FACULTY OF COMPUTER SCIENCE

Assignment 3

In
The Class of

CSCI 5411: ADVANCED CLOUD ARCHITECTING

by

Shreya Kapoor (B00957587)

Table of Contents

Iteration 1

Ques 1. For an international news agency dealing with high volumes of multilingual text data and requiring advanced text search capabilities for their Java-based application, which Amazon RDS database engine would you choose and why? Consider the need for high availability, scalability, and distributed access in your answer

When handling vast amounts of multilingual data with the need for advanced text search capabilities, Amazon Aurora with the PostgreSQL-compatible edition as the Amazon RDS database engine is an ideal choice.

1. **High Availability:** Amazon Aurora ensures high availability through Multi-AZ (Availability Zone) deployments, backed by a 99.99% uptime SLA. It offers global replication with cross-region disaster recovery in under one minute. By replicating data across multiple AZs, it maintains database availability even during infrastructure failures. This reliability is critical for a news agency that relies on uninterrupted access to its database [1].

2. **Scalability:** Amazon Aurora can effortlessly scale to accommodate increasing data volumes and access demands. It supports both horizontal scaling by adding read replicas to distribute read loads and vertical scaling by adjusting compute and storage resources. This flexibility ensures that the database can grow seamlessly as the news agency's workload expands [1].

3. **Distributed Access:** With a globally dispersed team of journalists and analysts, the news agency requires a database solution that facilitates distributed access. Amazon Aurora's cross-region replication enables the database to be replicated to other regions, providing local access and reducing latency for geographically diverse users [1].

Additionally, the PostgreSQL-compatible edition of Amazon Aurora offers robust text search capabilities, including full-text search, fuzzy matching, and linguistic features. These tools are well-suited for analyzing and exploring large volumes of multilingual text data, making it an excellent fit for the international news agency’s unique needs. This solution provides a reliable and efficient database for their Java-based application.

Ques 2. Given that the initial user base is around 50,000 with a data inflow of approximately 50GB/day, and you anticipate the user base will grow to approximately 200,000 within a year, with an estimated data inflow growth rate of 10% per quarter. Additionally, the application is read-intensive with frequent complex queries. What instance type and storage capacity would you select for the above selected RDS database at launch, and how would you plan for the projected growth?

Instance Type Selection: Given the expected expansion and the necessity for excellent performance, it is recommended to choose an instance type with ample compute power and

memory. RDS memory-optimized or compute-optimized instances, such as the R5 or R6 series, are suitable choices. These instances provide an optimal balance of CPU, memory, and network performance, enabling efficient query processing [2].

Storage Capacity Planning: The data intake rate and anticipated growth can be used to determine the required storage capacity. With an initial data intake of 50GB per day and a quarterly growth rate of 10%, we can estimate the storage requirements as follows:

- **Initial Storage:** 50GB/day * 365 days = 18,250GB (for the first year)
- **Projected Growth:**
  - Growth rate per quarter = 10% * 50GB/day = 5GB/day
  - Projected growth for the year = 5GB/day * 365 days = 1,825GB (for the first year)
- **Total Storage:** 18,250GB (initial storage) + 1,825GB (projected growth) = 20,075GB (for the first year)

Based on this estimate, it is advisable to allocate approximately 20,075GB of storage space for the first year. Additionally, to accommodate any unexpected increases in data influx, it is prudent to provision extra storage beyond the projected growth.

Performance Monitoring and Optimization: When selecting the instance type and storage size, it is crucial to consider performance monitoring and optimization strategies. To manage increasing demand, it may be necessary to scale up the instance type or expand storage capacity. Overall, choosing an appropriate instance type and allocating sufficient storage capacity while considering projected growth will ensure that the RDS database can efficiently handle the expanding user base, data influx, and complex queries. Additionally, RDS supports manual horizontal scaling (by adding read replicas) and manual vertical scaling (by upgrading or downgrading existing instances) [3].

