

# 開発環境の構築手順

ここで示す手順では、以下の開発環境を構築し、Pythonコンテナを動かしてみるところまで行います。

### 構築する環境構成

5	Pythonコンテナ	-	-	-	-
4	Docker Engin	Git	Docker Desktop	-	-
3	AlmaLinux-9		-	-	-
2	Windows Subsystem Linux 2			VSCode + 拡張機能	Git
1	Windows 10 or 11 (x86_64)				

Mac や Copilot+ PC などの Intel/AMD 以外の CPU 搭載の PC は対象外です

## 前提条件

- **GitHub のアカウント登録とリポジトリの作成が済んでいること**  
VSCode をインストールした後、リポジトリのクローン/コミット/同期を行い動作確認をします
- リポジトリの作成が済んでいない場合で、作り方がわからない場合は、こちらを参考にしてください。「[GitHub でウェブサイトを公開する/Zenn](#)」

## 1. 構築の流れ

1. WSL2 のインストール
2. AlmaLinux-9 のインストール
3. VSCode のインストール
4. Git(Windows用) のインストール
5. Docker Desktop for Windows のインストール
6. Docker Enjin のインストール
7. Git(Linux用) のインストール
8. Python コンテナの作成

## 2. 構築手順

### 2-1. WSL2 のインストール

1. Windows + x を押下し、メニューから「ターミナル（管理者）」をクリックします



2. 「ユーザ アカウント制御」が表示された場合は、[はい] をクリックします



3. 以下のコマンドを実行して、WSL2 をインストールします

```
wsl --install
```

要求された操作は正常に終了しました。変更を有効にするには、システムを再起動する必要があります。

と表示されること

Ubuntu のインストールが始まった場合は、そのまま続けます  
Ubuntu で使用するユーザー名、パスワードを入力してください  
`exit` と入力し、Wsl を一旦終了し、「2-2. AlmaLinux-9 のインストール」へ進みます

4. ターミナルを閉じ、Windows を再起動します

## 2-2. AlmaLinux-9 のインストール

1. もう一度、Windows ターミナルを起動します
2. 以下のコマンドを実行して、ディストリビューションの一覧を表示します

```
wsl --list --online
```

AlmaLinux-9 が表示されていること

### 3. 以下のコマンドを実行して、AlmaLinux-9 をインストールします

```
wsl --install AlmaLinux-9
```

ディストリビューションが正常にインストールされました。'wsl.exe -d AlmaLinux-9' を使用して起動できます

と表示されること

- 「Linux 用 Windows サブシステムへようこそ」のウィンドウが表示された場合は、閉じてしまっても問題ありません。  
これをもう一度開きたい場合は、メニューから「WSL Settings」を開き、左下の「WSLへようこそ」をクリックします。

### 4. 以下のコマンドを実行して、AlmaLinux-9 を起動します

```
wsl -d AlmaLinux-9
```

### 5. Enter new UNIX username: と表示されるので、使用したいユーザー名を入力します

### 10. New password と表示されるので、パスワードを入力します

- 1文字以上
- 入力したキーは表示されません

### 6. Retype new password: と表示されるので、もう一度同じパスワードを入力します

[...] \$ が表示されること

- ☐ の中は個々人の環境で違います

### 7. 以下のコマンドを実行して、特権ユーザーに切り替えます

```
sudo su -
```

- ユーザーパスワードを聞かれるので入力します

8. 以下のコマンドを実行して、`sudoers` を編集します  
これにより、`sudo` コマンドを実行してもパスワードを聞かれないようにします

```
vim /etc/sudoers
```

9. 以下のコマンドを入力して、行番号を表示します

```
:set number
```

- `:` キーを押すと、ターミナル下部に「:」が表示されるので、続けて `set number` と入力し、`Enter` を押します
- 行頭に行番号が表示されます

10. 111行目にカーソルキーで移動し、`i` キーを押して、編集モードにします

- 下部に `-- INSERT --` と表示された状態にします

11. ここに以下の行を追記します

```
{username}    ALL=(ALL)    NOPASSWD: ALL
```

- `{username}` は、登録したユーザー名を入力します
- スペースの数は、110行目等を参考にしてください
- 改行は任意で構いません
- `Warning: Changing a readonly file` と表示されていますがそのまま大丈夫です

