



2019年10月18日

技術講習会
DOCKER 環境の構築 改訂版



内容

序	2
Docker 環境の構築	3
Windows10 Home	3
VirtualBOX, Git for Windows, DockerToolbox のダウンロード	3
CPU 仮想化の確認	3
(無効になっていた場合) BIOS 設定画面の起動&VT の有効化	3
VirtualBOX, Git for Windows のインストール	4
DockerToolbox のインストール	4
(初回の Docker Quickstart Terminal の実行でエラーが出る場合)	6
NDIS6 の有効化	6
VM の共有フォルダの設定	6
VM のポートフォアリングの設定	7
VM の再起動	7
Window10 Professional	9
Hyper-V の有効化	9
コンテナの保存場所の変更	9
Docker Desktop for windows のインストールと設定	10
MSYS2 のインストールと設定	10
(オプション) docker-ホスト OS との通信許可設定	11
MacOS	12
Docker Desktop for Mac のインストール (方法-1)	12
Docker Desktop for Mac のインストール (方法-2)	12
File Sharing の設定	12
コンテナの作成	14
rnakato/singlecell_jupyter の解析環境構築	14
イメージからコンテナを作成	14
その他	16
よく使用する Docker コマンド	17

序

この資料は 2019 年 10 月 02 日～04 日にかけて、中戸隆一郎先生（東京大学定量生命科学研究所）が主催された「新学術領域細胞ダイバース第 2 回技術講習会」の 1 日目の配布資料について、加筆・修正を加えたものです。各自が解析に使用する OS ごとに Docker 環境を整備し、中戸隆一郎先生が作成した Docker (rnakato/singlecell_jupyter) による解析パイプラインを確立することを目的とします。

講習会 2 日目で使用した教材(<https://singlecellanalysis.herokuapp.com/index.html> ID や PW はメールにて伝達済) の内容を実行できることは確認していますが、各自の PC の設定によっては、うまくいかない部分があるかもしれません。その場合は、個別に対応するので、中戸 (rnakato@iam.u-tokyo.ac.jp) または林 (hiroton-h@iam.u-tokyo.ac.jp) までご連絡ください。

Docker での解析環境は一度構築してしまえば、どこにでも同じ環境を容易に再現することができる、という大きなメリットがあります。また、サーバーの構築など研究外にも活用が可能ですので、積極的に活用してみてください。

皆様の快適で実りある 1 細胞解析ライフをお祈りいたします。

2019 年 10 月某日 林 寛敦（東京大学定量生命科学研究所秋山研究室）

更新履歴

2019.10.18 ver. 1.0

致命的なミスや docker 環境の大幅な変更があったら更新予定

Docker 環境の構築

Windows10 Home

VirtualBOX, Git for Windows, DockerToolbox のダウンロード

いずれも最新版を使用する。

下記のページから最新版のインストーラーをダウンロードする。

VirtualBox

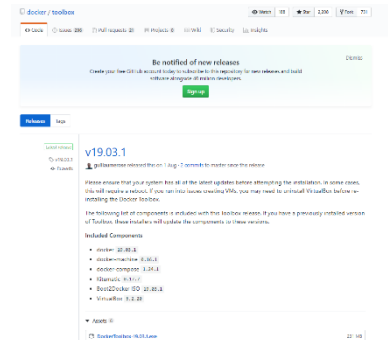
<https://www.virtualbox.org/>

Git for Windows

<https://gitforwindows.org/>

DockerToolbox

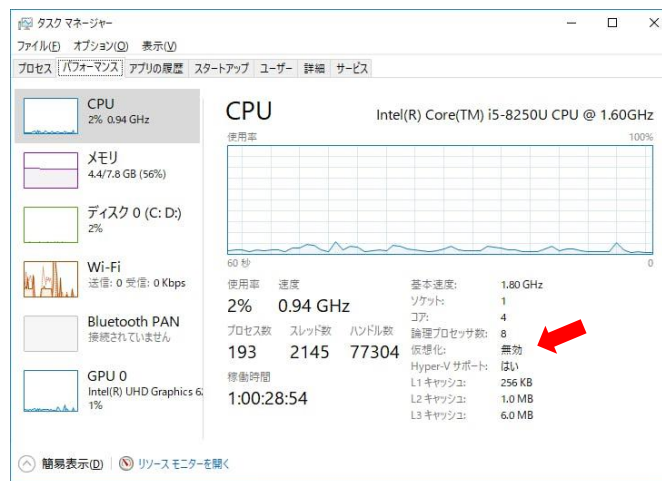
<https://github.com/docker/toolbox/releases>



CPU 仮想化の確認

インストールの前に CPU の仮想化が有効になっているか確認し、無効になっていた場合は、Virtualization Technology を有効に変更する。

1. タスクマネージャ⇒パフォーマンスタブ⇒CPU
2. 右下の[仮想化]が"無効"の場合、BIOS の設定から有効化



(無効になっていた場合) BIOS 設定画面の起動&VT の有効化

再起動中に BIOS 設定画面への移行のためのキーが表示されるので、素早く該当のキーを押して BIOS 設定画面へ入る。

大抵の場合は、Del キーまたは F2 キー (Chipset によって異なる) を押すことで入るこ

とができる。PC のスペックが良いとキーの入力タイミングがシビアなので、再起動後に連打推奨。

BIOS 設定画面に入ったら、CPU の設定項目に移動し、Intel VT (Virtualization Technology) (←必ずしも左記の記載通りではないが、類似した内容がある)を有効に切り替え、BIOS の設定を保存して BIOS 設定画面を終了する。

