Term Assignment

Introduction

The "OCRbuddy" project is a simple web application designed to simplify extracting text from images or PDFs and converting them into downloadable PDFs. Leveraging Amazon Web Services (AWS), OCRbuddy seamlessly integrates AWS Lambda, API Gateway, S3, Elastic Beanstalk, and SNS services to deliver a reliable, scalable, and secure OCR (Optical Character Recognition) solution. The primary goal of OCRbuddy is to help users effortlessly extract and manage textual information from images or PDF documents, enhancing productivity; this app can help students, professors, educators, and Anyone who can use it.

Project Objectives

1. Text Extraction: OCRbuddy aims to provide users with a straightforward platform to upload images or PDFs and automatically extract the textual content using AWS TextExtract services. By leveraging the power of OCR technology, users can transform non-searchable documents into editable text. Isn't that amazing!!

2. PDF Generation: Upon extracting the text from the uploaded files, OCRbuddy converts it into a well-structured PDF document. Users can then download the PDF, making sharing, storing, and accessing the information more accessible.

3. Email Notification: OCRbuddy allows users to receive email notifications containing links to the generated PDFs. By integrating AWS SNS (Simple Notification Service), users can subscribe and conveniently receive links to their converted documents in their email inbox.

Target Users:

OCRbuddy targets a diverse range of users across various domains, including students, educators, professionals, researchers, and businesses. Students and educators can use OCRbuddy to digitize and organize their study materials, handwritten notes, and textbooks. Professionals and researchers can efficiently manage documents, research papers, and articles, while businesses can streamline their document digitization and management processes.

Performance Targets:

OCRbuddy focuses on delivering a user-friendly and responsive web application that ensures users' efficient and seamless experience. Key performance targets include:

- Fast response times for file upload and text extraction.

- High availability and reliability to prevent service downtime.

- Scalability to accommodate a growing user base and varying workloads.

- Data security and privacy to protect user information and files.

Meeting Menu item Requirements:

To meet the menu item requirements for the OCRbuddy web application, I needed to implement various AWS services to handle different tasks, such as text extraction from images, PDF conversion, storage, serverless computing, and notification delivery. Below is a list of the selected AWS services, along with a comparison of alternative services considered and the reasons for choosing the preferred services:

1. Text Extraction from Images and PDFs:

Selected Service: AWS Text Extract Service

Alternative Considered: Tesseract OCR

Comparison and Reasoning:

- AWS Text Extract Service: I chose AWS Text Extract Service as it is a fully managed service provided by AWS that makes it easy to extract text from images and PDFs. It has built-in machine-learning models and offers high accuracy for text extraction, making it ideal for our OCR application. Additionally, it scales automatically to handle varying workloads, ensuring efficiency and reliability.

- Tesseract OCR: Tesseract OCR is an open-source OCR engine widely used for text extraction. While it is a powerful tool, it requires additional setup and maintenance, which may introduce complexity and potential issues. Since AWS Text Extract Service provides a managed and scalable solution with high accuracy, it was the preferred choice for my application.

2. PDF Conversion:

Selected Service: Third-party Libraries such as Pdf lib and pdf kit

Comparison and Reasoning:

- Third-Party Libraries: I utilized AWS Lambda for PDF generation as it supports Node.js, which provides various third-party libraries(pdf-lib and pdf kit) for PDF generation. AWS Lambda is a serverless computing service that allows one to execute code without managing servers, making it cost-effective and easy to scale. It also integrates seamlessly with other AWS services, such as S3, for storing the generated PDFs.

3. Storage:

Selected Service: AWS S3 (Simple Storage Service)

Alternative Considered: AWS EBS (Elastic Block Store)

Comparison and Reasoning:

- AWS S3: I opted for AWS S3 for storing both the uploaded files and the generated PDFs. S3 is a highly durable and scalable object storage service for handling various data types, including images and

documents. It also provides features like versioning, access control, and lifecycle policies, ensuring data security and cost optimization.

