

AWS INNOVATE

ONLINE CONFERENCE 2019

MACHINE LEARNING & AI EDITION

Build Your Data Lake Easily with AWS Lake Formation (Level 200)

Unni Pillai, Specialist Solutions Architect, AWS, ASEAN



In this session...

Why Data Lakes ?

Why did we build AWS Lake Formation?

Understand AWS Lake Formation Components?



There is **more data** than people think

Data

grows
>10x
every 5 years

Data platforms need to

live for
15
years

scale
1,000x

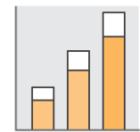
* IDC, Data Age 2025: The Evolution of Data to Life-Critical Don't Focus on Big Data, Focus on the Data That's Big, April 2017.



Data Scientists



Business Users



Analysts



Applications

Secure

Real time

Flexible

Scalable

There are more people
accessing data

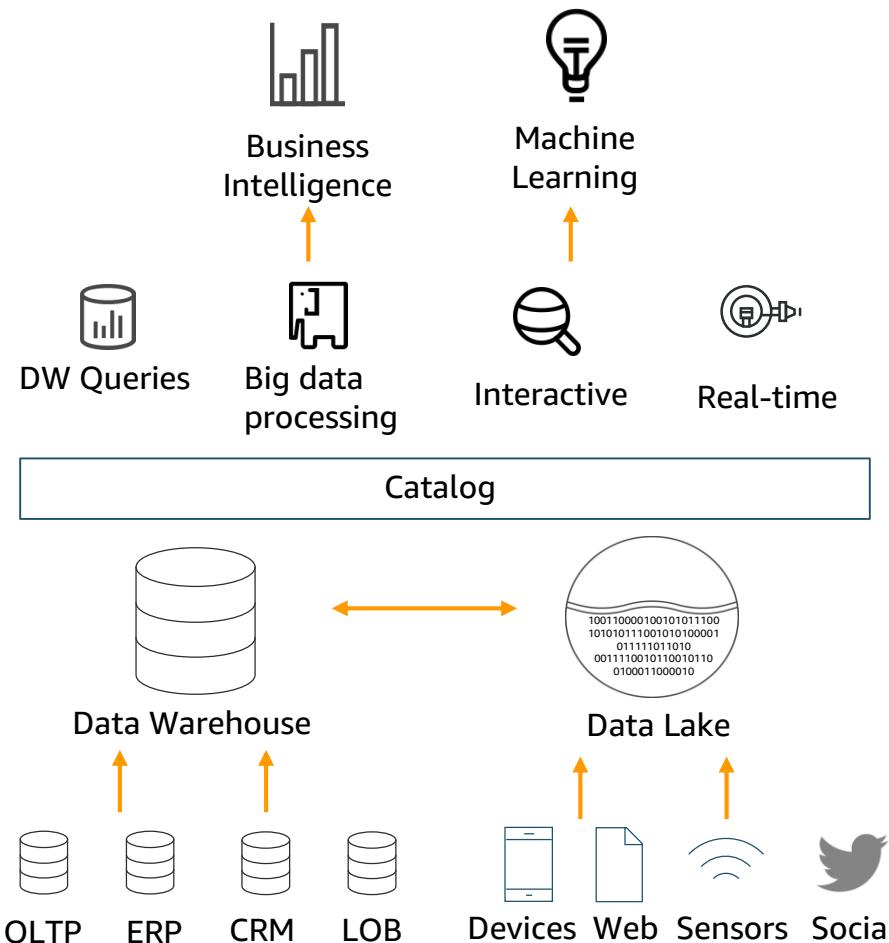
And more requirements
for making data available

A data lake is a centralized repository that allows you to store all your structured and unstructured data at any scale

More data lakes & analytics on AWS than anywhere else



Why data lakes?



Data Lakes provide:

Relational and non-relational data

Scale-out to EBs

Diverse set of analytics and machine learning tools

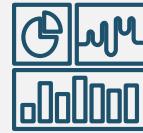
Work on data without any data movement

Designed for low cost storage and analytics



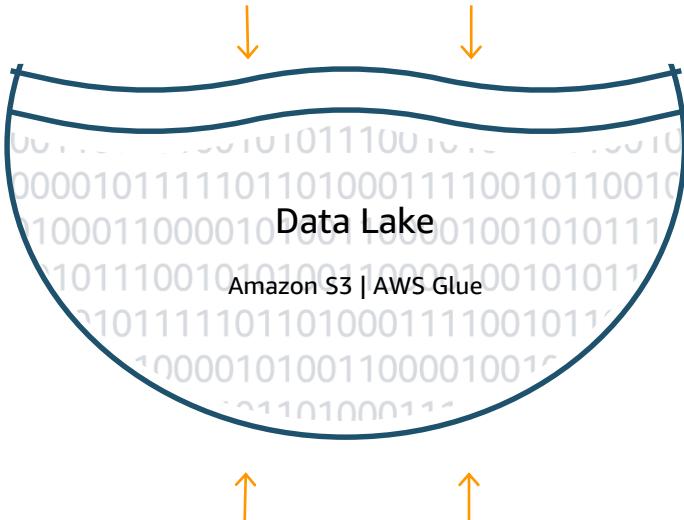
Machine Learning

- Amazon SageMaker
- AWS Deep Learning AMIs
- Amazon Rekognition
- Amazon Lex
- AWS DeepLens
- Amazon Comprehend
- Amazon Translate
- Amazon Transcribe
- Amazon Polly



Analytics

- Amazon Athena
- Amazon EMR
- Amazon Redshift
- Amazon Elasticsearch Service
- Amazon Kinesis
- Amazon QuickSight



On-premises Data Movement

- AWS Direct Connect
- AWS Snowball
- AWS Snowmobile
- AWS Database Migration Service

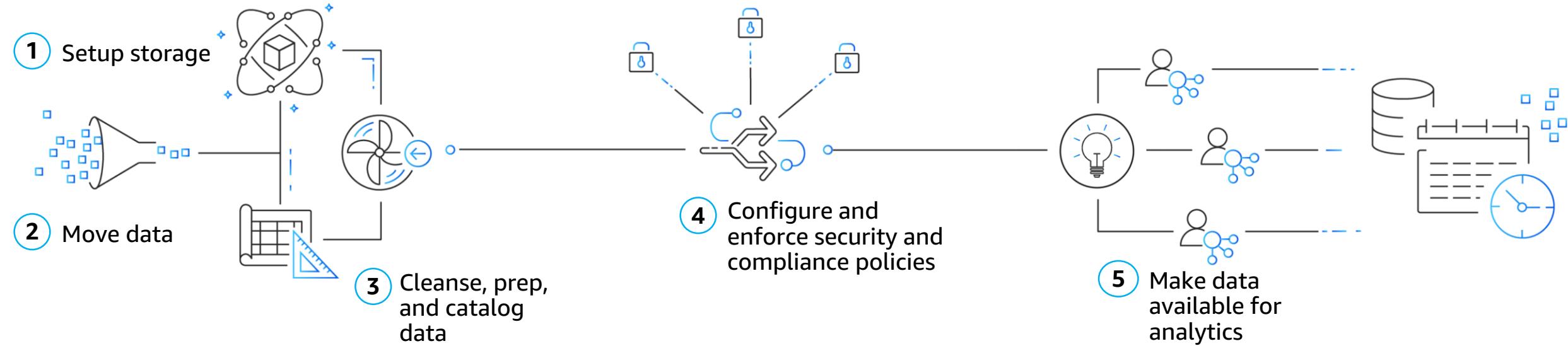


Real-time Data Movement

- AWS IoT Core
- Amazon Kinesis Data Firehose
- Amazon Kinesis Data Streams
- Amazon Kinesis Video Streams

