Project Brief: PathHelm

- **Project Name:** PathHelm
- **One-Line Description:** An intelligent, AI-powered API gateway and security proxy designed to protect and control web traffic.
- **Core Technologies:** Python, FastAPI, Docker, Docker Compose, scikit-learn, pandas.

Project Status (as of July 21, 2025)

We have completed the foundational work and the first stages of development.

- **Phase 0: Foundation & Setup**
  - Chose the name "PathHelm".
  - Created a GitHub repository.
  - Established the project file structure (`app/`, `tests/`, etc.).
- **Phase 1: The Core Proxy**
  - Built the basic API gateway using FastAPI to forward requests to a mock backend service.
  - Successfully containerized the application (`pathhelm`) and the `mock-backend` using Docker and `docker-compose.yml`.
- **Phase 2: The Simple Guard**
  - Implemented a simple, in-memory rate limiter based on IP address to block users making too many requests (e.g., >20 requests in 60 seconds).
- **Phase 3A: Generate Training Data**
  - Created a Python script (`create_dataset.py`) to generate a sample dataset (`api_traffic_data.csv`) of normal and anomalous API traffic features.

Current Code Snippets

`Dockerfile`

Dockerfile

```
FROM python:3.11-slim

WORKDIR /code
COPY ./requirements.txt /code/requirements.txt
RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt
COPY ./app /code/app

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

`docker-compose.yml`

YAML

```
version: '3.8'

services:
  pathhelm:
    build: .
    ports:
      - "8000:8000"
    depends_on:
      - mock-backend

  mock-backend:
    image: python:3.11-slim
    command: >
      sh -c "pip install Flask && python /app/mock_service.py"
    volumes:
      - ./mock_service.py:/app/mock_service.py
```

`requirements.txt`

```
fastapi==0.111.0
uvicorn[standard]==0.29.0
requests==2.32.3
pandas==2.2.2
scikit-learn==1.5.0
```

`app/main.py` (with rate limiter)

Python

```
import requests
import time
from collections import defaultdict
from fastapi import FastAPI, Request, Response

RATE_LIMIT_THRESHOLD = 20
RATE_LIMIT_TIMEFRAME = 60
request_logs = defaultdict(list)

TARGET_SERVICE_URL = "http://mock-backend:5000"

app = FastAPI(title="PathHelm Gateway")

@app.api_route("/{path:path}", methods=["GET", "POST", "PUT", "DELETE",
"PATCH"])
async def proxy(request: Request, path: str):
    client_ip = request.client.host
    current_time = time.time()

    request_logs[client_ip] = [t for t in request_logs[client_ip] if
current_time - t < RATE_LIMIT_TIMEFRAME]

    if len(request_logs[client_ip]) >= RATE_LIMIT_THRESHOLD:
        return Response(content="Too Many Requests", status_code=429)

    request_logs[client_ip].append(current_time)

    try:
        response = requests.request(
            method=request.method,
            url=f"{TARGET_SERVICE_URL}/{path}",
            headers={k: v for k, v in request.headers.items() if k.lower() !=
'host'},
            params=request.query_params,
            data=await request.body(),
            stream=True
        )
        return Response(
            content=response.content,
            status_code=response.status_code,
            headers=dict(response.headers)
        )
    except requests.exceptions.RequestException as e:
        return Response(content=f"An error occurred while proxying: {e}",
status_code=502)
```

The Roadmap Ahead

- **Current Step: Phase 3B: Train the AI Model**
  - Use a Jupyter Notebook to load `api_traffic_data.csv`.
  - Train an `IsolationForest` model from `scikit-learn` to distinguish between normal (0) and anomalous (1) traffic.
  - Save the trained model to a file (e.g., `model.pkl`).
- **Upcoming Steps:**
  - **Phase 3C: Integrate the AI Model:** Load the saved `model.pkl` into the FastAPI application. Replace the simple rate-limiting logic with predictions from the model to decide whether to block a request.
  - **Phase 4: Management & Analytics API:** Create new internal endpoints for PathHelm (e.g., `/status`, `/analytics`) to see how many requests have been processed, blocked, etc.
  - **Phase 5: Documentation & Release:** Create a high-quality `README.md` on GitHub explaining what PathHelm is, how to use it, and how to configure it.