머신러닝 파이프라인

도커 소개

송호연

목차

도커 소개

학습목표

도커 소개

01. 도커 개요에 대해 이해한다.

    도커 기술이 왜 중요한 지, 어떤 기능을 갖고 있는지 이해한다.

02. 도커 설치법을 이해한다.

    도커 설치 방법을 배운다.

03. 도커 기본 사용법에 대해 이해한다.

    도커 기본 사용법을 따라해보면서 익혀봅니다.

도커 개요

도커 개요

도커가 필요한 이유

Docker는 애플리케이션을 개발, 제공 및 실행하기위한 개방형 플랫폼입니다.

Docker를 사용하면 애플리케이션을 인프라에서 분리 할 수 있으므로 소프트웨어를 빠르게 제공 할 수 있습니다. Docker를 사용하면 애플리케이션을 관리하는 것과 동일한 방식으로 인프라를 관리 할 수 있습니다. 코드를 신속하게 전달, 테스트 및 배포하기위한 Docker의 방법론을 활용하면 코드 작성과 프로덕션 실행 사이의 지연을 크게 줄일 수 있습니다.

*출처 : 출처 작성

도커 개요

도커 플랫폼

Docker는 컨테이너라고하는 느슨하게 격리 된 환경에서 애플리케이션을 패키징하고 실행할 수있는 기능을 제공합니다. 격리 및 보안을 통해 주어진 호스트에서 여러 컨테이너를 동시에 실행할 수 있습니다.

컨테이너는 가볍고 애플리케이션을 실행하는 데 필요한 모든 것이 포함되어 있으므로 현재 호스트에 설치된 항목에 의존 할 필요가 없습니다. 작업하는 동안 컨테이너를 쉽게 공유 할 수 있으며 공유하는 모든 사람이 동일한 방식으로 작동하는 동일한 컨테이너를 갖게됩니다.

*출처 : 출처 작성

도커 개요

도커 플랫폼

Docker는 컨테이너의 수명주기를 관리하기위한 도구와 플랫폼을 제공합니다.

- 컨테이너를 사용하여 애플리케이션 및 지원 구성 요소를 개발합니다.
- 컨테이너는 애플리케이션을 배포하고 테스트하는 단위가됩니다.
- 준비가되면 애플리케이션을 컨테이너 또는 오케스트레이션 된 서비스로 프로덕션 환경에 배포합니다. 이는 프로덕션 환경이 로컬 데이터 센터, 클라우드 공급자 또는 둘의 하이브리드이든 상관없이 동일하게 작동합니다.

*출처 : 출처 작성

도커 개요

Docker는 어디에 사용할 수 있습니까?

빠르고 일관된 애플리케이션 제공

Docker는 개발자가 애플리케이션과 서비스를 제공하는 로컬 컨테이너를 사용하여 표준화 된 환경에서 작업 할 수 있도록함으로써 개발 수명주기를 간소화합니다. 컨테이너는 지속적 통합 및 지속적 배포 (CI / CD) 워크 플로에 적합합니다.

예제 시나리오

- 개발자는 로컬에서 코드를 작성하고 Docker 컨테이너를 사용하여 동료와 작업을 공유합니다.
- Docker를 사용하여 애플리케이션을 테스트 환경으로 푸시하고 자동 및 수동 테스트를 실행합니다.
- 개발자가 버그를 발견하면 개발 환경에서 버그를 수정하고 테스트 및 검증을 위해 테스트 환경에 재배포 할 수 있습니다.
- 테스트가 완료되면 업데이트 된 이미지를 프로덕션 환경에 푸시하는 것만 큼 간단하게 고객에게 수정 사항을 제공 할 수 있습니다.

*출처 : 출처 작성

도커 개요

Docker는 어디에 사용할 수 있습니까?

반응형 배포 및 확장

Docker의 컨테이너 기반 플랫폼은 이동성이 뛰어난 워크로드를 허용합니다. Docker 컨테이너는 개발자의 로컬 랩톱, 데이터 센터의 물리적 또는 가상 머신, 클라우드 공급자 또는 혼합 환경에서 실행할 수 있습니다.

Docker의 이식성과 경량 특성은 또한 비즈니스 요구에 따라 거의 실시간으로 워크로드를 동적으로 관리하고 애플리케이션 및 서비스를 확장하거나 해체 할 수 있도록합니다.

