MLOps trên AWS (EC2, S3, ECR, ECS): Từ lý thuyết đến thực hành

Time Series Team

Mục lục

1. Mở đầu: MLOps là gì và khác gì với DevOps?

1.1. DevOps: Nền tảng tự động hoá phần mềm

DevOps là tư duy và tập hợp thực hành kết nối **phát triển phần mềm (Dev)** và **vận hành hệ thống (Ops)** nhằm rút ngắn vòng đời phát triển, tăng chất lượng và triển khai liên tục. Cốt lõi của DevOps là *tự động hoá* (kiểm thử, tích hợp, triển khai), *giám sát* và *phản hồi nhanh.*

Vòng đời DevOps:

Lập kế hoạch → Lập trình → Biên dịch → Kiểm thử → Phát hành → Triển khai → Vận hành → Giám sát

Nhóm công cụ tiêu biểu:

- Quản lý mã nguồn: Git, GitHub, GitLab.
- CI/CD: Jenkins, GitHub Actions, GitLab CI.
- Đóng gói: Docker, Podman.
- Điều phối: Kubernetes.
- Hạ tầng như mã (IaC): Terraform, CloudFormation.
- Giám sát: Prometheus, Grafana; nhật ký: ELK.
- Quản lý bí mật: HashiCorp Vault.

1.2. MLOps: DevOps cho học máy

MLOps mở rộng DevOps sang vòng đời học máy, tập trung xử lý **dữ liệu, mô hình và thí nghiệm** một cách chuẩn hoá, có thể tái lập và mở rộng. Các yêu cầu bổ sung gồm: phiên bản hoá dữ liệu/mô hình, huấn luyện lại tự động, theo dõi sai lệch (drift), và quản trị mô hình.

Chu trình MLOps:

Thu thập dữ liệu → Tiền xử lý → Huấn luyện → Thẩm định → Triển khai → Giám sát → Phản hồi

Công cụ thường dùng:

Hạng mục	Công cụ
Dòng chảy dữ liệu / Thử nghiệm	DVC, MLflow, Airflow
Huấn luyện mô hình	TensorFlow, PyTorch
Triển khai/Phục vụ	Docker, Kubernetes, SageMaker, FastAPI
Giám sát	Prometheus, Grafana

2. AWS EC2: Nền tính toán linh hoạt cho MLOps

Amazon EC2 cung cấp năng lực tính toán theo nhu cầu (IaaS) để chạy API, huấn luyện/phục vụ mô hình ML/DL, hoặc tác vụ lô (batch).

2.1. Tính năng trọng yếu

- **Khả năng mở rộng**: thay đổi cấu hình máy ảo theo tải.
- **Elastic IP**: địa chỉ IP tĩnh cho máy quan trọng.
- **Security Group**: tường lửa kiểm soát luồng vào/ra.
- **AMI**: ảnh máy ảo cài sẵn hệ điều hành/phần mềm.
- **User Data**: chạy script khởi tạo (cài driver, thư viện) khi máy boot lần đầu.

2.2. Họ máy (Instance families) và tình huống dùng

Họ máy	Mục đích	Ví dụ	Ứng dụng điển hình
General Purpose	Cân bằng CPU/RAM	M7i, T4g	Web, API, môi trường dev
Compute Optimized	Tính toán nặng CPU	C6i, C7a	HPC, xử lý video, batch
Memory Optimized	Dung lượng RAM lớn	R6g, X2idn	Big Data, in-memory DB
Storage Optimized	I/O đĩa cao	I3, D3	Kho dữ liệu, NoSQL
Accelerated	GPU/FPGA tăng tốc	P5, G6	Huấn luyện/Inference ML, đồ hoạ
Burstable	Tải dao động	T3.micro	Dev/Test, dịch vụ nhẹ

2.3. Mô hình chi phí

- **On-Demand**: linh hoạt, không cam kết.
- **Spot**: tiết kiệm lớn nhưng có thể bị thu hồi; phù hợp workload chịu lỗi.
- **Reserved**: cam kết 1–3 năm, tiết kiệm cao cho tải ổn định.
- **Savings Plan**: cam kết mức sử dụng theo giờ, linh hoạt loại máy.

