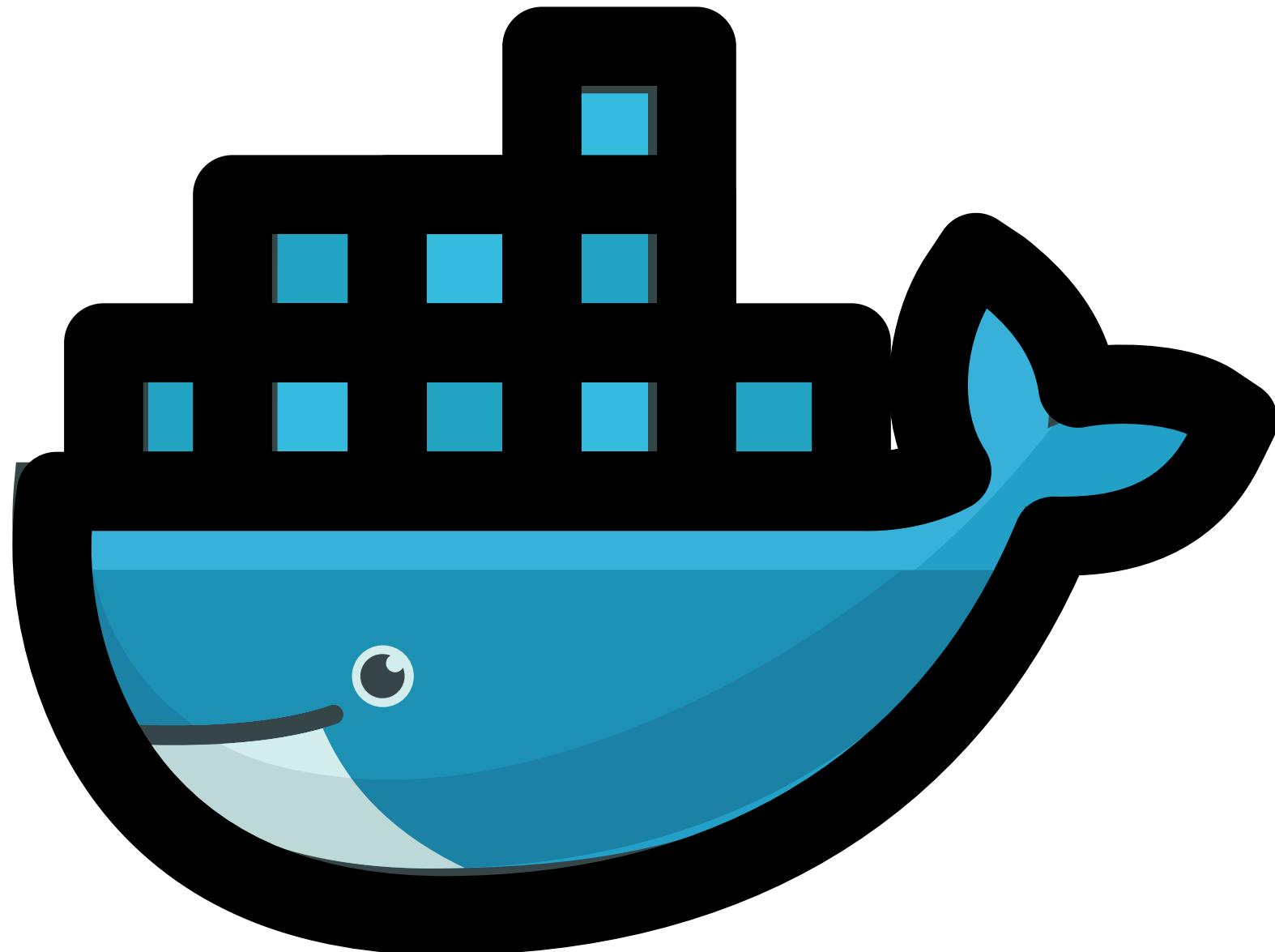


# DOCKER ESSENTIALS



- **Class:** 22MMT - Linux operating system and applications
- **Teacher:** Le Ha Minh
- **Group:** 01
- **Students:**
  - 22127154 - Nguyen Gia Huy
  - 22127210 - Pham Anh Khoi
  - 22127420 - Nguyen Ha Nam Tran

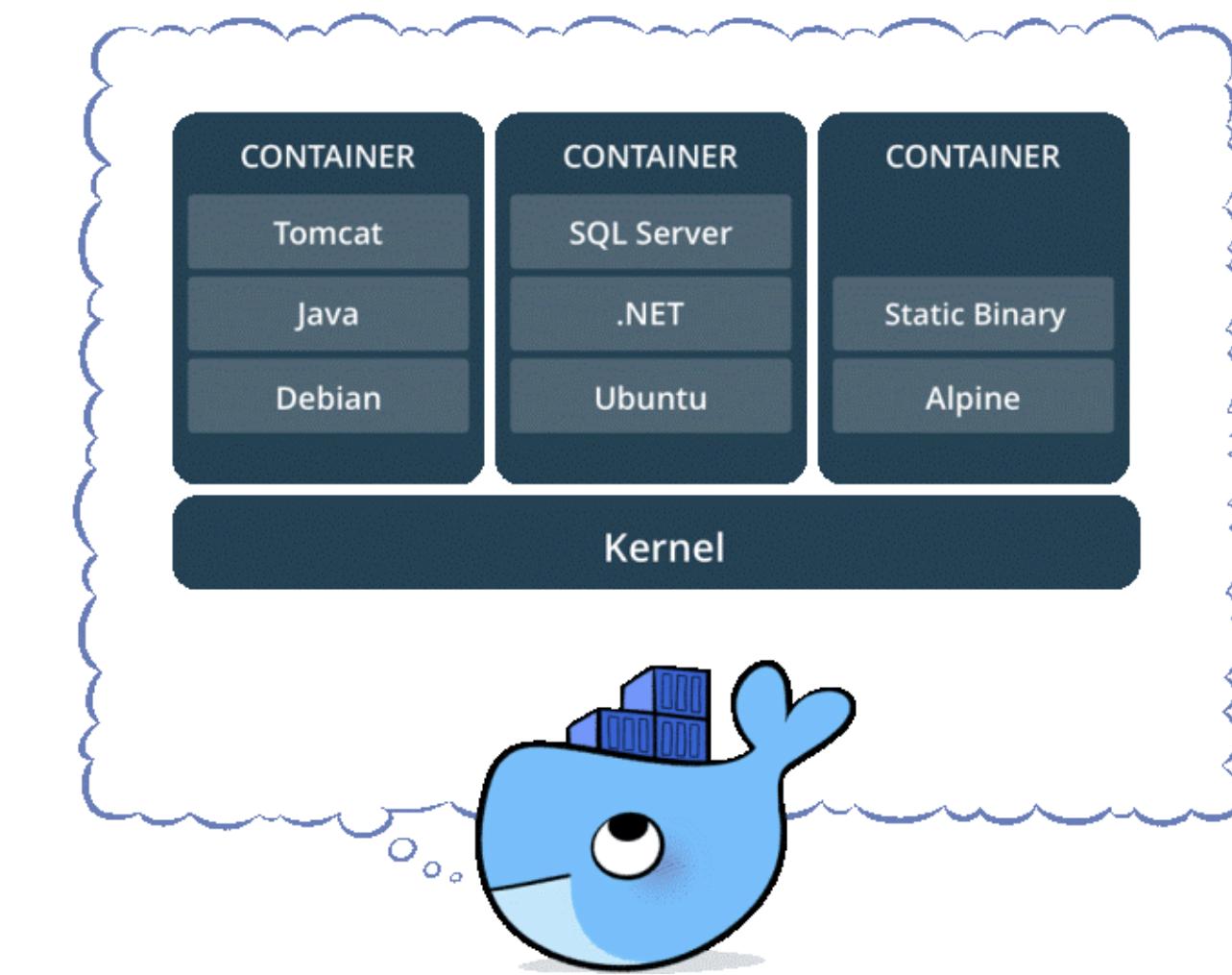
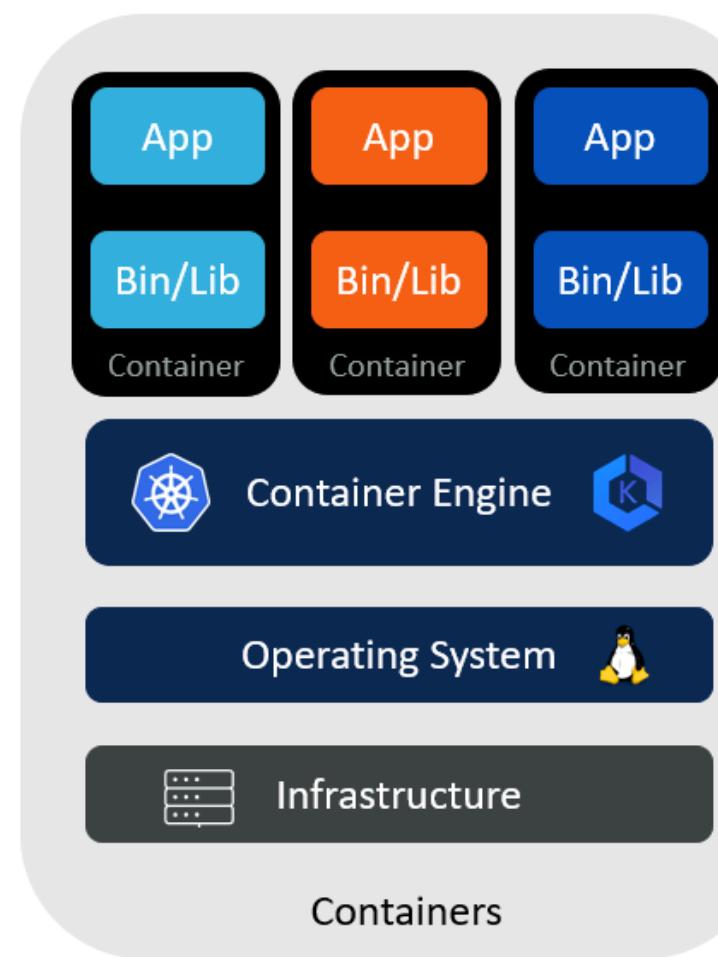
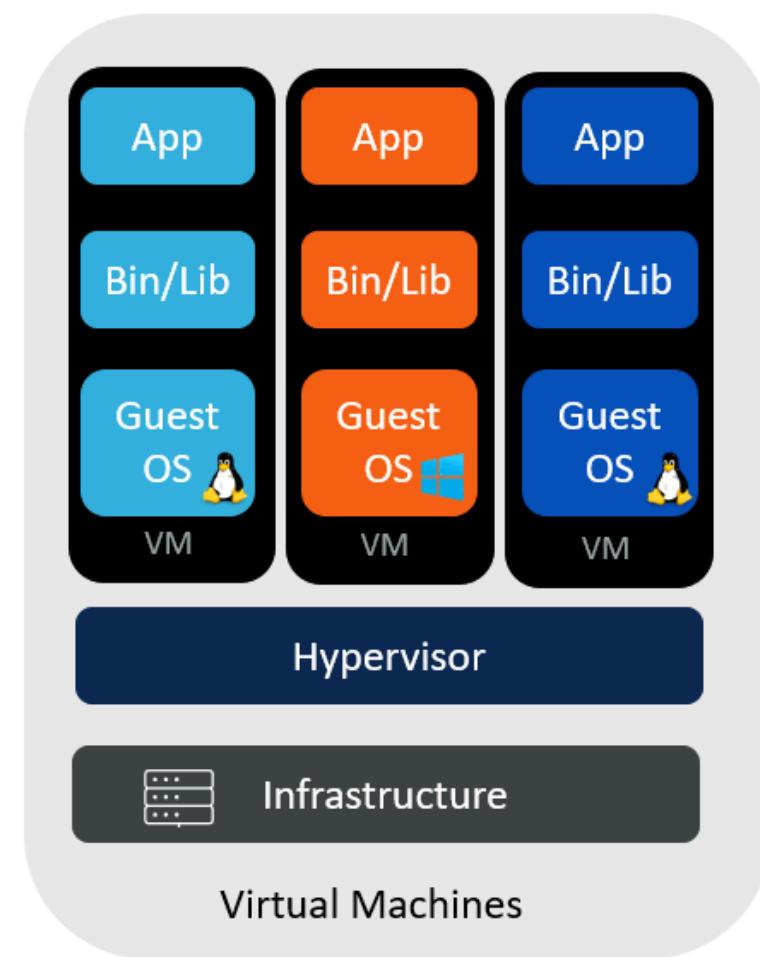
# TABLE OF CONTENT

