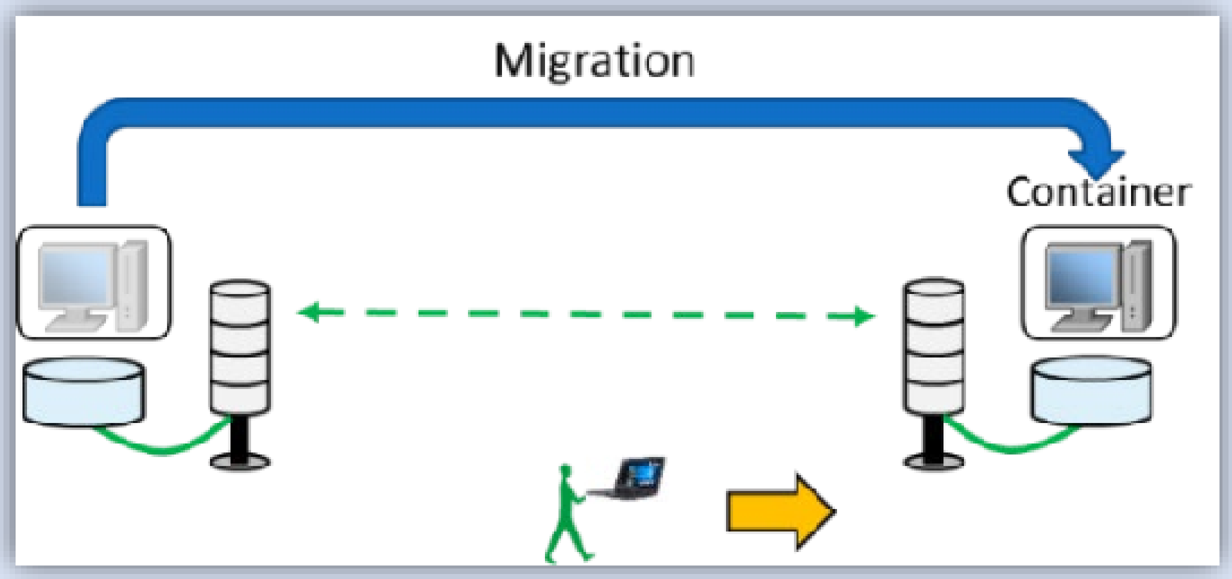
Live Migration & Checkpointing in Podman & Docker

Seamlessly Save, Transfer, and Restore Containers with Minimal Downtime



What is Checkpoint?

Checkpointing in **Podman and Docker** is a feature that allows you to **save the state of a running container** and restore it later. It uses **CRIU (Checkpoint/Restore in Userspace)** to freeze the container, store its state, and restart it when needed.

How It Works?

1. **Create a checkpoint** – The container's memory, process state, and network connections are saved.
2. **Export checkpoint** – The checkpoint files can be transferred to another system.
3. **Restore container** – The container resumes from the saved state, even on a different machine.

Use Cases

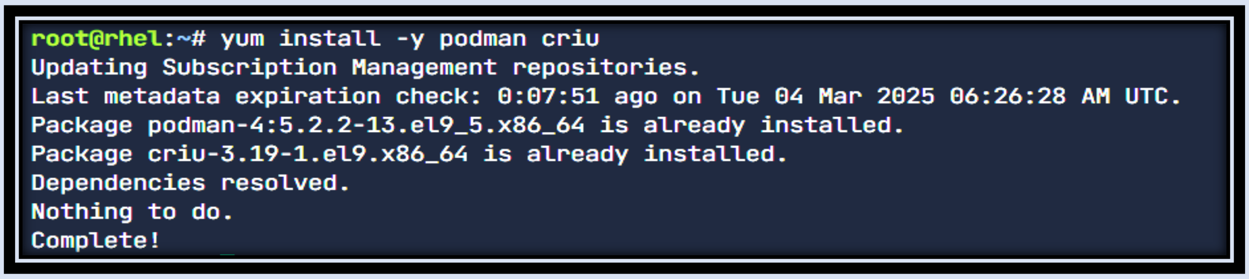
- **Live migration** – Move running containers across hosts without downtime.
- **Fault tolerance** – Restore failed containers to their last known state.
- **Reducing startup time** – Resume long-running processes instead of restarting from scratch.
- **Snapshot-based rollback** – Revert to a stable state if something goes wrong.

