# 도커에서 이미지 설치

Docker pull (이미지)

# 실행중인 docker container 확인

Docker ps

# 실행중인 것과 stop 된 container 확인

Docker ps –a (-all)

# container stop

Docker strop (name)

# stop 된 컨테이너 다시 시작

Docker start (name)

#container 이름변경

Docker rename container-name container-rename

Ex) docker rename webserver web

Docker ps

# Ubuntu, centos:7 image download

Docker pull Ubuntu

Docker pull centos:7 # centos:version(tag)

# image 상세보기 – image config dockerfile 조회

Docker image inspect (image-name)

# prune은 사용하지 않는 image들을 한꺼번에 삭제한다.

# 이미지를 특정해서 삭제하려면 rm으로 사용.

# 이미지를 사용하는 container가 있다면, 이미지가 삭제 안된다.

# Exit 해도 컨테이너가 있다면 컨테이너부터 지워야 한다.

Docker container(<-생략가능) rm (container\_name)

# image 삭제 – image를 사용하는 container가 있을 경우, 사용하는 container 부터 삭제 후 삭제.

Docker image rm (image\_name of image\_ID)

#ubuntu, centos:7 container 생성/실행(create/start)

# centos의 /bin/cal – calender 실행 –it:표준입출력 사용

docker run -it --name centos\_cal centos:7 /bin/cal

docker ps –a

docker run –it –name centos\_shell centos:7 /bin/bash

111 111 111

(7 7 7 : 모든 권한이 모두에게 개방)

111 101 101 : -rwxr-xr-x

(7 5 5 : admin 은 읽고, 쓰고, 실행 가능. 나머지는 쓰기는 불가능)

수정 하려면 chmod 755 처럼 수정 가능

# exit된 container를 다시 시작

Docker start –I (container\_name)

이 상태에서 ctrl + p + q 를 하면 container가 up되어 있는 상태에서 도커로 돌아오게 됨.

이후 다시 container로 돌아가려면

Docker attach (container\_name)

root@:/#

#docker image 검색

#docker search image명

docker search nginx

#docker image download

#docker pull image명

docker pull nginx

#docker image list

#docker images 또는 docker image ls

docker images

#conatiner 실행

#docker container run --name conatiner-name -d -p port:port image명

#-d :detach(background로 실행), -p:publish(포트 포워딩)

docker container run --name webserver -d -p 80:80 nginx

#실행중인 container 확인

docker ps

#실행중이거나 실행했던 constainer 확인

docker ps -a

#Exited된 container 삭제

#docker rm container-name

docker rm hellworld

docker ps -a

#container 이름변경

#docker rename container-name container-rename

docker reanme webserver web

docker ps -a

#container 중지

#docker stop container-name

docker stop web

#container 실행

docker start web

#ubuntu, centos:7 image download

docker pull ubuntu

docker pull centos:7 #centos:version(tag)

docker images

#image 상세보기 - image config .Dockerfile 조회

docker image inspect ubuntu

#image 삭제 - image사용하는 container가 있을 경우 사용하는 container 삭제 후 삭제

docker image rm imageID

#사용하지 않는 image삭제

docker image prune

#ubuntu, centos:7 conatiner 생성/실행(create/start)

#centos의 /bin/cal - calendar 실행 -it:표준입출력 사용

docker run -it --name centos\_cal centos:7 /bin/cal

docker ps -a

docker run -it --name centos\_shell centos:7 /bin/bash //centos:7 bash명령실행

root@:/#adduser test1 //test1 user 생성

root@:/#su test //test1 user로 변경

test@:/$ ls -al //목록보기

test@:/$ exit //root로 가기

root@:/# exit //docker 로 가기

docker ps -a //centos\_shell은 exited 상태

root@:/# //ctrl+p+q docker 로 가기

docker ps -a //centos\_shell은 Up 상태

docker attach centos\_shell //root로

root@:/#cat /etc/issue //centos 버전확인

root@:/#ctrl+p+q

#ubuntu 실행 확인

docker run -it --name ubuntu\_shell ubuntu /bin/bash

root@:/#cat /etc/issue //ubuntu 버전확인 - centos와 비교

root@:/#adduser user1 //user생성

root@:/#passwd root //root계정 암호설정

root@:/#

root@:/# ctrl+p+q

docker ps //web,ubuntu\_shell, centos\_shell 실행 중

docker ps -q //실행중인 container id만 출력

docker stop `docker ps -q` //실행중인 id 모두 stop

docker start `docker ps -a -q` //모든 container start

docker ps

for index in `docker ps -q`;do echo $index; done //실행중인 container id list

for index in `docker ps -q`;do echo $index; docker stop $index;done //실행중인 id 모두 stop

# 도커 파일로부터 도커 이미지 만들기

Docker build [option] [image이름]:[버전] [dockerfile위치]

Docker build –t (이미지 이름):(버전) (위치)

Docker build

Ex) docker build –t nodeweb:1 .

