

CDP Cookbook

- Apache Hadoop Befehle
 - Überblick
 - appendToFile
 - cat
 - chgrp
 - chmod
 - chown
 - copyFromLocal
 - copyToLocal
 - count
 - cp
 - du
 - dus
 - expunge
 - get
 - getfacl
 - getmerge
 - ls
 - lsr
 - mkdir
 - moveFromLocal
 - moveToLocal
 - mv
 - put
 - rm
 - rmr
 - setfacl
 - setrep
 - stat
 - tail
 - test
 - text
 - touchz
- HUE - HIVE Befehle
 - Allgemein
 - Tealium JSON-Dateien verarbeiten
 - Array<string> auswerten
 - MAP<string Strukt<name:string, ... >> auswerten
 - Datenbank - „Location“ anpassen (<http://gaganonthenet.com/2015/02/23/hive-change-location-for-database-or-schema/>)
 - External-Tabelle-Partition erzeugen
 - Tabelle-Partition erzeugen
 - Workflow hängt ohne ersichtlichen Grund

Apache Hadoop Befehle

Überblick

Die FS-Shell enthält verschiedene Shell-ähnliche Befehle, die direkt mit dem Hadoop Distributed File System (HDFS) sowie anderen von Hadoop unterstützten Dateisystemen wie Local FS, HFTP FS, S3 FS und anderen interagieren. Die FS-Shell wird aufgerufen von:

- bin / hadoop fs <args>

Alle FS-Shell-Befehle verwenden Pfad-URIs als Argumente. Das URI-Format ist Schema: // Autorität / Pfad. Für HDFS ist das Schema hdfs, und für das lokale FS ist das Schema eine Datei. Das Schema und die Autorität sind optional. Wenn nicht angegeben, wird das in der Konfiguration angegebene Standardschema verwendet. Eine HDFS-Datei oder ein HDFS-Verzeichnis wie / parent / child kann als hdfs: // namenodehost / parent / child oder einfach als / parent / child angegeben werden (vorausgesetzt, dass Ihre Konfiguration auf hdfs: // namenodehost verweist). Die meisten Befehle in der FS-Shell verhalten sich wie entsprechende Unix-Befehle. Unterschiede werden mit jedem der Befehle beschrieben. Fehlerinformationen werden an stderr gesendet und die Ausgabe wird an stdout gesendet.

- <https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/FileSystemShell.html>
- Berechtigungshandbuch <https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-hdfs/HdfsPermissionsGuide.html>
- <https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html>

appendToFile

Verwendung: `hdfs dfs-appendToFile <localrc> ... <dst>`

Hängen Sie einzelne Quellen oder mehrere Quellen vom lokalen Dateisystem an das Zieldateisystem an. Liest auch die Eingabe von stdin und hängt das an das Zieldateisystem.

- `hdfs dfs -appendToFile lokale Datei / Benutzer / hadoop / hadoopfile`
- `hdfs dfs -appendToFile localfile1 localfile2 / Benutzer / hadoop / hadoopfile`
- `hdfs dfs -appendToFile lokale Datei hdfs: //nn.example.com/hadoop/hadoopfile`
- `hdfs dfs-appendToFile - hdfs: //nn.example.com/hadoop/hadoopfile` Liest die Eingabe von stdin.

Beendigungscode:

Gibt 0 bei Erfolg und 1 bei Fehler zurück.

cat

Verwendung: `hdfs dfs -cat URI [URI ...]`

Kopiert Quellpfade in stdout.

Beispiel:

- `hdfs dfs -cat hdfs: //nn1.example.com/file1 hdfs: //nn2.example.com/file2`
- `hdfs dfs -cat datei: /// datei3 / benutzer / hadoop / datei4`

Beendigungscode:

Gibt 0 bei Erfolg und -1 bei Fehler zurück.

chgrp

Verwendung: `hdfs dfs -chgrp [-R] GROUP URI [URI ...]`

Ändern Sie die Gruppenzuordnung von Dateien. Der Benutzer muss der Eigentümer von Dateien oder ein Super-Benutzer sein. Weitere Informationen finden Sie im [Berechtigungshandbuch](#).

Optionen

- Mit der Option -R wird die Änderung rekursiv über die Verzeichnisstruktur vorgenommen.

chmod

Verwendung: `hdfs dfs -chmod [-R] <MODE [, MODE] ... | OKTALMODUS> URI [URI ...]`

Ändern Sie die Berechtigungen von Dateien. Nehmen Sie mit -R die Änderung rekursiv über die Verzeichnisstruktur vor. Der Benutzer muss der Eigentümer der Datei oder ein Super-Benutzer sein. Weitere Informationen finden Sie im [Berechtigungshandbuch](#).

