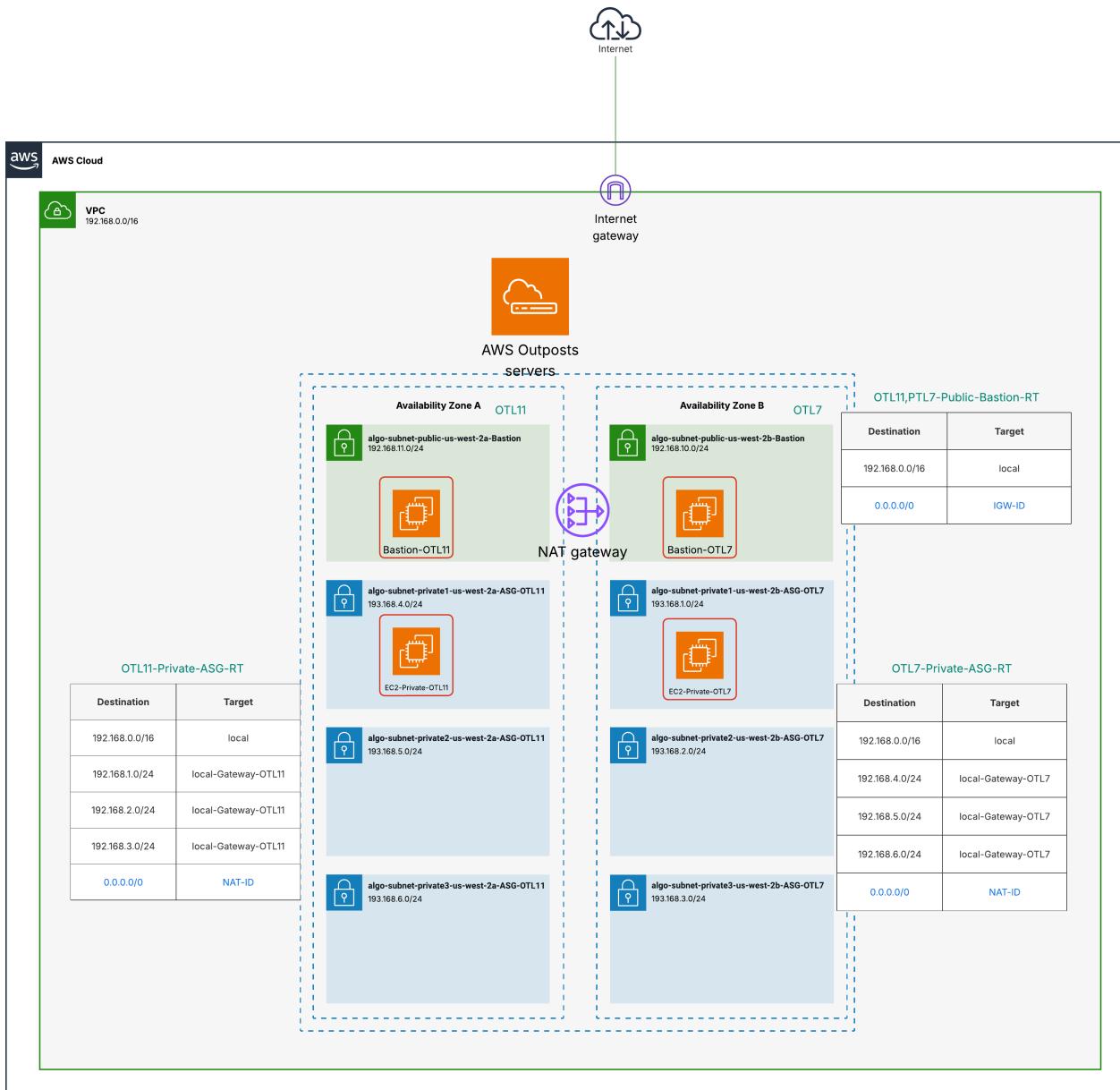


Done Documentation Cloud (Master) & HA Worker Nodes on Outposts ✓

Outpost Architecture Overview ✓



Connection between 2 AWS Outposts

Communication Status between 2 Outposts

lgw-rtb-00fce8ca84c141ba9

Find LGW routes			
CIDR	Target	Route status	Type
0.0.0.0/0	lgw-vif-grp-01b2085d5fea05f73	Active	static
10.77.11.0/24	lgw-vif-grp-01b2085d5fea05f73	Active	static
10.78.11.0/24	lgw-vif-grp-01b2085d5fea05f73	Active	static
192.168.3.0/24	subnet-0bee68aa87ffa762b	Active	propagated
192.168.1.0/24	subnet-0ad247a6ca814ac0a	Active	propagated
192.168.2.0/24	subnet-0f1ef6f86539863eb	Active	propagated

