aws Community Day

11 March 2026, KOŠICE,
SLOVAKIA

DevOpsGroup

LABYRINTH LABS

ABB SOLUTIONS SLOVAKIA

eset Cybersecurity Progress. Protected.

T Systems

stormit

HACKIHUB

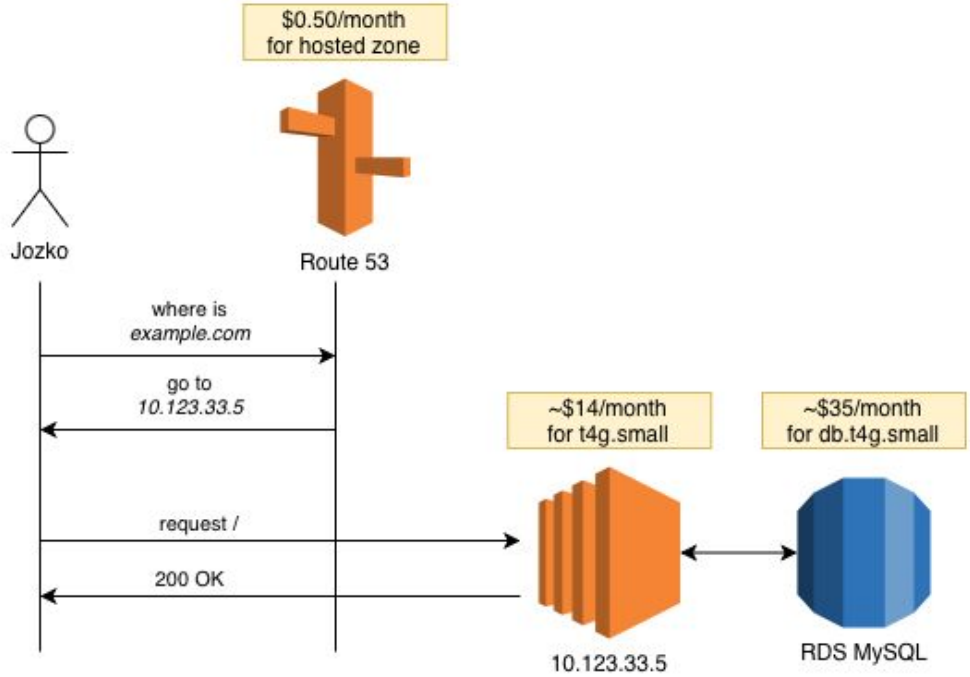
Building small:
Leverage AWS Serverless for Scalable applications

👋 Show of hands

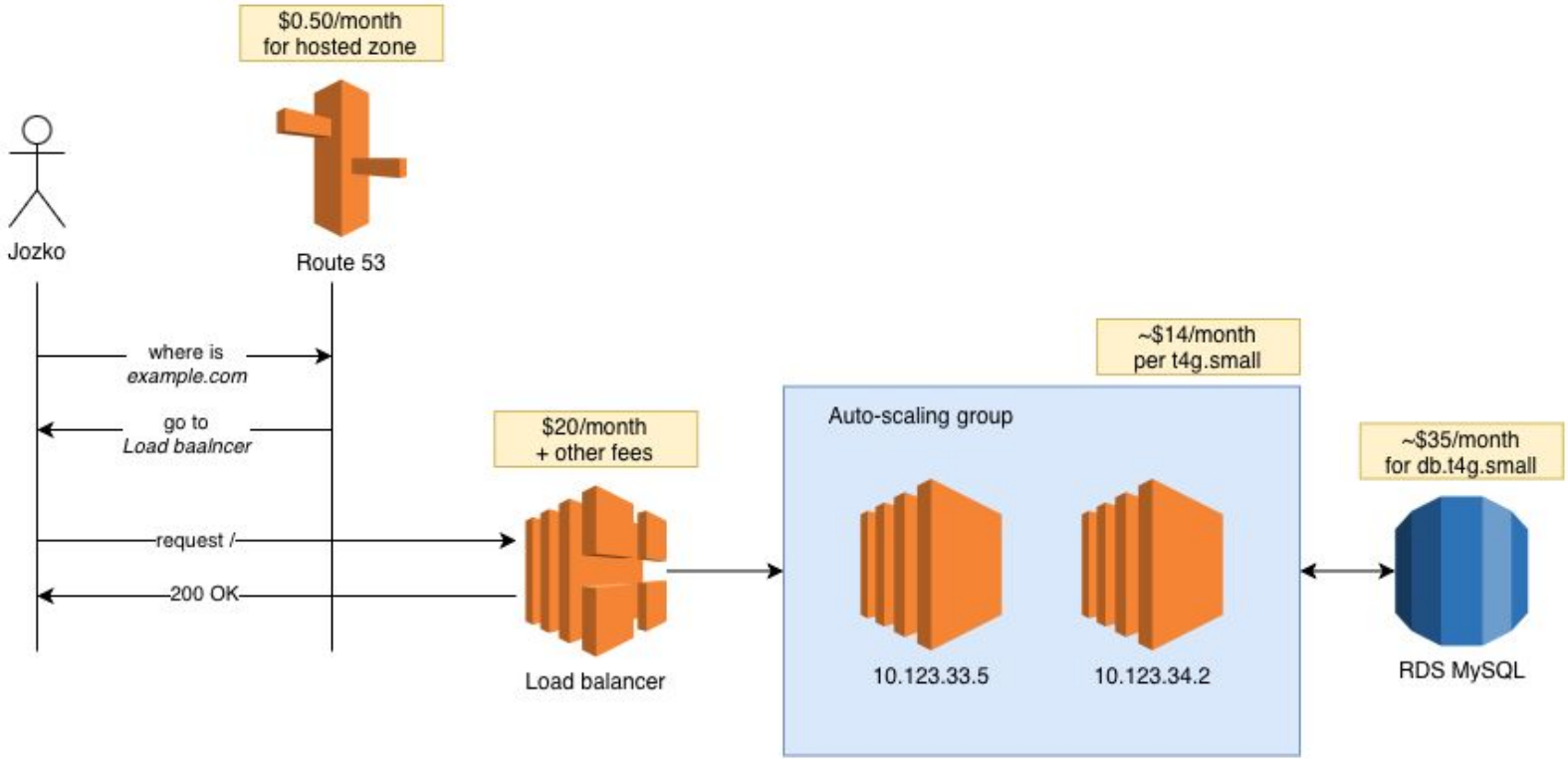
Traditional solution - single t4g.small EC2 instance

