

클라우드 애플리케이션 개발 (Cloud Application Development)

Multi Container 애플리케이션 구성

Wonjo Yoo (wonjo.yoo@oracle.com)
Principal Sales Consultant
Cloud Infrastructure / aPaaS
Oracle Korea

클라우드 애플리케이션 개발(Cloud App. Development)

- 클라우드 환경에서 애플리케이션 개발과 관리

- 강의 목표 : Multi Container 애플리케이션 구성

| | | | | |
|----|--|-----|-------|-------|
| 8 | 가상환경에서의 Web Application 개발 환경 구성 | 유원조 | 10/25 | 10/26 |
| 9 | Container 애플리케이션 개발 개요 및 환경 구성 (Local Docker 환경 구성, Dockerfile 이해) | 김영규 | 11/01 | 11/02 |
| 10 | Single Container 애플리케이션 구성(Git, Docker hub를 이용한 빌드/배포 자동화) | 김영규 | 11/08 | 11/09 |
| 11 | Multi Container 애플리케이션 구성(WEB-WAS-DB로 구성된 Multi-tier 애플리케이션 환경 구성 및 개발) | 유원조 | 11/15 | 11/16 |
| 12 | Opensource 기반의 Multi Container 애플리케이션 구성(Spring/Tomcat/MySQL) | 김영규 | 11/22 | 11/23 |
| 13 | Git, Wercker를 이용한 Container 애플리케이션 구축(node.js-cassandra / node.js-MongoDB) | 유원조 | 11/29 | 11/30 |
| 14 | 애플리케이션 개발 트렌드(Microservices, Polyglot) | 이미남 | 12/06 | 12/07 |
| 15 | 기말고사 | | 12/13 | 12/14 |

클라우드 애플리케이션 개발(Cloud App. Development)

- 클라우드 환경에서 애플리케이션 개발과 관리

- 강의 자료 URL
- <https://cloudnativeapp.slack.com/messages/C7SE712BF/>
- Lab 관련 소스 URL
- <https://github.com/wjyoo/dku>

Lab 4-1

JDBC Connection Pool 사용하기

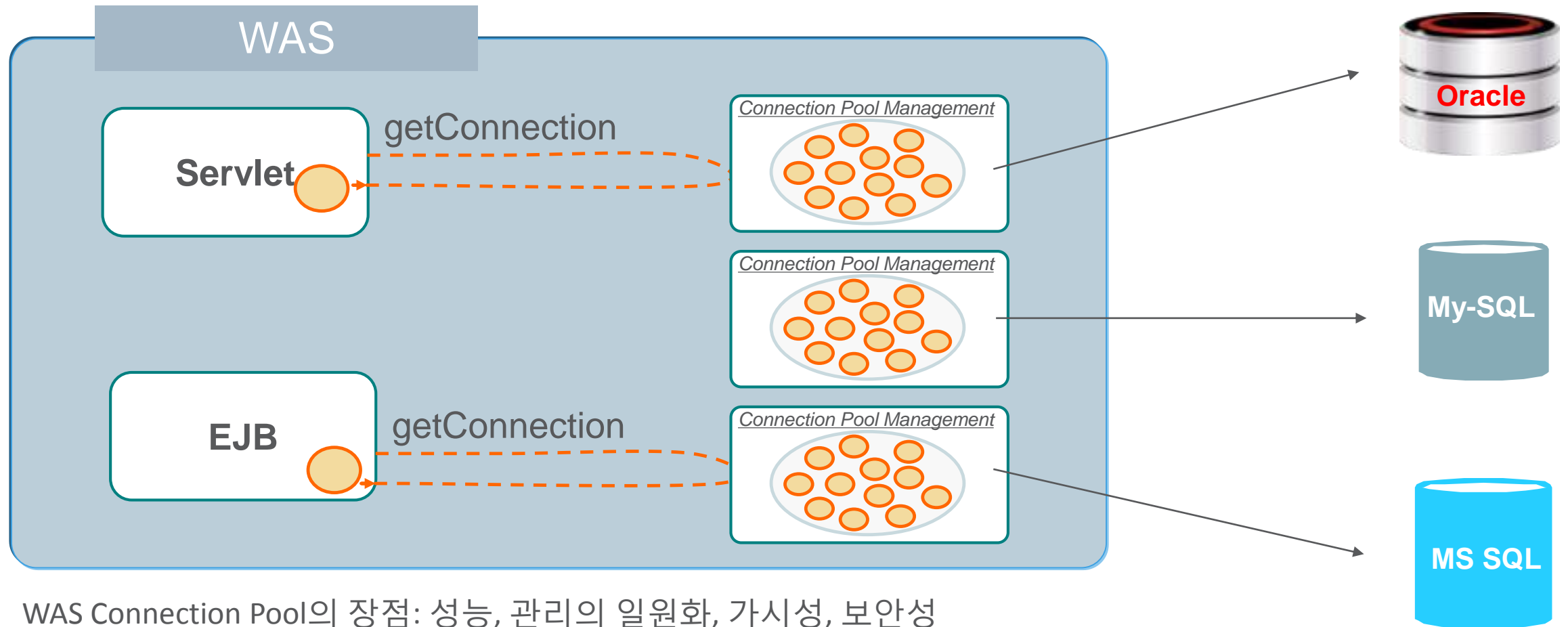
소요시간 : 30분

JDBC (Java Database Connectivity)

4가지 JDBC Driver의 Types

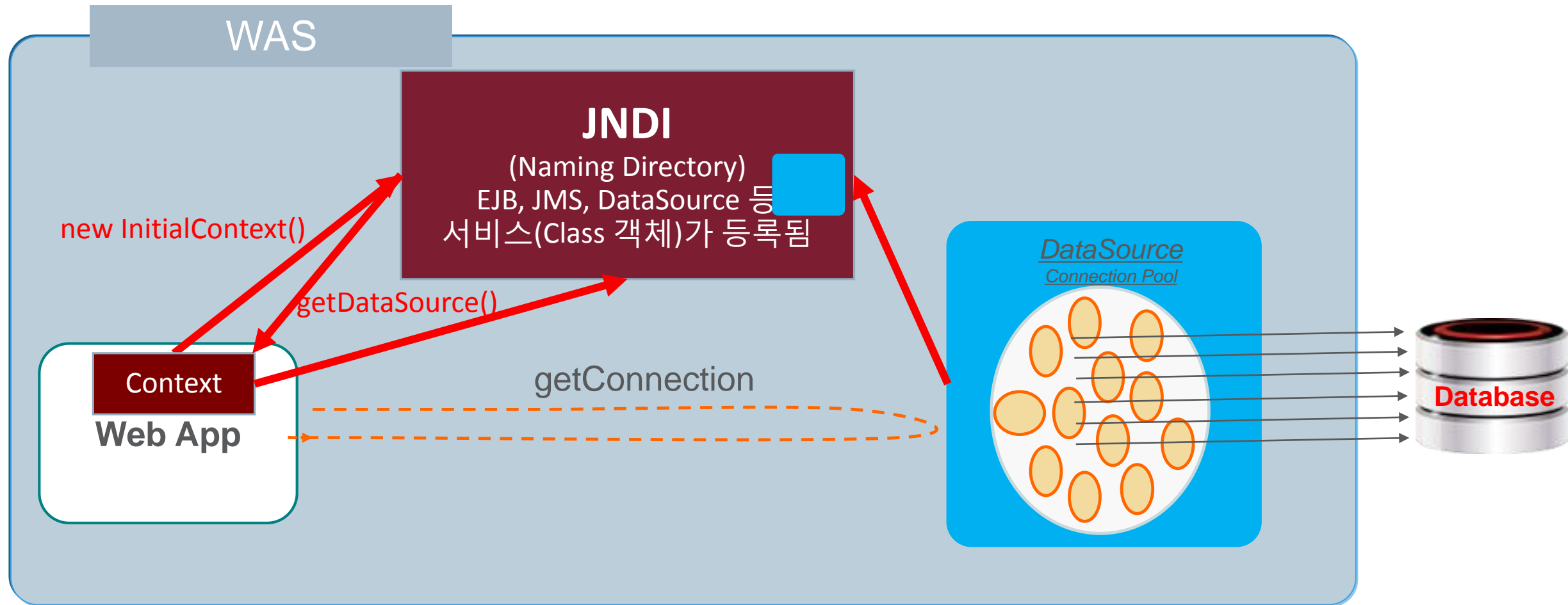
- Type 1: JDBC-ODBC bridge
- Type 2: Java driver → Native DB Driver
(예: Oracle OCI)
- Type 3: Net protocol Driver
- Type 4: 100% 순수 Java Driver

JDBC Connection Pool



WAS Connection Pool의 장점: 성능, 관리의 일원화, 가시성, 보안성

JDBC Connection Pool



Lab 4-1 JDBC Connection Pool 사용하기1

Tomcat - MySQL 연동 구성

1. jdbc파일 복사 -> ~/apache-tomcat-8.5.23/lib

```
cp ~/install/mysql-connector-java-5.1.44/mysql-connector-java-5.1.44-bin.jar ~/apache-tomcat-8.5.23/lib/
```

