BÁO CÁO:

IOT - LAB 3

Ngô Quang Minh - N22DCCN053

Nguyễn Khắc Tùng Dương - N22DCCN018

---

1. Tóm tắt Giới thiệu

Việc triển khai (deploy) một mô hình Machine Learning không chỉ dừng lại ở việc huấn luyện mô hình. Để mô hình có thể được sử dụng trong một ứng dụng thực tế, nó cần được "gói" trong một API và vận hành như một dịch vụ (service) ổn định, có khả năng mở rộng.

Quy trình này mô tả kiến trúc hiện đại sử dụng **Docker** để đóng gói ứng dụng, **Amazon Elastic Container Registry (ECR)** để lưu trữ các image, và **Amazon Elastic Container Service (ECS)** để tự động hóa việc triển khai, vận hành và nhân rộng dịch vụ.

Kiến trúc tổng quan:

1. **Local/CI/CD:** Mã nguồn ứng dụng (ví dụ: Flask/Python) và mô hình đóng gói vào một Docker image.
2. **Amazon ECR:** Docker image này được đẩy (push) và lưu trữ an toàn trong một kho lưu trữ (repository) riêng tư.
3. **Amazon ECS:** Dịch vụ sẽ đọc image từ ECR, sau đó tự động khởi chạy và duy trì các container.
4. **Client:** Người dùng cuối truy cập vào dịch vụ thông qua một địa chỉ IP công cộng hoặc một Bộ cân bằng tải (Load Balancer).

2. Các Công nghệ Chính

- **Docker:** Nền tảng để đóng gói ứng dụng (mã nguồn, thư viện, mô hình) vào một đơn vị độc lập gọi là "container image".
- **Amazon ECR (Elastic Container Registry):** Dịch vụ lưu trữ Docker image được quản lý hoàn toàn, có độ an toàn và tin cậy cao, tích hợp chặt chẽ với hệ sinh thái AWS.

- **Amazon ECS (Elastic Container Service):** Dịch vụ điều phối container (container orchestration) của AWS. ECS cho phép bạn chạy, dừng và quản lý các Docker container trên một cụm (cluster) máy chủ. Chúng ta sẽ tập trung vào chế độ **Fargate**, là chế độ "serverless" (không cần quản lý máy chủ).

3. Quy trình Triển khai Chi tiết

Đây là các bước kỹ thuật để triển khai từ máy tính cá nhân lên đám mây AWS.

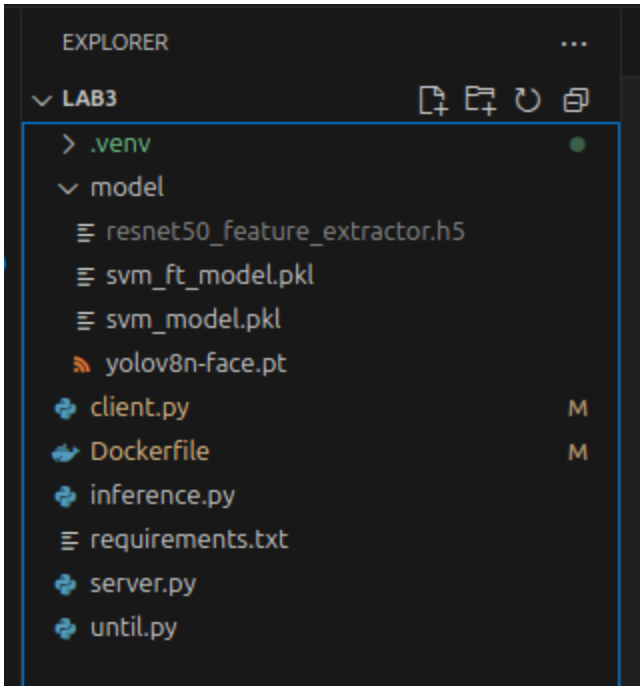
Bước 1: Docker hóa Ứng dụng Machine Learning

Trước tiên, ứng dụng Python (máy chủ Flask) và các mô hình của bạn cần được đóng gói.

1. **Cấu trúc thư mục (Ví dụ):**

/warning-mask-detection
|-- /models
| |-- restnet_50.h5
| |--svm.pkl
| |--yolo.pt
|-- server.py
|-- requirements.txt
|-- Dockerfile
|-- untils

2. **Viết `Dockerfile`:** Đây là tệp hướng dẫn Docker cách xây dựng image.
   Lưu ý quan trọng: Log của bạn cho thấy bạn đang dùng máy chủ phát triển (dev server) của Flask. Điều này **không** an toàn và không hiệu quả cho production. Bạn nên dùng một máy chủ WSGI như **Gunicorn**.
   Đây là một `Dockerfile` mẫu cho production:
3. Dockerfile

