



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

Rocky Enterprise Linux 9.2 Manual Pages on command 'buildah-run.1'

\$ man buildah-run.1

buildah-run(1) General Commands Manual buildah-run(1)

NAME

buildah-run - Run a command inside of the container.

SYNOPSIS

buildah run [options] [--] container command

DESCRIPTION

Launches a container and runs the specified command in that container using the container's root filesystem as a root filesystem, using configuration settings inherited from the container's image or as specified using previous calls to the buildah config command.

To execute buildah run within an interactive shell, specify the --tty option.

OPTIONS

--add-history

Add an entry to the history which will note what command is being invoked. Defaults to false.

Note: You can also override the default value of --add-history by setting the BUILDAH_HISTORY environment variable. export BUILDAH_HISTORY=true

--cap-add=CAP_xxx

Add the specified capability to the set of capabilities which will be granted to the specified command. Certain capabilities are granted by default; this option can be used to add more beyond the defaults, which may have been modified by --cap-add and --cap-drop options used with the buildah from invocation which created the container.

--cap-drop=CAP_xxx

Add the specified capability from the set of capabilities which will be granted to the

specified command. The CAP_AUDIT_WRITE, CAP_CHOWN, CAP_DAC_OVERRIDE, CAP_FOWNER, CAP_FSETID, CAP_KILL, CAP_MKNOD, CAP_NET_BIND_SERVICE, CAP_SETFCAP, CAP_SETGID, CAP_SETPCAP, CAP_SETUID, and CAP_SYS_CHROOT capabilities are granted by default; this option can be used to remove them from the defaults, which may have been modified by --cap-add and --cap-drop options used with the buildah from invocation which created the container.

If a capability is specified to both the --cap-add and --cap-drop options, it will be dropped, regardless of the order in which the options were given.

--cni-config-dir=directory

Location of CNI configuration files which will dictate which plugins will be used to configure network interfaces and routing inside the running container, if the container will be run in its own network namespace, and networking is not disabled.

--cni-plugin-path=directory[:directory[:directory[...]]]

List of directories in which the CNI plugins which will be used for configuring network namespaces can be found.

--env, -e env=value

Temporarily add a value (e.g. env=value) to the environment for the running process. Unlike buildah config --env, the environment will not persist to later calls to buildah run or to the built image. Can be used multiple times.

--hostname

Set the hostname inside of the running container.

--ipc how

Sets the configuration for the IPC namespaces for the container. The configured value can be "" (the empty string) or "private" to indicate that a new IPC namespace should be created, or it can be "host" to indicate that the IPC namespace in which buildah itself is being run should be reused, or it can be the path to an IPC namespace which is already in use by another process.

--isolation type

Controls what type of isolation is used for running the process. Recognized types include oci (OCI-compatible runtime, the default), rootless (OCI-compatible runtime invoked using a modified configuration, with --no-new-keyring added to its create invocation, reusing the host's network and UTS namespaces, and creating private IPC, PID, mount, and user namespaces; the default for unprivileged users), and chroot (an internal wrapper that leans more toward chroot(1) than container technology, reusing the host's control group,

network, IPC, and PID namespaces, and creating private mount and UTS namespaces, and creating user namespaces only when they're required for ID mapping).

Note: You can also override the default isolation type by setting the BUILDDAH_ISOLATION environment variable. export BUILDDAH_ISOLATION=oci

--mount=type=TYPE,TYPE-SPECIFIC-OPTION[,...]

Attach a filesystem mount to the container

Current supported mount TYPES are bind, and tmpfs. [1] ?#Footnote1?

e.g.

type=bind,source=/path/on/host,destination=/path/in/container

type=tmpfs,tmpfs-size=512M,destination=/path/in/container

Common Options:

? src, source: mount source spec for bind and volume. Mandatory for bind.

? dst, destination, target: mount destination spec.

? ro, read-only: true or false (default).

Options specific to bind:

? bind-propagation: shared, slave, private, rshared, rslave, or rprivate(default). See also mount(2).

. bind-nonrecursive: do not setup a recursive bind mount. By default it is recursive.

Options specific to tmpfs:

? tmpfs-size: Size of the tmpfs mount in bytes. Unlimited by default in Linux.

? tmpfs-mode: File mode of the tmpfs in octal. (e.g. 700 or 0700.) Defaults to 1777 in Linux.

--network, --net=mode

Sets the configuration for the network namespace for the container.

? none: no networking;

? host: use the host network stack. Note: the host mode gives the container full access to local system services such as D-bus and is therefore considered insecure;

? ns:path: path to a network namespace to join;

? private: create a new namespace for the container (default)

--pid how

Sets the configuration for the PID namespace for the container. The configured value can be "" (the empty string) or "private" to indicate that a new PID namespace should be created, or it can be "host" to indicate that the PID namespace in which buildah itself is being run should be reused, or it can be the path to a PID namespace which is already in

use by another process.

`--runtime path`

The path to an alternate OCI-compatible runtime. Default is `runc`, or `crun` when machine is configured to use `cgroups V2`.

Note: You can also override the default runtime by setting the `BUILDDAH_RUNTIME` environment variable. `export BUILDDAH_RUNTIME=/usr/bin/crun`

`--runtime-flag flag`

Adds global flags for the container runtime. To list the supported flags, please consult the `manpages` of the selected container runtime. Note: Do not pass the leading `--` to the flag. To pass the `runc` flag `--log-format json` to `buildah run`, the option given would be

`--runtime-flag log-format=json`.

