High Performance and Distributed Computing for Big Data

Unit 3: AWS - Using S3 and Lambda

Jordi Mateo Fornés jordi.mateo@udl.cat

Universitat Rovira i Virgili and Universitat de Lleida

Today's lecture

Today’s lecture

- AWS S3 - Storing files in the cloud
    - How to build our own data lake?

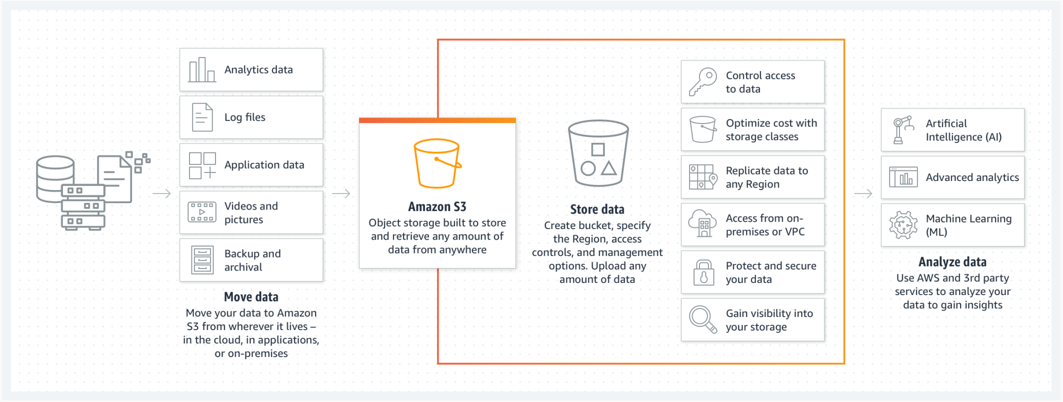
Today’s lecture

- AWS S3 - Storing files in the cloud
    - How to build our own data lake?
- AWS Lambda - Running code in response to events
    - How to automate cell counting?

S3 - Storing files in the cloud

What is S3?

Amazon S3 (Simple Storage Service) is an object storage service that offers industry-leading scalability, data availability, security, and performance. It is designed to store and retrieve any amount of data from anywhere on the web.



Analytics data
Log files
Application data
Videos and pictures
Backup and archival
Move data
Move your data to Amazon S3 from wherever it lives – in the cloud, in applications, or on-premises
Amazon S3
Object storage built to store and retrieve any amount of data from anywhere
Store data
Create bucket, specify the Region, access controls, and management options. Upload any amount of data
Control access to data
Optimize cost with storage classes
Replicate data to any Region
Access from on-premises or VPC
Protect and secure your data
Gain visibility into your storage
Artificial Intelligence (AI)
Advanced analytics
Machine Learning (ML)
Analyze data
Use AWS and 3rd party services to analyze your data to gain insights

S3 - Buckets and Objects

- **Buckets**: In Amazon S3, a bucket is a unique container for objects. Every object is stored within a bucket.
    - The bucket name must be globally unique across all existing bucket names in Amazon S3.
    - Buckets are used to store and organize objects.
- **Objects**: Objects are the fundamental entities within a bucket. They consist of data and metadata.
    - The information within an object is stored as a **key-value** pair.
    - The key, which is a unique identifier, is used to organize objects within the bucket. It is often formatted as a prefix to the object name.

For example, we can create a bucket called hdcb-{your-name} and store objects organized by session or other criteria.

```
hdcb-{your-name}/
    session1/data.csv
    session2/data.csv
```

Terms

- **Key** = *prefix* + *object name*
    - **prefix** = session1/ or session2/
    - **object name** = data.csv

S3 Pricing

Amazon S3 pricing is based on five factors:

1. **Storage Class**: The cost depends on the storage class used (Standard, Intelligent-Tiering, One Zone-IA, etc.).
2. **Storage**: The total volume of data stored per month.
3. **Requests**: The number and type of requests made.
4. **Data Transfer**: The cost of transferring data can vary by region and is also affected by whether data is transferred in or out.
5. **Management & Replication**: Additional features like data replication or management operations can also affect the cost.

For detailed information, you can refer to the Amazon S3 Pricing Page.

