Table of contents

Container Engine app - Basics

Container Engine app - Basics

Container Engine app for ctrlX CORE

The ctrlX Container Engine app allows to run a Docker container on ctrlX CORE.

A Docker engine is provided in the ctrlX Container Engine app. The Docker REST API can be accessed via the ctrlX CORE reverse proxy.

A Docker image is deployed to the ctrlX CORE via a separate ctrlX Container Image app. In addition to the Docker image, this app contains the corresponding Docker configuration files required to run a Docker container application.

The most important terms are explained in the ↘ Glossary.

Introduction

The ctrlX Container Engine app provides a Docker engine on ctrlX CORE. It allows Docker images to be executed via a Docker container configuration file **docker-compose.yml**.

One or more Docker images can be installed on ctrlX CORE via separate Docker image apps. The images are configured and started via **docker-compose.yml**.

A snap template can be used to create a Docker image application for the ctrlX CORE from a Docker image and the corresponding Docker configuration files. This Container Image app, like all other ctrlX CORE apps, can be installed on a ctrlX CORE via the ctrlX Online Store or via the ctrlX Device Portal.

Installation and integration into the ctrlX CORE web interface

The app installation is described in the documentation at ctrlX CORE runtime, refer to the Web documentation.

By installing the app, the ctrlX CORE web interface around the *"Container Engine"* node with the windows *"Images"* and Containers in the ctrlX CORE web interface is extended, see

↘ Images

↘ Containers

Licensing

Required license to operate the Container Engine app on a ctrlX CORE control:

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

Users and authorizations

The Container Engine user authentication is connected to the ctrlX user management. To access the Container Engine app, authentication (login) is required in the ctrlX CORE control system. In the section *"User & permissions"*, Container engine-specific permissions can be specified, which control the data access to the Data Layer of the controller, see web documentation.

The following permissions exist for Container Engine:

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

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

Interfaces

The ctrlX Container Engine app includes a Docker engine with Docker daemon (dockerd) and Docker-Command-Line-Interface (Docker CLI). The app provides two content interfaces. Via these interfaces, data can be exchanged between the ctrlX Container Engine app and a ctrlX Container image app.

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

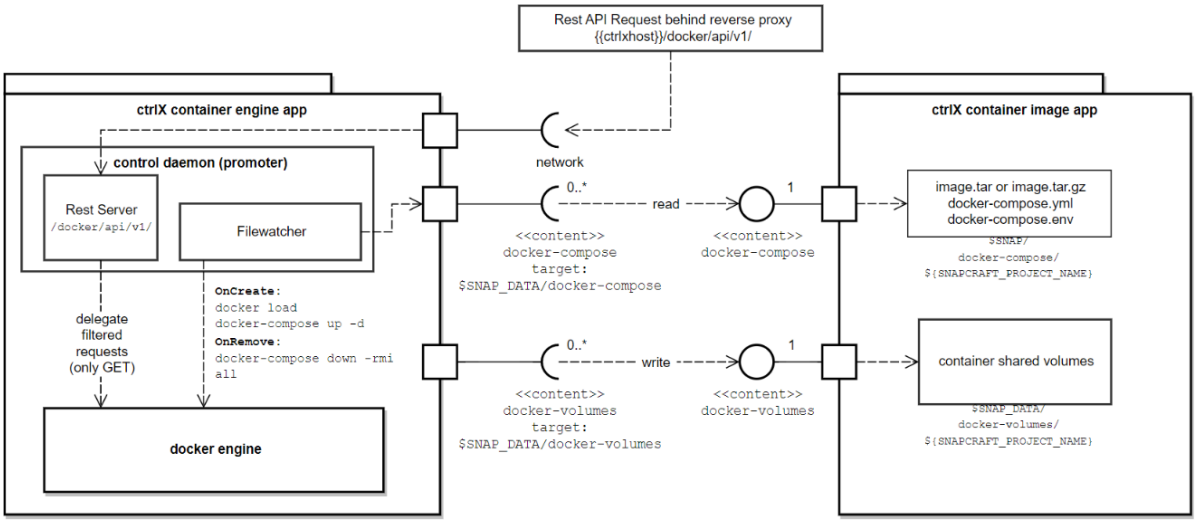

Fig. 1: Interfaces and data flow

Content interface "docker-compose

Using the Docker Compose interface, the Docker images and associated configuration files are provided to the ctrlX 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 be accessed from the ctrlX Container Engine app via this interface to a writable directory

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

of the Container Image app.

Here is an example excerpt from a docker-compose.yml for mapping 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 https://{{host}}/docker/api/v1 address.

A full description of the Docker REST API can be found here:

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}}"
```