- very simple
- predicable price
- fixed resources (2 vCPU, 2GiB)
- easy deployment (SSH/SCP)

- no autoscaling out of box
- no redundancy
- deployment outages



Traditional solution - scalability fix



What is Serverless?

misnomer - a name that is incorrectly or unsuitably applied

The biggest secret of Serverless

(and many other serverless services)

Hundreds of thousands of
EC2 instances under-the-hood

Usman Khalid

He/His/Him

Head of Serverless Compute

linkedin.com/in/ahmedusmankhalid/

hundreds of thousands of servers.

Serverless

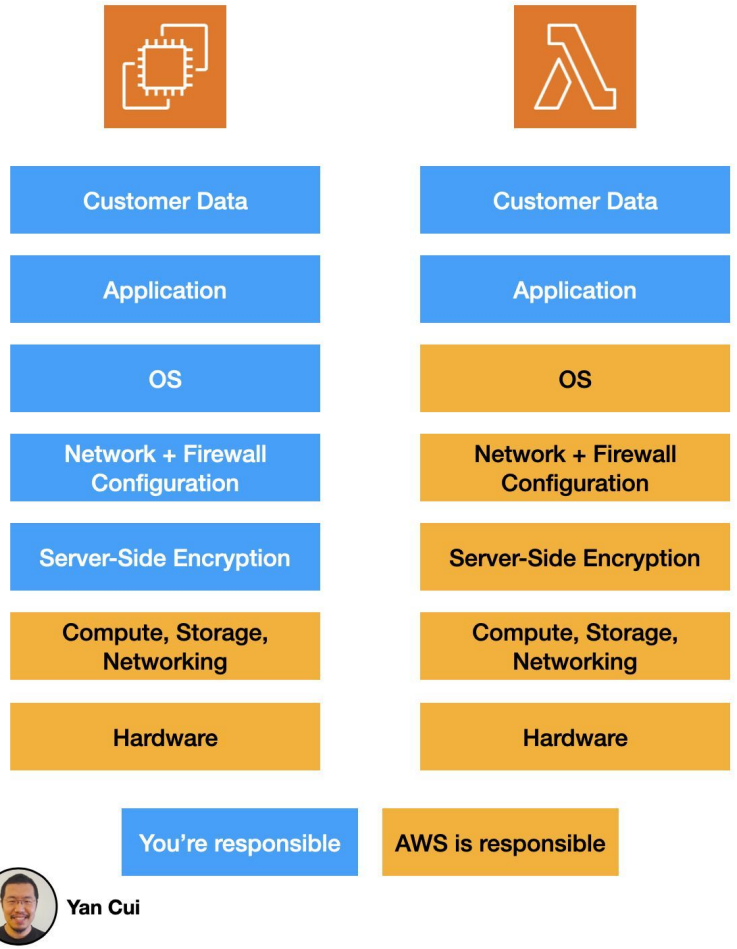
Managed by AWS

- management of servers
- management of runtimes
- scaling
- load balancing
- request routing
- patching
- …

Pricing model

- pay per request
- pay per usage
- pay per duration
- scaling to 0
- on demand pricing

“I don’t want to manage my own cluster - that’s what AWS is for.”



Why you should try serverless?

- learning new approaches and architectures
- leaning towards stateless architectures (cattle vs pets)
- very cheap for small workloads (personal projects)
- easily reproducible infrastructure with Infrastructure as a Code (CDK)
- highly available and scalable from day 1

✅ Works the same on small and big projects.

Why SvelteKit?

- modern TypeScript framework (fit for TypeScript CDK)
- outperforms React (bundle size etc)
- server-side rendering (SSR), client-side rendering (CSR), pre-rendering
- out-of-box router
- uses the Web platform (eg. Fetch API)
- easy local development (connect to public DynamoDB or Local DynamoDB)

✅ CDK construct

`kit-on-lambda`

Be honest with yourself

What is the actual scale of the project?

Are you going to deploy internationally or regional?

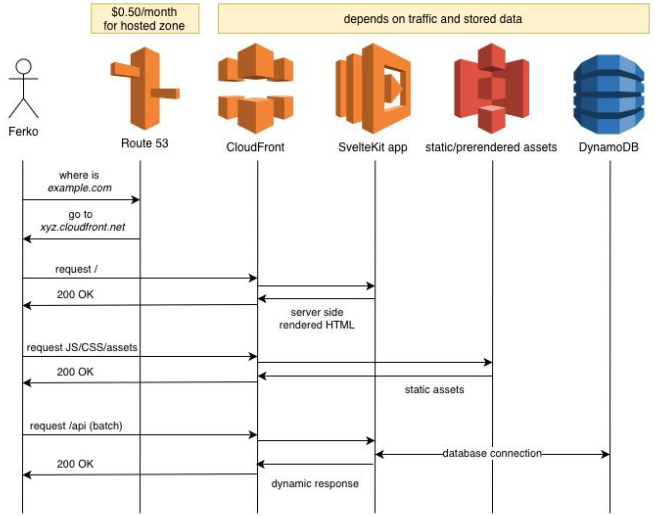
How many users are you going to serve?

