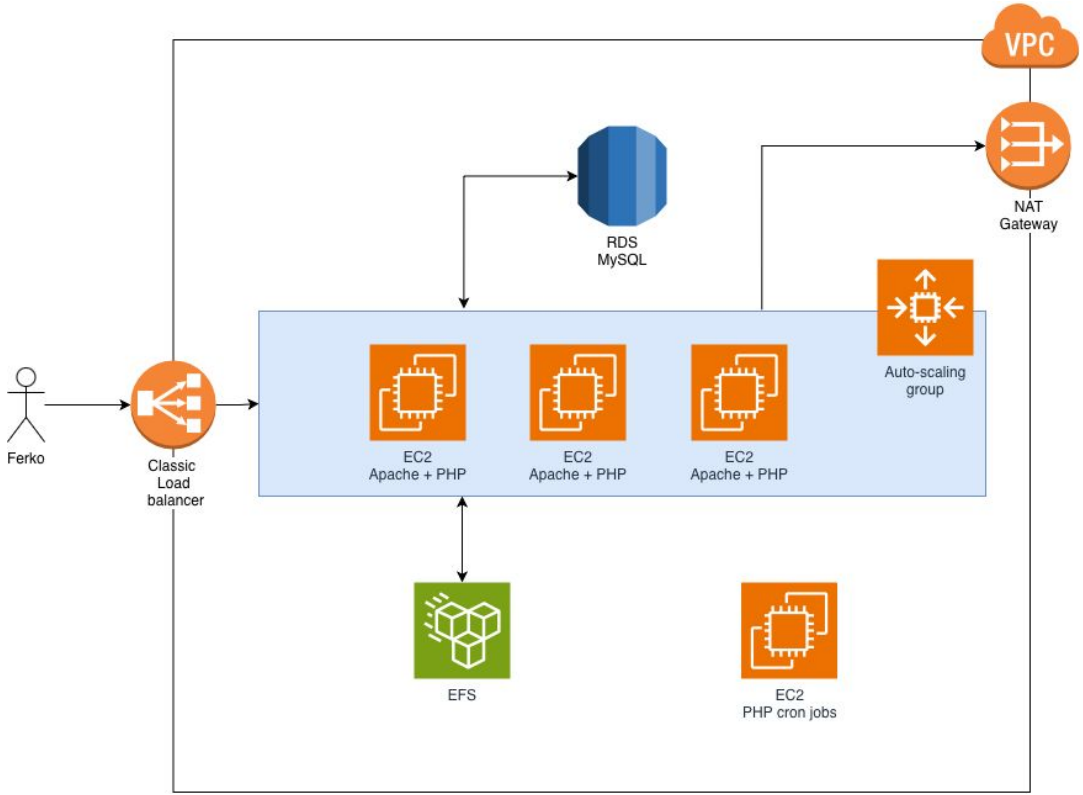
You probably shouldn’t use Kubernetes

AWS Serverless Architecture

👋 Show of hands

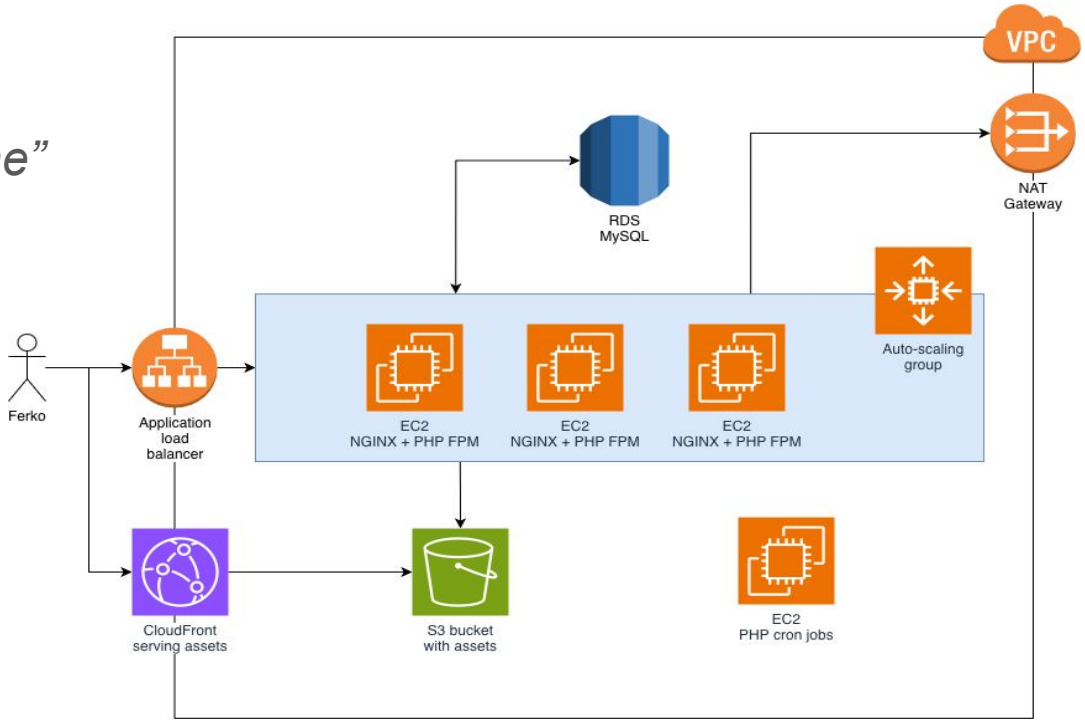
PHP web development ~2014

- no Docker
- Symfony PHP framework
- Amazon Linux
- manual patches
- ssh + git deployment
- scaling ~10-20 minutes



Dockerization ~2016

- still PHP
- easier maintenance
- no more *“Works on my machine”*
- `docker-compose` abstraction
- bash wrapper abstraction
- developers had no clue
- still managing servers
- scaling ~10-20 minutes



Intermezzo

- **we hated JavaScript**

