Project – GoGreen Insurance Company

CIS 3367

By: Cassandra Barrera , Jeremy Rodriguez , Jeremy Duong , Heeya Paul

Table of Contents

Solution – Identify AWS Services

The list provided below some of the tools needed to meet the company's project objectives and requirements. We are able to cover computing, storage and networking power featuring scalability alongside monitoring and automation. Having these services will ensure that GoGreen's architecture is secure and reliable. However, it does not hurt to provide additional services that we think may benefit GoGreen in the long run.

1. ***Amazon EC2***

- Having multiple instances allows us to have scalability and computing capacity within the cloud. We can tag and assign the EC2 instances for GoGreen's web, application and database layers. This will be followed up by Auto Scaling in order to meet the flexibility they require and meet all kinds of traffic demands. We must also be cautious of over-provisioning to avoid unnecessary costs.

2. ***Amazon RDS***

- As part of our solution, we must have a well-managed service that is able to support multiple database languages. Amazon RDS allows us to deploy in multiple Availability Zones, which ensures GoGreen with high availability and automation.

3. ***Amazon S3***

- Amazon offers services that allows companies to store their files. Since we are considering it to be cost-effective, we can use Amazon's S3 service in order to store documents, images, and other static files that allows us to safely transition data.

4. ***Elastic Load Balancer***

- An ELB allows for the consistent distribution of traffic across multiple EC2 instances. It ensures that no instance becomes overloaded with traffic, which may disrupt our services. We must consider multiple ELBs: An Application Load Balancer for the Web Tier and a Network Load Balancer, which specializes in handling high-output

5. ***AWS Auto Scaling***

- Using the Auto Scaling service allows our architecture to automatically scale itself according to demand. In this case, it ensures that GoGreen's infrastructure is always fit for the varying workload. It also ensures that we are being cost-effective.

6. ***AWS KMS***

- Has the ability to manage keys for secure storage of data at rest. Able to be integrated within S3, RDS, and our EBS, which meets the requirements for encryption for GoGreen

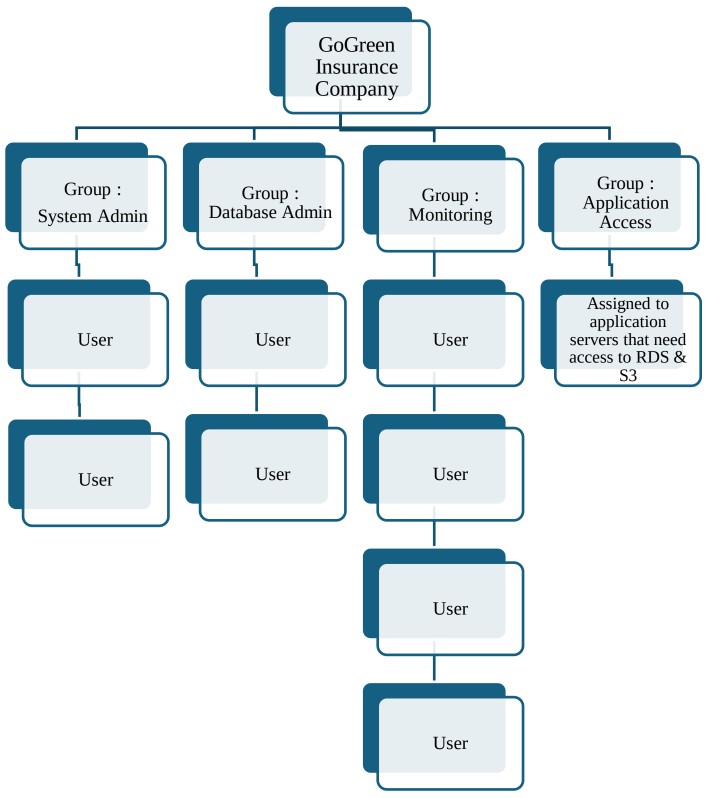
7. ***AWS CloudWatch***

- CloudWatch gives admins the ability to monitor resources and applications used for our solution. This is done through metrics and logs which allow the admin team to track performance and potential issues.

8. ***Amazon CloudFront***

- A CDN is able to cache and deliver content to its users, improving performance and reducing latency. It will be essential if GoGreen plans to have multiple distributions across different regions.

Solution – User Authentication Chart


GoGreen Insurance Company
Group : System Admin
Group : Database Admin
Group : Monitoring
Group : Application Access
User
User
User
User
User
User
User
User
Assigned to application servers that need access to RDS & S3

To align with AWS best practices for permissions, we've defined three user groups: System Administrators (2 users), Database Administrators (2 users), and Monitoring Group (4 users), all with access to infrastructure resources like EC2, S3, and RDS for the app.

Our System Admin and Database Admins we have two users to separate roles and responsibilities for better security and operational efficiency.

Our Monitoring Group includes 4 users due to the fact that typically monitoring responsibilities often require broader coverage and continuous oversight thus it's for 24/7 Coverage, Scalability, and Redundancy.

Solution – User Authentication Chart

Group/Role Name	Permissions
System Admin	Full access to AWS Management Console (EC2 , RDS , S3 ,IAM , & VPC) Require MFA for console login
Database Admin	Access to manage and configure RDS Databases , with permission to view and modify database instances . No access to non- database resources. Requires MFA for console login.
Monitoring	Read-only access to CloudWatch and CloudTrail to monitor performance metrics and access logs. The permissions is limited viewing metrics and receiving alert, without modification rights.
Application Access	Assigned to application servers that needs access to RDS & S3

The access permissions outlined above are designed to ensure security and effective role management.

System Admins have full access to critical AWS services, including EC2, RDS, S3, IAM, and VPC, with MFA required for console login to enhance security.

Database Admins are restricted to managing RDS databases, with permissions to view and modify instances but no access to non-database resources, also requiring MFA.Thus it also increases redundancy -which reduces the potential for a single point of failure. It limits the scope of access for any one user, aligning with the principle of least privilege

Monitoring roles have read-only access to CloudWatch and CloudTrail, allowing them to view metrics and receive alerts without the ability to make changes.

Lastly, Application Access is assigned to servers, granting necessary permissions to interact with RDS and S3 for application functionality.

Solution – User Authentication Solution Justification

Other Security Options	**Solution Justifications**
Should be at least 8 characters and have 1 uppercase, 1 lowercase, 1 special character, and a number.	***AWS IAM*:** IAM allows the user to set defined attributes for password policies. You're able to configure different policies that include rules such as minimum/maximum password length, character requirements, and expiration policies. In general, IAM ensures basic compliance in security processes.
Change passwords every 90 days and ensure that the previous three passwords can't be re used.	***AWS Secrets Manager:*** AWS Secrets Manager allows users to store and manage credentials in rotations. For example, it encrypts and stores old credentials after they have been used for a certain amount of time, and it makes sure that the same credentials are not being used repeatedly. AWS Secrets Manager will automatically rotate credentials and save them through encryption without interruption.
All administrators require programmatic access.	***AWS IAM:*** IAM allows the host to create new users who require any kind of access for their roles. IAM roles can be configured to have specific policies, which may enable them to have different permissions when compared to others. The host is also able to give other users temporary permissions if they so desire.
Administrator sign-in to the AWS Management Console requires the use of Virtual MFA.	***AWS IAM and MFA:*** As previously mentioned, IAM allows the user to create specific permissions for certain entities. Furthermore, it can force other users to use MFA to access the Management Console. This adds an extra layer of security.

To ensure strong security for user authentication, we employed AWS tools tailored to our specific needs. We used AWS Identity and Access Management (IAM) to establish and enforce password policies that require complexity. For credential management, we chose AWS Secrets Manager, which securely stores and automatically rotates credentials to prevent reuse and enhance security. To facilitate programmatic access for administrators, we managed this through IAM roles with customized permissions.

Additionally, IAM integrated with Virtual Multi-Factor Authentication (MFA) enforces multi-factor authentication for administrator console access, adding a crucial layer of security. Together, these services enhance compliance and effectively protect our systems.

