

Putting a Node guestbook on Zerops

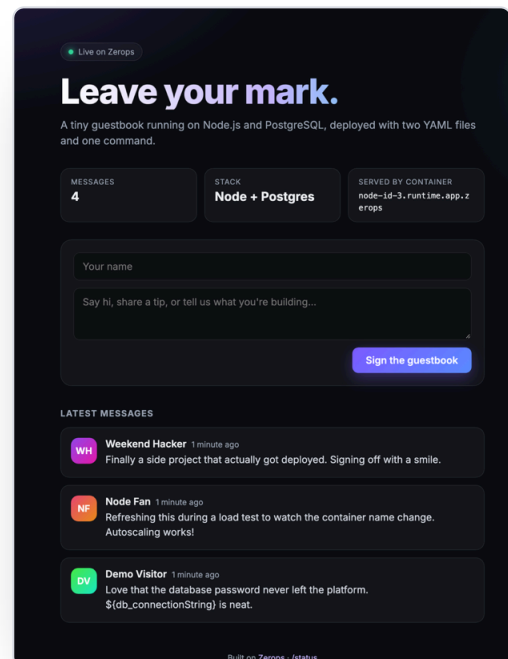
Express, PostgreSQL, and the two files it took.

Deni · @DeniCodes

Here's the project if you wanna add some thoughts:
app-3244-3000.prg1.zerops.app

Well, why a guestbook? I wanted the smallest app that still has every piece a real app needs. It takes input from strangers, stores it in a database, reads it back, and must be reachable over HTTPS. A static page would skip the parts I actually wanted to test, and anything bigger would bury the deployment under app code. A guestbook sits right in between, and it gives anyone who visits something simple to do.

I deployed it on Zerops to see what hosting it looks like when someone else runs the servers. Zerops is a PaaS that runs on its own hardware. Not a dashboard on top of AWS, btw.



The finished guestbook.

The guestbook

Visitors leave a name and a short message, and Express saves them in PostgreSQL. The app reads `DATABASE_URL`, listens on port 3000, and has no Zerops SDK in it.

SERVER.JS

```
import express from 'express';
import pg from 'pg';
import os from 'node:os';

const pool = new pg.Pool({ connectionString: process.env.DATABASE_URL });
const app = express();
app.use(express.urlencoded({ extended: false }));

app.get('/', async (_req, res) => {
  const { rows } = await pool.query(
    'SELECT name, message, created_at FROM entries ORDER BY created_at DESC LIMIT 50'
  );
  res.send(page(rows)); // HTML, including os.hostname()
});

app.post('/entries', async (req, res) => {
  const { name, message } = req.body;
  if (name?.trim() && message?.trim()) {
    await pool.query(
      'INSERT INTO entries (name, message) VALUES ($1, $2)',
      [name, message]
    );
  }
  res.redirect('/');
});

app.get('/status', async (_req, res) => {
  await pool.query('SELECT 1');
  res.json({ status: 'ok' });
});

await pool.query(`CREATE TABLE IF NOT EXISTS entries (
  id SERIAL PRIMARY KEY,
  name TEXT NOT NULL,
  message TEXT NOT NULL,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now()
)`);

app.listen(3000);
```

The snippet leaves out `package.json` and `page()`, the HTML template. The imports use ESM, so `package.json` needs `"type": "module"`, and it needs a start script because the run config calls `npm start`. It also needs `express` and `pg` as dependencies and a committed `package-lock.json`, because the build runs `npm ci`. And `page()` has to escape the name and message before printing them, since strangers write both.

PACKAGE.JSON

```
{
  "type": "module",
  "scripts": {
    "start": "node server.js"
  }
}
```

Make sure to pay attention to these three things though:

1. The page prints `os.hostname()`. With one container that tells you nothing. Once a second container starts, the name on the page changes, so I can watch the load balancer spread requests without opening the dashboard.
2. `/status` runs `SELECT 1`. It's the health check, so if the app can't reach the database, the check fails.
3. The table gets created at startup. One container is fine, but multiple containers starting at the same time could race each other. Before raising the container count, I'd move it into a migration. `initCommands` in the run block runs on every container start, so wrap the migration in `zsc execOnce` to make sure only one container does the work.

The import

On Zerops, a project holds your services. Every project gets its own firewall, HTTP balancer, logs, and stats, and its services share a private network. The app reaches the database at the hostname `db`. I didn't set up any networking for that.

This is the import file used:

ZEROPS-PROJECT-IMPORT.YML

```
project:
  name: zerops-guestbook

services:
  - hostname: db
    type: postgresql:single@16
    priority: 1

  - hostname: app
    type: nodejs@22
    enableSubdomainAccess: true
    minContainers: 1
    maxContainers: 3
```

It's slightly different from the one I actually imported. Mine said `type: postgresql@16` with `mode: NON_HA`, which is what their Node recipe still shows, and it worked fine. The current docs put the mode inside the type instead. `postgresql:single@16` is one container. `postgresql:ha@16` is a three-node cluster with failover. You can't switch a service from one to the other later; you have to delete it and recreate it. The available versions are 18, 17, 16, and 14.

`priority: 1` makes the database start first. `enableSubdomainAccess` gives the app a Zerops subdomain with HTTPS terminated in front of it, though the docs mark that subdomain as not suitable for production. The app can run anywhere from 1 to 3 containers.

