**MISSION**

**1. Docker의 개념 이해하기**

**목표**: Docker가 무엇인지, 왜 사용되는지 이해한다.

**활동**:

• Docker의 개념, 특징, 그리고 사용 사례를 조사

• Docker와 가상 머신(Virtual Machine)의 차이점을 정리

**CONTENT**

1. Docker의 개념

* Docker란 개발자가 컨테이너를 구축, 배포, 실행, 업데이트 및 관리할 수 있게 해주는 오픈소스 플랫폼.
* 소프트웨어를 소프트웨어의 실행에 필요한 모든 것을 포함하는 완전한 파일 시스템 안에 감싼다.

1. Docker의 특징

* Docker는 확장성과 이식성이 좋아 설치만 되어있으면 어디서든 컨테이너 실행 가능하다.
* Docker는 기본적으로 Linux를 제공하기 때문에 Window나 MacOS에서는 클라우드 및 가상머신에 설치된다.
* Docker를 사용하면 로컬 테스트 환경과 프로덕션 환경이 다르더라도 동일한 구조를 갖도록 하며 실행하는데 문제가 없다.
* 프로세스 격리 기술을 사용하여 하나의 서버에서 여러 개의 서비스를 관리한다.

1. Docker의 사용사례

* 구름 IDE
  + 구름 IDE는 웹상에서 소프트웨어 개발을 지원하는 통합개발환경
  + 사용자는 원하는 개발 스택을 선택하면 개발환경을 자동으로 구축해줌
  + 이때, Docker 컨테이너가 사용됨!

1. Docker 와 가상머신(VM)의 차이

* 가상머신(VM)은 하이퍼바이저를 활용하여 논리적으로 분리된 공간에서 VM이라는 독립된 가상의 공간을 만들고 호스트 시스템에서 VM에 깔린 게스트OS를 구동
* Docker는 호스트OS의 커널을 각각의 애플리케이션이 공유하는 형식
* 둘의 가장 큰 차이점

GuestOS의 존재 유무 = HostOS의 커널을 공유 여부 차이!( VM은 GuestOS를 사용하여 각각의 커널을 사용하지만, Docker는 GuestOS를 사용하지 않고 HostOS의 커널을 공유함)

* VM과 비교하였을 때 Docker의 장점은?
  + GuestOS를 사용하면서 생기는 자원 사용량의 오버헤드를 줄일 수 있음

**MISSION**

**2. Docker의 주요 구성 요소 조사하기**

**목표**: Docker를 구성하는 주요 요소들에 대한 이해를 높인다.

**활동**:

• Docker의 주요 구성 요소(예: Docker Engine, Docker Images, Docker Containers, Docker Hub, Docker Compose 등)를 조사하고 각각의 역할을 정리

• 각 요소의 기능과 실제 사용 예시

**CONTENT**

1. Docker의 주요 구성 요소의 개념/역할/사용 예시

* Docker Engine
* 개념: 클라이언트-서버 아키텍처를 따르는 애플리케이션
* 역할: 클라이언트-CLI 🡪 REST API 🡨서버-Docker daemon,
* 사용예시: 도커엔진을 활용하여 사용자의 적절한 명령어에 따라 Image를 빌드하고, Containers를 실행시킨다.
* Docker Images
  + 개념: 이미지는 **컨테이너 실행에 필요한 파일과 설정 값 등을 포함하고 있는 것**으로 상태값을 가지지 않고 변하지 않는다.
  + 역할: Docker Images를 빌드하여 Docker Containers를 실행할 수 있다.
  + 사용예시: “docker images build” 명령어를 사용하여 이미지를 빌드한다.
* Docker Containers
  + 개념: 애플리케이션과 그것을 실행하는 데 필요한 모든 것을 격리된 공간에서 프로세스가 동작하도록 패키징하는 기술
  + 역할: 이미지로 컨테이너를 생성하면 이미지의 목적에 맞는 파일이 들어있는 파일 시스템과 격리된 시스템 자원 및 네트워크를 사용할 수 있는 독립된 공간이 생성된다.
  + 사용예시: “docker run -t [이미지명:태그명] [실행 명령어]를 사용하여 컨테이너를 실행한다.
* Docker Hub
  + 개념: Docker Images를 공유하는 클라우드 기반 저장소
  + 역할: 자신이 만든 Docker Images를 push 하거나 다른 사람이 만든 Docker Images를 pull 받을 수 있음.
  + 사용 예시: 자신이 만든 개발환경이 구축된 컨테이너를 실행시키기 위한 이미지를 “docker push” 명령어로 공유하거나, “docker pull”을 사용해 받아올 수 있다.
* Docker Compose
  + 개념: 여러 개의 Docker Container들을 하나의 서비스로 정의하고 구성하여 하나의 묶음으로 관리할 수 있는 애플리케이션을 만드는 것
  + 역할: 의존성이 높은 여러 개의 컨테이너를 하나의 묶음으로 만들어서 관리하기 간편하게 한다.
  + 사용 예시: SpringBoot 를 실행하기 위한 컨테이너와 SQL을 실행하기 위한 컨테이너가 있을 경우 Docker Compose를 사용하여 멀티-컨테이너 환경을 구동할 수 있다.