Any analytic workload, any scale, at the lowest possible cost

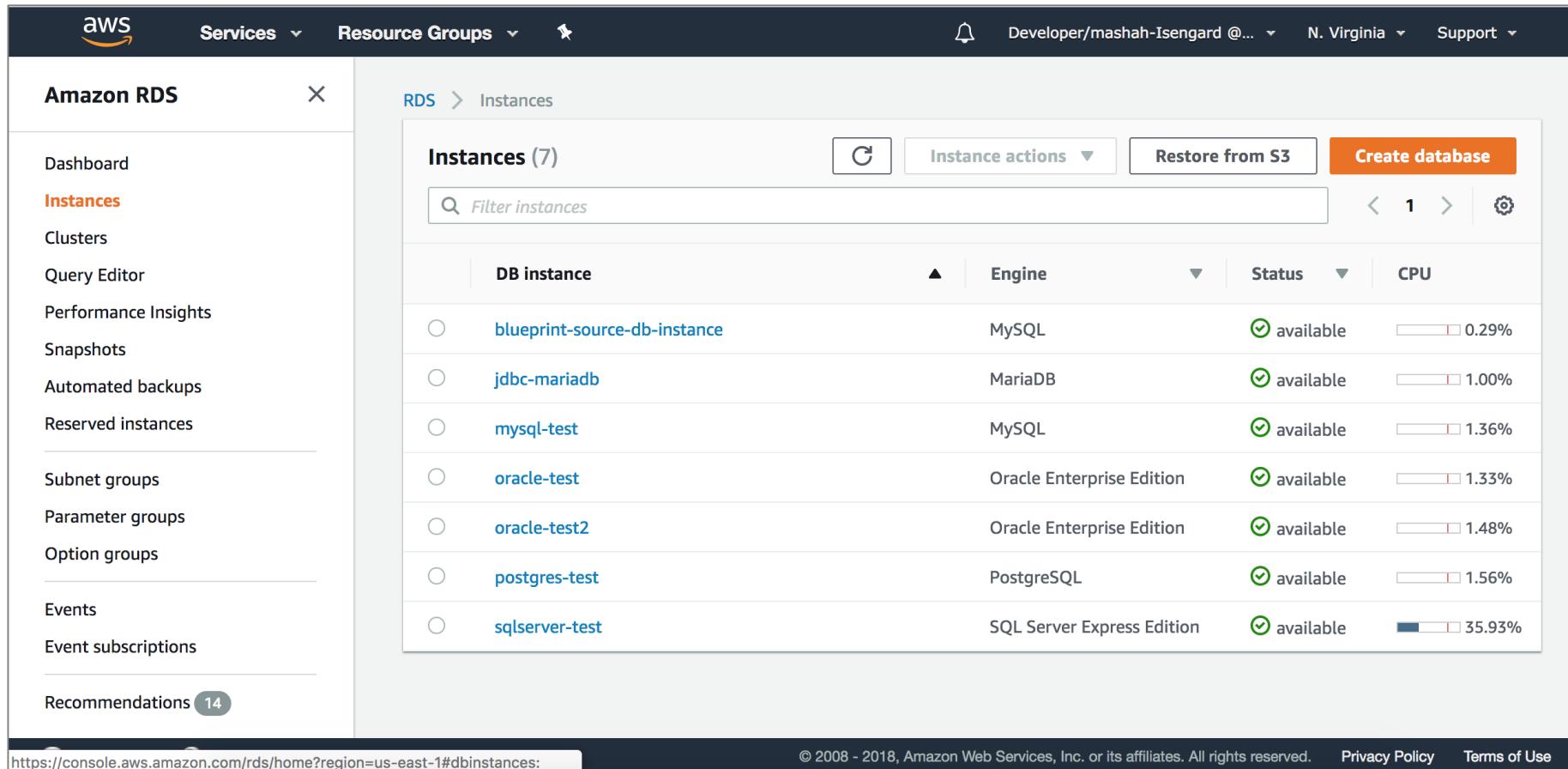
Typical steps of building a data lake



Building data lakes can still take months

Sample of steps required

Find sources



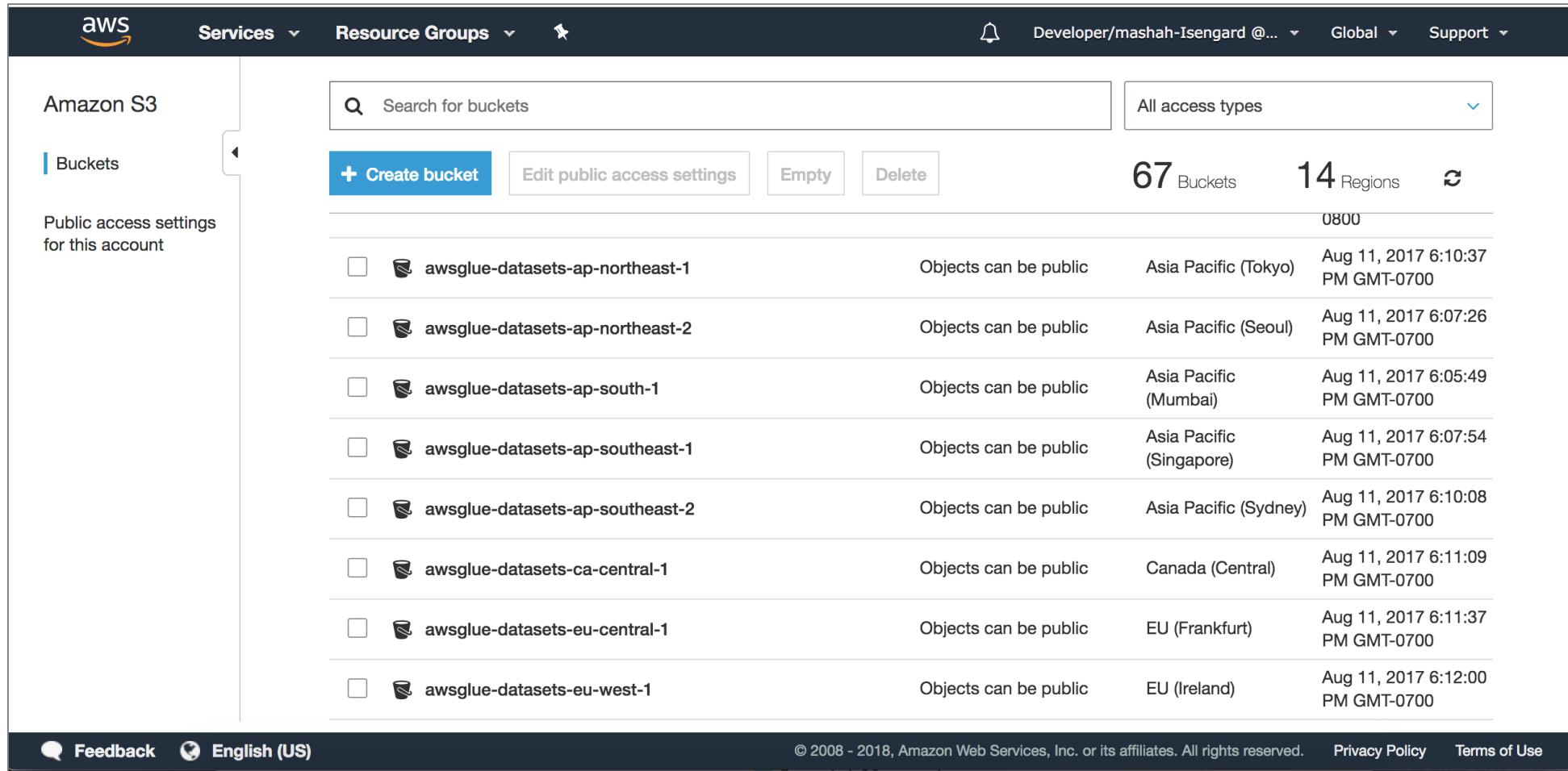
The screenshot shows the AWS RDS Instances page. The left sidebar lists various RDS management options like Dashboard, Instances, Clusters, and Query Editor. The main content area displays a table of seven database instances:

DB instance	Engine	Status	CPU
blueprint-source-db-instance	MySQL	available	0.29%
jdbc-mariadb	MariaDB	available	1.00%
mysql-test	MySQL	available	1.36%
oracle-test	Oracle Enterprise Edition	available	1.33%
oracle-test2	Oracle Enterprise Edition	available	1.48%
postgres-test	PostgreSQL	available	1.56%
sqlserver-test	SQL Server Express Edition	available	35.93%

The URL in the browser bar is <https://console.aws.amazon.com/rds/home?region=us-east-1#dbinstances>.