例：<https://bibabosi-rizumu.com/bios-virtualization/>

VirtualBOX, Git for Windows のインストール

- VirtualBOX :

VirtualBOX のインストーラーからインストールを実行する。

その際、念のためインストーラーを右クリックして「管理者として実行」からインストール作業を開始する。

特に設定等の変更は必要ないので、手順に従ってインストールを完了させる。

- Git for Windows (オプション)

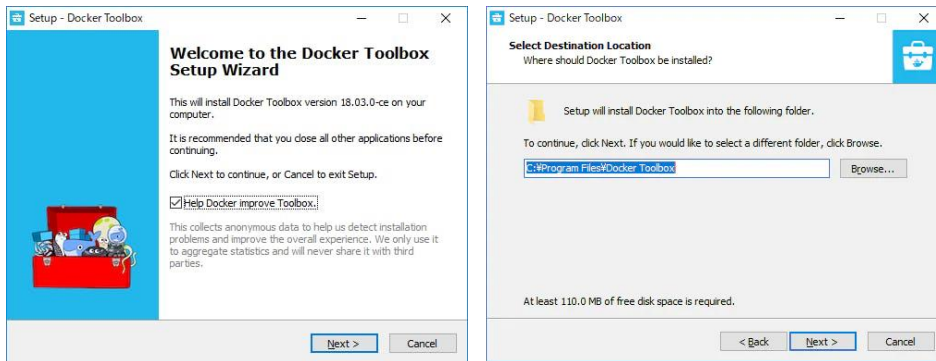
Docker QuickStart Terminal 以外のターミナルに興味がある場合は、別途インストールする。Git for Windows 以外のターミナルでも特に問題はなく、好みのターミナルを選んでよい (<https://qiita.com/Ted-HM/items/9a60f6fcf74bbd79a904> Windows で使えるシェル環境 参照)。ただし、複数のターミナルを同時にインストールしないこと。

VirtualBOX と同様に、念のためインストーラーを右クリックして「管理者として実行」からインストール作業を開始する。

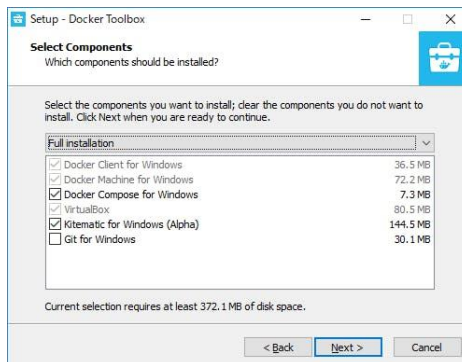
Git for Windows のインストール作業については、特に設定等の変更は必要ないので手順に従ってインストールを完了させる。

DockerToolbox のインストール

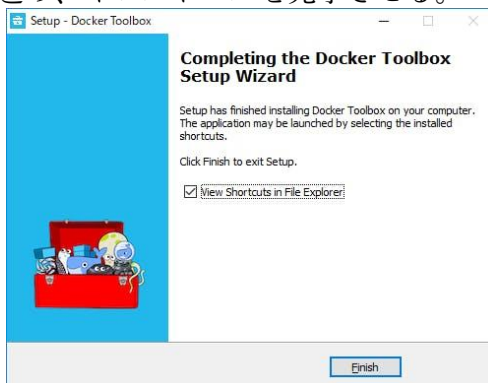
- ダウンロードしたインストーラーを右クリックして「管理者として実行」からインストール作業を開始する。下記に記載の変更点以外はデフォルトの設定で良い。



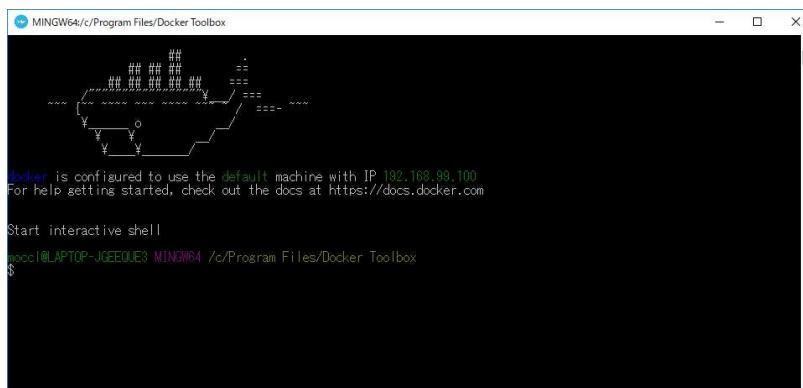
- VirtualBox はインストール済みのため、「Custom Installation」で VirtualBox のチェックを外す。Git for Window もチェックしない。



