

SNPcaster

インストール・ 操作マニュアル

2026年2月25日

SNPcaster 対応バージョン

v.0.9.6 以降

注意事項

本書に記載されている情報は、正確を期すために慎重に作成したのですが、誤りがないことを保障するものではありません。万一、本書に記載されている情報の誤りに起因する損害が使用者に生じた場合におきましても、作製者は一切その責任を負いません。

目次

目次	2
0. クイックスタート	6
1. 解析端末の必要スペック、基本的な使い方	7
1.1. 解析端末の必要スペック	7
1.2. コマンド入力アプリ	8
1.2.1. Windows	8
1.2.2. Mac	11
1.2.3. Linux	11
1.3. コマンドライン操作	12
1.4. ディレクトリ説明、移動	12
1.4.1. 移動ディレクトリ、パスとは	12
1.4.2. 指定したディレクトリでコマンド入力アプリを開く	13
1.4.3. カレントディレクトリの変更	15
2. Docker のインストール	18
2.1. Rancher Desktop のインストール	19
2.1.1. Windows	19
2.1.2. Mac	25
2.1.3. Rancher Desktop の起動設定	29
2.1.4. Rancher Desktop の動作確認	30
2.2. Docker Engine のインストール	31
2.2.1. Windows	31
2.2.2. Mac	33
2.2.3. Linux	38
2.2.4. Docker Engine の動作確認	42
3. SNPcaster の環境構築	43
3.1. SNPcaster のダウンロード	43
3.1.1. ブラウザ(Microsoft Edge, Google Chrome など)を使ってダウンロード	43
3.1.2. コマンドライン(Git)を使ってダウンロード	44

3.2. (参考) フォルダ・ファイル構成	45
3.3. 解析端末への SNPcaster インストール	48
3.3.1. .env ファイルの編集	48
3.3.2. ビルドの実行	49
3.3.3. ビルドに失敗した場合	51
3.4. SNPcaster のバージョンアップ	57
3.4.1. ブラウザ(Internet Explorer, Google Chrome など)を使ってダウンロードした場合	57
3.4.2. コマンドライン(Git)を使ってダウンロードした場合	59
4. 起動・停止方法	61
4.1. 起動方法	61
4.1.1. 起動コマンド実行	61
4.1.2. ポートフォワード設定 (リモートアクセスを行っている場合のみ)	62
4.1.3. Jupyter Lab へのアクセス	63
4.2. 停止方法	63
5. 解析実行手順	65
5.1. フォルダ構成	65
5.2. Jupyter Lab の使い方を学ぶ	66
5.2.1. Jupyter Lab チュートリアル	66
5.2.2. Jupyter Lab の基本的な使い方	67
5.3. プロジェクト作成	69
5.3.1. プロジェクトの新規作成実行	69
5.3.2. プロジェクトの新規作成実行	71
5.3.3. 解析の実行	72
5.3.4. 基本的な使い方	74
5.3.5. SNP 再解析の実行	78
6. コマンドライン操作 (上級者向け)	80
6.1. コマンド入力アプリの起動	80
6.1.1. Windows	80
6.1.2. Mac	80
6.1.3. Linux	80
6.2. Docker イメージ・コンテナの生成	81
6.3. Docker コンテナから実行	81
6.3.1. データの配置	81
6.3.2. コンテナ内に bash でアクセス	81
6.4. Docker イメージから実行	82

6.4.1. データの配置	82
6.4.2. イメージ ID の確認	82
6.4.3. コマンドの実行	83
7. プログラム解説	85
7.1. SNPcaster	85
7.1.1. プログラムの利点	85
7.1.2. 解析の流れ	87
7.1.3. 出力ファイル	90
7.2. grape_qc_assembly	95
8. トラブルシューティング、Q & A	97
9. 資料	100
9.1. Dockerについて	100
9.2. Docker Desktop のインストール方法	102
9.2.1. Windows	102
9.2.2. Mac	105
9.3. Docker Desktop のアンインストール方法	109
9.3.1. 作成済みのコンテナ・イメージのバックアップ	109
9.3.2. Docker Desktop のアンインストール	111
9.3.3. バックアップの復元	111
9.4. Rancher Desktop のアンインストール方法	112
9.4.1. Windows	112
9.4.2. Mac	115
10. 執筆者・引用法	117
10.1. 執筆者	117
10.2. Citation	117

改訂履歴

Ver.	日付	改訂内容(概要)
1.3	2026-02-25	v 0.9.6 に対応した内容に修正
1.2	2025-12-22	v 0.9.5 に対応した内容に修正 ユーザーからのフィードバックを基に詳細な説明を追加
1.1	2025-05-28	配布方法変更に伴う変更点を修正 v. 0.9.1 に対応
1.0	2024-06-07	初版発行

0. クイックスタート

まずは次の手順に従って、必要ソフトのインストールおよび基本的な解析を実行します。

「[1 解析端末の必要スペック、基本的な使い方](#)」にて解析端末の必要スペックやコマンド入力の基礎について確認します。コマンド操作に慣れている方は読み飛ばしていただいて構いません。

「[2 Docker のインストール](#)」に従って、Docker をインストールします。

「[3 SNPcaster の環境構築](#)」に従って、SNPcaster をインストールします。

「[4 起動・停止方法](#)」に従って、SNPcaster の Docker 環境を起動します。

「[5 解析実行手順](#)」に従って、解析を実行します。まずは、SNPcaster_quickstart.ipynb による解析を実行してみましょう。

なお、本マニュアルは主に SNPcaster のインストールに関わる説明を記載しております。

詳細な解析手順や出力ファイルの説明は Jupyter ノートブック中に記載しておりますので、そちらをご覧ください。

(Jupyter ノートブックの基本的な使い方は、「[5 解析実行手順](#)」に説明があります。)

1. 解析端末の必要スペック、基本的な使い方

1.1. 解析端末の必要スペック

次のスペックを推奨します。

※必須のスペックではありません。プログラムが動作しなかった場合に確認すれば十分です。

- OS: Windows 10 以降、Mac、Linux
※Linux のディストリビューションは、Ubuntu でのみ動作確認を行っています。
- メモリ: 40GB 以上(10GB 程度でも実行可能です)
- CPU: intel Core シリーズと同等以上の性能、または Apple チップ(M1,M2 等)
- Docker がインストールされるドライブの空き容量: 50 GB 以上
(snpcaster のインストールで約 20GB が消費されます)

1.2. コマンド入力アプリ

本プログラムを使用するには、多少のコマンドライン操作が必要です。まずコマンド入力の基本について、説明します。

OSごとに様々なコマンドアプリが使用できます。本マニュアルでは、これらをまとめて「**コマンド入力アプリ**」と記載します。

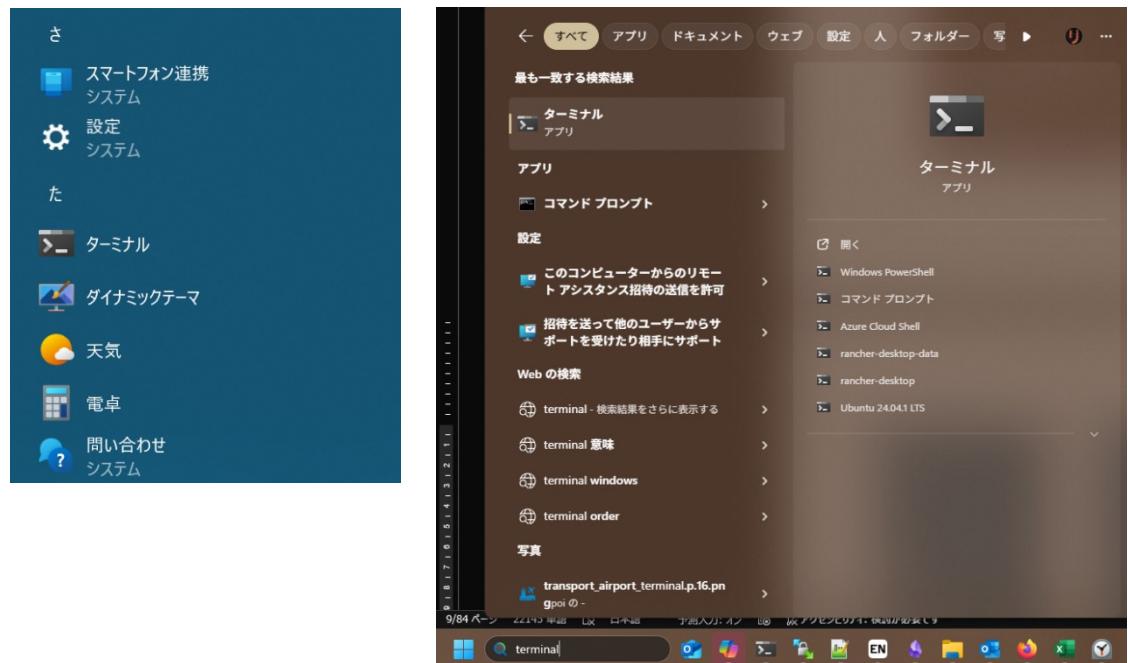
1.2.1. Windows

1.2.1.1. コマンド入力アプリについて

Windows PowerShell や Windows Terminal(「ターミナル」と表示される)等があります。Windows Terminal は、初期状態でインストールされていない場合があります。その場合は、Windows Store からインストールしてください。

Windows Terminal では、後述する「ターミナルで開く」が使用できますので、インストールするのがおすすめです。

(Windows 11 では初期状態でインストールされています)

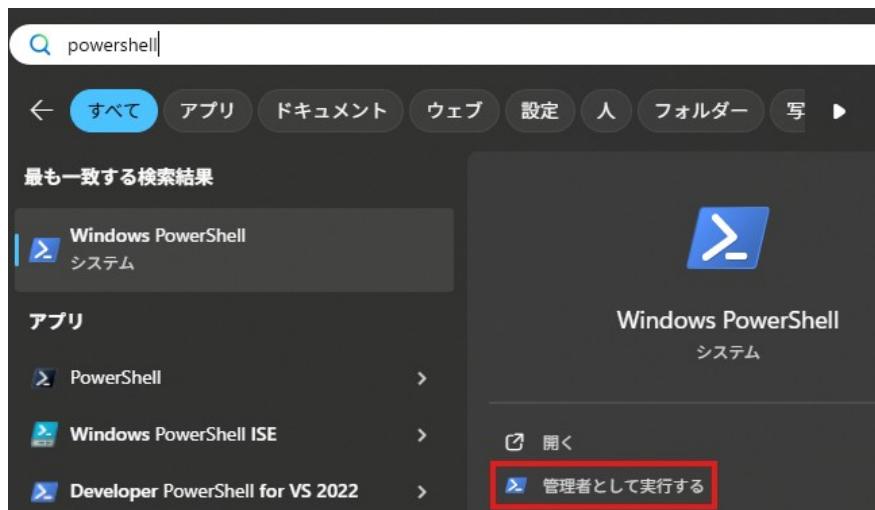


1.2.1.2. Windows Subsystem for Linux (WSL) のインストール

Windows では、Docker は WSL 上で起動するため、本項で WSL2 を有効化します。

※本項はメモリ設定を含むため、WSL をインストール済みの方もご一読ください※

Windows Powershell またはターミナルを管理者として実行します。



WSL を起動したことがない場合は、以下のコマンドを実行して初回起動します。

```
wsl --install -d Ubuntu-24.04
```

※「Ubuntu」部分は、Linux のディストリビューションを指します。別のディストリビューションでも動作すると考えられますが、Ubuntu 以外での検証は行っていません。

初回起動時は、以下のようにアカウントのユーザー名を求められます。

お好きなユーザー名を入力し、Enter を押して下さい。

数秒後、パスワードを求められるので入力し(画面に表示されません)、Enter を押してください。**※パスワードはインストール時に使用するので、必ず覚えておいてください。**

```
PS C:\Users\t-yamagishi> 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 ...
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: test
New password: |————— パスワードは入力した文字が表示されません
```

パスワード確認のため、再入力を求められます。パスワードを再度入力し、Enter を押します。

パスワードが正しければ、インストールが完了します。

```
Retype new password: [REDACTED]
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 5.15.167.4-microsoft-standard-WSL2 x86_64)

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

System information as of Tue Dec 10 15:27:37 JST 2024
```

次に、WSL に対するメモリ割り当ての設定を行います。

WSL はデフォルトでは最大 8GB しかメモリが割り当てられないため、設定を変更します。

ファイルエクスプローラー等で、C:\Users\[ユーザー名]\([ユーザー名]はお使いのユーザー名)を開いてください。

.wslconfig というファイルを作成します(すでにある場合は、そのファイルを編集してください)。

.wslconfig をテキストエディタ(「メモ帳」など)で開き、以下の内容を入力してください。

```
[wsl2]
memory={確保したいメモリサイズ}GB
```

{確保したいメモリサイズ}には、Docker に割り振り可能なメモリサイズを数値で入力します。

この数値は、お使いの PC のメモリ容量よりも 1GB 以上小さい値にしてください(Docker がメモリをすべて使うのを防ぐため)。

※メモリは 16GB 以上の設定を推奨します。

メモリの最大値は、Windows ロゴを右クリック > システム をクリックすると表示される画面の右側、「実装 RAM」に表示されます。

システム > バージョン情報



.wslconfig の変更を反映させるため、ターミナルを開いて以下のコマンドを実行します（WSL の再起動）。

```
wsl --shutdown  
wsl -d Ubuntu-24.04
```

これで、WSL の使用準備が整いました。

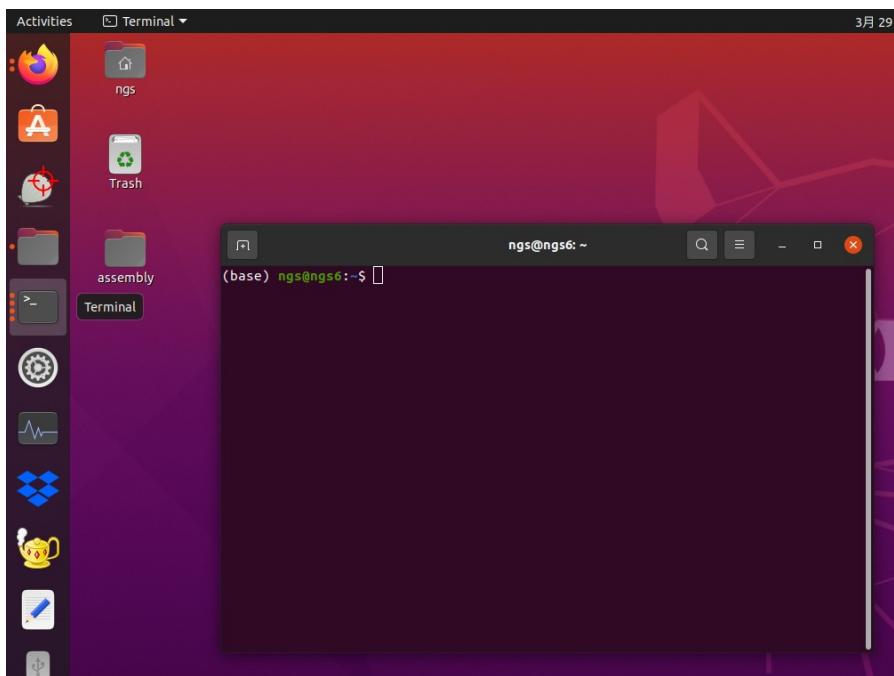
1.2.2. Mac

初期状態で「ターミナル」が使用可能です。



1.2.3. Linux

初期状態で「Terminal」が利用可能です。



1.3. コマンドライン操作

本マニュアルでは、コマンドライン部分(コマンド入力アプリに入力する部分)については、以下のような灰色網掛けで表示します。

```
$ echo hello world
```

「\$」マークは、コマンドを先頭を意味するもので、入力しません。ここでは、「echo hello world」のみを入力します。

また、コマンドを実行した結果表示される内容を、以下のような黒枠内に表示します。
表示される内容は、実行環境によって異なる可能性がありますことに、留意ください。

```
hello world
```

1.4. ディレクトリ説明、移動

1.4.1. 移動ディレクトリ、パスとは

「ディレクトリ」とは Windows では「フォルダ」と呼ばれているもので、ファイルが存在する場所を指します。

また、「パス」とは「ファイルの場所を示す場所」です。

コマンドラインの操作では、現在作業をしているディレクトリ(カレントディレクトリ)を意識して作業する必要があります。例えば、カレントディレクトリ内にあるファイルは、ファイル名(=相対パス)のみで指定ができますが、別のディレクトリにあるファイルはファイル名だけでは読み込みができません。その場合、絶対パス等を入れる必要があります。よく分からぬ場合は、絶対パスを使うのが無難です。

ディレクトリ、パスについて慣れていない方は、下記ウェブサイトの説明を必ず一読してください。

参考サイト:

<https://qiita.com/usamaro/items/f95680f239295d401b51>

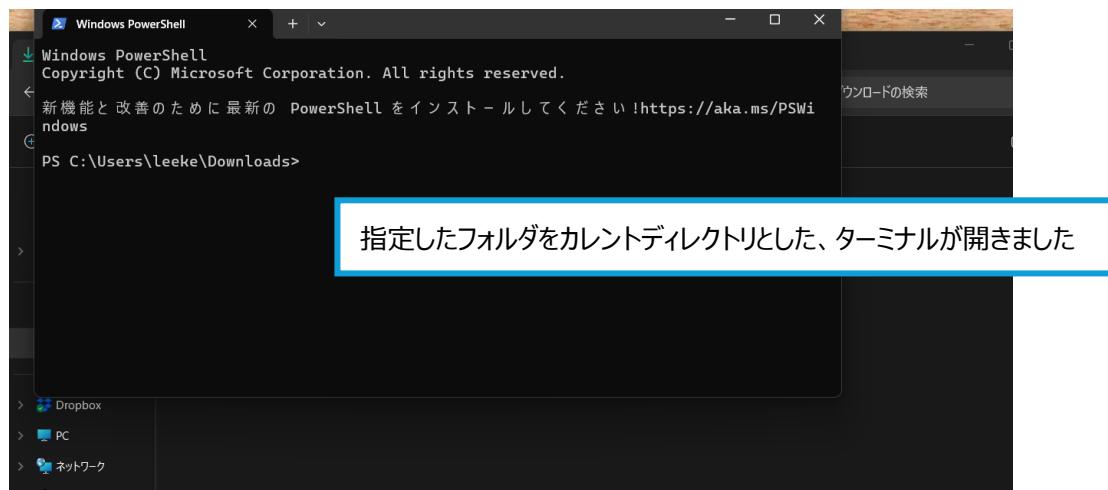
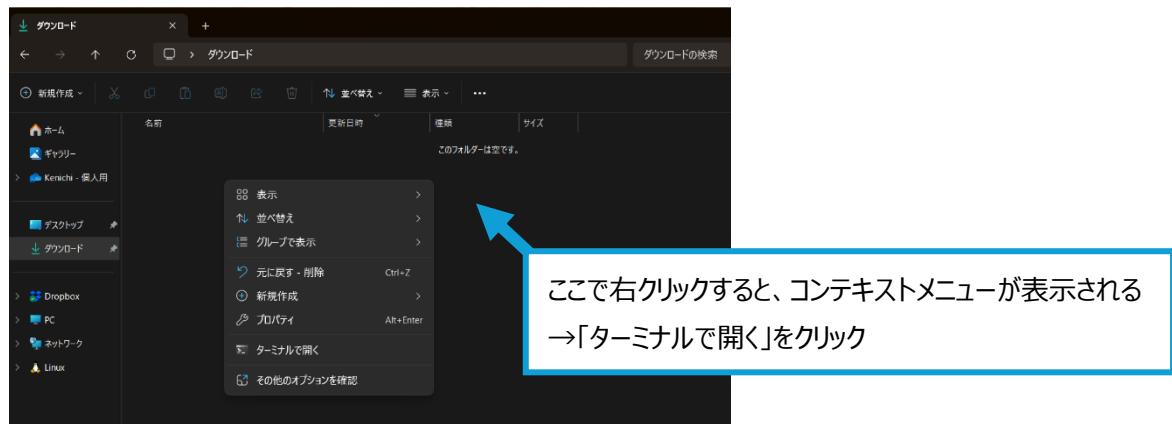
<https://ichiri.biz/tech/linux-directory-beginners/>

1.4.2. 指定したディレクトリでコマンド入力アプリを開く

まずは、作業を行いたいフォルダで以下の操作を行い、同フォルダをカレントディレクトリとした、コマンド入力アプリを開きましょう。

1.4.2.1. Windows

作業を行いたいフォルダを開き、右クリックのコンテキストメニューからターミナルを開きます。

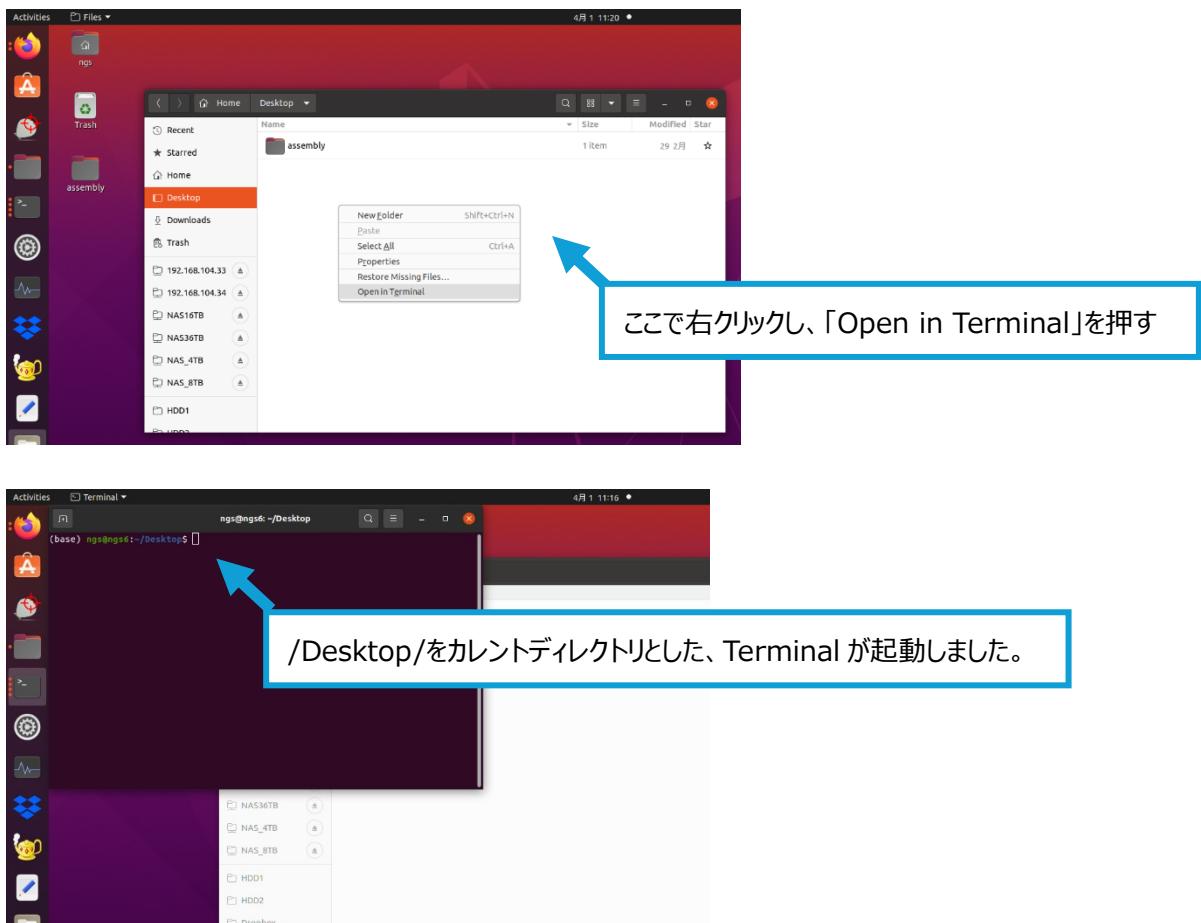


1.4.2.2. Mac

上述の Windows と同様に、Finder 上のコンテキストメニューから「フォルダに新規ターミナル」で、表示されているフォルダをカレントディレクトリとしたターミナルが開きます。

1.4.2.3. Linux

上述の Windows と同様に、フォルダ上のコンテキストメニューから「Open in Terminal」で、表示されているフォルダをカレントディレクトリとしたターミナルが開きます。



1.4.3. カレントディレクトリの変更

必要に応じてカレントディレクトリを変更しましょう。ディレクトリの移動は、Windows, Mac, Linux のいずれの場合も「cd」コマンドで行えます。

1.4.3.1. Windows

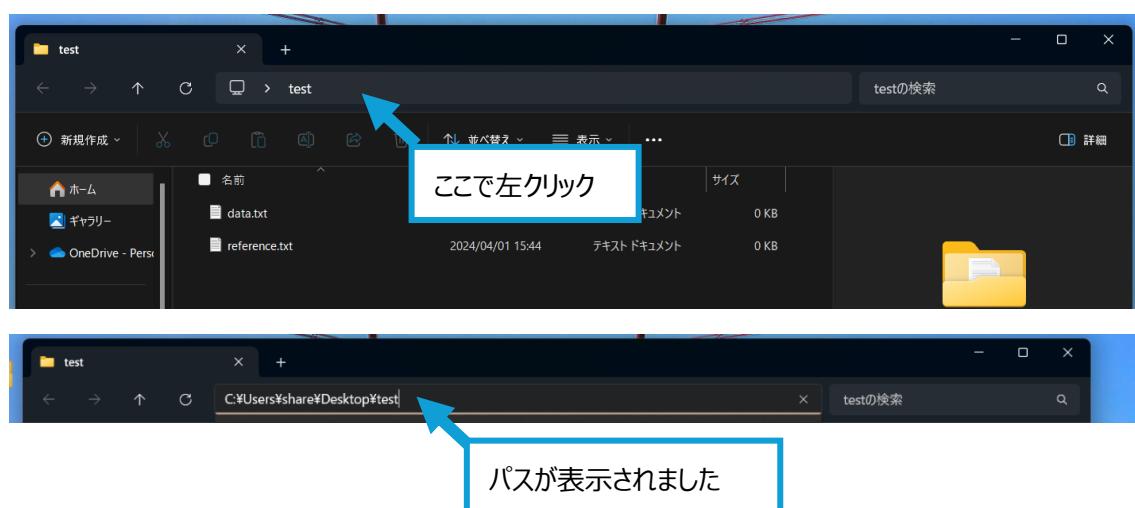
「cd」コマンドでディレクトリ変更ができます。絶対パスは、エクスプローラーのアドレスバーから確認できます。ただし、コマンド入力アプリを使用する際には、以下の点に注意する必要があります。

- ・アドレスバーの「¥(円)」または「¥(バックスラッシュ)」を、「/(スラッシュ)」に変える必要があります。
- ・別のドライブに移動する場合には、「mnt/移動先ドライブのアルファベット/」を入力する必要があります。

例えば、C ドライブの「C:¥Users¥share」から、D ドライブの「D:¥Genomedata¥2024-04-01_analysis」へカレントディレクトリを変更する場合には、以下のように入力します。

```
$ cd mnt/d/Genomedata/2024-04-01_analysis
```

※ドライブのパスは環境によって、多少異なる可能性があります。ご使用の環境に合わせてコマンドを修正してください。

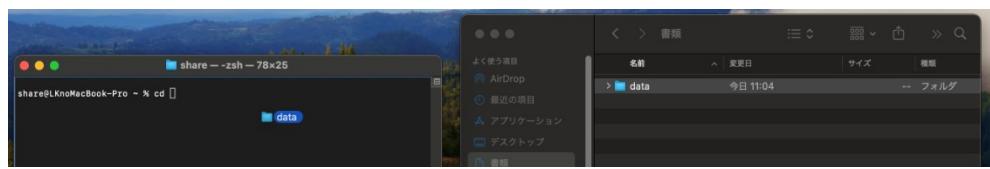


1.4.3.2. Mac

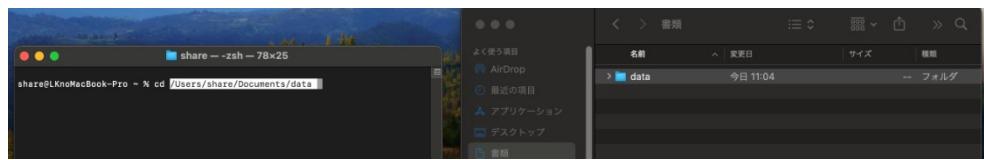
Mac および Linux はほぼ同様の方法で、カレントディレクトリの変更が可能です。主な 2 通りの方法を記載します。

