

Lyonのトリセツ

コンピュータサイエンス学部

青木・佐々木研究室

北條海斗

目次

- Lyonの仕様
- WSLのインストール
- Docker構築方法
- Colabと同等環境を動かす方法
- Jupyterの実装方法

※本ドキュメントの画像・コマンドはソフトウェアの更新により
利用不能・変更となる場合があります。ご了承ください。

分からなければ手を上げて質問してください

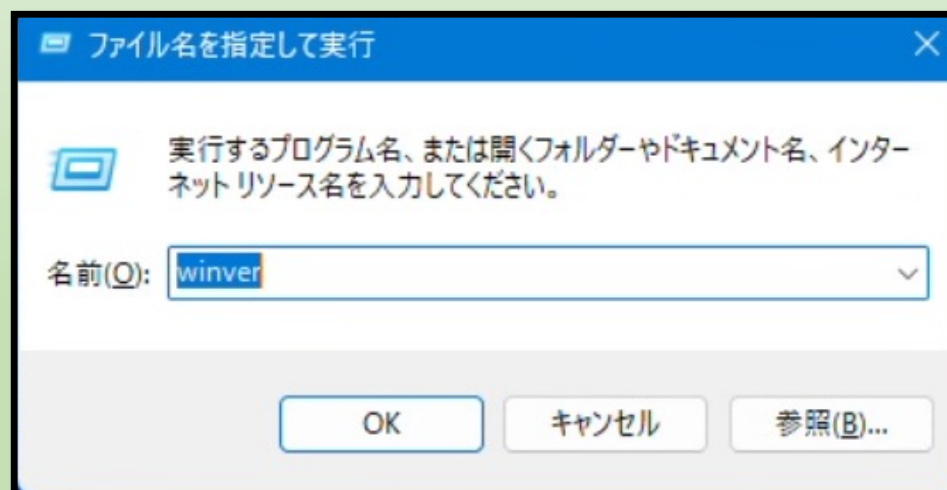
Lyonの仕様

- それぞれ専用のユーザーが作成されている
 - 各ユーザは自分のユーザフォルダ内でDockerを使ってシステムの構築を行う
- ファイルの読み書きについて
 - ほかユーザのファイルは見ることはできるが書き込みができない。（注意）
- VPNで使用可能
- 管理者権限は有していない
 - 管理担当アカウントに管理者権限は帰属。
 - 個別でほしいライブラリがあれば，管理者まで（現状青木先生）。

WSLのインストール

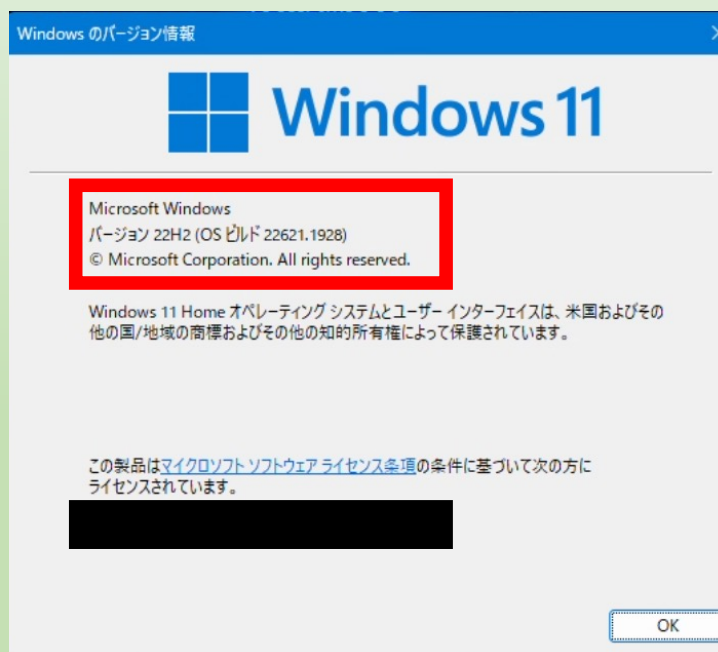
WindowsPCのバージョンを確認する。

- キーボードのWindowsキー+Rを押す
- ファイル名を検索して実行の部分に「winver」と入力
- Enterキーを押下



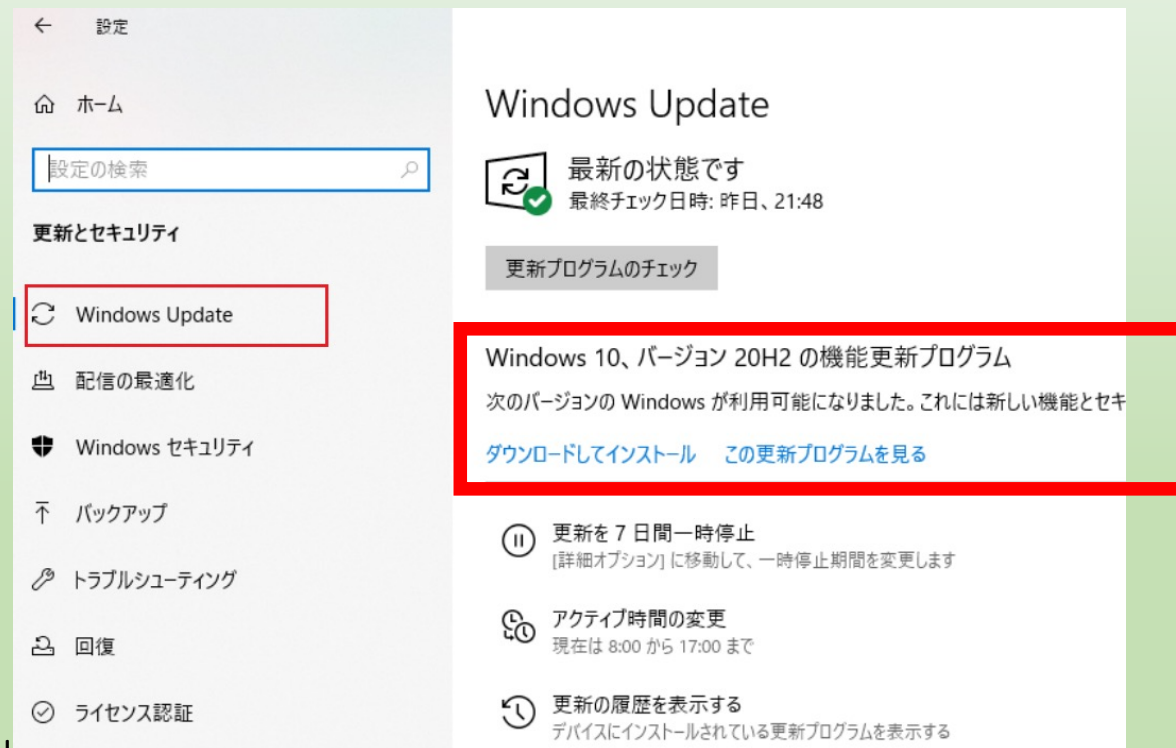
Windowsのバージョン情報を参照

- WindowsバージョンがWindows11もしくは
- Windows10-バージョン2004以上であることを確認



Windowsのバージョンが低い場合

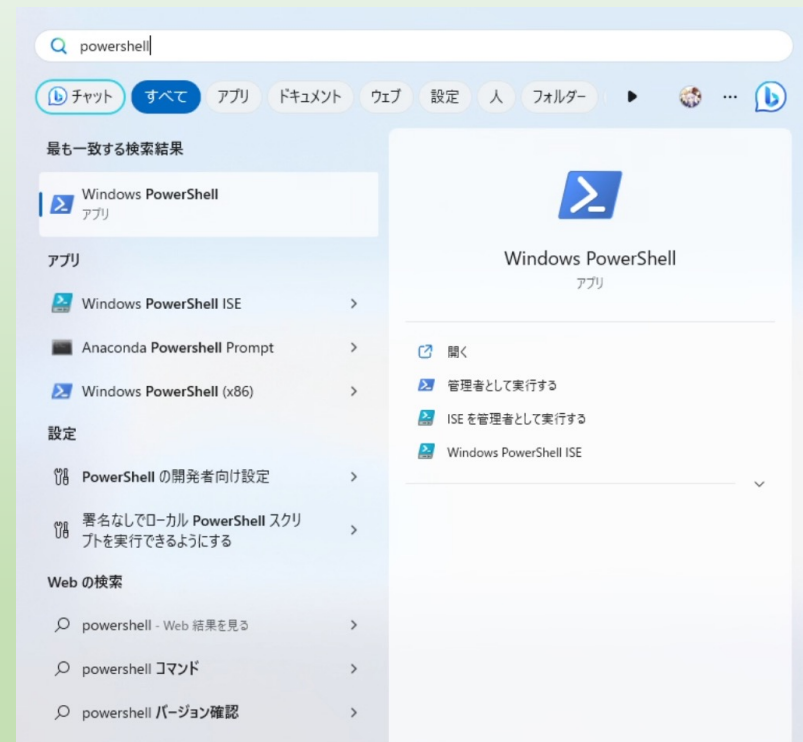
- ・ 設定 > windowsUpdate から表示されているバージョンをダウンロードしてインストール



Keyword : commandprompt, shell, zsh, bash, sh

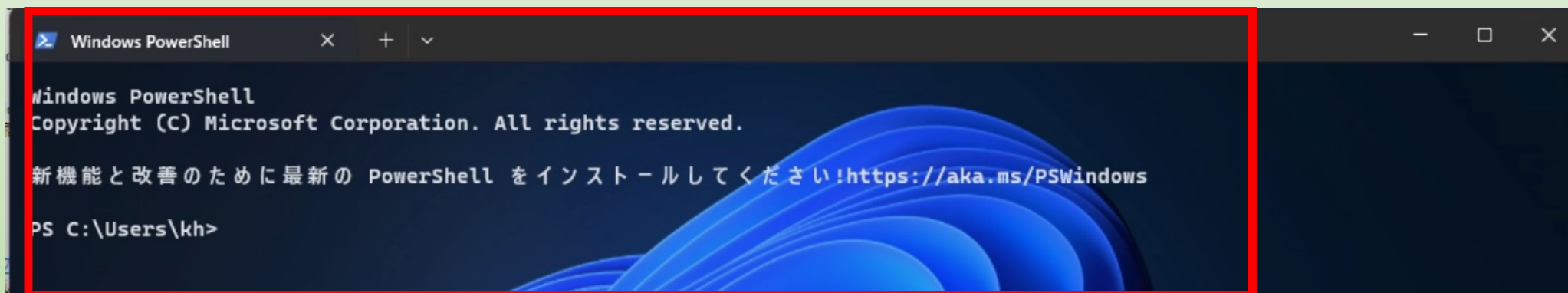
PowerShellを開く

キーボードのWindowsキーを押す
検索画面にpowershellと入力
一番上のやつを開く。



PowerShellを開く.