3. インストール作業を進め、インストールを完了させる。



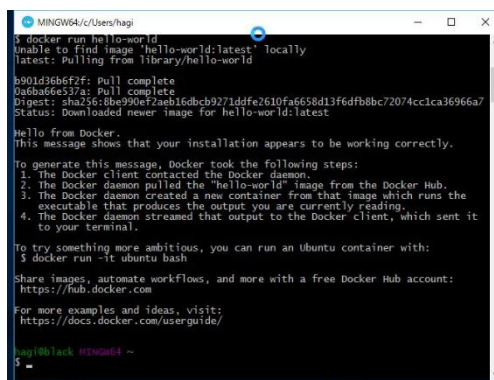
4. デスクトップに生成された Docker Quickstart Terminal を実行する。クジラが現れて、ターミナルにコマンドが入力できるようになれば無事、インストール完了です。(途中、VirtualBox によるコンピュータの変更ウィンドウが表示された場合は、"はい"を選択する)。



5. 下記のコマンドを入力する。

```
docker run hello-world
```

DockerToolBOX が正しくインストールされていれば、「hello-world」の Docker イメージが DockerHub から取得され実行される。下記の画面のようになれば DockerToolBOX が正常に稼働していることになる。



```
exit
```

一度、ターミナルを閉じます。

(初回の Docker Quickstart Terminal の実行でエラーが出る場合)

NDIS6 の有効化

```
Failed to open/create the internal network 'HostInterfaceNetworking-VirtualBox Host-Only Ethernet Adapter' (VERR_INTNET_FLT_IF_NOT_FOUND).  
Failed to attach the network LUN (VERR_INTNET_FLT_IF_NOT_FOUND).
```

最初に Docker Quickstart Terminal を実行した際、ネットワークの設定で上記のようなエラーが出ることもある。その場合は、下記の手順に従って、NDIS6 の有効化を再度行う。

- 1 「ネットワークと共有センター」を開く。
- 2 「アダプターの設定の変更」を開く。
- 3 「VirtualBox Host-Only Network #N」のプロパティを開く。
(#N の N はエラーの際に表示されている番号)
- 4 「VirtualBox NDIS6 Bridged Networking Driver」にチェックを入れる。
- 5 「インターネットプロトコル バージョン 6(TCP/IPv6)」のチェックを外す。
- 6 [OK] を押し、プロパティウィンドウを閉じる。
- 7 「VirtualBox Host-Only Network #N」の右クリックメニューで「無効」にする。
- 8 再度、「有効」に設定し直す。

参考資料：

VirtualBox で Failed to open/create the internal network 'HostInterfaceNetworking-VirtualBox Host-Only Ethernet Adapter' が出た時の対処

https://qiita.com/ExA_DEV/items/ae80a7d767144c2e1992

VM の共有フォルダの設定

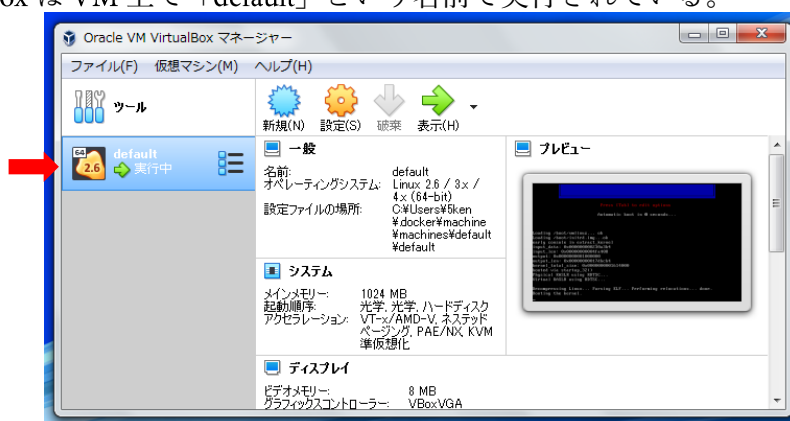
ホスト (Windows10 Home) と Docker でデータをやりとりするための共有するフォルダを設定する。Docker の起動を終了すると解析結果のファイルなども消去されてしまうが、共有フォルダにファイルを残しておけば消去されないため、Docker を使用する上で必須の設定となる。

Docker Toolbox は VM 上で起動している為、「Docker⇄VM⇄ホスト」で共有フォルダの設定をする必要がある。まずは、「VM⇄ホスト」の設定を行う。

「Docker⇄VM」については、後述のコンテナの作成の際に設定する。

VirtualBox を起動する。

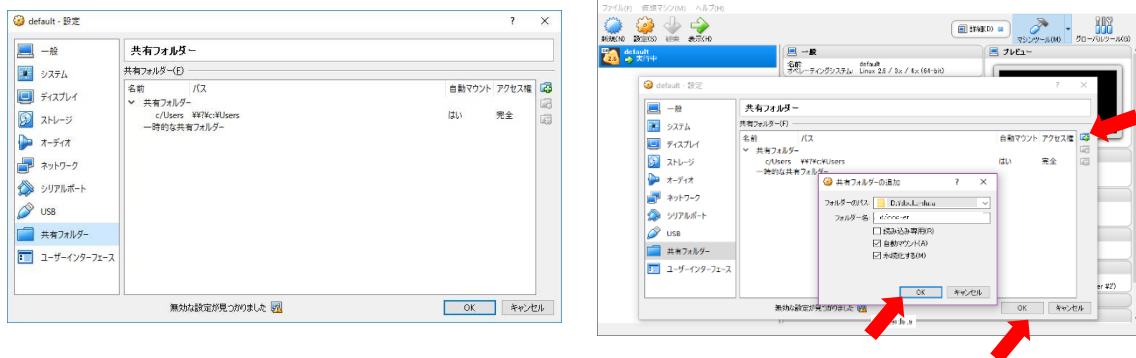
Docker Toolbox は VM 上で「default」という名前で実行されている。