Sample of steps required

Create Amazon Simple Storage Service (Amazon S3) locations



The screenshot shows the AWS S3 console interface. The left sidebar has 'Amazon S3' selected under 'Buckets'. A message on the sidebar says 'Public access settings for this account'. The main area displays a search bar, a dropdown for 'All access types', and buttons for '+ Create bucket', 'Edit public access settings', 'Empty', and 'Delete'. It shows statistics: 67 Buckets and 14 Regions. Below is a table of buckets:

Bucket Name	Object Access	Region	Last Modified
awsglue-datasets-ap-northeast-1	Objects can be public	Asia Pacific (Tokyo)	Aug 11, 2017 6:10:37 PM GMT-0700
awsglue-datasets-ap-northeast-2	Objects can be public	Asia Pacific (Seoul)	Aug 11, 2017 6:07:26 PM GMT-0700
awsglue-datasets-ap-south-1	Objects can be public	Asia Pacific (Mumbai)	Aug 11, 2017 6:05:49 PM GMT-0700
awsglue-datasets-ap-southeast-1	Objects can be public	Asia Pacific (Singapore)	Aug 11, 2017 6:07:54 PM GMT-0700
awsglue-datasets-ap-southeast-2	Objects can be public	Asia Pacific (Sydney)	Aug 11, 2017 6:10:08 PM GMT-0700
awsglue-datasets-ca-central-1	Objects can be public	Canada (Central)	Aug 11, 2017 6:11:09 PM GMT-0700
awsglue-datasets-eu-central-1	Objects can be public	EU (Frankfurt)	Aug 11, 2017 6:11:37 PM GMT-0700
awsglue-datasets-eu-west-1	Objects can be public	EU (Ireland)	Aug 11, 2017 6:12:00 PM GMT-0700

At the bottom, there are links for 'Feedback', 'English (US)', 'Privacy Policy', and 'Terms of Use'.

Sample of steps required

Configure access policies

The screenshot shows the AWS S3 Bucket Policy editor. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, user information 'Developer/mashah-Isengard @...', 'Global' dropdown, and 'Support' dropdown. Below the navigation bar, there are tabs: 'Overview' (disabled), 'Properties' (disabled), 'Permissions' (disabled), 'Management' (disabled), and 'Bucket Policy' (selected). Underneath these tabs, there are four buttons: 'Public access settings' (disabled), 'Access Control List' (disabled), 'Bucket Policy' (selected and highlighted in blue), and 'CORS configuration' (disabled). The main content area is titled 'Bucket policy editor ARN: arn:aws:s3:::awsglue-datasets-us-east-1'. It contains a text area with a placeholder 'Type to add a new policy or edit an existing policy in the text area below.' Below the text area are three buttons: 'Delete', 'Cancel' (disabled), and 'Save'. The text area itself displays a JSON-based bucket policy document with line numbers from 1 to 20:

```
1 {  
2   "Id": "Policy1543402505352",  
3   "Version": "2018-11-28",  
4   "Statement": [  
5     {  
6       "Sid": "Stmt1543402503273",  
7       "Action": [  
8         "s3:GetObject",  
9         "s3>ListBucket",  
10        "s3>ListBucketByTags",  
11        "s3:PutObject"  
12      ],  
13      "Effect": "Allow",  
14      "Resource": "arn:aws:s3:::awsglue-datasets-us-east-1",  
15      "Principal": {  
16        "AWS": [  
17          "arn:aws:iam::<account#>:user/test-user"  
18        ]  
19      }  
20    }  
}
```

At the bottom of the editor, there are two links: 'Documentation' and 'Policy generator'. The footer of the page includes links for 'Feedback', 'English (US)', 'Privacy Policy', and 'Terms of Use'. The footer also contains copyright information: '© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.'

Sample of steps required

Map tables to Amazon S3 locations

The screenshot shows the AWS Glue 'Add table' interface. The top navigation bar includes the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, user information 'Developer/mashah-Isengard @...', region 'N. Virginia', and 'Support' dropdown.

The main title is 'Add table'. On the left, a sidebar lists configuration steps: 'Table properties' (checked), 'Data store' (checked), 'Data format' (checked), 'Schema' (unchecked), and 'Review' (unchecked). The 'Table properties' section shows 'Name: githubarchive_demo' and 'Database: githubarchive'. The 'Data store' section shows 's3://awsglue-datasets-us-east-1/'. The 'Data format' section shows 'JSON'.

The central area is titled 'Define a schema'. It features an 'Add column' button. Below it is a table with three columns: 'Column name', 'Data type', and 'Key'. The table rows are:

	Column name	Data type	Key	Comment
1	user_id	string		X
2	event_type	string		X
3	payload	STRUCT		X

Below the table, a message says 'Showing: 1 - 3 of 3 < >'. At the bottom are 'Back' and 'Next' buttons. The footer includes links for 'Feedback', 'English (US)', 'Privacy Policy', and 'Terms of Use'.

Sample of steps required

ETL jobs to load and clean data

Job: github_2_csv Action ▾ Save Run job Generate diagram ⓘ Insert template at cursor ⓘ Source Target Target Location Transform Spigot ⓘ X

Database Name gitarchive Table Name 2015

Transform Name ApplyMapping

Transform Name ResolveChoice

Transform Name DropNullFields

Path s3://glue-sample-target/output-dir

```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7
8 ## @params: [JOB_NAME]
9 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
10
11 sc = SparkContext()
12 glueContext = GlueContext(sc)
13 spark = glueContext.spark_session
14 job = Job(glueContext)
15 job.init(args['JOB_NAME'], args)
16 ## @type: DataSource
17 ## @args: [database = "gitarchive", table_name = "2015", transformation_ctx = "datasource0"]
18 ## @return: datasource0
19 ## @inputs: []
20 datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "gitarchive", table_name = "2015", transformation_ctx = "datasource0")
21 ## @type: ApplyMapping
22 ## @args: [mapping = [{"id": "string", "id": "string"}, {"type": "string", "type": "string"}, {"actor.login": "string", "actor": "string"}, {"repo.name": "string", "repo.id": "string"}], transformation_ctx = "applymapping1"]
23 ## @return: applymapping1
24 ## @inputs: [frame = datasource0]
25 applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [{"id": "string", "id": "string"}, {"type": "string", "type": "string"}, {"actor.login": "string", "actor": "string"}, {"repo.name": "string", "repo.id": "string"}], transformation_ctx = "applymapping1")
26 ## @type: ResolveChoice
27 ## @args: [choice = "make_struct", transformation_ctx = "resolvechoice2"]
28 ## @return: resolvechoice2
29 ## @inputs: [frame = applymapping1]
30 resolvechoice2 = ResolveChoice.apply(frame = applymapping1, choice = "make_struct", transformation_ctx = "resolvechoice2")
31 ## @type: DropNullFields
32 ## @args: [transformation_ctx = "dropnullfields3"]
33 ## @return: dropnullfields3
34 ## @inputs: [frame = resolvechoice2]
35 dropnullfields3 = DropNullFields.apply(frame = resolvechoice2, transformation_ctx = "dropnullfields3")
36 ## @type: DataSink
37 ## @args: [connection_type = "s3", connection_options = {"path": "s3://glue-sample-target/output-dir"}, format = "parquet", transformation_ctx = "datasink4"]
38 ## @return: datasink4
39 ## @inputs: [frame = dropnullfields3]
40 datasink4 = glueContext.write_dynamic_frame.from_options(frame = dropnullfields3, connection_type = "s3", connection_options = {"path": "s3://glue-sample-target/output-dir"}, format = "parquet", transformation_ctx = "datasink4")
41 job.commit()
```

