

ft_service 개정판!

우와 서브젝트가 조오금? 많이? 많이 바뀌었습니다! 그에 맞춘 새로운 개정판 등판!

<https://minikube.sigs.k8s.io/docs/handbook/dashboard/>

흐한의 home brew 설치하기 (C 빼곤 모든것을 아는 남자... 그는.. 대체..)

```
# 42seoul_global_random - Apr 3rd
hhan 8:54 PM
curl -fsSL https://rawgit.com/gcamerli/42brew/master/set.sh | zsh
○ 이거 하나면 되지 않나요?
```

```
curl -fsSL https://rawgit.com/gcamerli/42brew/master/set.sh | zsh
```

위의 명령어를 복붙으로 실행하시고~ 터미널을 재 시작 해주시면 적용이 됩니다~
저는 처음에 재 시작 해야 되는지 몰라서 왜 안될까 고민 했지만 흐한님이 해결 해 주셨습니다.

아래의 틀 박스를 활용해서~~용량을 확보 해주시면 된답니다.

틀 사용법은 굉장히 쉽기 때문에 ~ 패스!!!

<https://github.com/alexandregy/42toolbox>

브루를 설치하셨다면!! 한방 컷!

<https://kubernetes.io/ko/docs/tasks/tools/install-minikube/>

```
brew install minikube
```

그리고 Launchpad -> managed software center -> virtualbox 를 설치해주세요!!

```
minikube start --driver=virtualbox
```

```
c1r7s7% minikube start --driver=virtualbox
😊 Darwin 10.14.6 위의 minikube v1.11.0
✨ 유저 환경 설정 정보에 기반하여 virtualbox 드라이버를 사용하는 중
💿 가상 머신 부트 이미지 다운로드 중 ...
> minikube-v1.11.0.iso.sha256: 65 B / 65 B [-----] 100.00% ? p/s 0s
> minikube-v1.11.0.iso: 174.99 MiB / 174.99 MiB [] 100.00% 27.28 MiB p/s 7s
👍 Starting control plane node minikube in cluster minikube
💻 Downloading Kubernetes v1.18.3 preload ...
> preloaded-images-k8s-v3-v1.18.3-docker-overlay2-amd64.tar.lz4: 526.01 MiB
🔥 virtualbox VM (CPUs=2, Memory=6000MB, Disk=20000MB) 를 생성하는 중 ...
🐳 쿠버네티스 v1.18.3 을 Docker 19.03.8 런 타임으로 설치하는 중
🔍 Verifying Kubernetes components...
⭐ Enabled addons: default-storageclass, storage-provisioner
🌟 끝났습니다! 이제 kubectl 이 "minikube" 를 사용할 수 있도록 설정되었습니다
```

이렇게 하면 설치가 끝납니다. 그러나 우리는 용량의 문제로 인해서 goinfre로 옮겨야 할 수 도 있어요!

ryukim의 용량 아끼는 방법!!
goinfre에 옮겨서 용량의 문제에서 자유로 워 질 수 있습니다.

```
mv .minikube goinfre/minikube  
ln -s goinfre/minikube/.minikube .minikube  
minikube delete  
minikube start --driver=virtualbox
```

minikube docker-env 명령어를 실행하고 나오는 마지막 줄을 추가하시면 도커 로컬 이미지를 사용 할 수 있습니다.

- **The Kubernetes web dashboard. This will help you manage your cluster.**

ft_service 디렉토리 구조

```
ft_service 현재 디렉토리 구조  
└── setup.sh  
    └── srcs  
        └── clean.sh
```

새로 추가 되는 디렉토리, 파일은 초록색으로 표시 하도록 하겠습니다.



클러스터의 백도어인 대시보드를 설치하여 한눈에 모든 클러스터 관리!

미니쿠브에서는 간단하게 활성화 해주면 오케이! 별도의 설치 없음!

<https://minikube.sigs.k8s.io/docs/handbook/dashboard/>

setup.sh

```
"대시보드 활성화"  
https://minikube.sigs.k8s.io/docs/handbook/dashboard/
```

대시보드는 모든것을 하고 나중에 작동하기 위해 일단 넣어두기만 하고 주석 처리 하겠습니다.

clean.sh

```
아직 아무것도 없음
```

- **The Load Balancer which manages the external access of your services. It will be**

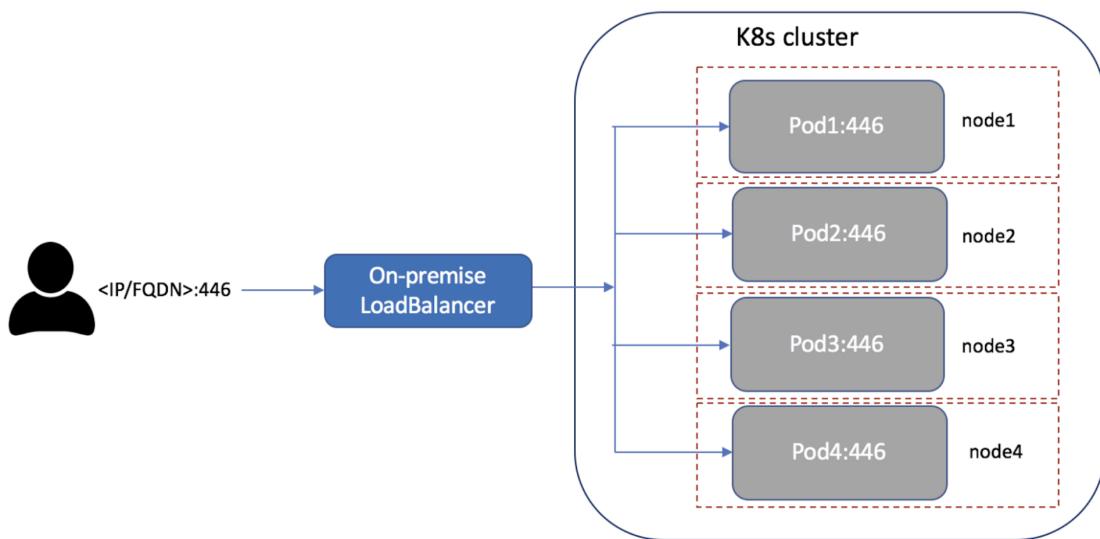
the only entry point to your cluster. You must keep the ports associated with the service

ft_service 디렉토리 구조

```
ft_service 현재 디렉토리 구조
├── setup.sh
└── srcs
    ├── clean.sh
    └── metallb (디렉토리)
        └── config.yaml
```

