SAMSUNG SW ACADEMY FOR YOUTH

對外秘

일일 과제 명세서

프론트엔드 개발자도 Docker 를 써봅시다

對外秘

목차

1. 과제 개요

본 과제는 SSAFY 의 2 학기에 앞서 명세서를 보고 시스템을 모의 구축해보는 과제입니다. 프로젝트를 시작함에 있어 가장 기본이 되는 내용이지만, 관심이 없으면 알기 어려운 것이 시스템 입니다. 매 프로젝트마다 반드시 필요하지만 잘 알 수 없어서 시간을 많이 빼앗아 가던 시스템설정, 그 중에서도 가장 많이 찾으며, 세팅하는데 시간을 소모하는 Docker 의 기본개념을 백엔드 개발자들뿐 아니라 프론트엔드 개발자들도 사용 할 수 있도록 학습해 보도록 하겠습니다.

* Docker 가이드는 따라하기는 매우 쉽지만, 전반적인 Network 와 보안, Server 의 Stable Resource 등을 눈치채는 센스와 CS 지식을 기반으로 한 검색능력이 점점 중요해지기 때문에, 꾸준한 CS 학습을 요구하며, 전체적인 데이터의 흐름을 볼 줄 아는 시각이 필요합니다. 이 부분을 중요시하며 작성해봅시다.

※ 참고자료

분류	제목	URL
Docker	Get Start	https://www.docker.com/get-started/
Docker	Install Docker Engine on Ubuntu	https://docs.docker.com/engine/install/ubuntu/
Docker image	Docker Image 생성하기	0 부터 시작하는 Docker 공부 - Docker Image 생성하기
Docker image	도커이미지생성	https://ajdkfl6445.gitbook.io/study/devops/docker/make-image
Docker Spec	Docker Default Command Specifications	https://docs.docker.com/engine/reference/builder/

2. 과제 목표

가) **SSAFY 1 학기가 마무리 될 때 관통프로젝트를 경험하셨을 겁니다. 그처럼 통합 프로젝트는 FrontEnd, BackEnd, Infra 등으로 구성되는데, 그중에서 BackEnd 를 Resource 에 있어서 재사용성이 뛰어나고, 효율적이며, 보안적으로 독립적인 프로세스로 동작시키기 위해 Docker 가 사용됩니다.**

나) **Linux 에서 Docker 를 사용하려면 어떤 준비가 필요한지,** Docker 로 사용할수 있는 API Server 에는 어떤것들이 있는지 생각해봅시다.

A. 특히, API 서버를 선정할때는 개발자 본인이 서버 내부 시스템에 대해 얼마나 알고 있는지 파악하는 것이 절대적으로 중요합니다.

B. 따라서 별도의 Solution 등을 사용하는게 아니라면, 가능한 본인이 직접 참여한 프로젝트를 기준으로 개발을 고려하는 것이 가장 좋은데, 그런 의미에서 1 학기 마지막에 개발한 관통 프로젝트는 매우 좋은 사례라 할수 있습니다.

다) **Ubuntu Linux 20.04 LTS 를 설치해봅니다**.

A. 아직도 현업에서는 RedHat 계열, RHEL 이나 CentOS 를 사용하고 있는 경우가 많은데, CentOS 의 제공 서비스 변경이후로는 Ubuntu Server 가 매우 맹렬히 점유율을 올리고 있는 상황입니다.

B. 물론 2022 년 이후 RHEL 과 CentOS 의 대안으로 떠 오르고 있는 Rocky Linux 도 존재하지만, 우리는 Ubuntu Server 를 기준으로 시작하겠습니다. 차후 Rocky Linux 도 익혀두시면 매우 좋습니다.

C. LTS 는 Long-Term Support 의 약자로 한국말로 치환하면 "장기지원" 과 같은 의미가 됩니다.

라) **Ubuntu Linux 20.04 LTS 의 최신 System Update 를 진행합시다.**

A. 24 년 12 월 기준 Ubuntu 20.04 의 최신버전은 "20.04.6" 입니다.

마) **Ubuntu Linux 20.04 LTS 에 Docker 를 설치해봅시다.**

A. 24 년 12 월 기준 Docker Engine 최신버전은 "27.4.0" 입니다.

바) **설치한 Docker 에 본인의 1 학기 관통프로젝트의 서버 파트를 Docker 로 기동해 봅시다.**

A. 신규 프로젝트를 Docker 로 기동시키기 위해서 선행해야할 작업이 있습니다.

B. 해당 선행 작업 및 Dockerizing 까지가 과제의 핵심입니다.

3. 동작 코드

1. 가) - 다) Ubuntu Linux 20.04 LTS 설치까지는 설명하지 않습니다.
2. 라) Docker Install Ready

Ubuntu Linux 20.04 LTS 최신 버전 Update & Upgrade & docker installation

라) A - 0	Docker 를 설치하기에 앞서, Ubuntu Linux 20.04 LTS 를 최신버전으로 Update 합니다. sudo = SuperUser DO (관리자가 명령한다 - 정도의 의미로 이해하시면 됩니다.) apt-get = ubuntu linux 의 배포관리 솔루션 명칭 update = ubuntu linux 에서 update 대상을 확인하는 명령어 upgrade = update 에서 확인한 대상들을 모두 upgrade (update 실행) 하는 명령어 sudo apt-get upgrade 명령어 입력후, 설치하시겠습니까? 질문이 나오면 Y 입력
console	$> sudo apt-get update … $> sudo apt-get upgrade
라) A - 0	Ubuntu Linux upgrade 까지 완료후 Ubuntu linux 의 버전을 확인해야 하는데, 몇가지 방법이 dlTwlaks, 보통 lsb_release -a 명령어를 많이 사용합니다.
console	$> **lsb_release -a** No LSB modules are available. Distributor ID: Ubuntu Description: Ubuntu 20.04.6 LTS Release: 20.04 Codename: focal
라) A - 1	Docker 를 사용하기 위한 기본 구성 설치
console	$> sudo apt-get install \ > ca-certificates \ > curl \ > gnupg \ > lsb-release; Reading package lists... Done Building dependency tree Reading state information... Done lsb-release is already the newest version (11.1.0ubuntu2). lsb-release set to manually installed. ca-certificates is already the newest version (20230311ubuntu0.20.04.1). ca-certificates set to manually installed. curl is already the newest version (7.68.0-1ubuntu2.21). curl set to manually installed. gnupg is already the newest version (2.2.19-3ubuntu2.2). gnupg set to manually installed. 0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