- WindowsPowerShellが起動するのを確認
- 以下のテキストが表示されることを確認

A screenshot of a Windows PowerShell terminal window. The window has a title bar with the text 'Windows PowerShell' and standard window controls (minimize, maximize, close). The terminal content is as follows:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

新機能と改善のために最新の PowerShell をインストールしてください!https://aka.ms/PSWindows

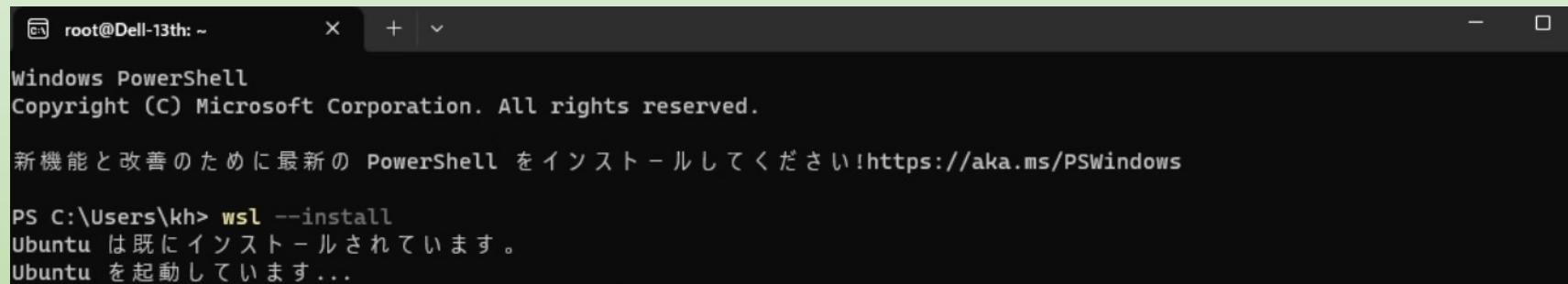
PS C:\Users\kh>
```

The text is displayed in a light blue font on a dark blue background. A red rectangular border highlights the terminal content area.

WSLをインストール

※すでにインストール済みの人はP12以降を参照する

- コマンド「`wsl --install`」と入力
- 画面に従ってPCを再起動する。



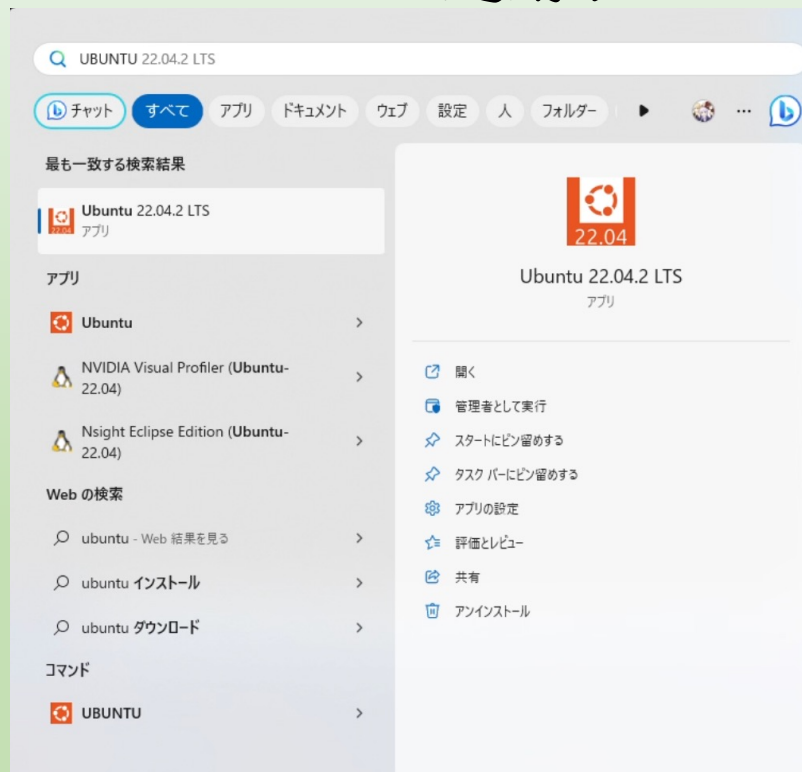
```
root@Dell-13th: ~
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

新機能と改善のために最新の PowerShell をインストールしてください!https://aka.ms/PSWindows

PS C:\Users\kh> 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...
```

WSLの起動

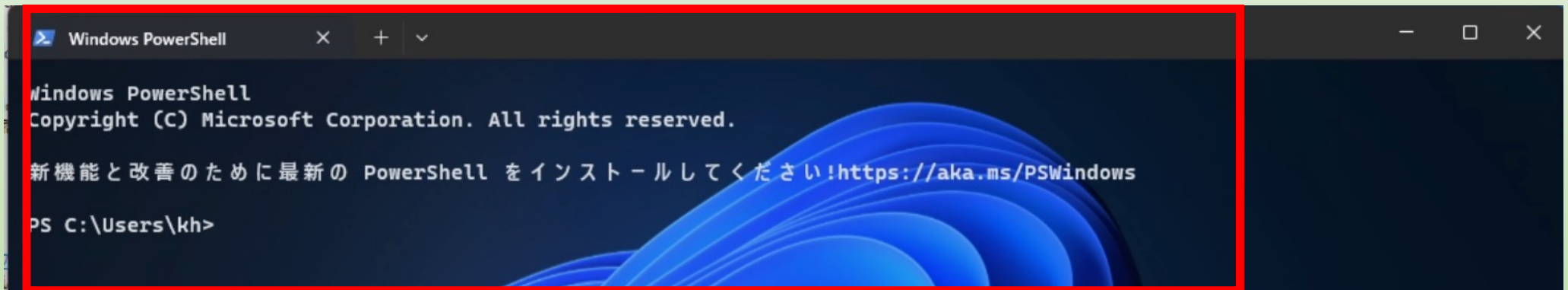
・インストールしたUbuntuを起動する



すでにWSLがインストール済みの場合

PowerShellを開く.

- WindowsPowerShellが起動するのを確認
- 以下のテキストが表示されることを確認

A screenshot of a Windows PowerShell terminal window. The window has a title bar that says "Windows PowerShell" with standard window controls (minimize, maximize, close). The terminal content is as follows:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

新機能と改善のために最新の PowerShell をインストールしてください!https://aka.ms/PSWindows

PS C:\Users\kh>
```

The text is displayed in a light blue font on a dark blue background. A red rectangular border highlights the terminal content area.

WSLのバージョンを確認する

- PowerShellに「wsl -l -v」と入力し実行 (Enter)
- バージョンの欄に注目する

```
PS C:\Users\kh> wsl -l -v
```

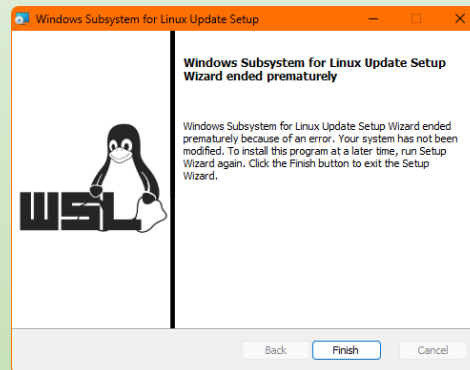
NAME	STATE	VERSION
* Ubuntu-22.04	Running	2

```
PS C:\Users\kh> |
```

- バージョンが2未満であればバージョンの更新を行う
- その際、変更したいディストリビューションのNameを覚えておく (緑枠)

カーネルの更新

- WSLのカーネル更新プログラム
(https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi)をダウンロードする



- セットアップウィザードに従ってインストール(Nextを押してください)