*출처 : 출처 작성

도커 개요

Docker는 어디에 사용할 수 있습니까?

동일한 하드웨어에서 더 많은 워크로드 실행

Docker는 가볍고 빠릅니다. 하이퍼 바이저 기반 가상 머신에 대한 실행 가능하고 비용 효율적인 대안을 제공하므로 더 많은 컴퓨팅 용량을 사용하여 비즈니스 목표를 달성 할 수 있습니다. Docker는 더 적은 리소스로 더 많은 작업을 수행해야하는 고밀도 환경과 중소 규모 배포에 적합합니다.

*출처 : 출처 작성

도커 개요

Docker 아키텍쳐

Docker 아키텍처

Docker는 클라이언트-서버 아키텍처를 사용합니다.

Docker 클라이언트 는 Docker 컨테이너를 빌드, 실행 및 배포하는 무거운 작업을 수행 하는 Docker 데몬 과 통신합니다. Docker 클라이언트와 데몬 은 동일한 시스템에서 실행되거나 Docker 클라이언트를 원격 Docker 데몬에 연결할 수 있습니다.

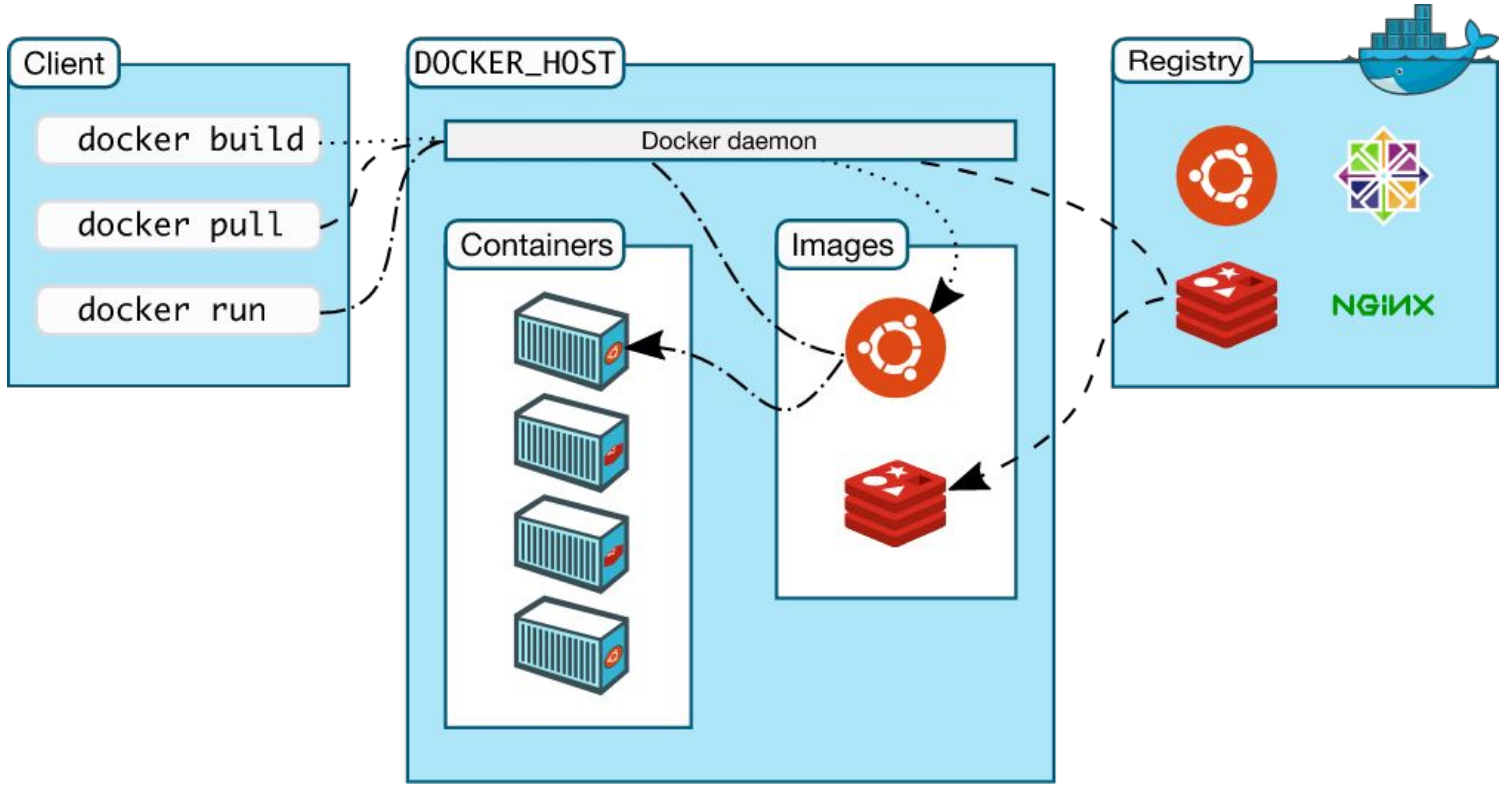
Docker 클라이언트와 데몬은 UNIX 소켓 또는 네트워크 인터페이스를 통해 REST API를 사용하여 통신합니다. 또 다른 Docker 클라이언트는 Docker Compose로, 컨테이너 세트로 구성된 애플리케이션으로 작업 할 수 있습니다.

