SERENA®

DIMENSIONS® CM 14.3.3

Docker Registry Guide

Serena Proprietary and Confidential Information

Publication date: June 2017

Table of Contents

Chapter 1

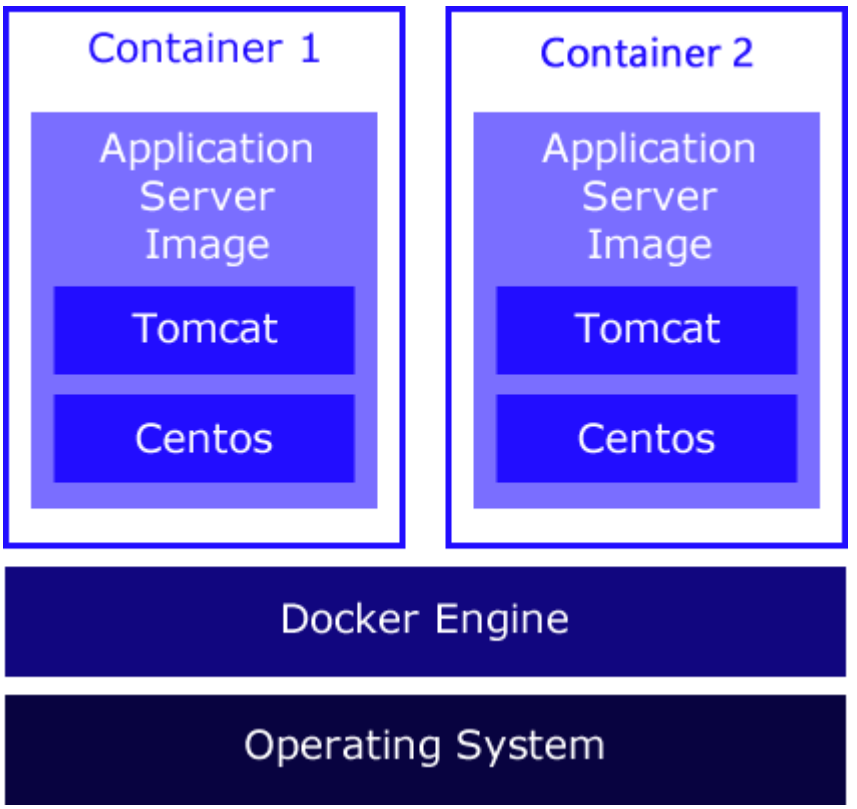
Introduction

What is Docker?

Docker enables you to create lightweight environments, called *containers*, in which you can run applications (Docker images). Containers run as isolated processes on a host operating system and are not tied to any specific infrastructure. Docker runs on any machine, on any infrastructure, and on any cloud. Images are stored in a Docker Registry.

An image's environment, such as an application and the dependencies required to run it, are stored in layers. Containers share the host operating system kernel with other containers. Containers start and stop instantly and make more efficient use of RAM. You can run multiple containers of the same image.

In the example below there are two containers, each running the same application server image comprised of two layers, Tomcat and Centos. The containers are running on the same Docker Engine and share the host operating system.



Docker is a commercial company, for information see this web site:

https://www.docker.com/what-docker

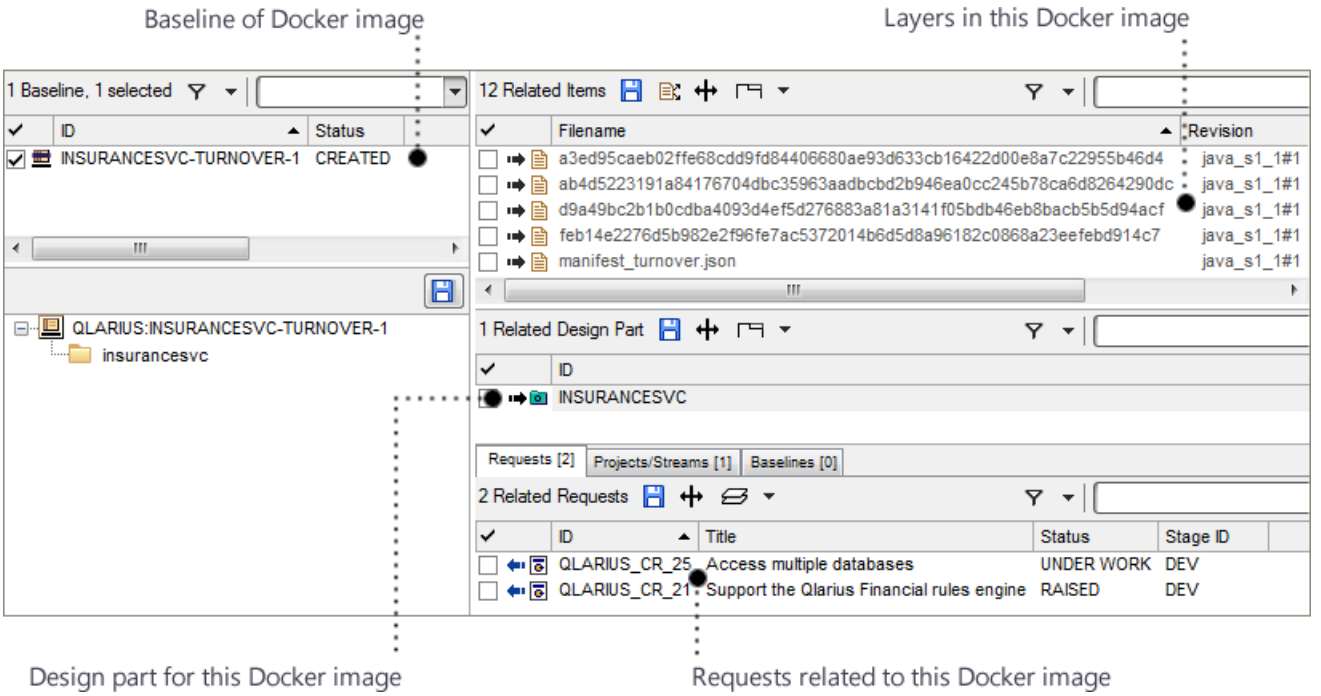
Dimensions CM Integration with Docker

The Dimensions CM integration provides a Docker Private Registry for your images and has the following key features:

- Stores and version controls images. Changes to images over time are recorded so that you can recall specific versions later.
- Provides an audit trail of all changes to each image.
- Uses CM privileges and roles assignments to control which images can be pulled (downloaded). An image is only visible to the user who created it but is available to other users *after* it has been approved. For details about privileges, roles assignments, and relationships see the *Process Configuration Guide*.
- Enables change control of Docker images using Dimensions CM requests.

Dimensions CM Docker Registry images are stored in a secure CM repository:

- Each time an image is pushed (delivered) to CM a new baseline is created.
- Each image is a new design part.
- Each layer in an image is a new item with a unique filename.
- Layers can be shared between images using CM's usage relationships.



Installing and Licensing

The integration with Docker is installed and licensed separately from Dimensions CM. For details contact Serena Software.

Chapter 2

Starting Docker

Starting a Docker Registry

NOTE If you are using Docker version 10, login to the registry using the Docker `login` command before performing any other registry operations.

To start a Dimensions CM Docker Registry run this command (parameters in square brackets are optional):

```
java -jar docker-registry.jar
--port=<port number>
--cm-url=scm:dimensions://<cm server>/<database>
--cm-product=<product>
--cm-stream=<product>:<stream>
--cm-part="<product>:<design part>"
[--cm-part-type=<type>]
[--cm-baseline-type=<type>]
[--cm-baseline-state=<state>]
[--cm-create-parts=<true or false>]
[--cm-cache=<true or false>]
[--cm-cache-dir=<cache directory>]
[--cm-cache-size=<maximum cache size>]
[--work=<path>]
[--log-file=<path>]
[--manage-path=<path>]
[--manage-user=<user>]
[--manage-password=<password>]
[--ssl=false]
```

where:

- `--port=<port number>`

  Specifies the port number of a Docker Registry.

  Default: 5000

- `--cm-url=scm:dimensions://<cm server>/<database>`

  Specifies a CM server and database, for example:

  `--cm-url=scm:dimensions://mycmserver/cm_typical@dim14`

- `--cm-product=<product>`

  Specifies a CM product for Docker Registry images, for example:

  `--cm-product=QLARIUS`

- `--cm-stream=<product>:<stream>`

  Specifies a CM stream where new images are delivered, for example:

  `--cm-stream=QLARIUS:DOCKER_STREAM`

- `--cm-part="<product>:<design part>"`

  Specifies a design part that will own pushed images, for example:

