Name: Freya Jayant Vora

B00 Number: B00961402

Net ID: fr793929@dal.ca

Course: CSCI5409 - Adv Topic in Cloud Computing (Sec 01) - 2023/2024 Winter

Application Name: Task Management Application (ToDoApp)

Overview

ToDoApp is a task management app made to increase output and simplify task arrangement. The ToDoApp combines scalability, dependability, and security to provide customers with a smooth task management experience through the usage of several AWS services. The service Firebase Authentication, offered by Google Firebase, is used to handle user authentication. Users may safely sign in using a multitude of authentication methods, such as email and password, with Firebase Authentication, which also provides secure user identity management. The Spring boot backend services of the application are powered by Amazon EC2 instances. Installed on these instances, an elastic load balancer efficiently distributes incoming application traffic among both EC2 instances. This ensures fault tolerance and high availability of the backend services, enabling the application to execute user requests fast and continuously. An entirely managed solution for launching and scaling web applications and services, Amazon Elastic Beanstalk is used to implement the React front end of the application. All the task details are stored on RDS database, and I have created a backup of the RDS database using AWS Backup for fault tolerance. Furthermore, I have created a Virtual Private Cloud with three private subnets and two public subnets for security of my application. The RDS database is in the private subnets and EC2 instances on the public subnets. Also routing table is created to connect the EC2 to the internet gateway and thus make it publicly available. I have used AWS CloudFormation to provision the above resources.

Problem Description

Whether it's in an academic, professional, or personal context, people and teams frequently struggle to manage activities efficiently in a variety of scenarios. Task prioritization, project status tracking, and priority and urgency assessment are difficult for humans to accomplish without a centralized task management system. Missed deadlines, repeated attempts, and general inefficiency can be caused by this disorganization. Collaboration further gets difficult when members of the team find it difficult to assign duties, communicate, or provide updates on time. Expect delays and misunderstandings as a result of this lack of collaboration. With its all-inclusive approach to improving job prioritization, work tracking, and transparency, task management software seeks to address these problems.

Project Design

i) Chosen Services

Service Category	Services Used	Application
Compute	AWS EC2	This EC2 instance serves as the hosting environment for the backend services, providing the necessary computing resources to run the Spring Boot application.
Compute	AWS Elastic Beanstalk	AWS Elastic Beanstalk is utilized for deploying the frontend of the application. The frontend is developed using React.
Storage	AWS Relational Database Service	AWS Relational Database Service (RDS) is utilized in our task management application for storing task details.
Network	AWS Virtual Private Cloud (VPC)	Private and Public subnet created in VPC for security of the application.
General	AWS Backup	To backup the data stored in RDS.
General	AWS Elastic Load Balancing	Load Balancing is utilized to accommodate varying levels of traffic, optimizing performance, and minimizing latency for users accessing the application.
	AWS CloudFormation	For Provisioning resources on cloud automatically.

Table 1: Chosen Services.

ii) Comparison among Alternative Services

Service Used	Alternative Service	Justification of Services Used
AWS RDS	Google Cloud SQL	Offers a wider range of database engine options, including MySQL, PostgreSQL, and SQL Server.
AWS VPC	Google Cloud VPC	Provides robust networking features such as subnet creation, route tables, security groups, and VPN connections. This feature-rich environment allows for greater control and customization of network.
AWS Backup	Google Cloud Backup	Flexible retention policies.
AWS Elastic Load Balancer	Google Cloud Load Balancing	Elastic Load Balancer provides built-in support for various types of load balancing algorithms, including round-robin, least connections, and IP hash.

Table 2: Alternative of Services used with justification.

Deployment Model

I have chosen public cloud deployment due to following reasons:

- Scalability: It's easy to raise or lower capacity in response to demand because public cloud providers offer practically infinite scalability. Applications can efficiently handle workload fluctuations in this way without over- or under-provisioning their resources.

- Cost-effectiveness: Pay-as-you-go models, which enable users to only pay for the resources they really use, are frequently offered by public cloud services. As a result, the running expenses for upkeep of the on-site machinery are decreased.

- Global reach: AWS runs data centers all over the world. This worldwide reach facilitates rapid content retrieval and assures a consistent user experience for users everywhere.

- Reliability and resilience: High levels of redundancy, fault tolerance, and disaster recovery capabilities are offered by public cloud providers. This helps to maintain

operations and cut down on downtime.

