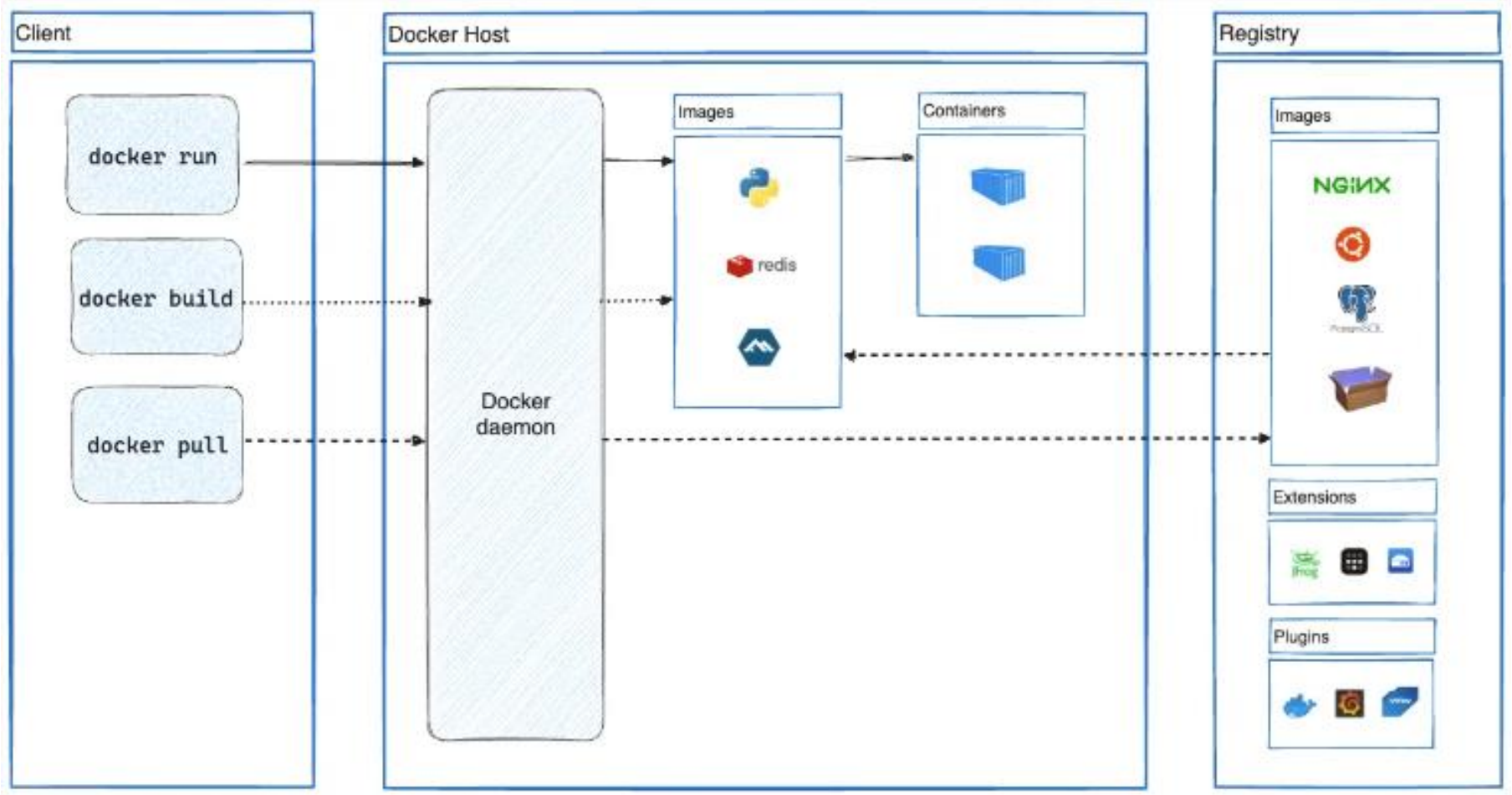

DOCKER

Docker

- Docker
  - 데이터 또는 프로그램을 격리시키는 기능을 제공하는 소프트웨어
  - 주로 서버에 사용되는 기술
  - 아파치, MySQL 등 여러 프로그램을 격리하여 실행
  - 애플리케이션을 개발, 제공 및 실행하기 위한 개방형 플랫폼
  - 애플리케이션을 관리하는 것과 동일한 방식으로 인프라를 관리
  - 코드 전달, 테스트 및 배포를 위한 Docker의 방법론을 활용
  - 코드 작성과 프로덕션 환경 실행 사이의 지연 시간을 크게 줄임

Docker Architecture

- Client-Server Architecture
  - Docker Daemon: Docker 컨테이너를 구축, 실행 및 배포하는 무거운 작업을 수행
  - Docker Client: Docker Daemon과 통신



Docker Software

◆ Docker Engine

- 구성 요소

  도커 데몬 (Docker Daemon): 컨테이너, 이미지 등을 관리하는 서버 프로세스

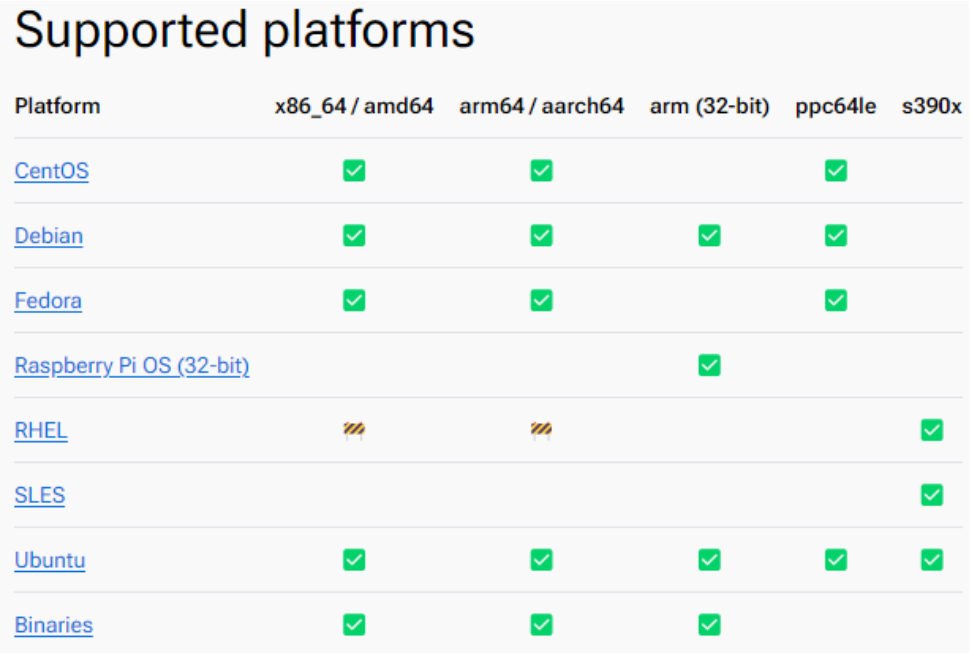
  도커 클라이언트 (Docker Client): 사용자가 도커 데몬과 상호작용하는 명령줄 도구

  REST API: 도커 데몬과 상호작용하기 위해 클라이언트가 사용하는 API

  도커 레지스트리 (Docker Registry): 도커 이미지를 저장하고 배포하는 저장소

◆ 도커 엔진의 전체 구조

- 도커 클라이언트가 명령을 전달
- 도커 데몬이 명령을 처리하고 필요한 작업 수행
- REST API를 통해 클라이언트와 데몬 간 통신
- 도커 레지스트리를 통해 이미지 저장 및 배포



Docker Software

- Docker Daemon(dockerd)
  - 도커 엔진의 서버 역할
  - Docker API 요청 수신
  - 이미지, 컨테이너, 네트워크, 볼륨과 같은 Docker 개체를 관리

- Docker Client(docker)
  - 많은 Docker 사용자가 Docker와 상호 작용하는 기본적인 방법
  - 클라이언트는 docker run 명령을 dockerd에 보내고 도커데몬이 명령을 수행
  - docker명령은 Docker API를 사용
  - Docker 클라이언트는 둘 이상의 데몬과 통신 가능

Docker

◆ Docker Container

- 앱의 각 구성 요소에 대해 격리된 프로세스
- React 앱, Python API 엔진, 데이터베이스 등 각 구성 요소는 완전히 격리된 환경에서 실행

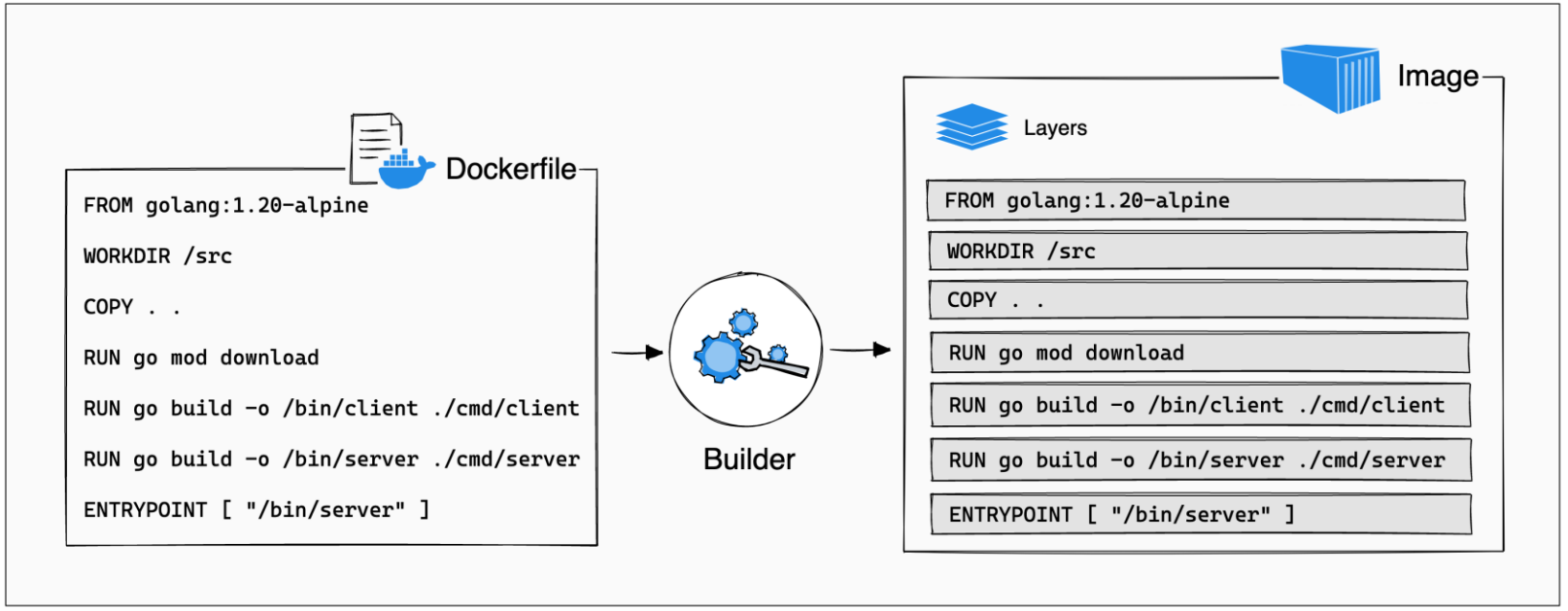
◆ 특징

- Self-contained(자체 완비)
  각 컨테이너에는 호스트 시스템에 사전 설치된 종속성에 의존하지 않고 작동하는 데 필요한 모든 것이 있음
- Isolated(격리)
  컨테이너는 격리되어 실행되므로 호스트 및 기타 컨테이너에 미치는 영향 최소화
  애플리케이션 보안 향상
- Independent(독립)
  각 컨테이너는 독립적으로 관리.
  하나의 컨테이너를 삭제해도 다른 컨테이너에는 영향을 미치지 않음
- Portable(휴대)
  컨테이너는 어디에서나 실행될 수 있음.
  개발 머신에서 실행되는 컨테이너는 데이터 센터나 클라우드 어디에서나 동일한 방식으로 작동

Docker

- Docker Image
  - Docker 컨테이너를 생성하기 위한 지침이 포함된 읽기 전용 템플릿
  - 자신만의 이미지를 생성 가능
  - 기존 이미지를 커스터마이징하여 새로운 이미지 생성 가능
  - Dockerfile 활용
  - 레이어(Layer)로 구성

- 도거 허브(Regestry)
  - https://hub.docker.com/



Docker

◆ 도커의 장점

- 한 대의 물리적 서버에 여러 대의 서버를 독립된 환경으로 구동

  여러 개의 컨테이너를 띄울 수 있으며 똑같은 애플리케이션도 여러 개 띄울 수 있음

  그 중 일부를 교체하거나 수정할 경우 다른 컨테이너에 영향을 주지 않음

- 이미지 활용

  처음부터 이미지를 만드는 것이 아닌 기존 이미지를 기반으로 작성이 가능
  구축이 단순하므로 교체가 쉽고 업데이트가 쉬움
  이동이 쉬워지고 개발환경을 동일하게 배포하기 좋음

- 컨테이너는 OS가 아닌 Process

  컨테이너 내에 OS의 핵심인 커널이 없고 호스트의 하드웨어에 의존하므로 가벼움
  다양한 배포판을 활용할 수 있음

Docker

◆ 도커의 단점?