*출처 : 출처 작성

도커 개요

Docker 아키텍쳐

Docker 아키텍처



*출처 : 출처 작성

도커 개요

Docker 데몬

Docker 데몬

Docker 데몬 (dockerd)은 Docker API 요청을 수신하고 이미지, 컨테이너, 네트워크 및 볼륨과 같은 Docker 객체를 관리합니다. 데몬은 다른 데몬과 통신하여 Docker 서비스를 관리 할 수도 있습니다.

*출처 : 출처 작성

도커 개요

Docker 클라이언트

Docker 클라이언트

Docker 클라이언트 (docker)는 많은 Docker 사용자가 Docker와 상호 작용하는 기본 방법입니다.

docker run와 같은 명령을 사용하면 클라이언트가이 명령을 dockerd에 전송하여 실행합니다.

이 docker명령은 Docker API를 사용합니다.

Docker 클라이언트는 둘 이상의 데몬과 통신 할 수 있습니다.

*출처 : 출처 작성

도커 개요

Docker 레지스트리

Docker 레지스트리

Docker 레지스트리 는 Docker 이미지를 저장합니다. Docker Hub는 누구나 사용할 수있는 공용 레지스트리이며 Docker는 기본적으로 Docker Hub에서 이미지를 찾도록 구성됩니다. 자신의 개인 레지스트리를 실행할 수도 있습니다.

docker pull또는 docker run명령 을 사용하면 구성된 레지스트리에서 필수 이미지를 가져옵니다. docker push명령 을 사용하면 이미지가 구성된 레지스트리로 푸시됩니다.

*출처 : 출처 작성

도커 개요

Docker 객체

Docker 객체

Docker를 사용하면 이미지, 컨테이너, 네트워크, 볼륨, 플러그인 및 기타 객체를 생성하고 사용하게됩니다. 이 섹션은 이러한 개체 중 일부에 대한 간략한 개요입니다.

*출처 : 출처 작성

도커 개요

Docker 이미지Image

이미지Image

이미지 도커 컨테이너를 만들기위한 지침 읽기 전용 템플릿입니다. 종종 이미지는 몇 가지 추가 사용자 정의와 함께 다른 이미지를 기반으로 합니다. 예를 들어 이미지를 기반으로하는 이미지를 빌드 할 수 ubuntu 있지만 Apache 웹 서버와 애플리케이션은 물론 애플리케이션을 실행하는 데 필요한 구성 세부 사항도 설치합니다.

자신의 이미지를 만들거나 다른 사람이 만들고 레지스트리에 게시 한 이미지 만 사용할 수 있습니다. 고유 한 이미지를 빌드하려면 이미지 를 만들고 실행하는 데 필요한 단계를 정의하는 간단한 구문 으로 Dockerfile 을 만듭니다. Dockerfile의 각 명령어는 이미지에 레이어를 만듭니다. Dockerfile을 변경하고 이미지를 다시 빌드하면 변경된 레이어 만 다시 빌드됩니다. 이것은 다른 가상화 기술과 비교할 때 이미지를 매우 가볍고, 작고, 빠르게 만드는 요소의 일부입니다.

