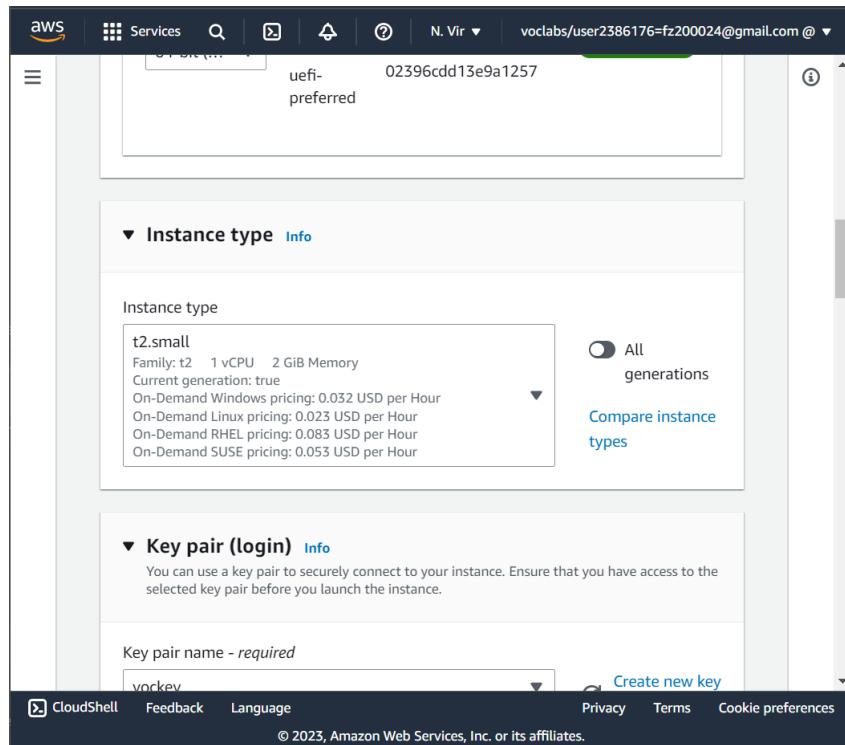
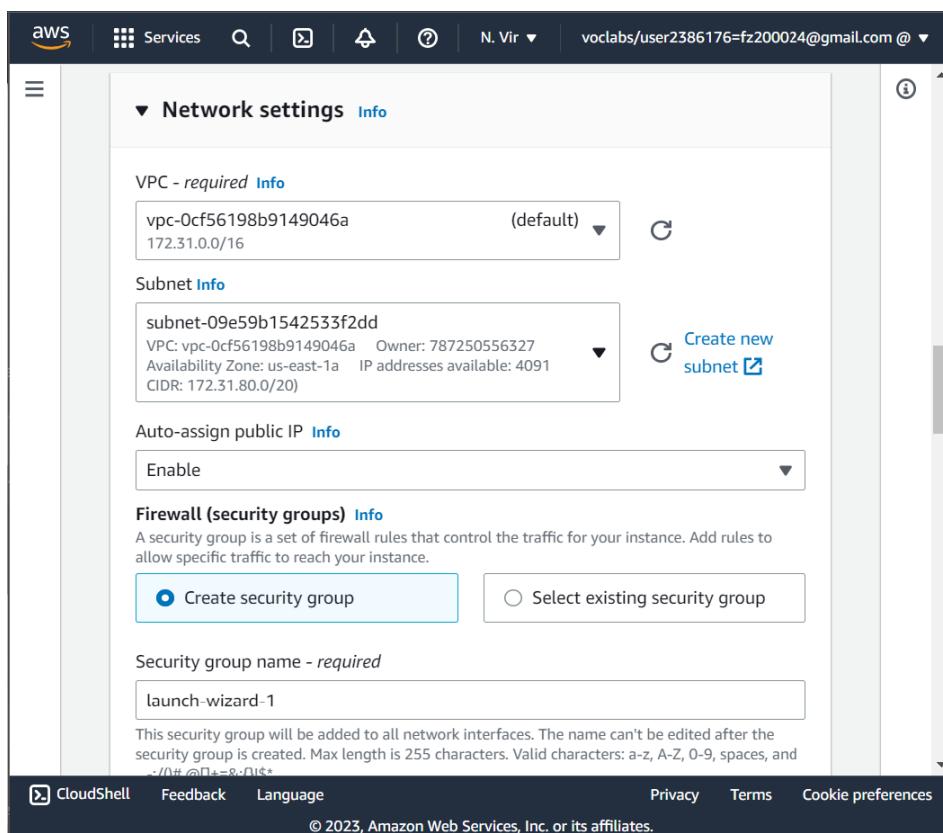


## **PART A:**

To begin the creation of the Virtual Machine (VM) with the given requirements, an instance of the EC2 server is created by the name of “Main Server” with the t2.small instance type having a single core with 2GBs of RAM.



