[리눅스로 한 학기 살기 프로젝트 - 1주차 보고서]

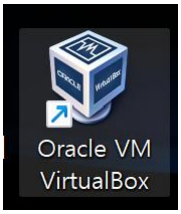
2023105412 김민서

1. 리눅스 환경

: 우분투

2. 설치 과정

1) Oracle VM VirtualBox 설치

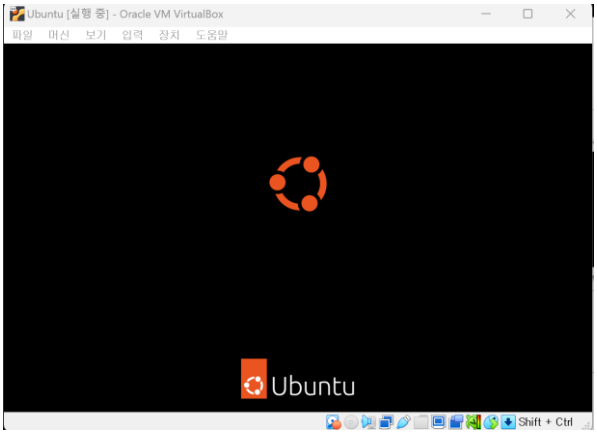


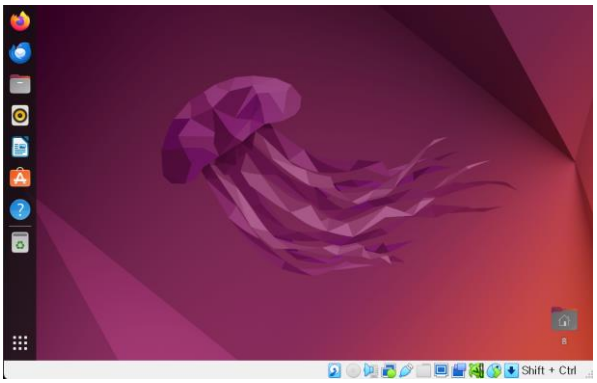
- 가장 먼저 Oracle VM VirtualBox를 설치하였다.

2) Oracle VM VirtualBox 내에 Ubuntu를 설치하고 환경을 설정하였다.

- 위 사진은 Oracle VM VirtualBox 내에 설치한 Ubuntu 그리고, 그것의 환경을 설정한 모습이다.

3) Ubuntu 실행





- 다음은 우분투를 실행하였을 때 나타나는 화면들이다.
- 첫 번째 사진은 우분투가 실행되고 있는 화면이고, 두 번째 사진은 첫 번째 사진이 끝난 뒤, 즉, 우분투가 실행된 뒤의 화면이다.

3. 환경 설정 중 어려움과 해결법

우분투를 설치하고 난 뒤에 계속해서 실행이 되지 않거나, Oracle VM VirtualBox 내에 검은색 화면이 뜨는 등의 문제점이 발생하였다.

문제를 해결하기 위해 Oracle VM VirtualBox 내의 상단에 있는 설정 파트에서 일부 설정 요소들을 변경하는 과정 등을 진행하였고, 이를 통해 성공적으로 우분투를 실행할 수 있게 되었다.

[리눅스로 한 학기 살기 프로젝트 - 2주차 보고서]

2023105412 김민서

1. 다운로드한 소프트웨어

: 주피터 노트북

2. 다운로드한 이유

주피터 노트북은 코드, 시각화, 문서화를 하나의 인터페이스에서 처리할 수 있는 개발 환경을 제공하고 있다. 이는 소프트웨어융합학과의 데이터 사이언스 트랙인 나에게, 트랙 관련 프로젝트의 프로토타이핑과 실험을 쉽고 빠르게 진행할 수 있다는 장점이 있다고 판단하였다.

또한, 주피터 노트북의 경우, 코드와 함께 분석 결과를 문서화할 수 있으며, Matplotlib, Seaborn, Plotly와 같은 시각화 라이브러리와의 통합을 통해, 데이터 분석 결과를 직관적으로 시각화할 수 있기 때문에 해당 소프트웨어를 설치하였다.

3. 다운로드 중 직면한 어려움과 해결법

처음 해보는 리눅스 설치였기에 하나부터 열까지 낯선 부분들 뿐이었다. 단순히 '$pip install jupyter'라는 명령어만 입력하면 설치가 진행되는 것인 줄 알았는데, 직접 설치하는 과정들을 거쳐보니 우분투 내에서 처음으로 소프트웨어를 설치해서인지, 이 명령어만으론 진행되지 않았다. 결국, 추가적인 조사를 통해 pip를 업그레이드하는 과정, 가상 환경을 생성하는 과정 등 필요한 과정들이 존재한다는 사실을 깨닫게 되었다.

4. 실행 코드

1) Python3 설치하기

```
kimminseo@kimminseo-VirtualBox:~$ sudo apt install python3
```

- 해당 환경 내에서 Jupyter Notebook을 설치하기 위해선, 전제 조건으로서 Python 가상 환경을 만들어야 한다고 한다는 점을 알게 되었다.
- 따라서 위와 같은 명령 코드를 통해, Ubuntu 22.04에 python3을 설치하였다.

2) pip 업그레이드하기

```
kimminseo@kimminseo-VirtualBox:~$ sudo pip3 install --upgrade pip
```

- 앞으로 이루어질 여러 단계에서 pip가 필요하다는 점을 고려해, 위와 같은 명령 코드로 pip를 업그레이드하였다.

3) virtualenv 도구 설치

```
kimminseo@kimminseo-VirtualBox:~$ sudo pip3 install virtualenv
```

- 가상 환경을 만드는 데 사용할 python의 virtualenv 도구를 설치하였다.
- 이때, virtualenv는 시스템 전역의 Python 설치를 변경하지 않고, 필요한 패키지를 설치할 수 있게 하며, 시스템 라이브러리와의 충돌을 방지하고, 시스템의 안정성을 유지하는 데 도움이 된다고 한다.

4) 새 디렉터리 & 가상 환경 생성

```
kimminseo@kimminseo-VirtualBox:~$ mkdir jupy
kimminseo@kimminseo-VirtualBox:~$ cd jupy
```

- 가상 환경을 더 잘 사용하기 위해 새 디렉터리를 만들고, 그 안에 가상 환경을 만들었다.

5) Jup_notebook 환경 생성

```
kimminseo@kimminseo-VirtualBox:~/jupy$ virtualenv jup_notebook
```

- Jup_notebook이라는 Python 환경을 생성하였다.

6) 가상 환경을 활성화

```
kimminseo@kimminseo-VirtualBox:~/jupy$ source jup_notebook/bin/activate
(jup_notebook) kimminseo@kimminseo-VirtualBox:~/jupy$ pip3 install jupyter
```

