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

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'buildah-build.1' command

$ man buildah-build.1

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

NAME

buildah-build - Build an image using instructions from Containerfiles

SYNOPSIS

buildah build [options] [context]

buildah bud [options] [context]

DESCRIPTION

Builds an image using instructions from one or more Containerfiles or Dockerfiles and a specified build context directory. A Containerfile uses the same syntax as a Dockerfile internally. For this document, a file referred to as a Containerfile can be a file named either 'Con? tainerfile' or 'Dockerfile'.

The build context directory can be specified as the http(s) URL of an archive, git repository or Containerfile.

If no context directory is specified, then Buildah will assume the cur? rent working directory as build context, which should contain a Con? tainerfile.

Containerfiles ending with a ".in" suffix will be preprocessed via cpp(1). This can be useful to decompose Containerfiles into several reusable parts that can be used via CPP's #include directive. Notice, a Containerfile.in file can still be used by other tools when manually preprocessing them via cpp -E. Any comments (Lines beginning with #) in included Containerfile(s) that are not preprocess commands, will be

printed as warnings during builds.

When the URL is an archive, the contents of the URL is downloaded to a temporary location and extracted before execution.

When the URL is a Containerfile, the file is downloaded to a temporary location.

When a Git repository is set as the URL, the repository is cloned lo? cally and then used as the build context. A non-default branch (or commit ID) and subdirectory of the cloned git repository can be used by including their names at the end of the URL in the form myrepo.git#my? branch:subdir, myrepo.git#mycommit:subdir, or myrepo.git#:subdir if the subdirectory should be used from the default branch.

OPTIONS

--add-host=[]

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

Add a line to /etc/hosts. The format is hostname:ip. The --add-host op? tion can be set multiple times. Conflicts with the --no-hosts option.

--all-platforms

Instead of building for a set of platforms specified using the --plat? form option, inspect the build's base images, and build for all of the platforms for which they are all available. Stages that use scratch as a starting point can not be inspected, so at least one non-scratch stage must be present for detection to work usefully.

--annotation annotation[=value]

Add an image annotation (e.g. annotation=value) to the image metadata. Can be used multiple times. If annotation is named, but neither = nor a value is provided, then the annotation is set to an empty value.

Note: this information is not present in Docker image formats, so it is discarded when writing images in Docker formats.

--arch="ARCH"

Set the ARCH of the image to be built, and that of the base image to be pulled, if the build uses one, to the provided value instead of using the architecture of the host. (Examples: arm, arm64, 386, amd64, ppc64le, s390x)

--authfile path

Path of the authentication file. Default is ${XDG_\RUNTIME_DIR}/con?

tainers/auth.json. If XDG_RUNTIME_DIR is not set, the default is

/run/containers/$UID/auth.json. This file is created using buildah lo?

gin.

If the authorization state is not found there, $HOME/.docker/con?

fig.json is checked, which is set using docker login.

Note: You can also override the default path of the authentication file

by setting the REGISTRY_AUTH_FILE environment variable. export REG?

ISTRY_AUTH_FILE=path

--build-arg arg=value

Specifies a build argument and its value, which will be interpolated in

instructions read from the Containerfiles in the same way that environ?

ment variables are, but which will not be added to environment variable

list in the resulting image's configuration.

Please refer to the BUILD TIME VARIABLES ?#build-time-variables? sec?

tion for the list of variables that can be overridden within the Con?

tainerfile at run time.

--build-context name=value

Specify an additional build context using its short name and its loca?

tion. Additional build contexts can be referenced in the same manner as

we access different stages in COPY instruction.

Valid values could be: * Local directory ? e.g. --build-context

project2=../path/to/project2/src * HTTP URL to a tarball ? e.g.

--build-context src=https://example.org/releases/src.tar * Container

image ? specified with a container-image:// prefix, e.g. --build-con?

text alpine=container-image://alpine:3.15, (also accepts docker://,

docker-image://)

On the Containerfile side, you can reference the build context on all

commands that accept the ?from? parameter. Here?s how that might look:

FROM [name]

COPY --from=[name] ...

RUN --mount=from=[name] ?

The value of [name] is matched with the following priority order:

? Named build context defined with --build-context [name]=..

? Stage defined with AS [name] inside Containerfile

? Image [name], either local or in a remote registry

--cache-from

Repository to utilize as a potential list of cache sources. When speci?
fied, Buildah will try to look for cache images in the specified repos?
itories and will attempt to pull cache images instead of actually exe?
cuting the build steps locally. Buildah will only attempt to pull pre?
viously cached images if they are considered as valid cache hits.

Use the --cache-to option to populate a remote repository or reposito?
ries with cache content.

Example

populate a cache and also consult it

buildah build -t test --layers --cache-to registry/myrepo/cache --cache-from registry/myrepo/cache .

