



IT 334 - DevOps Engineering on AWS cloud

PROJECT DOCUMENTATION

Prepared by:
Ajla Korman
Kerim Šabić

Jun, 2024

In this document you can find the solution for building a Highly Available, Scalable Web Application.

The problem is following:

Example University is preparing for the new school year.

The admissions department has received complaints that their web application for student records is slow or not available during the peak admissions period because of the high number of inquiries.

You are a cloud engineer. Your manager has asked you to create a proof of concept (POC) to host the web application in the AWS Cloud.

Your manager would like you to design and implement a new hosting architecture that will improve the experience for users of the web application.

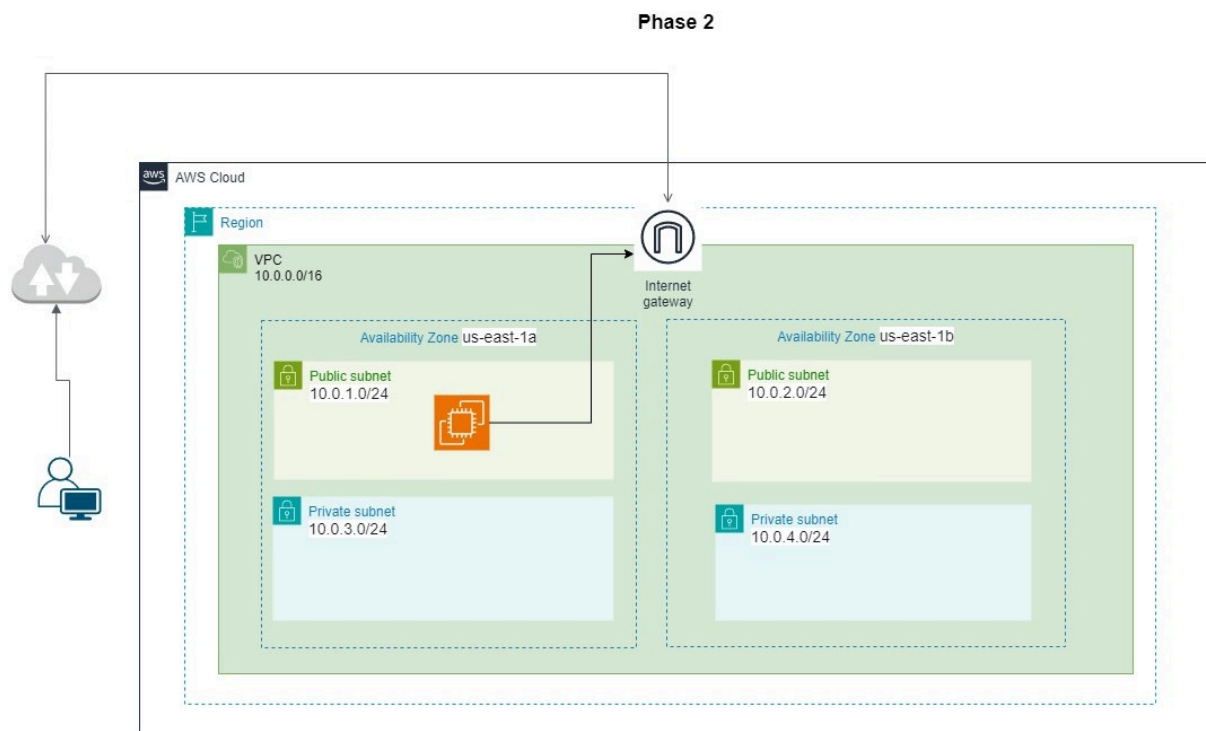
You're responsible for building the infrastructure to host the student records web application in the cloud. Your challenge is to plan, design, build, and deploy the web application to the AWS Cloud in a way that is consistent with best practices of the AWS Well-Architected Framework.

During the peak admissions period, the application must support thousands of users, and be highly available, scalable, load balanced, secure, and high performing.

Phase 2 - Creating a basic functional web application

In this phase we will create a basic architecture for our web application.

The diagram below shows the structure for this phase:



The first step is creating a VPC, Virtual Private Cloud.

Go to the VPC service, and click Create VPC.

In the name tag section put the name of the VPC, in our case it will be called “DevOps-group11”.

In the IPV4 CIDR enter the CIDR block size which is in our case “10.0.0.0/16”, this determines the number of IP addresses available for your resources and subnets.

Click on the Create VPC button and your VPC should be created.

