

Build modern JavaScript applications on AWS

cloud native, simplicity, high availability, scalability

Show of hands 🙋



Ivan Barlog

AWS Solutions architect @ BeeSolve

Fullstack TypeScript developer



AWS certified Solutions Architect



GitHub: [ivanbarlog](https://github.com/ivanbarlog)

email: ivan@barlog.sk

LinkedIn: [ivan-barlog](https://www.linkedin.com/in/ivan-barlog)



Use case: **Group Expense with Ease**

- personal hobby project
- inspired by SplitWise (splitwise.com)
- free of charge
- no ads, no tracking
- simplified - just basic features
- assumes you like your friends - splitting expenses equally
- in production but not finished



build with  by

BeeSolve



group-expense-with-ease.com

Code stack

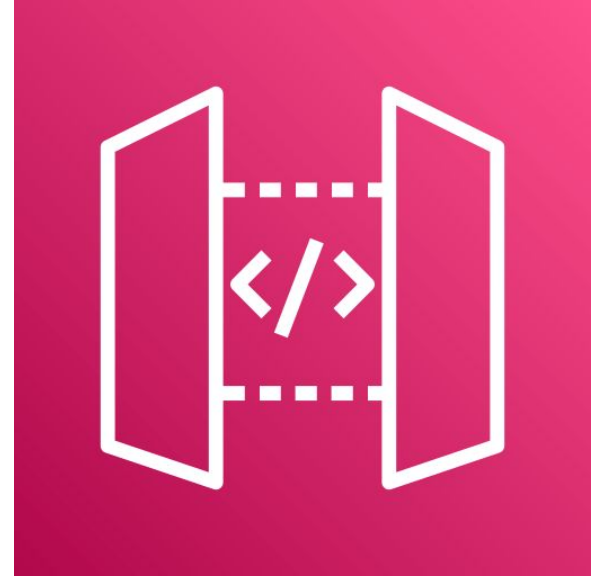
- Bun - TypeScript runtime, bundler and package manager
- React + TanStack Router - frontend rendering and routing
- tRPC + TanStack Query - API communication + state management
- AWS SDK
- Valibot
- AWS CDK

