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

NAME

buildah-from - Creates a new working container, either from scratch or using a specified image as a starting point.

SYNOPSIS

buildah from [options] image

DESCRIPTION

Creates a working container based upon the specified image name. If the supplied image name is "scratch" a new empty container is created. Image names use a "transport":"details" format.

Multiple transports are supported:

dir:path
An existing local directory path containing the manifest, layer tar? balls, and signatures in individual files. This is a non-standardized format, primarily useful for debugging or noninvasive image inspection.

docker://docker-reference (Default)
An image in a registry implementing the "Docker Registry HTTP API V2". By default, uses the authorization state in $XDG_RUN?

XDG_RUNTIME_DIR is not set, the default is /run/contain? ers/$UID/auth.json. If the authorization state is not found there, $HOME/.docker/config.json is checked, which is set using (docker lo? gin).

If docker-reference does not include a registry name, localhost will be consulted first, followed by any registries named in the registries configuration.

docker-archive:path

An image is retrieved as a podman load formatted file.

docker-daemon:docker-reference

An image docker-reference stored in the docker daemon's internal storage. docker-reference must include either a tag or a digest. Al? ternatively, when reading images, the format can also be docker-dae? mon:algo:digest (an image ID).

oci:path:tag**

An image tag in a directory compliant with "Open Container Image Lay? out Specification" at path.

oci-archive:path:tag

An image tag in a directory compliant with "Open Container Image Lay? out Specification" at path.

DEPENDENCIES

Buildah resolves the path to the registry to pull from by using the /etc/containers/registries.conf file, containers-registries.conf(5). If the buildah from command fails with an "image not known" error, first verify that the registries.conf file is installed and configured appropriately.

RETURN VALUE

The container ID of the container that was created. On error 1 is re? turned.

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.

--arch="ARCH"

Set the ARCH of the image to be pulled 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

--cap-add=CAP_xxx

Add the specified capability to the default set of capabilities which will be supplied for subsequent buildah run invocations which use this container. Certain capabilities are granted by default; this option can be used to add more.

--cap-drop=CAP_xxx

Remove the specified capability from the default set of capabilities

this container. The CAP_CHOWN, CAP_DAC_OVERRIDE, CAP_FOWNER, CAP_FSETID, CAP_KILL, CAP_NET_BIND_SERVICE, CAP_SETFCAP, CAP_SETGID, CAP_SETPCAP, and CAP_SETUID capabilities are granted by default; this option can be used to remove them. The list of default capabilities is managed in containers.conf(5).

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 IPC namespaces when the container is subse? quently used for buildah run. 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.

--cidfile ContainerIDFile

Write the container ID to the file.

--cpu-period=0

Limit the CPU CFS (Completely Fair Scheduler) period

Limit the container's CPU usage. This flag tells the kernel to restrict the container's CPU usage to the period you specify.

--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.

--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

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 or devices under a directory to the container. The format is <device-on-host>[:<device-on-container>][:<permissions>] (e.g. --device=/dev/sdc:/dev/xvdc:rwm)

--dns=[]

Set custom DNS servers

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

--dns-search=[]

Set custom DNS search domains

--format, -f oci | docker

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

--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.

--http-proxy

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.

Defaults to true

Sets the configuration for IPC namespaces when the container is subsequently used for buildah run. The configured value can be "" (the empty string) or "container" 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 processes under buildah run. 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

--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.

--name name

A name for the working container

--network=mode, --net=mode

Sets the configuration for network namespaces when the container is subsequently used for buildah run.

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.

network stack. This is the default for rootless containers. It is possible to specify these additional options, they can also be set with network_cmd_options in containers.conf:

? allow_host_loopback=true|false: Allow slirp4netns to reach the host loopback IP (default is 10.0.2.2 or the second IP from slirp4netns cidr subnet when changed, see the cidr op? tion below). The default is false.

? mtu=MTU: Specify the MTU to use for this network. (Default is 65520).

? cidr=CIDR: Specify ip range to use for this network. (De? fault is 10.0.2.0/24).

? enable_ipv6=true|false: Enable IPv6. Default is true. (Re? quired for outbound_addr6).

? outbound_addr=INTERFACE: Specify the outbound interface slirp binds to (ipv4 traffic only).

? outbound_addr=IPv4: Specify the outbound ipv4 address slirp binds to.

slirp binds to (ipv6 traffic only).

? outbound_addr6=IPv6: Specify the outbound ipv6 address slirp binds to.

? pasta[:OPTIONS,...]: use pasta(1) to create a user-mode net? working stack.
This is only supported in rootless mode.
By default, IPv4 and IPv6 addresses and routes, as well as the pod interface name, are copied from the host. If port forward? ing isn't configured, ports are forwarded dynamically as ser? vices are bound on either side (init namespace or container namespace). Port forwarding preserves the original source IP address. Options described in pasta(1) can be specified as comma-separated arguments.
In terms of pasta(1) options, --config-net is given by de? fault, in order to configure networking when the container is started, and --no-map-gw is also assumed by default, to avoid direct access from container to host using the gateway ad? dress. The latter can be overridden by passing --map-gw in the pasta-specific options (despite not being an actual pasta(1) option).
Also, -t none and -u none are passed to disable automatic port forwarding based on bound ports. Similarly, -T none and -U

tainer to host.

Some examples:

? pasta:--map-gw: Allow the container to directly reach the host using the gateway address.

? pasta:--mtu,1500: Specify a 1500 bytes MTU for the tap in? terface in the container.

