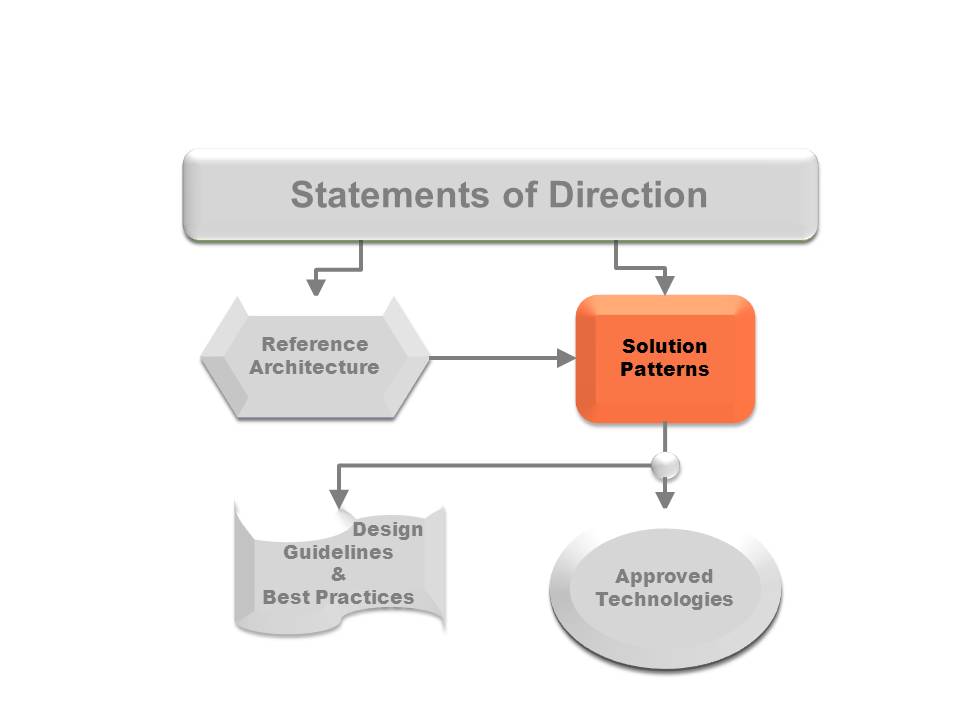


Solution Pattern – Approved Preferred

Universal Data Loader(UDL)

Revision 1.1

06/04/2019



# Table of Contents

[Table of Contents 2](#_Toc10649465)

[1.0 Overview 3](#_Toc10649466)

[2.0 Reference Architecture 4](#_Toc10649467)

[3.0 Solution Pattern Diagram 5](#_Toc10649468)

[4.0 Description of Solution Pattern Components 8](#_Toc10649469)

[5.0 Technology and Product listing 17](#_Toc10649470)

[6.0 Related Documents and Best Practices 18](#_Toc10649471)

[7.0 Glossary 19](#_Toc10649472)

[Revision History and Contributors 20](#_Toc10649473)

[Meta Tags 21](#_Toc10649474)

# Overview

This solution pattern serves to outline the Advanced Analytics (AA) Universal Data Loader (UDL), an event driven AWS architecture designed for moving data from a source(specific S3 Buckets) to a target as source data is continuously being submitted to UDL.

As part of Advanced Analytics migration to Cloud, several Cloud based analytical consumption data stores have been identified that are aligned to specific consumption use cases. First use case is Snowflake MPP Database as a replacement for on Premise Teradata DB

Data is delivered to S3 by upstream data providers (e.g., Ab Initio) in the order it is intended to be loaded into target system. S3 folder structure (prefixes) organized by Line of Business, Product, Sub-Product, Table, Partition Date. Avro files land in the Partition Date directory, split into multiple parts with similar filenames and a sequence number

This solution pattern addresses following topics

* Reusable framework for source teams to load data into analytical targets
* How UDL platform enables secure data migration from S3 to multiple cloud targets
* Self-service, Logging and Reporting of UDL.
* Event based orchestration to support load at different latencies – Batch, Micro-batch

This architecture runs on each DFS account individually. As AA’s account strategy is built out, the UDL solution will be deployed via Terraform/Ansible/Chef to each new account.