Then select the created VPC, go to actions and click edit VPC settings. Make sure you enable dns resolutions and hostnames like this:

DNS settings

☒ Enable DNS resolution

Info

☒ Enable DNS hostnames

Info

At the end your configuration should look like this:

vpc-006a73ed23fe2f972 / DevOps-group11

Details

Resource map




CIDRs

Flow logs

Tags

Integrations

Details

<div>VPC ID</div> <div> vpc-006a73ed23fe2f972</div>	<div>State</div> <div> Available</div>	<div>DNS hostnames</div> <div>Enabled</div>	<div>DNS resolution</div> <div>Enabled</div>
<div>Tenancy</div> <div>Default</div>	<div>DHCP option set</div> <div>dopt-03f7a3e68c9fdc755</div>	<div>Main route table</div> <div>rtb-0f6df04ad614b482e</div>	<div>Main network ACL</div> <div>acl-0082eca343fb31732</div>
<div>Default VPC</div> <div>No</div>	<div>IPv4 CIDR</div> <div>10.0.0.0/16</div>	<div>IPv6 pool</div> <div>—</div>	<div>IPv6 CIDR (Network border group)</div> <div>—</div>
<div>Network Address Usage metrics</div> <div>Disabled</div>	<div>Route 53 Resolver DNS Firewall rule groups</div> <div>—</div>	<div>Owner ID</div> <div> 917837302529</div>	

Next up we will create an Internet Gateway that is used by the resources in the VPC to communicate with the internet.




In the menu on the left select Internet gateways and click Create Internet gateway. Put the name of the internet gateway, in our case it will be DevOps-group11IG and click create internet gateway. Then choose actions and attach the Internet gateway to the VPC we have created.

igw-02c022c4dbb6f2b45 / DevOps-group11IG

Details

Tags

Details

<div>Internet gateway ID</div> <div> igw-02c022c4dbb6f2b45</div>	<div>State</div> <div> Attached</div>	<div>VPC ID</div> <div>vpc-006a73ed23fe2f972 DevOps-group11</div>	<div>Owner</div> <div> 917837302529</div>
---	--	---	--

Next we will create 2 subnets in one availability zone and again 2 subnets in the second availability zone to make our application highly available. In each availability zone we will have one private and one public subnet.

Go to the subnet in the left menu and click Create subnets.

Select the VPC we have created and enter the IPV4CIDR address range and create the subnet.

The first public subnet in the first availability zone should look like this:

Details			
Subnet ID subnet-0490096a0cd1f2e26	Subnet ARN arn:aws:ec2:us-east-1:917837302529:subnet/subnet-0490096a0cd1f2e26	State Available	IPv4 CIDR 10.0.1.0/24
Available IPv4 addresses 246	IPv6 CIDR -	Availability Zone us-east-1a	Availability Zone ID use1-az6
Network border group us-east-1	VPC vpc-006a73ed23fe2f972 DevOps-group11	Route table rtb-0c6d9f771894c7701 DevOps-group11-publicRT	Network ACL acl-0082eca343fb31732
Default subnet No	Auto-assign public IPv4 address Yes	Auto-assign IPv6 address No	Auto-assign customer-owned IPv4 address No
Customer-owned IPv4 pool -	Outpost ID -	IPv4 CIDR reservations -	IPv6 CIDR reservations -
IPv6-only No	Hostname type IP name	Resource name DNS A record Disabled	Resource name DNS AAAA record Disabled
DNS64 Disabled	Owner 917837302529		

The first private subnet in the first availability zone should look like this:

Details			
Subnet ID subnet-004975e1d1812fae2	Subnet ARN arn:aws:ec2:us-east-1:917837302529:subnet/subnet-004975e1d1812fae2	State Available	IPv4 CIDR 10.0.3.0/24
Available IPv4 addresses 251	IPv6 CIDR -	Availability Zone us-east-1a	Availability Zone ID use1-az6
Network border group us-east-1	VPC vpc-006a73ed23fe2f972 DevOps-group11	Route table rtb-05e5ed80f0604697e DevOps-group11-privateRT	Network ACL acl-0082eca343fb31732
Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No	Auto-assign customer-owned IPv4 address No
Customer-owned IPv4 pool -	Outpost ID -	IPv4 CIDR reservations -	IPv6 CIDR reservations -
IPv6-only No	Hostname type IP name	Resource name DNS A record Disabled	Resource name DNS AAAA record Disabled
DNS64 Disabled	Owner 917837302529		

The first public subnet in the second availability zone should look like this:

Details			
Subnet ID subnet-05b1429f75d573ef5	Subnet ARN arn:aws:ec2:us-east-1:917837302529:subnet/subnet-05b1429f75d573ef5	State Available	IPv4 CIDR 10.0.2.0/24
Available IPv4 addresses 248	IPv6 CIDR -	Availability Zone us-east-1b	Availability Zone ID use1-az1
Network border group us-east-1	VPC vpc-006a73ed23fe2f972 DevOps-group11	Route table rtb-0c6d9f771894c7701 DevOps-group11-publicRT	Network ACL acl-0082eca343fb31732
Default subnet No	Auto-assign public IPv4 address Yes	Auto-assign IPv6 address No	Auto-assign customer-owned IPv4 address No
Customer-owned IPv4 pool -	Outpost ID -	IPv4 CIDR reservations -	IPv6 CIDR reservations -
IPv6-only No	Hostname type IP name	Resource name DNS A record Disabled	Resource name DNS AAAA record Disabled
DNS64 Disabled	Owner 917837302529		

