YASTD: A Simple Set of CLI Tools to Manage Docker Containers

S. P. Polyakov[a], A. P. Kryukov, A. P. Demichev

Skobeltsyn Institute of Nuclear Physics, M.V.Lomonosov Moscow State University (SINP MSU),
1(2), Leninskie gory, GSP-1, Moscow, 119991, Russia

E-mail: [a] s.p.polyakov@gmail.com

We present a set of tools to manage Docker containers named YASTD (Yet Another Simple Tools for Docker). It has three primary purposes:

- to allow users to create containers remotely accessible via secure shell (SSH);
- to let users configure their containers and save the changes as new images;
- to isolate users from each other and restrict their access to the Docker features that could potentially disrupt the work of a server.

The tools are accessible via a simple command line interface.

The commands for managing containers allow creating containers from available images, listing the available containers, stopping and restarting containers, pausing and unpausing all processes within a container, and removing containers. Also available are the commands to create new images from the changes made to the containers, list the available images, and remove images. The users cannot see, modify, or remove containers and images created by other users.

We also give an assessment of the security level of the tools and outline the possible approaches to its improvement.

Keywords: cloud computing, container virtualization, Docker

The work was supported by RFBR. Grant No 15-07-09309

Introduction

Let us suppose we have a server with computational or storage capabilities we want to share between several users. Two of the approaches to doing so are giving the users accounts on the server, or creating a separate virtual machine for each user. Taking former approach means that users will be limited to a specific version of a specific operating system, will not be able to install some software or configure their environment, and when the necessary software is installed it may not be useful for some users due to dependencies conflicts. The latter approach has none of these drawbacks, but emulating hardware has significant overhead. Container virtualization [Soltesz S. et al., 2007] provides a middle ground between the two, allowing to have some of the isolation with small overhead [Felter W. et al., 2015] and some additional benefits we discuss in the next section.

The tools we present are using the container virtualization approach. They allow users to create and manage their own containers (container virtualization analogue of virtual machines) with pre-configured SSH access. This includes access to a privileged user (root) account inside a container that can configure the container and install the necessary software. Thus a user can run multiple instances of a software, or break up a software into several interacting components set up in isolated environments. This is achieved with a simple program (basically a set of scripts) leveraging some of the features of a container virtualization tool Docker [Docker].

Section 2 outlines some possibilities of container virtualization and features of Docker. Section 3 presents the YASTD tools and describes their functionality. Section 4 is about the security limitations of the tools. Section 5 concludes the paper and outlines some possibilities for further development of the tools.

Container virtualization and Docker

Container virtualization is a virtualization method in which the kernel of an operating system allows the existence of multiple isolated user-space instances, or containers. Thus a software needs to be compatible with the host OS to be run within a container.

Docker is a rapidly developing but already very popular container virtualization tool for Linux. Filesystems of Docker containers have read-only parts called images. Changes to the filesystem made while the container is working will be stored as a separate layer, and can be saved as a new image without the need to copy the read-only part. This approach is called copy-on-write and allows to store multiple images with small variations between them using very little disk space.

Docker provides a number of tools for creating, monitoring, and manipulating containers. One of its features is a volumes option which allows to map an arbitrary host directory into a container directory. In addition to all the benefits of this feature, it has a side effect that a user with full access to Docker commands can get an access to any file of the host. Therefore if we want users to be able to create their own containers without accidently disrupting the work of the server, we need to restrict their access to Docker commands.

YASTD features and tools

YASTD (Yet Another Simple Tools for Docker) is an intermediary giving users a restricted access to some of the Docker commands. The users can create containers, save modified containers as new images and create new containers from these images. A very basic set of Docker commands for managing the images and containers is also available.

Features

YASTD has four features that set it apart from a crippled version of Docker.

A (Access). Each user has automatically configured SSH access to any container they create, both as a regular user and as a privileged one (root). This allows users to configure their containers and work with them as they prefer.

B (Borders). YASTD isolates users from each other. Users cannot see or modify containers and images created by other users. (A container created by a user can still be configured by that user to give access to others.)

C (CLI). The users also have restricted access to the host: the only way a user can directly interact with it is via a simple command line interface.

D (Directories). Each user has two home directories within any container that are mapped from the storage directories assigned to the user on the host. As a result, the contents of these directories are the same for all containers of the user, and they are not saved as a part of the user-created images.

How it works

A server administrator needs to install Docker, copy YASTD files on the server and configure YASTD, specifying a range of ports to be used, location of the storage directory, and some other data. At least one Docker image should be preconfigured to automatically launch `sshd` and allow SSH login with public keys. New user accounts can be added by a script that configures their access to the server so that CLI is started automatically when they attempt to log in, and copies their public SSH keys to `.ssh/authorized_keys` files in the two storage directories assigned to the user.

When a user submits a valid command to create new container, YASTD uses volumes option to map the storage directories of the user into home and root directories inside the container respectively, allowing both A and D features mentioned earlier. The container is assigned a random free port from the specified range and its 22 port is mapped to the assigned server port, giving the user SSH access the container.

List of YASTD tools

YASTD capabilities for managing containers:
- create a container from an image,
- list the containers started by the user,
- stop and restart a container,
- pause and unpause all processes within a container,
- remove a container.

YASTD capabilities for managing images:
- create a new image from a container's changes,
- list the images available to the user,
- remove an image.

Security

Docker is not considered secure when users are allowed to execute arbitrary commands within a container («Containers do not contain», [Walsh]), and giving root access inside a container further weakens the security. The security can be strengthened using Docker's settings to give root a restricted set of capabilities instead of full privileges, as well as using some measures of security in addition to those provided by Docker [Petazzoni].

As it stands, YASTD is meant to be used in situations where users can be trusted to make no attempts to break out of their containers, otherwise impede the functioning of the host and work of other users, and to give no access to their accounts or privileged access to their containers to outside parties.

Conclusions and further work

We have presented a set of tools that allow users to remotely create Docker containers that can be configured by the user, save and access the changes made to the containers, and manage the containers. The users are isolated from each other. The tools can be accessed via a simple command line interface.

The ideas for further development include adding measures to improve the security of the server, giving users the option to migrate their images to outside repositories, and creating a Web interface.

References

Soltesz S. et al. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors // ACM SIGOPS Operating Systems Review. — 2007. — Vol. 41, No. 3. — P. 275–287.

Felter W. et al. An updated performance comparison of virtual machines and linux containers // Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On. — 2015. — P. 171–172.

Docker – Build, Ship, and Run Any App, Anywhere. [Electronic resource]. URL: https://www.docker.com

Walsh D. Are Docker containers really secure? [Electronic resource]. URL: https://opensource.com/business/14/7/docker-security-selinux (accessed 30.10.2016).

Petazzoni J. Containers & Docker: How Secure Are They? [Electronic resource]. URL: https://blog.docker.com/2013/08/containers-docker-how-secure-are-they/ (accessed 30.10.2016).