Design: Web Tier Chart

Requirement	Solution
Architecture must be flexible and handle any peak in traffic or performance.	For this requirement , we're using Auto Scaling, Load Balancing, and the CloudFront service. EC2 Auto Scaling allows the system to adjust the number of EC2 instances based on traffic and performance needs, using predetermined thresholds to scale up or down. To ensure high availability, it's advisable to use multiple Availability Zones (AZs). This way, if one AZ fails, others can continue to operate and manage traffic effectively. The Elastic Load Balancer (ELB) distributes traffic evenly across instances, preventing overloads and monitoring the health of instances to maintain operational status. Additionally, integrating the CloudFront service enhances content delivery and reduces latency issues, ensuring the architecture can handle peak performance efficiently.
The Overall acceptable incoming network bandwidth is between 300 Mbps and 700 Mbps	To ensure the architecture withstands continuous traffic, an Application Load Balancer will evenly distribute traffic across instances to prevent bandwidth fluctuations. Autoscaling policies will be configured to meet traffic thresholds, ensuring scalability and reliability. To monitor the Load Balancer and network performance, Amazon CloudWatch will track key metrics like latency, bandwidth, and packet loss. If bandwidth falls outside the acceptable 300-700 Mbps range, CloudWatch Network Monitor allows for quick troubleshooting. Alarms can also be set for faster issue detection and resolution, minimizing impact on users. This approach ensures robust performance and effective monitoring.
Application administrators want to be notified by email if there are more than 100 “400 HTTP errors” per minute in the application.	Since our Admins must ensure continuous product availability and efficient issue resolution. We’ll be using Amazon CloudWatch by configuring metric filters on the Application Load Balancer to track 400-level HTTP errors in 1-minute intervals. To enhance responsiveness, we’ll also use Amazon SNS , which will send email notifications by creating a topic that routes error alerts to business or personal emails. When an alarm is triggered, admins receive immediate notifications, enabling quick action to address HTTP errors.
Web Tier Instances should be tagged as “Key=Name” and “Value = “web-tier"	To ensure that our instances are correctly tagged as their appropriate values, we will need to implement our AWS Auto Scaling using launch templates and configurations. When setting up the Auto Scaling, using Launch Templates, each tagged instance will be able to inherit their tagged values without manually adjusting them. Then, we’ll be using Lambda to trigger EventBridge for the tags.

For our Web Tier Layer , we’ll be ensuring that our architecture meet each requirement listed above. For example, for our architecture will be flexible and handle any peak in traffic so we’ll be relying on Auto Scaling, Load Balancing, and CloudFront. Auto Scaling dynamically adjusts our EC2 instances based on our traffic needs, while multiple Availability Zones (AZs) will provide us high availability and resilience.

The Elastic Load Balancer (ELB) will distribute traffic evenly and ensures operational health, with CloudFront reducing latency for efficient content delivery.

To maintain performance, our acceptable bandwidth is 300-700 Mbps, so they’ll be monitored via CloudWatch for key metrics like latency and packet loss. And alarms triggers for immediate notifications for quick resolution. For instance, admins are alerted via Amazon SNS if 400 HTTP errors exceed 100 per minute, ensuring prompt action to maintain availability.

And lastly , to ensure that our Web Tier instances are tagged as Key=Name and Value=app-tier, ours tags will be applied automatically by using Launch Templates with pre-defined tags by configuring a Lambda function triggered by Event Bridge to apply tags dynamically.

Design: Application Tier Solution Justifications

Requirement	Solution
Architecture must be flexible and handle any peak in traffic or performance	Amazon Elastic Load Balancing (ELB) will be distributeing application traffic across multiple targets, such as in our EC2 instances, containers, and IP addresses, across Availability Zones. By balancing traffic and managing surges, ELB ensures high availability and performance, which is vital for GoGreen's regions in Europe, South America, and Southern California. Amazon EC2 Auto Scaling ensures additional instances are launched during peak traffic and scaled down during low demand, delivering real-time resource adjustment for high availability and cost efficiency. Amazon CloudFront further enhances performance by caching content close to users, reducing latency and improving user experience during traffic spikes.
Server capacity should be between 50% and 60%	AWS Elastic Beanstalk is ideal for our client who aims to maintain server utilization between 50% and 60%, thanks to its automated scaling, monitoring, and resource management. It handles backend tasks like provisioning, load balancing, and scaling, reducing manual effort. With Auto Scaling policies, clients can set utilization thresholds to adjust instances automatically, ensuring responsive performance while avoiding overuse and unnecessary costs. Its integration with Application Load Balancers distributes traffic evenly, preventing capacity issues and maintaining performance. Elastic Beanstalk leverages CloudWatch to monitor metrics and apply automatic scaling as needed, offering real-time performance tracking and fine-tuning. This solution ensures us an efficient, cost-effective infrastructure while simplifying capacity management and maintaining reliable application performance.
Overall memory and CPU utilization should not go above 80% and 75% respectively or below 30% for either	We will utilize CloudWatch for monitoring memory and CPU usage. By setting up CloudWatch alarms, we will ensure that memory utilization remains between 30% and 80%, and CPU usage stays within 30% to 75%. These alarms can trigger scaling actions and send notifications if the usage deviates from these specified thresholds.
Internet access is required for patching and updates without exposing the servers.	To meet the requirement for internet access for patching and updates without exposing the servers, we will deploy application instances in private subnets. These private subnets ensure that the servers are not directly accessible from the internet, providing a secure layer of isolation. For necessary outbound internet connectivity, such as downloading patches and updates, we will configure a NAT Gateway within the architecture. The NAT Gateway will allow private instances to initiate outbound connections to the internet for maintenance tasks while preventing any unsolicited inbound traffic from reaching these instances. This ensures that the servers remain secure while still being able to perform essential updates and patches. Additionally, we can enhance security by restricting the outbound traffic from the NAT Gateway using route table rules or by applying strict IAM policies and network ACLs. This setup maintains the integrity of the private environment and ensures compliance with the requirement for secure patching and updates.

Application Tier instances should be tagged as “Key=Name” and “Value=app-tier”.	To ensure Application Tier instances are tagged as Key=Name and Value=app-tier, ours tags will be applied automatically by using Launch Templates with pre-defined tags by configuring a Lambda function triggered by EventBridge to apply tags dynamically. Pros: The combination of EventBridge and Lambda can handle tagging across multiple regions and scales seamlessly with varying instance creation rates. It also requires minimal setup and maintenance compared to broader compliance tools like AWS Config. Which is perfect for multi-regions in which GoGreen has.

In our chart provided, we successfully addressed all key requirements.

To ensure the architecture is flexible and can handle peak performance, we implemented Amazon Elastic Load Balancer (ELB) and EC2 Auto Scaling. This allows the system to dynamically adjust resources based on demand, ensuring consistent performance during traffic surges. Additionally, CloudFront enhances user experience by reducing latency during high-demand periods.

For optimizing server capacity, Elastic Beanstalk and Auto Scaling policies were configured to maintain utilization between 50% and 60%, reducing server strain from the current 90% usage. This ensures efficient resource management without overloading or underutilizing resources. Amazon CloudWatch alarms were set up to monitor memory and CPU usage, keeping them within 30%-80% and 30%-75% thresholds, respectively. These alarms trigger automatic scaling actions and send notifications if thresholds are exceeded, maintaining optimal performance.

To enable secure internet access for patching and updates, application instances were deployed in private subnets with a NAT Gateway. This setup allows outbound internet access for maintenance tasks without exposing the servers to unsolicited inbound traffic, ensuring both functionality and security. Finally, Application Tier instances are automatically tagged as "Key=Name" and "Value=app-tier" using EventBridge and Lambda functions. This approach ensures consistent tagging across GoGreen's multi-regional environment, simplifying resource identification and management.

Through these solutions, we have met all requirements, delivering a scalable, secure, and efficient infrastructure for GoGreen.

Design: Database Tier

Requirement	Solution
Database needs consistent storage performance at 21,000 IOPS.	To adhere to the Well-Architected Framework, we'll be using AWS Aurora (RDS) with Aurora I/O-Optimized for this requirement. This high-performance database is compatible with MySQL and PostgreSQL and is designed to prioritize both performance efficiency and cost optimization. By utilizing Aurora I/O-Optimized and its configurations, we can automatically adjust IOPS to meet our workload demands. This feature eliminates the need for manual provisioning of IOPS, making it an excellent choice for scalability. As a managed service by AWS, Aurora supports Operational Excellence by simplifying database management, backups, and maintenance tasks, which in turn reduces operational complexity. Its design allows it to handle high-demand workloads with ease, making it an ideal solution for applications that require both high IOPS and reliability.
High availability is a requirement.	Configuring the database in a multi availability zone deployment would be the best solution. This would automatically create a secondary instance in another availability zone with synchronous replication. Just in case there is a failure, automatic failover ensures minimal downtime, which meets the high availability requirements for the scenario.
Ability to patch and update must be available	Both AWS RDS and Aurora provide automated maintenance windows where the routine patches and updates can be applied with little disruption. The windows can be scheduled during off-peak hours to ensure that the updates do not interfere with regular operations. RDS ensures patches are only applied when they are critical for performance and security. By using a multi-availability zone deployment, the patches and updates can be applied to the standby instance first. AWS would then promote the updated standby as the primary instance, minimizing any downtime and maintaining database availability throughout the process. For our scenario, it would allow GoGreen to achieve right near zero downtime for maintenances and patches, because failover occurs automatically after the patching of the secondary instance
No change to the database schema can be made at this time	AWS Aurora is an excellent choice for a large company like GoGreen, who are looking to go paperless, particularly given the restriction/requirement of making no changes to the database schema. Due to Aurora being fully managed, this compatibility allows it to provide high performance and availability without requiring any schema modifications, which is ideal when schema changes are restricted. Aurora's support for MySQL and PostgreSQL syntax means that migration from existing databases can be

	seamless, integrating with new digital processes without requiring extensive rework. For a company transitioning to a paperless system, this ease of integration is invaluable, enabling it to modernize without disrupting existing data structures.