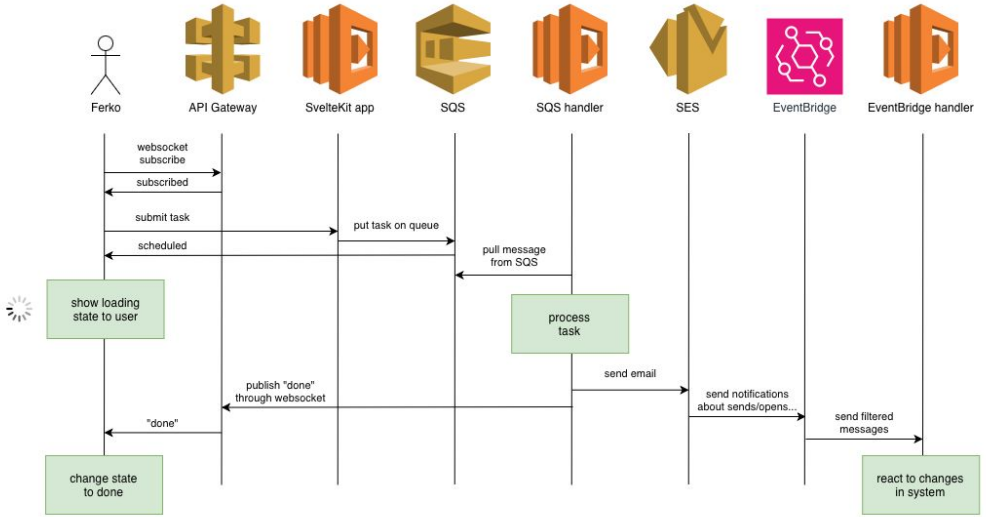
Do you really need relational database?

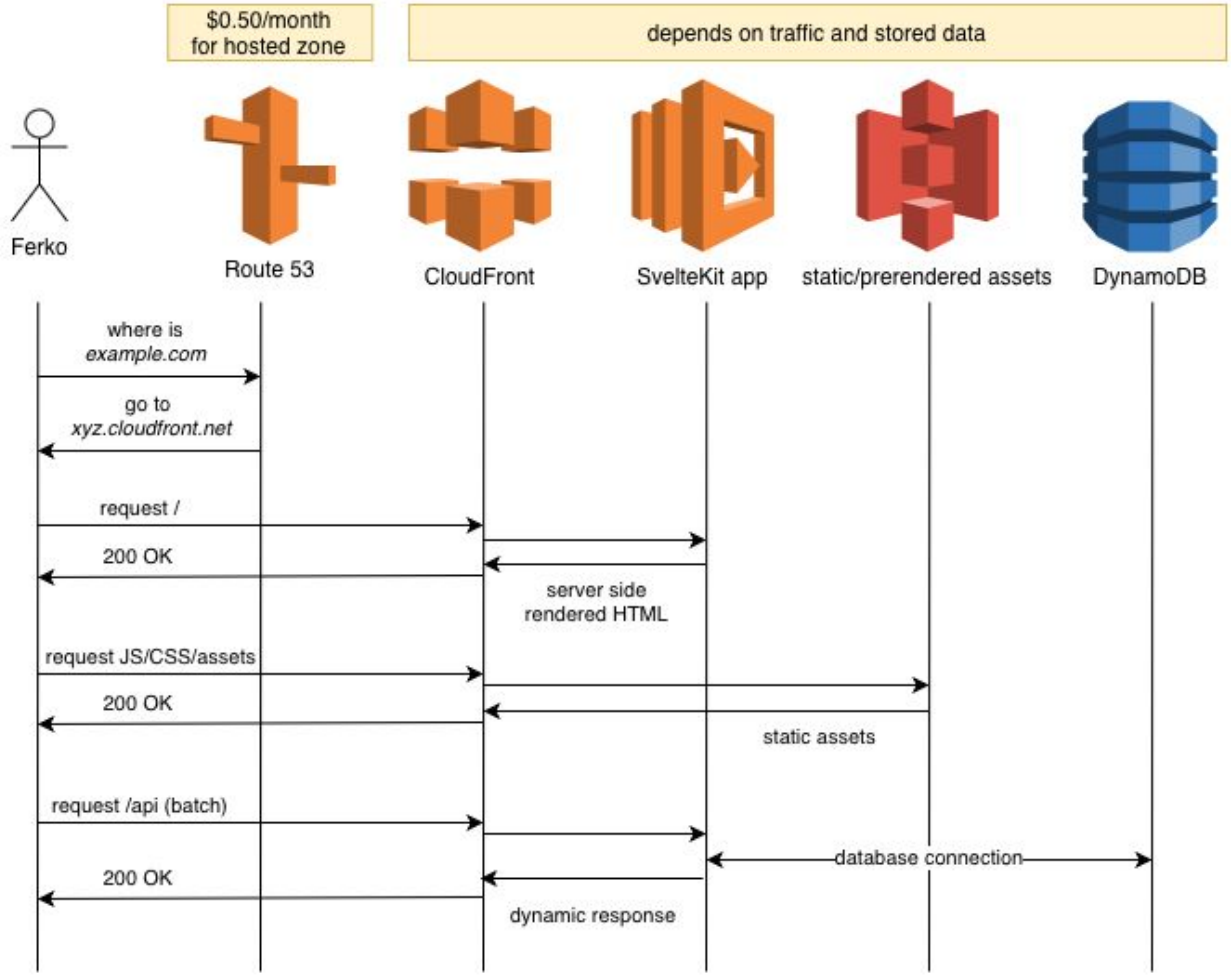
Are you building for the purpose or out of habit?

