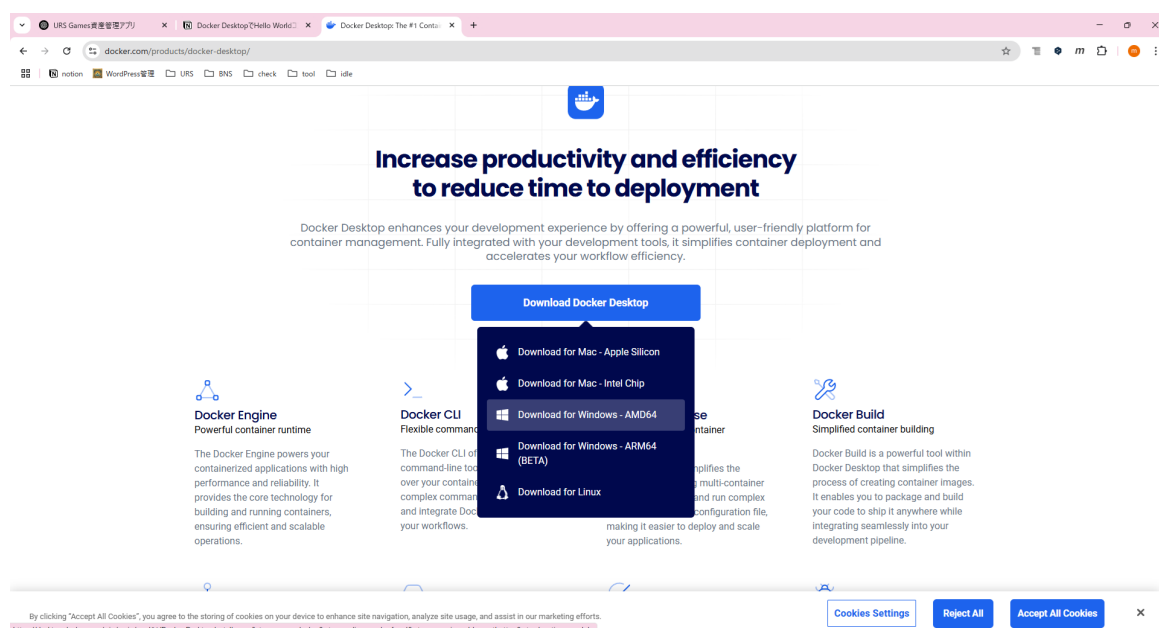




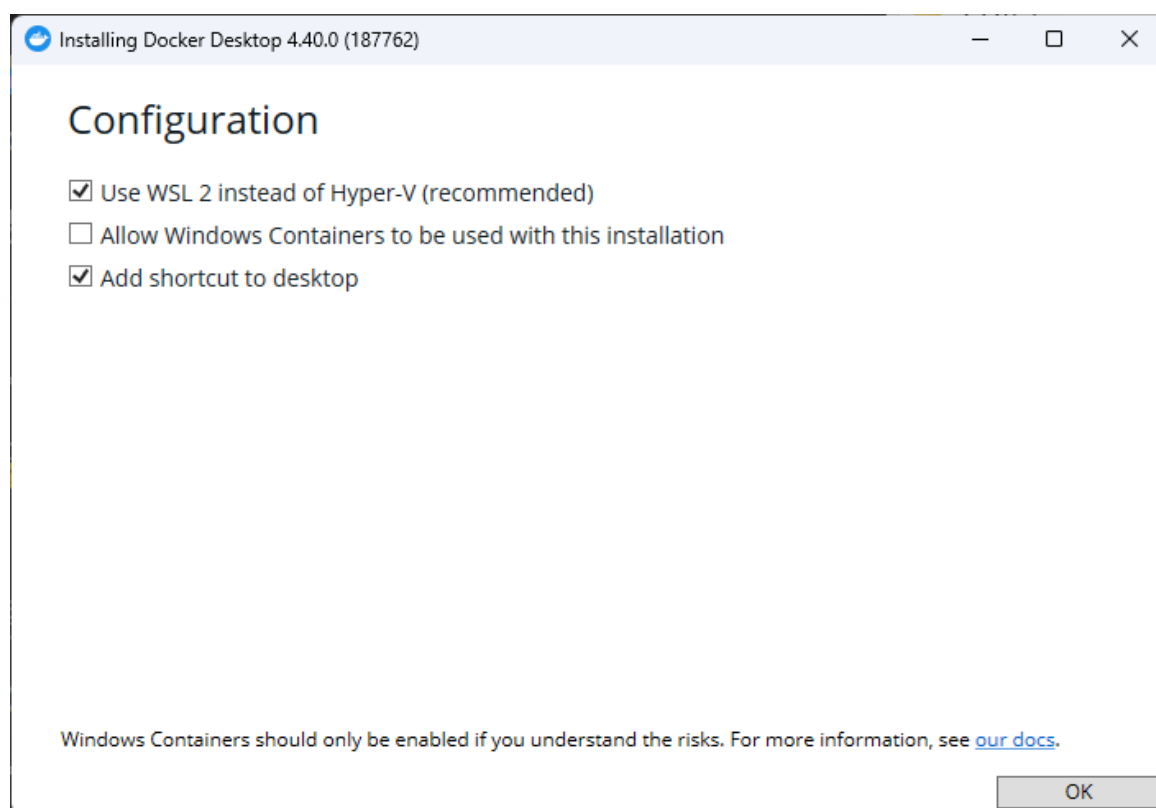
Docker DesktopでHello World コンテナを動かす手順 (Windows)

▼ Docker Desktopのインストール (Windows)

