✅ **Create an AWS ECR repository** for patient-service.js  
✅ **Push the container image to ECR**  
✅ **Deploy the container on AWS Fargate (ECS)**  
✅ **Expose the service via an Application Load Balancer (ALB)**

**1️ Terraform Configuration (main.tf)**

**🔹 Create a VPC, Subnets, and Security Groups**

provider "aws" {

region = "us-east-1"

}

# VPC

resource "aws\_vpc" "ecs\_vpc" {

cidr\_block = "10.0.0.0/16"

}

# Public Subnets

resource "aws\_subnet" "public\_subnet\_1" {

vpc\_id = aws\_vpc.ecs\_vpc.id

cidr\_block = "10.0.1.0/24"

availability\_zone = "us-east-1a"

map\_public\_ip\_on\_launch = true

}

resource "aws\_subnet" "public\_subnet\_2" {

vpc\_id = aws\_vpc.ecs\_vpc.id

cidr\_block = "10.0.2.0/24"

availability\_zone = "us-east-1b"

map\_public\_ip\_on\_launch = true

}

# Internet Gateway

resource "aws\_internet\_gateway" "ecs\_igw" {

vpc\_id = aws\_vpc.ecs\_vpc.id

}

# Route Table

resource "aws\_route\_table" "ecs\_route\_table" {

vpc\_id = aws\_vpc.ecs\_vpc.id

}

resource "aws\_route" "internet\_access" {

route\_table\_id = aws\_route\_table.ecs\_route\_table.id

destination\_cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.ecs\_igw.id

}

resource "aws\_route\_table\_association" "subnet\_1\_assoc" {

subnet\_id = aws\_subnet.public\_subnet\_1.id

route\_table\_id = aws\_route\_table.ecs\_route\_table.id

}

resource "aws\_route\_table\_association" "subnet\_2\_assoc" {

subnet\_id = aws\_subnet.public\_subnet\_2.id

route\_table\_id = aws\_route\_table.ecs\_route\_table.id

}

# Security Group

resource "aws\_security\_group" "ecs\_sg" {

vpc\_id = aws\_vpc.ecs\_vpc.id

ingress {

from\_port = 3000

to\_port = 3000

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

}

**🔹 Create an ECR Repository for patient-service.js**

hcl

CopyEdit

# ECR Repository

resource "aws\_ecr\_repository" "patient\_service" {

name = "patient-service"

}

output "ecr\_repository\_url" {

value = aws\_ecr\_repository.patient\_service.repository\_url

}

NOTE: ECR Repository is created by CICD pipeline based on the name configured in github secret

**🔹 Create an ECS Cluster**

resource "aws\_ecs\_cluster" "ecs\_fargate" {

name = "ecs-fargate-cluster"

}

**🔹 IAM Role for ECS Task Execution**

resource "aws\_iam\_role" "ecs\_task\_execution\_role" {

name = "ecs-task-execution-role"

assume\_role\_policy = jsonencode({

Version = "2012-10-17"

Statement = [{

Effect = "Allow"

Principal = { Service = "ecs-tasks.amazonaws.com" }

Action = "sts:AssumeRole"

}]

})

}

resource "aws\_iam\_policy\_attachment" "ecs\_task\_execution\_attachment" {

name = "ecs-task-execution-attachment"

roles = [aws\_iam\_role.ecs\_task\_execution\_role.name]

policy\_arn = "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"

}

**🔹 Create an ECS Task Definition for patient-service.js**

resource "aws\_ecs\_task\_definition" "patient\_service\_task" {

family = "patient-service-task"

requires\_compatibilities = ["FARGATE"]

network\_mode = "awsvpc"

memory = "512"

cpu = "256"

execution\_role\_arn = aws\_iam\_role.ecs\_task\_execution\_role.arn

container\_definitions = jsonencode([{

name = "patient-service"

image = aws\_ecr\_repository.patient\_service.repository\_url

cpu = 256

memory = 512

essential = true

portMappings = [{ containerPort = 3000, hostPort = 3000 }]

}])

}

**🔹 Deploy ECS Service for patient-service.js**

resource "aws\_ecs\_service" "patient\_service" {

name = "patient-service"

cluster = aws\_ecs\_cluster.ecs\_fargate.id

task\_definition = aws\_ecs\_task\_definition.patient\_service\_task.arn

desired\_count = 1

launch\_type = "FARGATE"

network\_configuration {

subnets = [aws\_subnet.public\_subnet\_1.id, aws\_subnet.public\_subnet\_2.id]

security\_groups = [aws\_security\_group.ecs\_sg.id]

assign\_public\_ip = true

}

}

**🔹 Create an Application Load Balancer (ALB)**

resource "aws\_lb" "ecs\_alb" {

name = "ecs-alb"

internal = false

load\_balancer\_type = "application"

security\_groups = [aws\_security\_group.ecs\_sg.id]

subnets = [aws\_subnet.public\_subnet\_1.id, aws\_subnet.public\_subnet\_2.id]

}

# Target Group

resource "aws\_lb\_target\_group" "ecs\_tg" {

name = "ecs-tg"

port = 3000

protocol = "HTTP"

vpc\_id = aws\_vpc.ecs\_vpc.id

target\_type = "ip"

}

# Listener

resource "aws\_lb\_listener" "ecs\_listener" {

load\_balancer\_arn = aws\_lb.ecs\_alb.arn

port = 3000

protocol = "HTTP"

default\_action {

type = "forward"

target\_group\_arn = aws\_lb\_target\_group.ecs\_tg.arn

}

}

# Attach ECS Service to Target Group

resource "aws\_lb\_target\_group\_attachment" "ecs\_tg\_attach" {

target\_group\_arn = aws\_lb\_target\_group.ecs\_tg.arn

target\_id = aws\_ecs\_service.patient\_service.id

}

**2️ Deploy Using Terraform**

**Step 1: Initialize Terraform**

terraform init

**Step 2: Plan the Deployment**

terraform plan

**Step 3: Apply the Changes**

terraform apply -auto-approve

**3️ Build & Push Your Docker Image to ECR**

Once Terraform creates the ECR repo, **build and push your container**:

**Authenticate with ECR**

aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 897722687643.dkr.ecr.us-east-1.amazonaws.com/patient-service

897722687643.dkr.ecr.us-east-1.amazonaws.com/appointment-service

**Build the Docker Image**

docker build -t patient-service -f Dockerfile.patient .