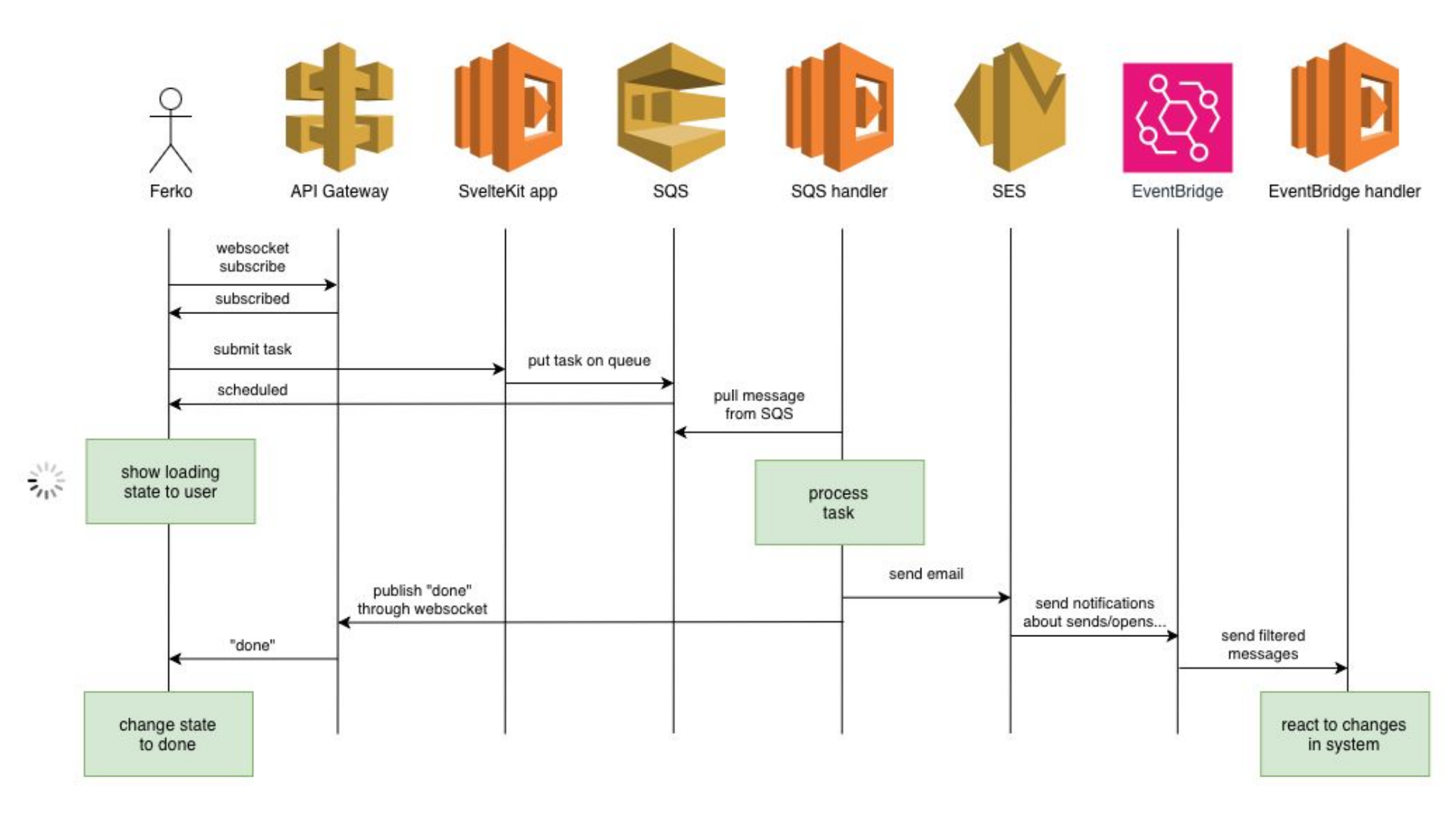
Are you the only contributor?

Web application - serverless solution






$0.50/month
for hosted zone
depends on traffic and stored data
Ferko
Route 53
CloudFront
SvelteKit app
static/prerendered assets
DynamoDB
where is
example.com
go to
xyz.cloudfront.net
request /
200 OK
server side
rendered HTML
request JS/CSS/assets
200 OK
static assets
request /api (batch)
database connection
200 OK
dynamic response


Ferko
API Gateway
SvelteKit app
SQS
SQS handler
SES
EventBridge
EventBridge handler
websocket subscribe
subscribed
submit task
put task on queue
scheduled
pull message from SQS
show loading state to user
process task
send email
publish "done" through websocket
send notifications about sends/opens...
"done"
send filtered messages
change state to done
react to changes in system

AWS Lambda

- FaaS (Function as a Service)
- various runtimes (official and unofficial)
- Docker lambda, Lambda Web Adapter
- **128MB - 10 240MB RAM, 1 - 6 vCPU**
- **15 minutes timeout**
- rapid auto scaling
- fully managed
- Event Source Mapping (ESM)