2. apache-tomcat-8.5.23/conf/context.xml 파일의 <Context> 다음에 추가

```
<Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
    maxTotal="100" maxIdle="30" maxWaitMillis="10000"
    username="root" password="welcome1" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/dku"/>
```

3. Web App 에서 참조하도록 web.xml 에 추가

(위치: apache-tomcat-8.5.23 /webapps/test/WEB-INF/)

```
<web-app>
  <resource-ref>
    <res-ref-name>jdbc/TestDB</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
  </resource-ref>
```


Lab 4-1 JDBC Connection Pool 사용하기2

Tomcat - MySQL 연동 구성

4. 소스에서 수정 (위치: apache-tomcat-8.5.23/webapps/test)

```
cp db.jsp db2.jsp
```

5. Connection 얻는 부분 소스에서 수정

vi db2.jsp 실행 후 아래 코드로 변경

<삭제할 부분>

```
Class.forName("com.mysql.jdbc.Driver");
```

```
Conn = DriverManager.getConnection(URL, USER, PASS);
```

<추가할 부분>

```
Context context = new InitialContext();
```

```
DataSource ds = (DataSource) context.lookup("java:/comp/env/jdbc/TestDB");
```

```
conn = ds.getConnection();
```

Lab 4-1 JDBC Connection Pool 사용하기3

Tomcat - MySQL 연동 구성

6. Tomcat 재기동 `cd ~/apache-tomcat-8.5.23/bin`

`./shutdown.sh` → `./startup.sh`

7. 브라우저에서 Test <http://localhost:8080/test/db2.jsp>

<정상적으로 확인이 안될 경우 : Tomcat 로그 확인>

`cd ~/apache-tomcat-8.5.23/logs`

`tail -f Catalina-오늘날짜.log`

Lab 4-1 JDBC Connection Pool 사용하기4

8. JMC(Java Mission Control)을 사용하여 JVM 상태 모니터링 하기

Oracle Java Mission Control

File Edit Window Help

JVM Browser Event Types

[1.8.0_144] The JVM Running Mission Control

[1.8.0_144] org.apache.catalina.startup.Bootstrap start (8364)

MBean Server

MBean Tree

Filter:

Catalina

Connector

DataSource

localhost

/

/docs

/examples

/host-manager

/manager

/test

DataSource

connections

1

jdbc/TestDB

MBean Features

| Name | Value | Update Interval |
|------------------------------|---------------------------|-----------------|
| logExpiredConnections | true | Default |
| loginTimeout | [N/A] | Default |
| maxConnLifetimeMillis | -1 | Default |
| maxIdle | 30 | Default |
| maxOpenPreparedStatements | -1 | Default |
| maxTotal | 100 | Default |
| maxWaitMillis | 10000 | Default |
| minEvictableIdleTimeMillis | 1800000 | Default |
| minIdle | 0 | Default |
| modelerType | org.apache.tomcat.dbcp... | Default |
| numActive | 0 | Default |
| numIdle | 1 | Default |
| numTestsPerEvictionRun | 3 | Default |
| password | welcome1 | Default |
| poolPreparedStatements | false | Default |
| removeAbandonedOnBorrow | false | Default |
| removeAbandonedOnMaintenance | false | Default |
| removeAbandonedTimeout | 300 | Default |
| rollbackOnReturn | true | Default |

Overview MBean Browser Triggers System Memory Threads Diagnostic Commands

1. 터미널에서 jmc 실행
2. org.apache.Catalina 선택
3. Mbean Broswer 선택
4. MBeanTree에서 Catalina/DataSource/localhost /test 선택하여 데이터 조회

numActive : 현재 사용중인 Connection수
numIdle : 현재 대기중인 Connection수 (사용가능)

Lab 4-2

Jmeter를 이용한 부하 테스트

소요시간: 30분

Lab 4-2 부하 테스트

Jmeter를 활용한 부하테스트

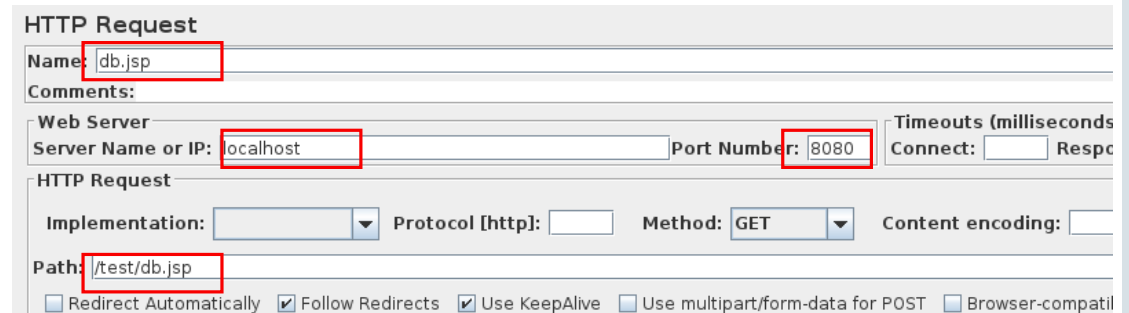
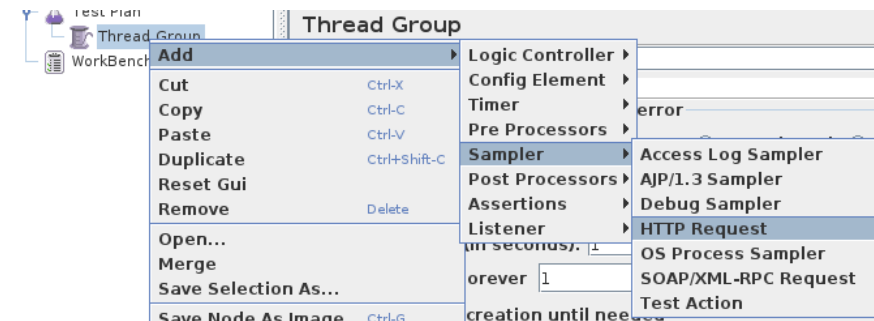
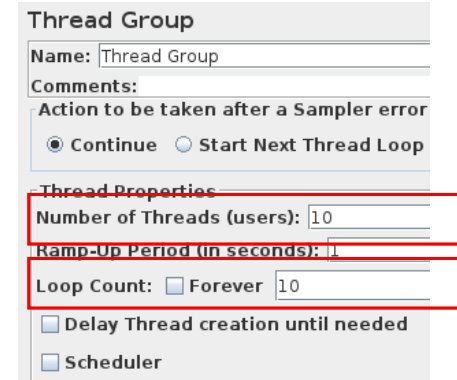
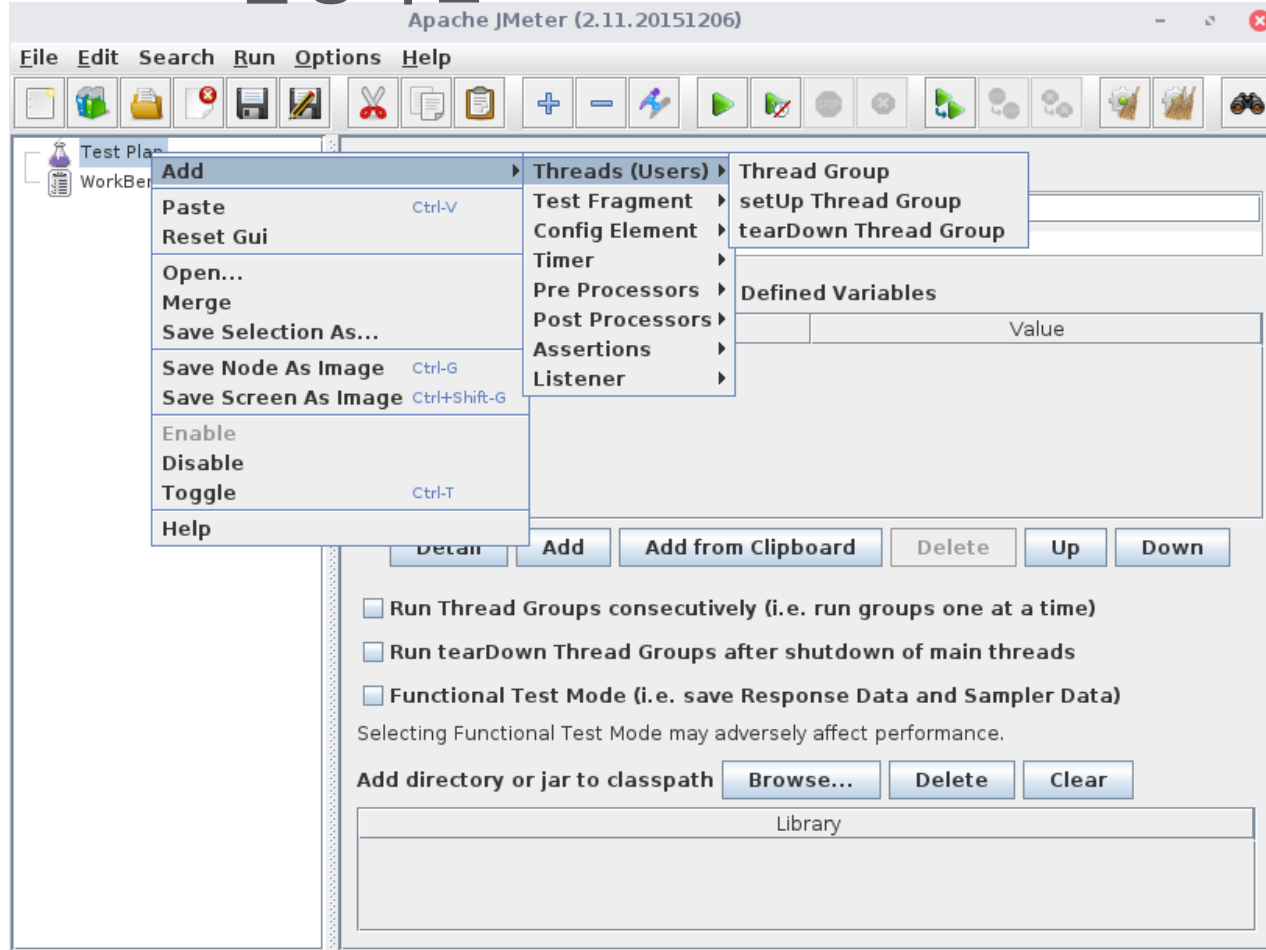
1. JMeter 설치 및 설정