By implementing these strategies, the RDS database will be well-equipped to manage the growing demands of the application.

Ques 3. How would you devise a backup and recovery strategy capable of restoring at least a week's worth of data?

To establish a reliable backup and recovery plan capable of restoring data from the past week for our RDS database, following are the steps:

1. **Multi-AZ Deployment:** Use Amazon RDS Multi-AZ deployment, which keeps an up-to-date copy of our main database in a different zone. If our main database fails, RDS switches to the standby instance seamlessly, reducing downtime and ensuring data availability.

2. **Automated Backups:** Enable regular automated backups for our RDS instance. These backups are scheduled to run automatically (like daily or every few hours), capturing all database changes to ensure comprehensive recovery options.

3. **Retention Period:** Keep automated backups for at least a week. This ensures you can restore data as it was up to seven days ago. Adjust this period based on our

business needs and compliance requirements.

4. **Database Snapshots:** Take manual database snapshots regularly. These snapshots capture our entire database at a specific point in time, providing additional recovery options if needed to restore to a specific state.

5. **Offsite Backup Strategy:** Implement an offsite backup plan by copying backups or snapshots to another AWS region or storage service like Amazon S3. Offsite backups add an extra layer of protection against regional outages or unexpected events.

6. **Regular Testing:** Test our restore process regularly to ensure it works as expected. This helps identify any issues with our backup plan and ensures our data is recoverable in case of emergencies.

7. **Monitoring and Alerts:** Set up monitoring and alerts to track backup operations. Receive alerts for backup completion and any failures, allowing you to address issues promptly and maintain the reliability of our backup strategy [4].

Ques 4. The organization requires a secure architecture that provides data encryption and access control. What would be your strategy?

The following technique can be used to establish a safe architecture with data encryption and access control

1. **Data Encryption at Rest:** Encrypt the storage volumes where data is stored using Amazon RDS encryption. This ensures data remains encrypted and protected against unauthorized access if physical theft or direct storage access occurs [5].

2. **Data Encryption in Transit:** Use SSL/TLS encryption for database connections. This secures data transmitted between the application and the database, protecting it from being intercepted or altered during transmission [5].

3. **Precise Access Control:** Implement detailed access controls to limit database access. Use permissions at the database and object levels to assign appropriate rights to users and roles. Follow the principle of least privilege to ensure users only access necessary data and functions.

4. **Network Security Measures:** Deploy the database within a Virtual Private Cloud (VPC) and configure network security. Control inbound and outbound traffic using security groups and network ACLs. Consider secure connectivity options like VPC peering or VPN connections.

5. **Strong Authentication and Authorization:** Employ robust authentication methods for database access. Implement two-factor authentication (2FA) for user logins. Set up auditing and monitoring to track user activities effectively.

6. **Monitoring and Logging:** Enable logging and auditing tools to monitor and record user actions such as logins, data changes, and access modifications. Centralize logs and regularly review them to detect any unauthorized activities.

7. **Regular Security Updates:** Stay current with the latest security patches and updates for the database engine. Monitor security alerts and apply patches promptly to fix any identified vulnerabilities.

8. **Security Audits and Testing:** Conduct routine security audits and penetration testing to identify and address potential security weaknesses. Engage third-party security experts or use automated tools to conduct thorough assessments of our architecture.

Iteration 2

Ques 1. Design a Virtual Private Cloud (VPC) for the agency's AWS infrastructure. Specifically, how many subnets would you create, in which Availability Zones (AZs), and why?

The following approach can be used to construct a Virtual Private Cloud (VPC) for the news agency's AWS infrastructure:

- **Subnet Layout:** Create multiple subnets to balance accessibility and security within the VPC. Subnets segment and isolate different parts of the network.

- **Availability Zones (AZs):** Spread subnets across multiple Availability Zones (AZs) to ensure high availability and resilience against AZ-level failures. AZs are separate data centers within a region, enhancing reliability by distributing resources.

