Lomba Kompetensi Siswa Sekolah Menengah Kejuruan Tingkat Provinsi Jawa Barat Tahun 2023

Modul 3 - Scalable Cloud Architecture

May 10, 2023

Bidang Lomba Cloud Computing

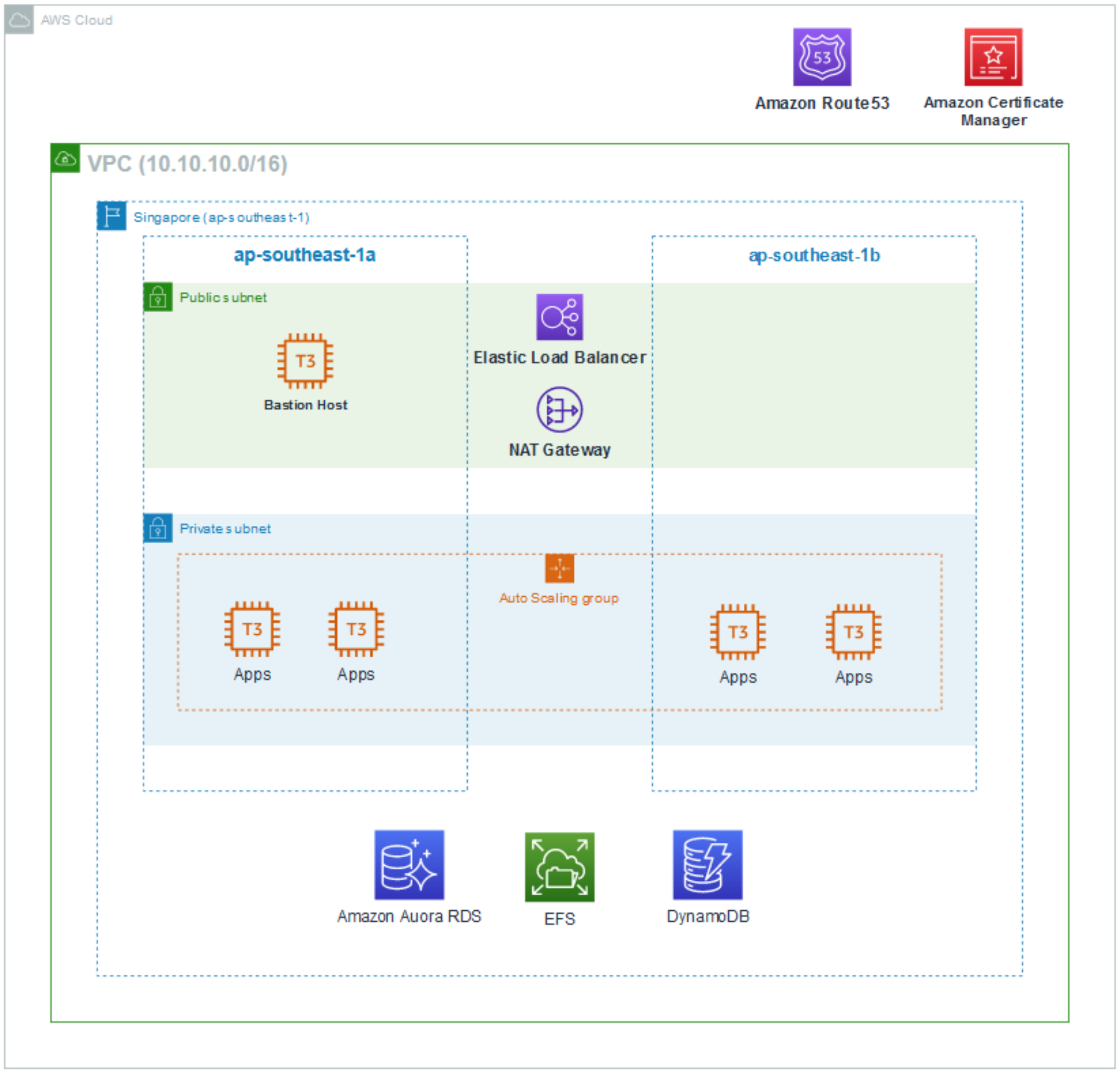
1. Overview

This project aims to deploy and scale a News web application with Laravel as our technology stack. As we know, many requests come to our website and the traffic is getting really high! At first, everything is normal and our infrastructure has zero problems with this. Until we realized our News web application become popular and we couldn't handle the traffic requests from a ton of clients. Our resources are high either CPU or memory, thus the server we used is stuck. So, we need a scalable cloud infrastructure that will handle these issues and not worry about incoming requests or traffic any more. Amazon Web Services come to the rescue. You must build and deploy our web application into AWS infrastructure. Before doing tasks in this project, please read carefully the technical details below.

2. General Rules

1. Failure to comply with the rules will result in immediate disqualification.
2. You have 3 hours to finish the tasks.
3. Please share your AWS account on this link https://lksjabar2023.awan.id. For the EC2 Keypair file, you can upload it later after finish your tasks.
4. You may use AWS Console and AWS CLI to deploy the solutions. You may not use SAM, CloudFormation, or CDK.
5. Between and after the event, you may not access your account. Any activity on AWS during this period is not allowed.
6. During the event, multiple logins are not permitted.
7. If you have any questions, do not hesitate to ask.

3. Architecture



1. The application will be deployed to the AWS cloud infrastructure with VPC as the network stack.
2. You need to set up NAT Gateway for accessing the internet connection from a private instance (in a private subnet).
3. Amazon Route53 and Amazon Certificate Manager as hosted zone layer. You can set up your domain name here.
4. Amazon EC2 for deploying web server for Laravel application and the source code will be stored on Amazon EFS.
5. Amazon Elastic Load Balancer automatically distributes network traffic to improve application scalability.
6. Amazon Aurora RDS for storing news data.

