

PBL 第 1 回レポート

22X3057 菅原颯太

1. 目的

第7回までの授業において、実際の開発環境で必要となる基本的なスキルを身に付けることを目標とし、Linux 環境の導入、Google Cloud での仮想サーバの設定と管理方法と SSH 接続方法、Python のランタイムバージョン管理方法、Git と GitHub を用いたコードのバージョン管理方法、FastAPI を利用したローカルと仮想マシン上での Web サーバの構築方法について、その仕組みを学び、コードを実行して理解する。

2. 方法

2-1 Linux 環境の導入

最初に WSL をインストールする。Windows コマンドプロンプトを右クリックして管理者として実行し、以下のコマンドを入力した。

```
ws1 --install
```

次に、ディストリビューションとして Ubuntu をインストールした。

```
ws1 --install -d Ubuntu
```

次に、バージョンを確認した。

```
ws1 -l -v
```

2-2 Google Cloud での仮想サーバの作成と SSH 接続

Google Cloud で新しいプロジェクトからプロジェクトを作成し、Compute Engine からインスタンスを作成を選択し、名前、リージョン、マシンの構成などを設定して VM インスタンスを作成した。

次に、作成したサーバに SSH 接続を行った。最初に、VSCode で Ubuntu に WSL 接続し、

```
mkdir ~/.ssh
```

でディレクトリを作成した。次に、

```
cd .ssh
```

で移動した後、

```
ssh-keygen -t rsa -C "sugawara"
```

で公開鍵 id_rsa.pub と秘密鍵 id_rsa を作成した。次に、

```
cat id_rsa.pub
```

で公開鍵の中身を確認し、コピーして VM インスタントに公開鍵として登録した。最後に、

```
ssh sugawara@35.203.152.47 -i ~/.ssh/id_rsa
```

を実行後、設定したパスワードを入力して仮想サーバに接続した。

2-3 Python ランタイムバージョン管理

Python ランタイムバージョン管理ツールとして、mise を導入した。最初に、

```
curl https://mise.run | sh
```

で mise をインストールした。次に、

```
~/.local/bin/mise --version
```

でインストールされた mise のバージョンを確認した。次に、

```
echo 'eval "$( ~/.local/bin/mise activate bash )"' >> ~/.bashrc
```

で Bash シェルが起動するたびに mise が有効になるように設定した。次に、

```
sudo apt update; sudo apt install build-essential libssl-dev zlib1g-dev \
libbz2-dev libreadline-dev libsqlite3-dev curl git \
libncursesw5-dev xz-utils tk-dev libxml2-dev libxmlsec1-dev libffi-dev \
liblzma-dev
```

でパッケージリストを最新の状態に更新した。次に、

```
mise use -q python@3.11
```

で mise を使用して Python のバージョン 3.11 をインストールし、このバージョンを使用するように設定した。次に、VSCode で作成した pbl ディレクトリ内に以下の内容の main.py ファイルを作成した。

```
print("hello world")
```

次に、VSCode の右下のインタープリターパスを /home/sugawara/.local/share/mise/install/python/3.11/bin/python に設定し、▶を押して実行した。

2-4 Git/GitHub でのコードのバージョン管理

Git と GitHub を用いて main.py ファイルのコードのバージョンを管理した。最初に、pbl ディレクトリに移動し、

```
git init
```

で pbl ディレクトリを Git リポジトリとして初期化した。次に、

```
git add main.py
```

で main.py ファイルをステージに追加した。次に、

```
git status
```

でリポジトリの現在の状態を確認した。次に、

```
git commit -m "first commit"
```

でステージに追加された変更をコミットした。次に、

```
git log
```

でコミット履歴を確認した。次に、main.py を以下の内容に変更した。

```
print("hello")
```

次に、

```
git add main.py
```

で変更された main.py を再びステージに追加した。次に、

```
git commit -m "second commit"
```

で変更をコミットした。次に、

```
git log
```

で再びコミット履歴を確認した。次に、

```
git reset --hard HEAD^
```

で直前のコミットを取り消し、一つ前のコミットに戻した。次に、

```
git log
```

で再びコミット履歴を確認した。

続いて ssh のディレクトリに移動し、GitHub 用の鍵を

```
ssh-keygen -t rsa -f id_rsa_git
```

で作成した。次に、以下の内容の config ファイルを作成した。

```
Host github.com github
  HostName github.com
  IdentityFile ~/.ssh/id_rsa_git
  IdentitiesOnly yes
  IPQoS lowdelay
  User git
```