Vertical scaling is on by default. CPU, RAM, and disk follow actual usage, and RAM can drop as low as 0.125 GB (0.25 GB for Postgres). I ran Node 22, but 24, 20, and 18 are supported too.

You still create the account and access token by hand. You'll find Token Management under Settings in the dashboard. After that, it's three commands:

SHELL

```
curl -L https://zerops.io/zcli/install.sh | sh
zcli login <your-token>
zcli project project-import zerops-project-import.yml
```

I was on Lightweight, the free core plan. Both services showed up as active in the notification feed.

zerops.yml

The docs call this file `zerops.yaml`, but `zerops.yml` worked too. The `setup` value has to match the service hostname, otherwise the push has nowhere to go.

```
ZEROPS.YML

zerops:
  - setup: app
    build:
      base: nodejs@22
      buildCommands:
        - npm ci
      deployFiles:
        - ./server.js
        - ./package.json
        - ./node_modules
      cache: node_modules
    run:
      base: nodejs@22
      ports:
        - port: 3000
          httpSupport: true
      envVariables:
        NODE_ENV: production
        DATABASE_URL: ${db_connectionString}
      start: npm start
      healthCheck:
        httpGet:
          port: 3000
          path: /status
```

The build runs in its own container. It runs `npm ci`, caches `node_modules`, and ships only the files listed under `deployFiles`. `httpSupport` marks port 3000 as an HTTP port, which lets the balancer terminate HTTPS and route traffic to it.

When Zerops creates the Postgres service, it generates a user, a password, and a connection string. The user and the database are both called `db`. Other services don't see those variables automatically. You reference them with the hostname, an underscore, and the key name.

Watch the spelling. A wrong reference isn't caught at deploy time. The app receives the literal text and fails when it tries to connect.

The health check calls `/status` from inside the container. If it keeps failing, Zerops restarts the container.

The push

My first `zcli push` failed before it built anything:

```
ZCLI PUSH
```

```
ERR folder is not initialized via git init and flag --noGit was not set
```

It wants a git repository, not a GitHub remote. Running `git init` and making one commit is enough. The other way out is the flag from the error. The command reference spells it `--no-git`, but my CLI printed `--noGit`, so use whichever your version asks for. Keep in mind that the flag uploads the folder exactly as it is, including `node_modules` if you left it there.

I ran it again and followed along with `zcli service log --show-build-logs --follow`.

From running the command to seeing “Application is deploying” took about a minute. So that's pretty quick. After that, the page was live on the subdomain over HTTPS. Every redeploy starts a new container, so the name on the page changes each time. The last one I saw was `node-id-3.runtime.app.zerops`.

The app ended up at `app-3244-3000.prg1.zerops.app`. The `prg1` part suggests Prague. I wouldn't count on that URL staying the same (you can always get a domain).

What came with it

The dashboard already includes logs, resource graphs, and pipeline history with no setup. The timings above come from the build log. Here are two entries from the pipeline history:

PIPELINE HISTORY

```
Pipeline 6JFzD7...MqkCfw for zerops-guestbook/app  
Finished in 1 minutes 2 secs
```

```
Pipeline BrpCHT...Un5IYw for zerops-guestbook/app  
Finished in 59 secs
```

The database doesn't have a public port. To reach it from my laptop, I went through the VPN:

SHELL

```
zcli vpn up  
psql "postgresql://db:<password>@db:5432/db?sslmode=disable"
```

Port 5432 doesn't support TLS, so if your client requests it, the handshake fails. That's why the connection uses `sslmode=disable`; the VPN tunnel already handles the encryption. The user and password are under "Peek access details" on the service. The environment variables don't come through the VPN with you.

Cost

Containers are billed by the minute, with rates listed per 30 days.

CONTAINER RATES, PER 30 DAYS

Resource	Rate
Shared CPU	\$0.60
Dedicated CPU	\$6.00
RAM	\$0.75 per 0.25 GB
Disk	\$0.05 per 0.5 GB

There's no per-seat fee. Each project also picks a core plan:

PROJECT CORE PLANS

	Lightweight	Serious
Price	Free	\$10 per 30 days
Build time	15 hours	150 hours
Egress	100 GB	3 TB
Backup storage	5 GB	25 GB

Past those limits, egress is \$0.02 per GB, backup is \$0.50 per 5 GB, and build time is \$0.50 per 15 hours. A build this short won't come anywhere near the free hours. Full details: zerops.io/pricing

Because CPU and RAM scale on their own, you aren't billed for the size you started with. Their quickstart puts a small app with a database at about \$3 to \$5 a month, and it recommends starting from a recipe rather than writing these two files yourself. I skipped the recipe because I wanted to see the config.

Where I wouldn't use it

Static sites, simple landing pages, or something personal.

If there's no server and no database, Vercel, Netlify or TiinyHost works quick (drag and drop as well).

Apps that need to run in Asia, South America or Africa.

Zerops has Prague, New York and Seattle today.

Projects where you want years of Stack Overflow answers and blog posts to search when something breaks.

Zerops is young and small, so their docs and the team are your main support. They're pretty active on X, so I wouldn't assume long waiting times in any case!

For a small Node app with a database, though, it came down to two files and one push. I never had to worry about the backend, servers, or anything like that.

If you've deployed on Zerops, I want to know how it went.

Deni · @DeniCodes