지금 가이드를 적고 있는 이 순간에도 네트워크 개념이 없어서.... 혼돈과 공포.. 계속 찾아보게 되는.....
아마도 저와 같은 분들이 많지 않을까 싶습니다.

부하 분산을 위해서 가상IP를 통해 여러 서버에 접속하도록 분배하는 기능!!이라고 나오더군요!



외부에서 접속 할 수 있고 그것은 각각의 IP를 갖는 것 입니다.

LoadBalancer access

A LoadBalancer service is the standard way to expose a service to the internet. With this method, each service gets its own IP address.

과제에서 친절하게 metallb를 사용 하라고 나와 있어서 이것을 쉽게 사용 하도록 합시다.

아래의 가이드를 복사 붙여넣기를 통해 잘 따라 가도록 합니다!

<https://metallb.universe.tf/installation/>

혹시라도 사이트 가서 하기 번거로운 분들을 위해 남겨 둡니다!

ARP 모드 활성화

ARP는 주소 결정 프로토콜로 A → B에게 전송하려고 할 때 B의 주소를 모른다면 IP주소를 기반으로 MAC주소를 알아낸다고 하는데 이 부분은 베이커라서 잘 모르겠네요~ 찾아보세요!!
이 부분은 학습 할때 처음에 설정을 변경하면 될 것 같아서 setup.sh에 포함하지 않겠습니다!

```
# see what changes would be made, returns nonzero returncode if different
kubectl get configmap kube-proxy -n kube-system -o yaml | \
sed -e "s/strictARP: false/strictARP: true/" | \
kubectl diff -f - -n kube-system

# actually apply the changes, returns nonzero returncode on errors only
kubectl get configmap kube-proxy -n kube-system -o yaml | \
sed -e "s/strictARP: false/strictARP: true/" | \
kubectl apply -f - -n kube-system
```

그리고 명령어를 실행하면 무언가 수정 되었다고 나옵니다.
그렇다면 O.K! 넘어 갑시당!

MetaLB 설치

```
kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/namespace.yaml
kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/metallb.yaml
# On first install only
kubectl create secret generic -n metallb-system memberlist --from-literal=secretkey="$(openssl rand -base64 128)"
```

잘 설치 되었는지 확인 해 보도록 합시다~

```
kkubectl get all -n metallb-system
```

```
~
x ~ (zsh)
Last login: Sun Jul  5 20:28:14 on ttys003
d
~
(base) base > kubectl get all -n metallb-system
NAME           READY   STATUS    RESTARTS   AGE
pod/controller-57f648cb96-xvqv8   1/1     Running   0          6m36s
pod/speaker-2mc26      1/1     Running   0          6m36s

NAME        DESIRED  CURRENT  READY   UP-TO-DATE  AVAILABLE  NODE SELECTOR   AGE
daemonset.apps/speaker  1         1        1       1           1          beta.kubernetes.io/os=linux  6m36s

NAME           READY   UP-TO-DATE  AVAILABLE  AGE
deployment.apps/controller  1/1     1           1          6m36s

NAME        DESIRED  CURRENT  READY   AGE
replicaset.apps/controller-57f648cb96  1         1        1       6m36s

~
(base) base > 
```

-n : 네임스페이스를 의미합니다. 그리고 파드가 정상적으로 running중이면 됩니다!!

아래의 링크로 이동해서 구성파일을 만들어 봅시다!

<https://metallb.universe.tf/configuration/>

config.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
      addresses:
      - 192.168.99.95-192.168.99.105
```

그리고 !

```
kubectl apply -f config.yaml
```

여기까지 왔다면 우리는 minikube에서 말하는 로드밸런싱 서비스를 구축 하였습니다!!!

이 구축 된 것이 정상적으로 외부에서 접속 할 수 있는 IP를 할당하고 그것을 브라우저를 통해 접속 할 수 있는지
kubernetes docs의 예시를 통해 빠르게 확인 해보도록 하고 챕터를 마치도록 하겠습니다.

<https://kubernetes.io/ko/docs/tutorials/stateless-application/expose-external-ip-address/>

위의 예시를 따라해보시고!! externalip:port로 접속 할 수 있다면 제대로 작동!

테스트 하시고 지우는 것을 잊지 마세요!!

setup.sh

```
echo "metallB manifest"
kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/namespace.yaml
kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/metallb.yaml
kubectl create secret generic -n metallb-system memberlist --from-literal=secretkey="$(openssl rand -base64 128)"
cd srcs/metallb
kubectl apply -f config.yaml

# echo "dashboard activate"
# https://minikube.sigs.k8s.io/docs/handbook/dashboard/
```

clean.sh

```
echo "metallB manifest delete"
kubectl delete -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/namespace.yaml
```

네임스페이스를 정리하기 때문에 나머지 부분은 삭제하지 않아도 자동으로 삭제 됩니다!

- A Nginx server listening on ports 80 and 443. Port 80 will be in http and should be a systematic redirection of type 301 to 443, which will be in https. The page displayed does not matter.

우리는 위에서 external ip를 노출해서 크롬 또는 curl을 통해 접속 할 수 있음을 알게 되었습니다.

이제 우리는 nginx container를 만들어서 80으로 접속하면 443으로 리다이렉트 되도록 만들겠습니다.

ft_service 디렉토리 구조