Advantages

✓ **Minimizes downtime** – Ideal for critical applications.
✓ **Improves flexibility** – Containers can be moved between hosts.
✓ **Enhances recovery** – Restores containers quickly after failures.
✓ **Optimizes resources** – Saves CPU/memory by suspending unused containers.

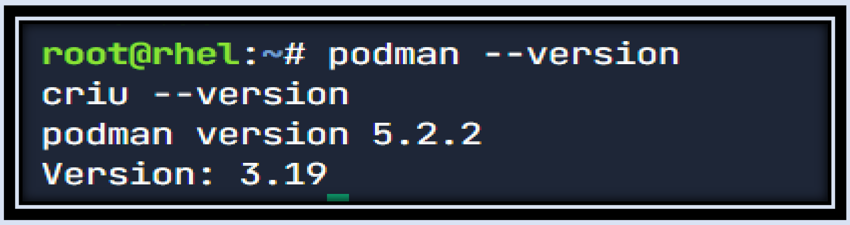
Step 1: Install Required Packages

- **Ensure you have Podman and CRIU installed:**
  - Yum install -y podman criu



```
root@rhel:~# yum install -y podman criu
Updating Subscription Management repositories.
Last metadata expiration check: 0:07:51 ago on Tue 04 Mar 2025 06:26:28 AM UTC.
Package podman-4:5.2.2-13.el9_5.x86_64 is already installed.
Package criu-3.19-1.el9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

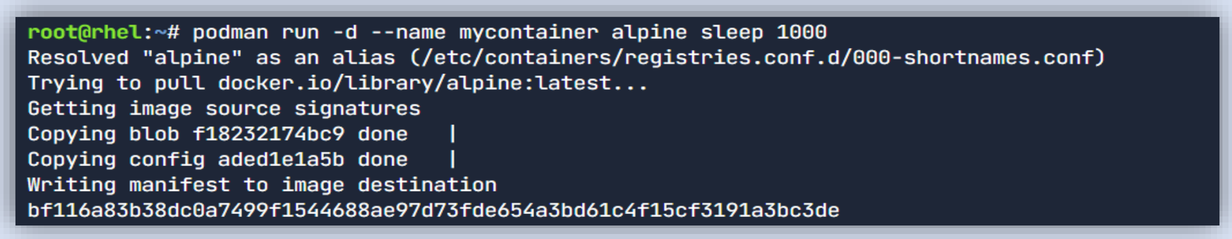
- **Verify installation:**
  - podman --version
  - criu --version



```
root@rhel:~# podman --version
criu --version
podman version 5.2.2
Version: 3.19
```

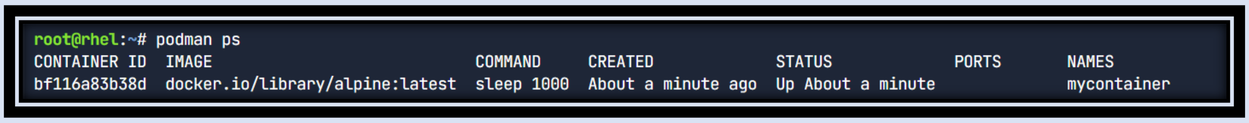
Step 2: Run a Container

- **Start a container that you want to checkpoint. Example:**
  - podman run -d --name mycontainer alpine sleep 1000

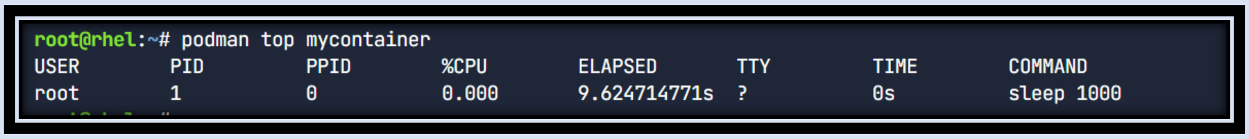


