[Docker] ARM64_환경 설정 가이드

Instruction

1. Open the terminal and go to the directory where you want to perform your tasks

2. Create a file named 'ubuntu-lts.yaml' with the following content

```
arch: "aarch64"
images:
- location: "https://cloud-images.ubuntu.com/releases/22.04/release/ubuntu-22.04-server-cloudimg-arm64.img"
  arch: "aarch64"

mounts: []
containerd:
  system: false
  user: false
```

- caution! → you must use `aarch64` for “arch”
- you can check whether what you need to use for “arch”

  ```
  uname -m
  ```

  - arm64 → M1/M2/M3 계열
  - 이 경우 Lima VM도 보통 arch: "aarch64" 로 맞추는 게 자연스러움

3. Install lima

```
brew install lima docker docker-compose
```

4. Create lima container and wait a sec...

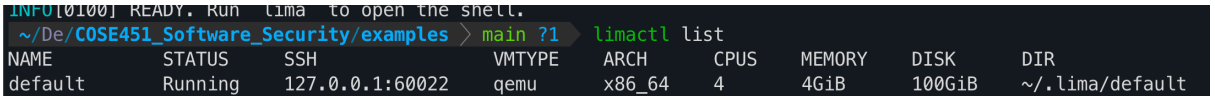
```
limactl start --name=default ./ubuntu-lts.yaml
```

- Press enter at "> Proceed with the current configuration" and wait a sec

4-1. amd64로 빌드해버린 경우...

```
# 실행중인 lima 이미지 확인
ps aux | grep lima

# 이미지 실행 중단
kill -9 <PID1> <PID2> <PID3>

# 남은 소켓 이미지 제거
rm -f ~/.lima/default/*.sock

# default 인스턴스 디렉토리 완전 삭제
rm -rf ~/.lima/default

# 다시 빌드 (yaml 파일 수정 후)
limactl start --name=default ./ubuntu-lts.yaml
```

5. Enter 'limactl list' to check if the container has been created

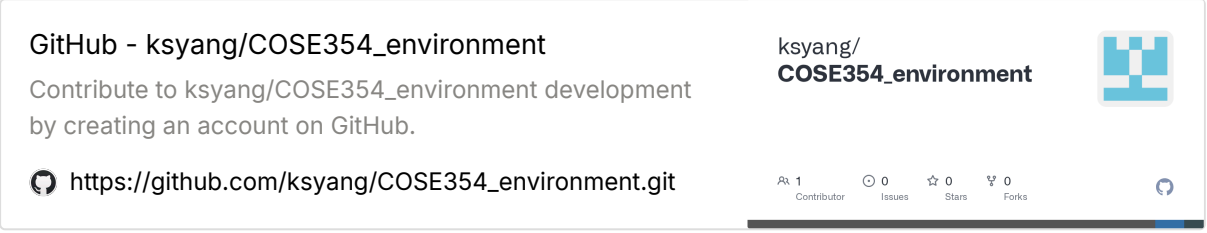


6. Enter 'lima' to access the container shell

```
lima
```

```
~/De/COSE451_Software_Security/examples  main ?1  lima
pumice@lima-default:~$ uname -a
Linux lima-default 5.15.0-101-generic #111-Ubuntu SMP Tue Mar 5 20:16:58 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux
pumice@lima-default:~$ ls
pumice@lima-default:~$
```

7. Clone below repository



8. Move to cloned directory

9. Add stable repository for install docker

```
echo \
  "deb [arch=arm64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
wget https://download.docker.com/linux/ubuntu/dists/focal/pool/stable/arm64/docker-buildx-plugin_0.10.4-1~ubuntu.20.04~focal_arm64.deb
```

```
sudo dpkg --install ./docker-buildx-plugin_0.10.4-1~ubuntu.20.04~focal_arm64.deb
```

10. apt-get update & install docker

```
sudo apt-get update
sudo apt-get install docker.io
```

11. Build docker

```
sudo docker build --progress=plain --tag test_cose354 .
```

11-1. 아래와 같은 오류가 나는 경우

```
potato@lima-default:~/COSE354_environment$ sudo docker build --progre
ss=plain --tag test_cose354 .
...
#7 0.588 Reading state information...
#7 0.594 Package gcc-multilib is not available, but is referred to by another
package.
#7 0.594 This may mean that the package is missing, has been obsoleted,
or
#7 0.594 is only available from another source
#7 0.594
#7 0.595 E: Package 'gcc-multilib' has no installation candidate
------
Dockerfile:6
--------------------
  4 |
  5 |     RUN apt-get update
  6 | >>> RUN apt-get -y install gcc gdb gcc-multilib git vim python3-pip g
dbserver netcat
  7 |     RUN python3 -m pip install --upgrade pip
  8 |     RUN python3 -m pip install --upgrade pwntools
--------------------
ERROR: failed to solve: process "/bin/sh -c apt-get -y install gcc gdb gcc-
multilib git vim python3-pip gdbserver netcat" did not complete successfull
y: exit code: 100
```

- 아래처럼 Dockerfile을 수정

```
# nano 에디터로 Dockerfile 열기
potato@lima-default:~/COSE354_environment$ nano Dockerfile
```

```
FROM ubuntu:22.04

ENV user cose-354

RUN apt-get update
RUN apt-get -y install gcc gdb git vim python3-pip gdbserver netcat # 수정된 부분.
# gcc-multilib 삭제
RUN python3 -m pip install --upgrade pip
RUN python3 -m pip install --upgrade pwntools

RUN adduser $user

ADD ./Stage1_Mallory/. /home/$user/Stage1_Mallory/
ADD ./Stage2_Trent/. /home/$user/Stage2_Trent/
ADD ./Stage3_Mallory/. /home/$user/Stage3_Mallory/
ADD ./Stage4_Trent/. /home/$user/Stage4_Trent/
ADD ./Stage5_Trent/. /home/$user/Stage5_Trent/
ADD ./Stage6_Trent/. /home/$user/Stage6_Trent/
ADD ./Stage7_Bob/. /home/$user/Stage7_Bob/
....
```

- 이후 다시 빌드

12. To verify the image ID of the built image using the “sudo docker images” command

```
pumice@pumice-MS-7D42:~/.../COSE354_environment$ docker images
REPOSITORY                          TAG       IMAGE ID       CREATED          SIZE
test_cose354                        latest    f87a526694bd   10 minutes ago   949MB
```

13. Execute the following command to run the container and access the bash shell

```
sudo docker run -it {image id} /bin/bash
```

14. Great! If the output of the 'ls' command works, your setup is complete!