How to Setup Local Harbor Server

설치 전 과정

1. 권장 실습 환경

   ① Windows 11

   ② 한글 계정명 사용 금지

   ③ C Drive 여유공간 50GB 이상

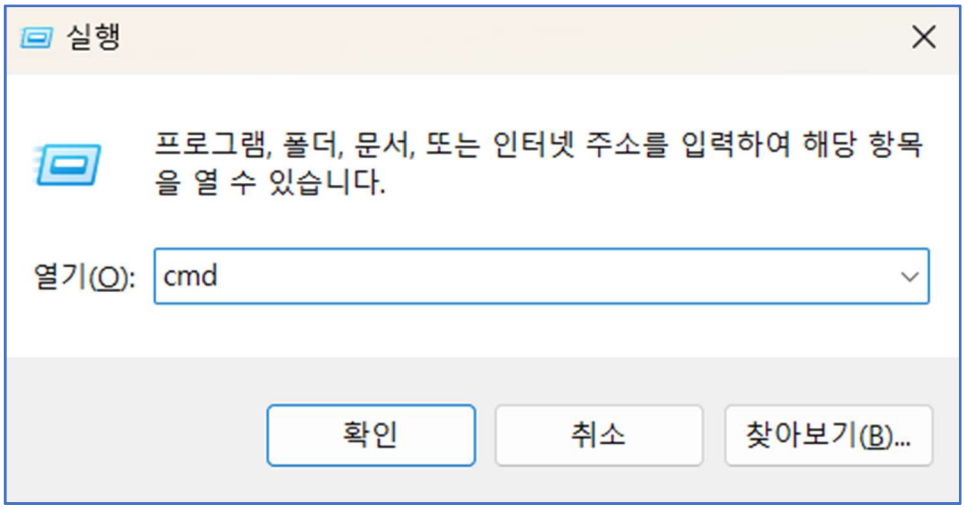
   ④ RAM 16GB 이상 권장

   ⑤ 안정적인 Network 환경(가급적 유선연결)

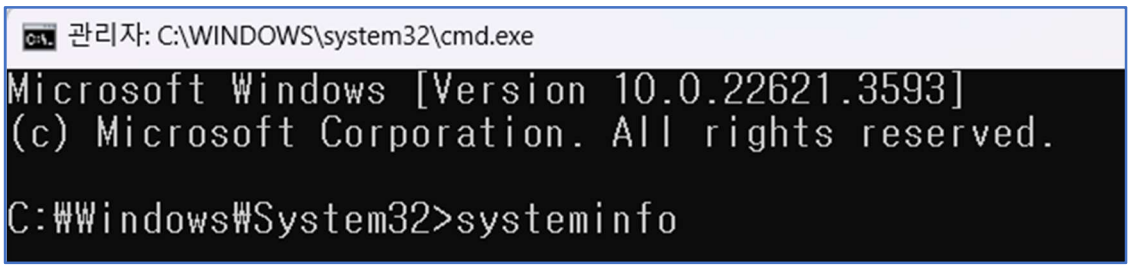
2. Hyper-V 설정 확인

   ※ 실습용 Windows 시스템에 Hyper-V 기능이 활성화되어 있으면 실습 불가

   ① Windows Key + r > 실행 창 > cmd > Ctrl + Shift + Enter



   ② **systeminfo**

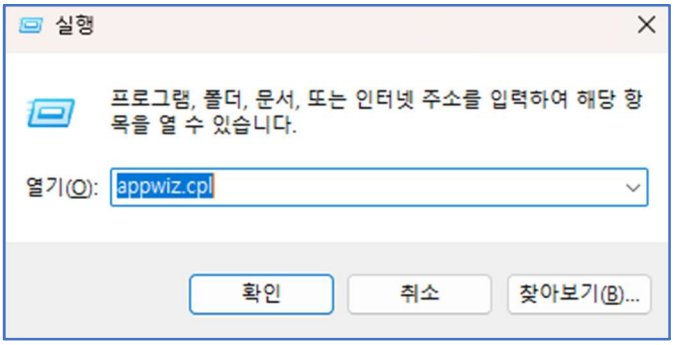


③ 출력 결과 확인

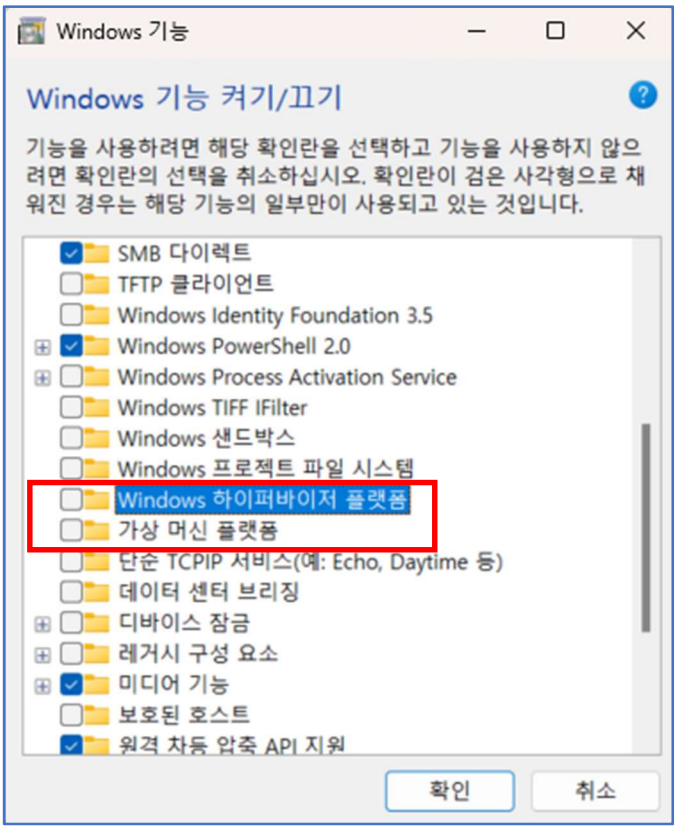
1) 추가 작업 필요

```
                                상태:         미디어 연결이 끊어짐
Hyper-V 요구 사항:      하이퍼바이저가 검색되었습니다. Hyper-V에 필요한 기능이 표시되지 않습니다.
```

- 실행 창에 **[appwiz.cpl]** 입력 > **[Windows 기능 켜기/끄기]**

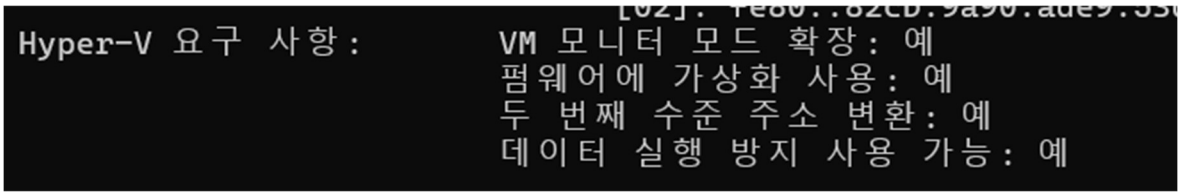


- 아래 그림과 같이 **[Windows 하이퍼바이저 플랫폼]**과 **[가상 머신 플랫폼]**을 체크 해제한다.



