Predictive Auto Scaling using AWS CloudWatch and Flask

Author: Anurag Thakur

Abstract

This project demonstrates a predictive auto-scaling system built using AWS EC2, CloudWatch, and Flask. The system monitors CPU utilization metrics, predicts future usage trends, and provides scaling recommendations based on machine learning models. The approach helps optimize resource allocation while maintaining performance and cost efficiency.

Introduction

Auto-scaling in cloud computing allows infrastructure to dynamically adjust capacity based on workload demands. Traditional scaling methods react to real-time metrics, while predictive scaling uses historical data to anticipate future needs. In this project, AWS CloudWatch metrics are analyzed using Python and Flask to build a lightweight predictive auto-scaler.

Objective

The objective of this project is to design and implement a predictive auto-scaling system that analyzes AWS EC2 instance performance metrics and provides scaling recommendations based on forecasted resource utilization.

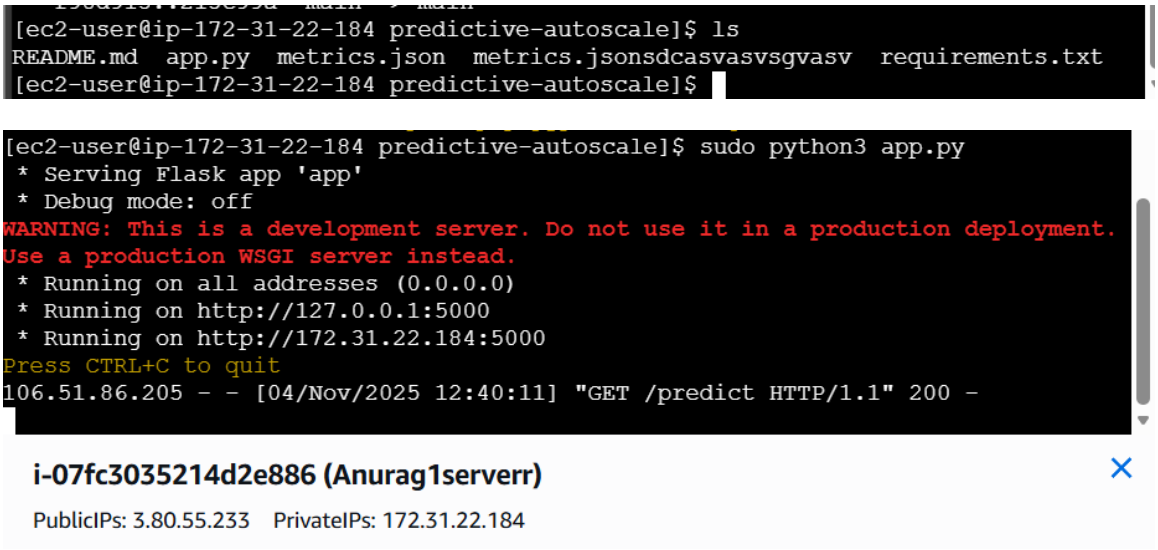
Methodology

1. Launch an AWS EC2 instance with the required permissions.
2. Configure AWS CLI and CloudWatch access.
3. Collect real-time CPU utilization metrics using CloudWatch APIs.
4. Use Python libraries such as NumPy and Flask to process and serve data.
5. Develop a Flask application to display current and predicted metrics.
6. Implement a simple prediction algorithm to recommend scaling actions.

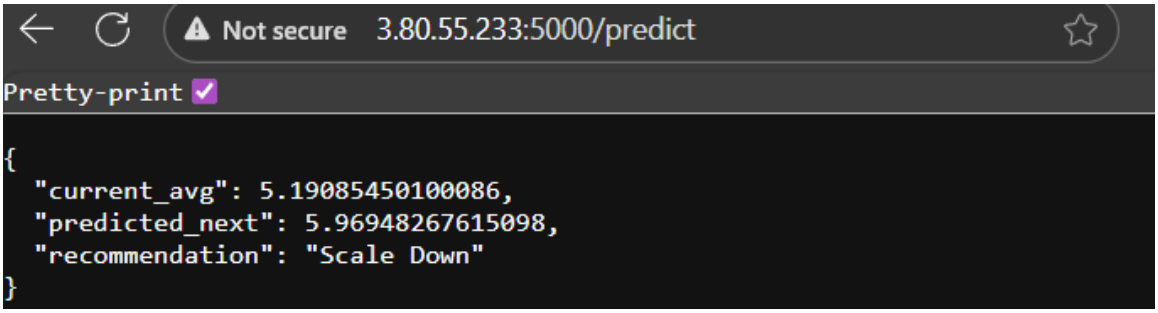
Results

Below are the main outputs and screenshots from the project:

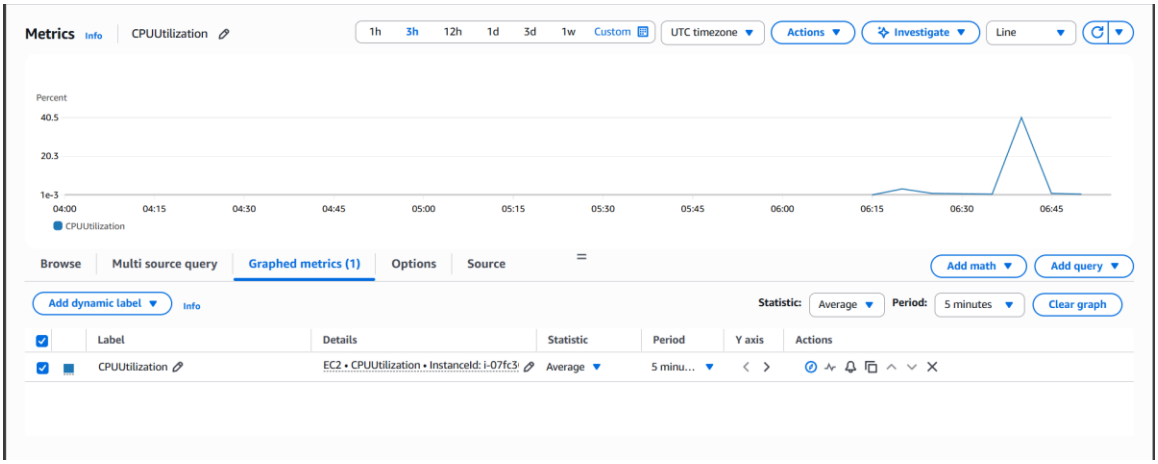
1. AWS EC2 Instance Screenshot:



2. Flask Application Prediction Output:



3. CPU Utilization Graph (CloudWatch):



Conclusion

The predictive auto-scaling system successfully demonstrates how machine learning and cloud monitoring tools can be integrated to improve cloud infrastructure efficiency. By

forecasting CPU utilization trends, it enables smarter scaling decisions and reduces unnecessary costs.

Future Work

Future improvements can include:
- Implementing advanced forecasting algorithms like ARIMA or LSTM.
- Automating scale-up/scale-down actions directly using AWS APIs.
- Integrating a user dashboard for real-time visualization and control.

References

- AWS CloudWatch Documentation
- Flask Web Framework
- NumPy and Pandas Python Libraries
- AWS EC2 User Guide