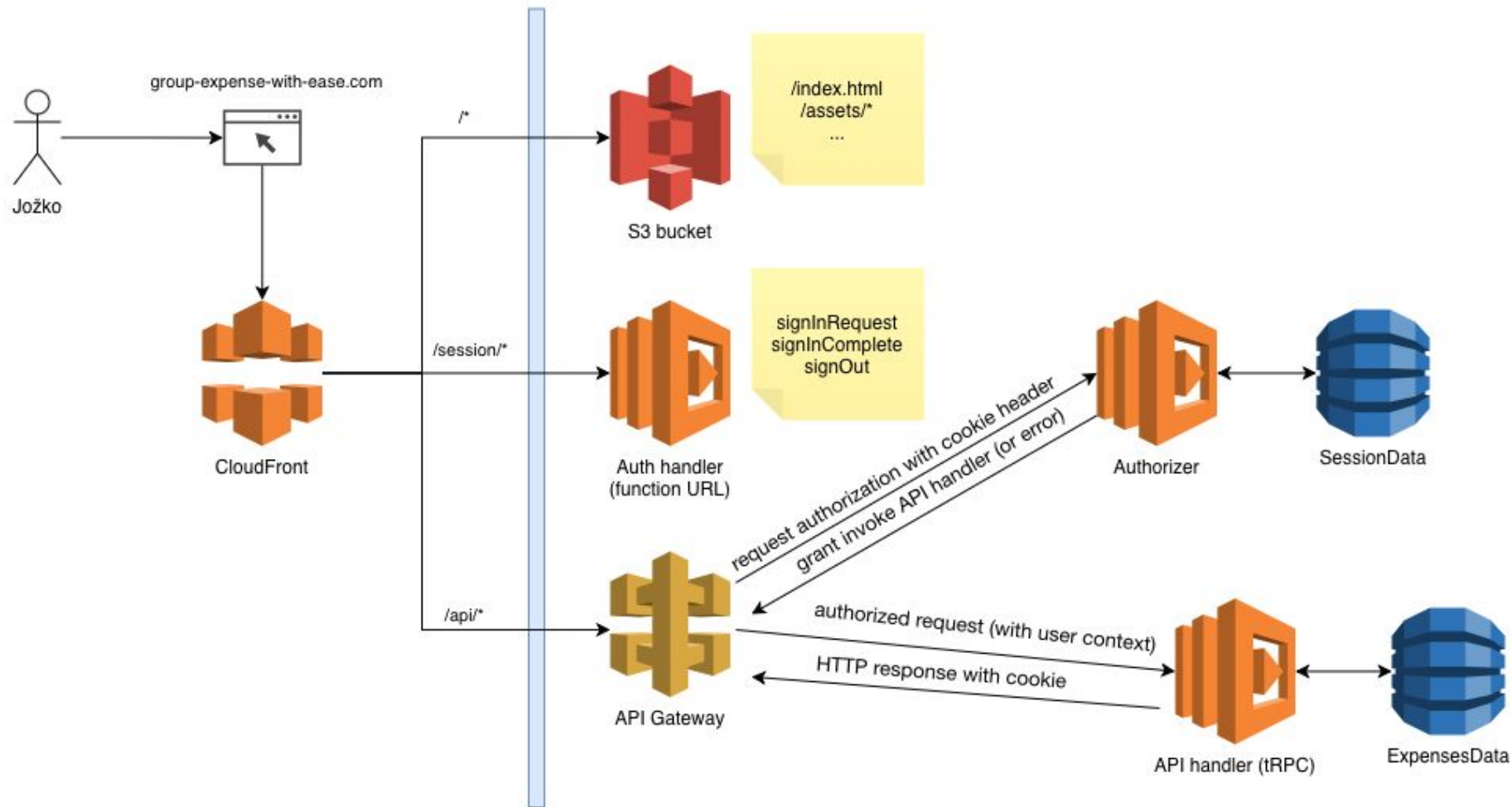
Infrastructure stack

- AWS native services - no Kubernetes, no abstractions
- Lambda - compute
- DynamoDB - NoSQL key-value store
- HTTP API Gateway + Lambda Authorizer
- CloudFront
- SQS
- SES
- S3

Authentication and authorization

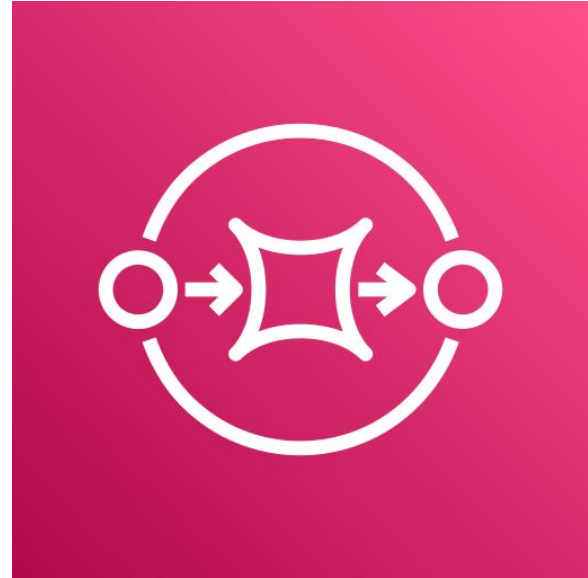
- custom solution
- cookie based authorization [#use-the-platform](#)
- `__Host-` cookies
- HTTP API Gateway + Lambda authorizer