# Reference Architecture

This solution is a part of the Advanced Analytics Reference Architecture (TBD)

# Solution Pattern Diagram



Figure – UDL High Level Flow



Figure 2 UDL Architecture

**Life Cycle of UDL Event**

An S3 event is generated upon file landing into etl buckets. UDL Service actively listens for all S3 ObjectCreated and ObjectRemoved events. Below are the steps performed for each event generated

* The associated file has to be a .avro or a .manifest file
* The file has to belong to a table that has been on-boarded to UDL
* The file has to have metadata indicating its load-type unless it's a manifest file then metadata is not required
* If the file is part of a manifest file it should have metadata of 'transactional' set to true indicating that this file will be part of a manifest file in which it will be accounted for as part of the manifest
* In case the table is not setup in UDL, the event is discarded and never processed by UDL
* In case the table is setup in UDL and the file adheres to the requirements as mentioned
  + ObjectCreated Event:
    - Ensures that the file doesn't already exist in DynamoDB in case the event already made it prior into UDL and places the message payload into the DynamoDB events table
  + ObjectRemoved Event:
    - Finds the corresponding event in Events Table (DynamoDB). Updates the event as "Stale" if the event hasn't been processed yet by UDL. This in turn cancels the job in UDL for this file and UDL will know to skip it unless it has already loaded it
* If the "Target" status is set to red in the lookup table, the traffic controller is instructed to not route the event to that target's queue and discard it, meaning that the event will not be queued and the corresponding record in the events table failed
* If a file is a .avro file it is routed to the Snowflake FIFO queue whereas if it is a .manifest file it is routed to the pre-validation manifest queue
* Non-Manifest/Avro Route(Single/Small Files):
  + Snowflake Loader(Lambda) reads of the FIFO Queue and generates SQL Statements based on the load-type(Insert, Update, Delete, Replace All)
  + Runs the SQL statements against snowflake and commits upon success
  + Upon Failure, Table status is marked red in lookup table and all events for table are routed to failed events table. Also event status is marked as ‘Failed’ in the events table
* Manifest Route(Transaction/Large Files):
  + Manifest Validator verifies the JSON for valid workload entries and metadata
  + If Valid, manifest files are routed to post manifest FIFO
  + Snowflake Loader(EC2) reads of the FIFO queue and generates SQL transactions for each workload
  + Each transaction is executed sequentially against snowflake and any failure will rollback all the manifest workloads
  + Upon Failure, Table status is marked red in lookup table and all events for table are routed to failed events table. Also event status is marked as ‘Failed’ in the events table

# Description of Solution Pattern Components

**Description of UDL Components**

* S3 - Location where the files will be landed. The files are landed into s3 will be Avro format and will have metadata attached. A S3 event will be configured which would trigger a call to the Record Loader Lambda function.
* Record Loader Lambda - This lambda function is responsible for taking the S3 event and loading the message into DynamoDB and the Tx Events Table based on the type of file that gets landed in S3
* DynamoDB (Tx Events Table) - This is a DynamoDB table that will hold the file information for the files that get landed in s3. This is mainly used for UDL team's validation of events occurring in S3 without having to query S3 itself
* DynamoDB (Events Table) - This is a DynamoDB table that will hold the information of all the files that are to be processed by UDL
* DynamoDB Streams (Events Table) - This is a DynamoDB Stream that will maintain the order in which records get inserted into DynamoDB events table and pass those messages to the Traffic Controller Lambda Function.
* Traffic Controller Lambda - This lambda function is responsible for reading messages off the stream and routing them to the appropriate path for the message/record/file to be processed.
* DynamoDB (Target Route Lookup Table) - DynamoDB table used by the Traffic Controller Lambda function to instruct on which queues to route the message to.
* Pre\_Validation\_Manifest.fifo - Fifo queue to hold the message containing the information about the manifest file before being validated. Traffic controller will send those messages here based on the type of file received.
* Manifest Validator - Lambda function that will validate a manifest file. Manifest is  json file that includes aggregated information about multiple .avro files for UDL
* Post\_Validation\_Manifest.fifo - Fifo queue to hold the validated manifest messages. Messages will be landed here by the manifest validator.
* EC2 (Behind Load Balancer) - This will hold the python application which will read messages from the post\_validation\_manifest.fifo queue to get the list of manifest files and based on the instructions in the message it will call the appropriate classes to load files from manifest in the target locations.
* SQS.fifo - Fifo queue to hold the messages to be processed synchronously. Only for insert/upsert/delete file types.
* Target Loader Lambda (Triggered By SNS event) - This lambda function takes the message of the SQS.fifo queue and loads the file into the target system.
* Error Info Queue - SQS queue that will accept error and info messages from all lambda's and ec2 that run python code.
* Error\_Info Queue Processing Lambda - Lambda function that takes messages of the Error Info Queue and send them to the enterprise logging system.
* DynamoDB Streams (Target route lookup table) - This is a DynamoDB Stream that will send records to lambda as and when they get updated which would send them to elastic search.
* Elastic Loader Lambda - This lambda function takes messages from DynamoDB streams and send them to elastic search.
* Elastic Search Service - This is a service with would hold copies of the DynamoDB tables so we can report against them in real-time.
* DynamoDB (Failed Events Table) - This table holds messages that get taken off the fifo queues if those tables/groups are in a red state.