Windowsの設定変更

- PowerShellに戻り、以下のコマンドを入力→実行
- 「dism.exe /online /enable-feature
/featurename:VirtualMachinePlatform /all /norestart」

```
PS C:\> dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

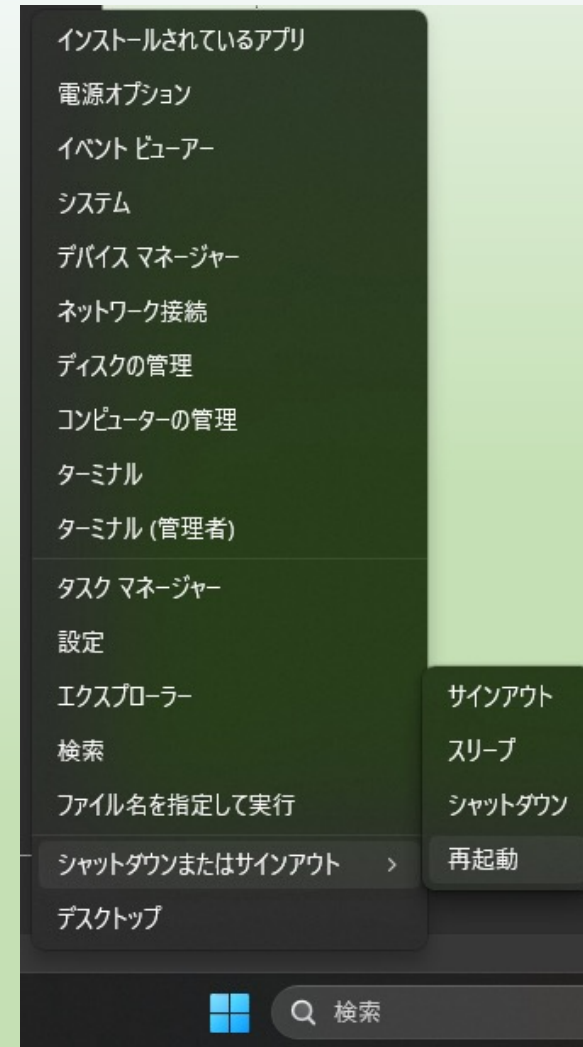
```
展開イメージのサービスと管理ツール  
バージョン: 10.0.19041.1
```

```
イメージのバージョン: 10.0.19041.264
```

```
機能を有効にしています  
[=====100.0%=====]  
操作は正常に完了しました。
```

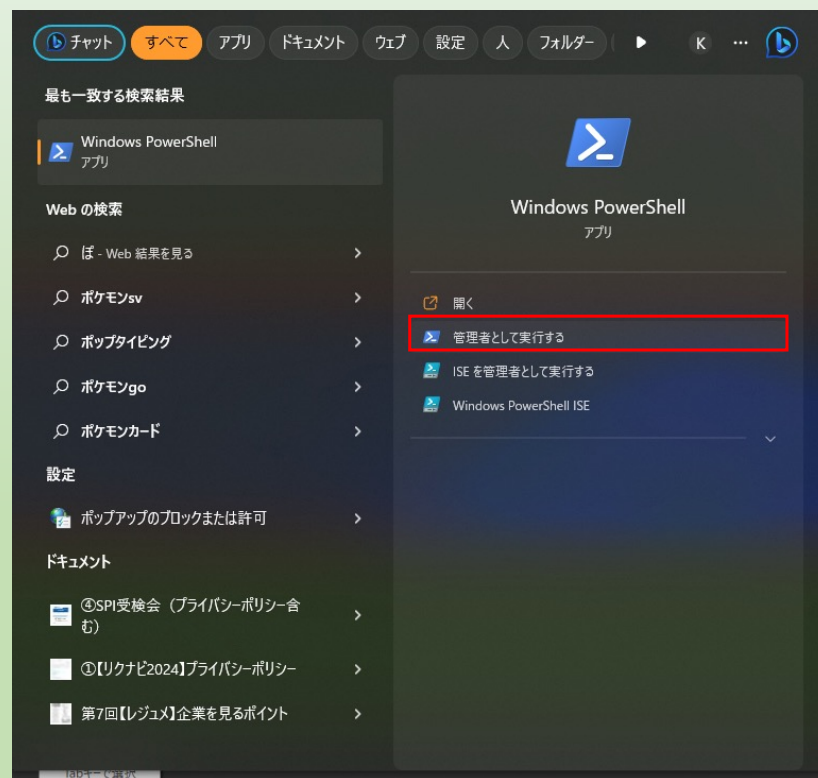

マシンを再起動する

- 手段は問わない。
- 普段の方法で再起動



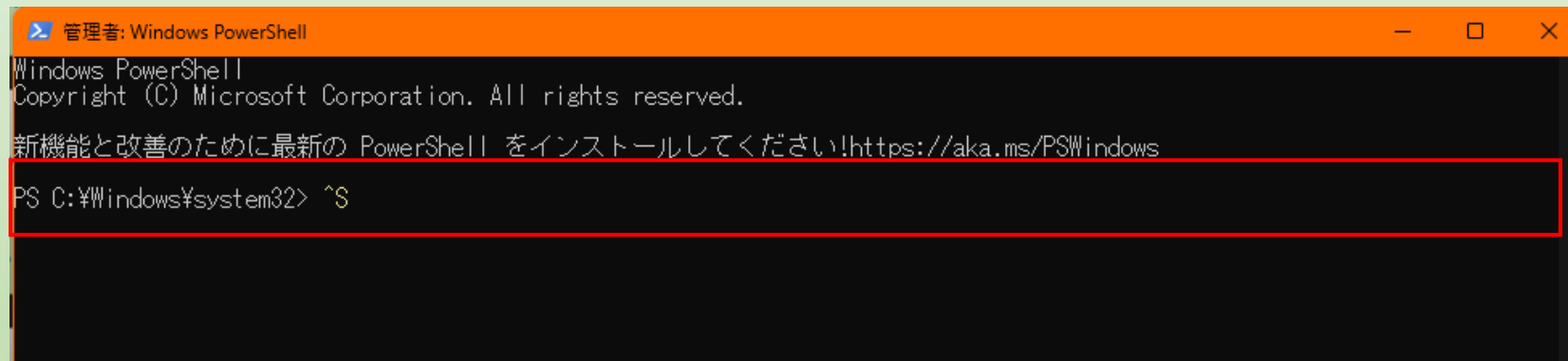
PowerShellを管理者権限で開く。

- 管理者として実行をするをクリックする。



PowerShellを開く.

- WindowsPowerShellが起動するのを確認
- 上記の画面が表示されることを確認。

A screenshot of a Windows PowerShell console window. The title bar is orange and reads "管理者: Windows PowerShell". The window content is black with white text. It displays the PowerShell logo, the text "Windows PowerShell", "Copyright (C) Microsoft Corporation. All rights reserved.", and a message in Japanese: "新機能と改善のために最新の PowerShell をインストールしてください!https://aka.ms/PSWindows". Below this, the prompt "PS C:\Windows\system32>" is followed by the command "^S". A red rectangular box highlights the command input area.

```
管理者: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

新機能と改善のために最新の PowerShell をインストールしてください!https://aka.ms/PSWindows

PS C:\Windows\system32> ^S
```

WSLのデフォルトバージョンを変更

- PowerShellに「`wsl --set-default-version 2`」と入力

```
PS C:\Users\kh> wsl --set-default-version 2
WSL 2 との主な違いについては、https://aka.ms/wsl2
を参照してください
この操作を正しく終了しました。
PS C:\Users\kh> |
```

- この操作を正しく終了しましたと表示されればOK

WSLの更新

- P14で読みだしたディストリビューションの名前を利用

- PowerShellに

`wsl --set-version ディストリビューション名 2`を入力

今回の場合、「`wsl --set-version ubuntu22.04 2`」となる

※ディストリビューション名は、利用者によって異なるのでP14をよく確認する。

```
PS C:\> wsl --set-version ubuntu 2
変換中です。この処理には数分かかることがあります...
WSL 2 との主な違いについては、https://aka.ms/wsl2 を参照してください
変換が完了しました。
```

変換が完了しましたと表示されればOK

仮想マシン機能が有効できなかった場合

- **WSLの更新はせずに**，インストールされているものを利用して
ください

Windows の仮想マシン プラットフォーム機能を有効にして、BIOS で仮想化が有効になっていることを確認してください。 詳細については、<https://aka.ms/wsl2-install> を参照してください

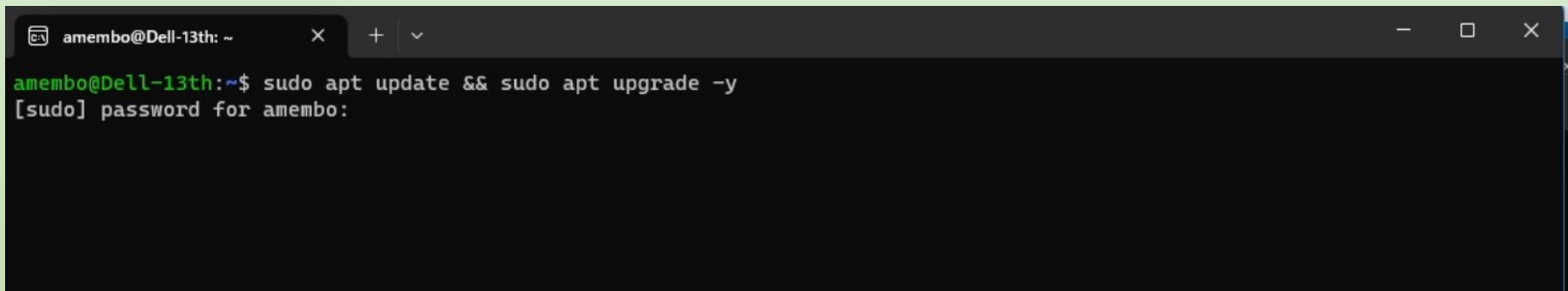
Ubuntuを起動

- 初期セットアップが終わるのを待つ
- 画面に従いユーザー名とパスワードを決める
(自分の好きなやつでいいが必ず覚えるかメモる.)