Build : 도커 이미지를 만드는 명령어

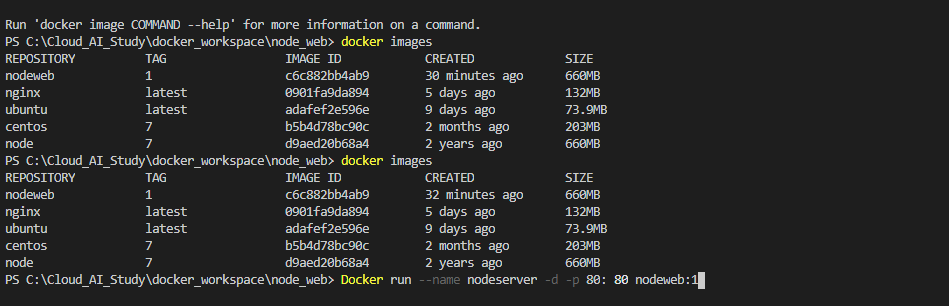
-t : 만들 이미지에 태그값(버전을 명시해준다는 의미)

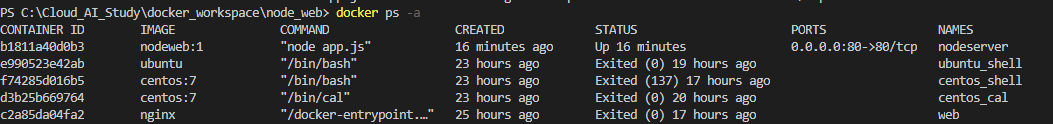
# 생성한 이미지 container 생성/실행

Docker run –name nodeserver –d –p 80: 80 nodeweb:1

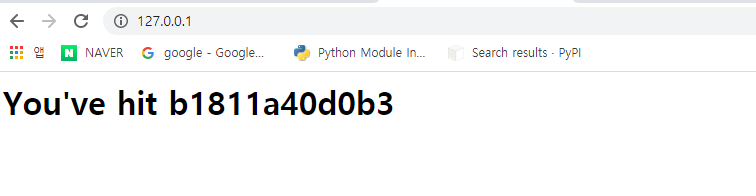
Docker ps

Curl localhost // WebBrowser http://



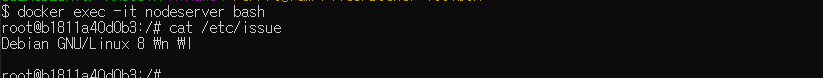


<구현된 웹페이지>

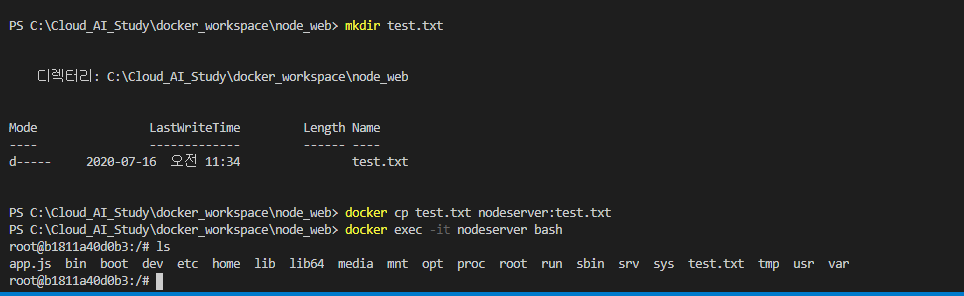


# container 확인

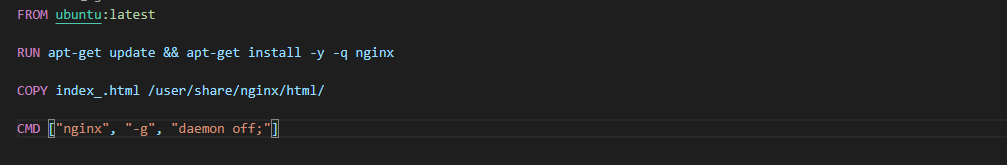
Docker exec –it nodeserver bash

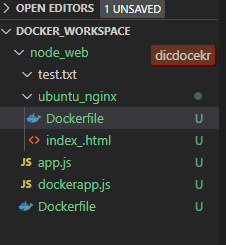


컨테이너 shell을 실행.



