



BS-Software Engineering

Software Development & Construction

AWS Architecture Diagram

Instructor: Dr Zunnurain Hussain

Teaching Assistant: Umair Makhdoom

Project Title: Cloud Based E – Commerce Platform

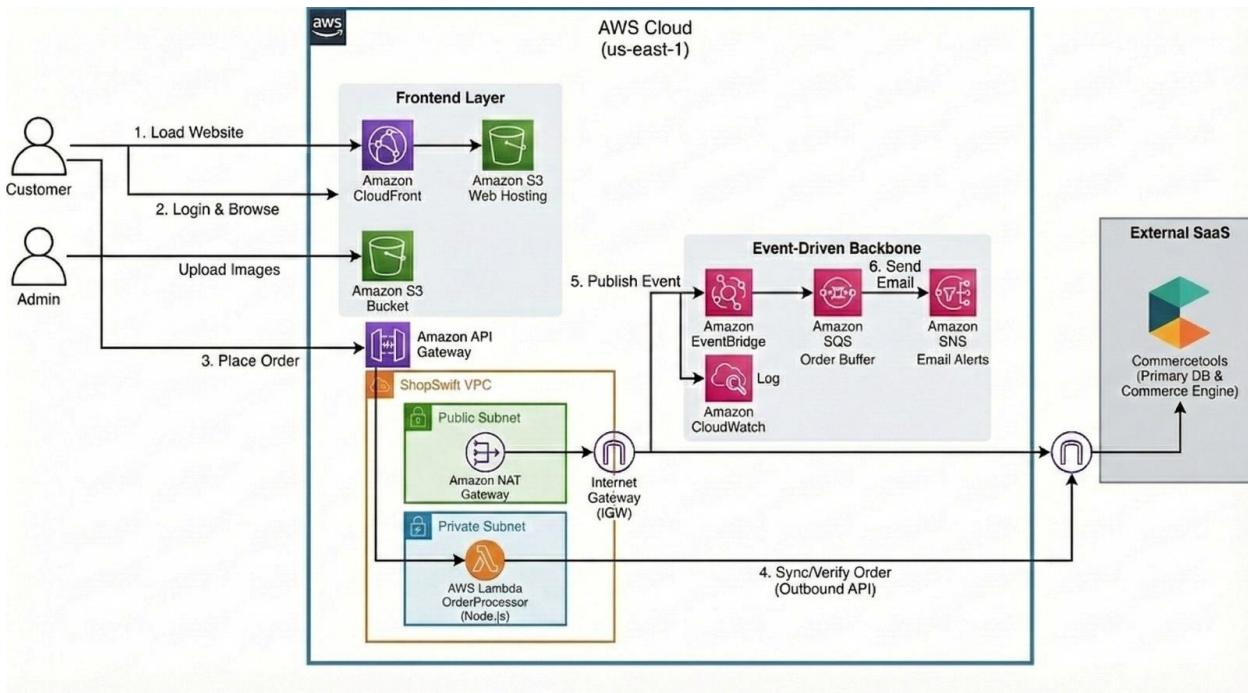


Group Members:

- Muhammad Numan Tahir – BSSE23069
- Muhammad Zubair Akhter – BSSE23079

AWS Architecture Diagram:

A professional, AWS-standard architecture diagram was designed using correct AWS icons and clear data flows. The diagram accurately represents frontend delivery through Amazon S3 and CloudFront, backend processing via API Gateway and Lambda, secure networking using VPC subnets, and event-driven integration with EventBridge, SQS, SNS, and external Commercetools services. All components and interactions are logically connected and easy to understand.



Implementation Steps:

1. An **Amazon S3 bucket** was created to store all website-related static resources, including HTML files, CSS, JavaScript, and product images. S3 was chosen due to its high durability, scalability, and cost-effective storage. This allowed the website content to be securely stored and accessed without the need for managing physical servers.
2. **Amazon CloudFront** was configured with the S3 bucket as its origin to deliver website content to users with low latency. CloudFront uses a global

network of edge locations to cache content closer to users, which significantly improved website loading speed and reduced the load on the origin server.

