01-26 도커공부

13장 Docker 컴포즈

Docker 컴포즈란?

개발 머신 상태를 모른다.

docker network ls —>지우고 초기 상태로 돌아간다.

docker network create fleetman-network

docker container run -d —network fleetman-network —name database -e MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=fleetman mysql

name database —> datasource의 url database와 일치해야함.

docker container logs -f database

docker container run -d —network fleetman-network —name fleetman-webapp -p 80:8080 virtualpairprogrammers/fleetman-production

Docker compose 로 아주 간단한 텍스트 파일을 작성할 수 있고 이 텍스트 파일에는 명령줄에 입력한 모든 정보가 포함된다.

모든 컨테이너의 구성데이터를 넣을 수 있다.

```
docker-compose -v

docker-compose.yaml
version: "3"

services:

   fleetman-webapp:
      image: virtualpairprogrammers/fleetman-production
      networks:
         - fleetman-network
```

```
      ports:
         - 80:8080
      depends_on:
         - database

   database:
      image: mysql:5
      networks:
         - fleetman-network
      environment:
         - MYSQL_ROOT_PASSWORD=password
         - MYSQL_DATABASE=fleetman

networks:
   fleetman-network:
```

컴포즈 파일

Docker Compose로 전체 실행 아키텍처 구성을 정의하는 텍스트 파일을 작성할 수 있다.

Docker Compose의 첫 번째 개념은 서비스를 실행할 컨테이너를 호출하는 겁니다.

컨테이너는 시스템 내의 서비스입니다.

리스트 요소 앞에는 - 로 표시한다.

Docker Compose에서 또 다른 목표는 이 파일을 콜드 상태로 실행할 수 있어야 한다는 겁니다.

즉, 로컬 환경에서 아무것도 구성하지 않았더라도 이 파일은 실행될 수 있습니다. —>networks

시작순서

어떤 방법으로든, 이 컨테이너가 실행되고 데이터베이스가 완전히 작동하기 전에는 컨테이너를 시작하지 말라고 Docker Compose에 명령할 수 있다면 좋을 겁니다.

depends_on 옵션으로 서비스가 시작할 순서를 제어하고, Compose는

종속성 순서에 따라 컨테이너를 시작할 겁니다. 문서를 더 살펴보기 전에

작동 방식을 보여드릴게요.

시판된 Kubernetes와 같은 다른 툴이 있다는 걸 아시죠.

현재까지 가장 규모가 큰 툴이고요.

Kubernetes를 살펴보면, Kubernetes는 컨테이너가 정상인지 확인하는 작업을

훌륭하게 수행합니다.

그러니 여기 보이는 문제는 Kubernetes 등을 사용할 경우에 문제가 되지 않습니다.

Docker Compose는 프로덕션 표준이 아니라고 생각하셨으면 좋겠네요.

docker-compose 실행

앞서 언급한 것처럼, 웹 앱이 데이터베이스와 연결하려고 합니다. 데이터베이스가 완전히 가동되기 전에 말이죠.

이 경우, 컨테이너를 중지할 Ctrl C를 수행하여

docker-compose up을 다시 실행합니다. 큰 차이는 새 컨테이너를 처음부터 시작하지 않아도 된다는 겁니다.

중지된 컨테이너를 다시 시작할 뿐이죠.

즉, 데이터베이스가 훨씬 실행된다는 겁니다. 사실 1초도 걸리지 않았어요.

```
docker-compose up 기본 설정이 docker-compose.yaml

docker-compose up -d --> 백그라운드 실행

docker-compose logs -f fleetman-webapp(서비스 이름)

docker-compose down (모든 컨테이너 중지+ 삭제)
```

배포 변경

컨테이너 중 하나를 바꾸고, 다른 컨테이너는 다시 시작하지 않으면서 해당 컨테이너만 다시 시작하는 상황.

사소한 수정이 있었을 경우 다시 clean package로 메이븐 빌드를 해도 이전 버전으로 나타나고 변화가 적용되지 않는다. 이때는

docker-compose up -d 를 하면, 업데이트가 반영 된것을 볼 수 있다. 해당 컨테이너만 stop 명령어로 중지하고 다시 시작할 필요가 없다.