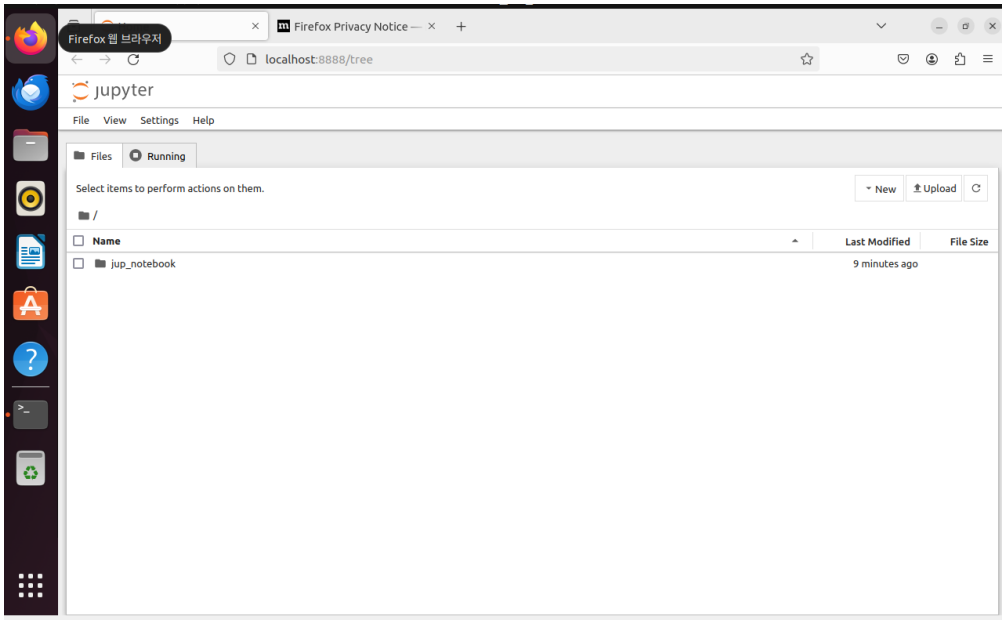
- 다음 명령을 통해 해당 가상 환경을 활성화하고 내부로 들어갔다.
- 아래줄의 (jup_notebook) 출력을 통해 가상 환경이 성공적으로 활성화되었음을 알 수 있었다.

- 가상 환경 내에 Jupyter를 설치하는데 필요한 모든 패키지를 수집, 다운로드 및 설치하였다.

7) 터미널에서 웹 인터페이스 시작

```
(jup_notebook) kimminseo@kimminseo-VirtualBox:~/jupy$ jupyter notebook
```

- 위와 같은 명령을 사용하여 터미널에서 웹 인터페이스를 시작하였다.



[리눅스로 한 학기 살기 프로젝트 - 3주차 보고서]

2023105412 김민서

1. 다운로드한 소프트웨어

: Visual Studio Code

2. 다운로드한 이유

VS Code는 파이썬, R 등 데이터 사이언스에 널리 사용되는 프로그래밍 언어들을 지원하기 때문에, 학생으로서 하나의 통합된 개발 환경에서 여러 언어로 작업할 수 있다는 장점이 있다.

또한, VS Code는 방대한 확장 프로그램 라이브러리를 제공한다는 장점이 있기에 다양한 데이터 사이언스 작업을 효율적으로 수행할 수 있을 뿐만 아니라 전에 설치하였던 주피터 노트북과의 통합으로 주피터 노트북을 직접 편집하고 실행할 수 있는 기능을 사용할 수 있다는 장점이 있기 때문에 설치하였다.

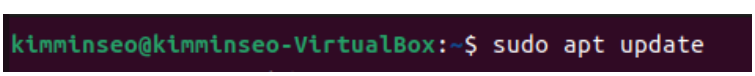
3. 다운로드 중 직면한 어려움과 해결법

가장 먼저 VS Code를 설치하는 코드인 sudo snap install code를 입력하였을 땐, 오류가 발생하였다. 이를 해결하기 위해 앞서 작성한 코드에 --classic을 덧붙였고, 그 결과 정상적으로 프로그램이 설치되었다.

Classic 모드의 경우, 애플리케이션이 시스템의 더 넓은 범위에 접근할 수 있도록 함으로써 개발 도구나 에디터와 같이 광범위한 시스템 접근 권한이 필요한 소프트웨어에 적합하다고 한다. 특히, VS Code와 같은 개발 환경은 파일 시스템, 네트워크 설정, 외부 도구 등에 대해 광범위한 접근이 필요하기 때문에 classic 모드로 설치하는 것이 일반적이라는 것을 알게 되었다.

4. 실행코드

1) 패키지 리스트 업데이트



- 해당 코드를 사용하여 패키지 리스트를 최신 상태로 업데이트하였다.

2) 필요한 종속성 설치

```
kimminseo@kimminseo-VirtualBox:~$ sudo apt install software-properties-common apt-transport-https wget
```

- VS CODE를 설치하기 전에, 필요한 종속성을 설치해야 할 수 있다는 사실에 따라 위와 같은 명령어를 입력하였다.

3) Microsoft GPG 키 추가

```
kimminseo@kimminseo-VirtualBox:~$ wget -q https://packages.microsoft.com/keys/microsoft.asc -O- | sudo apt-key add-
Usage: apt-key [--keyring file] [command] [arguments]

Manage apt's list of trusted keys

  apt-key add <file>          - add the key contained in <file> ('-' for stdin)
  apt-key del <keyid>         - remove the key <keyid>
  apt-key export <keyid>      - output the key <keyid>
  apt-key exportall           - output all trusted keys
  apt-key update              - update keys using the keyring package
  apt-key net-update          - update keys using the network
  apt-key list                - list keys
  apt-key finger              - list fingerprints
  apt-key adv                 - pass advanced options to gpg (download key)

If no specific keyring file is given the command applies to all keyring files.
```

- Microsoft의 공개 키를 시스템에 추가하여 패키지의 무결성을 검증하였다.

4) 오류 발생

```
kimminseo@kimminseo-VirtualBox:~$ sudo snap install code
오류: This revision of snap "code" was published using classic confinement and thus may perform
       arbitrary system changes outside of the security sandbox that snaps are usually confined
       to, which may put your system at risk.
```

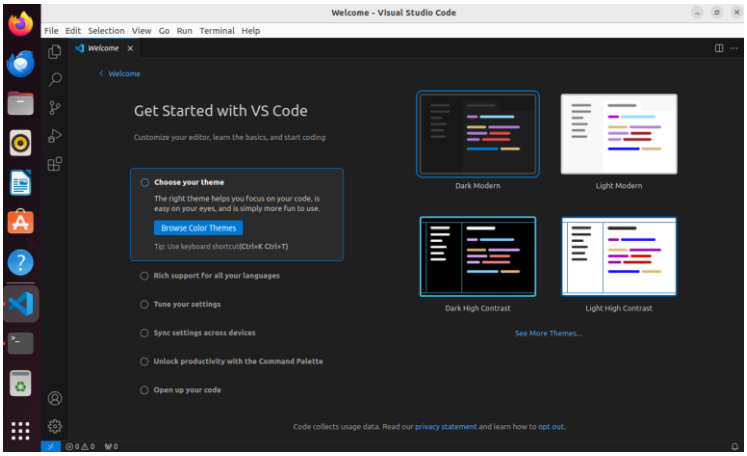
- VS Code를 설치하는 코드를 입력하였으나 오류가 발생하였다.

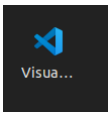
5) 오류 해결 & VS Code 설치하기

```
kimminseo@kimminseo-VirtualBox:~$ sudo snap install code --classic
```

- 기존의 설치 코드에서 --classic을 추가하였더니, 프로그램이 정상적으로 설치되었다.

6) VS Code 실행





- 첫 번째 사진은 VS Code를 실행하는 명령어이다.
- 두 번째 사진은 첫 번째 사진의 코드를 통해 실행된 VS Code의 모습이다.
- 세 번째 사진은 설치된 VS Code 앱이다.

[리눅스로 한 학기 살기 프로젝트 - 4주차 보고서]

2023105412 김민서

1. 다운로드한 소프트웨어

: Chrome

2. 다운로드한 이유

데이터 사이언스의 경우, 대용량 데이터를 처리하고, 다양한 데이터 시각화 도구를 웹 기반으로 사용하기 때문에 속도가 빠른 브라우저를 필요로 한다. 이때, chrome은 이러한 요구를 수용할 수 있는 소프트웨어이다.

