DevOps and Cloud

March 2025

Advanced Containerization Concepts

Homework (M2)

Vasil Atanasov

@VasAtanasov

Summary

To repeat the steps from the practice, I will use a PostgreSQL Docker image. One key point, which will become evident in the steps involving volumes, is that each PostgreSQL container instance sharing a common data volume with another instance will cause the PostgreSQL server to detect this and create a copy of the volume or fail. If the database is modified by one server, the other server will not see the changes. To see the changes a custom docker image where only PostgreSQL client is installed whit which we will connect to the remote server will be queried.

Postgres Docker Image: https://github.com/docker-library/postgres

Network and Volumes

Networks

Default Network

Containers on the default bridge network cannot resolve each other by container name, unlike user-defined bridge networks.

```
docker network ls
docker network inspect bridge
```

```
docker run -d \
  --name postgres1 \
  -e POSTGRES_PASSWORD=Password1 \
  -e POSTGRES_DB=bulgaria \
  -p 5432:5432 \
  postgres

docker run -d \
  --name postgres2 \
  -e POSTGRES_PASSWORD=Password1 \
  -e POSTGRES_DB=bulgaria \
  -p 5433:5432 \
  postgres
```

```
docker network inspect bridge -f '{{range .Containers}}{{.Name}}, {{println .IPv4Address}}{{end}}'
```

```
vagrant@docker:~$ docker run -d \
  --name postgres1 \
  -e POSTGRES_PASSWORD=Password1 \
  -e POSTGRES_DB=bulgaria \
  -p 5432:5432 \
  postgres
Unable to find image 'postgres:latest' locally
latest: Pulling from library/postgres
dad67da3f26b: Pull complete
eb3a531023c8: Pull complete
05b641b3bdab: Pull complete
64e8f1b2b243: Pull complete
603ef9fcdd8e: Pull complete
8a1f652e0c97: Pull complete
c6def2c6e21d: Pull complete
b47a445a47f0: Pull complete
c95f49cc11b3: Pull complete
3664068a9b37: Pull complete
abfd68ef219e: Pull complete
928d00623a6e: Pull complete
db3ab53631e4: Pull complete
f4ce9941f6e3: Pull complete
Digest: sha256:6cf6142afacfa89fb28b894d6391c7dcbf6523c33178bdc33e782b3b533a9342
Status: Downloaded newer image for postgres:latest
8dee23e243d7e3cacf49f639a4616588c745c9e3f38a3874875647f3b55597cb
vagrant@docker:~$ docker run -d \
  --name postgres2 \
  -e POSTGRES_PASSWORD=Password1 \
  -e POSTGRES_DB=bulgaria \
  -p 5433:5432 \
  postgres
d60ffc14aa747290b3ff8aface7f91862b3907466d13a2a408452ed3fec83dff
vagrant@docker:~$ docker network inspect bridge -f '{{range .Containers}}{{.Name}}, {{println .IPv4Address}}{{end}}'
postgres1, 172.17.0.2/16
postgres2, 172.17.0.3/16
```

We can query for the version of the PostgreSQL server running on postgres1 from postgres2 container using only IP address of the container because default bridge network does not support name resolution.

```
docker exec -it postgres1 psql -h 172.17.0.2 -U postgres -c "SELECT version();"
```

```
docker exec -it postgres2 psql -h 172.17.0.3 -U postgres -c "SELECT version();"
```

```
vagrant@docker:~$ docker exec -it postgres1 psql -h 172.17.0.2 -U postgres -c "SELECT version();"
Password for user postgres:
                                                 version
---------------------------------------------------------------------------------------------------------
 PostgreSQL 17.5 (Debian 17.5-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
(1 row)
```

```
vagrant@docker:~$ docker exec -it postgres2 psql -h 172.17.0.3 -U postgres -c "SELECT version();"
Password for user postgres:
                                                 version
---------------------------------------------------------------------------------------------------------
 PostgreSQL 17.5 (Debian 17.5-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
(1 row)
```

If we try to use the container name as host, we will get an error.

```
vagrant@docker:~$ docker exec -it postgres1 psql -h postgres2 postgres -c "SELECT version();"
psql: error: could not translate host name "postgres2" to address: Name or service not known
vagrant@docker:~$
```

Custom Network

Now we will create a custom bridge network **pg-net.**

```
docker network create -d bridge --subnet 10.0.0.0/24 pg-net
```

The two PostgreSQL containers are running so we can connect them to the new network.

```
docker network connect pg-net postgres1
docker network connect pg-net postgres2
```

Or we can stop them and start them again with the –network option passing the name of the network.

```
docker run -d \
  --name postgres1 \
  --network pg-net \
  -e POSTGRES_DB=bulgaria \
  -e POSTGRES_PASSWORD=Password1 \
  -p 5432:5432 \
  postgres
```

```
docker network inspect pg-net -f '{{range .Containers}}{{.Name}}, {{println .IPv4Address}}{{end}}'
```

```
vagrant@docker:~$ docker network create -d bridge --subnet 10.0.0.0/24 pg-net
379477fec7b27c817322b1279eed0693e39627c5c8f0a0d2359a58d758f2604a
vagrant@docker:~$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
027e839e7efb   bridge    bridge    local
dc4d686d7222   host      host      local
c02d941938f4   none      null      local
379477fec7b2   pg-net    bridge    local
vagrant@docker:~$ docker network connect pg-net postgres1
vagrant@docker:~$ docker network connect pg-net postgres2
vagrant@docker:~$ docker network inspect pg-net -f '{{range .Containers}}{{.Name}}, {{println .IPv4Address}}{{end}}'
postgres1, 10.0.0.2/24
postgres2, 10.0.0.3/24

vagrant@docker:~$
```

```
vagrant@docker:~$ docker exec -it postgres1 psql -h postgres2 -U postgres -c "SELECT version();"
Password for user postgres:
                                                 version
---------------------------------------------------------------------------------------------------------
 PostgreSQL 17.5 (Debian 17.5-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
(1 row)
```

Volumes

On The Fly

Note: Postgres Dockerfile has an instruction to create a volume that is why there is a volume already created on the fly when run.

https://github.com/docker-library/postgres/blob/38b3c10a487945e08b7f63dee25dc4f7b86a79d1/Dockerfile-debian.template#L200

```
ENV PGDATA /var/lib/postgresql/data
# this 1777 will be replaced by 0700 at runtime (allows semi-arbitrary "--user" values)
RUN install --verbose --directory --owner postgres --group postgres --mode 1777 "$PGDATA"
VOLUME /var/lib/postgresql/data
```

