Day 19 — Logging & Monitoring ML API with AWS CloudWatch

Goal

Capture all Flask app logs inside the Docker container and send them to AWS CloudWatch Logs for real-time monitoring of API activity and errors.

Steps

Step 1 — Install CloudWatch Agent on EC2

```
sudo apt update
sudo apt install amazon-cloudwatch-agent -y
```

Step 2 — Create IAM Role for EC2

1. Go to AWS Console → IAM → Roles → Create Role
2. Choose **EC2**
3. Attach policy → **CloudWatchAgentServerPolicy**
4. Name it → **EC2CloudWatchRole**
5. Attach this role to your EC2 instance (EC2 → Instances → Select → Actions → Security → Modify IAM Role)

Step 3 — Configure CloudWatch Agent

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-config-wizard
```

- Do you want to collect logs? → **Yes** - Path to log files? → `/var/log/titanic.log` - Save config → `/opt/aws/amazon-cloudwatch-agent/bin/config.json`

Start and enable agent:

```
sudo systemctl start amazon-cloudwatch-agent
sudo systemctl enable amazon-cloudwatch-agent
```

Step 4 — Update Flask App for Logging (`app.py`)

```
import logging
from flask import Flask, request, jsonify
import joblib

app = Flask(__name__)
model = joblib.load("titanic_rf_model.pkl")

# Configure logging
logging.basicConfig(filename="/var/log/titanic.log", level=logging.INFO,
                    format="%(asctime)s - %(levelname)s - %(message)s")

@app.route("/predict", methods=["POST"])
def predict():
    try:
        data = request.get_json()
        features = data["features"]
        prediction = model.predict([features])[0]
        logging.info(f"Prediction made: Input={features}, Output={prediction}")
        return jsonify({"prediction": int(prediction)})
    except Exception as e:
        logging.error(f"Error: {str(e)}")
        return jsonify({"error": str(e)}), 400

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

Step 5 — Rebuild Docker Image

```
docker build -t titanic-flask-app:v2 .
docker run -d --name titanic-logging -p 5000:5000 titanic-flask-app:v2
```

Step 6 — Test API

```
curl -X POST http://<ec2-ip>:5000/predict -H "Content-Type: application/json" -
d "{\"features\": [3,1,22,0,0,7.25]}"
```

Check logs inside EC2:

```
cat /var/log/titanic.log
```

Step 7 — Verify in AWS CloudWatch

- Go to AWS Console → CloudWatch → Logs → Log groups
- You should see your log group with Titanic API logs.

Sample API Test Inputs

```
curl -X POST http://<ec2-ip>:5000/predict -H "Content-Type: application/json" -
d "{\"features\": [1,0,38,1,0,71.28]}"
curl -X POST http://<ec2-ip>:5000/predict -H "Content-Type: application/json" -
d "{\"features\": [3,1,26,0,0,7.88]}"
curl -X POST http://<ec2-ip>:5000/predict -H "Content-Type: application/json" -
d "{\"features\": [2,1,35,1,0,53.1]}"
curl -X POST http://<ec2-ip>:5000/predict -H "Content-Type: application/json" -
d "{\"features\": [3,0,27,0,0,8.05]}"
curl -X POST http://<ec2-ip>:5000/predict -H "Content-Type: application/json" -
d "{\"features\": [1,0,54,0,2,51.86]}"
```

Key Takeaway

Real-time logging and monitoring make ML APIs production-ready, helping track predictions and quickly debug errors.