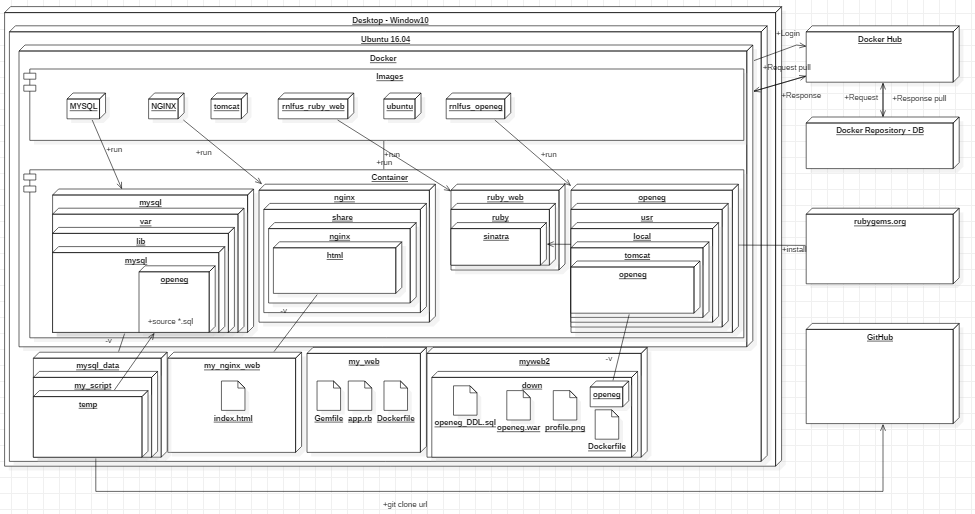
======================2019.5.27==========================

1. openeg 웹과 MySQL 연동해보기

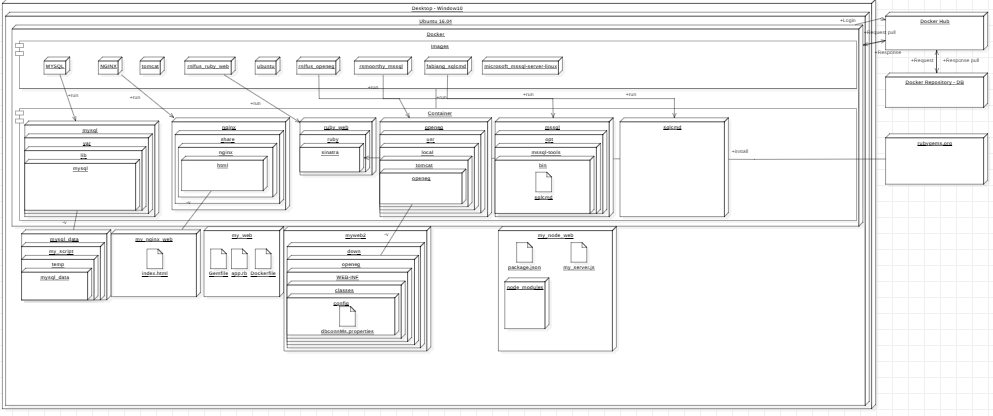
|  |
| --- |
| 내컴>vmware>ubuntu18> docker stop mysql  내컴>vmware>ubuntu18> docker rm –f mysql  내컴>vmware>ubuntu18> mkdir mysql\_data  내컴>vmware>ubuntu18> docker run –d –p 3306:3306 –e MYSQL\_ALLOW\_EMPTY\_PASSWORD=true --name mysql –v $(pwd)/mysql\_data:/var/lib/mysql mysql:5.7  내컴>vmware>ubuntu18> cd mysql\_data  내컴>vmware>ubuntu18> ls (mysql 데이터 관련 폴더와 파일이 여러 개 보임)  내컴>vmware>ubuntu18> mkdir my\_script  내컴>vmware>ubuntu18> cd my\_script  내컴>vmware>ubuntu18> git clone <https://github.com/javanism/temp>  내컴>vmware>ubuntu18> cd temp  내컴>vmware>ubuntu18> ls (openeg\_DDL.sql이 보인다)  내컴>vmware>ubuntu18> docker exec –ti mysql bash;  내컴>vmware>ubuntu18>docker>Debian9> mysql –uroot  내컴>vmware>ubuntu18>docker>Debian9>mysql> show databases;  내컴>vmware>ubuntu18>docker>Debian9>mysql> create database openeg;  내컴>vmware>ubuntu18>docker>Debian9>mysql> use openeg;  내컴>vmware>ubuntu18>docker>Debian9>mysql> source openeg\_DDL.sql;  내컴>vmware>ubuntu18>docker>Debian9>mysql> show tables; (openeg 웹 관련 테이블 5개가 보여야 한다)  내컴>vmware>ubuntu18>docker>Debian9>mysql> select \* from board\_member;  내컴>vmware>ubuntu18>docker>Debian9>mysql> use mysql;  내컴>vmware>ubuntu18>docker>Debian9>mysql> update user set authentication\_string=password('apmsetup') where user='root'; (루트 패스워드 바꾸기)  내컴>vmware>ubuntu18>docker>Debian9>mysql> flush privileges;  내컴>vmware>ubuntu18>docker>Debian9>mysql> quit;  내컴>vmware>ubuntu18>docker>Debian9> mysql –u root -p  Enter password: apmsetup  내컴>vmware>ubuntu18>docker>Debian9>mysql> use openeg;  내컴>vmware>ubuntu18>docker>Debian9>mysql> show tables;  내컴>vmware>ubuntu18>docker>Debian9>mysql> quit;  내컴>vmware>ubuntu18>docker>Debian9> exit;  내컴>vmware>ubuntu18> docker stop mysql  내컴>vmware>ubuntu18> docker rm –f mysql  내컴>vmware>ubuntu18> docker run –d –p 3306:3306 --name mysql –v $(pwd)/mysql\_data:/var/lib/mysql mysql:5.7  내컴>vmware>ubuntu18> docker inspect mysql (mysql 컨테이너 ip 알아내기 ex. 172.17.0.2)  내컴>vmware>ubuntu18> docker stop openeg  내컴>vmware>ubuntu18> cd (openeg mount directory/WEB-INF/classes/config/)  내컴>vmware>ubuntu18> vi dbconn.properties  (mysql url 중 ip 주소만 172.17.0.2로 바꾼다)  내컴>vmware>ubuntu18> docker start openeg  내컴>vmware>ubuntu18> 웹브라우저에 admin/openeg로 로그인 해본다 |