- 리눅스용 소프트웨어만 지원
- 물리서버의 하드웨어를 공유하므로 물리서버에 문제가 생기면 안됨
- 물리서버의 문제 상황 발생에 대한 대처가 필요

◆ 도커의 주 용도?

- 팀원들에게 동일한 개발환경 제공
- 새로운 버전 테스트
- 동일한 서버가 여러 대 필요한 경우
- 애플리케이션 배포(CI/CD)

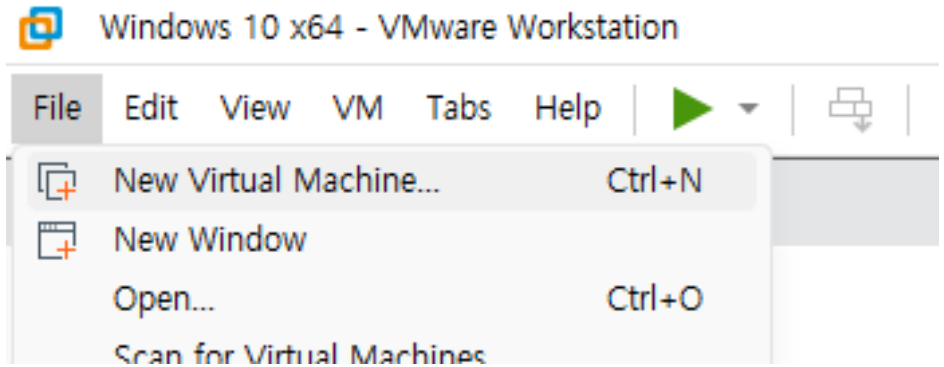
Docker 환경 구성하기

- 도커 사용 방법
  - 리눅스 OS가 설치된 컴퓨터에 도커 엔진을 설치하여 사용
  - 가상머신으로 게스트OS를 리눅스로 설치하고 도커 엔진을 설치하여 활용
  - 윈도우/Mac 용 Docker Desktop 사용(내부에 리눅스 같은 것이 동작됨)
  - Docker는 64bit OS에서만 동작

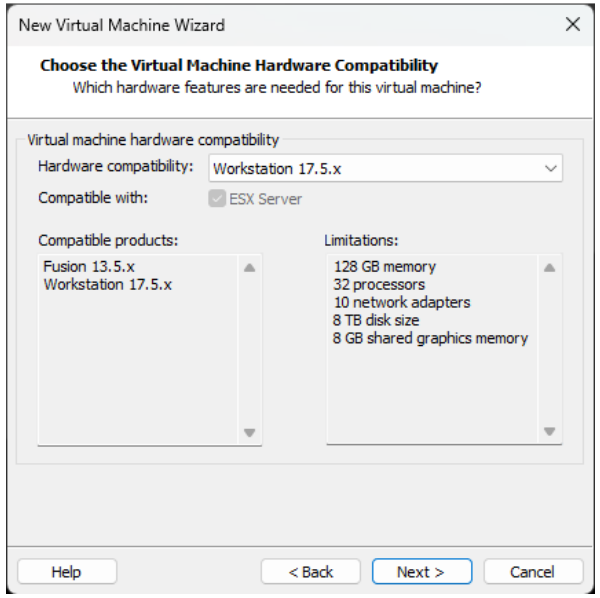
Docker 환경 구성

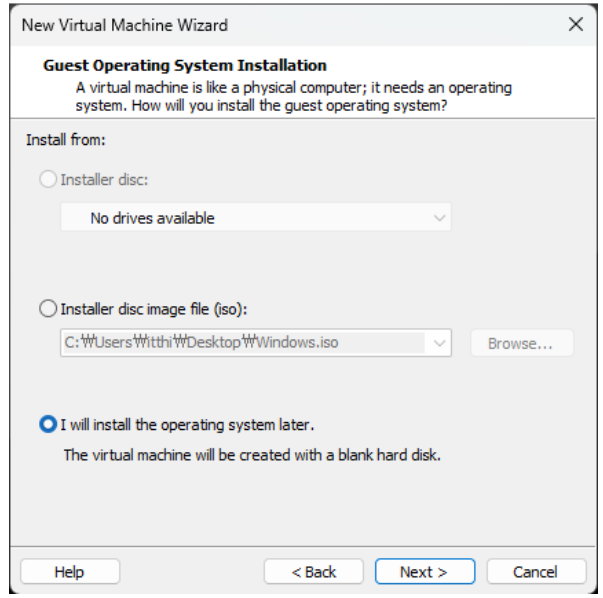
- 가상머신에서 환경 구성
  - VMware or VirtualBox 활용

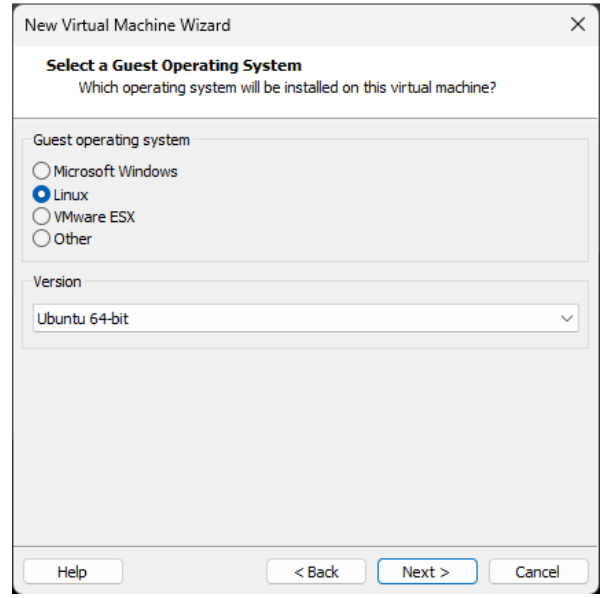
- 가상 머신 생성(VMware Workstation)

