# Konzeption von Referenzarchitekturen für IoT-Zeitreihenverarbeitung in der Cloud

Bachelorarbeit

vorgelegt am 10. Mai 2021


Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

Kurs WWI2018C

von

Lukas Fruntke



Betreuer in der Ausbildungsstätte:

SPIRIT/21 GmbH
Philipp Arnold
Architekt Native Cloud Solutions

DHBW Stuttgart:

Prof. Dr. Thomas Specht


Unterschrift des Betreuers

Böblingen, 07.05.2021

**Abstract**

The goal of this bachelorthesis is, to construct reference architectures for addressing timeseries analytics in the Amazon Web Services (AWS) Cloud. Both modes of analytics, near realtime and batch are inspected and get an own reference architecture which are compared afterwards. To construct the reference architectures, a set of offerings from the AWS Cloud is to be examined and compared against each other. To ensure maximum utility, feedback and input is collected in various forms throghout the construction process from SPIRIT/21 GmbH, the company applying the constructed reference architectures. **To do** Improve this draft (1)

# Inhaltsverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| **API** | Application Programming Interface |
| **AVX2** | Advanced Vector Extensions 2 |
| **AWS** | Amazon Web Services |
| **CSV** | Comma seperated values |
| **DC2** | Dense Compute 2 |
| **DMS** | Database Migration Service |
| **EBS** | Elastic Block Storage |
| **EC2** | Elastic Compute Cloud |
| **ECS** | Elastic Container Service |
| **EFS** | Elastic Filesystem |
| **EMR** | Elastic Map Reduce |
| **ETL** | Extract, Transform, Load |
| **FaaS** | Function as a service |
| **HDD** | Hard Disk Drive |
| **IaaS** | Infrastructure as a Service |
| **IIoT** | Industrial Internet of Things |
| **IoT** | Internet of Things |
| **IOPS** | Input/Output Operations Per Second |
| **JDBC** | Java Database Connectivity |
| **JSON** | JavaScript Object Notation |
| **LoRaWAN** | Long Range Wide Area Network |
| **MoM** | Message oriented Middleware |
| **MQTT** | Message Queuing Telemetry Transport |
| **MSK** | Managed Streaming for Apache Kafka |
| **NAT** | Network Address Translation |
| **NCS** | Native Cloud Solutions |
| **OLAP** | Online Analytical Processing |
| **PaaS** | Platform as a Service |
| **QoS** | Quality of Service |
| **RAM** | Random-Access Memory |
| **R&D** | Research & Development |
| **SaaS** | Software as a Service |

| | |
|---|---|
| **SLA** | Service Level Agreement |
| **SNS** | Simple Notification Service |
| **SQL** | Structured Query Language |
| **SQS** | Simple Queue Service |
| **S3** | Simple Storage Service |
| **UDF** | User Defined Function |
| **UTC** | Universal Time Coordinated |
| **VPC** | Virtual Private Cloud |

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 1 Einleitung

Im Folgenden wird für die vorliegende Arbeit anhand der Problemstellung und der daraus folgenden Zielsetzung der Aufbau und damit das Vorgehen für die nachfolgenden Kapitel erläutert.

## 1.1 Problemstellung und Zielsetzung

Im Rahmen des Ausbaus der Datenanalysefähigkeiten der SPIRIT/21 GmbH mangelt es bis dato an einem Konzept, um Zeitreihen-Daten in Echtzeit oder nach Ablage in einer Datenbank zu analysieren. Diese Zeitreihendaten könnten beispielsweise Messungen von Internet of Things (IoT)-Sensoren sein, die in regelmäßigen Intervallen übermittelt werden. Die Vorteile der Public Cloud, wie Effizienzgewinne, Skalierbarkeit und nutzungsbasierte Abrechnung, welche die Cloudkunden der SPIRIT/21 GmbH gewohnt sind, sollen auch bei diesen Datenanalysen zum Tragen kommen. Aufgrund der strategischen Priorisierung des Cloudanbieters Amazon Web Services (AWS) sollen primär die offerierten Dienste dieses Unternehmens verwendet werden. Als Lösungsvorlage für künftige Kundenprobleme sollen Referenzarchitekturen erstellt werden, welche wiederverwendet werden können. Besonders die Datenanalyse von Daten in der Datenbank und die Echtzeitanalyse sind dabei für die Zeitreihendaten von Relevanz, weshalb für beide Fälle eine Referenzarchitekktur ausgearbeitet wird. Zusätzlich soll eine Empfehlung ausgearbeitet werden, welche Referenzarchitektur und welche zugehörige Technik bei welchen Problemstellungen eingesetzt werden können. Als konkreter Anwendungsfall dienen IoT-Daten, die bereits von mehreren Datenlieferanten, wie Sensoren, in der Cloud gespeichert werden und bei welchen viele Datenpunkte zur Analyse vorliegen. Ausgehend von diesen IoT-Daten können die konzipierten Referenzarchitekturen auch für andere Zeitreihendaten, welche beispielsweise beim Anwendungs-Monitoring anfallen, verwendet werden. Die SPIRIT/21 GmbH antizipiert einen Anstieg von Kundenprojekten mit Datenanalyseanteilen, weshalb eine Bachelorarbeit, die ein Konzept für solche Datenanalysen ausarbeitet, benötigt wird. Zusätzlich sieht sich die SPIRIT/21 GmbH in einer ähnlichen Situation, wie die Mehrheit von 545 in einer Umfrage befragten Unternehmen, die angaben, unzureichendes Know-how für Datenanalyse zu besitzen.[1] Da eine Referenzarchitektur für gewöhnlich als Sammlung von best practices und zumindest als Inspiration für eigene Architekturen dient, kann so auch der Know-how Aufbau der Cloud Mitarbeiter initiiert werden.

Im Rahmen der Arbeit werden Konzepte für Referenzarchitekturen entwickelt, die zeigen, wie Daten sowohl in Echtzeit als auch mit Zeitverzögerung in der Public Cloud analysiert werden können. Ebenso soll die Arbeit Entscheidungskriterien liefern, wann welche Form der Datenanalyse für die verschiedenen in der Arbeit ausgewählten Analysen verwendet werden soll. Dabei werden für die konzipierten Referenzarchitekturen jeweils die unter den definierten Randbedingungen optimalen Analysedienste im Rahmen der Bachelorarbeit ausgesucht.

[1] Vgl. Business Application Research Center o.J.

**To do** final überarbeiten (2)

## 1.2 Aufbau der Arbeit

Am Anfang der Arbeit werden anhand einer Literaturrecherche die Grundlagen der Datenverarbeitung mit speziellem Fokus auf die möglichen Auswertungen und die präferierten Verarbeitungsarten dargestellt.

Ebenfalls werden infrage kommende Analysedienste beleuchtet. Daraufhin wird mittels Literaturrecherche die später zu verwendende Methodik der Referenzmodellierung dargestellt. Des Weiteren wird die verwendete Methodik der Anforderungserhebung und der durchgeführten Interviews geschildert. Im Folgenden werden die Kriterien und das Vorgehen beschrieben, nach denen die Analysedienste verglichen und ausgewählt werden.

Im Anwendungsteil werden zuerst die Rahmenbedingungen der Datenverarbeitung dargestellt, woraufhin existierende Anwendungsfälle, die analysierbare Daten produzieren, erläutert werden. Im Anschluss werden für die Batch- und die Echtzeitverarbeitung eine Dienstauswahl nach den vorgestellten Kriterien durchgeführt.

Folgend werden im Modellierungskapitel Referenzarchitekturen basierend auf den Anforderungen der interviewten Stakeholder und basierend auf den selektierten Diensten entworfen. Für diese Referenzarchitekturen ergeben sich aufgrund gefundener Stärken und Schwächen der Ansätze passende Einsatzmodelle, welche erläutert werden. Abschließend werden die Ergebnisse der Arbeit reflektiert und in einer Handlungsempfehlung die Ergebnisse zusammengefasst.

**To do** final überarbeiten (3)

# 2 Theoretische Grundlagen zu Zeitreihen

In diesem Kapitel werden die theoretischen Grundlagen der Zeitreihe und möglicher Analyseformen, genauso wie die Theorie der Referenzmodellierung und die Systematik der Anforderungserhebung und Dienstauswahl erläutert.

## 2.1 Grundlagen der Datenanalyse

Mathematisch ausgedrückt besteht eine Zeitreihe (im Englischen auch „time series“ genannt) aus einer endlichen Menge an zeitlich aufsteigend sortierten Messwerten $x_{t_1}, x_{t_2}, ..., x_{t_T}; x_{t_k} \in \mathbb{R}^n, k = 1, 2, ..., T$, wofür $t_1 < t_2 < t_T$ gilt.[2] Zeitreihendaten sind in verschiedenen Feldern zu finden, beispielsweise im Bereich der Aktienmärkte, wo die Preise der einzelnen Kurse abgebildet werden, im Bereich der Gesundheitsforschung, wo die Ansteckungsrate ansteckender Krankheiten wie Covid-19 verfolgt wird oder im Bereich der Sozialwissenschaften, in welchen der Verlauf von Geburtsraten über die Zeit analysiert werden soll.[3] In Abbildung 1 findet sich beispielhaft die Zeitreihe der zwei messbaren Feinstaubkategorien von null bis ein Uhr am 01.01.2020 in Stuttgart.

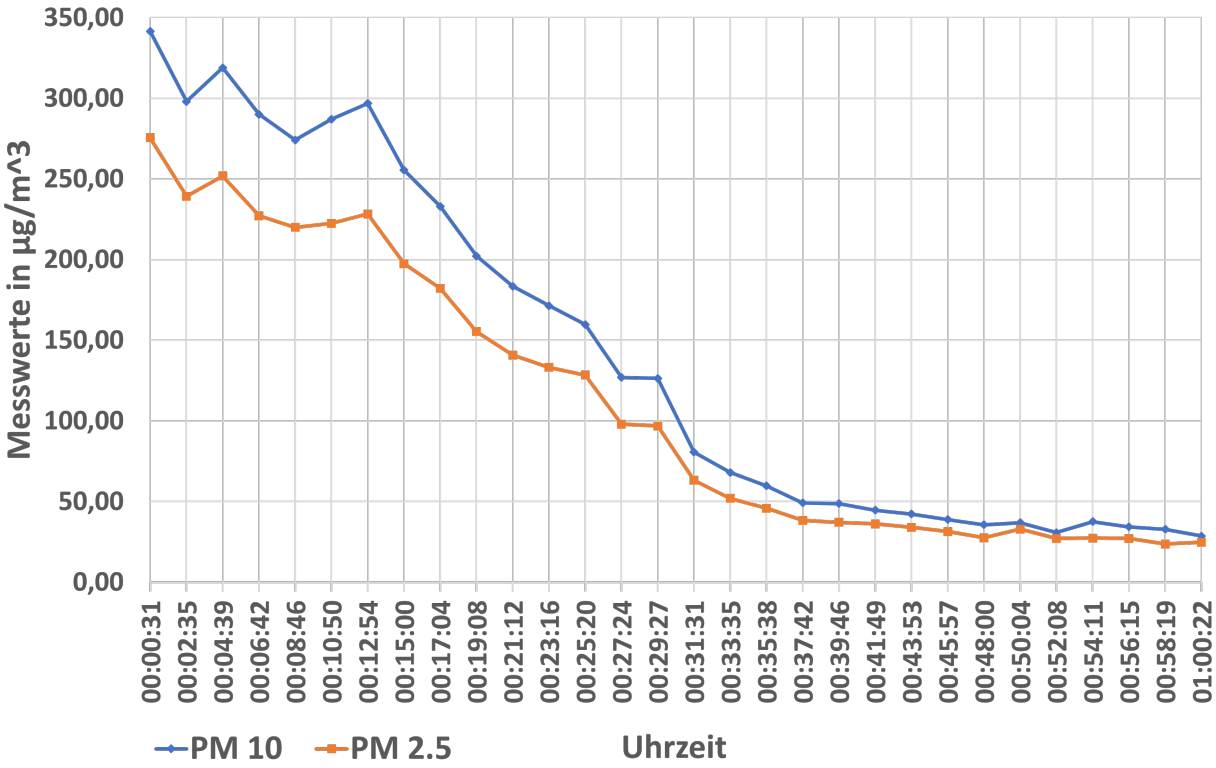


Abb. 1: Beispielhafte Feinstaubwerte in der Stunde nach Silvester 2020 in Stuttgart-Mitte

[2] Vgl. Deistler/Scherrer 2018, S. 1
[3] Vgl. Shumway/Stoffer 2017a, S. 1

Die möglichen Anwendungen in der Informatik sind ebenfalls vielzählig. So können über virtuelle oder physikalische Sensoren Messwerte, wie beispielsweise die CPU Auslastung eines gegebenen Servers über die Zeit oder Temperaturmessungen eines IoT Gerätes über die Zeit gemacht und gespeichert werden. So nehmen mit zunehmender Digitalisierung von Fertigungslinien in Fabriken auch die Messwerte von Sensoren zu, die dort übermittelt und ausgewertet werden müssen.

Ein wichtiges Merkmal von Zeitreihen ist die Distanz zwischen den Messwerten, im Sinne der Messfrequenz, in welcher Daten betrachtet werden. **To do** belegen (4) Ist eine Zeitreihe äquidistant, wurde mit gleichbleibender Frequenz gemessen und die zeitliche Distanz zwischen einzelnen Messwerten ist gleich. Für die in dieser Arbeit diskutierten Auswertungsarten wird eine Äquidistanz der gemessenen Daten angenommen, da andernfalls ein Bias/eine Verfälschung bei der Analyse nicht ausgeschlossen werden kann. Gleichfalls ist es technisch möglich, einzelne, nicht äquidistante Messwerte auszusortieren oder fehlende Messwerte zu interpolieren. Die in Abbildung 1 abgebildete Zeitreihe ist äquidistant, da die Werte alle 124 Sekunden erhoben wurden.

Es ist von zwei vorliegenden Typen von Zeitreihendaten auszugehen. Zum einen existiert der mit Zeitstempel versehene Messwert, welcher einen oder in manchen Fällen auch mehrere diskrete Messwerte mit Zeitstempel der Erfassung übermittelt. Zum anderen existieren auch Ereignisse, welche keine Messwerte enthalten, sondern Ergebnisse einer Vorauswertung innerhalb des übermittelnden Systems sind. So wäre ein Ereignis beispielsweise ein niedriger Batteriestand eines verbundenen Sensors. Abhängig von der Vorauswertung werden Ereignisse einmal, nach Auftreten mit Zeitstempel oder periodisch mit Zeitstempel bis zum Beheben der Ursache versendet.[4]

### 2.1.1 Analysewert verarbeiteter Daten

Bei der Verarbeitung von Daten ist zu beachten, dass der Wert, bzw. die Erkentnisse die aus den Daten abgleitet werden können, über die Zeit reduziert wird.[5] Gemäß Nucleus Research, Inc. haben dabei verschiedene Unternehmen verschiedene Zeiträume, in denen Daten nützlich sind, da sie sich in eine von drei Entscheidungstempi einkategorisieren lassen.[6] Die Entscheidungstempi sind taktisch, operativ und strategisch. Beim taktischen Entscheidungstempo werden Änderungen sehr schnell, nahe Echtzeit getroffen und implementiert. Im Gegensatz dazu werden Entscheidungen der operativen und strategischen Entscheidungstempi respektive erst in Tagen bzw. Wochen oder innerhalb einem Quartal oder länger implementiert. Je nach Entscheidungstempo sind Analysen von eingehenden Daten also wesentlich früher notwendig oder können beispielsweise auch nur einmal täglich erstellt werden.

Für Unternehmen mit taktischem Entscheidendungstempo haben, gemäß der in Abbildung 2 gezeigten Befragungsergebnisse von Nucleus Research, Inc., Daten nach maximal 30 Minuten die

[4] Siehe auch Anhang 2
[5] Vgl. auch im Folgenden Nucleus Research, Inc. 2012
[6] Vgl. auch im Folgenden Nucleus Research, Inc. 2012, S. 3

Hälfte des Wertes eingebüßt.[7] Für operative Entscheidungstempi ist die durchschnittliche Halbwertszeit nach acht Stunden erreicht, für strategische Entscheidungstempi nach ca. 56 Stunden, also nach über 2 Tagen.

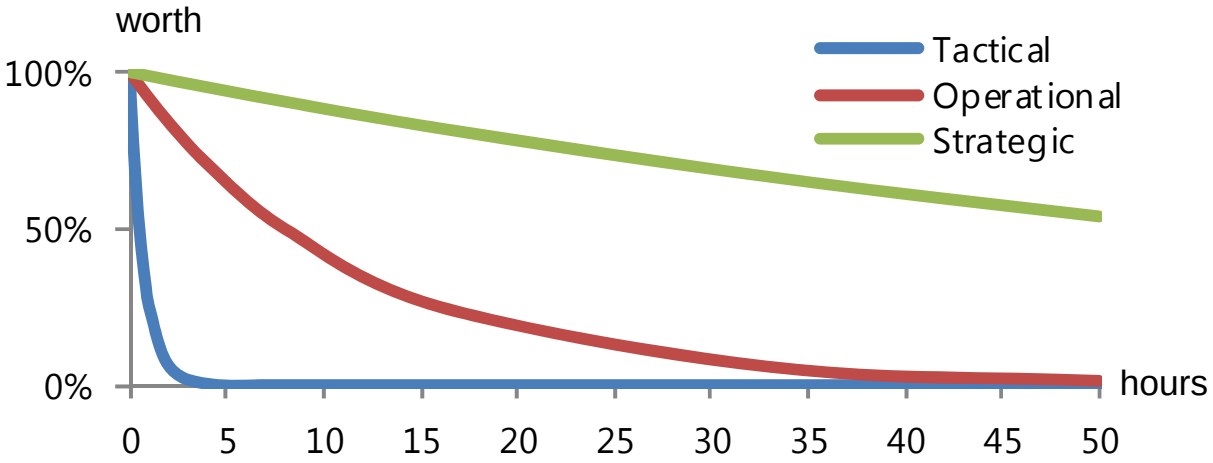


Abb. 2: Die Halbwertszeit von Daten[8]

Aus diesen abweichenden Halbwertszeiten und damit aus den abweichenden Zeiträumen, in denen die erhobenen Daten den höchsten Wert haben, ergibt sich die Notwendigkeit von verschiedenen Datenverarbeitungsstrategien, um entsprechend taktischen, operativen oder strategischen Entscheidenden die werthaltigsten Daten als Entscheidungsgrundlage zu präsentieren.

### 2.1.2 Arten der Auswertung

Diese Arbeit soll anhand einiger weniger Auswertungen demonstrieren, wozu die jeweilige Referenzarchitektur und die verwendeten Dienste fähig sind. Im Folgenden werden dazu einige einfachere Auswertungsmethoden für Zeitreihendaten vorgestellt. Die Zeitreihenanalyse und einsetzbare Werkzeuge werden im statistischen Sinn von weiterführenden Werken, wie von Shumway/Stoffer behandelt. Es ist davon auszugehen, dass auch weiterführende Auswertungen wie fourieranalytischen Methoden, sofern in der jeweiligen Umgebung bereits vorhanden oder programmierbar, eingesetzt werden können.

#### Median und Quantile

Eine der wichtigsten statistischen Lageparameter sind die Quantile eines Datensatzes und der Median, welcher ein spezielles, 50 prozentiges Quantil ist. Innerhalb eines gewissen Betrachtungsfensters können für Zeitreihendaten, wie für viele andere Datensätze auch, Quantile und

[7] Vgl. auch im Folgenden Nucleus Research, Inc. 2012, S. 6
[8] Mit Änderungen entnommen aus: Nucleus Research, Inc. 2012

der Median erfasst werden. Das Quantil erfasst mit der jeweiligen Prozentangabe und dem zugehörigen, ermittelten Wert die Grenze an Werten, die kleiner sind. Bei einem 75-%-Quantil von 2 wären entsprechend nur 25% der sich im Datensatz befindlichen Daten größer als 2. Da ein Median das 50-%-Quantil ist, bildet der Median entsprechend die Grenze ab, bei welcher 50% der Werte im Datensatz kleiner und der Rest größer ist. In Abbildung 3 sind das 25, 50 und 75-%-Quantil einer Messreihe gezeigt.

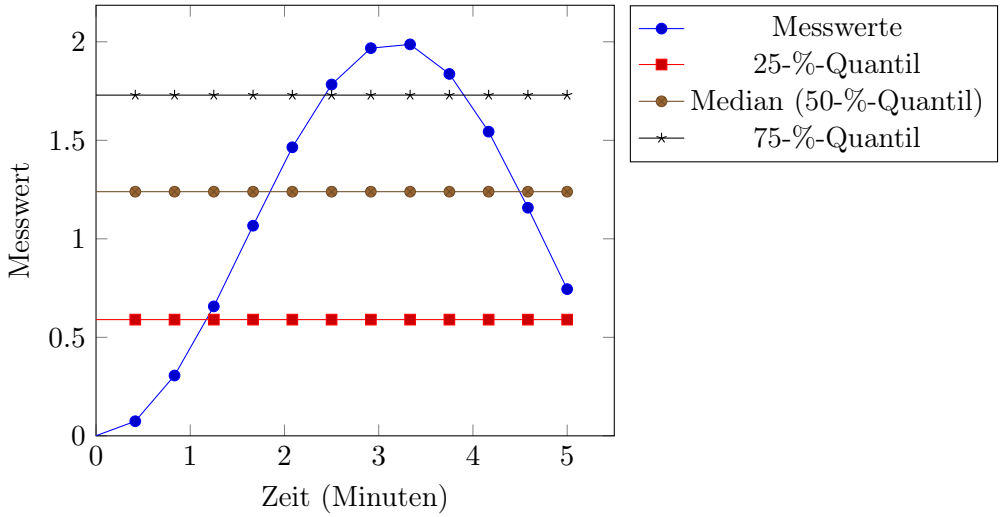


Abb. 3: Median und Quantile angewendet auf eine Messreihe

Gängig in der Datenanalyse als Sonderform der Quantile sind die sogenannten Perzentile, welche in einprozentigen Schritten existieren. Häufig verwendet werden insbesondere das 90-% und das 99% Perzentil.

### Anomaliedetektion

Eine weitere gängige Datenanalysemethode ist die Anomaliedetektion. Dabei kann jeder Datenpunkt als Anomalie gesehen werden, der die Komplexität eines Modells, welches die Daten beschreibt, substantiell erhöhen würde.[9] Verwandt ist dabei das Feld der „Ausreissererkennung“/outlier detection. Verwendet werden können dabei diverse Methodiken. Innerhalb von AWS Diensten wird zur Anomalieerkennung ein Algorithmus basierend auf Random Cut Forest verwendet.[10] Weitere Methodiken wären aber auch Random Walk oder Logik basierend auf Standardannahmen.[11,12] Anomalien können kausal mit unterliegenden Problemen z.B. des Sensors zusammenhängen, weshalb es wichtig ist, auch abseits von den in Unterunterabschnitt 2.1.2 gezeigten Schwellwertüberprüfungen die Daten auf Anomalien zu prüfen.

[9] Vgl. Guha u. a. 2016
[10] Vgl. Guha u. a. 2016, S. 1
[11] Vgl. Moonesinghe/Tan, P.-N. 2006
[12] Vgl. Angiulli/Ben-Eliyahu - Zohary/Palopoli 2008

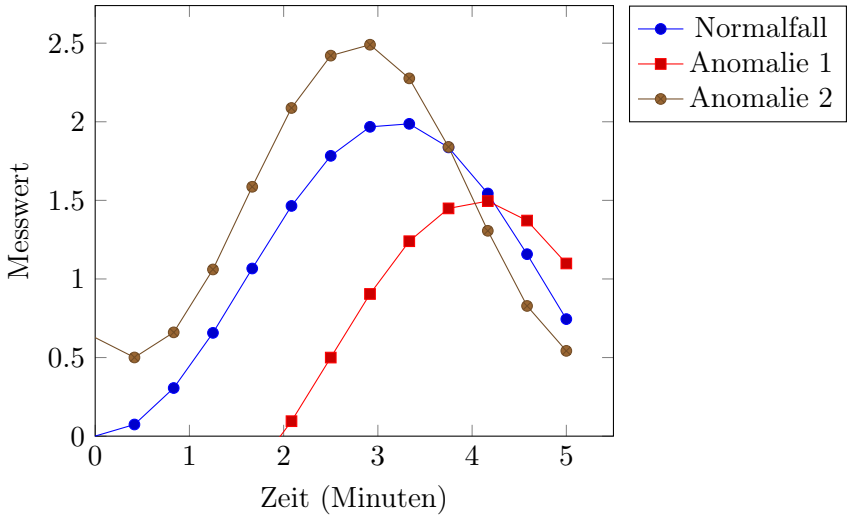


Abb. 4: Zu erkennende Anomalien einer Messreihe

Abgebildet in Abbildung 4 sind zwei Anomalien, die direkt in mehreren Punkten vom Normalfall abweichen.

### Schwellwertüberschreitung

Bei vielen gemessenen und sonstig erfassten Zeitreihendaten, bei denen akzeptierbare Werte und Wertbereiche, die Aktionen erfordern bekannt sind, sind Schwellwerte bereits ausreichend oder komplementär verwendbar. Dabei wird, wie in Abbildung 5 gezeigt, ein Schwellwert definiert (in diesem Fall 1.75). Zusätzlich dargestellt ist eine Karenzzeit von 12 Sekunden. Erst nach kontinuierlicher Überschreitung des Schwellwerts von 12 Sekunden wird ein Alarm ausgelöst.

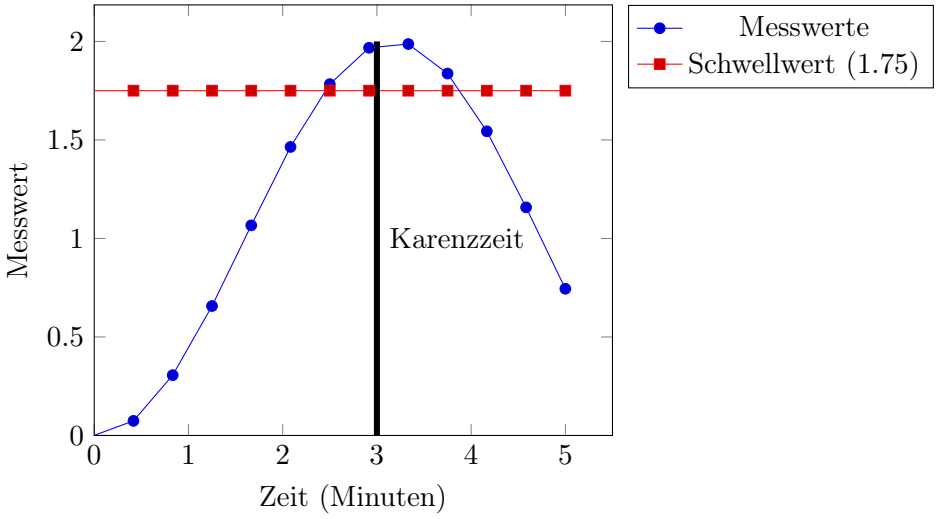


Abb. 5: Schwellwertüberschreitung mit Karenzzeit einer Messreihe

Gleichzeitig ist es aber auch möglich, zählerbasiert einen Alarm/eine Aktion auszulösen. Bei einer Messdistanz von angenommenen 10 Sekunden und dreifacher Auslösung wäre eine 30 sekündige Karenzzeit ebenfalls implementierbar.

#### Trenderkennung/gleitender Durchschnitt

Gleitende Durchschnitte, wie in Abbildung 6 gezeigt, sind eine Form der Kurvenglättung. Sie glätten ausreissende Kurven und ermöglichen einen groben Trend der vorliegenden Daten anhand der Steigung der geglätteten Kurve vorauszusagen.

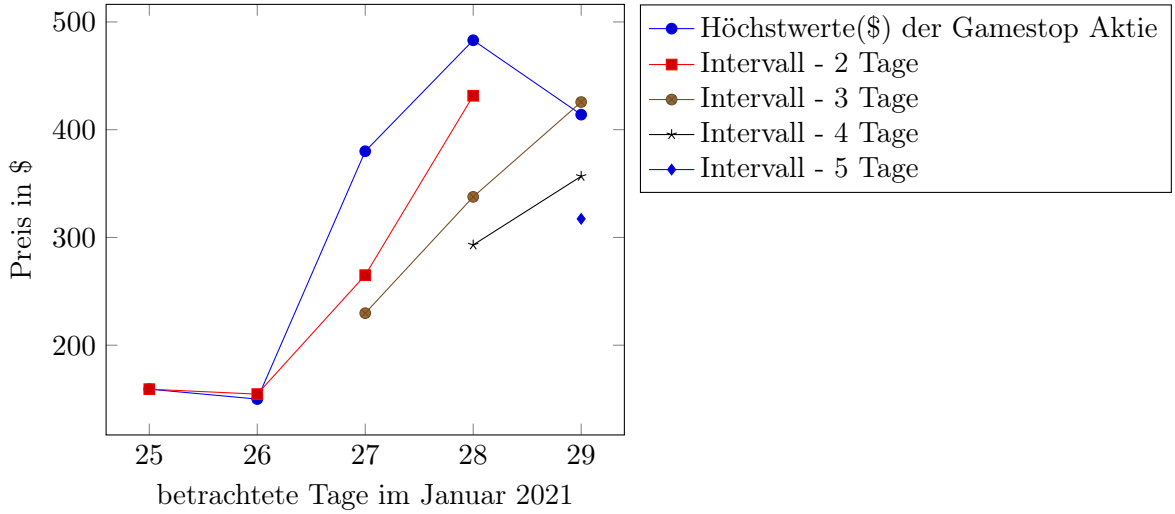


Abb. 6: Gleitender Durchschnitt des Tageshöchstwertes eines Aktienkurses

### 2.1.3 Bestehende Referenzarchitekturkategorien

Im Bereich der Streamingarchitekturen gibt es bereits etablierte Referenzarchitekturen, welche verschiedene mögliche Aufbauarten einer Verarbeitung von Streaming/Zeitreihendaten zeigen.

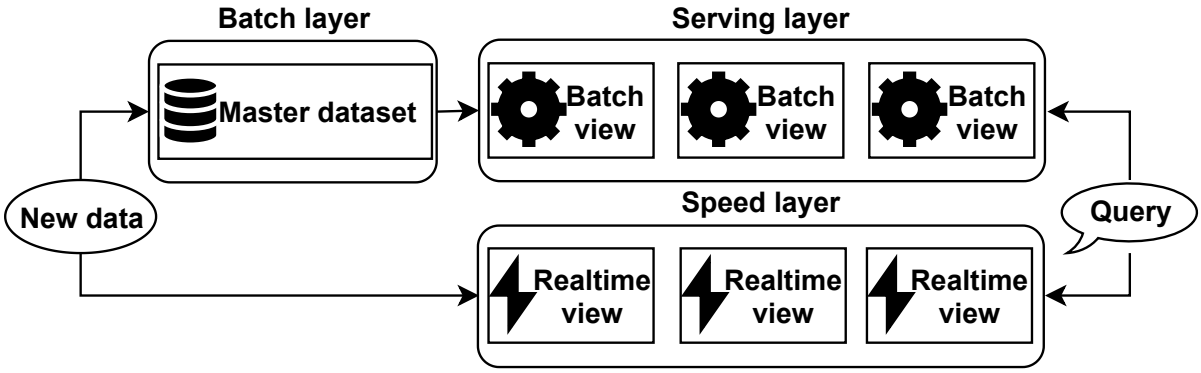


Abb. 7: Die $\lambda$-Datenstreaming Referenzarchitetktur.[13]

Die von Marz/Warren vorgestellte $\lambda$/Lambda-Architektur, welche in Abbildung 7 gezeigt wird, ist dabei eine der sehr bekannten Referenzarchitekturen. Der Name ist dabei nicht mit dem AWS Dienst Lambda zu verwechseln, sondern ist wohl auf den gedrehten Buchstaben $\lambda$ zurückzuführen, also $\prec$.[14] Die $\lambda$-Architektur sieht, ausgehend von den hereingeladenen Daten, zwei verschiedene Wege für die Daten vor. Zum einen den „Speed Layer“, welcher Daten direkt nach dem Eingang verarbeitet und nicht im Layer selbst speichert, sondern nur Aggregate oder Ergebnisse zur Verfügung stellt. Zum anderen gibt es den „Batch layer“, in welchem Daten zuerst in einem Master dataset gespeichert werden und dann in einem festen Intervall („Batch jobs“) ausgewertet werden. Verschiedene Datenverarbeitungsintervalle machen speziell im Sinne der verschiedenen, in Abbildung 2 gezeigten, Datenhalbwertszeiten Sinn. So sind manche Auswertungen, die präzise historische Daten benötigen, in einem Batch layer besser möglich als in einem Speed layer. Der Speed layer bietet dagegen durch die Geschwindigkeit der Auswertungen die Möglichkeit, agil auf erkannte Ereignisse oder Veränderungen im Generellen zu reagieren.

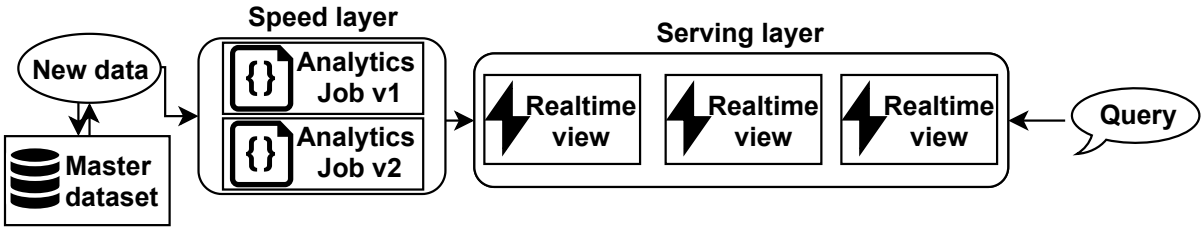


Abb. 8: Die $\kappa$-Datenstreaming Referenzarchitektur.[15]

Die $\kappa$/Kappa Referenzarchitektur von Kreps, dargestellt in Abbildung 8 basiert auf der $\lambda$-Architektur, spart jedoch den „Batch layer“ mit zugehörigen „Batch jobs“ aus. Das Konzept von einem Master Dataset existiert dabei weiterhin, jedoch in Form von Nachrichten, die in einem Messagebroker gespeichert werden. Analysen werden in Form von einzeln versionierten Jobs über die vorhandenen Werte erstellt. Wird die Analyse in irgendeiner Weise verändert (z.B. durch Codeanpassungen) werden alle zwischengespeicherten Nachrichten erneut durch eine neue, unveränderliche Version des Jobs analysiert. Diese Unveränderlichkeit hat den Vorteil, dass keine unerwünschten Seiteneffekte durch Analysen, die gegenseitig Ergebnisse überschreiben, auftreten.

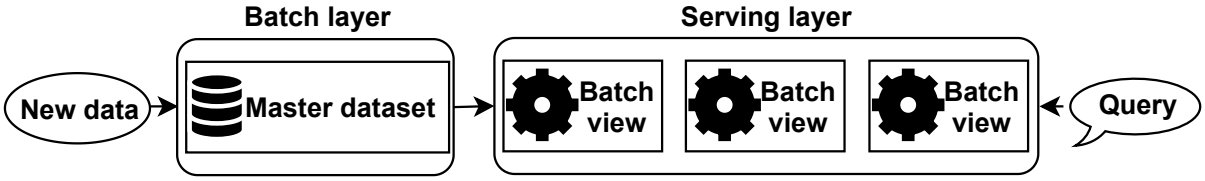


Abb. 9: OLAP Referenzarchitetktur.[16]

[13] Mit Änderungen entnommen aus: Marz/Warren 2015, S. 28
[14] Vgl. auch im Folgenden Berle 2017
[15] Mit Änderungen entnommen aus: Kreps 2014, Berle 2017
[16] Mit Änderungen entnommen aus: Kreps 2014

Aus der $\lambda$ Referenzarchitektur lässt sich auch eine, zur $\kappa$ Architektur gegenteilige Architektur aufzeigen. Diese ist im Stile des von Codd, E. F./Codd, S. B./Salley geprägten Online Analytical Processing (OLAP) gehalten. Diese Architektur basiert, wie in Abbildung 9 gezeigt, auf einer periodischen Verarbeitung der Daten im Master dataset. Dieses Vorgehen ist bei traditioneller Datenanalyse weit verbreitet, bietet jedoch womöglich wichtige Einsichten erst, nachdem die Datenhalbwertszeit bereits überschritten wurde.

### 2.1.4 Echtzeitverarbeitung

Gemäß der in Abbildung 8 gezeigten $\kappa$-Architektur gibt es Nutzungsfälle, in welchen eine reine Echtzeitauswertung basierend auf einer Datenquelle, wie beispielsweise einem Messagebroker, Sinn machen kann. Belur sieht dabei vier verschiedene Verwendungszwecke, in welchen die niedrige Verarbeitungslatenz besonders wichtig ist und den maximalen Wert aus den Daten zieht.[17] Durch Echtzeitreporting und die Erstellung von Dashboards können aktuelle Daten schnell übersichtlich aufbereitet werden. Mittels erstellter Regeln, die Schwellwertüberschreitungen und Anomalien detektieren, können Nutzende benachrichtigt werden, sobald es zu einer Abweichung kommt. Ebenfalls sinnvoll ist der Einsatz von Machine Learning zum Auffinden von Mustern in Daten, was verbesserte Anomalierkennung, Vorhersagen und ähnliche Features ermöglicht. Ein weiterer valider Usecase der $\kappa$-Architektur ist die Transformation von Daten in gewisse Zielformate, um beispielsweise Drittsysteme anzusprechen.

### 2.1.5 Batch Verarbeitung

Gemäß der in Abbildung 9 gezeigten OLAP Architektur, gibt es, wie für die Echtzeitverarbeitung auch nutzende Unternehmen, die ein Entscheidungstempo mit weiterem Horizont haben. Für diese Entscheidungstypen ist dabei wichtig, dass eine Analyse möglichst viele historische Daten umfasst. Es ist dabei aber zweitrangig, wie zeitnah diese erstellt werden kann. Eine gängige Möglichkeit um effizient alte Daten zu analysieren, stellen OLAP Datenbanken dar, welche durch spezielle Indexstrukturen und Optimierungen auf komplexe lesende Abfragen bei großen Datenmengen optimiert wurden.[18] Mittels dieser OLAP Datenbanken lassen sich dank der jeweiligen Abfragesprachen und -dialekte komplizierte Abfragen realisieren. Bekanntes Beispiel für so eine Abfragesprache wäre Structured Query Language (SQL), welche einige verschiedene Implementierungen (Dialekte) hat, die verwendet werden können.

[17] Vgl. auch im Folgenden Belur 2020
[18] Vgl. Codd, E. F./Codd, S. B./Salley 1993, S. 5 f.

## 2.2 Referenzarchitektur

Nach Bass/Clements/Kazman ist eine Referenzarchitektur ein spezialisiertes Referenzmodell, wie in Abbildung 10 gezeigt, welche in Softwarearchitekturen instanziiert werden kann.[19]

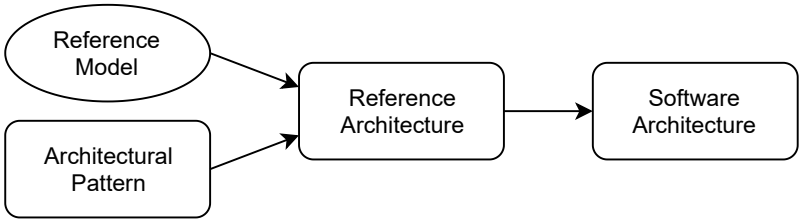


Abb. 10: Beziehungen zwischen Referenzmodellen, Architekturpatterns, Referenzarchitekturen und Softwarearchitekturen.[20]

In diesem Kapitel sollen deshalb die theoretische Definition des Referenzarchitekturbegriffs, genauso wie mögliche Vorgehensmodelle betrachtet werden.

### 2.2.1 Referenzmodelle

Gemäß des konstruktionsprozessorientierten Referenzmodellbegriffs von vom Brocke ist ein Referenzmodell als solches zu erkennen, wenn der Gegenstand und/oder[21] der Inhalt des Referenzmodells bei der Konstruktion des Gegenstandes und/oder des Inhaltes eines zu konstruierenden Anwendungsmodells wiederverwendet werden kann.[22] Dabei hat ein Referenzmodell einen Empfehlungscharakter und stellt eine „best practice“ dar.[23]

Ein Referenzmodell kann nach vom Brocke nicht objektiv allgemeingültig sein und auch keinen objektiven Empfehlungscharakter haben, sondern muss subjektiv beurteilt werden.[24] Dabei ist zumindest von den Interessensgruppen der Konstruierenden und der Nutzenden auszugehen, welche das Referenzmodell subjektiv unterschiedlich nach Allgemeingültigkeit und Empfehlungscharakter bewerten. Je nachdem welche Beurteilung höher gewichtet wird und früher einfließt, kann also entweder von der Situation ausgegangen werden, dass das Referenzmodell vom Konstruierenden zu einem solchen erklärt wird oder ein Modell, ob vom Konstruierenden beabsichtigt oder nicht, von den Nutzenden zu einem solchen erhoben wird.

[19] Vgl. Bass/Clements/Kazman 2010, S. 17 f.
[20] Mit Änderungen entnommen aus: Bass/Clements/Kazman 2010, S. 18
[21] Die Verwendung von „und/oder“ wurde hier gewählt, da der Autor der Quelle das „oder“ aus der boolschen Algebra gewählt hat um explizit beide Fälle einzuschliessen.
[22] Vgl. vom Brocke 2015, S. 34
[23] Vgl. vom Brocke 2015, S. 31
[24] Vgl. auch im Folgenden vom Brocke 2015, 31 f.

### 2.2.2 Referenzarchitektur

**To do** Kapitel ist ziemlich lang (P) (5) Der IEEE Standard 1471-2000 definiert Architektur im Kontext von softwareintensiven Systemen wie folgt: „The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.“[25]. Als softwareintensives System kann jedes System gesehen werden, bei dem Software essentielle Einflüsse auf das Design, die Erstellung, das Deployment oder die Evolution des Systems hat.[26]

Wird dieser Architekturbegriff auf bekannte Bereitstellungsmodi aus der Cloud, wie Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS) oder Function as a service (FaaS) angewendet, wird klar, dass von einer Architektur im Sinne des IEEE Standards 1471-2000 ausgegangen werden kann, sobald Software involviert ist. Im Rahmen dieser Arbeit werden auch Dienste, die sich nach der NIST Cloud Definition unter SaaS Dienste zählen lassen, behandelt. Als SaaS Dienst gilt dabei jeder Dienst, bei dem Nutzende die unterliegende Infrastruktur nicht verwalten und die Applikation nur über limitierte Konfigurationen verwalten können. Sollte aber die Konfiguration mittels einer Programmiersprache bzw. Datenabfragesprache wie SQL erfolgen, ist die Bedingung erfüllt, dass Software wesentliche Einflüsse auf das System hat.