Design – Additional AWS Services

Additional Services

1. ***AWS WAF (Web Application Firewall)***

- Security is a must. The purpose of AWS WAF is to protect the applications from common exploits that may affect availability. This service can be configured with out Application Load Balancer, allowing us to filter out unwanted traffic.

2. ***Amazon GuardDuty***

- We could strengthen our security by introducing GuardDuty in our architecture. It is categorized as a TDS that constantly monitors for malicious activity, providing alerts for GoGreen's admin team in case there are threats. Overall, it enhances the security backbone.

3. ***Amazon Aurora***

- Aurora is a MySQL compatible RDS built for high performance and availability. It is a great cost-effective option compated to the traditional MySQL service, while also offering auto-scaling and automated backups across multiple availability zones.

4. ***AWS Systems Manager***

- In an environment with heavy workloads and constant operations, the ability for resource managing is a must. AWS Systems Manager offers a unified interface for different applications. A great tool that can be used within SM is the "Parameter Store" which allows us to manage configuration data, streamlining resource management.

5. ***Amazon S3 Glacier***

- Unlike the regular S3 service, Glacier is ideal for documents and data that need to be retained for more than 5 years. Typically, this is for data that are is rarely seen or accessed, which helps GoGreen lower storage costs.

6. ***AWS Elastic Beanstalk***

- This service is a beginner-friendly environment that allows for the deployment and scaling of web-applications. It can automatically handle capacity provisioning, balancing and perform Auto Scaling within health monitoring. It is a great fit if GoGreen plans to simplify app management.

7. ***Amazon Textract***

- Since GoGreen plans to go paperless, Textract is expected to be a perfect fit within the architecture. It is a machine learning service that extracts text and various data from scanned documents that the company may encounter. Documents are digitized and extracted from customer documents and streamline paperless operations.

8. ***AWS Global Accelerator***

- The goal for this architecture is the deployment within three different regions: NA, SA, and the EU. AWS Global Accelerator is a networking service that improves the availability and performance of the applications used by global users. It is able to route traffic through the global network, improving latency for users accessing from different regions.

9. ***Secrets Manager:***

- Secrets Manager is a robust solution designed to securely store and manage sensitive data, including database credentials and API keys. It automates the rotation of these secrets, minimizing the risk of unauthorized access and ensuring that sensitive information remains protected from exposure.

10. ***Certificate Manager:***

- This service streamlines the management of SSL/TLS certificates, which are essential for establishing secure and encrypted connections between various services. By simplifying the deployment process, Certificate Manager enhances the security of communication, particularly for applications that operate globally across multiple regions.

11. ***VPC Peering:***

- VPC Peering allows for secure and private communication between Virtual Private Clouds (VPCs) in distinct regions or accounts. This connectivity solution enhances the overall architecture of multi-region applications, enabling seamless data exchange and interaction between different environments while maintaining robust security.

12. ***S3 Cross-Region Replication:***

- S3 Cross-Region Replication is a feature that ensures critical data is duplicated across different geographic locations. This strategy not only enhances data availability but also provides a solid disaster recovery solution, aligning perfectly with GoGreen's commitment to a resilient multi-region architecture.

13. ***EventBridge:***

- EventBridge serves as an event-driven automation platform that facilitates the integration of various services by triggering specific actions based on defined events. This capability is invaluable for streamlining operational workflows and enabling applications to scale effectively in response to demand.

14. ***Aurora (RDS):***

- Aurora, part of Amazon's Relational Database Service (RDS), provides a high-performance, scalable database solution that is ideal for managing data across multiple regions. With features like automated backups and high availability, Aurora ensures that businesses can efficiently handle their data needs while maintaining reliability.

15. ***IAM (Identity and Access Management):***

- IAM is a critical service that oversees access control and permissions for users and services within an organization. It allows for the configuration of fine-grained access policies, ensuring that individuals and applications only have the permissions necessary for their roles, thereby enhancing overall security.

16. ***MFA (Multi-Factor Authentication):***

- Multi-Factor Authentication (MFA) introduces a vital layer of security to user accounts by requiring multiple forms of verification before granting access. This additional safeguard helps to protect the GoGreen environment from unauthorized access, ensuring that only legitimate users can enter sensitive areas.

17. ***CloudTrail:***
- CloudTrail is an invaluable tool for monitoring and auditing API calls and activities within your AWS environment. By providing detailed logs of actions taken by users and services, it aids in compliance efforts and helps organizations quickly identify and respond to suspicious activities.

18. ***Elastic Beanstalk:***
- Elastic Beanstalk simplifies the process of deploying applications by abstracting the underlying infrastructure management. Developers can easily deploy and scale their applications, allowing for rapid iteration and innovation without the overhead of managing the infrastructure directly.

19. ***Application Load Balancer:***
- The Application Load Balancer intelligently distributes incoming traffic across multiple servers, ensuring that no single server becomes overwhelmed. This balanced traffic flow enhances fault tolerance, improves application performance, and allows applications operating in various regions to maintain consistent user experiences.

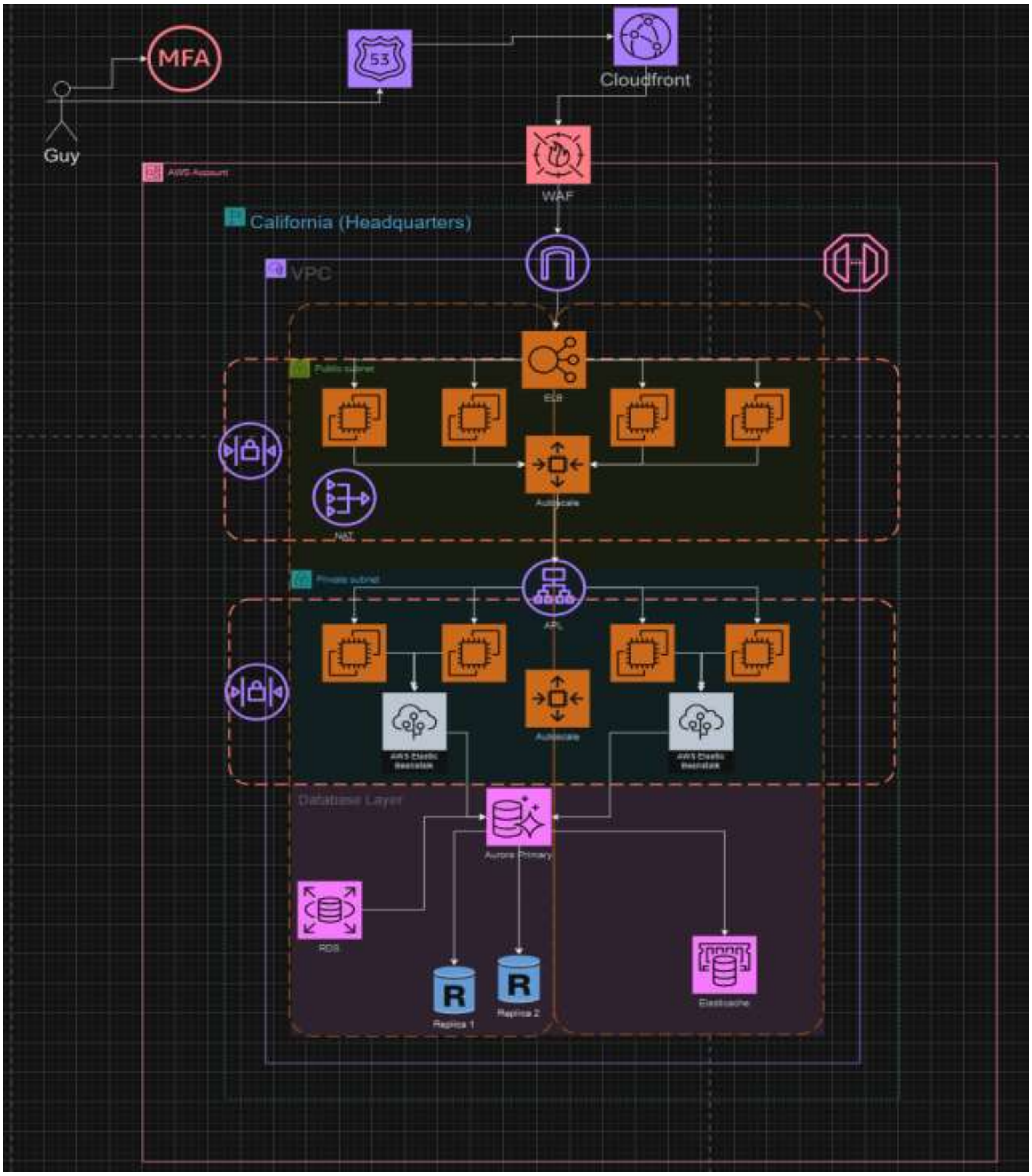
20. ***NAT Gateway:***
- A NAT Gateway facilitates secure outbound internet access for resources residing in private subnets. This enables these resources to perform necessary functions, such as downloading updates or patches, while keeping them hidden from the public internet to safeguard against security risks.

21. ***Network ACL (Access Control List):***
- Network ACLs provide an additional security layer by managing traffic flow at the subnet level. By setting rules to allow or deny specific traffic, they complement security groups to enhance network security, offering more granular control over access management.