This Main Server is assigned to the us-east-1a availability zone.



The security group rules are configured for making the server accessible through ports 80 and 443 respectively.

The screenshot shows the AWS Management Console interface for managing security group rules. The top navigation bar includes the AWS logo, Services, a search bar, and user information. The main content area is titled "Inbound security groups rules". It displays two rules:

- Security group rule 1 (TCP, 80, 0.0.0.0/0)**:
  - Type: HTTP
  - Protocol: TCP
  - Port range: 80
  - Source type: Anywhere
  - Description: e.g. SSH for admin desktop
- Security group rule 2 (TCP, 443, 0.0.0.0/0)**:
  - Type: HTTPS
  - Protocol: TCP
  - Port range: 443
  - Source type: Anywhere
  - Description: e.g. SSH for admin desktop

A warning message is present: "⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only." An "Add security group rule" button is available at the bottom left. A "Advanced network configuration" link is also visible.

In the user data text field, the following HTML code is entered for the server to be able to render an HTML page that displays the private IP address of the EC2 instance.

The screenshot shows the AWS Launch Instance wizard. The left pane contains fields for "Select" and "User data - optional". The "User data" field contains the following shell script:

```
#!/bin/bash
yum update -y
yum -y install httpd
systemctl enable httpd
systemctl start httpd
echo "This is server $(hostname -f)" > /var/www/html/index.html
```

A checkbox "User data has already been base64 encoded" is present. The right pane displays the "Summary" section with the following details:

- Number of instances: 1
- Software Image (AMI): Amazon Linux 2023 AMI 2023.0.2...read more  
ami-0889a44b331db0194
- Virtual server type (instance type): t2.small
- Firewall (security group): New security group
- Storage (volumes): 1 volume(s) - 8 GiB

Buttons include "Cancel", "Launch instance" (highlighted in orange), and "Review commands". The bottom navigation bar includes CloudShell, Feedback, Language, Privacy, Terms, and Cookie preferences.

The Main Server EC2 instance has been created.

Instance summary for i-0b10842cc3c6c6523 (Main Server) [Info](#)  
Updated less than a minute ago

Instance ID i-0b10842cc3c6c6523 (Main Server)	Public IPv4 address 3.88.10.247   <a href="#">open address</a>	Private IPv4 addresses 172.31.91.231
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-3-88-10-247.compute-1.amazonaws.com   <a href="#">open address</a>
Hostname type IP name: ip-172-31-91-231.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-91-231.ec2.internal	Elastic IP addresses -
Answer private resource DNS name -	Instance type t2.small	AWS Compute Optimizer finding <a href="#">Opt-in to AWS Compute Optimizer for recommendations.</a>   <a href="#">Learn more</a>
Auto-assigned IP address 3.88.10.247 [Public IP]	VPC ID vpc-0cf56198b9149046a	Auto Scaling Group name -
IAM Role -	Subnet ID subnet-09e59b1542533f2dd	

[Details](#) [Security](#) [Networking](#) [Storage](#) [Status checks](#) [Monitoring](#) [Tags](#)

Upon entering the public IP address into the search bar as highlighted above, the following HTML page is displayed:

← → C Not secure | 3.88.10.247

YouTube WhatsApp Banner Moodle C

This is server ip-172-31-91-231.ec2.internal

Now to create the RDS instance, the creation method will be “Easy Create” and will use the PostgreSQL configuration.

RDS > Create database

Create database

Choose a database creation method [Info](#)

Easy create  
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Standard create  
You set all of the configuration options, including ones for availability, security, backups, and maintenance.

Configuration

Engine type [Info](#)

Aurora (MySQL Compatible)

Aurora (PostgreSQL Compatible)

MySQL

MariaDB

PostgreSQL

Oracle

Microsoft SQL Server

PostgreSQL

PostgreSQL is a powerful, open-source object-relational database system with a strong reputation of reliability, stability, and correctness.

- High reliability and stability in a variety of workloads.
- Advanced features to perform in high-volume environments.
- Vibrant open-source community that releases new features multiple times per year.
- Supports multiple extensions that add even more functionality to the database.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.
- Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
- The most Oracle-compatible open-source database.

CloudShell Feedback Language © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The DB instance size is selected as Free Tier and is named “amazonrds”. It is also configured to autogenerate a password upon creation of the instance.

