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 'podman-help.1' command

$ man podman-help.1

podman(1) General Commands Manual podman(1)

NAME

podman - Simple management tool for pods, containers and images

SYNOPSIS

podman [options] command

DESCRIPTION

Podman (Pod Manager) is a fully featured container engine that is a simple daemonless tool. Podman provides a Docker-CLI comparable com? mand line that eases the transition from other container engines and allows the management of pods, containers and images. Simply put: alias docker=podman. Most Podman commands can be run as a regular user, without requiring additional privileges.

Podman uses Buildah(1) internally to create container images. Both tools share image (not container) storage, hence each can use or manip? ulate images (but not containers) created by the other.

Default settings for flags are defined in containers.conf. Most set? tings for Remote connections use the server's containers.conf, except when documented in man pages.

podman [GLOBAL OPTIONS]

GLOBAL OPTIONS

--cgroup-manager=manager

The CGroup manager to use for container cgroups. Supported values are cgroupfs or systemd. Default is systemd unless overridden in the con?

tainers.conf file.

Note: Setting this flag can cause certain commands to break when called on containers previously created by the other CGroup manager type.

Note: CGroup manager is not supported in rootless mode when using CGroups Version V1.

--conmon

Path of the conmon binary (Default path is configured in contain? ers.conf)

--connection, -c

Connection to use for remote podman, including Mac and Windows (exclud? ing WSL2) machines, (Default connection is configured in contain? ers.conf) Setting this option will switch the --remote option to true.

Remote connections use local containers.conf for default.

--events-backend=type

Backend to use for storing events. Allowed values are file, journald, and none. When file is specified, the events are stored under <tm? pdir>/events/events.log (see --tmpdir below).

--help, -h

Print usage statement

--hooks-dir=path

Each *.json file in the path configures a hook for Podman containers. For more details on the syntax of the JSON files and the semantics of hook injection, see oci-hooks(5). Podman and libpod 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 (oci-hooks(5) discusses directory precedence).

For the annotation conditions, libpod 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 libpod inserts by default (e.g. /dev/shm) are not considered.

If --hooks-dir is unset for root callers, Podman and libpod will cur?

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

Podman and libpod currently support an additional precreate state which is called before the runtime's create operation. Unlike the other stages, which receive the container state on their standard input, precreate hooks receive the proposed runtime configuration on their standard input. They may alter that configuration as they see fit, and write the altered form to their standard output.

WARNING: the precreate hook allows powerful changes to occur, such as adding additional mounts to the runtime configuration. That power also makes it easy to break things. Before reporting libpod errors, try running a container with precreate hooks disabled to see if the problem is due to one of the hooks.

--identity=path

Path to ssh identity file. If the identity file has been encrypted, podman prompts the user for the passphrase. If no identity file is provided and no user is given, podman defaults to the user running the podman command. Podman prompts for the login password on the remote server.

Identity value resolution precedence:

- command line value
- environment variable CONTAINER_SSHKEY, if CONTAINER_HOST is found
- containers.conf Remote connections use local containers.conf for de?
fault.

--log-level=level

Log messages at and above specified level: debug, info, warn, error, fatal or panic (default: "warn")

--namespace=namespace

Set libpod namespace. Namespaces are used to separate groups of con?
tainers and pods in libpod's state. When namespace is set, created containers and pods will join the given namespace, and only containers

and pods in the given namespace will be visible to Podman.

--network-cmd-path=path

Path to the command binary to use for setting up a network. It is cur? rently only used for setting up a slirp4netns(1) or pasta(1) network. If "" is used then the binary is looked up using the $PATH environment variable.

--network-config-dir=directory

Path to the directory where network configuration files are located. For the netavark backend "/etc/containers/networks" is used as root and "$graphroot/networks" as rootless. For the CNI backend the default is "/etc/cni/net.d" as root and "$HOME/.config/cni/net.d" as rootless. CNI will be deprecated from Podman in the future for netavark.

--noout

Redirect stdout to /dev/null. This command will prevent all stdout from the Podman command. The --noout option will not block stderr or stdout from containers.

--remote, -r

When true, access to the Podman service will be remote. Defaults to false. Settings can be modified in the containers.conf file. If the CONTAINER_HOST environment variable is set, the --remote option de? faults to true.

