



# Der sprechende Kickertisch

---

Marco Buss



## Marco Buss

Senior Consultant | AWS Certified Solutions Architect - Associate

0173-727 9018

[marco.buss@opitz-consulting.com](mailto:marco.buss@opitz-consulting.com)

Tempelhofer Weg 64

12347 Berlin



[WWW.OPITZ-CONSULTING.COM](http://WWW.OPITZ-CONSULTING.COM)



[@OC\\_WIRE](https://twitter.com/OC_WIRE)



[OPITZCONSULTING](https://www.youtube.com/OPITZCONSULTING)



[opitzconsulting](https://www.linkedin.com/company/opitzconsulting)



[opitz-consulting-bcb8-1009116](https://wa.me/opitz-consulting-bcb8-1009116)

## ■ ■ ■ Überraschend mehr Möglichkeiten!

„Mit unserer Leidenschaft für neue Technologien und unserem Anspruch an herausragende Beratung sind wir bei unseren Kunden der Motor für die digitale Transformation.“

Bernhard Opitz, CEO von OPITZ CONSULTING



### Portfolio:

- Spezialist für digitale Transformation
- Software Development
- BI & Big Data
- BPM & Systemintegration
- Cloud & Infrastruktur
- Internet der Dinge
- Managed Services
- Oracle Lizenzmanagement
- Strategy & Change

### Überblick:

- Gründung 1990
- 8 Standorte in Deutschland
- 3 Standorte in Polen
- Ca. 400 Mitarbeiter
- Inhabergeführt
- Breites Themenportfolio
- Herstellerunabhängige Beratung

### Märkte:

- Branchenübergreifend
- 600 Kunden
- 2/3 aller Dax-Unternehmen
- 29% Handel/Logistik/Service
- 29% Industrie/Versorgung/Telko
- 42% Finanzsektor/Public



