Kubernetes and Checkpoint Restore

Adrian Reber
Principal Software Engineer

2021, October 14

Agenda

Introduction

Use cases

Details

Definition:

Container Live Migration

Transfer Running Container

Serialize on Source System

Transfer to Destination System

Checkpoint/Restore in Userspace

CRIU

Multiple Integrations Exist

Container Live Migration

OpenVZ

Container Live Migration

Borg

Container Live Migration

LXC/LXD

Container Live Migration

Docker

Container Live Migration

Podman

Container Live Migration

CRI-O

`https://github.com/cri-o/cri-o/pull/4199`

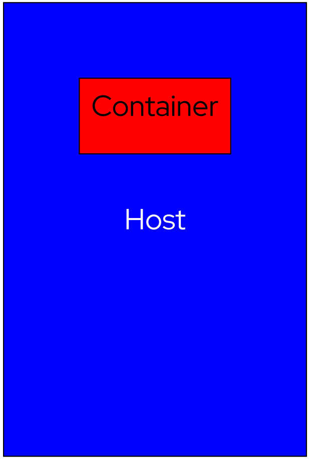
Container Live Migration

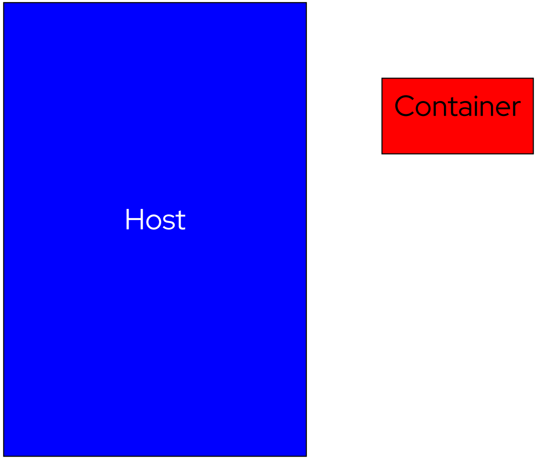
Kubernetes

`https://github.com/kubernetes/kubernetes/issues/3949`