Note: --cache-from option is ignored unless --layers is specified.

--cache-to

Set this flag to specify list of remote repositories that will be used
to store cache images. Buildah will attempt to push newly built cache
image to the remote repositories.

Note: Use the --cache-from option in order to use cache content in a
remote repository.

Example

populate a cache and also consult it

buildah build -t test --layers --cache-to registry/myrepo/cache --cache-from registry/myrepo/cache .

Note: --cache-to option is ignored unless --layers is specified.

--cache-ttl duration

Limit the use of cached images to only consider images with created
timestamps less than duration ago. For example if --cache-ttl=1h is
specified, Buildah will only consider intermediate cache images which
are created under the duration of one hour, and intermediate cache im?
ages outside this duration will be ignored.

Note: Setting --cache-ttl=0 manually is equivalent to using --no-cache

in the implementation since this would effectively mean that user is not willing to use cache at all.

--cap-add=CAP_xxx

When executing RUN instructions, run the command specified in the in? struction with the specified capability added to its capability set. Certain capabilities are granted by default; this option can be used to add more.

--cap-drop=CAP_xxx

When executing RUN instructions, run the command specified in the in? struction with the specified capability removed from its capability set. 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.

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

--cert-dir path

Use certificates at path (*.crt, *.cert, *.key) to connect to the reg? istry. The default certificates directory is /etc/containers/certs.d.

--cgroup-parent=""

Path to cgroups under which the cgroup for the container will be cre? ated. If the path is not absolute, the path is considered to be rela? tive to the cgroups path of the init process. Cgroups will be created if they do not already exist.

--cgroupns how

Sets the configuration for cgroup namespaces when handling RUN instruc? tions. The configured value can be "" (the empty string) or "private" to indicate that a new cgroup namespace should be created, or it can be "host" to indicate that the cgroup namespace in which buildah itself is being run should be reused.

--compress

This option is added to be aligned with other containers CLIs. Buildah

doesn't send a copy of the context directory to a daemon or a remote server. Thus, compressing the data before sending it is irrelevant to Buildah.

--cpp-flag=""

Set additional flags to pass to the C Preprocessor cpp(1). Container? files ending with a ".in" suffix will be preprocessed via cpp(1). This option can be used to pass additional flags to cpp. Note: You can also set default CPPFLAGS by setting the BUILDAH_CPPFLAGS environment vari? able (e.g., export BUILDAH_CPPFLAGS="-DDEBUG").

--cpu-period=0

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 CPU limits may not be allowed for non-root users. For more details, see https://github.com/containers/pod?man/blob/main/troubleshooting.md#26-running-containers-with-cpu-limits-fails-with-a-permissions-error

--cpu-quota=0

Limit the CPU CFS (Completely Fair Scheduler) quota

Limit the container's CPU usage. By default, containers run with the full CPU resource. This flag tells the kernel to restrict the con? tainer's CPU usage to the quota you specify.

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

--cpu-shares, -c=0

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 weighting of all other running containers.

To modify the proportion from the default of 1024, use the --cpu-shares

flag to set the weighting to 2 or higher.

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 you add a fourth container 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 you start one container {C0} with -c=512 running one process, and another con? tainer {C1} with -c=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

--cpuset-cpus=""

CPUs in which to allow execution (0-3, 0,1)

--cpuset-mems=""

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

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

--creds creds

The [username[:password]] to use to authenticate with the registry if required. If one or both values are not supplied, a command line

prompt will appear and the value can be entered. The password is en?
tered without echo.

--decryption-key key[:passphrase]

The [key[:passphrase]] to be used for decryption of images. Key can point to keys and/or certificates. Decryption will be tried with all keys. If the key is protected by a passphrase, it is required to be passed in the argument and omitted otherwise.

--device=device

Add a host device to the container. Optional permissions parameter can be used to specify device permissions, it is combination of r for read, w for write, and m for mknod(2).

Example: --device=/dev/sdc:/dev/xvdc:rwm.

Note: if _hostdevice is a symbolic link then it will be resolved first.

The container will only store the major and minor numbers of the host device.

Note: if the user only has access rights via a group, accessing the de?
vice from inside a rootless container will fail. The crun(1) runtime offers a workaround for this by adding the option --annotation run.oci.keep_original_groups=1.

--disable-compression, -D

Don't compress filesystem layers when building the image unless it is required by the location where the image is being written. This is the default setting, because image layers are compressed automatically when they are pushed to registries, and images being written to local stor?
age would only need to be decompressed again to be stored. Compression can be forced in all cases by specifying --disable-compression=false.

--disable-content-trust

This is a Docker specific option to disable image verification to a Container registry and is not supported by Buildah. This flag is a NOOP and provided solely for scripting compatibility.