1

Übersicht

2

AWS IoT

3

Alexa Skill Kit

4

Alexa Voice Services

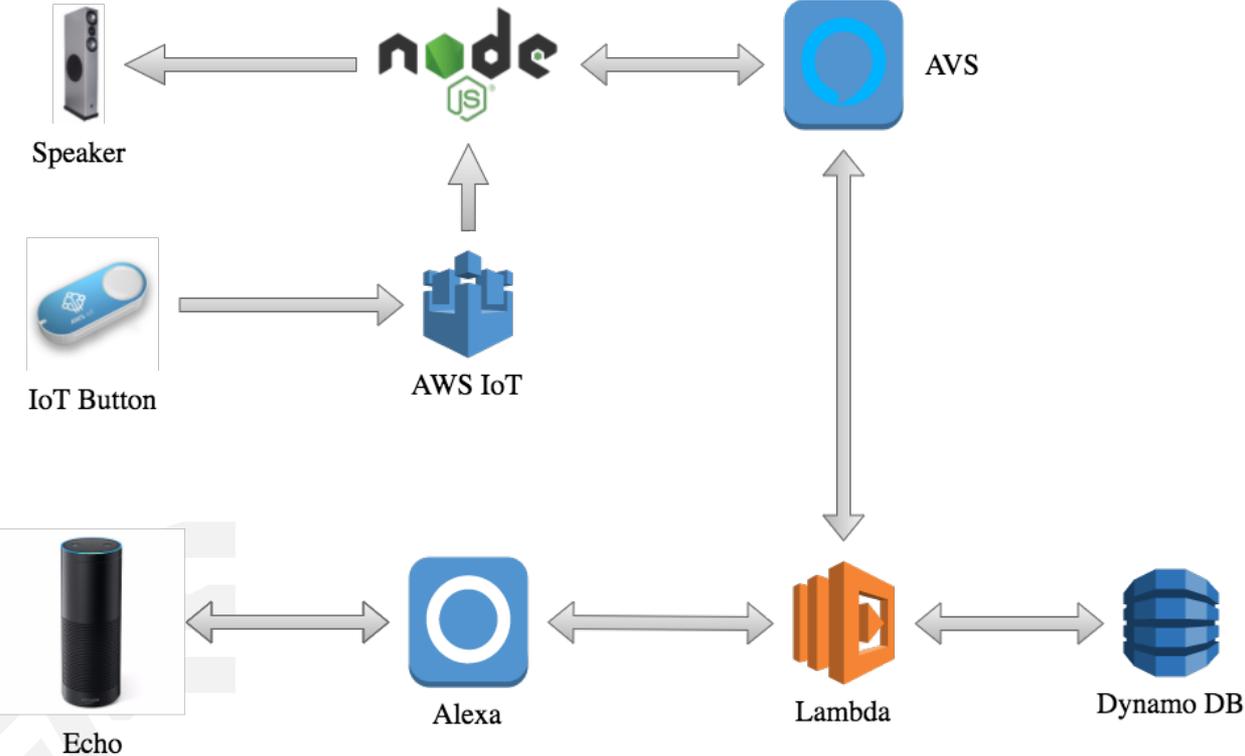


# Übersicht

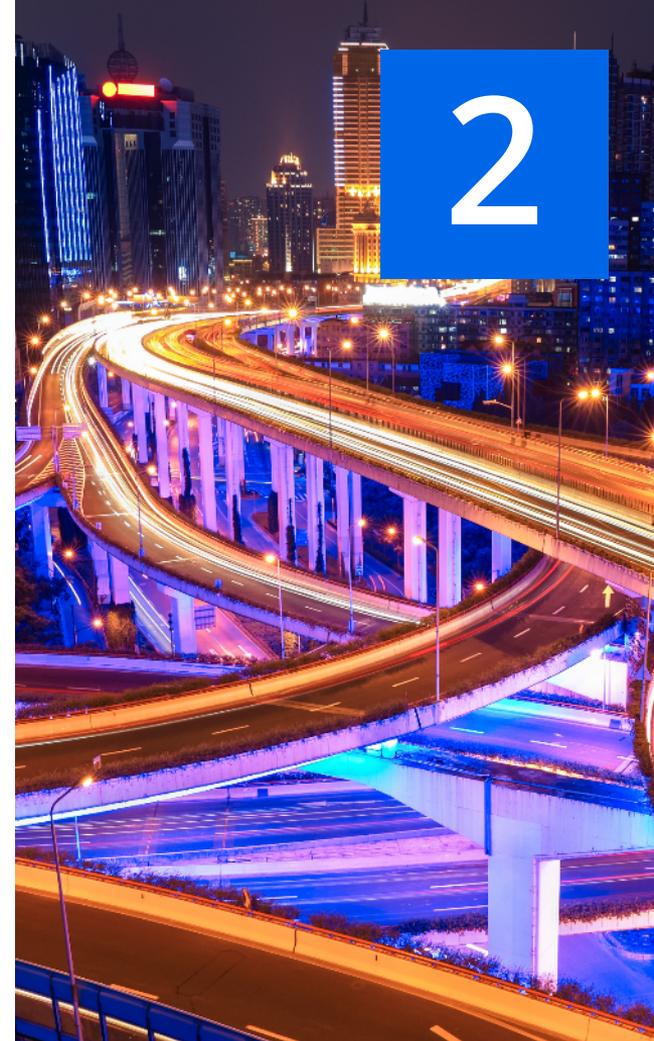
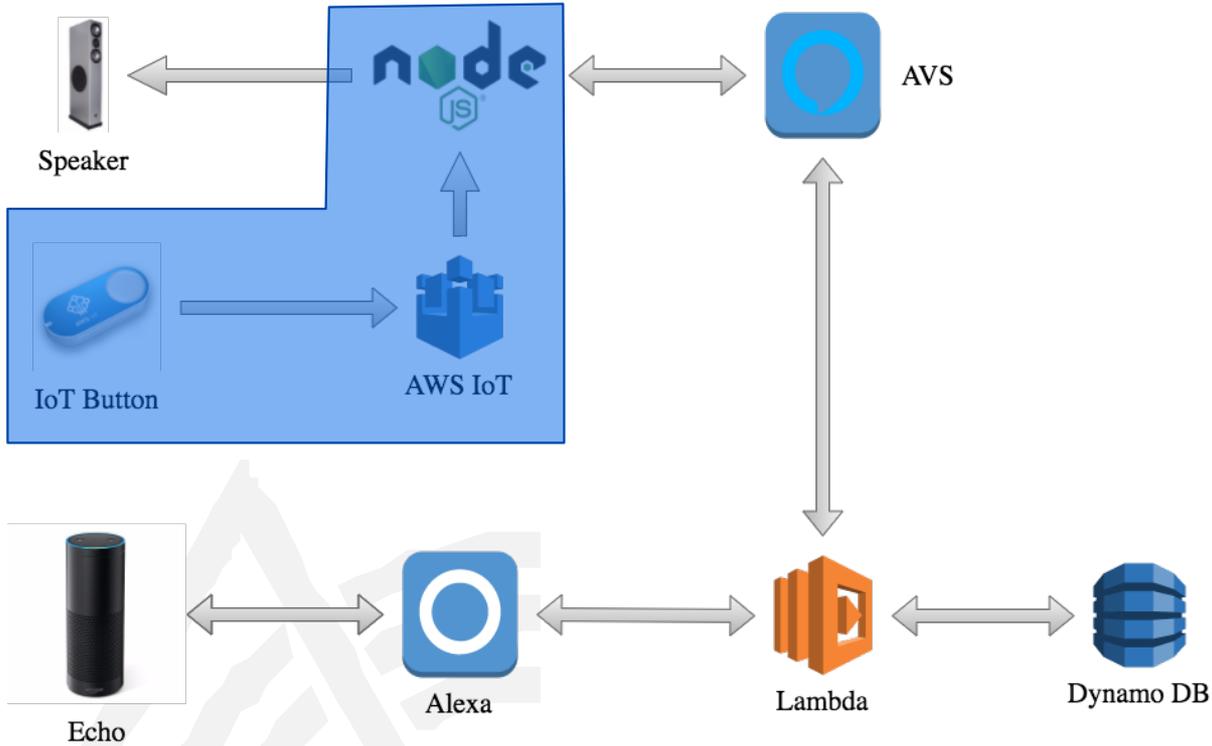