--root=value

Storage root dir in which data, including images, is stored (default: "/var/lib/containers/storage" for UID 0, "$HOME/.local/share/contain? ers/storage" for other users). Default root dir configured in contain? ers-storage.conf(5).

Overriding this option will cause the storage-opt settings in contain? ers-storage.conf(5) to be ignored. The user must specify additional options via the --storage-opt flag.

--runroot=value

Storage state directory where all state information is stored (default: "/run/containers/storage" for UID 0, "/run/user/$UID/run" for other users). Default state dir configured in containers-storage.conf(5).

--runtime=value

Name of the OCI runtime as specified in containers.conf or absolute path to the OCI compatible binary used to run containers.

--runtime-flag=flag

Adds global flags for the container runtime. To list the supported flags, please consult the manpages of the selected container runtime (runc is the default runtime, the manpage to consult is runc(8). When the machine is configured for cgroup V2, the default runtime is crun, the manpage to consult is crun(8).).

Note: Do not pass the leading -- to the flag. To pass the runc flag --log-format json to podman build, the option given would be --runtime-flag log-format=json.

--ssh=value

This option allows the user to change the ssh mode, meaning that rather than using the default golang mode, one can instead use --ssh=native to use the installed ssh binary and config file declared in contain? ers.conf.

--storage-driver=value

Storage driver. The default storage driver for UID 0 is configured in containers-storage.conf(5) in rootless mode), and is vfs for non-root users when fuse-overlayfs is not available. The STORAGE_DRIVER envi? ronment variable overrides the default. The --storage-driver specified driver overrides all.

Overriding this option will cause the storage-opt settings in contain? ers-storage.conf(5) to be ignored. The user must specify additional options via the --storage-opt flag.

--storage-opt=value

Specify a storage driver option. Default storage driver options are configured in containers-storage.conf(5). The STORAGE_OPTS environment variable overrides the default. The --storage-opt specified options override all. Specify --storage-opt="" so no storage options will be used.

--syslog

Output logging information to syslog as well as the console (default false).

On remote clients, including Mac and Windows (excluding WSL2) machines, logging is directed to the file $HOME/.config/containers/podman.log.

--tmpdir=path

Path to the tmp directory, for libpod runtime content. Defaults to $XDG_RUNTIME_DIR/libpod/tmp as rootless and /run/libpod/tmp as rootful.

NOTE --tmpdir is not used for the temporary storage of downloaded im? ages. Use the environment variable TMPDIR to change the temporary storage location of downloaded container images. Podman defaults to use /var/tmp.

--transient-store

Enables a global transient storage mode where all container metadata is stored on non-persistent media (i.e. in the location specified by --runroot). This mode allows starting containers faster, as well as guaranteeing a fresh state on boot in case of unclean shutdowns or other problems. However it is not compatible with a traditional model where containers persist across reboots.

Default value for this is configured in containers-storage.conf(5).

--url=value

URL to access Podman service (default from containers.conf, rootless unix://run/user/$UID/podman/podman.sock or as root unix://run/pod? man/podman.sock). Setting this option will switch the --remote option to true.

? CONTAINER_HOST is of the format <schema>://[<user[:<pass? word>]@]<host>[:<port>][<path>]

Details:

- schema is one of:

* ssh (default): a local unix(7) socket on the named host and port, reachable via SSH

* tcp: an unencrypted, unauthenticated TCP connection to the named host and port

* unix: a local unix(7) socket at the specified path, or the default

for the user

- user will default to either root or the current running user (ssh only)

- password has no default (ssh only)

- host must be provided and is either the IP or name of the machine hosting the Podman service (ssh and tcp)

- port defaults to 22 (ssh and tcp)

- path defaults to either /run/podman/podman.sock, or /run/user/$UID/podman/podman.sock if running rootless (unix), or must be explicitly specified (ssh)

URL value resolution precedence:

- command line value

- environment variable CONTAINER_HOST

- containers.conf service_destinations table

- unix://run/podman/podman.sock

Remote connections use local containers.conf for default.

Some example URL values in valid formats:

- unix://run/podman/podman.sock

- unix://run/user/$UID/podman/podman.sock

- ssh://notroot@localhost:22/run/user/$UID/podman/podman.sock

- ssh://root@localhost:22/run/podman/podman.sock

- tcp://localhost:34451

- tcp://127.0.0.1:34451

--version, -v

Print the version

--volumepath=value

Volume directory where builtin volume information is stored (default: "/var/lib/containers/storage/volumes" for UID 0, "$HOME/.lo?cal/share/containers/storage/volumes" for other users). Default volume path can be overridden in containers.conf.