### Events Table Content

### Overview

DynamoDB stores docs, where each record corresponding for a file event is a dict representing all the information needed to load and update an event as it's being processed by UDL. Below is a sample of what a doc looks like in the events table. Most of the doc is the even payload itself from S3, from which UDL's event loader creates initial fields to add to the doc and place in DynamoDB. In this section, we will go through fields that are actively leveraged whether for loading or reporting out of UDL

{ "bucketName": "dfs-analytics-dlt-use1-ent-etl", "complete-time": "2019-05-16T23:10:52.861Z", "createdDate": "2019-05-16T22:19:50.087Z", "dataAssetName": "dirbnk/card/udl\_test\_table", "event": { "Records": [ { "awsRegion": "us-east-1", "eventName": "ObjectCreated:Put", "eventSource": "aws:s3", "eventTime": "2019-05-16T22:19:47.885Z", "eventVersion": "2.1", "requestParameters": { "sourceIPAddress": "10.52.31.47" }, "responseElements": { "x-amz-id-2": "1OgKOZCzmauqHUqDHoEPtZ65KIuvDldzZuCBe8I8EW01ETqKUx51GrHivdHYlRhSlc6wHLps7P8=", "x-amz-request-id": "12E359F5B3FBEC01" }, "s3": { "bucket": { "arn": "arn:aws:s3:::dfs-analytics-dlt-use1-ent-etl", "name": "dfs-analytics-dlt-use1-ent-etl", "ownerIdentity": { "principalId": "A2HREGGS08DHHO" }, "size": "11378" }, "configurationId": "udl-replay-event", "object": { "eTag": ""b5c55a7de247e55e360ddeaf4aeb9b4d"", "key": "dirbnk/card/udl\_test\_table/src\_dt\_pc=20190301T000000Z/xml\_type\_perfect\_metadata\_sample4.avro", "sequencer": "005C110D39EE57CF44", "size": null, "versionId": "JoXsioM4gjptOBgASRZIp\_qoZf6K20ry" }, "s3SchemaVersion": "1.0" }, "userIdentity": { "principalId": "AWS:AIDAJPQ76ZPXSWNCLQMSQ" } } ] }, "eventSource": "udl-replay-event", "eventTime": "2019-05-16T22:19:47.885Z\_475847aa-004e-452a-b568-85bbdaf21d83", "extended\_status": { "2019-05-16T23:10:41.381Z": "Locked file for Processing", "2019-05-16T23:10:52.861Z": "status changed to Success" }, "file\_uuid": "dirbnk/card/udl\_test\_table", "lastUpdated": "2019-05-16T23:10:52.861Z", "lock-time": "2019-05-16T23:10:41.284Z", "metadata": { "load-type": "Insert", "record-count": "2", "source": "Test.Manual", "source-type": "Table" }, "objectKey": "dirbnk/card/udl\_test\_table/src\_dt\_pc=20190301T000000Z/xml\_type\_perfect\_metadata\_sample4.avro", "schemaValidated": "N", "status": "Success", "tableName": "udl\_test\_table", "tags": { "org-code": "Not Specified" }, "uuid": "475847aa-004e-452a-b568-85bbdaf21d83" }