- Foundations
- Orchestration
- Demo

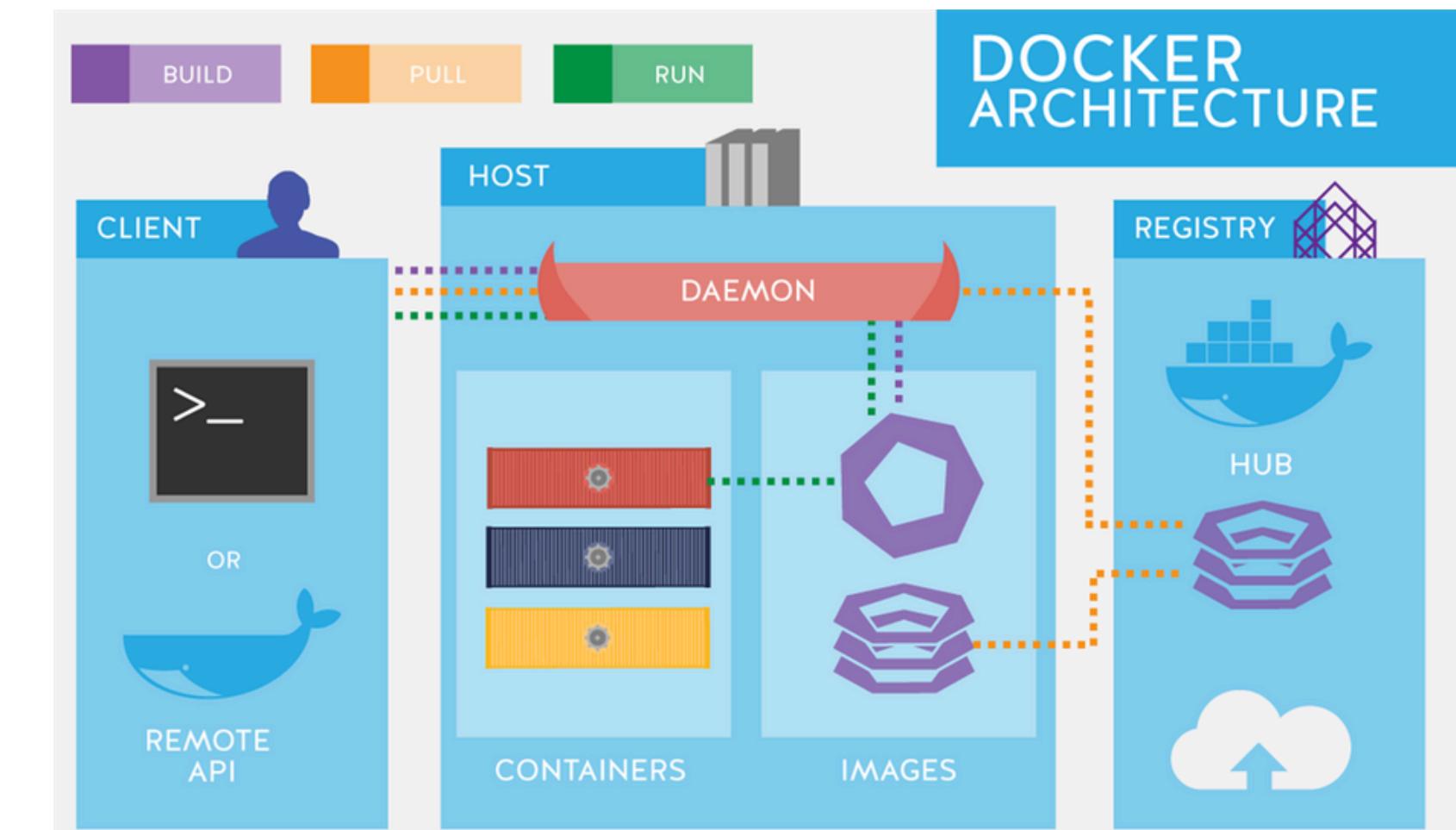
# Introduce to Docker

## What is Docker?

- Docker is a platform to build, ship, and run applications inside containers.
- Containers include everything needed to run the app: code, dependencies, runtime.



