

# **Evershop Deployment on AWS EC2**

**Name : Goutham K**

# **Evershop Deployment on AWS EC2**

## **Objective:**

**To deploy the open-source Evershop e-commerce platform on AWS EC2, ensuring it runs smoothly, is accessible using a Load Balancer, and is scalable using Auto Scaling Groups. The project is divided into 3 main tasks.**

- **Task 1: Setup and configuration of EC2 and server environment**
- **Task 2: Deployment and configuration of the Evershop application**
- **Task 3: Implementation of Load Balancer and Auto Scaling**

## **Task 1 : EC2 Instance Setup and Server Environment Configuration**

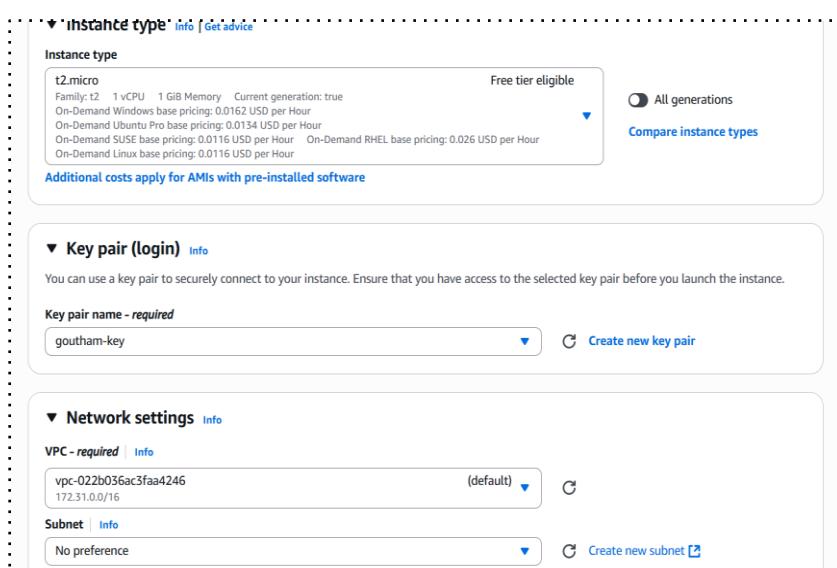
### **Objective:**

**Launch and configure an Ubuntu EC2 instance with all required software to run a Node.js e-commerce application.**

# Steps Performed:

## 1. Create EC2 Instance:

- AMI: Ubuntu 22.04 LTS
- Instance Type: t2.medium
- Created a new key pair (for SSH access).
- Downloaded the .pem file safely.



Now after selecting the required configurations check once again if everything looks fine and then allow all https and ssh in edit , then finally click launch.



## 2. Security Group Configuration:

### Inbound Rules:

- SSH (Port 22) – From My IP
- HTTP (Port 80) – From Anywhere (0.0.0.0/0)
- Custom TCP (Port 3000) – From Anywhere (Evershop runs here)

The screenshot shows the AWS EC2 'Create security group' interface. In the 'Basic details' section, the security group name is 'web-server-sg' and the description is 'Allow HTTP and SSH access from my IP'. The VPC is set to 'vpc-01e488a2d2dfeecd6'. In the 'Inbound rules' section, there are two rules: 'Inbound rule 1' (SSH port 22 from 0.0.0.0/0) and 'Inbound rule 2' (Custom TCP port 3000 from 0.0.0.0/0).

**Create security group Info**

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

**Basic details**

Security group name Info  
web-server-sg  
Name cannot be edited after creation.

Description Info  
Allow HTTP and SSH access from my IP

VPC Info  
vpc-01e488a2d2dfeecd6

**Inbound rules Info**

**Inbound rule 1** Delete

Security group rule ID sgr-007e5a0fe2bb7ab68	Type <small>Info</small> SSH	Protocol <small>Info</small> TCP
Port range <small>Info</small> 22	Source type <small>Info</small> Custom	Source <small>Info</small> 0.0.0.0/0 <small>X</small>
Description - optional <small>Info</small> 		

**Inbound rule 2** Delete

Security group rule ID sgr-0c5508a019dcea474	Type <small>Info</small> Custom TCP	Protocol <small>Info</small> TCP
Port range <small>Info</small> 3000	Source type <small>Info</small> Custom	Source <small>Info</small> 0.0.0.0/0 <small>X</small>
Description - optional <small>Info</small> 		