The screenshot shows the AWS RDS "Create DB Instance" wizard. In the "DB instance size" section, the "Free tier" option is selected. Other options shown include "Production" (db.r5g.xlarge) and "Dev/Test" (db.r5g.large). The "DB instance identifier" field contains "amazonrds". The "Master username" is set to "postgres". The "Auto generate a password" checkbox is checked. A note indicates that Amazon RDS can generate a password or you can specify your own. Below this, a section titled "▶ Set up EC2 connection - optional" provides instructions for setting up an EC2 connection after database creation.

The RDS instance has been created, with the following configurations:

The screenshot shows the AWS RDS "Databases" page with the "amazonrds" instance selected. The "Summary" tab is active, displaying basic information: DB identifier (amazonrds), Role (Instance), GPU usage (5.95%), Current activity (0 Connections), Status (Available), Engine (PostgreSQL), Class (db.t3.micro), and Region & AZ (us-east-1b). Below the summary, the "Configuration" tab is selected, showing detailed configuration settings. These include:

- Configuration**: DB instance ID (amazonrds), Engine version (14.6), DB name (\*), License model (PostgreSQL License), Option groups (default\_postgres-14), Amazon Resource Name (ARN) (arn:aws:rds:us-east-1:707250556327:db:amazonrds), Resource ID (db-36WKC0UJSHHGV0BVGUKQSOONQ), Created time (May 02, 2023, 15:42 (UTC+04:00)), DB instance parameter group (default\_postgres-14), and Deletion protection (Disabled).
- Instance class**: Instance class (db.t3.micro), vCPU (2), RAM (1 GB), and Availability (Master username: postgres, Master password: \*\*\*\*\*).
- Storage**: Storage type (General Purpose SSD (gp2)), Storage (20 GiB), Provisioned IOPS (-), Storage throughput (-), Storage autoscaling (Enabled), and Maximum storage threshold (1000 GiB).
- Performance Insights**: Performance Insights enabled (Turned off).

After this, two DynamoDB tables are created. The first one is created with the name "Customer\_Information". The "ID" is used as partition key, which is of type Number, and the "Name" is used as sort key, which is of type String.

The screenshot shows the 'Create table' wizard in the AWS DynamoDB console. The 'Table details' step is active. In the 'Table name' section, 'Customer\_Information' is entered into the input field. Below it, the description states: 'Between 3 and 255 characters, containing only letters, numbers, underscores (\_), hyphens (-), and periods (.).'. In the 'Partition key' section, 'ID' is selected as the primary key type, which is a Number. The description below says: 'The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.' In the 'Sort key - optional' section, 'Name' is selected as the secondary key type, which is a String. The description below says: 'You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.' At the bottom, there is a 'Table settings' section.

Table settings are customized to have the DynamoDB Standard Table Class, and the table is created.

The screenshot shows the 'Create table' wizard continuing through the steps. In the 'Table settings' section, the 'Customize settings' option is selected. In the 'Table class' section, the 'DynamoDB Standard' option is selected. The description for 'DynamoDB Standard' states: 'The default general-purpose table class. Recommended for the vast majority of tables that store frequently accessed data, with throughput (reads and writes) as the dominant table cost.' The 'Capacity calculator' button is visible at the bottom of this section. The footer of the page includes links for CloudShell, Feedback, Language, Privacy, Terms, and Cookie preferences, along with a copyright notice for 2023, Amazon Web Services, Inc. or its affiliates.

The same steps are followed for the creation of the second table i.e., the “Customer\_Purchases” table, with the “Product\_Name” of type String used as partition key, and “ID” of type Number used as sort key. Following is the result of the creation of the two DynamoDB tables:

The screenshot shows the AWS DynamoDB console interface. On the left, there's a sidebar with navigation links like 'Dashboard', 'Tables', 'Update settings', etc. The main area displays a success message: 'The Customer\_Purchases table was created successfully.' Below this, the 'Tables' list shows two entries:

Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode
Customer_Information	Active	ID (N)	Name (S)	0	Off	Provisioned with auto scaling (1)	Provisioned with auto scaling (1)
Customer_Purchases	Active	Product_Name (S)	ID (N)	0	Off	Provisioned with auto scaling (1)	Provisioned with auto scaling (1)

To store the customer data in the DynamoDB tables according to the given schema, the respective items in each database are entered individually. For the Customer\_Information table, the items are entered as follows:

The screenshot shows the 'Create item' dialog for the 'Customer\_Information' table. At the top, there are tabs for 'Form' (which is selected) and 'JSON view'. The main area is titled 'Attributes' and contains two rows of data:

Attribute name	Value	Type
ID - Partition key	1	Number
Name - Sort key	Ahmed	String

At the bottom right, there are 'Cancel' and 'Create item' buttons.