### Events Table Fields description

| **Field Name** | **Description** | **Sample Value** |
| --- | --- | --- |
| bucketName | The bucket where the file originated from | "dfs-analytics-dlt-use1-ent-etl" |
| createdDate | The time the event was created in DynamoDB Events table. If the event wasn't replayed manually this should align with the time the file was created in S3 | "2019-05-16T22:19:50.087Z" |
| dataAssetName | The table level prefix to which the event/file belongs to | "dirbnk/card/udl\_test\_table" |
| eventSource | Indicates whether an event was placed in Dynamo manually via replay utility or came from S3 | "udl-replay-event" |
| eventTime | A combination of S3 event time and file\_uuid. Used as the sort key for the dynamo table to sort time ordering | "2019-05-16T22:19:47.885Z\_475847aa-004e-452a-b568-85bbdaf21d83" |
| extended\_status | Contains a history (Stored as dict) of high level logs of UDL manipulating/loading the event | "{"2019-05-16T23:10:41.381Z": "Locked file for Processing","2019-05-16T23:10:52.861Z": "status changed to Success }" |
| file\_uuid | The table level prefix to which the event/file belongs to. Used as the partition key of this Dynamo table | "dirbnk/card/udl\_test\_table" |
| lastUpdated | Last time this event record was updated by UDL (Usually when marking event as success/failed) | "2019-05-16T23:10:52.861Z" |
| lock-time | Last time UDL locked this event to start loading corresponding file into target | "2019-05-16T23:10:41.284Z" |
| metadata | Metadata of file needed by UDL to load into target. Most important metadata is load-type | "{"load-type": "Insert","record-count": "2", "source": "Test.Manual", "source-type": "Table"}" |
| objectKey | The S3 object key to be loaded | "dirbnk/card/udl\_test\_table/src\_dt\_pc=20190301T000000Z/xml\_type\_perfect\_metadata\_sample4.avro" |
| schemaValidated | Feature not currently supported. Intended for schema validation | "N" |
| status | Status of load into target corresponding to event. Can be "Ready", "Processing", "Failed" or "Stale" | "Success" |
| tableName | table name without prefix | "udl\_test\_table" |
| tags | S3 tags on object | {"org-code": "Not Specified"} |
| uuid | UDL generated uuid for each object key (Using python generic uuid generator) | "475847aa-004e-452a-b568-85bbdaf21d83" |

### DynamoDB Lookup Table Content

### Overview

The Lookup Table DybamoDB is essentially the list of table prefixes found in S3 that should be leveraging UDL for continuous data loading (i.e: "dirbnk/card/acaps\_10") with each prefix record containing information such as where the table should be loaded downstream (i.e Snowflake, Redshift etc..), whether the table is currently paused or not, which queue(s) in the UDL architecture are setup for this table for events to be routed to (And hence loaded for the targets).

A record in the lookup table looks like the below:

{ "avro\_file\_schema\_hash": "noschemahash5", "data\_asset\_name": "dirbnk/card/udl\_test\_table", "target\_arns": { "manifest": { "snowflake": { "group\_status": "green", "lambda\_arn": "arn:aws:lambda:us-east-1:500869459081:function:data-lambda-analytics-udl-manifest-validator", "lastUpdated": "2019-05-01T16:01:38.050Z", "message": "error with dirbnk/card/udl\_test\_table has turned this group red", "queue\_arn": "arn:aws:sqs:us-east-1:500869459081:data-udl-analytics-post-validation-manifest.fifo" } }, "non\_manifest": { "snowflake": { "group\_status": "green", "lambda\_arn": "arn:aws:lambda:us-east-1:500869459081:function:data-lambda-analytics-udl-snowflake-target-loader", "lastUpdated": "2019-05-18T18:57:44.411Z", "message": "Updated by Lookup Controller", "queue\_arn": "arn:aws:sqs:us-east-1:500869459081:data-udl-analytics-snowflake-target.fifo" } } }, "targets": { "manifest": { "snowflake": { "status": "green", "subscribers": { "error": "No concatact Info", "info": "No Contact Info" } } }, "non\_manifest": { "snowflake": { "status": "green", "subscribers": { "error": "No concatact Info", "info": "No Contact Info" } } } }, "UDL\_Active": "yes", "version": 1 }