2.4. Cân bằng tải và tự động mở rộng

ELB (Elastic Load Balancer) phân phối lưu lượng, tránh điểm lỗi đơn.
ASG (Auto Scaling Group) tự thêm/bớt instance theo tải, hỗ trợ scaling động, theo lịch, và dự đoán.

2.5. Gợi ý thiết kế môi trường ML trên EC2

- Chọn hệ điều hành (Ubuntu/Amazon Linux), bản CUDA/phù hợp GPU.
- Phân bổ vCPU/RAM/GPU theo bài toán; ổ đĩa EBS/EFS cho dữ liệu/chia sẻ.
- Cấu hình Security Group theo nguyên tắc tối thiểu quyền.
- Dùng User Data để cài đặt tự động hoá (driver, Python env, agent giám sát).

3. AWS S3: Lưu trữ đối tượng cho dữ liệu học máy

3.1. Giới thiệu tổng quan

Amazon S3 (Simple Storage Service) là dịch vụ **lưu trữ đối tượng (object storage)** được sử dụng rộng rãi nhất trên nền tảng AWS. Khác với ổ đĩa thông thường (file system) hay cơ sở dữ liệu (database), S3 được thiết kế để lưu trữ lượng dữ liệu **rất lớn, có thể lên tới petabyte**, với **độ bền (durability)** cực cao – lên tới **99.999999999% (11 số 9)**.

Trong các hệ thống học máy (Machine Learning), S3 thường đóng vai trò là **"trung tâm lưu trữ dữ liệu"** – nơi chứa:

- Dữ liệu thô (raw data) thu thập từ nhiều nguồn khác nhau.
- Dữ liệu đã tiền xử lý (processed data) phục vụ huấn luyện.
- Mô hình học máy đã huấn luyện (model weights, checkpoints).
- File log, kết quả huấn luyện, biểu đồ hoặc artifact khác.

S3 là nền tảng lý tưởng trong pipeline MLOps vì:

- Có thể truy cập trực tiếp từ các dịch vụ AWS khác (EC2, ECS, SageMaker, Lambda).
- Bảo mật linh hoạt nhờ quản lý quyền truy cập bằng IAM.
- Hoạt động ổn định, chi phí thấp và dễ mở rộng khi dữ liệu tăng dần.

3.2. Đặc tính chính

- **Khả năng mở rộng (Scalability):** S3 cho phép bạn lưu trữ lượng dữ liệu gần như không giới hạn, từ vài MB đến hàng trăm terabyte, mà không cần quản lý hạ tầng vật lý.
- **Độ bền và khả dụng cao (Durability & Availability):** Mỗi file được tự động sao lưu nhiều bản trên nhiều máy chủ và trung tâm dữ liệu trong cùng một vùng (region). Nếu một ổ cứng hỏng, dữ liệu vẫn an toàn nhờ cơ chế nhân bản nội bộ.
- **Bảo mật (Security):** Mọi truy cập vào S3 đều có thể kiểm soát chi tiết bằng:
  - Chính sách IAM (Identity and Access Management).
  - Bucket Policy (chính sách riêng cho từng bucket).
  - Cơ chế mã hóa dữ liệu khi truyền (SSL/TLS) và khi lưu trữ (SSE-S3, SSE-KMS).
- **Chi phí linh hoạt (Cost Efficiency):** S3 có nhiều lớp lưu trữ khác nhau tùy nhu cầu:
  - **Standard:** Dùng cho dữ liệu truy cập thường xuyên.
  - **Infrequent Access (IA):** Dùng cho dữ liệu ít truy cập, rẻ hơn.
  - **Glacier / Deep Archive:** Dùng cho lưu trữ dài hạn, truy xuất chậm nhưng chi phí cực thấp.

  Người dùng chỉ trả tiền cho dung lượng thật sự sử dụng, giúp tối ưu chi phí vận hành.

3.3. Mô hình dữ liệu và khái niệm cơ bản

S3 lưu dữ liệu theo mô hình **Bucket** – **Object** thay vì thư mục như hệ điều hành thông thường.