- ・ドラッグアンドドロップ

対象フォルダをターミナル上にドラッグアンドドロップすることで、絶対パスが入力されます。



該当するフォルダをターミナルにドラッグアンドドロップします。



フォルダの絶対パスが入力されました。

・Finder でファイルパスを表示する方法や、フルパスをコピーする方法は、下記ウェブサイトに参考にしてください。

<https://tcd-theme.com/2020/11/mac-file-path.html>

1.4.3.3. Linux

Mac の場合と同様、主な 2 通りの方法を記載します。

- ・ドラッグアンドドロップ

上述の Mac の場合と同様に、対象フォルダをターミナルにドラッグアンドドロップすることで、絶対パスが入力されます。

- ・パスの表示方法

Ubuntu のデフォルトのファイルマネージャー(Windows でのエクスプローラーに相当するソフト)では、以下の方法でパスを表示させることができます。



※高機能なファイルマネージャー(Nemo 等、追加インストールが必要)では、アドレスバーに常にパスを表示させることもできます。

2. Docker のインストール

SNPcaster は Docker 上で動作する前提で作製しております。

この章では、Docker のインストール方法を説明します。

Docker のインストールに使用される最も有名なツールは Docker Desktop ですが、ご所属の組織規模によっては有償となります。

そのため、無償で利用できる Rancher Desktop のご利用をおすすめしております。

また、コマンドでのインストールとなります。Docker のコア機能である Docker Engine も無償で使用可能ですが(いずれも、2024 年 12 月時点の情報です)。

以上のことから、以下のような基準で Docker のインストール方法をお選びください。

いずれか 1 つの方法でインストールすれば SNPcaster がご利用いただけます。

- Windows もしくは Mac で簡単かつ無償で Docker をインストールしたい方
 - [Rancher Desktop のインストール](#)へお進みください。
- Linux ユーザー、または、コマンドの扱いになれている。もしくは、Rancher Desktop のインストールがうまくいかなかった方
 - [Docker Engine のインストール](#)へお進みください。
- Docker Desktop を使いたい方(Docker Desktop の有償の対象外もしくはすでにライセンス契約している)
 - [Docker Desktop のインストール方法](#)にインストール方法の記載があります。

2.1. Rancher Desktop のインストール

Rancher Desktop は、Docker のコマンドライン等の機能と Docker イメージやコンテナを管理できる画面を提供するアプリケーションです。本項では、Rancher Desktop のインストール方法を記載します。

※Docker の仕組み等については、[9.1. Dockerについて](#)をご参考ください。

※Linux 用のインストーラーもありますが SNPcaster の動作は未検証です。Linux ユーザーの方は、[Docker Engine のインストール](#)をご利用ください。

※ご注意※

Rancher desktop は Docker Desktop と同時起動すると正常に動作しません。 Rancher desktop を使用する場合は、[Docker Desktop のアンインストール方法](#)の内容を実施して DockerDesktop をアンインストールしてください。

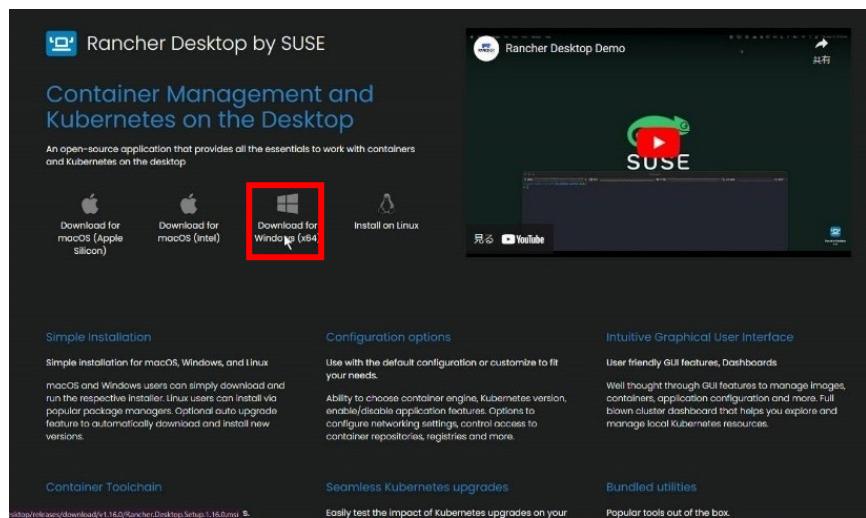
2.1.1. Windows

2.1.1.1. WSL の設定が完了していない場合は、Windows Subsystem for Linux (WSL) のインストールを見ながら WSL の設定を完了させてください。

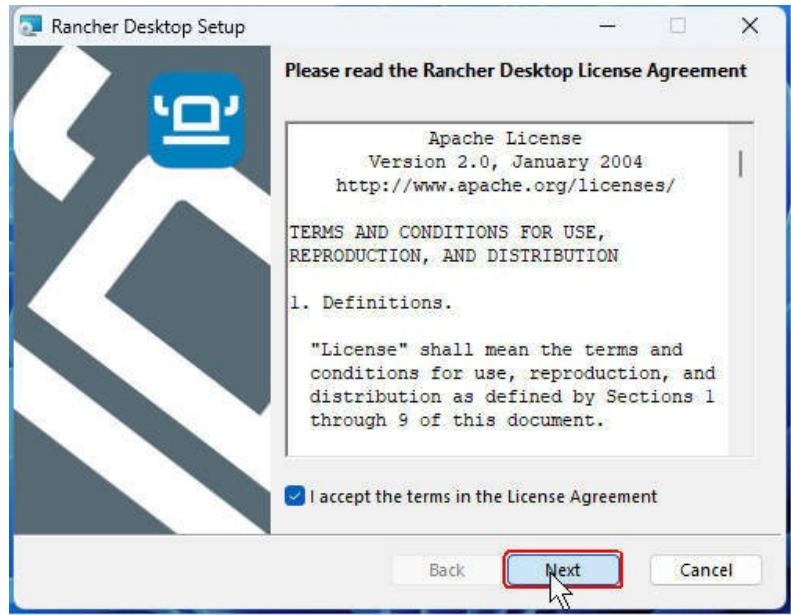
※(補足) 本来、Rancher Desktop はインストール時に WSL の設定を自動で実施しますが、検証したところ WSL の設定がされていないとインストールに失敗するケースが多かったため、このステップを追加しています。

2.1.1.2. 下記サイトから、Rancher Desktop をダウンロードします。下記サイトの Windows のリンクをクリックすると自動的にインストーラーがダウンロードされます。

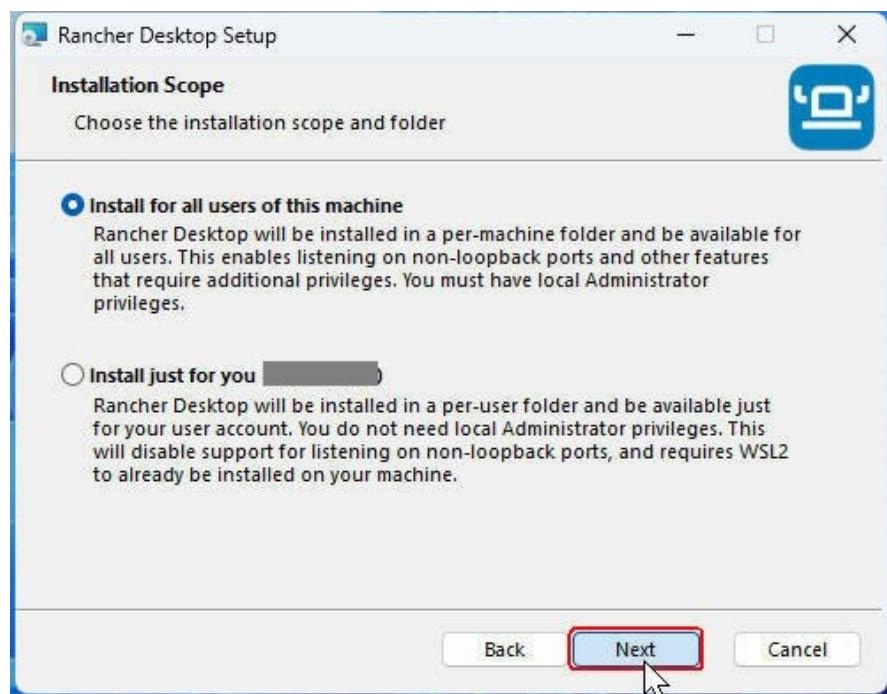
URL:<https://rancherdesktop.io/>



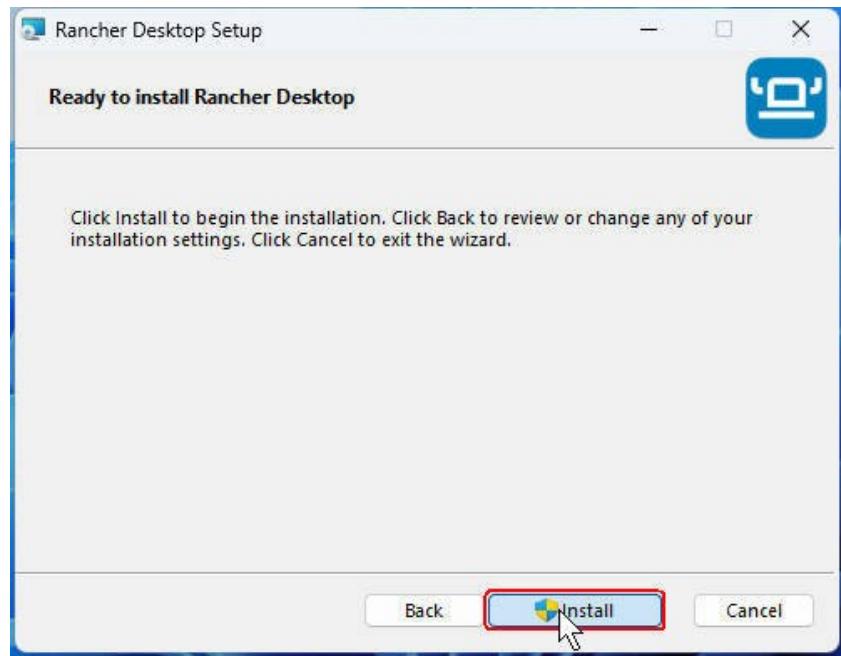
- 2.1.1.3. ダウンロードしたインストーラーをダブルクリックすると、ライセンス同意画面が表示されます。内容を確認した上で、I accept… のチェックを入れて Next をクリックします。



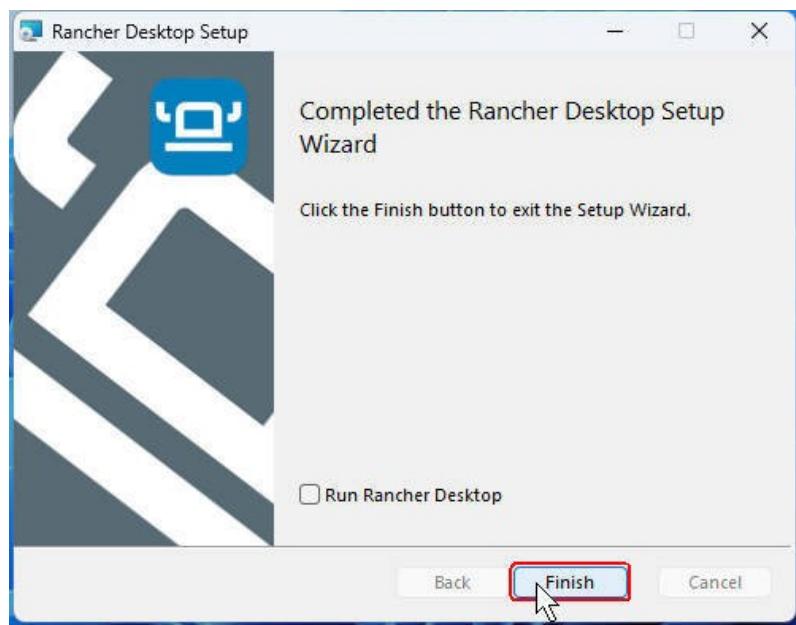
- 2.1.1.4. インストールする対象をマシン中の全ユーザーとするか、今ログインしているユーザーのみとするか選択します。どちらをお選びいただいても問題ありません。選択したら、Next をクリックします。



- 2.1.1.5. インストール開始の確認が表示されるので、Install をクリックします。
「このアプリがデバイスに変更を加えることを許可しますか？」という画面が出たら、「はい」を選択してください。



インストールが正常に完了すると完了画面が表示されるので、Finish をクリックします。



- 2.1.1.6. デスクトップ上に Rancher Desktop のアイコンが作成されるので、ダブルクリックして起動します。

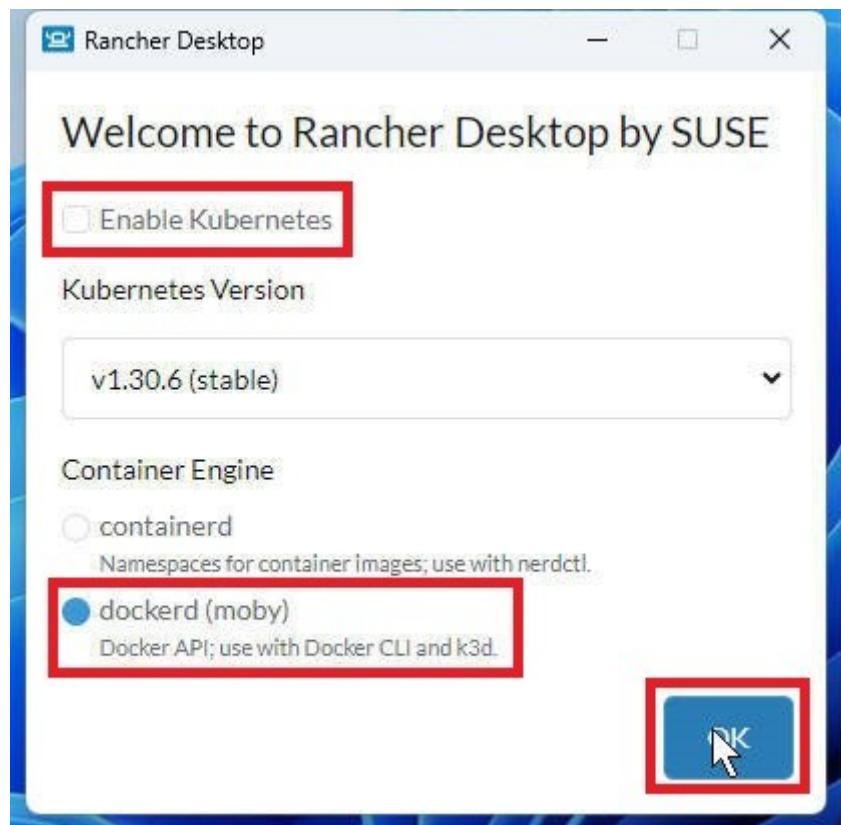


- 2.1.1.7. 初回設定画面が表示されます。

Enable Kubernetes のチェックを外してください。

Container Engine の **dockerd(moby)** を選び、右下の OK をクリックします。

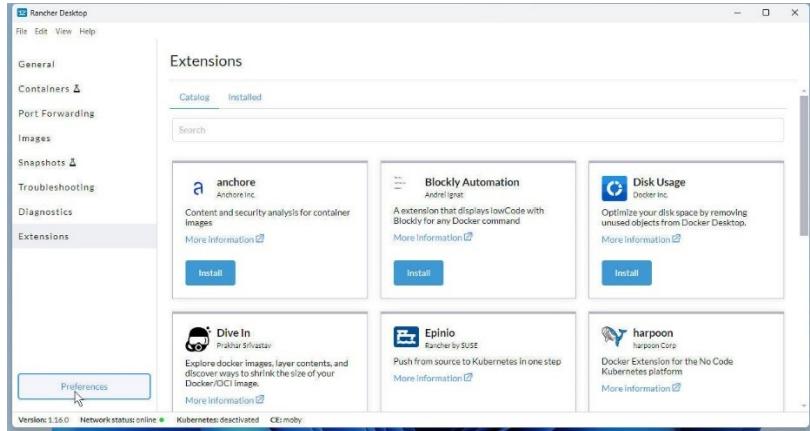
※(補足) Enable Kubernetes の設定は ON でも SNPcaster の実行が可能ですが、検証した範囲では、Rancher Desktop の起動に失敗することが多かったため、OFF を推奨しております。



2.1.1.8. (プロキシ設定が必要な方のみ) プロキシの設定を行います。

※不要な方は、[Rancher Desktop の起動設定](#)に進んでください※

画面左下の Preferences をクリックします。



左側パネルの WSL を選択し、画面中央の Proxy タブを選択すると、プロキシ設定画面が表示されます。

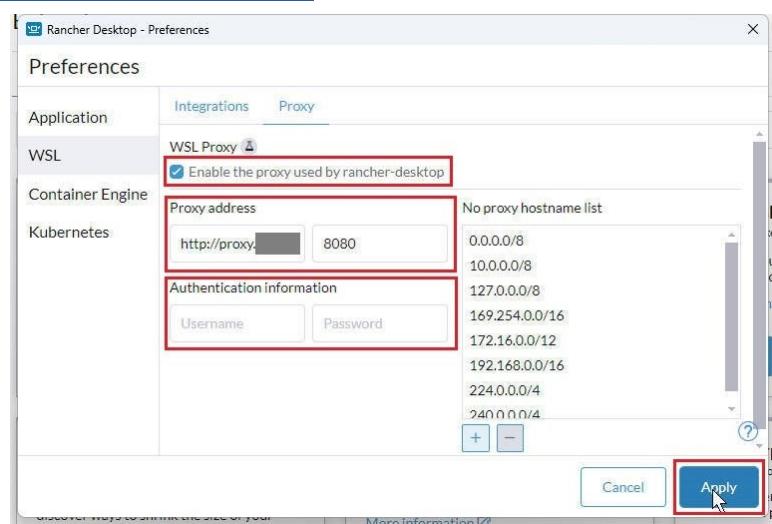
「Enable the proxy used by rancher-desktop」にチェックを入れます。

その下の「Proxy address」の左側のテキストボックスにプロキシサーバーの URL を、右側にポート番号を入力します。

その下の「Authentication Information」にはプロキシサーバー利用時の認証情報(ユーザー名とパスワード)を入力します。※認証がない場合は、空欄のままにしてください。

右下の Apply を押したら、プロキシ設定は完了です。

[Rancher Desktop の起動設定](#)に進んでください。



2.1.2. Mac

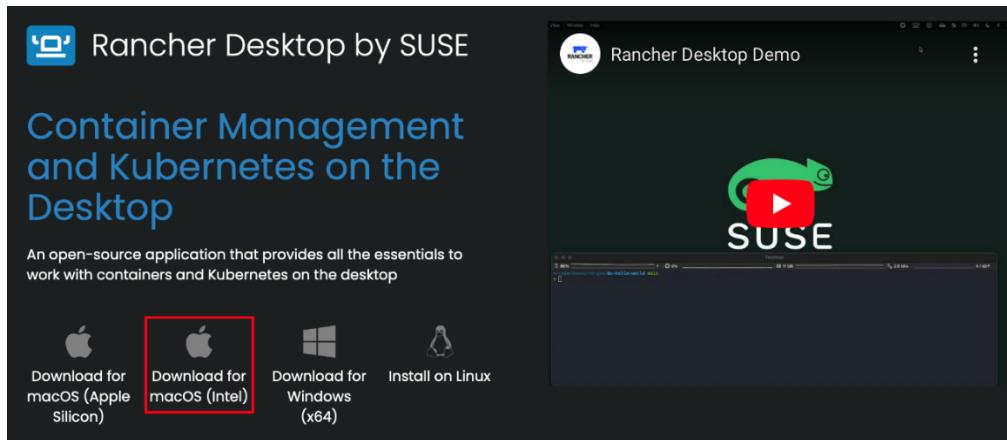
- 2.1.2.1. デスクトップ画面上部のアップルマーク「この mac について」を順番にクリックし、チップの種類が intel か Apple (M1、M2 等)かを確認します。



- 2.1.2.2. 下記サイトから、Rancher Desktop をダウンロードします。

前項で確認したチップに応じて、下記サイトの「Download for mac OS (Apple Silicon)」もしくは「Download for mac OS (Intel)」をクリックすると自動的にインストーラーがダウンロードされます。※画像の□は Intel チップの場合。Apple チップの場合はその左を選択する。

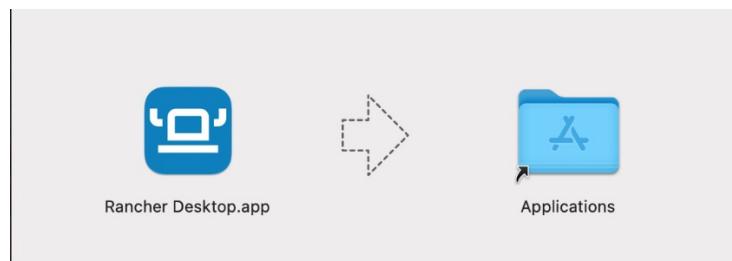
URL:<https://rancherdesktop.io/>



2.1.2.3. ダウンロードしたインストーラーをダブルクリックし、メッセージに従いインストールします。

Rancher Desktop アイコンを Applications アイコンへドラッグ&ドロップします。

特権的なアクセスが必要というメッセージが表示された場合、"OK"をクリックして、特権 アクセスを許可して下さい。

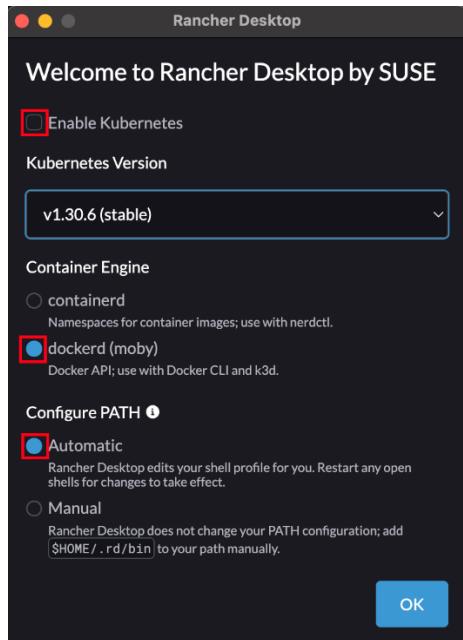


2.1.2.4. 表示されるメッセージに従い、インストールを完了させてください。インストール後、Rancher Desktop を起動させると、初回設定画面が表示されます。

Enable Kubernetes のチェックを外してください。

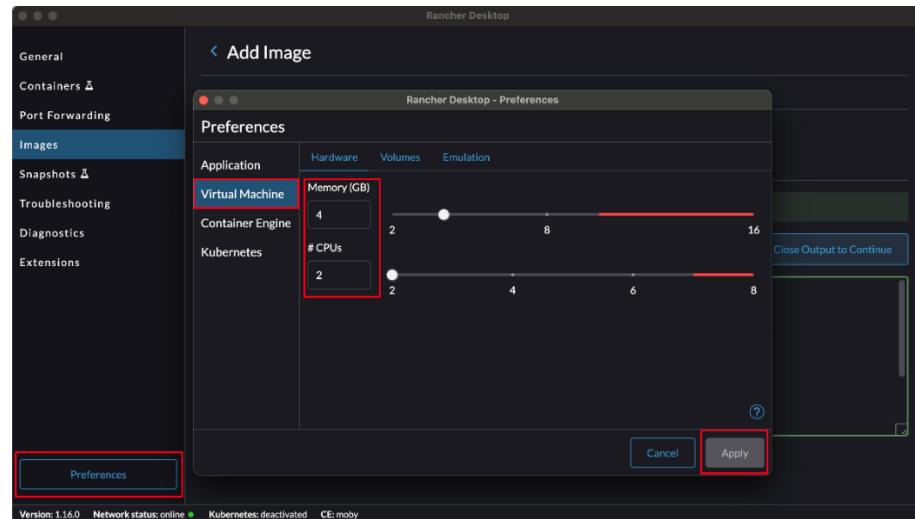
Container Engine の **dockerd(moby)** を選び、Configure PATH の **Automatic** を選んで右下の OK をクリックします。

※Enable Kubernetes の設定は ON でも SNPcaster の実行が可能ですが、検証した範囲では、Rancher Desktop の起動に失敗することが多かったため、OFF を推奨しております。



2.1.2.5. Rancher Desktop が使用できるメモリと CPU を設定します。

Preferences > Virtual Machine>Hardware を選び、**Memory** と **CPUs** の値を調整して **Apply** をクリックしてください。必要なスペックは、[解析端末の必要スペック](#) をご覧ください。



2.1.2.6. (プロキシ設定が必要な方のみ) プロキシの設定を行います。

不要な方は [Rancher Desktop の起動設定](#)に進んでください。

ターミナルを開き(開き方は、[Mac](#)をご覧ください)、以下のコマンドを入力し実行します。

```
LIMA_HOME="$HOME/Library/Application Support/rancher-
desktop/lima" "/Applications/Rancher
Desktop.app/Contents/Resources/resources/darwin/lima/bi
n/limactl" shell 0
```

lima-rancher-desktop(Rancher Desktop の仮想環境)にログインした状態となるので、以下のコマンドを実行します。

```
sudo vi /etc/init.d/docker
```

Vi エディタが実行されるので、最後の行に以下のプロキシ設定を追記します。

```
export http_proxy=<プロキシサーバーURL>:<ポート番号>/
export https_proxy=<プロキシサーバーURL>:<ポート番号>/
```

以下のコマンドで docker を再起動させたらプロキシ設定の完了です。

```
sudo service docker restart
```

ターミナルを閉じて、[Rancher Desktop の起動設定](#)に進んでください。

2.1.3. Rancher Desktop の起動設定

Rancher Desktop の起動に関する設定を行います。

Rancher Desktop はデフォルトでは PC を再起動した後に自動起動しません。

本項の手順を実施することで、PC 起動時に自動で Rancher Desktop が起動し、それに伴い Docker が使用できるようになります。

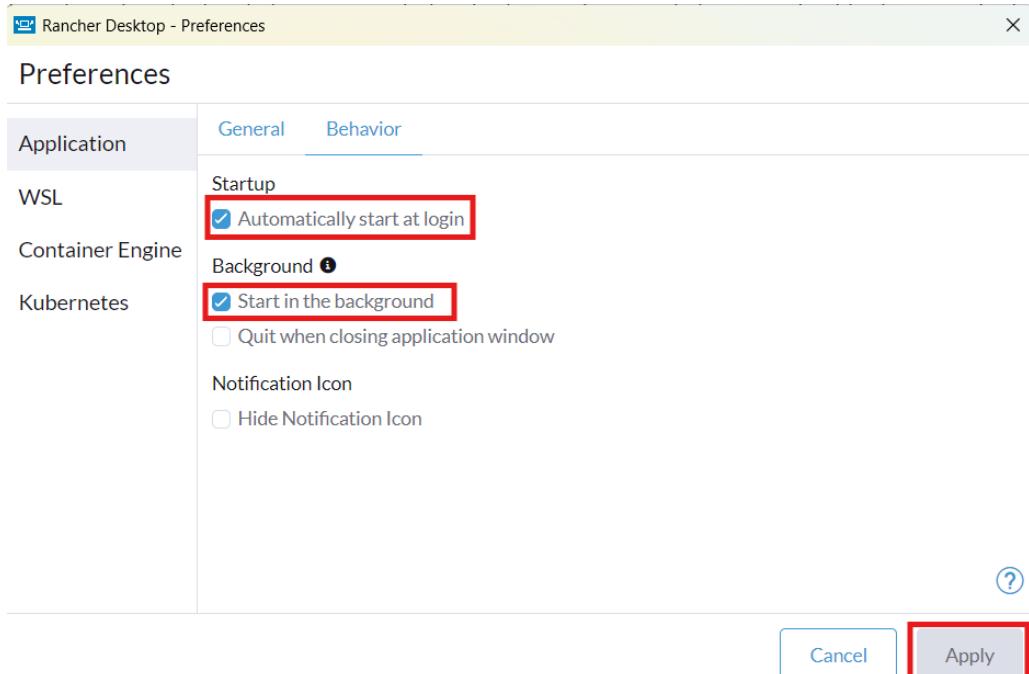
なお、本項では、Windows の画面を例示しますが、Windows/Mac 共通の設定内容です。

2.1.3.1. Rancher Desktop の設定を変更します。

メイン画面左下の、Preferences > Application > Behavior タブを開きます。

Automatically start at login と **Start in the background** にチェックを入れ、Applyをクリックすると、設定が反映されます。

※**Start in the background** は Rancher Desktop が自動起動した際に、Rancher Desktop の画面を表示しないようにするためのものです。必須ではないので、お好みで変更してください。



