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

aws

In this session…

Why Data Lakes ?

Why did we build AWS Lake Formation?

Understand AWS Lake Formation Components?

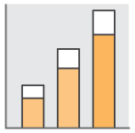
There is more data than people think

Data	Data platforms **need to**	
grows	live for	scale
>10x	**15**	**1,000x**
every 5 years	years	

* IDC, Data Age 20215: 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

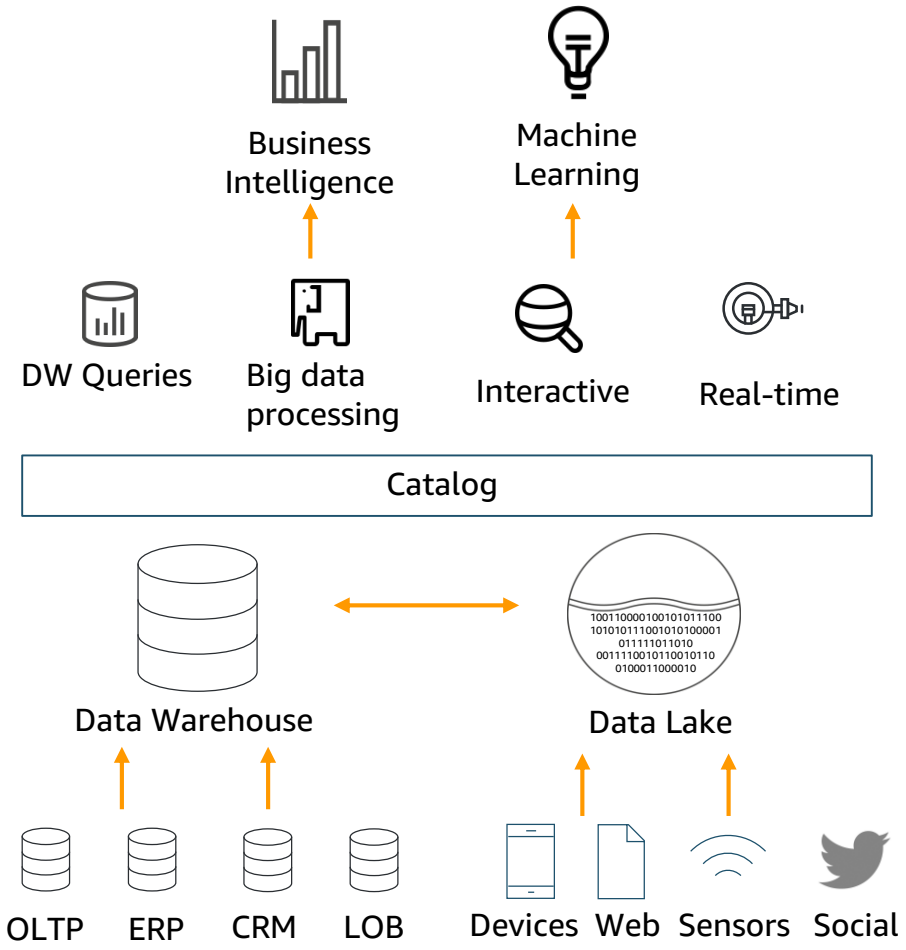
And more requirements for making data available

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

aws

Why data lakes?



Data Lakes provide:

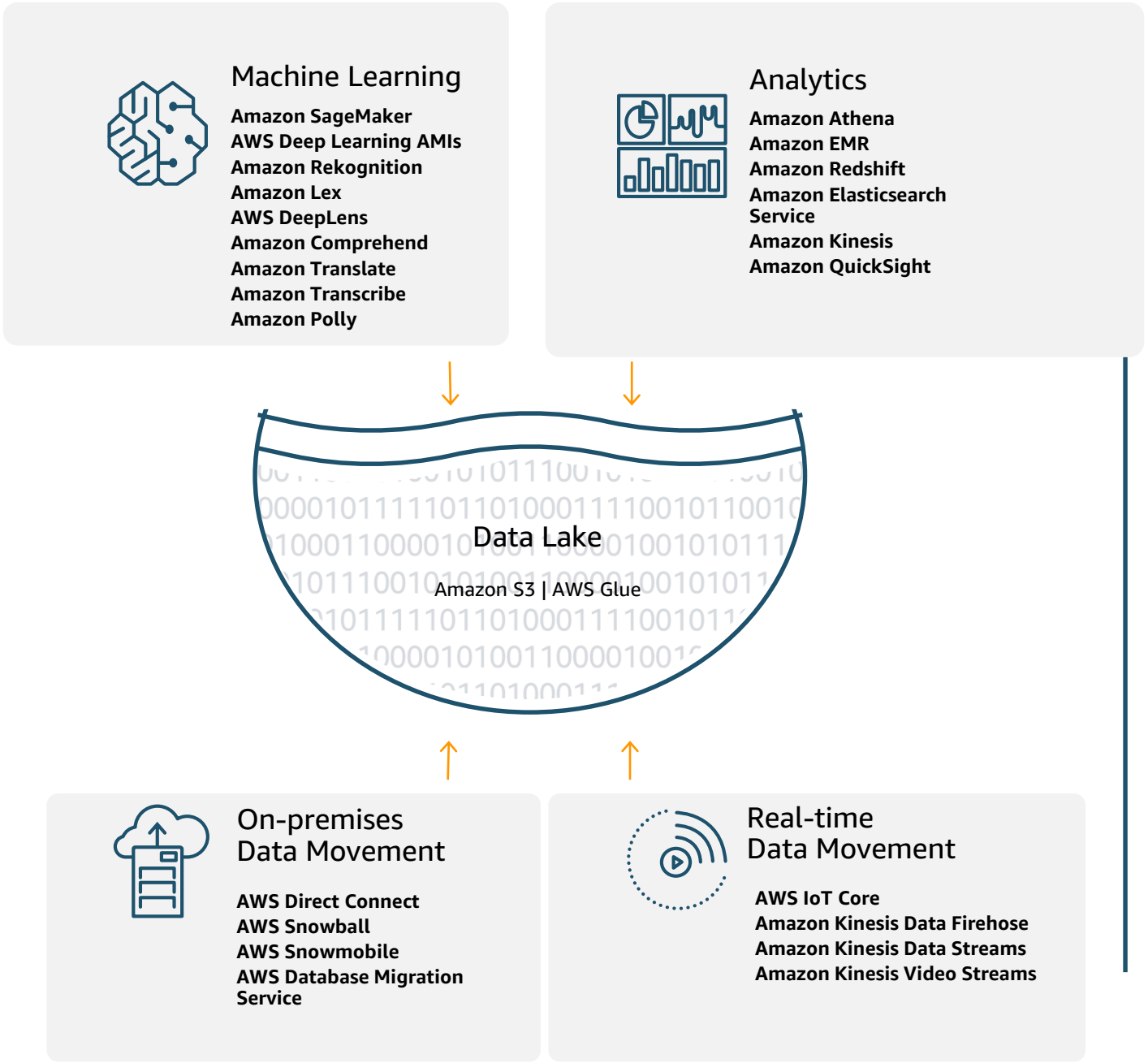
Relational and non-relational data

Scale-out to EBs

Diverse set of analytics and machine learning tools

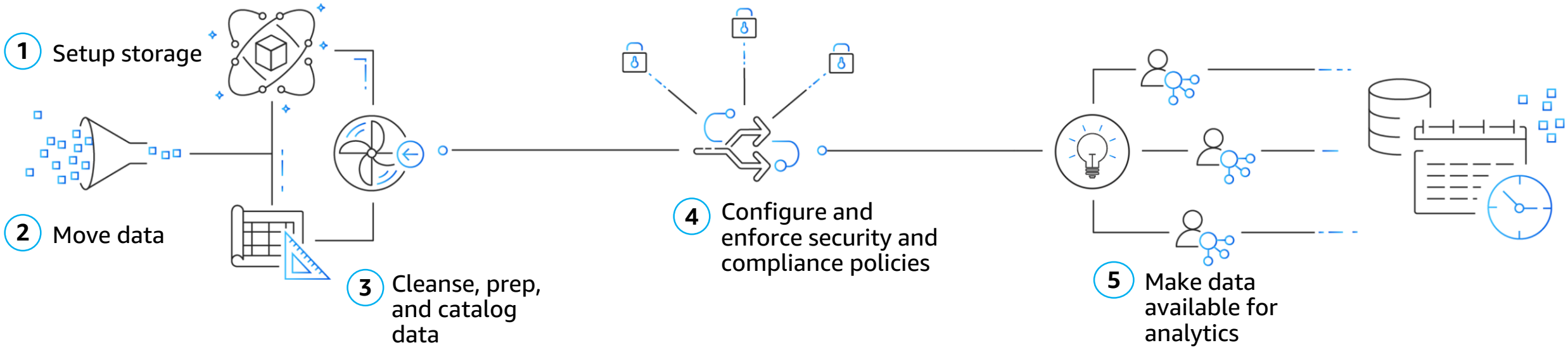
Work on data without any data movement

Designed for low cost storage and analytics



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

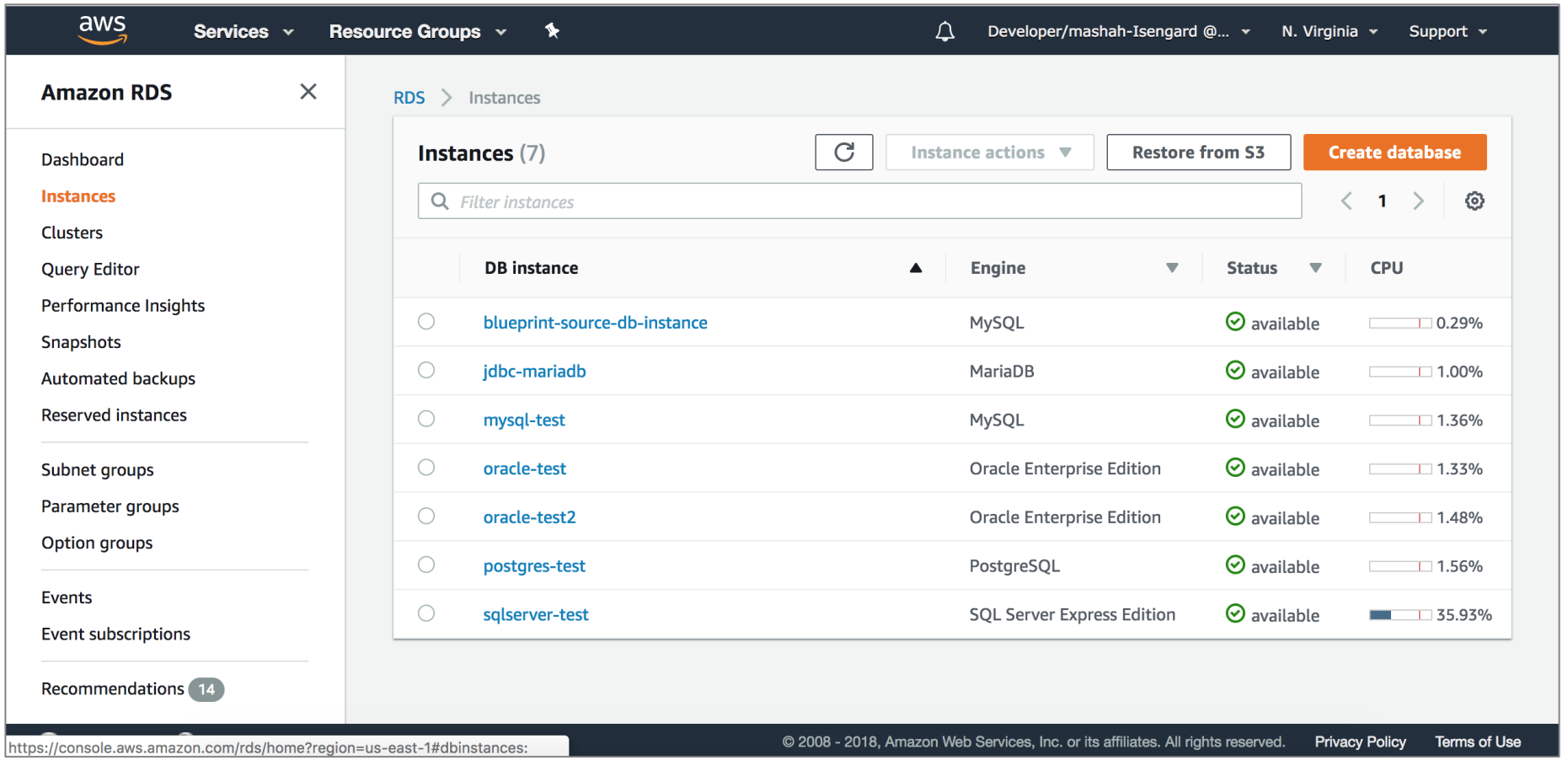
Typical steps of building a data lake



Building data lakes can still take months

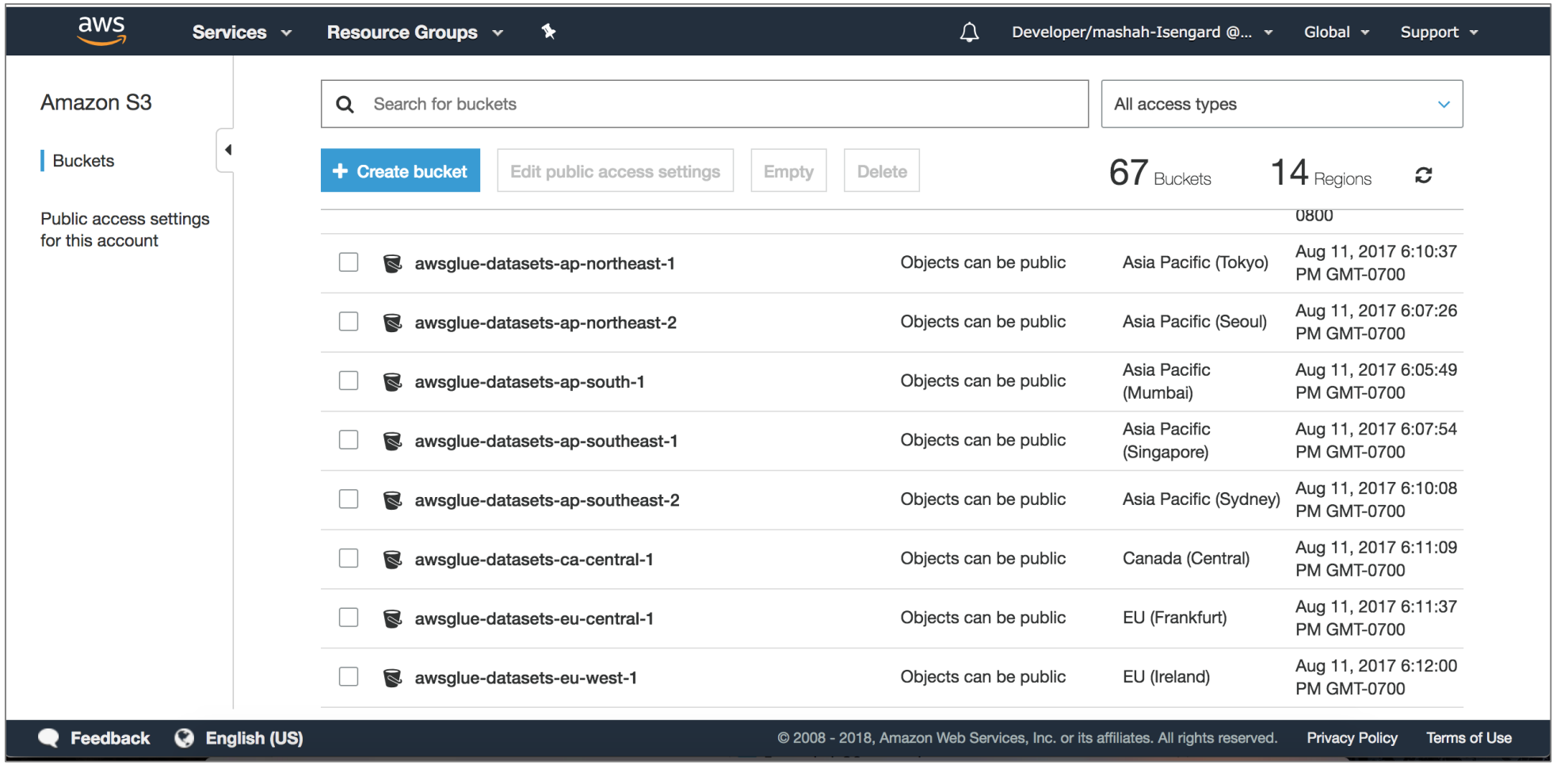
Sample of steps required

Find sources



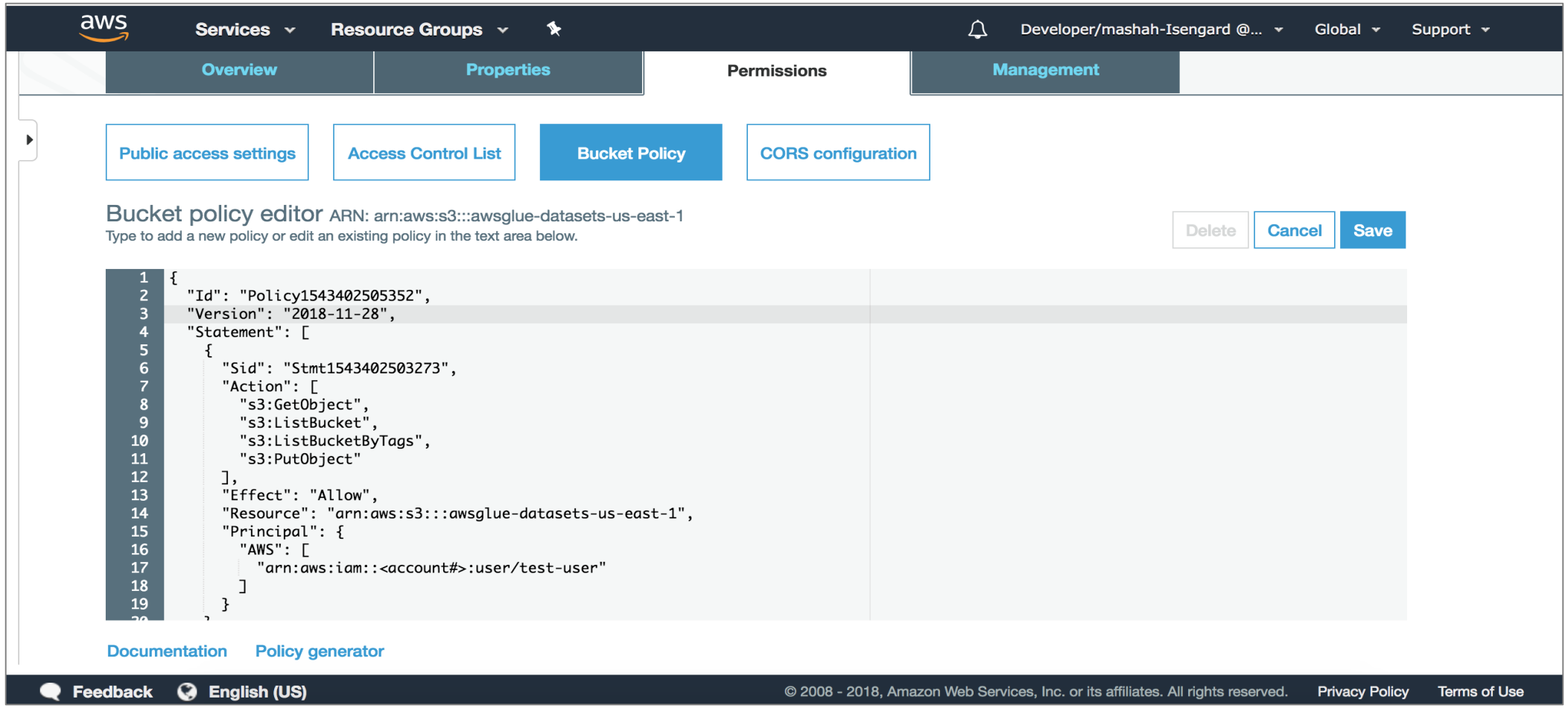
Sample of steps required

Create Amazon Simple Storage Service (Amazon S3) locations



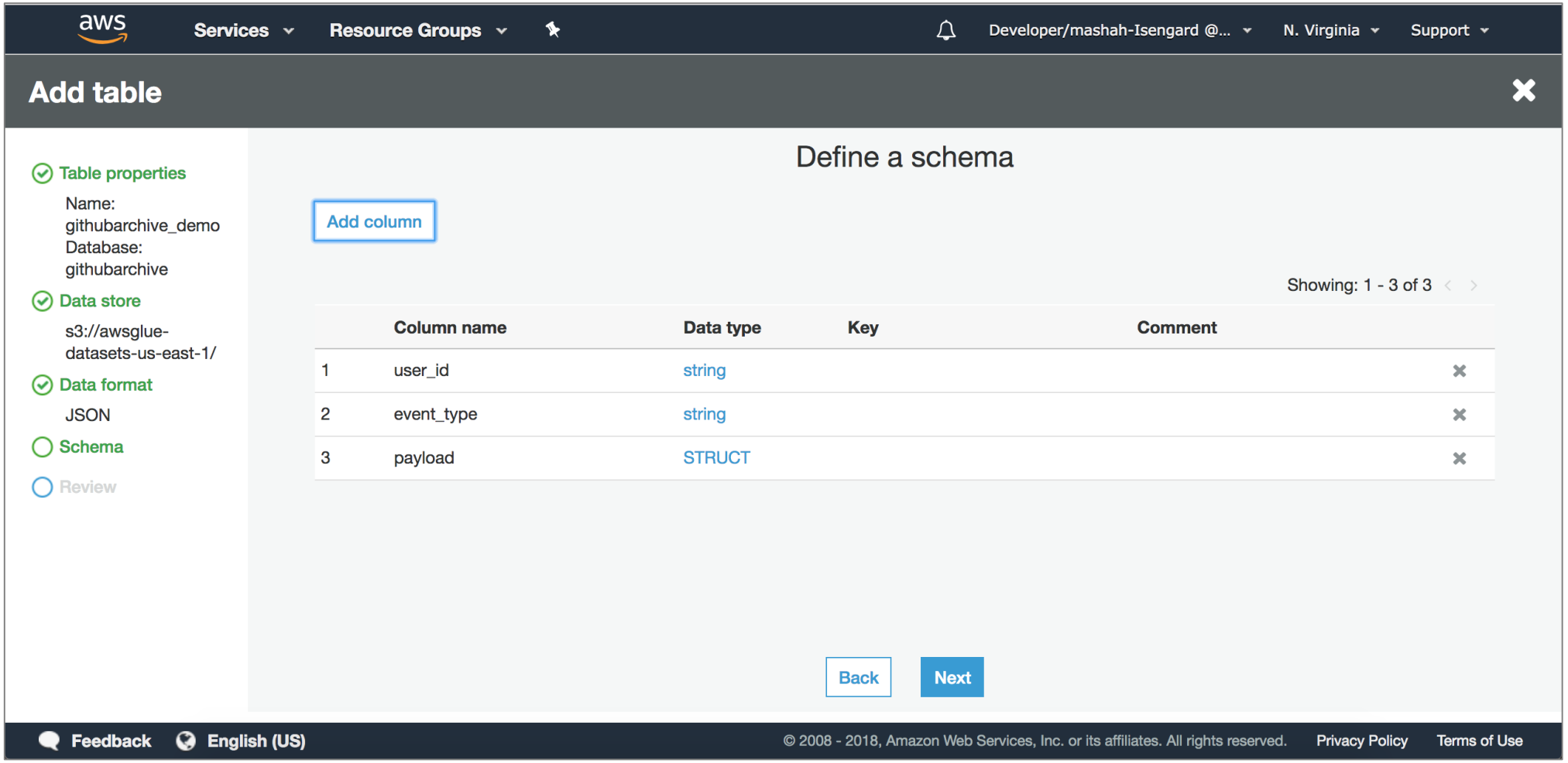
Sample of steps required