**Security group (sg-0d8c3f912a4bebc8a | web-server-sg) was created successfully**

**Details**

Security group name web-server-sg	Security group ID sg-0d8c3f912a4bebc8a	Description Allow HTTP and SSH access from my IP	VPC ID vpc-01e488a2d2dfeecd6
Owner 127854693057	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

**Inbound rules** | Outbound rules | Sharing - new | VPC associations - new | Tags

**Inbound rules (2)**

Name	Security group rule ID	IP version	Type
-	sgr-088f41631e0b456cd	IPv4	SSH
-	sgr-09c364c05047c6b29	IPv4	HTTP

## Connect to EC2 via SSH:

Open your windows powershell using administrator and enter this command

“chmod 400 your-key.pem”

“ssh -i "your-key.pem" ubuntu@your-ec2-public-ip”

## Update System:

“sudo apt update && sudo apt upgrade -y”

```
ubuntu@ip-172-31-82-96:~$ curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
2025-06-23 21:17:05 - Installing pre-requisites
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
```

**With this the task 1 is done let's move on to 2nd task**

## **Task 2 : Evershop web Deployment on EC2**

### **Objective:**

**To run the Evershop Node.js application, configure it for production, and set up admin access.**

### **Steps Performed:**

- 1. Extract the EverShop in terminal from github source in VS-Code**

#### **Create a file : In Vs code terminal**

**“mkdir evershop”**

#### **Navigate it:**

**“cd evershop”**

#### **NOW :**

**Start the evershop app**

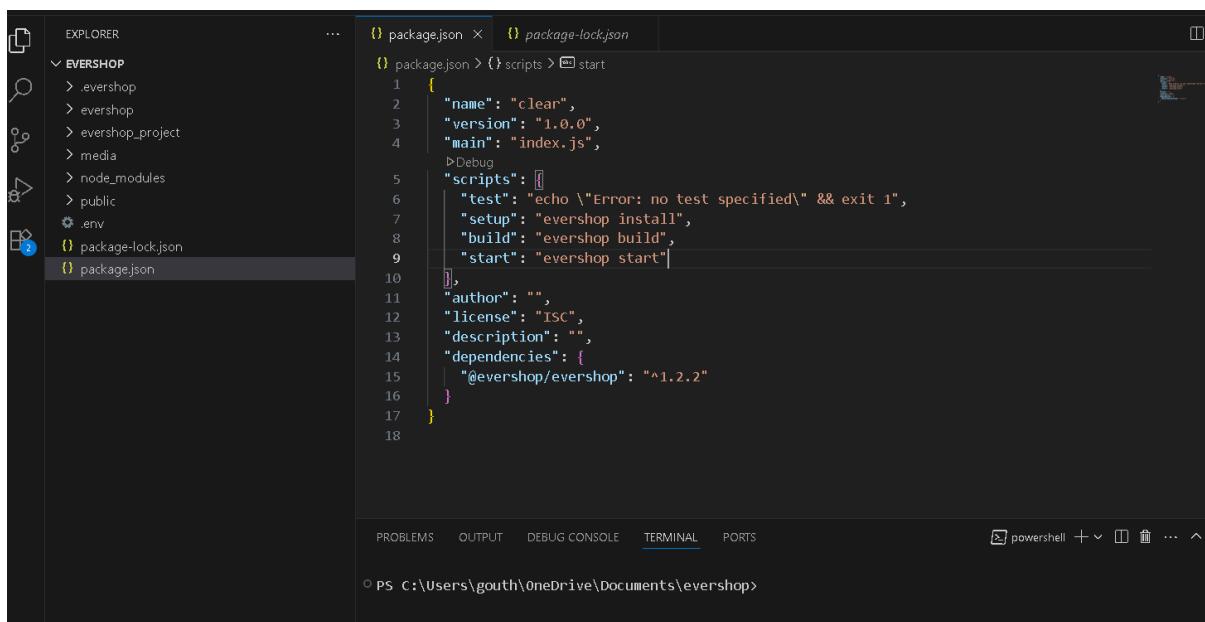
**“npm init”**

**“npm run setup”**

**“npm run build”**

**“npm run start”**

**After working these commands in your VS-code terminal you will get to see the code like this and you will be seeing a command Local Host running .**