設定は以上です。これで、PC 起動時に Rancher Desktop が自動起動するようになりました。

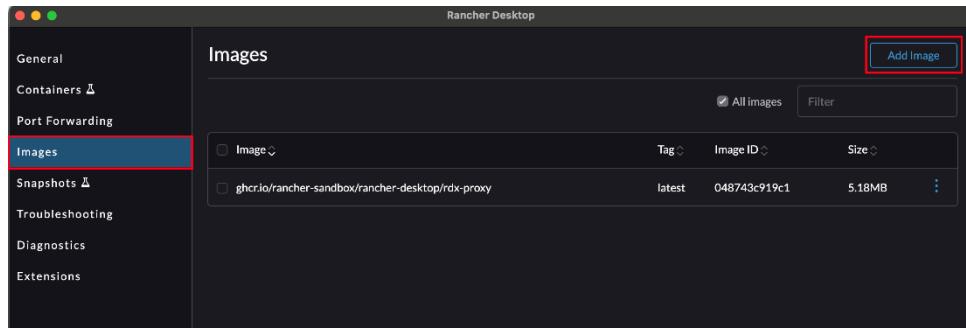
2.1.4. Rancher Desktop の動作確認

Rancher Desktop が正常に動作するか、確認していきます。

画面は Mac のものですが、Windows も同じ手順です。

2.1.4.1. Rancher Desktop の動作チェックを行います。

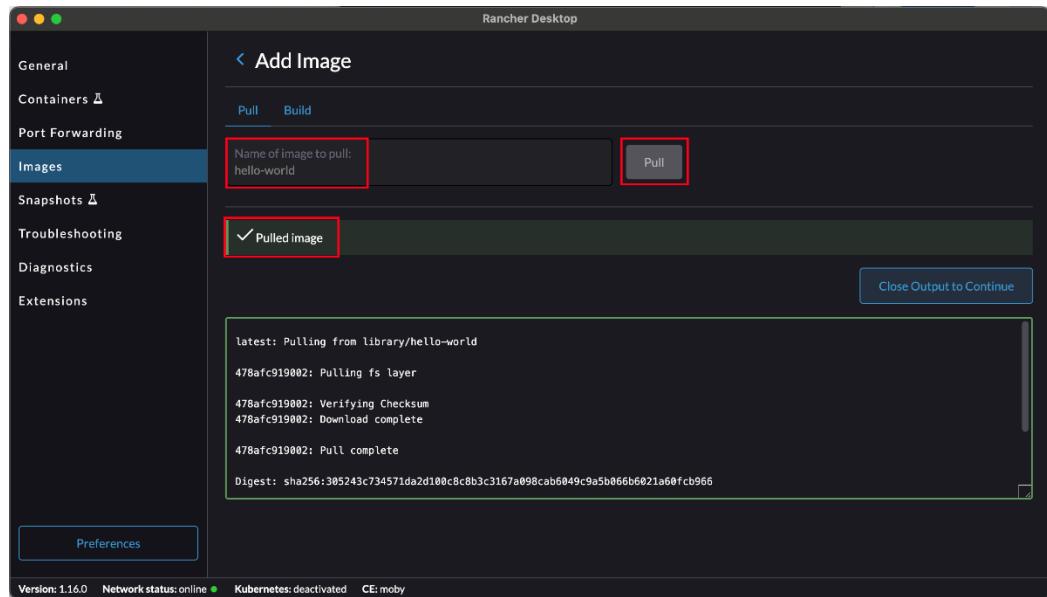
メイン画面から、Images>Add Image を選択します。



Name of image to pull: の箇所に **hello-world** と入力し、Pull をクリックします。

Pulled image と表示されれば正常に動作しています。

※エラーが出る場合、プロキシ設定の問題である場合が多いです。前のステップのプロキシ設定を見直してください。



これで Rancher desktop を利用した Docker の準備が完了しました。

[SNPCaster の環境構築へ進んでください。](#)

2.2. Docker Engine のインストール

Docker Engine はコマンドを使ってインストール可能です。

Mac をご利用の方は、[Mac](#)へお進みください。

Linux をご利用の方は、[Linux](#)へお進みください。

2.2.1. Windows

この項では、Windows 環境への Docker Engine のインストールのための準備を行います。

2.2.1.1. WSL の設定が完了していない場合は、Windows Subsystem for Linux (WSL) のインストールを見ながら WSL の設定を完了させてください。

2.2.1.2. 以下のコマンドを PowerShell で実行し、WSL を起動します。

```
wsl -d Ubuntu-24.04
```

```
PS C:\Users\[\REDACTED]\> wsl -d Ubuntu-24.04  
test@[\REDACTED]:/mnt/c/Users/[\REDACTED] $ |
```

WSL (Ubuntu) が起動すると、左側の表示が変わります。

2.2.1.3. SNPcaster フォルダを配置したフォルダに移動します。

以下のコマンドで移動してください。WSL 上でのパスがわからない場合は、[Windows のカレントディレクトリの変更](#)をご確認ください。

```
cd /mnt/{Windows 上の SNPcaster をダウンロードしたフォルダパス}
```

2.2.1.4. (プロキシ設定が必要な方のみ) 不要な方は、次のステップに進んでください。

プロキシ情報を記載した.env ファイルを作成します。

[.env ファイルの編集](#)の作業を実施してください。

.env ファイルを作成後、以下のコマンドを.env ファイルと同じフォルダ中で実行するとプロキシ設定が環境変数に登録されます。

```
bash ./preparation/docker-engine/set_proxy.sh
```

実行時、パスワード入力を求められたら入力してください。

Done!と表示されたら完了です。

`set_proxy.sh` 実行直後は追加した環境変数が読み込まれていないため、以下のコマンドを実行して更新します。

```
source ~/bashrc
```

これで、`docker engine` インストール時に必要なプロキシ設定は完了です。

- 2.2.1.5. `docker engine` のインストールを行います。以下のコマンドを実行してください。

```
bash ./preparation/docker-engine/install_docker_engine.sh
```

パスワード入力が求められるので、入力します(下図の□、パスワードは表示されません)。

```
[sudo] password for test: [REDACTED]
```

Done!と表示されたら Docker Engine のインストールは完了です。

- 2.2.1.6. 一度ターミナルを閉じて、再度開きます。

以下のコマンドで、ターミナルからログアウトします。

```
exit
```

再ログインを行います。[2.2.1.2](#) で実施した手順と同一です。

以上で、Docker Engine のインストールは完了です。

以降、SNPcaster を利用したい場合は、ターミナルを起動して以下のコマンドを実行すると、`docker` コマンドが利用可能になります。

```
$ wsl.exe -d Ubuntu-24.04
```

[Docker Engine の動作確認](#)に進んで、Docker の動作確認をしてください。

2.2.2. Mac

2.2.2.1. (プロキシ設定が必要な方のみ) 不要な方は、次のステップに進んでください。

ターミナルにプロキシ設定を登録します。

[ターミナルを開き](#)、下記のコマンドを実行します。

※<プロキシサーバーURL>:<ポート番号>はご利用の環境に置き換えてください。

例) http://proxy.example.com:8080

```
echo 'export http_proxy=<プロキシサーバーURL>:<ポート番号>' >> ~/.zshrc
echo 'export https_proxy=<プロキシサーバーURL>:<ポート番号>' >> ~/.zshrc
echo 'export HTTP_PROXY=<プロキシサーバーURL>:<ポート番号>' >> ~/.zshrc
echo 'export HTTPS_PROXY=<プロキシサーバーURL>:<ポート番号>' >> ~/.zshrc
```

設定した内容を読み込むために、以下のコマンドを実行します。

```
source ~/.zshrc
```

これで、ターミナルへのプロキシ設定は完了しました。

2.2.2.2. Mac 向けパッケージマネージャーである、Homebrew をインストールします。

以下のコマンドを実行します。

```
cd ~
mkdir .homebrew
curl -L https://github.com/Homebrew/brew/tarball/master | tar xz --strip 1 -
C .homebrew
```

Homebrew に対してパスを通すために、以下のコマンドを実行します。

```
echo 'export PATH=$HOME/.homebrew/bin:$PATH' >> ~/.zshrc
echo 'export HOMEBREW_CACHE=$HOME/.homebrew/caches' >> ~/.zshrc
source ~/.zshrc
```

2.2.2.3. Docker CLI client をインストールします。

以下のコマンドを実行します。

```
brew install docker docker-compose
```

以下のような表示が出てきます。これは、この後の設定に使用しますので記録しておいてください。

```
Compose is a Docker plugin. For Docker to find the plugin, add
"cliPluginsExtraDirs" to ~/.docker/config.json:
"cliPluginsExtraDirs": [
    "/opt/homebrew/lib/docker/cli-plugins"
]
```

※**/opt/homebrew** の箇所は、**/Users/{Mac ユーザー名}/.homebrew** になっている可能性があります。

続いて、docker CLI の設定を行います。

お好きなテキストエディタで、**/Users/{Mac ユーザー名}/.docker/config.json** を編集します。

以下 2 点の作業を実施します。

- **credsStore** の行を削除してください。
- docker-compose インストール時に表示された **cliPluginsExtraDirs** の内容を末尾に追加します。
その際、直前の行の末尾にカンマ(,)を追加してください(直前の行末にすでにカンマがある場合は追加不要です)。

```
Users > share > .docker > {} config.json > [ ] cliPluginsExtraDirs
1   {
2     "auths": {},
3     "credsStore": "osxkeychain", ← 行を削除
4     "currentContext": "lima-snpcaster-vm", ← 追加
5     "cliPluginsExtraDirs": [] ← 追加
6       "/Users/share/.homebrew/lib/docker/cli-plugins"
7
8 }
```

上記の編集が終わったら、保存します。

2.2.2.4. Docker Server を起動する環境として、lima をインストールします。

```
brew install lima
```

下記のようなエラーが表示された場合、メッセージに記載されている通り、追加でインストールが必要です。

```
[share@wf-toyama242-10 ~ % brew install lima  
Error: No developer tools installed.  
Install the Command Line Tools:  
xcode-select --install
```

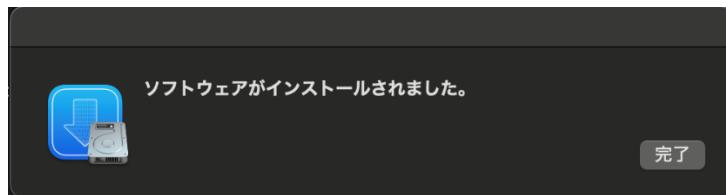
以下のコマンドを実行します。

```
xcode-select --install
```

さらに、**xcode-select: note: install requested for command line developer** というエラーが表示された場合、以下のようなインストールアプリが起動するので、インストールします。



インストールが完了すると、以下の画面が表示されるので完了をクリックします。



もう一度、下記のコマンドを実行して lima をインストールします。

```
brew install lima
```

2.2.2.5. lima 上に Docker Server を起動する仮想マシンを作成します。

以下のコマンドを実行します。

※--cpus=は割り当てるCPUコア数、--memory=は割り当てるメモリ量(GB)です。お使いのマシンの性能に合わせて、この値は修正してください。
必要スペックは、[解析端末の必要スペック](#)をご確認ください。

```
limactl create --name=snpcaster-vm \
    --cpus=6 --memory=14 \
    --vm-type=vz --mount-type=virtiofs --rosetta \
    template://docker
```

以下のように確認が表示されるので、**Proceed with the current configuration** が選択されている状態で Enter を押します。

```
[share@wf-toyama242-10 ~ % limactl create --name=snpcaster-vm \
    --cpus=6 --memory=14 \
    --vm-type=vz --mount-type=virtiofs --rosetta \
    template://docker
? Creating an instance "snpcaster-vm" [Use arrows to move, type to filter]
> Proceed with the current configuration
Open an editor to review or modify the current configuration
Choose another template (docker, podman, archlinux, fedora, ...)
Exit
```

以下のコマンドを実行し、NAME列が **snpcaster-vm** になっている行があれば作成完了です。

```
limactl list
```

続いて、lima の設定を行います。

/Users/{Mac ユーザー名}/.docker/config.json をお好きなエディタで編集します。

mounts:と記載された直後にある- **location: “~”**のすぐ下に **writable: true** を追記します。必ず、半角スペース 2つを行頭に入れてください。

```
24
25   mounts:
26   - location: "~"
27     | writable: true
28   - location: "/tmp/lima"
29     | writable: true
```

保存したら、設定完了です。

なお、上記の設定は、**home** フォルダ(/Users/{Mac ユーザー名}/)以下を

docker が書き込み可能とするための設定です。

そのため、SNPcaster フォルダを配置・実行する場所も/**Users/{Mac ユーザー名}/以下にしてください** (SNPcaster が実行結果を出力できず、エラーが出ます)。

2.2.2.6. 作成した仮想マシンを起動します。以下のコマンドを実行します。

```
limactl start snpcaster-vm
```

※以降、PC を再起動するたびに上記コマンドで VM を起動する必要があります。

2.2.2.7. Docker CLI を前のステップで起動した Docker Server に接続します。

以下のコマンドを実行します。

※以下のコマンドは前のステップで仮想マシンを起動したときに画面に表示されますので、そちらをコピーしてお使いください。

```
docker context create lima-snpcaster-vm --docker  
"host=unix:///Users/{Mac ユーザー名}/.lima/snpcaster-  
vm/sock/docker.sock"  
docker context use lima-snpcaster-vm
```

以上で、Docker Engine のインストールが完了しました。

今後は、ターミナルから docker コマンドが利用可能です。

Docker Engine の動作確認に進んで、Docker の動作確認をしてください。

2.2.3. Linux

Docker CLI、Docker Compose をインストールします。
実行は、root または sudo 権限のあるユーザーで行ってください。

2.2.3.1. ターミナルを開き、以下のコマンドで Docker に必要なパッケージをインストールします。

(root ユーザーの場合、sudo は不要です。)

- Ubuntu

```
sudo apt install -y \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
```

- CentOS

```
sudo yum install -y \ yum-utils \ device-mapper- \
    persistent-data \ lvm2
```

2.2.3.2. 以下のコマンドで、Docker 公式サイトからインストールスクリプト「get-docker.sh」を取得し、実行します。

- Ubuntu・CentOS

```
curl -fsSL get.docker.com -o get-docker.sh
sh get-docker.sh
```

※curl が動作しない場合、プロキシ設定をすることで解決する場合があります。

```
nano ~/.curlrc
```

下記のように内容を編集します。(i を押すと、編集モードになります)
(**サーバーIP およびポート番号は、実際の環境に合わせてください。**)

下記はプロキシ IP アドレスが proxy.go.jp、ポート番号が 8080 の場合

```
proxy=http://proxy.go.jp:8080
```

上記のように記載したら、保存します。

(「Ctrl+X」で終了→保存するか聞かれたら「Y」を入力すると、保存される)

2.2.3.3. 以下のコマンドで、サーバ・PC の再起動時に Docker を常に起動する設定へ変更します。

(root ユーザーの場合、sudo は不要です。)

- Ubuntu・CentOS

```
sudo systemctl enable docker  
sudo systemctl start docker
```

2.2.3.4. Docker コマンドを sudo なしで実行するため、以下のコマンドでユーザーを docker グループへ所属させます。

- Ubuntu・CentOS

```
sudo gpasswd -a $(whoami) docker
```

現在のログインユーザーとは別のユーザーで docker を使用する場合

\$(whoami)の部分を、そのユーザー名に置き換えてください。

```
sudo gpasswd -a $(whoami) docker
```

2.2.3.5. 設定を再読み込みするため、一度ターミナルを閉じて、再度開きます。

以下のコマンドを実行し、バージョン番号が表示されたら OK です。

次のステップに進んでください。

```
docker compose version
```

```
(base) :~$ docker compose version  
Docker Compose version v2.35.1
```

2.2.3.6. (必要な方のみ) プロキシサーバの設定を以下の手順で行います。

※不要な方は [Docker Engine の動作確認へ進んでください](#)※

※こちらの設定で解決しない場合は、組織の IT 部門にお問い合わせください。

systemd の設定ファイルに対して、プロキシサーバーの設定を書き込みます。下記コマンドで設定ファイルを編集します。

- Ubuntu・CentOS

```
sudo systemctl edit docker
```

ターミナル中でエディターが起動するので、下記の内容を追記(既にある場合は修正)します。

(サーバーIP およびポート番号は、実際の環境に合わせてください。)

Anything between here and the comment below…contents of the drop-in file という行から
Edits below this comment will be discarded の
行の間に記載 してください。それ以外の行に記載しても反映されませんのでご注意ください。

追記内容(プロキシ IP アドレスが proxy.go.jp、ポート番号が 8080 の場合の例)

※追記するのは、**[Service]**から始まる 3 行(黄色塗りの箇所)です。

```
### Editing /etc/systemd/system/docker.service.d/override.conf
### Anything between here and the comment below will become the
contents of the drop-in file
[Service]
Environment="HTTP_PROXY=http://proxy.go.jp:8080"
Environment="HTTPS_PROXY=http://proxy.go.jp:8080"

### Edits below this comment will be discarded
```

完了したら、保存してエディターを閉じます。

(「Ctrl+X」で終了→保存するか聞かれたら「Y」を入力すると、保存される)

その後、以下のコマンドで Docker を再起動します。

- Ubuntu・CentOS

```
sudo systemctl daemon-reload  
sudo systemctl restart docker
```

次に、コンテナ内での通信をできるようにするために、Docker クライアントを設定します。

設定ファイルは、コンテナを起動するユーザーのホームディレクトリ以下の「`~/.docker/config.json`」です。

以下のコマンドで、`~/.docker` ファイルの作製と config.json の編集を行います。

```
mkdir -p ~/.docker  
nano ~/.docker/config.json
```

下記のフォーマットで内容を編集します。

(サーバーIP およびポート番号は、実際の環境に合わせてください。)

※下記はプロキシ IP アドレスが proxy.go.jp、ポート番号が 8080 の場合

```
{  
  "proxies": {  
    "default": {  
      "httpProxy": "http://proxy.go.jp:8080",  
      "httpsProxy": "http://proxy.go.jp:8080"  
    }  
  }  
}
```

上記のように記載したら、保存してエディターを閉じます。

(「`Ctrl+X`」で終了→保存するか聞かれたら「`Y`」を入力すると、保存される)

以上でプロキシ設定は完了です。[Docker Engine の動作確認へ進んでください。](#)

2.2.4. Docker Engine の動作確認

この章では、Docker Engine の動作確認を行います。Docker Engine をインストールした上で実施してください。

2.2.4.1. Docker Engine の動作確認用のコンテナを起動します。

docker コマンドが使用可能なターミナルを起動し、以下のコマンドを実行します。

```
docker run hello-world
```

以下の通り、Hello from Docker!と表示されれば正常に動作しています。

```
test@NGSW:/mnt/d/test/SNPcaster_docker_engine$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:305243c734571da2d100c8c8b3c3167a098cab6049c9a5b066b6021a60fc966
Status: Downloaded newer image for hello-world:latest

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

To generate this message, Docker took the following steps:
```

動作確認に使用したコンテナと Docker image は不要なので、以下のコマンドで削除します。

```
docker image rm -f hello-world:latest
```

以上で、動作確認は終了です。

[3.SNPcaster の環境構築](#)へ進んでください。

3. SNPcaster の環境構築

ここでは、Docker 上で動作する SNPcaster の環境構築を実施します。

3.1. SNPcaster のダウンロード

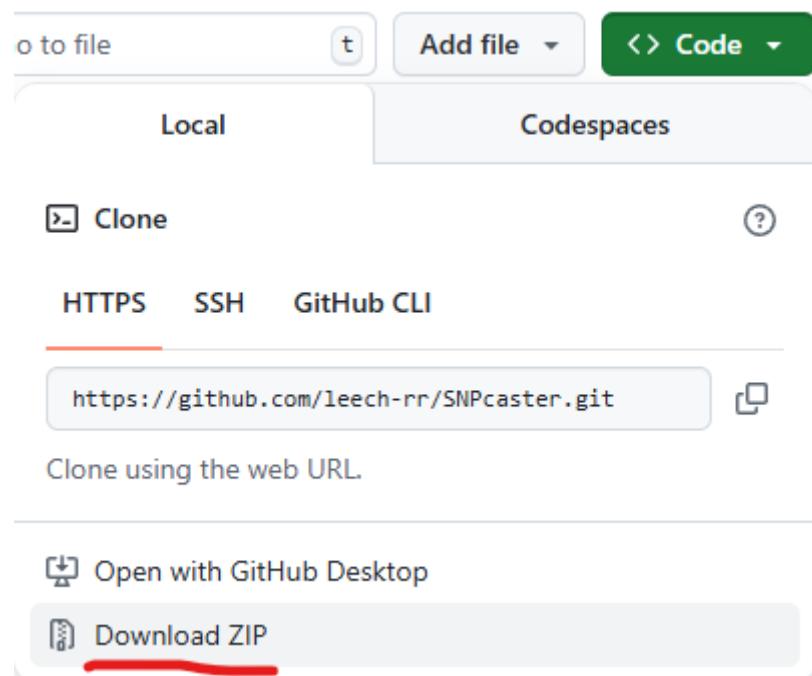
まずは GitHub 上で公開されている SNPcaster フォルダをダウンロードします。

ダウンロードは、ブラウザ経由(クリック操作)またはコマンドラインのいずれかで行ってください。

3.1.1. ブラウザ(Microsoft Edge, Google Chrome など)を使ってダウンロード

<https://github.com/leech-rr/SNPcaster/tree/main> を開きます。

右上の Code をクリックし、その後「Download Zip」をクリックするとダウンロードが開始します。



ダウンロードが完了したら、作業を行いたいフォルダに移動して解凍してください。

3.1.2. コマンドライン(Git)を使ってダウンロード

Git を使ってダウンロードする場合、今後のアップデート時に git pull するだけで更新ができるので便利です。一方、Git の操作には少し慣れが必要なので、時間がない場合はブラウザ(Microsoft Edge, Google Chrome など)を使ってダウンロードの方でダウンロードすることをお勧めします。

Git を使ってダウンロードする場合、事前に Git のインストールが必要です。インストールしていない場合は、以下の URL にインストール方法が記載されているので、Git をインストールしてください。

<https://git-scm.com/book/ja/v2/%E4%BD%BF%E3%81%84%E5%A7%8B%E3%82%81%E3%82%8B-Git%E3%81%AE%E3%82%A4%E3%83%B3%E3%82%B9%E3%83%88%E3%83%BC%E3%83%AB>

Windows PowerShell(Windows)、ターミナル(Mac)、端末(Linux)等を起動します。(これらのツールが分からぬ場合は、コマンド入力アプリについてをご覧ください)

ダウンロードする前に、以下のコマンドを実行してください。

※特に Windows で、以下の設定をしないとインストールに失敗することが確認されています。

```
git config core.autocrlf false
```

以下のコマンドを実行すると、git からダウンロードできます。

なお、{ダウンロードしたいフォルダ}の箇所は、ダウンロードしたいフォルダのパスを記載してください。

```
cd {ダウンロードしたいフォルダパス}  
git clone https://github.com/leech-rr/SNPcaster.git
```

3.2. (参考) フォルダ・ファイル構成

SNPcaster の構成は以下の通りです(ファイルは一部を除き省略しています)。

```
snpcaster
|   .env.example
|   .gitignore
|   COPYING
|   docker-compose.yml
|   NOTICE.md
|   README.md
|   SNPcaster_bugs.md
├── app
|   |   Dockerfile
|   ├── notebook
|   |   :
|   ├── setup
|   |   :
|   └── src
|       ├── grape_qc_assembly
|       |   :
|       └── snpcaster
|           :
└── doc
    :
├── preparation
    :
└── project
    └── .gitignore
```

構成については、以下の通りです。

フォルダ/ファイル名	内容
.env.example	環境設定に使う env ファイルのテンプレートです。
.gitignore	git 管理用ファイルです。編集しないでください。
COPYING	SNPcaster が GPLv3.0 であることを示すファイルです。
docker-compose.yml	Docker コンテナを起動するための設定ファイルです。
NOTICE.md	SNPcaster が利用している
README.md	プロジェクトの readme です。
SNPcaster_bugs.md	報告されたバグをまとめた一覧です。
app	SNPcaster の構築に必要なフォルダ・ファイルをまとめたフォルダです。
app/ Dockerfile	Docker イメージを構築する手順を書いたファイルです。
app/ notebook	Jupyter notebook を用いた解析に利用するプログラム等を格納したフォルダです。
app/ setup	Docker 環境の作成や起動時に使用するスクリプトをまとめたフォルダです。
app/ src	解析実行時に必要なプログラムファイル等を格納したフォルダです。
app/src/ grape_qc_assembly	grape 実行時に必要なプログラム等を格納したフォルダです。
app/src/ snpcaster	SNPcaster 実行時に必要なプログラム等を格納したフォルダです。
doc	ドキュメント類をまとめたフォルダです
preparation	Docker engine のインストールなど、Docker 自体を使用するために必要な事前準備を行うのに必要なプログラム等を配置するフォルダです。

project	解析を実行する場所となるフォルダです。解析結果等はこのフォルダに出力されます。 コンテナの起動後、このフォルダ直下に notebook フォルダが作成され、jupyter notebook の出力はそちらに配置されます。
project / .gitignore	git 管理用ファイルです。編集しないでください。

3.3. 解析端末への SNPcaster インストール

解析端末へ SNPcaster をインストールために、Docker イメージのビルドとそれを使ってコンテナを立ち上げます。

3.3.1. .env ファイルの編集

※このステップはプロキシ環境をご利用の方のみ作業し、不要な方はスキップしてください※

まず、ビルド時に使用する環境変数を設定するため、.env ファイルを作成します。

ダウンロードした SNPcaster フォルダをファイルエクスプローラー等で開きます。

.env.example をコピーし、コピーしたファイルの名前を.env に変更します。

名前	更新日時	種類	サイズ
.git	2024/11/20 16:21	ファイル フォルダー	
jupyter	2024/11/20 15:49	ファイル フォルダー	
.env	2024/11/20 16:14	EXAMPLE ファイル	1 KB
.env.example	2024/11/20 16:14	EXAMPLE ファイル	1 KB
.gitignore	2024/11/20 15:35	テキスト ドキュメント	4 KB
docker-compose.yml	2024/11/20 16:20	Yaml ソース ファイル	1 KB
SNPcaster_bugs.md	2024/11/20 15:22	Markdown ソース フ...	4 KB

環境変数の編集を行います。

先ほど作成した.env ファイルを、お好きなテキストエディタで開きます。

10~11 行目の”# “を削除し、以下に示すように設定を記載してください。

(サーバーURL およびポート番号は、実際の環境に合わせてください。)

```
new 6 env
1 LOCAL_UID=1000
2 LOCAL_GID=1000
3
4 # Proxy設定を行いたい場合、以下に設定してください。
5 # HTTPのプロキシURL-> HTTP_PROXYに記載(文頭の# を削除する)
6 # HTTPSのプロキシURL-> HTTPS_PROXYに記載(文頭の# を削除する)
7 # 書き方の例: HTTP_PROXY=http://proxy.example.com:8080
8 # *proxy.example.com:8080の箇所は{URL}:{port}の形式で環境に応じて書き換えて下さい
9 # *認証が必要な場合は、http://{認証id}:{認証パスワード}@{URL}:{port} の形式にして下さい。
10 HTTP_PROXY=http://proxy.example.com:8080
11 HTTPS_PROXY=http://proxy.example.com:8080
```

各行の# を削除する

HTTP_PROXY: HTTP での設定を{URL}:{port} の形式で同じ内容を記載

HTTPS_PROXY: HTTPS での設定を{URL}:{port} の形式で同じ内容を記載

※認証が必要な場合は、http://{認証 id}:{認証パスワード}@{URL}:{port} で記載

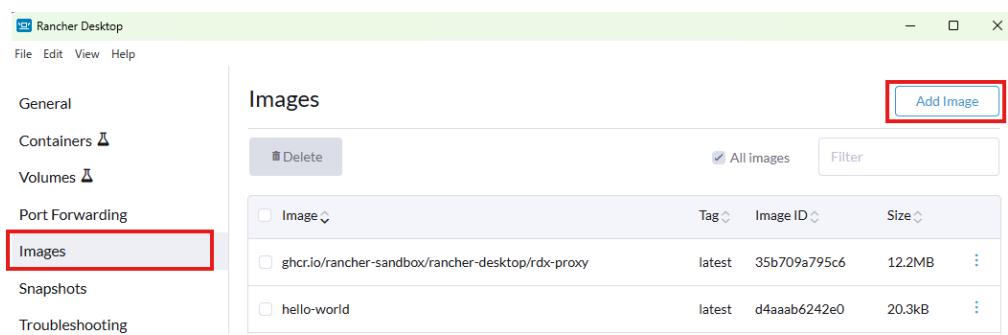
※APT_PIPELINING と APT_NO_CACHE は上級者向けで、設定不要です。プロキシ接続がうまくいかない場合に、.env.example の説明をご確認ください。

3.3.2. ビルドの実行

Windows で Rancher Desktop を使い、なおかつ、プロキシ設定が必要な方のみ、まずは以下の操作を行ってください。

※対象外の方は、次のページに進んでください。

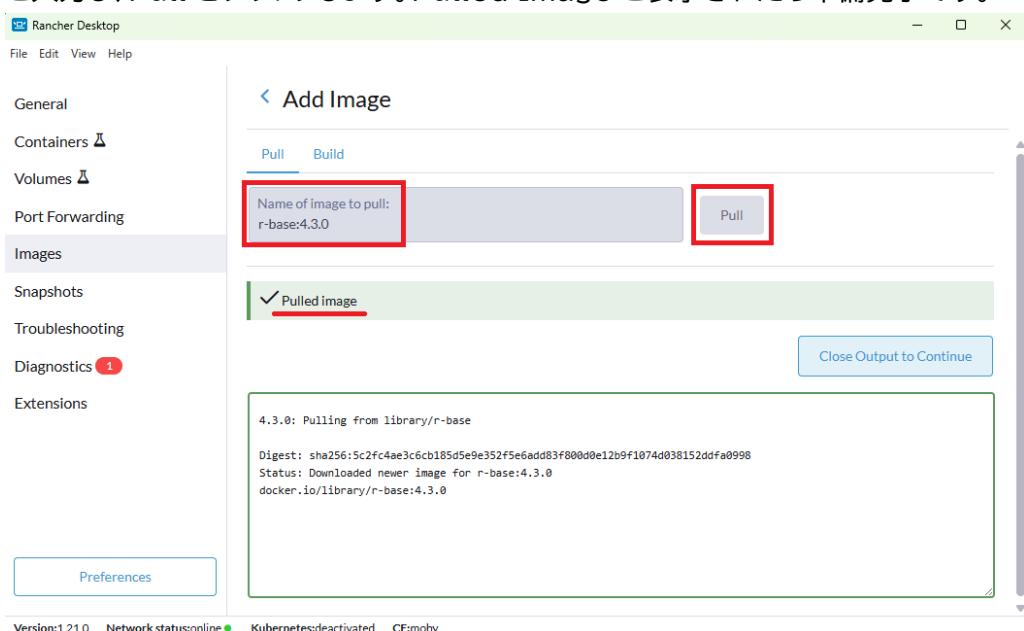
(Windows + Rancher desktop + プロキシ環境の場合のみ以下を実行)
Rancher Desktop を立ち上げ、Images > Add Image をクリックします。



