## **🧱 Step 1: Create Your Project Folder**

1. Open VS Code (or your terminal).
2. Run the following:

mkdir cloudguard-s3-auditor

cd cloudguard-s3-auditor

python -m venv venv   
  
# create virtual environment  
use: .\venv\Scripts\Activate.ps1  
  
Inside the folder, create this structure:

mkdir app  
New-Item -ItemType File app\main.py

## **🧠 Step 2: Write Python Code**

🔹 Open app/main.py and paste this code:

import boto3

from fastapi import FastAPI

import json

app = FastAPI()

@app.get("/scan")

def scan\_buckets():

s3 = boto3.client("s3")

public\_buckets = []

buckets = s3.list\_buckets()["Buckets"]

for bucket in buckets:

name = bucket["Name"]

try:

acl = s3.get\_bucket\_acl(Bucket=name)

for grant in acl["Grants"]:

grantee = grant.get("Grantee", {})

if grantee.get("URI", "") == "http://acs.amazonaws.com/groups/global/AllUsers":

public\_buckets.append({

"Bucket": name,

"Permission": grant["Permission"]

})

except Exception as e:

print(f"Could not check {name}: {e}")

return {"public\_buckets": public\_buckets}

**📦 Step 3: Install the Required Python Packages**

In your terminal (make sure you're inside the cloudguard-s3-auditor folder and your virtual environment is activated):

pip install boto3 fastapi uvicorn

## **🧪 Step 4: Run the App Locally**

In your terminal, run:

uvicorn app.main:app --reload

You should see something like:

Uvicorn running on http://127.0.0.1:8000

Now go to your browser and open:  
 👉<http://127.0.0.1:8000/scan>

It will attempt to scan your AWS account's S3 buckets for public access.

**☁️ Step 5: Connect to AWS**

Before this works, make sure you've configured your AWS CLI with credentials.

aws configure

It will ask for:

* Access Key ID
* Secret Key
* Region (e.g. us-east-1)

You can get these from your AWS console → IAM → Users → Your User → "Security Credentials".

## **🐳 Step 6: Dockerize the App**

### **🔹 Create a Dockerfile in your project root:**

# Use a lightweight Python image

FROM python:3.10-slim

# Set working directory

WORKDIR /app

# Copy the project files

COPY app/ ./app

COPY requirements.txt .

# Install dependencies

RUN pip install --no-cache-dir -r requirements.txt

# Start FastAPI app using Uvicorn

CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]

Start Docker Daemon if not running then

### **🔹 Build the Docker image:**

In your terminal (make sure you're in the project root):

docker build -t cloudguard-s3-auditor .

### **🔹 Run the Docker container:**

docker run -p 8000:8000 cloudguard-s3-auditor

Now go to<http://localhost:8000/scan> again — should work just like before!

## **📦 Step 7: Push to Amazon ECR (Elastic Container Registry)**

This lets you deploy your Docker image in the cloud.

### **🔹 1. Create an ECR repository:**

In your terminal:

aws ecr create-repository --repository-name cloudguard-s3-auditor

### **🔹 2. Authenticate Docker to ECR:**

aws ecr get-login-password | docker login --username AWS --password-stdin <your-account-id>.dkr.ecr.<region>.amazonaws.com

(Replace with your real account ID and region)

### **🔹 3. Tag & Push your Docker image:**

docker tag cloudguard-s3-auditor:latest <your-account-id>.dkr.ecr.<region>.amazonaws.com/cloudguard-s3-auditor

docker push <your-account-id>.dkr.ecr.<region>.amazonaws.com/cloudguard-s3-auditor

**🚀 Step 8: Deploy to AWS ECS Fargate using Terraform**

Now we’ll automate everything: the service, the task, IAM roles, etc.

### **🔹 Create a terraform/ folder in your project:**

mkdir terraform

cd terraform

New-Item -Path . -Name "main.tf" -ItemType "File"

New-Item -Path . -Name "variables.tf" -ItemType "File"

New-Item -Path . -Name "outputs.tf" -ItemType "File"

### **📄 terraform/main.tf**

provider "aws" {

region = var.aws\_region

}

resource "aws\_ecs\_cluster" "this" {

name = "cloudguard-cluster"

}

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

name = "ecsTaskExecutionRole"

assume\_role\_policy = jsonencode({

Version = "2012-10-17"

Statement = [{

Action = "sts:AssumeRole"

Principal = {

Service = "ecs-tasks.amazonaws.com"

}

Effect = "Allow"

}]

})

managed\_policy\_arns = [

"arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"

]

}

# Add your ECR repository URL here:

variable "image\_url" {

default = "<your-ecr-image-url>"

}

resource "aws\_ecs\_task\_definition" "app" {

family = "cloudguard-task"

network\_mode = "awsvpc"

requires\_compatibilities = ["FARGATE"]

cpu = "256"

memory = "512"

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

container\_definitions = jsonencode([

{

name = "cloudguard"

image = var.image\_url

portMappings = [

{

containerPort = 8000

protocol = "tcp"

}

]

}

])

}

**📄 terraform/variables.tf**

variable "aws\_region" {

default = "us-east-1"

}

variable "image\_url" {

description = "The ECR image URL"

type = string

}

variable "app\_name" {

default = "cloudguard"

}

**📄 terraform/outputs.tf**

output "load\_balancer\_url" {

value = aws\_lb.app\_lb.dns\_name

}

Initialize and Deploy

Make sure you're in the terraform/ folder and run:

terraform init

terraform plan -var="image\_url=<your-ecr-image-url>"

terraform apply -var="image\_url=<your-ecr-image-url>"

When it completes, you'll get a load\_balancer\_url – open it in your browser and hit:

http://<LB\_URL>/scan