- **Bucket:** Là không gian lưu trữ chính trên S3, tương tự "ổ cứng" hoặc "thư mục gốc" của bạn. Mỗi bucket có tên **duy nhất trên toàn cầu** (toàn bộ AWS), và được tạo theo vùng địa lý (region). Ví dụ: `mlops-data-vietnam-2025`.
- **Object:** Là từng file cụ thể lưu trong bucket, ví dụ: `raw/movie_reviews.csv`, `models/sentiment_bert.p`
  Mỗi object có bốn thành phần:
  - **Data:** Nội dung của file (hình ảnh, văn bản, mô hình, v.v.).
  - **Key:** Tên đường dẫn trong bucket (giúp truy cập nhanh).
  - **Metadata:** Thông tin mô tả (loại file, ngày tạo, quyền sở hữu).
  - **Version ID:** Mã phiên bản (nếu bật chế độ versioning để theo dõi thay đổi).
- **Prefix (tiền tố):** Cho phép mô phỏng cấu trúc thư mục bằng cách thêm dấu "/" trong tên object, ví dụ: `dataset/train/images/img1.jpg`.

3.4. Các thao tác cơ bản

Một số lệnh thường dùng với AWS CLI:

```
# 1. Tạo bucket mới
aws s3 mb s3://mlops-demo-data --region ap-southeast-1

# 2. Tải file dữ liệu lên S3
aws s3 cp data/train.csv s3://mlops-demo-data/raw/train.csv

# 3. Tải mô hình đã huấn luyện lên
aws s3 cp model/sentiment_model.pt s3://mlops-demo-data/models/

# 4. Liệt kê các file trong bucket
aws s3 ls s3://mlops-demo-data/models/
```

```
# 5. Tải file từ S3 xuống máy cục bộ
aws s3 cp s3://mlops-demo-data/models/sentiment_model.pt ./model.pt
```

3.5. Tình huống sử dụng trong học máy và MLOps

- **Kho dữ liệu Data Lake:** Lưu trữ tập dữ liệu thô (raw data) và dữ liệu đã qua xử lý (processed data) phục vụ huấn luyện mô hình. Ví dụ: tập IMDB reviews được chia thành `train/` và `test/`.
- **Lưu checkpoint và mô hình đã huấn luyện:** Sau khi huấn luyện, mô hình được lưu ở định dạng `.pt`, `.pkl`, hoặc `.onnx` để phục vụ tái huấn luyện và triển khai. Việc lưu trên S3 đảm bảo có thể tái sử dụng mô hình từ bất kỳ môi trường nào (EC2, ECS, SageMaker).
- **Quản lý artifact và log huấn luyện:** MLflow hoặc TensorBoard có thể ghi log, biểu đồ loss/accuracy và upload lên S3 để lưu trữ và giám sát tiến trình huấn luyện.
- **Lưu trữ website tĩnh hoặc tài liệu:** Ngoài dữ liệu ML, S3 còn có thể được dùng để host website tĩnh, tài liệu hướng dẫn hoặc dashboard kết quả.
- **Sao lưu và khôi phục thảm hoạ:** Các file cấu hình hoặc trọng số mô hình quan trọng có thể được sao lưu định kỳ sang bucket Glacier để đảm bảo an toàn lâu dài.

3.6. Tích hợp S3 trong pipeline MLOps

Trong một pipeline học máy điển hình:

1. Nhà khoa học dữ liệu thu thập dữ liệu và tải lên S3.
2. EC2 hoặc SageMaker lấy dữ liệu từ S3 để huấn luyện mô hình.
3. Sau khi huấn luyện xong, mô hình (`model.pt`) được lưu trở lại S3.
4. ECS hoặc Lambda tải mô hình từ S3 để triển khai inference API.
5. CloudWatch giám sát log và lưu báo cáo kết quả huấn luyện/nghiệm thu.

Nhờ tính linh hoạt, độ bền và khả năng tích hợp mạnh mẽ, Amazon S3 trở thành **trung tâm lưu trữ dữ liệu cốt lõi** trong mọi hệ thống MLOps hiện đại.

4. AWS ECR: Kho lưu trữ Docker image

4.1. Khái niệm và vai trò

Amazon ECR (Elastic Container Registry) là dịch vụ quản lý **kho lưu trữ Docker image riêng tư** được cung cấp bởi AWS. Nó đóng vai trò như "GitHub của thế giới container" — nơi bạn có thể đẩy (push) lên các image đã build, và các dịch vụ khác như ECS hoặc EKS có thể kéo (pull) xuống để triển khai.

