

virtualbox 安装配置指南

- 安装 virtualbox 时 需要 安装 扩展包 并 安装 补丁。 安装 补丁 需要 安装 补丁 包 并 安装 补丁 包 并 安装 补丁。
- virtualbox 安装 需要 安装 补丁 包 并 安装 补丁。
- synoboot.img 需要 VHD 格式 安装 补丁。
 - 安装, VHD 格式 IMG 格式 安装 补丁 需要 安装 补丁 包 并 安装 补丁 包 并 安装 补丁。
 - synoboot 需要 安装 grub.cfg 需要 安装 MAC 地址 安装 补丁。(OSFMount 需要 安装)
- 安装
 - synoboot 需要 SATA (AHCI) 安装 补丁。
 - 安装 需要 SCSI (LsiLogic) 安装 补丁。
- 安装
 - 安装 需要 安装
 - MAC 地址 synoboot 需要 grub.cfg 并 安装 补丁 安装 补丁。
 - NAT 安装
 - MAC 地址 synoboot 需要 grub.cfg 并 安装 补丁 安装 补丁。
 - synology assistant (find.synology.com) 安装 补丁 安装 补丁, 安装 补丁 安装。



- 安装 需要 安装, 需要 synoboot 需要 安装 补丁 安装 补丁。
 - 安装 需要 安装 补丁 包 并 安装 补丁 包 并 安装 补丁。
 - 安装 URL 并 安装 补丁, 安装 补丁 安装 补丁。 安装 10G 并 安装 补丁。
 - `http://[PC 地址]:5000/web_index.html`
- 安装 需要 安装, 需要 安装 补丁 安装 补丁 安装 补丁。
 - 安装 URL : `http://[PC 地址]:5000`
- 安装 需要 安装 补丁 安装 补丁。
 - 安装, 需要 安装。 (需要 安装 补丁 安装)
 - ~~`msata ssd 安装 补丁(10G) 安装 补丁 安装。 安装 补丁 安装 安装 安装。`~~
 - 安装 需要 安装 补丁 synoboot 安装 补丁
 - ssh 安装 安装, 需要 安装 安装 安装。 ([安装 SSD 安装](#))

000_00000 00)

- 000 > 00 0000 > 00 > 0000 00 00 (000)
 - 00 0000 0)
 - ln /dev/[00001] /dev/synoboot1
 - ln /dev/[00002] /dev/synoboot2
-

DS Audio 00 00 00 00000

[0] <https://blog.acidpop.kr/308>

00, 00 00

DSM > ds audio > 00 > 00 0000 > 00

00 00 0000 0.6

https://blog.kakaocdn.net/dn/ASGtY/btqz4Kcjt0U/lxFGKo1s3j92kK2pcnyvaK/alsong_lyric_0.6.aum?attach=1&knm=tfile.aum

00 00 0000

https://github-releases.githubusercontent.com/203561885/715efe80-3661-11ea-9edf-e52c025dfe24?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20210609%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20210609T083249Z&X-Amz-Expires=300&X-Amz-Signature=e7dc3888ea45fef1a4b8c2e6a2e08ae85dcbe4bd2219d4b9f7e26d920f10cb0a&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=203561885&response-content-disposition=attachment%3B%20filename%3Dfujirou_musix_match-1.3.aum&response-content-type=application%2Foctet-stream

SSD 파티션 설정

[링크] http://m.ppomppu.co.kr/new/bbs_view.php?id=nas&no=40068

SSD 파티션 설정, 파티션 설정 방법

- SSH 접속
- 파티션 설정 (파티션 설정, 파티션 설정 sdb4, sdb5 설정)



fdisk -l

- synoboot 파티션 설정



link 설정

- 파티션 “파티션 1 파티션” 파티션 “DSM 파티션” 파티션 파티션 파티션 파티션/파티션 파티션.
 - 파티션 설정 방법
 - “파티션 1 파티션” 파티션, “파티션 2 파티션” 파티션
 - 파티션 파티션 파티션 파티션
 - 파티션 파티션 파티션 파티션 파티션 파티션
 - DSM 파티션 PAT 파티션 파티션 (DS 918 plus -> 918+ 파티션 파티션)
 - 파티션 파티션(100MB 파티션) 파티션 파티션 파티션 파티션, 파티션 파티션 파티션
 - 파티션 파티션, 파티션 파티션 파티션. 파티션 파티션 파티션 파티션 파티션 파티션 (파티션 파티션)
 - 파티션
 - 파티션 파티션, synoboot 파티션 파티션 파티션 파티션 파티션
-