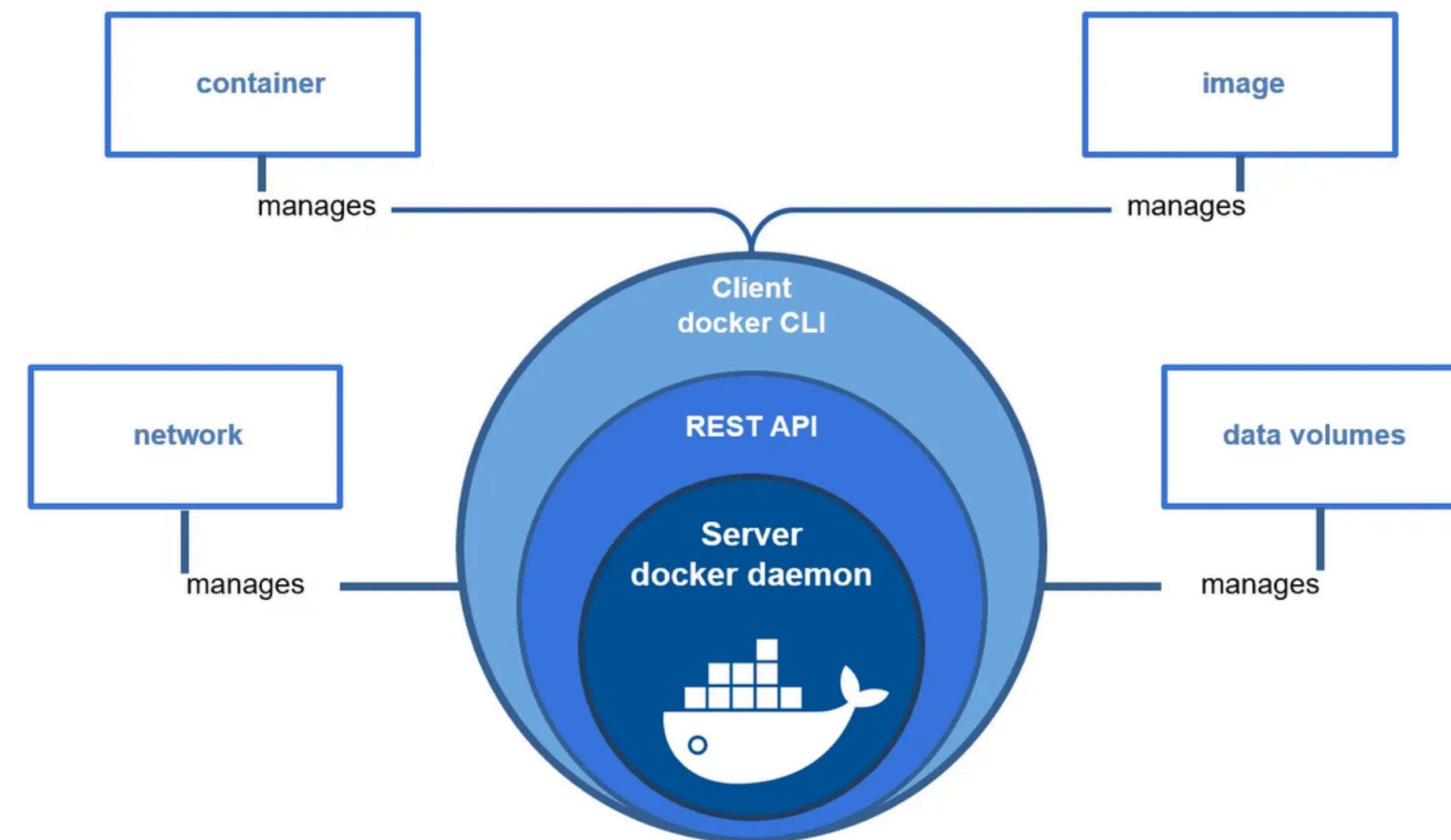
# Docker Architecture



# Docker Engine

The core client-server technology that runs and manages containers.

- **Docker Daemon:** Manages containers, images, networks, and volumes
- **Docker API:** Interface for tools and Docker Client to interact with the Daemon.
- **Docker CLI:** Command-line tool for sending commands

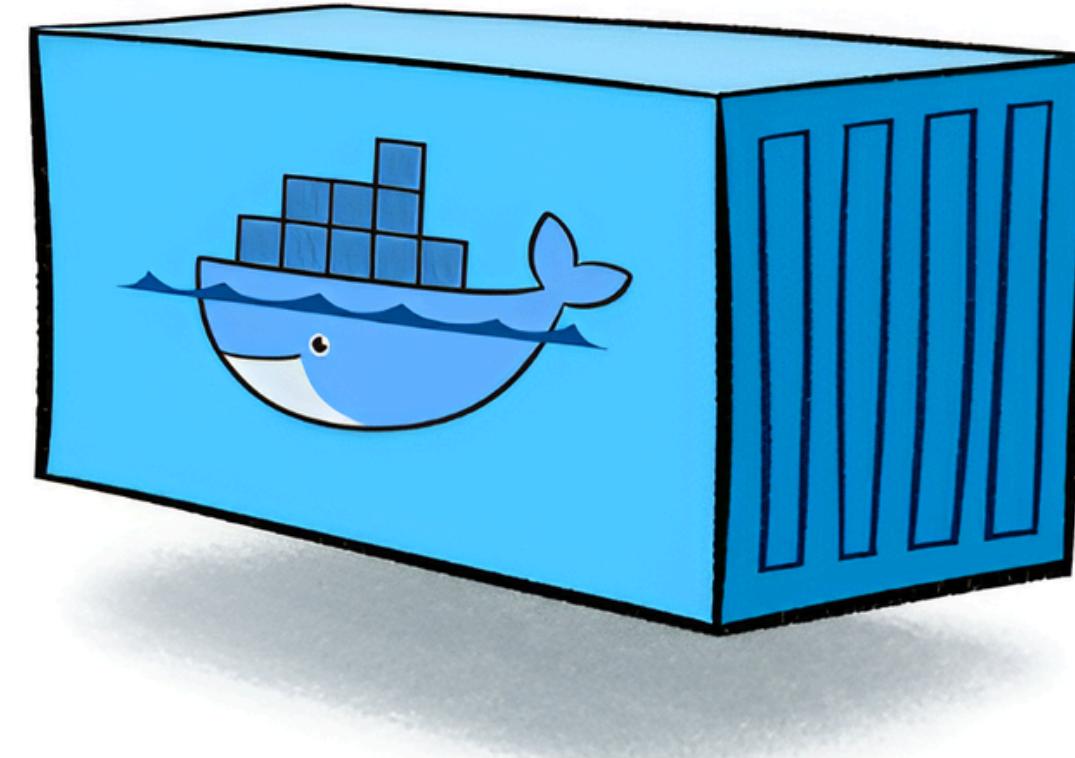


# Docker Container

Runnable instance of an image containing an app and its dependencies.

Components:

- **Application:** Core code or service (e.g., web server, database).
- **Dependencies:** Libraries, runtime, and configs for consistent execution.
- **Isolation:** Runs in a sandbox, sharing the host OS kernel.

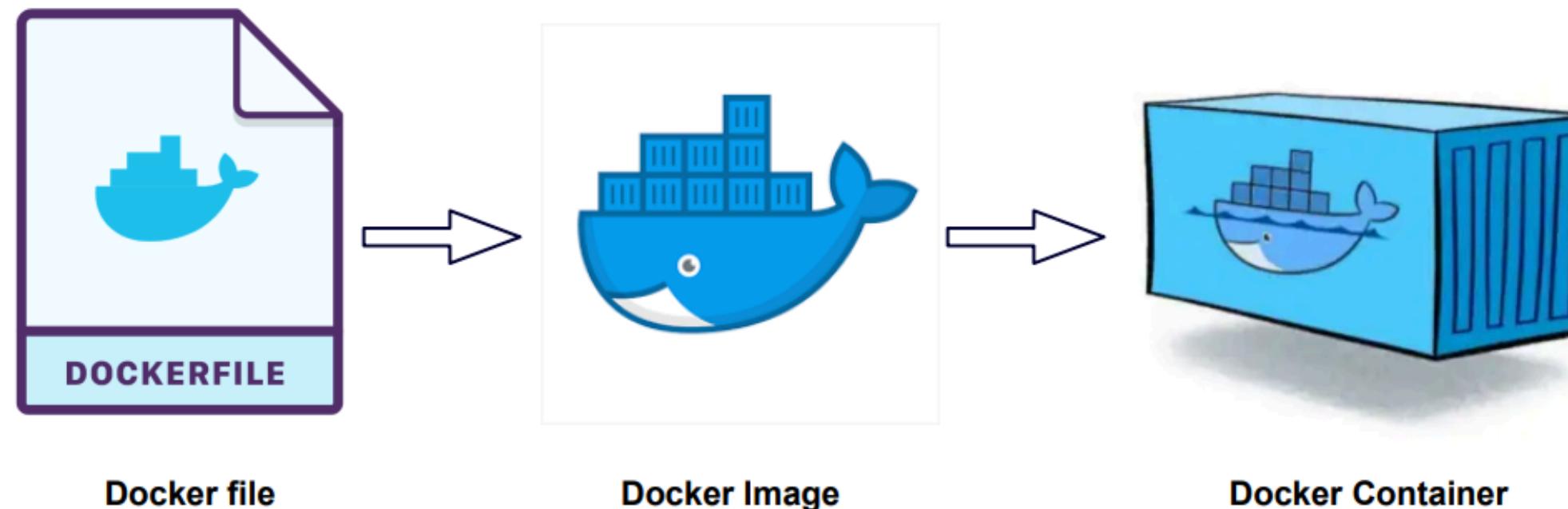


# DockerFile & Image

Read-only template for creating Docker Containers.

