

Net 実験

Docker 環境構築

はじめに: Dockerとは？



- コンテナ型の仮想化技術

- 仮想化: ホスト (≡ 物理マシンに載っているOS) の上でゲスト (仮想マシン) を動かすこと
 - 普通の仮想マシンは、ホストがCPUやメモリーなどの機能をエミュレートして動かす
- コンテナ型: ホスト (主にLinuxに限るが) の名前空間などの機能を使ってホストとゲストを分離しつつも、ホストのプロセスの1つとしてゲストが動く

- Pros: 従来の仮想マシンより軽量
- Cons: ゲストの機能 (カーネル等) がホストに依存

今回のケースごとのDocker環境

ホストがLinux以外の場合は
何らかの仮想化技術を挟む必要あり

- 実験室 (すでに構築済み)
 - Ubuntu 18.04 (物理マシン) の上でDockerコンテナを動かす
- 個人の Mac (High Sierra以降)
 - Macの仮想化機能 (Hypervisor Framework) の上でDockerコンテナを動かす
- 個人の Windows
 - VirtualBoxで仮想マシンのUbuntuを起動し、その上でDockerコンテナを動かす
- 個人の Linux系OS
 - Ubuntu、CentOS、Debian、Fedora 等
 - そのまま Docker が動く (本資料では触れない)
 - 例: Ubuntu での Docker 導入方法 <https://docs.docker.com/engine/install/ubuntu>
- それ以外
 - 上記以外のOSでは、Dockerは非対応 or 導入が容易ではないため本資料では触れない

Mac での Docker 環境構築

注意事項

- 2010年以降のモデルのMac
- **OSがHigh Sierra以降であること**
- 最小4GBのメモリ
 - また、余裕がなさそうなら不必要なアプリケーションは落として実験すること
- バージョン4.3.30以前のVirtualBox をもしインストールしている場合、
Docker と競合して動かないので注意
 - その場合はVirtualBoxを更新 or アンインストールを推奨

Dockerのインストール

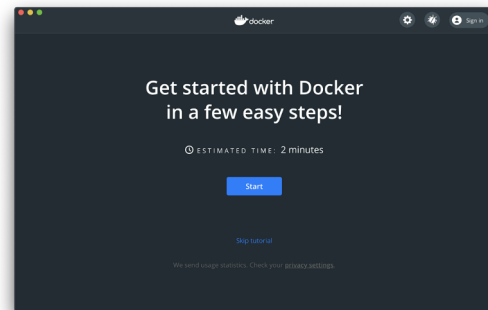
- Homebrewをインストール済みであれば、ターミナルから以下を実行すればインストールが完了する

```
brew cask install docker
```

- そうでなければ以下にアクセスして「Get Stable」ボタンからダウンロード & 実行
 - <https://hub.docker.com/editions/community/docker-ce-desktop-mac>
 - インストラクションに従って進める。管理者権限のためのパスワードを途中求められるので入力する

Docker の起動と終了

- インストール後、LaunchPadやspotlightからDockerを起動
- 右の画面が出たら起動完了。Skip Tutorialしてこの画面は閉じて良い
 - メニューバーに下のようにアイコンが出て、アニメーションが終われば起動完了のサイン



