# Deploy Machine Learning Pipeline on Google Cloud Platform

## Cloud Computing Project

Andrea Ciccotti - Andrea Bernini - Donato Francesco Pio Stanco

# Introduction

# The scenario

We imagined the following real scenario:

- a user enters a series of data about a football match
    - the match date, the name of the home team, and the away team.
- Once these parameters have been entered, they will be processed and a prediction of the outcome of the match will be provided in the output, i.e., victory of the home team, draw, or away team.

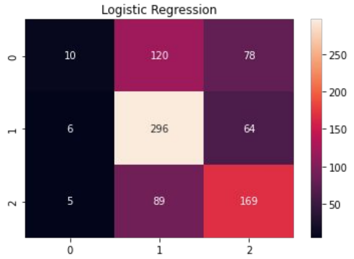
# Tools

- **Flask**,  for the creation of the web app through the use of Python, HTML, and CSS code.
- **Docker**, for the creation of a containerized environment in which to run our application.
- **Kubernetes**, for the management of containerized applications, or to completely manage their life cycle.
- **Google Cloud Platform** (GCP), provides Google Kubernetes Engine which is an implementation of the open source Kubernetes framework, as well as various cloud computing services.

# Implementation

# Machine Learning Model

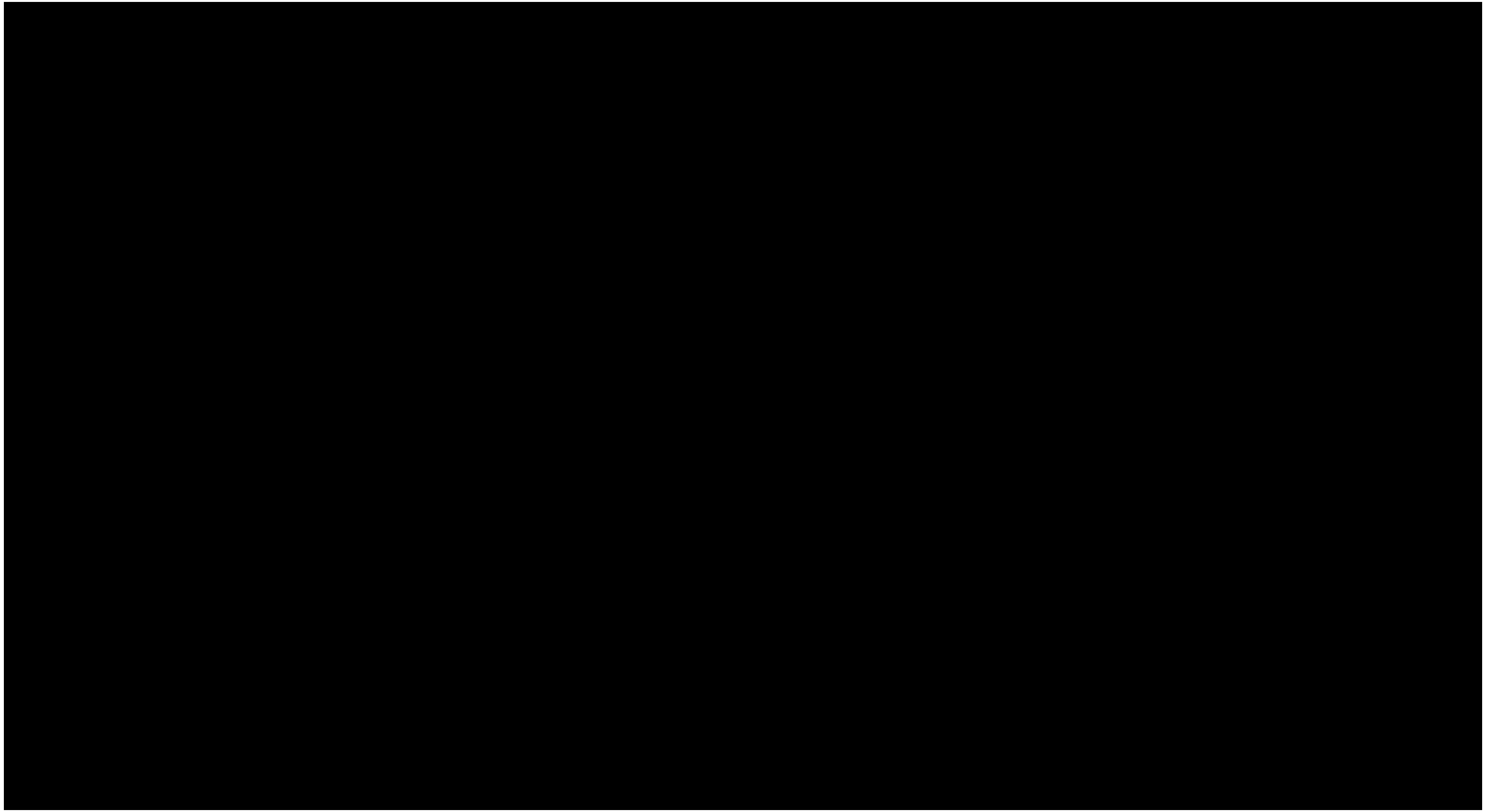
- We used a dataset that contains the last 10 season of Serie A.
    - 70 features and approximately 4000 rows.
- During the **preprocessing** phase:
    - Extrapolation of new features, using the existing ones, to avoid **overfitting.**
    - Transformation of the categorical features using a JSON to save the `string:number`combinations.
- We have tested several models locally (on Google Colab), and have chosen **Logistic Regression.**
    - it has the best trade-off between *Accuracy* and *Prediction Time.*



