A Report on Docker containers debloating via DockerSlim

Amir Andalib

SEP 2021

DockerSlim is a free tool that analyzes a container and tries to debloate it by removing unnecessary resources such as files, software packages, etc. In this report, a web application which is composed using 14 containers, is analyzed and slimmed using DockerSlim. In the following, a brief about the procedure and results is discussed.

1 Microservice : Socks

This service needs 14 containers to run. Some of them are databases that use "MySQL" or "MongoDB". For every image in the docker-compose file "docker-slim" is run to build a smaller version of the image. In the following sub-sections, every container and its slimmed version will be compared in detail, nevertheless, some general challenges and findings are listed below:

- As it was mentioned in the DockerSlim documents[1], some of the web applications written in languages such as Python and Ruby, require service interaction to load everything in the application. Therefore, for our application, we need to have the "HTTP-probe" enabled for almost all of the containers.
- On some containers running "HTTP-probe" is failed with the error "connection reset by peer". Due to some research on the Internet, this error can be caused by several issues such as:
  - The webserver is bound to "127.0.0.1", so it will be available only inside the container[2]. The general solution for this issue is to bind the webserver to "0.0.0.0", so it can be accessed from outside of the container. But it is not the case here, because when we run the service via Docker-Compose or run every container individually, the mapped address for the ports are accessible from outside, and additionally the whole service works fine. However, the configuration files of the webservers have been checked individually.

[1] https://github.com/docker-slim/dockerslim#running-containerized

[2] https://nickjanetakis.com/blog/docker-tip-54-fixing-connection-reset-by-peer-or-similar-errors

  - It also can occur when there is no application on the exposed port [3]. This was also verified by checking the used ports by the application inside the container.
  - Firewall can be another cause for this problem. It was disabled on the machine running DockerSlim. Since I use my machine for other works, I thought, the problem might stem from some other misconfiguration that I have already done in the operating system. Hence, I installed new Ubuntu from scratch on my machine and one other machine and tried the docker-slim there. But still, I was getting the same error.
  - This error seems to arise from the inside of the container because the "HTTP-probe" is successfully done for some other containers which also has web services running inside such as "MongoDB", "weavedemosfrontend."

- Applied solution for the "connection reset by peer" error: One way to deal with this error is to disable HTTP-probe. However, disabling it ends in a container that has literally nothing in it, and obviously, it fails to run the needed services. When the "build" command runs in Docker-Slim to make the slimmed version of an image, it executes an interactive container of the image and, the build procedure pauses and lets the user interact with the container. In this stage, the user can start needed services inside of the container, so the slimmed image will contain that service. There is an "include" option in running the "build" command which can tell the DockerSlim to not remove a special file or path. For example, "weaveworksdemos/catalogue-db" (400 MB), is one of the containers which does not allow "HTTP-probe". When DockerSlim builds a slim version of it in "HTTP-probe-off" mode, the slim docker image size would be 34 MB. When we use the slimmed version only for this container while other are the original ones in the compose file, there is no catalogs of socks in the web application. In the interactive mode when the "MySql" service is started, the slimmed container size would be 52 MB. On the other hand, in some containers such as "weaveworksdemos/orders", there is no bash to interact with.

1.1 weaveworksdemos/front-end

This container is minified with no error, the "http-probe" is accomplished successfully. According to the "xray" of the original and minified containers, the followings can be considered:

- The "Ash Shell" and "Bourne Shell" has been removed. Hence, no "exec" interaction with "/bin/bash/" is available in slim container.
- The tmp directory from the path "/var/tmp" has been removed

[3] https://serverfault.com/questions/769578/curl-56-recv-failure-connection-reset-by-peer-when-hitting-docker-container

- All includes such as "../include/node/v8.h", "..opensslsafestack.h", "opensslobj_mac.h", "openssl/ssl.h" has been removed.
- User folder : "/home/myuser" has been removed which apparently contains a cache folder
- The only problem with slimmed front-end container is the login-register bar. I think a mistakly removed .js or a node.js script is responsible for that.

Unfortunately, I had a busy days at work, besides, I spend more that 90% of my time trying to figure out the "connection reset by peer" problem. Therefore, I had a limited time to find the exact path for the script that is responsilbe for login bar and, exclude it from being processed by build procedure.