That is why when the image is run it will create a volume on the fly.

```
vagrant@docker:~$ docker volume ls
DRIVER    VOLUME NAME
vagrant@docker:~$ docker run -d \
  --name postgres1 \
  --network pg-net \
  -e POSTGRES_DB=bulgaria \
  -e POSTGRES_PASSWORD=Password1 \
  -p 5432:5432 \
  postgres
e089cf89af456308493f5800324fea5cad6ee8dc8a44d3224c22753740ddcd3d
vagrant@docker:~$ docker volume ls
DRIVER    VOLUME NAME
local     3cdc4dd828540e0687896e8ac2fb81085c43013310083c30ae430df46212180e
vagrant@docker:~$
```

We will populate the database and start a second container using the volume from the first container.

```
docker exec -i postgres1 psql -U postgres -d bulgaria <<-EOSQL
CREATE TABLE cities (
          id SERIAL PRIMARY KEY,
          city_name VARCHAR(50),
          population INT
);

INSERT INTO cities (city_name, population) VALUES ('София', 1248452);
INSERT INTO cities (city_name, population) VALUES ('Пловдив', 343070);
INSERT INTO cities (city_name, population) VALUES ('Варна', 332686);
INSERT INTO cities (city_name, population) VALUES ('Бургас', 199571);
INSERT INTO cities (city_name, population) VALUES ('Русе', 137533);
INSERT INTO cities (city_name, population) VALUES ('Стара Загора', 124599);
INSERT INTO cities (city_name, population) VALUES ('Плевен', 93214);
INSERT INTO cities (city_name, population) VALUES ('Сливен', 83740);
INSERT INTO cities (city_name, population) VALUES ('Добрич', 79269);
INSERT INTO cities (city_name, population) VALUES ('Шумен', 72342);
EOSQL
```

```
vagrant@docker:~$ docker volume ls
DRIVER    VOLUME NAME
vagrant@docker:~$ docker run -d \
  --name postgres1 \
  --network pg-net \
  -e POSTGRES_DB=bulgaria \
  -e POSTGRES_PASSWORD=Password1 \
  -p 5432:5432 \
  postgres
e089cf89af456308493f5800324fea5cad6ee8dc8a44d3224c22753740ddcd3d
vagrant@docker:~$ docker volume ls
DRIVER    VOLUME NAME
local     3cdc4dd828540e0687896e8ac2fb81085c43013310083c30ae430df46212180e
vagrant@docker:~$ docker exec -i postgres1 psql -U postgres -d bulgaria <<-EOSQL
CREATE TABLE cities (
        id SERIAL PRIMARY KEY,
        city_name VARCHAR(50),
        population INT
);

INSERT INTO cities (city_name, population) VALUES ('София', 1248452);
INSERT INTO cities (city_name, population) VALUES ('Пловдив', 343070);
INSERT INTO cities (city_name, population) VALUES ('Варна', 332686);
INSERT INTO cities (city_name, population) VALUES ('Бургас', 199571);
INSERT INTO cities (city_name, population) VALUES ('Русе', 137533);
INSERT INTO cities (city_name, population) VALUES ('Стара Загора', 124599);
INSERT INTO cities (city_name, population) VALUES ('Плевен', 93214);
INSERT INTO cities (city_name, population) VALUES ('Сливен', 83740);
INSERT INTO cities (city_name, population) VALUES ('Добрич', 79269);
INSERT INTO cities (city_name, population) VALUES ('Шумен', 72342);
EOSQL
CREATE TABLE
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
vagrant@docker:~$ docker volume ls
DRIVER    VOLUME NAME
local     3cdc4dd828540e0687896e8ac2fb81085c43013310083c30ae430df46212180e
```

```
vagrant@docker:~$ docker run -d \
  --name postgres2 \
  --network pg-net \
  -e POSTGRES_DB=bulgaria \
  -e POSTGRES_PASSWORD=Password1 \
  -p 5433:5432 \
  --volumes-from postgres1 \
  postgres
a3630e7fe7da47f88887ecba532e0df63713ce5b5011d1a5c3fa6c8126ff8344
vagrant@docker:~$ docker volume ls
DRIVER    VOLUME NAME
local     3cdc4dd828540e0687896e8ac2fb81085c43013310083c30ae430df46212180e
vagrant@docker:~$
```

We can see still only one volume exists.

If we query the database, we will get:

```
docker exec -it postgres2 psql \
  -h postgres1 \
  -U postgres \
  -d bulgaria \
  -c "SELECT * FROM cities;"
```

```
vagrant@docker:~$ docker exec -it postgres2 psql \
  -h postgres1 \
  -U postgres \
  -d bulgaria \
  -c "SELECT * FROM cities;"
Password for user postgres:
 id |  city_name   | population
----+--------------+------------
  1 | София        |    1248452
  2 | Пловдив      |     343070
  3 | Варна        |     332686
  4 | Бургас       |     199571
  5 | Русе         |     137533
  6 | Стара Загора |     124599
  7 | Плевен       |      93214
  8 | Сливен       |      83740
  9 | Добрич       |      79269
 10 | Шумен        |      72342
(10 rows)
```

