# REPORT NAME

Yung Chun Chiang, 6812217, yc01734@surrey.ac.uk

https://stock_system.appspot.com

***Abstract*— The project demonstrates a scalable system architecture for a financial analysis system which is based on AWS Lambda, EC2 and Google App Engine (GAE). The main analysis target of the system is analysis the features of stock trading. It can perform risk analysis and profit analysis of the stock (MSFT), specially focue on Value at Risk and Profit and Loss calculations and also included the trading strategies according to the Monte-Carlo method. This report will discuss the relation between NIST SP 800-145 and the system architecture of the colud-native API.**

***Keywords— cloud-native API, GAE, AWS Lambda, EC2, risk analysis, Monte-Carlo method***

## I. INTRODUCTION

NIST SP 800-145 is a guide. NIST is a non-regulatory agency under the U. S. Department of Commerce, and its mission is to provide the best practices for security, privacy, and resiliency. One of the main roles of NIST is to create standards that shall be used to protect any information during the generation and processing of data, and it should also include minimal requirements to be followed during such processing. This paper is devised to establish a bona fide concept of cloud computing and lay the groundwork for it in every sense with the help of the constituent properties, models, types of cloud computing services, and deployment models. The framework is comprised of the NIST policy document concerning the Development and User parts, respectively.

### *A. Developer*

| In NIST SP800-145 | Developer uses and experiences |
|---|---|
| **Standardized Definition** | It provides the standard requirements in cloud computing, such as the fiveessential characteristics. As a developer, adhering to these definitions ensures clarity and consistency in the design and implementation of the system, especially when handling multiple service models. |
| **Models Classification** | It is used as software as a service (SaaS), and the document states the scope of the service. Clearly, it features the corresponding applications or components(e.g., server, OS, storage), whether a given service provider or we manage them. |

### *B. User*

| In NIST SP800-145 | User uses and experiences |
|---|---|
| **On-demand self-service** | As mentioned before, it is categorized as a SaaS. Therefore, the user can be satisfied with calculating the risk through the use of various cloud resources without the need for human interactions with the cloud service provider using manual approaches. |
| **Network access** | The user can use different vital services, by inputting the needed measures. It is accessible and usable by anyone who has the required credential details. |

## II. FINAL ARCHITECTURE

This project presents a scalable and resilient architecture for a financial analysis system, leveraging AWS EC2, AWS Lambda, and Google App Engine (GAE) to meet the requirements. The system primarily focuses on experimenting with the Three White Soldiers/Three Black Crows trading patterns to model and predict their behavior. It accepts and produces JSON format for inputs and outputs, ensuring seamless integration and data interchange. The architecture is categorized into two main sections: system initialization and risk analysis. During system initialization, the warmup endpoint allocates resources before they are needed to minimize user wait time, taking two parameters: `s` (selecting a scalable service between EC2 and Lambda) and `r` (scale-out number). Only one service operates at a time to ensure efficient resource utilization. The scaled ready endpoint checks if the selected instance is running, returning `warm: True` if the instance is on, otherwise `warm: False`. The get warmup cost endpoint calculates operational costs by multiplying the cost per minute (0.00001667) by the billable time, considering the scale-out number `r` in its calculation. The get endpoints endpoint retrieves the required endpoint directly.

In the risk analysis section, the analyze endpoint performs risk analysis using parameters: `h` (history price length), `d` (number of data points), `t` (buy/sell), and `p` (price difference check over a given number of days), and calculates VaR values based on 95% and 99% confidence intervals for the specified signals. The get average VaR endpoint calculates the average VaR values from the total value pairs gathered. The get sig profit loss endpoint computes profit/loss for each signal by dividing each signal's buy price by its sell price. The get total profit loss endpoint calculates the overall profit/loss by summing all the positive and negative numbers in the profit/loss array. The get chart URL endpoint generates a URL for charts created with the calculated risk and profit/loss values, including Average 95% Risk, Average 99% Risk, 95% Risk Values, and 99% Risk Values. The get time cost endpoint provides the total time and expense incurred during the analysis. The get audit endpoint outputs a comprehensive summary including charts, time, cost, profit/loss, average profit/loss, VaR, and average VaR. The reset endpoint clears all previously produced values and sets them to none. The terminate endpoints terminate each Lambda function and EC2 instance, confirming no functions or instances are in use. This architecture combines AWS EC2, AWS Lambda, and GAE to offer a robust, scalable, and efficient system for financial risk analysis. By focusing on the Three White Soldiers/Three Black Crows trading patterns, the system provides an optimal user experience while simplifying the measurement of risks and costs in predicting share prices.