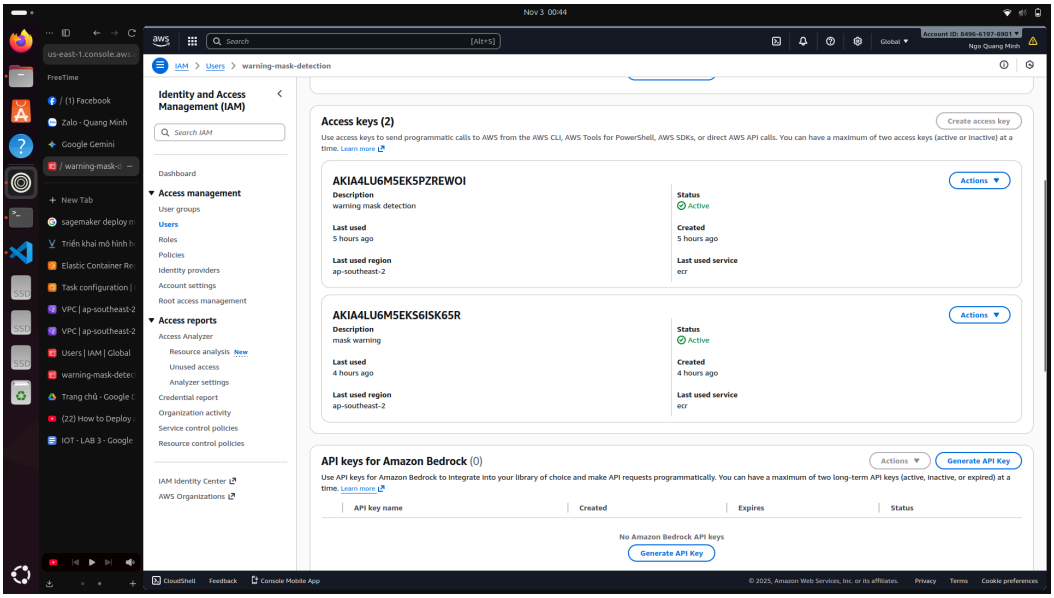


Bước 2: Đẩy Image lên Amazon ECR

Sau khi có `Dockerfile`, bạn cần xây dựng và đẩy image lên "kho" ECR.

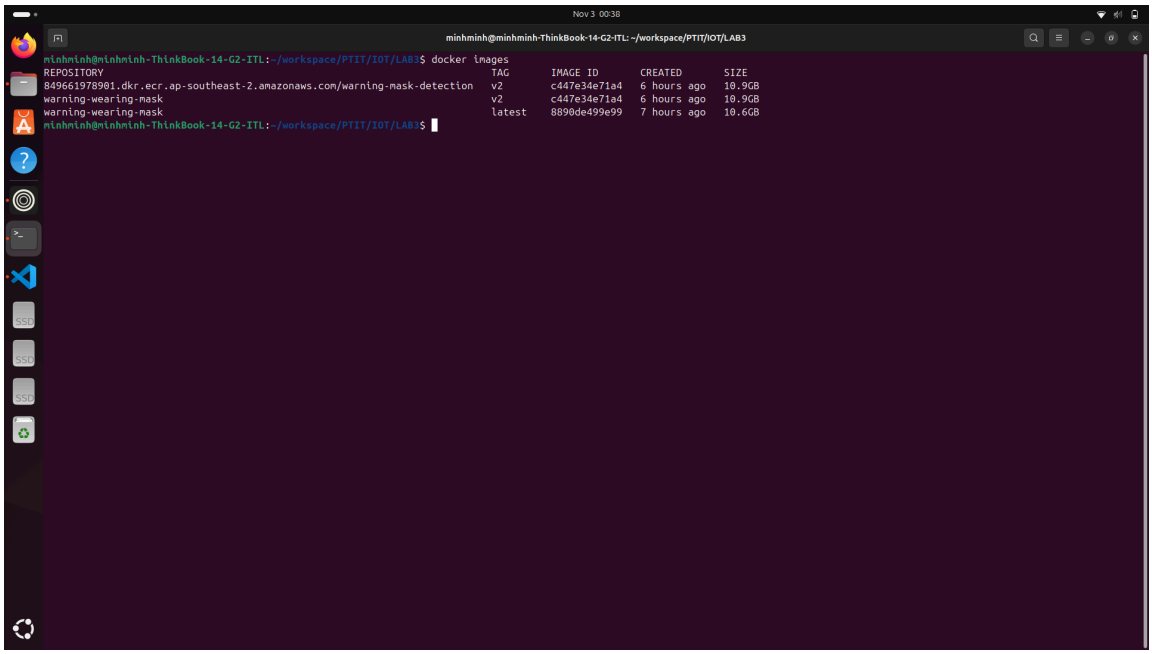
1. **Tạo một Repository trên ECR:**
    - Truy cập Bảng điều khiển (Console) AWS > **ECR**.
    - Nhấp "Create repository" (Tạo kho).
    - Đặt tên (ví dụ: `warning-mask-detection`) và giữ các cài đặt khác ở chế độ riêng tư (private).
2. **Xác thực Docker với ECR:**
    - Cấp phép cho Docker CLI trên máy của mình để được đẩy image lên ECR.

