Docker

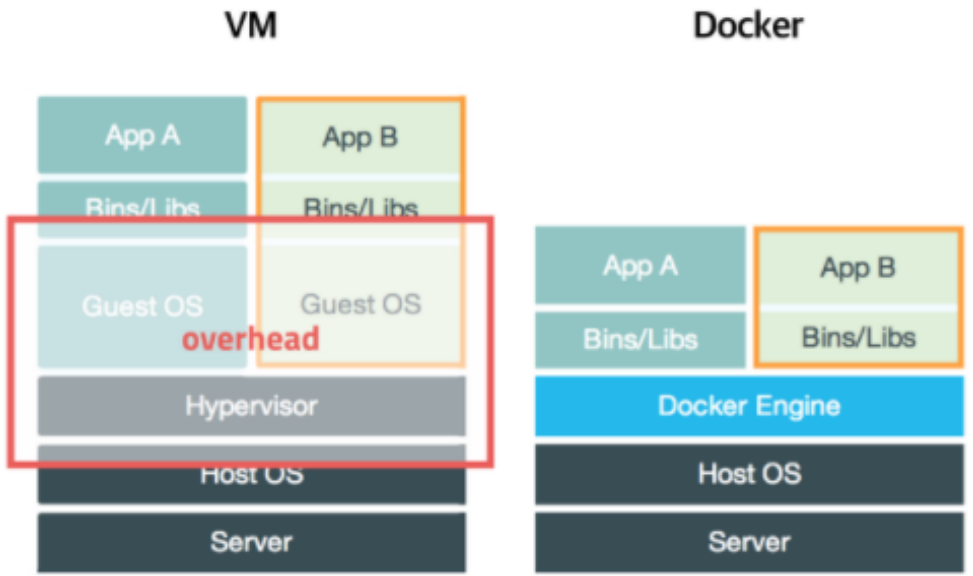
Docker

도커는 **컨테이너 기반의 오픈소스 가상화 플랫폼**
컨테이너는 격리된 공간에서 프로세스가 동작하는 기술

Docker를 이용한 가상 환경 플랫폼

- 이미지 및 컨테이너로 구성
- 설정된 값을 이미지로 구성하여 컨테이너에서 실행
- Guest OS가 없어 VM 대비 월등한 실행 속도
- 하드웨어 비가상화로 메모리 엑세스, 성능 향상
- Docker 이미지를 활용, 동일한 환경 재현 용이
- 서버 환경에 대한 버전 관리 가능
- 웹서버 환경 구축과 관련 반복적인 설정 작업 축소

https://sudarlife.tistory.com/m/entry/%EB%8F%84%EC%BB%A4%EA%B0%80-%EB%AC%B4%EC%97%87%EC%9D%B8%EA%B3%A0-%EA%B0%84%EB%8B%A8%ED%95%98%EA%B2%8C-%EC%95%8C%EC%95%84%EB%B3%B4%EA%B3%A0-%EA%B0%80%EC%9E%90-%EA%B0%9C%EB%85%90-%EC%A0%95%EB%A6%AC