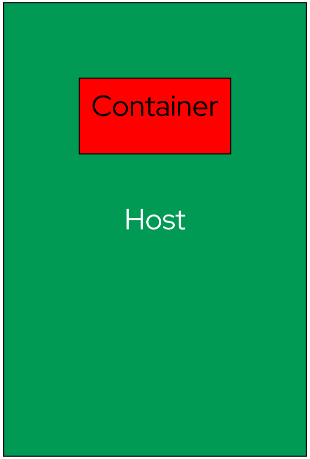
Use Cases

Reboot and Save State

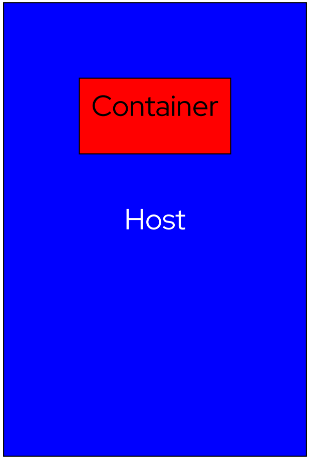

Container
Host

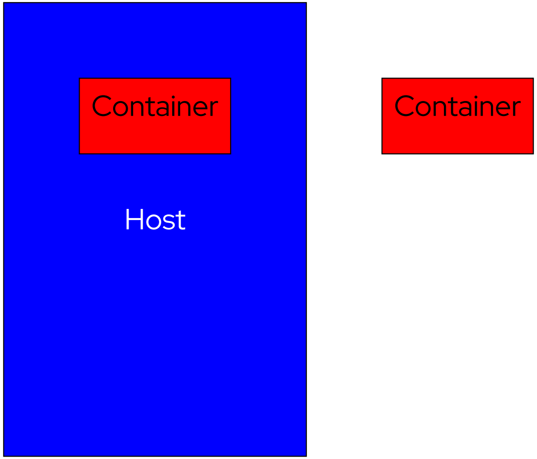

Host
Container

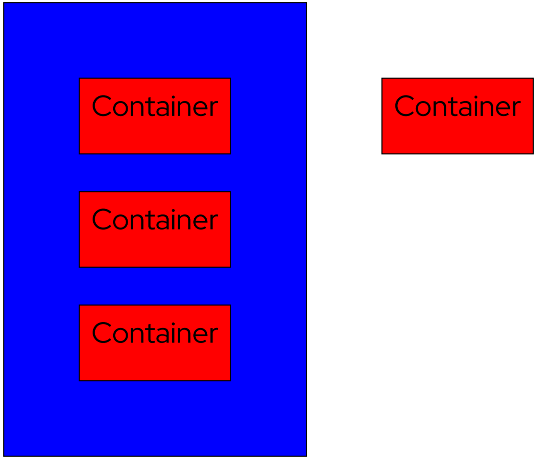
Container


Container
Host

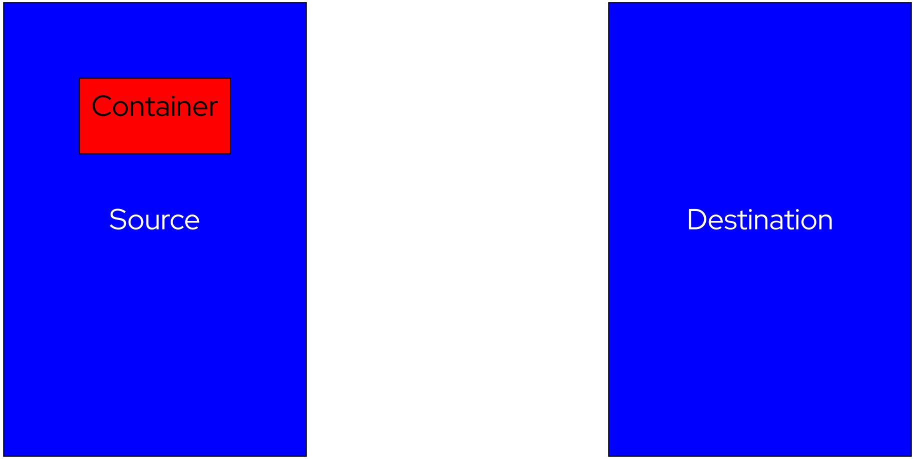
Quick Startup

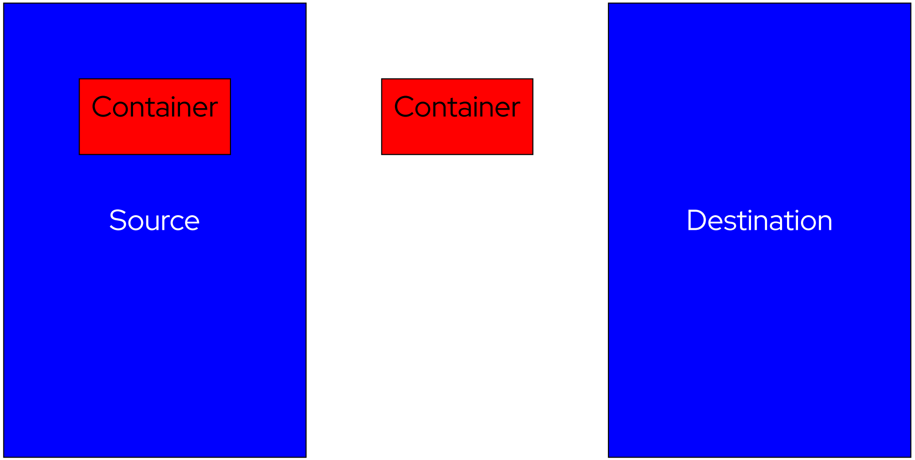

Container
Host

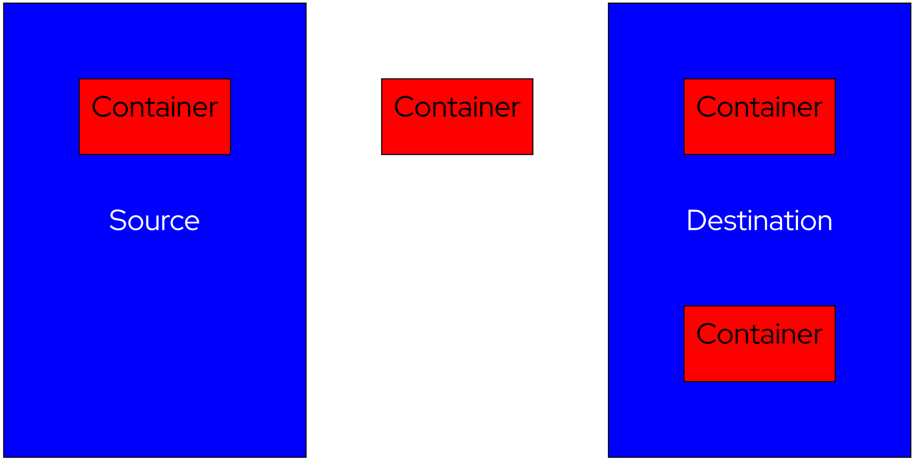

Container
Host
Container


Container
Container
Container
Container

Container Live Migration


Container
Source
Destination


Container
Source
Container
Destination


Container
Source
Container
Container
Destination
Container

Forensic Container Checkpointing

`https://github.com/kubernetes/enhancements/pull/1990`
`https://github.com/kubernetes/kubernetes/pull/104907`

