get/p/3/file/mysql-connector-j-8.0.33.tar.gz>

tar xzvf mysql-connector-j-8-0.33.tar.gz

cp mysql-connector-j-8.0.33/mysql-connector-j-8.0.33.jar \$SQOOP_HOME/lib/

- StringUtils 관련 오류 예방으로 commons 라이브러리 설치

wget <https://dlcdn.apache.org/commons/lang/binaries/commons-lang-2.6-bin.tar.gz>

tar zxvf commons-lang-2.6-bin.tar.gz

cp commons-lang-2.6/commons-lang-2.6.jar \$SQOOP_HOME/lib

Sqoop 예제

hadoop 계정에 sudo 권한주기

1. 기본 계정으로 접속
2. sudo passwd
3. su

```
ubuntu@ubuntu-VirtualBox:~$ sudo passwd
[sudo] password for ubuntu:
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: password updated successfully
ubuntu@ubuntu-VirtualBox:~$ su
Password:
root@ubuntu-VirtualBox:/home/ubuntu# exit
```

Sqoop 예제

hadoop 계정에 sudo 권한주기

sudoers 파일은 readonly 파일이라 일반적인 파일과 수정 방법이 다름.

vi /etc/sudoers

root 계정 아래 내용 추가

hadoop ALL=(ALL:ALL) ALL

w!

q!

exit 로 root 빠져나온 뒤 su hadoop 계정 접속

```
# Cnmd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
hadoop  ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Sqoop 예제

■ MySQL 설치

`sudo apt install -y mysql-server`

`sudo systemctl status mysql.service`

```
hdoop@ubuntu-VirtualBox:/home/ubuntu$ sudo systemctl status mysql.service
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2024-03-26 02:38:00 KST; 33s ago
     Process: 29465 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
    Main PID: 29473 (mysqld)
      Status: "Server is operational"
        Tasks: 38 (limit: 2261)
       Memory: 352.5M
          CPU: 1.571s
      CGroup: /system.slice/mysql.service
              └─29473 /usr/sbin/mysqld
```

Sqoop 예제

- MySQL 패스워드 설정
- 초기 mysql 접속 시 패스워드 없이 접속 가능

`mysql -u root -p`

패스워드 설정 (대소문자, 숫자, 특수기호, 8자리 이상 들어가야함.)

`ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'Dankook1!';`

```
hdoop@ubuntu-VirtualBox:/home/ubuntu$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.36-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'Dankook1!';
Query OK, 0 rows affected (0.02 sec)

mysql>
```


Sqoop 예제

- MySQL Test 데이터베이스 다운 및 압축풀기

wget https://github.com/datacharmer/test_db/releases/download/v1.0.7/test_db-1.0.7.tar.gz

tar -xvzf test_db-1.0.7.tar.gz

- MySQL 에 데이터 로드

cd test_db/

mysql -u root -p < employees.sql

```
hadoop@ubuntu-VirtualBox:~/test_db$ mysql -u root -p < employees.sql
Enter password:
INFO
CREATING DATABASE STRUCTURE
INFO
storage engine: InnoDB
INFO
LOADING departments
INFO
LOADING employees
INFO
LOADING dept_emp
INFO
LOADING dept_manager
INFO
LOADING titles
INFO
LOADING salaries
data_load_time_diff
00:01:18
```

Sqoop 예제

- 테스트 데이터베이스 조회하기

```
mysql -u root -p  
show databases;  
use employees;  
show tables;
```

```
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| employees |  
| information_schema |  
| mysql |  
| performance_schema |  
| sys |  
+-----+  
5 rows in set (0.05 sec)
```

```
mysql> show tables;  
+-----+  
| Tables_in_employees |  
+-----+  
| current_dept_emp |  
| departments |  
| dept_emp |  
| dept_emp_latest_date |  
| dept_manager |  
| employees |  
| salaries |  
| titles |  
+-----+  
8 rows in set (0.01 sec)
```



Sqoop 예제

- sqoop으로 MySQL 특정 데이터베이스의 테이블 조회하기
sqoop list-tables --connect jdbc:mysql://localhost/employees ₩
--username root ₩
--password Dankook1!
- sqoop으로 MySQL -> HDFS 데이터 보내기
sqoop import --connect jdbc:mysql://localhost/employees ₩
--username root --password Dankook1! --table employees -m 1

Sqoop 예제

```
2024-03-26 03:41:51,677 INFO mapreduce.Job: Job job_1711383692141_0001 running in uber mode : false
2024-03-26 03:41:51,689 INFO mapreduce.Job: map 0% reduce 0%
2024-03-26 03:42:12,855 INFO mapreduce.Job: map 100% reduce 0%
2024-03-26 03:42:17,590 INFO mapreduce.Job: Job job_1711383692141_0001 completed successfully
2024-03-26 03:42:18,984 INFO mapreduce.Job: Counters: 33
  File System Counters
    FILE: Number of bytes read=0
    FILE: Number of bytes written=246278
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=87
    HDFS: Number of bytes written=13821993
    HDFS: Number of read operations=6
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=2
    HDFS: Number of bytes read erasure-coded=0
  Job Counters
    Launched map tasks=1
    Other local map tasks=1
    Total time spent by all maps in occupied slots (ms)=18891
    Total time spent by all reduces in occupied slots (ms)=0
    Total time spent by all map tasks (ms)=18891
    Total vcore-milliseconds taken by all map tasks=18891
    Total megabyte-milliseconds taken by all map tasks=19344384
  Map-Reduce Framework
    Map input records=300024
    Map output records=300024
    Input split bytes=87
    Spilled Records=0
    Failed Shuffles=0
    Merged Map outputs=0
    GC time elapsed (ms)=242
    CPU time spent (ms)=5220
    Physical memory (bytes) snapshot=139124736
    Virtual memory (bytes) snapshot=2494951424
    Total committed heap usage (bytes)=32571392
    Peak Map Physical memory (bytes)=139124736
    Peak Map Virtual memory (bytes)=2494951424
  File Input Format Counters
    Bytes Read=0
  File Output Format Counters
    Bytes Written=13821993
2024-03-26 03:42:19,059 INFO mapreduce.ImportJobBase: Transferred 13.1817 MB in 155.7499 seconds (86.6648 KB/sec)
2024-03-26 03:42:19,072 INFO mapreduce.ImportJobBase: Retrieved 300024 records.
```

Sqoop 예제

- localhost:8088



▼ Cluster

- [About](#)
- [Nodes](#)
- [Node Labels](#)
- [Applications](#)
 - [NEW](#)
 - [NEW SAVING](#)
 - [SUBMITTED](#)
 - [ACCEPTED](#)
 - [RUNNING](#)
 - [FINISHED](#)
 - [FAILED](#)
 - [KILLED](#)
- [Scheduler](#)

► Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	...
1	0	0	1	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	...
1	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type
Capacity Scheduler	[memory-mb (unit=M), vcores]

Show 20 ▼ entries

ID	User	Name	Application Type	Queue	Application Priority
application_1711383692141_0001	hadoop	employees.jar	MAPREDUCE	default	0

Showing 1 to 1 of 1 entries



Sqoop 예제

- 명령어로 하둡 데이터 조회하기
- `hdfs dfs -ls`