- AWS EBS: AWS EBS is block-level storage designed for use with EC2 instances. While it is suitable for attaching to EC2 instances and running databases, it may not be the best choice for storing user files and PDFs, as it doesn't offer the same level of scalability and accessibility as S3.

4. Serverless Computing:

Selected Service: AWS Lambda

Alternative Considered: AWS EC2 (Elastic Compute Cloud)

Comparison and Reasoning:

- AWS Lambda: We chose AWS Lambda for serverless computing due to its event-driven nature and automatic scaling capabilities. Since our application involves pre-decide usage (text extraction and PDF generation requests), using Lambda allows me to execute code on demand without needing to manage and provision servers continuously. This approach optimizes costs and ensures efficient resource utilization.

- AWS EC2: AWS EC2 provides virtual servers for running applications in a traditional server-based model. However, this would require continuous management of server instances and might lead to underutilization or over-provisioning of resources for my workload.

5. Notification Delivery

Selected Service: AWS SNS (Simple Notification Service)

Alternative Considered: AWS SES (Simple Email Service)

Comparison and Reasoning:

- AWS SNS: I have utilized AWS SNS for notification delivery as it provides a flexible and scalable platform for sending messages to multiple endpoints. It supports various communication protocols, including email, SMS, and mobile push notifications. With SNS, we can easily send email notifications to users with the link to the generated PDFs and handle subscriptions for email confirmations.

- AWS SES: AWS SES is a powerful email service that sends transactional and marketing emails. While it can handle email notifications, I wanted to use that SES service for email notifications, but I cloud not, as aws academy does not allow access to it! Bad luck…

6. Compute Service

Selected Service: AWS Elastic Load Balancer

Alternative Consideration: EC2 service

- AWS Elastic Load Balancer: I have used this Service for frontend deployment as it is relatively easy to deploy the application with that!
- EC2: This was also an option, but it would take much time to configure and have to manually scale the application as needed where Elastic Load Balancer provides automatic scaling

7. AWS API Gateway

- There was no alternative for this Service! An API gateway is simply a robust REST API management service to use! It allowed me to send the higher payload from the front end, as I am sending Base64 data through post request, which would be enormous, and I need to increase the payload size for API! So API gateway was one way to go for me! I could have used Lambda's Function Url, but it does not allow flexibility as API Gateway allows.

By carefully evaluating and selecting the above AWS services, I created an efficient and scalable solution for my OCR buddy web application, effectively meeting the menu item requirements. The chosen services provided a well-integrated and managed cloud infrastructure, which allowed us to focus on building and deploying the application's core functionality without worrying about the underlying infrastructure management.

Deployment Model - Public Cloud

OCRbuddy is deployed using the Public Cloud deployment model, explicitly leveraging the services provided by Amazon Web Services (AWS). Public Cloud refers to the cloud infrastructure and services made available to the general public over the Internet. In this case, OCRbuddy is hosted on the AWS Public Cloud, allowing users from anywhere to access the web application through their web browsers.

Advantages of the Public Cloud Deployment Model for OCRbuddy:

1. Scalability: Public Cloud services like AWS Elastic Beanstalk and Lambda offer automatic scaling capabilities, allowing OCRbuddy to handle varying user loads without manual intervention.

2. Cost-Effectiveness: The Public Cloud follows a pay-as-you-go pricing model, enabling OCRbuddy to pay only for the resources it consumes, which can be cost-effective, especially for smaller applications.

3. Global Accessibility: Being hosted on a Public Cloud, OCRbuddy can be accessed by users from different geographic locations, promoting wider accessibility.

4. High Availability: AWS provides a reliable infrastructure with redundancy and failover mechanisms, ensuring high availability of the application.

5. Elasticity: OCRbuddy can easily adjust its resource allocation based on demand, allowing it to scale up or down as needed, ensuring optimal performance.

Delivery Model - Combination of PaaS and FaaS

OCRbuddy adopts a delivery model combining PaaS (Platform as a Service) and FaaS (Function as a Service).

- PaaS: The front end of OCRbuddy is hosted on AWS Elastic Beanstalk, a PaaS offering. Elastic Beanstalk abstracts away the complexities of infrastructure management, making it easier for developers to deploy and manage web applications. It provides a platform where developers can focus on application development rather than infrastructure setup.