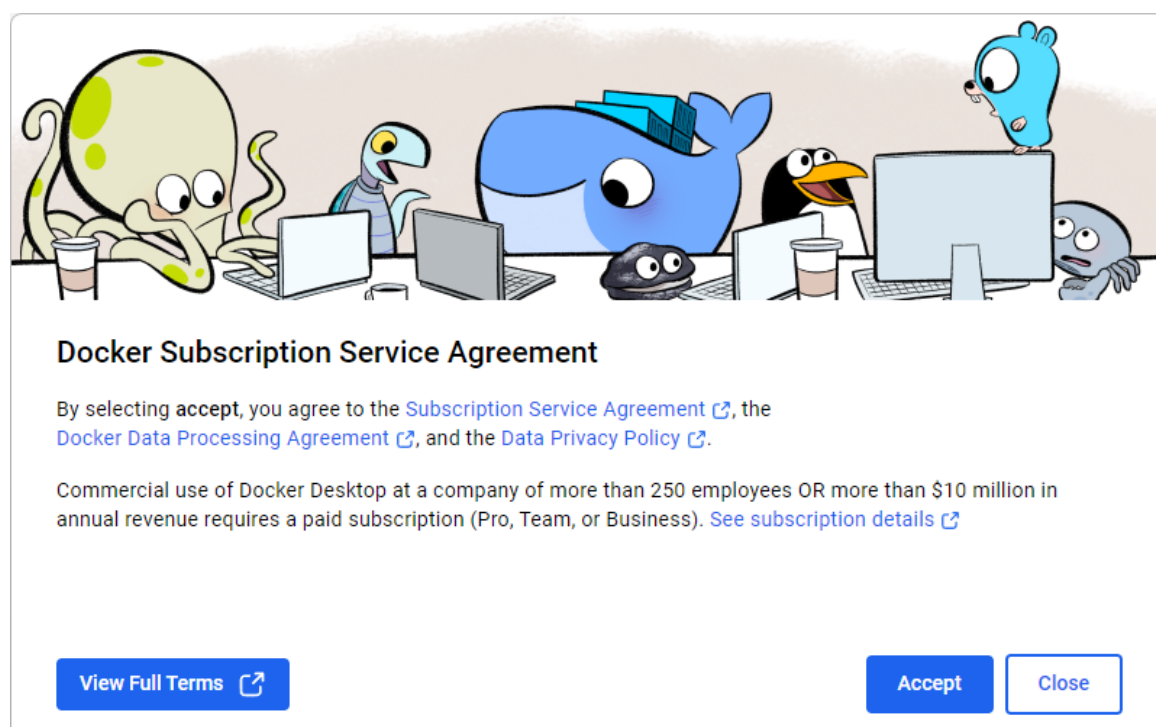
<https://www.docker.com/products/docker-desktop/> へアクセスし、Docker Desktop Installer.exe をダウンロードする。