# Übersicht



# AWS IoT



# AWS IoT 101

- Amazons IoT Plattform
- Serverless Architekture
- Beahlt wird für die Nachrichten (5\$ / 1 Mio Nachrichten)
- SDK´s verfügbar

## SDKs

### Embedded C

[Get source in GitHub](#)

[Developer Guide](#)

[Porting Guide](#)

### JavaScript

[Get source in GitHub](#)

[Developer Guide](#)

### Arduino Yún

[Get source in GitHub](#)

[Developer Guide](#)

### Java

[Get source in GitHub](#)

[Developer Guide](#)

### Python

[Get source in GitHub](#)

[Developer Guide](#)

### iOS

[Get source in GitHub](#)

[Developer Guide](#)

[Samples](#)

### Android

[Get source in GitHub](#)

[Developer Guide](#)

[Samples](#)

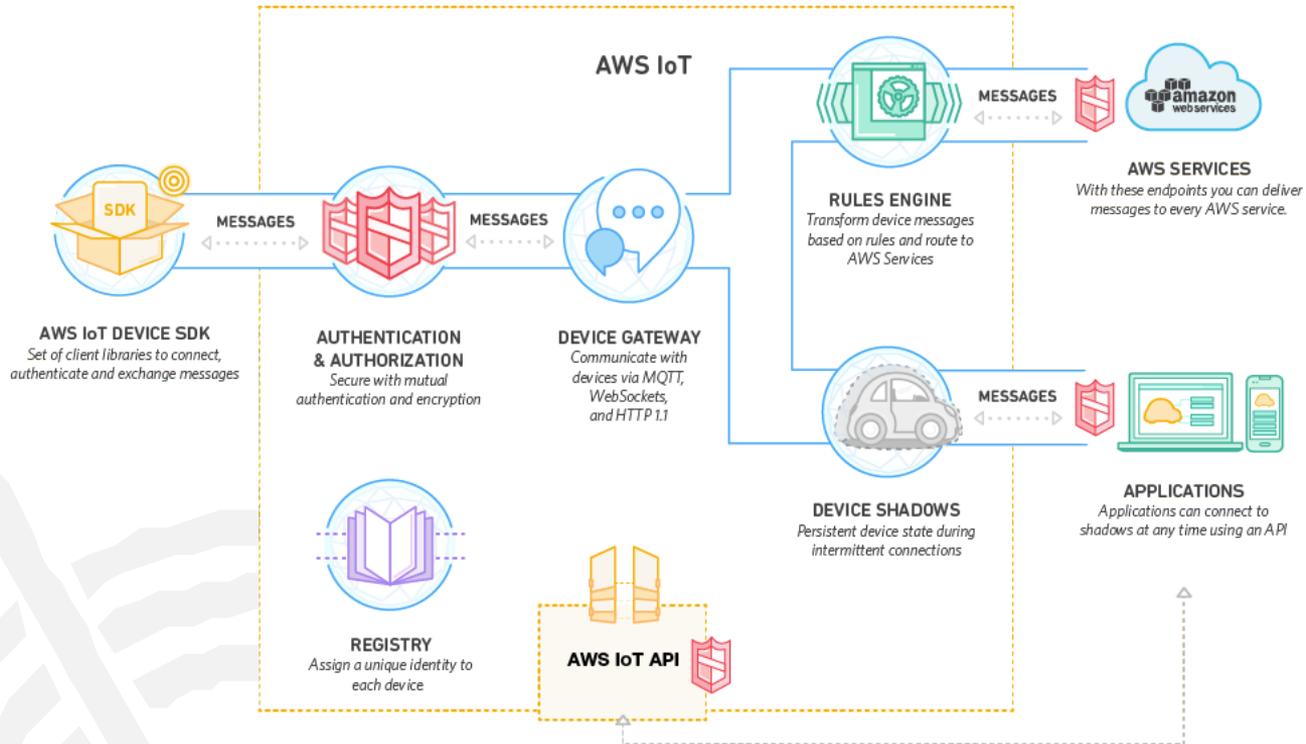
### C++ SDK

[Get source in GitHub](#)

