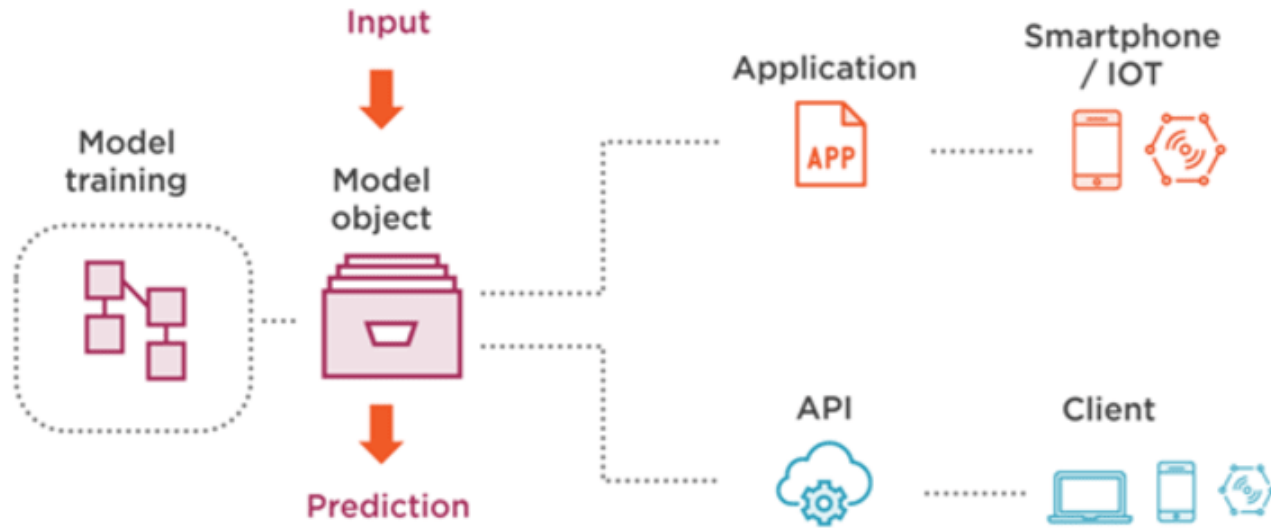


MLflow 모델 서빙

모델 서빙

모델 서빙

MLOps에서 모델 서빙은 머신러닝 모델을 사용자와 상호작용할 수 있도록 시스템을 구축하는 것을 의미합니다. 모델의 성능, 안정성, 신뢰성을 보장하고, 실시간으로 모델을 업데이트하고 모델의 예측 결과를 모니터링할 수 있도록 합니다.



■ 온라인 서빙

- 클라이언트가 ML모델 서버에 HTTP를 통해 요청 → ML모델 서버가 예측값 생성 → 생성한 예측값을 반환
- 요청(Request)이 들어올 때마다 실시간으로 예측 및 반환(Response)

■ 배치 서빙

- 모델이 주기적으로(Batch) 학습 및 예측을 수행
- workflow scheduler(Airflow, Cron Job 등)를 통해 모델의 작업 스케줄을 관리

모델 서빙 종류

종류	회사	모델 포맷	속도	장단점
TensorFlow Serving	Google	TensorFlow, Keras, SavedModel	높음	분산 시스템을 지원하며, TensorFlow 모델을 위해 최적화 되어있음.
TorchServe	Facebook	PyTorch, TorchScript	중간	PyTorch 모델을 위해 최적화 되어있으며, 다양한 모델 포맷을 지원.
Triton Inference Server	Nvidia	TensorFlow, TensorRT, ONNX	매우 높음	다양한 모델 포맷을 지원하며, GPU 가속을 통해 빠른 서빙 가능.
BentoML	BentoML	다양	중간	모델 서빙뿐만 아니라, 모델 빌드, 테스트, 배포를 종합적으로 제공. 다양한 프레임워크와 클라우드 서비스를 지원하며, 코드 중복을 최소화할 수 있음.
MLflow	Databricks	다양	중간	모델 학습, 추적, 서빙을 포함한 엔드 투 엔드 기계 학습 플랫폼. 다양한 프레임워크와 클라우드 서비스를 지원하며, 모델 버전 관리와 협업 기능을 제공.

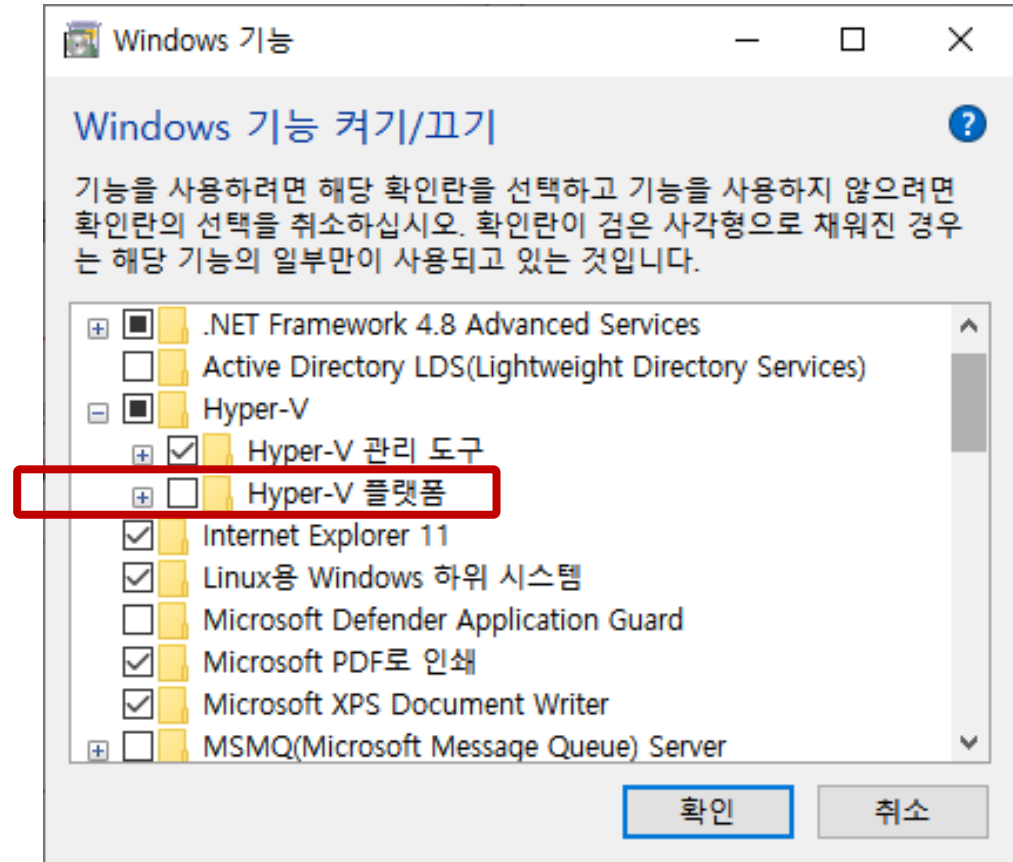
로컬서버 구축

Windows : Hyper-V 비활성화

대부분의 가상화 애플리케이션은 윈도우즈 Hyper-V와 호환되지 않습니다.

제어판에서 Hyper-V를 사용하지 않도록 설정

1. 제어판에서 **프로그램 및 기능**을 선택합니다.
2. **Windows 기능 켜기/끄기**를 선택합니다.
3. **Hyper-V**를 확장하고 **Hyper-V 플랫폼**을 확장한 다음
Hyper-V 하이퍼바이저 확인란의 **선택**을 취소합니다.



VirtualBox 설치

<https://www.virtualbox.org/>



Oracle VM VirtualBox

virtualbox.org

VirtualBox

Welcome to VirtualBox.org!

VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. No VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only solution that is freely available as Open Source Software under the terms of the GNU General Public License 3. See "About VirtualBox" for an introduction.

Presently, VirtualBox runs on Windows, Linux, macOS, and Solaris hosts and supports a large number of guest systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD.

VirtualBox is being actively developed with frequent releases and has an ever growing list of features, support operating systems and platforms it runs on. VirtualBox is a community effort backed by a dedicated company encouraged to contribute while Oracle ensures the product always meets professional quality criteria.

Download VirtualBox 7.0

Hot picks:

- Pre-built virtual machines for developers at [Oracle Tech Network](#)
- Hyperbox** Open-source Virtual Infrastructure Manager [project site](#)
- phpVirtualBox** AJAX web interface [project site](#)

ORACLE

Contact - Privacy policy - Terms of Use

설치파일 다운로드 및 설치

VirtualBox 7.0.14 platform packages

- Windows hosts
- macOS / Intel hosts
- Linux distributions
- Solaris hosts
- Solaris 11 IPS hosts



VirtualBox-7.0.14-161095-Win.exe

VirtualBox 7.0.14 Oracle VM VirtualBox Extension Pack

- All supported platforms



Oracle_VM_VirtualBox_Extension_Pack-7.0.14.vbox-extpack

Support VirtualBox RDP, disk encryption, NVMe and PXE boot for Intel User Manual for an introduction to this Extension Pack. The Extension the VirtualBox Personal Use and Evaluation License (PUEL). Please install pack as your installed version of VirtualBox.