- **Public Subnets:** Establish public subnets in each AZ. These subnets are linked to a route table containing an Internet Gateway (IGW). Public subnets enable EC2 instances running reporting tasks to connect directly to the internet securely.

- **Private Subnets:** Deploy private subnets in each AZ. These subnets lack direct internet connectivity and are suitable for hosting backend services, databases, or sensitive resources accessed only by authorized users [6].

It's recommended to initially create at least two subnets in different AZs: one public subnet for EC2 instances needing internet access for reporting tasks, and one private subnet for internal services. This separation enhances security and management control. Each subnet should have its own route table defining outbound traffic rules. As the infrastructure expands, additional subnets and AZs can be added for scalability and resilience.

Ques 2. How would you configure Security Groups for your RDS instances and other EC2 instances that need access to the internet, considering both inbound and outbound rules?

When configuring Security Groups for RDS instances and EC2 instances needing internet access, it's crucial to manage both inbound and outbound traffic effectively to uphold network security. Following is the approach:

Setting Up Security Groups:

- **Inbound Rules:**
  - **RDS Instances:** Define inbound rules to allow traffic from specific sources or IP ranges required to access the RDS instances. This ensures that only authorized entities, such as particular EC2 instances or designated IP addresses, can connect to the RDS instances [7].
  - **EC2 Instances:** Configure inbound rules for EC2 instances based on the applications they host. Permit inbound connections on specific ports necessary for these applications to function properly.

- **Outbound Rules:**
  - **RDS Instances:** By default, RDS instances can send outbound traffic to the internet. Verify that the Security Group associated with RDS allows the necessary outbound traffic for the database operations.
  - **EC2 Instances:** Specify outbound rules for EC2 instances according to their operational needs. Permit outbound connections to specific IP addresses or ranges required for accessing external services or APIs essential for application functionality.

Applying Least Privilege:

- Implement the principle of least privilege by restricting inbound and outbound traffic to only what is necessary for the functioning of RDS and EC2 instances. Avoid overly permissive rules that could expose instances to unnecessary risks.

Regular Review and Updates:

- Regularly review Security Group rules to ensure they align with current application requirements and security policies. Update rules promptly to accommodate any changes in application needs or security standards.

By configuring Security Groups with appropriate inbound and outbound policies, the organization can effectively manage and restrict network traffic to and from RDS and EC2 instances. This approach ensures secure and controlled access to the internet while minimizing potential security vulnerabilities.

Ques 3. Discuss the rules you would set for Network Access Control Lists (NACLs) at the subnet level to provide an additional layer of security. How would these rules differ for public and private subnets?

When creating Network Access Control Lists (NACLs) at the subnet level to provide an extra layer of security, the rules for public and private subnets differ based on their network access needs.

1. Public Subnets

   Inbound Rules:

- Typically, public subnets allow inbound traffic from the internet to resources within the subnet.
- Common rules include allowing HTTP (port 80) and HTTPS (port 443) traffic for web servers, and SSH (port 22) for administrative access.

Outbound Rules:

- Public subnets generally allow outbound traffic to the internet, enabling resources within the subnet to connect to external services or APIs.
- Standard protocols like HTTP, HTTPS, and DNS (port 53) are typically allowed.

2. Private Subnets

Inbound Rules:

- Private subnets usually have stricter inbound rules to block direct access from the internet.
- Inbound traffic may be restricted to specific trusted sources, such as other internal subnets, VPN connections, or a bastion host/jump server.
- Only specific protocols and ports required by internal applications are allowed, while unwanted external access is blocked.

Outbound Rules:

- Outbound rules in private subnets are more relaxed to permit necessary communication to trusted destinations.
- Outbound connections might be allowed to internal subnets, databases, APIs, or other required services.
- These rules are carefully defined to prevent unauthorized access or data leaks.