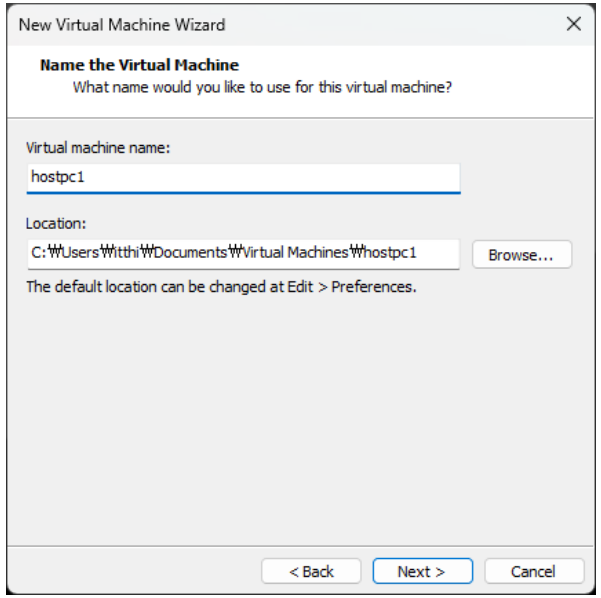


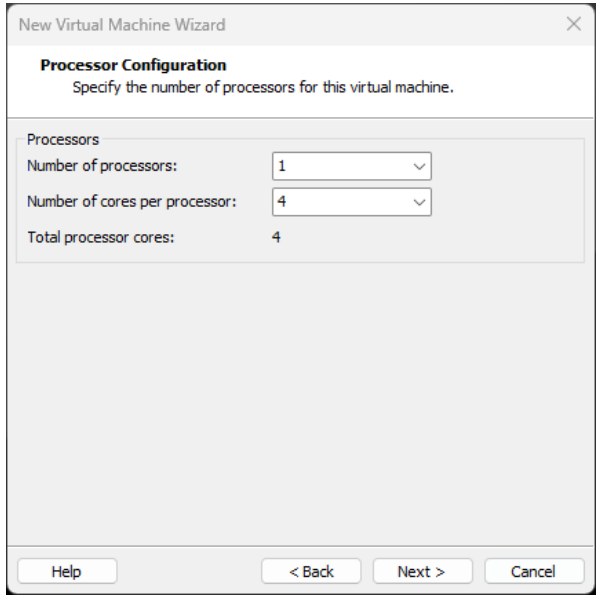
Docker Virtual Machine Create

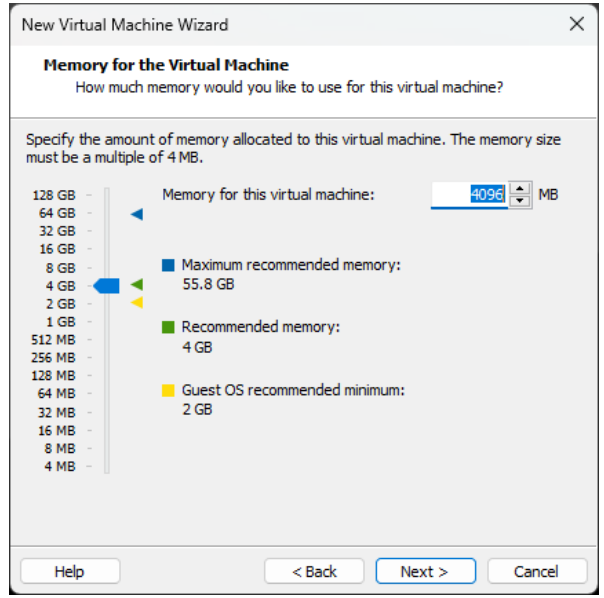


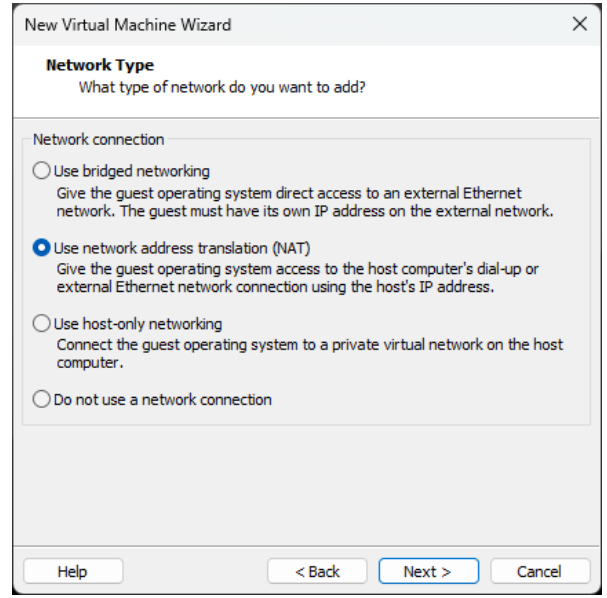




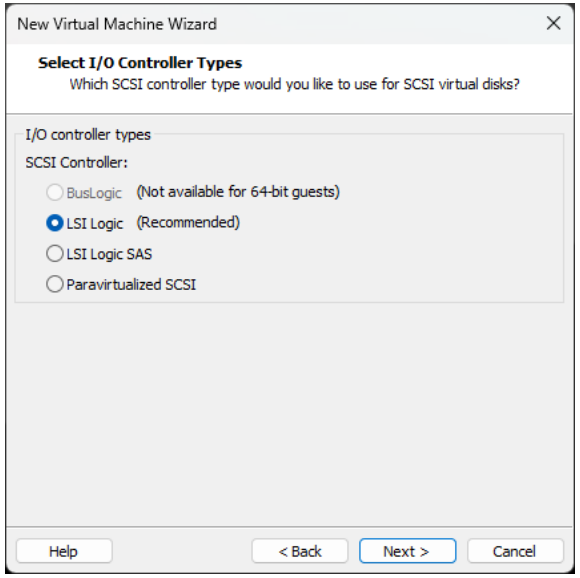


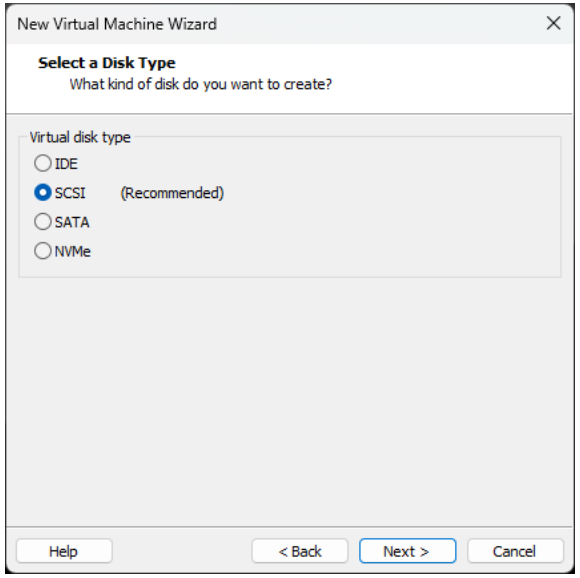


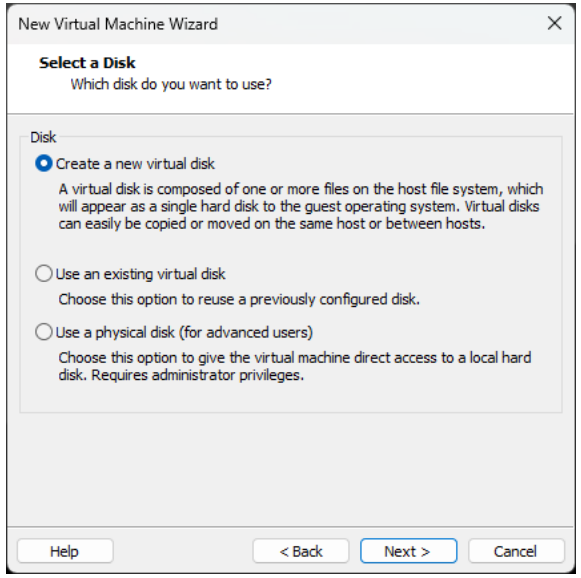


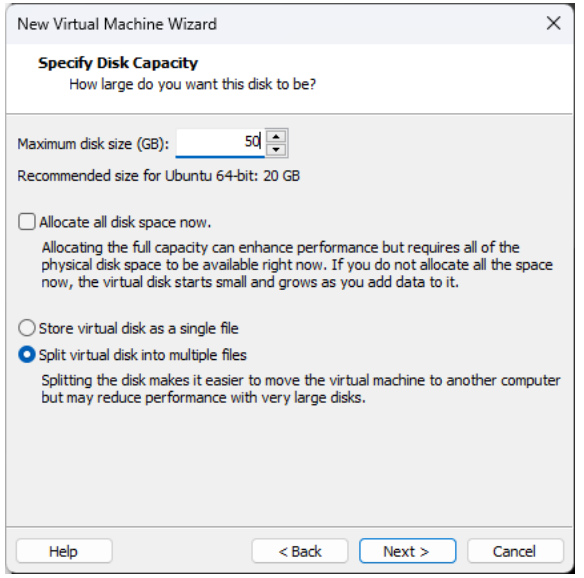


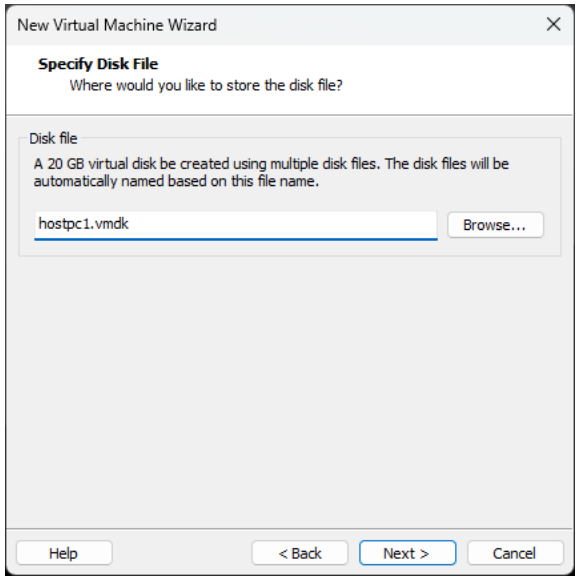
Docker Virtual Machine Create

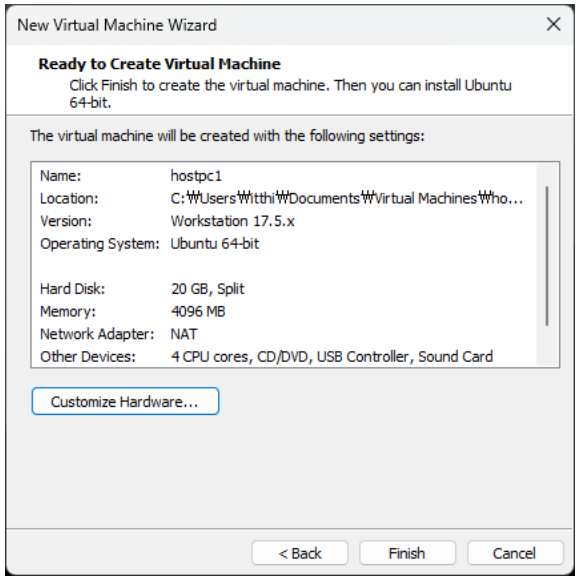


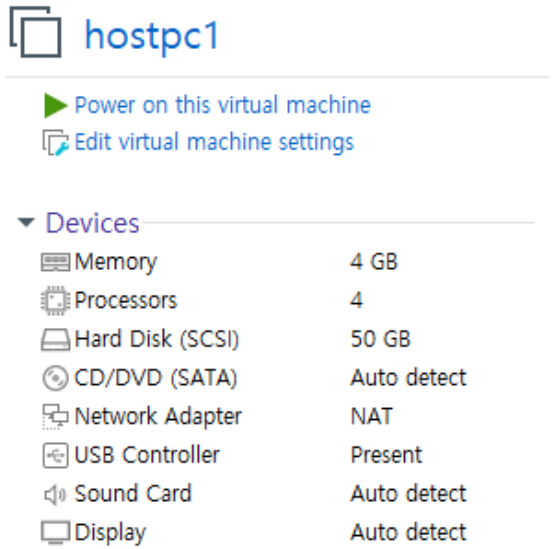






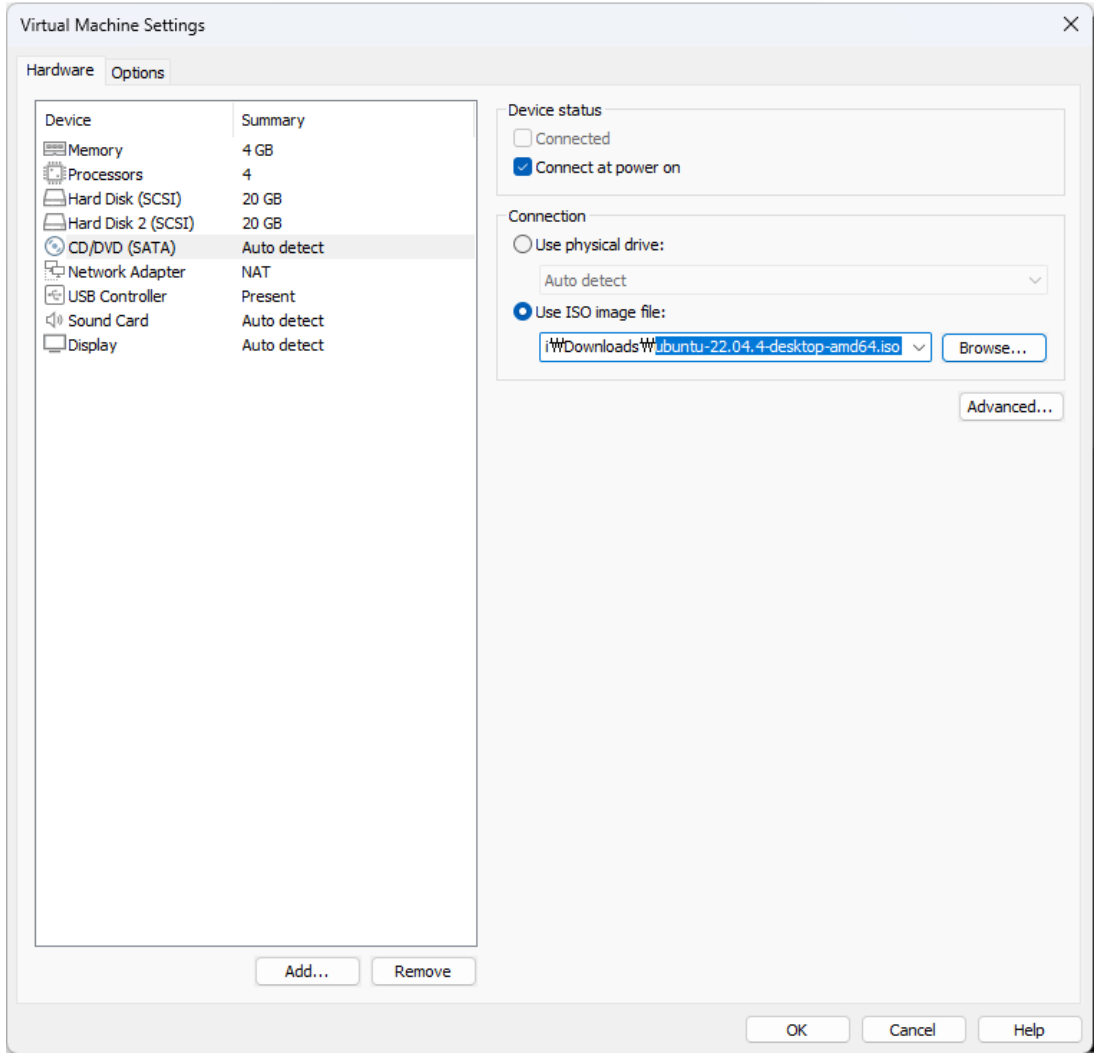






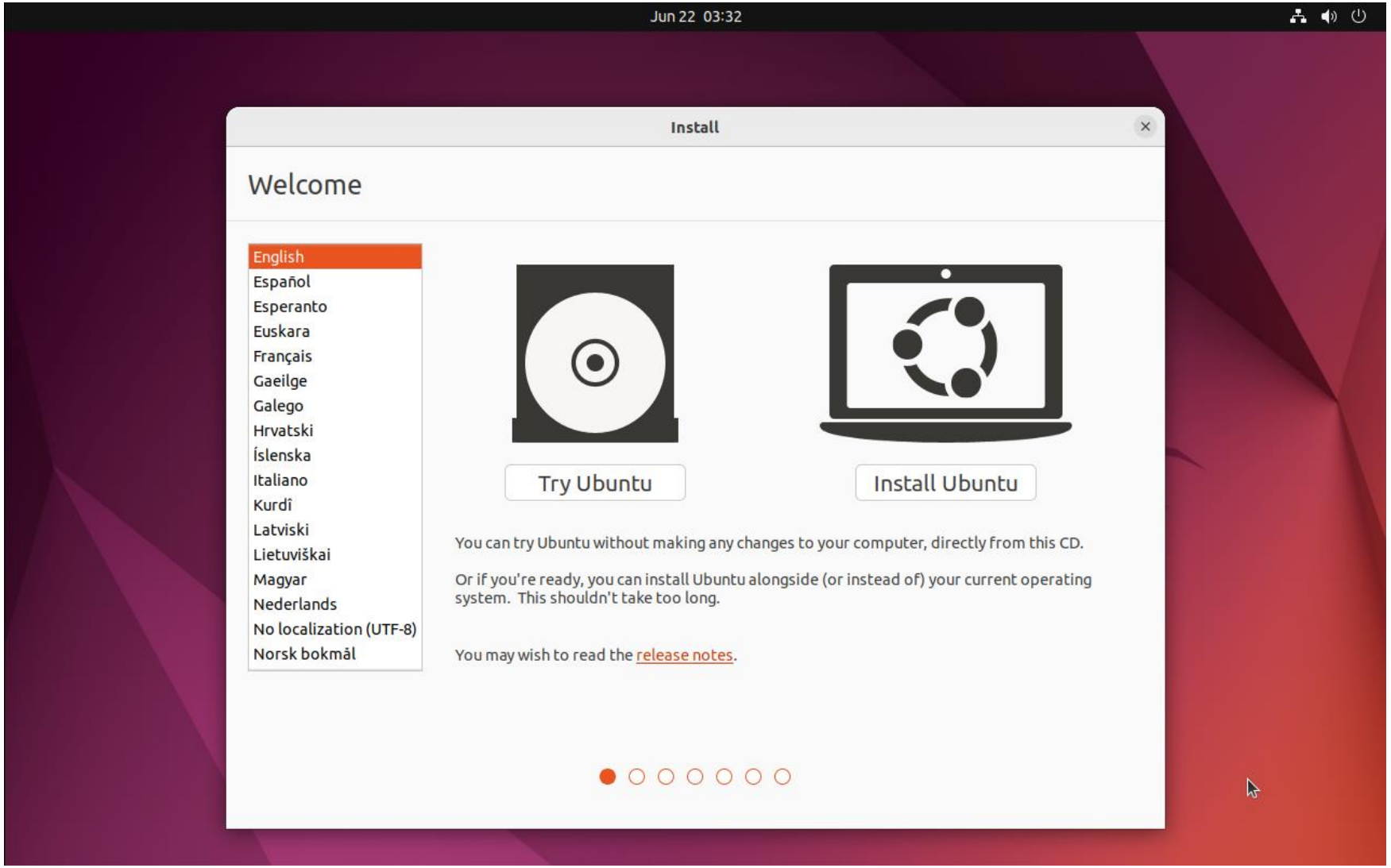
Docker

- Ubuntu 22.04 Install(ISO 우분투 이미지 삽입 후 부팅)