Optionen

- Mit der Option -R wird die Änderung rekursiv über die Verzeichnisstruktur vorgenommen.

chown

Verwendung: `hdfs dfs -chown [-R] [EIGENTÜMER] [: [GROUP]] URI [URI]`

Ändern Sie den Eigentümer der Dateien. Der Benutzer muss ein Super-Benutzer sein. Weitere Informationen finden Sie im [Berechtigungshandbuch](#).

Optionen

- Mit der Option -R wird die Änderung rekursiv über die Verzeichnisstruktur vorgenommen.

copyFromLocal

Verwendung: `hdfs dfs-copyFromLocal <localrc> URI`

Ähnlich wie `put`-Befehl, nur dass die Quelle auf eine lokale Dateireferenz beschränkt ist.

Optionen:

- Die Option `-f` überschreibt das Ziel, falls es bereits existiert.

copyToLocal

Verwendung: `hdfs dfs -copyToLocal [-ignorecrc] [-crc] URI <lokale_dst>`

Ähnlich dem Befehl `get`, außer dass das Ziel auf eine lokale Dateireferenz beschränkt ist.

count

Verwendung: `hdfs dfs -count [-q] <Pfade>`

Zählen Sie die Anzahl der Verzeichnisse, Dateien und Bytes unter den Pfaden, die dem angegebenen Dateimuster entsprechen. Die

Ausgabespalten mit `-count` lauten: `DIR_COUNT, FILE_COUNT, CONTENT_SIZE FILE_NAME`

Die Ausgabespalten mit `-count -q` sind: `QUOTA, REMAINING_QUOTA, SPACE_QUOTA, REMAINING_SPACE_QUOTA, DIR_COUNT, FILE_COUNT, CONTENT_SIZE, FILE_NAME`

Beispiel:

- `hdfs dfs -count hdfs://nn1.example.com/file1 hdfs://nn2.example.com/file2`
- `hdfs dfs -count -q hdfs://nn1.example.com/file1`

Beendigungscode:

Gibt 0 bei Erfolg und -1 bei Fehler zurück.

cp

Verwendung: `hdfs dfs -cp [-f] URI [URI ...] <dest>`

Kopieren Sie Dateien von der Quelle zum Ziel. Dieser Befehl erlaubt auch mehrere Quellen. In diesem Fall muss das Ziel ein Verzeichnis sein.

Optionen:

- Die Option `-f` überschreibt das Ziel, falls es bereits existiert.

Beispiel:

- `hdfs dfs -cp /Benutzer/Hadoop/Datei1 /Benutzer/Hadoop/Datei2`
- `hdfs dfs -cp /benutzer/hadoop/datei1 /benutzer/hadoop/datei2 /benutzer/hadoop/dir`

Beendigungscode:

Gibt 0 bei Erfolg und -1 bei Fehler zurück.

du

Verwendung: `hdfs dfs -du [-s] [-h] URI [URI ...]`

Zeigt die Größe der im angegebenen Verzeichnis enthaltenen Dateien und Verzeichnisse oder die Länge einer Datei an, falls es sich nur um eine Datei handelt.

Optionen:

- Die Option `-s` führt zu einer Zusammenfassung der angezeigten Dateilängen anstelle der einzelnen Dateien.
- Die Option `-h` formatiert Dateigrößen in "lesbarer" Form (zB 64.0m anstelle von 67108864)

Beispiel:

- `hdfs dfs -du / Benutzer / hadoop / Verzeichnis1 / Benutzer / hadoop / Datei1` `hdfs: //nn.example.com/user/hadoop/dir1`

Exit Code: Gibt 0 bei Erfolg und -1 bei Fehler zurück.

dus

Verwendung: `hdfs dfs -dus <args>`

Zeigt eine Zusammenfassung der Dateilängen an. Dies ist eine alternative Form von `hdfs dfs -du -s`.

expunge

Verwendung: `hdfs dfs -expunge`

Den Papierkorb leeren. Weitere Informationen zum Papierkorb finden Sie im [HDFS-Architekturhandbuch](#).

get

Verwendung: `hdfs dfs -get [-ignorecrc] [-crc] <src> <lokale_dst>`

Kopieren Sie Dateien in das lokale Dateisystem. Dateien, die die CRC-Prüfung nicht bestehen, können mit der Option `-ignorecrc` kopiert werden. Dateien und CRCs können mit der Option `-crc` kopiert werden.