12. 入力が終わったら、`Esc` を押します

13. 強制上書き保存とエディターを終了するため、以下のコマンドを入力します

```
:wq!
```

- 入力したら、`Enter` を押します

14. 以下のコマンドを入力し、特権ユーザーから抜けます

```
exit
```

15. 以下のコマンドを実行して、設定ができているか確認します

```
sudo ls /root
```

- パスワードの入力を求められなければ設定できています

16. 以下のコマンドを実行して、ディストリビューションをアップデートします

```
sudo dnf update -y
```

**Complete!** と最後に表示されること

17. 以下のコマンドを2回入力して、wsl と ターミナルを終了します

```
exit
```

## 2-3. VSCode のインストール

1. VSCode をダウンロードします

- ここから [Download Visual Studio Code](#) ダウンロード

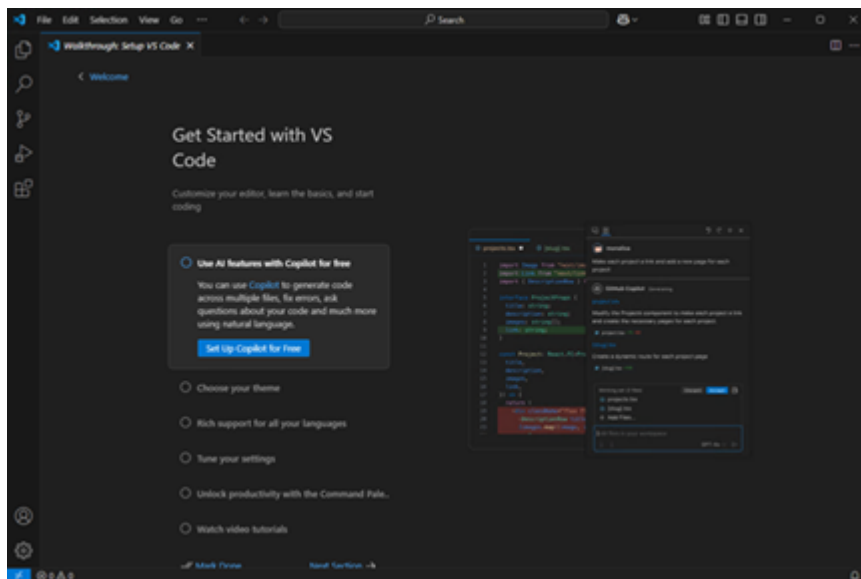
2. インストーラーを実行します

- オプション等は基本的にそのままで大丈夫ですが、デスクトップアイコンがほしい場合は、「追加のタスクの選択」で、アイコンを追加するにチェックを入れておいてください。

アイコンを追加する:

- ☒ デスクトップ上にアイコンを作成する

### 3. セットアップウィザードが完了したら、VSCode を起動します

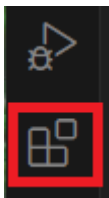


#### 起動すること

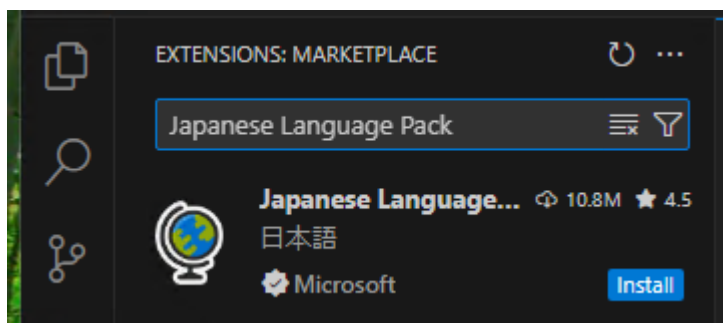
### 4. 拡張機能をインストールします

#### • Japanese Language Pack for Visual Studio

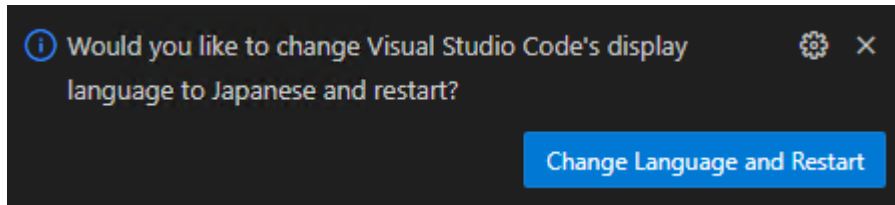
- Extensions のアイコンをクリックします



- 検索窓に「Japanese Language Pack」と入力し、Japanese Language Pack for Visual Studio の **Install** をクリックします