실행화면



Oracle VM VirtualBox 관리자

파일(F) 머신(M) 도움말(H)

도구

환경 설정(P) 가져오기 내보내기 새로 만들기(N) 추가(A)

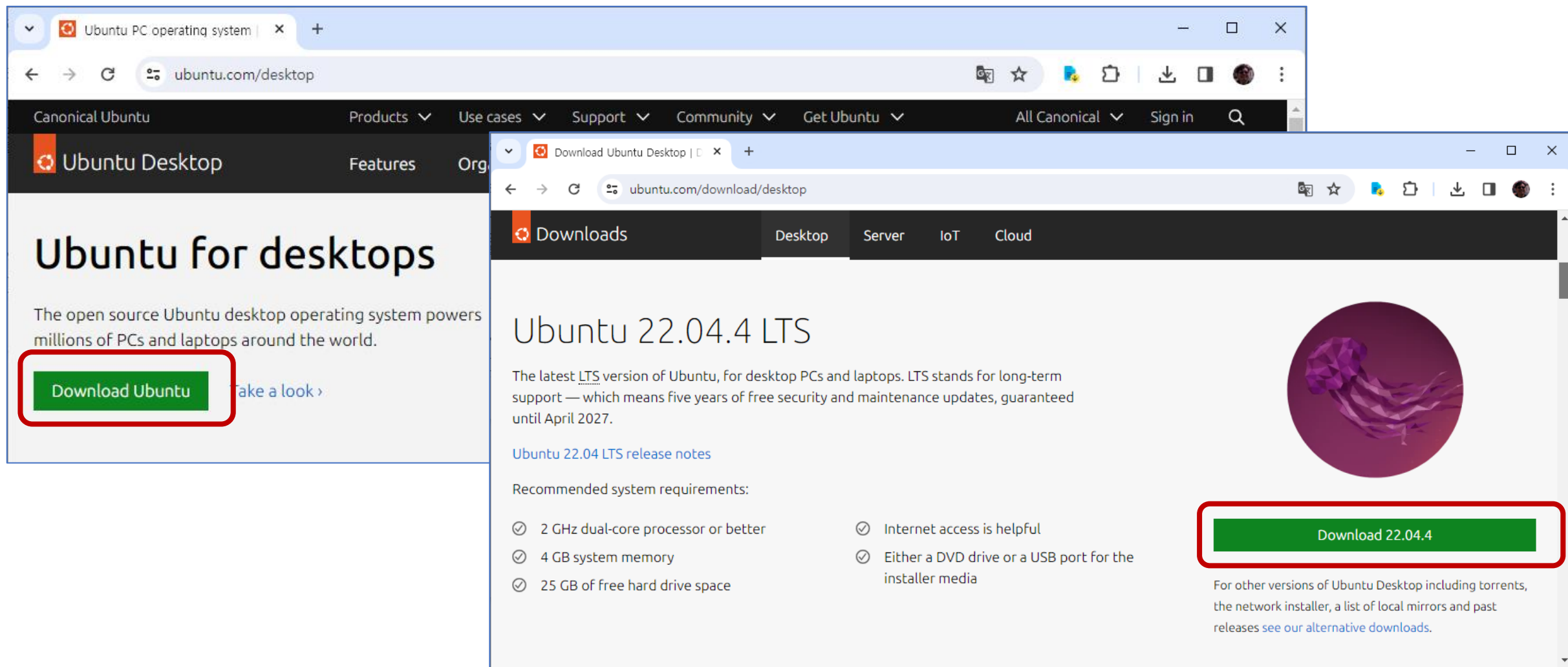
VirtualBox에 오신 것을 환영합니다!

이 프로그램의 왼쪽 부분은 전역 도구 및 컴퓨터에 있는 모든 가상 머신과 가상 머신 그룹 목록을 표시합니다. 도구 모음의 단추를 사용하여 새로운 가상 머신을 만들거나, 추가하거나, 가져올 수 있습니다. 현재 선택한 구성 요소에 사용할 수 있는 도구 모음 단추를 눌러 해당하는 도구 모음을 호출할 수 있습니다.

F1 키를 누르면 상황에 맞는 도움말을 볼 수 있으며, 최근 정보와 뉴스를 보려면 www.virtualbox.org를 방문하십시오.

Ubuntu 설치 - Ubuntu 22.04 LTS 다운로드

<https://ubuntu.com/desktop>



Canonical Ubuntu Products Use cases Support Community Get Ubuntu All Canonical Sign in

Ubuntu Desktop

Features Org

Ubuntu for desktops

The open source Ubuntu desktop operating system powers millions of PCs and laptops around the world.

[Download Ubuntu](#) [Take a look >](#)

Downloads

Desktop Server IoT Cloud

Ubuntu 22.04.4 LTS

The latest [LTS](#) version of Ubuntu, for desktop PCs and laptops. LTS stands for long-term support — which means five years of free security and maintenance updates, guaranteed until April 2027.

[Ubuntu 22.04 LTS release notes](#)

Recommended system requirements:

- ✓ 2 GHz dual-core processor or better
- ✓ 4 GB system memory
- ✓ 25 GB of free hard drive space
- ✓ Internet access is helpful
- ✓ Either a DVD drive or a USB port for the installer media

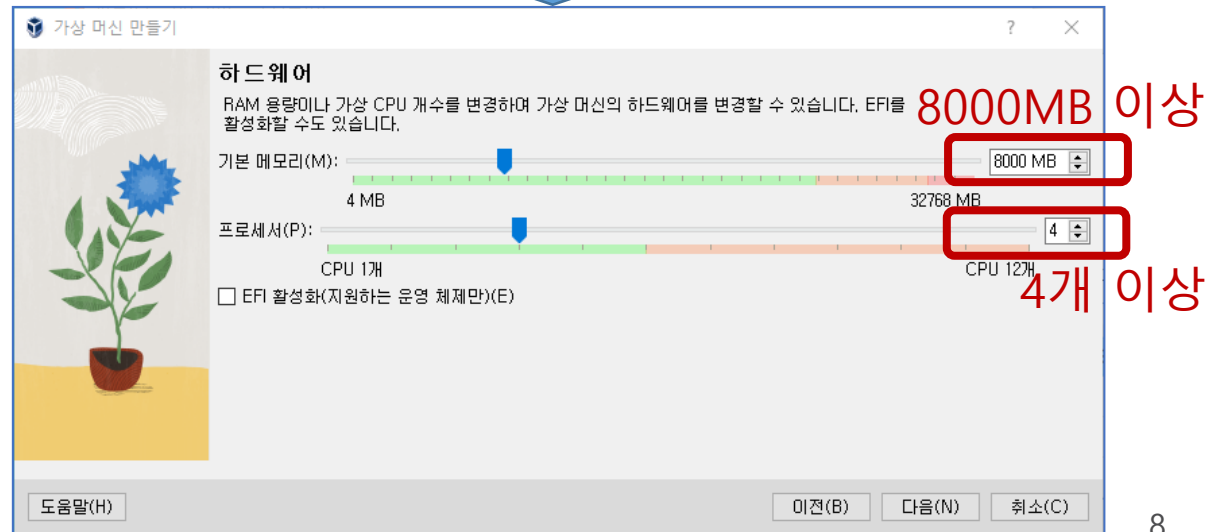
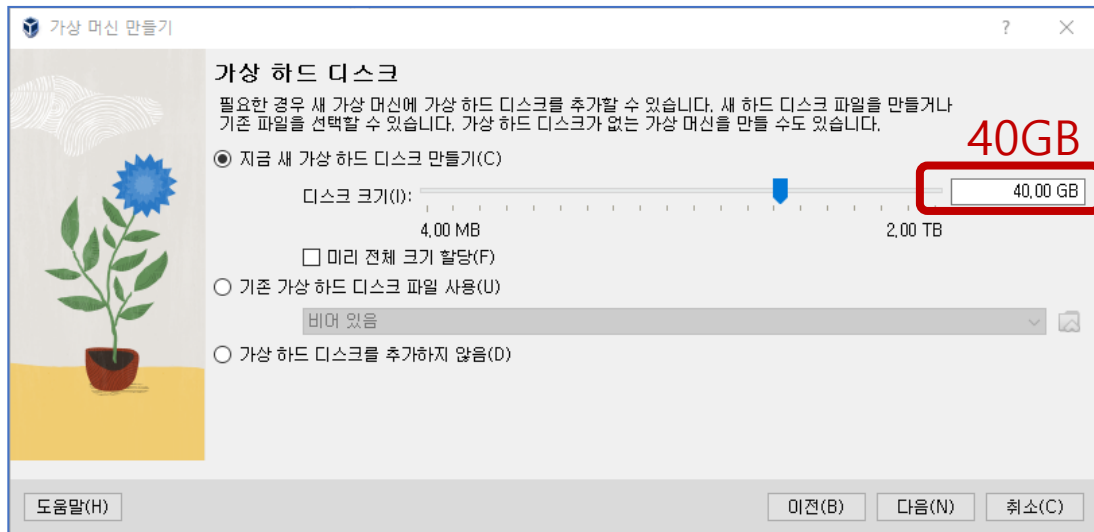
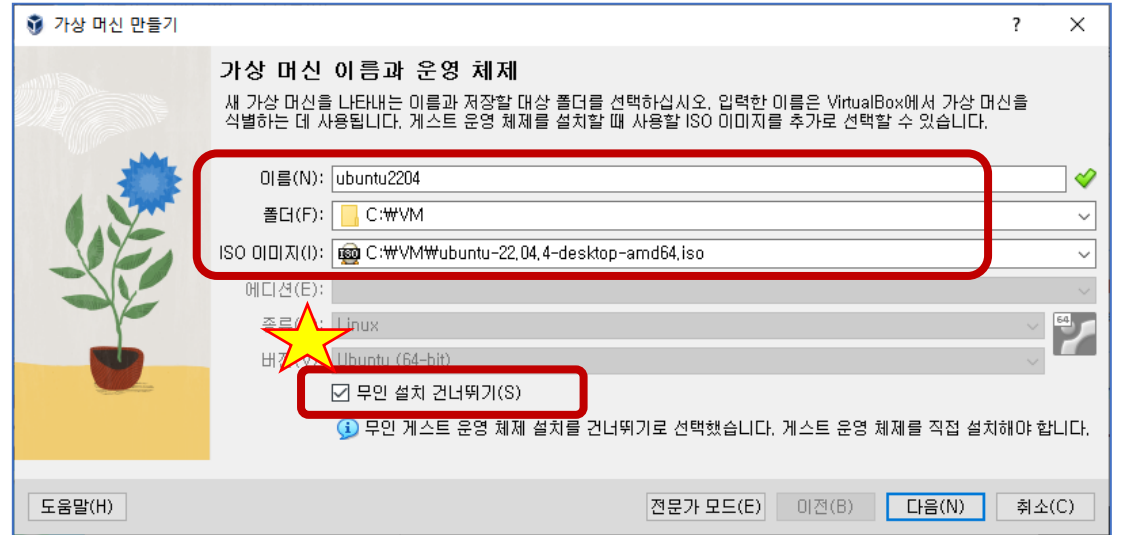
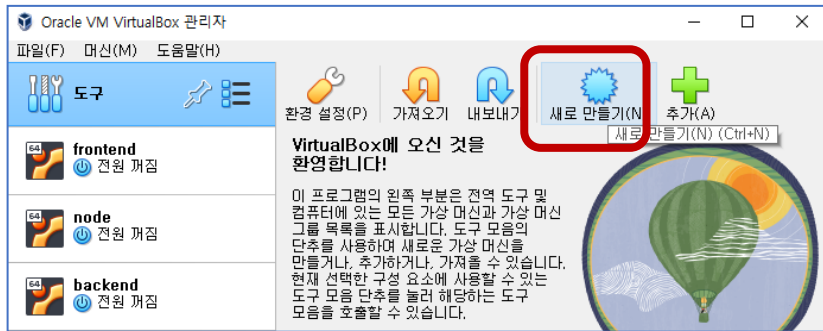
[Download 22.04.4](#)

For other versions of Ubuntu Desktop including torrents, the network installer, a list of local mirrors and past releases [see our alternative downloads](#).

