南昌大学实验报告

姓名：Qing Liu

学号：6130116184

邮箱地址：1119637652@qq.com

专业班级：Class 164 of Computer Science and Technology

实验日期：THU. April 18th, 2019

课程名称：Cloud Computing Technology Experiments

实验项目名称

Live Migration

实验目的

- **Understanding the basic techniques for VM migration**
- **Using docker container as an example to test your migration skill**
- **Understanding the concept of checkpoint and restore**
- **Successfully migrate your image from one host to another**
- **Writing a decent report**

实验基础

- **Hardware: Lenovo Ideapad 700 - 15ISK**
- **Software: Vmware Workstation Pro, Ubuntu 18.04 LTS and Docker CE**

实验步骤

Configure Docker Experimental

- Edit the `/etc/docker/daemon.json` to add the configuration `"experimental": true`.

  ```
  $ sudo vim /etc/docker/daemon.json
  ```

  Content like this:

  ```
  {
     "registry-mirrors": ["http://f1361db2.m.daocloud.io"],
  ```

```
    "experimental": true
}
```

Reboot the docker.

```
$ systemctl restart docker.service
```

Install CRIU in OS

- Installing build dependencies

```
$ sudo apt-get install \
libprotobuf-dev \
libprotobuf-c0-dev \
protobuf-c-compiler \
protobuf-compiler \
python-protobuf

$ sudo apt-get --no-install-recommends install \
pkg-config python-ipaddress \
libbsd-dev iproute2 \
libcap-dev libnl-3-dev \
libnet-dev libaio-dev \
python3-future

$ sudo apt-get install asciidoc

$ sudo apt-get install xmlto
```

- Download the source package

```
$ wget http://download.openvz.org/criu/criu-3.11.tar.bz2
```

- Uncompress, compile and install

```
$ tar xvf criu-3.11.tar.bz2
$ cd criu-3.11
$ git init
$ make
$ sudo make install
```

- Checking that it works

```
$ sudo criu check
```

```
cleo@vm-ubuntu:~/Downloads/criu-3.11$ sudo criu check
Looks good.
cleo@vm-ubuntu:~/Downloads/criu-3.11$
```

Test docker checkpoint and restore

Creat a checkpoint and restore into the same container.

- 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'
```

```
cleo@vm-ubuntu:~$ 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'
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
fc1a6b909f82: Pull complete
Digest: sha256:f79f7a10302c402c052973e3fa42be0344ae6453245669783a9e16da3d56d5b4
Status: Downloaded newer image for busybox:latest
ebb659a080a3c8a6dce2865d961686ecc6a5e710abf25cfe570c7b379113e98a
```

- Verify the container is running by printings its logs

```
$ docker logs looper
```

```
63
64
65
66
67
68
69
cleo@vm-ubuntu:~$
```

```
88
89
90
91
92
93
cleo@vm-ubuntu:~$
```

- Checkpoint the container

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

```
cleo@vm-ubuntu:~$ docker checkpoint create looper checkpoint1
checkpoint1
cleo@vm-ubuntu:~$
```

execute `docker logs looper`, the result will be same.

```
138
139
140
141
142
143
cleo@vm-ubuntu:~$
```

```
138
139
140
141
142
143
cleo@vm-ubuntu:~$
```

- Restoring from a checkpoint

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

```
cleo@vm-ubuntu:~$ docker start --checkpoint checkpoint1 looper
cleo@vm-ubuntu:~$
```

execute `docker logs looper`, it will continue to output.