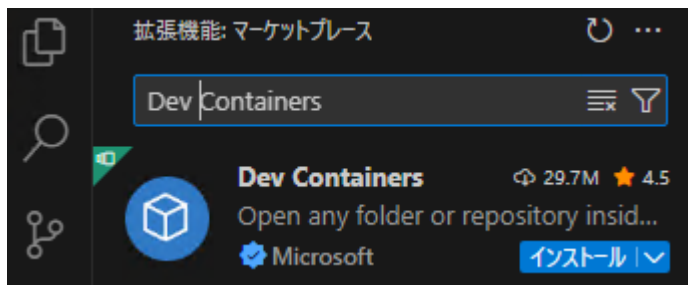
- Change Language and Restart と右下に表示されるので、これをクリックして、VSCode を再起動します



日本語表記になっていること

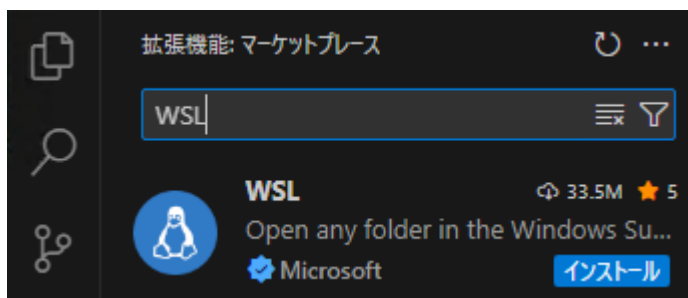
- **Dev Containers**

- Extensions のアイコンをクリックします
- 検索窓に「Dev Containers」と入力し、Dev Containers の **インストール** をクリックします



- **WSL**

- Extensions のアイコンをクリックします
- 検索窓に「WSL」と入力し、WSL の **インストール** をクリックします



5. VSCode を終了します

## 2-4. Git(Windows用) のインストール

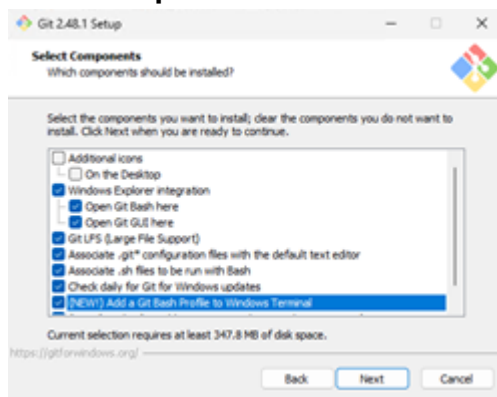
1. Git for Windows をダウンロードします

- ここから [Download for Windows](#) ダウンロード

2. インストーラーを実行します

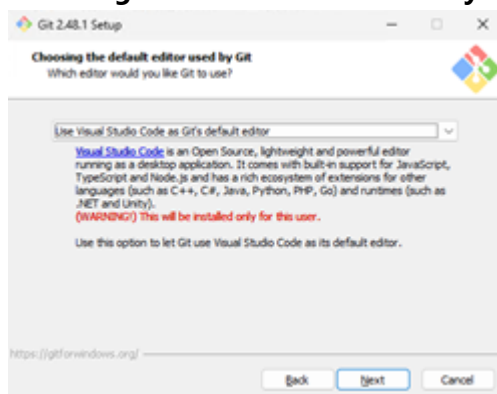
- オプション設定は以下の画像を参考に選択してください
- 画像がない部分については、そのままNextで進んでください