Logs Schema

Sample of steps required

Create metadata access policies

Create policy

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. [Learn more](#)

[Visual editor](#) [JSON](#) [Import managed policy](#)

```
 5  "Version": "2012-10-17",
 6  "Statement": [
 7    {
 8      "Effect": "Allow",
 9      "Action": [
10        "glue:GetTables"
11      ],
12      "Resource": [
13        "arn:aws:glue:us-west-2:123456789012:catalog",
14        "arn:aws:glue:us-west-2:123456789012:database/db1",
15        "arn:aws:glue:us-west-2:123456789012:table/db1/store_sales",
16        "arn:aws:glue:us-west-2:123456789012:table/db1/stores"
17      ]
18    }
19  ]
20 }
```

[Cancel](#) [Review policy](#)

Sample of steps required

Configure access from analytics services

```
[finals=#  
[finals=#  
[finals=#  
[finals=#  
[finals=#  
[finals=# grant select(sid,name), update, insert ON grades to test_user;  
GRANT  
[finals=# grant select on enrolled to test_user;  
GRANT  
[finals=# \dp+  
                                         Access privileges  
Schema |      Name       | Type | Access privileges | Column privileges | Policies  
-----+-----+-----+-----+-----+-----+-----+  
public | enrolled      | table | mashah=awdDxt/mashah+| test_user=r/mashah |  
public | enrolled_scores | table |  
public | grades         | table | mashah=awdDxt/mashah+| sid:                   +|  
                  |                         | test_user=aw/mashah | test_user=r/mashah+|  
                  |                         | name:                 +|  
                  |                         | test_user=r/mashah |  
public | overall_pct    | table |  
public | scores          | table |  
(5 rows)  
  
finals=# ]
```

Sample of steps required

Rinse and repeat for other:
data sets, users, and end-services

And more:

- manage and monitor ETL jobs
- update metadata catalog as data changes
- update policies across services as users and permissions change
- manually maintain cleansing scripts
- create audit processes for compliance

...