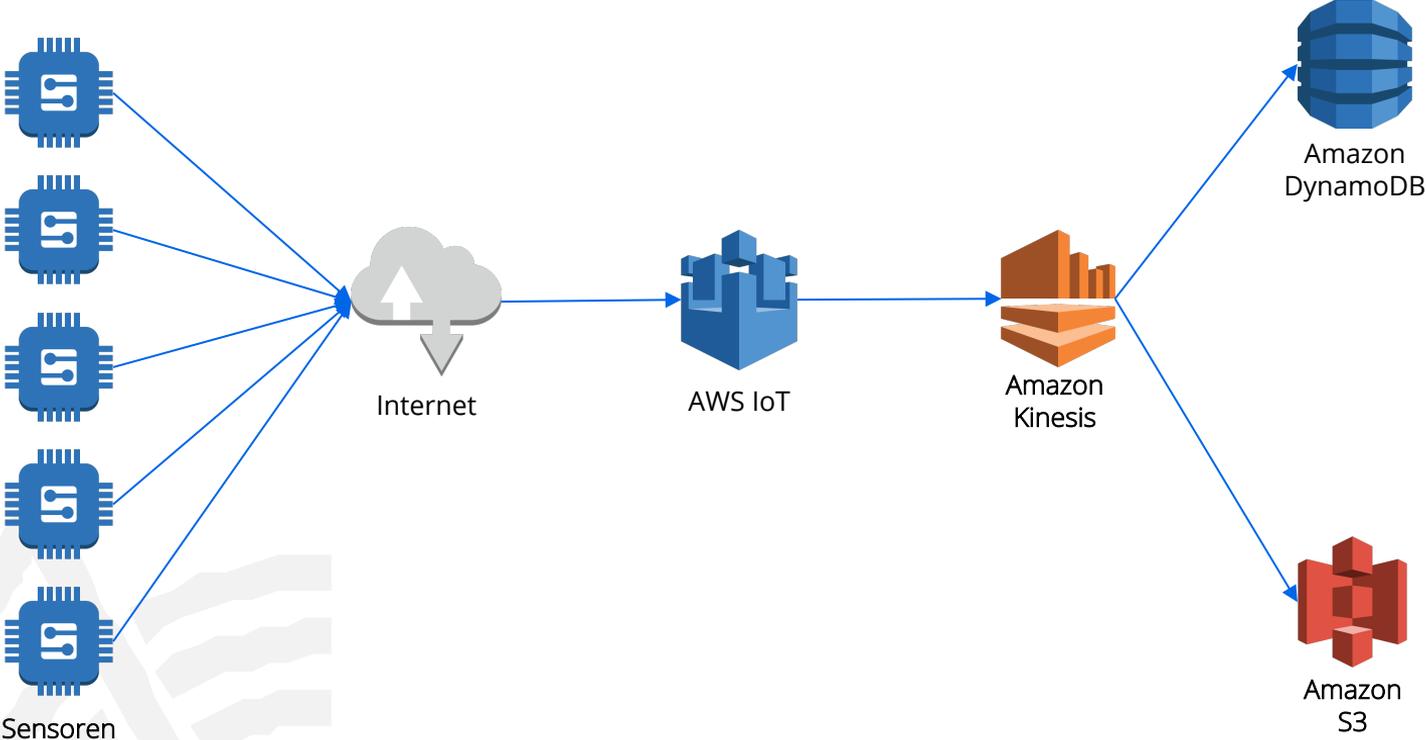
[Developer Guide](#)

[Samples](#)