7. And DynamoDB for storing cache (login session and query cache).
8. Amazon EC2 Auto Scaling instance that will help you maintain application availability and lets you automatically add or remove EC2 instances using scaling policies that you define. Dynamic or predictive scaling policies let you add or remove EC2 instance capacity to service established or real-time demand patterns.
9. As mentioned before in the overview section, the application is written in PHP with Laravel web framework. So you need to install the required dependencies that will be explained in the task section.
10. On our news application, users can access and read the news.
11. Also, users can sign in or sign up for creating an account and then write, update, or delete their news.

4. Information

1. The application is written with PHP using the Laravel web framework.
2. In addition, you need to install NodeJS and NPM for building Laravel assets.
3. This solution must be deployed in the **ap-southeast-1** **(Singapore)** region. Deploying in another region will result in a major point reduction.

5. Task

1. First thing first, you must create an Access Key on your account IAM security credentials. Download and save the Access Key ID and Secret Access Key in the safest place. The Access Key ID and Secret Access Key will be used by our News application (Laravel).
2. Create a VPC on your AWS account:
   - Set IPv4 CIDR: 10.0.0.0/16
   - Add a tag to the VPC: KEY=LKS-ID, VALUE=MODUL3-VPC

1. Create public and private subnets with the following requirement:
   - Public subnet on availability zone A with network 10.0.0.0/24, add a tag KEY=LKS-ID, VALUE=MODUL3-PUBLIC-SUBNET-A
   - Public subnet on availability zone B with network 10.0.1.0/24, add a tag KEY=LKS-ID, VALUE=MODUL3-PUBLIC SUBNET-B
   - Private subnet on availability zone A with network 10.0.2.0/24, add a tag KEY=LKS-ID, VALUE=MODUL3-PRIVATE-SUBNET-A
   - Private subnet on availability zone B with network 10.0.3.0/24, add a tag KEY=LKS-ID, VALUE=MODUL3-PRIVATE-SUBNET-B
2. Create an Internet Gateway and attach it to VPC that you have created before:
   - Add a tag to the internet gateway: KEY=LKS-ID, VALUE=MODUL3-IGW
3. Create a Public NAT gateway on this VPC, thus EC2 instances in the private subnet could access the internet to install required dependencies:

- Add a tag to the NAT gateway: KEY=LKS-ID, VALUE=MODUL3-NAT
4. Create 2 route tables for Public and Private subnets:
    - Add a tag to the public route: KEY=LKS-ID, VALUE=MODUL3-PUBLIC-ROUTE
    - Add a tag to the private route: KEY=LKS-ID, VALUE=MODUL3-PRIVATE-ROUTE
5. Associate public subnets with the public route table and add a route to 0.0.0.0/0 through Internet Gateway
6. Associate private subnets with the private route table and ad a route on 0.0.0.0/0 through NAT Gateway
7. Create a security group that will allow incoming requests (inbound) to EC2 instances in private subnet:
    - Allow HTTP and HTTPS
    - Allow PostgreSQL
    - Allow EFS
    - Allow SSH from the same network
    - Add a tag to the security group: KEY=LKS-ID, VALUE=MODUL3-SG-WEB
8. Create a security group that will allow incoming SSH requests for the EC2 Bastion Host instance in the public subnet:
    - Add a tag to the security group: KEY=LKS-ID, VALUE=MODUL3-SG-BASTION
9. Create an EFS for storing the source code from the GitHub repository
    - Select the private subnet and security group that you have created before
    - Add a tag to EFS: KEY=LKS-ID, VALUE=MODUL3-EFS
10. Create a DynamoDB for storing cache login sessions and data queries from the database:
    - Set the DynamoDB name to **lksccjabar2023-dynamodb**
    - Set partition key: **key** (S)
    - For the table settings, select customize setting, on the table class using the DynamoDB standard
    - For read and write capacity, select On-demand for cost optimization
    - Add a tag on the DynamoDB: KEY=LKS-ID, VALUE=MODUL3-DYNAMODB
11. Create a database using Amazon Aurora RDS with the following specification:
    - Select Aurora with PostgreSQL compatibility
    - Select PostgreSQL version 14.7
    - Specify the database cluster name: **lksccjabar2023-rds**
    - Specify the database name: **lksjabarmodul3**
    - For the master username, use the default username: **postgres**
    - Specify your master password (always remember your password and keep it in the safest place in the world)
    - Instance configuration using Serverless v2

- Make sure you create a multi-az deployment for this database
- Select your VPC for this database
- Don't let anyone access your database from the internet. Please disable public access.
- Select your private (MODUL3-SG-WEB) security group for this database. Only instances in a private subnet that can access this database
- Use default PostgreSQL port, 5432
- On additional configuration, specify the initial database name to **lksjabarmodul3**
- Disable deletion protection

12. Create EC2 launch templates with the following specification:
    - Operating Systems could be Ubuntu or Amazon Linux. Using a custom AMI is not allowed
    - EC2 instance type: t3.small
    - Create or use existing key pair
    - Use the security group that you have created web (MODUL3-SG-WEB)
    - In advanced details, you need to install the following required applications and dependencies using user data:
        - ● Nginx or Apache2 (http2) as web server
        - ● PHP version 8.1
        - ● And all PHP modules that are required by Laravel
        - ● NodeJS version 18.15.0
    - You must configure the EFS and clone the source code repository using user data:
        - ● Mount EFS on your EC2 instances in a private subnet, the directory for mounting this EFS is /var/www/modul3
        - ● Clone the Laravel source code on mounted EFS storage (/var/www/modul3)
    - Add a tag on the launch template: KEY=LKS-ID, VALUE=MODUL3-EC2-LAUNCH-TEMPLATE
13. Create an application load balancer for this Auto Scaling Group and attach it to Auto Scaling Group (later)
    - Specify load balancer name: **lksccjabar2023-elb**
    - Select the application load balancer
    - Select the internet-facing load balancer scheme
    - Create a new target group with the tag: KEY=LKS-ID, VALUE=MODUL3-ELB-TARGET
    - Add a tag to this load balancer: KEY=LKS-ID, VALUE=MODUL3-ELB
    - Enable health check
14. Create Auto Scaling group with the following specification:
    - Select your launch template
    - Select your VPC for this project