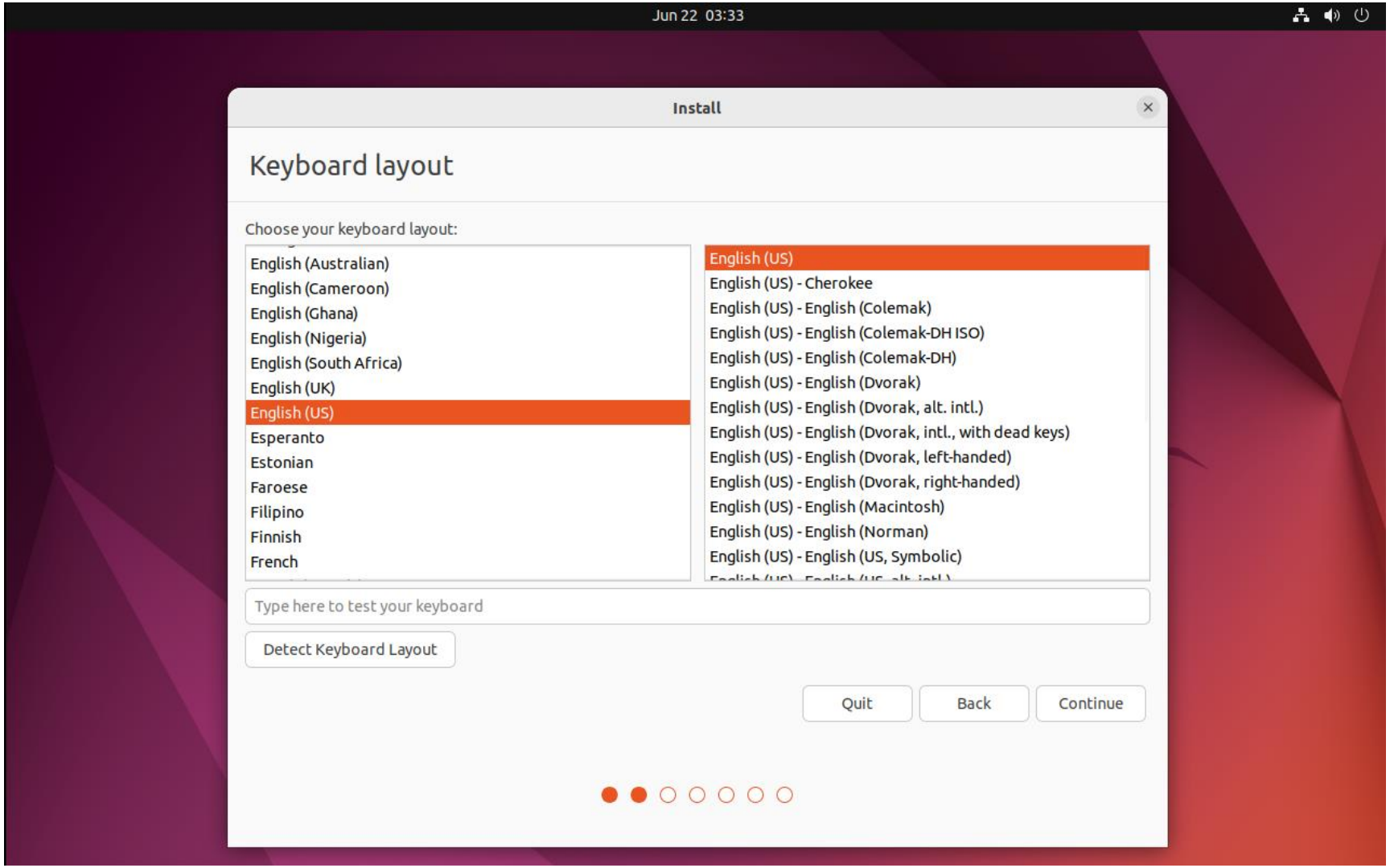
Docker

- 언어 설정(영문으로)



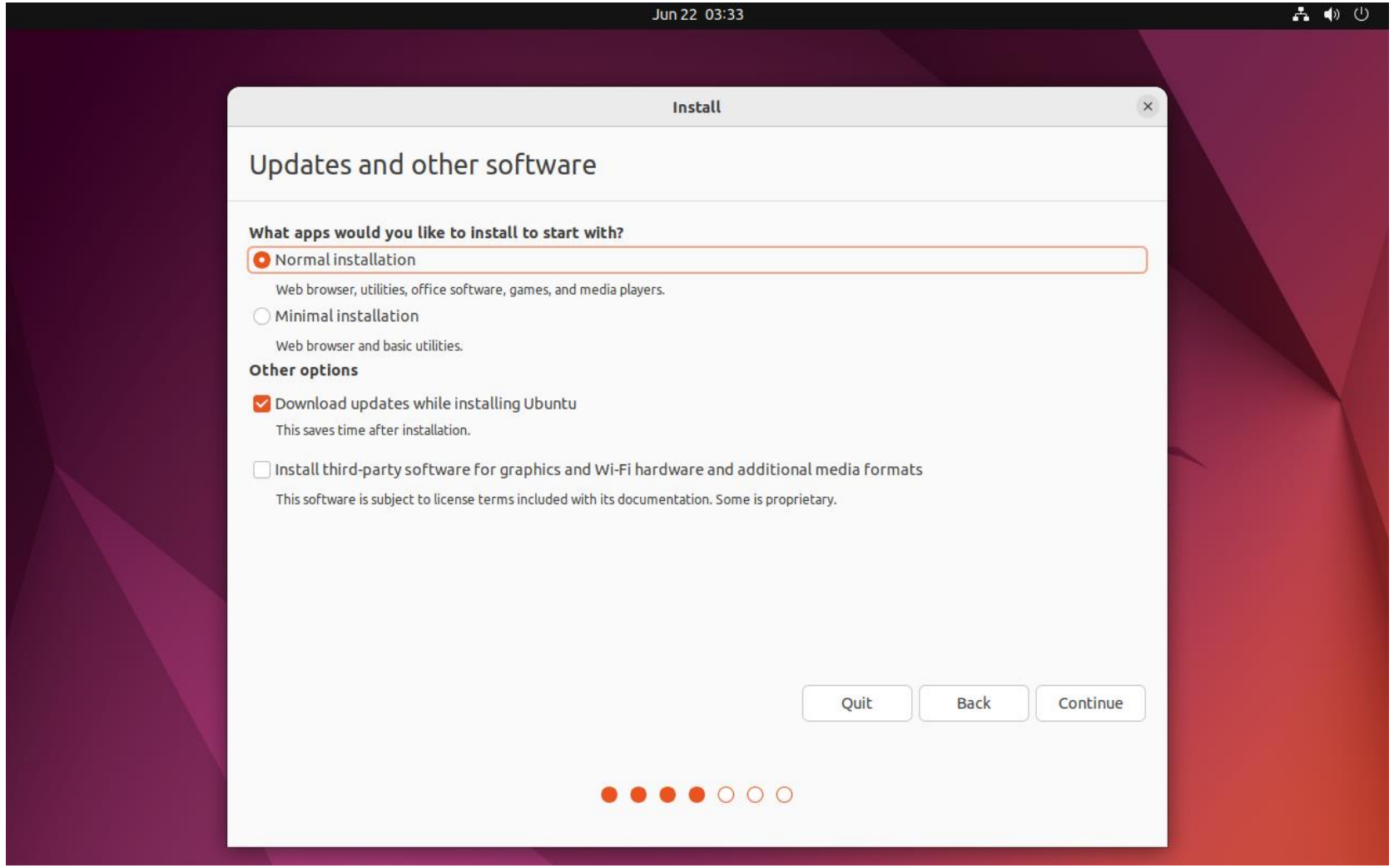
Docker

◆ 키보드 설정(영문)



