One-Shot File Service on AWS

1. Project Goal

Create a lightweight web service that lets users upload any file **exactly once** and share a single-use download link.The implementation takes place in Amazon Web Services in order to operate this service in the cloud.

2. Feature Highlights

- Simple HTML/JS front-end (React + Vite) for drag-and-drop upload
- Go back-end exposes two endpoints: `POST /upload` & `GET /d/:id`
- Uploads stored as files on disk; metadata in SQLite
- Download link becomes *410 Gone* after first successful fetch
- Zero external dependencies – perfect for cheap t3a.micro

3. Architecture Overview

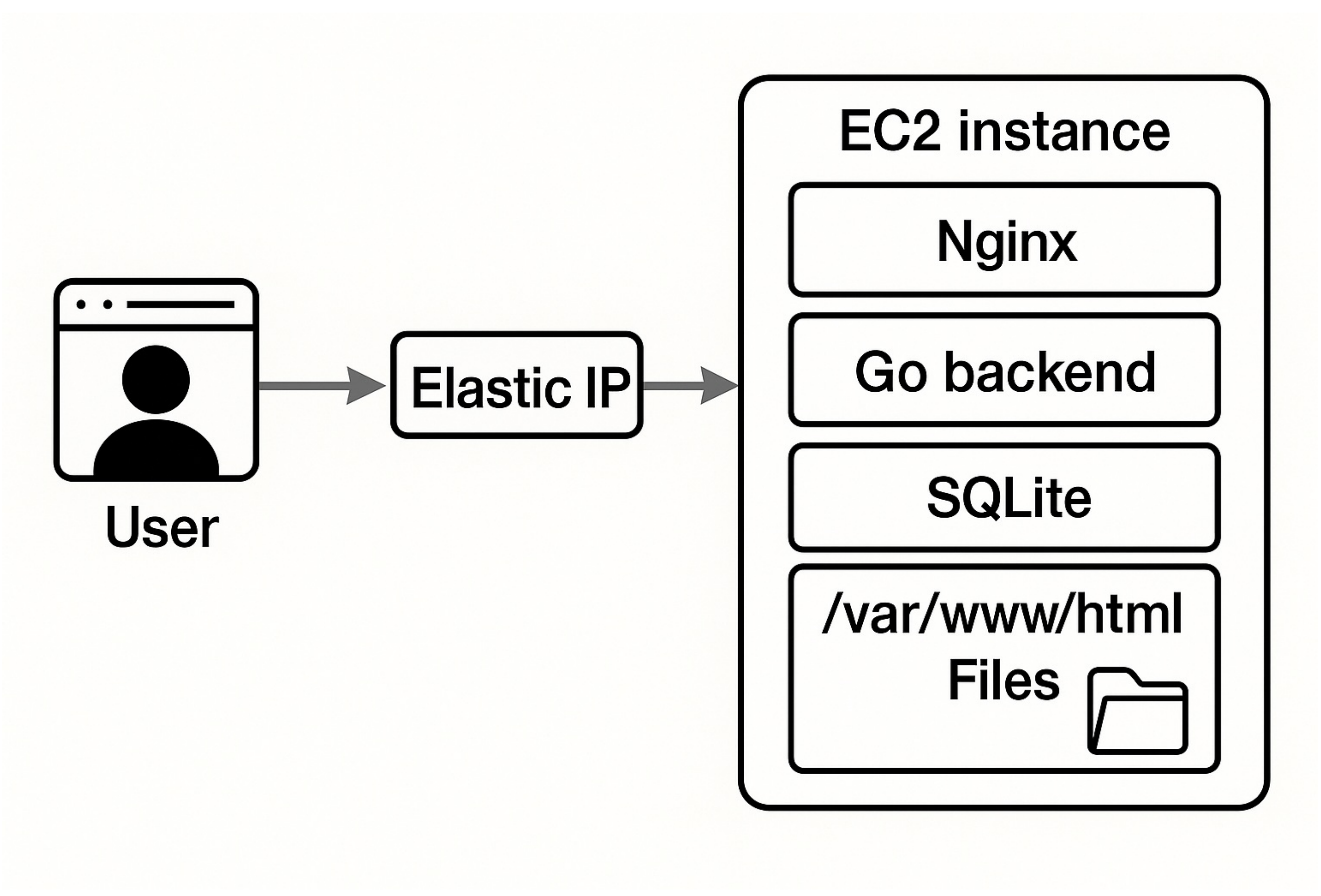


Figure 1 – Overall AWS architecture

3.1 AWS Components

Service	Purpose in Project
EC2 (t3a.micro)	Hosts Go API + Nginx static front-end
Elastic IP	Provides permanent public address
Security Group	Opens ports 80 & 22 (SSH)