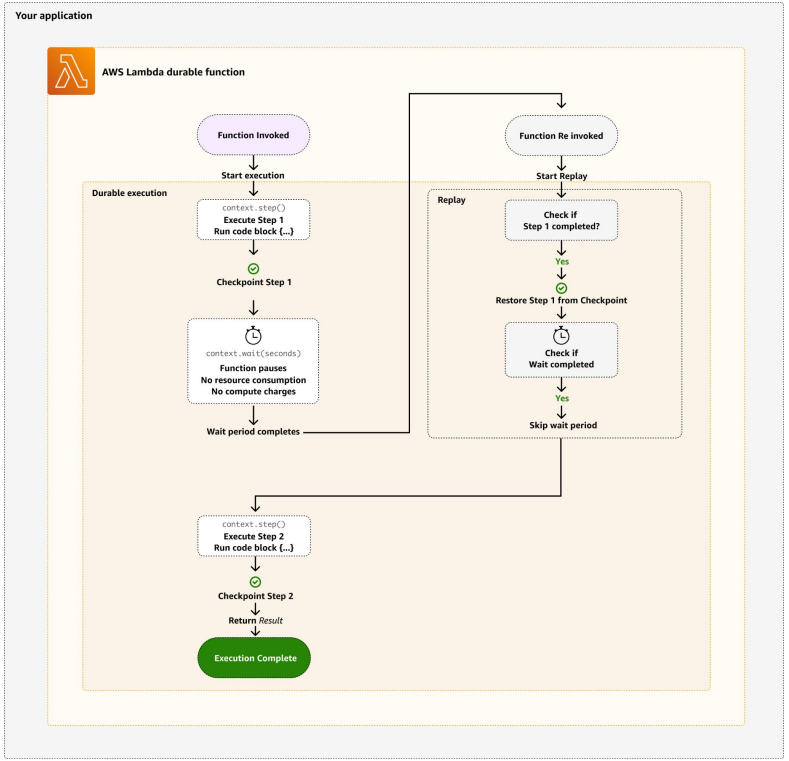
When Lambda hits limit

✅ use Lambda Managed Instances (LMI)

- same model as Lambda (no need to change mindset)
- leverage Saving plans
- great for steady traffic/processing
- choose hardware, let AWS manage it

✅ use Lambda Durable Functions

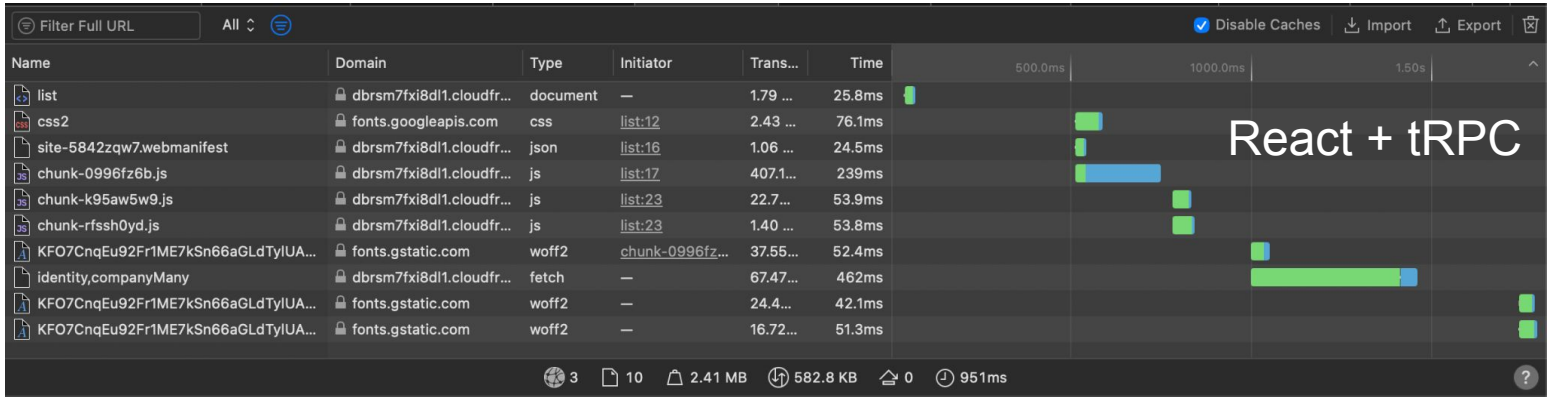
- similar to Step-Functions
- long running tasks with checkpoints