以下に、Dドライブに「dockerdata」フォルダを作成し、共有設定する場合の例を記す。

- 1 VirtualBOX を立ち上げ、「default」という名前の VM が実行中であることを確認する。
- 2 「default」を選択し、「設定」をクリックする。
- 3 「共有フォルダ」を選択し、右上の追加ボタンをクリックする。

- 4 共有したホスト OS のパスと「default」上での共有フォルダ名を設定して OK を押す。
 - A) フォルダのパス (ホスト側で共有設定したいフォルダの絶対パス: D:\dockerdata)
 - B) フォルダ名 (VM 上の共有フォルダの名前。任意。後で「docker⇔VM」の共有設定で使用する: dockerdata)
 - C) 「永続化」、「自動マウント」をチェック



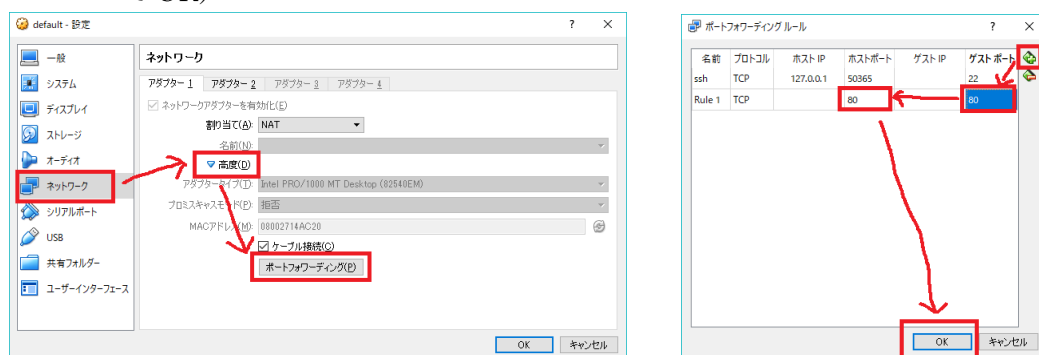
- 5 共有フォルダ設定のウィンドウで OK を押して設定を完了させる。

VM のポートフォアリングの設定

次に、「VM⇔ホスト OS」のポートフォアリングを設定する。デフォルトの設定で Jupyter は「8888」、Rstudio は「8787」を使用するので、両方のポートを開放する。

以下に、jupyter 用にポート「8888」を開放する例を記す。Rstudio 用には、下記の「8888」を「8787」に置き換えて同様の設定をすればよい。

1. 共有フォルダ設定の時と同様に、VM 上の「default」マシンの「設定」→「ネットワーク」を選択する。
2. 「高度」タブを展開し、ポートフォアリングをクリックする。
3. 左上の追加ボタンを押し、ホストポート、ゲストポートに「8888」を設定する。(名前は任意、プロトコルは TCP、ホスト IP は「127.0.0.1」、ゲスト IP はなしで OK)



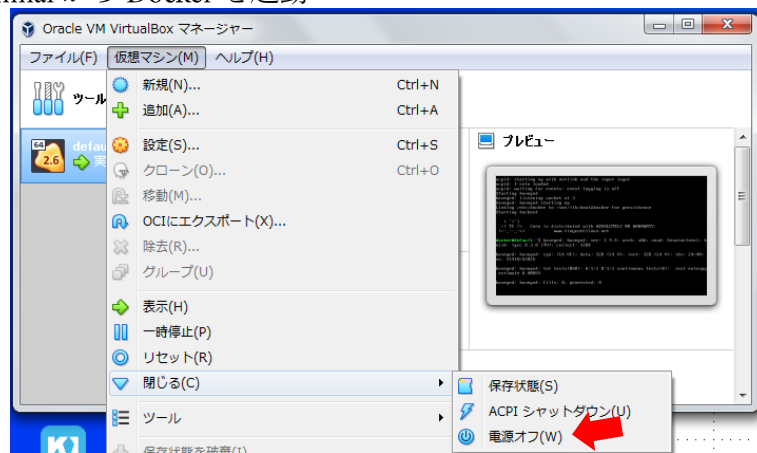
これで、「Docker (8888) ⇔ VM (8888) ⇔ホスト (8888)」とポートを通す準備が完了です。