```
138
139
140
141
142
143
Warn  (criu/cr-service.c:290): parse_options returns 0
144
145
146
147
148
```

Creat a checkpoint and restore into the same container.

- Create container

```
$ 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'
```

```
cleo@vm-ubuntu:~$ 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'
00ee5ac63e4cda9b627abd0b0acd192c627950d390ef71b7a124cc2e30464126
```

- Checkpoint the container

```
$ docker checkpoint create --checkpoint-dir=/tmp looper2 checkpoint2
```

```
cleo@vm-ubuntu:~$ docker checkpoint create --checkpoint-dir=/tmp looper2 checkpoint2
checkpoint2
```

- Create another container

```
$ 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'
```

```
cleo@vm-ubuntu:~$ 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'
312803c7e38f21c39d73aa98204d3dfb596232939363e6287caebd969bc6bfc9
```

- Restore into the new container

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

```
cleo@vm-ubuntu:~$ docker start --checkpoint-dir=/tmp --checkpoint=checkpoint2 looper-clone
Error response from daemon: custom checkpointdir is not supported
```

From the output, we can see that error occurred. My Docker version is v18.09.5, whick remove the arguement `--checkpoint-dir`. It means criu is not supported for the latest version of Docker.

I found out a solution for it from https://github.com/moby/moby/issues/37344 :

rst0git commented on 2 Jan Contributor

Hi **@harishanand95**

The workaround would be to manually copy the checkpoint directory inside `/var/lib/docker/containers/<CONTAINER ID>/checkpoints/`

For example:

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

docker checkpoint create --checkpoint-dir=/tmp looper2 checkpoint2

mv /tmp/checkpoint2 /var/lib/docker/containers/$(docker ps -aq --no-trunc --filter name=looper2)/ch

docker start --checkpoint=checkpoint2 looper2
```

Here are my steps:

```
$ sudo mv /tmp/checkpoint2 /var/lib/docker/containers/$(docker ps \
-aq --no-trunc --filter name=looper-clone)/checkpoints/
$ docker start --checkpoint=checkpoint2 looper-clone
```

Build a docker image and create a docker repository

- **Create a repostory in Docker Hub**

Create Repository

cleo0625 getpath

This is my first golang repository in Docker Hub

Visibility

Using 0 of 1 private repositories. Get more

Public
Public repositories appear in Docker Hub search results

Private
Only you can view private repositories

Build Settings *(optional)*

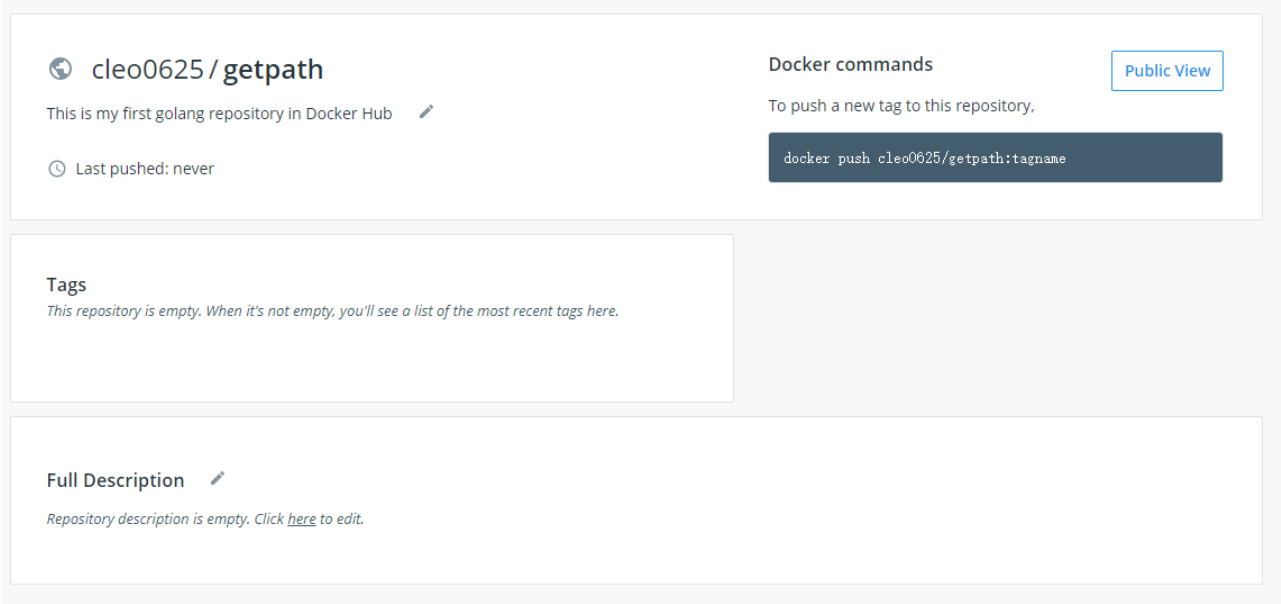
Autobuild triggers a new build with every **git push** to your source code repository. Learn More.

