# 🧠 AI Copilot for EHR System – Full Deployment Guide (AWS GPU EC2)

This document outlines a systematic plan to build, host, and integrate an AI Copilot into your Electronic Health Record (EHR) system using open-source models on an AWS GPU EC2 instance. It is optimized for low cost, high performance, and ease of integration into your existing infrastructure.

## 1. Goals and Capabilities

The AI Copilot will:  
- Suggest diagnoses based on patient symptoms  
- Recommend treatment plans based on medical guidelines (WHO, NHS, Ghana STG)  
- Provide drug dosing assistance tailored to demographics  
- Offer real-time answers to clinical queries

## 2. Technology Stack (Open-Source + AWS)

- Model: LLaMA 3 (8B), Mistral 7B, or Mixtral (quantized or full)  
- Inference Engine: text-generation-webui or vLLM  
- Embedding: HuggingFace Sentence Transformers  
- Vector Store: FAISS  
- Backend: FastAPI  
- Deployment: EC2 GPU instance (g4dn.xlarge or g5.xlarge)  
- Orchestration: LangChain or LlamaIndex

## 3. Deployment Guide

Step-by-step instructions:

### Step 1: Launch EC2 GPU Instance

- Use g4dn.xlarge or g5.xlarge  
- Ubuntu 22.04 (Deep Learning AMI recommended)  
- Open ports: 22, 8000, 443  
- EBS volume: 100 GB

### Step 2: Install Dependencies

Refer to `install\_ai\_copilot.sh` script provided.

### Step 3: Download and Serve the Model

Using text-generation-webui, download Mistral or LLaMA quantized models and run the inference server.

### Step 4: Build Embedding Pipeline

Use `sentence-transformers/all-MiniLM-L6-v2` with FAISS to embed and index treatment documents.

### Step 5: Build RAG Pipeline

Use LangChain to combine the vector DB and LLM server with RetrievalQA.

### Step 6: API Layer with FastAPI

Build an API that allows your EHR system to send questions and receive answers.

### Step 7: EHR Integration

Add a chat widget to your existing frontend and connect it to the API via fetch or axios.

## 4. Automation Tools

- Bash Installer: `install\_ai\_copilot.sh`  
- Dockerfile: For GPU containerized deployment  
- Uvicorn: For serving the FastAPI backend

## 5. Cost Optimization & Security

- Use spot instances or auto-stop script when idle  
- Use quantized models to reduce GPU usage  
- Secure endpoints with NGINX reverse proxy and HTTPS  
- Use IAM roles and security groups properly