Gallagher definiert eine Referenzarchitektur als eine generalisierte Architektur mehrerer Endsysteme, die eine oder mehrere Domänen teilen.[27] Die Referenzarchitektur definiert nach Sicht des Autors dabei die gemeinsame Infrastruktur der Endsysteme und die Schnitstellen der Komponenten, die in den Endsystemen enthalten sein sollen. Dabei ist eine Referenzarchitektur zu instanziieren, um eine spezifische Softwarearchitektur zu erstellen. Gallagher definiert die Aufgaben einer Referenzarchitektur wie folgt: Zum einen werden übergreifende Funktionen und Konfigurationen generalisiert und extrahiert und zum anderen wird eine kosteneffiziente und verlässliche Basis geschaffen, um Zielsysteme abzuleiten/zu instanziieren.[28]

Trefke schränkt in seiner Definition die Instanziierung insoweit ein, dass individuelle Besonderheiten abstrahiert werden müssen, um eine Allgemeingültigkeit der Referenzarchitektur in einer speziellen Domäne zu erhalten.[29] Zusätzlich fügt Trefke der Referenzarchitektur als optionale Aufgaben die Definition von Leitlinien für die Verwendung, Evolution und Verantwortlichkeiten hinzu. Zurückgreifend auf vom Brocke legt Trefke fest, dass eine Referenzarchitektur als spezifischeres Referenzmodell seinen Empfehlungscharakter entweder durch Erfahrungen und hohe Nutzerakzeptanz oder durch Festsetzung von Erschaffenden erhält.

Nach Angelov/Grefen/Greefhorst gibt es zwei Typen und damit verbundene Zielsetzungen der Referenzarchitektur: Die standardisierende Referenzarchitektur, welche darauf zielt eine Stan-

[25] IEEE 2000, S. 3
[26] Vgl. IEEE 2000, S. 1
[27] Vgl. auch im Folgenden Gallagher 2000, S. 3
[28] Vgl. Gallagher 2000, S. 3
[29] Vgl. auch im Folgenden Trefke 2012

dardarchitektur für spezielle Anwendungsfälle zu schaffen und die unterstützende/erleichternde Referenzarchitektur, welche Personen in Architekturrollen unterstützen sollen, ähnliche Probleme leichter zu lösen.[30] Nach Angelov/Grefen/Greefhorst sind standardisierende Referenzarchitekturen nicht zur Verwendung von innovativen, also kaum getesteten oder noch nicht von Experten akzeptierten Elementen geeignet. Die unterstützenden/erleichternden Referenzarchitekturen hingegen können solche innovativen Elemente durchaus verwenden und auch eine Technologievorauswahl treffen.

Um einen möglichst hohen Nutzen stiften zu können, müssen die organisatorischen Rahmenbedingungen, in welchen ein Referenzmodell eingesetzt werden soll, analysiert werden.[31]

Muller empfiehlt, eine Referenzarchitektur zur Generalisierung von vorhandenen Architekturen zu verwenden.[32] Für neue Technologien und Applikationen, die bislang in der Form kaum verwendet wurden, schlägt Muller stattdessen ein inkrementelles Vorgehen vor. Das inkrementelle Vorgehen von Muller beinhaltet dabei die Erstellung von Prototypen und Einholung von Feedback der Zielstakeholder.

### 2.2.3 Diskussion der Qualitätskriterien der Referenzarchitekturen

Muller schlägt sieben Qualitätskriterien vor, welche von einer guten Referenzarchitektur erfüllt werden sollten:[33]

1. Verständlichkeit für eine breite, heterogene Gruppe an Stakeholdern (Kunden, Projektmanager, Entwickler, etc.)
2. Zugänglichkeit und Zugriff durch die Mehrheit der Organisation
3. Adressierung der Hauptprobleme der spezifischen Problemdomäne
4. Zufriedenstellende Qualität
5. akzeptabel
6. „up-to-date“ und wartbar
7. wertschöpfend für den Betrieb

AWS definiert im Rahmen des sogenannten Well Architected Frameworks für verschiedene „Lenses“, also Spezialisierungen, Themenfelder, die bei exzellenten Architekturen zu beachten sind. Für Datenanalysen gibt es die Analytics Lens, welche entsprechend für Referenzarchitekturen genauso Anwendung finden sollte. Kriterien sind dabei die Folgenden:[34]:

[30] Vgl. auch im Folgenden Angelov/Grefen/Greefhorst 2012, S. 422 ff.
[31] Vgl. vom Brocke/Buddendick 2004
[32] Vgl. auch im Folgenden Muller 2020, S. 7
[33] Vgl. auch im Folgenden Muller 2020, S. 8
[34] Vgl. Ravirala u. a. 2020, S. 6

1. Automate data ingestion
2. Design ingestion for failures and duplicates => Siehe Robustness and fault tolerance
3. Preserve original source data
4. Describe data with metadata
5. Establish data lineage
6. Use the right ETL tool for the job
7. Orchestrate ETL workflows
8. Tier storage appropriately
9. Secure, protect, and manage your entire analytics pipeline
10. Design for scalable and reliable analytics pipelines => Scalability, robustness and fault tolerance

**To do** In Text einbetten, Erklärung welche passen (6)

### 2.2.4 Vorgehensmodell

Schütte unterteilt Referenzmodellierung generell in vier Phasen.[35] Die erste Phase, die Problemdefinition ist mit der Darstellung des behandelten Problems in der Einleitung dieser Arbeit bereits vorgenommen worden. Laut Schütte kann der Wirklichkeitszugang des erstellten Modells nur über Bilder erfolgen, welche entsprechend zu modellieren sind.[36] Da momentan keine Erfahrungen in den zu verwendenden Technologien für die Referenzmodellierung vorliegt, handelt es sich um eine Top-down Referenzmodellierung/Referenzarchitektur. Bei der Konstruktion des Referenzmodellrahmens wird durch das „Was“ motiviert, welche Unternehmensspezifika zu beachten sind.[37] In der vorliegenden Arbeit wäre beispielsweise Teil des Referenzmodellrahmens, dass cloudbasiert gearbeitet werden soll und dass die Zeitreihendaten vorerst vornehmlich von IoT Anwendungsfällen stammen, was sich später dennoch ändern könnte.[38] In der „Wie“ Phase, die dieses Kapitel behandeln soll, wird die Referenzarchitektur strukturiert und Darstellungsarten gezeigt, welche folgend angewendet werden können.

[35] Vgl. auch im Folgenden Schütte 1998, S. 184 f.
[36] Vgl. auch im Folgenden Schütte 1998, S. 185 f.
[37] Vgl. auch im Folgenden Schütte 1998, S. 186
[38] Vgl. auch im Folgenden Schütte 1998, S. 187 f.

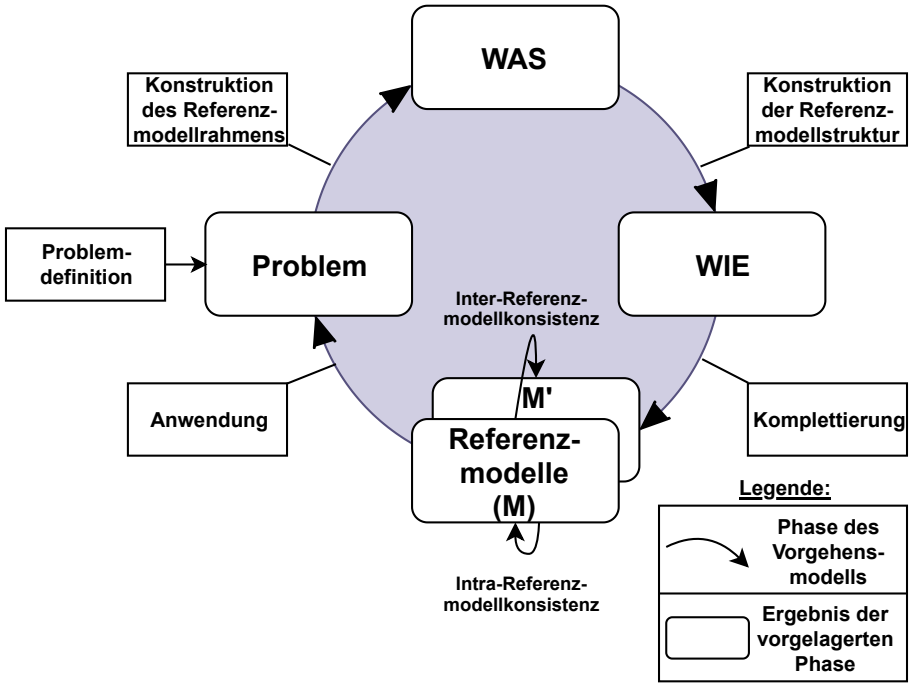


Abb. 11: Vorgehensmodell Referenzmodellierung nach Schütte.[39]

Nach Muller hat eine Referenzarchitektur mehrere Dekompositionen, in beispielsweise eine funktionale, eine konstruktionsorientierte oder eine infrastrukturorientierte Komposition.[40] Diese Dekompositionsschichten lassen sich ebenfalls in bekannten Architekturframeworks wie arc42 finden, weshalb diese mit Anpassungen zur Visualisierung dienen sollen. Dies deckt sich auch mit der Auffassung von Schütte zur Referenzmodellierung, welcher Modellierung über Bilder erfolgen lassen möchte.[41]

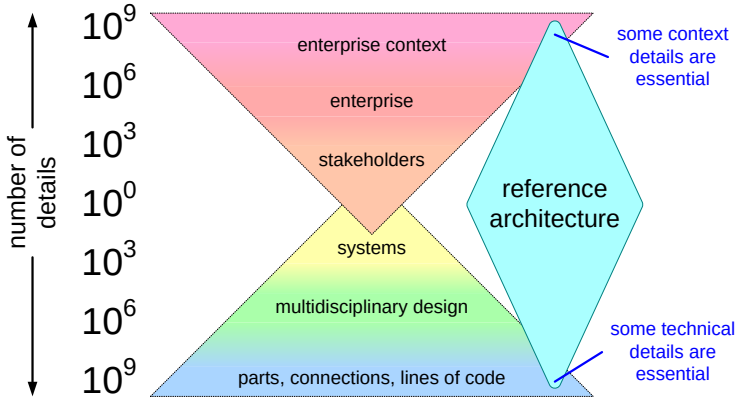


Abb. 12: Gewünschter Detailgrad von Referenzarchitekturen nach Muller.[42]

[39] Mit Änderungen entnommen aus: Schütte 1998, S. 185
[40] Vgl. Muller 2020, S. 7
[41] Vgl. Schütte 1998, S. 185
[42] Entnommen aus: Muller 2020, S. 11

Mehrere Dekompositionen machen insbesondere auch Sinn, da wie im Diagramm von Muller - Abbildung 12 - eine Addressierung von unterschiedlichen Aspekten wie Systemgestaltung, aber auch Stakeholder oder Kontext erfolgen sollte. Eine Adressierung der Stakeholder und des Kontextes ist insofern gegeben, dass, wie in Abschnitt 2.3 gezeigt, Interviews durchgeführt und im Anhang dieser Arbeit transkribiert werden. An Stellen, an denen ein Einsatz von Teilen der Referenzarchitektur nontrivial scheint, kann durch Codebeispiele oder konkrete, kopierbare „Schnipsel“ gezeigt werden, auf was speziell im technischen Bereich zu achten ist.

Zusätzlich sollen Dekompositionen unterschiedliche Aufgaben erfüllen. Speziell für die Darstellung der verwendeten Diensten von AWS in den Referenzarchitekturen und dem Datenfluss soll die erste Stufe der Bausteinsicht des Architekturstandards arc42 verwendet werden. Das Konzept jener Bausteinsicht, also wie sie zu gestalten ist, findet sich in Abbildung 13. In der finalen Umsetzung werden die offiziellen AWS Icons die entsprechenden Services darstellen, die verwendet werden.

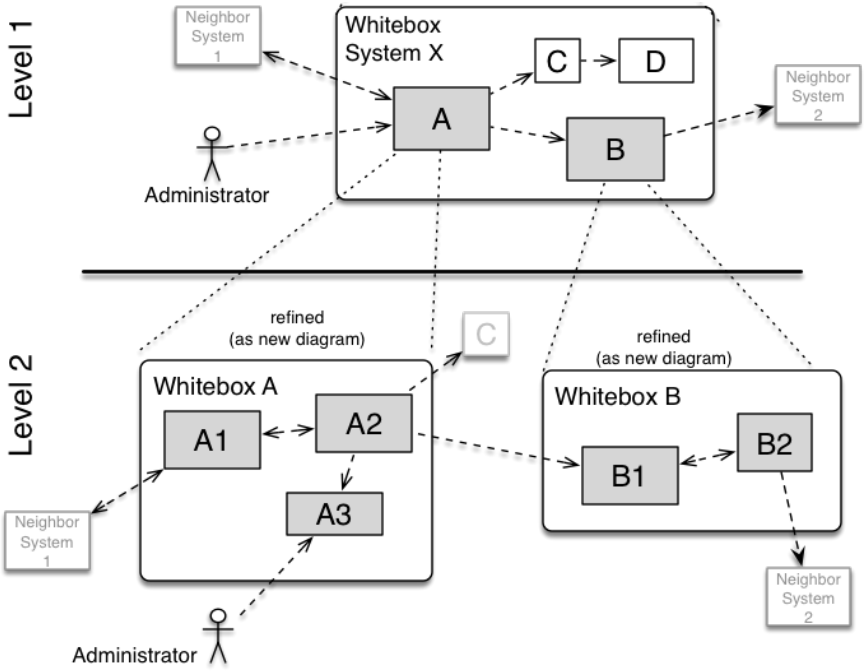


Abb. 13: Stufe 1 der Bausteinsicht in arc42.[43]

Zusätzlich zu der Bausteinsicht können, wie in Abbildung 14 geizeigt, weitere Diagrammtypen eingesetzt werden, um verschiedene Dekompositionen darstellen zu können.

[43]Mit Änderungen entnommen aus: Starke o.J.

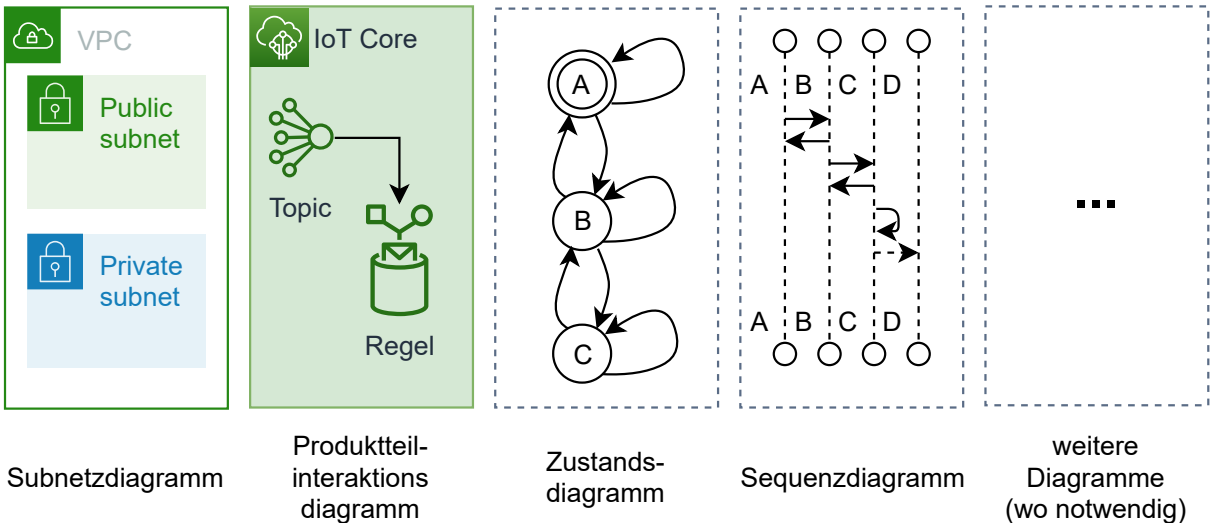


Abb. 14: Ergänzende Dekompositionen

Wie von Muller vorgeschlagen und im vorherigen Unterkapitel erläutert, ist ein inkrementeller Ansatz unter Verwendung von Prototypen und kontinuierlichem Feedback der Zielstakeholder unerlässlich.[44]

Sehr wichtig für eine Referenzarchitektur ist auch die Dokumentation, wie die Wiederverwendung zu handhaben ist. Ein möglicher Ansatz wäre dabei die gezielte Integration und Dokumentation von Variationspunkten, wie von Webber vorgeschlagen.[45] Mittels der Variationspunkte kann eine statische Referenzarchitektur konstruiert werden, welche an spezifisch definierten Punkten angepasst werden muss, um einzigartige Architekturen zu erzeugen.[46] Dabei gibt es vier verschiedene Ansichten, aus denen Variationspunkte definiert werden können:[47]

1. Requirement-Variation-Point View
2. Component-Variation-Point View
3. Static-Variation-Point View
4. Dynamic-Variation-Point View

Dabei sind für diese Arbeit, in der keine implementierungsnahe (im Sinne von Programmierung) Softwarearchitektur entworfen wird, hauptsächlich die anforderungsbasierte und die komponentenbasierte Variationspunktsicht aus Punkt 1 und Punkt 2 wichtig. Die statische und dynamische Variationspunktsicht aus Punkt 3 und Punkt 4 agieren stärker auf Implementierungsebene.[48] Auf dieser können beispielsweise mittels objektorientierter Programmierung Klassen bereitgestellt

[44] Vgl. Muller 2020, S. 7
[45] Vgl. Webber 2001, S. 24 ff.
[46] Vgl. Webber 2001, S. 24
[47] Vgl. Webber 2001, S. 25 f.
[48] Vgl. auch im Folgenden Webber 2001, S. 25 f.

werden, von welchen geerbt werden kann. Gleichzeitig kann die Verhaltensweise des Programms auch durch z.B. Callbacks oder Parameterisierung des Aufrufes verändert werden.

Variationspunkte können, wie in Abbildung 15 dargestellt innerhalb der verschiedenen, dargestellten Schichten wie folgt dargestellt werden:


Abb. 15: Darstellung Variationspunkte

In den Architekturebenen werden entsprechend die Variationspunkte mit alphanumerischen Identifikationen versehen. So können gleiche Variationspunkte in mehreren Dekompositionen referenziert werden.

## 2.3 Theorie der Anforderungserhebung

Gemäß der in Unterabschnitt 2.2.3 definierten Qualitätskriterien und der von vom Brocke aufgestellten Allgemeingültigkeitskriterien kann ein Referenzmodell und damit auch eine Referenzarchitektur nicht objektiv allgemeingültig sein. Zusätzlich ergeben sich als weitere wichtige Eingaben zur Konstruktion eines Referenzmodells die Dekompositionstiefe und die Anwendbarkeit. Zusammen lassen sich diese Dimensionen als Kiviat Diagramm,[49] wie in Abbildung 16 gezeigt, abbilden. Die Dekompositionstiefe misst die Anzahl an Schichten und die Detailtiefe der verschiedenen Dekompositionsschichten die im Rahmen der Architektur verwendet werden. Ein niedriger Wert entspricht einer geringen Anzahl, welche keine große Detailtiefe aufweisen. In Abbildung 14 sind mögliche Dekompositionen mit unterschiedlicher Tiefe gezeigt. Die Anwendbarkeit, als Gegensatz zur Abstraktion misst, wie gut und schnell sich eine Referenzarchitektur instanziieren lässt, um eine spezifische Architektur abzuleiten. Ein niedriger Wert bedeutet eine sehr abstrakte Referenzarchietktur, deren Instanziierung mit viel Aufwand verbunden ist. Die Allgemeingültigkeit beschreibt die Menge an Anwendungsfällen und die Losgelöstheit von Firmenspezifika beziehungsweise die übergreifende Gültigkeit in mehreren Teams. Idealerweise ist eine Referenzarchitektur spezifisch für eine Zielorganisation, aber nicht zu spezifisch, sondern erlaubt Anwendungen in anderen Teams.

[49] Vgl. Kolence 1973, S. 33 ff.

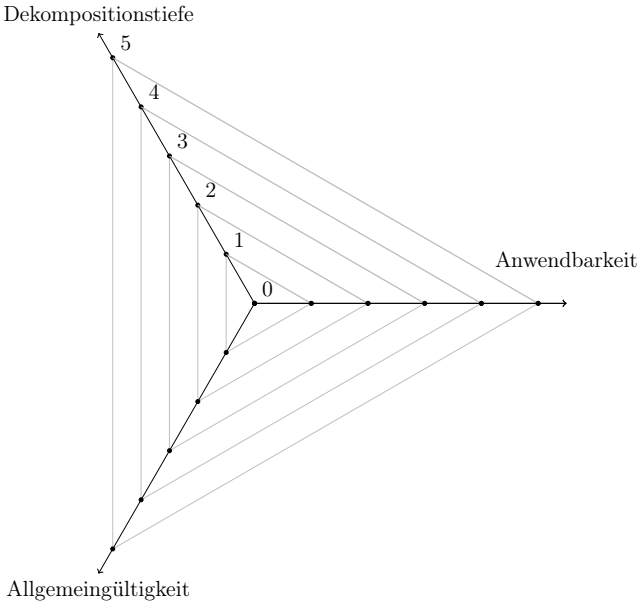


Abb. 16: Referenzarchitekturdimensionen

Ziel einer Referenzarchitektur sollte also sein, das optimale Verhältnis der drei Dimensionen für die Zielstakeholder zu finden und daraus eine Referenzarchitektur zu erstellen. Um dieses Ziel zu erreichen, sind Interviews mit Zielstakeholdern der SPIRIT/21 zu führen, welche dem Interviewleitfaden in Tabelle 1 folgen. Im referenzierten Leitfaden stehen dabei Fragen mit einem F-Präfix für allgemeine Fragen und Fragen mit einem D-Präfix für Fragen in Bezug auf die drei Dimensionen. Kombiniert mit mindestens einem Review der Referenzmodelle werden die Modelle zu einem nutzenstiftenden Artefakt.

| ID | Beschreibung |
|---|---|
| | Allgemeine Fragen |
| F1 | Rolle innerhalb der SPIRIT/21 |
| F2 | Anwendungsgebiete der Referenzarchitekturen |
| F3 | Kompatibilität der Referenzarchitekturen zueinander? |
| | Priorisierungen |
| P1 | Priorisierung der Qualitätskriterien (siehe Unterabschnitt 2.2.3) |
| P2 | Priorisierung der Datennutzungstypen (siehe Abschnitt 2.1) |
| | Dimensionen der Referenzarchitekturen |
| D1 | Anforderungen an Anwendung der Referenzarchitekturen |
| D2 | Anforderungen an Allgemeingültigkeit der Referenzarchitekturen |
| D3 | Dekompositionstiefe der Referenzarchitekturen |

Tab. 1: Interviewleitfaden für Schlüsselstakeholder

## 2.4 Vergleichsmethodik für die Dienstauswahl

Marz/Warren, die bereits die λ-Architektur geprägt haben, haben folgende,erwünschte Eigenschaften eines Big Data Systems festgelegt [*englisches Original geklammert*]:[50]

1. Robustheit und Fehlertoleranz [*Robustness and fault tolerance*] - Systeme sollen Herausforderungen, wie beispielsweise Paralellität, Datenduplikate oder technische Ausfälle verkraften. Zusätzlich ist Resilienz gegenüber menschlichen Fehlern wünschenswert, so dass händische Änderungen rückgängig gemacht werden können (also, dass beispielsweise Analysecode „immutable“ ist).

2. Lese- und Schreibzugriffe mit niedriger Latenz [*Low latency reads and updates*] - Lesezugriffe auf Daten sollen mit niedriger Latenz stattfinden. Wie bereits beschrieben, kann aufgrund der Messdistanz eine Aktualisierung von Daten durchaus längere Zeit benötigen, jedoch sollte ein Big Data System in der Lage sein, Datenaktualisierungen mit niedriger Latenz durchzuführen.

3. Skalierbarkeit [*Scalability*] - Das Big Data System sollte durch transparente oder intransparente Provisionierung weiterer Ressourcen in der Lage sein, gleiche Performance in verschiedenen Belastungssituationen zu liefern. Dies deckt sich mit einem der Kernversprechen der Public Clouds nach NIST Definition [*rapid elasticity*].[51]

4. Generalisierung [*Generalization*] - Ein Big Data System sollte in der Lage sein, verschiedene Anwendungen zu unterstützen. Da die Zielsetzung dieser Bachelorarbeit auf Zeitreihendaten aufbaut, welche wie in Abschnitt 2.1 gezeigt, einen großen Einsatzspielraum haben, ist diese Bedingung bei ausreichender Generalisierung der Referenzarchitekturen erfüllt.

5. Erweiterbarkeit [*Extensibility*] - Das zu gestaltende Big Data System soll erweiterbar sein und neue Funktionen oder Änderungen ohne größeren Aufwand ermöglichen.

6. Sofortige Abfrage [*Ad hoc queries*] - Diverseste Abfragen sollen schnellstmöglich auf dem Datensatz der Big Data Anwendung möglich sein.

7. geringer Wartungsaufwand [*Minimal maintenance*] - Eine Big data Anwendung soll wartbar bleiben, indem Komplexität in den Kernkomponenten, welche nach Ansicht von Marz/ Warren zu erhöhtem Wartungsaufwand führt, möglichst gering ist.

8. Fehlertransparenz [*Debuggability*] - Innerhalb eines Big Data Systems soll es möglich sein, nachzuverfolgen, wie Werte entstanden sind, um mögliche Fehler verfolgen zu können.

Um die folgenden Vergleiche aggregieren zu können und die Vergleichskiterien zu priorisieren, ist eine Umfrage durchzuführen. Diese Umfrage soll mindestens alle Stakeholder, die bereits interviewt wurden umfassen, könnte aber auch zusätzliches technisches Personal des Native Cloud Solutions (NCS) Teams umfassen. **To do** deutlich ausbauen (7)

[50] Vgl. Marz/Warren 2015, S. 7 ff.
[51] Vgl. Mell/Grance 2011, S. 2

### 2.4.1 Features des Dienstes

Es soll auf die Mindestverfügbarkeit folgender Fähigkeiten überprüft werden:

- Auswertungen nach Unterabschnitt 2.1.2

**To do** ausbauen (8)

### 2.4.2 Performancegarantien

Es sind die Angaben des Herstellers zu bewerten und eventuelle Nutzungsfälle aufzulisten, bei denen der Dienst erfolgreich bei erhöhten Anforderungen (großes Datenvolumen, großer Durchsatz, ...) eingesetzt wurde.

### 2.4.3 Gesamtkosten

Um einen sinnvollen Kostenvergleich aufzustellen, sind folgende Annahmen zu treffen:

- Es existieren 200 Geräte/Sensoren. Es geht eine Nachricht mit einem kB pro Minute pro Gerät ein (0,0432 GB/Gerät/Monat und 8,64 GB/Monat).
- Es ist eine Vergleichsoperation auf einen Schwellwert auszuführen und, wo möglich, eine Zählung aller Schwellwertüberschreitungen der letzten drei Monate durchzuführen (historische Daten also mindestens 25,92 GB).
- Es ist die AWS-Region Frankfurt (eu-central-1) mit Abrechnungswährung US-Dollar (Umrechnung in € erfolgt bei AWS bei Abrechnung) zu wählen. Alternativ ist die Region Irland (eu-west-1) bei Nichtverfügbarkeit der Diensleistung in Frankfurt zu wählen.
- Dienste, die diese Analyse alleine nicht bewerkstelligen können, müssen unter zusätzlicher Verwendung von Rechendiensten wie Lambda oder Elastic Compute Cloud (EC2) angesetzt werden mit permanentem Speicher im Simple Storage Service (S3).
- Analysen, die individuell auslösbar sind, erfolgen alle 10 Minuten an Werktagen zwischen 9 und 17 Uhr, also monatlich 960mal.
- Es wird angenommen, dass der Schwellwert 5 mal pro Gerät pro Monat überschritten wird (1000 Überschreitungen insgesamt).
- Für die Zwischenspeicherung in S3, wenn benötigt, wird folgendes Datenschema angenommen:

```
[
    {
        "deviceId": "Sensor-1",
        "timestamp": "2021-01-01T01:00:00.000Z",
        "value": 695
    }
]
```

Codeausschnitt 1: Beispiel JavaScript Object Notation (JSON)

Um Historien über 3 Monate bereitzustellen, sind entsprechend 3000 Einträge nötig, was eine Dateigröße von  455,32 KB ergibt. Das Berechnungsskript ist im Anhang 5 abgedruckt.

| Dimension | Preis/Einheit | Summe |
|---|---|---|
| Beispiel | x$/100.000 Datenpunkte | x$ |

Tab. 2: Kostenvergleich Schema

Alle Abrechnungsdimensionen werden in der in Tabelle 2 gezeigten Form dokumentiert.

# 3 Anwendungsfälle

In diesem Kapitel werden die bestehenden Anwendungsfälle erläutert, welche innerhalb der SPIRIT/21 Daten zur Analyse liefern können.

## 3.1 Rahmenbedingungen der Datenverabeitung

Aufgrund bereits getroffener Architekturentscheidungen durch das IoT Team der SPIRIT/21 GmbH wird für die Kommunikation zwischen Geräten und dem verarbeitenden Backend das Message Queuing Telemetry Transport (MQTT) Protokoll verwendet. MQTT ist nach eigener Aussage ein extrem leichtgewichtiges Standardprotokoll für publish-/subscribe-basierten Nachrichtentransport.[52] Und ist nach Analystenmeinung der de-facto Standard für IoT Kommunikation.[53,54]

In nachfolgenden Anwendungsfällen wird angenommen, dass eine technologische Vorauswahl für unterstützende AWS Dienste erfolgt ist. So könnte beispielsweise ein beliebiger, MQTT kompatibler Messagebroker eingesetzt werden, es wird jedoch davon ausgegangen, dass der eingesetzte Message Broker aus Kompatibilitätsgründen zu den anderen AWS Diensten IoT Core ist. Zusätzlich ist von einem Sendeintervall von 5 Minuten für die Sensoren auszugehen, welches gesetzt wurde, um eine hohe Akkulebensdauer zu ermöglichen. Diese Einschränkung trifft aber nicht auf Abschnitt 3.3 zu.

## 3.2 Raumklimamonitoring

Innerhalb des Hauptsitzes der SPIRIT/21 in Böblingen wurden im Rahmen der Covid-19 Bekämpfung Raumklimasensoren in allen Besprechungsräumen installiert, welche kontinuierlich die $CO_2$-Konzentration der Raumluft messen. Die $CO_2$-Konzentration dient dabei laut aktueller Studienlage als Messhilfe für die Konzentration von möglichen Virenaerosolen im Raum und zur Lüftungsindikation, wenn ein kritischer Wert überschritten wird.[55,56] Zur Messung wurden die „ERS-$CO_2$“ Sensoren von Elsys in den Besprechungsräumen installiert. Zum Zeitpunkt der Fertistellung dieser Arbeit existieren 8 Sensoren in Besprechungsräumen in Böblingen. Die Daten werden mittels dem Funkstandard Long Range Wide Area Network (LoRaWAN) an ein Gateway gesendet, welches die Daten mittels MQTT an ein Backend, wie beispielsweise die Open Source Low-Code Plattform Node-RED übermittelt.

[52] Vgl. o.V. 2020
[53] Vgl. Skerrett 2019
[54] Vgl. Cabé 2018
[55] Vgl. Hartmann/Kriegel 2020
[56] Vgl. Peng/Jimenez 2020

| Attribut | Datentyp | Einheit |
|---|---|---|
| deviceName | String | - |
| temperature | Double | °C |
| humidity | Integer | % |
| light | Integer | Lux |
| motion | Integer | Anzahl Bewegungen |
| $co_2$ | Integer | ppm |
| $v_{dd}$ (Spannung) | Integer | mV |

Tab. 3: Datenschema Elsys ERS $CO_2$ Sensor.[57]

**To do** Notwendig?? (9) Nach Übermittlung in das Backend haben die Daten das in Tabelle 3 gezeigte Format. Auf diesem normalisierten Format, welches dann in JSON-Syntax übergeben wird, können diverse Analysen zur $CO_2$ Konzentration in den Räumen, Belegung und ähnlichem durchgeführt werden. Das Datenübermittlungsintervall der Geräte beträgt 3 Minuten, wobei jedoch anzumerken ist, dass die Geräte nicht synchron alle 3 Minuten Daten senden, sondern jeweils einen eigenen Senderythmus haben.

## 3.3 Sensor Simulator

Mithilfe von Node-RED und der Erweiterung (in der Plattform auch „Knoten“ genannt) „iot-device-simulator-1-mqtt“ des Github Nutzenden phyunsj[58] kann ein IoT Sensor simuliert werden, welcher in beliebiger Frequenz verschiedene Werte in festgelegten Bereichen generiert. Diese Werte bieten bei höherer Frequenz einen Anhaltspunkt, wie sich die Lösungen unter Last verhalten.

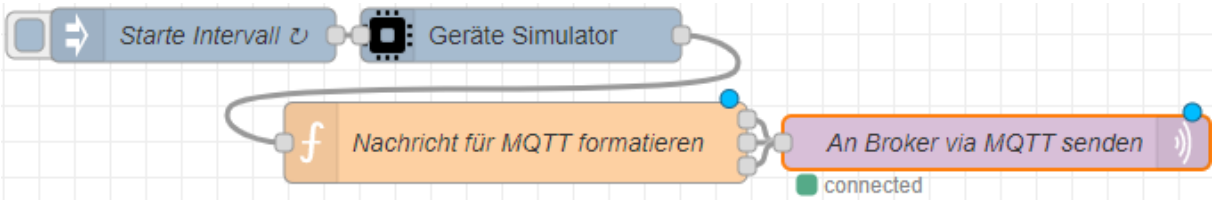


Abb. 17: Node-RED Flow des Sensorsimulators

In Abbildung 17 ist ein Screenshot des Node-RED Flows zu sehen, in dem die verschiedenen Knoten, die jeweils eigene Aufgaben übernehmen, gezeigt sind. Da der Gerätesimulator die Nachrichten falsch formattiert übergibt, ist es notwendig, eigene Logik zur Reformattierung für den Transfer mit MQTT einzusetzen. Diese ist mit einem $f$ gekennzeichnet.

[57] Mit Änderungen entnommen aus: Elektroniksystem i Umeå AB 2019
[58] Siehe auch `https://github.com/phyunsj/iot-device-simulator-1-mqtt`

# 4 Dienstauswahl

Im nachfolgenden Kapitel werden mittels der in Abschnitt 2.4 vorgestellten Methodik die Dienste für die finalen Referenzarchitekturen unter den in Abschnitt 3.1 vorgestellten Rahmenbedingungen verglichen und ausgewählt.

## 4.1 Angewandte Methodik

Grundlage der hier gezeigten Bewertungskriterien sind die in Abschnitt 2.4 gezeigten Kriterien von Marz/Warren, die ISO 9126 Norm, die im Interview in Anhang 3 eingebracht wurde und das Kriterium „serverlessness“, welches nicht die volle Punktzahl erreichen kann, wenn Skalierungsoperationen nicht automatisch durchgeführt werden können. Bei den Kriterien von Marz/ Warren wurde Ad hoc queries gestrichen, da manche Dienste nur zeitlich vorgeplante Auswertungen durchführen können, oder die Fähigkeit ausserhalb des eigenen Ausführungsplans Anfragen zu bearbeiten nicht besitzen. Ebenfalls wurde die Nachverfolgbarkeit als Kriterium ausgenommen, da ein geringer Wartungsaufwand im Wartungsfall diese bedingt. Die Performancegarantien werden genau wie die Kosten, welche ein eigenes Kriterium sind, nach eigener, in Abschnitt 2.4 beschriebener Methodik evaluiert. Aus diesem Grund wurde „Lese- und Schreibzugriffe mit niedriger Latenz“ in Performancegarantien umbenannt. Ausgehend von dieser Prioritätenliste wurde eine Umfrage zur Priorisierung durchgeführt, welche die final gewichtete Prioritätenliste ergibt. Diese Umfrage befindet sich mit Ergebnissen in Anhang 6. Insgesamt sind die Kriterien nach Priorität absteigend geordnet, welche gleichzeitig die Summe an Punkten darstellt, die maximal für dieses Kriterium zulässig sind.

| Kriterium | max. Punkte | Beispiel 1 | Beispiel 2 |
|---|---|---|---|
| Übertragbarkeit zwischen Clouds (ISO 9126) | 1 | 0 | 0 |
| Integration mit anderen AWS Diensleistungen | 3 | 0 | 0 |
| Generalisierung | 4 | 0 | 0 |
| Erweiterbarkeit | 4 | 0 | 0 |
| Fehlertransparenz/ Debugability | 5 | 0 | 0 |
| geringer Wartungsaufwand | 7 | 0 | 0 |
| Skalierbarkeit & „serverlessness“ | 7 | 0 | 0 |
| Kosten | 7 | 0 | 0 |
| Performancegarantien | 8 | 0 | 0 |
| Robustheit & Fehlertoleranz | 9 | 0 | 0 |
| Auswertungen (Unterabschnitt 2.1.2) | 11 | 0 | 0 |
| Summe | 66 | **0** | **0** |

Tab. 4: Bewertungsmatrix Beispiel

Die Ergebnisse der Bewertung werden, wie in Tabelle 4 gezeigt, dargestellt. Da innerhalb der Kategorien ähnliche Funktionsweisen vorliegen können, die zu einer Abwertung führen, wird eine solche generelle Einschränkung für die Kategorie zur Einleitung vorgenommen.

Ausgehend von der Gesamtsumme der Punkte kann eine Rangliste erstellt werden, welche Dienstleistungen für die Referenzarchitektur verwendet werden können.

## 4.2 Dienste für Echtzeitverarbeitung

In Abbildung 18 werden verwendbare Dienste von AWS, gemeinsam mit ihren jeweiligen Einsatzgebieten gezeigt. In diesem Abschnitt soll besonders auf die Dienste zum Datenstreaming und zur Datenverarbeitung eingegangen werden. Gezeigt werden jedoch auch Dienste für -Speicherung, -Visualisierung und Machine Learning, da diese komplementär oder mit den prozessierten Daten verwendet werden können.

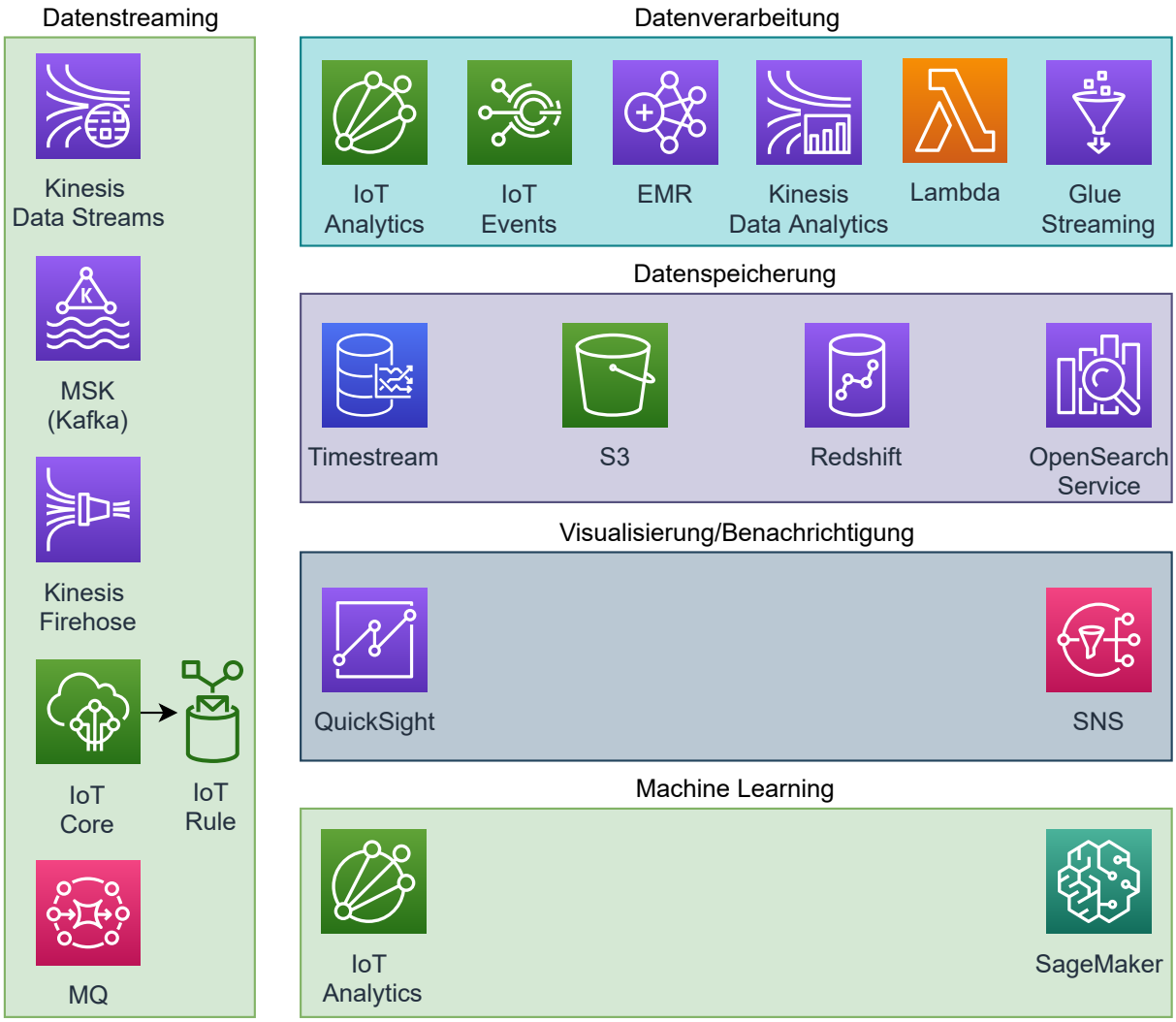


Abb. 18: Einsetzbare Dienste im Bereich Echtzeitverarbeitung

**To do** Hier noch besserer Übergang (10)

Innerhalb des IoT Core Brokers ist es möglich, Regeln zu definieren, die einzelne Nachrichten aus Topics in andere Dienste weiterzuleiten. Dazu müssen besagte Nachrichten selektiert werden, was mittels eines SQL Dialekts möglich ist. Eine beispielhafte Selektion könnte folgendermaßen aussehen: `SELECT * FROM 'iot-demo-sensor' WHERE device <> 'test'` Alle Attribute jeder Nachricht, die Nicht vom Gerät test stammen, aus dem Topic iot-demo-sensor werden dabei selektiert und können dann z.B. weitergeleitet werden.

### 4.2.1 AWS IoT Events

Im Rahmen der AWS IoT Familie von Diensten gibt es neben dem bereits angesprochenen AWS IoT Analytics ebenfalls AWS IoT Events. Der Dienst dient nach Angaben von Amazon der Konfiguration von „If-Then-Else“ Regeln, mit denen Ereignisse (also Events) erkannt und

verarbeitet werden sollen, indem Aktionen ausgelöst werden.[59] Die Abläufe können dabei, ähnlich wie bei Node-RED graphisch konfiguriert werden. Unterstützte Aktionen von AWS IoT Events sind beispielsweise der Aufruf von Lambda oder Benachrichtigung via Simple Notification Service (SNS).[60]

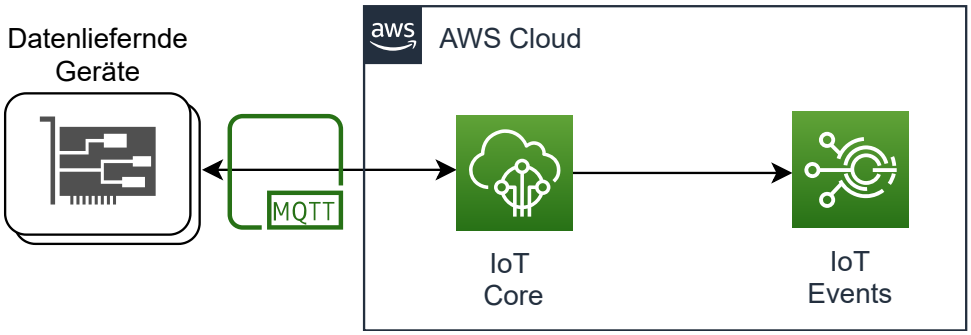


Abb. 19: Grobarchitektur des Ablaufes für IoT Events

AWS IoT Events basiert auf abgebildeten Zuständen, die basierend auf ihren Übergängen Aktionen auslösen. Abbildung 20 zeigt 3 beispielhaft 3 definierte Zustandsübergänge in der Weboberfläche von AWS IoT Events. Jeder Kreis, welcher einen Zustand simbolisiert, hat 3 eigene Ereignisse, nämlich OnEnter, OnInput und OnExit. Zusätzlich sind Zustände mit Zustandsübergangspfeilen verbunden, welche basierend auf einer Ausführungskondition ausgelöst werden. Eine solche Ausführungskondition (auch Trigger genannt) wäre beispielsweise `$input.Input1.value > $variable.threshold`. Insgesamt funktioniert AWS IoT Events also wie ein deterministischer Automat, da Zustandsübergänge genau definiert sind.