또한, chrome의 웹 스토어의 경우, 다양한 확장 프로그램을 제공함으로써 사용자의 필요에 맞게 브라우저 기능을 확장할 수 있다는 특징이 있다. 이는 데이터 사이언스 학생들로 하여금 SQL 데이터 베이스 관리 등 다양한 확장 프로그램을 사용하여 작업 효율을 높일 수 있게 한다.

따라서, c빠른 속도로 다양한 확장 프로그램을 사용하여 프로젝트를 진행해보고자 chrome을 설치하게 되었다.

3. 설치코드

1) Wget 설치

```
kimminseo@kimminseo-VirtualBox:~$ sudo apt install wget
```

- 크롬을 다운로드하기 위해 필요한 wget을 설치하였다.

2) 구글 크롬 다운로드

```
kimminseo@kimminseo-VirtualBox:~$ wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
```

```
google-chrome-stable_current_amd 100%[===================================================>] 102.64M   273KB/s    / 2m 2s
```

- 위와 같은 명령어를 사용하여 최신 버전의 구글 크롬 '.deb' 패키지를 다운로드 하였다.

3) 패키지 설치

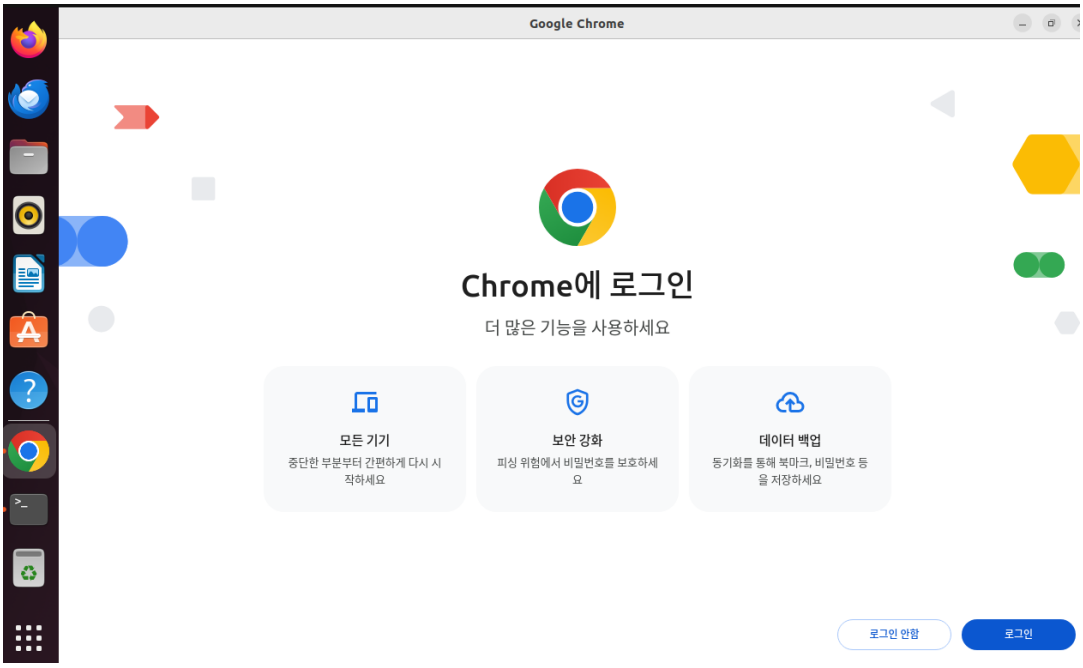
```
kimminseo@kimminseo-VirtualBox:~$ sudo dpkg -i google-chrome-stable_current_amd64.deb
```

```
kimminseo@kimminseo-VirtualBox:~$ sudo apt install -f
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다... 완료
상태 정보를 읽는 중입니다... 완료
```

- 다운로드한 '.deb' 파일을 'dpkg'를 사용하여 설치하였다.

4) 크롬 실행

```
kimminseo@kimminseo-VirtualBox:~$ google-chrome
Created TensorFlow Lite XNNPACK delegate for CPU.
```



- 첫 번째 사진과 같은 코드를 통해 크롬을 실행하도록 하였다.
- 두 번째 사진은 첫 번째 사진에 나타난 코드의 결과로 나타난 창이다.
- 세 번째 사진은 우분투에 크롬이 설치된 모습이다.

[리눅스로 한 학기 살기 프로젝트 - 5주차 보고서]

2023105412 김민서

1. 다운로드한 소프트웨어

: LibreOffice Draw

2. 다운로드한 이유

학교 강의 시간에 drawing tool들 중 하나로서 LibreOffice Draw라는 프로그램을 알게 되었다. 프로젝트를 진행함에 있어서는 코드의 구현 방식이나 흐름을 나타내는 것이 중요한 개발자에게, 다이어그램과 흐름도 작성을 도와주면서도 다양한 파일 형식을 지원하는 LibreOffice Draw의 특징은 매우 유용하게 다가올 수 있음을 깨달았다.

따라서 리눅스 환경 내에서 해당 프로그램을 직접 설치해봄으로써 "design_by_figure"를 경험해보고자 해당 프로그램을 설치하였다.

3. 실행코드

1) 시스템 패키지 목록 업데이트

```
kimminseo@kimminseo-VirtualBox:~$ sudo apt update
```

- 패키지 목록을 업데이트하였다.

2) LibreOffice Draw 설치

```
kimminseo@kimminseo-VirtualBox:~$ sudo apt install libreoffice-draw
```

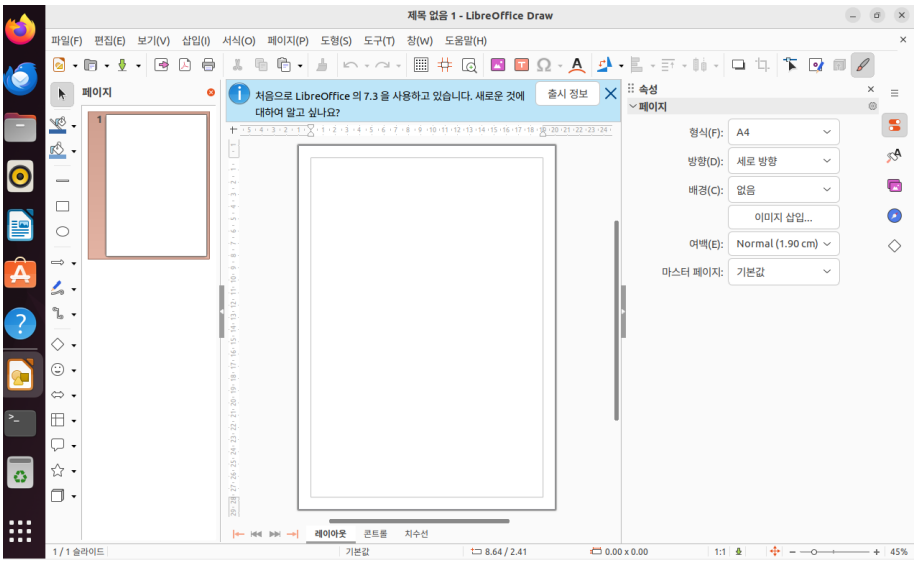
- 다음 코드를 통해 LibreOffice Draw를 설치하였다.
- 이때, LibreOffice Draw는 LibreOffice 패키지에 포함이 되어있다는 점을 고려할 때, 일반적으로 libreoffice-draw 패키지 이름으로 설치가 가능하다고 한다.
- 실제로 위와 같은 코드를 작성하자, 오류 없이 정상적으로 프로그램이 설치되었다.

