DALHOUSIE UNIVERSITY

CSCI 5409 – Advanced Cloud Computing

Professor: Dr. Tami Meredith

Final Project Report

Submitted by,
Sujahid Basha (B00925849)

Resources Utilized: -

Compute
EC2
Lambda
Storage
S3
DynamoDB
Network
API Gateway
General
SNS
Kendra

A. Project Resources.

A.1 Compute: -

EC2 can is the best option to host a web app on AWS if we need full control over the virtual machine's configuration, while Elastic Beanstalk can simplify the deployment and management of web applications. ECS and ECR can be a good fit if we want to use containers, while Lambda is useful for building event-driven architectures. While step Functions can help automate complex workflows, and IoT 1-Click can help simplify IoT device interactions with your web application. EC2 allows to choose from a wide range of instance types, operating systems, and software configurations to match out application's requirements. This flexibility can be beneficial for my applications over other compute resources.

A.2 Storage: -

1. AWS S3: -
   AWS S3 is an excellent choice for simple and scalable file storage needs. This provides simplicity, scalability, high availability, and integration with other AWS services, making it a versatile and cost-effective solution for my use cases. S3 integrates seamlessly with other AWS services, such as EC2, Lambda.
2. AWS DynamoDB: -
   DynamoDB is a NoSQL database, which means it can store structured, semi-structured, and unstructured data. My application requires flexible data modeling, low-latency access, and high scalability. As it provides scalability, high availability, flexible data modeling, and integration with other AWS services like Lambda, making it a powerful and cost-effective solution for my use cases.

A.3 Network: -

1. AWS API Gateway: -
   AWS API Gateway is a fully managed service that makes it easy to create, publish, and secure APIs at any scale. It allows to build RESTful APIs and WebSocket APIs that can integrate with other AWS services, such as Lambda, DynamoDB, S3, and others. AWS API Gateway is an excellent choice for building and securing APIs. It offers robust security, high scalability, efficient request handling, and seamless integration with other AWS services, making it an ideal choice for my use cases.

A.4 General: -

1. AWS SNS: -
   SNS can decouple the sender from the receiver of the messages. This allows to send notifications to multiple subscribers at once, without having to know their individual addresses or endpoints. As soon as user registers in the web application, the user will be automatically added to a SNS subscription. Such that whenever the user uploads a file and clicks to send an email then all the registered users will be,

notified by an email.

2. AWS Kendra: -

   AWS Kendra can index and search a wide variety of data sources, including documents stored in S3. main advantages of Kendra are its ability to provide natural language search capabilities which makes an excellent choice for adding powerful and intelligent search.

   capabilities to my web applications. It offers natural language search, supports a wide variety of data sources, and provides features to improve search relevance and accuracy, making it an ideal choice for my use cases.

B. Deployment Model →Public Cloud

   a. Scalability: - Public cloud providers offer virtually unlimited scalability, allowing us to quickly and easily increase or decrease their resources as needed.
   b. Cost-effectiveness: - Public cloud providers offer a pay-as-you-go pricing model, allowing us to only pay for the resources they use. This can be much more cost-effective than maintaining an on-premises infrastructure.
   c. Reliability: - Public cloud providers typically have robust infrastructure and redundancy measures in place to ensure high availability and reliability.
   d. Flexibility: - Public cloud providers offer a wide range of services and tools, allowing us to choose the ones that best meet our needs and customize their infrastructure as needed.
   e. Security: - Public cloud providers have dedicated teams of security experts and use advanced security technologies and measures to protect their infrastructure and our data.
   f. Global reach: - Public cloud providers have data centers located around the world, allowing us to easily deploy our services and reach customers in different regions.

C. Delivery Model → Platform as a Service (PAAS)

   a. PaaS provides a complete platform for developing, testing, and deploying applications, which includes the underlying infrastructure, operating systems, databases, and middleware. We can use pre-built components and tools to develop and deploy applications without worrying about the underlying infrastructure or management. This can save significant time and effort, allowing us to focus on building and improving the application itself rather than managing the infrastructure.
   b. PaaS is particularly useful for us who want to speed up their development processes and release new features and applications quickly. It is also beneficial for those who want to reduce the costs and complexity of managing their own infrastructure and focus on their core business activities.

D. Final System Architecture → Serverless Architecture.

   a. The application logic is broken down into small, independent functions

that are executed in response to specific events or requests. AWS Lambda is used to run these functions, which can be triggered by events such as API requests or changes to data in S3.

b. API Gateway is used to expose the application's functions as a RESTful API that can be accessed by the web application. S3 and DynamoDB is used for storing files and retrieving data, while DynamoDB is used store user details, while Kendra is used for advanced search capabilities.

c. This architecture allows for highly scalable and flexible applications, as the underlying infrastructure is managed by AWS and automatically scales to handle changes in demand. It also reduces operational overhead and costs by only charging for the actual usage of the functions, rather than for a fixed amount of server capacity.

E. Security Analysis: -
   a. As soon as user registers, their password is encrypted and stored in DynamoDB, as well as decrypted by retrieving the encrypted password. This adds extra security to the application.
   b. Whenever a user uploads the file, those are stored in S3 bucket which has access to only those users who have a valid login token.
   c. Whenever the user is inactive for a longer time, the session automatically terminates, forcing the user to login again into the application.