Beispiel:

- `hdfs dfs -get / user / hadoop / Datei lokale Datei`
- `hdfs dfs -get hdfs: //nn.example.com/user/hadoop/file lokaleDatei`

Beendigungscode:

Gibt 0 bei Erfolg und -1 bei Fehler zurück.

getfacl

Verwendung: `hdfs dfs -getfacl [-R] <Pfad>`

Zeigt die Zugriffssteuerungslisten (ACLs) von Dateien und Verzeichnissen an. Wenn ein Verzeichnis über eine Standard-ACL verfügt, zeigt `getfacl` auch die Standard-ACL an.

Optionen:

- `-R`: Listet die ACLs aller Dateien und Verzeichnisse rekursiv auf.
- `Pfad` : Datei oder Verzeichnis zur Liste.

Beispiele:

- `hdfs dfs -getfacl / Datei`
- `hdfs dfs -getfacl -R / dir`

Beendigungscode:

Gibt 0 bei Erfolg und nicht Null bei Fehler zurück.

getmerge

Verwendung: `hdfs dfs -getmerge <src> <localdst> [addnl]`

Nimmt ein Quellverzeichnis und eine Zieldatei als Eingabe und verkettet Dateien in `src` in die lokale Zieldatei. Optional kann `addnl` so eingestellt werden, dass am Ende jeder Datei ein Zeilenvorschubzeichen hinzugefügt werden kann.

ls

Verwendung: `hdfs dfs -ls <args>`

Für eine Datei gibt stat die Datei in folgendem Format zurück:

Berechtigungen number_of_replicas Benutzer-ID Gruppendatei Dateigröße Änderungsdatum Änderungszeitpunkt Dateiname
Für ein Verzeichnis gibt es eine Liste seiner direkten Kinder wie in Unix zurück. Ein Verzeichnis ist aufgelistet als:
Berechtigungen Benutzer-ID Gruppenname modification_date modification_time dirname
Beispiel:

- `hdfs dfs -ls / Benutzer / Hadoop / Datei1`

Beendigungscode:

Gibt 0 bei Erfolg und -1 bei Fehler zurück.

lsr

Verwendung: `hdfs dfs -lsr <args>`

Rekursive Version von ls. Ähnlich wie Unix ls -R.

mkdir

Verwendung: `hdfs dfs -mkdir [-p] <Pfade>`

Nimmt Pfad-URLs als Argument und erstellt Verzeichnisse.

Optionen:

- Das Verhalten der Option -p ähnelt weitgehend Unix mkdir -p und erstellt übergeordnete Verzeichnisse entlang des Pfades.

Beispiel:

- `hdfs dfs -mkdir / Benutzer / Hadoop / Verzeichnis1 / Benutzer / Hadoop / Verzeichnis2`
- `hdfs dfs -mkdir hdfs: //nn1.example.com/user/hadoop/dir hdfs: //nn2.example.com/user/hadoop/dir`

Beendigungscode:

Gibt 0 bei Erfolg und -1 bei Fehler zurück.

moveFromLocal

Verwendung: `dfs-moveFromLocal <localsrc> <dst>`

Ähnlich dem put-Befehl, nur dass die Quelle localsrc nach dem Kopieren gelöscht wird.

moveToLocal

Verwendung: `hdfs dfs -moveToLocal [-crc] <src> <dst>`

Zeigt eine Meldung "Noch nicht implementiert" an.

mv

Verwendung: `hdfs dfs -mv URI [URI ...] <dest>`

Verschiebt Dateien von der Quelle zum Ziel. Dieser Befehl erlaubt auch mehrere Quellen. In diesem Fall muss das Ziel ein Verzeichnis sein. Das Verschieben von Dateien zwischen Dateisystemen ist nicht zulässig.

Beispiel:

- `hdfs dfs -mv / Benutzer / Hadoop / Datei1 / Benutzer / Hadoop / Datei2`
- `hdfs dfs -mv hdfs: //nn.beispiel.com/datei1 hdfs: //nn.beispiel.com/datei2 hdfs: //nn.beispiel.com/datei3 hdfs: //enn.beispiel.com/dir1`

Beendigungscode:

Gibt 0 bei Erfolg und -1 bei Fehler zurück.

put

Verwendung: `hdfs dfs -put <localrc> ... <dst>`

Kopieren Sie einzelne Quelldateien oder mehrere Quelldateien vom lokalen Dateisystem in das Zieldateisystem. Liest auch die Eingabe von stdin und schreibt in das Zieldateisystem.