Ubuntu 설치 - 새로 만들기

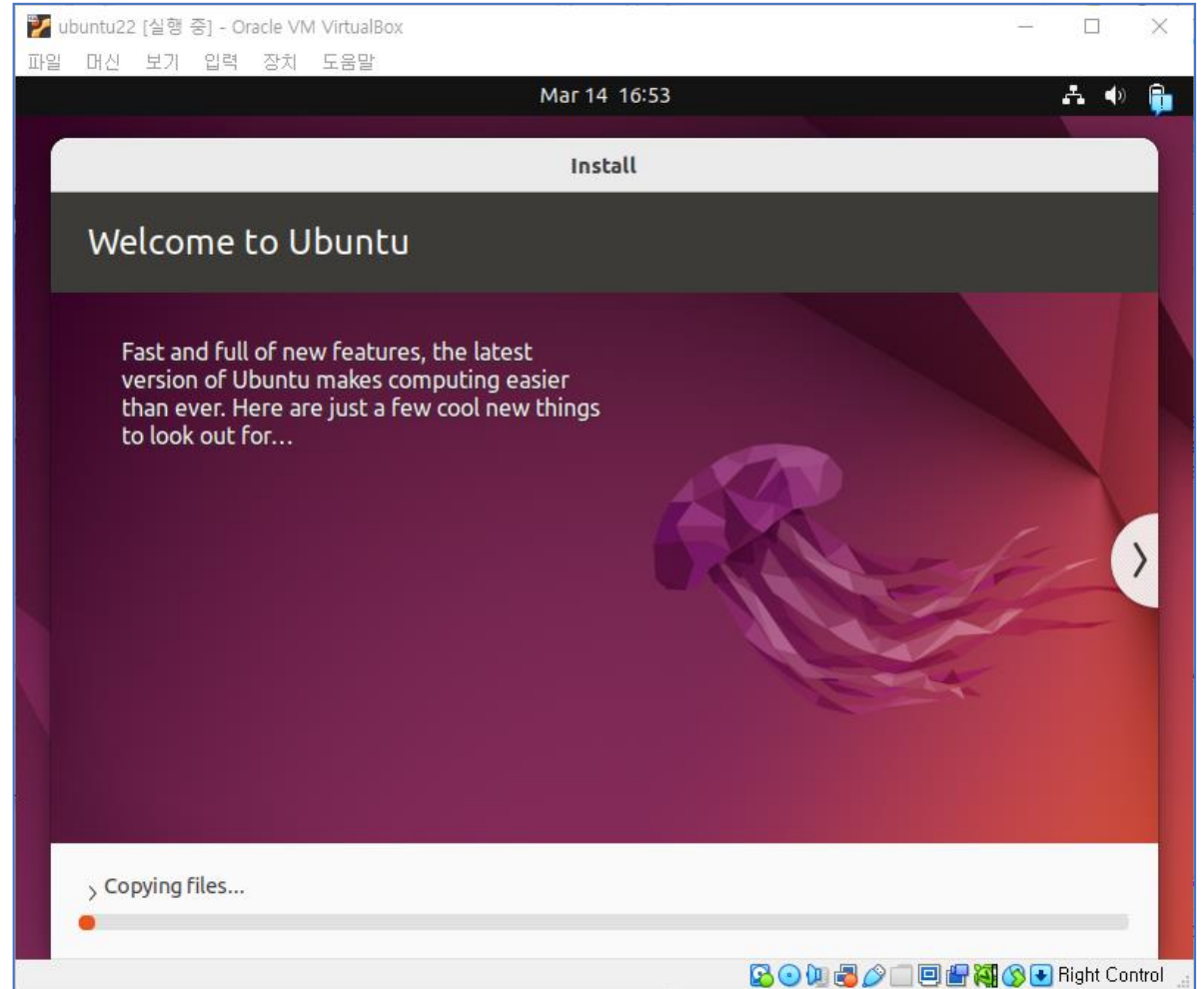
이름 입력
설치 폴더 선택
ubuntu-22.04.4-desktop-amd64.iso 선택