Manual | Error-prone | Time consuming

AWS Lake Formation

Build a secure data lake in days



Identify, ingest,
clean, and transform
data

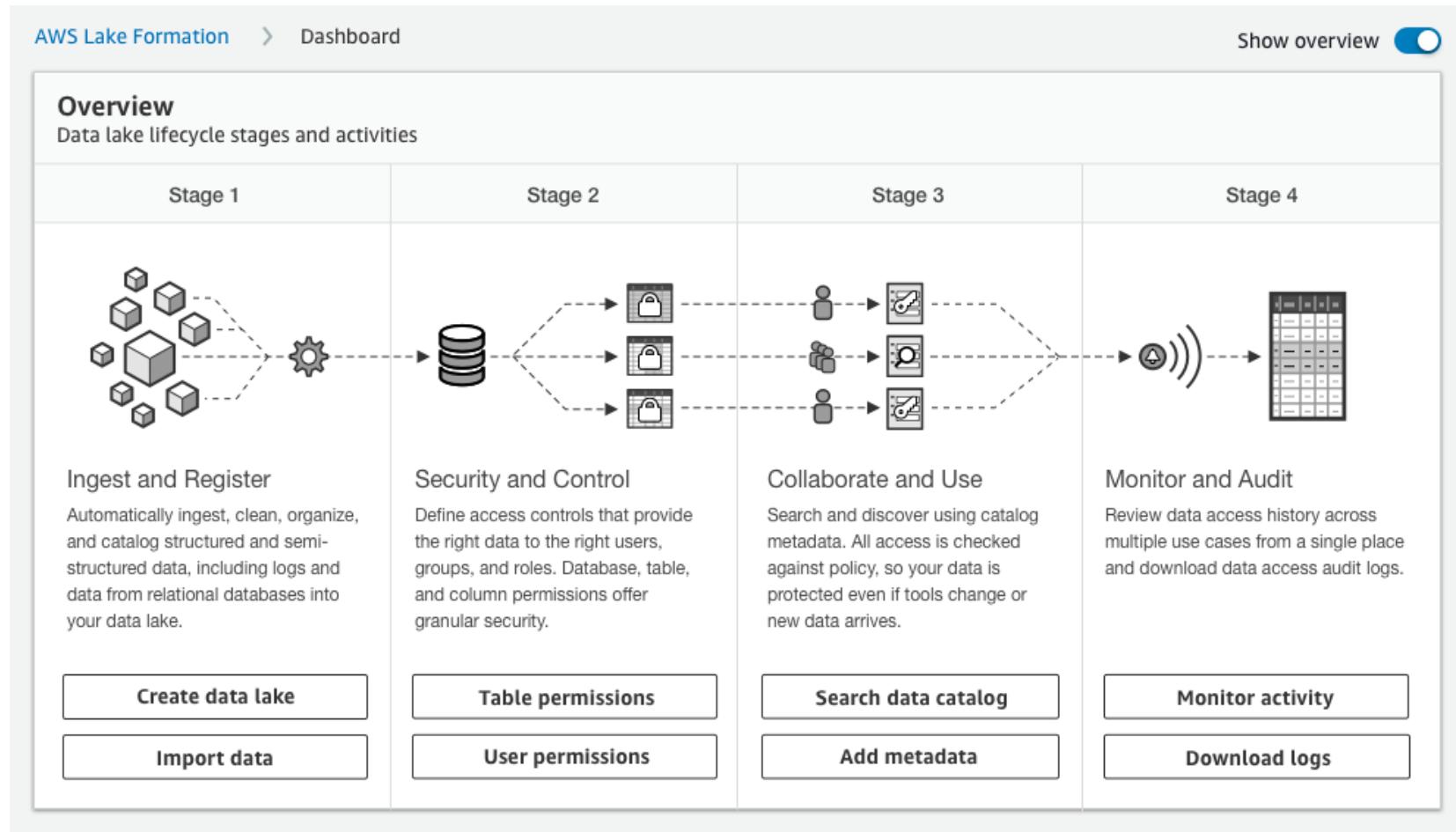


Enforce security
policies across
multiple services



Gain and manage new
insights

How it works



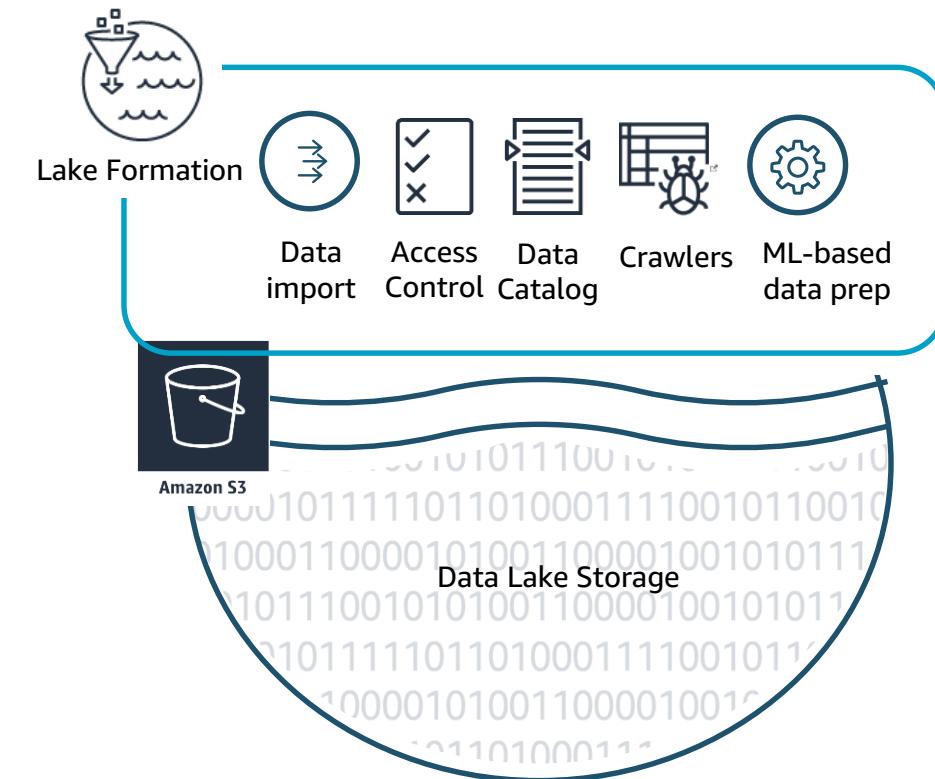
Register existing data or import new

Amazon S3 forms the storage layer for AWS Lake Formation

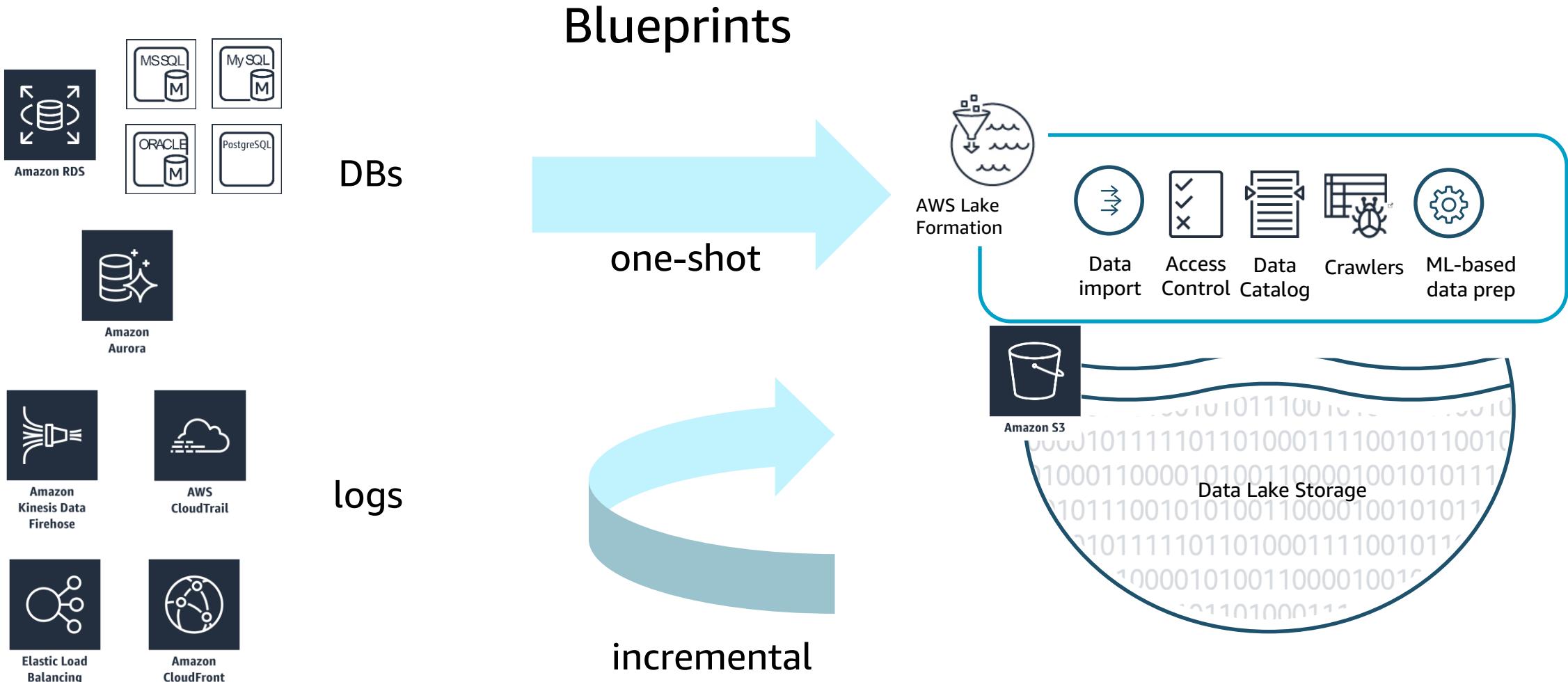
Register existing Amazon S3 buckets that contain your data

Ask AWS Lake Formation to create required Amazon S3 buckets and import data into them

Data is stored in your account. You have direct access to it. No lock-in.



Easily load data to your data lake



With blueprints

You

1. Point us to the source
2. Tell us the location to load to in your data lake
3. Specify how often you want to load the data

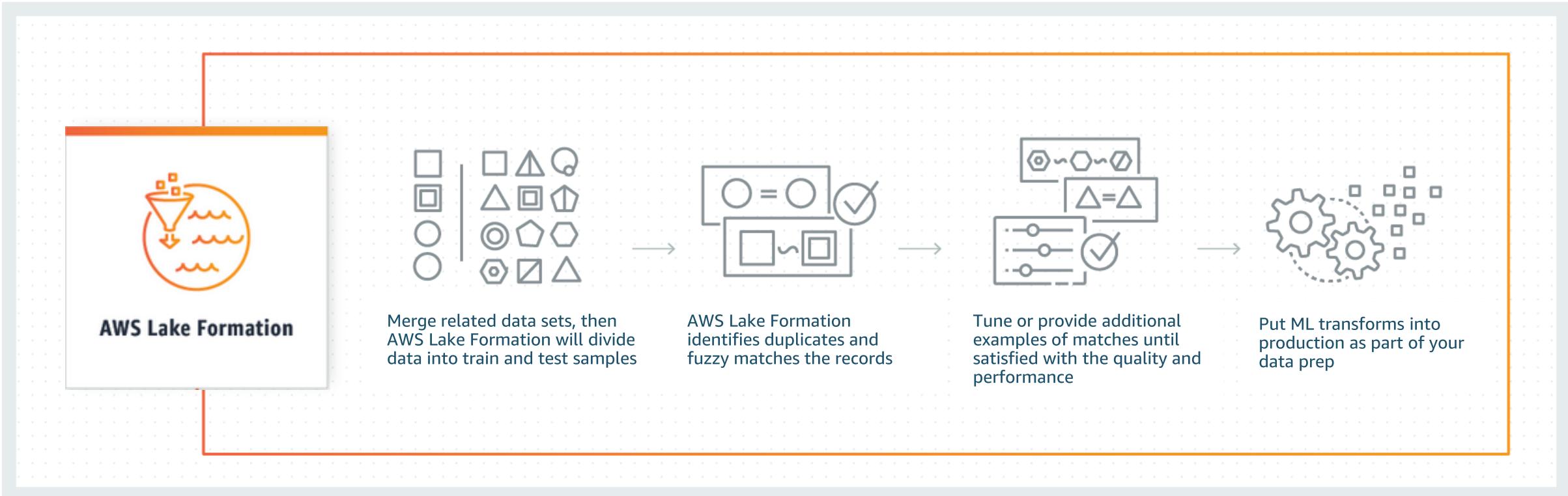
Blueprints

1. Discover the source table(s) schema
2. Automatically convert to the target data format
3. Automatically partition the data based on the partitioning schema
4. Keep track of data that was already processed
5. You can customize any of the above