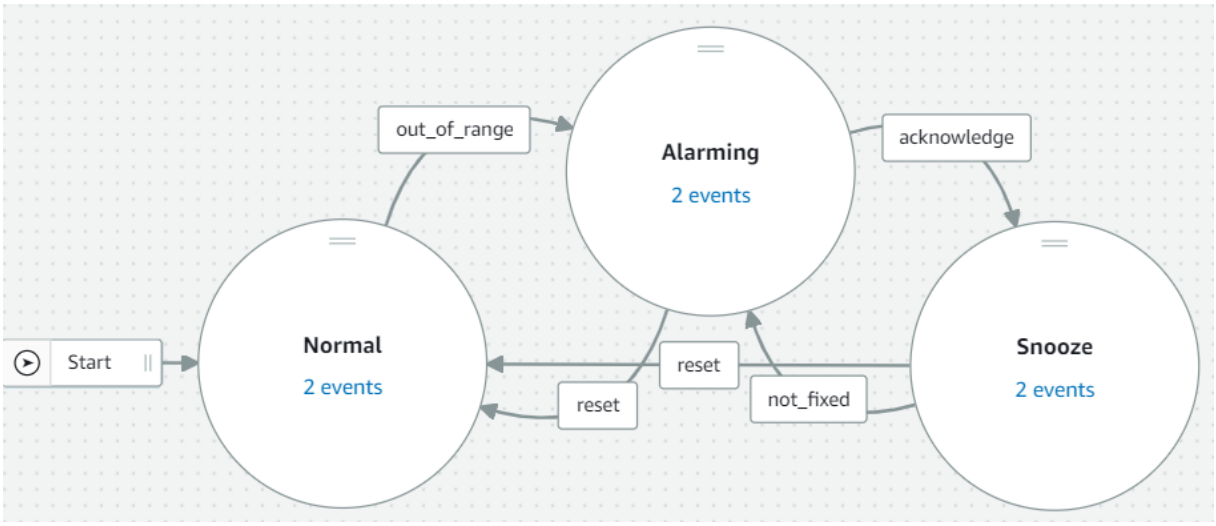


Abb. 20: Beispiel IoT Events

Sollte ein Alarmzustand erreicht werden, können andere Dienste wie Lambda oder SNS zur

[59] Vgl. Amazon Web Services, Inc. o.J.(bj)
[60] Vgl. Amazon Web Services, Inc. o.J.(ay)

Verarbeitung oder Benachrichtigung integriert werden.

### Features des Dienstes

AWS IoT Events unterstützt reine „if-then-else“ Überprüfungen. Trotzdem sind Variablen zur Evaluation selbstgeschriebener Logik verfügbar, womit sich zumindest ein Teil der gewünschten Features umsetzen lässt. Fakt ist dennoch, dass nur Analysen in einem endlichen zeitlichen Fenster durchführbar sind, welches durch Eingangsfrequenz und Anzahl der verwendeten Variablen beschränkt ist.[61] Eine Kalkulation eines Medians wäre (angenommen, dass die Werte sortiert gespeichert werden) wie folgt möglich, wenn 10 Variablen angenommen werden: `0.5*($variable.pastmeasure5 + $variable.pastmeasure6))` Abseits von Schwellwertüberprüfungen, welche vorher definiert wurden, ist AWS IoT Events nur mit großem Aufwand bei beschränkter Evaluationssprache zu weitergehenden Auswertungen fähig, welche immer von dem Zeitfenster, welches die Variablen abdecken abhängig ist. Ebenfalls sind keine selbstständigen Algorithmen zur Anomalieerkennung integriert.

### Performancegarantien

AWS gibt in der zu AWS IoT Events zugehörigen Service Level Agreement (SLA) keine Performancegarantien, sondern lediglich eine Verfügbarkeitsgarantie mit Penalen in Form von Rückzahlungen.[62] AWS IoT Events hat dazu noch Limitierungen, wie beispielsweise das unveränderliche Limit von 10 Nachrichten pro Sekunde, die an einen Detektor gesendet werden können (also bei denen eigene Logik ausgeführt werden kann) oder das anpassbare Limit von 1000 Nachrichten, die pro Sekunde insgesamt evaluiert werden können.[63]

### Gesamtkosten

Im Folgenden werden für den Beispielusecase entsprechend der offiziellen Preisaufstellung von AWS die monatlichen Nutzungskosten berechnet.[64]

[61] Vgl. Amazon Web Services, Inc. o.J.(ai)
[62] Vgl. Amazon Web Services, Inc. o.J.(aa)
[63] Vgl. Amazon Web Services, Inc. o.J.(z)
[64] Vgl. Amazon Web Services, Inc. o.J.(y)

| Dimension | Preis($)/Einheit | Summe ($) |
|---|---|---|
| evaluierte Regeln | 0,000018/Regel<br>(teuerster Preis - ohne Volumenrabatt) | 155,52 |
| Alarme | 0,1/Alarm<br>(pro Sensor) | 20 |
| SNS (Push) | 0,00002/Nachricht<br>(angenommen 5 Alarme/Gerät/Monat) | 0,02 |
| Summe | | 175,6445 |

Tab. 5: Kostenvergleich AWS IoT Events

### 4.2.2 Amazon Kinesis

Wenn über Kinesis gesprochen wird, ist zwischen mehreren Diensten zu differenzieren. Für diese Arbeit relevant sind Kinesis Data Streams, Kinesis Data Analytics und am Rande Kinesis Data Firehose, es gibt beispielsweise aber auch Kinesis Video Streams.

Mit Kinesis Data Streams ist der Dienst gemeint, der die Schnitstellen und Logik für das Streamen von Daten bereitstellt und Konsumenten wie Kinesis Data Analytics, EC2 oder Lambda unterstützt.

Kinesis Data Analytics ist dafür ausgelegt, die Daten aus z.B. Kinesis Data Analytics nahe Echtzeit mittels SQL Abfragen zu analysieren und z.B. Alarme auszulösen.

Kinesis Data Firehose dient dem Zweck, Daten aus z.B. Kinesis Data Analytics in Speichermedien/Datenbanken wie S3, Redshift oder Elasticsearch Service zu übertragen.

Amazon Kinesis Data Analytics ist im Gegensatz zu AWS IoT Analytics nicht allein auf die Analyse von IoT Daten spezialisiert. Kinesis eignet sich vielmehr für generelle Analysen von allerlei Streamingdaten. Zusätzlich ist Amazon Kinesis (Data Streams) älter als AWS IoT Analytics und bildet die technische Grundlage für die Verarbeitung AWS IoT Events.[65] Kinesis Data Streams wirbt damit, dass Daten in 70 Milisekunden nach Eingang zum Konsum verfügbar sind.[66]

Alternativ zur Kinesis Familie gibt es auch Open-Source Stream Analytics Dienste, wie von Singh/Hoque/Tarkoma dargestellt.[67] Diese kommen aufgrund der fehlenden Integration mit AWS und des erhöhten Aufwands durch Eigenbetrieb nicht in Frage.

[65] Vgl. Pogosova 2020
[66] Vgl. Amazon Web Services, Inc. o.J.(i)
[67] Vgl. Singh/Hoque/Tarkoma 2016

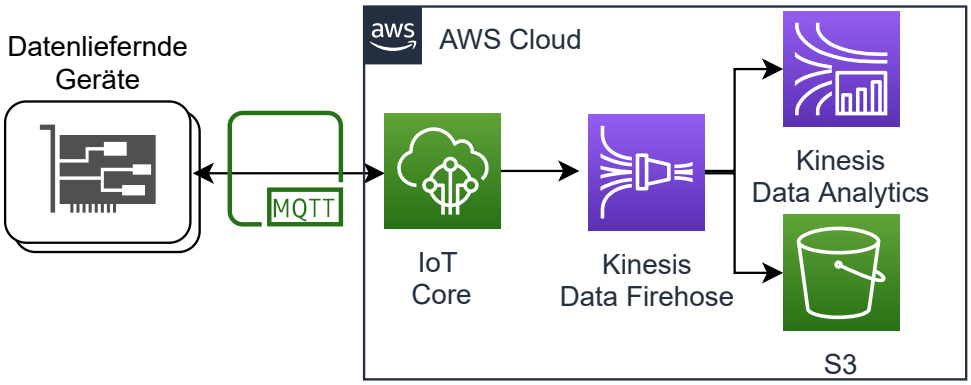


Abb. 21: Grobarchitektur des Ablaufes für Kinesis Analytics

In Abbildung 21 ist das Zusammenspiel der Dienste aus der Kinesis Familie mit anderen Diensten dargestellt. Angenommen werden dabei die in Abschnitt 3.1 erläuterten Rahmenbedingungen, weshalb IoT Core als Message Broker eingesetzt ist. Wie in der in Unterabschnitt 4.4.1 beschriebenen Architektur, muss auch hier für die Datenverarbeitung eine Regel im IoT Core Broker angelegt werden, um relevante Nachrichten an Kinesis Data Firehose weiterzuleiten.[68] Die tatsächliche Analyse übernimmt der komplementäre Dienst Kinesis Data Analytics. In diesem Fall wurde angenommen, dass Kinesis Data Firehose die Datenübertragung vornimmt, da bei Kinesis Data Streams einzelne Shards, wie in Abbildung 22 gezeigt, zu verwalten sind. Die Wahl von Kinesis Data Firehose bringt den NAchteil mit sich, dass die Daten nicht in Kinesis aufbewahrt werden können, was bei Data Streams möglich ist.[69] Da die Daten aber in S3 von Kinesis Data Firehose persistiert werden und einzig das Volumen der durchgeleiteten Daten abgerechnet wird, ist die Verwendung von Firehose statt Streams unkomplizierter. Trotzdem kann, wenn benötigt Firehose mit Data Streams ersetzt werden, da beide mit Kinesis Data Analytics kompatibel sind.

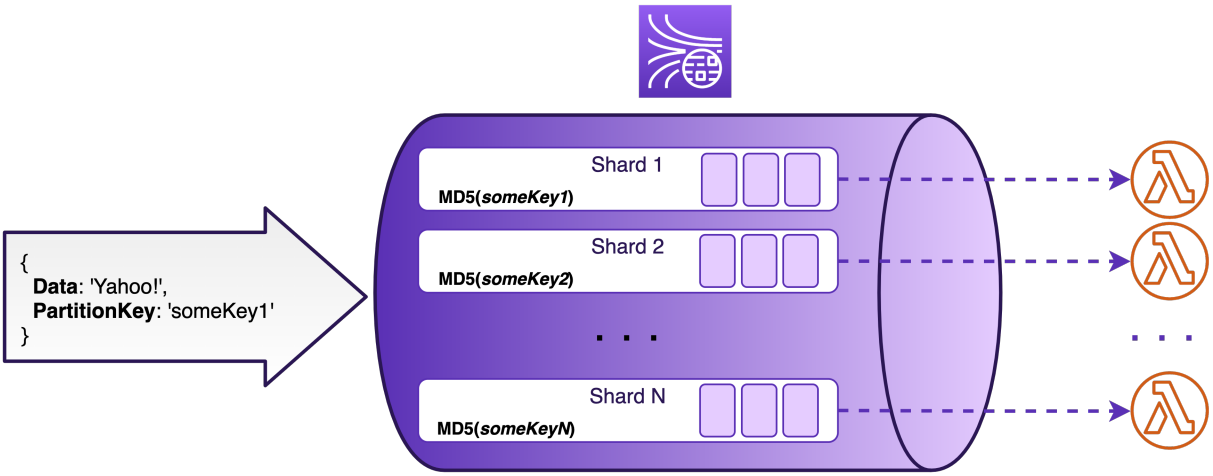


Abb. 22: Funktionsweise von Kinesis.[70]

[68] Vgl. Amazon Web Services, Inc. o.J.(an)
[69] Vgl. Amazon Web Services, Inc. o.J.(bf)

Kinesis Data Streams unterstützt, im Gegensatz zu Data Firehose, eine erweiterte Aufbewahrung [*Data Retention*], bei welcher Daten bis maximal 365 Tage nach initialem Einspielen erneut an Konsumenten wie Kinesis Data Analytics zur Verarbeitung gesendet werden können. Dies zieht Zusatzkosten nach sich. Treten jedoch Fehler in einer Analyse auf, kann diese für den gewählten Aufbewahrungszeitraum wiederholt werden.

Kinesis Data Analytics unterstützt sowohl die Datenanalyse mittels SQL als auch mittels der programmatischen Schnittstellen, die das Open Source Projekt Apache Flink anbietet.[71] Der Analysecode, der die Schnittstellen von Flink verwendet, kann in Java, Scala oder Python geschrieben sein.

### Features des Dienstes

Folgend werden die Funktionen der SQL-Analyse von Kinesis Data Analytics dargestellt, da die Funktionalitäten der Flink-Schnittstelle abhängig sind von Programmiersprache und verwendeter Bibliotheken. Eine direkte Funktion um den Median oder Quantile zu berechnen, ist nicht vorhanden.[72] Statdessen ist aber eine Lösung mittels der `group_rank` Funktion möglich, wie von RyanN im AWS Forum gezeigt.[73,74] Mittels dem von Guha u. a. gezeigten Random Cut Forest Algorithmus können Anomalien erkannt werden.[75] Dieser Algorithmus ist in Kinesis Data Analytics in Form der
`RANDOM_CUT_FOREST_WITH_EXPLANATION` Funktion integriert.[76] Die Schwellwertüberschreitungserkennung ist mittels einer SQL `WHERE` Bedingung in der Abfrage machbar. Der gleitende Durchschnitt lässt sich für Intervalle mittels der `EXP_AVG` Funktion berechnen.[77] Herman kritisiert bei Kinesis Data Analytics den angepassten SQL Dialekt, welcher keine Interoperabilität zulässt und die fehlende Testbarkeit außerhalb von AWS Werkzeugen.[78]

### Performancegarantien

Kinesis Data Streams bietet einen MB Durchsatz pro Sekunde und provisioniertem „Shard“. Da Kinesis Data Firehose auf Kinesis Data Streams aufbaut, ist ähnliches für Data Firehose anzunehmen[79] AWS bietet laut eigener Aussage mittels einer spezieller HTTP/2 Application Programming Interface (API) auch eine Leseverzögerung von 70 Milisekunden, oder weniger.[80]

[70] Entnommen aus: Pogosova 2020
[71] Vgl. Amazon Web Services, Inc. 2020b
[72] Vgl. RyanN 2018
[73] Vgl. Amazon Web Services, Inc. o.J.(ak)
[74] Vgl. RyanN 2018
[75] Vgl. Guha u. a. 2016, S. 1
[76] Vgl. Amazon Web Services, Inc. o.J.(at)
[77] Vgl. Amazon Web Services, Inc. o.J.(ah)
[78] Vgl. Herman 2020
[79] Vgl. Pogosova 2020
[80] Vgl. Amazon Web Services, Inc. o.J.(i)

Laut der AWS-eigenen Dokumentation liegt die Ende zu Ende Verzögerung von Dateneinspeisung bis Konsumption typischerweise unter einer Sekunde. Dies weicht das eigentliche Marketingversprechen von 70 Milisekunden schon auf.[81] Die Kinesis SLA enthält auch keine Klausel zur eigentlichen Performance, nur eine Garantie zur Verfügbarkeit von 99,99% pro Monat.[82]

Es gab bis jetzt ein schweres Ereignis, bei dem Kinesis in einer Region für den Zeitraum von 12 Stunden beeinträchtigt war und andere, von Kinesis abhängige Dienste beeinträchtigte.[83] Dieses Ereignis war jedoch auf die „us-east-1“ Region begrenzt und die Auswirkungen konnten durch Migration auf andere Regionen, wie beispielsweise „eu-central-1“ für Nutzende mitigiert werden.

#### Gesamtkosten

Für die Gesamtschätzung wurde Kinesis Data Firehose mit Streaming zu S3 und Kinesis Data Analytics angesetzt.[84] Zusätzlich wurde angenommen, dass Analysen mittels SQL erfolgen, da wie gezeigt genügend der erwünschten Features vorhanden sind. Kinesis Data Firehose rundet Datensätze auf die nächsten 5KB auf.

| Dimension | Preis($)/Einheit | Summe ($) |
|---|---|---|
| Kinesis Data Firehose | 0,033/GB | 1,38 |
| S3-Speicher | 0,0245/GB Speicher | 0,64 |
| Kinesis Data Analytics Processing Unit | 0,132/h | 32,12 |
| SNS (Push) | 0,00002/Nachricht (angenommen 5 Alarme/Gerät/Monat) | <0,02 |
| Lambda Ausführungen | 0,0000002/Ausführung 0,0000166667/Sekunde (angenommen: 5) 0,0000000167/GB RAM/ms | 0,08 |
| Summe | | 34,24 |

Tab. 6: Kostenvergleich Amazon Kinesis

Zusätzlich wird angenommen, das SQL-Statements ausgeführt werden, welche als „Processing units“ abgerechnet werden.[85] Da Kinesis Data Analytics nicht selbstständig mit SNS kommunizieren kann, wird Lambda als Weiterleitungsplattform für Alarme mit angesetzt.[86]

Preise für Kinesis Data Streams wären die Folgenden, unter der Annahme, dass S3 wegfällt, da Kinesis Data Streams die Daten selber zwischenspeichern kann:[87]

[81] Vgl. Amazon Web Services, Inc. o.J.(bg)
[82] Vgl. Amazon Web Services, Inc. o.J.(k)
[83] Vgl. auch im Folgenden Amazon Web Services, Inc. 2020h
[84] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(g)
[85] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(f)
[86] Vgl. Amazon Web Services, Inc. 2017
[87] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(j)

| Dimension | Preis($)/Einheit | Summe ($) |
|---|---|---|
| Kinesis Data Streams Shard-Stunde | 0,018/h | 13,14 |
| Kinesis Data Streams PUT-Operationen | 0,0175/Million | 0,15 |
| Kinesis Data Analytics Processing Unit | 0,132/h | 32,12 |
| SNS (Push) | 0,00002/Nachricht (angenommen 5 Alarme/Gerät/Monat) | <0,02 |
| Lambda Ausführungen | 0,0000002/Ausführung 0,0000166667/Sekunde (angenommen: 5) 0,0000000167/GB RAM/ms | 0,08 |
| Summe | | 45,51 |

Tab. 7: Kostenvergleich Amazon Kinesis Data Streams

### 4.2.3 AWS Lambda

Bei AWS Lambda handelt es sich um die Amazon Implementierung eines FaaS Dienstes. Innerhalb dieser Arbeit wird Lambda als einzige generelle Computing Plattform betrachtet, da Alternativen wie EC2, welches virtuelle Maschinen anbietet oder Elastic Container Service (ECS), welches Container anbietet einen von einzelnen Events unabhängigen Lebenszyklus haben. So laufen Container auf ECS oder virtuelle Maschinen auf EC2, wenn nicht anders konfiguriert kontinuierlich und holen/ „fetchen“ Daten. In Zeiträumen, in denen keine Daten bereitstehen, sind die entsprechenden Container und virtuellen Maschinen im Leerlauf, was unnötige Kosten erzeugt. Lambda hingegen wird von unterstützenden Diensten zur Verarbeitung von Events aufgerufen. Dabei ist je nach Dienst einstellbar, ob ein Aufruf pro Event stattfinden soll, oder Events zu einer konfigurierbaren Anzahl gruppiert werden und dann an Lambda übergeben werden. Lambda eignet sich besonders für analytische Workloads, seit der kürzlichen Addition von Intels Advanced Vector Extensions 2 (AVX2).[88] AVX2 ist ein spezieller CPU-Instruktionssatz, der die Verarbeitung von Vektorinstruktionen beschleunigt. Diese kommen besonders häufig in der Statistik oder bei dem Machine Learning vor. Aufgrund der zentralen Rolle im Bereich Compute bei AWS, können viele Dienste Events an Lambda senden. In Abbildung 23 sind als Integrationsbeispiele die Message oriented Middlewares (MoMs) IoT Core, MQ und Kinesis Data Streams als Eventlieferanten gezeigt.

[88] Vgl. auch im Folgenden Beswick 2020a

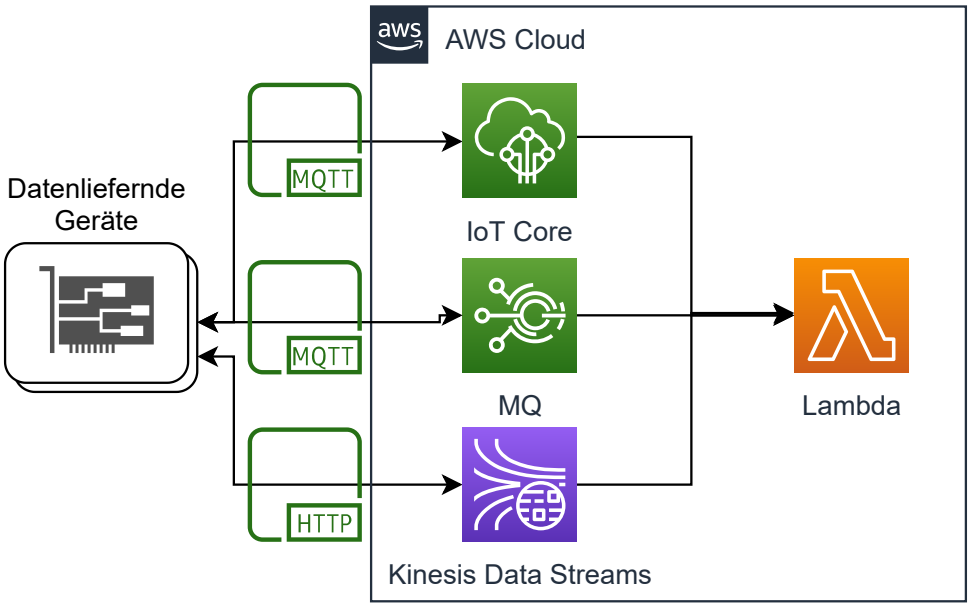


Abb. 23: Grobarchitektur des Ablaufes für Lambda

### Features des Dienstes

Da Lambda eine programmierbare Plattform ist, welche mehrere Sprachen unterstützt, muss im Folgenden eine Programmiersprache angenommen werden. Python hat in einer Umfrage von Kaggle unter Datenwissenschaftlern die höchste Popularität, gefolgt von SQL ausgemacht. Aus diesem Grund wird im Folgenden von der Verwendung der Programmiersprache Python ausgegangen (speziell auch, da sich eine Umsetzung mit SQL in Lambda schwierig gestalten würde).[89] Gemäß einer Analyse des Entwicklerportals Stack Overflow ist das Paket „pandas“ dabei die populärste Programmbibliothek für Datenwissenschaft.[90] Aus diesem Grund wird im Folgenden die Verwendung von Python mit pandas in Lambda angenommen. Angenommen wird, dass das an Lambda übermittelte Bearbeitungsfenster groß genug war, um Analysen zuzulassen. In pandas können Quantile mittels der `pandas.DataFrame.quantile()` Methode berechnet werden.[91] Für den Median ist eine eigene Methode verfügbar, `pandas.DataFrame.median()`.[92] Bartos/Mullapudi/Troutman haben den Random Cut Forest Algorithmus von Guha u. a. in Python zur Verwendung mit pandas als seperate Bibliothek implementiert und OpenSource bereitgestellt.[93,94] Es gibt, wie bei den anderen Diensten gezeigt, viele weitere Methoden zur Anomalieerekennung, welche programmatisch implementiert werden könnten. Schwellwertüberschreitungen können mittels `pandas.DataFrame.gt()` überprüft werden, wobei „gt“ für „greater-than“ steht.[95] Ein

[89] Vgl. Hayes 2020
[90] Vgl. Robinson 2017
[91] Vgl. o.V. o.J.(c)
[92] Vgl. o.V. o.J.(b)
[93] Siehe: `https://github.com/kLabUM/rrcf`
[94] Vgl. Bartos/Mullapudi/Troutman 2019
[95] Vgl. o.V. o.J.(a)

exponentieller gleitender Durchschnitt lässt sich, wie von Sharma gezeigt, mittels der folgenden, verketteten, Methoden berechnen: `pandas.DataFrame.ewm().mean()`.[96]

### Performancegarantien

AWS bietet für Lambda in dem SLA eine 99,**95** % garantierte Verfügbarkeit an.[97]

Die Zuweisung von Random-Access Memory (RAM) und daran gekoppelt vCPUs erfolgt bei Lambda dynamisch und ist vom Nutzer einzustellen. Um dies für Nutzende einfacher zu machen, gibt es das Projekt Lambda Power Tuning, welches die optimale RAM-/Leistungskonfiguration für Funktionen ermittelt durch mehrere Tests.[98] Da höhere RAM Einstellungen mehr kosten, kann für optimale Leistung oder für das Preis/Leistungsoptimum optimiert werden.

Wie von Madden/Bawcom gezeigt, eignet sich Lambda für die Verarbeitung großer Datensätze. Dies wurde mit der Verarbeitung von 259 TB Daten in knapp 20 Minuten demonstriert.[99] Speziell heben Madden/Bawcom auch hervor, dass Lambda von keiner Last zu knapp zwei Millionen Einträgen pro Sekunde und zurück skaliert hat.

### Gesamtkosten

Tabelle 8 zeigt die möglichen Kosten, wenn für jede eingehende Nachricht eine Lambdafunktion aufgerufen und ausgeführt wird (unabhängig davon, dass das mit den Standardeinstellungen für Parallelität nicht realisierbar wäre).

| Dimension | Preis($)/Einheit | Summe ($) |
|---|---|---|
| Lambda Ausführungen | 0,0000002/Ausführung | 1,728 |
| Lambda RAM | 0,0000000167/GB-Sekunde | 720 |
| S3-Speicher | 0,0245/GB Speicher | 0,0006 |
| SNS (Push) | 0,00002/Nachricht (angenommen 5 Alarme/Gerät/Monat) | 0,02 |
| Summe | | 721,75 |

Tab. 8: Kostenvergleich AWS Lambda Maximal

Um diese hohen Kosten zu mitigieren, gibt es mehrere Varianten. Folgend wird auf die Zwischenschaltung einer MoM, nämlich AWS Simple Queue Service (SQS) und auf die Vorfilterung der Nachrichten durch AWS IoT Core Rules näher eingegangen.

Abbildung 24 zeigt die Zwischenschaltung von SQS als Puffer, von dem asynchron gelesen werden kann.

[96] Vgl. Sharma 2019
[97] Vgl. Amazon Web Services, Inc. 2019f
[98] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(ar)
[99] Vgl. auch im Folgenden Madden/Bawcom 2019

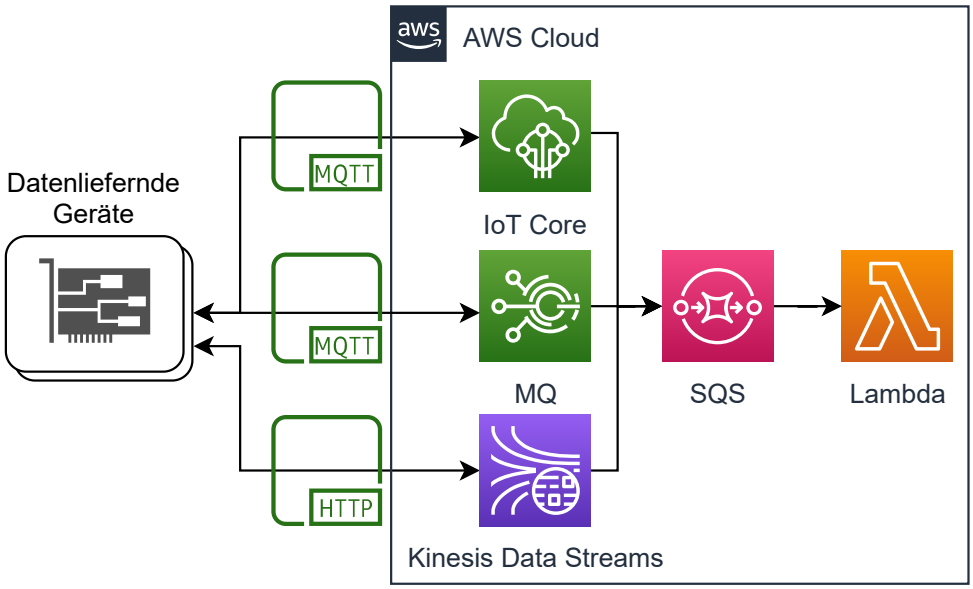


Abb. 24: Lambda Sammelverarbeitung via SQS

Seit November 2020 ist es in Lambda möglich, mittels der Einstellung „MaximumBatchingWindowInSeconds“ das Übertragungsfenster frei zu definieren, in welchem auf Nachrichten von SQS gewartet wird.[100] Dies erlaubt innerhalb eines maximalen Fensters von fünf Minuten Nachrichten zu sammeln und an Lambda zu übermitteln. Dies reduziert entsprechend die Ausführungen auf 20 pro Stunde und ist, wie in Tabelle 9 zu sehen, wesentlich günstiger.

| Dimension | Preis($)/Einheit | Summe ($) |
|---|---|---|
| Lambda Ausführungen | 0,0000002/Ausführung | 0,0029 |
| Lambda RAM | 0,0000000167/GB-Sekunde | 1,2 |
| S3-Speicher | 0,0245/GB Speicher | 0,0006 |
| SQS-Durchsatz | 0,40/Million Anfragen | 3,456 |
| SNS (Push) | 0,00002/Nachricht (angenommen 5 Alarme/Gerät/Monat) | 0,02 |
| Summe | | 4,6795 |

Tab. 9: Kostenvergleich AWS Lambda Sammelverarbeitung

Alternativ ist mittels einer angepassten Regel in AWS IoT Rules, die in Codeausschnitt 2 gezeigt ist, eine Ausführung nur bei Überschreitung des Schwellwerts möglich.

```
SELECT *, value
FROM 'iot-topic'
WHERE value > 60
```

Codeausschnitt 2: IoT Rule Schwellwertregel

[100] Vgl. auch im Folgenden Amazon Web Services, Inc. 2020g

Diese Regel reduziert die rechnerischen Ausführungen auf 1000 pro Monat. Wie in Tabelle 10 gezeigt, fallen die Kosten bei 1000 Ausführung noch geringer als bei der gepufferten Variante mit SQS aus. Es ist abhängig, ob eine Schwellwertüberprüfung in AWS IoT Rules, speziell unter dem Paradigma der „Immutable Infrastructure“, welche unveränderliche, versionsverwaltete Infrastruktur fordert, als akzeptabel angesehen wird.[101] Für Änderungen an Schwellwerten müsste nämlich entsprechend die AWS IoT Rule angepasst werden, was im SQS durch eine einfach einzuspielende Codeänderung machbar wäre.

| Dimension | Preis($)/Einheit | Summe ($) |
|---|---|---|
| Lambda Ausführungen | 0,0000002/Ausführung | 0,0002 |
| Lambda RAM | 0,0000000167/GB-Sekunde | 0,0833 |
| S3-Speicher | 0,0245/GB Speicher | 0,0006 |
| SNS (Push) | 0,00002/Nachricht<br>(angenommen 5 Alarme/Gerät/Monat) | 0,02 |
| Summe | | 0,1041 |

Tab. 10: Kostenvergleich AWS Lambda IOT Sammelverarbeitung

### 4.2.4 Amazon MSK / ksqlDB

Bei Managed Streaming for Apache Kafka (MSK) handelt es sich um einen Managed Service für die Open Source Lösung Apache Kafka. Im Gegensatz zu anderen, hier im Kapitel aufgeführten Lösungen wie IoT Analytics, hat Amazon MSK nicht von Grund auf selber entwickelt, sondern einen großen Teil des Codes von Apache Kafka übernommen. Dies erklärt auch, warum die Anbindung an andere Dienste von Amazon bedeutend schwieriger ist, als beispielsweise an IoT Core. In der in Abbildung 25 abgebildeten Grobarchitektur müsste zur Anbindung von Apache Kafka an zuliefernde Geräte ein Intermediär wie IoT Core verrwendet werden, da Apache Kafka ein eigenes, binäres Protokoll hat, welches sonst in die Geräte implementiert werden müsste. **To do** letzten Halbsatz belegen (11) Fraglich ist, ob sich eine Implementierung des Kafka Protokolls auf allen Endgeräten lohnt, speziell im Licht besser unterstützter Standards wie MQTT. Umgangen werden kann die Implementierung des Kafka Protokolls auf zuliefernden Geräten auf dreierlei Arten: Zum einen lassen sich mittels Kafka Connect for MQTT MQTT Broker als Eventquellen anbinden.[102] Alternativ kann Kafka auch als MQTT Proxy dienen, was bedeutet, dass Kafka als eigenständiger MQTT Broker agiert, wobei zu beachten ist, dass Kafka weit nicht alle MQTT Standardelemente implementiert und eine paralelle Weiterverarbeitung in anderen Amazon Diensten nicht möglich ist.[103] Zuletzt gibt es noch die Möglichkeiten, die Nachrichten via IoT Core innerhalb von AWS an MSK weiterzuleiten, was auch in Abbildung 25 dargestellt ist.

[101] Vgl. Amazon Web Services, Inc. o.J.(bd)
[102] Vgl. Erber 2021
[103] Vgl. Erber 2021

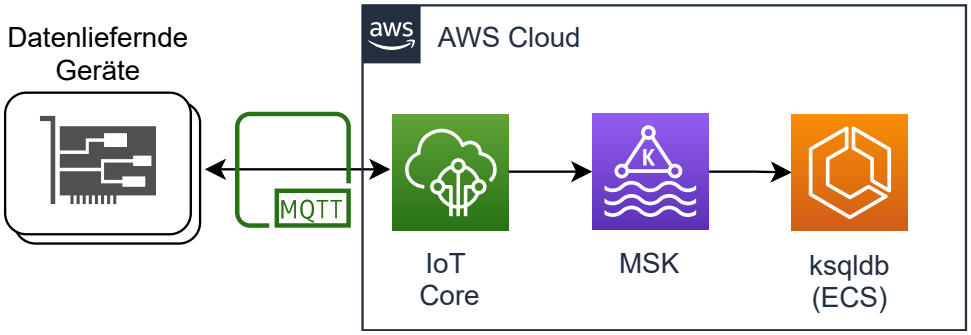


Abb. 25: Grobarchitektur des Ablaufes für Managed Streaming for Apache Kafka

Kafka selber fungiert nur als Broker für Daten. Da sich aber im Laufe der Zeit im Kafka Ökosystem Angebote wie ksqlDB entwickelt haben, welche nur mit Kafka funktionieren werden diese betrachtet. Möglich, aber kostenintensiv (durch Weiterleitung IoT Core → MSK → Lambda) wäre die Weiterleitung auch an Lambda. Stattdessen soll aber ksqlDB die Verarbeitung übernehmen. Dieses wird wie sein Vorgänger KSQL hautpsächlich von der Firma Confluent, Inc. als OpenSource entwickelt und dient der Verarbeitung von Kafka Daten mittels SQL in Form einer Streamverarbeitung.[104,105] KSQL/ksqlDB kann wie von Penz gezeigt, in AWS in einem Container in ECS oder in einer virtuellen Maschine, in EC2 betrieben werden.

### Features des Dienstes

Im Folgenden wird der Featureumfang von ksqlDB näher beleuchtet. Die Berechnung eines Medians/von Perzentilen scheint in ksqlDB nicht trivial machbar zu sein. Wie von Waehner gezeigt, ist es möglich in den ksql eigenen User Defined Functions (UDFs), in welchen eigener Code ausgeführt wird, Anomalieerkennung basierend auf Machine Learning durchzuführen.[106] Eine Schwellwertüberschreitung kann mittels einer `WHERE` Bedingung festgestellt werden. Ein gleitender Durchschnitt kann ebenfalls mittels einer UDF realisiert werden.[107] Diese berechnet den gleitenden Durchschnitt mit eigenem Code.

### Performancegarantien

Da die Ressourcen für ksqlDB selbst eingestellt werden können, ist eine absolute Performanceaussage nicht möglich. Zusätzlich müssen zugehörige Ressourcen entweder selbst verwaltet werden oder als managed service von Confluent bezogen werden. Die SLA von MSK definiert als Verfügbarkeitsziel 99,9%, jedoch keine weiteren Performanceziele.[108] Die Performance von MSK ist aufgrund des unterliegenden Instanzmodells wesentlich durch die Nutzenden beeinflussbar und

[104] Vgl. Kreps 2019
[105] Vgl. Narkhede 2017
[106] Vgl. Waehner 2018
[107] Vgl. Confluent, Inc. o.J.
[108] Vgl. Amazon Web Services, Inc. 2019c

ggf. durch vertikale Skalierung verbesserbar. In einem Performancetest, den AWS für Statz durchgeführt hat, wurde als machbare Eingangsrate 310MB/Sekunde bei 15 provisionierten Brokern für möglich erklärt.[109]

### Gesamtkosten

Wie von Beswick dargestellt und in Abbildung 26 gezeigt, muss MSK in mindestens zwei Availability Zones gestartet werden.[110] Dabei ist es für weniger kritische Systeme, wie hier gezeigt möglich, nur ein Network Address Translation (NAT)-Gateway zu verwenden, welches Konnektivität zum Internet ermöglicht (ohne eingehenden Traffic zu erlauben). Für kritische Lasten sollte ksqldb und das NAT-Gateway in das zweite Subnetz repliziert werden.

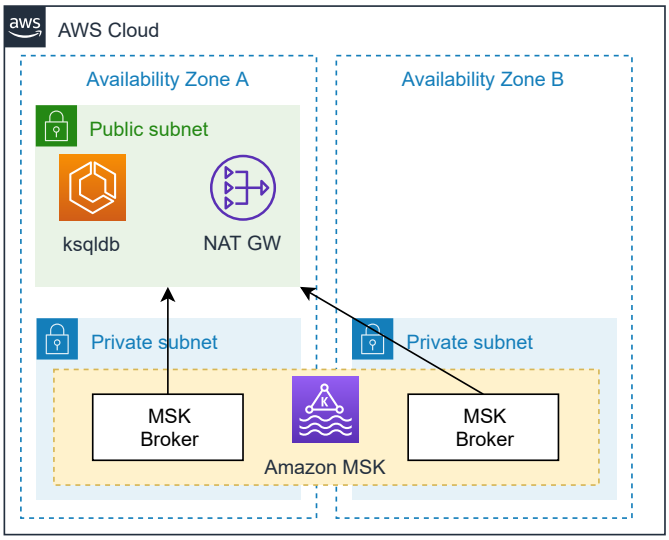


Abb. 26: Networking MSK

Um ksqlDB zu betreiben, wird im folgenden angenommen, dass ein Elastic Container Service (ECS) Container mit 1vCPU, 4GB RAM und 10GB Elastic Filesystem (EFS) Speicherplatz provisioniert werden muss. Zusätzlich werden zwei MSK-Brokerinstanzen, welche auf unterliegenden EC2-Servern basieren, provisioniert.[111] Zusätzlich wurde ein NAT-Gateway (für Updates und ausgehenden Traffic) mit 2GB Datendurchsatz angesetzt. Der Kostenvergleich ist in Tabelle 11 gezeigt.

[109] Vgl. Statz 2019
[110] Vgl. auch im Folgenden Beswick 2020b
[111] Vgl. Amazon Web Services, Inc. o.J.(m)

| Dimension | Preis($)/Einheit | Summe ($) |
|---|---|---|
| Broker Instanz (t3.small - 2 mal) | 0,0526/h | 76,7960 |
| Broker Storage | 0,119/h | 0,714 |
| NAT Gateway Zeit | 0,052/h | 37,96 |
| NAT Gateway Durchsatz | 0,052/GB | 0,104 |
| ECS Fargate vCPU (ksqlDB - 1 vCPU) | 0,04656/vCPU/h | 33,989 |
| ECS Fargate RAM (ksqlDB - 4GB) | 0,00511/GB/h | 14,921 |
| SNS (Push) | 0,00002/Nachricht (angenommen 5 Alarme/Gerät/Monat) | 0,02 |
| EFS Storage | 0,36/h | 3,6 |
| Summe | | 168,104 |

Tab. 11: Kostenvergleich Amazon MSK

In einem Vergleich zwischen AWS und dem Mitbewerber Confluent, stellte Statz in einer Modellrechnung fest, dass AWS preisgünstiger war.[112]

### 4.2.5 Auswahl

Da Kafka übertragbar ist, genauso wie der Programmcode von Lambda, wurden bei beiden Diensten kein Punktabzug für die Übertragbarkeit vorgenommen. Alle Dienste mit Ausnahme von der Kombination aus MSK und ksqlDB integrieren sich gut mit dem Dienstleistungsumfang von AWS. Speziell ist dabei Lambda zu erwähnen, welches als generalistische Plattform mit vielen Diensten interagieren kann. Da IoT Events speziell auf die Erkennung von Ereignissen anhand von „if-then-else“ Bedingungen entwickelt wurde, sind kaum andere, generalisiertere Anwendungsfälle realisierbar. Ebenfalls ist eine Erweiterbarkeit mit AWS IoT Events nicht gegeben und auch eine Erfüllung der vorgegebenen Auswertungen ist nur mit höchster Implementierungskreativität in Teilen machbar. Abzüge zur Fehlertransparenz gab es bei allen Dienstleistungen, speziell auch in Lambda, da Fehlertriage nur durch Replikation der Inputs und Debugging des Codes machbar ist. Da statt Kinesis Data Streams Kinesis Data Firehose für den Vergleich angenommen wurde, welches nicht manuell skaliert werden muss, wurden keine Abzüge beim Wartungsaufwand und der Skalierbarkeit gemacht. Da der Wartungsaufwand bei Lambda stark von der Stabilität des laufenden Programmcodes abhängig ist und genau überprüft werden muss, ob Fehler bei der Verarbeitung auftreten, wurden Abzüge gemacht. MSK mit ksqlDB zu skalieren ist eine aufwändige Aufgabe, welche häufiges nachjustieren und viel Zeit erfordert. Im Gegensatz dazu ist AWS IoT Events nur zu warten, wenn das Nachrichtenformat vom initial konfigurierten abweicht. Da Lambda und AWS IoT Events serverless bzw. im Fall von AWS IoT Events auch vollständig

[112] Vgl. Statz 2019

verwaltet sind, ist die Skalierbarkeit und „serverlessness“ gegeben. Kinesis hat im Normalbetrieb die geringsten monatlichen Kosten, während es bei Lambda von der konkreten Konfiguration und der Anzahl an übermittelten Nachrichten abhängig ist, wie hoch die Kosten ausfallen. AWS IoT Events ist im Vergleich zu den anderen Diensten Preis/Leistungs-technisch stark unterlegen. Bei MSK wurden Abzüge gemacht, da viele einzeln zu skalierende Teile, die einzeln teurer werden könnten abgerechnet werden und mindestens zwei Broker betrieben werden müssen. Kinesis ist ein System, was überwiegend stabil läuft und auch vor Nutzerverursachten Fehlern gefeit ist, was einer der Gründe sein dürfte, weshalb andere AWS Dienste auf Kinesis aufbauen. Trotzdem wurde ein Punkt beim Wartungsaufwand abgezogen, da im Falle eines Ausfalles von Kinesis schnell agiert werden muss, um neu eintreffende Daten nicht zu verlieren. Lambda hingegen ist je nach Ausgestaltung des Programmcodes nicht besonders fehlertolerant. Da MSK aus Redundanzgründen in mindestens zwei Availability Zones gestartet werden muss, ist es standardmäßig robust, jedoch kommt durch ksqlDB ein weiterer „Point of failure“ hinzu, welcher mit größerem Aufwand robust und fehlertolerant gemacht werden muss. AWS IoT Events ist sehr robust und fehlertolerant, da es komplett verwaltet wird und die einzigen Fehler durch Nutzende eingeführt werden können.

| Kriterium | max. Punkte | Amazon Kinesis | AWS Lambda | Amazon MSK& ksqlDB | AWS IoT Events |
|---|---|---|---|---|---|
| Übertragbarkeit zwischen Clouds (ISO 9126) | 1 | 0 | 1 | 1 | 0 |
| Integration mit anderen AWS Diensleistungen | 3 | 3 | 3 | 1 | 3 |
| Generalisierung | 4 | 4 | 4 | 4 | 1 |
| Erweiterbarkeit | 4 | 4 | 4 | 4 | 1 |
| Fehlertransparenz/ Debugability | 5 | 4 | 3 | 4 | 1 |
| geringer Wartungsaufwand | 7 | 6 | 5 | 3 | 7 |
| Skalierbarkeit & „serverlessness“ | 7 | 7 | 7 | 4 | 7 |
| Kosten | 7 | 7 | 6 | 5 | 1 |
| Performancegarantien | 8 | 8 | 8 | 6 | 4 |
| Robustheit & Fehlertoleranz | 9 | 8 | 5 | 7 | 9 |
| Auswertungen (Unterabschnitt 2.1.2) | 11 | 11 | 11 | 11 | 4 |
| Summe | 66 | **62** | **57** | **50** | **38** |

Tab. 12: Bewertungsmatrix Echtzeit

Insgesamt ist der präferierte Dienst aus der Echtzeitkategorie, wie in Tabelle 12 zu sehen, Kinesis.