Attach a Prepopulated (Existing) Folder

Due to the nature of Postgres, shared data directory is not allowed, and Postgres makes a copy of the data. If a row is deleted in the first container database, it will not affect the database of the second container. It is recommended to connect to the database container as a client. This way only one server manages the db. That is why we will run the first container, populate it with data and then run the second container. When we query the database in the second container we will see there is data there.

```
docker run -d \
  --name postgres1 \
  --network pg-net \
  -e POSTGRES_DB=bulgaria \
  -e POSTGRES_PASSWORD=Password1 \
  -p 5432:5432 \
  -v $(pwd)/postgres-data:/var/lib/postgresql/data \
  postgres

docker exec -i postgres1 psql -U postgres -d bulgaria <<-EOSQL
CREATE TABLE cities (
          id SERIAL PRIMARY KEY,
          city_name VARCHAR(50),
          population INT
);
```

```
INSERT INTO cities (city_name, population) VALUES ('София', 1248452);
INSERT INTO cities (city_name, population) VALUES ('Пловдив', 343070);
INSERT INTO cities (city_name, population) VALUES ('Варна', 332686);
INSERT INTO cities (city_name, population) VALUES ('Бургас', 199571);
INSERT INTO cities (city_name, population) VALUES ('Русе', 137533);
INSERT INTO cities (city_name, population) VALUES ('Стара Загора', 124599);
INSERT INTO cities (city_name, population) VALUES ('Плевен', 93214);
INSERT INTO cities (city_name, population) VALUES ('Сливен', 83740);
INSERT INTO cities (city_name, population) VALUES ('Добрич', 79269);
INSERT INTO cities (city_name, population) VALUES ('Шумен', 72342);
EOSQL

docker run -d \
  --name postgres2 \
  --network pg-net \
  -e POSTGRES_DB=bulgaria \
  -e POSTGRES_PASSWORD=Password1 \
  -p 5433:5432 \
  -v $(pwd)/postgres-data:/var/lib/postgresql/data \
  postgres
```

```
vagrant@docker:~$ docker run -d \
  --name postgres1 \
  --network pg-net \
  -e POSTGRES_DB=bulgaria \
  -e POSTGRES_PASSWORD=Password1 \
  -p 5432:5432 \
  -v $(pwd)/postgres-data:/var/lib/postgresql/data \
  postgres
51059d3c4fd453b20198767ddedd43ab9d690bf052ac891815621634f6aaab83
vagrant@docker:~$ docker exec -i postgres1 psql -U postgres -d bulgaria <<-EOSQL
CREATE TABLE cities (
        id SERIAL PRIMARY KEY,
        city_name VARCHAR(50),
        population INT
);

INSERT INTO cities (city_name, population) VALUES ('София', 1248452);
INSERT INTO cities (city_name, population) VALUES ('Пловдив', 343070);
INSERT INTO cities (city_name, population) VALUES ('Варна', 332686);
INSERT INTO cities (city_name, population) VALUES ('Бургас', 199571);
INSERT INTO cities (city_name, population) VALUES ('Русе', 137533);
INSERT INTO cities (city_name, population) VALUES ('Стара Загора', 124599);
INSERT INTO cities (city_name, population) VALUES ('Плевен', 93214);
INSERT INTO cities (city_name, population) VALUES ('Сливен', 83740);
INSERT INTO cities (city_name, population) VALUES ('Добрич', 79269);
INSERT INTO cities (city_name, population) VALUES ('Шумен', 72342);
EOSQL
CREATE TABLE
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
```

```
vagrant@docker:~$ docker run -d \
  --name postgres2 \
  --network pg-net \
  -e POSTGRES_DB=bulgaria \
  -e POSTGRES_PASSWORD=Password1 \
  -p 5433:5432 \
  -v $(pwd)/postgres-data:/var/lib/postgresql/data \
  postgres
72cfde2871f8ceb873e4e0f399ea62d220c7c9f886f69773067fc466be665505
```

```
vagrant@docker:~$ docker exec -it postgres2 psql \
  -U postgres \
  -d bulgaria \
  -c "SELECT * FROM cities;"
 id |  city_name   | population
----+--------------+------------
  1 | София        |    1248452
  2 | Пловдив      |     343070
  3 | Варна        |     332686
  4 | Бургас       |     199571
  5 | Русе         |     137533
  6 | Стара Загора |     124599
  7 | Плевен       |      93214
  8 | Сливен       |      83740
  9 | Добрич       |      79269
 10 | Шумен        |      72342
(10 rows)

vagrant@docker:~$ docker inspect postgres1 \
--format '{{ range .Mounts }}{{ .Source }} -> {{ .Destination }}{{ "\n" }}{{ end }}'
/home/vagrant/postgres-data -> /var/lib/postgresql/data

vagrant@docker:~$ docker inspect postgres2 \
--format '{{ range .Mounts }}{{ .Source }} -> {{ .Destination }}{{ "\n" }}{{ end }}'
/home/vagrant/postgres-data -> /var/lib/postgresql/data
```

Dedicated Volume

```
docker volume create postgres-data
```

```
vagrant@docker:~$ docker volume ls
DRIVER    VOLUME NAME
vagrant@docker:~$ docker volume create postgres-data
postgres-data
vagrant@docker:~$ docker volume ls --format "{{.Name}}: {{.Driver}}: {{.Mountpoint}}"
postgres-data: local: /var/lib/docker/volumes/postgres-data/_data
vagrant@docker:~$ docker volume ls
DRIVER    VOLUME NAME
local     postgres-data
vagrant@docker:~$
```

As before we will start the first PostgreSQL instance populate the data and then start the second instance.

```
docker run -d \
  --name postgres1 \
  --network pg-net \
  -e POSTGRES_DB=bulgaria \
  -e POSTGRES_PASSWORD=Password1 \
  -p 5432:5432 \
  -v postgres-data:/var/lib/postgresql/data \
  postgres
```

Population of data omitted for brevity.

```
docker run -d \
  --name postgres2 \
  --network pg-net \
  -e POSTGRES_DB=bulgaria \
  -e POSTGRES_PASSWORD=Password1 \
  -p 5433:5432 \
  -v postgres-data:/var/lib/postgresql/data \
  postgres
```

```
vagrant@docker:~$ docker container ls -a
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS          PORTS                                       NAMES
58108bd9cf87   postgres   "docker-entrypoint.s…"   5 seconds ago    Up 5 seconds    0.0.0.0:5433->5432/tcp, [::]:5433->5432/tcp   postgres2
58c866bfefd6   postgres   "docker-entrypoint.s…"   43 seconds ago   Up 43 seconds   0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp   postgres1
```

```
vagrant@docker:~$ docker run -it --rm --network pg-net postgres psql \
  -h postgres1 \
  -U postgres \
  -d bulgaria \
  -c "SELECT * FROM cities;"
Password for user postgres:
 id |  city_name   | population
----+--------------+------------
  1 | София        |    1248452
  2 | Пловдив      |     343070
  3 | Варна        |     332686
  4 | Бургас       |     199571
  5 | Русе         |     137533
  6 | Стара Загора |     124599
  7 | Плевен       |      93214
  8 | Сливен       |      83740
  9 | Добрич       |      79269
 10 | Шумен        |      72342
(10 rows)

vagrant@docker:~$ docker run -it --rm --network pg-net postgres psql \
  -h postgres2 \
  -U postgres \
  -d bulgaria \
  -c "SELECT * FROM cities;"
Password for user postgres:
 id |  city_name   | population
----+--------------+------------
  1 | София        |    1248452
  2 | Пловдив      |     343070
  3 | Варна        |     332686
  4 | Бургас       |     199571
  5 | Русе         |     137533
  6 | Стара Загора |     124599
  7 | Плевен       |      93214
  8 | Сливен       |      83740
  9 | Добрич       |      79269
 10 | Шумен        |      72342
(10 rows)

vagrant@docker:~$ docker volume ls
DRIVER    VOLUME NAME
local     postgres-data
```

Volume Containers

To be useful the database volume we will start the PostgreSQL server, populate it with data and then stop it and use its volume to start other instances.

```
docker run -d \
  --name bulgaria-db-base \
  --network pg-net \
  -e POSTGRES_DB=bulgaria \
  -e POSTGRES_PASSWORD=Password1 \
  -p 5431:5432 \
  -v postgres-data:/var/lib/postgresql/data \
  postgres
```

```
docker container stop bulgaria-db-base
```

Population of data omitted for brevity.

```
docker run -d \
  --name postgres1 \
  --network pg-net \
  -e POSTGRES_DB=bulgaria \
  -e POSTGRES_PASSWORD=Password1 \
  -p 5432:5432 \
  --volumes-from bulgaria-db-base \
  postgres
```