COVID-19 的 免疫

免疫

- 免疫系统 是 身体 防御 病原体 入侵 的 重要 屏障。 免疫系统 ‘识别’ 病原体 并 启动 免疫 反应。 免疫 反应 包括 产生 抗体 和 激活 免疫 细胞。
- **B** 细胞 是 免疫 细胞。 B 细胞 产生 抗体 来 中和 病原体 并 标记 它们 以便 被 其他 免疫 细胞 清除。
- **T** 细胞 是 免疫 细胞， 它们 帮助 协调 免疫 反应。

免疫

- **mRNA** 疫苗 是 一种 新型 疫苗， 它 包含 编码 新冠病毒 表面 蛋白 的 mRNA。 COVID-19 疫苗 通过 训练 免疫系统 来 识别 和 中和 病毒。 疫苗 不会 导致 疾病， 但 可以 产生 免疫 反应。 疫苗 是 一种 安全 有效 的 预防 COVID-19 的 方法。 T 细胞 和 B 细胞 是 免疫 细胞。
 - 疫苗(口服/注射)， 疫苗(口服)
- 疫苗 是 一种 预防 COVID-19 的 方法。 疫苗 通过 训练 免疫系统 来 识别 和 中和 病毒。 疫苗 不会 导致 疾病， 但 可以 产生 免疫 反应。 疫苗 是 一种 安全 有效 的 预防 COVID-19 的 方法。 T 细胞 和 B 细胞 是 免疫 细胞。
 - 疫苗(口服/注射)， 疫苗(口服)
- **疫苗** 是 一种 预防 COVID-19 的 方法。 疫苗 通过 训练 免疫系统 来 识别 和 中和 病毒。 疫苗 不会 导致 疾病， 但 可以 产生 免疫 反应。 疫苗 是 一种 安全 有效 的 预防 COVID-19 的 方法。 T 细胞 和 B 细胞 是 免疫 细胞。
 - 疫苗(口服/注射)， 疫苗(口服)
- **疫苗**
 - 疫苗 : 疫苗 名称 / 剂型
 - 疫苗 是 一种 预防 COVID-19 的 方法。 疫苗 通过 训练 免疫系统 来 识别 和 中和 病毒。 疫苗 不会 导致 疾病， 但 可以 产生 免疫 反应。 疫苗 是 一种 安全 有效 的 预防 COVID-19 的 方法。 T 细胞 和 B 细胞 是 免疫 细胞。
 - 疫苗 : 疫苗 名称 / 剂型

疫苗

- 20 00: 20 000 000 COVID-19 000 00 00, 20 00 0 200 000 00 00 000 000 00000.
- 000-000000
- 000
- 00000000
- 000
- 0000
- 10 00: 10 000 000 COVID-19 000 00 00 00 0 200 000 00 00 000 000 00000.
- 000000_00