```
hdoop@ubuntu-VirtualBox:~$ hdfs dfs -ls
Found 1 items
drwxr-xr-x  - hdoop supergroup          0 2024-03-26 03:42 employees
```



TABA_DB/SQL실습3

쿠버네티스란?



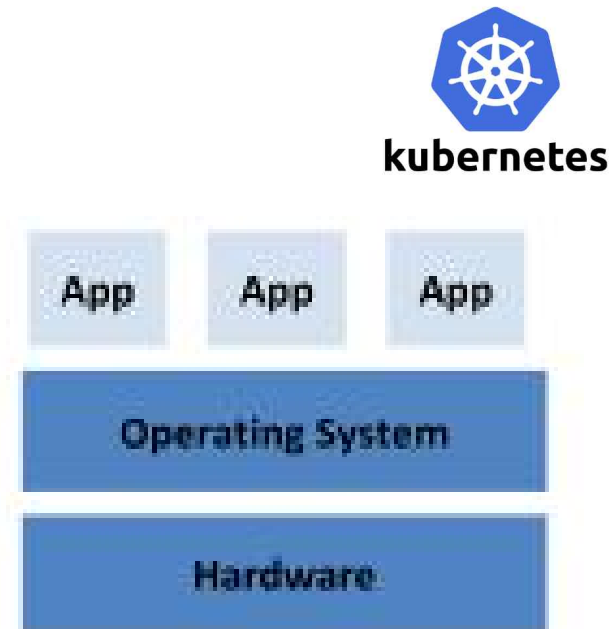
kubernetes

- 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?

■ 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



kubernetes

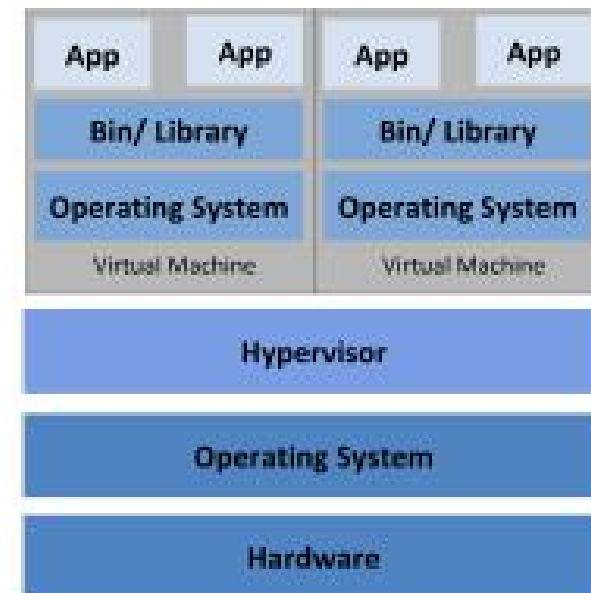
쿠버네티스란?



kubernetes

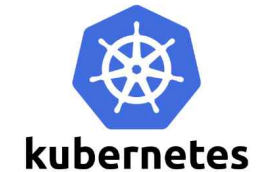
■ 가상화된 배포 시대(Virtualized Deployment)

- 전통적인 배포 시대의 해결책으로 가상화 도입
- 단일 물리서버의 **CPU**에서 여러 가상 머신 실행
- 가상화를 사용하면 **VM**간에 격리로 보안 제공
- 물리서버에서 리소스를 효율적으로 사용



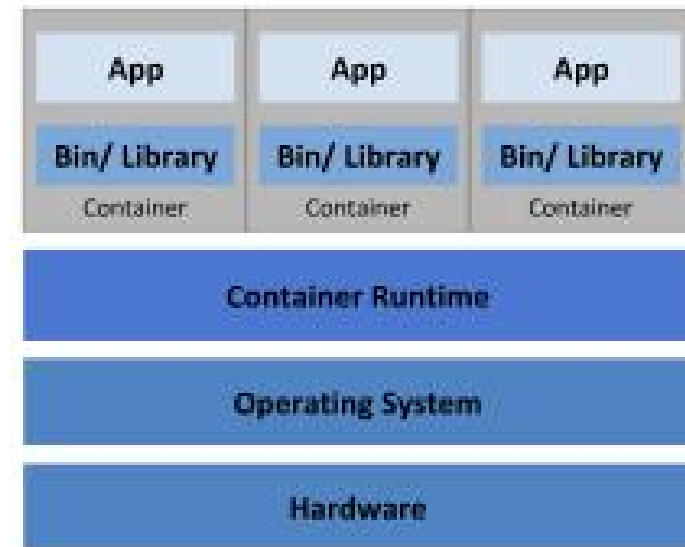
Virtualized Deployment

쿠버네티스란?



■ 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

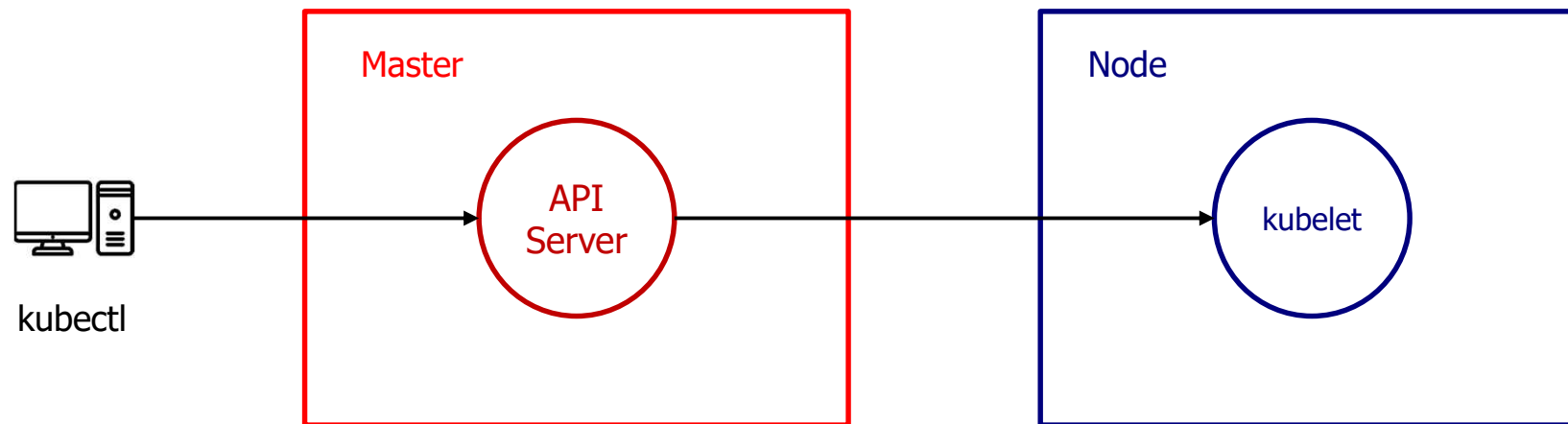


쿠버네티스란?

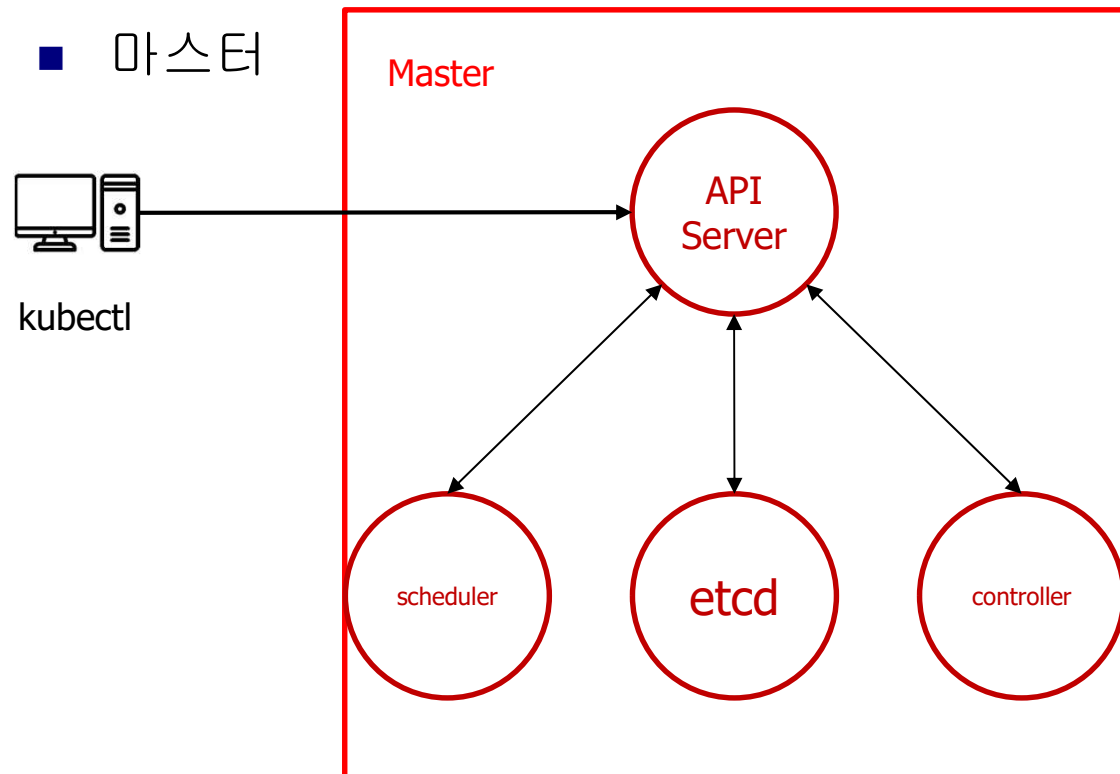
- 컨테이너 오케스트레이션
 - 컨테이너 기반 애플리케이션 배포 관리, 제어 및 모니터링, 스케일링, 네트워킹 관리 도구
- 컨테이너 오케스트레이션 종류
 - 도커 스웜(Docker Swarm), 아파치 메소스(Apache Mesos), 노마드(Nomad)
 - 쿠버네티스가 컨테이너 기반 인프라 시장에서 사실상 표준

쿠버네티스 구조

■ 마스터 - 노드 구조

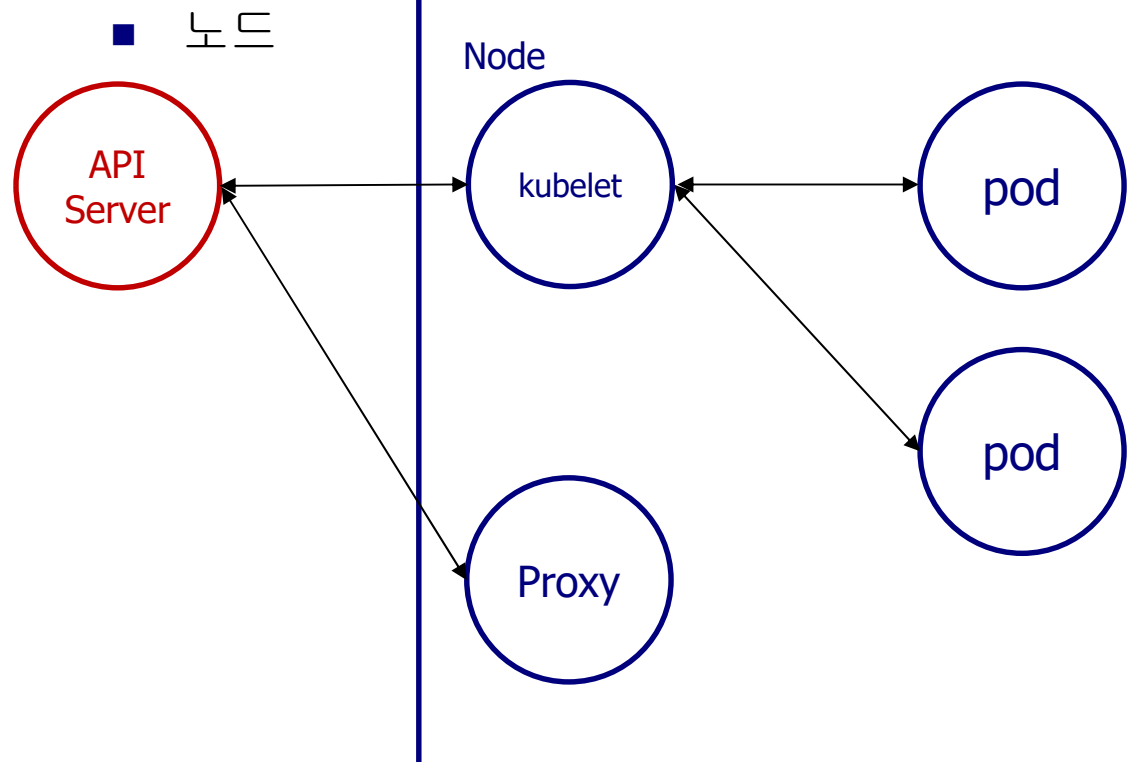


쿠버네티스 구조



- 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

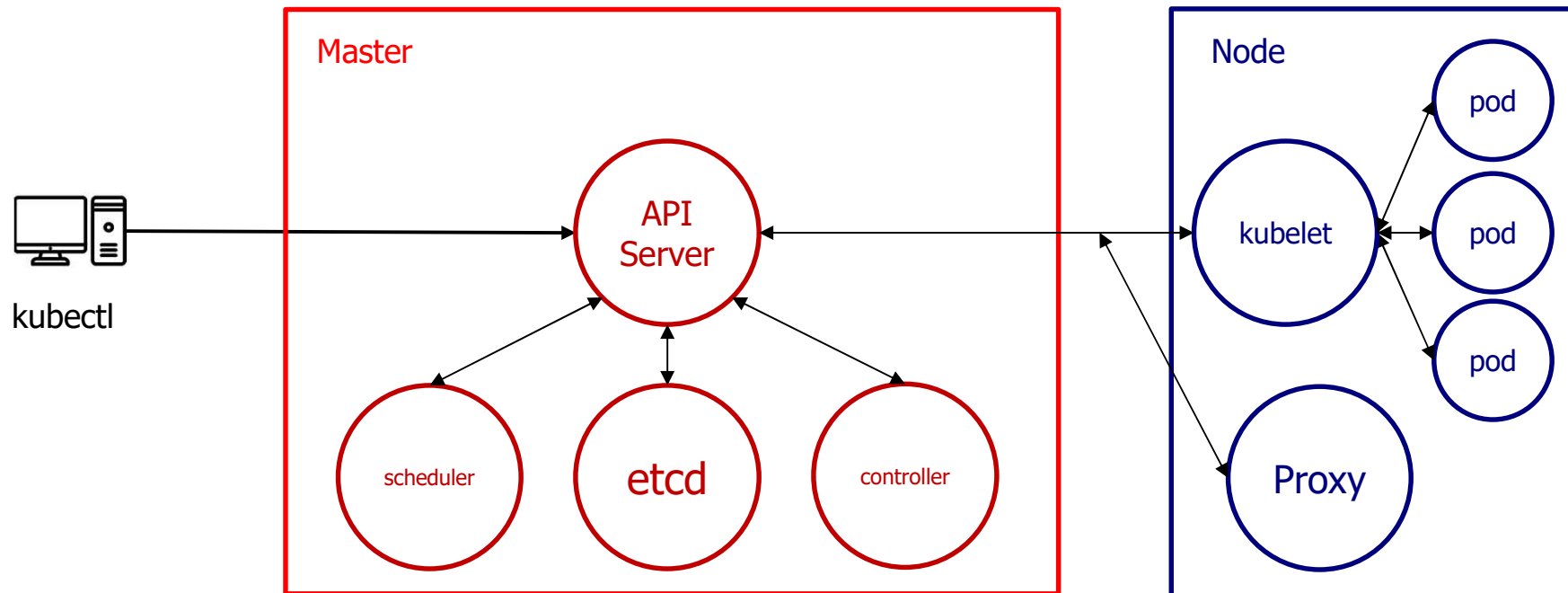
쿠버네티스 구조



- 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 n 개의 노드 서버가 필요.
- mac, Linux, Windows에서 K8s 클러스터를 빠르게 설정해주는 도구
- minikube를 이용하면 로컬에서 K8s 클러스터를 만들 수 있다.



minikube Install

■ 업데이트 및 유틸리티 설치