Custom Container Images

Create image from Dockerfile and add content

For this step we will use the files in the *bulgaria-db* folder. Inside there is a Dockerfile which will extend postgres docker image and using custom population script to populate the database with data on run.

```
vagrant@docker:~$ ls -hl /vagrant/bulgaria-db/
total 4.5K
-rwxrwxrwx 1 vagrant vagrant 1.8K Jun 20 14:43 create-database.sh
-rwxrwxrwx 1 vagrant vagrant  134 Jun 22 00:47 Dockerfile
vagrant@docker:~$
```

```
vagrant@docker:~$ docker build -f /vagrant/bulgaria-db/Dockerfile \
  --build-arg BASE_IMAGE_TAG=17-alpine \
  -t bulgaria-db \
  /vagrant/bulgaria-db
[+] Building 14.9s (7/7) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 173B
 => [internal] load metadata for docker.io/library/postgres:17-alpine
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
 => => transferring context: 1.84kB
 => [1/2] FROM docker.io/library/postgres:17-alpine@sha256:fbe21607052bb5c298674f2fd8cf044a63aa3ddf50b81627f894f91f40f50bcb
 => => resolve docker.io/library/postgres:17-alpine@sha256:fbe21607052bb5c298674f2fd8cf044a63aa3ddf50b81627f894f91f40f50bcb
 => => sha256:fbe21607052bb5c298674f2fd8cf044a63aa3ddf50b81627f894f91f40f50bcb 10.29kB / 10.29kB
 => => sha256:f40315d0e8a6b27dcd87b69bc98f8012e8bd361f7587b5050cd9599e052cbe77 8.49kB / 8.49kB
 => => sha256:fa4fc0c3be6ffcf1c9af30ce05889ae823cf88de5134ce1bc7050303995ca7dd 1.12MB / 1.12MB
 => => sha256:1bc31b94cb037509ba25aed30f6f2903256ec3d122fcc2ae465ab988d09d3c80 2.87kB / 2.87kB
 => => sha256:fe07684b16b82247c3539ed86a65ff37a76138ec25d380bd80c869a1a4c73236 3.80MB / 3.80MB
 => => sha256:8fdb66080d86d3ba868842a9d311a52aa05db8ab6b365ec7e3142d4efdb5c392 971B / 971B
 => => sha256:0796d800157f2997aba819bbf29bab014a2f30c3c54e688ed89081b573cf6976 114B / 114B
 => => extracting sha256:fe07684b16b82247c3539ed86a65ff37a76138ec25d380bd80c869a1a4c73236
 => => sha256:beb76af926b788b465efe67ff6ba39e1dd989df57b2b9a8643e7d674c0b9db2d 105.85MB / 105.85MB
 => => sha256:ec2b26b9d4c9c374b4a1924edc41e7e920e6edb8a9c2e8fc5332b74fc0cc7d2c 9.88kB / 9.88kB
 => => sha256:e0dc4151d8ff252b1ed082c377991e8f31383dfa14c28d8646c1af8960120e4a 129B / 129B
 => => extracting sha256:8fdb66080d86d3ba868842a9d311a52aa05db8ab6b365ec7e3142d4efdb5c392
 => => sha256:490b710f5dec79d3af2501c19d899cf58fe84c2efa7959640067e07659be0f5c 168B / 168B
 => => extracting sha256:fa4fc0c3be6ffcf1c9af30ce05889ae823cf88de5134ce1bc7050303995ca7dd
 => => sha256:72bd8efbdd17658c636e5faacda2bdbc703c20011676c01012024fd5faf54efd 5.93kB / 5.93kB
 => => extracting sha256:0796d800157f2997aba819bbf29bab014a2f30c3c54e688ed89081b573cf6976
 => => sha256:b6579d264f5b4020ea3f3cab4eae9f3221ee29280fe41048d5cd28c6c03473d9 187B / 187B
 => => extracting sha256:beb76af926b788b465efe67ff6ba39e1dd989df57b2b9a8643e7d674c0b9db2d
 => => extracting sha256:ec2b26b9d4c9c374b4a1924edc41e7e920e6edb8a9c2e8fc5332b74fc0cc7d2c
 => => extracting sha256:e0dc4151d8ff252b1ed082c377991e8f31383dfa14c28d8646c1af8960120e4a
 => => extracting sha256:490b710f5dec79d3af2501c19d899cf58fe84c2efa7959640067e07659be0f5c
 => => extracting sha256:72bd8efbdd17658c636e5faacda2bdbc703c20011676c01012024fd5faf54efd
 => => extracting sha256:b6579d264f5b4020ea3f3cab4eae9f3221ee29280fe41048d5cd28c6c03473d9
 => [2/2] COPY create-database.sh /docker-entrypoint-initdb.d/create-database.sh
 => exporting to image
 => => exporting layers
 => => writing image sha256:dedc4c9ae35efbc8675a8ad8633f995507539b8612d22aa0a5aa778b502a61a2
 => => naming to docker.io/library/bulgaria-db
vagrant@docker:~$
```