Cp 명령어를 사용하여 현재 내 디렉토리에 있는 파일이나 폴더를 container에 올리는 것도 가능하다.





#container 생성 및 실행

Docker run –name unginxserver –d –p 80:80 ubuntu\_nginx:1

#bash 실행

Docker exec –it unginxserver bash

@root# ls –al /usr/share/nginx/html //index.html 확인

@root# cat index.html // 작성한 html 실행.

<docker 실행 개요>

Create -> start -> exec/attach/stop -> exit/up state

조회 : ps(실행 되고있는 컨테이너) ps –al(생성된 모든 컨테이너)

빌드 : image build

#mysql

#mysql image download

Docker pull mysql:5.7

#container 생성/실행

docker run --name (some-mysql) -e MYSQL\_ROOT\_PASSWORD=(pw) -d –p 3306:3306 mysql:(tag)

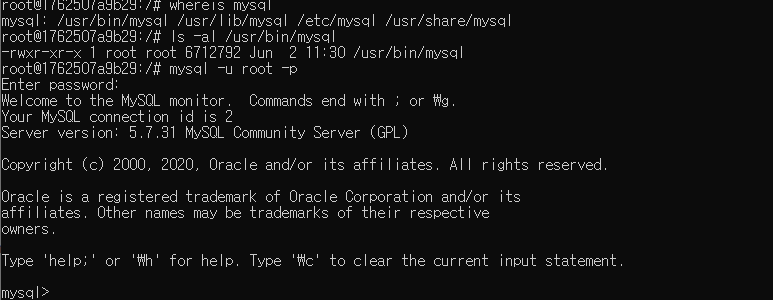
#container 명령실

Docker exec –it mysql5 bash

@root# cat /etc/issue // Debian 10 운영체제 확인

@root#mysql –u root –p

Enter password : password

Mysql> 유저마다 grant 라는 권한을 준다 (root/adminstrator)

#centos base image에 mariadb 설치하기

#cetos\_mariadb.zip download 압축풀기

~\centos\_mariadb>docker build -t ai/mariadb55:1

docker images //확인

#container 생성

docker run --name=mariadb -d -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=password ai/mariadb55:1

docker ps

docker exec -it mariadb bash

@root#mysql -u root -p

Enter Password : password

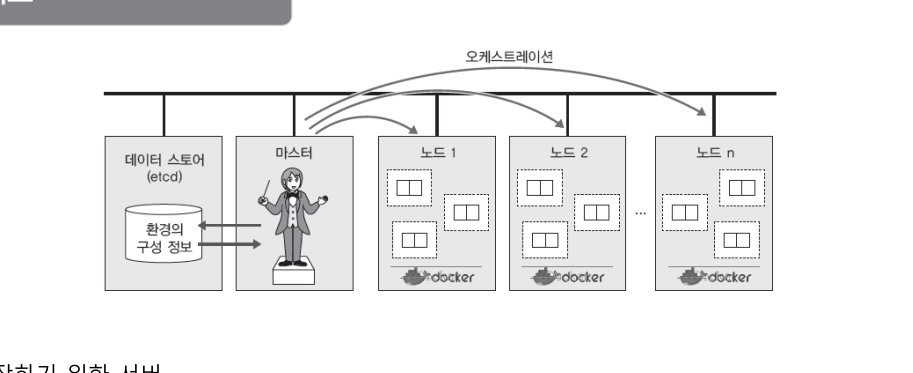
mysql>show databases; //mysql shell에서 databases 확인