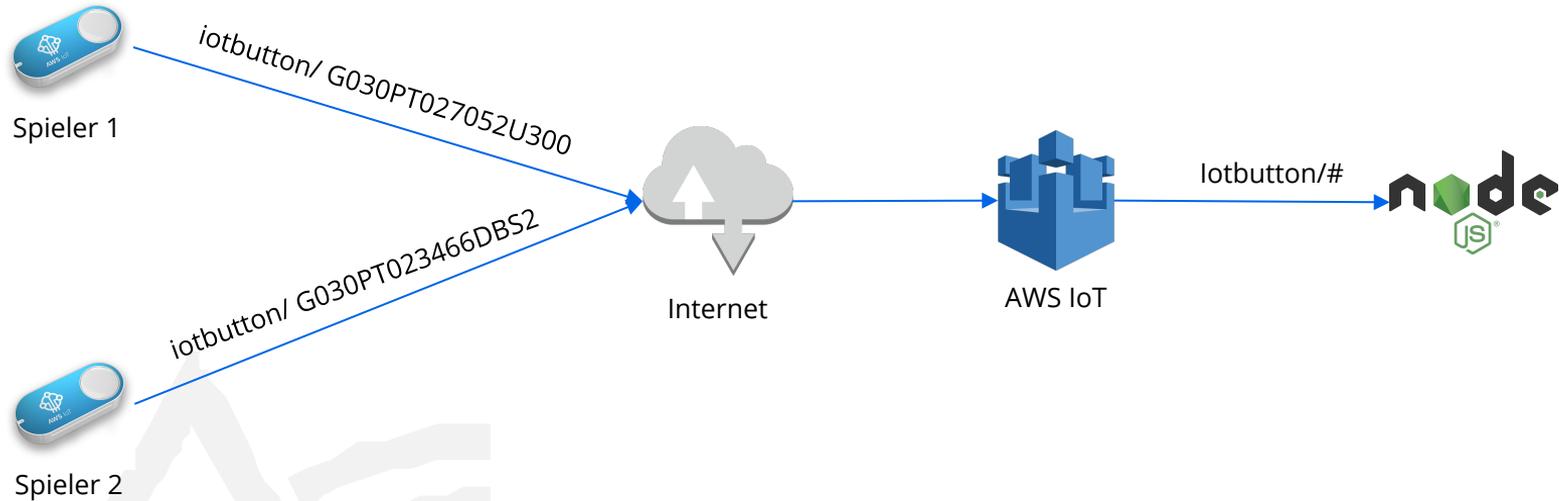
# AWS IoT 101



# Common Use Case



# Anwendung für Kickertisch



# Some Code

```
    });
  });
  const pingDownstream = function () {
    pingTimeout = setTimeout(pingDownstream, 300 * 1000);

    console.log([' + new Date().toISOString() + ' ] + "Ping!");

    getAccessToken().then(token => {
      var options = {
        host: 'aws-alexa-eu.amazon.com',
        path: '/ping',
        scheme: 'https',
        method: 'GET',
        headers: {'Authorization': 'Bearer ' + token}
      };

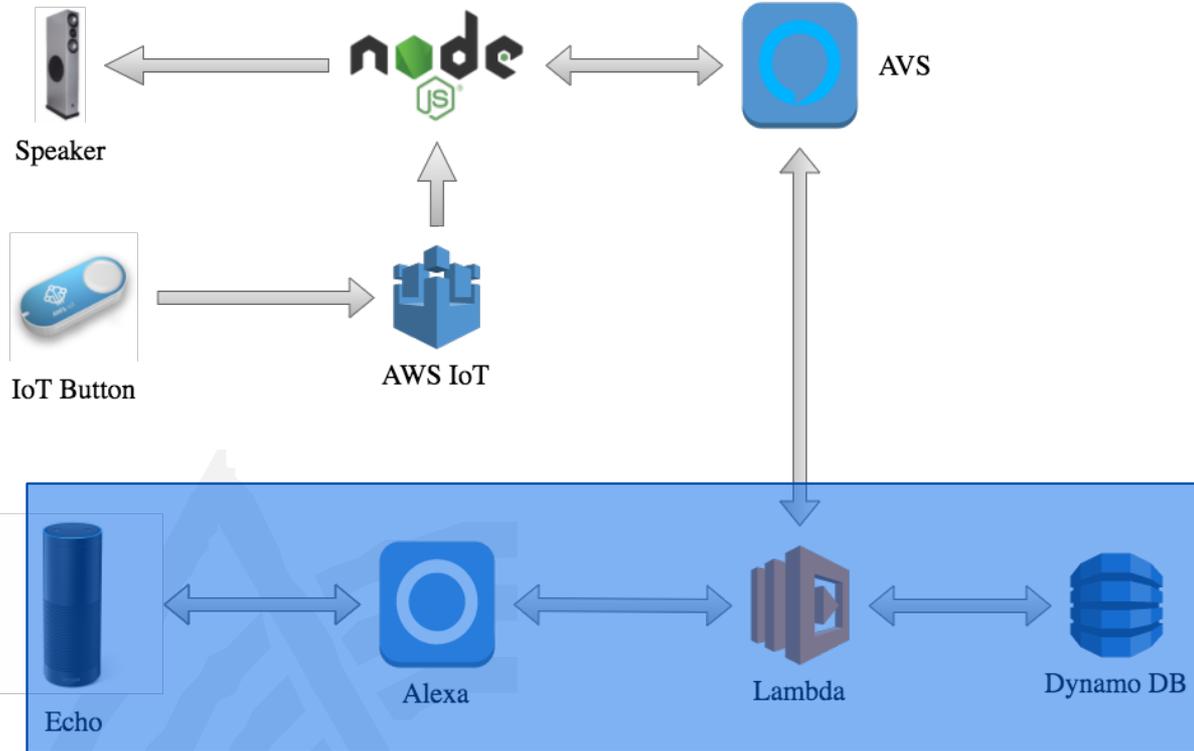
      var request = http2.request(options, function (res) {
        console.log('Ping STATUS: ' + res.statusCode);
        console.log('Ping HEADERS: ' + JSON.stringify(res.headers));
      });

      request.on('error', error => {console.error(JSON.stringify(error));});
      request.end();
    });
  });

  const synchronize_payload = dash + boundary + newLine
  + 'Content-Disposition: form-data; name="metadata"' + newLine
  + 'Content-Type: application/json; charset=UTF-8' + newLine
  + newLine
  + JSON.stringify({
    "event": {
      "header": {
        "namespace": "SpeechRecognition",
        "name": "RecognizerState",
        "payload": {
          "ALB": {}
        }
      }
    },
    "event": {
      "header": {
        "namespace": "System",
        "name": "SynchronizeState",
        "messageId": "message-123"
      },
      "payload": {}
    }
  }) + newLine
  + newLine
  + dash + boundary + dash;
```

```
Run: server.js test.js
/Users/mbu/.nvm/versions/node/v6.2.2/bin/node /Users/mbu/Documents/workspace/tabletopsoccer/PiTabletopSoccerServer/server.js
init AVS Service
[Tue, 19 Sep 2017 12:03:12 GMT] =====> Reconnect
create Downstream
IoT connect
[ { topic: 'iotbutton/#', qos: 0 } ]
received new Access_Token and Refresh_Token
Downstream STATUS: 200
Downstream HEADERS: {'access-control-allow-origin': '*', 'x-amzn-requestid': '02edb9fffe594ab0-00002d7b-00006485-ddf04d80c2d99734-2770dd61-1', 'content-type': 'multipart/related; boundary=-----abcde123; type=application/json'}
Synchronize
Sync STATUS: 204
Sync HEADERS: {'access-control-allow-origin': '*', 'x-amzn-requestid': '02edb9fffe594ab0-00002d7b-00006485-ddf04d80c2d99734-2770dd61-3'}
AVS Service Module initialized
iotbutton/G030PT023466DBS2
{ serialNumber: 'G030PT023466DBS2',
  batteryVoltage: '1741mV',
```