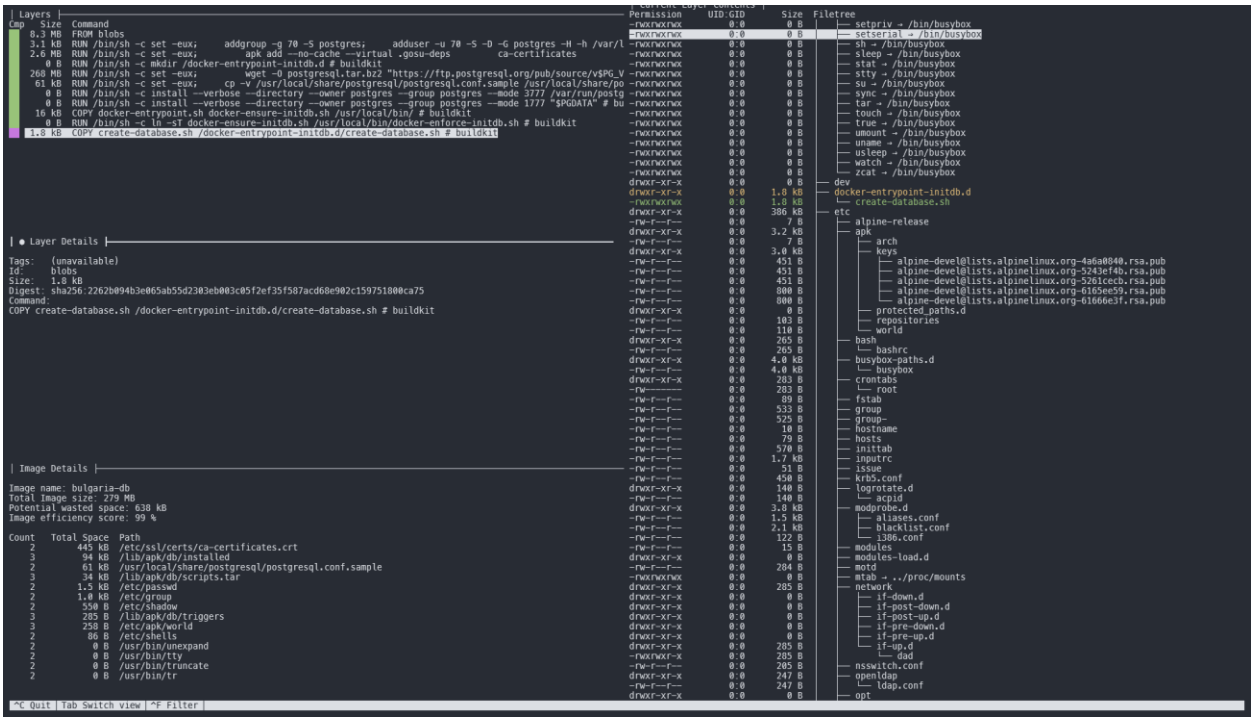
```
vagrant@docker:~$ docker run -d \
  --name bulgaria-db \
  --network pg-net \
  -e POSTGRES_DB=bulgaria \
  -e POSTGRES_PASSWORD=Password1 \
  -p 5432:5432 \
  bulgaria-db
6e78b4341664db1340441200a3b99273d0e4489c0fa88f0054e9330294fe17c9
vagrant@docker:~$ docker exec -it bulgaria-db psql \
  -U postgres \
  -d bulgaria \
  -c "SELECT * FROM cities;"
 id |  city_name   | population
----+--------------+------------
  1 | София        |    1248452
  2 | Пловдив      |     343070
  3 | Варна        |     332686
  4 | Бургас       |     199571
  5 | Русе         |     137533
  6 | Стара Загора |     124599
  7 | Плевен       |      93214
  8 | Сливен       |      83740
  9 | Добрич       |      79269
 10 | Шумен        |      72342
(10 rows)
```

Inspecting the image with Dive

Repo: https://github.com/wagoodman/dive

```
dive bulgaria-db
```

We can see layers information, how the content of the image changes, details about the layer etc.



Publishing an image

```
docker login
```

```
vagrant@docker:~$ docker login

USING WEB-BASED LOGIN

i Info → To sign in with credentials on the command line, use 'docker login -u <username>'


Your one-time device confirmation code is: LPGT-PSHJ
Press ENTER to open your browser or submit your device code here: https://login.docker.com/activate

Waiting for authentication in the browser…

WARNING! Your credentials are stored unencrypted in '/home/vagrant/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

Login Succeeded
vagrant@docker:~$
```

```
docker image tag bulgaria-db vasatanasov/bulgaria-db:demo
```

```
docker image push vasatanasov/bulgaria-db:demo
```

```
vagrant@docker:~$ docker image tag bulgaria-db vasatanasov/bulgaria-db:demo
vagrant@docker:~$ docker image ls
REPOSITORY                TAG       IMAGE ID       CREATED         SIZE
bulgaria-db               latest    cc54fb823584   5 minutes ago   279MB
vasatanasov/bulgaria-db   demo      cc54fb823584   5 minutes ago   279MB
vagrant@docker:~$ docker image push vasatanasov/bulgaria-db:demo
The push refers to repository [docker.io/vasatanasov/bulgaria-db]
2262b094b3e0: Pushed
1664c2c5d5ac: Mounted from library/postgres
383a8ae6e590: Mounted from library/postgres
dbd66ba91b6d: Mounted from library/postgres
0b5565b41414: Mounted from library/postgres
240002faf7ad: Mounted from library/postgres
b2276f7c8ac2: Mounted from library/postgres
0ac3b956fb95: Mounted from library/postgres
ae94c5b0fc56: Mounted from library/postgres
a670c37a1951: Mounted from library/postgres
fd2758d7a50e: Layer already exists
demo: digest: sha256:8d383546e668535c5cae0c33f1e1e2acc7f839c4d3ba542dfb33a2eb14b82368 size: 2610
vagrant@docker:~$
```

```
vagrant@docker:~$ docker run -d \
  --name bulgaria-db \
  -e POSTGRES_DB=bulgaria \
  -e POSTGRES_PASSWORD=Password1 \
  -p 5432:5432 \
  vasatanasov/bulgaria-db:demo
Unable to find image 'vasatanasov/bulgaria-db:demo' locally
demo: Pulling from vasatanasov/bulgaria-db
fe07684b16b8: Already exists
8fdb66080d86: Already exists
fa4fc0c3be6f: Already exists
0796d800157f: Already exists
beb76af926b7: Already exists
ec2b26b9d4c9: Already exists
e0dc4151d8ff: Already exists
490b710f5dec: Already exists
72bd8efbdd17: Already exists
b6579d264f5b: Already exists
34a411f453f4: Already exists
Digest: sha256:8d383546e668535c5cae0c33f1e1e2acc7f839c4d3ba542dfb33a2eb14b82368
Status: Downloaded newer image for vasatanasov/bulgaria-db:demo
1f75f1866db32037d44d03acd061cac9a702970d840f34e0b83280ec04c9853d
vagrant@docker:~$ docker exec -it bulgaria-db psql \
  -U postgres \
  -d bulgaria \
  -c "SELECT * FROM cities;"
 id |  city_name   | population
----+--------------+------------
  1 | София        |    1248452
  2 | Пловдив      |     343070
  3 | Варна        |     332686
  4 | Бургас       |     199571
  5 | Русе         |     137533
  6 | Стара Загора |     124599
  7 | Плевен       |      93214
  8 | Сливен       |      83740
  9 | Добрич       |      79269
 10 | Шумен        |      72342
(10 rows)
```

ENTRYPOINT and CMD

For the demonstration of how ENTRYPOINT and CMD work together I will create a custom Dockerfile with which will prepare an image that will act as executable. The application will query the remote server which will be running our bulgari-db and search for a city by its name and return results on the command line. We will assume that the remote database is up and running.

```
vagrant@docker:~$ docker container ls -a
CONTAINER ID   IMAGE                          COMMAND                  CREATED         STATUS         PORTS                                                  NAMES
1f75f1866db3   vasatanasov/bulgaria-db:demo   "docker-entrypoint.s…"   8 minutes ago   Up 8 minutes   0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp   bulgaria-db
vagrant@docker:~$
```

Files for the custom docker image are in the *city-search* folder.

```
vagrant@docker:~$ ls -hl /vagrant/city-search/
total 2.0K
-rwxrwxrwx 1 vagrant vagrant  97 Jun 22 01:43 city-search.sh
-rwxrwxrwx 1 vagrant vagrant 407 Jun 22 08:41 Dockerfile
-rwxrwxrwx 1 vagrant vagrant 532 Jun 22 01:37 run-query.sh
vagrant@docker:~$
```