1.2 weaveworksdemos/edge-router

The edge-router can get minified without error with "http-probe" enabled, and no changes in the application performance was detected. The results of comparing slim and original containers are:

- Like front-end container all bashes has been removed. In addition, it deleted all of the "mozila" certeficate links and hashes
- Apparently in slim mode DockerSlim removes all of the image layers and bring needed binaries to a single layer
- "libcrypto", "libssl", and certificate file has been removed. It may result in some security issues

1.3 weaveworksdemos/catalogue

The "catalogue" is one of those container which makes "connection reset by peer" error. All aformentioned efforts about this error has been applied to this container except it has no bashes, so I could not interact with it and cannot see if there are any critical paths or services that can be saved. However, apparently the application website operates with no error when the slimmed version is replaced in the docker-compose file. The following items can be concluded from comparing x-rays :

- Like others the temp folder on /var/temp and all shells has been removed
- Many libraries and binaries such as libcrypto and getcaps has been removed. I do not know what can removing the ssl and libcrypto libraries can do when program runs in large scale. But the binaries such as getcap which indicates capabilities on pids are allowed to be removed, because they are not mandetory

1.4 catalogue-db - MySQL

The catalogue-db container uses Mysql, this container also has the ”connection reset by peer” problem. Although in the interaction mode the Mysql service was started, the container would not operate properly in the compose set. When all other containers set to the original images and only the catalogue-db.slim is used, the catalogues (socks in the website) do not appear in the application. The following items can be concluded from comparing x-rays :

- The paths related to MySQL exist in the slimmed container.
- Some commands contains MySQL such as adding new user and new group user has been removed.
- The directory which includes database data can be excluded from being removed, however, the service has to be run properly in the slimmed container first

1.5 mongoDB

Two databases, cart-db and order-db use mongoDB. The mongoDB image is successfully built by DockerSlim using http-probe. It operates in the application without error. The following items can be concluded from comparing x-rays :

- The main container has 3 shells, and it has been removed in slim mode.
- Bundles, links, hashes and private link keys of certificates has been removed.
- The original image has 13 layers which like other minified images there is on in slimmed one

1.6 Running Slimmed Containers

First, all containers in the compose file are replaced with their slimmed version. Then, Docker-compose starts the service successfully. Nonetheless, first page of the application does not load anything but contact information. After this, the original compose file is used and the slimmed versions of each container is added one by one and the result is verified. After some trial and error, a combination of some slimmed and original containers are finally used to run the service. Table 1 shows what containers used in slimmed mode and how its build procedure is done.

Figure 1 and 2 show memory and cpu usage of the containers in original and slim(some of containers are in original debloated mode) mode which are generated by (weavedemos_scope). Table 2 show the summary of the memory usage in both mode. Although the debloated containers have smaller sizes (with questionable performance), they do not use less memory.

Table 1: Summary of Final Docker-Compose File

container	Used Version	Description
front-end	Original	The slimmed version was built successfully but failed to operate properly
edge-route	Slim	HTTP-probe Enabled
catalogue	Slim	HTTP-probe Disabled, no interaction available
catalogue-db	Original	MySQL is started in interactive mode but still no catalogs in the website
orders	Original	No HTTP-probe, Not interactive, Cannot proceed to shipping
shipping	Original	HTTP-probe disabled, Not interactive, cannot set credit card
carts	Original	No HTTP-probe, The cart button disappears
carts-db (MongoDB)	Slim	It uses mongoDB slimmed version, HTTP-probe enabled
shipping	original	HTTP-probe error, not interactive, in the slimmed version the credit card information cannot be assigned
queue-master	Slim	HTTP-probe disabled, Since it may control queue in corporation with rabbitMQ, it may show its mal-operation in heavy load
rabbitMQ	Slim	HTTP-probe disabled, not interactive
payment	Slim	HTTP-probe disabled, Not interactive
user	Slim	HTTP-probe disabled, Not interactive
user-db	Original	Successfully build with HTTP-probe enabled, User register cannot be down in slim mode