次に、GitHub のアカウントにログインし、設定から SSH 公開鍵を登録し、Create Repository からレポジトリを作成した。次に、pbl ディレクトリに移動し、

```
git remote add origin https://github.com/skz0407/New-repository.git
```

でリモートリポジトリの URL をローカルリポジトリに追加した。次に、

```
git branch -M main
```

でブランチの名前を main にした。次に、

```
git push -u origin main
```

でローカルリポジトリの内容をリモートリポジトリにプッシュした。

2-5 FastAPI による Web サーバの構築

最初に、

```
pip install fastapi
```

で FastAPI をインストールした。次に、main.py を以下の内容に変更した。

```
from typing import Union

from fastapi import FastAPI
```

```

app = FastAPI()

@app.get("/")
def read_root():
    return {"Hello": "World"}

@app.get("/items/{item_id}")
def read_item(item_id: int, q: Union[str, None] = None):
    return {"item_id": item_id, "q": q}

```

次に、

```
fastapi dev main.py
```

で FastAPI サーバを構築した。次に、ブラウザ上で `http://127.0.0.1:8000` にアクセスし、サーバにアクセスできているか確認した。最後に、作成した `main.py` を以下のコードで GitHub にプッシュした。

```

$ git add main.py
$ git commit -m "new commit"
$ git push

```

続いて、

```
ssh sugawara@35.203.152.47 -i ~/.ssh/id_rsa
```

で仮想サーバに SSH 接続し、

```
git clone https://github.com/skz0407/New-repository
```

で GitHub からレポジトリをクローンした。次に、2-3 で示したのと同じ方法で VM 上で `mise` を用いた Python 環境を構築し、先に示した方法と同様に FastAPI のインストールを行った。次に、Compute Engine のファイアウォールから http トラフィックをオンにし、VPC ネットワークからファイアウォールルールを作成した。次に、`New-repository` に移動した後

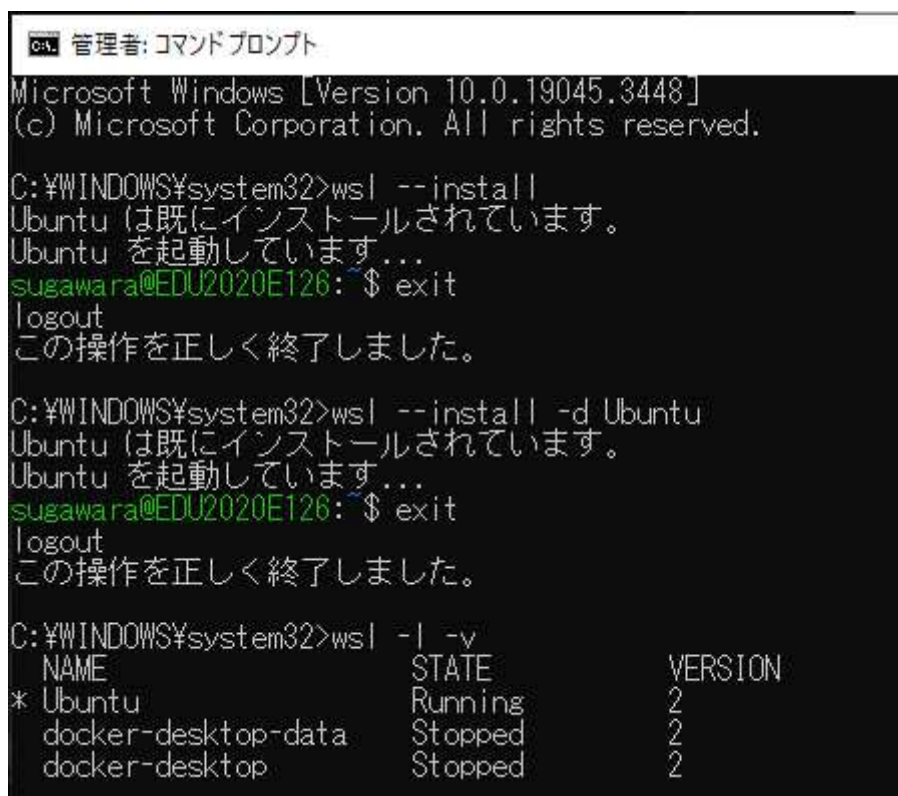
```
uvicorn main:app --host 0.0.0.0 --port 8000
```