# Web-App

Our web-app can be divided into two parts:

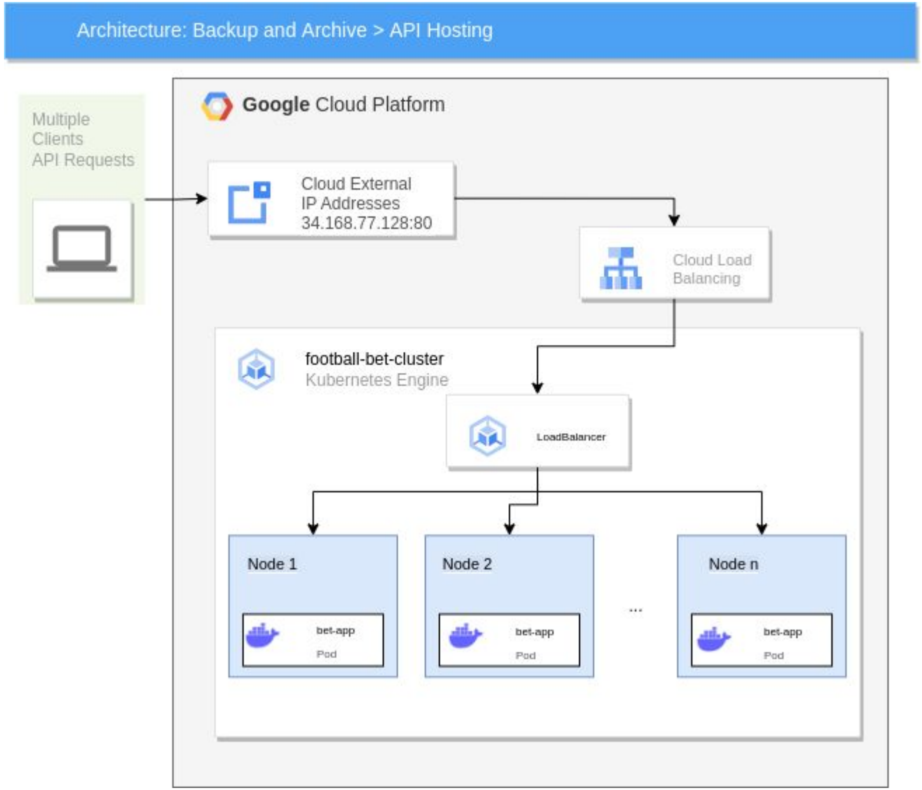
- **Front-end**, designed using HTML and CSS.
  - in which we have a FORM where the user can enter the data on which he wants to obtain a prediction.
- **Back-end**, developed using *Flask*, to render HTML code and manage HTTP Methods.
  - The main web-path of our application is `/predict` which allows to manage POST calls obtaining a prediction of the result.



Football-Bet Web App

# Deploy Web-App on GKE

1. Create a **Cluster** called *football-bet-cluster*, on **Google Kubernetes Engine.**
2. Build **Docker Image**, and upload it to **Google Container Registry**.
3. Deploy the image on *football-bet-cluster* in order to distribute and manage the application.
4. Finally expose the application, http://34.154.93.188:80.