  `--cm-part="QLARIUS:DOCKER_IMAGES.A;1"`

- `--cm-part-type`

  Specifies the design part type for images that you push. Default: `SUB-SYSTEM`

- `--cm-baseline-type=<type>`

  Each time an image is pushed (delivered) to CM a new baseline is created. Use this parameter to specify a type, for example:

  `--cm-baseline-type=MYTYPE`

  Default: `BASELINE`

- `--cm-baseline-state=<state>`

  Specifies the 'approved' state for baselines. Only images in 'approved' baselines can be searched and pulled. For example:

  `--cm-baseline-state=APPROVED`

  Default: `APPROVED`

- `--cm-create-parts=<true or false>`

  By default, the registry automatically maps each image to a new, unique design part. Alternatively, you can use upload rules to identify the owning design part for each image.

- `--cm-cache=<true or false>`

  You can cache the image layers inside the registry to avoid fetching the same content from the server to the registry on every pull operation. Serena recommends setting this parameter if your registry is remote from your Dimensions CM server.

- `--cm-cache-dir=<cache directory>`

  (Only valid if you specify `cm-cache=true`) Specifies the directory where the cache will be stored.

- `--cm-cache-size=<maximum cache size>`

  (Only valid if you specify `cm-cache=true`) Specifies the maximum size of the cache in megabytes.

- `--work=<path>`

  By default Dimensions CM Docker Registry uses the following path to store temporary files when images are pushed and pulled:

  Windows: `C:\ProgramData\Serena\Docker\_work`

  UNIX: `/opt/docker-registry/work`

  Use this parameter to specify a different location, for example:

  `--work=/tmp/docker-registry`

- `--log-file=<path>`

  By default Dimensions CM Docker Registry uses the following path to store log files:

  Windows: `C:\ProgramData\Serena\Docker\_logs\docker-registry.log`

  UNIX: `/opt/docker-registry/logs/docker-registry.log`

  Use this parameter to specify a different location, for example:

  `--log-file=/tmp/Serena/Docker/_logs/docker-registry.log`

- `--manage-path=<path>`

  `--manage-user=<user>`

  `--manage-password=<password>`

  By default Dimensions CM Docker Registry uses `/system` as the prefix for all diagnostic endpoints. The default user is `admin` and the default password is `secret`. You can specify your own path prefix and security credentials, for example:

  ```
  --manage-path=/manager

  --manage-user=superuser

  --manage-password=password
  ```

- `--ssl=false`

  Starts Dimensions CM Docker Registry without Secure Socket Layer (SSL) protection.

Using a Docker Registry without SSL

Docker prefers registries to be protected with SSL. If you have a non-SSL registry you must tell Docker to make an exception and allow access.

1. Use Secure Shell (SSH) to log into the system running your Docker Engine. If you are using the Docker Toolbox on Windows this is a Virtual Image running under VirtualBox. You will need the IP address of that system to SSH into it. On Linux, SSH into the Linux system itself. If you are using the `boot2docker` image you can login as:

   - Username: `docker`
   - Password: `tcuser`

2. Edit the Docker profile and specify that the registry is insecure:

   ```
   cd /var/lib/boot2docker
   sudo sh
   vi profile
   ```

   At the top of this file `EXTRA_ARGS` is defined. Add this line:

   ```
   --insecure-registry=myregistryserver:5000
   ```

3. Change the host name and port number as needed and save the file.

4. Restart Docker:

   ```
   /etc/init.d/docker restart
   ```

5. Log into a Dimensions CM Docker Registry:

   ```
   docker login <myregistryserver>:<port number>
   ```

6. Enter Dimensions CM login credentials. If the login is successful you have configured Docker to use the non-SSL registry.

Protecting a Docker Registry with SSL

This section is an example of how to SSL protect a Dimensions CM Docker Registry using a self-signed certificate.

Creating a Certificate Authority

1. Use SSH to log into the `boot2docker` image (or the Linux system) that hosts your Docker engine.
2. Run `sudo sh` to change to a root shell.
3. Generate a Certificate Authority (CA) certificate and key pair (you can optionally change the password and subject):

```
cd /usr
mkdir ca_certs
chmod 700 ca_certs
cd ca_certs
mkdir certs private newcerts
echo 1000 > serial
touch index.txt
openssl req -new -x509 -days 3650 -extensions v3_ca \
   -keyout private/cakey.pem -out cacert.pem \
   -config /etc/ssl/openssl.cnf -passout pass:passwordformyca -subj \
   "/C=UK/ST=Surrey/L=Richmond/O=Acme/OU=Development/CN=Acme"
```

Generating Certificates and Keys

Generate a certificate/key pair for the machine that will run your Dimensions CM Docker Registry and sign it with the CA you created in the previous step.