## ○ Select Components



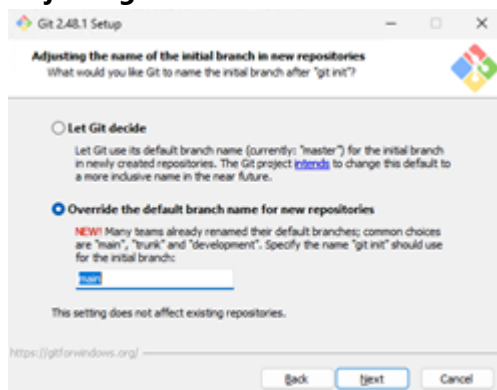
- 以下の2項目は必ずチェックを入れてください。
  - ☒ Check daily for Git for Windows updates
  - ☒ Add a Git Bash Profile to Windows Terminal
- 以下の項目はチェックを入れても外しても任意で大丈夫です
  - ☐ Additional icons
    - ☐ On the Desktop
  - ☒ Windows Explorer intergration
    - ☒ Open Git Bash here
    - ☒ Open Git GUI here

## ○ Choosing the default editor used by Git



- Use Visual Studio Code as Git's default editor を選択します

## ○ Adjusting the name of the initial branch in new repositories

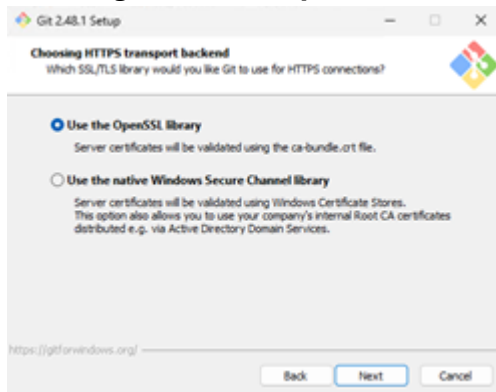


- Override the default branch name for repositories を選択します



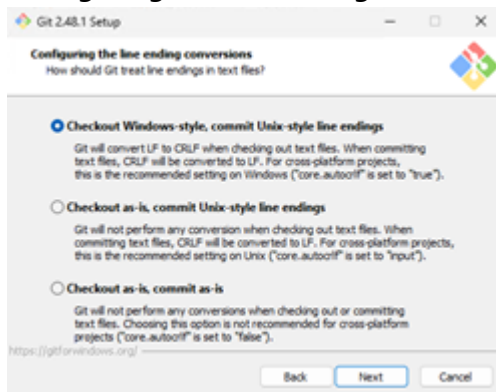
- `main` はそのままにしてください。

## ◦ Choosing HTTPS transport backend



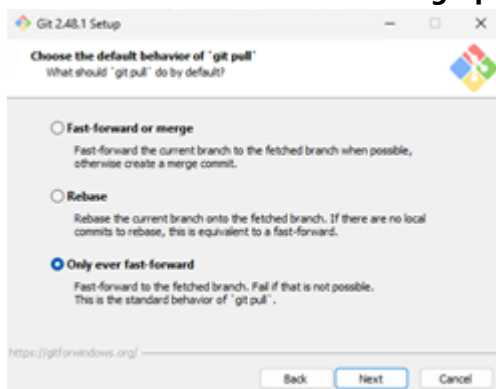
- Use the OpenSSL library を選択します

## ◦ Configuring the line ending conversions



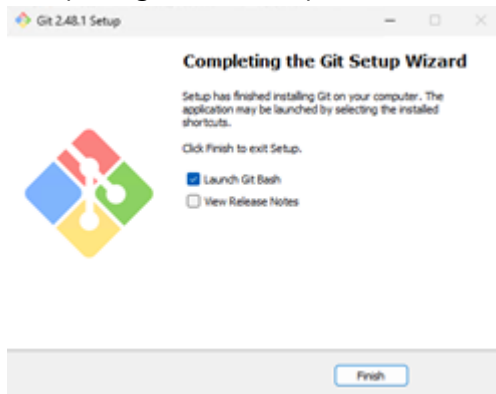
- Checkout Windows-style, commit Unix-style line endings を選択します

## ◦ Choose the default behavior of 'git pull'



- Only ever fast-forward を選択します

### ◦ Completing the Git Setup Wizard



- Launch Git Bash を選択します
- View Release Notes はどちらでもいいです

### 3. 以下のコマンドを実行して、Git Bash の動作確認をします

```
git --version
```

バージョン情報が表示されること

- `exit` コマンドを実行して、Git Bash を終了します

### 4. C:\Users\{username} ディレクトリにある `.gitconfig` を編集します

```
[core]
    editor = \"C:\\Users\\{username}\\AppData\\Local\\Programs\\Microsoft VS
Code\\bin\\code\" --wait
[user]
    name = {username}
    email = {email}
[pull]
    ff = only
    rebase = false
[core]
    eol = lf
    ignorecase = false
    quotepath = false
[init]
    defaultBranch = main
[fetch]
    prune = true
[rebase]
    autosquash = true
```