```
root@rhel:~# podman run -d --name mycontainer alpine sleep 1000
Resolved "alpine" as an alias (/etc/containers/registries.conf.d/000-shortnames.conf)
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Copying blob f18232174bc9 done   |
Copying config aded1e1a5b done   |
Writing manifest to image destination
bf116a83b38dc0a7499f1544688ae97d73fde654a3bd61c4f15cf3191a3bc3de
```

- **Check the running container:**
  - podman ps



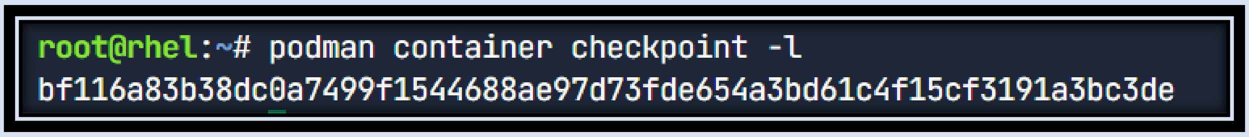
```
root@rhel:~# podman ps
CONTAINER ID  IMAGE                            COMMAND     CREATED             STATUS              PORTS       NAMES
bf116a83b38d  docker.io/library/alpine:latest  sleep 1000  About a minute ago  Up About a minute               mycontainer
```



```
root@rhel:~# podman top mycontainer
USER        PID         PPID        %CPU        ELAPSED         TTY         TIME        COMMAND
root        1           0           0.000       9.624714771s    ?           0s          sleep 1000
```

Step 3: Checkpoint the Container

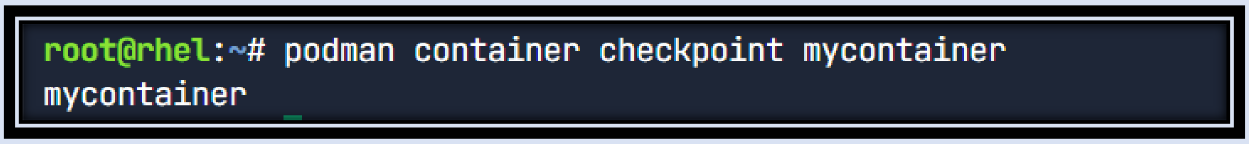
- **Save the container's state using:**
  - podman container checkpoint -l



```
root@rhel:~# podman container checkpoint -l
bf116a83b38dc0a7499f1544688ae97d73fde654a3bd61c4f15cf3191a3bc3de
```

OR

  - podman container checkpoint mycontainer



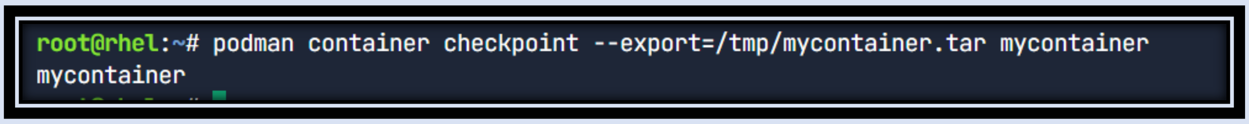
```
root@rhel:~# podman container checkpoint mycontainer
mycontainer
```

This creates checkpoint files in /var/lib/containers/storage/overlay-containers/<container_id>/userdata/checkpoint/.

- **To verify checkpoint creation:**
  - ls /var/lib/containers/storage/overlay-containers/$(podman inspect --format '{{.Id}}' mycontainer)/userdata/checkpoint/

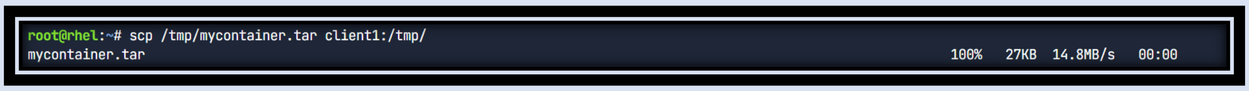
Step 4: Export the Checkpoint (Optional - For Migration)

- **To migrate a container to another system, export the checkpoint:**
  - podman container checkpoint --export=/tmp/mycontainer.tar mycontainer



