2024.07.17

오전

도커
docker run -d
--memory="1g"
--name memory_1g
nginx

Memory 제한

- 제한 단위는 b, k, m, g로 할당

옵션	의미
--memory, -m	컨테이너가 사용할 최대 메모리 양을 지정
--memory-swap	컨테이너가 사용할 스왑 메모리 영역에 대한 설정 메모리+스왑 생략 시 메모리의 2배가 설정됨
--memory-reservation	--memory 값보다 적은 값으로 구성하는 소프트 제한 값 설정
--oom-kill-disable	OOM Killer가 프로세스 kill하지 못하도록 보호

docker stats memory_1g
docker run -it --name swap_500m
--memory=200m
--memory-swap=500m

ubuntu:14.04**

cpu 제한

옵션	의미
--cpu-shares	컨테이너에 가중치를 설정해 해당 컨테이너가 cpu를 상대적으로 얼마나 사용할 수 있는지 나타냄 컨테이너에 cpu를 한 개씩 할당하는 방식이 아닌, 시스템에 존재하는 cpu를 어느 비중만큼 나눠 쓸 것인지를 명시하는 옵션 cpu 비중을 1024 값을 기반으로 설정되므로 2048이라고 할 경우 기본값보다 두 배 많은 cpu 자원을 할당
--cpuset-cpu	호스트에 cpu가 여러 개 있을 때 해당 옵션을 지정해 컨테이너가 특정 cpu만 사용하도록 설정 cpu 집중적인 작업이 필요하다면 여러 개의 cpu를 사용하도록 설정해 작업을 적절하게 분배하는 것이 좋음
--cpus	cpu의 개수를 직접 지정. 예를 들어 --cpus 옵션에 0.5를 설정하면 컨테이너의 cpu를 제한할 수 있음

docker run -d --name cpuset_2
--cpu-shares 2048
ubuntu:14.04

docker run -d --name cpuset_2
--cpuset-cpus=2
ubuntu:14.04**

docker run -d --name wordpressdb -e MYSQL_ROOT_PASSWORD=encore -e MYSQL_DATABASE=wordpress mysql:5.7

docker run -d -e WORDPRESS_DB_USER=root -e WORDPRESS_DB_PASSWORD=encore -e WORDPRESS_DB_NAME=wordpress --name wordpress --link wordpressdb:mysql -p 80:80 wordpress

docker run -d --volume=/:/rootfs:ro --volume=/var/run:/var/run:rw --volume=/sys:/sys:ro --volume=/var/lib/docker/:/var/lib/docker:ro -p 8080:8080 --name=cadvisor gcr.io/cadvisor/cadvisor:latest

docker pull registry:latest

DOCKER_OPTS=--insecure-registry localhost:5000

sudo vim /etc/docker/daemon.json
{ "insecure-registries": ["docker-registry:5000"] }**

sudo service docker restart
docker run --name personal-registry -d -p 5000:5000 registry

- mkdir temp && cd temp
- vim Dockerfile
- Dockerfile 생성

FROM ubuntu:22.04
MAINTAINER encore
CMD echo 'Hello, Docker!'

docker build -t encore/hello_docker .
docker run encore/hello_docker
docker tag encore/hello_docker localhost:5000/hello_docker
docker push localhost:5000/hello_docker

curl -X GET http://localhost:5000/v2/_catalog
{"repositories":["hello_docker"]}

- 서버에 있는 hello_docker 이미지 삭제
  docker rmi [image hash]
- 이미지 받기
- docker pull localhost:5000/hello_docker
  1 . 컨테이너 정지 및 삭제
- docker stop $(docker ps -a -q)
- docker rm $(docker ps -a -q)
  -> 컨테이너 삭제되면 내용 다 삭제됨...(안의 내용 못살림)
- 모든 이미지 삭제
- docker image prune (댕글링 이미지 삭제)
- docker rmi $(docker images | awk '{ print $3 }')

쿠버네티스 깔기

- myserver01
- myserver02
- myserver03
  sudo vim /etc/hosts 수정
  sudo vim /etc/netplan/00-installer-config.yaml

sudo netplan apply
hostname 변경

- 예) sudo hostnamectl set-hostname myserver01
  ssh 설정
- 3대 컴퓨터 암호 없이 서로 로그인
  **## 방화벽 off
  sudo ufw status
- acitvate 상태이면
- sudo ufw disable**
- 스왑 확인
- free -h
- cat /proc/swaps
- swap off 하기
- sudo swapoff --all
- 영구적으로 off하기
  sudo vim /etc/fstab
  `#/swap`.img none swap sw 0 0
- containerd
  sudo apt update
  sudo apt install containerd.io

