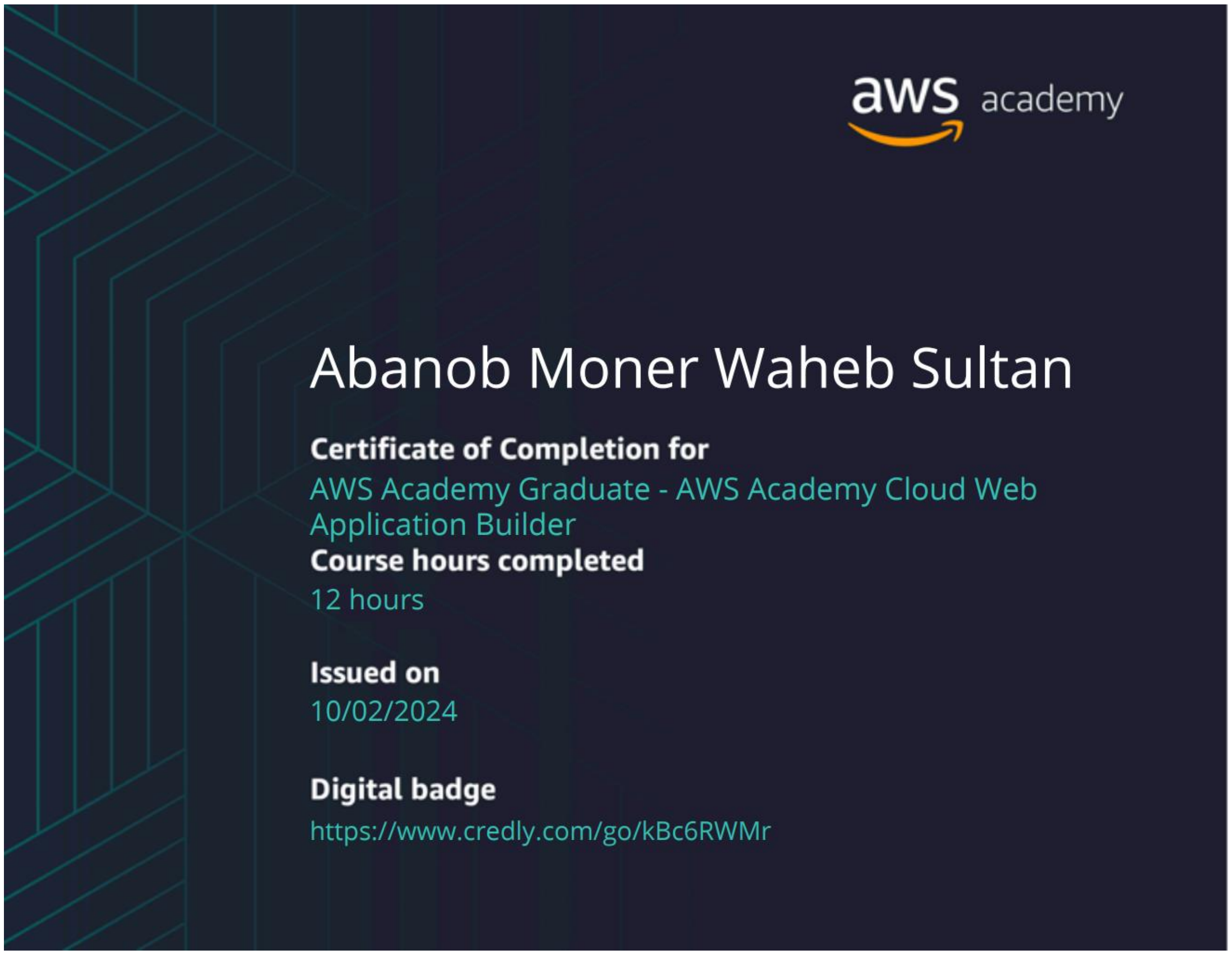
Cloud Web Application Builder Project



Abanob Moner Waheb Sultan

Certificate of Completion for
AWS Academy Graduate - AWS Academy Cloud Web Application Builder
Course hours completed
12 hours

Issued on
10/02/2024

Digital badge
https://www.credly.com/go/kBc6RWMr

Project Overview

The goal of this project was to develop a cloud-based web application for managing student records at Example University. This initiative was prompted by performance issues experienced during peak admissions periods, necessitating a robust solution capable of handling high traffic volumes efficiently.

We're challenged for designing an architecture that supports high availability and scalability, ensuring secure access to sensitive data, implementing load balancing to manage user traffic effectively and maintaining optimal performance under varying loads.