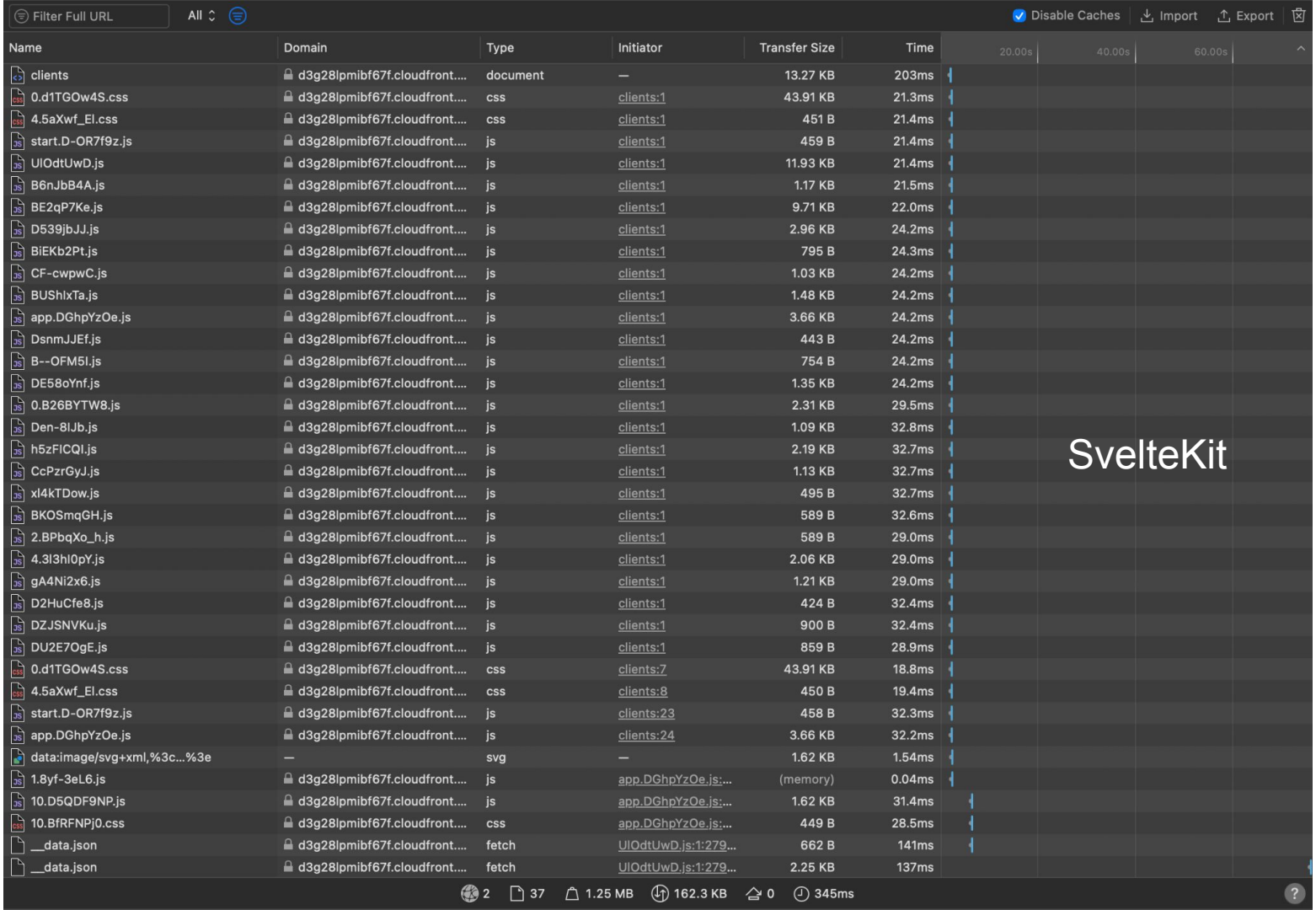


Practical tips for you

Move static assets to CDN

- no compute needed - static means “already computed”
- your “server” traffic handles just dynamic requests
- assets are cached and closer to your clients
- low latency
- very cheap




Filter Full URL
All
Disable Caches
Import
Export
Name
Domain
Type
Initiator
Transfer Size
Time
20.00s
40.00s
60.00s
clients d3g28lpmibf67f.cloudfront.... document — 13.27 KB 203ms
0.d1TGOw4S.css d3g28lpmibf67f.cloudfront.... css clients:1 43.91 KB 21.3ms
4.5aXwf_El.css d3g28lpmibf67f.cloudfront.... css clients:1 451 B 21.4ms
start.D-OR7f9z.js d3g28lpmibf67f.cloudfront.... js clients:1 459 B 21.4ms
UlOdtUwD.js d3g28lpmibf67f.cloudfront.... js clients:1 11.93 KB 21.4ms
B6nJbB4A.js d3g28lpmibf67f.cloudfront.... js clients:1 1.17 KB 21.5ms
BE2qP7Ke.js d3g28lpmibf67f.cloudfront.... js clients:1 9.71 KB 22.0ms
D539jbJJ.js d3g28lpmibf67f.cloudfront.... js clients:1 2.96 KB 24.2ms
BiEKb2Pt.js d3g28lpmibf67f.cloudfront.... js clients:1 795 B 24.3ms
CF-cwpwC.js d3g28lpmibf67f.cloudfront.... js clients:1 1.03 KB 24.2ms
BUShlxTa.js d3g28lpmibf67f.cloudfront.... js clients:1 1.48 KB 24.2ms
app.DGhpYzOe.js d3g28lpmibf67f.cloudfront.... js clients:1 3.66 KB 24.2ms
DsnmJJEf.js d3g28lpmibf67f.cloudfront.... js clients:1 443 B 24.2ms
B--OFM5I.js d3g28lpmibf67f.cloudfront.... js clients:1 754 B 24.2ms
DE58oYnf.js d3g28lpmibf67f.cloudfront.... js clients:1 1.35 KB 24.2ms
0.B26BYTW8.js d3g28lpmibf67f.cloudfront.... js clients:1 2.31 KB 29.5ms
Den-8IJb.js d3g28lpmibf67f.cloudfront.... js clients:1 1.09 KB 32.8ms
h5zFICQI.js d3g28lpmibf67f.cloudfront.... js clients:1 2.19 KB 32.7ms
CcPzrGyJ.js d3g28lpmibf67f.cloudfront.... js clients:1 1.13 KB 32.7ms
xl4kTDow.js d3g28lpmibf67f.cloudfront.... js clients:1 495 B 32.7ms
BKOSmqGH.js d3g28lpmibf67f.cloudfront.... js clients:1 589 B 32.6ms
2.BPbqXo_h.js d3g28lpmibf67f.cloudfront.... js clients:1 589 B 29.0ms
4.3l3hl0pY.js d3g28lpmibf67f.cloudfront.... js clients:1 2.06 KB 29.0ms
gA4Ni2x6.js d3g28lpmibf67f.cloudfront.... js clients:1 1.21 KB 29.0ms
D2HuCfe8.js d3g28lpmibf67f.cloudfront.... js clients:1 424 B 32.4ms
DZJSNVKu.js d3g28lpmibf67f.cloudfront.... js clients:1 900 B 32.4ms
DU2E7OgE.js d3g28lpmibf67f.cloudfront.... js clients:1 859 B 28.9ms
0.d1TGOw4S.css d3g28lpmibf67f.cloudfront.... css clients:7 43.91 KB 18.8ms
4.5aXwf_El.css d3g28lpmibf67f.cloudfront.... css clients:8 450 B 19.4ms
start.D-OR7f9z.js d3g28lpmibf67f.cloudfront.... js clients:23 458 B 32.3ms
app.DGhpYzOe.js d3g28lpmibf67f.cloudfront.... js clients:24 3.66 KB 32.2ms
data:image/svg+xml,%3c...%3e — svg — 1.62 KB 1.54ms
1.8yf-3eL6.js d3g28lpmibf67f.cloudfront.... js app.DGhpYzOe.js:... (memory) 0.04ms
10.D5QDF9NP.js d3g28lpmibf67f.cloudfront.... js app.DGhpYzOe.js:... 1.62 KB 31.4ms
10.BfRFNPj0.css d3g28lpmibf67f.cloudfront.... css app.DGhpYzOe.js:... 449 B 28.5ms
__data.json d3g28lpmibf67f.cloudfront.... fetch UlOdtUwD.js:1:279... 662 B 141ms
__data.json d3g28lpmibf67f.cloudfront.... fetch UlOdtUwD.js:1:279... 2.25 KB 137ms
2 37 1.25 MB 162.3 KB 0 345ms
SvelteKit