Environment Variables

Podman can set up environment variables from env of [engine] table in containers.conf. These variables can be overridden by passing environ?

ment variables before the podman commands.

CONTAINERS_CONF

Set default locations of containers.conf file

CONTAINERS_REGISTRIES_CONF

Set default location of the registries.conf file.

CONTAINERS_STORAGE_CONF

Set default location of the storage.conf file.

CONTAINER_CONNECTION

Override default --connection value to access Podman service. Also en?

abled --remote option.

CONTAINER_HOST

Set default --url value to access Podman service. Also enabled --remote

option.

CONTAINER_SSHKEY

Set default --identity path to ssh key file value used to access Podman

service.

STORAGE_DRIVER

Set default --storage-driver value.

STORAGE_OPTS

Set default --storage-opts value.

TMPDIR

Set the temporary storage location of downloaded container images. Pod?

man defaults to use /var/tmp.

XDG_CONFIG_HOME

In Rootless mode configuration files are read from XDG_CONFIG_HOME when

specified, otherwise in the home directory of the user under

$HOME/.config/containers.

XDG_DATA_HOME

In Rootless mode images are pulled under XDG_DATA_HOME when specified,

otherwise in the home directory of the user under $HOME/.lo?

cal/share/containers/storage.

XDG_RUNTIME_DIR

In Rootless mode temporary configuration data is stored in ${XDG_RUN?

TIME_DIR}/containers.

Remote Access

The Podman command can be used with remote services using the --remote flag. Connections can be made using local unix domain sockets, ssh or directly to tcp sockets. When specifying the podman --remote flag, only the global options --url, --identity, --log-level, --connection are used.

Connection information can also be managed using the containers.conf file.

Exit Codes

The exit code from podman gives information about why the container failed to run or why it exited. When podman commands exit with a non-zero code, the exit codes follow the chroot standard, see below:

125 The error is with podman itself

```
$ podman run --foo busybox; echo $?
Error: unknown flag: --foo
125
```

126 Executing a contained command and the command cannot be invoked

```
$ podman run busybox /etc; echo $?
  Error: container_linux.go:346: starting container process caused "exec: \"/etc\": permission denied": OCI runtime error
126
```

127 Executing a contained command and the command cannot be found

```
$ podman run busybox foo; echo $?
Error: container_linux.go:346: starting container process caused "exec: \"foo\": executable file not found in $PATH": OCI runtime error
127
```

Exit code contained command exit code

```
$ podman run busybox /bin/sh -c 'exit 3'; echo $?
3
```

COMMANDS

Command	Description

??

?podman-attach(1) ? Attach to a running container. ?

??

?podman-auto-update(1) ? Auto update containers accord? ?

? ? ing to their auto-update pol? ?

? ? icy ?

??

?podman-build(1) ? Build a container image using ?

? ? a Containerfile. ?

??

?podman-commit(1) ? Create new image based on the ?

? ? changed container. ?

??

?podman-completion(1) ? Generate shell completion ?

? ? scripts ?

??

?podman-container(1) ? Manage containers. ?

??

?podman-cp(1) ? Copy files/folders between a ?

? ? container and the local ?

? ? filesystem. ?

??

?podman-create(1) ? Create a new container. ?

??

?podman-diff(1) ? Inspect changes on a container ?

? ? or image's filesystem. ?

??

?podman-events(1) ? Monitor Podman events ?

??

?podman-exec(1) ? Execute a command in a running ?

? ? container. ?

??

?podman-export(1) ? Export a container's filesys? ?

? ? tem contents as a tar archive. ?

??

?podman-generate(1) ? Generate structured data based ?

? ? on containers, pods or vol? ?

? ? umes. ?

??

?podman-healthcheck(1) ? Manage healthchecks for con? ?

? ? tainers ?

??

?podman-history(1) ? Show the history of an image. ?

??

?podman-image(1) ? Manage images. ?

??

?podman-images(1) ? List images in local storage. ?

??

?podman-import(1) ? Import a tarball and save it ?

? ? as a filesystem image. ?

??

?podman-info(1) ? Displays Podman related system ?

? ? information. ?

??

?podman-init(1) ? Initialize one or more con? ?