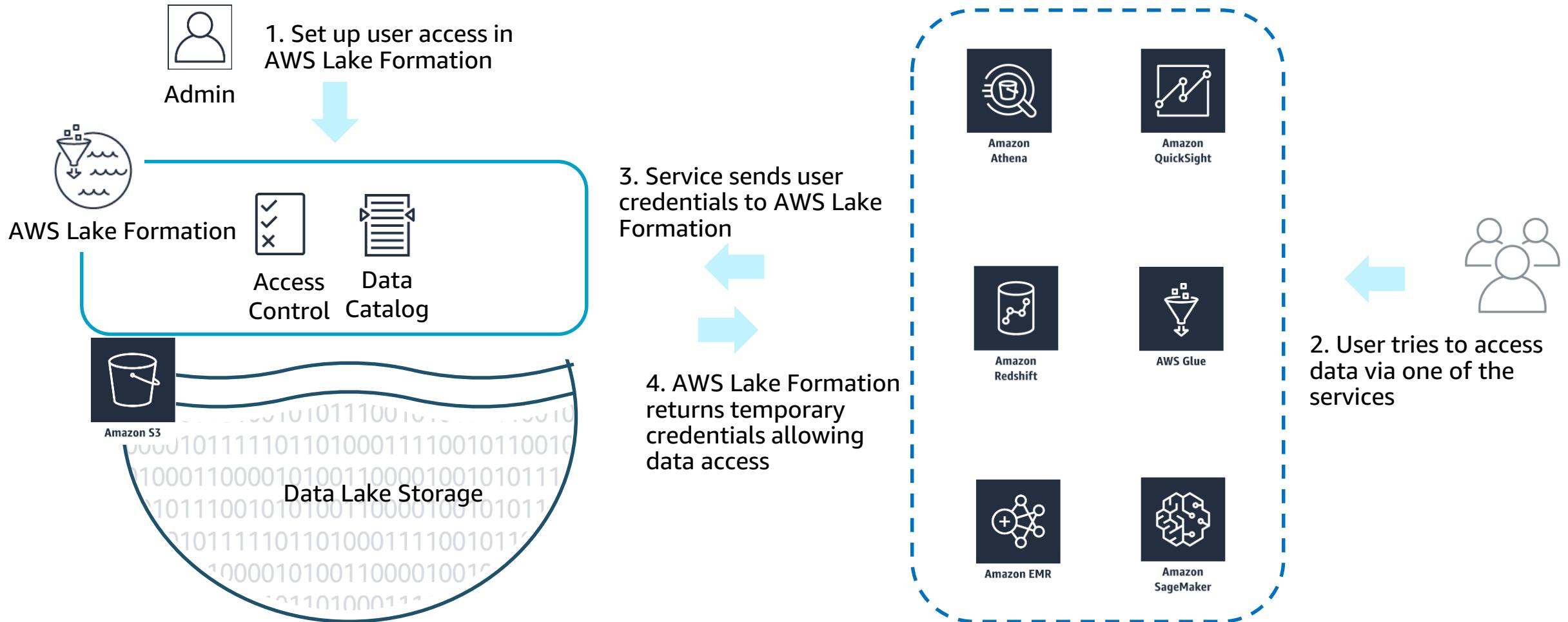
Blueprints build on AWS Glue



Easily de-duplicate your data with ML transforms



Secure once, access in multiple ways



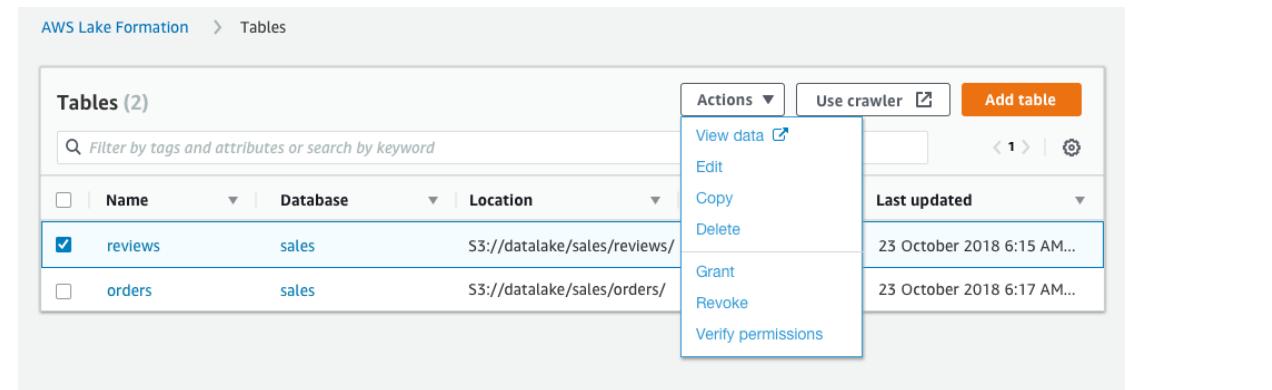
Security permissions in AWS Lake Formation

Control data access with simple grant and revoke permissions

Specify permissions on tables and columns rather than on buckets and objects

Easily view policies granted to a particular user

Audit all data access at one place



The screenshot shows the AWS Lake Formation 'Tables' page. There are two tables listed: 'reviews' and 'orders'. The 'orders' table has its 'Name' field checked. On the right side, there's a vertical 'Actions' menu with options like 'View data', 'Edit', 'Copy', 'Delete', 'Grant', 'Revoke', and 'Verify permissions'. The 'Grant' option is highlighted with a blue border. Below this, there's a modal window titled 'Grant permissions to table orders'. It contains fields for 'User, group, or role name' (with 'johnd', 'salesgrp', and 'analyst' listed), 'Permissions' (with 'Specific permissions' selected and 'Create', 'Select', 'Insert', and 'Alter' checkboxes checked), and 'Columns - optional' (with a single asterisk '*' listed). At the bottom of the modal are 'Cancel' and 'Save' buttons.

Security permissions in AWS Lake Formation

Search and view permissions granted to a user, role, or group in one place

Verify permissions granted to a user

Easily revoke policies for a user

The screenshot shows the AWS Lake Formation User permissions interface. At the top, there's a search bar labeled "Search user permissions". Below it is a table titled "User permissions (3)" with columns: Name, Type, Permissions, and Last modified. The table lists:

Name	Type	Permissions	Last modified
johnd	User	3 permissions	11/28/2018 14:24
sales	Database	Create, Select, Insert	11/28/2018 13:37
reviews	Table	Administrator	11/28/2018 13:37
orders	Table	Create, Drop	11/28/2018 12:44

A modal dialog titled "Verify permissions for database sales" is open, showing a list of IAM users and roles: "johnd X" and "salesgrp X". Another modal dialog titled "Permissions for selected users (2)" shows the same two entries. A third modal dialog titled "Revoke all permissions for johnd" contains a text input field with placeholder "To confirm, type revoke all into the field." and "Cancel" and "Revoke" buttons.