VirtualBox 관리자에서 [새로만들기] 클릭



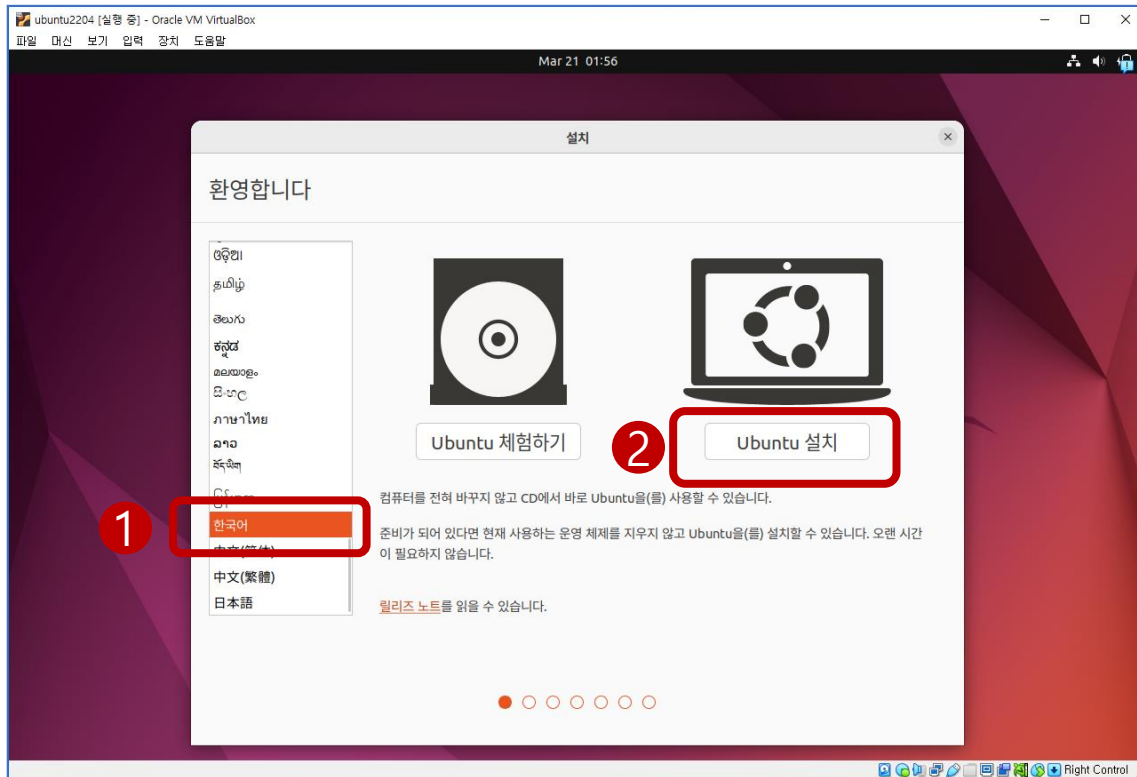
Ubuntu 설치 - Install Ubuntu

[시작(T)] 버튼 클릭

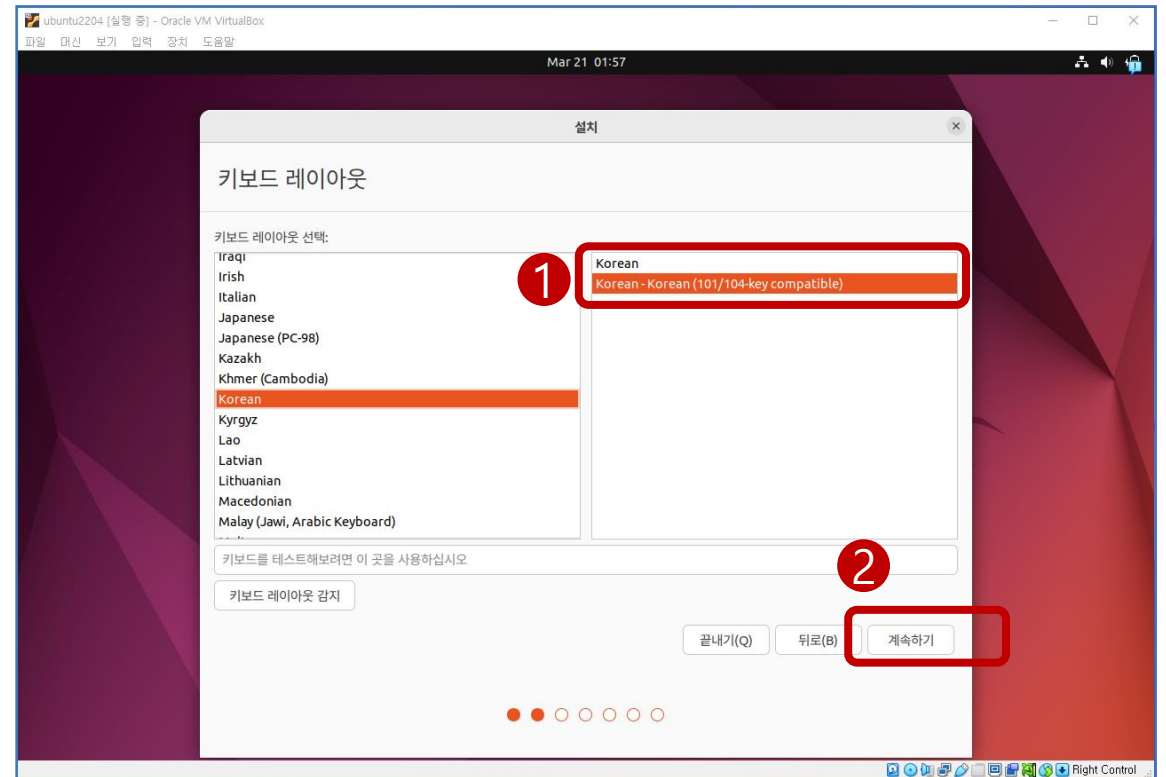


Ubuntu 설치 - 언어 및 키보드 선택

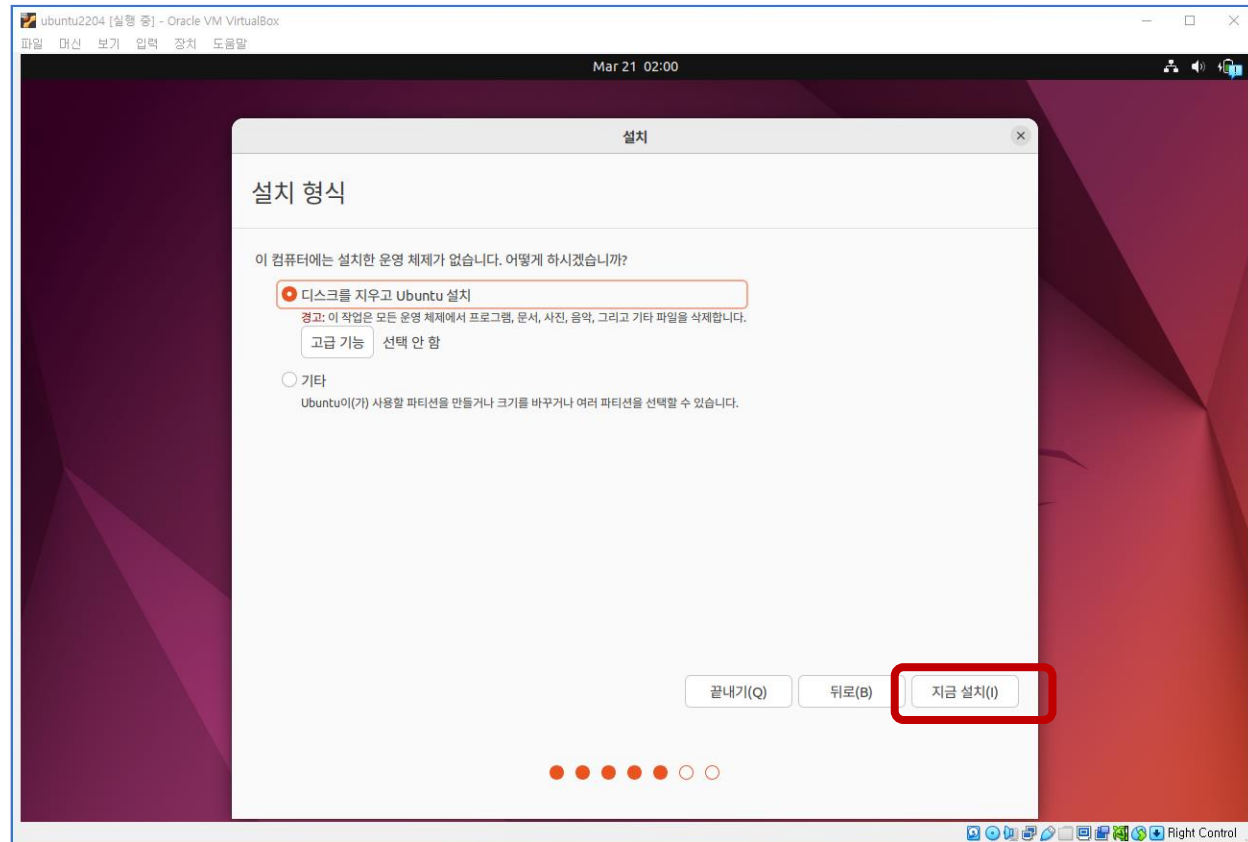
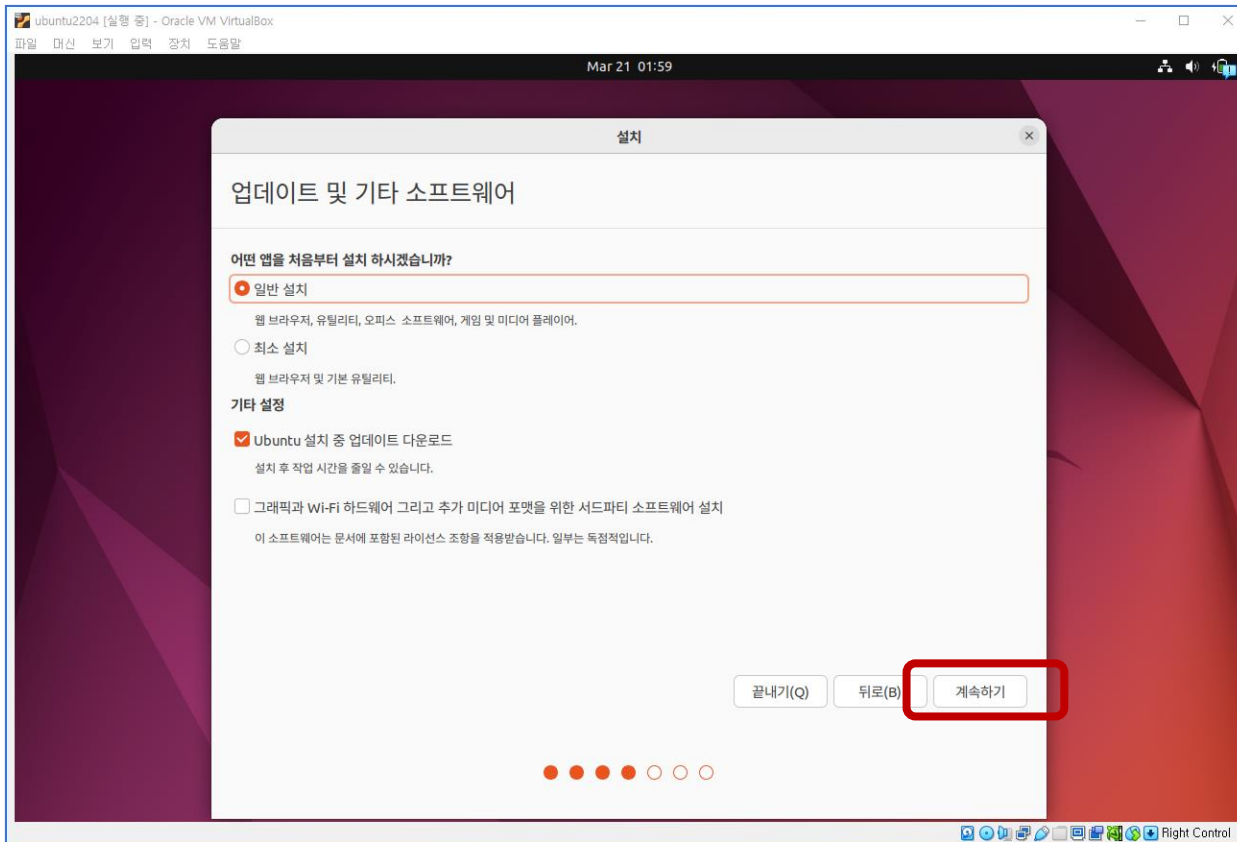
- ① [한국어] 선택
- ② [ubuntu 설치] 선택하고 [Enter Key] 누름