Configure access policies



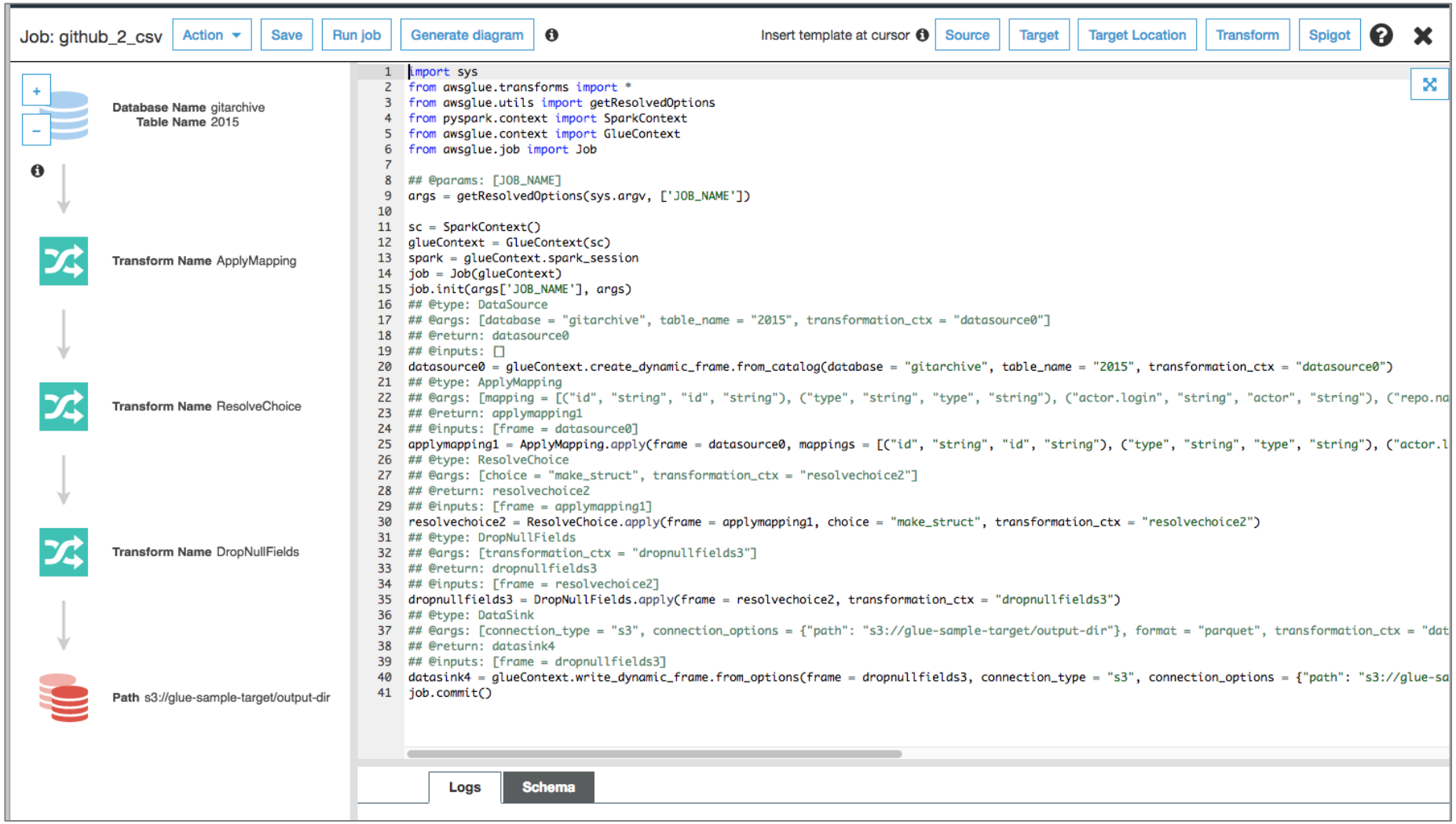
Sample of steps required

Map tables to Amazon S3 locations



Sample of steps required

ETL jobs to load and clean data



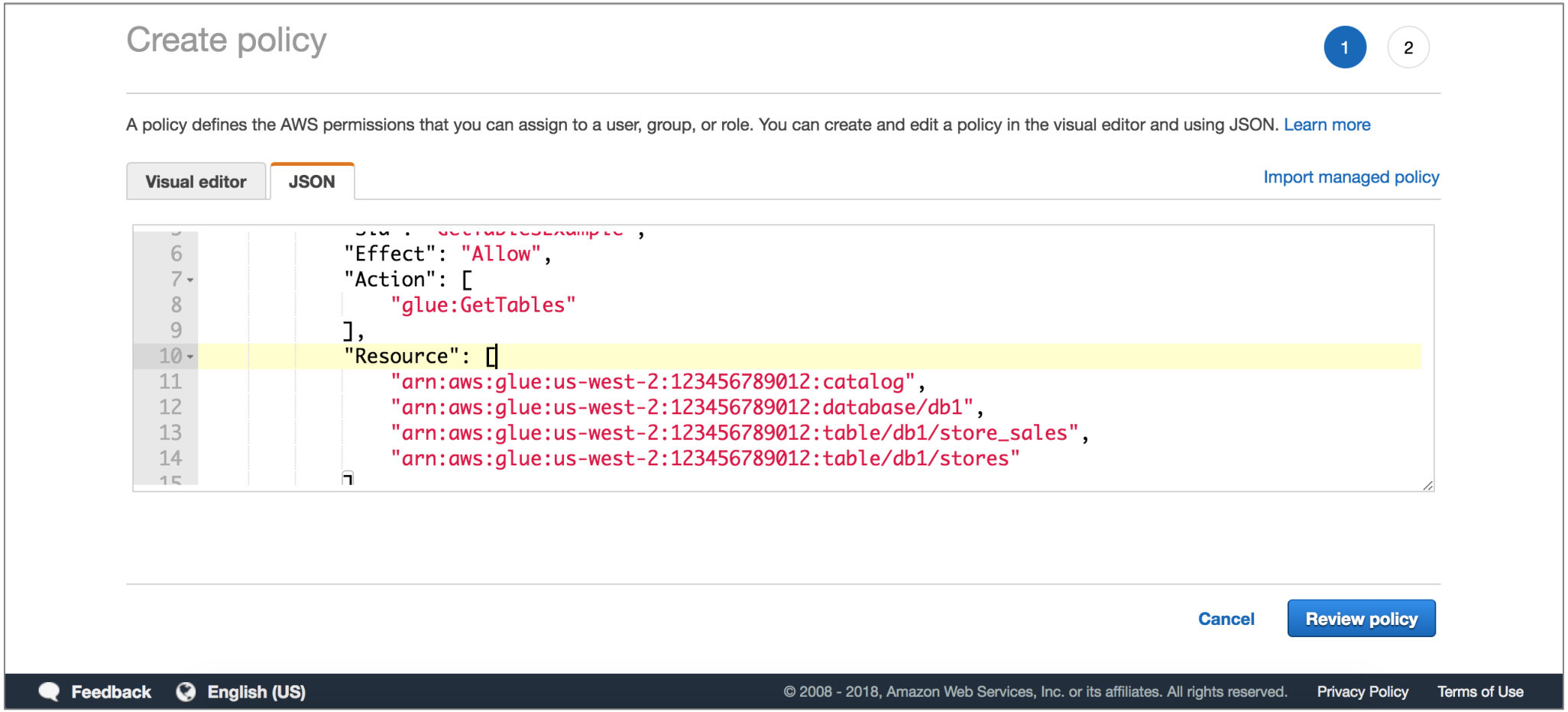

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)
## @type: DataSource
## @args: [database = "gitarchive", table_name = "2015", transformation_ctx = "datasource0"]
## @return: datasource0
## @inputs: []
datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "gitarchive", table_name = "2015", transformation_ctx = "datasource0")
## @type: ApplyMapping
## @args: [mapping = [("id", "string", "id", "string"), ("type", "string", "type", "string"), ("actor.login", "string", "actor", "string"), ("repo.na
## @return: applymapping1
## @inputs: [frame = datasource0]
applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [("id", "string", "id", "string"), ("type", "string", "type", "string"), ("actor.l
## @type: ResolveChoice
## @args: [choice = "make_struct", transformation_ctx = "resolvechoice2"]
## @return: resolvechoice2
## @inputs: [frame = applymapping1]
resolvechoice2 = ResolveChoice.apply(frame = applymapping1, choice = "make_struct", transformation_ctx = "resolvechoice2")
## @type: DropNullFields
## @args: [transformation_ctx = "dropnullfields3"]
## @return: dropnullfields3
## @inputs: [frame = resolvechoice2]
dropnullfields3 = DropNullFields.apply(frame = resolvechoice2, transformation_ctx = "dropnullfields3")
## @type: DataSink
## @args: [connection_type = "s3", connection_options = {"path": "s3://glue-sample-target/output-dir"}, format = "parquet", transformation_ctx = "dat
## @return: datasink4
## @inputs: [frame = dropnullfields3]
datasink4 = glueContext.write_dynamic_frame.from_options(frame = dropnullfields3, connection_type = "s3", connection_options = {"path": "s3://glue-sa
job.commit()
```

Sample of steps required

Create metadata access policies



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

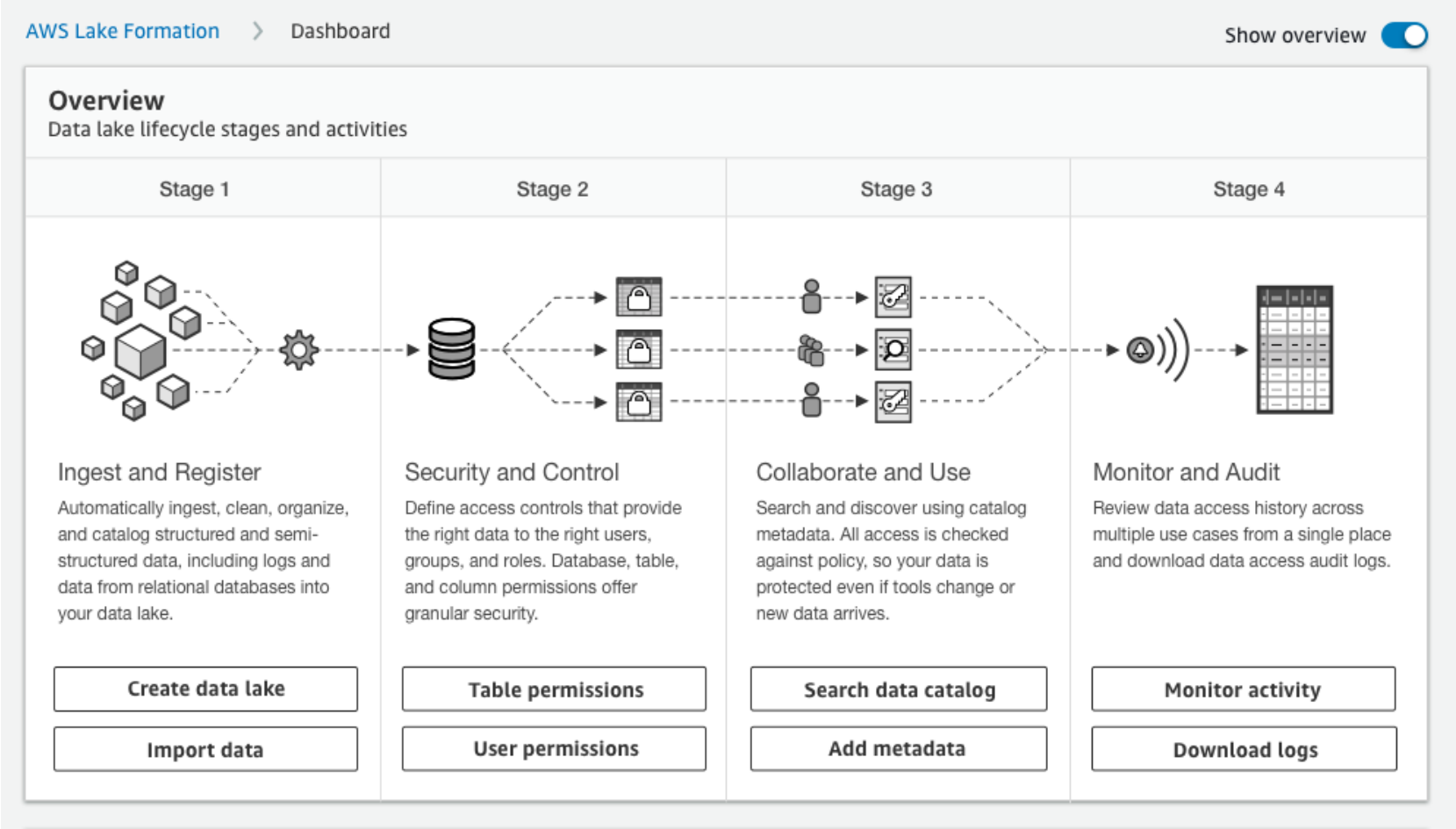
Build a secure data lake in days

Identify, ingest, clean, and transform data

Enforce security policies across multiple services

Gain and manage new insights

How it works



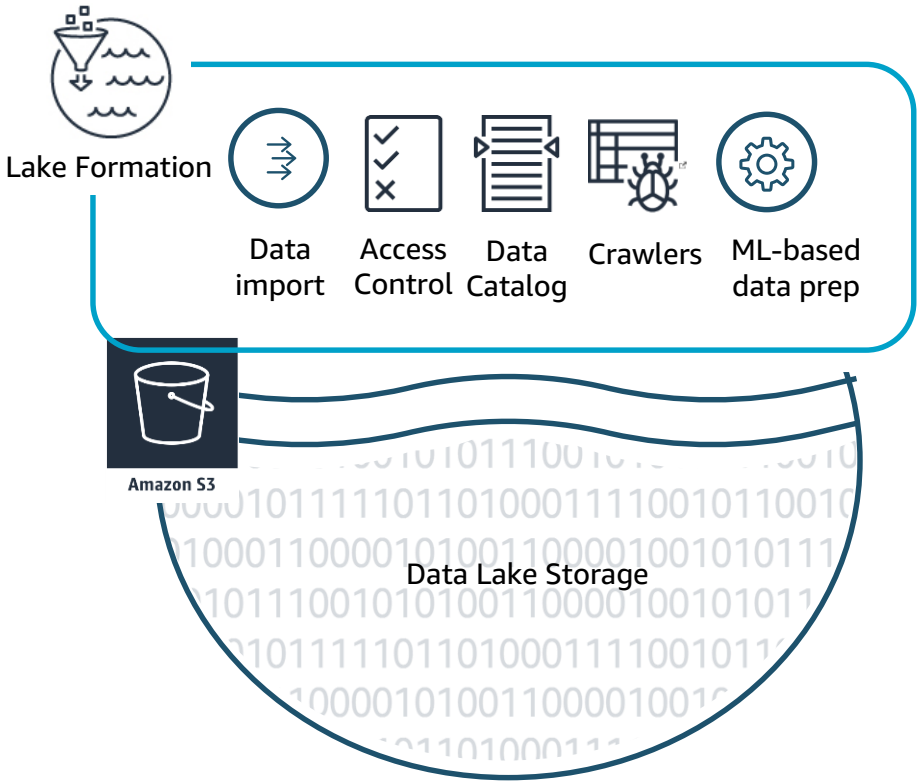