(Architecture diagram)

docker stop mysql 이전의 UML



docker stop mysql 부터의 UML

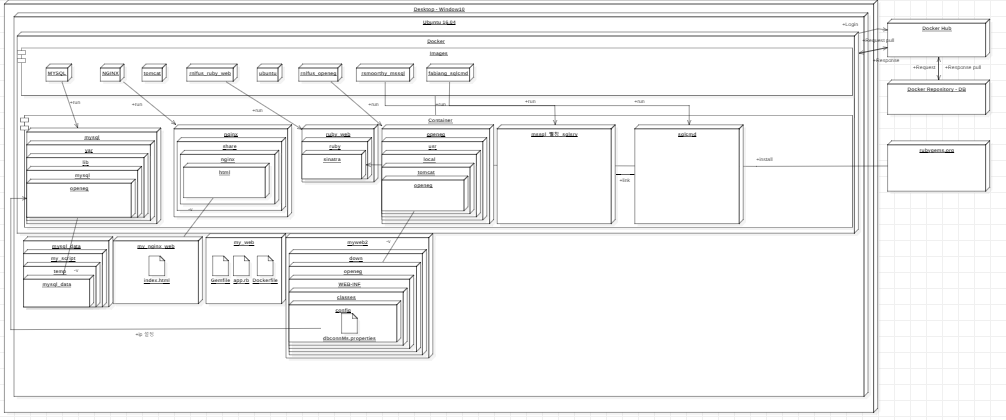


1. openeg 웹과 MS-SQL 연동해보기

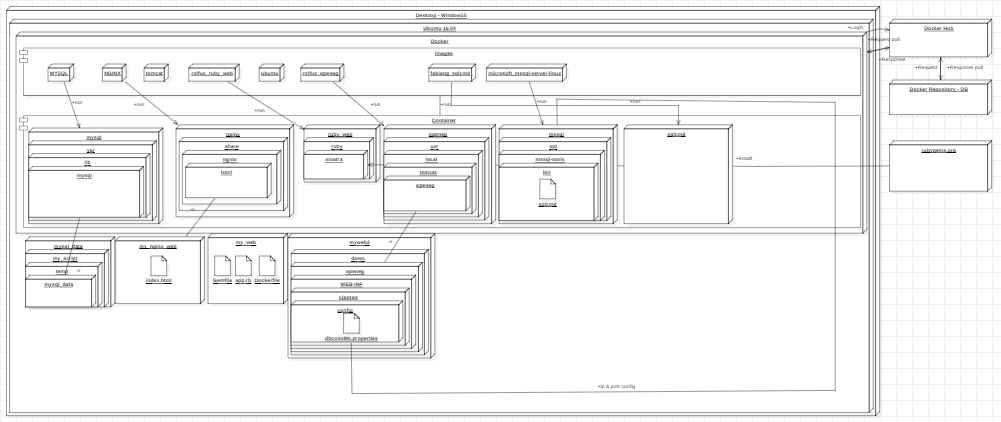
|  |
| --- |
| 내컴>vmware>ubuntu18> cat my\_tomcat\_web/down/openeg/WEB-INF/classes/config/dbconnMs.properties (MS SQL을 사용한다는 것을 본다)  내컴>vmware>ubuntu18> docker search mssql  내컴>vmware>ubuntu18> docker pull rsmoorthy/mssql  내컴>vmware>ubuntu18> docker create --name mssql -p 1433:1433 rsmoorthy/mssql (도커 허브에서 정보를 얻어 컨테이너를 생성한다)  내컴>vmware>ubuntu18> docker start mssql  내컴>vmware>ubuntu18> docker ps  내컴>vmware>ubuntu18> docker exec –it mssql /bin/bash (bash가 설치되어 있지 않은 것을 확인한다)  내컴>vmware>ubuntu18> docker exec –it mssql sqlcmd (sqlcmd도 설치되어 있지 않은 것을 확인한다)  내컴>vmware>ubuntu18> docker search sqlcmd  내컴>vmware>ubuntu18> docker pull fabiang/sqlcmd  내컴>vmware>ubuntu18> docker create –name sqlcmd –link mssql:sqlsrv fabiang/sqlcmd –S sqlsrv –U sa  내컴>vmware>ubuntu18> docker start sqlcmd  (여기 뭐가 안됐더라,,,?)  엄청난 search 삽질을 통해…  내컴>vmware>ubuntu18> docker stop mssql  내컴>vmware>ubuntu18> docker rm –f mssql  내컴>vmware>ubuntu18> docker rmi –f (mssql 관련 이미지)  내컴>vmware>ubuntu18> lsof –i:1433 (1433 포트를 쓰고 있는 PID가 보일 수 있다)  내컴>vmware>ubuntu18> kill (PID) (여러 개 있으면 모두 죽여야 한다)  내컴>vmware>ubuntu18> docker run --name mssql -e 'ACCEPT\_EULA=Y' -e 'SA\_PASSWORD=Qwer@1234’ -e 'MSSQL\_PID=Express' -p 1433:1433 -d mcr.microsoft.com/mssql/server:2017-latest-ubuntu  