The screenshot shows the AWS DynamoDB console interface. On the left, there's a navigation sidebar with links like Dashboard, Tables, Update settings, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Reserved capacity, and Settings. Below that is a section for DAX with Clusters, Subnet groups, Parameter groups, and Events. The main area is titled "DynamoDB" and shows a table named "Customer\_Information". The table has four items returned:

ID	Name
4	Mulham
3	Ali
2	Dana
1	Ahmed

At the bottom right of the table area, there's a message: "Completed. Read capacity units consumed: 0.5".

The same is done to add items and store items in the Customer\_Purchases table as follows:

This screenshot shows the AWS DynamoDB console interface, similar to the previous one but for a different table. The navigation sidebar and overall layout are identical. The main area is titled "DynamoDB" and shows a table named "Customer\_Purchases". The table has four items returned:

Product_Name	ID
Server	4
Tablet	3
Laptop	2
iPhone	1

At the bottom right of the table area, there's a message: "Completed. Read capacity units consumed: 0.5".

Some sample queries are run on the tables with the following results:

e.g. Item from table Customer\_Information with ID 4 is retrieved:

The screenshot shows the AWS DynamoDB console interface. On the left, there's a sidebar with options like Dashboard, Tables, Update settings, Explore items (which is selected), PartiQL editor, Backups, Exports to S3, Imports from S3, Reserved capacity, and Settings. Below that is a DAX section with Clusters, Subnet groups, Parameter groups, and Events.

The main area is titled "Table - Customer\_Information". It has a dropdown for "Select attribute projection" set to "All attributes". Under "ID (Partition key)", the value "4" is entered. There's a "Name (Sort key)" section with "Equal to" selected and "Mulham" entered. A "Run" button is at the bottom.

A green message box says "Completed. Read capacity units consumed: 0.5". Below it, a table titled "Items returned (1)" shows one item: ID 4 with Name Mulham.

At the bottom, there are links for CloudShell, Feedback, Language, and a footer with copyright information and links for Privacy, Terms, and Cookie preferences.

e.g. Item from Customer\_Information table with Product\_Name “Laptop” is retrieved:

The screenshot shows the AWS DynamoDB console interface, similar to the previous one but for a different table. The sidebar and top navigation are identical.

The main area is titled "Table - Customer\_Purchases". It has a dropdown for "Select attribute projection" set to "All attributes". Under "Product\_Name (Partition key)", the value "Laptop" is entered. There's a "ID (Sort key)" section with "Equal to" selected and "2" entered. A "Run" button is at the bottom.

A green message box says "Completed. Read capacity units consumed: 0.5". Below it, a table titled "Items returned (1)" shows one item: Product\_Name Laptop with ID 2.

At the bottom, there are links for CloudShell, Feedback, Language, and a footer with copyright information and links for Privacy, Terms, and Cookie preferences.

## QUESTION: Can you perform a table join in DynamoDB?

**ANSWER: NO**, a table join cannot be performed in DynamoDB. DynamoDB is a NoSQL database and unlike the traditional relational databases, there are no relations between tables and it does not support table join operations. It has standalone capabilities and also, no such querying option is made available on the Amazon Web Services Platform.

## PART B:

Now to deploy web applications and implement load balancing, web server environments are configured using Elastic Beanstalk to deploy the two web applications. The first one is configured for the “Docker Web App” with the “Web server environment” tier.

The screenshot shows the 'Configure environment' step of the AWS Elastic Beanstalk setup. On the left, a sidebar lists steps: Step 1 (Configure environment), Step 2 (Configure service access), Step 3 - optional (Set up networking, database, and tags), Step 4 - optional (Configure instance traffic and scaling), Step 5 - optional (Configure updates, monitoring, and logging), and Step 6 (Review). The main area is titled 'Configure environment' and contains three sections: 'Environment tier' (selected 'Web server environment'), 'Application information' (Application name: Docker Web App), and 'Environment information' (Environment name: DockerWebApp-env).

The platform is a Docker platform of type Managed Platform and sample application (shown clearly in upcoming review screenshot).

The screenshot shows the 'Environment information' step of the AWS Elastic Beanstalk setup. It includes fields for 'Environment name' (DockerWebApp-env), 'Domain' (Leave blank for autogenerated value .us-east-1.elasticbeanstalk.com), 'Check availability' button, 'Environment description' (empty), 'Platform type' (selected 'Managed platform' - Platforms published and maintained by Amazon Elastic Beanstalk), 'Platform' (Docker), and 'Platform branch' (Docker running on 64bit Amazon Linux 2).

The service access for the Docker app is configured using the existing service role of “LabRole”, with the “vockey” key pair, and EC2 instance profile of “LabInstanceProfile”.