Pull タブの Name of Image to pull:に

r-base:4.3.0

と入力し、Pull をクリックします。Pulled Image と表示されたら準備完了です。



(全ての環境共通)

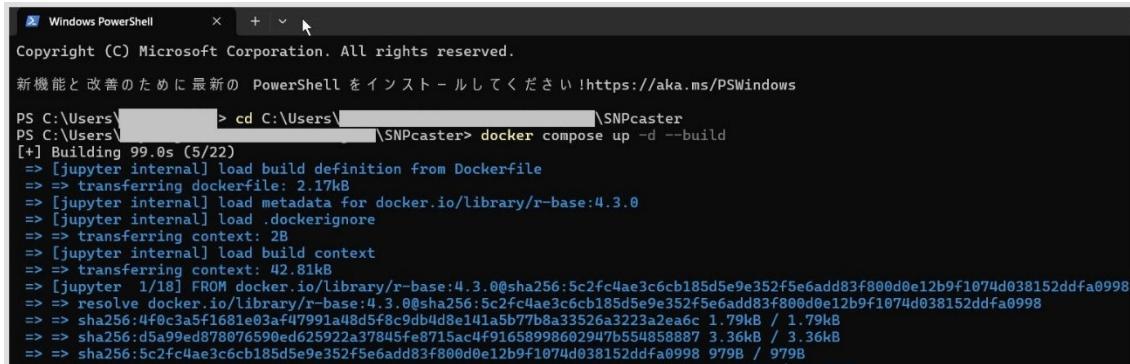
SNPcaster をインストールするために、

SNPcaster フォルダでコマンド入力アプリを開き、次のコマンドを入力します。

```
docker compose up -d --build
```

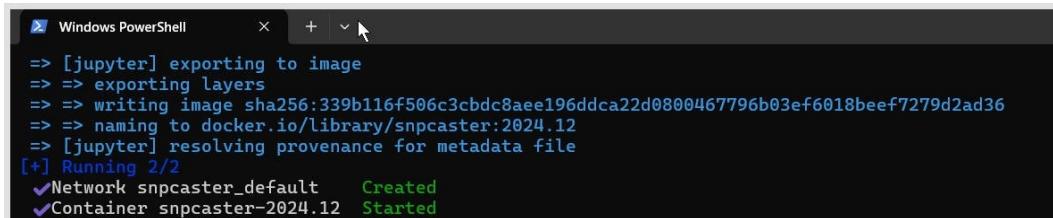
上記のコマンドを実行すると、

Docker image のビルドが開始され、以下のように経過が表示されます。



```
PS C:\Users\...> cd C:\Users\...\SNPcaster
PS C:\Users\...\SNPcaster> docker compose up -d --build
[+] Building 99.0s (5/22)
=> [jupyter internal] load build definition from Dockerfile
=> => transferring dockerfile: 2.17kB
=> [jupyter internal] load metadata for docker.io/library/r-base:4.3.0
=> [jupyter internal] load .dockerrcignore
=> => transferring context: 2B
=> [jupyter internal] load build context
=> => transferring context: 42.81kB
=> [jupyter 1/18] FROM docker.io/library/r-base:4.3.0@sha256:5c2fc4ae3c6cb185d5e9e352f5e6add83f800d0e12b9f1074d038152ddfa0998
=> => resolve docker.io/library/r-base:4.3.0@sha256:5c2fc4ae3c6cb185d5e9e352f5e6add83f800d0e12b9f1074d038152ddfa0998
=> => sha256:4f0c3a5f1681e03af47991a48d5f8c9db4d8e141a5b77b8a33526a3223a2ea6c 1.79kB / 1.79kB
=> => sha256:d5a99ed878076590ed625922a37845fe8715ac4f91658998602947b554858887 3.36kB / 3.36kB
=> => sha256:5c2fc4ae3c6cb185d5e9e352f5e6add83f800d0e12b9f1074d038152ddfa0998 979B / 979B
```

ビルドが完了すると、Container snpcaster-<バージョン番号> Started と表示されます。



```
=> [jupyter] exporting to image
=> => exporting layers
=> => writing image sha256:339b116f506c3cbdc8aee196ddca22d0800467796b03ef6018beef7279d2ad36
=> => naming to docker.io/library/snpcaster:2024.12
=> [jupyter] resolving provenance for metadata file
[+] Running 2/2
  ✓ Network snpcaster_default    Created
  ✓ Container snpcaster-2024.12  Started
```

「docker ps」を入力して、作成されたコンテナを確認します。

```
docker ps
```

Names の列が snpcaster-<バージョン> となっているものが表示されれば確認 OK です。

これで、インストールは完了です。[4 起動・停止方法](#)に進んでください。



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
df01a805ea59	snpcaster:0.9.0	"/usr/local/bin/entr_"	48 seconds ago	Up 47 seconds	0.0.0.0:59829->8888/tcp, [::]:59829->8888/tcp	snpcaster-0.9.0

3.3.3. ビルドに失敗した場合

プロキシをご利用の環境でビルドに失敗する場合、何かしらの理由でインターネット接続が失敗している可能性があります。

その場合、プロキシがない環境(ご自宅など)でビルドをしていただくとビルドがうまくいくケースがあります。一度ビルドできればそのイメージは PC 内に保存されるので、以降はインターネット接続がない状態でも SNPcaster を起動可能です。

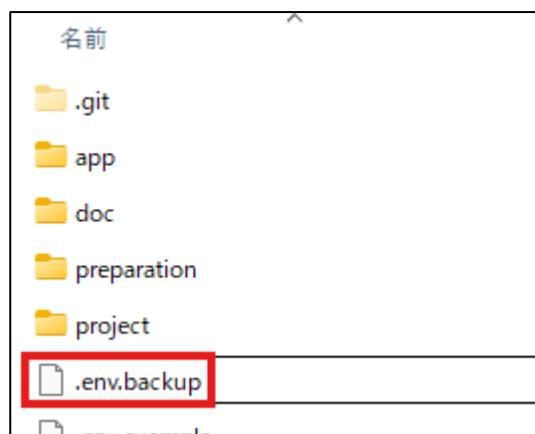
プロキシなしの環境でお試しになる場合、ここまでで実施したプロキシの設定をすべてオフにしてからビルドする必要があります。

以下の手順でプロキシ設定をオフにしてください。

3.3.3.1. OS 共通の作業

【.env の無効化】

SNPcaster フォルダに配置していた.env ファイルを.env.backup と名前を変更してください。



次の作業は、OS ごとに異なります。

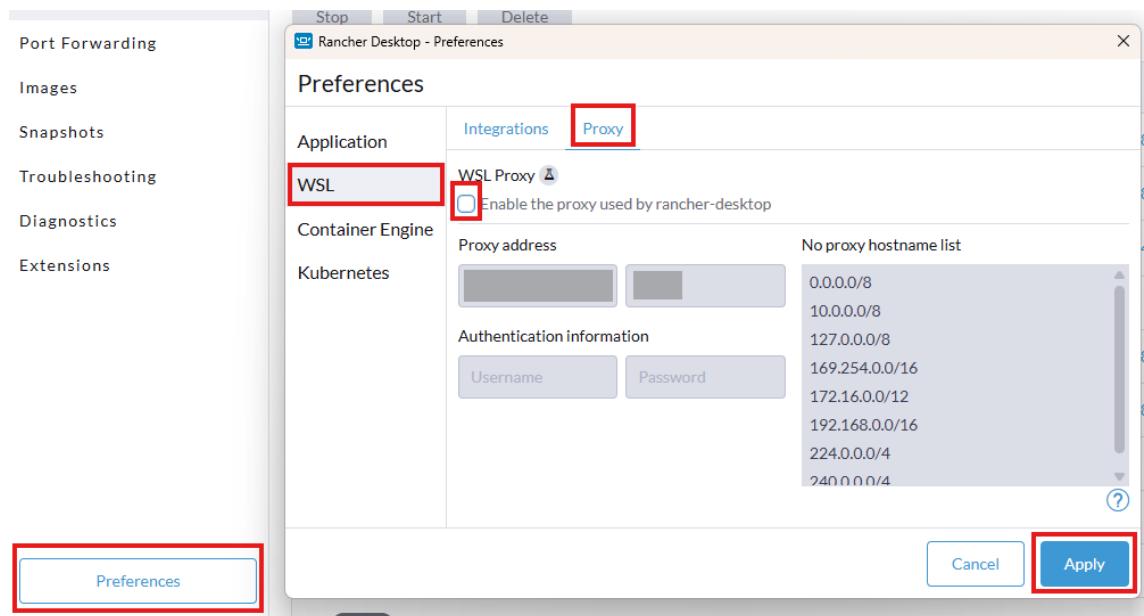
3.3.3.2. Windows

RancherDesktop か DockerEngine かで内容が変わります。

【RancherDesktopをご利用の場合】

RancherDesktop を起動して

左下の Preferences > Preferences 画面の左側 WSL > Proxy タブから
「Enable the proxy by rancher-desktop」のチェックを外して「Apply」をク
リックしてください。



プロキシなし環境での Docker イメージビルトへ進んでください。

【DockerEngineをご利用の場合】

ターミナルを開き、以下のコマンドを実行して WSL にログインします。

```
wsl -d Ubuntu-24.04
```

ログイン後、以下のコマンドを実行します。

```
nano ~/.bashrc
```

編集画面が表示されるので一番下までスクロールし、
以下の箇所の export PATH…以外の行頭に#をつきます。

```
# Porxy setteing by set_proxy.sh
#export http_proxy=""
#export HTTP_PROXY=""
#export https_proxy=
#export HTTPS_PROXY=
#export no_proxy="localhost, 127.0.0.
#export NO_PROXY="$no_proxy"
export PATH="/opt/conda/bin:${PATH}"
```

Ctrl+X を押した後、Y を入力して保存します。

保存後、以下のコマンドを実行すると適用されます。

```
source ~/.bashrc
```

[プロキシなし環境での Docker イメージビルトへ進んでください。](#)

3.3.3.3. Mac

RancherDesktop か DockerEngine かで内容が変わります。

【RancherDesktopをご利用の場合】

ターミナルを開き(開き方は、[Mac](#)をご覧ください)、以下のコマンドを入力し実行します。

```
sudo vi /etc/init.d/docker
```

Vi エディタが実行されるので、最後の行に記入した以下のプロキシ設定を削除します。

```
export http_proxy=<プロキシサーバーURL>:<ポート番号>/
export https_proxy=<プロキシサーバーURL>:<ポート番号>/
```

以下のコマンドで docker を再起動させたらプロキシ設定が解除されます。

```
sudo service docker restart
```

[プロキシなし環境での Docker イメージビルトへ進んでください。](#)

【DockerEngineをご利用の場合】

ターミナルを開き(開き方は、[Mac](#)をご覧ください)、以下のコマンドを入力し実行し

ます。

```
sudo vi ~/.zshrc
```

Vi エディタが起動するので、下にスクロールして
以下の 4 つの記載がある箇所で export の前に#をつけて保存します。

```
#export http_proxy=""
#export HTTP_PROXY=""
#export https_proxy=
#export HTTPS_PROXY=
```

設定した内容を読み込むために、以下のコマンドを実行します。

```
source ~/.zshrc
```

これで、ターミナルへのプロキシ設定がオフになりました。

プロキシなし環境での Docker イメージビルトへ進んでください。

3.3.3.4. Linux

systemd の設定ファイルに書き込んだプロキシサーバーの設定を無効化します。下記コマンドで設定ファイルを編集します。

- Ubuntu・CentOS

```
sudo systemctl edit docker
```

ターミナル中でエディターが起動するので、

[Service]から始まる 3 行(黄色塗りの箇所)の文頭に#をつけてください。

```
### Editing /etc/systemd/system/docker.service.d/override.conf
### Anything between here and the comment below will become the
contents of the drop-in file
# [Service]
# Environment="HTTP_PROXY=http://proxy.go.jp:8080"
# Environment="HTTPS_PROXY=http://proxy.go.jp:8080"

### Edits below this comment will be discarded
```

完了したら、保存してエディターを閉じます。

(「Ctrl+X」で終了→保存するか聞かれたら「Y」を入力すると、保存される)

次に、Docker クライアントの設定を無効化します。

「`~/.docker/config.json`」のファイル名を以下のコマンドで変更することで無効化します。

```
mv ~/.docker/config.json ~/.docker/config.json.backup
```

その後、以下のコマンドで Docker を再起動します。

- Ubuntu・CentOS

```
sudo systemctl daemon-reload  
sudo systemctl restart docker
```

以上でプロキシ設定の無効化は完了です。

プロキシなし環境での Docker イメージビルドへ進んでください。

3.3.3.5. プロキシなし環境での Docker イメージビルド

プロキシのないインターネットに接続し、[3.3.2 ビルドの実行](#)を実施してください。

ビルドが完了しましたら、プロキシ設定なしのままでも SNPcaster は動作します。

※ただし、サンプルデータダウンロードなど、一部のインターネット接続が必要な処理は使用できません。

ビルドを実施したマシンで解析する場合は、[4 起動・停止方法](#)へ進んでください。

ビルドした Docker イメージを他のマシンに移動して使用したい場合、次のステップに進んでください。

3.3.3.6. Docker イメージを他のマシンに移動する場合

ビルドした Docker イメージを他のマシンに移動して使用したい場合、ターミナルから以下のコマンドでイメージファイルを出力します。

```
docker save -o snpcaster.tar snpcaster:{バージョン番号}
```

※{バージョン番号}は使用している SNPcaster のバージョン番号を記載してください(例:0.9.5)。

コマンドを実行したフォルダに `snpcaster.tar` が生成されるので、USB メモリ等の外付けメモリを使って移動先のマシンにコピーしてください。

移動先のフォルダ上でターミナルを開き、以下のコマンドでイメージを復元します。

```
docker load -i snpcaster.tar
```

復元後、[4 起動・停止方法](#)へ進んで起動してください。

3.4. SNPcaster のバージョンアップ

SNPcaster の新バージョンを利用したい場合、SNPcaster をダウンロードしたときの方法に応じてアップデートを実施してください。

[3.4.1 ブラウザ\(Internet Explorer, Google Chrome など\)を使ってダウンロードした場合](#)

[3.4.2 コマンドライン\(Git\)を使ってダウンロードした場合](#)

3.4.1. ブラウザ(Internet Explorer, Google Chrome など)を使ってダウンロードした場合

3.1.1 ブラウザ(Microsoft Edge, Google Chrome など)を使ってダウンロードの方法で SNPcaster をインストールした場合、以下の手順でアップデートしてください。

3.4.1.1. 古いバージョンの SNPcaster コンテナの削除

まず、以下の手順で古いバージョンの Docker コンテナを削除します。

コマンド入力アプリを開き、ダウンロードした「snpcaster」フォルダへ移動します。

移動したら、下記コマンドで Docker コンテナの削除を行います。

※コンテナを削除しても、project フォルダに置かれている解析データは残ります。

※もし、コマンド実行等で project フォルダの外にデータを置いている場合、削除前にバックアップを取ってください。

```
docker compose down -v
```

```
[+] Running 2/2
✓ Container snpcaster    Removed  0.7s
✓ Network k2077_snpcaster_notebook_default      Removed  0.3s
```

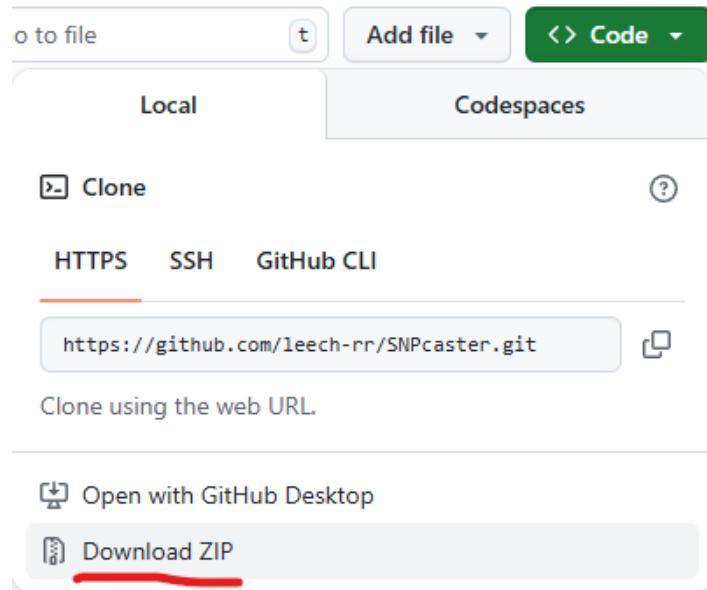
Jupyter Lab にアクセスした Web ブラウザはそのまま閉じて問題ありません。

3.4.1.2. 新しいバージョンのダウンロード

お好きなブラウザで <https://github.com/leech-rr/SNPcaster/tree/main> を開きます。

右上の Code をクリックし、その後「Download Zip」をクリックすると最新の

SNPcaster がダウンロードされます。

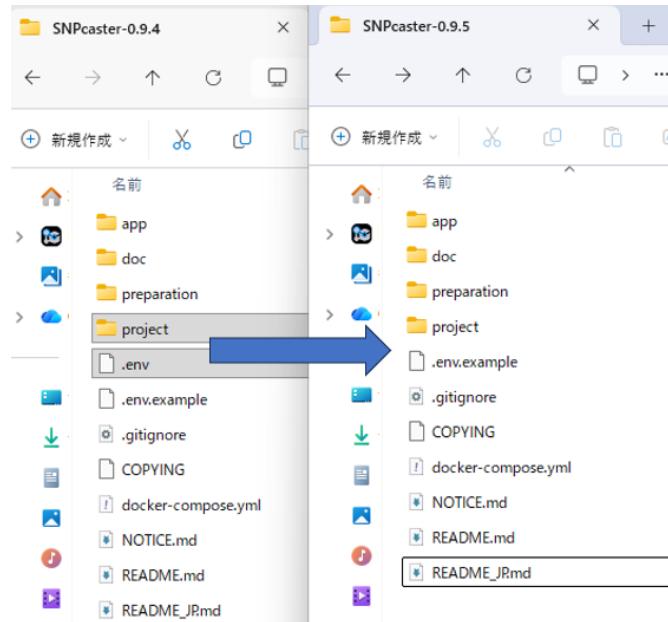


ダウンロードが完了したら、作業を行いたいフォルダに移動して解凍してください。

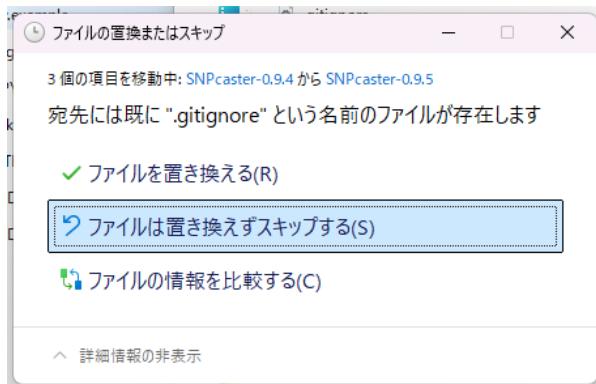
3.4.1.3. データの移行

古いバージョンのフォルダから、以下のファイル・フォルダを移動します。

- project フォルダ
- (プロキシ設定用に作製していれば) .env ファイル



project フォルダを移動する際、以下のような確認画面が表示されます。
「ファイルは置き換えずスキップする」を選んでください。
※「ファイルを置き換える」を選んでしまった場合でも特に支障はありません。



3.4.1.4. SNPcaster のビルドと起動

後の手順は起動時と同じです。

[4.1 起動方法](#)へ進んでください。

3.4.2. コマンドライン(Git)を使ってダウンロードした場合

[3.1.2 コマンドライン\(Git\)を使ってダウンロード](#)の方法でダウンロードした場合、

以下の 3 つのコマンドだけでアップデート可能です。

3.4.2.1. 古いバージョンの SNPcaster コンテナの削除

まず、以下の手順で古いバージョンの Docker コンテナを削除します。

コマンド入力アプリを開き、古いバージョンの「snpcaster」フォルダへ移動します。

移動したら、下記コマンドで Docker コンテナの削除を行います。

※コンテナを削除しても、project フォルダに置かれている解析データは残ります。

※もし、コマンド実行等で project フォルダの外にデータを置いている場合、削除前にバックアップを取ってください。

```
docker compose down -v
```

```
[+] Running 2/2
✓ Container snpcaster    Removed  0.7s
✓ Network k2077_snpcaster_notebook_default      Removed  0.3s
```

Jupyter Lab にアクセスした Web ブラウザはそのまま閉じて問題ありません。

3.4.2.2. 最新の SNPcaster の取得

以下のコマンドを実行することで、git から最新の SNPcaster をダウンロードできます。

なお、{ダウンロードしたフォルダ}の箇所は、最初に SNPcaster をダウンロードしたフォルダのパスを記載してください。

```
cd {ダウンロードしたフォルダパス}  
git pull
```

3.4.2.3. SNPcaster のビルドと起動

後の手順は起動時と同じです。

[4.1 起動方法](#)へ進んでください。

4. 起動・停止方法

本章では、SNPcaster の Docker コンテナの起動/停止方法を説明します。

もし [3.3.2 ビルドの実行](#)の直後であれば起動済みの状態ですので、[5 解析実行手順](#)へ進んで解析を行ってください。

4.1. 起動方法

4.1.1. 起動コマンド実行

コマンド入力アプリを開き、ダウンロードした「snpcaster」フォルダへ移動します。
移動したら、下記コマンドで Docker コンテナの起動を行います。

```
docker compose up -d --build
```

```
[+] Running 1/0
✓ Container snpcaster Running
```

「✓ Container snpcaster-<バージョン> Running」と出力されれば、起動完了です。

※起動に失敗した場合について※

以下のような文言で、エラーが出て起動に失敗する場合があります。

ポートが使用済みの場合のエラー

```
[+] Running 0/1
- Container snpcaster Starting
          0.4s
Error response from daemon: driver failed programming external connectivity on
endpoint snpcaster (f2ce4c3e2fd6a520c04ea148c23dc863e84d3d9019d68e30c46d038d1
02fd59c): Bind for 0.0.0.0:59829 failed: port is already allocated
```

同名のコンテナが存在する場合のエラー

```
[+] Running 0/1
- Container snpcaster-0.9.0 Creating
          2.8s
Error response from daemon: Conflict. The container name "/snpcaster-0.9.0" is
already in use by container "76d4ceaafffc34de24a0fe8b6b795027703469b858d808ed0
6b243342bea20265". You have to remove (or rename) that container to be able to
reuse that name.
```

これらは、古いバージョンの SNPcaster のコンテナがすでに作製または起動済みの場合に発生します。

コンテナの[停止方法](#)の手順で古いバージョンのコンテナを止めてから、再度起動コマンドを実行してください。

4.1.2. ポートフォワード設定（リモートアクセスを行っている場合のみ）

※自分のPC上でSNPcasterを起動している場合、この設定は不要です※

リモートサーバを使っている場合、ローカルの59829ポートにアクセスしたときに、リモートサーバの59829ポートにアクセスできるように、SSHポートフォワーディングの設定をすると便利です。

ローカルで下記のコマンドを実行します。

※ユーザー名とサーバーIPは、サーバーの環境に合わせて変更してください。

※ローカルマシンにDockerコンテナを起動している場合は不要です。

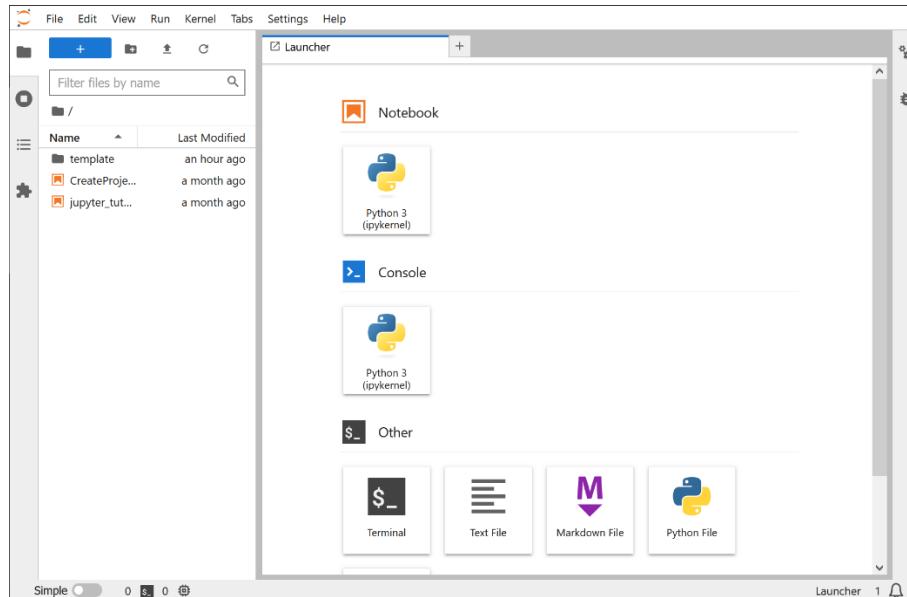
```
# ssh -L 59829:localhost:59829 <ユーザー名>@<リモートサーバーIP>
```

```
ssh -L 59829:localhost:59829 ubuntu@192.168.32.55
```

4.1.3. Jupyter Lab へのアクセス

Google Chrome などの Web ブラウザを起動し、URL 欄に「<http://localhost:59829/>」を入力して Enter を押します。

しばらくすると、Jupyter Lab が開きます。下記の画面が表示されれば完了です。



これで解析が可能となったので、[解析実行手順](#)に進んでください。

4.2. 停止方法

※停止は必須ではありませんが、PC の再起動時に JupyterLab がうまく立ち上がらない場合があります。そのような場合には、以下の方法で停止後、再度 sncpcaster を起動してください。

コマンド入力アプリを開き、ダウンロードした「sncpcaster」フォルダへ移動します。

移動したら、下記コマンドで Docker コンテナの停止(実際には削除)を行います。

※コンテナを削除しても、project フォルダに置かれている解析データは残ります。

```
docker compose down -v
```

```
[+] Running 2/2
✓ Container sncpcaster    Removed  0.7s
✓ Network k2077_sncpcaster_notebook_default      Removed  0.3s
```

Jupyter Lab にアクセスした Web ブラウザはそのまま閉じて問題ありません。

※起動している SNPcaster のフォルダが分からない場合※

起動しているコンテナの一覧から、停止するコンテナを探します。

コマンド入力アプリを開き、以下のコマンドを実行してください。

```
docker ps -a
```

以下の画像のようの一覧が表示されるので、

「NAMES」の列(画像の右端)で **snpcaster-<バージョン番号>** となっているコンテナを探し、

そのコンテナの「CONTAINER ID」(画像の左端)をコピーします。

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
76d4ceaaafffc3	snpcaster:0.9.0	"/usr/local/bin/entr..."	21 hours ago	Up 11 minutes	0.0.0.0:59829->8888/tcp, [::]:59829->8888/tcp	snpcaster-0.9.0

以下のコマンドを実行すると、コンテナの停止+削除を実行できます。

※{コンテナ ID}の箇所は、先ほどコピーした CONTAINER ID に差し替えてください。

```
docker rm -f {コンテナ ID}
```

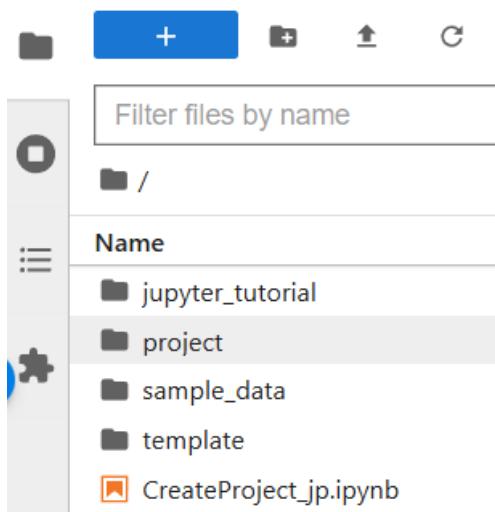
これで、コンテナの停止が完了しました。

5. 解析実行手順

[Jupyter Lab へのアクセス](#)を行った後、以下の手順で解析を行ってください。

5.1. フォルダ構成

初期のフォルダ構成は、以下の通りです。



- jupyter_tutorial
 - Jupyter の使い方を学ぶためのノートブック等が含まれるフォルダ
- project
 - 解析プロジェクトを配置するフォルダ
 - SNPcaster フォルダ中の project フォルダと共有されます
- sample_data
 - STEC 用のサンプル参照配列などを配置したフォルダ
- template
 - 解析ノートブックのテンプレートフォルダ
- CreateProject.ipynb
 - 新規プロジェクト作成用ノートブック