Register existing data or import new

Amazon S3 forms the storage layer for AWS Lake Formation

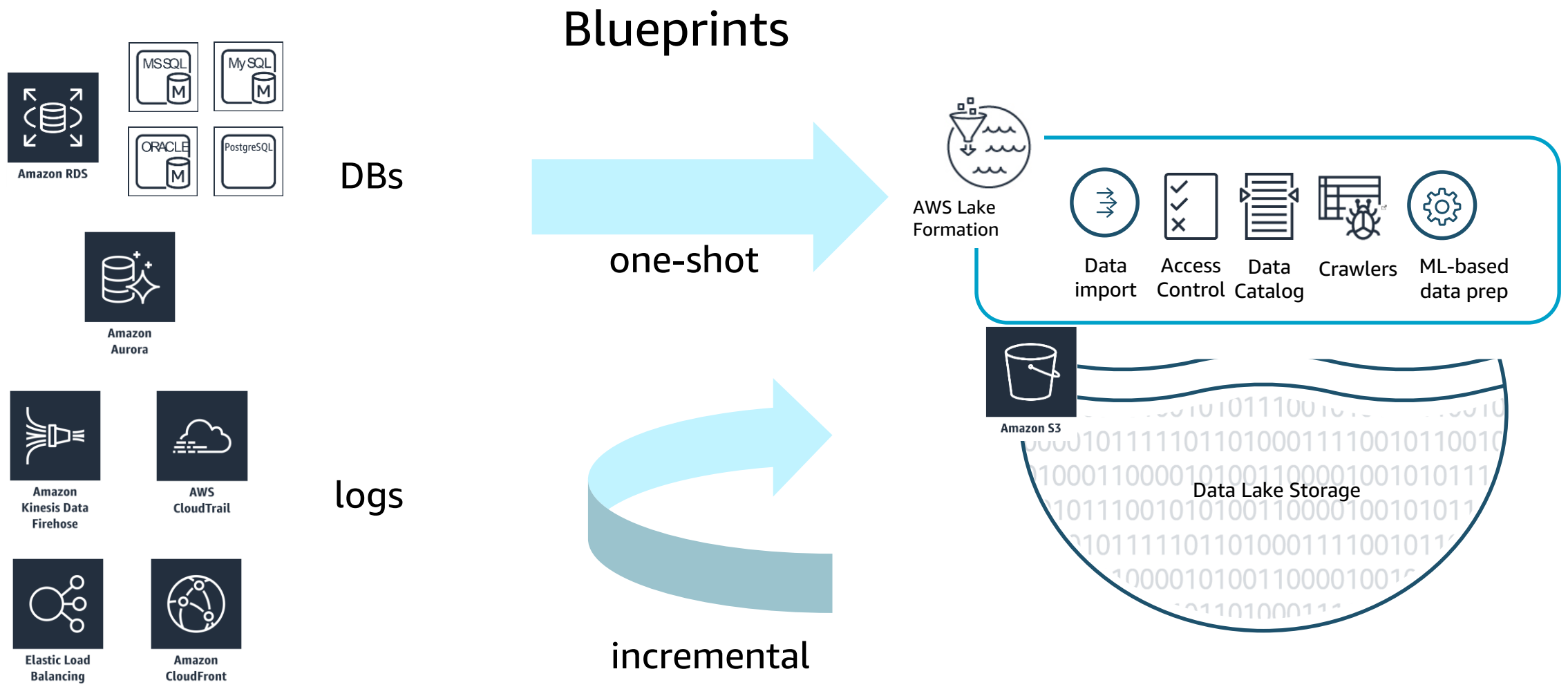
Register existing Amazon S3 buckets that contain your data

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

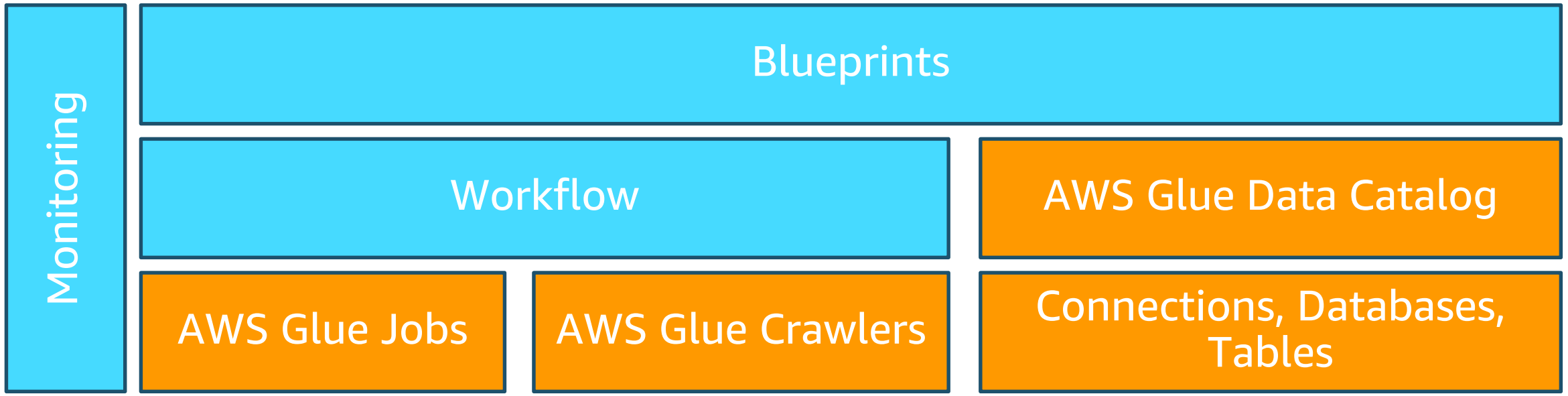
You

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

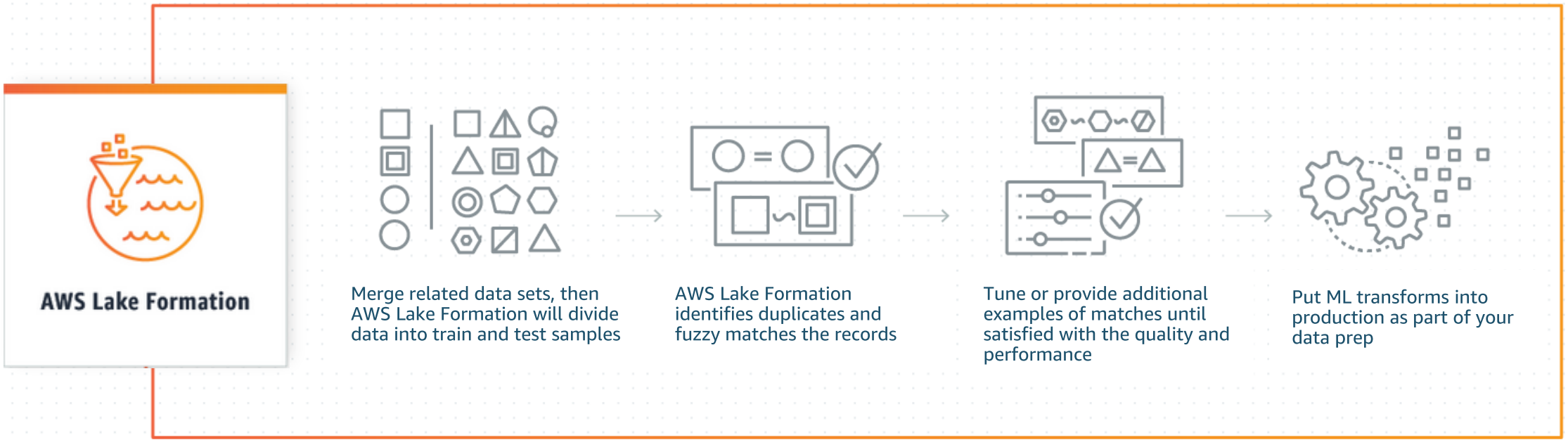
Blueprints

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

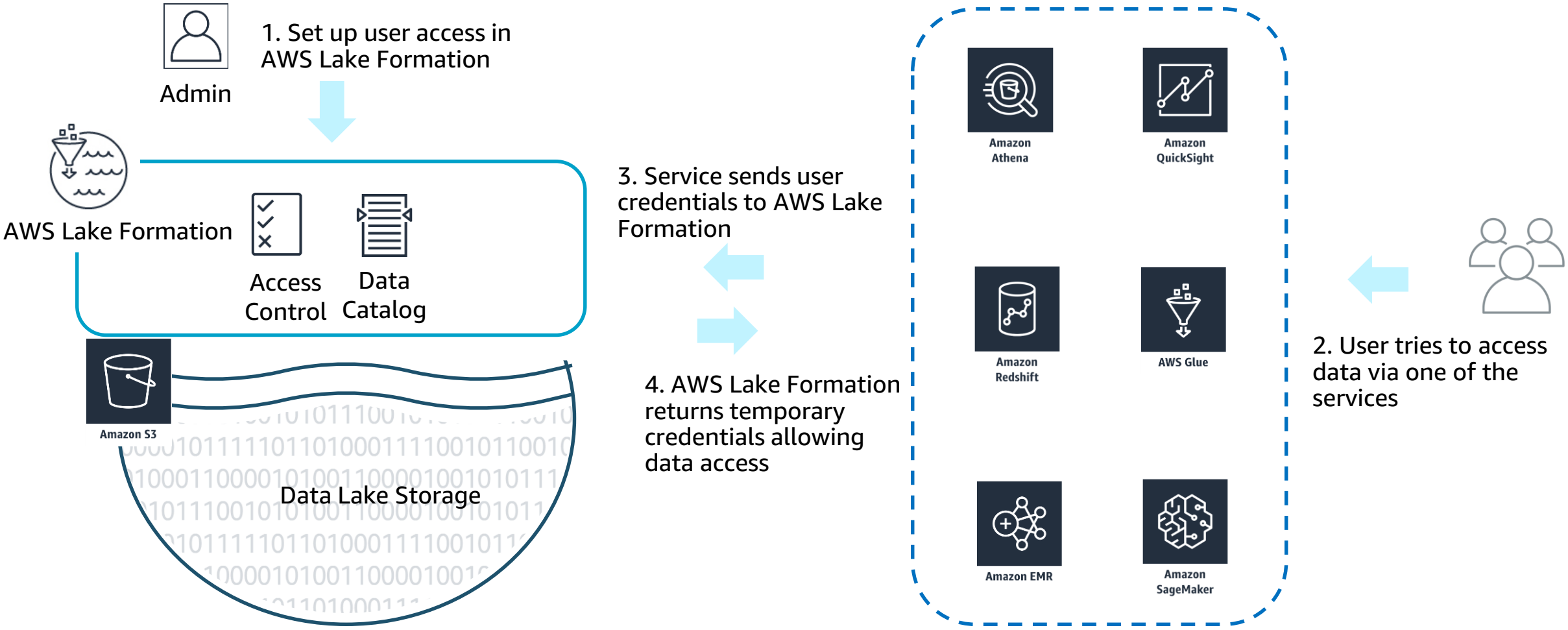
Blueprints build on AWS Glue



Easily de-duplicate your data with ML transforms



Secure once, access in multiple ways



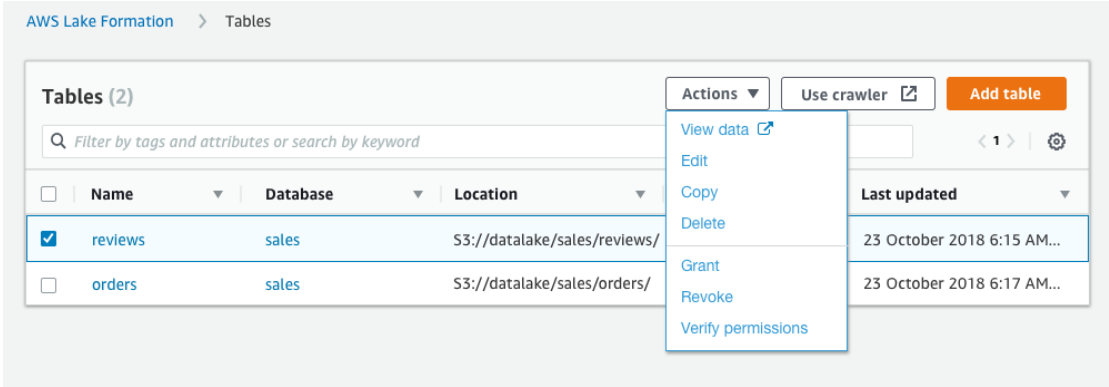
Security permissions in AWS Lake Formation

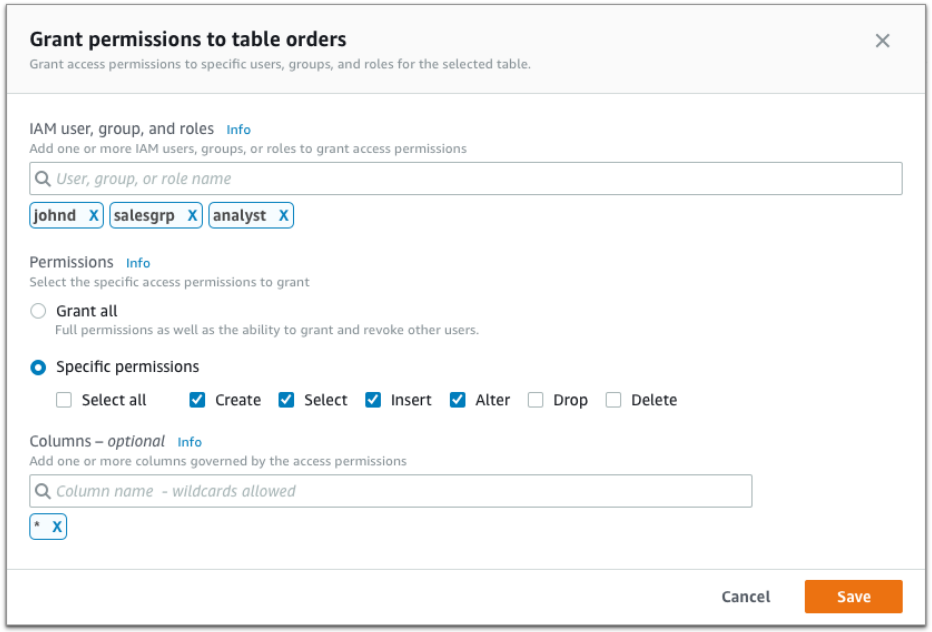