CRIU

First Step: Checkpointing

Seize Process Using

`ptrace()`

Collect Details From

```
/proc/<PID>/*
```

Parasite Code

Parasite Code

Injected into the process

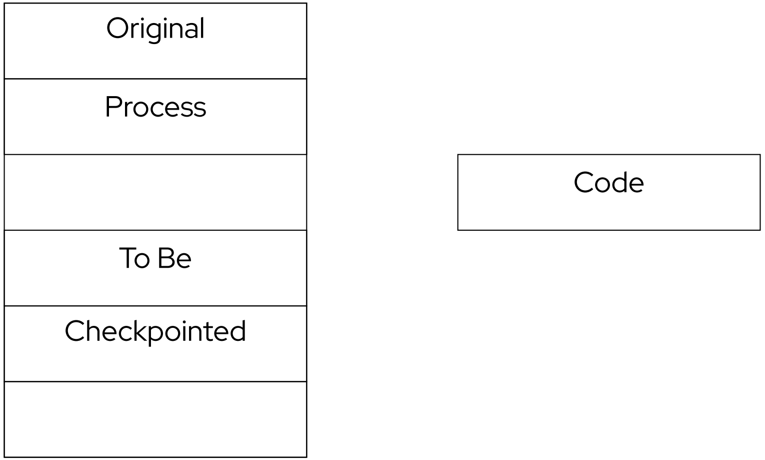
Parasite Code

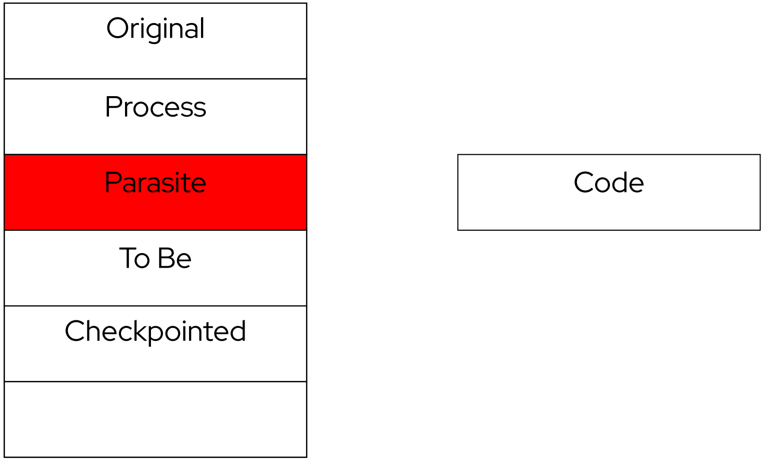
Daemon waiting for commands

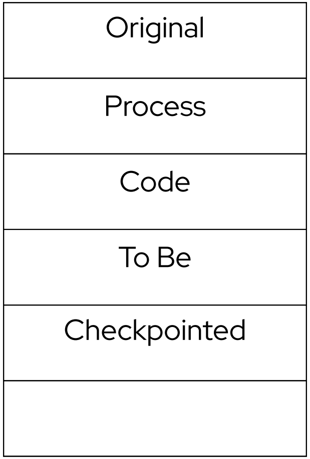
Parasite Code

Removed after usage

Original
Process
Code
To Be
Checkpointed


Original
Process
To Be
Checkpointed
Code


Original
Process
Parasite
To Be
Checkpointed
Code


Original
Process
Code
To Be
Checkpointed

Checkpointing Finished

Checkpointing Finished

All relevant information written

Checkpointing Finished

Target process is killed

Checkpointing Finished

Or continues to run

Second/Last Step: Restoring

Read Checkpoint Images

`clone()` For Each PID/TID

`clone3()` with Linux 5.5

CRIU Morphs Itself

Open and position file descriptors

CRIU Morphs Itself

Map memory pages

CRIU Morphs Itself

Load security settings

CRIU Morphs Itself

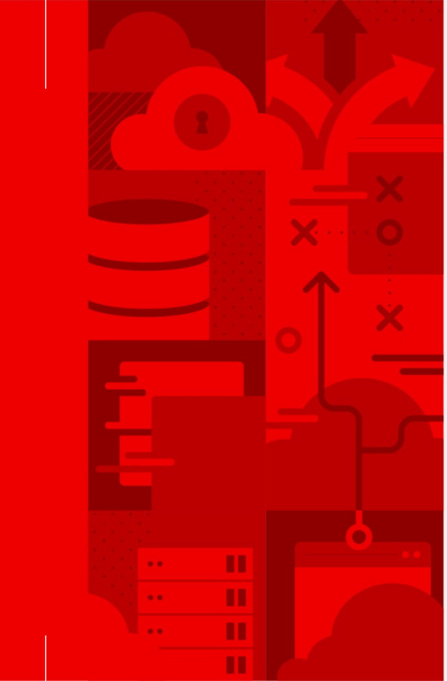
Jump into restored process

```
# podman run --rm -d adrianreber/wildfly-hello
699f33eb7fecbc5bbb00400be0aa79c888dbc63a54cac7bd2eed836a57d8a68a
# podman inspect -l --format "{{.NetworkSettings.IPAddress}}"
10.88.0.247
# curl 10.88.0.247:8080/helloworld/
0
