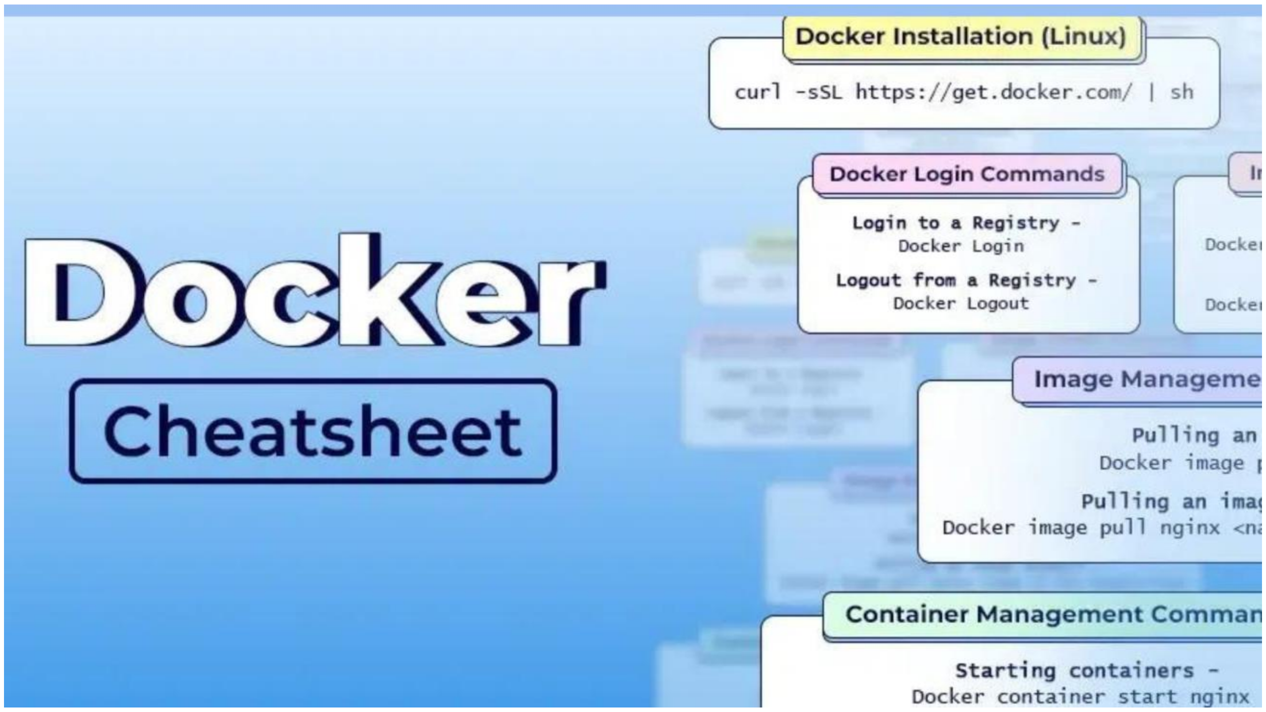

```
curl -sSL https://get.docker.com/ | sh
Docker image pull nginx <na
Docker container start nginx
```

Docker Cheat Sheet for DevOps Engineers

Docker Basics

Docker is an open-source platform that automates the deployment of applications inside lightweight containers, ensuring consistency across multiple development and release cycles.

Docker Commands

Docker Installation

```
# Install Docker on Ubuntu
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Basic Commands

```
# Check Docker version
docker --version

# Pull an image from Docker Hub
docker pull <image_name>

# List all Docker images
docker images

# Run a container from an image
docker run -it <image_name> /bin/bash

# List running containers
docker ps

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

# Stop a running container
docker stop <container_id>

# Remove a container
docker rm <container_id>

# Remove an image
docker rmi <image_name>
```

Real-Life Example

```
# Pull and run a simple web server
docker pull nginx
docker run -d -p 8080:80 nginx
```

Access the Nginx web server on your browser at `http://localhost:8080`.

Docker Compose

Docker Compose is a tool for defining and running multi-container Docker applications.

Compose File Example

```
version: '3'
services:
  web:
    image: nginx
    ports:
      - "8080:80"
  db:
    image: postgres
    environment:
      POSTGRES_PASSWORD: example
```

Docker Compose Commands

```
# Start services defined in the Compose file
docker-compose up

# Start services in detached mode
docker-compose up -d

# Stop services
docker-compose down
```

```
# View service logs
docker-compose logs
```

Real-Life Example

Create a `docker-compose.yml` file with the content above, then run:

```
docker-compose up
```

This will start both an Nginx web server and a PostgreSQL database.

Docker Networking

Docker networking allows containers to communicate with each other, either on the same host or across different hosts.

Basic Networking Commands

```
# List networks
docker network ls

# Create a network
docker network create <network_name>

# Connect a container to a network
docker network connect <network_name> <container_name>

# Disconnect a container from a network
docker network disconnect <network_name> <container_name>
```

Real-Life Example

```
# Create a network
docker network create my_network

# Run two containers in the same network
docker run -d --name web --network my_network nginx
docker run -d --name db --network my_network postgres

# Connect to the web container and ping the db container
docker exec -it web /bin/bash
ping db
```

Docker Images

A Docker image is a read-only template with instructions for creating a Docker container.

Managing Images

```
# Build an image from a Dockerfile
docker build -t <image_name> .

# Tag an image
docker tag <image_id> <repository>/<image_name>:<tag>

# Push an image to a repository
docker push <repository>/<image_name>:<tag>

# Remove an image
docker rmi <image_name>
```

Real-Life Example

Create a `Dockerfile` with the following content:

```
FROM ubuntu:latest
RUN apt-get update && apt-get install -y nginx
CMD ["nginx", "-g", "daemon off;"]
```

Build and run the image:

```
docker build -t my_nginx_image .
docker run -d -p 8080:80 my_nginx_image
```

Docker Containers

A Docker container is a runnable instance of an image. You can create, start, stop, move, or delete a container using Docker commands.

Basic Container Commands

```
# Run a container
docker run -d --name <container_name> <image_name>

# Start a stopped container
docker start <container_id>

# Stop a running container
docker stop <container_id>

# Restart a container
docker restart <container_id>

# View container logs
docker logs <container_id>

# Execute a command in a running container
docker exec -it <container_id> <command>
```

Real-Life Example

```
# Run a detached Nginx container
docker run -d --name my_nginx -p 8080:80 nginx

# View logs of the Nginx container
docker logs my_nginx

# Execute a bash shell inside the Nginx container
docker exec -it my_nginx /bin/bash
```

Conclusion

Docker simplifies the deployment process by encapsulating applications in containers, providing consistency across different environments. By leveraging Docker Compose, you can easily manage multi-container applications. Docker networking ensures seamless communication between containers, while Docker images and containers allow for efficient application management and scaling.

With this cheat sheet, you should have a foundational understanding of Docker and its core components, enabling you to streamline your DevOps workflows effectively.

Connect and Follow Me on Socials

LINKDIN | **GITHUB** |**TWITTER**