🐳Docker & MongoDB - Full Notes (SDET Focused)

1. 🦝Basic Docker Concepts

✨What is Docker?

Docker is a platform to build, run, and manage containers. Containers are lightweight, isolated environments to run applications.

🦠Common Docker Terminology

Term	Meaning
Image	Blueprint of the application (like a class)
Container	Running instance of an image (like an object)
Dockerfile	Set of instructions to build a Docker image
Volume	Mechanism to persist data from a container
Bind Mount	Mount a specific host folder into the container
Docker Hub	Public repo of prebuilt images (like GitHub for containers)

---

2. 🌳Common Docker Commands

```
# Run hello-world container
$ docker run hello-world

# List all containers (running and stopped)
$ docker ps -a

# Remove a container
$ docker rm <container_name>

# Remove all containers
$ docker container prune

# Build Docker image
$ docker build -t image-name .

# Run image as container
$ docker run -it --rm image-name
```

```
# Run container with volume
$ docker run -v /host/path:/container/path image-name
```

---

3. 🦠Dockerfile Example (Playwright Project)

```
FROM mcr.microsoft.com/playwright/python:v1.44.0

WORKDIR /app
COPY . .

RUN pip install --no-cache-dir -r requirements.txt
RUN playwright install --with-deps

CMD ["pytest", "Playwright/jobs/test_play.py", "--html=Playwright/jobs/
report.html"]
```

🦆requirements.txt

```
pytest
requests
pytest-html
pandas
openpyxl
python-docx
pytest-playwright
pymongo
```

---

4. 🦏MongoDB with Docker

Run MongoDB Container:

```
$ docker run -d --name mongo -p 27017:27017 mongo
```

- Exposes MongoDB server on localhost:27017

Access Mongo Shell:

```
$ docker exec -it mongo mongosh
```

- You can run:

```
use testresults
db.testlogs.find().pretty()
```

---

5. Key MongoDB Tools

Tool	Use
`mongosh`	CLI client to manually interact with MongoDB
`MongoClient`	Python library (via pymongo) to programmatically connect
MongoDB Image	Official Docker image running Mongo server

---

6. 🍁Python & Mongo Integration

Sample Python Code (used in tests):

```python
from pymongo import MongoClient
import datetime

def log_to_mongo(data):
    client = MongoClient("mongodb://host.docker.internal:27017")
    db = client.testresults
    collection = db.testlogs
    collection.insert_one(data)
```

`host.docker.internal` works for Windows/Mac to refer to host system On Linux: use Docker networks or pass IP address of host manually

---

7. 🦗Docker Compose (Next Step)

To simplify multi-container setup (Playwright + Mongo), use a `docker-compose.yml`:

```
version: '3.8'
services:
  mongo:
    image: mongo
    ports:
      - "27017:27017"

  scraper:
    build: .
    volumes:
```

```
    - ./Playwright:/app/Playwright
  depends_on:
    - mongo
  command: pytest Playwright/jobs/test_play.py
```

Use:

```
$ docker-compose up --build
```

🦜 What You Know Now (SDET Perspective)

- Run containers
- Build images using Dockerfile
- Use volumes for file output
- Use official base images (Playwright, Mongo)
- Connect Python to MongoDB in containerized environment
- Ready to use Docker Compose for multi-container setup

This is a solid Docker foundation for your SDET work.

Shall we now move to Docker Compose implementation and complete the setup?