- sudo apt-get install jmeter
- Jmeter 실행 (터미널에서 jmeter)
- Plan생성 : Test Plan 오른쪽 마우스 → Add → Threads → Thread Group
 - Number of Threads : 10, Loop Count : 10
- ThreadGroup 오른쪽 마우스 → Add → Sampler → Http Request
 - Name : db Server Name : localhost Path: /test/db.jsp
- ThreadGroup 오른쪽 마우스 → Add → Listener → Summary Report

2. 부하 실행 (Ctrl + R)

Lab 4-2 부하 테스트

Jmeter 설정화면



Lab 4-2 부하 테스트

부하 결과 확인

Summary Report

Name:

Comments:

Write results to file / Read from file

Filename **Log/Display Only:** ☐ Errors ☐ Successes

| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | KB/sec | Avg. Bytes |
|--------|-----------|---------|-----|-----|-----------|---------|------------|--------|------------|
| db.jsp | 100 | 391 | 55 | 553 | 103.57 | 0.00% | 21.1/sec | 5.21 | 253.0 |
| TOTAL | 100 | 161 | 55 | 553 | 103.57 | 0.00% | 21.1/sec | 5.21 | 253.0 |

Samples: 수행횟수
Average : 평균 응답시간(ms)
Min: 최소 응답시간(ms)
Max: 최대 응답시간(ms)
Throughput : 초당 수행 건수

Lab 4-2 부하 테스트

JDBC Connection pool 사용시 부하 결과 비교

1. JDBC Connection Pool을 사용하는 페이지 테스트

- 이전에 테스트 한 db.jsp 는 오른쪽 마우스 눌러서 Disable
- ThreadGroup 오른쪽 마우스 → Add → Sampler → Http Request
 - Name : db2 Server Name : localhost Path: /test/**db2.jsp**

2. 부하 실행후 결과 확인 (Ctrl + R)

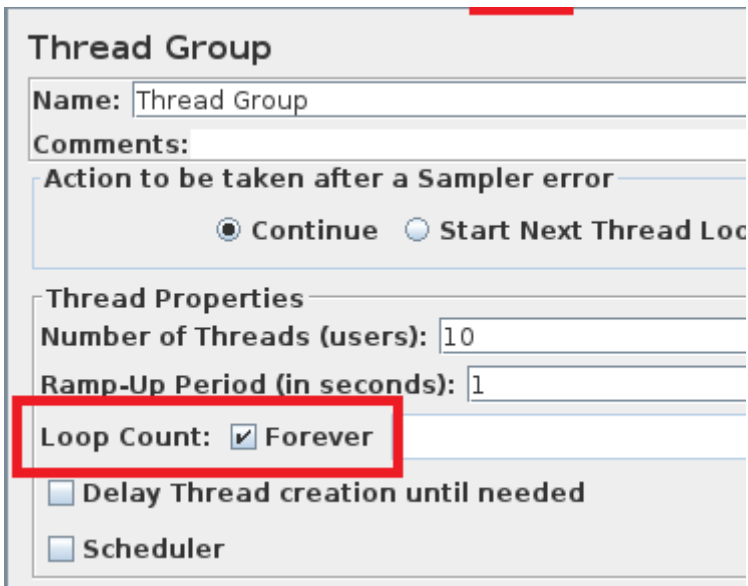
| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | KB/sec | Avg. Bytes |
|---------|-----------|---------|-----|-----|-----------|---------|------------|--------|------------|
| db2.jsp | 100 | 4 | 1 | 20 | 3.61 | 0.00% | 99.3/sec | 24.63 | 254.0 |
| TOTAL | 100 | 161 | 1 | 20 | 3.61 | 0.00% | 99.3/sec | 24.63 | 254.0 |

평균응답시간: 4ms, Throughput : 초당 처리율이 99개
Pool을 사용하지 않은 db.jsp 에 비해 성능이 개선된 것을 확인할 수 있음.

Lab 4-2 부하 테스트

부하시의 JDBC Connection Pool 확인

1. Thread Group에서 Loop Count를 Forever로 변경
2. 부하 실행후 JMC로 JDBC Connection Pool 모니터링



The screenshot shows the 'Thread Group' configuration dialog in Apache JMeter. The 'Name' field is 'Thread Group'. The 'Comments' field is empty. The 'Action to be taken after a Sampler error' section has 'Continue' selected. The 'Thread Properties' section shows 'Number of Threads (users)' as 10 and 'Ramp-Up Period (in seconds)' as 1. The 'Loop Count' is set to 'Forever' and is highlighted with a red box. The 'Delay Thread creation until needed' and 'Scheduler' checkboxes are unchecked.

| Thread Group | |
|--|---|
| Name: | Thread Group |
| Comments: | |
| Action to be taken after a Sampler error | |
| <input checked="" type="radio"/> | Continue |
| <input type="radio"/> | Start Next Thread Loc |
| Thread Properties | |
| Number of Threads (users): | 10 |
| Ramp-Up Period (in seconds): | 1 |
| Loop Count: | <input checked="" type="checkbox"/> Forever |
| <input type="checkbox"/> | Delay Thread creation until needed |
| <input type="checkbox"/> | Scheduler |

Lab 4-2 부하 테스트

JMC로 부하시의 JVM 상태 확인하기

MBean Tree

Filter:

+ /host-manager

+ /manager

- /test

- DataSource

- connections

+ 1

+ 10

+ 2

+ 3

+ 4

+ 5

+ 6

+ 7

+ 8

+ 9

"jdbc/TestDB"

"jdbc/TestDB"

MBean Features

Attributes

Operations

Notifications

Metadata

| Name | Value |
|--------------------------------|---------------|
| minEvictableIdleTimeMillis | 1800000 |
| minIdle | 0 |
| modelerType | org.apache... |
| numActive | 7 |
| numIdle | 3 |
| numTestsPerEvictionRun | 3 |
| password | welcome1 |
| poolPreparedStatements | false |
| removeAbandonedOnBorrow | false |
| removeAbandonedOnMaintenance | false |
| removeAbandonedTimeout | 300 |
| rollbackOnReturn | true |
| softMinEvictableIdleTimeMillis | -1 |
| testOnBorrow | true |
| testOnCreate | false |
| testOnReturn | false |
| testWhileIdle | false |