3. Additional Considerations

- **Deny Rules:** Both public and private subnets should include explicit deny rules for unnecessary protocols or ports. This reduces the attack surface and prevents unauthorized access attempts.
- **Logging:** Enable logging in NACLs to monitor and analyze network traffic for security and troubleshooting purposes [8].

Ques 4. As a proactive measure, the news agency wants to minimize the impact of a potential Distributed Denial of Service (DDoS) attack. Suggest a mitigation strategy using AWS services. How would this protect your VPC and the resources within it?

To reduce the impact of a potential Distributed Denial of Service (DDoS) attack, the news agency can use AWS services as part of the following mitigation strategy:

- **AWS Shield:** Enable AWS Shield, a managed DDoS protection service provided by AWS. It automatically protects against common and large-scale DDoS attacks.

- **AWS WAF (Web Application Firewall):** Use AWS WAF to filter and block malicious traffic at the application layer. Set up custom rules to detect and stop known attack patterns, such as HTTP floods or SQL injection attempts.

- **Elastic Load Balancer (ELB):** Deploy an Elastic Load Balancer in front of the agency's web servers, like Application Load Balancer (ALB) or Network Load Balancer (NLB). Load balancers distribute traffic evenly across multiple instances, making it harder for attackers to overwhelm a single server.

- **Network ACLs and Security Groups:** Configure Network ACLs and Security Groups to control and limit network traffic at the subnet and instance levels. Setting appropriate inbound and outbound rules helps filter out malicious traffic and allows only legitimate connections.

- **Auto Scaling:** Implement Auto Scaling to automatically adjust the number of EC2 instances based on demand. This helps handle sudden traffic spikes and provides extra resilience against DDoS attacks.

- **VPC Traffic Mirroring:** Enable VPC Traffic Mirroring to capture and analyze network traffic in real-time. This allows for advanced traffic analysis and monitoring, helping detect and mitigate malicious behavior.

- **Amazon CloudFront:** Use Amazon CloudFront, a content delivery network service, to distribute content globally and protect against large-scale DDoS attacks. CloudFront can absorb and mitigate attack traffic, reducing the load on the agency's infrastructure.

By using this strategy, the news agency can leverage AWS's strong DDoS protection features. This helps protect the VPC and its resources by filtering out unwanted traffic, distributing the load, and automatically scaling capacity to withstand and mitigate DDoS attacks [9].

Iteration 3

Ques 1. In light of a major world event, user demand and data inflow are projected to increase significantly (triple the user base and a ten-fold increase in data inflow). How would you modify the architecture to accommodate this surge?

To handle the rise in user demand and data intake caused by a significant global event, the architecture can be adjusted as follows:

1. Scaling Compute Resources:

    - **Increase Compute Capacity:** Add more EC2 instances or increase the size of existing ones to meet the higher user demand. This can be done

through vertical scaling (upgrading instance size) or horizontal scaling (adding more instances).
- **Auto Scaling:** Set up Auto Scaling to automatically adjust the number of instances based on demand. This ensures that resources are available as needed and scales down when demand decreases.

2. Database Scaling:

   - **Expand Database Capacity:** Scale the database infrastructure to handle the increased data inflow and user base. This can involve upgrading the existing database instance or using database sharding to distribute data across multiple instances.
   - **Database Caching:** Use caching solutions like Amazon ElastiCache to reduce database load and improve response times. Frequently requested data can be cached to decrease demand on the database and enhance system performance.

3. Content Delivery Network (CDN):

   - **Amazon CloudFront:** Use Amazon CloudFront to deliver content closer to users' locations, reducing latency. This ensures that the increased number of users can access content quickly and efficiently, regardless of their location.

4. Auto Scaling:

   - **Dynamic Resource Scaling:** Configure Auto Scaling groups to automatically adjust the number of instances based on demand. This enables the system to scale up or down as needed, ensuring optimal performance and cost-efficiency.

