

# AWS Architecture for Cloud-Based PHP Application

## 1. Introduction

This document describes the AWS solution architecture for hosting a **PHP-based dynamic web application** using AWS managed services.

The goal is to design a **secure, scalable, highly available, and cost-effective** cloud environment using a **Load Balancer for high availability**.

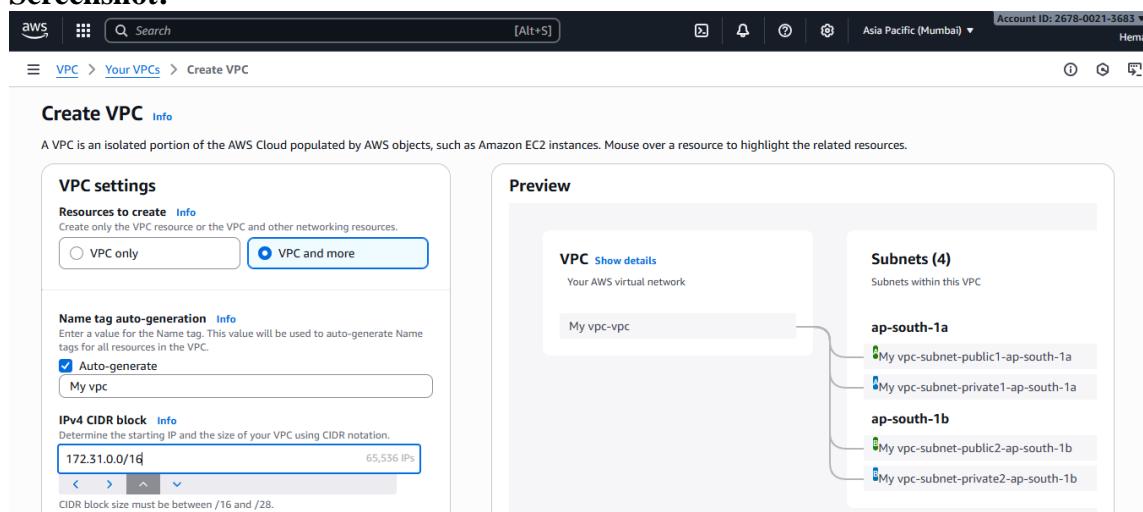
---

### Create a Secure VPC

#### Action:

1. Go to **VPC** in AWS.
2. Click **Create VPC → VPC and more**.
3. Configure:
  - o **VPC CIDR:** 172.31.0.0/16
  - o **2 Public Subnets:** For load balancer
  - o **2 Private Subnets:** For EC2 and RDS
4. Create an **Internet Gateway (IGW)** and attach it to the VPC.
5. Create a **NAT Gateway** in a public subnet to allow private subnets internet access.
6. Set up **Route Tables**:
  - o Public route table → IGW
  - o Private route table → NAT Gateway
7. Enable **DNS hostnames** and **DNS resolution** in the VPC settings.

#### Screenshot:



aws Search [Alt+S] Account ID: 2678-0021-3683 Hema

VPC > Your VPCs > Create VPC

**Number of public subnets** [Info](#)  
The number of public subnets to add to your VPC. Use public subnets for web applications that need to be publicly accessible over the internet.  
0 **2**

**Number of private subnets** [Info](#)  
The number of private subnets to add to your VPC. Use private subnets to secure backend resources that don't need public access.  
0 **2** **4**

**Customize subnets CIDR blocks**

**NAT gateways (\$)** [Info](#)  
Choose the number of Availability Zones (AZs) in which to create NAT gateways. Note that there is a charge for each NAT gateway.  
**None** **In 1 AZ** **1 per AZ**

**VPC endpoints** [Info](#)  
Endpoints can help reduce NAT gateway charges and improve security by accessing S3 directly from the VPC. By default, full access policy is used. You can customize this policy at any time.  
**None** **S3 Gateway**