VM の再起動

VM 「default」を再起動する。

1. 「default」を選択した状態で、メニューの「仮想マシン」から「閉じる」→「電源 OFF」を選択する。

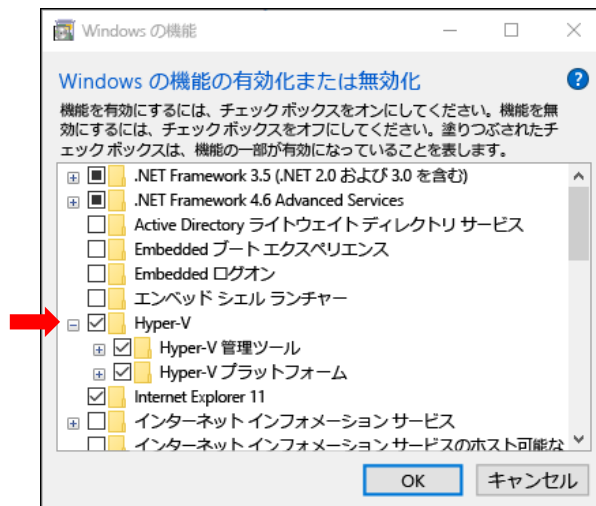
2. DockerQuickTerminal から Docker を起動



Window10 Professional

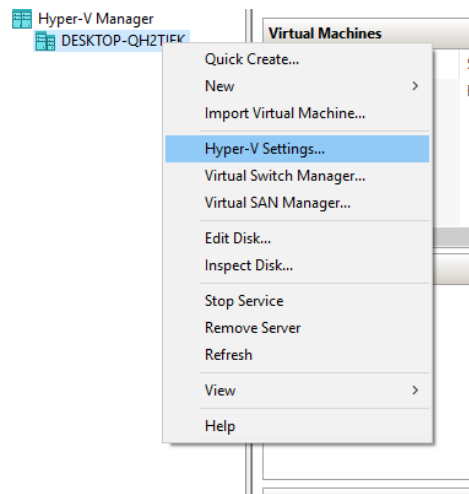
Hyper-V の有効化

1. [設定] で Hyper-V ロールを有効にする
2. Windows ボタンを右クリックし、[アプリと機能] を選択する。
3. 適切な [関連設定] の下にある [プログラムと機能] を選択する。
4. [Windows の機能の有効化または無効化] を選択する。
5. [Hyper-V] にチェックを入れて、[OK] をクリックする。

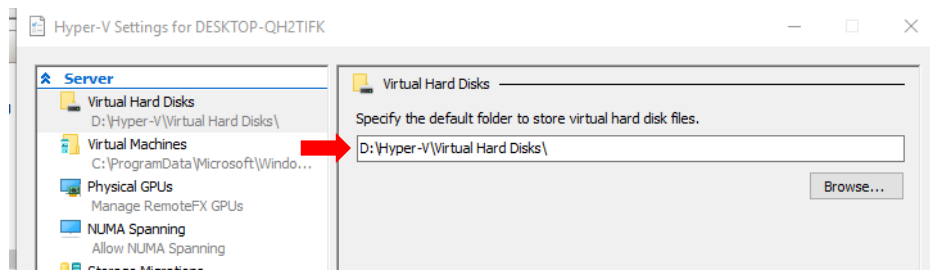


コンテナの保存場所の変更

1. 検索 window に「Hyper-V」と入力し、「Hyper-V Manager」を起動する。仮想マシンが表示されているので、右クリックして「Hyper-V の設定」を選択する。注) 図は英語版

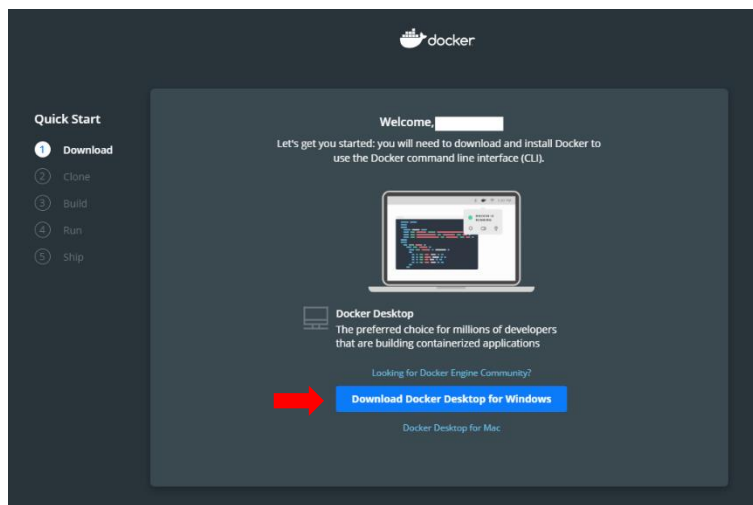


2. Virtual Hard Disks を選択し、コンテナを保存したいフォルダ (任意) を指定する。(空き容量が豊富なディスクにあるフォルダを指定する、など)

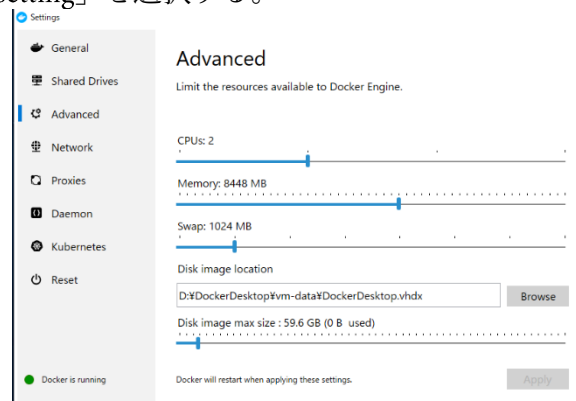


Docker Desktop for windows のインストールと設定

1. DockerID を登録して Docker for windows をダウンロードする
<https://hub.docker.com/?overlay=onboarding>
 DockerHub にログイン後の画面



2. インストーラーをダウンロードして、Docker Desktop を実行する。
3. 準備が整うとタスクバーの常駐アプリに Docker Desktop が表示されているので、右クリックして「setting」を選択する。



