

Distributed Systems Challenge

Pastebin

Tobias Keel und Noah Fleischmann

Project description

You all know the requirements (Dockerized, Frontend, Backend, Scheduled Task, Persistence...)

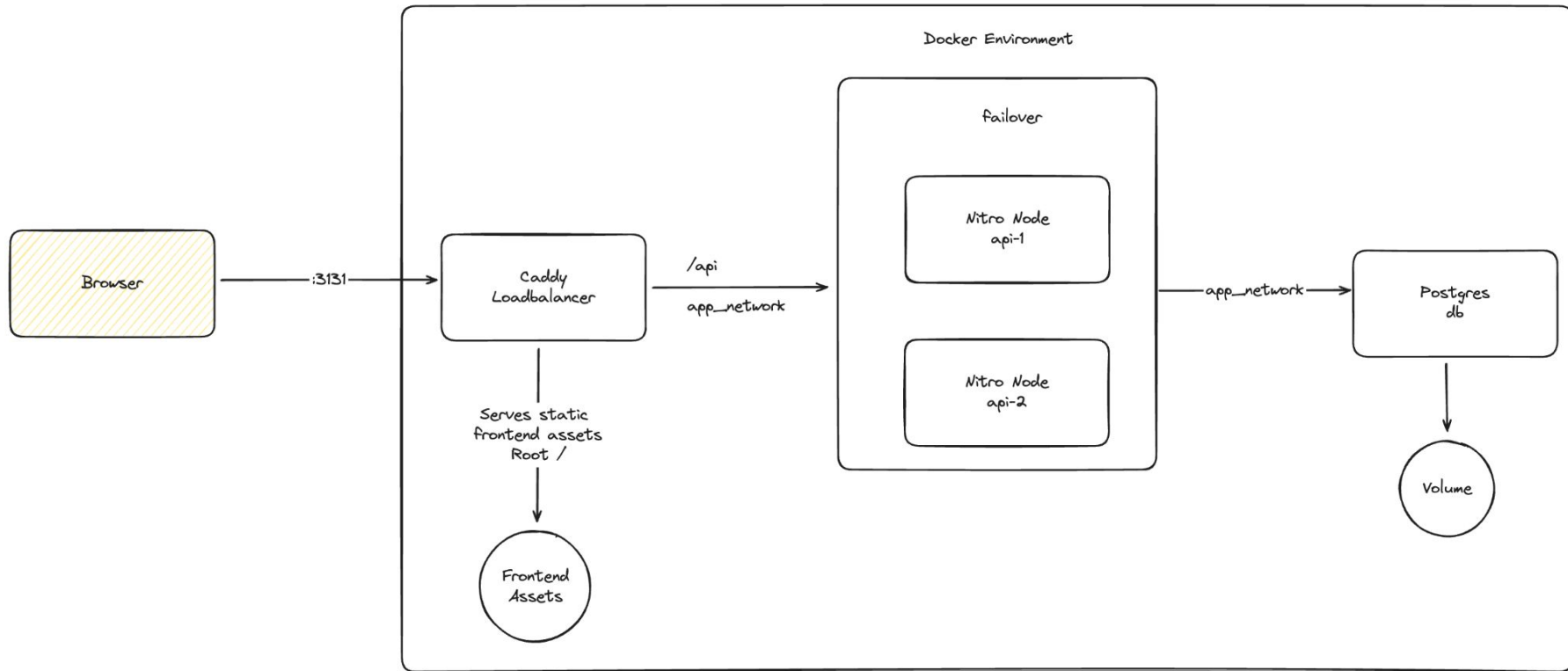
Pastebin Clone - Submit and View text

Extremely simple code - focus on distributed infrastructure

React Vite Frontend, Nitro Node.js Backend, Postgres, Caddy

Orchestrated with Docker Compose

Architecture



Architecture

Caddy load balancer terminates all incoming connections

The path /api is proxied to the backend instances through the Docker internal network

Everything else is served by the frontend, which is a React app that is compiled into static assets at build time

The API is a Nitro Node.js app that is connected to a Postgres database with failover enabled

The Postgres Database has a Volume to persist data

Frontend

React App served by Caddy

Built using Vite



The screenshot shows the 'Budget Pastebin' web application. At the top, the title 'Budget Pastebin' is displayed in a monospace font. Below the title, there is a form with two main sections. The first section contains a text input field labeled 'Enter paste ID' and a blue button labeled 'Access Paste'. The second section contains a larger text input field labeled 'Enter new paste text' and a blue button labeled 'Create Paste' at the bottom left.

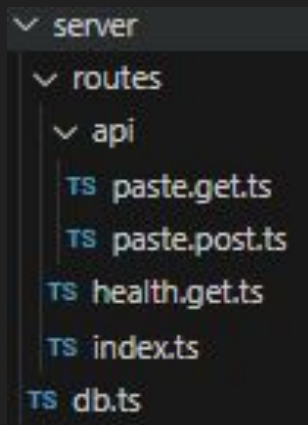
Backend

Multiple instances

Routed through Caddy

Rest API

Node.js & Nitro



```

server
├── routes
│   ├── api
│   │   ├── TS paste.get.ts
│   │   ├── TS paste.post.ts
│   │   ├── TS health.get.ts
│   │   ├── TS index.ts
│   │   └── TS db.ts

```

Load Balancing

Caddy!

All incoming connections go through Caddy

Reverse Proxy, sends /api traffic to backend

CORS is not an issue

Load Balancing through Round Robin

Health checks every 5s

```
reverse_proxy /api/* api-0:3000 api-1:3000 {  
  
    lb_policy round_robin  
  
    lb_retries 2  
    fail_duration 10s  
  
    health_uri /health  
    health_interval 5s  
    health_timeout 2s  
    health_status 2xx  
}
```

Database

Postgres

Database and both backend instances get credentials from .env

Data persisted through Docker volume

Script to create tables on startup (IF NOT EXIST)

```
CREATE TABLE IF NOT EXISTS pastes (  
  id SERIAL PRIMARY KEY,  
  paste TEXT NOT NULL,  
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

Scheduled Task

Cron running in container

Deleting 'Pastes' that are older than two days

Writes to log file

```
DELETE FROM pastes WHERE created_at < now() - interval '2 days';
```


Live Demo