**Preview**

**VPC** [Show details](#)  
Your AWS virtual network  
My vpc-vpc

**Subnets (4)**  
Subnets within this VPC

- ap-south-1a**
  - My vpc-subnet-public1-ap-south-1a
  - My vpc-subnet-private1-ap-south-1a
- ap-south-1b**
  - My vpc-subnet-public2-ap-south-1b
  - My vpc-subnet-private2-ap-south-1b

aws Search [Alt+S] Account ID: 2678-0021-3683 Hema

VPC > Your VPCs

**Your VPCs (1/2) Info** Last updated 2 minutes ago

Name	VPC ID	State	Block Public...	IPv4 CIDR
vpc-079af82f0aa441fc0	vpc-09ad80535324ce68a	Available	Off	172.31.0.0/16
<b>My vpc-vpc</b>	<b>vpc-09ad80535324ce68a</b>	<b>Available</b>	<b>Off</b>	<b>172.31.0.0/16</b>

**vpc-09ad80535324ce68a / My vpc-vpc**

**Details**

VPC ID vpc-09ad80535324ce68a	State <b>Available</b>	Block Public Access Off	DNS hostnames Enabled
DNS resolution Enabled	Tenancy default	DHCP option set dopt-081105d438cf35ae1	Main route table rtb-0cdcc19b41017f6d33
Main network ACL	Default VPC	IPv4 CIDR	IPv6 none

aws Search [Alt+S] Account ID: 2678-0021-3683 Hema

VPC > Subnets

**Subnets (6) Info** Last updated 2 minutes ago

Name	Subnet ID	State	VPC	Block Publi...
My vpc-subnet-private1-ap-south-1a	subnet-03395084c139459bd	Available	vpc-09ad80535324ce68a   My v...	Off
My vpc-subnet-public2-ap-south-1b	subnet-07ff23a8c41eedc5	Available	vpc-09ad80535324ce68a   My v...	Off
My vpc-subnet-private2-ap-south-1b	subnet-0760f4aca77476e34	Available	vpc-09ad80535324ce68a   My v...	Off
My vpc-subnet-public1-ap-south-1a	subnet-0ed674589d45399e2	Available	vpc-09ad80535324ce68a   My v...	Off

**Select a subnet**

aws Search [Alt+S] Account ID: 2678-0021-3683 Hema

VPC > Route tables

Route tables (1/5) Info Last updated 7 minutes ago Actions Create route table

Name	Route table ID	Explicit subnet assoc...	Edge associations	Main	VPC
My vpc-rtb-private2-ap-south-1b	rtb-045025a4f05dfcd93	subnet-0760f4aca77476...	-	No	vpc
-	rtb-0aad67375b9263337	-	-	Yes	vpc
My vpc-rtb-private1-ap-south-1a	rtb-09ed57af73b6bc724	subnet-03395084c13945...	-	No	vpc
My vpc-rtb-public	rtb-004cea4f361aa7857	2 subnets	-	No	vpc

rtb-004cea4f361aa7857 / My vpc-rtb-public

Details Routes Subnet associations Edge associations Route propagation Tags

Routes (2) Both Edit routes

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	igw-099fd6280daeece48	Active	No	Create Route
172.31.0.0/16	local	Active	No	Create Route Table

aws Search [Alt+S] Account ID: 2678-0021-3683 Hema

VPC > Route tables

Route tables (1/5) Info Last updated 7 minutes ago Actions Create route table

Name	Route table ID	Explicit subnet assoc...	Edge associations	Main	VPC
My vpc-rtb-private2-ap-south-1b	rtb-045025a4f05dfcd93	subnet-0760f4aca77476...	-	No	vpc
-	rtb-0aad67375b9263337	-	-	Yes	vpc
My vpc-rtb-private1-ap-south-1a	rtb-09ed57af73b6bc724	subnet-03395084c13945...	-	No	vpc
My vpc-rtb-public	rtb-004cea4f361aa7857	2 subnets	-	No	vpc

rtb-004cea4f361aa7857 / My vpc-rtb-public

Details Routes Subnet associations Edge associations Route propagation Tags

Routes (2) Both Edit routes

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	igw-099fd6280daeece48	Active	No	Create Route
172.31.0.0/16	local	Active	No	Create Route Table

aws Search [Alt+S] Account ID: 2678-0021-3683 Hema

VPC > Internet gateways

Internet gateways (1/2) Info Actions Create internet gateway

Name	Internet gateway ID	State	VPC ID
My vpc-igw	igw-099fd6280daeece48	Attached	vpc-09ad80535324ce68a   My vpc-vpc
-	igw-0abf88bb4fce14012	Attached	vpc-079fd82f0a441fc0

igw-099fd6280daeece48 / My vpc-igw

Details Tags

Details

Internet gateway ID igw-099fd6280daeece48	State Attached	VPC ID vpc-09ad80535324ce68a   My vpc-vpc	Owner 267800213683
--	-------------------	--	-----------------------