**MISSION**

**3. Docker 설치 및 기본 명령어 학습**

**목표**: Docker를 실제로 설치하고 기본 명령어를 익힌다.

**활동**:

• 자신의 개발 환경에 Docker를 설치

• 예시 명령어 학습   
: docker run, docker ps, docker stop, docker rm, docker images, docker pull, docker rmi 등등

**Content**

1. Docker 설치

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1. Docker 기본 명령어 정리
2. 1. **docker run**: 새 컨테이너를 생성하고 실행합니다.
   * 예: docker run -d -p 8080:80 nginx
3. **docker stop**: 실행 중인 컨테이너를 중지합니다.
   * 예: docker stop <container\_id>
4. **docker start**: 중지된 컨테이너를 다시 시작합니다.
   * 예: docker start <container\_id>
5. **docker restart**: 컨테이너를 재시작합니다.
   * 예: docker restart <container\_id>
6. **docker rm**: 컨테이너를 삭제합니다.
   * 예: docker rm <container\_id>
7. **docker rmi**: 이미지를 삭제합니다.
   * 예: docker rmi <image\_name>
8. **docker pull**: Docker Hub 또는 다른 레지스트리에서 이미지를 다운로드합니다.
   * 예: docker pull nginx
9. **docker push**: 이미지를 Docker Hub 또는 다른 레지스트리에 업로드합니다.
   * 예: docker push <username>/<image\_name>
10. **docker images**: 로컬 시스템에 있는 이미지 목록을 보여줍니다.
11. **docker ps**: 실행 중인 컨테이너 목록을 보여줍니다.
    * -a 옵션을 추가하면 중지된 컨테이너를 포함한 모든 컨테이너를 보여줍니다.
12. **docker logs**: 컨테이너의 로그를 출력합니다.
    * 예: docker logs <container\_id>
13. **docker exec**: 실행 중인 컨테이너 내부에서 명령어를 실행합니다.
    * 예: docker exec -it <container\_id> /bin/bash
14. **docker build**: Dockerfile을 사용하여 이미지를 생성합니다.
    * 예: docker build -t <tag\_name> .
15. **docker network**: Docker 네트워크 관련 명령어들입니다.
    * 예: docker network ls, docker network create
16. **docker volume**: Docker 볼륨 관련 명령어들입니다.
    * 예: docker volume create, docker volume inspect
17. **exit  :**빠져나오기
18. 실습 정리
19. Docker pull nginx – 도커 이미지 도커 허브에서 pull 받아오기

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

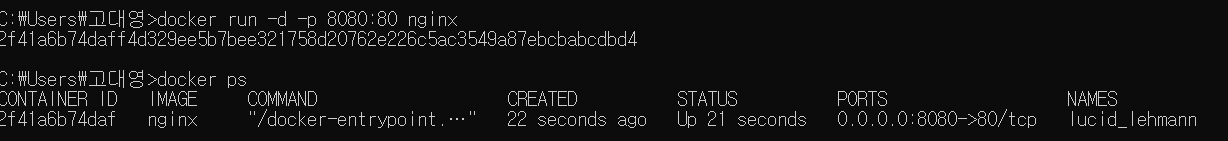
자동 생성된 설명

1. Docker images – docker image 확인하기

텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

1. Docker run , Docker ps – 이미지 빌드하여 컨테이너 생성 , 실행중인 컨테이너 확인



1. Docker stop, Docker ps – 컨테이너 중지, 실행중인 컨테이너 확인

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

1. Docker ps -a , Docker rm, Docker ps -a- 모든 컨테이너 확인, 컨테이너 삭제, 모든 컨테이너 확인

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

1. Docker images, Docker rmi, Docker images- 도커 이미지 확인, 도커 이미지 삭제, 도커 이미지 확인

텍스트, 스크린샷, 폰트, 흑백이(가) 표시된 사진

자동 생성된 설명

**MISSION**

**3-1. Docker Hub 실습**

1. **Docker Hub 개요 조사**:

• Docker Hub의 정의, 주요 기능, 사용 사례를 조사합니다.