1. Use SSH to log into the `boot2docker` image (or any system where you are running the Docker Engine).
2. Run `sudo sh`
3. Run:

```
cd /usr
mkdir certs
openssl genrsa -out /usr/certs/domain.key 2048
openssl req -new -key /usr/certs/domain.key -out /usr/certs/ \
   domain. csr -subj "/C=UK/ST=Surrey/L=Richmond \
   /O=Acme/OU=Development/CN=<host running Docker Registry>"
openssl x509 -req -days 365 -in /usr/certs/domain.csr -signkey \
   /usr/certs/domain.key -out /usr/certs/domain.crt -CAkey /usr \
   ca_certs/private/cakey.pem -CA /usr/ca_certs/cacert.pem \
   -CAcreateserial  -CAserial /usr/ca_certs/serial
```

where:

- `/usr/ca_certs/private/cakey.pem`

  Is the private key for the CA.

- `/usr/ca_certs/cacert.pem`

  Is the public certificate for the CA.

- `/usr/certs/domain.key`

  Is the private key for the registry server.

- `/usr/certs/domain.crt`

  Is the public certificate for the registry server.

Trusting a Certificate Authority

Next you need to tell the Linux operating system inside the `boot2docker` image to trust the generated CA certificate. Usually Linux has scripts to do this but they are missing from the `boot2docker` Linux distribution so you need to do it manually.

Replace `4c05dafl` with whatever hash is returned by the `openssl x509` hash command:

```
cp /usr/ca_certs/cacert.pem /usr/local/share/ca-certificates/yourca.pem
cd /etc/ssl/certs
ln -s /usr/local/share/ca-certificates/yourca.pem
openssl x509 -hash -in serenaca.pem
ln -s serenaca.pem 4c05dafl
```

Configuring Docker to Trust a Registry

Every Docker Engine machine from which you want to access a remote Docker Registry requires a directory that matches your registry URL. Add your CA certificate and restart docker.

1. Use SSH to log into the `boot2docker` image on each remote host.
2. Run:

   ```
   mkdir -p /etc/docker/certs.d/myregistryserver:5000
   cd /etc/docker/certs.d/myregistryserver:5000
   ```

3. Create a file called `ca.crt` containing the contents of the CA public certificate in: `/usr/ca_cert/cacert.pem`
4. Restart Docker:

   ```
   /etc/init.d/docker restart
   ```

Creating a Keystore

1 Create a PKCS12 keystore that the Dimensions CM Docker Registry can use to identify itself:

```
openssl pkcs12 -export -in /usr/certs/domain.crt -inkey /usr/ \
   certs/domain.key out /usr/certs/keystore.p12 -name \
   tomcat -CAfile /usr/ca_certs/cacert.pem -caname root -chain
```

2 Copy the generated `keystore.p12` file to the system running the Docker CM Registry JAR.

Starting a Docker Registry in SSL Mode

Now that you have a SSL keystore, specify that it can be used when you start a Dimensions CM Docker Registry:

```
java -jar docker-registry.jar --port=5000 --ssl=true \
   --keystore=C:\temp\keystore.p12 \
   --keystore-type=PKCS12 \
   --keystore-pass=changeit \
   --cm-url=scm:dimensions://mycmserver/cm_typical@dim14 \
   --cm-product=QLARIUS --cm-stream=QLARIUS:DOCKER_STREAM \
   --cm-part=QLARIUS:DOCKER_IMAGES.A;1
```

Use the keystore password that you entered when you created the keystore.

Logging in Securely to a Remote Docker Registry

Login securely to the remote Docker Registry:

```
docker login myregistryserver:5000
```

If the login is successful you have secured your Dimensions CM Docker Registry using SSL.

Chapter 3

Using Docker

Creating a Docker Image

- To create a Docker image:

  ```
  docker build -t <image name> .
  ```

  This is an example of how to build an image. Using Dimensions CM Docker Registry does not affect how you create images.