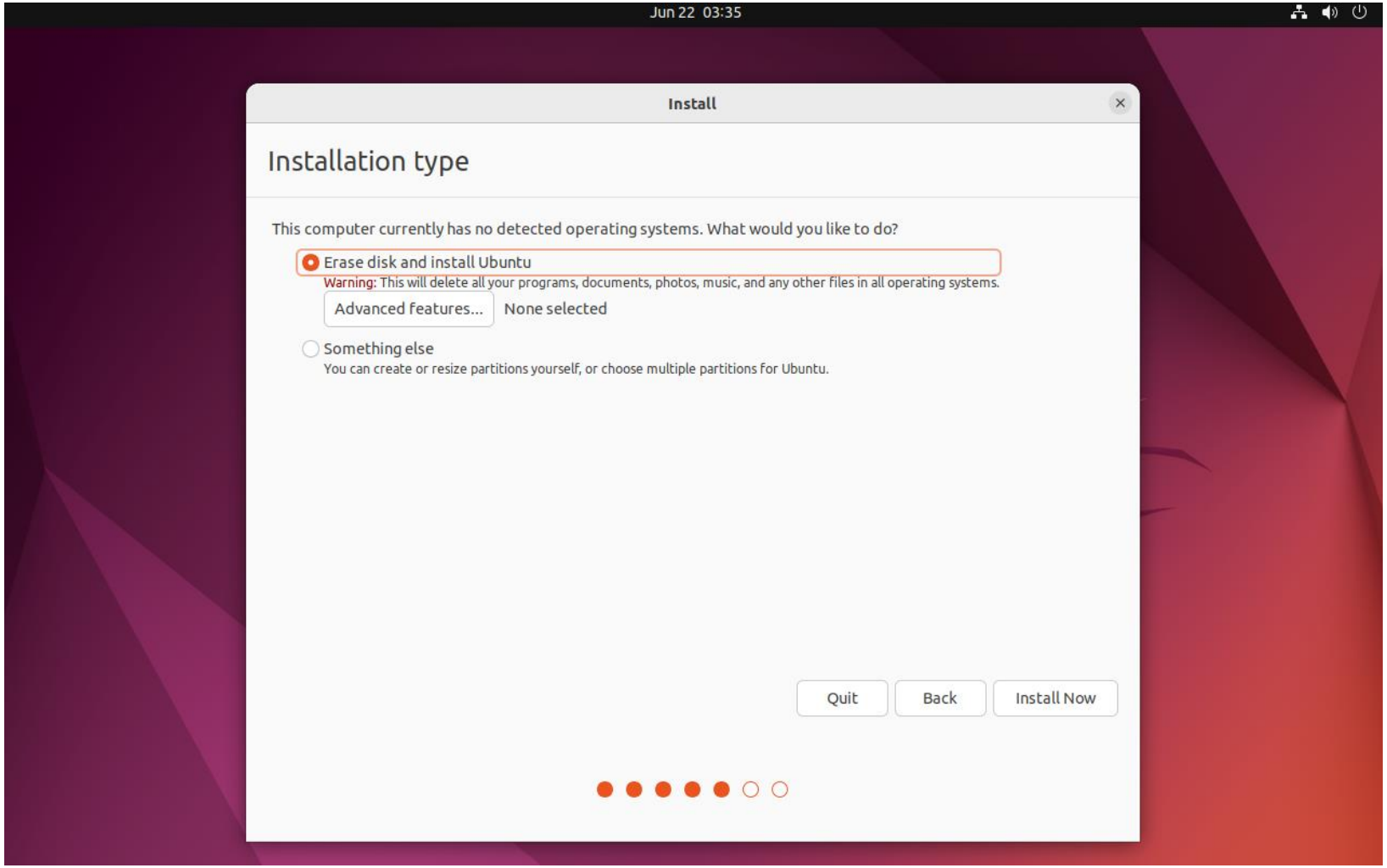
Docker

- 소프트웨어 업데이트 관련 설정



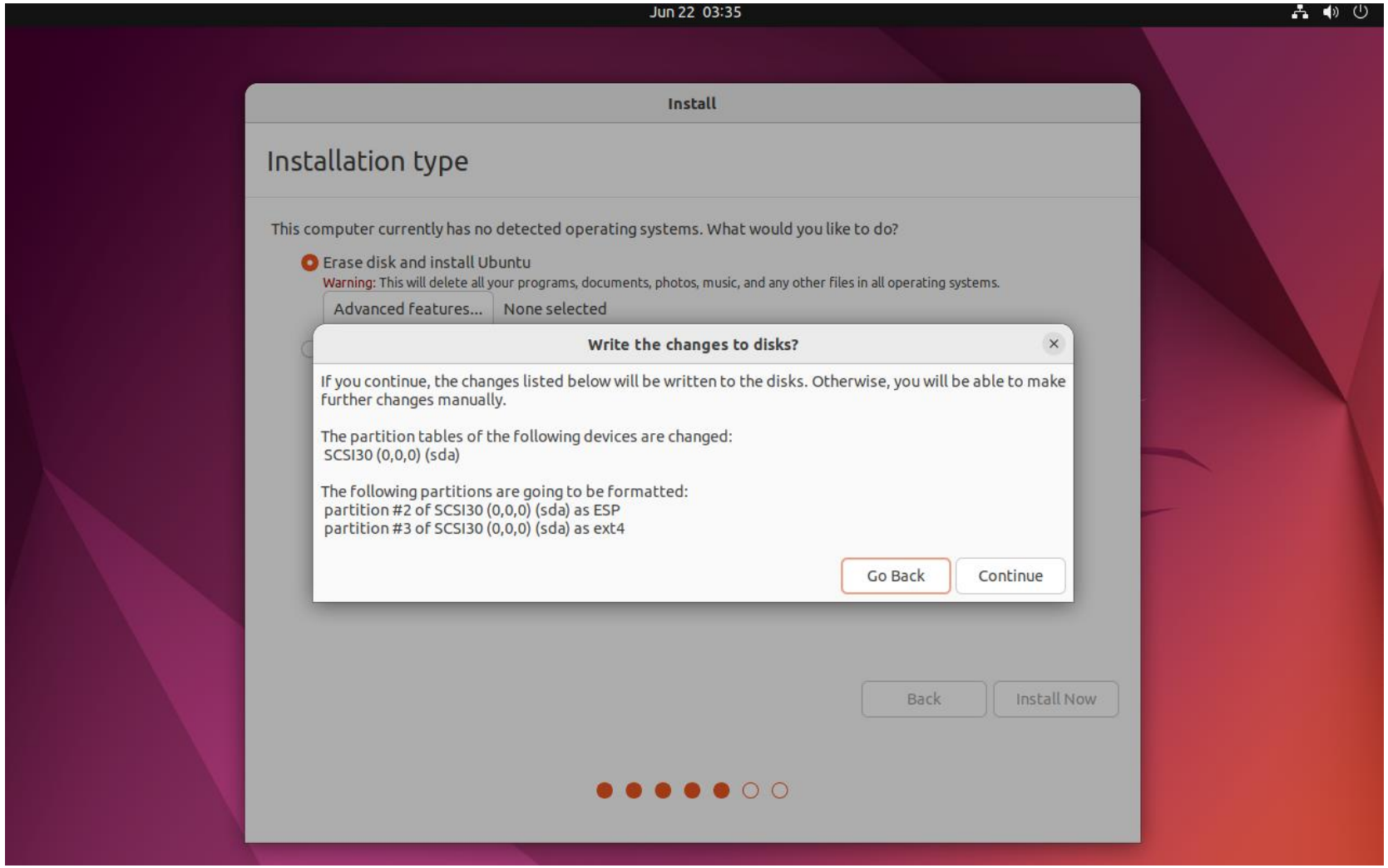
Docker

- 파티션 구성(여기서는 편의를 위해 자동)
  - 실제 사용할 도커 서버를 만드는 경우 파티션 설정 직접 진행(별도의 docker용 HDD추가 권장)



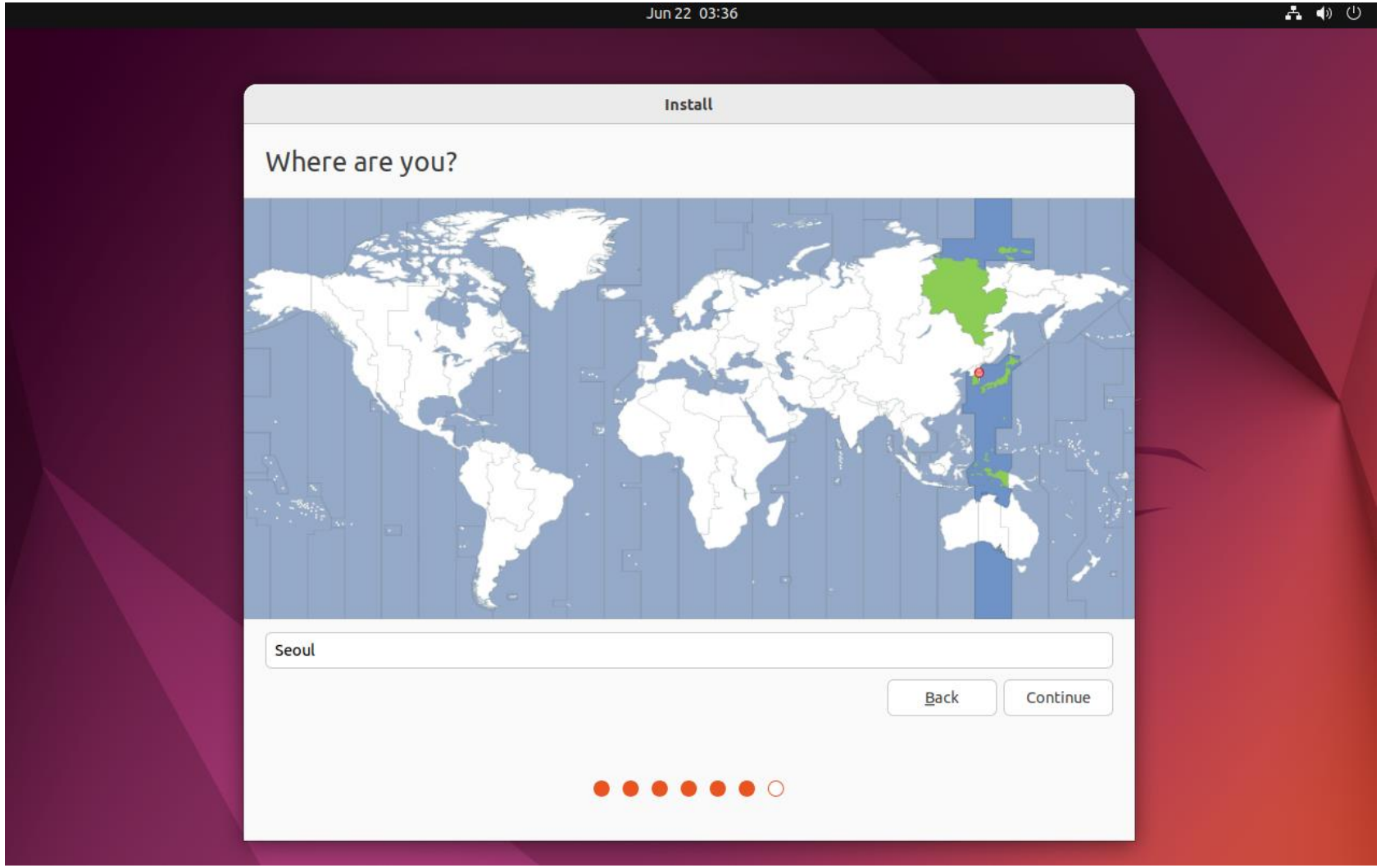
Docker

- 확인


Jun 22 03:35
Install
Installation type
This computer currently has no detected operating systems. What would you like to do?
Erase disk and install Ubuntu
Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.
Advanced features...
None selected
Write the changes to disks?
If you continue, the changes listed below will be written to the disks. Otherwise, you will be able to make further changes manually.
The partition tables of the following devices are changed:
SCSI30 (0,0,0) (sda)
The following partitions are going to be formatted:
partition #2 of SCSI30 (0,0,0) (sda) as ESP
partition #3 of SCSI30 (0,0,0) (sda) as ext4
Go Back
Continue
Back
Install Now