Grant table and column-level permissions

Column name	Data type
marketplace	string
customer_id	bigint
review_id	string
product_id	string
product_parent	bigint
product_title	string
star_rating	string
helpful_votes	bigint
total_votes	bigint
vine	string
verified_purchase	string
review_headline	string
review_body	string
review_date	string
product_category	string

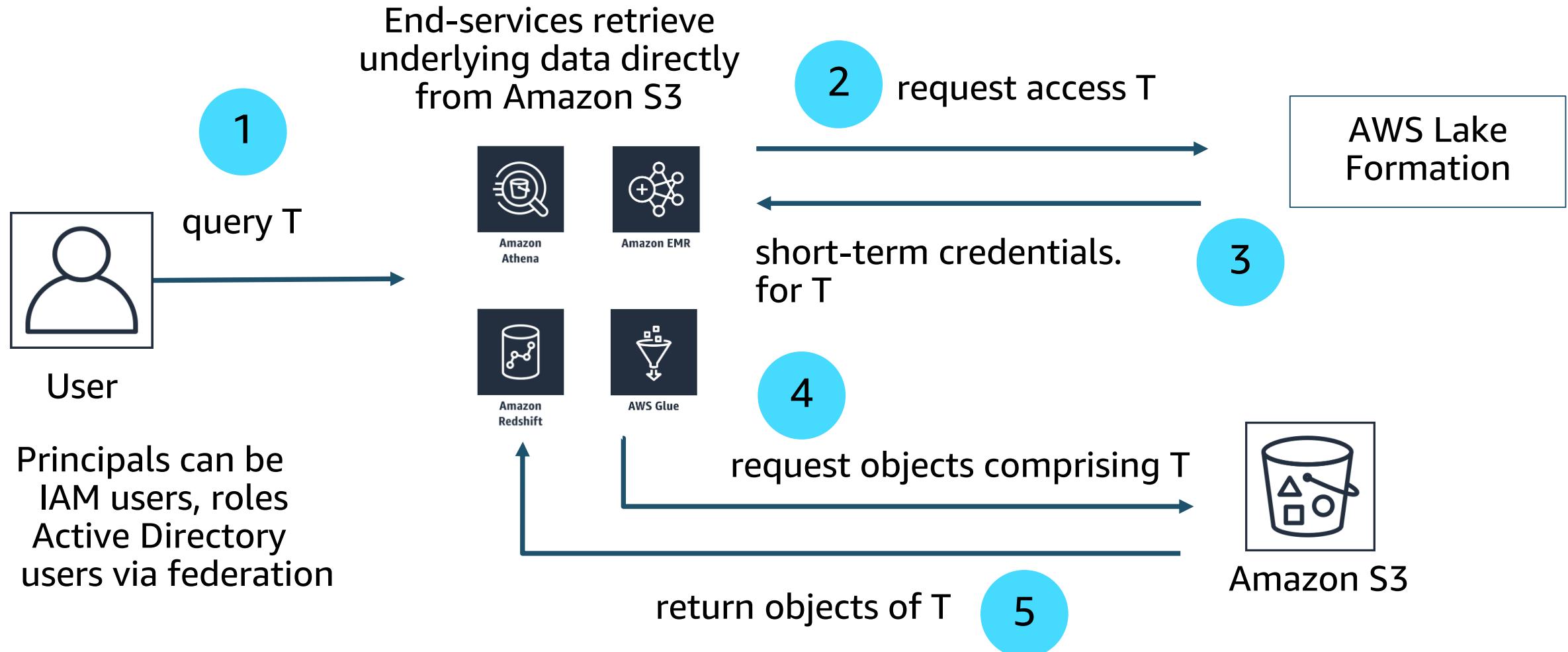


User 1



User 2

Security – deep dive



Search and collaborate across multiple users

Text-based, faceted search across all metadata

Add attributes like Data owners, stewards, and other as table properties

Add data sensitivity level, column definitions, and others as column properties

Text-based search and filtering

Tables A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

The screenshot shows the AWS Glue Data Catalog interface. At the top, there's a search bar with 'log' and a filter bar with 'Filter or search for tables...'. Below that is a table list with columns 'Name', 'Location', and 'Owner'. One row is selected: 'cloudtraildata' with location 's3://[REDACTED]/CloudTrailData/' and owner 'mys3'. A context menu is open over this row, with 'Edit table details', 'View details', 'View data', and 'Delete table' options. An orange arrow points from the 'View data' option in the menu to the bottom right of the screen. The bottom right shows the Amazon Athena results page with a query editor containing a SELECT statement and a results table with 10 rows of log data. Another orange arrow points from the 'Tables' section at the top left to the 'View data' button in the context menu.

Name	Location	Owner
cloudtraildata	s3://[REDACTED]/CloudTrailData/	mys3
elb_logs	s3://athena-examples-us-east-1/elb/plain...	
exportedlogs	s3://[REDACTED]/exportedlogs/	

Results

eventversion	eventid	eventtime	sharedeventid	requestparameters.durationseconds
1.05	4641cd80-5604-4008-a050-3804f1bf163	2017-10-23T12:24:08z	b86dd80-8506-448a-9f8c-c762e38ac1e	3600
2.05	29279603-980d-42ef-9d9f-ca70342881d	2017-10-23T12:24:21z	47927aca-6499-4591-8ff6-29e56198a70d	3600
3.05	d6614097-b33f-412d-80a2-f61d5d5d5ed	2017-10-23T12:24:37z	49730ab-5af1-4465-9441-98b198202a4	3600
4.05	c6bb0d138-1180-4035-853c-2d992e1d4d8	2017-10-23T12:24:41z	85841bf-0544-470e-85f0-ia0d554881b	3600
5.05	c21882a4-6a02-4a11-9329-901c208a208	2017-10-23T12:24:45z	2395175b-d749-4497-8a93-d88fbcb1045	3600
6.05	41608b47-aa05-4215-a059-4d4e4f620d98	2017-10-23T12:24:49z	63993da4-b552-4c09-a370-4d81726a88ba	3600
7.05	77dd0d42-8030-432b-8c7e-15bae65452e0	2017-10-23T12:24:51z	a9f257c9-d9bc-4549-88e4-1b627be5e14	3600
8.05	c0505bf9-6120-4a3b-90ab-f0adff7ca7a	2017-10-23T12:25:19z	f999a02-23f1-4009-9a16-ff9007763a3	3600
9.05	643c185a-ad36-41af-e82f-e9adfb680bc	2017-10-23T12:26:19z	f4ad4c7-d9f5-4b68-be97-dc107240e64	3600
10.05	0856c3ef-d928-4536-ea05-7fedee01011	2017-10-23T12:26:19z	c90ece15-497d-e49d-e490-67a1719b4ab3	3600

Audit and monitor in real time

See detailed alerts in the console

Download audit logs for further analytics

Data ingest and catalog notifications also published to Amazon CloudWatch events

The screenshot shows the AWS Lake Formation Dashboard. The 'Overview' section displays four stages: Stage 1 (Ingest and Register), Stage 2 (Security and Control), Stage 3 (Collaborate and Use), and Stage 4 (Monitor and Audit). Below each stage is a brief description and a row of buttons. A large orange circle highlights the 'Recent activity' section at the bottom right, which lists four audit events:

Event Description	Type	Resource	Date
Create access was granted to JohnD	Grant	data_science:trend_extract	November 28, 2018 14:24
Select access was revoked for TimK	Revoke	sales:invoices:customer, address	November 28, 2018 13:37
Data was accessed via Athena by DaveM	Access	finance:2018_monthly	November 28, 2018 13:06
A new table was added by DaveM	New table	finance:2019_projections	November 28, 2018 12:44

Example: a data lake in 3 easy steps

1. Use blueprints to ingest data
2. Grant permissions to securely share data
3. Query the data (Amazon Athena)

Step 1: Blueprints to ingest data

The screenshot shows the 'Create blueprint' page in the AWS Lake Formation console. At the top, the navigation bar includes 'Services', 'Resource Groups', and user information ('shyamsh @ 7821-0400-8917', 'N. Virginia', 'Support'). The breadcrumb path is 'AWS Lake Formation > Data load blueprints > Create blueprint'. The main section is titled 'Create blueprint' under 'Blueprint type', with a sub-instruction 'Choose a blueprint type based on the import data and options'. Two options are listed: 'Data load blueprints' (selected) and 'Database - bulk load' (disabled). The 'Database - incremental' option is selected, highlighted with a blue border. Below this, there are sections for 'Incremental database configuration' (configure database type, source, target, and frequency) and 'Source details' (Blueprint name: 'wordpress_import'). The bottom navigation bar includes 'Feedback', 'English (US)', and links to '© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.', 'Privacy Policy', and 'Terms of Use'.