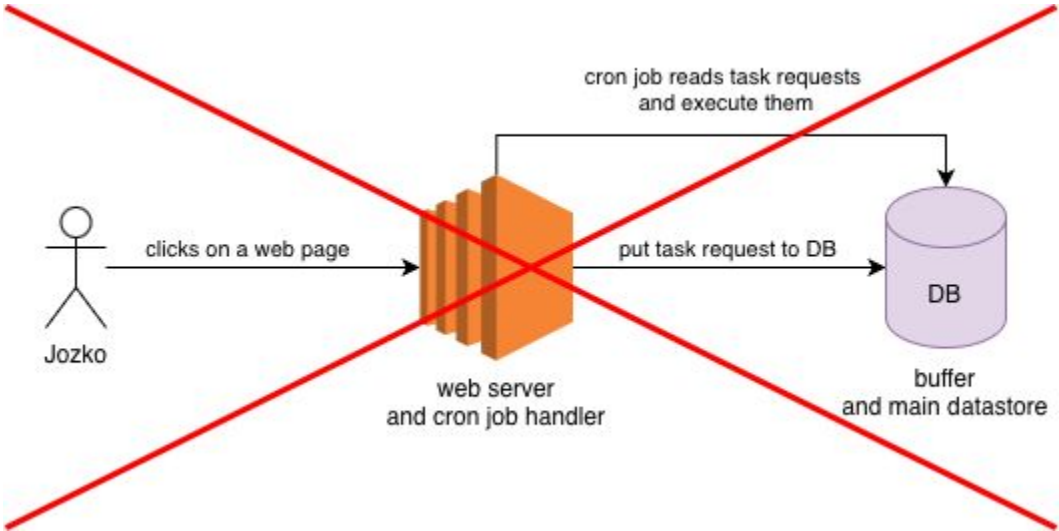
Respond quickly

- respond within 1s (API and web)
- process tasks asynchronously
- give user feedback (pending states, optimistic responses)
- update client state via WebSockets (IoT Core or API Gateway)

✅ Use systems you build in order to find out if they are usable.

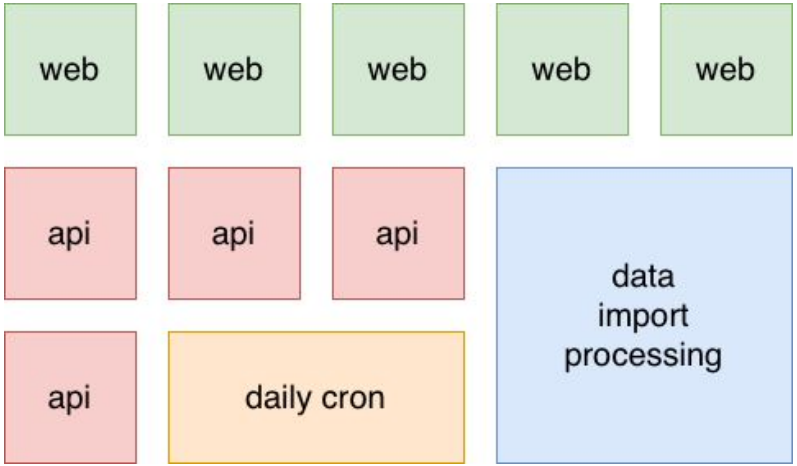
Use queues for asynchronous tasks

- decouple processing from web traffic
- buffer your requests with SQS, protect downstream from spikes
- dead-letter queue (DLQ) for failed messages
- use `reservedConcurrentExecutions` to limit SQS pollers number
- out-of-box retrying



Split your application for independent scaling

- not every task needs same CPU/memory ratio
- not every task needs same timeout
- leverage multiple DynamoDB tables



Automate with Infrastructure as a Code (IaC)

- all infrastructure should be easily replicable (multiple environments)
- use CDK for staying in your domain (TypeScript)
- you can automate other resources (custom resources)

✅ Ideally use one language eg. TypeScript for IaC and full stack application.

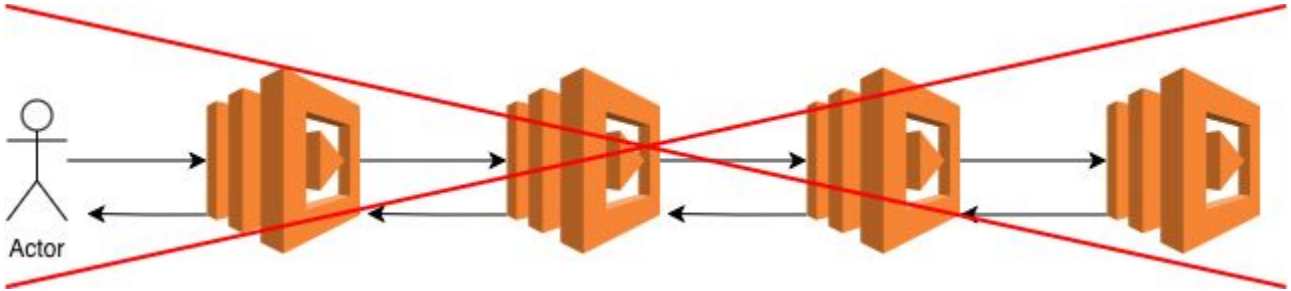
Keep your code bundle small

- size of bundle affect cold starts
- do not bundle dead code
- AWS SDK could be pre-bundled with runtime
- pre-render any templates (eg. `react-email`)

✅ Single responsibility principle keeps your bundle small.