4. Implementation Details

4.1 Back-End (Go 1.23)

- `chi` router for minimal routing
- `sqlite3` stores `id | path | name | used`
- Systemd unit `zkn-share.service` ensures auto-restart

4.2 Front-End (React 18 + Vite 5)

- Single static SPA in `dist/`; served by Nginx
- Relative API calls → easy prod/dev switch

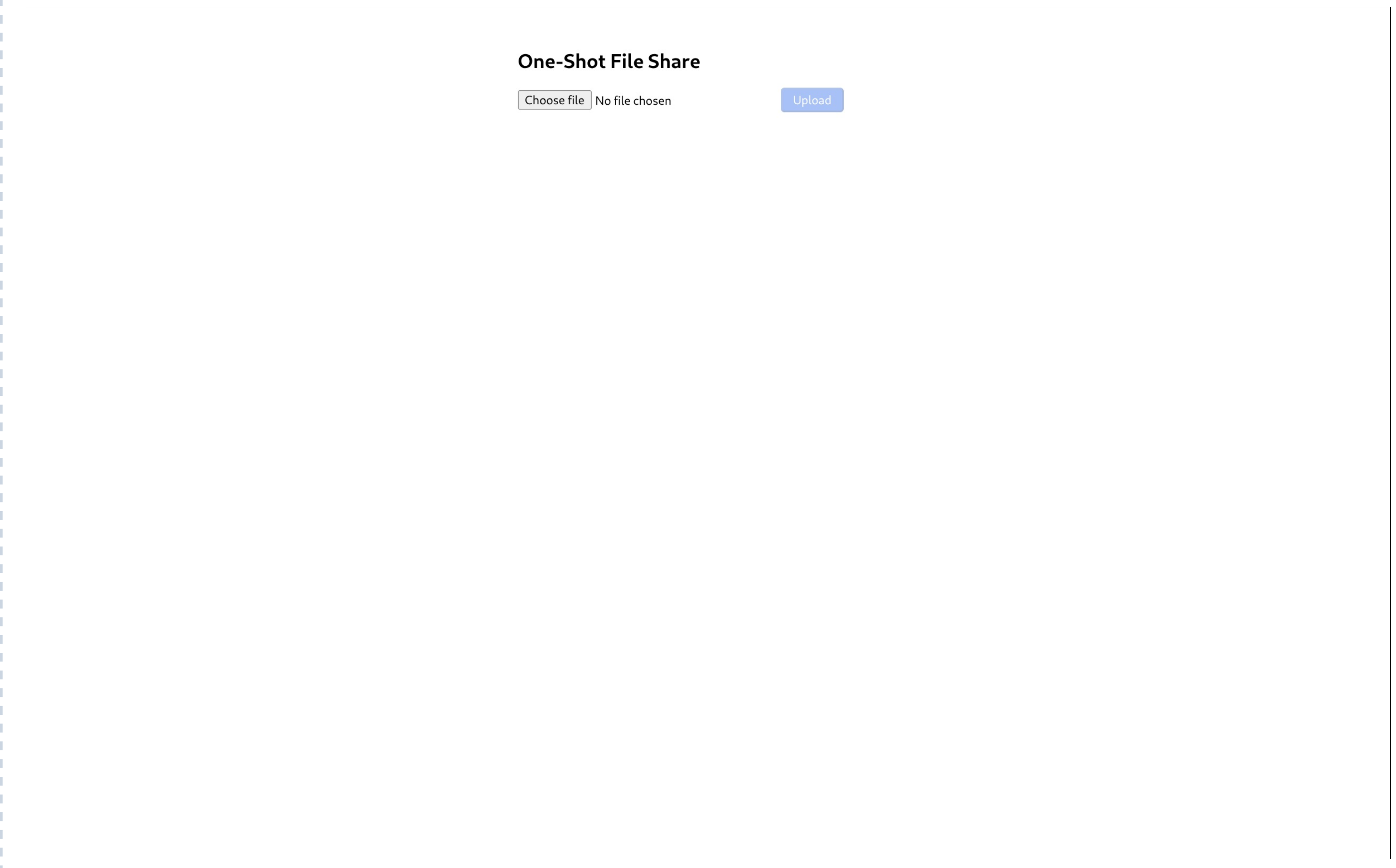
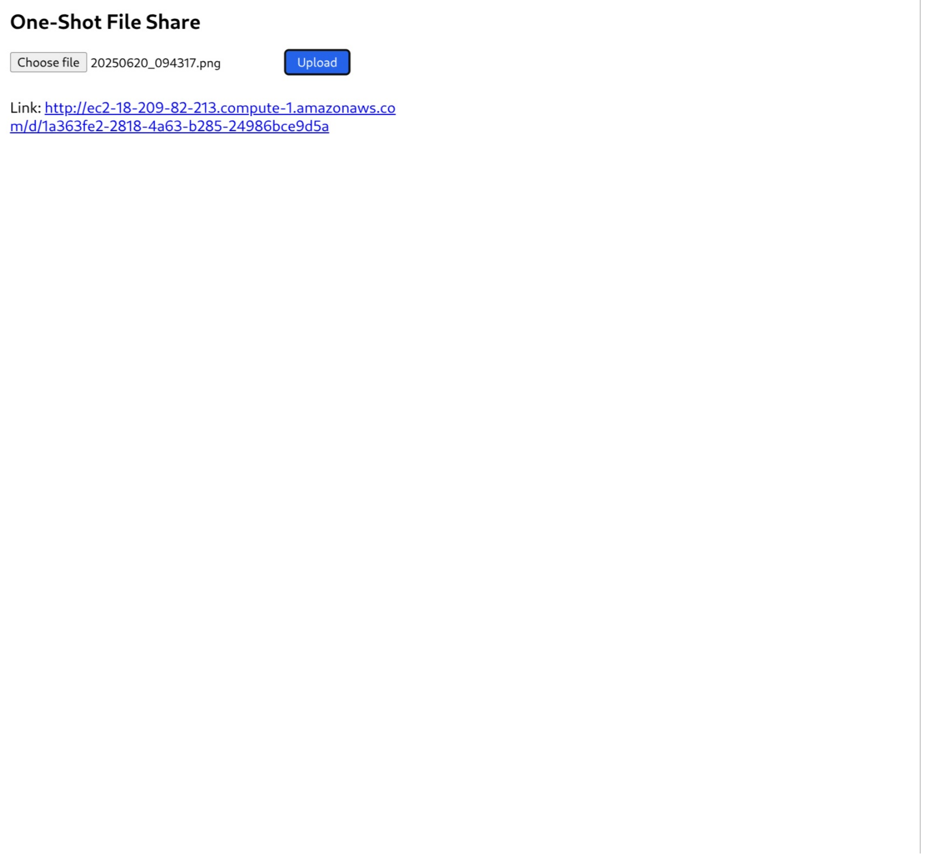