```
ft_service 현재 디렉토리 구조
├── setup.sh
└── srcs
    ├── clean.sh
    ├── metallb (디렉토리)
    │   └── config.yaml
    └── nginx (디렉토리)
        ├── 10-listen-on-ipv6-by-default.sh
        ├── 20-envsubst-on-templates.sh
        ├── default.conf
        ├── docker-entrypoint.sh
        ├── Dockerfile
        ├── Makefile
        └── nginx.yaml
```

파일이 많다고 겁 먹지 마세요!! nginx 관련 된 것들은 모두 nginx official 입니다.

우선 !! 아래의 링크로 가셔서 nginx 실행에 필요한 것들을 다운로드 받아 주세요!

<https://github.com/nginxinc/docker-nginx/tree/2ef3fa66f2a434cd5e44e35a02f4ac502cf50808/mainline/alpine>

잘 복사하셨다면 저 링크에서 얻지 못하는 Makefile, default.conf, nginx.yaml만 만들면 됩니다.

Makefile

우리는 openssl을 만들고 지울 수 있는 Makefile을 만들어 보겠습니다.

```
KEY = /Users/nakim/Desktop/nginx.key
CERT = /Users/nakim/Desktop/nginx.crt

keys:
    # The CName used here is specific to the service specified in nginx-app.yaml.
    openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout $(KEY) -out $(CERT) -subj "/CN=nginxsvc/0=nginxsvc"

clean:
    rm $(KEY)
    rm $(CERT)
```

바탕화면에 key, crt를 만들고 그것을 통해 ssl 설정을 하도록 하겠습니다.

경로는 닉네임이 nakim이 들어 있는데 이 부분은 자신의 아이디로 교체해주세요 !!

default.conf

80포트로 접근하는 모든 것을 443으로 리 디렉트 하도록 설정 할 겁니다.

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name _;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl default_server;
    root /usr/share/nginx/html;
    index index.html;

    server_name localhost;
    ssl_certificate /etc/nginx/ssl/tls.crt;
    ssl_certificate_key /etc/nginx/ssl/tls.key;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

nginx에서 http → https로 리 디렉트하는 방법은 별도의 서버 블록을 만들고 그것을 301 리 �렉션을 해주면 됩니다!!

nginx.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress-nginx
  labels:
    run: nginx
spec:
  type: LoadBalancer
  ports: #포트가 두개 일때 포트 이름을 명시해줘야 합니다.
  - port: 80
    protocol: TCP
    name: http
  - port: 443
    protocol: TCP
    name: https
  selector:
    run: nginx
    tier: backend
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-nginx
spec:
  selector:
```

```

matchLabels:
  run: nginx
  tier: backend
replicas: 1
template:
  metadata:
    labels:
      run: nginx
      tier: backend
  spec:
    volumes:
      - name: secret-volume
        secret:
          secretName: nginxsecret
      - name: configmap-volume
        configMap:
          name: nginxconfigmap
    containers:
      - name: nginx
        image: nakim_nginx:1.0
        imagePullPolicy: Never #이미지가 로컬에 있다고 가정합니다.
        ports:
          - containerPort: 443
          - containerPort: 80
        # livenessProbe:
        #   httpGet:
        #     path: /readme.html
        #     port: 80
        #   initialDelaySeconds: 30
        #   timeoutSeconds: 1
        volumeMounts: #nginx, ssl 설정을 볼륨 마운트를 통해 전달!
          - mountPath: /etc/nginx/ssl
            name: secret-volume
          - mountPath: /etc/nginx/conf.d
            name: configmap-volume

```

livenessprobe를 실질적으로 서버를 가동 할때 사용하는 옵션입니다.

서비스 도중 어떠한 오류로 인해 파드안에서 실행 중인 nginx가 구동을 멈추어도 pod는 계속 정상 작동 중으로 나옵니다 .그런 경우 위의 옵션을 통해 지속적으로 정상 응답 하는지 확인하고 그렇지 않다면 재시작 시킵니다.
우리의 과제엔 필요 없을 것 같아서 이런것도 있구나 하고 참고사항으로 넣어 두었습니다.



eval \$(minikube -p minikube docker-env)

이 명령어는 minikube 인스턴스 내에서 도커 데몬을 사용 할 수 있게 해줍니다. 그래서 우리가 로컬에 저장된 도커 이미지를 사용 할 수 있게 해줍니다. 이 명령어를 깜빡 하신다면 컨테이너 에러가 발생하게 될 겁니다.