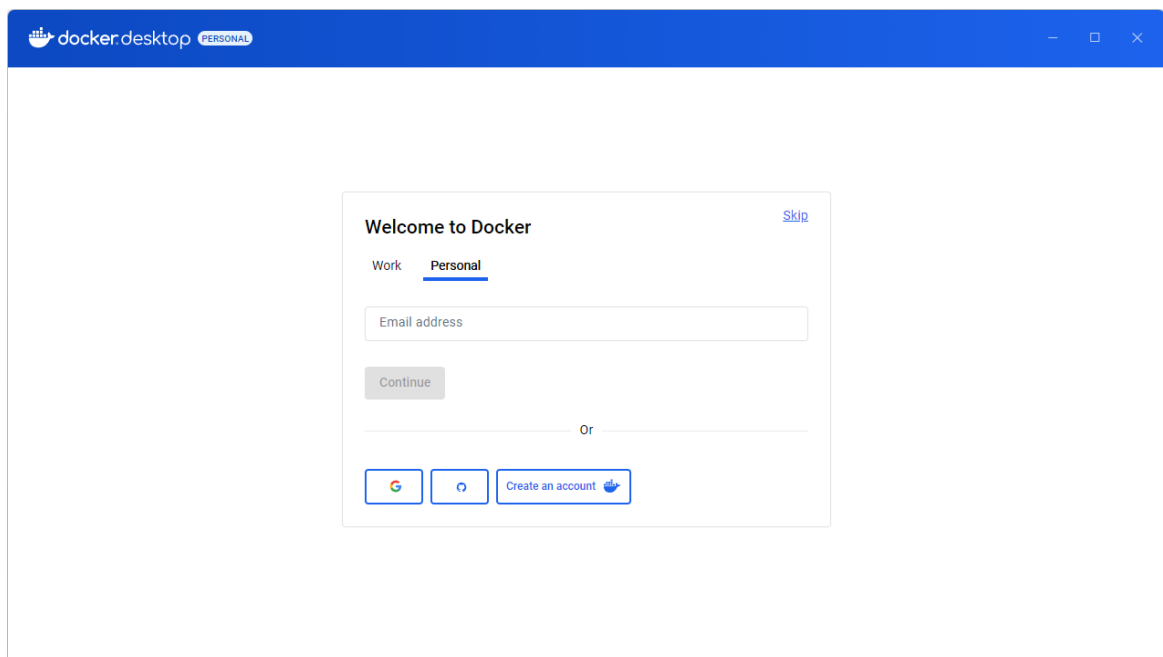
ダウンロード完了後、インストーラーを起動する。インストールウィザードに従い(デフォルトのままでOK)、インストールを開始する。