내컴>vmware>ubuntu18> docker exec –it mssql bash  내컴>vmware>ubuntu18>docker>ubuntu16> /opt/mssql-tools/bin./sqlcmd –S localhost –U sa –P Qwer@1234  내컴>vmware>ubuntu18>docker>ubuntu16>sqlcmd>1> create database board  내컴>vmware>ubuntu18>docker>ubuntu16>sqlcmd>1> select name from sys.databases  내컴>vmware>ubuntu18>docker>ubuntu16>sqlcmd>1> go  내컴>vmware>ubuntu18>docker>ubuntu16>sqlcmd>1> use board  내컴>vmware>ubuntu18>docker>ubuntu16>sqlcmd>1> CREATE TABLE board\_member (IDX int IDENTITY(1,1) PRIMARY KEY,USERID VARCHAR(70) NOT NULL,USERPW VARCHAR(50) NOT NULL,USERNAME VARCHAR(50) NOT NULL,JOINDATE DATETIME );  내컴>vmware>ubuntu18>docker>ubuntu16>sqlcmd>1> insert into board\_member(userid,userpw,username,joindate) values('admin','1234','관리자',getdate());  내컴>vmware>ubuntu18>docker>ubuntu16>sqlcmd>1> insert into board\_member(userid,userpw,username,joindate) values('test','test','테스터',getdate());  내컴>vmware>ubuntu18>docker>ubuntu16>sqlcmd>1> go  내컴>vmware>ubuntu18>docker>ubuntu16>sqlcmd>1> select \* from board\_member  내컴>vmware>ubuntu18>docker>ubuntu16>sqlcmd>1> go  내컴>vmware>ubuntu18>docker>ubuntu16>sqlcmd>1> quit  내컴>vmware>ubuntu18>docker>ubuntu16> exit  내컴>vmware>ubuntu18> docker ps (mssql이 여전히 1433포트에서 떠있는 것을 확인한다)  내컴>vmware>ubuntu18> docker inspect mssql ( mssql 이 돌고 있는 컨테이너의 IP를 확인한다)  내컴>vmware>ubuntu18> vi my\_tomcat\_web/down/openeg/WEB-INF/classes/config/dbconnMs.properties (IP와 포트, 패스워드를 고친다)  내컴>vmware>ubuntu18> 웹브라우저에 test을 입력하고 실행해본다 |