*출처 : 출처 작성

도커 개요

Docker 객체 - 컨테이너Container

컨테이너Container

컨테이너는 이미지의 실행 가능한 인스턴스입니다. Docker API 또는 CLI를 사용하여 컨테이너를 생성, 시작, 중지, 이동 또는 삭제할 수 있습니다. 컨테이너를 하나 이상의 네트워크에 연결하거나 스토리지를 연결하거나 현재 상태를 기반으로 새 이미지를 만들 수도 있습니다.

기본적으로 컨테이너는 다른 컨테이너 및 호스트 시스템과 비교적 잘 격리되어 있습니다. 컨테이너의 네트워크, 스토리지 또는 기타 기본 하위 시스템이 다른 컨테이너 또는 호스트 시스템에서 분리되는 방식을 제어 할 수 있습니다.

컨테이너는 이미지와 사용자가 생성하거나 시작할 때 제공하는 구성 옵션에 의해 정의됩니다. 컨테이너가 제거되면 영구 저장소에 저장되지 않은 상태 변경 사항이 사라집니다.

*출처 : 출처 작성

도커 개요

Docker 명령 예시

명령 예시 - docker run

다음 명령은 ubuntu컨테이너를 실행하고 로컬 명령 줄 세션에 대화 형으로 연결 한 다음 /bin/bash.

```
$ docker run -i -t ubuntu /bin/bash
```

*출처 : 출처 작성

도커 개요

Docker 명령 예시

```
$ docker run -i -t ubuntu /bin/bash
```

이 명령을 실행하면 다음이 발생합니다 (기본 레지스트리 구성을 사용하고 있다고 가정).

1. ubuntu로컬에 이미지 가없는 경우 Docker는 docker pull ubuntu수동으로 실행 한 것처럼 구성된 레지스트리에서 이미지를 가져옵니다 .

2. Docker는 docker container create 명령을 수동으로 실행 한 것처럼 새 컨테이너를 만듭니다 .

3. Docker는 읽기-쓰기 파일 시스템을 최종 레이어로 컨테이너에 할당합니다. 이를 통해 실행중인 컨테이너는 로컬 파일 시스템에서 파일과 디렉토리를 만들거나 수정할 수 있습니다.

4. Docker는 네트워킹 옵션을 지정하지 않았으므로 컨테이너를 기본 네트워크에 연결하는 네트워크 인터페이스를 만듭니다. 여기에는 컨테이너에 IP 주소 할당이 포함됩니다. 기본적으로 컨테이너는 호스트 머신의 네트워크 연결을 사용하여 외부 네트워크에 연결할 수 있습니다.

5. Docker는 컨테이너를 시작하고 /bin/bash. 컨테이너가 대화 형으로 실행되고 터미널에 연결되어 있기 때문에 (-i및 -t 플래그 로 인해) 출력이 터미널에 기록되는 동안 키보드를 사용하여 입력을 제공 할 수 있습니다.

6. 명령 exit을 종료하기 위해 입력 /bin/bash하면 컨테이너가 중지되지만 제거되지는 않습니다. 다시 시작하거나 제거 할 수 있습니다.

*출처 : 출처 작성

도커 설치

도커 설치

도커 설치

아래에서 선호하는 운영 체제를 선택하여 Docker를 다운로드하십시오.

Mac 용 Docker Desktop 다운로드
https://desktop.docker.com/mac/stable/Docker.dmg

Windows 용 Docker Desktop 다운로드
https://desktop.docker.com/win/stable/Docker%20Desktop%20Installer.exe

Linux에 Docker Engine 설치
https://docs.docker.com/engine/install/

*출처 : 출처 작성

도커 실습

도커 개요

도커 설치

bash 창을 열고 다음 명령을 실행하십시오.
몇 가지 플래그가 사용되는 것을 알 수 있습니다. 이에 대한 추가 정보는 다음과 같습니다.