--dns=[]

Set custom DNS servers. Invalid if using --dns with --network=none.

This option can be used to override the DNS configuration passed to the

container. Typically this is necessary when the host DNS configuration is invalid for the container (e.g., 127.0.0.1). When this is the case the --dns flag is necessary for every run.

The special value none can be specified to disable creation of /etc/re? solv.conf in the container by Buildah. The /etc/resolv.conf file in the image will be used without changes.

--dns-option=[]

Set custom DNS options. Invalid if using --dns-option with --net? work=none.

--dns-search=[]

Set custom DNS search domains. Invalid if using --dns-search with --network=none.

--env env[=value]

Add a value (e.g. env=value) to the built image. Can be used multiple times. If neither = nor a *value* are specified, but env is set in the current environment, the value from the current environment will be added to the image. The value of env can be overridden by ENV instruc? tions in the Containerfile. To remove an environment variable from the built image, use the --unsetenv option.

--file, -f Containerfile

Specifies a Containerfile which contains instructions for building the image, either a local file or an http or https URL. If more than one Containerfile is specified, FROM instructions will only be accepted from the first specified file.

If a local file is specified as the Containerfile and it does not ex? ist, the context directory will be prepended to the local file value.

If you specify -f -, the Containerfile contents will be read from stdin.

--force-rm bool-value

Always remove intermediate containers after a build, even if the build fails (default false).

--format

Control the format for the built image's manifest and configuration

data. Recognized formats include oci (OCI image-spec v1.0, the de?
fault) and docker (version 2, using schema format 2 for the manifest).
Note: You can also override the default format by setting the BUIL?
DAH_FORMAT environment variable. export BUILDAH_FORMAT=docker
--from
Overrides the first FROM instruction within the Containerfile. If
there are multiple FROM instructions in a Containerfile, only the first
is changed.
--group-add=group | keep-groups
Assign additional groups to the primary user running within the con?
tainer process.
? keep-groups is a special flag that tells Buildah to keep the
supplementary group access.
Allows container to use the user's supplementary group access. If file
systems or devices are only accessible by the rootless user's group,
this flag tells the OCI runtime to pass the group access into the con?
tainer. Currently only available with the crun OCI runtime. Note: keep-
groups is exclusive, other groups cannot be specified with this flag.
--help, -h
Print usage statement
--hooks-dir path
Each *.json file in the path configures a hook for buildah build con?
tainers. For more details on the syntax of the JSON files and the se?
mantics of hook injection. Buildah currently support both the 1.0.0 and
0.1.0 hook schemas, although the 0.1.0 schema is deprecated.
This option may be set multiple times; paths from later options have
higher precedence.
For the annotation conditions, buildah uses any annotations set in the
generated OCI configuration.
For the bind-mount conditions, only mounts explicitly requested by the
caller via --volume are considered. Bind mounts that buildah inserts by
default (e.g. /dev/shm) are not considered.
If --hooks-dir is unset for root callers, Buildah will currently de?

fault to /usr/share/containers/oci/hooks.d and /etc/contain?
ers/oci/hooks.d in order of increasing precedence. Using these defaults
is deprecated, and callers should migrate to explicitly setting
--hooks-dir.

--http-proxy=true

By default proxy environment variables are passed into the container if
set for the buildah process. This can be disabled by setting the
--http-proxy option to false. The environment variables passed in in?
clude http_proxy, https_proxy, ftp_proxy, no_proxy, and also the upper
case versions of those.

--identity-label bool-value

Adds default identity label io.buildah.version if set. (default true).

--ignorefile file

Path to an alternative .containerignore (.dockerignore) file.

--iidfile ImageIDfile

Write the built image's ID to the file. When --platform is specified
more than once, attempting to use this option will trigger an error.

--ipc how

Sets the configuration for IPC namespaces when handling RUN instruc?
tions. The configured value can be "" (the empty string) or "con?
tainer" to indicate that a new IPC namespace should be created, or it
can be "host" to indicate that the IPC namespace in which buildah it?
self 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 processes as part
of RUN instructions. Recognized types include oci (OCI-compatible run?
time, the default), rootless (OCI-compatible runtime invoked using a
modified configuration, with --no-new-keyring added to its create invo?
cation, reusing the host's network and UTS namespaces, and creating
private IPC, PID, mount, and user namespaces; the default for unprivi?
leged 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 BUILDAH_ISOLATION environment variable. export BUILDAH_ISOLATION=oci

--jobs N

Run up to N concurrent stages in parallel. If the number of jobs is greater than 1, stdin will be read from /dev/null. If 0 is specified, then there is no limit on the number of jobs that run in parallel.

--label label[=value]

Add an image label (e.g. label=value) to the image metadata. Can be used multiple times. If label is named, but neither = nor a value is provided, then the label is set to an empty value.