5. Monitoring and Alerting:

   - **Amazon CloudWatch:** Implement monitoring and alerting with Amazon CloudWatch to track system performance, resource usage, and potential bottlenecks. This allows for proactive detection and resolution of issues during peak periods.

6. Performance Optimization:

   - **Continuous Optimization:** Regularly analyze and optimize application and infrastructure performance to manage the increased load efficiently. Fine-tune database queries, optimize code, and use performance monitoring tools to identify and address any bottlenecks.

By making these changes to the architecture, the system will be better equipped to handle the increased user demand and data intake caused by a major global event. The improved capacity, scalability, and redundancy will ensure a consistent user experience while maintaining application availability and performance under heavy load conditions.

Ques 2. How would you utilize RDS Performance Insights to monitor your database performance and identify bottlenecks?

To effectively use RDS Performance Insights for monitoring the database performance and pinpointing bottlenecks, follow these steps:

1. **Enable Performance Insights:**

   - **How to Enable:** Turn on Performance Insights for our RDS instance through the AWS Management Console, AWS CLI, or AWS SDKs.
   - **Purpose:** This will start collecting database performance metrics.

2. **Access the Dashboard:**

   - **Navigate to the Dashboard:** Once enabled, go to the Performance Insights dashboard.
   - **What You'll See:** This dashboard shows visualizations and key metrics such as CPU usage, latency, and active sessions.

3. **Examine Performance Metrics:**

   - **Review Metrics:** Analyze the performance data displayed on the dashboard.
   - **Look for Patterns:** Identify unusual patterns or anomalies in metrics like CPU usage, query latency, and concurrent sessions that might indicate bottlenecks.

4. **Identify Top Resource Consumers:**

   - **Find Heavy Users:** Performance Insights highlights the top queries, users, or applications consuming the most resources.
   - **Why It Matters:** Identifying these can help you locate potential performance issues and areas needing optimization.

5. **Dive into Query Details:**

   - **Investigate Queries:** Drill down into the details of individual queries.
   - **Check Execution Plans:** Look at execution plans, resource usage, and the duration of long-running or resource-intensive queries to find problematic ones.

6. **Follow Recommendations:**

   - **Get Suggestions:** Performance Insights might provide recommendations based on the identified patterns and anomalies.
   - **Optimize Based on Advice:** These can include query optimizations, creating indexes, or configuration changes. Implementing these can help resolve bottlenecks.

7. **Set Performance Baselines:**

    - **Establish Baselines:** Use Performance Insights to set performance benchmarks.
    - **Monitor Over Time:** Continuously monitor and compare performance against these baselines to detect deviations or trends for proactive management.

8. **Configure Alerts:**

    - **Set Up Alarms:** Use CloudWatch Alarms to get notifications when performance metrics exceed predefined thresholds.
    - **Be Proactive:** This enables you to quickly respond to and address performance issues.

9. **Ongoing Monitoring and Optimization:**

- **Regular Reviews:** Continuously review Performance Insights and other performance metrics.
- **Optimize Continuously:** Based on the insights, keep optimizing database configurations, query patterns, and resource allocations to maintain optimal performance [10].

Ques 3. When you design your RDS database architecture, how would you ensure data integrity and consistency?

To ensure strong data integrity and consistency in our RDS database architecture, it's essential to implement the following best practices:

1. **Choose Appropriate Data Types:** Selecting the correct data type for each column in our database schema is crucial. This ensures that data is stored accurately and prevents issues like data truncation or incorrect data formats that can lead to corruption.

2. **Define Constraints:** Utilize constraints such as primary keys, unique keys, foreign keys, and check constraints. These enforce rules at the database level, ensuring that only valid and consistent data is inserted or modified. Constraints also help maintain relational integrity across our database tables.

3. **Enable Database-Level Constraints:** Leverage built-in database features like unique constraints and referential integrity constraints. These mechanisms enforce data consistency and prevent anomalies such as orphaned records or inconsistent data relationships.

