container healthCheck 처리

아래와 같이 HEALTHCHECK 를 compose 파일에 추가한다.

```
HEALTHCHECK --interval=5s --timeout=2s CMD curl --fail http://localhost || kill 1
```

HEALTHCHECK 를 추가한 경우, 에러서 재기동 될 수 있도록 restart 를 compose 에 추가한다.

```
      healthcheck:
           test: ["CMD-SHELL", "curl --silent --fail localhost:9600 || exit 1"]
        interval: 10s
        timeout: 10s
        retries: 3
      restart: on-failure
```

---

harbor letsencrypt 인증서 갱신

Nginx Proxy Manager 를 통하여 let’s encrypt ssl 인증서를 갱신한다.

해당 인증서의 디렉토리를 확인한다 (예 npm-43)

인증서를 복사한다.

```
cd [harbor설치홈]/data/secret
cp   [proxyManager설치홈]/letsencrypt/live/npm-43/fullchain.pem cert/server.crt
cp    [proxyManager설치홈]/letsencrypt/live/npm-43/privkey.pem cert/server.key
```

harbor 를 재시작한다.

```
docker-compose down
docker-compose up -d
```

Banyazavi T-Sharp 설치

https://www.clien.net/service/board/cm_nas/16430755

---

docker 방화벽 설정

docker 컨테이너의 보안을 위해 iptables 을 설정하는 방안에 대해 기술해 놓고자 한다.

docker는 iptables 에 3개의 체인을 관리한다.

- 종류
    - DOCKER-USER
        - 관리자가 지정하는 룰
    - DOCKER
        - 외부설정 포트에서 내부 아이피의 포트 연동
    - DOCKER-ISOLATION-STAGE-#
        - 잘못된 인터페이스 경우 시, 삭제처리

패킷은 항상 DOCKER, DOCKER-USER 를 통해서 진입한다. (기본적으로 iptables을 수정하는 것을 비추천한다고 함. 자동 설정을 믿고 써라?)

---

docker-compose 프로젝트명 지정

docker-compose 로 실행 시, 기본적으로 디렉토리명이 프로젝트명으로 선언되어 컨테이너가 실행된다.

예를 들어, “디렉토리”_”컨테이너명”_1 과 같은 형식이다.

하지만, `jenkins` 등으로 컨테이너를 올릴 경우 다른 이름으로 프로젝트명이 지정되는 경우가 발생한다. 이럴 경우 명령창에서 수행한 컨테이너와 이름이 틀려 다시 올라가는데, 이때 포트가 충돌하여 안올라가는 경우가 발생한다.

그럼 프로젝트명을 고정할 수 없을까?

환경변수로 `COMPOSE_PROJECT_NAME` 을 지정하면 된다.

예를 들어, `.env` 파일을 만들어 `COMPOSE_PROJECT_NAME=sample` 이라고 기록해 놓으면 실행시, 프로젝트명은 디렉토리명이 아닌 `sample` 이 된다.

`.env` 파일처럼 앞에 점(.)을 붙이기 어려우면, `sample.env` 등과 같이 별도 파일에 기록한 후, `docker-compose.yml` 파일에 환경설정파일을 지정하면 된다.

지정방법 : 각 컨테이너 하위에 아래와 같이 기록

```
env_file : sample.env
```

인식하는 순서 : `docker-compose` 파라메터 `(-p) > .env > sample.env`

---

iRedMail 도커 구축기

- 구축 목적
    - iRedMail 을 도커로 설치해 손쉽게 업데이트 및 관리하기 위함
    - 인증서를 자동갱신하여 적용
- 구축 방법
    - iRedMail 설치
        - 이미지 : `iredmail/mariadb:stable`
        - `docker-compose` 사용
        - 참고글
            - https://techviewleo.com/how-to-run-iredmail-server-in-docker-containers/
    - 인증서 준비
        - `nginxProxyManager` 에서 여러 도메인을 하나의 인증서로 생성
        - 도메인, mail.도메인, smtp.도메인, imap.도메인, pop3.도메인,

webmail.도메인

    - 인증서 목록에서 오른쪽 … 을 누르면 “certificate #번호” 가 보인다. 이 번호를 적어둔다.
  - nginxProxyManager 로 만든 인증서 파일 적용
    - iRedMail 에 볼륨 연결
    - letsencrypt 폴더를 /opt/iredmail/ssl/letsencrypt 로 연결
    - 컨테이너 내부로 이동
      - docker-compose exec iredmail bash
    - 인증서 링크 연결
      - cd /opt/iredmail/ssl
      - ln -s letsencrypt/live/npm-{번호}/fullchain.pem cert.pem
      - ln -s letsencrypt/live/npm-{번호}/fullchain.pem combined.pem
      - ln -s letsencrypt/live/npm-{번호}/privkey.pem key.pem
    - 참고글
      - https://docs.iredmail.org/letsencrypt.html#renew-the-cert-automatically
  - iRedMail 재시작
- iRedMail 사용
  - 웹메일 : https://도메인/mail
  - 관리자 : https://도메인/iredadmin
- iRedMail migration 작업
  - db이동
    - vmail > mailbox 테이블 특정 필드에 (null) 이 아닌 “\n”이 들어가면서 로그인이 안되는 경우가 발생함
  - mailboxes 폴더 이관
- 특이사항
  - iRedMail 재시작 시 postmaster 계정이 생긴다.
    - 보안을 위해서 삭제해도 생긴다.
    - 재시작 시, 삭제한다. (삭제하지 않아도 비밀번호는 환경설정에 있어 1차 보안은 되긴 한다)
      - 관리자에서 삭제해도 사용자 아이디는 남는다. 사용자 아이디에 가서 삭제하면 한번에 삭제 된다.
  - 인증서 갱신 시, iRedMail을 재시작하지 않으면 반영되지 않을 수 있다.