- [확인] 버튼을 클릭하고 시스템을 재부팅한다.

2) 정상인 경우



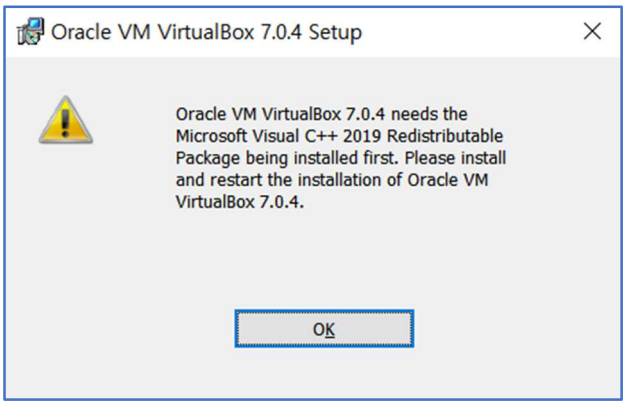
3. Oracle VirtualBox 설치

① VirtualBox 다운로드 및 설치 (VMware, Hyper-V 제거 필요)

② https://www.virtualbox.org, VirtualBox-7.0.6-155176-Win.exe

③ 설치파일 다운로드 후 관리자 권한으로 실행 (한글 계정명 사용 금지)

④ Microsoft Visual C++ 2019 Redistributable Package Error 발생 시



- Microsoft Visual C++ 2019 Redistributable Package 설치 후 진행

https://learn.microsoft.com/en-US/cpp/windows/latest-supported-vc-redist?view=msvc-170

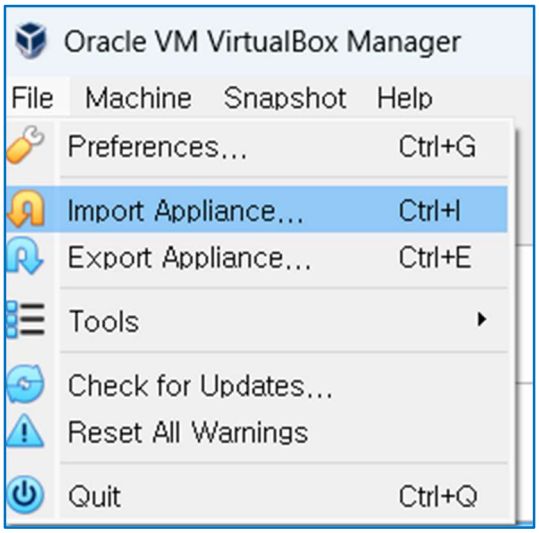
- VC_redist.x64.exe 설치 후 다시 VirtualBox 설치할 것

⑤ VirtualBox 호스트키 조합 설정

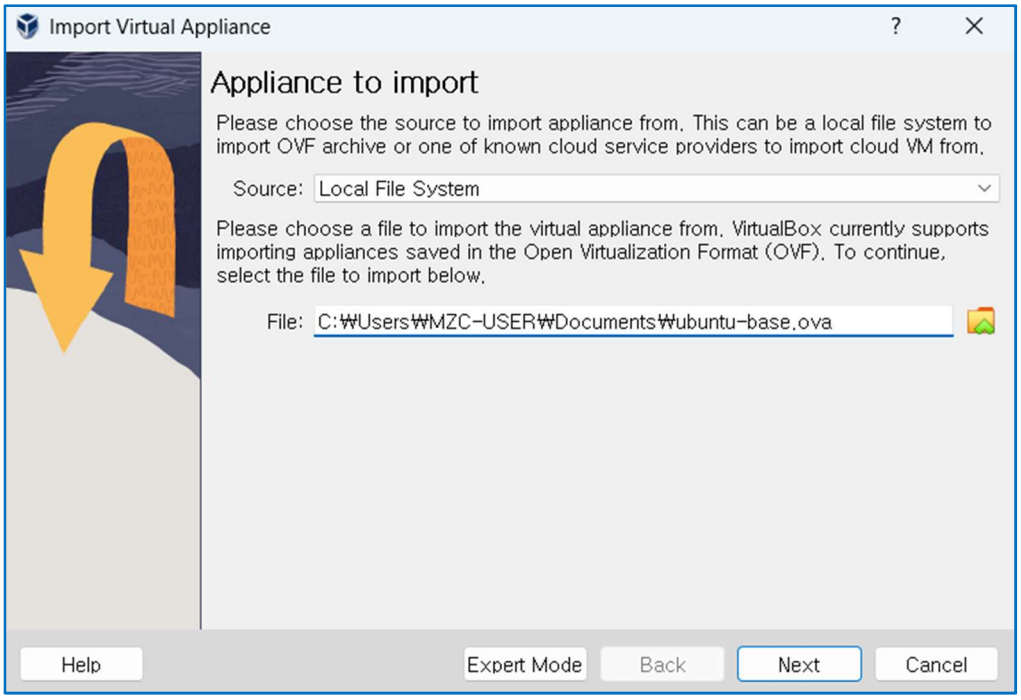
- 파일 > 환경 설정 > 입력 > 가상 머신
- 호스트 키 조합 > **F12**

