Docker

This article describes the status of CRIU integration with Docker, and how to use it.

Contents

Docker Experimental

Naturally, Docker wants to manage the full lifecycle of processes running inside its containers, so CRIU should be run by Docker (rather than separately). This feature is available in the *experimental* mode for Docker (since Docker 1.13, so every later version, like Docker 17.03, should work).

To enable experimental features (incl. CRIU), you need to do something like this:

```
echo "{₩"experimental₩": true}" >> /etc/docker/daemon.json
systemctl restart docker
```

In addition to having a recent version of Docker, you need **CRIU** 2.0 or later installed on your system (see Installation for more info).

checkpoint

There's a top level `checkpoint` sub-command in Docker, which lets you create a new checkpoint, and list or delete an existing checkpoint. These checkpoints are stored and managed by Docker, unless you specify a custom storage path.

Here's an example of creating a checkpoint, from a container that simply logs an integer in a loop.

First, we create container:

```
$ docker run -d --name looper --security-opt seccomp:unconfined busybox ₩
        /bin/sh -c 'i=0; while true; do echo $i; i=$(expr $i + 1); sleep 1; done'
```

You can verify the container is running by printings its logs:

```
$ docker logs looper
```

If you do this a few times you'll notice the integer increasing. Now, we checkpoint the container:

```
$ docker checkpoint create looper checkpoint1
```

You should see that the process is no longer running, and if you print the logs a few times no new logs will be printed.

restore

Unlike creating a checkpoint, restoring from a checkpoint is just a flag provided to the normal container **start** call. Here's an example:

```
$ docker start --checkpoint checkpoint1 looper
```

If we then print the logs, you should see they start from where we left off and continue to increase.

Restoring into a new container

Beyond the straightforward case of checkpointing and restoring the same container, it's also possible to checkpoint one container, and then restore the checkpoint into a completely different container. This is done by providing a custom storage path with the `--checkpoint-dir` option. Here's a slightly revised example from before:

```
$ docker run -d --name looper2 --security-opt seccomp:unconfined busybox ₩
        /bin/sh -c 'i=0; while true; do echo $i; i=$(expr $i + 1); sleep 1; done'

# wait a few seconds to give the container an opportunity to print a few lines, then
$ docker checkpoint create --checkpoint-dir=/tmp looper2 checkpoint2

$ docker create --name looper-clone --security-opt seccomp:unconfined busybox ₩
        /bin/sh -c 'i=0; while true; do echo $i; i=$(expr $i + 1); sleep 1; done'

$ docker start --checkpoint-dir=/tmp --checkpoint=checkpoint2 looper-clone
```

You should be able to print the logs from `looper-clone` and see that they start from wherever the logs of `looper` end.

Passing additional options

Configuration files can be used to set additional CRIU options when performing checkpoint/restore of Docker containers. These options should be added in the file `/etc/criu/runc.conf` (in order to **overwrite** the ones set by runc/Docker). Note that the options stored in `~/.criu/default.conf` or `/etc/criu/default.conf` will be **overwritten** by the ones set via RPC by Docker.

For example, in order to checkpoint and restore a container with established TCP connections CRIU requires the `--tcp-established` option to be set. However, this option is set to false by default and it is currently not possible to change this behaviour via the command-line interface of Docker. This feature can be enabled by adding `tcp-established` in the file `/etc/criu/runc.conf`. Note that for this functionality to work, the version of [runc (https://github.com/opencontainers/runc)] must be recent enough to have the commit [e157963 (https://github.com/opencontainers/runc/commit/e157963054e1be28bcd6612f15df1ea561c62571)] applied.

An alternative solution is to use Podman (https://podman.io/) which has support to specify `--tcp-established` on the command-line.

Synopsis

Checkpoint

```
# docker checkpoint create --help
Usage: docker checkpoint create [OPTIONS] CONTAINER CHECKPOINT
```

```
Create a checkpoint from a running container
```

```
Options:
    --checkpoint-dir string   Use a custom checkpoint storage directory
    --help                    Print usage
    --leave-running           Leave the container running after checkpoint
```

Restore

```
 # docker start --help
 Usage:        docker start [OPTIONS] CONTAINER [CONTAINER...]
```

```
Start one or more stopped containers
```

```
Options:
-a, --attach                  Attach STDOUT/STDERR and forward signals
    --checkpoint string       Restore from this checkpoint
    --checkpoint-dir string   Use a custom checkpoint storage directory
    --detach-keys string      Override the key sequence for detaching a container
    --help                    Print usage
-i, --interactive             Attach container's STDIN
```

Compatibility Notes

The latest versions of the Docker integration require at least version 2.0 of CRIU in order to work correctly. Additionally, depending on the storage driver being used by Docker, and other factors, there may be other compatibility issues that will attempt to be listed here.

TTY

Checkpointing an interactive container is supported by CRIU, runc and containerd, but not yet enabled in Docker. (See [PR 38405 (https://github.com/moby/moby/pull/38405)] for more information.)

Seccomp

You'll notice that all of the above examples disable Docker's default seccomp support. In order to use seccomp, you'll need a newer version of the Kernel. **Update Needed with Exact Version**

OverlayFS

There is a bug in OverlayFS that reports the wrong mnt_id in /proc/<pid>/fdinfo/<fd> and the wrong symlink target path for /proc/<pid>/<fd>. Fortunately, these bugs have been fixed in the kernel v4.2-rc2. The following small kernel patches fix the mount id and symlink target path issue:

- torvalds: 155e35d4da (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=155e35d4da) by David Howells
- torvalds: df1a085af1 (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=df1a085af1) by David Howells
- torvalds: f25801ee46 (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=f25801ee46) by David Howells
- torvalds: 4bacc9c923 (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=4bacc9c923) by David Howells
- torvalds: 9391dd00d1 (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=9391dd00d1) by Al Viro

Assuming that you are running Ubuntu Vivid (Linux kernel 3.19), here is how you can patch your kernel:

```
git clone  git://kernel.ubuntu.com/ubuntu/ubuntu-vivid.git
cd ubuntu-vivid
git remote add torvalds  git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
git remote update

git cherry-pick 155e35d4da
git cherry-pick df1a085af1
git cherry-pick f25801ee46
git cherry-pick 4bacc9c923
git cherry-pick 9391dd00d1

cp /boot/config-$(uname -r) .config
make olddefconfig
make -j 8 bzImage modules
sudo make install modules_install
sudo reboot
```

Async IO

If you are using a kernel older than 3.19 and your container uses AIO, you need the following AIO kernel patches from 3.19:

- torvalds: bd9b51e79c (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=bd9b51e79c) by Al Viro
- torvalds: e4a0d3e720 (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=e4a0d3e720) by Pavel Emelyanov

External checkpoint/restore

Although it's not recommended, you can also learn more about using CRIU without integrating with Docker. See Docker External for more info.
