Instalação e configurações Duplicati

Procedimentos de instalação e configuração.

- Instalação Duplicati (Linux Server)

Instalação Duplicati (Linux Server)

Link: https://github.com/linuxserver/docker-duplicati

18/05/2025

The LinuxServer.io team brings you another container release featuring:

- regular and timely application updates
- easy user mappings (PGID, PUID)
- custom base image with s6 overlay
- weekly base OS updates with common layers across the entire LinuxServer.io ecosystem to minimise space usage, down time and bandwidth
- regular security updates

Find us at:

- Blog - all the things you can do with our containers including How-To guides, opinions and much more!
- Discord - realtime support / chat with the community and the team.
- Discourse - post on our community forum.
- Fleet - an online web interface which displays all of our maintained images.
- GitHub - view the source for all of our repositories.
- Open Collective - please consider helping us by either donating or contributing to our budget

linuxserver/duplicati

Scarf.io pulls GitHub Stars GitHub Release GitHub Package Repository GitLab Container Registry Quay.io Docker Pulls Docker Stars Jenkins Build LSIO CI

Duplicati is a backup client that securely stores encrypted, incremental, compressed backups on local storage, cloud storage services and remote file servers. It works with standard protocols like FTP, SSH, WebDAV as well as popular services like Microsoft OneDrive, Amazon S3, Google Drive, box.com, Mega, B2, and many others.

duplicati

Supported Architectures

We utilise the docker manifest for multi-platform awareness. More information is available from docker here and our announcement here.

Simply pulling `lscr.io/linuxserver/duplicati:latest` should retrieve the correct image for your arch, but you can also pull specific arch images via tags.

The architectures supported by this image are:

Architecture	Available	Tag
x86-64	✅	amd64-<version tag>
arm64	✅	arm64v8-<version tag>
armhf	✅	

Version Tags

This image provides various versions that are available via tags. Please read the descriptions carefully and exercise caution when using unstable or development tags.

Tag	Available	Description
latest	✅	Stable releases of Duplicati
development	✅	Beta releases of Duplicati

Application Setup

The webui is at `<your ip>:8200`.

For local backups select `/backups` as the destination. For more information see Duplicati.

Usage

To help you get started creating a container from this image you can either use docker-compose or the docker cli.

Note

Unless a parameter is flaged as 'optional', it is *mandatory* and a value must be provided.

docker-compose (recommended, click here for more info)

```
---
services:
  duplicati:
    image: lscr.io/linuxserver/duplicati:latest
    container_name: duplicati
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Etc/UTC
      - SETTINGS_ENCRYPTION_KEY=
      - CLI_ARGS= #optional
      - DUPLICATI__WEBSERVICE_PASSWORD= #optional
    volumes:
      - /path/to/duplicati/config:/config
      - /path/to/backups:/backups
      - /path/to/source:/source
    ports:
      - 8200:8200
    restart: unless-stopped
```

docker cli (click here for more info)

```
docker run -d \
  --name=duplicati \
  -e PUID=1000 \
  -e PGID=1000 \
  -e TZ=Etc/UTC \
  -e SETTINGS_ENCRYPTION_KEY= \
  -e CLI_ARGS= `#optional` \
  -e DUPLICATI__WEBSERVICE_PASSWORD= `#optional` \
  -p 8200:8200 \
  -v /path/to/duplicati/config:/config \
  -v /path/to/backups:/backups \
  -v /path/to/source:/source \
  --restart unless-stopped \
```

```
lscr.io/linuxserver/duplicati:latest
```

Parameters

Containers are configured using parameters passed at runtime (such as those above). These parameters are separated by a colon and indicate `<external>:<internal>` respectively. For example, `-p 8080:80` would expose port `80` from inside the container to be accessible from the host's IP on port `8080` outside the container.

Parameter	Function
`-p 8200:8200`	http gui
`-e PUID=1000`	for UserID - see below for explanation
`-e PGID=1000`	for GroupID - see below for explanation
`-e TZ=Etc/UTC`	specify a timezone to use, see this list.
`-e SETTINGS_ENCRYPTION_KEY=`	Encryption key for settings database. Minimum 8 characters, alphanumeric.
`-e CLI_ARGS=`	Optionally specify any CLI variables you want to launch the app with
`-e DUPLICATI__WEBSERVICE_PASSWORD=`	Password for the webui. If left unset will default to `changeme` and can be changed from the webui settings.
`-v /config`	Contains all relevant configuration files.
`-v /backups`	Path to store local backups.
`-v /source`	Path to source for files to backup.