4. Ubuntu VM Import – Ubuntu Linux Server 22.04 LTS

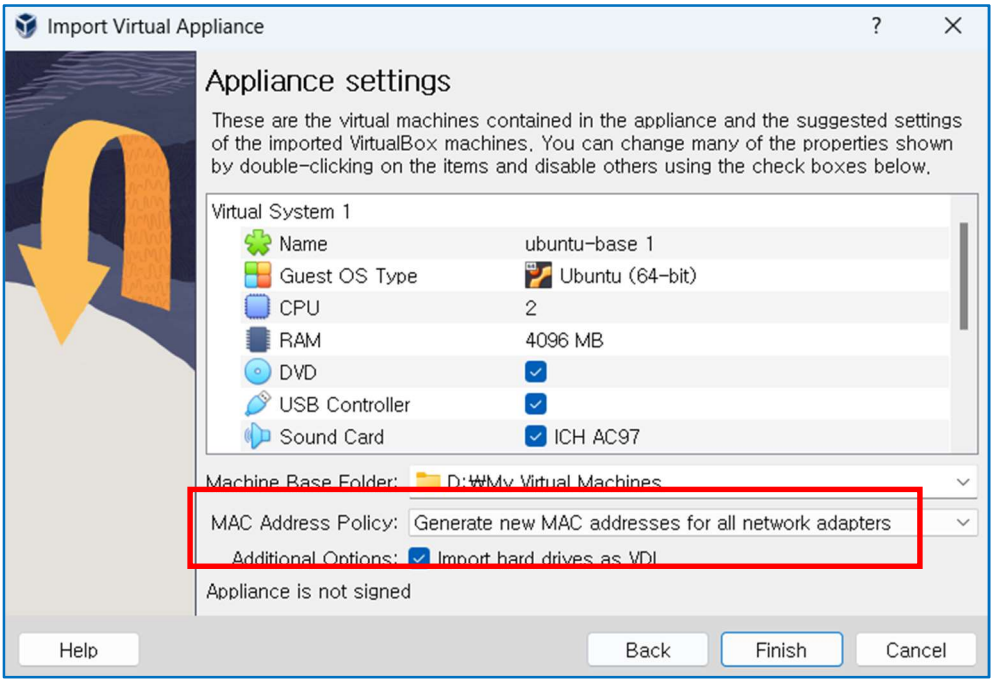
① **[파일]** > **[가상 시스템 가져오기(Import Appliance)]** > 파일



② "ubuntu-base.ova" 선택 > **[열기]** > **[다음]**



③ **[Appliance settings]** 창에서, **[MAC 주소 정책(MAC Address Policy)] : [모든 네트워크 어댑터의 새 MAC 주소 생성(Generate new MAC addresses for all network adapters)]** → 반드시 선택하여 변경할 것 > **[Finish]**



5. VirtualBox - NatNetwork 설정(192.168.137.0/24)

① **[도구(Tools)]** > **[network]** > **[만들기(Create)]** > **NatNetwork**

② **[General Options]** > **[IP4 Prefix]** : **192.168.137.0/24** > **[적용(Apply)]**

③ **[포트 포워딩(Port Forwarding)]** > 새 포트 포워딩 규칙 추가(+)

- **[이름]** : ubuntu
- **[호스트 IP]** : 192.168.56.1
- **[호스트 포트]** : 100
- **[게스트 IP]** : 192.168.137.100
- **[게스트 포트]** : 22

이름	프로토콜	호스트 IP	호스트 포트	게스트 IP	게스트 포트
ubuntu	TCP	192.168.56.1	100	192.168.137.100	22

④ **[적용(Apply)]**

6. Ubuntu VM IP 설정 및 Hostname 변경하기

① 로그인 후, **00-installer-config.yaml** 파일 수정

vi /etc/netplan/00-installer-config.yaml ← 각 라인의 들여쓰기는 반드시 2칸이다.

```
# This is the network config written by 'subiquity'
network:
  renderer: NetworkManager
  ethernets:
    enp0s3:
      dhcp4: no
      addresses:
      - 192.168.137.110/24
      routes:
      - to: default
        via: 192.168.137.1
      nameservers:
        addresses: [8.8.8.8,8.8.7.7]
  version: 2
~
```

netplan try

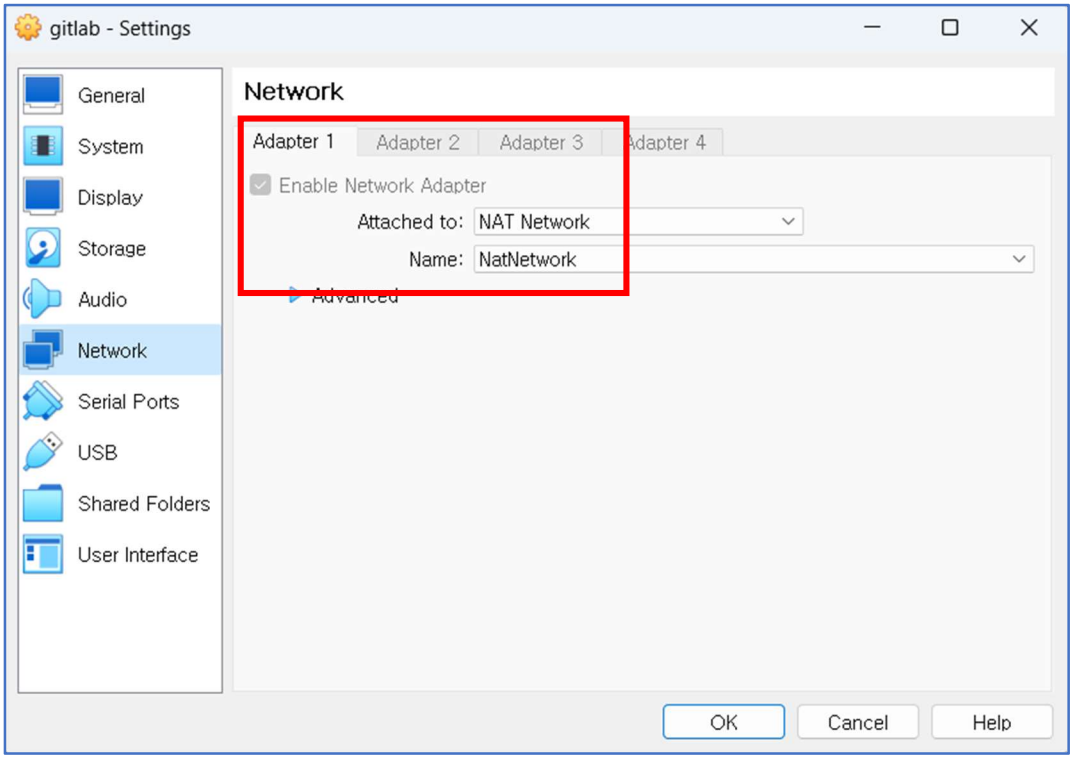
netplan apply

ip a

```
root@harbor:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:86:07:04 brd ff:ff:ff:ff:ff:ff
    inet 192.168.137.110/24 brd 192.168.137.255 scope global noprefixroute enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe86:704/64 scope link
       valid_lft forever preferred_lft forever
root@harbor:~#
```

② Ubuntu VM Network Adapter 변경하기

- Ubuntu VM > **[Settings]** > **[Network]** > **[Apater1]** > **[Attached to]** : NAT에서 **NAT Network**로 변경
- 이름이 **NatNetwork**로 변경 확인 **[OK]** 버튼 클릭하여 적용



