

## Server?

**Wo wir hingehen brauchen wir keine Server...**

Serverless Programmieren mit AWS Lambda



Werner Eberling (@Wer\_Eb)

werner.eberling@mathema.de

www.mathema.de

# Der Sprecher



## Werner Eberling

Principal Consultant / Autor

Email: [werner.eberling@mathema.de](mailto:werner.eberling@mathema.de)

Twitter: [@Wer\\_Eb](https://twitter.com/Wer_Eb)



Verfluchte Sie  
Ein Blick auf den JBoss AS 7  
VON WERNER EBERLING



### Effiziente Architekturen mit Java

- Mobile Softwareentwicklung
- Spring als Basis für flexible Softwarearchitekturen
- Ein Vergleich von XML, JSON und Binärformaten

Ein Leitfaden mit Java für Hamburgs Wähe  
Spezialrezepte in Produktionsumgebungen

mit Java über Group Technology



### Enterprise JavaBeans 3.1

DAS EJB3-PRAKTIKUM  
FÜR EIN- UND UMSCHWINGER

### ENTERPRISE JavaBeans 3.1

DAS EJB-PRAKTIKUM  
FÜR EIN- UND UMSCHWINGER

2. Auflage

HANSER

# Serverless - Was man so hört...

Don't pay for idle!!!

Cloud native Entwicklung

No Ops

# Wirklich keine Server mehr?



# Keine Gedanken mehr über Infrastruktur

- Backend as a Service (BaaS)
  - Individuelle Frontends nutzen bestehende (cloud-basierte) Dienste von Drittanbietern
  - Rich Clients halten die spezielle Business Logik

# Keine Gedanken mehr über Infrastruktur

- Backend as a Service (BaaS)
  - Individuelle Frontends nutzen bestehende (cloud-basierte) Dienste von Drittanbietern
  - Rich Clients halten die spezielle Business Logik
  
- Function as a Service (FaaS)
  - Zustandslose, flüchtige Laufzeitcontainer
  - Bereitgestellt durch Drittanbieter (Cloud-basiert)
  - „Eigene Funktionen als BaaS“

# Keine Gedanken mehr über Infrastruktur

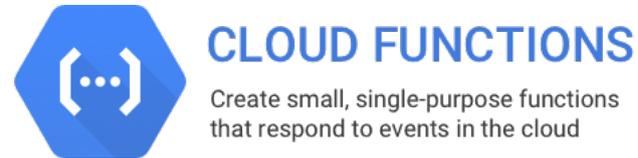
- Backend as a Service (BaaS)
  - Individuelle Frontends nutzen bestehende (cloud-basierte) Dienste von Drittanbietern
  - Rich Clients halten die spezielle Business Logik
- Function as a Service (FaaS)
  - Zustandslose, flüchtige Laufzeitcontainer
  - Bereitgestellt durch Drittanbieter (Cloud-basiert)
  - „Eigene Funktionen als BaaS“

# Keine Gedanken mehr über Infrastruktur

- Konzentration auf Funktionalität statt auf Infrastruktur
  - „SoC auf neuem Level“

# Keine Gedanken mehr über Infrastruktur

- Konzentration auf Funktionalität statt auf Infrastruktur
  - „SoC auf neuem Level“



IBM Cloud Functions



Apache OpenWhisk



Amazon Lambda

# Keine Gedanken mehr über Infrastruktur

- Konzentration auf Funktionalität statt auf Infrastruktur
  - „SoC auf neuem Level“



IBM Cloud Functions



Apache OpenWhisk



Amazon Lambda

# AWS Lambda in Stichworten

- AWS Service seit November 2014
- Unterstützte Sprachen
  - Node.js, Java, C#, Python, Go, PowerShell, Ruby, eigene Runtime
- Zugriff auf AWS Services
  - S3, DynamoDB, SNS, SES, Kinesis, IAM, SSO, ...



