**Project\_23\_Building-an-E-Commerce-Web-Application-on-AWS: 3-Tier-Architecture-with-SSM-WAF-and-CloudFront**

By: Gabin Sime

A diagram of a computer network

AI-generated content may be incorrect.

We’re deploying a production-grade infrastructure to support a modern e-commerce web application. This environment follows a traditional 3-tier model: a web application tier (presentation layer), an application tier (business logic), and a data tier (database).

The infrastructure leverages AWS Systems Manager (SSM) for secure instance access, AWS Web Application Firewall (WAF) for threat mitigation, and Amazon CloudFront to accelerate global content delivery. To secure internal data operations, we attach a VPC Gateway Endpoint to restrict S3 access over the public internet. The environment is engineered for high availability, scalability, and security.

**Introduction**

Throughout my professional experience designing hardware and cloud-based systems, I’ve come to rely on detailed planning and meticulous execution. This has proven essential in building scalable infrastructures on AWS, particularly for complex workloads like e-commerce.

To begin this build, I provisioned a fully isolated Virtual Private Cloud (VPC) environment. The networking layer includes dedicated public and private subnets, Internet Gateway (IGW), NAT Gateways, custom route tables, and tight security group configurations. With the networking foundation complete, I launched Amazon RDS instances for persistent data storage, configured EC2 instances for application workloads, implemented Auto Scaling Groups (ASG) with ALBs, and secured traffic paths using WAF and CloudFront.

An additional security measure included private S3 access via VPC Gateway Endpoints, which eliminates exposure of data transfers over the internet.

**Project Description**

This implementation delivers a complete, production-grade AWS 3-tier architecture:

1. **Web Application Tier (Presentation Layer):**  
   This layer handles all customer-facing UI components. For example, in an online clothing store, users interact with product catalogs, filter options, and initiate checkout processes.
2. **Application Tier (Business Logic Layer):**  
   This component handles all back-end functions such as authentication, order processing, cart logic, and payment integration.
3. **Data Tier (Database Layer):**  
   The persistent storage tier, hosted on Amazon RDS, maintains structured datasets like product metadata, customer accounts, orders, and historical transactions.

**Key Components and AWS Services**

* **VPC/Subnets**: Core network isolation, defining public/private subnet ranges for resource segmentation.
* **NAT Gateways & Route Tables**: Enable outbound internet for private instances.
* **ALBs & ASGs**: Ensure traffic is evenly distributed and automatically scale EC2 capacity.
* **AWS Systems Manager (SSM)**: Secure, keyless connectivity to EC2 without bastion hosts or SSH.
* **Amazon S3 (with VPC Endpoint)**: Secure object storage access within private networks.
* **AWS WAF**: Provides application-layer protection against common web threats.
* **Amazon CloudFront**: Serves static/dynamic content globally with reduced latency.

**Implementation**

**Create IAM Role for EC2 (SSM Access)**

We begin by creating an IAM Role that allows EC2 instances to securely communicate with AWS Systems Manager.

* Navigate to: **IAM** → **Roles** → **Create Role**
* Trusted entity: AWS service
* Use case: EC2
* Permissions policy:
  + Attach the AWS managed policy: *AmazonSSMManagedInstanceCore*
* Create and Attach a Custom Inline Policy

We'll now create a custom policy that grants additional permissions for full SSM functionality and directory service integration:

* Click **Next**, then go to **Add Permissions** → **Create Policy**
* Choose the **JSON** tab and paste the following:

{

  "Version": "2012-10-17",

  "Statement": [

    {

      "Effect": "Allow",

      "Action": [

        "ssm:\*",

        "ec2messages:\*",

        "cloudwatch:PutMetricData",

        "ds:CreateComputer",

        "ds:DescribeDirectories"

      ],

      "Resource": "\*"

    }

  ]

}

* Click **Next**, give it the name *CustomSSMRolePolicy*, and choose **Create policy**
* Back in the role creation workflow, refresh the policy list and attach the newly created *CustomSSMRolePolicy*
* Role name: *SSMRole*
* (Optional) Add tags if needed for organization
* Review the trust relationship and attached policies:
  + - *AmazonSSMManagedInstanceCore*
    - *CustomSSMRolePolicy*
* Click **Create role**

A screenshot of a computer

AI-generated content may be incorrect.

**I. Networking & Security Layer**

**Virtual Private Cloud (VPC)**

A logically isolated network that houses all AWS resources for the application. Set the CIDR block to 10.0.0.0/16.

* AWS Console → **VPC** → **Create VPC**
* Name tag: e.g., Ecom\_VPC
* IPv4 CIDR: 10.0.0.0/16
* Click: **Create VPC**

A screenshot of a computer

AI-generated content may be incorrect.