The screenshot shows the 'Configure service access' step of the AWS Elastic Beanstalk environment setup. The 'Service role' dropdown is set to 'LabRole'. The 'EC2 key pair' dropdown is set to 'vockey'. The 'EC2 instance profile' dropdown is set to 'LabInstanceProfile'. At the bottom, there are 'Cancel', 'Skip to review', 'Previous', and 'Next' buttons.

The environment for the Docker Web App has been configured and created.

The screenshot shows the 'Review' step of the AWS Elastic Beanstalk environment setup. It displays the environment information and service access details. Environment information includes tier (Web server environment), name (Docker Web App), and platform (Docker). Service access details show the service role (arn:aws:iam::787250565327:role/Lab), EC2 key pair (vockey), and EC2 instance profile (LabInstanceProfile).

The screenshot shows the AWS Elastic Beanstalk environment overview for 'DockerWebApp-env'. The environment is successfully launched. The 'Events' section shows three recent events:

- May 4, 2023 00:47:11 (UTC+4) - INFO: Environment health has transitioned from Pending to Ok. Initialization completed 22 seconds ago and took 3 minutes.
- May 4, 2023 00:46:48 (UTC+4) - INFO: Successfully launched environment: DockerWebApp-env
- May 4, 2023 00:46:47 (UTC+4) - INFO: Application available at DockerWebApp-env.eba-e8dpw7x6.us-east-1.elasticbeanstalk.com.

The Docker web application is then uploaded and deployed.

The screenshot shows the AWS Elastic Beanstalk console with the 'DockerWebApp-env' environment selected. A modal window titled 'Upload and deploy' is open, prompting the user to choose a file. A file named 'docker.zip' has been selected. The 'Version label' field contains 'Docker Web App-version-1'. At the bottom right of the modal, there are 'Cancel' and 'Deploy' buttons, with 'Deploy' being the active button.

The screenshot shows the AWS Elastic Beanstalk console with the 'DockerWebApp-env' environment selected. At the top, two green notifications appear: one stating 'Environment update successfully completed.' and another stating 'Successfully uploaded file docker.zip to S3, created application version and started deployment with new application version'. The main interface shows the 'Environment overview' with the 'Running version' listed as 'Docker Web App-version-1'. The left sidebar shows the application configuration and recent environments.

The given domain link is copied and pasted into the search bar to check the working of the deployed web app.

The screenshot shows a web browser displaying the deployed Docker application. The main content area features a large 'Congratulations!' message. Below it, a note states: 'Your Docker Container is now running in Elastic Beanstalk on your own dedicated environment in the AWS Cloud.' Further down, it says: 'This environment is launched with Elastic Beanstalk Docker Platform'. To the right of the main message, there are three sections: 'Video Tutorials' (links to YouTube), 'Sample Apps' (links to GitHub), and 'Documentation' (links to AWS Elastic Beanstalk documentation).

Similar set of steps are followed to configure the web environment and deploy the Ruby web application with the name of “Ruby Web App”.

**Configure environment**

**Environment tier** Info  
Amazon Elastic Beanstalk has two types of environment tiers to support different types of web applications.

**Web server environment**  
Run a website, web application, or web API that serves HTTP requests. [Learn more](#)

**Worker environment**  
Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. [Learn more](#)

**Application information** Info

**Application name**  
Ruby Web App  
Maximum length of 100 characters.

**Application tags (optional)**

**Environment information** Info  
Choose the name, subdomain and description for your environment. These cannot be changed later.

**Environment name**  
RubyWebApp-env  
Must be from 4 to 40 characters in length. The name can contain only letters, numbers, and hyphens. It can't start or end with a hyphen. This name must be unique within a region in your account.

**Platform** Info

**Platform type**

**Managed platform**  
Platforms published and maintained by Amazon Elastic Beanstalk. [Learn more](#)

**Custom platform**  
Platforms created and owned by you. This option is unavailable if you have no platforms.

**Platform**  
Ruby

**Platform branch**  
Ruby 3.0 running on 64bit Amazon Linux 2

**Platform version**  
3.6.7 (Recommended)

**Application code** Info

**Sample application**

**Existing version**  
Application versions that you have uploaded.

**Upload your code**  
Upload a source bundle from your computer or copy one from Amazon S3.

**Service access**  
IAM roles, assumed by Elastic Beanstalk as a service role, and EC2 instance profiles allow Elastic Beanstalk to create and manage your environment. Both the IAM role and instance profile must be attached to IAM managed policies that contain the required permissions. [Learn more](#)

**Service role**

Create and use new service role

Use an existing service role

**Existing service roles**  
Choose an existing IAM role for Elastic Beanstalk to assume as a service role. The existing IAM role must have the required IAM managed policies.

