Cheat sheet

Buildah

Buildah is a tool for creating OCI-compliant container images. Also, Buildah provides the capability to create a container based on a particular container image, update the contents of that container, and then create a brand new container image based on the altered container.

Although you can use Buildah to run a container on a local machine in a limited manner, the tool's fundamental intention is to provide versatility for creating container images and pushing them to a container registry.

The sections that follow show you how to use Buildah to work with existing container images and build new ones. There is also a section that goes over the basics of pushing a container to a container registry.

The `$` symbol in the examples in the sections below represents the console prompt for a terminal window.

Installing buildah

```
yum -y install buildah
```

Or

```
dnf -y install buildah
```

Working with container images

The following sections describe how to:

- List container images.
- Pull a container image.
- Create a container image.
- Delete a container image.

There is also a section that demonstrates how to create a new container image based on an existing container image.

List all local container images

```
buildah images
```

Example:

The following example demonstrates how to list container images stored on the local machine:

```
$ buildah images

REPOSITORY                       TAG           IMAGE ID      CREATED
SIZE
docker.io/library/busybox        latest        1a80408de790  5 weeks ago
1.46 MB
quay.io/app-sre/ubi8-nodejs-14   latest        528baa338298  8 months ago
659 MB
docker.io/library/node           12.18-alpine  e13d60032d4d  19 months
ago   93.8 MB
docker.io/reselbob/pinger        latest        c5fa4df9cfe4  3 years ago
74.8 MB
```

Pulling a container image

```
buildah from <repo>/<container_image_name>:<tag>
```

Example:

The following example pulls the container image `busybox:latest` from the remote registry `docker.io`:

```
$ buildah from docker.io/busybox:latest

Trying to pull docker.io/library/busybox:latest...
Getting image source signatures
Copying blob 50e8d59317eb done
Copying config 1a80408de7 done
Writing manifest to image destination
Storing signatures
1a80408de790c0b1075d0a7e23ff7da78b311f85f36ea10098e4a6184c200964
```

Deleting a container image from a local machine

```
buildah rmi <repo>/<container_image_name>:<tag>
```

or

```
buildah rmi <container_image_id>
```

Example:

The following example demonstrates how delete a container image with the name `buildah rmi docker.io/library/busybox`:

```
$ buildah rmi docker.io/library/busybox
untagged: docker.io/library/busybox:latest
1a80408de790c0b1075d0a7e23ff7da78b311f85f36ea10098e4a6184c200964
```

Deleting all container images from local machine

```
buildah rmi --all
```

Example:

The following example demonstrates how to remove all container images stored on the local machine:

```
$ buildah rmi --all

untagged: docker.io/library/node:12.18-alpine
untagged: quay.io/app-sre/ubi8-nodejs-14:latest
untagged: docker.io/reselbob/pinger:latest
untagged: docker.io/library/busybox:latest
untagged: docker.io/library/httpd:latest
untagged: registry.access.redhat.com/ubi8/ubi:latest
e13d60032d4d14e88485db13b65a7e38b2588bc3101456278b5e2daddec7e862
528baa33829859e2b854b1f7c2356a8223e449aff78d18ee4cc2fad298199611
c5fa4df9cfe436469dab3d89be4dafcbfb61bc9e594778e12de02ee89ca7fa9a
1a80408de790c0b1075d0a7e23ff7da78b311f85f36ea10098e4a6184c200964
c58ef9bfbb5789a9882cee610ba778b1368d21b513d6caf32e3075542e13fe81
1264065f6ae851d6a33d7be03ffde100356592e385b9b72f65f91b5d9b944b92
```

Building a container image

```
buildah bud -t <image_name> <container_file_path>
```

Example:

The following example creates a container file and then builds a container image using that file.