ECR được thiết kế đặc biệt để tích hợp chặt chẽ với các dịch vụ container của AWS như:

- **ECS (Elastic Container Service)** – để chạy container ở quy mô lớn.
- **EKS (Elastic Kubernetes Service)** – để điều phối container bằng Kubernetes.
- **Lambda** – để chạy function dựa trên container image.

ECR thay thế hoàn hảo cho Docker Hub trong các môi trường sản xuất (production), vì nó:

- Bảo mật cao hơn (tích hợp với IAM, mã hóa dữ liệu khi lưu trữ và truyền tải).
- Dễ kiểm soát truy cập (ai được phép đẩy/pull image).
- Không giới hạn băng thông nội bộ khi truy cập từ dịch vụ AWS cùng vùng (region).
- Hỗ trợ **image scanning** để phát hiện lỗ hổng bảo mật.

4.2. Cấu trúc cơ bản của ECR

ECR tổ chức image theo:

- **Repository (Repo)** – tương tự như "thư mục" chứa nhiều image của một ứng dụng. Ví dụ: `imdb-sentiment`, `ml-inference-api`.
- **Tag** – nhãn xác định phiên bản của image, ví dụ: `:v1.0`, `:latest`, `:test`.
- **Image Digest (SHA)** – mã băm duy nhất nhận dạng image; dùng để đảm bảo tính toàn vẹn.

Một repository có thể chứa nhiều image khác nhau (mỗi image là một phiên bản hoặc môi trường riêng biệt).

4.3. Quy trình thao tác cơ bản

Quy trình làm việc với ECR thường gồm ba bước chính:

1. **Đăng nhập (Authenticate)**: Sử dụng lệnh AWS CLI để đăng nhập ECR, cho phép Docker client xác thực với tài khoản AWS.

```
aws ecr get-login-password --region <REGION> \
| docker login --username AWS --password-stdin \
<ACCOUNT_ID>.dkr.ecr.<REGION>.amazonaws.com
```

2. **Build và gắn nhãn image**: Tạo image từ Dockerfile và gắn nhãn phù hợp với repository của bạn.

```
docker build -t myapp:latest .
docker tag myapp:latest <ACCOUNT_ID>.dkr.ecr.<REGION>.amazonaws.
   com/myapp:latest
```

3. **Đẩy image lên ECR (Push)**: Đưa image lên repository trong ECR để lưu trữ lâu dài và phục vụ triển khai.

```
docker push <ACCOUNT_ID>.dkr.ecr.<REGION>.amazonaws.com/myapp:
   latest
```

4.4. Quy trình chi tiết minh họa

Giả sử bạn có một ứng dụng web AI tên là `sentiment-api`, quy trình cụ thể như sau:

1. Viết Dockerfile để đóng gói ứng dụng và các thư viện cần thiết.
2. Chạy `docker build -t sentiment-api .` để build image cục bộ.
3. Tạo repository trong ECR bằng lệnh:

```
aws ecr create-repository --repository-name sentiment-api
```

4. Gắn nhãn và đẩy image lên ECR:

```
docker tag sentiment-api:latest <ACCOUNT_ID>.dkr.ecr.ap-southeast
   -1.amazonaws.com/sentiment-api:latest
docker push <ACCOUNT_ID>.dkr.ecr.ap-southeast-1.amazonaws.com/
   sentiment-api:latest
```

5. Kiểm tra lại image trong ECR Console để đảm bảo upload thành công.

4.5. Thực hành tốt (Best Practices)

- **Multi-stage build**: Dùng nhiều giai đoạn build để giảm kích thước image cuối cùng (loại bỏ file tạm, thư viện không cần thiết).
- **Gắn nhãn có ý nghĩa**: Ví dụ `:1.0`, `:staging`, `:prod`, hoặc gắn theo mã commit Git (`:2025-10-30-sha1234`).
- **Thiết lập Lifecycle Policy**: ECR cho phép tự động xoá image cũ (ví dụ chỉ giữ lại 10 bản mới nhất).
- **Kích hoạt Image Scan**: Quét tự động để phát hiện lỗ hổng bảo mật trong image.
- **Giới hạn quyền IAM**: Chỉ cho phép nhóm DevOps được phép `push`, các dịch vụ khác (ECS, EKS) chỉ cần quyền `pull`.