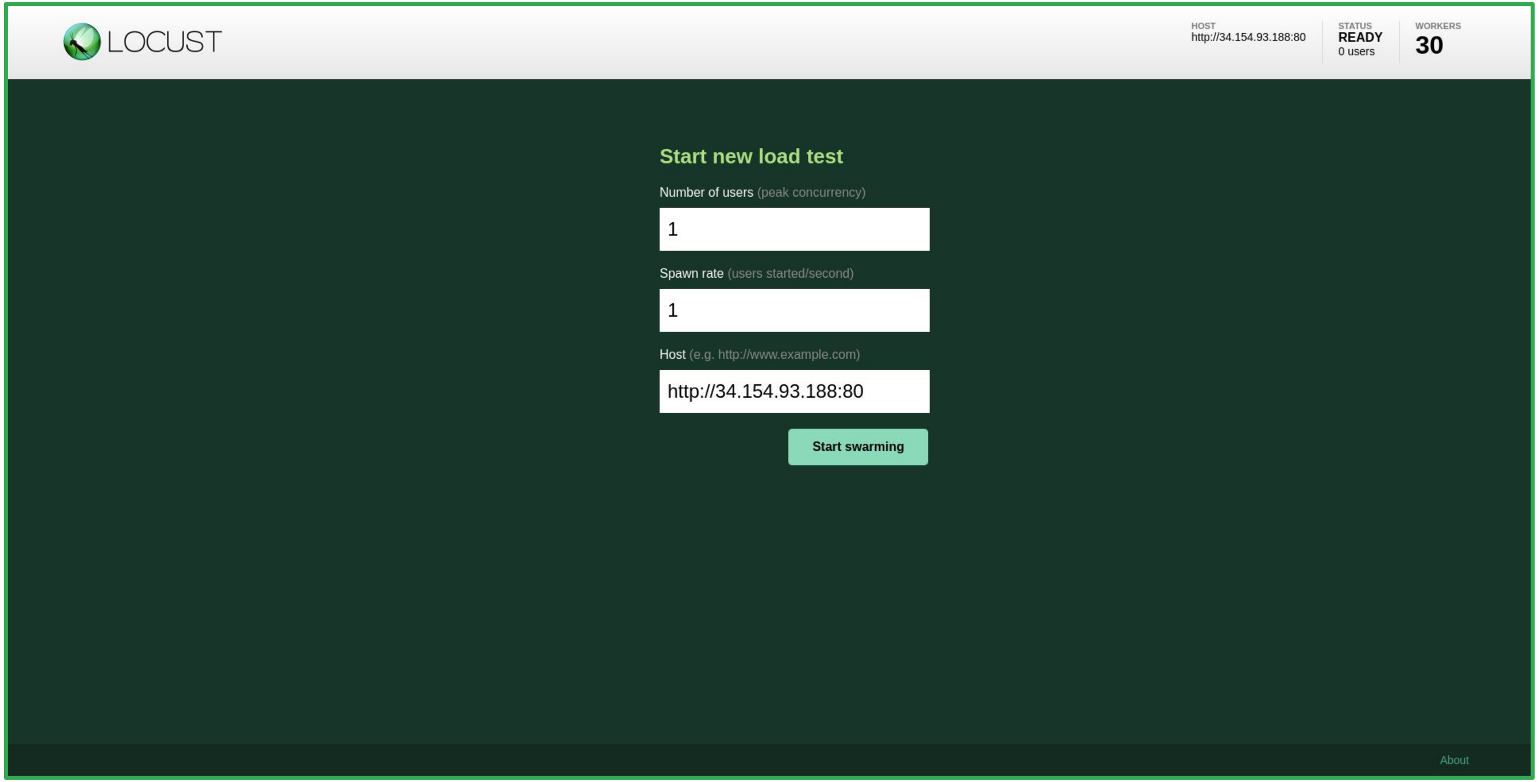

# Web App Architecture

# Load Test

# Simulation with Locust

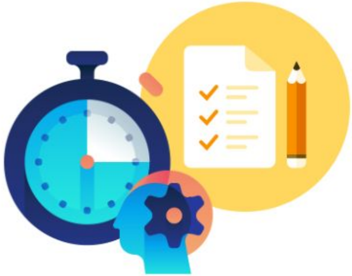
We used Locust to perform load test:

