COLOGNE INTELLIGENCE

White Paper

Einrichtung eines Data Lakes auf AWS

verfasst von:
Giovanni Maccaferri
Version: 1.0

Köln, den 10. Oktober 2019

Richtig entscheiden. Ihre Unternehmensdaten sind Goldwert.

Für uns sind Daten nicht nur Nullen und Einsen. Unsere Business Intelligence, Big Data und Data Science Experten entwickeln die passenden Lösungen zu Ihren individuellen Fragen. Ob klassisches Dashboard im Data Warehouse oder Vorhersage in der cloudbasierten Big Data Umgebung – mit Liebe zum Detail, mathematischer Kompetenz, Machine Learning Algorithmen und viel Erfahrung im Data Warehousing führen wir Sie nachhaltig zum Erfolg.

Ihr Ansprechpartner für Data Lakes

Giovanni Maccaferri
Big Data Engineer

Inhaltsverzeichnis

1 Einleitung

Die Business Intelligence vollzieht derzeit große Veränderungen, um den Anforderungen die durch Big Data entstanden sind gerecht zu werden. Laut einer Studie des IDC [dat18] wird sich die jährlich generierte Datenmenge bis zum Jahr 2025 um den Faktor zehn vergrößern (von 33 Zettabyte auf 175 Zettabyte). Da Unternehmen vermehrt danach streben aus Daten verschiedener Systeme (ERP, DWH, NoSQL, Webseiten, Social Media) neue Erkenntnisse zu gewinnen, benötigen die jetzigen BI-Lösungen eine Erweiterung um für steigende strukturierte und unstrukturierte Datenmengen gewappnet zu sein. Die bisherigen konventionellen Data Warehouses (DWH) erreichen ihre Grenzen wenn Terrabyte an Daten aus den verschiedensten Quellen aggregiert und gespeichert werden müssen. An dieser Stelle setzen Data Lakes (DL) an. Da es sich mehrheitlich um unstrukturierte Daten handelt, sind diese nicht direkt verwertbar. Trotzdem können die Daten für Unternehmen, zu einem späteren Zeitpunkt, eine wichtige Quelle für Auswertungen und Unternehmensausrichtungen sein. In diesem White Paper zeigen wir was für die Einrichtung eines Data Lakes notwendig ist und wie man auf diesen Reporten kann. Die ganze Infrastruktur wird auf AWS Basis nach dem best practice Ansatz realisiert. Darüber hinaus werden wir einen aktuellen Bug beseitigen, dass das Ausrollen der Infrastruktur verhindert.

2 Data Lake Definition

Die folgende Definition für den Data Lake setzt sich zusammen aus einem Text- und Tabellen-Teil. Abstrakt gesehen ist der Data Lake ein zentrales Speichermedium für strukturierte und unstrukturierte Daten, welcher viele Schnittstellen zu unterschiedlichen Quellen bietet, sowie mit neuen Verarbeitungstechnologien und Programmiersprachen angebunden werden kann [FTB+]. Die Tabelle 2 beinhaltet die Eigenschaften eines Data Lakes.

Eigenschaften	Data Lake
Skalierung	Skalierung sehr großer Datenmengen, geringe Kosten
Access	SQL-ähnliche Systeme, eigene Skripte u. Programme
Workload	Batch- und Streamverarbeitung
Daten	Rohdaten und veredelte Daten
Datenkomplexität	Komplexe Verarbeitungsmöglichkeiten
Kosten/Effizienz	Effiziente Speicher- und Weiterverarbeitungsmöglichkeiten
Benefits	Einfaches Konsumieren der Daten
	Schnelle Antwortzeit
	Mature governance
	Single enterprise-wide view of data
	Skaliert schnell auf huderten von Servern
	Kein Toolzwang
	Echtzeitanalyse
	Ermöglicht die Verwendung von strukturierten und unstrukturierten Daten aus einer Quelle
	Unterstützt Agile Modellierung der User wird ermöglicht Modell, Applikation und Abfragen zu gestalten
	Big Data Analyse
Drawbacks	Komplexität des Big Data Ökosystems
	Versumpfung des Data Lake wenn nicht gemanaged und organisiert
	Big Data skills gap

Tabelle 2.1: Eigenschaften eines Data Lakes [FTB+].

2.1 Referenzarchitektur

In Abbildung 2.1 wird eine Referenzarchitektur für einen Data Lake dargestellt. Der Data Lake wird in verschiedene Zonen aufgeteilt, wobei jede Zone eine bestimmte Aufgabe übernimmt. Die Verarbeitung der Daten läuft von links nach rechts, beginnend mit den Quellen (Source System) und der Vorverarbeitung (Transient Loading Zone) sowie der Raw Zone. Die Raw Zone beinhaltet die Rohdaten, die Analysten zu einem späteren Zeitpunkt in Ihre Zone kopieren können, um eine Auswertung durchzuführen. Im Anschluss daran befinden sich die Analysezonen, die eingeteilt sind in Trusted Zone, Refined Zone und Sandbox. In der Trusted Zone befinden sich Daten die als integer gelten. In der Refined Zone

befinden sich verschiedene Datentypen die soweit verfeinert wurden, dass diese für eine Auswertungen verwendet werden können. Die Sandbox dient dazu experimentelle Datenauswertungen durchzuführen ohne die Daten in der Raw Zone zu kompromittieren [FTB+] [GG18].