Users can set a special LABEL io.containers.capabilities=CAP1,CAP2,CAP3 in a Containerfile that specifies the list of Linux capabilities re? quired for the container to run properly. This label specified in a container image tells container engines, like Podman, to run the con? tainer with just these capabilities. The container engine launches the container with just the specified capabilities, as long as this list of capabilities is a subset of the default list.

If the specified capabilities are not in the default set, container en? gines should print an error message and will run the container with the default capabilities.

--layers bool-value

Cache intermediate images during the build process (Default is false).

Note: You can also override the default value of layers by setting the BUILDAH_LAYERS environment variable. export BUILDAH_LAYERS=true

--logfile filename

Log output which would be sent to standard output and standard error to the specified file instead of to standard output and standard error.

--logsplit bool-value

If --logfile and --platform is specified following flag allows end-users to split log file for each platform into different files with

naming convention as ${logfile}_${platform-os}_${platform-arch}.

--manifest listName

Name of the manifest list to which the built image will be added. Cre? ates the manifest list if it does not exist. This option is useful for building multi architecture images. If listName does not include a registry name component, the registry name localhost will be prepended to the list name.

--memory, -m=""

Memory limit (format: [], where unit = b, k, m or g)

Allows you to constrain the memory available to a container. 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).

--memory-swap="LIMIT"

A limit value equal to memory plus swap. Must be used with the -m (--memory) flag. The swap LIMIT should always be larger than -m (--mem? ory) value. By default, the swap LIMIT will be set to double the value of --memory.

The format of LIMIT is <number>[<unit>]. Unit can be b (bytes), k (kilobytes), m (megabytes), or g (gigabytes). If you don't specify a unit, b is used. Set LIMIT to -1 to enable unlimited swap.

--network, --net=mode

Sets the configuration for network namespaces when handling RUN in? structions.

Valid mode values are:

? none: no networking. Invalid if using --dns, --dns-opt, or --dns-search;

? 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)

? <network name|ID>: Join the network with the given name or ID, e.g. use --network mynet to join the network with the name mynet. Only supported for rootful users.

--no-cache

Do not use existing cached images for the container build. Build from the start with a new set of cached layers.

--no-hosts

Do not create /etc/hosts for the container.

By default, Buildah manages /etc/hosts, adding the container's own IP address. --no-hosts disables this, and the image's /etc/hosts will be preserved unmodified. Conflicts with the --add-host option.

--omit-history bool-value

Omit build history information in the built image. (default false).

This option is useful for the cases where end users explicitly want to set --omit-history to omit the optional History from built images or when working with images built using build tools that do not include History information in their images.

--os="OS"

Set the OS of the image to be built, and that of the base image to be pulled, if the build uses one, instead of using the current operating system of the host.

--os-feature feature

Set the name of a required operating system feature for the image which will be built. By default, if the image is not based on scratch, the base image's required OS feature list is kept, if the base image speci? fied any. This option is typically only meaningful when the image's OS is Windows.

If feature has a trailing -, then the feature is removed from the set of required features which will be listed in the image.

--os-version version

Set the exact required operating system version for the image which will be built. By default, if the image is not based on scratch, the

base image's required OS version is kept, if the base image specified one. This option is typically only meaningful when the image's OS is Windows, and is typically set in Windows base images, so using this op? tion is usually unnecessary.

--output, -o=""

Output destination (format: type=local,dest=path)

The --output (or -o) option extends the default behavior of building a container image by allowing users to export the contents of the image as files on the local filesystem, which can be useful for generating local binaries, code generation, etc.

The value for --output is a comma-separated sequence of key=value pairs, defining the output type and options.

Supported keys are: - dest: Destination path for exported output. Valid value is absolute or relative path, - means the standard output. - type: Defines the type of output to be used. Valid values is documented below.

Valid type values are: - local: write the resulting build files to a directory on the client-side. - tar: write the resulting files as a single tarball (.tar).

If no type is specified, the value defaults to local. Alternatively, instead of a comma-separated sequence, the value of --output can be just a destination (in the **dest** format) (e.g.--output some-path,--output -) where--output some-pathis treated as if **type=local** and--output -` is treated as if type=tar.

--pid how

Sets the configuration for PID namespaces when handling RUN instruc? tions. 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.

--platform="OS/ARCH[/VARIANT]"

Set the OS/ARCH of the built image (and its base image, if your build

uses one) to the provided value instead of using the current operating system and architecture of the host (for example linux/arm).

The --platform flag can be specified more than once, or given a comma-separated list of values as its argument. When more than one platform is specified, the --manifest option should be used instead of the --tag option.