      Access key:

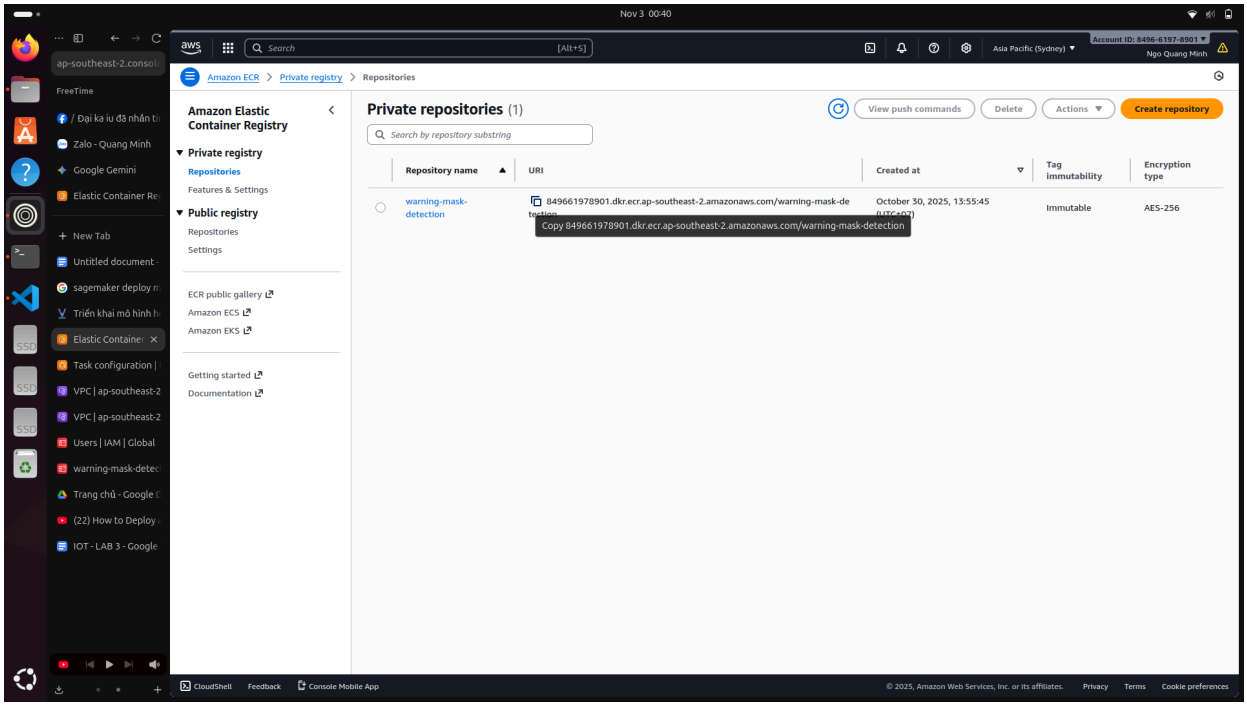


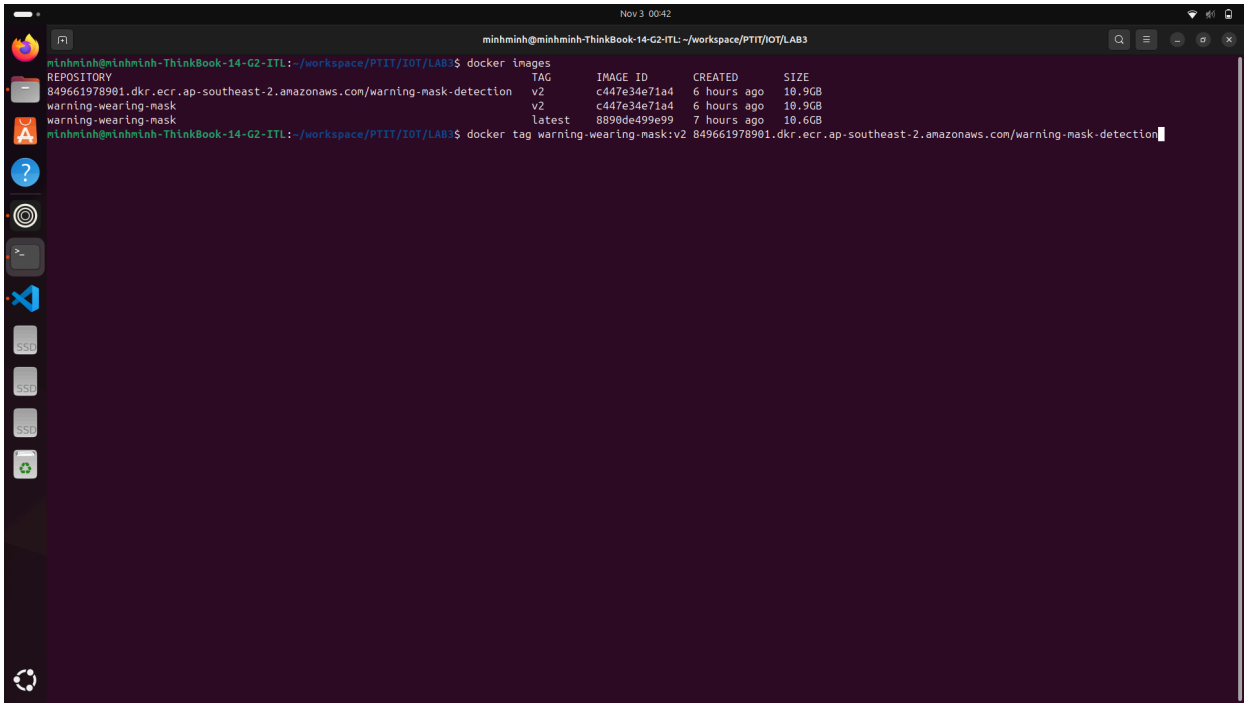
Bước 3: Build, Tag, và Push Image:

- **Build:** Xây dựng image từ `Dockerfile`.

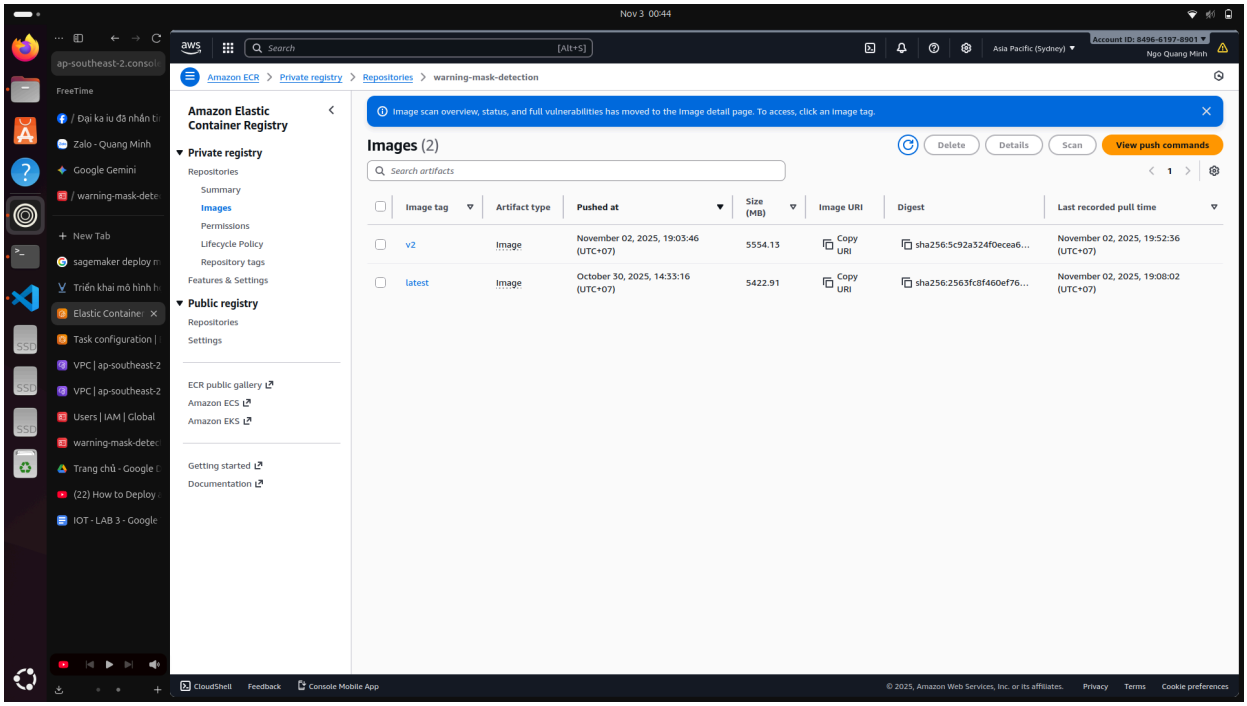


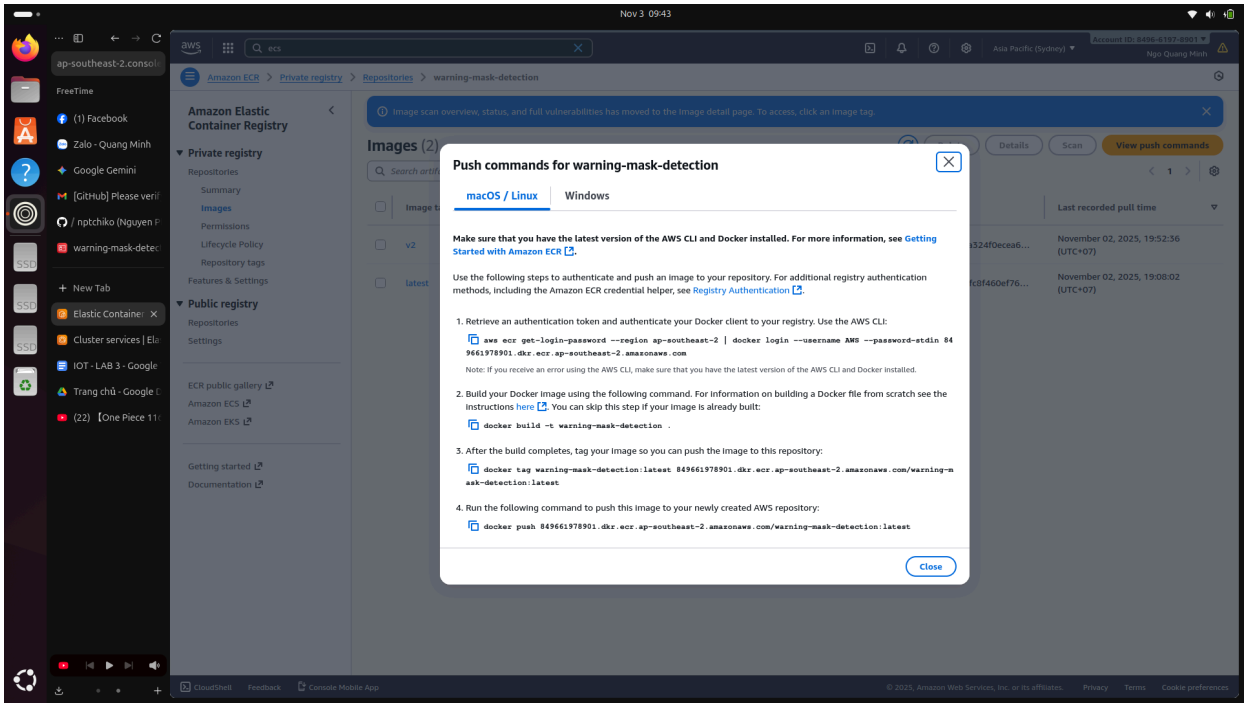
- **Tag:** Gắn thẻ image theo định dạng ECR yêu cầu.





- **Push:** Đẩy image lên kho lưu trữ ECR.

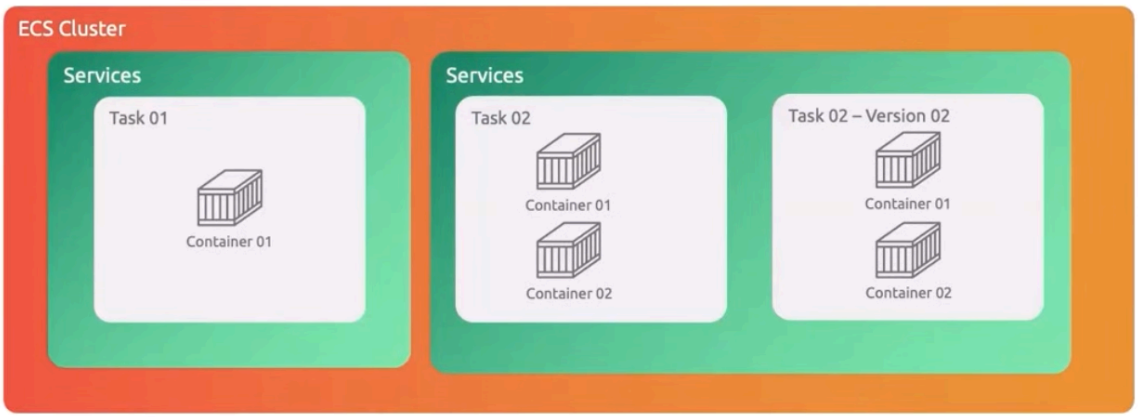




Bước 4. Vận hành Dịch vụ trên Amazon ECS

Đây là bước đang thực hiện. Chúng ta sẽ ECS lấy image từ ECR và chạy nó.

Understanding Services and Tasks



1. **Tạo ECS Cluster (Cụm):**
    - Truy cập Bảng điều khiển (Console) AWS > **ECS**.
    - Nhấp "Create cluster" (Tạo cụm).
    - Chọn "Networking only" (chỉ Mạng) để sử dụng **AWS Fargate**
    - Đặt tên cho Cluster (ví dụ: `hospitable-penguin-rfkjpf`.
2. **Tạo Task Definition (Định nghĩa Tác vụ):**
    - Đây là "bản thiết kế" cho container của bạn.
    - Trong ECS, vào "Task Definitions" > "Create new Task Definition".
    - Chọn "Fargate" làm launch type.
    - **Task role:** (Tùy chọn) Nếu ứng dụng của bạn cần gọi các dịch vụ AWS khác (như S3), hãy gán quyền ở đây.
    - **Operating system:** Linux.
    - **Task size:** Cung cấp CPU và Bộ nhớ (ví dụ: 1 vCPU, 4GB Memory - các mô hình ML thường cần nhiều RAM).
    - **Container Definitions (Quan trọng):**
        - Nhấp "Add container".
        - **Image (Quan trọng):** Dán đường dẫn URI của image ECR (ví dụ: `<account-id>.dkr.ecr.ap-southeast-2.amazonaws.com/warning-mask-detection:v2`).
        - **Port mappings (Quan trọng):** Thêm một mục: `Container port: 5000` (đây là cổng `EXPOSE` trong Dockerfile của bạn).
3. **Tạo Service (Dịch vụ):**

