



Chapter 5

Week 5

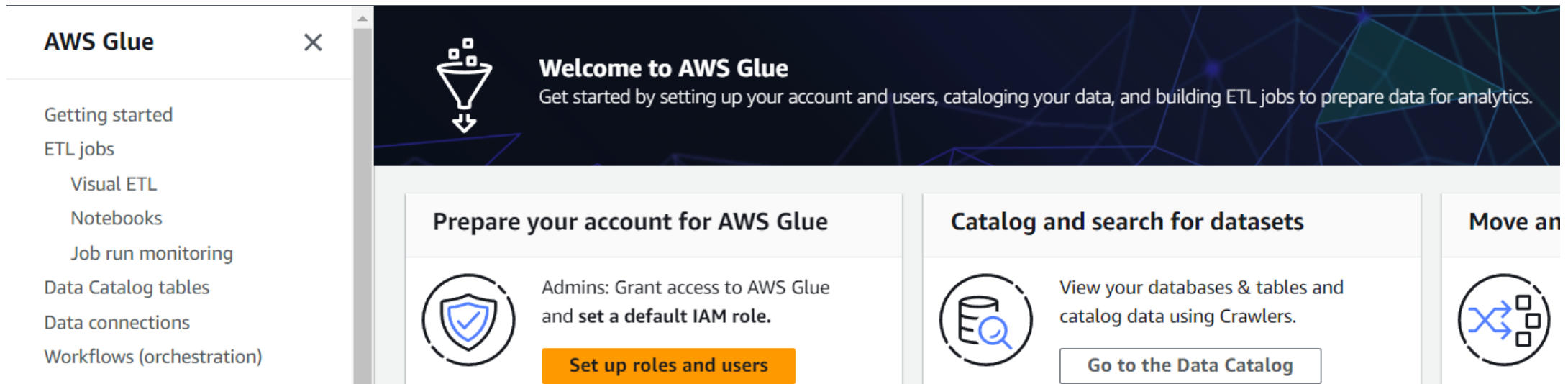
Data Engineer

Trainer: Balazs Balogh



AWS Glue

- AWS Glue is a serverless data integration service that makes it easy for analytics users to discover, prepare, move, and integrate data from multiple sources.



AWS Glue

- > We can create ETL jobs, Databases, Crawlers and so on.
- > We use the databases and crawlers from the offerings.
- > Crawlers are used to populate AWS Glue Data Catalog tables. It goes through the data, and infer it's schema, to make it queryable.

The screenshot displays the AWS Glue console interface. On the left is a navigation sidebar with the following menu items: 'Getting started', 'ETL jobs' (with sub-items 'Visual ETL' and 'Notebooks'), 'Job run monitoring', 'Data Catalog tables', 'Data connections', 'Workflows (orchestration)', 'Data Catalog' (expanded, showing 'Databases', 'Tables', 'Stream schema registries', 'Schemas', 'Connections', 'Crawlers', and 'Classifiers'), and 'Classifiers'. The main content area is titled 'AWS Glue > Crawlers'. Below the title is a description: 'A crawler connects to a data store, progresses through the data, and infers the schema of the data to create tables in the Data Catalog.' There is a search bar labeled 'Filter crawlers'. Below the search bar is a table with 6 crawlers, each with a checkbox, a name, and a state. All crawlers are in a 'Ready' state.

<input type="checkbox"/>	Name	State
<input type="checkbox"/>	community_areas...	✓ Ready
<input type="checkbox"/>	company_crawler	✓ Ready
<input type="checkbox"/>	date_crawler	✓ Ready
<input type="checkbox"/>	payment_type_cr...	✓ Ready
<input type="checkbox"/>	taxi_trips_crawler	✓ Ready
<input type="checkbox"/>	weather_crawler	✓ Ready

AWS Glue - Crawler

➤ Create a new crawler, and choose a name for it.

[AWS Glue](#) > [Crawlers](#) > Add crawler

Step 1
Set crawler properties

Step 2
Choose data sources and classifiers

Step 3
Configure security settings

Step 4
Set output and scheduling

Step 5
Review and create

Set crawler properties

Crawler details [Info](#)

Name

Name can be up to 255 characters long. Some character set including control characters are prohibited.

Description - *optional*

Descriptions can be up to 2048 characters long.

► **Tags - optional**

Use tags to organize and identify your resources.

AWS Glue - Crawler

➤ On the next page, add your data source.

