NAME

docker-container-create - Create a new container

SYNOPSIS

docker container create [OPTIONS] IMAGE [COMMAND] [ARG...]

DESCRIPTION

Creates a writeable container layer over the specified image and prepares it for running the specified command. The container ID is then printed to STDOUT. This is similar to **docker run -d** except the container is never started. You can then use the **docker start** command to start the container at any point.

The initial status of the container created with **docker create** is 'created'.

OPTIONS

The `CONTAINER-DIR` must be an absolute path such as `/src/docs`. The `HOST-DIR` can be an absolute path or a `name` value. A `name` value must start with an alphanumeric character, followed by `a-z0-9`, _ (underscore), `.` (period) or - (hyphen). An absolute path starts with a / (forward slash).

If you supply a `HOST-DIR` that is an absolute path, Docker bind-mounts to the path you specify. If you supply a `name`, Docker creates a named volume by that `name`. For example, you can specify either `/foo` or `foo` for a `HOST-DIR` value. If you supply the `/foo` value, Docker creates a bind mount. If you supply the `foo` specification, Docker creates a named volume.

You can specify multiple **-v** options to mount one or more mounts to a container. To use these same mounts in other containers, specify the **--volumes-from** option also.

You can supply additional options for each bind mount following an additional colon. A `:ro` or `:rw` suffix mounts a volume in read-only or read-write mode, respectively. By default, volumes are mounted in read-write mode.
You can also specify the consistency requirement for the mount, either `:consistent` (the default), `:cached`, or `:delegated`. Multiple options are separated by commas, e.g. `:ro,cached`.

Labeling systems like SELinux require that proper labels are placed on volume content mounted into a container. Without a label, the security system might prevent the processes running inside the container from using the content. By default, Docker does not change the labels set by the OS.

To change a label in the container context, you can add either of two suffixes `:z` or `:Z` to the volume mount. These suffixes tell Docker to relabel file objects on the shared volumes. The `z` option tells Docker that two containers share the volume content. As a result, Docker labels the content with a shared content label. Shared volume labels allow all containers to read/write content. The `Z` option tells Docker to label the content with a private unshared label. Only the current container can use a private volume.

By default bind mounted volumes are `private`. That means any mounts done inside container will not be visible on host and vice-a-versa. One can change this behavior by specifying a volume mount propagation property. Making a volume `shared` mounts done under that volume inside container will be visible on host and vice-a-versa. Making a volume `slave` enables only one way mount propagation and that is mounts done on host under that volume will be visible inside container but not the other way around.

To control mount propagation property of volume one can use `:[r]shared`, `:[r]slave` or `:[r]private` propagation flag. Propagation property can be specified only for bind mounted volumes and not for internal volumes or named volumes. For mount propagation to work source mount point (mount point where source dir is mounted on) has to have right propagation properties. For shared volumes, source mount point has to be shared. And for slave volumes, source mount has to be either shared or slave.

Use `df <source-dir>` to figure out the source mount and then use `findmnt -o TARGET,PROPAGATION <source-mount-dir>` to figure out propagation properties of source mount. If `findmnt` utility is not available, then one can look at mount entry for source mount point in `/proc/self/mountinfo`. Look at `optional fields` and see if any propagation properties are specified. `shared:X` means mount is `shared`, `master:X` means mount is `slave` and if nothing is there that means mount is `private`.

To change propagation properties of a mount point use `mount` command. For example, if one wants to bind mount source directory `/foo` one can do `mount --bind /foo /foo` and `mount --make-private --make-shared /foo`. This will convert /foo into a `shared` mount point. Alternatively one can directly change propagation properties of source mount. Say `/` is source mount for `/foo`, then use `mount --make-shared /` to convert `/` into a `shared` mount.

> **Note**: When using systemd to manage the Docker daemon's start and stop, in the systemd unit file there is an option to control mount propagation for the Docker daemon itself, called `MountFlags`. The value of this setting may cause Docker to not see mount propagation changes made on the mount point. For example, if this value is `slave`, you may not be able to use the `shared` or `rshared` propagation on a volume.

To disable automatic copying of data from the container path to the volume, use the `nocopy` flag. The `nocopy` flag can be set on named volumes, and does not apply to bind mounts..

OPTIONS

--add-host= Add a custom host-to-IP mapping (host:ip)

-a, **--attach=** Attach to STDIN, STDOUT or STDERR

--blkio-weight=0 Block IO (relative weight), between 10 and 1000, or 0 to disable (default 0)

--blkio-weight-device=[] Block IO weight (relative device weight)

--cap-add= Add Linux capabilities

--cap-drop= Drop Linux capabilities

--cgroup-parent="" Optional parent cgroup for the container

--cgroupns="" Cgroup namespace to use (host|private)
default-cgroupns-mode option on the daemon (default)

--cidfile="" Write the container ID to the file

--cpu-count=0 CPU count (Windows only)

--cpu-percent=0 CPU percent (Windows only)

--cpu-period=0 Limit CPU CFS (Completely Fair Scheduler) period

--cpu-quota=0 Limit CPU CFS (Completely Fair Scheduler) quota

--cpu-rt-period=0 Limit CPU real-time period in microseconds

--cpu-rt-runtime=0 Limit CPU real-time runtime in microseconds

-c, **--cpu-shares**=0 CPU shares (relative weight)

--cpus= Number of CPUs

--cpuset-cpus="" CPUs in which to allow execution (0-3, 0,1)

--cpuset-mems="" MEMs in which to allow execution (0-3, 0,1)