```
ohtsu@Tiger64: ~  
Installing, this may take a few minutes...  
Please create a default UNIX user account. The username does not need to match you  
For more information visit: https://aka.ms/wslusers  
Enter new UNIX username: ohtsu  
Enter new UNIX password:  
Retype new UNIX password:  
passwd: password updated successfully  
Installation successful!  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.
```

Ubuntuのライブラリを更新

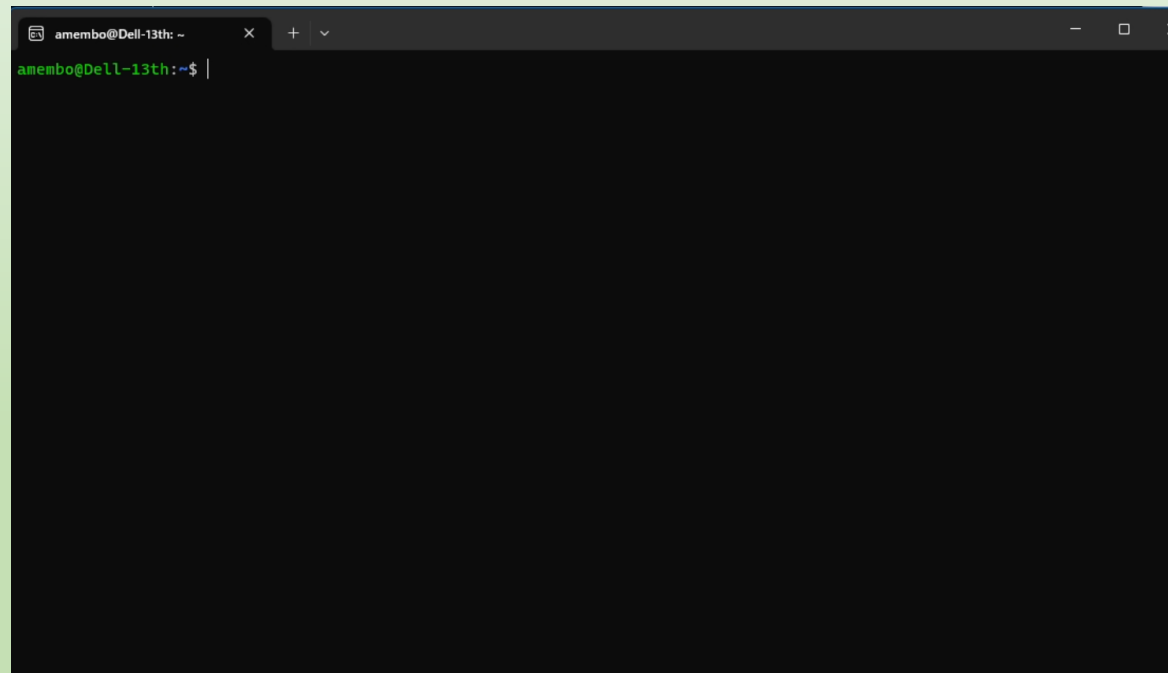
- 先程起動したUbuntuに以下のコマンドを入力
「sudo apt update && sudo apt upgrade -y」
- パスワードを聞かれるので先程決めたパスワードを入力



```
amembo@Dell-13th: ~  
amembo@Dell-13th:~$ sudo apt update && sudo apt upgrade -y  
[sudo] password for amembo:
```


Ubuntuのセットアップ完了

- ディスプレイにシェルの「\$」マークが出てれば完了

A screenshot of a terminal window with a dark background. The window title bar shows 'amembo@Dell-13th: ~' and standard window controls. The terminal content shows the prompt 'amembo@Dell-13th:~\$' in green text, followed by a vertical cursor bar. The rest of the terminal is empty.

```
amembo@Dell-13th: ~  
amembo@Dell-13th:~$ |
```

Lyonへの接続（共用VM）

Lyonのホスト名を探す。

- Lyon公式ページから, 有効なホスト名を探す (なんでもいい)

<http://lyonreport.cloud.cs.priv.teu.ac.jp/report.php>

→今回はlyon001.cloud.cs.priv.teu.ac.jpをホスト名とする。

Lyon VM(共用)の情報

約5分おきにリソース情報を取得しています。ホスト名でsshでログインしてください。

操作方法やコマンドは[こちら](#)を参照してください。

ホスト名	vCPU使用率	RAM使用率	ディスク使用率	GPU使用率	GPU RAM使用率
lyon001.cloud.cs.priv.teu.ac.jp	vCPU:1.0%/4vCPUs	RAM:16.2%/16GB	Disk:2.6%/16233GB	GPU:0%	GPU RAM:0.4%/8GB

SSHのコマンドを決める.

共用VMの場合

ネットワーク利用ID : 学籍番号 + 任意の文字列 2 文字

例 :) c0b22999 → c0b22999ff (事前に大学から交付済み)

ssh ネットワーク利用ID@ホスト名 (P26参照) でログインする

例 :) 「ssh c0b22999ff@lyon001.cloud.cs.priv.teu.ac.jp」

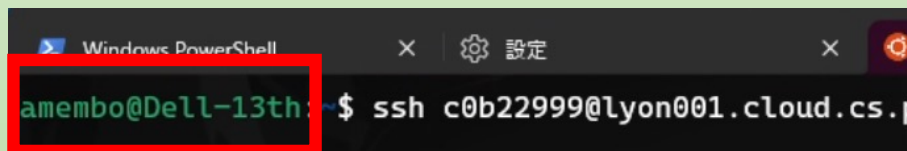
先程決めたコマンドを入力

- Ubuntuのシェルに先程決めたコマンドを入力する。

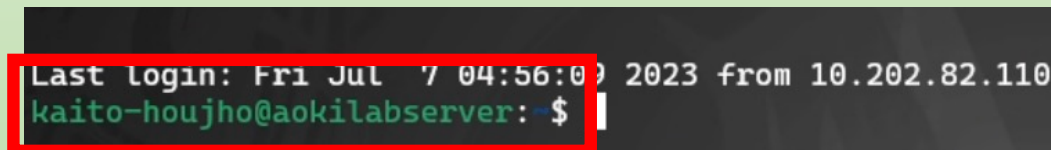
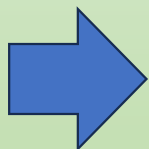
A screenshot of a Windows PowerShell terminal window. The title bar shows three tabs: 'Windows PowerShell', '設定' (Settings), and 'amembo@Dell-13th: ~'. The terminal content shows the prompt 'amembo@Dell-13th:~\$' followed by the command 'ssh c0b22999@lyon001.cloud.cs.priv.teu.ac.jp' with a cursor at the end. The background of the terminal has a dark theme with a faint anime-style illustration of a character with orange hair.

ログインする.

- Enterを押してログインする.
- 緑色の文字が入力前と変わっていればOK



```
Windows PowerShell
amembo@Dell-13th: ~$ ssh c0b22999@lyon001.cloud.cs.p
```



```
Last login: Fri Jul  7 04:56:09 2023 from 10.202.82.110
kaito-houjho@aokilabserver: ~$
```

Lyonへの接続（専用VM）

SSHのコマンドを決める.

Kaggle専用VMの場合

ssh 学籍番号@10.204.227.44

この場合ホスト名は10.204.227.44となる。

例:) 「ssh c0b23999@10.204.227.44」

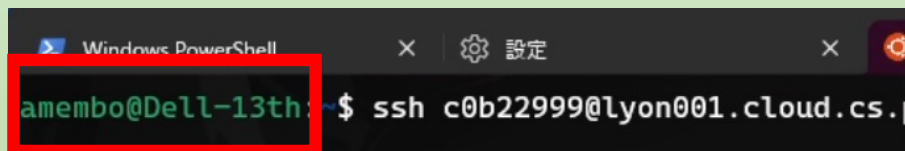
先程決めたコマンドを入力

- Ubuntuのシェルに先程決めたコマンドを入力する.

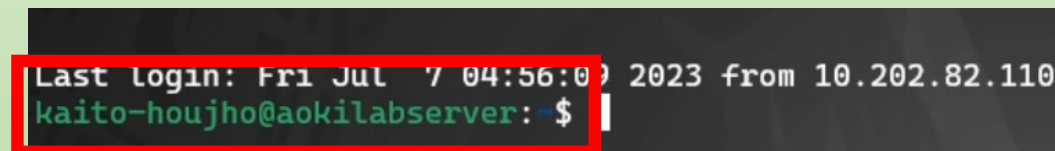
```
This message is shown once a day. To disable it please create the  
/home/amembo/.hushlogin file.  
amembo@RyNvidia:~$ ssh kaito-houjho@10.204.227.44
```

ログインする.

- Enterを押してログインする.
- 緑色の文字が入力前と変わっていればOK



A screenshot of a Windows PowerShell terminal window. The title bar shows 'Windows PowerShell' and a '設定' (Settings) icon. The prompt is 'amembo@Dell-13th:~\$'. The command entered is 'ssh c0b22999@lyon001.cloud.cs.p'. A red box highlights the prompt and the first part of the command.



A screenshot of a terminal window showing the output of the SSH command. The first line is 'Last login: Fri Jul 7 04:56:09 2023 from 10.202.82.110'. The second line is 'kaito-houjho@aokilabserver:~\$'. A red box highlights the second line.

【必須】パスワードの変更する¹

はじめにログインするときは必ず変更を行う

ログインしているユーザーのパスワードを変更する

「passwd」と入力し、Enterを押す

→Current password: と表示されたらログイン時のパスワードを入力

```
kaito-houjho@ex002:~$ passwd
kaito-houjho 用にパスワードを変更中
Current password: █
```

【必須】パスワードの変更する²

→New password: と表示されたら変更後のパスワードを入力

→Retype new password: と表示されたら再度変更後のパスワードを入力

passwd: password updated successfully と表示され、再度 \$ マークが表示されたら完了

