

Simple 3-Tier Web Application

This project is a simple 3-tier web application with a static frontend and a Node.js backend. The frontend is served by Nginx, while the backend is built with Express.js.

Project Structure

```
simple-3-tier-app/
├── assets/          # Contains project images
├── Backend/         # Node.js (Express) API
├── FrontEnd/        # Static HTML served by Nginx
├── K8s-Local/        # Deploy K8s through minikube
├── K8s-EKS-Region/   # Deploy k8s manifests through Amazon EKS
└── docker-compose.yml
    └── README.md
```

Test Local firstly

to test locally

```
Deploy simple-3-tier-app locally

Backend/
  .env
    PORT=3000
    // MONGO_URI=mongodb://mongo-service:27017/counterdb # EKS
    MONGO_URI=mongodb://localhost:27017/counterdb           # Local

  app.js
  npm install express cors mongoose dotenv
    node_modules/
      package.json

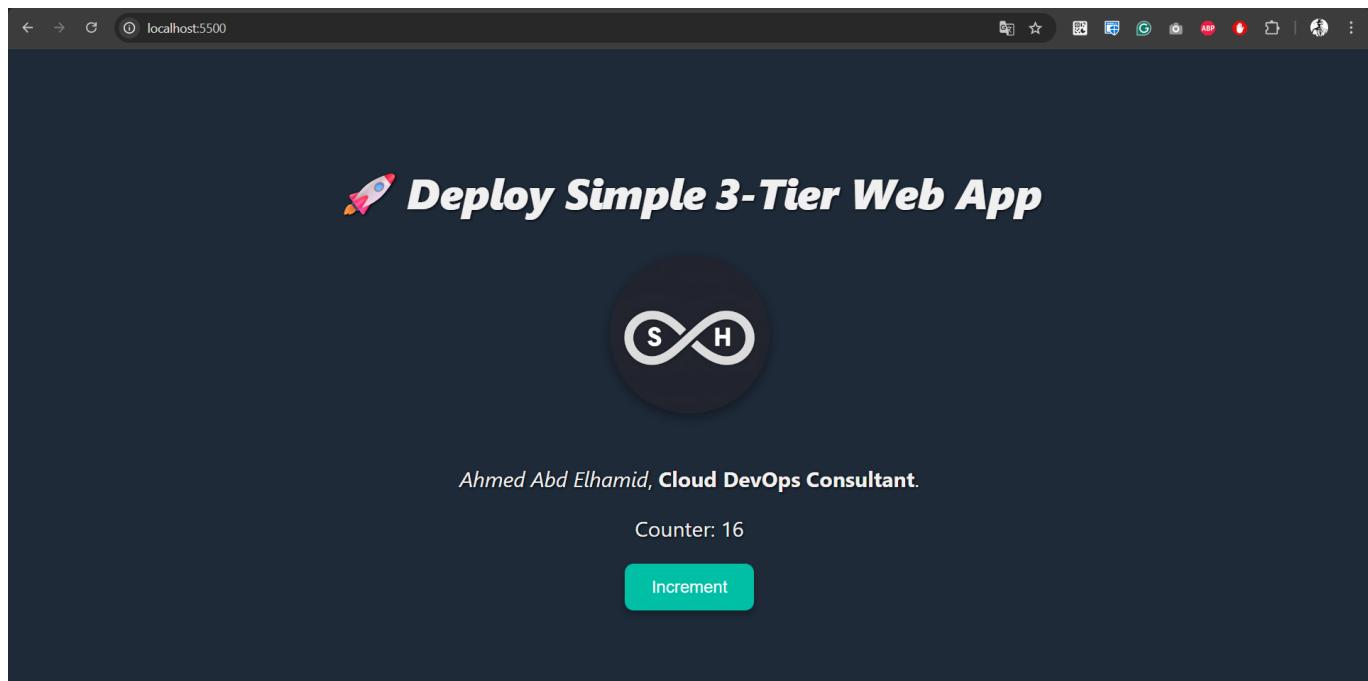
  # Run MongoDB locally using Docker
  docker run -d -p 27017:27017 --name mongo mongo

  # Run Backend API
  node app.js

FrontEnd/
  index.html
    // const backendUrl = "/api/increment";                  # EKS /

Ingress
```

```
|     |     const backendUrl = "http://localhost:3000/api/increment"; # Local
testing
|     |
| # Run Frontend server
└─ python -m http.server 5500
```



📄 Prepare Dockerfile

📁 Backend Structure

```
Backend/
├── app.js
├── package.json
├── package-lock.json (.gitignore)
├── node_modules (.gitignore)
├── .env
└── Dockerfile
```

📄 Dockerfile

```
FROM node:20.11.1-alpine3.19
WORKDIR /app
COPY package*.json ./
RUN npm install --only=production
COPY . .
# Expose backend port
EXPOSE 3000
CMD ["node", "app.js"]
```

```
docker build -t backend-app .
```

📁 Frontend Structure

```
FrontEnd/
├── index.html
├── Dockerfile
└── nginx.conf  (import for Ingress/api to work well)
```