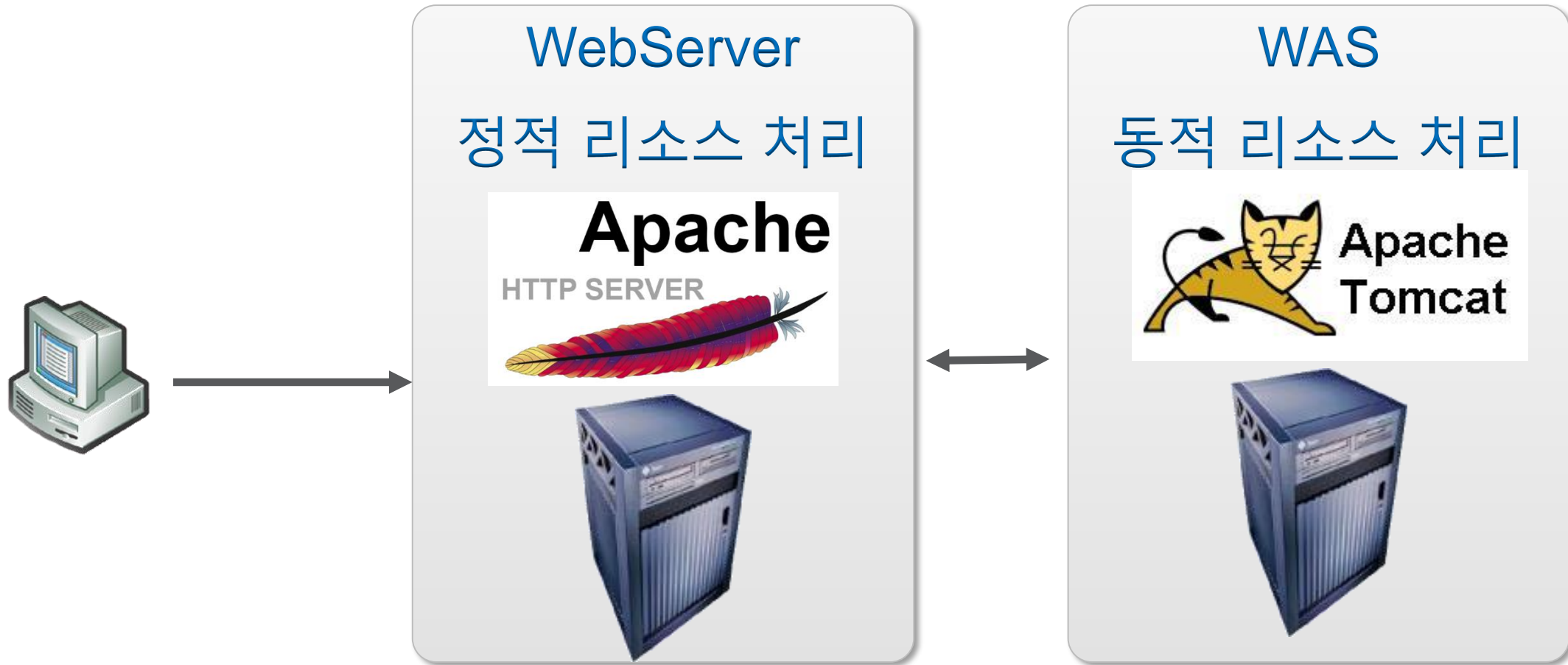
Lab 4-3

Web Server와 WAS(Web Application Server)의 구성

소요시간: 20분

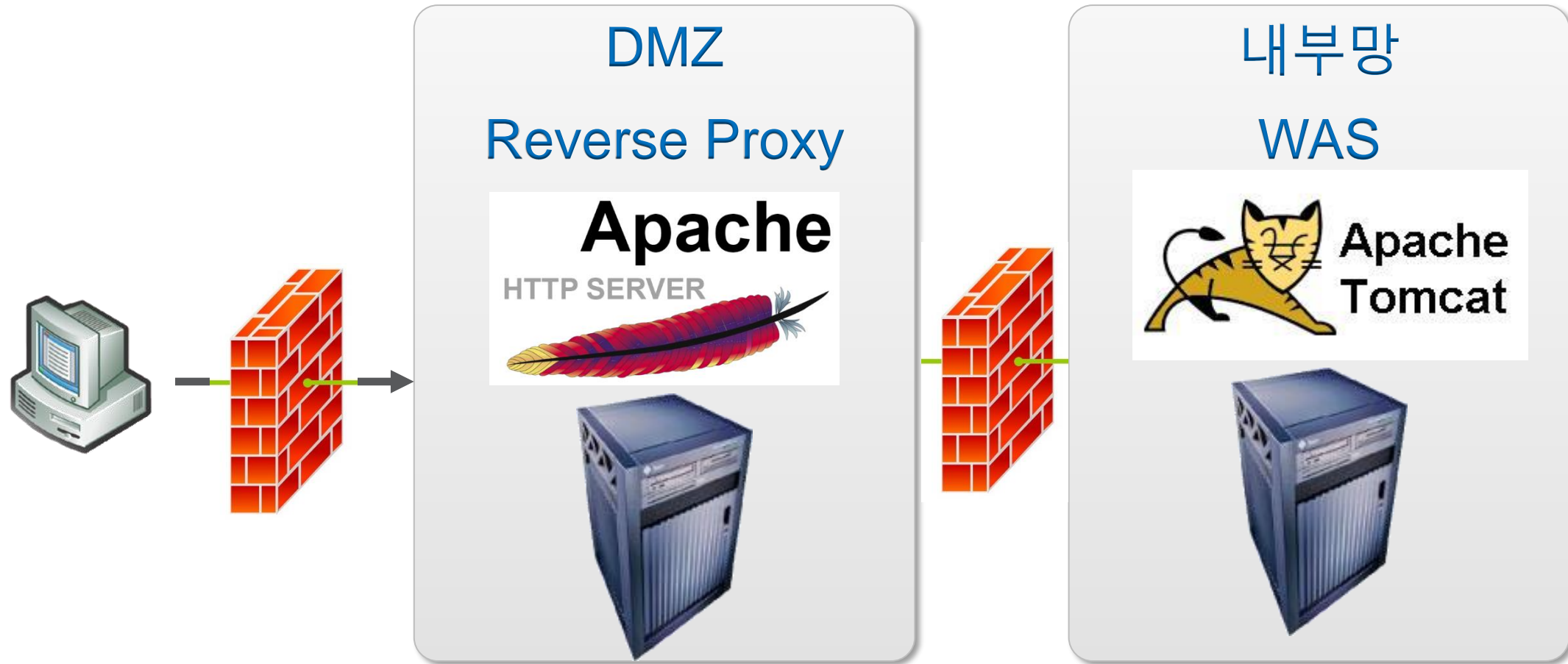
WebServer 와 WAS

업무 분담을 통한 성능 분산



WebServer 와 WAS

Security



Lab 4-3 웹서버 설치 (5분)

Apache 설치

- `sudo apt-get install apache2`
- 브라우저에서 확인 <http://localhost/>
- 설정 파일 위치 `/etc/apache2.conf`
 - `sudo service apache2 stop`
 - `sudo service apache2 start`
- Document Root : `/var/www/html`
- test.html 파일을 만들어서 브라우저로 확인해 보기

Lab 4-3 Apache + Tomcat 연동하기 (10분)

mod_proxy

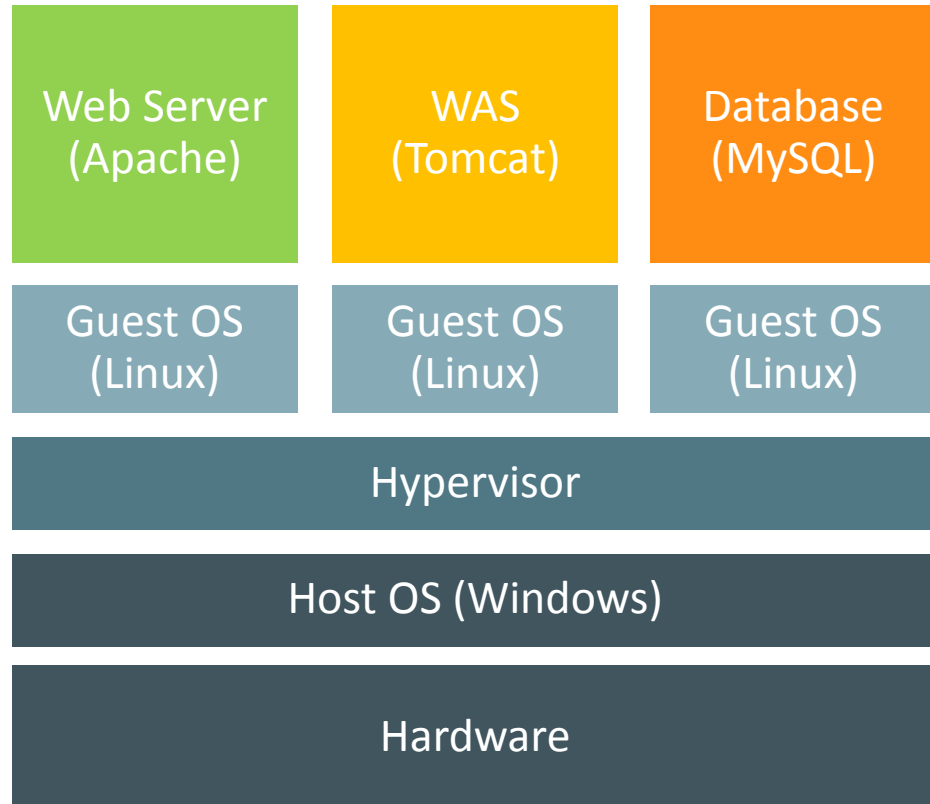
- 설정파일 복사 : `cd /etc/apache2/mods-available`
`sudo cp proxy.conf ../mods-enabled/`
`sudo cp proxy.load ../mods-enabled/`
`sudo cp proxy_http.load ../mods-enabled/`
- 연동설정 : `sudo vi /etc/apache2/mods-enabled/proxy.conf`
`ProxyPass / http://localhost:8080/`
`ProxyPassReverse / http://localhost:8080/`
- 재시작 `sudo service apache2 restart`
- 브라우저에서 확인 <http://localhost/test/>