# curl 10.88.0.247:8080/helloworld/
1
# podman container checkpoint -l --export=/tmp/chkpt.tar.gz
699f33eb7fecbc5bbb00400be0aa79c888dbc63a54cac7bd2eed836a57d8a68a
# scp /tmp/chkpt.tar.gz rhel08:/tmp
# podman container restore --import=/tmp/chkpt.tar.gz
699f33eb7fecbc5bbb00400be0aa79c888dbc63a54cac7bd2eed836a57d8a68a
# podman inspect -l --format "{{.NetworkSettings.IPAddress}}"
10.88.0.247
# curl 10.88.0.247:8080/helloworld/
2
# curl 10.88.0.247:8080/helloworld/
3
```

```
# podman container restore --import=/tmp/chkpt.tar.gz -n hello1
d02feeec894d77f66cc82484fe77ae369396a85f6d05594dc156c21e685942dd
# podman container restore --import=/tmp/chkpt.tar.gz -n hello2
735efb4fee6961d3eee069beb28dde5cbc6fc46c1a32a43ecc993d04c02015b2
# podman inspect --format "{{.NetworkSettings.IPAddress}}" hello1
10.88.0.248
# podman inspect --format "{{.NetworkSettings.IPAddress}}" hello2
10.88.0.249
# curl 10.88.0.248:8080/helloworld/
2
# curl 10.88.0.249:8080/helloworld/
2
```

Summary

- CRIU can checkpoint and restore containers
- Integrated in different containers engines
- Used in production
- Reboot into new kernel without losing container state
- Start multiple copies
- Migrate running containers
- Forensic container checkpointing (KEP #2008)



Thank you