[AWS Glue](#) > [Crawlers](#) > Add crawler

Step 1
[Set crawler properties](#)

Step 2
Choose data sources and classifiers

Step 3
Configure security settings

Step 4
Set output and scheduling

Step 5
Review and create

Choose data sources and classifiers

Data source configuration

Is your data already mapped to Glue tables?

☒ Not yet
Select one or more data sources to be crawled.

Data sources (1) [Info](#)

The list of data sources to be scanned by the crawler.

	Type	Data source
<input type="radio"/>	S3	s3://cubix-chica

► **Custom classifiers - *optional***

A classifier checks whether a given file is in a format the crawler can handle that matches that data format.

AWS Glue - Crawler

- Create an IAM Role which has S3FullAccess, or extend the default one.

[AWS Glue](#) > [Crawlers](#) > Add crawler

Step 1
[Set crawler properties](#)

Step 2
[Choose data sources and classifiers](#)

Step 3
Configure security settings

Step 4
Set output and scheduling

Step 5
Review and create

Configure security settings

IAM role [Info](#)

Existing IAM role

AWSGlueServiceRole-Crawler

▼

↻

View [↗](#)

Create new IAM role

Update chosen IAM role

Only IAM roles created by the AWS Glue console and have the prefix "AWSGlueServiceRole-" can be updated.

Lake Formation configuration - optional

Allow the crawler to use Lake Formation credentials for crawling the data source. [Learn more.](#) [↗](#)

☐ Use Lake Formation credentials for crawling S3 data source

Checking this box will allow the crawler to use Lake Formation credentials for crawling the data source. If the data s

AWS Glue - Crawler

- Choose the target database, if you don't have one, create it here.
- Choose "On demand" for scheduling, you need this to run only once.

[AWS Glue](#) > [Crawlers](#) > Add crawler

Step 1
[Set crawler properties](#)

Step 2
[Choose data sources and classifiers](#)

Step 3
[Configure security settings](#)

Step 4
Set output and scheduling

Step 5
Review and create

Set output and scheduling

Output configuration [Info](#)

Target database

chicago_taxi_db


[Clear selection](#) [Add database](#) 

Table name prefix - *optional*

Type a prefix added to table names

Maximum table threshold - *optional*


This field sets the maximum number of tables the crawler is allowed to error. If not set, the crawler will automatically generate the number of

Type a number greater than 0

► Advanced options

AWS Glue - Crawler

- When it's created, click on "Run crawler" to run it.
- It creates the new table under your database.

 **One crawler successfully created**
The following crawler is now created: "test_crawler"

[AWS Glue](#) > [Crawlers](#) > test_crawler

test_crawler

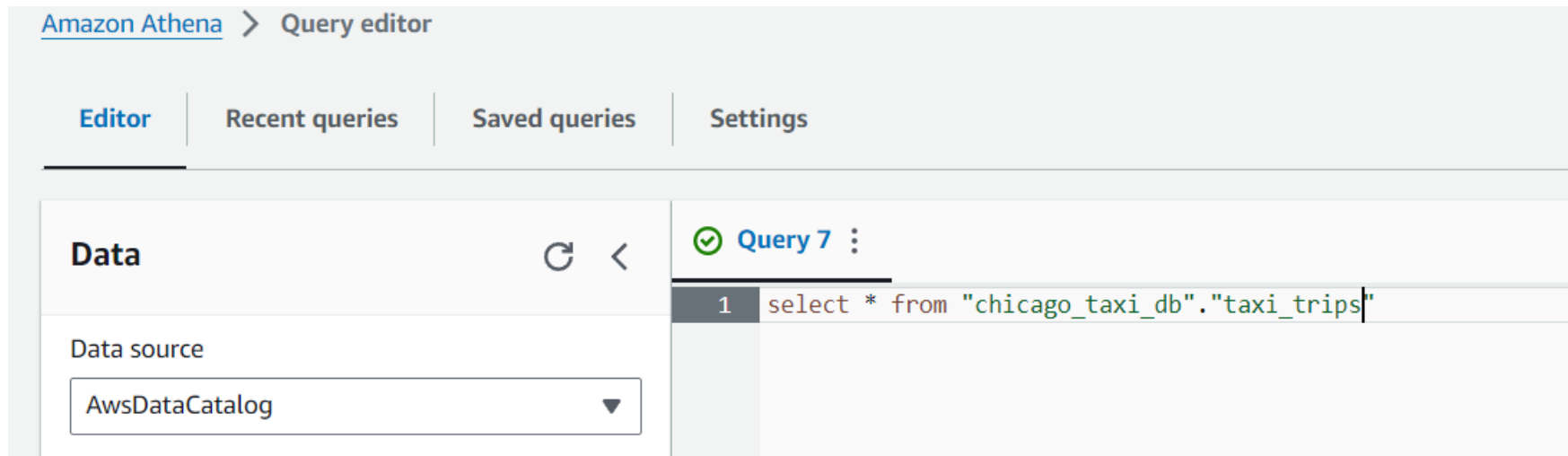
Last updated (UTC)
January 9, 2024 at 20:17:57