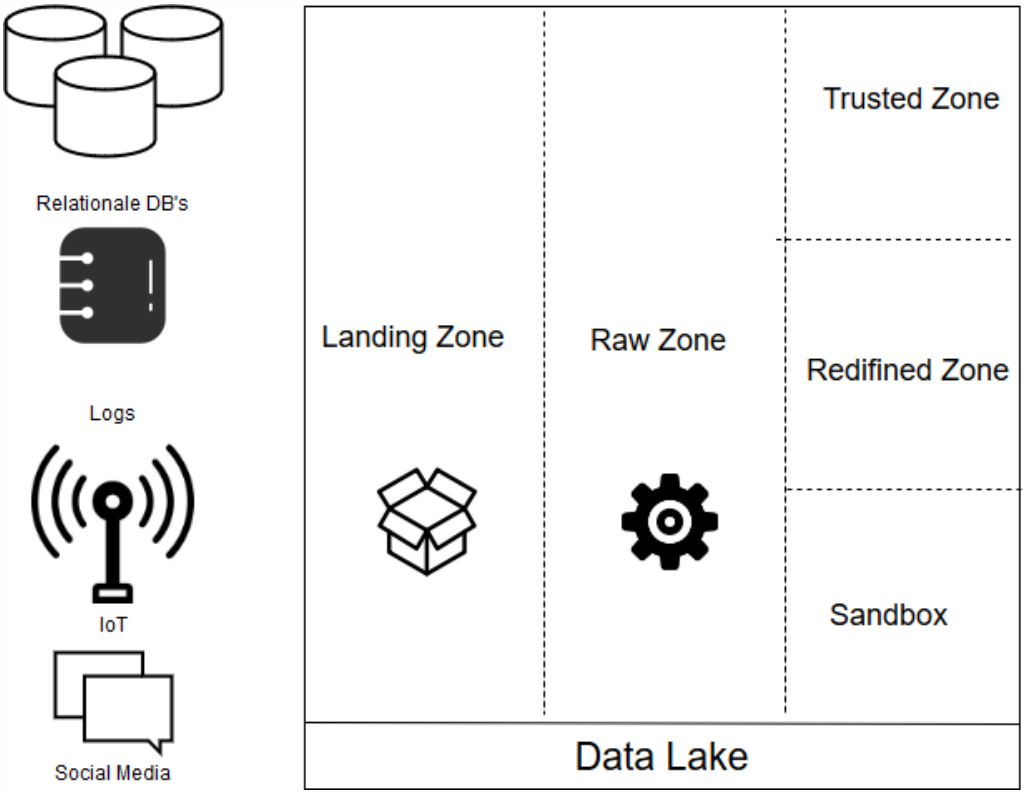

Abbildung 2.1: Referenzarchitektur für einen Data Lake.

2.2 AWS Architektur

Die AWS Architektur ist eine Umsetzung der oben aufgeführten Referenzarchitektur. Die in der Abbildung 2.2 dargestellte Data Lake Architektur wird mit Hilfe von AWS CloudFormation provisioniert. Die oben Beschriebenen Zonen werden hier als S3-Bucket realisiert, diese bilden nicht alle Zonen ab. Es besteht die Möglichkeit die Architektur mit weiteren Zonen zu erweitern. Der Core des Data Lakes ist das API Gateway der den Zugang zu den Microservices bereitstellt. Die Microservices sind auf Basis von AWS Lambda implementiert worden. Mit folgenden Microservices erhält der Data Lake seine Businesslogik:

- Erstellen von Daten-Paketen

- Hochladen von Daten
- Suche nach erstellten Daten-Paketen
- Cart-System für Daten-Pakete
- Generierung von Datenmanifesten
- Adminverwaltung

Die aufgeführten Microservices nehmen folgende Dienste in Anspruch AWS S3, AWS Glue, Amazon Athena, Amazon DynamoDB, Amazon ES und Amazon Cloud-Watch. Mit Hilfe der Data Lake Konsole kann der Lake über ein Webfrontend oder CLI (command line interface) verwaltet werden. Das Webfrontend wird auf einem S3-Bucket gehostet [aws18].