4. **Implement Transactions:** Transactions group database operations into logical units. This ensures that all operations within a transaction either succeed or fail together, maintaining data integrity even during concurrent access or system failures.

5. **Ensure ACID Compliance:** Adhere to ACID (Atomicity, Consistency, Isolation, Durability) principles in our database design. Atomicity ensures that transactions are indivisible, Consistency guarantees that only valid data is committed, Isolation prevents interference between concurrent transactions, and Durability ensures that committed transactions are permanent.

6. **Apply Data Validation:** Implement robust data validation at the application level before data is sent to the database. Validate inputs to ensure they meet expected criteria, including data format validation, input sanitization, and business rule validation. This prevents the insertion of invalid or inconsistent data into the database.

7. **Perform Regular Data Backups:** Establish a scheduled backup strategy to create backups of our database at regular intervals. Backups serve as a safeguard against data loss due to accidental deletion, corruption, or system failures. They also provide recovery points to restore data to a known good state.

8. **Monitor and Audit:** Implement comprehensive monitoring and auditing mechanisms to track database activities and detect anomalies or deviations from expected behavior. Monitoring helps identify performance bottlenecks, while auditing ensures compliance with data integrity policies and regulations.

9. **Implement Data Replication and Failover:** Use features like Multi-AZ deployments and read replicas to enhance data availability and redundancy. Data replication ensures that changes made to one database instance are synchronized across others, providing fault tolerance and minimizing downtime during instance failures.

10. **Routine Maintenance:** Regularly perform maintenance tasks such as index optimization, query optimization, and database statistics updates. These activities optimize database performance, reduce query execution times, and ensure efficient use of database resources while maintaining data integrity.

Iteration 4

Ques 1. A hurricane threatens the location of one of your major data centers. Create a disaster recovery plan that includes replication of the RDS database across regions. How would you incorporate your plan on this?

To prepare for the threat of a hurricane or any other major disaster that could impact our primary data center, it's crucial to establish a comprehensive disaster recovery plan that includes replicating our RDS database across different AWS regions. Following is how we can effectively implement this strategy:

1. **Select a Remote Region:** Begin by identifying and selecting an AWS region that is geographically distant from the location of our primary data center. This region should ideally be less susceptible to the same natural disasters, ensuring continuity of operations even in the face of regional disruptions.

2. **Set Up a Backup RDS Instance:** Create a secondary RDS instance in the chosen remote region. This instance serves as a standby or backup for our primary RDS instance located in the potentially affected region. By configuring this backup, we ensure that critical data and services can quickly resume from an unaffected location.

3. **Configure Replication:** Enable replication between the primary RDS instance in our primary region and the standby instance in the remote disaster recovery region. AWS provides tools like Multi-AZ deployments and read replicas that facilitate robust data replication, ensuring that updates and changes to the primary database are promptly mirrored to the standby instance.

4. **Monitor Replication Status:** Implement continuous monitoring of the replication process between the primary and standby instances. Monitoring ensures that data synchronization remains consistent and that any discrepancies or issues are promptly identified and addressed. This proactive approach helps maintain data integrity and readiness for failover.

5. **Automate Failover Procedures:** Implement automated failover procedures to streamline the process of switching from the primary RDS instance to the standby instance in the event of a disaster. Automated failover can be orchestrated using AWS services such as Route 53 for DNS failover or AWS Lambda functions for triggering failover actions based on predefined criteria.

6. **Regularly Test the Disaster Recovery Plan:** Conduct regular tests and drills of the disaster recovery plan to validate its effectiveness. Testing scenarios should include simulating various disaster scenarios, initiating failover procedures, verifying data consistency, and evaluating the time required to recover operations (RTO) and the acceptable data loss (RPO). These tests ensure that the plan is robust and reliable when needed most [11].