-d -분리 모드에서 컨테이너 실행 (백그라운드에서)
-p 80:80 -호스트의 포트 80을 컨테이너의 포트 80에 매핑
docker/getting-started -사용할 이미지

```
docker run -d -p 80:80 docker/getting-started
```

*출처 : 출처 작성

도커 실습

Dockerfile

Dockerfile는 사용자가 이미지를 조합하기 위해 명령 줄에서 호출 할 수있는 모든 명령이 포함된 텍스트 문서입니다.

```
$ docker build -f /path/to/a/Dockerfile .
```

*출처 : 출처 작성

도커 실습

Dockerfile

빌드가 성공하면 새 이미지를 저장할 저장소와 태그를 지정할 수 있습니다.

```
$ docker build -t shykes/myapp .
```

*출처 : 출처 작성

도커 실습

Dockerfile

빌드 후 이미지를 여러 저장소에 태그하려면 -t다음 build명령 을 실행할 때 여러 매개 변수를 추가하십시오 .

```
$ docker build -t chrisai/myapp:1.0.2 -t chrisai/myapp:latest .
```

*출처 : 출처 작성

도커 실습

Dockerfile

Docker 데몬은에서 지침을 실행하기 전에의 Dockerfile예비 유효성 검사를 수행
Dockerfile하고 구문이 잘못된 경우 오류를 반환합니다.

```
$ docker build -t test/myapp .

Sending build context to Docker daemon 2.048 kB
Error response from daemon: Unknown instruction: RUNCMD
```

*출처 : 출처 작성

도커 실습

Dockerfile

Docker 데몬은 Dockerfile최종적으로 새 이미지의 ID를 출력하기 전에 필요한 경우 각 명령의 결과를 새 이미지에 커밋하면서 하나씩 지침을 실행합니다. Docker 데몬은 사용자가 보낸 컨텍스트를 자동으로 정리합니다.

각 명령어는 독립적으로 실행되며 새 이미지가 생성되므로 RUN cd /tmp다음 명령어에 영향을 미치지 않습니다.

가능할 때마다 Docker는 중간 이미지 (캐시)를 재사용하여 docker build프로세스를 크게 가속화합니다 . 이는 Using cache콘솔 출력 의 메시지로 표시됩니다 . (자세한 내용은 Dockerfile모범 사례 가이드를 참조하십시오 .

*출처 : 출처 작성

도커 실습

Dockerfile

```
$ docker build -t chrisai/ambassador .

Sending build context to Docker daemon 15.36 kB
Step 1/4 : FROM alpine:3.2
 ---> 31f630c65071
Step 2/4 : MAINTAINER SvenDowideit@home.org.au
 ---> Using cache
 ---> 2a1c91448f5f
Step 3/4 : RUN apk update &&      apk add socat &&        rm -r /var/cache/
 ---> Using cache
 ---> 21ed6e7fbb73
Step 4/4 : CMD env | grep _TCP= | (sed 's/.*_PORT_\([0-9]*\)_TCP=tcp:\/\/\(.*\):\(.*\)/socat -t
100000000 TCP4-LISTEN:\1,fork,reuseaddr TCP4:\2:\3 \&/' && echo wait) | sh
 ---> Using cache
 ---> 7ea8aef582cc
Successfully built 7ea8aef582cc
```

*출

도커 실습

Dockerfile

명령어는 대소 문자를 구분하지 않습니다. 그러나 규칙은 인수와 더 쉽게 구별하기 위해 대문자로 표시하는 것입니다.

Docker는 Dockerfile순서대로 지침을 실행합니다 . A Dockerfile 는 FROM 명령으로 시작해야합니다 . 이는 파서 지시문 , 주석 및 전역 범위 ARG 뒤에있을 수 있습니다 . FROM 명령은 지정 부모의 이미지 당신이 구축하고있는합니다 . FROM의 행에서 ARG사용되는 인수를 선언 하는 하나 이상의 명령어 만 앞에 올 수 있습니다

```
RUN echo 'we are running some # of cool things'
```

*출처 : 출처 작성

도커 실습

Dockerfile

Dockerfile 명령어가 실행되기 전에 주석 줄이 제거됩니다. 즉, 다음 예제의 주석은 echo명령을 실행하는 셸에서 처리되지 않으며 아래 두 예제 모두 동일합니다.

```
RUN echo hello \

world
```

*출처 : 출처 작성

도커 실습

CMD 명령

CMD명령은 세 가지 형태가 있습니다 :

CMD ["executable","param1","param2"](exec 형식, 선호하는 형식)
CMD ["param1","param2"](ENTRYPOINT의 기본 매개 변수로)
CMD command param1 param2(쉘 형태)