**LabRole**

**EC2 key pair**  
Select an EC2 key pair to securely log in to your EC2 instances. [Learn more](#)

vockey

**EC2 instance profile**  
Choose an IAM instance profile with managed policies that allow your EC2 instances to perform required operations.

LabinstanceProfile

[View permission details](#)

[Cancel](#) [Skip to review](#) [Previous](#) [Next](#)

Screenshot of the AWS Elastic Beanstalk Step 1: Configure environment review screen.

**Review** Info

**Step 1: Configure environment**

**Environment information**

Environment tier	Application name
Web server environment	Ruby Web App
Environment name	Application code
RubyWebApp-env	Sample application
Platform	arn:aws:elasticbeanstalk:us-east-1::platform/Ruby 3.0 running on 64bit Amazon Linux 2/3.6.7

**Step 2: Configure service access**

**Service access** Info

Configure the service role and EC2 instance profile that Elastic Beanstalk uses to manage your environment. Choose an EC2 key pair to securely log in to your EC2 instances.

Service role	EC2 key pair	EC2 instance profile
arn:aws:iam::787250556327:role/Lab	vockey	LabInstanceProfile

Screenshot of the AWS Elastic Beanstalk Environment successfully launched screen.

**Elastic Beanstalk**

**RubyWebApp-env** Info

**Environment overview**

Health	Environment ID
OK	e-bmblbmgbmu
Domain	Application name
RubyWebApp-env.eba-epnp8xf.us-east-1.elasticbeanstalk.com	Ruby Web App

**Events** Info

Events (11) Info

Time	Type	Details
May 4, 2023 01:24:52 (UTC+4)	INFO	Successfully launched environment: RubyWebApp-env
May 4, 2023 01:24:51 (UTC+4)	INFO	Application available at RubyWebApp-env.eba-epnp8xf.us-east-1.elasticbeanstalk.com.
May 4, 2023 01:24:50 (UTC+4)	INFO	Environment health has transitioned from Pending to OK. Initialization completed 12 seconds ago and took 2 minutes.

**Platform**

Platform: Ruby 3.0 running on 64bit Amazon Linux 2/3.6.7

Running version: -

Screenshot of the AWS Elastic Beanstalk Upload and deploy dialog box.

**Upload and deploy**

To deploy a previous version, go to the Application versions page

Upload application: Choose file

File name: ruby.zip

Version label: Ruby Web App-version-1

Current number of EC2 instances: 0

Events (11) Info

Time Type Details

May 4, 2023 01:24:52 (UTC+4) INFO Successfully launched environment: RubyWebApp-env

May 4, 2023 01:24:51 (UTC+4) INFO Application available at RubyWebApp-env.eba-epnp8xf.us-east-1.elasticbeanstalk.com.

May 4, 2023 01:24:50 (UTC+4) INFO Environment health has transitioned from Pending to OK. Initialization completed 12 seconds ago and took 2 minutes.

**Platform**

Platform: Ruby 3.0 running on 64bit Amazon Linux 2/3.6.7

Running version: -

The given domain link is copied and pasted into the search bar to check the working of the deployed web app.

The given domain link is copied and pasted into the search bar to check the working of the deployed web app.

What's Next?

- [AWS Elastic Beanstalk overview](#)
- [AWS Elastic Beanstalk concepts](#)
- [Deploy a Ruby on Rails Application to AWS Elastic Beanstalk](#)
- [Deploy a Sinatra Application to AWS Elastic Beanstalk](#)
- [Customizing and Configuring a Ruby Container](#)
- [Working with Logs](#)

As a result of deploying the two web applications using Elastic Beanstalk, two new EC2 instances are also created with them:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
DockerWebApp-env	i-0d9d75a426b8efd74	Running	t3.micro	2/2 checks passed	No alarms	us-east-1b
Main Server	i-0b10842cc3c6c6523	Running	t2.small	2/2 checks passed	No alarms	us-east-1a
RubyWebApp-env	i-01c2bd33795f5262c	Running	t3.micro	2/2 checks passed	No alarms	us-east-1a

Now a load balancer of name “ElasticLoadBalancer” is created to load balance between the two deployed web applications. An application load balancer is selected to fulfil this purpose.

The screenshot shows the AWS Services console with the URL [https://aws.amazon.com/console/home?region=us-east-1#/services/compute/load-balancing/load-balancers](#). The page title is "Select load balancer type". It compares three types of load balancers:

- Application Load Balancer**: Handles HTTP and HTTPS traffic, operating at the request level. It uses advanced routing features like target groups and listeners.
- Network Load Balancer**: Handles TCP, UDP, and TLS traffic, operating at the transport layer. It supports VPC and centralized certificate deployment.
- Gateway Load Balancer**: Handles traffic from third-party virtual appliances supporting GENEVE. It focuses on security, compliance, and policy controls.