Amazon Lambda

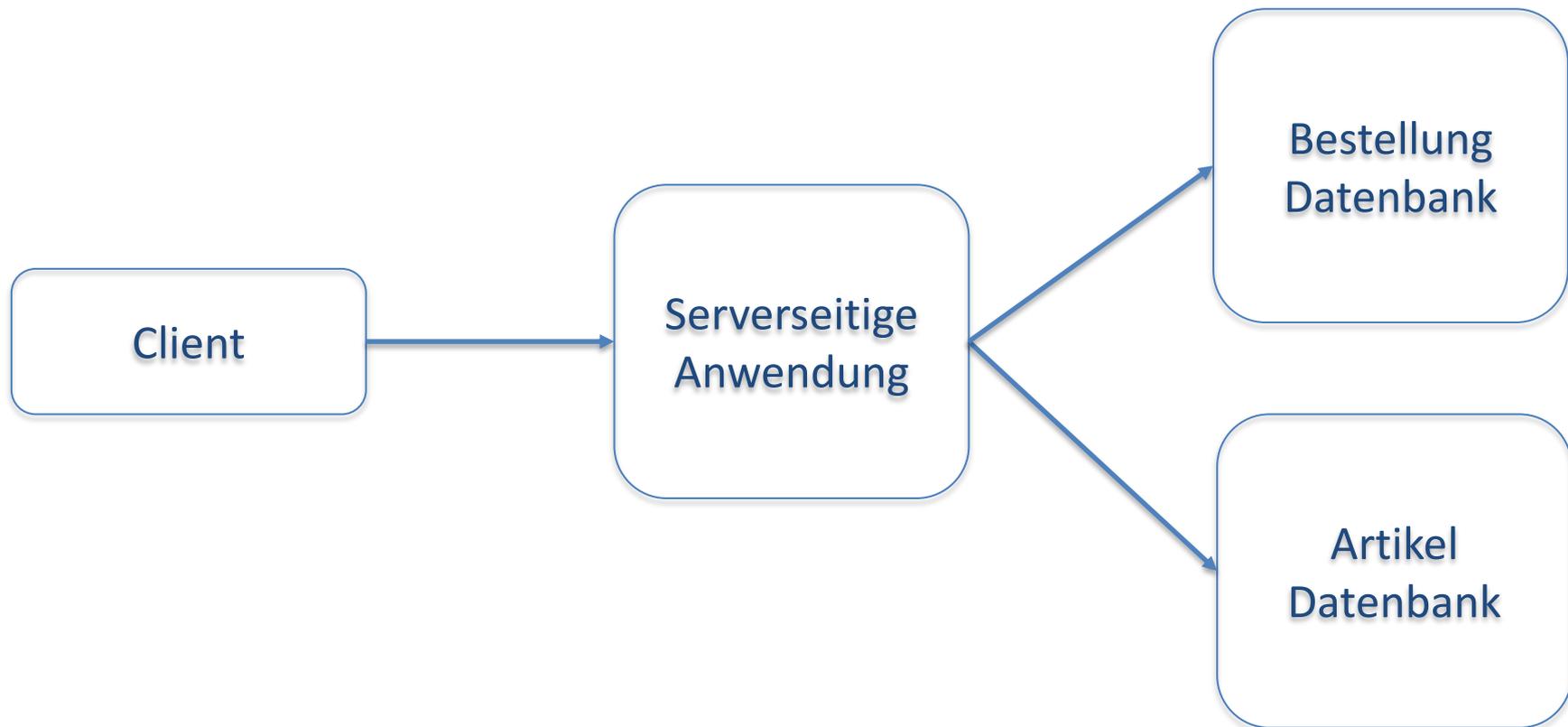
# Prinzipien einer Serverless Architektur (1)

- Funktionalität ist in unabhängige, zustandslose Funktionen mit klarer Verantwortung geteilt
- Einzelne Funktionalitäten können isoliert und „nur bei Bedarf“ genutzt werden (Execution on demand)
- Verarbeitung erfolgt ereignisbasiert in Form von Verarbeitungspipelines