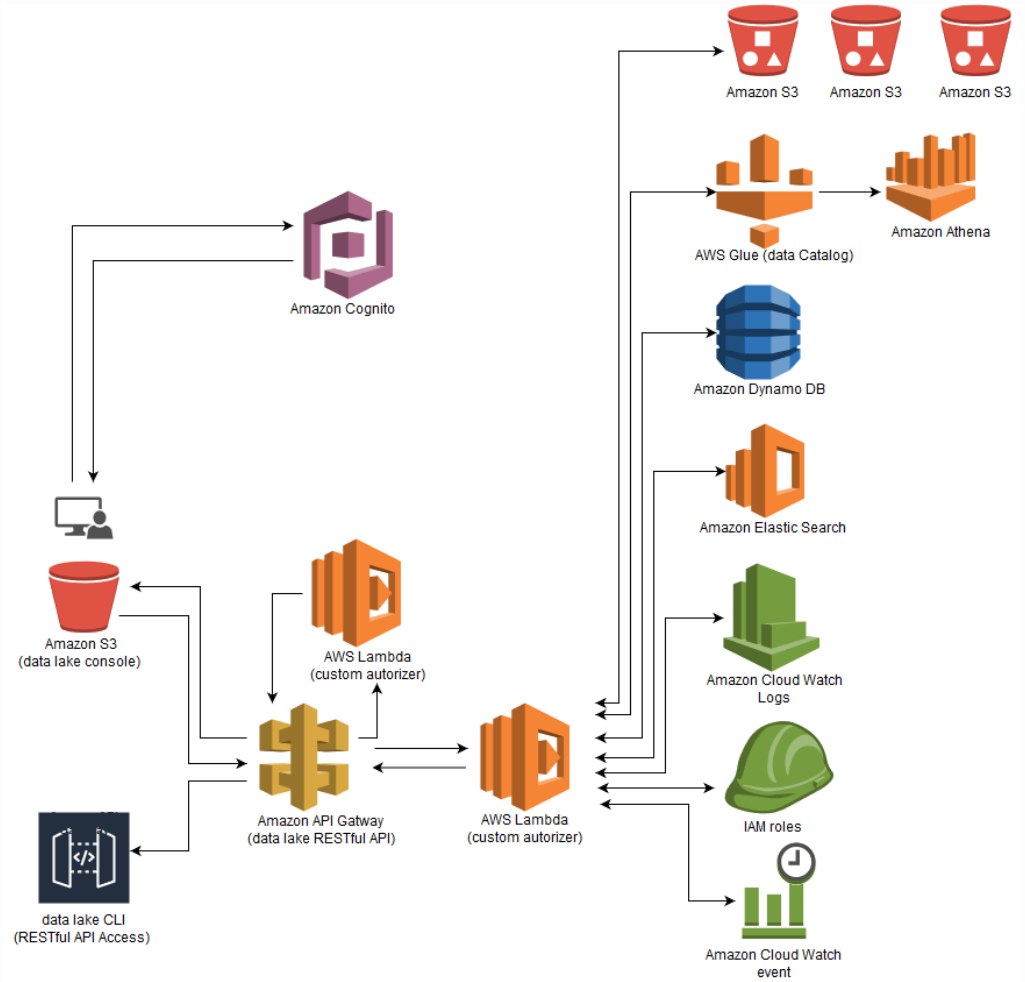

Abbildung 2.2: Data Lake Architektur auf AWS [aws18].

3 Implementation einer DL-Umgebung

Im folgenden Abschnitt werden die Modifikation, das Ausrollen und das Terminieren der Infrastruktur erläutert und mit Screenshots visualisiert. Im Anschluss wird auf Problematiken des Deployment näher eingegangen.

3.1 Aufbau der DL-Infrastruktur

Die Infrastruktur ist nach dem Ansatz infrastructure as a code (IAC) entwickelt worden. Die Konfiguration der Infrastruktur wird programmatisch erstellt. Aus diesem Konzept heraus setzt sich das Deployment aus fünf Konfigurationsdateien zusammen, welche im folgenden erläutert werden:

- data-lake-deploy-federated.template
- data-lake-deploy.template
- data-lake-api.yaml
- data-lake-services.yaml
- data-lake-storage.yaml

Die Template-Datei data-lake-deploy-federated.template wird in diesem Data Lake nicht verwendet. Der Hintergrund dabei ist, dass kein Active-Directory eingebunden wird. In der data-lake-deploy.template wird das Hauptgerüst des DL beschrieben. Dabei werden die jeweiligen YAML-Dateien für API, Services und Storage ausgerollt. Von dem Dateinamen ableitbar werden die entsprechenden Bereiche der DL-Infrastruktur ausgerollt. Die oben aufgeführte Grundstruktur stellt AWS zur Verfügung. In diesen Dateien kann man an den entsprehenden Stellen die Architektur-Erweiterungen und Modifikationen vornehmen. Die Modifikationen werden im nächsten Abschnitt erläutert. Um den Code auf Cloud-Formationen auszurollen muss ein Paket gebaut werden. Dafür sind folgende Schritte notwendig:

Build-Skript Ausführen

```
chmod +x build-s3-dist.sh

./build-s3-dist.sh \$DEPLOY_BUCKET \$VERSION\_CODE

aws s3 cp ./dist s3://\$DEPLOY_BUCKET-\${AWS_REGION}/data-lake/latest
--recursive --acl bucket-owner-full-control
```

Codeblock 3.1.0.1: Build-Skript Befehle

In dem Build-Skript werden alle notwendigen nodejs abhängigen Librarys abgerufen und ein Deploy-Ordner erstellt mit allen notwendigen Dateien, damit dieser anschließend in ein S3-Bucket hochgeladen werden kann.

3.2 Modifikation der Infrastruktur

Bei dem Data Lake werden folgende Modifikationen Beispielsweise durchgeführt: ES-Server verkleinern um Kosten im Demobetrieb zu reduzieren, die Anonyme-Nutzungsstatistik von AWS deaktivieren und die S3-Buckets verschlüsseln .

ES-Server
In der Datei data-lake-storage.yaml wird der ES-Server modifiziert. Hier werden jeweils die Values für InstanceType und DedicatedMasterType in allen Regionen angepasst. Für die Nodes werden Micro-EC2 Instanzen gesetzt und für die Masternode eine Small-EC2 Instanz. Im folgenden Codeblock 3.2.0.2 sieht man einen Ausschnitt der Datei.