`--no-pivot`

Do not use `pivot root` to jail process inside `rootfs`. This should be used whenever the `rootfs` is on top of a `ramdisk`.

Note: You can make this option the default by setting the `BUILDDAH_NOPIVOT` environment variable. `export BUILDDAH_NOPIVOT=true`

`-t, --tty, --terminal`

By default a pseudo-TTY is allocated only when `buildah`'s standard input is attached to a pseudo-TTY. Setting the `--tty` option to `true` will cause a pseudo-TTY to be allocated inside the container connecting the user's "terminal" with the `stdin` and `stdout` stream of the container. Setting the `--tty` option to `false` will prevent the pseudo-TTY from being allocated.

`--user user[:group]`

Set the user to be used for running the command in the container. The user can be specified as a user name or UID, optionally followed by a group name or GID, separated by a colon (`:`). If names are used, the container should include entries for those names in its `/etc/passwd` and `/etc/group` files.

`--uts how`

Sets the configuration for the UTS namespace for the container. The configured value can be `""` (the empty string) or `"private"` to indicate that a new UTS namespace should be created, or it can be `"host"` to indicate that the UTS namespace in which `buildah` itself is being run should be reused, or it can be the path to a UTS namespace which is already in use by another process.

--volume, -v source:destination:options

Create a bind mount. If you specify, -v /HOST-DIR:/CONTAINER-DIR, Buildah bind mounts /HOST-DIR in the host to /CONTAINER-DIR in the Buildah container. The OPTIONS are a comma delimited list and can be: [1] ?#Footnote1?

? [rw|ro]

? [U]

? [z|Z]

? [[r]shared|[r]slave|[r]private]

The CONTAINER-DIR must be an absolute path such as /src/docs. The HOST-DIR must be an absolute path as well. Buildah bind-mounts the HOST-DIR to the path you specify. For example, if you supply /foo as the host path, Buildah copies the contents of /foo to the container filesystem on the host and bind mounts that into the container.

You can specify multiple -v options to mount one or more mounts to a container.

Write Protected Volume Mounts

You can add the :ro or :rw suffix to a volume to mount it read-only or read-write mode, respectively. By default, the volumes are mounted read-write. See examples.

Chowning Volume Mounts

By default, Buildah does not change the owner and group of source volume directories mounted into containers. If a container is created in a new user namespace, the UID and GID in the container may correspond to another UID and GID on the host.

The :U suffix tells Buildah to use the correct host UID and GID based on the UID and GID within the container, to change the owner and group of the source volume.

Labeling Volume Mounts

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, Buildah 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 Buildah to relabel file objects on the shared volumes. The z option tells Buildah that two containers share the volume content. As a result, Buildah labels the content with a shared content label. Shared volume labels allow all containers to read/write content. The Z option tells Buildah 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 the host and vice versa. This behavior can be changed by specifying a volume mount propagation property.

When the mount propagation policy is set to shared, any mounts completed inside the container on that volume will be visible to both the host and container. When the mount propagation policy is set to slave, one way mount propagation is enabled and any mounts completed on the host for that volume will be visible only inside of the container. To control the mount propagation property of the volume use the `:[r]shared`, `:[r]slave` or `:[r]private` propagation flag. The propagation property can be specified only for bind mounted volumes and not for internal volumes or named volumes. For mount propagation to work on the source mount point (the mount point where source dir is mounted on) it has to have the right propagation properties. For shared volumes, the source mount point has to be shared. And for slave volumes, the source mount has to be either shared or slave. [1] [?#Footnote1?](#)

Use `df <source-dir>` to determine the source mount and then use `findmnt -o TARGET,PROPAGATION <source-mount-dir>` to determine propagation properties of source mount, if `findmnt` utility is not available, the source mount point can be determined by looking at the mount entry in `/proc/self/mountinfo`. Look at optional fields and see if any propagation properties are specified. `shared:X` means the mount is shared, `master:X` means the mount is slave and if nothing is there that means the mount is private. [1] [?#Footnote1?](#)

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

`--workingdir` directory

Temporarily set the working directory for the running process. Unlike `buildah config --workingdir`, the `workingdir` will not persist to later calls to `buildah run` or the built image.

NOTE: End parsing of options with the `--` option, so that other options can be passed to the command inside of the container.

EXAMPLE

```
buildah run containerID -- ps -auxw
```

```
buildah run --hostname myhost containerID -- ps -auxw
buildah run containerID -- sh -c 'echo $PATH'
buildah run --runtime-flag log-format=json containerID /bin/bash
buildah run --runtime-flag debug containerID /bin/bash
buildah run --tty containerID /bin/bash
buildah run --tty=false containerID ls /
buildah run --volume /path/on/host:/path/in/container:ro,z containerID sh
buildah run -v /path/on/host:/path/in/container:z,U containerID sh
buildah run --mount type=bind,src=/tmp/on:host,dst=/in:container,ro containerID sh
```

SEE ALSO

[buildah\(1\)](#), [buildah-from\(1\)](#), [buildah-config\(1\)](#), [namespaces\(7\)](#), [pid_namespaces\(7\)](#), [crun\(1\)](#), [runc\(8\)](#)

FOOTNOTES

1: The Buildah project is committed to inclusivity, a core value of open source. The `master` and `slave` mount propagation terminology used here is problematic and divisive, and should be changed. However, these terms are currently used within the Linux kernel and must be used as-is at this time. When the kernel maintainers rectify this usage, Buildah will follow suit immediately.

buildah

March 2017

[buildah-run\(1\)](#)