の起動オプションで FastAPI サーバを構築した。最後に、ブラウザから `http://35.203.152.47:8000` にアクセスし、サーバにアクセスできているか確認した。

3. 結果

3-1 Linux 環境の導入

図 1 に、各コマンドの実行結果を示す。



```
管理: コマンドプロンプト
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>wsl --install
Ubuntu は既にインストールされています。
Ubuntu を起動しています...
sugawara@EDU2020E126:~$ exit
logout
この操作を正しく終了しました。

C:\WINDOWS\system32>wsl --install -d Ubuntu
Ubuntu は既にインストールされています。
Ubuntu を起動しています...
sugawara@EDU2020E126:~$ exit
logout
この操作を正しく終了しました。

C:\WINDOWS\system32>wsl -l -v
  NAME                                STATE      VERSION
* Ubuntu                              Running    2
  docker-desktop-data                 Stopped    2
  docker-desktop                      Stopped    2
```

図 1 Linux 環境導入のコマンド実行結果

図 1 より、既にインストールされており、Ubuntu が起動した。

3-2 Google Cloud での仮想サーバの作成と SSH 接続

図 2 に、.ssh ディレクトリ作成後のディレクトリ一覧表示結果と、SSH 鍵生成後の.ssh ディレクトリ内の表示結果を示す。

```
● sugawara@EDU2020E126:~$ ls -a
.          .cache      .mise.toml      .vscode-remote-containers
..         .config     .motd_shown     .vscode-server
.aws       .docker     .profile        .zshrc
.azure     .dotnet     .python_history pbl
.bash_history .gitconfig  .ssh
.bash_logout .lessht     .sudo_as_admin_successful
.bashrc     .local      .viminfo
● sugawara@EDU2020E126:~$ cd .ssh
● sugawara@EDU2020E126:~/.ssh$ ls
config  id_rsa  id_rsa.pub  id_rsa_git  id_rsa_git.pub  known_hosts  known_hosts.old
```

図 2 .ssh ディレクトリと SSH 鍵生成後のディレクトリ内容

図 2 より、コマンドによってディレクトリが生成され、公開鍵、秘密鍵が生成された。

図 3 に、公開鍵の内容を表示した結果を示す。

```
● sugawara@EDU2020E126:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDQSNyngjiMwGg6X8aV/jmupHcSBHXx8S5aEiSU1E2plwKka
riQFTl1VJUEmt7lo0SPV91j6367AM1V8KRtc00CouxjvPFxOn8EGeAmiNspaUtr3hHgrND34KmDrSi9j5K1
8H4EILJ2WwU3G4GlimvVfpdzKeYDnoYrRaE2SGnBkSwxT828njGGLUEM4r8aX8LRsf206iXOHVKDU5NArUqgF
9rbTYY646McdMvFscZaYSKA06Swhw5IJHQCNAhw0KvhYooap2bQeVj8VvkRdD1ZkjrirAUgaQj8IeuDzxoJhw
tcUeH95LkL2PVZORJQIXnYouEfVYIXEqUwDno1ZTepm0x6Qb7byiBsKqGg8wvvezUpinE5iWLacj7h5g3YrTi
19YGRxF+vbpmo7biq2I8xrsN8y6/aKBvOFwpfDgps0Ne/TCz+ahQ21qEubCyY/w6dwXK1mtZgpcbe2AXbLhYt
H07z8mvuMDQdffSi8pt/PB67qMA+F52bIy/D0PkH0= sugawara
```

図 3 公開鍵の内容

図 3 より、公開鍵の中身と設定したユーザ名 sugawara が表示された。

図 4 に、SSH 接続結果を示す。

```
○ sugawara@EDU2020E126:~/.ssh$ ssh sugawara@35.203.152.47 -i ~/.ssh/id_rsa
Enter passphrase for key '/home/sugawara/.ssh/id_rsa':
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1058-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Thu Jun 27 21:31:44 UTC 2024

System load:  0.0                       Processes:            102
Usage of /:   14.4% of 28.89GB           Users logged in:      0
Memory usage: 24%                       IPv4 address for ens4: 10.138.0.2
Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

18 updates can be applied immediately.
4 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
Last login: Thu Jun 27 13:16:49 2024 from 115.163.112.178
sugawara@server-1:~$ ls
New-repository
```

