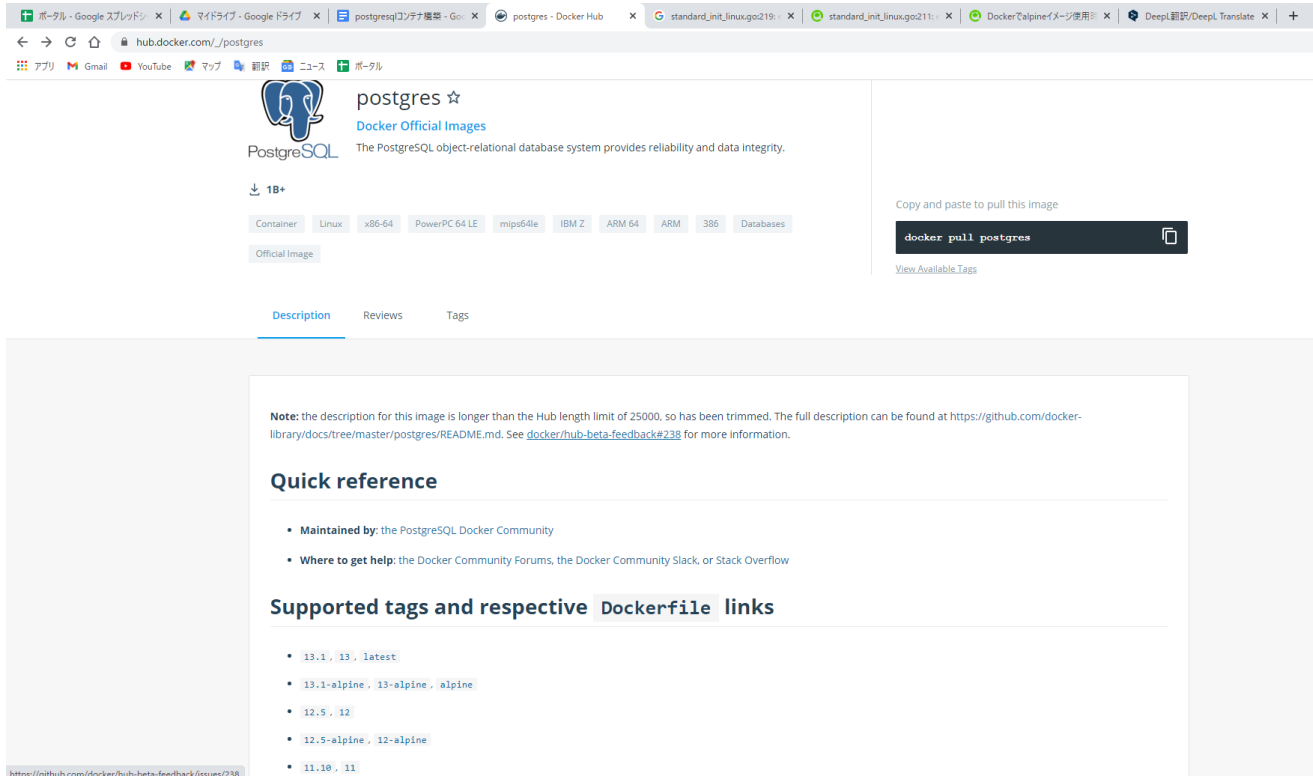


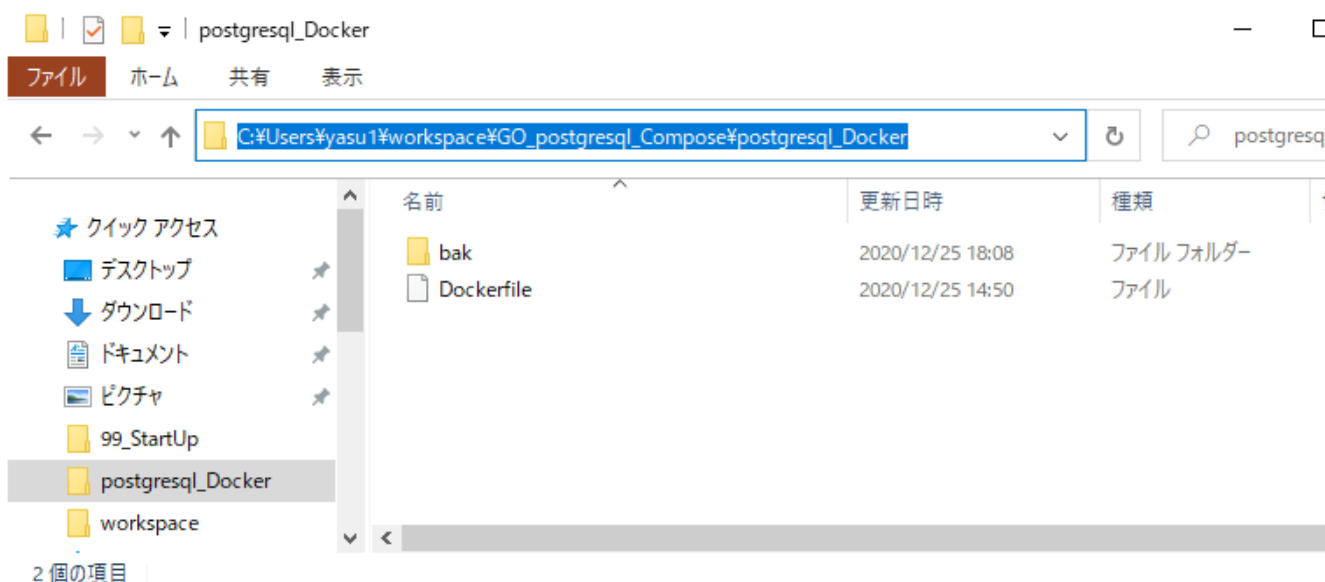
postgresqlコンテナ構築

1. Dockerfile作成

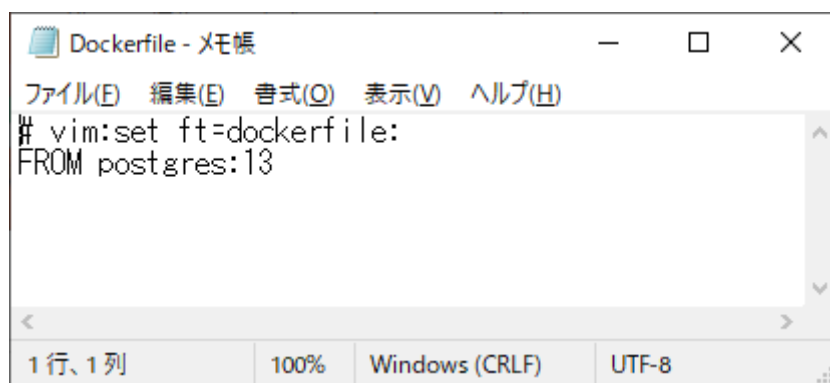
- a. 「https://hub.docker.com/_/postgres」にアクセス。
コンテナイメージ名"postgres"とタグ名"13"を確認。



- b. 「C:\Users\yasu1\workspace\GO_postgresql_Compose\postgresql_Docker\Dockerfile」ファイルを作成



- c. Dockerfileの中を以下のように記述。コンテナイメージ名"postgres"とタグ名"13"。



```
# vim:set ft=dockerfile:
FROM postgres:13
```