  we've created AjaxCom (similar to HTMX)
- **we loved PHP**

  (Symfony framework, OOP, hexagonal, DDD, clean code, PSR)

Pressure from the client to use **React** ~ JavaScript

Fork on the road ~2018

- AWS Elastic Container Service (ECS) ~2014
- Kubernetes ~2014
- AWS Lambda ~2014

- React ~2013
- AjaxCom ~2014 😂

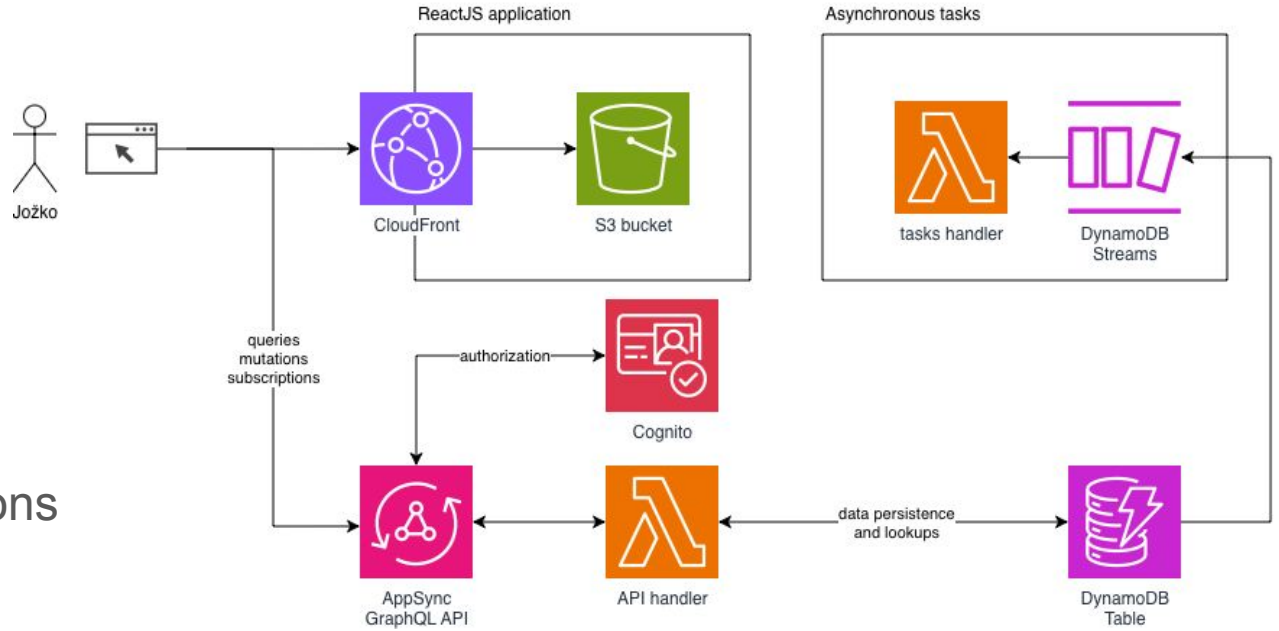
We choose Serverless

- ~~JavaScript~~ **TypeScript** on frontend and backend
- No more frameworks
- Everything fit within one function

  (not 10s of files for simple feature - PSR)
- Serverless Framework ~2015
- AWS Cloud Development Kit (CDK) GA ~2019