1. **Docker Hub에서 이미지 다운로드 및 컨테이서 실행**

• Docker Hub 에서 public 이미지 ( ex: Nginx 등 ) 을 받고 자신의 로컬 pc 에 컨테이너를 직접 실행

**CONTENT**

1. Docker Hub

* 정의: 컨테이너 이미지의 원격 저장소
* 주요 기능: 도커 컨테이너 이미지를 빌드, 저장 및 공유할 수 있는 중앙 리포지토리로 사용된다.
* 사요 사례: [docker pull ‘이미지명/태그’] 명령어를 사용하여 docker hub로부터 이미지를 받아올 수 있다.

1. Docker Hub에서 이미지 다운로드 받고 컨테이너 실행
   1. [Docker pull httpd] 명령어를 사용해 httpd 관련 이미지를 pull 받는다.

→ 태그를 지정하지 않는경우 default tag인 latest(최신버전)이 pull 된다.

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

* 1. [docker images] 명령어를 활용해 이미지를 확인한다.

텍스트, 폰트, 번호, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**MISSION**

**3-2. Docker 컨테이너의 포트 포워딩**

**목표**: Docker 컨테이너의 포트 포워딩을 이해하고, 이를 통해 컨테이너에 외부에서 접근할 수 있도록 설정한다.

**활동**:

• 포트 포워딩의 개념을 조사하고, Docker에서 어떻게 설정하는지 학습

예시:

docker run -d -p 8080:80 nginx 명령어를 사용하여 Nginx 웹 서버 컨테이너를 실행하고, 호스트의 8080 포트를 컨테이너의 80 포트에 포워딩하여 웹 서버에 접근

**CONTENT**

1. 포트포워딩이란?

* NAT란?
  + 물리적 주소를 논리적 주소로 매핑 시켜서 IP 할당 시 주소가 부족한 문제를 해결
  + 공인 IP와 사설 IP 로 구분되는데, 이때 Docker 컨테이너는 네트워크 구성 상 NAT 내부망에서 동작하고 사설 IP를 할당 받는다.
  + 즉, 기본적으로 Docker 컨테이너 외부(인터넷망)에서 내부의 접속이 불가능하다.
  + NAT가 동작하는 라우터/게이트웨이의 특정 포트로 유입되는 트래픽을 NAT 내부에서 동작하는 시스템의 특정 포트로 전달하는 기법이 포트 포워딩이다.
  + 포트 포워딩을 통해서 내부망에서 작동하는 Docker Container를 외부망에서 접속할 수 있도록 한다.

1. Docker에서 설정 방법은?

* Docker는 컨테이너 실행 시 -p 옵션을 추가하여 실행함으로써 포트 포워딩 설정을 할 수 있다.
  + docker run -p <host port number>:<container port number>/<protocol> [IMAGE NAME] [OTHER OPTIONS…]
    - host port number: 호스트 시스템에서 사용되는 포트 번호
    - container port number: 컨테이너 내에서 사용되는 포트 번호
    - protocol: 프로토콜 유형 – tcp , udp, sctp 등

1. 실습 – docker pull nginx, docker run -p 8080:80 nginx, docker ps -a , docker restart,

텍스트, 소프트웨어, 폰트, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

**MISSION**

**4. Dockerfile 및 이미지 빌드**

**목표**: Dockerfile을 작성 및 Docker 이미지를 직접 빌드

**활동**:

• Dockerfile의 기본 구조와 명령어들을 조사합니다.

• 간단한 Node.js 애플리케이션을 위한 Dockerfile을 작성해 보고, 이를 통해 Docker 이미지를 빌드   
-> <https://ranna.tistory.com/12> 를 참고하셔서 진행해보셔도 될 것 같습니다.

• 빌드한 이미지를 기반으로 컨테이너를 실행하고, 애플리케이션이 제대로 동작하는지 확인합니다.