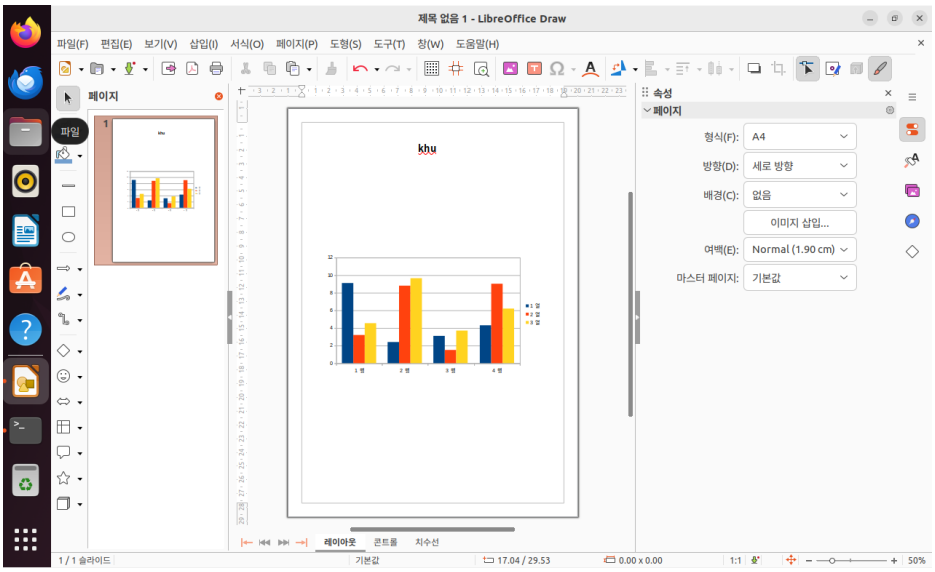
3) LibreOffice Draw 실행하기

```
kimminseo@kimminseo-VirtualBox:~$ libreoffice --draw
```

- 위와 같은 코드를 터미널 내에 작성함으로써 LibreOffice Draw 프로그램이 실행되도록 하였다.

4) 실행 결과





- 새 파일을 생성하고, 편집하는 과정을 거쳤다.
- 첫 번째 사진은 프로그램을 실행하자마자 나타난 화면이고, 두 번째 사진은 해당 프로그램에서 문자나 그래프를 삽입해본 모습이다.

[리눅스로 한 학기 살기 프로젝트 - 6주차 보고서]

2023105412 김민서

1. 다운로드한 소프트웨어

: Slack

2. 다운로드한 이유

지난 강의 시간, 소프트웨어 프로젝트 내에서 팀플의 중요성을 알게 되었다. 특히, 원활한 팀플을 위한 도구 중 하나로 slack이라는 것을 알게 되었는데, 지난 학기 디자인적 사고라는 팀플 수업에서 해당 도구를 사용했더라면 팀원들 간의 소통이 더욱 편리했을 것 같다는 생각이 들었다.

이번 기회를 통해 slack이라는 소통 도구를 리눅스 터미널 내에서 직접 깔아보고 사용해봄으로써 앞으로 마주하게 될 수많은 팀플들을 잘 진행해 나가고자 slack 앱을 설치하게 되었다.

3. 다운로드 중 직면한 어려움과 해결법

run configure hook of "slack" snap if present

설치 중 위와 같은 메시지를 마주하게 되었다. 아무런 동작이 하지 않는 것처럼 보였고, 오랜 시간동안 위와 같은 메시지만 뜬 채 변하는 것이 없어 당황했다. 인터넷에 찾아보니 해당 메시지는 Slack Snap 패키지가 설치되거나 업데이트된 후 구성 스크립트(또는 'hook')를 실행하려고 시도할 때 나타날 수 있다는 사실을 알게 되었다. 이 hook은 일반적으로 패키지의 특정 구성을 설정하거나 필요한 추가 단계를 수행하는 데 사용되며 자동으로 처리되기 때문에 사용자가 직접 개입할 필요가 없다는 사실을 알게 되었다.

실제로 아무런 조작도 하지 않은 채 기다리니 slack의 설치가 완료되었다.

4. 실행코드

1) Snap 버전 확인하기

```
kimminseo@kimminseo-VirtualBox:~$ snap version
snap    2.61.1
snapd   2.61.1
series  16
ubuntu  22.04
kernel  6.5.0-25-generic
```

- Slack을 설치하기 위해, 우분투 내에 기본적으로 설치되어 있는 snap의 버전을 확인하였다.

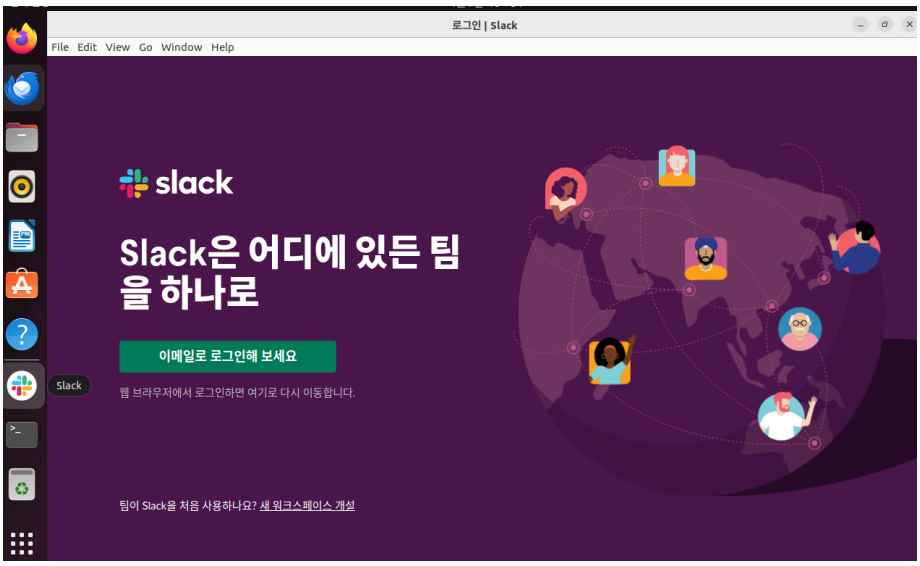
2) Slack 설치하기

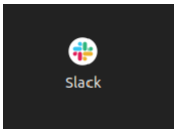
```
kimminseo@kimminseo-VirtualBox:~$ sudo snap install slack --classic
```

- Snap을 사용하여 slack을 설치하였다.
- 이때 '–classic'은 slack이 시스템의 일부 필수 요소에 접근할 수 있도록 허용하는 것이다.

3) 작성

```
kimminseo@kimminseo-VirtualBox:~$ slack
```





- 아래 마지막 사진은 우분투 내에 slack 앱이 설치된 모습이고, 첫번째 사진은 터미널 내에서 명령어를 통해 slack을 실행시키는 코드이며 두번째 사진은 그 명령어를 통해 실행된 slack의 모습이다.

[리눅스로 한 학기 살기 프로젝트 - 7주차 보고서]

2023105412 김민서

1. 다운로드한 소프트웨어

: QGIS

2. 다운로드한 이유

QGIS는 오픈소스 지리정보시스템 툴으로, 데이터의 공간적 분석과 시각화에 유용하다는 특징을 가지고 있다고 한다. 또한, QGIS는 인구 통계 데이터, 기후 데이터, 지리 데이터 등을 결합하여 보다 풍부한 인사이트를 제공하는 등, 다양한 형식과 출처의 데이터를 통합하고 분석할 수 있는 기능을 제공한다.

해당 소프트웨어를 통해 지리와 관련된 데이터를 다룰 수 있는 기회를 접할 수 있을 뿐 아니라 관련된 다양한 데이터와 결합할 수 있는 좋은 기회가 될 것이라는 생각에 설치하였다.