The first private subnet in the second availability zone should look like this:

Details			
Subnet ID subnet-045c1a33f2a5affdb	Subnet ARN arn:aws:ec2:us-east-1:917837302529:subnet/subnet-045c1a33f2a5affdb	State Available	IPv4 CIDR 10.0.4.0/24
Available IPv4 addresses 251	IPv6 CIDR -	Availability Zone us-east-1b	Availability Zone ID use1-az1
Network border group us-east-1	VPC vpc-006a73ed23fe2f972 DevOps-group11	Route table rtb-05e5ed80f0604697e DevOps-group11-privateRT	Network ACL acl-0082eca343fb31732
Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No	Auto-assign customer-owned IPv4 address No
Customer-owned IPv4 pool -	Outpost ID -	IPv4 CIDR reservations -	IPv6 CIDR reservations -
IPv6-only No	Hostname type IP name	Resource name DNS A record Disabled	Resource name DNS AAAA record Disabled
DNS64 Disabled	Owner 917837302529		

Make sure you enable auto-assign public IPV4 address on the public subnets in each availability zone.

After this we will create NAT Gateway. You can use a NAT gateway so that instances in a private subnet can connect to services outside your VPC but external services cannot initiate a connection with those instances.

Click on the create NAT Gateway and create a nat gateway so it looks like this:

Details			
NAT gateway ID nat-03ad6713da717eabe	Connectivity type Public	State Available	State message -
NAT gateway ARN arn:aws:ec2:us-east-1:917837302529:natgateway/nat-03ad6713da717eabe	Primary public IPv4 address 52.73.6.37	Primary private IPv4 address 10.0.1.187	Primary network interface ID eni-00cd8794a2b5d5c10
VPC vpc-006a73ed23fe2f972 / DevOps-group11	Subnet subnet-0490096a0cd1f2e26 / DevOps-group11-us-east-1a-public1	Created Tuesday, June 4, 2024 at 13:11:56 GMT+2	Deleted -

Next we will create private and public route tables.

Click on Create route table, put the name of the public route table and select the VPC we created. We will create routes to route the internet traffic to the internet gateway we created and request within our VPC to be routed locally.

When created click on routes, edit routes and add the following routes:

Routes (2)			
<input type="text" value="Filter routes"/> Both Edit routes			
Destination	Target	Status	Propagated
0.0.0.0/0	igw-02c022c4dbb6f2b45	Active	No
10.0.0.0/16	local	Active	No

Click on the subnet associations and associate to the both public subnets we created.

Next we will create a private route table.

Click on Create route table, put the name of the private route table and select the VPC we created. We will create routes to route the internet traffic to the NAT gateway we created and request within our VPC to be routed locally.

When created click on routes, edit routes and add the following routes:

rtb-05e5ed80f0604697e / DevOps-group11-privateRT

Details Routes Subnet associations Edge associations Route propagation Tags

Routes (2)				Both ▼	Edit routes
<input type="text" value="Filter routes"/>				< 1 >	ⓘ
Destination ▼	Target ▼	Status ▼	Propagated ▼		
0.0.0.0/0	nat-03ad6713da717eabe	✔ Active	No		
10.0.0.0/16	local	✔ Active	No		

Click on the subnet associations and associate the route table with both private subnets we created.

Next we will create an EC2 instance to host our web application.

Find the EC2 service and click on the Create Instance button.

Put the name of the instance, for the application on the os image select Ubuntu, instance type should be t2.micro, for keypair chose vockey

In the network settings click edit. Select our VPC, put the insane in the first public subnet we created, enable auto assign public ip

Click create security group, here we will create security group for our ec2 instance

Enter the security group name and add the rules so at the end the security group look like this:

sg-08fa909fdf9ae1447 - DevOps-group11SG1

Details Inbound rules Outbound rules Tags

Inbound rules (2)												⌂	Manage tags	Edit inbound rules
<input type="text" value="Search"/>												< 1 >	ⓘ	
<input type="checkbox"/>	Name ▼	Security group rule ID ▼	IP version ▼	Type ▼	Protocol ▼	Port range ▼	Source ▼	Description						
<input type="checkbox"/>	-	sgr-080a0f7ff237d391a	IPv4	HTTP	TCP	80	0.0.0.0/0	-						
<input type="checkbox"/>	-	sgr-0a3f2412cbd0cdf32	IPv4	MYSQL/Aurora	TCP	3306	10.0.0.0/16	-						

sg-08fa909fdf9ae1447 - DevOps-group11SG1

Details Inbound rules Outbound rules Tags

Outbound rules (1)											⌂	Manage tags	Edit outbound rules
<input type="text" value="Search"/>											< 1 >	ⓘ	
<input type="checkbox"/>	Name ▼	Security group rule ID ▼	IP version ▼	Type ▼	Protocol ▼	Port range ▼	Destination ▼	Description					
<input type="checkbox"/>	-	sgr-0bec10de5db5c0a89	IPv4	All traffic	All	All	0.0.0.0/0	-					