mysql>use mysql; //mysql db 사용

mysql>quit; //mysql shell 종료

@root#exit; //bash shell 종료

# 구글 Kubernetes 개념

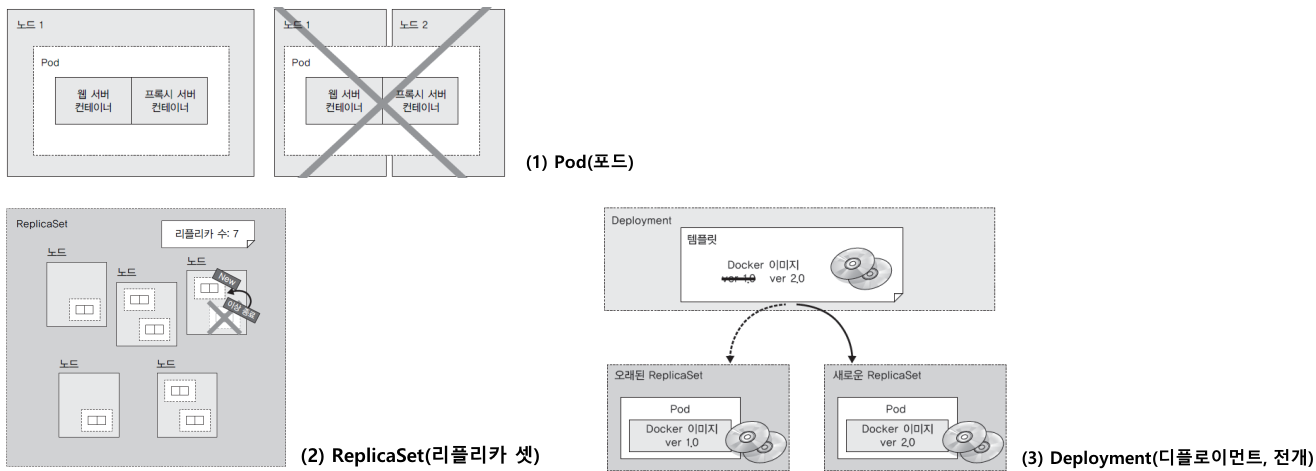


각 노드안에는 pod가 있으며 그 pod안에 컨테이너들을 넣을 수 있다.

마스터 노드를 통해서 각각의 pod의 컨테이너들을 관리한다고 볼 수 있다.

Pod는 내부의 컨테이너들을 관리할 수 있는 컨테이너들의 ip들을 가지고 있고,

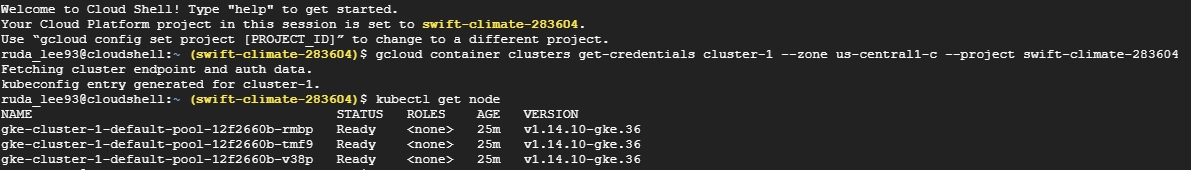
마스터 노드는 각 노드의 ip를 가지고 있으며 외부에선 노드ip를 통해서 오케스트레이션 한다.



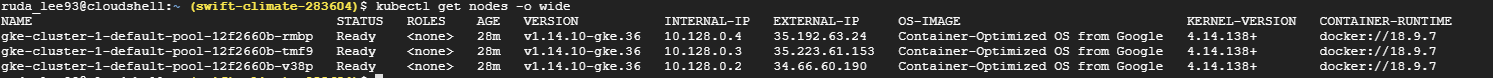
도커에 의해 생성된 컨테이너를 서비스는 클라우드를 통해 서비스를 하는데, 컨테이너(인스턴스)의 라이프 사이클을 관리해주기 위해 kubernetes 엔진을 통해 레플리카, 삭제 등을 하게 된다. 그렇게 하기 위해 컨테이너들을 pod로 감싸게 된 것.



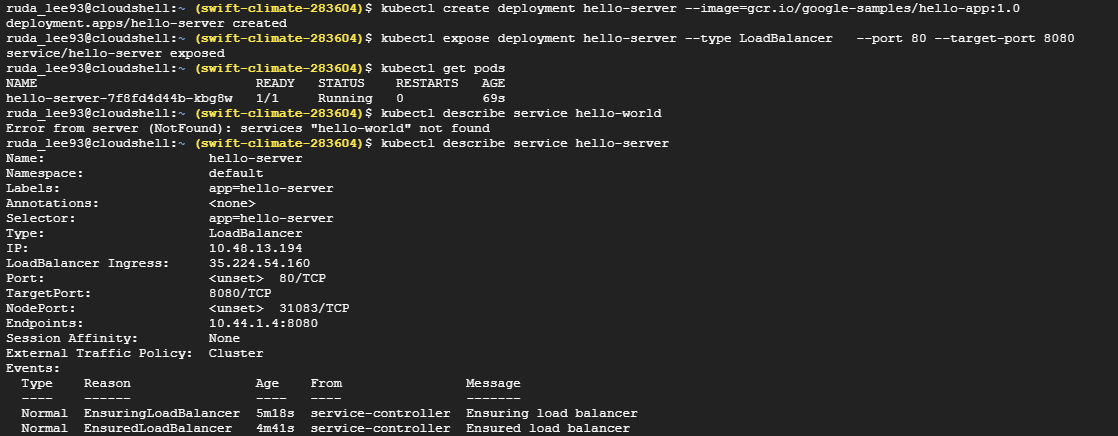
쿠버네티스의 클러스터에서 ‘연결’을 클릭하면 연결할 수 있는 shell 명령어를 복사해서 사용 할 수 있다.