4. 「Advanced」を選択し、docker 実行時のマシンパワー（CPUのコア数）やメモリーの最大値、docker image を保存する場所などを設定（各自の好み）する。
5. タスクバーの常駐アプリにある Docker Desktop を右クリックして「Restart」を選択し、docker Desktop for windows を再起動する。

MSYS2 のインストールと設定

1. x86_64 版（64bit）をダウンロードしてインストールを実行する。
<http://www.msys2.org/>



2. 検索 window に「msys2」と打つと表示される MSYS2 MinGW 64-bit を右クリックして、「ファイルの場所を開く」を選択する。
3. MSYS2 MinGW 64-bit を右クリックして「送る」→「デスクトップにショートカットを作成」を選択する。
4. デスクトップに作成された MSYS2 MinGW 64-bit を右クリックして「管理者として実行」を選択する。
5. MSYS2 に付属するソフトウェア管理ソフト pacman のアップデートを実行する。

```
pacman -Syu
- -- 省略 ---
WARNING: the shell starting scripts have been unified. Please update your
shortcuts to the following targets, otherwise they will STOP WORKING:
MSYS2_ROOT¥msys2_shell.cmd -mingw32
MSYS2_ROOT¥msys2_shell.cmd -mingw64
MSYS2_ROOT¥msys2_shell.cmd -msys
- -- 省略 ---
警告: terminate MSYS2 without returning to shell and check for updates again
警告: for example close your terminal window instead of calling exit
```

6. 警告は気にせずターミナルを閉じる
7. 再度、MSYS2 MinGW 64-bit を「管理者として実行」で起動する。
8. アップデートの確認と git と winpty をインストール（多少、時間がかかる）する。

```
pacman -Su
pacman -S git
pacman -S winpty
```

9. Docker を起動するためのパスとエイリアスを設定する。

```
export PATH=$PATH:/c/Program Files/Docker/Docker/resources/bin
alias docker='winpty docker'
alias docker-compose='winpty docker-compose'
```

10. docker が実行できることを確認する。

```
docker --version
docker run hello-world
```

正しい情報が帰ってくれば設定完了です。Docker イメージ「hello-world」の正しい出力結果は Windows10 Home の項を参照（5 ページ）。

（オプション）docker-ホスト OS との通信許可設定

セキュリティソフトを使用している場合、「127.0.0.1」「127.0.0.2」との通信を許可しておく必要がある。

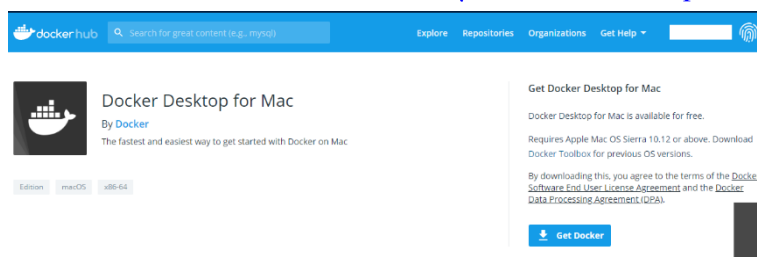
設定方法の詳細は、使用しているセキュリティソフトのヘルプを参照すること。

MacOS

Docker Desktop for Mac のインストール (方法-1)

1. 公式サイトから Docker のアカウントを作ってログインし、DockerHub から Docker Desktop for Mac をダウンロードする。

<https://hub.docker.com/editions/community/docker-ce-desktop-mac>



2. インストーラーを起動し、流れに従ってインストールする。
3. docker が実行できることを確認する。

```
docker -version  
docker run hello-world
```

正しい情報が帰ってくれば設定完了です。Docker イメージ「hello-world」の正しい出力結果は Windows10 Home の項を参照 (5 ページ)。

Docker Desktop for Mac のインストール (方法-2)

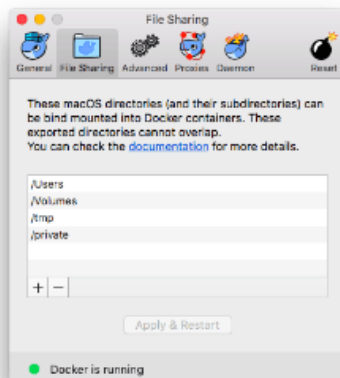
Homebrew と Homebrew-Cask が導入済みの場合、下記のインストール方法も可能です。。

```
brew update  
brew cask install docker  
docker --version  
#アプリケーションに Docker.app を実行するか下記のコマンドで初期設定  
#やっていることは同じで、つまりは docker の起動  
open /Applications/Docker.app
```

File Sharing の設定

ホスト (Mac OS) と Docker でデータをやりとりするための共有するフォルダを設定する。Docker の起動を終了すると解析結果のファイルなども消去されてしまうが、共有フォルダにファイルを残しておけば、消去されないため、Docker を使用する上で必須の設定となる。

ホスト側でコンテナと共有するディレクトリの場所を Docker for mac アプリの「File Shearing」タブで指定する。



デフォルトで「/Users」下のファイルは共有できるようになっている。なので、/Users 以下のディレクトリに共有ディレクトリを設定する場合は、特に追加の設定は必要としない。