Lab 4-4

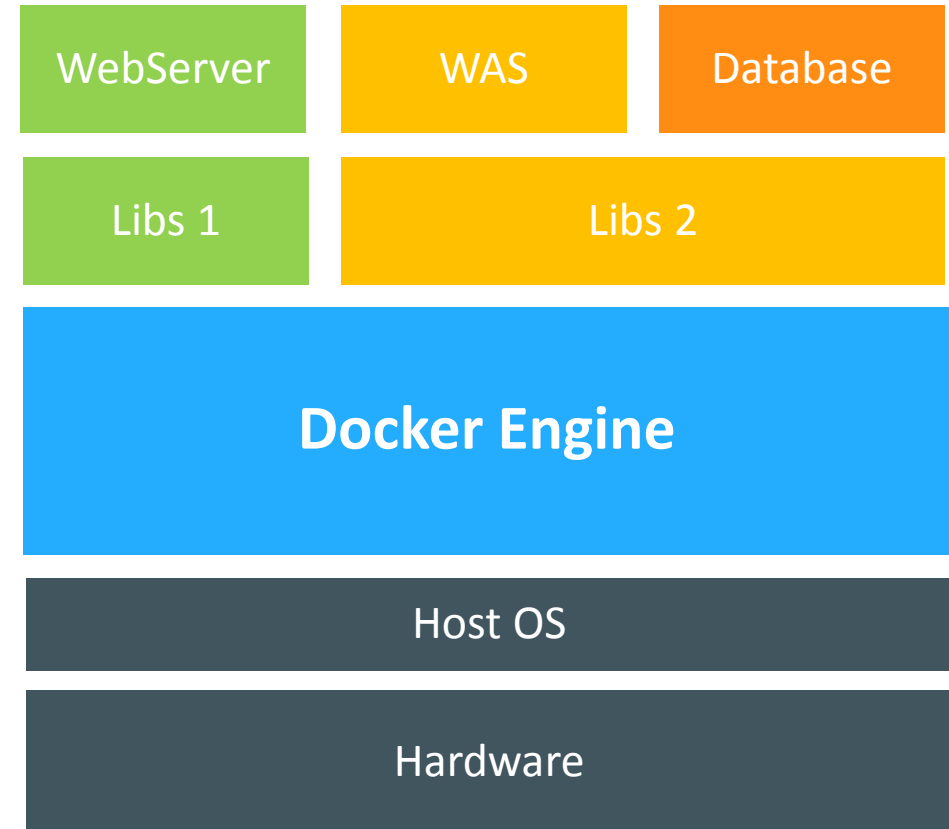
VM별로 분리구성하기

소요시간: 40분

Machine 별 Web/WAS/Database를 분리하여 구성하기



Virtual Machine



Container

Lab 4-4 VM별로 구동하기

VM별로 Apache, Tomcat, MySQL 별도로 구동

1. VM 3개로 복사
2. 방화벽 해제
3. MySQL VM 에서 모든 ip에 대해 접속 권한 변경

use mysql;

```
GRANT ALL PRIVILEGES ON *.* TO 'root' @ '%' IDENTIFIED BY 'welcome1' ;
```

4. MySQL Listen 주소 설정 변경

```
sudo vi /etc/mysql/mysql.conf.d/mysqld.conf
```

Tomcat쪽 app (db.jsp)에서 DB 접속 IP 변경

5. Apache Tomcat 연동 IP 변경

Lab 4-4 VM 별로 구동하기 (1)

VM 별로 Apache, Tomcat, MySQL 별도로 구동

1. VM 복사
2. 방화벽 해제 : 메뉴 → Settings → firewall configuration 선택 Status → off
3. MySQL VM 에서 접속 권한 변경

Use mysql;

```
GRANT ALL PRIVILEGES ON *.* TO 'root' @ '%' IDENTIFIED BY 'welcome1' ;
```

4. MySQL Listen 주소 설정 변경

```
sudo vi /etc/mysql/mysql.conf.d/mysqld.conf
```

#bind-address = 127.0.0.1 ← 왼쪽과 같이 주석처리하여 모든 주소에서 listen하도록 변경

Mysql 기동 : sudo service mysql start

아래 주소 정상 binding 여부 확인

```
root@diku-vm:/etc/mysql/mysql.conf.d# netstat -na | grep LISTEN | grep 3306
```

```
tcp6      0      0 :::3306          :::*              LISTEN
```

Lab 4-4 VM 별로 구동하기 (2)

VM 별로 Apache, Tomcat, MySQL 별도로 구동

1. Tomcat VM에서 mysql 접속 주소 변경

- Application 소스 db.jsp 파일에서 localhost → mysql VM의 IP 주소로 변경
- 접속 주소 확인 방법: mysql VM에서 ifconfig -a 로 확인

```
root@diku-vm:/etc/mysql/mysql.conf.d# ifconfig -a
```

```
enp0s3  Link encap:Ethernet  HWaddr 08:00:27:23:8e:a5
```

```
        inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
```

```
...
```

```
enp0s8  Link encap:Ethernet  HWaddr 08:00:27:d4:80:f6
```

```
        inet addr:192.168.56.103 Bcast:192.168.56.255  Mask:255.255.255.0
```

→ db.jsp 의 다음 변경 String URL = "jdbc:mysql://**192.168.56.103**:3306/diku";

2. 브라우저에서 확인

Tomcat VM의 주소를 확인 후에 VM외부의 Local PC에서 아래와 같이 브라우저로 확인

http://192.168.56.102:8080/test/db.jsp

Lab 4-4 VM별로 구동하기 (3)