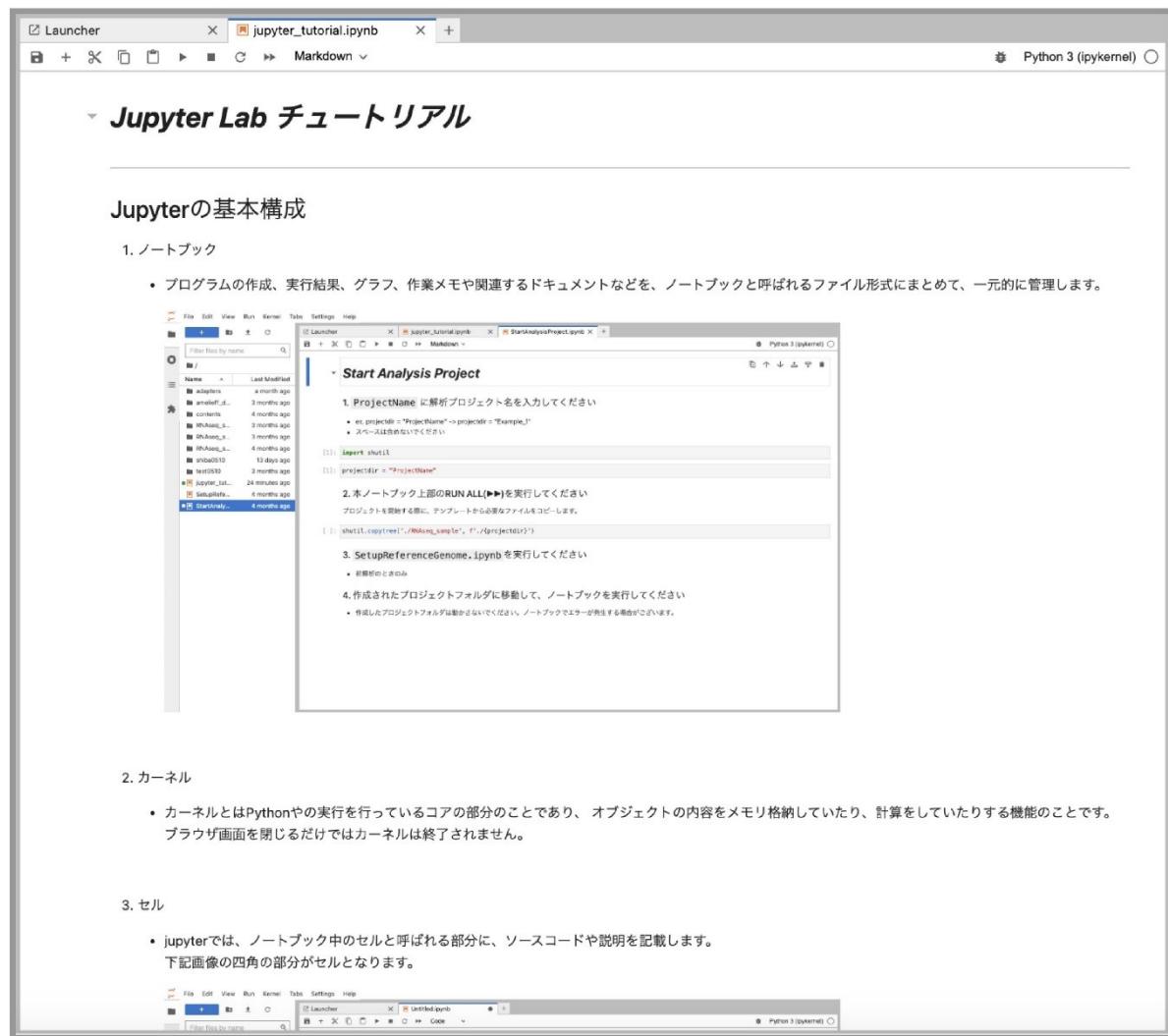
5.2. Jupyter Lab の使い方を学ぶ

Jupyter Lab 初心者の方向けに、基本的な使い方をこの章では記載しています。

不要な方は、次の[プロジェクト作成](#)に進んでください。

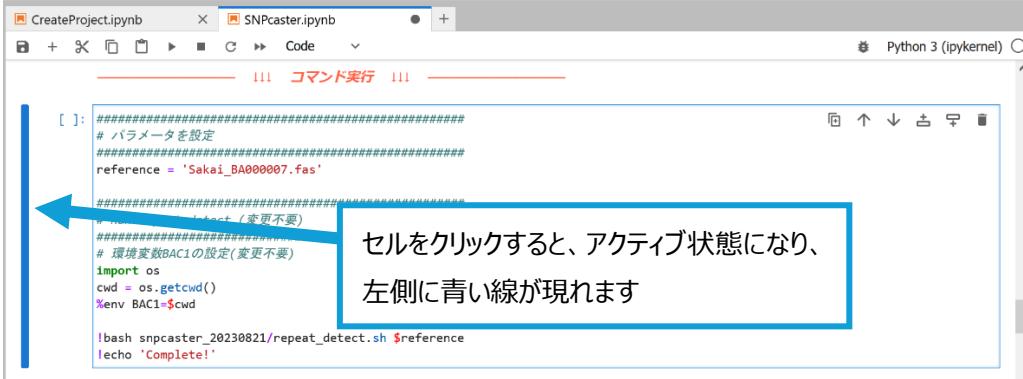
5.2.1. Jupyter Lab チュートリアル

左メニューから「jupyter_tutorial」フォルダをダブルクリックし、さらに、「jupyter_tutorial.ipynb」をダブルクリックすると右側にノートブックが開きます。このノートブックには、Jupyter Lab の使用方法、Jupyter Notebook の編集・実行方法が記載されています。Jupyter 初心者の方は、こちらのノートブックを一読することをお勧めします。



5.2.2. Jupyter Lab の基本的な使い方

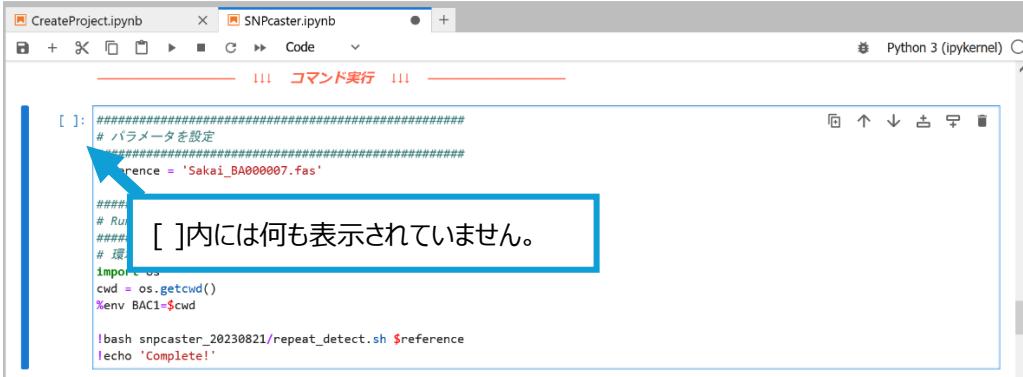
実行するセルをクリックします。



The screenshot shows the Jupyter Lab interface with two tabs at the top: 'CreateProject.ipynb' and 'SNPcaster.ipynb'. The 'Code' tab is selected. A code cell in the 'SNPcaster.ipynb' tab contains Python and Bash code. The first line of the cell is highlighted with a blue arrow pointing to its left margin. A callout box with a blue border contains the text: 'セルをクリックすると、アクティブ状態になり、左側に青い線が現れます' (When you click a cell, it becomes active, and a blue line appears on the left side). The cell itself contains the following code:

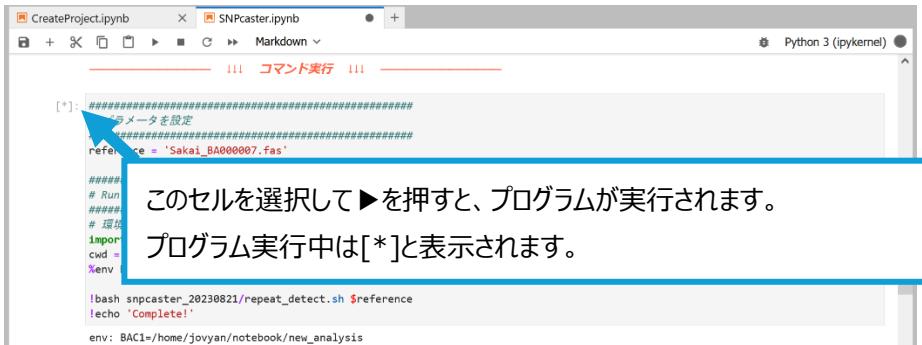
```
[1]: #####  
# パラメータを設定  
#####  
reference = 'Sakai_BA000007.fas'  
#####  
# Run  
#####  
# 環  
import os  
cwd = os.getcwd()  
%env BAC1=$cwd  
  
!bash snpcaster_20230821/repeat_detect.sh $reference  
echo 'Complete!'
```

セル内のプログラムの実行前には、左側の[]には何も表示されません。



The screenshot shows the Jupyter Lab interface with the same tabs and configuration as the previous screenshot. However, the code cell in the 'SNPcaster.ipynb' tab is now empty, indicating it has been executed. A blue arrow points to the left margin of the cell, and a callout box with a blue border contains the text: '[]内には何も表示されていません。' ([] does not contain any displayed text).

各セルでプログラムを実行すると、セル左側の[]内が [*] 表示となります。



```
[*]: ##### メータを設定#####
reference = 'Sakai_BA00007.fas'

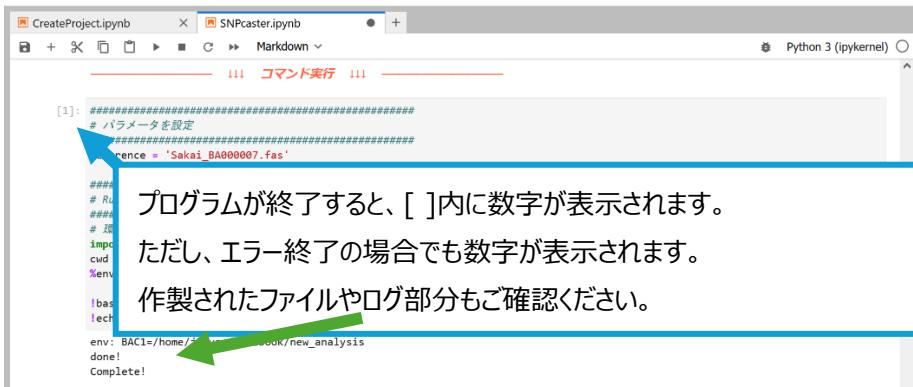
#####
# Run
#####
# 現在
import
cwd =
%env

!bash snpcaster_20230821/repeat_detect.sh $reference
!echo 'Complete!'

env: BAC1=/home/jovyan/notebook/new_analysis
```

このセルを選択して▶を押すと、プログラムが実行されます。
プログラム実行中は[*]と表示されます。

プログラムが終了すると[]内に数字が表示されます。



```
[1]: ##### メータを設定#####
reference = 'Sakai_BA00007.fas'

#####
# Run
#####
# 現在
import
cwd =
%env

!bash
!echo

env: BAC1=/home/jovyan/notebook/new_analysis
done!
Complete!
```

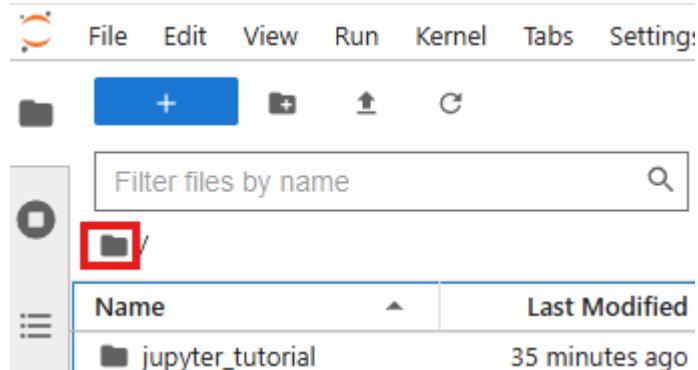
プログラムが終了すると、[]内に数字が表示されます。
ただし、エラー終了の場合でも数字が表示されます。
作製されたファイルやログ部分もご確認ください。

5.3. プロジェクト作成

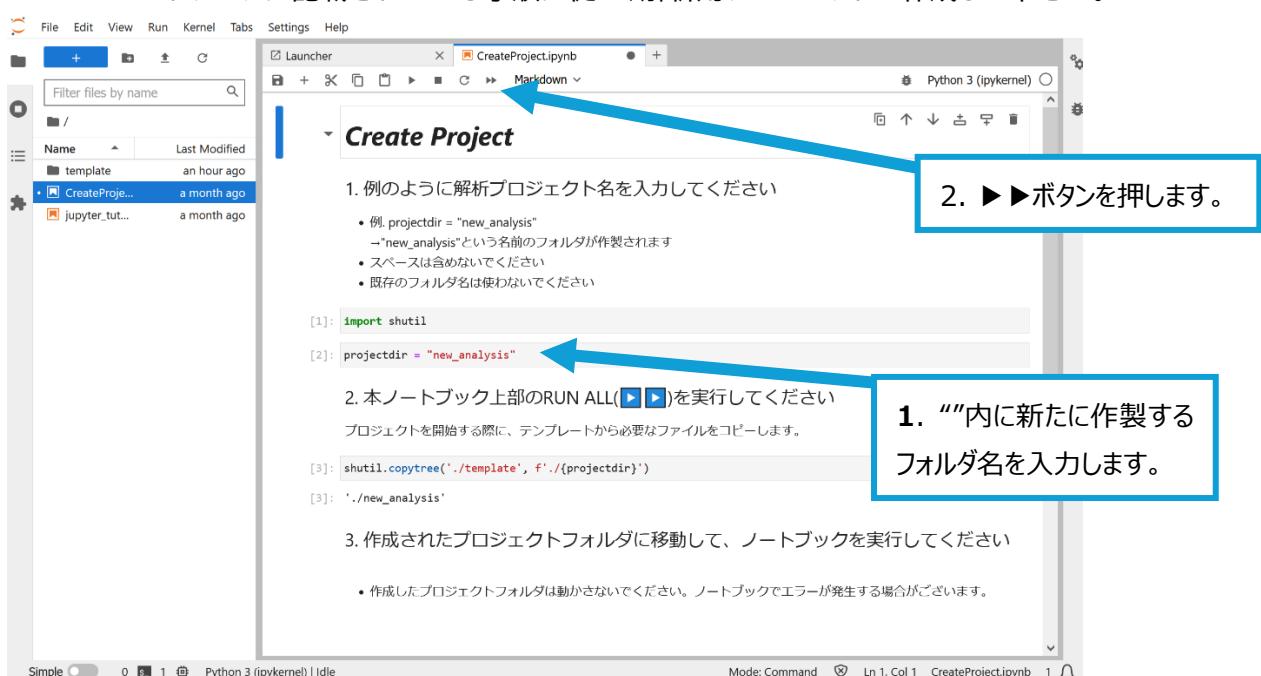
SNPcaster/grape は比較を行う 1 まとまりの菌株群を 1 つのフォルダ中で処理する構成としています。この 1 つのフォルダが 1 つのプロジェクトに該当します。
ここでは、新しいプロジェクトの作成方法を説明します。

5.3.1. プロジェクトの新規作成実行

左側メニューバーの検索ボックス下にある  (下記画像の ) をクリックします。
これにより、ルートフォルダに移動し、「 /」と表示されます。



メニューの一覧にある「CreateProject.ipynb」をダブルクリックすると、日本語プロジェクト作成用のノートブックが右側に表示されます。
ノートブックに記載されている手順に従い、解析用プロジェクトを作成して下さい。



1. 例のように解析プロジェクト名を入力してください

- 例. projectdir = "new_analysis"
→ "new_analysis"という名前のフォルダが作製されます
- スペースは含めないでください
- 既存のファイル名は使わないでください

2. ▶▶ボタンを押します。

1. """内に新たに作製する
フォルダ名を入力します。

2. 本ノートブック上部のRUN ALL(▶▶)を実行してください
プロジェクトを開始する際に、テンプレートから必要なファイルをコピーします。

```
[1]: import shutil  
[2]: projectdir = "new_analysis"
```

3. 作成されたプロジェクトフォルダに移動して、ノートブックを実行してください

- 作成したプロジェクトフォルダは動かさないでください。ノートブックでエラーが発生する場合がございます。

「CreateProject_jp.ipynb」の実行が完了すると、
新しい解析プロジェクトフォルダ(以下では「test」フォルダ)が
「project」フォルダの配下に作成されます。

The screenshot shows the Jupyter Notebook interface. On the left, a file browser displays a directory structure with a 'project' folder highlighted. In the center, a notebook cell contains code for creating a project:

```
[1]: import shutil  
[2]: projectdir = "test"
```

To the right, a 'Create Project' section provides instructions:

1. 例のように解析プロジェクト名を入力
 - 例. projectdir = "new_analysis"
→ "new_analysis"という名前のフォルダが作製されます
 - スペースは含めないでください
 - 既存のフォルダ名は使わないでください
2. 本ノートブック上部のRUN ALL(▶▶)

A large orange arrow points downwards from the 'Create Project' section to the file browser, indicating the result of the process. The file browser now shows a 'project' folder containing a 'test' folder.

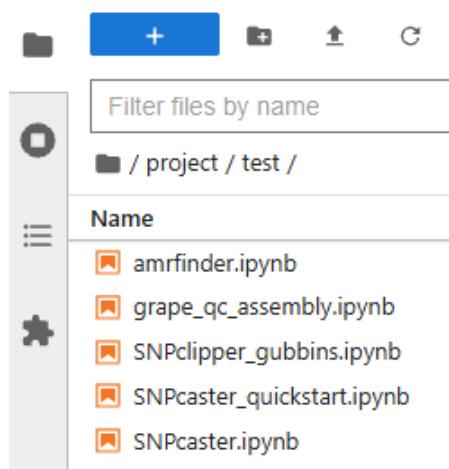
※ご注意※

project フォルダ以外の場所にフォルダやファイルを作製・配置しないでください。
ホスト OS(Windows や Mac など)と共有されていないため、Docker コンテナを削除した際に(docker compose down をしたとき)、コンテナと一緒に削除されてしまいます。

5.3.2. プロジェクトの新規作成実行

作成した新しい解析プロジェクトフォルダをダブルクリックします。

初期状態では、以下のファイルが配置されています。



- amrfinder.ipynb
 - AMR 遺伝子等の解析ノートブック
- grape_qc_assembly.ipynb
 - grape_qc の解析ノートブック
- SNPcaster_quickstart.ipynb
 - SNPcaster の簡易解析ノートブック
- SNPcaster.ipynb
 - SNPcaster 解析ノートブック
- SNPclipper_gubbins.ipynb
 - SNPclipper/gubbins 実行ノートブック

5.3.3. 解析の実行

以下の箇条書きの中から、解析したいノートブックをダブルクリックで開いて解析を行ってください。

詳細な解析手順はノートブック内に記載しておりますので、そちらの説明をご覧ください。

基本的な使い方は、次の基本的な使い方の章をご覧ください。

- amrfinder.ipynb
 - 薬剤耐性遺伝子や、病原性遺伝子を検出するプログラムです。
- grape_qc_assembly.ipynb
 - ショートリードデータのアセンブリおよびクオリティチェックを行います。
- SNPcaster_quickstart.ipynb
 - 志賀毒素産生性大腸菌(STEC)用の、SNP 解析の簡易版プログラムです。
 - 参照配列等は全て用意されていますので、ショートリードデータおよび菌株リストを用意するだけで解析が開始できます。
- SNPcaster.ipynb
 - マスク領域の生成など、SNPcaster 実行のために必要な準備を含めて実行できるプログラムです。
 - 参照配列からのリピート領域検出や、ドラフトゲノムを参照配列にした解析など、細かな設定が可能です。出力ファイルの詳細な説明も含まれています。
- SNPclipper_gubbins.ipynb
 - SNPcaster 実行後、マスク処理やクラスターSNP を異なる条件で実行する、もしくは追加で gubbins を実行できるプログラムです。

※注意点※

- メモリの関係で、一度に複数の解析を実行すると解析が中断される場合があります。その場合は、一度に一つの解析のみ実行してください。
- SNPcaster は一度に多数の株を解析すると解析が中断されることがあります。株数は供試菌のゲノムサイズや端末のメモリ容量によります(192 GB メモリで 300 株位)。その場合は、一度に解析する株数を減らして解析してください。
- 廉価版の CPU(Intel Atom, Celeron など AVX 非対応のもの)や、ARM アーキテクチャの CPU(Apple silicon など)の場合、系統樹解析が動作しないことが確認されています。その場合、系統樹解析は Web 版をご利用ください。

➤ <https://www.hiv.lanl.gov/content/sequence/IQ TREE/iqtree.html>

目的別の使い方ガイド

- 動作確認をしたい
→ `SNPcaster_quickstart.ipynb` で動作確認をしてください
- SNP 解析をしたい
→ データのチェックを `grape_qc_assembly.ipynb` で行ってから、
`SNPcaster.ipynb` で解析することをお勧めします。
腸管出血性大腸菌の場合は、`SNPcaster_quickstart.ipynb` も利用できます。
- SNP 解析した後に Gubbins を行いたい。クラスターSNP 等の条件検討を行いたい。
→ `SNPclipper_gubbins.ipynb` をご使用ください。
- 既に解析したサンプルを再解析したい
→ [5.3.5 SNP 再解析の実行](#)をご参照ください。
- コマンドラインで使用したい。
→ [6 コマンドライン操作（上級者向け）](#)をご参照ください。

5.3.4. 基本的な使い方

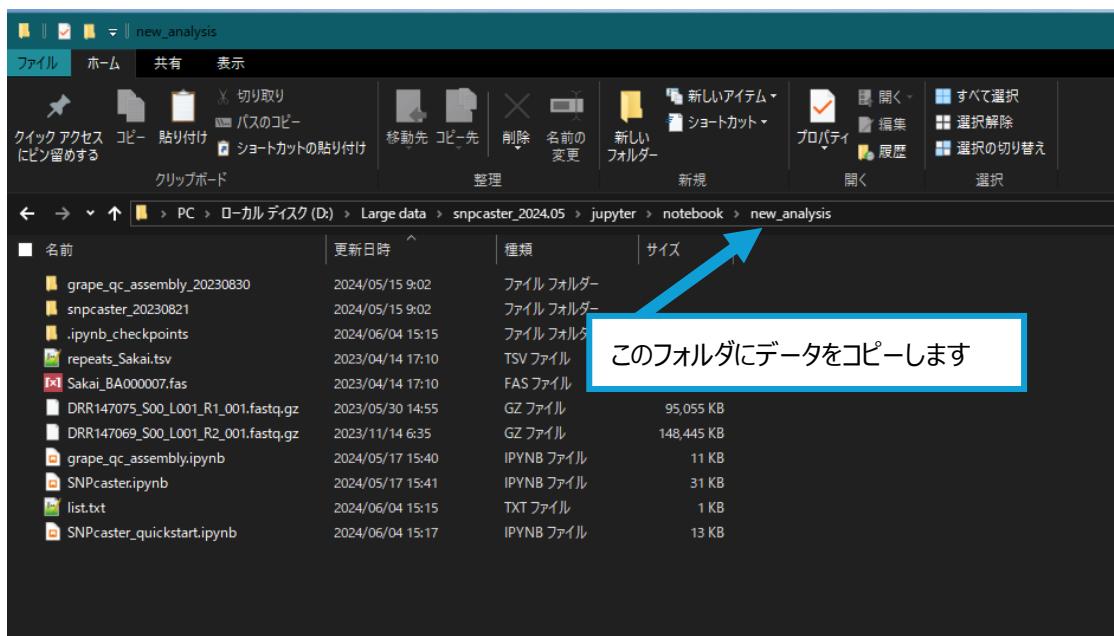
「SNPcaster_quickstart.ipynb」を例として、基本的な使い方を解説します。

5.3.4.1. 入力ファイルのアップロード

JupyterLab サイドバーのボタンまたは、下記のようにフォルダにドラッグアンドドロップでデータをアップロード(コピー)してください。



「new analysis」というプロジェクトで開始した場合、以下のようなフォルダが作製されているので、そこへファイルをコピーします。



5.3.4.2. 菌株リストの作製

JupyterLab に記載の方法で、リストを作製してください。

2.2. の方法で作製する場合、Windows で作製するとエラーの原因となります。

Linux または Mac で作製することをお勧めします。

2.1. または 2.3. の方法で作製する場合には、どの OS でも可能です。



The screenshot shows the JupyterLab interface with a code cell containing the following Python code:

```
[ ]: # リスト入力用のファイルを読み込む
file_name = "list.txt"
# 必要リストを読み込む
user_input = []
for i in range(10000):
    user_input.append(str(i))
# 最初の行を削除
user_input.pop(0)
# 空間に改行を追加
user_input.append("\n")
# 入力された要素をファイルに書き込む
with open(file_name, "w") as file:
    file.write(user_input)
```

A small pop-up window is visible in the bottom right corner, asking if the user wants to receive official Jupyter news. It includes links to "Open privacy policy" and "Yes" or "No" buttons.

5.3.4.3. FASTQ リストの作製

JupyterLab に記載されている 3.1 と 3.2 の方法で、FASTQ リストを作製してください。

FASTQ リストとは、菌株名とそれに紐づくペアエンドのショートリードファイル名(FASTQ ファイル名)2 つを一行に記載したリストのことです。

SNPcaster 中で実行される BactSNP は、このリストを入力として実行されます。

3. FASTQリストの作成

2.で作成した菌株リスト中の各菌株に紐づくFASTQファイル名(ショートリードデータのファイル名)一覧の作成を以下の手順で行います。

3.1. find_strain_pairs.pyの実行

- 「file_name =」の後に、前の項で作成した株名リスト名をダブルクオート(")で囲って入力してください。
- パラメータを設定 に記載されている内容で修正したいものがあれば修正し、実行してください。
 - 記載されている内容のまま実行することも可能です。

----- コマンド実行 -----

```
[ ]: import subprocess
#####
# パラメータを設定
#####
# 2. で作成した菌株リストのファイル名を入力
file_name = "list.txt"
# FASTQファイルの拡張子を入力
file_extension = "fastq.gz"
# 作成するFASTQリストのファイル名
fastq_list_name = "list_fastq.tsv"
# ペアが見つからなかつた菌株一覧のファイル名
unpaired_list = "unpaired_fastq.tsv"

#####
# Run find_strain_pairs.py
# ※以下は変更しないでください※
#####
command = [
    "find_strain_pairs.py", file_name,
    "-file_extension", file_extension,
    "-paired_list", fastq_list_name,
    "--unpaired_list", unpaired_list
]
subprocess.run(command, capture_output=False, text=True)
!echo 'Complete!'
```

3.2. FASTQリストの確認・修正

- list_fastq.tsv (FASTQリスト)を開いて確認し、各菌株名に対し2つのFASTQファイル(Read1と2)名が正しく記載されていることを確認します。
- unpaired_fastq.tsv を開いて確認し、FASTQファイルのペアが見つからなかつた菌株がないか確認します。
 - あった場合、list_fastq.tsv にその菌株名をコピーして、タブ区切りでFASTQファイルのペアを記載してください。
- FASTQリストの記載例は以下の通りです(各列はタブ区切り、改行コードは\n)。

```
strain1 strain1_R1.fastq.gz strain1_R2.fastq.gz
strain2 strain2_R1.fastq.gz strain2_R2.fastq.gz
```

5.3.4.4. SNPcaster プログラムの実行

以下の画像に記載の順に作業して SNPcaster を実行します。

The screenshot shows a Jupyter Notebook interface with a code cell containing a Python script. A blue arrow points from the top right towards the cell, with a callout box labeled "3. ▶ボタンを押します。". Another blue arrow points from the bottom right towards the first line of the code, with a callout box labeled "1. このセルをクリックすると、このセルがアクティブ状態になります (左側に青い線が現れる)". A third blue arrow points from the bottom right towards the middle of the code, with a callout box labeled "2. この部分のみを編集します。EHEC の解析の場合は、リスト名（上記 2 で作製したファイル名）、スレッド数（解析端末のスペックに合わせてください）のみ変更してください。".

```
# パラメータを設定
reference = "/path/to/reference.fa"
list = "list.txt"
cluster = 0
reports = "reports_Sekai1.tsv"
threads = 6
gubbins = 1

# Run SNPcaster
# Run SNPcaster
# 実行する際は、引数を変更しない
import os
cmd = os.getenv()
cmd.append("SNPcaster")
!bash sncaster_20230821/sncaster.sh \
    -r $reference \
    -l $list \
    -c $cluster \
    -d $reports \
    -t $threads \
    -g $gubbins
echo "Complete!"
```

5.3.4.5. 系統樹の作製

SNPcaster の結果ファイルは 1 つのフォルダにまとめられます。

フォルダ名は、"snpcaster_[日付]_[時間]_[リスト名]"となっており、解析ごとに異なります。

系統樹の作製には、結果フォルダ内のファイルを指定する必要がありますので、`input` ファイル名を変更してください。

The screenshot shows a Jupyter Notebook interface with a code cell containing a command-line configuration for RAXML-NG. A blue arrow points from the bottom right towards the configuration parameters, with a callout box labeled "この部分のみを編集します。解析フォルダ名を実際のフォルダ名に変更してください。また、スレッド数も解析端末のスペックに合わせてください。".