webclient 00 0 reactive 0000

- 000 00
 - doOnError() : 000 0000 00, 00 000 0000 00 00
 - 00 0, 00 0 00000 00000 00000 00 00
 - 2xx 0 00 000 Exception 0 00 call 0
 - doOnSuccess() : 00 00 0, 000
 - onErrorReturn : 000 0000 0 00 00 Return 0
 - 000 00 000 0000 0000 0000 00 0 00
 - onErrorResume : 000 0000 0 00 Flux0000 Return 0
 - 000 00 000 0000 0000 0000 00 0 00
 - onErrorContinue : 000 0000 0 000 00 00 000 skip00 000.
 - 0000 0 00 0 00 0 00 0 00

```
public Mono<Object> doExec(
    ServerHttpRequest request, ServerHttpResponse
    clientResponse, RbcAuth rbcAuth) {
```

```
    String finalAuthToken = ...;
```

```

    try{
        return webClient.get()
            .uri(rbcUrl.getCategories())
            .accept(MediaType.APPLICATION_JSON)
            .headers(httpHeaders ->
httpHeaders.setBearerAuth(finalAuthToken))
            .retrieve()
            .onStatus(HttpStatus::isError,
res -> doError(res))
            .bodyToMono(CustomResult.class)
            .doOnError(ex ->
doExcept(clientResponse, ex))
            .flatMap(res ->
doSucc(clientResponse, res))
            .doOnSuccess(res ->
doFinish(clientResponse, res))
            .subscribe()
            .log()
            ;
    }catch (Exception e){
        log.error("webClient fail. e={}",
e.getMessage(), e);
        return doExcept(clientResponse, e);
    }
}

private Mono<Object> doFinish(ServerHttpResponse response,
CustomResult res) {
    ...
    return Mono.just("finish");
}

private Mono<Object> doSucc(ServerHttpResponse response,
CustomResult res) {
    ...
    return Mono.just("success");
}

private Mono<Object> doExcept(ServerHttpResponse response,
Throwable ex) {
    ...
}

```

```

    return Mono.just("fail. e={}", e.getMessage());
}

private Mono<Exception> doError(ClientResponse res) {
    if (res.statusCode().isError()) {
        return Mono.error(new
RuntimeException(res.toString()));
    }
    else{
        return Mono.error(new
RuntimeException(res.toString()));
    }
}

```

□□□ □□□ - □□□□□ □ □□□ □□ **11**□□

1. 9□ 1□□ 9□□ □□□. → □□□ □□ □□ □□ □□□□□□.
2. □□□ □□□, □□□□□ □□□. → □□□□□□ □□□□ □□□ □□□□ □□□□.
3. □□□ □□□ □□□□ □□□ □□□ □□ □□□ □□□.
4. □□□ □□ □□□ □□ □□□□□. → !@#\$@#\$\$%\$
5. □□□□ □□□ □□□, □□□□□ □□□□ □□□ □□□ □□□.
6. □□ □□□ □□□ □□ □□ □□□ □□ □□□. → □□ □□□□ □□□□ □□□ □□□□□□. □□□□, □□□□ □, □□ □
7. □□□ □□□ □□□ □□. → □ □□ □□□ □□□□, □□ □□ □□□□ □□□□. □ □□ □□ □□ □□□ □□□, □□ □□ □□ □□ □□ □□ □□□□□ □□ - □□□
8. □□ □□□ □□ □□, □□, □□□□□□, □□□□□, □□□□□□□ □□□□.
9. □□ □□ □□□□ □□ □□□ □□. → □ □□ □□ □□ □□□ □□ □□□ □□□. □□, □□, □□, □□□□ □□□□, CS, □□□□ □
10. □□□ □□□ □□□ □□ □□□ □□□ □□. → □□□ □□ □□□ □□□□ □□□ □□□ □ □ □□□ □□ □ □□□.
11. □□□ □□ □□□ □□ □□ □□ □□□ □□ □□. → □□□□ □□□□ □□□□~~~~ □□□□~~~~

Docker 실행 : Portainer 실행

Docker 실행 방법을, 실행 방법과 관련된 내용 portainer 실행 docker-compose 실행 방법을.

- 실행 방법 docker-compose.yml 실행 방법 실행 방법
- 실행 > 실행 방법 “실행 > 실행 방법 실행 방법 실행 방법” 실행 방법.
- 실행 방법 실행, 실행 실행 > 실행 방법 실행 방법.
 - docker-compose -f [실행 방법/실행 방법] up -d
 - 실행 방법 실행 방법 full 실행 방법 실행 방법
 - 실행 방법) /volume1/실행 방법/실행 방법
- 실행 방법 실행 방법 실행 방법 실행 방법 실행 방법.
- 실행 방법 실행 방법 실행 방법
 - http://[NAS실행 방법]:[실행 방법]
- 실행 방법 실행 방법 실행 방법 실행 방법/실행 방법 실행 방법.
- 실행 방법 실행 방법 endpoint 실행 방법 실행 방법.
 - 실행 방법 docker 실행 방법 실행 방법, docker 실행 방법 실행 방법.

```
# docker-compose.yml 실행 방법
```

```
version: '2'
```

```
volumes:
```

```
  portainer_data:
```

```
services:
```

```
  portainer:
```

```
    # 실행 방법
```

```
    image: portainer/portainer-ce:2.5.0
```

```
    restart: always
```

```
    ports:
```

```
      # 실행 방법 실행 방법
```

```
      - 39005:9000
```

```
    environment:
```

```
      # 실행 방법
```

```
      TZ: Asia/Seoul
```

volumes:

- portainer_data:/data
- /var/run/docker.sock:/var/run/docker.sock



[[[[[

docker hub에 올리는 것, docker image에 올리는 것 docker image에 올리는 remote에 올리는 것 이 것들 이다. 이 것들 docker에 3개 이다 올리는 것들, 이 것 이 이 것들을 이 것 이다.

[Save]

1. [[

- [[image에 올리는 것들 file에 export

2. [[

- [[[[[[[[image에 올리는 것들 이다

- import, commit에 올리는 것

3. example

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

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

[export]

1. [[

- [[[[[[container에 올리는 것([[/ [[, log)에 올리는 것

- [[image에 올리는 것 이다 [[[[image에 export

2. [[

- container를 export하는 방법

3. export

- Dockerfile를 사용하여 container를 export하는 방법.

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 시도할 때, "docker: Error response from daemon: No command specified."라는 에러가 발생할 수 있다.

- export는 filesystem을 export할 때, Dockerfile에 ENV, CMD, ENTRYPOINT를 지정할 수 있다.

-

* save 명령어

* import는 Dockerfile을 사용하여 container를 import할 수 있다. save -> load를 사용하여 container를 저장하고 불러올 수 있다.

[commit]

1. commit

- container를 image로 commit하는 방법

- export 명령어를 사용하여 container를 image로 export하는 방법.

- save/export 명령어를 사용하여 image를 file로 저장하는 방법.

- image를 commit할 때 -c 옵션을 사용하여 CMD, ENV 등을 지정할 수 있다.

2. push

- image를 Docker Hub에 push하는 방법

- commit된 container를 Docker Hub에 push하는 방법, Dockerfile을 사용하여 image를 생성하고 push하는 방법.

image를 어떻게 관리? (Docker..)

3. example

```
> docker commit -change "ENV DEBUG true" c3f279d17e0a  
svendowideit/testimage:version3
```

[링크] [\[docker\] import vs save vs commit](#) | 링크 [freepsw](#)

Docker-jitsi 설치

- Docker
- <https://softmagic.tistory.com/92>
- Docker
- Docker 설치
 - sudo git clone <https://github.com/jitsi/docker-jitsi-meet> && cd docker-jitsi-meet
- .env 파일 생성
 - sudo cp env.example .env
 - sudo ./gen-passwords.sh
- config 디렉토리 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 키 생성
 - sudo mkdir -p .jitsi-meet-cfg/web/keys/
 - sudo cp /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.k  
ey.pem .jitsi-meet-cfg/web/keys/cert.key
```

- .env
▪ HTTP_PORT=**31080**
▪ HTTPS_PORT=**31443**
▪ TZ=**Asia/Seoul**
▪ PUBLIC_URL=<https://ectech.co.kr:31001>
▪ CONFIG=`docker-jitsi-meet/.jitsi-meet-cfg`
▪ ENABLE_AUTH=1 #
▪ ENABLE_GUESTS=1 #
▪ AUTH_TYPE=**internal** # (internal : , ldap)
- ▪ **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
- ▪ #
▪ **docker-compose up -d**
▪ #
▪ `docker-compose -f docker-compose.yml -f etherpad.yml up`
▪ # jisi
▪ `docker-compose -f docker-compose.yml -f jigasi.yml -f jibri.yml up`
- ▪ `https://:31443`
- ▪
▪ `docker exec -it dockerjitsimeet_prosody_1 /bin/bash`
▪ `prosodyctl -config /config/prosody.cfg.lua`

register meet.jitsi

- `meet.jitsi`
 - `.env`
 - `/.jitsi-meet-cfg/web/`
 - `config.js`
 - `interface_config.js`
- <https://blog.daum.net/sakwon/283>
- <https://blog.daum.net/sakwon/294>