Preserve Docker volumes before updating a container

Replacing a Docker container can be routine when its persistent data is understood. It can lose important state when files exist only in the container's writable layer. These steps are for an administrator of a Linux Docker host. They prepare an update; they do not remove containers, prune volumes or restore over running data.

Build a storage map

```
docker ps -a --format 'table {{.Names}}□{{.Image}}□{{.Status}}'
docker inspect --format '{{json .Mounts}}' example-container
docker volume ls
```

Record each mount's type, source, destination and read/write state. A named volume, anonymous volume and bind-mounted host directory have different management paths. Save the Compose file or equivalent deployment definition privately. Do not publish complete inspect output because it may include environment secrets and internal details.

Check the application's documentation for every persistent path. A mounted data directory does not prove that uploads, encryption keys and configuration are also persistent. If important files live only in the writable layer, stop and design their migration before recreating anything. Avoid manually editing Docker's own storage directories.

Choose a consistent backup method

A database copied while it is writing may not be recoverable. Use the database's supported export or a documented application backup mode. Some services require a controlled stop to capture all related files consistently. Record that requirement and communicate the expected interruption before starting.

Keep the archive outside the source volume and outside a public web root. Restrict access, encrypt sensitive exports before off-server storage and check free space. Include the application version, exact image reference, configuration and ownership information needed to interpret the data. A tag such as latest is not an adequate record of what was running.

Prove the recovery path separately

Verify archive checksums after transfer. Restore into a separate test volume or directory and an isolated application instance, following the vendor's instructions. Prevent the test instance from sending mail, running scheduled jobs or connecting to live write services. Never attach two independent database instances to the same writable database directory.

A volume mount can obscure files already present at its target inside the image. Empty-volume initialization behaviour also matters. Check that the updated image sees the intended persistent data rather than silently creating a fresh installation in a new volume.

Plan the image change

Review supported upgrade order and database migrations. Keep the previous deployment definition and image reference, but understand that rolling back an image alone may not reverse a data-format change. Identify the backup that belongs with the old version and define a point beyond which restoration is required.

Recreate only the intended service through the established deployment workflow after these prerequisites pass. Avoid volume-removal flags and broad cleanup commands. Once running, check application health, data visibility, permissions and scheduled tasks. A container reporting running is not proof that the correct dataset loaded.

If validation fails

Stop further changes and preserve logs. Follow the documented application rollback or restore procedure in a maintenance window. Do not repeatedly switch images against an already migrated database. Keep the old backup until the updated application has passed a meaningful period of verification and your retention policy permits its removal.

Related instructions

Verify a downloaded backup. Export a MariaDB database.

Reference: Official documentation. Check the documentation installed on your own operating-system release when an option differs.

Revision #1
Created 2026-11-09 10:42:42 UTC by CloudExa Hosting Admin
Updated 2026-11-09 10:42:42 UTC by CloudExa Hosting Admin