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

Stanislav Polyakov, Alexander Kryukov, Andrey Demichev

Skobeltsyn Institute of Nuclear Physics, Moscow State University

The problem

Let's suppose we want to share the resources of a server between several users in such a way that they can:

- access the server remotely,
- configure their working environment and install the necessary software,
- save the changes for future use.

Possible solutions

- direct access to the server −

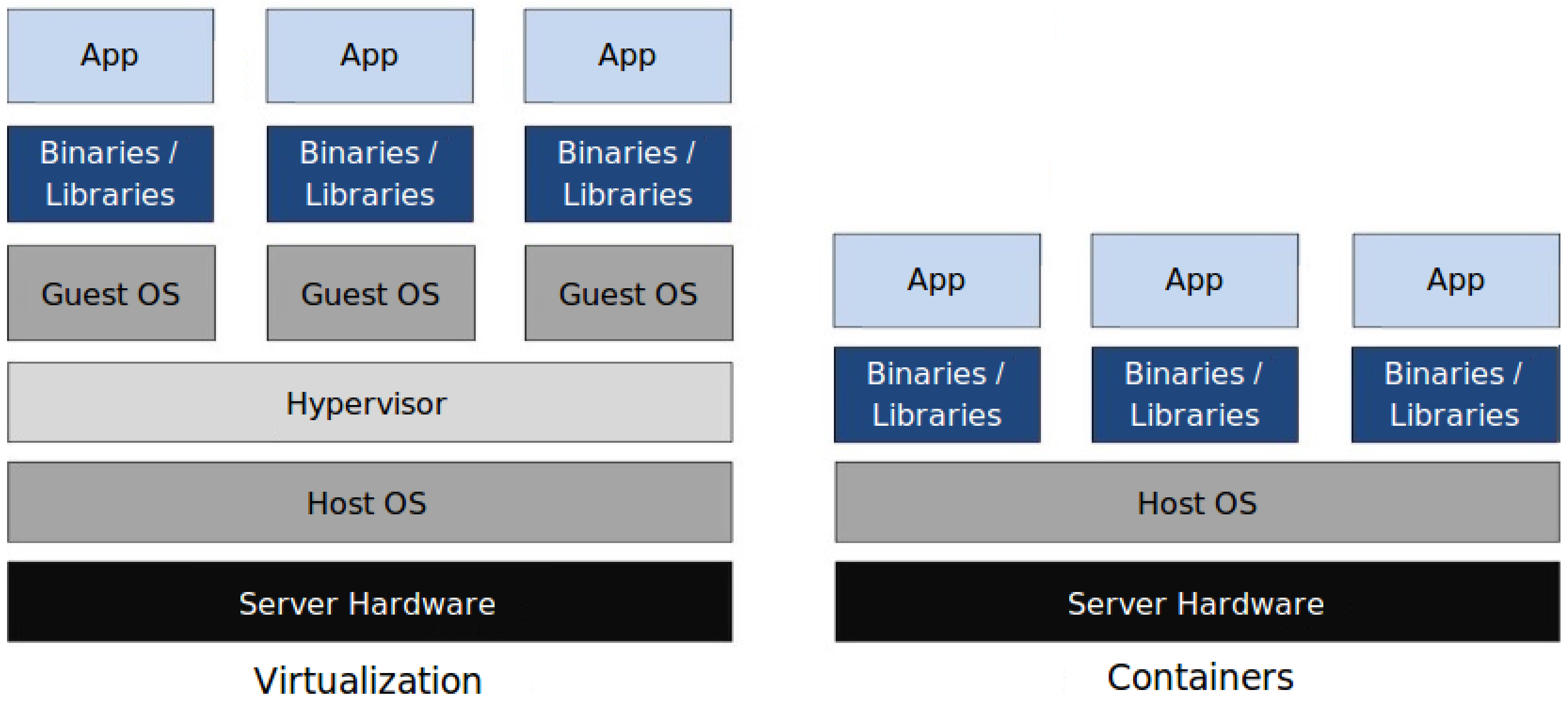
  (either no privileges to install software or no isolation between users; dependency conflicts)
- virtual machines +
- containers ?