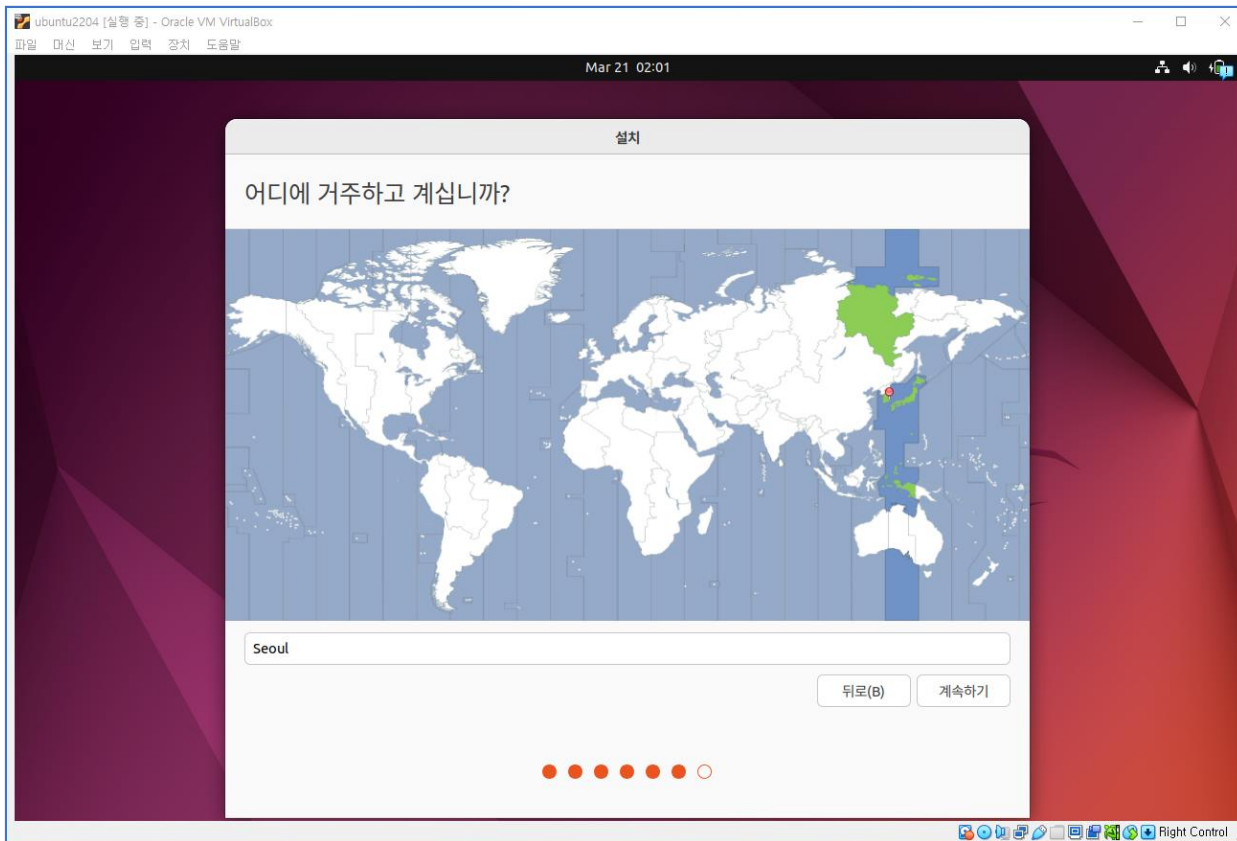
- ① 키보드 레이아웃 선택
- ② [계속하기] 클릭



Ubuntu 설치 - 업데이트 및 설치 형식 선택

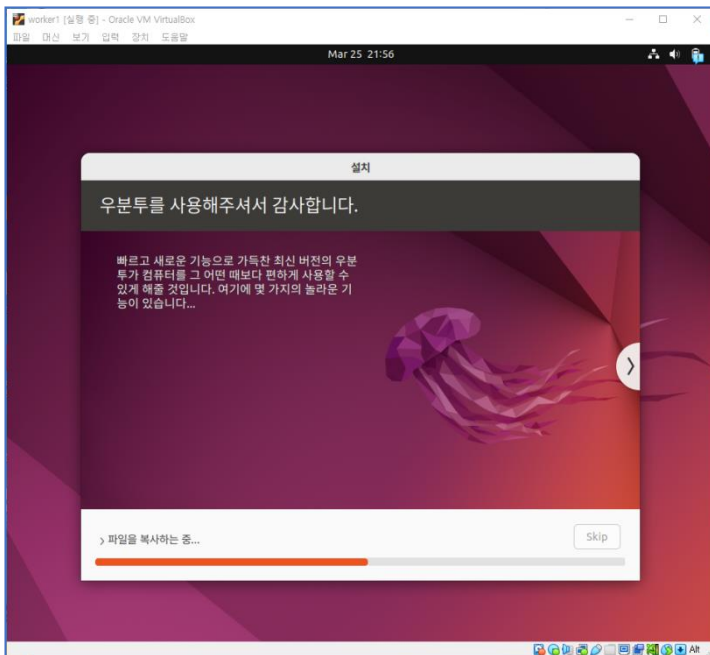


Ubuntu 설치 - 지역 선택 및 계정 설정

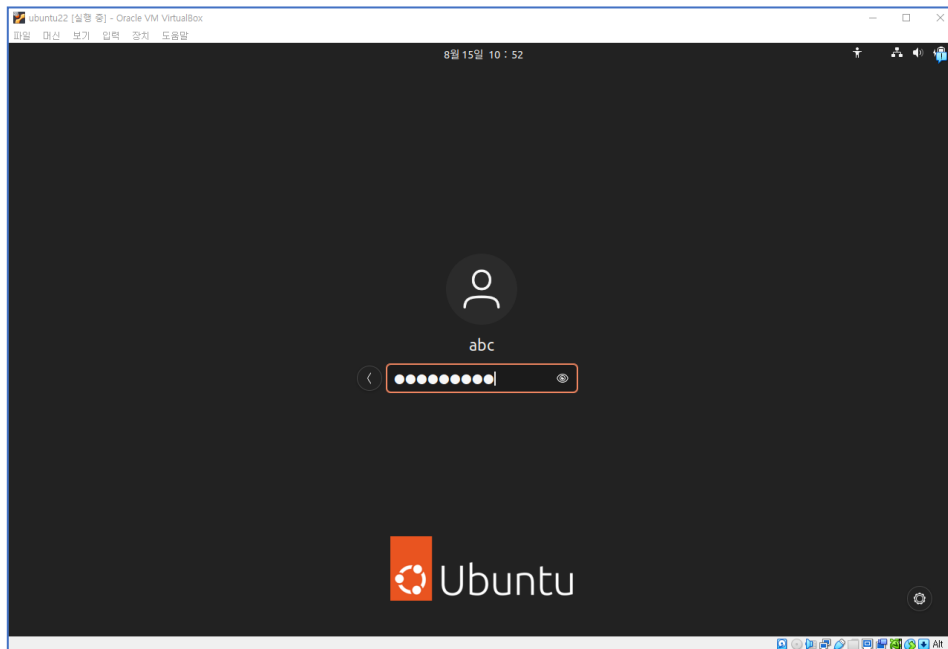


Ubuntu 설치 - 설치 완료 및 재부팅

설치 진행 화면



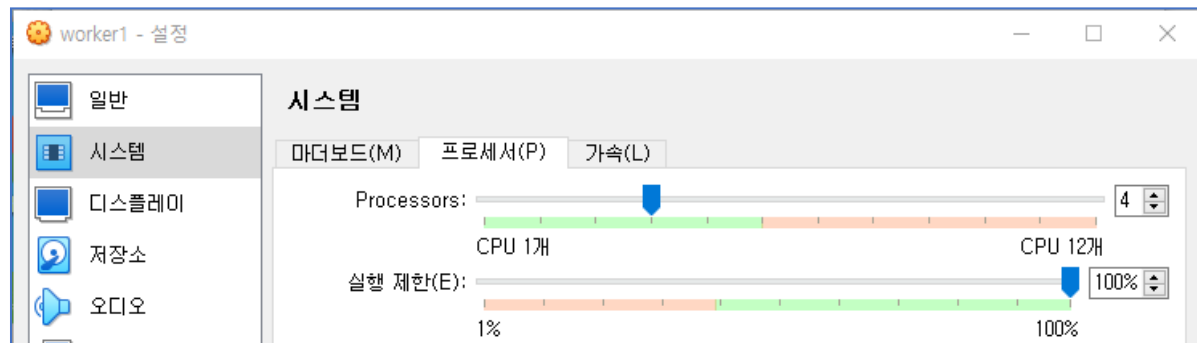
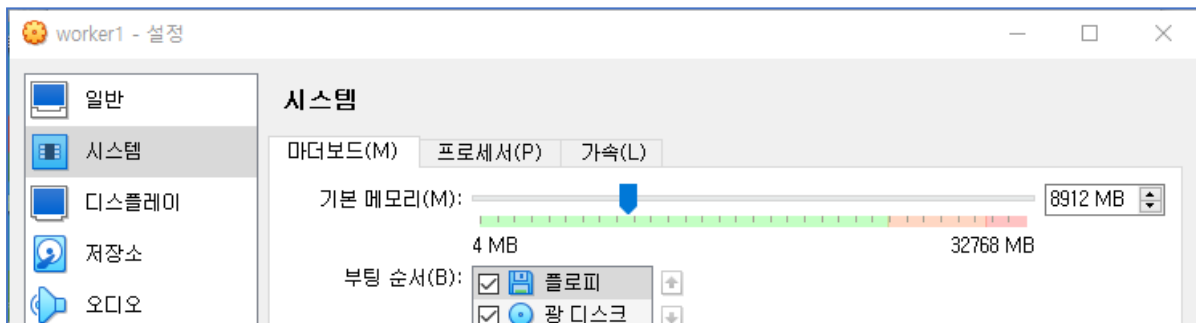
로그인 화면



가상머신 운영 최소 권장 사항

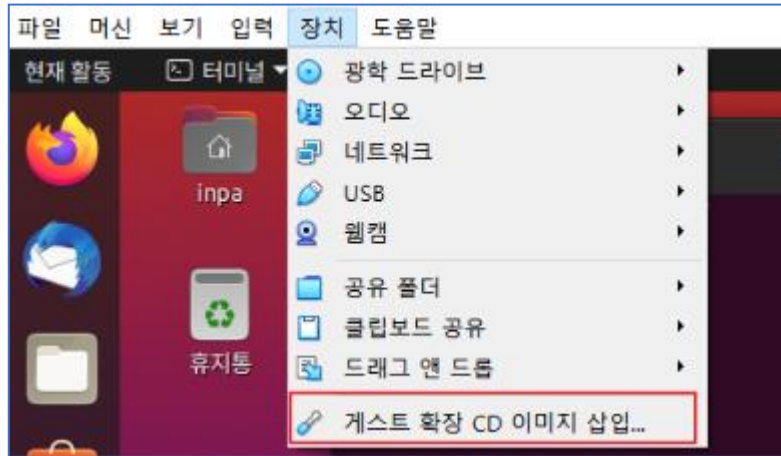
메모리 : 8G 이상

CPU : 4개 이상



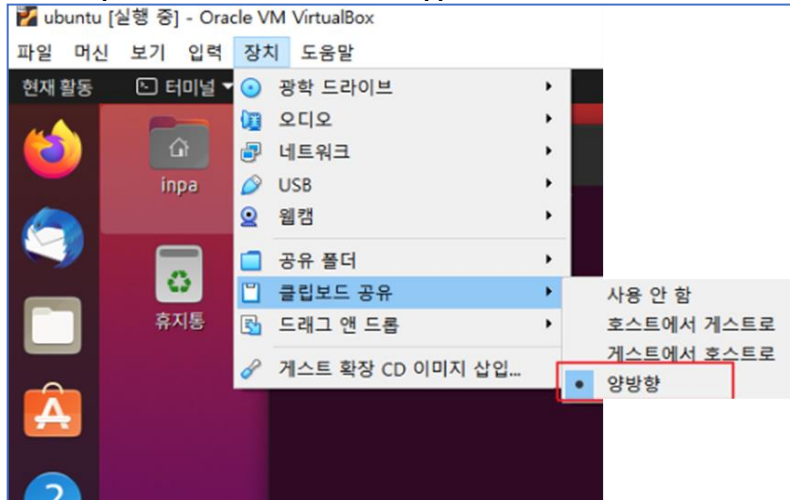
Ubuntu 설정

■ 게스트 확장 설치



■ 클립보드 공유 설정

장치 → 클립보드 공유 → 양방향



■ 복사-붙여넣기 단축키

■ 가상머신에서 복사 및 붙여넣기

가상머신에서 복사 : Ctrl + Insert

가상머신에 붙여넣기 : Shift + Insert

■ 윈도우에서 가상머신으로 복사하기

윈도우에서 복사 : Ctrl + C

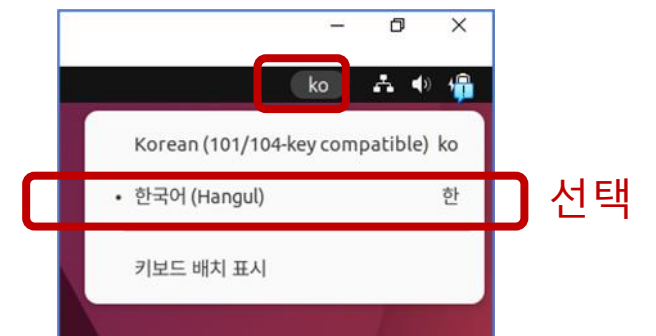
가상머신에서 붙여넣기 : Shift + Insert

■ 가상머신에서 윈도우로 복사하기

가상머신에서 복사 : Ctrl + Insert

윈도우에 붙여넣기 : Ctrl + V

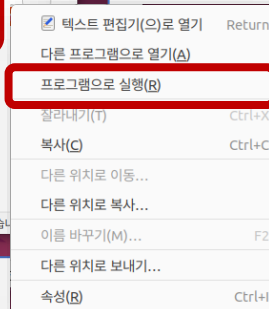
■ 한-영 전환 단축키



Shift + Space



autorun.sh



Pyenv 설치

Pyenv는 다양한 버전의 파이썬을 관리하기 위한 오픈소스입니다.