S3-Verschlüsselung
Der Verschlüsselungsparameter wird in der Template-Datei gesetzt, dieser wird mit den Properties übergeben. Der folgende Source Code 3.2.0.3 zeigt wie der Verschlüsselungsparameter gesetzt wird.

```
Mappings:
    RegionMap:
        ...
        eu-west-1:
            "InstanceType": "t2.micro.elasticsearch"
            "DedicatedMasterType": "t2.small.elasticsearch"
        eu-west-2:
            "InstanceType": "t2.micro.elasticsearch"
            "DedicatedMasterType": "t2.small.elasticsearch"
        eu-central-1:
            "InstanceType": "t2.micro.elasticsearch"
            "DedicatedMasterType": "t2.small.elasticsearch"
        ...
```

Codeblock 3.2.0.2: Deploy-Datei ES-Server

```
...
    DataLakeS3Resources:
        Type: "Custom::LoadLambda"
        Properties:
            BucketEncryption:
                ServerSideEncryptionConfiguration:
                    - ServerSideEncryptionByDefault:
                    SSEAlgorithm: AES256
            ServiceToken:
                Fn::GetAtt:
                    - "DataLakeHelper"
                    - "Arn"
            Region:
                - Ref: "AWS::Region"
            dataLakeDefaultBucket: !Join ["", ["data-lake-", Ref: "AWS::Region",
            "-", Ref: "AWS::AccountId"]]
            dataLakeWebsiteBucket: !Join ["", ["datalakeweb-", Ref: "AWS::Region",
            "-", Ref: "AWS::AccountId"]]
            customAction: "configureDatalakeBuckets"
...
```

Codeblock 3.2.0.3: Deploy-Datei für die S3-Verschlüsselung

Anonyme-Nutzungsstatistik

Die Anonyme-Nutzungsstatistik wird über den Parameter SendAnonymousUsageData gesteuert. Wenn dieser auf No gesetzt ist, werden keine Daten gesendet. Möchte man sicher gehen, dass keine Nutzungsdaten generiert werden, sollte man weiter unten im Code den angezeigten Codeblock 3.2.0.4 entfernen.

```
    DataLakeAnonymousMetric:
        Type: "Custom::LoadLambda"
        Properties:
            ServiceToken:
                Fn::GetAtt:
                    - "DataLakeHelper"
                    - "Arn"
            Region:
                - Ref: "AWS::Region"
            solutionId: "SO0017"
            UUID: !GetAtt DataLakeUuid.UUID
            version: "1"
            anonymousData: !FindInMap ["Solution", "Data", "SendAnonymousUsageData"]
            customAction: "sendMetric"
```

Codeblock 3.2.0.4: Codeblock Anonyme Nutzungsstatistik

3.3 Deployment und Terminieren

Das Deployment und Terminieren der DL-Infrastruktur kann entweder über das Webinterface oder wahlweise über die Console von AWS CloudFormation erfolgen. Für das Deployment wird die URL des Buckets mit dem DL-Template benötigt. Hier sollte darauf geachtet werden, in welcher Region man sich gerade befindet, da diese für das Ausrollen verwendet wird.

Um den DL zu terminieren, benötigt man lediglich den Stack-Name (siehe Codeblock 3.3.0.5).

```
aws cloudformation delete-stack --stack-name DataLakeProject1 --retain-resources
"DataLakePackageCrawlerRole" "DataLakePackagesRole" --profil accountadmin
```

Codeblock 3.3.0.5: Terminieren des DL

3.4 Probleme beim Deployment im CloudFormation Skript

Während der Entwicklung des eigenen Deployment-Skripts, ist es zu Abbrüchen beim Ausrollen auf CloudFormation gekommen. Dieser Fehler trat auf wenn der AWS Source Code von Github [git] verwendet wurde. Der Ursprung des Fehlers war, dass beim builden die Region mit den Template-Bucket-Namen konkateniert wird. Hier gibt es zwei mögliche Workarounds, welche im Laufe des Projektes erarbeitet worden sind. Beim ersten Workaround werden in der Template-Datei alle S3-Bucket-Links so modifiziert, dass die Region gelöscht wird. Der zweite Workaround lässt sich von dem ersten ableiten, beim Upload wird der Regionsname übergeben (siehe 3.1).

Hier ein Beispiel: Bucket-Name data-lake-template, Region: eu-west-1. Das Template-Bucket: data-lake-template-eu-west-1

4 Workflows in der DL-Umgebung

Im folgendem Abschnitt werden die Workflows eines Adminstrators und Analysten in der Data Lake Umgebung analysiert und visualisiert. Es wird auf die Data Governance Möglichkeiten eingegangen, sowie auf die Datenauswertung.

4.1 Workflow Admin

Als Admin übernimmt man folgende Aufgaben, das Verwalten von Benutzern, Rollen, Gruppen. Die Acces-Keys-Generierung, Datenverwaltung, Verwalten von Metainformationen.

Benutzerverwaltung
Die Benutzerverwaltung ist intuitiv und übersichtlich gehalten siehe Abbildung 4.1. Bei den Rollen existieren zwei Auswahlmöglichkeiten, Admin oder Member. Eine Erweiterung der Rechte ist über das Webfrontend nicht vorgesehen.