```
root@rhel:~# podman container checkpoint --export=/tmp/mycontainer.tar mycontainer
mycontainer
```

- **Copy the checkpoint to another system (example using SCP):**
  - scp /tmp/mycontainer.tar user@remote_host:/tmp/



```
root@rhel:~# scp /tmp/mycontainer.tar client1:/tmp/
mycontainer.tar                                                                  100%   27KB  14.8MB/s   00:00
```

Step 5: Restore the Container

A) Restore on the Same System

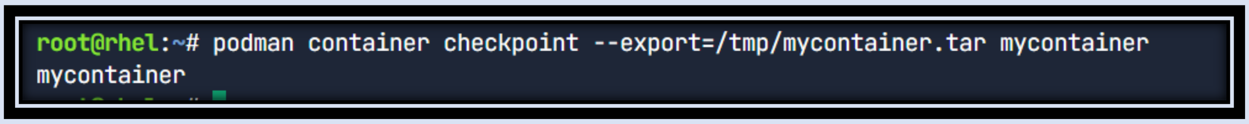
- **To restore the container:**
  - podman container restore -l

OR

- podman container restore mycontainer

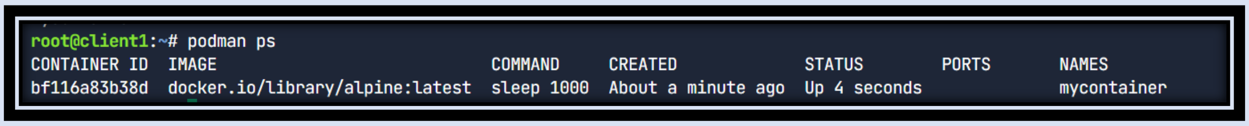
B) Restore on a Different System (After Importing Checkpoint)

- **On the new system, import the checkpoint:**
  - podman container restore --import=/tmp/mycontainer.tar

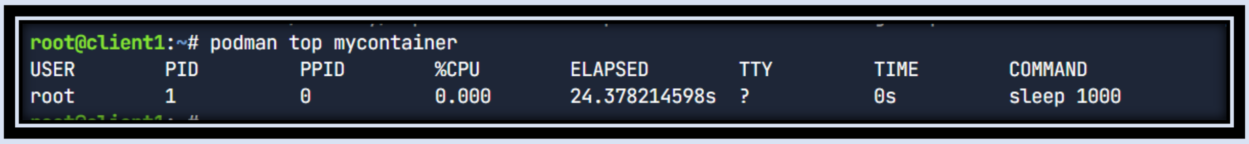


```
root@rhel:~# podman container checkpoint --export=/tmp/mycontainer.tar mycontainer
mycontainer
```

- **Check if the container is running again:**
  - podman ps



```
root@client1:~# podman ps
CONTAINER ID  IMAGE                            COMMAND     CREATED             STATUS        PORTS       NAMES
bf116a83b38d  docker.io/library/alpine:latest  sleep 1000  About a minute ago  Up 4 seconds              mycontainer
```

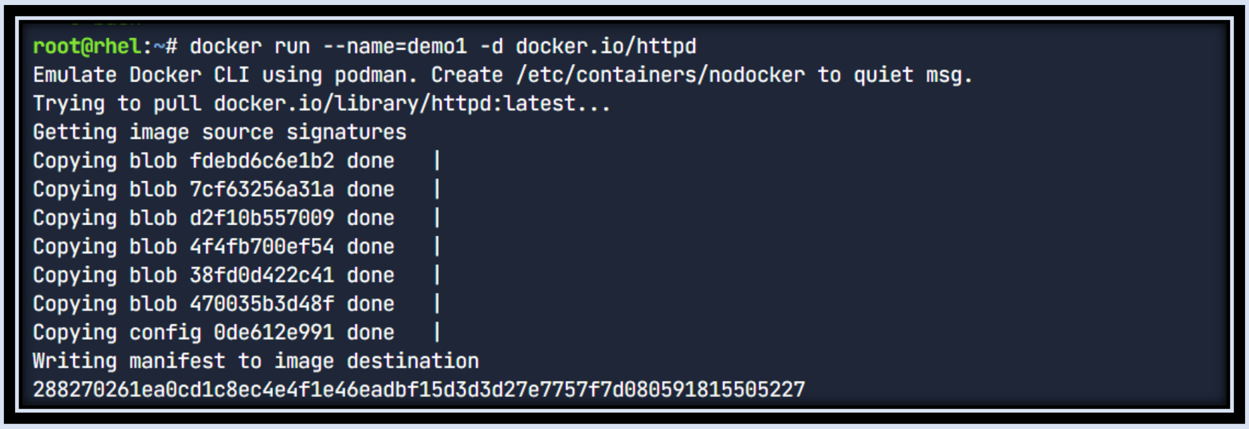