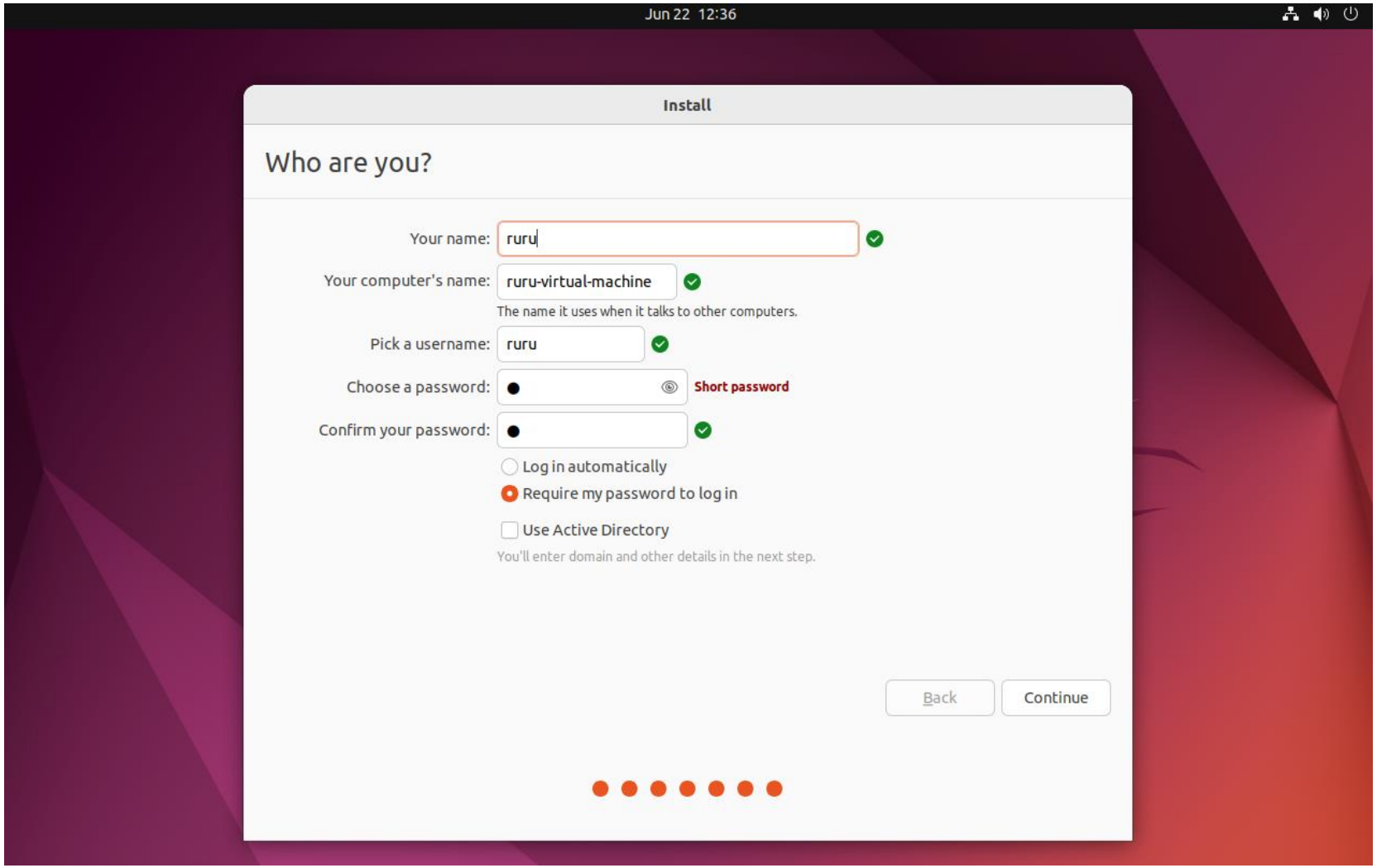
Docker

- 지역/시간대 설정



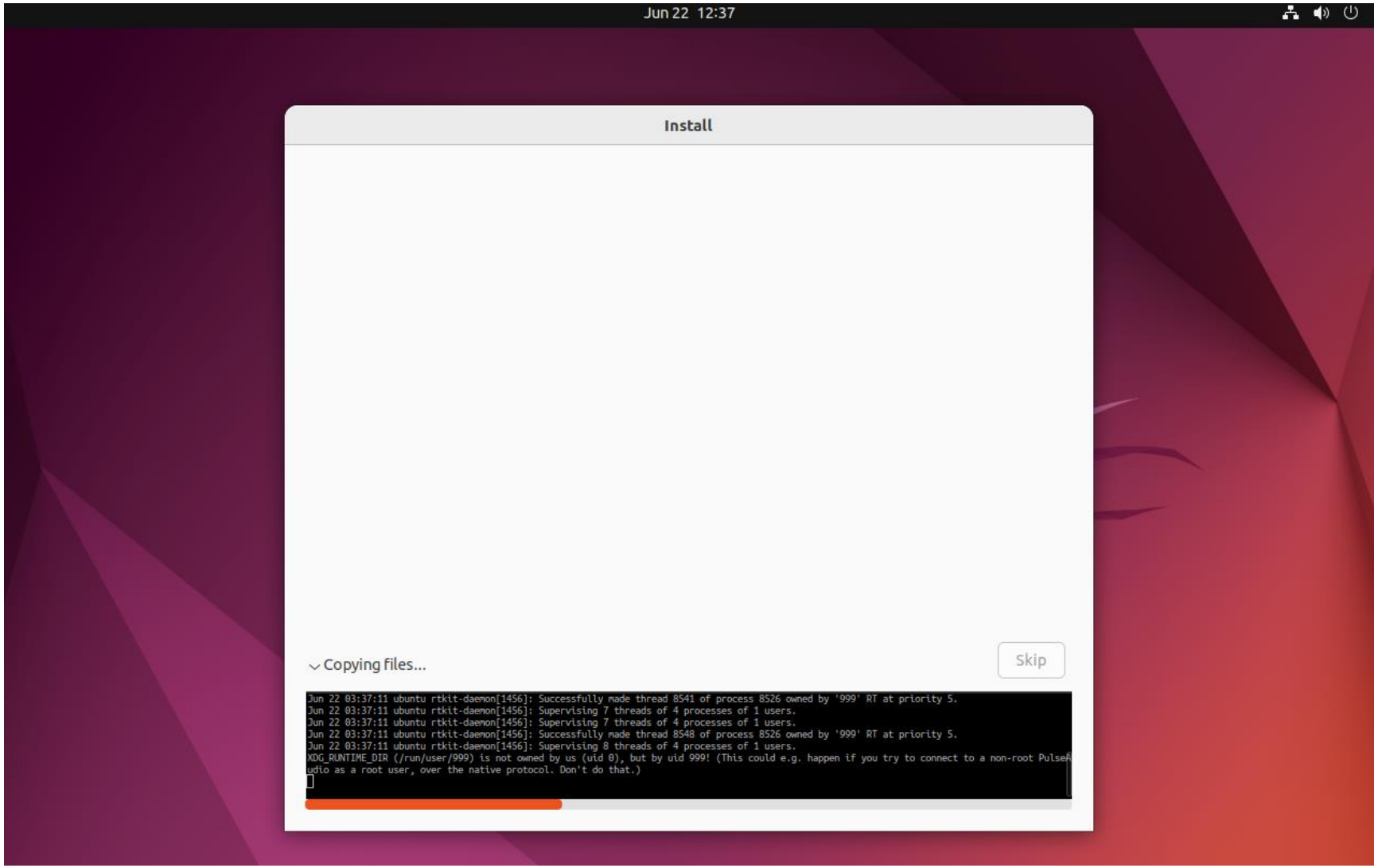
Docker

- 일반 사용자 생성



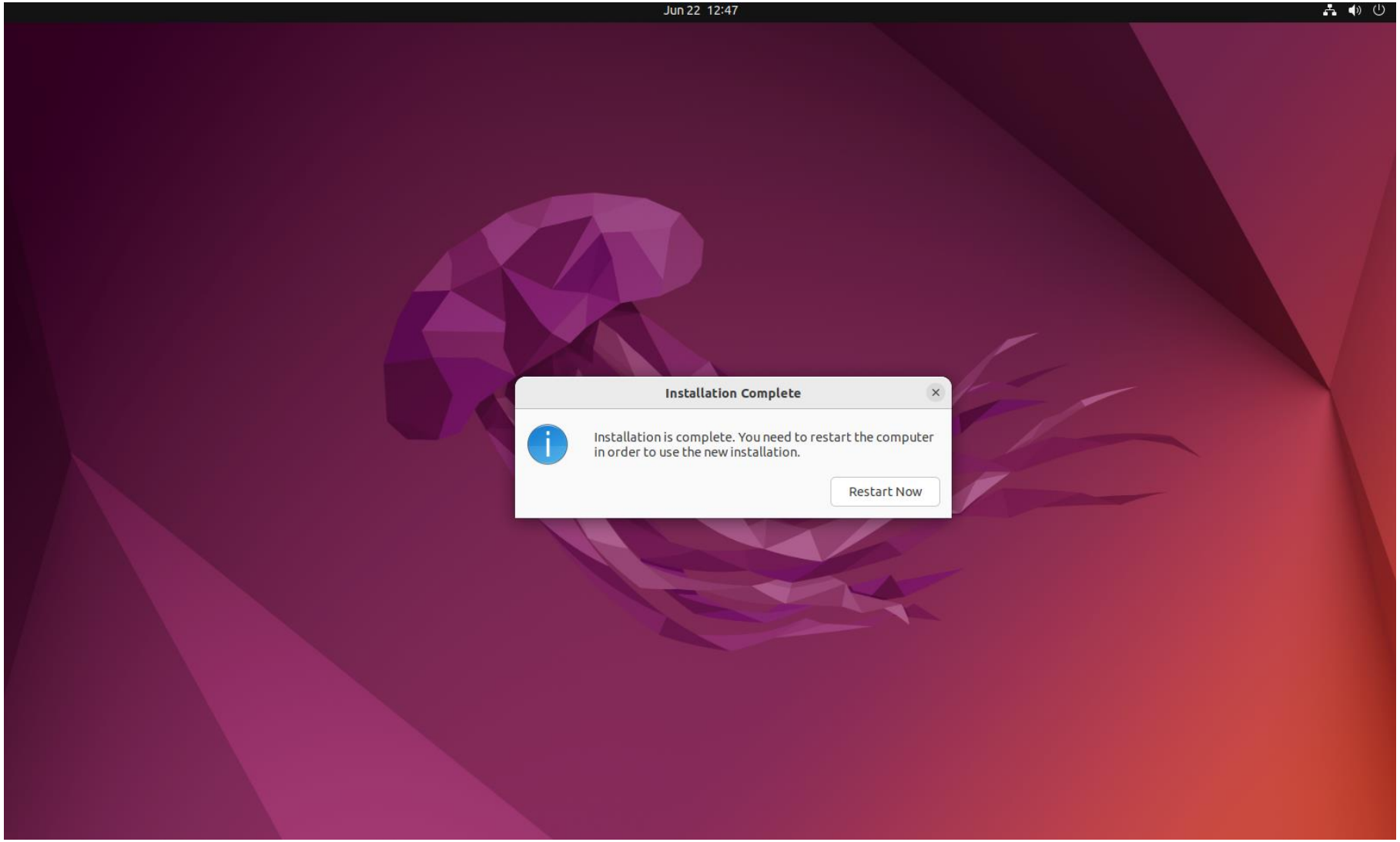
Docker

- Install..