3. 마) Docker Install

마) A – 1	Docker's 공식 GPG key 추가
Console	`$> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg`
마) A – 2	Stable repository 설정
Console	`$> echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null`
마) A – 3	Docker Engine 설치
Console	`$> sudo apt-get install docker-ce docker-ce-cli containerd.io`
마) A – 4	Docker Engine 버전 확인
Console	`$> sudo docker version`
마) A – 5	Ubuntu Linux System 에 Docker 등록
Console	`$> sudo systemctl enable docker` `sudo systemctl start docker`

4. 바) Create Docker Image from SpringBoot Server

기존의 관통프로젝트 기준이기 때문에, SpringBoot Server 는 이미 존재하고 있다 간주한다.

바) A - 1	Dockerfile 만들기 docker build 라는 명령어를 통해서 Docker 가 Dockerfile 을 읽어, 자동으로 도커 이미지를 빌드한다. Dockerfile 은 FROM 절부터 순차적으로 읽어 처리하게 된다.의 위치는 프로젝트 최상위 루트 경로에 위치해야하며, 파일이름은 Dokcerfile 이라고 강제해야하고, Dockerfile 은 FROM 절부터 순차적으로 읽어 처리하게 되니, Command 순서에 주의.

Dockerfile	`FROM openjdk:17-jdk-alpine`
	`CMD ["./mvnw", "clean", "install"]`
	`ARG JAR_FILE_PATH=target/*.jar`
	`COPY ${JAR_FILE_PATH} app.jar`
	`ENTRYPOINT ["java", "-jar", "app.jar"]`

바) A - 2	Dockerfile 해설
descryption	1) OpenJDK 를 17 버전 (LTS) 로 사용하겠다는 의미. alpine 은 필요한 데이타만 존재하는 경량화버전이며, 보안이 강화되었기에, Build 시에 많이 사용된다. 2) CMD 는 Commnad 의 약자로, 옵션기능이며, 상황에따라 생략 가능하다. DockerFile 내에서는 한번만 사용가능하며 해당내용은 Maven build, install command 를 순차적으로 실행하라는 의미이다. .jar 파일을 생성하려는 목적인경우, install 대신에 package 도 사용가능하다. 3) 대상 jar 파일의 프로젝트 내 위치. 4) ARG 에서 지정한 JAR_FILE_PATH 의 파일명을 app.jar 로 Command 에서 지정. 5) Maven Build 된 app.jar 파일을 실행하도록, 컨테이너 시작시 실행되는 Scrypt.
바) A - 3	Docker Engine Terminal Excute Docker Engine 에서 Dockerfile 을 위부터 순차적으로 읽어 build 한다. -t 옵션을 사용하면, name:tag 가 적용된다.
console	`$> docker build -t {Dockerfile 경로}/docker-dev:dev.0.0.1`
바) A - 4	Docker image 생성여부 확인
console	`$> docker images`
바) A - 5	Docker image 생성 성공시 console 메시지.

console	`$> docker images`
	`REPOSITORY  TAG  IMAGE ID  CREATED  SIZE`
	`docker-dev  dev.0.0.1  571207645393  1 min ago  402MB`

바) A - 6	Docker image 실행. 외부포트 8080 : 내부포트 8080 으로 지정하여 실행하였다. 각 포트번호는 변경가능. 내부포트는 이슈가 적지만, 외부포트는 점유프로세스를 확인하며 사용상에 주의할 것.
console	`$> docker run -p 8080:8080 docker-dev`

5. 사) Docker Hub 회원가입 및 image push (Upload)

Docker Hub 회원가입절차는 구글링 자료가 많으므로 Skip.

사) A - 1	Docker Hub Repository 생성 Create Repository 에서 계정 -> Repository 주소 -> 공개여부 (Public, Private) 선택하여 진행 이 예시는 tester/docker-dev 라 생성하였다고 간주하고 설명한다.
descryption	Create Repository 관련자료가 많으므로 Skip. 단, Repository 생성시, {계정}/{RepositoryName} 구조임을 명심할 것.
사) A - 2	Docker image push (Upload)
descryption	`$> docker push tester/docker-dev`
사) A - 3	Docker Hub 에서 image 검색하여, Upload 여부 확인
descryption	

덧)
`$> docker run -p 8080:8080 docker-dev` 라고 실행할 경우,
local 장비에 Docker image 가 있다면, `docker-dev` 이미지가 바로 실행되겠지만,
local 장비에 Docker image 가 없는 경우, Docker Hub 에서 동일한 imageName {계정}/{RepositoryName} 으로 내려받아 실행된다.
본인의 프로젝트가 아닌 경우, Docker Hub 가 가장 최신인 경우가 많으므로,
가급적이면 `$> docker rmi {dockerImageName}` 으로 구버전 Docker image 를 삭제하고,
새로 내려받아 실행할 것.