How to create:

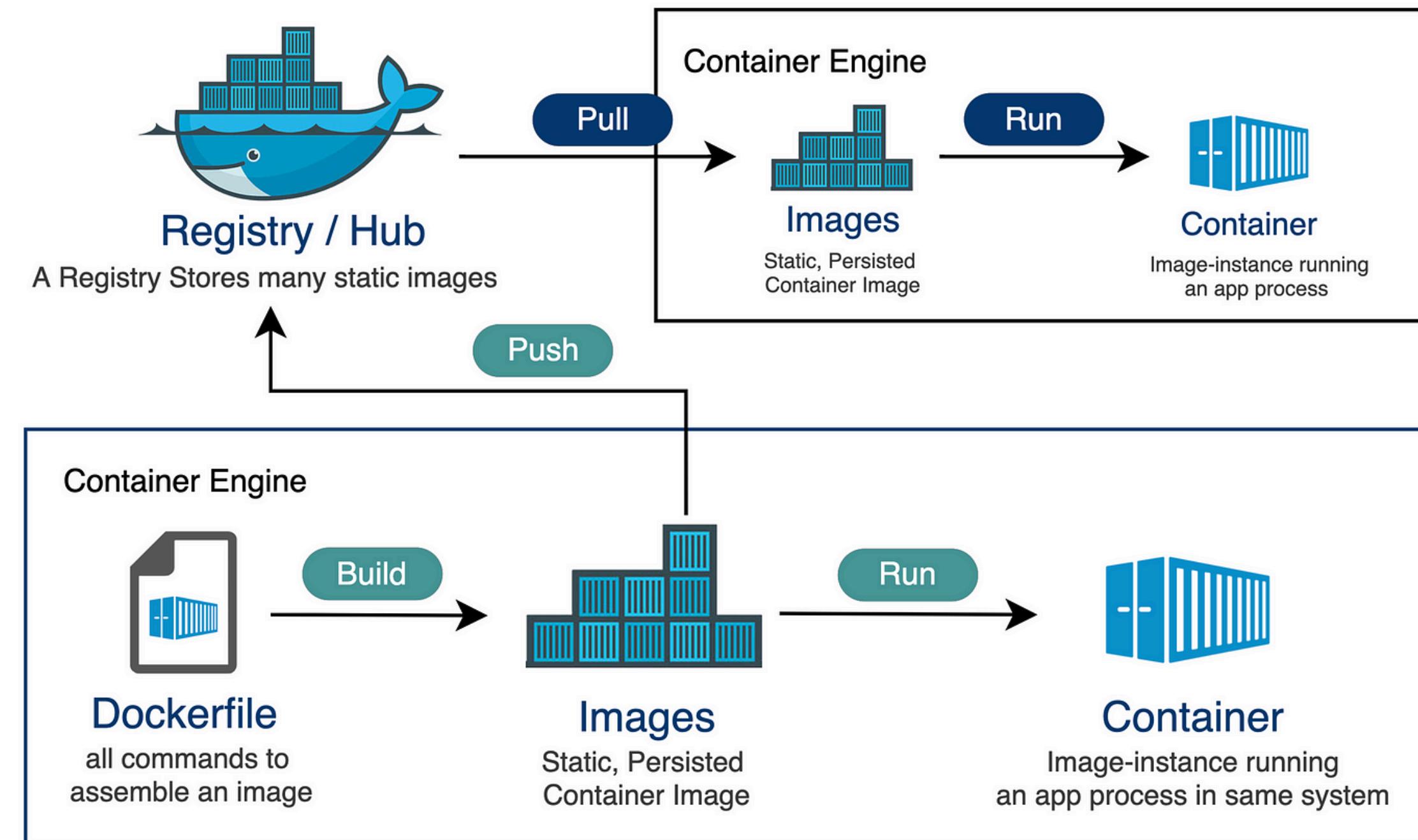
- **Dockerfile:** Script defining image build steps.
- **Contents:** Includes OS, app code, and libraries.



```
portfolio > 🐳 Dockerfile > ...
You, 2 hours ago | 3 authors (namtran1205 and others)
1 # Multi-stage build for Next.js application
2 FROM node:22-alpine AS base
3
4 # Install dependencies only when needed
5 FROM base AS deps
6 RUN apk add --no-cache libc6-compat
7 WORKDIR /app
8
9 # Copy package files
10 COPY package.json package-lock.json* ./
11 RUN npm ci
12
13 # Rebuild the source code only when needed
14 FROM base AS builder
15 WORKDIR /app
16 COPY --from=deps /app/node_modules ./node_modules
17 COPY . .
18
19 # Next.js collects completely anonymous telemetry data
20 ENV NEXT_TELEMETRY_DISABLED=1
21
22 # Build the application
23 RUN npm run build
```

# Docker Registry

Storage and distribution system for Docker images.

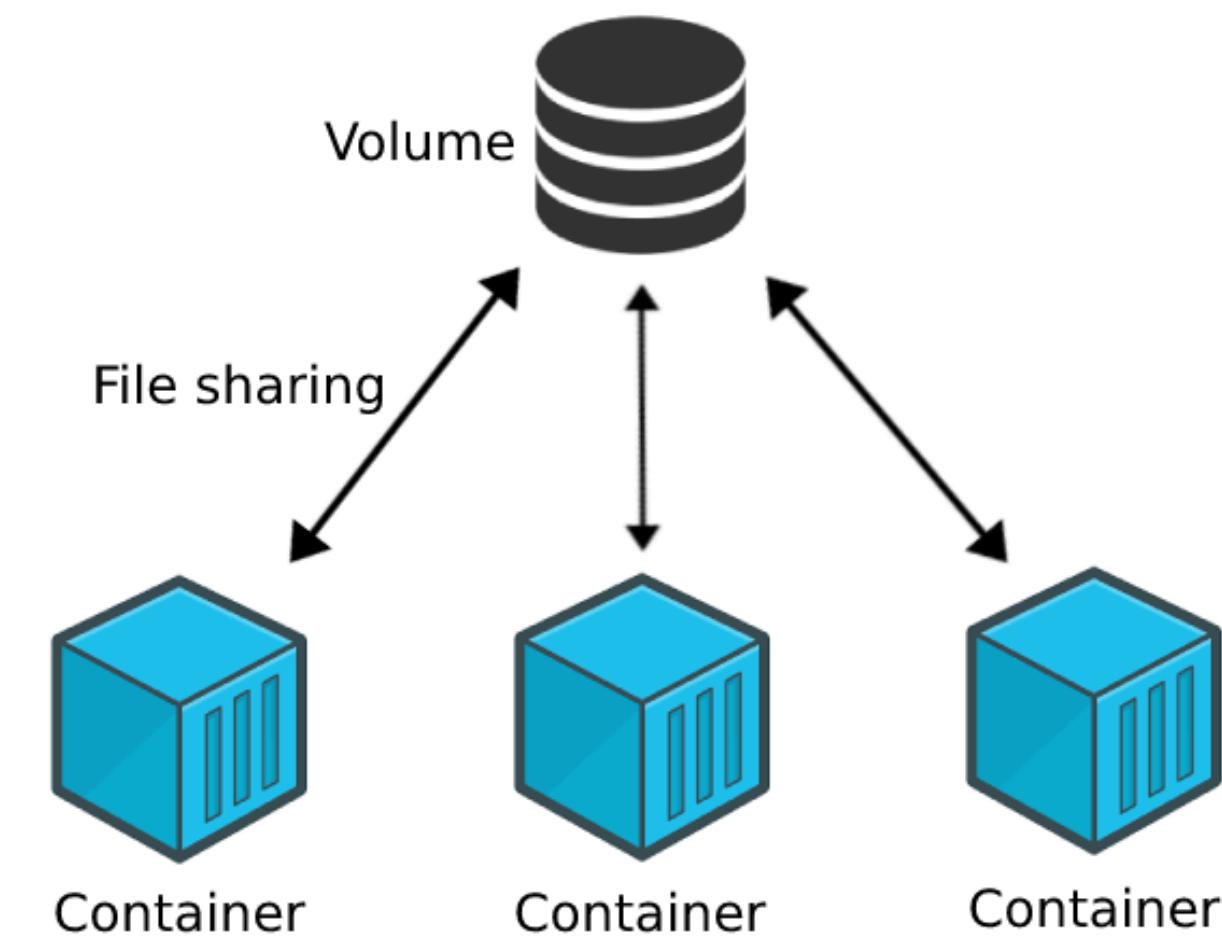
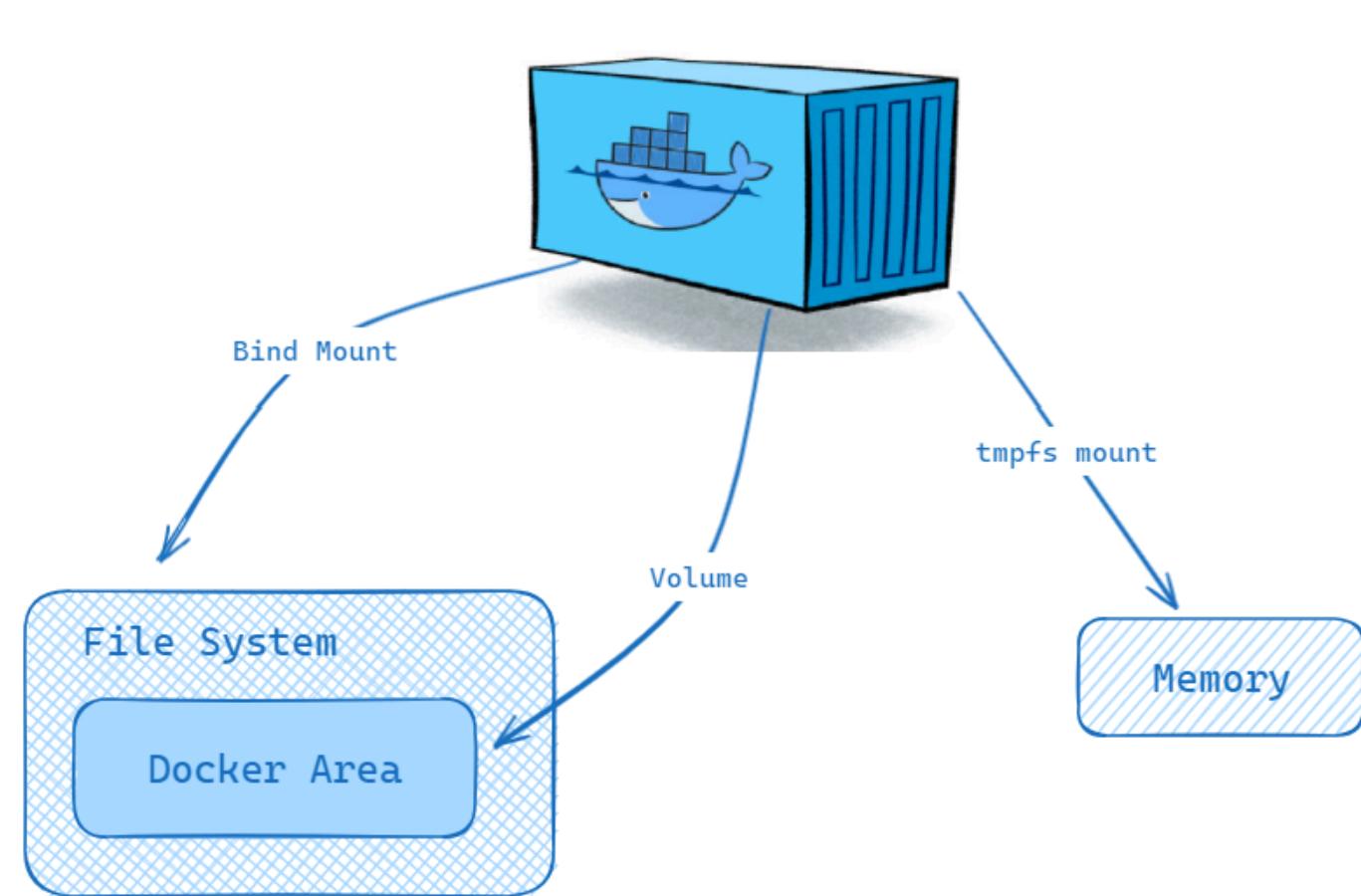


