BUILD & DEPLOYMENT INSTRUCTIONS

Prerequisites

1. Python (3.8 or higher recommended)

2. pip (Python package manager)

3. Virtual Environment (recommended, but optional)

4. Access to the required APIs (Make sure the URLs in billing_api_url and usage_api_url are correct and accessible).

Step 1: Set Up the Project Environment

1. Clone or create a project directory:

mkdir billing-usage-forecasting

cd billing-usage-forecasting

2. Create and activate a virtual environment:

python3 -m venv venv

source venv/bin/activate # On Windows, use venv\Scripts\activate

3. Install dependencies: Create a requirements.txt file to list the dependencies. Add the following lines to it:

pandas

requests

scikit-learn

prophet # Or 'fbprophet' if using an older version

Then, install the dependencies:

pip install -r requirements.txt

Step 2: Set Up API Authentication (If Needed)

If the APIs require authentication, set up environment variables or configuration files for storing the API keys securely.

1. Environment Variables (Recommended):

In your terminal:

export BILLING_API_URL="https://api.vultr.com/v1/billing"

export USAGE_API_URL="https://api.vultr.com/v1/usage"

export API_KEY="your_api_key" # Replace with your actual API key if needed

2. Configuration File:

Alternatively, you can create a .env file in your project directory:

BILLING_API_URL=https://api.vultr.com/v1/billing

USAGE_API_URL=https://api.vultr.com/v1/usage

API_KEY=your_api_key

Then, use the dotenv library to load this file in your code (you'd need to add python-dotenv to requirements.txt and import it in your script).

Step 3: Run the Application Locally

1. Run the script:

python your_script_name.py

2. Verify Output: Ensure each module (DataProcessor, CostForecaster, AnomalyDetector, and RealTimeOptimizer) prints expected results.

Step 4: Dockerize the Application (Optional but Recommended)

To make deployment easier, you can package your application in a Docker container.

1. Create a Dockerfile in your project directory:

Use an official Python runtime as a parent image

FROM python:3.8-slim

Set the working directory

WORKDIR /app

Copy the current directory contents into the container at /app

COPY . /app

Install dependencies

RUN pip install --no-cache-dir -r requirements.txt

Run the script

CMD ["python", "your_script_name.py"]

2. Build the Docker image:

docker build -t billing-usage-forecasting .

3. Run the Docker container:

docker run --env BILLING_API_URL=$BILLING_API_URL --env USAGE_API_URL=$USAGE_API_URL --env API_KEY=$API_KEY billing-usage-forecasting

---

Step 5: Deploy the Application

Choose a cloud provider (e.g., AWS, Azure, Google Cloud, DigitalOcean) or a platform-as-a-service (PaaS) provider (e.g., Heroku) to deploy your Dockerized app.

Here’s an example for deploying on Heroku:

1. Log in to Heroku:

heroku login

2. Create a Heroku app:

heroku create billing-usage-forecasting

3. Set environment variables on Heroku:

heroku config:set BILLING_API_URL=https://api.vultr.com/v1/billing

heroku config:set USAGE_API_URL=https://api.vultr.com/v1/usage

heroku config:set API_KEY=your_api_key

4. Deploy the Docker container to Heroku:

heroku container:push web --app billing-usage-forecasting

heroku container:release web --app billing-usage-forecasting

5. Access your application:

Visit the URL Heroku provides for your app (e.g., https://billing-usage-forecasting.herokuapp.com/).