(Architecture diagram)

삽질 이전까지



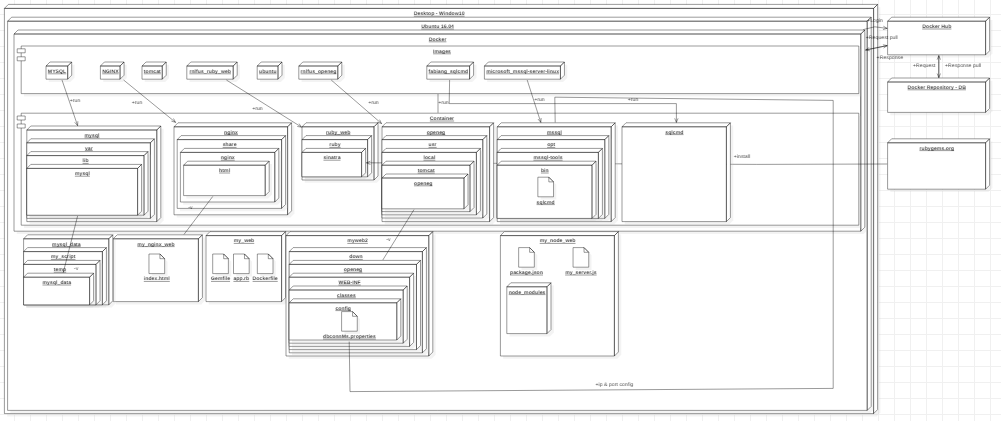
삽질 이후부터



1. 호스트 머쉰에 node 웹 서버 구성하기

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 내컴>vmware>ubuntu18> mkdir my\_node\_web  내컴>vmware>ubuntu18> vi package.json   |  | | --- | | {  "name": "jes\_node\_server",  "version": "0.0.0",  "private": true,  "scripts": {  "start": "node my\_server.js"  },  "description": "NodeJS Test Server",  "author": {  "name": "JES"  },  "dependencies": {  "express": "^4.16.3",  "uuid": "^3.2.1"  }  } |   내컴>vmware>ubuntu18> npm install  내컴>vmware>ubuntu18> ls (node\_modules라는 디렉토리가 보인다)  내컴>vmware>ubuntu18> vi my\_server.js   |  | | --- | | var express = require('express');  var app = express();  const port = 3000;    var server = app.listen(port, function () {  console.log("Express server has started on port : "+port);  });    app.get('/', function (req, res) {  res.send('Hello jes');  }); |   내컴>vmware>ubuntu18> npm start  내컴>vmware>ubuntu18> 웹브라우저로 localhost:3000 확인한다  내컴>vmware>ubuntu18> vi my\_server.js (내용을 Hello Node~! 가 출력되도록 고쳐본다)  내컴>vmware>ubuntu18> 웹브라우저를 다시 고침해도 반영되지 않는다.  내컴>vmware>ubuntu18> Ctrl+C (node 서버를 종료한다)  내컴>vmware>ubuntu18> npm start (node 서버 재가동)  내컴>vmware>ubuntu18> 웹브라우저에 반영된다  내컴>vmware>ubuntu18> Ctrl+C (node 서버를 종료한다)  내컴>vmware>ubuntu18> npm install nodemon –g  내컴>vmware>ubuntu18> vi package.json   |  | | --- | | "scripts": {  "start": "nodemon my\_server.js"  }, |   내컴>vmware>ubuntu18> vi my\_server.js   |  | | --- | | app.get('/', function(req, res) {  res.send('Hello Nodemon ');  }); |   내컴>vmware>ubuntu18> npm start  내컴>vmware>ubuntu18> 웹브라우저로 확인한다  내컴>vmware>ubuntu18> vi my\_server.js (내용을 Hello Nodemon ^^~! 가 출력되도록 고쳐본다)  내컴>vmware>ubuntu18> 웹브라우저를 새로 고침하면 반영되어 있다 |

