실증적SW개발프로젝트 주간보고 (##주차)

작성일: 2025/03/30 팀명: ##쿠버조

|  |  |  |
| --- | --- | --- |
| 팀 활동  보고 | 활동일시 | 2025 0324~ 2025 0330 |
| 장 소 | 학교 및 자택 |
| 참석자 | 이경준, 정원희, 이우영, 권기원, 손성곤 |
| 특이사항 | x |
| 이번주  진행사항 | 1. 개발내용  현재 쿠버네티스 및 도커 학습과정중에 있음  2. 팀원별 활동내용  - 권기원 (팀장) : docker-study  docker 공부 기록  도커란?  컨테이너를 관리하기 위해 만들어진 소프트웨어이다. 그렇다면 컨테이너는 무엇일까 큰 서버를 나눠서 사용하기 위한 가상화라는 기술이다.  먼저, 도커를 이해하기 전에 서버와 가상화 기술이 무엇인지 그리고 왜 사용하는지에 대해서 공부하고자 한다.  서버는 무엇일까 서버는 하드웨어와 그 하드웨어에서 실행중인 소프트웨어까지 모드 포함하는 단어이다. 하드웨어만 있어도 서버이고 소프트웨어만 있어도 서버이기때문에 문맥에 따라 이해하면 된다. 클라이언트에 요청에 소프트웨어에 따라 결과값은 달라질리 몰라도 결과를 주는 근본적인 역할은 모든 서버가 동일한 이야기이다  서버는 크게 4가지로 나뉜다. 파일서버, DB서버, 웹서버(WEB) ,웹애플리케이션서버(WAS)  가상화 기술은 실제로 존재하는 컴퓨터가 아니지만 마치 존재하는 것처럼 만들어 주는 기술이다  앤터프라이즈 환경에서는 많은 양의 서버를 운영해야 하는데 크게 3가지로 방법이 나뉜다.  베이멘탈  하이퍼바이저  컨테이너  여기서 하이버바이저와 컨테이너 방식이 가상화 기술을 적용한 서버 운영 방식이다. 각 방식에 대해 요약하자면 아래와 같다.  방식 - 하이퍼바이저 (VM) 컨테이너 (Docker, Kubernetes)실행 방식 -각 VM마다 개별 OS 포함 호스트 OS 공유 (경량)성능 - OS 오버헤드 발생 경량 & 빠름격리 수준 - 강한 보안 격리 프로세스 수준 격리사용 사례 - 서버 가상화 (기업, 클라우드) 마이크로서비스, DevOps] 컨테이너 가상화  컨테이너 가상화는 리눅스 커널이 제공하는 LXC라는 자체 격리 기술에서 출발했다  LXC기술은 커널의 네임스페이스와 CGroups라는 기능을 활용한다  네임스페이스는 리소스를 나누는 기준의 역할을 하고 CGroups는 리소스의 사용량을 배분하는 기술  컨테이너 가상화는 하이퍼바이저 없이 커널이 자체 기술을 활용한 가상화다.  컨테이너는 커널이 있는 HostOS 커널을 공유해서 사용하는데 컨테이너 가상화의 가장 중요한 특징 중 하나이다 . 그렇기 때문에 적은 오버헤드와 빠른 부팅이 가능하다  도커와 아키텍처는라는 소프트웨어는 이 커널의 컨테이너 가상화 기술을 편리하게 사용하기 위한 만들어진 소프트웨어이면 도커를 통해 컨테이너를 만들고 운영할 수 있다. 도커는 이 커널의 가상화 기술을 활용할 수 있게 도와주는 보조 도구이다.  도커의 아키텍처  컨터이너 엔진과 컨테이너 런타임으로 구성돼있음  컨테이너 엔진은 사용자의 요청을 받아서 컨테이너를 관리해주는 역할을 하고 컨테이너 런타임(run c)을 직접 커널과 통신하면서 실제로 격리된 공간을 만드는 역할을 수행함  도커에도 사용자의 명령을 전달해주는 클라이언트와 실제로 컨테이너를 관리해주는 도커 데몬이 존재함  도커데몬은 컨테이너를 관리하는 기능을 제공하기 위해서 api를 클라이언트에게 줌 하지만 클라이언트가 매번 api양식에 맞게 작성하기 번거롭기 때문에 도커는 Docker CLI 라는 클라이언트 툴을 제공함  일반적인 순서  클라이언트 명령어 실행  사용자 명령 api에 맞추어 변환  데몬에 api 요청  커널을 통해 