```
sudo yum update -y && sudo yum install -y utils yum-utils
```

■ Docker-ce 레포 추가

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

■ Docker 설치

```
sudo yum install -y docker-ce docker-ce-cli containerd.io
```



minikube Install

- Docker 실행 및 확인

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

```
sudo systemctl status docker
```

- 도커 그룹에 유저 추가

```
sudo groupadd docker #(docker group 생성)
```

```
sudo usermod -aG docker $USER #(docker group 해당유저 추가)
```

```
newgrp docker #(적용하기)
```



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.io/release/stable.txt)/bin/linux/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.io/release/stable.txt)/bin/linux/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 설정 확인
kubectl delete pod pod명 --grace-period=0 --force : pod 강제 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
- **Pod**는 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 **default, kube-node-lease, kube-public, kube-system**이라는 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: news1
```

- kubectl create -f my-namespace.yaml

- 아래 명령어로 네임스페이스 생성 가능
kubectl create namespace | ns news2

kubectl get ns

```
spark@ubuntu-VirtualBox:~$ kubectl create -f my-namespace.yaml
namespace/news1 created
spark@ubuntu-VirtualBox:~$ kubectl create namespace news2
namespace/news2 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
news1               Active   18s
news2               Active    5s
```

kubectl delete ns news2 #네임스페이스 삭제



쿠버네티스 오브젝트

■ 서비스(Service)

- 파드는 언제든지 다른 노드로 옮겨지거나 삭제될 수 있음.
- 생성될 때마다 새로운 **IP**를 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 **IP**를 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
- **Docker**의 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 **PV**를 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

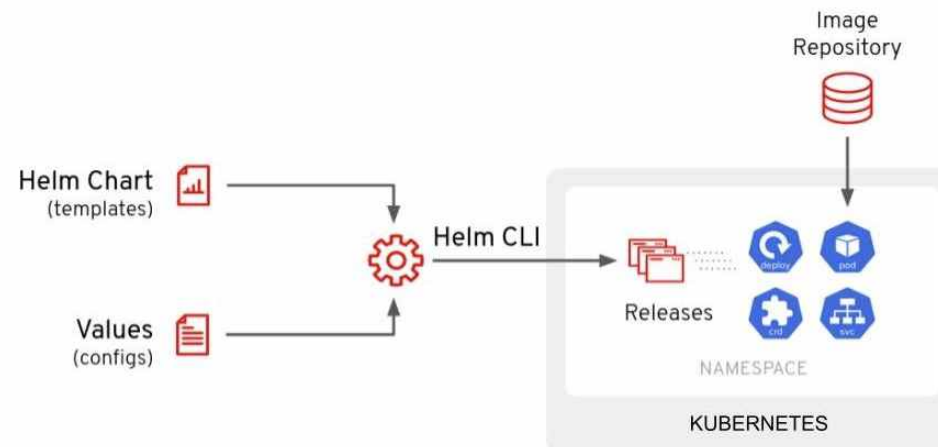
```
spark@ubuntu-VirtualBox:~$ kubectl create -f dempPV.yaml
persistentvolume/demo-pv created
spark@ubuntu-VirtualBox:~$ kubectl create -f dempPVC.yaml
persistentvolumeclaim/demo-pvc created
spark@ubuntu-VirtualBox:~$ kubectl get pv
```

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

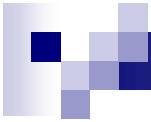
■ Helm 설치

```
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 `helm`차트를 `helm hub`에서 검색
- `helm install chart명` : 특정 `chart`패키지 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)

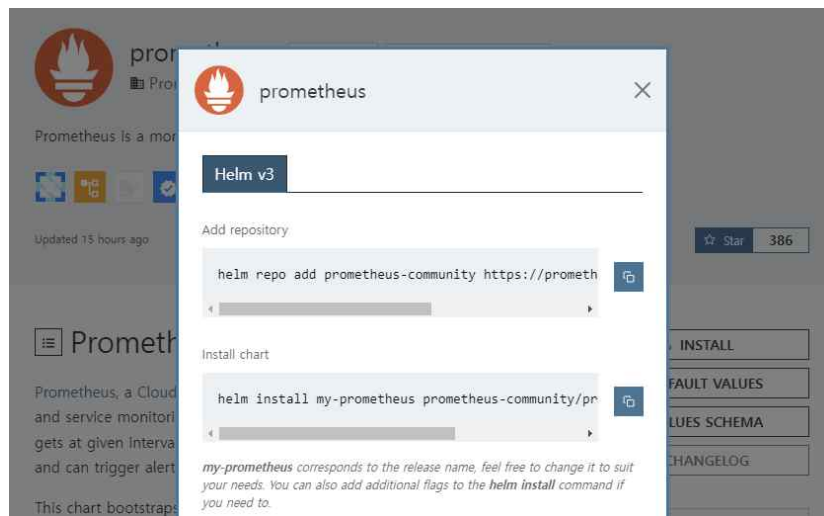
- SoundCloud에서 만든 오픈소스 모니터링 툴
- Kubernetes환경에서 모니터링하기 원하는 리소스로부터 **metric**을 수집하고 해당 **metric**을 이용해 모니터링

■ 그라파나(Grafana)

- 오픈소스 시각화 분석 도구
- 여러 데이터 소스에 대한 대시보드 템플릿 제공
- 프로메테우스도 **UI**를 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
```

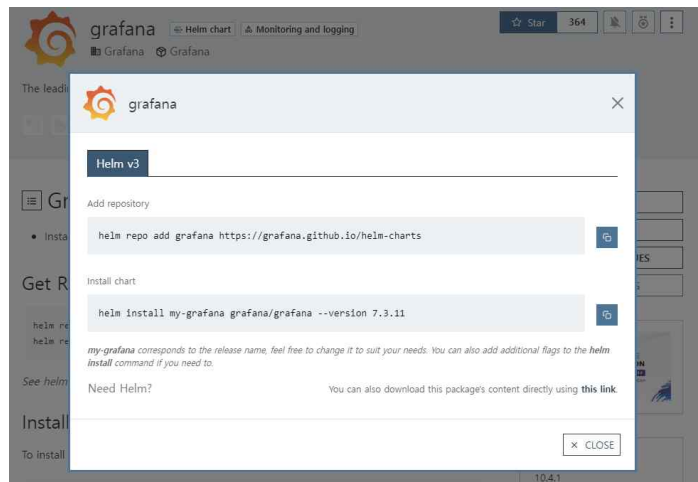
The image shows a terminal window and a web browser. The terminal window displays the command `minikube service prometheus-server-ext` and its output, which is a table showing the service details.