### Lookup Table Fields description

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| data\_asset\_name | The table level prefix to which the event/file belongs to. Used as the partition key of this Dynamo table | "dirbnk/card/udl\_test\_table" |
| avro\_file\_schema\_hash | Currently not supported: The schema of the avro file in hash format | "xhwydanfoada" |
| target\_queue\_arns.non\_manifest.{target}.queue\_arn | Queue Name for non-manifest files | arn:aws:sqs:us-east-1:500869459081:data-udl-analytics-snowflake-target.fifo |
| target\_queue\_arns.non\_manifest.{target}.lambda\_arn | Lambda Arn for Non-Manifes route | arn:aws:lambda:us-east-1:500869459081:function:data-lambda-analytics-udl-snowflake-target-loader |
| targets.manifest.{target}.subscribers.info | Info Message Subscribers |  |
| targets.manifest.{target}.subscribers.info | Error Message Subscribers |  |
| target\_queue\_arns.manifest.{target}.queue\_arn | Queue Name for Manifest Post Validation | arn:aws:sqs:us-east-1:500869459081:data-udl-analytics-post-validation-manifest.fifo |
| target\_arns | Payload containingAWS Arns of various services for each target | "{ "manifest" : { "M" : { "snowflake" : { "M" : { "group\_status" : { "S" : "green" }, "lambda\_arn" : { "S" : "arn:aws:lambda:us-east-1:500869459081:function:data-lambda-analytics-udl-manifest-validator" }, "lastUpdated" : { "S" : "2019-05-01T16:01:38.050Z" }, "message" : { "S" : "error with dirbnk/card/udl\_test\_table has turned this group red" }, "queue\_arn" : { "S" : "arn:aws:sqs:us-east-1:500869459081:data-udl-analytics-post-validation-manifest.fifo" } } } } }, "non\_manifest" : { "M" : { "snowflake" : { "M" : { "group\_status" : { "S" : "green" }, "lambda\_arn" : { "S" : "arn:aws:lambda:us-east-1:500869459081:function:data-lambda-analytics-udl-snowflake-target-loader" }, "lastUpdated" : { "S" : "2019-05-18T18:57:44.411Z" }, "message" : { "S" : "Updated by Lookup Controller" }, "queue\_arn" : { "S" : "arn:aws:sqs:us-east-1:500869459081:data-udl-analytics-snowflake-target.fifo" } } } } } }" |
| UDL\_active | Value designating if table prefix is activated in UDL or not (i.e capturing events from S3 prefix or not) | "yes" |

#### Manifest Payload

Manifest is  json file that includes aggregated information about multiple .avro files for UDL. Below is the Json Schema for the Manifest file

<https://github.discoverfinancial.com/aap-cde-org/udl/wiki/UDL-Manifest-Schema>

#### Elastic Search(Kibana)

UDL Event history will be continuously published to Elastic Search and made available to users via Kibana dashboards. Kibana dashboard provides information of target load status to the users.

**UDL Framework Configuration**

* Data Security
  + Data At-Rest
    - DynamoDB tables are encrypted with a customer managed key in Advanced Analytics’ KMS.
      * Keys are rotated as per enterprise security standards
  + Data At-Rest (data in Snowflake Storage layer, S3)
    - Tri-Secret Secure (Figure 2) with a customer managed key in Advanced Analytics’ KMS (AES 256)
      * Keys are rotated as per enterprise security standards
  + Data In-Transit (TLS 1.2)
    - AWS HTTPS calls are all transmitted via TLS
      * In-transit calls contain all of the following
        + S3 get calls
        + DynamoDB read and write calls
        + Lambda execution calls
        + SNS and SQS publishing
* Data Retention
  + - UDL meta store data does not have an active data retention period; kept forever. Instead of deleting data, data is marked as Stale and viewed as inactive.
  + RPO/RTO
    - DynamoDB maintains continuous backups of your table for the last 30 days
    - DynamoDB tables require rebuild in order to recover from a backup; RTO has been observed at <10min
* Network Security
  + Direct Connect
    - Our established Enterprise communication pattern to AWS requires leveraging the Direct Connect connection from the DFS network to AWS.
  + VPC Endpoints
    - To restrict communications to AA account VPC network; VPCEs are used for the all services. Communication to these services are restricted to VPCE origination.
  + Firewall
    - AWS Security Groups are used to permit only HTTPS (443) traffic to interface VPCE originating from permitted security groups/subnets.
    - All Lambda functions execute within the VPC and use security groups for their associated ENIs.
* Credential Management
  + Snowflake connections leverage key pair authentication rather than the typical username/password authentication. Encrypted private keys are stored to AWS Secrets manager and accessed via IAM roles.
* Audit Logging
  + Each function associated with UDL writes logs to CloudWatch. Both default and custom logs are writing to the log group. Also Logs are published to enterprise logging system
* Audit Reporting
  + UDL intends to leverage the Enterprise Reporting System; external team effort that is still in progress. As an intermediary effort, notifications are emailed to UDL administrators for specific errors. Service Now tickets are generated and assigned to appropriate teams
* Client Access Methods
* Linux CLI
  + A command line interface (CLI) was built on Python to be ran from DFS imaged EC2s. The CLI allows users to onboard new tables, pause/resume table loads, mark events stale.
* Web Portal
  + Kibana enables users to search for table load status and also monitor any table failures.