コンテナの作成

rnakato/singlecell_jupyter の解析環境構築

ターミナルで下記を実行する。

```
docker pull [オプション] [Docker イメージ名 またはレジストリホスト/Docker イメージ名]:タグ名
docker pull rnakato/singlecell_jupyter
```

docker pull コマンドは、Docker イメージ名のみを指定した場合は、デフォルトで Docker 公式レジストリに接続する。Docker イメージ名を指定する際にタグ名を省略すると、「latest」タグが使用される。

```
docker images
```

rnakato/singlecell_jupyter のイメージが存在することを確認

イメージからコンテナを作成

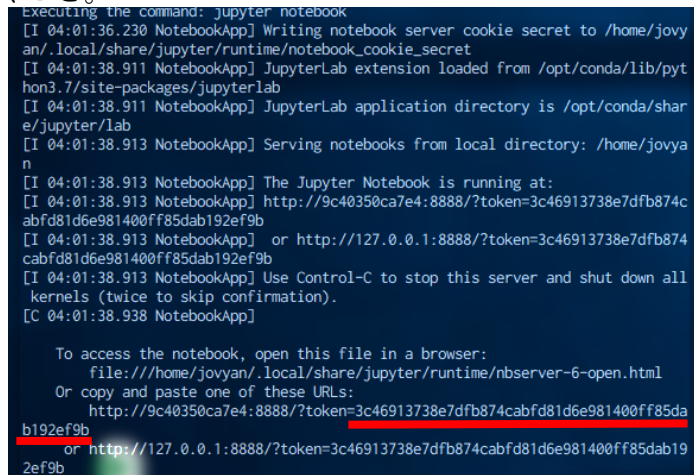
下記に記すのは一例で、docker の使用方法に応じて様々な設定を盛り込んでコンテナを作成することが可能です。

例 1

コンテナの構成を変更する予定がない場合にお勧めです。

```
docker run ¥
--name notebook ¥
-p 8888:8888 ¥
-v (your directory):/home/jovyan/work ¥
rnakato/singlecell_jupyter ¥
start-notebook.sh
```

ターミナルに表示される token は、Jupyter notebook を起動する際に必要な場合があるのでコピーしておくこと。



```
Executing the command: jupyter notebook
[I 04:01:36.230 NotebookApp] Writing notebook server cookie secret to /home/jovyan/.local/share/jupyter/runtime/notebook_cookie_secret
[I 04:01:38.911 NotebookApp] JupyterLab extension loaded from /opt/conda/lib/python3.7/site-packages/jupyterlab
[I 04:01:38.911 NotebookApp] JupyterLab application directory is /opt/conda/share/jupyter/lab
[I 04:01:38.913 NotebookApp] Serving notebooks from local directory: /home/jovyan
[I 04:01:38.913 NotebookApp] The Jupyter Notebook is running at:
[I 04:01:38.913 NotebookApp] http://9c40350ca7e4:8888/?token=3c46913738e7dfb874cabfd81d6e981400ff85dab192ef9b
[I 04:01:38.913 NotebookApp] or http://127.0.0.1:8888/?token=3c46913738e7dfb874cabfd81d6e981400ff85dab192ef9b
[I 04:01:38.913 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 04:01:38.938 NotebookApp]

To access the notebook, open this file in a browser:
file:///home/jovyan/.local/share/jupyter/runtime/nbserver-6-open.html
Or copy and paste one of these URLs:
http://9c40350ca7e4:8888/?token=3c46913738e7dfb874cabfd81d6e981400ff85dab192ef9b
or http://127.0.0.1:8888/?token=3c46913738e7dfb874cabfd81d6e981400ff85dab192ef9b
```

例 2

さらにツールのインストールなどコンテナの構成を変更する可能性がある場合は、下記をお勧めします。

```
docker run --rm -it ¥
rnakato/singlecell_jupyter //bin/bash -c ¥
"python -c 'from notebook.auth import passwd;print(passwd())'"
```

#パスワードの入力を 2 回求められるので、jupyter にログインするためのパスワード (任意) を入力

#キーが表示されるのでコピーしておく


```
docker run ¥
  --user root ¥
  -e GRANT_SUDO=yes ¥
  -e TZ=Asia/Tokyo ¥
  -e JUPYTER_ENABLE_LAB=yes ¥
  -p 8888:8888 ¥
  --name notebook ¥
  -v (your directory):/home/jovyan/work ¥
  rnakato/singlecell_jupyter:latest ¥
  start-notebook.sh ¥
  --NotebookApp.password='上記でコピーしたキーを張り付け'
```

オプションの説明

-v

共有ディレクトリのマウント設定。

Windows10 Home

-v [VMの共有フォルダ]:[コンテナ内の共有フォルダ]

[VMの共有フォルダ]には先に設定したVMの共有フォルダ名を記載すればよい。

共有フォルダ名を dockerdata とした場合、 /dockerdata:/home/jovyan/work

Mac

-v [ホスト (Mac OS) の共有フォルダ]:[コンテナ内の共有フォルダ]

ホストの共有フォルダには、File Sharing に記載した場所 (PATH) を記載すればよい。

--user root

-e GRANT_SUDO=yes

Docker の変更には root 権限が必要なため、新たにツールを加えるなどコンテナの構成を変更する予定がある場合は、ユーザー名を root にして、命令文の実行を常に root 権限で行う (sudo) の設定にしておくとう便利。

