MyWebUniversity
python
Manual Pages
ubuntu
FPDF Library
PDF generator
FPDF
Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 22.4.5 Manual Pages on command 'docker-container-cp.1'

$ man docker-container-cp.1

DOCKER(1) Docker User Manuals DOCKER(1)

NAME

docker-container-cp - Copy files/folders between a container and the local filesys?

tem

SYNOPSIS

docker container cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|- docker cp [OP?

TIONS] SRC_PATH|- CONTAINER:DEST_PATH

DESCRIPTION

The docker container cp utility copies the contents of SRC_PATH to the DEST_PATH.

You can copy from the container's file system to the local machine or the reverse,

from the local filesystem to the container. If - is specified for either the

SRC_PATH or DEST_PATH, you can also stream a tar archive from STDIN or to STDOUT.

The CONTAINER can be a running or stopped container. The SRC_PATH or DEST_PATH can

be a file or directory.

The docker container cp command assumes container paths are relative to the con?

tainer's / (root) directory. This means supplying the initial forward slash is op?

tional; The command sees compassionate_darwin:/tmp/foo/myfile.txt and compassion?

ate_darwin:tmp/foo/myfile.txt as identical. Local machine paths can be an absolute

or relative value. The command interprets a local machine's relative paths as rela?

tive to the current working directory where docker container cp is run.

The cp command behaves like the Unix cp -a command in that directories are copied

recursively with permissions preserved if possible. Ownership is set to the user

and primary group at the destination. For example, files copied to a container are
created with UID:GID of the root user. Files copied to the local machine are cre?
ated with the UID:GID of the user which invoked the docker container cp command.
If you specify the -L option, docker container cp follows any symbolic link in the
SRC_PATH. docker container cp does not create parent directories for DEST_PATH if
they do not exist.
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
      ? DEST_PATH does not exist and ends with /
        ? Error condition: the destination 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 basename 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 di?
          rectory 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 does not end with /. (that is: slash followed by dot)
          ? the source directory is copied into this directory
        ? SRC_PATH does end with /. (that is: 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 link, not the target, is
copied by default. To copy the link target and not the link, specify the -L option.
A colon (:) is used as a delimiter between CONTAINER and its path. You can also use

: when specifying paths to a SRC_PATH or DEST_PATH on a local machine, for example file:name.txt. If you use a : in a local machine path, you must be explicit with a relative or absolute path, for example:

```
`/path/to/file:name.txt` or `./file:name.txt`
```

It is not possible to copy certain system files such as resources under /proc, /sys, /dev, tmpfs, and mounts created by the user in the container. However, you can still copy such files by manually running tar in docker exec. For example (consider SRC_PATH and DEST_PATH are directories):

```
$ docker exec foo tar Ccf $(dirname SRC_PATH) - $(basename SRC_PATH) | tar Cxf DEST_PATH -
```

or

```
$ tar Ccf $(dirname SRC_PATH) - $(basename SRC_PATH) | docker exec -i foo tar Cxf DEST_PATH -
```

Using - as the SRC_PATH streams the contents of STDIN as a tar archive. The com? mand extracts the content of the tar to the DEST_PATH in container's filesystem. In this case, DEST_PATH must specify a directory. Using - as the DEST_PATH streams the contents of the resource as a tar archive to STDOUT.

EXAMPLES

Suppose a container has finished producing some output as a file it saves to some? where in its filesystem. This could be the output of a build job or some other com? putation. You can copy these outputs from the container to a location on your local host.

If you want to copy the /tmp/foo directory from a container to the existing /tmp directory on your host. If you run docker container cp in your ~ (home) directory on the local host:

```
$ docker container cp compassionate_darwin:tmp/foo /tmp
```

Docker creates a /tmp/foo directory on your host. Alternatively, you can omit the leading slash in the command. If you execute this command from your home directory:

```
$ docker container cp compassionate_darwin:tmp/foo tmp
```

If ~/tmp does not exist, Docker will create it and copy the contents of /tmp/foo from the container into this new directory. If ~/tmp already exists as a directory, then Docker will copy the contents of /tmp/foo from the container into a directory at ~/tmp/foo.

When copying a single file to an existing LOCALPATH, the docker container cp com? mand will either overwrite the contents of LOCALPATH if it is a file or place it into LOCALPATH if it is a directory, overwriting an existing file of the same name if one exists. For example, this command:

```
$ docker container cp sharp_ptolemy:/tmp/foo/myfile.txt /test
```

If /test does not exist on the local machine, it will be created as a file with the contents of /tmp/foo/myfile.txt from the container. If /test exists as a file, it will be overwritten. Lastly, if /test exists as a directory, the file will be copied to /test/myfile.txt.

Next, suppose you want to copy a file or folder into a container. For example, this could be a configuration file or some other input to a long running computation that you would like to place into a created container before it starts. This is useful because it does not require the configuration file or other input to exist in the container image.

If you have a file, config.yml, in the current directory on your local host and wish to copy it to an existing directory at /etc/my-app.d in a container, this com? mand can be used:

```
$ docker container cp config.yml myappcontainer:/etc/my-app.d
```

If you have several files in a local directory /config which you need to copy to a directory /etc/my-app.d in a container:

```
$ docker container cp /config/. myappcontainer:/etc/my-app.d
```

The above command will copy the contents of the local /config directory into the directory /etc/my-app.d in the container.

Finally, if you want to copy a symbolic link into a container, you typically want to copy the linked target and not the link itself. To copy the target, use the -L option, for example:

```
$ ln -s /tmp/somefile /tmp/somefile.ln
$ docker container cp -L /tmp/somefile.ln myappcontainer:/tmp/
```

This command copies content of the local /tmp/somefile into the file /tmp/some? file.ln in the container. Without -L option, the /tmp/somefile.ln preserves its symbolic link but not its content.

OPTIONS

-a, --archive[=false]   Archive mode (copy all uid/gid information)

-L, --follow-link[=false]   Always follow symbol link in SRC_PATH

-h, --help[=false]   help for cp

SEE ALSO

docker-container(1)