OS/ARCH pairs are those used by the Go Programming Language. In sev? eral cases the ARCH value for a platform differs from one produced by other tools such as the arch command. Valid OS and architecture name combinations are listed as values for $GOOS and $GOARCH at https://golang.org/doc/install/source#environment, and can also be found by running go tool dist list.

While buildah bud is happy to use base images and build images for any platform that exists, RUN instructions will not be able to succeed without the help of emulation provided by packages like qemu-user-static.

NOTE: The --platform option may not be used in combination with the --arch, --os, or --variant options.

--pull

When the flag is enabled or set explicitly to true (with --pull=true), attempt to pull the latest image from the registries listed in reg? istries.conf if a local image does not exist or the image is newer than the one in storage. Raise an error if the image is not in any listed registry and is not present locally.

If the flag is disabled (with --pull=false), do not pull the image from the registry, use only the local version. Raise an error if the image is not present locally.

If the pull flag is set to always (with --pull=always), pull the image from the first registry it is found in as listed in registries.conf. Raise an error if not found in the registries, even if the image is present locally.

If the pull flag is set to missing (with --pull=missing), pull the im? age only if it could not be found in the local containers storage.

Raise an error if no image could be found and the pull fails.

If the pull flag is set to never (with --pull=never), Do not pull the image from the registry, use only the local version. Raise an error if the image is not present locally.

Defaults to true.

--quiet, -q

Suppress output messages which indicate which instruction is being pro? cessed, and of progress when pulling images from a registry, and when writing the output image.

--retry attempts

Number of times to retry in case of failure when performing push/pull of images to/from registry.

Defaults to 3.

--retry-delay duration

Duration of delay between retry attempts in case of failure when per? forming push/pull of images to/from registry.

Defaults to 2s.

--rm bool-value

Remove intermediate containers after a successful build (default true).

--runtime path

The path to an alternate OCI-compatible runtime, which will be used to run commands specified by the RUN instruction. Default is runc, or crun when machine is configured to use cgroups V2.

Note: You can also override the default runtime by setting the BUIL? DAH_RUNTIME environment variable. export BUILDAH_RUNTIME=/usr/bin/crun

--runtime-flag flag

Adds global flags for the container rutime. 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 build, the option given would be --run? time-flag log-format=json.

--secret=id=id,src=path

Pass secret information to be used in the Containerfile for building

images in a safe way that will not end up stored in the final image, or be seen in other stages. The secret will be mounted in the container at the default location of /run/secrets/id.

To later use the secret, use the --mount flag in a RUN instruction within a Containerfile:

RUN --mount=type=secret,id=mysecret cat /run/secrets/mysecret

--security-opt=[]

Security Options

"apparmor=unconfined" : Turn off apparmor confinement for the container

"apparmor=your-profile" : Set the apparmor confinement profile for the container

"label=user:USER" : Set the label user for the container

"label=role:ROLE" : Set the label role for the container

"label=type:TYPE" : Set the label type for the container

"label=level:LEVEL" : Set the label level for the container

"label=disable" : Turn off label confinement for the container

"no-new-privileges" : Disable container processes from gaining addi? tional privileges

"seccomp=unconfined" : Turn off seccomp confinement for the container

"seccomp=profile.json : White listed syscalls seccomp Json file to be used as a seccomp filter

--shm-size=""

Size of /dev/shm. The format is <number><unit>. number must be greater than 0. Unit is optional and can be b (bytes), k (kilobytes), m(megabytes), or g (gigabytes). If you omit the unit, the system uses bytes. If you omit the size entirely, the system uses 64m.

--sign-by fingerprint

Sign the built image using the GPG key that matches the specified fin? gerprint.

--skip-unused-stages bool-value

Skip stages in multi-stage builds which don't affect the target stage. (Default is true).

--squash

Squash all layers, including those from base image(s), into one single layer. (Default is false).

By default, Buildah preserves existing base-image layers and adds only one new layer on a build. The --layers option can be used to preserve intermediate build layers.

--ssh=default|id[=socket>|[,]

SSH agent socket or keys to expose to the build. The socket path can be left empty to use the value of default=$SSH_AUTH_SOCK

To later use the ssh agent, use the --mount flag in a RUN instruction within a Containerfile:

RUN --mount=type=secret,id=id mycmd

--stdin

Pass stdin into the RUN containers. Sometimes commands being RUN within a Containerfile want to request information from the user. For example apt asking for a confirmation for install. Use --stdin to be able to interact from the terminal during the build.

--tag, -t imageName

Specifies the name which will be assigned to the resulting image if the build process completes successfully. If imageName does not include a registry name component, the registry name localhost will be prepended to the image name.

--target stageName

Set the target build stage to build. When building a Containerfile with multiple build stages, --target can be used to specify an interme? diate build stage by name as the final stage for the resulting image. Commands after the target stage will be skipped.

