Docker Containers Manager: A Simple Toolkit for Isolated Work with Shared Computational, Storage, and Network Resources

Stanislav Polyakov[1], Alexander Kryukov, Andrey Demichev
Skobeltsyn Institute of Nuclear Physics, Lomonosov State University, Moscow, Russia
[1] s.p.polyakov@gmail.com

Containers are isolated user-space instances run by the same OS kernel, allowing their users to enjoy many of the benefits of virtual machines with little overhead. With containers, multiple applications can be installed on the same computer without the risk of dependencies conflict, and run without additional costs for emulating hardware and different operating systems. Docker is one of the most popular container platforms for Linux. It provides numerous tools for configuring and managing containers. However, giving users full access to Docker capabilities on a server amounts to giving them unrestricted access to the server.

We present a simple set of command line interface tools called Docker Containers Manager (DCM) that allow users to create and manage Docker containers with pre-configured SSH access while keeping the users isolated from each other and restricting their access to the Docker features that could potentially disrupt the work of the server. The tools allow to deploy and debug medium-sized distributed systems for simulation in different fields on one or several local computers.

Users can access DCM server via SSH and are automatically redirected to DCM interface tool. From there, they can create new containers, view the status of the existing containers (only those started by the user are visible), stop, restart, pause, unpause, and remove them. The containers will also be accessible via SSH using the same private key(s), but on different server ports. To create a new container, a user must specify an image to be used as the container's root filesystem. Some images are provided by the DCM server administrators. After a user has configured a container to meet one's needs, the changes can be committed to a new image from which the user can create copies of the container. Users can also view the list of available images and remove their own images. Potentially these images can be migrated to publicly accessible repositories.

When creating a new container, a user can request additional publicly available ports to be mapped to the respective ports of the container, allowing for some network services to be run within the container. Server administrators can also permit certain users to use additional options when creating their containers, e.g. setting a restarting policy or lifting some security restrictions.

Lastly, server administrators can use scripts to create new users (setting up their access and storage directories to be mapped to every container) and delete users, also removing all their containers and (optionally) images.

All commands were implemented as Python scripts.