③ 호스트 이름은 다음의 명령으로 변경한다.

```
# hostnamectl set-hostname gitlab
```

④ 호스트 이름 변경 후 확인한다.

```
# hostname
```

```
root@ubuntu:~# hostname
harbor
root@ubuntu:~#
```

⑤ /etc/hosts 파일 수정하기

```
# vi /etc/hosts
```

```
127.0.0.1 localhost
127.0.1.1 harbor

192.168.137.110    harbor.example.com
```

⑥ Network Test를 위해 다음의 명령을 수행한다.

apt update

⑦ 만일 오류가 발생하면 다음의 명령을 수행한다.

vi /etc/resolv.conf

nameserver 8.8.8.8

```
# This is /run/systemd/resolve/stub-resolv.conf managed by man:systemd-resolved(8).
# Do not edit.
#
# This file might be symlinked as /etc/resolv.conf. If you're looking at
# /etc/resolv.conf and seeing this text, you have followed the symlink.
#
# This is a dynamic resolv.conf file for connecting local clients to the
# internal DNS stub resolver of systemd-resolved. This file lists all
# configured search domains.
#
# Run "resolvectl status" to see details about the uplink DNS servers
# currently in use.
#
# Third party programs should typically not access this file directly, but only
# through the symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a
# different way, replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes of
# operation for /etc/resolv.conf.

nameserver 8.8.8.8
options edns0 trust-ad
search .
```

⑧ 저장하면 바로 적용됨

ping -c 4 www.google.com

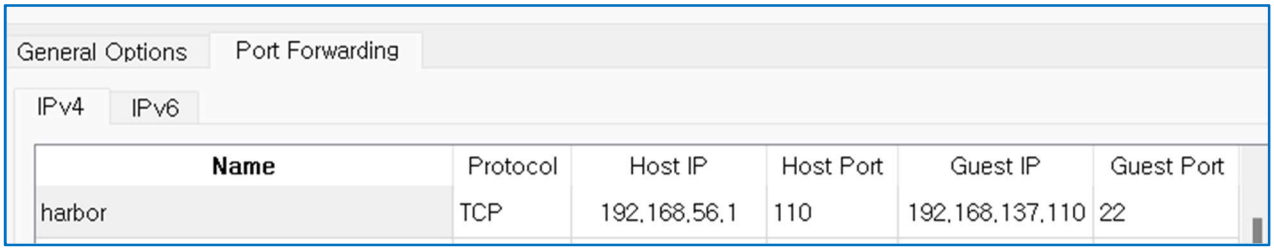
```
ubuntu@ubuntu:~$ ping -c 4 www.google.com
PING www.google.com (142.250.206.196) 56(84) bytes of data.
64 bytes from kix07s07-in-f4.1e100.net (142.250.206.196): icmp_seq=1 ttl=112 time=58.8 ms
64 bytes from kix07s07-in-f4.1e100.net (142.250.206.196): icmp_seq=2 ttl=112 time=56.5 ms
64 bytes from kix07s07-in-f4.1e100.net (142.250.206.196): icmp_seq=3 ttl=112 time=56.9 ms
64 bytes from kix07s07-in-f4.1e100.net (142.250.206.196): icmp_seq=4 ttl=112 time=57.0 ms

--- www.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3050ms
rtt min/avg/max/mdev = 56.507/57.304/58.766/0.866 ms
ubuntu@ubuntu:~$ _
```

⑨ Network Test를 위해 다음의 명령을 다시 수행한다.

```
# apt update
```

```
root@ubuntu:~# apt update
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:2 http://kr.archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://kr.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1,475 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [254 kB]
Get:6 http://kr.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:7 http://kr.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1,687 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [854 kB]
Get:9 http://kr.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [313 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [165 kB]
Get:11 http://kr.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1,076 kB]
Get:12 http://kr.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [247 kB]
Get:13 http://kr.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [67.1 kB]
Get:14 http://kr.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [27.2 kB]
Fetched 6,548 kB in 6s (1,013 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
31 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ubuntu:~#
```

7. VirtualBox에서 [Port Forwarding]을 다음과 같이 추가한다.

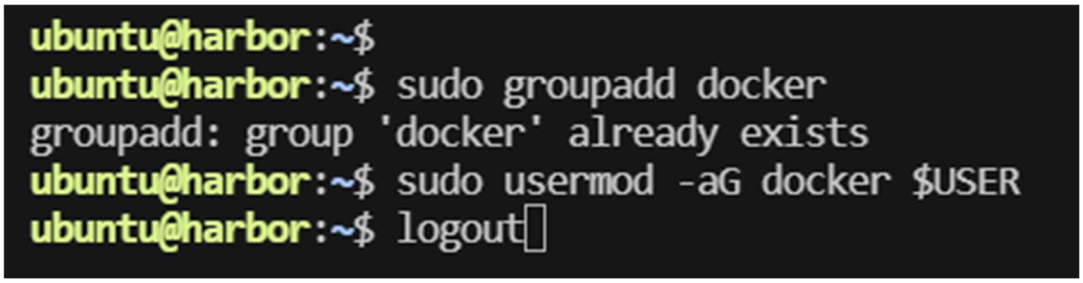


8. Ubuntu VM 종료 후 Ubuntu VM의 이름은 **harbor**로 변경 후 **[시작]** > **[Headless Start]** 선택 > SSH Client Tool(PuTTY, Tabby, MobaXterm 등)으로 로그인하기

Harbor Server 설치하기

1. 다음과 같이 harbor을 설치한다. 먼저 Docker를 설치한다. (ref https://goharbor.io/docs/2.10.0/install-config/)

```
$ sudo apt-get update

$ sudo apt full-upgrade

$ sudo apt install apt-transport-https ca-certificates curl

$ sudo install -m 0755 -d /etc/apt/keyrings

$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

$ sudo chmod a+r /etc/apt/keyrings/docker.asc


$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

$ sudo groupadd docker

$ sudo usermod -aG docker $USER

$ logout
```



2. 다시 로그인 후, 현재 계정이 docker의 권한이 있는지 확인한다.

   $ **id**

   $ **newgrp docker**

```
ubuntu@harbor:~$ id
uid=1000(ubuntu) gid=1000(ubuntu) groups=1000(ubuntu),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lxd),999(docker)
ubuntu@harbor:~$ newgrp docker
ubuntu@harbor:~$
```

3. 다음과 같이 Harbor 다운로드한다.