```
root@client1:~# podman top mycontainer
USER        PID         PPID        %CPU        ELAPSED         TTY         TIME        COMMAND
root        1           0           0.000       24.378214598s   ?           0s          sleep 1000
```

Docker

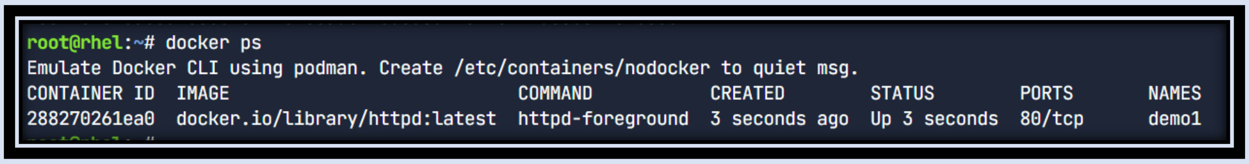
Step 1: Run a Container

- **Start a test container:**
  - docker run -d --name mygame sidhu1504/snake-game:latest

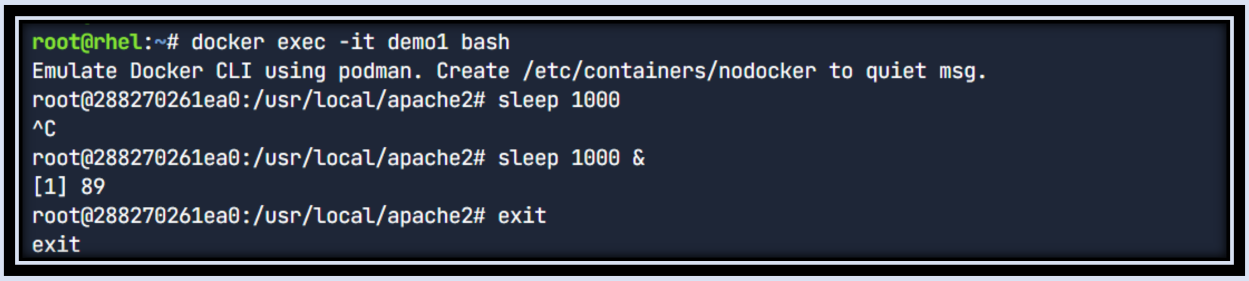


```
root@rhel:~# docker run --name=demo1 -d docker.io/httpd
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
Trying to pull docker.io/library/httpd:latest...
Getting image source signatures
Copying blob fdebd6c6e1b2 done   |
Copying blob 7cf63256a31a done   |
Copying blob d2f10b557009 done   |
Copying blob 4f4fb700ef54 done   |
Copying blob 38fd0d422c41 done   |
Copying blob 470035b3d48f done   |
Copying config 0de612e991 done   |
Writing manifest to image destination
288270261ea0cd1c8ec4e4f1e46eadbf15d3d3d27e7757f7d080591815505227
```

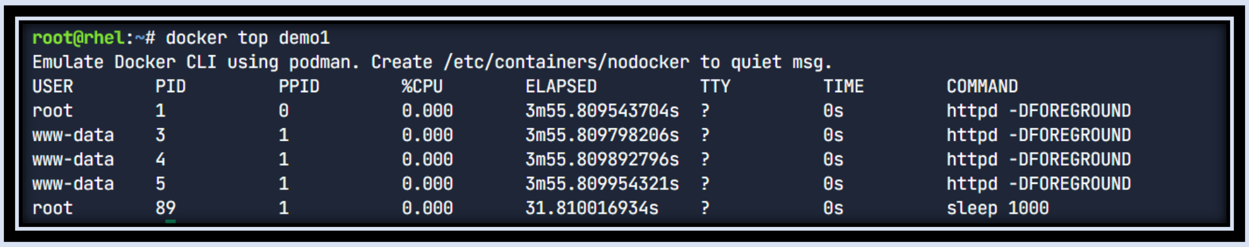
- **Check the running container:**
  - docker ps