The screenshot shows the AWS VPC NAT gateways console. On the left, there's a navigation sidebar with sections like Virtual private cloud, Security, and NAT gateways. The main area displays a table titled 'NAT gateways (1/1)'. The table has columns for Name, NAT gateway ID, Connectivity..., State, State message, and Primary public IP... The single entry is 'My vpc-nat-public1...' with the ID 'nat-019b0169a12db1d09'. It shows 'Public' under Connectivity, 'Available' under State, and '35.154.242.201' under Primary public IP. Below the table, there's a detailed view for 'nat-019b0169a12db1d09 / My vpc-nat-public1-ap-south-1a' with tabs for Details, Secondary IPv4 addresses, Monitoring, and Tags. The 'Details' tab shows various configuration details.

## Create Security Groups

### Action:

1. Create the Security Groups:
  - o Database security group: Allow inbound MySQL (3306) from EC2 SG only.
2. Apply the security groups to the respective resources.

### Screenshot:

The screenshot shows the AWS Security Groups console. The top navigation bar includes 'VPC > Security Groups > Create security group'. The main area is titled 'Create security group' with a sub-section 'Basic details'. It asks for a 'Security group name' (set to 'Database security group') and a 'Description' (also set to 'Database security group'). Under 'VPC Info', it shows 'vpc-09ad80535324ce68a (My vpc-vpc)'. Below this, there's an 'Inbound rules' section which states 'This security group has no inbound rules.' and a 'Add rule' button.

## Create Amazon RDS (Database)

### Action:

1. Go to **RDS** → **Create Database**.
2. Choose **Standard Create** → **MySQL**
3. Enable **Multi-AZ Deployment** for high availability.
4. Under **Connectivity**:
  - o Select your custom **VPC**
  - o Create a **DB Subnet Group** with private subnets
  - o Attach the **RDS Security Group**
5. Note the **DB endpoint** and credentials for your application.

### Screenshot:

The image contains two screenshots of the AWS Aurora and RDS console.

**Screenshot 1: Subnet groups**

This screenshot shows the "Subnet groups" page. A green success message at the top says "Successfully created My db subnet. View subnet group". The table below shows one entry:

Name	Description	Status	VPC
my db subnet	My db subnet	Complete	vpc-09ad80535324ce68a

**Screenshot 2: Databases**

This screenshot shows the "Databases" page. A green success message at the top says "Successfully created database database-1". It includes a note about generating a master password and a link to "View connection details". The table below shows one entry:

DB identifier	Status	Role	Engine	Region ...	Size
database-1	Available	Instance	MySQL Co...	us-east-1a	db.t3.mic

---

## Store Secrets in AWS Systems Manager Parameter Store

### Action:

1. Navigate to **Systems Manager → Parameter Store**.
2. Create **SecureString Parameters** for:
  - o /app/db/username
  - o /app/db/password
  - o /app/db/host
3. Use **AWS KMS default key** for encryption.

### Screenshot:

The screenshot shows two consecutive screenshots of the AWS Systems Manager Parameter Store interface.

**Top Screenshot (Parameter Type Selection):**

- The "Type" section is open, showing three options:
  - String: Any string value.
  - StringList: Separate strings using commas.
  - SecureString: Encrypt sensitive data using KMS keys from your account or another account.
- The "KMS key source" dropdown is set to "My current account".
- The "KMS Key ID" dropdown is set to "alias/aws/ssm".
- A tooltip message is displayed: "You have selected the default AWS managed key. AWS managed keys cannot be shared with other AWS accounts, and all users in this AWS account and Region have access to the key. Use a customer managed key for sharing and access control." with a "Learn more" link.

**Bottom Screenshot (Create Parameter Form):**

- The "Create parameter" form is shown under "Parameter details".
- The "Name" field is populated with "your-db-host.rds.amazonaws.com".
- The "Description" field is empty.
- The "Tier" section shows two options:
  - Standard: Stores up to 10,000 standard parameters. Store parameter values up to 4 KB. Parameter policies and sharing with other AWS accounts are not available. No additional charge.
  - Advanced: Stores up to 100,000 advanced parameters. Store parameter values up to 8 KB. Add parameter policies. Share with other AWS accounts. Charges apply.
- A note at the bottom states: "Standard parameters cannot be shared with other AWS accounts." with a "Learn more" link.