   $ **cd /tmp**

   $ **curl -s https://api.github.com/repos/goharbor/harbor/releases/latest | grep browser_download_url | cut -d '"' -f 4 | grep '\.tgz$' | wget -i -**

```
ubuntu@harbor:~$ cd /tmp
ubuntu@harbor:/tmp$ curl -s https://api.github.com/repos/goharbor/harbor/releases/latest | grep browser_download_url | cut -d '"' -f 4 | grep '\.tgz$' | wget -i -
--2024-06-05 02:39:47--  https://github.com/goharbor/harbor/releases/download/v2.10.2/harbor-offline-installer-v2.10.2.tgz
Resolving github.com (github.com)... 20.200.245.247
Connecting to github.com (github.com)|20.200.245.247|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/50613991/5c0c9e98-4eed-400a-8849-911d6c8e0060?X-Amz-Algorithm=AWS4-HMAC-SHA25
F20240605%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20240605T023947Z&X-Amz-Expires=300&X-Amz-Signature=98dd1bdd4450db010049edec9367aafffaa0310ae74127a96a04a6568f69
y_id=0&repo_id=50613991&response-content-disposition=attachment%3B%20filename%3Dharbor-offline-installer-v2.10.2.tgz&response-content-type=application%2Foctet-strea
--2024-06-05 02:39:47--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/50613991/5c0c9e98-4eed-400a-8849-911d6c8e0060?X-Amz-Algorithm=
setproduction%2F20240605%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20240605T023947Z&X-Amz-Expires=300&X-Amz-Signature=98dd1bdd4450db010049edec9367aafffaa0310ae7412
t&actor_id=0&key_id=0&repo_id=50613991&response-content-disposition=attachment%3B%20filename%3Dharbor-offline-installer-v2.10.2.tgz&response-content-type=applicatio
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.109.133, 185.199.111.133, 185.199.108.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 672994516 (642M) [application/octet-stream]
Saving to: 'harbor-offline-installer-v2.10.2.tgz'

harbor-offline-installer-v2.10.2.tgz              100%[=============================================================================================================

2024-06-05 02:40:04 (38.9 MB/s) - 'harbor-offline-installer-v2.10.2.tgz' saved [672994516/672994516]
```

4. 다운로드 받은 파일의 압축 푼다.

   $ **tar -xzvf harbor-offline-installer-v2.10.2.tgz**

```
ubuntu@harbor:/tmp$ tar -xzvf harbor-offline-installer-v2.10.2.tgz
harbor/harbor.v2.10.2.tar.gz
harbor/prepare
harbor/LICENSE
harbor/install.sh
harbor/common.sh
harbor/harbor.yml.tmpl
```

5. 압축 푼 디렉토리를 **opt** 디렉토리로 이동시키고 **harbor.yml.tmpl** 을 **harbor.yml**로 이름 변경한다.

```
$ sudo mv harbor /opt/

$ cd /opt/harbor

$ cp harbor.yml.tmpl harbor.yml
```

6. 이름 변경한 harbor.yml 파일을 열고 다음과 같이 수정한다.

```
$ sudo vi harbor.yml

-hostname : harbor.example.com

-https ← 모두 주석 처리한다.
```

```
# Configuration file of Harbor

# The IP address or hostname to access admin UI and registry service.
# DO NOT use localhost or 127.0.0.1, because Harbor needs to be accessed by external clients.
hostname: harbor.example.com

# http related config
http:
  # port for http, default is 80. If https enabled, this port will redirect to https port
  port: 80

# https related config
#https:
  # https port for harbor, default is 443
  #  port: 443
  # The path of cert and key files for nginx
  #certificate: /your/certificate/path
  #private_key: /your/private/key/path
  ## enable strong ssl ciphers (default: false)
  ## strong_ssl_ciphers: false

# # Uncomment following will enable tls communication between all harbor components
# internal_tls:
#   # set enabled to true means internal tls is enabled
#   enabled: true
#   # put your cert and key files on dir
#   dir: /etc/harbor/tls/internal
```

7. 그리고 파일을 아래로 더 내려서 harbor의 admin 패스워드를 확인한다.

-harbor_admin_password: Harbor12345

```
# The initial password of Harbor admin
# It only works in first time to install harbor
# Remember Change the admin password from UI after launching Harbor.
harbor_admin_password: Harbor12345

# Harbor DB configuration
database:
  # The password for the root user of Harbor DB. Change this before any production use.
  password: root123
  # The maximum number of connections in the idle connection pool. If it <=0, no idle con
  max_idle_conns: 100
  # The maximum number of open connections to the database. If it <= 0, then there is no
  # Note: the default number of connections is 1024 for postgres of harbor.
  max_open_conns: 900
  # The maximum amount of time a connection may be reused. Expired connections may be clo
```

8. 다음과 같이 설치 스크립트를 실행한다.

$ sudo ./prepare

$ sudo ./install.sh

```
ubuntu@harbor:/opt/harbor$ sudo ./prepare
prepare base dir is set to /opt/harbor
Unable to find image 'goharbor/prepare:v2.10.2' locally
v2.10.2: Pulling from goharbor/prepare
969313660a9f: Pull complete
fe3af7be4f17: Pull complete
69ae26cf39c4: Pull complete
54b8836e9ccb: Pull complete
2ae2c7e23174: Pull complete
7b64bd9a31ee: Pull complete
6c3c24c1ab06: Pull complete
4ed95387ff2a: Pull complete
06071e53e40a: Pull complete
4c2f6b0039ce: Pull complete
Digest: sha256:016db1a2011585b9142dc2e0422cdaf192f8f7ff1d9c79b0c453e760e5233151
Status: Downloaded newer image for goharbor/prepare:v2.10.2
WARNING:root:WARNING: HTTP protocol is insecure. Harbor will deprecate http protocol in the future. Please make sure to upgrade to https
Generated configuration file: /config/portal/nginx.conf
Generated configuration file: /config/log/logrotate.conf
Generated configuration file: /config/log/rsyslog_docker.conf
Generated configuration file: /config/nginx/nginx.conf
Generated configuration file: /config/core/env
Generated configuration file: /config/core/app.conf
Generated configuration file: /config/registry/config.yml
Generated configuration file: /config/registryctl/env
Generated configuration file: /config/registryctl/config.yml
Generated configuration file: /config/db/env
Generated configuration file: /config/jobservice/env
Generated configuration file: /config/jobservice/config.yml
Generated and saved secret to file: /data/secret/keys/secretkey
Successfully called func: create_root_cert
Generated configuration file: /compose_location/docker-compose.yml
Clean up the input dir
ubuntu@harbor:/opt/harbor$
```

```
ubuntu@harbor:/opt/harbor$ sudo ./install.sh

[Step 0]: checking if docker is installed ...

Note: docker version: 26.1.3

[Step 1]: checking docker-compose is installed ...

Note: Docker Compose version v2.27.0

[Step 2]: loading Harbor images ...
a8840e4ae316: Loading layer [==================================================>]  21.63MB/21.63MB
515f0ee642d6: Loading layer [==================================================>]  173.8MB/173.8MB
7e1a15fa7f7c: Loading layer [==================================================>]  25.5MB/25.5MB
4cd39a5aa67d: Loading layer [==================================================>]  18.27MB/18.27MB
3daae6a42996: Loading layer [==================================================>]  5.12kB/5.12kB
e23a6fbc3acf: Loading layer [==================================================>]  6.144kB/6.144kB
87a67eef40a6: Loading layer [==================================================>]  3.072kB/3.072kB
5439373f0cf5: Loading layer [==================================================>]  2.048kB/2.048kB
44c50cb14a57: Loading layer [==================================================>]  2.56kB/2.56kB
e343431b8ac0: Loading layer [==================================================>]  7.68kB/7.68kB
Loaded image: goharbor/harbor-db:v2.10.2
a7437080fdff: Loading layer [==================================================>]  17.16MB/17.16MB
89c54c70338a: Loading layer [==================================================>]  3.584kB/3.584kB
927d190fe457: Loading layer [==================================================>]  2.56kB/2.56kB
dab52f006605: Loading layer [==================================================>]  44.92MB/44.92MB
```

```
Note: stopping existing Harbor instance ...
WARN[0000] /opt/harbor/docker-compose.yml: `version` is obsolete


[Step 5]: starting Harbor ...
WARN[0000] /opt/harbor/docker-compose.yml: `version` is obsolete
[+] Running 10/10
 ✓ Network harbor_harbor        Created
 ✓ Container harbor-log         Started
 ✓ Container registryctl        Started
 ✓ Container redis              Started
 ✓ Container harbor-db          Started
 ✓ Container harbor-portal      Started
 ✓ Container registry           Started
 ✓ Container harbor-core        Started
 ✓ Container harbor-jobservice  Started
 ✓ Container nginx              Started
✓ ----Harbor has been installed and started successfully.----
ubuntu@harbor:/opt/harbor$
```