The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying a project structure for 'EVERSHOP' containing files like '.env', 'node\_modules', 'public', 'evershop', 'evershop\_project', 'media', and 'evershop.lock.json'. The 'package.json' file is selected in the Explorer and is open in the main editor area. The code in the editor is:

```
1 {  
2   "name": "clear",  
3   "version": "1.0.0",  
4   "main": "index.js",  
5   "scripts": [  
6     "test": "echo \\\"Error: no test specified\\\" && exit 1",  
7     "setup": "evershop install",  
8     "build": "evershop build",  
9     "start": "evershop start"  
10    ],  
11    "author": "",  
12    "license": "ISC",  
13    "description": "",  
14    "dependencies": {  
15      "@evershop/evershop": "^1.2.2"  
16    }  
17  }  
18
```

The bottom status bar shows the terminal tab is active, and the command line indicates the user is in a PowerShell session at 'C:\Users\gouth\OneDrive\Documents\evershop>'.

**Now make sure to download the file and make it as a zip and save it**

**In the clean location that you can you it**

**Now in powershell “run as administrator” and paste this comment it will**

**Extract the zip file.**

**“scp -i "your-key.pem" evershop.zip ubuntu@your-ec2-public-ip:/home/ubuntu/”**

```
ec2-user@ip-172-31-89-213:~/evershop$ 
inflating: evershop/node_modules/yargs/locales/ru.json
inflating: evershop/node_modules/yargs/locales/th.json
inflating: evershop/node_modules/yargs/locales/tr.json
inflating: evershop/node_modules/yargs/locales/zh_CN.json
inflating: evershop/node_modules/yargs/locales/zh_TW.json
inflating: evershop/node_modules/yargs/package.json
inflating: evershop/node_modules/yargs/README.md
inflating: evershop/node_modules/yargs/yargs
inflating: evershop/node_modules/yargs-parser/browser.js
inflating: evershop/node_modules/yargs-parser/build/index.cjs
inflating: evershop/node_modules/yargs-parser/build/lib/index.js
inflating: evershop/node_modules/yargs-parser/build/lib/string-utils.js
inflating: evershop/node_modules/yargs-parser/build/lib/tokenize-arg-string.js
inflating: evershop/node_modules/yargs-parser/build/lib/yargs-parser-types.js
inflating: evershop/node_modules/yargs-parser/build/lib/yargs-parser.js
inflating: evershop/node_modules/yargs-parser/CHANGELOG.md
inflating: evershop/node_modules/yargs-parser/LICENSE.txt
inflating: evershop/node_modules/yargs-parser/package.json
inflating: evershop/node_modules/yargs-parser/README.md
inflating: evershop/node_modules/zero-decimal-currencies/lib/error-factory.js
inflating: evershop/node_modules/zero-decimal-currencies/lib/index.js
inflating: evershop/node_modules/zero-decimal-currencies/LICENSE
inflating: evershop/node_modules/zero-decimal-currencies/package.json
inflating: evershop/node_modules/zero-decimal-currencies/README.md
inflating: evershop/package-lock.json
inflating: evershop/package.json
ec2-user@ip-172-31-89-213 ~]$ cd evershop
ec2-user@ip-172-31-89-213 evershop]$ ls
evershop evershop_project media node_modules package-lock.json package.json public
ec2-user@ip-172-31-89-213 evershop]$
```

**Now again use your SSH and get into EC2 instance**

## **2.Unzipped the File:**

**“unzip evershop.zip”**

**“cd evershop\_project”**

```
os: name for https://github.com : 
-bash: cd: evershop: No such file or directory
ubuntu@ip-172-31-86-203:~$ cd ~
wget https://github.com/openstarcloud/evershop-backup/archive/refs/heads/main.zip -O evershop.zip
unzip evershop.zip
mv evershop-backup-main evershop
```

**“sudo npm install -g pm2”**

```
ubuntu@ip-172-31-19-43:~$ sudo npm install -g pm2
pm2 startup
```

```
“curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash”  
“sudo apt install -y nodejs”  
“node -v && npm -v”
```

```
ubuntu@ip-172-31-19-43:~$ curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -  
sudo apt install -y nodejs  
node -v  
npm -v■
```

## 5. Install and Configure PostgreSQL:

```
“sudo apt install postgresql postgresql-contrib -y”  
“sudo -u postgres psql”
```