Configure Source...

Incremental database configuration
Configure your incremental database blueprint including database type, source, target, and frequency

Source details

Blueprint name
Enter a blueprint name

Database connection
Choose the database connection to use
 ▼ C
[Create connection](#)

IAM role
Choose an IAM role
 ▼

Database path
Enter a path to source data

Incremental table and column

Table name	Bookmark keys - <i>optional</i>	Bookmark keys sort order - <i>optional</i>
<input type="text" value="wp_users"/>	<input type="text" value="ID"/> Enter a comma delimited list of bookmark keys	<input type="text" value="Ascending"/> ▼
<input type="text" value="wp_comments"/>	<input type="text" value="Type bookmark keys..."/> Enter a comma delimited list of bookmark keys	<input type="text" value="Descending"/> ▼ Remove

Monitor the import

The screenshot shows the AWS Lake Formation Data load blueprints interface. At the top, a green banner displays the message: "Your blueprint: wordpress_import has been started." Below the banner, the breadcrumb navigation shows: AWS Lake Formation > Data load blueprints > wordpress_import. The main content area is titled "wordpress_import" and contains two sections: "Blueprint details" and "Load status".

Blueprint details

- Name: wordpress_import
- IAM role: arn:aws:iam::782104008917:role/Glue_DefaultRole
- Target S3 location: s3://aws-glue-ingestor-demo-us-east-1/wordpress_import/
- Target database: wordpress_import
- Target format: PARQUET
- Status: CREATED

Load status

- Run state: SUCCESSFUL
- Imported table count: 12

Imported data as table in the data lake

AWS Lake Formation > Tables > wordpress_import_797a0017_wordpress_db_wp_users

wordpress_import_797a0017_wordpress_db_wp_users

[View properties](#) [Edit](#) [Delete](#) [Compare versions](#) [Edit schema](#) Version 1 (Current version) ▾

Table details		
Table Name	Column number	Column name
Database		Data type
wordpress_import_797a0017_wordpress_db_wp_users	1	user_status
wordpress_import	2	user_email
Location	3	user_login
s3://aws-glue-ingestor-demo-us-east-1/wordpress_import/wordpress_import_797a0017_wordpress_db_wp_users/version_0/	4	user_url
Last Updated	5	user_nicename
Tue Nov 20 2018 10:29:38 GMT-0800 (Pacific Standard Time)	6	user_registered
Output format	7	id
org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat	8	user_activation_key
Serde parameters	9	display_name
field.delim ,	10	user_pass

Schema

Filter Columns

Column number	Column name	Data type
1	user_status	int
2	user_email	string
3	user_login	string
4	user_url	string
5	user_nicename	string
6	user_registered	timestamp
7	id	bigint
8	user_activation_key	string
9	display_name	string
10	user_pass	string

Step 2: Grant permissions to securely share data

The screenshot shows the AWS Lake Formation User permissions interface. On the left, the navigation pane includes links for Dashboard, Ingest and register, Data catalog, Users and permissions (with User permissions selected), and Monitor and audit. The main area displays the 'User permissions' list, which shows three entries: 'johnd' (User, 3 permissions, last modified 11/28/2018 14:24), 'salesgrp' (Group, 1 permission, last modified 11/28/2018 13:37), and 'shyamsh' (User, 1 permission, last modified 11/28/2018 14:22). A modal window titled 'Grant permissions to table wordpress_import_797a0017_wordpress_db_wp_users' is open, showing the 'shyamsh' user selected in the 'IAM user, group, and roles' list. Under 'Permissions', the 'Specific permissions' radio button is selected, and the 'Select' checkbox is checked. The 'Columns - optional' section is empty. The 'Grant' button at the bottom right of the modal is highlighted.

Name	Type	Permissions	Last modified
johnd	User	3 permissions	11/28/2018 14:24
sales	Database	Create, Select, Insert	11/28/2018 13:37
reviews	Table	Administrator	11/28/2018 13:37
orders	Table	Create, Drop	11/28/2018 12:44
salesgrp	Group	1 permission	11/28/2018 13:37
sales	Database	Administrator	11/28/2018 13:37
analyst	Role	1 permission	11/28/2018 13:37
reviews	Table	Create, Select, Insert, Alter, Drop	11/28/2018 13:37
shyamsh	User	1 permission	11/28/2018 14:22
wordpress_import_797a00...	Table	Select	11/28/2018 14:22

Step 3: Run query in Amazon Athena

The image displays two side-by-side screenshots of the AWS Athena Query Editor. Both screenshots show the same basic layout: a top navigation bar with the AWS logo, Services dropdown, Resource Groups dropdown, a user profile section (circled in orange), N. Virginia region, Support dropdown, and a 'What's new' badge (10+). Below the navigation is a tabs bar with 'Athena' selected and 'Query Editor' highlighted.

Left Screenshot (shyamsh):

- Database:** wordpress_import
- Tables (1):** wordpress_import_797a0017_wordpress_db_wp_users
- Views (0):** None
- Query Editor:** A query window titled 'New query 1' containing the SQL command: `1 SELECT * FROM "wordpress_import"."wordpress_import_797a0017_wordpress_db_wp_users" limit 10;`. Below the query are buttons for 'Run query', 'Save as', and 'Create'. A note says '(Run time: 2.77 seconds, Data scanned: ...)'.
- Results:** A table showing two rows of data:

	user_status	user_email	user_login	user_url	user_nicename
1	0	galazzah@amazon.com	galazzah	galazzah	
2	0	shyamsh@amazon.com	shyamsh	shyamsh	

Right Screenshot (auslee):

- Database:** wordpress_import
- Tables (1):** wordpress_import_797a0017_wordpress_db_wp_users
- Views (0):** None
- Query Editor:** An empty 'New query 1' window with buttons for 'Run query', 'Save as', and 'Create'.
- Results:** An empty table area labeled 'Results'.

AWS Lake Formation Pricing

No additional charges – Only pay for the underlying services used.

What customer's have to say ...



"We are very excited about the launch of AWS Lake Formation, which provides a **central point of control** to easily load, clean, secure, and catalog data from thousands of clients to our AWS-based data lake, **dramatically reducing our operational load**. ... Additionally, AWS Lake Formation will be **HIPAA compliant** from day one ..."

Aaron Symanski, CTO, Change Healthcare



"I can't wait for my team to get our hands on AWS Lake Formation. With an enterprise-ready option like AWS Lake Formation, we will be able to **spend more time deriving value from our data** rather than doing the heavy lifting involved in manually setting up and managing our data lake."

Joshua Couch, VP Engineering, Fender Digital

Thank You for Attending AWS Innovate

We hope you found it interesting! A kind reminder to **complete the survey**. Let us know what you thought of today's event and how we can improve the event experience for you in the future.

 aws-apac-marketing@amazon.com

 twitter.com/AWSCloud

 facebook.com/AmazonWebServices

 youtube.com/user/AmazonWebServices

 slideshare.net/AmazonWebServices

 twitch.tv/aws

Learn why customers choose AWS for machine learning



Demo Arena

Watch how machine learning is used in real life applications



Ask the Experts

Get your questions answered by AWS Experts



Machine Learning on AWS

<https://aws.amazon.com/machine-learning/>

Learn from AWS experts. Advance your skills and knowledge. Build your future in the AWS Cloud.



Digital Training

Free, self-paced online courses built by AWS experts



Classroom Training

Classes taught by accredited AWS instructors



AWS Certification

Exams to validate expertise with an industry-recognized credential

Ready to begin building your cloud skills?
Get started at: <https://www.aws.training/>