- Đây là bước khởi chạy và duy trì Tác vụ của bạn.
- Đi đến Cluster của bạn, nhấp vào tab "Services" > "Create".
- **Launch type:** Fargate.
- **Task Definition:** Chọn Task Definition bạn vừa tạo.
- **Service name:** Đặt tên (ví dụ: `warning-mask-v2-service-gku889s1`).
- **Desired tasks:** 1 (để chạy 1 bản sao của container).
- **Networking (Bước bạn đang gặp sự cố):**
  - **VPC & Subnets:** Chọn VPC và các Subnet (thường là mặc định).
  - **Security Groups (Rất quan trọng):**
    - Tạo một Security Group (nhóm bảo mật) MỚI.
    - Thêm một **"Inbound rule" (Quy tắc đi vào)**.
    - Loại (Type): `Custom TCP`.
    - Port Range (Dải cổng): `5000` (đây là cổng container).
    - Nguồn (Source): `Anywhere (0.0.0.0/0)` (để cho phép truy cập từ Internet).
  - **Auto-assign public IP (Rất quan trọng):** PHẢI đặt thành **ENABLED** (ĐÃ BẬT). Nếu không, container sẽ không có IP công cộng và không thể truy cập nó từ bên ngoài.
- **Load Balancing (Tùy chọn nhưng được khuyến nghị):**
  - Để sử dụng cho production (và truy cập qua cổng 80/443 thân thiện), bạn nên chọn "Application Load Balancer" (ALB).
  - ALB sẽ lắng nghe trên cổng `80` (HTTP) và chuyển tiếp lưu lượng truy cập đến "Target Group" (Nhóm mục tiêu).
  - Nhóm mục tiêu (Target Group) sẽ được cấu hình để gửi lưu lượng truy cập đến cổng `5000` của Tác vụ (Task).

4. **Truy cập Dịch vụ:**
   - Sau khi Dịch vụ (Service) chuyển sang trạng thái "RUNNING" (Đang chạy), hãy đi tới tab "Tasks" (Tác vụ) và nhấp vào Tác vụ (Task) đang chạy.
   - Trong phần "Network", tìm **"Public IP"** (IP công cộng) của nó.
   - Truy cập dịch vụ của bạn trong trình duyệt bằng: `http://<Public-IP-cua-ban>:5000`

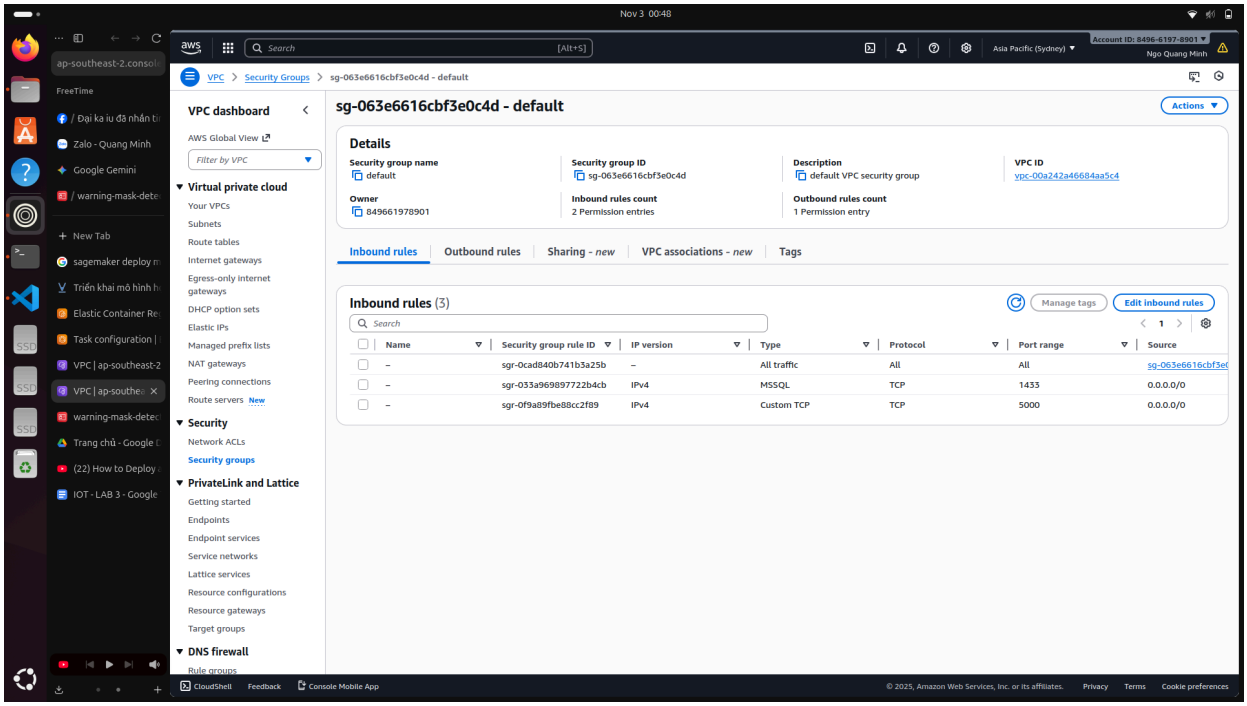
4. Kết luận và Khắc phục sự cố

Kiến trúc **ECR + ECS Fargate** là một phương pháp mạnh mẽ, hiện đại và có khả năng mở rộng cao để triển khai các ứng dụng, đặc biệt là các mô hình ML.

