# AWS Essentials – Regular Exam

A client wants you to provide a service that processes files. The allowed extensions are .pdf, .jpg and .png. In the event of an upload of another file extension (e.g. .txt, .docx, .xlsx), an error notification must be sent to the client.

You need to provide them with a simple web interface which has only a button, which chooses a file to upload. This web interface must be accessible from a public IPv4 address. You can configure the instance by hand using the console. After uploading this file, a process must be started automatically. The client wants to have file metadata stored in a NoSQL database. For each file upload, they need an entry in the DB with the file size, file extension and date of upload. They must be able to retrieve all entries from the DB with a specified file extension within milliseconds.

When the DB entry is successfully entered, the client needs to receive an E-Mail notification with data about the object (file extension, file size and date of upload).

The files must be stored for 30 minutes, after which they must be deleted automatically.

For future planning, explain in free text how the web interface can be made more available and scalable in the event of more traffic. Explain briefly what type of scaling they will need for this use case.

The client's E-Mail address is [hristo.zhelev@yahoo.com](mailto:hristo.zhelev@yahoo.com).

You can assume that 20% of the file uploads will be in the wrong format.

The client wants the project in a public **GitHub repository**, **including a CI / CD pipeline** that ensures the quality of the code after each push to the master branch.  
This repository should include tests for the stack. Add meaningful logs wherever possible to facilitate debugging.

Document your project as follows:

## 1. Architecture:

1. Describe the architecture and explain why you selected these services.
2. Create an architectural diagram using [https://draw.io](https://draw.io/) which illustrates the relationships between services and resources.
3. Calculate the monthly cost for the services under the following conditions:
   * Region: eu-central-1.
   * 15 000 000 files uploaded per month (excluding the free tier).

## 2. Implementation Documentation:

1. Describe the infrastructure. Explain what AWS Services must be used for each feature.
2. Describe the steps for building the infrastructure. Include details about the configuration of the web interface.
3. Describe the safety precautions that need to be met for a secure access to AWS. (IAM)
4. Describe the process of creating the needed resources and how they're connected.
5. Document any errors encountered and their solutions. (If you face any)

## Criteria:

1. **Correct Services (5×4 points = 20 points):**
   * AWS Lambda, EC2, DynamoDB, SES/SNS, CloudWatch.
2. **Properly Created Resources (4×10 points = 40 points):**
   * 10 points each for:
     + Correct EC2 web interface with ssh access and a nginx server running and accessible publicly.
     + DynamoDB Table with a secondary index for the file extension attribute.
     + An S3 Bucket with a TTL configuration.
     + A Lambda Function that is being triggered on S3 uploads.
     + A Lambda Function that is being triggered on new DynamoDB entries, which sends E-Mails.
     + SNS Topic + Subscription.
3. **Orchestration (5×6 points = 30 points):**
   * Correct integration of services:
     + EC2 → S3
     + S3 → Lambda 1
     + Lambda 1 → DynamoDB entry
     + DynamoDB → Lambda 2
     + Lambda 2 → SNS
4. **Documentation (20 points):**
   * Describe the safety measures that must be configured. (IAM Users, Tokens, etc.)
   * SSH connection to the web interface.
5. **Bonus (20 points):**
   * Reusable CDK resources for future development and futureproofing.
   * Proper and readable diagrams.
   * Useful logs.
   * Understandable commit messages in the repository.
6. **Cost Calculation (20 points):**
   * With details and breakdown by service.
   * **Total:**  
     20 + 40 + 30 + 20 + 10 = 130 (+ Bonus 20) = 150

## Grades:

|  |  |  |
| --- | --- | --- |
| **Grade** | **Points** | **Explanation** |
| 2 | 0 - 49 | Insufficient understanding of AWS basics. |
| 3 | 50 - 74 | Understanding of basics but lacks complex connections. |
| 4 | 75 - 99 | Good: Correct explanations and services, with some gaps. |
| 5 | 100 - 124 | Very Good: Well-executed with minor omissions. |
| 6 | 125 - 150 | Excellent: Great execution, no gaps, deep understanding. |