



# Docker\_backup

최초 작성일: 24.02.20

최종 작성일: 24.02.21

## 1. docker 컨테이너를 이미지로 압축해서 scp로 주고받기

### 1) host(mk) → host(mk)

서로 ssh접근되는지 확인

```
ssh {whoami로 나오는 이름}@{ip주소}
```

'scpfile'이라는 컨테이너 생성

```
docker exec -it scpfile /bin/bash
```

생성한 컨테이너를 tar 이미지로 만들기

```
docker save -o /scpfileimage.tar b69a5adb9126
```

- **-o** : 경로 지정
- **scpfileimage.tar** : {파일 이름}.{압축 확장자}

scp로 보내기(host pc → host pc)

- **b69a5adb9126** : 이미지 ID



## tar vs archive

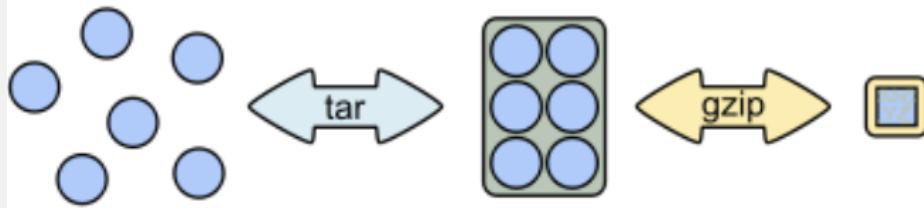
- window vs linux
    - window
      - 선택된 파일이나 폴더들을 묶으면서 동시에 압축( compress )
      - 압축파일이름.zip 파일 한 개로 만들면서 압축
      - 확장자: tar (tape archives)
    - linux
      - 압축하거나 관리할 파일들을 먼저 하나의파일로 묶음(아카이빙)
- 이렇게 묶인 파일을 따로 추가로 압축을 해서 용량을 줄임
- gzip 사용해 압축
  - 확장자: .tar.gz

### 💡 Tip

아카이빙 Archiving : 단일 파일에 파일과 디렉터리들을 넣어 묶기

압축 Compression : 파일이나 아카이브가 디스크에서 차지하는 공간을 줄여주는 소프트웨어 도구 (알고리즘을 통해)





<https://inpa.tistory.com/entry/LINUX-📁-아카이브-압축-명령어-정리-tar-compress-gzip-bzip2-zip>

**scp scpfileimage.tar jd@192.168.137.1:/**

⇒ 하지만 denied 뜸

```
root@ubuntu:/# scp scpfileimage.tar jd@192.168.137.1:/
The authenticity of host '192.168.137.1 (192.168.137.1)' can't be established.
ED25519 key fingerprint is SHA256:3o0hnM1g9RANtstrL7HkRDXn03QVGXFlKAacXvY0g1o.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.137.1' (ED25519) to the list of known hosts.
jd@192.168.137.1's password:
scp: /scpfileimage.tar: Permission denied
```

- 이는 민경 → 지원의 루트로 접근 권한이 없어서 뜨는 오류

**scp scpfileimage.tar jd@192.168.137.1:/home/jd** 명령어로 해주기

```

root@ubuntu:/# scp scpfileimage.tar jd@192.168.137.1:/home/jd
jd@192.168.137.1's password:
scpfileimage.tar
root@ubuntu:/# ssh jd@192.168.137.1
jd@192.168.137.1's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-94-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Tue Feb 20 05:24:20 AM UTC 2024

System load:  0.044921875      Processes:           155
Usage of /:   50.5% of 18.53GB Users logged in:     1
Memory usage: 22%             IPv4 address for docker0: 172.17.0.1
Swap usage:   0%              IPv4 address for enp0s3:  10.0.2.15

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

```

지원pc로 들어온 것 확인(결과)

```

jd@docker:~$ ls
apahce.tar  scpfileimage.tar
jd@docker:~$ █