# Docker Volumes

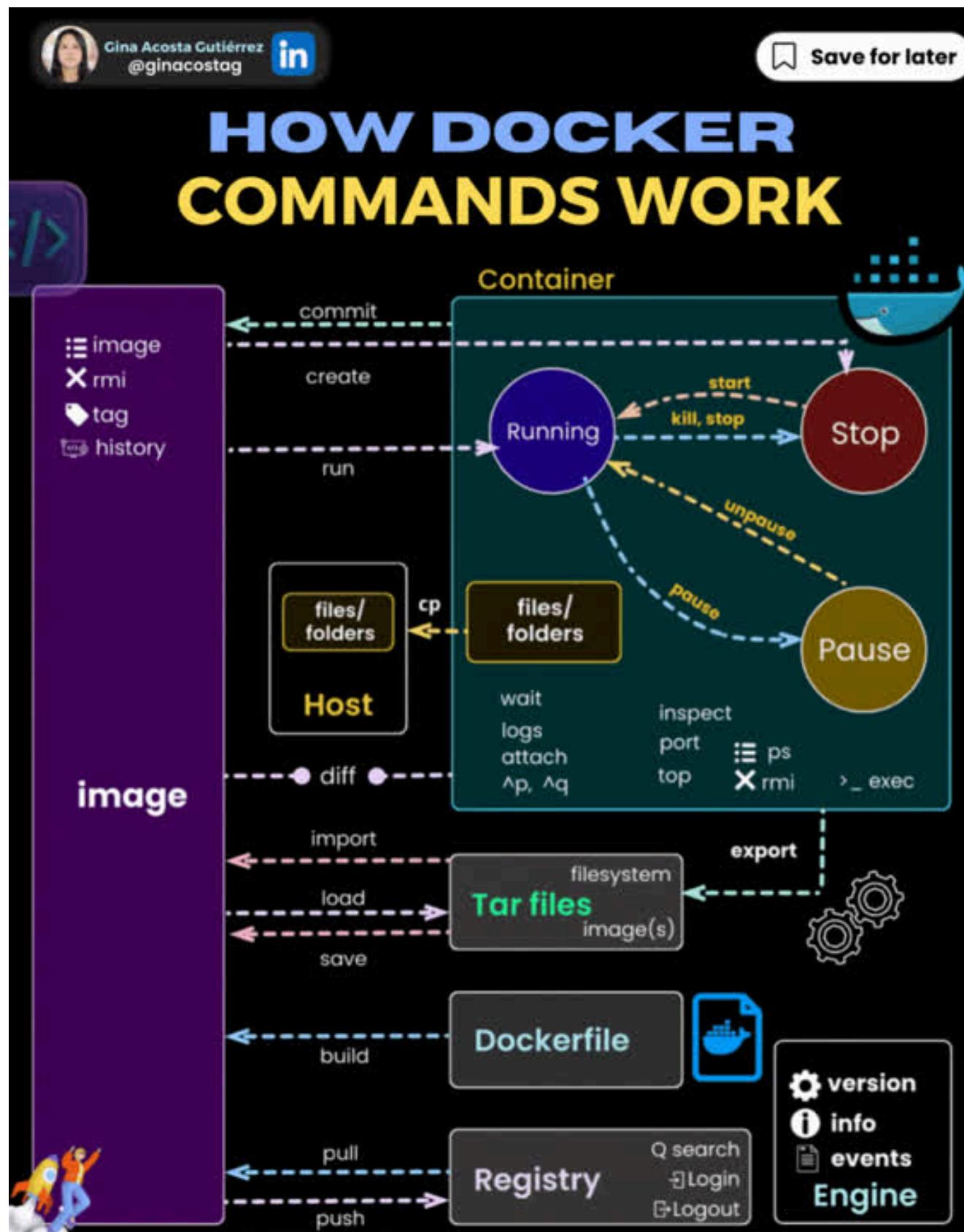
**Purpose:** Preserves data when containers stop or are deleted.

**Types:**

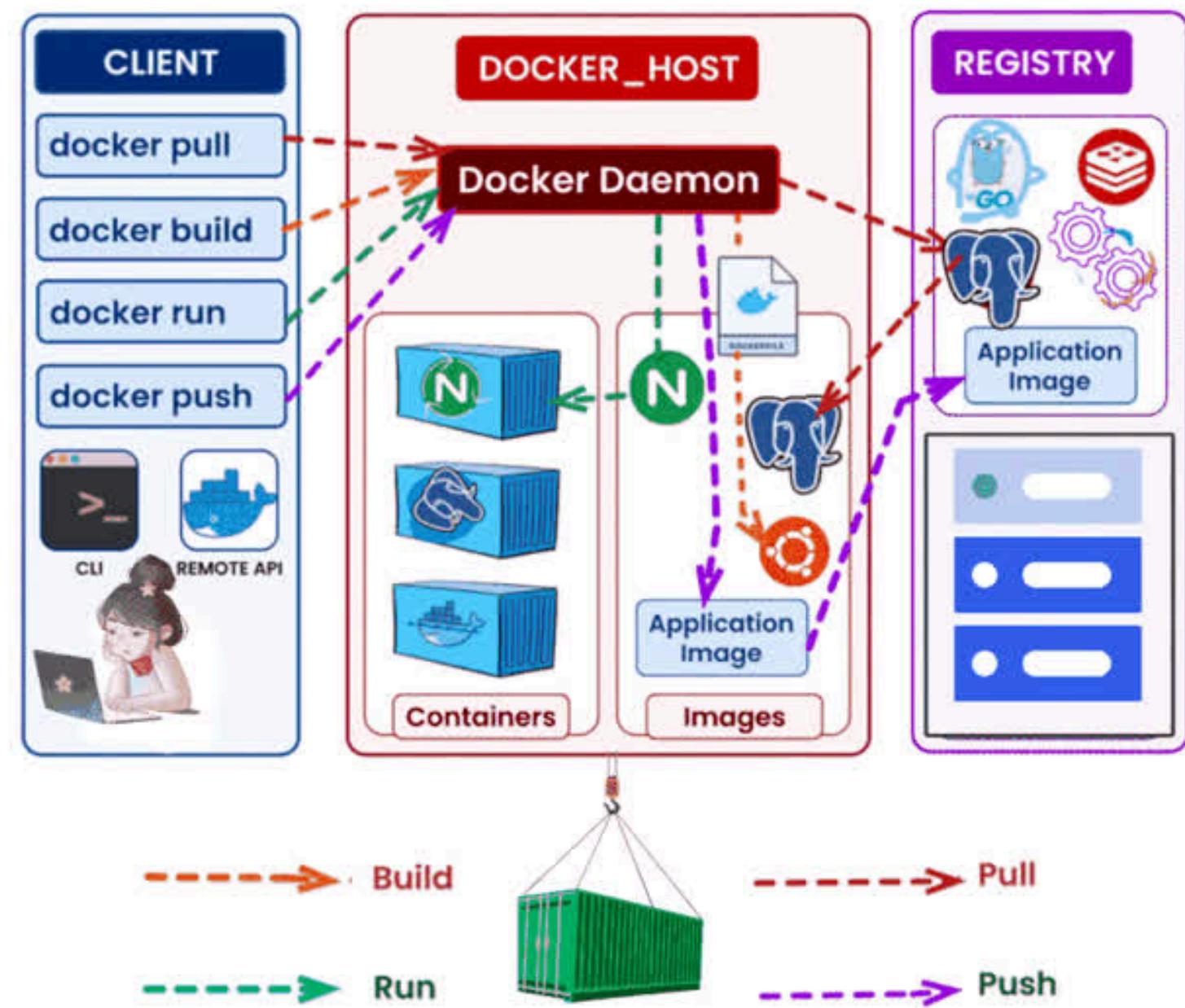
- Bind Mounts: Links container to host filesystem.
- Named Volumes: Docker-managed, reusable storage.
- Anonymous Volumes: Temporary, Docker-managed storage.