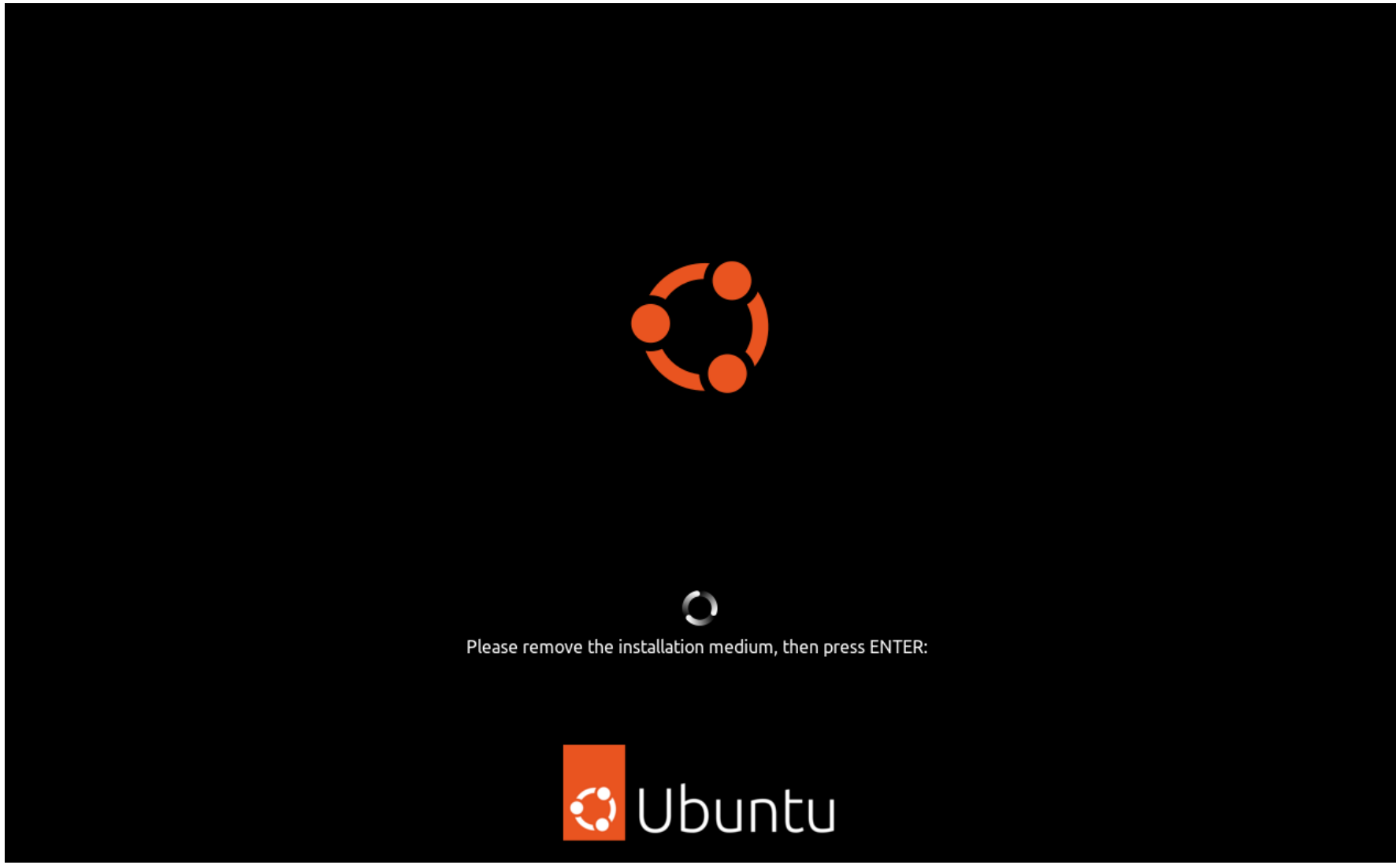
Docker

- Restart Now



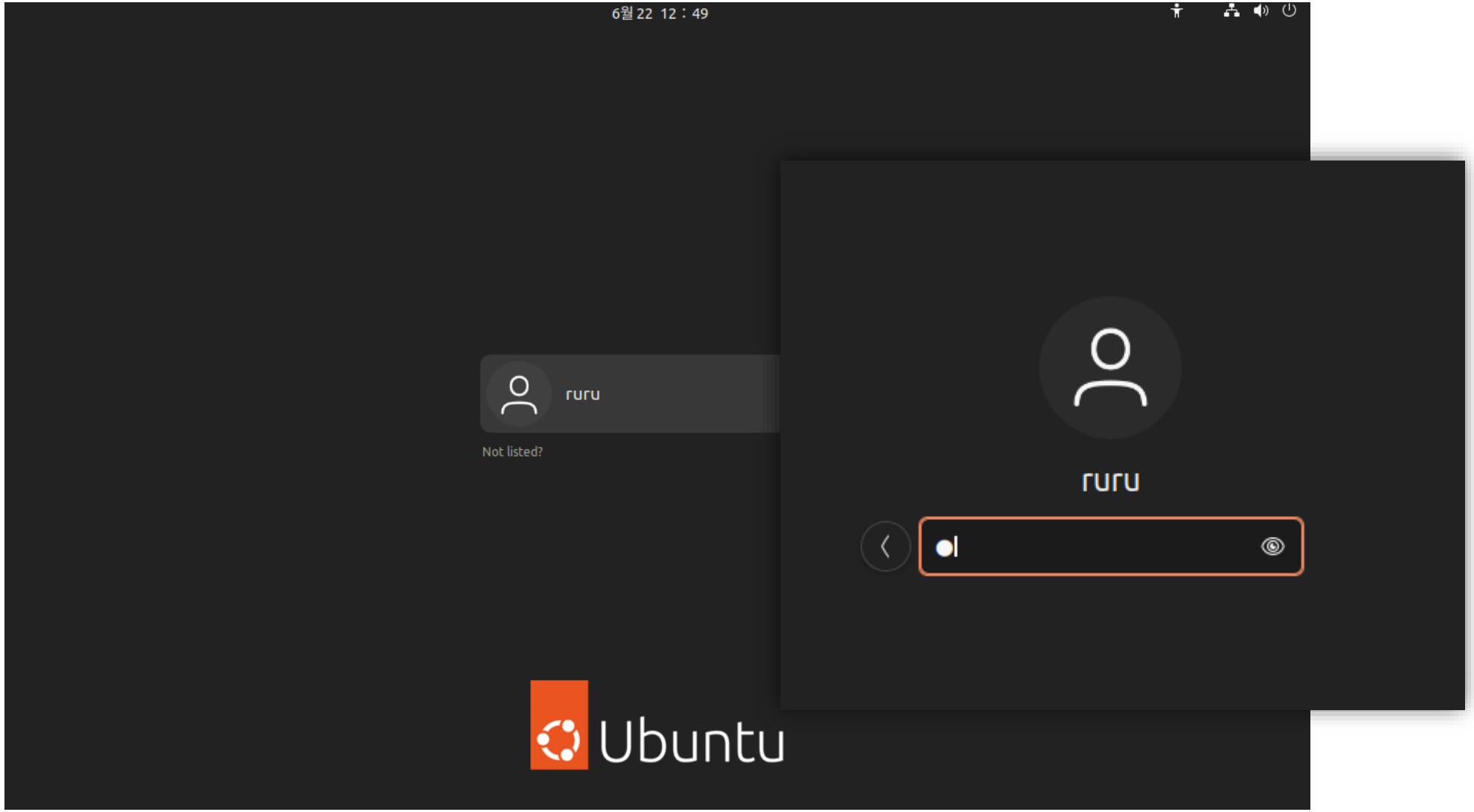
Docker

- ENTER 입력



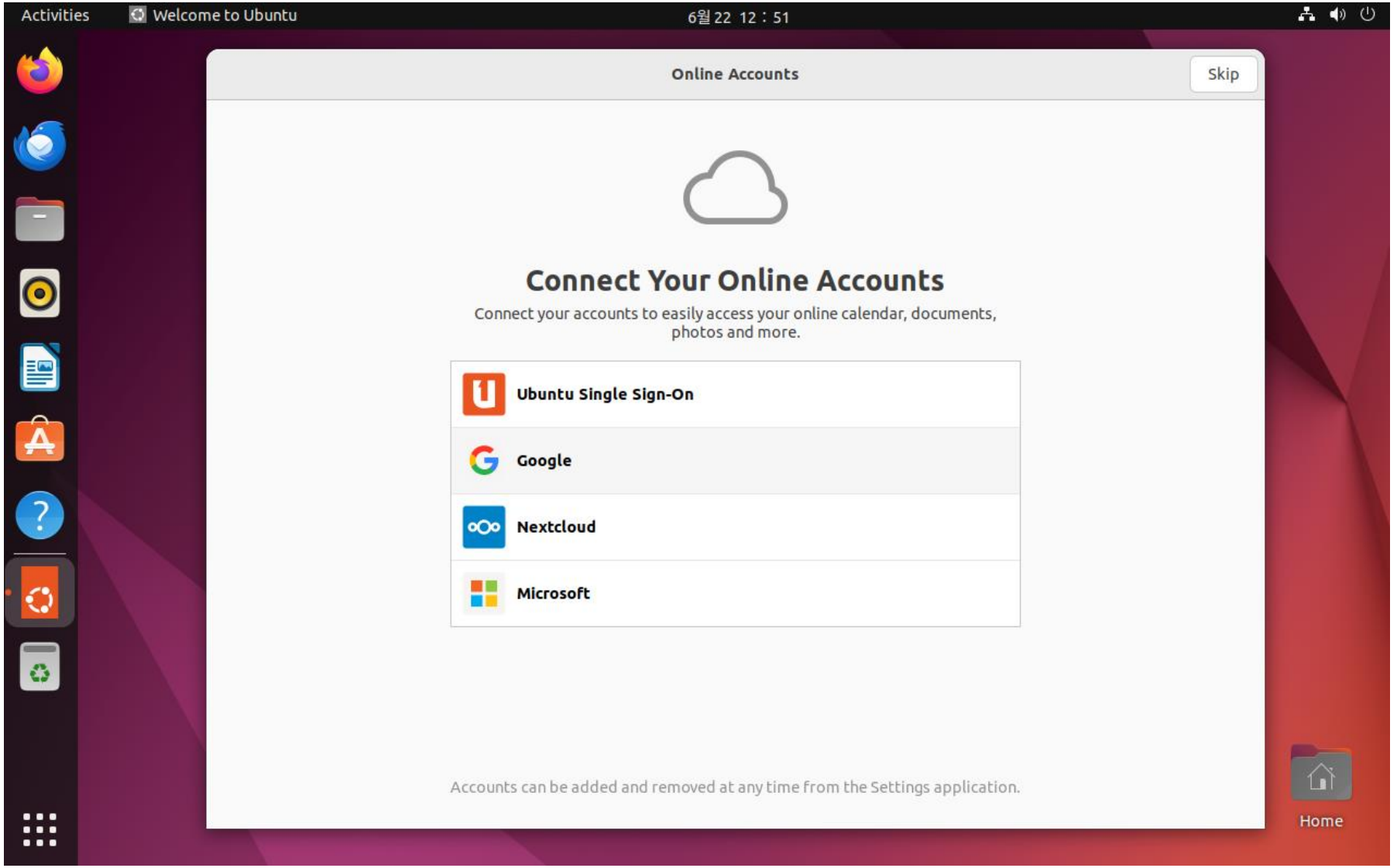
Docker

- 로그인



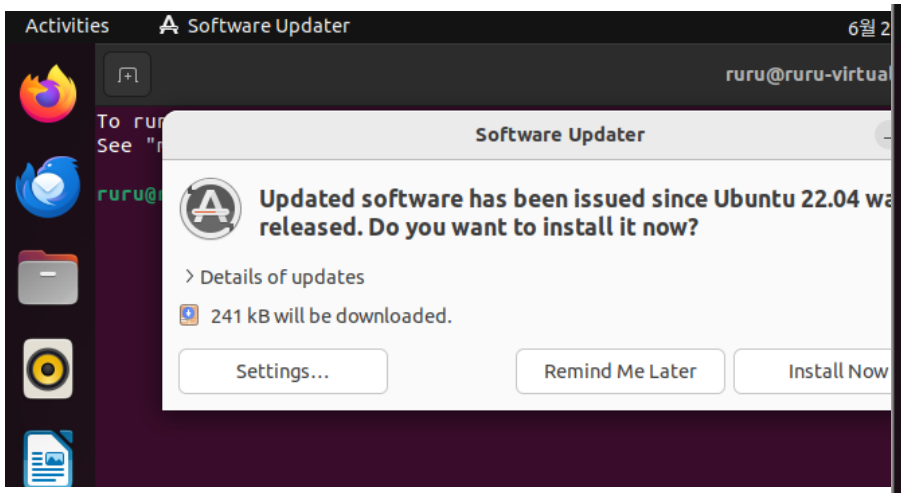
Docker

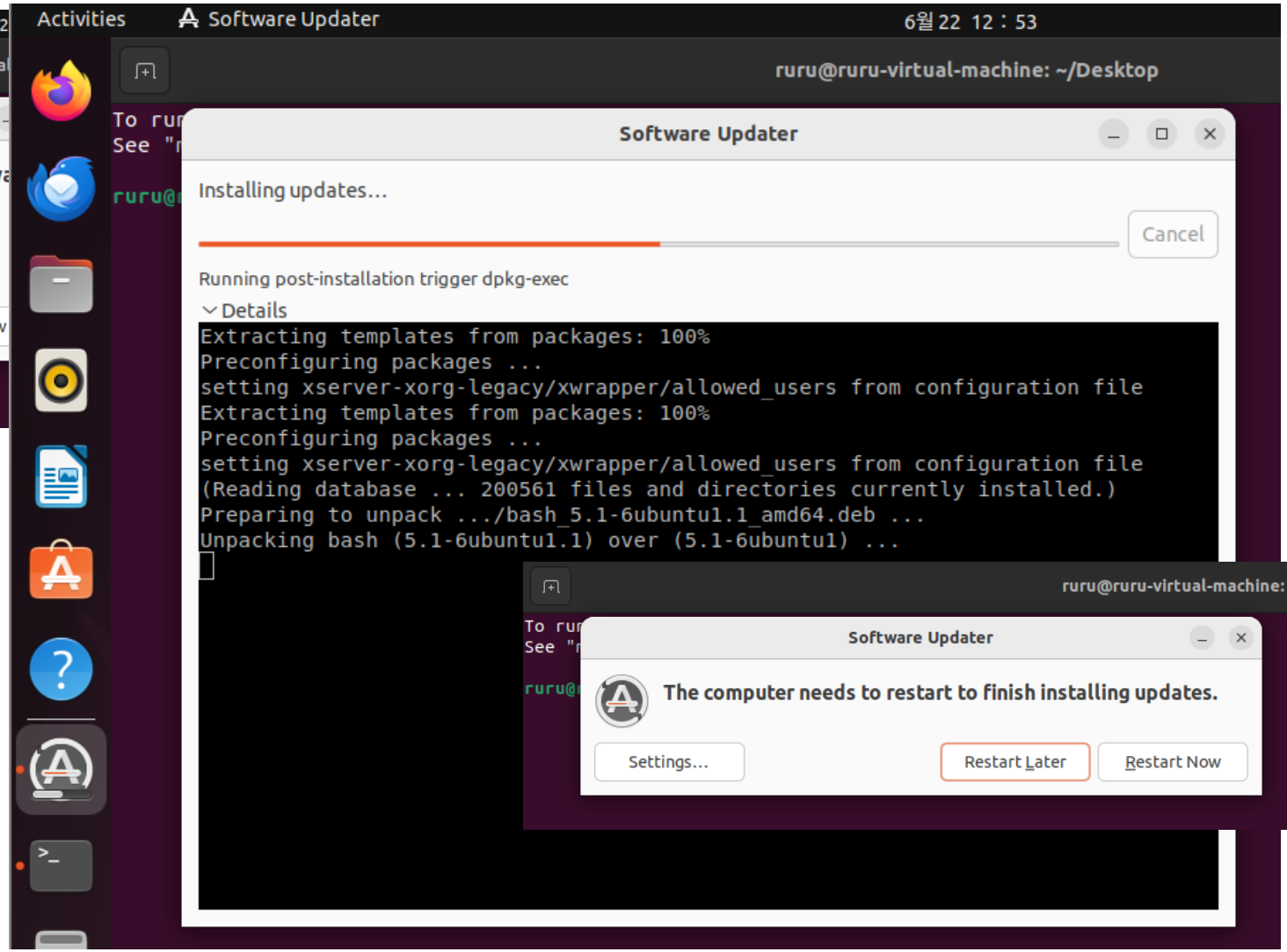
- Skip


Activities
Welcome to Ubuntu
6월 22 12 : 51
Online Accounts
Skip
Connect Your Online Accounts
Connect your accounts to easily access your online calendar, documents, photos and more.
Ubuntu Single Sign-On
Google
Nextcloud
Microsoft
Accounts can be added and removed at any time from the Settings application.
Home