- it gives you the ability to spread your performance tests across multiple machines.
- the architecture involves **two main components**:
  - Locust Docker container image
  - Container orchestration and management mechanism
- to perform load test we created a separate cluster called *loadtesting*

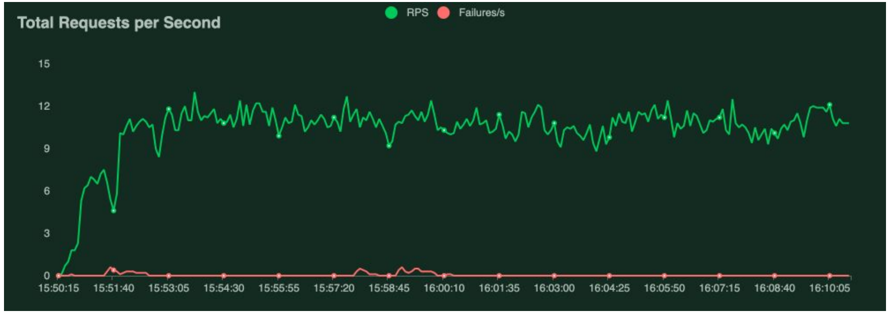


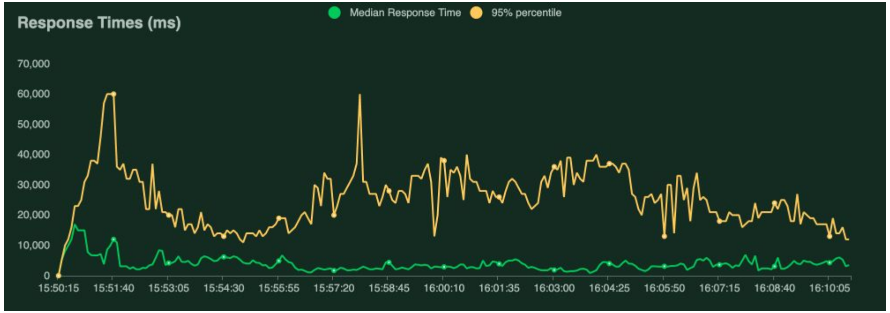

## Locust Website

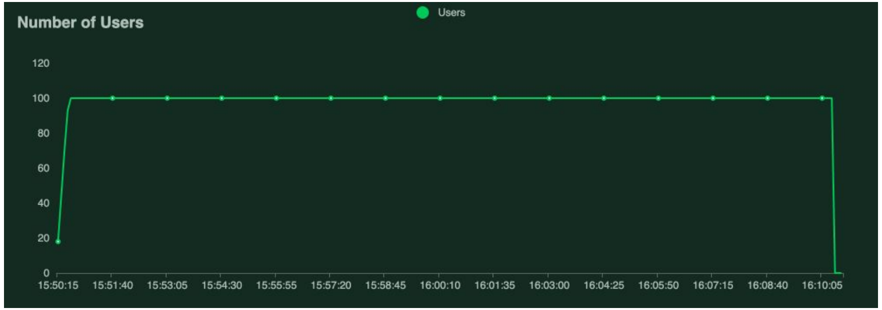
# Test



- We have decided to apply **horizontal scaling**, adding more ”machines” to our resource pool.
    - The goal is not to verify the correctness of the code or data, and furthermore, the responsiveness of the different page elements was not tested in the load/performance test.
- The path we tested for our application is `/predict_test` (managed back-end with Flask).
- As a case study we have analyzed a load of **100 users** and these are entered into the system with a spawn rate of **5 users/second**.
- The parameter we analyzed in the tests was the **CPU utilization**.
    - We did a series of tests to be able to understand what was the optimal number of pods to **obtain a lower response time** and also to **not have a large number of failures** during requests.