9. HTTP를 통한 Harbor 접속을 위해서 docker 디렉토리 하위에 **daemon.json** 파일을 생성한다.

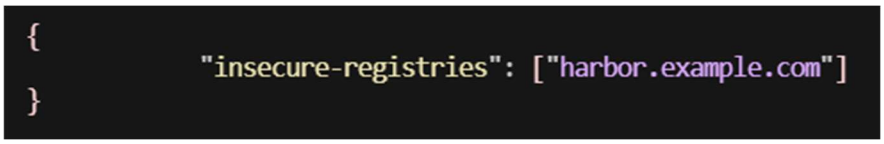
   $ **cd /etc/docker**

   $ **sudo vi daemon.json**

```
ubuntu@harbor:/opt/harbor$ cd /etc/docker
ubuntu@harbor:/etc/docker$ sudo vi daemon.json
[sudo] password for ubuntu:
```

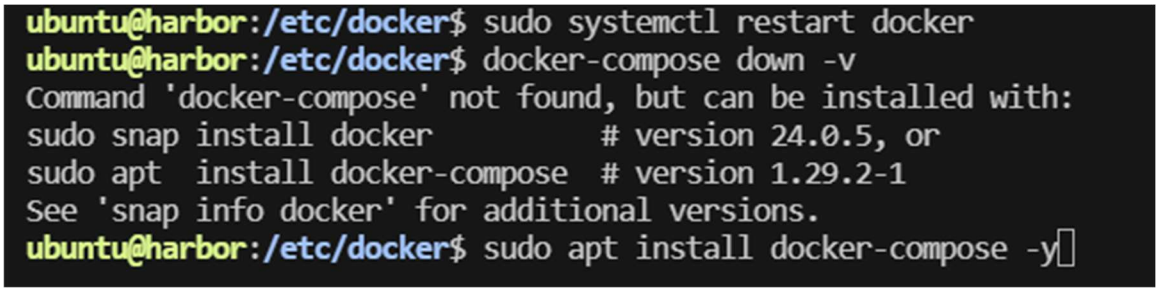
10. **daemon.json** 파일의 내용은 다음과 같다.

```
{
        "insecure-registries": ["harbor.example.com"]
}
```

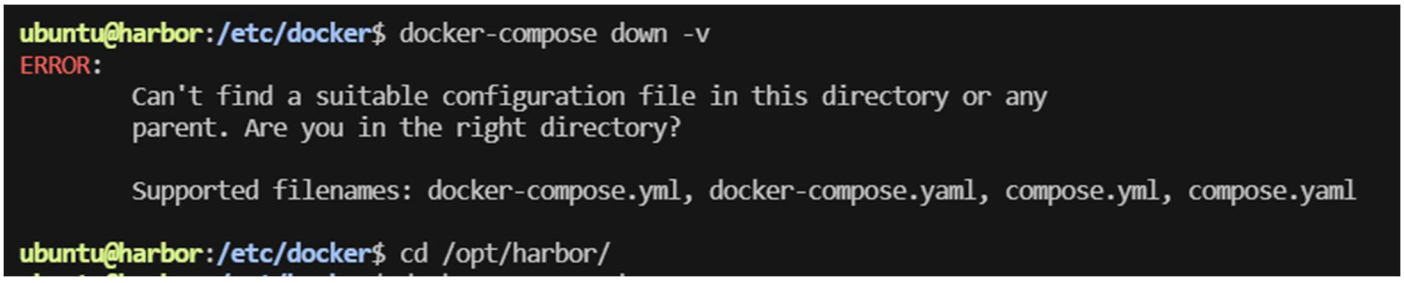


11. docker 서비스를 재 시작한다. 또한 docker-compose를 통해 harbor 서비스를 재 가동한다. docker-compose가 미리 설치되어 있지 않으면 설치한다.

```
$ sudo systemctl restart docker

$ docker-compose down -v

$ sudo apt install docker-compose -y
```



12. docker-compose 설치 후에 다시 재가동한다. 문제는 이 명령을 실행하는 디렉토리가 harbor가 아니기 때문에 다음과 같은 오류를 만날 것이다.