# Alexa Skill Kit



# Alexa Skill Kit 101

- Es gibt mehrere Skill Typen
  - Custom Interaction Skills
  - Smart Home Skills
  - Flash Briefing Skills
  - Video Skills
  
- Zwei Hauptbestandteile
  - Interaction Model
  - Service Funktion

# Interaction Model: Built In Intents

- Von Amazon bereits bereitgestellte Intents
- Decken Grundfunktionalitäten ab
- Es müssen keine Sample Uterances definiert werden

```
{"intents": [  
  {"intent": "AMAZON.HelpIntent"},  
  {"intent": "AMAZON.StopIntent"},  
  {"intent": "AMAZON.CancelIntent"},  
  {"intent": "AMAZON.YesIntent"},  
  {"intent": "AMAZON.NoIntent"}  
]}
```

# Interaction Model: Custom Intents

- Hauptbestandteil für eigenes Interaktionsmodell
- Können Slots enthalten
- Slots sind die Parameter der Intents
- Mehrere Build-In Slot Types + Eigene

```
{ "intents": [  
  { "intent": "StartGame"},  
  { "intent": "Punkttestand"},  
  { "intent": "PlayerPoint",  
    "slots": [  
      {  
        "name": "Player",  
        "type": "SPIELER_PUNKT"  
      }  
    ]  
  },  
]  
}
```

# Interaction Model: Sample Uterances

- Definieren welche Sätze auf welchen Intent matchen
- Viel hilft viel!!!
- Slots werden mit {} markiert

StartGame starte neues match  
StartGame beginne neues match  
StartGame starte ein neues spiel  
StartGame beginne ein neues spiel

Punktstand wie ist der punktstand  
Punktstand wie steht es  
Punktstand gib mir den punktstand  
Punktstand wer führt  
Punktstand wer gewinnt

PlayerPoint Punkt für {PLAYER\_POINT}

# Interaction Model:

<https://jaxenter.de/chatbot-alexa-google-home-siri-54265>



## **Christian Ochsenkühn**

Christian Ochsenkühn arbeitet bei der OPITZ CONSULTING Deutschland GmbH im Bereich Software Development. Zudem beschäftigt er sich für die Abteilung Business Development & Innovation mit aktuellen Trends und innovativen Lösungen für digitale Problemstellungen. Als Betreiber eigener Webangebote kennt er die Herausforderungen des schnellen technischen Wandels und konzentriert sich auf eine optimale Benutzererfahrung.