NAMESPACE	NAME	TARGET PORT	URL
default	prometheus-server-ext	80	http://192.168.58.2:32753

The web browser shows the Prometheus UI at the address `192.168.58.2:32753/graph?g0.expr=&g0.tab=1&g0.display_mode=lines&g0.show_exemplars=0&g0.expr=&g0.tab=1&g0.display_mode=lines&g0.show_exemplars=0`. The UI includes a search bar, a query input field, and a graph area.

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-cx19x            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-cx19x           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 비밀번호 변경

kubectl exec --namespace default -it pod명 -- /bin/bash #pod 접속

grafana-cli admin reset-admin-password Tiber01! #Tiber01!는 비밀번호

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

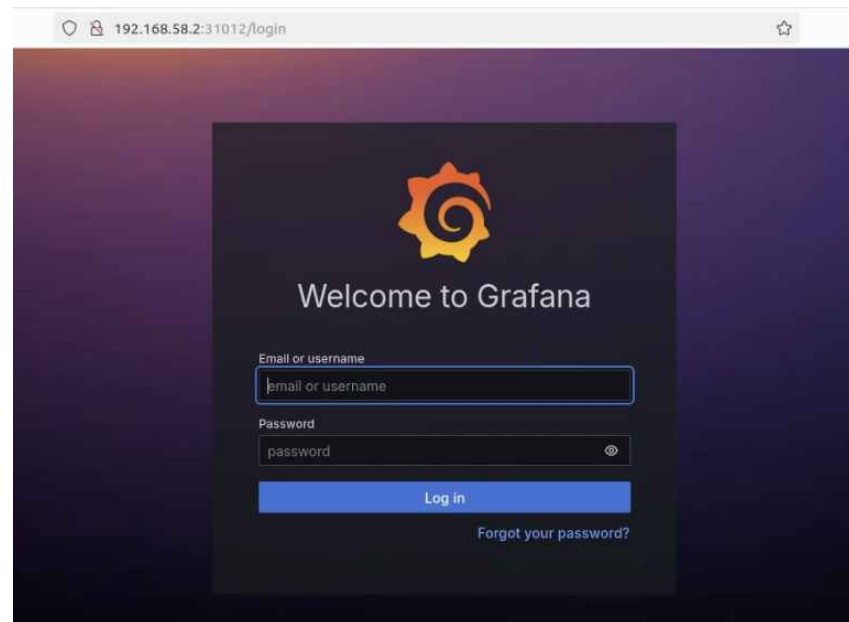
Helm을 이용한 모니터링 시스템 만들기

- 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...
```



- ```

ubuntu@ubuntu-VirtualBox:~$ he 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
 ## 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_LB_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'> Optional: fa
 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
```



# TABA\_DB/SQL실습4

# Apache Kafka


## ■ Apache Kafka


- 비즈니스 소셜 네트워크 링크드인(LinkedIn)에서 개발
- 데이터 파이프라인, 스트리밍 분석, 데이터 통합에 사용하는 오픈소스 분산 이벤트 스트리밍 플랫폼
- Publish - Subscribe 시스템으로 데이터가 이동하는 대량의 파이프라인을 구성
- Fortune 100개 기업 중 80% 이상이 Kafka를 사용


### APACHE KAFKA


More than **80% of all Fortune 100 companies** trust, and use Kafka.

Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.

  
**10 OUT OF 10**  
MANUFACTURING

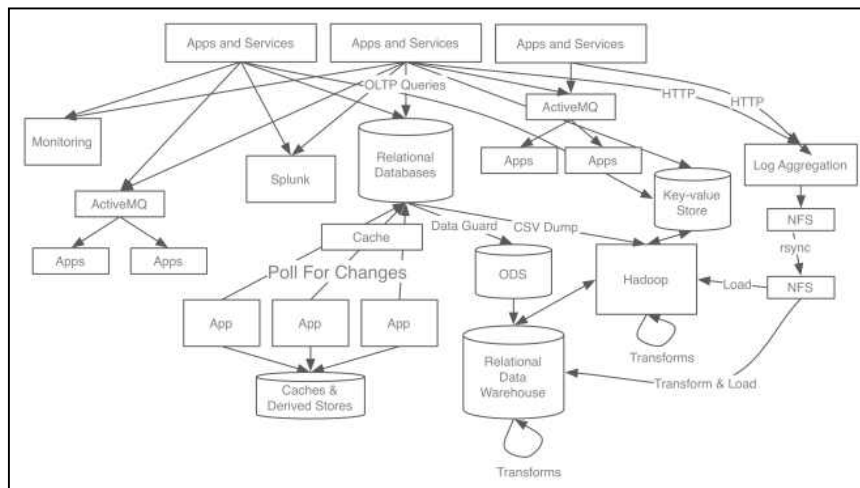
  
**7 OUT OF 10**  
BANKS

  
**10 OUT OF 10**  
INSURANCE

  
**8 OUT OF 10**  
TELECOM

# Apache Kafka

## ■ Before Kafka



## ■ 시스템 복잡도 증가

- 데이터 흐름 파악 및 시스템 관리 어려움
- 특정 부분 장애 발생 시 조치 시간 증가(연결된 앱을 체크해야함)