setup.sh

```

echo "metallB manifest"
kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/namespace.yaml
kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/metallb.yaml
kubectl create secret generic -n metallb-system memberlist --from-literal=secretkey="$(openssl rand -base64 128)"
cd srcs/metallb
kubectl apply -f config.yaml
echo "fieldPath: status.hostIP"
echo "nginx directory"
cd ../nginx
echo "ssl create"
make keys
echo "nginx ssl secret create"
kubectl create secret tls nginxsecret --key /Users/nakim/Desktop/nginx.key --cert /Users/nakim/Desktop/nginx.crt
echo "nginx configmap create"
kubectl create configmap nginxconfigmap --from-file=default.conf
echo "nginx image build"
docker build -t nakim_nginx:1.0 .
echo "nginx deployment"
kubectl apply -f nginx.yaml

# echo "dashboard activate"
# https://minikube.sigs.k8s.io/docs/handbook/dashboard/

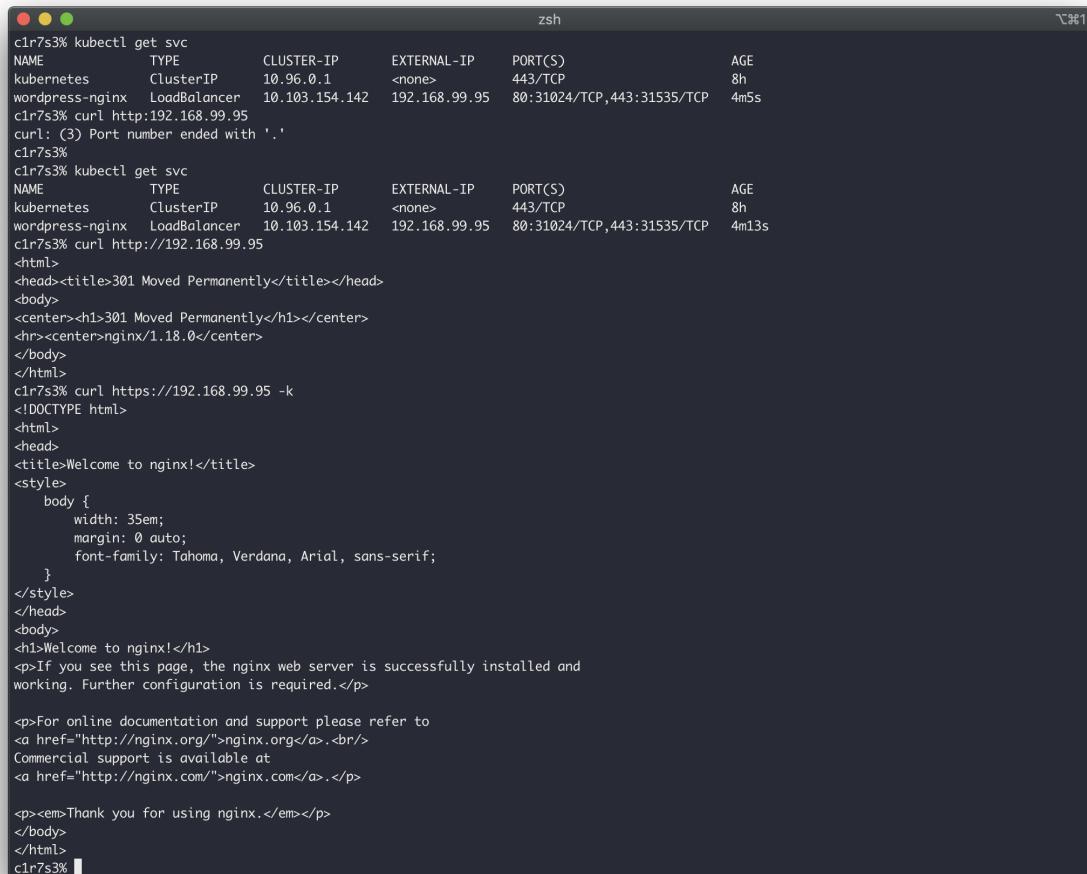
```

clean.sh

```
echo "metalLB manifest delete"
kubectl delete -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/namespace.yaml
echo "nginx directory"
cd ../nginx
echo "ssl delete"
make clean
echo "nginx ssl secret delete"
kubectl delete secret nginxsecret
echo "nginx configmap create"
kubectl delete configmap nginxconfigmap
echo "nginx deployment delete"
kubectl delete -f nginx.yaml
sleep 5
echo "nginx image delete"
docker rmi nakim_nginx:1.0
```

setup.sh와 clean.sh가 다른 점은 deployment를 먼저 삭제해주시고 이미지를 지워야 지워진다는 것 입니다.
deployment를 낱두고 이미지를 삭제 하려고 하면 이미지가 사용 중이라서 삭제가 되지 않는다고 에러 발생!!

여기까지 작성 하였다면 이제 setup.sh를 실행하고 결과를 확인 해 봅시다.



```
c1r7s3% kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes   ClusterIP  10.96.0.1    <none>        443/TCP     8h
wordpress-nginx  LoadBalancer  10.103.154.142  192.168.99.95  80:31024/TCP,443:31535/TCP  4m5s
c1r7s3% curl http://192.168.99.95
curl: (3) Port number ended with '.'
c1r7s3%
c1r7s3% kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes   ClusterIP  10.96.0.1    <none>        443/TCP     8h
wordpress-nginx  LoadBalancer  10.103.154.142  192.168.99.95  80:31024/TCP,443:31535/TCP  4m13s
c1r7s3% curl http://192.168.99.95
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx/1.18.0</center>
</body>
</html>
c1r7s3% curl https://192.168.99.95 -k
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
}
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
c1r7s3%
```

서비스의 external ip를 확인하고 curl 아이피해서 리다이렉트 되는 것을 확인 하면 됩니다!

그리고 크롬 브라우저에서 http로 접근해도 https로 나오는 것을 확인 할 수 있을 겁니다.

- A FTPS server listening on port 21

nakim은 이 부분을 하다가 시간을 많이 날렸는데.. 추후에 하시는 분들은 빠르게 고고싱 하시길!!