Please re-link a GitHub or Bitbucket account

We've updated how Docker Hub connects to GitHub and Bitbucket. You'll need to re-link a GitHub or Bitbucket account to create new automated builds. Learn More

Disconnected Disconnected

Cancel Create Create & Build



- **Login in**

```
$ docker login
```

```
cleo@vm-ubuntu:~/Code/go/app$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have
e one.
Username: cleo0625
Password:
WARNING! Your password will be stored unencrypted in /home/cleo/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
cleo@vm-ubuntu:~/Code/go/app$
```

- **Create directory `/home/cleo/Code/go/app`, and enter the direcroty, edit the source code `getpath.go` and `Dockerfile`**

```
$ mkdir -p /home/cleo/Code/go/app && cd /home/cleo/go/app
$ vim getpath.go
$ vim Dockerfile
```

`getpath.go`:

```go
package main

import (
    "fmt"
    "log"
    "net/http"
)
```

```
var count int

const headContent = `
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Qing Liu's Page</title>
</head>
<body>
  <h2 align="center">Qing Liu(6130116184)</h2>
  <h3 align="center">path:
`

const tailContent = `
  </h3>
</body>
</html>
`

func indexHandler(w http.ResponseWriter, r *http.Request) {
    path := r.URL.Path
    html := headContent + path + tailContent
    fmt.Fprintf(w, html)
    count++
    log.Printf("%d times to access.", count)
}

func main() {
    http.HandleFunc("/", indexHandler)
    http.ListenAndServe(":8000", nil)
}
```

`Dockerfile`:

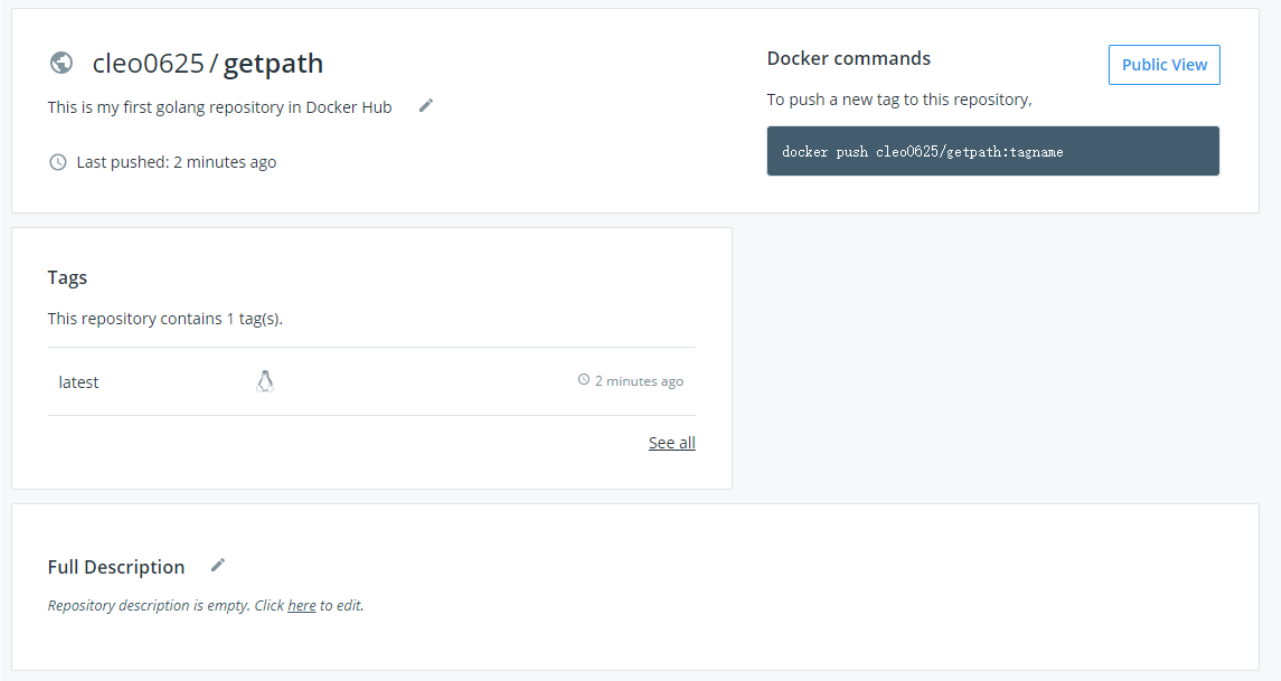
```
FROM golang:latest

WORKDIR /go/src/getpath
COPY . /go/src/getpath

RUN go build .

EXPOSE 8000

ENTRYPOINT ["./getpath"]
```