Creating a S3 bucket

1. Go to the S3 service.
2. Click on Create bucket.
3. Fill the form with the following settings:
    - Name: hdcb-{your-name}
    - Region: US East (N. Virginia)
    - Mark Genereal purpose as the use case
    - ACLs: Disable - All objects are owned by the bucket owner. (Recommended for most use cases)
    - Keep block all public access on. This ensures that only the bucket owner can access the objects in the bucket.
    - Keep bucket versioning off. This means that multiple versions of an object will not be kept in the bucket.
    - Keep server-side encryption with Amazon S3-managed keys (SSE-S3)
    - Keep bucket keys enabled.

The bucket url will be: **s3://hdcb-{your-name}**.

Uploading a file to the bucket

1. Click on the bucket.
2. Create a folder called **session2**.
3. Click on the folder.
4. Click on Upload.
5. Click on Add files and select the file.
6. Upload the files data.csv and metadata.txt (take the files from the virtual campus).
7. Click Upload.

Our bucket (hdcb-{your-name}/session2) contains two files: data.csv and metadata.txt. The files are stored in **s3://hdcb-{your-name}/session2**. This files are private by default. Only the owner can access them.

Accessing the file from the notebook

Let’s access the file using the boto3 library. This library allows Python developers to write software that makes use of services like Amazon S3 on AWS.

```
!pip install boto3
!pip install pandas
import boto3
import pandas as pd
from botocore import UNSIGNED
from botocore.client import Config
session = boto3.Session()
s3 = boto3.client('s3',
    region_name='us-east-1',
    config=Config(signature_version=UNSIGNED))
s3.download_file('hdcb-jordi', 'session2/data.csv', 'data.csv')
df = pd.read_csv('data.csv')
print(df)
```

Adding credentials to the EC2 instance

1. Copy the *aws_access_key_id* and the *aws_secret_access_key* from the AWS Educate platform and save them in the EC2 instance.

```
aws configure
# paste the credentials
# region: us-east-1
# keep the default output format

# add the token
aws configure set aws_session_token
# paste the token
```

2. Restart the jupyter and restart the kernel.

3. Run the code again.

```
!pip install boto3
!pip install pandas
import boto3
import pandas as pd
from botocore import UNSIGNED
from botocore.client import Config
session = boto3.Session()
s3 = boto3.client('s3',
    region_name='us-east-1')
s3.download_file('hdcb-jordi',
    'session2/data.csv', 'data.csv')
df = pd.read_csv('data.csv')
print(df)
```

RECAP: Accessing the file from the notebook

To access the file from the notebook we have used our aws-credentials for the current user (owner of the bucket).

RECAP: Accessing the file from the notebook

To access the file from the notebook we have used our aws-credentials for the current user (owner of the bucket).

Be aware!
Someone could log into your jupyter instance in the browser and access the file using your credentials.

RECAP: Accessing the file from the notebook

To access the file from the notebook we have used our aws-credentials for the current user (owner of the bucket).

Be aware!
Someone could log into your jupyter instance in the browser and access the file using your credentials.

What to do?
In real life, we would use IAM roles to give the notebook the necessary permissions to access the file. **But, in AWS Educate, we can not use IAM roles**.

RECAP: Accessing the file from the notebook

To access the file from the notebook we have used our aws-credentials for the current user (owner of the bucket).

Be aware!
Someone could log into your jupyter instance in the browser and access the file using your credentials.

What to do?
In real life, we would use IAM roles to give the notebook the necessary permissions to access the file. **But, in AWS Educate, we can not use IAM roles**.

Another option
Make the file public and access it without credentials :)

Exercise: Sync our local directory with the bucket

1. Install the AWS CLI on your local machine.
    - Follow the instructions on the official AWS documentation to install the AWS CLI.
2. Configure the AWS CLI with your credentials.
    - Run aws configure in your terminal and enter your AWS Access Key ID, Secret Access Key, and default region when prompted.
3. Sync your local directory with the S3 bucket.
    - Navigate to the directory you want to sync with the S3 bucket.
    - Run the command aws s3 sync . s3://hdcb-jordi/session2 to sync the files in the current directory with the files in the bucket.