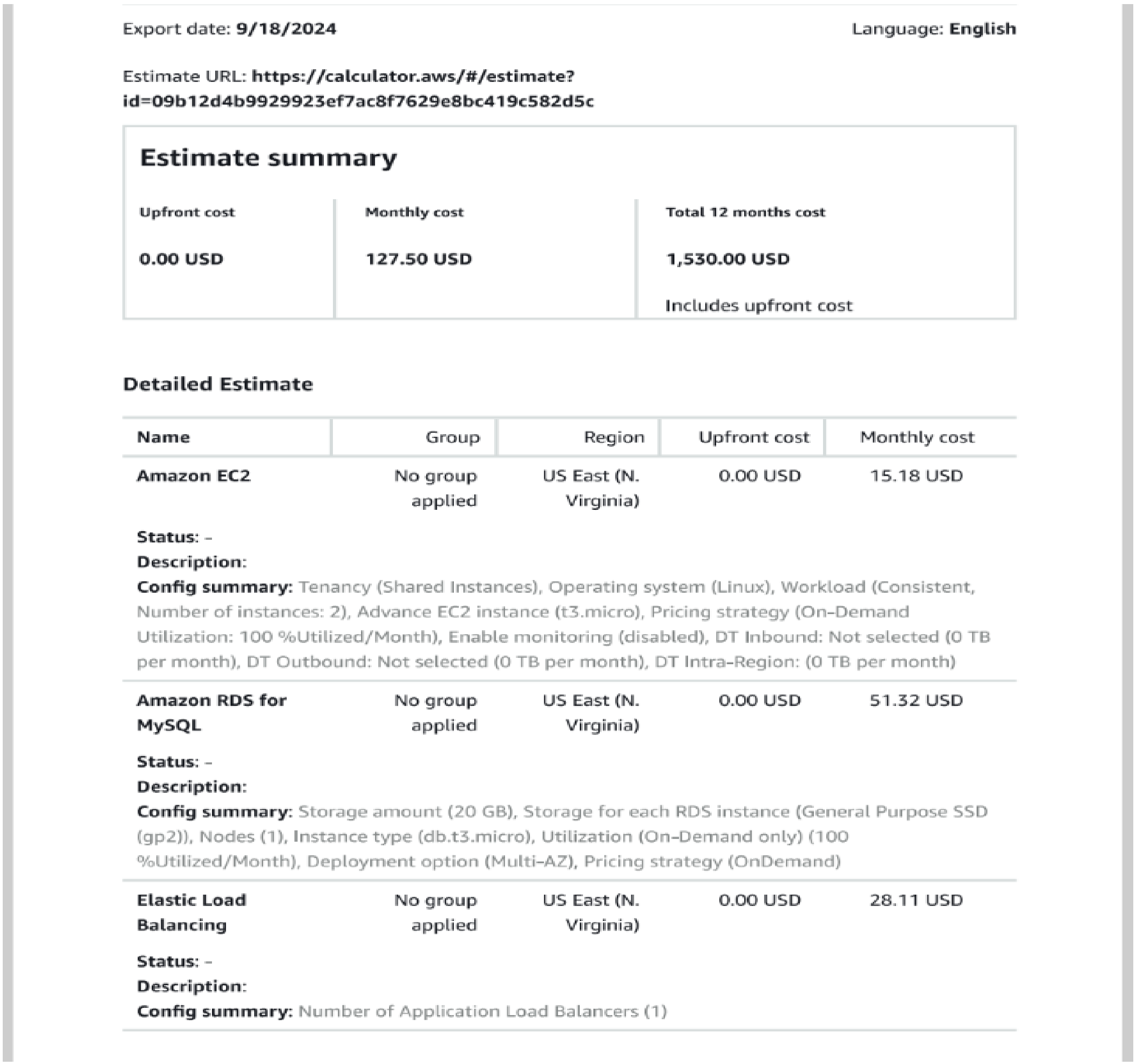
Phase 1: Planning the design and estimating cost

Task 1: Creating an architectural diagram

- **Create a VPC**: Set up a VPC with an IPv4 CIDR block of 10.0.0.0/16.
- **Define Subnets**:
    - **Public Subnets**: Two public subnets (10.0.1.0/24 and 10.0.2.0/24) for the web server testing.
    - **Private Subnets**: Two private subnets (10.0.3.0/24 and 10.0.4.0/24) for RDS and web servers.