- Dockerを終了させたい場合はメニューバーのDockerアイコンをクリックして、終了 (Quit) を選ぶ

Dockerを試す

- ターミナルを開いて、以下を実行してみる
`docker run hello-world`
- 以下のように表示されればOK

```
~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:4cf9c47f86df71d48364001ede3a4fcd85ae80ce02ebad74156906caff5378bc
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:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```


今回配布した Docker 環境を試す

- Mac で ics-exp-lan.zip を解凍し、適当な場所へ配置する
 - 今はホームディレクトリ直下へ置いたとする
- あとは以下のように実行していけば良い。詳細はics-exp-lan/README.mdを読むこと
 - `cd ~/ics-exp-lan`
 - `make build`
 - `make start`
 - `make login-nod1`
 - ...

Docker、Mac 間でのファイル共有

- Dockerの中の `/home/ubuntu/share` は Mac の `~/ics-exp-lan/share` と自動で同期される
 - 注: Mac でホームディレクトリ直下へ配布ファイルを置いた場合の話
- ファイルの確認やコピーには、これらを上手く使うこと

WindowsでのDocker環境構築

WindowsでのケースごとのDocker環境

- Windows 10 Home (64 bit)
 - 方法1: VirtualBoxで手動でLinuxをインストールして、その上でDockerを動かす
 - 方法2: WSL2 (Windows Subsystem for Linux) の上でDockerを動かす ✖
- Windows 10 Pro / Enterprise / Education (64bit)
 - 方法1: VirtualBoxで手動でLinuxをインストールして、その上でDockerを動かす
 - 方法2: WSL2 (Windows Subsystem for Linux) の上でDockerを動かす ✖
 - 方法3: Windows 10 Pro専用の仮想化機能 (Hyper-V) の上でDockerを動かす
- 上記以外のWindows
 - VirtualBoxで手動でLinuxをインストールして、その上でDockerを動かす

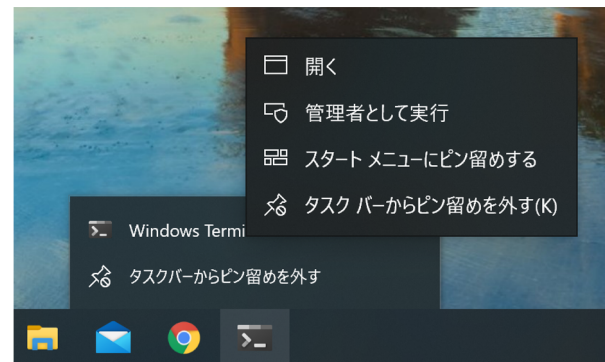
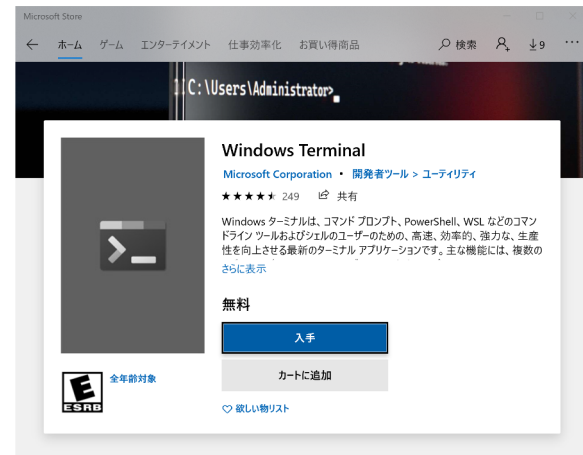
今回はVirtualBoxで手動でDockerを動かす方法とWSL2上でDockerを動かす方法について解説
✖ **WSL2の場合は実験で想定した帯域や遅延の制限が反映されないため**、参考程度にしておくこと
(Docker公式はWSL2を使うのを推奨しているので、今後のWSL2のアップデートに注目しよう)

(推奨) Windows Terminalの導入

- PowerShellなどを開ける上、使いやすい
- Microsoft Storeで検索してインストールするだけ

- 管理者権限での実行方法

- タスクバーのアイコンを右クリック、
出たメニューの「Windows Terminal」をさらに右クリック、
「管理者として実行」
- タスクバーにない場合はスタートメニュー内の
Windows Terminalを右クリックすれば似たようなメニューが
出てくるはず



VirtualBoxで手動でLinuxを
インストールし、Dockerを動かす

VirtualBox と Vagrant のインストール

- VirtualBox: 仮想マシンを動かすためのソフト
 - 通常、OSは自分でインストール
- Vagrant: VirtualBoxなどの仮想化ソフトを使いやすくするソフト
 - 設定をすませたOSなどを簡単にインストールできる
- まず、それぞれ以下のURLからダウンロード & インストールしよう
 - <https://www.virtualbox.org/wiki/Downloads>
 - <https://www.vagrantup.com/downloads>
- その後、Windows を再起動する

Vagrant による Ubuntu のインストール

- Windows Terminal を開き、以下のように実行していく
 - `cd ~/Desktop`
 - `mkdir foobar` # デスクトップに foobar フォルダができるはず
 - `cd foobar`
 - `vagrant init bento/ubuntu-20.04`
 - `vagrant up`
- 多少時間がかかるがUbuntu 20.04のイメージをダウンロードし、Ubuntuが起動する。その後、以下のコマンドを打てばUbuntuにログインできる