## 4.3 Dienste für Batchverarbeitung

In Abbildung 27 werden verwendbare Dienste für die Batch („OLAP“) Verarbeitung von AWS, gemeinsam mit ihren jeweiligen Einsatzgebieten gezeigt. In diesem Abschnitt soll besonders auf die Dienste zur Verarbeitung nach dem Laden der Daten in eine der gezeigten Datenquellen eingegangen werden. Gezeigt werden jedoch auch Dienste für Datenvisualisierung und Machine Learning, da diese komplementär oder mit den prozessierten Daten verwendet werden können. Da die einzelnen Datenquellen jeweils verschiedene Sprachen, bzw. Dialekte von SQL verwenden, sind die spezifischen Abfragesprachen mit einem generischen Eintrag in der Grafik abgebildet. Dabei ist es bei AWS durchaus üblich, Interoperabilität zu anderen AWS Diensten, wie beispielsweise Lambda, in den jeweiligen SQL Dialekt einzubauen.

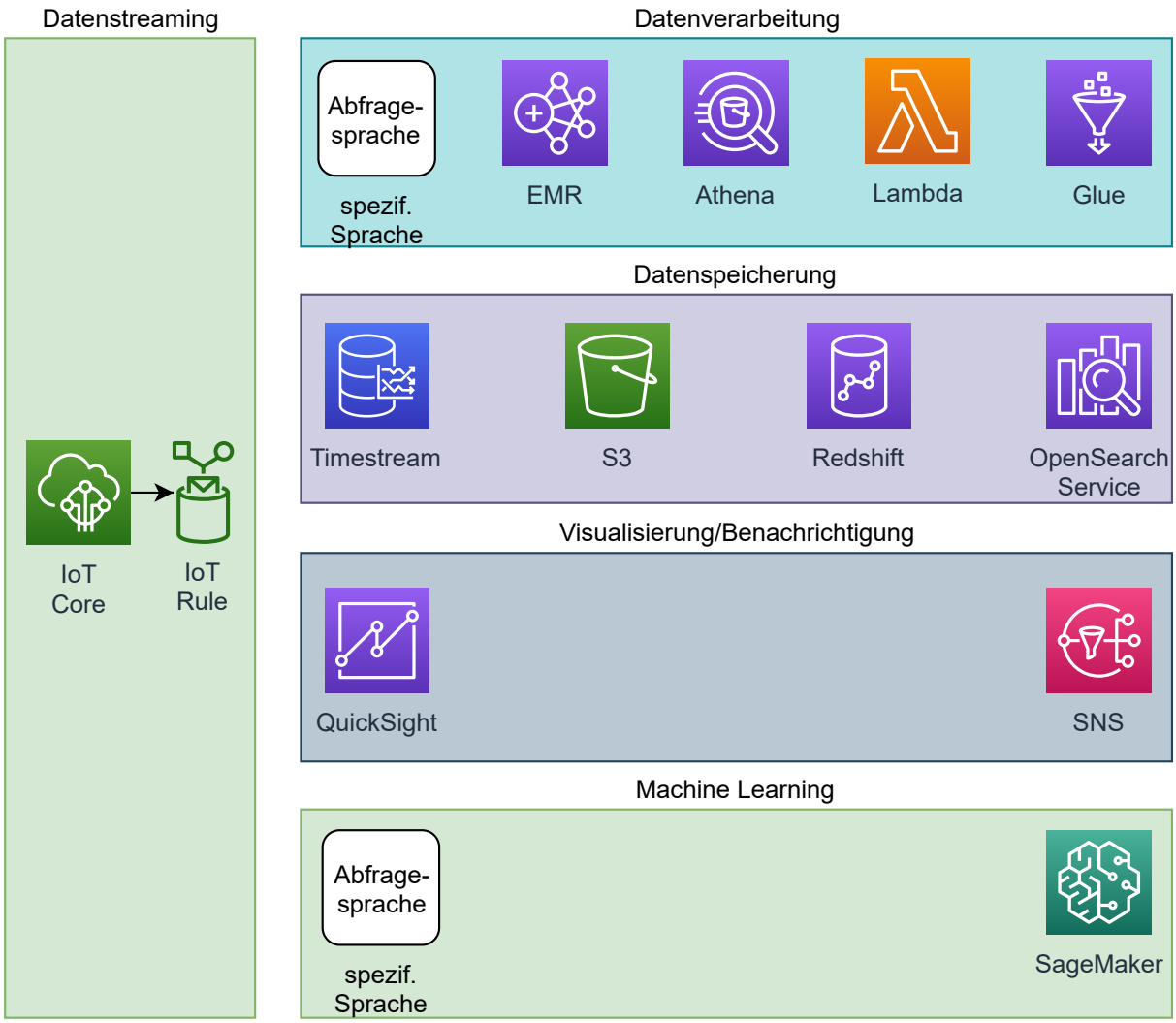


Abb. 27: Einsetzbare Dienste im Bereich Datenbankverarbeitung

### 4.3.1 Amazon Timestream

Timestream ist nach Aussage des Herstellers eine schneller, skalierbare und speziell für Zeitreihendaten entwickelte Datenbank, welche über AWS als Dienstleistung bezogen werden kann.[113] Timestream integriert zwei verschiedene Speichertypen, nämlich RAM-basierten Speicher, in dem Daten, auf welche schnell zugegriffen werden sollen, gespeichert werden können und Festplatten-basierten Speicher, welcher für historische Daten dienen soll. Timestream besitzt zusätzlich einen eigenen SQL-Dialekt, welcher um Funktionen zur Analyse von Zeitreihendaten erweitert wurde.

Timestream ist nicht selbst in der Lage, Abfragen zu planen. AWS gibt in der Dokumentation an, dass Lambda verwendet werden kann, um periodisch eine Abfrage in Timestream auszuführen und basierend auf dieser Anfrage Alarme via SNS zu versenden.[114] Abgebildet ist diese Interaktion in Abbildung 28.

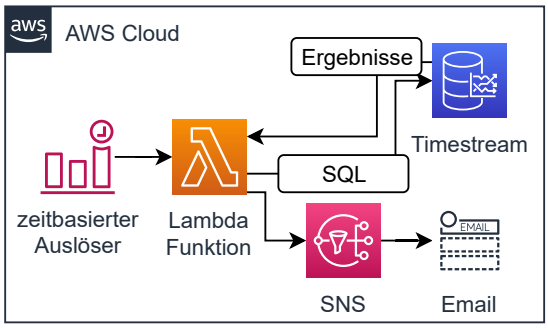


Abb. 28: Orchestrierung von zeitbasierten Timestream Abfragen

#### Features des Dienstes

Perzentile lassen sich mit der eingebauten Funktion `approx_percentile(x, percentage)` berechnen, der Mittelwert via `avg(x)` oder `geometric_mean(x)` Eine native Anomalieerkennung bietet Timestream nicht, jedoch kann die Machine Learning Dienstleistung Amazon SageMaker Daten von Timestream analysieren und Anomalieerkennung ausführen.[115] In SageMaker kann auch der nativ in Kinesis Data Analytics verbaute Random Cut Forest Algorithmus verwendet werden, um Anomalien zu erkennen. Andernfalls kann, wie von Salgado beschrieben, eine einfache Anomalieerkennung durch Überprüfung des Wertes auf Lage zwischen dem 25. und 75. Quantil erfolgen.[116] Die Schwellwertüberschreitungserkennung ist mittels einer einfachen `WHERE` Bedingung machbar. Ein gleitender Durchschnitt ist, wie von Ross gezeigt, in SQL mittels der Bearbeitungsfenster, die Timestream, wie viele andere SQL Implementierungen anbietet, möglich.[117]

[113] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(bh)
[114] Vgl. Amazon Web Services, Inc. o.J.(ad)
[115] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(q)
[116] Vgl. Salgado 2019
[117] Vgl. Ross 2020

Gleichzeitig unterstützt Amazon Timestream die Verwendung von Ableitungen als Werkzeug zur Trenderkennung.[118]

### Performancegarantien

Die SLA von Timestream bietet allein 99,99% Verfügbarkeit pro Verrechnungsmonat.[119] AWS verspricht aber eine bis zu tausendfache Geschwindigkeitsverbesserung gegenüber relationalen Datenbanken.[120]

Mitbewerber von AWS im Bereich der Zeitseriendatenbanken haben in ihren Tests festgestellt, dass Timestream langsamer als ihre Konkurrenzdienste/Konkurrenzprodukte waren.[121,122] Dabei sind die Testskripte von Booz Open Source und damit die Messungen theoretisch reproduzierbar. Gleichzeitig gibt AWS aber in einem Blogeintrag aus dem November an, Datensätze im Bereich von einem bis 21,7 TB analysieren zu können, ohne 100 Sekunden Ausführungszeit zu überschreiten.[123] Auch für diese Tests ist der Quellcode open source verfügbar.

Es kann keine finale Aussage über die genauen Performancegarantien getroffen werden, da entweder die Daten der Mitbewerber von AWS als gültig anzunehmen sind, oder die Daten von AWS selbst. Da es im kommerziellen Interesse aller Parteien liegt Timestream entweder auf- oder abzuwerten ist eine stichhaltige Aussage nicht möglich.

### Gesamtkosten

Timestream ist momentan noch nicht in Frankfurt verfügbar, weshalb die Kosten in der einzigen europäischen Zone mit verfügbarem Timestream, Irland, als Maßstab verwendet werden. Angenommen wird, dass die Daten eine Stunde im RAM vorgehalten werden und danach in die Festplattenspeicherung überführt werden.

Ausgehend von dem beschriebenen Szenario lässt sich nicht genau errechnen, welche Datenmenge von den Anfragen genau erfasst wird. Dies ist dem Fakt geschuldet, dass Timestream die Daten optimiert abspeichert und nur den tatsächlichen Messwert speichert. Numerische Daten können als 32-bit int, als 64bit BigInt oder als 64bit double gespeichert werden.[124,125] Wenn dazu ein String für die Sensorid im Stile „Sensor-123“ angenommen wird, der 19 bytes zur Darstellung benötigt und der Zeitstempel addiert wird, der 8 bytes benötigt, ergibt sich eine Speicherbelegung von 91 Bytes. Speicherkosten werden aber bei Werten unter einem Kilobyte auf einen Kilobyte

[118] Vgl. Amazon Web Services, Inc. o.J.(ag)
[119] Vgl. Amazon Web Services, Inc. 2020e
[120] Vgl. Amazon Web Services, Inc. o.J.(r)
[121] Vgl. Booz 2020
[122] Vgl. Crate.io, Inc. 2020
[123] Vgl. Das/Rath 2020
[124] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(ba)
[125] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(s)

hochgerundet, weshalb dies in der Speicherrechnung keine Beachtung findet. Wenn man weitere Optimierungen, die die Abfrageengine macht außer Acht lässt, muss mit 0,78624GB pro abgefragtem Monat gerechnet werden.

| Dimension | Preis($)/Einheit | Summe ($) |
|---|---|---|
| Schreibzugriffe | 0,5654/Mio./KB | 4,8851 |
| RAM Speicherung | 0,0407/GB/h | 0,3516 |
| Festplatten Speicherung | 0,0339/GB/Monat | 0,8787 |
| Anfragen | 0,011308/GB abgefragt | 25,6055 |
| Step Functions | 0,025/1000 Zustandsübergänge | 1,08 |
| Lambda Ausführungen | 0,0000002/Ausführung | 0,0002 |
| Lambda RAM | 0,0000000167/GB-Sekunde | 0,08 |
| SNS (Push) | 0,00002/Nachricht (angenommen 5 Alarme/Gerät/Monat) | 0,02 |
| Summe | | 32,9011 |

Tab. 13: Kostenvergleich AWS Timestream

### 4.3.2 Amazon Athena/Amazon S3

Amazon Athena ist nach Aussage des Herstellers ein voll verwalteter Query Dienst, welcher das Durchsuchen von großen Datenmengen im S3 Speicherdienst möglich macht.[126] Athena basiert dabei auf dem ursprünglich von Facebook entwickelten Presto, welches mittlerweile Open Source ist. Innerhalb von Athena können Daten verschiedener Formate (Comma seperated values (CSV), Apache Parquet, Apache ORC, JSON, ...) mittels SQL verarbeitet werden. Zusätzlich ist seit November 2020 mittels der sogenannten „Federated Queries“ auch die Abfrage von anderen Datenquellen wie Apache HBase, Amazon Document DB oder von Datenquellen, die durch eigenentwickelte Verbindungselemente verknüpft werden.[127] Technisch werden diese Federated Queries via Lambda abgewickelt.

[126] Vgl. Barr 2016
[127] Vgl. Amazon Web Services, Inc. o.J.(b)

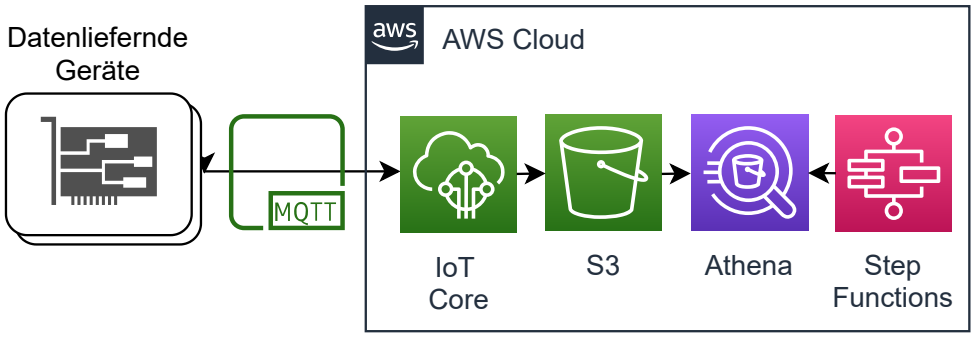


Abb. 29: Grobarchitektur des Ablaufes für Athena

In Abbildung 29 wird die Grobarchitektur unter Einsatz von AWS IoT Core, S3, Athena und StepFunctions gezeigt. StepFunctions dient als Orchestrierungsdienst, für die wiederholte Ausführung der Abfragen und entsprechende Weiterleitung an SNS. Bei verändernden Datenschemata wird empfohlen, den Dienst AWS Glue Crawler komplementär einzusetzen, welcher die Schemata aus S3 automatisch extrahiert und in Athena hinterlegt. Andernfalls können die Schemata auch manuell hinterlegt werden.

### Features des Dienstes

Die technische Grundlage für Athena, ist das OpenSource Projekt Presto. Dieses bietet mit `approx_percentile(x, percentage)` eine Funktion zur Kalkulation von Perzentilen an.[128] Eine Möglichkeit, um Anomalien in Athena zu erkennen, ist es alle Werte, die ausserhalb der Spanne zwischen dem 25. und 75. Quantil liegen als Anomalie zu klassifizieren.[129] Dies wäre mit der `approx_percentile(x, percentage)` Funktion von Athena machbar. Andernfalls könnte wie von Megler vorgeschlagen, der Amazon Elastic Map Reduce (EMR) Dienst benutzt werden, um Anomalien mittels der Bildung von Clustern und der Kalkulation der Distanz von Werten zu diesen Clustern zu detektieren.[130] Schwellwertüberschreitungen können via einer `WHERE` Bedingung, wie bei Timestream auch, erkannt werden. Wie bei Timestream auch kann nach der Methode von Ross ein gleitender Durchschnitt mittels der Verarbeitungsfenster kalkuliert werden.[131]

### Performancegarantien

Hartland/Frost/Love beschreiben einen Anwendungsfall, in dem sie Athena zur Analyse von Logdaten innerhalb der Datenanalyseinfrastruktur des ATLAS Experiments am Large Hadron Collider im CERN Forschungszentrum bei Genf verwenden.[132]

[128] Vgl. The Presto Foundation o.J.
[129] Vgl. Salgado 2019
[130] Vgl. Megler 2016
[131] Vgl. Ross 2020
[132] Vgl. Hartland/Frost/Love 2018

Gleichzeitig stellen Hartland/Frost/Love auch fest, dass es Teil des Entwicklungsprozesses ist, die Anfragen zu optimieren, um Kosten zu sparen.[133] Dies entstammt der Natur von SQL basierten Abfragemechanismen, da verschiedene Abfragestile und Operationen auf verschieden viele Speicherpartitionen zugreifen müssen. Da Athena nach Volumen der Speicherzugriffe abrechnet, kann es also sinnvoll sein, den Speicher nach Abfragearten zu partitionieren oder die Abfragen zu optimieren. Dazu können der im April 2021 vorgestellten `EXPLAIN` SQL-Befehl und die damit zusammenhängenden query execution plans verwendet werden.[134] Diese zeigen nach Aussage von AWS auf, wie eine Abfrage ausgeführt werden würde und wie Laufzeiten optimiert werden können.

Die reale Performance der bearbeiteten Anfragen garantiert AWS nicht. Da Athena ein serverless Dienst ist, dessen unterliegende Kapazität vollständig von AWS verwaltet wird, können Nutzende keinen Einfluss auf die provisionierten Ressourcen nehmen. Zusätzlich hängt die Performance stark von der Partitionierung der Daten und der Kompression ab.[135] Zusätzlich zu beachten ist, dass Athena eine Limitierung von 20 (25 in der North Virginia Region) parallel laufenden/wartenden Anfragen hat und Anfragen nach 30 Minuten Laufzeit abgebrochen werden.[136] Diese Limitierungen können jedoch auf Anfrage erhöht werden.

Amazon garantiert vertraglich keine Performance, sondern nur die 99,99% Verfügbarkeit des Dienstes in dem SLA.[137] In Vergleichen von Levy und Khadtare mit dem Mitbewerber Google BigQuery zeigte Athena schlechtere Performance, gemessen an den Antwortzeiten der Abfragen, war aber günstiger.[138,139] Da AWS Ende 2020 mit Athena engine version 2 wesentliche Performanceverbesserungen angekündigt hat, ist nicht bekannt, ob die Daten noch aktuell sind.[140]

#### Gesamtkosten

Für die folgende Kostenbewertung wird angenommen, dass Athena die Daten aus S3 indiziert. Dort liegen die Daten im JSON Format gespeichert vor. Zu beachten ist, dass effizientere Formate wie CSV, oder sogar Apache Parquet und ORC verfügbar sind. Diese können komprimiert Daten speichern. Eine Verwendung von Parquet oder ORC würde laut AWS 30-90% Kosten sparen.[141] Gleichzeitig kann aber eine Speicherung im JSON Format via AWS IoT Rule erfolgen, ohne dass weitere Verarbeitung notwendig ist.

In Tabelle 14 wird davon ausgegangen, dass 2MB an Daten im Zeitraum von 10 Minuten neu hinzukommen. Da aber eine Historie gebildet werden soll, müssen sowieso die gesamten dreimonatigen historischen Daten abgefragt werden. Ausgehend von 200 kB pro Minute von allen

[133] Vgl. Hartland/Frost/Love 2018, S. 5
[134] Vgl. auch im Folgenden Amazon Web Services, Inc. 2021a
[135] Vgl. Levy 2021
[136] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(av)
[137] Vgl. Amazon Web Services, Inc. 2019a
[138] Vgl. Levy 2019
[139] Vgl. Khadtare 2018
[140] Vgl. Amazon Web Services, Inc. 2020a
[141] Vgl. Amazon Web Services, Inc. o.J.(c)

Geräten ergeben sich 25,92 GB S3-Datenvolumen an historischen Daten, welche 960 mal im Monat abgefragt werden sollen. Athena rundet dabei auf das nächste MegaByte auf und hat ein Minimum von 10 MB erfasster Daten pro Anfrage.[142] Das Datenschema für Athena stellt Glue bereit, welches Kosten für Abfragen aus dem Datenkatalog und Speicherkosten für den Datenkatalog erhebt.[143] Zur Orchestrierung wird ein Ablauf als Zustandsmaschine in StepFunctions abgebildet, welches für jeden Zustandsübergang eine Gebühr erhebt.[144]

| Dimension | Preis($)/Einheit | Summe ($) |
|---|---|---|
| Athena Abfragen | 0,005/TB abgefragte Daten | 121,50 |
| S3-Speicher | 0,0245/GB Speicher | 0,64 |
| Glue Data Catalog Speicher | 1/100.000 Objekte | <0,0001 |
| Glue Data Catalog Abfragen | 1/Million Abfragen | <0,0001 |
| Step Functions | 0,025/1000 Zustandsübergänge | 1,08 |
| SNS (Push) | 0,00002/Nachricht (angenommen 5 Alarme/Gerät/Monat) | 0,02 |
| Summe | | 123,24 |

Tab. 14: Kostenvergleich Amazon Athena

### 4.3.3 Amazon Redshift

Amazon Redshift ist der Data Warehouse Dienst von AWS, welcher nach Aussage des Herstellers „enterprise-level“ ist und auf ein Datenvolumen von Petabytes skalieren kann.[145] Redshift ist dabei ein klassischer OLAP Dienst, welcher für eine Vielzahl verschiedener Daten effizient Auswertungen bereitstellen kann. Redshift basiert auf der bekannten Open Source Datenbank PostgreSQL, weicht jedoch in der Implementierung diverser Kommandos und Features ab, die Amazon als irrelevant für OLAP Anwendungen hält.[146,147] Kern von Redshift sind sogenannte Cluster, welche aus einer oder mehrerer Berechnungsknoten („compute nodes“) und Anführerknoten („leader nodes“) besteht. Applikationen interagieren allein mit den Anführerknoten, die existierenden Berechnungsknoten sind zwar transparent für die Anwendung, werden jedoch von den Anführerknoten mit Ausführungsplänen versorgt, die diese entwickelt um Anfragen effizient zu verarbeiten.[148]

Redshift bietet zusätzlich mit Redshift Spectrum einen Dienst an, welcher auf den ersten Blick dem in Unterabschnitt 4.3.2 vorgestellten Athena gleicht. Beide rufen via SQL Daten von S3

[142] Vgl. Amazon Web Services, Inc. o.J.(c)
[143] Vgl. Amazon Web Services, Inc. o.J.(u)
[144] Vgl. Amazon Web Services, Inc. o.J.(ae)
[145] Vgl. Amazon Web Services, Inc. 2020c, S. 1
[146] Vgl. Amazon Web Services, Inc. 2020c, S. 4
[147] Vgl. Amazon Web Services, Inc. 2020c, S. 428 ff.
[148] Vgl. Amazon Web Services, Inc. 2020c, S. 4

ab und beide kosten 5$/TB gescannte Daten.[149] Wichtige Unterschiede liegen dabei aber in der Art, wie Ressourcen verwaltet und genutzt werden können. Während Redshifft Spectrum nur in Kombination mit einem Redshift Cluster verwendet werden kann, funktioniert Athena ohne Kopplung an verwaltende Ressourcen. Gleichzeitig ist ein stärkerer Einfluss auf die Performance bei Redshift möglich, da man zusätzliche Clusterressourcen einfach provisionieren kann, während Athena vollständig von AWS verwaltet wird. Entsprechend erlaubt Redshift Spectrum zwar größeren Einfluss auf die Performance von Datenabfragen, dieser Einfluss muss jedoch in Form von zusätzlich abzurechnenden Clusterressourcen abgerechnet werden, was Redshift für Anwendungsfälle, die keine gesichert gleichbleibende Performance benötigen, unattraktiv macht.

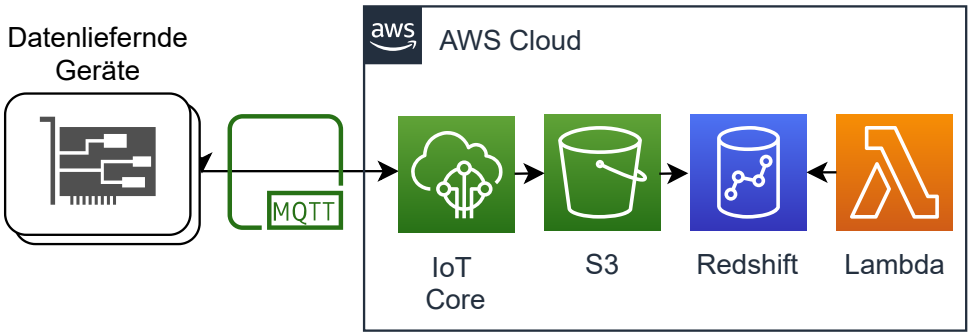


Abb. 30: Grobarchitektur des Ablaufes für Redshift

In Abbildung 30 ist beispielhaft die Verwendung von Redshift mit S3 (ohne Spectrum) gezeigt. Dabei lädt Redshift die Daten von S3 und löscht diese anschliessend. Anfragen werden via der Java Database Connectivity (JDBC)-Schnittstelle von Redshift mit Lambda getätigt und ausgewertet. Alternativ könnte Kinesis Data Firehose zum Laden der Daten in Redshift zum Einsatz kommen.

### Features des Dienstes

Redshift verfügt über die Funktionen `APPROXIMATE PERCENTILE_DISC(percentile)` und `PERCENTILE_CONT(percentile)`, welche Perzentile kalkulieren können durch Annahme einer diskreten oder kontinuierlichen Datenverteilung.

Wie bereits bei Timestream und Athena gezeigt, kann Salgados Vorschlag verwendet werden, um alle Werte, ausserhalb der Spanne zwischen dem 25. und 75. Quantil liegen als Anomalie zu klassifizieren.[150] Obenstehende Perzentilfunktionen könnten dafür genutzt werden. Wie bereits bei Timestream und Athena vorgeschlagen, könnten auch externe Tools verwendet werden, um Anomalien mittels Machine Learning oder statistischen Methoden zu entdecken. Eine weitere Möglichkeit wäre die mittlere absolute Abweichung vom Median in einem Verarbeitungsfenster

[149] Vgl. auch im Folgenden Smallcombe 2020
[150] Vgl. Salgado 2019

zu analysieren und größere Abweichungen als Anomalie anzuerkennen.[151] Schwellwertüberschreitungen können via einer `WHERE` Bedingung, wie bei Timestream und Athena auch, erkannt werden. Wie bei Timestream und Athena kann nach der Methode von Ross ein gleitender Durchschnitt mittels der Verarbeitungsfenster kalkuliert werden.[152,153] Diese Methode wird auch von Ubiq für Redshift vorgeschlagen.[154]

### Performancegarantien

AWS sichert vertraglich für Redshift ebenfalls keine feste Performance im Rahmen des SLAs zu. Es werden allein Abschläge auf den zu zahlenden Preis angeboten, wenn die Verfügbarkeit des Dienstes unter 99,99% des Monats lag.[155] Basierend auf den Leistungsdaten der Instanz, welche ausgewählt wurde, um Redshift zu betreiben und weiterer Faktoren, wie z.B. ob durch horizontale Skalierung mehrere Instanzen in einem Cluster zusammengefasst wurden, kann sich die Performance verändern. Zusätzlich sind wie bei vielen SQL-basierten Datenbanken Optimierungen der Leistung durch Optimierung der gestellten Abfragen möglich.[156] In der Erhebung von Tan, J. u. a. wurde Redshift (ohne Spectrum) ein Performancevorteil gegenüber Athena bescheinigt, während Spectrum schlechter abschnitt, als Redshift.[157]

### Gesamtkosten

Im Vergleich von Gupta u. a., bei dem alle Autoren AWS angehören, bescheinigen sie Redshift ein „disruptives“ Preismodell gegenüber anderer DataWarehouse Lösungen.[158] Ob für den Usecase dieser Arbeit Redshift ebenfalls ein „disruptives“ und ansprechendes Preismodell hat, soll im Folgenden erläutert werden.

Da Redshift Spectrum durch die zusätzlich zu provisionierenden Clusterresourcen einen Kostennachteil hat, wie auch von Tan, J. u. a. festgestellt, wird von einem Kostenvergleich für Spectrum abgesehen.[159]

Stattdessen werden die Kosten für ein „shared-nothing“[160] Redshift OLAP Cluster berrechnet. Dabei ist zu beachten, dass AWS IoT keine direkte Regel bietet, um Daten in Redshift abzulegen. Stattdessen müssen die Daten via Kinesis Data Firehose oder AWS Lambda eingefügt werden. Zum Zwecke der Datenübertragung wird in diesem Fall Kinesis Data Firehose kalkuliert. Wie bei Elasticsearch Service gibt es auch bei Redshift verschiedene unterliegende Instanzen zur

[151] Vgl. Peak 2017
[152] Vgl. Ross 2020
[153] Vgl. Ubiq o.J.
[154] Vgl. Ubiq o.J.
[155] Vgl. Amazon Web Services, Inc. 2019d
[156] Vgl. Amazon Web Services, Inc. o.J.(bc)
[157] Vgl. Tan, J. u. a. 2019, S. 2176
[158] Vgl. Gupta u. a. 2015
[159] Vgl. Tan, J. u. a. 2019, S. 2178
[160] Vgl. Tan, J. u. a. 2019, S. 2172

Auswahl.[161] Im Vergleich werden, den Empfehlungen von AWS folgend, eine Instanz der Dense Compute 2 (DC2) Klasse verwendet, welche sich für unkomprimierte Datenmengen kleiner ein TB eignen. Die kleinste verfügbare DC2 Instanz ist „dc2.large“ mit 2 vCPUs, 15GiB Hauptspeicher und 160 GB Festplattenspeicher. Redshift muss innerhalb eines Virtual Private Clouds (VPCs) gestartet werden, um Netzwerkisolation sicherzustellen. Aus diesem Grund ist ein Aufpreis auf Kinesis Data Firehose zu zahlen, welches Datenübertragung in ein VPC mit einem Aufschlag berechnet.[162] Kinesis Data Firehose rundet dazu noch Daten zu den nächsten 5 KB auf, was eine effektive Datenmenge von 41,77GB ergibt.

Zusätzlich müssen Lambda Ausführung dazu gerechnet werden, die die Ausführungen der hinterlegten SQL Abfragen zur Auswertung ausführen und danach SNS benachrichtigen.

| Dimension | Preis($)/Einheit | Summe ($) |
|---|---|---|
| dc2.large Instanz (OnDemand) | 0,324/h | 233,60 |
| Firehose Dateneingang | 0,033/GB | 1,38 |
| Firehose VPC | 0,01/GB<br>0,012/h | 8,84 |
| Lambda Ausführungen | 0,0000002/Ausführung | 0,000192 |
| Lambda RAM | 0,0000000167/GB-Sekunde | 0,08 |
| SNS (Push) | 0,00002/Nachricht<br>(angenommen 5 Alarme/Gerät/Monat) | 0,02 |
| Summe | | 243,920192 |

Tab. 15: Kostenvergleich Amazon Redshift

### 4.3.4 Amazon OpenSearch Service

Amazon OpenSearch Service ist die verwaltetete Variante des Elasticsearch Forks von Amazon, OpenSearch.[163] OpenSearch Service, dessen Namensänderung weg von Elasticsearch Service im April 2021 angekündigt wurde, bietet die Datenbank Elasticsearch als Service an, welche ursprünglich von Elastic entwickelt wurde. Elasticsearch ist dabei nach Aussage des Herstellers eine verteilte, freie und offene Analytics- und Suchplattform für viele verschiedenen Datenformen.[164] Elasticsearch basiert auf der Open Source Bibliothek Apache Lucene, welche besonders für Suchabfragen in großen Datenmengen optimiert ist. Elasticsearch ist neben Anwendungsfällen im Bereich Logverarbeitung und Monitoring auch für Industrial Internet of Things (IIoT) Anwendungsfälle bekannt.[165,166]

[161] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(o)
[162] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(h)
[163] Vgl. auch im Folgenden Meadows u. a. 2021
[164] Vgl. Elasticsearch, Inc. o.J.(d)
[165] Vgl. Mantfeld 2019
[166] Vgl. Bajer 2017

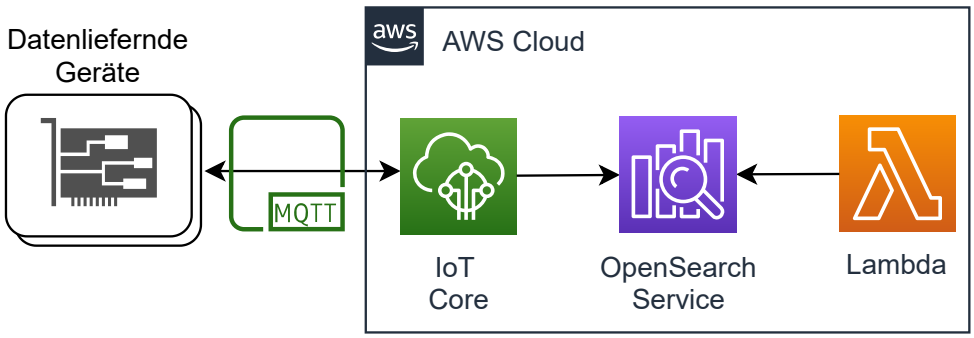


Abb. 31: Grobarchitektur des Ablaufes für OpenSearch Service

Wie in Abbildung 31 gezeigt, kann OpenSearch/Elasticsearch Service nativ via AWS IoT Core Daten empfangen. Lambda dient zur Orchestrierung der Abfragen und zur folgenden Auswertung.

### Features des Dienstes

Da der OpenSearch Fork von Amazon auf der Elasticsearch Basis basiert, und AWS explizit plant, vorest keine API Abweichungen zur bereits bekannten API einzubauen, werden im Folgenden die Elasticsearch Fähigkeiten dargestellt.[167]

Mithilfe der „Percentiles aggregation“ können beliebige Perzentile eines Datensatzes berechnet werden. Dies wird in Codeausschnitt 4 im Anhang 7 mit der Elasticsearch eigenen Abfragesprache gezeigt. In Elasticsearch/OpenSearch Service ist aufgrund von Amazon eigenen Anpassungen eine Anomalieerkennung basierend auf Random Cut Forest verfügbar.[168] Alternativ lassen sich basierend auf der mittleren absolute Abweichung vom Median mit Elasticsearch eigenen Mitteln ebenfalls Aussreisser/Anomalien erkennen.[169] In der Elasticsearch eigenen Abfrage gibt es mit dem `minimum_should_match` Parameter, eine Möglichlkeit Abfragen auf Schwellwertüberschreitungen zu stellen.[170] Gleichzeitig können in Kibana/Open Search Dashboards Schwellwerte mit Alarmen konfiguriert werden.[171] In Elasticsearch kann ein gleitender Durchschnitt mittels der eigenen Abfragesprache kalkuliert werden. Die Berechnung kann gewichtet oder ungewichtet erfolgen.

### Performancegarantien

AWS sichert vertraglich für OpenSearch Service ebenfalls keine feste Performance im Rahmen des SLAs zu. Auch hier werden nur Abschläge auf den zu zahlenden Preis angeboten, wenn

[167] Vgl. Meadows u. a. 2021
[168] Vgl. Amazon Web Services, Inc. o.J.(t)
[169] Vgl. Elasticsearch, Inc. o.J.(a)
[170] Vgl. Elasticsearch, Inc. o.J.(b)
[171] Vgl. Handler 2019

die Verfügbarkeit des Dienstes unter 99,99% des Monats lag.[172] Da OpenSearch Service ein Instanzbasiertes Modell verfolgt, sind eventuelle Performanceprobleme jedoch durch einen Wechsel auf eine Instanzklasse mit stärkerer Rechenleistung (vCPUs) oder Hauptspeicher (RAM) lösbar. Dabei zeigt der Anwendungsfall der Mayo Klinik, den Chen u. a. vorstellen, dass die unterliegende Software, Elasticsearch, auch mit Datensätzen von mehr als 25 Millionen JSON-Einträgen Anfragen mit einer Latenz von weniger als 0,2 Sekunden beantworten kann.[173]

#### Gesamtkosten

Der AWS OpenSearch Service wird auf unterliegenden EC2 Instanzen betrieben und wie diese in gewissen Klassen abgerechnet. Dabei stehen sowohl OnDemand Abrechnugsmodelle wie auch reservierte Kapazität zur Verfügung.[174] Zusätzlich steht mit „UltraWarm“ eine besondere Instanzklasse zur Verfügung, welche für das Vorhalten großer Datenmengen konzipiert ist. Zusätzlich zu den Instanzen wird noch der verbrauchte Elastic Block Storage (EBS)-Speicherplatz abgerechnet. Dieser Speicherplatz kann in der Standard Klasse oder speziell für hohen Datendurchsatz (provisionierte Input/Output Operations Per Second (IOPS)) gebucht werden. Der Speicherplatz für UltraWarm wird aber nicht via EBS abgerechnet, sondern separat. Ebenfalls steht wieder ein „Free Tier“ zur Verfügung, welches aus Vergleichsgründen nicht in die Berechnung einfliessen soll. Aus Vergleichsgründen kommen nur OnDemand abgerechnete Instanzen für den Vergleich in Frage. Für Vergleichszwecke soll eine t3.medium.elasticsearch Instanz geschätzt werden, welche mit 2 vCPUs und 4 GiB RAM ausgestattet ist. Der provisionierte EBS-Speicherplatz soll bei 10GB liegen. Benachrichtigungen werden über die native Integration von Kibana (Name nach Umbenennung: OpenSearch Dashboards) und SNS abgewickelt, wobei Kibana die Wertüberwachung übernimmt.[175]

| Dimension | Preis($)/Einheit | Summe ($) |
|---|---|---|
| t3.medium Instanz (OnDemand) | 0,084/h | 61,32 |
| EBS Speicher | 0,161/GB | 1,61 |
| SNS (Push) | 0,00002/Nachricht (angenommen 5 Alarme/Gerät/Monat) | 0,02 |
| Summe | | 62,95 |

Tab. 16: Kostenvergleich Amazon OpenSearch Service

### 4.3.5 Auswahl

Der von Tan, J. u. a. konstatierte Preisvorteil von Redshift gegenüber Athena hat sich in dem durchgeführten Vergleich nicht gezeigt.[176] Dies könnte womöglich dem Fakt geschuldet sein, dass

[172] Vgl. Amazon Web Services, Inc. 2019b
[173] Vgl. Chen u. a. 2017
[174] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(d)
[175] Vgl. Amazon Web Services, Inc. o.J.(a)
[176] Vgl. Tan, J. u. a. 2019, S. 2178 f.

die Datenmengen der Studie einen „Break-even“ Punkt überschritten haben, an dem das Athena Abrechnungsmodell im Vorteil gegenüber Redshift war. Die Performance der verglichenen Dienste variierte aufgrund verschiedener unterliegender Faktoren, wie beispielsweise provisionierter Leistung oder der Optimierung für spezifische Anfragen, was andere Anfragen verlangsamte. Letzten Endes ist es fraglich, ob OLAP Analysen, die zeitgesteuert erstellt werden eine garantierte Performance im Sub-Sekunden Bereich brauchen, oder ob es reicht, die Analyse in längerer Zeit zu erledigen, da die Daten sowieso zeitversetzt sind. Übertragbare Dienste/Produkte sind Amazon OpenSearch und Amazon Athena. OpenSearch, welches größtenteil Kompatibilität zu ElasticSearch hat (mit Ausnahme des Amazon eigenen Anomalieeerkennungsplugins) kann mit Anpassungen mit dem ElasticSearch Code in anderen Public Clouds verwendet werden. Da Athena auf Presto basiert, sind Abfragen kompatibel und portabel. Dies erfordert entsprechende Unterstützung für Presto in der Zielcloud oder eine eigene Installation. Bei den verglichenen Diensten handelt es sich mit Ausnahme von OpenSearch um von Amazon teilweise oder komplett entwickelte properitäre Dienste, weshalb die Integration mit anderen AWS Dienstleistungen gegeben ist. Auch OpenSearch wird mittlerweile, als Fork von Elasticsearch, hauptsächlich von AWS betreut und ist bereits gut mit anderen Dienstleistungen integriert. Auch wenn die verglichenen Datenbankdienste für jeweils individuelle Zwecke entwickelt wurden, sind sie doch generalisiert für mehrere Usecases verwendbar und erweiterbar. Im Bereich Fehlertransparenz gab es Abzug für Athena, da fehlschlagende Abfragen nur mit generischen Fehlermeldungen beantwortet werden, was die Fehlersuche erschwert. Punktabzüge im Wartungsaufwand und der Skalierbarkeit gab es für die auf Instanzen basierenden Dienste OpenSearch und Redshift, welche durch die von Nutzenden vorzunehmende vertikale Skalierung einen höheren Wartungs- und Überprüfungsaufwand fordern, als Timestream und Athena, die verwaltet sind. Die Gesamtkosten scheinen bei Redshift zumindest im vorliegenden Fall, ohne konkrete Optimierungen besonders hoch zu sein, genauso wie bei Athena. Kostenführer ist Timestream, gefolgt von OpenSearch Service. Alle verglichenen Dienste können die Auswertungen in angemessener Zeit anfertigen. Athena hat Punktabzüge im Bereich Robustheit & Fehlertoleranz, da ein sich veränderndes Datenschema zu schwerwiegenderen Problemen führen kann. Alle Dienste bieten die gewünschten Auswertungen an. Technisch sind aufgrund der unterschiedliche Implementierungen trotzdem Unterschiede vorhanden.

| Kriterium | max. Punkte | Amazon Timestream | Amazon OpenSearch | Amazon Athena | Amazon Redshift |
|---|---|---|---|---|---|
| Übertragbarkeit zwischen Clouds (ISO 9126) | 1 | 0 | 1 | 1 | 0 |
| Integration mit anderen AWS Diensleistungen | 3 | 3 | 3 | 3 | 3 |
| Generalisierung | 4 | 4 | 4 | 4 | 4 |
| Erweiterbarkeit | 4 | 4 | 4 | 4 | 4 |
| Fehlertransparenz/ Debugability | 5 | 5 | 5 | 4 | 5 |
| geringer Wartungsaufwand | 7 | 7 | 5 | 7 | 5 |
| Skalierbarkeit & „serverlessness“ | 7 | 7 | 5 | 7 | 5 |
| Kosten | 7 | 7 | 6 | 5 | 4 |
| Performancegarantien | 8 | 8 | 8 | 8 | 8 |
| Robustheit & Fehlertoleranz | 9 | 9 | 9 | 6 | 9 |
| Auswertungen (Unterabschnitt 2.1.2) | 11 | 11 | 11 | 11 | 11 |
| Summe | 66 | **65** | **61** | **60** | **58** |

Tab. 17: Bewertungsmatrix Batch

## 4.4 Dienste, die mehrere Modi unterstützen

Aufgrund der hohen Popularität der $\lambda$-Architektur, gibt es Dienste, die sowohl Echtzeitverarbeitung, als auch Batch/OLAP-Verarbeitung unterstützen.

