🐳 Lab 2: Docker

Student Name: Kong Samnang

GitHub Repository: https://github.com/SamnangKong426/kong-samnang-lab1/tree/lab2-Docker

---

Part 1: Containerize an Application

Dockerfile

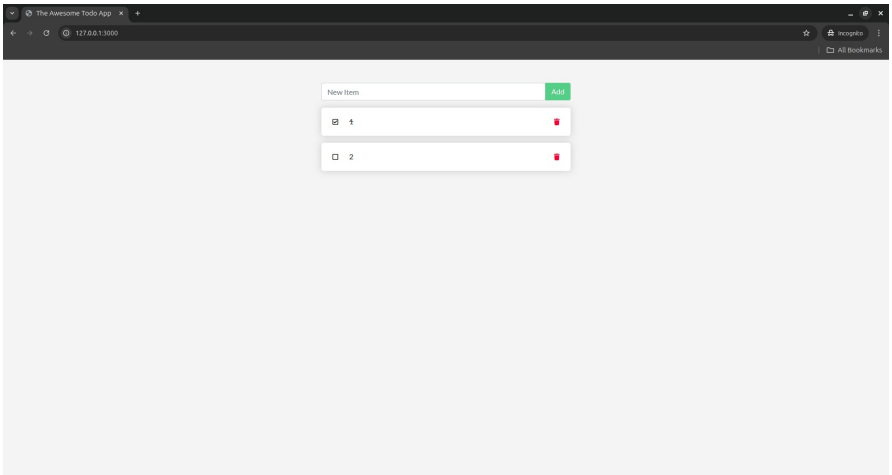
```
# syntax=docker/dockerfile:1

FROM node:lts-alpine
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000
```

Build and Run Commands

```
# Build the image
docker build -t getting-started .

# Start an app container
docker run -d -p 127.0.0.1:3000:3000 getting-started

# List all containers
docker ps
```

```
user@dev: ~/Documents/getting-started-app
user@dev: ~/Documents/getting-started-app 80x24
user@dev:~$ cd Documents/getting-started-app/
user@dev:~/Documents/getting-started-app$ docker build -t getting-started .
[+] Building 1.3s (12/12) FINISHED                              docker:default
 => [internal] load build definition from Dockerfile                      0.0s
 => => transferring dockerfile: 200B                                      0.0s
 => resolve image config for docker-image://docker.io/docker/dockerfile:1  1.1s
 => CACHED docker-image://docker.io/docker/dockerfile:1@sha256:b6afd42430  0.0s
 => [internal] load metadata for docker.io/library/node:lts-alpine        0.0s
 => [internal] load .dockerignore                                         0.0s
 => => transferring context: 64B                                          0.0s
 => [1/5] FROM docker.io/library/node:lts-alpine                          0.0s
 => [internal] load build context                                         0.0s
 => => transferring context: 4.73kB                                       0.0s
 => CACHED [2/5] WORKDIR /app                                             0.0s
 => CACHED [3/5] COPY package.json yarn.lock ./                           0.0s
 => CACHED [4/5] RUN yarn install --production                            0.0s
 => CACHED [5/5] COPY . .                                                 0.0s
 => exporting to image                                                    0.0s
 => => exporting layers                                                   0.0s
 => => writing image sha256:bff67aabb73defca191e234ca87571366506152ea9bdb  0.0s
 => => naming to docker.io/library/getting-started                        0.0s
user@dev:~/Documents/getting-started-app$
```

```
user@dev: ~/Documents/getting-started-app
user@dev: ~/Documents/getting-started-app 80x24
user@dev:~/Documents/getting-started-app$ docker ps
CONTAINER ID   IMAGE             COMMAND                  CREATED             ST
ATUS         PORTS                    NAMES
ec28e57388ca   node:lts-alpine   "docker-entrypoint.s…"   8 minutes ago       Up
 9 seconds   0.0.0.0:3000->3000/tcp   getting-started-app-app-1
8ba1dc15fad8   mysql:8.0         "docker-entrypoint.s…"   About an hour ago   Up
 9 seconds   3306/tcp, 33060/tcp      getting-started-app-mysql-1
user@dev:~/Documents/getting-started-app$
```