22. ***Lambda:***
- AWS Lambda empowers developers to execute code in response to events without provisioning or managing servers. This event-driven architecture allows for seamless automation of processes such as resource tagging, data processing, and alerting, significantly boosting the scalability and efficiency of applications.

23. ***ElastiCache:***
- ElastiCashe will enhances GoGreen by caching frequent queries, managing sessions, and supporting real-time analytics, leaderboards, and personalized recommendations. It reduces backend load, improves response times, and enables scalability, ensuring a faster, more efficient user experience.

Proposed VPC Architecture Diagram

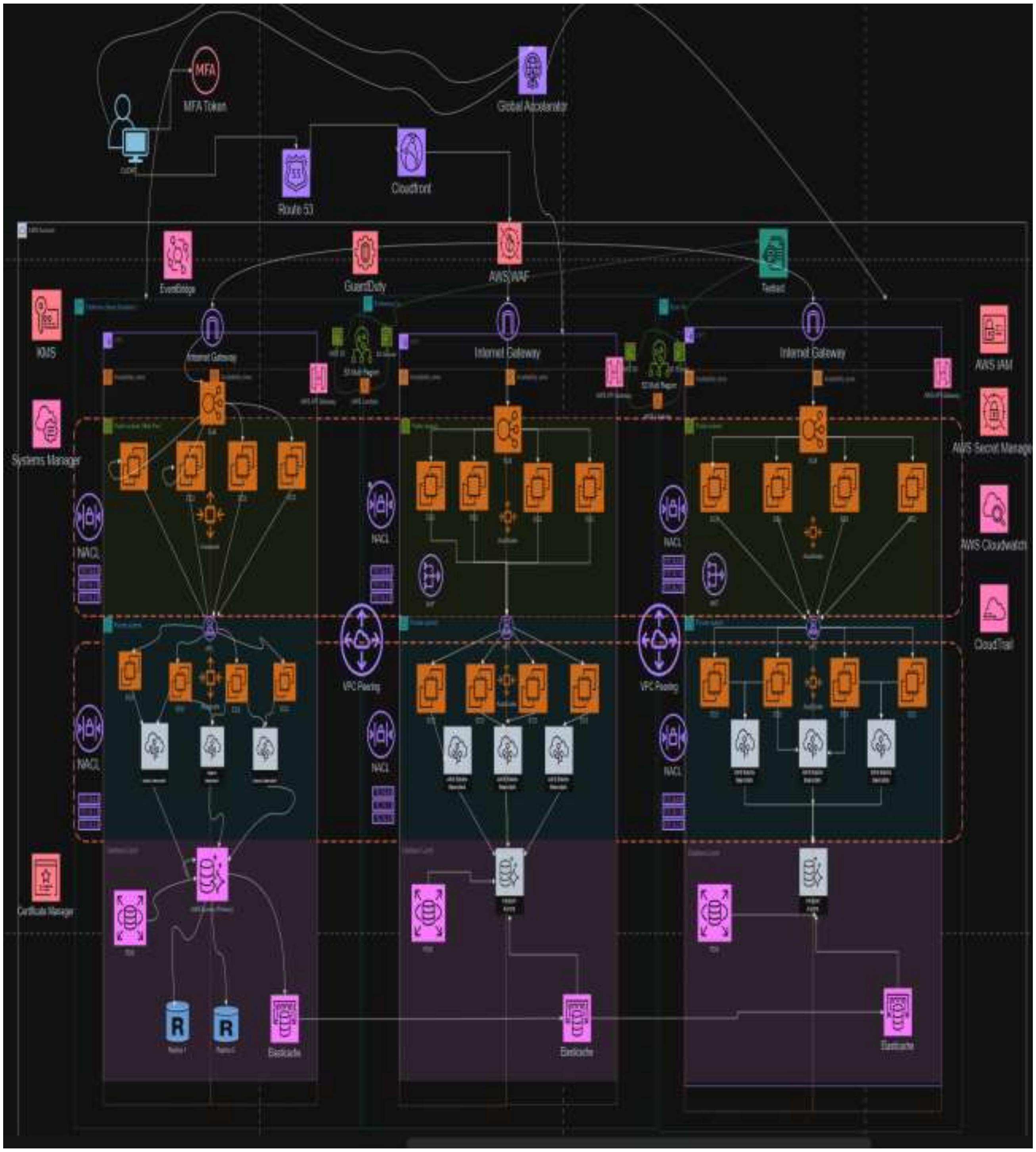


Design – Network Document the VPC solution

VPC	Region	Purpose	Subnets	AZs	CIDR Range
1	**Europe (London)**	**Web Application Hosting**	**Public & Private**	**Eu-west-2b**	**10.0.2.0/16**
2	**US East (N.California)**	**Web Application Hosting**	**Public & Private**	**Us-east-1a**	**10.0.1.0/24**
3	**South America (Sao Paulo)**	**Web Application Hosting**	**Public & Private**	**Sa-east-1**	**10.0.3.0/24**

Subnet Name	VPC	Subnet Type (Public/Private)	AZ	Subnet Addresses
Public-Subnet-1	**1**	**Public**	**Us-east-1a**	**10.0.1.0/24**
Public-Subnet-2	**2**	**Public**	**Eu-west-1a**	**10.0.3.0/24**
Public-Subnet-3	**3**	**Public**	**Sa-east-1**	**10.0.5.0/24**
Private-Subnet-1	**1**	**Private**	**Eu-west-2**	**10.0.4.0/24**
Private-Subnet-2	**2**	**Private**	**US-east-1a**	**10.0.2.0/24**
Private-Subnet-3	**3**	**Private**	**Sa-east-1**	**10.0.6.0/24**

Proposed Architecture Diagram



Design – Security Chart

Security Groups (SG)	SG Name	Rule
ELB Load Balancer	ELB-SG	Inbound: Allow HTTP (Port 80) from 0.0.0.0/.0 to enable public web access. Allow HTTPS (Port 443) from 0.0.0.0/0 to enable secure web access. Outbound: Allow the traffic to the Web-Tier Security Group (SG-Web-Tier) on necessary ports for application data flow.
Web Tier	SG-Web-Tier	Inbound: Allow HTTP (Port 80) from ELB-SG to allow traffic from the Load Balancer Allow HTTPS (Port 443) from ELB-SG to only allow secure traffic from the Load Balancer. Outbound: Allow traffic to the Application Tier Security Group (SG-App-Tier) on the application-specific ports. Allow traffic to the internet for any necessary software updates.
App Tier	SG-App-Tier	Inbound: Allow traffic from SG-Web-Tier on application-specific ports Outbound: Allow traffic to the Database Tier Security Group (SG-DB-Tier) on the database port (ex: port 3306 for MySQL)
Database Tier	SG-DB-Tier	Allow traffic on the database port (ex: port 3306 for MySQL) from SG-App-Tier only and restricts the database access to the application tier.

		Outbound: Minimal outbound access and restricted only to backup services or administrative requirements.

Other Security Options	Solution Justifications
AWS WAF	AWS WAF is essential for our GoGreen project to protect our web applications from common vulnerabilities such as SQL injection, cross-site scripting , and other exploits targeting application security. By integrating WAF with our Application Load Balancer or Amazon CloudFront distributions, we can filter and block malicious requests before they reach our application tier, ensuring a secure user experience. Its customizable rule sets allow us to define policies tailored to the specific traffic patterns in our regions, including Europe, South America, and Southern California. With real-time visibility into web traffic and actionable insights, WAF empowers us to respond quickly to threats while maintaining application availability and performance.
AWS Guard Duty	To safeguard our applications from Distributed Denial of Service (DDoS) attacks, AWS Shield will provide us protection. By leveraging Shield, we'll be ensuring that our services remain available and performant even during large-scale attacks, a critical requirement for our multi-region architecture that serves diverse users across the globe.
AWS Shield	GuardDuty enhances our project's overall security by continuously monitoring and analyzing activity across our AWS accounts for potential threats. Using machine learning and integrated threat intelligence, it detects anomalies such as unauthorized API calls, unusual network traffic, or potential account compromise. With GuardDuty, we gain actionable insights to identify and mitigate threats before they escalate. This service is particularly beneficial for our GoGreen project, as it operates seamlessly across regions, allowing us to maintain consistent threat detection and response capabilities for our infrastructure in Europe, South America, and Southern California. By leveraging GuardDuty, we can ensure the security of our resources while focusing on scaling and optimizing our cloud environment.

Design – Encryption

Requirement	Solution
Encryption option for data at rest	We'll be using AWS Key Management Service (KMS) for managing encryption keys across resources like Amazon S3 and RDS, applying AES-256 encryption. And for Amazon S3 specifically, we'll be using Server-Side Encryption (SSE) with AWS-managed keys (SSE-S3).
Encryption option for data in transit	We'll be adopting TLS (Transport Layer Security) to enhance the security of our data transmission. In addition, implementing HTTPS alongside TLS certificates is essential for our web applications, and we can efficiently manage these certificates through AWS Certificate Manager. Furthermore, our AWS services, such as RDS and S3, support TLS, ensuring that our communications are secure. For internal data transfers within AWS, we will leverage VPC Peering to maintain the privacy of our data and keep it within the AWS network. This approach will significantly strengthen our overall security posture.

