Full credit is given to the above companies including the OS that this PDF file was generated!

Rocky Enterprise Linux 9.2 Manual Pages on command 'podman-container-clone.1'

$ man podman-container-clone.1

podman-container-clone(1)   General Commands Manual podman-container-clone(1)

NAME

podman-container-clone - Creates a copy of an existing container

SYNOPSIS

podman container clone [options] container name image

DESCRIPTION

podman container clone creates a copy of a container, recreating the original with an identical configuration. This command takes three ar? guments: the first being the container id or name to clone, the second argument in this command can change the name of the clone from the de? fault of $ORIGINAL_NAME-clone, and the third is a new image to use in the cloned container.

OPTIONS

--blkio-weight=weight

Block IO relative weight. The weight is a value between 10 and 1000.

This option is not supported on cgroups V1 rootless systems.

--blkio-weight-device=device:weight

Block IO relative device weight.

--cpu-period=limit

Set the CPU period for the Completely Fair Scheduler (CFS), which is a duration in microseconds. Once the container's CPU quota is used up, it will not be scheduled to run until the current period ends. Defaults to 100000 microseconds.

On some systems, changing the resource limits may not be allowed for non-root users. For more details, see https://github.com/contain?
ers/podman/blob/main/troubleshooting.md#26-running-containers-with-re?
source-limits-fails-with-a-permissions-error

This option is not supported on cgroups V1 rootless systems.

If none is specified, the original container's cpu period is used

--cpu-quota=limit

Limit the CPU Completely Fair Scheduler (CFS) quota.

Limit the container's CPU usage. By default, containers run with the full CPU resource. The limit is a number in microseconds. If a number is provided, the container will be allowed to use that much CPU time until the CPU period ends (controllable via --cpu-period).

On some systems, changing the resource limits may not be allowed for non-root users. For more details, see https://github.com/contain?
ers/podman/blob/main/troubleshooting.md#26-running-containers-with-re?
source-limits-fails-with-a-permissions-error

This option is not supported on cgroups V1 rootless systems.

If none is specified, the original container's CPU quota are used.

--cpu-rt-period=microseconds

Limit the CPU real-time period in microseconds.

Limit the container's Real Time CPU usage. This option tells the kernel to restrict the container's Real Time CPU usage to the period speci?
fied.

This option is only supported on cgroups V1 rootful systems.

If none is specified, the original container's CPU runtime period is used.

--cpu-rt-runtime=microseconds

Limit the CPU real-time runtime in microseconds.

Limit the containers Real Time CPU usage. This option tells the kernel
to limit the amount of time in a given CPU period Real Time tasks may
consume. Ex: Period of 1,000,000us and Runtime of 950,000us means that
this container could consume 95% of available CPU and leave the remain?
ing 5% to normal priority tasks.

The sum of all runtimes across containers cannot exceed the amount al?
lotted to the parent cgroup.

This option is only supported on cgroups V1 rootful systems.

--cpu-shares, -c=shares

CPU shares (relative weight).

By default, all containers get the same proportion of CPU cycles. This
proportion can be modified by changing the container's CPU share
weighting relative to the combined weight of all the running contain?
ers. Default weight is 1024.

The proportion will only apply when CPU-intensive processes are run?
ning. When tasks in one container are idle, other containers can use
the left-over CPU time. The actual amount of CPU time will vary depend?
ing on the number of containers running on the system.

For example, consider three containers, one has a cpu-share of 1024 and
two others have a cpu-share setting of 512. When processes in all three
containers attempt to use 100% of CPU, the first container would re?
ceive 50% of the total CPU time. If a fourth container is added with a
cpu-share of 1024, the first container only gets 33% of the CPU. The
remaining containers receive 16.5%, 16.5% and 33% of the CPU.

On a multi-core system, the shares of CPU time are distributed over all
CPU cores. Even if a container is limited to less than 100% of CPU
time, it can use 100% of each individual CPU core.

For example, consider a system with more than three cores. If the con?
tainer C0 is started with --cpu-shares=512 running one process, and an?
other container C1 with --cpu-shares=1024 running two processes, this
can result in the following division of CPU shares:

```
????????????????????????????????????????
?PID ? container ? CPU ? CPU share    ?
```

```
??????????????????????????????????????
?100 ? C0       ? 0   ? 100% of CPU0 ?
??????????????????????????????????????
?101 ? C1       ? 1   ? 100% of CPU1 ?
??????????????????????????????????????
?102 ? C1       ? 2   ? 100% of CPU2 ?
??????????????????????????????????????
```

On some systems, changing the resource limits may not be allowed for
non-root users. For more details, see https://github.com/contain?
ers/podman/blob/main/troubleshooting.md#26-running-containers-with-re?
source-limits-fails-with-a-permissions-error

This option is not supported on cgroups V1 rootless systems.

If none are specified, the original container's CPU shares are used.

--cpus

Set a number of CPUs for the container that overrides the original con?
tainers CPU limits. If none are specified, the original container's
Nano CPUs are used.

This is shorthand for --cpu-period and --cpu-quota, so only --cpus or
either both the --cpu-period and --cpu-quota options can be set.

This option is not supported on cgroups V1 rootless systems.

--cpuset-cpus=number

CPUs in which to allow execution. Can be specified as a comma-separated
list (e.g. 0,1), as a range (e.g. 0-3), or any combination thereof
(e.g. 0-3,7,11-15).