연결 후, kubectl get nodes 명령어 확인.

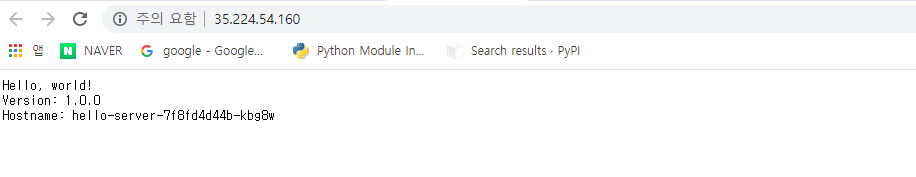


상세 정보를 제공하려면 –o wide 옵션을 주면 된다.

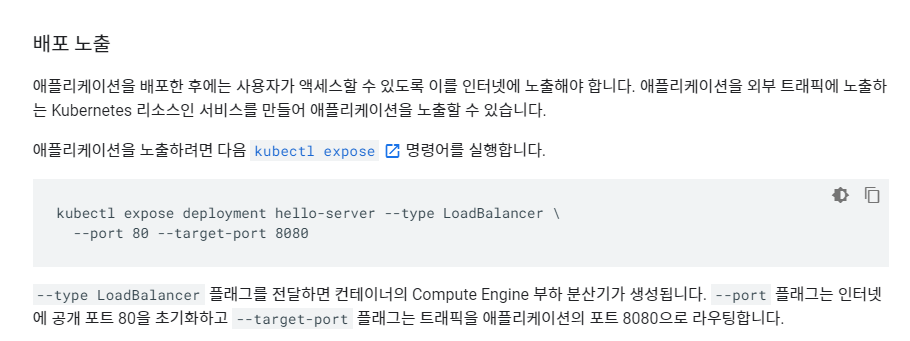




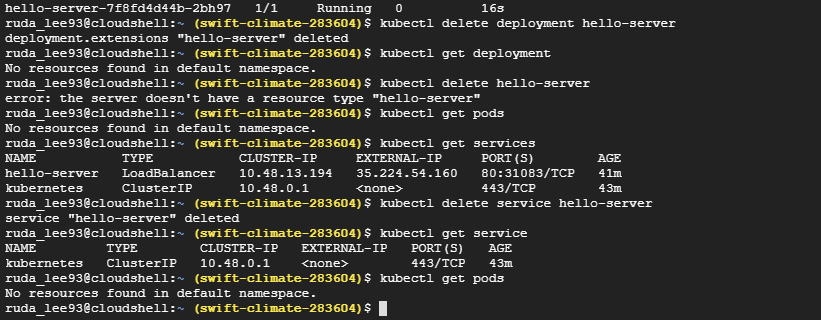
Hello-server라는 pod를 생성, 노출 후 외부 IP(LoadBalancer Ingress) 확인.



해당 ip를 입력하여 결과를 확인 하였음.



Service의 외부 ip를 loadbalancer로 잡아서 할당한다.



Deployment를 삭제하면 pod가 사라짐. (deployment가 살아있으면 pod를 지워도 지워지지 않음.)

이후, service를 따로 삭제 하게 됨.

7월 20일

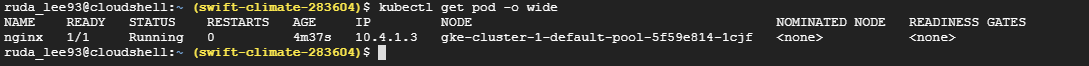


생성한 .yaml파일을 드래그앤 드롭으로 구글 쉘에 넣어주면 파일이 들어간다.



Pod.yaml파일로 pod 생성

Kubectl create –f pod.yaml



생성된 pod를 확인.

Kubectl exec –it nginx bash

생성된 컨테이너를 실행한다.



Kubectl port-forward nginx 8080:80 로 포트 포워드를 실행

Google cloud shell – 웹미리보기 8080 실행 // nginx container 확인

Kubectl get pod –o yaml 로 yaml 파일을 바로 얻어낼 수 있다.

#삭제

Kubectl delete all –-all //pod, node, sevice, replica, peployment 삭제 (namespace 등은 삭제안됨. 이후 클러스터 지워줘야 비용이 덜 발생.)