4.6. Tích hợp với CI/CD

Trong pipeline CI/CD (ví dụ GitHub Actions hoặc Jenkins), mỗi lần code được cập nhật:

1. Workflow tự động build image mới.
2. Đẩy image lên ECR.
3. ECS hoặc EKS tự động kéo image mới về và triển khai lại mà không gián đoạn dịch vụ.

Nhờ vậy, ECR đóng vai trò là **nút trung gian quan trọng** giữa quá trình phát triển (Dev) và triển khai (Ops) trong MLOps pipeline.

5. AWS ECS: Điều phối container cho dịch vụ ML

5.1. Giới thiệu

AWS ECS (Elastic Container Service) là nền tảng giúp **quản lý và triển khai container Docker** ở quy mô lớn. ECS có thể coi là "bộ não điều khiển" giúp tự động phân phối, mở rộng, và giám sát các container đang chạy trên EC2 hoặc Fargate.

5.2. Hai chế độ hoạt động của ECS

- **EC2 Launch Type**: container chạy trên cụm EC2 do bạn quản lý. Bạn chịu trách nhiệm tạo, bảo trì và tối ưu máy chủ.
- **Fargate Launch Type**: chạy hoàn toàn serverless, AWS tự động cung cấp tài nguyên phù hợp, bạn chỉ định CPU/RAM.

5.3. Các khái niệm cốt lõi

- **Cluster**: Nhóm tài nguyên tính toán (EC2/Fargate) dùng để chạy container.
- **Task Definition**: File JSON định nghĩa chi tiết container – image, tài nguyên (CPU/RAM), port, biến môi trường, IAM role, log driver.
- **Task**: Một bản chạy thực tế của Task Definition.
- **Service**: Đảm bảo số lượng Task luôn duy trì ở mức mong muốn. Nếu một Task bị lỗi, ECS tự động khởi động Task mới.

5.4. Các bước triển khai tối thiểu

1. **Đưa image lên ECR**: build và push image ứng dụng (ví dụ API ML).
2. **Tạo ECS Cluster**: chọn loại (EC2 hoặc Fargate).
3. **Tạo Task Definition**: chỉ định image, tài nguyên, log driver, và quyền IAM.

4. **Tạo Service**: triển khai ít nhất 2 task để đảm bảo sẵn sàng cao; gắn Application Load Balancer nếu cần phân phối tải.
5. **Thiết lập Auto Scaling và CloudWatch**: ECS có thể tự động mở rộng/thu hẹp số lượng container theo tải CPU hoặc độ trễ yêu cầu.

5.5. Lợi ích khi dùng ECS

- Không cần quản lý hạ tầng vật lý – giảm chi phí vận hành.
- Tích hợp chặt với ECR, CloudWatch, IAM, và ALB.
- Dễ dàng mở rộng quy mô ứng dụng chỉ với vài cú click hoặc script.
- Hỗ trợ zero-downtime deployment (triển khai không gián đoạn).

Tổng hợp lại, ECR là nơi lưu trữ ứng dụng (image) và ECS là nơi "vận hành" ứng dụng đó. Hai dịch vụ này kết hợp cùng nhau tạo thành nền tảng vững chắc cho việc triển khai mô hình học máy trong môi trường sản xuất.

6. Ví dụ ứng dụng: Dịch vụ phân tích cảm xúc IMDB

Phần này trình bày một ví dụ cụ thể để minh hoạ quy trình triển khai mô hình học máy trên AWS bằng các thành phần đã học. Ứng dụng được chọn là **dịch vụ phân tích cảm xúc IMDB** (IMDB Sentiment Analysis Service), một hệ thống cho phép người dùng nhập nhận xét (review) phim, và mô hình AI sẽ dự đoán cảm xúc tương ứng là "tích cực" hay "tiêu cực".

6.1. Mục tiêu bài toán

Mục tiêu của hệ thống là:

- Tiếp nhận văn bản đánh giá phim từ người dùng qua giao diện web hoặc API.
- Gửi nội dung này đến mô hình AI đã được huấn luyện trước để dự đoán cảm xúc.
- Trả lại kết quả nhanh chóng, ổn định và có thể mở rộng khi lượng truy cập tăng.

6.2. Kiến trúc tổng quát

Hệ thống được xây dựng theo pipeline chuẩn MLOps trên AWS:

Người dùng → API (Docker Container) → ECR → ECS (Fargate hoặc EC2) → ALB (Load Balancer) → CloudWatch

Dữ liệu và mô hình được lưu trữ an toàn trong **S3**, quyền truy cập được quản lý bằng **IAM**.