🏠 📄 nginx.conf

```
server {
    listen 80;

    location / {
        root /usr/share/nginx/html;
        index index.html;
        try_files $uri /index.html;
    }

    location /api/ {
        #(backend:3000) → Name of Backend Service in "Kubernetes"
        proxy_pass http://backend:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

📄 Dockerfile

```
FROM nginx:alpine

# Remove default config
RUN rm /etc/nginx/conf.d/default.conf

# Copy custom nginx config
COPY nginx.conf /etc/nginx/conf.d/default.conf

COPY index.html /usr/share/nginx/html/index.html

EXPOSE 80
```

```
docker build -t frontend-app .
```

👉 Test locally through "Docker"

```
# Create Network
docker network create app-net

docker run -d --name mongo --network app-net mongo

// Container Name of "backend-app" image must be the same name in "nginx.conf" -->
http://backend:3000
docker run -d --name backend --network app-net -e
MONGO_URI=mongodb://mongo:27017/counterdb backend-app

docker run -d --name frontend --network app-net -p 8000:80 frontend-app
```

```

cmd
C:\Users\ahmed (master)
\ docker images
IMAGE
backend-app:latest
frontend-app:latest
gcr.io/k8s-minikube/kicbase:v0.0.46
gcr.io/k8s-minikube/kicbase@sha256:fd2d45ddcc33ebc5cb68a17e6219ea207ce63c005095ea1525296da2d1a279
mongo:latest
portainer/portainer-ce:lets

C:\Users\ahmed (master)
\ docker network create app-net
a1520fb0d27860bdcb183fc28e696baa77d2af8798cc5cfb4e87c56782523de

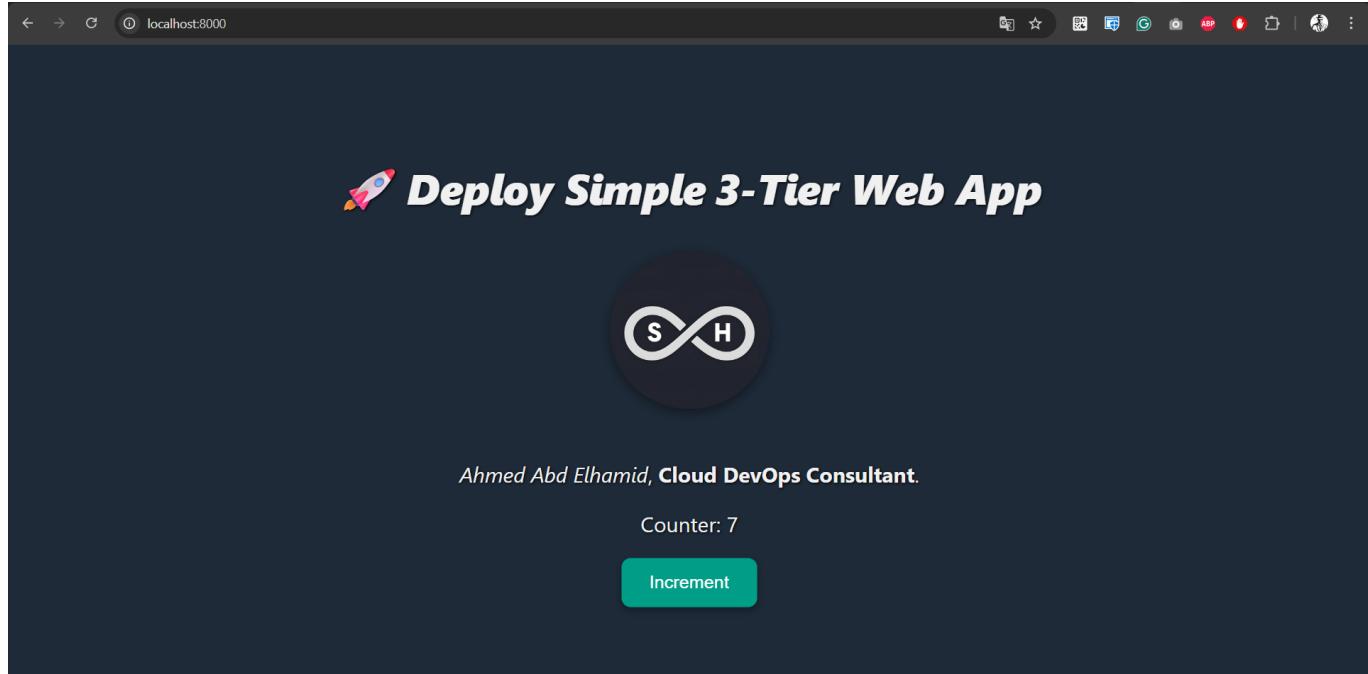
C:\Users\ahmed (master)
\ docker run -d --name mongo --network app-net mongo
fc2f9fba236027affc9fc6a50b30443f2af0f13b9065bdf25425e3e30b7846fe

C:\Users\ahmed (master)
\ docker run -d --name backend --network app-net -e MONGO_URI=mongodb://mongo:27017/counterdb backend-app
a632c347f9c1a2e43ae127ddefbb0e109205db614bd6517e688d0c39a5dbcfc3f