- HW 및 SW 업그레이드 시 관리 포인트 및 작업시간 증가

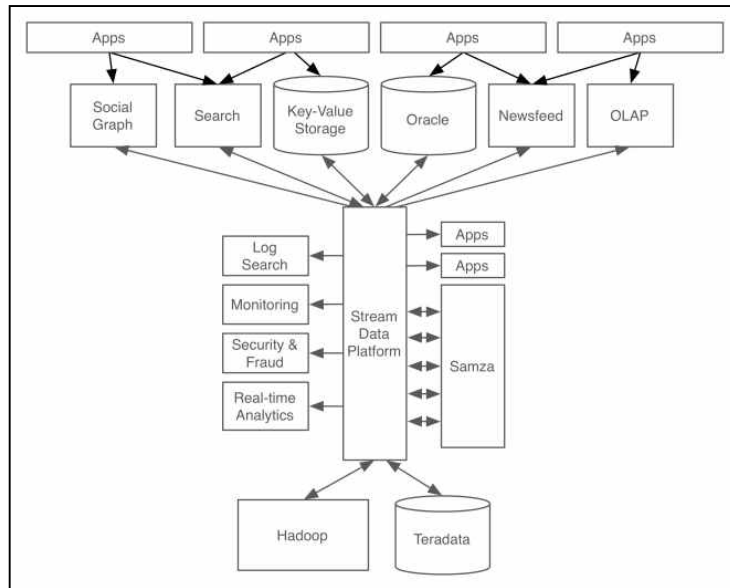
## ■ 데이터 파이프라인 관리 어려움

- 각 앱마다 시스템 간 별도의 파이프라인 존재
- 파이프라인마다 데이터 포맷 및 처리 방식이 다름
- 새로운 파이프라인 확장이 어려움
- 데이터 불일치 가능성



# Apache Kafka

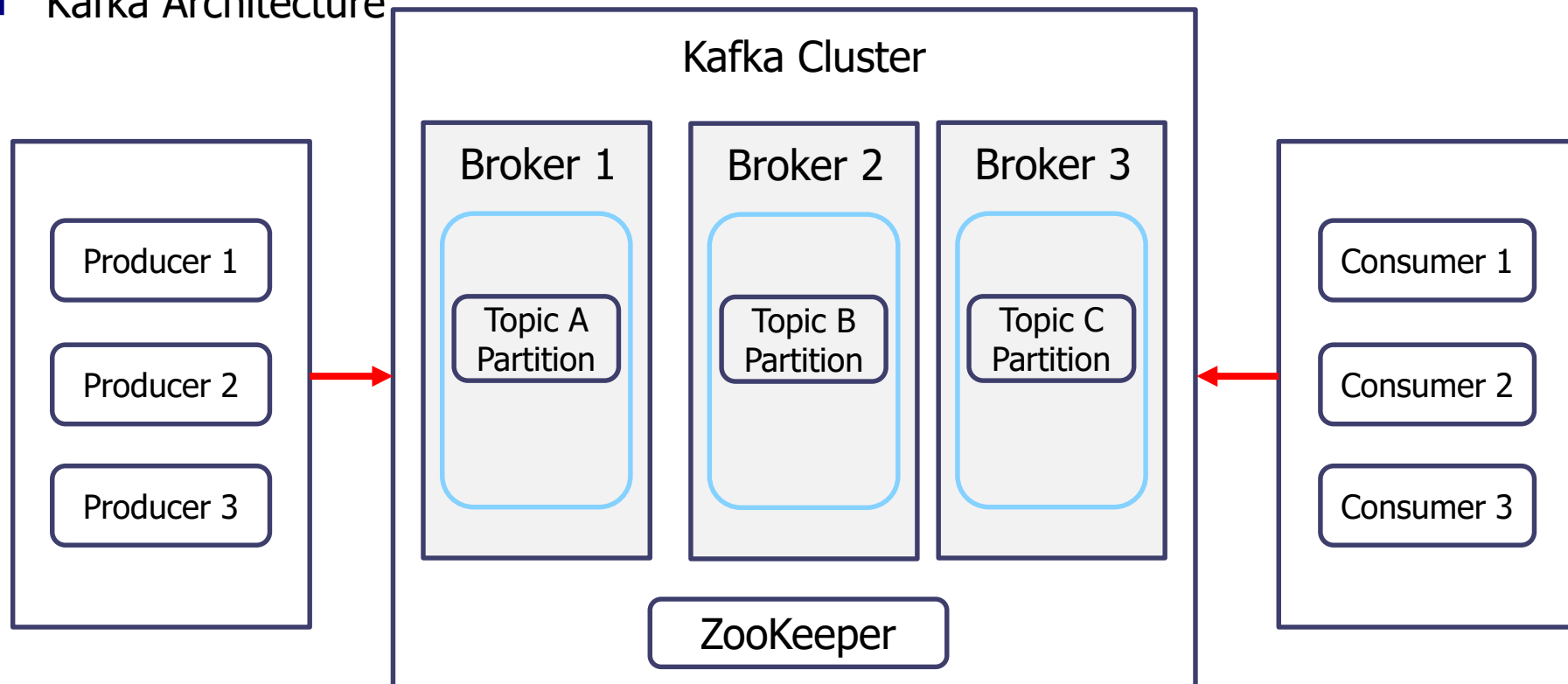
## ■ After Kafka



- 모든 이벤트/ 데이터 흐름을 중앙에서 관리
- 새로운 서비스 및 시스템이 추가되어도 **Kafka**에서 제공하는 표준 포맷으로 연결
- 개발자는 각 서비스 연결이 아닌, 서비스들의 비즈니스 로직에 집중 가능

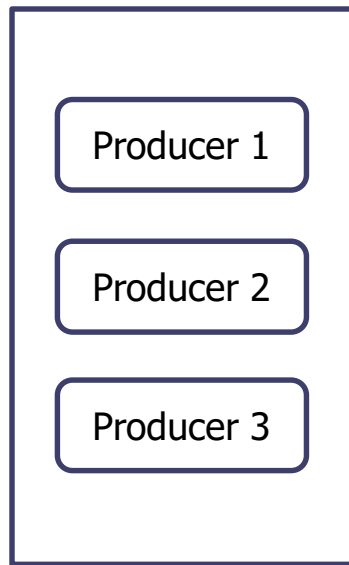
# Apache Kafka

## ■ Kafka Architecture





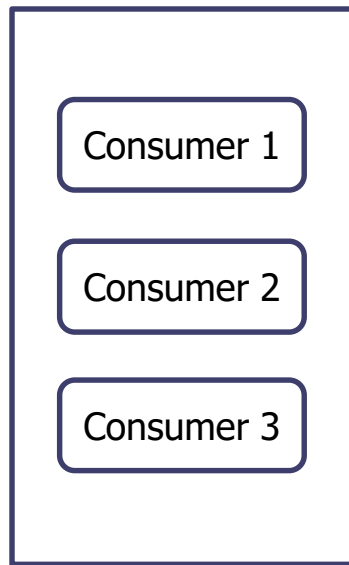
# Apache Kafka



## ■ Producer

- 메시지를 만들어서 카프카 클러스터에 전송
- 메시지 전송 시 **Batch** 처리 가능
- **Key**값을 지정해 특정 파티션으로만 전송 가능

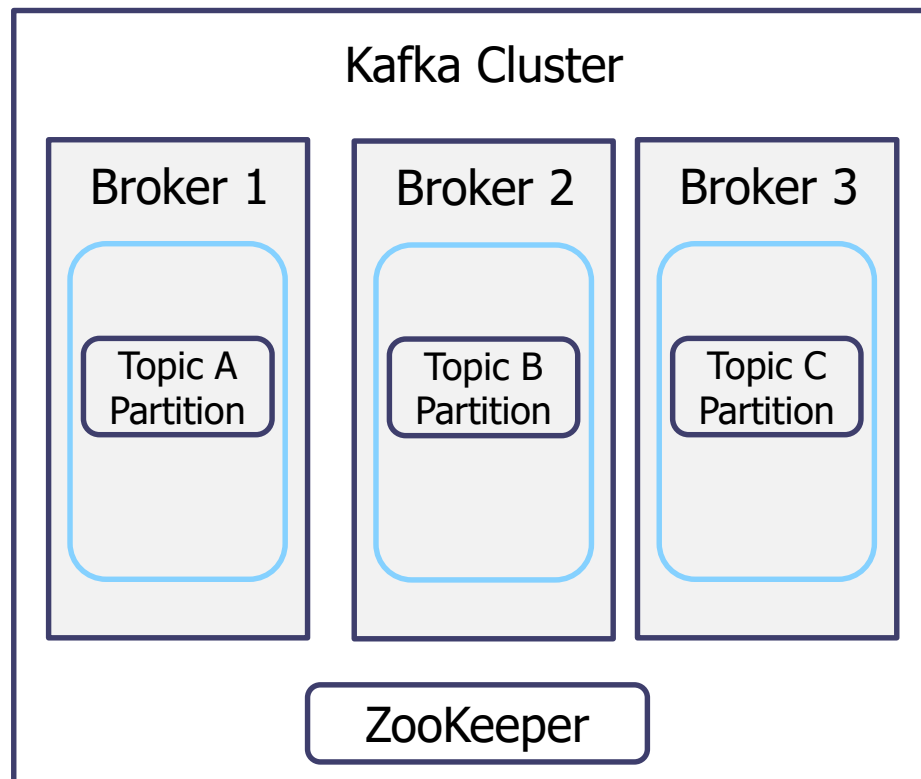
# Apache Kafka



## ■ Consumer

- 카프카 클러스터에서 메시지를 읽고 처리
- 한 개의 컨슈머는 여러 개의 토픽을 처리할 수 있음
- 한 번 저장된 메시지를 여러 번 소비 가능
- 컨슈머는 컨슈머 그룹에 속함
  - 어떤 컨슈머가 다운된다면, 다른 컨슈머가 해당 파티션의 메시지를 다시 구독한다.
  - **offset** 정보를 그룹간 공유하기 때문에 다운되기 전 마지막으로 읽었던 메시지 위치로부터 시작함

# Apache Kafka



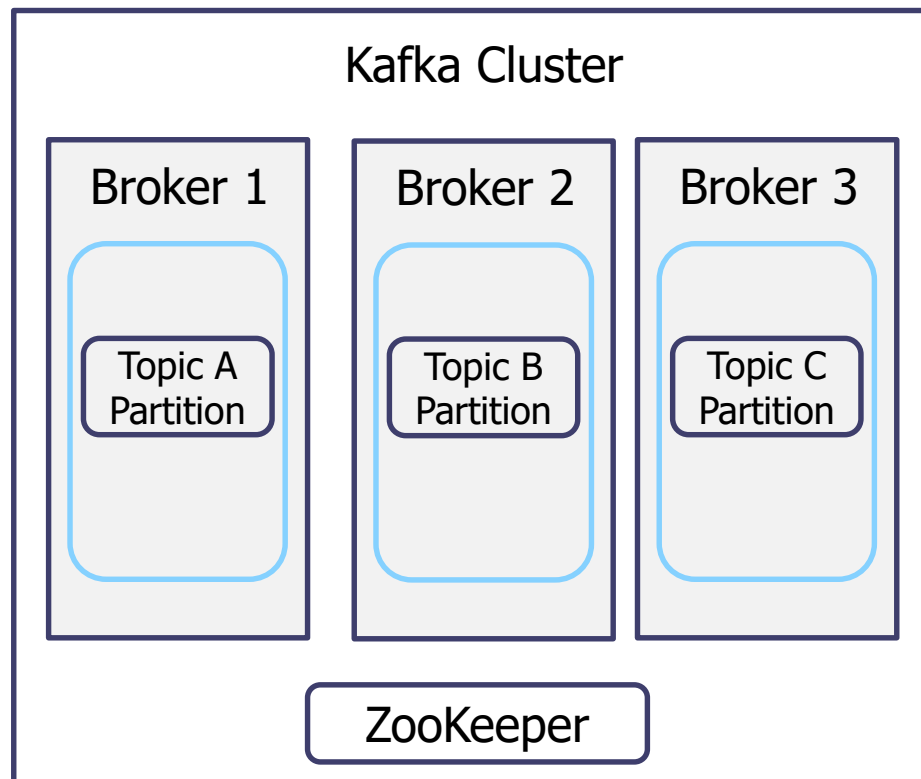
## ■ Topic

- 각각의 메시지를 목적에 맞게 구분
- 메시지를 전송 및 소비할 때 토픽을 반드시 입력
- 컨슈머는 자신이 담당하는 토픽의 메시지를 처리
- 한 개의 토픽은 한 개 이상의 파티션으로 구성

## ■ Partition

- 분산 처리를 위해 사용
- 토픽 생성 시 파티션 개수 지정 가능
- 파티션 내부에 메시지는 **offset**으로 구분
- 파티션이 많으면 처리량이 좋지만 장애 복구 시간 증가

# Apache Kafka



## ■ Offset

- 컨슈머에서 메시지를 어디까지 읽었는지 저장하는 값
- 컨슈머 그룹의 컨슈머들은 각각의 파티션에 자신이 가져간 메시지의 위치 정보(**offset**)을 기록
- 컨슈머 장애 발생 후 복구되면 마지막으로 읽었던 위치에서 다시 읽음

## ■ Broker

- 실행된 카프카 서버
- 프로시저와 컨슈머는 별도의 앱으로 구성
- 브로커는 **Kafka Cluster** 내부 존재
- 저장하고 관리하는 역할 수행

## ■ ZooKeeper

- 분산 앱 관리를 위한 코디네이션 시스템



# Install Kafka on minikube

## ■ Strimz Operator

- Strimz 는 쿠버네티스 환경에서 kafka를 운영하기 위해 만들어진 operator
- 간단하게 카프카 클러스터, 구성요소 배포 및 관리, 설정, 브로커 관리, 토픽, 유저 생성 등 가능
- <https://strimzi.io/>
- Cluster Operator : Apache Kafka cluster, Kafka Connect 등 관리
- Entity Operator : 토픽 및 사용자로 구성



# Install Kafka on minikube

- Install Kafka

권장하는 메모리 사양