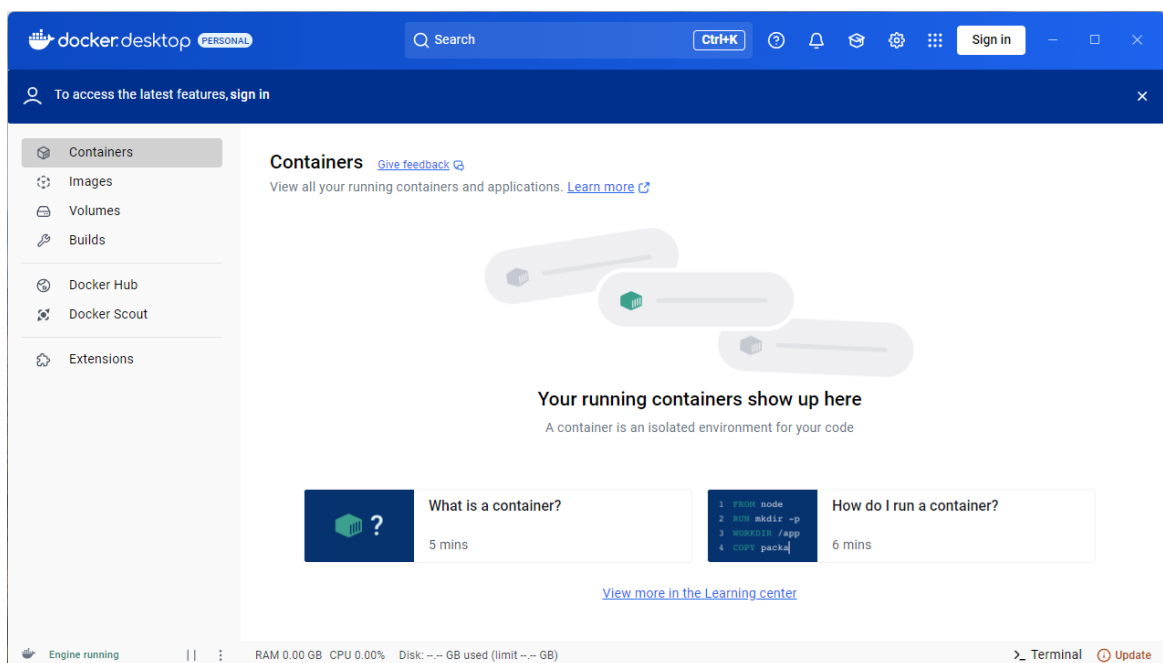
インストール完了後、再起動を行う。再起動後、Dockerの利用規約（ライセンス）に関する案内が表示されるので、Acceptする。



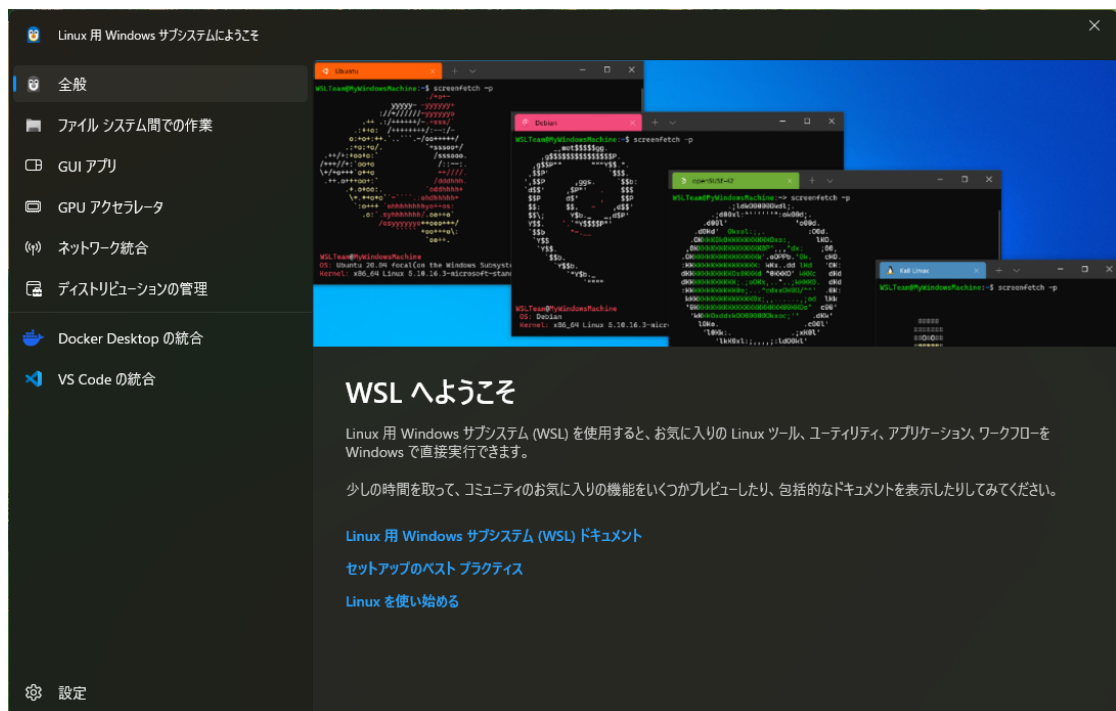
Docker Desktop の初回ログイン画面で、Docker Hubとの連携を促されるが、Skipして先に進む。(後からログインできる。)