VM별로 Apache, Tomcat, MySQL 별도로 구동

1. Apache 설정 변경

– 연동설정 수정 : `sudo vi /etc/apache2/mods-enabled/proxy.conf`

ProxyPass / <http://localhost:8080/> → tomcat VM IP로 변경

ProxyPassReverse / <http://localhost:8080/> → tomcat VM IP로 변경

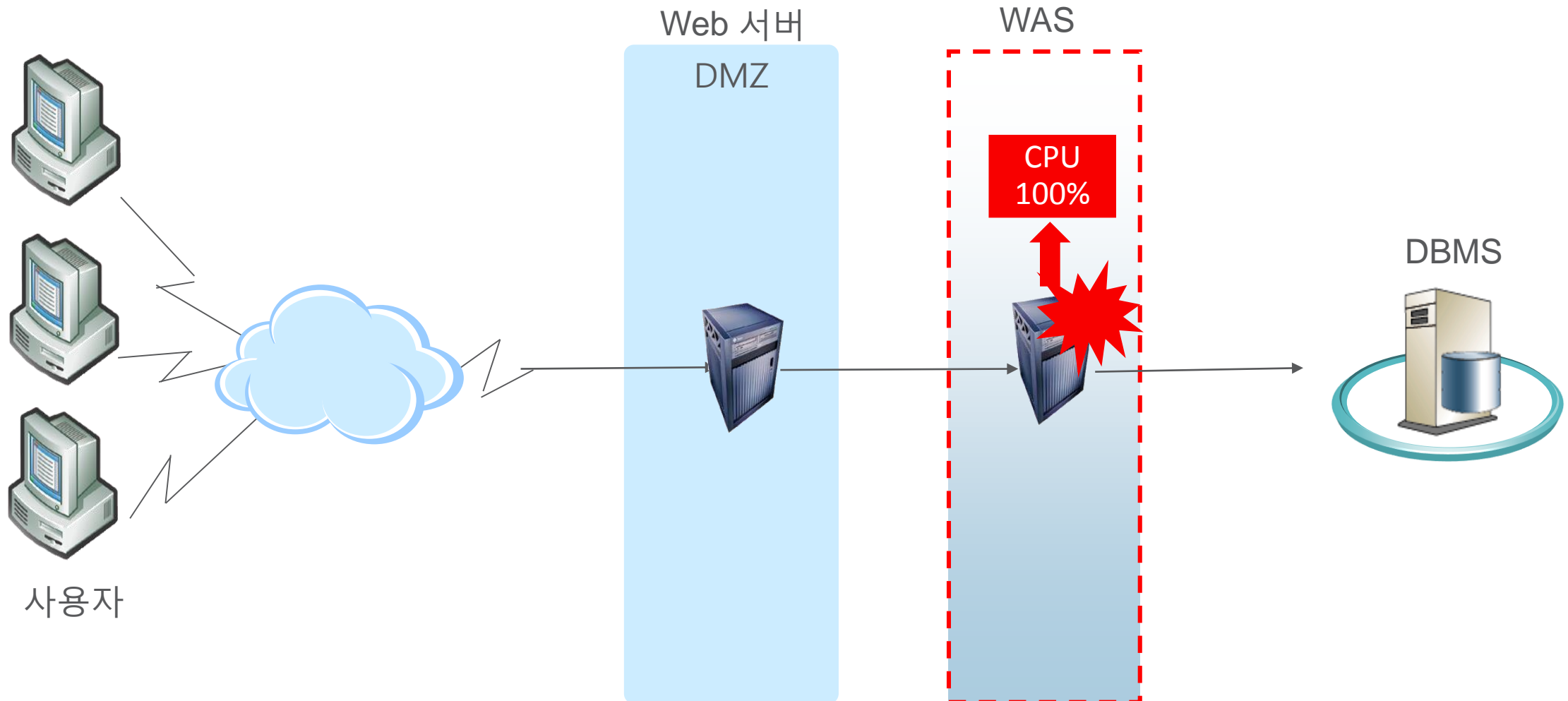
2. 재시작 `sudo service apache2 restart`

3. 브라우저에서 Apache VM IP로 접속하여 확인

<http://192.168.56.101/test/db.jsp>

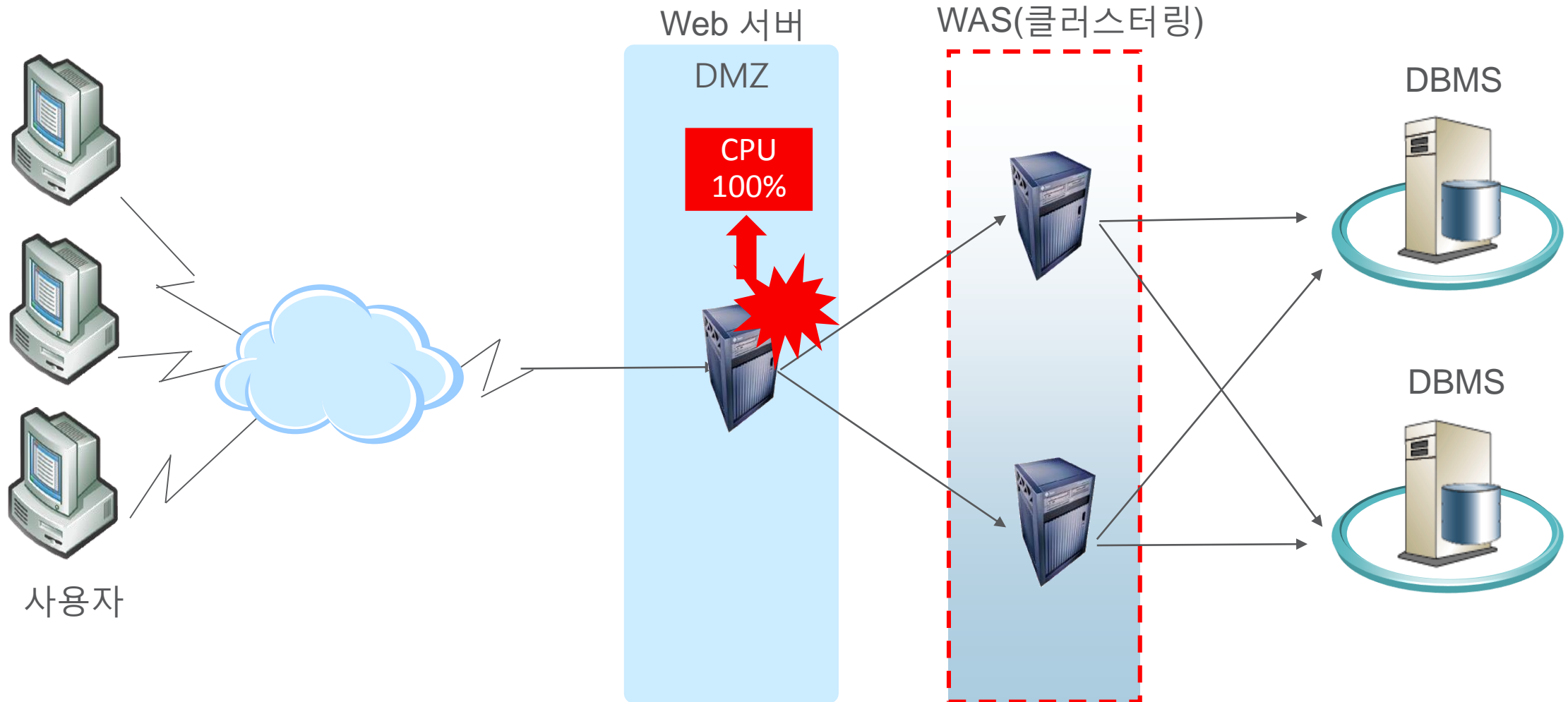
WAS (Web Application Server) 구성 Architecture

3 Tier Architecture



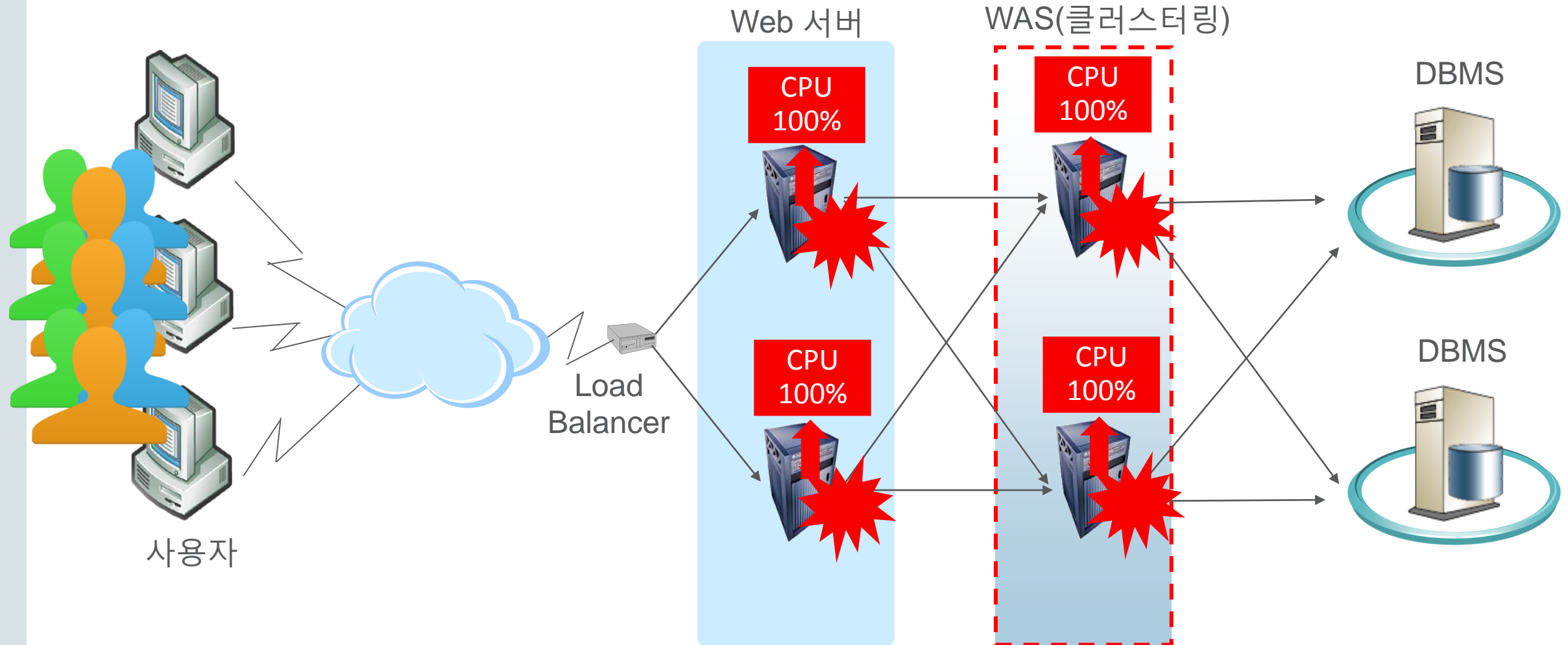
WAS (Web Application Server) 구성 Architecture