```

## 2. public hub로 이미지 주고받기

scp로 파일을 계속 주고받기에는 한계가 있음

따라서 public hub로 주고받는 실습을 진행함

전제: 이미 만들어놓은 estrellasia/web 이미지 활용함

**docker image tag estrellasia/web:nginx1 estrellasia/web:1.0**

```

root@ubuntu:~# docker image tag estrellasia/web:nginx1 estrellasia/web:1.0
root@ubuntu:~# docker images
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
scpfileimage        latest       b69a5adb9126   2 hours ago   185MB
estrellasia/web     1.0         37f28cce3eef   5 hours ago   187MB
estrellasia/web     nginx1      37f28cce3eef   5 hours ago   187MB
estrella/web        nginx1      3fc2b5a4e075   5 hours ago   187MB
nginx               latest       e4720093a3c1   5 days ago    187MB
jenkins/jenkins     latest       db75aa81ca2b   6 days ago    475MB
ubuntu              latest       3db8720ecbf5   6 days ago    77.9MB

```

**docker login** //docker hub의 계정 로그인하기

**docker push estrellasia/web:1.0**

```

root@ubuntu:~# docker push estrellasia/web:1.0
The push refers to repository [docker.io/estrellasia/web]
828cbd7d639b: Layer already exists
61a7fb4dabcd: Layer already exists
bcc6856722b7: Layer already exists
188d128a188c: Layer already exists
7d52a4114c36: Layer already exists
3137f8f0c641: Layer already exists
84619992a45b: Layer already exists
ceb365432eec: Layer already exists
1.0: digest: sha256:6dbde946d403bfff35e301110f2c675c63e95380a6fa9a5d19422ca92263eef76 size: 1985
root@ubuntu:~#

```

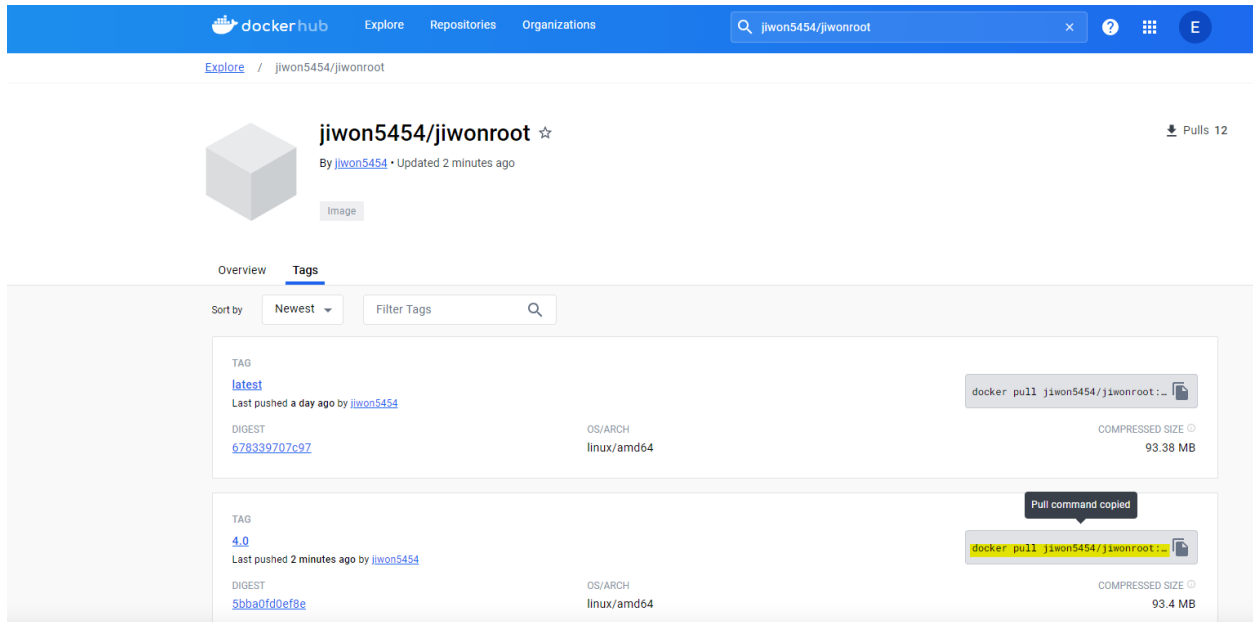
public hub에 올라간 것 확인(결과 확인)

## Tags

This repository contains 2 tag(s).