- `hdfs dfs -put lokaleDatei / user / hadoop / hadoopfile`
- `hdfs dfs -put localfile1 lokaledatei2 / user / hadoop / hadoopdir`
- `hdfs dfs -put lokale Datei hdfs: //nn.example.com/hadoop/hadoopfile`
- `hdfs dfs -put - hdfs: //nn.example.com/hadoop/hadoopfile` Liest die Eingabe von stdin.

Beendigungscode:

Gibt 0 bei Erfolg und -1 bei Fehler zurück.

rm

Verwendung: `hdfs dfs -rm [-skipTrash] URI [URI ...]`

Löschen Sie Dateien, die als Argumente angegeben sind. Löscht nur nicht leere Verzeichnisse und Dateien. Wenn die Option `-skipTrash` angegeben ist, wird der Papierkorb, falls aktiviert, umgangen und die angegebenen Dateien werden sofort gelöscht. Dies kann nützlich sein, wenn Sie Dateien aus einem Verzeichnis mit Überlaufkontingenten löschen müssen. Lesen Sie `rmr` für rekursive Löschungen.

Beispiel:

- `hdfs dfs -rm hdfs: //nn.example.com/file/user/hadoop/leerdemir`

Beendigungscode:

Gibt 0 bei Erfolg und -1 bei Fehler zurück.

rmr

Verwendung: `hdfs dfs -rmr [-skipTrash] URI [URI ...]`

Rekursive Version von delete. Wenn die Option `-skipTrash` angegeben ist, wird der Papierkorb, falls aktiviert, umgangen und die angegebenen Dateien werden sofort gelöscht. Dies kann nützlich sein, wenn Sie Dateien aus einem Verzeichnis mit Überlaufkontingenten löschen müssen.

Beispiel:

- `hdfs dfs -rmr / Benutzer / hadoop / dir`
- `hdfs dfs -rmr hdfs: //nn.example.com/user/hadoop/dir`

Beendigungscode:

Gibt 0 bei Erfolg und -1 bei Fehler zurück.

setfacl

Syntax: `hdfs dfs setfacl [R] [<ac:structured macro ac:name="anchor" ac:echoma version="1" ac:macro id="faa6e0d0-ee29-4620-b5ee-7203a0d23a0d"><ac:parameter ac:name="">a b k</ac:parameter></ac:structured macro> b | k <ac:structured macro ac:name="anchor" ac:echoma version="1" ac:macro id="a3f345a2-5ae5-478d-b4d6-a99a0b128d"><ac:parameter ac:name="">a m x_acl_spec</ac:parameter></ac:structured macro> m | x <acl_spec> <Pfad>] [set <acl_spec> <Pfad>]`

Legt Zugriffssteuerungslisten (ACLs) für Dateien und Verzeichnisse fest.

Optionen:

- `-b`: Entferne alle außer den Basis-ACL-Einträgen. Die Einträge für Benutzer, Gruppe und andere werden aus Gründen der Kompatibilität mit Berechtigungsbits beibehalten.
- `-k`: Entfernen Sie die Standard-ACL.
- `-R`: Anwenden von Operationen auf alle Dateien und Verzeichnisse rekursiv.
- `-m`: Ändern Sie die ACL. Neue Einträge werden der ACL hinzugefügt und vorhandene Einträge bleiben erhalten.
- `-x`: Entfernt die angegebenen ACL-Einträge. Andere ACL-Einträge bleiben erhalten.
- `--set`: Ersetzen Sie die ACL vollständig, indem Sie alle vorhandenen Einträge verwerfen. `acl_spec` muss Einträge für Benutzer, Gruppe und andere enthalten, um Kompatibilität mit Berechtigungsbits zu gewährleisten.
- `acl_spec` : Durch Kommas getrennte Liste von ACL-Einträgen.
- `Pfad`: Zu ändernde Datei oder Verzeichnis.

Beispiele:

- ~~hdfs dfs -setfacl -m Benutzer: hadoop: rw / Datei~~
- hdfs dfs -setfacl -x Benutzer: Hadoop / Datei
- hdfs dfs -setfacl -b / Datei
- hdfs dfs -setfacl -k / dir
- ~~hdfs dfs -setfacl -set Benutzer: rw, Benutzer: hadoop: rw-, Gruppe :: r, andere :: r / Datei~~
- hdfs dfs -setfacl -R -m Benutzer: hadoop: rx / dir
- hdfs dfs -setfacl -m default: benutzer: hadoop: rx / dir

Beendigungscode:

Gibt 0 bei Erfolg und nicht Null bei Fehler zurück.

setrep

Verwendung: hdfs dfs -setrep [-R] [-w] <numReplicas> <Pfad>

Ändert den Replikationsfaktor einer Datei. Wenn *Pfad* ein Verzeichnis ist, ändert der Befehl rekursiv den Replikationsfaktor aller Dateien unter dem Verzeichnisbaum, der auf *Pfad* basiert.