### Inside psql:

**Make sure to do this thing for admin set up**

```
“CREATE USER my_evershop WITH PASSWORD 'admin123';  
CREATE DATABASE evershop;  
GRANT ALL PRIVILEGES ON DATABASE evershop To  
my_evershop  
\q”
```

**Edit PostgreSQL config for password auth:**

```
“sudo nano /etc/postgresql/14/main/pg_hba.conf”
```

“sudo systemctl restart postgresql”

## 6. Set up Environment for PostgreSQL

**DB\_CLIENT=pg**

**DB\_HOST=localhost**

**DB\_PORT=5432**

**DB\_USER=my\_evershop**

**DB\_PASSWORD=admin123**

**DB\_DATABASE=evershop**

## 7. Admin User Setup:

Go to PostgreSQL and manually insert admin user: # by using “sudo -u postgres psql”

“**INSERT INTO admin\_user (id, uuid, status, email, password, full\_name, created\_at, updated\_at)**

**VALUES (1, gen\_random\_uuid(), true, 'admin@example.com', '\$2b\$10\$YQa8D3G2gxyXony6lyhwfe/rVkJrEfEw0EkekjC98Dua5cGX5l6y', 'Admin', now(), now());”**

Now if you done the things successfully you can get into the admin page successfully

## 8. Admin User Setup:

**“npm install”**

**“npm run build”**

```
ubuntu@ip-172-31-86-203:~/evershop$ npm run build
> evershop@1.0.0-beta build
> evershop build

WARNING: NODE_ENV value of 'production' did not match any deployment config file names.
WARNING: See https://github.com/node-config/node-config/wiki/Strict-Mode

• Client [██████████] building (10%) 0/54 entries 2160/2200 dependencies 2/350 modules 242 active
  ...ader > packages/evershop/src/modules/cod/pages/frontStore/checkout/CashOnDelivery.jsx

• Server [██] building (10%) 0/108 entries 2251/2304 dependencies 1/342 modules 236 active
  ...r > packages/evershop/src/modules/checkout/pages/frontStore/checkout/ShippingNote.jsx

Browserslist: browsers data (caniuse-lite) is 9 months old. Please run:
  npx update-browserslist-db@latest
  Why you should do it regularly: https://github.com/browserslist/update-db#readme
```

## 9. Run with PM2:

**“npm run start”**

```
ubuntu@ip-172-31-87-250:~/evershop-clean/evershop-app$ export NODE_ENV=production
npm run start

> evershop@1.0.0 start
> evershop start

----- EverShop -----
Your website is running at "http://localhost:3000"
```

**So by this you successfully deployed the evershop folder in the EC2 instance and it is in the running state and yet there is a process to go.**

**We need to save this in the background of my PM2 so even after closing the instance it will still run the site.**

**“sudo npm install pm2@latest -g”**

**“pm2 start npm --name "evershop" -- start”**

**“pm2 startup”**

**“pm2 save”**

```
6Z-debug-0.log
ubuntu@ip-172-31-87-250:~/evershop-clean/evershop-app$ sudo npm install -g pm2
changed 135 packages in 7s

13 packages are looking for funding
  run `npm fund` for details
ubuntu@ip-172-31-87-250:~/evershop-clean/evershop-app$ pm2 start npm --name evershop -- start
[PM2] Starting /usr/bin/npm in fork_mode (1 instance)
[PM2] Done.



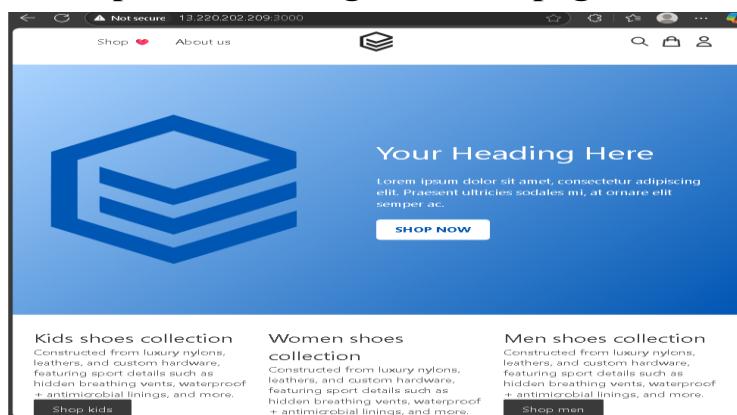
| id | name     | mode | pid | status | cpu | memory |
|----|----------|------|-----|--------|-----|--------|
| 0  | evershop | fork | 0   | online | 0%  | 15.3mb |



ubuntu@ip-172-31-87-250:~/evershop-clean/evershop-app$ pm2 save
[PM2] Saving current process list...
[PM2] Successfully saved in /home/ubuntu/.pm2/dump.pm2
ubuntu@ip-172-31-87-250:~/evershop-clean/evershop-app$ pm2 startup
[PM2] Init System found: systemd
```