Ques 2. There is a need to develop a failover mechanism using AWS RDS Multi-AZ deployments to ensure minimal downtime during disaster recovery scenarios. How does your mechanism guarantee minimal service disruption?

To guarantee minimal service disruption during disaster recovery scenarios, our failover mechanism harnesses the robust capabilities of AWS RDS Multi-AZ deployments. Here's how each component ensures seamless continuity:

1. **Multi-AZ Deployment:** AWS automatically sets up a standby replica of our primary database in a different Availability Zone (AZ). This redundancy ensures that in case of an AZ failure or during planned maintenance, there is always a synchronized standby ready to take over as the primary database. The failover process is automatic and swift, triggered by AWS when needed, minimizing downtime.

2. **Synchronous Replication:** Within Multi-AZ deployments, data changes made to the primary database are immediately replicated synchronously to the standby replica. This ensures that both databases are consistently up-to-date. When a failover occurs, the standby replica has the latest data, ensuring no loss of information and enabling a smooth transition to continue operations without interruption.

3. **Automatic Failover:** AWS RDS monitors the health of the primary database continuously. Upon detecting a failure or maintenance event that impacts the primary database, AWS automatically initiates a failover. This process involves promoting the standby replica to become the new primary database. AWS updates the DNS endpoint associated with the RDS instance to point to the new primary, redirecting application traffic seamlessly.

4. **DNS Endpoint Management:** RDS provides a DNS endpoint that abstracts the underlying database infrastructure details. During failover, AWS updates this DNS endpoint to point to the new primary database. Applications can continue to access the database using the same endpoint, eliminating the need for manual intervention in updating connection strings or configurations.

5. **Application Resilience:** Designing applications to handle failovers gracefully is crucial. Implementing retry mechanisms and managing connection timeouts allows applications to automatically reconnect to the database after failover. By incorporating these practices, applications can maintain continuity even during transient disruptions caused by failover events [11].

References

[1] "Amazon Aurora features," *Amazon.com*. [Online]. Available: https://aws.amazon.com/rds/aurora/features/. [Accessed: July 11, 2024].

[2] "Amazon RDS instance types," *Amazon.com*. [Online]. Available: https://aws.amazon.com/rds/instance-types/. [Accessed: July 11, 2024].

[3] “Scaling Your Amazon RDS Instance Vertically and Horizontally,” *Amazon.com*. [Online]. Available: https://aws.amazon.com/blogs/database/scaling-your-amazon-rds-instance-vertically-and-horizontally/. [Accessed: July 11, 2024].

[4] “Amazon RDS Backup and Restore,” *Amazon.com*. [Online]. Available: https://aws.amazon.com/rds/features/backup/. [Accessed: July 11, 2024].

[5] “Encrypting Data-at-Rest and Data-in-Transit,” *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/whitepapers/latest/logical-separation/encrypting-data-at-rest-and--in-transit.html. [Accessed: July 11, 2024].

[6] “Subnets for your VPC,” *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/configure-subnets.html. [Accessed: July 11, 2024].

[7] “Controlling access with security groups,” *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.RDSSecurityGroups.html. [Accessed: July 11, 2024].

[8] “Control subnet traffic with network access control lists,” *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html. [Accessed: July 11, 2024].

[9] “Mitigation techniques,” *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/whitepapers/latest/aws-best-practices-ddos-resiliency/mitigation-techniques.html#:~:text=AWS%20Shield%20DDoS%20mitigation%20systems,them%20to%20the%20protected%20service. [Accessed: July 11, 2024].

[10] “Performance Insights,” *Amazon.com*. [Online]. Available: https://aws.amazon.com/rds/performance-insights/. [Accessed: July 11, 2024].

[11] “Implementing a disaster recovery strategy with Amazon RDS,” *Amazon.com*. [Online]. Available: https://aws.amazon.com/blogs/database/implementing-a-disaster-recovery-strategy-with-amazon-rds/. [Accessed: July 11, 2024].