Ensuring Uptime

Step 1: Stop On Fail

Docker considers any containers to exit with a non-zero exit code to have crashed. By default a crashed container will remain stopped.

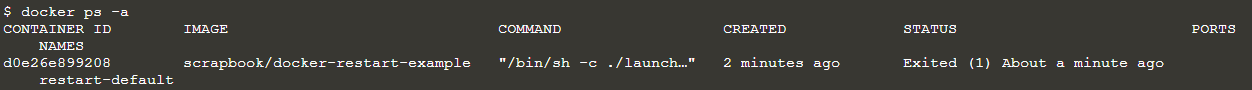
Example:

We've created a special container which outputs a message and then exits with code 1 to simulate a crash.

You can launch an instance using docker run -d --name restart-default scrapbook/docker-restart-example



If you list all the containers, including stopped, you will see the container has crashed docker ps -a



While the logs will output our message, which in real-life would hopefully indicate an issue.

docker logs restart-default



Step 2:

Restart On Fail

Depending on your scenario, restarting a failed process might correct the problem. Docker can automatically retry to launch the Docker a specific number of times before it stops trying.

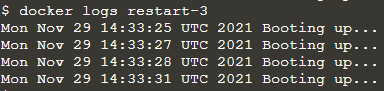
Example: The option *--restart=on-failure:#* allows you to say how many times Docker should try again. In the example below, Docker will restart the container three times before stopping.

docker run -d --name restart-3 --restart=on-failure:3 scrapbook/docker-restart-example



As we can see from the logs, it was launched on three occasions.

docker logs restart-3

docker logs restart-3

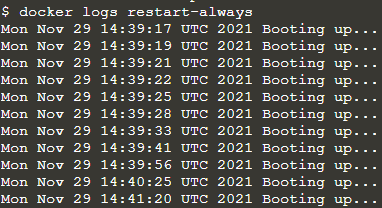
Step 3: Always Restart:

Finally Docker can always restart a failed container, in this case, Docker will keep trying until the container is explicitly told to stop.

Example: Use the *always* flag to automatically restart the container when is crashes for example

docker run -d --name restart-always --restart=always scrapbook/docker-restart-example

You can view the restart attempting via the log

Y

### **ge, which in real-life would hopefully indicate information to help us diagnose the issue.While the logs will output our message, which in real-life would hopefully indicate information to help us diagnose the issue.**

### 

### 

### **Ensuring Uptime**