Container virtualization

Container virtualization, or operating-system-level virtualization, is an operating system feature in which the kernel allows the existence of multiple isolated user-space instances. The instances are called containers (sometimes partitions or jails).

Container virtualization has evolved from Linux chroot mechanism that allows to change an apparent root directory for a process and its children.

Containers and virtual machines



Copy-on-write

Copy-on-write technique used in most container virtualization systems allows multiple containers that need copies of the same file use the file without actually copying it until one of the containers tries to change it.

Docker

Docker is a container virtualization platform first released in March 2013.

Google released their container virtualization system named lmctfy in March 2013. They have stopped developing it in 2015: "We have been collaborating with Docker over libcontainer and are in process of porting the core lmctfy concepts and abstractions to libcontainer. We are not actively developing lmctfy further and have moved our efforts to libcontainer."

About Docker

In Docker, a container is started as a process inside a read-only filesystem called image. Thanks to copy-on-write mechanism images may have multiple layers with each layer only recording changes, and often take up very little disk space.

Docker has a volumes mechanism that allows containers to have host directories mapped into them. Container ports can be mapped to the host ports as well.

Can we let users create their own containers?

Because of the volumes mechanism and privileged mode for containers we can't let users of plain Docker start their own arbitrary containers without giving them control over the server similar to that of the root.

So we need to restrict the users’ access to the Docker’s features somehow.

Docker Container Manager

The main purpose of the Docker Container Manager (DCM) is to let users:

- create containers with pre-configured SSH access;
- save changes made in a container to a new image (and start containers from these images);
- manage containers (stop, restart, delete etc.) and images.

DCM keeps users isolated from each other and does not let them access Docker commands directly.

Creating containers with DCM

New container is created by invoking a `docker run` command with the options that ensure that

- SSH port of the container is mapped to a random port of the server,
- home directories /root and /home/<user> are mapped from two storage directories assigned to the user on the server.

As a result, these two directories are the same in all containers created by the user, and by default the containers are accessible to the user only.

DCM containers: additional options

Aside from mapping the SSH port of a container to a random server port, a user can choose one or several ports of a container to be mapped directly to the server ports with the same numbers. Several users can use the same port number if the server has IP aliases.

Server administrators can let specific users use specific additional options of docker run command - e.g. to map additional server directories to their containers, or choose restarting policy.

Managing containers

DCM allows users to:

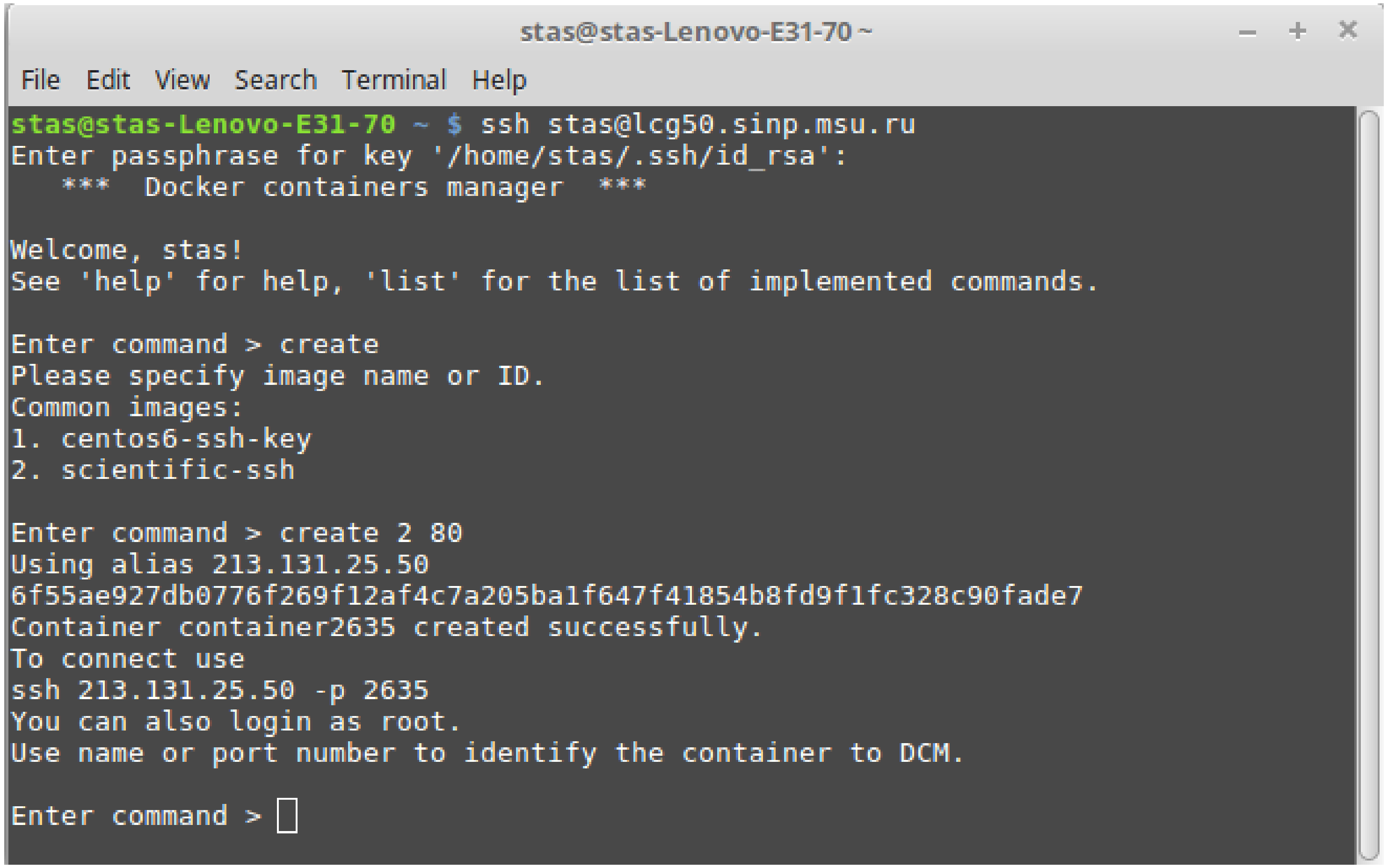
- list the containers started by the user,
- stop and restart a container,
- pause and unpause all processes within a container,
- remove a container.

Creating and managing images

DCM users can:

- create a new image from a container's changes,
- list the images available to the user (including read-only images provided by the server administrators and the images created by the user),
- remove an image.

DCM interface


stas@stas-Lenovo-E31-70 ~
File Edit View Search Terminal Help

```
stas@stas-Lenovo-E31-70 ~ $ ssh stas@lcg50.sinp.msu.ru
Enter passphrase for key '/home/stas/.ssh/id_rsa':
   ***  Docker containers manager  ***

Welcome, stas!
See 'help' for help, 'list' for the list of implemented commands.

Enter command > create
Please specify image name or ID.
Common images:
1. centos6-ssh-key
2. scientific-ssh

Enter command > create 2 80
Using alias 213.131.25.50
6f55ae927db0776f269f12af4c7a205ba1f647f41854b8fd9f1fc328c90fade7
Container container2635 created successfully.
To connect use
ssh 213.131.25.50 -p 2635
You can also login as root.
Use name or port number to identify the container to DCM.

Enter command > 
```

Tools for administrators

DCM administrators have scripts to create DCM users (setting up their access to the interface, creating their storage directories, and copying their public key files to these directories) and remove them (also removing their storage directories, containers, and optionally images).

Security

In earlier versions of Docker, it was considered unsafe to let even unprivileged users to run arbitrary commands within a container. By now some security features such as user namespaces (not used by default) and seccomp filtering were added to make it more secure. But without additional security measures it is probably best to only allow users who can be trusted to make no intentional attempts to break out of their containers.

The end