**Docker**

**Introduction**

Containers allow the packaging of your application (and everything that you need to run it) in a "container image". Inside a container, you can include a base operational system, libraries, files and folders, environment variables, volumes mount points, and application binaries.

A "Docker image" is a template for the execution of a container It means that you can have multiple containers running from the same image, all sharing the same behavior, which promotes the scaling and distribution of the application. These images can be stored in a remote registry to ease distribution.

Once a container is created, the execution is managed by the "Docker Engine" aka "Docker Daemon". You can interact with the Docker Engine through the "docker" command. These three primary components of Docker (client, engine, and registry) are diagramed below:

****

**Docker Engine**

**Container related commands**

Examples:

All examples provided here work in RHEL.

**1. Run a container in an interactive mode:**

$ docker run it ubuntu /bin/bash # Run a bash shell inside an image.

**2. Run a container in detached mode:**

$ docker run name mywildfly d p 8080:8080 jboss/wildfly

**3. Run a detached container in a previously created docker network:**

$ docker network create mynetwork

$ docker run name mywildflynet d net mynetwork p 8080:8080 jboss/wildfly

**4. Run a detached container mounting a local folder inside the container:**

$ docker run name mywildflyvolume d \

v myfolder/:/opt/jboss/wildfly/standalone/deployments/ \

p 8080:8080 jboss/wildfly

**5. Follow the logs of a specific container**

$ docker logs f mywildfly

$ docker logs f <containername>

**6. List containers**

$ docker ps # List only active containers

$ docker ps a # List all containers

**7. Stop a container**

$ docker stop <containername> # Stop a container

$ docker stop t 1 <containername> # Stop a container (timeout = 1 second)

**8. Remove a container**

$ docker rm <containername> # Remove a stopped container

$ docker rm f <containername> # Remove a stopped container. Force stop if it is active

$ docker rm f $(docker ps aq) # Remove all containers

$ docker rm $(docker ps q f "status=exited”) # Remove all stopped containers

**9. Execute a new process in an existing container**

$ docker exec it mywildfly /bin/bash

# Executes and access bash inside a WildFly container

**Docker Cheat Sheet**

|  |  |
| --- | --- |
| attach | Attach a running container to view its ongoing output or to control it interactively |
| commit | Create a new image from a container’s changes |
| cp | Copy files/folders between a container and the local filesystem |
| create | Create a new container |
| diff | Inspect changes on a container’s filesystem |
| exec | Run a command in a running container |
| export | Export the contents of a container’s filesystem as a ‘.tar’ archive |
| kill | Kill a running container using SIGKILL or a specified signal |
| logs | Fetch the logs of a container |
| pause | Pause all processes within a container |
| port | List portmappings, or lookup the publicfacing port that is NATed to the PRIVATE\_PORT |
| ps | List all containers |
| rename | Rename a container |
| restart | Restart a container |
| rm | Remove/delete one or more containers |
| run | Run a command in a new container |
| start | Start one or more containers |
| stop | Stop a container by sending SIGTERM then SIGKILL after a grace |
| top | Display the running processes of a container |
| unpause | Unpause all processes within a container |

**Image-related commands.**

**Examples**

**1. Build an image using a Dockerfile**

$ docker build t [username/]<imagename>[:tag] <dockerfilepath>

# Build an image

$ docker build t myimage:latest .

# Build an image called myimage using the Dockerfile in the same folder where the command was executed.

**2. Check the history of an image**

$ docker history jboss/wildfly #Check the history of the jboss/wildfly image

$ docker history [username/]<imagename>[:tag]

# Check the history of an image

**3. List the images**

$ docker images

**4. Remove an image from the local registry**

$ docker rmi [username/]<imagename>[:tag]

**5. Tag an image**

$ docker tag jboss/wildfly myimage:v1

# Creates an image called "myimage" with the tag "v1" for the image jboss/wildfly:latest

$ docker tag <imagename> <newimagename>

# Creates a new image with the latest tag

$ docker tag <imagename>[:tag] [username/]<newimagename>[:newtag]

# Creates a new image specifying the "new tag" from an existing image and tag.

**6. Exporting and Importing and image to an external file**

$ docker save o <filename>.tar [username/]<imagename>[:tag]

