Table of contents

Introduction and overview

Introduction and overview

Explanation of terms

The following documentation uses terms that are explained in the ↘ Glossary.

Useful web links

ctrlX Store in the web

HOW-TO section

ctrlX AUTOMATION FORUM

ctrlX AUTOMATION Community

Container Engine App - Basics

Operating principle

The Container Engine App provides a Docker engine on ctrlX devices, which enables the execution of Docker images.

Multiple Docker images can be installed and operated on the ctrlX device via separate Docker image apps.

Docker images are configured and started via the Docker container configuration file **docker-compose.yml**, see: ↘ Example

Snap templates for ctrlX devices

A snap template can be used to create a Docker image application for ctrlX devices from a Docker image and the corresponding Docker configuration files.

Installation and integration into the web interface of the ctrlX device

The Container Engine App can be transferred and installed on a ctrlX device either via the ctrlX Store or via the ctrlX Device Portal, see:

ctrlX Store in the web

Web documentation for the ctrlX Device Portal

The procedure for installing apps on ctrlX devices is described in the documentation for the ctrlX runtime, see: Web documentation

Installing the Container Engine App adds the following elements to the web interface of the ctrlX device:

↘ Window – *"Images"*

↘ Window – *"Containers"*

Licensing

The operation of the Container Engine App on a ctrlX device is subject to licensing and requires the following license:

Type code	Part number
SWL-XC*-DOE-DOCKERENGINE*-NNNN	R911409943

Users and authorizations

The user authentication of the Container Engine App is linked to the user administration of the ctrlX device, see: Web documentation

For the configuration and operation of the Container Engine App, the following authorizations are required:

- Manage Container Engine configuration
- Start/Stop Container and view Container Engine configuration
- View Container Engine configuration

The authorizations can be managed and configured in the web interface of the ctrlX device, see: Web documentation

In case of insufficient authorizations, sporadically no data is displayed or buttons are inactive.

Interfaces

The Container Engine App contains a Docker engine with Docker daemon (dockerd) and the Docker command line interface (Docker CLI).

The app also provides two content interfaces that can be used to exchange data between the Container Engine App and a container image app.

The following UML diagram schematically represents the interfaces and the data flow:

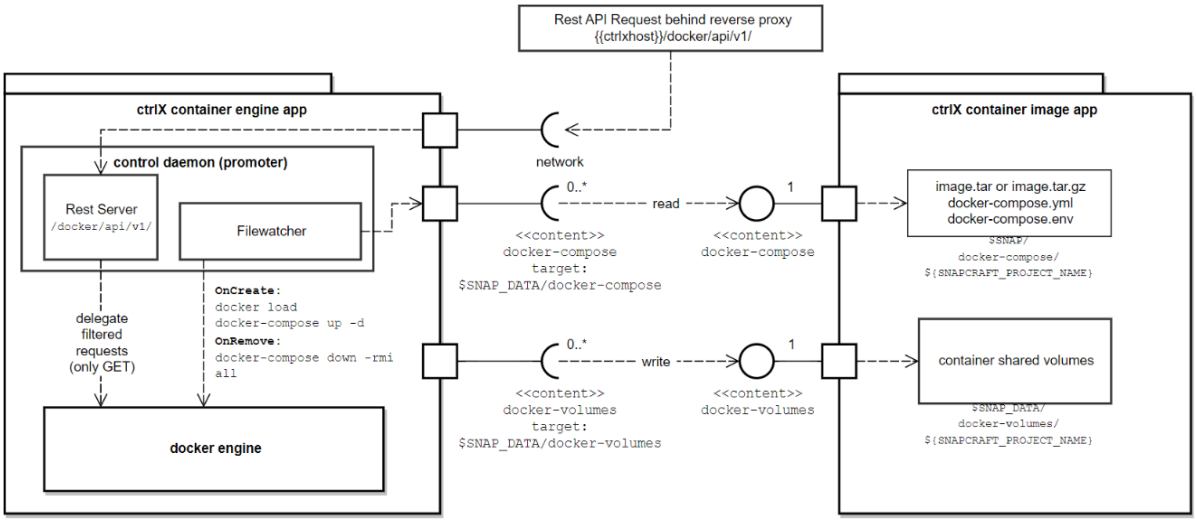

Fig. 1: Interfaces and data flow

Content interface "docker-compose"

The Docker Compose interface is used to provide the Docker images and the associated configuration files for the Container Engine App.

The following files are provided via Docker Compose:

```
└── docker-compose
    ├── docker-compose.env          ← Docker-Compose Variablen (optional)
    ├── docker-compose.yml          ← Docker-Container Konfiguration
    ├── image-1.tar / image-1.tar.gz
    ├── ...                         ← Docker-Image Archiv(e)
    └── image-n.tar / image-n.tar.gz
```

Content interface "docker-volumes"

The Docker-Volumes content interface allows write access from a Docker container to an image app directory. A Docker container can write from the ctrlX Container Engine App to a writable directory of the container image app via this interface.

Example directory:

```
$SNAP_DATA/docker-volumes/{snap-name}
```

Exemplary excerpt from a docker-compose.yml for the assignment of a

Docker volume with write access:

```
services:
  {service}:
    image: {image}
    volumes:
      - ${SNAP_DATA}/docker-volumes/{snap-name}/data:/data:rw
```

Docker Engine API

The REST interface of the Docker Engine API can be accessed via the following address:

https://{host}/docker/api/v1

A complete description of the Docker REST API can be found under the following web link:

https://docs.docker.com/engine/api/latest/

Example of a REST command via curl - here: retrieving the existing Docker images as json:

```
curl --location --request GET "https://{host}/docker/api/v1/images/json" --header
"Authorization: {token}"
```