`vagrant ssh`

Tips: Vagrant の使い方

- 以下はすべて `vagrant init` したディレクトリ (今回はfoobar以下) で行うこと
 - `vagrant up` # 仮想マシンの起動
 - `vagrant ssh` # 仮想マシンへのSSH (ログイン)
 - `vagrant halt` # 仮想マシンの停止
 - `vagrant destroy -f` # 仮想マシンを完全に削除
- 仮想マシンのCPUやメモリーを増やしたい、というときはいったん仮想マシンを停止し、foobar以下にある `Vagrantfile` を編集してから仮想マシンを起動すると良い
 - 途中のコメントアウトされている部分を参考にとすると良い
 - 例: `config.vm.provider "virtualbox" do |vb| ~ end` の部分のコメントアウトを外し、その中で `vb.memory = "4096"` (MB単位) や `vb.cpu = "4"` などと書いて、仮想マシンを起動すればOK
 - 自分のパソコンのスペックを考慮しながら行うこと！

トラブルシューティング

- `vagrant up` 時などに以下のようなエラーが表示される場合、BIOSで仮想化機能がオフになっている可能性がある
`VT-x/AMD-V hardware acceleration is not available on your system`
- パソコンの機種によってBIOS設定方法は異なるが、起動時にF2を連打するとBIOS設定画面になることが多い
 - 十字キーやタブキー、Enterキーなどの操作方法が画面に表示されていると思うので、それに従って「Virtualization Technology」や「仮想化」の項目を探し出してオンにして起動し直す

Docker を Ubuntu 20.04 にインストール

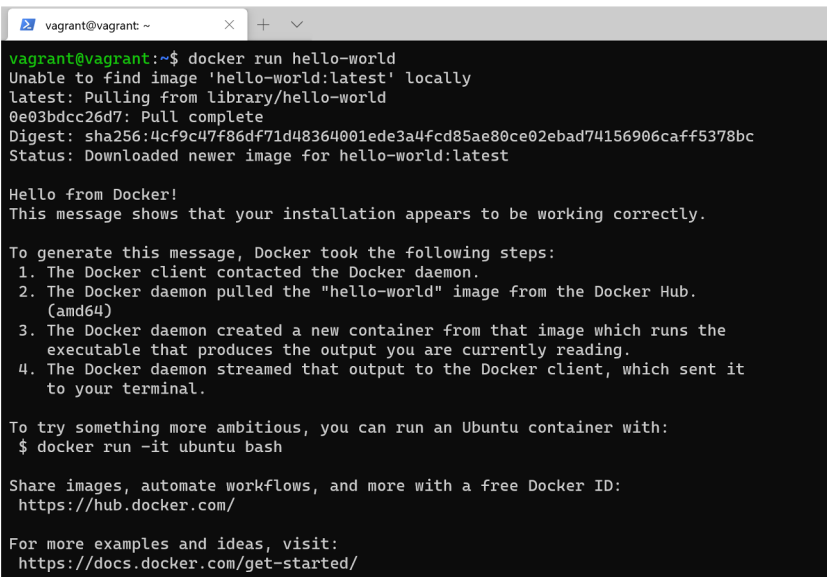
- 基本は <https://docs.docker.com/engine/install/ubuntu> の通り。以下のように **Ubuntu 上** (Windows上でやってもダメ！)でコマンドを実行すれば良い
 - `sudo apt update`
 - `sudo apt install apt-transport-https ca-certificates curl gnupg-agent software-properties-common`
 - `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`
 - `sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"`
 - `sudo apt-get update`
 - `sudo apt-get install docker-ce docker-ce-cli containerd.io`
 - `(sudo groupadd docker)` (おそらくすでにdockerグループは作成されているので不要)
 - `sudo usermod -aG docker $USER`

docker-compose を Ubuntu 20.04 にインストール

- 以下のように **Ubuntu** 上で実行すれば良い
 - `sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose`
 - `sudo chmod +x /usr/local/bin/docker-compose`

Docker を試す

- Ubuntu 上で以下を実行してみる
 - `docker run hello-world`
- 以下のように表示されればOK

A terminal window titled 'vagrant@vagrant ~' with standard window controls. The terminal shows the command 'docker run hello-world' and its output. The output indicates that the 'hello-world:latest' image was pulled from the Docker Hub library. It then shows a 'Hello from Docker!' message and a detailed explanation of the steps Docker took to generate this message: contacting the daemon, pulling the image, creating a container, and streaming the output. Finally, it provides a suggestion to run an Ubuntu container and links to Docker's documentation and image hub.