Create the container file:

```
$ echo "echo This container works!" > myecho

$ chmod 755 myecho

$ cat << 'EOF' > Containerfile
FROM registry.access.redhat.com/ubi8/ubi
ADD myecho /tmp
ENTRYPOINT "/tmp/myecho"
EOF

$ buildah bud -t myecho-image Containerfile
```

Build the container image:

```
STEP 1/3: FROM registry.access.redhat.com/ubi8/ubi
STEP 2/3: ADD myecho /tmp
STEP 3/3: ENTRYPOINT "/tmp/myecho"
COMMIT myecho_image
Getting image source signatures
Copying blob 5bf135c4a0de skipped: already exists
Copying blob 773711fd02f0 skipped: already exists
Copying blob 12113fa850f7 done
Copying config b479141386 done
Writing manifest to image destination
Storing signatures
--> b47914138 61
Successfully tagged localhost/myecho_image:latest
b47914138610245023d9781857000709f5c4ea22d464d16fcc6ce1b5daee2d5
```

List the container image:

```
$ buildah images
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
localhost/myecho-image      latest    636de016ba7a   9 seconds ago    225
MB
```

Inspect a container image

```
buildah inspect --type image image-id
```

or

```
buildah inspect --type image image-name
```

The `buildah inspect` command returns a very large JSON object that describes the many details of a container image.

Example:

The following example demonstrates executing the `buildah inspect` command against the image id `a134be2e5346`. The command produces a good deal of screen output. Thus the example shows a snippet of output:

```
buildah inspect --type image a134be2e5346

{
    "Type": "buildah 0.0.1",
    "FromImage": "localhost/instrumentreseller:latest",
    "FromImageID":
"a134be2e5346307e5999d059bbfabafa43763318b90be569454474e9d2289cf9",
    "FromImageDigest":
"sha256:ab86f8d2e3907728f9dcdeb62271e9f165b9dff6aa4507e352df97fc2e81e367",
    "Config": "{\"created\":\"2022-06-14T18:16:42.578103429Z\",
\"architecture\":\"amd64\",\"os\":\"...........\"}

}
```

Inspect a container image

Example:

The following example creates a working container based on the container image `myecho-image`. Then, a new file is created that echoes a message. There is an older version of the new file already in the container. The older file has an older message.

The command `buildah copy` is used to replace the older file with the contents of the new file. Finally the command `buildah commit` is used to create a new container image named `new-myecho-image`. The container image `new-myecho-image` has the content of the new file under the same file name as the legacy container image.

Create the working container:

```
$ buildah from myecho-image
myecho-image-working-container
```

Create the new file with new content:

```
$ echo "echo This is another container that works!"" > myecho
```

Copy the new file contents into the running working container:

```
buildah copy myecho-image-working-container myecho /tmp/myecho
5f270702af64a52e355b3bcff955bdde2648418bea6e9e4d5d68cbba91450598
```

Exercise the running working container to verify that the contents of the new file will be displayed:

```
$ buildah run myecho-image-working-container sh /tmp/myecho
This is another container that works!
```

Create a new container image based on the file system of the legacy container image that also has replacement content in the script file `/tmp/myecho`:

```
$ buildah commit myecho-image-working-container new-myecho-image
Getting image source signatures
Copying blob 5bf135c4a0de skipped: already exists
Copying blob 773711fd02f0 skipped: already exists
Copying blob 80062d3ed257 skipped: already exists
Copying blob c823fae997d4 done
Copying config f6dc970a52 done
Writing manifest to image destination
Storing signatures
f6dc970a528ce2c94eba3d957170ac612537e1bd9a9f6def15e246d5b965f4e5
```

List the local container images on the machine to verify that the new container image has been created:

```
$ buildah images
REPOSITORY                     TAG          IMAGE ID       CREATED
SIZE
localhost/new-myecho-image     latest       f6dc970a528c   10
seconds ago   225 MB
localhost/myecho-image         latest       636de016ba7a   3 hours
ago      225 MB
```

Working with a container image registry

The following sections show you how to:

- Log into a container image registry.
- Push a container image to a registry.
- Add an additional tag to a container image.

Logging into a remote container image registry

```
buildah login <registry_domain_name>
```

Example:

The following example executes buildah login. The command prompts for a username and password:

```
buildah login quay.io

Username:
Password:
Login Succeeded!
```

Pushing a container image to a container image registry

```
buildah push <local_image_name>:<optional_tag> <registry_domain_name>/
<repo_username>/<image_name>:<optional_tag>
```

Example:

The following command pushes the local container image to the repository of a user named cooluser on to the remote quay.io quay.io:

```
buildah push localhost/myecho-image quay.io/cooluser/myecho-image:v1.0
```

Create an additional image tag on an existing image

```
buildah tag <image_name>:<existing-image-tag> <image_name>:<new-image-tag>
```

Example:

The following example creates a new tag, verylatest and applies it to the existing container image docker.io/library/nginx that has the tag latest. Notice that the values of the IMAGE ID are identical:

```
$ buildah images
REPOSITORY                TAG      IMAGE ID       CREATED      SIZE
docker.io/library/nginx   latest   de2543b9436b   2 days ago   146 MB
```

```
$ buildah tag docker.io/library/nginx:latest docker.io/library/
nginx:verylatest

$ buildah images
REPOSITORY                 TAG          IMAGE ID       CREATED      SIZE
docker.io/library/nginx    latest       de2543b9436b   2 days ago   146 MB
docker.io/library/nginx    verylatest   de2543b9436b   2 days ago   146 MB
```

Working with containers

The following sections show you how to:

- List all working containers.
- Run a working container.
- Display details about a working container.
- Delete a working container.

List all working containers

```
buildah containers
```

The command `buildah containers` lists all working containers. A working container is a container that has been created using the `buildah from <container_image>` command.

Example:

The following example creates three working containers using the `buildah from` command. Then, the working directories are listed using the `buildah containers` command

Create the containers

```
$ buildah from httpd
httpd-working-container
$ buildah from busybox
busybox-working-container
$ buildah from nginx
nginx-working-container
```

List the containers

```
$ buildah containers

$ buildah containers
CONTAINER ID  BUILDER  IMAGE ID     IMAGE NAME
CONTAINER NAME
7071c5bab4ff     *     c58ef9bfbb57 docker.io/library/httpd:latest   httpd-
working-container
da51dced0afe     *     1a80408de790 docker.io/library/busybox:latest
busybox-working-container
bc1473702c2d     *     de2543b9436b docker.io/library/nginx:latest   nginx-
working-container
```

Running a container with buildah

```
buildah run [options] <working_container> <command>
```

Example:

The following example builds a working container from the image `httpd`. Since the image might exist in a number of remote registries, `buildah` displays a interactive list of registries to choose from:

```
$ buildah from httpd

  Please select an image:
    quay.io/httpd:latest
    registry.fedoraproject.org/httpd:latest
    registry.access.redhat.com/httpd:latest
    registry.centos.org/httpd:latest
  ▸ docker.io/library/httpd:latest

  httpd-working-container
```

The `buildah run` command is then executed against the working container created by `buildah from`. The example executes the `ls /var` command, listing the contents of the `/var` directory located within the working container:

```
$ buildah run httpd-working-container ls /var
backups  cache   lib  local  lock  log  mail  opt  run  spool  tmp
```

Display details about a container

```
buildah inspect [options] <container_id>
```

or

```
buildah inspect [options] <container_name>
```

Example:

The following example inspects the working container image named `registry.access.redhat.com/ubi8/ubi`. The option `--format '{{.IDMappingOptions}}'` is used to display only the information associated with the `IDMappingOptions` property of the JSON object:

```
$ buildah inspect --format '{{.IDMappingOptions}}' --type image
registry.access.redhat.com/ubi8/ubi
{true true [] []}
```

Delete a container

```
buildah delete <container_id>
```

or

```
buildah delete <container_name>
```

Example:

The following examples demonstrate deleting containers created by buildah:

```
$ buildah delete 35b88d7ef180
35b88d7ef1807a4d5e085472a23cea6425920ac94845fdcb33c036d89a804f3e
```

or

```
$ buildah delete httpd-working-container
f892d7f36f5f1d0b70fd40ebb00c0861cab44260f6b44add9574381673307ef5
```

Delete all containers on a machine, technique 1

```
buildah rm --all
```

Example:

The following example deletes all containers on the local machine:

```
$ buildah rm --all

4666ea9b554494c204dcc5c30ae0fcad8f8195a3d896845d100899b4e956313f
9b181a3172cefa5c92e33bd7ff2619bd6ac2bab9d87ab2a2bd9a226f70016282
```

Delete all containers on a machine, technique 2

```
buildah delete $(buildah list -a -q)
```

Example:

The following example deletes all containers created under `buildah run`. If no containers are running, the command will throw an error:

```
$ buildah delete $(buildah list  -a -q)

7071c5bab4ff60de473b37c5a152b2c566e0f6a8d401ba916ba761d77ad88d7a
da51dced0afec9db1178eb48631433462d26853baa2f472d67b587b2f04c7866
bc1473702c2d82f0a14741a49747a8077149bf2945177e107ad4057d7c9b67dc
```