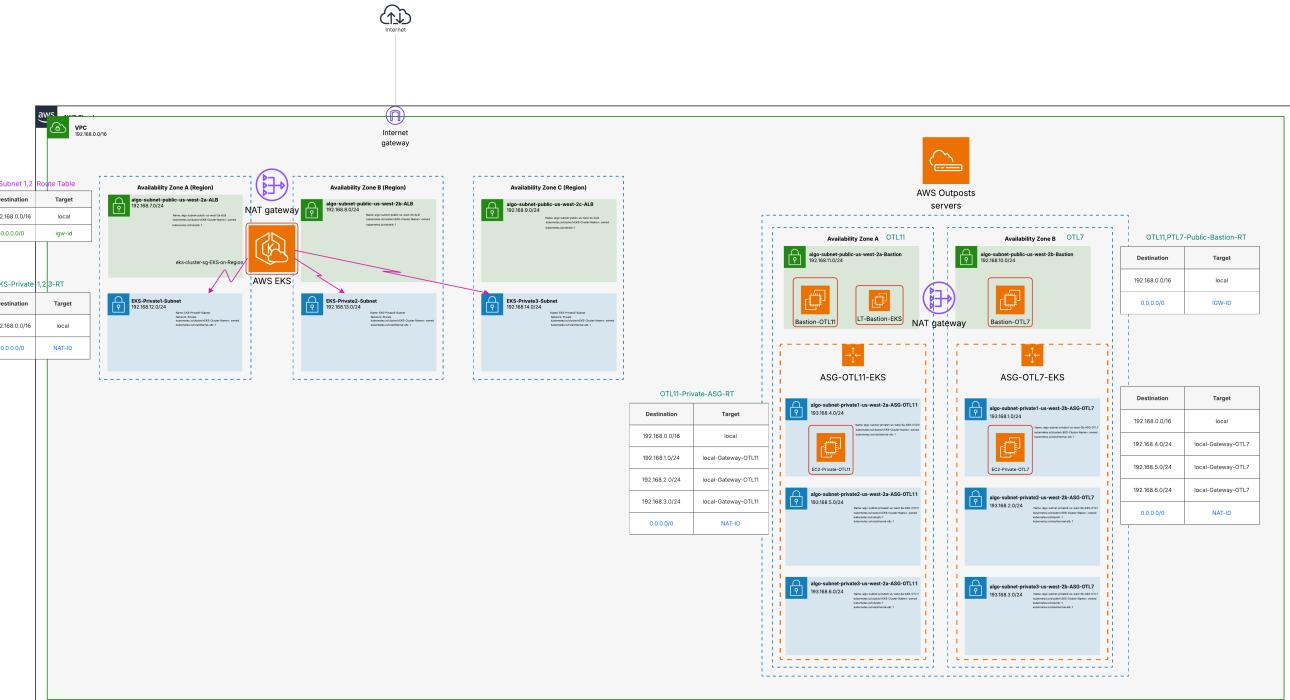
lgw-rtb-07d2eb0deaef88456b

Clear filters			
CIDR	Target	Route status	Type
0.0.0.0/0	lgw-vif-grp-0407a60df52242eab	Active	static
10.77.3.0/24	lgw-vif-grp-0407a60df52242eab	Active	static
10.77.7.0/24	lgw-vif-grp-0407a60df52242eab	Active	static
10.78.7.0/24	lgw-vif-grp-0407a60df52242eab	Active	static
192.168.5.0/24	subnet-0766bbadb27e054d1	Active	propagated
192.168.4.0/24	subnet-09265fab85d4063f8	Active	propagated
192.168.6.0/24	subnet-0af02ab05d77a78c6	Active	propagated