Task 2: Developing a cost estimate

- **AWS Pricing Calculator**: Used to estimate costs for the following services:
    - **EC2 Instances**: Based on instance types and expected usage.
    - **RDS**: Configured with db.t3.micro, estimating costs based on the instance type and storage.
    - **Elastic Load Balancer**: Costs associated with traffic distribution.
    - **NAT Gateway**: Pricing for NAT usage to allow private subnet instances to access the internet.



Export date: **9/18/2024** Language: **English**

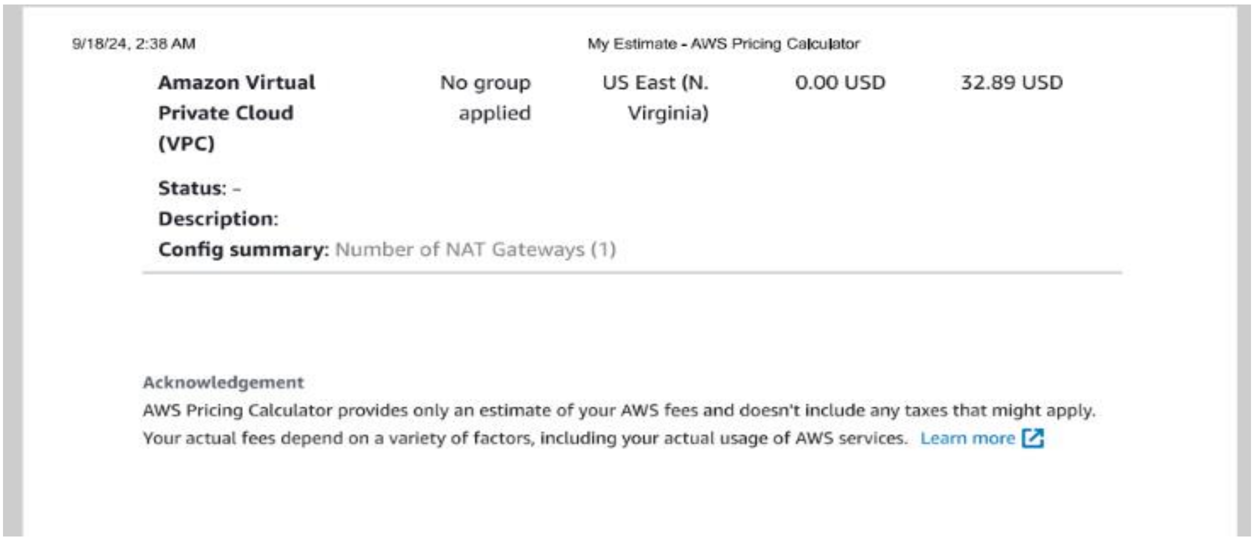
Estimate URL: **https://calculator.aws/#/estimate?id=09b12d4b9929923ef7ac8f7629e8bc419c582d5c**

Estimate summary

Upfront cost	Monthly cost	Total 12 months cost
0.00 USD	**127.50 USD**	**1,530.00 USD** Includes upfront cost

Detailed Estimate

Name	Group	Region	Upfront cost	Monthly cost
Amazon EC2	No group applied	US East (N. Virginia)	0.00 USD	15.18 USD
Status: - **Description:** **Config summary:** Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 2), Advance EC2 instance (t3.micro), Pricing strategy (On-Demand Utilization: 100 %Utilized/Month), Enable monitoring (disabled), DT Inbound: Not selected (0 TB per month), DT Outbound: Not selected (0 TB per month), DT Intra-Region: (0 TB per month)				
Amazon RDS for MySQL	No group applied	US East (N. Virginia)	0.00 USD	51.32 USD
Status: - **Description:** **Config summary:** Storage amount (20 GB), Storage for each RDS instance (General Purpose SSD (gp2)), Nodes (1), Instance type (db.t3.micro), Utilization (On-Demand only) (100 %Utilized/Month), Deployment option (Multi-AZ), Pricing strategy (OnDemand)				
Elastic Load Balancing	No group applied	US East (N. Virginia)	0.00 USD	28.11 USD
Status: - **Description:** **Config summary:** Number of Application Load Balancers (1)				



9/18/24, 2:38 AM My Estimate - AWS Pricing Calculator

Amazon Virtual Private Cloud (VPC)	No group applied	US East (N. Virginia)	0.00 USD	32.89 USD
Status: - **Description:** **Config summary:** Number of NAT Gateways (1)				

Acknowledgement

AWS Pricing Calculator provides only an estimate of your AWS fees and doesn't include any taxes that might apply. Your actual fees depend on a variety of factors, including your actual usage of AWS services. Learn more