Tag	OS	Type	Pulled	Pushed
 1.0		Image	5 hours ago	a few seconds ago

지원님것 pull하기



```
root@ubuntu:~# docker pull jiwon5454/jiwonroot:4.0
4.0: Pulling from jiwon5454/jiwonroot
d66d6a6a3687: Pull complete
2e8c8f9af7ba: Pull complete
4f4fb700ef54: Pull complete
46c46c531ed6: Pull complete
8a93d5693d36: Pull complete
Digest: sha256:5bba0fd0ef8ef68f476fa78b817cad9ba64b13ea165d8fa0c2f681947f82ec24
Status: Downloaded newer image for jiwon5454/jiwonroot:4.0
docker.io/jiwon5454/jiwonroot:4.0
```

결과확인

```
root@ubuntu:~# docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
scpfimage            latest      b69a5adb9126  2 hours ago   185MB
estrellasias/web     1.0        37f28cce3eef  5 hours ago   187MB
estrellasias/web     nginx1     37f28cce3eef  5 hours ago   187MB
estrella/web         nginx1     3fc2b5a4e075  5 hours ago   187MB
jiwon5454/jiwonroot  4.0        ad21482b211e  23 hours ago  234MB
```

### 3. private hub로 컨테이너 이미지 올려서 주고받기

public hub로 주고받을 경우 보안 상의 문제가 발생함

따라서 private hub로 컨테이너 이미지를 올리고 주고받는 실습을 진행함



## private hub

- 개인적으로만 사용하도록 개인 서버에 구축하는 도커 이미지 저장소
- 회사 사내에서 사용하거나 개인적으로 만든 이미지를 올리고 관리하기 위해서 사용됨
- Private registry를 구축하기 위해서는 도커에서 제공하는 "Registry" 라는 이미지를 사용함
  - Registry도 도커 컨테이너이므로, 여기 올리는 이미지들을 계속 유지하려면 볼륨을 다른 외부 스토리지 등 안정성있는 스토리지로 사용해야 함

<https://watch-n-learn.tistory.com/43>

virtualbox에서 ubuntu server에 hdd 추가하기

가상 하드 디스크 만들기



### 가상 하드 디스크 파일 형식

새 가상 하드 디스크 파일 형식을 선택하십시오. 다른 가상화 소프트웨어에서 디스크를 사용하지 않으려 변경하지 않아도 됩니다.

- ☒ VDI (VirtualBox 디스크 이미지)
- ☐ VHD (가상 하드 디스크)
- ☐ VMDK (가상 머신 디스크)



## VDI vs VHD vs VMDKS

각각 다른 가상 디스크 파일 형식을 나타냄

- **VDI (VirtualBox Disk Image)**

- Oracle VirtualBox에서 사용되는 디스크 이미지 형식
- 단일 파일로 가상 디스크의 모든 데이터를 저장함
- 일반적으로 `.vdi` 확장자를 가짐
- VirtualBox에서 생성하고 사용됨