Environment variables from files (Docker secrets)

You can set any environment variable from a file by using a special prepend `FILE__`.

As an example:

```
-e FILE__MYVAR=/run/secrets/mysecretvariable
```

Will set the environment variable `MYVAR` based on the contents of the `/run/secrets/mysecretvariable` file.

Umask for running applications

For all of our images we provide the ability to override the default umask settings for services started within the containers using the optional `-e UMASK=022` setting. Keep in mind umask is not chmod it subtracts from permissions based on it's value it does not add. Please read up here before asking for support.

User / Group Identifiers

When using volumes (`-v` flags), permissions issues can arise between the host OS and the container, we avoid this issue by allowing you to specify the user `PUID` and group `PGID`.

Ensure any volume directories on the host are owned by the same user you specify and any permissions issues will vanish like magic.

In this instance `PUID=1000` and `PGID=1000`, to find yours use `id your_user` as below:

```
id your_user
```

Example output:

```
uid=1000(your_user) gid=1000(your_user) groups=1000(your_user)
```

Docker Mods

Docker Mods Docker Universal Mods

We publish various Docker Mods to enable additional functionality within the containers. The list of Mods available for this image (if any) as well as universal mods that can be applied to any one of our images can be accessed via the dynamic badges above.

Support Info

- Shell access whilst the container is running:
  ```
  docker exec -it duplicati /bin/bash
  ```

- To monitor the logs of the container in realtime:

```
docker logs -f duplicati
```

- Container version number:

```
docker inspect -f '{{ index .Config.Labels "build_version" }}' duplicati
```

- Image version number:

```
docker inspect -f '{{ index .Config.Labels "build_version" }}' lscr.io/linuxserver/duplicat
```

Updating Info

Most of our images are static, versioned, and require an image update and container recreation to update the app inside. With some exceptions (noted in the relevant readme.md), we do not recommend or support updating apps inside the container. Please consult the Application Setup section above to see if it is recommended for the image.

Below are the instructions for updating containers:

Via Docker Compose

- Update images:
  - All images:

```
docker-compose pull
```

  - Single image:

```
docker-compose pull duplicati
```

- Update containers:
  - All containers:

```
docker-compose up -d
```

  - Single container:

```
docker-compose up -d duplicati
```

- You can also remove the old dangling images:

```
docker image prune
```

Via Docker Run

- Update the image:

```
docker pull lscr.io/linuxserver/duplicati:latest
```

- Stop the running container:

```
docker stop duplicati
```

- Delete the container:

```
docker rm duplicati
```

- Recreate a new container with the same docker run parameters as instructed above (if mapped correctly to a host folder, your `/config` folder and settings will be preserved)
- You can also remove the old dangling images:

```
docker image prune
```

Image Update Notifications - Diun (Docker Image Update Notifier)

Tip

We recommend Diun for update notifications. Other tools that automatically update containers unattended are not recommended or supported.

Building locally

If you want to make local modifications to these images for development purposes or just to customize the logic:

```
git clone https://github.com/linuxserver/docker-duplicati.git
cd docker-duplicati
docker build \
  --no-cache \
  --pull \
  -t lscr.io/linuxserver/duplicati:latest .
```

The ARM variants can be built on x86_64 hardware and vice versa using `lscr.io/linuxserver/qemu-static`

```
docker run --rm --privileged lscr.io/linuxserver/qemu-static --reset
```

Once registered you can define the dockerfile to use with `-f Dockerfile.aarch64`.

Versions

- **31.01.25:** - Make `latest` stable releases, move beta releases to `development`.
- **28.01.25:** - Add xz-utils.
- **03.12.24:** - Add mscorefonts for captcha support.
- **29.11.24:** - Rebase to Noble, add support for settings DB encryption.
- **15.02.23:** - Rebase to Jammy.
- **03.08.22:** - Deprecate armhf.
- **25.04.22:** - Rebase to mono:focal.
- **01.08.19:** - Rebase to Linuxserver LTS mono version.
- **16.07.19:** - Allow for additional command line arguments in an environment variable.
- **28.06.19:** - Rebase to bionic.
- **23.03.19:** - Switching to new Base images, shift to arm32v7 tag.
- **28.02.19:** - Allow access from all hostnames, clarify info on image tags.
- **13.01.19:** - Use jq instead of awk in dockerfiles.
- **11.01.19:** - Multi-arch image.
- **09.12.17:** - Fix continuation lines.
- **31.08.17:** - Build only beta or release versions (thanks deasmi).
- **24.04.17:** - Initial release.