6.3. Giải thích chi tiết luồng hoạt động

1. **Lưu trữ dữ liệu và mô hình trên S3:** Tập dữ liệu IMDB (gồm hơn 50.000 đánh giá phim) được lưu trong bucket S3. Sau khi huấn luyện mô hình (ví dụ dùng LSTM hoặc BERT), file trọng số `model.pt` cũng được tải lên cùng bucket để dùng trong giai đoạn phục vụ (inference). S3 đảm bảo dữ liệu luôn sẵn sàng, an toàn, và có thể truy cập từ các dịch vụ khác (EC2, ECS).
2. **Đóng gói ứng dụng bằng Docker:** Một API nhỏ được viết bằng `FastAPI` hoặc `Flask`, có chức năng nhận đầu vào (đoạn văn bản), nạp mô hình từ S3, chạy inference và trả kết quả. File `Dockerfile` được tạo để đóng gói mã nguồn, mô hình và thư viện Python thành một image duy nhất. Ví dụ, trong Dockerfile có thể thêm một endpoint '/health' dùng để kiểm tra tình trạng container (health check).

3. **Lưu trữ image lên ECR:** Sau khi image được build và test thành công, ta gắn nhãn (`tag`) và đẩy lên **Amazon ECR** – nơi lưu trữ Docker image. Điều này giúp đảm bảo mỗi phiên bản ứng dụng có thể được quản lý và triển khai lại dễ dàng trong tương lai.
4. **Triển khai container lên ECS:** ECS được dùng để chạy container từ ECR trên cụm EC2 hoặc môi trường serverless **Fargate**. Mỗi container chạy một bản sao của ứng dụng inference. ECS tự động giám sát tình trạng container, khởi động lại nếu lỗi, và phân phối tải qua nhiều vùng.
5. **Cân bằng tải qua ALB (Application Load Balancer):** Khi người dùng gửi yêu cầu đến API, ALB sẽ tự động định tuyến yêu cầu đến container khỏe mạnh trong cụm ECS. Điều này giúp đảm bảo dịch vụ luôn sẵn sàng kể cả khi có một số container ngừng hoạt động.
6. **Giám sát hệ thống bằng CloudWatch:** CloudWatch thu thập thông tin về CPU, bộ nhớ, thời gian phản hồi và lỗi ứng dụng. Dữ liệu này được hiển thị trong biểu đồ (dashboard) giúp người vận hành phát hiện sớm vấn đề. Khi có cảnh báo (ví dụ: CPU sử dụng vượt 80%), CloudWatch sẽ gửi thông báo đến nhóm quản trị qua email hoặc SNS.
7. **Quản lý bảo mật với IAM:** Mỗi dịch vụ chỉ được cấp quyền tối thiểu. Ví dụ:
   - ECS có quyền `GetObject` để tải mô hình từ S3.
   - CloudWatch có quyền `PutMetricData` để ghi log.
   - Người dùng chỉ được phép gọi API công khai qua ALB mà không truy cập trực tiếp vào container.
8. **Tự động mở rộng (Auto Scaling):** Khi lưu lượng truy cập tăng, ECS tự động nhân thêm container để đáp ứng nhu cầu. Ngược lại, khi tải giảm, hệ thống giảm số container đang chạy để tiết kiệm chi phí.

6.4. Checklist triển khai chi tiết

- **Chuẩn bị mô hình:** Huấn luyện mô hình sentiment analysis trên tập IMDB bằng PyTorch hoặc TensorFlow, lưu `model.pt` lên S3.
- **Tạo ứng dụng API:** Viết ứng dụng FastAPI với hai endpoint:
  - `/predict`: nhận câu đánh giá, trả về kết quả "positive" hoặc "negative".
  - `/health`: kiểm tra tình trạng container (trả về HTTP 200 khi hoạt động tốt).
- **Viết Dockerfile:** Đóng gói toàn bộ mã nguồn và mô hình; dùng lệnh:

```
docker build -t imdb-sentiment:latest .
docker tag imdb-sentiment:latest <ACCOUNT_ID>.dkr.ecr.<REGION>.
    amazonaws.com/imdb-sentiment:latest
docker push <ACCOUNT_ID>.dkr.ecr.<REGION>.amazonaws.com/imdb-
    sentiment:latest
```