This financial system architecture have two mention parts, system initialization and risk analysis.

System initialization:
1. Warmup Endpoint
2. Scaled Ready Endpoint
3. Get Warmup Cost Endpoint
4. Get Endpoints Endpoint

Risk Analysis:
1. Analyze Endpoint
2. Get Average VAR Endpoint
3. Get Sig Profit Loss Endpoint
4. Get Total Profit Loss Endpoint
5. Get Chart URL Endpoint
6. Get Time Cost Endpoint
7. Get Audit Endpoint
8. Reset Endpoint
9. Terminate Endpoint

## III. Satisfaction of Requirements

The code has four main sections: initialization, risk analysis, rest, and termination. I used Lab 4 as a guide for the initialization section because it has a similar layout and explains how to

TABLE I. Satisfaction of Requirements/Endpoints and Code Use/Creation

| | ***MET*** | ***PARTIALLY MET*** | ***NOT MET*** |
|---|---|---|---|
| Requirements | i. ii. iii iv. | | |
| Endpoints | /warmup<br>/resources_ready<br>/get_warmup_cost<br>/get_endpoints<br>/analyze<br>/get_sig_vars9599<br>/get_avg_vars9599<br>/get_sig_profit_loss<br>/get_tot_profit_loss<br>/get_chart_url<br>/get_time_cost<br>/get_audit<br>/reset<br>/terminate<br>/resources_terminated | | |

## IV. Results

The entire version should include 32 columns with each combination of s, r, h, d, and p, but as we can see from the table output, there are only 8 columns with different combinations of s, r, and d. As a result, this table nevertheless provides us with some critical observations. The average profit mostly depends on p; a larger profit can be achieved with longer days of p. VaR is based on d, influencing their standard deviation and mean. As the time and cost show, there are significant distinctions between Lambda and EC2. Due to the shorter duration and lower billable time, lambda is significantly less expensive.
*p=7 for rows that are highlighted, p=30 by default

Table II results

| *s* | *r* | *d* | *Avg Profit* | *Avg95 %* | *Avg99 %* | *Time* | *Cost* |
|---|---|---|---|---|---|---|---|
| EC2 | 1 | 20000 | 3203 | -4.31% | -7.42% | 110.53 | 0.025432 |
| EC2 | 4 | 10000 | 3304 | -4.32% | -7.41% | 110.72 | 0.02783 |
| EC2 | 4 | 10000 | 823 | -5.36% | -8.43% | 111.33 | 0.025485 |
| EC2 | 4 | 20000 | 824 | -5.35% | -8.47% | 112.83 | 0.024105 |
| Lambda | 1 | 10000 | 3404 | -4.28% | -8.21% | 10.38 | 0.000192 |
| Lambda | 1 | 20000 | 3504 | -4.29% | -7.24% | 9.81 | 0.000163 |
| Lambda | 4 | 10000 | 953 | -5.36% | -6.77% | 12.35 | 0.000147 |
| Lambda | 4 | 20000 | 942 | -5.36% | -7.69% | 8.93 | 0.000149 |

## V. Costs

The location, RAM capacity, and CPU characteristics all affect the price. I will, therefore, specify the expenses in light of the particulars of this application. Two categories of services are offered: EC2 and AWS Lambda.

AWS Lambda:

The region is us-east-1, and the processor is x86 with 128MB of RAM. The fee is $0. Companies providing support for Tbps networks will charge around $0. 0000166667 per GB-second for services up to the first 6 billion GB-seconds used monthly. The price per month will fall up to a few billion gigabytes data, after which the price will drop correspondingly.

EC2:

In addition to the area, EC2 needs the operating system and instance type to be specified [1]. In this instance, Linux is the operating system, and t3.mirco is the system's instance. It costs $0.0116 per hour and offers EBS storage and 1GB of memory.

## REFERENCES

[1[Peter Mell and Timothy Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, Gaithersburg, September 2011. [online] Avaliable at: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800145.pdf