C:\Users\ahmed (master)
\ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
d2a3eab58452 frontend-app "/docker-entrypoint..." 6 seconds ago Up 5 seconds 0.0.0.0:8000->80/tcp, [::]:8000->80/tcp frontend
a632c347f9c1 backend-app "docker-entrypoint.s..." 16 seconds ago Up 15 seconds 3009/tcp backend
fc2f9fba2360 mongo "docker-entrypoint.s..." 35 seconds ago Up 34 seconds 27017/tcp mongo
fe024ee9675a portainer/portainer-ce:lets "/portainer" 3 weeks ago Up 3 hours 0.0.0.0:9000->9000/tcp, [::]:9000->9000/tcp portainer

C:\Users\ahmed (master)
}

```



🏷️🚀 Tag & Push image

```

# Tag Image
docker tag <Image-Name> <Docker_Hub-User>/<Image-Name>
# Push Image
docker push <Docker_Hub-User>/<Image-Name>

```

```

cmd
C:\Users\ahmed (master)
  docker images
  IMAGE
  backend-app:latest
  frontend-app:latest
  gcr.io/k8s-minikube/kicbase:v0.0.46
  gcr.io/k8s-minikube/kicbase@sha256:fd2d445ddcc33ebc5c6b68a17e6219ea207ce63c005095ea1525296da2d1a279
  mongo:latest
  portainer/portainer-ce:ltls
  C:\Users\ahmed (master)
  docker tag frontend-app ahmed1399/frontend-app
  C:\Users\ahmed (master)
  docker tag backend-app ahmed1399/backend-app
  C:\Users\ahmed (master)
  docker images
  IMAGE
  ahmed1399/backend-app:latest
  ahmed1399/frontend-app:latest
  backend-app:latest
  frontend-app:latest
  gcr.io/k8s-minikube/kicbase:v0.0.46
  gcr.io/k8s-minikube/kicbase@sha256:fd2d445ddcc33ebc5c6b68a17e6219ea207ce63c005095ea1525296da2d1a279
  mongo:latest
  portainer/portainer-ce:ltls
  C:\Users\ahmed (master)
  docker push ahmed1399/frontend-app:latest
  The push refers to repository [docker.io/ahmed1399/frontend-app]
  f7ddf42b6dce: Pushed
  9bf7fa179405: Pushed
  e0965404205d: Pushed
  z66fab40e05e: Pushed
  25f4530d4fd3: Pushed
  567f84da6fb0: Pushed
  2a946e2bc7ff: Pushed
  da7c973d8b92: Pushed
  33f95aa0f3229: Pushed
  5d962fb41d5d: Pushed
  085c5e3aa8e6: Pushed
  0abf9e567266: Pushed
  1074353ee0d: Pushed
  latest: digest: sha256:9132d2fb15198384dec560f5e4724e81fd57793742fcda6404be7d8582c08d8 size: 856
  C:\Users\ahmed (master)
  docker push ahmed1399/backend-app:latest
  The push refers to repository [docker.io/ahmed1399/backend-app]
  092226d52cac: Pushed
  5da03fa05510: Pushed
  331c57194f4d: Pushed
  4abcf2066143: Pushed
  8f22cf2c7d81: Pushed
  07060fc08b64: Pushed
  0493dfb2ff9c: Pushed
  6a17ab3e173e: Pushed
  9f16480e2ff5: Pushed
  latest: digest: sha256:22ca300c2122b519d4e1e28f4bf6db12c6b9f3989734acd2c1b82ee056911389 size: 856
  C:\Users\ahmed (master)
  cmd.exe

```

Repositories

All repositories within the ahmed1399 namespace.

Name	Last Pushed	Contains	Visibility	Scout
ahmed1399/backend-app	2 minutes ago	IMAGE	Public	Inactive
ahmed1399/frontend-app	3 minutes ago	IMAGE	Public	Inactive

Prepare Kubernetes manifest files

You can check (K8s manifest files) from here [EKS Files](#)

Deploy or Remove using:

```

#!/bin/bash

kubectl apply -f namespace.yaml
kubectl apply -f mongodb.yaml
kubectl apply -f backend.yaml
kubectl apply -f frontend-configmap.yaml
kubectl apply -f frontend.yaml

# kubectl delete -f mongodb.yaml
# kubectl delete -f backend.yaml

```

```
# kubectl delete -f frontend-configmap.yaml
# kubectl delete -f frontend.yaml
# kubectl delete -f namespace.yaml
```

```
minikube service frontend-service -n abdelhamid-ns
```

```
MINGW64 /d/End/Sheetssssssss/ITI-Materials/AWS SAA/New folder/EKS-Cluster/simple-3-tier-app/K8s
$ kubectl get deployment -n abdelhamid-ns -o yaml
NAME      READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES          SELECTOR
backend   1/1     1            1           39m   backend       ahmed1399/backend-app:latest   app=backend
Frontend  1/1     1            1           39m   frontend      ahmed1399/frontend-app:latest   app=frontend