Docker Desktopの起動成功！



▼ WSL (Windows Subsystem for Linux) **の紹介・設定画面が出てくる場合



Docker Desktopを使ってLinuxコンテナを扱う場合は、「WSL 2 の有効化」「UbuntuなどのLinuxディストリビューションのインストール」が必要。この2点を確認する。

```
wsl --list --verbose
```

結果

NAME	STATE	VERSION
* docker-desktop	Running	2

この状態は、「UbuntuなどのLinuxディストリビューションがインストールされていない」状態。なのでUbuntu をインストール。

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

PCを再起動して、wsl でUbuntuを起動できるか確認する。

```
wsl --list --verbose
```

結果

NAME	STATE	VERSION
------	-------	---------

```
* docker-desktop  Running    2
  Ubuntu          Stopped    2
```

▼ Hello World コンテナを動かす

```
docker run hello-world
```

結果

```
docker: error during connect: Head "http://%2F%2F.%2Fpipe%2FdockerD
```

「docker: error during connect」: Docker デーモン（サービス）が起動していない

ので、Docker Desktop アプリを再起動する。タスクトレイのアイコンが灰色だが「Docker Desktop Running」だったので、再度コマンドを実行

```
docker run hello-world
```

結果

```
Unable to find image 'hello-world:latest' locally
```

```
latest: Pulling from library/hello-world
```

```
e6590344b1a5: Pull complete
```

```
Digest: sha256:c41088499908a59aae84b0a49c70e86f4731e588a737f163
```

```
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:

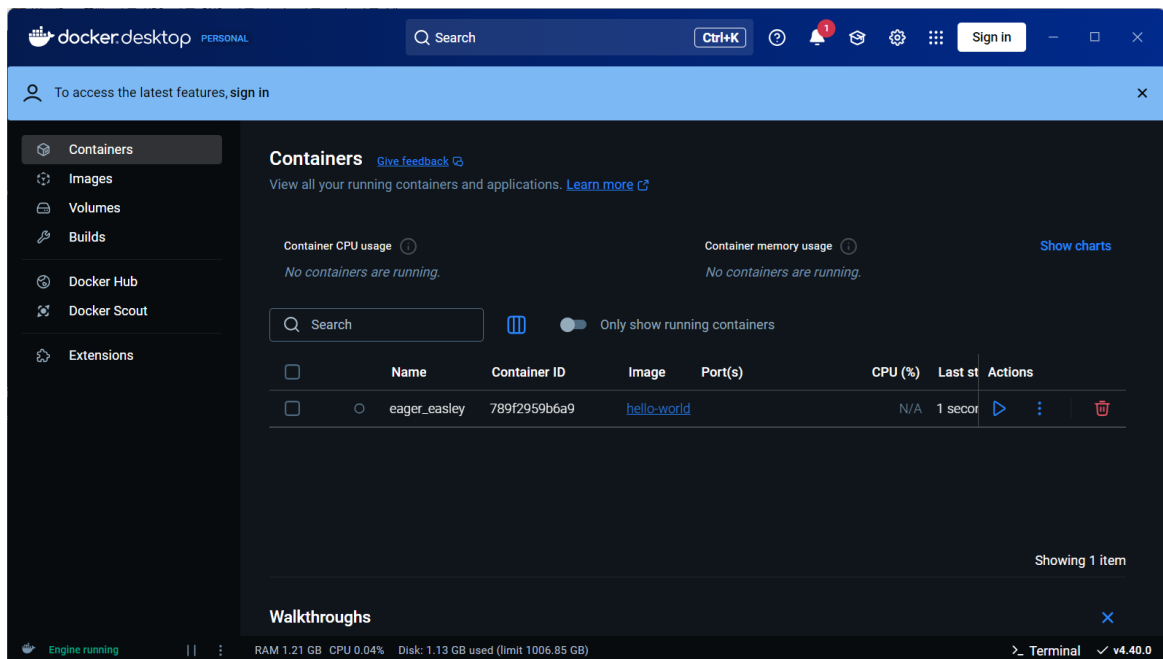
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub (amd64)
3. The Docker daemon created a new container from that image which runs executable that produces the output you are currently reading.