Exercise: Deploy a Static Website on S3

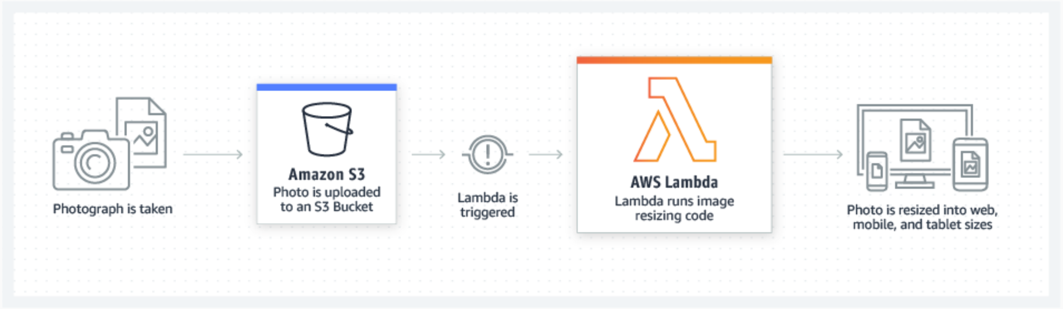
1. Create a new S3 bucket.
2. Upload an HTML file.
3. Make the HTML file public.
4. Enable static website hosting.
5. Access your website.

~~~
```html
<!DOCTYPE html>
<html>
<body>
    <h1>Hello, World!</h1>
    <p>Welcome to my website hosted on Amazon S3!</p>
</body>
</html>
```
~~~

AWS Lambda

What is AWS Lambda?

AWS Lambda is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you. You can use AWS Lambda to extend other AWS services with custom logic, or create your own back-end services that operate at AWS scale, performance, and security.



Photograph is taken
Amazon S3
Photo is uploaded to an S3 Bucket
Lambda is triggered
AWS Lambda
Lambda runs image resizing code
Photo is resized into web, mobile, and tablet sizes

Image processing with AWS Lambda

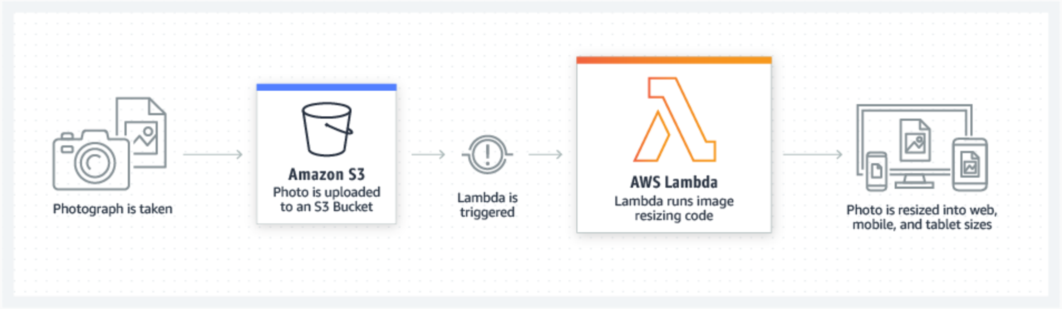

Photograph is taken
Amazon S3
Photo is uploaded to an S3 Bucket
Lambda is triggered
AWS Lambda
Lambda runs image resizing code
Photo is resized into web, mobile, and tablet sizes

Figure 1: Overview of image processing with AWS Lambda

Use Cases of AWS Lambda in Biomedicine

What use case do you think is the most interesting for you? Or your work?

Use Cases of AWS Lambda in Biomedicine

What use case do you think is the most interesting for you? Or your work?

- **Real-time data processing**: Lambda can analyze sensor data from wearable devices to identify irregular patterns.

Use Cases of AWS Lambda in Biomedicine

What use case do you think is the most interesting for you? Or your work?

- **Real-time data processing**: Lambda can analyze sensor data from wearable devices to identify irregular patterns.
- **Data aggregation**: Lambda can collect data from multiple clinical trial sites, aggregating information on patient outcomes, adverse events, and treatment efficacy preparing a dashboard for the researchers.

Use Cases of AWS Lambda in Biomedicine

What use case do you think is the most interesting for you? Or your work?

- **Real-time data processing**: Lambda can analyze sensor data from wearable devices to identify irregular patterns.
- **Data aggregation**: Lambda can collect data from multiple clinical trial sites, aggregating information on patient outcomes, adverse events, and treatment efficacy preparing a dashboard for the researchers.
- **Data validation**: Lambda can validate lab results (such as blood tests) by checking for outliers or inconsistencies. For example, abnormal values outside the expected range may require further investigation.

Use Cases of AWS Lambda in Biomedicine

What use case do you think is the most interesting for you? Or your work?