```
amembo@RyNvidia:~$ passwd
Changing password for amembo.
Current password:
New password:
Retype new password:
passwd: password updated successfully
amembo@RyNvidia:~$ AA
```

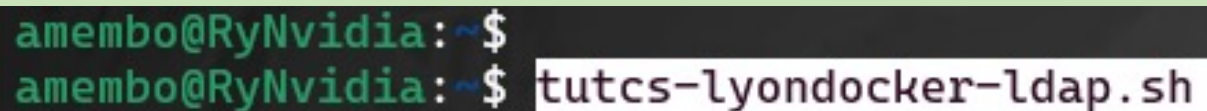
Dockerのインストール | 共用VM

Dockerのインストール¹

Dockerをインストールする。 | インストールから初期設定まで
やってもらえる

`$tutcs-lyondocker-ldap.sh`

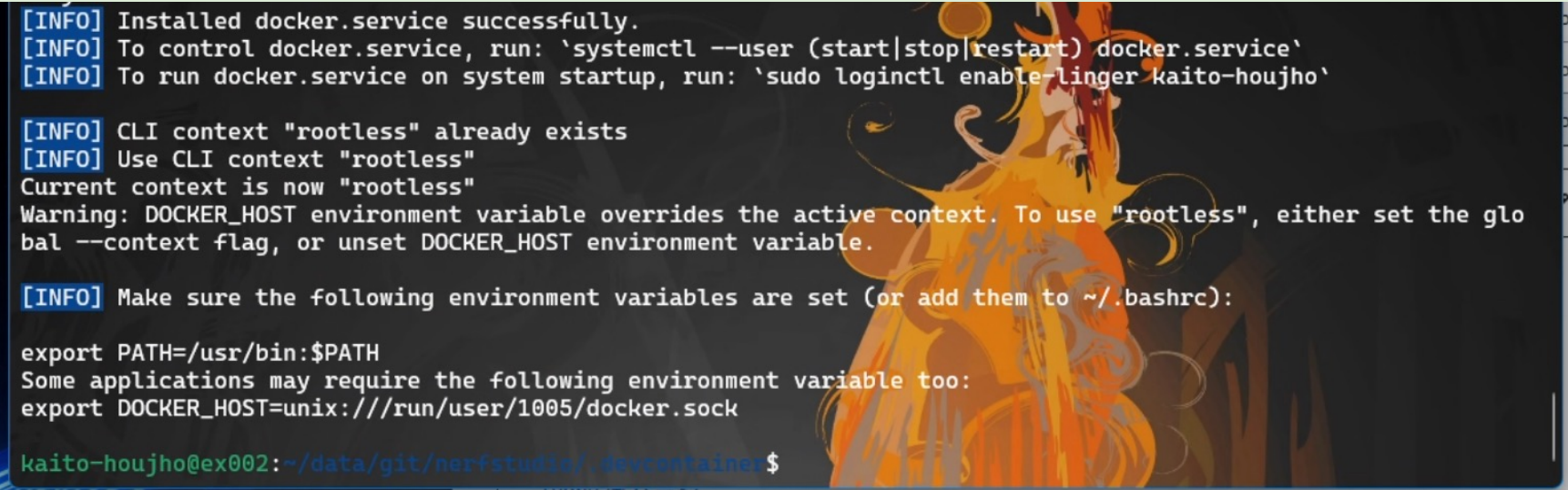
条件にあったコマンドをそのまま入力

A terminal window with a dark background. The prompt 'amembo@RyNvidia:~\$' is shown in green. The command 'tutcs-lyondocker-ldap.sh' is entered and highlighted with a white selection box.

```
amembo@RyNvidia:~$  
amembo@RyNvidia:~$ tutcs-lyondocker-ldap.sh
```

Dockerのインストール²

シェルの\$マークが出てきたら環境構築完了



```
[INFO] Installed docker.service successfully.
[INFO] To control docker.service, run: `systemctl --user (start|stop|restart) docker.service`
[INFO] To run docker.service on system startup, run: `sudo loginctl enable-linger kaito-houjho`

[INFO] CLI context "rootless" already exists
[INFO] Use CLI context "rootless"
Current context is now "rootless"
Warning: DOCKER_HOST environment variable overrides the active context. To use "rootless", either set the global --context flag, or unset DOCKER_HOST environment variable.

[INFO] Make sure the following environment variables are set (or add them to ~/.bashrc):

export PATH=/usr/bin:$PATH
Some applications may require the following environment variable too:
export DOCKER_HOST=unix:///run/user/1005/docker.sock

kaito-houjho@ex002:~/data/git/nerfstudio/.devcontainer$
```

Dockerのインストール | 専有VM

Dockerのインストール

Dockerをインストールする。 | インストールから初期設定まで
やってもらえる

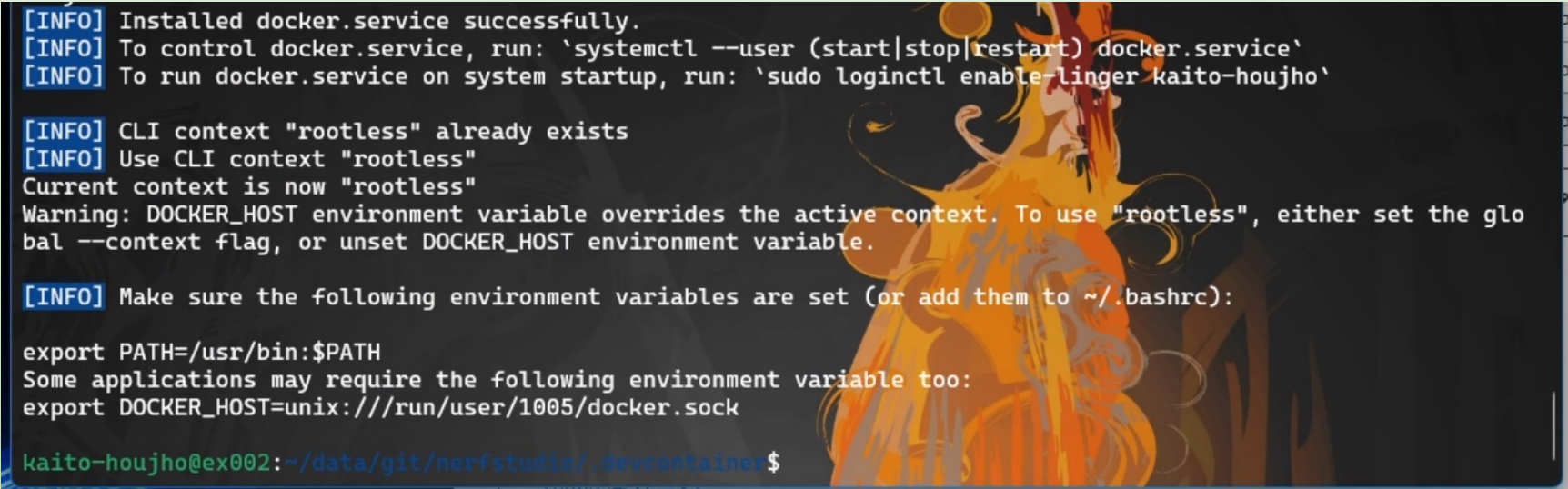
\$ tutcs-lyondocker-normal.sh

条件にあったコマンドをそのまま入力

```
kaito-houjho@ex002:~/data/git/nerfstudio/.devcontainer$ tutcs-lyondocker-normal.sh
```

Dockerのインストール²

シェルの\$マークが出てきたら環境構築完了



```
[INFO] Installed docker.service successfully.
[INFO] To control docker.service, run: `systemctl --user (start|stop|restart) docker.service`
[INFO] To run docker.service on system startup, run: `sudo loginctl enable-linger kaito-houjho`

[INFO] CLI context "rootless" already exists
[INFO] Use CLI context "rootless"
Current context is now "rootless"
Warning: DOCKER_HOST environment variable overrides the active context. To use "rootless", either set the global --context flag, or unset DOCKER_HOST environment variable.

[INFO] Make sure the following environment variables are set (or add them to ~/.bashrc):

export PATH=/usr/bin:$PATH
Some applications may require the following environment variable too:
export DOCKER_HOST=unix:///run/user/1005/docker.sock

kaito-houjho@ex002:~/data/git/nerfstudio/.devcontainer$
```

Dockerを起動・停止

Dokcerの起動

コマンドをそのまま入力
「dockerstart.sh」

```
kaito-houjho@ex002:~/data/git/nerfstudio/.devcontainer$ dockerstart.sh
```

Dokcerの停止

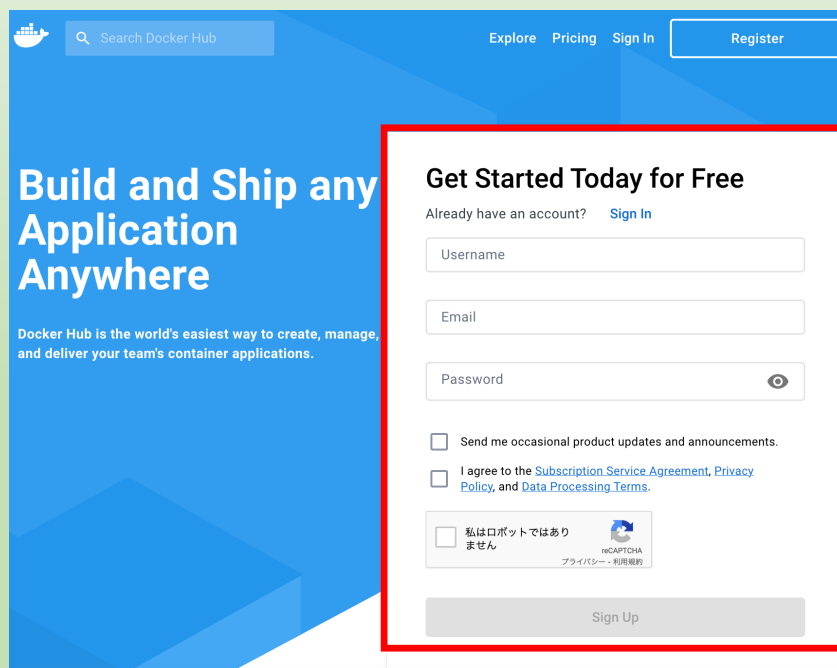
コマンドをそのまま入力
「dockerstop.sh」