우리는 클라이언트 → 서버로 접속해서 ftps를 사용 할 것이기 때문에 패시브 모드로 진행하게 됩니다.

ft_service 디렉토리 구조

```
ft_service 현재 디렉토리 구조
├── setup.sh
└── srcs
    ├── clean.sh
    ├── metallb (디렉토리)
    │   └── config.yaml
    ├── nginx (디렉토리)
    │   ├── 10-listen-on-ipv6-by-default.sh
    │   ├── 20-envsubst-on-templates.sh
    │   ├── default.conf
    │   ├── docker-entrypoint.sh
    │   ├── Dockerfile
    │   ├── Makefile
    │   └── nginx.yaml
    └── ftps (디렉토리)
        ├── Dockerfile
        ├── ftps.yaml
        ├── Makefile
        ├── run-vsftpd.sh
        ├── vsftpd.conf
        └── FileZilla
```

<https://github.com/lhauspie/docker-vsftpd-alpine>

위의 깃을 클론 하시거나 다운로드 받고 시작 하도록 하겠습니다.

Makefile

우리는 openssl을 만들고 지울 수 있는 Makefile을 만들어 보겠습니다.

```
PEM = vsftpd.pem

pem:
    # The CName used here is specific to the service specified in nginx-app.yaml.
    openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout $(PEM) -out $(PEM) -subj "/CN=nginxsvc/0=nginxsvc"

clean:
    rm $(PEM)
```

nginx의 ssl을 만들던 makefile을 활용해서 pem을 ftps 폴더 내부에 만들도록 하겠습니다.

같은 파일에 아웃을 두번 하게 되면 key, cert를 모두 갖고 있는 pem파일이 만들어 집니다.

Dockerfile

수정 되거나 추가된 부분은 빨간색으로 표시 하였으니 그 부분만 추가 해주시면 됩니다.

```
FROM alpine:3.9.4

MAINTAINER Logan HAUSPIE <logan.hauspie.pro@gmail.com>
LABEL Description="vsftpd Docker image based on Alpine. Supports passive mode and virtual users." \
      License="Apache License 2.0" \
      Usage="docker run --rm -it --name vsftpd -p [HOST_CONNECTION_PORTS]:20-22 -p [HOST_FTP_PORTS]:21100-21110 lhauspie/vsftpd-alpine" \
      Version="${VERSION}"

# RUN apk update and install dependencies
RUN apk update \
    && apk upgrade \
    && apk --update --no-cache add \
        bash \
        vsftpd

ENV FTP_USER=user \
```

```

FTP_PASS=pass \
PASV_ENABLE=YES \
PASV_ADDRESS= \
PASV_ADDRESS_INTERFACE=eth0 \
PASV_ADDR_RESOLVE=YES \
PASV_MIN_PORT=30010 \
PASV_MAX_PORT=30015

RUN mkdir -p /home/vsftpd/
RUN chown -R ftp:ftp /home/vsftpd/

COPY vsftpd.conf /etc/vsftpd/vsftpd.conf
COPY run-vsftpd.sh /usr/sbin/
COPY vsftpd.pem /etc/ssl/private/vsftpd.pem
RUN chmod +x /usr/sbin/run-vsftpd.sh

EXPOSE 20-22 21100-21110

CMD /usr/sbin/run-vsftpd.sh

```

우리는 패시브 모드에서 사용 할 포트 5개와 ssl설정을 위해 pem파일을 COPY명령어를 통해 보낼 것 입니다.

ftps.yaml

이 부분에 대해 굉장히 많이 헤맸는데 정확한 아키텍쳐를 찾지 못해서~!!
 명시적으로 nodeport를 사용해두 될까? 고민을 했는데 로드 밸런서는 노드포트를 포함한 상위 개념이라서..
 그냥 사용하기로 하였습니다 service의 타입이 load balancer이기 때문에 무관!!하다고 생각 합니다.

```

apiVersion: v1
kind: Service
metadata:
  name: ftps
  labels:
    app: ftps
spec:
  ports:
    - port: 21
      protocol: TCP
      name: ftps
    - port: 20
      protocol: TCP
      name: ftps-data
    - port: 30010
      protocol: TCP
      nodePort: 30010
      name: ftps-pasv1
    - port: 30011
      protocol: TCP
      nodePort: 30011
      name: ftps-pasv2
    - port: 30012
      protocol: TCP
      nodePort: 30012
      name: ftps-pasv3
    - port: 30013
      protocol: TCP
      nodePort: 30013
      name: ftps-pasv4
      nodePort: 30013
    - port: 30014
      protocol: TCP
      nodePort: 30014
      name: ftps-pasv5
  selector:
    app: ftp
  type: LoadBalancer
---
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: ftps
  labels:
    app: ftps
spec:
  selector:
    matchLabels:
      app: ftps
  replicas: 1
  strategy:
    type: Recreate
  template:

```

```

metadata:
  labels:
    app: ft�
spec:
  containers:
    - image: nakim_ft�:1.0
      imagePullPolicy: Never
      name: ft�
      env:
        - name: NODE_IP
          valueFrom:
            fieldRef:
              fieldPath: status.hostIP
      ports:
        - containerPort: 21
        - containerPort: 30010
        - containerPort: 30011
        - containerPort: 30012
        - containerPort: 30013
        - containerPort: 30014
        - containerPort: 30015

```

패시브모드에서 NAT이 포함되어 구현하게 되면 약간 문제가 생깁니다.

클라이언트는 service의 external ip(ex:111.111.111.111)로 요청하게 되고 로드 밸런서 뒤에 있는 FTP서버가 개방 할 포트와 ip(222.222.222.222:30010~30015)를 전달 합니다.

그래서 클라이언트가 제대로 된 응답이라고 알 수 있도록 pasv_address를 설정하게 됩니다.

pasv_address는 minikube_ip이므로 쿠버네티스 환경변수를 사용해서 값을 저장 합니다.

run-vsftpd.sh

이 부분에서는 한줄만 수정하면 됩니다!