--device= Add a host device to the container

--device-cgroup-rule= Add a rule to the cgroup allowed devices list

--device-read-bps=[] Limit read rate (bytes per second) from a device

--device-read-iops=[] Limit read rate (IO per second) from a device

--device-write-bps=[] Limit write rate (bytes per second) to a device

--device-write-iops=[] Limit write rate (IO per second) to a device

--disable-content-trust[=true] Skip image verification

--dns= Set custom DNS servers

--dns-option= Set DNS options

--dns-search= Set custom DNS search domains

--domainname="" Container NIS domain name

--entrypoint="" Overwrite the default ENTRYPOINT of the image

-e, **--env**= Set environment variables

--env-file= Read in a file of environment variables

--expose= Expose a port or a range of ports

--gpus= GPU devices to add to the container ('all' to pass all GPUs)

--group-add= Add additional groups to join

--health-cmd="" Command to run to check health

--health-interval=0s Time between running the check (ms|s|m|h) (default 0s)

--health-retries=0 Consecutive failures needed to report unhealthy

--health-start-period=0s Start period for the container to initialize before starting health-retries countdown (ms|s|m|h) (default 0s)

--health-timeout=0s Maximum time to allow one check to run (ms|s|m|h) (default 0s)

--help[=false] Print usage

-h, **--hostname=""** Container host name

--init[=false] Run an init inside the container that forwards signals and reaps processes

-i, **--interactive**[=false] Keep STDIN open even if not attached

--io-maxbandwidth=0 Maximum IO bandwidth limit for the system drive (Windows only)

--io-maxiops=0 Maximum IOps limit for the system drive (Windows only)

--ip="" IPv4 address (e.g., 172.30.100.104)

--ip6="" IPv6 address (e.g., 2001:db8::33)

--ipc="" IPC mode to use

--isolation="" Container isolation technology

--kernel-memory=0 Kernel memory limit

-l, **--label=** Set meta data on a container

--label-file= Read in a line delimited file of labels

--link= Add link to another container

--link-local-ip= Container IPv4/IPv6 link-local addresses

--log-driver="" Logging driver for the container

--log-opt= Log driver options

--mac-address="" Container MAC address (e.g., 92:d0:c6:0a:29:33)

-m, **--memory**=0 Memory limit

--memory-reservation=0 Memory soft limit

--memory-swap=0 Swap limit equal to memory plus swap: '-1' to enable unlimited swap

--memory-swappiness=-1 Tune container memory swappiness (0 to 100)

--mount= Attach a filesystem mount to the container

--name="" Assign a name to the container

--network= Connect a container to a network

--network-alias= Add network-scoped alias for the container

--no-healthcheck[=false] Disable any container-specified HEALTHCHECK

--oom-kill-disable[=false] Disable OOM Killer

--oom-score-adj=0 Tune host's OOM preferences (-1000 to 1000)

--pid="" PID namespace to use

--pids-limit=0 Tune container pids limit (set -1 for unlimited)

--platform="" Set platform if server is multi-platform capable

--privileged[=false] Give extended privileges to this container

-p, **--publish=** Publish a container's port(s) to the host

-P, **--publish-all**[=false] Publish all exposed ports to random ports

--pull="missing" Pull image before creating ("always"|"missing"|"never")

--read-only[=false] Mount the container's root filesystem as read only

--restart="no" Restart policy to apply when a container exits

--rm[=false] Automatically remove the container when it exits

--runtime="" Runtime to use for this container

--security-opt= Security Options

--shm-size=0 Size of /dev/shm

--stop-signal="SIGTERM" Signal to stop a container

--stop-timeout=0 Timeout (in seconds) to stop a container

--storage-opt= Storage driver options for the container

--sysctl=map[] Sysctl options

--tmpfs= Mount a tmpfs directory

-t, **--tty**[=false] Allocate a pseudo-TTY

--ulimit=[] Ulimit options

-u, **--user**="" Username or UID (format: [:])

--userns="" User namespace to use

--uts="" UTS namespace to use

-v, **--volume**= Bind mount a volume

--volume-driver="" Optional volume driver for the container

--volumes-from= Mount volumes from the specified container(s)

-w, **--workdir**="" Working directory inside the container

EXAMPLE

Specify isolation technology for container (--isolation)

This option is useful in situations where you are running Docker containers on Windows. The ‘--isolation=<value>‘ option sets a container’s isolation technology. On Linux, the only supported is the ‘default‘ option which uses Linux namespaces. On Microsoft Windows, you can specify these values:

* ‘default‘: Use the value specified by the Docker daemon’s ‘--exec-opt‘ . If the ‘daemon‘ does not specify an isolatio
* ‘process‘: Namespace isolation only.
* ‘hyperv‘: Hyper-V hypervisor partition-based isolation.

Specifying the ‘--isolation‘ flag without a value is the same as setting ‘--isolation="default"‘.

Dealing with dynamically created devices (--device-cgroup-rule)

Devices available to a container are assigned at creation time. The assigned devices will both be added to the cgroup.allow file and created into the container once it is run. This poses a problem when a new device needs to be added to running container.

One of the solution is to add a more permissive rule to a container allowing it access to a wider range of devices. For example, supposing our container needs access to a character device with major ‘42‘ and any number of minor number (added as new devices appear), the following rule would be added:

docker create --device-cgroup-rule=’c 42:* rmw’ -name my-container my-image

Then, a user could ask ‘udev‘ to execute a script that would ‘docker exec my-container mknod newDevX c 42 <mino the required device when it is added.

NOTE: initially present devices still need to be explicitly added to the create/run command

SEE ALSO

docker-container(1)