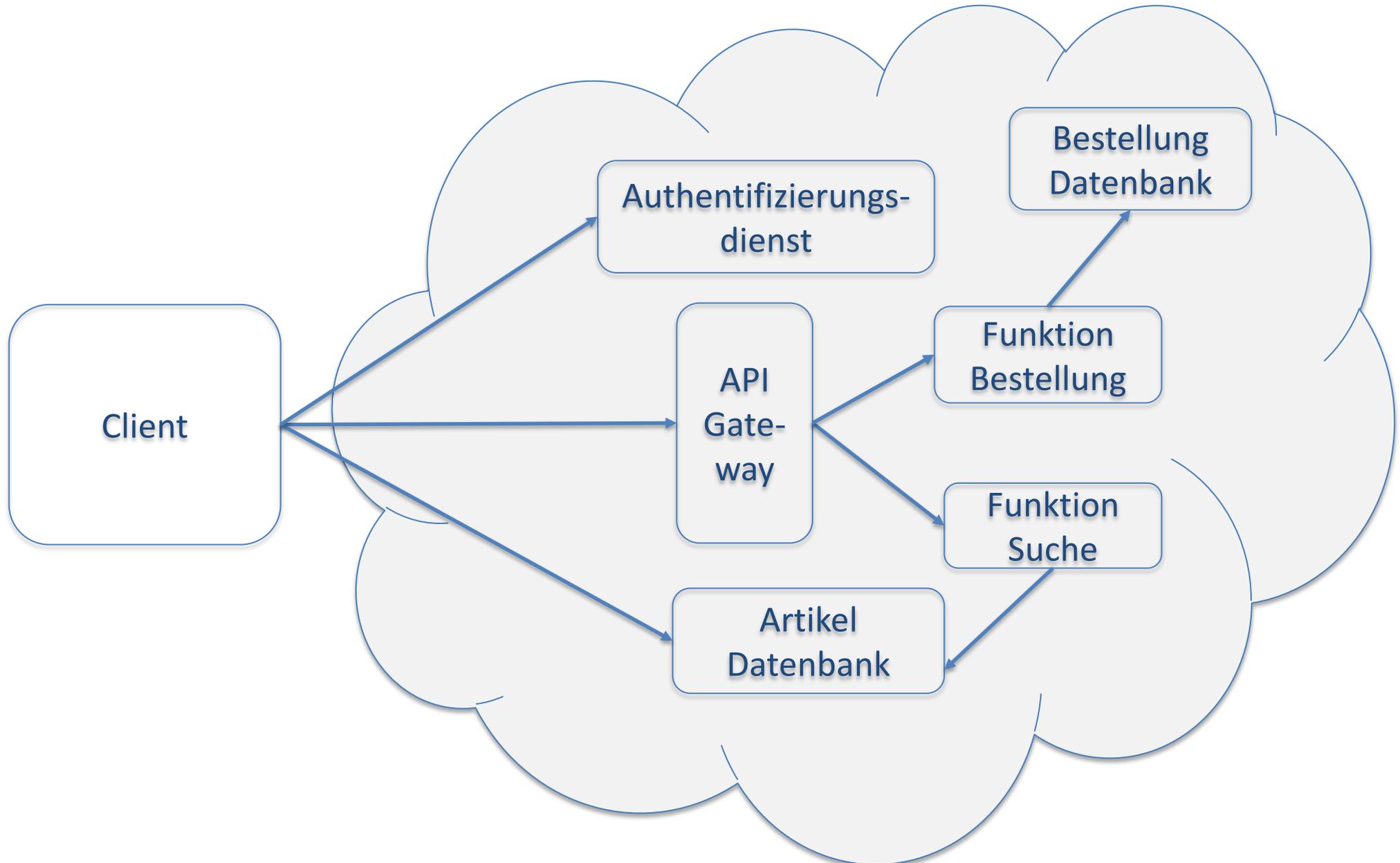
# Prinzipien einer Serverless Architektur (2)

- Einsatz von Diensten von Drittanbietern ist möglich / gewünscht
- Mehr Logik und Verantwortung im Frontend (Thick Clients)

