CSCI 5409

Advanced Topics in Cloud Computing

FINAL REPORT

Submitted By

Jeffry Paul Suresh Durai [B00952114]

Contents

Introduction

First, introduce your project. What is it? What is it supposed to do? Give the markers the information they need to evaluate whether your choices in later questions were good choices or bad choices. In other words, give them the context of your software and what it is supposed to achieve, who its users are, and what its performance targets should be.

My project is a web application that extracts text and labels from image files. Users can upload images through the site and choose how they want the information extracted. The application uses Amazon Textract to get text from the images and Amazon Rekognition to identify labels.

Once processed, the extracted data is shown to the user, making it easy to get useful information from the images. This application is useful for tasks like digitizing documents and analysing content.

Potential Users

Businesses and Organizations

- Companies with paper documents can turn these into digital text, making it easier to store and manage information.
- Companies with images, like product photos, can use the app to categorize and tag these images for better organization.

Individuals and Hobbyists

- People who want to organize their personal photos or documents can use the app to extract and categorize information.
- Hobbyists can use the app to get text and labels from images for projects like scrapbooks or personal archives.

Educational Institutions

- Students and teachers can use the app for projects that involve extracting text from images or analysing visual data.
- Schools and universities can digitize and manage educational materials and documents more easily.

Performance Targets

Accuracy

- The application should correctly extract text and identify labels from the uploaded images.
- The text should exactly match what is in the image and the labels should accurately describe the image content.

Speed

- The application should be able to process the uploaded images and generate outputs quickly.
- Ideally, it should only take a few seconds to analyse and extract information from an image, so users don't have to wait too long.

Usability

- The application should be easy and straightforward to use, particularly for those using it for the first time.
- The interface should be simple and clear so that users can upload images and get the results easily.

Scalability

- The application should be able to handle several images without any performance issues.
- It should be able to process large image files and work efficiently even if several people are using it at the same time.

Reliability

- The application should have a high level of availability and reliability.
- Users should be able to use it whenever they need to, without unexpected errors or downtime.

Menu Requirements

Describe how you met your menu item requirements: you will list the services you selected, and provide a comparison of alternative services, explaining why you chose the services in your system over the alternatives.

My application meets the menu item requirements as follows:

Compute

- **AWS Lambda:** This service was used to run the serverless functions in the application to perform the text and label extraction process on the uploaded images.
- **AWS EC2:** This service was used to host the front end of the application.

Storage

- **AWS S3:** This service was used to store the image files uploaded by the users from the frontend of the application.

Network

- **AWS API Gateway:** This service was used to create the REST API endpoints that trigger the Lambdas upon file upload.

General

- **Amazon Textract**: This service was used to perform the text extraction process on the uploaded images.
- **Amazon Rekognition:** This service was used to perform the label extraction process from the uploaded images.

The following is a comparison of alternative services and why I chose the above services over them:

Compute

I chose AWS Lambda over AWS Fargate for my project due to its lightweight nature and suitability for short-running tasks, which perfectly aligns with my needs [1]. Lambda also provides a default maximum concurrency limit of **1000**, allowing up to 1000 simultaneous executions, with the option to request an increase if higher concurrency is needed [2]. In contrast, AWS Fargate, while effective for containerized applications, has a longer startup time, with containers taking around **60** seconds to initialize. This is significantly slower compared to Lambda functions, which can start within **5** seconds [3]. This rapid startup and scalability make Lambda a better choice for my application.

I chose AWS EC2 over Elastic Beanstalk for hosting the frontend of my application, as it offers more flexibility, when compared to Elastic Beanstalk, allowing me to have more control over the infrastructure. While Beanstalk provides a simplified, managed environment with automatic scaling and integration with other AWS services, it may not support all application requirements and could potentially lead to vendor lock-in [4]. Additionally, EC2 provides high availability and good fault tolerance, making it an excellent choice for my application.