컨테이너 리스트를 불러옴  json형태로 CLI로 넘겨줌  CLI는 클라이언트가 보기 좋게 테이블형태로 화면에 표시  정리하자면 도커는 클라이언트 서버 모델로 실행됨 클라이언트는 CLI, 서버는 도커 데몬으로 구성    - 이경준 : 2주차 실습 진행 : 간단한 도커 이미지 다운로드 후 docker run 실행  Linux환경에서 도커를 재 설치하여 터미널 상에서 도커 실행 후 이미지 다운로드  - 첫 발표를 위해 천세진 교수님 연구실 연구원과 미팅  ->> 미팅내용 요약  도커쿠버네티스>>리눅스AWS-페어데이때탄소배출량이 가장적은 클라우드를 찾는 알고리즘 설계>>관리하는 agent클라우드스케줄링1. 자료기존에있던거 하는거 차별점 왜하는지너무 깊게가진말고각주마다 목표  - 이우영 : 2주차 보고서 내용 간단 요약.  도커의 이론을 학습하고 실습을 하기위해 관련 강의를 들었다.  - 도커의 개념 / 가상화 기술  - 이미지와 컨테이너  - 이미지와 레지스트리  - 이미지 빌드  구체적인 내용은 아래와 같다.  1. 도커(Docker) 개요 / 가상화 기술  도커는 컨테이너 기반의 오픈소스 가상화 플랫폼으로, 애플리케이션을 운영체제 수준에서 격리된 환경에서 실행할 수 있도록 지원한다. 전통적인 가상 머신(VM)과 달리, 도커는 호스트 OS 위에서 직접 컨테이너를 실행하므로 훨씬 가볍고 빠르며, 이식성과 확장성 면에서 강점을 가진다.  주요 구성 요소  이미지 (Image) : 컨테이너 실행을 위한 파일 시스템과 설정이 포함된 정적인 템플릿  컨테이너 (Container) : 이미지를 실행한 상태, 애플리케이션을 구동하는 실행 단위  도커파일 (Dockerfile) : 이미지를 정의하는 코드 기반 명세서  레지스트리 (Registry) : 이미지 저장소, 대표적으로 Docker Hub가 있음  2. 이미지와 컨테이너  도커 이미지는 여러 레이어(layer)로 구성되어 있으며, 각 레이어는 읽기 전용이다. 컨테이너는 이러한 이미지를 기반으로 생성되며, 읽기/쓰기가 가능한 컨테이너 레이어를 위에 추가하여 실행된다.  주요 명령어  명령어 설명  “docker pull 이미지” 이미지 다운로드  “docker run -d -p 8080:80 이미지명” 컨테이너 실행 및 포트 매핑  “docker ps” 실행 중인 컨테이너 확인  “docker exec -it 컨테이너 bash” 컨테이너 내부 접근  “docker stop / docker rm“ 컨테이너 중지 및 삭제  3. 이미지와 레지스트리  이미지는 Docker Hub와 같은 레지스트리(Registry)에 저장되고 공유된다. 사용자는 docker push 명령어를 통해 자신이 만든 이미지를 업로드할 수 있으며, docker pull 명령어로 다른 사용자의 이미지를 내려받을 수 있다.  이미지 이름 구성  레지스트리/계정명/이미지명:태그  예: nginx:1.23.3, myrepo/myapp:latest  또한, 기업이나 개인 서버에서 자체적인 프라이빗 레지스트리를 운영할 수도 있다.  4. 이미지 빌드  도커 이미지는 수동 수정 후 저장하는 커밋 방식과, Dockerfile을 기반으로 자동화된 빌드를 수행하는 방식으로 만들 수 있다.  4.1 이미지 커밋  컨테이너에서 파일을 수정하고 docker commit 명령으로 새로운 이미지를 생성  4.2 Dockerfile 빌드  Dockerfile에 명시된 단계를 따라 이미지를 자동으로 생성  예시 지시어:  FROM: 베이스 이미지 지정  COPY: 파일 복사  RUN: 명령어 실행  CMD / ENTRYPOINT: 실행 명령어 정의  ENV, ARG: 환경 변수 설정  WORKDIR: 작업 디렉터리 설정  4.3 빌드 최적화  레이어 재사용(Caching): 동일한 Dockerfile 단계는 캐시되어 빌드 속도 향상  .dockerignore: 빌드 컨텍스트에서 제외할 파일 정의  멀티스테이지 빌드(Multi-stage Build): 빌드와 실행 단계를 분리하여 이미지 크기를 최소화할 수 있음  - 정원희 : 도커 관련 강의 시청1. 