네트워크 설정

- ipv4 포워딩 -> iptables
  cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
  overlay
  br_netfilter
  EOF

cat /etc/modules-load.d/k8s.conf
sudo modprobe overlay
sudo modprobe br_netfilter
lsmod | grep br_netfilter
lsmod | grep overay
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
EOF

sudo mkdir -p /etc/containerd
containerd config default | sudo tee /etc/containerd/config.toml > /dev/null
sudo vim /etc/containerd/config.toml
**아래 내용으로 변경 false -> true로 저장
명령어 모드 : /SystemdCgroup

- SystemdCgroup = true**
  sudo systemctl restart containerd
  sudo systemctl status containerd

sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.26/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.26/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl

sudo kubelet --version
sudo kubeadm version
sudo kubectl version --output=yaml

sudo kubeadm certs check-expiration
sudo kubeadm config images list
sudo kubeadm config images pull**

//sudo kubeadm config images pull --cri-socket /run/containerd/containerd.sock

master node에서
sudo kubeadm init --apiserver-advertise-address=[IP] --pod-network-cidr=192.168.0.0/16 --cri-socket /run/containerd/containerd.sock

문제가 생기면 전체에서 다 하고
sudo kubeadm reset
sudo kubeadm init --apiserver-advertise-address=[IP] --pod-network-cidr=192.168.0.0/16 --cri-socket /run/containerd/containerd.sock
mkdir -p
$HOME/.kube sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config sudo chown $(id -u) :

(id -g) $HOME/.kube/config
kubectl create -f ./calico/manifests/tigera-operator.yaml
kubectl create -f ./calico/manifests/custom-resources.yaml

- calico 설정

메인에서
git clone -b v3.26.3 https://github.com/projectcalico/calico.git
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/configsudochown$(id -u) :(id -g) $HOME/.kube/config
kubectl create -f ./calico/manifests/tigera-operator.yaml
kubectl create -f ./calico/manifests/custom-resources.yaml
kubectl get namespaces
kubectl get pods -n calico-system
kubectl get pods --all-namespaces
kubectl describe po tigera-operator-776b7d494d-9tmb9 -n calico-system
이미지가안되는데요

wget https://github.com/derailed/k9s/releases/download/v0.13.7/k9s_0.13.7_Linux_i386.tar.gz
tar xvzf k9s_0.13.7_Linux_i386.tar.gz
sudo mv k9s /usr/bin
k9s

```
ding: https://www.docker.com/increase-rate-limit
  Warning  Failed     38m (x4 over 39m)   kubelet            Error: ErrImagePull
  Warning  Failed     37m (x6 over 39m)   kubelet            Error: ImagePullBackOff
  Normal   BackOff    0s (x167 over 39m)  kubelet            Back-off pulling image "docker.io/calico/pod2daemon-flexvol
:v3.26.3"
```

scp -r images/ encore@IP:/home/encore
마스터노드에서 vim ~/.bashrc
export CONTAINER_RUNTIME_ENDPOINT=unix:///run/containerd/containerd.sock # containerd 사용 시
sudo ctr images import images/node
kubectl describe pod -n calico-system

안되면 wife바꿔서 접근해보기

오류 해결바법

마스터 노드를 제외하고는

sudo kubeadm reset
이랑 join만하기

마스터노드도
sudo kubeadm reset
sudo kubeadm init --apiserver-advertise-address=[IP] --pod-network-cidr=192.168.0.0/16 --cri-socket /run/containerd/containerd.sock
mkdir -p $HOME/.kube sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config sudo chown $(id -u):$(id -g) $HOME/.kube/config
kubectl create -f ./calico/manifests/tigera-operator.yaml
kubectl create -f ./calico/manifests/custom-resources.yaml
안되면 calico 파일 rm -rf 로 지우고
다시 git clone -b v3.26.3 https://github.com/projectcalico/calico.git

성공한방법

sudo kubeadm init --apiserver-advertise-address=[IP] --pod-network-cidr=192.168.0.0/16 --cri-socket /run/containerd/containerd.sock
mkdir -p $HOME/.kube sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config sudo chown $(id -u):$(id -g) $HOME/.kube/config
git clone -b v3.26.3 https://github.com/projectcalico/calico.git
kubectl create -f ./calico/manifests/tigera-operator.yaml
kubectl create -f ./calico/manifests/custom-resources.yaml