Limit Lambda call chain

- do not break your code to nano-services
- group functions within single Lambda
- each Lambda in chain adds latency (especially on cold start)
- use Lambda Durable Functions for complex architectures
- use asynchronous tasks



Set up logging properly

- use native JSON logging
- set up log retentions
- use CloudWatch Logs insights

Sample queries Learn more

- **Common queries**
- **Lambda**
  - **View latency statistics for 5-minute intervals.**
  - **Determine the amount of overprovisioned memory.**
  - **Find the most expensive requests.**

Set up retention policies

- set up CloudWatch retention and log level
- identify data which can be deleted after period of time
- offload archive data to cheaper storage (S3/Glacier)
- use S3 intelligent tiering for your assets
- use DynamoDB infrequent access table class

✅ Do not store logs/data indefinitely.

Leverage Lambda Function URL

- not every microservice needs API Gateway
- when using API Gateway, start with cheaper HTTP API Gateway
- put Lambda behind CloudFront

✅ instead of OAC (origin access control) share custom header with CloudFront

Important

If you use `PUT` or `POST` methods with your Lambda function URL, your users must compute the SHA256 of the body and include the payload hash value of the request body in the `x-amz-content-sha256` header when sending the request to CloudFront. Lambda doesn't support unsigned payloads.

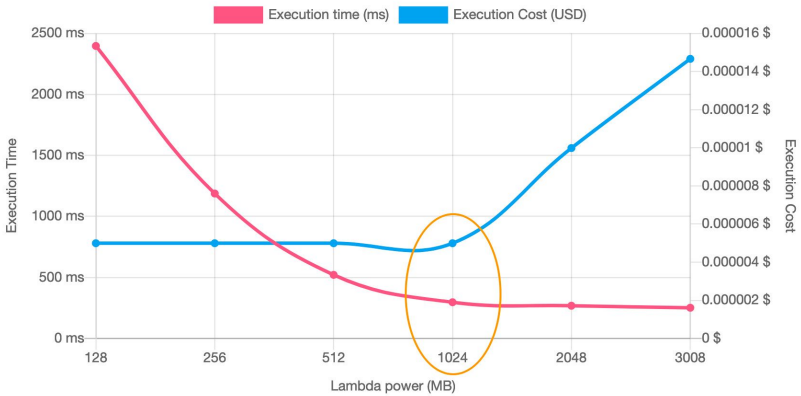
VPC - Virtual Private Cloud

- use Egress-only Internet Gateway (only IPv6) instead of NAT Gateway
- if you need NAT Gateway you can try cheaper NAT instance
- use free DynamoDB and S3 VPC Gateway endpoints

✅ avoid VPC for pet projects

“Right-size” your Lambda functions

- use AWS Lambda Power Tuning tool to find best size of Lambda
- consider using Node.js `worker_threads` in multi-vCPU Lambda functions
- for better network performance use larger Lambda



Lambda vCPU vs memory allocations

Configured Memory (MB)	Allocated cores
128-1769	1
1770-3538	2
3539-5307	3
5308-7076	4
7077-8845	5
8846-10240	6

Source: AWS re:Invent 2020: What's new in serverless

Don’t forget to `await` asynchronous functions

- starting from Node.js 24 callback-based function handlers are not supported
- meaning `callbackWaitsForEmptyEventLoop` cannot be set

✅ set up require-await ESLint rule

This is what happens when you forget to `await`

```
export const handler = async () => {
  console.log(1);
  logAsync(2); // missing await 🥲
  console.log(3);
};

async function logAsync(...data: any[]) {
  return new Promise((resolve) => {
    setTimeout(() => {
      console.log(...data);
      resolve(undefined);
    });
  });
}
```

Microservices

- split your application into multiple microservices using separate Lambda functions and DynamoDB tables
- use single Lambda function for whole API router (do not split CRUD operations)
- start with Lambda, optimize afterwards (ask Tomas Sabol)

TOMÁŠ SABOL
CLOUD SOLUTIONS ARCHITECT
@LENUS

Try DynamoDB

- manage your schema in application layer (with `valibot`)
- add schema version to each item for easier management
- filter/sort data in memory (for small datasets)
- implement counters/aggregations with transactions/DynamoDB streams
- combine with OpenSearch/Athena when needed

Cost-Aware Modeling in DynamoDB: Designing for Performance & Spend

by Alex DeBrie

Try Aurora DSQL

- distributed PostgreSQL-compatible database
- highly available and scalable
- fully serverless
- optimistic concurrency control

✅ It’s not 1:1 replacement for Postgres or MySQL.

Not every problem has the same solution

- architect for current situation not for next 10
- don’t overthink it, keep it simple
- you are building for your customers
- when building for pleasure try new things

✅ Don’t be afraid to break things!

Ivan Barlog

AWS Solutions Architect

Github ivanbarlog
beesolve

Email ivan@barlog.sk

Web barlog.sk
beesolve.com

LinkedIn ivan-barlog

Go build something!

Please rate the sessions 😊