Run crawler

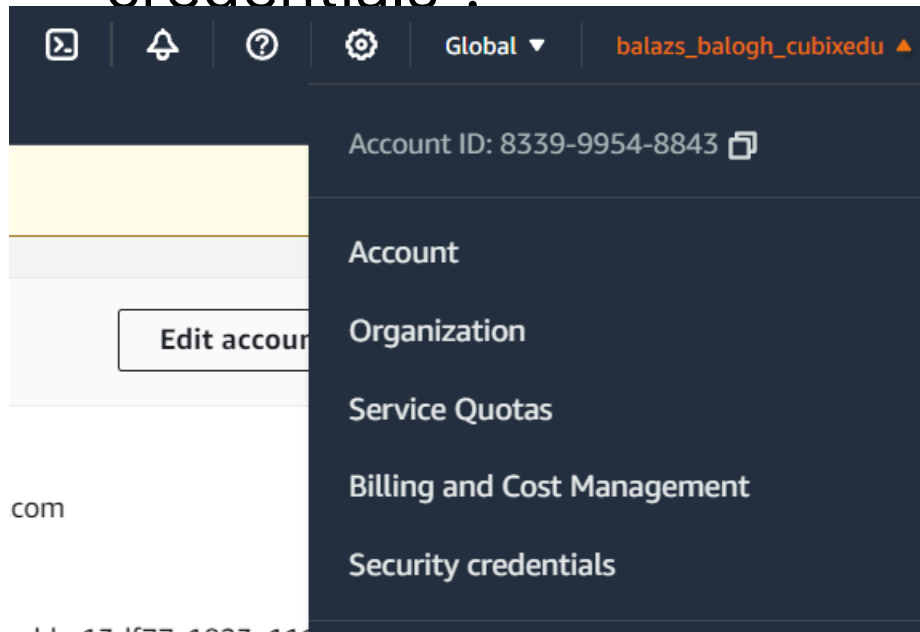
AWS Athena

- Amazon Athena is an interactive query service that makes it simple to analyze data directly in Amazon S3 using standard SQL.
- Athena is serverless, so there is no infrastructure to setup or manage, and you can choose to pay based on the queries you run or compute needed by your queries.



Local visualisations

- For local visualisations, you need an ACCESS_KEY and SECRET_KEY from AWS, to be able to download data from your S3 bucket.
- In the top right corner, click on your account, choose “Security credentials”.



Local visualisations

- > Under Access keys, click on “Create access key” and save your Access key ID, and Secret.

Access keys (2)

