

FaaS

Eirik, Orestis, Rurik

Institute for Economic Affairs — ML-faktion

2019

What will we talk about?

- ▶ What do we mean by FaaS?
- ▶ Benefits of FaaS
- ▶ Challenges of FaaS
- ▶ Understanding cost and cost implications
- ▶ Concrete examples
 - ▶ REST API
 - ▶ CD-platform
- ▶ Summary

What do we mean by FaaS?

Simply put: AWS Lambda and managed services like DynamoDB

Benefits of FaaS

- ▶ Lower costs
- ▶ Simplicity
- ▶ More testable architecture
- ▶ Replicability
- ▶ Driftlessness

Challenges of FaaS I

Challenges:

- ▶ Vendor lock-in
- ▶ New idiom, a lot to learn
- ▶ Not suited to every task
- ▶ Accidental costs can accumulate
- ▶ Possible to make a real mess

Challenges of FaaS II

How to deal with them:

- ▶ Portable code
 - ▶ but don't strive for project-defeating generics
- ▶ Understand the offerings, learn about Infrastructure As Code (IAC)
- ▶ Many things can be solved with Lambdas. (But should they?)
- ▶ Keep an eye on what costs are accumulating by setting up cost reports, automate resource deprovisioning
- ▶ Discipline is important:
 - ▶ TDD & BDD/FDD are essential
 - ▶ Code hygiene and QA equally so
 - ▶ having continuous deployment as a constant and understood aim
 - ▶ IAC is an absolute

Understanding costs & cost implications

Cheap:

- ▶ Some stuff costs a lot
- ▶ Some stuff costs little, if it is used correctly

Cheap isn't always good

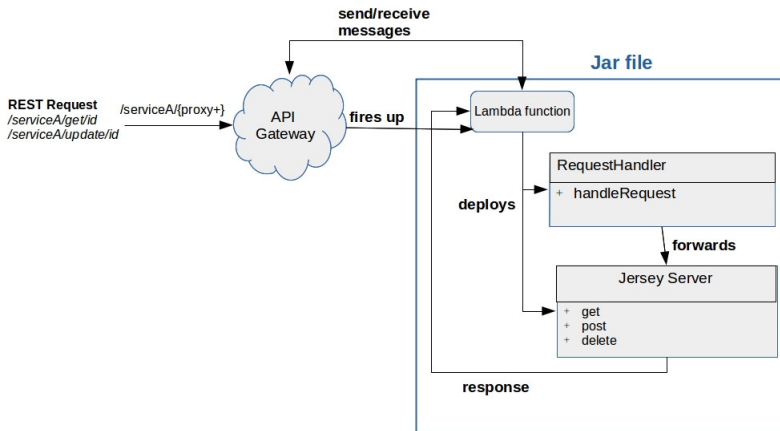
- ▶ Some stuff is cheaper to implement in different ways
 - ▶ Cost for light usage for some services is high (CloudSearch)
- ▶ Some stuff is more performant if implemented in different ways
 - ▶ Long-running and/or processor-intensive processes are better run in EC2 as Lambda becomes expensive in these cases

Basically, know your needs; know the platform; use the appropriate technology.

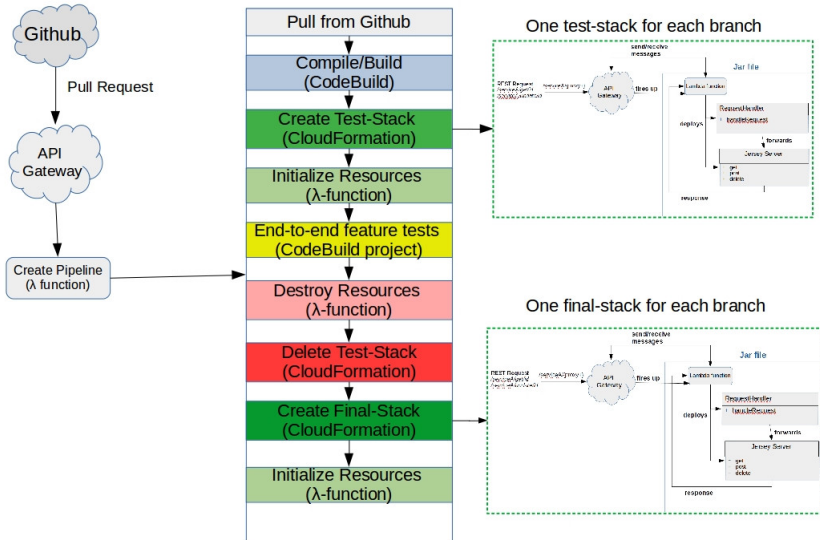
Concrete example: REST API (A Serverless Server)

- ▶ A Jersey server as a function.
- ▶ Advantages
 - ▶ The server starts with very small delay (milliseconds)
 - ▶ Server dies after few minutes without requests
 - ▶ Cost for low/moderate traffic: low
 - ▶ Scales automatically
 - ▶ No maintenance
 - ▶ The Jersey server can be ported into an EC2 instance with very few code modifications
 - ▶ Extremely easy to replicate the production environment
- ▶ Disadvantages
 - ▶ Cost for high traffic: unknown
 - ▶ Can have problems with timeouts for very heavy queries.

Concrete example: CD-platform



Concrete example: CD-platform



Summary

- ▶ Infrastructure as code is alpha & omega
- ▶ Automated deployment of services ("Driftløs")
- ▶ Choose the right DevOps approach and develop it
 - ▶ Test-first, test-last and test everything inbetween
 - ▶ CD
 - ▶ Stamp out laziness
- ▶ Read the DORA report for inspiration