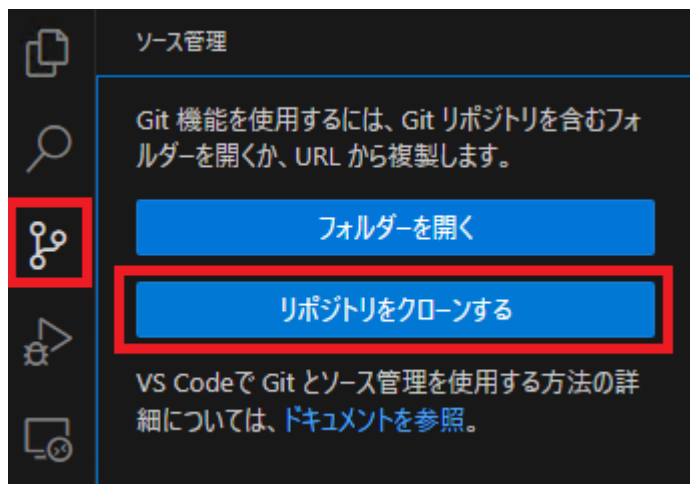
- 1 - 2行目は既に入力されていると思います
- 3行目([user]セクション)以降を転記してください
- name = {username} : GitHub のユーザ名に置き換えてください
- email = {email} : GitHub が提供しているコミットメールアドレスに置き換えてください

登録メールアドレスでもいいのですが、コミット情報として自分のメールアドレスが公開されてしまいます

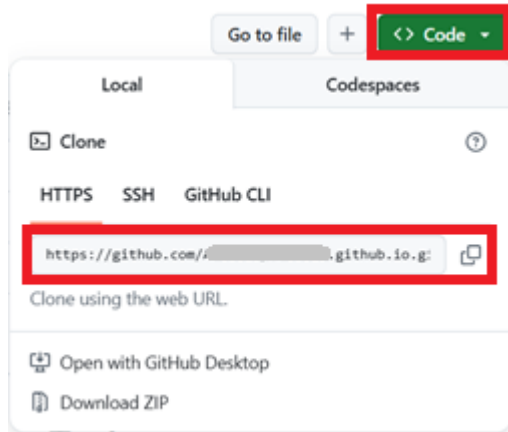
- コミットメールアドレスの確認の方法
  1. GitHub にログインします
  2. 右上の自分のアバターから Settings を選択します
  3. 左のメニューから Emails を選択します
  4. 画面を下の方にスクロールし "**Keep my email addresses private**" という項目をみつけます
  5. チェックボックスにチェックが入っていない場合は、チェックを入れます
  6. この文章中にある ID+USERNAME@users.noreply.github.com を控えておきます

## 5. VSCode と GitHub を連携しておきます

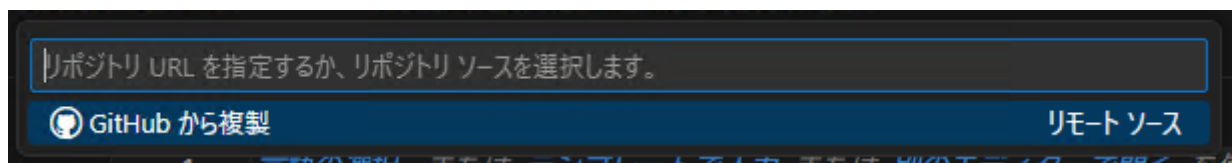
- 「ソースの管理」アイコンをクリックし、「リポジトリをクローンする」をクリックします



## 6. GitHub から自分のリポジトリの URL をコピーします



7. コピーした URL を VSCode に貼り付けます



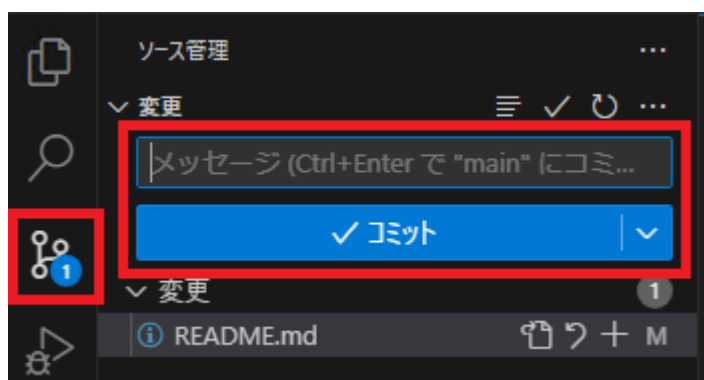
8. リポジトリの保存場所を聞かれるので、適宜フォルダを選択もしくは新規作成して、[リポジトリの宛先として選択] をクリックします