4. The Docker daemon streamed that output 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/>



Dockerクライアント → Dockerデーモンへの接続、hello-world イメージの取得と実行すべて問題なく完了できた。

▼ Go言語のWebサーバーコンテナをつくる(ビルドする)

▼ プロジェクトフォルダを作成し、Goモジュールの初期化

```
mkdir go-jwt-auth
cd go-jwt-auth
go mod init go-jwt-auth
```

- httpハンドラを待機させてクライアント接続を待つ、goファイル(main.go)を作成し、Buildを確認する。(作成されたmain.exeは不要)

- Dockerfileを作成

▼ コンテナをビルド・実行

```
cd go-jwt-auth
docker build -t go-jwt-auth .
```

▼ エラー。

```
ERROR: failed to solve: process "/bin/sh -c go build -o app" did not complete successfully
```

```
# 原因のコマンド
RUN go build -o app
```

ビルド対象となる main.go や go.mod が Docker イメージ内の作業ディレクトリに存在していないことがよくある原因らしいので、確実に Docker イメージ内にコピーし、作業ディレクトリを明示しておくよう Dockerfile を下記のように修正。

Dockerfile

- ローカル環境の Go が 1.23.3 なのでベースとなる公式イメージを「golang:1.22-alpin」に変更。「golang:1.23.1」もあるのでそこまであげた。

▼ エラー

```
ERROR: failed to solve: process "/bin/sh -c go mod download" did not complete successfully
```

Docker コンテナの中で go mod download を実行したときに、何かしらの理由で失敗した。失敗の理由はおそらく必要なファイル(main.go, go.mod) がコピーされていないことではないかと思われるので、下記のように、ls, cat コマンドを入れて再ビルド

```
# 原因調査用コマンド
RUN ls -la /app
RUN cat /app/main.go

# 結果
⇒ CACHED [2/7] WORKDIR /app
```

```
⇒ CACHED [3/7] COPY go.mod .  
⇒ CACHED [4/7] COPY main.go .  
⇒ [5/7] RUN ls -la /app  
⇒ [6/7] RUN cat /app/main.go
```

どうも COPY ステップがキャッシュされているのが原因で、実際の main.go がビルド時に反映されていない可能性が高そう。キャッシュ無効にしてビルドし直す。