▪ Step 1: 프로그램 설치

```
sudo apt update  
sudo apt install -y make build-essential libssl-dev zlib1g-dev libbz2-dev libreadline-dev libsqlite3-dev wget  
curl llvm libncurses5-dev libncursesw5-dev xz-utils tk-dev libffi-dev liblzma-dev git
```

▪ Step 2: Pyenv 설치

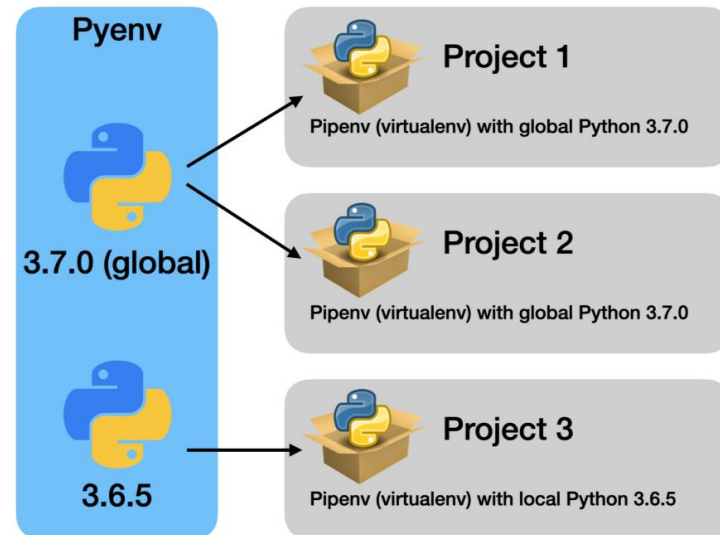
```
curl https://pyenv.run | bash
```

▪ Step 3: 환경 설정

```
echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc  
echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bashrc  
echo -e 'if command -v pyenv 1>/dev/null 2>&1; then\n eval "$(pyenv init -)"\nfi' >> ~/.bashrc  
source ~/.bashrc
```

▪ Step 4: 사용

```
pyenv install -l | more  
pyenv install 3.11.8  
pyenv versions  
pyenv global 3.11.8  
python --version
```



Pipenv 및 패키지 설치

pipenv는 기존의 pip와 virtualenv의 기능을 통합하여 효율적인 패키지 관리 및 가상 환경 설정을 가능하게 합니다.
Pipfile과 Pipfile.lock 파일로 프로젝트의 의존성을 명확하고 가독성 있게 관리할 수 있도록 해줍니다.

▪ pipenv 설치

```
sudo apt install python3-pip  
sudo pip3 install pipenv virtualenv
```

```
pipenv --python 3.11.8
```

```
pipenv shell
```

```
pipenv install virtualenv notebook jupyterlab mlflow scikit-learn tensorflow hyperop
```

▪ pipenv 사용법

1. pipenv 설치
`pip install pipenv`

2. 가상환경 생성
`pipenv --python 3.X`

3. 가상환경 제거
`pipenv --rm`

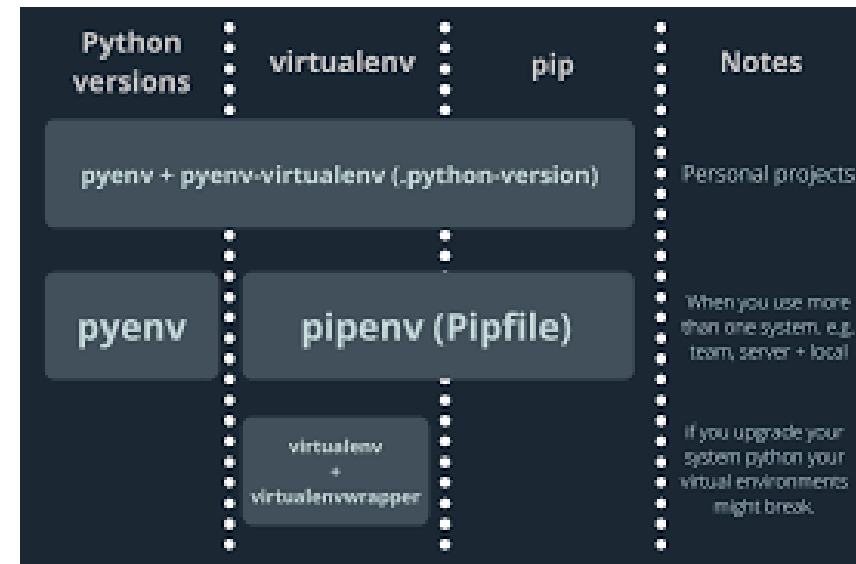
4. 가상환경 터미널 실행
`pipenv shell`

5. 가상환경 터미널 종료
`exit`

6. 패키지 설치
`pipenv install [package]`

7. pipfile에 명시된 패키지 설치
`pipenv install`

8. 설치된 라이브러리 출력
`pipenv graph`



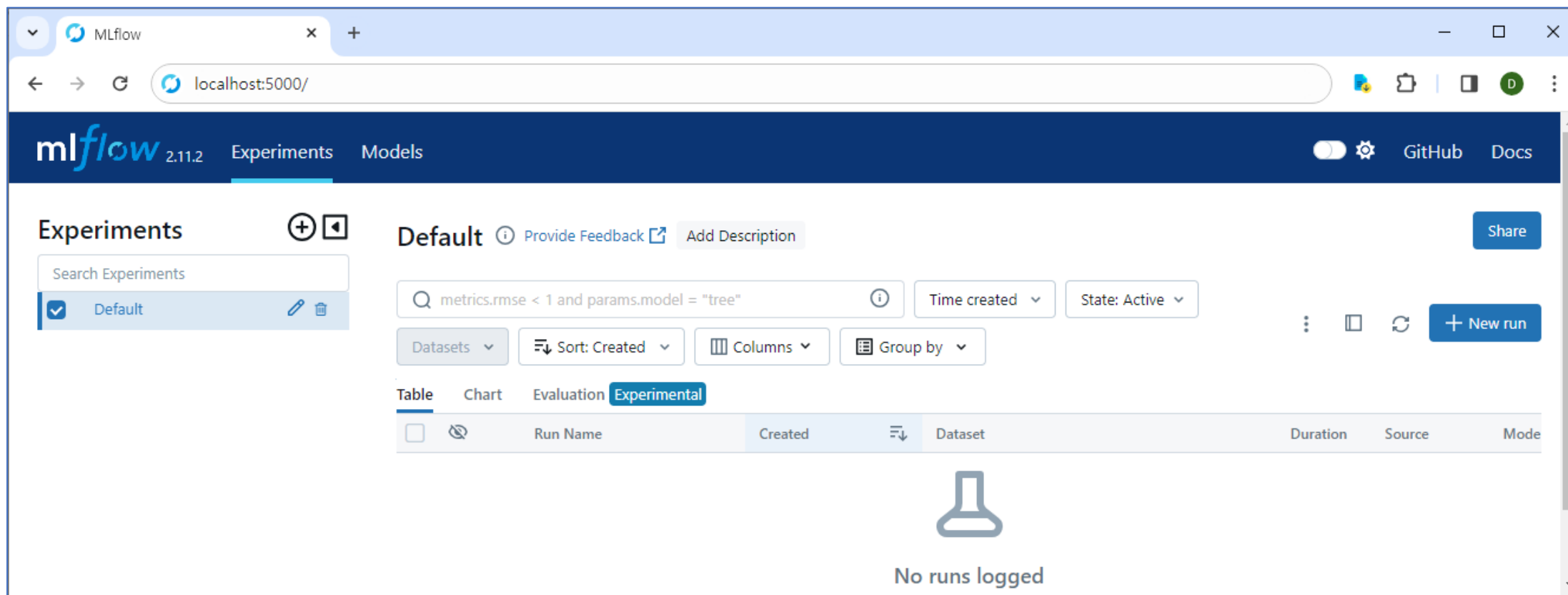
MLflow Server

- **MLflow server 시작**

`mlflow server -h 0.0.0.0`

- **MLflow server 접속**

웹브라우저에서 <http://localhost:5000/> 접속



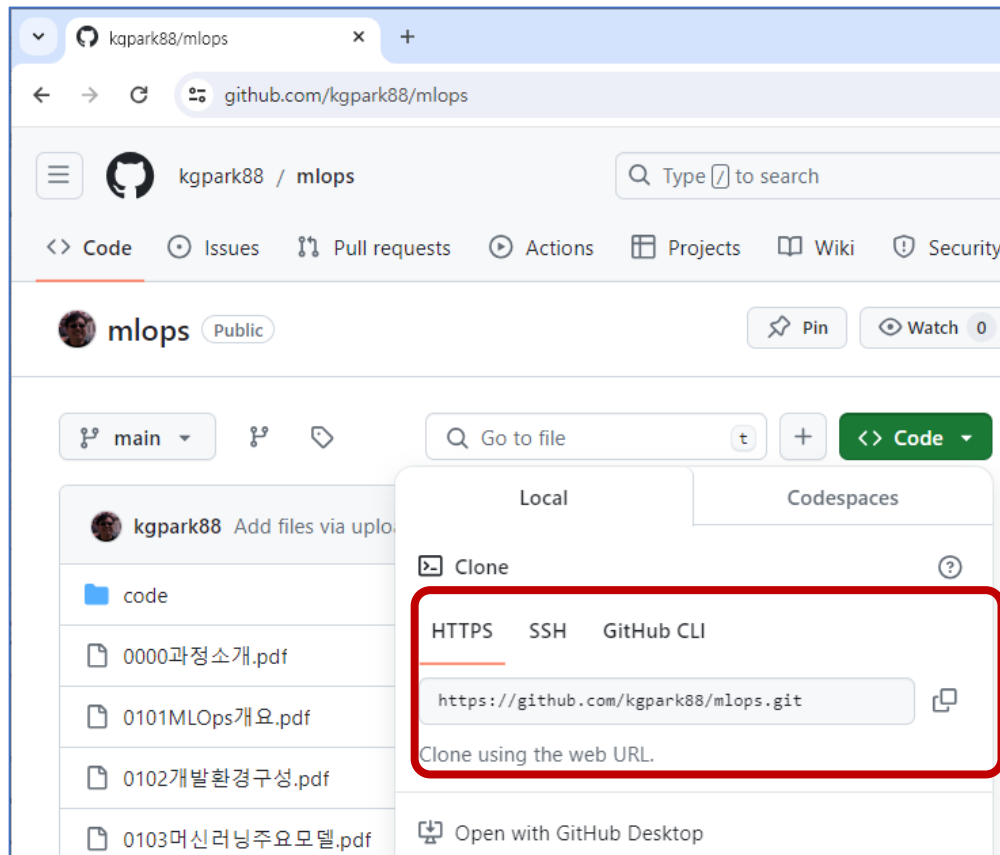
로컬서버 모델 서빙

로컬 서버 구성

- Windows : Hyper-V 비활성화
- VirtualBox 설치
- Ubuntu 가상머신 설치
- Pyenv 설치
- Pipenv 설치
- MLflow 설치
- MLflow 서버 시작

▪ 실습소스 다운로드

```
git clone https://github.com/kgpark88/mlops  
cd ~/mlops/code  
jupyter notebook
```



로컬서버 모델 서빙

compare_runs_deploy.ipynb

기본 포트인 5000으로 Tracking 서버를 실행합니다.

```
mlflow server
```

MLflow를 사용하면 모든 실행 또는 모델 버전에서 생성된 모델을 쉽게 서빙할 수 있습니다.