- Select Availability and Subnets
    - MODUL3-PRIVATE-SUBNET-A
    - MODUL3-PRIVATE-SUBNET-B
- Attach this Auto Scaling Group to the existing load balancer (the ELB that you have created before)
- On group size, you must specify the following requirements:
    - Desired capacity: 2
    - Minimum capacity: 2
    - Maximum capacity: 4
- For scaling policies, you must set up:
    - Metric type: Average CPU utilization
    - Target Value: 85
- Add a tag to this Auto Scaling Group: KEY=LKS-ID, VALUE=MODUL3-ASG

15. You also need to create an instance as Bastion Host for accessing (remote) EC2 instances in a private subnet:
    - Operating System could be Ubuntu or Amazon Linux. Using a custom AMI is not allowed
    - EC2 instance type: t3.small
    - Add a tag on this EC2 instance: KEY=MODUL3-BASTION
16. Set Amazon Route53, and add your hosted zone with a domain name that you ordered from Amazon Route53 itself, or from your favorite ISP:
    - Add a tag to the zone: KEY=LKS-ID, VALUE=MODUL3-ROUTE53
17. We need HTTPS protocol for securely accessing our website. So, you have to create a certificate on Amazon Certificate Manager for *.[YOUR_DOMAIN] and use it on Elastic Load Balancer:
    - Add a tag to ACM: KEY=LKS-ID, VALUE=MODUL3-ACM
18. Go to ELB that you have created before while and set up the following requirements:
    - Listener HTTP (80) must redirect to HTTPS (443)
    - Use your certificate from ACM on this load balancer listener
19. Back to Route53, create a zone for ELB with domain name modul3.[YOUR_DOMAIN]. Thus, our users can easily access https://modul3.[YOUR_DOMAIN].
20. Now, let’s set up our news application. After you run the Auto Scaling Group, then the instances will be created. In that process, the user data script that you entered on EC2 Launch Template will be executed. Thus, you don’t need to install any applications and dependencies anymore (it’s automated by user data script).
21. The final step is, you need to configure the Laravel application. This Laravel application is the source code that you clone with user data script on EC2 launch templates. Access your private instances through a bastion host, then run the following command on the document root:
    - Install Composer version 2.5.2

- Generate a new key: **php artisan key:generate**
- Clear config cache:
    - **php artisan cache:clear**
    - **php artisan config:clear**
- Database migrate: **php artisan migrate –force –isolated**
- Create config cache:
    - **php artisan config:cache**
    - **php artisan route:cache**
    - **php artisan view:cache**
- Activate storage link: **php artisan storage:link**

6. How to test your deployment

1. Access our news web application https://modul3.[YOUR_DOMAIN]
2. You can sign up and sign in, then write your own news!
3. If you can sign in, then the cache or session for authentication works.
4. If you can write, update, or delete news, then our application can connect to the RDS database, congratulations!
5. For testing the Auto Scaling Group, destroy or terminate one instance in the private subnet. If the application still works and the user can access our News application, then the scalable cloud infrastructure in this project is perfect!

7. References

- Amazon Virtual Private Cloud
- Amazon Elastic Compute Cloud
- Elastic Load Balancing
- Amazon EC2 Auto Scaling
- Amazon Route 53
- AWS Certificate Manager
- Amazon Aurora
- Amazon Elastic File System
- NodeJS and NPM Documentation
- Laravel
- Composer
- PHP
- PHP
- Nginx
- Nginx