Storage

I chose S3 for storage, over database services such as DynamoDB, as my project requires only a simple storage service for storing the uploaded files. With S3, I could store virtually any number of files in the S3 bucket with no significant drops in performance. It is also very efficient, as it is fully elastic and automatically grows and shrinks as we add or remove files from the bucket. It provides **99.999999999%** (11 nines) data durability and **99.99%** availability by default, making it the ideal choice for the storage component of my application [5].

Network

I chose AWS API Gateway for network to provide RESTful API endpoints for users to upload image files to an S3 bucket. API Gateway allows me to easily create, deploy and attach the created API Gateways as triggers for executing Lambda functions, streamlining the process of building and managing APIs. Additionally, API Gateway automatically handles any level of incoming traffic, ensuring that the API remains responsive and scalable under varying loads [6]. This seamless integration with other AWS services, combined with its ability to manage heavy traffic, makes API Gateway an ideal solution for my application's needs.

General

For general services, I have chosen AWS Textract and AWS Rekognition over other similar services such as AWS Comprehend and AWS SageMaker. My application only requires basic image recognition and analysis services, making these services a more suitable option over the others. Services such as SageMaker, are more complex to set up and use, and are also more expensive [7]. Textract and Rekognition are comparatively easier to set up and integrate with other AWS services such as AWS Lambda or S3.

Deployment Model

Describe your deployment model. Why did you choose this deployment model?

The deployment model that I have chosen for the project is the **Public Cloud**. In a public cloud, a cloud service provider makes computing resources available to users over the public internet. These cloud service providers sell resources according to subscription-based or pay-per-usage pricing models [8]. In my case, I am using AWS as my public cloud provider.

I have chosen this deployment model for the following reasons:

Minimal Investment

Public cloud resources are much more affordable and cost-effective. There are no maintenance costs, and I only need to pay for the resources that I have used. Expenditure is much lower and more predictable [9]. I can also avoid the investment required to deploy and maintain on-premises IT infrastructure, and only pay for the resources and services used, which aids cost savings [8].

Easy to Set Up

With this model, I can focus on building the application without having to spend time configuring any physical infrastructure, as they are handled by the public cloud provider. Also, it is possible for any infrastructure to get outdated over time. But as the sole focus of the public cloud provider is to maintain and manage infrastructure to the highest quality [9], this ensures that my application is always running on the latest infrastructure.

Quick Deployments

With the public cloud model, I can quickly and easily deploy my application with minimal technical expertise. There are no upfront costs or lock-in contracts, and I can also try out various services for free before committing them to scale [9]. With the prebuilt services and tools, I can deploy the services in a fraction of the time it would take with traditional methods [10].

Reliability

Public cloud providers invest heavily in infrastructure and maintains multiple data centres worldwide. This allows me to get access to the latest hardware and software as they ensure all upgrades and patches are up to date. Using a public cloud, I can also access automatic backup and disaster recovery solutions, which helps ensure data resilience. I am also assured of my

cloud application having high availability and reliability from automatic redundancy coupled with technologies like content delivery networks and load balancers [10].

Sustainability

On-premises resources typically contribute more to the carbon footprint when compared to virtualized resources. Public clouds can yield overall energy savings of **50%** or higher and help bring down the carbon footprint [11]. Cloud service providers offer tools and resources to monitor the environmental impact of cloud services usage and help reduce the environmental footprint [10]. Hence by using a public cloud, I can significantly reduce my carbon footprint.

Delivery Model

Describe your delivery model. Why did you choose this delivery model?

The delivery model that I have chosen for the project is the **Software as a Service (SaaS)** model. In SaaS, the application is made available on a remote cloud network that is accessed through the web or an API [12].

My reasons for choosing this delivery model are as follows:

Configuration and Installation

In SaaS, the application is already installed and configured in the cloud. I will only need to provision the server for an instance in cloud, and in a couple of hours, I can have the application up and running. This reduces the time that I will need to spend on installation and configuration and can reduce the issues that get in the way of deploying my software [12].

Cost-Effectiveness

SaaS helps lower operating costs since the application usually resides in a shared or multi-tenant environment, where the hardware and software license costs are low compared with the traditional model [12]. Maintenance costs are reduced as well, since the SaaS provider owns the environment. As the SaaS provider manages the IT infrastructure that is running the software, the overall costs of hosting my application are greatly reduced [13].