# Export the image to an external file

$ docker load i <filename>.tar

# Import an image from an external file

**7. Push an image to a registry.**

$ docker tag <imagename>[:tag] [username/]<newimagename>[:newtag]

# Creates a new image specifying the "new tag" from an existing image and tag.

**8. Exporting and Importing and image to an external file**

$ docker save o <filename>.tar [username/]<imagename>[:tag]

# Export the image to an external file

$ docker load i <filename>.ta

# Import an image from an external file

**9. Push an image to a registry.**

$ docker push [registry/][username/]<imagename>[:tag]

**Docker Cheat Sheet**

|  |  |
| --- | --- |
| build | Build Docker images from a Dockerfile |
| history | Show the history of an image |
| images | List images |
| import | Create an empty filesystem image and import the contents of the tarball into it |
| inspect | Return lowlevel information on a container or image |
| load | Load an image from a ‘.tar’ archive or STDIN |
| pull | Pull an image or a repository from the registry |
| push | Push an image or a repository to the registry |
| rmi | Removeone or more images |
| save | Save one or more images to a ‘.tar’ archive (streamed to STDOUT by default) |
| search | Search the Docker registry for images |
| tag | Tag an image into a repository |

**Network related commands**

|  |  |
| --- | --- |
| connect | Connects a container to a network |
| create | Creates a new network with the specified name |
| disconnect | Disconnects a container from a network |
| inspect | Displays detailed information about on a network |
| ls | Lists all the networks created by the user |
| rm | Deletes one or more networks |

**Registry related commands**

|  |  |
| --- | --- |
| login | Log in to a Docker registry server. If no server is specified, then the default is used |
| logout | Log out from a Docker registry server. If no server is specified then the default is used. |

**Volume related commands**

|  |  |
| --- | --- |
| create | Create a volume |
| inspect | Return lowlevel information on a volume |
| ls | List volumes |
| rm | Remove a volume |

**Related commands**

|  |  |
| --- | --- |
| docker events | Get realtime information from the server |
| docker info | Display systemwide information |
| docker version | Show the docker version information |
| systemctl status docker | Check if the docker service is running |

**Dockerfile**

The Dockerfile provides the instructions to build a container image through the `docker build t [username/]<imagename>[:tag] <dockerfilepath>` command. It starts from a previous existing Base image (through the FROM clause) followed by any other needed Dockerfile instructions.

This process is very similar to a compilation of a source code into a binary output, but in this case the output of the Dockerfile will be a container image.

**Example Dockerfile**

#Use the existing WildFly image

FROM jboss/wildfly

# Add an administrative user

RUN /opt/jboss/wildfly/bin/adduser.sh admin Admin

#70365 silent

#Expose the Administrative port

EXPOSE 8080 9990

# Bind the WildFly management to all IP addresses

CMD ["/opt/jboss/wildfly/bin/standalone.sh", "b", "0.0.0.0", "bmanagement", "0.0.0.0"]

**Using the example Dockerfile**

# Build the WildFly image

$ docker build t mywildfly .

# Run a WidFly server

$ docker run it p 8080:8080 p 9990:9990 mywildfly

# Access the WildFly administrative console and log in with the credentials admin/Admin#70365

open http://<dockerdaemonip>:9990 in a browser

**Dockerfile INSTRUCTION arguments**

|  |  |
| --- | --- |
| FROM | Sets the Base image for subsequent instructions |
| MAINTAINER | Sets the author field of the generated images |
| RUN | Executes commands in a new layer on top of the current image and commits the results |
| CMD | Allowed only once (if many, then only the last one takes effect) |
| LABEL | Adds metadata to an image |
| EXPOSE | Informs Docker that the container listens on the specified network ports at runtime. |
| ENV | Sets an environment variable |
| ADD | Copies new files, directories, or remote file URLs into the filesystem of the container |
| COPY | Copies new files or directories into the filesystem of the container |
| ENTRYPOINT | Allows you to configure a container that will run as an executable |
| VOLUME | Creates a mount point and marks it as holding externally mounted volumes from the native host or other containers |
| USER | Sets the username or UID to use when running an image |
| WORKDIR | Sets the working directory for any RUN, CMD, ENTRYPOINT, COPY, and ADD commands |
| ARG | Defines a variable that users can pass at build time to the builder using --build-arg |
| ONBUILD | Adds an instruction to be executed later, when the image is used as the base for another build |