위에서 언급한 pasv_address를 설정 하는 부분입니다.

```

#!/bin/bash

# You can set PASV_ADDRESS_INTERFACE to the name of the interface you'd like to
# bind to and this will look up the IP and set the proper PASV_ADDRESS value.
if [ -z "$PASV_ADDRESS" ]; then
  echo "PASV_ADDRESS env variable is not set"
  if [ -n "$PASV_ADDRESS_INTERFACE" ]; then
    echo "attempt to guess the PASV_ADDRESS from PASV_ADDRESS_INTERFACE"
    PASV_ADDRESS=$(ip -o -4 addr list $PASV_ADDRESS_INTERFACE | head -n1 | awk '{print $4}' | cut -d/ -f1)
    if [ -z "$PASV_ADDRESS" ]; then
      echo "Could not find IP for interface '$PASV_ADDRESS_INTERFACE', exiting"
      exit 1
    fi
    echo "==> Found address '$PASV_ADDRESS' for interface '$PASV_ADDRESS_INTERFACE', setting PASV_ADDRESS env variable..."
  fi
else
  echo "PASV_ADDRESS is set so we use it directly"
fi

# Add the FTP_USER, change his password and declare him as the owner of his home folder and all subfolders
addgroup -g 433 -S $FTP_USER
adduser -u 431 -D -G $FTP_USER -h /home/vsftpd/$FTP_USER -s /bin/false $FTP_USER
echo "$FTP_USER:$FTP_PASS" | /usr/sbin/chpasswd
chown -R $FTP_USER:$FTP_USER /home/vsftpd/$FTP_USER

# Update the vsftpd.conf according to env variables
sed -i "s/anonymous_enable=YES/anonymous_enable=NO/" /etc/vsftpd/vsftpd.conf
sed -i "s/pasv_enable=.*/pasv_enable=$PASV_ENABLE/" /etc/vsftpd/vsftpd.conf
sed -i "s/pasv_address=.*/pasv_address=$NODE_IP/" /etc/vsftpd/vsftpd.conf
# sed -i "s/pasv_address=.*/pasv_address=$PASV_ADDRESS/" /etc/vsftpd/vsftpd.conf
sed -i "s/pasv_addr_resolve=.*/pasv_addr_resolve=$PASV_ADDR_RESOLVE/" /etc/vsftpd/vsftpd.conf
sed -i "s/pasv_max_port=.*/pasv_max_port=$PASV_MAX_PORT/" /etc/vsftpd/vsftpd.conf
sed -i "s/pasv_min_port=.*/pasv_min_port=$PASV_MIN_PORT/" /etc/vsftpd/vsftpd.conf

# Run the vsftpd server
/usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf

```

pass_address를 컨테이너 초기화 후에 환경 변수가 존재 하면 그것을 통해 설정을 변경해 주는 부분입니다.

run-vsftpd.sh

```
# Example config file /etc/vsftpd.conf
#
# The default compiled in settings are fairly paranoid. This sample file
# loosens things up a bit, to make the ftp daemon more usable.
# Please see vsftpd.conf.5 for all compiled in defaults.
#
# READ THIS: This example file is NOT an exhaustive list of vsftpd options.
# Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's
# capabilities.
#
# Allow anonymous FTP? (Beware - allowed by default if you comment this out).
anonymous_enable=NO
#
# Uncomment this to allow local users to log in.
#local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
#write_enable=YES
#
# Default umask for local users is 077. You may wish to change this to 022,
# if your users expect that (022 is used by most other ftppd's)
#local_umask=022
#
# Uncomment this to allow the anonymous FTP user to upload files. This only
# has an effect if the above global write enable is activated. Also, you will
# obviously need to create a directory writable by the FTP user.
#anon_upload_enable=YES
#
# Uncomment this if you want the anonymous FTP user to be able to create
# new directories.
#anon_mkdir_write_enable=YES
#
# Activate directory messages - messages given to remote users when they
# go into a certain directory.
dirmessage_enable=YES
#
# Activate logging of uploads/downloads.
xferlog_enable=YES
#
# Make sure PORT transfer connections originate from port 20 (ftp-data).
connect_from_port_20=YES
#
# If you want, you can arrange for uploaded anonymous files to be owned by
# a different user. Note! Using "root" for uploaded files is not
# recommended!
#chown_uploads=YES
#chown_username=whoever
#
# You may override where the log file goes if you like. The default is shown
# below.
#xferlog_file=/var/log/vsftpd.log
#
# If you want, you can have your log file in standard ftppd xferlog format.
# Note that the default log file location is /var/log/xferlog in this case.
#xferlog_std_format=YES
#
# You may change the default value for timing out an idle session.
#idle_session_timeout=600
#
# You may change the default value for timing out a data connection.
#data_connection_timeout=120
#
# It is recommended that you define on your system a unique user which the
# ftp server can use as a totally isolated and unprivileged user.
#nopriv_user=ftpsecure
#
# Enable this and the server will recognise asynchronous ABOR requests. Not
# recommended for security (the code is non-trivial). Not enabling it,
# however, may confuse older FTP clients.
#async_abor_enable=YES
#
# By default the server will pretend to allow ASCII mode but in fact ignore
# the request. Turn on the below options to have the server actually do ASCII
# mangling on files when in ASCII mode.
# Beware that on some FTP servers, ASCII support allows a denial of service
# attack (DoS) via the command "SIZE /big/file" in ASCII mode. vsftpd
# predicted this attack and has always been safe, reporting the size of the
# raw file.
# ASCII mangling is a horrible feature of the protocol.
#ascii_upload_enable=YES
#ascii_download_enable=YES
#
```

```

# You may fully customise the login banner string:
#ftpd_banner=Welcome to blah FTP service.
#
# You may specify a file of disallowed anonymous e-mail addresses. Apparently
# useful for combatting certain DoS attacks.
#deny_email_enable=YES
# (default follows)
#banned_email_file=/etc/vsftpd.banned_emails
#
# You may specify an explicit list of local users to chroot() to their home
# directory. If chroot_local_user is YES, then this list becomes a list of
# users to NOT chroot().
# (Warning! chroot'ing can be very dangerous. If using chroot, make sure that
# the user does not have write access to the top level directory within the
# chroot)
#chroot_local_user=YES
#chroot_list_enable=YES
# (default follows)
#chroot_list_file=/etc/vsftpd.chroot_list
#
# You may activate the "-R" option to the builtin ls. This is disabled by
# default to avoid remote users being able to cause excessive I/O on large
# sites. However, some broken FTP clients such as "ncftp" and "mirror" assume
# the presence of the "-R" option, so there is a strong case for enabling it.
#ls_recurse_enable=YES
#
# When "listen" directive is enabled, vsftpd runs in standalone mode and
# listens on IPv4 sockets. This directive cannot be used in conjunction
# with the listen_ipv6 directive.
listen=YES
#
# This directive enables listening on IPv6 sockets. To listen on IPv4 and IPv6
# sockets, you must run two copies of vsftpd with two configuration files.
# Make sure, that one of the listen options is commented !!
#listen_ipv6=YES
local_enable=YES
chroot_local_user=YES
allow_writeable_chroot=YES
background=NO
ftpd_banner=Welcome to FTP Server
dirmessage_enable=YES
max_clients=10
max_per_ip=5
write_enable=YES
local_umask=022
passwd_chroot_enable=yes
pasv_enable=YES
listen_ipv6=NO
seccomp_sandbox=NO

# TO BE REPLACE BY sed in the run-vsftpd.sh
pasv_enable=$PASV_ENABLE
pasv_address=$PASV_ADDRESS
pasv_addr_resolve=$PASV_ADDR_RESOLVE
pasv_max_port=$PASV_MAX
pasv_min_port=$PASV_MIN
rsa_cert_file=/etc/ssl/private/vsftpd.pem
rsa_private_key_file=/etc/ssl/private/vsftpd.pem
ssl_enable=YES
allow_anon_ssl=NO
force_local_data_ssl=YES
force_local_logins_ssl=YES
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO
require_ssl_reuse=NO
ssl_ciphers=HIGH