The screenshot shows the AWS Parameter Store interface. A modal window is open for creating a new parameter. The 'Value' field contains the value 'database-1.cqnsasm6isg71.us-east-1.rds.amazonaws.com'. A note at the top of the modal states: 'You have selected the default AWS managed key. AWS managed keys cannot be shared with other AWS accounts, and all users in this AWS account and Region have access to the key. Use a customer managed key for sharing and access control.' Below the note is a link 'Learn more'. The 'Tags — Optional' section is present but empty. At the bottom right of the modal are 'Cancel' and 'Create parameter' buttons. The background shows a list of existing parameters: db-host, db-name, db-pass, and db-username, all of which are of type SecureString and were created on Wednesday, 15 Oct 2025.

## Configure IAM Roles and Policies

### Action:

1. Go to **IAM → Roles → Create Role**.
2. Select **EC2** as trusted entity.
3. Attach managed policies:
  - o **AmazonSSMReadOnlyAccess** (to read Parameter Store)
  - o **AmazonS3FullAccess** (optional, for storing assets)
4. Name the role (**WebappRole**)

## Screenshot:

The screenshot shows the AWS IAM Roles page. The top navigation bar includes the AWS logo, a search bar, and account information (Account ID: 2678-0021-3683). The main content area displays a success message: "Role webapp-role created." Below this, there are details like "Last activity" and "Maximum session duration". A tabs menu at the top right includes "View role", "Permissions", "Trust relationships", "Tags", "Last Accessed", and "Revoke sessions". Under the "Permissions" tab, a table lists two managed policies: "AmazonS3FullAccess" and "AmazonSSMReadOnlyAccess", both of which are AWS managed policies.

## Create S3 Bucket for Application Assets

### Action:

1. Navigate to S3 → Create Bucket.
2. Name: phpwebapp1-bucket
3. Enable:
  - Server-Side Encryption (SSE-S3)

## Screenshot:

The screenshot shows the AWS S3 Buckets page. The top navigation bar includes the AWS logo, a search bar, and account information (Account ID: 2678-0021-3683). The main content area shows a table for "General purpose buckets" with one entry: "phpwebapp1-bucket" (US East (N. Virginia) us-east-1). The table includes columns for Name, AWS Region, and Creation date. To the right of the table are three informational boxes: "Account snapshot" (Updated daily), "View dashboard", and "Storage Lens provides visibility into storage usage and activity trends"; "External access summary - new" (Updated daily), and "External access findings help you identify bucket permissions that allow public access or access from other AWS accounts".

## Deploy PHP Application using Elastic Beanstalk

### Action:

1. Go to **Elastic Beanstalk → Create Application.**
2. Application Name: `php-webapp`
3. Platform: **PHP**
4. Upload your application ZIP file.
5. Under **Configuration:**
  - o Environment Type: **Load balanced, auto-scaling**
  - o Select your custom **VPC** and **subnets**
  - o Attach **EC2AppRole** IAM role
6. Deploy the application and verify environment health.

### Screenshot:

The screenshot shows the 'Configure environment' step of the 'Create environment' wizard. On the left, a sidebar lists optional steps: 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 two sections: 'Environment tier' and 'Application information'. In 'Environment tier', 'Web server environment' is selected. In 'Application information', the 'Application name' field is filled with 'phpwebapp'. A note states: 'Maximum length of 100 characters.'

The screenshot shows the same 'Configure environment' step, but the 'Platform' dropdown in the 'Platform' section is open, displaying options: Choose a platform, .NET Core on Linux, .NET on Windows Server, Docker, Go, Java, Node.js, PHP, Python, and Ruby. The 'PHP' option is highlighted.

Screenshot of the AWS Elastic Beanstalk 'Create environment' wizard Step 1: Application code

**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.

**Presets** [Info](#)

Start from a preset that matches your use case or choose custom configuration to unset recommended values and use the service's default values.