- **Build and push**

```
$ docker build -t cleo0625/getpath .
$ docker push cleo0625/getpath
```

```
cleo@vm-ubuntu:~/Code/go/app$ docker build -t cleo0625/getpath .
Sending build context to Docker daemon  3.584kB
Step 1/6 : FROM golang:latest
 ---> b860ab44e93e
Step 2/6 : WORKDIR /go/src/getpath
 ---> Running in 3ee4e70c1f3c
Removing intermediate container 3ee4e70c1f3c
 ---> 0bac56d6c748
Step 3/6 : COPY . /go/src/getpath
 ---> 704872bb2608
Step 4/6 : RUN go build .
 ---> Running in 725bf5a4cf16
Removing intermediate container 725bf5a4cf16
 ---> 2f35ad52b194
Step 5/6 : EXPOSE 8000
 ---> Running in aaf41c18b753
Removing intermediate container aaf41c18b753
 ---> 102cdf569842
Step 6/6 : ENTRYPOINT ["./getpath"]
 ---> Running in 9bf51f794751
Removing intermediate container 9bf51f794751
 ---> 12d50187f082
Successfully built 12d50187f082
Successfully tagged cleo0625/getpath:latest
cleo@vm-ubuntu:~/Code/go/app$
```

```
cleo@vm-ubuntu:~/Code/go/app$ docker push cleo0625/getpath
The push refers to repository [docker.io/cleo0625/getpath]
f7e4bf74b52d: Pushed
cdc1226ddc80: Pushed
0d8e786f1097: Pushed
39747431f79f: Mounted from library/golang
fcfab44ef5d3: Mounted from library/golang
f4907c4e3f89: Mounted from library/golang
b17cc31e431b: Mounted from library/golang
12cb127eee44: Mounted from library/golang
604829a174eb: Mounted from library/golang
fbb641a8b943: Mounted from library/golang
latest: digest: sha256:68b1d30b0ab13f877f128c3fbaa74ddbf84334520aa40403b548d9aadc996b93 size: 2421
cleo@vm-ubuntu:~/Code/go/app$
```




```
$ docker run --name qingliu -p 8000:8000 -d cleo0625/getpath
```

```
cleo@vm-ubuntu:~$ docker run --name qingliu -p 8000:8000 -d cleo0625/getpath
e822d518fdb2994f3ba86552aba588aa9410346015fd4915f736dc901388101e
```

We can test this container by execute:

```
$ curl localhost:8000
```

```
cleo@vm-ubuntu:~$ curl localhost:8000

<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>Qing Liu's Page</title>
</head>
<body>
        <h2 align="center">Qing Liu(6130116184)</h2>
        <h3 align="center">path:
/
        </h3>
</body>
</html>
cleo@vm-ubuntu:~$ curl localhost:8000

<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>Qing Liu's Page</title>
</head>
<body>
        <h2 align="center">Qing Liu(6130116184)</h2>
        <h3 align="center">path:
/
        </h3>
</body>
</html>
```