Docker

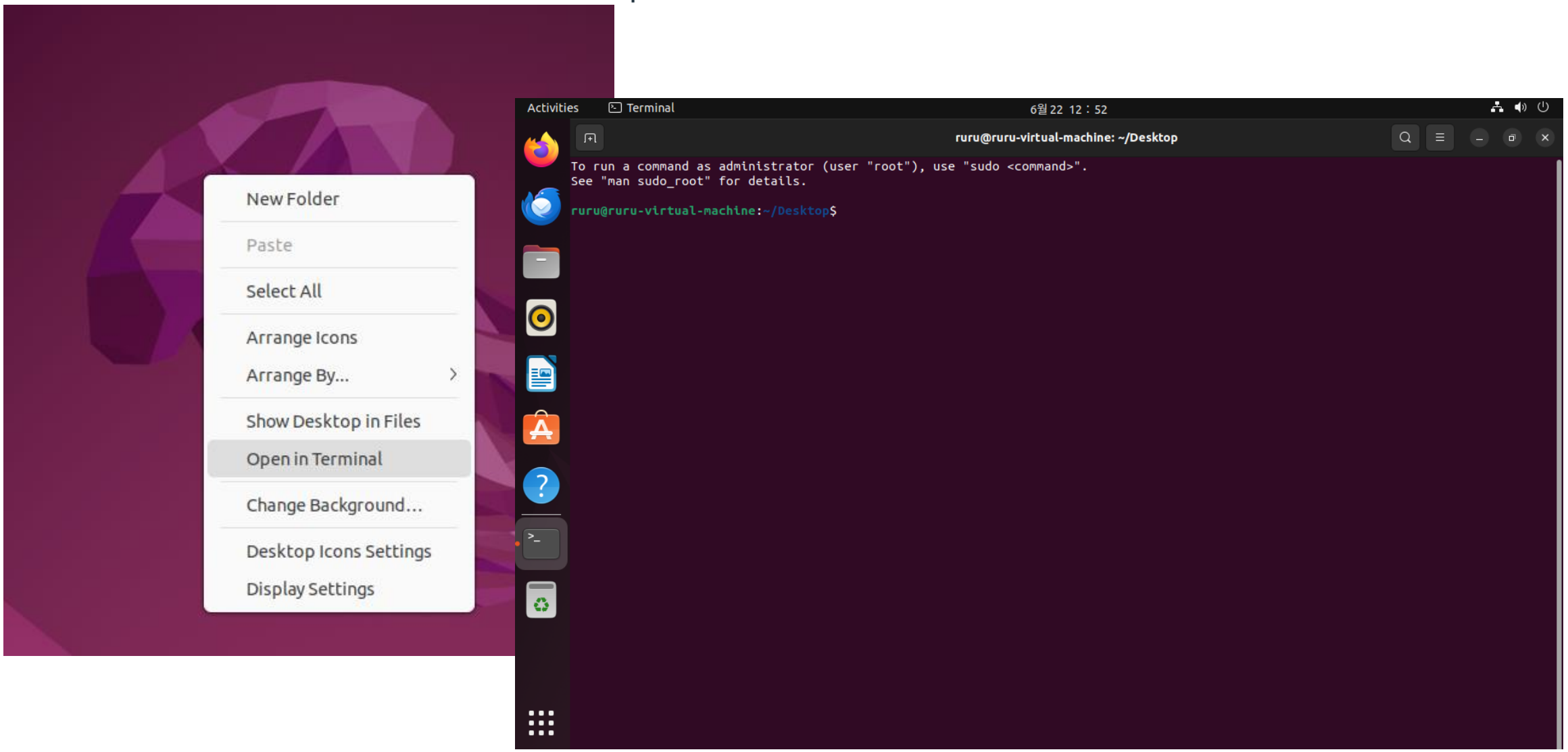
- Software Update –> Install Now –> Updated –> Restart Now





Docker

- Reboot 후 바탕화면에 우 클릭 하여 Open Terminal 선택



Docker Install

◆ Docker Engine Install을 위한 작업

- https://docs.docker.com/engine/install/

```
   # Add Docker's official GPG key:
>  sudo apt-get update
>  sudo apt-get install 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

   # Add the repository to Apt sources:
>  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
```

Docker Install

◆ 시스템의 패키지 목록을 최신 상태로 업데이트

```
ruru@ruru-virtual-machine:~/Desktop$ sudo apt-get update
[sudo] password for ruru:
Hit:1 http://kr.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://kr.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://kr.archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
```

ca-certificates
HTTPS 연결을 위한 인증서를 관리하는 패키지

curl
URL을 통해 데이터를 전송할 수 있는 도구

◆ 필수 패키지 설치

```
ruru@ruru-virtual-machine:~/Desktop$ sudo apt-get install ca-certificates curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20230311ubuntu0.22.04.1).
ca-certificates set to manually installed.
The following packages were automatically installed and are no longer required:
  libwpe-1.0-1 libwpebackend-fdo-1.0-1
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  curl
0 upgraded, 1 newly installed, 0 to remove and 3 not upgraded.
Need to get 194 kB of archives.
After this operation, 454 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://kr.archive.ubuntu.com/ubuntu jammy-updates/main amd64 curl amd64 7.81.0-1ubuntu1.16 [194 kB]
Fetched 194 kB in 2s (97.9 kB/s)
Selecting previously unselected package curl.
(Reading database ... 200690 files and directories currently installed.)
Preparing to unpack .../curl_7.81.0-1ubuntu1.16_amd64.deb ...
Unpacking curl (7.81.0-1ubuntu1.16) ...
Setting up curl (7.81.0-1ubuntu1.16) ...
Processing triggers for man-db (2.10.2-1) ...
```

Docker Install

- /etc/apt/keyrings 디렉토리를 생성하고, 접근 권한을 설정
- 도커의 공식 GPG 키를 다운로드하여 /etc/apt/keyrings/docker.asc에 저장
- 모든 사용자에게 GPG 키 파일에 대한 읽기 권한을 부여

```
ruru@ruru-virtual-machine:~/Desktop$ sudo install -m 0755 -d /etc/apt/keyrings
ruru@ruru-virtual-machine:~/Desktop$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
ruru@ruru-virtual-machine:~/Desktop$ sudo chmod a+r /etc/apt/keyrings/docker.asc
```

- 저장소를 Apt 소스 목록에 추가(Docker 저장소를 /etc/apt/sources.list.d/docker.list에 추가)

```
ruru@ruru-virtual-machine:~/Desktop$ 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
```

dpkg --print-architecture: 시스템의 아키텍처를 출력 (예: amd64).
. /etc/os-release && echo "$VERSION_CODENAME": Ubuntu 버전 코드 가져오기 (예: focal).
signed-by=/etc/apt/keyrings/docker.asc: 다운로드된 GPG 키로 서명된 패키지들만 신뢰

- 새로운 Docker 저장소를 포함한 패키지 목록을 업데이트

```
ruru@ruru-virtual-machine:~/Desktop$ sudo apt-get update
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://kr.archive.ubuntu.com/ubuntu jammy InRelease
Get:4 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [33.7 kB]
Hit:5 http://kr.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:6 http://kr.archive.ubuntu.com/ubuntu jammy-backports InRelease
Fetched 82.5 kB in 1s (59.8 kB/s)
Reading package lists... Done
```

Docker Install

◆ Docker Engine Install

```
ruru@ruru-virtual-machine:~/Desktop$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libwpe-1.0-1 libwpebackend-
Use 'sudo apt autoremove' to
The following additional pack
  docker-ce-rootless-extras g
Suggested packages:
  aufs-tools cgroupfs-mount |
  git-mediawiki git-svn
The following NEW packages wi
  containerd.io docker-buildx
  liberror-perl libslirp0 pig
0 upgraded, 12 newly installe
Need to get 126 MB of archive
After this operation, 455 MB
Do you want to continue? [Y/n
Get:1 https://download.docker
Get:2 http://kr.archive.ubunt
Get:3 https://download.docker
Get:4 https://download.docker
Get:5 http://kr.archive.ubunt
Get:6 http://kr.archive.ubunt
```

docker-ce (Docker Community Edition)

Docker의 커뮤니티 에디션으로, Docker 데몬 및 관련 도구들을 포함
Docker 컨테이너를 관리하고 실행하는 핵심 소프트웨어

docker-ce-cli (Docker Command-Line Interface)

Docker CLI는 명령줄에서 Docker를 관리하기 위한 도구
사용자는 이를 통해 컨테이너를 시작, 중지, 관리할 수 있음

containerd.io

Containerd는 컨테이너의 생명주기를 관리하는 컨테이너 런타임
Docker 데몬이 컨테이너를 관리하기 위해 containerd를 사용

docker-buildx-plugin

Docker Buildx는 Docker CLI의 확장 기능으로, 멀티 플랫폼 빌드를 지원
이를 통해 다양한 아키텍처에 맞는 이미지를 빌드할 수 있음

docker-compose-plugin

Docker Compose는 멀티 컨테이너 Docker 애플리케이션을 정의하고 실행하기 위한 도구
Docker Compose 파일을 사용하여 여러 컨테이너를 한번에 설정하고 실행할 수 있음

Docker Install

◆ 설치 확인

- 버전 확인

```
ruru@ruru-virtual-machine:~/Desktop$ sudo docker --version
Docker version 26.1.4, build 5650f9b
```

- Docker Status

```
ruru@ruru-virtual-machine:~/Desktop$ sudo systemctl status docker
[sudo] password for ruru:
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2024-06-22 13:21:53 KST; 16min ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 976 (dockerd)
      Tasks: 9
     Memory: 110.2M
        CPU: 1.578s
     CGroup: /system.slice/docker.service
             └─976 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

 6월  22 13:21:49 ruru-virtual-machine dockerd[976]: time="2024-06-22T13:21:49.931955367+09:00" level=info msg="Starting up"
 6월  22 13:21:50 ruru-virtual-machine dockerd[976]: time="2024-06-22T13:21:50.010470405+09:00" level=info msg="detected 127.0.0.53 >
 6월  22 13:21:51 ruru-virtual-machine dockerd[976]: time="2024-06-22T13:21:51.363404325+09:00" level=info msg="[graphdriver] using >
 6월  22 13:21:51 ruru-virtual-machine dockerd[976]: time="2024-06-22T13:21:51.403734266+09:00" level=info msg="Loading containers: >
 6월  22 13:21:53 ruru-virtual-machine dockerd[976]: time="2024-06-22T13:21:53.197217226+09:00" level=info msg="Default bridge (dock>
 6월  22 13:21:53 ruru-virtual-machine dockerd[976]: time="2024-06-22T13:21:53.451200517+09:00" level=info msg="Loading containers: >
 6월  22 13:21:53 ruru-virtual-machine dockerd[976]: time="2024-06-22T13:21:53.633100529+09:00" level=info msg="Docker daemon" commi>
 6월  22 13:21:53 ruru-virtual-machine dockerd[976]: time="2024-06-22T13:21:53.634003953+09:00" level=info msg="Daemon has completed>
 6월  22 13:21:53 ruru-virtual-machine dockerd[976]: time="2024-06-22T13:21:53.927281915+09:00" level=info msg="API listen on /run/d>
 6월  22 13:21:53 ruru-virtual-machine systemd[1]: Started Docker Application Container Engine.
```

Docker Install

- Docker Hello World

```
ruru@docker-master:~/Desktop$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:266b191e926f65542fa8daaec01a192c4d292bff79426f47300a046e1bc576fd
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Docker Install

- Docker 명령을 편하게 사용하기 위해 그룹 권한 설정

```
ruru@ruru-virtual-machine:~/Desktop$ cat /etc/group | grep docker
docker:x:999:
ruru@ruru-virtual-machine:~/Desktop$ sudo usermod -aG docker ruru
ruru@ruru-virtual-machine:~/Desktop$ cat /etc/group | grep docker
docker:x:999:ruru
```

  - reboot해야 적용 됨

```
ruru@ruru-virtual-machine:~/Desktop$ docker ps
permission denied while trying to connect to the Docker daemon socket at unix://
/var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.45/containers/js
on": dial unix /var/run/docker.sock: connect: permission denied
ruru@ruru-virtual-machine:~/Desktop$ reboot
```

- 그룹 권한 설정 전(docker 명령 사용 시 sudo 필요)

```
ruru@ruru-virtual-machine:~/Desktop$ docker ps
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fva
r%2Frun%2Fdocker.sock/v1.45/containers/json": dial unix /var/run/docker.sock: connect: permission denied
ruru@ruru-virtual-machine:~/Desktop$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

- 권한 설정 후(sudo 없이 docker 명령 사용 가능)

```
ruru@ruru-virtual-machine:~/Desktop$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```