**Configuration presets**

Single instance (free tier eligible)

Single instance (using spot instance)

High availability

High availability (using spot and on-demand instances)

Custom configuration

[Cancel](#) [Next](#)

Screenshot of the AWS Elastic Beanstalk 'Create environment' wizard Step 2: Set up networking, database, and tags - optional

**Step 1**

- Configure environment
- Configure service access
- Set up networking, database, and tags
- Configure instance traffic and scaling
- Configure updates, monitoring, and logging
- Review

**Instance settings**  
Choose a subnet in each AZ for the instances that run your application. To avoid exposing your instances to the Internet, run your instances in private subnets and load balancer in public subnets. To run your load balancer and instances in the same public subnets, assign public IP addresses to the instances. [Learn more](#)

**VPC**  
Launch your environment in a custom VPC instead of the default VPC. You can create a VPC and subnets in the VPC management console. [Learn more](#)

vpc-08883208ed1d53513 | (172.31.0.0/16) | My vpc-vpc [Edit](#) [Create VPC](#)

**Public IP address**  
Assign a public IP address to the Amazon EC2 instances in your environment.  
 Enable

**Instance subnets**

Filter instance subnets	Availability Zone	Subnet	CIDR	Name
	us-east-1b	subnet-025b170fa991ec0fa	172.31.144.0/20	My vpc-subnet-private2-u...

Screenshot of the AWS Elastic Beanstalk 'Create environment' wizard Step 3: Capacity

**Capacity** [Info](#)  
Configure the compute capacity of your environment and auto scaling settings to optimize the number of instances used.

**Auto scaling group**

**Environment type**  
Select a single-instance or load-balanced environment. You can develop and test an application in a single-instance environment to save costs and then upgrade to a load-balanced environment when the application is ready for production. [Learn more](#)

Load balanced

**Min instances** 2 instances

**Max instances** 2 instances

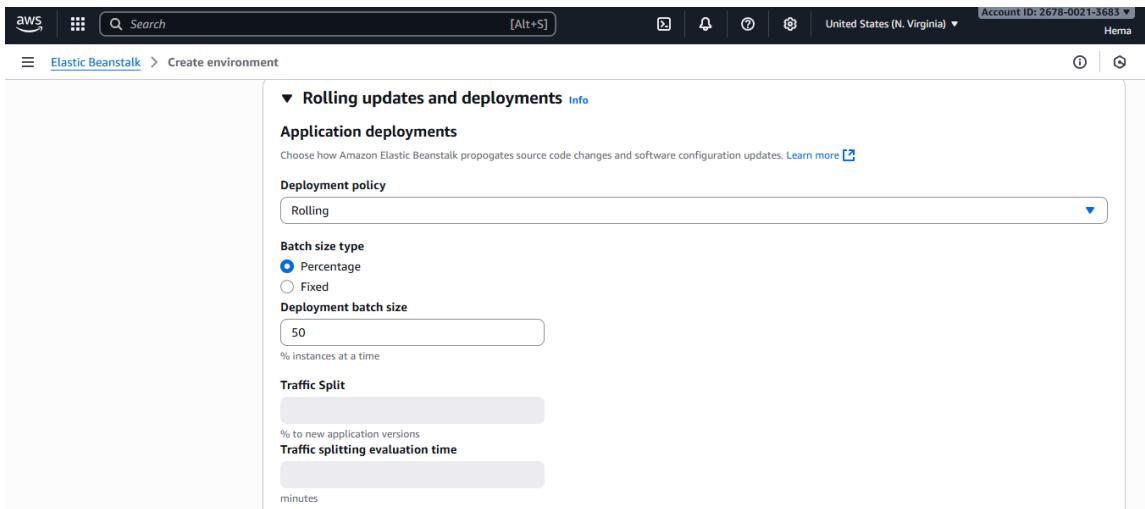
**Fleet composition**  
Spot instances are launched at the lowest available price. [Learn more](#)

On-Demand instances

Combine purchase options and instances

**Architecture**  
The processor architecture determines the instance types that are made available. You can't change this selection after you create the environment. [Learn more](#)

x86\_64  
This architecture uses x86 processors and is compatible with most third-party tools and libraries.



--→Create environment

---

## Set Up Load Balancer and Auto Scaling

### Action:

1. Open **Elastic Beanstalk** → **Configuration** → **Capacity**.
2. Configure:
  - o **Minimum Instances:** 2
  - o **Maximum Instances:** 2
  - o **Scaling Trigger:** CPU Utilization
  - o Scale Out: CPU > 70%
  - o Scale In: CPU < 30%
3. Elastic Beanstalk automatically creates a **Load Balancer** to distribute traffic among EC2 instances.