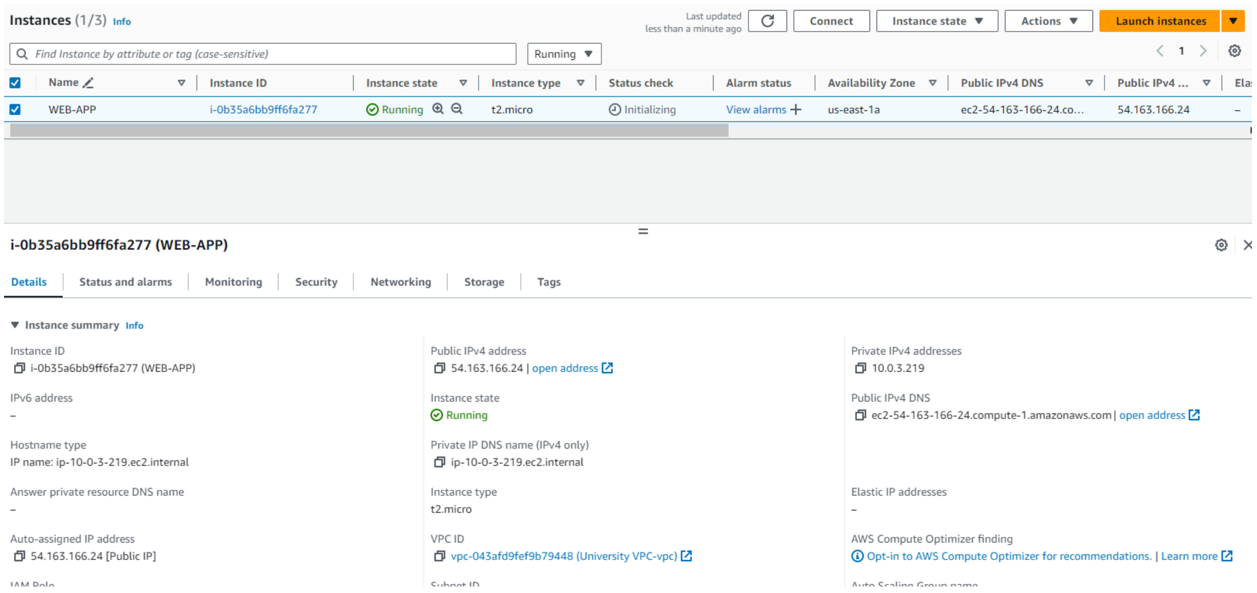
Phase 2: Creating a basic functional web application

Task 1: Creating the VPC

- **Navigate to the VPC Dashboard** in the AWS Management Console.
- **Create a New VPC** with the CIDR block 10.0.0.0/16.
- **Create Subnets**: Define two public and two private subnets as specified.

Task 2: Launching EC2 Instances

- **Select an AMI**: Choose the latest **Ubuntu** AMI.
- **Configure Instance Details**:
  - Select the VPC and appropriate subnet.
  - Enable Auto-assign Public IP for instances in public subnets.
  - Add the user data for the application.


Instances (1/3) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
Running
1

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Ela
WEB-APP	i-0b35a6bb9ff6fa277	Running	t2.micro	Initializing	View alarms +	us-east-1a	ec2-54-163-166-24.co...	54.163.166.24	–

i-0b35a6bb9ff6fa277 (WEB-APP)

Instance ID i-0b35a6bb9ff6fa277 (WEB-APP)	Public IPv4 address 54.163.166.24 | open address	Private IPv4 addresses 10.0.3.219
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-54-163-166-24.compute-1.amazonaws.com | open address
Hostname type IP name: ip-10-0-3-219.ec2.internal	Private IP DNS name (IPv4 only) ip-10-0-3-219.ec2.internal	
Answer private resource DNS name –	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address 54.163.166.24 [Public IP]	VPC ID vpc-043afd9fef9b79448 (University VPC-vpc)	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more

1. **Configure Security Groups**:

   1- Create a security group allowing HTTP (port 80) and SSH (port 22) access for the web server

Task 3: Testing the Deployment

Access the public IP address of the EC2 instance to ensure the web application is accessible.