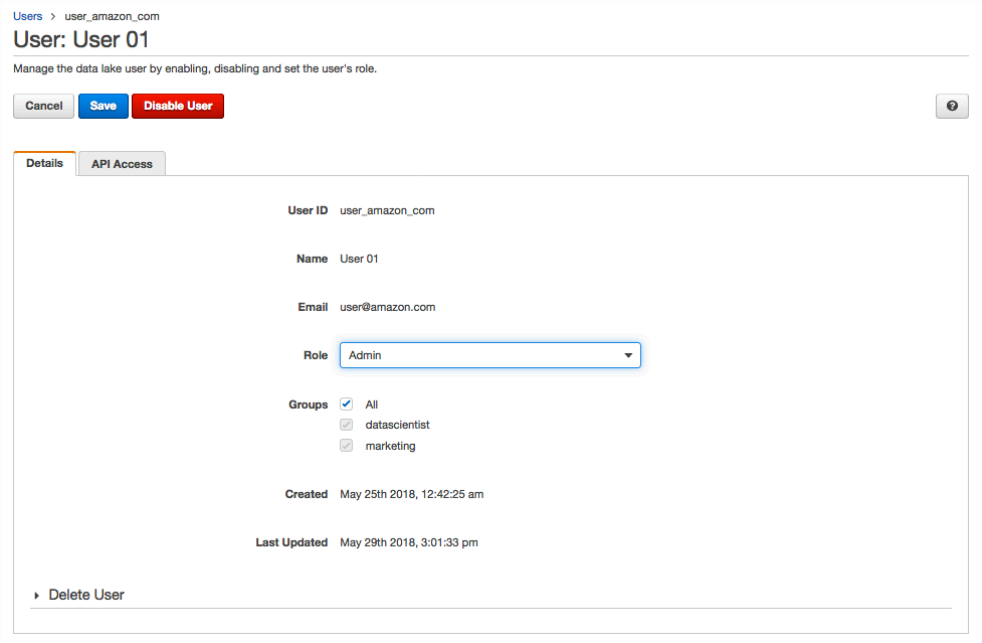

Abbildung 4.1: Data Lake Benutzerverwaltung

Gruppen

Die Gruppen lassen sich beliebig erstellen, damit ist eine Steuerung des Datenzugriffs realisierbar. Beim hochladen von Daten besteht die Möglichkeit nur bestimmten Gruppen den Zugriff zu ermöglichen.

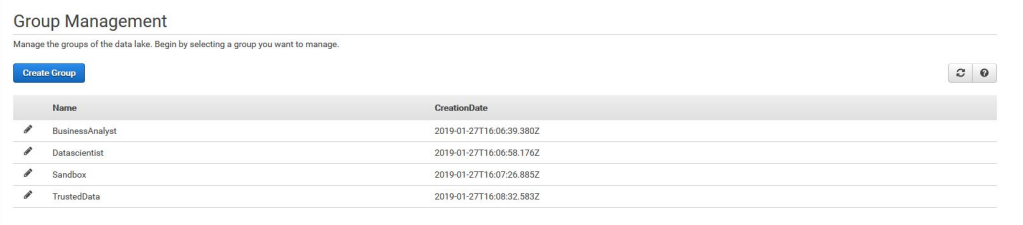

Abbildung 4.2: Data Lake Gruppenrverwaltung

Access-Keys

Die Acces-Keys sind dafür gedacht bestimmten Benutzern einen Zugang über die CLI zu ermöglichen. Darüber können automatisierte Datenuploads gesteuert werden. Das Manko dabei ist, dass diese Data Lakes spezifische CLI nur von Linux und OS X unterstützt wird.

Metainfomrationen

Die Metainformationen die von eingehenden Daten gesammelt werden, lassen sich nicht direkt einsehen. Diese werden im ES-Server gespeichert. Hier besteht

die Möglichkeit über Kibana die Daten zu visualisieren. Die URL zu dem Link befindet sich unter Settings -> General -> Amazon Elasticsearch Kibana Url. Über die Webansicht besteht die Möglichkeit die Metadaten zu erweitern, dabei kann festgelegt werden ob diese verpflichtend sind. Diese Option befindet sich unter Settings -> Governance (siehe Abbildung 4.3).

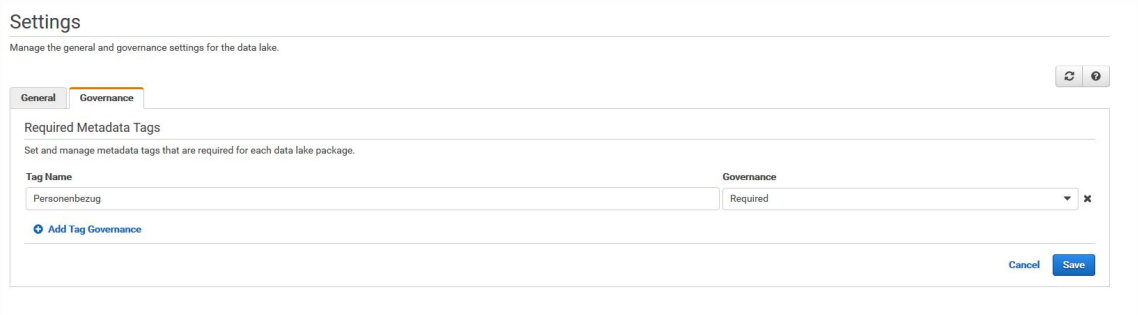

Abbildung 4.3: Data Lake Metadata Tag

Im Anhang A befindet sich eine komplette Metainformation, die von dem Glue Crawler erstellt worden ist. Hier erhält man einen Einblick darüber, welche der Informationen protokolliert werden. Diese Metadaten bilden die Basis für AWS Athena. Darüber hinaus besteht die Möglichkeit ein DG-Dashboard zu implementieren, um KPIs wie Data quantity, Data volume oder interne Leistungsspitzen zu identifizieren.