```
vagrant@vagrant:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:4cf9c47f86df71d48364001ede3a4fcd85ae80ce02ebad74156906caff5378bc
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:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

今回配布した Docker 環境を試す

- Windowsで ics-exp-lan.zip を解凍し、デスクトップにあるfoobarディレクトリの中に放り込む
- すると仮想マシンの Ubuntu 内では `/vagrant/ics-exp-lan` が見えるようになるはずである
 - Windows と Ubuntu 間で自動でファイル共有するように Vagrant がやっている
 - 仮想マシンを再起動したりする必要はない
- あとは、例えば以下のようにして実行していけば良い。詳細はics-exp-lan/README.mdを読むこと
 - `cd /vagrant/ics-exp-lan`
 - `make build`
 - `make start`
 - `make login-node1`
 - ...

Docker、Ubuntu、Windows 間でのファイル共有

- 指示通りに進めてきた場合、Windowsの上のUbuntuの上でDockerが動いている状態になっているはずである
- Dockerの中の `/home/ubuntu/share` は Ubuntuの `/vagrant/ics-exp-lan/share` と自動で同期される
- さらにここは Windowsの `foobar/ics-exp-lan/share` (場所はデスクトップのはず) とも同期される
- ファイルの確認やコピーには、これらを上手く使うこと

WSL2の上でDockerを動かす

注意: 実験で想定した帯域や遅延の制限が行われないので、
現状は参考程度にしておくこと

注意事項

- Windows 10 以降であること
- 64bit版であること
- Windows Version 20H1 (2004) 以降であること
- 最小4GBのメモリ
 - また、メモリに余裕がなさそうなら不必要なアプリケーションは落として実験すること
- 管理者権限を持つユーザーであること

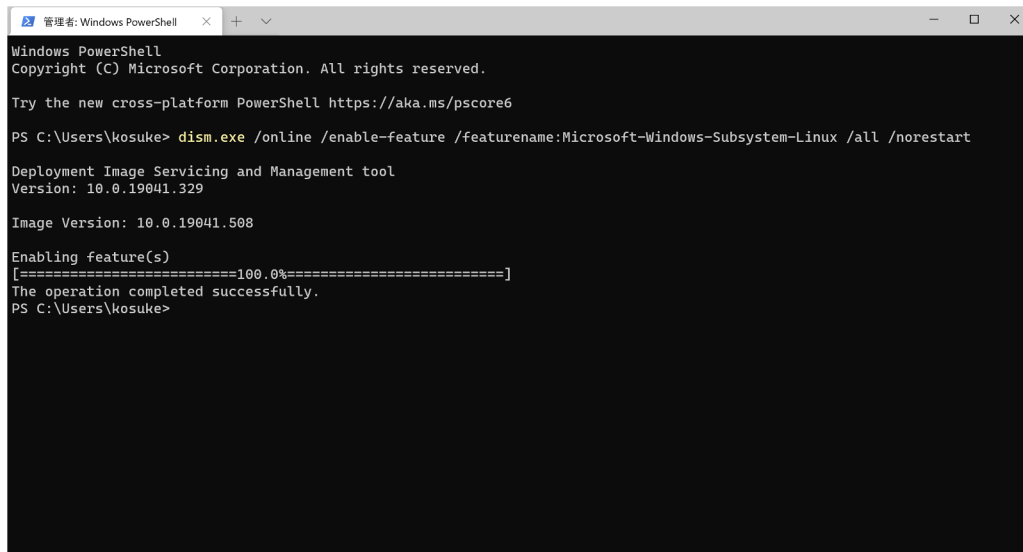
Windowsの最新

- Home版でWSL2を使う場合、以下の条件が必須
 - x64 システムの場合:バージョン 1903 以降、ビルド 18362 以上
 - ARM64 システムの場合:バージョン 2004 以降、ビルド 19041 以上
- 確認方法
 - Windows ロゴ キー + R キーを押して、「winver」と入力し、[OK] を選択すると右のような表示が出る
 - 自分のPCがx64なのかARM64なのかは、Windows Terminal などで`systeminfo | find "System Type"`と打てば情報が表示される
- Windowsのバージョンが低かった場合、更新しておくこと



WSL2の有効化 / Step 1

- 管理者権限でWindows TerminalでPowerShellを開き、以下を実行
`dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart`
- 以下のように表示されればOK



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

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\kosuke> dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart

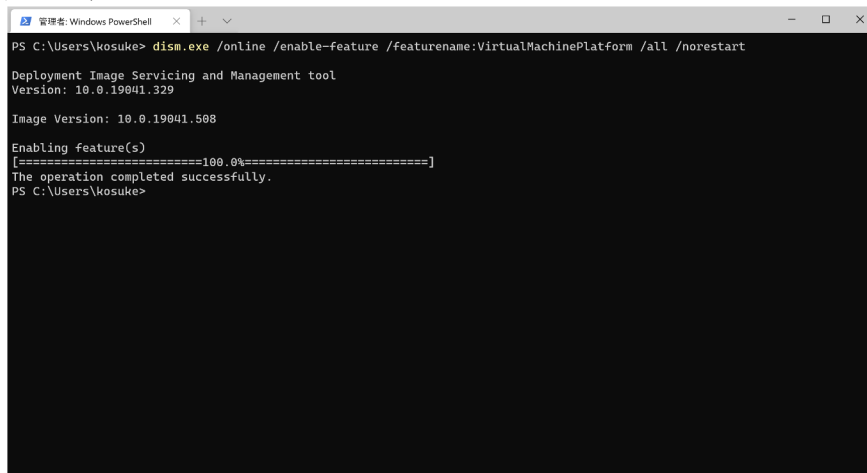
Deployment Image Servicing and Management tool
Version: 10.0.19041.329

Image Version: 10.0.19041.508

Enabling feature(s)
[=====100.0%=====]
The operation completed successfully.
PS C:\Users\kosuke>
```

WSL2の有効化 / Step 2

- 同じく管理者権限のWindows Terminal (のPowerShell) で以下を実行
`dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart`
- 以下のように表示されればOK



```
PS C:\Users\kosuke> dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
Deployment Image Servicing and Management tool
Version: 10.0.19041.329

Image Version: 10.0.19041.508

Enabling feature(s)
[=====100.0%=====]
The operation completed successfully.
PS C:\Users\kosuke>
```

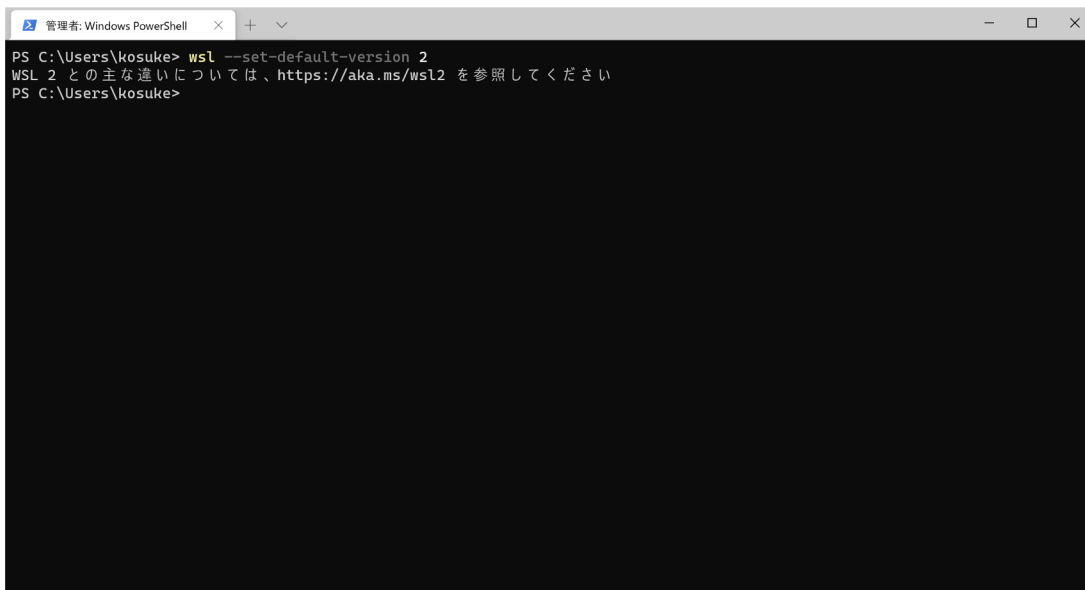
- その後、Windowsを再起動する

WSL2の有効化 / Step 3

- 以下のLinuxカーネル更新プログラムをダウンロード & インストールする
 - https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi
 - (ARM64版: https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_arm64.msi)

WSL2の有効化 / Step 4

- 管理者権限で Windows Terminal (のPowerShell) を開いて以下を実行
`wsl --set-default-version 2`
- 以下のように表示されればOK