図 4 SSH 接続結果

図 4 より、パスフレーズを入力した後に VM に接続された。

3-3 Python ランタイムバージョン管理

図 5 に、mise 導入のコマンド入力結果を示す。

```
sugawara@EDU2020E126:~$ curl https://mise.run | sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 6300 100 6300    0     0 16900      0 --:--:-- --:--:-- --:--:-- 16935
mise: installing mise...
##### 100.0%
mise: installed successfully to /home/sugawara/.local/bin/mise
mise: run the following to activate mise in your shell:
echo "eval \"\${/home/sugawara/.local/bin/mise activate bash}\"" >> ~/.bashrc

mise: this must be run in order to use mise in the terminal
mise: run `mise doctor` to verify this is setup correctly
sugawara@EDU2020E126:~$ ~/.local/bin/mise --version
2024.6.6 linux-x64 (409d6e4 2024-06-20)
sugawara@EDU2020E126:~$ echo 'eval "$( ~/.local/bin/mise activate bash )"' >> ~/.bashrc
```

図 5 mise 導入コマンド入力結果

図 5 より、mise のインストールに成功し、バージョンが表示され、bash への設定が実行された。

図 6 に、パッケージリストの更新結果を示す。

```
sugawara@EDU2020E126:~$ sudo apt update; sudo apt install build-essential libssl-dev
zlib1g-dev \
libbz2-dev libreadline-dev libsqlite3-dev curl git \
libncursesw5-dev xz-utils tk-dev libxml2-dev libxmlsec1-dev libffi-dev liblzma-dev
[sudo] password for sugawara:
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Ign:2 https://download.docker.com/linux/ubuntu \ InRelease
Err:3 https://download.docker.com/linux/ubuntu \ Release
      404 Not Found [IP: 13.32.50.74 443]
Hit:4 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1566 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1775 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [266 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [20
07 kB]
Get:11 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [34
2 kB]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [875
kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [323 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [2067
kB]
Get:15 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [170
kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [352
kB]
Get:17 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1091 k
B]
Get:18 http://archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [253 kB
]
Get:19 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [27.6
kB]
Get:20 http://archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [16.5
kB]
Reading package lists... Done
E: The repository 'https://download.docker.com/linux/ubuntu \ Release' does not have
a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled
by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
build-essential is already the newest version (12.9ubuntu3).
libbz2-dev is already the newest version (1.0.8-5build1).
libffi-dev is already the newest version (3.4.2-4).
liblzma-dev is already the newest version (5.2.5-2ubuntu1).
libreadline-dev is already the newest version (8.1.2-1).
libxmlsec1-dev is already the newest version (1.2.33-1build2).
tk-dev is already the newest version (8.6.11+1build2).
xz-utils is already the newest version (5.2.5-2ubuntu1).
curl is already the newest version (7.81.0-1ubuntu1.16).
git is already the newest version (1:2.34.1-1ubuntu1.11).
libncursesw5-dev is already the newest version (6.3-2ubuntu0.1).
libsqlite3-dev is already the newest version (3.37.2-2ubuntu0.3).
libssl-dev is already the newest version (3.0.2-0ubuntu1.16).
libxml2-dev is already the newest version (2.9.13+dfsg-1ubuntu0.4).
zlib1g-dev is already the newest version (1:1.2.11.dfsg-2ubuntu9.2).
0 upgraded, 0 newly installed, 0 to remove and 47 not upgraded.
```

図 6 パッケージリストの更新結果

図 6 より、パッケージリストの更新に成功した。

図 7 に、mise で Python のバージョン 3.11 をダウンロードし、インタプリタパスを設定した後の main.py 実行結果を示す。

```
• sugawara@EDU2020E126:~$ mise use -q python@3.11
mise ~/.mise.toml tools: python@3.11.9
• sugawara@EDU2020E126:~$ /home/sugawara/.local/share/mise/installs/python/3.11/bin/python /home/sugawara/pbl/main.py
hello world
```

図 7 mise による Python 3.11 インストールと main.py 実行結果

図 7 より、Python の 3.11 がインストールされ、インタプリタパスが設定された main.py が実行されて hello world が表示された。

3-4 Git/GitHub でのコードのバージョン管理

図 8 に、pbl ディレクトリを Git リポジトリとして初期化し、main.py ファイルをステージに追加しコミットしてそのログを表示する各コマンドの実行結果を示す。

```
● sugawara@EDU2020E126:~$ cd pbl
● sugawara@EDU2020E126:~/pbl$ git init
Reinitialized existing Git repository in /home/sugawara/pbl/.git/
● sugawara@EDU2020E126:~/pbl$ git add main.py
● sugawara@EDU2020E126:~/pbl$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   main.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        __pycache__/
        database.py
        docker-compose.yaml
        main1.py
        main2.py
        models.py

● sugawara@EDU2020E126:~/pbl$ git commit -m "first commit"
[main 85b00f6] first commit
 1 file changed, 1 insertion(+), 25 deletions(-)
 rewrite main.py (100%)
● sugawara@EDU2020E126:~/pbl$ git log
commit 85b00f630e6b8d40581027cadb816a4042526e57 (HEAD -> main)
Author: skz0407 <sugakko0507@gmail.com>
Date:   Fri Jun 28 20:27:18 2024 +0900

    first commit
```