Accessibility

As the application is already deployed and hosted on the cloud, users need not configure or install any resources to run the application. Users can access the application from anywhere, and at any time.

Scalability

SaaS allows me to easily scale up and down, depending on my needs. For example, when my application gains more users, I can easily switch to one of the many subscription options that the SaaS provider offers and keep my application running smoothly [13].

Easy Upgrades

With SaaS, the costs and efforts that are associated with upgrades and new releases are lower than the traditional model [13]. This means that I need not spend too much time dealing with any hardware or software upgrades as they are all handled by the SaaS provider.

Final Architecture

How do all of the cloud mechanisms fit together to deliver your application?

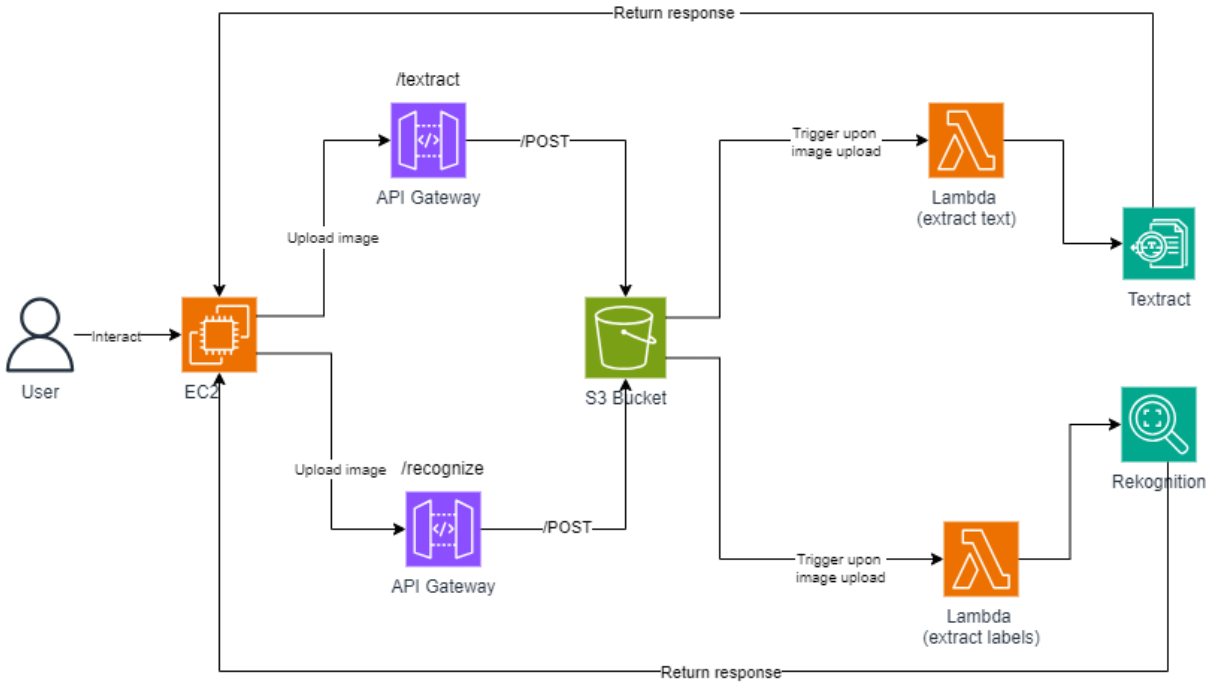

Figure 1: Architecture Diagram

The various cloud mechanisms fit together to deliver the application as follows:

User Interaction

- The user interacts with the frontend of the application that is hosted through an EC2 instance.
- The application displays a form to upload image files and has options for users to choose to extract either text or labels from their uploaded images.

File Upload

- The user uploads their image files through the frontend via an API Gateway.
- These files are stored in an S3 bucket.