Optionen:

- Das Flag -w fordert an, dass der Befehl auf den Abschluss der Replikation wartet. Dies kann möglicherweise sehr lange dauern.
- Das -R-Flag wird aus Gründen der Abwärtskompatibilität akzeptiert. Es hat keine Wirkung.

Beispiel:

- hdfs dfs -setrep -w 3 / Benutzer / hadoop / dir1

Beendigungscode:

Gibt 0 bei Erfolg und -1 bei Fehler zurück.

stat

Verwendung: hdfs dfs -stat URI [URI ...]

Gibt die Statistikinformationen zum Pfad zurück.

Beispiel:

- hdfs dfs -stat Pfad

Exit Code: Gibt 0 bei Erfolg und -1 bei Fehler zurück.

tail

Verwendung: hdfs dfs -tail [-f] URI

Zeigt das letzte Kilobyte der Datei als Standard an.

Optionen:

- Die Option -f gibt angehängte Daten aus, wenn die Datei wächst, wie in Unix.

Beispiel:

- hdfs dfs -tail Pfadname

Exit Code: Gibt 0 bei Erfolg und -1 bei Fehler zurück.

test

Verwendung: hdfs dfs -test - [ezd] URI

Optionen:

- Die Option -e prüft, ob die Datei existiert und gibt 0 zurück, wenn sie wahr ist.
- Die Option -z prüft, ob die Datei die Länge 0 hat und gibt 0 zurück, wenn sie wahr ist.
- Die Option -d überprüft, ob es sich bei dem Pfad um ein Verzeichnis handelt, und gibt 0 zurück, wenn es wahr ist.

Beispiel:

- `hdfs dfs -test -e Dateiname`

text

Verwendung: `hdfs dfs -text <src>`

Nimmt eine Quelldatei und gibt die Datei im Textformat aus. Die zulässigen Formate sind zip und TextRecordInputStream.

touchz

Verwendung: `hdfs dfs -touchz URI [URI ...]`

Erstellen Sie eine Datei mit der Länge null.

Beispiel:


- `hadoop -touchz Pfadname`

Exit Code: Gibt 0 bei Erfolg und -1 bei Fehler zurück.

HUE - HIVE Befehle

Allgemein

Hive Built In Functions: <http://hadooptutorial.info/hive-built-in-functions/#tc-comment-title>

- Aggregate Functions (count , avg(col), sum(DISTINCT col))
- Date Functions (Date data types do not exist in Hive. In fact the dates are treated as strings in Hive.)
- Collection Functions (size(Array), map_keys(Map)
- Conditional Functions (COALESCE, Case)
- Mathematical Functions (round, pi)
- String Functions (concat, get_json_object)
- Table Generating Functions (explode(ARRAY), json_tuple(jsonStr, k1, k2, ...))
- Type Conversion Function (binary(string|binary), cast(expr as))

Hive Date Functions: http://hadooptutorial.info/hive-date-functions/#DATE_SUB_string_date_int_days

- from_unixtime
- from_utc_timestamp
- to_utc_timestamp
- unix_timestamp
- unix_timestamp()
- unix_timestamp(string date)
- unix_timestamp(string date, string pattern)
- from_unixtime(bigint number_of_seconds [, string format])
- To_Date(string timestamp)
- MONTH(string date)
- DAY(string date), DAYOFMONTH(date)
- HOUR(string date)
- MINUTE(string date)
- [SECOND\(string date \)](#)
- WEEKOFYEAR(string date)
- DATEDIFF(string date1, string date2)
- DATE_ADD(string date, int days)
- DATE_SUB(string date, int days)
- DATE CONVERSIONS

Hive String Functions: <http://hadooptutorial.info/string-functions-in-hive/#tc-comment-title>

- Creating Table in HIVE
- String Functions and Normal Queries
- ASCII
- CONCAT
- CONCAT_WS
- FIND_IN_SET
- LENGTH
- LOWER or LCASE
- UPPER or UCASE
- LPAD
- RPAD
- TRIM
- REPEAT and REVERSE
- SPACE
- SPLIT
- SubString
- Format
- INSTR
- Locate
- N-Grams
- Parse URL
- Printf
- Regexp_Extract
- Regexp_Repalce
- Sentences
- Str_to_map
- Translate

