Background:

You are recently promoted from a Cloud Engineer to a Cloud Architect and assigned a project to prepare a new environment in the cloud, to which your team will later migrate their applications.

Goal:

To architect a solution that is secure, scalable, highly available, and cost-effective using AWS.

Requirements:

* They are concerned about the security of the environment, so they have decided to virtually isolate their network from the rest of the customers and the rest of the environments in the same AWS Cloud Account
* Due to the budget issue, the company cannot afford a dedicated DB engineer, so they are willing to outsource the DB management from a Cloud provider, to store and maintain the customer information received by PHP application. You must pick the right solution from AWS, which should be a Platform as a Service. It should also provide high availability, patching, and back-ups. (hint: Create DB subnet group).
* And about disaster recovery, you should have enough backups for both the Web and Database server, so if in case the environment crashes, we can launch a new environment from the disaster recovery backups.
* Design a dynamic website where the customers can enter their details, which should be stored in a database.
* They are uncertain about the traffic pattern that how low or high it can be, so they have a requirement that the environment should be running at least two EC2 servers all time, and when there is a high load, they must burst up to four servers in total.
* Now the company cannot afford a dedicated engineer for monitoring, so you must automate the incident management through an event notification. Anytime there is an increase and decrease in the VM's due to high or low traffic, you must receive a notification via email.
* The application should be highly available, even if a VM fails to respond to queries, there should be a mechanism to shift the connection to another healthy VM automatically.
* Your Dynamic Website should also be cached globally, so users worldwide can access it with less latency. The customer is okay if we get an unfriendly AWS generated URL for accessing the website.

To meet the requirements of a secure, scalable, highly available, and cost-effective solution using AWS, you can follow the architecture outlined below:

1. **Secure Network Isolation:**
   * Create a Virtual Private Cloud (VPC) to isolate the network from other customers and environments within the same AWS Cloud Account.
   * Configure subnets within the VPC: public subnets for the web servers and private subnets for the database.
   * Use security groups and Network Access Control Lists (NACLs) to control inbound and outbound traffic between subnets and the internet.
2. **Managed Database Solution:**
   * Choose Amazon RDS (Relational Database Service) as a managed database solution for storing and maintaining customer information received by the PHP application.
   * Create a DB subnet group to place the database instances in the private subnets for enhanced security.
   * Configure Multi-AZ deployment for high availability, automated backups, and automated patching.
3. **Backup and Disaster Recovery:**
   * Set up regular automated backups for both the web servers and the database using AWS services like Amazon RDS automated backups and Amazon EBS snapshots.
   * Store the backups in a separate location and verify the restore process periodically to ensure the recoverability of the environment.
4. **Dynamic Website with PHP and Database Integration:**
   * Set up EC2 instances with PHP and web server software (e.g., Apache) to host the dynamic website.
   * Establish a connection from the PHP application to the managed database (Amazon RDS) to store and retrieve customer information.
5. **Auto Scaling for High Traffic:**
   * Create an Auto Scaling Group (ASG) with a minimum of two EC2 instances to ensure high availability and load distribution.
   * Configure scaling policies based on CPU utilization or other metrics to automatically adjust the number of instances.
   * Define a maximum limit of four instances to handle high traffic loads.
6. **Automated Incident Management:**
   * Utilize Amazon CloudWatch Events and AWS Simple Notification Service (SNS) to automate incident management.
   * Set up CloudWatch alarms to trigger events based on EC2 instance count changes, and configure SNS to send email notifications when these events occur.
7. **Load Balancing and High Availability:**
   * Use an Elastic Load Balancer (ELB) or Application Load Balancer (ALB) to distribute incoming traffic across the EC2 instances.
   * Configure health checks to monitor the health of the instances. If an instance becomes unhealthy, the load balancer will automatically route traffic to healthy instances.
8. **Global Caching with AWS CloudFront:**
   * Utilize Amazon CloudFront, a global Content Delivery Network (CDN), to cache and serve the dynamic website content with reduced latency worldwide.
   * Configure CloudFront to distribute and cache static and dynamic content, enhancing the performance for global users.
   * Obtain an AWS-generated URL for accessing the website through CloudFront.

By implementing this architecture, you can achieve a secure, scalable, highly available, and cost-effective solution using AWS services.