Text Extraction

- If the user had chosen to extract text from the image file, the corresponding lambda function is triggered through the /textract API Gateway endpoint.
- The text extraction process is carried out on the uploaded image file with the lambda function calling the Textract service.

Label Extraction

- If the user had chosen to extract labels from the image file, the corresponding lambda function is triggered through the /recognize API Gateway endpoint.
- The label extraction process is carried out on the uploaded image file with the lambda function calling the Rekognition service.

Outputs

- Upon completion of the text/label extraction, the response from the lambda function is formatted and displayed to the user through the frontend.
- The user can also choose to download the extracted text/labels as a .txt file.

Where is data stored?

The data is stored in an S3 Bucket.

What programming languages did you use (and why) and what parts of your application required code?

The programming languages that I have used are:

- Python – For the backend (Lambda functions)
- Next.js – For the frontend of the application

I have chosen Python for my Lambda functions due to its simplicity in scripting and seamless integration with other AWS services. Python's syntax makes it easy to read and process files, which is essential for my application that extracts text and labels from image files.

I have chosen Next.js for the frontend because it offers a structured framework with many features that simplify development and improve performance. Next.js takes care of routing, data fetching, and other essential tasks, so I can focus on building my application's core features.

How is your system deployed to the cloud?

The system is deployed to the cloud through CloudFormation.

Data Security

How does your application architecture keep data secure at all layers? If it does not, if there are vulnerabilities, please explain where your data is vulnerable and how you could address these vulnerabilities with further work.

My application architecture keeps data secure at all layers in the following ways:

EC2 Instance

The EC2 instance is secured by assigning a security group that allows only necessary inbound traffic. This reduces the attack surface by limiting the network access to the instance and acts as a virtual firewall to control incoming and outgoing traffic [14].

API Gateways

The API Gateways have certain vulnerabilities as they currently accept requests from any origin. Also, there is no authentication mechanism on the API gateways which makes them vulnerable to cross-origin attacks and data leaks.

S3 Bucket

S3 buckets have encryption configured by default, and all hence all files that are uploaded to the S3 bucket are automatically encrypted at rest [18]. I am also using presigned URLs with restricted permissions and short expiration times when uploading files to the S3 bucket which further enhances its security.

Lambda Functions

The Lambda functions are secured by authorizing only the specified IAM roles to execute them. This limits access to these functions to trusted entities only. Their security is further enhanced by following the principle of least privilege by scoping precise permissions to specific resources [19].

Textract and Rekognition Services

The data processed by the Textract and Rekognition services is kept secure by authorizing specified IAM roles to the Lambda functions that runs these services and by limiting access by following the principle of least privilege. Additionally, these services are protected by AWS global network security [20][21].

With further work, it is possible to secure the API Gateways in my application by implementing the following mechanisms:

i) Web Application Firewall

With AWS WAF, I can create a web application firewall that I can use to create rules to allow or block requests from specified IP address ranges, requests from CIDR blocks, requests that originate from a specific country or region, requests that contain malicious SQL code, or requests that contain malicious script [15].

ii) SSL Certificates

I can also secure the API Gateways by generating an SSL certificate and then using its public key in the backend to verify that HTTP requests are from API Gateway. This allows the HTTP backend to control and accept only requests that originate from the API Gateway, even if the backend is publicly accessible [16].

iii) Request Throttling

With request throttling, the API gateway can only accept a specific number of simultaneous client requests during a given time. This helps prevent Distributed Denial of Service (DDoS) attacks [17].

Which security mechanisms are used to achieve the data security described in the previous question? List them, and explain any choices you made for each mechanism (technology you used, algorithm, cloud provider service, etc.)

The security mechanisms that are used to achieve the data security in my application are:

Security Groups

The EC2 instance in my application is assigned security groups with inbound and outbound rules to control the incoming and outgoing traffic respectively. Inbound rules control the incoming traffic to the instance, and outbound rules control the outgoing traffic from the instance, which helps prevent unauthorized access [14].