Hive Aggregate Functions: <http://hadooptutorial.info/hive-aggregate-functions/#tc-comment-title>

- Average (avg)
- Collections (collect_set, collect_list)
- Correlation (corr)
- Count
- Covariance (covar_pop, covar_samp)
- Histogram (histogram_numeric)
- Maximum (max)
- Minimum (min)
- NTile
- Percentile
- Standard Deviation (stddev_pop, stddev_samp)
- Sum
- Variance

Tealium JSON-Dateien verarbeiten

Es wird täglich folgende Datei eingelesen: [config-YYYY-MM-DD.json](#)

```
1  {
2    "account": "axelspringer",
3    "archivedFilteredStreams": [{
115  "audiences": {
153    "bulk augmentation definitions": [],
154    "cooldownGroups": {
165    "dataSources": [{
176    "display": {
503    "environments": [{
513    "eventDefinitions": [],
514    "jobs": [],
515    "labels": [{
635      "maxQuantifierId": 5644,
636      "maxRuleId": 5207,
637      "maxServiceId": 5001,
638      "maxStreamId": 107,
639      "maxTransformationId": 5240,
640    "playbook": {
649    "previousVersionInfo": {
657    "profile": "dip-main",
658    "publicSettings": {
711    "quantifiers": [{
11330  "rules": [{
14007  "services": [{
14064  "settings": {
14093  "transformations": [{
19045  "utui profile revisions": {
19048  "version info": {
19056  }
```

Folgender Inhalt ist zurzeit relevant:

```
(
account string,
profile string,
labels array<struct< id:string, name:string, color:string, createdByPlay:array<string> >>,
quantifiers array<struct< id:int, name:string, description:string, type:string, context:string, eventKey:string, hidden:string, editable:string,
preloaded:string, eventDBEnabled:string, audienceDBEnabled:string, refers:int, transformationIds:array<int>, config:string, labelIds:array<string>,
utuiDuplicates:array<string>, createdByPlay:array<string>, duplicateAttributes:array<string> >>,
rules array<struct< id:int, name:string, description:string, logic:string, hidden:string, editable:string, preloaded:string, createdByPlay:array<string>,
labelIds:array<string> >>,
transformations array<struct< id:int, name:string, description:string, action:string, trigger:string, type:int, hidden:string, editable:string, preloaded:
string, rules:array<int>, actionData:struct< quantifierId:int, incrementValue:string>, orderIndex:int, createdByPlay:array<string> >>,
version_info struct< version:string, revision:string, description:string, lastModifiedBy:string, published:string, hasBeenPublished:string>,
settings struct< accountEnabled:string, visitorRetentionDays:int, preloadedVisitorRetentionDays:int, stitchingEnabled:string, eventStoreEnabled:
string, eventDBEnabled:string, audienceStoreEnabled:string, audienceDBEnabled:string, audienceDBVersion:int, dataAccessDBExpirationDays:
int, collectClientIp:string, frequencyCapActionCooldownHours:int, eventStreamEnabled:string, legacyBulkDownloadEnabled:string,
visitorQueryMaxCountPerDay:int, region:string, eventStoreBucket:string, dataAccessClusters:string, useLegacyEventBucket:string,
useLegacyDataAccessCluster:string, ruleDependencyEnabled:string, visitorEventStreamDefinition:string>
)
```

- ein quantifiers kann mehrere transformationen [5024,5024] besitzen
- ein quantifiers kann mehrere labels [„e3906abd-df77-4e8b-a5d3-f212a72ef1e8“, „9723e8dc-65d0-4342-a3e5-ba3ed9fa1ca1“] besitzen
- eine transformations kann eine quantifierId [„quantifierid“:5200] im Feld „actiondata“ besitzen
- eine transformations kann mehrere rules [5031,5032] besitzen
- eine rules kann eine labelid [„009bda08-0e5f-4484-d094-a5861356ffaa“] haben

Array<string> auswerten

Beispiel: Auszug aus JSON-Datei

```
"<span style="color: #00b0f0">labels</span>": <span style="color: #ff0000">[</span>{
  "id": "e72b4604-bec0-4b71-8cf7-768a6f61415e",
  "name": "unit_bild",
  "color": "blue",
  "createdByPlay": []
},
{
```