        - 확인 못함, 3개월 후 확인해 봐야 될 듯
    - `iRedMail` 설치 시, 도메인이 인증서와 달리 설치한 경우, 인증서가 적용되지 않을 수 있다.
        - 다시 설치한다.
        - 기존 볼륨 데이터는 백업해 두고, 다시 설치한다.
        - 설치 완료 > 종료 후, 기존 볼륨 데이터로 덮어 쓰고 재시작한다.

`glusterFS` 설치

`glusterFS` 는 노드 볼륨을 상호 동기화 하기 위한 도구

주요 작업 내용

- 각 노드에 `gluster` 서버 기동
- 노드 간 클러스터링 설정
- 동기화 할 볼륨 생성
- 각 노드에 사용할 경로에 볼륨 마운트

1. 노드별 `gluster Server` 생성

- 기존 도커 중지
    - `docker stop gluster-server`
- 도커 실행

```
docker run —rm \
-v glusterfs-lib:/var/lib/glusterd:z \
-v glusterfs-fs:/gluster/fs \
-d —privileged=true —net=host \
—restart on-failure —name gluster-server gluster/gluster-centos
```

2. gluster Server (컨테이너) 작업

한 컨테이너에서 실행

```
docker exec -it $(docker ps -q -f name=gluster) bash
```

노드간 클러스터링 설정

- export replica=클러스터링 노드수
    - export node0=노드1 IP
    - export node1=노드2 IP
- 클러스터링 노드 등록 (접속한 노드는 등록하지 않아도 됨)
    - gluster peer probe ${node0}
    - gluster peer probe ${node1}
- 클러스터링 등록/연결 상태 조회
    - gluster peer status

볼륨 생성

- 볼륨 삭제
    - gluster volume stop gVol
    - gluster volume delete gVol
- 볼륨 생성
    - gluster volume create gVol replica ${replica} transport tcp ${node0}:/gluster/fs ${node1}:/gluster/fs force
- 볼륨 설정
    - gluster volume set gVol network.ping-timeout 1
    - gluster volume set gVol ctime off
- 볼륨 사용
    - gluster volume start gVol
- 상태 조회
    - gluster volume info gVol
    - gluster volume status gVol

3. 노드별 볼륨 마운트

gluster client 설치 (glusterfs 파일시스템 사용을 위함)

- 우분투
    - `apt install glusterfs-client`
- centos
    - `yum install glusterfs-client`

볼륨 마운트

- 마운트 제거
    - `umount ${GVOL_HOME}`
- 마운트 생성
    - `mount -t glusterfs localhost:/gVol /data/test`
- 상태 확인
    - `mount |grep gVol`
- 권한 설정
    - `chmod -R 777 ${GVOL_HOME}`

Docker 관리 : Portainer 설치

Docker 관리 컨테이너 중, 본인이 사용이 편리하다고 판단한 portainer 설치 docker-compose 내용을 공유합니다.

- 시놀로지 폴더에 docker-compose.yml 파일을 아래와 같이 작성
- 제어판 > 작업 스케줄러에서 "생성 > 트리거된 작업 >사용자 정의 스크립트" 를 선택한다.
- 작업명을 쓰고, 작업 설정 > 실행명령을 작성한다.
    - `docker-compose -f [파일경로/파일명] up -d`
    - 파일경로는 시스템의 full 경로 필요
    - 예) /volume1/디렉토리/파일명
- 작업 스케줄러에서 실행하면 잠시 후 실행된다.

- 웹에서 실행 확인
    - http://[NAS서버]:[공유포트]
- 처음 접속 시, 관리자 아이디/패스워드를 생성한다.
- 다음으로 관리할 endpoint를 등록한다.
    - 로컬 docker 를 관리하므로, docker 를 선택한다.

```
# docker-compose.yml 파일
version: '2'

volumes:
  portainer_data:

services:

  portainer:
    # 이미지
    image: portainer/portainer-ce:2.5.0
    restart: always
    ports:
      # 웹접근 포트
      - 39005:9000
    environment:
      # 시간설정
      TZ: Asia/Seoul
    volumes:
      - portainer_data:/data
      - /var/run/docker.sock:/var/run/docker.sock
```

도커 이미지 옮기기

docker hub를 이용하지 않고, docker image를 배포하기 위해서는 기존 docker image를 복제하여 remote 서버에서 사용할 수 있어야 한다. 이를 위해서 docker는 3가지 방법을 제공하고 있는데, 각각의 특징 및 활용방법을 확인해 본다.