```
kaito-houjho@ex002:~/data/git/nerfstudio/.devcontainer$ dockerstop.sh
```

VMを使い終わったらこれを実行

DockerHubのアカウント作成

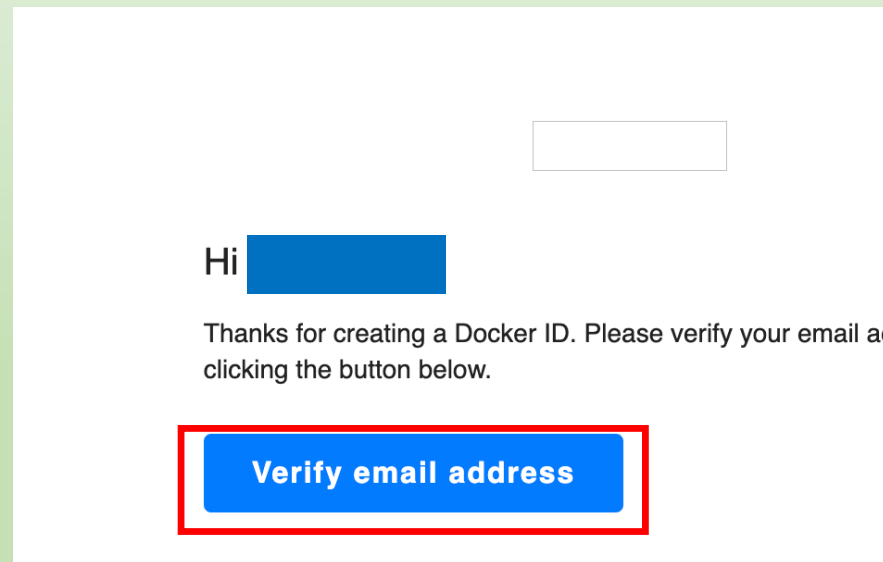
DockerHubのアカウントを (<https://hub.docker.com/>) 作成。
ボックスに必要情報を入力しサインアップを押す



The screenshot shows the Docker Hub website. The main heading is "Build and Ship any Application Anywhere". Below it, a subheading states: "Docker Hub is the world's easiest way to create, manage, and deliver your team's container applications." The navigation bar includes "Explore", "Pricing", "Sign In", and a "Register" button. A red rectangular box highlights the "Get Started Today for Free" registration form. This form includes fields for "Username", "Email", and "Password" (with a toggle for visibility). Below these fields are three checkboxes: "Send me occasional product updates and announcements.", "I agree to the Subscription Service Agreement, Privacy Policy, and Data Processing Terms.", and a CAPTCHA checkbox labeled "私はロボットではありません" (I am not a robot) with the text "CAPTCHA プライバシー・利用規約" (CAPTCHA Privacy Policy Terms of Use). At the bottom of the form is a "Sign Up" button.

DockerHubのアカウント作成

- 登録したメールアドレス宛に確認メールが送られる
- Verify email addressを押下しメール認証を行う。



DockerHubにログインする

「docker login」

→先程入力したユーザー名とパスワードを入力してログインする

```
amembo@Dell-13th:~$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username:
```


Docker-Composeのインストール

DockerComposeのインストールする¹

DockerComposeをインストールするフォルダを作りパスを通す

- ・パスを追加する, bin直下に環境変数を通す

「mkdir ~/bin && echo “export PATH=~/bin:\$PATH” >> ~/.bashrc」を入力する.

```
kaito-houjho@ex002:~/data/git/nerfstudio/.devcontainer$ mkdir ~/bin && echo "export PATH=~/bin:$PATH">> ~/.bashrc
```

DockerComposeのインストールする²

作成したフォルダにDockerComposeを追加する.

「curl -L
“https://github.com/docker/compose/releases/download/v2.17.3/docker-compose-\$(uname -s)-\$(uname -m)” -o ~/bin/docker-compose」を入力する.

```
kaito-houjho@ex002:~/data/git/nerfstudio/.devcontainer$ curl -L "https://github.com/docker/compose/releases/download/v2.17.3/docker-compose-$(uname -s)-$(uname -m)" -o ~/bin/docker-compose
```

DockerComposeのインストールする³

先程インストールしたdocker-composeに権限をふる

「`chmod +x ~/bin/docker-compose`」と入力

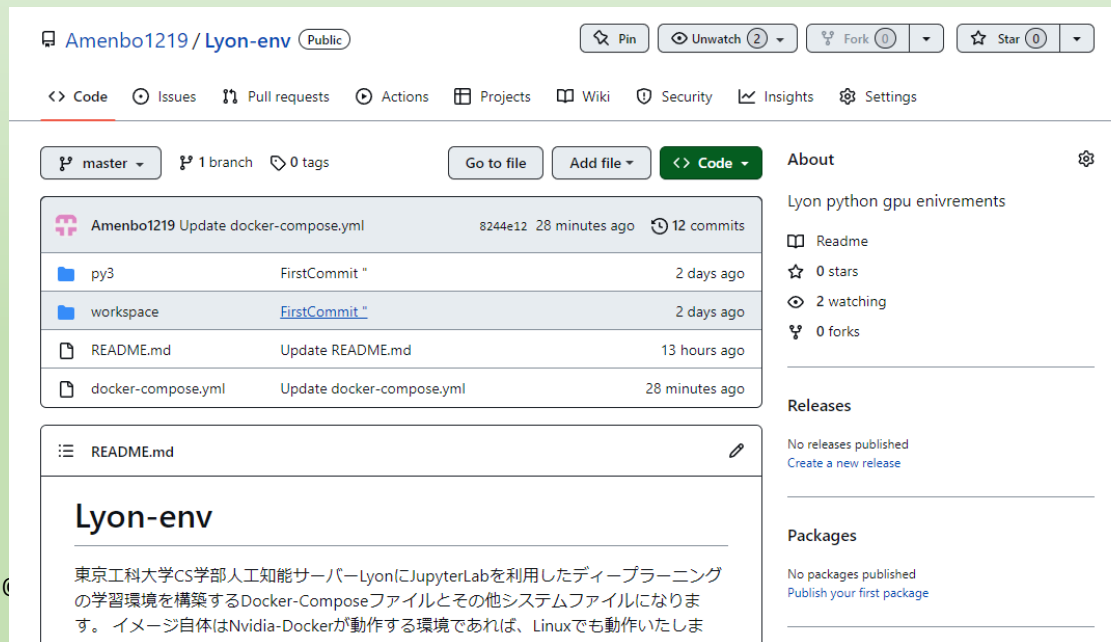
```
kaito-houjho@ex002:~/data/git/nerfstudio/.devcontainer$ chmod +x ~/bin/docker-compose
```

Colabのイメージを起動する

Colabと同等環境を動かす

システムのバージョン管理を行うシステム。
今回はコードを共有するためのツールとして利用。

<https://github.com/Amenbo1219/Lyon-env>

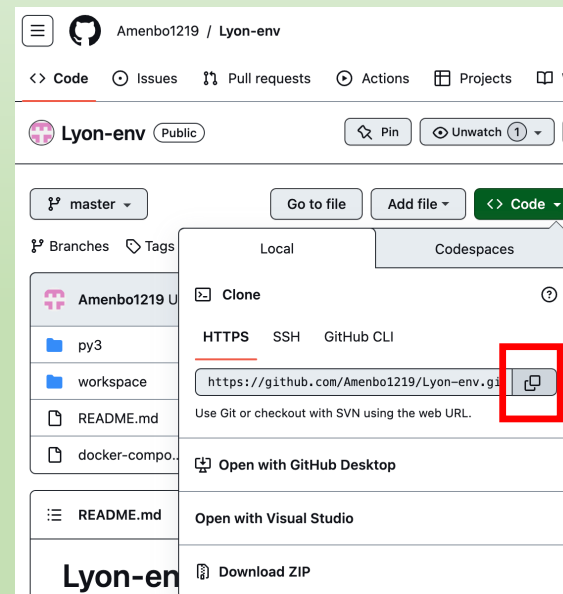


Gitを使って，環境を引っ張ってくる。

<https://github.com/Amenbo1219/Lyon-env>のページに飛ぶ

→<>Codeボタンを押す

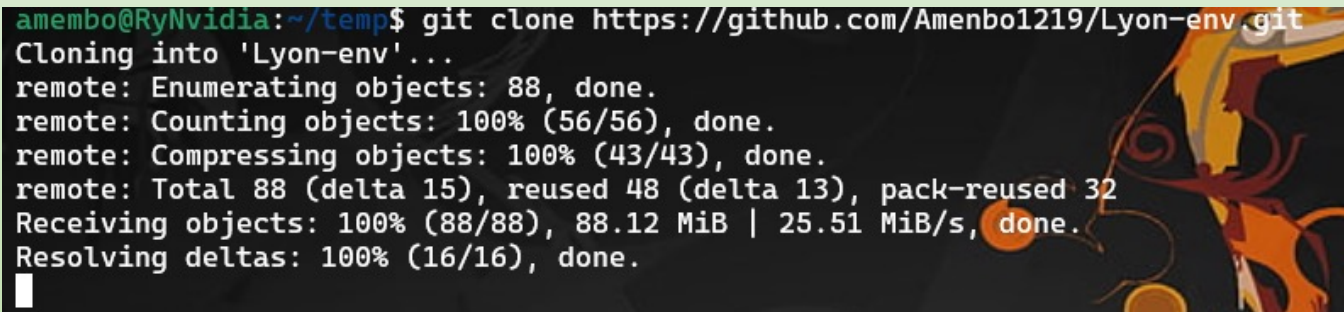
→Local>HTTPS>リンク>  を押してリンクをコピー



レポジトリをクローンする。

git clone コピーしたリンクをペースト する。

「git clone https://github.com/Amenbo1219/Lyon-env.git」と
入力

A terminal window with a dark background and a colorful, abstract pattern on the right side. The text shows the command 'git clone https://github.com/Amenbo1219/Lyon-env.git' being executed. The output indicates that the repository is being cloned into a directory named 'Lyon-env'. It shows progress for enumerating, counting, and compressing objects, and finally receiving the objects and resolving deltas. The process completes successfully, and a cursor is visible at the end of the last line.