- "labels": Name des JSON-Elements
- **[]** Start – Ende des JSON-Elements
- **{ }** Array Start – Ende

MAP<string Strukt<name:string, ... >> auswerten

```
Gespeichertes Format in der Tabelle: audiences map<span style="color: #ff0000"><</span>string,struct<span style="color: #ff0000"><</span>name:string, logic:string, perspective:string, visitorRetentionDays:int, createdByPlay:array<string>, editable:string, labelIds:array<string><span style="color: #ff0000">>></span>
<span style="color: #00b0f0">audiences</span>": <span style="color: #ff0000">{</span>
"<span style="color: #806000">axelspringer_dip-main_101</span></span>": <span style="color: #00b050">
```

"axelspringer_dip-main_107":

```
SELECT <span style="color: #806000">audience_key</span>,<br>
<span style="color: #7030a0">audience_val</span>.name,<br>
<span style="color: #7030a0">audience_val</span>.logic,<br>
<span style="color: #7030a0">audience_val</span>.perspective,<br>
<span style="color: #7030a0">audience_val</span>.visitorRetentionDays,<br>
<span style="color: #7030a0">audience_val</span>.createdByPlay,<br>
<span style="color: #7030a0">audience_val</span>.editable,<br>
<span style="color: #7030a0">audience_val</span>.labelIds<br>
FROM dip_global_stg_sven.stg_tl_metadata_json_ext<br>
LATERAL VIEW <span style="color: #ff0000">explode</span>(<span style="color: #00b0f0">audiences</span>) exploded_audience as <span<br>
style="color: #806000">audience_key</span>,<span style="color: #7030a0">audience_val</span>;<br>
Ergebnis:
```

audience_key	name	logic	perspective	visitorRetention Days	createdBy Play	editable	labelIds	
axelspringer_ dip-main_101	Include in Audience Store	{"\$or": <ac:structured- macro ac:name=" unmigrated-wiki- markup" ac: schema-	badges	0	[]	true	[]]]></ac:plain-text- body></ac: structured- macro>

		<pre> version="1" ac: macro-id=" 1cb572df-64f1- 4dd9-8843- 94c44eae7ec1" ><ac:plain-text- body><![CDATA [[{"\$and": }]]></ac:plain-text- body></ac: structured- macro> <ac:structured- macro ac:name=" unmigrated-wiki- markup" ac: schema- version="1" ac: macro-id=" 8ad1520b-978c- 492f-825a- b563ec946741" ><ac:plain-text- body><![CDATA [[{"baseOperator" :null, }]]></ac:plain-text- body></ac: structured- macro> "operator": "greater_than_or_ equal_to", "operand1": "metrics.22", <ac:structured- macro ac:name=" unmigrated-wiki- markup" ac: schema- version="1" ac: macro-id=" cd6afd52-a611- 4415-a94b- 1ae838c53f5a" ><ac:plain-text- body><![CDATA ["operand2":2]]]] </pre>						
<p>Axelspringer</p> <pre> <ac:structured- macro ac:name=" unmigrated-wiki- markup" ac: schema- version="1" ac: macro-id=" 04decc5f-19ce- 484b-b08c- 170d62fff984" ><ac:plain-text- body><![CDATA[_dip-main_107 </pre>	Online Purchase affinity	<pre> {"\$or":[{"\$and": [{"baseOperator": null,"operator": "exists", ... </pre>	badges	0	[]	true	[]]]></ac:plain-text-body></ac:structured-macro>



Datenbank - „Location" anpassen (<http://gaganonthenet.com/2015/02/23/hive-change-location-for-database-or-schema/>)

Beispiel:

- auf den Server / Konsole per ssh - pasit-hadpm02v.linux.asinfra.net einloggen
- das Verzeichnis „dip_global_stg_event.db" muss im Verzeichnis „/data/dip/global/warehouse/" enthalten sein – wenn nicht, dann muss das Verzeichnis angelegt werden
- auf die MySql-Datenbank (metastore) einloggen (mysql metastore -u hive -p) – Passwort-Safe unter <https://passwords.asv.local/>
- SELECT * FROM DBS WHERE NAME = 'dip_global_stg_event';
- UPDATE **DBS** SET DB_LOCATION_URI = '/data/dip/global/warehouse/dip_global_stg_event.db' WHERE "NAME" = 'dip_global_stg_event';

External-Tabelle-Partition erzeugen

Es gibt zwei Möglichkeiten eine Partition zu einer externen Tabelle hinzuzufügen:

- **MSCK REPAIR TABLE** dip_global_stg_sven.stg_tl_metadata_json_ext; das "MSCK REPAIR TABLE" aktualisiert automatisch alle Partitionen – Verzeichnisse(z.B.: **file_date**=2018-01-14) die in einem bestimmten Verzeichnis(**Location**) hinterlegt sind, werden dazu herangezogen. Dafür muss der PartitionName(PARTITIONED BY **file_date**) gleich dem Namen des Verzeichnisses (**file_date**=...) entsprechen. Verzeichnisstruktur:- Location: /data/dip/global/rawdata/tealium/metadata_daily
 - /file_date=2018-01-14
 - /file_date=2018-01-15
 - /file_date=2018-01-16
 - /file_date=2018-01-17
 - ...

Vorteil: wenn beim Einlesen mal etwas schiefgegangen sein sollte, braucht man sich nicht um ggf. nicht zugewiesene Partitionen kümmern – MSCK geht jedes mal das Verzeichnis(**Location**) komplett durch.

- **ALTER TABLE** dip_global_stg_sven.stg_tl_metadata_json_ext ADD IF NOT EXISTS PARTITION (**file_date** = \${DATE}); Falls die Methode mit „MSCK REPAIR TABLE“ Performance kostet, kann man auch Partitionen mit ALTER TABLE erzeugen

Die Tabelle wird folgendermaßen angelegt:

```
CREATE EXTERNAL TABLE dip_global_stg_sven.stg_tl_metadata_json_ext
(
  account string,
  profile string,
  audiences map<string,struct<
    name:string,
    logic:string,
    perspective:string,
    visitorRetentionDays:int,
    createdByPlay:array<string>,
    editable:string,
    labelIds:array<string>
  >>,
  labels array<struct<
    id:string,
    name:string,
    color:string,
    createdByPlay:array<string>
  >>
)
PARTITIONED BY (`file_date` date)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 'hdfs://nameservice1/data/dip/global/rawdata/tealium/metadata_daily';
```

Tabelle-Partition erzeugen

Um bei einer z.B.: Core-Tabelle auf die Satelliten Tabelle eine Partition zu legen, kann das u.a. mit

- Wenn Sie versuchen, den obigen Code auszuführen, besteht eine gute Chance, dass Sie aufgrund der von Ihnen festgelegten Eigenschaften einen Fehler erhalten. Erstens wird es nicht funktionieren, wenn Sie die dynamische Partitionierung deaktiviert haben, also stellen Sie sicher: **SET hive.exec.dynamic.partition=true;**
- Dann treffen Sie möglicherweise auf einen Fehler, wenn Sie nicht mindestens eine statische Partition vor den dynamischen Partitionen partitionieren. Diese Einschränkung würde Sie davor bewahren, versehentlich eine Root-Partition zu entfernen, wenn Sie ihre Unterpartitionen mit dynamischen Partitionen überschreiben wollten. Meiner Erfahrung nach war dieses Verhalten nie hilfreich und war oft ärgerlich, aber Ihre Laufleistung kann variieren. Auf jeden Fall ist es leicht zu ändern: ggf. **SET hive.exec.dynamic.partition.mode=nonstrict;** und
- **INSERT OVERWRITE TABLE** dip_global_co_sven.co_tl_metadata_labels_s PARTITION (**DWH_EFFECTIVE_DATE**)...

Die Tabelle wird folgendermaßen angelegt:

```
CREATE TABLE dip_global_co_sven.co_tl_metadata_labels_s
(
```

```
DWH_ID string,  
DWH_CR_LOAD_ID int,  
DWH_CR_JOB_ID int,  
DWH_LOAD_DATE string,  
DWH_LOAD_FILENAME string,  
DWH_LABELS_ID_HUB string,  
DWH_ACCOUNT_BK string,  
DWH_PROFILE_BK string,  
LABEL_ID string,  
LABEL_NAME string,  
LABEL_COLOR string,  
LABEL_CREATEDBYPLAY ARRAY<string>  
)  
PARTITIONED BY (DWH_EFFECTIVE_DATE string);
```

Workflow hängt ohne ersichtlichen Grund

Obwohl alles korrekt aussieht, hat man den Eindruck, der gestartete Workflow hat sich aufgehängt.
Es kein Fehler gemeldet.

- Ein Grund kann sein, dass die Tabellen, die gefüllt werden, **gelockt** sind mit „**show locks**“ werden dann alle gelockten Tabellen aufgelistet- unlock table Tabellennamen; - unlock table Tabellennamen partition(file_date='2018-01-18');