- FaaS: The backend OCR processing is handled by AWS Lambda, which is a FaaS service. Lambda allows developers to run code in response to events without managing servers. In this case, Lambda functions

are used to process OCR requests triggered by users through API Gateway. FaaS enables serverless computing, where developers can focus on writing code in the form of functions executed on demand. Also, I had the flexibility to increase the timeout period of the process as the image extraction is a heavy task; I have to increase the lambda function timeout period

In conclusion, OCRbuddy's deployment model is based on the Public Cloud, specifically utilizing AWS services. The delivery model combines PaaS (Elastic Beanstalk) for the front end and FaaS (Lambda) for the backend OCR processing, offering benefits such as scalability, cost efficiency, and developer productivity.

Final Architecture:

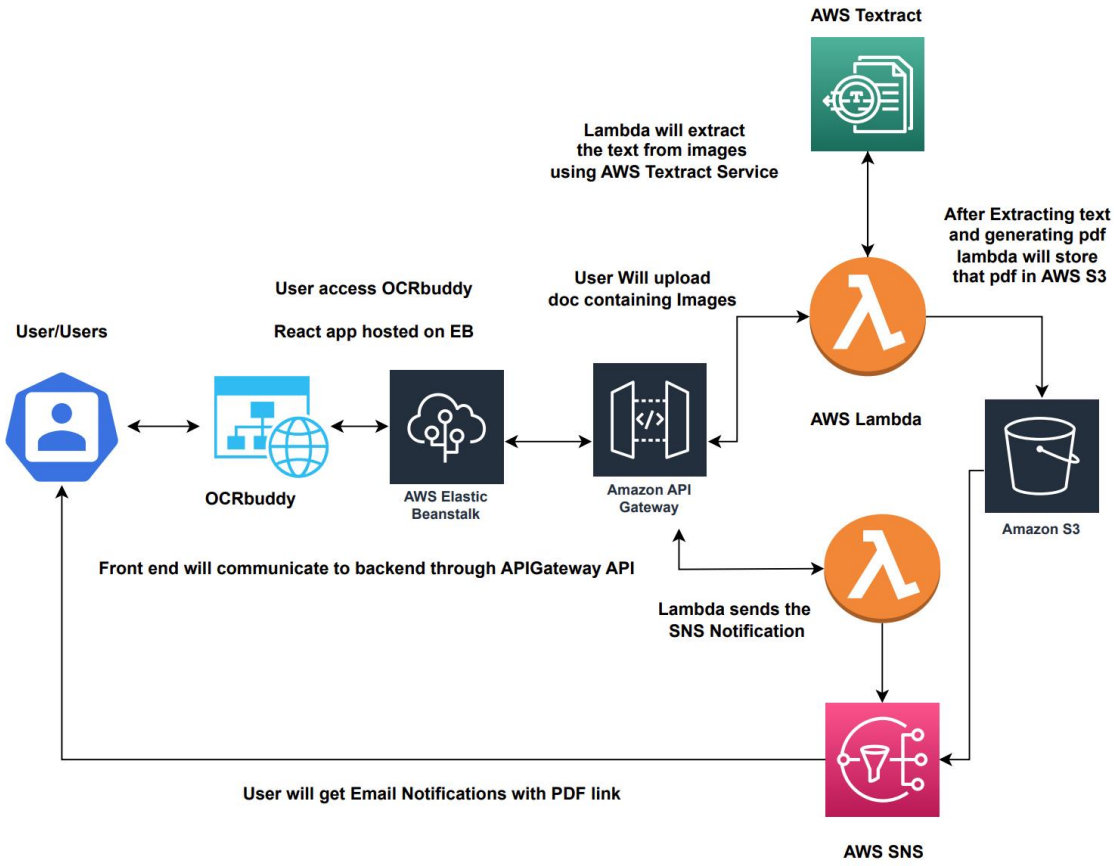

Fig 1: OCRbuddy AWS Architecture (Draw.io)

OCRbuddy's final architecture is to leverage various AWS cloud mechanisms to deliver a scalable and efficient OCR (Optical Character Recognition) web application. The cloud mechanisms used in the architecture are:

1. AWS Elastic Beanstalk: Hosts the frontend application, providing an easy-to-use platform for deploying, managing, and scaling web applications.