On some systems, changing the resource limits may not be allowed for
non-root users. For more details, see https://github.com/contain?
ers/podman/blob/main/troubleshooting.md#26-running-containers-with-re?
source-limits-fails-with-a-permissions-error

This option is not supported on cgroups V1 rootless systems.

If none are specified, the original container's CPUset is used.

--cpuset-mems=nodes

Memory nodes (MEMs) in which to allow execution (0-3, 0,1). Only effec?
tive on NUMA systems.

If there are four memory nodes on the system (0-3), use --cpuset-mems=0,1 then processes in the container will only use memory from the first two memory nodes.

On some systems, changing the resource limits may not be allowed for non-root users. For more details, see https://github.com/contain?ers/podman/blob/main/troubleshooting.md#26-running-containers-with-re?source-limits-fails-with-a-permissions-error

This option is not supported on cgroups V1 rootless systems.

If none are specified, the original container's CPU memory nodes are used.

--destroy

Remove the original container that we are cloning once used to mimic the configuration.

--device-read-bps=path:rate

Limit read rate (in bytes per second) from a device (e.g. --device-read-bps=/dev/sda:1mb).

On some systems, changing the resource limits may not be allowed for non-root users. For more details, see https://github.com/contain?ers/podman/blob/main/troubleshooting.md#26-running-containers-with-re?source-limits-fails-with-a-permissions-error

This option is not supported on cgroups V1 rootless systems.

--device-write-bps=path:rate

Limit write rate (in bytes per second) to a device (e.g. --device-write-bps=/dev/sda:1mb).

On some systems, changing the resource limits may not be allowed for non-root users. For more details, see https://github.com/contain?ers/podman/blob/main/troubleshooting.md#26-running-containers-with-re?source-limits-fails-with-a-permissions-error

This option is not supported on cgroups V1 rootless systems.

--force, -f

Force removal of the original container that we are cloning. Can only be used in conjunction with --destroy.

--memory, -m=number[unit]

Memory limit. A unit can be b (bytes), k (kibibytes), m (mebibytes), or g (gibibytes).

Allows the memory available to a container to be constrained. If the host supports swap memory, then the -m memory setting can be larger than physical RAM. If a limit of 0 is specified (not using -m), the container's memory is not limited. The actual limit may be rounded up to a multiple of the operating system's page size (the value would be very large, that's millions of trillions).

This option is not supported on cgroups V1 rootless systems.

If no memory limits are specified, the original container's will be used.

--memory-reservation=number[unit]

Memory soft limit. A unit can be b (bytes), k (kibibytes), m (mebibytes), or g (gibibytes).

After setting memory reservation, when the system detects memory con?tention or low memory, containers are forced to restrict their consump?tion to their reservation. So always set the value below --memory, oth?erwise the hard limit will take precedence. By default, memory reserva?tion will be the same as memory limit.

This option is not supported on cgroups V1 rootless systems.

If unspecified, memory reservation will be the same as memory limit from the container being cloned.

--memory-swap=number[unit]

A limit value equal to memory plus swap. A unit can be b (bytes), k (kibibytes), m (mebibytes), or g (gibibytes).

Must be used with the -m (--memory) flag. The argument value should always be larger than that of

-m (--memory) By default, it is set to double the value of --memory.

Set number to -1 to enable unlimited swap.

This option is not supported on cgroups V1 rootless systems.

If unspecified, the container being cloned will be used to derive the swap value.

--memory-swappiness=number

Tune a container's memory swappiness behavior. Accepts an integer be?
tween 0 and 100.

This flag is only supported on cgroups V1 rootful systems.

--name

Set a custom name for the cloned container. The default if not speci?
fied is of the syntax: -clone

--pod=name

Clone the container in an existing pod. It is helpful to move a con?
tainer to an existing pod.   The container will join the pod shared
namespaces, losing its configuration that conflicts with the shared
namespaces.

--run

When set to true, this flag runs the newly created container after the
clone process has completed, this specifies a detached running mode.

EXAMPLES

```
# podman container clone d0cf1f782e2ed67e8c0050ff92df865a039186237a4df24d7acba5b1fa8cc6e7
6b2c73ff8a1982828c9ae2092954bcd59836a131960f7e05221af9df5939c584
# podman container clone --name=clone
d0cf1f782e2ed67e8c0050ff92df865a039186237a4df24d7acba5b1fa8cc6e7
6b2c73ff8a1982828c9ae2092954bcd59836a131960f7e05221af9df5939c584
# podman container clone --destroy --cpus=5
d0cf1f782e2ed67e8c0050ff92df865a039186237a4df24d7acba5b1fa8cc6e7
6b2c73ff8a1982828c9ae2092954bcd59836a131960f7e05221af9df5939c584
# podman container clone 2d4d4fca7219b4437e0d74fcdc272c4f031426a6eacd207372691207079551de
new_name fedora
Resolved "fedora" as an alias (/etc/containers/registries.conf.d/shortnames.conf)
Trying to pull registry.fedoraproject.org/fedora:latest...
Getting image source signatures
Copying blob c6183d119aa8 done
Copying config e417cd49a8 done
Writing manifest to image destination
Storing signatures
5a9b7851013d326aa4ac4565726765901b3ecc01fcbc0f237bc7fd95588a24f9
```

SEE ALSO

podman-create(1), cgroups(7)

HISTORY

January 2022, Originally written by Charlie Doern cdoern@redhat.com

?mailto:cdoern@redhat.com?