- You can optionally add a tag to identify an image:

  ```
  docker build -t <image name>:<tag name> .
  ```

  For example:

  ```
  docker build -t myimage:turnover .
  ```

Logging into a Dimensions CM Docker Registry

1. Run this command and specify the CM host machine and port number:

   ```
   docker login <registry server>:<port>
   ```

   For example:

   ```
   docker login myregistryserver:5000
   ```

2. Enter credentials to log into CM.

Pushing an Image

To push (deliver) an image to a Dimensions CM Docker Registry:

```
docker push <registry server>:<port>/<image name>:<tag>
```

For example:

```
docker push myregistryserver:5000/insurancesvc:turnover
```

NOTE

- If Docker image layers do not already exist, they are delivered as items at the top of the stream.
- The item type and format used are determined by CM upload rules. Check that you have an upload rule, with the format `BINARY`, to handle files with no file extension.
- By default, the registry automatically maps each image to a new, unique CM design part called `<image name>`. The new part is created in a *product* and owned by a *parent design part*. The product and design part are configured by your administrator when the Docker Registry is started, see page 10 for details.

- To use upload rules to identify the owning CM design part for an image, specify this parameter when you start the registry:

  `--cm-create-parts=false`

  For example, assume you have the following upload rule defined in the CM administration console:

  Path: `test/**`

  Part: `QLARIIUS:TEST.A;1`

  Format: `BINARY`

  When you push an image called `test/insurancesvc` the manifest of the image is created as an item owned by the `QLARIIUS:TEST.A;1` design part. A new baseline is created from that design part containing the layers used by the image and the image manifest file:

  `<image name>-<tag>-<version>`

  For example:

  `TEST/INSURANCESVC-TURNOVER-1`

- To save space in your repository, after you push an image:
  - Any layer that is no longer required is automatically deleted.
  - Only baselines that are at the *Approved* lifecycle state are preserved. All other baselines of the image are deleted.

Approving an Image

Images are only visible to the user who created them. To make an image visible to all users:

1 Log into a CM client as the owner of the baseline associated with the image.

2 Action the baseline to the Approved lifecycle state (this state was configured when the Docker Registry started, see page 10).

NOTE

- By default all images require approval though this can be switched off by a CM administrator when a Docker Registry is started.
- After an image is approved all users can search for and pull it, providing they have the required Dimensions CM privileges and roles.

Searching for Images

- To list all images:

  ```
  docker search <registry server>:5000/
  ```

  Example output:

  ```
  NAME                DESCRIPTION
  hello-world         latest hello world image
  insurancesvc        Docker image
  ubuntu              Some Ubuntu image
  web/qlarius          Qlarius web app
  ```

  This is a list of all the images that you created or approved.

- To search for images using a keyword:

  ```
  docker search <registry server>:5000/ubun
  ```

  Example output:

  ```
  NAME                DESCRIPTION
  ubuntu              Some Ubuntu image
  ```

- To search for images in a namespace:

  ```
  docker search <registry server>:5000/web/q
  ```

  Example output:

  ```
  NAME                DESCRIPTION
  web/qlarius          Qlarius web app
  ```

Pulling an Image

To pull a specific image tag:

```
docker pull <registry server>:5000/<image>:<tag>
```

For example:

```
docker pull myregistryserver:5000/insurancesvc:turnover
```

The pull command downloads the image from the corresponding baseline. All Dimensions CM privileges and roles are honored by the pull request.

Specifying Change Requests

When you build or modify a Docker image you can specify Dimensions CM change requests using a LABEL called `com.serena.requests` inside your Docker file. For example if the Docker file contains:

```
FROM tomcat:8-jre8
LABEL com.serena.requests="QLARIUS_CR_21,QLARIUS_CR_25"
CMD ["catalina.sh", "run"]
```

when the image is pushed to the Dimensions CM Docker Registry, the resulting baseline is related to requests QLARIUS_CR_21 and QLARIUS_CR_25.

Chapter 4

Using the Example Docker Image

Overview

A sample Docker image file is included with the Dimensions CM server installation and is placed in the same folder as the JAR file. The image is based on Alpine Linux (a compact Linux distribution ideal for containers) and includes:

- Oracle Java JDK 8 (1.8.0_74-b02)
- The Dimensions CM Docker Registry JAR file
- A shell script to launch the registry

Building the Sample Image

To build the image go to the directory containing the Docker file and run Docker build, for example:

```
docker build -t cmregistry .
```