2. AWS Lambda: Executes serverless functions for OCR processing, text extraction, PDF generation, and email notification. Lambda allows automatic scaling based on request volume.

3. AWS API Gateway: Serves as the entry point for users to interact with the application. It forwards HTTP requests to the corresponding Lambda functions for processing.

4. Amazon S3: Stores the generated PDFs, making them accessible for users to download. S3 provides high durability, availability, and security for data storage.

5. AWS Simple Notification Service (SNS): Enables email notifications for users who request them. It sends PDF links to the provided email addresses; the message only goes if the user Confirms the subscription.

6. AWS Textract: which takes an image or pdf as input, processes the machine learning OCR, and gives the text as output, which then is used to generate the pdf.

Programming Languages and Code:

OCRbuddy is built using ReactJS for the front end and Node.js for the back end. ReactJS is used on the front end due to its component-based architecture, which facilitates modularity and reusability. Node.js, known for its event-driven, non-blocking I/O model, is chosen for the backend to handle asynchronous operations efficiently.

The frontend code is responsible for user interactions, uploading files, and displaying results. The backend code, implemented as AWS Lambda functions, handles OCR processing, text extraction, PDF generation, and email notification.

Deployment to the Cloud:

The application is deployed to AWS using CloudFormation, which allows for infrastructure as code. The CloudFormation template defines all the AWS resources required for the application, including the Elastic Beanstalk environment, Lambda functions, API Gateway, S3 buckets, etc.

Once the CloudFormation stack is created, it automatically provisions and configures the necessary resources, making the application ready for use.

Architectural Comparison:

OCRbuddy's architecture combines PaaS (Elastic Beanstalk) and FaaS (Lambda) services. While the combination of Elastic Beanstalk and Lambda is not explicitly taught as a single architecture in the course, it is a reasonable and efficient approach to building scalable web applications.

Elastic Beanstalk streamlines the deployment and management of the front end, allowing developers to focus on application logic rather than infrastructure management. On the other hand, Lambda enables serverless backend processing, reducing operational overhead and providing automatic scaling based on demand.

Overall, the chosen architecture demonstrates a well-thought-out integration of cloud services to achieve a functional and secure OCR web application. AWS services offer a reliable and cost-effective solution, leveraging the benefits of serverless computing and managed platform services.

Data Security:

OCRbuddy's application architecture implements several security mechanisms to ensure data security.

AWS IAM Roles: IAM LAB role controls access to AWS resources. Which Is predefined in AWS Academy. I gave a single IAM Role to all the services.

Encryption at Rest: Data stored in Amazon S3 buckets is encrypted using AWS S3 server-side encryption with Amazon S3-managed keys (SSE-S3). This ensures that data remains encrypted while at rest in S3, providing an additional layer of protection against unauthorized access; also, I have given the bucket a policy role, which defines bucket access policy to allow get objects from the bucket to all the users and put the function to only specified Lambda. So there is a vulnerability in terms of security, as Anyone can access the S3 object if they have the S3 object URL

Encryption in Transit: All communication between the frontend, backend, and API Gateway is secured using HTTPS/SSL encryption. This ensures that data transmitted over the network remains encrypted and safe from eavesdropping.

CloudFormation Security: The CloudFormation template is stored securely, and access to modify or create CloudFormation stacks is restricted to authorized users only.

SNS Access Control: Access to AWS SNS for sending email notifications is tightly controlled using IAM roles and permissions; only Lambda can create topics and subscriptions.

Monitoring and Logging: AWS CloudWatch monitors and logs application events and metrics. This allows for quick identification of potential security issues or anomalies.