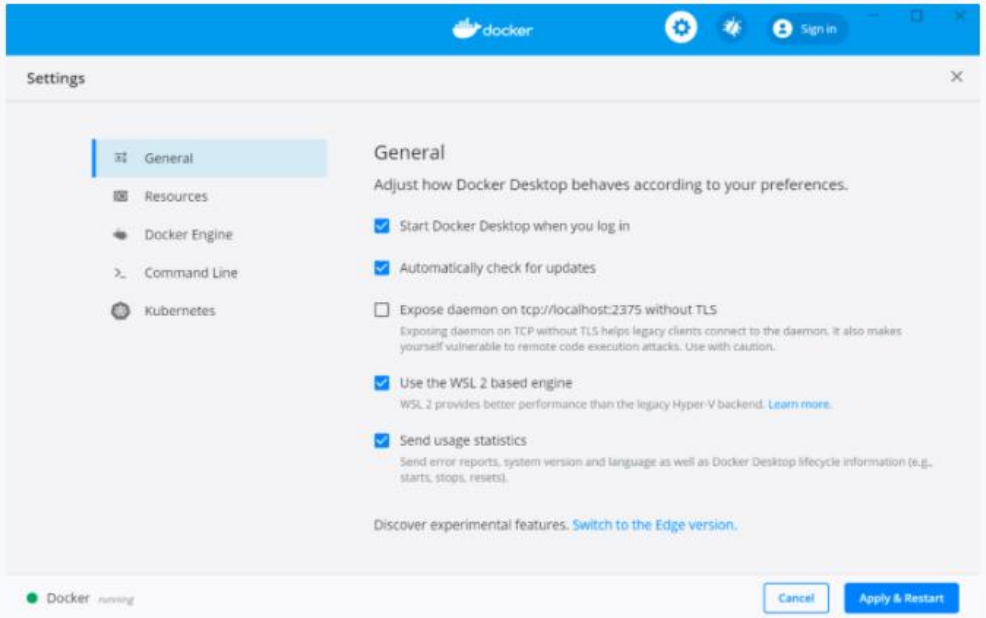


VM과 Docker 비교

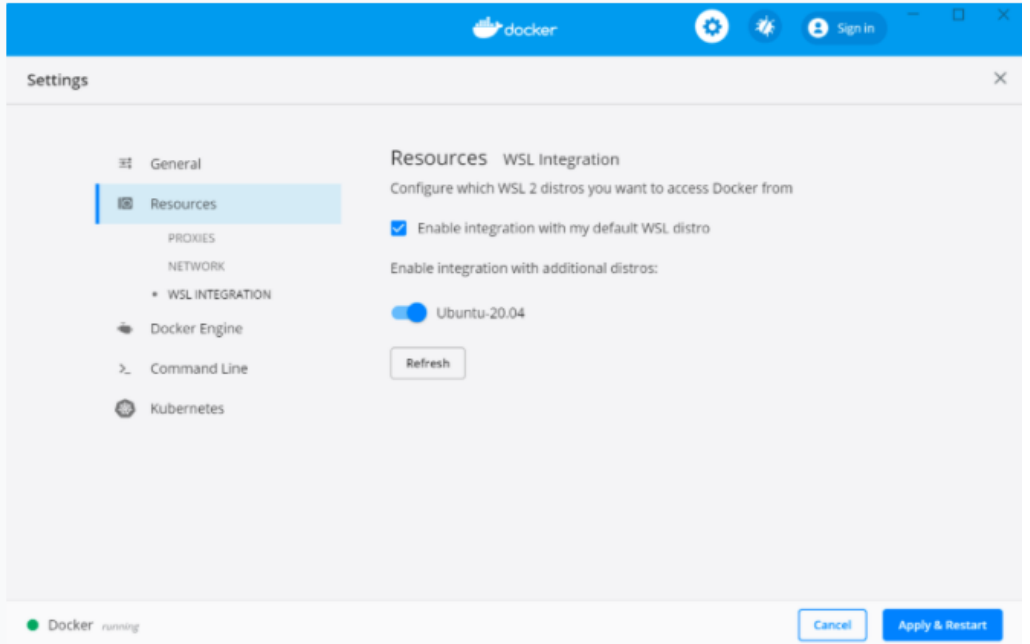
Docker – WSL2 에서 도커 데스크탑으로 서버 실행

- Docker Desktop for Windows - Docker Hub

도커 데스크탑을 설치하고 설정 페이지의 **General** 탭에서 **Use the WSL2 based engine** 옵션을 체크해줍니다.



Use the WSL2 based engine 옵션을 체크해줍니다



Resource -> WSL Integration에서 활성화된 리눅스 배포판을 확인합니다

도커 데스크탑을 설치하고 정상적으로 설정되어있다면, 바로 WSL2 우분투 터미널에서 도커 명령어를 사용할 수 있습니다.