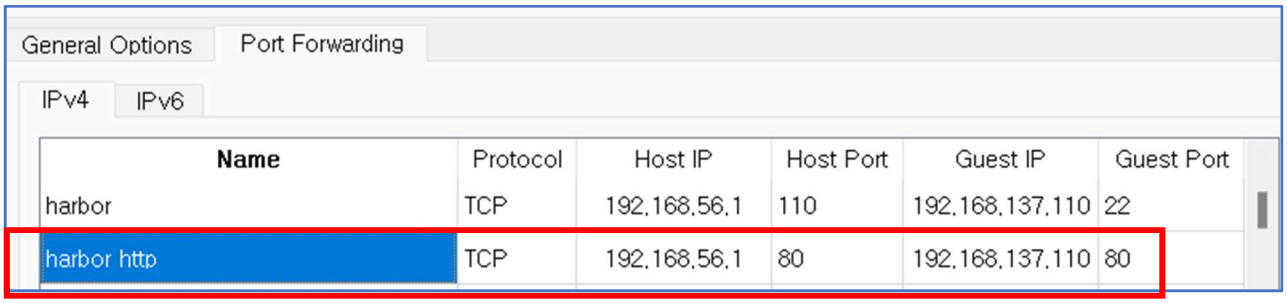


13. harbor 디렉토리로 이동하고 관리자 권한으로 harbor 서비스를 재가동한다.

```
$ cd /opt/harbor

$ sudo docker-compose down -v

$ sudo docker-compose up -d
```

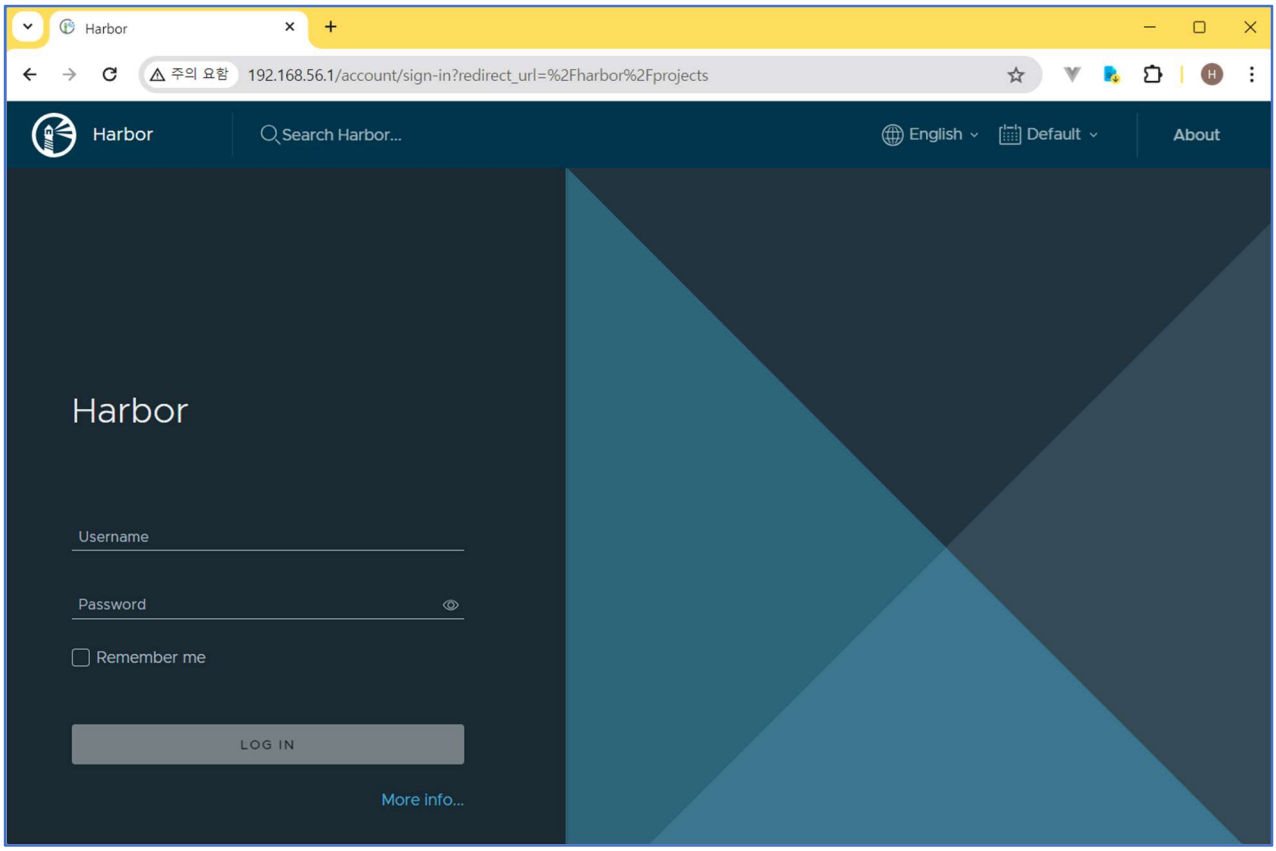
```
ubuntu@harbor:/opt/harbor$ sudo docker-compose down -v
Stopping harbor-jobservice ... done
Stopping redis             ... done
Stopping registry          ... done
Stopping registryctl       ... done
Stopping harbor-portal     ... done
Stopping harbor-log        ... done
Removing harbor-jobservice ... done
Removing nginx             ... done
Removing harbor-core       ... done
Removing redis             ... done
Removing registry          ... done
Removing registryctl       ... done
Removing harbor-portal     ... done
Removing harbor-db         ... done
Removing harbor-log        ... done
Removing network harbor_harbor
ubuntu@harbor:/opt/harbor$
ubuntu@harbor:/opt/harbor$ sudo docker-compose up -d
Creating network "harbor_harbor" with the default driver
Creating harbor-log ... done
Creating redis          ... done
Creating registry       ... done
Creating registryctl    ... done
Creating harbor-portal ... done
Creating harbor-db      ... done
Creating harbor-core    ... done
Creating harbor-jobservice ... done
Creating nginx             ... done
ubuntu@harbor:/opt/harbor$
```

14. VirtualBox에서 **[Port Forwarding]**을 다음과 같이 새로 생성한다. Port를 80으로 한 이유는 위에서 gitlab을 설치할 때 EXTERNAL_URL의 주소에 포트를 별도로 지정하지 않았기 때문이다. 만일 EXTERNAL_URL의 설정 시 포트를 8080이나 다른 포트 번호를 지정했다면 동일한 포트 넘버는 설정해야 한다.



15. 모든 설정을 마치면 브라우저를 열고 다음과 같이 harbor에 접속한다.