```
amembo@RyNvidia:~/temp$ git clone https://github.com/Amenbo1219/Lyon-env.git
Cloning into 'Lyon-env'...
remote: Enumerating objects: 88, done.
remote: Counting objects: 100% (56/56), done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 88 (delta 15), reused 48 (delta 13), pack-reused 32
Receiving objects: 100% (88/88), 88.12 MiB | 25.51 MiB/s, done.
Resolving deltas: 100% (16/16), done.
█
```

→\$マークが出てきたらクローン完了。

クローンしたファイルを確認

- 「cd Lyon-env」でフォルダの中に移動

```
amembo@RyNvidia:~/temp$ cd Lyon-env/  
amembo@RyNvidia:~/temp/Lyon-env$
```

- 「ls」でフォルダの中身を表示

```
amembo@RyNvidia:~/temp/Lyon-env$ ls  
README.md  docker-compose.yml  py3  workspace  
amembo@RyNvidia:~/temp/Lyon-env$
```


上記のようになっていることを確認する。

ポート番号を変更する。

「vi docker-compose.yml」と入力する
→エディタの編集画面が立ち上がる

```
Version: "3.9"
services:
  py3:
    build:
      context: ./py3
      dockerfile: Dockerfile
      # メモリのサイズを変える
      shm_size: '8gb'
    restart: always
    entrypoint: >
      jupyter-lab
      --allow-root
      --ip=0.0.0.0
      --port=8888
      --no-browser
      --notebook-dir=/workspace
    expose:
      - "8888"
    # この左のポート番号を好きな値に設定
    ports:
      - "8888:8888"
    volumes:
      - ./py3/root_jupyter:/root/.jupyter
      - ./workspace:/workspace

    # GPUを使う場合の設定
    environment:
      "docker-compose.yml" 35L, 768B
```



ポート番号を変更

- portsの下にある左の4桁の数字を好きな数字に変更する

```
#ここの左のポート番号を好きな値に設定  
ports:  
- "8888:8888"
```

なるべく他の人と被らないような数字に変更する
この変更した数字がPort番号となるので覚えておく。

イメージのビルドをする

- 「docker compose build」と入力

```
amembo@RyNvidia:~/temp/Lyon-env$ docker compose build
[+] Building 1.5s (14/14) FINISHED
=> [py3 internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                       0.0s
=> [py3 internal] load build definition from Dockerfile            0.0s
=> => transferring dockerfile: 1.07kB                                0.0s
=> [py3 internal] load metadata for docker.io/tensorflow/tensorflow:2.12.0-gpu 1.3s
=> [py3 1/9] FROM docker.io/tensorflow/tensorflow:2.12.0-gpu@sha256:ce32f3c53f94938de1513b3fe5fa25a 0.0s
=> [py3 internal] load build context                                0.0s
=> => transferring context: 4.26kB                                    0.0s
=> CACHED [py3 2/9] RUN ln -sf /usr/share/zoneinfo/Asia/Tokyo /etc/localtime 0.0s
=> CACHED [py3 3/9] RUN apt-get update && apt-get install -y tzdata git wget libglib 0.0s
=> CACHED [py3 4/9] RUN pip3 install torch torchvision torchaudio --index-url https://download.pytor 0.0s
=> CACHED [py3 5/9] RUN pip install --no-cache-dir pip install praat-parselmouth 0.0s
=> CACHED [py3 6/9] COPY requirements.txt /install/requirements.txt 0.0s
=> CACHED [py3 7/9] RUN pip3 install --upgrade pip 0.0s
=> CACHED [py3 8/9] RUN pip3 install -r /install/requirements.txt 0.0s
=> CACHED [py3 9/9] RUN export TF_CPP_MIN_LOG_LEVEL=2 0.0s
=> [py3] exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:6b3bd2569ec574a6503c59fb3a70aa4f1bfe9a39ad6ff20627059c061f408a31 0.0s
=> => naming to docker.io/library/lyon-env-py3 0.0s
amembo@RyNvidia:~/temp/Lyon-env$
```

\$マークが表示されるまでしばらく待つ

イメージを起動する

「docker compose up -d」と入力しイメージを起動する

```
amembo@RyNvidia:~/temp/Lyon-env$ docker compose up -d
[+] Building 0.0s (0/0)
[+] Running 2/2
  ✓ Network lyon-env_default    Created      0.1s
  ✓ Container lyon-env-py3-1    Started     0.9s
amembo@RyNvidia:~/temp/Lyon-env$
```

\$マークが表示されるまでしばらく待つ

JupyterのTokenを確認する

「docker compose logs」と入力しイメージのログを確認する
このような画面が表示される

```
lyon-env-py3-1 | [I 2023-07-08 02:08:48.274 ServerApp] panel.io.jupyter_server_extension | extension was su  
ccessfully linked.  
lyon-env-py3-1 | [I 2023-07-08 02:08:48.302 ServerApp] notebook_shim | extension was successfully loaded.  
lyon-env-py3-1 | [I 2023-07-08 02:08:48.302 LabApp] JupyterLab extension loaded from /usr/local/lib/python3  
.8/dist-packages/jupyterlab  
lyon-env-py3-1 | [I 2023-07-08 02:08:48.302 LabApp] JupyterLab application directory is /usr/local/share/ju  
pyter/lab  
lyon-env-py3-1 | [I 2023-07-08 02:08:48.305 ServerApp] jupyterlab | extension was successfully loaded.  
lyon-env-py3-1 | [I 2023-07-08 02:08:48.311 ServerApp] nbclassic | extension was successfully loaded.  
lyon-env-py3-1 | [I 2023-07-08 02:08:48.311 ServerApp] panel.io.jupyter_server_extension | extension was su  
ccessfully loaded.  
lyon-env-py3-1 | [I 2023-07-08 02:08:48.312 ServerApp] Serving notebooks from local directory: /workspace  
lyon-env-py3-1 | [I 2023-07-08 02:08:48.312 ServerApp] Jupyter Server 1.24.0 is running at:  
lyon-env-py3-1 | [I 2023-07-08 02:08:48.312 ServerApp] http://2a7f64d63d63:8888/lab?token=872bf3fbe9ad966d8  
57f7bf6b2e25d1270dd0ab10a4a1fef  
lyon-env-py3-1 | [I 2023-07-08 02:08:48.312 ServerApp] or http://127.0.0.1:8888/lab?token=872bf3fbe9ad966d  
857f7bf6b2e25d1270dd0ab10a4a1fef  
lyon-env-py3-1 | [I 2023-07-08 02:08:48.312 ServerApp] Use Control-C to stop this server and shut down all  
kernels (twice to skip confirmation).  
lyon-env-py3-1 | [C 2023-07-08 02:08:48.315 ServerApp]  
lyon-env-py3-1 |  
lyon-env-py3-1 | To access the server, open this file in a browser:  
lyon-env-py3-1 | file:///root/.local/share/jupyter/runtime/jpservice-1-open.html  
lyon-env-py3-1 | Or copy and paste one of these URLs:  
lyon-env-py3-1 | http://2a7f64d63d63:8888/lab?token=872bf3fbe9ad966d857f7bf6b2e25d1270dd0ab10a4a1fe  
f  
lyon-env-py3-1 | or http://127.0.0.1:8888/lab?token=872bf3fbe9ad966d857f7bf6b2e25d1270dd0ab10a4a1fe  
f  
amembo@RyNvidia:~/temp/Lyon-env$
```


JupyterのTokenを確認する

先程の出力から、`http://....?token=....`の部分を探す
token=以降をコピーする（それがToken）