Dockerfile에는 CMD명령이 하나만 있을 수 있습니다.
둘 이상의 CMD를 나열하면 마지막 CMD 항목만 적용됩니다.

```
FROM ubuntu
CMD echo "This is a test." | wc -
```

*출처 : 출처 작성

도커 실습

LABEL 명령

이 LABEL지침은 이미지에 메타 데이터를 추가합니다. A LABEL는 키-값 쌍입니다. LABEL값 내에 공백을 포함하려면 명령 줄 구문 분석에서와 같이 따옴표와 백 슬래시를 사용합니다.

```
LABEL <key>=<value> <key>=<value> <key>=<value> ...
```

```
LABEL "com.example.vendor"="ACME Incorporated"
LABEL com.example.label-with-value="foo"
LABEL version="1.0"
LABEL description="This text illustrates \
that label-values can span multiple lines."
```

*출처 : 출처 작성

도커 실습

LABEL 명령

이미지의 레이블을 보려면 docker image inspect명령을 사용하십시오 . 이 --format옵션을 사용 하여 레이블 만 표시 할 수 있습니다 .

```
$ docker image inspect --format='' myimage
```

```
{
  "com.example.vendor": "ACME Incorporated",
  "com.example.label-with-value": "foo",
  "version": "1.0",
  "description": "This text illustrates that label-values can span multiple lines.",
  "multi.label1": "value1",
  "multi.label2": "value2",
  "other": "value3"
}
```

*출처 : 출처 작성

도커 실습

EXPOSE 명령

이 EXPOSE명령은 컨테이너가 런타임에 지정된 네트워크 포트에서 수신 대기한다는 것을 Docker에 알립니다. 포트가 TCP 또는 UDP에서 수신하는지 여부를 지정할 수 있으며 프로토콜이 지정되지 않은 경우 기본값은 TCP입니다.

```
EXPOSE <port> [<port>/<protocol>...]
```

```
EXPOSE 80/tcp
EXPOSE 80/udp
```

*출처 : 출처 작성

도커 실습

EXPOSE 명령

EXPOSE설정에 관계없이 -p플래그 를 사용하여 런타임에이를 재정의 할 수 있습니다 . 예를 들면

```
docker run -p 80:80/tcp -p 80:80/udp ...
```

*출처 : 출처 작성

도커 실습

ENV 명령

ENV지시는 환경 변수 설정 <key>값을 <value>. 이 값은 빌드 단계의 모든 후속 지침에 대한 환경에 있으며 많은 곳에서도 인라인 으로 바꿀 수 있습니다 . 값은 다른 환경 변수에 대해 해석되므로 이스케이프되지 않으면 따옴표 문자가 제거됩니다. 명령 줄 구문 분석과 마찬가지로 따옴표와 백 슬래시를 사용하여 값 내에 공백을 포함 할 수 있습니다.

```
ENV MY_NAME="John Doe"
ENV MY_DOG=Rex\ The\ Dog
ENV MY_CAT=fluffy
```

*출처 : 출처 작성

도커 실습

ENV 명령

를 사용하여 설정된 환경 변수 ENV는 결과 이미지에서 컨테이너가 실행될 때 유지됩니다. 를 사용하여 값을 docker inspect보고를 사용하여 변경할 수 있습니다.

```
docker run --env <key>=<value>
```

*출처 : 출처 작성

도커 실습

ENV 명령

를 사용하여 설정된 환경 변수 ENV는 결과 이미지에서 컨테이너가 실행될 때 유지됩니다. 를 사용하여 값을 docker inspect보고를 사용하여 변경할 수 있습니다.

```
RUN DEBIAN_FRONTEND=noninteractive apt-get update && apt-get install -y ...
```

*출처 : 출처 작성

짚어보기

도커 소개

01. 도커 개요에 대해 이해한다.

    도커 기술이 왜 중요한 지, 어떤 기능을 갖고 있는지 이해한다.

02. 도커 설치법을 이해한다.

    도커 설치 방법을 배운다.

03. 도커 기본 사용법에 대해 이해한다.

    도커 기본 사용법을 따라해보면서 익혀봅니다.

머신러닝 파이프라인

도커 소개

송호연

감사합니다.

THANKS FOR WATCHING