### 4.4.1 AWS IoT Analytics

AWS IoT Analytics ist ein Dienst der AWS IoT Familie, der nach Aussage des Herstellers weitreichende Analysen von IoT Daten, die beispielsweise via AWS IoT Core geladen werden können, zulässt.[177] Dabei bietet der Dienst Lösungen für Sammlung, Verarbeitung, Speicherung, Analyse und Benachrichtigung/Visualisierung von Daten an, bzw. bietet Schnittstellen, um diese Aufgaben zu erledigen. Im speziellen deckt AWS IoT Analytics die Aufgaben einer $\lambda$-Architektur ab. So besitzt AWS IoT Analytics beispielsweise eine integrierte Zeitreihendatenbank, welche ergänzend zu den Echtzeitdaten von AWS IoT Core für Analysen genutzt werden kann.

[177] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(bi)

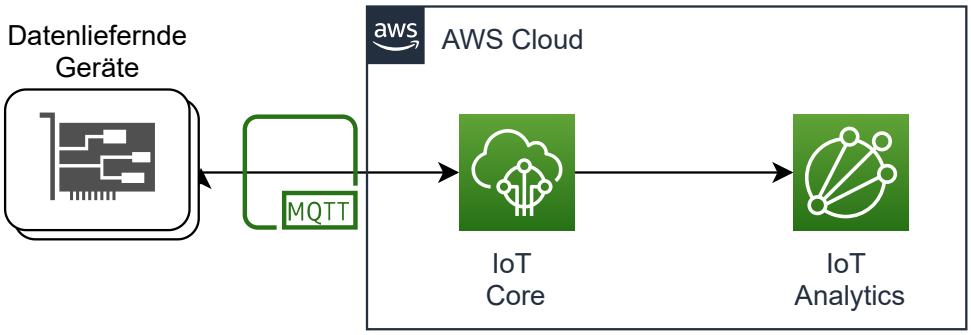


Abb. 32: Grobarchitektur des Ablaufes für IoT Analytics

In Abbildung 32 ist die Grobarchitektur und Verknüpfung mit anderen Diensten unter Annahme der Vorraussetzungen aus Abschnitt 3.1 gezeigt. Datenliefernde Geräte, wie beispielsweise Sensoren, liefern Zeitreihen-Messwerte via dem MQTT Protokoll an. Die Weiterleitung zu IoT Analytics erfolgt mittels einer eingerichteten Regel im IoT Core Messagebroker, welche mittels eines Dialekts der SQL Sprache gewisse Topics vorselektiert oder alle Topics zulässt.

Für erweiterte Analysen stellt AWS IoT Analytics sogenannte Notebooks zur Verfügung, die auf den „Jupyter Notebooks“ basieren. Diese Notebooks werden verwendet, um Python Programmabläufe zu visualieieren und zu modularisieren. Da in den Jupyter Notebooks der volle Paketumfang von Python verfügbar ist, inklusive z.B. Machine Learning Bibliotheken wie Tensorflow, sind wesentlich erweiterte Analysen möglich. Die Ressourcen für Notebooks sind seperat zu provisionieren und werden in Analytics Compute Units abgerechnet, wobei eine Compute Unit 4vCPU-Kerne und 16 GB RAM hat.[178] Diese Compute Units werden sekundengenau abgerechnet und kosten nur für die Laufzeit Geld.

#### Features des Dienstes

Der SQL-Analyseteil von IoT Analytics basiert wie Athena auf Presto und unterstützt die selben Funktionen und Operatoren.[179] Entsprechend sind die Anforderungen im selben Grad erfüllt wie bei Athena. Da AWS IoT Analytics aber über die selbst programmierbaren Notebooks verfügt, lassen sich Verbesserungen an den Ansätzen mittels einer eigenen, angepassten Implementierung einzelner Analysen machen. Da die Notebooks unter Python laufen können und ebenfalls pandas einbinden können, sind die selben Features wie bei Unterabschnitt 4.2.3 gegeben.

#### Performancegarantien

Die SLA von AWS IoT Analytics garantiert einzig die 99,9% Verfügbarkeit im Monat.[180] Es gibt auch *soft-limits*, welche auf Anfrage hochgesetzt werden können, welche einen Durchsatz von

[178] Vgl. Amazon Web Services, Inc. o.J.(w)
[179] Vgl. Amazon Web Services, Inc. o.J.(aw)
[180] Vgl. Amazon Web Services, Inc. 2019e

maximal 100.000 Nachrichten pro Sekunde vorsehen.[181] Einzig die Performance der Analytics Compute Units ist durch vertikale Skalierung, also eine größere Buchung von Compute Units durch Nutzende beeinflussbar.

Bei AWS IoT Analytics ist laut AWS mit einer Latenz von Minuten oder Sekunden zu rechnen, was hoch ist im Vergleich zu Kinesis, wo mit Sekunden oder Milisekunden Latenz zu rechnen ist.[182]

#### Gesamtkosten

In Tabelle 18 sind die kalkulierten Preise nach gängiger Preismatrix dargestellt.[183] Die Nutzung von externen Ausführungsdiensten (z.B. StepFunctions/Lambda) für SQL-Abfragen ist nicht notwendig, da die Abfragen im Dienst selbst terminiert werden können.

| Dimension | Preis($)/Einheit | Summe ($) |
|---|---|---|
| Datenspeicherung (roh+verarbeitet) | 0,03/GB (verarbeitet)<br>0,023 (roh) | 7,77 |
| Anfragenbearbeitung | 0,00634/GB | 121,50 |
| Eigene Analyselogik<br>960 Abfragen * 5 Sekunden | 0,36/h | 0,48 |
| Summe | | 129,75 |

Tab. 18: Kostenvergleich AWS IoT Analytics

Das bestehende „Free Tier“, welches AWS für den Dienst anbietet, wird ignoriert, da es nur in den ersten zwölf Monaten der Nutzung des Dienstes verrechnet wird. Bei S3 wird angenommen, dass die Standard Speicherklasse verwendet wird und Volumenrabattierungen bei Datenvolumina >50TB im Monat nicht relevant sind.[184] Andere Speicherklassen sind günstiger, somit gibt die Schätzung eine Obergrenze für die S3 Preise. Es wurde ebenfalls keine eigene Logik in Notebooks kalkuliert, welche 0,36$ pro Stunde und Compute Unit im Betrieb kosten würden. Dies ist dem Fakt geschuldet, dass für dieses Beispiel die Analysen mit SQL bewältigbar sind.

### 4.4.2 Auswahl

Im Bereich Multimode gibt es nur einen zu vergleichenden Dienst, weshalb Abzüge nicht auf relativen Kriterien zu den anderen Diensten basieren, sondern auf absoluten Abzügen.

[181] Vgl Amazon Web Services, Inc. o.J.(x)
[182] Vgl. Amazon Web Services, Inc. o.J.(v)
[183] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(w)
[184] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(p)

| Kriterium | max. Punkte | AWS IoT Analytics |
|---|---|---|
| Übertragbarkeit zwischen Clouds (ISO 9126) | 1 | 0 |
| Integration mit anderen AWS Diensleistungen | 3 | 3 |
| Generalisierung | 4 | 3 |
| Erweiterbarkeit | 4 | 4 |
| Fehlertransparenz/ Debugability | 5 | 4 |
| geringer Wartungsaufwand | 7 | 4 |
| Skalierbarkeit & „serverlessness“ | 7 | 6 |
| Kosten | 7 | 5 |
| Performancegarantien | 8 | 7 |
| Robustheit & Fehlertoleranz | 9 | 7 |
| Auswertungen (Unterabschnitt 2.1.2) | 11 | 11 |
| Summe | 66 | **54** |

Tab. 19: Bewertungsmatrix Multimode

Klar ist, dass zwar bei AWS IoT Analytics die Jupyter Notebooks plattformunabhängig sind, aber die SQL-Statements nicht übertragbar sind und zwingend ein AWS IoT Core Broker benötigt wird. Aufgrund dieses *vendor-lockins* kann ein großer Teil des Setups nicht übertragen werden. AWS IoT Analytics integriert sich gut mit anderen Dienstleistungen von AWS, z.B. mit AWS IoT Core und den anderen AWS IoT Diensten, SageMaker, QuickSight , Lambda, SNS und weiteren. AWS IoT Analytics ist speziell für IoT Daten gebaut, weshalb Funktionalitäten eingebaut sind, die speziell in Kombination mit den anderen AWS IoT Diensten Sinn machen. Die Verwendung dieser Funktionalitäten ist aber nicht zwingend notwendig. Es können auch andere Daten, z.B. IT-Monitoring eingespeist werden, wenn sie über MQTT und den AWS IoT Core Broker geladen werden. Die Erweiterbarkeit ist gegeben durch die Möglichkeiten, die SQL in Verbindung mit der generischen Programmierbarkeit der Notebooks bietet und zusätzlich dadurch, dass durch die integrierte Datenbank Daten erneut mit anderer Logik verarbeitet werden können. Durch Integration mit Cloudwatch, der Monitoring Lösung von AWS sind ansteigende Fehlerraten leicht zu entdecken. Einzig, dass eingehende Daten einige Zeit benötigen, bis sie von der AWS IoT Analytics Konsole angezeigt werden, kann die Fehlersuche erschweren.[185] Im Bereich „serverlessness“ wurden Abzüge getätigt, da Analyitcs Compute Units nicht selbstständig skalieren. Da die Abfragekosten mittels SQL proportional zu den anderen Kosten hoch erscheinen, wurden Abzüge gemacht. Die gesetzten Performancelimits des Dienstes erscheinen sinnvoll. Eine Fehlertoleranz ist gegeben, wenn der Anwender keine Fehler selbst einführt, oder AWS IoT

[185] Vgl. Amazon Web Services, Inc. 2020f

Analytics interne Fehler aufweist. Da alle Auswertugnen sogar in multiplen Wegen machbar sind, wurde die volle Punktzahl für diese Kategorie vergeben.

# 5 Modellierung

In diesem Kapitel sollen die Anforderungen aus den Interviews praktisch erhoben werden, die Referenzarchitekturen erstellt werden und die konstruierten Referenzarchitekturen folgend auch verglichen werden.

## 5.1 Anforderungserhebung

Wie in Abschnitt 2.2 beschrieben, müssen Referenzmodelle einen subjektiven Empfehlungscharakter besitzen, damit sie akzeptiert und wiederverwendet werden. Dafür muss ein Abgleich mit den Anforderungen der Nutzenden geschehen. Um dies zu erreichen, wurden im Anhang transkribierte Interviews (vgl. Anhang 1, Anhang 2, Anhang 3) durchgeführt. Daraus ergibt sich das in Abbildung 36 gezeigte Diagramm, welches die Anforderungen der individuellen Stakeholder an Dekompositionstiefe, Anwendbarkeit und Allgemeingültigkeit darstellt.

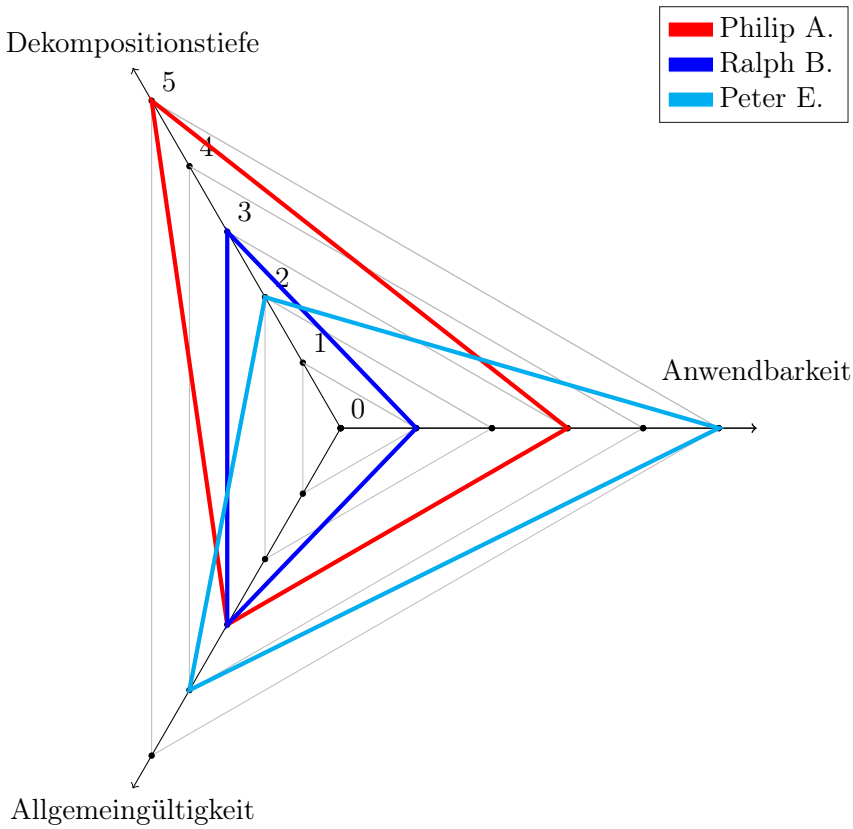


Abb. 33: Ergebnisse der Interviews

Durch die Interviews liessen sich folgende Durchschnitte errechnen: Dekompositionstiefe wurde im Schnitt mit $3,\overline{3}$ bewertet. Die Anwendbarkeit wurde ebenfalls mit $3,\overline{3}$ bewertet. Die Allgemeingültigkeit hingegen hat nur einen Schnitt von 3. Entsprechend sollten Dekompositionstiefe

und Anwendbarkeit priorisiert werden, während die Referenzarchitektur organisationsspezifischer sein darf. In den folgenden Referenzarchitekturen wird in mehrere Dekompositionen unterteilt. In der Datenverarbeitungssequenz werden mit Hilfe eines Sequenzdiagramms die Abläufe zur Datenverarbeitung mit dem zu betrachtenden Dienst, ausgehend von AWS IoT Core als Messagebroker gezeigt. Die Verteilungssicht soll sowohl die Interaktion der Dienste untereinander, als auch grob das durchzuführende Deployment zeigen. Die Bausteinsicht zeigt wichtige Elemente der einzelnen Dienste auf, die konkret untereinander interagieren. So ist die konkrete Untereinheit, die Daten von AWS IoT Core an andere Dienste versendet eine AWS IoT Core Rule, welche aufzuzeigen wäre. **To do** Genauere Auswirkungen (12)

Für die Operations wird vorausgesetzt, dass der AWS-native Dienst CloudWatch zur Überwachung eingesetzt wird.

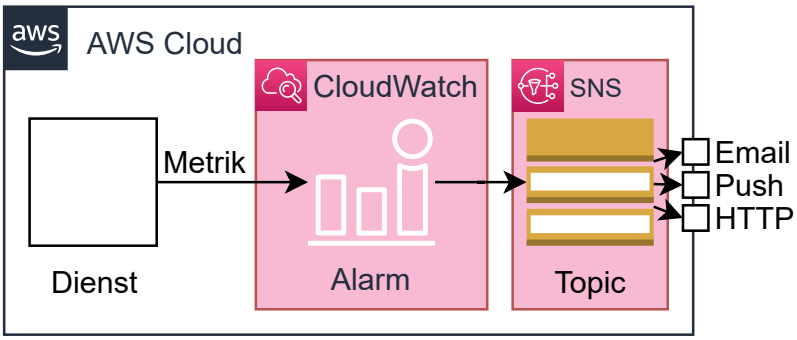


Abb. 34: CloudWatch Monitoring

CloudWatch erfasst zentralisiert Metriken aller Dienste und löst bei nutzerdefinierten Überschreitungen einen Alarm aus, welcher dann via SNS versendet werden kann (siehe Abbildung 34). Dabei kann bei Metriken mit hoher Varianz die Cloudwatch eigene Anomalienerkennung verwendet werden oder die Schwellwerterkennung.

Zusätzlich haben sich folgende Anforderungen ergeben:

- Anwendbarkeit auf Monitoringdaten (IT) (Anhang 1, Anhang 2)
- Anwendbarkeit auf Sensordaten (IoT) (Anhang 1, Anhang 2, Anhang 3)
- Wertschöpfung für das Unternehmen wichtig
- akzeptabel und problemlösend für Domäne
- Handling von Events, Messwerten und „Streaming“ (Anhang 2)

Im Folgenden werden die Referenzarchitekturen entworfen und miteinander verglichen, um mögliche Stärken und Einsatzgebiete zu identifizieren.

## 5.2 Echtzeitverarbeitung

Aufgrund des in Tabelle 12 durchgeführten Vergleiches, den Kinesis (Data Streams und Analytics) anführte, wird im Folgenden die Referenzarchitektur für die Echtzeitverarbeitung mit Kinesis Data Streams und Analytics entworfen. Diese Referenzarchitektur entspricht dem in Unterabschnitt 2.1.3 vorgestellten Konzept einer $\kappa$-Architektur.

### 5.2.1 Datenverarbeitungssequenz

In Abbildung 35 wird die durchlaufene Sequenz für eingehende Daten gezeigt. Nach initialer Übertragung via MQTT an AWS IoT Core, folgt die Übertragung in Kinesis Data Streams, welche die Daten gepuffert an Kinesis Data Analytics sendet.

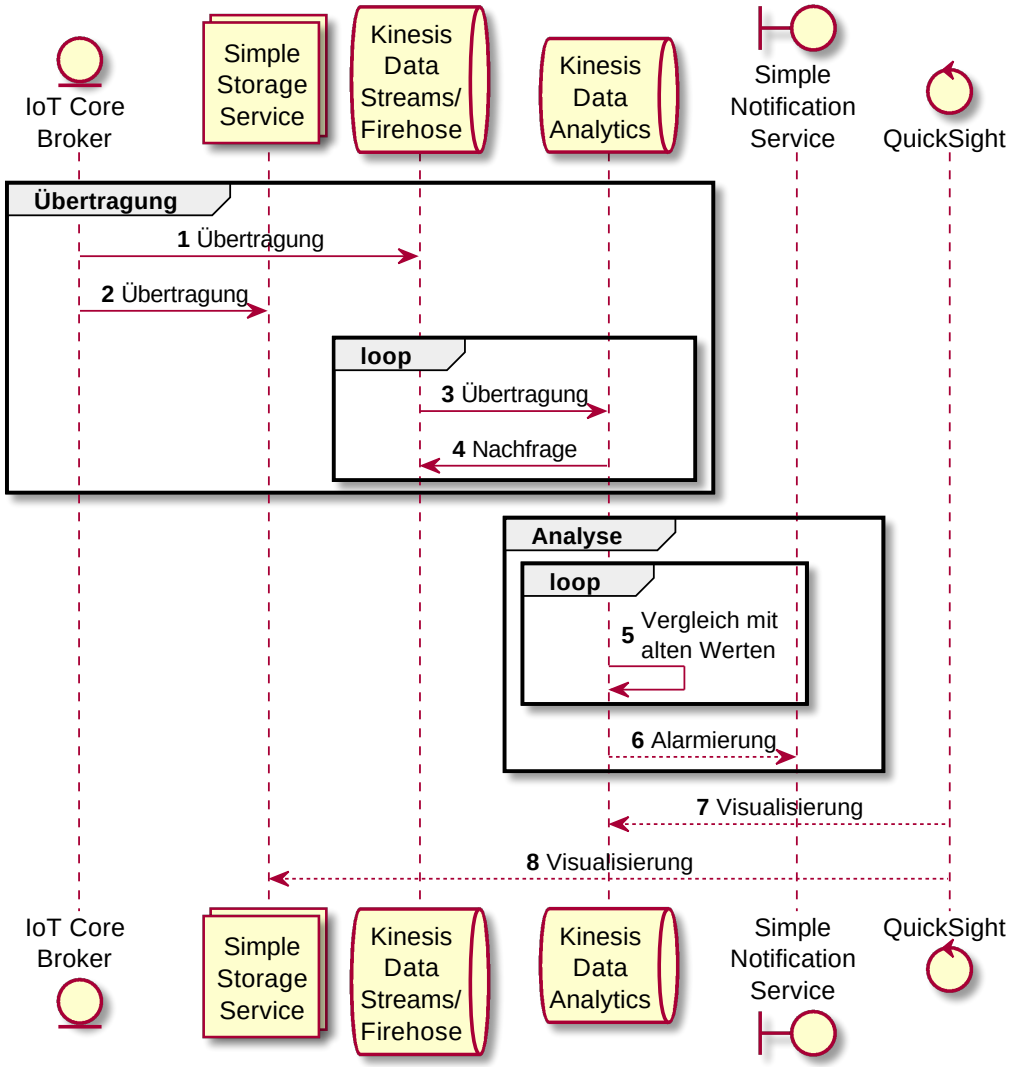


Abb. 35: Sequenzdiagramm Echtzeitreferenzarchitektur

### 5.2.2 Verteilungssicht

Folgend ist die Verteilungssicht der Echtzeitreferenzarchitektur gezeigt. Gemeinsame Variationspunkte dieser und folgender Dekompositionen bekommen den Buchstaben G und eine fortlaufende Nummer und werden nur einmal erklärt.

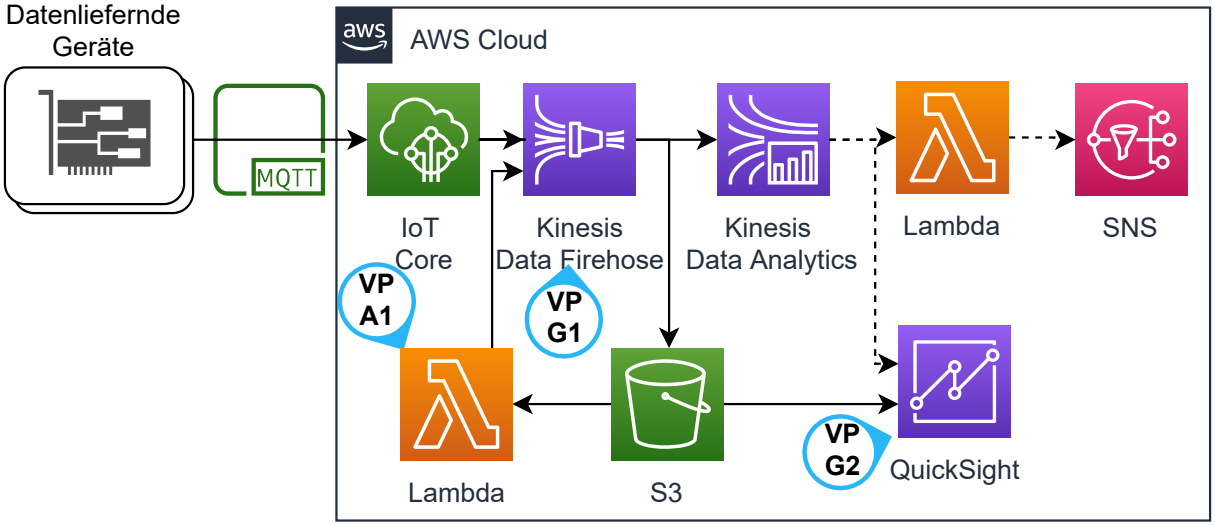


Abb. 36: Verteilungssicht mit Data Firehose

**Variationspunkt G1**: Je nach Anforderung kann Kinesis Data Firehose oder Kinesis Data Streams verwendet werden. Während die höhere Abstraktion und das einfachere Abrechnungsmodell von Kinesis Data Firehose einen reduzierten Wartungsaufwand hat, bietet Kinesis Data Streams mehr Kontrolle über unterliegende Faktoren wie Datenaufbewahrung und Durchsatz. Kinesis Data Firehose benötigt zwingend ein „Delivery ziel“. Dies kann beispielsweise ein S3-Bucket, eine Redshift Datenbank oder eine Elasticsearch Datenbank sein. In diesem Fall wurde aus Kosten- und Umsetzungserwägungen ein S3-Bucket gewählt.

**Variationspunkt G2**: QuickSight als AWS native Dashboardlösung ist gut geeignet, um schnell Übersicht in Datenanalysen aus Kinesis Data Analytics zu bekommen. Alternativ können auch andere Visualisierungslösungen wie Tableau eingesetzt werden, welche gegebenenfalls jedoch keinen (vollen) Zugriff auf Kinesis Data Analytics haben. Ein weiterer managed Service, den AWS für Dashboards anbietet, ist der Amazon Managed Service for Grafana, welcher das Open Source Visualisierungstool Grafana mit den AWS eigenen Metriken integriert.[186] In diesem Fall kann der S3 Bucket verwendet werden, um Dashboards über die Rohdaten zu erstellen.

**Variationspunkt A1**: Sollte es nicht gewünscht sein, Daten erneut in Kinesis Data Firehose einzuspielen, kann auf die Lambdafunktion verzichtet werden. Diese liest, wenn manuell aktiviert, den S3-Speicher ein und spielt die erfassten Nachrichten erneut in der selben Sequenz in Kinesis Data Firehose ein. Notwendig wird diese Lambda, wenn historische Daten mit abweichender Analyselogik analysiert werden sollen.

[186] Vgl. Dutt 2020

Im Folgenden wird die Verteilungssicht im zweiten Fall von Variationspunkt G1, der Verwendung von Kinesis Data Streams gezeigt. Die Auswahl von Kinesis Data Streams ist insbesondere angezeigt, wenn die Nachrichten direkt aufbewahrt werden sollen und direkter Einfluss auf die Performance erwünscht ist.

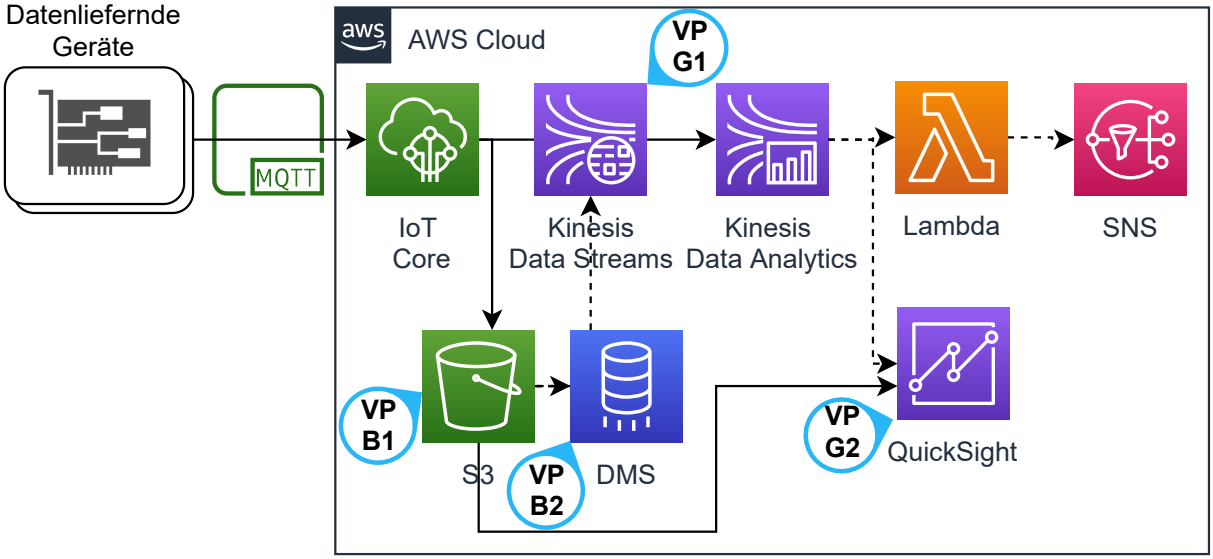


Abb. 37: Verteilungssicht mit Data Streams

**Variationspunkte G1, G2**: Siehe oben: Variationspunkt G1, Variationspunkt G2

**Variationspunkt B1**: Rohdaten in S3 zu speichern kann Sinn machen, um die Daten später noch einmal analysieren zu können. Nimmt man aber die Theorie der Datenhalbwertszeit zur Hilfe, macht es vielleicht Sinn die Daten statdessen maximal 48h in Kinesis Data Streams zwischenzuspeichern und auf S3 zu verzichten. Mittels der erweiterten Aufbewahrung [*Data Retention*] können Analysen mehrfach im Fehlerfall angefordert werden. Da die Preise nach sieben Tagen Aufbewahrung ansteigen und für Aufbewahrung und Abruf doppelt abgerechnet wird, ist zu empfehlen, nach sieben Tagen die Daten zu verwerfen.[187] Dieser Variationspunkt ist abhängig vom Variationspunkt G1, da Kinesis Data Firehose keine erweiterte Aufbewahrung unterstützt und die Daten in S3 abgelegt werden müssen.

**Variationspunkt B2**: Database Migration Service (DMS) ist in diesem Szenario dafür gedacht, einmal abgelegte Daten in S3 wieder in Kinesis Data Streams einspielen zu können. Je nach Szenario ist dies, wie bei Variationspunkt B1 schon erläutert, nicht notwendig.

### 5.2.3 Bausteinsicht

Unter Berücksichtigung von Variationspunkt G1 gibt es zwei Bausteinsichten. Dies ist bedingt durch die sich ergebenden architekturellen Änderungen, beim Einsatz von Kinesis Data Streams

[187] Vgl. Amazon Web Services, Inc. o.J.(j)

oder Kinesis Data Firehose.

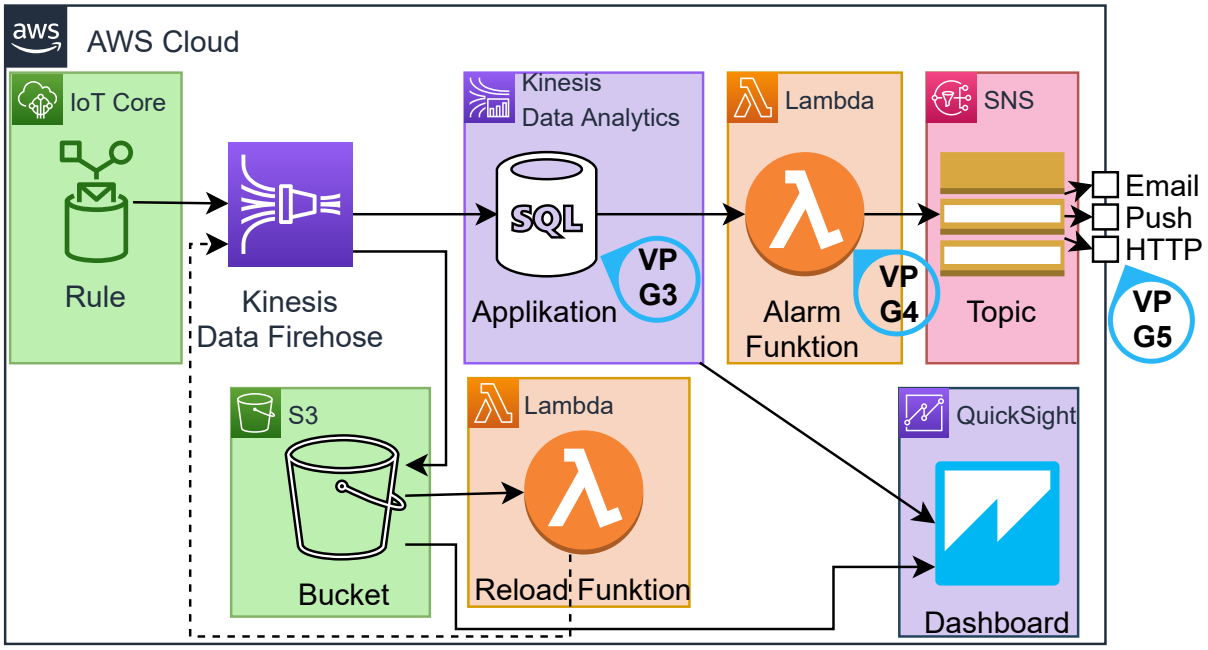


Abb. 38: Bausteinsicht mit Data Firehose

**Variationspunkt G3**: Der SQL Programmcode, der in Kinesis Data Analytics läuft ist anzupassen. So sind Verarbeitungsfenster, Attributsnamen und aufgerufene Funktionen nach Anforderung zu ändern. Andernfalls kann auch die Funktionalität zur Ausführung eigenen Codes via Apache Flink in Kinesis Data Analytics genutzt werden (dies erlaubt Ausführung von Java, Scala, Python). Dies ist angezeigt, wenn der SQL-Dialekt die gewünschten Auswertungen nicht unterstützt, oder eine eigene Implementierung vorgesehen ist.

**Variationspunkt G4**: Aufgrund der Notwendigkeit einer Lambda Funktion, um Alarme zu versenden, kann der Code selbst gestaltet werden. Wichtig ist dabei, dass Kinesis Data Analytics die Zustellung von Datensätzen wiederholt, wenn die Lambdafunktion als Rückgabewert ein Array mit den Ids und dem Status wie folgt zurückgibt: `[{"recordId": "<ID>", "result": "DeliveryFailed"}]`.[188] Die übermittelten Alarme lassen dabei Möglichkeit zur Anpassung. So kann neben dem Titel der Nachricht auch der eigentliche Inhalt angepasst werden. Beispielhaft ist in Anhang 8 gezeigt, wie eine in JavaScript geschriebene Lambdafunktion aussehen könnte, die via Kinesis Data Analytics angesteuert wird. Diese Funktion gibt selbstständig fehlerhafte Nachrichten zur Wiederverarbeitung an Kinesis Data Analytics zurück, versendet SNS Alarme und kann via MQTT eine Shutdown Nachricht an das Gerät übermitteln.

**Variationspunkt G5**: SNS unterstützt mehrere Protokolle für die Übermittlung von Nachrichten. Es können HTTP Webhooks genauso wie mobile Pushbenachrichtigungen oder auch Emails versendet werden. Welches Protokoll mit welchem Verteiler zu wählen ist, muss im SNS Topic

[188] Vgl. Amazon Web Services, Inc. o.J.(be)

eingestellt werden. Innerhalb der Cloud Native Solution der SPIRIT/21 hat sich bewährt, den Versand via Email zu nutzen und als Ziel den Email-Verteiler eines Monitoring Teams innerhalb des Tools Microsoft Teams einzustellen. Microsoft Teams zeigt eingegangene Emails an den Emailverteiler dann als Chatnachricht innerhalb des Teams an und benachrichtigt alle Teilnehmenden.

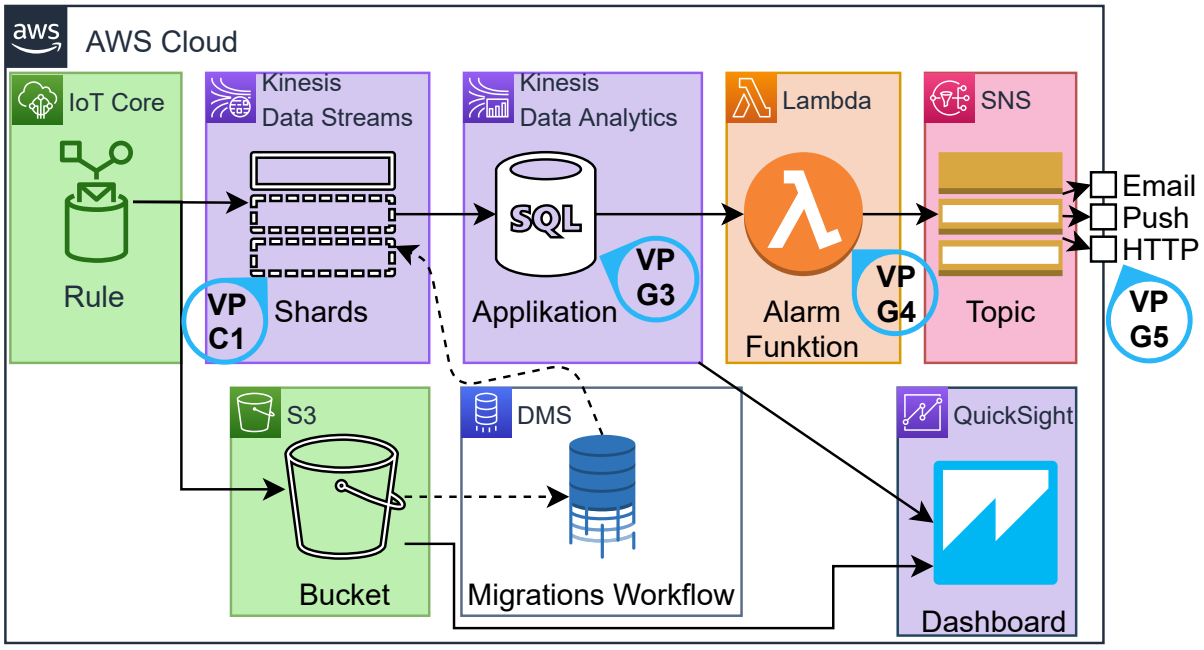


Abb. 39: Bausteinsicht mit Data Streams

**Variationspunkte G3, G4, G5**: Siehe oben: Variationspunkt G3, Variationspunkt G4, Variationspunkt G5

**Variationspunkt C1**: Die Anzahl an Shards ist essentiell für die Performance von Kinesis Data Streams. Für Workloads mit einem vorhersehbaren Workload ist Anzahl an Shards nach einem Preis/Leistungs Optimum zu ermitteln und zu konfigurieren. Wenn der Workload nicht vorhersehbar ist oder schnell skalieren können soll, sind Alarme im AWS eigenen Monitoring Tool CloudWatch zu erstellen. Im Beispielusecase für die Kostenschätzung ist ein einziger Shard (1MiB/Sekunde, 1000 Nachrichten eingehend) ausreichend. Dies ist bedingt, da Nachrichten mit einer Größe von 1KB, also ca. 1KiB geschrieben werden, bei einem Maximum von 200 pro Sekunde und einem Konsumenten. Es ist besonders auf die „WriteProvisionedThroughputExceeded“ Metrik zu achten, welche bei höheren Werten anzeigt, dass das Hinzufügen von zusätzlichen Shards angebracht wäre. Ebenfalls ist die Metrik „Incoming Records“ zu beachten. Verändert diese sich, deutet das auf einen Fehler im vorgelagerten AWS IoT Core oder in einem Teil der Datenlieferanten hin.

### 5.2.4 Anforderungen

- Anwendbarkeit auf Monitoringdaten (IT)
  Kinesis als System ist gut auf diverse Zeitseriendaten anwendbar. Problematisch ist das eigene Übertragungsformat, welches von Datenproduzenten verlangt, spezielle Schnittstellen zu implementieren. Der von AWS vorgesehene Weg, die Kinesis Producer Library ist in Java geschrieben und bindet eine ausführbare C++ Datei ein.[189] Im Einsatz mit Monitoringdaten würde dies erfordern, dass die Daten in einem Standardformat aggregiert und dann mittels eines in Java geschriebenen Programms transformiert werden müsste. CloudWatch bietet genau diese Funktionalität mittels der CloudWatch Metric Streams und der CloudWatch Log Subscriptions an.[190] Bei Metric Streams werden Metriken auf Wunsch in das OpenTelemetry oder das JSON Format konvertiert und dann an Kinesis Data Firehose zur Weiterverarbeitung übermittelt. Dabei werden aber nur Metriken erfasst, die einen Zeitstempel jünger als zwei Stunden haben, was manche Metriken, die einmal am Tag versendet werden ausschließt. Zusätzlich muss ein Metric Stream in jeder Region angelegt werden, wo CloudWatch Logs anfallen, was eine Herausforderung in stark verteilten AWS-Accounts darstellen kann. Metric Streams kosten 0,003$ pro 1000 verarbeitete Metriken und zusätzlich die entsprechend anfallenden Data Firehose Gebühren. CloudWatch Log Subscriptions bietet sowohl das Streamen an Kinesis Data Streams, als auch an Kinesis Data Firehose an.[191] Für jede Loggruppe, die einem Dienst oder einer einzelnen Ressource, wie beispielsweise einer Lambdafunktion zugeordnet sein kann, ist eine Subscription zu erstellen. Um dies zu erleichtern, sollte von Infrastructure as Code Gebrauch gemacht werden, um die Subscription automatisch für jede erstellte Loggruppe einzurichten. Es können benutzerdefinierte Filter eingerichtet werden, um nur relevante Logs zu übermitteln.

  Insgesamt scheint die Kinesis Dienstfamilie gut geeignet, um sowohl Logs als besondere Zeitreihendaten, als auch Metriken skalierbar zu verarbeiten.

- Anwendbarkeit auf Sensordaten (IoT)
  Kinesis ist generalisiert ausgelegt, durch die Integration mit AWS IoT Core wird jedoch die Verarbeitung von IoT Daten leicht gemacht.

- Handling von Events, Messwerten und „Streaming“
  Da die Verarbeitungslogik in Kinesis Data Analytics selbst zu schreiben ist, ist die unterschiedliche Behandlung von Events, niedrigfrequenten Messwerten und Streaming implementierungsabhängig. Dabei wäre es zu empfehlen ein Attribut in die übermittelten Nachrichten einzufügen, welches den geanauen Typ der Nachricht definiert und entsprechende Verarbeitungslogiken vereinfacht. Zu diesem Zweck soll das Attribut `{"messageType": "<string>"}` dienen. Für Events, die Definitionsgemäß keinen Messwert beinhalten, ist das Attribut mit dem Wert `{"messageType": "event"}` zu belegen. Messwerte sollen

[189] Vgl. Amazon Web Services, Inc. o.J.(bl)
[190] Vgl. auch im Folgenden Barr 2021b
[191] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(au)

`{"messageType": "meas_low_freq"}` als Wert verwenden. Für hochfrequentes Streaming ist `{"messageType": "meas_high_freq"}` zu verwenden.

- automatisierte operative Entscheidungen
  Automatisierte Entscheidungen bzw. Handlungen sind mit Kinesis Data Analytics möglich. So könnte die selbe Lambdafunktion, die für die Alarmierung benutzt wird, auch Aktionen auslösen. Vorstellbar wäre, dass die Lambdafunktion über MQTT Aktoren ansteuert, weitere Akteure informiert (z.B. die Werksfeuerwehr) oder selbstständig Anweisungen auslöst, die den Alarm beheben (so könnte bei niedrigem Batteriestand eine neue Batterie für einen Sensor geordert werden).