After creating the security group select it in the EC2 creation menu.
Scroll down to the advanced details, expand it and paste the user data that is given in the UserdataScript-phase2

Our instance should be running now and we should be able to access our application by our public IPV4 DNS.

i-01559bad6526b0256 (DevOps-gr11-1)

Details

Status and alarms

Monitoring

Security

Networking

Storage

Tags

▼ Instance summary info

Instance ID

i-01559bad6526b0256 (DevOps-gr11-1)

IPv6 address

—

Hostname type

IP name: ip-10-0-1-49.ec2.internal

Answer private resource DNS name

—

Auto-assigned IP address

52.90.17.106 [Public IP]

IAM Role

—

IMDSv2

Optional

⚠️ EC2 recommends setting IMDSv2 to required | [Learn more](#)

Public IPv4 address

52.90.17.106 | [open address](#)

Instance state

🟢 Running

Private IP DNS name (IPv4 only)

ip-10-0-1-49.ec2.internal

Instance type

t2.micro

VPC ID

vpc-006a73ed23fe2f972 (DevOps-group11) |

Subnet ID

subnet-0490096a0cd1f2e26 (DevOps-group11-us-east-1a-public1) |

Instance ARN

arn:aws:ec2:us-east-1:917837302529:instance/i-01559bad6526b0256

Private IPv4 addresses

10.0.1.49

Public IPv4 DNS

ec2-52-90-17-106.compute-1.amazonaws.com | [open address](#)

Elastic IP addresses

—


AWS Compute Optimizer finding

[Opt-in to AWS Compute Optimizer for recommendations.](#) | [Learn more](#)

Auto Scaling Group name

—

We can now perform all CRUD operations on our application:



XYZ University

Home

Students list

All students

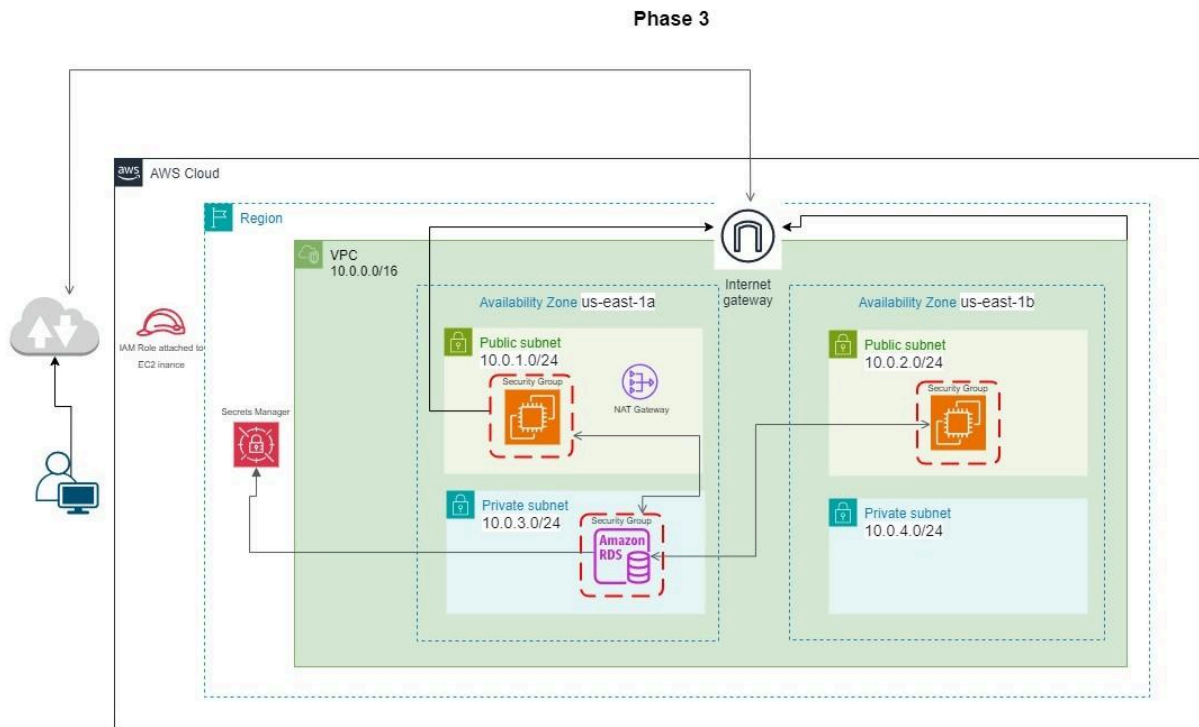
Name	Address	City	State	Email	Phone	
Kerim	AasdaDS	Sarajevo	Federation of Bosnia and Herzegovina	tarik@gmail.com	060 000 000	edit
asdf	AasdaDS	Sarajevo	Federation of Bosnia and Herzegovina	kerimsabic3@gmail.com	062046137	edit
K	Sarajevo	Sarajevo	Federation of Bosnia and Herzegovina	kerim.sabic@stu.ibu.edu.ba	16546843	edit