--timestamp seconds

Set the create timestamp to seconds since epoch to allow for determin? istic builds (defaults to current time). By default, the created time? stamp is changed and written into the image manifest with every commit, causing the image's sha256 hash to be different even if the sources are exactly the same otherwise. When --timestamp is set, the created time? stamp is always set to the time specified and therefore not changed,

allowing the image's sha256 to remain the same. All files committed to the layers of the image will be created with the timestamp.

--tls-verify bool-value

Require HTTPS and verification of certificates when talking to con? tainer registries (defaults to true). TLS verification cannot be used when talking to an insecure registry.

--ulimit type=soft-limit[:hard-limit]

Specifies resource limits to apply to processes launched when process? ing RUN instructions. This option can be specified multiple times. Recognized resource types include:

"core": maximum core dump size (ulimit -c)

"cpu": maximum CPU time (ulimit -t)

"data": maximum size of a process's data segment (ulimit -d)

"fsize": maximum size of new files (ulimit -f)

"locks": maximum number of file locks (ulimit -x)

"memlock": maximum amount of locked memory (ulimit -l)

"msgqueue": maximum amount of data in message queues (ulimit -q)

"nice": niceness adjustment (nice -n, ulimit -e)

"nofile": maximum number of open files (ulimit -n)

"nofile": maximum number of open files (1048576); when run by root

"nproc": maximum number of processes (ulimit -u)

"nproc": maximum number of processes (1048576); when run by root

"rss": maximum size of a process's (ulimit -m)

"rtprio": maximum real-time scheduling priority (ulimit -r)

"rttime": maximum amount of real-time execution between blocking syscalls

"sigpending": maximum number of pending signals (ulimit -i)

"stack": maximum stack size (ulimit -s)

--unsetenv env

Unset environment variables from the final image.

--userns how

Sets the configuration for user namespaces when handling RUN instruc? tions. The configured value can be "" (the empty string) , "private"

or "auto" to indicate that a new user namespace should be created, it can be "host" to indicate that the user namespace in which buildah it? self is being run should be reused, or it can be the path to an user namespace which is already in use by another process.

auto: automatically create a unique user namespace.

The --userns=auto flag, requires that the user name containers and a range of subordinate user ids that the build container is allowed to use be specified in the /etc/subuid and /etc/subgid files.

Example: containers:2147483647:2147483648.

Buildah allocates unique ranges of UIDs and GIDs from the containers subordinate user ids. The size of the ranges is based on the number of UIDs required in the image. The number of UIDs and GIDs can be overrid? den with the size option.

Valid auto options:

- ? gidmapping=CONTAINER_GID:HOST_GID:SIZE: to force a GID mapping to be present in the user namespace.
- ? size=SIZE: to specify an explicit size for the automatic user namespace. e.g. --userns=auto:size=8192. If size is not speci? fied, auto will estimate a size for the user namespace.
- ? uidmapping=CONTAINER_UID:HOST_UID:SIZE: to force a UID mapping to be present in the user namespace.

--userns-gid-map mapping

Directly specifies a GID mapping which should be used to set ownership, at the filesystem level, on the working container's contents. Commands run when handling RUN instructions will default to being run in their own user namespaces, configured using the UID and GID maps.

Entries in this map take the form of one or more colon-separated triples of a starting in-container GID, a corresponding starting host-level GID, and the number of consecutive IDs which the map entry repre? sents.

This option overrides the remap-gids setting in the options section of /etc/containers/storage.conf.

If this option is not specified, but a global --userns-gid-map setting

is supplied, settings from the global option will be used.

--userns-gid-map-group group

Specifies that a GID mapping which should be used to set ownership, at the filesystem level, on the working container's contents, can be found in entries in the /etc/subgid file which correspond to the specified group. Commands run when handling RUN instructions will default to be? ing run in their own user namespaces, configured using the UID and GID maps. If --userns-uid-map-user is specified, but --userns-gid-map-group is not specified, buildah will assume that the specified user name is also a suitable group name to use as the default setting for this option.

Users can specify the maps directly using --userns-gid-map described in the buildah(1) man page.

NOTE: When this option is specified by a rootless user, the specified mappings are relative to the rootless usernamespace in the container, rather than being relative to the host as it would be when run rootful.

--userns-uid-map mapping

Directly specifies a UID mapping which should be used to set ownership, at the filesystem level, on the working container's contents. Commands run when handling RUN instructions will default to being run in their own user namespaces, configured using the UID and GID maps.

Entries in this map take the form of one or more colon-separated triples of a starting in-container UID, a corresponding starting host-level UID, and the number of consecutive IDs which the map entry repre? sents.

This option overrides the remap-uids setting in the options section of /etc/containers/storage.conf.

If this option is not specified, but a global --userns-uid-map setting is supplied, settings from the global option will be used.