3. A **Virtual Private Cloud (VPC)** was set up with both public and private subnets to ensure network security and isolation. Public subnets were used for services that needed internet access, while private subnets were used for backend services. This separation improved security by limiting direct access to critical components.
4. **Amazon API Gateway** was deployed to act as the main entry point for all backend requests from the frontend. It handled request routing, authentication, and traffic management. API Gateway enabled secure and scalable communication between the frontend application and backend services.
5. Backend business logic was implemented using **AWS Lambda (or Amazon ECS)** within private subnets. Lambda allowed serverless execution of backend code without managing servers, automatically scaling based on demand. This approach reduced operational overhead and ensured high availability during traffic spikes.
6. **Commercetools APIs** were integrated to manage core e-commerce functionalities such as product catalog, shopping carts, order processing, and payments. Using Commercetools provided a headless commerce solution that allowed ShopSwift to scale and update features independently from the frontend.
7. Customer session data and profile information were stored in **Amazon DynamoDB**, a fully managed NoSQL database service. DynamoDB was selected for its low latency, automatic scaling, and high availability, ensuring fast access to user data even during high traffic periods.
8. **Amazon EventBridge** was configured to publish events whenever a new order was placed. This enabled an event-driven architecture where different services could react to order events independently, improving system flexibility and decoupling services.

9. **Amazon SQS** was used to buffer order messages to ensure reliable processing, even during high load. At the same time, **Amazon SNS** was used to send notifications such as order confirmations to customers. This combination ensured fault tolerance and reliable message delivery.
10. Finally, **Amazon CloudWatch** was enabled to monitor system performance, collect logs, and track metrics across all services. CloudWatch provided visibility into application health, allowed real-time monitoring, and helped in quickly identifying and resolving issues.

Screenshots:

1) S3:

The screenshot shows the AWS S3 console interface. On the left, there's a navigation sidebar with sections like 'Buckets', 'Access management and security', 'Storage management and insights', and 'Account and organization settings'. The main content area is titled 'General purpose buckets' and shows a table with two entries:

Name	AWS Region	Creation date
shopswift-frontend-23069	US East (N. Virginia) us-east-1	December 14, 2025, 18:26:10 (UTC+05:00)
zubnum	US East (N. Virginia) us-east-1	December 14, 2025, 18:27:58 (UTC+05:00)

On the right, there are three informational boxes: 'Account snapshot', 'External access summary - new', and 'Storage Lens provides visibility into storage usage and activity trends'.

2) VPC

The screenshot shows the AWS VPC dashboard. On the left, there's a sidebar with options like VPC dashboard, AWS Global View, Virtual private cloud (Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, NAT gateways, Peering connections, Route servers), and Security (Network ACLs, Security groups). The main area is titled "Your VPCs" and shows two entries:

Name	VPC ID	State	Encryption c...	Encryption control ...	Block Public...	IPv4 CIDR
-	vpc-0336d178b2525f00b	Available	-	-	Off	172.31.16.0/2
shopswift-vpc	vpc-035ab7348922026af	Available	-	-	Off	10.0.1.0/24

Below the table, it says "Select a VPC above".

3) Subnets:

The screenshot shows the AWS VPC dashboard. The sidebar is identical to the previous one. The main area is titled "Subnets (8) Info" and shows eight entries:

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
-	subnet-06dfacd0b7d87ce86	Available	vpc-0336d178b2525f00b	Off	172.31.16.0/2
-	subnet-082f6ff8311a1d275	Available	vpc-0336d178b2525f00b	Off	172.31.80.0/2
-	subnet-05d91bf1fc4ac116	Available	vpc-0336d178b2525f00b	Off	172.31.64.0/2
-	subnet-01dd9af40c3656f2	Available	vpc-0336d178b2525f00b	Off	172.31.0.0/20
-	subnet-08b474276dc0d1c5d	Available	vpc-0336d178b2525f00b	Off	172.31.32.0/2
shopswift-public-subnet	subnet-0d276448e41810603	Available	vpc-035ab7348922026af sho...	Off	10.0.1.0/24

Below the table, it says "Select a subnet".