I could have also used network ACLs (Access Control List) which would allow or deny specific inbound or outbound traffic at the subnet level [22]. However, since security groups are more versatile when compared to network ACLs [23], I have decided to use security groups as the security mechanism for my EC2 instance.

Data Encryption

The S3 bucket in my application has encryption configured by default, and all new objects that are uploaded to it are automatically encrypted at rest [24]. The various options for encrypting the uploaded objects are as follows [25]:

1) Server-side encryption with Amazon S3 managed keys (SSE-S3)
2) Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS)
3) Dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS)
4) Server-side encryption with customer-provided keys (SSE-C)

Out of these, I opted for the default encryption option, which is the server-side encryption with Amazon S3 managed keys (SSE-S3), as it is a simple and cost-effective solution for my application.

Presigned URLs

Presigned URLs are used to upload objects to the S3 bucket. By generating these URLs with specific permissions, we ensure that only authorized users can upload files. This mechanism allows us to set an expiration time for the URL, granting temporary access to the user. Additionally, the URLs can be configured to permit only certain actions, such as uploading, further enhancing security. This approach simplifies access management by eliminating the need for users to have AWS credentials, while still maintaining strict control over who can upload objects to the bucket [26].

IAM Roles

The Lambda functions, Textract and Rekignition services are secured by IAM roles with fine-grained access control to grant only specific permissions to users. This approach helps limit access to certain resources or actions in the AWS environment, securing resources and preventing unauthorized access or modifications [27].

CORS (Cross-Origin Resource Sharing)

CORS enhances the security of our cloud application by specifying how resources on our server can be requested from different origins. It allows us to define which external domains are permitted to access our resources and under what conditions, thus preventing unauthorized access. This security mechanism is particularly used to secure the data in our S3 Bucket and API Gateways [28]. By enabling CORS, we restrict cross-origin requests to trusted domains, control the methods (GET, POST, etc.) that are allowed, and specify allowed headers. This level of control helps ensure that only authorized interactions occur with our cloud resources.

Reproducing Architecture

What would your organization have to purchase to reproduce your architecture in a private cloud while providing relatively the same level of availability as your cloud implementation? Try to give a rough estimate of what it would cost, don't worry if you are far off. These systems are complicated and you don't know all the exact equipment and software you would need to purchase. Just explore and try your best to figure out the combination of software and hardware you would need to buy to reproduce your app on-premise.

In order to reproduce my architecture in a private cloud, we would require the following:

Servers

First, we would require multiple servers to handle different functions, such as application servers, database servers, and dedicated servers for specific tasks like image processing. These servers usually have an upfront cost of around $1800 and another $40 per month for power and cooling expenses [29].

Assuming that we need a minimum of 4 servers for our private cloud, this comes down to

- $1800 * 4 = $7200 for upfront costs and
- $40 * 4 = $160 per month for electricity

Storage Space

To store large amounts of data, we would require a NAS (Network Attached Storage) which costs a minimum of $1200 [30].

Switches and Routers

We also need to have proper networking equipment such as switches and routers to ensure high-speed and secure connections. The combined costs of network switches and routers for our servers is estimated to be around $600 [30].

Server OS

In order to run our servers, we need a suitable server operating system such as Linux Ubuntu Server, Red Hat Enterprise Server or CentOS [31]. Suppose we go with Linux Ubuntu Server, it would cost us around $3500 for each server per year [32].

Since we have 4 servers, that would be

- $3500 * 4 = $14000 per year

Antivirus

To safeguard our system from malicious attacks, we would need to invest in a good antivirus software. This usually costs an average of $25/ year [33].

Management and Monitoring Tools

We also need to monitor and manage the health of our infrastructure with certain health monitoring tools such as Nagios [34]. This would cost around $8000 per server.

So, for 4 servers it would be

- $8000 * 4 = $32000

Load Balancers

To distribute the traffic, we would need load balancers. A load balancer such as the LoadMaster XHC55-NG starts at $515100 [35].