```

빨간색으로 처리 된 부분이 ftps를 사용 할 수 있게 해주는 것인데 으흠.. 여기는 따로 설명하기 보다는 링크를 통해 어떠한 역할을 하는지 체크 하는 것이 좋을 것 같습니다.

<https://www.digitalocean.com/community/tutorials/how-to-set-up-vsftpd-for-a-user-s-directory-on-debian-10>

setup.sh

```

echo "metallB manifest"
kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/namespace.yaml
kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/metallb.yaml

```

```

kubectl create secret generic -n metallb-system memberlist --from-literal=secretkey="$(openssl rand -base64 128)"
cd srcs/metallb
kubectl apply -f config.yaml
echo "fieldPath: status.hostIP"
echo "nginx directory"
cd ../nginx
echo "ssl create"
make keys
echo "nginx ssl secret create"
kubectl create secret tls nginxsecret --key /Users/nakim/Desktop/nginx.key --cert /Users/nakim/Desktop/nginx.crt
echo "nginx configmap create"
kubectl create configmap nginxconfigmap --from-file=default.conf
echo "nginx image build"
docker build -t nakim_nginx:1.0 .
echo "nginx deployment"
kubectl apply -f nginx.yaml
cd ../ftps
echo "ftps ssl pem create"
make pem
echo "ftps image build"
docker build -t nakim_ftps:1.0 .
echo "ftps deployment"
kubectl apply -f ftps.yaml

# echo "dashboard activate"
# https://minikube.sigs.k8s.io/docs/handbook/dashboard/

```

clean.sh

```

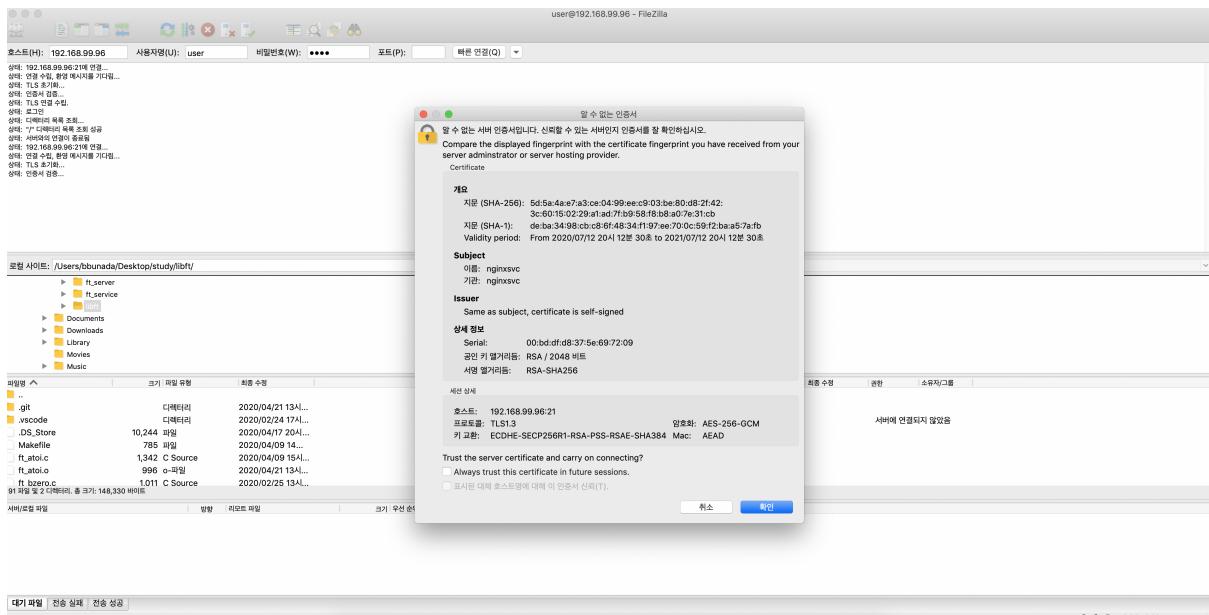
echo "metallB manifest delete"
kubectl delete -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/namespace.yaml
echo "nginx directory"
cd ../nginx
echo "ssl delete"
make clean
echo "nginx ssl secret delete"
kubectl delete secret nginxsecret
echo "nginx configmap create"
kubectl delete configmap nginxconfigmap
echo "nginx deployment delete"
kubectl delete -f nginx.yaml
sleep 5
echo "nginx image delete"
docker rmi nakim_nginx:1.0
cd ../ftps
echo "ftps ssl pem delete"
make clean
echo "ftps deployment delete"
kubectl delete -f ftps.yaml
sleep 15
echo "ftps image delete"
docker rmi nakim_ftps:1.0