We can output the log of this container.

```
$ docker logs qingliu
```

```
cleo@vm-ubuntu:~$ docker logs qingliu
2019/04/20 13:11:12 1 times to access.
2019/04/20 13:11:15 2 times to access.
```

- **Make a checkpoint for other's restoring**

```
$ docker checkpoint create --checkpoint-dir=/tmp qingliu checkpoint1
```

```
cleo@vm-ubuntu:/tmp$ docker checkpoint create --checkpoint-dir=/tmp qingliu checkpoint1
checkpoint1
cleo@vm-ubuntu:/tmp$ ls
checkpoint1
```

After checkpoint, I pack up the directory into `checkpoint1.tar.gz` to send to my classmate by TIM.

```
$ cd /tmp
$ sudo tar -czvf checkpoint1.tar.gz checkpoint1
$ sudo cp checkpoint1.tar.gz ~/Downloads
```

```
cleo@vm-ubuntu:/tmp$ ls
checkpoint1
checkpoint1.tar.gz
config-err-UsfOOW
```

Restore the container of my classmate

- **Pull the of image of my classmate from Docker Hub**

```
$ docker pull zedididi/homework:v2
```

```
cleo@vm-ubuntu:~$ docker pull zedididi/homework:v2
v2: Pulling from zedididi/homework
e79bb959ec00: Already exists
d4b7902036fe: Already exists
1b2a72d4e030: Already exists
d54db43011fd: Already exists
963c818ebafc: Already exists
9eee6e7073aa: Already exists
83e75b35417b: Already exists
4d115857013b: Pull complete
fb729d4535e8: Pull complete
9ff584e62deb: Pull complete
Digest: sha256:6b2c5e68f247ba81839807fa099867796114fd630904ed5ea6f5d24e6dc01056
Status: Downloaded newer image for zedididi/homework:v2
```

- **Create a container for the image pulled from Docker Hub**

```
$ docker create --name zediliu-clone zedididi/homework:v2
```

```
cleo@vm-ubuntu:~$ docker create --name zediliu-clone zedididi/homework:v2
477491e96577c13573fd0f6f33cbb23cad3bb98618b37de9eb60fb299334cb49
```

- **Move the checkpoint to the new container's directory**

```
$ cd ~/Downloads/
$ ls
$ tar -xvf checkpoint1.tar.xz
$ sudo mv checkpoint1 /var/lib/docker/containers/$(docker ps \
-aq --no-trunc --filter name=zediliu-clone)/checkpoints/
```

```
cleo@vm-ubuntu:~/Downloads$ ls
checkpoint1  checkpoint1.tar.xz
```

```
cleo@vm-ubuntu:~/Downloads$ sudo mv checkpoint1 /var/lib/docker/containers/$(docker ps \
>   -aq --no-trunc --filter name=zediliu-clone)/checkpoints/
[sudo] password for cleo:
cleo@vm-ubuntu:~/Downloads$
```

- **Restore the container**

```
$ docker start --checkpoint=checkpoint1 zediliu-clone
```

```
cleo@vm-ubuntu:~/Downloads$ docker start --checkpoint=checkpoint1 zediliu-clone
cleo@vm-ubuntu:~/Downloads$
```

实验数据或结果

After restored Zedi Liu's container, I checked the status and logs of the restored container.

```
$ docker ps
```

```
cleo@vm-ubuntu:~/Downloads$ docker ps
CONTAINER ID        IMAGE                 COMMAND             CREATED             STATUS              PORTS               NAMES
477491e96577        zedididi/homework:v2  "./main"            8 minutes ago       Up 2 minutes                            zediliu-clone
```