$ kubectl get pods -n abdelhamid-ns -o yaml
NAME        READY   STATUS    RESTARTS   AGE   IP           NODE   NOMINATED NODE   READINESS GATES
backend-54dd97c-hfsrf  1/1   Running   1 (38m ago)   39m  10.244.0.17  minikube <none>           <none>
Frontend-598f97ff75-xlrb5  1/1   Running   0           35m  10.244.0.19  minikube <none>           <none>
mongo-0    1/1   Running   0           39m  10.244.0.18  minikube <none>           <none>

$ kubectl get statefulset -n abdelhamid-ns -o yaml
NAME      AGE   CONTAINERS   IMAGES
mongo     1/1   40m         mongo:6.0

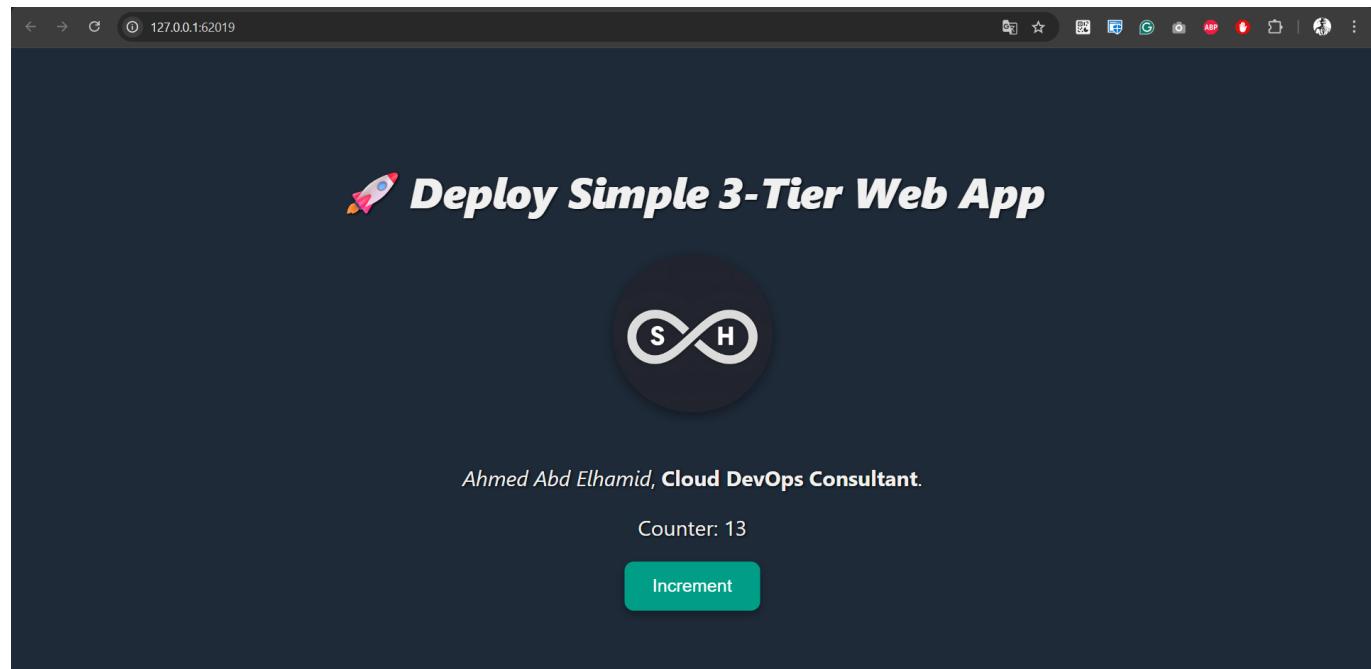
$ kubectl get service -n abdelhamid-ns -o yaml
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE   SELECTOR
backend-service  ClusterIP      10.107.1.196   <none>        3000/TCP    40m   app=backend
Frontend-service NodePort      10.106.156.92  <none>        80:31052/TCP 40m   app=Frontend
mongo-service   ClusterIP      None          <none>        27017/TCP   40m   app=mongo

$ minikube service Frontend-service -n abdelhamid-ns
```

NAMESPACE	NAME	TARGET PORT	URL
abdelhamid-ns	frontend-service	80	http://192.168.49.2:31052

NAMESPACE	NAME	TARGET PORT	URL
abdelhamid-ns	frontend-service		http://127.0.0.1:62019

* Starting tunnel for service frontend-service.
* Starting tunnel for service frontend-service.
* Opening service abdelhamid-ns/frontend-service in default browser...
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.