프로덕션 시 이용하도록 되어 있지 않아요.

그러면 프로덕션에서 애플리케이션을 실행할 방법은 무엇일까요?

다음 몇 개의 챕터에서 다룰 내용입니다.

Docker Swarm 및 Docker 스택에 대한 흥미로운 내용이죠.

다음 챕터에서 뵙겠습니다.

14장 Swarm

오케스트레이션 시스템

클러스터에서 컨테이너를 실행할 수 있게 된다. SWARM으로 중단 시간 없이 다시 배포하는 작업.

Swarm은 오케스트레이션 시스템 중 하나, 즉 프로덕션에서 컨테이너를 관리할 수 있다는 뜻.

쿠버네티스는 구글에서 개발된 오케스트레이션 시스템.

쿠버네티스와 스웜은 경쟁 관계. 대부분은 쿠버네티스를 사용한다.

Kubernetes보다 훨씬 간단해요.

즉, 오케스트레이션 시스템 사용 방법과 컨테이너를

클라우드의 클러스터에 넣을 방법을 배우기 좋습니다.

시작하기 좋습니다.

Swarm 소개

하지만 프로덕션에서 시스템을 롤아웃할 때 확장 문제를 겪을 수도 있어요.

하지만 컨테이너를 수십 개, 수백 개로 확장한 다음에는 이를 처리할 수 있는

서버를 찾는 게 어려울 수 있습니다.

많은 컨테이너를 실행하려면 매우 비싼 서버가 필요할 겁니다.

그래서 수평적으로 확장하는 게 흔합니다. 다수의 물리적 컴퓨터에 컨테이너를

분산하는 거죠.

그리고 Docker Swarm이 바로 이러한 작업을 수행할 수 있는 도구입니다.

그리고 Docker Swarm이 바로 이러한 작업을 수행할 수 있는 도구입니다.

사실, Swarm으로 더 많은 걸 할 수 있어요.

컨테이너 오케스트레이션 시스템이라는 완전한 시스템이죠.

Docker Swarm의 기본 개념은 단일 호스트 컴퓨터에서 실행하는 게 아니라,

다수의 노드에 걸쳐 분산한다는 겁니다.

그런 다음 Swarm에 컨테이너를 추가하도록 요청하면 이번에도 Swarm이
컨테이너를 실행할 노드를 결정합니다.
대략적으로 일종의 밸런싱 시스템으로, 다른 노드보다 컨테이너가 많은
노드가 없게 됩니다.
이후 챕터에서는 Amazon EC2 인스턴스 몇 가지를 구동해볼 겁니다. 4개, 5개 정도로
마이크로서비스 아키텍처를 배포하는 작업을 실험할 겁니다.
Play with Docker

Swarm 서비스

```
docker container ls -a 로 아무것도 없는 상태에서 시작.

인스턴스 하나로만 Docker Swarm 시작해서 연습 (실제 프로덕션에선 이렇지 않음)

docker swarm --> 명령어 보기.

docker swarm init --> manager가 된다.

docker service create -d --network fleetman-network
 -e MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=fleetman --name database mysql:5

docker network create --driver overlay fleetman-network
docker network rm fleetman-network
docker network create --driver overlay fleetman-network

docker service create -d --network fleetman-network
 -e MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=fleetman --name database mysql:5

docker service ls
```

지금부터는 서비스를 Swarm 내부의 컨테이너로 생각하세요.
전부터 사용해왔던 fleetman-network에 문제가 생겼다고 합니다.
정확히, 서비스로 사용할 수 없으며 Swarm에 적용되는 네트워크만
사용할 수 있다고 하죠. 꽤 모호합니다.
Swarm을 사용할 때는 특별한 네트워크 유형이 필요하다는 게 이유입니다.
물론, 큰 차이점은 컨테이너가 다른 물리적 노드에 있을 가능성이 높기 때문이죠.
그래서 같은 호스트 컴퓨터에서 노드끼리 통신하지 않습니다.
서로 다른 물리적 컴퓨터 사이에서 통신하겠죠.
그래서 Swarm에서는 다른 유형의 네트워크를 이용합니다.

오버레이 네트워크라고 합니다.

오버레이 네트워크는 아주 만들기 쉽습니다.

docker network create --driver에 이어 overlay를 입력합니다.

