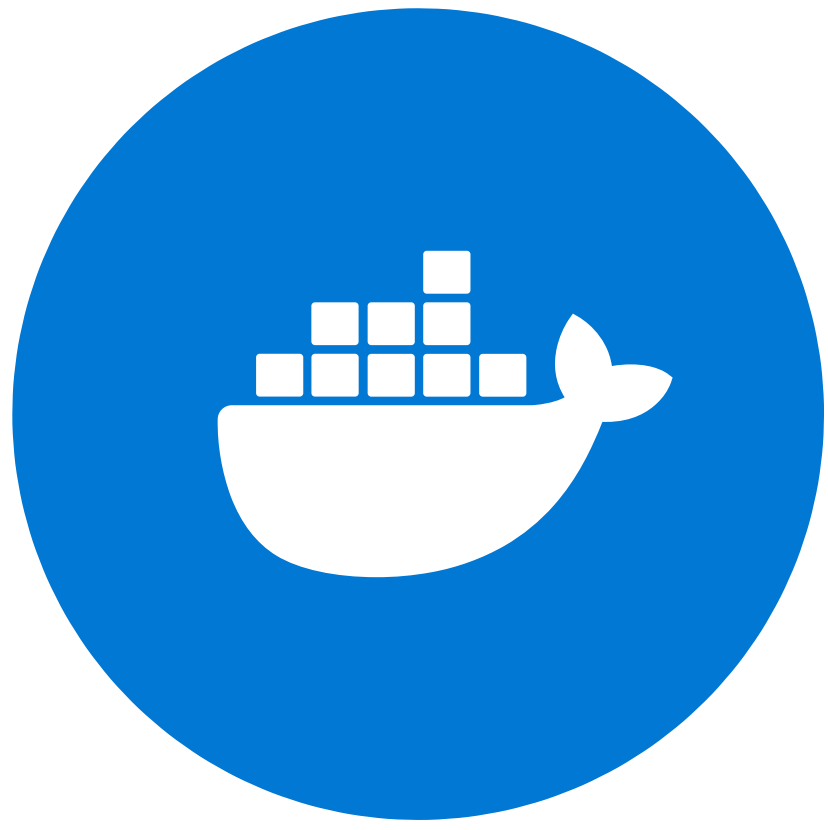
Wali Muhammad
@full-stackengineer

Docker Done Right: 10 Best Practices for Developers



Wali Muhammad
@full-stackengineer

1. Always Use Official Docker Images

- **Why:** Official images are maintained by trusted sources, ensuring security, reliability, and optimized performance.

- **Example:** Official Node.js Docker Image

```
# Use the official Node.js image from Docker Hub
FROM node:14
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "app.js"]
```

- **Tip:** Check the source of your Docker images to avoid vulnerabilities.

Wali Muhammad
@full-stackengineer

2. Reduce Docker Image Layers

- **Why:** Fewer layers lead to smaller, faster images.
- **Example:** Optimized Multi-Stage Dockerfile

```dockerfile
# Stage 1: Build the application
FROM node:14 as builder
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

# Stage 2: Create a lightweight runtime image
FROM node:14-slim
WORKDIR /app
COPY --from=builder /app/node_modules ./node_modules
COPY --from=builder /app .
EXPOSE 3000
CMD ["node", "app.js"]
```

- **Tip:** Combine related commands and use multi-stage builds to optimize your Dockerfile.

Wali Muhammad
@full-stackengineer

3. Minimize Docker Image Size

- **Why:** Smaller images are faster to deploy and use less storage.

- **Example:** Slim Dockerfile with Alpine

```
FROM node:alpine
WORKDIR /app
COPY package.json ./
RUN npm install --production
COPY . .
RUN rm -rf /tmp/* /var/tmp/*
CMD ["npm", "start"]
```

- **Tip:** Use smaller base images like “alpine” and remove unnecessary files.

Wali Muhammad
@full-stackengineer

4. Use .dockerignore to Exclude Unnecessary Files

- **Why:** Exclude unnecessary files to reduce image size and speed up builds.
- **Example:** .dockerignore Example

```
node_modules
npm-debug.log
.DS_Store
```

- Tip: Regularly update your “.dockerignore” file as your project evolves.

Wali Muhammad
@full-stackengineer

5. Leverage Docker Compose for Multi-Container Applications

- **Why:** Simplifies the management of multi-container applications.
- **Example:** Docker Compose for Web & DB

```yaml
version: "3.8"
services:
  web:
    image: nginx:latest
    ports:
      - "80:80"
    volumes:
      - ./my-app:/var/www/html
  database:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: secret
    volumes:
      - mysql-data:/var/lib/mysql
volumes:
  mysql-data:
```

- **Tip:** Use Docker Compose to easily manage and scale your services.

Wali Muhammad
@full-stackengineer

6. Utilize Docker Volumes for Persistent Data

- **Why:** Ensures data persists beyond container lifecycle.
- **Example:** Docker Volumes for Persistent Data

```
version: '3'
services:
  db:
    image: postgres:latest
    volumes:
      - db_data:/var/lib/postgresql/data
volumes:
  db_data:
```

- **Tip:** Use named volumes to manage data storage outside containers.

Wali Muhammad
@full-stackengineer

7. Implement Resource Constraints to Optimize Performance

- **Why:** Prevents any single container from hogging resources, ensuring stable operations.
- **Example:** Resource Constraints in Docker Compose

```
version: '3'
services:
  app:
    image: myapp:latest
    deploy:
      resources:
        limits:
          cpus: '0.5'
          memory: 512M
```

- **Tip:** Set CPU and memory limits to manage resources effectively.

Wali Muhammad
@full-stackengineer

8. Secure Your Docker Environment

- **Why:** Protect your containers from vulnerabilities and unauthorized access.

Example: Enable Docker Content Trust

```
export DOCKER_CONTENT_TRUST=1
```

- **Tip:** Enable Docker Content Trust to ensure only trusted images are used.

Wali Muhammad
@full-stackengineer

9. Monitor and Debug with Docker Tools

- **Why:** Regular monitoring helps maintain container health and quickly resolve issues.

- **Tip:** Use tools like ‘docker logs’, ‘docker stats’, and ‘docker events’ to stay on top of your containers.

Wali Muhammad
@full-stackengineer

10. Automate with CI/CD for Streamlined Workflows

- **Why:** Automation speeds up deployment and reduces errors.

- **Tip:** Integrate Docker with CI/CD tools like GitHub Actions, Jenkins, or GitLab CI for automated builds and deployments.

Wali Muhammad
@full-s tackengineer

Following these best practices will help you build efficient, secure, and reliable Docker containers.

Start mastering Docker today! Share this post to help others elevate their Docker skills.