Vấn đề phổ biến nhất: "Không thể truy cập URL."

Nguyên nhân luôn là do **Networking (Mạng)**:

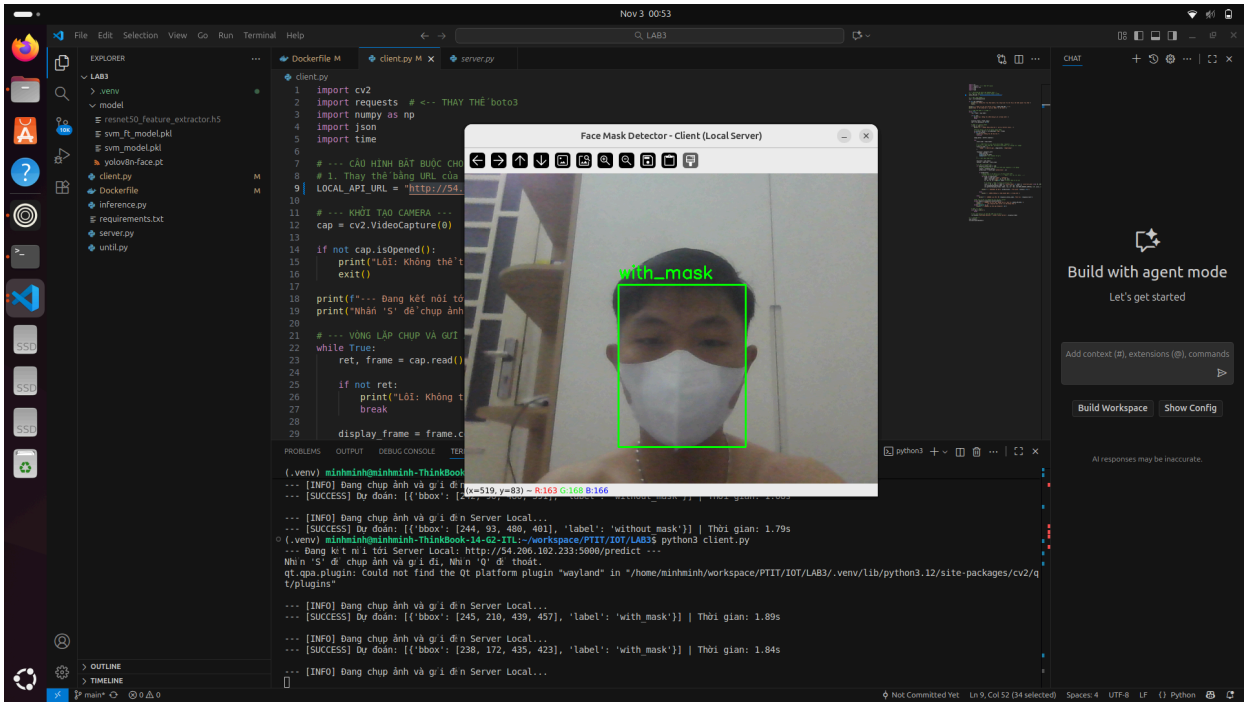
1. **Security Group:** Đã chặn cổng. Đảm bảo cổng (`5000`) được mở cho `0.0.0.0/0` trong Inbound Rules.
2. **Public IP:** Tác vụ (Task) không được gán IP công cộng. Đảm bảo "Auto-assign public IP" được `ENABLED` khi tạo Dịch vụ.
3. **Port Mapping:** Cổng trong Task Definition (`5000`) không khớp với cổng mà ứng dụng Flask/Gunicorn của bạn đang chạy (`5000`).