? ? tainers ?

??

?podman-inspect(1) ? Display a container, image, ?

? ? volume, network, or pod's con? ?

? ? figuration. ?

??

?podman-kill(1) ? Kill the main process in one ?

? ? or more containers. ?

??

?podman-load(1) ? Load image(s) from a tar ar? ?

? ? chive into container storage. ?

??

?podman-login(1) ? Login to a container registry. ?

??

?podman-logout(1) ? Logout of a container reg? ?

? ? istry. ?

??

?podman-logs(1) ? Display the logs of one or ?

? ? more containers. ?

??

?podman-machine(1) ? Manage Podman's virtual ma? ?

? ? chine ?

??

?podman-manifest(1) ? Create and manipulate manifest ?

? ? lists and image indexes. ?

??

?podman-mount(1) ? Mount a working container's ?

? ? root filesystem. ?

??

?podman-network(1) ? Manage Podman networks. ?

??

?podman-pause(1) ? Pause one or more containers. ?

??

?podman-kube(1) ? Play containers, pods or vol? ?

? ? umes based on a structured in? ?

? ? put file. ?

??

?podman-pod(1) ? Management tool for groups of ?

? ? containers, called pods. ?

??

?podman-port(1) ? List port mappings for a con? ?

? ? tainer. ?

??

?podman-ps(1) ? Prints out information about ?

? ? containers. ?

??

?podman-pull(1) ? Pull an image from a registry. ?

??

?podman-push(1) ? Push an image, manifest list ?

? ? or image index from local ?

? ? storage to elsewhere. ?

??

?podman-rename(1) ? Rename an existing container. ?

??

?podman-restart(1) ? Restart one or more contain? ?

? ? ers. ?

??

?podman-rm(1) ? Remove one or more containers. ?

??

?podman-rmi(1) ? Removes one or more locally ?

? ? stored images. ?

??

?podman-run(1) ? Run a command in a new con? ?

? ? tainer. ?

??

?podman-save(1) ? Save image(s) to an archive. ?

??

?podman-search(1) ? Search a registry for an im? ?

? ? age. ?

??

?podman-secret(1) ? Manage podman secrets. ?

??

?podman-start(1) ? Start one or more containers. ?

??

?podman-stats(1) ? Display a live stream of one ?

? ? or more container's resource ?

? ? usage statistics. ?

??

?podman-stop(1) ? Stop one or more running con? ?

? ? tainers. ?

??

?podman-system(1) ? Manage podman. ?

??

?podman-tag(1) ? Add an additional name to a ?

? ? local image. ?

??

?podman-top(1) ? Display the running processes ?

? ? of a container. ?

??

?podman-unmount(1) ? Unmount a working container's ?

? ? root filesystem. ?

??

?podman-unpause(1) ? Unpause one or more contain? ?

? ? ers. ?

??

?podman-unshare(1) ? Run a command inside of a mod? ?

? ? ified user namespace. ?

??

?podman-untag(1) ? Removes one or more names from ?

? ? a locally-stored image. ?

??

?podman-update(1) ? Updates the cgroup configura? ?

? ? tion of a given container. ?

??

?podman-version(1) ? Display the Podman version in? ?

? ? formation. ?

??

?podman-volume(1) ? Simple management tool for ?

? ? volumes. ?

??

?podman-wait(1) ? Wait on one or more containers ?

? ? to stop and print their exit ?

? ? codes. ?

??

CONFIGURATION FILES

containers.conf (/usr/share/containers/containers.conf, /etc/contain?

ers/containers.conf, $HOME/.config/containers/containers.conf)

Podman has builtin defaults for command line options. These defaults

can be overridden using the containers.conf configuration files.

Distributions ship the /usr/share/containers/containers.conf file with

their default settings. Administrators can override fields in this file

by creating the /etc/containers/containers.conf file. Users can fur?

ther modify defaults by creating the $HOME/.config/containers/contain?

ers.conf file. Podman merges its builtin defaults with the specified

fields from these files, if they exist. Fields specified in the users

file override the administrator's file, which overrides the distribu?

tion's file, which override the built-in defaults.

Podman uses builtin defaults if no containers.conf file is found.

If the CONTAINERS_CONF environment variable is set, then its value is

used for the containers.conf file rather than the default.

mounts.conf (/usr/share/containers/mounts.conf)

The mounts.conf file specifies volume mount directories that are auto?

matically mounted inside containers when executing the podman run or

podman start commands. Administrators can override the defaults file by

creating /etc/containers/mounts.conf.

