Duplicati

Sistema de backups

- Instalação e configurações Duplicati
  - Instalação Duplicati (Linux Server)
- Customizações Duplicati
  - Duplicati Monitor
  - Duplicati Dashboard
  - Duplicati Dashboard2

Instalação e configurações Duplicati

Procedimentos de instalação e configuração.

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.

Customizações Duplicati

Duplicati Monitor

Link: https://github.com/RafaMunoz/duplicati-monitor git clone https://github.com/RafaMunoz/duplicati-monitor.git

Duplicati Monitor

Duplicati Monitor is a docker-packaged monitoring solution for Duplicati.

It allows you to control the status of your backups and statistics for each report and notifies you by any means supported by **Apprice**. It is meant to be self-hosted and powered by docker.

Quick Start

Notifications

To send notifications and satisfy the end user, it has been implemented with the Apprise library, which allows you to send a notification to almost all the most popular notification services available today, such as: Telegram, Discord, Slack, Amazon SNS, Gotify, etc.

You can check all the services available in the official documentation.

Templates

You can create your own templates to receive the information you exactly need.

By default we use the following two:

```
# Succes backup
□□□□<Extra.backup-name>
```

```
# Error backup
□□□□<Extra.backup-name>
```

You can create your own templates in a simple way. You only have to use the symbols `<` and `>` delimit the fields and separating keys with dots `.` to send along with the message that you want.

In the docs/examples_report folder of this repository you have two examples of the JSON reports that Duplicati sends and in which you can see the fields that compose it.

For example, with the following template, the result shown in the telegram image would be obtained.

```
□□□□Backup: <Extra.backup-name>\n - Examined Files: <Data.ExaminedFiles>\n - Duration:
<Data.Duration>\n - Status: *<Data.TestResults.ParsedResult>*
```

telegram-example-notication

Environment Variables

ENVIRONMENT VARIABLE	TYPE	DEFAULT	DESCRIPTION
`URI_NOTIFICATION`	required		URI in Apprise format where the notifications will be sent.
`TEMPLATE_SUCCESS`	opcional	□□ □□ <Extra.backup-name>	Message template to be sent when the backup is successful.
`TEMPLATE_ERROR`	opcional	□□ □□ <Extra.backup-name>	Message template that will be sent when the backup is executed in an erroneous way.
`PORT`	opcional	8000	Listening port on which the service is set up to receive the reports.

Docker Run

To start the container you can do it with the following command.

```
docker run -d --name=duplicati-monitor -p 8000:8000 -e
URI_NOTIFICATION=tgram://<TOKEN_TELEGRAM_BOT>/<CHANEL_ID>/?format=markdown rafa93m/duplicati-
monitor
```

Or you can also use the following docker compose.

```
version: "3"
services:
    duplicati-monitor:
        container_name: duplicati-monitor
        image: rafa93m/duplicati-monitor
        ports:
            - 8000:8000
        environment:
            URI_NOTIFICATION: "tgram://<TOKEN_TELEGRAM_BOT>/<CHANEL_ID>/?format=markdown"
            TEMPLATE_SUCCESS: "□□□□Backup: <Extra.backup-name>"
            TEMPLATE_ERROR: "□□□□Backup <Extra.backup-name> failed □□□□"
        restart: unless-stopped
```

Setup Duplicati

Add theses two options for each backup you want to monitor:

- **send-http-result-output-format**: json
- **send-http-url**: http://**IP_ADDRESS**:**PORT**/report

advanced-options

Duplicati Dashboard

Link: https://github.com/fabien-github/duplicati_dashboard?tab=readme-ov-file#demo

git clone https://github.com/fabien-github/duplicati_dashboard.git

Duplicati Dashboard is a monitoring solution for Duplicati.

It allows you to monitor your backups status, collects stats for each reports and alerts you by email when a backup fails. It is intended to be self-hosted and works with docker-compose.

Everything is already pre-configured and ready to be deployed.

- Duplicati Dashboard
- Demo
- Quick Start
  - Running with docker-compose
  - Setup Duplicati
  - Connect to your dashboard
- Configuration
  - Env file
- Notes
  - Grafana configuration locked
  - Backup over more than 30 days rotation
  - Alerting graph
  - Deleting removed backup data
  - Docker-compose
- Other informations
- License

Demo



> The right side of the video is not integrated in the dashboard. You can't control your backup with it.

Quick Start

Running with docker-compose

```
git clone https://github.com/fabien-github/duplicati_dashboard.git
cd duplicati_dashboard
docker-compose up -d
```

Setup Duplicati

Add theses two options for each backup you want to monitor:

- **send-http-result-output-format:** json
- **send-http-url:** http://localhost:8080



> This assumes that your Duplicati instance is on the same host as your Duplicati Dashboard.

