

Deploy docker-compose apps (Wordpress, MySQL, Redis, Keycloak ...)

You are about to learn how to deploy docker-compose applications to a CI/CD target. This is useful when you don't need a Git workflow and your images are already available on a public or private docker registry.

Use cases:

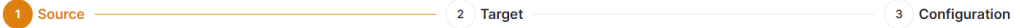
- Deploy one or multiple instances of stateful apps like Wordpress / Directus / MySQL / Redis / RabbitMQ / ...
- Deploy an internal service available only to your private network
- Deploy an app built somewhere else and published to a private docker registry

First, open the [Elestio dashboard and click on CI/CD](#)

1) Select your app to deploy

When you select Docker to compose the deployment method, we provide a few samples of applications that you can deploy

Create CI/CD pipeline



1. Deployment Method

Select the deployment method of your Service between Github, Gitlab and Docker. When using the Github/Gitlab deployment method, each time a change is pushed to your repository, a new deployment of your service will occur.

Github
 GitHub, Inc. is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and...

Gitlab
 GitLab Inc. is the open-core company that provides GitLab, the DevOps software that combines the ability to develop, secure, and...

Docker compose
 Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your...

Select a docker-compose template

Search templates by name

Custom docker-compose
 Add your custom docker-compose content

Redis + RedisInsight
 Redis is the most popular in-memory database, cache and message broker.

Postgres + PgAdmin
 Postgres is a relational database system known for reliability and performance.

TimescaleDB + PgAdmin
 TimescaleDB is the leading relational database with support for time-series data.

MySQL + PhpMyAdmin
 MySQL is a relational database that runs on almost all platforms.

MariaDB + PhpMyAdmin
 The open source relational database.

MongoDB + MongoExpress
 MongoDB is a document-oriented NoSQL database used for high-volume data storage.

ClickHouse + Tabix
 ClickHouse is a column-oriented DBMS for online analytical processing.

OpenSearch + Dashboards
 Open source distributed and RESTful search engine.

MSSQL + SQLPad
 Microsoft SQL Server is a relational database management system developed by Microsoft

RabbitMQ
 RabbitMQ is the most widely deployed message broker

Squid
 Lightweight, Fast & powerful proxy server

WG-Easy
 The easiest way to run WireGuard VPN + Web-based Admin UI

Ubuntu Desktop
 Full Desktop experience in the browser with a selection of tools + Windows compatibility with Wine.

MetaTrader 5
 Trade on Forex & stock markets in a virtual desktop from your browser

Airbyte
 Airbyte is an ETL platform that helps you replicate your data in your warehouses, lakes and databases.

Documize
 Modern Confluence alternative designed for internal & external docs, built with Go + EmberJS

MinIO
 MinIO is a leader in hybrid cloud and multi-cloud object storage.

Guacamole
 Apache Guacamole is a clientless remote desktop gateway. It supports standard protocols like VNC, RDP, and SSH.

MeshCentral
 Connect to your home or office devices from anywhere with MeshCentral. Management and Remote desktop.

Keycloak
 Identity and access management solution

Metabase
 Simply and quickly gathers business intelligence and analytics for your company.

Bookstack
 Easy-to-use platform for organising and storing information.

Directus
 Open Data Platform for managing the content of any SQL database.

Uptime-Kuma
 Uptime Kuma is a self-hosted monitoring tool like Uptime Robot

SFTPGo
 Fully featured SFTP server with optional HTTP/S, FTP/S and WebDAV support

Ghost
 More than a blog. Turn your audience into a business.

WORDPRESS
 Create a beautiful website, blog or app.

WooCommerce
 WooCommerce is a customizable, open-source eCommerce platform built on WordPress.

From there click on Github or Gitlab, and you will be asked to provide authorization to list your projects in Elestio.

Then you will be able to browse Organizations & Repositories detected on your account. You can also use the search to find directly your project to deploy. Once you found it, click on Import, then click on next.