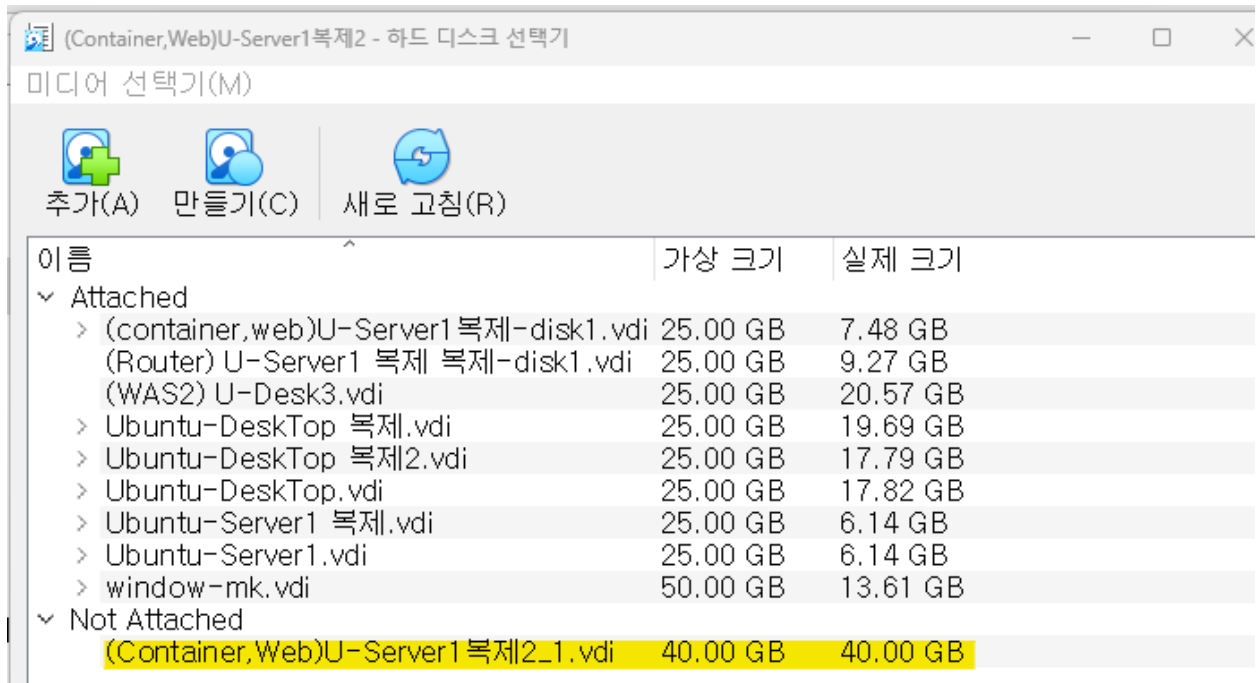
- **VHD (Virtual Hard Disk)**

- Microsoft Hyper-V 및 Virtual PC와 같은 가상화 플랫폼에서 사용되는 디스크 이미지 형식
- Windows 가상 머신에 사용됨
- `.vhd` 확장자를 가짐
- Hyper-V나 Disk Management 같은 도구를 사용하여 생성할 수 있음

- **VMDK (Virtual Machine Disk)**

- VMware 가상화 소프트웨어에서 사용되는 디스크 이미지 형식
- 가장 널리 사용되는 가상 디스크 형식 중 하나임
- `.vmdk` 확장자를 가짐
- VMware Workstation, VMware Fusion, VMware ESXi 등에서 생성하고 사용됨





**fdisk -l** //시스템 내부적으로 확인

**fdisk /dev/sdb** //파티션 설정

- 파티션 설정



#### 하드디스크 파티션

- 하드디스크를 논리적으로 나눈 구역
  - 하나의 디스크를 여러 개의 파티션으로 나눌 수 있고, 이를 각각의 드라이브로 인식할 수 있음
  - ex) 500G 하드디스크를 250GB, 250GB로 나뉘어 흔히 C드라이브, D드라이브로 생성해서

- 그 다음 설정하기

<https://noldit.tistory.com/entry/virtualbox-버추얼박스-우분투에-하드-추가>

- m 입력하기

- m을 입력하면 도와준다고 하니 m을 입력하기

```
Command (m for help): m
Help:

DOS (MBR)
a  toggle a bootable flag
b  edit nested BSD disklabel
c  toggle the dos compatibility flag

Generic
d  delete a partition
F  list free unpartitioned space
l  list known partition types
n  add a new partition
p  print the partition table
t  change a partition type
v  verify the partition table
i  print information about a partition

Misc
m  print this menu
u  change display/entry units
x  extra functionality (experts only)

Script
I  load disk layout from sfdisk script file
O  dump disk layout to sfdisk script file

Save & Exit
w  write table to disk and exit
q  quit without saving changes

Create a new label
g  create a new empty GPT partition table
G  create a new empty SGI (IRIX) partition table
o  create a new empty DOS partition table
s  create a new empty Sun partition table
```