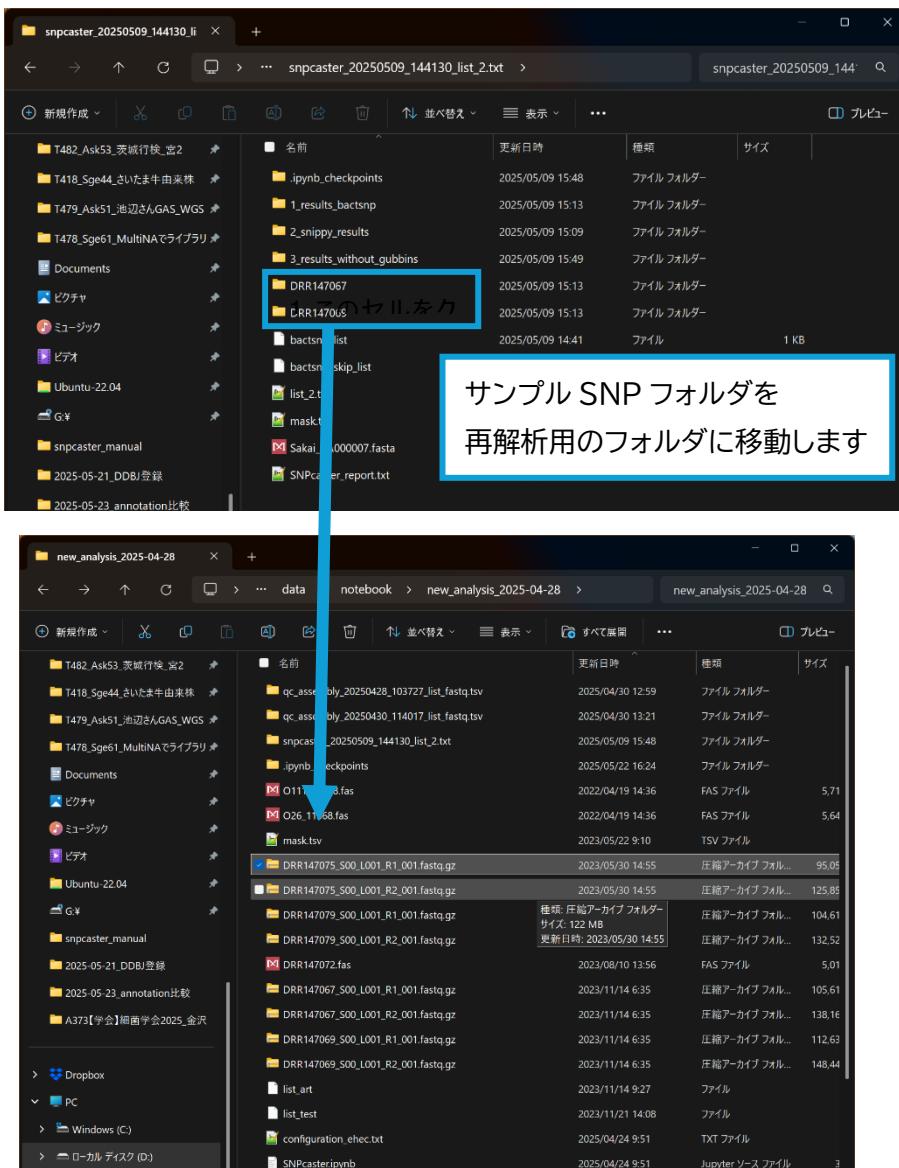
```
4. 統系樹の作製
# RAXML-NGの実行
# 実行して、Input.txt の中に解析するSNPcasterの結果を入力してください。
# RAXML-NG20240115_090704.RaxML results/sncaster_gubbinsFinal.log
# "主の引数": "-G<path>/RAxML-NG20240115_090704.RaxML results/sncaster_gubbinsFinal.log", "T": <スレッド数>
# 必要にして、各配列の最後(「」)は取り飛ばしたり、削除したりしてください。
# コマンド
# bash sncaster_20230821/sncaster.sh input [threads] [bootstrap]
# オプション
# パラメータ 必要か 意味
# input: • 入力ファイル名(例: input), input alignment: in RAXML-HA仕様でAUTO+format
# threads: • moderation-type: (例: 1000, 2000, 3000, 4000, 5000, 6000)
# bootstrap: bootstrap: (例: 1000, 2000, 3000, 4000, 5000, 6000)

# ピックは、実行メモリとのressourcesを照合してください。【実行の前によく確認】
# パラメータを設定
# RAXML-NG20240115_090704.RaxML results/sncaster_gubbinsFinal.log
# threads=4
# bootstrap=1000
# Raxml
# RAXML-NG20240115_090704.RaxML results/sncaster_gubbinsFinal.log
# bash sncaster_20230821/sncaster.sh input threads bootstrap > raxml.log
echo "Complete!"
```

5.3.5. SNP 再解析の実行

コアゲノム SNP の抽出の際に、snippy-core の機能を用いる事で、迅速な再解析が可能です。例えば、初回の解析で ABC の 3 株を解析したとします。新たに D 株を含めた株(つまり ABCD の 4 株)の SNP 解析を行う場合には、初回解析で得られたサンプル SNP フォルダを用いる事で、ABC 株については BactSNP の解析をスキップし、D 株のみ BactSNP を行います。解析時間の大部分は BactSNP によるものなので、大幅な時間短縮が見込めます。逆に、AB 株のみで再解析する場合には、BactSNP は全く行われません。この場合にも大幅な時間短縮が見込めます。

5.3.5.1. 解析するフォルダに、サンプル SNP フォルダを移動します。



5.3.5.2. snpcaster.sh を実行します。

この際、fastq 用リストに再解析株（サンプル SNP フォルダが存在する株）の情報が含まれている必要はありません。

6. コマンドライン操作 (上級者向け)

jupyter notebook 上で実行できるスクリプトはすべて Docker コンテナ内で PATH が通っているので、Docker コンテナ内であれば任意のフォルダ上で実行可能です。

本章では基本的なコマンドライン実行方法について何通りか記載しますので、お好みの方法で実行してください。

スクリプトのファイル名やコマンドライン引数は jupyter notebook をご確認ください。

6.1. コマンド入力アプリの起動

インストールした Docker の種類や OS に応じて起動するアプリが変わります。

以下からご自身の環境にあったものを起動してください。

6.1.1. Windows

6.1.1.1. Rancher Desktop もしくは Docker Desktop の場合

ターミナルもしくは PowerShell を起動し、次の [Docker イメージ・コンテナの生成](#) に進んでください。

6.1.1.2. Docker Engine の場合

ターミナルもしくは PowerShell を起動後、以下のコマンドで WSL にログインした上で、次の [Docker イメージ・コンテナの生成](#) に進んでください。

```
wsl.exe -d Ubuntu-24.04
```

6.1.2. Mac

6.1.2.1. Rancher desktop もしくは Docker Desktop の場合

ターミナルを起動し、次の [Docker イメージ・コンテナの生成](#) に進んでください。

6.1.2.2. Docker Engine の場合

ターミナルを起動後、以下のコマンドで snpcaster 用の vm を起動した上で、次の [Docker イメージ・コンテナの生成](#) に進んでください。

```
limactl start snpcaster-vm
```

6.1.3. Linux

6.1.3.1. Docker Engine の場合

Terminal を起動し、次の [Docker イメージ・コンテナの生成](#) に進んでください。

6.2. Docker イメージ・コンテナの生成

コマンドラインで実行する場合も、`docker compose` による Docker イメージもしくはコンテナの作成が事前に必要です。

もしまだ実施していない場合は、[起動コマンド実行](#) の項を実施してください。

その後、[Docker コンテナから実行](#) もしくは [Docker イメージから実行](#) のいずれかの方法でコマンド実行を実施してください。

6.3. Docker コンテナから実行

Docker コンテナを生成した後の実行方法です。

6.3.1. データの配置

ホスト上(=お使いの PC の OS 環境)で [SNPcaster ダウンロードフォルダ]/project フォルダに解析に必要なフォルダやリードデータを配置してください。

このフォルダをコンテナ(=仮想環境)にマウントしています。マウントとは共有フォルダのようなもので、このフォルダにホスト上でファイルやフォルダを配置すると、コンテナ上では `/home/snpcaster/notebook/project` に配置されたものとして扱われます。

6.3.2. コンテナ内に bash でアクセス

[起動コマンド実行](#)によりコンテナを起動します。

以下のコマンドを使って、実行に使用するコンテナの ID もしくは名前を確認します(どちらでも OK)。

```
docker ps
```

以下の通り、一番左の列が ID、一番右の列が名前です。

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
012c6b62d47e	snpcaster:0.9.0	"/usr/local/bin/entr..."	5 hours ago	Up 5 hours	0.0.0.0:59829->8888/tcp, [::]:59829->8888/tcp	snpcaster-0.9.0

以下のコマンドを実行するとコンテナ環境上で bash ターミナルが起動した状態となります。

※[コンテナ名 or ID]の箇所は上記で確認したものに差し替えてください。

```
docker exec -it [コンテナ名 or ID] bash
```

bash の画面に切り替わるので、以下のコマンドでマウントされたフォルダに移動します。

```
cd /home/snpcaster/notebook/project
```

先ほどホスト上で配置したフォルダ・ファイルがあるので、あとはお好きなコマンドを実行してください。

6.4. Docker イメージから実行

Docker イメージを使って実行する方法です。Docker イメージから一時的にコンテナが生成され、その中で指定したコマンドが実行できます。

6.4.1. データの配置

ホスト上(=お使いの PC の OS 環境)のお好きなフォルダに解析に必要なフォルダやリードデータを配置してください。

6.4.2. イメージ ID の確認

以下のコマンドを実行し、SNPcaster イメージの一覧を表示します。

```
docker images -f "reference=snpcaster"
```

TAG の列がバージョン番号なので、使用したい SNPcaster バージョンのイメージを一覧から探し、IMAGE ID をコピーします。

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
snpcaster	0.9.0	d457f9a73f20	5 hours ago	16.6GB

6.4.3. コマンドの実行

以下のコマンドでお好きなコマンドが実行可能です。

```
docker run -it --shm-size 8g --rm -v "{マウントするフォルダの絶対パス}:
/home/snpcaster/notebook/project" {イメージ ID} {実行したいコマンド+引
数}
```

※SNPcaster イメージには、コマンドライン実行用に
`/home/snpcaster/notebook/project` をデフォルトの実行フォルダに設定してあります(Dockerfile の WORKDIR 命令で指定)。
※そのため、上記のようにお好きなフォルダを
`/home/snpcaster/notebook/project` にマウントしていただくと、マウントしたフォルダに含まれるファイルが実行フォルダ直下に配置されている状態で実行可能です。

以下は実行例です。

マウントするフォルダの絶対パスを記載する必要がある都合上、Windows か Mac/Linux かで実行する際のコマンドが若干異なります(\$(\$PWD.Path)か \$(pwd)かの違いのみ)。

Windows(ターミナル/PowerShell)での snpcaster.sh の実行コマンド例

```
docker run -it --shm-size 8g --rm -v "
$(($PWD.Path))/new_analysis:/home/snpcaster/notebook/project"
d457f9a73f20 snpcaster.sh -i list.txt -r Sakai_BA000007.fasta
-f list_fastq.tsv
```

Mac/Linux の snpcaster.sh の実行コマンド例

```
docker run -it --shm-size 8g --rm -v
"$(pwd)/new_analysis:/home/snpcaster/notebook/project"
d457f9a73f20 snpcaster.sh -i list.txt -r Sakai_BA000007.fasta
-f list_fastq.tsv
```

また、上記の「実行したいコマンド」部分を「bash」とすることで、現在のワーキングディレクトリ以下の階層が`/home/snpcaster/notebook/project`にマウントされた状態で、snpcaster コンテナが起動します。

実行例

```
docker run -it --shm-size 8g --rm -v "(pwd):  
/home/snpcaster/notebook/project" {イメージ ID} bash
```

このようにすることで、TAB を用いた入力補完機能が働き、コマンド入力がスムーズにできます（上述の「`docker run` … 実行したいコマンド」の方法では、入力補完は使えません）。

終了する際には「exit」を入力します。

7. プログラム解説

7.1. SNPcaster

7.1.1. プログラムの利点

SNPcaster は、主に BactSNP、Snippy、Gubbins を組み合わせプログラムです。その主な利点は次の 3 点です；

1) 正確な SNP 抽出、2) 迅速な再解析が可能、3) 組換え領域除去等を自動的に行う。

まず、既存の SNP 抽出ソフトである BactSNP および Snippy の利点・欠点を挙げます。

表 7.1. BactSNP および Snippy の利点および欠点

	BactSNP	Snippy
利点	正確な SNP 抽出	再解析が迅速 Gubbins 用ファイルが自動的に作製される
欠点	・再解析時には 1 からやり直す必要がある ・Gubbins 用ファイルの作製に手間がかかる ・多数の株の処理でメモリ不足によるプログラム停止が発生することがある	データによっては、SNP が過剰に抽出される場合がある

表のように、BactSNP は、偽陽性 SNP の検出率が高く、正確性が高い SNP 抽出パイプラインです(1,2)。一方で、再解析時には全ての菌株について解析をやり直すため時間がかかります。また、Gubbins 等の別プログラムで使用するためには、適宜出力ファイルの編集をする必要があります。Snippy は、世界中で広く用いられているプログラムです (<https://github.com/tseemann/snippy>)。再解析時には、菌株ごとに生成されるフォルダを使用すれば、SNP 抽出済みの株の解析をスキップするため、高速な再解析が行えます。一方で、偽陽性 SNP が発生することが知られています(1,2)。SNPcaster では BactSNP の結果を、Snippy(中の snippy-core 機能)で解析できるように編集後、snippy-core で解析することによって、両者の利点を活かせるプログラムとなっています。デメリットは、BactSNP よりも解析時間が多少ですが長くなることです。しかし、全体の解析を自動化していますので、プログラムを動かすまでの準備および出力ファイルを整える手間を考えると、むしろ全体の解析時間は短くなると考えられます。

SNPcaster の利点としては次の点が挙げられます；

・BactSNP と同等の高い正確性の SNP が得られる。

- ・株ごとのサンプル SNP フォルダ(シードゲノムと vcf ファイルが入っている)を用いることで、再解析時には BactSNP を省略し、高速な再解析を行うことができる。
- ・自動化パイプラインによって、Gubbins 等による組換え領域の除去が自動で行える。
- ・上記の解析について、菌株リストのみを用意すれば自動で行える。

表 7.2.BactSNP, Snippy, および SNPcaster の比較

	BactSNP	Snippy	SNPcaster
SNP の正確性	○	△	○ (BactSNP と同等)
再解析時の迅速性	×	○	○
Gubbins 用データ の作製	△	○	○
解析の自動化	×	×	○
多数の株の処理	△	○	○

7.1.2. 解析の流れ

本プログラムは、BactSNP、Snippy、および Gubbins を中心的なプログラムとし、それらをスクリプトでつないでワークフロー化しています(図 7.1)。

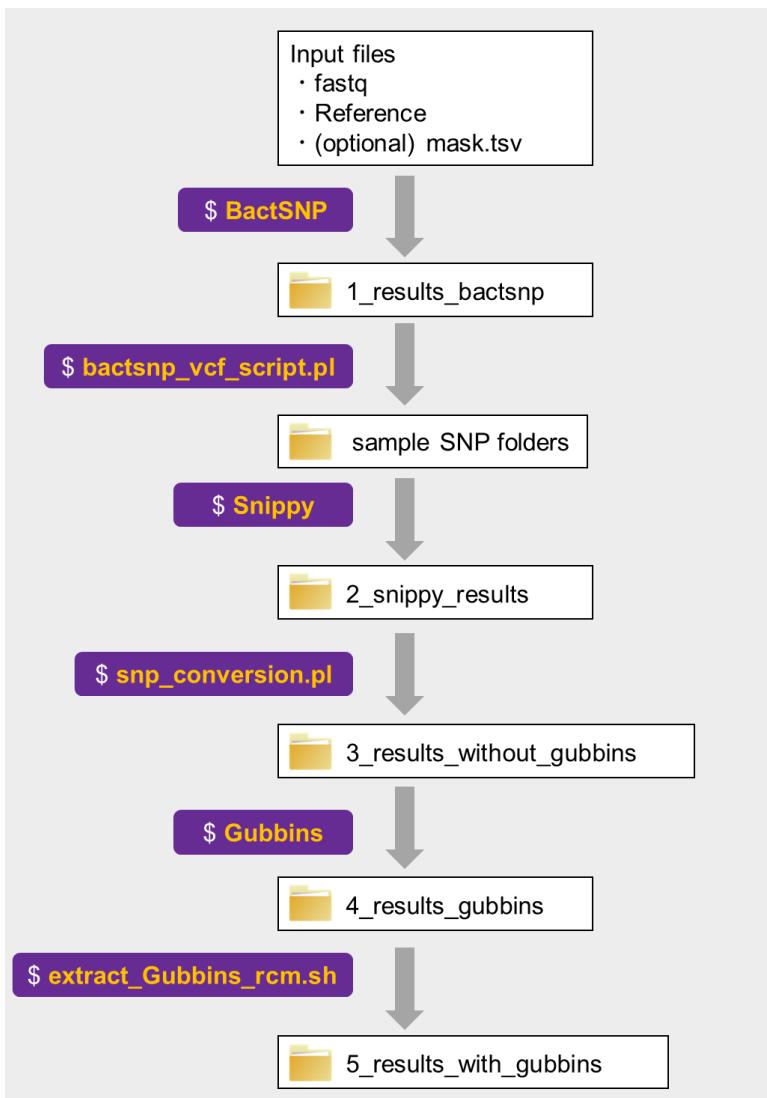
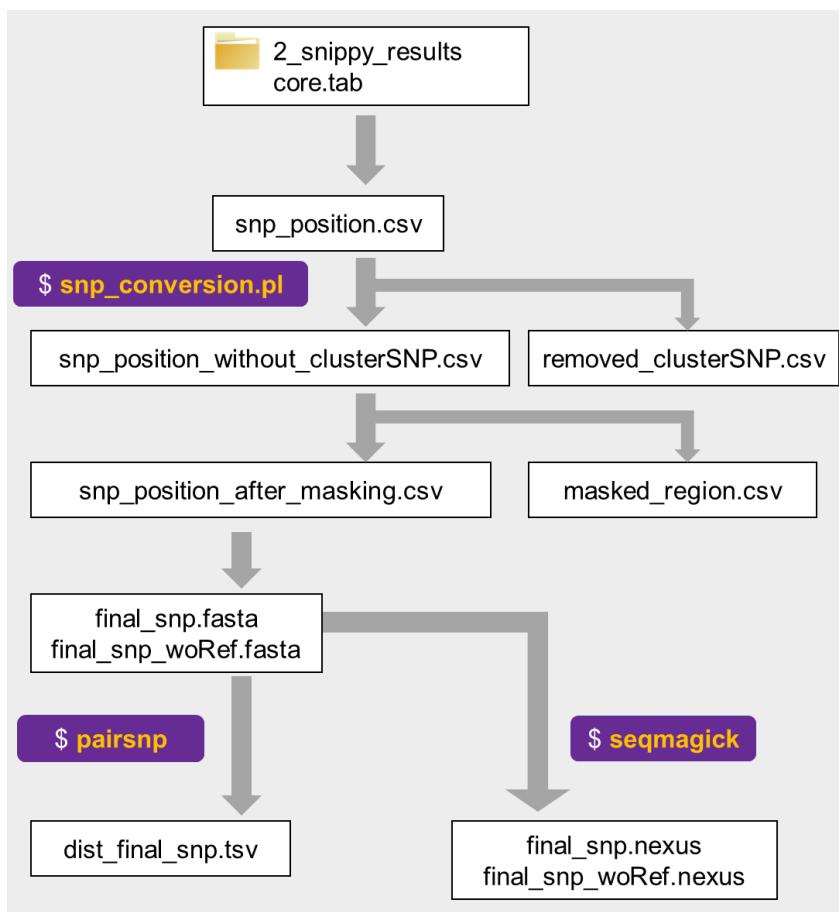


図 7.1. SNPcaster の流れ

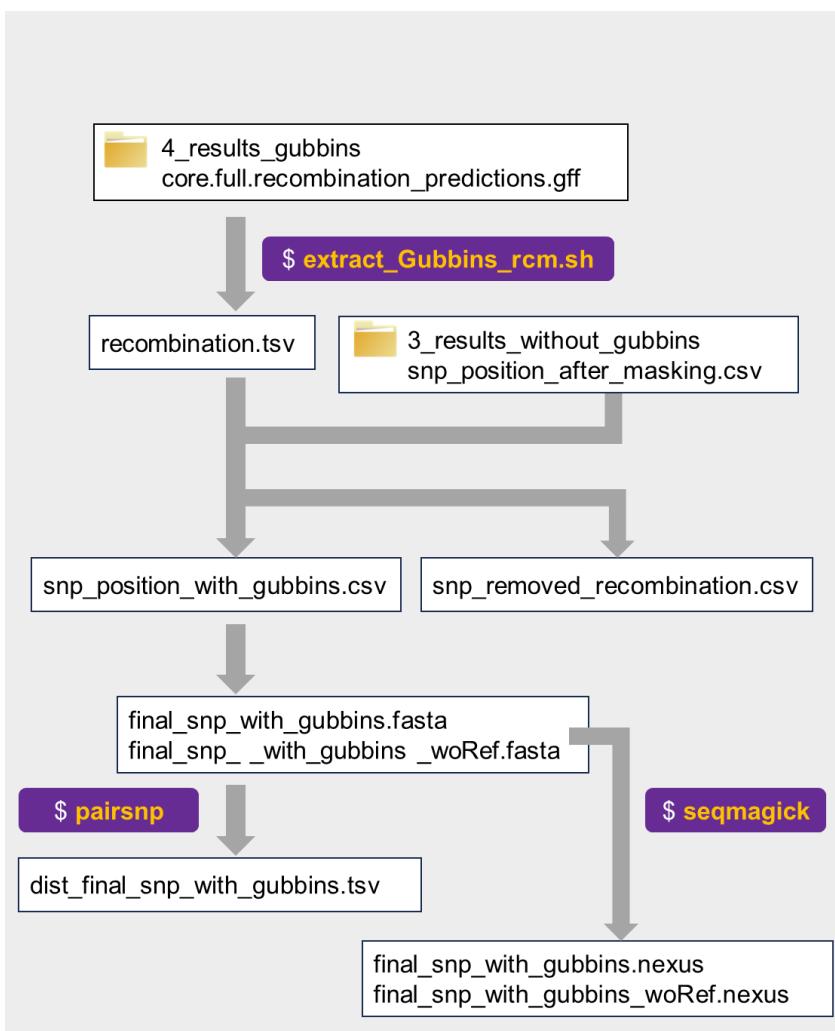
まず、BactSNP および Snippy(snippy-core)を行い、SNP を抽出します。各プログラムから出力されるファイルはそれぞれ、1_bactsnp_results および 2_snippy_results に格納されます。これらのファイルから、クラスターSNPやマスク領域を除去し(いずれもオプションとして指定した場合)、Gubbinsを行わない場合の結果ファイルが作製されます(図 7.2)。これらのファイルは、3_results_without_gubbins フォルダに格納されます。

図7.2. Snippy 後の
ファイル処理フロー



Gubbins による組換え領域の検出を行う場合、snippy で得られた core.full.aln をもとに gubbins が実行されます。Gubbins の結果ファイルは 4_results_gubbins フォルダに格納されます。この解析で得られた組換え領域の情報と、3_results_without_gubbins フォルダ中にある SNP の情報から、gubbins によって検出された組換え領域の除去を行います（図 7.3）。この結果、gubbins による組換え領域を除いた SNP 情報が 5_results_with_gubbins フォルダに格納されます。

図 7.3. Gubbins 後のファイル処理フロー



3_results_without_gubbins および 5_results_with_gubbins 中のファイルは、
相同なものが多くあります。例えば、final_snp_without_gubbins.fasta と
final_snp_with_gubbins.fasta です。これらのファイルを比べることで、gubbins による
組換え領域の除去がどのような影響を与えるか、容易に比べることができます。

7.1.3. 出力ファイル

SNPcaster を実行すると、snpcaster_[日付]_[時間]_[リスト名]のフォルダが作製されます。例えば、2025年3月9日の午前8時25分15秒に list というリストファイルを用いて解析を開始したとすると、「snpcaster_20250309_082515_list」というフォルダが作製されます。

フォルダ中には、次の**ファイル**が出力されます(**フォルダ**については次項で説明します)。

- SNPcaster report.txt:用いたパラメータやSNP抽出後の情報が記載されます(後述)
- 参照配列ファイル
- リストファイル
- マスク領域ファイル
- (bactsnp がスキップされた株がある場合=サンプルSNPフォルダがあった場合)
bactsnp_skip_list:スキップされた菌株のリスト
- (解析株の fastq ファイルがない場合)missing_list:指定された fastq ファイルがない株のリスト

SNPcaster report.txt ファイルには、次の情報が含まれます。

Version	SNPcaster プログラムのバージョン番号
Strain list	用いた菌株リストファイル名
Fastq list	用いた fastq リストファイル名
Mask file	マスク(解析で除く)領域のファイル名
Reference file	用いた参照配列ファイル名
Clustered SNP removal (bp)	クラスターSNPとして除去したSNPの間隔(-c オプションの値)
Threads	並列処理の値(-t オプションの値)
BactSNP allele-freq	BactSNPで用いるアリール頻度の閾値。例えば、0.9 の場合、マッピングされたショートリードの90%以上が当該SNPを支持しないとSNPとして検出されない (-a オプションの値)
BactSNP jobs	BactSNPの並列処理の値(-j オプションの値)
Recombinogenic region detection (Gubbins)	Gubbinsを行ったか否か(ON/OFF)
Core genome size (bp)	
Before Gubbins	Gubbins を行う前のコアゲノムサイズ。指定していれば、マスク領域も除かれています
After Gubbins	Gubbins を行った後のコアゲノムサイズ
Number of informative SNP sites (with reference)	
No masking	参照配列における SNP が存在する位置の数 (=final.snp.fasta における配列の長さ)
After cluster SNP removal	(クラスターSNP を指定した場合のみ)クラスターSNP を除去した後の SNP の数
After masking	(マスク領域を指定した場合のみ)マスク領域を除去した後の SNP の数
After Gubbins	(Gubbins を実行した場合のみ)Gubbins による組換え領域を除去した後の SNP の数
Number of informative SNP sites (without reference)	
No masking	上記の SNP の位置から、サンプル間で違がある部分のみの数
After cluster SNP removal	同上
After masking	同上
After Gubbins	同上
Removed SNPs	
Cluster SNP	クラスターSNP として除去された SNP の数
Masked region	マスク領域として除去された SNP の数
Gubbins	Gubbins による組換え領域として除去された SNP の数

結果フォルダに格納されている**フォルダ**は次の通りです。

- **サンプル名フォルダ**: サンプルごとの SNP 情報が含まれる。詳細は後述。
- **1_results_bactsnp**: BactSNP の結果ファイル。実行されなければ作製されない。
詳細は <https://github.com/IEkAdN/BactSNP>
- **2_snippy_results**: Snippy の結果ファイル
詳細は <https://github.com/tseemann/snippy>
- **3_results_without_gubbins**: Gubbins による組換え領域の除去を行わなかった場合の結果ファイルが含まれる。詳細は後述。
- **4_results_gubbins**: Gubbins の結果ファイル
詳細は <https://github.com/nickjcroucher/gubbins>
- **5_results_with_gubbins**: Gubbins による組換え領域の除去を行った場合の結果ファイルが含まれる。詳細は後述。

サンプル名フォルダ

本フォルダには、BactSNP によって得られた結果を、snippy-core で利用するために変換した pseudogenome ファイル(snps_aligned.fa)と、SNP 範囲を示したファイル(snps.vcf)が含まれます。再解析時には、本フォルダを解析するディレクトリに置いておくと、BactSNP 部分が省略されるので、時間の短縮になります。

3_results_without_gubbins

本フォルダには、次のファイルが含まれます。系統解析に用いるのは final.snp が付くファイルです。他のファイルは必要に応じて確認してください。

SNP 配列。コアゲノム中で違いのある部分のみを集めたものです。系統解析等にはこれらを使います。	
final.snp.fasta	FASTA 形式。参照配列あり。
final.snp.nex	NEXUS 形式。参照配列あり。
final.snp_woRef.fasta	FASTA 形式。参照配列なし。
final.snp_woRef.nex	NEXUS 形式。参照配列なし。
次のファイルは SNP の位置や、株間の SNP 数が記載されています。必要に応じて確認してください	
core.full.fasta	コアゲノム領域(core_region.tsv)の配列。
core_region.tsv	コアゲノム領域(全株に共通する領域)。
core_region_after_masking.tsv	マスク処理後のコアゲノム領域。このファイルは、マスクされた領域(mask.tsv)を core_region.tsv から除去して作製されます。
core_region_final.tsv	各種処理後のコアゲノム領域です。マスクが有効な場合は core_region_after_masking.tsv と同じ内容、そうでなければ core_region.tsv と同じ内容です。
dist_final.snp.tsv	ペアワイズ SNP 距離(株間の SNP の数)。
dist_final.snp_matrix.tsv	dist_final.snp.tsv のマトリックス形式。
snp_position.csv	snippy によって生成された cgSNP。このファイルの情報は、snippy が生成する「core.tab」ファイルと同じです。
snp_position_sample_only.csv	snp_position.csv から抽出された、サンプル間で塩基が異なる SNP のリスト。
snp_position_without_clusterSNP.csv	クラスター-SNP を除いた cgSNP。クラスター-SNP 距離が 0 に設定されている場合、このファイルは作製されません。
snp_position_without_clusterSNP_sample_only.csv	snp_position_without_clusterSNP.csv から抽出された、サンプル間で塩基が異なる SNP のリスト。
removed_clusterSNP.csv	除去されたクラスター-SNP。クラスター-SNP 距離が 0 に設定されている場合、このファイルは作製されません。
snp_position_after_masking.csv	クラスターおよびマスクされた領域を除いた cgSNP。このファイルの SNP は final.snp.fasta と同じです。
snp_position_after_masking_sample_only.csv	snp_position_after_masking.csv から抽出された、サンプル間で塩基が異なる SNP のリスト。
masked_region.csv	マスクされた領域で除去された cgSNP。
snp_position_final.csv	クラスター-SNP の除去とマスク処理が適用された後の cgSNP。

5_results_with_gubbins

本フォルダには、次のファイルが含まれます。系統解析に用いるは final.snp が付くファイルです。他のファイルは必要に応じて確認してください。3_results_without_gubbins 内のファイルと相同のものが多く含まれています。

SNP 配列	
final.snp_after_gubbins.fasta	FASTA 形式。参照配列あり。
final.snp_after_gubbins.nex	NEXUS 形式。参照配列あり。
final.snp_after_gubbins_woRef.fasta	FASTA 形式。参照配列なし。
final.snp_after_gubbins_woRef.nex	NEXUS 形式。参照配列なし。
その他ファイル	
core_summary_with_gubbins.tsv	コアゲノム領域。
dist_final.snp_with_gubbins.tsv	ペアワイズ SNP 距離。
dist_final.snp_matrix_without_recombination.tsv	dist_final.snp_with_gubbins.tsv のマトリックス形式。
recombination_region.csv	Gubbins によって組換え領域として除去された cgSNP。
snp_position_after_gubbins.csv	クラスター、マスク領域、Gubbins で検出された組換え領域の SNP を除いた SNP 位置。このファイルの SNP は final.snp_with_gubbins.fasta と同じです。
snp_position_after_gubbins_sample_only.csv	snp_position_with_gubbins.csv から抽出された、サンプル間で塩基が異なる SNP のリスト。

引用文献

- 7.1. Yoshimura D., Kajitani R., Gotoh Y., Katahira K., Okuno M., Ogura Y., Hayashi T., and Itoh T. Evaluation of SNP calling methods for closely related bacterial isolates and a novel high-accuracy pipeline: BactSNP. 2019. Microbial genomics 5.
- 7.2. Derelle R., von Wachsmann J., Maklin T., Hellewell J., Russell T., Lalvani A., Chindelevitch L., Croucher N. J., Harris S. R., and Lees J. A. Seamless, rapid, and accurate analyses of outbreak genomic data using split k-mer analysis. 2024. Genome Res. 34:1661-1673.