9. リポジトリを開くか聞かれるので、[開く] をクリックします

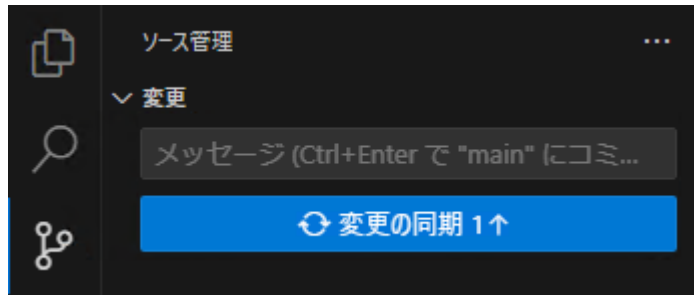
- リポジトリにアップしているファイルが表示されていれば OK です

10. いずれかのファイルを編集し上書き保存します

11. 「ソースの管理」アイコンをクリックし、コミット用のメッセージを入力して「コミット」をクリックします



## 12. 「変更の同期」をクリックします



- 認証を求められることがありますので、適宜認証してください
- git fetch するか聞かれた場合は、「はい」を選択しておいてください
- GitHub の自分のリポジトリを確認し、編集した内容が反映されていれば OK です
- その他、各種確認のメッセージが表示されることがありますので、よく読んで回答してください

## 13. VSCode を終了します

# 2-5. Docker Desktop for Windows のインストール

## 1. Docker Desktop for Windows をダウンロードします

- ここから [Docker Desktop](#) をダウンロード
- Windows用をダウンロード - AMD64 を選択してください



## 2. インストーラーを実行します

- WSL2 を使用するので、オプション設定はそのままにしてください
- インストール完了後再起動します
- Docker が自動起動するので待ちます

### 3. サブスクリプション契約について確認があるので、同意します



#### Docker Subscription Service Agreement

By selecting **accept**, you agree to the [Subscription Service Agreement](#), the [Docker Data Processing Agreement](#), and the [Data Privacy Policy](#).

Commercial use of Docker Desktop at a company of more than 250 employees OR more than \$10 million in annual revenue requires a paid subscription (Pro, Team, or Business). [See subscription details](#)

[View Full Terms](#)

**Accept**

Close

### 4. 個人用アカウントを登録済みの場合は、Personal タブを開き、メールアドレスを入力し、「Continue」をクリックします。未登録の場合は、右上の Skip をクリックします。

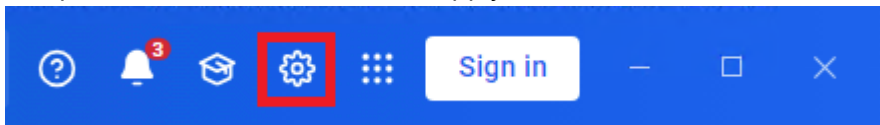
### 5. 立場を聞かれますので、該当する箇所を選択するか、Skip します。

### 6. Docker Desktop の起動完了を待ちます

- 右下に Multi-platfrom を有効化するか聞いてきますが、とりあえずそのままにしておいてください  
自動的に消えます
- Docker Desktop の起動中に、「WSLとの統合で予期せぬエラーで停止しました」と表示された場合は、「Restart」を押してください
- 右下に、New Version available と表示されていますが、とりあえずそのままにします

## 7. 自動起動設定をしておきます

- 右上の歯車アイコンをクリックし、「Start Docker Desktop when you sign in to your computer」にチェックを入れ、「Apply & restart」をクリックします



- Windows を起動した際に、Docker Desktop が開く設定になっています  
これを解除したい場合は、「Open Docker Dashboard when Docker Desktop starts」のチェックを外し、「Apply & restart」をクリックします