- n 입력하기

- 우리가 할 것은 파티션을 나누는 것이니 n 입력

Command (m for help): n

- p입력하기 //파티션 타입 설정하기

```
Partition type
  p  primary (0 primary, 0 extended, 4 free)
  e  extended (container for logical partitions)
```

- 1 입력하기 //파티션 number 설정하기

- 파티션 1의 저장공간으로 어디서부터 어디까지 사용할 것인지 결정

- 2048 입력하기 //시작 섹터 선택하기

First sector (2048-104857599, default 2048): 2048

- 20973568 //마지막 섹터 선택하기

```
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-52428799, default 52428799): 20973568
Created a new partition 1 of type 'Linux' and of size 10 GiB.
```

- 이렇게 하면 파티션 1의 크기를 10GiB로 잡는 것

### mkfs.ext4 /dev/sdb1

- mkfs.ext4 /dev/{새로 생성된 hdd}
- /dev/sdb 디스크의 첫 번째 파티션에 ext4 파일 시스템을 생성하기



#### ext4 파일

- 리눅스에서 사용되는 파일 시스템 중 하나
- 디스크 파티션을 포맷하고 파일 시스템을 생성할 때 사용됨
- 파일 시스템은 파일과 디렉터리를 저장하는 방법을 결정함

cd /

ls

mkdir /newdisk1 //디스크 폴더 생성하기

- mkdir /{폴더명 마음대로}

ls

mount /dev/sdb1 /newdisk1 //마운트하기

- /dev/sdb1 디스크 파티션을 /newdisk1 디렉터리에 마운트하는 명령어
- 지정된 디스크 파티션에 대해 파일 시스템을 마운트하여 해당 디렉터리에 파일 및 디렉터리에 액세스할 수 있도록 함

- 디스크 파티션을 마운트하면 해당 디렉터리에 파일 시스템이 연결되어 해당 파티션에 있는 파일과 디렉터리를 볼 수 있게 됨

**vim /etc/fstab** //전원 on시, 자동 마운트 설정하기

- 아래처럼 설정하기

/dev/(새로 생성된 하드디스크)	/(마운트될 디렉터리)	(파일시스템)	(속성)	(dump 사용여부)	(파일시스템 체크여부)
/dev/sdb	/newdisk	ext4	defaults	0	0



#### 설정 설명

- **/dev/sdb1** : 마운트할 디스크 파티션
- **/newdisk1** : 마운트할 디렉터리
- **ext4** : 파일 시스템 유형
- **defaults** : 기본 마운트 옵션
- **0** : 백업 시스템을 사용하지 않음
- **0** : 부팅 시 자동으로 마운트하는 순서

**sudo reboot**

**fdisk -l** //시스템 내부적으로 확인

```

Disk /dev/sdb: 40 GiB, 42949672960 bytes, 83886080 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x4884ba13

Device            Boot Start      End  Sectors  Size Id Type
/dev/sdb1          2048 83886079 83884032   40G 83 Linux
  
```

<https://jihyeong-jj99hy99.tistory.com/210> (하드디스크 생성과 마운트)

우리가 만든 파티션-docker를 연결할 것임

## docker info

```
quick connect... 2. 192.168.76.3
Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local splunk syslog
Swarm: inactive
Runtimes: io.containerd.runc.v2 runc
Default Runtime: runc
Init Binary: docker-init
containerd version: ae07eda36dd25f8a1b98dfbf587313b99c0190bb
runc version: v1.1.12-0-g51d5e94
init version: de40ad0
Security Options:
  apparmor
  seccomp
   Profile: builtin
  cgroupns
Kernel Version: 5.15.0-91-generic
Operating System: Ubuntu 22.04.3 LTS
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 1.918GiB
Name: ubuntu
ID: fca95661-dd93-4543-b760-8e60883b9ec1
Docker Root Dir: /var/lib/docker
Debug Mode: false
Username: estrellasia
Experimental: false
Insecure Registries:
  127.0.0.0/8
Live Restore Enabled: false
```