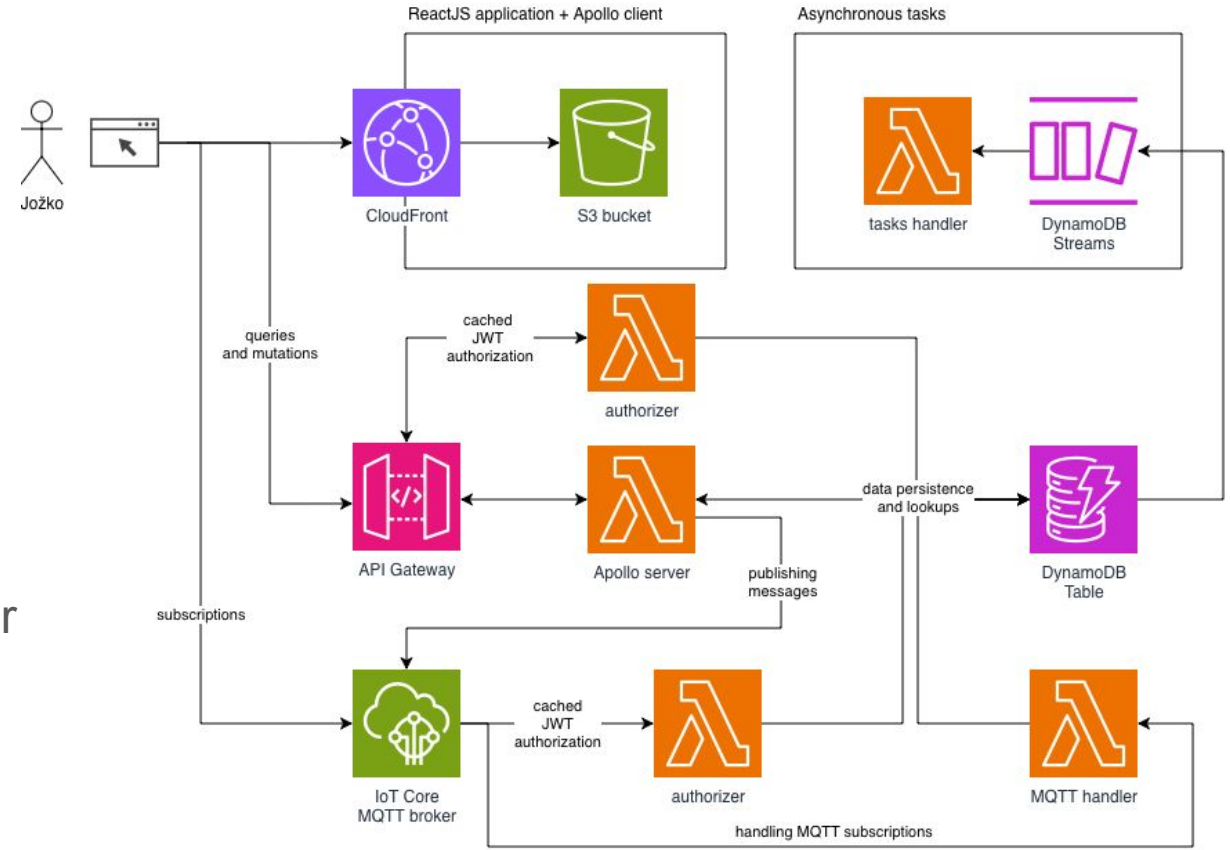
Our first take on Serverless ~2018

- NodeJS lambda
- single table design
- AppSync subscriptions



Final architecture of auction system ~2018

- Apollo GraphQL
- AWS IoT Core MQTT
- custom JWT authorizer



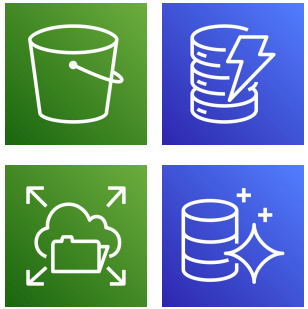
Serverless on AWS after ~7 years

Compute

- Lambda
- Fargate

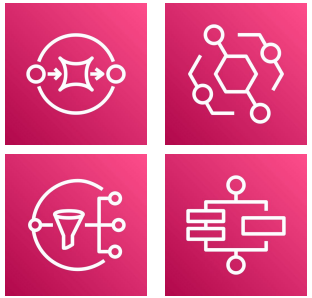
Storage

- S3
- EFS
- DynamoDB
- Aurora



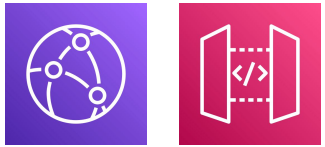
Events

- SQS
- EventBridge
- SNS
- Step Functions



Distribution

- CloudFront
- API Gateway



and others…

Key takeaways

- Every tool has its purpose
- You are shipping software not tools
- Don’t fight the inevitable
- Simple things are hard
- Kubernetes is not always the answer
- Serverless is not always the answer
- Learn about your options
- Let someone else manage your “cluster”

Ivan Barlog

AWS Solutions Architect

BeeSolve

Github ivanbarlog

Email ivan@barlog.sk

Web barlog.sk
beesolve.com

LinkedIn ivan-barlog

Go build something!