--userns-uid-map-user user

Specifies that a UID mapping which should be used to set ownership, at the filesystem level, on the working container's contents, can be found in entries in the /etc/subuid file which correspond to the specified

user. Commands run when handling RUN instructions will default to be?
ing run in their own user namespaces, configured using the UID and GID
maps. If --userns-gid-map-group is specified, but --userns-uid-map-
user is not specified, buildah will assume that the specified group
name is also a suitable user name to use as the default setting for
this option.
NOTE: When this option is specified by a rootless user, the specified
mappings are relative to the rootless usernamespace in the container,
rather than being relative to the host as it would be when run rootful.
--uts how
Sets the configuration for UTS namespaces when handling RUN instruc?
tions. The configured value can be "" (the empty string) or "con?
tainer" to indicate that a new UTS namespace should be created, or it
can be "host" to indicate that the UTS namespace in which buildah it?
self is being run should be reused, or it can be the path to a UTS
namespace which is already in use by another process.
--variant=""
Set the architecture variant of the image to be pulled.
--volume, -v[=[HOST-DIR:CONTAINER-DIR[:OPTIONS]]]
Mount a host directory into containers when executing RUN instructions
during the build. The OPTIONS are a comma delimited list and can be:
[1] ?#Footnote1?

? [rw|ro]

? [U]

? [z|Z|O]

? [[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 vol? ume 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 la? bels the content with a shared content label. Shared volume labels al? low 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.

Overlay Volume Mounts

The :O flag tells Buildah to mount the directory from the host as a temporary storage using the Overlay file system. The RUN command con? tainers are allowed to modify contents within the mountpoint and are stored in the container storage in a separate directory. In Overlay FS terms the source directory will be the lower, and the container storage directory will be the upper. Modifications to the mount point are de?

stroyed when the RUN command finishes executing, similar to a tmpfs mount point.

Any subsequent execution of RUN commands sees the original source di? rectory content, any changes from previous RUN commands no longer ex? ist.

One use case of the overlay mount is sharing the package cache from the host into the container to allow speeding up builds.

Note:

- The `O` flag is not allowed to be specified with the `Z` or `z` flags. Content mounted into the container is labeled with the private label.

On SELinux systems, labels in the source directory must be readable by the container label. If not, SELinux container separation must be disabled for the container to work.

- Modification of the directory volume mounted into the container with an overlay mount can cause unexpected failures. It is recommended that you do not modify the directory until the container finishes running.

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 prop? erty.

When the mount propagation policy is set to shared, any mounts com? pleted 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 in? ternal 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 prop?

erties 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 propaga? tion 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 com? mand. 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.

BUILD TIME VARIABLES

The ENV instruction in a Containerfile can be used to define variable values. When the image is built, the values will persist in the con? tainer image. At times it is more convenient to change the values in the Containerfile via a command-line option rather than changing the values within the Containerfile itself.

The following variables can be used in conjunction with the --build-arg option to override the corresponding values set in the Containerfile using the ENV instruction.

- ? HTTP_PROXY
- ? HTTPS_PROXY
- ? FTP_PROXY
- ? NO_PROXY

Please refer to the Using Build Time Variables ?#using-build-time-vari? ables? section of the Examples.

EXAMPLE

Build an image using local Containerfiles

```
buildah build .

buildah build -f Containerfile .

cat ~/Containerfile | buildah build -f - .
```

buildah build -f Containerfile.simple -f Containerfile.notsosimple .

buildah build --timestamp=$(date '+%s') -t imageName .

buildah build -t imageName .

buildah build --tls-verify=true -t imageName -f Containerfile.simple .

buildah build --tls-verify=false -t imageName .

buildah build --runtime-flag log-format=json .

buildah build -f Containerfile --runtime-flag debug .

buildah build --authfile /tmp/auths/myauths.json --cert-dir ~/auth

--tls-verify=true --creds=username:password -t imageName -f Container?

file.simple .

buildah build --memory 40m --cpu-period 10000 --cpu-quota 50000

--ulimit nofile=1024:1028 -t imageName .

buildah build --security-opt label=level:s0:c100,c200 --cgroup-parent

/path/to/cgroup/parent -t imageName .

buildah build --arch=arm --variant v7 -t imageName .

buildah build --volume /home/test:/myvol:ro,Z -t imageName .

buildah build -v /home/test:/myvol:z,U -t imageName .

buildah build -v /var/lib/dnf:/var/lib/dnf:O -t imageName .

buildah build --layers -t imageName .

buildah build --no-cache -t imageName .

buildah build -f Containerfile --layers --force-rm -t imageName .

buildah build --no-cache --rm=false -t imageName .

buildah build --dns-search=example.com --dns=223.5.5.5 --dns-op?

tion=use-vc .

buildah build -f Containerfile.in --cpp-flag="-DDEBUG" -t imageName .

buildah build --network mynet .

buildah build --env LANG=en_US.UTF-8 -t imageName .

buildah build --env EDITOR -t imageName .

buildah build --unsetenv LANG -t imageName .

buildah build --os-version 10.0.19042.1645 -t imageName .

buildah build --os-feature win32k -t imageName .

buildah build --os-feature win32k- -t imageName .

