**VM 및 Docker 기반 웹시스템**

**아키텍처 구축실습**

**1 Tier 과제**

|  |  |
| --- | --- |
| **학번** | 20935217 |
| **이름** | 김동현 |

**1. 서론**

CLI 환경을 통해 소통하던 PC통신 시절을 지나 다양한 언어와 기술로 환경을 구축하는 현대에 이르기까지 다양한 웹들이 등장해왔다. 웹 환경은 빠르게 변화하는 비즈니스 환경에 대응하기 위해 유연하고 확장 가능한 아키텍처를 필요로 한다. 가상화 기술은 이러한 요구사항을 충족시키기 위한 효과적인 도구로서, 소프트웨어의 독립성을 보장하며 개발자와 시스템 관리자에게 유용한 환경을 제공한다. 최근의 웹 개발에서 가상화 기술은 핵심적인 역할을 수행하고 있으며, VM(Virtual Machine) 및 Docker와 같은 가상화 기술은 웹 시스템의 효율적인 운영, 확장성, 그리고 유지보수를 위한 필수 도구로 자리잡았습니다. 이를 통해 시스템의 안정성과 확장성을 높이고, 자원의 효율적인 사용을 통해 비용을 절감할 수 있다. 이 과제를 통해 가상화 기술을 활용하여 웹 시스템의 효율적인 관리와 배포를 가능하게 하는 것이 이번의 목표이다. VM 및 Docker를 활용하여 웹 시스템의 효율적인 아키텍처를 구축하고, 이를 통해 웹 시스템의 관리 및 운영을 효율적으로 수행하는 방법을 탐구할 것이다.

**2. 본론**

(1) 아키텍처 구축 상황

이 과제에서 사용한 아키텍처는 3 Tier로 구축하였다. 일반 사용자가 직접 엑세스할 수 있는 프레젠테이션 계층은 Docker의 image로 만들어진 Nginx로 구축하였다. Nginx는 웹 서버로써 클라이언트의 요청을 받아 비즈니스 로직 계층으로 전달한다. 그 다음으로 WAS로 대표되는 어플리케이션 서버를 사용하고 어플리케이션에 대한 비즈니스 논리를 처리하는 비즈니스 로직 계층은 Docker의 image로 만들어진 tomcat으로 구축하였다. Tomcat은 자바 기반의 웹 어플리케이션 서버로써, 클라이언트의 요청을 받아 데이터베이스 계층으로부터 데이터를 가져와 응답을 생성하여 클라이언트에게 전달한다. 마지막으로 응용 프로그램 데이터에 대한 엑세스를 제공하는 DBMS에 해당하는 데이터베이스 계층은 Docker의 image로 만들어진 MariaDB를 이용해서 구축하였다. MariaDB는 RDBMS(관계형 데이터베이스 관리 시스템)로써 응용프로그램에서 필요한 데이터를 저장하고 관리한다. 요약하면, 이 아키텍처에서는 클라이언트의 요청을 Nginx에서 받아 Tomcat으로 전달하고, Tomcat은 필요한 데이터를 MariaDB에서 가져와 응답을 생성하여 클라이언트에게 제공한다. 추가적으로, 이 과정에서 환경에 따른 문제를 최소화하기 위해 CentOS7이 설치된 VM 환경 내에서 진행하였다.

(2) 아키텍처 구축 절차

A. Web Server

이 아키텍처에서 웹 서버로 사용하는 프로그램은 Nginx이고, Docker를 통해 구축하기로 하였으므로 먼저 도커로 nginx 컨테이너를 진행하는 작업부터 진행한다.

|  |
| --- |
| docker pull nginx |

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

이후 이미지를 컨테이너로 실행한다.

|  |
| --- |
| docker run --name ws\_nginx -d -p 80:80 nginx |

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

그 다음으로 실제로 잘 작동하는지 확인한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

B. Web App Server

이 아키텍처에서 웹 앱 서버로 사용하는 프로그램은 Tomcat이고, Docker를 통해 구축하기로 하였으므로 먼저 도커로 Tomcat 컨테이너를 진행하는 작업부터 진행한다.

|  |
| --- |
| docker pull tomcat |

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

이후 이미지를 컨테이너로 실행한다.

|  |
| --- |
| docker run -d -i -t --name was\_tomcat -p 8080:8080 tomcat |

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

그 다음으로 실제로 잘 작동하는지 확인한다.

텍스트, 폰트, 라인, 번호이(가) 표시된 사진

자동 생성된 설명

Tomcat에는 연결이 된 것을 확인했다. 이후, 실제로 웹페이지가 잘 뜨는지 확인하기 위해 예시 페이지를 만든다.

|  |
| --- |
| docker exec -it was\_tomcat /bin/bash  cd ./webapps  mkdir ROOT  cd ./ROOT  vi index.jsp |

텍스트, 폰트, 화이트, 디자인이(가) 표시된 사진

자동 생성된 설명

잘 되는 것을 확인했다.

C. Database

이 아키텍처에서 데이터베이스로 사용하는 프로그램은 MariaDB이고, Docker를 통해 구축하기로 하였으므로 먼저 도커로 MariaDB 컨테이너를 진행하는 작업부터 진행한다.

|  |
| --- |
| docker pull mariadb |

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

이후 이미지를 컨테이너로 실행한다.

|  |
| --- |
| docker run --name mariadb -d -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=qwer1234! mariadb |

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

그 다음으로 실제로 잘 작동하는지 확인한다.

|  |
| --- |
| docker exec -it mariadb /bin/bash  mariadb -u root -p  show databases; |

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

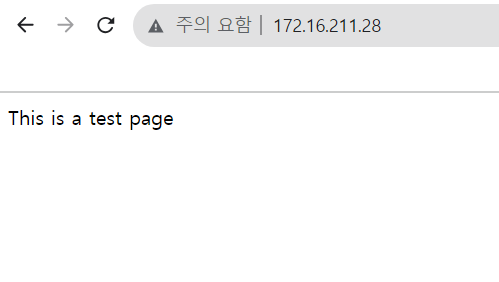
D. WS-WAS 연동

연동을 하기 위해 설정 파일인 nginx.conf을 수정한다.

|  |
| --- |
| docker exec -it ws\_nginx /bin/bash  vi /etc/nginx/nginx.conf |

이후 재시작을 하고, 잘 연동이 되었는지 확인한다.

|  |
| --- |
| docker restart ws\_nginx |



E. WAS-DB 연동

연동을 하기 위해 먼저 tomcat 컨테이너에 들어가서 파일을 생성한다.

|  |
| --- |
| docker exec -it was\_tomcat /bin/bash  cd webapps/ROOT  vi dbconnection.jsp |

이후 java를 통해 mariadb와 연동해주는 작업을 거친다.

|  |
| --- |
| cd /usr/local/tomcat/lib  wget https://downloads.mariadb.com/Connectors/java/connector-java-3.2.0/mariadb-java-client-3.2.0.jar |

이후 재시작을 하고, 잘 연동이 되었는지 확인한다.

|  |
| --- |
| docker restart was\_tomcat |

텍스트, 스크린샷, 폰트, 화이트이(가) 표시된 사진

자동 생성된 설명

**3. 결론**

(1) 아키텍처 구축 결과

Nginx – tomcat -MariaDB 간 연동이 완료되었다. 각각의 페이지는 정상적으로 작동한다.

(2) 느낀 점

생각보다 웹 개발을 하면서 챙겨야 할 점이 많다는 생각이 들었다. 이번 프로젝트에서도 여러가지를 고민하면서 진행을 해야겠다는 생각이 들었다.