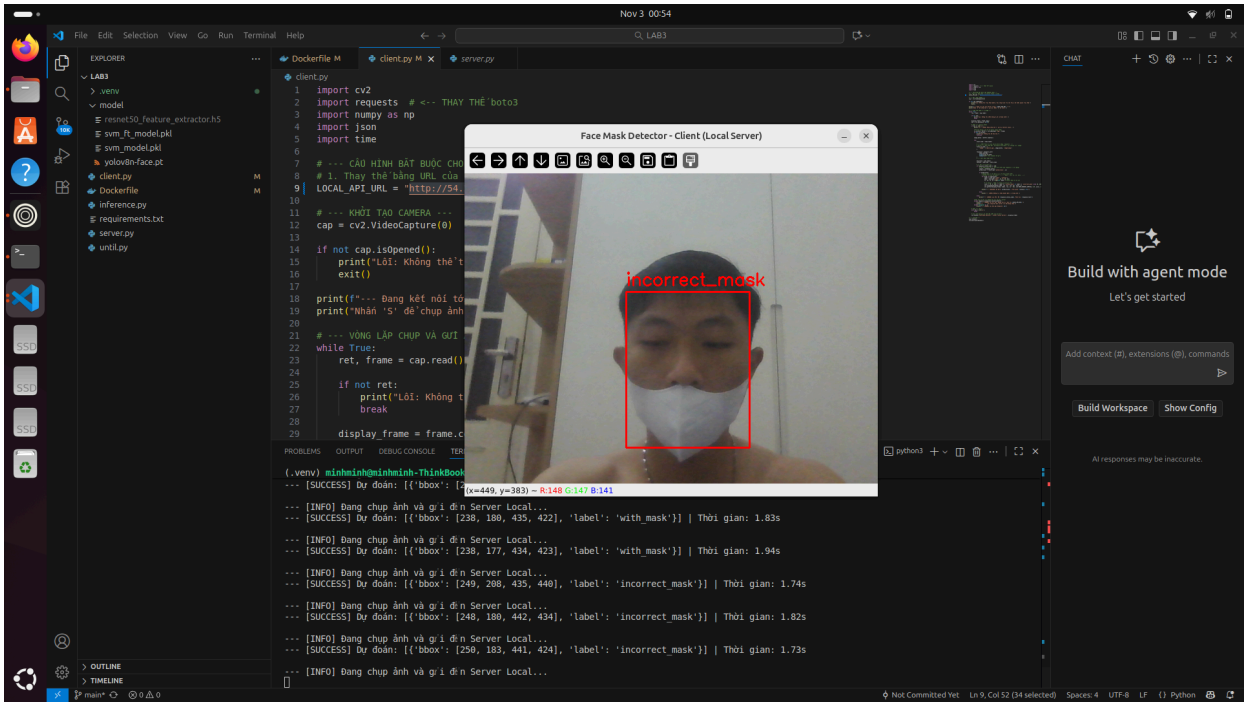
5. DEMO

Public server ip: `http://54.206.102.233:5000/predict`

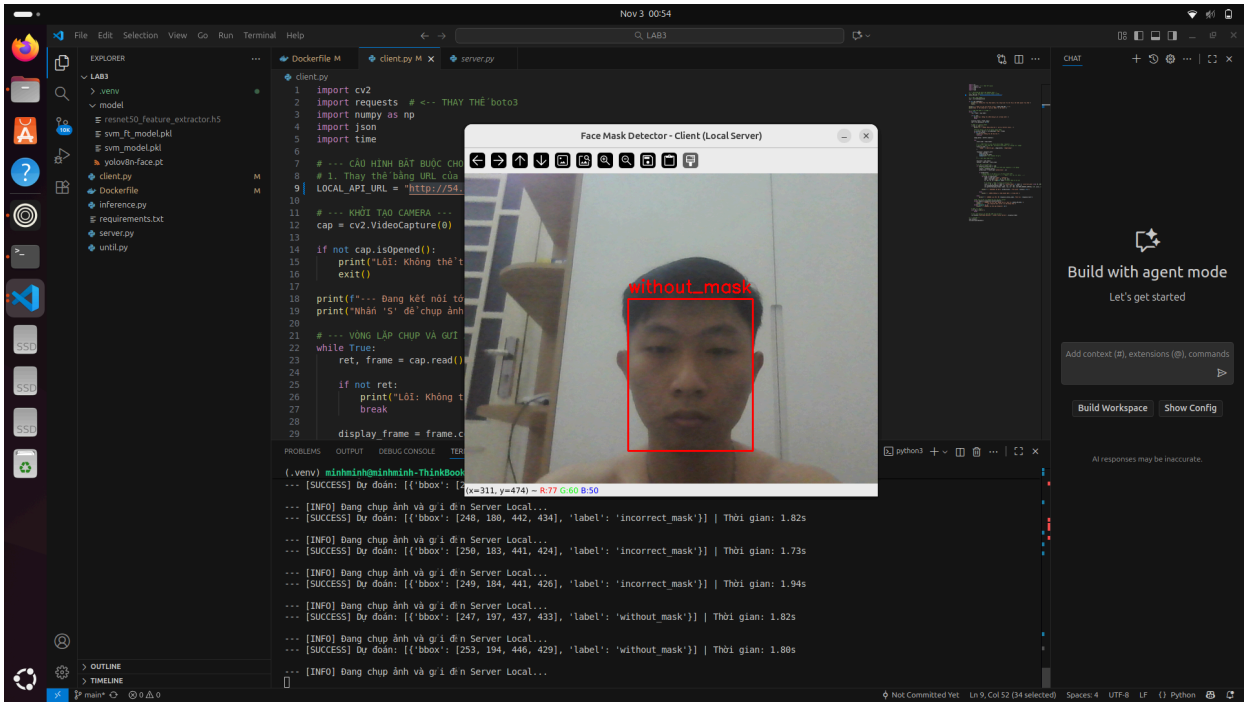
1. **Mask:**



2. Incorrect mask:



3. No mask:


Face Mask Detector - Client (Local Server)
without_mask
(x=311, y=474) ~ R:77 G:60 B:50
Build with agent mode
Let's get started
Build Workspace
Show Config