Building an multi-architecture image using the --manifest option (requires

emulation software)

buildah build --arch arm --manifest myimage /tmp/mysrc

buildah build --arch amd64 --manifest myimage /tmp/mysrc

buildah build --arch s390x --manifest myimage /tmp/mysrc

buildah bud --platform linux/s390x,linux/ppc64le,linux/amd64 --manifest myimage /tmp/mysrc

buildah bud --platform linux/arm64 --platform linux/amd64 --manifest myimage /tmp/mysrc

buildah bud --all-platforms --manifest myimage /tmp/mysrc

Building an image using (--output) custom build output

buildah build -o out .

buildah build --output type=local,dest=out .

buildah build --output type=tar,dest=out.tar .

buildah build -o - . > out.tar

Building an image using a URL

This will clone the specified GitHub repository from the URL and use it as context. The Containerfile or Dockerfile at the root of the reposi? tory is used as the context of the build. This only works if the GitHub repository is a dedicated repository.

buildah build https://github.com/scollier/purpletest

Note: Github does not support using git:// for performing clone opera? tion due to recent changes in their security guidance (https://github.blog/2021-09-01-improving-git-protocol-security-github/). Use an https:// URL if the source repository is hosted on Github.

Building an image using a URL to a tarball'ed context

Buildah will fetch the tarball archive, decompress it and use its con? tents as the build context. The Containerfile or Dockerfile at the root of the archive and the rest of the archive will get used as the context of the build. If you pass an -f PATH/Containerfile option as well, the system will look for that file inside the contents of the tarball.

buildah build -f dev/Containerfile https://10.10.10.1/buildah/con?

text.tar.gz

Note: supported compression formats are 'xz', 'bzip2', 'gzip' and 'identity' (no compression).

Using Build Time Variables

Replace the value set for the HTTP_PROXY environment variable within the Containerfile.

buildah build --build-arg=HTTP_PROXY="http://127.0.0.1:8321"

ENVIRONMENT

BUILD_REGISTRY_SOURCES

BUILD_REGISTRY_SOURCES, if set, is treated as a JSON object which con? tains lists of registry names under the keys insecureRegistries, blockedRegistries, and allowedRegistries.

When pulling an image from a registry, if the name of the registry matches any of the items in the blockedRegistries list, the image pull attempt is denied. If there are registries in the allowedRegistries list, and the registry's name is not in the list, the pull attempt is denied.

TMPDIR The TMPDIR environment variable allows the user to specify where temporary files are stored while pulling and pushing images. Defaults to '/var/tmp'.

Files

.containerignore/.dockerignore

If the .containerignore/.dockerignore file exists in the context direc? tory, buildah build reads its contents. If both exist, then .container? ignore is used. Use the --ignorefile flag to override the ignore file path location. Buildah uses the content to exclude files and directo? ries from the context directory, when executing COPY and ADD directives in the Containerfile/Dockerfile

Users can specify a series of Unix shell globals in a

Buildah supports a special wildcard string ** which matches any number of directories (including zero). For example, */.go will exclude all files that end with .go that are found in all directories.

Example .containerignore file:

```
# exclude this content for image
*/*.c
**/output*
src
```

/.c Excludes files and directories whose names end with .c in any top level subdirectory. For example, the source file include/rootless.c.

**/output* Excludes files and directories starting with output from any directory.

src Excludes files named src and the directory src as well as any con? tent in it.

Lines starting with ! (exclamation mark) can be used to make exceptions to exclusions. The following is an example .containerignore/.dockerig? nore file that uses this mechanism:

```
*.doc
!Help.doc
```

Exclude all doc files except Help.doc from the image.

This functionality is compatible with the handling of .containerignore files described here:

https://github.com/containers/buildah/blob/main/docs/containerig? nore.5.md

registries.conf (/etc/containers/registries.conf)

registries.conf is the configuration file which specifies which con? tainer registries should be consulted when completing image names which do not include a registry or domain portion.

policy.json (/etc/containers/policy.json)

Signature policy file. This defines the trust policy for container im? ages. Controls which container registries can be used for image, and whether or not the tool should trust the images.

SEE ALSO

buildah(1), cpp(1), buildah-login(1), docker-login(1), namespaces(7), pid_namespaces(7), containers-policy.json(5), containers-reg? istries.conf(5), user_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.