4) Route Tables:

Route tables (3) Info

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC
shopswift-public-rt	rtb-0be9e8903bc87090e	subnet-0d276448e41810...	-	No	vpc-035ab7348922026af sh...
-	rtb-011d4911c1418df82	-	-	Yes	vpc-0336d178b2525f00b
-	rtb-0d2968c9cff7169e7	-	-	Yes	vpc-035ab7348922026af sh...

Select a route table

5) Internet Gateway:

Internet gateways (2) Info

Name	Internet gateway ID	State	VPC ID	Owner
shopswift-igw	igw-0148de7d8b79c675e	Attached	vpc-035ab7348922026af shopswift-vpc	680754311801
-	igw-017c94f894f32c4ea	Attached	vpc-0336d178b2525f00b	680754311801

Select an internet gateway above

6) Security Groups:

Security Groups (3) Info

Name	Security group ID	Security group name	VPC ID	Description
-	sg-04f7932cf77ae2e7c	default	vpc-0336d178b2525f00b	default VPC security
-	sg-0112749070323bb45	launch-wizard-1	vpc-035ab7348922026af	launch-wizard-1 cre...
-	sg-006bb7cb3d49733f7	default	vpc-035ab7348922026af	default VPC security

7) Lambda

Lambda

Functions (6)

Function name	Description	Runtime	Last modified
RedshiftEventSubscription	Create Redshift event subscription to SNS Topic.	Python 3.10	4 weeks ago
MainMonitoringFunction	-	Python 3.10	4 weeks ago
ModLabRole	updates LabRole to allow it to assume itself	Python 3.10	4 weeks ago
RoleCreationFunction	Create SLR if absent	Python 3.10	4 weeks ago
RedshiftOverwatch	Deletes Redshift Cluster if the count is more than 2.	Python 3.10	4 weeks ago
ShopSwift-Backend	-	Node.js 20.x	2 weeks ago

8) Tables (DynamoDB):

DynamoDB

Tables (1) Info

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity	Write capacity
ShopSwift-UserData	Active	userid (\$)	-	0	0	Off	☆	On-demand	

9) EC2 Instances:

EC2

Instances (1) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
ShopSwift-We...	i-0750f674c4792d2c7	Running	t3.micro	3/3 checks passed	View alarms	us-east-1a	-

10) SNS

Amazon SNS

Topics (2)

Name	Type	ARN
RedshiftSNS	Standard	arn:aws:sns:us-east-1:680754311801:RedshiftSNS
ShopSwift-Alerts	Standard	arn:aws:sns:us-east-1:680754311801:ShopSwift-Alerts

11) Event Bridge:

The screenshot shows the 'Schedule groups' section of the Amazon EventBridge console. On the left, there's a sidebar with links like 'Dashboard', 'Developer resources' (Learn, Sandbox, Quick starts), and 'Buses' (Event buses, Rules). The main area is titled 'Schedule groups (1)' and contains a table with one row. The row shows a checkbox, the name 'default', its status as 'Active', and its ARN as 'arn:aws:scheduler:us-east-1:680754311801:schedule-group/default'. There are buttons for 'Edit', 'Delete', and 'Create schedule group' at the top right.

12) Final Output

The screenshot shows a product listing page from the ShopSwift website. The header includes a logo, navigation links for 'Shop' and 'Admin', and an 'ADMIN ACCESS' button. The main content displays a grid of four products: 'Starter Motor' (MOTOR, \$800.00), 'Air Filter' (FILTER, \$60000.00), 'A789 BC Deep Mining Excavator' (BC, \$890000.00), and 'Pin and Bushing Kit' (AND, \$1500.00). Below this grid, there are two more product cards partially visible: a coiled hose and a small orange vehicle.

Security & IAM:

Security best practices were followed throughout the system. IAM roles were configured using the principle of least privilege, VPC isolation was implemented with public and private subnets, and outbound access was controlled using a NAT Gateway. Logging and monitoring were enabled through CloudWatch, and data access was secured to ensure reliability and compliance.