[Add a new student](#)

Phase 3 - Decoupling the application components

In this phase we will create a RDS MySQL database that will run on the private subnet in the same availability zone as our application. We will transfer the data from the database running on the EC2 instance to our RDS database. With this approach we can create new instances of our application and connect it to the rds database instead of saving data to that instance. This will help to make our application highly available.

The diagram below shows the structure for this phase:



First create subnet groups for the database.

Go to the rds service, and click create subnet groups. When creating associate 2 private subnets we have created.

Subnet group details		
VPC ID vpc-006a73ed23fe2f972		
ARN arn:aws:rds:us-east-1:917837302529:subgrp:devops-gr1-subnetgroup		
Supported network types IPv4		
Description Database subnet group		
Subnets (2)		
Availability zone	Subnet ID	CIDR block
us-east-1a	subnet-004975e1d1812fae2	10.0.3.0/24
us-east-1b	subnet-045c1a33f2a5affdb	10.0.4.0/24

Go to the RDS service in the AWS console and create a new database instance. For the engine type choose MySQL, engine version should be the latest stable one, for templates use Production, select single db instance. For the database instance id put Students and, for the master username put nodeapp, click self managed and add a password for the database. DB instance class should be db.t3.micro, add your VPC and subnet group we created previously.

Created database should look like this:

Connectivity & security		
Endpoint & port Endpoint students.cxkilkdbcrd.us-east-1.rds.amazonaws.com Port 3306	Networking Availability Zone us-east-1a VPC DevOps-group11 (vpc-006a73ed23fe2f972) Subnet group devops-gr1-subnetgroup Subnets subnet-004975e1d1812fae2 subnet-045c1a33f2a5affdb Network type IPv4	Security VPC security groups DevOps-group11SG1 (sg-08fa909df9ae1447) Active Publicly accessible No Certificate authority Info rds-ca-rsa2048-g1 Certificate authority date May 26, 2061, 01:34 (UTC+02:00) DB instance certificate expiration date June 05, 2025, 20:16 (UTC+02:00)

Open Cloud9 service and create a new environment. New environment should be created with a type of new EC2 instance, use t3.micro for platform use linux for platform, for connection use ssh, use created VPC and put in the wanted subnet, in our case first public one in the first availability zone.

After creating instance, open the cloud9 environment.

Now we need to create a secret manager for storing our database secret. In the terminal paste the following command:

```
aws secretsmanager create-secret \
  --name Mydbsecret \
  --description "Database secret for web app" \
```

--secret-string

```
"{\"user\":\"nodeapp\",\"password\":\"ADD_PASSWORD\",\"host\":\"students.cxjilqdbcrd.us-east-1.rds.amazonaws.com\",\"db\":\"STUDENTS\"}"
```

This will create a new secret with AWS Secrets manager which you can see when you go to this service, select the Mydbsecret and click on the button to retrieve secret value.

Next up we will drop the database that sits on the ec2 instance we first created into the data.sql file. We can do this by pasting the following script:

```
mysqldump -h <enter private ip of your ec2 instance> -u nodeapp -p --databases STUDENTS > data.sql
```

New file should be created called data.sql, we will now import this data into our database we created with rds.

Paste the following script:

```
mysql -h <here put the endpoint of the rds database you created> -u nodeapp -p STUDENTS < data.sql
```

Now create a new EC2 instance following the same guidelines as for the first, just in this one in the network section put it inside the second public subnet in other availability zone and also for the user data paste the user data from the UserDataScript-phase3.

Also attach the user role to the instance. User role is LabRole.