Now let us summarize and breakdown the total upfront and recurring costs associated with the infrastructure discussed so far:

- **Total Upfront Cost:** $515100 + $72000 + $1200 + $600 = $524100
- **Total Yearly Recurring Cost:** $14000 (Server OS) + $25 (Antivirus) = $14025

Table 1: Breakdown of Costs

Item	Cost per Unit	Quantity	Total Cost	Recurring Cost (Per Month)
Servers	$1800 (upfront)	4	$7200	$160 ($40 x 4)
Power and Cooling for Servers	$40 per month	4	N/A	$160
Storage (NAS)	$1200 (upfront)	1	$1200	N/A
Switches and Routers	$600 (upfront)	1	$600	N/A
Server OS (Linux Ubuntu Server)	$3500 per year	4	$14000 per year	N/A
Antivirus	$25 per year	1	$25 per year	N/A
Management and Monitoring Tools	$8000 per server	4	$32000	N/A
Load Balancer	$515100 (upfront)	1	$515100	N/A

Cloud Monitoring

Which cloud mechanism would be most important for you to add monitoring to in order to make sure costs do not escalate out of budget unexpectedly? In other words, which of your cloud mechanisms has the most potential to cost the most money?

The cost estimates of the cloud mechanisms used in my application are as follows:

AWS EC2

EC2 Instances are charged by the hour or second (minimum of 60 seconds) and varies based on the type of instance. In my application, I have used a t2.micro instance which has an on-demand hourly rate of $0.0116 [36].

AWS Lambda

The cost of executing the Lambda functions varies based on the number of requests and the duration it takes for the code to execute. It is typically $0.20 per 1M requests [37].

AWS API Gateway

For the API Gateways, we are charged only for the API calls that are received and the amount of data transferred out. In my application, I have used REST APIs which are listed at $3.50 per million requests, as a starting price [38].

AWS S3

In Amazon S3, pricing depends on a number of factors such as the size of the object uploaded to the bucket, duration of storage in the bucket and so on. For the standard storage class, pricing begins at $0.023 per GB for the first 50 TB / Month [39].

Amazon Textract and Rekognition

The pricing for Amazon Textract and Rekognition services depends on several factors such as the type of documents being processed, number of pages scanned and so on. These services usually start at around $0.0010 - $0.0015 per image processed [40][41].

Upon analysing the various cloud mechanisms used and their prices, we can conclude that it would be most important to add monitoring to the **EC2 instance**. EC2 instances, when not managed properly, have a high potential for cost escalation. Although the t2.micro instance in my application has an on-demand hourly rate of $0.0116, costs can add up quickly if left running unintentionally or if it needs to be scaled up to larger instance types to handle increased loads.

Further Development

How would your application evolve if you were to continue development? What features might you add next and which cloud mechanisms would you use to implement those features?

In the future, I could further develop my application to include the following features:

Document Translation

To enhance the functionality of the application, I could integrate Amazon Translate. This feature would allow users to translate documents into different languages, making the application more accessible to a global audience.

Text-to-Speech

I could also integrate Amazon Polly to convert text documents into speech. This text-to-speech capability will make content more accessible for users with visual impairments or for those who prefer listening rather than reading long documents.

Admin Dashboard

Adding an admin dashboard integrated with Amazon QuickSight could provide insights into how the application is being used. It would allow me to view data and trends about user activity and other important metrics.

Chatbot Integration

By using Amazon Lex, I could create a chatbot to help users navigate the application. This chatbot would provide real-time guidance, answer frequently asked questions, and improve overall user support and engagement.

User Feedback

I could collect user feedback and run surveys to understand how to improve my application. Amazon Simple Email Service (SES) can be used to send out survey emails, and Amazon QuickSight can help analyse the results and visualize feedback trends.

References

[1] M. Rosam, “Fargate vs Lambda: a comparison of serverless technologies”, *Quix* [Online]. Available: https://quix.io/blog/fargate-vs-lambda-comparison [Accessed: Aug. 2, 2024].

[2] “Understanding Lambda function scaling”, *AWS* [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/lambda-concurrency.html#:~:text=By%20default%2C%20Lambda%20provides%20your,functions%20in%20an%20AWS%20Region [Accessed: Aug. 2, 2024].