이전에는 16진수였는데 이제 보니까 연속되는 영숫자가

나타납니다. 서비스의 고유한 ID입니다.

docker service ls로 실행 중인 서비스를 확인할 수 있습니다.

이 모든 작업은 백그라운드에서 수행됩니다.

Swarm 모드를 실행하고 있지만, docker container ls 커맨드로 돌아갈 수 있죠.

이후 강의에서 Amazon EC2 인스턴스를 사용할 예정입니다.

하지만 EC2로 실행해보고 싶지 않다면, 이를 경험해볼 아주 매력적인 선택지가

있어요.

Play with Docker라는 겁니다.

▼ adblocker

광고 차단기가 설치되어 있으면 labs.play-with-docker.com/이 작동하지 않는다는 것을 알게 되었습니다. "차단" 버튼이 응답하지 않습니다. 나는 그들이 나쁜 짓을 하고 있다고 생각하지 않고, 단지 사이트의 버그일 뿐이다.

play-with-docker.com을 화이트리스트에 추가하여 광고 차단기를 비활성화하면 됩니다. 걱정하지 마십시오. 사이트에 광고가 없고 안전한 사이트입니다!

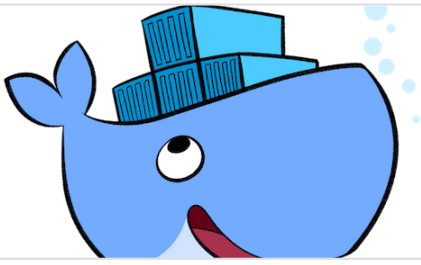
또 다른 메모 - 제가 비디오를 녹화한 이후 로그인 프로세스가 약간 변경되었습니다. 이제 DockerHub 계정이나 GitHub 계정으로 로그온하는 것이 아니라, 여러분 모두가 적어도 하나는 가지고 있기를 바랍니다!

Play with Docker

Play with Docker

Play with Docker (PWD) is a project hacked by Marcos Liljedhal and Jonathan Leibiusky and sponsored by Docker Inc. PWD is a Docker playground which allows users to run Docker commands

https://labs.play-with-docker.com/



물리적 컴퓨터의 여러 인스턴스를 구동할 수 있는 클라우드 기반 시스템이죠.

실제로는 물리적 인스턴스가 아닙니다.

물론 다 가상 머신이죠. 전혀 문제가 되진 않아요.

무료로 마음껏 프로비저닝할 수 있는 독립 실행형 컴퓨터처럼 느껴질 거예요.

그래도 Docker를 경험해볼 수 있는 훌륭한 방법입니다. 비용이 발생하지 않으며

EC2나 Microsoft Azure 등 클라우드 서비스의 복잡성이 없죠.

```
ADD NEW INSTANCE 클릭
ls pwd whoami docker--version

docker swarm init
docker swarm init --advertise-addr (IP 주소)

docker swarm join --token ~~~~~~~~ 복사
다른 노드에서 붙여넣기.

아주 간단하게 Swarm에 포함 시킬 수 있다.

다시 매니저 노드로 돌아와서
docker node ls 로 확인

docker network create --driver overlay fleetman-network

docker network ls

노드2 에서는 네트워크 목록 보이지 않음(워커이기 때문)
노드1에서
docker service create -d --network fleetman-network
 -e MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=fleetman --name database mysql:5

docker service ls
```

이제 진짜 재밌는 걸 시작할 수 있어요.

이 두 노드를 모두 Docker Swarm에 포함시키겠습니다.

워커의 요점입니다.
매니저 노드에서 사용하는 커맨드 대부분은 여기서 작동하지 않아요.

노드에서 작업하며 기본적으로 커맨드를 실행하는 데 쓸 수 없습니다. 워커에서 실행 중인

컨테이너가 있더라도 말이죠.

워커 노드에서 Swarm 커맨드를 실행할 수는 없지만 일반적인 Docker 커맨드는 모두 실행할 수 있어요.

docker container ls

이 노드에 있는 모든 컨테이너 목록을 반환합니다. 여러분의 데이터베이스 컨테이너는

어쩌면 이 노드에서 실행 중일 수 있겠죠. 약간 무작위입니다.