Docker - 실습

도커 이미지는 도커 컨테이너를 만들기 위한 템플릿. 우분투 같은 운영 체제로 구성된 파일 시스템은 물론 컨테이너 위에서 실행하기 위한 애플리케이션이나 그 의존 라이브러리, 도구에 어떤 프로세스를 실행할지 등의 실행환경의 설정 정보까지 포함하는 아카이브

도커 이미지 빌드 > 이미지 허브 저장 > 이미지 내려 받기 > 실행

- docker image build -t example/helloworld (-t옵션으로 이미지명 지정)
- docker images
- docker images push amicokb/helloworld
- docker image pull amicokb/helloworld
- docker run -t --rm amicokb/helloworld
- docker search ubuntu : 이미지 검색하기
- docker pull 이미지이름 : 이미지 다운로드
- docker rmi 이미지이름 : 이미지 삭제 (컨테이너가 실행중이면 삭제 불가)
- docker run [options] 이미지이름 : 이미지 실행
- options 참고 : https://www.daleseo.com/docker-run/
  https://house-of-e.tistory.com/entry/docker-%EC%9E%90%EC%A3%BC-%EC%82%AC%EC%9A%A9%ED%95%98%EB%8A%94-%EB%AA%85%EB%A0%B9%EC%96%B4

Docker - 실습

도커 컨테이너는 실행 중, 정지, 파기의 3가지 상태를 갖는다.
지정된 도커 이미지를 기반으로 컨테이너가 생성되면 이 이미지를 생성했던 Dockerfile에 포함된 CMD 및 ENTRYPOINT 인스트럭션에 정의된 애플리케이션이 실행된다.
컨테이너를 정지하거나 종료된 경우에는 컨테이너가 자동으로 정지 상태가 된다. 다시 실행 가능
한 번 파기한 컨테이너는 다시는 실행할 수 없다.

컨테이너 실행

- docker (container) run -it --name ubuntu ubuntu:latest /bin/bash: 바로 컨테이너 들어가서 작업하기
  ※ 컨테이너 생성/실행 -i 상호 입출력, -t bash 쉘을 사용
- docker exec -it ubuntu /bin/bash : 컨테이너 들어가서 작업하기
- docker stop 컨테이너이름 : 실행중인 컨테이너 중지
- docker stop djp1-web-1 : 컨테이너 재시작
- docker container restart djp1-web-1
- docker start ubuntu : 컨테이너 실행
- docker ps : 현재 실행중인 프로세스들(컨테이너)
- docker ps -al : 종료된 컨테이너까지 출력
- docker stop 컨테이너이름 : 실행중인 컨테이너 중지
- dokcer rm 컨테이너이름 : 컨테이너 삭제 (컨테이너가 실행중이면 삭제 불가)

Docker - 실습

컨테이너 실행

- docker run --rm -it python:3.8
- >>> import sys
- >>> sys.platform
- >>> sys.version
- docker run --rm -it python:3.8 sh 컨테이너를 정지할 때 함께 삭제
- docker run -d --publish 8080:80 --name mynginx nginx
- docker stop mynginx 실행중인 컨테이너 종료
- docker rm mynginx
- mkdir html
- vim html/index.html
- 현재 경로에 html/index.html을 생성하고 컨테이너 경로로 마운팅
  docker run --rm -p 9000:80 -v `pwd`/html:/usr/share/nginx/html nginx
- docker ps --filter "name=echo1“ 특정 조건을 만족하는 컨테이너 목록
- docker ps –a 이미 종료된 컨테이너를 포함한 컨테이너 목록
- docker restart echo1 정지 상태 컨테이너를 재시작
- docker rm -f echo1 현재 실행중인 컨테이너 삭제

Docker - 실습

- docker container restart : 컨테이너 재시작하기
- docker container exec –it echo sh : 컨테이너 내부를 조작