```
minikube start --memory=4096
```

- kafka 네임스페이스 생성

```
kubectl create namespace kafka
```

- [strimzi.io](https://strimzi.io)에서 다운받는 ClusterRoles 및 ClusterRoleBindings용 YAML에는 myproject라는 네임스페이스가 기본

- 네임스페이스를 kafka로 사용하도록 변경.

```
kubectl create -f 'https://strimzi.io/install/latest?namespace=kafka' -n kafka
```



# Install Kafka on minikube

- 생성된 pod 및 CRD 확인
- CRD(Custom Resources) :Kafka, 토픽에 사용할 사용자 정의 리소스

kubectl get crd -n kafka

kubectl get all -n kafka

```
ubuntu@ubuntu-VirtualBox:~$ kubectl get all -n kafka
NAME READY STATUS RESTARTS AGE
pod/strimzi-cluster-operator-b9c59999f-2bdlr 1/1 Running 0 59s

NAME READY UP-TO-DATE AVAILABLE AGE
deployment.apps/strimzi-cluster-operator 1/1 1 1 59s

NAME DESIRED CURRENT READY AGE
replicaset.apps/strimzi-cluster-operator-b9c59999f 1 1 1 59s
```

# Install Kafka on minikube

## ■ Apache Kafka cluster 생성

kubectl apply -f <https://strimzi.io/examples/latest/kafka/kraft/kafka-single-node.yaml> -n kafka

위 YAML을 확인하려면 아래 링크로 접속하면 됨

<https://github.com/strimzi/strimzi-kafka-operator/blob/main/examples/kafka/kraft/kafka-single-node.yaml>

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaNodePool
metadata:
 name: dual-role
 labels:
 strimzi.io/cluster: my-cluster
spec:
 replicas: 1
 roles:
 - controller
 - broker
 storage:
 type: jbod
 volumes:
 - id: 0
 type: persistent-claim
 size: 100Gi
 deleteClaim: false
 kraftMetadata: shared

```

controller와 broker가 생성되는  
것을 확인

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
 name: my-cluster
 annotations:
 strimzi.io/node-pools: enabled
 strimzi.io/kraft: enabled
spec:
 kafka:
 version: 3.7.0
 metadataVersion: 3.7-IV4
 listeners:
 - name: plain
 port: 9092
 type: internal
 tls: false
 - name: tls
 port: 9093
 type: internal
 tls: true
 config:
 offsets.topic.replication.factor: 1
 transaction.state.log.replication.factor: 1
 transaction.state.log.min.isr: 1
 default.replication.factor: 1
 min.insync.replicas: 1
```

listener의 정보 확인

plain(암호화되지 않고 전송 또는 저장된 데이터를 의미)

9092로 접속하면 통신이 가능

# Install Kafka on minikube

## ■ 생성된 pod 확인

kubectl get all -n kafka

```
ubuntu@ubuntu-VirtualBox:~$ kubectl get all -n kafka
NAME READY STATUS RESTARTS AGE
pod/my-cluster-dual-role-0 1/1 Running 0 39m
pod/my-cluster-entity-operator-d6c5c645-tkksn 2/2 Running 0 38m
pod/strimzi-cluster-operator-b9c59999f-2bd1r 1/1 Running 0 43m

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
service/my-cluster-kafka-bootstrap ClusterIP 10.97.28.196 <none> 9091/TCP,9092/TCP,9093/TCP 39m
service/my-cluster-kafka-brokers ClusterIP None <none> 9090/TCP,9091/TCP,8443/TCP,9092/TCP,9093/TCP 39m

NAME READY UP-TO-DATE AVAILABLE AGE
deployment.apps/my-cluster-entity-operator 1/1 1 1 38m
deployment.apps/strimzi-cluster-operator 1/1 1 1 43m

NAME DESIRED CURRENT READY AGE
replicaset.apps/my-cluster-entity-operator-d6c5c645 1 1 1 38m
replicaset.apps/strimzi-cluster-operator-b9c59999f 1 1 1 43m
```



# Send and receive msg

- 두 개의 putty 실행
- 1번 putty에는 Producer , 2번 putty에는 Consumer
- Producer 생성

```
kubectl -n kafka run kafka-producer -ti --image=quay.io/strimzi/kafka:0.41.0-kafka-3.7.0 --rm=true ₩
--restart=Never -- bin/kafka-console-producer.sh --bootstrap-server my-cluster-kafka-bootstrap:9092 ₩
--topic my-topic
```

- Consumer 생성

```
kubectl -n kafka run kafka-consumer -ti --image=quay.io/strimzi/kafka:0.41.0-kafka-3.7.0 --rm=true ₩
--restart=Never -- bin/kafka-console-consumer.sh --bootstrap-server my-cluster-kafka-bootstrap:9092 ₩
--topic my-topic --from-beginning
```

# Send and receive msg

- 프로시저에서 메시지를 보내면 컨슈머에서 메시지 확인 가능
- 단, topic 및 port가 동일해야함

```
ubuntu@ubuntu-VirtualBox:~$ kubectl -n kafka run kafka-producer -ti --image=quay.io/strimzi/kafka:0.41.0-kafka-3.7.0 --rm=true --restart=Never -- bin/kafka-console-producer.sh --bootstrap-server=cluster-kafka-bootstrap:9092 --topic my-topic
If you don't see a command prompt, try pressing enter.
>dankook
>
```

```
ubuntu@ubuntu-VirtualBox:~$ kubectl -n kafka run kafka-consumer -ti --image=quay.io/strimzi/kafka:0.41.0-kafka-3.7.0 --rm=true --restart=Never -- bin/kafka-console-consumer.sh --bootstrap-server=cluster-kafka-bootstrap:9092 --topic my-topic --from-beginning
If you don't see a command prompt, try pressing enter.
hello
hello
dankook
```



# Kafka Connect

- Kafka Connect(커넥트)
  - Kafka의 컴포넌트 중 하나
  - 커넥터(Connector)를 동작하도록 실행해주는 프로세스
  - 커넥터를 사용하기 위해서는 커넥트가 실행되어야 함.
  - 커넥트를 이용해 **kafka**로부터 데이터를 보내거나, 데이터를 가지고 올 수 있음.
  - 이미 만들어진 기존 커넥트를 활용할 수 있고, 운영 환경에서 변경 가능



# Kafka Connect

## ■ Kafka Connect

- 단일 실행 모드 커넥트
  - 간단한 데이터 파이프라인을 구성하거나 개발용으로 사용
- 분산 모드 커넥트
  - 실질적으로 상용화에 활용하려면 분산 모드 커넥트로 구성해서 운영
  - 여러 개의 커넥트를 묶어서 운영하는 방식
  - 클러스터로 묶은 커넥트는 일부 커넥트에 장애가 발생해도 파이프라인을 자연스럽게 장애 조치해서 실행중인 커넥트를 처리할 수 있게 함.



# Kafka Connect

## ■ Kafka Connector(커넥터)

- 커넥터는 실질적으로 데이터를 처리하는 코드가 담긴 **JAR** 패키지
- 따라서 커넥터 안에는 파이프라인에 필요한 메서드, 설정 등 코드가 있음
- 싱크 커넥터(Sink Connector)
  - 데이터를 싱크한다는 뜻으로, 특정 토픽에 있는 데이터를 오라클, MySQL 등 특정 DB로 보내는 역할을 하는 커넥터 (컨슈머와 같은 역할)
- 소스 커넥터(Source Connector)
  - DB로부터 데이터를 가져와서 토픽에 넣는 역할 (프로듀서와 같은 역할)



# Kafka Connect 예제

- 아래와 같은 환경 구축
- MySQL - Kafka - PostgreSQL을 커넥터로 연결
- MySQL에서 데이터를 생성하면 데이터 PostgreSQL에서 받기



# Kafka Connect 예제

## ■ MySQL on Kubernetes

mysql.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: mysql
 namespace: kafka
spec:
 selector:
 matchLabels:
 app: mysql
 strategy:
 type: Recreate
 template:
 metadata:
 labels:
 app: mysql
 spec:
 containers:
 - image: mysql:5.6
 name: mysql
 env:
 - name: MYSQL_ROOT_PASSWORD
 value: Dankook1!
 - name: LC_ALL
 value: C.UTF-8
 ports:
 - containerPort: 3306
 name: mysql
 volumeMounts:
 - name: mysql-initdb
 mountPath: /docker-entrypoint-initdb.d
 volumes:
 - name: mysql-initdb
 configMap:
 name: mysql-config
 items:
 - key: initdb.sql
 path: initdb.sql
```