```
root@rhel:~# docker ps
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
CONTAINER ID  IMAGE                          COMMAND           CREATED        STATUS        PORTS      NAMES
288270261ea0  docker.io/library/httpd:latest httpd-foreground  3 seconds ago  Up 3 seconds  80/tcp     demo1
```



```
root@rhel:~# docker exec -it demo1 bash
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
root@288270261ea0:/usr/local/apache2# sleep 1000
^C
root@288270261ea0:/usr/local/apache2# sleep 1000 &
[1] 89
root@288270261ea0:/usr/local/apache2# exit
exit
```



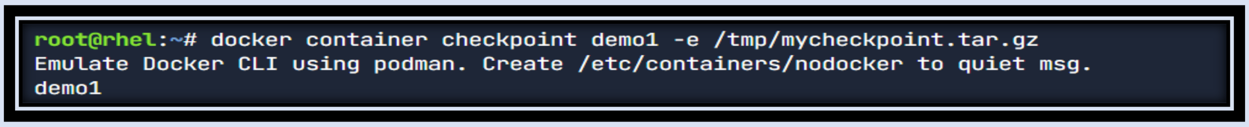
```
root@rhel:~# docker top demo1
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
USER       PID        PPID       %CPU       ELAPSED           TTY        TIME       COMMAND
root       1          0          0.000      3m55.809543704s   ?          0s         httpd -DFOREGROUND
www-data   3          1          0.000      3m55.809798206s   ?          0s         httpd -DFOREGROUND
www-data   4          1          0.000      3m55.809892796s   ?          0s         httpd -DFOREGROUND
www-data   5          1          0.000      3m55.809954321s   ?          0s         httpd -DFOREGROUND
root       89         1          0.000      31.810016934s     ?          0s         sleep 1000
```

Step 2: Checkpoint the Container

- **Save the container's state:**
  - docker checkpoint create mygame checkpoint1
- **To verify checkpoint creation:**
  - ls /var/lib/docker/containers/$(docker inspect --format '{{.Id}}' mygame)/checkpoints/

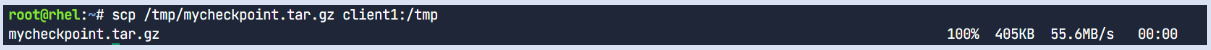
Step 3: Export the Checkpoint (Optional - For Migration)

- **To migrate the checkpoint to another system, export it:**
  - docker checkpoint create --checkpoint-dir=/tmp mygame checkpoint1



```
root@rhel:~# docker container checkpoint demo1 -e /tmp/mycheckpoint.tar.gz
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
demo1
```

- **Copy the checkpoint to another system using scp:**
  - scp -r /tmp/checkpoint1 user@remote_host:/tmp/