**[Alle Beiträge von Christian Ochsenkühn](#)**

# Service Function

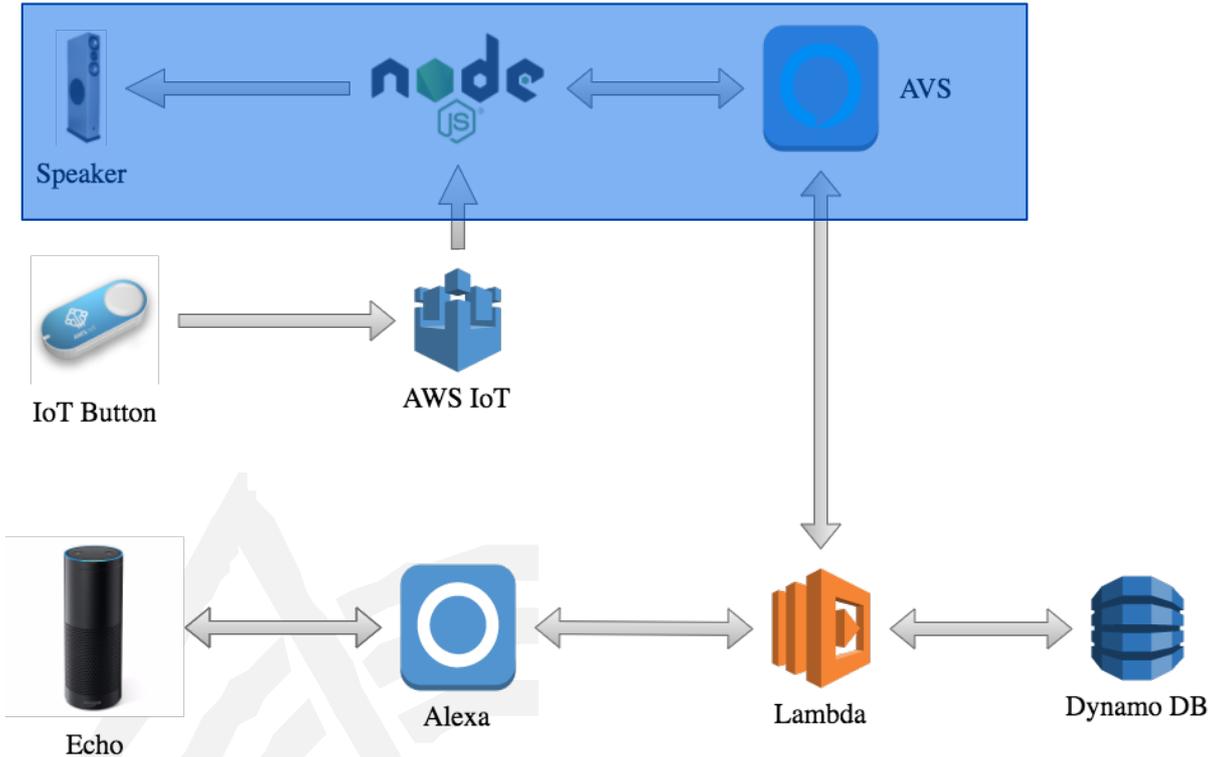
- Werden entweder als AWS Lambda Function oder Webservice implementiert
- Für Lambda existieren mehrere SDK's
- NodeJS und Python haben die größte Verbreitung

# Some Code

```
};  
});  
const pingDownstream = function () {  
  pingTimeout = setTimeout(pingDownstream, 300 * 1000);  
  
  console.log([' + new Date().toISOString() + ' ] + "Ping!");  
  
  getAccessToken().then(token => {  
    var options = {  
      host: 'aws-alexa-eu.amazon.com',  
      path: '/ping',  
      scheme: 'https',  
      method: 'GET',  
      headers: {'Authorization': 'Bearer ' + token}  
    };  
  
    var request = http2.request(options, function (res) {  
      console.log('Ping STATUS: ' + res.statusCode);  
      console.log('Ping HEADERS: ' + JSON.stringify(res.headers));  
    });  
  
    request.on('error', error => {console.error(JSON.stringify(error));  
    request.end();  
  });  
});  
  
const synchronize_payload = dash + boundary + newLine  
+ 'Content-Disposition: form-data; name="metadata"' + newLine  
+ 'Content-Type: application/json; charset=UTF-8' + newLine  
+ newLine  
+ JSON.stringify({  
  "event": {  
    "header": {  
      "namespace": "SpeechRecognition",  
      "name": "RecognizerState",  
      "payload": {  
        "ALB": {}  
      }  
    },  
    "event": {  
      "header": {  
        "namespace": "System",  
        "name": "SynchronizeState",  
        "messageId": "message-123"  
      },  
      "payload": {}  
    }  
  }) + newLine  
+ newLine  
+ dash + boundary + dash;  
});  
});
```

```
Run: server.js test.js  
+ /Users/mbu/.nvm/versions/node/v6.2.2/bin/node /Users/mbu/Documents/workspace/tabletopsoccer/PiTabletopSoccerServer/server.js  
init AVS Service  
[Tue, 19 Sep 2017 12:03:12 GMT] =====> Reconnect  
create Downstream  
IoT connect  
[ { topic: 'iotbutton/#', qos: 0 } ]  
received new Access_Token and Refresh_Token  
Downstream STATUS: 200  
Downstream HEADERS: {'access-control-allow-origin': '*', 'x-amzn-requestid': '02edb9fffe594ab0-00002d7b-00006485-ddf04d80c2d99734-2770dd61-1', 'content-type': 'multipart/related; boundary=-----abcde123; type=application/json'}  
Synchronize  
Sync STATUS: 204  
Sync HEADERS: {'access-control-allow-origin': '*', 'x-amzn-requestid': '02edb9fffe594ab0-00002d7b-00006485-ddf04d80c2d99734-2770dd61-3'}  
AVS Service Module initialized  
iotbutton/G030PT023466DBS2  
{ serialNumber: 'G030PT023466DBS2',  
  batteryVoltage: '1741mV',
```

