



**Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!**

### ***Rocky Enterprise Linux 9.2 Manual Pages on command 'podman-auto-update.1'***

**\$ man podman-auto-update.1**

podman-auto-update(1)() podman-auto-update(1)()

#### NAME

podman-auto-update - Auto update containers according to their auto-update policy

#### SYNOPSIS

podman auto-update [options]

#### DESCRIPTION

podman auto-update looks up containers with a specified io.containers.autoupdate label (i.e., the auto-update policy).

If the label is present and set to registry, Podman reaches out to the corresponding registry to check if the image has been updated. The label image is an alternative to registry maintained for backwards compatibility. An image is considered updated if the digest in the local storage is different than the one of the remote image. If an image must be updated, Podman pulls it down and restarts the systemd unit executing the container. The registry policy requires a fully-qualified image reference (e.g., quay.io/podman/stable:latest) to be used to create the container. This enforcement is necessary to know which image to actually check and pull. If an image ID was used, Podman would not know which image to check/pull anymore.

Alternatively, if the autoupdate label is set to local, Podman will compare the image a container is using to the image with its raw name in local storage. If an image is updated locally, Podman simply restarts the systemd unit executing the container.

If io.containers.autoupdate.authfile label is present, Podman reaches out to the corresponding authfile when pulling images.

At container-creation time, Podman looks up the PODMAN\_SYSTEMD\_UNIT environment variable

and stores it verbatim in the container's label. This variable is now set by all systemd units generated by podman-generate-systemd and is set to %n (i.e., the name of systemd unit starting the container). This data is then being used in the auto-update sequence to instruct systemd (via DBUS) to restart the unit and hence to restart the container.

Note that podman auto-update relies on systemd. The systemd units are expected to be generated with podman-generate-systemd --new, or similar units that create new containers in order to run the updated images. Systemd units that start and stop a container cannot run a new image.

## Systemd Unit and Timer

Podman ships with a podman-auto-update.service systemd unit. This unit is triggered daily at midnight by the podman-auto-update.timer systemd timer. The timer can be altered for custom time-based updates if desired. The unit can further be invoked by other systemd units (e.g., via the dependency tree) or manually via `systemctl start podman-auto-update.service`.

## OPTIONS

`--authfile=path`

Path of the authentication file. Default is `${XDG_RUNTIME_DIR}/containers/auth.json`, which is set using `podman login`. If the authorization state is not found there, `$HOME/.docker/config.json` is checked, which is set using `docker login`.

Note: There is also the option to override the default path of the authentication file by setting the `REGISTRY_AUTH_FILE` environment variable. This can be done with `export REGISTRY_AUTH_FILE=path`.

`--dry-run=true|false`

Check for the availability of new images but do not perform any pull operation or restart any service or container. The `UPDATED` field indicates the availability of a new image with "pending".

`--format=format`

Change the default output format. This can be of a supported type like 'json' or a Go template. Valid placeholders for the Go template are listed below:

`--rollback=true|false`

If restarting a systemd unit after updating the image has failed, rollback to using the previous image and restart the unit another time. Default is true.

Please note that detecting if a systemd unit has failed is best done by the container

sending the READY message via SDNOTIFY. This way, restarting the unit will wait until having received the message or a timeout kicked in. Without that, restarting the systemd unit may succeed even if the container has failed shortly after.

For a container to send the READY message via SDNOTIFY it must be created with the `--sdnotify=container` option (see `podman-run(1)`). The application running inside the container can then execute `systemd-notify --ready` when ready or use the `sdnotify` bindings of the specific programming language (e.g., `sd_notify(3)`).

??

?Placeholder ? Description ?

??

?Unit ? Name of the systemd unit ?

??

?ContainerName ? Name of the container ?

??

?ContainerID ? ID of the container ?

??

?Container ? ID and name of the container ?

??

?Image ? Name of the image ?

??

?Policy ? Auto-update policy of the con? ?

? ? tainer ?

??

?Updated ? Update status: true,false,failed ?

??

## EXAMPLES

Autoupdate with registry policy

```
### Start a container
```

```
$ podman run --label "io.containers.autoupdate=registry"
```

```
  --label "io.containers.autoupdate.authfile=/some/authfile.json"
```

```
  -d --name=test registry.fedoraproject.org/fedora:latest sleep infinity
```

```
bc219740a210455fa27deacc96d50a9e20516492f1417507c13ce1533dbdcd9d
```

```
### Generate a systemd unit for this container
```

```

$ podman generate systemd --new --files
bc219740a210455fa27deacc96d50a9e20516492f1417507c13ce1533dbdcd9d

/home/user/container-bc219740a210455fa27deacc96d50a9e20516492f1417507c13ce1533dbdcd9d.service
### Load the new systemd unit and start it

$ mv ./container-bc219740a210455fa27deacc96d50a9e20516492f1417507c13ce1533dbdcd9d.service
~/config/systemd/user/container-test.service

$ systemctl --user daemon-reload

### If the previously created containers or pods are using shared resources, such as ports, make sure to remove
them before starting the generated systemd units.

$ podman stop bc219740a210455fa27deacc96d50a9e20516492f1417507c13ce1533dbdcd9d

$ podman rm bc219740a210455fa27deacc96d50a9e20516492f1417507c13ce1533dbdcd9d

$ systemctl --user start container-test.service

### Check if a newer image is available

$ podman auto-update --dry-run --format "{{.Image}} {{.Updated}}"
registry.fedoraproject.org/fedora:latest pending

### Autoupdate the services

$ podman auto-update

UNIT          CONTAINER          IMAGE          POLICY  UPDATED
container-test.service  08fd34e533fd (test) registry.fedoraproject.org/fedora:latest registry  false

Autoupdate with local policy

### Start a container

$ podman run --label "io.containers.autoupdate=local"
-d busybox:latest top

be0889fd06f252a2e5141b37072c6bada68563026cb2b2649f53394d87ccc338

### Generate a systemd unit for this container

$ podman generate systemd --new --files
be0889fd06f252a2e5141b37072c6bada68563026cb2b2649f53394d87ccc338

/home/user/container-be0889fd06f252a2e5141b37072c6bada68563026cb2b2649f53394d87ccc338.service
### Load the new systemd unit and start it

$ mv ./container-be0889fd06f252a2e5141b37072c6bada68563026cb2b2649f53394d87ccc338.service
~/config/systemd/user

$ systemctl --user daemon-reload

### If the previously created containers or pods are using shared resources, such as ports, make sure to remove

```

them before starting the generated systemd units.

```
$ podman stop be0889fd06f252a2e5141b37072c6bada68563026cb2b2649f53394d87ccc338
```

```
$ podman rm be0889fd06f252a2e5141b37072c6bada68563026cb2b2649f53394d87ccc338
```

```
$ systemctl --user start
```

```
container-be0889fd06f252a2e5141b37072c6bada68563026cb2b2649f53394d87ccc338.service
```

```
### Get the name of the container
```

```
$ podman ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
01f5c8113e84	docker.io/library/busybox:latest	top	2 seconds ago	Up 3 seconds ago		inspiring_galileo

```
### Modify the image
```

```
$ podman commit --change CMD=/bin/bash inspiring_galileo busybox:latest
```

```
### Auto-update the container
```

```
$ podman auto-update
```

```
[...]
```

SEE ALSO

[podman\(1\)](#), [podman-generate-systemd\(1\)](#), [podman-run\(1\)](#), [sd\\_notify\(3\)](#), [systemd.unit\(5\)](#)

[podman-auto-update\(1\)\(\)](#)