-e JUPYTER_ENABLE_LAB=yes

Jupyter には Jupyterlab という後継がある。Jupyter notebook は Jupyter lab への移行が正式にアナウンスされている。Jupyter lab を起動する場合は、「JUPYTER_ENABLE_LAB」を「yes」とする。

-p

通信に使用するポートの設定。

Jupyter notebook はデフォルトの設定では「8888」を使用する。使用するツールによって適宜変更する。

-rm

上記では指定していないが、コンテナ終了時にコンテナを自動で削除したい場合は、このオプションをつける。(コンテナがたまって容量を圧迫するのを防ぐ)

-d

上記では指定していないが、バックグラウンドで docker を起動し続けたい場合に、デタッチモードで起動するためのオプション。

Jupyter notebook の立ち上げ

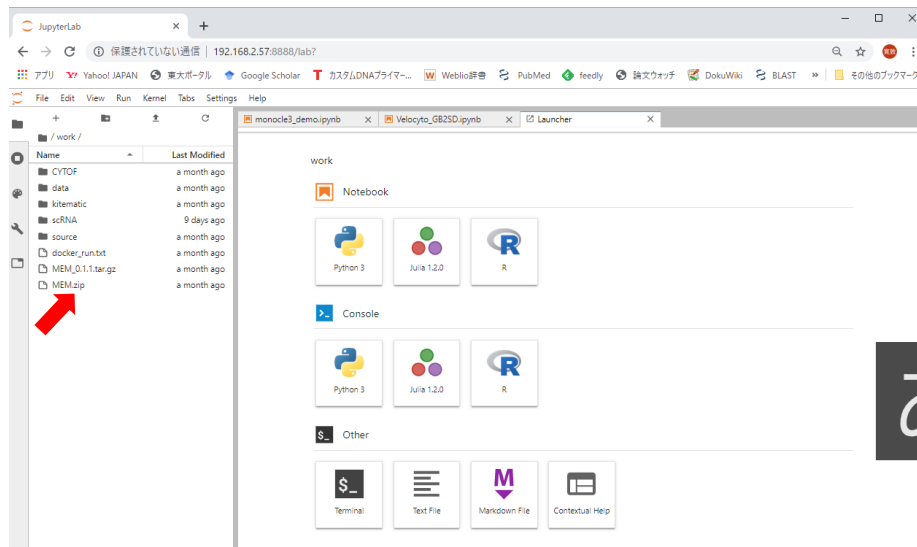
```
#ブラウザに下記を入力
```

```
http://localhost:8888 http://127.0.0.1:8888 または http://\[ホスト IP アドレス\]:8888
```

ホスト IP アドレスの確認法: <https://www.cman.jp/network/term/ip/p2/>

予めホスト OS 側の共有フォルダにファイルまたはフォルダを置き、jupyter 上で該当のファイルまたはフォルダが確認できれば正しく共有フォルダの設定がされている。右側のウィンドウで、notebook (Python3、Julia または R と何でもよい) を選択して、新し

く notebook が作成されることを確認する。(下図は Jupyter lab)



共有設定がうまくいっていない場合

→ Windows10 Home

Jupyter notebook からログアウトし、コンテナを停止。VM「default」を再起動する。

→ その他 OS、上記で解決しない場合

共有設定コマンドを再確認する。それでもうまくいかない場合は、個別に問い合わせてください。

その他

2019年10月18日現在、Windows10 Homeにおいて、Jupyter notebook上でlibrary(velocyto.R)を実行しようとするとうエラーが出る場合があることを確認しています。

対応予定ですが、急ぎの場合は下記を実行してください。

コンテナ内に入り、ターミナルでRを起動する。

```
docker exec -it コンテナ名 /bin/bash
#コンテナの中に入ったら、Rを起動
R
#大文字
```

velocyto.Rを再インストールする。

```
install.packages("devtools")
devtools::install_github('velocyto-team/velocyto.R',force=TRUE)
```

よく使用する Docker コマンド

```
docker
```

とターミナルに打つと docker のコマンド一覧が表示される。
以下によく使用するコマンドを抜粋した。

コンテナへのログイン

```
docker exec -it [コンテナ ID またはコンテナ名] /bin/bash
```

起動中のコンテナの確認

```
docker ps
```

停止中のコンテナの確認

```
docker ps -a
```

コンテナの停止

```
docker stop [コンテナ ID またはコンテナ名]
```

コンテナの起動

```
docker start [コンテナ ID またはコンテナ名]
```

コンテナの削除

```
docker rm [コンテナ ID またはコンテナ名]
```

イメージの削除

```
docker rmi [コンテナ ID またはコンテナ名]
```

コンテナをイメージとして保存

コンテナ内で解析環境を変更した場合（例えば、新しいツールを入れたなど）のバックアップ。保存したいコンテナは予め停止しておく。

[コンテナ ID] は「docker ps -a」で確認。[イメージ名]:[タグ名]は任意。

```
docker commit [コンテナ ID] [イメージ名]:[タグ名] [オプション]
```

イメージの書き出し

```
docker save [イメージ ID またはイメージ名] -o 任意の名前.tar
```

イメージの読み込み

```
docker load -i 任意の名前.tar
```

コンテナの情報

IP アドレスや共有フォルダの情報など

```
docker inspect [コンテナ ID またはコンテナ名]
```

Docker におけるコンテナのライフサイクル
<http://enakai00.hatenablog.com/entry/20140628/1403933390>