```
管理: Windows PowerShell
PS C:\Users\kosuke> wsl --set-default-version 2
WSL 2 との主な違いについては、https://aka.ms/wsl2 を参照してください
PS C:\Users\kosuke>
```

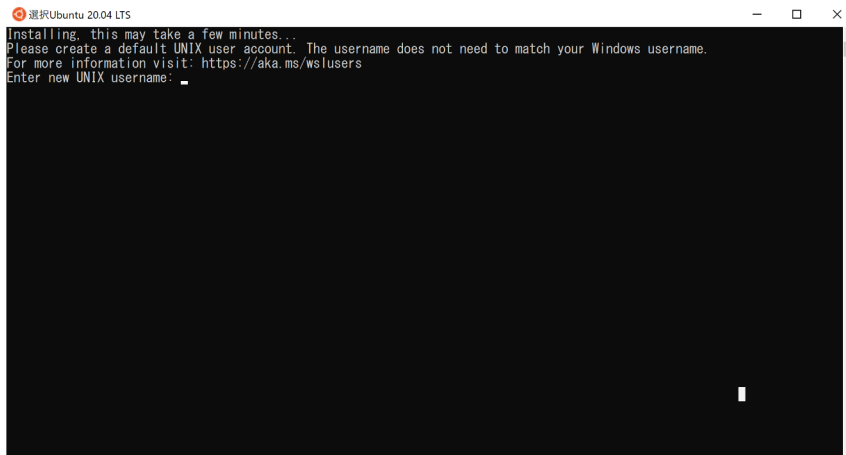
WSL2でLinuxのインストール 1/2

- Microsoftストアから適当なLinuxをインストール
 - 今回の実験ではMakefileさえちゃんと動けば良いので、大体のLinuxで問題ないはず
 - 例としてUbuntu 20.04 LTSを選択する
 - 「入手」ボタンを押してダウンロード。その後「起動」ボタンに変わるので、再度クリック



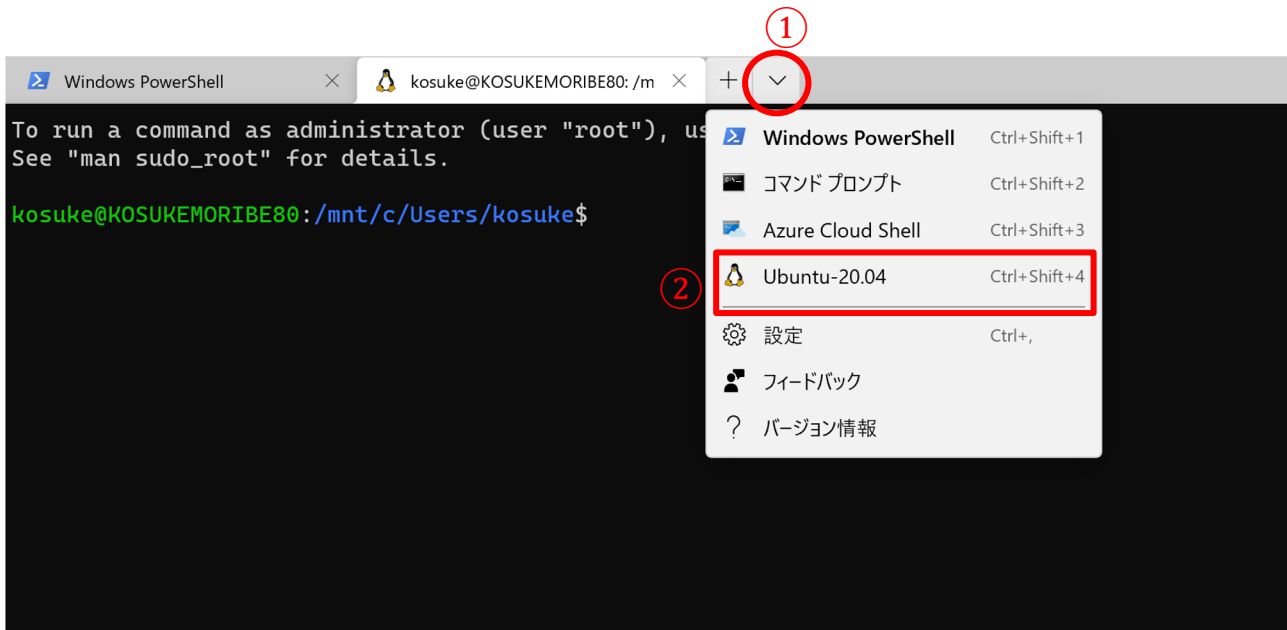
WSL2でLinuxのインストール 2/2

- 「起動」をクリックすると以下の画面が出てくるので、それに従ってLinuxにユーザー名とパスワードを設定する
 - ユーザー名は英語にしておくことを推奨する
 - パスワードは2回聞かれるので同じものを設定する
 - 設定が完了したらそのターミナルでLinuxが使える。いったん閉じてもスタートメニューからインストールしたLinuxを探してクリックすれば再度ターミナルは表示できる



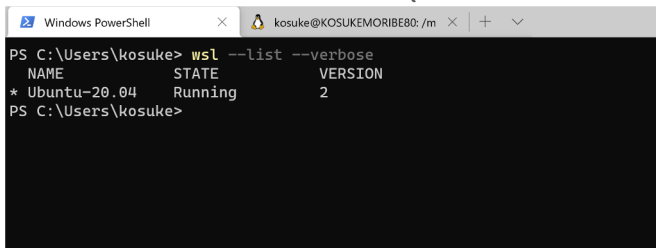
おまけ: Windows TerminalでWSL2にアクセス

- Windows Terminalから以下の①->②の順にクリックすれば、WSL2でインストールしたLinuxでターミナルを開くことができる
 - Linuxインストール直後は1度Windows Terminalを終了させる必要があるかも



確認: インストールしたLinuxのWSLのバージョン

- 管理者権限で Windows Terminal (のPowerShell) を開き以下を実行
`wsl --list --verbose`
- 以下のようになっていれば良い (例はUbuntu 20.04をインストールした場合)

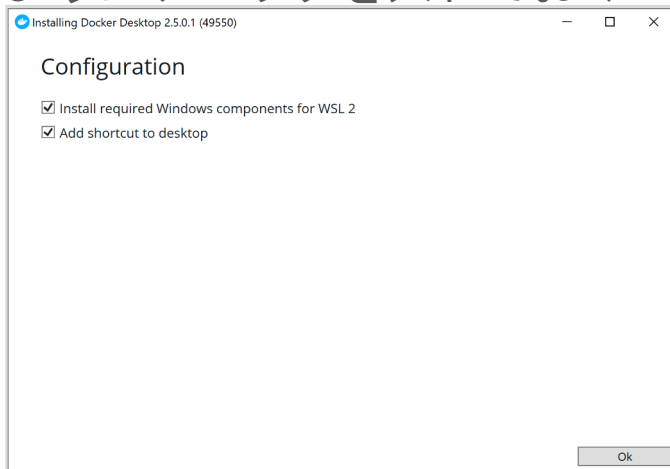