```

cmd
D:\End\Sheetssssssss\ITI-Materials\AWS SAA\New folder\EKS-Cluster\simple-3-tier-app\K8s
└ kubectl get all -n abdelhamid-ns
  NAME           READY   STATUS    RESTARTS   AGE
  pod/backend-54ddd97c-hfsrf   1/1    Running   1 (83m ago)   84m
  pod/frontend-598f97ff75-xlrb5  1/1    Running   0          80m
  pod/mongo-0      1/1    Running   0          84m

  NAME        TYPE    CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
  service/backend-service ClusterIP  10.107.1.196 <none>       3000/TCP   84m
  service/frontend-service NodePort   10.106.156.92 <none>       80:31052/TCP 84m
  service/mongo-service   ClusterIP  None          <none>       27017/TCP   84m

  NAME        READY   UP-TO-DATE   AVAILABLE   AGE
  deployment.apps/backend  1/1        1           1           84m
  deployment.apps/frontend 1/1        1           1           84m

  NAME        DESIRED  CURRENT  READY   AGE
  replicaset.apps/backend-54ddd97c  1        1       1   84m
  replicaset.apps/frontend-598f97ff75  1        1       1   80m
  replicaset.apps/frontend-77b67f4979  0        0       0   84m

  NAME        READY   AGE
  statefulset.apps/mongo  1/1     84m

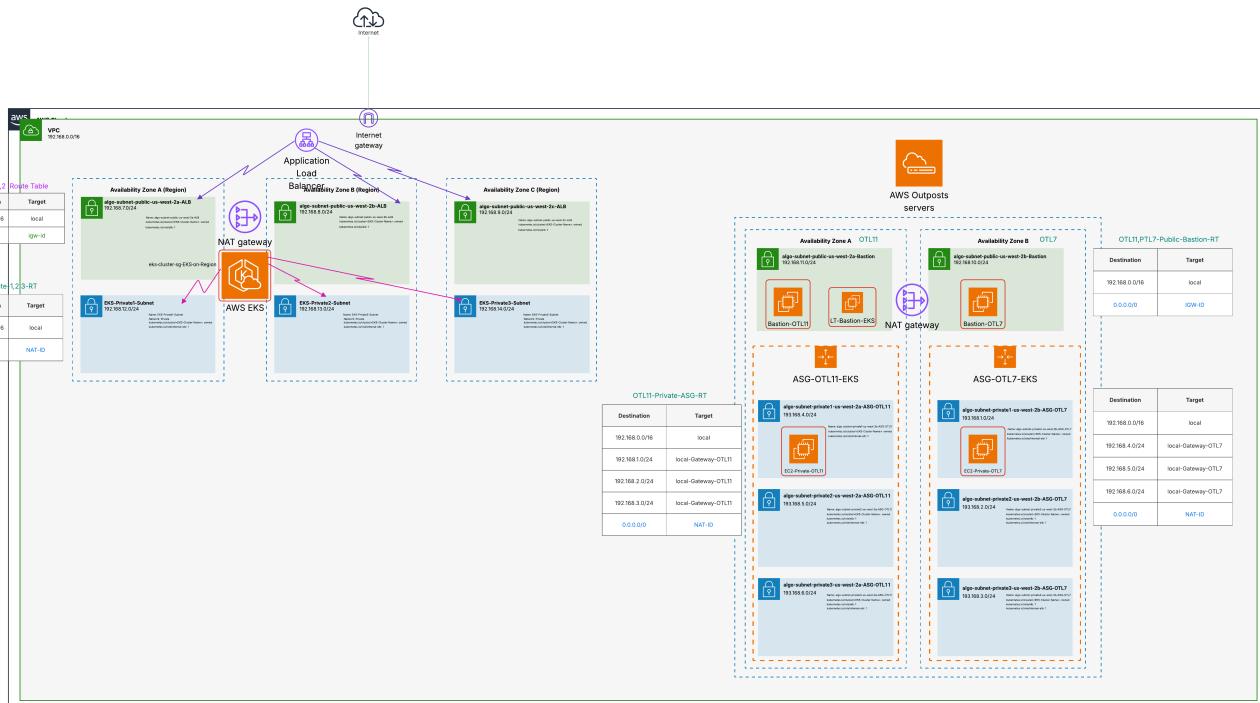
D:\End\Sheetssssssss\ITI-Materials\AWS SAA\New folder\EKS-Cluster\simple-3-tier-app\K8s
└ kubectl get ingress -n abdelhamid-ns
  NAME   CLASS  HOSTS   ADDRESS      PORTS   AGE
  app-ingress  nginx  app.local  192.168.49.2  80   18m