# „Klassische“ Server Architektur



# Serverless Architektur (UI-getrieben)



# Beispiel

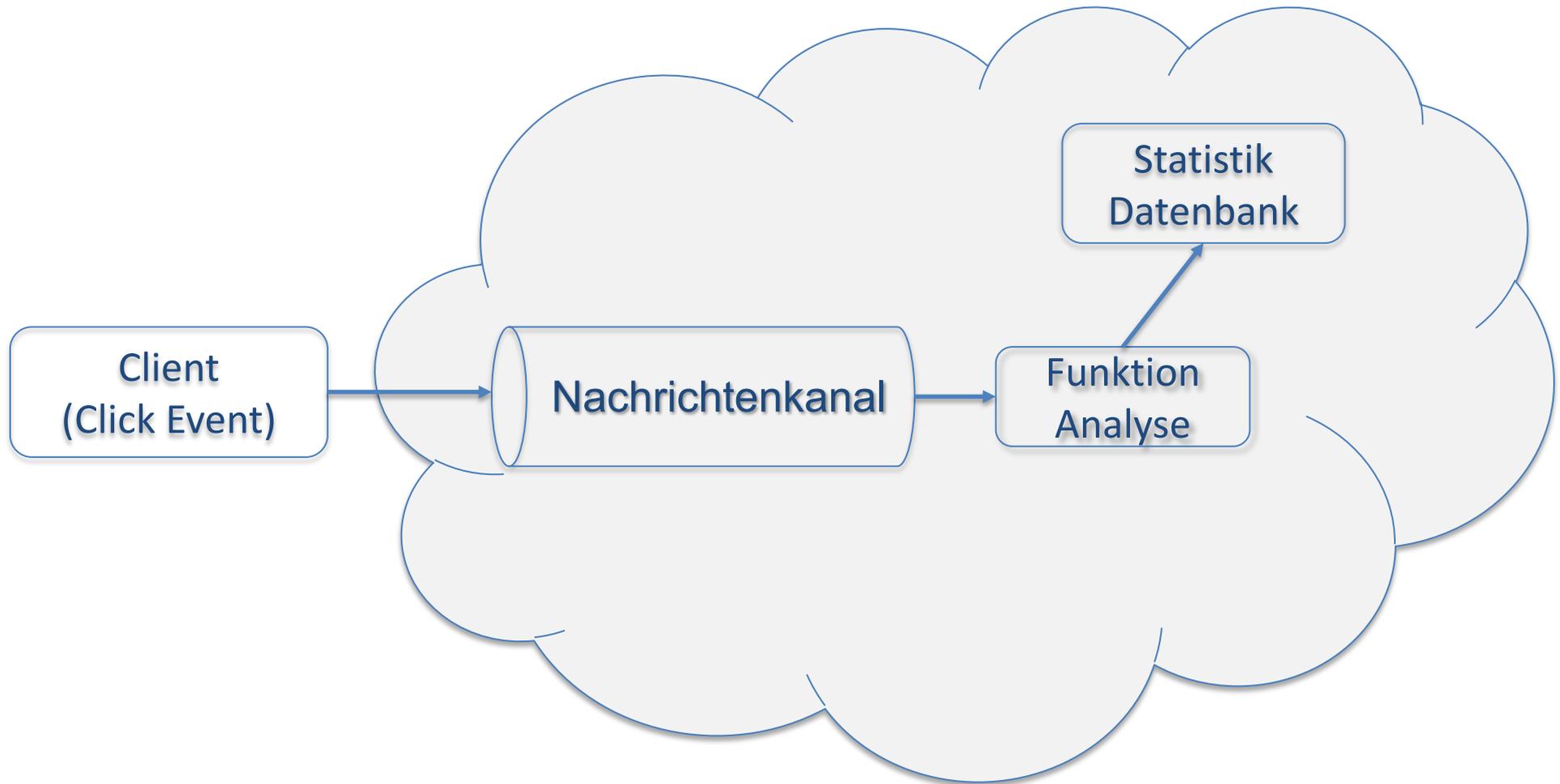
The screenshot displays the AWS API Gateway console interface. The breadcrumb navigation shows the path: **APIs** > **ToUpperLambda (ragztijtfd)** > **Resources** > **/toUpper (axutpp)** > **GET**. The main content area is titled **/toUpper - GET - Method Execution**. On the left, a sidebar lists various API Gateway components, with **Resources** selected under the **ToUpperLambda** API. The central diagram illustrates the request flow:

- Client**: Initiates the request.
- Method Request**: Contains **Auth: AWS\_IAM** and **ARN: arn:aws:execute-api:eu-central-1:072100388290:ragztijtfd/\*/\***.
- Integration Request**: Shows **Type: LAMBDA\_PROXY**.
- Integration Response**: Contains the message: "Proxy integrations cannot be configured to transform responses."
- Method Response**: Shows **HTTP Status: Proxy** and **Models: application/json => Empty**.
- Lambda toUpperCaseViaHttp**: The target Lambda function.

