Full Stack Engineering Challenge: M250

Pet Image API Challenge: M250

Background

You're tasked with building a simple yet powerful API service that handles images labeled as either "cat" or "dog." Users should be able to upload labeled images, and later retrieve a randomly selected image based on a specific label.

Objective

- Develop two API endpoints via AWS API Gateway connected to AWS Lambda functions:
  - An endpoint to upload images labeled as "cat" or "dog."
  - An endpoint to retrieve a random image based on a given label.

Requirements

- **Services:** Utilize AWS Lambda, AWS API Gateway, AWS S3 & AWS Parameter Store (optional).
- **Programming Language:** Python 3.12 (vanilla). The only permitted external library is the AWS SDK for Python (Boto3).
- **Production Readiness:** Implement a solution that is secure, scalable, and cost-efficient.

Implementation Instructions

Image Upload Endpoint

- Create an API endpoint using AWS API Gateway that invokes a Lambda function.
- Endpoint format: `POST https://exampleurl.com/upload`
- Request body must include:
  - The image file (binary).
  - Label metadata indicating either `"cat"` or `"dog"`.
- Validate label input. Reject requests with invalid labels with a `400 Bad Request`.
- Successfully uploaded images should be stored securely in an AWS S3 bucket, structured by label (`cat/`, `dog/`).

Random Image Retrieval Endpoint

- Create another API endpoint connected to a Lambda function.
- Endpoint format: `GET https://exampleurl.com/random?label={label}`
- `label` must be either `"cat"` or `"dog"`. Reject invalid labels with a `400 Bad Request`.
- If no images exist for the given label, return a `404 Not Found` error.
- In successful cases, return a randomly selected image binary with status `200`.

Optimization

Implement optimization strategies including but not limited to:

- Efficient retrieval of random images (minimize latency).
- Properly structured storage for quick lookups.
- Consider weighted randomness, giving certain images higher probability of being selected.

Considerations

- Images are of the type `.jpg`, `.png` & `.webp`

Bonus Points

1. Deploy infrastructure using AWS CloudFormation in a YAML template.
2. Implement image retrieval with weighted randomness where each pet image has its own randomness weight. A higher weight indicates it's more likely to be randomly chosen, whereas a lower weight indicates a lower probability of the image being randomly selected.

Submission Instructions

1. **Source Code Repository:**
    - Include a link to the public repository (e.g., GitHub) where the submission files are hosted.
2. **Deployed on AWS:**
    - Share the URL to your deployed API endpoints
3. **README.md**:
    - Include a README.md with any interesting design choices
    - Also include the costs associated with your service for different levels of scale.
    - Consider adding any technical challenges and handling any edge/extreme cases.

Be ready to discuss your implementation decisions clearly and confidently. Happy coding!