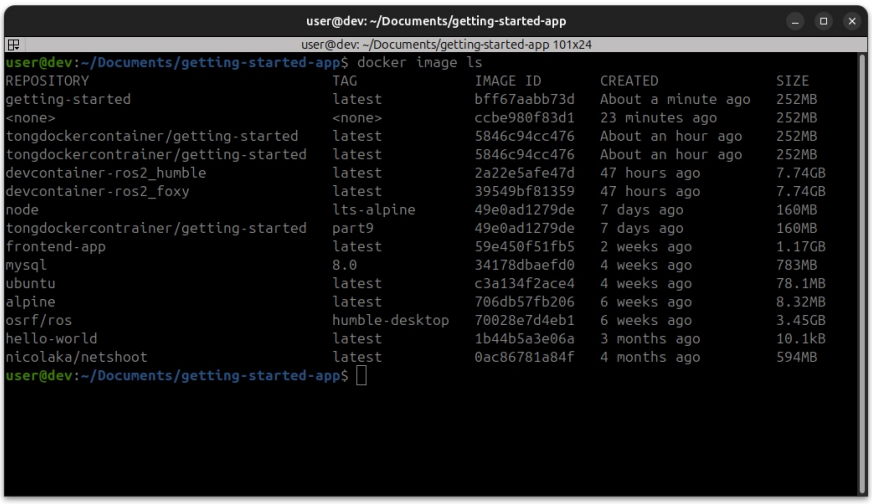


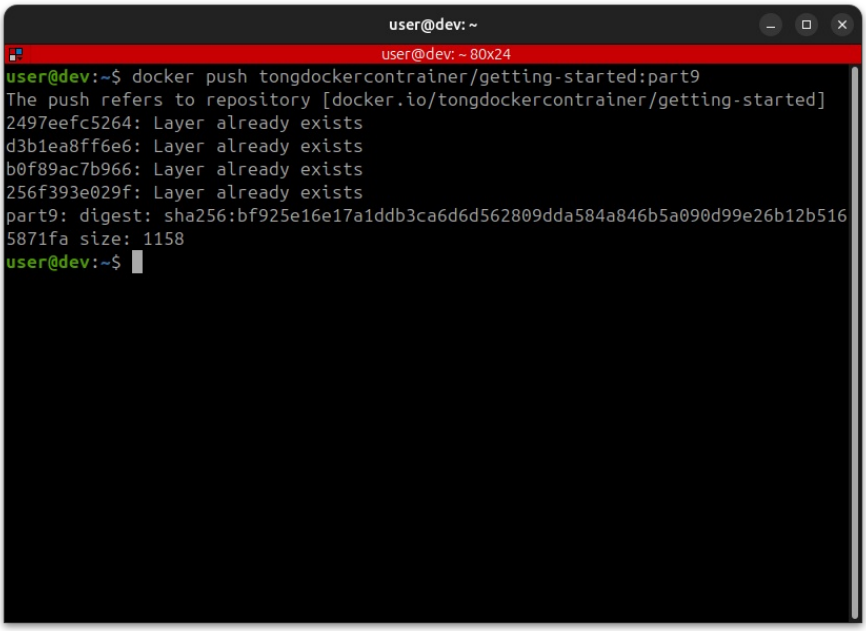
Part 2: Update the Application

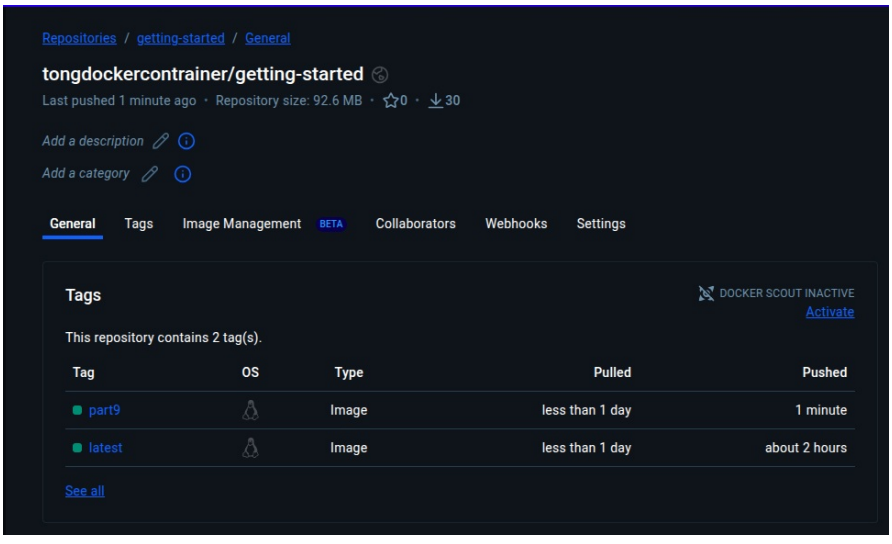
```
        # Build updated version of image
        docker build -t getting-started .

        # Stop container (Replace <the-container-id> with actual ID)
        docker stop <the-container-id>

        # Remove container (Replace <the-container-id> with actual ID)
        docker rm <the-container-id>
```

Part 3: Share the Application

```
        # List all images
        docker image ls

        # Rename docker image using tag command (Replace YOUR-USER-NAME)
        docker tag getting-started YOUR-USER-NAME/getting-started

        # Build image based on specific OS (Replace YOUR-USER-NAME)
        docker build --platform linux/amd64 -t YOUR-USER-NAME/getting-
started .

        # Push the image (Replace YOUR-USER-NAME)
        docker push YOUR-USER-NAME/getting-started
```

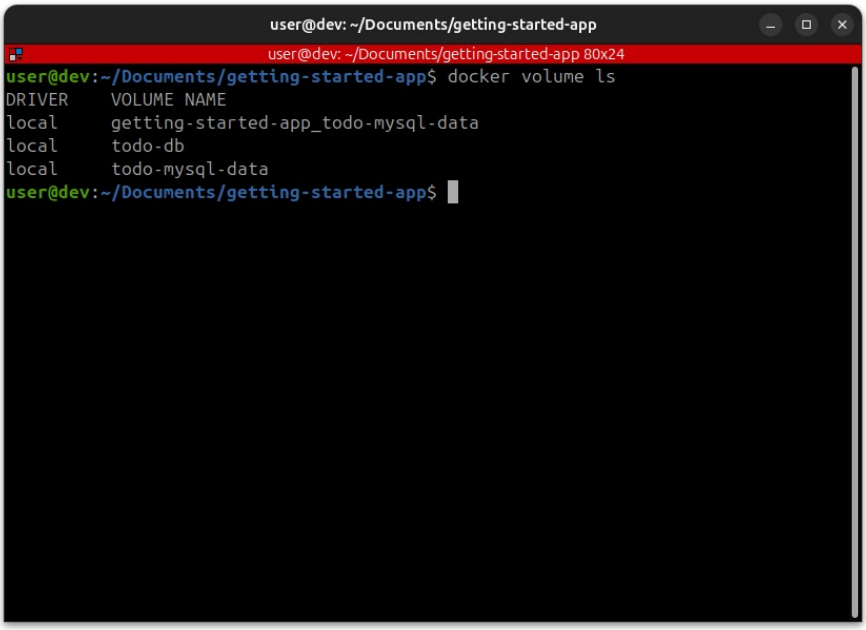

user@dev: ~/Documents/getting-started-app