We will assume that pg-net exist and the bulgaria-db container is connected to it.

```
docker build -f /vagrant/city-search/Dockerfile \
  -t city-search \
  /vagrant/city-search
```

```
vagrant@docker:~$ docker build -f /vagrant/city-search/Dockerfile \
  -t city-search \
  /vagrant/city-search
[+] Building 11.4s (6/9)
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 446B
 => [internal] load metadata for docker.io/library/debian:bookworm-slim
 => [auth] library/debian:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/4] FROM docker.io/library/debian:bookworm-slim@sha256:e5865e6858dacc255bead044a7f2d0ad8c362433cfaa5acefb670c1edf54dfef
 => => resolve docker.io/library/debian:bookworm-slim@sha256:e5865e6858dacc255bead044a7f2d0ad8c362433cfaa5acefb670c1edf54dfef
 => => sha256:6ba4ddadd113af5ae1a6014da46e3074d5adcdf58a973c8b653f54b1374778cb 453B / 453B
 => => sha256:dad67da3f26bce15939543965e09c4059533b025f707aad72ed3d3f3a09c66f8 28.23MB / 28.23MB
 => => sha256:e5865e6858dacc255bead044a7f2d0ad8c362433cfaa5acefb670c1edf54dfef 8.56kB / 8.56kB
 => => sha256:f957d7d19c4171aded89330899c1501c03707d879f200001aa47af87d98b3c70 1.02kB / 1.02kB
 => => extracting sha256:dad67da3f26bce15939543965e09c4059533b025f707aad72ed3d3f3a09c66f8
 => [internal] load build context
 => => transferring context: 577B
 => [2/4] RUN apt-get update &&     apt-get install -y --no-install-recommends     postgresql-client     && apt-get clean     && rm -rf /var/lib/apt/lists/*
 => => # Preparing to unpack .../15-libsasl2-2_2.1.28+dfsg-10_amd64.deb ...
 => => # Unpacking libsasl2-2:amd64 (2.1.28+dfsg-10) ...
 => => # Selecting previously unselected package libldap-2.5-0:amd64.
 => => # Preparing to unpack .../16-libldap-2.5-0_2.5.13+dfsg-5_amd64.deb ...
 => => # Unpacking libldap-2.5-0:amd64 (2.5.13+dfsg-5) ...
 => => # Selecting previously unselected package libpq5:amd64.
```

We can search for the exact name.

```
docker run -it --rm --network pg-net city-search "Шумен"
```

A match.

```
docker run -it --rm --network pg-net city-search "С"
```

When no argument is passed all results are returned.

```
docker run -it --rm --network pg-net city-search
```

```
vagrant@docker:~$ docker run -it --rm --network pg-net city-search "Шумен"
[
  {
    "id": 10,
    "city_name": "Шумен",
    "population": 72342
  }
]
vagrant@docker:~$ docker run -it --rm --network pg-net city-search "С"
[
  {
    "id": 1,
    "city_name": "София",
    "population": 1248452
  },
  {
    "id": 4,
    "city_name": "Бургас",
    "population": 199571
  },
  {
    "id": 5,
    "city_name": "Русе",
    "population": 137533
  },
  {
    "id": 6,
    "city_name": "Стара Загора",
    "population": 124599
  },
  {
    "id": 8,
    "city_name": "Сливен",
    "population": 83740
  }
]
vagrant@docker:~$
```

To use it as executable we can create e bash script (*city-search.sh*) and place it in */usr/local/bin* so we can have it as a bash command.

```
#!/bin/bash

docker run -it --rm --network pg-net city-search "$1"
```

```
vagrant@docker:~$ sudo cp /vagrant/city-search/city-search.sh /usr/local/bin/city-search
vagrant@docker:~$ city-search "Ш"
[
  {
    "id": 10,
    "city_name": "Шумен",
    "population": 72342
  }
]
vagrant@docker:~$
```

Create image from container

We will run our city database and add new city to it.

```
docker run -d \
  --name bulgaria-db \
  --network pg-net \
  -e POSTGRES_DB=bulgaria \
  -e POSTGRES_PASSWORD=Password1 \
  -p 5432:5432 \
  vasatanasov/bulgaria-db:demo
```

```
docker exec -it bulgaria-db bash
```

Insert new city in the create-database.sh script and exit Ctrl-PQ

```
vi docker-entrypoint-initdb.d/create-database.sh
```

```
INSERT INTO cities (city_name, population) VALUES ('Благоевград', 172446);
```

Commit the container with the updated database.

```
docker container commit --author "SoftUni Student Vasil Atanasov" \
  bulgaria-db \
  bulgaria-db-v2
```

```
docker image ls bulgaria-db-v2
```

Run the container.

```
docker run -d \
  --name bulgaria-db-v2 \
  --network pg-net \
  -e POSTGRES_DB=bulgaria \
  -e POSTGRES_PASSWORD=Password1 \
  -p 5433:5432 \
  bulgaria-db-v2=
```

```
vagrant@docker:~$ docker container commit --author "SoftUni Student Vasil Atanasov" \
  bulgaria-db \
  bulgaria-db-v2
sha256:ebce9dda2b048432a01ba18e55a7156a327fde59838f2c083e1dcbcda1e28028
vagrant@docker:~$ docker image ls bulgaria-db-v2
REPOSITORY       TAG       IMAGE ID       CREATED         SIZE
bulgaria-db-v2   latest    ebce9dda2b04   2 seconds ago   279MB
vagrant@docker:~$
```

```
vagrant@docker:~$ docker exec -i bulgaria-db-v2 psql \
  -U postgres \
  -d bulgaria \
  -c "SELECT * FROM cities WHERE city_name = 'Благоевград';"
 id |  city_name  | population
----+-------------+------------
 11 | Благоевград |     172446
(1 row)

vagrant@docker:~$
```