At the end EC2 should look like this:

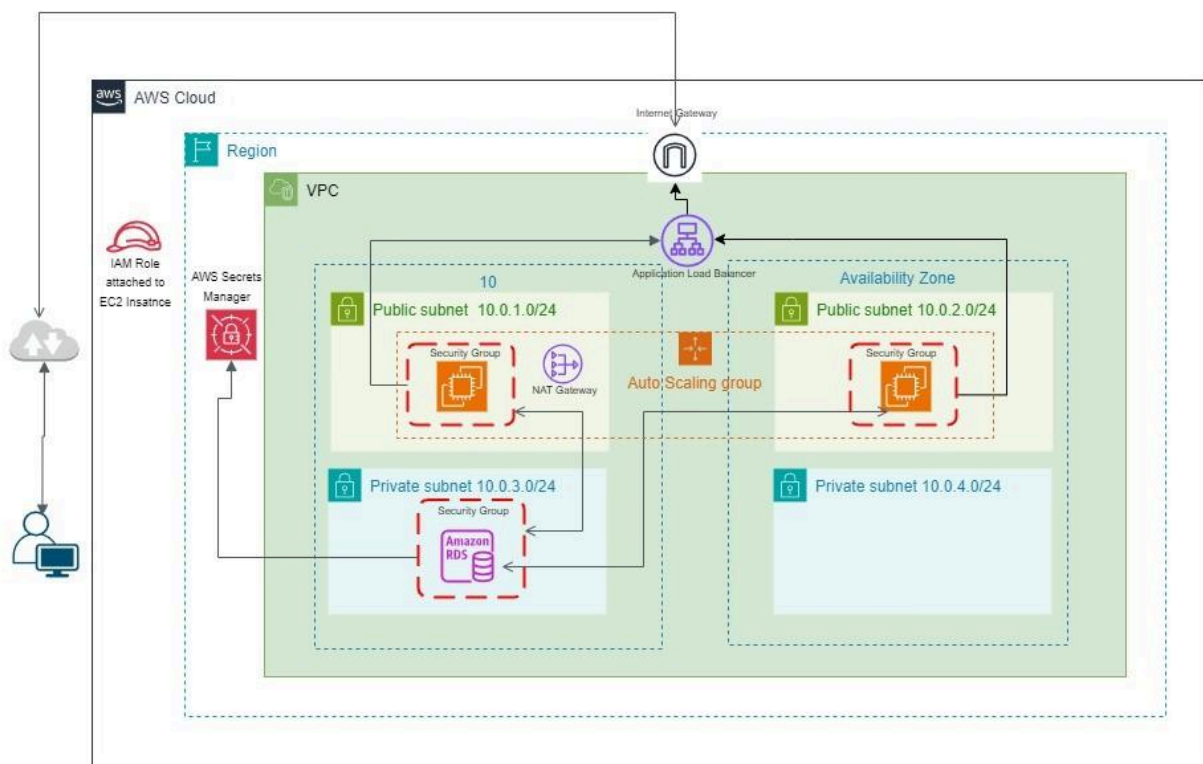
The screenshot displays the AWS Management Console for an EC2 instance. The instance is named 'i-0111f0d70213c9b00 (DevOps-gr11-3)' and is currently in a 'Running' state. The console shows various configuration details organized into sections: Instance summary, Instance ID, IPv6 address, Hostname type, Answer private resource DNS name, Auto-assigned IP address, IAM Role (LabRole), IMDSv2 status (Optional, with a warning to set to required), Public IPv4 address (3.238.188.94), Instance state (Running), Private IP DNS name (ip-10-0-2-33.ec2.internal), Instance type (t2.micro), VPC ID (vpc-006a73ed23fe2f972), Subnet ID (subnet-05b1429f75d573ef5), and Instance ARN (arn:aws:ec2:us-east-1:917837302529:instance/i-0111f0d70213c9b00). It also lists Private IPv4 addresses (10.0.2.33), Public IPv4 DNS (ec2-3-238-188-94.compute-1.amazonaws.com), Elastic IP addresses, and an AWS Compute Optimizer finding recommending to opt-in to AWS Compute Optimizer for recommendations.

Now the database should be connected to the ec2 instances and they query data from the RDS endpoint.

Phase 4 - Implementing high availability and scalability

In this phase we create fully functional, highly available and scalable applications by creating load balancer, load balancer security group, target group, Amazon machine image, launch template and auto scaling group.

The diagram below shows the final structure:



The first step is creating a load balancer.

Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones.

It monitors the health of its registered targets, and routes traffic only to the healthy targets.

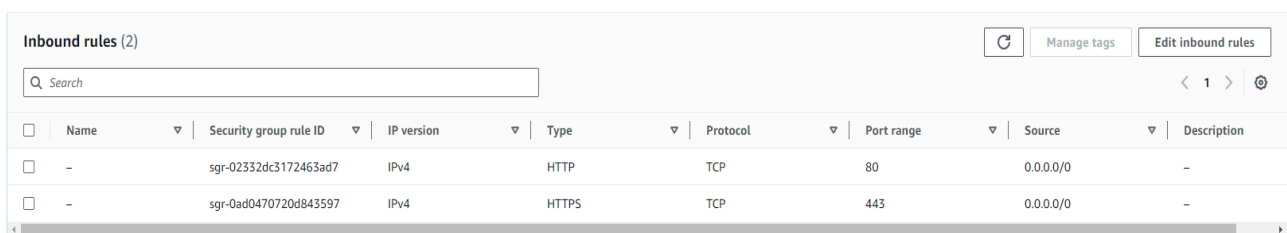
We will create an application load balancer.