**By this the task 2 is done and your website will be running in the background so you can test it by “<http://<PublicIP>:3000>”**

**To access the admin page “<http://<PublicIP>:3000/admin>” now put your username and password to login into the page**



**Now you can access the dashboard and access thing through this**

**Link - [EC2-IP](#)**

**By this Task 2 is done.**

**Task 3: Load Balancing and Auto Scaling**

**1. Launch Template:**

create a launch template using existing EC2 config.

Set port **3000** as the app runs there.



Just do the same what we did while creating EC2 instance and attach the same security group in this and click the advanced tab and in user data

```
#!/bin/bash
```

```
# Update system
```

```
apt update -y && apt upgrade -y
```

```
# Install dependencies
```

```
apt install -y curl git unzip build-essential postgresql postgresql-contrib
```

```
# Install Node.js (v18 LTS)
```

```
curl -fsSL https://deb.nodesource.com/setup_18.x | bash -
```

```
apt install -y nodejs
```

```
cd evershop-clean/evershop-app
```

```
# Setup .env for PostgreSQL
```

```
cat <<EOF > .env
```

```
DB_CLIENT=pg
```

```
DB_HOST=localhost
```

```
DB_PORT=5432
```

```
DB_USER=my_evershop
```

```
DB_PASSWORD=admin123
```

```
DB_DATABASE=evershop
```

```
EOF
```

```
# Install node modules and build app
```

```
npm install
```

```
npm run build
```

```
# Setup PM2
```

```
npm install -g pm2
```

```
pm2 start npm --name "evershop" -- start
```

```
pm2 startup
```

```
pm2 save
```

This User Data script automatically installs all dependencies, sets up PostgreSQL config, builds the app, and starts it using PM2 on EC2 launch.

## 2. Create Target Group:

- Type: Instance
- Protocol: HTTP, Port: 3000

- Health check path: **/health**

Protocol	Port
Protocol for load balancer-to-target communication. Can't be modified after creation.	Port number where targets receive traffic. Can be overridden for individual targets during registration.  HTTP ▾ 3000 1-65535

**In EC2 Type “nano server.js”**

“app.get('/health', (req, res) => res.status(200).send('OK'));” -  
paste this

**Press : ctrl + o - to save**

**Press : enter**

**Press : ctrl + x - to exit**

**Now this will be saved in your js server**

### **3. Application Load Balancer:**

- Create ALB with public subnets.

- Attach the target group.

**Load balancers** (C) Actions ▾ Create load balancer ▼

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Filter load balancers

**Basic configuration**

**Load balancer name**  
Name must be unique within your AWS account and can't be changed after the load balancer is created.  
**evershop-alb**

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

**Scheme** Info  
Scheme can't be changed after the load balancer is created.

**Internet-facing**

- Serves Internet-facing traffic.
- Has public IP addresses.
- DNS name resolves to public IPs.
- Requires a public subnet.

**Internal**

- Serves internal traffic.
- Has private IP addresses.
- DNS name resolves to private IPs.
- Compatible with the **IPv4** and **Dualstack** IP address types.

**Load balancer IP address type** Info  
Select the front-end IP address type to assign to the load balancer. The VPC and subnets mapped to this load balancer must include the selected IP address types. Public IPv4 addresses have an additional cost.

**IPv4**  
Includes only IPv4 addresses.

**Dualstack**  
Includes IPv4 and IPv6 addresses.

**Dualstack without public IPv4**  
Includes a public IPv6 address, and private IPv4 and IPv6 addresses. Compatible with **internet-facing** load balancers only.

