When we get error for Docker installation on Amazon EC2 (Redhat instance)

Error: Package: docker-ce-17.06.0.ce-1.el7.centos.x86\_64 (docker-ce-stable)

Requires: container-selinux >= 2.9

You could try using --skip-broken to work around the problem

You could try running: rpm -Va --nofiles --nodigest

The problem comes due to the selinux container package. It is required to resolve this.

Installing the Selinux from the Centos repository worked :

1. Go to http://mirror.centos.org/centos/7/extras/x86\_64/Packages/ check for latest package

2. Find the latest version for container-selinux i.e. container-selinux-2.21-1.el7.noarch.rpm

3. sudo yum install -y http://mirror.centos.org/centos/7/extras/x86\_64/Packages/container-selinux-2.21-1.el7.noarch.rpm

Note: the container version is constantly being updated, that is why you should look for the latest version in the Centos' repository

This resolves the issue.Docker gets installed.

For docker installation refer: https://docs.docker.com/cs-engine/1.12/

Package Links: https://docs.docker.com/cs-engine/1.13/#general-commands

===================================================================================================

Process based Architecture is docker. Only required process will run and for rest it uses hardware of parent machine.

Commands to install Docker:

1. yum install docker (On GCB Docker gets installed with this one command, there occurs no issue as in AWS platform)

2. systemctl start docker.service

3. systemctl enable docker.service

4. systemctl status docker.service

5. docker (command with help options for docker)

6. docker info (Display system-wide information)

7. docker -v (Displays the docker version)

8. docker search (Search the Docker Hub for images)

9. docker images (Displays no of docker images downloaded)

10.docker ps (Displays running container information)

11.docker ps -a (Displays all containers info running as well as exited)

12.docker pull (Pulls an image from docker hub)

13.docker run -itd <image\_name> /bin/bash

14.docker run -itd -p 80:80 --name <New\_name> <image\_name> /bin/bash (used to start a container with customized name and also opens port 80:80 for both container and local machine)

15.docker attach <container-id> (attaches/moves in to the container)

16.docker exec -it <container-id> /bin/bash (To enter into a running container)

17.docker commit <container-id> <NewImage\_name> (We can create a new customized image from a container’s changes)

18.docker cp /tmp/index.html Apache\_server:/var/www/html (Used to copy file from local machine to docker container)

19.docker stop $(docker ps -a -q) (Will stop all the running containers in one go)

20.docker rm $(docker ps -a -q) (Will remove all the containers in one go)

21.docker inspect <container\_id> (Gives all the details of the container)

22.docker run --help (List of options available in docker for ex to mount a volume, allocate RAM CPU to docker containers, etc)

23. docker diff <container\_Id> (Inspect changes to files or directories on a container’s filesystem)

24. docker events –since ‘2019-01-24’ (Get real time events from the server –-since

|  |
| --- |
| shows all events created since timestamp on the host machine  25. docker export red\_panda > latest.tar (Export a container’s filesystem as a tar archive)  26 docker history <container\_Id> (Show the history of an image)  27.docker stats -a (Display a live stream of container(s) resource usage statistics) |
|  |

===================================================================================================

Key Points:

When a container is started using /bin/bash then it becomes the containers PID 1 and docker attach is used to get inside PID 1 of a container. So docker attach < container-id > will take you inside the bash terminal as it's PID 1 as we mentioned while starting the container. Exiting out from the container will stop the container.

Whereas in docker exec command you can specify which shell you want to enter into. It will not take you to PID 1 of the container. It will create a new process for bash. docker exec -it < container-id > bash. Exiting out from the container will not stop the container.

Wordpress details

Site title: ServerWorld

Username : wordpress

Password: Hx%6nLmMQXs\*&v^Rbc

Docker Compose

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application’s services. Then, with a single command, you create and start all the services from your configuration.

Firstly we need to install download the Docker Compose binary from the [Compose repository release page on GitHub](https://github.com/docker/compose/releases).

1. Run this command to download the latest version of Docker Compose:

sudo curl -L "https://github.com/docker/compose/releases/download/1.23.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

1. Apply executable permissions to the binary:

sudo chmod +x /usr/local/bin/docker-compose

1. docker-compose --version

Docker compose version command gives us the version of the installed docker version.It indicates that compose is installed successfully.

Wordpress Installation using Docker compose

1.mkdir my\_wordpress (create a project directory)

2. cd /mywordpress

3. Create a docker-compose.yml file that starts your WordPress blog and a separate MySQL instance with a volume mount for data persistence:

1. vi docker-compose.yml

version: '3.3'

services:

db:

image: mysql:5.7

volumes:

- db\_data:/var/lib/mysql

restart: always

environment:

MYSQL\_ROOT\_PASSWORD: somewordpress

MYSQL\_DATABASE: wordpress

MYSQL\_USER: wordpress

MYSQL\_PASSWORD: wordpress

wordpress:

depends\_on:

- db

image: wordpress:latest

ports:

- "8000:80"

restart: always

environment:

WORDPRESS\_DB\_HOST: db:3306

WORDPRESS\_DB\_USER: wordpress

WORDPRESS\_DB\_PASSWORD: wordpress

volumes:

db\_data: {}

We can use either a .yml or .yaml extension for this file. They both work.

1. docker-compose up -d (This runs [docker-compose up](https://docs.docker.com/compose/reference/up/) in detached mode, pulls the needed Docker images, and starts the wordpress and database containers.)
2. WordPress Multisite works only on ports 80 and/or 443
3. We need to allow firewall rules if we use GCB for Docker compose.
4. <http://MACHINE_VM_IP:8000> (Gives dashboard for wordpress installation)
5. Link which defines all the parameters and explanation for writing a docker compose file. <https://www.linode.com/docs/applications/containers/how-to-use-docker-compose/>
6. docker-compose stop (This stops your containers, but it won’t remove them)
7. docker-compose down (This command will stop your containers, also removes the stopped containers as well as any networks that were created)
8. docker-compose down -v (This command removes all containers as well as the volumes associated)

Important links for docker study:

<https://docs.docker.com/get-started/part2/> (All aspects explained on digital oceans)

<https://www.linode.com/docs/applications/containers/how-to-use-docker-compose/> ---

Details for docker compose and containts of the compose file.

<https://rominirani.com/docker-tutorial-series-writing-a-dockerfile-ce5746617cd>

https://www.tecmint.com/build-and-configure-docker-container-images-with-dockerfile

(Describes how to write a docker file)

## Why containerized application setup is required--- Interview Question

The explanation with all the details is best explained on below given link:

<https://www.cio.com/article/2924995/what-are-containers-and-why-do-you-need-them.html>