user@dev: ~/Documents/getting-started-app 101x24
user@dev:~/Documents/getting-started-app$ docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
getting-started latest bff67aabb73d About a minute ago 252MB
<none> <none> ccbe980f83d1 23 minutes ago 252MB
tongdockercontainer/getting-started latest 5846c94cc476 About an hour ago 252MB
tongdockercontrainer/getting-started latest 5846c94cc476 About an hour ago 252MB
devcontainer-ros2_humble latest 2a22e5afe47d 47 hours ago 7.74GB
devcontainer-ros2_foxy latest 39549bf81359 47 hours ago 7.74GB
node lts-alpine 49e0ad1279de 7 days ago 160MB
tongdockercontrainer/getting-started part9 49e0ad1279de 7 days ago 160MB
frontend-app latest 59e450f51fb5 2 weeks ago 1.17GB
mysql 8.0 34178dbaefd0 4 weeks ago 783MB
ubuntu latest c3a134f2ace4 4 weeks ago 78.1MB
alpine latest 706db57fb206 6 weeks ago 8.32MB
osrf/ros humble-desktop 70028e7d4eb1 6 weeks ago 3.45GB
hello-world latest 1b44b5a3e06a 3 months ago 10.1kB
nicolaka/netshoot latest 0ac86781a84f 4 months ago 594MB
user@dev:~/Documents/getting-started-app$