[Save]

1. 특징

– 원본 image를 그대로 복제하여 file로 export

2. 활용

– 인터넷 접속이 어려운 상황에서 image를 직접 배포하는 용도

– import, commit과 차이 없음

3. example

```
> docker save ubuntu | gzip > ubuntu-golden.tar.gz
```

```
> zcat ubuntu-golden.tar.gz | docker load
```

[export]

1. 특징

– 현재 구동중인 container의 상태(파일/폴더추가, log 등)를 그대로 복제

– 원본 image와 다른 현재 운영상태를 image로 export

2. 활용

– 현재 운영상태를 그대로 유지하여 container를 확장하는 경우

3. 고려사항

– Dockerfile이 유지되지 않아, 새로운 설정을 추가하기 어려움.

4. example

```
> docker export <containerid> | gzip > mariadb-10-1.tar.gz
```

```
> zcat mariadb-10-1.tar.gz | docker import – mariadb:10.1
```

5. Troubleshooting

– export -> import -> docker run을 실행하면, 아래 에러 발생 “docker: Error response from daemon: No command specified.”

– export는 변경된 filesystem만 export하지, Dockerfile의 ENV, CMD, ENTRYPOINT등을 반영하지 못함.

– 해결

* save 명령어를 사용

* import시에 실행할 명령어를 옵션으로 추가할 수 있으나, save -> load를 이용하여 처리하는 것이 안정적일 것 같다.

[commit]

1. 특징

– 실행중인 container를 image로 저장

– export와 다른 점은 실행결과가 새로운 image를 생성하는 것.

– save/export는 모두 image를 file로 생성.

– 새로운 image를 생성하면서 -c 옵션으로 CMD, ENV등의 명령어를 적용할 수 있음.

2. 활용

– 일반적인 상황에서는 대부분 export를 많이 사용.

– commit은 임시 container에서 작업한 내용을 다른 곳에서 활용하고자 할때, 작업 내용이 사라지지 않도록 image로 저장하는 용도? (추측..)

3. example

```
> docker commit –change “ENV DEBUG true” c3f279d17e0a svendowideit/testimage:version3
```

[출처] [docker] import vs save vs commit|작성자 freepsw

화상회의-jitsi 설치

- 참고사이트
    - https://softmagic.tistory.com/92
- 설치방법
    - 소스 다운로드
        - `sudo git clone https://github.com/jitsi/docker-jitsi-meet && cd docker-jitsi-meet`
    - .env 파일 생성 및 패스워드 셋팅
        - `sudo cp env.example .env`
        - `sudo ./gen-passwords.sh`
    - config 폴더 생성
        - `sudo mkdir -p /data1/jitsi/docker-jitsi-meet/.jitsi-meet-cfg/{web/letsencrypt,transcripts,prosody/config,prosody/prosody-plugins-custom,jicofo,jvb,jigasi,jibri}`
    - SSL 복사
        - `sudo mkdir -p .jitsi-meet-cfg/web/keys/`
        - `sudo cp -p /etc/apache2/ssl/ectech.co.kr_202003113US6.crt.pem .jitsi-meet-cfg/web/keys/cert.crt`
        - `sudo cp -p /etc/apache2/ssl/ectech.co.kr_202003113US6.key.pem .jitsi-meet-cfg/web/keys/cert.key`
    - .env 환경 파일 편집
        - HTTP_PORT=**31080**
        - HTTPS_PORT=**31443**
        - TZ=**Asia/Seoul**
        - PUBLIC_URL=**https://ectech.co.kr:31001**
        - CONFIG=설치위치/docker-jitsi-meet/.jitsi-meet-cfg
        - ENABLE_AUTH=1 # 아무나 방을 만들지 않도록 인증 처리
        - ENABLE_GUESTS=1 # 손님도 개설된 방에 들어올 수 있도록 처리

        - AUTH_TYPE=**internal** # 원하는 인증방법을 등록 (internal : 아래 사용자 등록방법 있음, ldap)
- 방화벽 오픈
    - **10000 UDP** – for general network video/audio communications
    - **3478 UDP** – for quering the stun server (coturn, optional, needs config.js change to enable it)
    - **5347 TCP** – for fallback network video/audio communications over TCP (when UDP is blocked for example), served by coturn
- 실행
    - # 기본
        - **docker-compose up -d**
    - # 문서 공유 및 편집
        - docker-compose -f docker-compose.yml -f etherpad.yml up
    - # jisi 포함
        - docker-compose -f docker-compose.yml -f jigasi.yml -f jibri.yml up
- 접속
    - https://도메인:31443
- 기타
    - 사용자 등록
        - docker exec -it dockerjitsimeet_prosody_1 /bin/bash
        - prosodyctl –config /config/prosody.cfg.lua register 아이디meet.jitsi 비밀번호
    - 설정 변경 파일들..
        - 설치위치 하위
            - .env
            - /.jitsi-meet-cfg/web/
                - config.js
                - interface_config.js
    - 튜닝
        - https://blog.daum.net/sakwon/283
        - https://blog.daum.net/sakwon/294