Buildah Build OCI images

CONCEPTS

No daemon: buildah works on local storage directly and runs fine rootless. Working containers are build scratchpads, not services.

Image base or scratch
↓ `buildah from`
Working container run, copy, config
↓ `buildah commit`
New image local storage
↓ `buildah push`
Registry / archive docker://, oci:, dir:

Containerfile and Dockerfile are the same format; buildah build reads either. Scripting from/run/commit gives you control that a file cannot.

Unprivileged mount and filesystem edits only work inside buildah unshare, which gives you a user namespace with root mapped to you.

BUILD FROM A FILE

buildah build -t *<image>* **.**
Build an image from a Containerfile

buildah build -f *<file>* **-t** *<image>* *<dir>*
Use a specific Containerfile and context

buildah build --build-arg *<k>=<v>* **.**
Set a build argument

buildah build --layers .
Cache intermediate layers (off by default)

buildah build --target *<stage>* **.**
Stop at a named stage

WORKING CONTAINERS

buildah from *<image>*
Create a working container, print its name

buildah from scratch
Start from an empty root filesystem

WORKING CONTAINERS (cont.)

buildah containers
List working containers

buildah rm *<ctr>*
Remove a working container

buildah run *<ctr>* **--** *<cmd>*
Run a command inside the container

buildah run -t *<ctr>* **-- sh**
Interactive shell with a terminal

ADD CONTENT

buildah copy *<ctr>* *<src>* *<dest>*
Copy files into the container

buildah copy --chown *<user>:<group>* *<ctr>* *<src>* *<dest>*
Set owner while copying

buildah add *<ctr>* *<url|archive>* *<dest>*
Copy, fetching URLs and unpacking archives

buildah mount *<ctr>*
Print the container root filesystem path

buildah unshare
Shell in a rootless user namespace

CONFIGURE AND COMMIT

buildah config --entrypoint '*<json-array>***'** *<ctr>*
Set the entrypoint

buildah config --cmd '*<cmd>***'** *<ctr>*
Set the default command

buildah config --env *<k>=<v>* *<ctr>*
Set an environment variable

buildah config --port *<port>***/tcp** *<ctr>*
Declare an exposed port

buildah commit *<ctr>* *<image>*
Create an image from the container

buildah commit --rm *<ctr>* *<image>*
Commit, then delete the container

IMAGES AND REGISTRIES

buildah images
List local images

buildah inspect *<name>*
Show config of a container or image

buildah pull *<image>*
Fetch an image into local storage

buildah push *<image>* *<dest>*
Push to a registry or other transport

buildah push *<image>* **docker://***<registry>***/***<repo>***:***<tag>*
Push to a registry explicitly

buildah push *<image>* **oci-archive:***<file>***.tar:***<tag>*
Export as an OCI archive

buildah tag *<image>* *<new-name>*
Add another name to an image

buildah rmi *<image>*
Remove an image

buildah login *<registry>*
Log in to a registry

SYSTEM

Image names without a registry get localhost/ prepended on build; use a full name such as <registry>/<repo>:<tag> before pushing.

RECIPES

Scripted image from scratch

```
ctr=$(buildah from alpine)
buildah run $ctr -- apk add --no-cache curl
buildah copy $ctr ./app /usr/local/bin/app
buildah config --entrypoint '["/usr/local/bin/app"]' $ctr
buildah commit --rm $ctr localhost/myapp:1.0
```

Minimal rootfs with host package manager

```
buildah unshare bash -c '
  ctr=$(buildah from scratch)
  mnt=$(buildah mount $ctr)
  dnf -y install --installroot=$mnt --releasever=<n> bash
  buildah commit --rm $ctr localhost/mini
'
```