Create image with heredoc

We will create e docker image from postgres which will have an empty cities table.

```
docker image build -t bulgaria-db - << 'EOF'
ARG BASE_IMAGE_TAG=latest
FROM postgres:${BASE_IMAGE_TAG}

RUN printf '#!/bin/bash\npsql -U postgres <<EOFSQL\nCREATE TABLE IF NOT EXISTS cities (\n  id SERIAL PRIMARY KEY,\n  city_name TEXT,\n  population INTEGER\n);\nEOFSQL\n' \
  > /docker-entrypoint-initdb.d/create-database.sh && chmod +x /docker-entrypoint-initdb.d/create-database.sh
EOF
```

```
docker run -d --name bulgaria-db -e POSTGRES_PASSWORD=Password1 bulgaria-db
```

```
docker exec -it bulgaria-db psql -U postgres -d postgres -c 'SELECT * FROM cities;'
```

```
vagrant@docker:~$ docker image build -t bulgaria-db - << 'EOF'
ARG BASE_IMAGE_TAG=latest
FROM postgres:${BASE_IMAGE_TAG}

RUN printf '#!/bin/bash\npsql -U postgres <<EOFSQL\nCREATE TABLE IF NOT EXISTS cities (\n  id SERIAL PRIMARY KEY,\n  city_name TEXT,\n  population INTEGER\n);\nEOFSQL\n' \
  > /docker-entrypoint-initdb.d/create-database.sh && chmod +x /docker-entrypoint-initdb.d/create-database.sh
EOF
[+] Building 1.1s (6/6) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 380B
 => [internal] load metadata for docker.io/library/postgres:latest
 => [internal] load .dockerignore
 => => transferring context: 2B
 => CACHED [1/2] FROM docker.io/library/postgres:latest@sha256:6cf6142afacfa89fb28b894d6391c7dcbf6523c33178bdc33e782b3b533a9342
 => [2/2] RUN printf '#!/bin/bash\npsql -U postgres <<EOFSQL\nCREATE TABLE IF NOT EXISTS cities (\n  id SERIAL PRIMARY KEY,\n  city_name TEXT,\n  population INTEGER\n);\nEOFS
 => exporting to image
 => => exporting layers
 => => writing image sha256:f0f2ae608b8c0a5f8290671dc513e2741eb04a198d371d513bef0ca96dc97d06
 => => naming to docker.io/library/bulgaria-db
vagrant@docker:~$ docker run -d --name bulgaria-db -e POSTGRES_PASSWORD=Password1 bulgaria-db
0958e370dce833267f4d86797a0289c98f9619b5e4a607eb92a11fef264fac87
vagrant@docker:~$ docker exec -it bulgaria-db psql -U postgres -d postgres -c 'SELECT * FROM cities;'
 id | city_name | population
----+-----------+------------
(0 rows)
```

Archive and transfer images

```
vagrant@docker:~$ docker image save -o bulgaria-db vasatanasov/bulgaria-db:demo
vagrant@docker:~$ ls
bulgaria-db
vagrant@docker:~$ docker image rm vasatanasov/bulgaria-db:demo
Untagged: vasatanasov/bulgaria-db:demo
Untagged: vasatanasov/bulgaria-db@sha256:8d383546e668535c5cae0c33f1e1e2acc7f839c4d3ba542dfb33a2eb14b82368
Deleted: sha256:cc54fb823584a49bea1bb6761e7d7726973b5eea92c4f59f092802ff3ae370a8
vagrant@docker:~$ docker image ls
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
vagrant@docker:~$ docker image load -i bulgaria-db
Loaded image: vasatanasov/bulgaria-db:demo
vagrant@docker:~$ docker image ls
REPOSITORY                TAG       IMAGE ID       CREATED       SIZE
vasatanasov/bulgaria-db   demo      cc54fb823584   3 hours ago   279MB
vagrant@docker:~$
```

Archive and transfer containers

```
docker container export -o bulgaria-db.tar vasatanasov/bulgaria-db:demo
```

Using docker image import, creates a new image from a tarball (usually from docker export) and does not preserve metadata such as the user (USER postgres), environment variables, or entrypoint from the original image. That is why we need specify all on the command line.

This example is not complete because the postgres Dockerfile have more envs.

```
docker image import \
  bulgaria-db.tar \
  --change 'CMD ["postgres"]' \
  --change 'USER postgres' \
  new-bulgaria-db
```

Custom Apache Web Server Docker Image

The files of the task are in the *apache* folder.

```
vagrant@docker:~$ ls -hl /vagrant/apache/
total 1.0K
-rwxrwxrwx 1 vagrant vagrant 231 Jun 22 00:34 Dockerfile
-rwxrwxrwx 1 vagrant vagrant 335 Jun 22 00:40 index.html
vagrant@docker:~$
```

```
FROM almalinux:9

RUN dnf -y update && \
    dnf -y install httpd && \
    dnf clean all

COPY index.html /var/www/html/index.html

EXPOSE 80

CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]
```

```
vagrant@docker:~$ docker run -d --name web -p 8080:80 web
651611390c02c0bd33ce942fbfca96c9e79f3d53a110943c82043c8eb327d683
vagrant@docker:~$ curl http://localhost:8080
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Advanced Containerization Concepts</title>
</head>
<body>
<h1>Hello from my first container!</h1>
</body>
</html>vagrant@docker:~$
```

Best Practices and Troubleshooting

We will base the following steps on the custom Apache Web Server docker image.

Files are located under the *best-practice-and-troubleshooting* folder.

Best Practices

Provide Details via Labels

```
FROM almalinux:9

LABEL version="1.0" \
      description="A sample web application that displays It Works"

RUN dnf -y update && \
    dnf -y install httpd && \
    dnf clean all

COPY index.html /var/www/html/index.html

EXPOSE 80

CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]
```

```
docker build \
-f /vagrant/best-practice-and-troubleshooting/label.Dockerfile -t web \
/vagrant/best-practice-and-troubleshooting
```

```
docker image inspect --format='{{json .Config.Labels}}' labels
```

```
vagrant@docker:~$ docker build \
-f /vagrant/best-practice-and-troubleshooting/label.Dockerfile -t web \
/vagrant/best-practice-and-troubleshooting
[+] Building 0.7s (8/8) FINISHED
 => [internal] load build definition from label.Dockerfile
 => => transferring dockerfile: 335B
 => [internal] load metadata for docker.io/library/almalinux:9
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/3] FROM docker.io/library/almalinux:9@sha256:d5e4140306a7d52d7dfaa851b040698fa29cce9335bd2b37dd663ebe8a1f1775
 => [internal] load build context
 => => transferring context: 32B
 => CACHED [2/3] RUN dnf -y update &&     dnf -y install httpd &&     dnf clean all
 => CACHED [3/3] COPY index.html /var/www/html/index.html
 => exporting to image
 => => exporting layers
 => => writing image sha256:2457391898052501d71a47a4851b104243f68a0ea8c44049d0e01aee897f709c
 => => naming to docker.io/library/web
vagrant@docker:~$ docker image inspect --format='{{json .Config.Labels}}' labels
{"description":"A sample web application that displays It Works","version":"1.0"}
vagrant@docker:~$
```

Use the Right Base Image

```
FROM almalinux:9.6-minimal

RUN microdnf -y update && \
    microdnf -y install httpd && \
    microdnf clean all

COPY index.html /var/www/html/index.html

EXPOSE 80

CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]
```

```
docker build \
-f /vagrant/best-practice-and-troubleshooting/minimal.Dockerfile \
-t web-minimal \
/vagrant/best-practice-and-troubleshooting
```

```
vagrant@docker:~$ docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED              SIZE
web-minimal   latest    838de1d59b5e   About a minute ago   149MB
web           latest    871d8ff13dae   2 minutes ago        215MB
vagrant@docker:~$
```

We can easily see the difference in size between the two images.