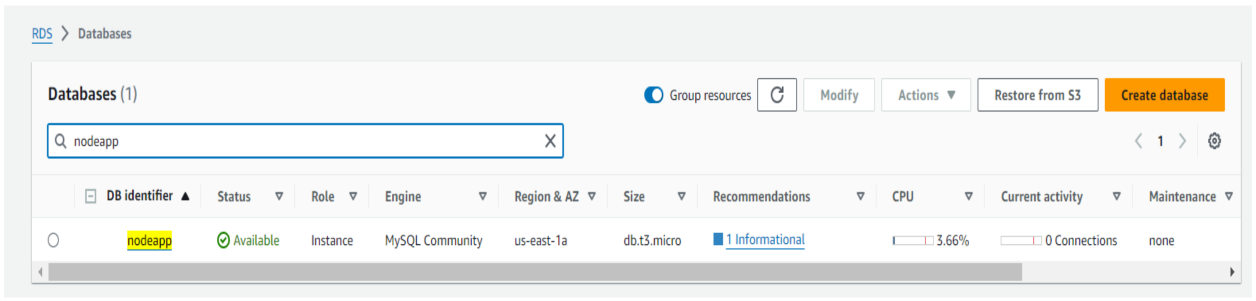
<u>Phase 3: Decoupling the application components</u>

Task 1: Configuring Amazon RDS

- Create an RDS Instance:

- Choose MySQL as the database engine.
- Set the instance type to db.t3.micro and configure storage options.
- Place the database in a private subnet for security.

- Configure Security Groups: Ensure the security group for the RDS allows inbound traffic on port 3306 from the web application security group.



DB identifier	Status	Role	Engine	Region & AZ	Size	Recommendations	CPU	Current activity	Maintenance
nodeapp	Available	Instance	MySQL Community	us-east-1a	db.t3.micro	1 Informational	3.66%	0 Connections	none

Task 2: Setting Up the Development Environment

- Provision AWS Cloud9:

- Create a Cloud9 environment using an EC2 instance.
- Install necessary tools (e.g., AWS CLI, MySQL client).

Phase 4: High Availability and Scalability

Task 1: Setting Up Load Balancer

- Create an Application Load Balancer:

- Choose the VPC and subnets for the load balancer.
- Configure listeners to route traffic to the EC2 instances.
- Associate the security group that allows HTTP traffic from everywhere.

Task 2: Configuring Auto Scaling

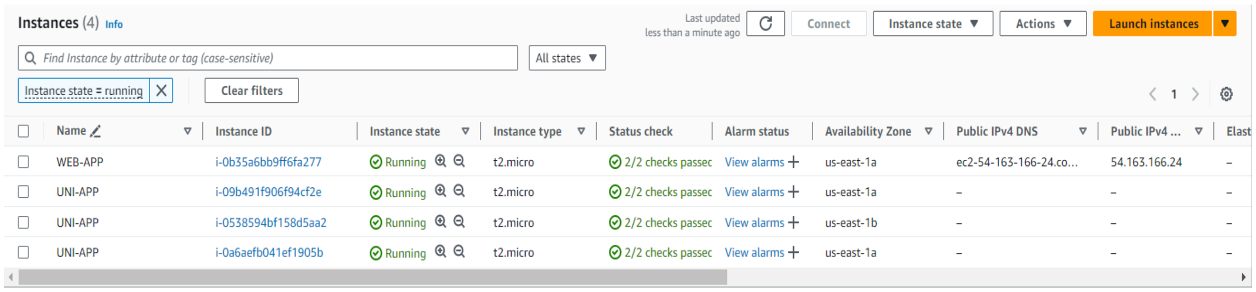
- Create an Auto Scaling Group:

- Use a launch template based on the EC2 instance configuration.
- Specify the IAM role LabRole for the instances to access AWS resources securely.
- Add the user data for the application
- Set minimum and maximum instance counts based on expected traffic.
- Enable scaling policies to adjust the number of instances automatically.

Task 4: Load Testing the Application

- Use a tool like load test to simulate traffic and evaluate performance:
- Execute the command to generate load: loadtest --rps 1000 -c 500 <ELB URL>
- Monitor application behavior and response times.

- You will find other instances initialized



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elast
WEB-APP	i-0b35a6bb9ff6fa277	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	us-east-1a	ec2-54-163-166-24.co...	54.163.166.24	–
UNI-APP	i-09b491f906f94cf2e	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	us-east-1a	–	–	–
UNI-APP	i-0538594bf158d5aa2	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	us-east-1b	–	–	–
UNI-APP	i-0a6aefb041ef1905b	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	us-east-1a	–	–	–

Conclusion

The project successfully established a cloud-based web application using AWS services. The architecture ensures high availability and scalability while maintaining security and performance.

The integration of services like VPC, EC2, RDS, load balancers, and AWS Secrets Manager creates a resilient environment capable of handling peak traffic efficiently. This foundational work sets the stage for future enhancements and projects within AWS.