```
lyon-env-py3-1 | or copy and paste one of these URLs:  
lyon-env-py3-1 | http://2a7f64d63d63:8888/lab?token=872bf3fbe9ad966d857f7bf6b2e25d1270dd0ab10a4a1fe  
f  
lyon-env-py3-1 | or http://127.0.0.1:8888/lab?token=872bf3fbe9ad966d857f7bf6b2e25d1270dd0ab10a4a1fef
```

この出力の場合、
「872bf3fbe9ad966d857f7bf6b2e25d1270dd0ab10a4a1fe」
がTokenとなる。

TokenからURLを導出

- 先程SSH接続する際に利用したホスト名を用意する (P27, P32)
 - 先程設定したポート番号も用意する (P59)
- 今回はlyon001.cloud.cs.priv.teu.ac.jpをホスト名とする。

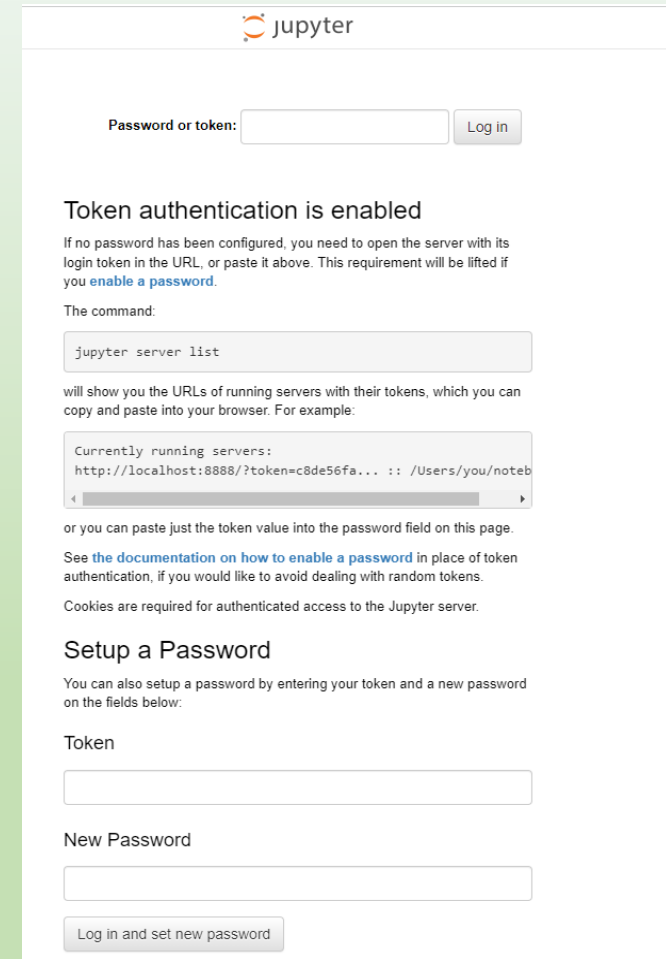
以下のルールでURLを導出する

→http://ホスト名:ポート番号←これがURLとなる。

例：) <http://lyon001.cloud.cs.priv.teu.ac.jp:8888>

自分の決めたURLにアクセスする

- ブラウザを起動
 - アドレスバーにP64のURLをペースト
- このような画面が表示されればOK→

A screenshot of the Jupyter web interface. At the top, the Jupyter logo is visible. Below it, there is a login section with a label "Password or token:" followed by a text input field and a "Log in" button. A message states "Token authentication is enabled" and explains that if no password is configured, the user must provide a login token in the URL. It includes a link to "enable a password" and shows the command "jupyter server list" in a code block. Below this, it says "will show you the URLs of running servers with their tokens, which you can copy and paste into your browser. For example:" and displays a list of "Currently running servers:" with an example URL: "http://localhost:8888/?token=c8de56fa... :: /Users/you/noteb". It then says "or you can paste just the token value into the password field on this page." and refers to documentation on enabling a password. A note mentions "Cookies are required for authenticated access to the Jupyter server." Below this is a section titled "Setup a Password" which explains that a password can be set by entering a token and a new password. It includes input fields for "Token" and "New Password", and a "Log in and set new password" button at the bottom.

jupyter

Password or token:

Token authentication is enabled

If no password has been configured, you need to open the server with its login token in the URL, or paste it above. This requirement will be lifted if you [enable a password](#).

The command:

```
jupyter server list
```

will show you the URLs of running servers with their tokens, which you can copy and paste into your browser. For example:

```
Currently running servers:
http://localhost:8888/?token=c8de56fa... :: /Users/you/noteb
```

or you can paste just the token value into the password field on this page.

See [the documentation on how to enable a password](#) in place of token authentication, if you would like to avoid dealing with random tokens.

Cookies are required for authenticated access to the Jupyter server.

Setup a Password

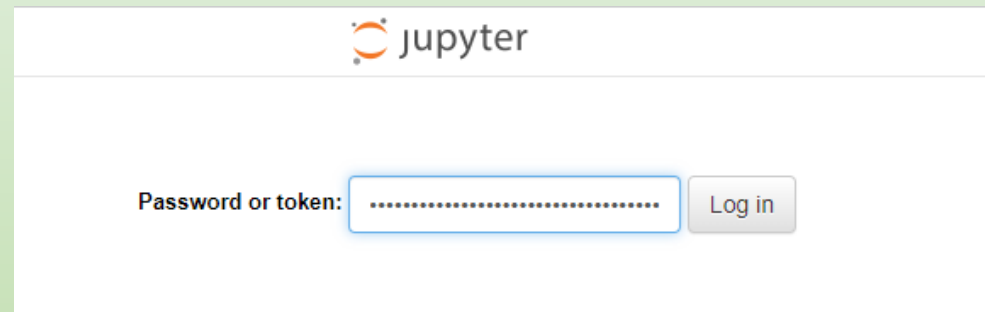
You can also setup a password by entering your token and a new password on the fields below:

Token

New Password

JupyterLabにログインする

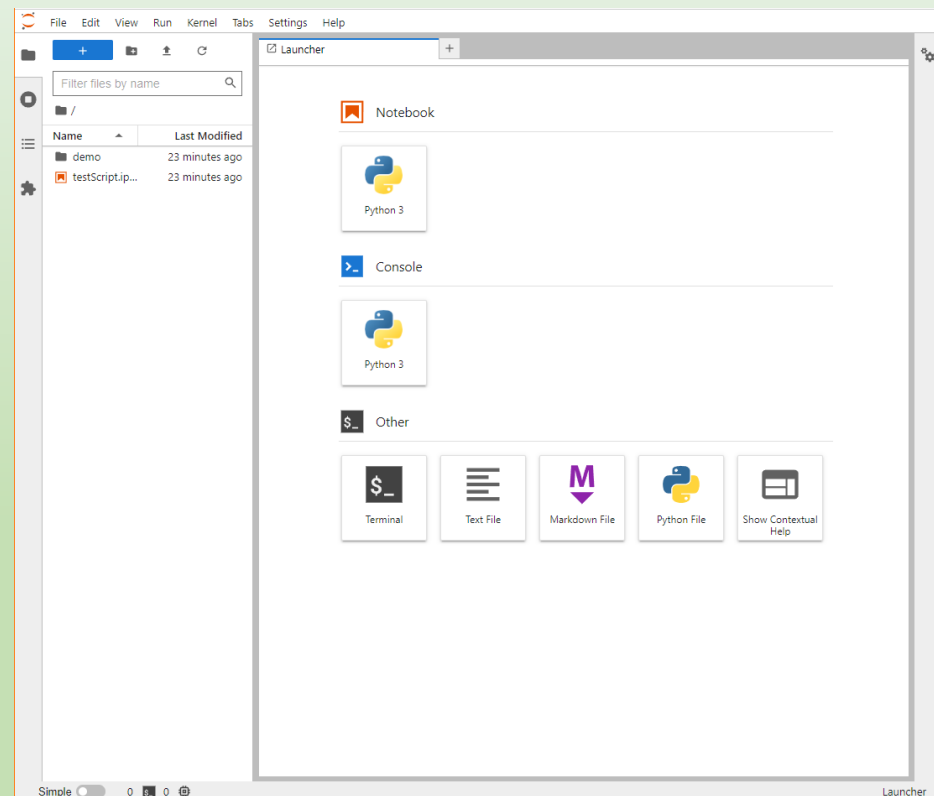
- 先程のToken or Passwordの部分にP63で導出したTokenをペースト

A screenshot of the Jupyter login interface. At the top, there is a header with the Jupyter logo (an orange circle with a white 'j' inside) and the word 'jupyter' in a sans-serif font. Below the header, the text 'Password or token:' is displayed. To the right of this text is a text input field with a blue border and a series of dots inside, indicating a password or token. To the right of the input field is a grey button with the text 'Log in' in a sans-serif font.

ログイン完了

- とりあえずこの画面が出ればOK

お疲れ様でした。



実験終了後

- 使用したコンテナを使わないときは落としておく必要がある。
- 再びUbuntuに移動する。
- 「docker compose down 」と入力

```
amembo@RyNvidia:~/temp/Lyon-env$ docker compose down  
[+] Running 2/2  
✓ Container lyon-env-py3-1   Removed  
✓ Network lyon-env_default   Removed  
amembo@RyNvidia:~/temp/Lyon-env$
```

両方RemovedになってればOK。

もっとDockerImageを活用したい

- <https://github.com/Amenbo1219/Lyon-env>のReadMeにイメージ関連の情報の記載あり
- 興味があればぜひご一読を！

Lyon-env version:1.1a

東京工科大学CS学部人工知能サーバーLyonにJupyterLabを利用したディープラーニングの学習環境を構築するDocker-Composeファイルとその他システムファイルになります。イメージ自体はNvidia-Dockerが動作する環境であれば、Linuxでも動作いたします。

[Lyon公式ページ](#)

Requirement

- GPUサーバー環境（SSH接続でのログインを想定）
- Dockerシステムの環境構築済みであること
- DockerCompose環境

[LyonにDockerを設定する](#) | [LyonクラウドVMの操作](#)

[LyonにDockerComposeをインストールする](#) | [準備（はじめに）](#)

Installation

GitHubからシステムを取得する

```
git clone https://github.com/Amenbo1219/Lyon-env.git
```

Lyon-Envのディレクトリに移動する

```
cd Lyon-env
```

Docker-compose をビルドする

注意事項

共用環境の場合，ディスクを過剰に消費しないよう，以下の点に注意してください．

- dockerを起動する必要がない場合は，dockerstop.shにてdockerプロセスを終了させてください．これをしないと，ログアウト後も残っています．
- コンテナやdockerイメージで不要なものがあればこまめに削除してください（contdel_7d.shやimdel_7d.sh等）
- ホームディレクトリで不要なものがあれば，こまめに削除してください．
- **本Kaggle（専用VM）をKaggle道場以外の用途に使わないでください．**

引用：

<https://sites.google.com/edu.teu.ac.jp/csccloud#h.bnoodj595jg>

[演習課題]専有VMに接続しよう！

- 課題内容：専有VMに接続しJupyterを起動しよう
- ルール：
 - 専用VMに接続しDemoスクリプトを動かす実行する。
 - わからないところは必ず質問する（友達と相談してもOK！）
 - **必ずパスワードを変更する**
- ヒント：
 - Lyonの接続方法：P31-
 - Dockerのインストール方法：P40-

接続情報

- ホスト名：P3Iを参照
- ユーザ名：学籍番号
- パスワード：hoge hoge(初回ログイン後、必ず変更すること)

お疲れさまでした