D:\End\Sheetssssssss\ITI-Materials\AWS SAA\New folder\EKS-Cluster\simple-3-tier-app\K8s
└

```

Production (EKS)

You can check (K8s manifest files) from here: [EKS Files](#)



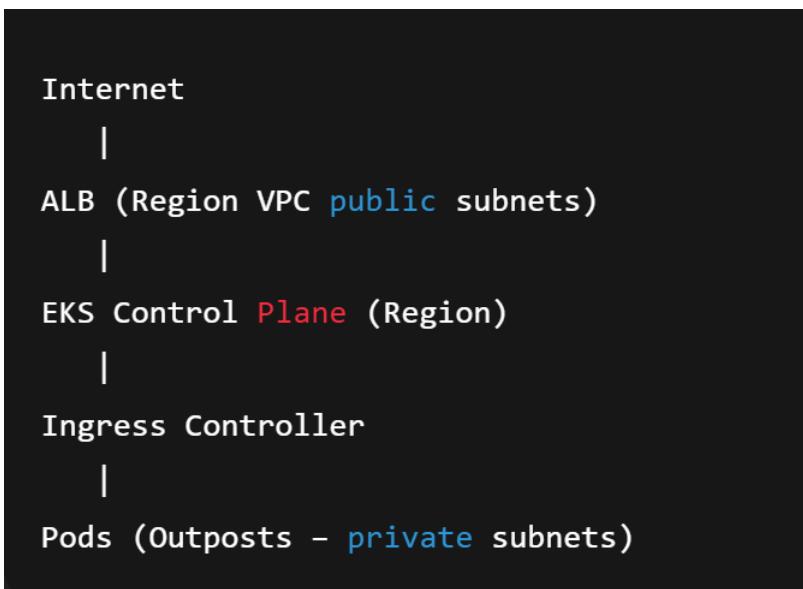
Deployment, Service, Ingress

Workflows

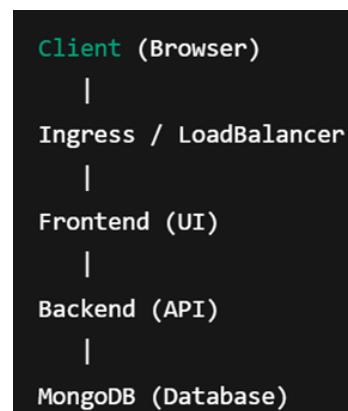
ALB Workflow

3-Tier Application Workflow

ALB Workflow



3-Tier Application Workflow



- from within **Bastion Host**

- Firstly, create a **namespace**
 - Create **Deployment & Service**
 - Create **Ingress**
 - `namespace.yaml`
 - `mongodb.yaml`
 - To save data in MongoDB persistently, we will use the **Amazon EBS CSI Driver**
 - Some problems will face you, so to solve them. Check the steps to solve the problem with the **Amazon EBS CSI Driver (Deprecated)**.
 - `backend.yaml`
 - `frontend-configmap.yaml`, `frontend.yaml`
 - `ingress.yaml`
 - Amazon EBS CSI Driver CrashLoopBackOff (mongodb) (Deprecated)**
 - `ebs-csi-controller-sa`

```

# The problem that will face you
kubectl get pods -n kube-system | grep ebs    # **CrashLoopBackOff**

# **Create IAM Role** & attach it with Service Account to talk with EBS &
create volume
eksctl create iamserviceaccount \
--cluster <EKS-Cluster-Name> \
--namespace kube-system \
--name **ebs-csi-controller-sa** \
--role-name ebs-csi-controller-sa-abdelhamid \
--attach-policy-arn arn:aws:iam::aws:policy/service-
role/AmazonEBSCSIDriverPolicy \
  
```

```
--region <Region> \
--approve \
--override-existing-serviceaccounts

# Ensure that **Service Account** exist
kubectl get sa **ebs-csi-controller-sa** -n kube-system

# Display **Role ARN**
role_arn=$(aws iam get-role --role-name ebs-csi-controller-sa-abdelhamid --query 'Role.Arn' --output text)
# Update add-ons with **new Role**
aws eks update-addon \
--cluster-name <EKS-Cluster-Name> \
--addon-name ***aws-ebs-csi-driver*** \
--service-account-role-arn $role_arn \
--region <Region>

# Ensure that **Add-ons active**
aws eks describe-addon --cluster-name <EKS-Cluster-Name> --addon-name
***aws-ebs-csi-driver*** --region <Region> --query "addon.status"

# to accelerate the process
kubectl delete pod -n kube-system -l app.kubernetes.io/name=aws-ebs-csi-
driver

# Ensure that **pod of ebs** **run**
# ebs-csi-controller, ebs-csi-node (**Run**)
kubectl get pods -n kube-system | grep ebs

# persistentVolume(PV) will be created automatically
kubectl get pv
# Bound
kubectl get pvc -n <namespace-name>
kubectl get storageclass
```

Volumes (1/1) Info		Last updated 1 minute ago	Actions	Create volume
Saved filter sets	Choose filter set	Search		
vol-08d1811a8422524d4	X	Clear filters		
Name	Volume ID	Type	Size	IOPS
EKS-on-Region-dynamic-pvc-a10b7349-a02c-411d-bb75-8933b720af2c	vol-08d1811a8422524d4	gp2	1 GiB	100

The screenshot shows the AWS EKS console with the 'Amazon EBS CSI Driver' configuration page. It includes sections for 'Category' (storage), 'Status' (Active), 'Version' (v1.54.0-eksbuild.1), 'EKS Pod Identity' (Not set), and 'IAM role for service account (IRSA)' (arn:aws:iam::22704829:3427:role/ebs-csi-controller-sa-abdelhamid). A 'View in IAM' link is also present.

- If you face a problem during activation **Amazon EBS CSI Driver** install **Amazon EBS CSI Driver** through **Manual manifest files (for full control)**

```
# -----> Install EBS-CSI-Driver through **manifest**  
  
# **Create IAM Role** & attach it with Service Account to talk with EBS &  
create volume  
eksctl create iamserviceaccount \  
  --cluster <EKS-Cluster-Name> \  
  --namespace kube-system \  
  --name **ebs-csi-controller-sa** \  
  --role-name *My-EBS-CSI-Role* \  
  --attach-policy-arn arn:aws:iam::aws:policy/service-  
role/AmazonEBSCSIDriverPolicy \  
  --region <Region> \  
  --approve \  
  --override-existing-serviceaccounts  
  
