Issue

If a container is no longer running, use the following command to find the status of the container:

```
docker container ls -a
```

This article explains possible reasons for the following exit code:

```
"task: non-zero exit (137)"
```

With exit code 137, you might also notice a status of `Shutdown` or the following failed message:

```
Failed 42 hours ago
```

Resolution

The `"task: non-zero exit (137)"` message is effectively the result of a `kill -9` (`128 + 9`). This can be due to a couple possibilities (seen most often with Java applications):

1. The container received a `docker stop`, and the application didn't gracefully handle `SIGTERM` (`kill -15`) — whenever a `SIGTERM` has been issued, the docker daemon waits 10 seconds then issue a `SIGKILL` (`kill -9`) to guarantee the shutdown.
   To test whether your containerized application correctly handles `SIGTERM`, simply issue a `docker stop` against the container ID and check to see whether you get the `"task: non-zero exit (137)"`.
   This is not something to test in a production environment, as you can expect at least a brief interruption of service. Best practices would be to test in a development or test Docker environment.

2. The application hit an OOM (out of memory) condition.
   With regards to OOM condition handling, review the node's kernel logs to validate whether this occurred. This would require knowing which node the failed container was running on, or proceed with checking all nodes.
   Run something like this on your node(s) to help you identify whether you've had a container hit an OOM condition:

   ```
   journalctl -k | grep -i -e memory -e oom
   ```

   Another option would be to inspect the (failed) container:

   ```
   docker inspect <container ID>
   ```

   Review the application's memory requirements and ensure that the container it's running in has sufficient memory. Conversely, set a limit on the container's memory to ensure that wherever it runs, it does not consume memory to the detriment of the node.

   If the application is Java-based, you may want to review the maximum memory configuration settings.

References

- `docker run` command line options (https://docs.docker.com/engine/reference/commandline/run/#options)
- Specify hard limits on memory available to containers (https://success.docker.com/api/asset/.%2Fwhat-

causes-a-container-to-exit-with-code-137%2F-m, –memory)
(https://docs.docker.com/engine/reference/commandline/run/#specify-hard-limits-on-memory-available-to-containers--m-memory)