[3] William, “Fargate vs Lambda: The Battle of the Future”, *ClickIt* [Online]. Available: https://www.clickittech.com/devops/fargate-vs-lambda/#:~:text=Fargate%20functions%20must%20be%20packaged,get%20started%20within%205%20seconds [Accessed: Aug. 2, 2024].

[4] M. Mickiewicz, “AWS Elastic Beanstalk vs EC2: A Detailed Comparison”, *SitePoint* [Online]. Available: https://www.sitepoint.com/aws-elastic-beanstalk-vs-ec2/ [Accessed: Aug. 2, 2024].

[5] “Amazon S3”, *AWS* [Online]. Available: https://aws.amazon.com/s3/ [Accessed: Aug. 2, 2024].

[6] “Amazon API Gateway Features”, *AWS* [Online]. Available: https://aws.amazon.com/api-gateway/features/ [Accessed: Aug. 2, 2024].

[7] R. Venkatesh, “Amazon Rekognition vs. Amazon SageMaker: Choosing the right service for your image recognition and classification projects”, *Cevo* [Online]. Available: https://cevo.com.au/post/amazon-rekognition-vs-amazon-sagemaker-choosing-the-right-service-for-your-image-recognition-and-classification-projects/#:~:text=Amazon%20Rekognition%20is%20a%20good,and%20advanced%20data%20processing%20features [Accessed: Aug. 2, 2024].

[8] S. Susnjara, “Public cloud vs. private cloud vs. hybrid cloud: What’s the difference?”, *IBM* [Online]. Available: https://www.ibm.com/blog/public-cloud-vs-private-cloud-vs-hybrid-cloud/ [Accessed: Aug. 2, 2024].

[9] “What’s the Difference Between Public Cloud and Private Cloud?”, *AWS* [Online]. Available: https://aws.amazon.com/compare/the-difference-between-public-cloud-and-private cloud/#:~:text=Organizations%20save%20on%20costs%20by,cases%20than%20the%20private%20cloud [Accessed: Aug. 2, 2024].

[10] “What is a Public Cloud?”, *AWS* [Online]. Available: https://aws.amazon.com/what-is/public-cloud/ [Accessed: Aug. 2, 2024].

[11] “Is Cloud Computing Always Greener?”, *NRDC* [Online]. Available: https://www.nrdc.org/sites/default/files/cloud-computing-efficiency-IB.pdf [Accessed: Aug. 2, 2024].

[12] "Top 5 advantages of Software as a Service (SaaS)", *IBM* [Online]. Available: https://www.ibm.com/think/insights/software-as-a-service-advantages [Accessed: Aug. 2, 2024].

[13] "Computer software for business", *nibusinessinfo.co.uk* [Online]. Available: https://www.nibusinessinfo.co.uk/content/advantages-and-disadvantages-software-service-saas [Accessed: Aug. 2, 2024].

[14] "Amazon EC2 security groups for your EC2 instances", *AWS* [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security-groups.html [Accessed: Aug. 3, 2024].

[15] "Use AWS WAF to protect your REST APIs in API Gateway", *AWS* [Online]. Available: https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-control-access-aws-waf.html [Accessed: Aug. 3, 2024].

[16] "Generate and configure an SSL certificate for backend authentication in API Gateway", *AWS* [Online]. Available: https://docs.aws.amazon.com/apigateway/latest/developerguide/getting-started-client-side-ssl-authentication.html [Accessed: Aug. 3, 2024].

[17] "API gateway security", *solo.io* [Online]. Available: https://www.solo.io/topics/api-gateway/api-gateway-security/ [Accessed: Aug. 3, 2024].

[18] "Security best practices for Amazon S3", *AWS* [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/security-best-practices.html [Accessed: Aug. 3, 2024].

[19] "Applying the principles of least privilege", *AWS* [Online]. Available: https://docs.aws.amazon.com/lambda/latest/operatorguide/least-privilege.html [Accessed: Aug. 4, 2024].