Each section includes a diagram and a brief description of its use case.

The screenshot shows the AWS Services console with the URL [https://aws.amazon.com/console/home?region=us-east-1#/services/compute/load-balancing/load-balancers/create-application-load-balancer](#). The page title is "Create Application Load Balancer". It provides instructions on how the load balancer works and allows basic configuration:

- How Elastic Load Balancing works**: A summary of how the load balancer distributes traffic across targets.
- Basic configuration**:
  - Load balancer name**: Name must be unique within your AWS account and can't be changed after the load balancer is created. The input field contains "ElasticLoadBalancer".
  - Scheme**: Options are "Internet-facing" (selected) and "Internal".
  - IP address type**: Options are "IPv4" (selected) and "Dualstack".

The security groups are selected as the ones that were created with the creation of the web applications.

The screenshot shows the AWS Services console with the URL [https://aws.amazon.com/console/home?region=us-east-1#/services/compute/load-balancing/load-balancers/security-groups](#). The page title is "Security groups". It lists existing security groups:

Security group	Description	Action
awseb-e-bmibmpbmy-stack-AWSEBSecurityGroup-1DQKXGIGOOWH	sg-098cc5d450ce2dc69	X
awseb-e-sfspmjwbtbs-stack-AWSEBSecurityGroup-1UH4Q58HG6WMS	sg-08784c9e3ac194edb	X

The required subnets i.e. us-east-1a and us-east-1b are specified for load balancing.

Network mapping info  
For internet-facing load balancers, the IPv4 addresses of the nodes are assigned by AWS. For internal load balancers, the IPv4 addresses are assigned from the subnet CIDR.

VPC vpc-0cf56198b9149046a IP address type IPv4

Mappings info Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.

us-east-1a (use1-az2)

Subnet subnet-09e59b154253f2dd

IPv4 settings Assigned by AWS

us-east-1b (use1-az4)

Subnet subnet-0d3b48bddf3fd9684

IPv4 settings Assigned by AWS

us-east-1c (use1-az6)

A target group by the name of “WebApps” is created to be used for the load balancer. HTTP is chosen as the protocol for target health check with the health check path as “/index.html”.

WebApps A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol Port HTTP : 80

VPC Select the VPC with the instances that you want to include in the target group.

vpc-0cf56198b9149046a IPv4: 172.31.0.0/16

Protocol version

**HTTP1** Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.

**HTTP2** Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.

**gRPC** Send requests to targets using gRPC. Supported when the request protocol is gRPC.

Health checks The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.

Health check protocol HTTP

Health check path Use the default path of “/” to ping the root, or specify a custom path if preferred. /index.html Up to 1024 characters allowed.

The two instances for the deployed web applications are registered in the target group.

Specify group details Step 2 Register targets This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.

Available instances (3)

Instance ID	Name	State	Security groups	Zone	Subnet ID
i-09ef5a420be8fd74	DockerWebApp-env	Running	awss-e-5shpnjrb-stack-AWSE5SecurityGroup-1H42Q8H0GWHS	us-east-1b	subnet-0d3b48bddf3fd9684
i-0b10842cc3c68523	Main Server	Running	Launch-wizard-1	us-east-1a	subnet-09e59b154253f2dd
i-01c2bd53795f5262c	RubyWebApp-env	Running	awss-e-6mbmpbmbyu-stack-AWSE5SecurityGroup-1DQCKXGIG0OWH	us-east-1a	subnet-09e59b154253f2dd

Selected Ports for routing traffic to the selected instances. 80 1-65339 (separate multiple ports with commas)

Include as pending below 2 selections are now pending below. Include more or register targets when ready.

Review targets Targets (2)

Remove	Health status	Instance ID	Name	Port	State	Security groups	Zone	Subnet ID
X	Pending	i-01c2bd53795f5262c	RubyWebApp-env	80	Running	awss-e-6mbmpbmbyu-stack-AWSE5SecurityGroup-1DQCKXGIG0OWH	us-east-1a	subnet-09e59b154253f2dd
X	Running	i-0b10842cc3c68523	DockerWebApp-env	80	Running	awss-e-5shpnjrb-stack-AWSE5SecurityGroup-1H42Q8H0GWHS	us-east-1b	subnet-0d3b48bddf3fd9684

The “WebApps” target group has been created.