2) Select your target (where to deploy)

Here you have to indicate where the app should be deployed, it can be a "New infrastructure", in that case, you can select your preferred provider/region/instance size. Or an existing infrastructure, then you just have to pick it from the list.

The screenshot displays the 'Create CI/CD pipeline' workflow in the Elestio dashboard. The interface is divided into three main steps: 1. Source, 2. Target, and 3. Configuration. The 'Target' step is currently active and highlighted with a blue circle and a '2' above it. A progress bar at the top shows the sequence of steps. On the left, a sidebar contains the Elestio logo, navigation icons, and a menu for the 'shared-dev' project, including options for Services, Volumes, Load Balancer, CI/CD (highlighted), Members, Billing, Settings, and Audit Trail. Below the menu, account information is shown: \$426.12 CREDITS and \$0.0420/hour SPENDING, with an 'Add credits' link. The main content area for the 'Target' step includes a question 'Where do you want this service to be deployed?' with two options: 'New Infrastructure' (selected) and 'Existing Infrastructure'. Below this, the 'Deployment mode' is set to 'Single Node' (selected) over 'Cluster'. The next step is '2. Select Service Cloud Provider', showing a grid of providers: HETZNER, DigitalOcean, Amazon Lightsail (selected), linode, and VULTR. The final step is '3. Select Service Cloud Region', with 'North America' selected. Two regions are listed: 'ca-central-1' (Canada - Montreal) and 'us-east-1' (USA - N. Virginia).

elestio

PROJECT: shared-dev

- Services
- Volumes
- Load Balancer
- CI/CD**
- Members
- Billing
- Settings
- Audit Trail

\$426.12 CREDITS
\$0.0420/hour SPENDING

us-west-2
USA - Oregon


4. Select Service Plan

- SMALL-1C-2G**
1 CPU 2 GB RAM 60 GB Storage 3 TB Bandwidth included
- MEDIUM-2C-4G**
2 CPU 4 GB RAM 80 GB Storage 4 TB Bandwidth included
- LARGE-2C-8G**
2 CPU 8 GB RAM 160 GB Storage 5 TB Bandwidth included
- XLARGE-4C-16G**
4 CPU 16 GB RAM 320 GB Storage 6 TB Bandwidth included
- 2XLARGE-8C-32G**
8 CPU 32 GB RAM 640 GB Storage 7 TB Bandwidth included

3) Configure your app

This is the last step of the process where you can adjust the app settings, docker-compose, env vars, and reverse proxy configuration.

a) Docker-compose stack



PROJECT: shared-dev

- Services
- Volumes
- Load Balancer
- CI/CD**
- Members
- Billing
- Settings
- Audit Trail

\$422.19 CREDITS

\$0.0280/hour SPENDING

[Add credits](#)

Create CI/CD pipeline



1. Docker compose

```

docker-compose.yml
1 version: '3'
2 services:
3
4   postgres:
5     image: timescale/timescaledb:${SOFTWARE_VERSION_TAG}
6     restart: always
7     environment:
8       POSTGRES_DB: postgres
9       POSTGRES_USER: postgres
10      POSTGRES_PASSWORD: ${SOFTWARE_PASSWORD}
11      PGDATA: /var/lib/postgresql/data
12      TS_TUNE_MAX_CONNS: 100
13     volumes:
14       - ./data:/var/lib/postgresql/data
15     ports:
16       - '172.17.0.1:5432:5432'
17
18   pgadmin4:
19     image: dpage/pgadmin4:latest
20     restart: always
21     environment:
22       PGADMIN_DEFAULT_EMAIL: ${ADMIN_EMAIL}
23       PGADMIN_DEFAULT_PASSWORD: ${ADMIN_PASSWORD}
24       PGADMIN_LISTEN_PORT: 8080
25     ports:
26       - "172.17.0.1:8080:8080"
27     volumes:
28       - ./servers.json:/pgadmin4/servers.json
29
  
```

Use Private Repository

Environment variables +

Reverse proxy configuration +

b) Environment variables

In most cases, you will have to indicate configuration for your app through env vars. This is useful to pass various configurations to your app like database connection string, S3 bucket details, email address to use, and other global configurations.