2. コンテナイメージのビルド

- a. Power Shellを起動し、以下のコマンドを実行する。"docker image ls ..."の結果の「REPOSITORY」(postgres)がイメージ名となる。

```
01_Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

新しいクロスプラットフォームの PowerShell をお試しください https://aka.ms/powershell

PS C:\Users\yasul> cd .\workspace\
PS C:\Users\yasul\workspace> ls

ディレクトリ: C:\Users\yasul\workspace

Mode                LastWriteTime         Length Name
----                -
d-----          2020/12/24      15:44             GO_postgresql_Compose
-a----          2020/11/02      17:32             174 hosts
-a----          2020/11/02      17:46             150 server.py

PS C:\Users\yasul\workspace> cd .\GO_postgresql_Compose\
PS C:\Users\yasul\workspace\GO_postgresql_Compose> ls

ディレクトリ: C:\Users\yasul\workspace\GO_postgresql_Compose

Mode                LastWriteTime         Length Name
----                -
d-----          2020/12/24      15:43             GO_Docker
d-----          2020/12/24      17:14             postgresql_Docker

PS C:\Users\yasul\workspace\GO_postgresql_Compose> cd .\postgresql_Docker\
PS C:\Users\yasul\workspace\GO_postgresql_Compose\postgresql_Docker> ls

ディレクトリ: C:\Users\yasul\workspace\GO_postgresql_Compose\postgresql_Docker

Mode                LastWriteTime         Length Name
----                -
-a----          2020/12/25      13:02             7562 Dockerfile

PS C:\Users\yasul\workspace\GO_postgresql_Compose\postgresql_Docker> █
```

```
01_Windows PowerShell
PS C:\Users\yasul\workspace\GO_postgresql_Compose\postgresql_Docker> docker image build ./ -t postgres
[+] Building 510.5s (14/14) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/alpine:3.12
=> [auth] library/alpine:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 1.61kB
=> [1/9] FROM docker.io/library/alpine:3.12@sha256:3c7487bf0c7af83428242db176e8f7805f2201d8fc5861f45be7a348b52
=> => resolve docker.io/library/alpine:3.12@sha256:3c7487bf0c7af83428242db176e8f7805f2201d8fc5861f45be7a348b52
=> => sha256:3c7487bf0c7af83428242db176e8f7805f2201d8fc5861f45be7a348b52 1.64kB / 1.64kB
=> => sha256:074d3636ebda5d446d0d00304c4454f408237fdecf00f60eeac00bcbfa1bac7 520B / 520B
=> => sha256:389fef7118515c70f8bc0edd50be75869942ea722ccb976507d7b007e54d5a23 1.47kB / 1.47kB
=> => sha256:801bfad3af2094d770c808815b8e2b8c1194728e5e754ef7bc764030e140cea 2.00MB / 2.00MB
=> => extracting sha256:801bfad3af2094d770c808815b8e2b8c1194728e5e754ef7bc764030e140cea
=> [2/9] RUN set -eux; addgroup -g 70 -S postgres; adduser -u 70 -S -D -G postgres -H -h /var/lib/postgresql
=> [3/9] RUN mkdir /docker-entrypoint-initdb.d
=> [4/9] RUN set -eux; wget -O postgresql.tar.bz2 "https://ftp.postgresql.org/pub/source/v13.1/postgresql-13
=> [5/9] RUN sed -ri 's/!(listen_addresses)%*=%*%S+.*!% = '*!'" /usr/local/share/postgresql/postgresql.conf
=> [6/9] RUN mkdir -p /var/run/postgresql && chown -R postgres:postgres /var/run/postgresql && chmod 2777 /var/
=> [7/9] RUN mkdir -p /var/lib/postgresql/data && chown -R postgres:postgres /var/lib/postgresql/data && ch
=> [8/9] COPY docker-entrypoint.sh /usr/local/bin/
=> exporting to image
=> => exporting layers
=> => writing image sha256:75ce7e4fe4ec8538f74ff3b12572ca9bed04cc52d1e19ae78fc45ad15fa154f
=> => pushing to docker.io/library/postgres
PS C:\Users\yasul\workspace\GO_postgresql_Compose\postgresql_Docker> docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
postgres latest 75ce7e4fe4ec 3 minutes ago 159MB
PS C:\Users\yasul\workspace\GO_postgresql_Compose\postgresql_Docker> .
```

3. postgresqlのストレージを永続化

```
01_Windows PowerShell
PS C:\Users\yasul\workspace\GO_postgresql_Compose\postgresql_Docker> docker container e
xec -it postgresqlContainerSys01 bash
root@bfc1bf7cbc29:/# psql -U postgres
psql (13.1 (Debian 13.1-1.pgdg100+1))
Type "help" for help.

postgres=# CREATE TABLE products (
postgres=#     product_no integer,
postgres=#     name text,
postgres=#     price numeric
postgres=# );
ERROR: relation "products" already exists
postgres=# INSERT INTO products (product_no, name, price) VALUES
postgres=#     (1, 'Cheese', 9.99),
postgres=#     (2, 'Bread', 1.99),
postgres=#     (3, 'Milk', 2.99);
INSERT 0 3
postgres=# select * from products;
 product_no | name  | price
-----+-----+-----
          1 | Cheese |  9.99
          2 | Bread |  1.99
          3 | Milk  |  2.99
          1 | Cheese |  9.99
          2 | Bread |  1.99
          3 | Milk  |  2.99
(6 rows)

postgres=# exit
root@bfc1bf7cbc29:/# exit
exit
PS C:\Users\yasul\workspace\GO_postgresql_Compose\postgresql_Docker> .
```

a.