**Subnets**

We create six subnets across two Availability Zones (AZs):

* 2 Public Subnets [Web-Pub-Subnet-2a & Web-Pub-Subnet-2b ] 🡺 Web Tier
* 2 Private Subnets [App-Priv-Subner-2a & App-Priv-Subnet-2b] 🡺 Application Tier
* 2 Private Subnets [DB-Priv-Subnet-2a & DB-Priv-Subnet-2b] 🡺 Database Tier
* AWS Console → Subnets → Create Subnet
* VPC: Select previously created VPC
* Subnet names: e.g., Web-Pub-AZ1, App-Priv-AZ1, etc.
* AZs: Distribute evenly (AZ1 and AZ2)
* IPv4 CIDRs: Use /24 ranges, e.g., 10.0.1.0/24, 10.0.2.0/24

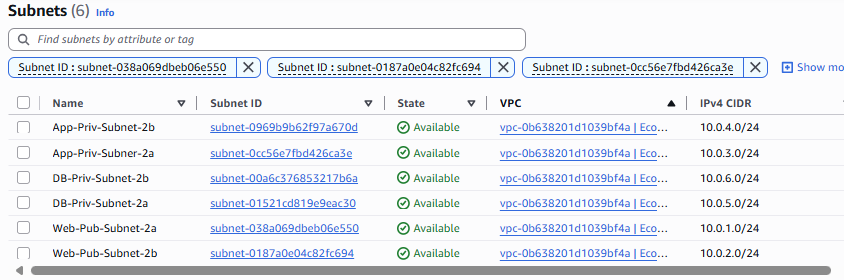
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* Repeat this step 4 more times (2 for App Tier and 2 for Database Tier)



**Internet Connectivity:**

**Internet Gateway (IGW)**

Allows public subnets to reach the internet.

* AWS Console → **Internet Gateways** → **Create Internet Gateway**
* Name: e.g., Ecom-IGW
* Click: **Create**

A screenshot of a computer

AI-generated content may be incorrect.

* Attach to VPC: Select the VPC

A white rectangular object with black text

AI-generated content may be incorrect.

**NAT Gateways**

Enable outbound internet access for instances in private subnets.

* AWS Console → NAT Gateways → Create NAT Gateway
* Subnet: Select each public subnet
* Allocate Elastic IP
* Name: NAT-2a, NAT-2b and Click: **Create**

A screenshot of a computer

AI-generated content may be incorrect.

**Routing Configuration:**

**Route Tables**

Configure a route that forwards all outbound traffic from the VPC to the internet gateway. This involves adding a route entry with the destination 0.0.0.0/0—representing all non-local traffic—pointing to the internet gateway as the target. This ensures that any traffic not matching the VPC CIDR range is directed externally through the designated internet gateway.

**Public Subnet Routing**

* AWS Console → Route Tables → Create
* Name: Publi-RT
* VPC: Select VPC
* Add route: 0.0.0.0/0 → Target: IGW

A screenshot of a computer

AI-generated content may be incorrect.

* Associate: Public subnets

A screenshot of a computer

AI-generated content may be incorrect.

**App Tier Routing (per AZ)**

These route tables are configured to direct outbound application layer traffic—originating from private subnets and destined for addresses outside the VPC—to the NAT gateway within the corresponding Availability Zone. This ensures secure, zone-resilient internet access for resources in the private subnets without exposing them to inbound public traffic.

* Create 2 Route Tables: App-2a-RT, App-2b-RT
* Add route: 0.0.0.0/0 → NAT Gateway (AZ-specific)

A screenshot of a computer

AI-generated content may be incorrect.

* Associate with private app subnets

A screenshot of a computer

AI-generated content may be incorrect.

We should have 3 route tables: Public-RT, App-2a-RT, App-2b-RT

A screenshot of a computer

AI-generated content may be incorrect.

**Security Groups**

Security groups serve as stateful virtual firewalls that enforce fine-grained control over inbound and outbound traffic to our Elastic Load Balancers and EC2 instances. By explicitly defining permitted protocols, ports, and source IP ranges, we ensure only authorized traffic can reach critical infrastructure components, aligning with the principle of least privilege and enhancing the overall security posture of the application.

**a. External ALB SG:**

* Inbound: HTTP from 0.0.0.0/0
* Name: External-ALB-SG

A screenshot of a computer

AI-generated content may be incorrect.

**b. Web Tier SG:**

The second security group is assigned to the public instances in the web tier. Its inbound rule is configured to allow traffic originating from the internet-facing load balancer that was created earlier. This ensures that only traffic routed through our public-facing load balancer is permitted to reach the underlying web instances, enforcing a controlled and secure entry point.

* Inbound: HTTP from External ALB SG
* Inbound: HTTP from My IP

A screenshot of a computer

AI-generated content may be incorrect.

**c. Internal ALB SG:**

The third security group is designated for our internal load balancer. We'll configure it with an inbound rule that explicitly allows traffic from the security group associated with our public web tier instances. This setup ensures that only traffic originating from the web tier is permitted to reach the internal load balancer, maintaining a secure, layered architecture between the application tiers.

* Inbound: HTTP from Web Tier SG

A screenshot of a computer

AI-generated content may be incorrect.

**d. App Tier SG:**

The fourth security group is configured to enable connectivity to our private instances in the application tier. Its inbound rules will allow TCP traffic on port 4000, specifically from the security group associated with our internal load balancer. This permits the load balancer to forward traffic to the application instances where the service is running on port 4000. Additionally, we'll add a rule to allow access from our own IP address for testing and validation purposes, ensuring secure administrative access during deployment.

* Inbound: TCP 4000 from Internal ALB SG
* Add rule for your IP for testing

A screenshot of a computer

AI-generated content may be incorrect.

**e. Database SG:**

The fifth security group is designed to secure our database instances. It includes an inbound rule that permits traffic on port 3306—the default port for MySQL Aurora—exclusively from the security group associated with our private application instances. This ensures that only authorized traffic from the application tier can access the database layer, enforcing strict separation of concerns and minimizing the attack surface.

* Inbound: MySQL (3306) from App Tier SG

A screenshot of a computer

AI-generated content may be incorrect.