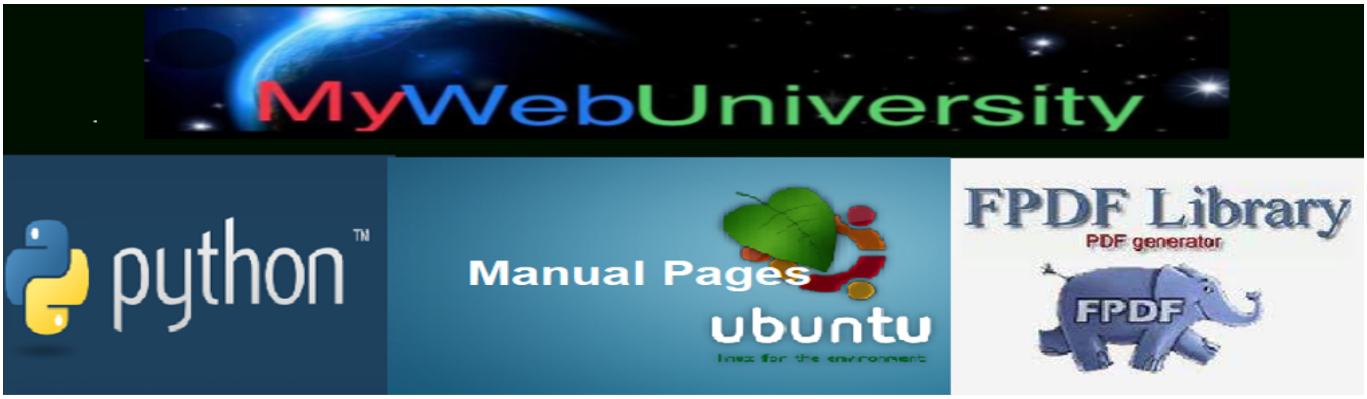

Rocky Enterprise Linux 9.2 Manual Pages on command 'docker-cp.1'

$ man docker-cp.1

NAME

podman-cp - Copy files/folders between a container and the local filesystem

SYNOPSIS

podman cp [options] [container:]src_path [container:]dest_path

podman container cp [options] [container:]src_path [container:]dest_path

DESCRIPTION

podman cp allows copying the contents of src_path to the dest_path. Files can be copied from a container to the local machine and vice versa or between two containers. If - is specified for either the SRC_PATH or DEST_PATH, one can also stream a tar archive from STDIN or to STDOUT.

The containers can be either running or stopped and the src_path or dest_path can be a file or directory.

IMPORTANT: The podman cp command assumes container paths are relative to the container's root directory (/), which means supplying the initial forward slash is optional and there? fore sees compassionate_darwin:/tmp/foo/myfile.txt and compassionate_darwin:tmp/foo/my? file.txt as identical.

Local machine paths can be an absolute or relative value. The command interprets a local machine's relative paths as relative to the current working directory where podman cp is run.

Assuming a path separator of /, a first argument of src_path and second argument of dest_path, the behavior is as follows:

src_path specifies a file:

- dest_path does not exist

- the file is saved to a file created at dest_path (note that parent directory must exist).

- dest_path exists and is a file

- the destination is overwritten with the source file's contents.

- dest_path exists and is a directory

- the file is copied into this directory using the base name from src_path.

src_path specifies a directory:

- dest_path does not exist

- dest_path is created as a directory and the contents of the source directory are copied into this directory.

- dest_path exists and is a file

- Error condition: cannot copy a directory to a file.

- dest_path exists and is a directory

- src_path ends with /

- the source directory is copied into this directory.

- src_path ends with /. (i.e., slash followed by dot)

- the content of the source directory is copied into this directory.

The command requires src_path and dest_path to exist according to the above rules.

If src_path is local and is a symbolic link, the symbolic target, is copied by default.

A colon (:) is used as a delimiter between a container and its path, it can also be used when specifying paths to a src_path or dest_path on a local machine, for example, file:name.txt.

IMPORTANT: while using a colon (:) in a local machine path, one must be explicit with a relative or absolute path, for example: /path/to/file:name.txt or ./file:name.txt

Using - as the src_path streams the contents of STDIN as a tar archive. The command ex? tracts the content of the tar to the DEST_PATH in the container. In this case, dest_path must specify a directory. Using - as the dest_path streams the contents of the resource (can be a directory) as a tar archive to STDOUT.

Note that podman cp ignores permission errors when copying from a running rootless con? tainer. The TTY devices inside a rootless container are owned by the host's root user and hence cannot be read inside the container's user namespace.

Further note that podman cp does not support globbing (e.g., cp dir/*.txt). If you want

to copy multiple files from the host to the container you may use xargs(1) or find(1) (or similar tools for chaining commands) in conjunction with podman cp. If you want to copy multiple files from the container to the host, you may use podman mount CONTAINER and op? erate on the returned mount point instead (see ALTERNATIVES below).

OPTIONS

--archive, -a=true | false

Archive mode (copy all uid/gid information). When set to true, files copied to a con? tainer will have changed ownership to the primary UID/GID of the container. When set to false, maintain uid/gid from archive sources instead of changing them to the primary uid/gid of the destination container. The default is true.

ALTERNATIVES

Podman has much stronger capabilities than just podman cp to achieve copying files between the host and containers.

Using standard podman-mount(1) and podman-unmount(1) takes advantage of the entire linux tool chain, rather than just cp.

copying contents out of a container or into a container, can be achieved with a few simple commands. For example:

To copy the /etc/foobar directory out of a container and onto /tmp on the host, the fol? lowing commands can be executed:

```
mnt=$(podman mount CONTAINERID)
cp -R ${mnt}/etc/foobar /tmp
podman umount CONTAINERID
```

To untar a tar ball into a container, following commands can be executed:

```
mnt=$(podman mount CONTAINERID)
tar xf content.tgz -C ${mnt}
podman umount CONTAINERID
```

To install a package into a container that does not have dnf installed, following commands can be executed:

```
mnt=$(podman mount CONTAINERID)
dnf install --installroot=${mnt} httpd
chroot ${mnt} rm -rf /var/log/dnf /var/cache/dnf
podman umount CONTAINERID
```

By using podman mount and podman unmount, one can use all of the standard linux tools for

moving files into and out of containers, not just the cp command.

EXAMPLES

? Copy a file from host to a container.

```
podman cp /myapp/app.conf containerID:/myapp/app.conf
```

? Copy a file from a container to a directory on another container.

```
podman cp containerID1:/myfile.txt containerID2:/tmp
```

? Copy a directory on a container to a directory on the host.

```
podman cp containerID:/myapp/ /myapp/
```

? Copy the contents of a directory on a container to a directory on the host.

```
podman cp containerID:/home/myuser/. /home/myuser/
```

? Copy a directory on a container into a directory on another.

```
podman cp containerA:/myapp containerB:/yourapp
```

? Stream a tar archive from STDIN to a container.

```
podman cp - containerID:/myfiles.tar.gz < myfiles.tar.gz
```

SEE ALSO

podman(1), podman-mount(1), podman-unmount(1)

podman-cp(1)()