mysql-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
 name: mysql
 namespace: kafka
spec:
 ports:
 - port: 3306
 selector:
 app: mysql
```

MySQL 환경변수로 비밀번호 설정  
현재는 Dankook1!로 설정되어있음.

# Kafka Connect 예제

## ■ MySQL on Kubernetes

mysql-config.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
 name: mysql-config
 namespace: kafka
data:
 initdb.sql: |
 SET GLOBAL binlog_format = 'ROW';
 SET GLOBAL server_id = 184054;
 CREATE DATABASE test;
 use test;
 CREATE TABLE example (id INT(16) NOT NULL, name
 VARCHAR(32));
 INSERT INTO example VALUES (111, 'connecttest');
```

initdb.sql은 MySQL이 처음 실행될 때 수행되는 명령어

1. test 데이터베이스 생성
2. test 데이터베이스 사용
3. example 테이블 생성 및 데이터 생성

# Kafka Connect 예제

## ■ PostgreSQL on Kubernetes

postgresql.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: postgres
 namespace: kafka
spec:
 selector:
 matchLabels:
 app: postgres
 replicas: 1
 template:
 metadata:
 labels:
 app: postgres
 spec:
 containers:
 - name: postgres
 image: postgres:9.6.2-alpine
 ports:
 - containerPort: 5432
 env:
 - name: TZ
 value: 'Asia/Seoul'
 - name: POSTGRES_USER
 value: postgres
 - name: POSTGRES_PASSWORD
 value: Dankook1!
```

postgresql-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
 name: postgres-service
 namespace: kafka
 labels:
 app: postgres
spec:
 type: ClusterIP
 ports:
 - port: 5432
 selector:
 app: postgres
```

PostgreSQL의 환경 변수 설정  
패스워드는 Dankook1!

# Kafka Connect 예제

- 생성된 데이터베이스 확인

kubectl get po -n kafka

```
oem@ubuntu-server:~/strimzi-kafka-example$ kubectl get po -n kafka
NAME READY STATUS RESTARTS AGE
my-cluster-dual-role-0 1/1 Running 0 89m
my-cluster-entity-operator-d6c5c645-lc8xx 2/2 Running 0 89m
mysql-5f87695797-dqdcn 1/1 Running 0 19m
postgres-f9559496c-6pgzt 1/1 Running 0 18m
strimzi-cluster-operator-b9c59999f-667ng 1/1 Running 0 90m
```



# Kafka Connect 예제

- 스키마 레지스트리(Schema Registry)

- 카프카 클라이언트 사이에서 메시지의 스키마를 저장, 관리하는 웹 어플리케이션
- 데이터의 호환성 유지 및 관리를 위해 개발
- **Producer, Consumer**가 많을 수 있는 분산 시스템 환경에서 데이터에 대한 호환성을 유지
- **strimzi**에서는 제공하지 않기 때문에 **confluent**에서 제공하는 스키마 레지스트리 사용

# Kafka Connect 예제

## ■ Schema\_registry

### schema-registry.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: schema-registry
 namespace: kafka
spec:
 selector:
 matchLabels:
 app: schema-registry
 strategy:
 type: Recreate
 template:
 metadata:
 labels:
 app: schema-registry
 spec:
 containers:
 - name: my-cluster-schema-registry
 image: confluentinc/cp-schema-registry:7.0.1
 ports:
 - containerPort: 8081
 env:
 - name: SCHEMA_REGISTRY_KAFKASTORE_BOOTSTRAP_SERVERS
 value: my-cluster-kafka-bootstrap:9092
 - name: SCHEMA_REGISTRY_HOST_NAME
 valueFrom:
 fieldRef:
 fieldPath: status.podIP
 - name: SCHEMA_REGISTRY_LISTENERS
 value: http://0.0.0.0:8081
 - name: SCHEMA_REGISTRY_KAFKASTORE_SECURITY_PROTOCOL
 value: PLAINTEXT
```

### schema-registry-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
 name: schema-registry
 namespace: kafka
spec:
 ports:
 - port: 8081
 clusterIP: None
 selector:
 app: schema-registry
```

# Kafka Connect 예제

## ■ Kafka connect

kafka-connect.yaml

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
metadata:
 name: my-connect-cluster3
 namespace: kafka
 annotations:
 # use-connector-resources configures this KafkaConnect
 # to use KafkaConnector resources to avoid
 # needing to call the Connect REST API directly
 strimzi.io/use-connector-resources: "true"
spec:
 image: aldfkaks/kafka-connect-jdbc:v1.0
 replicas: 1
 bootstrapServers: my-cluster-kafka-bootstrap:9092
 config:
 config.storage.replication.factor: 1
 offset.storage.replication.factor: 1
 status.storage.replication.factor: 1
 config.providers: file
 config.providers.file.class: org.apache.kafka.common.config.provider.FileConfigProvider
 key.converter: io.confluent.connect.avro.AvroConverter
 key.converter.schema.registry.url: http://schema-registry.kafka.svc.cluster.local:8081
 value.converter: io.confluent.connect.avro.AvroConverter
 value.converter.schema.registry.url: http://schema-registry.kafka.svc.cluster.local:8081
```

spec.image : 사용할 컨넥트 이미지

spec.bootstrapServers : 카프카 클러스터 주소



# Kafka Connect 예제

## ■ MySQL Source connector

mysql-source-connector.yaml

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnector
metadata:
 name: jdbc-mysql-source-connector
 namespace: kafka
 labels:
 strimzi.io/cluster: my-connect-cluster3
spec:
 class: io.confluent.connect.jdbc.JdbcSourceConnector
 tasksMax: 1
 config:
 mode: "incrementing"
 incrementing.column.name: "id"
 connection.url: "jdbc:mysql://mysql.kafka.svc.cluster.local:3306/test"
 connection.user: "root"
 connection.password: "Dankook1!"
 table.whitelist: "example"
```

config.connection.url : 연결할 데이터베이스(test  
데이터베이스를 연결)

config.connection.user : root 유저

config.connection.password : 비밀번호

config.table.whitelist: 어떤 테이블에 대해 읽을지 필터링

# Kafka Connect 예제

## ■ MySQL Source connector

postgres-sink-connector.yaml

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnector
metadata:
 name: postgres-connector
 namespace: kafka
 labels:
 strimzi.io/cluster: my-connect-cluster3
spec:
 class: io.confluent.connect.jdbc.JdbcSinkConnector
 tasksMax: 1
 config:
 topics: "example"
 connection.url: "jdbc:postgresql://postgres-service.kafka.svc.cluster.local:5432/postgres"
 connection.user: "postgres"
 connection.password: "Dankook1!"
 auto.create: "true"
```

config.topics : 구독할 테이블  
config.connection.url: 저장 될 PostgreSQL의  
데이터베이스  
config.connection.user : root 유저  
config.connection.password : 패스워드  
config.table.whitelist:어떤 테이블에 대해 읽을지 필터링

# Kafka Connect 예제

- 각 데이터 베이스 접속하기

kubectl get po -n kafka

```
oem@ubuntu-server:~/strimzi-kafka-example$ kubectl get po -n kafka
NAME READY STATUS RESTARTS AGE
my-cluster-dual-role-0 1/1 Running 0 113m
my-cluster-entity-operator-d6c5c645-1c8xx 2/2 Running 0 112m
my-connect-cluster3-connect-0 1/1 Running 0 3m39s
mysql-6c46664b98-wccqv 1/1 Running 0 3m21s
postgres-f9559496c-tdqnb 1/1 Running 0 3m18s
strimzi-cluster-operator-b9c59999f-667ng 1/1 Running 0 113m
```

# Kafka Connect 예제

## ■ 각 데이터 베이스 접속하기

### MySQL

- `kubectl exec -it mysql-6c46664b98-wccqv -n kafka -- mysql -u root -p`
- 각 개인마다 pod명은 다름

```
dem@ubuntu-server:~/strimzi-kafka-example$ kubectl exec -it mysql-6c46664b98-wccqv -n kafka -- mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.6.51 MySQL Community Server (GPL)

Copyright (c) 2000, 2021, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

# Kafka Connect 예제

## ■ 각 데이터 베이스 접속하기

생성된 데이터베이스, 테이블 및 데이터 조회

show databases;

use test;

select \* from example;

exit; #데이터베이스 나가기

```
oem@ubuntu-server:~$ kubectl exec -it mysql-6987bc7f6b-jbf4p -n kafka -- mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.6.51 MySQL Community Server (GPL)

Copyright (c) 2000, 2021, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from example;
+-----+
| id | name |
+-----+
| 111 | connecttest |
+-----+
1 row in set (0.00 sec)
```

# Kafka Connect 예제

## ■ 각 데이터 베이스 접속하기

### PostgreSQL

- `kubectl exec -it postgres-f9559496c -n kafka -- psql -U postgres`
- `\dt`
- `\q` #데이터베이스 종료

```
oem@ubuntu-server:~$ kubectl exec -it postgres-75fcff5cc5-x2861 -n kafka -- psql -U postgres
psql (9.6.2)
Type "help" for help.

postgres=# \dt
 List of relations
 Schema | Name | Type | Owner
-----+-----+-----+-----
 public | example | table | postgres
(1 row)
```