7.2. grape_qc_assembly

本プログラムでは、ショートリード配列をトリミング（省略可）後、データ量（カバレッジなど）を計算します。同時に SPAdes または SKESA を用いて de novo assembly を行い、ドラフトゲノムの作製を行います。得られたドラフトゲノムをもとに、checkM および QUAST を行い、コンタミネーションの有無などの判定を行います。設定ファイル（configuration file）を使用した場合には、指定したカバレッジ以上のデータでのみアセンブリを行い、クオリティチェック（QC）を行った結果を出力します。

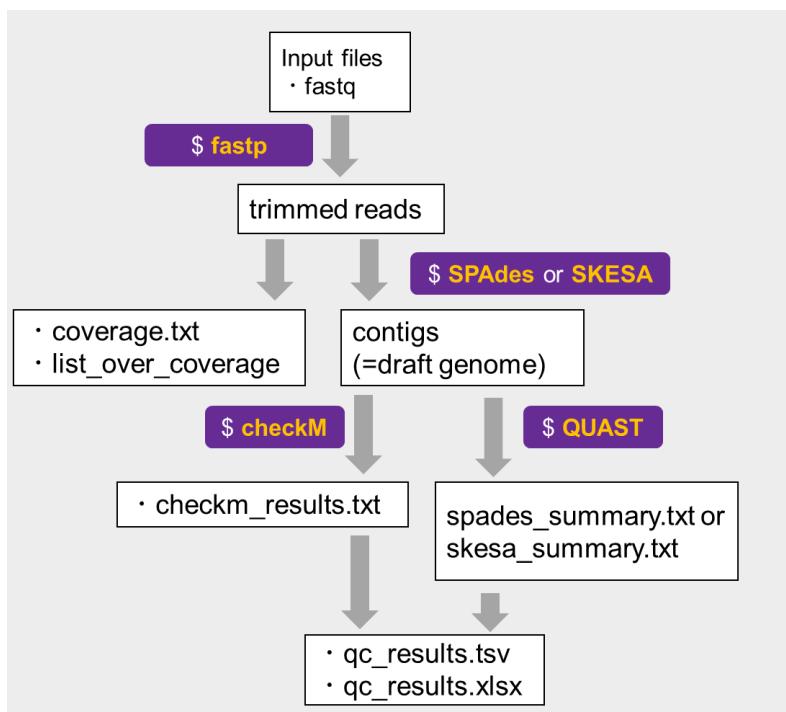


図 7.4.
grape_qc_assembly の
処理フロー

作製されるファイルは次の通りです。

[株名].fasta	ドラフトゲノム
assembly_summary.tsv	コンティグの長さなどの統計（QUASTで計算）をまとめたもの。
coverage.txt	カバレッジ一覧
list_assembly.tsv	list_over_coverageに含まれる菌株のFASTQファイルリスト。
list_over_coverage	設定ファイルで指定したカバレッジを上回った株のリスト。このリストにある株のみをアセンブルする。
qc_results.tsv	QUAST, CheckMの結果（ゲノムサイズ、コンタミネーションなど）から設定ファイルの範囲内にある場合にpass, 外れている場合にfailを記載する。全ての値でpassの場合のみ、qc_results(総合判定)の列がPASSとなる。
qc_results.xlsx	qc_results.tsvのエクセル版（内容は同じ）。

「cleanup=0」を指定すると、中間ファイルを含む次のフォルダが削除されずに残ります。

- fastp フォルダ : fastp でトリミングした以下の fastq ファイル
 - [菌株名]_1(2).fastq.gz: ペアが見つかった Read1 もしくは 2 のリードファイル
 - [菌株名]_u1(u2).fastq.gz: ペアが見つからなかった Read1 もしくは 2 のリードファイル
- quast フォルダ : QUAST の実行結果
 - 出力の詳細は下記をご覧ください。
<https://github.com/ablab/quast#output>
- checkm フォルダ : CheckM の実行結果
 - 出力の詳細は下記をご覧ください。
<https://github.com/Ecogenomics/CheckM>

8. トラブルシューティング、Q&A

※解析法や結果の解釈については、菌種によって異なります。菌種内でも系統によって異なる場合もあります。以下の例は大腸菌(特に腸管出血性大腸菌)では検証済みのものですが、他の菌種では異なる場合があることをご留意ください。

アセンブリ・QC プログラム

Q1. QC の基準はありますか？

A1. 統一的な基準はありません。CDC の PulseNet が公表している SOP では一定の基準が示されています。

[https://www.aphl.org/programs/global health/Pages/PulseNet-International-SOPs.aspx](https://www.aphl.org/programs/global-health/Pages/PulseNet-International-SOPs.aspx)

Q. アセンブラーは SPAdes と SKESA のどちらが良いですか？

A. 大きな違いはありません。SPAdes の方が長いコンティグが得られやすく、SKESA の方がコンティグの正確性が高いと言われています。詳しくは、次のページをご参照ください。

<https://github.com/rrwick/Trycycler/wiki/Guide-to-bacterial-genome-assembly>

SNP 解析

Q. SNP と SNV の違いは？

A. SNP は single nucleotide polymorphism、SNV は single nucleotide variation の略です。SNP は集団内に一定頻度(数%)で存在する変異、SNV は変異を広く指す用語です。厳密な定義は異なりますが、微生物ゲノムにおいてはほぼ同義で用いられることが多いです。

Q. clustered SNP とは何ですか？

A. SNP 間の距離が短い SNP のことです。例えば、ある SNP から 10 bp 以内に別の SNP が存在する場合を指します。この距離が近い場合、変異が独立して起こったのではなく、組換えによって一度に変化した可能性があります。距離についての基準はありませんが、解析から除外する場合があります。

CTACCGTGAGCTGGGGGGATCTCTACTCCAACG
CTACCGTGAGCTGGGGATCTCTACTCCAACG

Clustered SNP を除くかどうか、そして何 bp 以内の clustered SNP を除くかについては、菌種ごとに検討する必要があります。

大腸菌での集団感染の調査においては、ごく稀(数%)にしか起こりません。

そのため、まずは cluster SNP を除かずに解析し、疫学的に近いはずだが多数の SNP が認められる場合に除いてみる、という方針が良いと思われます。

Q. Gubbins は必要ですか？

A. Gubbins は組換え領域の除去に用いています。使用するかどうかは、cluster SNP と同様、菌種ごとに検討する必要があります。

大腸菌での集団感染の調査においては、Gubbins の使用で差が出ることはほとんどありません(1%以下)。

そのため、上述の cluster SNP と同様に、まずは用いずに解析し、疫学的に近いはずだが多数の SNP が認められる場合に使用してみる、という方針が良いと思われます。

Q. 参照配列の選び方は？

A. 近縁な株(例. 集団感染株のうちの 1 株)を用いた方が多数の SNP が検出されます。コアゲノムサイズ(SNP を検出する領域)が大きくなるためです。

参照配列の選定の際には、SNPcaster report で出力されるコアゲノムサイズも参考になります。例えば、ゲノムサイズが 5.5Mb の大腸菌の場合で、コアゲノムサイズが 2Mb だった場合、近い株を参照配列とすることでコアゲノムサイズが 4Mb 以上となることがあります。その場合、理論上 SNP の数は 2 倍以上になります。

Q. raxml-ng が動きません。

A. パス名が長すぎると正常にプログラムが動作しないことがあります。ファイルを移動するか、解析する SNP ファイルがあるフォルダ名を変更してください。

Q. なぜか動かない

A. Windows で作ったファイルが混ざっていると、改行コードが違うので動かない場合があります(特に菌株リスト)。その場合は、ファイルを Linux または Mac で作り直すか、下記コマンドを実行して改行コードを変更してください。

```
nkf -d --overwrite [ファイル名]
```

また、フォルダ名に全角文字が入っていると動作しない可能性があります。

Q. fastq ファイルが読み込まれない

A. SNPcaster では、イルミナシークエンサーから得られたペアエンドデータのみを入力ファイルとしています。また、ファイル名は次のような様式に合わせる必要があります。

イルミナフォーマット(e.g., strain_S00_L001_R1_001.fastq.gz)

シンプルフォーマット(e.g., strain_R1.fastq.gz)

Q. JupyterLab で解析中表示[*]がずっと続きます。

A. 一度実行を止めて、再度プログラムを実行してみてください。多数の株を解析していた場合は、株数を減らして実行してみてください。

9. 資料

9.1. Dockerについて

Docker とは、コンピューター内で仮想環境を作製、配布、実行するためのプラットフォームです。仮想環境とは、「1 つのコンピューターの中に、あたかも別のコンピューターが動いている」ようにすることです。従来の仮想環境構築には VMware や VirtualBox 等が使われていましたが、これらは各仮想マシン内に独立した OS を 1 から起動するため、起動が遅く、また、OS のすべてのデータが必要なのでファイルサイズが巨大になるという欠点があります。一方、Docker では Docker Engine というプログラムを介して、仮想環境が必要とする OS の機能を親となる OS(ホスト OS)から借りて仮想環境を動作させます。これにより、OS 処理の大部分を仮想環境が独自に動かす必要がなくなるため、Docker は起動が速く仮想環境のデータも軽量です。Docker はあらかじめ設定した複数のプログラムを動かすことができます。この複数のプログラムがインストールされた仮想環境をコンテナと呼びます。

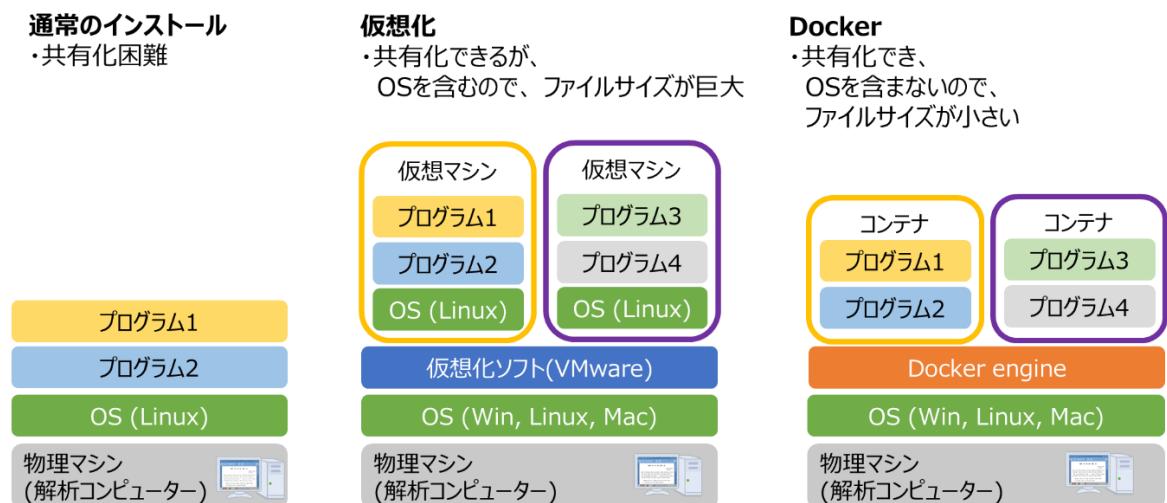


図 9.1. Dockerと仮想化の違い

Docker のコンテナを作製するには、イメージが必要です。コンテナとの違いが紛らわしいですが、イメージは設計図、コンテナは実際に PC 上で動いている仮想環境の実体、と考えることができます。イメージは Docker Hub という専用サーバーで世界に向けて共有が可能で、Python や R などがインストール済みのイメージなど、様々な企業・組織・個人が作製したイメージが公開されています。

SNPcaster では docker compose up コマンドで、イメージの作製→コンテナの作製→コンテナの起動の 3 段階を行います。イメージの作製時は BactSNP など様々なツールのインストール処理を行い、必要なプログラムすべてがインストールされた状態で 1 つのイメージファイルとして保存されます。初回起動時のみ、イメージ作製に時間がかかりますが、一度イメージができてしまえば、以降はイメージからコンテナを作製するだけなので 2 回目以降の起動は高速になります。

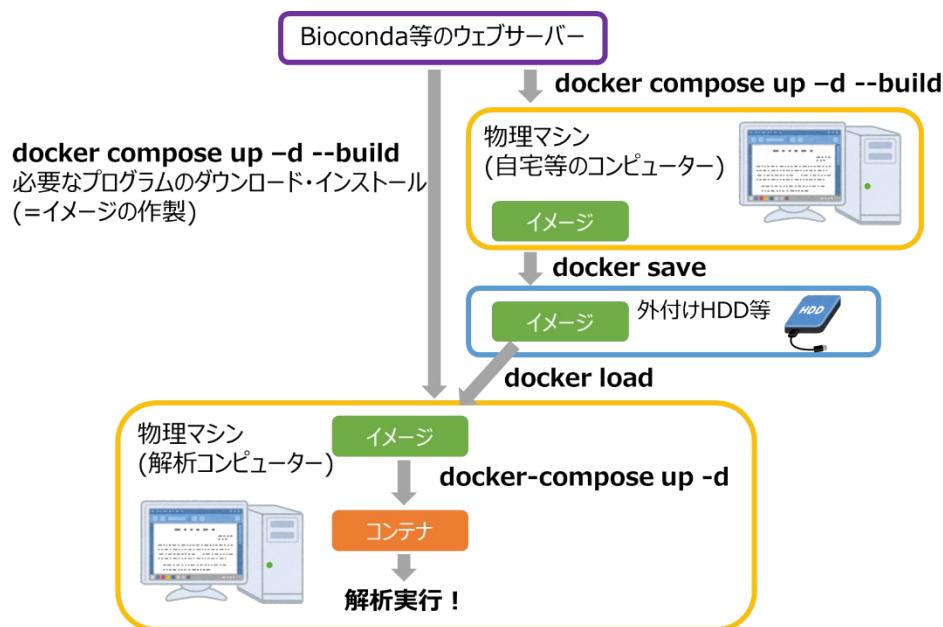


図 9.2. SNPcaster におけるイメージの作製から解析実行までの手順

docker compose コマンドによって、外部サーバーから必要なソフトをインストールします。プロキシ等、セキュリティ環境が厳しい場合には、セキュリティが厳しくない端末でイメージを作製し、外付けHDD等を用いてインポートすることも可能です(図中右側)。

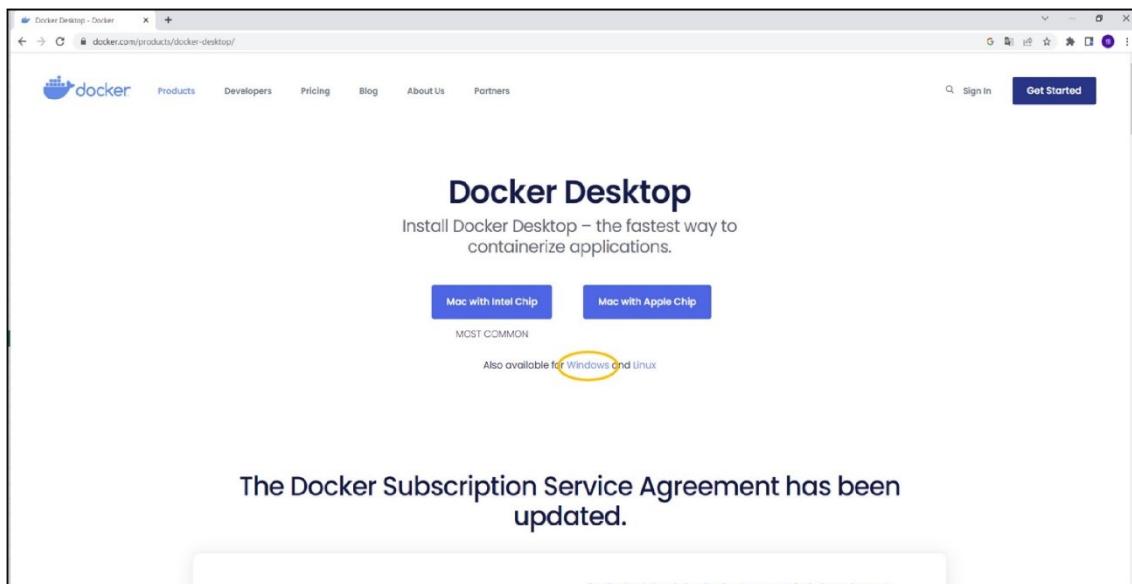
9.2. Docker Desktop のインストール方法

Docker Desktop の有償対象にならない、もしくは、すでにライセンス購入済みの方は以下の手順でインストールしてください。

9.2.1. Windows

9.2.1.1. 下記サイトから、DockerDesktop をダウンロードします。下記サイトの Widnows のリンクをクリックすると自動的にインストーラーがダウンロードされます。

URL:<https://www.docker.com/products/docker-desktop/>



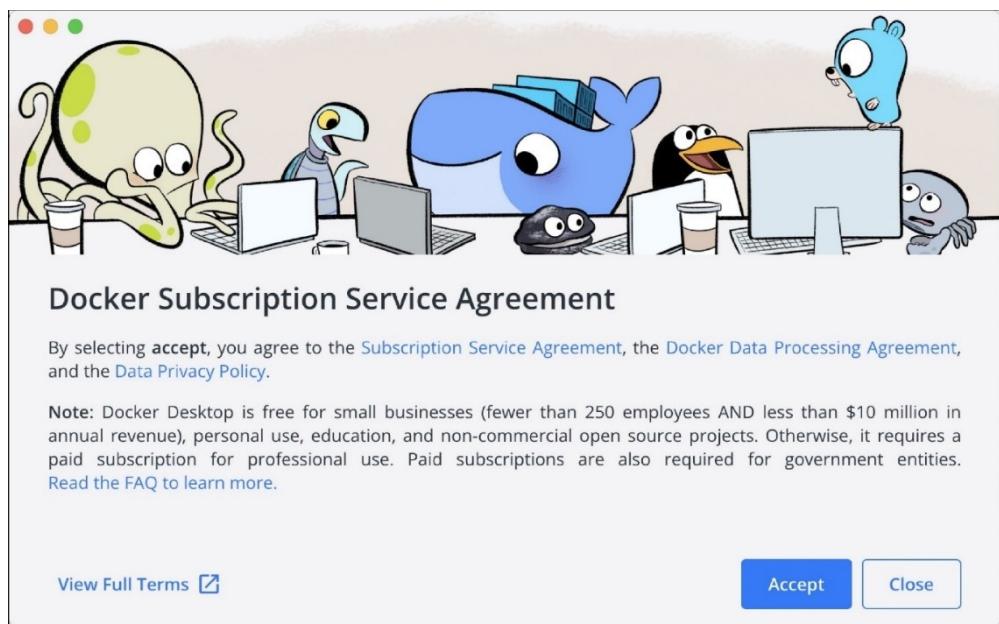
9.2.1.2. ダウンロードしたインストーラーをダブルクリックし、メッセージに従いインストールします。

インストール時に「WSL 2 installation is incomplete」というエラーが出た場合は、以下の URL を Web ブラウザのアドレスバーへ入力して、ダウンロードしたファイルを実行してください。

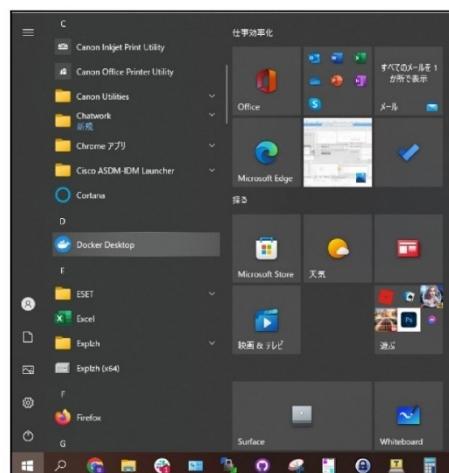
URL:

https://wslstorage.blob.core.windows.net/wslblob/wsl_update_x64.msi

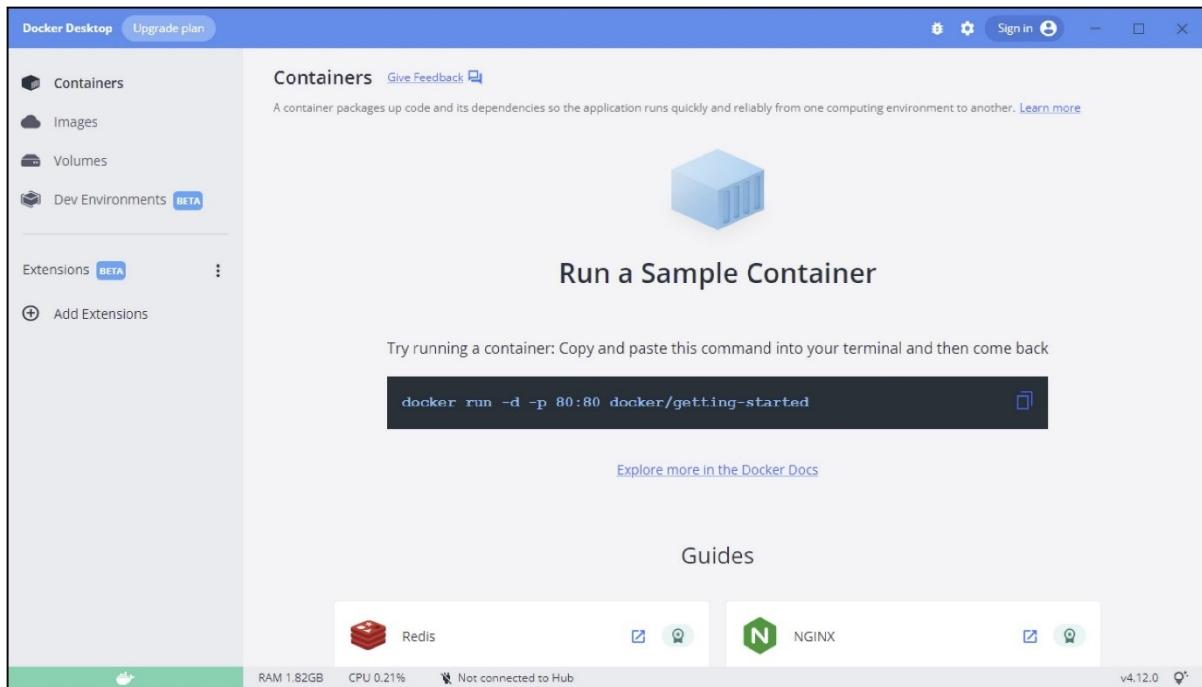
9.2.1.3. インストールが完了したら、下記画面が起動しますので、Docker のサービス内容をご確認いただき、承認して下さい。



※上記画面が自動的に起動しない場合、左下のウインドウズマークをクリックし、
プログラムの一覧から「DockerDesktop」をクリックしてください。



9.2.1.4. この状態でインストールした Docker Desktop を起動します。下図の画面が表示されましたら、DockerDesktop のインストールは完了です。



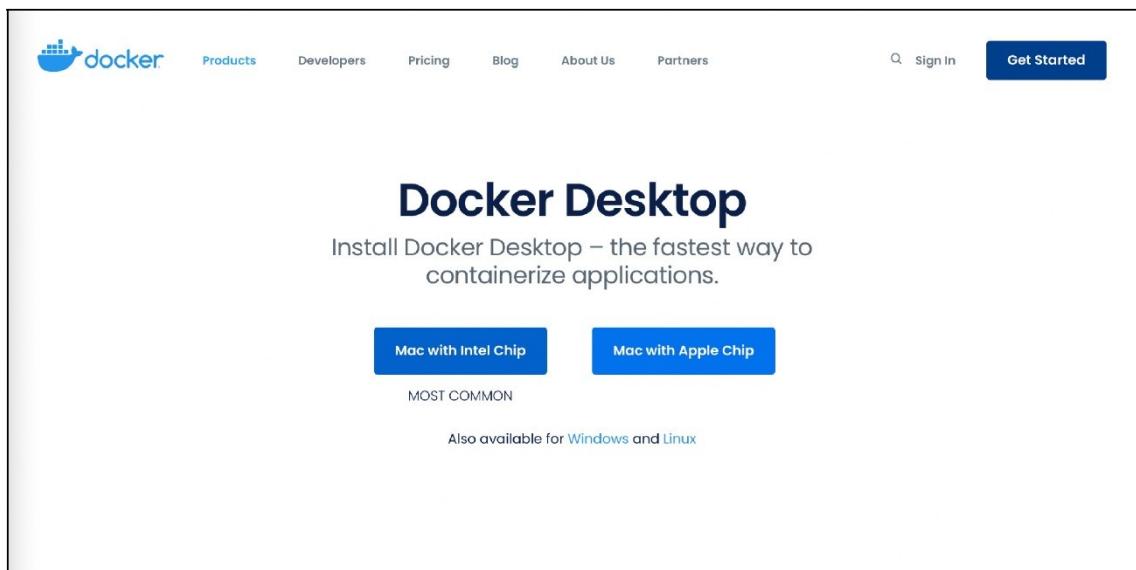
9.2.2. Mac

- 9.2.2.1. デスクトップ画面上部のアップルマーク→「この mac について」を順番にクリックし、チップの種類が intel か Apple (M1、M2 等)かを確認します。

