Docker, Tips & Commands.

KEEP SWIPING

/sahil-saxenadx

Tip #1: Simplify Your Dockerfiles

A clean and efficient Dockerfile is key to creating reliable Docker images.

```
# Use a smaller base image for smaller final images
FROM python:3.9-slim

# Set the working directory
WORKDIR /app

# Copy requirements and install dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application code
COPY . .

# Specify the command to run the app
CMD ["python", "app.py"]
```

Tip #2: Use Multi-Stage Builds

Reduce the size of your Docker images with multi-stage builds.

```
# First stage: build
FROM golang:1.16 AS build
WORKDIR /src
COPY . .
RUN go build -o myapp

# Second stage: runtime
FROM alpine:latest
WORKDIR /app
COPY --from=build /src/myapp .
CMD ["./myapp"]
```

Tip #3: Manage Containers Efficiently

Use these commands to manage your containers effectively.

```
# List all running containers
docker ps

# List all containers (including stopped ones)
docker ps -a

# Start a container
docker start <container_id>

# Stop a container
docker stop <container_id>

# Remove a container
docker rm <container_id>

# Remove all stopped containers
docker container prune
```

Tip #4: Docker Compose for Multi-Container Applications

Simplify your multi-container applications with Docker Compose.

```yaml
version: '3'
services:
  web:
    image: nginx:latest
    ports:
      - "80:80"
  app:
    build: .
    ports:
      - "5000:5000"
    environment:
      - FLASK_ENV=development
  db:
    image: postgres:latest
    environment:
      POSTGRES_DB: mydb
      POSTGRES_USER: user
      POSTGRES_PASSWORD: password
```

Tip #5: Debugging Containers

Quickly troubleshoot your containers.

```
# Access a running container
docker exec -it <container_id> /bin/bash

# View container logs
docker logs <container_id>

# Inspect container details
docker inspect <container_id>
```

Tip #6: Clean Up Resources

Keep your Docker environment clean to save space and resources.

```
# Remove all unused images
docker image prune

# Remove all unused volumes
docker volume prune

# Remove all unused networks
docker network prune

# Remove all unused data
docker system prune
```

Tip #7: Network Containers

Connect containers to different networks for better isolation and communication.

```
# Create a network
docker network create my_network

# Run a container on a specific network
docker run -d --network my_network --name my_container my_image
```