### General

- ☒ Start Docker Desktop when you sign in to your computer
- ☐ Open Docker Dashboard when Docker Desktop starts

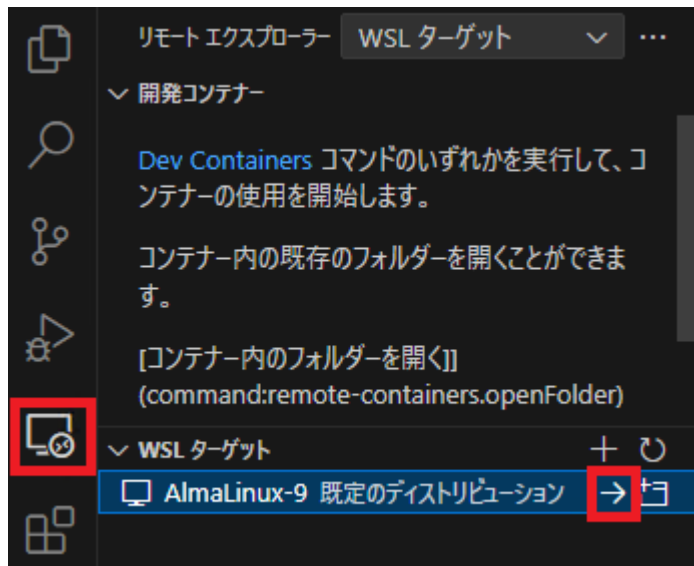
## 8. Docker Desktop は閉じてしまってもかまいません（常駐起動しています）

## 2-6. Docker Enjin のインストール

### 1. VSCode を起動します

- GitHub のリポジトリが開いたままになっている場合は、**Ctrl** + **K** を押した後、**F** を押して閉じておきます

### 2. リモートエクスプローラーで "AlmaLinux-9" に接続します



3. **Ctrl** + **@** を押し、ターミナルを開きます

4. インストール用リポジトリの追加します

```
sudo dnf config-manager --add-repo=https://download.docker.com/linux/centos/docker-ce.repo
```

Adding repo from: <https://download.docker.com/linux/centos/docker-ce.repo>

と表示されること

5. Docker Engine のインストールします

```
sudo dnf install -y docker-ce docker-ce-cli containerd.io
```

Complete!

と最終行に表示されること

6. Dockerを起動します

```
sudo sudo systemctl start docker
```

エラーが表示されないこと



## 7. Dockerの起動確認をします

```
sudo docker run hello-world
```

実行結果の 7-8 行目に以下の内容が表示されれば OK です

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

## 8. root ユーザー以外で Docker を実行できるようにします。

- この手順で構築した場合は、docker グループの作成やこのグループへの追加は不要ですが、以下のコマンドで確認できます

```
sudo cat /etc/group
```

- 実行結果の中に、`docker:x:1001:{username}` があれば OK です  
{username}は自分が登録したユーザー名  
数字の部分は違うかもしれません
- 見あたらなかった場合は、以下のコマンドを実行してください

```
sudo groupadd docker  
sudo usermod -aG docker $USER
```

- 一旦ログアウト (`exit`) して、再ログイン ( `Ctrl` + `@` ) します

## 9. sudo なしで、docker コマンドを実行できるか確認します

```
docker run hello-world
```

## 9. システムブート時の Docker の自動起動設定をします

```
sudo systemctl enable docker.service  
sudo systemctl enable containerd.service
```

## 2-7. Git(Linux用) のインストール

1. 以下のコマンドを実行して、Git インストールします

```
sudo dnf install -y git
```

```
Complete!
```

と最終行に表示されれば OK です

2. 以下のコマンドを実行して、Git バージョン確認します

```
git --version
```

```
git version 2.43.5
```

と表示されれば OK です

3. Git Credential Manager の設定をします

- WSLとWindowsホストの間で資格情報と設定を共有するために、WSLのLinuxのホームに `.gitconfig` を作成し、設定を書きます
- 以下のコマンドを実行して、ディレクトリを確認しておきます

```
pwd
```

```
/home/{username} になっていれば OK です
```