Click on the create load balancer, then choose application load balancer, put the name of the load balancer, select internet facing.

In the networks mapping section, select your VPC, the one we created, then select 2 availability zones where you will select 2 public subnets, one in each availability zone.

For the security group select create new security group. Here we will create a new security group for our load balancer.

Put the name of the security group, select our VPC and add inbound rules like this:

A screenshot of the AWS Management Console showing the 'Inbound rules (2)' configuration for a security group. The interface includes a search bar, a refresh button, and buttons for 'Manage tags' and 'Edit inbound rules'. A table lists two inbound rules. The first rule allows HTTP traffic (port 80) from any source (0.0.0.0/0). The second rule allows HTTPS traffic (port 443) from any source (0.0.0.0/0). Both rules are for IPv4 and use the TCP protocol.

<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
<input type="checkbox"/>	-	sgr-02332dc3172463ad7	IPv4	HTTP	TCP	80	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-0ad0470720d843597	IPv4	HTTPS	TCP	443	0.0.0.0/0	-

This will allow HTTP and HTTPS traffic to our load balancer.

After creating this security group, return to the load balancer configuration and for the security group select the newly created security group.

Then in the listening and routing section click on the create target group.

In new window select instances, put the name of the target group, select our VPC and in the advanced details configure following:

▼ Advanced health check settings

Restore defaults

Health check port

The port the load balancer uses when performing health checks on targets. By default, the health check port is the same as the target group's traffic port. However, you can specify a different port as an override.

☒ Traffic port
 ☐ Override

Healthy threshold

The number of consecutive health checks successes required before considering an unhealthy target healthy.

2-10

Unhealthy threshold

The number of consecutive health check failures required before considering a target unhealthy.

2-10

Timeout

The amount of time, in seconds, during which no response means a failed health check.

seconds

2-120

Interval

The approximate amount of time between health checks of an individual target

seconds

5-300

Success codes

The HTTP codes to use when checking for a successful response from a target. You can specify multiple values (for example, "200,202") or a range of values (for example, "200-299").

Target groups route requests to individual registered targets, such as EC2 instances, using the protocol and port number that you specify.

In the register targets select the EC2 instance that you want to be registered as a target.

Target group should look like this:

Target groups (1/1) info

🔄

Actions ▼

Create target group

< 1 > ⌛

<input checked="" type="checkbox"/>	Name ▼	ARN ▼	Port ▼	Protocol ▼	Target type ▼	Load balancer ▼	VPC ID ▼
<input checked="" type="checkbox"/>	DevOps-gr11-TargetGroup	arn:aws:elasticloadbalanci...	80	HTTP	Instance	DevOps-gr11LB	vpc-006a73ed23fe2f972

Target group: DevOps-gr11-TargetGroup

×

Details

Targets

Monitoring

Health checks

Attributes

Tags

Details

📄

arn:aws:elasticloadbalancing:us-east-1:917837302529:targetgroup/DevOps-gr11-TargetGroup/2e0a4d7cfc5bfb26

Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC vpc-006a73ed23fe2f972 🔗
IP address type IPv4	Load balancer DevOps-gr11LB 🔗		

2 Total targets	<div>🟢 2</div> <div>Healthy</div> <div>0 Anomalous</div>	<div>🔴 0</div> <div>Unhealthy</div>	<div>🟡 0</div> <div>Unused</div>	<div>🟡 0</div> <div>Initial</div>	<div>🟡 0</div> <div>Draining</div>
--------------------	--	-------------------------------------	----------------------------------	-----------------------------------	------------------------------------

Now return to the creation of the load balancer and in the target group select the newly created target group.

Scroll down to the end and create a load balancer.

Next up we will create an Amazon machine image (AMI) from the EC2 instance.

An Amazon Machine Image (AMI) is a supported and maintained image provided by AWS that provides the information required to launch an instance.

Go to the EC2 instances, select the second created instance and in the actions menu select image and templates and the select create image.

Put the name and description of the image and create AMI

Once created your AMI should look like this:

AMI ID: ami-0a2819dd7c40a0773			
<div>DetailsPermissionsStorageTags</div>			
AMI ID ami-0a2819dd7c40a0773	Image type machine	Platform details Linux/UNIX	Root device type EBS
AMI name DevOps-gr11-EC2-3AMI	Owner account ID 917837302529	Architecture x86_64	Usage operation RunInstances
Root device name /dev/sda1	Status Available	Source 917837302529/DevOps-gr11-EC2-3AMI	Virtualization type hvm
Boot mode uefi-preferred	State reason -	Creation date Wed Jun 05 2024 20:46:56 GMT+0200 (Central European Summer Time)	Kernel ID -
Description AMI for ec2 instance	Product codes -	RAM disk ID -	Deprecation time -
Last launched time -	Block devices /dev/sda1=snap-0b53eae8e2b419b8d8:true:gp3 /dev/sdb=ephemeral0 /dev/sdc=ephemeral1		Deregistration protection Disabled

Now when we create AMI, we will create a Launch template.