Actions ▼

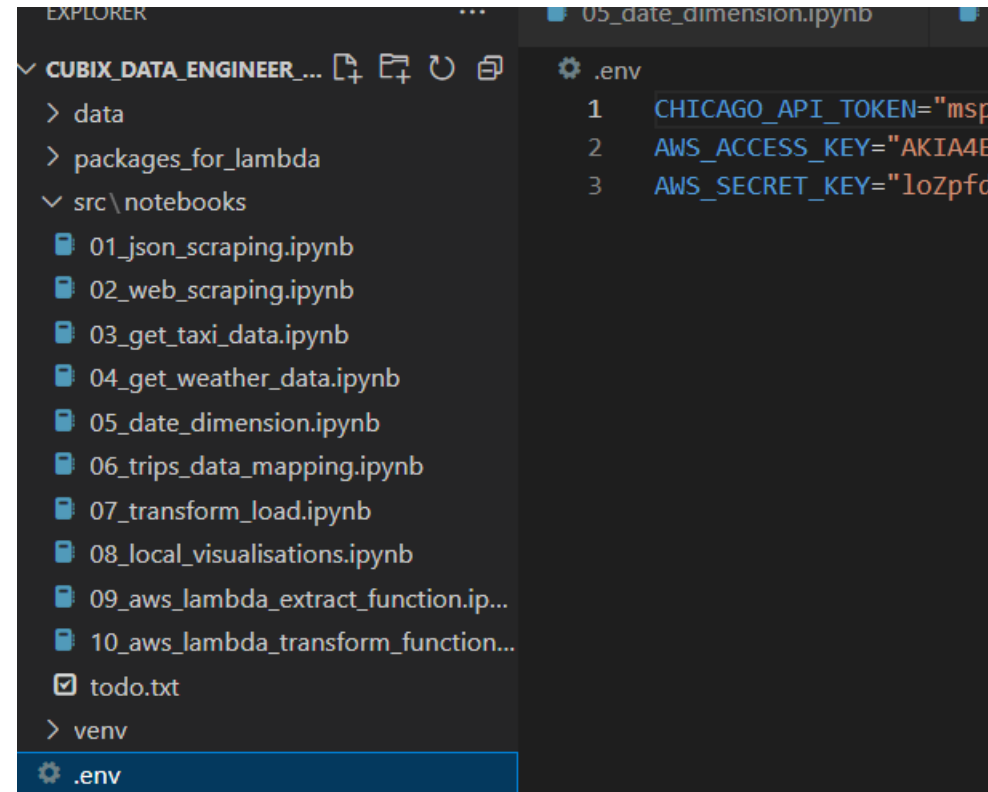
Create access key

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

Access key ID	Created on	Access key last used	Region last used	Service last used	Status
---------------	------------	----------------------	------------------	-------------------	--------

Local visualisations

- > In your root folder create the .env file, if you don't have it already, and add the "AWS_ACCESS_KEY" and "AWS_SECRET_KEY" variables with the keys you just got. You can choose different names for them.



The screenshot shows a code editor interface. On the left, a file explorer pane displays a directory structure for a project named 'CUBIX_DATA_ENGINEER...'. The structure includes folders 'data' and 'packages_for_lambda', and a 'src\notebooks' folder containing ten Jupyter Notebook files (01_json_scraping.ipynb to 10_aws_lambda_transform_function.ipynb), a 'todo.txt' file, and a 'venv' folder. The '.env' file is selected at the bottom of the explorer. On the right, the content of the '.env' file is displayed, showing three lines of configuration: '1 CHICAGO_API_TOKEN="msp...', '2 AWS_ACCESS_KEY="AKIA4E...', and '3 AWS_SECRET_KEY="loZpfc...'.

```
EXPLORER
CUBIX_DATA_ENGINEER...
├── data
├── packages_for_lambda
├── src\notebooks
│   ├── 01_json_scraping.ipynb
│   ├── 02_web_scraping.ipynb
│   ├── 03_get_taxi_data.ipynb
│   ├── 04_get_weather_data.ipynb
│   ├── 05_date_dimension.ipynb
│   ├── 06_trips_data_mapping.ipynb
│   ├── 07_transform_load.ipynb
│   ├── 08_local_visualisations.ipynb
│   ├── 09_aws_lambda_extract_function.ip...
│   ├── 10_aws_lambda_transform_function...
│   ├── todo.txt
│   └── venv
└── .env
```

```
.env
1 CHICAGO_API_TOKEN="msp
2 AWS_ACCESS_KEY="AKIA4E
3 AWS_SECRET_KEY="loZpfc
```

Local visualisations

- Now you can import the libraries, and use the two new environment variables.

```
from io import StringIO
import os

import boto3
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

pd.set_option("display.max_columns", 50)
```

[1]

```
aws_acces_key_id = os.getenv("AWS_ACCESS_KEY")
aws_secret_key = os.getenv("AWS_SECRET_KEY")
```

[2]

Local visualisations

- After reading the data, and joining all the tables together we have everything to make visualisations.
- This is not the usual task for a Data Engineer, it's here just to show you, what happens with the data after you organized it.

```
# 1 - Histogram of Trip Durations

sns.histplot(trips_full['trip_seconds'], bins=30, kde=False)
plt.title('Histogram of Trip Durations')
plt.xlabel('Trip Duration (seconds)')
plt.ylabel('Count')
plt.xlim(0, 5000) # Adjust the x-axis limits for better visibility, clear the outliers
plt.show()
```