図 8 リポジトリの初期化、main.py のステージング、コミット、ログ表示の実行結果

図 8 より、pbl ディレクトリ内の main.py の変更がステージに追加され、first commit としてコミットされた。

図 9 に、main.py の内容を変更した後に再びステージに追加し、コミットしてログを表示する各コマンドの実行結果を示す。

```
● sugawara@EDU2020E126:~/pbl$ git add main.py
● sugawara@EDU2020E126:~/pbl$ git commit -m "second commit"
[main b38e85c] second commit
 1 file changed, 1 insertion(+), 1 deletion(-)
● sugawara@EDU2020E126:~/pbl$ git log
commit b38e85c5e115952574f8137ba9daffc889616a7b (HEAD -> main)
Author: skz0407 <sugakko0507@gmail.com>
Date:   Fri Jun 28 20:28:17 2024 +0900

    second commit

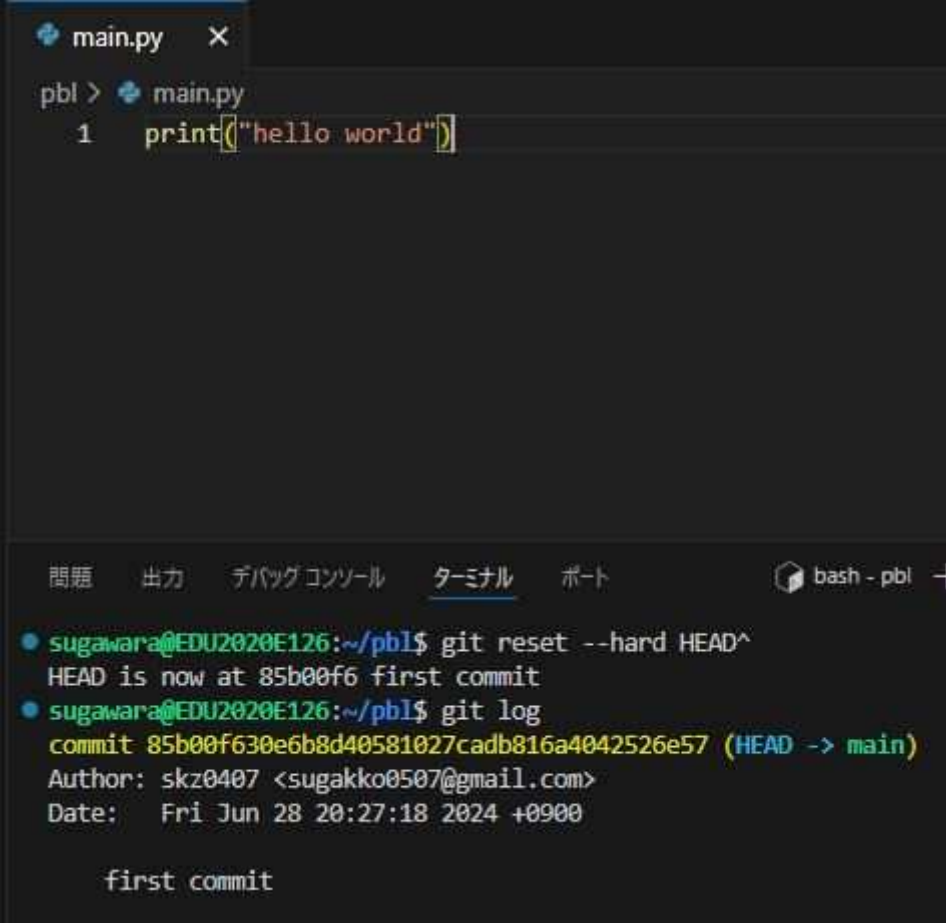
commit 85b00f630e6b8d40581027cadb816a4042526e57
Author: skz0407 <sugakko0507@gmail.com>
Date:   Fri Jun 28 20:27:18 2024 +0900

    first commit
```