모델 서빙 포트는 Tracking 서버 포트번호와 다르게 지정해야 합니다.

모델을 서빙하기 위해 runs:/<run_id> URI를 사용하거나,

[Artifact Store](#)어에 설명된 지원되는 URI를 사용할 수도 있습니다.

등록한 모델을 실행하여 서빙할 수 있습니다.

```
export MLFLOW_TRACKING_URI=http://localhost:5000
```

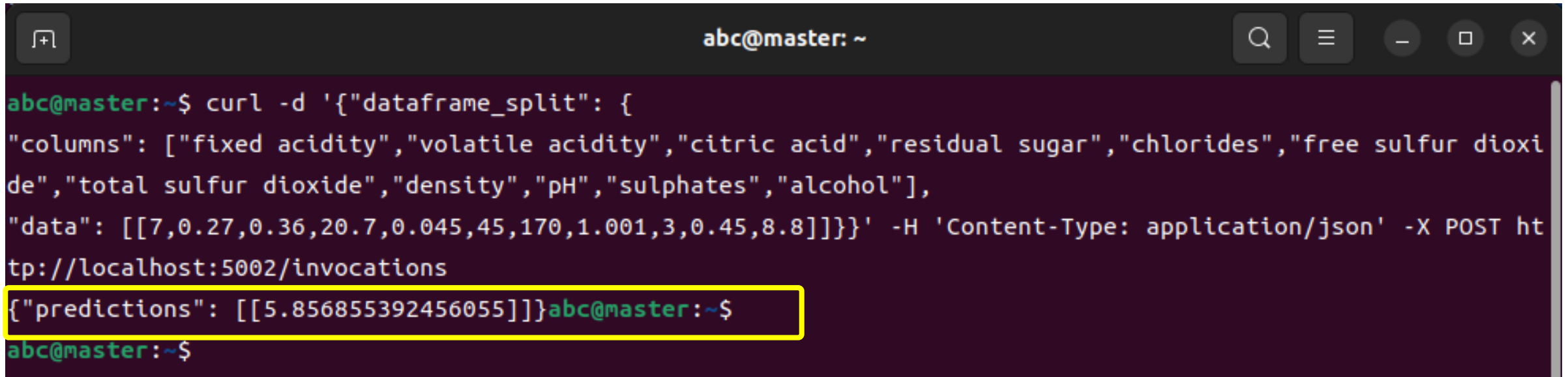
```
mlflow models serve -m "models:/wine-quality/1" --port 5002
```

로컬서버 모델 서빙

compare_runs_deploy.ipynb

curl 명령을 사용하여 REST API에 요청을 보내서 모델을 테스트 해봅니다.

```
curl -d '{"dataframe_split": { "columns": ["fixed acidity","volatile acidity","citric acid","residual sugar","chlorides","free sulfur dioxide","total sulfur dioxide","density","pH","sulphates","alcohol"], "data": [[7,0.27,0.36,20.7,0.045,45,170,1.001,3,0.45,8.8]]}}' -H 'Content-Type: application/json' -X POST http://localhost:5002/invocations
```

A terminal window titled 'abc@master: ~' with standard window controls. It shows a curl command being executed. The output of the command is highlighted with a yellow box.

```
abc@master:~$ curl -d '{"dataframe_split": {  
"columns": ["fixed acidity","volatile acidity","citric acid","residual sugar","chlorides","free sulfur dioxi  
de","total sulfur dioxide","density","pH","sulphates","alcohol"],  
"data": [[7,0.27,0.36,20.7,0.045,45,170,1.001,3,0.45,8.8]]}}' -H 'Content-Type: application/json' -X POST ht  
tp://localhost:5002/invocations  
{"predictions": [[5.856855392456055]]}abc@master:~$  
abc@master:~$
```

모델 레지스트리 (Registry)

MLflow 모델 레지스트리에는 몇 가지 핵심 구성 요소가 있습니다:

- 중앙 집중식 모델 스토어 : MLflow 모델을 위한 단일 위치로, 일관되고 효율적인 방식으로 모델 버전 관리, 공유 및 배포를 용이하게 합니다.
- API 세트 : 프로그래밍 방식으로 모델을 만들고, 읽고, 업데이트하고, 삭제할 수 있습니다.
- GUI : 중앙 집중식 모델 스토어에서 모델을 수동으로 보고 관리

MLflow 모델 레지스트리는 모델 개발 및 배포와 관련된 몇 가지 추가 기능을 제공합니다:

- 모델 버전 관리 : 비교와 서빙을 용이하게 하기 위해 모델의 다양한 iterations의 기록을 말합니다.
기본적으로 모델은 ID로 버전이 지정되고, 모델 버전에 별칭을 지정할 수도 있습니다.
- 모델 별칭 : 모델의 특정 버전에 변경 가능하고 이름이 지정된 참조를 할당하여 모델 배포를 간소화할 수 있습니다.
- 모델 태그 : 사용자가 사용자 지정 key-value 쌍으로 모델에 레이블을 지정하여 문서화 및 분류를 용이하게 해줍니다.
- 모델 주석 : 모델에 추가된 설명 메모입니다

▼ 1단계: 모델 등록하기

MLflow 모델 레지스트리를 사용하려면 MLflow 모델을 추가해야 합니다. 이는 아래 명령 중 하나를 통해 지정된 모델을 등록하면 됩니다:

- `mlflow.<model_flavor>.log_model(registered_model_name=<model_name>)` : 추적 서버에 로깅하면서 모델을 등록합니다.
- `mlflow.register_model(<model_uri>, <model_name>)` : 추적 서버에 로깅한 후 모델을 등록합니다.

이 명령을 실행하기 전에 모델을 로깅해야 모델 URI를 얻을 수 있습니다.

MLflow에는 다양한 모델 종류가 있습니다.

아래 예에서는 가장 간단한 모델 등록 방법을 보여드리기 위해 scikit-learn의 RandomForestRegressor를 활용하지만, 지원되는 모든 모델 flavor를 활용할 수 있습니다.

아래 코드 스니펫에서는 mlflow 실행을 시작하고 랜덤 포레스트 모델을 훈련합니다.

그런 다음 몇 가지 관련 하이퍼파라미터와 모델 평균제곱오차(MSE)를 기록하고 마지막으로 모델 자체를 기록 및 등록합니다.

```
[3]: from sklearn.datasets import make_regression
      from sklearn.ensemble import RandomForestRegressor
      from sklearn.metrics import mean_squared_error
      from sklearn.model_selection import train_test_split

      import mlflow
      import mlflow.sklearn

      mlflow.set_tracking_uri(uri="http://127.0.0.1:5000")
```

```

with mlflow.start_run() as run:
    X, y = make_regression(n_features=4, n_informative=2, random_state=0, shuffle=False)
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42
    )

    params = {"max_depth": 2, "random_state": 42}
    model = RandomForestRegressor(**params)
    model.fit(X_train, y_train)

    # Log parameters and metrics using the MLflow APIs
    mlflow.log_params(params)

    y_pred = model.predict(X_test)
    mlflow.log_metrics({"mse": mean_squared_error(y_test, y_pred)})

    # Log the sklearn model and register as version 1
    mlflow.sklearn.log_model(
        sk_model=model,
        artifact_path="sklearn-model",
        input_example=X_train,
        registered_model_name="sk-learn-random-forest-reg-model",
    )

```

모델을 등록하려면 `mlflow.sklearn.log_model()`에서 등록된 모델 이름 매개 변수를 활용하거나 모델을 로깅한 후 `mlflow.register_model()`을 호출할 수 있습니다. 일반적으로 전자가 더 간결하기 때문에 전자를 권장합니다.

모델 서명은 모델 입력과 출력에 대한 유효성 검사를 제공합니다. `log_model()`의 `input_example`은 자동으로 서명을 유추하여 기록합니다. 다시 말하지만, 간결하기 때문에 이 구현을 사용하는 것이 좋습니다

등록된 모델 살펴보기

이제 실험을 기록하고 해당 실험 실행과 관련된 모델을 등록했으므로 이 정보가 실제로 MLflow UI와 로컬 디렉터리에 어떻게 저장되는지 살펴봅시다. 프로그래밍 방식으로 이 정보를 얻을 수 있지만 설명하기 위해 MLflow UI를 사용하겠습니다.

1단계: mlruns 디렉토리 살펴보기

로컬 파일 시스템을 Tracking 서버 및 모델 레지스트리로 사용하고 있으므로, 이전 단계에서 파이썬 스크립트를 실행할 때 생성된 디렉터리 구조를 관찰해 보겠습니다.