When it comes to our forms of encryption, we were able to addresses these 2 requirements listed above.

For starters , we're implementing encryption for both data at rest and in transit. For data at rest, AWS Key Management Service (KMS) will manage encryption keys across resources like Amazon S3 and RDS, using AES-256 encryption (Advanced Encryption Standard). S3 will specifically utilize Server-Side Encryption with AWS-managed keys (SSE-S3).

For data in transit, we're adopting TLS for secure transmission and enabling HTTPS with TLS certificates managed through AWS Certificate Manager. AWS services like RDS and S3 will also utilize TLS for secure communication. Additionally, VPC Peering will ensure private and secure internal data transfers within AWS, reinforcing our security posture.

Design – Instance Details

Tier	AMI	Tag	Type	Size	Justification	# of instance
Web	**T2**	**MyWebServer**	**T2.micro**	**small**	**Good for handing incoming traffic with Auto Scaling & Load Balancing for peak performance**	**4**
App	**T2**	**MyApp**	**T2.micro**	**small**	**Manages business logic and data processing with scalability through Elastic Beanstalk & Auto Scaling**	**4**
DB	**n/a**	**N/a**	**Db.t3.medium**	**medium**	**Amazon RDS for consistent high IOPS, high availability with multi-AZ deployment and automatic failover**	**1**

The chart above provides a simple breakdown of the configurations needed for each tier within the architecture. We can see the different rows named as “Web, Application, and DB”. Each row is also detailed by their characteristics such as AMI, Tag, Type, Size, Justification and number of instances.

This will outline GoGreen’s balanced and scalable architecture, with each instance configured specifically for the company’s needs. This ensures that we meet scalability, high availability and cost-efficiency. It provides us with the robust environment suitable for deployment and low failovers.

Design: Recovery Point Objective

Q. How would you achieve a Recovery Point Objective (RPO) of four hours?

To achieve a four-hour Recovery Point Objective (RPO), we'll implement a comprehensive strategy that combines regular backups, data replication, and proactive monitoring. By scheduling automated backups to run every four hours or less, we ensure consistent data protection and minimize the risk of data loss in case of unexpected issues. We also use continuous or near-real-time data replication to a secondary location or disaster recovery site, such as AWS S3 Cross-Region Replication or database replication, to keep our data synchronized and compliant with the four-hour RPO.

Incremental backups are another key component, as they'll capture only the changes made since the last backup of ours, making the process more efficient while helping us meet our RPO target with ease. Additionally, a robust monitoring and alert system is essential to verify that backups and replication processes are completed within the required timeframe, allowing us to promptly address any potential issues.

By integrating these strategies, we confidently meet our four-hour RPO, providing our team with reliable data protection and peace of mind.

Design: Document Storage Chart

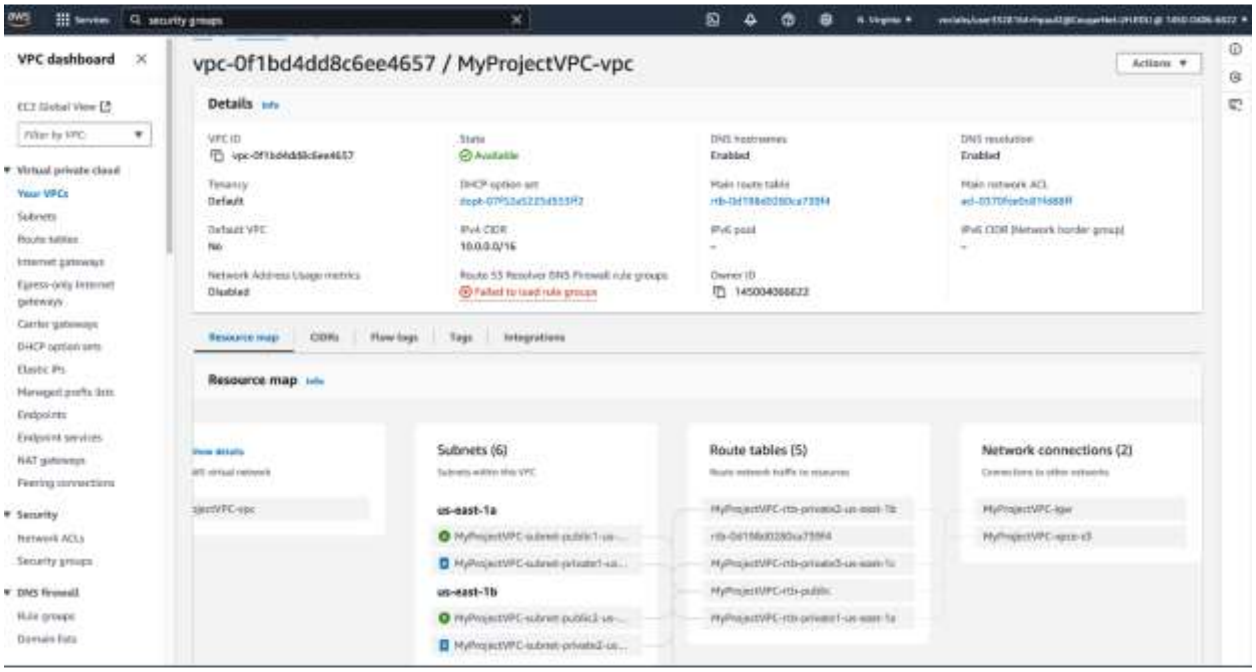
Storage/Archive Option	Details
Amazon S3	The S3 service enables storing active static documents and images that are constantly accessed. Even though it is a budget-friendly solution, it provides high durability and scalability for GoGreen’s storage. More importantly, it integrates with multiple services for security purposes, meeting the company’s data needs.
S3 Glacier	Offers the same functionality as the standard S3 service, however it is typically for documents and images that need to be retained within five years. This data is not accessed as much but offers an affordable solution for archiving purposes. We can also do automation transitions from the standard service to glacier classes.

These two AWS storage options provides GoGreen with document storage solutions that meet the company’s needs. Amazon S3 is typically used for static documents such as files and pictures that need to be frequently accessed.

S3 Glacier has the same purpose, but GoGreen may opt to store static data that are *not frequently* accessed as much.

Both of these are services that are budget friendly, while still offering scalability and storage power.

Proof of Concept Implementation Implement a proof of concept in the Learner Labs environment. Your design should guide implementation of the following



vpc-0f1bd4dd8c6ee4657 / MyProjectVPC-vpc

Details

VPC ID vpc-0f1bd4dd8c6ee4657	State Available	DNS hostnames Enabled	DNS resolution Enabled
Tenancy Default	DHCP option set dopt-07f52a5225d553ff2	Main route table rtb-0d198e0280ca739f4	Main network ACL acl-0370fce0c8f4d88ff
Default VPC No	IPv4 CIDR 10.0.0.0/16	IPv6 pool -	IPv6 CIDR (Network border group) -
Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups Failed to load rule groups	Owner ID 145004066622	