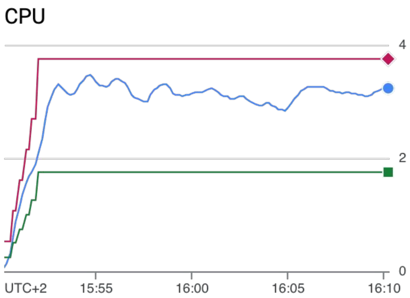

## Request Statistics

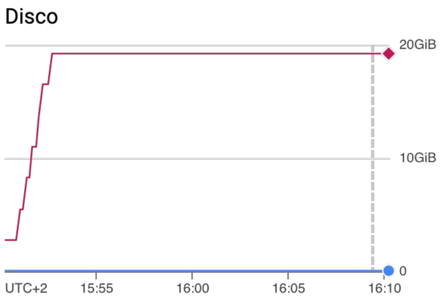
| Method | Name | # Requests | # Fails | Average (ms) | Min (ms) | Max (ms) | Average size (bytes) | RPS | Failures/s |
|---|---|---|---|---|---|---|---|---|---|
| POST | /predict_test | 12611 | 49 | 6473 | 1 | 60004 | 106 | 10.4 | 0.0 |
| | **Aggregated** | **12611** | **49** | **6473** | **1** | **60004** | **106** | **10.4** | **0.0** |

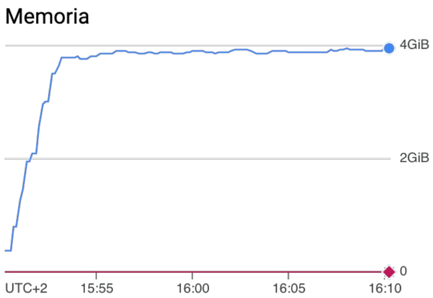
## Response Time Statistics

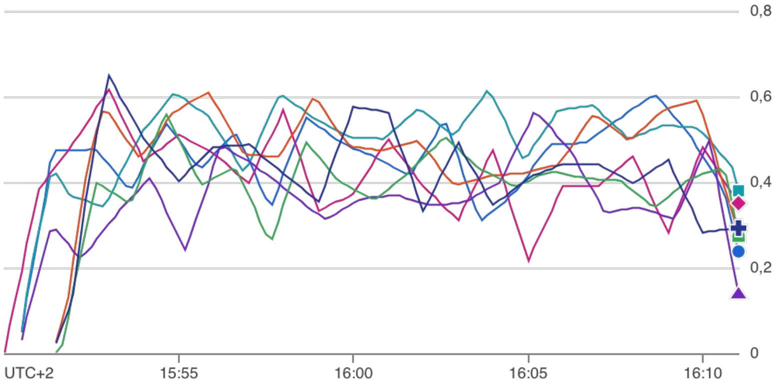
| Method | Name | 50%ile (ms) | 60%ile (ms) | 70%ile (ms) | 80%ile (ms) | 90%ile (ms) | 95%ile (ms) | 99%ile (ms) | 100%ile (ms) |
|---|---|---|---|---|---|---|---|---|---|
| POST | /predict_test | 3300 | 4900 | 6900 | 9700 | 15000 | 25000 | 44000 | 60000 |
| | **Aggregated** | **3300** | **4900** | **6900** | **9700** | **15000** | **25000** | **44000** | **60000** |

# Test with 7 pods: Locust




# Test with 7 pods: Google Resources

# Estimated Bill

# Costs of the project

- To analyze the costs of our project we used the **Google Cloud Price Calculator** service.
    - The cost of our project with the listed services is **$225.31** per month for a total of **$2703.72** for one year.
- Some aspects raises the price of our project:
    - the region chosen
    - the number of  clusters used
    - the boot disk capacity of each node

# Your Estimated Bill *

## Estimated Monthly Cost: USD 255.31

| | | | |
|---|---|---|---|
| CloudBuild | Cloud Build | 1 | USD 0.00 |
| | Inbound data processed by load balancer | 3.01361083984375e-06 GiB | USD 0.00 |
| | Outbound data processed by load balancer | 8.058547973632812e-06 GiB | USD 0.00 |
| 4 x FootballBet cluster | e2-medium | 2920 total hours per month | USD 61.64 |
| 4 x boot disk | Persistent Disk - GKE Standard | 100 GiB | USD 40.00 |
| 3 x LoadTesting cluster | e2-medium | 2190 total hours per month | USD 46.23 |
| 3 x boot disk | Persistent Disk - GKE Standard | 100 GiB | USD 30.00 |
| GKE Clusters Fee | Zonal Clusters | 1460 hours | USD 71.60 |
| | Inbound data processed by load balancer | 3.01361083984375e-06 GiB | USD 0.00 |
| | Outbound data processed by load balancer | 8.058547973632812e-06 GiB | USD 0.00 |
| Used on standard VM instances IP addresses | Assigned and used in standard VM IPs | 1460 | USD 5.84 |
| **Total Estimated Monthly Cost** | | | **USD 255.31** |

# Estimated Bill