**Select up to 5 security groups** (C)

evershop-sec  
sg-0ad1b41aba70a69eb VPC: vpc-022b036ac3faa4246

**Listeners and routing** Info

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

**Listener HTTP:80** Remove

<b>Protocol</b>	<b>Port</b>
HTTP	80 1-65535

**Default action** Info

<b>Forward to</b>	<b>evershop-tg</b>	<b>HTTP</b> <span>▼</span>
Target type: Instance, IPv4		

**Create target group** (C)

**Listener tags - optional**  
Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

**Add listener tag**

- Use a launch template.
- Attach to the ALB target group.
- Min: 1, Max: 2 instances.
- Health checks: ALB

Auto Scaling groups (0) Info

Last updated 15 minutes ago C

**Launch configurations** C

**Launch templates** C

**Actions** ▼ **Create Auto Scaling group**

**Search your Auto Scaling groups**

**Availability Zones and subnets**  
Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones... C

use1-az6 (us-east-1b) | X

subnet-  
0921b0a46ffc0910f  
172.31.32.0/20 Default

use1-az2 (us-east-1d) | X

subnet-  
09336338b02d23ff1  
172.31.80.0/20 Default

Create a subnet C

**Availability Zone distribution - new**  
Auto Scaling automatically balances instances across Availability Zones. If launch failures occur in a zone, select a strategy.

**Balanced best effort**  
If launches fail in one Availability Zone, Auto Scaling will attempt to launch in another healthy Availability Zone.

**Balanced only**  
If launches fail in one Availability Zone, Auto Scaling will continue to attempt to launch in the unhealthy Availability Zone to preserve balanced distribution.

**Cancel** **Skip to review** **Previous** **Next**

Enable EC2 to ELB in that and create it.

## 5. Validate Health Checks

- Make sure /health returns HTTP 200.
- Target Group shows healthy

CloudWatch Metrics

i-0725f46e1819c8ef2 us-east-1d (us...)

**Healthy**

Settings

Since its healthy everything is fine we can copy our DNS in ABL and use it in url and we can see our output

Not secure evershop-alb-1450038116.us-east-1.elb.amazonaws.com



SHOP NOW

**Kids shoes collection**  
Constructed from luxury nylons, leathers, and custom hardware, featuring sport details such as hidden breathing vents, waterproof + antimicrobial linings, and more.  
[Shop kids](#)

**Women shoes collection**  
Constructed from luxury nylons, leathers, and custom hardware, featuring sport details such as hidden breathing vents, waterproof + antimicrobial linings, and more.  
[Shop women](#)

**Men shoes collection**  
Constructed from luxury nylons, leathers, and custom hardware, featuring sport details such as hidden breathing vents, waterproof + antimicrobial linings, and more.  
[Shop men](#)

FEATURED PRODUCTS



 EVERSHOP A

Search

QUICK LINKS

- Dashboard
- New Product
- New Coupon

CATALOG

- Products
- Categories
- Collections
- Attributes

SALE

- Orders

CUSTOMER

- Customers

PROMOTION

- Coupons

CMS

- Pages
- SETTING

## Dashboard

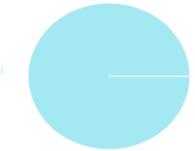
Sale Statistics

Daily Weekly Monthly



Lifetime Sales

0 orders  
\$0.00 lifetime sale  
0% of orders completed  
0% of orders cancelled



Best Sellers

All products

Look like you just started. No bestsellers yet.

We can see that the ALB is working fine and we can get into dash bord and change the things as how we like

**LINK - [EverShop](#) or <http://evershop-alb-1450038116.us-east-1.elb.amazonaws.com/>**

**Admin :**

User: admin@example.com

Pass: admin123

**You can access this like its in working condition.**

**Tools Used:**

- AWS EC2
- PostgreSQL
- Node.js, PM2
- ALB & Auto Scaling
- GitHub (custom repo)
- Admin setup via SQL

By this we completed every task which was assigned and the project was completed .

## **Intern Certificate:**

**NULCLASS**

Issued Jun 15 2025

### **CERTIFICATE OF TRAINING**

THIS CERTIFICATE IS PRESENTED TO

**Goutham .K**

has successfully completed online training on  
**Cloud Technology**  
and real-time project training on  
**Learn To Deploy Real Time Website in AWS**



Vetriselvan G.  
CEO



GSTIN: 33AAICN0803F1ZZ  
CIN: U03010TZ2022PTC08231  
DOC ID: 684e87742178383f96a95084