**CONTENT**

1. Dockerfile의 기본 구조와 명령어

* Dockerfile이란?
  + Docker file 문법으로 스크립트를 작성하여 이미지를 생성
  + 나만의 이미지를 생성하고 공유할 수 있다
* Docker의 개념
  + Dockerfile을 작성 후 소스경로를 포함하여 빌드 명령을 내리면 해당 경로를 기준으로 빌드가 실행된다. Ex) docker build /your/source/path
  + 빌드에서 제외할 파일은 .dockerignore파일에 명시
  + Docker file 안의 명령어는 독립적으로 실행되어 다음 실행되는 명령어에 영향을 미치지 않는다.
* Dockerfile의 기본구조
  + #을 이용하여 주석처리
  + “명령어 인자” 형식
  + 명령어는 대소문자 구분하지 않으나 가급적 대문자 이용
  + 각 명령어는 위에서부터 순서대로 실행됨
  + 첫 번 째 명령어는 반드시 FROM으로 시작되어야 함
  + ENV 명령어로 환경 변수 선언 할 수 있음 사용할 때에는 ‘$변수명’ OR ‘${변수명}
* Dockerfile의 기본 명령어
  + FROM – ex) FROM <IMAGE>:<TAG>
    - 생성할 이미지의 베이스가 될 이미지를 뜻합니다. FROM 명령어는 Dockerfile을 작성할 때 반드시 한 번 이상 입력해야 합니다.
  + MAINTAINER – ex) MAINTAINER <name>
    - 이미지를 생성한 개발자 정보를 나타냅니다.
  + RUN
    - 이미지를 만들기 위해 컨테이너 내부에서 명령어를 실행합니다.
    - RUN 명령어에 ["/bin/bash", "echo hello" >> test.html"] 같이 입력하면 /bin/bash 셸을 이용해 'echo hello >> test2.html'을 실행한다는 뜻
  + ADD
    - 파일을 이미지에 추가합니다.
    - Dockerfile이 위치한 디렉터리에서 파일을 가져옵니다.
    - ex) ADD test.html /var/www/html 이라면 Dockerfile과 같은 디렉토리에 있는 test.html 파일을 이미지 내부의 /var/www/html에 추가한다는 뜻입니다.
    - ADD보단 COPY 명령어를 주로 사용 권장
  + WORKDIR
    - 명령어를 실행할 디렉터리를 나타냅니다. 배시 쉘에서 cd 명령어를 입력하는 것과 같은 기능을 합니다. 즉, 컨테이너 내부에서 작동을 합니다.
  + EXPOSE -ex) EXPOSE <port> [<port>…]
    - Dockerfile의 빌드로 생성된 이미지에서 노출할 포트를 설정합니다. 그러나 EXPOSE를 설정한 이미지로 컨테이너를 생성했다고 해서 반드시 이 포트가 호스트의 포트와 바인딩되는 것은 아니며, 단지 컨테이너의 80번 포트를 사용할 것임을 나타내는 것뿐입니다.
  + CMD
    - CMD는 컨테이너가 시작될 때마다 실행할 명령어를 설정하며, Dockerfile에서 한 번만 사용할 수 있습니다.
    - 여러 개가 있는 경우 마지막 CMD만 실행됨.
  + COPY – ex) COPY <src>… <dest>
    - 파일 또는 디렉터리를 Docker 컨테이너에 복사.
    - ADD와 달리 URL 지정 불가, 압축을 자동으로 풀어주지 않음.
    - EX) COPY test.sh /root/test.sh
    - <src>에 있는 파일을 <dest>로 복사
    - <src>는 docker build에서 명시한 경로의 하위여야함
    - <dest>는 절대경로

1. 간단한 Node.js 애플리케이션을 위한 Dockerfile 작성하여 이미지 빌드

* Dockerfile 작성
* FROM node:16
* RUN mkdir -p C:\Users\고대영\tools\dockerFile
* WORKDIR C:\Users\고대영\tools\dockerFile
* COPY package.json ./
* RUN npm install
* COPY ./ .
* EXPOSE 8000
* CMD ["node","index.js"]
* 해석
  + - FROM: 인자로 작성한 베이스 이미지를 사용하여 이미지를 생성

RUN: 인자로 작성한 경로에 새로운 디렉터리를 생성하겠다

WORKDIR: 인자로 작성한 디렉터리르 컨테이너 루트 디렉터리로 설정

COPY: package.json을 생성한 WORKDIR 에 복사

RUN: package.json을 기반으로 필요 모듈 설치

COPY: 다른 파일 모두 생성한 WORKDIR 에 복사

EXPOSE: 해당 포트를 8000으로 설정

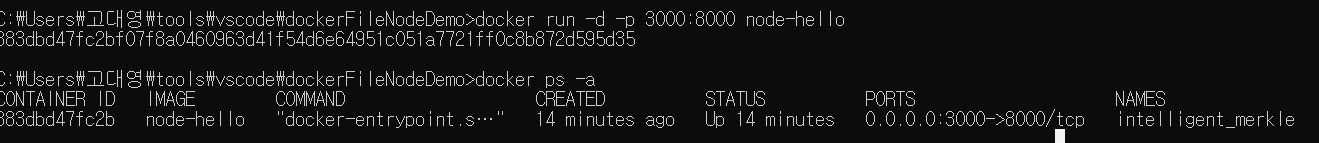
CMD: node index.js 명령어를 사용하여 실행

* 주의할 점
  + - Dockerfile을 작성할 때 파일명 철자가 틀리면 오류남(ex.DockerFile🡪오류)