Connect to your dashboard

http://localhost:3000

Login: `admin` Password: `password`

> For email alerting, you need to configure a SMTP relay. See Configuration > Env file

Configuration

Env file

The file `config.env` is used to configure some options. It will be shared between the 3 containers.

Only use this default configuration for testing purposes.

Influxdb variable will create and setup the database only on the first startup.

Variables	Default	Description
DOCKER_INFLUXDB_INIT_MODE	setup	Automatically bootstrap the system
DOCKER_INFLUXDB_INIT_USERNAME	telegraf_user	Influxdb superadmin user
DOCKER_INFLUXDB_INIT_PASSWORD	telegraf_password	Influxdb superadmin password
DOCKER_INFLUXDB_INIT_ORG	telegraf_org	Influxdb Organization (used by influxdb / telegraf / grafana)
DOCKER_INFLUXDB_INIT_BUCKET	telegraf	Influxdb bucket to store reports (used by influxdb / telegraf / grafana)
DOCKER_INFLUXDB_INIT_ADMIN_TOKEN	telegraf_token	Influxdb superadmin token (used by influxdb / telegraf / grafana)
DOCKER_INFLUXDB_INIT_RETENTION		Influxdb data retention, default will retain forever
INFLUXD_REPORTING_DISABLED	false	Disable InfluxData telemetry
TELEGRAF_LISTENER_PORT	8080	Port used by http_listener_v2 input, endpoint for the reports sent by Duplicati
TELEGRAF_LISTENER_PATH	/	Path to listen to
GF_SECURITY_ADMIN_USER	admin	Grafana superadmin user

Variables	Default	Description
GF_SECURITY_ADMIN_PASSWORD	password	Grafana superadmin password
GF_SERVER_ROOT_URL	http://localhost:3000	Grafana URL, used in some templates like email notifications
GF_DASHBOARDS_DEFAULT_HOME_DASHBOARD_PATH	/etc/grafana/provisioning/dashboards/duplicati_dashboard.json	Force Duplicati dashboard by default on home page
GF_SMTP_ENABLED	false	Set to true for email notifications
GF_SMTP_HOST	localhost:25	SMTP relay server. [host]:[port]
GF_SMTP_FROM_NAME	Grafana	Name of the email sender
GF_SMTP_USER		In case of SMTP auth
GF_SMTP_PASSWORD		In case of SMTP auth
GF_SMTP_FROM_ADDRESS	admin@grafana.localhost	Address used when sending out emails
GF_SMTP_EHLO_IDENTITY	${HOSTNAME}	Name to be used as client identity for EHLO in SMTP dialog (Default will be the container ID)
GF_SMTP_STARTTLS_POLICY		“OpportunisticStartTLS”, “MandatoryStartTLS”, “NoStartTLS”
NOTIFIER_EMAIL_RECIPIENT	example@example.com	Recipients for email notification (separated by a semicolon)
NOTIFIER_EMAIL_REMINDER_ENABLE	true	Re-send an email if alerts are still active
NOTIFIER_EMAIL_REMINDER_FREQUENCY	2h	Delay between email reminders

Notes

Grafana configuration locked

The dashboard is locked by the Grafana provisioning system. You can't edit the datasource, the dashboard or the alert notifier from the UI. You will need to copy the dashboard or disable the provisioning configuration.

Dashboard path: `./grafana/provisioning/dashboards/duplicati_dashboard.json`

The idea is to keep the stack easy to deploy for everyone without investing time to learn Grafana configuration.

Feel free to fork the project or directly edit files on your own.

More information here.

Backup over more than 30 days rotation

- Grafana will discover your backups name from the reports but only over the last 90 days. So if your backups are scheduled for more than 90 days, you will need to edit the request of the variable `Backup` in the dashboard configuration:

```
from(bucket: v.defaultBucket)
  |> range(start: -90d)
  |> filter(fn: (r) => true)
  |> toString()
  |> group(columns: ["backup-name"])
  |> distinct(column: "backup-name")
  |> keep(columns: ["_value"])
```

- Last reported status / Last reported variations / Alerting graph are based over the past 30 days. You will need to adapt each panel requests if your backups are scheduled over more than 30 days.

```
import "influxdata/influxdb/schema"

from(bucket: v.defaultBucket)
  |> range(start: -30d)
  ...
```

- In v2, alerts have their own provisioning file and query range can be edited here.

Alerting graph

The section "Alerting graph" is only used to trigger an alert when a backup fails. This is due to the lack of grafana alert support on other panel type. #6983

Backups status will be checked every minutes. An alert will be triggered after a pending status of 10min. Same delays are used on the recovery.

Warning reports don't trigger an alert.