You can use a launch template to store instance launch parameters so that you do not have to specify them every time you launch an instance. For example, you can create a launch template with the AMI ID, instance type, and network settings that you typically use to launch instances. When you launch an instance using the Amazon EC2 console, an AWS SDK, or a command line tool, you can specify the launch template instead of entering the parameters again.

Put the name of the template, for the AMI chose my amis, and select the AMI we have created just before.

For the instance type select the same as the instance type that already exists, that is t2.micro.

Key pair name should be vockey.

Select the security group we first created, that is attached to the other EC2 instances, and for user data paste the user data provided in UserDataScript-phase3.

Create a launch template.

It should look like this:

DevOps-gr11-LT (lt-0d23b6ea654018caf)

Launch template ID
lt-0d23b6ea654018caf

Launch template name
DevOps-gr11-LT

Default version
1

Owner
arn:aws:sts::917837302529:assumed-role/voclabs/user3144770:Kerim_Sabic

Details

Versions

Template tags

Launch template version details

Version
1 (Default)

Description
-

Date created
2024-06-05T19:07:10.000Z

Created by
arn:aws:sts::917837302529:assumed-role/voclabs/user3144770:Kerim_Sabic

Instance details

Storage

Resource tags

Network interfaces

Advanced details

AMI ID
ami-0a2819dd7c40a0773

Instance type
t2.micro

Availability Zone
-

Key pair name
vockey

Security groups
-

Security group IDs
sg-08fa909fd9ae1447

Finally create an auto scaling group.

An Auto Scaling group contains a collection of EC2 instances that are treated as a logical grouping for the purposes of automatic scaling and management.

Both maintaining the number of instances in an Auto Scaling group and automatic scaling are the core functionality of the Amazon EC2 Auto Scaling service.

The size of an Auto Scaling group depends on the number of instances that you set as the desired capacity.

You can adjust its size to meet demand, either manually or by using automatic scaling.

An Auto Scaling group starts by launching enough instances to meet its desired capacity.

It maintains this number of instances by performing periodic health checks on the instances in the group.

The Auto Scaling group continues to maintain a fixed number of instances even if an instance becomes unhealthy. If an instance becomes unhealthy, the group terminates the unhealthy instance and launches another instance to replace it.

Click on the Create auto scaling group. Put the name and for the launch template select the newly created template.

Configure the VPC to be the VPC we created and choose both of the public subnets we created.

For the load balancer select Attach to an existing load balancer and then choose the load balancer we created same with the target group.

For the health check period select 90 seconds

Next, for the desired capacity select 2, minimum 1 and maximum 2.

Don't select scaling policies and finally add tags.

Add key name and value add the name you want your insurance to have.

Finally create the auto scaling group.

Auto Scaling group: DevOps-gr11-ASG

Group details

Auto Scaling group name
DevOps-gr11-ASG

Desired capacity
2

Desired capacity type
Units (number of instances)

Amazon Resource Name (ARN)
arn:aws:autoscaling:us-east-1:917837302529:autoScalingGroup:90d7f698-ebb4-4b3b-8858-855d9b138cfcautoScalingGroup:DevOps-gr11-ASG

Date created
Wed Jun 05 2024 21:15:29 GMT+0200 (Central European Summer Time)

Minimum capacity
1

Status
-

Maximum capacity
2

Launch template

Edit

Launch template lt-Od23b6ea654018caf DevOps-gr11-LT	AMI ID ami-0a2819dd7c40a0773	Instance type t2.micro	Owner arn:aws:sts::917837302529:assumed-role/voclabs/user3144770=Kerim_Sabic
Version Default	Security groups -	Security group IDs sg-08fa909fd9ae1447	Create time Wed Jun 05 2024 21:07:10 GMT+0200 (Central European Summer Time)
Description -	Storage (volumes) -	Key pair name vockey	Request Spot Instances No

View details in the launch template console

Network

Edit

Availability Zones us-east-1a, us-east-1b	Subnet ID subnet-05b1429f75d573ef5, subnet-0490096a0cd1f2e26
--	---

Instance type requirements

Edit

Your Auto Scaling group adheres to the launch template for purchase option and instance type.

Load balancing

Edit

Load balancer target groups DevOps-gr11-TargetGroup	Classic Load Balancers -
--	-----------------------------

Tags (1)

Edit

Key	Value	Tag new instances
Name	Students-WebApp	Yes

We have successfully create highly available and scalable web application.

We can now access our web application through the application load balancer DNS name, and the load balancer will automatically distribute the requests to appropriate EC2 instances.