4.2 Bereitstellen von Daten

Das Konzept in diesem DL besteht darin Datensätze in Form eines Package bereitzustellen. Diese Packages werden durch Package Name, Description, Visability und custom Metainformationen beschrieben (siehe Abb. 4.3). Nach dem Anlegen eines Package können Daten hochgeladen werden. Dies kann auf zwei Arten erfolgen: jedes File einzeln oder über ein S3-Bucket. Für das Einbinden eines S3-Buckets wird ein Manifest benötigt, der Vorteil besteht darin, das neue Datensätze automatisiert eingebunden werden. Die Manifest-Source wird wie folgt formatiert:

Integration der Daten
Mit dem Hochladen der Datensätze wird ein AWS Glue crawler getriggert. Dieser Crawler scannt die Datensätze auf Metainformationen und hält diese in einem Data Catalog nach. Der Crawler ist dabei in der Lage zusammenhängende Datensätze zu erkennen und diese logisch zusammenzufassen. Ist ein Scan erfolgreich, können die Daten mit einem Analyse Service direkt ausgewertet werden, oder man exportiert sich die Daten über ein Manifest-File in ein eigenes S3-Bucket. Ist eine Integration erfolgreich, werden die Datensätze mit jeweils

```
{
    "dataStore": [
        {
            "includePath": "s3://datalake.cidd.test/chicago-taxi/"
        }
    ]
}
```

Codeblock 4.2.0.6: Manifest-Datei

zwei Verlinkungen aufgelistet. Eine verweist auf den Daten Katalog in Glue (1), die zweite zeigt auf Athena (2). In Abbildung 4.4 sind diese Verlinkungen mit 1 und 2 gekennzeichnet.

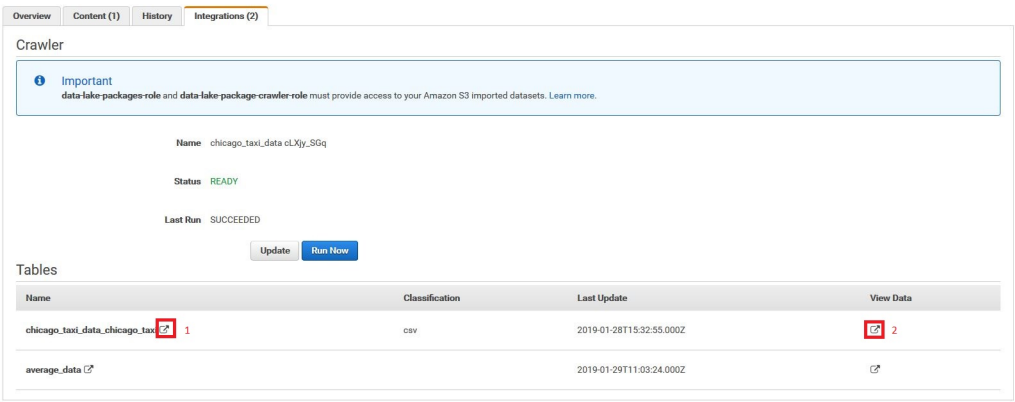

Abbildung 4.4: Data Lake Integration

4.3 Workflow Analyst

In diesem Abschnitt soll vermittelt werden, wie der Workflow eines Analysten aussieht. Für die Extraktion der Daten wird kein klassisches ETL benötigt, da Athena anhand von Metadaten direkt auf die Objektebene in S3 zugreifen kann. Das Bedeutet das eine SQL-Query keine Tabellen sondern Dateien durchsucht. Um das bekannte Look and Feel beizubehalten, ist die Oberfläche an gängigen SQL-Developers angelehnt, jedoch mit einigen Einschränkungen hinsichtlich der Funktionalitäten.

Nachdem Login befindet man sich im Dashboard des Webfrontends. Hier besteht die Möglichkeit nach angelegten Packages zu suchen. Die Suche basiert

auf Einträgen des Elastic Search Clusters, die beim Anlegen des Package gespeichert worden sind. Wird ein Package ausgewählt kann über den Integration-Tab (siehe Abb. 4.4), die Verlinkung zu Athena verwendet werden, um einen Überblick über die Daten zu erhalten. In Athena selbst können die Daten mit dem Query-Editor abgefragt werden (siehe Abb. 4.6). Unter anderem besteht die Möglichkeit Views anzulegen. An dieser Stelle hat der Analyst drei mögliche Verfahrensweisen:

- Manifest-Datei
- Export per Download
- Visualisierung

Das Ganze wird in Abbildung 4.5 in Form eines Flow-Chart visualisiert.

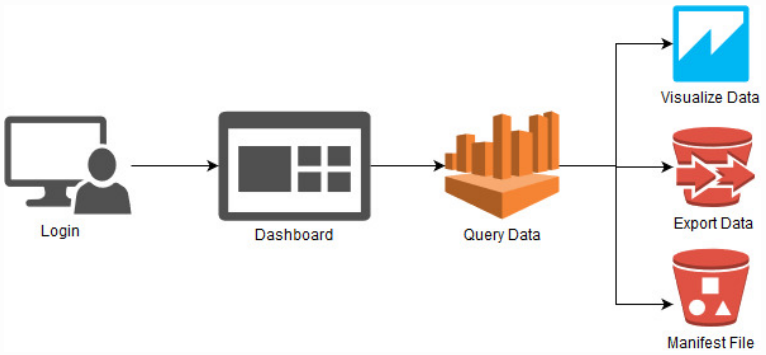

Abbildung 4.5: Workflow Business Analyst

Die Manifest-Datei sorgt dafür, dass jeweils der aktuellste Datensatz aus einem Package in einem persönlichen S3-Bucket transferiert wird. In Athena können die modifizierten Daten per Download heruntergeladen werden oder für eine direkte Visualisierung in Quick Sight verwendet werden.