# Alexa Voice Service



# Alexa Voice Service 101

- Integration von Alexa in eigenes Device
- SDK für C++ verfügbar
- HTTP/2 API

[heise online](#) > [News](#) > [2017](#) > [KW 37](#) > [Seat holt Alexa ins Auto](#)

## Seat holt Alexa ins Auto

13.09.2017 17:03 Uhr – Stefan Porteck

 vorlesen



# Warum in NodeJS?

Weil es geht!!!

Und ich kein C++ kann.

# AVS API v2

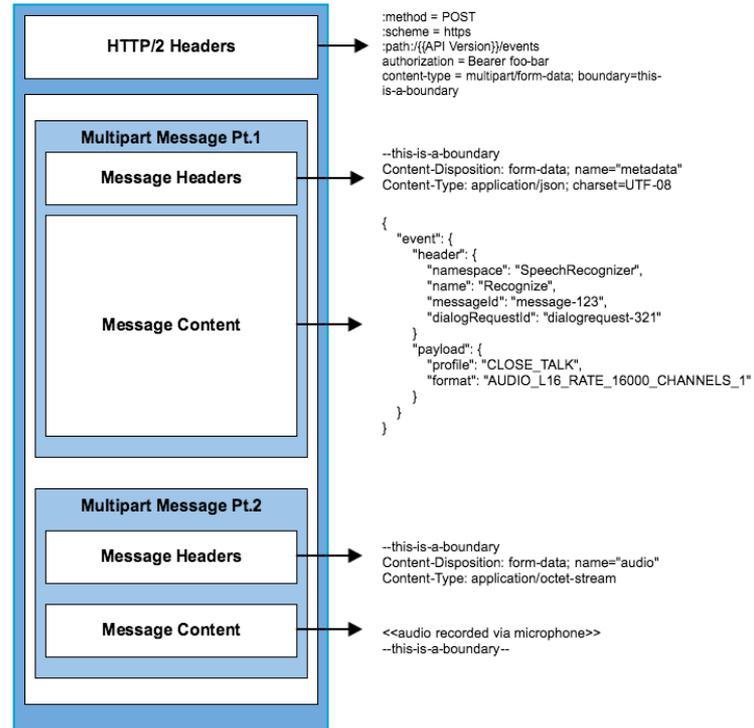
- Umstieg auf HTTP/2
- Erfordert Erzeugung eines „Downstream“ Kanals
- Ping alle 5 Minuten



# Authorization

- Login with Amazon Access Token wird benötigt
- Authorisierung per CompanionApp
- Authorisierung per Companion Site
- Authorisierung AVS Produkt

# Aufbau HTTP/2 Multipart Message



# SpeechRecognizer

- Einziger API Endpoint der im Projekt genutzt wurde
- Audio Sample mit dem Kommando wird an AVS gesendet
- Das Sample wird ausgewertet und der entsprechende Alexa Skill aktiviert
- Die Antwort des Skills wird über den Lautsprecher ausgegeben

# Probleme bei der Benutzung von NodeJS und AVS

- HTTP/2
  - HTTP/2 in NodeJS 8.4 funktionierte nicht für die Authorisierung
  - HTTP/2 (<https://github.com/molnarg/node-http2>) für Node 6.10 hatte Probleme bei größeren Audio Files
  - HTTP/2 in NodeJS 6.2.2 funktioniert
- Downchannel über längeren Zeitraum offen halten hat lange gedauert
- Erstes Modul für Audio Ausgabe war nicht gut



Fragen?