- /var/lib/docker
    - 우분트 이미지의 위치(컨테이너 이미지 데이터뿐만 아니라 컨테이너와 관련된 다양한 정보도 포함됨)
    - image등을 pull 하면 기본적으로 저장되는 위치
- ⇒ 이 공간이 부족할 것을 대비하여 저장공간이 풍부한 외부 디바이스로 재지정할 것임

**systemctl stop docker** //도커 다운하기

**systemctl status docker**

**df -h** //마운트할 저장 장치의 이름 확인하기

```
root@ubuntu:~# df -h
Filesystem                Size      Used Avail Use% Mounted on
tmpfs                     197M        1.2M    196M   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv 12G       9.2G     1.5G  86% /
tmpfs                     983M          0    983M   0% /dev/shm
tmpfs                     5.0M          0     5.0M   0% /run/lock
/dev/sdb1                  40G         24K     38G   1% /newdisk1
/dev/sda2                  2.0G       129M     1.7G   8% /boot
tmpfs                     197M        4.0K    197M   1% /run/user/1000
root@ubuntu:~#
```

- 빨간색 부분을 바꿔줄거임

**vim /etc/docker/daemon.json** //도커의 기본 저장 경로 바꾸기

```
{
    "data-root": "/dockerImage",
    "insecure-registries":["192.168.137.52:5000"]
}
```