Delivery Model

I have chosen Infrastructure as a service delivery model due to following reasons:

- Cost-saving: Pay-as-you-go allows for reduced expenses when services are used less.
- Flexibility and Scalability: IaaS gives companies the opportunity to adjust the size of their IT infrastructure in response to demand.
- Scalability: Ability to scale dynamically as per application needs.

Architecture Diagram

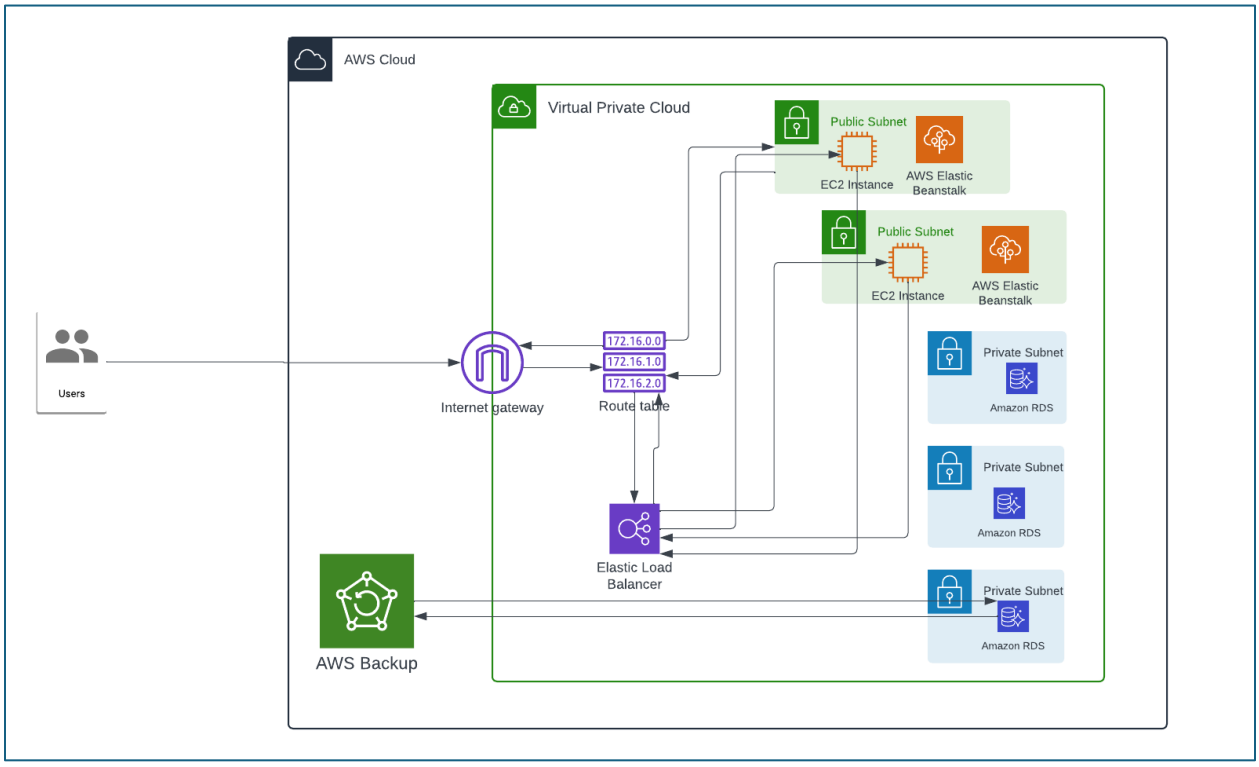

Figure 1: Application architecture, created using Lucid chart [1]

I have deployed my frontend to Elastic Beanstalk using docker image and backend is deployed to EC2 instances by copying the backend application jar file to the EC2 Instance and then executing the jar there.

Security Analysis

I am enhancing infrastructure security through the utilization of virtual private clouds (VPCs) and private subnets. By isolating the network from the public internet, VPC guards against external threats and unauthorized access. Private subnet resources improve data confidentiality by reducing direct access to essential components and reduce the attack surface for potential cyber threats. Important assets are protected from any security breaches and unauthorized access thanks to the robust security foundation.