**Tag & Push the Image**

docker tag patient-service:latest 897722687643.dkr.ecr.us-east-1.amazonaws.com/patient-service:latest

docker push 897722687643.dkr.ecr.us-east-1.amazonaws.com/patient-service:latest

Replace 897722687643.dkr.ecr.us-east-1.amazonaws.com/patient-service with the **ECR URL** from Terraform output.

**4️ Access the Patient Service**

Get the **Load Balancer DNS**:

aws elbv2 describe-load-balancers --query "LoadBalancers[\*].DNSName" --output text

ecs-alb-1112778657.us-east-1.elb.amazonaws.com

Open the **ALB URL** in your browser:

http://<ALB-DNS>:3000

http://ecs-alb-1112778657.us-east-1.elb.amazonaws.com:3000

**🚀 Next Steps**

Would you like to: ✅ **Add Auto-Scaling** for ECS?  
✅ **Enable CloudWatch Logs & Monitoring?**  
✅ **Use API Gateway Instead of ALB?**

A **503 Service Temporarily Unavailable** error usually means one of the following issues:

1️⃣ **ECS Task is Not Running** – The task might have failed to start.  
2️⃣ **ECS Task Failed to Register with Target Group** – The ALB is forwarding traffic, but the ECS task is not attached.  
3️⃣ **Container is Crashing** – The application might not be running properly inside the container.  
4️⃣ **Security Group or Subnet Issues** – The container may not have internet access.

**🔍 Step 1: Check ECS Task Status**

Run this command to check the **ECS service and task status**:

sh

CopyEdit

aws ecs describe-services --cluster ecs-fargate-cluster --services patient-service --query "services[\*].{Status: status, RunningCount: runningCount, DesiredCount: desiredCount}" --output table

* ✅ If RunningCount = 1 and DesiredCount = 1, the task is running fine.
* ❌ If RunningCount = 0, check logs (next step).

**🔍 Step 2: Check ECS Task Logs**

Run this to get recent logs from CloudWatch:

sh

CopyEdit

aws logs describe-log-groups --query "logGroups[\*].logGroupName" --output table

Find the log group for patient-service, then fetch logs:

sh

CopyEdit

aws logs tail /aws/ecs/containerinsights/ecs-fargate-cluster/application --follow

If you see errors like ECONNREFUSED or Cannot start service, the container might be crashing.

**🔍 Step 3: Verify Target Group Health**

Check if the ECS task is properly registered with the target group:

sh

CopyEdit

aws elbv2 describe-target-health --target-group-arn $(terraform output -raw aws\_lb\_target\_group.ecs\_tg.arn)

* ✅ If it says **healthy**, the task is fine.
* ❌ If it says **unhealthy**, describe why:

sh

CopyEdit

aws elbv2 describe-target-health --target-group-arn $(terraform output -raw aws\_lb\_target\_group.ecs\_tg.arn) --query "TargetHealthDescriptions[\*].TargetHealth.Reason"

**🔍 Step 4: Test Connectivity Inside the Container**

If the task is running, **exec into the container** and test the app:

sh

CopyEdit

aws ecs execute-command --cluster ecs-fargate-cluster --task <TASK\_ID> --container patient-service --command "/bin/sh" --interactive

Once inside, run:

sh

CopyEdit

curl http://localhost:3000/patients

* ✅ If it responds, the issue is with **networking or ALB**.
* ❌ If it **fails**, the app is not running properly.

**🔧 Possible Fixes**

**Fix 1: Ensure Your App Listens on 0.0.0.0**

Inside patient-service.js, update:

js

CopyEdit

const express = require('express');

const app = express();

app.use(express.json());

app.get('/patients', (req, res) => {

res.json([{ id: 1, name: "John Doe" }]);

});

const PORT = process.env.PORT || 3000;

app.listen(PORT, '0.0.0.0', () => console.log(`Server running on port ${PORT}`));

Then rebuild & push:

sh

CopyEdit

docker build -t patient-service .

docker tag patient-service:latest <ECR\_REPO\_URL>:latest

docker push <ECR\_REPO\_URL>:latest

Then update the ECS service:

sh

CopyEdit

aws ecs update-service --cluster ecs-fargate-cluster --service patient-service --force-new-deployment

**Fix 2: Ensure Security Groups Allow Traffic**

Check the security group of your ECS task and ALB:

sh

CopyEdit

aws ec2 describe-security-groups --group-ids <SECURITY\_GROUP\_ID>

Make sure: ✅ **ALB allows traffic on port 3000**  
✅ **ECS task allows traffic on port 3000**

**Fix 3: Restart the ECS Service**

If everything looks correct, restart the service:

sh

CopyEdit

aws ecs update-service --cluster ecs-fargate-cluster --service patient-service --desired-count 0

sleep 10

aws ecs update-service --cluster ecs-fargate-cluster --service patient-service --desired-count 1