## Screenshot:

**EC2 Instances**

Instances (2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
Phpwebapp-env	i-0c85a82d3e2658db0	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a
Phpwebapp-env	i-0cb6985aeba99ee27	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b

Select an instance

**EC2 Security Groups**

Security Groups (5) Info

Name	Security group ID	Security group name	VPC ID
Default	sg-03a70350d328c20		vpc-0e0e0000.10.0e0a0032
Database security gr...	sg-015dbc42c2c43ecce	Database security group	vpc-08883208ed1d53513
Phpwebapp-env	sg-083c877f4a1f64d29	awseb-e-pueaukvpsp-stack-AWSEBSecu...	vpc-08883208ed1d53513
Phpwebapp-env	sg-055a171cb48c1511	awseb-e-pueaukvpsp-stack-AWSEBLoa...	vpc-08883208ed1d53513

Select a security group

**Edit inbound rules**

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
-	MySQL/Aurora	TCP	3306	Cust...	sg-083c877f4a1f64d29

Add rule

Use: "sg-083c877f4a1f64d29"

CIDR blocks

Security Groups

awseb-e-pueaukvpsp-stack-AWSEBSecurityGroup-B4L1MR91Qd | sg-083c877f4a1f64d29

Phpwebapp-env

Save rules

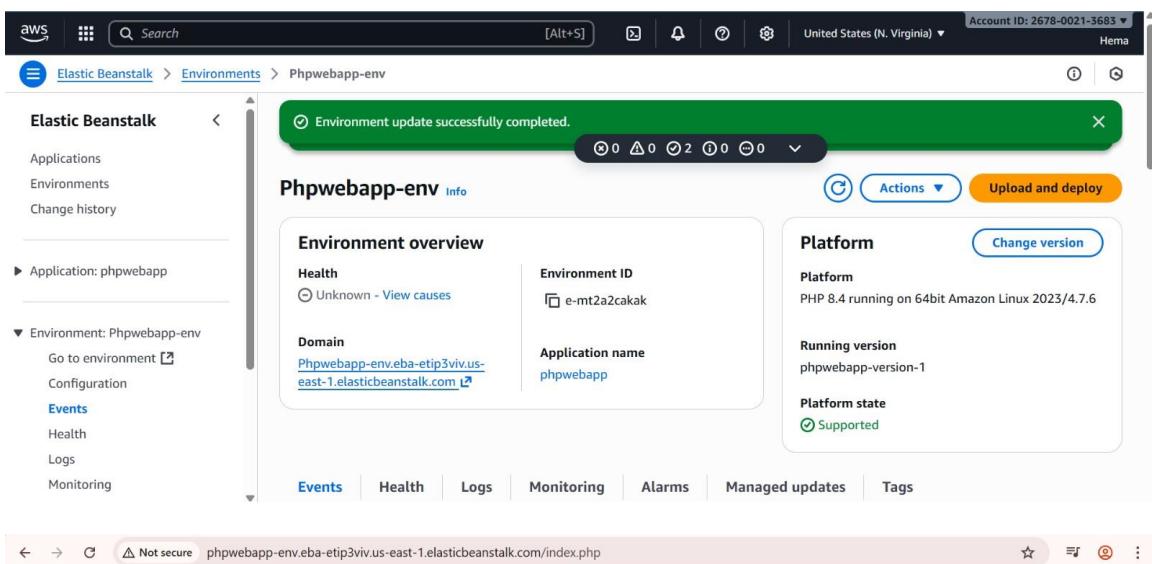
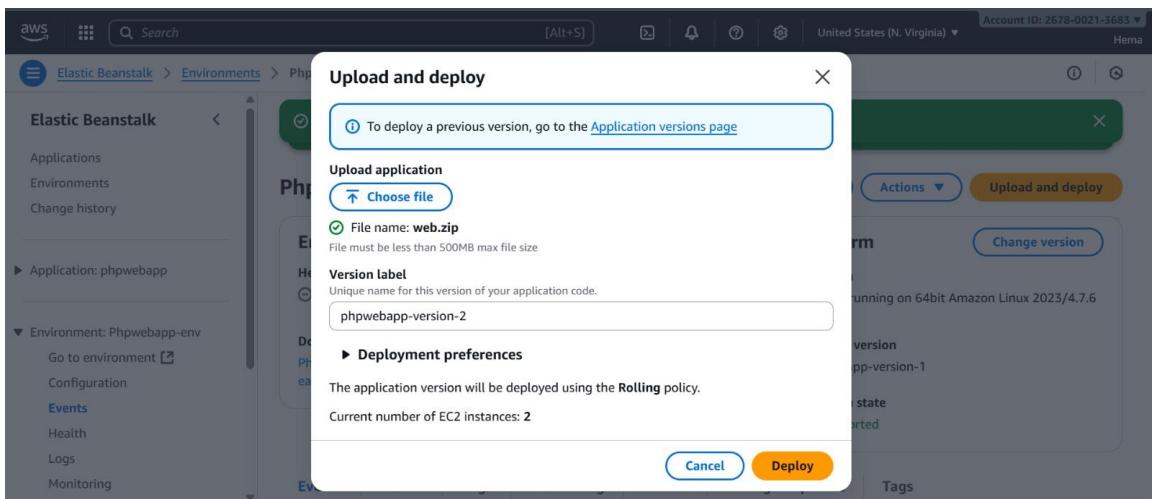
The screenshot shows the AWS EC2 Load Balancers console. On the left, a navigation sidebar lists various services: AMIs, AMI Catalog, Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), Load Balancing (Load Balancers, Target Groups, Trust Stores), and Auto Scaling. The 'Load Balancers' section is selected. The main pane displays a table titled 'Load balancers (1)'. The table has columns for Name, State, Type, Scheme, IP address type, and VPC ID. One row is listed: 'awseb--AWSEB-7QtmiE...' (Active, application, Internet-facing, IPv4, vpc-08883208ed1d5351). A message at the bottom says '0 load balancers selected'.