- **Triển khai ECS Service:** - Tạo Task Definition, chỉ định image ECR và port (ví dụ 8080). - Tạo Service với ít nhất 2 bản chạy để đảm bảo sẵn sàng cao (High Availability). - Gắn Application Load Balancer để tự động phân phối yêu cầu.
- **Cấu hình Auto Scaling và CloudWatch:** - Tạo chính sách mở rộng khi CPU vượt 70%. - Thiết lập cảnh báo (alarm) gửi thông báo khi hệ thống lỗi hoặc chậm. - Tạo dashboard hiển thị lưu lượng và độ trễ API.
- **Kiểm thử tải (Load Test):** Sử dụng `Apache JMeter` hoặc `Locust` để mô phỏng nhiều người dùng cùng gửi yêu cầu. Kiểm tra độ trễ trung bình, khả năng chịu tải và sự ổn định khi mở rộng tự động.
- **Đánh giá kết quả:** Khi triển khai thành công, người dùng có thể gửi câu: `"The movie was amazing!"` → Hệ thống trả về `"Positive"` trong vài trăm mili-giây. Toàn bộ log và

chỉ số được lưu trong CloudWatch để phục vụ giám sát liên tục.

6.5. Lợi ích của kiến trúc này

- Không cần quản lý thủ công máy chủ – sử dụng Fargate hoặc auto-scaling EC2.
- Mô hình được đóng gói sẵn, dễ dàng cập nhật và rollback phiên bản.
- Dịch vụ hoạt động ổn định, sẵn sàng cao, dễ bảo trì.
- Tối ưu chi phí – chỉ tính phí khi container đang chạy.
- Đảm bảo an toàn và kiểm soát truy cập hiệu quả thông qua IAM.

Kết quả cuối cùng là một dịch vụ AI hoàn chỉnh, có thể mở rộng từ mô hình cá nhân đến quy mô doanh nghiệp, minh chứng cho sức mạnh kết hợp giữa **MLOps và AWS Cloud**.

7. Gộp lại: Pipeline MLOps hoàn chỉnh trên AWS

Phần này tổng hợp toàn bộ các thành phần dịch vụ AWS đã giới thiệu ở trên thành một **pipeline MLOps đầy đủ**, mô tả cách dữ liệu, mô hình và ứng dụng di chuyển xuyên suốt trong hệ thống từ giai đoạn thu thập đến triển khai.

7.1. 1. Tổng quan luồng hoạt động

Pipeline MLOps trên AWS được thiết kế nhằm đảm bảo ba yếu tố: **tự động hoá, khả năng mở rộng và tính tái lập**. Các bước luồng chính gồm:

1. Thu thập và lưu trữ dữ liệu thô (raw data) trên **S3**.
2. Tiền xử lý, huấn luyện và đánh giá mô hình trên **EC2** hoặc SageMaker.
3. Đóng gói mô hình thành container Docker, lưu image tại **ECR**.
4. Triển khai dịch vụ inference (dự đoán) bằng **ECS**, có thể mở rộng tự động.
5. Giám sát hiệu suất và cảnh báo qua **CloudWatch**.
6. Quản lý truy cập người dùng, nhóm, vai trò với **IAM**.
7. Tích hợp chu trình tự động hoá build – test – deploy qua **CI/CD pipeline**.

7.2. 2. Vai trò chi tiết của từng thành phần

Thành phần	Vai trò và mô tả chi tiết
S3 (Simple Storage Service)	• Là trung tâm lưu trữ dữ liệu trong toàn bộ pipeline. • Lưu **dataset thô, dữ liệu đã xử lý, kết quả huấn luyện, file mô hình (model artifacts)**, cũng như log huấn luyện và file cấu hình. • Có thể tổ chức theo cấu trúc: `raw/`, `processed/`, `models/`, `logs/`. • Kết hợp cùng **IAM policy** để kiểm soát quyền đọc/ghi theo vai trò (developer, analyst, model trainer).

