

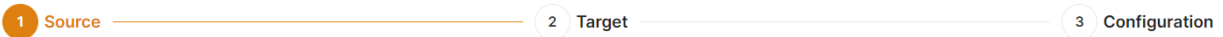
Deploy your first CI/CD pipeline on Elestio

You are about to learn how to deploy an application from a Git repository to production on any cloud.

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

1) Select your source

Create CI/CD pipeline



1. Deployment Method

i 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 applicatio...

2. Git repository

Import Git Repository

Clone Template

Import Git Repository

Git Account

amitshuklabag

Git Scope

elestio-examples

Search repo

airbyte

19 days ago

Import

angular

3 months ago

Import

astro

3 months ago

Import

bookstack

a month ago

Import

brunch

3 months ago

Import

Import Third-Party Git Repository →

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

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 Elestio dashboard on the left and the 'Create CI/CD pipeline' workflow on the right. The workflow consists of three steps: 1. Source, 2. Target (currently active), and 3. Configuration. The 'Target' step is titled '1. Choose Deployment Targets' and includes a question: 'Where do you want this service to be deployed?'. Below this, there are two buttons: 'New Infrastructure' (highlighted with an orange border) and 'Existing Infrastructure'. Underneath, the 'Deployment mode' section offers 'Single Node' (highlighted) and 'Cluster' options. The next step is '2. Select Service Cloud Provider', showing six provider cards: HETZNER, DigitalOcean, Amazon Lightsail (highlighted), linode, and VULTR. The final step is '3. Select Service Cloud Region', with tabs for Europe, North America (highlighted), and Asia. Under 'North America', two region options are shown: 'ca-central-1' (Canada - Montreal, highlighted with an orange border) 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 project

This is the last step of the process where you can configure your project name, branch, runtime, and all other settings about your build and environment configuration.

a) global settings

Select the Runtime & version matching your project needs. If you are using a framework select it in the framework dropdown, this will auto-populate the build/run commands.

Create CI/CD pipeline



Configure your nodejs-express-pug Project

i Define your project build and run

Project Name	nodejs-express-pug	Branch	main
Runtime	NodeJs	Version	16.15 (23-06-2022)
Framework	No Framework	Root Directory	/

- Build and Output Setting +
- Life cycle scripts (optional) +
- Environment variables +
- Volume config +
- Exposed Ports +
- Reverse proxy configuration +

b) Build settings

You can customize the install/build/run command to suit your requirements.

Build and Output Setting

Install command	Run command
npm install	npm start
Build command	Build Output dir
npm run build	/

c) Life cycle scripts

In some situations, you will need to execute scripts before or after the installation of a new pipeline to setup your env, install some dependencies, and copy the dataset, ... In those cases, you can define pre/post scripts to execute before/after an installation and other actions like backup/restore. To activate it just indicate your script path relative to the root folder of your git repository.

Life cycle scripts (optional)

Pre install command	Post install command
<input type="text" value="./scripts/preInstall.sh"/>	<input type="text" value="./scripts/postInstall.sh"/>
Pre Backup command	Post Backup command
<input type="text" value="./scripts/preBackup.sh"/>	<input type="text" value="./scripts/postBackup.sh"/>
Pre Restore command	Post Restore command
<input type="text" value="./scripts/preRestore.sh"/>	<input type="text" value="./scripts/postRestore.sh"/>