Environment variables



Use Environment variables to store configuration values. API keys and secrets. You can access them in your service like regular environment variables.

.env

```
1 ENV=production
2 MY_PARAM_1=true
3
```

c) Reverse proxy

To make your app accessible on the internet, indicate in the target port the same thing you have configured on the host port in the docker-compose ports binding.

Reverse proxy configuration



Configure the ports which should be exposed. When the ports are not exposed publicly, they are only accessible to other Services of the same App via the internal network.

Listen

Target

Protocol

Port

HTTPS

443



Protocol

IP

Port

Path

HTTP

172.17.0.1

8001

/

Require Basic Auth

Login

root

Password

azfhwglF-dq83-9dfSa8Hi

Protocol

Port

TCP

26379



Protocol

IP

Port

Path

TCP

172.17.0.1


6379

/

Require Basic Auth

+ Add Another

Finally, click on "Create CI/CD pipeline" to complete your deployment.



Provider
Amazon Lightsail

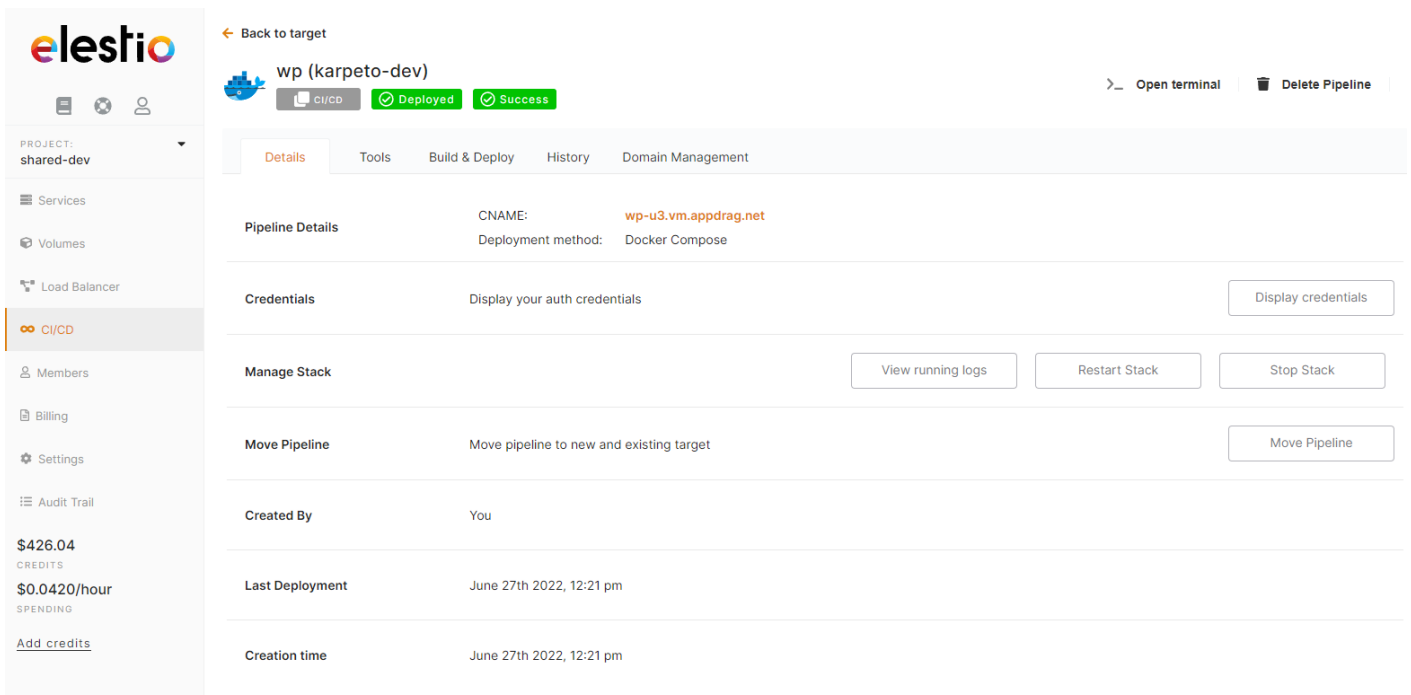
Region
North America, Canada
Montreal

Estimated Monthly Price*
\$0

*Estimated monthly price is based on 730 hours of usage.

[Create CI/CD pipeline](#)

After a few minutes, your app should be accessible on the CI/CD pipeline url, you can find it in the dashboard overview of your pipeline. Your generated credentials are visible in the "Build & Deploy" tab in the env var section or reverse proxy section if you have activated basic authentication.



The screenshot shows the Elestio dashboard for a project named 'shared-dev'. The main content area displays details for a pipeline named 'wp (karpeto-dev)'. The pipeline status is 'Deployed' and 'Success'. The pipeline details include the CNAME 'wp-u3.vm.appdrag.net' and the deployment method 'Docker Compose'. There are buttons for 'Open terminal' and 'Delete Pipeline'. The 'Build & Deploy' tab is active, showing a 