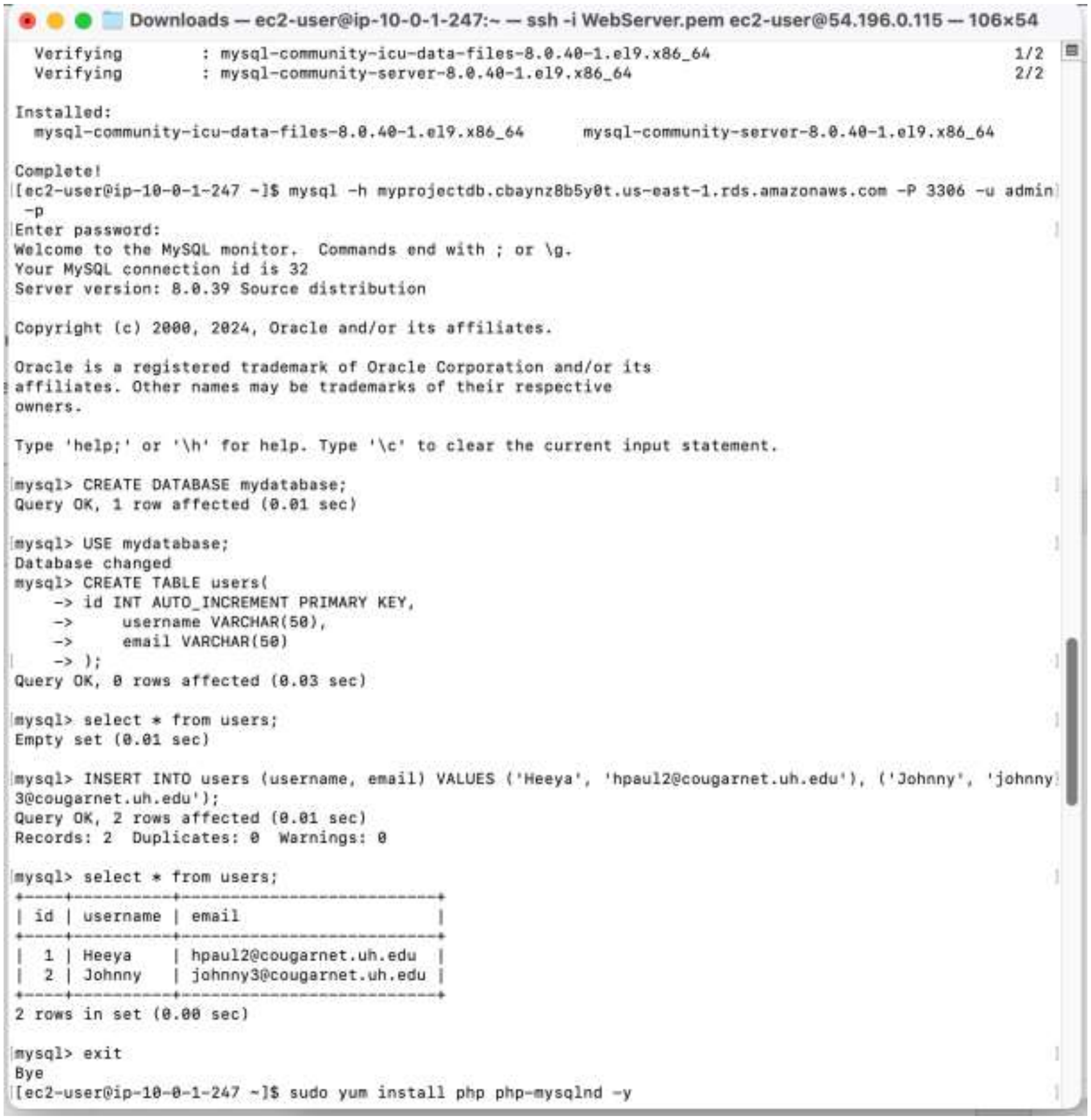
Resource map

Show details
AWS virtual network
MyProjectVPC-vpc

Subnets (6)
Subnets within this VPC
us-east-1a
MyProjectVPC-subnet-public1-us-...
MyProjectVPC-subnet-private1-us...
us-east-1b
MyProjectVPC-subnet-public2-us-...
MyProjectVPC-subnet-private2-us...

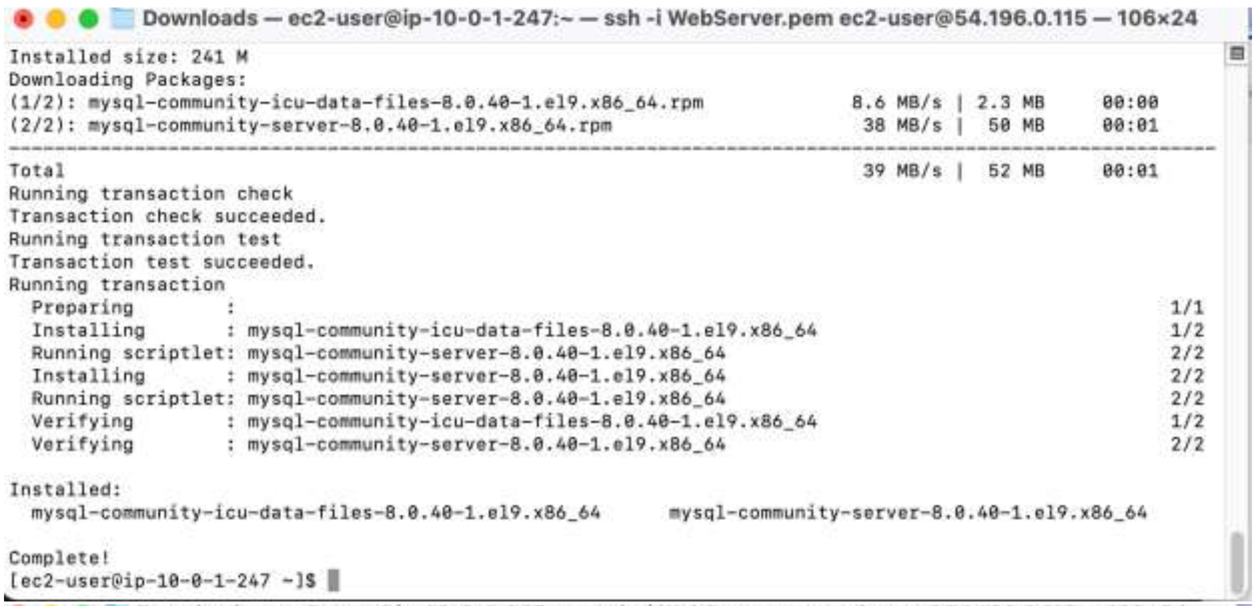
Route tables (5)
Route network traffic to resources
MyProjectVPC-rtb-private2-us-east-1b
rtb-0d198d0280ca739f4
MyProjectVPC-rtb-private3-us-east-1c
MyProjectVPC-rtb-public
MyProjectVPC-rtb-private1-us-east-1a

Network connections (2)
Connections to other networks
MyProjectVPC-igw
MyProjectVPC-vpce-s3



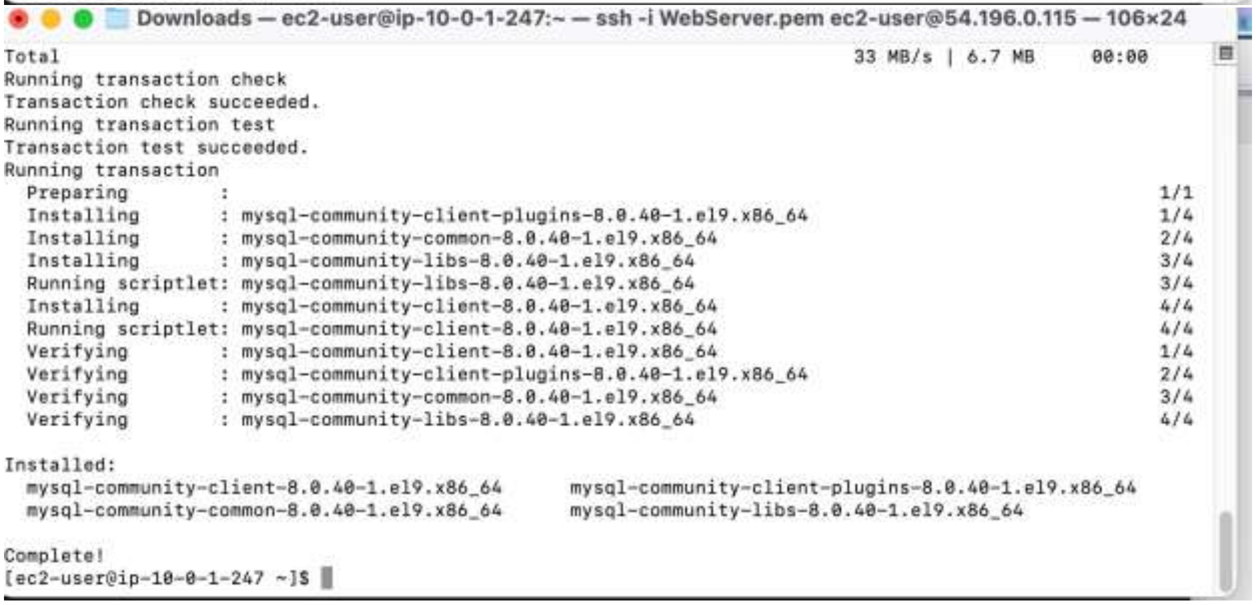
Downloads — ec2-user@ip-10-0-1-247:~ — ssh -i WebServer.pem ec2-user@54.196.0.115 — 106×54

```
  Verifying        : mysql-community-icu-data-files-8.0.40-1.el9.x86_64                                 1/2
  Verifying        : mysql-community-server-8.0.40-1.el9.x86_64                                         2/2

Installed:
  mysql-community-icu-data-files-8.0.40-1.el9.x86_64      mysql-community-server-8.0.40-1.el9.x86_64

Complete!
[ec2-user@ip-10-0-1-247 ~]$ mysql -h myprojectdb.cbaynz8b5y0t.us-east-1.rds.amazonaws.com -P 3306 -u admin
 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 8.0.39 Source distribution

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE mydatabase;
Query OK, 1 row affected (0.01 sec)

mysql> USE mydatabase;
Database changed
mysql> CREATE TABLE users(
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    ->     username VARCHAR(50),
    ->     email VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> select * from users;
Empty set (0.01 sec)

mysql> INSERT INTO users (username, email) VALUES ('Heeya', 'hpaul2@cougarnet.uh.edu'), ('Johnny', 'johnny
3@cougarnet.uh.edu');
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from users;
+----+----------+-------------------------+
| id | username | email                   |
+----+----------+-------------------------+
|  1 | Heeya    | hpaul2@cougarnet.uh.edu  |
|  2 | Johnny   | johnny3@cougarnet.uh.edu |
+----+----------+-------------------------+
2 rows in set (0.00 sec)

mysql> exit
Bye
[ec2-user@ip-10-0-1-247 ~]$ sudo yum install php php-mysqlnd -y
```