Control data access with simple grant and revoke permissions

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



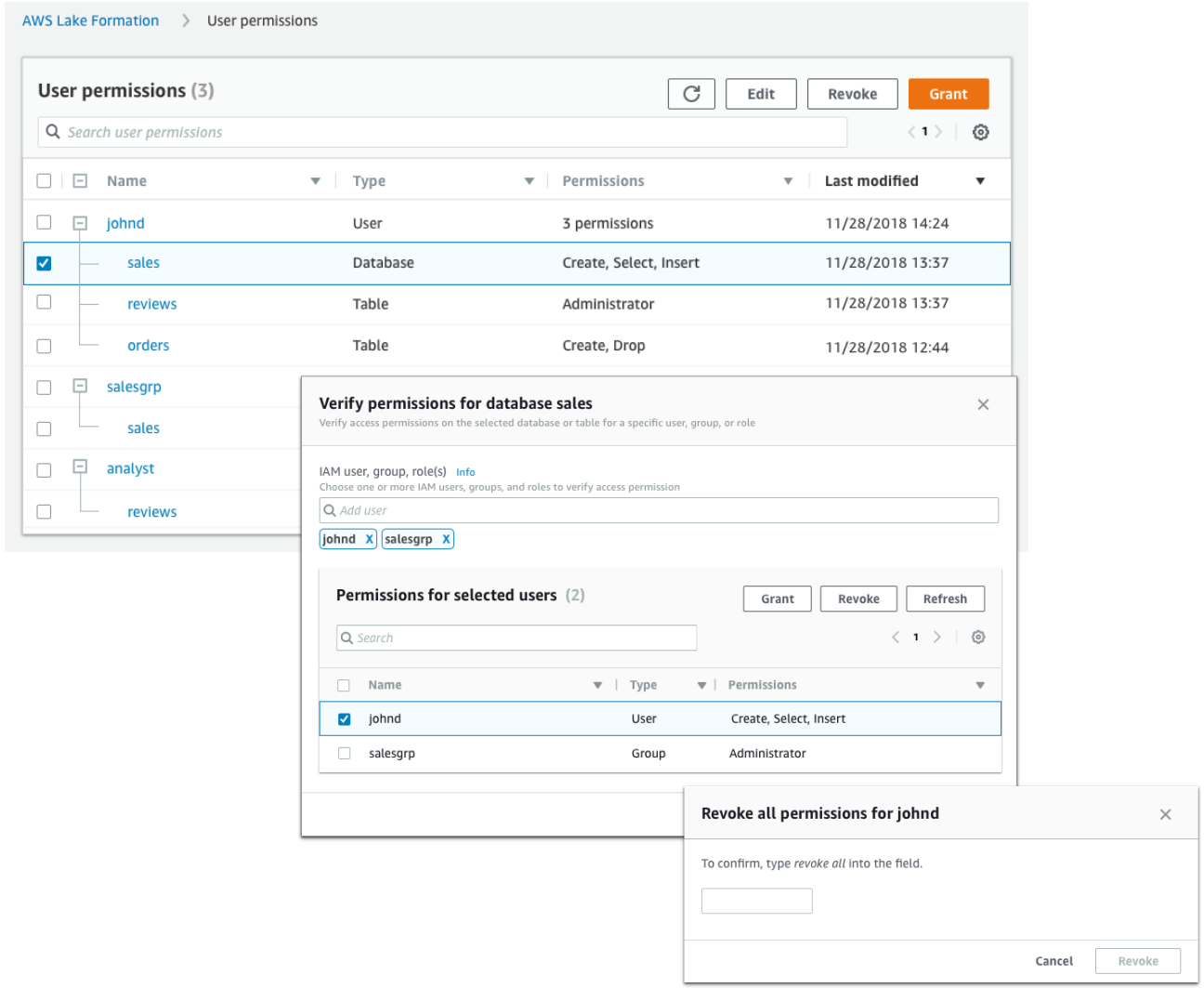


Security permissions in AWS Lake Formation

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

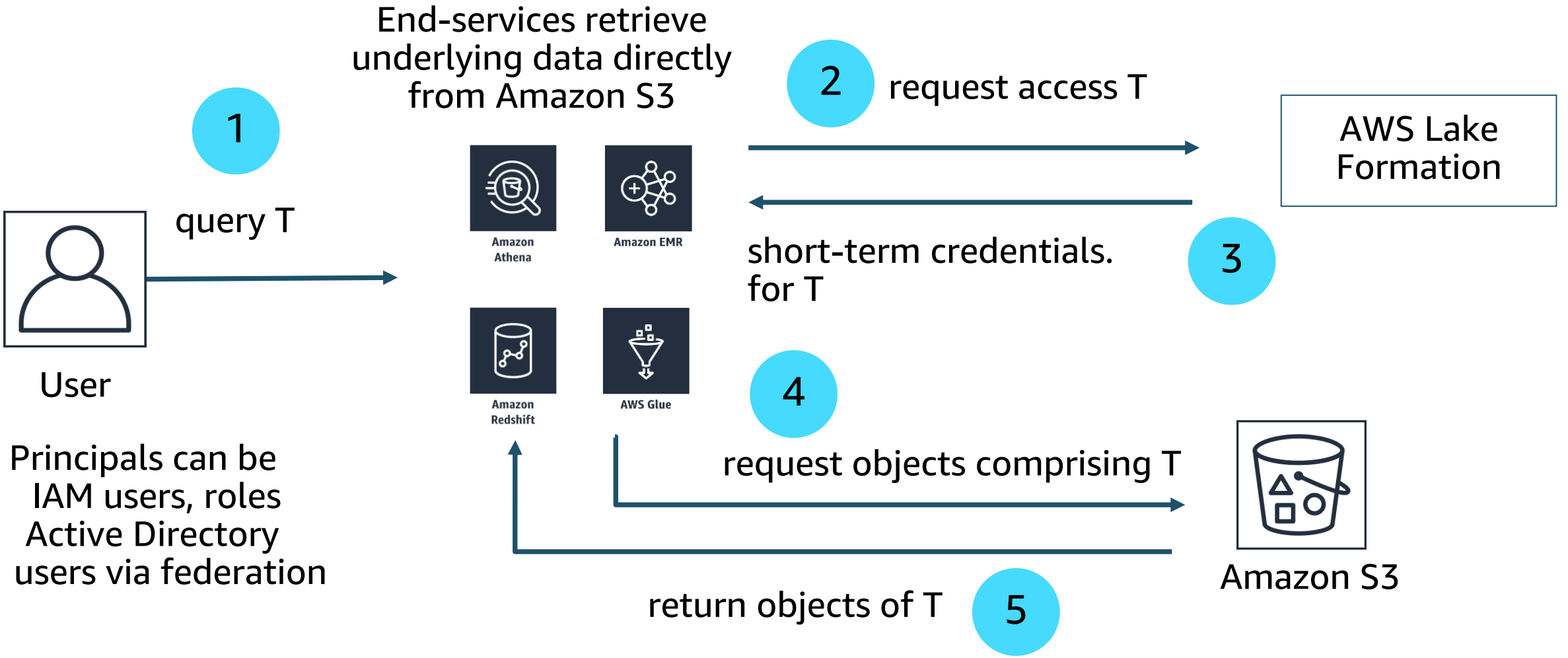
Verify permissions granted to a user

Easily revoke policies for a user



Grant table and column-level permissions

Security – deep dive

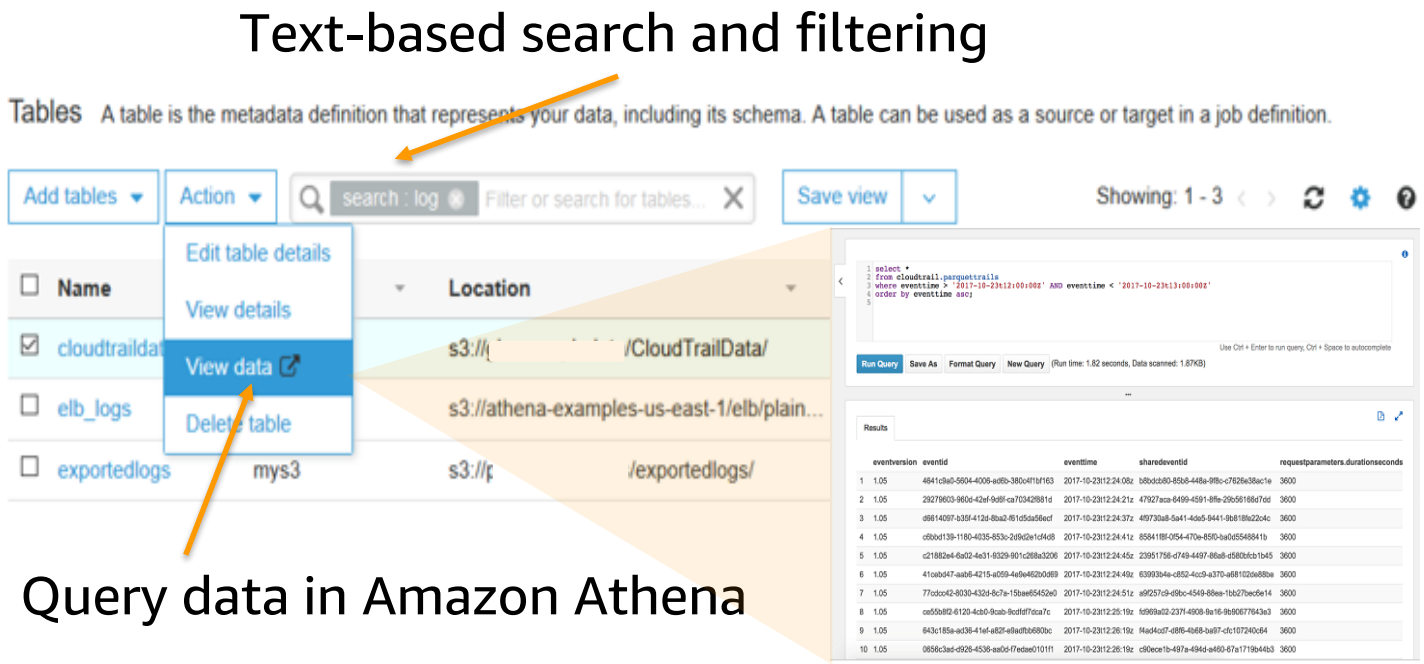


Search and collaborate across multiple users

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

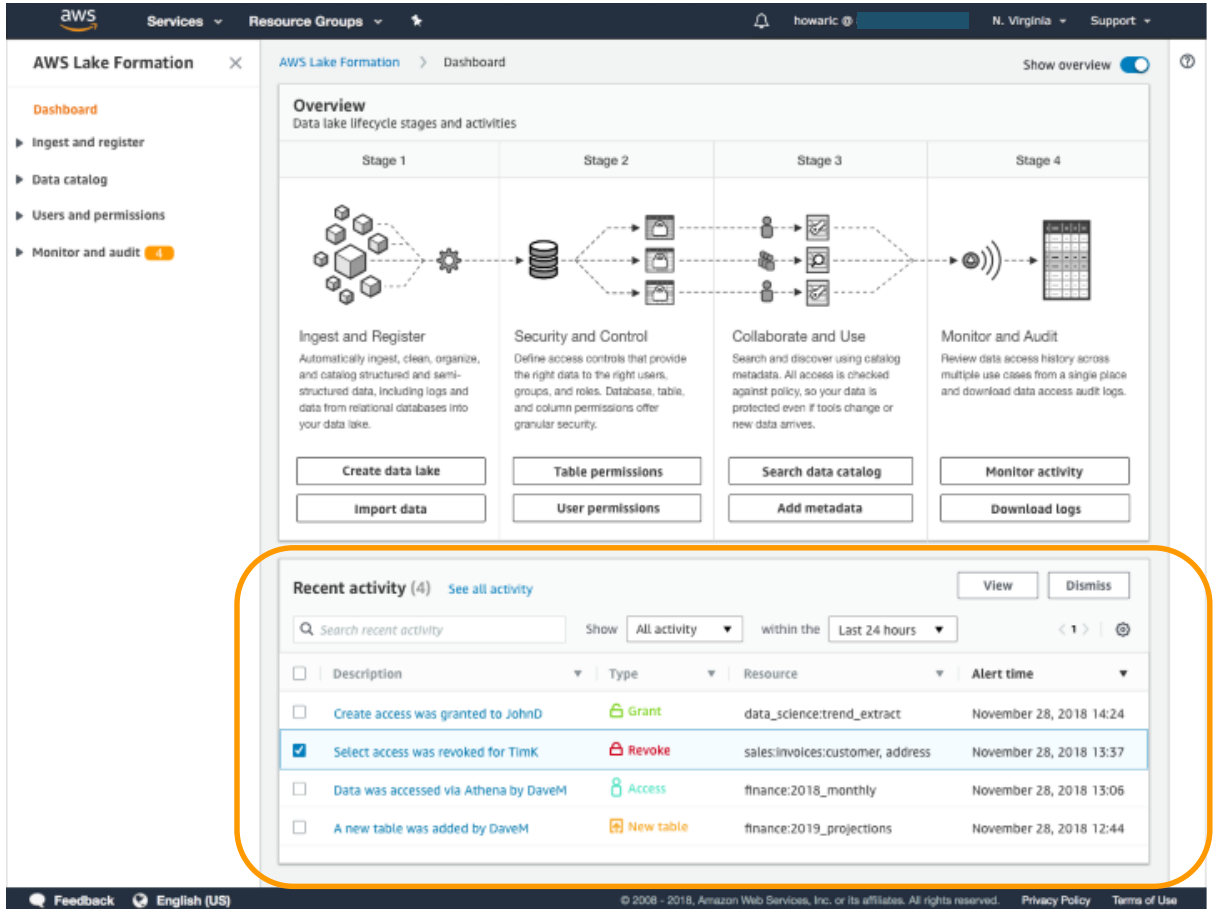


Audit and monitor in real time

See detailed alerts in the console

Download audit logs for further analytics