가상화 기술, 이미지, 컨테이너 개념 습득2. 이미지 레지스트리, 이미지 빌드 실습  - 손성곤 : 가상화 기술  하이퍼바이저  -> 프로그램(가상 머신)을 이용해 가상 OS를 만들어 실행 시킴  -> 각각의 종류의 OS마다 각자의 커널을 가짐, 이 커널간의 소통을 위해 사용되는 것이  하이퍼바이저임  컨테이너  -> 하이퍼바이저보다 가볍고 빠름  -> 하이퍼바이저와 다르게 HostOS의 커널을 공유해서 사용함  프로그램 : 실행 가능한 소프트웨어, 디스크 공간 차지  이미지 : 실행 가능한 소프트웨어 + 실행에 필요한 환경, 디스크 공간 차지,  컨테이너를 실행 하기 위해서는 해당하는 이미지가 필요함  이미지 레지스트리 : 이미지를 저장하는 공간, 프라이빗과 퍼블릭이 있음  이미지명 규칙 : 레지스트리주소/프로젝트명/이미지명:이미지태그(버전)  메타데이터 : 데이터에 대한 정보  Env -> 실행 시 참조할 설정 정보  Cmd -> 실행 시 프로세스 실행 명령어  도커 명령어를 이용한 컨테이너 생성 및 실습  docker image ls (이미지명) : 로컬 이미지 조회  docker run -d --name {컨테이너명} 이미지명 : 컨테이너 실행  -d(백그라운드 실행) -> 빼면 터미널에 출력됨  docker ps(프로세스) : 실행 중인 컨테이너 조회, -a(all) 붙이면 종료된 컨테이너 포함 조회  docker rm -f : 실행 중인 컨테이너 삭제  -f를 넣어야 실행 중인 컨테이너 삭제 가능 빼면 종료한 컨테이너만 삭제 가능  docker image inspect 이미지명 : 이미지 세부 정보 조회  docker container inspect 컨테이너명 : 컨테이너 세부 정보 조회  docker run 이미지명 (실행명령) : 컨테이너 실행 시 메타데이터 cmd 덮어쓰기  docker run --env KEY=VALUE 이미지명 : 컨테이너 실행 시 메타데이터 env 덮어쓰기  docker run --name 컨테이너명 이미지명 cat 경로 : 경로를 메타데이터에 덮어씌움 실행 하면 경로에 있는 파일만 보여주고 종료함  docker run --name 컨테이너명 --env KEY=VALUE 애플리케이션 또는 컨테이너 : 메타데이터 env 추가 후 실행  docker pull 이미지명 : 로컬 스트리지 이미지 다운로드  docker tag 기존이미지명 추가할이미지명 : 로컬스토리지 이미지명 추가  docker push 이미지명 : 이미지 레지스트리에 이미지 업로드  *(개별 활동들을 요약하세요.)* | |
| 개발계획 대비 진행현황 | …. 현재 진행현황: 중간고사기간 이전까지 쿠버네티스 및 도커 학습을 끝내는 것 현재 진행중  *(개발계획을 1~2주 단위로 분할하고, 개발 아이템별로 세분한 후 초과달성, 달성, 지연을 표현하세요. 테이블, PPT 도표, 엑셀 등 사용하는 툴은 자유롭게 선택하세요.)* | |
| 다음주  계획 | …다음주 계획으로는 우선 쿠버네티스와 도커 학습을 마무리하고 학습단계에서의 최종단계인 쿠버네티스 배포가 계획입니다  *(개발 아이템별로 세분한 후 아이템별로 다음주 계획을 작성하세요.)* | |
| 주요 결과물 | ….x  *(software version, 공모전, 논문, 특허 등 정리)* | |