3 Tier Architecture



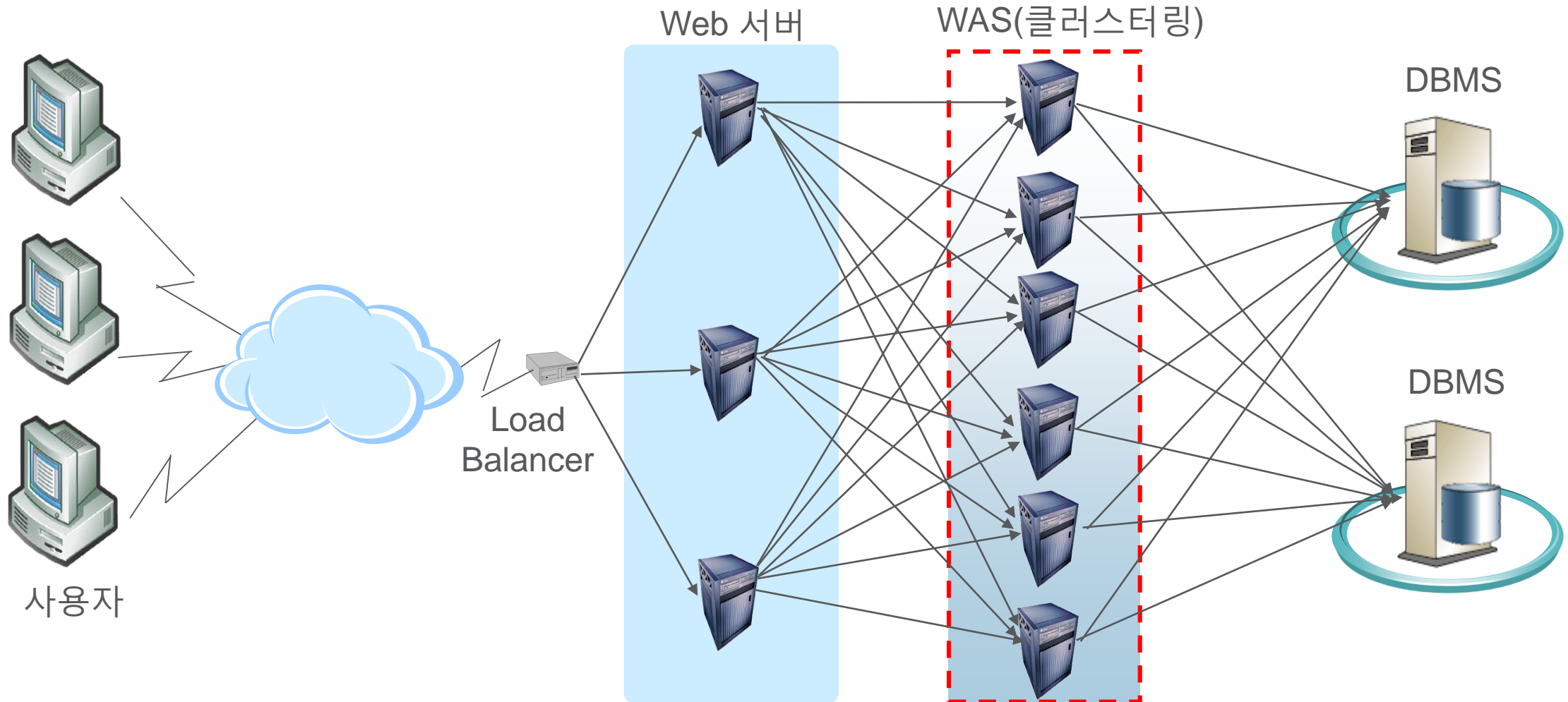
WAS (Web Application Server) 구성 Architecture

3 Tier Architecture



WAS (Web Application Server) 구성 Architecture

3 Tier Architecture



Lab 4-5

Docker 환경에서의 WEB/WAS구성

소요시간: 60분

Lab 4-5 Docker 환경에서의 WEB/WAS 구성

Docker 에서 Apache 기동하기

1. Docker이미지 안에 복사 할 내용 만들기

```
mkdir httpd
```

```
cd httpd
```

```
mkdir public-html
```

→ 정적인 처리를 하는 directory 생성

```
cd public-html
```

```
vi index.html 후 내용 입력 → index 파일 생성
```

```
cd ..
```

Lab 4-5 Docker 환경에서의 WEB/WAS 구성

Docker 에서 Apache 기동하기

2. vi Dockerfile 을 한 후에 아래 내용 입력

```
FROM httpd:2.4  
COPY ./public-html/ /usr/local/apache2/htdocs/
```

docker build -t my-apache . → 위 내용으로 이미지 생성
docker images 실행하여 my-apache 생성된 것 확인

3. 위에서 만든 이미지를 바탕으로 apache container 실행>
docker run -dit -p 8080:80 --name my-apache1 my-apache

4. 브라우저에서 확인 <http://localhost:8080/>

*윈도우의 경우는 localhost가 아니라 docker machine의 ip 입력(docker-machine ip)

Lab 4-5 Docker 환경에서의 WEB/WAS 구성

Docker 에서 Tomcat 기동하기

1. vi Dockerfile 을 한 후에 아래 내용 입력

```
FROM tomcat:latest
RUN mkdir -p /usr/local/tomcat/webapps/test/WEB-INF/lib
COPY ./mysql-connector-java-5.1.44-bin.jar /usr/local/tomcat/webapps/test/WEB-INF/lib
COPY ./web.xml /usr/local/tomcat/webapps/test/WEB-INF
COPY ./db.jsp /usr/local/tomcat/webapps/test
CMD ["catalina.sh", "run"]
```

2. 위 내용으로 이미지 만들기

```
docker build -t my-tomcat .
```

3. Docker 실행하기

```
docker run -dit --name myTomcat1 -p 8888:8080 my-tomcat
```

Lab 4-5 Docker 환경에서의 WEB/WAS 구성

Docker 에서 mysql 기동하기

1. vi Dockerfile 을 한 후에 아래 내용 입력

```
FROM mysql:5.7
ENV MYSQL_ROOT_PASSWORD="welcome1"
ENV MYSQL_DATABASE="dku"
ENV MYSQL_USER="dku"
ENV MYSQL_PASSWORD="welcome1"
EXPOSE 3306
```

2. 위 내용으로 이미지 만들기

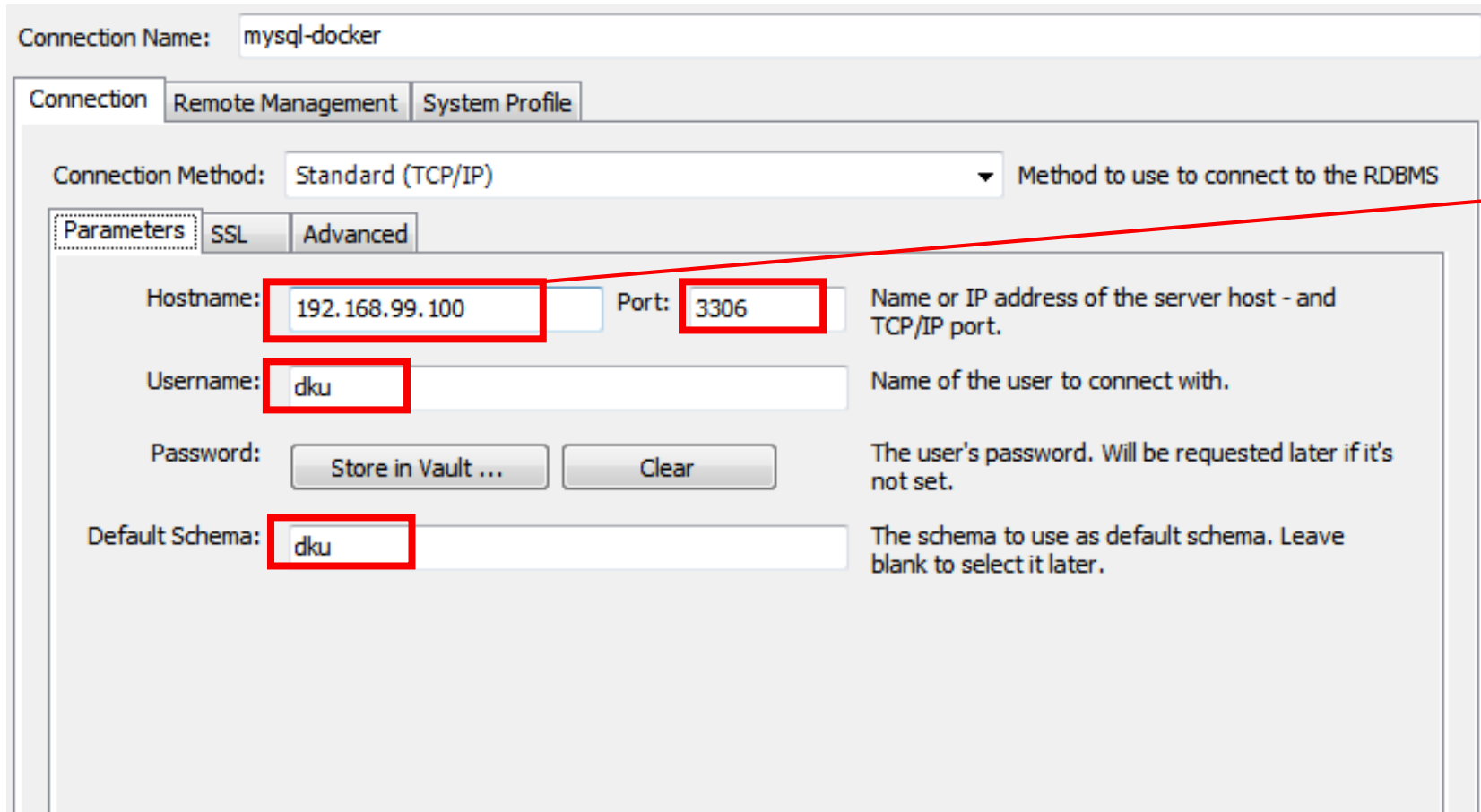
`docker build -t my-mysql .`

3. Docker 실행하기

`docker run -dit --name mysql1 -p 3306:3306 my-mysql`

Lab 4-5 Docker 환경에서의 WEB/WAS 구성

Mysql workbench 실행하여 docker안에 있는 mysql 에 접속하기



The screenshot shows the MySQL Workbench connection configuration window. The 'Connection Name' is 'mysql-docker'. The 'Connection Method' is 'Standard (TCP/IP)'. The 'Parameters' tab is selected. The 'Hostname' is '192.168.99.100' and the 'Port' is '3306'. The 'Username' is 'dku' and the 'Default Schema' is 'dku'. The 'Password' field is empty with 'Store in Vault ...' and 'Clear' buttons. A red arrow points from the 'Hostname' field to the text 'Linux : localhost' and '윈도우: docker의 machine-ip'.