user@dev: ~
user@dev: ~ 80x24
user@dev:~$ docker push tongdockercontrainer/getting-started:part9
The push refers to repository [docker.io/tongdockercontrainer/getting-started]
2497eefc5264: Layer already exists
d3b1ea8ff6e6: Layer already exists
b0f89ac7b966: Layer already exists
256f393e029f: Layer already exists
part9: digest: sha256:bf925e16e17a1ddb3ca6d6d562809dda584a846b5a090d99e26b12b516
5871fa size: 1158
user@dev:~$


Repositories / getting-started / General
tongdockercontrainer/getting-started
Last pushed 1 minute ago · Repository size: 92.6 MB · 0 · 30
Add a description
Add a category
General Tags Image Management BETA Collaborators Webhooks Settings
Tags
DOCKER SCOUT INACTIVE
Activate
This repository contains 2 tag(s).
Tag OS Type Pulled Pushed
part9 Image less than 1 day 1 minute
latest Image less than 1 day about 2 hours
See all

Part 4: Persist the DB (Using Volumes)

```
        # Create a volume
        docker volume create todo-db

        # Run container using a volume mount
        docker run -dp 127.0.0.1:3000:3000 --mount type=volume,src=todo-
db,target=/etc/todos getting-started

        # Inspect a created volume
        docker volume inspect todo-db
```



Part 5: Use Bind Mounts (Local Development)

```
        # Example: Run ubuntu container using bind mount to current working
directory
        docker run -it --mount type=bind,src="$(pwd)",target=/src ubuntu
bash

        # Run development container with bind mount and live reload
        docker run -dp 127.0.0.1:3000:3000 \
            -w /app --mount type=bind,src="$(pwd)",target=/app \
            node:lts-alpine \
            sh -c "yarn install && yarn run dev"

        # Watch the logs (Replace <container-id> with actual ID)
        docker logs -f <container-id>
```

Part 6: Multi-container Apps

```
        # Create a network
```

```
docker network create todo-app

# MySQL container attach to the created network
docker run -d \
    --network todo-app --network-alias mysql \
    -v todo-mysql-data:/var/lib/mysql \
    -e MYSQL_ROOT_PASSWORD=secret \
    -e MYSQL_DATABASE=todos \
    mysql:8.0

# Use exec to use bash terminal (Replace <mysql-container-id> with actual ID)
docker exec -it <mysql-container-id> mysql -u root -p
```

Part 7: Use Docker Compose

compose.yaml

```
services:
  app:
    image: node:lts-alpine
    command: sh -c "yarn install && yarn run dev"
    ports:
      - 127.0.0.1:3000:3000
    working_dir: /app
    volumes:
      - ./:/app
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: root
      MYSQL_PASSWORD: secret
      MYSQL_DB: todos

  mysql:
    image: mysql:8.0
    volumes:
      - todo-mysql-data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: todos

volumes:
  todo-mysql-data:
```

Run Command

```
# Run the application with just one command
docker compose up -d
```

Part 8: Image-building Best Practices

```
# Monitor image history
docker image history getting-started

# Use --no-trunc to get full output
```

```
docker image history --no-trunc getting-started
```

Key Optimization Principle: Optimize the layer caching by copying `package.json` and installing dependencies *before* copying all application code, and utilizing multi-stage builds (`--from=build`) to separate build-time and runtime environments for a smaller final image.

Part 9: What Next

- Learn about Container orchestration (e.g., Kubernetes).
- Explore Cloud Native Computing Foundation (CNCF) projects.
- Create a container from scratch.