포트포워딩

- docker container run -d -p [호스트 포트]:[컨테이너 포트] --name nx2 nginx
- docker run -d -p 80:80 docker/getting-started
- docker run -d --name tc -p 80:8080 --rm consol/tomcat-7.0 : 실행 종료 후 삭제
  curl http://localhost:80/
-

Docker - 이미지

도커 이미지는 도커 컨테이너를 만들기 위한 템플릿
- 우분투 같은 운영 체제로 구성된 파일 시스템
- 컨테이너 위에서 실행하기 위한 애플리케이션이나 그 의존 라이브러리
- 도구에 어떤 프로세스를 실행할지 등의 실행 환경의 설정 정보

도움말 확인 : docker image –help, docker image build –help
- -t : 이미지명과 태그명을 붙이는 것
- -f : 파일명으로 된 Dockerfile
- --pull : 베이스 이미지를 강제로 받아온다

이미지를 외부에 공개하기
docker image tag example/echo:latest amicokb/echo:latest : 이미지의 네임스페이스 변경
docker image push amicokb/echo:latest : 도커 허브 로그인 된 상태에서 도커 허브에 등록
공개 리포지토리에 등록할 이미지나 Dockerfile에는 패스워드나 API 키 같은 민감한 정보가 포함되지 않도록 유의

docker image prune : 태그가 붙지 않은 모든 이미지를 삭제

docker system prune : 사용하지 않는 도커 이미지 및 컨테이너, 볼륨, 네트워크 등 모든 도커 리소스를 일괄 삭제

Docker

- docker run -d -p 80:80 docker/getting-started
- docker run hello-world
- docker run -it ubuntu bash
- 우분투 버전 확인 : lsb_release –a, cat /etc/issue
- uname –a

- docker --version

- docker cp test.txt tc:/
- docker exec -it tc cat /test.txt
- docker cp tc:/test.txt ./test2.txt
- docker ps -a –q
- docker stop `docker ps -a -q`
- docker rm `docker ps -a –q`
- docker run -d --name tc -p 80:8080 --rm consol/tomcat-7.0 : 실행 종료 후 삭제
- docker stop tc

Docker - 컨테이너

docker container run 명령 옵션
- -i : 컨테이너 쪽 셀에 들어가서 명령을 실행
- -t : 유사 터미널 기능을 활성화
- --rm : 컨테이너 종료시 컨테이너를 파기
- -v : 호스트와 컨테이너 간에 디렉토리나 파일을 공유하기 위해 사용
- -p : 포트포워딩
- -q : 컨테이너 ID(축약형)만 추출
- -a : 이미 종료된 컨테이너를 포함한 컨테이너 목록을 볼 수 있다
- docker container ls –q : 컨테이너 ID 만 출력
- docker container ls --filter "name=awesome*“ : 특정 조건을 만족하는 컨테이너 목록 출력

- docker container run -d example/echo:latest : -d 업션을 붙여 백그라운드로 실행
- docker container restart awesome-pascal : 정지 상태에서 컨테이너 재시작
- docker container ls : 컨테이너 목록 확인
- curl http://localhost:56405 : command line tool(& library)이고, URL을 이용해 데이터 전송

Docker - 컴포즈

- 도커 컨테이너로 시스템을 구축하면 하나 이상의 컨테이너가 서로 통신하며 그 사이에 의존관계가 생긴다. 컨테이너의 동작을 제어하기 위한 설정 파일이나 환경 변수를 어떻게 전달할지 컨테이너의 의존관계를 고려할 때 포트포워딩을 어떻게 설정해야 하는지 등의 요소를 적절히 관리하는데 필요한 것이 도커 컴포즈.
- 도커 컴포즈는 yaml 포맷으로 기술된 설정 파일로 여러 컨테이너 실행을 한 번에 관리할 수 있게 해준다.
- docker-compose.yml 파일이 위치한 디렉토리에서 이 정의에 따라 여러 컨테이너를 한꺼번에 시작하려면 docker-compose up 명령을 사용