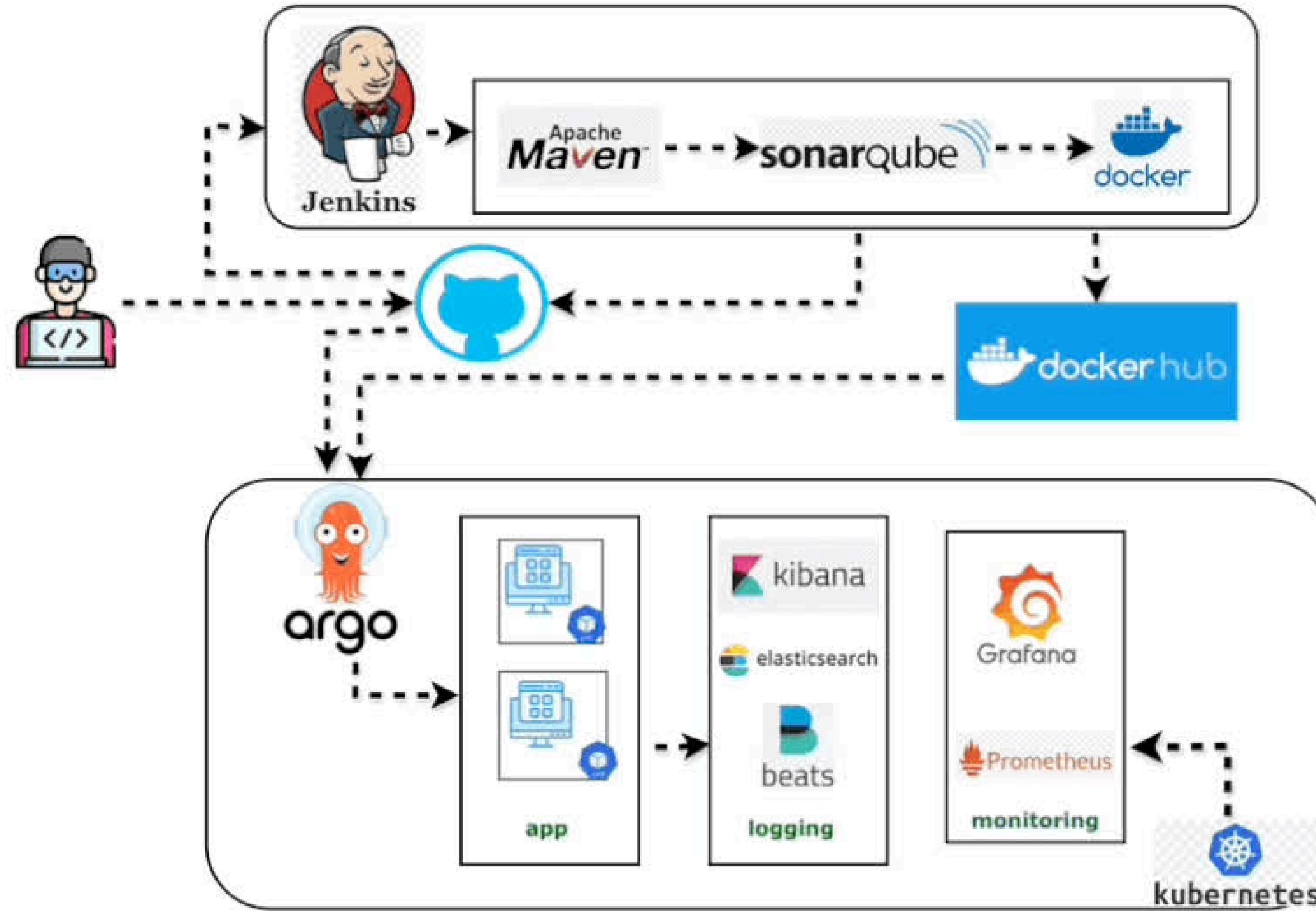
# Docker Commands



## DOCKER ARCHITECTURE



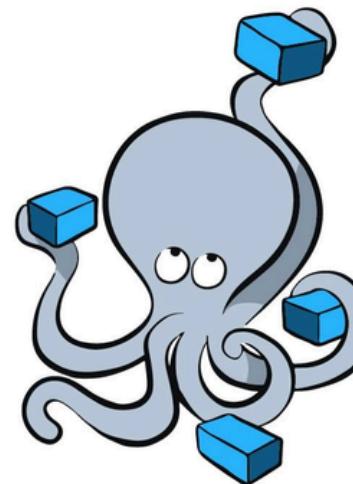
# Docker in CI/CD



# Container Orchestration



# Evolution: Compose → Swarm → Kubernetes



docker  
Compose



# Docker Compose Architecture

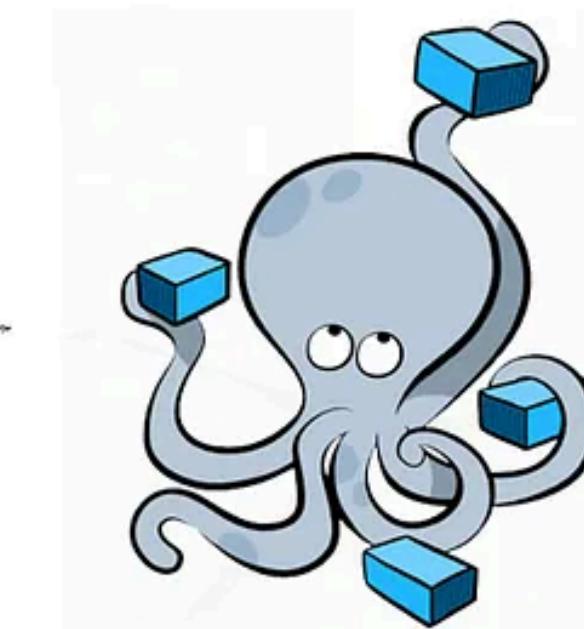
```
docker-compose.yaml
```

```
services:
  app:
    image: node:18-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

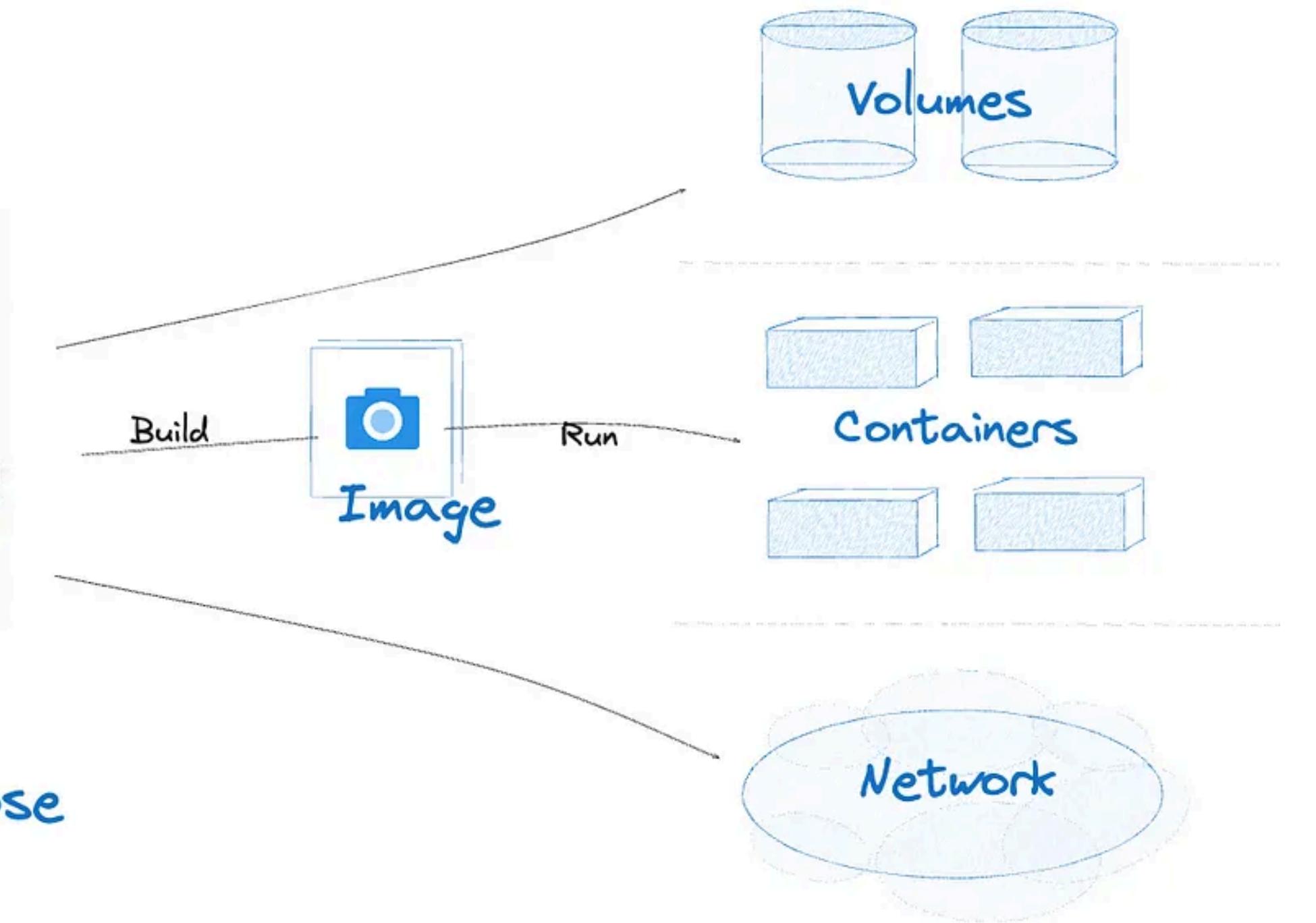
  mysql:
    image: mysql:8.0
    volumes:
      - todo-mysql-data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: todos

  volumes:
    todo-mysql-data:
```