Arrows indicate the flow from the Client to the Method Request, then to the Integration Request, then to the Integration Response, and finally to the Method Response, which is returned to the Client. A vertical bar on the right represents the Lambda function.

At the bottom of the console, there is a footer with **Feedback**, **English (US)**, and copyright information: © 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

# Serverless Architektur (Event-getrieben)



# Beispiel

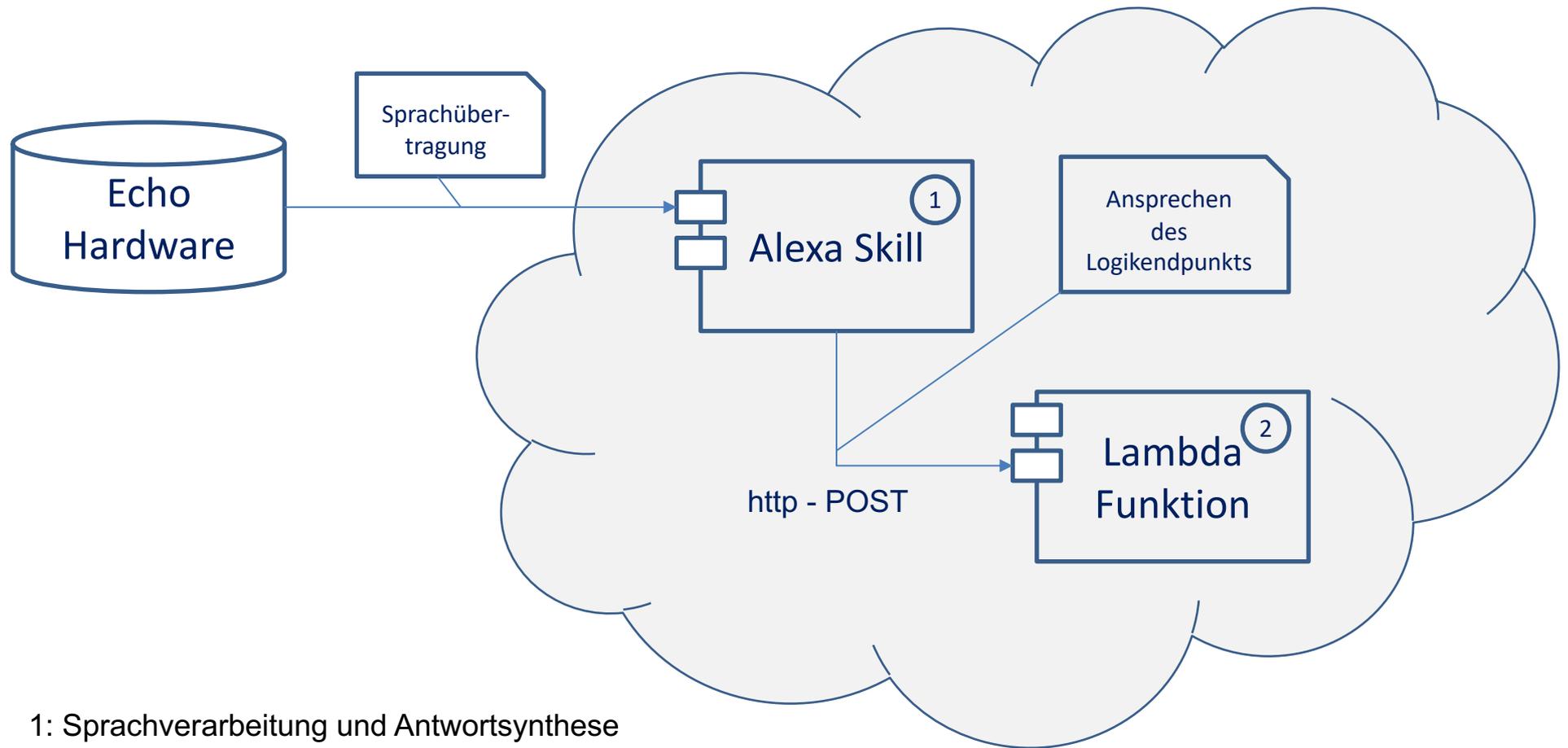
The screenshot shows the AWS Lambda console for a function named **S3Sample** in the **eu-central-1** region. The function's ARN is `arn:aws:lambda:eu-central-1:072100388290:function:S3Sample`. The console is in the **Configuration** tab, and the **Designer** view is active. In the Designer, an **S3** trigger is connected to the function. The function is also configured with **Amazon CloudWatch Logs** and **Amazon S3** resources. The **Function code** section shows the following configuration:

- Code entry type: `Edit code inline`
- Runtime: `Node.js 6.10`
- Handler: `index.handler`

The footer of the console displays the following information:

- Feedback
- English (US)
- © 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.
- Privacy Policy
- Terms of Use

# Serverless Architektur (für „neue“ Client-Szenarien)



# Beispiel

The screenshot shows the AWS Lambda console for a function named 'capitalSkillFunc' in the 'eu-west-1' region. The breadcrumb navigation is 'Lambda > Funktionen > capitalSkillFunc'. The ARN is 'arn:aws:lambda:eu-west-1:[:redacted]:function:capitalSkillFunc'. The console is in the 'Konfiguration' tab, with 'Überwachung' also visible. The 'Designer' section shows a visual representation of the function's configuration. On the left, under 'Auslöser hinzufügen', there is a list of triggers: API Gateway, AWS IoT, Alexa Skills Kit, Alexa Smart Home, Application Load Balancer, and CloudWatch Events. The 'capitalSkillFunc' function is shown in the center, with 'Layers' set to '(0)'. Below the function, 'Alexa Skills Kit' is selected as a trigger. On the right, 'Amazon CloudWatch Logs' and 'Amazon DynamoDB' are listed as resources. A dashed box indicates that these resources are those that the function role has access to. At the bottom, the 'Funktionscode' section shows 'Codeeingabetyp' set to 'Code inline bearbeiten', 'Laufzeit' set to 'Node.js 6.10', and 'Handler' set to 'index.handler'. The footer includes 'Feedback', 'Deutsch', and copyright information for Amazon Web Services, Inc. or its affiliates, along with links to 'Privacy Policy' and 'Terms of Use'.

# Vorteile einer Serverless Lösung

- Keine Server mehr
  - Schnelles Prototyping
- Geringere Kosten
  - „Don't pay for idle!“
- Weniger Code
  - Keine speziellen Bibliotheken oder Frameworks
- Skalierung und Verteilung „out of the box“

# Nachteile einer Serverless Lösung

- Ablauf in einer fremden, öffentlichen Umgebung (Public Cloud)
- Nur begrenzte Service Zusicherungen (SLAs)
- Nur begrenzte Einflussnahme auf die Laufzeitumgebung
- Vendor Lock-In
- Inhärente Verteilung (aka. Dezentralisierung)

# „The right tool for the right job“

- Serverless Ansätze sind nicht die Lösung aller Probleme!
- Serverless Ansätze sind gut für bestimmte Problemstellungen
- Hybride Lösungen nutzen die Vorteile ohne zu viele Nachteile in Kauf nehmen zu müssen
  - Bsp.: Log-Analyse, ...

# Einfach mal Ausprobieren

- Cloud Anbieter haben Angebote für
  - den Einstieg (z.B. die ersten 12 Monate)
  - Geringe Verarbeitungsvolumen (z.B. die ersten 1 Mio Aufrufe)

# Fragen?

**Vielen Dank!**

werner.eberling@mathema.de

**Twitter:** @Wer\_Eb

www.mathema.de



Bitte geben Sie uns jetzt Ihr Feedback!

Wo wir hin wollen, brauchen wir keine  
Server – Serverless Programmieren mit  
AWS Lambda  
*Werner Eberling*



### Nächste Vorträge in diesem Raum

**15:45** Vier Deployments für ein Halleluja –  
Aus dem Leben eines  
Softwaretherapeuten, *Michael Bruns*