- 以下のコマンドを実行して、`.gitconfig` を作成します

```
vim .gitconfig
```

- 設定値
  - {username}, {email} は、2-4. 4 を参考にしてください
  - 編集方法は、2-2 を参考にしてください  
ただし、保存終了は `:wq` で大丈夫です

```
[user]
  name = {username}
  email = {email}
[pull]
  ff = only
  rebase = false
[core]
  eol = lf
  ignorecase = false
  quotepath = false
[init]
  defaultBranch = main
[fetch]
  prune = true
[rebase]
  autosquash = true
[credential]
  helper = /mnt/c/Program\ Files/Git/mingw64/bin/git-credential-
manager.exe
```

- 設定値の確認

```
git config --list --global
```

`.gitconfig` に設定した内容が表示されれば OK です

#### 4. GitHub との連携

- 自分のGitHubのRepositoryをcloneしてみます

```
git clone {GitHub_Repository_URL}
```

エラーがなく、クローン作成が完了すること

#### 5. リポジトリのフォルダがあることを確認します

```
ls -l
```

リポジトリ名と同じフォルダ名が表示されていれば OK です

## 2-8. Python コンテナの作成

### 1. Python コンテナ用の json ファイルを用意します

- 既に ms-python をダウンロード済みの場合は次へ進みます
- まだ用意していない場合は、[ここから](#)ダウンロードして解凍すると、ms-python が入っています  
こちらには動作確認用の `sample.py` を入れてあります

### 2. Windows のエクスプローラーで以下のフォルダを開きます

```
\\wsl.localhost\AlmaLinux-9\home
```

- 自分のユーザー名のフォルダがあると思いますので、さらにそのフォルダを開きます

### 3. 1 で用意した ms-python をフォルダごと、2 のフォルダにコピーします

- 直接置きたくない場合は、適宜フォルダを作成し、そのフォルダにコピーしてください  
日本語のフォルダも作れますが、できれば半角英数にしておいてください

### 4. VSCode のターミナルに戻り、`ls -l` コマンドを実行してください

- ms-python があれば OK です
- フォルダを作った場合は `ls -l フォルダ名` を実行してください

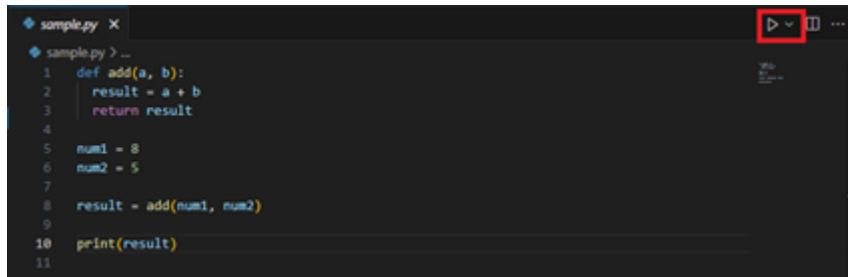
### 5. Python コンテナを起動します

- 以下のコマンドを実行します

```
cd ms-python/  
code .
```

- フォルダを作った場合は、`cd フォルダ名/ms-python/` にしてください
- `code .` を実行すると、初回は環境構築のためかなり時間がかかります
- 右下に「コンテナで再度開く」が表示されたら、必ずクリックしてください  
押しそびれるとコンテナが起動しないので、その場合は、左下の「WSL: AlmaLinux-9」をクリックして、画面上部に表示されたメニューから「コンテナで再度開く」を選択してください
-

6. sample.py を開き、右上の実行ボタンで実行してみます



```
sample.py X
sample.py > ...
1 def add(a, b):
2     result = a + b
3     return result
4
5 num1 = 8
6 num2 = 5
7
8 result = add(num1, num2)
9
10 print(result)
11
```

- 実行できたら、今度はデバッグを行ってみてください
- 適当な行にブレークポイントを設定します（行番号の左側をクリック、赤丸が付けば OK です）
- 実行ボタンの右のvをクリックし、Python デバッガー: Python ファイルのデバッグをクリックします
- 左側に変数ビューが表示されれば OK です

7. Python コンテナを終了します

- 左下の「開発コンテナ: ms-python」をクリックし、「リモート接続を終了する」をクリックします
- このウィンドウは閉じてしまいます

8. WSL を終了します

- 左下の「WSL: AlmaLinux-9」をクリックし、「リモート接続を終了する」をクリックします

9. VSCode を終了します

**以上、Python をコンテナで動かしてみるところまで実施できました  
お疲れさまでした**