- **Real-time data processing**: Lambda can analyze sensor data from wearable devices to identify irregular patterns.
- **Data aggregation**: Lambda can collect data from multiple clinical trial sites, aggregating information on patient outcomes, adverse events, and treatment efficacy preparing a dashboard for the researchers.
- **Data validation**: Lambda can validate lab results (such as blood tests) by checking for outliers or inconsistencies. For example, abnormal values outside the expected range may require further investigation.
- **Image processing**: Lambda can process images to identify patterns or anomalies.

Cell Counting Hands-on

What is the cell counting task?

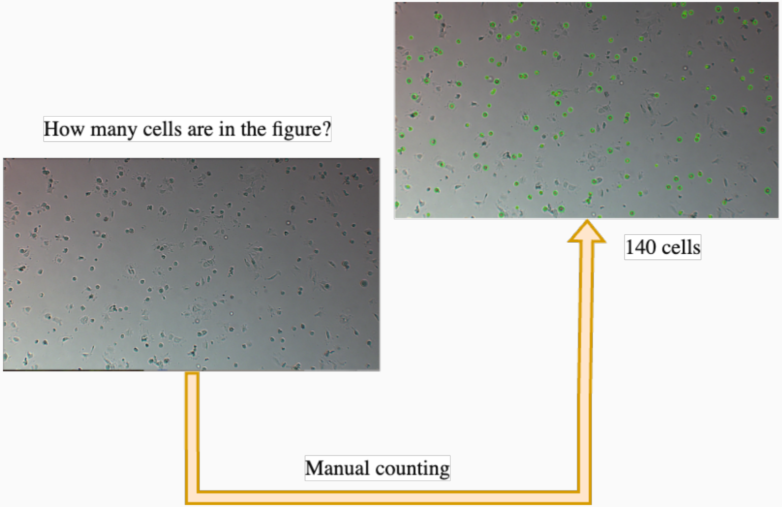


Figure 2: Overview of cell counting task

What is a common cell counting scenario?

Overview

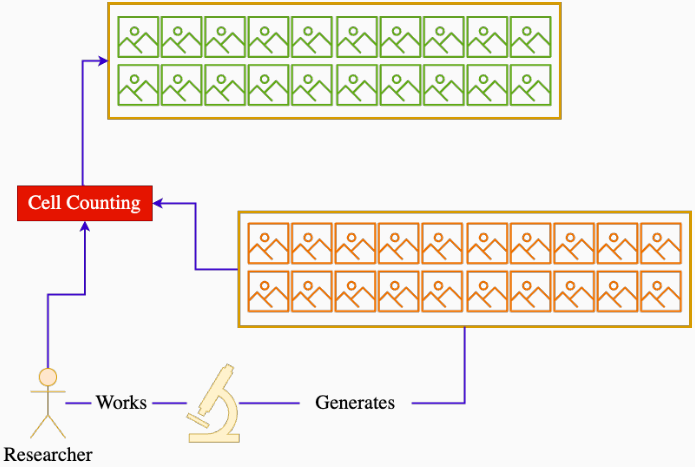


Figure 3: Cell counting

Scenario

- Let's assume that you work as a researcher in a lab.
- Most of the experiments you do involve counting cells in images.
- At the end of the day, you have a lot of images and you need to count the cells in the images.

Challenges

- Large volume of images to analyze
- Time-consuming, tedios, repetitive task
- Potential for human error

Cell Counting script in Python

```python
# Load the image
image = cv2.imread('image.png')
# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# Apply Gaussian blur
blurred = cv2.GaussianBlur(gray, (5, 5), 0)
# Apply Hough transform
circles = cv2.HoughCircles(blurred, cv2.HOUGH_GRADIENT, dp=1
minDist=20, param1=50, param2=30, minRadius=5, maxRadius=30)
# Count the cells
print('Cell count:', len(circles))
```

Cell Counting with AWS Lambda

Solution

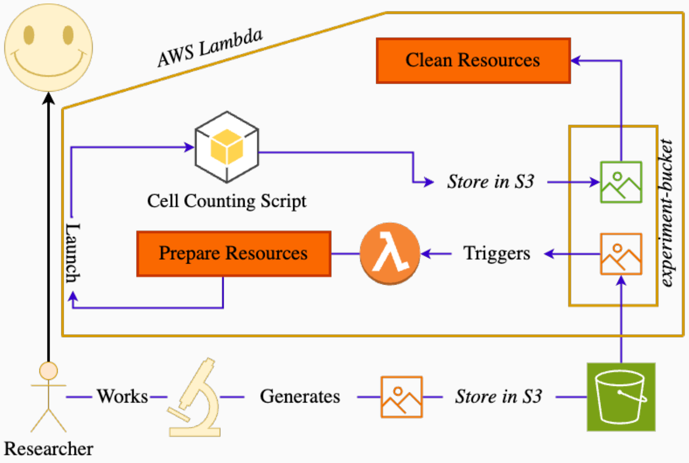


Figure 4: Cell Counting with AWS Lambda

Steps

1. Upload the images generated in the lab to an S3 bucket.
2. Create a cell counting script.
3. Configure AWS Lambda to run the script when a new image is uploaded to the bucket.