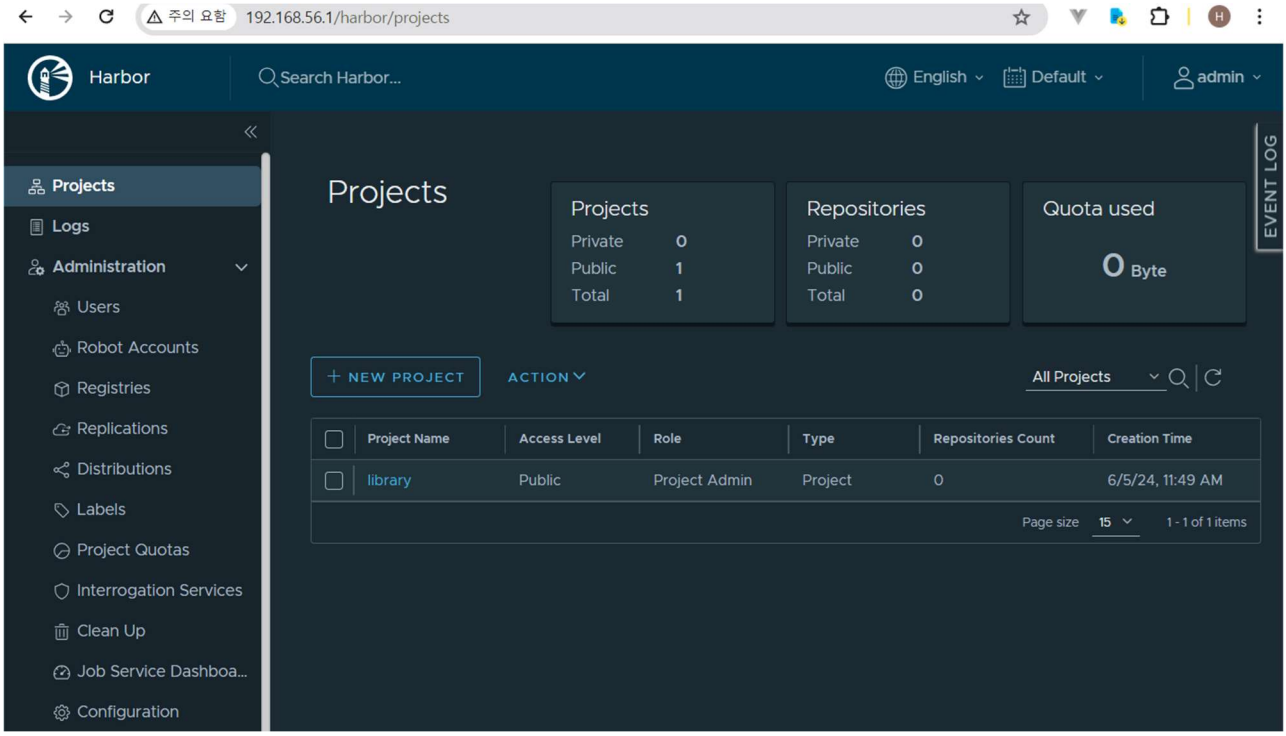
http://192.168.56.1



16. **harbor.yml** 파일에 저장한 **admin** 로그인한다.

① **[Username]** : `admin`

② **[Password]** : `Harbor12345`



Harbor Server에 Image Push하기

1. Harbor 서버가 설치됐기 때문에 테스트 이미지를 Push 해보자. 테스트를 위해 **nginx:alpine** 버전을 다운로드한다.

   $ **docker pull nginx:alpine**

```
ubuntu@harbor:~$ pwd
/home/ubuntu
ubuntu@harbor:~$ docker pull nginx:alpine
alpine: Pulling from library/nginx
4abcf2066143: Pull complete
b1e69ebc7f92: Pull complete
628158b45bce: Pull complete
346e52e95fa0: Pull complete
8c57fb1cd644: Pull complete
dc3800d1d0f2: Pull complete
e3227d68030d: Pull complete
8c50e1264d11: Pull complete
Digest: sha256:69f8c2c72671490607f52122be2af27d4fc09657ff57e42045801aa93d2090f7
Status: Downloaded newer image for nginx:alpine
docker.io/library/nginx:alpine
```

2. **nginx:alpine** 이미지가 잘 다운로드되었다.

```
ubuntu@harbor:~$ docker image ls
REPOSITORY                    TAG       IMAGE ID       CREATED         SIZE
nginx                         alpine    70ea0d8cc530   6 days ago      48.3MB
goharbor/harbor-exporter      v2.10.2   9befcab0cee2   8 weeks ago     111MB
goharbor/redis-photon         v2.10.2   9d1db211d49a   8 weeks ago     170MB
goharbor/trivy-adapter-photon v2.10.2   8f9e0b6b43ce   8 weeks ago     509MB
goharbor/harbor-registryctl   v2.10.2   e5a807ba1f59   8 weeks ago     155MB
goharbor/registry-photon      v2.10.2   850d2b3f27f3   8 weeks ago     89MB
goharbor/nginx-photon         v2.10.2   9282c21c2fee   8 weeks ago     159MB
goharbor/harbor-log           v2.10.2   f288fe2baa96   8 weeks ago     168MB
goharbor/harbor-jobservice    v2.10.2   a3247b57a920   8 weeks ago     146MB
goharbor/harbor-core          v2.10.2   6cd434d62456   8 weeks ago     174MB
goharbor/harbor-portal        v2.10.2   7e5a522c7853   8 weeks ago     167MB
goharbor/harbor-db            v2.10.2   cd385df354d4   8 weeks ago     274MB
goharbor/prepare              v2.10.2   bf4632d26b65   8 weeks ago     214MB
ubuntu@harbor:~$
```

3. docker hub에 로그인하듯이 앞에서 설치한 Harbor Server에 로그인한다.

```
$ docker login http://harbor.example.com
Username: admin
Password : Harbor12345
```

```
ubuntu@harbor:~$ docker login http://harbor.example.com
Username: admin
Password:
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu@harbor:~$
```

4. 먼저 docker 이미지 태깅을 하고 push한다.

```
$ docker tag nginx:alpine harbor.example.com/library/nginx:alpine
$ docker push harbor.example.com/library/nginx:alpine
```

```
ubuntu@harbor:~$ docker tag nginx:alpine harbor.example.com/library/nginx:alpine
ubuntu@harbor:~$ docker push harbor.example.com/library/nginx:alpine
The push refers to repository [harbor.example.com/library/nginx]
9cba8117003a: Pushed
b6d04dc5ecf7: Pushed
d38ed9b519d2: Pushed
3b4115e2edd1: Pushed
8d720e2faad3: Pushed
7b87df18a0ed: Pushed
a05d3326ce5a: Pushed
d4fc045c9e3a: Pushed
alpine: digest: sha256:090d69f2e5cb12b9b11c785c753fd5d095436921bc533404dec138558a13654e size: 1989
ubuntu@harbor:~$
```

5. 성공적으로 Harbor Server에 push했으면 Harbor Admin 페이지에서 확인해보자. 다음 그림과 같이 **library**하위에 **nginx**가 잘 올라온 것을 확인할 수 있다.