Final Configuration

- 3 Private Subnet on each AWS Outpost
- Route between them success & they can communicate with each other
- Test connection between 2-Outposts through (Bastion - Ping)

Amazon EKS Cluster (Region) Architecture Overview ✅



Amazon EKS Cluster Status ✅

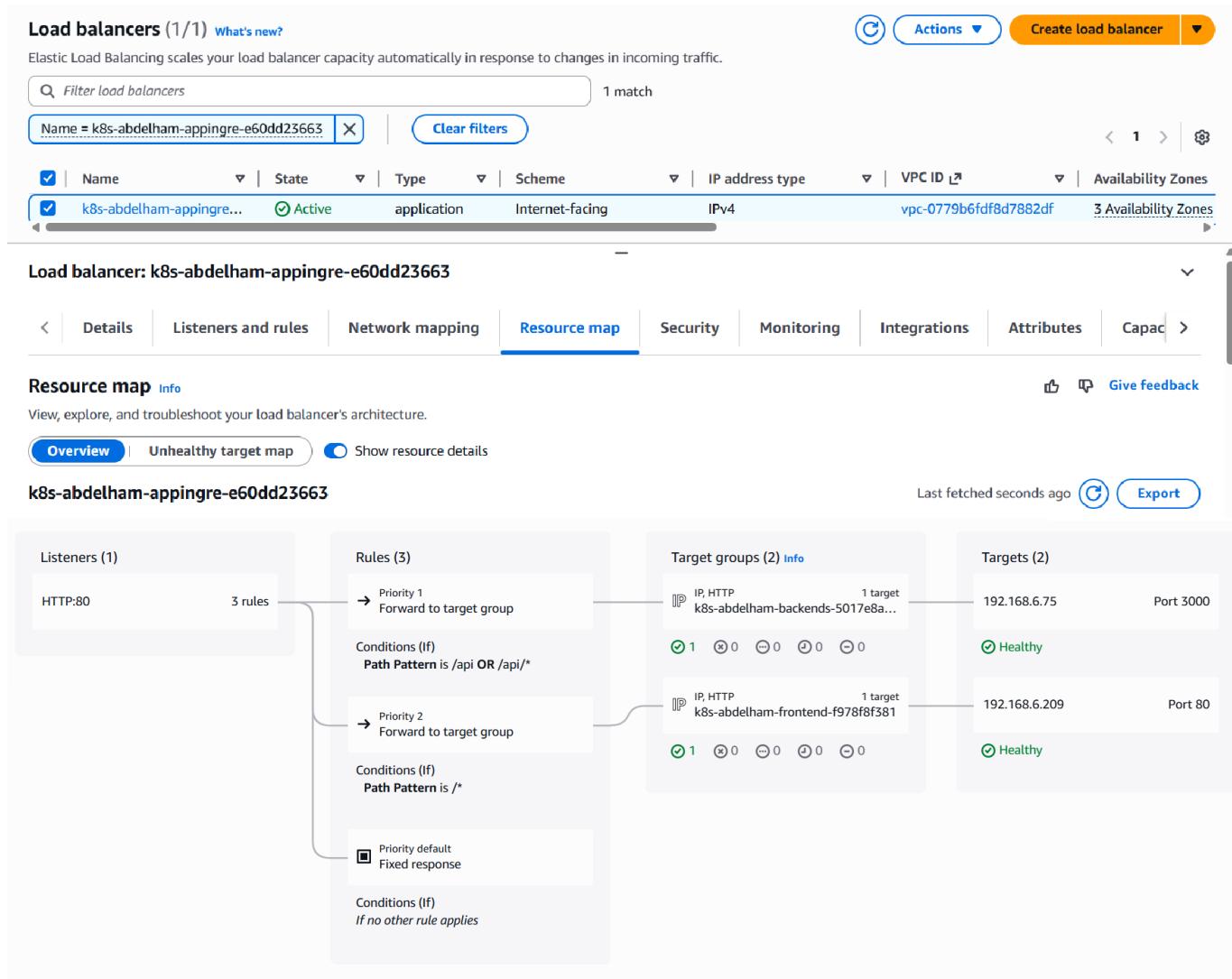
The screenshot shows the AWS EKS Cluster Status page for the 'EKS-on-Region' cluster. The left sidebar provides navigation to various services like Amazon ECR, AWS Batch, and Documentation. The main content area displays cluster information, including status, Kubernetes version (1.35), support period (Standard support until March 27, 2027), and provider (EKS). It also shows cluster health (0 issues), upgrade insights (4 green, 1 yellow), node health issues (0 issues), and capability issues (0 issues). The 'Compute' tab is selected, showing a table of nodes (6 total) with columns for Node name, Instance type, Compute provider, Managed by, Created, and Status. All nodes are listed as 'Ready'.