Data ingest and catalog notifications also published to Amazon CloudWatch events



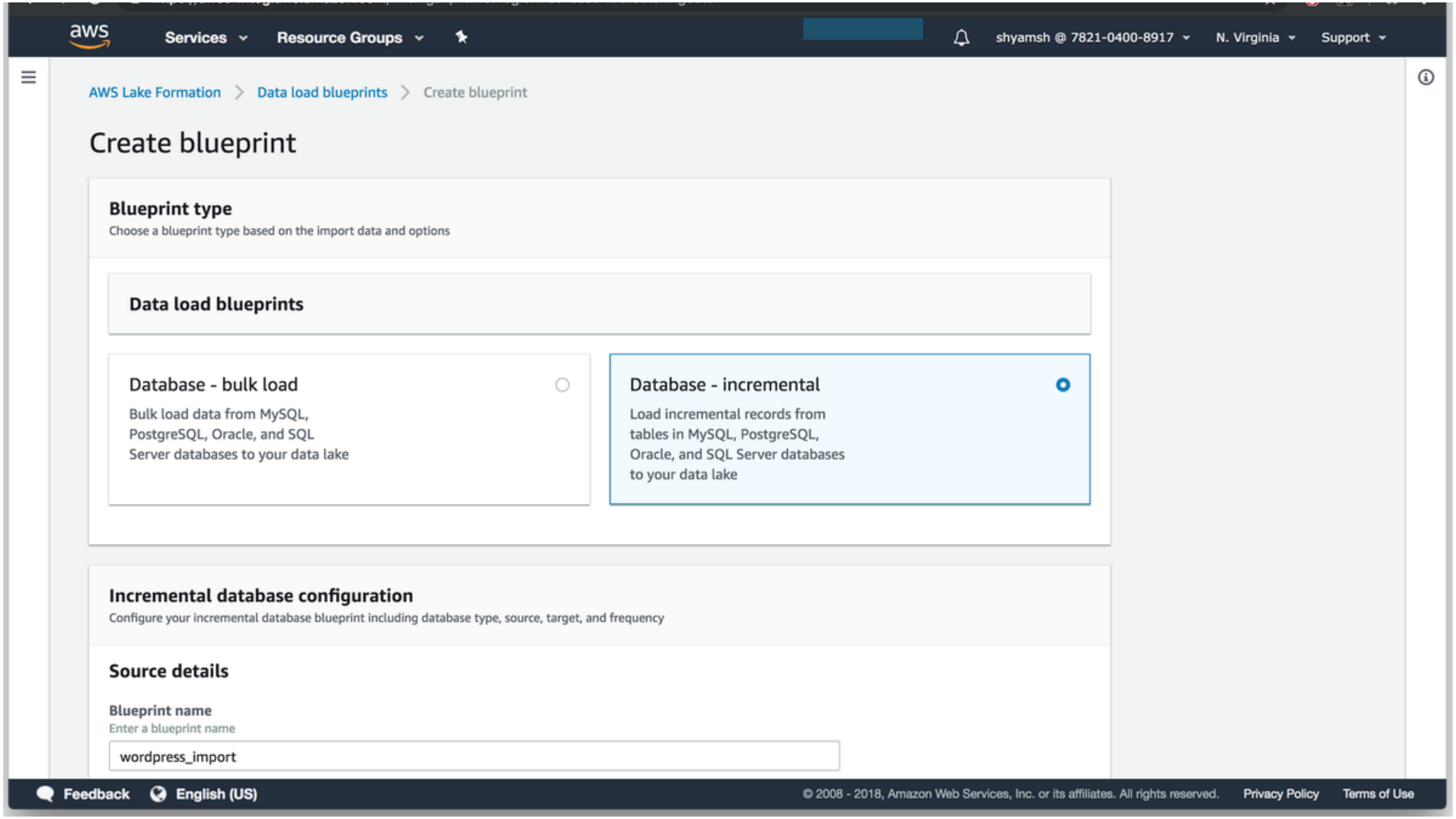
Example: a data lake in 3 easy steps

1. Use blueprints to ingest data

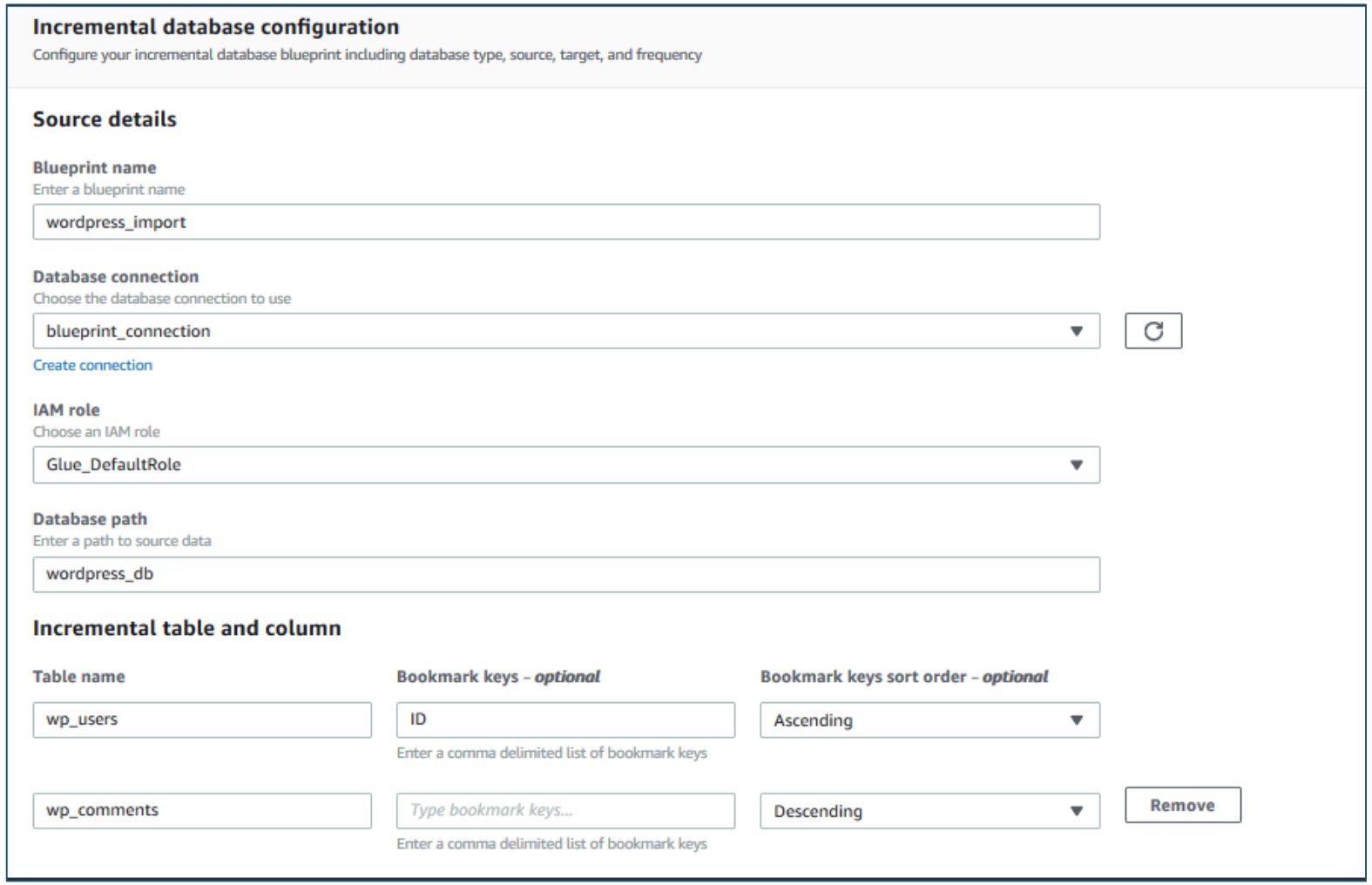
2. Grant permissions to securely share data

3. Query the data (Amazon Athena)

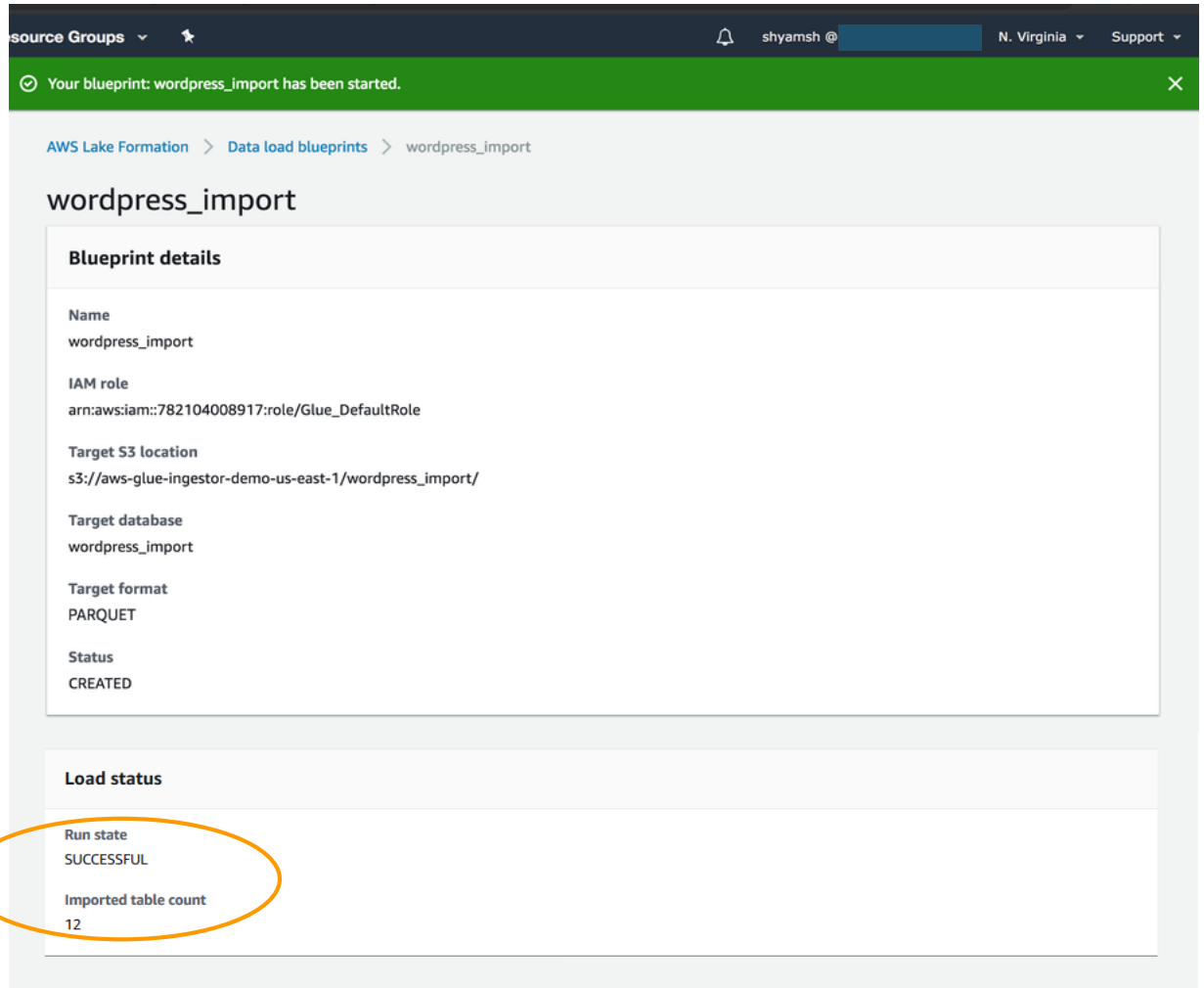
Step 1: Blueprints to ingest data



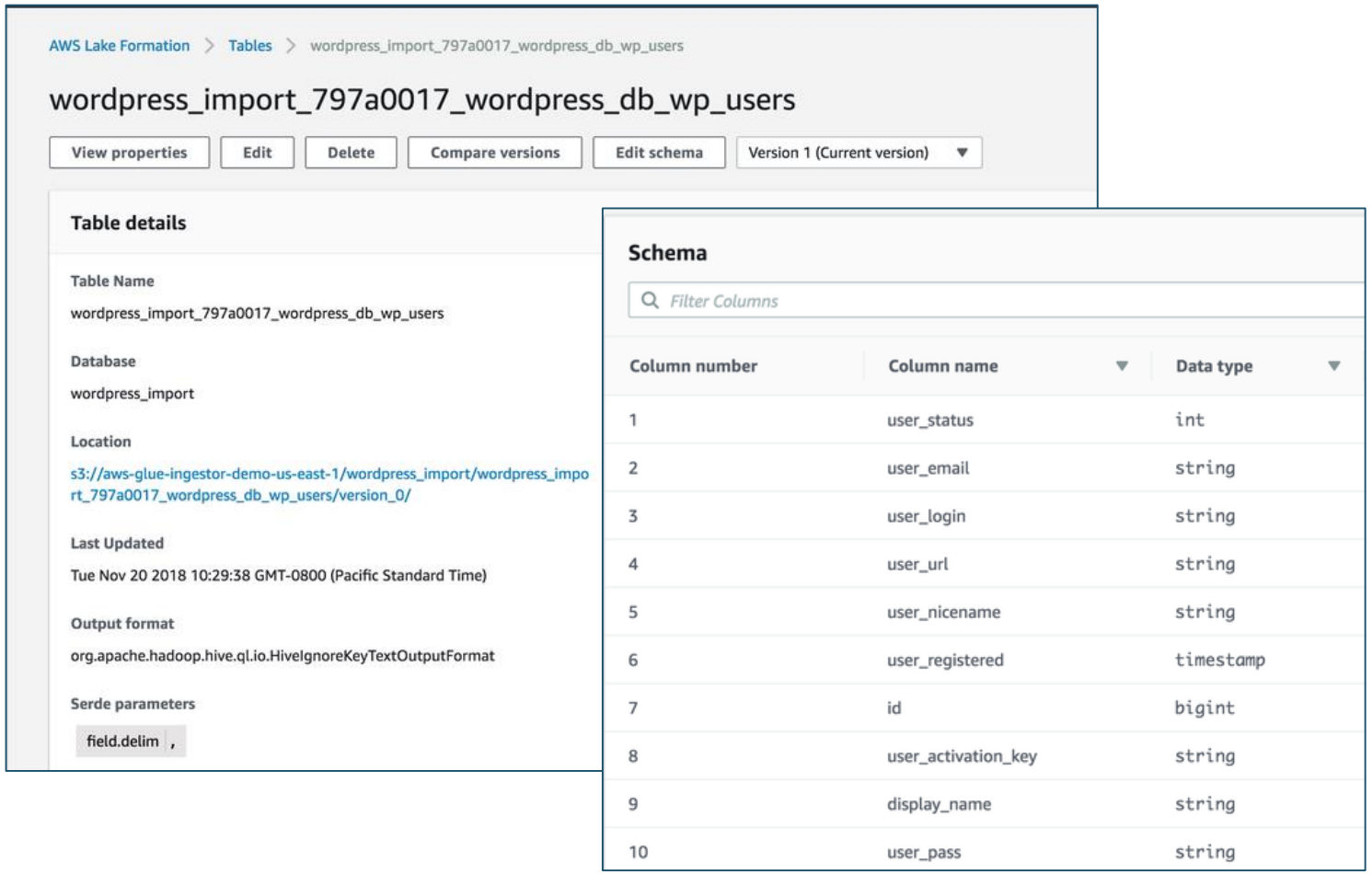
Configure Source…



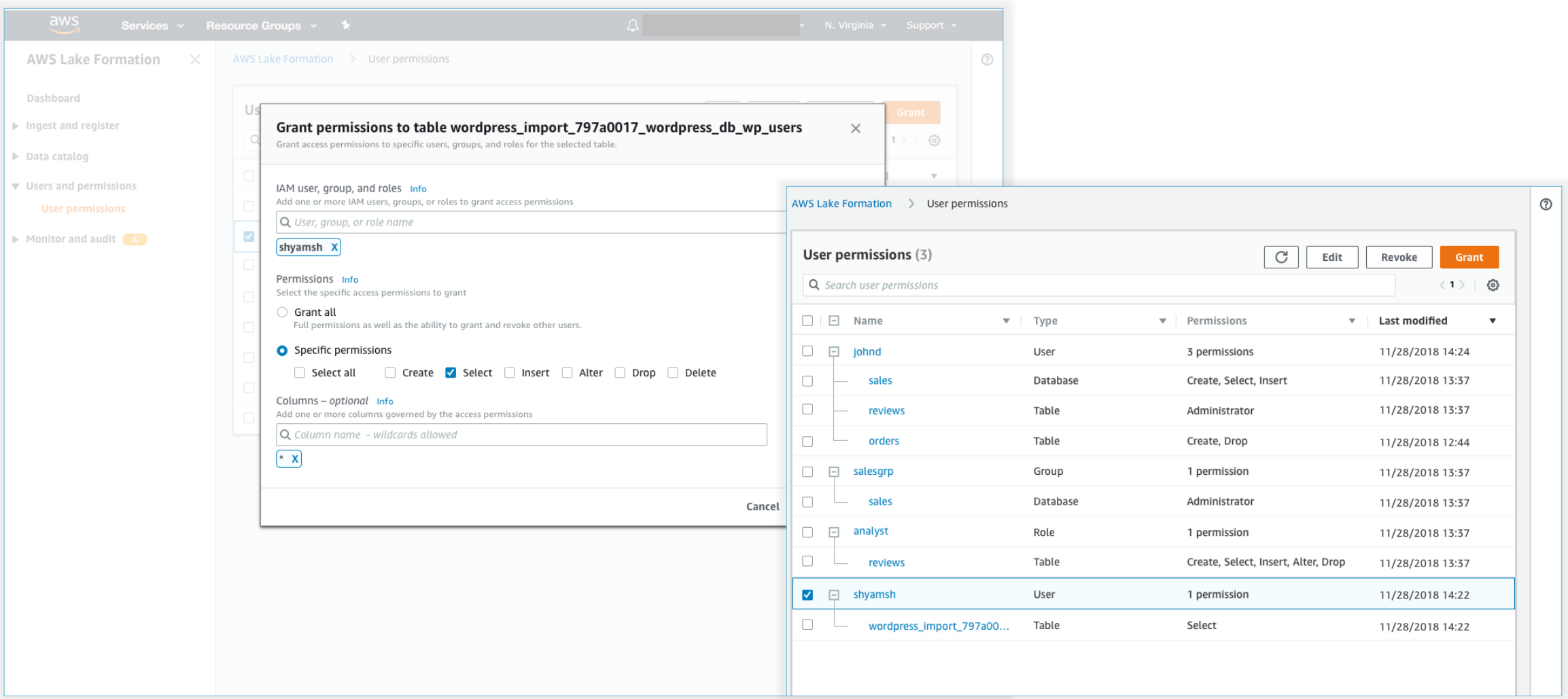
Monitor the import



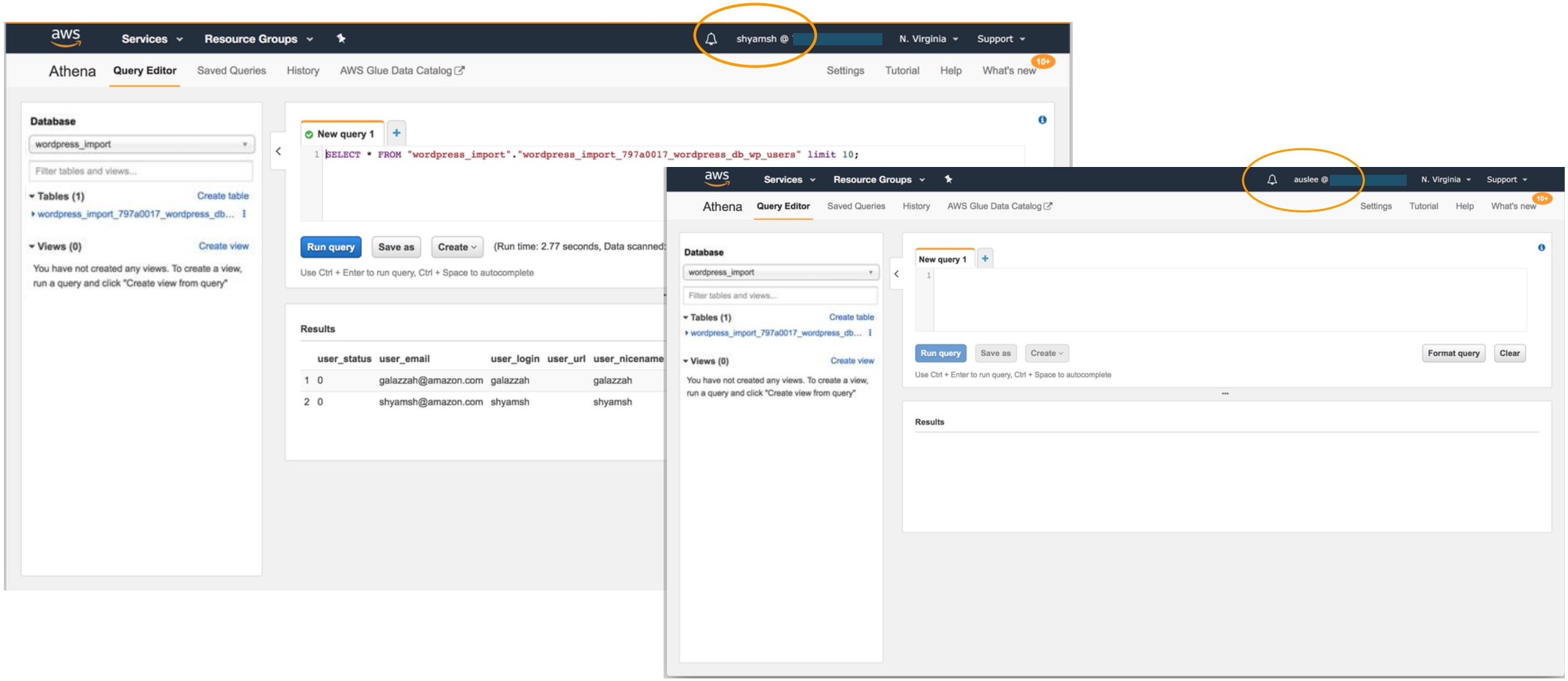
Imported data as table in the data lake



Step 2: Grant permissions to securely share data



Step 3: Run query in Amazon Athena



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/