Deleting removed backup data

After getting inside the influxdb container:

```
influx delete \
  --org telegraf_org \
  --bucket telegraf \
  --token "telegraf_token" \
  --start 1970-01-01T00:00:00Z \
  --stop $(date +"%Y-%m-%dT%H:%M:%SZ") \
  --predicate '_measurement="duplicati" AND "backup-name"="backup_to_delete"'
```

Docker-compose

- **Telegraf**: Receive JSON reports from Duplicati.
- **Influxdb**: Store reports converted by Telegraf.
- **Grafana**: Requests Influxdb to generate dashboard and alerts.

Telegraf endpoint provides a limited HTTP autentification.
The configuration file is located here : `./telegraf/telegraf.conf`

Feel free to add a proxy like traefik or nginx to protect the stack on an unsecure network. (TLS, IP Restrictions, ...)

Other informations

- Not sure of the scalabilty, requests to the database are not efficients. Timeseries databases are not really adapted for this kind of data. This is mainly due to the nested and uneven json format from the reports and the variation time between reports.
- Features are limited directly by the stack itself. For example, it's nearly impossible to add a managment system for the backups.
- This project has no link with the development of Duplicati and his team.

License

Distributed under the GNU General Public License v3.0 License. See LICENSE for more information.

Duplicati Dashboard2

Link: https://github.com/wchorski/duplicati-dashboard git clone https://github.com/wchorski/duplicati-dashboard.git

A NodeJS based server that collects JSON data from Duplicati backup logs
□□ Tech

Frontend: NextJS
API: NextJS
Database: InfluxDB

[!warning] Backups sharing the same name will cause issues. The backup's name must be unique across all Duplicati instances, it will be used as an ID for logging. MUST BE URL FRIENDLY example: "Laptop--Home_Folder_Backup", "Desktop--Home_Folder_Backup" is a good naming convention.

Usecase

Initally this was just some middle ware that serves as an endpoint for JSON friendly monitoring apps, but I also built a simple UI so it could be used as a standalone app.
Duplicati Setup

you can either add these settings for the globally or per backup in the Advanced options

send-http-result-output-format = json
send-http-url = "http://APPSDOMAIN/api/backups"

API

Here is a breakdown of what endpoints and search parameters that can be passed through.
URL Breakdown

http://APPDOMAIN/backups/BACKUP_ID?start=-5h&first=true

BACKUP_ID => the id (or name) of the backup saved
stuff after the "?" search query sets range of time of pulled data
start => how far back to you want to start pulling data i.e.
-40d 40 days ago [the default]
-5h 5 hours ago
-60m 60 miniutes ago

1999-12-31T00:00:00 starting on December 31st, 1999
1694117521 starting on this UNIX time (seconds)
stop => the end of the range
now() => my current time [the default]
all the other examples above as long as the date is after the start
both query parameters can be omitted
last => set to true if you'd like the last recorded point in the table
can also use the http://localhost:3000/api/backups/last/BACKUP_ID endpoint for cleaner GET (this endpoint also takes start and stop query parameters)
first => same as last but returns the first point of recoreded data within the range

[!note] make sure relative dates have a negative i.e. -5h as your are looking back in time. Positive time values will cause errors

Examples
query url
all backup stats in database http://APPDOMAIN/backups
single backup stats http://APPDOMAIN/backups/BACKUP_ID
last recorded backup stat http://APPDOMAIN/backups/last/BACKUP_ID
same as above http://APPDOMAIN/backups/BACKUP_ID?last=true
last recorded backup stat in the last 5 hours http://APPDOMAIN/backups/last/BACKUP_ID?start=-5h
⚙ Development

```
git clone https://github.com/wchorski/duplicati-dashboard.git && cd duplicati-dashboard
cp .env.template .env.local
set up InfluxDB instance
get InfluxDB API Token for .env.local
yarn install
yarn dev
```

□□ Production

```
git clone https://github.com/wchorski/duplicati-dashboard.git && cd duplicati-dashboard
cp .env.template .env
the INFLUX_TOKEN in .env should be a long ~88 character string
docker compose up -d
```

Home Assistant

```
rest:
- authentication: basic
  username: "admin"
  password: "password"
  scan_interval: 86400
  resource: http://APPDOMAIN.lan/api/backups/last/DUPLICATI_ID
```

```
  sensor:
    - name: "duplicati-DUPLICATI_ID-status"
      value_template: "{{ value_json.status }}"
    - name: "duplicati-DUPLICATI_ID-time"
      value_template: >
        {% set thistime = value_json.time %}
        {{ as_timestamp(thistime) | timestamp_custom("%Y %M, %d %H:%M") }}
```

#Todo

- create dynamic nav based on unique duplicati_ids from database
- add FAQ as a page inside the app
- mobile friendly (almost there)
- graph trends in app
- Home Assistant Template sensor
- Don't be lazy and figure out Types in TableClient.tsx component
- get real data for screenshots
- why is bg tile image weird when scrolling on mobile?
- human readable bytes formatter (gb tb)
- human readable duration formatter