F. Cost Metrics
   a. Resources cost using public cloud.
      i. Ec2-

         1. On-Demand Linux pricing: - 0.0116 USD/hour.
         2. Instance Type: - t2.micro.Cores: - 8.

      ii. S3-

         1. Storage - $0.023 per GB.
         2. Requests & Data retrievals - $0.005.
         3. Get objects - $0.0004.

      iii. DynamoDB-

         1. Write Request Unit (WRU)- $1.25 per Million write requests.
         2. Read Request Units (RRU) - $0.25 per Million read requests.
         3. Storage- $0.25 per GB/ month.

      iv. Lambda-

         1. $0.20 per 1 million requests.

      v. API Gateway-

         1. No. of. HTTP Requests/ month - $1.00/ 1 million.
         2. No. of. Rest Request/ month - $3.50/ 1 million.

vi. AWS Kendra-

1. Connector Usage document scanned - $0.000001 per document.
2. Connector Usage when syncing - $0.35 per hour per connector when syncing.

vii. AWS SNS-

1. Email/Email-JSON - 1,000 notifications - $2.00 per 100,000 notifications.

b. On-premises resource cost:

i. Server Cost- A good quality server can cost anywhere from a few thousand dollars to tens of thousands of dollars, depending on the specific requirements. In addition to the initial cost of the server hardware, there are ongoing costs to consider such as electricity, maintenance, and upgrades.

ii. Storage – can be categorized in various categories as
1. Hardware costs.
2. Software costs
3. Licensing costs:
4. Maintenance costs
5. Security costs:
6. Electricity and cooling costs.

G. Analysis of Cost Metrics: - Building a web application involves various cost metrics including upfront costs, ongoing costs, and additional costs. if the web application requires high availability and scalability, a cloud-based solution like AWS or Google Cloud may be more expensive than an on-premises solution but may provide the required level of reliability and scalability.

a. Upfront costs:
i. Development cost: This includes the cost of hiring developers, designers, and other team members involved in building the application. The cost of development can vary widely depending on the complexity of the application, the number of features, and the technology stack used. On average, it can range from $50,000 to $500,000.
ii. Infrastructure cost: This includes the cost of hardware and software required to host the application. If the application is hosted on-premises, this cost can be significant, including servers, network infrastructure, and storage. Alternatively, hosting the application in the cloud can significantly reduce upfront infrastructure costs.

b. Ongoing costs:

i. Hosting cost: This includes the cost of hosting the application on a server, which can be a dedicated server or a shared server. The cost can vary depending on the hosting provider, the location of the server, and the resources required to run the application.

ii. Maintenance cost: This includes the cost of maintaining and updating the application, including security updates, bug fixes, and new feature development. The cost can vary depending on the complexity of the application and the frequency of updates.

iii. Support cost: This includes the cost of providing customer support and resolving issues related to the application. The cost can vary depending on the size of the customer base and the complexity of the issues.

c. Additional costs:

i. Marketing cost: This includes the cost of promoting the application through various channels, such as social media, paid advertising, and email marketing. The cost can vary depending on the marketing strategy and the target audience.
ii. Analytics cost: This includes the cost of collecting and analyzing data related to the application's usage, such as user behavior, conversion rates, and engagement metrics. The cost can vary depending on the analytics tools used and the amount of data collected.

d. Alternative approaches:

i. Cloud hosting: Hosting the application on a cloud platform such as Amazon Web Services (AWS) or Microsoft Azure can significantly reduce upfront infrastructure costs, as the cloud provider takes care of the hardware and network infrastructure. Cloud hosting also offers scalability, flexibility, and high availability, which can reduce ongoing costs related to hosting, maintenance, and support.
ii. Open-source technology: Using open-source technologies such as WordPress, Drupal, or Joomla can significantly reduce upfront development costs, as these platforms offer pre-built templates and plugins that can be customized to fit the application's needs. However, ongoing maintenance and support costs may be higher for open-source platforms due to the need for frequent updates and security patches.
iii. Low-code or no-code platforms: Using low-code or no-code platforms such as Bubble, Webflow, or Wix can significantly reduce upfront development costs, as these platforms offer pre-built templates and drag-and-drop interfaces that allow non-technical users to build applications. However, these platforms may not offer the flexibility and customization options of traditional development approaches.

e. Justification for a more expensive solution:
i. High-performance requirements: If the application requires high-performance resources, such as high CPU or memory usage, a more expensive solution may be justified to ensure optimal performance and user experience.

ii. Enterprise-level security requirements: If the application requires enterprise-level security features, such as multi-factor authentication, data encryption, and intrusion detection, a more expensive solution may be justified to ensure data privacy and protection.
iii. Complex integration requirements: If the application requires integration with other systems, such as payment gateways, third-party APIs, or enterprise software, a more expensive solution may be justified to ensure seamless integration and data exchange.

f. Other alternatives that can save costs include:
   i. Using a content delivery network (CDN): A CDN can help improve website performance by caching content in multiple locations around the world. This can reduce the load on the web servers and improve website speed for users. Cloudflare and Akamai are examples of CDN providers that can help reduce server costs.
   ii. Using a managed database service: Instead of setting up and managing a database server, using a managed database service can save on infrastructure costs. Examples of managed database services include Amazon RDS and Google Cloud SQL
   iii. Using open-source software: Instead of using proprietary software, using open-source software can save on licensing costs. Popular open-source software for web applications includes the LAMP stack (Linux, Apache, MySQL, PHP) and MEAN stack (MongoDB, Express, AngularJS, Node.js).