Docker tool was introduced in order to make it easier for you to create, deploy, and run applications using containers. Containers provide you the packaging of your application with all the important components it requires, like libraries and other dependencies, and ship them all out as one package. Due to this, you as a developer can be assured that your application will run on any other machine.

**Docker Architecture**
- Registry - hosts the public and official images
- Images - can be downloaded from the registry directly or implicitly when starting a container
- Containers - instances of images. Multiple containers for a single image is possible.
- Docker daemon - creating, running and monitoring containers, building and storing images
- Client - talks to daemon via http

**Docker Commands**
- To initialize swarm mode and listen to a specific interface:
  - `Docker swarm init --advertise-addr 10.10.0.2:2377`
- Join an existing swarm as manager node:
  - `Docker swarm join --token 10.10.0.2:2377`
- Join a swarm as a worker node:
  - `Docker swarm join --token worker-token`
  - `Docker swarm join --token worker-token 10.10.0.2:2377`
- List all the nodes in the swarm:
  - `Docker swarm ls`
- Create a service from an image and deploy 3 instances:
  - `Docker service create --replicas 3 -p 80:80 name=webnginx`
- List services running in the swarm:
  - `Docker service ls`
- Scale a service:
  - `Docker service scale service_name=5`
- List tasks of a service:
  - `Docker service ps service_name`

**Build**
- To build the image from the docker file and tag it:
  - `Docker build -t myapp:1.0`
- List all images that are locally stored:
  - `Docker images`
- Delete an image from the docker store:
  - `Docker rmi myrepo/myalpine:3.4`
- To pull an image from the registry:
  - `Docker pull myalpine:3.4`
  - `Docker tag alpine:3.4 myrepo/myalpine:3.4`
- Log in to a registry:
  - `Docker login my.registry.com:8000`
- Push an image to a registry:
  - `Docker push myrepo/myalpine:3.4`

**Run**
- To create and run a command:
  - `Docker run --name container_name docker_image`
  - Flags used:
    - `-d` detach container on start
    - `-rm` remove container once it stops
    - `-p` publish host ip and host port to the container port
    - `-v` define and share volume across containers
  - `Docker run --read-only sets it to read only permission`

**Ship**
- To clean up unused/dangling images:
  - `Docker image prune`
- To remove images not used in containers:
  - `Docker image prune -a`
- To prune the entire system:
  - `Docker system prune`
- To leave a swarm:
  - `Docker swarm leave`
- To remove a swarm:
  - `Docker stack rm stack_name`
- To kill all running containers:
  - `Docker kill $(docker ps -q)`
- To delete all stopped containers:
  - `Docker rm $(docker ps -a -q)`
- To delete all images:
  - `Docker rmi $(docker images -q)`

**Important Terms**
- Registry/hub - the central place where images live
- Docker machine - a VM to run docker containers
- Docker compose - a VM to run multiple containers as a system

**Orchestrate**

**Clean Up**

**Services**

**Interaction Within a Container**

**Build**

**Run**

**Ship**