1. 빌드한 이미지를 기반으로 컨테이너 실행 및 애플리케이션 동작확인

* 빌드 및 컨테이너 실행 명령

1. Dockerfile이 위치한 경로로 이동
2. docker build -t node-hello:latest . (node-hello이름의 latest태그를 가진 이미지를 현재 위치에 빌드하겠다.
3. docker run -d -p 3000:8000 node-hello (접속 포트는 3000으로 컨테이너의 8000 번 포트를 포워딩) 🡪 localhost:3000으로 docker container의 8000번 포트에 접속.



텍스트, 스크린샷, 폰트, 화이트이(가) 표시된 사진

자동 생성된 설명

**MISSION**

**5. Docker Compose 이해 및 실습**

**목표**: Docker Compose를 사용하여 다중 컨테이너 환경을 구축

**활동**:

• Docker Compose의 개념과 사용 사례를 조사

• docker-compose.yml 파일을 작성해보고, 이를 이용해 다중 컨테이너들을 구축 및 관리

**CONTENT**

1. Docker Compose의 개념
   * Docker Compose란 여러 개의 Docker 컨테이너들을 하나의 서비스로 정의하고 구성해 하나의 묶음으로 관리할 수 있는 하나의 애플리케이션
2. Docker Compose의 사용 사례
   * YAML 형식으로 작성된 Docker Compose 파일을 사용해 서비스와 관련 설정들을 정의
   * Docker-compose up 명령어를 사용하여 Docker Compose 파일에 정의된 서비스들을 시작
   * Docker-compose ps, docker-compose logs, docker-compose exec 등의 명령어를 사용하여 서비스 관리
   * Docker-compose down 명령어를 사용하여 Docker Compose 파일에 정의된 서비스들을 중지하고 관련 컨테이너들을 삭제
3. Docker-compose 문법

* 실행 docker-compose up
* 백그라운드에서 실행 docker-compose up -d
* 서비스 중지 docker-compose stop
* 서비스 다운 docker-compose down
* 서비스 다운 후 볼륨삭제 docker-compose down –volumes
* 컨피그 확인하기 docker-compose config실행중인 서비스 컨테이너 명령어 실행하기 docker exec <서비스이름> <명령어>
* 실행중인 서비스 확인하기 docker-compose ps
* 서비스 로그 확인하기 docker-compose logs
* 서비스 로그 지속적으로 프린트하기 docker-compose logs -f
* 서비스 로그 지정해서 확인하기 docker-compose logs <서비스이름>
* 서비스 로그 여러개 지정해서 확인하기 docker-compose logs <서비스이름> <서비스이름> ...

1. YAML 문법 및 작성 방법
   * Version: docker compose 파일의 버전
   * Services: 컨테이너를 실행하기 위한 단위. 서비스 이름🡪서비스 옵션 단위로 작성
   * Build: build할 docker file의 경로
   * Ports: 포트포워딩 지정 옵션. <호스트 포트>:<컨테이너 포트> 순
   * Volume: 바인드 마운트, 볼륨 지정
2. Docker-compose.yml 파일을 통한 다중 컨테이너 구축 관리

* 실습목표
  + Node.js와 nginx 를 docker-compose를 사용하여 하나의 컨테이너로 관리한다.
  1. 루트 프로젝트( DOCKER\_NGINX\_NODEJS) 하위에 dockerFileNodeDemo 디렉터리와 nignx 디렉터리를 만든다.
  2. 텍스트, 스크린샷, 디스플레이, 폰트이(가) 표시된 사진

     자동 생성된 설명텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

     자동 생성된 설명각각의 디렉터리에 Dockerfile을 생성후 작성한다.
  3. 텍스트, 스크린샷, 폰트이(가) 표시된 사진

     자동 생성된 설명프로젝트 루트 디렉터리 하위에 docker-compose.yml 파일을 생성 후 작성한다.
  4. 텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

     자동 생성된 설명Docker-compose up 명령어로 yml 파일을 실행하고 컨테이너에 두개의 애플리케이션이 작동하는지 확인한다.