### 5.2.5 Operations

In Tabelle 20 sind die zu überwachenden Metriken von Kinesis Data Streams, SNS, AWS IoT Core, Kinesis Data Analytics und Kinesis Data Firehose gezeigt.[192,193,194,195,196]

| Dienst | Metrik | Ursache | Detektionsart |
|---|---|---|---|
| SNS | NumberOfNotificationsFailed | Dienstfehler | Schwellwert |
| AWS IoT Core | RuleMessageThrottled | Dienstfehler | Schwellwert |
| | Failure | Dienstfehler/ Benutzungsfehler | Schwellwert |
| Kinesis Data Analytics | MillisBehindLatest | Dienstfehler/ Benutzungsfehler | Anomalie |
| | LambdaDelivery.DeliveryFailedRecords | Dienstfehler/ Benutzungsfehler | Schwellwert |
| | LambdaDelivery.Duration | Dienstfehler/ Benutzungsfehler | Anomalie |
| Kinesis Data Streams (VP G1) | WriteProvisionedThroughputExceeded | Benutzungsfehler | Schwellwert |
| | ReadProvisionedThroughputExceeded | Benutzungsfehler | Schwellwert |
| | GetRecords.Latency | Dienstfehler | Anomalie |
| | PutRecords.ThrottledRecords | Dienstfehler/ Benutzungsfehler | Schwellwert |
| Kinesis Data Firehose (VP G1) | DeliveryToS3.Records/ DeliveryToS3.Success (Verhältnis) | Dienstfehler | Schwellwert |
| | ThrottledRecords | Dienstfehler | Schwellwert |
| | PutRecord.Latency | Dienstfehler | Anomalie |

Tab. 20: CloudWatch Metriken

Die Metriken werden von Kinesis einmal pro Minute an CloudWatch übermittelt.[197] Dies birgt die Gefahr, bei nicht konstanten Workloads, dass erst mit gewisser Verzögerung gehandelt wer-

[192] Vgl. Amazon Web Services, Inc. o.J.(am)
[193] Vgl. Amazon Web Services, Inc. o.J.(ao)
[194] Vgl. Amazon Web Services, Inc. o.J.(ac)
[195] Vgl. Amazon Web Services, Inc. o.J.(be)
[196] Vgl. Amazon Web Services, Inc. o.J.(ap)
[197] Vgl. auch im Folgenden Pogosova 2020

den kann. Bei besonders wechselhafter Last sollte also davon ausgegangen werden, dass nicht die tatsächliche Spitzenlast bekannt ist, sonder mit einem Aufschlag gearbeitet werden muss. Unterschieden wird in der Tabelle zwischen Fehlern, die auf den Dienst zurückzuführen sind und Fehlern, die durch Falschbedienung der Nutzenden entstehen können. Es ist auch möglich, dass Fehler durch mehrere verknüpfte Dienste kaskadieren und mehrere Metriken Alarme auslösen. Dies wäre beispielsweise der Fall, wenn sehr schnell viel mehr Nachrichten als im Normalzustand eingehen. Ausgehend von der Spalte Detektionsart können Alarme in CloudWatch aufgesetzt werden.

### 5.2.6 Know-how

Wie bereits beschrieben, bietet Kinesis Data Streams keine automatisierte Skalierung der Shards an. Dieses Problem wurde durch die Gemeinschaft aus Nutzenden und Programmierenden in vielerlei Art adressiert. Der von Nobile/Natali vorgestellte, AWS eigene Ansatz basiert auf CloudWatch Alarmen, die basierend auf den IncomingBytes und IncomingRecords Shards eine Lambda auslösen, welche die Skalierung verwaltet.[198,199] Pogosova kritisert, dass die Menge von fünf Diensten, die diese Lösung benötigt, kaum als autoscaling zu bezeichnen ist.[200] Stanley schlägt zur Lösung des Problems eine Lösung und eine Beispielimplementation in Python vor, die ebenfalls auf CloudWatch Alarmen basiert, aber eine MoM zwischenschaltet, die dann eine Lambdafunktion ausführt.[201] Prasath setzt auf einen vergleichbaren Ansatz wie Stanley, nur dass keine konkrete Implementierung vorgeschlagen wird.[202] Cui modifiziert den Ansatz unter Berücksichtigung des Faktes, dass das herunterskalieren von Shards teurer sein könnte, wenn der Datendurchsatz nicht genau bekannt ist.[203] Zur Mitigation schlägt Cui eine Herunterskalierung durch einen CloudWatch Auslöser vor, der erst 36h später auslöst. Allen Ansätzen eigen ist, dass eine Verzögerung wie in Unterabschnitt 5.2.5 geschildert, von 60 Sekunden bis zur Erfassung der aktuellen Metriken besteht. Dies birgt die Gefahr, dass für eine gewisse Zeit zu wenige Shards provisioniert sind. Aufgrund der Komplexität, ein passendes Autoscaling zu errichten ist es angezeigt, wenn die Performance von Data Firehose in Tests ausreicht, Data Firehose entsprechend dem Variationspunkt G1 zu verwenden.

Innerhalb des MQTT Protkolls, das AWS IoT Core in Teilen implementiert ist für die zwei Quality of Service (QoS) Modi 0 und 1 eine „at-least-once“ Semantik vorgesehen.[204] Der QoS Modus 2, welcher eine „exactly-once“ Semantik garantiert, wird von AWS IoT Core nicht unterstützt.[205] Zusätzlich ist eine „exactly-once“ Semantik bei der Ausführung von AWS IoT Core Rules nicht garantiert. Wenn doppelte Werte für Auswertungen nicht tolerierbar sind, muss

[198] Vgl. Nobile/Natali 2018
[199] Siehe auch: `https://github.com/aws-samples/aws-application-auto-scaling-kinesis`
[200] Vgl. Pogosova 2020
[201] Vgl. Stanley 2019
[202] Vgl. Prasath 2019
[203] Vgl. auch im Folgendn Cui 2017
[204] Vgl. auch im Folgenden OASIS Open Consortium 2014
[205] Vgl. Amazon Web Services, Inc. o.J.(aq)

entsprechend eine Deduplizierung eingeführt werden. Aufgrund des technischen Aufwandes, der hinter einer Deduplizierung und garantierter Idempotenz steht, muss genau abgewogen werden, ob die fachlichen Seite des Anwendungsfalls eine doppelte Verarbeitung mancher Records nicht tolerieren kann. So wäre beispielsweise ein doppelter Messwert, der eine Überschreitung anzeigt wenig kritisch. Bei Aggregationen wie dem gleitenden Durchschnitt verringert der Einfluss eines einzelnen doppelten Wertes eines Sensors sich mit wachsender Anzahl $n$ der angeschlossenen Sensoren. Eine mögliche Mitigation wäre, die Messzeit zusammen mit den Messwerten zur Deduplikation zu verwenden. Dabei ist zu beachten, dass die Möglichkeit besteht, dass die integrierte Uhr des Sensors falsch geht.

## 5.3 Batch Verarbeitung

Für diese Referenzarchitektur käme sowohl der Dienst der Multimode Klasse, AWS IoT Analytics, als auch Timestream, als bester der Batch-Klasse in Frage. Da Timestream im Vergleich besser abschnitt, wird im Folgenden die Referenzarchitektur mit Timestream konstruiert. Diese Referenzarchitektur entspricht dem in Unterabschnitt 2.1.3 vorgestellten Konzept einer OLAP-Architektur.

### 5.3.1 Datenverarbeitungssequenz

In Abbildung 40 ist die Datenverarbeitungssequenz der Referenzarchitektur zu sehen. Die Daten werden mittels einer properitären Verbindung von AWS IoT Core an Timestream überspielt und von Timestream gespeichert. In einem regelmäßigen Intervall (ähnlich zu den *cron-jobs* auf Linux) wird eine Lambda Funktion aufgerufen. Diese fragt Timestream ab, interpretiert die Resultate und löst im Alarmfall Nachrichten an SNS aus. QuickSight greift als Visualisierungslösung, wenn gewünscht, auf den gesamten Datenbestand von Timestream zu. Dies passiert entweder beim Abruf von nutzerersstellten Dashboards oder durch periodisches Laden von Daten in den Quickshight eigenen Cache, genannt *SPICE*.

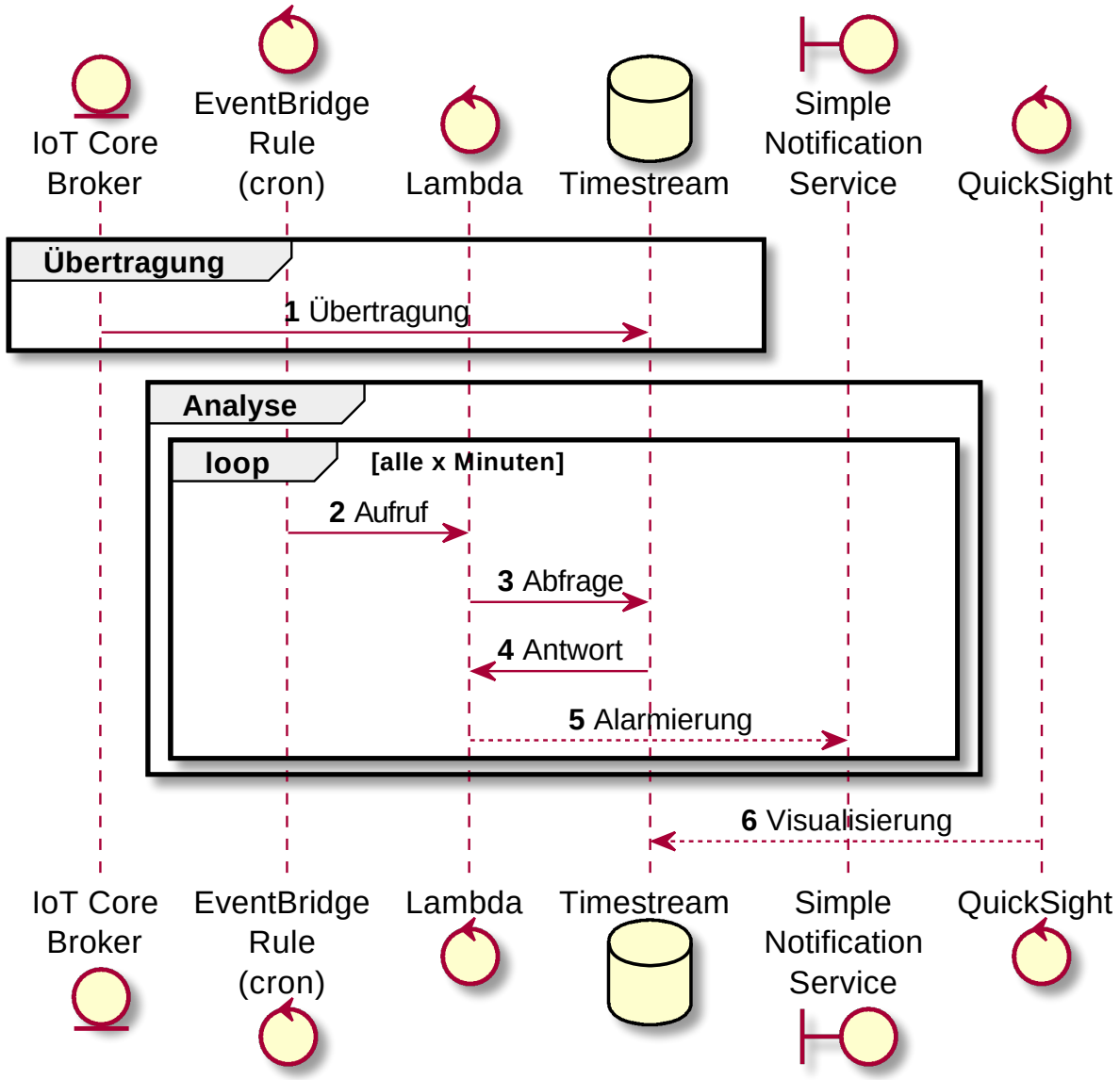


Abb. 40: Sequenzdiagramm Batch Verarbeitung

### 5.3.2 Verteilungssicht

In der folgenden Abbildung 41 ist die Verteilungssicht der Referenzarchitektur gemeinsam mit den Variationspunkten gezeigt. Variationspunkte mit Präfix G können dabei auch auf bereits definierte Variationspunkte der Echtzeitreferenzarchitektur referenzieren.

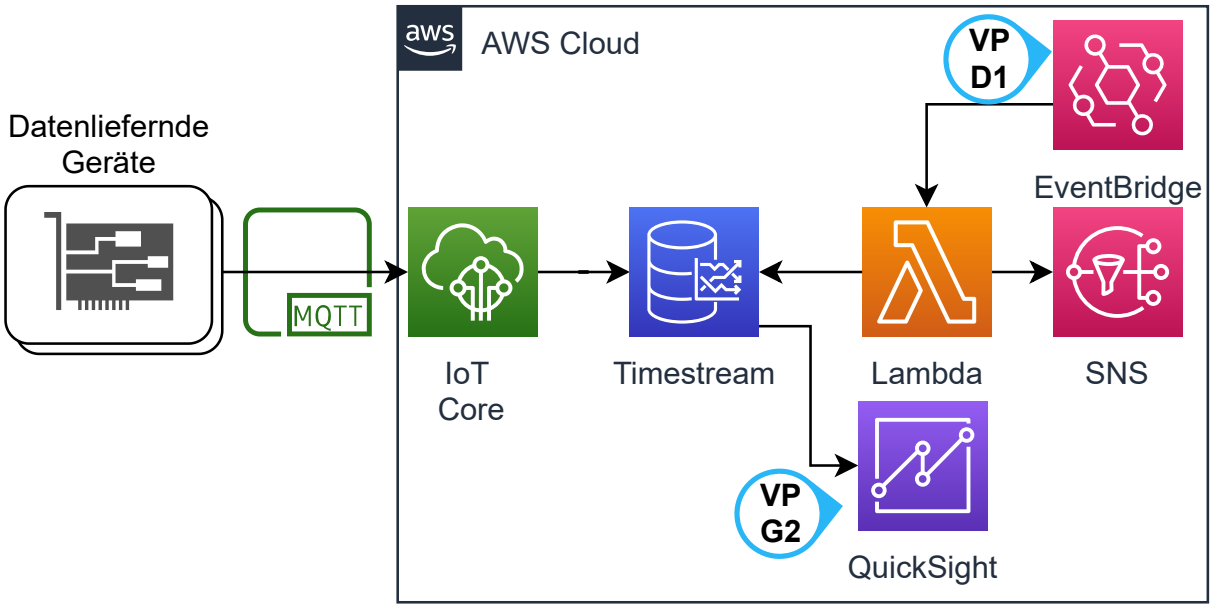


Abb. 41: Verteilungssicht

**Variationspunkt D1**: EventBridge bietet den zeitlich geplanten Aufruf von Zielen wie Lambda an. Wenn keine kontinuierliche Überwachung gewünscht ist, kann die Lambda auch auf Bedarf ausgelöst werden. Dies wäre beispielsweise durch ein vorgelagertes API Gateway möglich. Über dieses muss dann der Zeitraum übergeben werden, für welchen die aktuelle Analyse durchgeführt werden soll.

**Variationspunkt G2:** Hier wird eine Dashboardinglösung, im speziellen der AWS eigenen Dienst QuickSight vorgesehen. Dies erfolgt, damit Benachrichtigungen, die via SNS versendet werden, leicht für die Benachrichtigten nachvollziehbar sind. Wenn also beispielsweise eine Anomalie erkannt wurde, kann dies mit einer Visualisierung nachvollzogen werden, um dann entsprechend zu handeln. Wichtig ist, dass die Visualisierung sowohl von den Originaldaten aus dem Speicher von AWS IoT Analytics gespeist wird, als auch aus der Analyse. Das für und wieder des Einsatzes wurde auch bereits in Variationspunkt G2 der Echtzeitarchitektur diskutiert. Alternativ ist Timestream auch mit Grafana und damit dem Managed Service von Grafana integriert.[206,207] Grafana kann dabei ebenfalls als Dashboardinglösung verwendet werden und kostet dabei das selbe wie die Standard Edition von Quicksight, nämlich 9$ für Nutzende monatlich.[208,209] Erweiterte Kapazitäten bei QuickSight kosten 18$ pro Monat für Nutzende.

[206] Vgl. Amazon Web Services, Inc. o.J.(aj)
[207] Vgl. Dutt 2020
[208] Vgl. auch im Foglenden Amazon Web Services, Inc. o.J.(n)
[209] Vgl. Amazon Web Services, Inc. o.J.(l)

### 5.3.3 Bausteinsicht

In der folgenden Abbildung 42 wird die Bausteinsicht als tiefere Dekomposition der Verteilungssicht, samt Variationspunkten dargestellt.

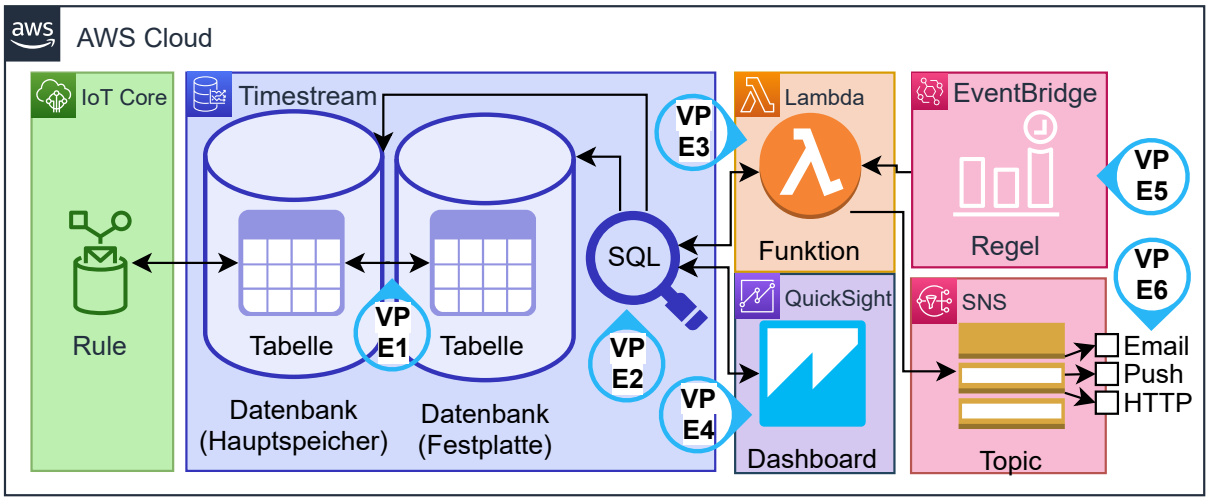


Abb. 42: Interagierende Dienstelemente

**Variationspunkt E1**: Timestream bietet aktuell zwei Speicherklassen an und hat eine Speicherklasse, die für die Zukunft angekündigt ist. Aktuell verfügbar sind RAM-basierter Speicher, der teurer ist und magnetischer Hard Disk Drive (HDD)-Speicher, der günstiger ist. Mittels sogenannter *retention policies* können Daten zwischen den Speicherklassen verschoben werden.[210] Dabei können Daten vom RAM-Speicher in den HDD-Speicher verschoben werden und vom HDD-Speicher gelöscht werden. Beim finalen Löschen der Daten sind die jeweiligen Anforderungen an Datenaufbewahrung der instanziierenden Architektur zu beachten. So können beispielsweise Langzeitanalysen notwendig sein, die auf einen Datenbestand von mehreren Monaten oder Jahren zugreifen müssen. Da Timestream aber auch nach Speicher abgerechnet wird, sind *retention policies* sowohl von RAM zu HDD als auch zur Löschung von Inhalt vom HDD-Speicher einzurichten. Zu beachten ist, dass die Gesamtaufbewahrungsdauer sich aus der Summe beider Aufbewahrungszeiten ergibt. Die Einstellung der RAM zu HDD retention policy ist abhängig von Variationspunkt E2 und Variationspunkt E5, da je nach Abfragerhythmus und abgefragter Datenmenge für Analysen eine verlängerte Aufbewahrung im RAM zur beschleunigten Analyse sinnvoll ist. Für die SQL-Abfragen wird kein Unterschied zwischen Speicherart gemacht, bis auf die technisch bedingte höhere Ausführugnsdauer auf HDD-Speichern.

**Variationspunkt E2**: SQL-Abfragen in Timestream sind auf mehrere Arten optimierbar. Zum einen wird die abgefragte Datenmenge in Rechnung gestellt, was eine präzise Einschränkung der Abfrage mit `WHERE` Bedingungen und einem genauen `SELECT` notwendig macht. Zusätzlich ist die Datenabfrage in zwei Modellen möglich: dem flachen Modell und dem Zeitreihenmodell.[211]

[210] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(ax)
[211] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(as)

Das flache Modell schreibt, wie in Tabelle 21 gezeigt, für jeden Messwert eine eigene Zeile. Entsprechend muss der Messwert in der `WHERE` Bedingung spezifiziert werden.

| time | Dimension A (Sensorname) | measure_name | measure_value ::double | measure_value ::bigint |
|---|---|---|---|---|
| 2021-05-10 23:59:59 | sensora | co2 | null | 500 |
| 2021-05-10 23:59:59 | sensorb | temperature | 25.5 | null |

Tab. 21: Beispiel flaches Datenabrufmodell

Gegensätzlich dazu gibt das Zeitreihenmodell, welches sich beispielsweise mit der `CREATE_TIME_SERIES` Funktion erzeugen lässt JSON Arrays zurück. Diese sehen wie folgt aus: `[{"time":"2021-05-10 23:59:59","value":500}]`. Timestream bietet für diese Zeitreihen spezielle Funktionen, wie beispielsweise die Interpolation fehlender Werte an.

**Variationspunkt E3**: Der Programmcode und die Laufzeit der Lambdafunktion in diesem Variationspunkt sind je nach Anforderungen und Kentnissen des Implementierungsteams bei der Instanziierung dieser Referenzarchitektur anzupassen/neu zu schreiben. In Anhang 9 ist beispielhaft eine Lambdafunktion, geschrieben in Javascript für die Node.js Laufzeitumgebung gezeigt. Diese Lambdafunktion sendet eine SQL-Abfrage zur Zählung von schwellwertüberschreitenden Werten einzelner Sensoren. Folgend werden die Ergebnisse ausgewertet und für jeden Sensor mit überschreitenden Messwerten eine Benachrichtigung via SNS und ein *shutdown*-Befehl an den Sensor via MQTT versendet.

**Variationspunkt E4**: Unabhängig von dem in Variationspunkt G2 gewählten Dienst ist es möglich, individuelle Dashboards mit unterschiedlichen Interaktionsmöglichkeiten zu erstellen. So können beispielsweise sogenannte *drill-downs* den Nutzenden Entscheidungsträgern helfen, dynamisch die Datenbereiche anzupassen. Ein Dashboard in QuickSight kann beispielsweise wie folgt aussehen:

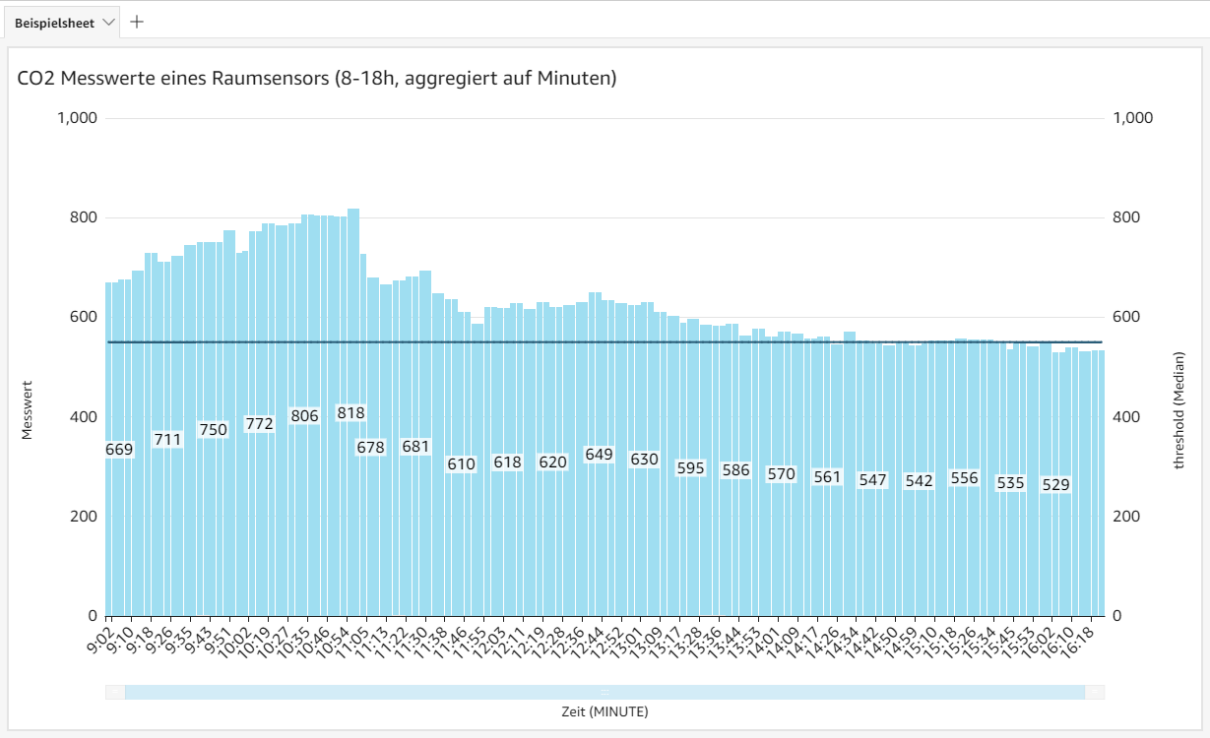


Abb. 43: Dashboard in Quicksight

**Variationspunkt E5**: EventBridge regeln können unterschiedlich eingestellt werden. Abhängig von den Anforderungen der instanziierenden Architektur, kann entweder ein Aufruf alle x Minuten, Stunden oder Tage eingestellt werden, oder für mehr Anpassung eine Rate mittels einer abgewandelten, *cron*-Syntax konfiguriert werden. Ein cron-Ausdruck um die Regel alle 10 Minuten an Werktagen zwischen 9 und 17.00 Uhr auszulösen sähe wie folgt aus: `0/10 9-17 ? * MON-FRI *`. Der Ausdruck wird in Universal Time Coordinated (UTC)-Zeit ausgeführt, was eine ein- oder zweistündige Verschiebung von der deutschen Zeit bedeutet.

**Variationspunkt G5**: Siehe Variationspunkt G5

### 5.3.4 Anforderungen

- Anwendbarkeit auf Monitoringdaten (IT) Timestream unterstützt wie die Kinesis Familie diverse Zeitreihendaten. Es ist ebenfalls prinzipiell möglich Logdaten aufzubewahren, da der interne Datentyp *varchar* Strings mit bis zu 2 GB Länge erlaubt.[212] Da aber große Datenmengen in einzelnen Variablen durch die doppelten Kosten für Speicherung und Abfrage sich weniger eignen als beispielsweise numerische Messwerte, sollte auf die Anwendung auf Logs verzichtet werden.

[212] Vgl. Amazon Web Services, Inc. o.J.(az)

Die Nutzung zur Speicherung von Metriken hingegen ist explizit im Rahmen von *DevOps*-Anwendungsfällen für Timestream vorgesehen.[213,214] Ohne spezialisierte Applikation, die beispielsweise in Kinesis Data Analytics mit Flink Laufzeitumgebung laufen könnte und von Kinesis Data Firehose gespeist wird, ist es jedoch nicht möglich CloudWatch Metriken in Timestream zu laden.[215] Alternativ können Metriken wie von Pochiraju vorgeschlagen, über MQTT und AWS IoT Core in Timestream eingespeist werden.[216] Das eine eigene Applikation verwendet werden muss, um Daten via Kinesis oder MQTT einzuspeisen, wertet die Tauglichkeit für Monitoring-Usecases gegenüber der Echtzeitarchitektur ab.

- Anwendbarkeit auf Sensordaten (IoT)
- Handling von Events, Messwerten und „Streaming“

  Ingestionlatenz Beispiel

- automatisierte operative Entscheidungen
  Wie in Anhang 9 gezeigt möglich.

### 5.3.5 Operations

In Tabelle 22 sind die zu überwachenden Cloud-Watch Metriken der in der Referenzarchitektur verwendeten Dienste gezeigt. Dies umfasst TimeStream, Lambda, SNS, AWS IoT Core und EventBridge.[217,218,219,220,221]

[213] Vgl. Amazon Web Services, Inc. o.J.(r)
[214] Vgl. Das/Rath 2020
[215] Vgl. Riddle/Patana-anake 2021
[216] Vgl. Pochiraju 2020
[217] Vgl. Amazon Web Services, Inc. o.J.(bb)
[218] Vgl. Amazon Web Services, Inc. o.J.(al)
[219] Vgl. Amazon Web Services, Inc. o.J.(ao)
[220] Vgl. Amazon Web Services, Inc. o.J.(ac)
[221] Vgl. Amazon Web Services, Inc. o.J.(e)

| Dienst | Metrik | Ursache | Detektionsart |
|---|---|---|---|
| SNS | NumberOfNotificationsFailed | Dienstfehler | Schwellwert |
| AWS IoT Core | RuleMessageThrottled | Dienstfehler/ Benutzungsfehler | Schwellwert |
| | Failure | Dienstfehler/ Benutzungsfehler | Schwellwert |
| TimeStream | SystemErrors | Dienstfehler | Schwellwert |
| | UserErrors | Benutzungsfehler | Schwellwert |
| | SuccessfulRequestLatency | Dienstfehler/ Benutzungsfehler | Anomalie |
| Lambda | Duration | Dienstfehler/ Benutzungsfehler | Anomalie |
| | Errors | Dienstfehler/ Benutzungsfehler | Schwellwert |
| | Throttles | Benutzungsfehler | Schwellwert |
| EventBridge | FailedInvocations | Dienstfehler | Anomalie |
| | ThrottledRules | Dienstfehler | Schwellwert |

Tab. 22: CloudWatch Metriken

AWS IoT Core, SNS und Timestream übermittelt Metriken in einminütiger Auflösung.[222,223,224] Bei Lambda können die übermittelten Metriken bis nach Ende einer Ausführung übermittelt werden, weshalb die Auflösung der Metriken unpräzise sein kann. Insgesamt lässt sich aufgrund der teilweise verketteten Metriken leicht erkennen, wenn kaskadierende Fehler auftreten. So sind beispielsweise *FailedInvocations* von EventBridge ein Hinweis darauf, dass es Probleme bei der Ausführung von Lambda Funktionen gab.

### 5.3.6 Know-how

Während Timestream zum Start 2020 allein in Irland (eu-west-1) für die EU verfügbar war, ist Timestream für die Region eu-central-1 (Frankfurt) mittlerweile verfügbar.[225,226] Dies erleichtert die Integration in bestehende Dienste, die bereits in Frankfurt provisioniert wurden.

Zeitreihen mit Messwerten und Metadaten werden innerhalb von Timestream als Dimensionen oder Messwerte hinterlegt. Dimensionen und Messwerte haben jeweils einen Namen und einen Wert. Dabei sind als Dimensionen alle Metadaten des übermittelnden Sensors oder der übermittelnden Applikation denkbar. Beispiele für solche Dimensionen wären Name oder Standort, während Messwerte beispielsweise $CO_2$ Messungen, Temperaturen, CPU Auslastung oder RAM-Verbrauch sein können. Timestream rechnet pro Schreibzugriff die Summe aller Namen von

[222] Vgl. Amazon Web Services, Inc. o.J.(ac)
[223] Vgl. Amazon Web Services, Inc. 2021b
[224] Vgl. Amazon Web Services, Inc. o.J.(bb)
[225] Vgl. Amazon Web Services, Inc. 2020d
[226] Vgl. Amazon Web Services, Inc. o.J.(s)

Dimensionen und Messwerten und deren Werte (inklusive der Zeitdimension mit aktuellem Zeitstempel) als Speicherplatz ab.[227] Bei gruppierten Schreibzugriffen können gemeinsame Attribute und Werte zusammengefasst werden, um die Anzahl an Schreibzugriffen zu vermindern. Aus der Unterteilung in Messwerte und Dimensionen ergeben sich auch Optimierungen der Datenmodellierung. So gibt es beispielsweise die Möglichkeit, derivative Werte, sofern sie eine niedrige Kardinalität besitzen als Dimensionen statt als Messwerte zu speichern.[228] Dies bringt aber den Nachteil mit sich, dass entsprechend drei verschiedene Zeitreihen entstehen würden, was bei der Abfrage zu beachten ist. Da Dimensionsnamen ebenfalls Speicherplatz verbrauchen, sind die Namen entsprechend des KISS-Prinzips/Ockhams Rasiermesser zu gestalten und die einfachste und kürzeste Variante auszuwählen. Bei der Datenmodellierung ist auch zu beachten, dass Messwerte entsprechend ihres eigentlichen Datentyps gespeichert werden und keine unbeabsichtigte Konversion in z.B. das *varchar* Format auftritt. Da Messwerte bei Erstanlage eines Messwertes in einer Tabelle fest mit einem Datentypen assoziiert werden, ist eine Neuanlage mit entsprechender Datenmigration notwendig um Fehler zu korrigieren.

## 5.4 Einsatzszenarien der Referenzarchitekturen

Im Folgenden soll beleuchtet werden, wofür sich welche der beiden entwickelten Referenzarchitekturen besser eignet. Für das Monitoring eignet sich die in Abschnitt 5.2 beschriebene Referenzarchitektur besser, da eine enge Integration sowohl mit CloudWatch Metrics als auch CloudWatch Logs besteht. So lassen sich Analysen zu in der AWS-Cloud laufenden Workloads einfach durchführen.

**To do** Vergleich RA - Deadline 03.05. (13)

- Wertschöpfung für das Unternehmen wichtig
  **To do** Interview (14) Siehe Anhang 4
- akzeptabel und problemlösend für Domäne
  **To do** ausfüllen (15)

Da die Referenzarchitekturen als verteilte Systeme mit mehreren Diensten diverse Fehlerquellen haben können, die sich dann symptomatisch in den gezeigten Metriken bemerkbar machen, sind genaue Tests zum Verständnis der fehlerzustände wichtig. Im Rahmen des *Chaos Engineerings* lassen sich gezielt Fehler in das System einführen um die Resilienz gegenüber diversester Fehlerquellen zu testen.[229] Bei den vorgestellten Referenzarchitekturen ist es zu empfehlen, regelmäßige Chaos Experimente durchzuführen um gezielt Fehlerszenarien wie doppelte Nachrichten, hohe Latenzen zwischen Diensten oder die temporäre Nichtverfügbarkeit einzelner Dienste zu simulieren und gezielt den Einfluss messen zu können. Folgend können Erkentnisse für den besseren

[227] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(bk)
[228] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(af)
[229] Vgl. Augsten 2020

Betrieb einer resilienten Dateninfrastruktur abgeleitet udn dokumentiert werden. Für AWS gibt es den Dienst Fault Injection Simulator, welcher Fehler in Schnittstellen zwischen Diensten erzeugen kann und gezielt auch einzelne Infrastrukturkomponenten beeinträchtigen kann.[230] Unter Verwendung dieses Dienstes ist es möglich, die beschriebenen Chaos Experimente durchzuführen.

[230] Vgl. Barr 2021a

# 6 Schlussbetrachtung

**To do** Schluss schreiben (16)

## 6.1 Fazit

## 6.2 Handlungsempfehlung

## 6.3 Ausblick

Innerhalb dieser Arbeit wurden Referenzarchitekturen für die Verarbeitung in der Cloud konstruiert. Ein Trend, der dabei ausgespart wurde, weil sich wichtige Komponenten nicht in der Cloud befinden, ist das sogenannte „Fog computing“. Nach der Definition von Vaquero/Rodero-Merino ist Fog computing ein Szenario, in dem heterogene, allgegenwärtige und dezentralisierte Geräte kommunizieren und kooperieren um Speicher- und Verarbeitungsaufgaben zu übernehmen.[231] In der Praxis führt dies dazu, dass Verarbeitungsaufgaben in Teilen, angelehnt an das „Edge computing“ von der Cloud in Richtung der Geräte ausgelagert wird.[232] Dies geschieht dabei beispielsweise an Netzwerkgateways, die sowieso mit der Cloud kommunizieren und folgend nur noch bereits ausgewertete Daten übertragen. AWS bietet mit dem Dienst Greengrass bereits eine Softwareplattform an, die auf diversen qualifizierten Gateways läuft.[233] Durch die lokale Ausführung von Code könnten Schwachstellen adressiert werden, wie beispielsweise schlechte Netzkonnektivität. So kann Code, der auf Greengrass ausgeführt wird in Form von Containern oder lokalen Lambdafunktionen Benachrichtigungen lokal ohne Konnektivität zur Cloud versenden und beispielsweise Aktoren auslösen. Dies würde die gezeigten Referenzarchitekturen ergänzen. Dass mit Greengrass Anomaliedetektion möglich ist, wurde auch von Shankar u. a. gezeigt.[234]

Publikation intern - Confluence (und ggf. GitHub?)

[231] Vgl. Vaquero/Rodero-Merino 2014, S. 30 f.
[232] Vgl. Bonomi u. a. 2012
[233] Vgl. auch im Folgenden Amazon Web Services, Inc. o.J.(ab)
[234] Vgl. Shankar u. a. 2020

# Anhang

## Anhangverzeichnis

# Anhang 1: Experteninterview Philipp A.

| | |
|---|---|
| Datum | 22.03.2021 |
| Thema | Initiales Anforderunginterview |
| Teilnehmende,<br>Position | Lukas Fruntke, Verfasser<br>Philipp A., Cloud Solution Architekt - NCS |

**Lukas F.:** Herzlich willkommen zum Interview und vielen Dank, dass du dich als Interviewpartner bereit gestellt hast

**Philipp A.:** Selbstverständlich.

**Lukas F.:** Ich habe direkt eine Frage an dich: Was ist deine Rolle innerhalb der SPIRIT und was ist deine Rolle im Spezifischen mit Perspektive auf die Referenzarchitekturen, die es zu entwickeln gilt?

**Philipp A.:** Ich sitze in der Spirit auf dem Posten des Cloud Solution Architects speziell für AWS. Diese Rolle fülle ich auch in der Native Cloud Solution aus. Das heißt, ich bin für Software und Infrastruktur Struktur Architekturen in dem Projekt/der Solution verantwortlich und kümmere mich darum, dass die Implementierung so gut wie möglich voran gehen kann. Dabei sollen keine Architekturprobleme verursacht werden. Ansonsten berate ich die Entwicklung bei technischen Fragen und Implementierungsfragen, die auftreten.

**Lukas F.:** Wenn wir jetzt speziell Richtung Referenzarchitektur schauen, würdest du dich dann eher als Nutzenden sehen oder eher als jemand, der zwar einen „Stake“ hat, dass es eine gute der Referenz Architektur wird, aber sie nicht konkret anwenden würde?

**Philipp A.:** Ich sehe mich auf beiden Seiten. Sowohl als Nutzenden, weil ich werde auf Basis der Referenzarchitekturen unsere Serverless Solutions designen, also die für uns und unsere Kunden. Ich glaube aber auf der anderen Seite auch, dass ich mitwirke an der Ausarbeitung.

**Lukas F.:** Verstehe, das ist schon mal aufschlussreich. Insgesamt, wir reden ja über Referenzarchitekturen, wo siehst du denn Anwendungsgebiete? Gibt es konkrete Customer Cases, wo wir jetzt auf Zeitreihendaten speziell schauen, oder gibt es da irgendwelche Gebiete, wo du sagst: „Oh, da könnte es besonders relevant sein“?

**Philipp A.:** Ja natürlich. Der größte/aktuellste Fall bei uns sind tatsächlich Sensordaten und Messdaten im IIoT Umfeld, bei denen wir solche Zeitreihendaten immer haben. Aber auch sämtliche Monitoring Daten, die von cloudbasierten Metriken abgezogen werden. Von diesen Metriken gibt es viele.

**Lukas F.:** So wie ich es verstehe, sowohl interne Anwendungsfälle mit Monitoring als auch Sachen, die für Kunden jetzt direkt relevant sind, wie IIoT-/Sensordaten

**Philipp A.:** genau

**Lukas F.:** Das wären denke ich so die Bereiche, in denen man die Referenz Architekturen anwenden würde/ spezialisieren würde.

**Philipp A.:** Da wir uns, in der SPIRIT eben auch mit Applikation Management befassen, ist es eben auch ein internes Thema. Wir wollen selber sehen, wie wir möglichst effizient Architekturen zum Betrieb, also auch zum Monitoring aufbauen können. Gleichzeitig ist es auch ein Kundenthema, weil wir öfters Anfragen kriegen zum Thema IIoT Umsetzung/ IIoT-Implementierung und deren Folgen, die Auswirkung der Daten.

**Lukas F.:** Insgesamt, wenn wir an Referenzarchitekturen denken, ich rede davon gleich mal im Plural, weil es aus meiner Sicht schon mal mindestens zwei geben muss, als Artefakt meiner Bachelorarbeit. Wie kompatibel sollen die denn zueinander sein, wenn wir uns vorstellen, es gibt zum Beispiel eine für die Echtzeit Verarbeitung und eine für die Batch-Verarbeitung von Daten? Müssen die austauschbar sein, also von derselben Quelle zum Beispiel „gefüttert“ werden? Oder ist es da okay jeweils recht spezifische Pipelines zu bauen, die dann programmatisch eingebunden werden müssen, relativ nah am Datenerzeugenden?

**Philipp A.:** Ich finde, das ist eine Frage, die die Arbeit beantworten sollte, denn ich kann das aus meiner aktuellen Position schwer abschätzen. Für mich ist es nicht klar, ob es sinnvoller ist individuell, also wenn man dem Erzeuger nahestehend die Daten dort auf ein Format zu bringen, die sich laden lassen, oder ob es sinnvoller ist, die Daten vom Erzeuger zu nehmen und dann in der Referenzarchitektur zu normalisieren und anzugleichen.

**Lukas F.:** Ich hatte jetzt nicht unbedingt an Daten direkt gedacht, sondern mehr an die Infrastruktur, die kompatibel sein muss. So dass es am Anfang eine Schnittstelle zum Beispiel geben würde, was eine Art Plug and Play mit beiden Ansätzen ermöglichen würde oder einfach das nur einen Ansatz quasi mit mit einer Customization sozusagen funktioniert?

**Philipp A.:** Ach so. Zu wünschen wäre es natürlich, wenn wir hinterher nur eine Hauptarchitektur hätten oder möglichst wenig Anpassungen an die Architekturen vornehmen müssen, um die Sachen zu switchen. Das ist natürlich aus der Architektenbrille die schönere Sache. Auf der anderen Seite glaube ich tatsächlich, wenn man sich einmal festgelegt hat auf eine von beiden Arten, dass man dann eh nicht mehr switchen wird.
Da muss man sich eben vorher klar werden, welchen Weg man gehen will. Es ist eigentlich völlig valide zu sagen, wenn ich weiß, dass ich Batch Verarbeitung mache, dann habe ich auch genau diese Architektur. Die Anforderung, dass man wechseln muss, ist zu vernachlässigen. Interessanter ist es eher, wenn man sagt ich brauche beides. Womöglich lohnt es sich da, beides zu inkludieren.