서비스 로그 모니터링

노드 1의 매니저 노드로 돌아가서 두 번째 서비스를 실행합니다

```
docker service create -d --network fleetman-network -p 80:8080 --name fleetman-webapp
virtualpairprogrammers/fleetman-production

docker service ls
매니저 노드로 돌아와서
docker service logs -f database
docker service logs -f fleetman-webapp
```

하지만 전처럼 docker container ls를 사용해 보면, 실행될 노드는

알 수 없습니다.

추측하자면 다른 노드일 겁니다. 컨테이너의 균형을 맞추려 하니까요.

한 노드에 컨테이너 두 개가 있고, 다른 노드에는 컨테이너가 없는 상황은 나타나지 않아요.

이 노드에서 docker container ls를 수행하면 컨테이너 하나만 표시됩니다.

하지만 다른 노드로 바꿔서 docker container ls를 반복하면

이 노드에서 실행 중인 일반적인 Docker 컨테이너가 확인됩니다.

제가 Docker Swarm에서 강조하려는 점이에요.

복잡한 몇 가지 오케스트레이션 툴로 구성한 관리 계층이 있지만, 이면에서 수행되는

모든 작업은 일반적인 Docker 컨테이너를 백그라운드에서 시작하고 중지하는 것입니다.

이 컨테이너가 워커 노드에서 실행되고 있더라도
매니저 노드에서 로그를 볼 수 있습니다.

맨 위에 작은 링크가 있는데, 이 링크는 노드에서

노출하는 모든 포트를 표시합니다.

하지만 다음 챕터에서는 Docker 스택이라는 개념을 살펴볼게요.

Swarm 작업이 훨씬 쉬워질 겁니다.

15장 스택

Manager와 worker 비교

스패너 모양의 도구 클릭하면 템플릿을 사용해서 시간 아낄 수 있음.

leader가 이닛한 노드, Reachable은 매니저.

매니저와 워커 모두에 컨테이너가 배포됨.

매니저에서는 docker servlce ls 가능, 워커는 불가능Raft 시스템으로 매니저 노드간 통신, 매니저가 하나면 매니저가 중단될 시 Swarm을 쓸 수 없는 문제가 있음.

핵심은 Swarm에서 매니저를 홀 수개로 유지하는걸 권장. 홀 수 일 때 작동하기가 쉽다.

템플릿에서 매니저가 3개인 이유.

스택 구축

매니저 노드 중 하나에 있는 완전한 애플리케이션이 stack이다.

```
docker stack ls
docker stack deploy -c
apk add --no-cache nano
nano docker-compose.yaml
오버레이 네트워크 지정하지 않아도 된다.
드라이버를 지정하지 않으면, Docker 스택은 기본 드라이버로 가정한다.
Swarm에서 기본은 오버레이, 로컬은 브릿지이다.
같은 Docker Compose 파일을 다시 사용할 수 있다.
 ctrl +o로 파일 작성 -->ctrl x

docker stack deploy -c docker-compose.yaml fleetman-stack

docker service ls
```

서비스 생명 주기

```
docker service ls
서비스가 시작되었다는걸 의미. 컨테이너 시작 순서는?
Compose 에서 설정했던 표시는 무시된다.
컨테이너는 병렬로 시작되며 시작순서를 마음대로 제어할 수 없다.

특성 순서로 컨테이너를 시작할 방법은?
기본적으로 Swarm은 컨테이너가 중단될 경우 컨테이너를
자동으로 다시 시작한다.
```

```
Swarm의 복원력이 크다는 걸 의미.

docker service ps fleetman-stack_fleetman-webapp, 프로세스 의미.
(일하고 있는 특정 노드로 이동)
docker container --help
docker container kill (id)
(매니저로 이동)
docker service ls
docker service logs -f fleetman-stack_fleetman-webapp
docker service ps fleetman-stack_fleetman-webapp
-> 실행중인 컨테이너로 가서,
docker container logs -f (id)
```

복제 서비스

라우팅 메시

Visualizer

롤링 업데이트

16장 EC2에서의 마이크로 서비스

Fleetman 마이크로 서비스 소개

EC2클러스터 시작

클러스터 구성

스택 배포

시스템 복구력

서비스 디스커버리