One-Shot File Share

Figure 2 – Upload screen


One-Shot File Share

Link: http://ec2-18-209-82-213.compute-1.amazonaws.com/d/1a363fe2-2818-4a63-b285-24986bce9d5a

Figure 3 – Uploaded file

5. Benchmarks

All measurements were executed from the developer workstation (100 Mbit/s uplink) using `ab`. Test file size: **20 MB**.

5.1 Download (100 req, 50 concurrency)

```
$ ab -n 100 -c 50 http://http://ec2-18-209-82-213.compute-1.amazonaws.com/test.bin
Requests/sec      : 0.46
Transfer rate     : 9.2 MB/s
90% completed in  : 142 s
Time per request  : 108.9 s (mean)
Failed requests   : 1 (length mismatch)
```

6. Deployment

- 🗂 Source code: github.com/ansnsr42/oneshot-share
- Local build (Go binary + Vite `dist/`)
- Upload to EC2 via `git pull`
- `systemctl restart one-share` & `nginx -s reload` – zero downtime

7. Conclusion

This project shows how a tiny stack (Go + Nginx + SQLite) enables a full-fledged AWS service. Through Elastic IP, the service is permanently under `http://ec2-18-209-82-213.compute-1.amazonaws.com/` accessible. Although a t3a.micro instance only delivers ~30 Mbit/s, this is sufficient for school and demo purposes; Scaling is possible at any time.