QA Consulting

Docker Container

TRAINING MATERIALS - MODULE HANDOUT

Contacts

robert.crutchley@qa.com
team.qac.all.trainers@qa.com

www.consulting.qa.com

Contents

Overview 1

Container Properties 1

- Container ID 1
- Image 2
- Command 2
- Ports 2
- Names 2

Managing Containers 2

- View Running Containers 2
- View All Containers 2
- Show Only the IDs 2
- Run a Container 2
- Detach the Container Logs 3
- Publish a Port 3
- Set the name of a container 3
- Execute a Command on a Running Container 3
- Interactive Shell on a Container 3
- Stop a Container 4
- Remove a Container 4
- Start a Container 4
- Rename a Container 4
- Copy Files Between the Host and the Container 4
- View Container Logs 5
- Stopping and Removing all Containers (in bash) 5

Tasks 5

Overview

Containers are what our applications are going to be run in to isolate them from the host and on a network level. This handout discusses common properties for containers and how to manage them.

Container Properties

Container ID

The unique identifier for the container, which great for referencing specific containers directly.

Image

Which Docker image is being run in the container

Command

Every container needs to have a main process running, the command property shows which script, command or executable was run to create this main process.

Ports

The ports that have been published or exposed. Unlike published ports, exposed ports don't alter anything as far as the network is concerned, just a form of documentation to understand which port the service running in the container is listening on. Publishing ports exposes the service so it can be reached from outside of the container.

Names

Name of the container, this is a property that can be set to something more memorable or easy to work with. By default, Docker will assign a generated name if one is not provided.

Managing Containers

View Running Containers

The **ps** command can give us an overview of the running containers.

ps

```
docker ps
```

View All Containers

Only the containers that are running are shown with the **ps** command alone, we can view all of the containers, even the stopped ones as well by passing the **-a** option.

-a

```
docker ps -a
```

Show Only the IDs

If we want to reference multiple containers, the IDs can be displayed with the **-q** (quiet) option

-q

```
docker ps -q
```

Docker Containers

Run a Container

Containers can be created from images by using the **run** command.

docker run <image>:<tag>

```
docker run nginx
```

Detach the Container Logs

In many cases we don't want the container logs being outputted once a run command has been issued, mainly because we can't access the terminal when they do.

-d

```
docker run -d nginx
```

Publish a Port

Publishing a port maps a port inside of the container to a port on the host so that it is accessible from outside of the Docker engine

-p <host-port>:<container-port>

```
docker run -d -p 80:80 nginx
```

Set the name of a container

To avoid Docker generating a name for you, a name can be specified. Giving a custom name for a container can make it much easier to reference in other commands.

--name <container-name>

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

Execute a Command on a Running Container

Executing commands on a container can be useful for completing simple tasks, anything that is a single, short command.

exec <container-name> <command>

```
docker exec nginx echo hi
```

Interactive Shell on a Container

For more in depth debugging or for development, gaining shell access to a container is very useful. The executable that you will want to run on the container will need to be a shell interpreter, most of the time this is **bash**. However some containers do not have bash installed so you will have to use shell instead (**sh**).

exec -it <container-name> bash

```
docker exec -it nginx bash
```

Stop a Container

Containers can be in a stopped or running state

stop <container-name>

```
docker stop nginx
```

Remove a Container

Once a container has been stopped it still exists, it's good housekeeping to remove the containers that you no longer need. A container needs to be stopped before it can be removed

rm <container-name>

```
docker rm nginx
```

Start a Container

Keep in mind if the main process for the container fails then it will automatically stop, no matter how many times you keep starting it.

start <container-name>

```
docker start nginx
```

Rename a Container

If you made a typo or simply forgot to provide a name for you container then you can still rename it

rename <old-container-name> <new-container-name>

```
docker rename naughty_dubinsky nginx
```

Copy Files Between the Host and the Container

When developing and debugging being able to copy files between the container and host can be very useful. For example copying a log file from the container, or copying configurations into the container.

Copy from the host:

cp <host-location> <container-name>:<container-location>

```
docker cp /etc/passwd nginx:/tmp/test
```

Copy from the container:

cp <container-name>:<container-location> <host-location>

```
docker cp nginx:/etc/passwd /tmp/test
```

View Container Logs

To get a better understanding why a service is being unresponsive or a container is failing you can view the logs

logs <container-name>

```
docker logs nginx
```

Stopping and Removing all Containers (in bash)

Sometimes we need start from a clean slate with all the containers that are running, we can use command substitution in bash for this:

Stop all Containers

```
docker stop $(docker ps -qa)
```

Remove all Containers

```
docker rm $(docker ps -qa)
```

Tasks

- Run a Jenkins container
- Name the Jenkins container: "**jenkins**"
- View the logs of the Jenkins container
- Show the contents of the **/etc/passwd** file in the Jenkins container
- Copy the **/etc/passwd** from the nginx container to anywhere on your machine
- Run five NGINX containers
- Stop all of the containers with one command
- Remove all of the containers with one command