3. 다운로드 중 직면한 어려움과 해결법

Qgis를 설치한 이번 주차에서는 이전 주차들과는 다르게 공식 저장소(리포지토리)를 사용하는 방법을 사용하였다. 낯선 방법이었고, 설치 순서나 작동 구현이 익숙하지 않아 힘들었다.

이를 해결하기 위해, 해당 코드 구현의 방식에 대해 조사를 진행하며 살펴보았다. 가장 먼저 진행했던 리포지토리 추가의 경우, 소프트웨어 저장소의 주소를 시스템의 패키지 관리 목록에 추가하는 것임을 알게되었다. 또한, 공개 키를 추가하는 과정은 패키지의 출처 등을 검증하기 위해 사용되는 시스템을 추가함으로써 저장소의 콘텐츠가 변경하지 않았음을 보증할 수 있는 단계임을 알게 되었다.

4. 실행코드

1) 리포지토리 추가

```
kimminseo@kimminseo-VirtualBox:~$ sudo add-apt-repository "deb https:// qgis.ogr/ubuntu $(1sb_release -c -s) main"
```

- Ubuntu 기반의 시스템에서는 QGIS를 설치하기 이전에 공식 QGIS 리포지토리를

시스템에 추가해야 하므로 위와 같은 코드를 입력하였다.

2) 키 가져오기

```
kimminseo@kimminseo-VirtualBox:~$ wget -o - https://qgis.org/downloads/qgis-2023.gpg.key | gpg --import
```

```
kimminseo@kimminseo-VirtualBox:~$ gpg --export --armor 51F523511C7028C3 | sudo apt-key add -
```

- 리포지토리의 공개 키를 가져옴으로써 패키지의 신뢰성을 검증할 수 있도록 하였다.

3) 패키지 목록 업데이트

```
kimminseo@kimminseo-VirtualBox:~$ sudo apt update
```

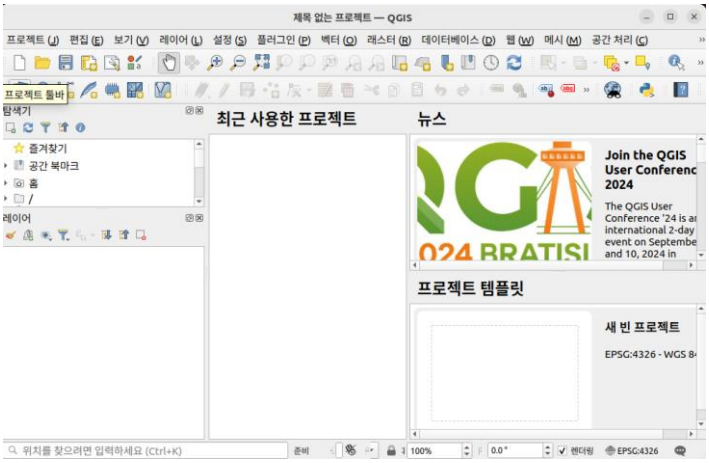
- 패키지 목록을 업데이트하였다.

4) QGIS

```
kimminseo@kimminseo-VirtualBox:~$ sudo apt install qgis-plugin-grass
```

- QGIS를 설치하였다.

5) 결과



- qgis를 입력함으로써 앱을 실행하였다.

[리눅스로 한 학기 살기 프로젝트 - 8주차 보고서]

2023105412 김민서

1. 다운로드한 소프트웨어

: Orange

2. 다운로드한 이유

Orange의 경우, 직관적인 비주얼의 인터페이스를 제공한다는 특징이 있으며 이는 프로그래밍 경험이 적은 학생들에게 매우 유용하다는 점을 알게 되었다. 또한, 이는 데이터 탐색과 결과를 시각화하는데 필요한 다양한 도구를 갖추고 있다고 하는데, 이전에 진행했던 동아리 내의 데이터 분석 프로젝트에서 데이터 탐색 과정에 흥미를 느꼈던 경험이 떠오르며 한 번 해당 프로그램을 설치하고 작동해보고 싶다는 생각이 들었다.

해당 프로그램을 설치하고, 작동해봄으로써 데이터 분석, 및 탐색과 관련된 경험을 쌓을 수 있을 것 같아 Orange를 설치하였다.

3. 다운로드 중 직면한 어려움과 해결법

파이썬의 설치 여부를 확인할 때나, python3 --version이라 작성해야 하는데, 계속해서 python --version이라 작성함에 따라 파이썬이 설치되어 있지 않다는 답변을 받았다. 초반엔 python이라고 작성했다는 사실을 깨닫지 못해 헤매기도 했으나, 다시 코드를 찬찬히 살펴보며 3을 적지 않았다는 이유를 깨닫게 되었다.

```
kimminseo@kimminseo-VirtualBox:~$ python --version
명령어 'python' 을(를) 찾을 수 없습니다. 다음 명령어로 시도하시겠습니까:
```

위는 python --version이라 작성하였을 때, 발생한 답변을 담은 사진이다. 당연한 이야기이지만, 코드를 실행하였을 때 오류가 발생하거나 원하는 답변이 나오지 않았을 때는 침착함을 유지하며 다시 한 번 차근차근히 코드를 돌아보아야 함을 몸소 느낀 과정이었다.

4. 실행코드

1) Python3, pip의 설치 여부 확인

```
kimminseo@kimminseo-VirtualBox:~$ python3 --version
Python 3.10.12
```

```
kimminseo@kimminseo-VirtualBox:~$ pip --version
pip 24.0 from /home/kimminseo/.local/lib/python3.10/site-packages/pip (python 3.10)
```

- Python3와 pip의 설치 여부를 확인하기 위해 위와 같은 코드를 작성하였고, 그 아래의 결과들을 통해 이미 설치가 되어있음을 확인하였다.

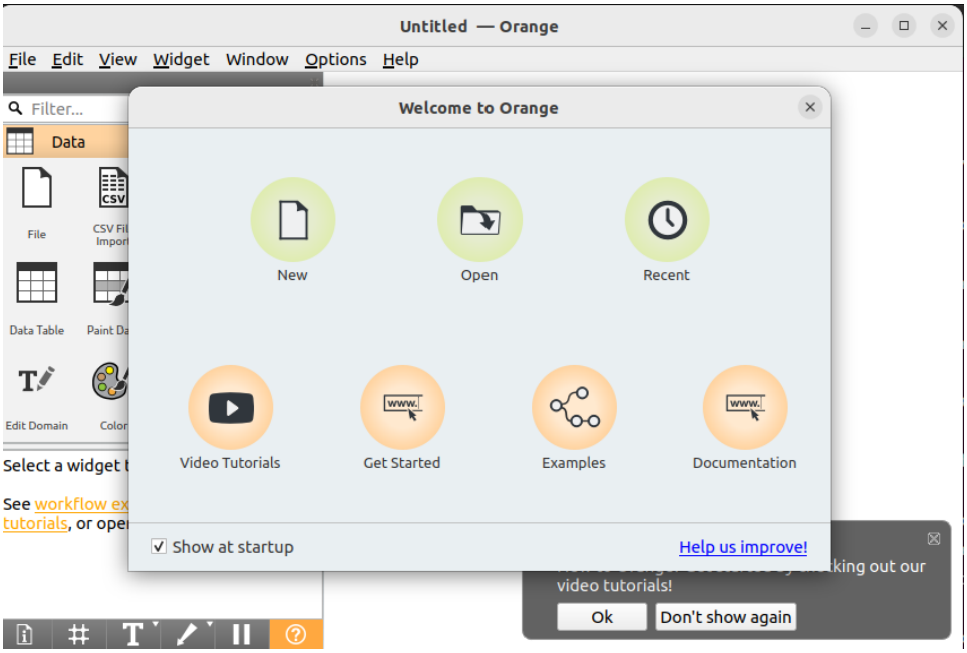
2) Orange 설치

```
kimminseo@kimminseo-VirtualBox:~$ pip install orange3
```

- pip을 사용해 orange를 설치하였다.