- json 형식임 → key-value 형식으로 넣기
- data-root를 새롭게 마운트한 /dockerImage하고 연결한다는 의미



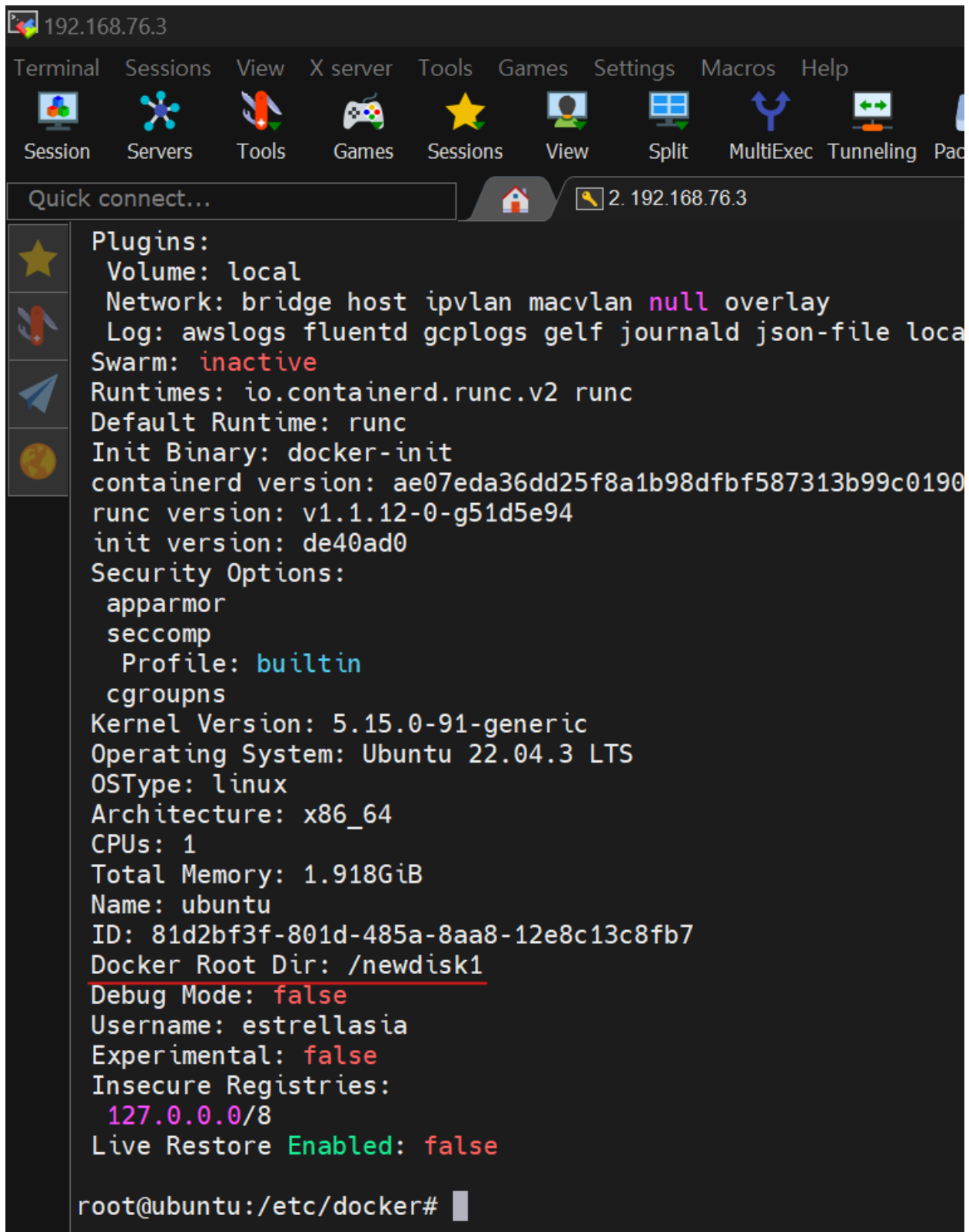
### **insecure-registries 옵션**

- **insecure-registries** 옵션은 Docker가 신뢰할 수 없는 (보안되지 않은) 도커 레지스트리에 대한 접근을 허용하는지를 설정
- Docker는 보안을 강화하기 위해 TLS(전송 계층 보안)를 사용하여 도커 레지스트리와 통신함
  - but 가끔은 보안되지 않은 도커 레지스트리에 연결해야 하는 경우가 있음
  - **insecure-registries** 옵션을 사용하여 Docker에게 보안이 적용되지 않은 레지스트리에 대한 접근을 허용
- **192.168.137.52:5000** 주소에 있는 도커 레지스트리에 대한 접근을 보안 없이 허용함.
  - 이렇게 함으로써 Docker 클라이언트는 TLS를 사용하지 않고도 해당 레지스트리에 이미지를 푸시하거나 풀할 수 있

**cd /etc/docker**

**systemctl restart docker**

**docker info**

The screenshot shows the Docker Desktop application window. The title bar indicates the IP address 192.168.76.3. The menu bar includes Terminal, Sessions, View, X server, Tools, Games, Settings, Macros, and Help. Below the menu bar is a toolbar with icons for Session, Servers, Tools, Games, Sessions, View, Split, MultiExec, Tunneling, and Pack. A 'Quick connect...' search bar is present, showing a home icon and the address 2. 192.168.76.3. The main content area displays system information for a Docker container. On the left, there are four icons: a star, a rocket, a paper plane, and a globe. The text in the main area is as follows:

```
Plugins:  
Volume: local  
Network: bridge host ipvlan macvlan null overlay  
Log: awslogs fluentd gcplogs gelf journald json-file local  
Swarm: inactive  
Runtimes: io.containerd.runc.v2 runc  
Default Runtime: runc  
Init Binary: docker-init  
containerd version: ae07eda36dd25f8a1b98dfbf587313b99c0190  
runc version: v1.1.12-0-g51d5e94  
init version: de40ad0  
Security Options:  
apparmor  
seccomp  
Profile: builtin  
cgroups  
Kernel Version: 5.15.0-91-generic  
Operating System: Ubuntu 22.04.3 LTS  
OSType: linux  
Architecture: x86_64  
CPUs: 1  
Total Memory: 1.918GiB  
Name: ubuntu  
ID: 81d2bf3f-801d-485a-8aa8-12e8c13c8fb7  
Docker Root Dir: /newdisk1  
Debug Mode: false  
Username: estrellasia  
Experimental: false  
Insecure Registries:  
127.0.0.0/8  
Live Restore Enabled: false  
  
root@ubuntu:/etc/docker#
```