Push Readme with the Image

Installing pushrm

```
install_pushrm() {
  log "Installing Docker Push Readme"
  local version
  version=$(curl -sL "https://api.github.com/repos/christian-korneck/docker-pushrm/releases/latest"
| grep '"tag_name":' | sed -E 's/.*"v([^"]+)".*/\1/')
  curl -fLo docker-pushrm "https://github.com/christian-korneck/docker-
pushrm/releases/download/v${version}/docker-pushrm_linux_amd64"
  chmod +x docker-pushrm
  mkdir -p "/usr/local/lib/docker/cli-plugins"
  mv docker-pushrm "/usr/local/lib/docker/cli-plugins/"
}
```

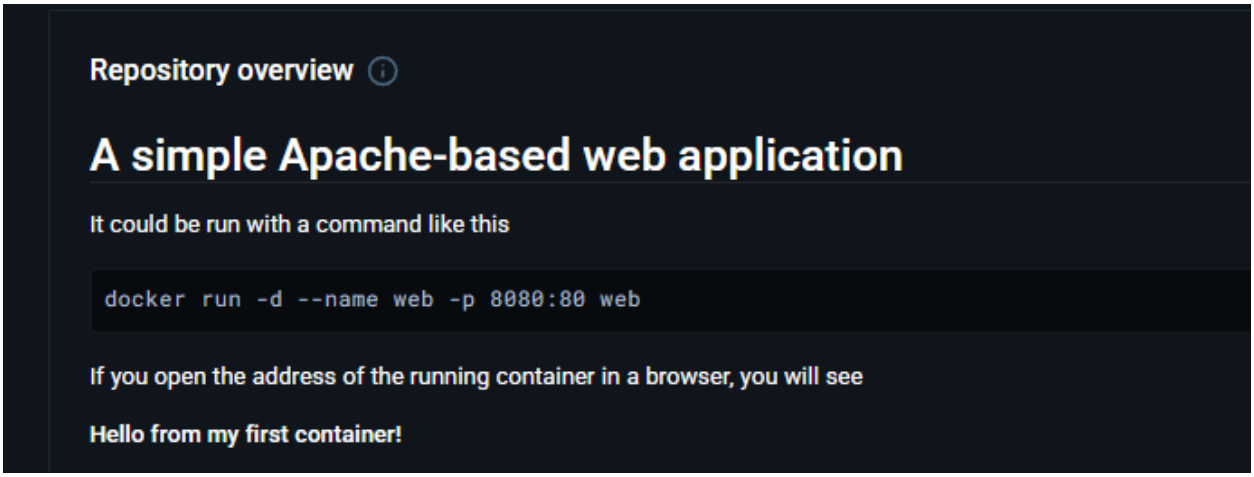
```
  plugin      Manage plugins
  pushrm*     Push Readme to container registry
  system      Manage Docker
  trust       Manage trust on Docker images
```

```
docker image tag web-minimal:latest vasatanasov/web-minimal:demo
```

```
docker image push vasatanasov/web-minimal:demo
```

```
docker pushrm \
  --file /vagrant/best-practice-and-troubleshooting/README.md \
  vasatanasov/web-minimal:demo
```

https://hub.docker.com/repository/docker/vasatanasov/web-minimal/general



Scan Images

Installing **Trivy**

```
install_trivy() {
  log "Installing Trivy"
  curl -sfL https://raw.githubusercontent.com/aquasecurity/trivy/main/contrib/install.sh | sh -s --
-b "$BIN_DIR" latest
}
```

```
vagrant@docker:~$ trivy --version
Version: 0.63.0
vagrant@docker:~$
```

```
trivy image vasatanasov/web-minimal:demo
```

Use a Linter

Installing **Hadolint**

```
install_hadolint() {
  log "Installing Hadolint"
  local version
  version=$(curl -sL "https://api.github.com/repos/hadolint/hadolint/releases/latest" | grep
'"tag_name":' | sed -E 's/.*"v([^"]+)".*/\1/')
  curl -fLo hadolint "https://github.com/hadolint/hadolint/releases/download/v${version}/hadolint-
Linux-x86_64"
  chmod +x hadolint   mv hadolint "$BIN_DIR/"
}
```

```
vagrant@docker:~$ hadolint --version
Haskell Dockerfile Linter 2.12.0
vagrant@docker:~$
```

```
hadolint /vagrant/best-practice-and-troubleshooting/minimal.Dockerfile
```

```
vagrant@docker:~$ hadolint /vagrant/best-practice-and-troubleshooting/minimal.Dockerfile
/vagrant/best-practice-and-troubleshooting/minimal.Dockerfile:3 DL3041 warning: Specify version with `dnf install -y <package>-<version>`.
vagrant@docker:~$
```

The fix:

```
FROM almalinux:9.6-minimal

RUN microdnf -y update && \
    microdnf -y install httpd-2.4.62-4.el9 && \
    microdnf clean all

COPY index.html /var/www/html/index.html

EXPOSE 80

CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]
```

```
hadolint /vagrant/best-practice-and-troubleshooting/linter-fix.Dockerfile
```

This outputs no errors.

Multi-stage Build

This example of Dockerfile builds a custom Keycloak image using a multi-stage approach. It installs extra tools like jq and curl from a minimal UBI 9 image, builds Keycloak with custom providers and config in a builder stage, and combines everything into a clean, secure final image. This setup keeps the final image lightweight and production ready. This Dockerfile I personally created and use in development.

```
FROM registry.access.redhat.com/ubi9 AS ubi-micro-build

RUN mkdir -p /mnt/rootfs && \
    dnf install --installroot=/mnt/rootfs jq tzdata curl --releasever=9 \
        --setopt=install_weak_deps=false --nodocs -y && \
    dnf --installroot=/mnt/rootfs clean all

FROM quay.io/keycloak/keycloak:${BASE_IMAGE_TAG} AS builder

ENV KC_DB=postgres

COPY keycloak/providers/ /opt/keycloak/providers/
COPY conf/keycloak.conf /opt/keycloak/conf/

RUN /opt/keycloak/bin/kc.sh build

FROM quay.io/keycloak/keycloak:${BASE_IMAGE_TAG}

COPY --from=builder /opt/keycloak/ /opt/keycloak/
COPY --from=ubi-micro-build /mnt/rootfs/ /

USER root

COPY scripts/ /scripts/
RUN chmod +x /scripts/keycloak/*.sh

USER 1000

WORKDIR /opt/keycloak

ENV KEYCLOAK_HOME=/opt/keycloak
ENV PATH=$PATH:$KEYCLOAK_HOME/bin
```

Troubleshooting

```
docker container logs web-minimal
```