3) Orange 실행

```
kimminseo@kimminseo-VirtualBox:~$ orange-canvas
```



- 위와 같은 코드를 작성함으로써 Orange를 실행하였고, 두 번째 사진은 그 코드의 결과로 실행된 Orange 프로그램의 화면이다.

[리눅스로 한 학기 살기 프로젝트 - 9주차 보고서]

2023105412 김민서

1. 다운로드한 소프트웨어

: Atom

2. 다운로드한 이유

우선, atom의 경우 파이썬, R, SQL, 자바스크립트 등 데이터 사이언스에 자주 사용되는 다양한 프로그래밍 언어를 지원하기에 한 플랫폼에서 여러 언어로 작업할 수 있다는 장점을 가지고 있다. 또한, Atom은 Github와 긴밀하게 통합되어 있는데, 이는 코드의 버전 관리가 중요한 데이터 사이언스 프로젝트에서 특히 유용하다는 특징을 가지고 있다고 한다. 구체적으로는 atom 내에서 직접 git 명령을 실행하고, 변경 사항을 추적하며 이를 github에 코드를 넣을 수 있다고 한다.

위와 같은 atom의 장점에 기반해, 해당 앱을 설치함으로써 다양한 tool들을 긴밀하게 연결해가며 사용할 수 있을 것이라는 생각에 해당 앱을 설치하게 되었다.

3. 다운로드 중 직면한 어려움과 해결법

Atom을 설치하는 방법은 크게 두 가지가 있음을 조사하였다. 첫 번째는 Snap 패키지를 통한 설치이며, 두 번째는 공식 PPA를 통한 설치가 있었다. 이때, 두 번째 방식을 선택하면, Atom의 최신 버전을 보다 쉽게 유지할 수 있다는 내용을 보고, PPA를 통한 설치를 진행하였다. 그러나, Atom 패키지를 찾을 수 없다는 답변을 받게 되었고, 그 이유들을 찾아보니, 최근에 Atom 텍스트 에디터가 공식적으로 지원이 종료됨에 따라 기존의 PPA를 통한 설치가 작동하지 않을 수 있다는 사실을 알게 되었다.

따라서, 다른 방법인 snap 패키지를 통한 설치를 진행하게 되었다.

4. 실행코드 -> PPA를 통한 설치

1) PPA 추가

```
kimminseo@kimminseo-VirtualBox:~$ sudo add-apt-repository ppa:webupd8team/atom
```

- Atom의 공식 PPA를 시스템에 추가하였다.

2) 패키지 목록 업데이트

```
kimminseo@kimminseo-VirtualBox:~$ sudo apt update
```

- PPA를 추가한 후, 패키지 목록을 업데이트하였다.

3) Atom 설치하기

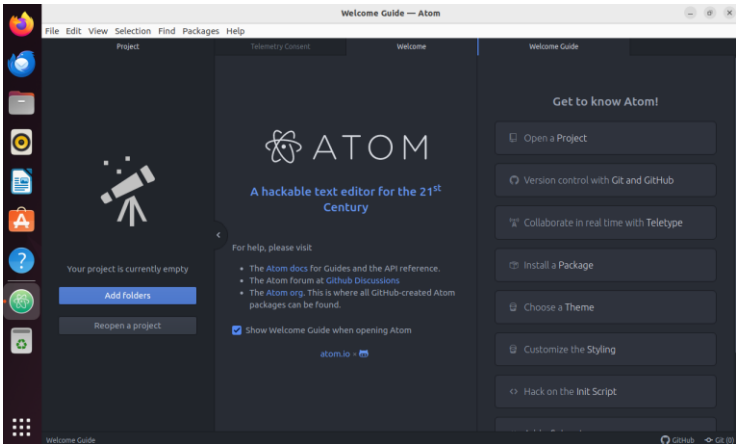
```
kimminseo@kimminseo-VirtualBox:~$ sudo apt install atom
```

```
E: atom 패키지를 찾을 수 없습니다
```

- Atom 설치를 시도하였다.
- 그러나 atom 패키지를 찾을 수 없다는 답변을 얻게 되었다.
- 그 이유는 Atom 텍스트 에디터가 공식적으로 지원이 종료됨에 따라, 기존의 PPA를 통한 설치가 작동하지 않을 수 있다는 사실일 수 있음을 알게 되었다.
- 따라서, PPA를 통한 설치가 아닌 snap 패키지를 통한 설치로 변경하여 진행하였다.

5. 실행코드 -> snap 패키지를 통한 설치

```
kimminseo@kimminseo-VirtualBox:~$ sudo snap install atom --classic
```



- Snap 패키지를 통해 atom을 설치하였다.

- 이때, 코드 내에 사용된 --classic 플래그는 atom이 시스템의 다른 부분과 통합될 수 있도록 해준다.
- 두 번째 사진은 해당 코드를 통해 설치된 atom 앱의 모습이다.

[리눅스로 한 학기 살기 프로젝트 - 10주차 보고서]

2023105412 김민서

1. 다운로드한 소프트웨어

: notion

2. 다운로드한 이유

프로젝트를 진행함에 있어서 notion은 편리함을 도모해줄 수 있는 유용한 tool 중 하나라고 생각했다. notion은 개인 페이지를 통해 다양한 단계의 내용을 체계적으로 정리할 수 있다는 장점이 있지만, 팀 페이지에서 또한 매우 유용하게 사용이 가능하다는 장점에 이끌려 해당 앱을 설치하였다. notion에서는 팀원 전체가 편집자의 권한을 부여받을 수 있으며, 권한이 허용된 페이지에 대해 수정이 가능하다는 점에서 팀 프로젝트를 진행할 때 보다 협력적인 에이자일을 도모할 수 있기에 해당 앱을 설치하였다.

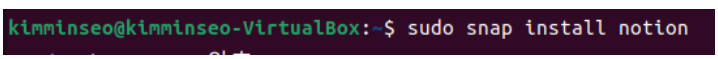
3. 다운로드 중 직면한 어려움과 해결법

단순히 sudo snap install notion이라는 코드를 작성하였을 때, notion을 찾을 수 없다는 답변과 함께, notion이 설치되지 않았다.

그 이유와 해결 방법을 찾는 과정을 통해 앞서 작성했던 코드 뒤에 -snap-reborn이라는 코드를 추가적으로 붙여야 함을 알게 되었고, 결론적으로 sudo snap install notion-snap-reborn이라는 코드를 작성함으로써 정상적으로 앱 설치를 진행할 수 있었다.

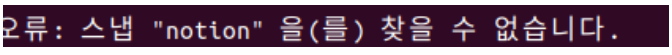
4. 실행코드

1) 첫 번째 시도



- sudo snap install notion이라는 코드를 작성하였다.

2) 첫 번째 시도



- 스냅 notion을 찾을 수 없다는 오류가 발생하였다.

3) 두 번째 시도

```
kimminseo@kimminseo-VirtualBox:~$ sudo snap install notion-snap-reborn
```

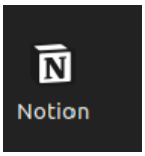
- 첫 번째 시도에서 작성하였던 코드 뒤에 -snap-reborn을 추가로 작성하였다.