**Lukas F.:** Zu meiner nächsten Frage: Ich habe dir im Vorfeld zwei Listen zugesendet, wo ich gerne deine Meinung hören würde, wie du die jeweils priorisieren würdest. Zum einen sind das die Qualitätskriterien von Referenzarchitekturen und zum anderen die datenbezogenen Entscheidertypen. Fangen wir am besten mit den Qualitätskriterien an.

**Philipp A.:** Genau

**Lukas F.:** Da hat es ja sieben. Welche würdest du denn relativ weit oben aus deiner Position raus positionieren? Und welche sind von eher nachrangiger Relevanz?

**Philipp A.:** Ich picke von den sieben mal drei raus und vielleicht kannst du mir zu dem einen oder anderen Punkt noch ein bisschen was erläutern. Der Punkt fünf „akzeptabel“, da verstehe ich nicht so wirklich, was du damit meinst. Beim Thema „wertschöpfend für den Betrieb“ versteh ich jetzt auch nicht so, was du damit meinst im Vergleich zur Adressierung der Hauptprobleme. Kannst du das nochmal kurz erläutern?

**Lukas F.:** „Wertschöpfend für den Betrieb“: So wie ich den Autor verstehe, mein das, das man einen Wert daraus hat, die Referenzarchitekturen anzuwenden und es sich nicht lohnt von „scratch“ anzufangen.

**Philipp A.:** Okay. Wie sieht es mit akzeptabel aus?

**Lukas F.:** Beim Kriterium „akzeptabel“ verstehe ich den Autor so, dass man sich als jemand mit Fachkenntnissen im Prinzip das anschaut und sagt ja, das kann ich akzeptieren mit meinen Fachkenntnissen und das ist jetzt nicht völlig an den Haaren herbeigezogen. Es ist quasi „reasonable“.

**Philipp A.:** Das ist ja hoffentlich jeder Architektur. Wenn man das nicht mal voraussetzen kann, dann muss man eigentlich Punkt fünf als ersten nehmen es muss natürlich logisch, also „reasonable“ sein. Das nächste muss sein, dass es wertschöpfend ist in irgendeiner Weise, denn man macht nichts, was nicht einen Mehrwert darstellt. Wo ich Schwierigkeiten habe, ist mit „reasonable“ weil „Adressierung der Hauptprobleme“ hängt natürlich mit dem „reasonable“ zusammen.
Damit eine Referenzarchitektur wertschöpfend ist, muss sie aber auch verständlich sein. Jetzt ist die Frage, wie breit Verständlichkeit gehen muss. Hier steht eine breitere, heterogene Gruppe. So weit würde ich jetzt nicht gehen. Aber es muss verständlich sein für diejenigen, die die Referenzarchitektur anwenden müssen und da tatsächlich relativ einfach verständlich. Aber wenn das andere nicht gegeben ist, dass sie „reasonable“ ist, oder das die Probleme der Domäne nicht abgebildet werden, hilft sowieso alles nix.

**Lukas F.:** Das heißt du würdest tendenziell, dass es akzeptabel ist und die Probleme adressiert vorne anstellen, gefolgt von der Verständlichkeit?

**Philipp A.:** Nein, wertschöpfend ist tatsächlich noch Nummer zwei, vielleicht sogar Nummer eins. Wenn es nicht wertschöpfend ist, dann brauche ich das nicht zu machen, dann habe ich nur Papierkrieg. Es muss einen Mehrwert für den Betrieb geben, das ist eigentlich Nummer eins, unser zweites ist, dass es akzeptabel sein muss, in Kombination mit der Problemlösung der Domäne. Dann kommen wir zu Thema Verständlichkeit.

**Lukas F.:** Verstehe. Wenn wir übergehen zu den Datenentscheider- oder -nutzungstypen gibt es ja drei Stück. Die taktischen, die operativen und die strategischen Entscheider. Wo siehst du denn uns am ehesten? Letztendlich ist das ja die Basis dafür, die wir die Daten analysieren

wollen, also in welchen Entscheidungshorizont wir agieren und mit welcher Dringlichkeit wir neue Daten brauchen, um Entscheidungen abzuleiten.

**Philipp A.:** Wer ist denn wir? Das hängt natürlich davon ab, was das für Daten sind und was der Ziel der Daten sind, beziehungsweise was das Ziel ist. Ist das Ziel Monitoring, dann habe ich da natürlich erst mal eine sehr kurzfristige Datenlage, die ich bewerten muss. Also wenn beispielsweise der Speicherplatz vollläuft, ist das wichtiger. Wenn ich irgendwelche IIoT Daten habe, kann es durchaus sein, dass das eher langfristige Informationen sind. Wenn ich die Temperatur messe oder den $CO_2$ Gehalt messe, habe ich auch durchaus Interesse an der Langfristigkeit der Daten.

**Lukas F.:** Verstehe also tendenziell würdest du, wenn es um Monitoringdaten geht eher dem taktischen Entscheidertyp folgen, der recht früh betrachtet, wenn sich was ändert und seine Entscheidung im Zweifelsfall anpasst. IIoT siehst du also eher Richtung operative Entscheider?

**Philipp A.:** Auch da hängt es wieder von der Art der Daten ab. Wenn es ein IIoT Sensor ist, der messen soll, ob es ein Feuer gibt, dann ist es eine taktische Entscheidung, dann den Feueralarm zu betätigen. Wenn es aber Daten sind, die das Wetter beobachten, dann ist es vielleicht interessanter als strategischer Entscheider ranzugehen.
Es ist sehr datenbezogen. Beim Monitoring vielleicht ein bisschen weniger. Auch aus Monitoringdaten kann ich natürlich Sachen ziehen, wenn ich nach einem halben Jahr Monitoring sehe, in welchen Intervallen meine Systeme besonders ausgelastet sind. Davon können natürlich auch Entscheidungen abgeleitet werden. Die meisten Informationen beim Monitoring sind aber tatsächlich kurzfristige. Und bei IIoT kann ich das kann ich das gar nicht einschätzen, weil da bin ich nicht so tief drin und die Systeme von IIoT sind so mannigfaltig. Das ist ja keine Domäne an sich, sondern es ist ja eher eine Infrastruktur, die auf verschiedene Domänen anwendbar ist, je nachdem, was das für Sensordaten sind oder auch in welchen Intervallen die abgefragt werden. Wenn es Sensoren gibt, die jede halbe Stunde Daten melden, dann ist die Echtzeit Entscheidung eher sekundär. Wenn es aber Daten sind, die alle zehn Sekunden anfallen, ist das eher interessanter für Echtzeitentscheidungen.

**Lukas F.:** Also du meinst es kommt wesentlich auf die Messdistanz der Sensordaten an?

**Philipp A.:** auf jeden Fall

**Lukas F.:** Verstehe. Zum zweiten Teil des Interviews: Ich habe ein Dimensionsmodell konstruiert für Referenzarchitekturen, wo es jetzt um die subjektive Allgemeingültigkeit, die Anwendbarkeit, und die Dekompositionstiefe gehen soll. Jetzt wäre meine Frage jeweils, wie du drei Punkte einschätzt auf einer Skala von Null bis fünf und wieso. Wie wichtig ist dir subjektive Allgemeingültigkeit, wie tief muss eine Dekomposition stattfinden, damit Referenz Architekturen gut sind und wie konkret oder abstrakt darf die Referenzarchitektur sein, dass sie nutzenstiftend für viele Anwendungfälle ist, aber trotzdem eingesetzt werden kann.

**Philipp A.:** Die Allgemeingültigkeit und die Anwendbarkeit hängen stark vom Teilnehmerkreis ab, also wie groß sehen wir den Teilnehmerkreis der Leute, die mit dieser Referenzarchitektur

arbeiten sollen? Je größer der ist, desto größer muss natürlich auch die Allgemeingültigkeit sein. Wenn wir das in dem relativ engen Rahmen sehen, auf Abteilungsebene oder vielleicht bisschen größer, dann können diese beiden Punkte relativ eng gefasst sein.
Bei der Dekompositionstiefe , das ist aus meiner Sicht eine Fleißarbeit. Je detaillierter die Dekompositionstiefe ist, desto einfacher ist es vermutlich, die Referenz Architektur in eine echte umzusetzen Architektur. Man hat da ja dann schon viele Hilfestellungen, Beispiele, etc. . Erfahrene Architekten können dann auch die Dekompositionstiefe wählen, die sie brauchen. Die Dekompositionstiefe ist bei der Anwendbarkeit Teil des Kontext. Da ist es eben die Frage, wie groß die Bandbreite ist, wenn wir sagen, wir haben genau diese zwei Usecases, nämlich Monitoring und IIoT, dann kann die schon relativ konkret sein. Wenn wir sagen, wir haben vor, das irgendwie über die komplette Organisation und Firma zu stülpen, dann ist es halt sehr abstrakt. Ich halte nicht so viel davon, Dinge zu abstrakt zu machen, weil sie dann tatsächlich oft nicht verstanden werden und auch nicht benutzt werden. Je abstrakter ich Sachen mache, desto geringere Dekompositionstiefe habe ich normalerweise. Insofern würde ich die das Thema Allgemeingültigkeit eher so im mittleren und unteren Bereich sehen, genauso wie die Anwendbarkeit und Dekompositionstiefe so tief wie möglich.

**Lukas F.:** Wenn du das auf einer Skala, wo null das schlechteste/geringst abstrakteste etc. ist und fünf die Vollausprägung, also sehr spezifische Allgemeingültigkeit beispielsweise, wo würdest du das dann jeweils sehen?

**Philipp A.:** Dann würde ich sagen, die subjektive Allgemeingültigkeit sollte irgendwo bei drei bis vier liegen, genauso die Anwendbarkeit im Kontext und die Dekomposition sollte sehr, sehr tief sein bei fünf.

**Lukas F.:** Herzlichen Dank, ich denke das ist für das erste Interview schon recht viel, was ich für meine Arbeit mitnehmen kann. Gerne würde ich mit dir ein weiteres Interview zum Abschluss der Arbeit machen, bei der wir die jetzt erarbeiteten Kriterien auf meine Artefakte anwenden.

**Philipp A.:** Können wir so machen.

**Lukas F.:** Gut, dann vielen Dank für das Interview und deine Zeit.

**Philipp A.:** Gerne.

## Anhang 2: Experteninterview Peter E.

| | |
|---|---|
| Datum | 24.03.2021 |
| Thema | Initiales Anforderungsinterview |
| Teilnehmende, Position | Lukas Fruntke, Verfasser<br>Peter E., Head of Solution - IIoT |

**Lukas F.:** Hallo Peter, herzlichen Dank dass du dir die Zeit genommen hast mir als Interviewapartner zur Verfügung zu stehen! Kannst du bitte kurz beschreiben, welche Rolle du in der SPIRIT inne hast und wie deine Rolle mit Zeitreihendaten und Referenzarchitekturen zusammenhängt?

**Peter E.:** Meine Rolle in der SPIRIT ist laut offizieller Bezeichnung Head of Solution IIoT und Automation. Das heißt ich kümmere mich um die gesamten IIoT Projekte, die in der SPIRIT abgewickelt werden. Sei das Geschäftsentwicklung, Kundenbetreuung, Personalaufbau, Presales, ... oder anderes. Wir beschäftigen uns hauptsächlich mit der Digitalisierung von Städten, z.B. mit der Prozessoptimierung im Energie- und Versorgungsbereich. Dabei digitalisieren wir die ganzen Strom-, Gas-, Wasserzähler und lesen diese via Funk aus. Ich arbeite eigentlich ausschliesslich mit Zeitreihendaten, weil ein typisches Merkmal von IoT Projekten ist, dass diese gesammelt und auf verschiedenste Arten ausgewertet werden.

**Lukas F.:** Verstehe. Das Ziel meiner Bachelorarbeit ist ja, Referenzarchitekturen für die Verarbeitung von Zeitreihendaten in der Cloud, speziell in AWS zu konstruieren. Wo siehst du denn Einsatzbereiche für solche Referenzarchitekturen, also Kundencases o.ä.?

**Peter E.:** Wir haben eigentlich immer die gleiche Grobarchitektur, nach der die Datenverarbeitung läuft. Im Feld sind Sensoren, die über Gateway Services, welche die Daten in ein einheitliches Format harmonisieren, Daten erfassen. Die Daten kommen dann in ein Backend, wo sie verarbeitet werden und werden abschliessend gespeichert. Vom Speicherort aus werden die Daten momentan weiterverarbeitet. Das ist dabei sowohl Visualisierung als auch Übergabe an z.B. Drittsysteme.

**Lukas F.:** Wenn man das auf mein Feld, die Cloud überträgt, wäre die Analyse „hinten raus“ ja eine der Möglichkeiten für eine Referenzarchitektur.

**Peter E.:** Ja, aber du hast ja abgesehen davon auch in der Cloud Services, um dir deine Datenbeschaffung zu ermöglichen, wie z.B. den AWS IoT Core, also den MQTT Broker. An den kann man ja die Gatewayservices anknüpfen. Wenn ich mich richtig erinnere, bietet aber auch AWS verwaltete Gatewayservices zur Datennormalisierung an. Letztenendes könnte man diese Architektur also auch komplett in AWS abbilden.

**Lukas F.:** Ja das wäre auch ein gute Option, ich denke aber, dass es nur bedingt Sinn macht das in AWS nochmal komplett neuzubauen, da ihr ja schon einige laufenden Konverter habt,

die die Daten der Sensoren in maschinenlesbare Formate umwandeln. Ausgehend von diesen umgewandelten Daten wäre aus meiner Sicht eine Analyse viel spannender.

**Peter E.:** Verstehe, da könnte man ja die Daten aus der IoT Plattform ausleiten und Richtung AWS senden. Dies tun wir bereits so ähnlich in einem Kundenprojekt via MQTT, wo wir die Daten zur TU Dresden für Analysen ausleiten. Wir haben ja aber auch die $CO_2$ Sensoren in unseren Konferenzräumen, die permanent Daten sammeln, die könnte man ähnlich weiterleiten.

**Lukas F.:** Ja, die $CO_2$Daten hatte ich auch schon im Kopf.

**Peter E.:** Wenn du Auswertungen aber fahren möchtest, die statistisch belastbar sind, dann brauchst du ja Daten, die häufig und in großer Menge ankommen.

**Lukas F.:** Ja, dafür habe ich den Gerätesimulator entworfen, mit dem ich in einem kurzen Intervall Daten in den MQTT Broker bringen kann.

**Peter E.:** Wichtig ist, dass du die Usecases nach Datenfluss vergleichst. Also was und wie viel kommt von außen rein und welche Analysen möchtest du drüber fahren? Letztenendes ist das, was du machen möchtest das, was Ralph mit der IoT Plattform über Jahre gemacht hat. Das ist auch eine Referenzarchitektur. Da hat er sich überlegt: „Was könnte man nehmen um Zeitreihendaten zu speichern ?“ - da kam er auf Elasticsearch. „Wie bekomme ich Daten in die Plattform?“ - Da ist er nach langer Suche auf MQTT gekommen. „Wie kann man eine Verarbeitungslogik gestalten?“ - Da kam er auf Node-RED.

**Lukas F.:** Im Prinzip geht es genau um solche Referenzarchitekturen, die explorativ zu erarbeiten.

**Peter E.:** Das erste was du brauchst für die Referenzarchitektur ist eine Datenbank, die große Mengen an Zeitreihendaten verarbeiten kann.

**Lukas F.:** Ja, das ist eine Möglichkeit. Ich möchte mir aber auch Streamingdaten vom Broker direkt anschauen. Für Anwendungsfälle wie Schwellenwerte brauche ich die Daten nicht in der Datenbank, sondern kann schon vorher sagen, ob man einen Alarm auslösen muss.

**Peter E.:** Ja wobei du bei der Schwellwertanalyse natürlich Karenzzeiten beachten musst. Wenn beispielsweise in einem Kühlhaus ein Temperatursensor vorne an der Tür ist und du einen Alarm auf 0 °C hast. Dann reicht es, wenn die Tür zum Ent- oder Beladen aufgemacht wird, um den Alarm auszulösen. Ohne Karenzzeit hat man hier nen Alarmzustand, mit entsprechender Karenzzeit von beispielsweise 10 Minuten verhindert man solche Fehlauslösung.

**Lukas F.:** Ja, das müsste man aber auch in einer Echtzeitpipeline abbilden können. Generell geht es darum sich die verscheidenen Verarbeitungswege von Daten in AWS anzuschauen.

**Peter E.:** Okay, dann musst du aber bei der Datenquelle Unterscheidungen treffen. Es gibt zum einen diskrete Daten, wie beispielsweise Zählertelegramme von Wasserzählern. Dieser „beamt“ ein Telegramm raus, in fünf Minuten das nächste und so weiter. Das ist kein kontinuierlicher Strom, sondern der sendet immer ein Datenpaket in einem Zeitabstand $x$. Oder man bekommt Daten von

einem Sensor, wenn sich etwas ändert. Das sind keine Streamingdaten, sondern Zeitreihendaten. Ich weiß dabei nicht, ob man kontinuierliche Daten über MQTT übermitteln kann.

Es gibt drei verschiedene Datenkategorien. Das eine sind Events. Das heißt die Infrastruktur meldet einen Status oder ein Ereigniss, auf dass dann in einer Infrastruktur reagiert wird. Diese Events sind auch Zeitreihendaten. Wenn wir in die IT schauen, gibt es in einem Server Events, wo beispielsweise übermittelt wird „Meine Platte ist voll“. Dieses Event wird alle paar Minuten übermittelt, weil es beispielsweise ein Schwellwert ist. Diese Ereignisse enthalten keine Messwerte, sondern sind ein Ergebniss einer Messwertauswertung. Trotzdem laufen sie als Zeitreihendaten ein, weil sie hintereinander kommen, wie Messwerte auch. In einer Auswertung dieser Daten könnte man die dann deduplizieren oder eine Korrelation zu Metadaten machen. Wenn in einem Rechner z.B. die Gehäusetemperatur und die CPU und weiteres zu hoch ist, lässt das auf überhohe Last schliessen.

Der nächste Typ wäre Metriken, also Messwerte. Aus der Auswertung dieser Werte kann man dann selbst solche Events generieren. Das ist dann aber die Aufgabe der eigenen Auswertungslogik. Messwerte können diskret haben, wenn diese in einem zeitlichen Abstand kommen.

Es gibt aber auch Sensoren die „kontinuierliche“ Messwerte liefern. Das ist dann Streaming. Das betrifft nicht nur IoT sondern Automatisierungstechnik generell. Bei diesen drei Arten unterscheidet sich jeweils die Verarbeitung. Gestreamte Messwerte wären beispielsweise, wenn man aneiner stromporduzierenden Windturbine hängt. Dabei misst man in einem relativ engen Zeitraster, wie der Verlauf der Leistungskurve aussieht. Oder man misst, wie die Stromkurve oder die Phasenverschiebung aussieht. Die Daten sind sehr eng zeitlich gerastert, damit man auch kleinste Änderungen mitbekommt.

**Lukas F.:** Du machst also den Unterschied zwischen diskreten Messwerten und „Streamingdaten“ abhängig von der Messdistanz, weenn ich das richtig verstehe?

**Peter E.:** Richtig. Im Prinzip kann man sagen, wenn die Daten einen gewissen zeitlichen Abstand unterschreiten, kann man von einem Stream sprechen.

**Lukas F.:** Generell soll das Ziel der Referenzarchitektur sein, verschiedene Messdistanzen zu erlauben, seien das jetzt Milisekunden oder Minuten.

**Peter E.:** Genau, und wenn du einen Milisekundenabstand hast, dann bist du sozusagen im Streaming mode. Da muss dann deine Infrastruktur dahinter permanent auf Höhe sein, damit die ja nix verpasst.

**Lukas F.:** Ja, das ist einer der Punkte wo aus meiner Sicht die Cloud interessant wird.

**Peter E.:** ja, aber die Cloud wird auch schon bei Daten interessant, wenn ein hohes Volumen von vielen Devices eintrifft. Wenn man beispielsweise eine Stadt nimmt, die Smart Meter breit einsetzt, dann haben die bei einer größeren Stadt über 70.000 Devices verteilt. Die senden dann vielleicht nur alle 15 Minuten ein Messwert. Wenn aber jetzt ein Stromzähler pro Telegram

50 Bytes sendet, alle 15 Minuten und das multipliziert mit 70.000 ist die Cloud schon sehr interessant.

**Lukas F.:** Ja, das sehe ich auch so. Es kommt ja zum einen auf die Menge $n$ der Sensoren an und zum anderen eben auf die Messdistanz.

**Peter E.:** Genau. Um mal bei dem Windradbeispiel zu bleiben: Ein Windrad kann eine Dateninfrastruktur schon ziemlich unter Stress setzen, wenn man zeitlich enge Raster braucht, weil man kleinste Abweichungen mitbekommen will. Kennst du da aus der Messtechnik das Shannon Theorem? *(Nyquist-Shannon-Abtasttheorem)*

**Lukas F.:** Nein.

**Peter E.:** Da geht es um Analog-Digital Wandlung. Auf einer Stromleitung hat man ja einen Sinus von 50 Hz. Das ist ja ein kontinuierliches Signal. Wenn man diese Kurve jetzt samplen, also digitalisieren will, möchte man ja eine Art Messpunkt an der Kurve anlegen, um die Sinuskurve in ein zeitlich enges Raster aus digitalen, diskreten Messwerten zu legen. Das Theorem sagt, dass man mindestens mit der doppelten Messfrequenz so ein Signal abtasten muss, um das richtig herauszubekommen. Also muss ein Analog Digital Wandler mindestens mit 100Hz abtasten. Wenn der Analog-Digital Wandler jetzt eine Auflösung von 16 bit hat, wird die Amplitude der Sinuskurve in $2^{16}$ „Stifte“ zerteilt. Das heißt pro Messample hat man 2 bytes, dann hat man 100 Samples pro Sekunde mal 2, also 200 bytes pro Sekunde, nur bei 50Hz. Bei einem schnelleren Signal muss man das entsprechend schneller abtasten. Normalerweise werden Signale mit der vierfachen Frequenz gesampled, also 200Hz und 200 Datenpunkte pro Sekunde. Das ist dann Streaming. Das kommt immer zum Einsatz, wenn man analoge Signale digitalisiert, wie hier die Spannung, die ins Netz eingespeist wird.

**Lukas F.:** Auch solche „Streaming“ Usecases sollte meine Architektur unterstützen. Idealerweise skalieren die eingesetzten Dienste ja auch.

**Peter E.:** Von diesen Lastanforderungen kann man ableiten, welche Dienste man im Hitnergrund von AWS braucht, um das zu verarbeiten. Diese Methodik habe ich beim HP in der IT-Datacenterautomatisierung/Überwachung/Monitoring schon gehabt. Das gleiche trifft aber auch so auf IoT zu. Das ist auch einer der Gründe, warum man die gleiche Software für IT-Automatisierung und für den Maschinenbau einsetzen kann. Das ist die gleiche Software. Wenn du so eine Referenzarchitektur also erstellt hast, kann man die nicht nur für IoT benutzen. Das ist wichtig. Man kann die auch für IT benutzen.

**Lukas F.:** Ja, die Idee die Referenzarchitektur auch für Monitoring zu verwenden, die besteht. Ganz viele Daten sind ja Zeitreihendaten.

**Peter E.:** Der Fakt, dass man solche Systeme sowohl für IoT als auch IT verwenden kann, ist für mich ein unique selling point. Das Beispiel dafür ist mein Windparkmanager, den ich beim HP gemacht habe. Da habe ich das andersrum gedreht. Da habe ich die IT-Monitoringsoftware genommen und für IoT Geräte verwendet. Die Kunden waren begeistert!

**Lukas F.:** Verstehe, diese Dual use Möglichkeit werde ich auf jeden Fall nochmal erwähnen in der Bachelorarbeit. Ich hätte auch noch ein paar Fragen vorbereitet. In der Literatur gibt es die Aussage, dass es drei verschiedene Entscheidertypen gibt (Erklärungen siehe Abbildung 2). Es gibt taktische, operative und strategische Entscheider. Welche Entscheider findest denn du am wichtigsten?

**Peter E.:** Die wichtigsten sind aus meiner Sicht die taktischen Entscheider, welche wohl den fachlichen Entscheidern in den Fachabteilungen entsprechen. Diese Leute sitzen im operativen Geschäft und müssen schnell entscheiden, bevor im Zweifelsfall etwas „abfackelt". Beispiel Windturbine - es gibt Fälle, wo die Windturbine in einen kritischen Zustand geht - da muss sofort gehandelt werden, weil sonst Leben bedroht sind. Es gibt aber auch Entscheider mit mitelfristigen Entscheidungen, die operativen Entscheider. Diese Entscheidenden wollen beispielsweise Trends sehen, um Entscheidungen zu treffen. Eine Ebene höher, bei den strategischen Entscheidern, quasi auf dem „C-Level" werden wesentlich größere Datenmengen für Lagebilder in anderen Perspektiven und „Blickhöhen benötigt". Leute auf den beiden oberen Ebenen nehmen die Daten als gegeben hin. Die haben keine Ahnung, welche technische Komplexität dahinter steht, Daten zu erfassen und aufzubereiten. Und damit diese Entscheider sich nicht darum kümmern müssen und davon nichts mitbekommen, da kommt deine Referenzarchitektur ins Spiel.

**Lukas F.:** Verstehe, ja das sehe ich genau so. Ich hätte noch ein paar kleiner Fragen dabei, wo ich deine Priorisierung bräuchte. Ich habe ja die Qualitätskriterien von Muller. Welche davon priorisierst du denn am höchsten oder was siehst du denn am wichtigsten?

**Peter E.:** Das Kriterium eins ist sehr wichtig. Verständlich für eine breite, und was wichtig ist eine heterogene Gruppe an Stakeholdern. Die Referenzarchitektur aus verschiedenen Perspektiven zu beleuchten ist wichtig. Was bei dir unter fünf steht, „akzeptabel", das würde ich unter Akzeptanz sehen und als zweites einstufen. Damit hängen andere Kriterien zusammen. Es ist akzeptabel, wenn die Qualität stimmt. Zugänglichkeit und Zugriff durch die Mehrheit der Organisation führt zu einer hohen Akzeptanz. Adressierung der Hauptprobleme ist auch wichtig, weil sich die meisten Problemkategorien im IoT Bereich auf wenige Kernprobleme herunterbrechen lässt.
Ich würde mich nochmal korrigieren, das wäre meine Reihenfolge:

1. Verständlichkeit
2. Adressierung der Hauptprobleme (daraus bedingt sich werstschöpfend für den Betrieb)
3. Akzeptanz durch die Anwender (bedingt durch):
    a) zufriedenstellende Qualität des Dienstes/Produktes (hängt von up-to date und wartbar ab)
    b) Qualität/„das es gescheit funktioniert"
    c) *einfache* Zugänglichkeit durch Mehrheit der Organisation

**Lukas F.:** Herzlichen Dank für die Priorisierung! Ich hätte auch noch das Dimensionsmodell. Wie würdest du die jeweils bewerten?

**Peter E.:** Wir gehen das Pferd immer von der folgenden Seite an - Wir versuchen diese Usecases auf ein allgemeingültiges Level in IIoT anzuheben. Unsere IIoT Plattform ist, wie ich vorher erklärt habe genauso abstrahiert.

**Lukas F.:** Die Anwendbarkeit wäre bei dir also die fünf?

**Peter E.:** Genau. Dein Modell hat ja Zielkonflikte. Der Idealfall wäre, wenn alles fünf wäre.

**Lukas F.:** In der perfekten Welt wären alle also eine fünf. Wenn wir jetzt ein bestcase Szenario annehmen, dann wäre die Anwendbarkeit ja fünf bei dir. Wie wären die anderen Werte?

**Peter E.:** Die Dekompositionstiefe sollte möglichst gering sein. Als Anwender möchte man gegebenenfalls gar nicht alle Low-Level Details sehen.

**Lukas F.:** Verstehe. Wenn du das in Zahlen fassen würdest wäre das eine?

**Peter E.:** Die Allgemeingültigkeit wäre bei einer fünf, die Anwendbarkeit zwischen vier und fünf. Die Dekompositionstiefe wäre für mich zwischen zwei und drei. Wenn du die Dekomposition zu detailliert machst, wird es womöglich zu komplex und ist nicht mehr anwendbar.

**Lukas F.:** Verstehe, alles klar. Das waren einige coole Insights, herzlichen Dank dafür und für deine Zeit!

**Peter E.:** Keine Ursache

## Anhang 3: Experteninterview Ralph B.

| | |
|---|---|
| Datum | 24.03.2021 |
| Thema | Initiales Anforderungsinterview |
| Teilnehmende, Position | Lukas Fruntke, Verfasser<br>Ralph B., Head of Solution - R&D |

**Lukas F.:** Herzlichen Dank Ralph, für deine Bereitschaft zum Interview. Beginnend möchte ich dich fragen, was deine Rolle/Tätigkeiten innerhalb der SPIRIT/21 sind und wie du mit Architekturen zu tun hast.

**Ralph B.:** Meine Rolle in der SPIRIT hat sich schon ien paar mal gewandelt. Aktuell bin ich für den Bereich EBSS Lead R&D Verantwortlicher für Forschung und Entwicklung. Als Vorsitzender des Tech- und Architekturboards bin ich für die technologische Qualifizierung von Entwicklungsthemen verantwortlich. Genauso koordiniere ich auch den Einsatz von bestimmten neuen Technologien in den einzelnen Solutions.

**Lukas F.:** Verstehe. Wo siehst du denn Anwendungsgebiete von Referenzarchitekturen in Richtung Zeitreihendatenverarbeitung?

**Ralph B.:** Welche Architekturen siehst du denn da im Scope? Softwarearchitekturen, oder Infrastrukturachitekturen oder eine andere Architektur?

**Lukas F.:** Ich denke an technische Architekturen, die einen Teil Software und Infrastrukturkomposition umfassen, best practices und eine Art Referenzvorgehen sind „wie löse ich dieses wiederkehrende Problem, dass immer wieder auftaucht“? Speziell in Richtung Cloud und AWS gesehen.

**Ralph B.:** Gut, AWS Cloud sehe ich jetzt firmenweit betrachtet nur als ein Teilthema von vielen. Generell zählen für mich da organisatorische und fachliche Richtlinien/Konzepte mit herein. Das geht über die wiederverwendbare technische Lösung hinaus. Vielleicht sollten auch Problem addressiert werden, die momentan nicht akut sind, aber in Zukunft wichtig werden könnten. Generell kann man nicht von einer schlechten Architektur oder einer schlechten Referenzarchitektur sprechen. Klassifizierung in gut oder schlecht ist schwierig. Stattdessen muss man schauen, ob die Architektur auf den Usecase passt oder nicht.

**Lukas F.:** Konkret Richtung Zeitreihendaten gedacht - wo siehst du konkret die Anwendungsgebiete, wo eine Referenzarchitektur unterstützen könnte?

**Ralph B.:** Das ist generisch immer ein wenig schwierig. Es kommt auf den Anwendungsfall an. Bei Datenerfassung bei IoT-Daten muss das Kriterium angelegt werden, ob sehr viele Daten in kurzer Zeit erfasst werden können. Ist die Lösung skalierbar? Wichtig ist aber auch das Ausgeben der Daten: müssen diese instant bereit stehen, oder habe ich da einen Zeitpuffer von fünf Sekunden, bis diese wieder bereit stehen müssen? Wenn ich Daten gespeichert habe, wie lange

brauchts die zu lesen? Wo sind Bottlenecks etc.? Im IoT Bereich ist speziell der Durchsatz, also die Messages pro Sekunde, die kommen könnten ein Problem, weil jeder Sensor einen Wert sendet, der dann gespeichert werden will. Das hat dann auch mit Verfügbarkeit zu tun - wie bekomme ich die Datenbank dahinter 100% verfügbar? Die konkreten Anforderungen sind dabei immer unterschiedlich. Wenn man jetzt z.B. LoRaWAN Sensordaten hat von 20.000 Sensoren, die alle x Sekunden Daten senden. Kann meine Datenbank diese speichern? Und wenn ich dann einen Report möchte einmal pro Woche, dann möchte ich nicht lange warten, sondern schnell in den Daten navigieren können.

**Lukas F.:** Wir haben ja jetzt schon ein wenig Richtung IIoT Daten geschaut. Speziell die Problematik mit den Auswertungen ist da wichtig. Entsprechend dem Datenhalbwertszeitmodell hat es ja drei verschiedene Entscheidertypen, die Daten in unterschiedlicher Zeit benötigen und Entscheidungen treffen. Taktische Entscheider brauchen Daten sehr schnell und trifft auch schnell Entscheidungen. Operative Entscheider brauchen Daten nicht unbedingt nahe Echtzeit und haben einen erweiterten Entscheidungshorizont auf Tage oder Wochen. Strategische Entscheider haben einen wesentlich größeren Entscheidungshorizont gleichzeitig aber auch geringere Anforderungen an die „Echtzeitigkeit“ der Daten. Welche siehst du denn als am wichtigsten an oder für welche würdest du im IoT Bereich optimieren?

**Ralph B.:** Ich denke die sind alle gleich wichtig. Im operativen Entscheidungsmodus sollten die Entscheidungsregeln idealerweise automatisiert sein. Einen ausgelösten Feuermelder mit einer Email an einen Verantwortlichen zu koppeln, macht weniger Sinn, als beispielsweise die Werksfeuerwehr zu rufen. Bei unseren bisherigen Projekten ist Latenz nicht so wichtig - da die Geräte bei Bedarf Nachrichten senden oder in Intervallen von 50/60 Sekunden. Latetnz wird aber wieder wichtig, wenn man beispielsweise einen Wassersensor hat, der eine Aktion in der Businesslogik auslöst, die das Wasser bei Kontakt abstellt. Von den historischen Daten kann man ein Dashboard befüllen, wo ein Entscheider einmal die Woche drüberschaut und Entscheidungen trifft. Dann gibt es noch langfristige Usecases - Billing Daten beispielsweise interessieren den Kunden nur einmal im Monat.

**Lukas F.:** Okay, verstehe. Ein bisschen von der Technik weg - es gibt ja Qualitätskriterien zu Referenzarchitekturen - die hatte ich dir schon gesendet. Wie siehst du denn die?

**Ralph B.:** Ich bin da noch eher klassisch. Wenn du jetzt die ISO 9126 anschaust für Qualität von Softwaresystemen (*siehe Abbildung 44*), hast du noch ein paar andere Kriterien. Was ist denn zufriedenstellende Qualität? Wenn der Kunde zufrieden ist, ist es dann zufriedenstellend? Qualitätsmerkmale sind immer schwammig, weil es nicht „die Qualität“ gibt. Es gibt nur eine passende Architektur, die eine passt besser, die andere schlechter. Qualität ist nicht absolut, sondern relativ anhand von Kriterien.

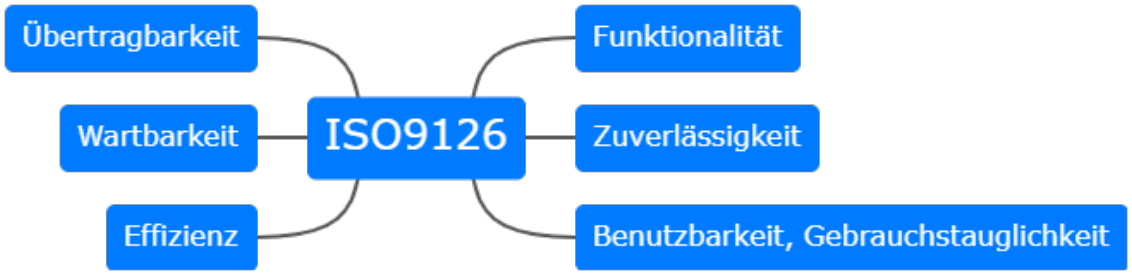


Abb. 44: Qualitätskriterien nach ISO 9126.[235]

**Lukas F.:** Ja, für mich gehts darum zu erfahren, welche Kriterien da für den maximierten Nutzen für unsere Zielstakeholder am wichtigsten sind.

**Ralph B.:** Übertragbarkeit zwischen Usecases ist wichtig. Gleichzeitig muss man aber schauen, dass man sich nicht zu sehr fokussiert auf ein Qualitätskriterium. Wenn man beispielsweise bewusst auf single-instance Datenbanken verzichtet und diese immer im Cluster errichtet, erhöht das die Zuverlässigkeit, sprengt aber vielleicht den Preisrahmen des Usecases.

**Lukas F.:** Da möchte ich mit den Variationspunkten ansetzen, wo ich Entscheidungsoptionen aufzeige wie man die Referenzarchitektur instanziieren kann.

**Ralph B.:** Das Problem ist ja, dass man meistens mit mehr Qualitätskriterien, die man berücksichtigt das Resultat teurer wird.

**Lukas F.:** Da macht das KISS Prinzip dann eher Sinn, oder?

**Ralph B.:** Genau. Der Kunde sollte halt immer den Mehrwert sehen hinter den extra Features die man implementiert. Achte auch auf die Zuverlässigkeit und Modifizierbarkeit.

(0:30) **To do** fertig machen (17)

[235] Mit Änderungen entnommen aus: Johner 2018

# Anhang 4: Experteninterview Philipp A.

| | |
|---|---|
| Datum | xx.04.2021 |
| Thema | Review Referenzarchitekturen |
| Teilnehmende,<br>Position | Lukas Fruntke, Verfasser<br>Philipp A., Cloud Solution Architekt - NCS |

**To do** transkribieren (18)

## Anhang 5: Berechnungsskript Dateigröße

Angefügt ist das verwendete Berechnungsskript für die Dateigröße, geschrieben in JavaScript, ausführbar mit der Node.JS Umgebung.

```
const fs = require('fs');
const path = require('path')
let sensorcount = 200;
let alertsPerMonth = 5;
let months = 3;

let result = [];

for (let i = 1; i <= sensorcount; i++) {
    for (let month = 0; month <= months; month++) {
        for (let alert = 1; alert <= alertsPerMonth; alert++) {
            let alertDate = new Date(2021, month, alert, 2, 0, 0)
            result.push({
                deviceId: `Sensor-${i}`,
                timestamp: alertDate.toISOString(),
                value: Math.floor(Math.random() * 1000) + 100
            })
        }
    }

}

let filtered = result.filter(element =>
    (element.deviceId == "Sensor-1")
    && element.timestamp == "2021-01-01T01:00:00.000Z")

const estimatePath = path.join(__dirname, '/estimate.json')
const fEstimatePath = path.join(__dirname, '/filtered-estimate.json')

fs.writeFileSync(estimatePath, JSON.stringify(result, null, 4))
fs.writeFileSync(fEstimatePath, JSON.stringify(filtered, null, 4))

let stats = fs.statSync(estimatePath)
console.log("Size in KB", stats.size / (1024))
```

Codeausschnitt 3: Berechnungsskript Dateigröße

## Anhang 6: Umfrage Kriterienpriorisierung

Mithilfe des Tools Questionpro wurde eine Umfrage erstellt, um Architekten der SPIRIT/21 GmbH im Bereich Native Cloud die Möglichkeit zu geben, die 11 Kriterien für die Produkt/-Dienslteistungsbewertung via „Drag & Drop“ selber zu priorisieren.

Die Umfrage präsentierte sich folgendermaßen:

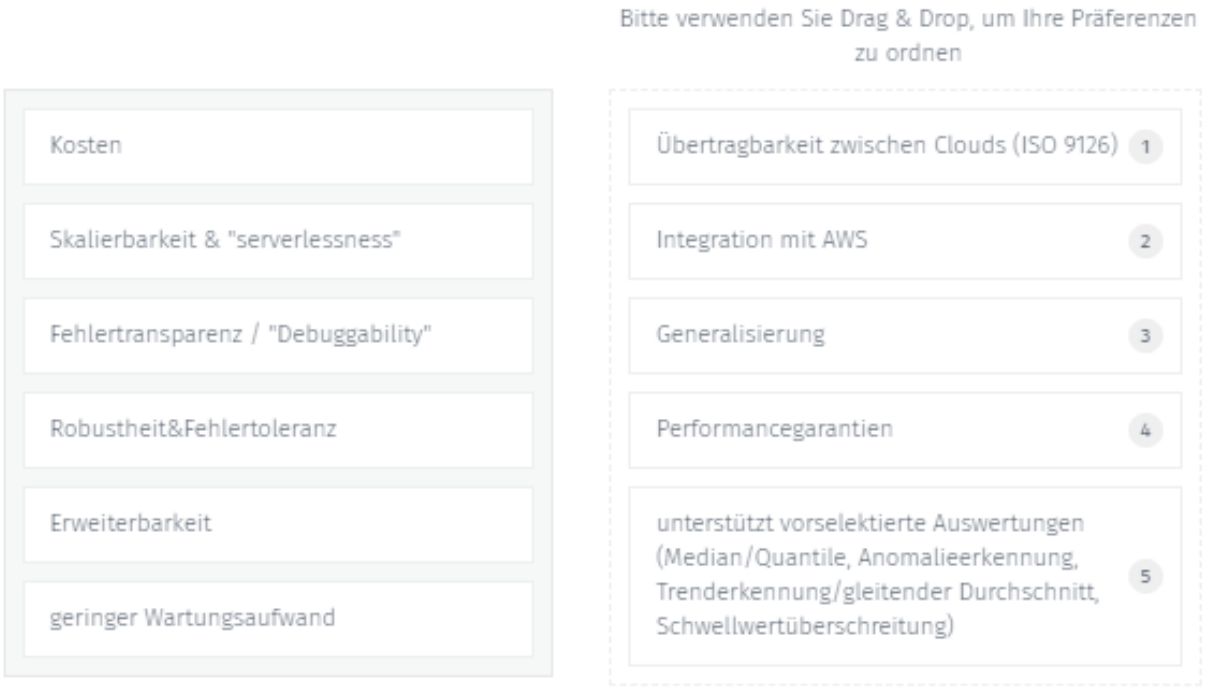


Abb. 45: Die Umfrage in QuestionPro

| Kriterium | Durchschnitt | gerundet |
|---|---|---|
| Übertragbarkeit zwischen Clouds (ISO 9126) | 1,00 | 1 |
| Integration mit AWS | 3,17 | 3 |
| Generalisierung | 4,33 | 4 |
| Erweiterbarkeit | 4,33 | 4 |
| Fehlertransparenz / „Debuggability“ | 4,67 | 5 |
| geringer Wartungsaufwand | 6,5 | 7 |
| Skalierbarkeit & „serverlessness“ | 7,17 | 7 |
| Kosten | 7,33 | 7 |
| Performancegarantien | 8,0 | 8 |
| Robustheit & Fehlertoleranz | 8,83 | 9 |
| unterstützt Auswertungen (Unterabschnitt 2.1.2) | 10,67 | 11 |
| Summe | | 66 |

Tab. 23: Auswertungen der Umfragen

Insgesamt nahmen $n = 6$ Personen teil. Die Ergebnisse sind in Tabelle 23 dargestellt.