図 9 内容を変更した main.py のステージング、コミット、ログ表示の実行結果

図 9 より、main.py の変更がステージに追加され、second commit としてコミットされた。

図 10 に、直前のコミットを取り消し、ログを表示する各コマンドの実行結果と main.py ファイルの内容を示す。



```
main.py x
pbl > main.py
1 print("hello world")

問題 出力 デバッグコンソール ターミナル ポート bash - pbl +
sugawara@EDU2020E126:~/pbl$ git reset --hard HEAD^
HEAD is now at 85b00f6 first commit
sugawara@EDU2020E126:~/pbl$ git log
commit 85b00f630e6b8d40581027cadb816a4042526e57 (HEAD -> main)
Author: skz0407 <sugakko0507@gmail.com>
Date: Fri Jun 28 20:27:18 2024 +0900

first commit
```

図 10 直前のコミット取り消し、ログ表示の実行結果と main.py の内容

図 10 より、直前の second commit が取り消され、main.py の内容が first commit の内容に巻き戻った。

図 11 に、作成した GitHub 用の SSH 鍵の生成結果と公開鍵の内容表示した結果を示す。

```
● sugawara@EDU2020E126:~/.ssh$ ls
config id_rsa id_rsa.pub id_rsa_git id_rsa_git.pub known_hosts known_hosts.old
● sugawara@EDU2020E126:~/.ssh$ cat id_rsa_git.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGD5SnU0CdVBVJe7Pd5rE88qcQIkSqpsEDaQL0H9ZPJTyGHe5
IhHmylIcn2VB+1k0jjx1hfp0dm/dTuwtCl0v98QDFs6w0f8I1QXzVylSqM/TqKHJ5JvOw14n0e7UvgQmEKDu5
jcJP2gcv2Tt/q3jHCKDHJgjQ5IKN7N3OKNe9d+mbpQ6CoTd1ZKqZBIvGGffmkBsShCoDtrRzppXlnSNZxMw/N
MLN0q330xxw8Q0Dbfqlw1Lgkf70xsomGEdlX00osx0ZYyuu0LiKdLYjem/O+HJd/SrUDf7949tt375xnuffi3n
v6UNDiDqorIe3KaIM2oGvBLQ6qdxTG2HRMfkOHb9uzdRZpIOkwowJsSjkGRVcqCI4kQMSHljTYD4zMaCKSPnr
i6HQ+om0p0zFARr7zJKDgb7RULAPjRD1wg9TvDoLgQkLQJx4rb9NR0R4Dku0YyFQz9PtXA0A8Ivnpdn5C8Ssr
EZjQtSmua1c9pfhsJbZ480oI9AYoQByRJDJpkVklk= sugawara@EDU2020E126
```

図 11 SSH 鍵の生成結果と公開鍵の内容

図 11 より、SSH 公開鍵の id_rsa_git.pub と秘密鍵 id_rsa_git が生成され、公開鍵の内容が表示された。

図 12 に、GitHub で SSH 公開鍵を設定し、新しいリポジトリを作成した後、ローカルリポジトリにリモートリポジトリの URL を追加し、デフォルトのブランチ名を main に変更し、コミットをリモートリポジトリに送信するコマンドの実行結果を示す。

```
● sugawara@EDU2020E126:~/pbl$ git remote add origin https://github.com/skz0407/New-repository.git
error: remote origin already exists.
● sugawara@EDU2020E126:~/pbl$ git branch -M main
● sugawara@EDU2020E126:~/pbl$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (1/1), done.
Writing objects: 100% (3/3), 259 bytes | 259.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/skz0407/New-repository.git
def194e..85b00f6 main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

図 12 リモートリポジトリへのコミット送信結果

図 12 より、リモートリポジトリの URL の追加は既に存在するとエラーがでた。また、コミットがリモートリポジトリに送信された。

図 13 に、GitHub で作成したリポジトリの内容を示す。

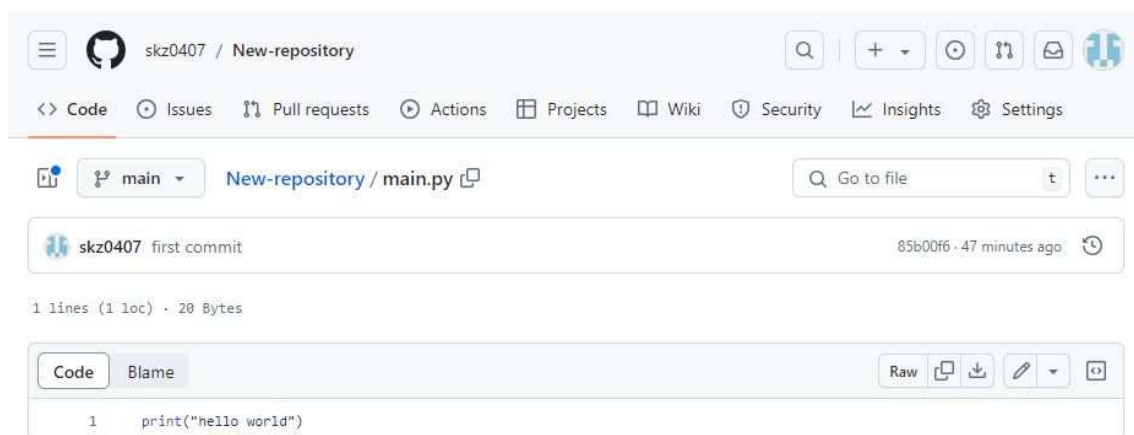


図 13 GitHub のリポジトリの内容

図 13 より、first commit として main.py ファイルが追加されていた。

3-5 FastAPI による Web サーバの構築

図 14 に、FastAPI をインストールし、main.py の内容を変更した後に、FastAPI サーバを起動した結果を示す。