자세히 알아보기 전에 MLflow는 사용자의 복잡성을 줄이기 위해 설계되었으며 이 디렉터리 구조는 단지 설명을 위한 것임을 알아두는 것이 중요합니다. 또한, 프로덕션 사용 사례에 권장되는 원격 배포의 경우 Tracking 서버는 객체 저장소(S3, ADLS, GCS 등)에 있고 모델 레지스트리는 관계형 데이터베이스(PostgreSQL, MySQL 등)에 있습니다

```

mlruns/
├── 0/                                     # Experiment ID
│   ├── bc6dc2a4f38d47b4b0c99d154bbc77ad/ # Run ID
│   │   ├── metrics/
│   │   │   └── mse                       # Example metric file for mean squared error
│   │   ├── artifacts/                   # Artifacts associated with our run
│   │   │   ├── sklearn-model/
│   │   │   │   ├── python_env.yaml
│   │   │   │   ├── requirements.txt      # Python package requirements
│   │   │   │   ├── MLmodel              # MLflow model file with model metadata
│   │   │   │   ├── model.pkl            # Serialized model file
│   │   │   │   ├── input_example.json
│   │   │   │   └── conda.yaml
│   │   ├── tags/
│   │   │   ├── mlflow.user
│   │   │   ├── mlflow.source.git.commit
│   │   │   ├── mlflow.runName
│   │   │   ├── mlflow.source.name
│   │   │   ├── mlflow.log-model.history
│   │   │   └── mlflow.source.type
│   │   ├── params/
│   │   │   ├── max_depth
│   │   │   └── random_state
│   │   └── meta.yaml
│   └── meta.yaml
├── models/                               # Model Registry Directory
│   ├── sk-learn-random-forest-reg-model/ # Registered model name
│   │   ├── version-1/                   # Model version directory
│   │   │   └── meta.yaml
│   │   └── meta.yaml

```

Tracking 서버는 Experiment ID와 Run ID 로 구성되며 Experiment artifacts, Parameters, Metrics를 저장하는 역할을 담당합니다.

반면에 모델 레지스트리는 Tracking 서버에 대한 포인터와 함께 메타데이터만 저장합니다.

autologging 지원하는 flavor는 기본적으로 많은 추가 정보를 제공합니다.

또한 관심 있는 모델에 대해 오토로깅을 지원하지 않더라도 명시적 로깅 호출을 통해 이 정보를 쉽게 저장할 수 있다는 점에 유의하세요.

한 가지 더 흥미로운 점은 기본적으로 모델 환경을 관리할 수 있는 세 가지 방법,

python_env.yaml(python virtualenv), requirements.txt(PyPi requirements), conda.yaml(conda env)이 제공되는 것입니다.

이제 로깅되는 내용에 대해 이해했으니 MLflow UI를 사용하여 이 정보를 확인해 보겠습니다.

2단계: Tracking Server 시작

- mlruns 폴더와 동일한 디렉터리에서 아래 명령을 실행합니다.

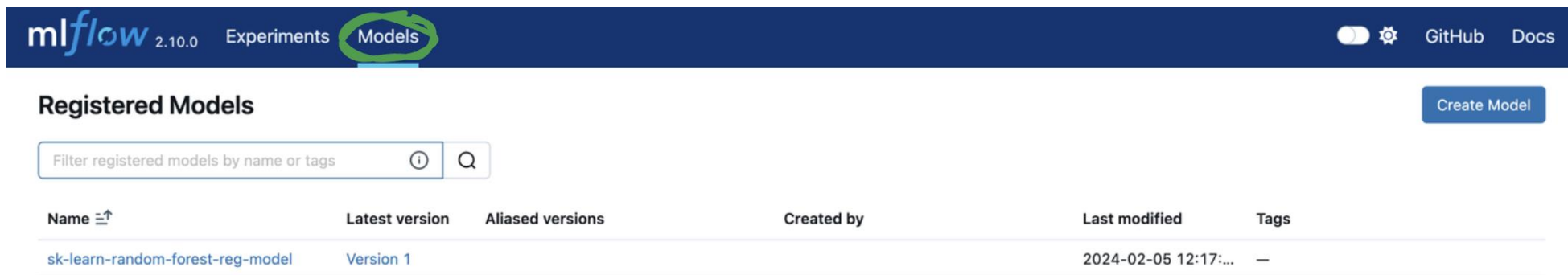
```
mlflow server --host 127.0.0.1 --port 8080
```

```
mlflow server --host 127.0.0.1 --port 8080
```

```
[2024-02-05 12:43:10 -0500] [26393] [INFO] Starting gunicorn 20.1.0
[2024-02-05 12:43:10 -0500] [26393] [INFO] Listening at: http://127.0.0.1:8080 (26393)
[2024-02-05 12:43:10 -0500] [26393] [INFO] Using worker: sync
[2024-02-05 12:43:10 -0500] [26414] [INFO] Booting worker with pid: 26414
[2024-02-05 12:43:11 -0500] [26416] [INFO] Booting worker with pid: 26416
[2024-02-05 12:43:11 -0500] [26428] [INFO] Booting worker with pid: 26428
[2024-02-05 12:43:11 -0500] [26437] [INFO] Booting worker with pid: 26437
```

3단계: 트래킹 서버 보기


웹 브라우저로 <http://localhost:8080>에 접속하여 MLflow UI를 볼 수 있습니다.
models 메뉴를 클릭하여 모델 레지스트리로 이동합니다.



mlflow 2.10.0 Experiments **Models** GitHub Docs

Registered Models Create Model

Filter registered models by name or tags Q

Name 	Latest version	Aliased versions	Created by	Last modified	Tags
sk-learn-random-forest-reg-model	Version 1			2024-02-05 12:17:...	—

다음으로 모델 배포를 용이하게 하기 위해 태그와 모델 버전 별칭을 추가해 보겠습니다.

모델 버전 표에서 해당 추가 링크 또는 연필 아이콘을 클릭하여 태그 및 별칭을 추가하거나 편집할 수 있습니다.

1. key는 `problem_type`으로 value는 `regression` 으로 모델 버전 태그를 추가합니다.

mlflow 2.10.0 Experiments Models

Registered Models >

sk-learn-random-forest-reg-model

Created Time: 2024-02-05 12:17:39 Last Modified: 2024-02-05 15:43:09

> Description Edit

> Tags

▼ Versions Compare

New model registry UI ☒

Version	Registered at ↕	Created by	Tags	Aliases	Description
✓ Version 1	2024-02-05 12:17:39		problem_type: regression	@ the_best_model_ever	

등록된 모델 로드

등록된 모델 버전에 대해 추론을 수행하려면 해당 모델을 메모리에 로드해야 합니다.
모델 버전을 찾는 방법은 여러 가지가 있지만, 사용 가능한 정보에 따라 가장 좋은 방법은 달라집니다.
아래 코드는 특정 모델 URI를 통해, 모델 레지스트리에서 모델을 로드하고 추론을 수행하는 가장 간단한 방법을 보여줍니다.

```
import mlflow.sklearn
from sklearn.datasets import make_regression

model_name = "sk-learn-random-forest-reg-model"
model_version = "latest"

# Load the model from the Model Registry
model_uri = f"models://{model_name}/{model_version}"
model = mlflow.sklearn.load_model(model_uri)

# Generate a new dataset for prediction and predict
X_new, _ = make_regression(n_features=4, n_informative=2, random_state=0, shuffle=False)
y_pred_new = model.predict(X_new)

print(y_pred_new)
```

sklearn을 사용하지 않는 경우, model flavor가 지원되는 경우,
특정 model flavor 로드 메서드를 사용해야 합니다. 예: `mlflow.<flavor>.load_model()`
model flavor가 지원되지 않는 경우 `mlflow.pyfunc.load_model()` 을 활용해야 합니다.
이 튜토리얼에서는 데모 목적으로 sklearn을 활용합니다.

예제 0: 추적 서버를 통한 로드

모델 URI는 직렬화된 모델의 고유 식별자입니다.

모델 아티팩트가 tracking 서버에 experiments와 함께 저장되어 있는 경우,
아래 모델 URI를 사용하여 모델 레지스트리를 우회하여 아티팩트를 메모리로 로드할 수 있습니다.

1. 절대 경로: `mlflow.sklearn.load_model("/Users/me/path/to/local/model")`
2. 상대 경로: `mlflow.sklearn.load_model("relative/path/to/local/model")`
3. Run ID : `mlflow.sklearn.load_model(f"runs:{mlflow_run_id}/{run_relative_path_to_model}")`

그러나, 모델을 로딩한 환경과 동일한 환경이 아니라면 일반적으로 위의 정보를 얻을 수 없습니다.
대신 모델의 이름과 버전을 활용하여 모델을 로드해야 합니다.

예제 1: 이름 및 버전을 통한 로드

model_name으로 모델을 메모리에 로드하려면 아래 방법을 사용합니다.

```
model = mlflow.sklearn.load_model(f"models:{model_name}/{model_version}")
```

이 방법은 빠르고 쉽지만, 단조롭게 증가하는 모델 버전은 유연성이 부족합니다.
모델 버전의 별칭을 활용하는 것이 더 효율적인 경우가 많습니다.

예제 2: 모델 버전 별칭을 통한 로드

모델 버전 별칭은 모델 버전에 대한 사용자 정의 식별자입니다.

모델 등록 후 변경할 수 있으므로 모델 버전을 사용하는 코드에서 모델 버전을 분리합니다.

예를 들어, 프로덕션 모델에 해당하는 `production_model`이라는 모델 버전 별칭이 있다고 가정해 보겠습니다.

우리 팀이 배포할 준비가 된 더 나은 모델을 빌드할 때, 서비스 워크로드 코드를 변경할 필요가 없습니다.

대신 MLflow에서 이전 모델 버전에서 새 모델 버전으로 `production_model` 별칭을 재할당하면 됩니다.

이 작업은 UI에서 간단히 수행할 수 있습니다.

API에서 동일한 모델 이름, 별칭 이름, 새 모델 버전 ID로 `client.set_registered_model_alias`를 실행하면 됩니다.

이전 페이지에서 모델에 모델 버전 별칭을 추가했지만, 여기에는 프로그래밍 방식의 예가 있습니다.

```

import mlflow.sklearn
from mlflow import MlflowClient

client = MlflowClient()

# Set model version alias
model_name = "sk-learn-random-forest-reg-model"
model_version_alias = "the_best_model_ever"
client.set_registered_model_alias(
    model_name, model_version_alias, 1
) # Duplicate of step in UI

# Get information about the model
model_info = client.get_model_version_by_alias(model_name, model_version_alias)
model_tags = model_info.tags
print(model_tags)

# Get the model version using a model URI
model_uri = f"models:{model_name}@{model_version_alias}"
model = mlflow.sklearn.load_model(model_uri)

print(model)

```

출력

```
{'problem_type': 'regression'}
```

```
RandomForestRegressor(max_depth=2, random_state=42)
```

Mlflow 실습



LogisticRegression.ipynb

LogisticRegression_MLflow.ipynb

Tensorflow_MNIST_MLflow.ipynb

PyTorch_MNIST_MLflow.ipynb

Thank you 😊