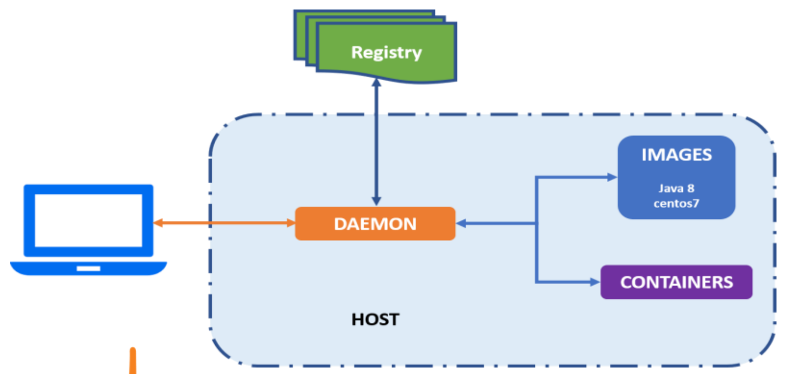
# DOCKER CHEAT SHEET

## Docker

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



Furthermore:
Docker Training Course

## Orchestrate

Commands :

- To initialize swarm mode and listen to a specific interface:
```
Docker swarm init --advertise-addr 10.1.0.2
```
- Join an existing swarm as manager node:
```
Docker swarm join --token<manager-token> 10.1.0.2:2377
```
- Join a swarm as a worker node:
```
Docker swarm join --token<worker-token> 10.1.0.2:2377
```
- List all the nodes in the swarm:
```
Docker node ls
```
- Create a service from an image and deploy 3 instances:
```
Docker service create --replicas 3 -p 80:80 name -webngix
```
- List services running in the swarm:
```
Docker service ls
```
- Scale a service:
```
Docker service scale web=5
```
- List tasks of a service:
```
Docker service ps web
```

## 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 alpine: 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 por
-v  define and share volume across containers
--read-only sets it to read only permission
```

## Ship

- To pull an image from the registry:
```
Docker pull alpine:3.4.
```
- Retag a local image with a new image name:
```
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
```

## Clean Up

- 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)
```

## Services

List of all services running in a swarm:
```
Docker service ls
```
To see all running services:
```
Docker stack services stack_name
```
To see all service logs:
```
Docker service logs stack_name service_names
```
To scale service across qualified nodes:
```
Docker service scale stack_name_service_name= replicas
```

## Interaction Within a Container

Run a command in the container:
```
Docker exe -ti container_name command.sh
```
Follow the container logs:
```
Docker logs -ft container name
```
Save a running container as an image:
```
Docker commit -m “commit message” -a “author” container_name
username/image_name: tag
```

## Important Terms

- Layer - read-only files to provision the system
- Image – a read only layer that is the base of the image
- Container - a runnable instance of the image
- 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