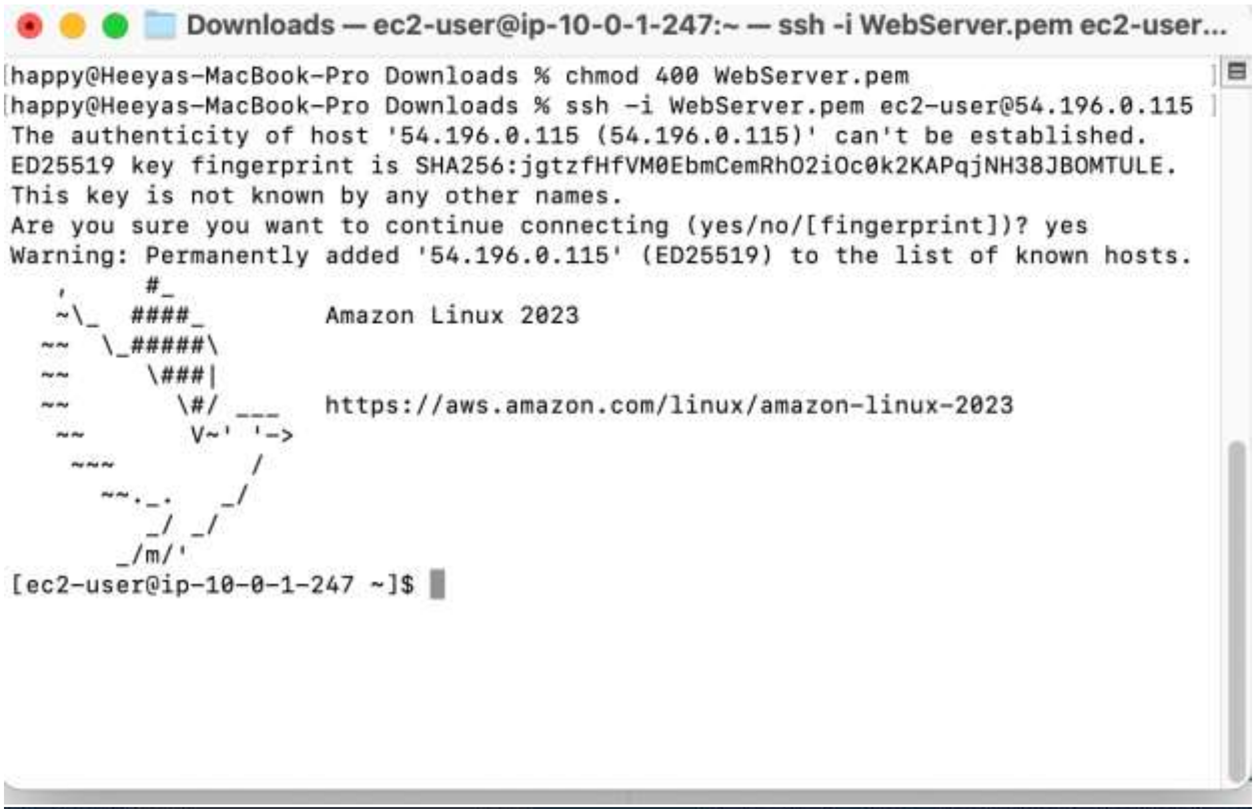
Downloads — ec2-user@ip-10-0-1-247:~ — ssh -i WebServer.pem ec2-user@54.196.0.115 — 106×24

```
Installed size: 241 M
Downloading Packages:
(1/2): mysql-community-icu-data-files-8.0.40-1.el9.x86_64.rpm          8.6 MB/s | 2.3 MB     00:00
(2/2): mysql-community-server-8.0.40-1.el9.x86_64.rpm                   38 MB/s |  50 MB     00:01
----------------------------------------------------------------------------------------------------------
Total                                                                   39 MB/s |  52 MB     00:01
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                              1/1
  Installing       : mysql-community-icu-data-files-8.0.40-1.el9.x86_64                          1/2
  Running scriptlet: mysql-community-server-8.0.40-1.el9.x86_64                                  2/2
  Installing       : mysql-community-server-8.0.40-1.el9.x86_64                                  2/2
  Running scriptlet: mysql-community-server-8.0.40-1.el9.x86_64                                  2/2
  Verifying        : mysql-community-icu-data-files-8.0.40-1.el9.x86_64                          1/2
  Verifying        : mysql-community-server-8.0.40-1.el9.x86_64                                  2/2

Installed:
  mysql-community-icu-data-files-8.0.40-1.el9.x86_64      mysql-community-server-8.0.40-1.el9.x86_64

Complete!
[ec2-user@ip-10-0-1-247 ~]$
```



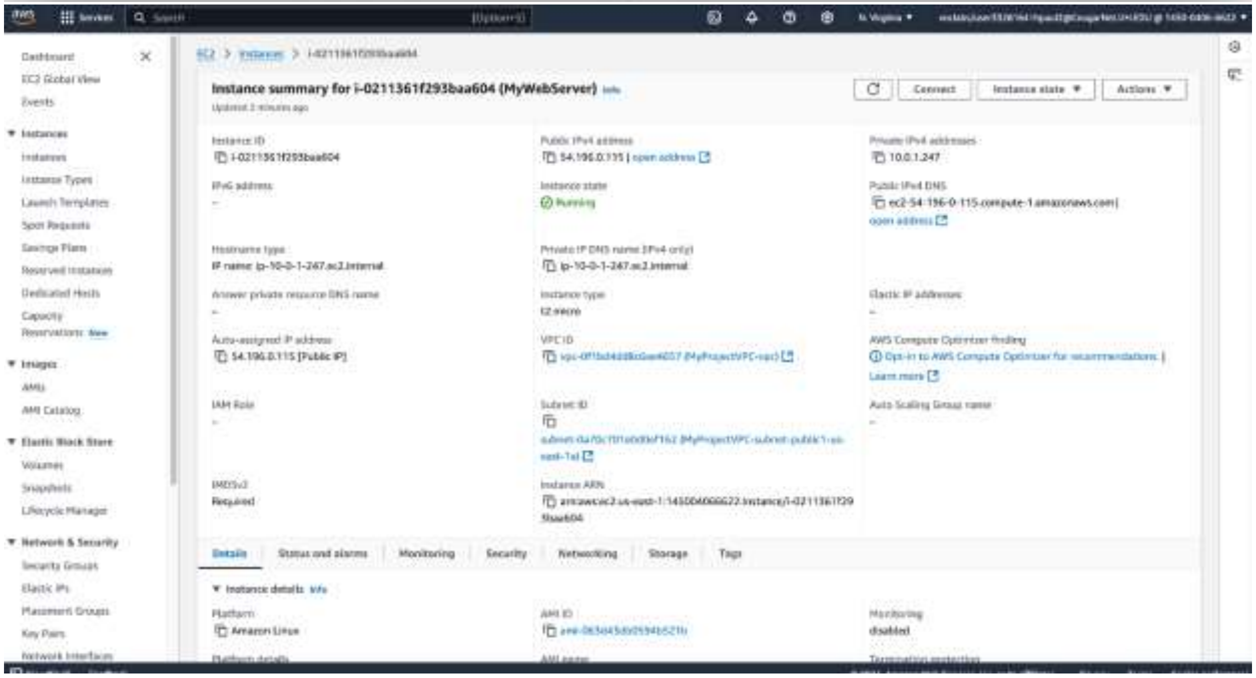
Downloads — ec2-user@ip-10-0-1-247:~ — ssh -i WebServer.pem ec2-user@54.196.0.115 — 106×24

```
Total                                                                   33 MB/s | 6.7 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                              1/1
  Installing       : mysql-community-client-plugins-8.0.40-1.el9.x86_64                          1/4
  Installing       : mysql-community-common-8.0.40-1.el9.x86_64                                  2/4
  Installing       : mysql-community-libs-8.0.40-1.el9.x86_64                                    3/4
  Running scriptlet: mysql-community-libs-8.0.40-1.el9.x86_64                                    3/4
  Installing       : mysql-community-client-8.0.40-1.el9.x86_64                                  4/4
  Running scriptlet: mysql-community-client-8.0.40-1.el9.x86_64                                  4/4
  Verifying        : mysql-community-client-8.0.40-1.el9.x86_64                                  1/4
  Verifying        : mysql-community-client-plugins-8.0.40-1.el9.x86_64                          2/4
  Verifying        : mysql-community-common-8.0.40-1.el9.x86_64                                  3/4
  Verifying        : mysql-community-libs-8.0.40-1.el9.x86_64                                    4/4

Installed:
  mysql-community-client-8.0.40-1.el9.x86_64      mysql-community-client-plugins-8.0.40-1.el9.x86_64
  mysql-community-common-8.0.40-1.el9.x86_64      mysql-community-libs-8.0.40-1.el9.x86_64

Complete!
[ec2-user@ip-10-0-1-247 ~]$
```