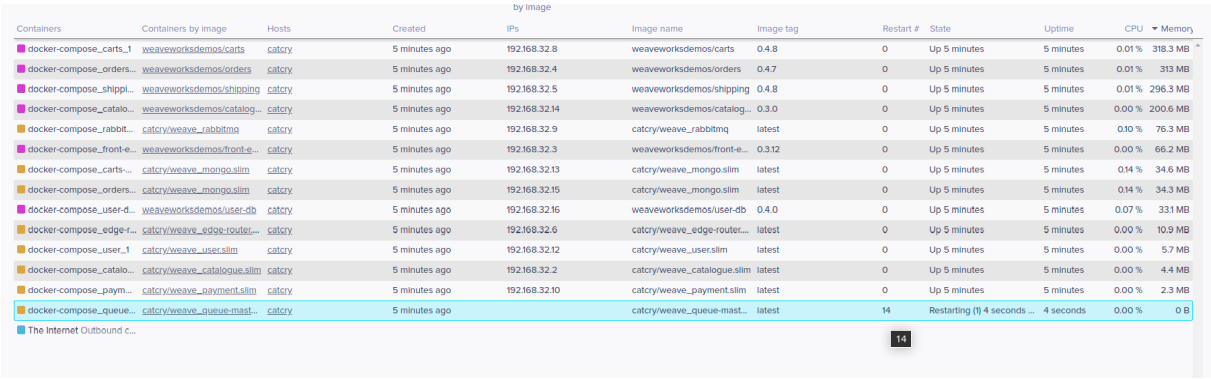

Figure 1: Memory and CPU usage of containers in debloated mode

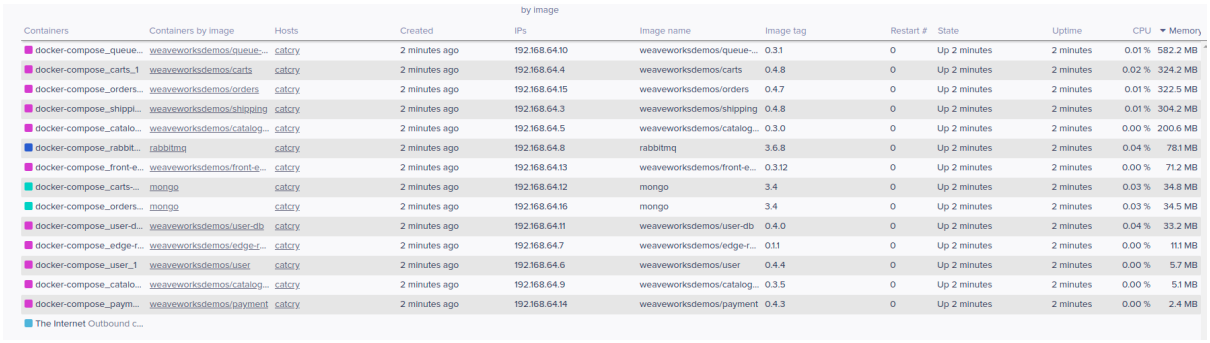

Figure 2: Memory and CPU usage of containers in bloated mode

Table 2: Summary of Final Docker-Compose File

container	Original Size(MB)	Slim Size (MB)	Original Mem(MB)	Slim Mem(MB)
front-end	120	27.8		
edge-route	21.9	16.7	11	10.9
catalogue	21.9	14.3	5.1	4.4
catalogue-db	400	52.6		
orders	198	36.9		
shipping	199	37.4		
carts	198	36.9		
carts-db (MongoDB)	685	66	34.5	34.6
shipping	199	37.4		
queue-master	179	43.6	528	0
rabbitMQ	179	32.1	78	76
payment	32.5	13.4	2.4	2.3
user	35.3	14.8	5.2	5.7
user-db	717	65.1		

2 Conclusion

To sum it up, DockerSlim should be used very carefully by manually adjusting some advanced option. Even when everything goes fine and the ”build” procedure accomplish its job successfully, there is a chance that the container does not work properly like front-end and user-db containers. This tool can provide some useful information about the container via its ”x-ray” tool. And, I think using this tool and manually removing one part at time from container w.r.t the x-ray results is more secure way to slim a container.

General Comments:

- Sometime it need 2 or 3 times refresh to load the first page properly
- Some username and password combinations are not valid

The ”docker-compose.yml” which pulls and runs the debloated application is provided in the report email. Also, all of the debloated containers are uploaded to docker hub: https://hub.docker.com/u/catcry