```
○ sugawara@EDU2020E126:~/pbl$ fastapi dev main.py
INFO: Using path main.py
INFO: Resolved absolute path /home/sugawara/pbl/main.py
INFO: Searching for package file structure from
      directories with __init__.py files
INFO: Importing from /home/sugawara/pbl

Python module file
┌───┴───
main.py

INFO: Importing module main
INFO: Found importable FastAPI app

Importable FastAPI app
┌───┴───
from main import app

INFO: Using import string main:app

FastAPI CLI - Development mode

Serving at: http://127.0.0.1:8000

API docs: http://127.0.0.1:8000/docs

Running in development mode, for production use:

fastapi run

INFO: Will watch for changes in these directories: ['/home/sugawara/pbl']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [32734] using WatchFiles
INFO: Started server process [32740]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:45208 - "GET / HTTP/1.1" 200 OK
```

図 14 FastAPI サーバ構築結果

図 14 より、サーバが 127.0.0.1 で実行され、ポート 8000 番でリクエストを受けており、GET リクエストに対するレスポンスが成功していた。

図 15 に、ブラウザで `http://127.0.0.1:8000` にアクセスした結果を示す。



図 15 ブラウザで `http://127.0.0.1:8000` へのアクセス結果

図 15 より、`"Hello": "World"`が表示された。

図 16 に、VM に SSH 接続し、`mise` と `FastAPI` のインストールを同様の方法で行い、ファイアウォールの設定を完了した後で、GitHub からクローンしたリポジトリに移動しサーバを起動した結果を示す。

```
sugawara@server-1:~$ git clone https://github.com/skz0407/New-repository
fatal: destination path 'New-repository' already exists and is not an empty d
irectory.
sugawara@server-1:~$ cd New-repository
sugawara@server-1:~/New-repository$ uvicorn main:app --host 0.0.0.0 --port 80
00
INFO:      Started server process [547178]
INFO:      Waiting for application startup.
INFO:      Application startup complete.
INFO:      Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:      115.163.112.178:20101 - "GET / HTTP/1.1" 200 OK
```

図 16 GitHub からのリポジトリのクローンとサーバ起動結果

図 16 より、既にリポジトリが存在するエラーが発生した。また、外部からの GET リクエストに対するレスポンスが成功していた。

図 17 に、ブラウザで `http://35.203.152.47:8000` にアクセスした結果を示す。



図 17 ブラウザで `http://35.203.152.47:8000` へのアクセス結果

図 17 より、`"Hello": "World"`が表示された。

4. 考察

4-1 Linux 環境の導入

WSL は Windows 上で Linux 環境を実行するため機能である。ここでは既に WSL をインストールして Ubuntu をディストリビューションとして設定して動作させていたため、各コマンドで Ubuntu が起動したと考えられる。

4-2 Google Cloud での仮想サーバの作成と SSH 接続

作成したディレクトリは「.」で始まる隠しディレクトリであるため、ls では表示されなかった。そのため ls -a で表示させた。

SSH 接続はクライアントが秘密鍵を用いて証明書を作成し、サーバの公開鍵を用いて証明書を検証して通信を行う。そのため、作成した公開鍵の内容を VM に登録する必要があり、接続する際は接続先サーバの IP アドレスと秘密鍵のパスが必要だったと考えられる。

クラウドサービスは SaaS (Software as a Service)、PaaS (Platform as a Service)、IaaS (Infrastructure as a Service) の 3 つに分けられる。SaaS はソフトウェアをインターネット経由でユーザに提供するサービスで、PaaS はアプリケーション開発やテスト、デプロイ、管理に必要なプラットフォームをユーザに提供するサービスで、IaaS は仮想マシンやストレージなどの基盤となるインフラをユーザに提供するサービスである。Google Compute Engine は仮想マシンを提供するサービスであるため、IaaS に分類されると考えられる。

4-3 Python ランタイムバージョン管理

mise をインストールした後で以下のコマンドを実行した。