Downloads — ec2-user@ip-10-0-1-247:~ — ssh -i WebServer.pem ec2-user...

```
happy@Heeyas-MacBook-Pro Downloads % chmod 400 WebServer.pem
happy@Heeyas-MacBook-Pro Downloads % ssh -i WebServer.pem ec2-user@54.196.0.115
The authenticity of host '54.196.0.115 (54.196.0.115)' can't be established.
ED25519 key fingerprint is SHA256:jgtzfHfVM0EbmCemRhO2iOc0k2KAPqjNH38JBOMTULE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.196.0.115' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-1-247 ~]$ 
```



EC2 > Instances > i-0211361f293baa604

Instance summary for i-0211361f293baa604 (MyWebServer)

Updated 2 minutes ago

Instance ID: i-0211361f293baa604

Public IPv4 address: 54.196.0.115 | open address

Private IPv4 addresses: 10.0.1.247

IPv6 address: –

Instance state: Running

Public IPv4 DNS: ec2-54-196-0-115.compute-1.amazonaws.com | open address

Hostname type: IP name: ip-10-0-1-247.ec2.internal

Private IP DNS name (IPv4 only): ip-10-0-1-247.ec2.internal

Answer private resource DNS name: –

Instance type: t2.micro

Elastic IP addresses: –

Auto-assigned IP address: 54.196.0.115 [Public IP]

VPC ID: vpc-[illegible] (MyProjectVPC-vpc)

AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. | Learn more

IAM Role: –

Subnet ID: subnet-[illegible] (MyProjectVPC-subnet-public1-us-east-1a)

Auto Scaling Group name: –

IMDSv2: Required

Instance ARN: arn:aws:ec2:us-east-1:[illegible]:instance/i-0211361f293baa604

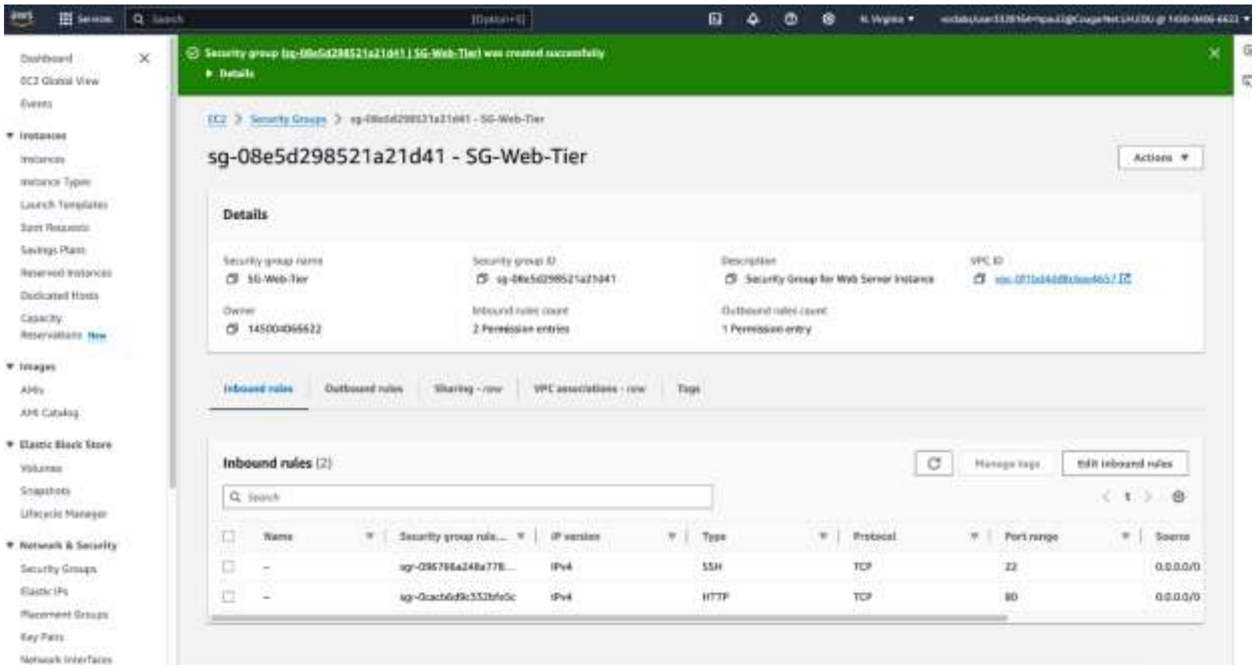
Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

▼ Instance details

Platform: Amazon Linux

AMI ID: ami-[illegible]

Monitoring: disabled



Security group sg-08e5d298521a21d41 | SG-Web-Tier was created successfully

EC2 > Security Groups > sg-08e5d298521a21d41 - SG-Web-Tier

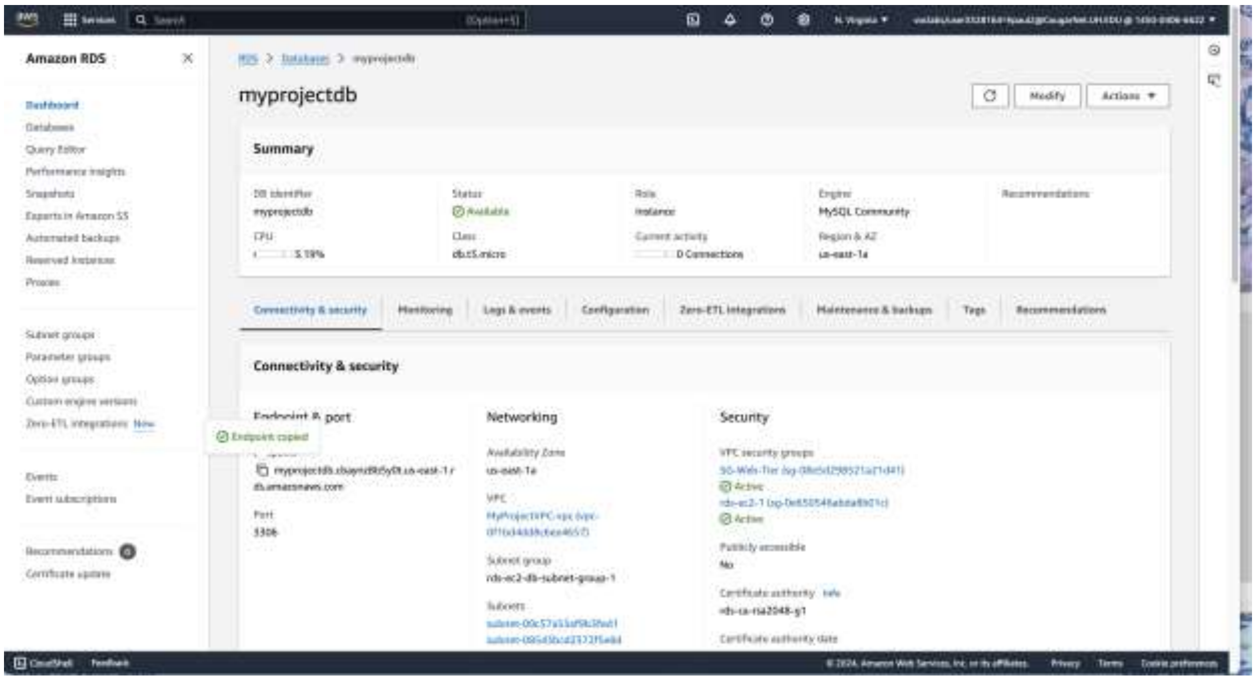
sg-08e5d298521a21d41 - SG-Web-Tier

Details

Security group name	Security group ID	Description	VPC ID
SG-Web-Tier	sg-08e5d298521a21d41	Security Group for Web Server Instance	[illegible]
Owner	Inbound rules count	Outbound rules count	
145004066622	2 Permission entries	1 Permission entry	

Inbound rules (2)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source
–	sgr-096766a248a778...	IPv4	SSH	TCP	22	0.0.0.0/0
–	sgr-0cacb6d9c332bfe5c	IPv4	HTTP	TCP	80	0.0.0.0/0



RDS > Databases > myprojectdb

myprojectdb

Summary

DB identifier	Status	Role	Engine	Recommendations
myprojectdb	Available	Instance	MySQL Community	
CPU	Class	Current activity	Region & AZ	
5.19%	db.t3.micro	0 Connections	us-east-1a	

Connectivity & security

Endpoint & port

Endpoint copied

myprojectdb.[illegible].us-east-1.rds.amazonaws.com

Port

3306

Networking

Availability Zone

us-east-1a

VPC

MyProjectVPC-vpc (vpc-[illegible])

Subnet group

rds-ec2-db-subnet-group-1

Subnets

subnet-[illegible]

subnet-[illegible]

Security

VPC security groups

SG-Web-Tier (sg-08e5d298521a21d41)

Active

rds-ec2-1 (sg-[illegible])

Active

Publicly accessible

No

Certificate authority

rds-ca-rsa2048-g1

Certificate authority date

© 2024, Amazon Web Services, Inc. or its affiliates.