Buildah Build OCI images

START HERE

`buildah from <image>`
Create a working container

`buildah commit <ctr> <image>`
Save it as an image

`buildah run <ctr> -- <cmd>`
Run a command in it

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

`buildah copy <ctr> <src> <dest>`
Copy files in

`buildah images`
List local images

`buildah config --entrypoint ...`
Set image metadata

`buildah push <image> <dest>`
Publish an image

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

CONCEPTS

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

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)
Without --layers only the final layer is created, so repeat builds redo every step. BUILDAH_LAYERS=true sets it permanently.

`buildah build --no-cache .`
Ignore cached layers

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

`buildah build --platform <os/arch> .`
Build for another platform

`buildah build --format docker .`
Docker manifest instead of OCI
The default is oci; docker format is needed for HEALTHCHECK and SHELL to be kept.

`buildah build --pull=always .`
Always refresh base images

`buildah build --secret id=<id>,src=<file> .`
Pass a secret to RUN mounts

BUILD FROM A FILE (cont.)

`buildah build --platform <list> --manifest <list> .`
Multi-arch build into a manifest list

WORKING CONTAINERS

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

`buildah from scratch`
Start from an empty root filesystem

`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 copy --from <image> <ctr> <src> <dest>`
Copy from another image or container

`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

`buildah unshare --mount <var>=<ctr> -- <cmd>`
Mount a container and export its path

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

`buildah commit --format docker <ctr> <image>`
Docker format instead of OCI

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

MANIFEST LISTS

`buildah manifest create <list>`
Create a manifest list

`buildah manifest add <list> <image>`
Add an image to the list

`buildah manifest push --all <list> docker://<registry>/<repo>:<tag>`
Push the list and its images

SYSTEM

`buildah info`
Show storage and host details

`buildah version`
Show version info

`buildah --root <dir> --runroot <dir> <cmd>`
Use a separate storage location

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
'
```

Multi-arch image

```
buildah build --platform linux/amd64,linux/arm64 \
  --manifest myapp .
buildah manifest push --all myapp \
  docker://<registry>/<repo>:<tag>
```

Build, hand off to another tool

```
buildah build -t myapp .
buildah push myapp oci-archive:myapp.tar
skopeo copy oci-archive:myapp.tar docker://<registry>/<repo>:<tag>
```