? pasta:--ipv4-only,-a,10.0.2.0,-n,24,-g,10.0.2.2,--dns-for? ward,10.0.2.3,-m,1500,--no-ndp,--no-dhcpv6,--no-dhcp, equiv? alent to default slirp4netns(1) options: disable IPv6, as? sign 10.0.2.0/24 to the tap0 interface in the container, with gateway 10.0.2.3, enable DNS forwarder reachable at 10.0.2.3, set MTU to 1500 bytes, disable NDP, DHCPv6 and DHCP support.

? pasta:-I,tap0,--ipv4-only,-a,10.0.2.0,-n,24,-g,10.0.2.2,--dns- forward,10.0.2.3,--no-ndp,--no-dhcpv6,--no-dhcp, equivalent to default slirp4netns(1) options with Podman overrides: same as above, but leave the MTU to 65520 bytes

? pasta:-t,auto,-u,auto,-T,auto,-U,auto: enable automatic port forwarding based on observed bound ports from both host and

? pasta:-T,5201: enable forwarding of TCP port 5201 from con?
tainer to host, using the loopback interface instead of the
tap interface for improved performance

--os="OS"

Set the OS of the image to be pulled to the provided value instead of using the current operating system of the host.

--pid how

Sets the configuration for PID namespaces when the container is subse?
quently used for buildah run. The configured value can be "" (the empty string) or "container" to indicate that a new PID namespace should be created, or it can be "host" to indicate that the PID name?
space 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 image to be pulled to the provided value instead of using the current operating system and architecture of the host (for

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 from is happy to pull an image for any platform that ex? ists, buildah run will not be able to run binaries provided by that im? age 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.

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 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

If an image needs to be pulled from the registry, suppress progress output.

--retry attempts

Number of times to retry in case of failure when performing pull of im? ages from registry.

--retry-delay duration

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

Defaults to 2s.

--security-opt=[]

Security Options

"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" : Not supported

"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

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

the container

--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.

--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 during buildah run. 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)

"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)

--userns how

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

--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 mapping

Directly specifies a GID mapping which should be used to set ownership, at the filesystem level, on the container's contents. Commands run us? ing buildah run 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 triples of a starting

ber of consecutive IDs which the map entry represents.

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.

If none of --userns-uid-map-user, --userns-gid-map-group, or --userns-gid-map are specified, but --userns-uid-map is specified, the GID map will be set to use the same numeric values as the UID map.

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-gid-map-group group

Specifies that a GID mapping which should be used to set ownership, at the filesystem level, on the container's contents, can be found in en? tries in the /etc/subgid file which correspond to the specified group. Commands run using buildah run will default to being 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,

group name to use as the default setting for this option.

--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 mapping

Directly specifies a UID mapping which should be used to set ownership, at the filesystem level, on the container's contents. Commands run us?

configured using the UID and GID maps.

Entries in this map take the form of one or more triples of a starting in-container UID, a corresponding starting host-level UID, and the num? ber of consecutive IDs which the map entry represents.

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.

If none of --userns-uid-map-user, --userns-gid-map-group, or --userns-uid-map are specified, but --userns-gid-map is specified, the UID map will be set to use the same numeric values as the GID map.

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-user user

Specifies that a UID mapping which should be used to set ownership, at the filesystem level, on the container's contents, can be found in en?

Commands run using buildah run will default to being 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.

--uts how

Sets the configuration for UTS namespaces when the container is subse? quently used for buildah run. The configured value can be "" (the empty string) or "container" to indicate that a new UTS namespace should be created, or it can be "host" to indicate that the UTS name? space 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.

--variant=""

Set the architecture variant of the image to be pulled.

--volume, -v[=[HOST-DIR:CONTAINER-DIR[:OPTIONS]]]

Create a bind mount. If you specify, -v /HOST-DIR:/CONTAINER-DIR, Buil? dah

container. The OPTIONS are a comma delimited list and can be: [1] ?#Footnote1?

? [rw|ro]

? [U]

? [z|Z|O]

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

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

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?

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:

mounted into the container is labeled with the private label.

On SELinux systems, labels in the source directory need to 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, [r]private or [r]unbindablepropagation 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

umes, the source mount has to be either shared or slave. [1] ?#Foot? note1?

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.

EXAMPLE

buildah from --pull imagename

buildah from --pull docker://myregistry.example.com/imagename

buildah from docker-daemon:imagename:imagetag

buildah from --name mycontainer docker-archive:filename

buildah from oci-archive:filename

buildah from --name mycontainer dir:directoryname

buildah from --pull-always --name "mycontainer" myregistry.exam?ple.com/imagename

buildah from --tls-verify=false myregistry/myrepository/imagename:im?agetag

buildah from --creds=myusername:mypassword --cert-dir ~/auth myreg?istry/myrepository/imagename:imagetag

buildah from --authfile=/tmp/auths/myauths.json myregistry/myreposi?tory/imagename:imagetag

buildah from --memory 40m --cpu-shares 2 --cpuset-cpus 0,2 --security-opt label=level:s0:c100,c200 myregistry/myrepository/imagename:imagetag

buildah from --ulimit nofile=1024:1028 --cgroup-parent

buildah from --volume /home/test:/myvol:ro,Z myregistry/myreposi?
tory/imagename:imagetag

buildah from -v /home/test:/myvol:z,U myregistry/myrepository/image?
name:imagetag

buildah from -v /var/lib/yum:/var/lib/yum:O myregistry/myrepository/im?
agename:imagetag

buildah from --arch=arm --variant v7 myregistry/myrepository/image?
name:imagetag

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

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

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), buildah-pull(1), buildah-login(1), docker-login(1), name? spaces(7), pid_namespaces(7), containers-policy.json(5), containers-registries.conf(5), user_namespaces(7), containers.conf(5)

FOOTNOTES

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.