The screenshot shows the AWS EC2 Target groups page. On the left, there's a navigation sidebar with options like Instances, Images, and Network & Security. The main area displays the 'WebApps' target group details. It shows 2 healthy targets (RubyWebApp-env and DockerWebApp-env) and 0 unhealthy targets. Below this is a table titled 'Registered targets (2)' listing the two instances. At the bottom, there are tabs for Targets, Monitoring, Health checks, Attributes, and Tags.

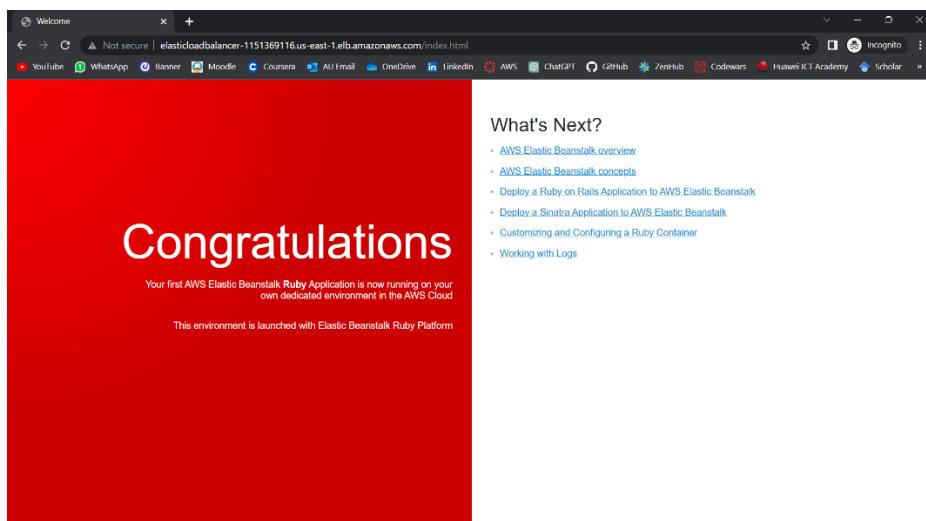
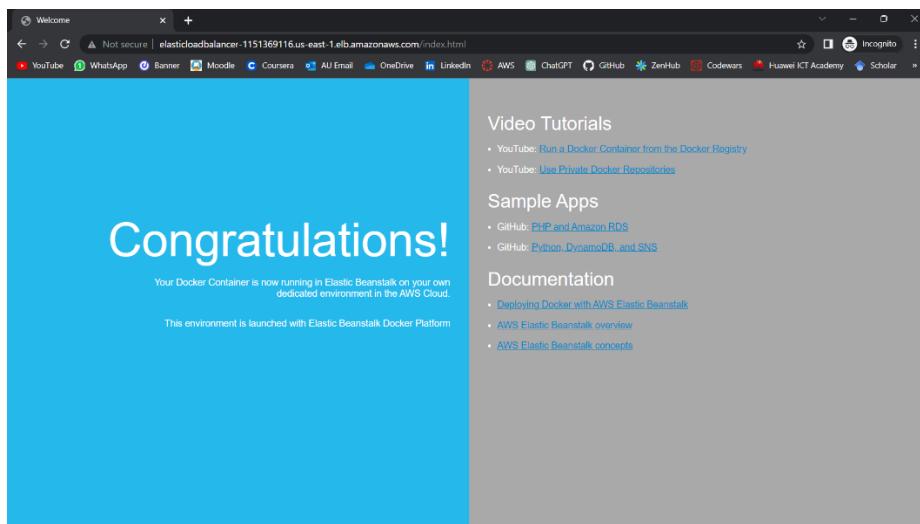
The WebApps target group is selected for the load balancer. (I forgot to screenshot this step at the time of actually adding the target group, so I added one screenshot for demonstration below, hence the target group in the dropdown list is greyed out).

The screenshot shows the AWS Load Balancers Listener and routing configuration page. It displays a 'Listener HTTP:80' configuration. The 'Default action' dropdown is set to 'Forward to' and is currently empty. A dropdown menu is open, showing 'Select a target group' and 'Create target'. Other options like 'In use' and 'WebApps' are also visible. There are tabs for 'Protocol', 'Port', and 'Info'.

The load balancer has been created after making the required configurations.

The screenshot shows the AWS EC2 Load balancers page. It lists a single load balancer named 'ElasticLoadBalancer'. The table includes columns for Name, DNS name, State, VPC ID, Availability Zones, Type, and Date. The 'Actions' button is highlighted in orange.

The DNS name link is copied and pasted into the search bar to check the working of the load balancer. Upon refreshing, the screen shows the webpage changing between the docker web app and the ruby web app.



The required cloud architecture has been designed and completed.