The screenshot shows the AWS EC2 Target groups console. The left sidebar includes the same service categories as the previous screenshot. The 'Target Groups' section is selected. The main pane shows a table titled 'Target groups (1/1)'. The table columns are Name, ARN, Port, Protocol, Target type, and Load balancer. One entry is shown: 'awseb-AWSEB-DK2GG8BRLI0G' (arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/DK2GG8BRLI0G, 80, HTTP, Instance, awseb--AWSEB-7QtmiE...). Below this, a detailed view for the target group 'awseb-AWSEB-DK2GG8BRLI0G' is shown, listing two healthy instances: 'i-0c85a82d3e2658db0' and 'i-0cb6983aeba99ee27', both associated with 'Phpwebapp-env' and port 80.

The screenshot shows the AWS Elastic Beanstalk console. The top bar indicates 'AWS Elastic Beanstalk'. The main message reads: 'Welcome to Your Elastic Beanstalk Application' and 'Congratulations! Your PHP application is now running on your own dedicated environment in the AWS Cloud.' A 'Learn More' button is present. Below this, a section titled 'Benefits of AWS Elastic Beanstalk' is described with the text: 'Discover why thousands of developers rely on AWS Elastic Beanstalk to deploy and manage their applications.'

## Benefits of AWS Elastic Beanstalk

Discover why thousands of developers rely on AWS Elastic Beanstalk to deploy and manage their applications.



## Contact Us

Name:  
Hema

Email:  
hems25112002@gmail.com

Message:  
For Test Check

Thanks for contacting us, Hema! We'll get back to you at hems25112002@gmail.com.