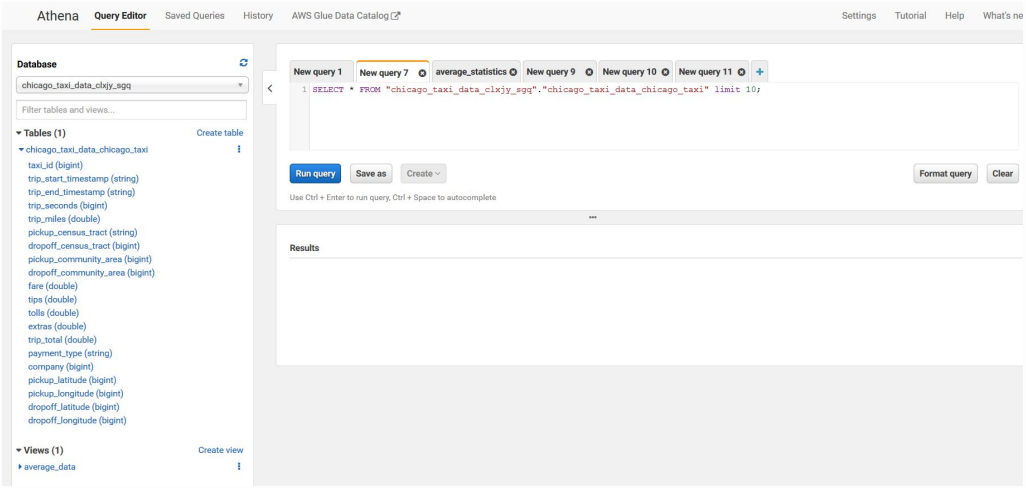

Abbildung 4.6: Athena Query Editor

Die Performance der Abfragen ist im Vergleich zu einer Datenbank um ein vielfaches langsamer. Ist aber für eine Abfrage auf Datei Ebene ohne Indexe und Caches jedoch akzeptabel. Um ein Gefühl für die Performance zu erhalten, wählen wir den Chicago Taxi Datensatz. Zunächst ein paar Basisinformationen zu diesem Datensatz:

- 2 GB (S3)
- 15 Dateien
- Format CSV
- 18 Columns
- 19.863.224 Rows

Es werden folgende Standard SQL-Abfragen durchgeführt: Select *, Count, Create View

Query-Typ	Data scanned in GB	Zeit in sec
Count	2.02	6.57
Select*	2.02	152
Select* ... Where=X	2.02	81
Create View	2.02	7.15
Union	4.05	9.85

Tabelle 4.1: Query Performance Athena

Wie der Tabelle 4.1 zu entnehmen ist, hängt die Dauer der Abfrage nicht zwangsläufig mit der Menge der Datensätze zusammen. Vergleicht man die Union Operation mit dem Count, liegt die Differenz bei 3,28 Sekunden, wobei beim Union die doppelte Menge an Daten gescannt worden sind. Bei den Select-Statements sieht man ebenfalls eine Abweichung der Laufzeit, sobald eine WHERE-Bedingung eingebaut wird läuft die Abfrage 71 Sekunden länger bei gleicher Datenmenge. Um Cache-Effekte auszuschließen wurde die Select-Abfrage mit dem WHERE-Statement nochmals verzögert ausgeführt, wobei die Laufzeit vergleichbar war. Die Messungen sollen lediglich ein Gefühl für die Laufzeiten eines solchen Systems vermitteln.

Visualisierung mit Quick sight

Mit Hilfe von Quick Sight kann ein Analyst Daten direkt visualisieren und als Dashboard bereitstellen oder als Email-Report versenden. Als Datenquelle kann entweder die Tabelle aus Athena angegeben werden oder die Daten werden in den Quick Sight Speicher geladen. Die zweite Option sorgt für einen schnelleren Zugriff. Der Speicher ist dabei auf 10 GB begrenzt.

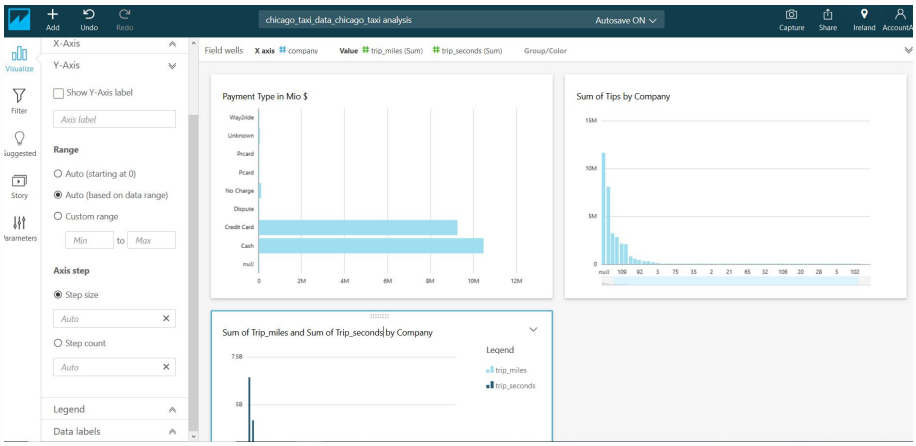

Abbildung 4.7: Quick Sight Editor

5 Fazit

Im abschließenden Kapitel werden folgende Aspekte diskutiert: Usability, Security Webfrontend und AWS als DL-Hoster.

5.1 Usability

In Kapitel 4.3 wird ein Analyse Workflow behandelt. Dabei ist aufgefallen, dass für eine Auswertung eine Weiterleitung auf Athena erfolgt. Es ist erforderlich die DL Web-Umgebung zu verlassen um die Auswertung durchzuführen. In der Testphase gab es keine Probleme, da die Testperson einen AWS-Benutzer hatte. Bei DL-Usern ohne AWS-Konto führt dies jedoch zu Problemen, weil diese den Service so nicht nutzen können. Abhilfe würde entweder eine Integration der Athena API schaffen oder die Verbindung der DL-User mit dem IAM von AWS. Dort müssen die entsprechenden Rechte für DL-Konten gesetzt werden. Durch die Vielzahl an Weiterleitungen zu anderen AWS Services, fühlt sich die DL-Umgebung sehr heterogen an.