```
echo 'eval "$($HOME/.local/bin/mise activate bash)'" >> ~/.bashrc
```

これは、mise を Bash シェル用に活性化させるコマンドを .bashrc ファイルの末尾に追加するコマンドである。これにより、新しくターミナルを起動すると自動的に mise が利用可能になったと考えられる。

Python は開発が活発な言語であり、頻繁に新しいバージョンがリリースされている。そのため、様々なプロジェクトを適切なバージョンで管理することが求められると考えた。今回導入した mise は、いろいろなバージョンの Python をインストールでき、インタプリタパスを設定することで簡単にバージョンが管理できることから、開発に重要な役割を果たしていると評価できる。

4-4 Git/GitHub でのコードのバージョン管理

Git によるコードのバージョン管理は、最初に `git init` によって現在のディレクトリに `.git` ディレクトリを作成し、Git の管理情報を格納している。次に、`git add` ファイルによってファイルの現在の状態をステージングエリアに追加している。次に、`git status` によってディレクトリのトラッキングされているファイルとされていないファイル、ファイルの変更がステージングエリアにあるかを確認できる。次に、`git commit -m “メッセージ”` によってステージングエリアの内容を新しいコミットとして Git のリポジトリにコミットされる。次に、`git log` によってリポジトリのコミット履歴を確認できる。次に、`git reset --hard HEAD^` によって HEAD つまり現在のコミットを取り除き、ステージングエリアやディレクトリの内容をリセットできる。次に、`git remote add origin URL` によって指定した URL のリモートリポジトリを `origin` という名前で追加し、GitHub のリモートリポジトリと通信が可能になる。次に、`git branch -M main` によって現在のブランチの名前を `main` に変更できる。最後に、`git push -u origin main` によって、ローカルリポジトリの `main` ブランチをリモートリポジトリの `main` ブランチにプッシュしている。また、`-u` オプションにより、デフォルトのプッシュ先が設定されている。以上の各コマンドの動作により、`main.py` のコードの変更が管理され、GitHub に送信されたと考えられる。

4-5 FastAPI による Web サーバの構築

FastAPI サーバが実行された IP アドレス `127.0.0.1` は自分自身のコンピュータを指す `localhost` であり、サーバはローカルマシン上で実行されている。そのため、自分自身しかこのサーバにはアクセスできないと考えられる。`main.py` ではルートデコレータによって、指定したパスへのリクエストに対して処理を返す関数を定義している。ここでは、`http://127.0.0.1:8000` へ GET リクエストを送信したため、`read_root()` 関数が呼び出され、“Hello”：“World”がレスポンスとして表示されたと考えられる。

VM 上で FastAPI サーバを実行し、外部からアクセスできるようにするためには、ファイアウォールの設定を行う必要があった。ここでは、任意のアドレス `0.0.0.0/0` から `8000` 番ポートへのアクセスを許可するように設定し、サーバを起動するコマンドもそのようにした。これにより、VM 上で動作するサーバへ誰でもアクセスできるようになったと考えられる。また同様に、`http://35.203.152.47:8000` への GET リクエストに対して“Hello”：“World”がレスポンスとして表示されたと考えられる。

5. 結論

Linux 環境を自身のパソコンで構築でき、Google Cloud で仮想サーバを作成して SSH 接続する方法を理解した。Python のランタイムバージョン管理方法として mise を導入した。Git と GitHub を用いたコードのバージョン管理するための各コマンドの動作を理解した。また、FastAPI を利用してローカルと仮想マシン上で Web サーバを構築し、実際にアクセスして動作を理解することができた。