Node name	Instance type	Compute	Managed by	Created	Status
ip-192-168-3-136.us-west-2.compute.internal	c5.xlarge	Self-managed	-	January 19, 2026, 10:47 (UTC+02:00)	Ready
ip-192-168-3-139.us-west-2.compute.internal	c5.xlarge	Self-managed	-	January 19, 2026, 10:47 (UTC+02:00)	Ready
ip-192-168-3-26.us-west-2.compute.internal	c5.xlarge	Self-managed	-	January 19, 2026, 10:48 (UTC+02:00)	Ready
ip-192-168-6-14.us-west-2.compute.internal	m5.xlarge	Self-managed	-	January 19, 2026, 10:47 (UTC+02:00)	Ready
ip-192-168-6-248.us-west-2.compute.internal	m5.xlarge	Self-managed	-	January 19, 2026, 10:48 (UTC+02:00)	Ready

Final Configuration ✅

- ✓ 3 Private Subnets on region for Amazon EKS Control Plane
- ✓ Create 3 Launch Templates
- ✓ Create 2 Auto Scaling Groups
- ✓ Run Worker Nodes on AWS outposts

Deploy 3-tier applications Architecture Overview ✅



💻 **Check My Repo to see the files that are used to deploy the 3-tier applications**

- GitHub Repository

Application Access Status

```
[ec2-user@ip-192-168-11-111 ~]$ kubectl get all -n abdelhamid-ns
NAME                                         READY   STATUS    RESTARTS   AGE
pod/backend-5ccf4f44f6-xmb5f   1/1     Running   0          5d8h
pod/frontend-5d9cc9f7c5-nv5x9  1/1     Running   0          4d
pod/mongo-0                           1/1     Running   0          5d14h

NAME                TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)        AGE
service/backend-service ClusterIP  10.100.6.179   <none>           3000/TCP      5d8h
service/frontend-service NodePort   10.100.42.192  <none>           80:32567/TCP  5d14h
service/mongo-service   ClusterIP  None           <none>           27017/TCP      5d14h

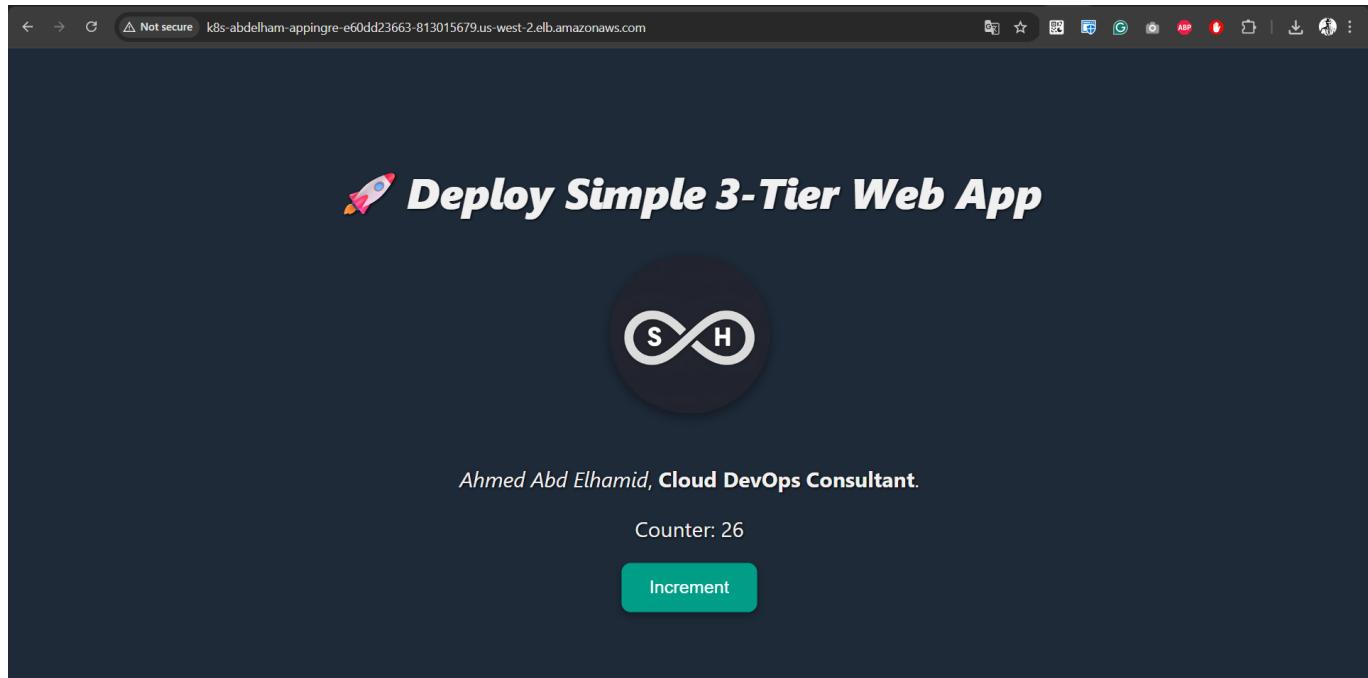
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/backend  1/1     1           1           5d8h
deployment.apps/frontend 1/1     1           1           5d14h

NAME            DESIRED  CURRENT  READY   AGE
replicaset.apps/backend-5ccf4f44f6  1        1        1       5d8h
replicaset.apps/frontend-5d9cc9f7c5 1        1        1       4d