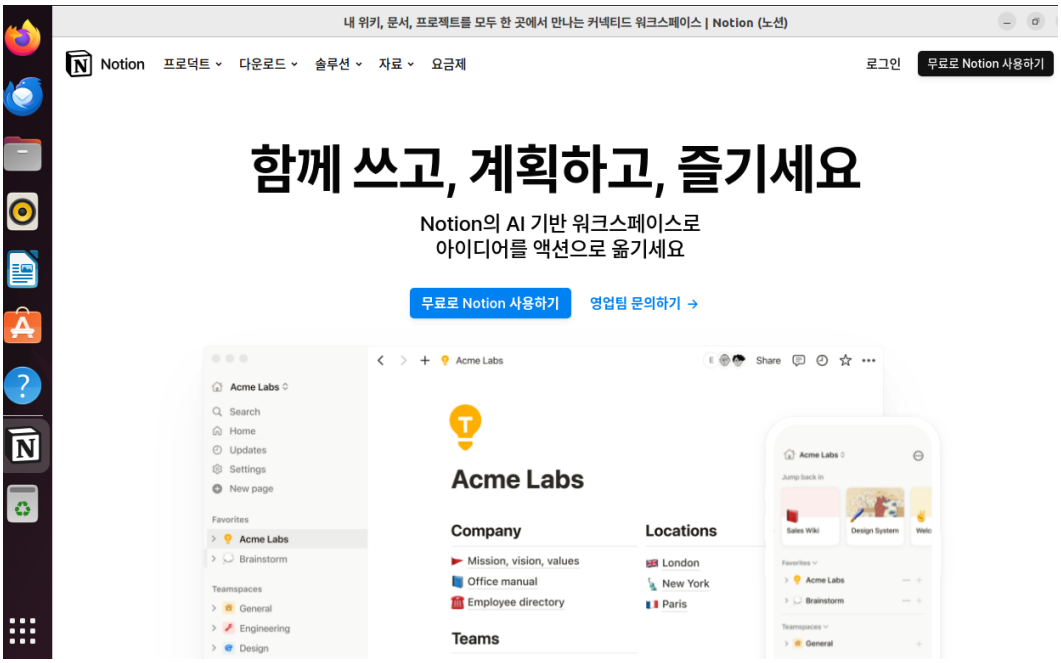
4) 두 번째 시도

```
notion-snap-reborn1.2.0 from Maxime (sunshio) installed
```

- 첫 번째 시도와는 달리 정상적으로 notion 앱이 설치되었다.

5) 설치된 노션





- 위 사진들은 우분투 내에 설치된 노션 앱의 모습이다.

[리눅스로 한 학기 살기 프로젝트 - 11주차 보고서]

2023105412 김민서

1. 다운로드한 소프트웨어

: Discord

2. 다운로드한 이유

디스코드는 팀 프로젝트를 진행함에 있어서 매우 유용하다. 아직까지 많은 팀 프로젝트를 진행해본 경험은 없지만, 지금껏 팀프로젝트를 진행하며 느꼈던 점은 팀원들 간 시간 맞추기가 상당히 어렵다는 점이다. 각자 시간표가 다르고, 수업 외에도 각자의 일정이 정해져있기 때문에, 대면 회의 시간 맞추기가 어려웠던 적이 많다.

그러한 상황에서, 디스코드와 같은 비대면 화상 회의 앱을 사용하면, 대면 회의보다 비교적 시공간의 제한이 적다는 장점이 있을 것이라 생각했다. 디스코드는 음성으로 말할 수도 있지만, 채팅창과 같은 요소를 사용하여 말할 수 없는 상황에서 또한 회의에 참여할 수 있다는 장점이 있다.

3. 실행코드

1) 설치 코드

```
kimminseo@kimminseo-VirtualBox:~$ sudo snap install discord
```

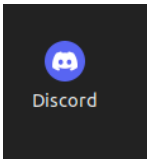
- 위와 같은 코드를 작성함으로써 discord 앱을 설치하였다.

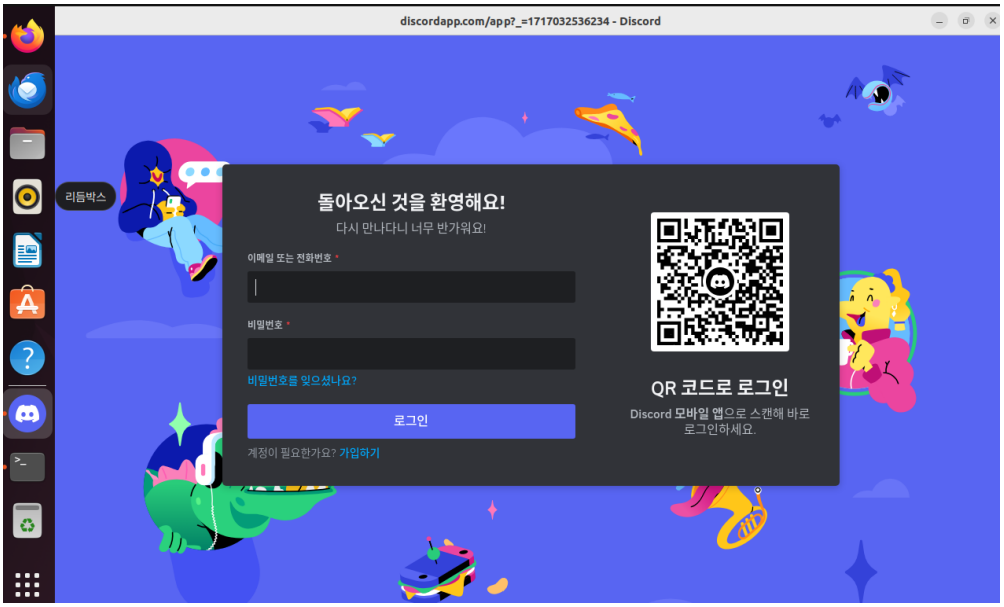
2) 설치 코드 결과

```
discord0.0.54 from Snapcrafters✪ installed
```

- 디스코드가 설치되었다는 답변을 얻었다.

3) 설치된 모습





- 위의 사진들은 설치된 디스코드 앱의 모습들이다.

4. 다운로드 중 궁금했던 점과 해결

전 주차에서는 notion 앱을 설치하였었다. Notion과 discord 모두 snap 패키지를 사용하여 다운로드를 진행하였는데, 이 과정에서 궁금증이 발생하였다. Notion의 경우, sudo snap install notion이라고 코드를 작성하였을 때 오류가 발생하여 해당 코드 뒤에 -snap-reborn을 추가적으로 붙임으로써 해결하였었는데, discord는 그저 snap install discord만 작성하더라도 정상적으로 설치가 된 점이었다.

위와 같은 상황이 발생하는 이유에 대해 알아보니, 이는 공식적으로 제공되는 snap 패키지 여부에 있음을 알게되었다. Discord의 경우는 discord 라는 이름으로 sanp 패키지가 등록되어 있기 때문에 이 이름만을 사용하여 설치할 수 있었을 뿐 아니라 공식적으로 Snapcraft에서 관리되기 때문에 단순히 snap install discord로 설치할 수 있었던 것이었다. 반면, notion-snap-reborn 같은 경우는 공식적으로 제공되지 않는 서드파티 패키지였다. 노션 자체가 공식적으로 리눅스용 snap 패키지를 제공하지 않으므로 커뮤니키에서 개발한 대안적인 패키지를 사용할 수밖에 없다는 점이었다.