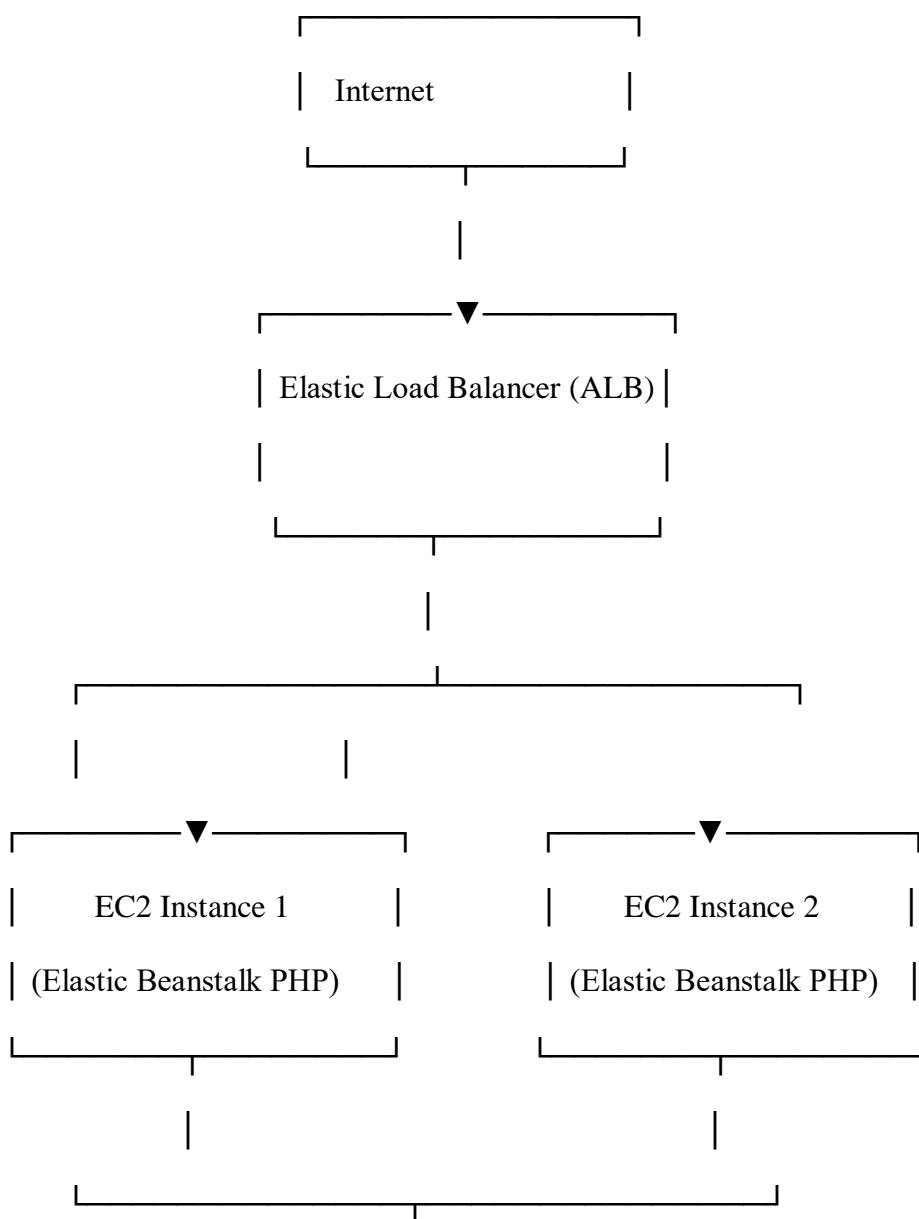
## Final Architecture Diagram

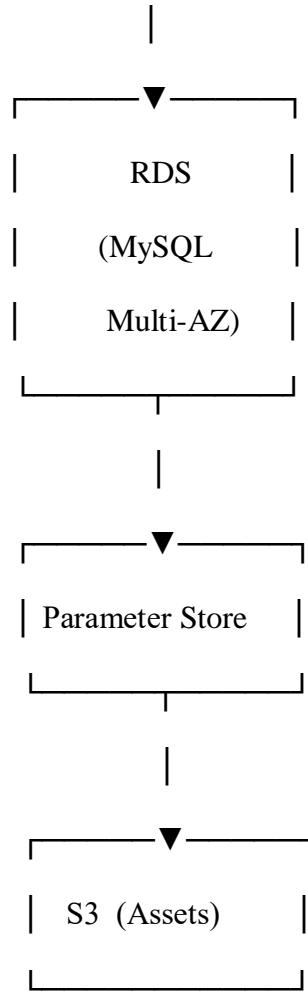
### Action:

Draw a diagram showing:

- VPC with public/private subnets
- Elastic Beanstalk (PHP app) in private subnet
- Load Balancer in public subnet
- RDS in private subnet
- S3 for static assets
- IAM roles and Parameter Store integration

### Screenshot:





---

### ❖ Conclusion

This architecture provides a **secure, scalable, and highly available AWS environment** for a PHP web application using:

- **VPC** for network isolation
- **Security Groups** for access control
- **RDS** for managed database
- **Parameter Store** for secrets
- **Elastic Beanstalk** for app deployment
- **EC2 + Load Balancer** for high availability
- **S3** for static assets