'Credentials' section with a 'Display credentials' button. Below this, there are 'Manage Stack' buttons: 'View running logs', 'Restart Stack', and 'Stop Stack'. The 'Move Pipeline' section has a 'Move Pipeline' button. The 'Created By' field shows 'You', and the 'Last Deployment' and 'Creation time' are both 'June 27th 2022, 12:21 pm'. The left sidebar shows the project name, services, volumes, load balancer, CI/CD, members, billing, settings, and audit trail. The bottom left shows a balance of \$426.04 in credits and a spending of \$0.0420/hour.

elestio

← Back to target

wp (karpeto-dev)

CI/CD Deployed Success

Open terminal Delete Pipeline

PROJECT: shared-dev

Services

Volumes

Load Balancer

CI/CD

Members

Billing

Settings

Audit Trail

\$426.04 CREDITS

\$0.0420/hour SPENDING

Add credits

Details Tools **Build & Deploy** History Domain Management

Resync Pipeline Apply Changes

1. Docker Compose

```
docker-compose.yml
17 wordpress:
18   depends_on:
19     - database
20   image: wordpress:${SOFTWARE_VERSION_TAG}
21   restart: always
22   user: "root:root"
23   ports:
24     - 172.17.0.1:9000:80
25   env_file:
26     - .env
27   environment:
28     - WORDPRESS_DB_HOST=database:3306
29     - WORDPRESS_DB_USER=${MYSQL_USER}
30     - WORDPRESS_DB_PASSWORD=${MYSQL_PASSWORD}
31     - WORDPRESS_DB_NAME=blog_wp
32   volumes:
33     - ./php.ini:/usr/local/etc/php/conf.d/custom.ini
34     - ./wordpress:/var/www/html
35   networks:
36     - hloa-network
```

Environment variables

Use Environment variables to store configuration values, API keys and secrets. You can access them in your service like regular environment variables.

```
.env
1 SOFTWARE_VERSION_TAG=latest
2 MYSQL_ROOT_PASSWORD=0kdRw1oa-C5r1-LtTpZ48x
3 MYSQL_USER=wpdbuser
4 MYSQL_PASSWORD=0kdRw1oa-C5r1-LtTpZ48x
5
```

Revision #13

Created 2022-06-29 15:21:33 UTC by Joseph Benguira

Updated 2022-09-29 13:34:44 UTC by Amit Shukla