# Install the **Driver** "***ebs-csi-controller-sa***" automatically  
kubectl apply -k "github.com/kubernetes-sigs/aws-ebs-csi-  
driver/deploy/kubernetes/overlays/stable/?ref=release-1.34" --server-side  
  
-----  
# **Regional STS**  
# Solve the problem of **STS** → **Driver** can't talk with **Global STS  
endpoint**  
# **5 Sidecars containers** (ebs-plugin, csi-provisioner, csi-attacher, csi-  
snapshotter, csi-resizer)  
****# Important for remote region  
for container in **ebs-plugin** **csi-provisioner** **csi-attacher** **csi-  
snapshotter** **csi-resizer**; do  
  kubectl set env deployment/ebs-csi-controller \  
    -n kube-system \  
    -c $container \  
    AWS_STS_REGIONAL_ENDPOINTS=regional  
done  
-----  
  
# Rerun pods  
kubectl rollout restart deployment ebs-csi-controller -n kube-system  
  
# If still pending "IAM Role has problem in **Trust Policy**"  
# must contain → **OIDC Provider**, **Name of** ***Service Account***  
aws iam get-role --role-name *My-EBS-CSI-Role* --query  
'Role.AssumeRolePolicyDocument'  
  
# Ensure that **pod of ebs** **run**  
# ebs-csi-controller, ebs-csi-node (**Run**)  
kubectl get pods -n kube-system | grep ebs  
  
# persistentVolume(PV) will be created automatically  
kubectl get pv  
# Bound  
kubectl get pvc -n <namespace-name>  
kubectl get storageclass
```

```
A newer release of "Amazon Linux" is available.
Version 2023.10.20260120:
Run "/usr/bin/dnf check-release-update" for full release and version update info
      _#_
     /###\
    / \##\ Amazon Linux 2023
   /~ \##\ https://aws.amazon.com/linux/amazon-linux-2023
  /~ \##\ V-+-->
 /~ \##\ /m/
Last login: Sun Feb 1 14:00:25 2026 from 18.237.140.165
[ec2-user@ip-192-168-11-111 ~]$ kubectl get pv
NAME          CAPACITY   ACCESS MODES  RECLAIM POLICY  STATUS   CLAIM
SS  REASON  AGE
pvc-27c0cf66-655c-4b5e-87a0-13c2fc019137  5Gi       RWO        Retain       Bound    abdelhamid-ns/mongo-storage-mongo-0  ebs-sc  <unset>
      5d6h
pvc-80af529f-513a-4378-802f-54b17005c1bc  1Gi       RWO        Delete       Bound    kube-system/mongo-storage-mongo-0  ebs-sc  <unset>
      6d1h
[ec2-user@ip-192-168-11-111 ~]$ kubectl get storageclass
NAME        PROVISIONER  RECLAIMPOLICY  VOLUMEBINDINGMODE  ALLOWVOLUMEEXPANSION  AGE
ebs-sc      ebs.csi.aws.com  Retain        WaitForFirstConsumer  false                5d6h
[ec2-user@ip-192-168-11-111 ~]$
```

i-077825c693e3953f1 (Bastion-EKS)

PublicIPs: 35.87.148.136 PrivateIPs: 192.168.11.111

🔗 Install ALB Ingress Controller on EKS

- Create **Ingress** → `ingress.yaml`
 - **Ingress annotations**

Steps to install Ingress Controller from [here](#)

- **install eksctl**
 - `eksctl version`
- **Enable OIDC Provider (Allow Kubernetes to talk with IAM)**

```
eksctl utils associate-iam-oidc-provider \
--region <Region> \
--cluster <EKS-Cluster-Name> \
--approve

# Ensure that **OIDC** Added
aws iam list-open-id-connect-providers

# if you want to delete **OIDC**
aws iam delete-open-id-connect-provider \
--open-id-connect-provider-arn <**OIDC-ARN**>
```

- **Create an IAM Role & Service Account for each other using `eksctl` (Step1)**

- Will not be installed until you **add admin access**, then **remove admin access**

```
# Download IAM Policy (Change Version according to document)
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.14.1/docs/install/iam_policy.json

# Create IAM Policy
```

```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json

eksctl create iamserviceaccount \
    --cluster=<EKS-Cluster-Name> \
    --namespace=kube-system \
    --name=**aws-load-balancer-controller** \
    --role-name AmazonEKSLoadBalancerControllerRole \
    --attach-policy-arn=arn:aws:iam::<Account-ID>:policy/AWSLoadBalancerControllerIAMPolicy \
    --approve \
    --region=<Region> \
    --override-existing-serviceaccounts
```

- **Install AWS Load Balancer Controller (Step2)**

- **Install Helm** from the script