```
root@rhel:~# scp /tmp/mycheckpoint.tar.gz client1:/tmp
mycheckpoint.tar.gz                                                     100%  405KB  55.6MB/s   00:00
```

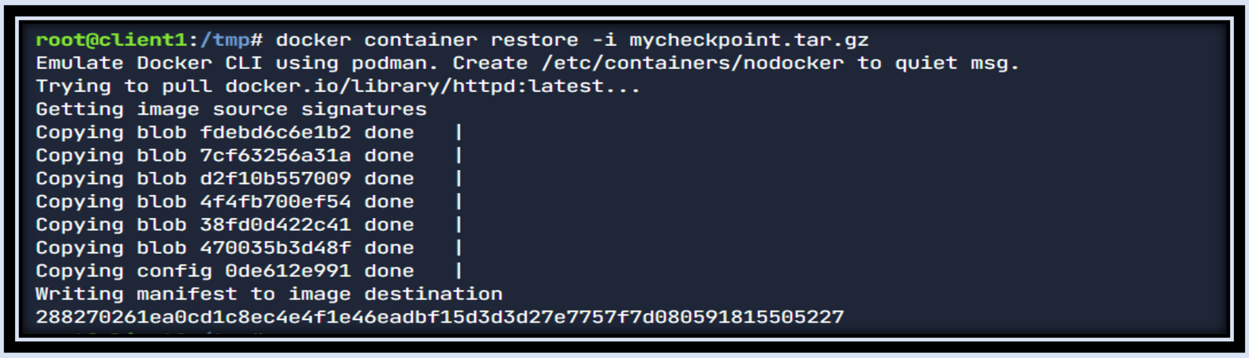
Step 6: Restore the Container

A) Restore on the Same System

- **To restore the container:**
  - docker start --checkpoint checkpoint1 mygame

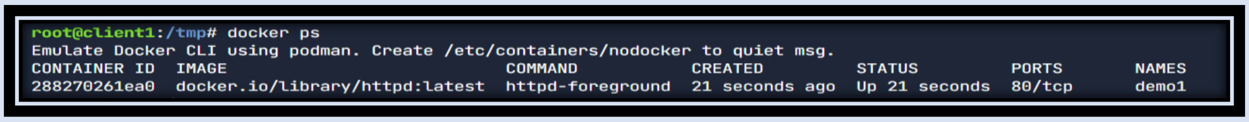
B) Restore on a Different System (After Importing Checkpoint)

- **On the new system, import the checkpoint and restore:**
  - docker start --checkpoint-dir=/tmp --checkpoint checkpoint1 mygame

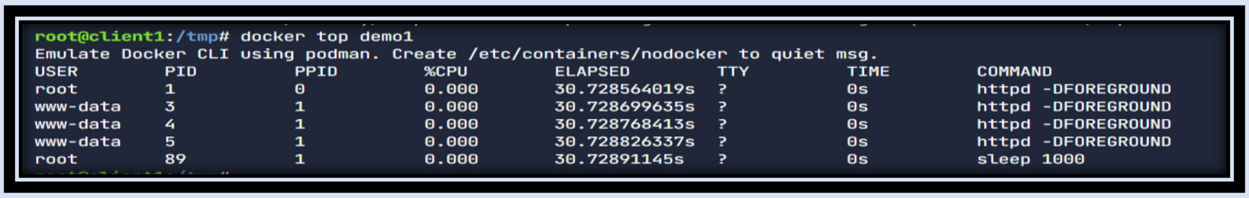


```
root@client1:/tmp# docker container restore -i mycheckpoint.tar.gz
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
Trying to pull docker.io/library/httpd:latest...
Getting image source signatures
Copying blob fdebd6c6e1b2 done   |
Copying blob 7cf63256a31a done   |
Copying blob d2f10b557009 done   |
Copying blob 4f4fb700ef54 done   |
Copying blob 38fd0d422c41 done   |
Copying blob 470035b3d48f done   |
Copying config 0de612e991 done   |
Writing manifest to image destination
288270261ea0cd1c8ec4e4f1e46eadbf15d3d3d27e7757f7d080591815505227
```

- **Check if the container is running:**
  - docker ps



```
root@client1:/tmp# docker ps
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
CONTAINER ID  IMAGE                          COMMAND           CREATED         STATUS         PORTS      NAMES
288270261ea0  docker.io/library/httpd:latest  httpd-foreground  21 seconds ago  Up 21 seconds  80/tcp     demo1
```



```
root@client1:/tmp# docker top demo1
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
USER        PID         PPID        %CPU        ELAPSED           TTY         TIME        COMMAND
root        1           0           0.000       30.728564019s     ?           0s          httpd -DFOREGROUND
www-data    3           1           0.000       30.728699635s     ?           0s          httpd -DFOREGROUND
www-data    4           1           0.000       30.728768413s     ?           0s          httpd -DFOREGROUND
www-data    5           1           0.000       30.728826337s     ?           0s          httpd -DFOREGROUND
root        89          1           0.000       30.72891145s      ?           0s          sleep 1000
```