コンテナイメージの稼働と「NAMES」を確認。イメージ名(postgres)は構文の末尾でなければならぬ。

```

01_Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

新しいクロスプラットフォームの PowerShell をお試しください https://aka.ms/pscore6

PS C:\Users\yasu1> docker run --rm -e POSTGRES_PASSWORD=mysecretpassword -d --name postgresqlContainerSys01 postgres
f0bb9db30f854b42c3d79e1a17b982f2254d01502b842e7e676cd2e9090d4f98
PS C:\Users\yasu1> docker container ls -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
NAME
f0bb9db30f85   postgres "docker-entrypoint.s...  5 seconds ago Up 4 seconds  5432
/tcp        postgresqlContainerSys01
PS C:\Users\yasu1>

```

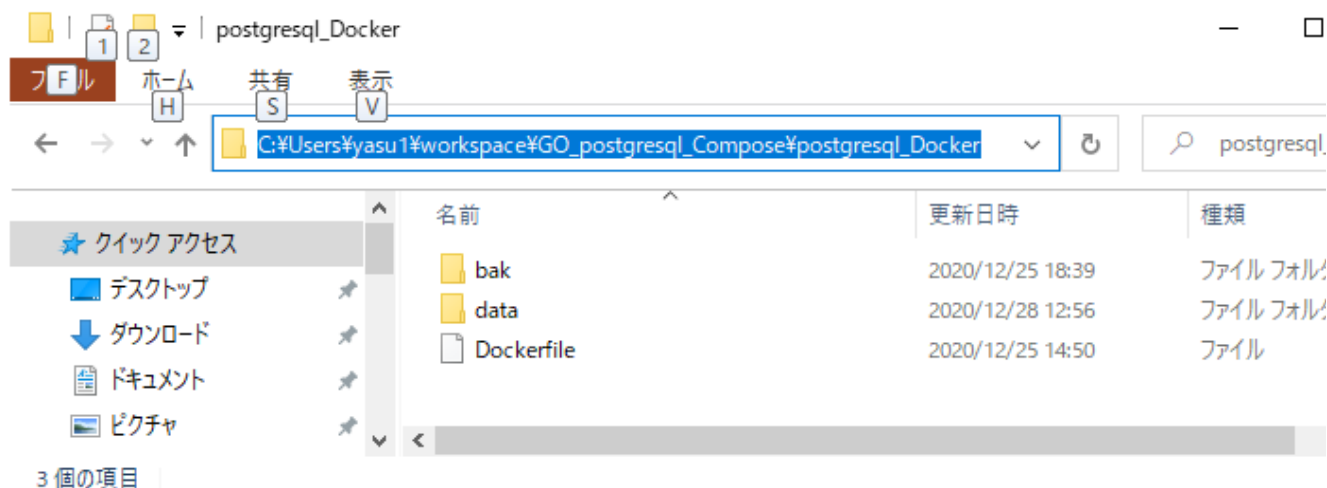
- b. コンテナ内にリモートログイン+bash実行し、postgresql永続化するデータフォルダを確認

```

01_Windows PowerShell
PS C:\Users\yasu1> docker container exec -it postgresqlContainerSys01 bash
root@f0bb9db30f85:/# cd /var/lib/postgresql/data/
root@f0bb9db30f85:/var/lib/postgresql/data# ls
base          pg_ident.conf  pg_serial      pg_tblspc      postgresql.auto.conf
global        pg_logical      pg_snapshots   pg_twophase     postgresql.conf
pg_commit_ts  pg_multixact   pg_stat        PG_VERSION     postmaster.opts
pg_dynshmem   pg_notify      pg_stat_tmp    pg_wal         postmaster.pid
pg_hba.conf   pg_replslot    pg_subtrans    pg_xact
root@f0bb9db30f85:/var/lib/postgresql/data# exit
exit

```

- c. “C:\Users\yasu1\workspace\GO_postgresql_Compose\postgresql_Docker\data”フォルダを作成