Cost Metrics Analysis

- Up-front expenses:

  There are upfront expenses associated with the creation of resources including VPCs, RDS instances, AWS Backup configurations, Elastic Beanstalk instances for front-end deployment, EC2 instances for back-end deployment, and Elastic Load Balancers. These upfront expenses are related to setting up and provisioning the required infrastructure parts. However, since most of these services follow a pay-as-you-go model, the up-front costs are relatively minimal compared to traditional on-premises infrastructure setups.

- Ongoing expenses:

  As a result of the pay-as-you-go model, the system's ongoing expenses are determined by how resources are used and consumed. Charges for EC2 instances, RDS usage, data transport, storage consumption, and any other services used in the AWS ecosystem are included in this. Over time, flexibility and cost-effectiveness are provided by continuous charges that are directly linked to the system's usage by only charging for the resources used. The initial setup and configuration of infrastructure components constitute the majority of the upfront expenditures overall. Ongoing costs, on the other hand, are determined by resource utilization and are incurred according to the pay-as-you-go model that is commonly found in cloud computing environments.

- Additional Costs

  Additional expenses for auto-scaling features or running extra EC2 instances to manage higher traffic loads. There can be expenses involved in recovering data from backups in the case of AWS Backup.

- Cost Saving Alternatives
  a. Reserved Instances: Compared to on-demand pricing, using AWS Reserved Instances (RIs) for EC2 instances and RDS databases can result in significant cost savings. In return for committing to a particular instance type, operating system, and region for a term of one or three years, RIs offer discounted hourly pricing.
  b. Autoscaling Policies: It is possible to maximize resource utilization and reduce expenses by implementing autoscaling policies based on workload characteristics and demand patterns. The process of autoscaling modifies the quantity of EC2

instances or containers by automatically adjusting it in response to predetermined parameters like CPU utilization, network traffic, or application demand.

Future Scope

This application can be further extended to include groups/teams for the task, record audio to text description of tasks using AWS Transcribe like applications.

Conclusion

In summary, the thorough examination of the application's features, architecture, security protocols, and cost metrics highlights how well it meets user needs while guaranteeing scalability, dependability, and cost-effectiveness. A strong and secure architecture is established by utilizing AWS services like EC2, RDS, Elastic Beanstalk, and Elastic Load Balancer in conjunction with security measures like VPC and private subnets. In addition, the application's simple and clean user interface ensures user satisfaction and engagement through its user-friendly interface and collaboration capabilities. Proactive cost management tactics also guarantee cost-effectiveness and budget alignment. Examples of these tactics are pay-as-you-go pricing structures and optimization techniques. The application's overall success in providing a seamless and valuable experience for users and stakeholders alike can be attributed to its architecture, functionality, security, and cost considerations.

References

[1] "Intelligent Diagramming", Lucidchart, [Online], Available: https://www.lucidchart.com/pages/. [Accessed 07 April 2024].

[2] "What is Amazon VPC?," Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html. [Accessed 07 April 2024].

[3] "What is Amazon EC2?," Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html. [Accessed 08 April 2024].

[4] "What is Amazon Relational Database Service (Amazon RDS)?," Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html. [Accessed 08 April 2024].

[5] "Getting started using Elastic Beanstalk," Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/GettingStarted.html. [Accessed 08 April 2024].

[6] "Load Balancer Getting Started," Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/load-balancer-getting-started.html. [Accessed 08 April 2024].

[7] "Getting Started with AWS Backup," Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/aws-backup/latest/devguide/getting-started.html. [Accessed 08 April 2024].

[8] S. Narula, "Guide to Adding Google Login to a React App," LogRocket Blog. [Online]. Available: https://blog.logrocket.com/guide-adding-google-login-react-app/. [Accessed 08 April 2024].

[9] "AWS RDS vs. Cloud SQL: A Comprehensive Comparison," Economize Cloud. [Online]. Available: https://blog.economize.cloud/aws-rds-vs-cloud-sql/. [Accessed 08 April 2024].

[10] S. Narula, "Cloud Network Showdown: Comparing AWS VPC vs. GCP VPC," Medium. [Online]. Available: https://medium.com/@samarthnarula13/cloud-network-showdown-comparing-aws-vpc-vs-gcp-vpc-b5719e17b0f0. [Accessed 08 April 2024].

[11] "AWS Elastic Load Balancing vs. Google Cloud Load Balancing," StackShare. [Online]. Available: https://stackshare.io/stackups/aws-elastic-load-balancing-vs-google-cloud-load-balancing. [Accessed 08 April 2024].