```
docker build --no-cache -t go-jwt-auth .
```

でも失敗。というかキャッシュとかじゃなくて、Goのバージョンが合わずにビルドできなかったのが原因。

- go.mod に指定されている Go のバージョンは 1.23.3
- でも使っている Docker イメージは Go 1.23.1

⇒ バージョンが足りないのでビルドできない
のでDockerfileのFromを変更。

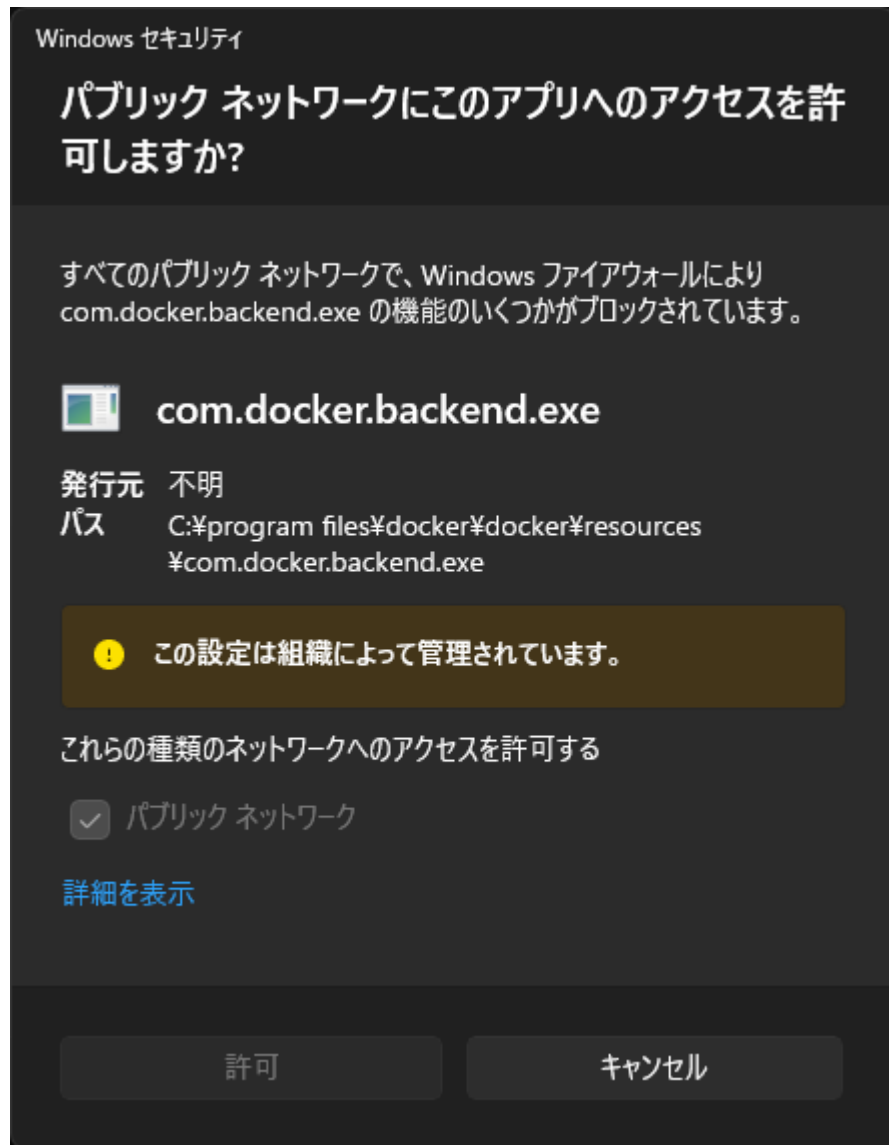
```
FROM golang:1.23.3
```

で、ようやく再ビルド成功。

▼ Go言語のWebサーバーコンテナを実行する

```
docker run -p 8080:8080 go-jwt-auth
```

Windowsから下記のようにWarningが。



Windows ファイアウォール が Docker のバックエンド実行ファイル（com.docker.backend.exe）の ネットワーク通信をブロックしようとしてる警告がでた。

管理者として PowerShell、Docker を起動して再実行したところ、ブラウザ、curlでのアクセスで挙動確認ができた