## Anhang 7: OpenSearch Abfragebeispiel

Folgend gezeigt wird, wie mittels der OpenSearch/Elasticsearch eigenen, JSON-basierten Abfragesprache Abfragen durchgeführt werden.

```
{
  "aggs": {
    "load_time_outlier": {
      "percentiles": {
        "field": "load_time",
        "percents": [ 25, 75, 95, 99, 99.9 ]
      }
    }
  }
}
```

Codeausschnitt 4: Perzentile in OpenSearch berechnen.[236]

[236] Mit Änderungen entnommen aus: Elasticsearch, Inc. o.J.(c)

## Anhang 8: Echtzeit Referenzarchitektur Lambda Codebeispiel

Folgend wird als Beispiel eine in JavaScript geschriebene Lambda Funktion gezeigt. Diese sendet, basierend auf den eingehenden Daten, Alarme via SNS und versendet via MQTT einen „shutdown“ Befehl.

```
const AWS = require('aws-sdk');
const MQTT = require("async-mqtt");
const SNS = new AWS.SNS();
const TopicArn = process.env.TOPIC_ARN;
const MqttEndpoint = process.env.MQTT_ENDPOINT
let client;

exports.handler = async (event) => {
    // Aufbau einer MQTT Verbindung zu IoT Core
    if (!client) {
        client = await MQTT.connect(MqttEndpoint);
    }

    let output = [];
    for (let record of event.records) {
        try {
            // Daten aus base64 dekodieren
            const data = Buffer.from(record.data, 'base64');

            // versendet Alarm via SNS
            await SNS.publish({ Subject: `Schwellwertüberschreitung!`,
                Message: `Rohdaten:\n ${JSON.stringify(data, null, 4)}`,
                TopicArn, MessageDeduplicationId: record.recordId
            });

            // versendet einen Shutdown Befehl via MQTT
            await client.publish("geraet/"+record.deviceId,
                { action: "shutdown" })
            // Nachricht erfolgreich verarbeitet
            output.push({recordId: record.recordId, result: 'Ok'});
        } catch (err) {
            // Fehler aufgetreten - erneuten Versuch via Kinesis Data Analytics
            console.error("Fehler bei", record.recordId, "wegen", err);
            output.push({recordId: record.recordId, result: 'DeliveryFailed'});
        }
    }
    return { records: output }
}
```

Codeausschnitt 5: Beispielcode zum Versenden von Alarmen und automatischer Aktion

## Anhang 9: Batch Referenzarchitektur Lambda Codebeispiel

Folgend wird eine von Anhang 8 abgewandelte Lambdafunktion gezeigt, die mit Timestream interagiert.

```
import { parseResult } from '@nordicsemiconductor/timestream-helpers'
const AWS = require('aws-sdk');
const TimestreamQuery = new AWS.TimestreamQuery();
const SNS = new AWS.SNS();
const TopicArn = process.env.TOPIC_ARN;
const Database = process.env.DATABASE;
const Table = process.env.TABLE;
let results;

async function getData(nextToken = null) {
    let queryData = await TimestreamQuery.query({
        // Code ist *nicht* SQL-Injection sicher!
        QueryString: "SELECT device_id, count(*) AS count" +
            ` FROM "${Database}"."${Table}"`+
            " WHERE measure_name = 'co2' AND time > ago(10m)" +
            " AND measure_value::bigint > 60" +
            " GROUP BY device_id",
        NextToken: nextToken
    }).promise()

    results = results.concat(parseResult(queryData))
    if (queryData.nextToken) {
        await getData(queryData.nextToken);
    }
}

exports.handler = async (event) => {
    try {
        let time = event.time ?  new Date(event.time) : new Date();

        await getData();
        for (let {device_id, count} of results) {
            // versendet Alarm via SNS
            await SNS.publish({Subject: `Schwellwertüberschreitung in
            ↪  ${device_id}!`,
                Message: `${count} mal in den letzten 15 Minuten überschritten`,
```

```
                TopicArn, MessageDeduplicationId: time+device_id
            }).promise();

            // versendet einen Shutdown Befehl via MQTT
            await client.publish("geraet/" + device_id,{ action: "shutdown" })
        }


    } catch (err) {
        console.error(err)
        // Fehler erneut werfen, damit CloudWatch Alarm geschalten wird
        throw err;
    }

}
```

Codeausschnitt 6: Beispielcode zur Interaktion mit Timestream

## To do...

- ☐ 1 (p. II): Improve this draft
- ☐ 2 (p. 1): final überarbeiten
- ☐ 3 (p. 2): final überarbeiten
- ☐ 4 (p. 4): belegen
- ☐ 5 (p. 12): Kapitel ist ziemlich lang (P)
- ☐ 6 (p. 14): In Text einbetten, Erklärung welche passen
- ☐ 7 (p. 20): deutlich ausbauen
- ☐ 8 (p. 21): ausbauen
- ☐ 9 (p. 24): Notwendig??
- ☐ 10 (p. 27): Hier noch besserer Übergang
- ☐ 11 (p. 38): letzten Halbsatz belegen
- ☐ 12 (p. 62): Genauere Auswirkungen
- ☐ 13 (p. 79): Vergleich RA - Deadline 03.05.
- ☐ 14 (p. 79): Interview
- ☐ 15 (p. 79): ausfüllen
- ☐ 16 (p. 81): Schluss schreiben
- ☐ 17 (p. 96): fertig machen
- ☐ 18 (p. 97): transkribieren

# Literaturverzeichnis


**Amazon Web Services, Inc.**, Hrsg. **(o.J.[a])**: Alerting for Amazon Elasticsearch Service. URL: https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/alerting.html (Abruf: 08. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[b])**: Amazon Athena adds support for running SQL queries across relational, non-relational, object, and custom data sources. URL: https://aws.amazon.com/about-aws/whats-new/2020/11/aws-what-s-new-for-athena-federated-query (Abruf: 07. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[c])**: Amazon Athena Pricing – Serverless Interactive Query Service – Amazon Web Services. URL: https://aws.amazon.com/athena/pricing (Abruf: 07. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[d])**: Amazon Elasticsearch Service Pricing - Amazon Web Services. URL: https://aws.amazon.com/elasticsearch-service/pricing (Abruf: 08. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[e])**: Amazon EventBridge monitoring. URL: https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-monitoring.html (Abruf: 01. 05. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[f])**: Amazon Kinesis Data Analytics pricing. Amazon Web Services, Inc. URL: https://aws.amazon.com/kinesis/data-analytics/pricing (Abruf: 05. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[g])**: Amazon Kinesis Data Firehose Pricing. URL: https://aws.amazon.com/kinesis/data-firehose/pricing (Abruf: 01. 05. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[h])**: Amazon Kinesis Data Firehose Pricing. URL: https://aws.amazon.com/kinesis/data-firehose/pricing (Abruf: 09. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[i])**: Amazon Kinesis Data Streams. URL: https://aws.amazon.com/kinesis/data-streams (Abruf: 10. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[j])**: Amazon Kinesis Data Streams pricing. URL: https://aws.amazon.com/kinesis/data-streams/pricing (Abruf: 05. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[k])**: Amazon Kinesis Service Level Agreement. URL: https://aws.amazon.com/kinesis/sla (Abruf: 10. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[l])**: Amazon Managed Service for Grafana Pricing. URL: https://aws.amazon.com/grafana/pricing/?nc=sn&loc=3 (Abruf: 02. 05. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[m])**: Amazon MSK Pricing. URL: https://aws.amazon.com/msk/pricing (Abruf: 06. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[n])**: Amazon QuickSight Pricing. URL: https://aws.amazon.com/quicksight/pricing (Abruf: 02. 05. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[o])**: Amazon Redshift Pricing. URL: https://aws.amazon.com/redshift/pricing (Abruf: 09. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[p])**: Amazon S3 pricing. URL: https://aws.amazon.com/s3/pricing (Abruf: 31. 03. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[q])**: Amazon SageMaker - Amazon Timestream. Amazon Web Services, Inc. URL: https://docs.aws.amazon.com/timestream/latest/developerguide/Sagemaker.html (Abruf: 11. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[r])**: Amazon Timestream. URL: https://aws.amazon.com/timestream (Abruf: 11. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[s])**: Amazon Timestream Pricing. URL: https://aws.amazon.com/timestream/pricing (Abruf: 02. 05. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[t])**: Anomaly Detection for Amazon Elasticsearch Service. URL: https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/ad.html (Abruf: 13. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[u])**: AWS Glue Pricing - Managed ETL Service - Amazon Web Services. Amazon Web Services, Inc. URL: https://aws.amazon.com/glue/pricing (Abruf: 07. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[v])**: AWS IoT Analytics FAQ Page. URL: https://aws.amazon.com/iot-analytics/faq (Abruf: 19. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[w])**: AWS IoT Analytics Pricing. URL: https://aws.amazon.com/iot-analytics/pricing (Abruf: 31. 03. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[x])**: AWS IoT Analytics quotas. URL: https://docs.aws.amazon.com/iotanalytics/latest/userguide/limits.html (Abruf: 18. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[y])**: AWS IoT Events Pricing. URL: https://aws.amazon.com/iot-events/pricing (Abruf: 03. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[z])**: AWS IoT Events quotas. URL: https://docs.aws.amazon.com/iotevents/latest/developerguide/iotevents-quotas.html (Abruf: 15. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[aa])**: AWS IoT Events Service Level Agreement. URL: https://aws.amazon.com/iot-events/sla (Abruf: 15. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[ab])**: AWS IoT Greengrass Features. URL: https://aws.amazon.com/greengrass/features (Abruf: 02. 05. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[ac])**: AWS IoT metrics and dimensions. URL: https://docs.aws.amazon.com/iot/latest/developerguide/metrics_dimensions.html (Abruf: 26. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[ad])**: AWS Lambda - Amazon Timestream. URL: https://docs.aws.amazon.com/timestream/latest/developerguide/Lambda.html (Abruf: 11. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[ae])**: AWS Step Functions Pricing. URL: https://aws.amazon.com/step-functions/pricing (Abruf: 07. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[af])**: Data Modeling - Timestream. URL: https://docs.aws.amazon.com/timestream/latest/developerguide/data-modeling.html (Abruf: 02. 05. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[ag])**: Derivatives functions - Amazon Timestream. URL: https://docs.aws.amazon.com/timestream/latest/developerguide/timeseries-specific-constructs.functions.derivatives.html (Abruf: 11. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[ah])**: EXP_AVG - Amazon Kinesis Data Analytics. URL: https://docs.aws.amazon.com/kinesisanalytics/latest/sqlref/sql-reference-exp-avg.html (Abruf: 15. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[ai])**: Expressions - AWS IoT Events. URL: https://docs.aws.amazon.com/iotevents/latest/developerguide/iotevents-expressions.html (Abruf: 15. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[aj])**: Grafana - Amazon Timestream. URL: https://docs.aws.amazon.com/timestream/latest/developerguide/Grafana.html (Abruf: 02. 05. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[ak])**: Group Rank - Amazon Kinesis Data Analytics. URL: https://docs.aws.amazon.com/kinesisanalytics/latest/sqlref/sql-reference-group-rank-udx.html (Abruf: 15. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[al])**: Important metrics for CloudWatch. Amazon Web Services, Inc. URL: https://docs.aws.amazon.com/lambda/latest/operatorguide/important-metrics.html (Abruf: 28. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[am])**: Kinesis Data Analytics Metrics and Dimensions. URL: https://docs.aws.amazon.com/kinesisanalytics/latest/dev/monitoring-metrics.html (Abruf: 26. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[an])**: Kinesis Data Streams. URL: https://docs.aws.amazon.com/iot/latest/developerguide/kinesis-rule-action.html (Abruf: 26. 02. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[ao])**: Monitoring Amazon SNS topics using CloudWatch. URL: https://docs.aws.amazon.com/sns/latest/dg/sns-monitoring-using-cloudwatch.html (Abruf: 26. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[ap])**: Monitoring Kinesis Data Firehose Using CloudWatch Metrics. URL: https://docs.aws.amazon.com/firehose/latest/dev/monitoring-with-cloudwatch-metrics.html (Abruf: 01. 05. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[aq])**: MQTT. URL: https://docs.aws.amazon.com/iot/latest/developerguide/mqtt.html#mqtt-qos (Abruf: 27. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[ar])**: Profiling functions with AWS Lambda Power Tuning. URL: https://docs.aws.amazon.com/lambda/latest/operatorguide/profile-functions.html (Abruf: 17. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[as])**: Query - Amazon Timestream. URL: https://docs.aws.amazon.com/timestream/latest/developerguide/queries.html (Abruf: 02. 05. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[at])**: RANDOM CUT FOREST WITH EXPLANATION. Amazon Kinesis Data Analytics. URL: https://docs.aws.amazon.com/kinesis

analytics/latest/sqlref/sqlrf-random-cut-forest-with-explanation.html (Abruf: 15. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[au])**: Real-time Processing of Log Data with Subscriptions. URL: https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/Subscriptions.html (Abruf: 01. 05. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[av])**: Service Quotas - Amazon Athena. URL: https://docs.aws.amazon.com/athena/latest/ug/service-limits.html (Abruf: 10. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[aw])**: SQL expressions in AWS IoT Analytics. URL: https://docs.aws.amazon.com/iotanalytics/latest/userguide/sql-support.html (Abruf: 17. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[ax])**: Storage - Amazon Timestream. URL: https://docs.aws.amazon.com/timestream/latest/developerguide/storage.html (Abruf: 02. 05. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[ay])**: Supported actions - AWS IoT Events. URL: https://docs.aws.amazon.com/iotevents/latest/developerguide/iotevents-supported-actions.html (Abruf: 15. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[az])**: Supported data types. URL: https://docs.aws.amazon.com/timestream/latest/developerguide/supported-data-types.html (Abruf: 02. 05. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[ba])**: Supported data types - Amazon Timestream. URL: https://docs.aws.amazon.com/timestream/latest/developerguide/supported-data-types.html (Abruf: 07. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[bb])**: Timestream Metrics and Dimensions. URL: https://docs.aws.amazon.com/timestream/latest/developerguide/metrics-dimensions.html (Abruf: 28. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[bc])**: Tuning query performance - Amazon Redshift. URL: https://docs.aws.amazon.com/redshift/latest/dg/c-optimizing-query-performance.html (Abruf: 10. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[bd])**: Use immutable infrastructure with no human access. Financial Services Industry Lens. URL: https://docs.aws.amazon.com/wellarchitected/latest/financial-services-industry-lens/use-immutable-infrastructure-with-no-human-access.html (Abruf: 07. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[be])**: Using a Lambda Function as Output. URL: https://docs.aws.amazon.com/kinesisanalytics/latest/dev/how-it-works-output-lambda.html#how-it-works-output-lambda-perms (Abruf: 26. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[bf])**: Using Amazon Kinesis Data Analytics - Amazon Kinesis Data Firehose. URL: https://docs.aws.amazon.com/firehose/latest/dev/data-analysis.html (Abruf: 30. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[bg])**: What Is Amazon Kinesis Data Streams? Benefits of Using Kinesis Data Streams. URL: https://docs.aws.amazon.com/streams/latest/dev/introduction.html#using-the-service (Abruf: 10. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[bh])**: What Is Amazon Timestream? URL: https://docs.aws.amazon.com/timestream/latest/developerguide/what-is-timestream.html (Abruf: 23. 03. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[bi])**: What is AWS IoT Analytics. URL: https://docs.aws.amazon.com/iotanalytics/latest/userguide/welcome.html (Abruf: 19. 03. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[bj])**: What is AWS IoT Events? URL: https://docs.aws.amazon.com/iotevents/latest/developerguide/what-is-iotevents.html (Abruf: 19. 03. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[bk])**: Writes. URL: https://docs.aws.amazon.com/timestream/latest/developerguide/metering-and-pricing.writes.html (Abruf: 02. 05. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(o.J.[bl])**: Writing to your Kinesis Data Stream Using the KPL - Amazon Kinesis Data Streams. URL: https://docs.aws.amazon.com/streams/latest/dev/kinesis-kpl-writing.html (Abruf: 29. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2017)**: Amazon Kinesis Data Analytics can now Output Real-Time SQL Results to AWS Lambda. URL: https://aws.amazon.com/about-aws/whats-new/2017/12/amazon-kinesis-data-analytics-can-now-output-real-time-sql-results-to-aws-lambda (Abruf: 05. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2019a)**: Amazon Athena Service Level Agreement. URL: https://aws.amazon.com/athena/sla (Abruf: 10. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2019b)**: Amazon Elasticsearch Service - Service Level Agreement. URL: https://aws.amazon.com/elasticsearch-service/sla (Abruf: 10. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2019c)**: Amazon Managed Streaming for Apache Kafka (MSK) - Service Level Agreement. URL: https://aws.amazon.com/msk/sla (Abruf: 17. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2019d)**: Amazon Redshift Service Level Agreement. URL: https://aws.amazon.com/redshift/sla (Abruf: 10. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2019e)**: AWS IoT Analytics Service Level Agreement. URL: https://aws.amazon.com/iot-analytics/sla (Abruf: 18. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2019f)**: AWS Lambda Service Level Agreement. URL: https://aws.amazon.com/lambda/sla (Abruf: 17. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2020a)**: Amazon Athena announces availability of engine version 2. URL: https://aws.amazon.com/about-aws/whats-new/2020/11/amazon-athena-announces-availability-of-engine-version-2 (Abruf: 10. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2020b)**: Amazon Kinesis Data Analytics now supports Apache Flink v1.11. URL: https://aws.amazon.com/about-aws/whats-new/2020/11/amazon-kinesis-data-analytics-now-supports-apache-flink (Abruf: 29. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2020c)**: Amazon Redshift. Database Developer Guide. URL: https://docs.aws.amazon.com/redshift/latest/dg/redshift-dg.pdf (Abruf: 22. 03. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2020d)**: Amazon Timestream is now Generally Available. URL: https://aws.amazon.com/about-aws/whats-new/2020/09/amazon-timestream-now-generally-available (Abruf: 02. 05. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2020e)**: Amazon Timestream SLA. URL: https://aws.amazon.com/timestream/sla (Abruf: 11. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2020f)**: AWS IoT Analytics - Benutzerhandbuch. URL: https://docs.aws.amazon.com/de_de/iotanalytics/latest/userguide/analytics-ug.pdf (Abruf: 18. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2020g)**: AWS Lambda now supports batch windows of up to 5 minutes for functions with Amazon SQS as an event source. URL: https://aws.amazon.com/about-aws/whats-new/2020/11/aws-lambda-now-supports-batch-windows-of-up-to-5-minutes-for-functions (Abruf: 05. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2020h)**: Summary of the Amazon Kinesis Event in the Northern Virginia (US-EAST-1) Region. URL: https://aws.amazon.com/message/11201 (Abruf: 19. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2021a)**: Amazon Athena now presents query execution plans to aid tuning. URL: https://aws.amazon.com/about-aws/whats-new/2021/04/amazon-athena-now-presents-query-execution-plans-to-aid-tuning (Abruf: 10. 04. 2021).

**Amazon Web Services, Inc.**, Hrsg. **(2021b)**: Amazon SNS unterstützt jetzt 1-Minuten-CloudWatch-Metriken. URL: https://aws.amazon.com/de/about-aws/whats-new/2021/01/amazon-sns-now-supports-1-minute-cloudwatch-metrics (Abruf: 02. 05. 2021).

**Angelov, S./Grefen, P./Greefhorst, D. (2012)**: A framework for analysis and design of software reference architectures. In: *Information and Software Technology* 54.4, S. 417–431. URL: https://www.sciencedirect.com/science/article/pii/S0950584911002333.

**Angiulli, F./Ben-Eliyahu - Zohary, R./Palopoli, L. (2008)**: Outlier detection using default reasoning. In: *Artificial Intelligence* 172.16-17, S. 1837–1872.

**Augsten, S. (2020)**: Was ist Chaos Engineering? URL: https://www.dev-insider.de/was-ist-chaos-engineering-a-971111 (Abruf: 10. 04. 2021).

**Bajer, M. (2017)**: Building an IoT Data Hub with Elasticsearch, Logstash and Kibana. In: *2017 5th International Conference on Future Internet of Things and Cloud workshops. W-FiCloud 2017 : Prague, Czech Republic, 21-23 August 2017 : proceedings*. 2017 IEEE 5th International Conference on Future Internet of Things and Cloud: Workshops (W-FiCloud) (Prague). Hrsg. von I. Awan/F. Portela/M. Younas. BigR & I u. a. Piscataway, NJ: IEEE, S. 63–68.

**Barr, J. (2016)**: Amazon Athena – Interactive SQL Queries for Data in Amazon S3. Hrsg. von Amazon Web Services, Inc. URL: https://aws.amazon.com/blogs/aws/amazon-athena-interactive-sql-queries-for-data-in-amazon-s3 (Abruf: 23. 03. 2021).

**Barr, J. (2021a)**: AWS Fault Injection Simulator – Use Controlled Experiments to Boost Resilience. Hrsg. von Amazon Web Services, Inc. URL: https://aws.amazon.com/blogs/aws/aws-fault-injection-simulator-use-controlled-experiments-to-boost-resilience (Abruf: 02. 05. 2021).

**Barr, J. (2021b)**: CloudWatch Metric Streams – Send AWS Metrics to Partners and to Your Apps in Real Time. Hrsg. von Amazon Web Services, Inc. URL: https://aws.amazon.com/blogs/aws/cloudwatch-metric-streams-send-aws-metrics-to-partners-and-to-your-apps-in-real-time (Abruf: 30. 04. 2021).

**Bartos, M./Mullapudi, A./Troutman, S. (2019)**: rrcf: Implementation of the Robust Random Cut Forest algorithm for anomaly detection on streams. In: *Journal of Open Source Software* 4.35, S. 1336.

**Bass, L./Clements, P./Kazman, R. (2010)**: Software architecture in practice. 2. ed., 14. print. SEI series in software engineering. Boston, Mass.: Addison-Wesley.

**Belur, V. (2020)**: Kappa Architecture. Easy Adoption with Informatica End-to-End Streaming Data Management Solution. Hrsg. von Informatica Corp. URL: https://blogs.informatica.com/2020/05/13/adopt-a-kappa-architecture-for-streaming-and-ingesting-data (Abruf: 15. 03. 2021).

**Berle, L. (2017)**: Streamingarchitekturen in der Praxis: Lambda vs. Kappa. URL: https://jaxenter.de/streaming-lambda-kappa-64573 (Abruf: 11. 03. 2021).

**Beswick, J. (2020a)**: Creating faster AWS Lambda functions with AVX2. Hrsg. von Amazon Web Services, Inc. URL: https://aws.amazon.com/blogs/compute/creating-faster-aws-lambda-functions-with-avx2 (Abruf: 09. 03. 2021).

**Beswick, J. (2020b)**: Using Amazon MSK as an event source for AWS Lambda. Hrsg. von Amazon Web Services, Inc. URL: https://aws.amazon.com/blogs/compute/using-amazon-msk-as-an-event-source-for-aws-lambda (Abruf: 06. 04. 2021).

**Bonomi, F./Milito, R./Zhu, J./Addepalli, S. (2012)**: Fog computing and its role in the internet of things. In: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC '12*. the first edition of the MCC workshop (Helsinki, Finland). Hrsg. von M. Gerla/D. Huang. New York, New York, USA: ACM Press, S. 13.

**Booz, R. (2020)**: TimescaleDB vs. Amazon Timestream: 6000x faster inserts, 5-175x query speed. Hrsg. von Timescale, Inc. URL: https://blog.timescale.com/blog/timescaledb-vs-amazon-timestream-6000x-higher-inserts-175x-faster-queries-220x-cheaper (Abruf: 11. 04. 2021).

**Business Application Research Center**, Hrsg. **(o.J.)**: The Most Common Problems Companies Are Facing With Their Big Data Analytics. URL: https://bi-survey.com/challenges-big-data-analytics (Abruf: 28. 03. 2021).

**Cabé, B. (2018)**: Key Trends from the IoT Developer Survey 2018. URL: https://blog.benjamin-cabe.com/2018/04/17/key-trends-iot-developer-survey-2018 (Abruf: 26. 02. 2021).

**Chen, D./Chen, Y./Brownlow, B. N./Kanjamala, P. P./Arredondo, C. A. G./Radspinner, B. L./Raveling, M. A. (2017)**: Real-Time or Near Real-Time Persisting Daily Healthcare Data Into HDFS and ElasticSearch Index Inside a Big Data Platform. In: *IEEE Transactions on Industrial Informatics* 13.2, S. 595–606.

**Codd, E. F./Codd, S. B./Salley, C. T. (1993)**: Providing OLAP to User-Analysts: An IT Mandate. Hrsg. von E.F. Codd Associates. URL: http://www.estgv.ipv.pt/PaginasPessoais/jloureiro/ESI_AID2007_2008/fichas/codd.pdf (Abruf: 22. 04. 2021).

**Confluent, Inc.**, Hrsg. **(o.J.)**: ksqlDB Tutorial: How to build a User-Defined Function (UDF) to transform events using ksqlDB. URL: https://kafka-tutorials.confluent.io/udf/ksql.html (Abruf: 17. 04. 2021).

**Crate.io, Inc.**, Hrsg. **(2020)**: Amazon Timestream is finally released: these are our first impressions. URL: https://crate.io/a/amazon-timestream-first-impressions (Abruf: 11. 04. 2021).

**Cui, Y. (2017)**: Auto-scaling Kinesis streams with AWS Lambda. URL: https://theburningmonk.com/2017/04/auto-scaling-kinesis-streams-with-aws-lambda (Abruf: 26. 04. 2021).

**Das, S./Rath, T. (2020)**: Deriving real-time insights over petabytes of time series data with Amazon Timestream. Hrsg. von Amazon Web Services, Inc. URL: https://aws.amazon.com/blogs/database/deriving-real-time-insights-over-petabytes-of-time-series-data-with-amazon-timestream (Abruf: 11. 04. 2021).

**Deistler, M./Scherrer, W. (2018)**: Zeitreihen und stationäre Prozesse. In: *Modelle der Zeitreihenanalyse*. Hrsg. von M. Deistler/W. Scherrer. Cham: Springer International Publishing, S. 1–29.

**Dutt, R. (2020)**: Our new partnership with AWS gives Grafana users more options. Hrsg. von Grafana Labs, Inc. URL: https://grafana.com/blog/2020/12/15/announcing-amazon-managed-service-for-grafana (Abruf: 26. 04. 2021).

**Elasticsearch, Inc.**, Hrsg. **(o.J.[a])**: Median absolute deviation aggregation. URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-metrics-median-absolute-deviation-aggregation.html (Abruf: 13. 04. 2021).

**Elasticsearch, Inc.**, Hrsg. **(o.J.[b])**: minimum_should_match parameter. URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-minimum-should-match.html (Abruf: 13. 04. 2021).

**Elasticsearch, Inc.**, Hrsg. **(o.J.[c])**: Percentiles aggregation. URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-metrics-percentile-aggregation.html (Abruf: 13. 04. 2021).

**Elasticsearch, Inc. (o.J.[d])**: Was ist Elasticsearch? Hrsg. von Elasticsearch, Inc. URL: https://www.elastic.co/what-is/elasticsearch (Abruf: 23. 03. 2021).

**Elektroniksystem i Umeå AB**, Hrsg. **(2019)**: ERS CO2 Datasheet. URL: https://elsys.se/public/datasheets/ERS_CO2_datasheet.pdf (Abruf: 15. 03. 2021).

**Erber, M. (2021)**: IoT Data Streaming - Warum MQTT und Kafka eine exzellente Kombination sind. URL: https://www.informatik-aktuell.de/betrieb/netzwerke/iot-data-streaming-warum-mqtt-und-kafka-eine-exzellente-kombination-sind.html (Abruf: 05. 03. 2021).

**Gallagher, B. (2000)**: Using the Architecture Tradeoff Analysis Method to Evaluate a Reference Architecture: A Case Study. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. URL: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5109.

**Guha, S./Mishra, N./Roy, G./Schrijvers, O. (2016)**: Robust Random Cut Forest Based Anomaly Detection on Streams. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. Hrsg. von M. F. Balcan/W. Q. Kilian. ICML'16. JMLR.org, S. 2712–2721.

**Gupta, A./Agarwal, D./Tan, D./Kulesza, J./Pathak, R./Stefani, S./Srinivasan, V. (2015)**: Amazon Redshift and the Case for Simpler Data Warehouses. In: *Compilation proceedings of the 2015 ACM Symposium on Principles of Database Systems, ACM SIGMOD International Conference on Management of Data, and SIGMOD/PODS 2015 PhD symposium, May 31 - June 4, 2015, Melbourne, VIC, Australia*. SIGMOD/PODS'15: International Conference on Management of Data (Melbourne Victoria Australia). Hrsg. von T. Sellis/S. B. Davidson/Z. Ives. Association for Computing Machinery u. a. New York, NY: ACM, S. 1917–1923.

**Handler, J. (2019)**: Set alerts in Amazon Elasticsearch Service | Amazon Web Services. Hrsg. von Amazon Web Services, Inc. URL: https://aws.amazon.com/blogs/big-data/setting-alerts-in-amazon-elasticsearch-service (Abruf: 13. 04. 2021).

**Hartland, T. G./Frost, W./Love, P. (2018)**: Using AWS Athena analytics to monitor pilot job health on WLCG compute sites. CERN. URL: https://cds.cern.ch/record/2649941 (Abruf: 10. 04. 2021).

**Hartmann, A./Kriegel, M. (2020)**: Risikobewertung von virenbeladenen Aerosolen anhand der CO2-Konzentration. Berlin: Technische Universität Berlin, Hermann-Rietschel-Institut. URL: https://depositonce.tu-berlin.de/handle/11303/11477.3.

**Hayes, B. (2020)**: Usage of Programming Languages by Data Scientists: Python Grows while R Weakens |. Hrsg. von Business Broadway. URL: http://businessoverbroadway.com/2020/06/29/usage-of-programming-languages-by-data-scientists-python-grows-while-r-weakens (Abruf: 17. 04. 2021).

**Herman, J. (2020)**: AWS Kinesis Data Analytics: a cautionary review. Hrsg. von Bigdata Republic B.V. URL: https://medium.com/bigdatarepublic/kinesis-data-analytics-sql-a-cautionary-review-fb9ddd06e5d9 (Abruf: 15. 04. 2021).

**IEEE (2000)**: IEEE recommended practice for architectural description of software-intensive systems. New York, N.Y: Institute of Electrical and Electronics Engineers. URL: https://ieeexplore.ieee.org/document/875998.

**Johner, C. (2018)**: Funktionale Anforderungen versus nicht-funktionale Anforderungen. URL: https://www.johner-institut.de/blog/iec-62304-medizinische-software/funktionale-und-nicht-funktionale-anforderungen (Abruf: 24. 03. 2021).

**Khadtare, S. (2018)**: Performance Comparison of AWS Athena and Google BigQuery. Hrsg. von Mindtree Ltd. URL: https://www.mindtree.com/blog/performance-comparison-aws-athena-and-google-bigquery (Abruf: 10. 04. 2021).

**Kolence, K. W. (1973)**: The software empiricist. In: *ACM SIGMETRICS Performance Evaluation Review* 2.2, S. 31–36.

**Kreps, J. (2014)**: Questioning the Lambda Architecture. URL: https://www.oreilly.com/radar/questioning-the-lambda-architecture (Abruf: 11. 03. 2021).

**Kreps, J. (2019)**: Introducing ksqlDB. Hrsg. von Confluent, Inc. URL: https://www.confluent.io/blog/intro-to-ksqldb-sql-database-streaming (Abruf: 17. 04. 2021).

**Levy, E. (2019)**: Benchmarking AWS Athena vs BigQuery: Performance, Price, Data Freshness. Hrsg. von Upsolver Ltd. URL: https://www.upsolver.com/blog/benchmarking-aws-athena-bigquery-performance-price (Abruf: 10. 04. 2021).

**Levy, E. (2021)**: How to Improve AWS Athena Performance: The Complete Guide. Hrsg. von Upsolver Ltd. URL: https://www.upsolver.com/blog/aws-athena-performance-best-practices-performance-tuning-tips (Abruf: 10. 04. 2021).

**Madden, C./Bawcom, A. (2019)**: Analyzing Performance and Cost of Large-Scale Data Processing with AWS Lambda. Hrsg. von Amazon Web Services, Inc. URL: https://aws.amazon.com/blogs/apn/analyzing-performance-and-cost-of-large-scale-data-processing-with-aws-lambda (Abruf: 17. 04. 2021).

**Mantfeld, M. (2019)**: Elasticsearch and IoT. A case for Elastic stack as an IoT analytics platform. URL: https://aginic.com/blog/elasticsearch-and-iot (Abruf: 23. 03. 2021).

**Marz, N./Warren, J. (2015)**: Big data. Principles and best practices of scalable real-time data systems. Shelter Island, NY: Manning. URL: http://proquest.tech.safaribooksonline.de/9781617290343.

**Meadows, C./Graybill, J./Davis, K./Shah, M. (2021)**: Introducing OpenSearch. Hrsg. von Amazon Web Services, Inc. URL: https://aws.amazon.com/blogs/opensource/introducing-opensearch (Abruf: 13. 04. 2021).

**Megler, V. (2016)**: Anomaly Detection Using PySpark, Hive, and Hue on Amazon EMR. Hrsg. von Amazon Web Services, Inc. URL: https://aws.amazon.com/blogs/big-data/anomaly-detection-using-pyspark-hive-and-hue-on-amazon-emr (Abruf: 13. 04. 2021).

**Mell, P./Grance, T. (2011)**: The NIST definition of cloud computing: Reccomendations of the National Institute of Standards and Technology. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf (Abruf: 18. 06. 2020).

**Moonesinghe, H. D. K./Tan, P.-N. (2006)**: Outlier Detection Using Random Walks. In: *Proceedings / 18th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2006. 13 - 15 November 2006, Arlington, Virginia*. 2006 18th IEEE International Conference on Tools with Artificial Intelligence (Arlington, VA). IEEE Computer Society u. a. Los Alamitos, Calif.: IEEE Computer Society, S. 532–539.

**Muller, G. (2020)**: A Reference Architecture Primer. URL: https://www.gaudisite.nl/ReferenceArchitecturePrimerPaper.pdf (Abruf: 16. 03. 2021).

**Narkhede, N. (2017)**: Introducing KSQL: Streaming SQL for Apache Kafka. URL: https://www.confluent.io/blog/ksql-streaming-sql-for-apache-kafka (Abruf: 17. 04. 2021).

**Nobile, G./Natali, D. (2018)**: Scale Amazon Kinesis Data Streams with AWS Application Auto Scaling. Hrsg. von Amazon Web Services, Inc. URL: https://aws.amazon.com/blogs/big-data/scaling-amazon-kinesis-data-streams-with-aws-application-auto-scaling (Abruf: 27. 04. 2021).

**Nucleus Research, Inc.**, Hrsg. **(2012)**: Measuring the half life of data. Guidebook. URL: https://nucleusresearch.com/wp-content/uploads/2018/05/m36-Guidebook-Measuring-the-half-life-of-data.pdf (Abruf: 12. 03. 2021).

**o.V. (o.J.[a])**: pandas.DataFrame.gt. URL: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.gt.html (Abruf: 17. 04. 2021).

**o.V. (o.J.[b])**: pandas.DataFrame.median. URL: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.median.html?highlight=median#pandas.DataFrame.median (Abruf: 17. 04. 2021).

**o.V. (o.J.[c])**: pandas.DataFrame.quantile. URL: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.quantile.html (Abruf: 17. 04. 2021).

**o.V. (2020)**: MQTT: The Standard for IoT Messaging. URL: https://mqtt.org (Abruf: 26. 02. 2021).

**OASIS Open Consortium**, Hrsg. **(2014)**: MQTT Version 3.1.1. OASIS Standard. URL: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc385349263 (Abruf: 27. 04. 2021).

**Peak, J. (2017)**: Median absolute deviation for time series outlier detection in Amazon Redshift. Hrsg. von StackExchange, Inc. URL: https://dba.stackexchange.com/q/142661 (Abruf: 13. 04. 2021).

**Peng, Z./Jimenez, J. L. (2020)**: Exhaled CO 2 as COVID-19 infection risk proxy for different indoor environments and activities. URL: https://www.medrxiv.org/content/10.1101/2020.09.09.20191676v2.full.pdf+html (Abruf: 15. 03. 2021).

**Penz, M. (2020)**: Building a Kafka playground on AWS — Part 3: Streaming database events and querying with KSQL. URL: https://maikelpenz.medium.com/building-a-kafka-playground-on-aws-part-3-streaming-database-events-and-querying-with-ksql-5f23978a0080 (Abruf: 17. 04. 2021).

**Pochiraju, T. (2020)**: From Metal To Alerts With AWS IoT, Timestream and QuickSight. URL: https://dev.to/tejpochiraju/from-metal-to-alerts-with-aws-iot-timestream-and-quicksight-2b5b (Abruf: 02. 05. 2021).

**Pogosova, A. (2020)**: Mastering AWS Kinesis Data Streams, Part 1. URL: https://dev.solita.fi/2020/05/28/kinesis-streams-part-1.html (Abruf: 09. 03. 2021).

**Prasath, H. O. (2019)**: Autoscaling with Kinesis stream. URL: https://medium.com/javarevisited/autoscaling-with-kinesis-stream-cogs-reduction-dfd87848ce9a (Abruf: 26. 04. 2021).

**Ravirala, R./Al-Saadoon, L./Printz, W./Ratan, U./Mukerje, N. (2020)**: Analytics Lens. AWS Well-Architected Framework. Hrsg. von Amazon Web Services, Inc. URL: https://docs.aws.amazon.com/wellarchitected/latest/analytics-lens/wellarchitected-analytics-lens.pdf (Abruf: 29. 03. 2021).

**Riddle, J./Patana-anake, T. (2021)**: Export cloudwatch metrics to timestream #16. URL: https://github.com/awslabs/amazon-timestream-tools/issues/16 (Abruf: 02. 05. 2021).

**Robinson, D. (2017)**: Why is Python Growing So Quickly? - Stack Overflow Blog. Hrsg. von Stack Overflow. URL: https://stackoverflow.blog/2017/09/14/python-growing-quickly (Abruf: 17. 04. 2021).

**Ross, E. (2020)**: Moving Averages in SQL. URL: https://skeptric.com/moving-averages-sql (Abruf: 13. 04. 2021).

**RyanN (2018)**: AWS Developer Forums: kinesis sql - median. Hrsg. von Amazon Web Services, Inc. URL: https://forums.aws.amazon.com/thread.jspa?threadID=264401 (Abruf: 15. 04. 2021).

**Salgado, R. (2019)**: Anomaly Detection With SQL. Towards Data Science. URL: https://towardsdatascience.com/anomaly-detection-with-sql-7700c7516d1d (Abruf: 13. 04. 2021).

**Schütte, R. (1998)**: Grundsätze ordnungsmäßiger Referenzmodellierung. Dissertation. Münster: Universität Münster.

**Shankar, K./Wang, P./Xu, R./Mahgoub, A./Chaterji, S. (2020)**: JANUS: Benchmarking Commercial and Open-Source Cloud and Edge Platforms for Object and Anomaly Detection Workloads. In: *2020 IEEE 13th International Conference on Cloud Computing. CLOUD 2020 : proceedings : 18-24 October 2020, virtual event*. 2020 IEEE 13th International Conference on Cloud Computing (CLOUD) (Beijing, China). Hrsg. von L. Khan/G. Huang. Piscataway, NJ: IEEE, S. 590–599.

**Sharma, A. (2019)**: Moving Averages in pandas. URL: https://www.datacamp.com/community/tutorials/moving-averages-in-pandas (Abruf: 17. 04. 2021).

**Shumway, R. H./Stoffer, D. S. (2017a)**: Characteristics of Time Series. In: *Time Series Analysis and Its Applications: With R Examples*. Hrsg. von R. H. Shumway/D. S. Stoffer. Cham: Springer International Publishing, S. 1–44.

**Shumway, R. H./Stoffer, D. S.**, Hrsg. **(2017b)**: *Time Series Analysis and Its Applications: With R Examples*. Cham: Springer International Publishing.

**Singh, M. P./Hoque, M. A./Tarkoma, S. (2016)**: A survey of systems for massive stream analytics. URL: https://arxiv.org/pdf/1605.09021 (Abruf: 07. 04. 2021).

**Skerrett, I. (2019)**: Why MQTT Has Become the De-Facto IoT Standard. URL: https://dzone.com/articles/why-mqtt-has-become-the-de-facto-iot-standard (Abruf: 26. 02. 2021).

**Smallcombe, M. (2020)**: Amazon Redshift Spectrum vs. Athena: A Detailed Comparison. Hrsg. von Xplenty Inc. URL: https://www.xplenty.com/blog/amazon-redshift-spectrum-vs-athena (Abruf: 09. 04. 2021).

**Stanley, B. (2019)**: Amazon Kinesis Data Streams: Auto-scaling the number of shards. Hrsg. von L. L. Slalom. URL: https://medium.com/slalom-data-analytics/amazon-kinesis-data-streams-auto-scaling-the-number-of-shards-105dc967bed5 (Abruf: 26. 04. 2021).

**Starke, G. (o.J.)**: Building Block View. URL: https://docs.arc42.org/section-5 (Abruf: 24. 02. 2021).

**Statz, D. (2019)**: An honest AWS MSK review. URL: https://dima-statz.medium.com/an-honest-aws-msk-review-july-19-75f23a00c1cc (Abruf: 17. 04. 2021).

**Tan, J./Ghanem, T./Perron, M./Yu, X./Stonebraker, M./DeWitt, D./Serafini, M./Aboulnaga, A./Kraska, T. (2019)**: Choosing a cloud DBMS. In: *Proceedings of the VLDB Endowment* 12.12, S. 2170–2182.

**The Presto Foundation**, Hrsg. **(o.J.)**: Aggregate Functions — Presto 0.217 Documentation. URL: `https://prestodb.io/docs/0.217/functions/aggregate.html` (Abruf: 13. 04. 2021).

**Trefke, J. (2012)**: Grundlagen der Referenzarchitekturentwicklung. In: *IT Architekturentwicklung im Smart Grid: Perspektiven für eine sichere markt- und standardbasierte Integration erneuerbarer Energien*. Hrsg. von H.-J. Appelrath/P. Beenken/L. Bischofs/M. Uslar. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 9–30.

**Ubiq**, Hrsg. **(o.J.)**: How to Calculate Moving Average in Redshift. URL: `https://ubiq.co/database-blog/how-to-calculate-rolling-average-in-redshift` (Abruf: 13. 04. 2021).

**Vaquero, L. M./Rodero-Merino, L. (2014)**: Finding your Way in the Fog. In: *ACM SIGCOMM Computer Communication Review* 44.5, S. 27–32.

**vom Brocke, J. (2015)**: Referenzmodellierung. Gestaltung und Verteilung von Konstruktionsprozessen. Zugl.: Münster, Univ., Diss., 2002. 2. Aufl. Bd. 4. Advances in information systems and management science. Berlin: Logos.

**vom Brocke, J./Buddendick, C. (2004)**: Organisationsformen in der Referenzmodellierung — Forschungsbedarf und Gestaltungsempfehlungen auf Basis der Transaktionskostentheorie. In: *Wirtschaftsinformatik* 46.5, S. 341–352.

**Waehner, K. (2018)**: Deep Learning KSQL UDF for Streaming Anomaly Detection of MQTT IoT Sensor Data - Kai Waehner. URL: `https://www.kai-waehner.de/blog/2018/08/02/deep-learning-kafka-ksql-udf-anomaly-detection-mqtt-iot-sensor` (Abruf: 17. 04. 2021).

**Webber, D. (2001)**: The Variation Point Model: A Graphical Representation of Variation Points for Use on a Family of Systems. In: *Fourth DoD Product Line Practice Workshop Report*. Product Line Systems Program. Pittsburgh, PA, S. 24–27.

# Erklärung

Ich versichere hiermit, dass ich meine Bachelorarbeit mit dem Thema: *Konzeption von Referenzarchitekturen für IoT-Zeitreihenverarbeitung in der Cloud* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Rutesheim, 07.05.2021

(Ort, Datum) (Unterschrift)