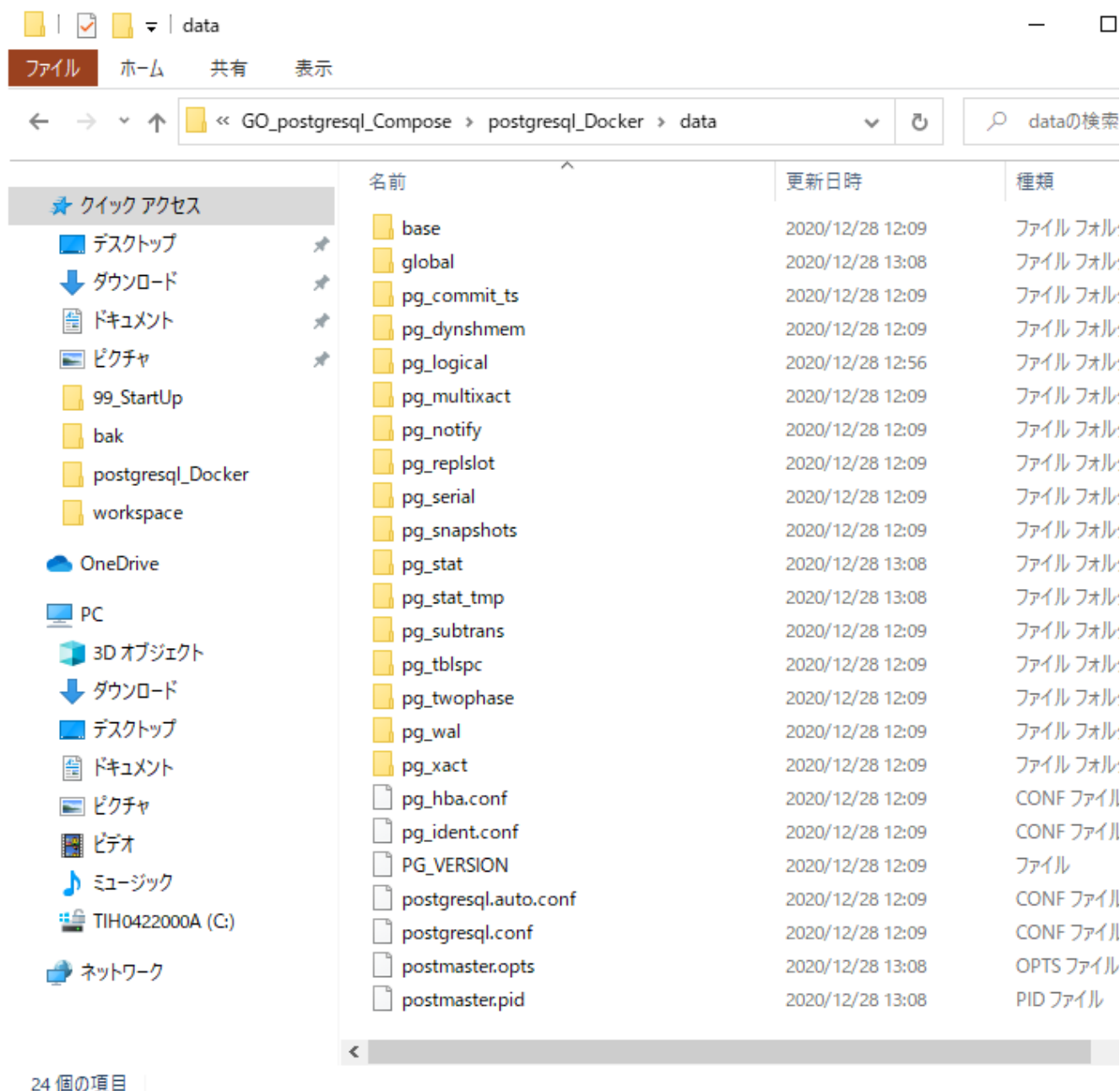
- d. コンテナをstopし、volumeをマウントしながらコンテナ起動

```

01_Windows PowerShell
PS C:\Users\yasu1\workspace\GO_postgresql_Compose\postgresql_Docker> docker container s
top postgresqlContainerSys01
postgresqlContainerSys01
PS C:\Users\yasu1\workspace\GO_postgresql_Compose\postgresql_Docker> docker run --rm -e
POSTGRES_PASSWORD=mysecretpassword -d -v C:\Users\yasu1\workspace\GO_postgresql_Compos
e\postgresql_Docker\data:/var/lib/postgresql/data --name postgresqlContainerSys01 postg
res
bfc1bf7cbc29dde783dd955bf013f51239cb5a1692956feb7aadf6663fee300e
PS C:\Users\yasu1\workspace\GO_postgresql_Compose\postgresql_Docker>

```

- e. “C:\Users\yasu1\workspace\GO_postgresql_Compose\postgresql_Docker\data”フォルダ 以下に
フォルダ・ファイルが作成される



4. postgresqlにデータを登録

- a. コンテナでbash、psql実行し、CREATE TABLE、INSERT、SELECTを実行。

