University Göttingen
GWDG AG-C
Freja Nordsiek

Tutorial 1 / 2023.10.18
HPC System Administration – Containers / WiSe 2023/24
11 Minutes Total

Learning Objectives

This is the tutorial on using Podman (`https://podman.io`) and other tools in the Docker/OCI ecosystem. The learning objectives in the tutorial are

- Install Podman (`https://podman.io`)
- Make sure subuids and subgids are setup for your user
- Make sure user namespaces are enabled
- Build and run basic containers

Tools

- Podman (`https://podman.io`)
- `shadow-utils-subid`
- user namespaces

Contents

For the Podman (`https://podman.io`) and other tools in the Docker/OCI ecosystem part of this course,

Install Podman 1: Tutorial (1 min)

Steps

1. `$ sudo dnf install podman`

Further Reading

- `dnf` – `https://dnf.readthedocs.io`

Setup subuid and subgid 2: Tutorial (2 min)

Steps

1. Determine your user name (look at the command prompt or use the `whoami` command)
2. Check that your user has a subuid and subgid range in the files `/etc/subuid` and `/etc/subgid`. The files are formatted like `USERNAME:STARTID:COUNT` where `STARTID` is the first UID number to use, and `COUNT` is how many to allocate. `STARTID` should be greater than 65536, and is traditionally an integer multiple of 100k. `COUNT` should be 65536.
3. If the entries in `/etc/subuid` and/or `/etc/subgid` are wrong:
   a) Remove them by
   ```
   $ sudo usermod --del-subuids STARTID-ENDID USERNAME
   ```
   or
   ```
   $ sudo usermod --del-subgids STARTID-ENDID USERNAME
   ```
   respectively where `ENDID` = `STARTID` + `COUNT`.

   b) If you have no entry in `/etc/subuid` and/or `/etc/subgid` or had to remove them, add subuids and/or subgids for yourself by
   ```
   $ sudo usermod --add-subuids STARTID-ENDID USERNAME
   ```
   or
   ```
   $ sudo usermod --add-subgids STARTID-ENDID USERNAME
   ```
   respectively where `STARTID` should be greater than 65536 and `ENDID` − `STARTID` = 65536. For example,
   ```
   $ sudo usermod --add-subgids 100000-165536 USERNAME
   ```

Further Reading

- `subuid` – https://man7.org/linux/man-pages/man5/subuid.5.html
- `usermod` – https://man7.org/linux/man-pages/man8/usermod.8.html

Setup user namespaces 3: Tutorial (2 min)

Steps

1. Check the maximum number of user namespaces that are allowed by
   ```
   $ sudo sysctl -a | grep user_namespaces
   ```
2. If it is zero, user namespaces are disabled since none can be created. It needs to be some positive integer, ideally several thousand. If you need to change it:
   a) Change it just for the current boot by
   ```
   $ sudo sysctl user.max_user_namespaces=VALUE
   ```
   where `VALUE` is some integer that is at least 1000. 1k is enough for personal use, but a machine with many users should have at least 10k.

   b) To set a different value on boot, it needs to be added to the `sysctl` config at `/etc/sysctl.conf` and the files in `/etc/sysctl.d/`. You need to first see if it is already set in one of those files and change its value. A fast way to check is with
   ```
   $ grep -r user_namespaces /etc/sysctl.*
   ```
   If it isn't in any of the files, add a file in `/etc/sysct.d` with the following line:

```
user.max_user_namespaces = VALUE
```

Further Reading

- `sysctl` – https://man7.org/linux/man-pages/man8/sysctl.8.html
- `/etc/sysctl.d/*` – https://man7.org/linux/man-pages/man5/sysctl.d.5.html
- `user_namespaces` – https://man7.org/linux/man-pages/man7/user_namespaces.7.html

Create and modify a container, and make a new container image from it 4: Tutorial (3 min)

Steps

1. Create the container from its container image with some `NAME` you choose using Podman's `container create` command, by
   `$ podman container create -t --name=NAME docker.io/library/busybox:musl`
   The `-t` option is critical. It creates tty for the container which will be used for accessing the shell inside.
2. The container is not yet ready to actually run or be worked with reliably. It needs to be initialized by Podman's `container init` command, by
   `$ podman container init NAME`
3. Now, you can run the shell inside the container and look around, do modifications, etc. Use Podman's `container start` with the `-a` (attach) and `-i` (interactive) options to enter its shell. Then, when you are done making modifications (do a quick modification), exit. This can be done by
   ```
   $ podman container start -a -i NAME
   $ WHATEVER_MODIFICATION_YOU_DO
   $ exit
   ```
4. Now, make a new container image from this modified container with whatever other `NAME2` and `TAG` you choose using Podman's `container commit` command, by
   `$ podman container commit NAME NAME2:TAG`. Note, the full address/name of this new container will be `localhost/NAME2:TAG`.

Further Reading

- `podman container create` – https://docs.podman.io/en/latest/markdown/podman-create.1.html
- `podman container init` – https://docs.podman.io/en/latest/markdown/podman-init.1.html
- `podman container start` – https://docs.podman.io/en/latest/markdown/podman-start.1.html
- `podman container commit` – https://docs.podman.io/en/latest/markdown/podman-commit.1.html

Run a container for a single command 5: Tutorial (1 min)

Steps

1. To run a container from a container image for a single command, use Podman's `container run` command with the container image's full location, name, tag or its `IMAGE ID`. We could use the `ls` command to see the contents of the container's `/bin` directory like
   ```
   $ podman container run docker.io/library/busybox:musl ls -lh /bin
   ```
2. If you want to get shell access or otherwise use an interactive program inside, you need to add the `-i` (interactive) and `-t` (tty) options (or simply `-it`). Try launching the shell and then exitting right away with `exit` like
   ```
   $ podman container run -it docker.io/library/busybox:musl /bin/sh
   $ exit
   ```
3. One potential issue with the above is that the created containers still exist and are not destroyed when the command is done. To have the container removed when it is done, you would add the `--rm` option like
   ```
   $ podman container run -it --rm docker.io/library/busybox:musl ls -lh /bin
   ```

Further Reading

- `podman container run` – https://docs.podman.io/en/latest/markdown/podman-run.1.html

Use a Dockerfile/Containerfile 6: Tutorial (2 min)

Steps

1. Build a new container image the same way you did in the previous tutorial, but with a `Dockerfile/Containerfil`
   First, write a `Dockerfile/Containerfile` like

   Listing 1: Dockerfile
   ```
   FROM docker.io/library/busybox:musl

   RUN WHATEVER_MODIFICATION_YOU_DO
   ```
   but remember, `RUN` directives can only run a single shell command. If you have more than one, they either need separate `RUN` directives or the subcommands need to be joined by `&&` to make a single larger command (often, each is put on its own line with `\` as a continuation marker)
2. Build it with Podman's `build` command, by
   `$ podman build --tag NAME2:TAG --file Dockerfile` Note, the full address/name of this new container will be `localhost/NAME2:TAG`.

Further Reading

- `podman build` – https://docs.podman.io/en/latest/markdown/podman-build.1.html