- /var/lib/docker → /newdisk1으로 바뀐 것을 알 수 있음



<https://pajamacoder.tistory.com/29> (docker root directory 변경)

Private registry 구현할 것임

아래의 2개 이미지 다운받

**docker pull registry**

**docker pull hyper/docker-registry-web**

**docker run -d -it --name registry1 -p 5000:5000 registry**

포트 포워딩 규칙					
이름	프로토콜	호스트 IP	호스트 포트	게스트 IP	게스트 포트
1	UDP	192.168.137.1	53	10.0.2.15	54
Rule 3	TCP	192.168.137.1	80	10.0.2.15	81
Rule 4	TCP	192.168.137.1	22	10.0.2.15	22
Rule 5	TCP	192.168.137.52	22	10.0.2.15	22
Rule 6	TCP	192.168.137.52	5000	10.0.2.15	5000

**docker pull estrellasia/web:1.0**

**docker tag estrellasia/web:1.0 localhost:5000/estrellasia:2.0**



#### 명령어 해석

**estrellasia/web:1.0** 이미지를 **localhost:5000/estrellasia:2.0**으로 태그하겠다는 의미

- **estrellasia/web:1.0** : 태그를 지정할 원래 이미지 이름과 태그
- **localhost:5000/estrellasia:2.0** : 새로운 태그가 지정될 이미지의 이름과 태그

**docker push localhost:5000/estrellasia:2.0**

```

root@ubuntu:~# docker push localhost:5000/estrellasia:2.0
The push refers to repository [localhost:5000/estrellasia]
828cbd7d639b: Pushed
61a7fb4dabcd: Pushed
bcc6856722b7: Pushed
188d128a188c: Pushed
7d52a4114c36: Pushed
3137f8f0c641: Pushed
84619992a45b: Pushed
ceb365432eec: Pushed
2.0: digest: sha256:6dbde946d403bff35e301110f2c675c63e95380a6fa9a5d19422ca92263eef76 size:

```

**docker exec -it registry1 sh**

결과 확인하기

```

/var/lib/registry/docker/registry/v2/repositories # ls
estrellasia

```

지원님이 내 private registry로 보내준 이미지를 다운받을 것

**docker pull 127.0.0.1:5000/jiwonweb:4.0**

```

root@ubuntu:~# docker pull 127.0.0.1:5000/jiwonweb:4.0
4.0: Pulling from jiwonweb
d66d6a6a3687: Pull complete
2e8c8f9af7ba: Pull complete
4f4fb700ef54: Pull complete
46c46c531ed6: Pull complete
8a93d5693d36: Pull complete
Digest: sha256:5bba0fd0ef8ef68f476fa78b817cad9ba64b13ea165d8fa0c2f681947f82ec24
Status: Downloaded newer image for 127.0.0.1:5000/jiwonweb:4.0
127.0.0.1:5000/jiwonweb:4.0

```

- docker images로 확인하기(결과) - 다운받아

```

root@ubuntu:~# docker images

```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
estrellasia/web	1.0	37f28cce3eef	7 hours ago	187MB
localhost:5000/estrellasia	2.0	37f28cce3eef	7 hours ago	187MB
127.0.0.1:5000/jiwonweb	4.0	ad21482b211e	25 hours ago	234MB
registry	latest	a8781fe3b7a2	3 weeks ago	25.4MB
httpd	latest	2776f4da9d55	4 weeks ago	167MB
hyper/docker-registry-web	latest	0db5683824d8	7 years ago	599MB