```
PS C:\Users\kosuke> wsl --list --verbose
NAME                STATE              VERSION
* Ubuntu-20.04      Running            2
PS C:\Users\kosuke>
```

- もしなっていないなら以下のように実行してVersionを2にしておくこと
`wsl --set-version Ubuntu-20.04 2`
 - この場合、`wsl --set-default-version 2` も再度実行しておくことをおすすめする

Dockerのインストール

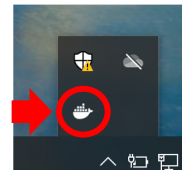
- 以下から Docker をダウンロード & インストール
 - <https://hub.docker.com/editions/community/docker-ce-desktop-windows>
- 途中、以下のようにチェックを入れておくこと



- (インストール後、ログアウトを推奨されたらそれに従いログアウト&ログイン)

Dockerの起動と終了

- 起動: スタートメニューからDockerを選択
 - スクバーに右のようにDockerアイコンが出て、ピコピコ動くアニメーションが終わったら起動完了
 - DockerのUIが表示されチュートリアル等が表示されるかもしれないが、その画面自体はいつ閉じてても問題ない。再表示させたい場合はタスクバーのDockerアイコンをクリックする
- 終了: タスクバーのDockerアイコンを右クリックして終了を選択

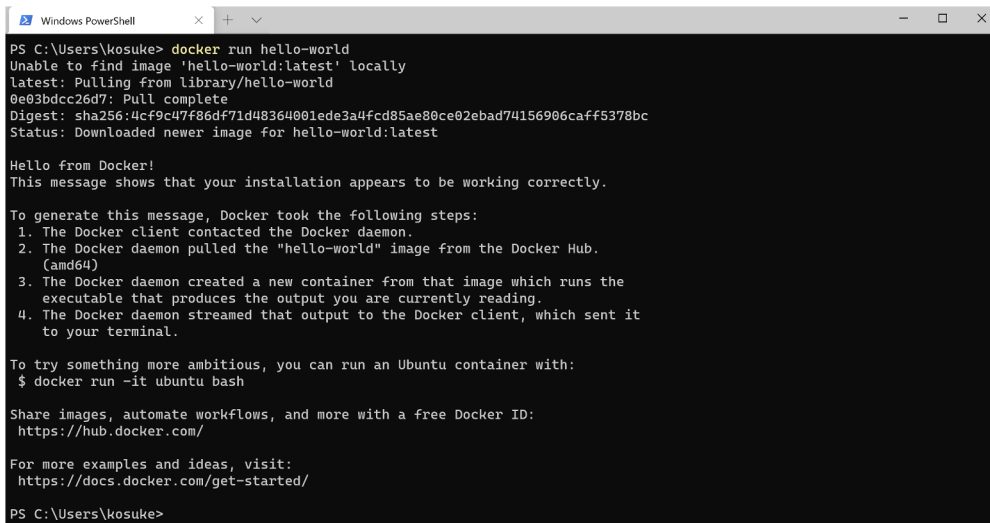


Dockerを試す (PowerShellで)

- Dockerを起動した状態で、Windows TerminalでPowerShell (管理者権限でなくて良い) を開いて、以下を実行してみる