- docker-compose up –d : 여러 컨테이너를 한꺼번에 실행
- docker-compose down : docker-comppose.yml에 정의된 모든 컨테이너가 정지 혹은 삭제
- docker-compose up –d –build : 도커 이미지를 강제로 다시 빌드하게 함

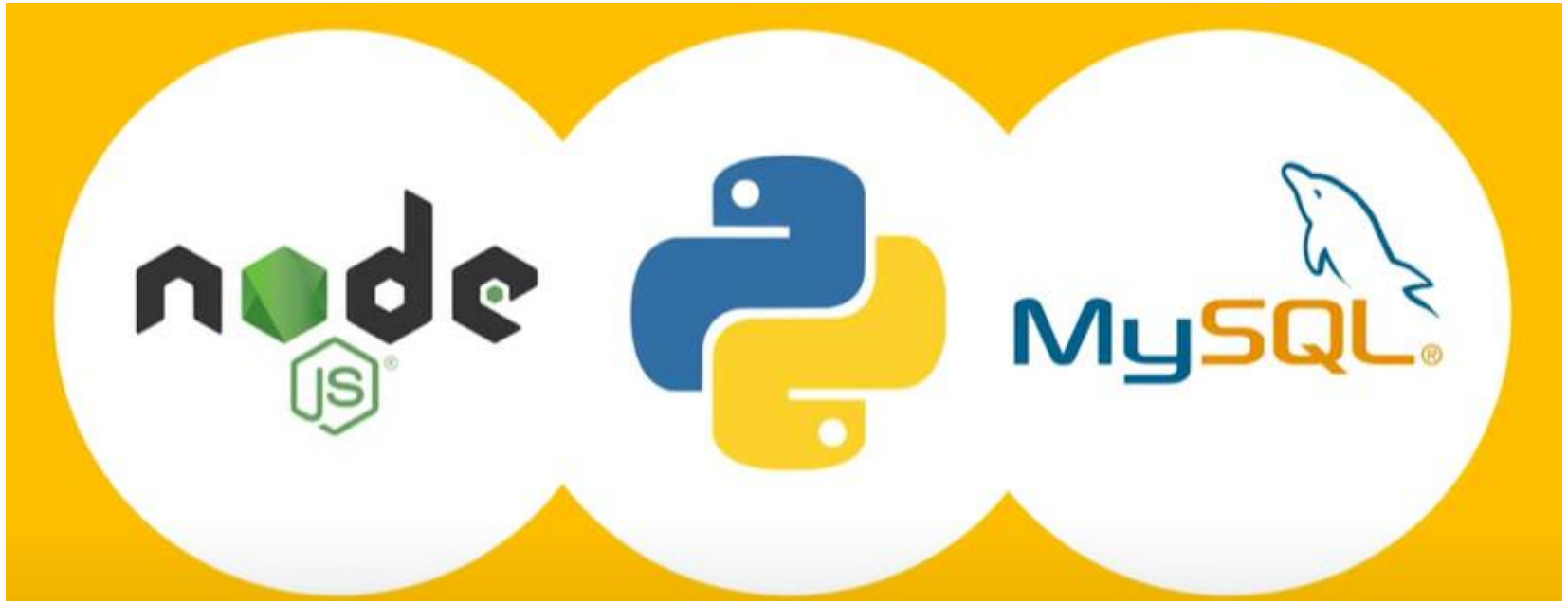
Docker – 스웜을 이용한 실전 애플리케이션 개발

MySQL Mater – Slave 구조 아키텍쳐
Master에서는 DDL, DML 쿼리 및 약간의 SELECT를 수행하고 Slave에서는 Select Quary만 지원
Master 한대에 Slave 여러대를 두어 부하를 분산