```
helm repo add eks https://aws.github.io/eks-charts
helm repo update

# Display VPC_ID
aws eks describe-cluster --name <EKS-Cluster-Name> --query
"cluster.resourcesVpcConfig.vpcId" --output text

helm upgrade --install aws-load-balancer-controller eks/aws-load-balancer-
controller \
    -n kube-system \
    --set clusterName=<EKS-Cluster-Name> \
    --set serviceAccount.create=false \
    --set serviceAccount.name=**aws-load-balancer-controller** \
    --set region=<Region> \
    --set vpcId=<VPC-ID> \
    --set enableShield=false \
    --set enableWaf=false \
    --rollback-on-failure
```

- Ensure that the **Webhook** works well

- `kubectl get pods -n kube-system | grep aws-load` `kubectl get svc -n kube-system | grep webhook`

```
[ec2-user@ip-192-168-11-122 ~]$ kubectl get pods -n kube-system | grep aws-load
aws-load-balancer-controller-7fd4b579b6-f1qqw 1/1 Running 0 72s
aws-load-balancer-controller-7fd4b579b6-ppq4w 1/1 Running 0 62s
[ec2-user@ip-192-168-11-122 ~]$ kubectl get svc -n kube-system | grep webhook
aws-load-balancer-webhook-service ClusterIP 10.100.171.82 <none> 443/TCP 109s
[ec2-user@ip-192-168-11-122 ~]$
```

i-009e5d97b7e9fd066 (Bastion-EKS)
Public IPs: 44.247.229.82 Private IPs: 192.168.11.122

- **Verify that the controller is installed (Step3)**

- `kubectl get deployment -n kube-system aws-load-balancer-controller`

```
[ec2-user@ip-192-168-11-122 ~]$ kubectl get deployment -n kube-system aws-load-balancer-controller
NAME                   READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2      2           2          90s
[ec2-user@ip-192-168-11-122 ~]$
```

i-009e5d97b7e9fd066 (Bastion-EKS)

PublicIPs: 44.247.229.82 PrivateIPs: 192.168.11.122



Application Load Balancer

The screenshot shows the AWS CloudShell interface with the following details:

- EC2 > Load balancers > k8s-abdelham-apppingre-e60dd23663**
- Details** tab:
 - Load balancer type:** Application
 - Status:** Active
 - Hosted zone:** Z1H1FLSHABSFS
 - VPC:** [vpc-0779b6fdf8d7882df](#)
 - Availability Zones:**
 - [subnet-0f5b413b75633350](#) us-west-2b (usw2-az1)
 - [subnet-0ab60e0ed8c595fcc1](#) us-west-2c (usw2-az3)
 - [subnet-0b5f159f5d7bf1c54](#) us-west-2a (usw2-az2)
 - Load balancer IP address type:** IPv4
 - Date created:** January 27, 2026, 15:49 (UTC+02:00)
- Listeners and rules** tab (selected):
 - Listeners and rules (1) Info**: A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.
 - Protocol:Port**: HTTP:80
 - Default action**: Return fixed response (Response code: 404, Response body, Response content type: text/plain)
 - Rules**: 3 rules
 - ARN**: Not applicable
 - Security policy**: Not applicable
 - Default SSL/TLS**: Not applicable

Target Group

The screenshot shows the AWS Target Groups page with the following details:

- Target groups (2)**: [Info](#) | [What's new?](#)
- Actions** button
- Create target group** button
- Filter target groups**: k8s-abdelham
- Clear filters** button
- Table Headers:** Name, ARN, Port, Protocol, Target type, Load balancer, VPC ID
- Data Rows:**

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
k8s-abdelham-backend...	arn:aws:elasticloadbalancin...	3000	HTTP	IP	k8s-abdelham-apppingre...	vpc-077c...
k8s-abdelham-frontend...	arn:aws:elasticloadbalancin...	80	HTTP	IP	k8s-abdelham-apppingre...	vpc-077c...

Backend Target Group

Target groups (1/2)

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
k8s-abdelham-backend...	arn:aws:elasticloadbalancin...	3000	HTTP	IP	k8s-abdelham-appinge...	vpc-0775
k8s-abdelham-frontend...	arn:aws:elasticloadbalancin...	80	HTTP	IP	k8s-abdelham-appinge...	vpc-0775

Target group: k8s-abdelham-backend-5017e8aaa2

Registered targets (1)

IP address	Port	Zone	Health status	Health status details	Administrative ove...	Overri...	Anomaly detectio...
192.168.6.75	3000	us-west-2a ...	Healthy	-	No override.	No overri...	Normal

Frontend Target Group

Target groups (1/2)

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
k8s-abdelham-backend...	arn:aws:elasticloadbalancin...	3000	HTTP	IP	k8s-abdelham-appinge...	vpc-0775
k8s-abdelham-frontend...	arn:aws:elasticloadbalancin...	80	HTTP	IP	k8s-abdelham-appinge...	vpc-0775

Target group: k8s-abdelham-frontend-f978f8f381

Registered targets (1)

IP address	Port	Zone	Health status	Health status details	Administrative ove...	Overri...	Anomaly detectio...
192.168.6.209	80	us-west-2a ...	Healthy	-	No override.	No overri...	Normal

Deployment, Service, Ingress

```
[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)

PublicIPs: 35.87.148.136 PrivateIPs: 192.168.11.111

Output