```
docker run hello-world
```

- 以下のように表示されればOK



```
PS C:\Users\kosuke> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:4cf9c47f86df71d48364001ede3a4fcd85ae80ce02ebad74156906caff5378bc
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:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

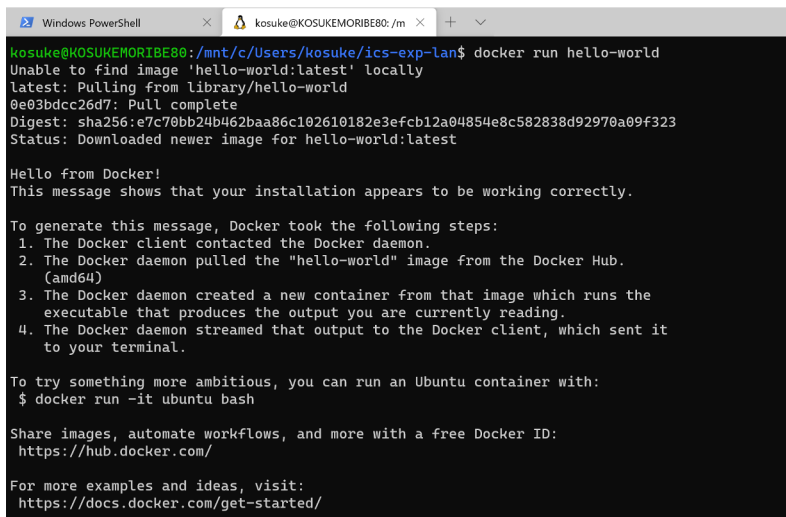
PS C:\Users\kosuke>
```

Dockerを試す (WSL2で)

- Dockerを起動した状態で、Windows Terminalで**WSL2のLinux**のターミナルを開き、以下のコマンドを実行する

```
docker run hello-world
```

- 以下のように表示されればOK



```
Windows PowerShell x kosuke@KOSUKEMORIBE80: /mnt/c/Users/kosuke/ics-exp-lan$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:e7c70bb24b462baa86c102610182e3efcb12a04854e8c582838d92970a09f323
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:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Tips: Windows Terminalからエクスプローラを起動

- PowerShellやWSL2のターミナルでは以下のコマンドで今いる場所をエクスプローラで表示できるので、覚えておこう

`explorer.exe .`

- 最後に. (ドット)が必要な点に要注意

今回配布した Docker 環境を試す

- Windowsで ics-exp-lan.zip を解凍し、C:¥Users¥ユーザー名に配置する
- するとWSL2からは /mnt/c/Users/ユーザー名/ics-exp-lan のように見えるはず
 - ユーザー名が日本語の場合はここは避けたほうが良いので、C:¥Users以下に直接置いたりしよう。その場合はWSL2からは /mnt/c/Users/ics-exp-lan などと見えるはず
- WSL2上のLinuxでmakeが使えるようにする
 - Ubuntuをインストールしているなら、`sudo apt install make` とすれば良い
 - パスワードはLinuxインストール時に自分で決めたやつ
- あとは以下のように進める。詳細は授業資料やREADME.mdを参考
 - `cd /mnt/c/Users/ユーザー名/ics-exp-lan`
 - `make build`
 - `make start` # ただし、ここで `Error: Specified qdisc not found.`とエラーが出て、想定された帯域制限が行われない点には要注意！
 - `make login-node1`
 - ...