NAME          READY   AGE
statefulset.apps/mongo  1/1     5d14h
[ec2-user@ip-192-168-11-111 ~]$ [ec2-user@ip-192-168-11-111 ~]$ [ec2-user@ip-192-168-11-111 ~]$ kubectl get ingress -n abdelhamid-ns
NAME          CLASS  HOSTS      ADDRESS          PORTS  AGE
app-ingress  alb    *          k8s-abdelham-appingre-e60dd23663-813015679.us-west-2.elb.amazonaws.com  80      5d8h
[ec2-user@ip-192-168-11-111 ~]$
```

i-077825c693e3953f1 (Bastion-EKS)

Public IPs: 35.87.148.136 Private IPs: 192.168.11.111



Final Configuration

- Successfully deployed a 3-tier application (Frontend, Backend, Database) on Amazon EKS.
- Verified persistence via EBS CSI Driver for MongoDB.
- Automated Application Load Balancer (ALB) provisioning and traffic routing.
- Verified end-to-end connectivity from Internet Gateway to Worker Nodes on Outposts.

Project Implementation Status

Component	Task Description	Status
Infrastructure	Hybrid VPC Configuration (Region + Outposts)	<input checked="" type="checkbox"/> Done
Networking	Connectivity between OTL11 and OLT7 via LGW	<input checked="" type="checkbox"/> Done
EKS Cluster	Management Plane deployed on AWS Region	<input checked="" type="checkbox"/> Done
Worker Nodes	Self-managed nodes joined from Outposts	<input checked="" type="checkbox"/> Done
Auto Scaling	ASG & Launch Templates for Outpost Nodes	<input checked="" type="checkbox"/> Done
Storage	EBS CSI Driver integration for MongoDB persistence	<input checked="" type="checkbox"/> Done
Traffic Control	AWS Load Balancer Controller (ALB Ingress)	<input checked="" type="checkbox"/> Done
Security	IAM Roles for Service Accounts (IRSA) via OIDC	<input checked="" type="checkbox"/> Done
Application	3-Tier Stack (React, Node.js, MongoDB)	<input checked="" type="checkbox"/> Done

Key Achievements:

- ❶ Achieved sub-millisecond latency for local service communication via AWS Outposts.
- ❷ Implemented seamless hybrid connectivity between Region-based EKS Control Plane and Outpost-based Worker Nodes.
- ❸ Automated ALB provisioning and Target Group registration using AWS Load Balancer Controller.