EC2 (Elastic Compute Cloud)	• Cung cấp môi trường tính toán linh hoạt để huấn luyện mô hình học máy. • Có thể chọn loại instance GPU (như P5, G6) để tăng tốc huấn luyện deep learning. • Sau khi huấn luyện xong, EC2 có thể được tắt để tiết kiệm chi phí, và toàn bộ kết quả được lưu ngược lên S3. • Cũng có thể dùng EC2 cho inference nội bộ (tạm thời) trước khi đưa mô hình sang ECS phục vụ quy mô lớn.
ECR (Elastic Container Registry)	• Là kho lưu trữ image Docker riêng tư, nơi chứa phiên bản container của mô hình sau khi đóng gói. • Mỗi image gồm code, model weights, dependencies (thư viện), và API phục vụ inference (ví dụ Flask/FastAPI). • Cho phép ECS hoặc các hệ thống CI/CD tự động kéo image mới nhất về để triển khai.
ECS (Elastic Container Service)	• Điều phối các container được lưu trong ECR. • Triển khai mô hình AI thành dịch vụ inference (REST API hoặc gRPC). • Quản lý số lượng bản chạy (task), tự động mở rộng (Auto Scaling) khi tải tăng. • Kết hợp với **Application Load Balancer (ALB)** để chia tải và đảm bảo không có thời gian chết (zero downtime).
IAM (Identity and Access Management)	• Đảm bảo an ninh và phân quyền trong toàn bộ pipeline. • Quản lý quyền của người dùng, nhóm, và dịch vụ AWS (ví dụ: EC2 chỉ có quyền đọc bucket S3, ECS chỉ có quyền kéo image từ ECR). • Thực hiện theo nguyên tắc **Least Privilege** – cấp quyền ở mức tối thiểu cần thiết.
CloudWatch	• Giám sát và thu thập các chỉ số hoạt động của từng thành phần (CPU, RAM, network, latency). • Theo dõi log ứng dụng (ví dụ lỗi inference, thời gian phản hồi, thông số huấn luyện). • Có thể thiết lập cảnh báo (alarm) tự động gửi email/SNS khi có bất thường (mô hình giảm chính xác, service downtime).

CI/CD Pipeline	• Tự động hoá quy trình phát triển: build code, test, đóng gói container và triển khai. • Một kịch bản phổ biến: 1. Code mới được đẩy lên GitHub. 2. GitHub Actions build và đẩy image mới lên ECR. 3. ECS tự động cập nhật dịch vụ bằng image mới mà không cần dừng hệ thống. • Đảm bảo việc triển khai mô hình hoặc ứng dụng mới diễn ra **liên tục, nhanh và an toàn**.

7.3. 3. Luồng tương tác tổng thể

- Dữ liệu được tải lên và lưu trữ lâu dài trong S3.
- EC2 truy cập dữ liệu này để huấn luyện mô hình, sau đó đẩy file trọng số lên lại S3.
- Mô hình và API được đóng gói thành Docker image và đưa lên ECR.
- ECS lấy image từ ECR, triển khai trên cụm container, kết nối qua ALB để phục vụ người dùng cuối.
- CloudWatch theo dõi và cảnh báo mọi vấn đề hiệu suất.
- CI/CD bảo đảm chu trình huấn luyện–triển khai lặp lại nhanh chóng.

7.4. 4. Ưu điểm của pipeline này

- **Tự động hoá hoàn toàn**: giảm công sức triển khai thủ công.
- **Mở rộng linh hoạt**: dễ tăng/giảm tài nguyên theo tải.
- **Chi phí tối ưu**: chỉ trả cho tài nguyên khi dùng.
- **Đảm bảo an toàn và giám sát liên tục**.
- **Dễ bảo trì**: mỗi thành phần tách biệt, dễ thay thế hoặc nâng cấp.

7.5. 5. Ứng dụng thực tế

Mô hình pipeline này có thể áp dụng cho:

- Hệ thống phân tích hình ảnh y tế (MRI/CT) – dữ liệu trên S3, huấn luyện GPU EC2, inference bằng ECS.
- Dịch vụ chatbot hoặc NLP – mô hình BERT đóng gói, triển khai qua ECR/ECS.
- Nền tảng khuyến nghị sản phẩm – pipeline CI/CD cập nhật mô hình mỗi ngày.

Như vậy, AWS cung cấp đầy đủ các mảnh ghép để xây dựng một hệ thống MLOps hiện đại: từ hạ tầng tính toán (EC2), lưu trữ (S3), triển khai (ECR/ECS), đến giám sát (CloudWatch) và bảo mật (IAM).