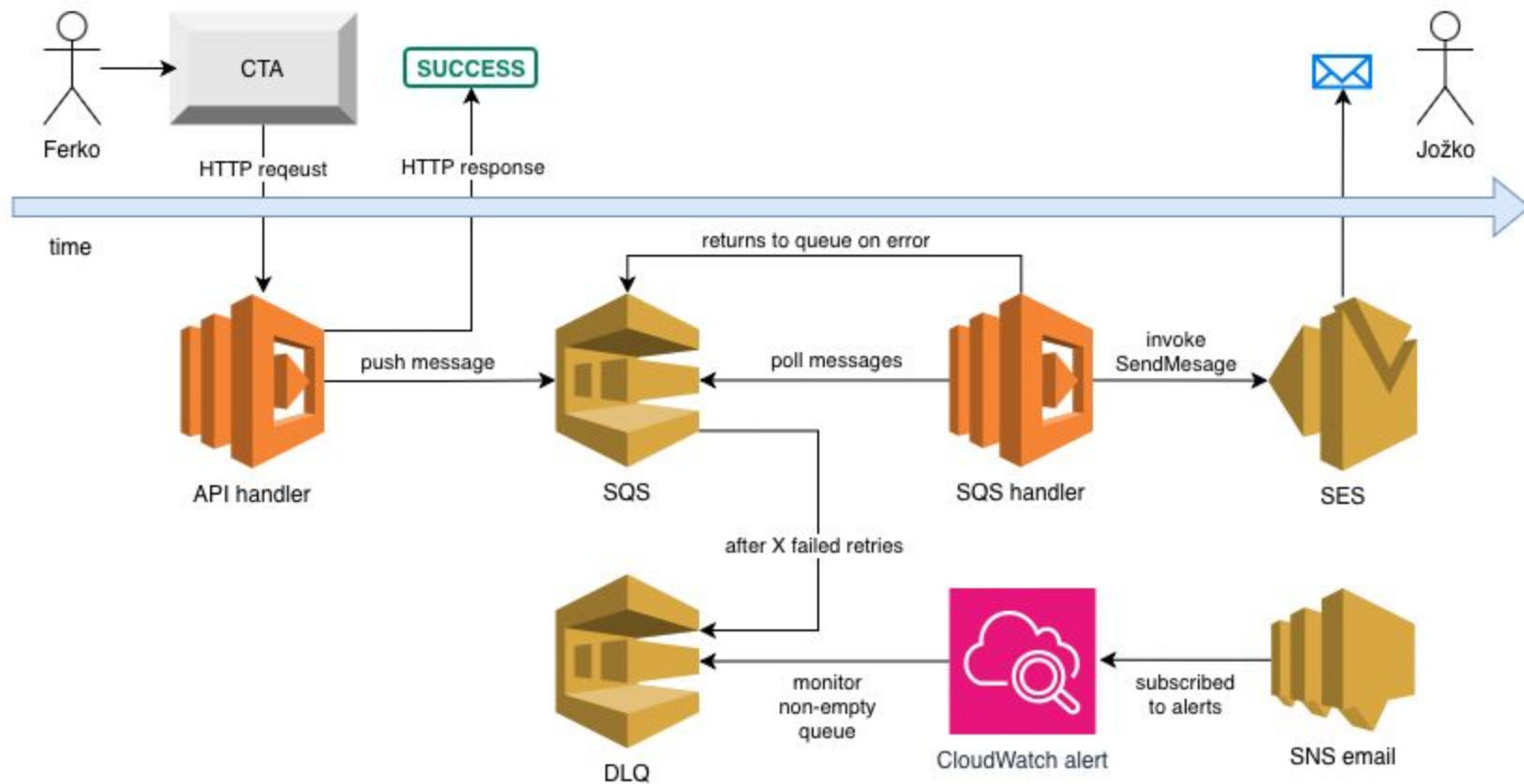




Asynchronous tasks with SQS

- decoupling
- quick response to user
- retry on failure
- buffering
- dead letter queue (DLQ)





Local development with Bun

- bundling React out-of-box - just import your html file
- native fetch API - Request/Response web standard
- easy local HTTPS with devcert
- HMR (hot module replacement) and everything
- it just works
- 1 file setup



Relations in NoSQL key-value store

- no schema enforcement
- no foreign keys enforcement
- denormalize your data
- logic and schema validation
is in your application (valibot)
- transactions vs streams
- great for OLTP not so great for OLAP
(but possible to some extent)



Infrastructure price - assumptions

1 user will do 100 requests per day when active

1 group is active in average for 7-14 days

1 user will use application in average 5-10 times (active groups) a year

$10 \text{ active groups} * 14 \text{ days} * 100 \text{ requests/day} \sim 14\text{k requests per year}$

$14\text{k} / 12 \text{ months} \sim 1200 \text{ requests per month}$

Infrastructure price - estimate for 1k active users

[AWS Pricing Calculator](#)

Estimate summary [Info](#)

Upfront cost

0.00 USD

Monthly cost

10.37 USD

Total 12 months cost

124.44 USD

Includes upfront cost

Maintenance

- try to have as little dependencies as possible
- build for your future self not for the computer
- build on latest runtimes
- monitor security updates and runtime support

Go build something!

...and maybe try
group-expense-with-ease.com