Running the Sample Image

Parameters

The command line parameters for the Dimensions CM Docker Registry are passed to the image as environment variables. Parameters in square brackets [ ] are optional.

```
docker run -it
   -e CM_URL=scm:dimensions://<server>/<base database>@<db connection>
   -e CM_PRODUCT=<product id>
   -e CM_STREAM=<product id>:<stream id>
   -e CM_PART=<product-id>:<part-id>.<variant>
   [-e CM_PART_TYPE=<part type>]
   [-e CM_BASELINE_TYPE=<baseline type>]
   [-e CM_BASELINE_STATE=<approval state>]
   [-e CM_CREATE_PARTS=<true or false>]
   [-e CM_CACHE=<true or false>]
   [-e CM_CACHE_DIR=<cache directory>]
   [-e CM_CACHE_SIZE=<maximum cache size>]
   [-e CM_LOG]
   -p <port number>:5000
   cmregistry
```

where:

- `CM_URL=scm:dimensions://<server>/<base database>@<db connection>`

  Specifies a Dimensions CM server name, base database, and database connection, for example:

  `CM_URL=scm:dimensions://mycmserver/cm_typical@dim14`

- `CM_PRODUCT=<product id>`

  Specifies a product ID, for example: `CM_PRODUCT=QLARIUS`

- `CM_STREAM=<product id>:<stream id>`

  Specifies a product ID and a stream, for example:

  `CM_STREAM=QLARIUS:DOCKER_IMAGES`

- `CM_PART="<product-id>:<part-id>.<variant>"`

  Specifies a complete design part specification, for example:

  `CM_PART="QLARIUS:DOCKER_IMAGES.A;1"`

- `CM_PART_TYPE=<part type>`

  Specifies a design part type.

  Default: `CM_PART_TYPE="SUB-SYSTEM"`

- `CM_BASELINE_TYPE=<baseline type>`

  Specifies a baseline type, for example:

  `CM_BASELINE_TYPE="BASELINE"`

- `CM_BASELINE_STATE=<approval state>`

  Specifies a baseline approval state, for example:

  `CM_BASELINE_STATE="approved"`

- `CM_CREATE_PARTS=<true or false>`

  - (Default) `true`: the registry maps each image to a single design part that is automatically created by the registry. Each design part has a unique name.
  - `false`: the registry uses upload rules to identify the owning design part for each image. The registry can support multiple images with the same name or namespace. For example, you can push `"lib1/server/tomcat"` and `"lib2/server/tomcat"`.

- `CM_CACHE=<true or false>`

  You can cache the image layers inside the registry to avoid fetching the same content from the server to the registry on every pull operation. Serena recommends setting this parameter if your registry is remote from your Dimensions CM server.

- `CM_CACHE_DIR=<cache directory>`

  (Only valid if you specify `CM_CACHE=true`) Specifies the directory where the cache will be stored. The default is the user's HOME directory:

  Windows: `%USERPROFILE%\Serena\Cache`

  UNIX: `~\.Serena\Cache`

- `CM_CACHE_SIZE=<maximum cache size>`

  (Only valid if you specify `CM_CACHE=true`) Specifies the maximum size of the cache in megabytes.

  Default: `CM_CACHE_SIZE=2000`

- `-p <port number>:5000`

  By default, the registry listens on port 5000, however you can map to a different port. For example, to listen on port 5555:

  `-p 5555:5000`

- `CM_LOG`

  Specifies the location for the log files inside the running container (not on the file system of the host running the container). Default:

  `/opt/docker-registry/logs/docker-registry.log`

When the container is running, use it the same as a Dimensions CM Docker Registry JAR file, for details see page 17.

Running an Image in SSL Mode

To run an image in Secure Socket Layer (SSL) mode, specify the `SSL`, `KEYSTORE`, and `KEYSTORE_PASS` environment variables.

```
docker run -it -v /c/Users/myuser/certs:/certs \
   -e CM_URL=scm:dimensions://mycmserver/cm_typical@dim14 \
   -e CM_PRODUCT=QLARIUS \
   -e CM_STREAM=QLARIUS:DOCKER_IMAGES \
   -e CM_PART="QLARIUS:DOCKER_IMAGES.A;1" \
   -e SSL=true \
   -e KEYSTORE=/certs/mykeystore.p12 \
   -e KEYSTORE_PASS=changeit \
   -p 5000:5000 \
   cmregistry
```

NOTE For SSL mode mount a volume containing your keystore to the `/certs` mount point using the `-v` option. In the example above `/c/Users/myuser/certs` is mounted to `/certs` inside the container. That directory contains the `mykeystore.p12` file containing certificates for the machine running the Docker engine.