Benefits

- Lambda scales automatically and prepares/cleans up the environment for you.
- You pay only for the compute time you consume.

Requirements

- We need to create two S3 buckets:
    - cell-counting-{your_name}-raw-images: to store the images generated in the lab.
    - cell-counting-{your_name}-results: to store the results of the cell counting task.

Requirements

- We need to create two S3 buckets:
  - cell-counting-{your_name}-raw-images: to store the images generated in the lab.
  - cell-counting-{your_name}-results: to store the results of the cell counting task.
- We need to create a Docker image with the cell counting script and the necessary libraries (OpenCV, numpy, etc).

Requirements

- We need to create two S3 buckets:
    - cell-counting-{your_name}-raw-images: to store the images generated in the lab.
    - cell-counting-{your_name}-results: to store the results of the cell counting task.
- We need to create a Docker image with the cell counting script and the necessary libraries (OpenCV, numpy, etc).

What is a Docker image?

Dockerizing the cell counting script

1. Create a project folder called cell-counting.
    - lambda_function.py
    - Dockerfile
    - requirements.txt
    - entrypoint.sh
2. Create the image using the following command:

```
docker build -t hdbc-cell-counting .
```

3. To check the code, you can see this github repository: cell-counting.

Creating a AWS ECR repository

1. Go to the Elastic Container Registry (ECR) service.
2. Click on Create repository.
3. Fill the form with the following settings:
    - Repository name: hdbc-cell-counting
    - Keep the default settings

Storing the Docker image in the AWS Elastic Container Registry (ECR)

1. Log in using the AWS CLI.

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS \
--password-stdin 654654389363.dkr.ecr.us-east-1.amazonaws.com
```

2. Tag the image.

```
docker tag hdbc-cell-counting:latest \
 654654389363.dkr.ecr.us-east-1.amazonaws.com hdbc-cell-counting:latest
```

3. Push the image to the ECR.

```
docker push 654654389363.dkr.ecr.us-east-1.amazonaws.com/hdbc-cell-counting:latest
```

Lambda creation

1. Go to the Lambda service.
2. Click on Create function.
3. Marck Container image.
4. Fill the form with the following settings:
    - Function name: cell-counting
    - Container Image URI: Browse images and select the cell-counting image.
    - Architecture: arm64
    - Change the default settings:
        - Execution role: LabRole
    - Keep the default settings
    - Check the box to acknowledge.
5. Click on Create function.

Triggering the Lambda function

1. Click on the cell-counting function.
2. Click on Add trigger.
3. Select the S3 trigger.
4. Fill the form with the following settings:
    - Bucket: cell-counting-{your_name}-raw-images
    - Event type: PUT
    - Keep the default settings
5. Edit the environment variables:
    - MPLCONFIGDIR: /tmp/matplotlib
    - DST_BUCKET: cell-counting-{your_name}-results
6. Edit settings:
    - Increase the memory to 1024 MB
    - Increase the timeout to 1 m 0s
    - Increase the storage to 512 MB

Testing the Lambda function

1. Go to the S3 service.
2. Click on the **cell-counting-{your_name}-raw-images** bucket.
3. Click on Upload.
4. Upload the image *cell_image.jpg*.
5. Go to the **cell-counting-{your_name}-results** bucket.
6. Click on the *cell_image_processed.jpg* file.
7. Click on the file to see the content.

Conclusions

Recap

- We have learned how to store files in the cloud using S3. We have created a bucket and uploaded files to the bucket.
- We have learned how to access the files from the notebook and how to give the EC2 instance the necessary permissions to access the files.
- We have also learned how to run code in response to events using AWS Lambda.
- We have automated the cell counting task using AWS Lambda.

Final discussion

What do you think about the potential of
AWS Lambda to automate your workloads?