Docker container driver

- Synopsis
- Usage
- Cache persistence
- QEMU
- Custom network
- Further reading

The Docker container driver allows creation of a managed and customizable BuildKit environment in a dedicated Docker container.

Using the Docker container driver has a couple of advantages over the default Docker driver. For example:

- Specify custom BuildKit versions to use.
- Build multi-arch images, see QEMU
- Advanced options for cache import and export

Synopsis

Run the following command to create a new builder, named `container`, that uses the Docker container driver:

```
$ docker buildx create \
  --name container \
  --driver=docker-container \
  --driver-opt=[key=value,...]
container
```

The following table describes the available driver-specific options that you can pass to `--driver-opt`:

Parameter	Type	Default	Description

`image`	String		Sets the BuildKit image to use for the container.
`memory`	String		Sets the amount of memory the container can use.
`memory-swap`	String		Sets the memory swap limit for the container.
`cpu-quota`	String		Imposes a CPU CFS quota on the container.
`cpu-period`	String		Sets the CPU CFS scheduler period for the container.
`cpu-shares`	String		Configures CPU shares (relative weight) of the container.
`cpuset-cpus`	String		Limits the set of CPU cores the container can use.
`cpuset-mems`	String		Limits the set of CPU memory nodes the container can use.
`default-load`	Boolean	`false`	Automatically load images to the Docker Engine image store.
`network`	String		Sets the network mode for the container.
`cgroup-parent`	String	`/docker/buildx`	Sets the cgroup parent of the container if Docker is using the "cgroupfs" driver.
`restart-policy`	String	`unless-stopped`	Sets the container's restart policy.
`env.<key>`	String		Sets the environment variable `key` to the specified `value` in the container.
`provenance-add-gha`	Boolean	`true`	Automatically writes GitHub Actions context into the builder for provenance.

Before you configure the resource limits for the container, read about configuring runtime resource constraints for containers.

Usage

When you run a build, Buildx pulls the specified `image` (by default, `moby/buildkit`). When the container has started, Buildx submits the build submitted to the containerized build server.

```
$ docker buildx build -t <image> --builder=container .
WARNING: No output specified with docker-container driver. Build result will only remain in the build cache. To push result image into registry use --push or to load image into docker use --load
#1 [internal] booting buildkit
#1 pulling image moby/buildkit:buildx-stable-1
#1 pulling image moby/buildkit:buildx-stable-1 1.9s done
#1 creating container buildx_buildkit_container0
#1 creating container buildx_buildkit_container0 0.5s done
#1 DONE 2.4s
...
```

Cache persistence

The `docker-container` driver supports cache persistence, as it stores all the BuildKit state and related cache into a dedicated Docker volume.

To persist the `docker-container` driver's cache, even after recreating the driver using `docker buildx rm` and `docker buildx create`, you can destroy the builder using the `--keep-state` flag:

For example, to create a builder named `container` and then remove it while persisting state:

```
# setup a builder
$ docker buildx create --name=container --driver=docker-container --use --bootstrap container
$ docker buildx ls
NAME/NODE       DRIVER/ENDPOINT             STATUS   BUILDKIT PLATFORMS
container *     docker-container
  container0    desktop-linux               running  v0.10.5  linux/amd64
$ docker volume ls
DRIVER    VOLUME NAME
local     buildx_buildkit_container0_state

# remove the builder while persisting state
$ docker buildx rm --keep-state container
$ docker volume ls
```

```
DRIVER    VOLUME NAME
local     buildx_buildkit_container0_state

# the newly created driver with the same name will have all the state of the previous one!
$ docker buildx create --name=container --driver=docker-container --use --bootstrap
container
```

QEMU

The `docker-container` driver supports using QEMU (user mode) to build non-native platforms. Use the `--platform` flag to specify which architectures that you want to build for.

For example, to build a Linux image for `amd64` and `arm64`:

```
$ docker buildx build \
  --builder=container \
  --platform=linux/amd64,linux/arm64 \
  -t <registry>/<image> \
  --push .
```

> [!NOTE]
>
> Emulation with QEMU can be much slower than native builds, especially for compute-heavy tasks like compilation and compression or decompression.

Custom network

You can customize the network that the builder container uses. This is useful if you need to use a specific network for your builds.

For example, let's create a network named `foonet`:

```
$ docker network create foonet
```

Now create a `docker-container` builder that will use this network:

```
$ docker buildx create --use \
  --name mybuilder \
  --driver docker-container \
  --driver-opt "network=foonet"
```

Boot and inspect `mybuilder`:

```
$ docker buildx inspect --bootstrap
```

Inspect the builder container and see what network is being used:

```
$ docker inspect buildx_buildkit_mybuilder0 --format={{.NetworkSettings.Networks}}
map[foonet:0xc00018c0c0]
```

Further reading

For more information on the Docker container driver, see the buildx reference.

□□□ #2
□ □□□ □□□ 2026-10-31 19:03:45 UTC
□ □□□ □□□ 2026-10-31 19:03:46 UTC