# Technology and Product listing

**Amazon Web Services (AWS)**

5.0.1 **Simple Storage Service (S3)**

5.0.2 **DynamoDB**

5.0.3 **Simple Queue Service (SQS)**

5.0.4 **Simple Notification Service (SNS)**

5.0.5 **Virtual Private Cloud (VPC)**

5.0.6 **VPC Endpoints (VPCE)**

5.0.7 **Direct Connect**

5.0.8 **Lambda**

5.0.9 **ELK**

5.0.10 **CloudWatch**

5.0.11 **AWS Secrets Manager**

5.0.12 **EC2**

**Enterprise Services**

5.0.13 Service Now

5.0.14 LDAP

# Related Documents and Best Practices

[UDL Detail Design Document](https://teams.microsoft.com/l/file/A58089DE-97A1-4A2E-ACD0-6728704A16D2?tenantId=f3f068cf-080c-4824-a912-f8c4633bd454&fileType=docx&objectUrl=https%3A%2F%2Fdiscoverfinancial.sharepoint.com%2Fsites%2FPlatformsCapabilities%2FShared%20Documents%2FSlalom%2FSnowflake%20Ongoing%20Data%20Load%20Pattern%20-%20Insert%2CUpsert%2CDelete%2CReplace%20Operations.docx&baseUrl=https%3A%2F%2Fdiscoverfinancial.sharepoint.com%2Fsites%2FPlatformsCapabilities&serviceName=teams&threadId=19:428a9854e8a34927b651b459a569c98b@thread.skype&groupId=461f1172-ca84-4bed-bd04-1c6bc5b81a59)

[UDL Support Documentation](https://teams.microsoft.com/l/file/3CD76AB7-99D5-4EBD-BFC4-1D9AB9D73391?tenantId=f3f068cf-080c-4824-a912-f8c4633bd454&fileType=docx&objectUrl=https%3A%2F%2Fdiscoverfinancial.sharepoint.com%2Fsites%2FPlatformsCapabilities%2FShared%20Documents%2FSlalom%2FUDL%20Support%20Ops%20Doc.docx&baseUrl=https%3A%2F%2Fdiscoverfinancial.sharepoint.com%2Fsites%2FPlatformsCapabilities&serviceName=teams&threadId=19:428a9854e8a34927b651b459a569c98b@thread.skype&groupId=461f1172-ca84-4bed-bd04-1c6bc5b81a59)

[UDL Wiki](https://github.discoverfinancial.com/aap-cde-org/udl/wiki/UDL-Architecture)

# Glossary

Note: Also refer to the [DFS Glossary](http://dtoday.discoverfinancial.com/glossary/)

# Revision History and Contributors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Revision Date | Approved By | Author | Changes | Revision |
| 6/4/2019 |  | Praveen Guduri | Initial Draft | 1.0 |
| 6/5/2019 |  | Praveen Guduri | Added Feedback from Sri, Shannon & Brent | 1.1 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Reviewers/Contributors:

|  |  |
| --- | --- |
| Name | Department/Contribution |
| Shannon Schultz | Advanced Analytics AWS Platforms - Reviewer |
| Brent Demar | Advanced Analytics Cloud Data Platforms - Reviewer |
| Srikanth Srinivasan | Cybersecurity - Reviewer |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Meta Tags

Note: Meta Tag in bold is the unique identifier

bt\_(title of solution pattern)\_solution\_pattern

bt\_solution\_pattern

bt\_(title of related reference architecture)\_reference\_architecture