5.2 Security Webfrontend

Durch den Unit Test beim Deployment werden veraltete oder nicht mehr unterstützte nodejs-Libs identifiziert. Dieses sollte vor einem Onlinegang überprüft werden um eine Kompromitierung durch Sicherheitslücken zu unterbinden.

5.3 AWS als DL-Host

Der Ansatz des SaaS und IaC bringt enorme Vorteile mit sich. Die Zeiteinsparung durch die Nutzung von Softwarediensten ist beträchtlich. Hierdurch kann direkt mit der Bewirtschaftung des DL angefangen werden. Es wird keine zusätzliche Zeit für die Installation und Konfiguration benötigt. AWS bietet unter anderem Open Source Software wie Hadoop und ES an. Diese sind fertig konfiguriert und

optimiert. Durch diesen Service entfallen Kosten für die Aufrechterhaltung des laufenden Betriebs. Der Ansatz IaC ermöglicht eine Versionierung der Infrastruktur über ein Git-Repository. Auch wenn der Code für die Infrastruktur anfänglich sehr umfangreich scheint und nicht leicht verständlich ist, überwiegen die Vorzüge der automatisierten Provisionierung. Die Dokumentationen der einzelnen AWS Dienste sind sehr ausführlich. Wobei die Anleitung für den DL auf Github einen Fehler enthält der nicht dokumentiert wurde. Dies führt dazu, dass das Deployment nicht ohne weitere Anpassung funktioniert (siehe Abschnitt 3.4). Durch die hohe Komplexität der Architektur ist die Erweiterung des DL durch weitere Dienste nicht trivial. Der Dienst CloudFormation, womit der DL ausgerollt wird ist zwar dokumentiert, zumeist werden aber nur simple Beispiele behandelt. Weiterbildungen sind größtenteils nur durch kostenpflichtige Schulungen möglich. Die Fehlerbehandlung beim Deployment ist sehr mühselig, weil es keinen Debugger gibt. Lediglich die Syntax des Templates wird überprüft.

A Metainfomration aus Kibana


```
{
  "took": 2,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 2,
    "max_score": 1,
    "hits": [
      {
        "_index": "data-lake",
        "_type": "package",
        "_id": "cLXjy_SGq",
        "_score": 1,
        "_source": {
          "owner": "testuser",
          "metadata": [
            {
              "tag": "Personenbezug",
              "value": "TRUE"
            }
          ],
          "column_name": [
            "taxi_id",
            "trip_start_timestamp",
            "trip_end_timestamp",
            "trip_seconds",
            "trip_miles",
            "pickup_census_tract",
            "dropoff_census_tract",
            "pickup_community_area",
            "dropoff_community_area",
            "fare",
```

```
            "tips",
            "tolls",
            "extras",
            "trip_total",
            "payment_type",
            "company",
            "pickup_latitude",
            "pickup_longitude",
            "dropoff_latitude",
            "dropoff_longitude"
          ],
          "groups": [
            "BusinessAnalyst",
            "Datascientist",
            "Sandbox"
          ],
          "created_at": "2019-01-28T14:09:05Z",
          "description": "Chicago Taxi Data 2019",
          "package_id": "cLXjy_SGq",
          "table_stats": {
            "sizeKey": 2170926889,
            "objectCount": 14,
            "recordCount": 12996909,
            "averageRecordSize": "164.00"
          },
          "table_desc": [],
          "deleted": false,
          "updated_at": "2019-01-28T14:16:39Z",
          "name": "Chicago Taxi Data",
          "column_comment": []
        }
      },
      {
        "_index": "data-lake",
        "_type": "package",
        "_id": "E3Bba6XZN",
        "_score": 1,
        "_source": {
          "updated_at": "2019-01-28T14:58:01Z",
          "groups": [
            "BusinessAnalyst",
            "Datascientist",
            "Sandbox",
            "TrustedData"
          ],
          "package_id": "E3Bba6XZN",
```

```
84          "created_at": "2019-01-28T14:34:13Z",
85          "deleted": false,
86          "owner": "testuser",
87          "description": "test",
88          "name": "test",
89          "metadata": []}}]}}
```

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[aws18] Data Lake Solution. https://s3.amazonaws.com/solutions-reference/data-lake-solution/latest/data-lake-solution-on-aws.pdf. Version: 2018

[dat18] Prognose zum Volumen der jährlich generierten digitalen Datenmenge weltweit in den Jahren 2018 und 2025 (in Zettabyte)). https://de.statista.com/statistik/daten/studie/267974/umfrage/prognose-zum-weltweit-generierten-datenvolumen/. Version: 2018

[FTB+] Farnham, Boston ; Tokyo, Sebastopol ; Boston, Beijing ; Sebastopol, Farnham ; Beijing, Tokyo: Architecting Data Lakes Data Management Architectures for Advanced Business Use Cases SECOND EDITION. – Forschungsbericht

[GG18] Gupta, S. ; Giri, V.: Practical Enterprise Data Lake Insights: Handle Data-Driven Challenges in an Enterprise Big Data Lake. Apress, 2018 https://books.google.de/books?id=kVdiDwAAQBAJ. – ISBN 9781484235225

[git] aws-data-lake-solution. https://github.com/awslabs/aws-data-lake-solution/tree/master/deployment