```

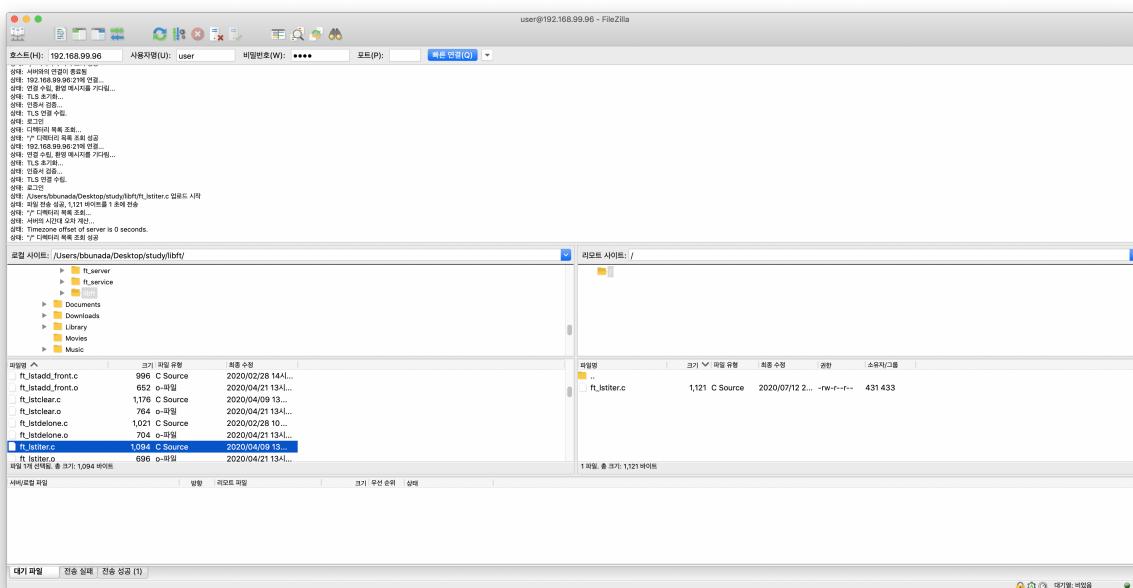
여기까지 잘 따라 오셨다면!!!

파일질라를 통해 연결하려고 하면 인증서를 누르셈!!! 확인 눌러!! 할 겁니다!

그러면 클하게 확인 눌러 주시면 됩니다.



그리고 나서 확인 누르고 파일을 업로드 다운로드해서 잘 되시면 끝!!



- A WordPress website listening on port 5050, which will work with a MySQL database.
Both services have to run in separate containers. The WordPress website will have several users and an administrator.

여기부터는 제 생각엔 우리 레벨에선 굳이..alpine을 기반으로 하나하나 가져다 쓰진 않을 것 같기도하고!!
아직 shell 스크립트도 잘 몰라서 !! 으흠 다 공부하고 하면 좋겠지만!! 요즘 말하는 마이크로 아키텍처엔 맞지

않는 것 같아 적절한 저스트 두희로!!! 지금까지도 그랬지만요!!

ft_service 디렉토리 구조

```
ft_service 현재 디렉토리 구조
├── setup.sh
└── srcts
    ├── clean.sh
    ├── metallb (디렉토리)
    │   └── config.yaml
    ├── nginx (디렉토리)
    │   ├── 10-listen-on-ipv6-by-default.sh
    │   ├── 20-envsubst-on-templates.sh
    │   ├── default.conf
    │   ├── docker-entrypoint.sh
    │   ├── Dockerfile
    │   ├── Makefile
    │   └── nginx.yaml
    └── ftps (디렉토리)
        ├── Dockerfile
        ├── ftps.yaml
        ├── Makefile
        ├── run-vsftpd.sh
        ├── vsftpd.conf
        └── FileZilla
```

이미 앞서 가신 분들의 !!! 깃을 훔쳐서 저스트 두희 하도록 하종!!

아래의 깃에선 워드프레스를 훔치겠습니다!!

https://github.com/solaldunckel/ft_services

그리고 다른 깃에서!! phpmyadmin!!

https://github.com/rchallie/ft_services/blob/master/srcts/containers/phpmyadmin/Dockerfile

여기서 훔치면 됩니다 !!!

- A Grafana platform, listening on port 3000, linked with an InfluxDB database.
Grafana will be monitoring all your containers. You must create one dashboard per service. InfluxDB and grafana will be in two distincts containers.

```
kubectl exec zk-0 -- ps -ef
```

```
kubectl exec zk-0 -- pkill java
```

```
kubectl get pods -w
```

```
ssh nakim@$(minikube ip) -p
```