Docker Cheat Sheet

Below I have listed a Docker Commands Cheatsheet. This Cheatsheet will help DevOps Engineers and others using docker to quickly find commonly used Docker commands. It is useful for both beginner as well as advanced users managing production workloads.

Commands:

1. Basic Docker Commands

- **To check Docker version:**

  ```
  docker --version
  ```

  Example*:* Docker version 24.0.5

- **To check system information:**

  ```
  docker info
  ```

  Displays environment details.

2. Working with Containers

- **Run a container:**

  ```
  docker run -d --name my-nginx -p 8080:80 nginx
  ```

  Runs an NGINX container in detached mode on port 8080. with name my-nginx.

- **List running containers:**

  ```
  docker ps
  ```

  Shows all active containers.

- **List all containers (including stopped ones):**

```
docker ps -a
```

- **Stop a running container:**

```
docker stop <container-id>
```

- **Restart a container:**

```
docker restart <container-id>
```

- **Remove a container:**

```
docker rm <container-id>
```

- **View container logs:**

```
docker logs <container-id>
```

- **Execute a command in a running container:**

```
docker exec -it <container-id> /bin/sh
```

  Opens an interactive shell inside the container.

3. Working with Images

- **List all images:**

```
docker images
```

- **Pull an image from Docker Hub:**

```
docker pull ubuntu
```

- **Build an image from a Dockerfile:**

```
docker build -t my-app .
```

  Builds a Docker image named `my-app` from the current directory.

- **Remove an image:**

```
docker rmi <image-id>
```

4. Networking in Docker

- **To list available networks:**

```
docker network ls
```

- **To create a new network:**

```
docker network create my-network
```

- **Connect a running container to a network:**

```
docker network connect my-network my-nginx
```

- **Disconnect a container from a network:**

```
docker network disconnect my-network my-nginx
```

- **Bind a container to a specific network during run:**

```
docker run -d --network=my-network --name my-app my-image
```

  Starts a container and attaches it to "my-network" immediately.

- **Inspect a network:**

```
docker network inspect my-network
```

- **Remove a network:**

```
docker network rm my-network
```

5. Docker Volumes (Persistent Storage)

- **Create a new Volume:**

```
docker volume create my-volume
```

- **List all Volumes:**

```
docker volume ls
```

- **Inspect a Volume:**

```
docker volume inspect my-volume
```

- **Remove a Volume:**

```
docker volume rm my-volume
```

- **Mount a Volume to a container:**

```
docker run -d -v my-volume:/data my-app
```

Mounts "my-volume" to the "/data" directory inside the container.

6. Advanced Docker Usage

- **Run a Container with Environment Variables:**

```
docker run -e APP_ENV=production my-app
```

- **Limit CPU and Memory Usage of a Container:**

```
docker run --cpus=2 -m 512m nginx
```

Limits the container to 2 CPUs and 512MB RAM.

- **Export a Running Container to a Tar File:**

```
docker export <container-id> > my-container.tar
```

- **Import a Tar File as an Image:**

```
docker import my-container.tar my-new-image
```

- **Clean Up Unused Docker Objects:**

```
docker system prune -a
```

Removes all stopped containers, networks, and dangling images.

7. Docker Compose (Multi-Container Applications)

- **Start Services Using `docker-compose.yml`:**

```
docker-compose up -d
```

- **Stop Services:**

```
docker-compose down
```

- **Rebuild and Restart Services:**

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

- **Check Logs for Services:**

```
docker-compose logs -f
```

8. Docker Registry (Pushing and Pulling Images)

- **Login to Docker Hub:**

```
docker login
```

- **Tag an Image for Pushing:**

```
docker tag my-app:latest myrepo/my-app:v1
```

- **Push an Image to Docker Hub:**

```
docker push myrepo/my-app:v1
```

- **Pull an Image from Docker Hub:**

```
docker pull myrepo/my-app:v1
```

9. Docker in Production Use Cases

- **Deploy a Highly Available Dockerized App with Load Balancing:**

```
docker run -d --name my-app -p 8080:80 --restart=always my-app
```

  Ensures the container restarts automatically if it crashes.

- **Monitor Running Containers in Production:**

```
docker stats
```

  Displays real-time resource usage of running containers.

This can be used as a quick reference in development, troubleshooting and for production environment.

Thanks for reading !

If you found this post useful, give it a like👍

Repost♻️

Follow @Bala Vignesh for more such posts 💯🚀