We can see the container ID is `477491e96577`, the image is `zedididi/homework:v2`, names is `zediliu-clone` and the status is `Up 2 minutes`.

```
$ docker logs zediliu-clone
```

```
cleo@vm-ubuntu:~$ docker logs zediliu-clone
Warn  (criu/cr-service.c:290): parse_options returns 0
2019/04/21 02:48:26 this is  15
2019/04/21 02:48:31 this is  16
2019/04/21 02:48:36 this is  17
2019/04/21 02:48:41 this is  18
2019/04/21 02:48:46 this is  19
2019/04/21 02:48:51 this is  20
2019/04/21 02:48:56 this is  21
2019/04/21 02:49:01 this is  22
2019/04/21 02:49:06 this is  23
2019/04/21 02:49:11 this is  24
2019/04/21 02:49:16 this is  25
```

Here is the checkpoint of his origin container.

```
lzd@ubuntu:/etc/systemd/system$ docker rm `docker ps -a -q`
693e68c6e065
351336f33716
lzd@ubuntu:/etc/systemd/system$ docker run -d --name lzd1 lzd_homework
f685c8f88db642b34dda9aa16323bf5d100e181942dfe287a476095e1be29b91
lzd@ubuntu:/etc/systemd/system$ docker logs lzd1
2019/04/20 14:08:49 this is  0
2019/04/20 14:08:54 this is  1
2019/04/20 14:08:59 this is  2
lzd@ubuntu:/etc/systemd/system$ sudo docker checkpoint create lzd1 checkpoint1
checkpoint1
lzd@ubuntu:/etc/systemd/system$ docker logs lzd1
2019/04/20 14:08:49 this is  0
2019/04/20 14:08:54 this is  1
2019/04/20 14:08:59 this is  2
2019/04/20 14:09:04 this is  3
2019/04/20 14:09:09 this is  4
2019/04/20 14:09:14 this is  5
2019/04/20 14:09:19 this is  6
2019/04/20 14:09:24 this is  7
2019/04/20 14:09:29 this is  8
2019/04/20 14:09:34 this is  9
2019/04/20 14:09:39 this is  10
2019/04/20 14:09:44 this is  11
2019/04/20 14:09:49 this is  12
2019/04/20 14:09:54 this is  13
2019/04/20 14:09:59 this is  14
lzd@ubuntu:/etc/systemd/system$ docker rm $(docker ps -a -q)
f685c8f88db6
```

实验思考

At the beginning, I plan to build NFS to transport the checkpoint, but I give up it because of the virtual machines's network environment. I can't ping the ip address of destination in vm's OS between two different phycisical machines in two different LANs.

We decided to transport the checkpoint directory by TIM/QQ, and it must be pack up by `tar` or `zip`.

But I tried the NFS in the two virtual machines installed in my machines, because they both are within same LAN. And I finished the live migration successfully.

As for the live migration, its essence is to convert the state of the process into files, and then restore the previous state of the process from the file, which are the same as VM migration.

Some errors occurred sometimes:

> Error response from daemon: open /var/lib/docker/containers/[CONTAINER_ID]/checkpoints/[CHECKPOINT_ID]/config.json: no such file or directory

> Error response from daemon: failed to retrieve OCI runtime container pid: open /run/docker/containerd/daemon/io.containerd.runtime.v1.linux/moby/[CONTAINER_ID]/init.pid: no such file or directory: UNKOWN

It seems like that the new version of docker has bugs, but I use different program to test, it will get different results. I use a web application to checkpoint and restore, there are no problems. But others just print the number in loop every seconds, it can not restore successfully. So I think maybe it is related to the program or configrations.

参考资料

- http://cn.linux.vbird.org/linux_server/0330nfs.php
- https://criu.org/Docker
- https://docs.docker.com/docker-hub/

- https://github.com/checkpoint-restore/criu/issues/450
- https://github.com/moby/moby/issues
- https://github.com/checkpoint-restore/criu
- https://github.com/ZhuangweiKang/Docker-CRIU-Live-Migration?tdsourcetag=s_pctim_aiomsg