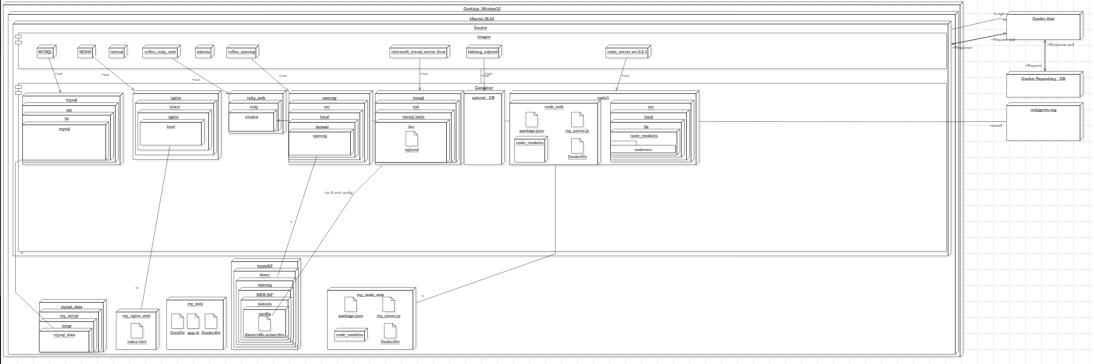
(Architecture diagram)



1. 도커 컨테이너로 node 웹 서버 구성하기

|  |  |
| --- | --- |
| 내컴>vmware>ubuntu18> Ctrl+C (node 서버를 종료한다)  내컴>vmware>ubuntu18> vi Dockerfile   |  | | --- | | FROM node:carbon  MAINTAINER jes@jes.com    #폴더 만들기 - NodeJS 어플리케이션 폴더  RUN mkdir -p /node\_web    #어플리케이션 폴더를 Workdir로 지정 - 서버가동용  WORKDIR /node\_web    #서버 파일 복사 COPY [어플리케이션파일 위치] [컨테이너내부의 어플리케이션 파일위치]  COPY ./ /node\_web    #패키지파일들 받기  RUN npm install  #nodemon패키지 받기 –g는 global /usr/local/lib/node\_modules  RUN npm install –g nodemon    #배포버젼으로 설정  ENV NODE\_ENV=production    #서버실행  CMD [“npm”,”start”] |   내컴>vmware>ubuntu18> docker build –tag node\_server:0.0.1 .  내컴>vmware>ubuntu18> docker images  내컴>vmware>ubuntu18> docker create –name node1 –p 3001:3000 –v $(pwd):/node\_web node\_server:0.0.1  내컴>vmware>ubuntu18> docker images  내컴>vmware>ubuntu18> docker start node  내컴>vmware>ubuntu18> docker ps  내컴>vmware>ubuntu18> 웹브라우저로 확인한다  내컴>vmware>ubuntu18> vi my\_server.js (내용을 Hello Docker Node Container ^^~! 가 출력되도록 고쳐본다)  내컴>vmware>ubuntu18> 웹브라우저를 새로 고침하면 반영되어 있다 |

(Architecture diagram)



1. 도커 컨테이너로 node 웹 서버 여러 개 구성하기

|  |
| --- |
| 내컴>vmware>ubuntu18> docker create –name node2 –p 3002:3000 –v $(pwd):/node\_web node\_server:0.0.1  내컴>vmware>ubuntu18> docker create –name node3 –p 3003:3000 –v $(pwd):/node\_web node\_server:0.0.1  내컴>vmware>ubuntu18> docker create –name node4 –p 3004:3000 –v $(pwd):/node\_web node\_server:0.0.1  내컴>vmware>ubuntu18> docker images  내컴>vmware>ubuntu18> docker start node2  내컴>vmware>ubuntu18> docker start node3  내컴>vmware>ubuntu18> docker start node4  내컴>vmware>ubuntu18> docker ps  내컴>vmware>ubuntu18> 웹브라우저로 확인한다  내컴>vmware>ubuntu18> vi my\_server.js (내용을 Hello 은수 ^^~! 가 출력되도록 고쳐본다)  내컴>vmware>ubuntu18> 웹브라우저를 새로 고침하면 반영되어 있다 |

(Architecture diagram)