Still, My App OCRBuddy lacks somewhere from a security perspective, As Anyone can access the S3 Object if they have the URL! So in the future, I would Use Authentication Mechanisms like JWT tokens to access the bucket object only if they are authorized.

Security Mechanisms Used:

The security mechanisms used in OCRbuddy's architecture include AWS IAM for access control, S3 server-side encryption for data at rest, HTTPS for data in transit, and input validation to prevent security exploits. Additionally, CloudWatch monitors and logs to help identify and address security incidents.

Reproduce My architecture in a private cloud.

Providing a precise cost estimation for a private cloud implementation can be challenging due to the many variables involved. However, I can give a rough estimate based on typical costs for the components needed to replicate OCRbuddy's architecture in a private cloud.

1. Private Cloud Infrastructure: The cost of physical servers or virtualization technologies for a small-scale implementation can range from $20000 - $40000 or more. This includes server hardware, storage, networking equipment, and cooling costs.

2. Web Servers: The cost of web servers, load balancers, and auto-scaling capabilities can vary based on the number of servers and their specifications. For a small-scale setup, the cost can range from $6000 to $20,000.

3. Application Server: The cost of the application server hosting Node.js applications can range from $10,000 to $20,000 or more, depending on the server specifications.

4. Containerization or Serverless: Implementing containerization technologies like Docker or serverless frameworks may involve additional software and operational costs, estimated at $5,000 to $8,000 or more.

5. Database: to store the pdf and user data in the database; the cost of a secure database server can range from $15,000 to $25,000 or more, depending on the type of database system and its capacity.

6. Networking and Security Appliances: The cost of firewalls, IDS, and other networking and security appliances can range from $10,000 to $20,000 or more.

7. Staffing and Operational Expenses: The organization would also need to consider the cost of hiring and maintaining a team of IT professionals to manage and maintain the private cloud infrastructure. These costs can vary based on the team's size and expertise. Roughly $20000 for a small team

Considering the above components, a rough estimation for the total cost of a private cloud implementation to replicate OCRbuddy's architecture could be $50000 to $100000 or more for a small to medium-scale setup.

Software Would need this setup: VMware vSphere for virtualization, Open stack for private cloud management, Cisco ACI for network management, Ceph (Storage management), and F5 BIG-IP (Load Balancer). Etc.

Sure, here are example answers for the 8th and 9th questions based on your project:

Monitoring for Cost Control:

The AWS Lambda service is the cloud mechanism that would be most important to add monitoring to ensure costs do not escalate out of budget unexpectedly. AWS Lambda charges based on the number of requests and the time our code executes. Since our application heavily relies on Lambda functions to process OCR, text extraction, and email sending, it has the potential to cost the most money if not monitored effectively. By monitoring and utilizing AWS CloudWatch, I can closely track the number of Lambda invocations, execution duration, and any errors or throttling to manage costs and optimize resource allocation proactively. Fine-tuning the Lambda function memory allocation and optimizing code to execute faster can also help control costs.

Application Evolution:

If the development of the OCRbuddy application were to continue, I would have several features in mind to enhance user experience and functionality. These features and their associated cloud mechanisms are as follows:

a. User Accounts and Authentication: Implement user account creation and authentication using AWS Cognito. This will allow users to securely log in, access their converted PDF history, and manage their preferences.

b. Language translation: I wanted to use the AWS Translate service to translate the extracted text to any other language, which would be very useful, but I could not, as AWS Academy did not allow me to do so. but In the future, I can use this Service to enhance the OCRbuddy platform functionality

f. User Analytics: Utilize AWS CloudFront and Amazon CloudFront Logs to gain insights into user behavior and application performance, helping to identify potential bottlenecks and optimize user experience. If privacy rules are in my favor, then!!

g. Auto-Deletion of Old PDFs: Implement an AWS Lambda function triggered by Amazon CloudWatch Events to automatically delete old PDFs from the S3 bucket, reducing storage costs.

By incorporating these features and leveraging various AWS services, OCRbuddy can become a powerful and user-friendly OCR application, providing enhanced text extraction and PDF conversion capabilities to its users while maximizing the benefits of cloud computing.