[리눅스로 한 학기 살기 프로젝트 - 12주차 보고서]

2023105412 김민서

1. 소감문

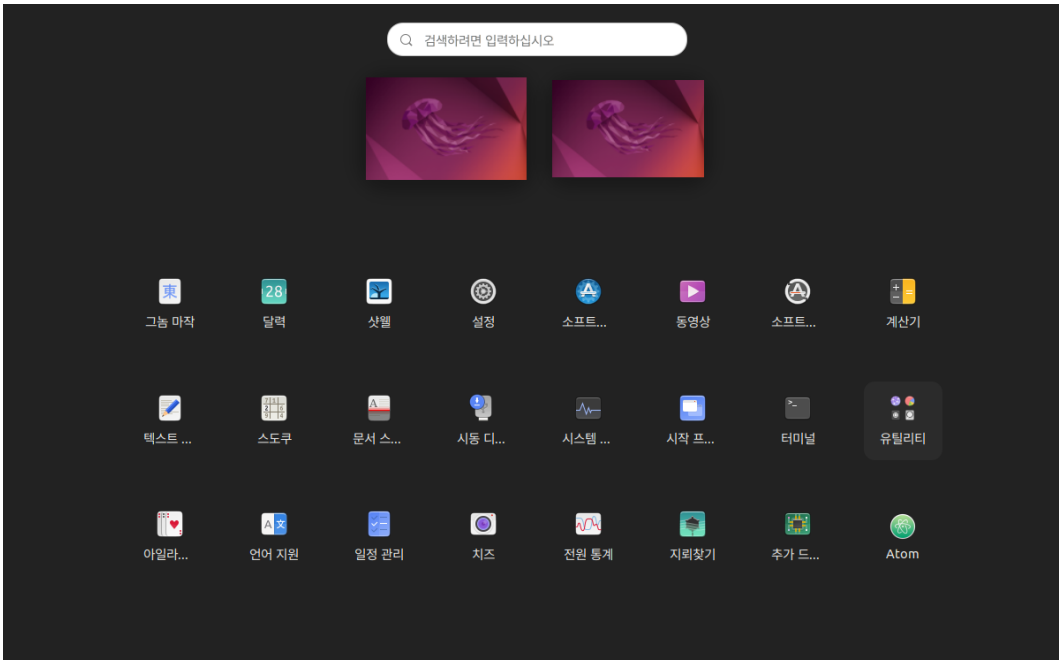
리눅스를 사용해본 적도, 터미널로 프로그램을 설치해본 적도 없었기에, 해당 '리눅스로 한 학기 살기 프로젝트'의 진행은 낯선 것들 투성이였다. 가상 머신 내에 우분투를 설치하는 것부터 시작해 Sudo, snap, wget 등과 같은 모든 것들은 무슨 목적으로 쓰이는 어떤 언어인지조차 알 수 없었다. 사실, 초반엔 내가 10개의 프로그램을 잘 설치할 수 있을지 걱정과 두려움이 느껴지기도 했다.

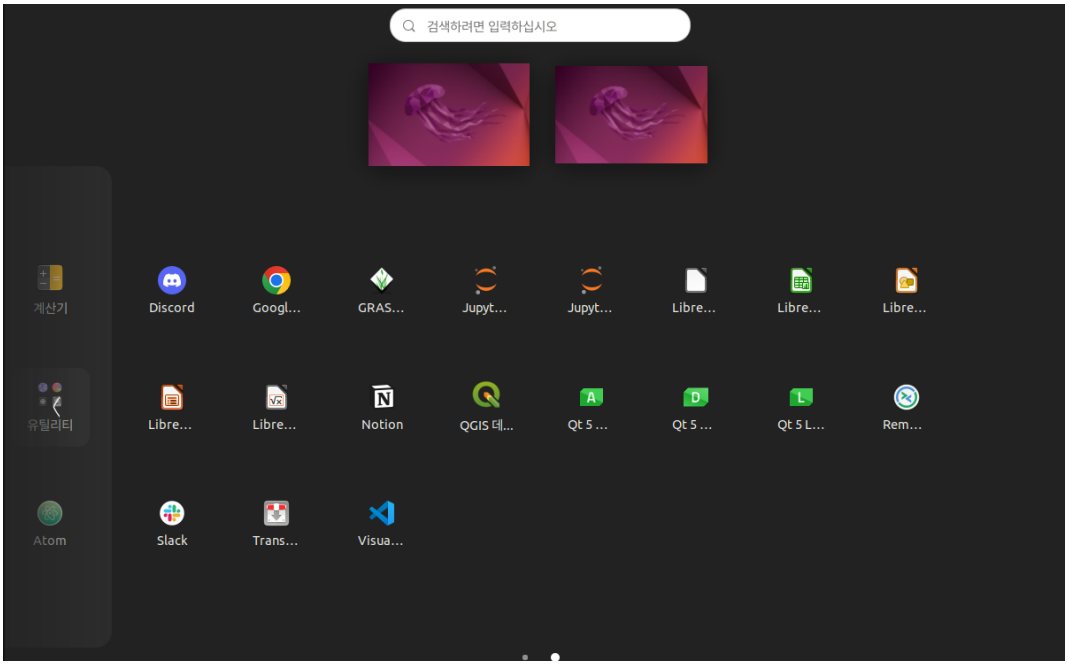
그러나, 프로그램을 설치하는 약 10주간, 매주 하나씩 프로그램을 설치하며 리눅스 터미널을 이용한 프로그램 설치 방식에 조금씩 익숙해지기 시작했다. 처음보던 낯선 언어들이 반복적으로 사용되었고, 상황에 따라 사용되는 단어들이 반복적으로 나타났다.

그러나, 이런 단어들과 설치 구현 방식 등이 익숙해질 때 즈음, 또다른 문제점이 발생했다. 프로그램에 따라 어떤 경우는 해당 방식을 사용해도 정상적으로 설치가 되는 반면, 같은 방식을 사용하였음에도 불구하고 정상적으로 설치가 되지 않고, 오류가 발생하는 경우도 더러 있었다. 다양한 방법을 시도해보며 에러를 해결하기 위해 노력하였고, 그 결과, 전에 사용했던 코드에 --classic이라는 코드를 덧붙이는 등 추가적인 코드를 붙여야 하는 경우도 존재함을 알게 되었다.

구체적인 예시로, 노션을 설치할 때에는 -snap-reborn을 덧붙임으로써 오류를 해결하였고, 슬랙이나 VS Code의 경우에는 --classic이라는 코드를 덧붙임으로써 오류를 해결하기도 하였다.

이런 상황을 겪고 나니 이렇게 프로그램에 따라 코드가 추가되는 경우와 그렇지 않은 경우는 어떤 차이점이 있는지 궁금해지기도 했다. 궁금증을 해소하기 위해 알아보니, Classic 모드의 경우, 애플리케이션이 시스템의 더 넓은 범위에 접근할 수 있도록 함으로써 개발 도구나 에디터와 같이 광범위한 시스템 접근 권한이 필요한 소프트웨어에 적합하기 때문에 광범위한 접근이 필요한 프로그램의 경우 classic 모드로 설치하는 것이 일반적이라는 것을 알게 되었다. 또한, -snap-reborn을 덧붙여야 하는 경우는 원래 snap 패키지와 비교하여 개선된 성능이나 추가 기능 등을 포함한 버전으로 다운로드해야 하거나, 공식 채널을 통해 제공되지 않는 경우 혹은 커뮤니티가 주도적으로 개발할 때 사용될 수 있음을 알게 되었다.





위 사진은 우분투 내에 설치된 프로그램들 중 일부분이 표현된 화면이다. 처음엔 상당히 많이 비어있던 이 부분이 한 주 한 주가 거듭될수록 채워졌고, 그 채워진 모습을 볼 때마다 뿌듯함 또한 느낄 수 있었다.

처음엔 낯설기만 하고, 두렵기만 했던 프로젝트에 대해 시간이 지날수록 새롭고 신기한 감정과 함께 '재밌다'는 생각을 느낄 수 있었던 소중한 경험을 얻었다는 생각이 든다. 앞으로는 해당 프로젝트는 아니라고 하더라도, 리눅스에 대해 더 자세히 알아보고, 터미널을 이용해 앱 및 프로그램을 설치해보는 등 지금보다 한층 더 주도적으로 공부를 이어나가보고 싶다.