Config YAML



Docker Compose



# Compose Deep Dive - File Structure

```
1 services:
2   # MongoDB Container
3   > mongodb: ...
4
5   # Redis Container
6   > redis: ...
7
8   # Backend API
9   > backend: ...
10
11   # Frontend Application
12   > frontend: ...
13
14   # Networks
15   networks:
16     quiz-network:
17       driver: bridge
18       name: docker-quiz-network
19
20   # Volumes (only used when MongoDB/Redis containers are active)
21   volumes:
22     mongodb_data:
23       driver: local
24       name: docker-quiz-mongodb-data
25     redis_data:
26       driver: local
27       name: docker-quiz-redis-data
```

```
1 services:
2   # MongoDB Container
3   ▷ Run Service
4   mongodb:
5     image: mongo:latest
6     container_name: docker-quiz-mongodb
7     restart: unless-stopped
8     profiles: ["local", "local-dev"]
9     environment:
10      MONGO_INITDB_ROOT_USERNAME: ${MONGO_ROOT_USER:-admin}
11      MONGO_INITDB_ROOT_PASSWORD: ${MONGO_ROOT_PASSWORD:-password123}
12      MONGO_INITDB_DATABASE: ${MONGO_INIT_DB:-docker_quiz_game}
13      # Pass environment for init script
14      ENVIRONMENT: ${ENVIRONMENT:-production}
15    ports:
16      - "${MONGO_PORT:-27017}:27017"
17    volumes:
18      - mongodb_data:/data/db
19      - ./mongo-init.js:/docker-entrypoint-initdb.d/mongo-init.js:ro
20    networks:
21      - quiz-network
22    healthcheck:
23      test: ["CMD", "mongosh", "--eval", "db.adminCommand('ping')"]
24      interval: 30s
25      timeout: 10s
26      retries: 3
27      start_period: 40s
28   # Redis Container
29   ▷ Run Service
30   redis:...
```

# Compose Deep Dive – File Structure

```
1 services:  
48 # Backend API  
49   ▷ Run Service  
50     backend:  
51       build:  
52         context: ./backend  
53         dockerfile: Dockerfile  
54         container_name: docker-quiz-backend  
55         restart: unless-stopped  
56     environment: ...  
57     ports:  
58       - "${BACKEND_PORT:-8000}:8000"  
59     networks:  
60       - quiz-network  
61     # Conditional dependencies based on profile  
62     depends_on:  
63       mongodb:  
64         condition: service_healthy  
65         required: false  
66       redis:  
67         condition: service_healthy  
68         required: false  
69     profiles: ...  
70     healthcheck:  
71       test: ["CMD", "curl", "-f", "http://localhost:8000/health"]  
72       interval: 30s  
73       timeout: 10s  
74       retries: 5  
75       start_period: 60s  
76   > deploy: ...  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101
```

docker-compose.override.yml

```
1 services:  
2   # Backend API (using pre-built image)  
3   ▷ Run Service  
4     backend:  
5       image: hzeroxium/docker-quiz-game-backend:latest  
6       container_name: docker-quiz-backend-prod  
7       restart: unless-stopped  
8       environment: ...  
9     ports:  
10       - "${BACKEND_PORT:-8000}:8000"  
11     networks:  
12       - quiz-network  
13     profiles:  
14       - cloud  
15     healthcheck:  
16       test: ["CMD", "curl", "-f", "http://localhost:8000/health"]  
17       interval: 30s  
18       timeout: 15s  
19       retries: 5  
20       start_period: 60s  
21     deploy:  
22       resources:  
23         limits:  
24           memory: 256M  
25           cpus: "0.5"  
26         reservations:  
27           memory: 128M  
28           cpus: "0.25"  
29     logging:  
30       driver: json-file  
31       options:  
32           max-size: "10m"  
33           max-file: "3"  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54
```

docker-compose.yml

# Compose Commands



```
docker-compose up -d --build --profile <PROFILE_NAME>
# Start containers in detached mode, rebuild images, using the specified profile.

docker-compose up <SERVICE_NAME>
# Start only the specified service and its dependencies in the foreground.

docker-compose down -v
# Stop and remove all containers, networks, and **volumes** created by `up`.

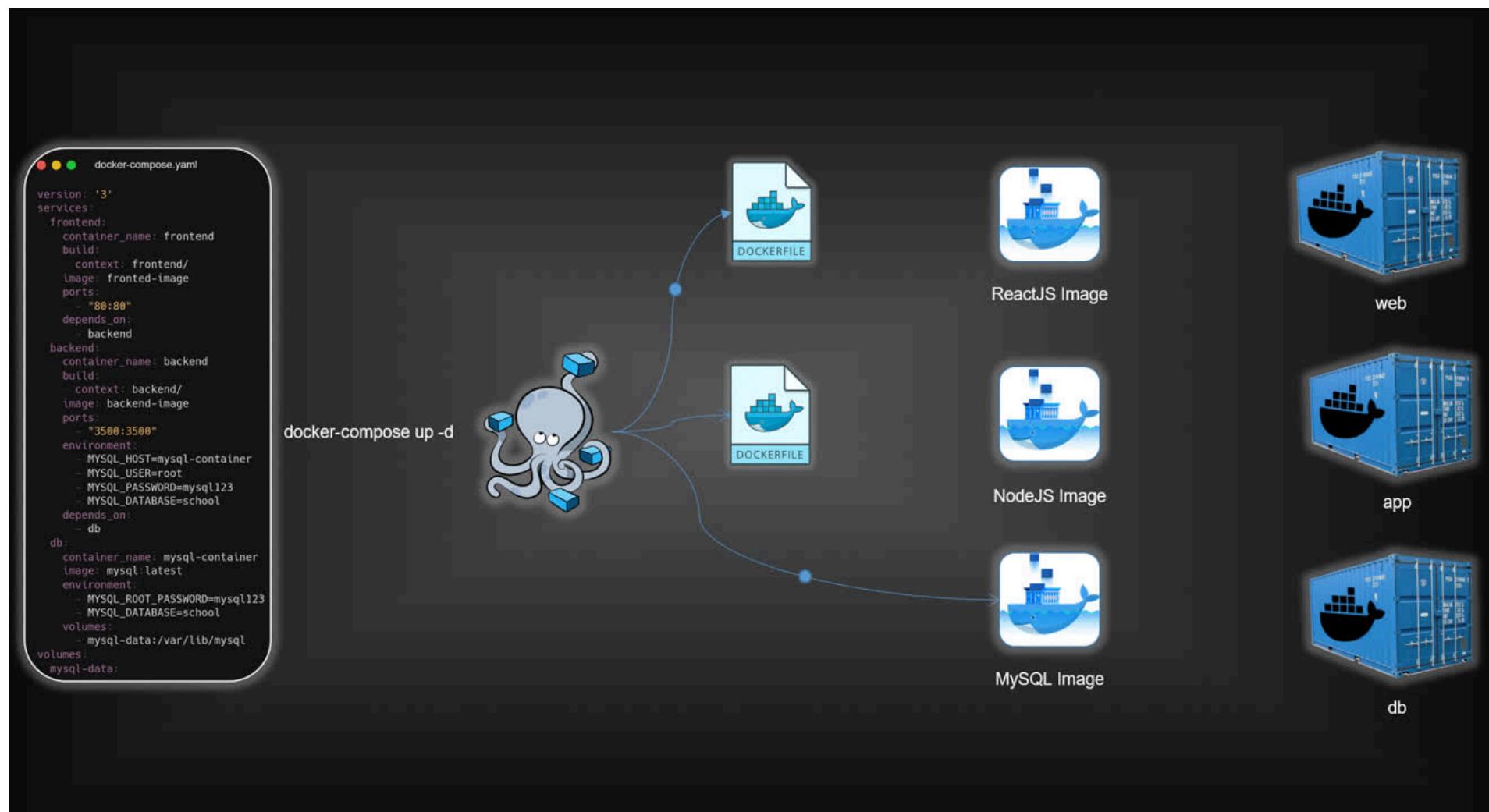
docker-compose stop <SERVICE_NAME>
# Stop the specified running service container without removing it.

docker-compose build <SERVICE_NAME>
# Build or rebuild the image for the specified service.

docker-compose ps
# List all running (and stopped) containers managed by this Compose project.

docker-compose logs
# Display logs from all services defined in the `docker-compose.yml`.

docker-compose pull
# Pull the latest images from the Docker registry for all services.
```