# 크롤링 데이터 DB에 저장하기

- Python에서 크롤링 데이터 MySQL에 저장하기

- Python Pod 생성

python.yaml

```
apiVersion: v1
kind: Pod
metadata:
 name: python
 namespace: kafka
 labels:
 app: python
spec:
 containers:
 - name: python
 image: python
 command: ["sleep", "infinity"]
```

# 크롤링 데이터 DB에 저장하기

- MySQL에 새로운 DB 및 유저 생성

- `kubect exec -it mysql-6c46664b98-wccqv -n kafka -- mysql -u root -p`

`CREATE DATABASE crawl_db DEFAULT CHARACTER SET utf8; #데이터베이스 생성`

`CREATE USER crawl_user IDENTIFIED BY 'Dankook1!'; #유저명 :crawl_user, 패스워드 Dankook1!`

`GRANT ALL ON crawl_db.* TO crawl_user; crawl_db 데이터베이스의 권한을 crawl_crawl_user에게 주기`

```
mysql> CREATE DATABASE crawl_db DEFAULT CHARACTER SET utf8;
Query OK, 1 row affected (0.00 sec)

mysql> CREATE USER crawl_user IDENTIFIED BY 'Dankook1!';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL ON crawl_db.* TO crawl_user;
Query OK, 0 rows affected (0.00 sec)
```



# 크롤링 데이터 DB에 저장하기

- Python Pod로 접속 후 파이썬 패키지 설치

- kubectl get po -n kafka

```
oem@ubuntu-server:~$ kubectl get po -n kafka
NAME READY STATUS RESTARTS AGE
my-cluster-dual-role-0 1/1 Running 0 26h
my-cluster-entity-operator-d6c5c645-1c8xx 2/2 Running 0 26h
my-connect-cluster3-connect-0 1/1 Running 0 24h
mysql-6987bc7f6b-jbf4p 1/1 Running 0 15h
postgres-75fcff5cc5-x286l 1/1 Running 0 14h
python 1/1 Running 0 6h43m
schema-registry-745c585464-qqr8 1/1 Running 0 14h
strimzi-cluster-operator-b9c59999f-667ng 1/1 Running 0 26h
```

kubectl exec -it python -n kafka -- /bin/bash #생성한 Python pod 접속

apt update && apt install vim #pod 접속 후 업데이트 및 vim 편집기 설치

pip install mysqlclient #pip을 이용해 mysqlclient 패키지 설치

# 크롤링 데이터 DB에 저장하기

## ■ Python - MySQL 연결 확인 테이블 생성

vi mysqltest.py

```
import MySQLdb
```

```
conn = MySQLdb.connect(
 user="crawl_user",
 passwd="Dankook1!",
 host="10.96.61.228",
 db="crawl_db")
```

```
print(type(conn))
<class 'MySQLdb.connections.Connection'>
cursor = conn.cursor()
print(type(cursor))
<class 'MySQLdb.cursors.Cursor'>
cursor.execute("CREATE TABLE test (id int)")
cursor.execute("insert into test values(123)")
```

```
conn.commit()
```

host 확인 하는 방법  
kubectl get svc -n kafka

```
dem@ubuntu-server:~$ kubectl get svc -n kafka
```

| NAME                            | TYPE      | CLUSTER-IP     | EXTERNAL-IP | PORT(S)            |
|---------------------------------|-----------|----------------|-------------|--------------------|
| my-cluster-kafka-bootstrap      | ClusterIP | 10.107.39.6    | <none>      | 9091/TCP, 9092/TCP |
| my-cluster-kafka-brokers        | ClusterIP | None           | <none>      | 9090/TCP, 9092/TCP |
| my-connect-cluster3-connect     | ClusterIP | None           | <none>      | 8083/TCP           |
| my-connect-cluster3-connect-api | ClusterIP | 10.107.230.207 | <none>      | 8083/TCP           |
| mysql                           | ClusterIP | 10.100.111.92  | <none>      | 3306/TCP           |
| postgres-service                | ClusterIP | 10.109.191.109 | <none>      | 5432/TCP           |
| schema-registry                 | ClusterIP | None           | <none>      | 8081/TCP           |

```
mysql> desc test;
```

| Field | Type    | Null | Key | Default | Extra |
|-------|---------|------|-----|---------|-------|
| id    | int(11) | YES  |     | NULL    |       |

```
1 row in set (0.00 sec)
```

# 크롤링 데이터 DB에 저장하기

## ■ vi melon.py # (1)

pip install requests bs4 # 패키지 설치

```
import MySQLdb
import requests
from bs4 import BeautifulSoup

if __name__ == "__main__":
 RANK = 100 # 멜론 차트 순위가 1 ~ 100위까지 있음

 header = {
 'User-Agent': 'Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko'
 }
 req = requests.get('https://www.melon.com/chart/week/index.htm',
 headers=header) # 주간 차트를 크롤링 할 것임
 html = req.text
 parse = BeautifulSoup(html, 'html.parser')

 titles = parse.find_all("div", {"class": "ellipsis rank01"})
 singers = parse.find_all("div", {"class": "ellipsis rank02"})

 title = []
 singer = []

 for t in titles:
 title.append(t.find('a').text)

 for s in singers:
 singer.append(s.find('span', {"class": "checkEllipsis"}).text)
 items = [item for item in zip(title, singer)]
```



# 크롤링 데이터 DB에 저장하기

## ■ vi melon.py # (2)

```
conn = MySQLdb.connect(
 user="crawl_user",
 passwd="Dankook1!",
 host="10.100.111.92",
 db="crawl_db"
 # charset="utf-8"
)
cursor = conn.cursor()
cursor.execute("DROP TABLE IF EXISTS melon")

cursor.execute("CREATE TABLE melon (`rank` int, title text, singer text)")

i = 1

for item in items:
 cursor.execute(
 "INSERT INTO melon (rank, title, singer) VALUES (%s, %s, %s)", (i, item[0], item[1]))
 i += 1

conn.commit()
```

# 크롤링 데이터 확인하기

## ■ MySQL Source Connector

mysql-source-connector.yaml

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnector
metadata:
 name: jdbc-mysql-source-connector
 namespace: kafka
 labels:
 strimzi.io/cluster: my-connect-cluster3
spec:
 class: io.confluent.connect.jdbc.JdbcSourceConnector
 tasksMax: 1
 config:
 mode: "bulk"
 poll.interval.ms: "86400000"
 connection.url: "jdbc:mysql://mysql.kafka.svc.cluster.local:3306/crawl_db"
 connection.user: "crawl_user"
 connection.password: "Dankook1!"
 table.whitelist: "melon"
```

### Connector mode

1. **incrementing** : 특정 칼럼의 증가분만 감지, 기존 행의 수정 및 삭제는 감지되지 않음. 따라서 특정 칼럼을 작성해줘야함.  
mode: "incrementing"  
incrementing.column.name: "id"
2. **bulk** : 데이터를 폴링할 때 마다 전체 테이블 복사. 폴링 시간을 적어줘야함.  
mode: "bulk"  
poll.interval.ms: "86400000"
3. **timestamp** : timestamp형 칼럼일 경우 새 행과 수정된 행을 감지  
mode: "timestamp"  
timestamp.column.name: "date"

# 크롤링 데이터 확인하기

## ■ MySQL Source Connector

postgres-sink-connector.yaml

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnector
metadata:
 name: postgres-connector
 namespace: kafka
 labels:
 strimzi.io/cluster: my-connect-cluster3
spec:
 class: io.confluent.connect.jdbc.JdbcSinkConnector
 tasksMax: 1
 config:
 topics: "melon"
 connection.url: "jdbc:postgresql://postgres-service.kafka.svc.cluster.local:5432/postgres"
 connection.user: "postgres"
 connection.password: "Dankook1!"
 auto.create: "true"
```

# 크롤링 데이터 확인하기

## ■ 커넥터 배포하기

```
kubectl apply -f mysql-source-connector.yaml
kubectl apply -f postgres-sink-connector.yaml
```

```
kubectl get KafkaConnector -n kafka
```

```
oem@ubuntu-server:~/strimzi-kafka-example$ kubectl get KafkaConnector -n kafka
```

| NAME                        | CLUSTER             | CONNECTOR CLASS                               | MAX TASKS | READY |
|-----------------------------|---------------------|-----------------------------------------------|-----------|-------|
| jdbc-mysql-source-connector | my-connect-cluster3 | io.confluent.connect.jdbc.JdbcSourceConnector | 1         | True  |
| postgres-connector          | my-connect-cluster3 | io.confluent.connect.jdbc.JdbcSinkConnector   | 1         | True  |

# 크롤링 데이터 확인하기

## ■ PostgreSQL 접속 및 데이터 확인

kubectrl exec -it postgres-75fcff5cc5-whcxx -n kafka -- psql -U postgres

\dt #스키마 조회

select \* from melon limit 10; #melon 테이블 상위 10개 출력

```
postgres=# \dt
 List of relations
 Schema | Name | Type | Owner
-----+-----+-----+-----
 public | melon | table | postgres
(1 row)

postgres=# select * from melon limit 10;
 rank | title | singer
-----+-----+-----
 1 | Supernova | aespa
 2 | SPOT! (feat. JENNIE) | 지코 (ZICO)
 3 | 헤아 (HEYA) | IVE (아이브)
 4 | Magnetic | 아일릿 (ILLIT)
 5 | 고민종류 | QWER
 6 | 나는 아픈 건 말 질색이니까 | (여자)아이들
 7 | 소나기 | 이클립스 (ECLIPSE)
 8 | 미안해 미워해 사랑해 | Crush
 9 | 첫 만남은 계획대로 되지 않아 | TWS (투어스)
 10 | 전상연 | 이창현
(10 rows)
```