Connection Name: mysql-docker

Connection Remote Management System Profile

Connection Method: Standard (TCP/IP) Method to use to connect to the RDBMS

Parameters SSL Advanced

Hostname: 192.168.99.100 Port: 3306 Name or IP address of the server host - and TCP/IP port.

Username: dku Name of the user to connect with.

Password: Store in Vault ... Clear The user's password. Will be requested later if it's not set.

Default Schema: dku The schema to use as default schema. Leave blank to select it later.

Linux : localhost
윈도우: docker의 machine-ip

Lab 4-5 Docker 환경에서의 WEB/WAS 구성

Table 생성 및 데이터 생성

- File->New Query Tab 선택하여 창을 연 후 아래 쿼리 실행(번개모양)
use dku;
create table test (id varchar(100), name varchar(100));
insert into test values('test1', 'this is test data');

Lab 4-5 Docker 환경에서의 WEB/WAS 구성

Docker Container 연결하기

- Docker 옵션으로 -link 이름:별칭
- 기존 container, 이미지 삭제하기
- `docker stop `docker ps -a -q``
- `docker rm `docker ps -a -q``
- `docker rmi my-apache`
- `docker run -dit --link mysql1:my-mysql --name myTomcat1 -p 8888:8080 my-tomcat`

Lab 4-5 Docker 환경에서의 WEB/WAS 구성

Docker 에서 Apache 내용 변경하기

1. httpd 폴더로 이동
2. vi Dockerfile 을 한 후에 아래 붉은 부분을 추가

```
FROM httpd:2.4
COPY ./public-html/ /usr/local/apache2/htdocs/
COPY ./add.txt /usr/local/apache2/
RUN cat /usr/local/apache2/add.txt >> /usr/local/apache2/httpd.conf
```

`docker build -t my-apache .` → 위 내용으로 이미지 생성
`docker images` 실행하여 my-apache 생성된 것 확인

3. 위에서 만든 이미지를 바탕으로 apache container 실행>

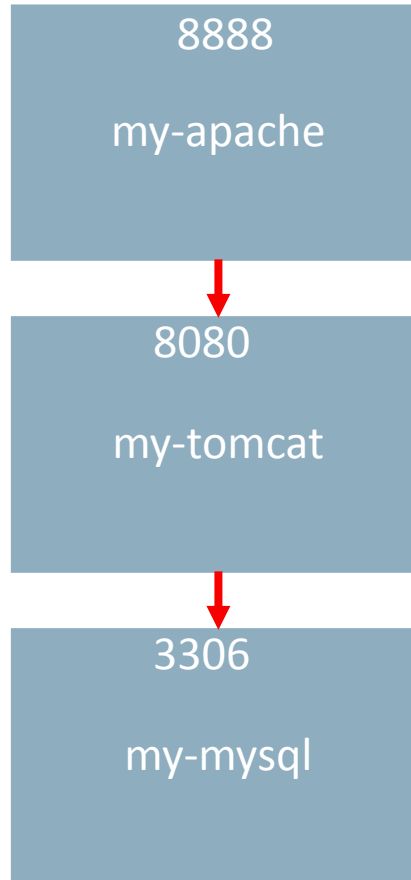
`docker run -dit -p 8080:80 --link myTomcat1:my-tomcat --name my-apache1 my-apache`

Lab 4-5 Docker 환경에서의 WEB/WAS 구성

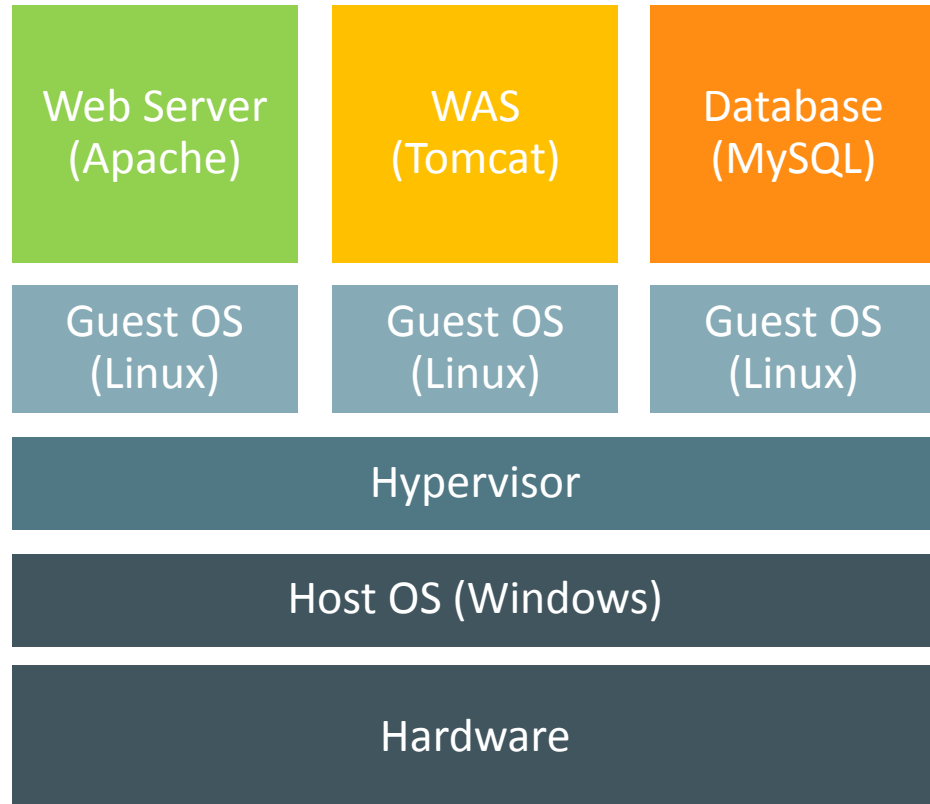
페이지 테스트하기

<http://localhost:8888/test/db.jsp>

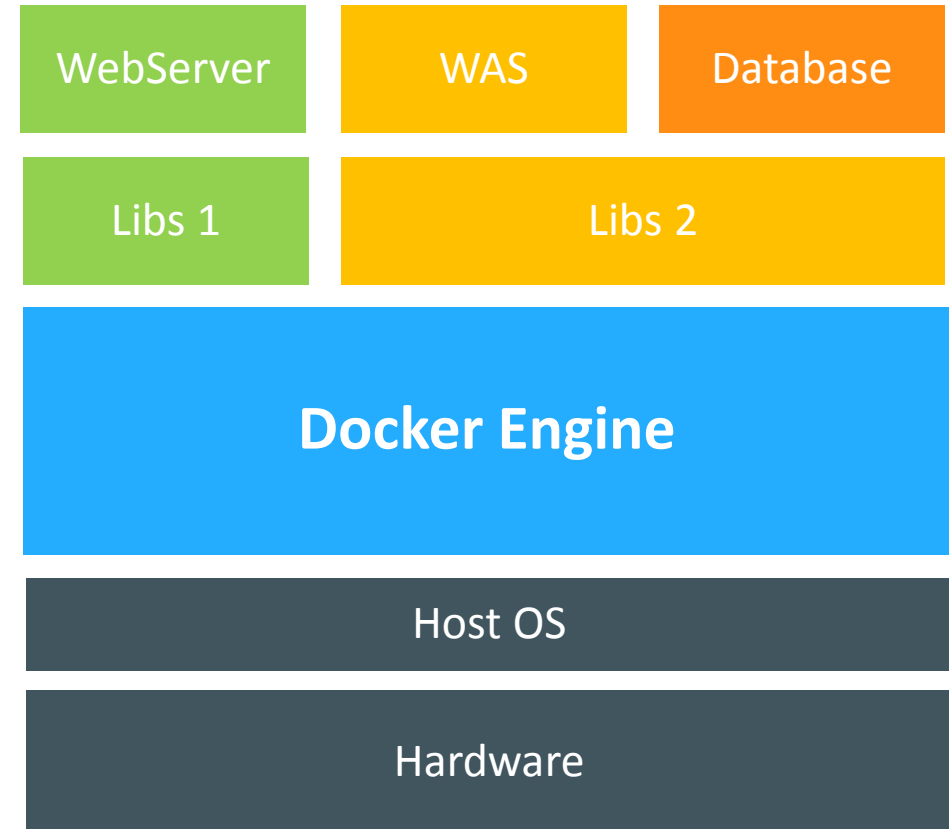
윈도우의 경우 docker-machine ip



Machine 별 Web/WAS/Database를 분리하여 구성하기



Virtual Machine



Container



ORACLE®