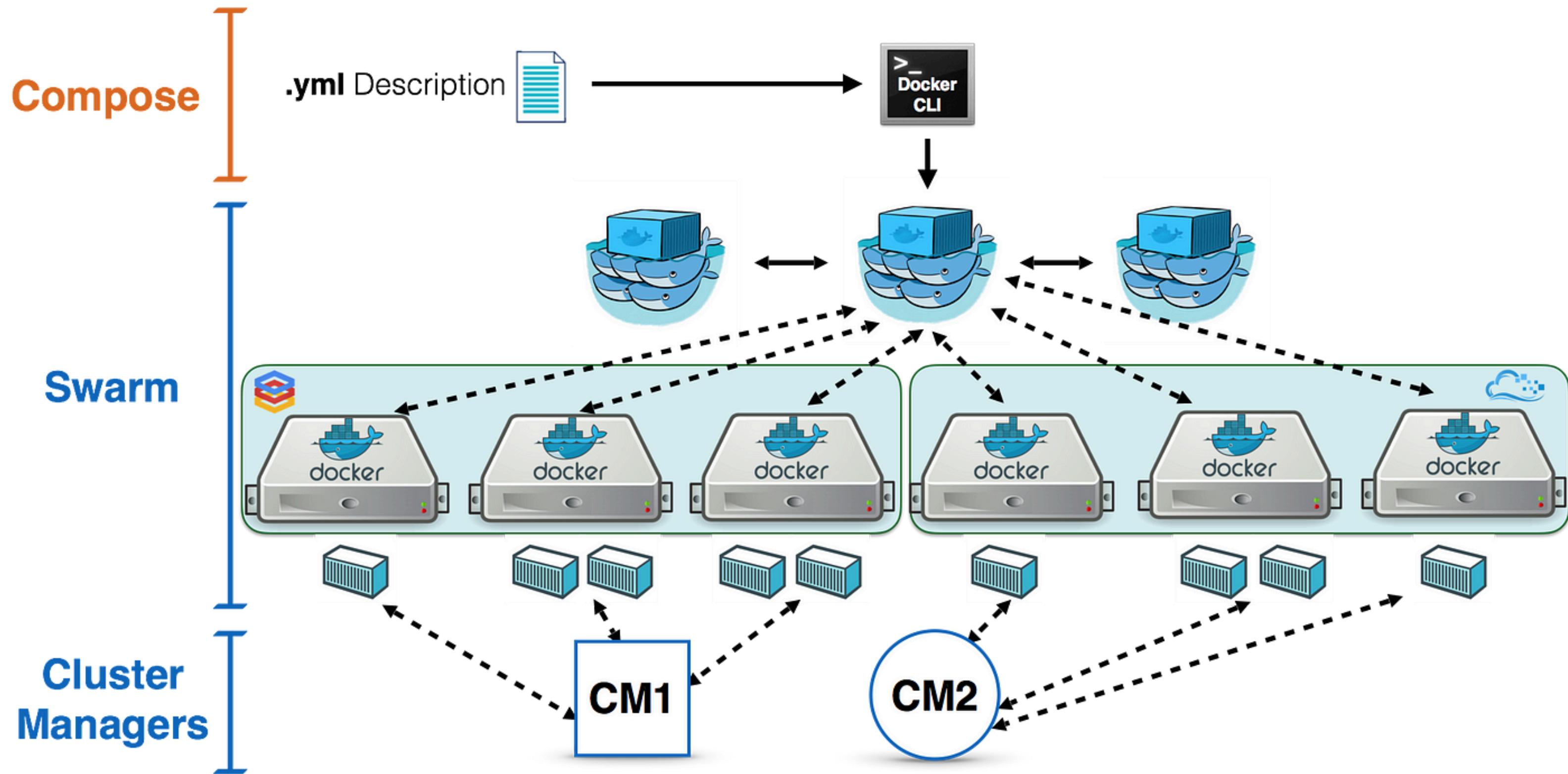


# Compose Best Practices

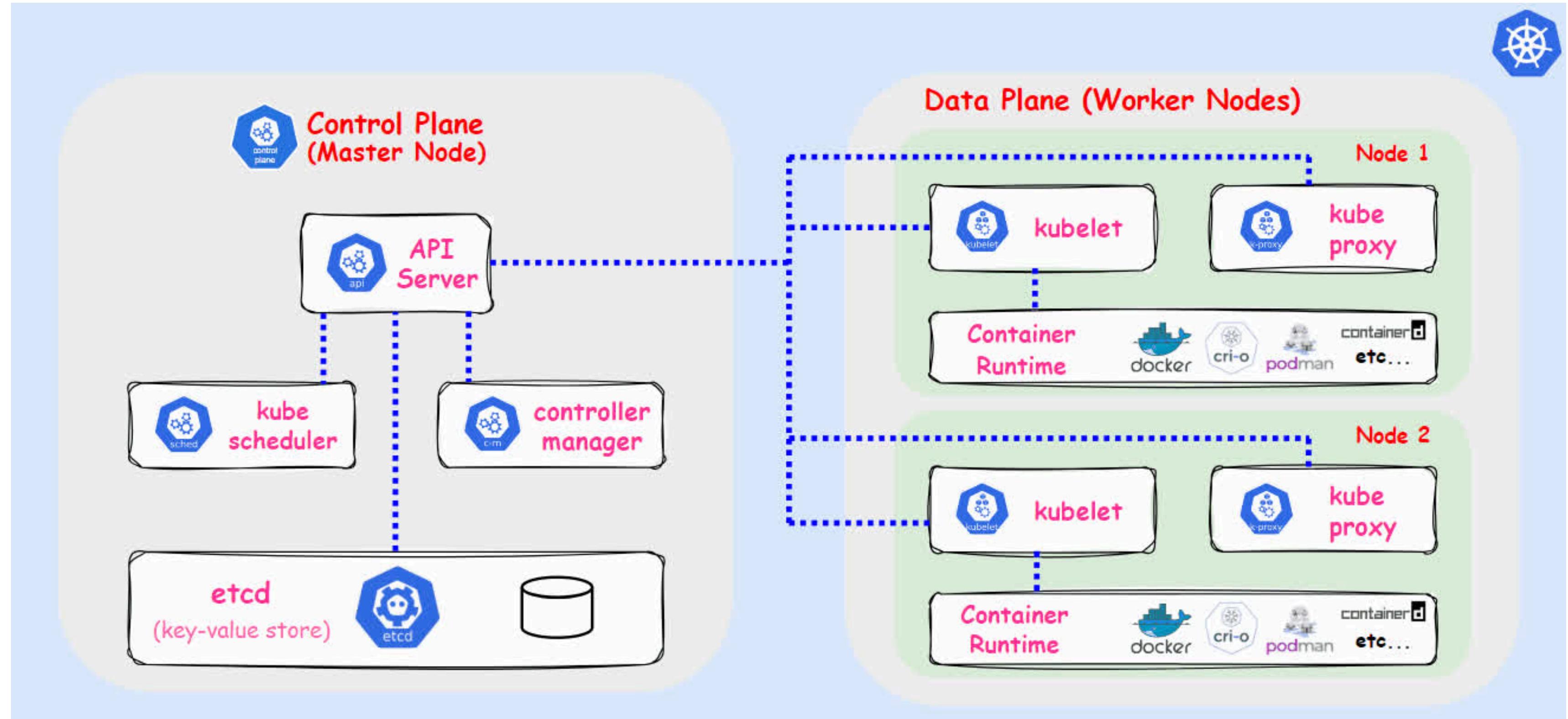
## Best Practices:

- Fix the image version to avoid discrepancies
- Set depends\_on to only start the order
- Use healthcheck to ensure service is ready
- Keep the file short, only declare services that are really needed
- ...

# Docker Swarm Overview



# Kubernetes Overview



# Demo

The image shows a split-screen demonstration. On the left, the Visual Studio Code interface displays a terminal window with deployment logs for a 'portfolio' application. The logs show the container being pulled, started, and the application running at <http://localhost:3000>. A large red play button is overlaid on the terminal window. On the right, a web browser window shows a portfolio website with a purple header featuring the letters 'TN'. The main content area says 'Xin chào, tôi là Tên của bạn' and describes the developer as a 'Full Stack Developer với niềm đam mê tạo ra những ứng dụng web hiện đại và trải nghiệm người dùng tuyệt vời'. There are buttons for 'Xem dự án' and 'Tải CV'.

```
10 NC='\[0m'
11
12 # Configuration
13 CONTAINER_NAME="portfolio_cloud"
14
15 print_success() {
16     echo -e "${GREEN} ${S1\$NC}"
17 }
18
19 print_info() {
20     echo -e "${YELLOW} ${S1\$NC}"
21 }
22
23 print_info "Stopping cloud deployment..."
24
25 # Stop and remove container
26 if docker ps -q -f name="$CONTAINER_NAME" | grep -q .; then
27     docker stop "$CONTAINER_NAME"
28     docker rm "$CONTAINER_NAME"
29     print_success "Container stopped and removed"
30 else
31     print_info "Container not running"
32 fi
33
34 print_success "Cloud deployment stopped"
```

Watch on [YouTube](#)

# THANK YOU

For your attention