[20] "Infrastructure Security in Amazon Textract", *AWS* [Online]. Available: https://docs.aws.amazon.com/textract/latest/dg/infrastructure-security.html#:~:text=As%20a%20managed%20service%2C%20Amazon,infrastructure%2C%20see%20AWS%20Cloud%20Security%20 [Accessed: Aug. 4, 2024].

[21] "Data protection in Amazon Rekognition", *AWS* [Online]. Available: https://docs.aws.amazon.com/rekognition/latest/dg/data-protection.html [Accessed: Aug. 4, 2024].

[22] "Control subnet traffic with network access control lists", *AWS* [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html [Accessed: Aug. 4, 2024].

[23] "Infrastructure security in Amazon VPC", *AWS* [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/infrastructure-security.html#VPC_Security_Comparison [Accessed: Aug. 4, 2024].

[24] "Protecting data with encryption", *AWS* [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingEncryption.html [Accessed: Aug. 4, 2024].

[25] "Protecting data with server-side encryption", *AWS* [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/serv-side-encryption.html [Accessed: Aug. 4, 2024].

[26] "Uploading objects with presigned URLs", *AWS* [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/PresignedUrlUploadObject.html [Accessed: Aug. 4, 2024].

[27] "AWS Lambda Security Best Practices", *dev.to* [Online]. Available: https://dev.to/sec_maestro/aws-lambda-security-best-practices-4f9 [Accessed: Aug. 5, 2024].

[28] "What is CORS?", *AWS* [Online]. Available: https://aws.amazon.com/what-is/cross-origin-resource-sharing/ [Accessed: Aug. 5, 2024].

[29] C. Downs, "How Much Does a Server Cost For a Small Business in 2024?", *ServerMania* [Online]. Available: https://www.servermania.com/kb/articles/how-much-does-a-server-cost-for-a-small-business [Accessed: Aug. 5, 2024].

[30] "Best Buy Canada", *bestbuy.ca* [Online]. Available: https://www.bestbuy.ca/en-ca/ [Accessed: Aug. 5, 2024].

[31] "How to set up a server for a small or mid-sized business", *Avast* [Online]. Available: https://blog.avast.com/setup-server-business-avast [Accessed: Aug. 5, 2024].

[32] "Ubuntu Pro: security, compliance, and support", *ubuntu.com* [Online]. Available: https://ubuntu.com/pricing/pro [Accessed: Aug. 6, 2024].

[33] G. Švažaitė, "Best cloud-based antivirus in 2024", *CyberNews* [Online]. Available: https://cybernews.com/best-antivirus-software/cloud-antivirus/ [Accessed: Aug. 6, 2024].

[34] "Nagios Pricing", *Nagios* [Online]. Available: https://www.nagios.com/pricing-plans/ [Accessed: Aug. 6, 2024].

[35] "LoadMaster XHC55-NG", *kemptechnologies.com* [Online]. Available: https://kemptechnologies.com/server-load-balancing/loadmaster-xhc55-ng [Accessed: Aug. 6, 2024].

[36] "Amazon EC2 On-Demand Pricing", *AWS* [Online]. Available: https://aws.amazon.com/ec2/pricing/on-demand/ [Accessed: Aug. 6, 2024].

[37] "AWS Lambda Pricing", *AWS* [Online]. Available: https://aws.amazon.com/lambda/pricing/ [Accessed: Aug. 6, 2024].

[38] "Amazon API Gateway pricing", *AWS* [Online]. Available: https://aws.amazon.com/api-gateway/pricing/ [Accessed: Aug. 6, 2024].

[39] “Amazon S3 pricing”, *AWS* [Online]. Available: https://aws.amazon.com/s3/pricing/ [Accessed: Aug. 6, 2024].

[40] “Amazon Textract pricing”, *AWS* [Online]. Available: https://aws.amazon.com/textract/pricing/ [Accessed: Aug. 6, 2024].

[41] “Amazon Rekognition pricing”, *AWS* [Online]. Available: https://aws.amazon.com/rekognition/pricing/ [Accessed: Aug. 6, 2024].