d) Environment variables

In most case, 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

i 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
```

e) Volumes (data storage)

A lot of apps are totally stateless and don't require any volumes, but some of them need persistent storage to store file uploads, config, logs and other files. You can define one or multiple volumes as folders from the host (CI/CD target instance) mounted into the container. That way the files are persisted and available to the container.

Volume config

Host path	Container path	
<input type="text" value="./data"/>	<input type="text" value="/data"/>	
+ Add Another		

Host path must be relative and must start with ./

f) Exposed ports

If your app is listening on port 3000, you should indicate Container port to 3000, then Host port can be the same or anything else. If your app is listening on multiple ports you can add them as additional rows by clicking on "Add another".

Exposed Ports

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.

Interface	Host Protocol	Host Port	Container Port	
<input type="text" value="172.17.0.1"/>	<input type="text" value="HTTP"/>	<input type="text" value="3000"/>	<input type="text" value="3000"/>	
+ Add Another				

If you need to deploy several instances of the same app on a single node you will have to change the host port in the exposed ports > host port and also in reverse proxy > targetport accordingly.

g) Reverse proxy

Finally 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 previous step, so here is port 3000.

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	→	Protocol	IP	Port	Path
HTTPS	443		HTTP	172.17.0.1	3000	/

Require Basic Auth

+ Add Another

It's possible to activate Basic Authentication if you check the corresponding checkbox and define login and password

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	→	Protocol	IP	Port	Path
HTTPS	443		HTTP	172.17.0.1	3000	/

Require Basic Auth

Login: root Password: 123

+ Add Another

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



Provider
Amazon Lightsail

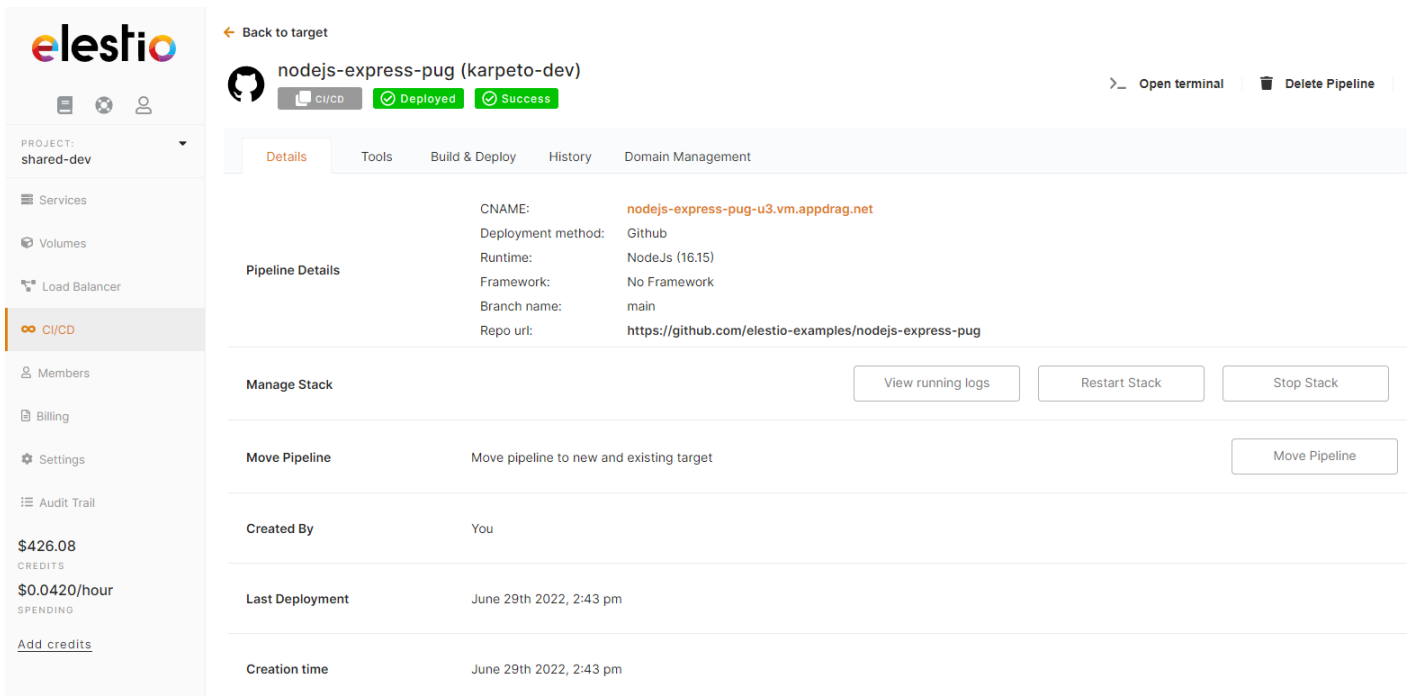
Region
North America, Canada
Montreal

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. Also, each time you commit to your repo code will be rebuilt & re-deployed.



The screenshot shows the Elestio dashboard for a project named 'shared-dev'. The 'CI/CD' section is active, displaying details for a pipeline named 'nodejs-express-pug (karpeto-dev)'. The pipeline status is 'Deployed' and 'Success'. The dashboard includes a sidebar with navigation options like Services, Volumes, Load Balancer, CI/CD, Members, Billing, Settings, and Audit Trail. The main content area shows pipeline details, stack management options (View running logs, Restart Stack, Stop Stack), and a 'Move Pipeline' button. The pipeline details include CNAME, deployment method (Github), runtime (NodeJs 16.15), framework (No Framework), branch name (main), and repo url (https://github.com/elestio-examples/nodejs-express-pug). The 'Created By' field shows 'You', and the 'Last Deployment' and 'Creation time' are both listed as 'June 29th 2022, 2:43 pm'.

Field	Value
CNAME	nodejs-express-pug-u3.vm.appdrag.net
Deployment method	Github
Runtime	NodeJs (16.15)
Framework	No Framework
Branch name	main
Repo url	https://github.com/elestio-examples/nodejs-express-pug
Created By	You
Last Deployment	June 29th 2022, 2:43 pm
Creation time	June 29th 2022, 2:43 pm

Revision #26

Created 2022-06-29 08:46:08 UTC by Joseph Benguira

Updated 2022-09-29 13:31:55 UTC by Amit Shukla