Docker - 실습

- 도커 허브 로그인
- docker login –u your_docker_id –p
- docker logout

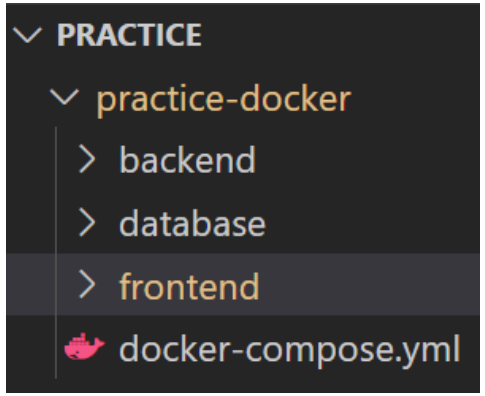
Docker - 실습

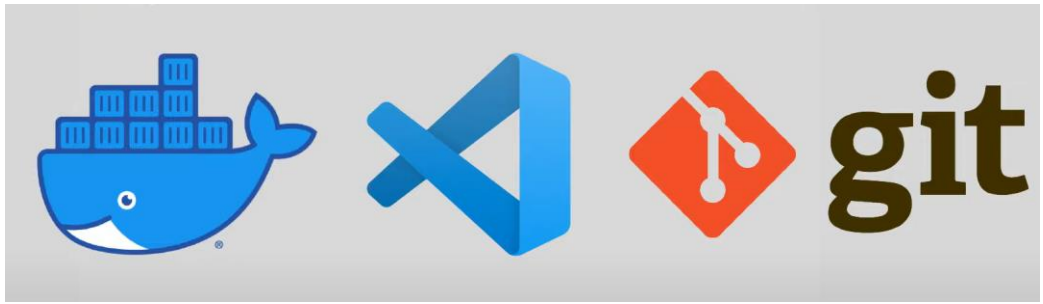


Front-end

Back-end
Flask

Database
My-SQL





Docker - 실습

- ubuntu에서 마운팅 /mnt$ cd ./c/Users/kevin/docker/practice-docker
- C:\Users\kevin\docker 폴더 만들어준 후 vscode 실행
- new terminal > powershell > git clone
  git clone https://gitlab.com/yalco/practice-docker.git

Docker - 실습

dockerfile : node 이미지에 덧붙여서 이 공식 이미지를 개조, 튜닝해서 만드는 것임

FROM node:12.18.4

이미지 생성 과정에서 실행할 명령어로 이미지를 빌드할 때 실행(애플리케이션 업데이트/배치)
RUN npm install –g http-server

NPM는 Node Package Manager의 약자이다. 자바스크립트 패키지 매니저이고 NodeJS에서 사용할 수 있는 모듈들을 패키지화하여 모아둔 저장소 역할을 하며 설치/관리를 수행할 수 있는 CLI를 제공

이미지 내에서 명령어를 실행할(현 위치로 잡을) 디렉토리 설정
WORKDIR /home/node/app

도커가 동작 중인 호스트 머신의 파일이나 디렉토리를 도커 컨테이너 안으로 복사
COPY 복사할 파일 /디렉토리

컨테이너 실행시 실행할 명령어로 컨테이너를 시작할 때 한 번 실행(애플리케이션 자체를 실행)
CMD ["http-server", "-p", "8080", "./public"]

Docker - 실습

이미지 생성 명령어 (현 파일과 같은 디렉토리에서). –t 이미지명 지정
docker build -t {이미지명:태그명} .(현재 작업 디렉토리)

컨테이너 생성 & 실행 명령어
docker run --name {컨테이너명} -v $(pwd):/home/node/app -p 8080:8080 {이미지명}

docker build -t frontend-img .
docker run --name frontend-con -v "$(pwd):/home/node/app" -p 8080:8080 frontend-img

docker stop $(docker ps -aq)
docker rm $(docker ps -aq)

docker run --name database-con -p 3306:3306 -d database-img
docker logs -f database-con

Docker - 실습

- 거시적 설계도인 docker-compose

  docker-compose up

  docker-compose down

- docker system prune –a : 사용되지 않는 모든 도커 요소(컨테이너, 이미지, 네트워크, 볼륨) 삭제

- 도커의 기본 명령어 : https://www.yalco.kr/36_docker/