When Podman runs in rootless mode, the file $HOME/.config/contain?

ers/mounts.conf will override the default if it exists. Please refer to

containers-mounts.conf(5) for further details.

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

Signature verification policy files are used to specify policy, e.g.

trusted keys, applicable when deciding whether to accept an image, or

individual signatures of that image, as valid.

registries.conf (/etc/containers/registries.conf, $HOME/.config/con?

tainers/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.

Non root users of Podman can create the $HOME/.config/containers/reg?

istries.conf file to be used instead of the system defaults.

If the CONTAINERS_REGISTRIES_CONF environment variable is set, then its

value is used for the registries.conf file rather than the default.

storage.conf (/etc/containers/storage.conf, $HOME/.config/contain?

ers/storage.conf)

storage.conf is the storage configuration file for all tools using con?

tainers/storage

The storage configuration file specifies all of the available container

storage options for tools using shared container storage.

When Podman runs in rootless mode, the file $HOME/.config/contain?

ers/storage.conf is used instead of the system defaults.

If the CONTAINERS_STORAGE_CONF environment variable is set, then its

value is used for the storage.conf file rather than the default.

Rootless mode

Podman can also be used as non-root user. When podman runs in rootless

mode, a user namespace is automatically created for the user, defined

in /etc/subuid and /etc/subgid.

Containers created by a non-root user are not visible to other users

and are not seen or managed by Podman running as root.

It is required to have multiple uids/gids set for a user. Be sure the

user is present in the files /etc/subuid and /etc/subgid.

Execute the following commands to add the ranges to the files

```
$ sudo usermod --add-subuids 10000-75535 USERNAME
$ sudo usermod --add-subgids 10000-75535 USERNAME
```

Or just add the content manually.

```
$ echo USERNAME:10000:65536 >> /etc/subuid
$ echo USERNAME:10000:65536 >> /etc/subgid
```

See the subuid(5) and subgid(5) man pages for more information.

Images are pulled under XDG_DATA_HOME when specified, otherwise in the home directory of the user under .local/share/containers/storage.

Currently slirp4netns or pasta is required to be installed to create a network device, otherwise rootless containers need to run in the net? work namespace of the host.

In certain environments like HPC (High Performance Computing), users cannot take advantage of the additional UIDs and GIDs from the /etc/subuid and /etc/subgid systems. However, in this environment, rootless Podman can operate with a single UID. To make this work, set the ignore_chown_errors option in the containers-storage.conf(5) file.

This option tells Podman when pulling an image to ignore chown errors when attempting to change a file in a container image to match the non-root UID in the image. This means all files get saved as the user's UID. Note this could cause issues when running the container.

NOTE: Unsupported file systems in rootless mode

The Overlay file system (OverlayFS) is not supported with kernels prior to 5.12.9 in rootless mode. The fuse-overlayfs package is a tool that provides the functionality of OverlayFS in user namespace that allows mounting file systems in rootless environments. It is recommended to install the fuse-overlayfs package. In rootless mode, Podman will au? tomatically use the fuse-overlayfs program as the mount_program if in? stalled, as long as the $HOME/.config/containers/storage.conf file was not previously created. If storage.conf exists in the homedir, add mount_program = "/usr/bin/fuse-overlayfs" under [storage.options.over? lay] to enable this feature.

The Network File System (NFS) and other distributed file systems (for example: Lustre, Spectrum Scale, the General Parallel File System (GPFS)) are not supported when running in rootless mode as these file systems do not understand user namespace. However, rootless Podman can make use of an NFS Homedir by modifying the $HOME/.config/contain? ers/storage.conf to have the graphroot option point to a directory stored on local (Non NFS) storage.

For more information, please refer to the Podman Troubleshooting Page.

SEE ALSO

containers-mounts.conf(5), containers.conf(5), containers-reg?istries.conf(5), containers-storage.conf(5), buildah(1), oci-hooks(5), containers-policy.json(5), crun(1), runc(8), subuid(5), subgid(5), slirp4netns(1), pasta(1), conmon(8)

HISTORY

Dec 2016, Originally compiled by Dan Walsh dwalsh@redhat.com ?mailto:dwalsh@redhat.com?

podman(1)