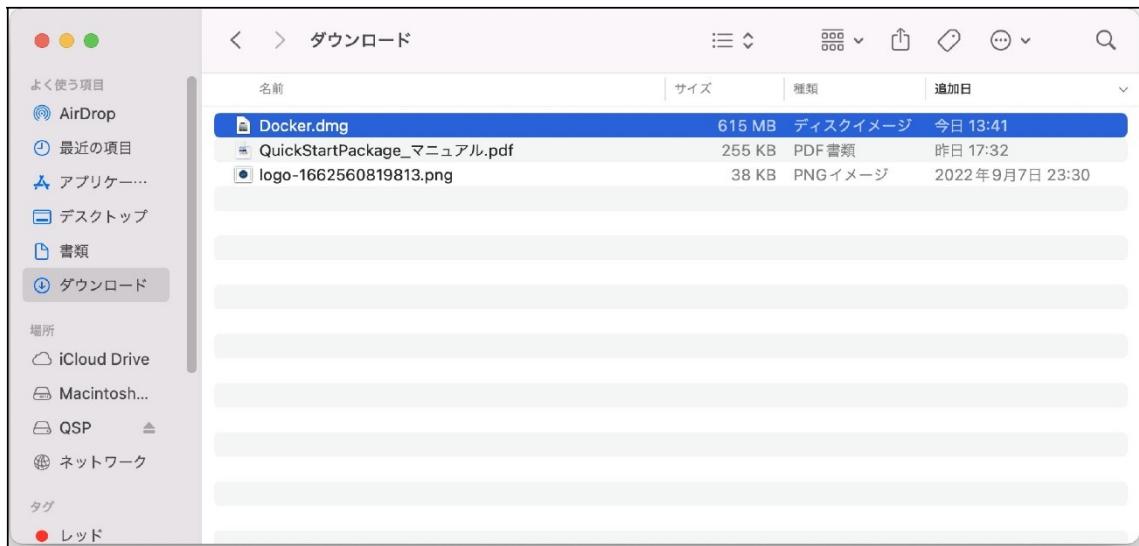


- 9.2.2.2. 下記サイトから、チェックしたチップ用の DockerDesktop をダウンロードします。

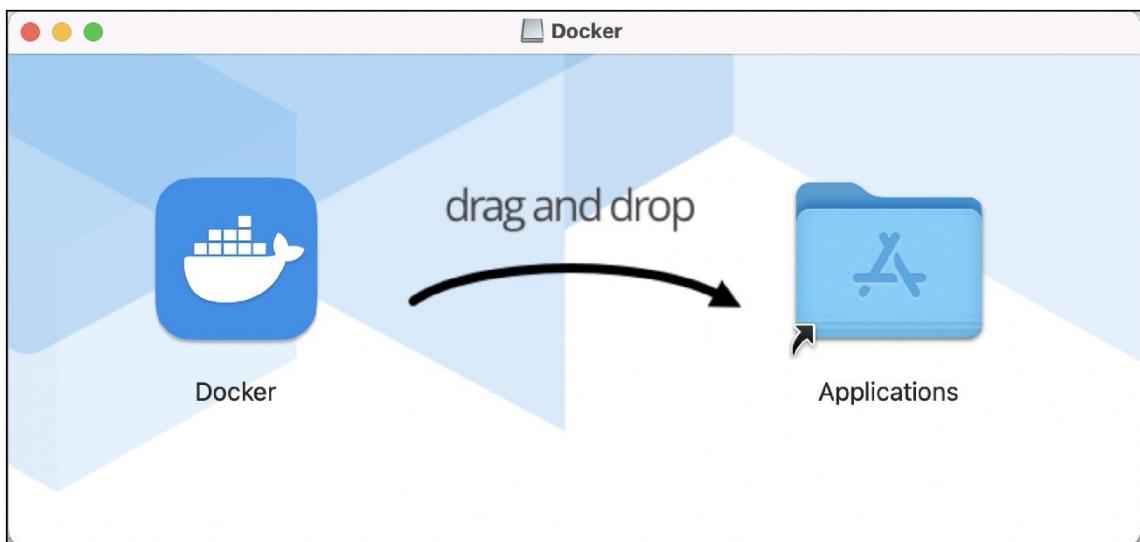
URL:<https://www.docker.com/products/docker-desktop/>



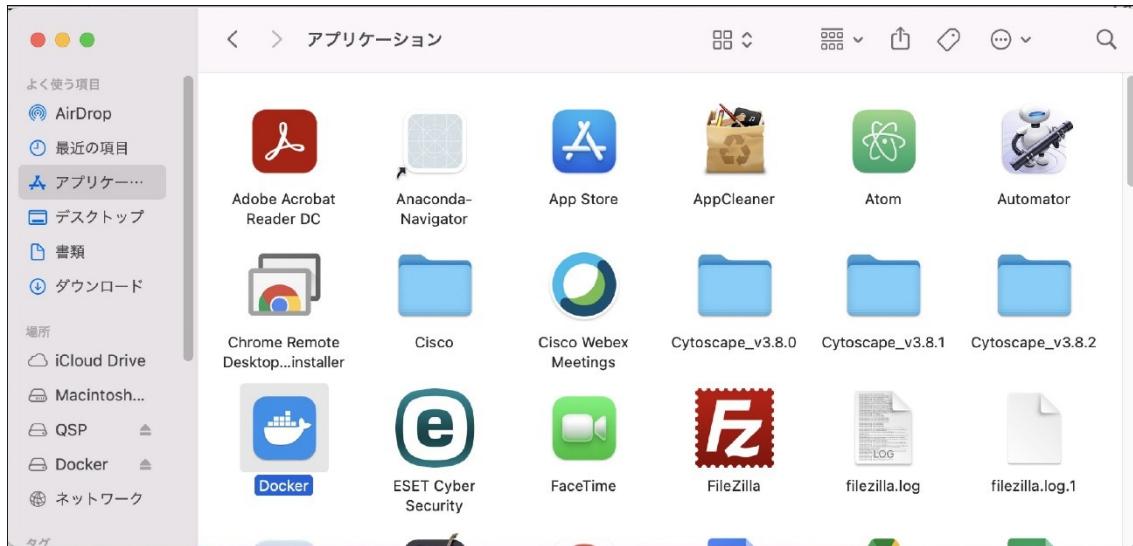
9.2.2.3. ダウンロードしたインストーラーをダブルクリックし、メッセージに従いインストールします。



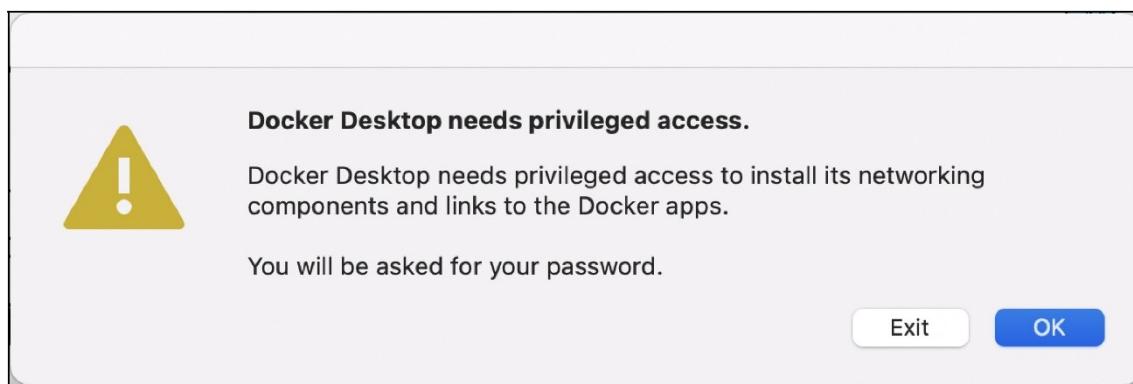
Mac の場合、Docker アイコンを Applications アイコンへドラッグ＆ドロップします。



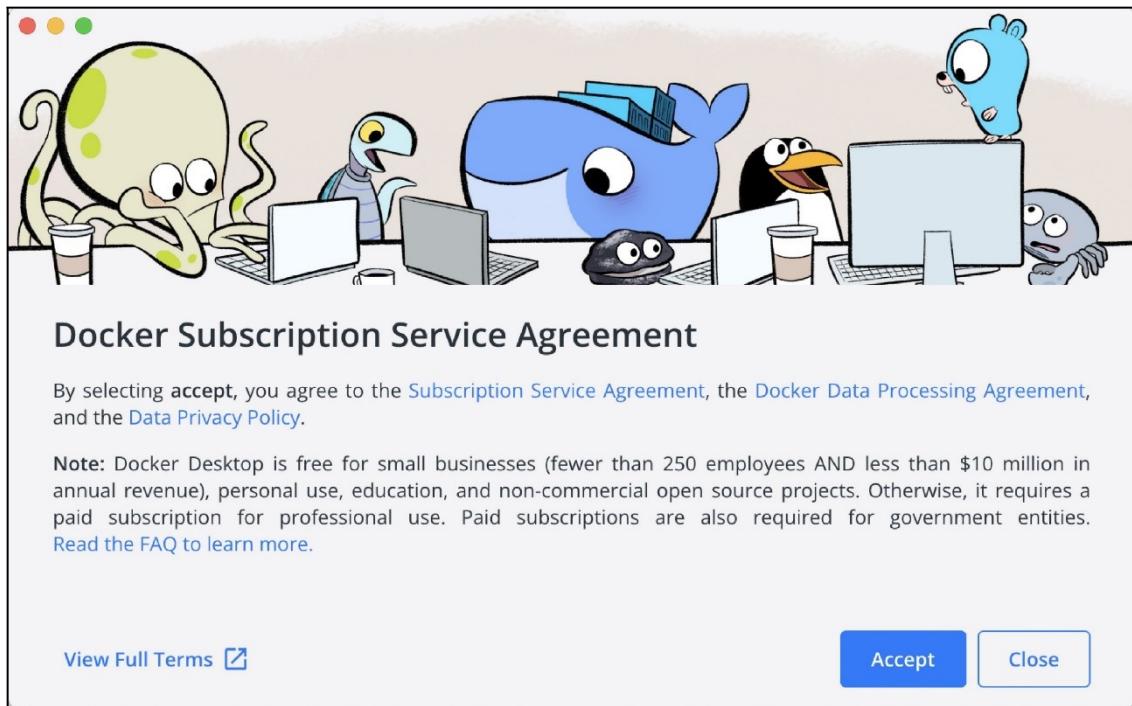
9.2.2.4. Finder でアプリケーションフォルダを開き、Docker アイコンをダブルクリックします。



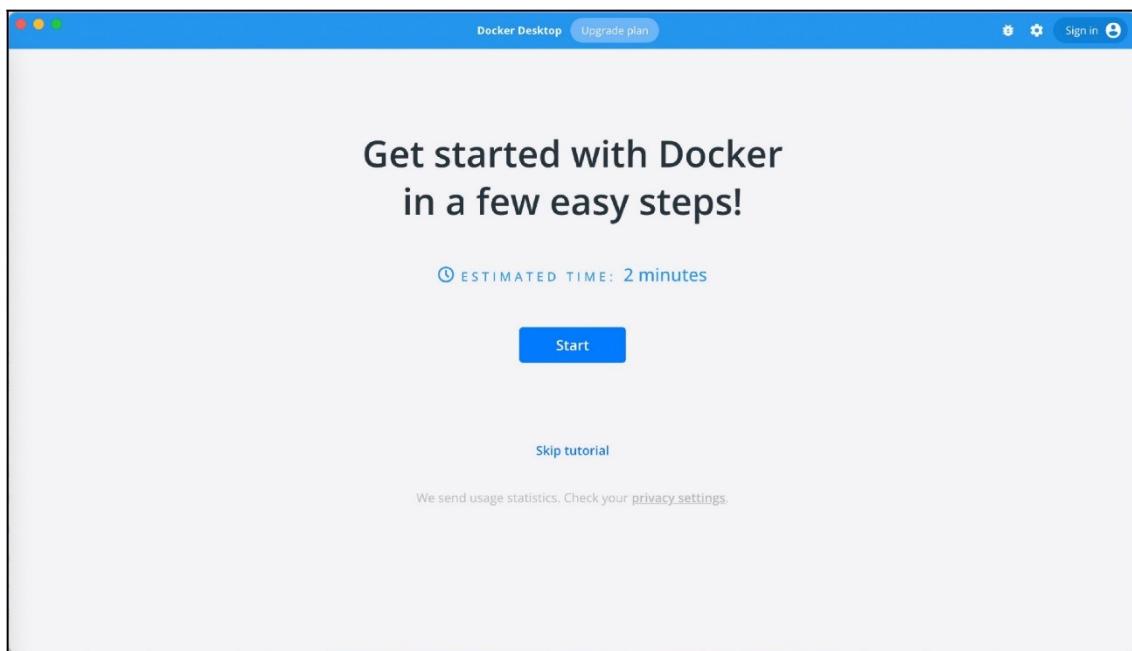
特権的なアクセスが必要というメッセージが表示された場合、"OK"をクリックして、特権 アクセスを許可して下さい。



9.2.2.5. Docker のサービス内容をご確認いただき、承認して下さい。



9.2.2.6. 下図の画面が表示されましたら、DockerDesktop のインストールは完了です。



9.3. Docker Desktop のアンインストール方法

この章では、Docker Desktop を削除したい場合の手順を説明します。

9.3.1. 作成済みのコンテナ・イメージのバックアップ

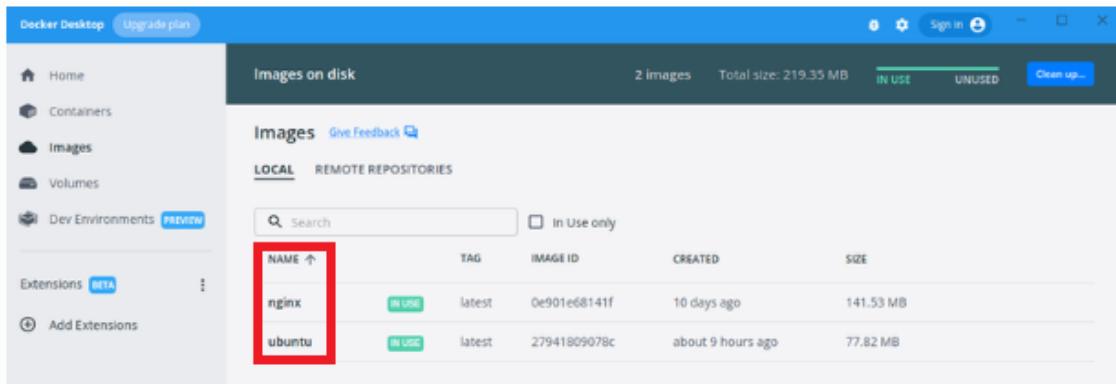
作成済みの Docker コンテナやイメージは、バックアップを作成すれば Rancher Desktop や Docker Engine に移行後も引き続き使用できます。移行後も継続して使用したいコンテナ・イメージがある場合は、この項の手順でバックアップを作成してください。

※Docker コンテナとイメージの関係については、[Dockerについて](#)をご覧ください。

※SNPcaster については、コンテナ起動後にご自身で追加のパッケージ等をインストールしないなければ、名称: **snpcaster:バージョン番号** の Docker イメージのみバックアップしていただければ十分です。方法は、次のステップ [9.3.1.1](#) をご覧ください。

9.3.1.1. Docker イメージをバックアップする場合、まずはバックアップするイメージの名前を確認します。

Docker Desktop を起動し、左側のパネルで **Images** を選択すると Docker イメージ一覧が表示されます。Name 列がイメージの名称です。バックアップしたいイメージの名前を確認してください。



ターミナルを起動し、以下のコマンドでバックアップを保存したいフォルダに移動します。

```
cd {バックアップを保存したいフォルダパス}
```

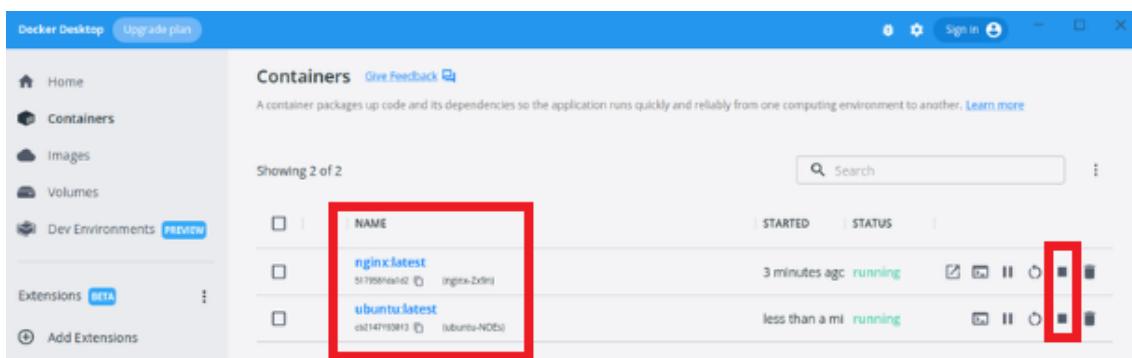
続けて、以下のコマンドを実行すると、移動したフォルダ直下にバックアップファイルが作成されます。※この処理は時間がかかる場合があります。

```
docker save {イメージ名} > {任意の名前}.tar
```

実行が完了すれば、1つのイメージのバックアップは完了です。
複数ある場合はこれを繰り返してください。

- 9.3.1.2. Docker コンテナのバックアップを取りたい場合は、まずはバックアップするコンテナを停止します。

Docker Desktop を起動し、左側パネルの **Containers** を選択します。
Name 列がコンテナの名称なので、バックアップを取りたいコンテナ名を確認します。
バックアップを取りたいコンテナの右側にある■をクリックするとコンテナが停止します。



次に、以下のコマンドで停止したコンテナをイメージ化します。
{任意の作成するイメージ名}には、お好きな名前を指定できます。

```
docker commit {コンテナ名} {任意の作成するイメージ名}
```

これでコンテナがイメージ化されたので、前項 [9.3.1.1](#) の手順でバックアップファイルを作成してください。

※ご注意※

Docker コンテナ起動時にホスト側(=Windows/Mac 側)とコンテナ側が共有しているフォルダ(ボリュームマウントと呼びます)はこの手順ではバックアップされません。

ボリュームマウントしているフォルダは Docker Desktop をアンインストールしても削除されないので問題ありませんが、残しておきたいデータを誤って手作業で削除しないようにご注意ください。

※SNPcaster コンテナは、project フォルダがボリュームマウントされています。

※詳細は、以下の参考情報をご覧ください。

参考: <https://docs.docker.jp/desktop/backup-and-restore.html>

9.3.2. Docker Desktop のアンインストール

バックアップが完了したら、以下の公式サイトに記載された手順で Docker Desktop をアンインストールしてください。

- **Windows:** <https://docs.docker.jp/desktop/install/windows-install.html#id9>
- **Mac:** <https://docs.docker.jp/desktop/install/mac-install.html#docker-desktop>

9.3.3. バックアップの復元

[Docker のインストール](#)に記載のいずれかの方法で Docker をインストールした後、先ほど作成した Docker イメージのバックアップを復元していきます。

9.3.3.1. Docker コンテナのバックアップを取りたい場合は、まずはバックアップするコンテナを停止します

ターミナルを起動し、以下のコマンドでバックアップを保存したフォルダに移動します。

```
cd {バックアップを保存したいフォルダパス}
```

続けて、以下のコマンドを実行すると、バックアップされていた Docker イメージが復元されます。※この処理は時間がかかる場合があります。

```
docker load -i {バックアップファイル名}.tar
```

実行が完了すると、以下のコマンドで復元されたイメージが確認できます。

```
docker image list
```

9.4. Rancher Desktop のアンインストール方法

Rancher Desktop をアンインストールしたい場合、以下の手順で行ってください。

9.4.1. Windows

Windows でアンインストールする場合、以下の手順で行ってください。

9.4.1.1. Windows ボタンをクリックし、検索窓に「インストールされているアプリ」と入力します。

インストールされているアプリが表示されるのでクリックします。

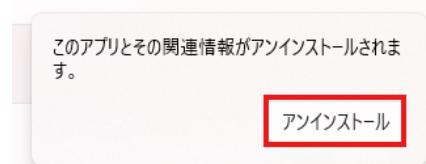


9.4.1.2. 画面上部の検索窓に「Rancher」と入力します。

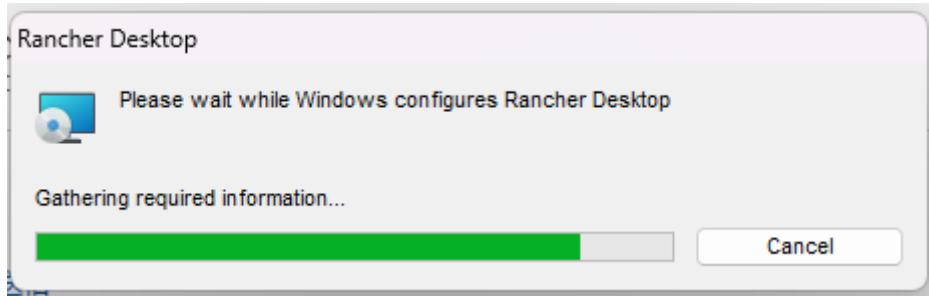
Rancher Desktop が表示されるので、右側にある…をクリックし、アンインストールをクリックします。



確認が出るので、アンインストールをクリックします。



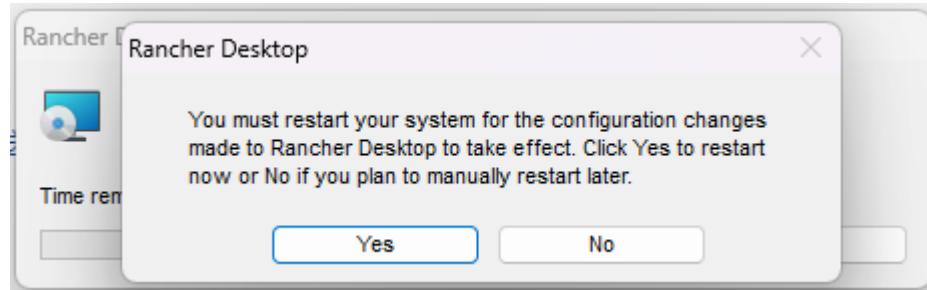
アンインストールが始まります。



途中、「このアプリが変更を加えるのを許可しますか？」という確認が出るので、はいを選択します。

アンインストールが終了すると、再起動を求めるダイアログが表示されます。

すぐに再起動できる場合は Yes を、後で自分で再起動したい場合は No を選択します。



再起動が実行されると、Rancher Desktop のアプリの削除が完了します。

しかし、Rancher Desktop インストール時に WSL 上に自動で作成されるディストリビューションが残ったままなので、次のステップで削除を行います。

9.4.1.3. ターミナルを開き、以下のコマンドで Rancher Desktop が作成したディストリビューションの確認を行います。

```
wsl.exe -l
```

rancher-desktop と **rancher-desktop-data** が残っていれば、次のステップで削除を実施してください。

```
PS C:\Users\██████████> wsl.exe -l
Linux 用 Windows サブシステム ディストリビューション：
Ubuntu (既定)
rancher-desktop
rancher-desktop-data
```

- 9.4.1.4. ターミナルに以下のコマンドを入力して、Rancher Desktop が作成したディストリビューションの削除を行います。

```
wsl --unregister rancher-desktop-data
wsl --unregister rancher-desktop
wsl.exe -l
```

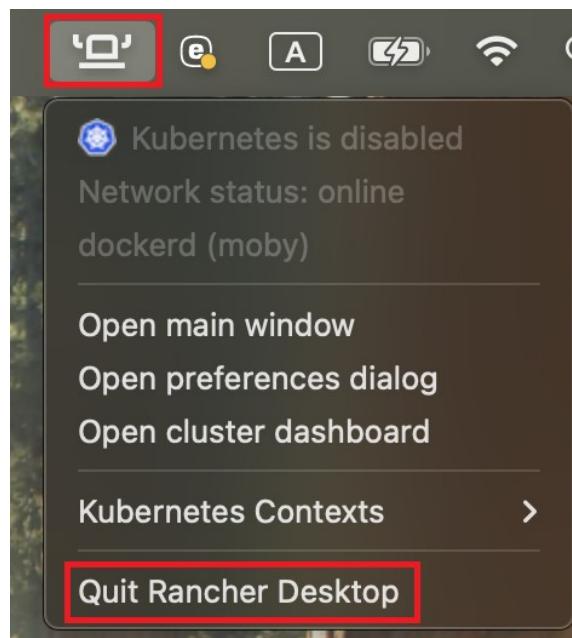
上記の最後のコマンドで **rancher-desktop** と **rancher-desktop-data** が消えていれば、正常に削除が完了しています。

```
PS C:\Users\██████████> wsl --unregister rancher-desktop-data
登録解除。
この操作を正しく終了しました。
PS C:\Users\██████████> wsl --unregister rancher-desktop
登録解除。
この操作を正しく終了しました。
PS C:\Users\██████████> wsl -l
Linux 用 Windows サブシステム ディストリビューション：
Ubuntu (既定)
```

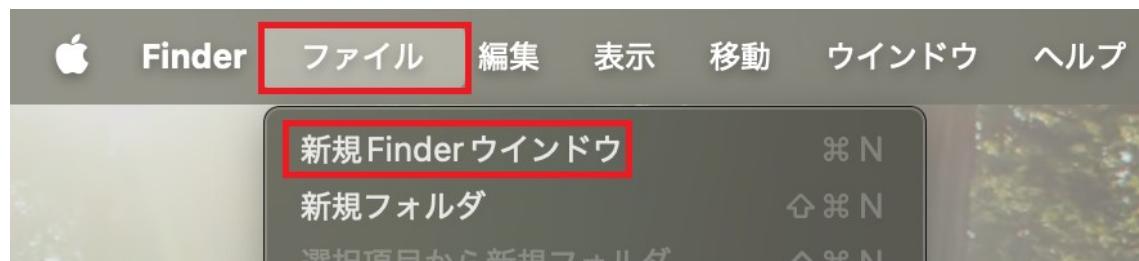
9.4.2. Mac

Mac でアンインストールする場合、以下の手順で行ってください。

- 9.4.2.1. Rancher Desktop が起動中であれば、画面右上のアイコンをクリックして Quit Rancher Desktop をクリックします。

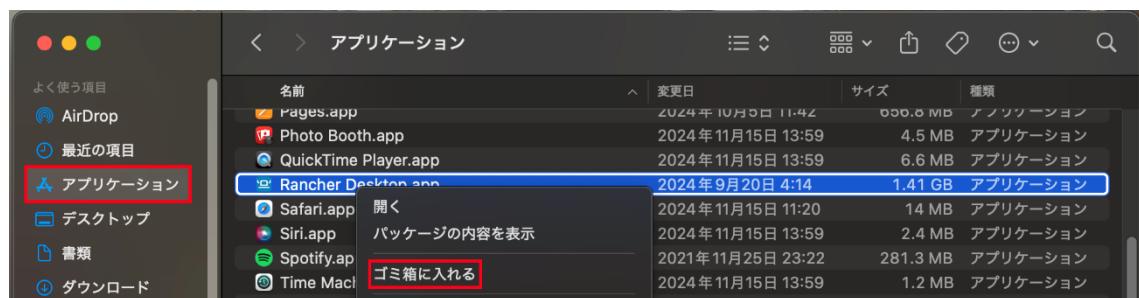


- 9.4.2.2. 新規 Finder ウィンドウを開きます。



アプリケーション > Rancher Desktop.app を探し、右クリックします。

ゴミ箱に入れるをクリックします。



- 9.4.2.3. 管理者でない場合、管理者アカウントを入力し、OK をクリックすることでゴミ箱に入ります。



- 9.4.2.4. 完全に削除したい場合は、Finder からゴミ箱を空にする...をクリックすると完全に削除されます。



以上で、アンインストールが完了しました。

10. 執筆者・引用法

10.1. 執筆者

李 謙一 国立感染症研究所・細菌第一部
山岸敏明 国立感染症研究所・細菌第一部

10.2. Citation

今後 GitHub や論文での公開を予定していますが、論文等で本プログラムを用いた方法を記載する際には、差し当たって GitHub ページ (<https://github.com/leech-rr/SNPeaster>) か次の論文をご参照ください。

- 10.1. Lee K, Iguchi A, Uda K, Matsumura S, Miyairi I, Ishikura K, Ohnishi M, Seto J, Ishikawa K, Konishi N, Obata H, Furukawa I, Nagaoka H, Morinushi H, Hama N, Nomoto R, Nakajima H, Kariya H, Hamasaki M, Iyoda S. 2021. Whole-genome sequencing of Shiga toxin-producing *Escherichia coli* OX18 from a fatal hemolytic uremic syndrome case. *Emerg Infect Dis* 27:1509-1512.
- 10.2. Lee K., Iguchi A., Terano C., Hataya H., Isobe J., Seto K., Ishijima N., Akeda Y., Ohnishi M., and Iyoda S. Combined usage of serodiagnosis and O antigen typing to isolate Shiga toxin-producing *Escherichia coli* O76:H7 from a hemolytic uremic syndrome case and genomic insights from the isolate. 2024. *Microbiology spectrum* 12:e0235523.