# 15-319 / 15-619 Cloud Computing

Recitation 13

November 17th 2020

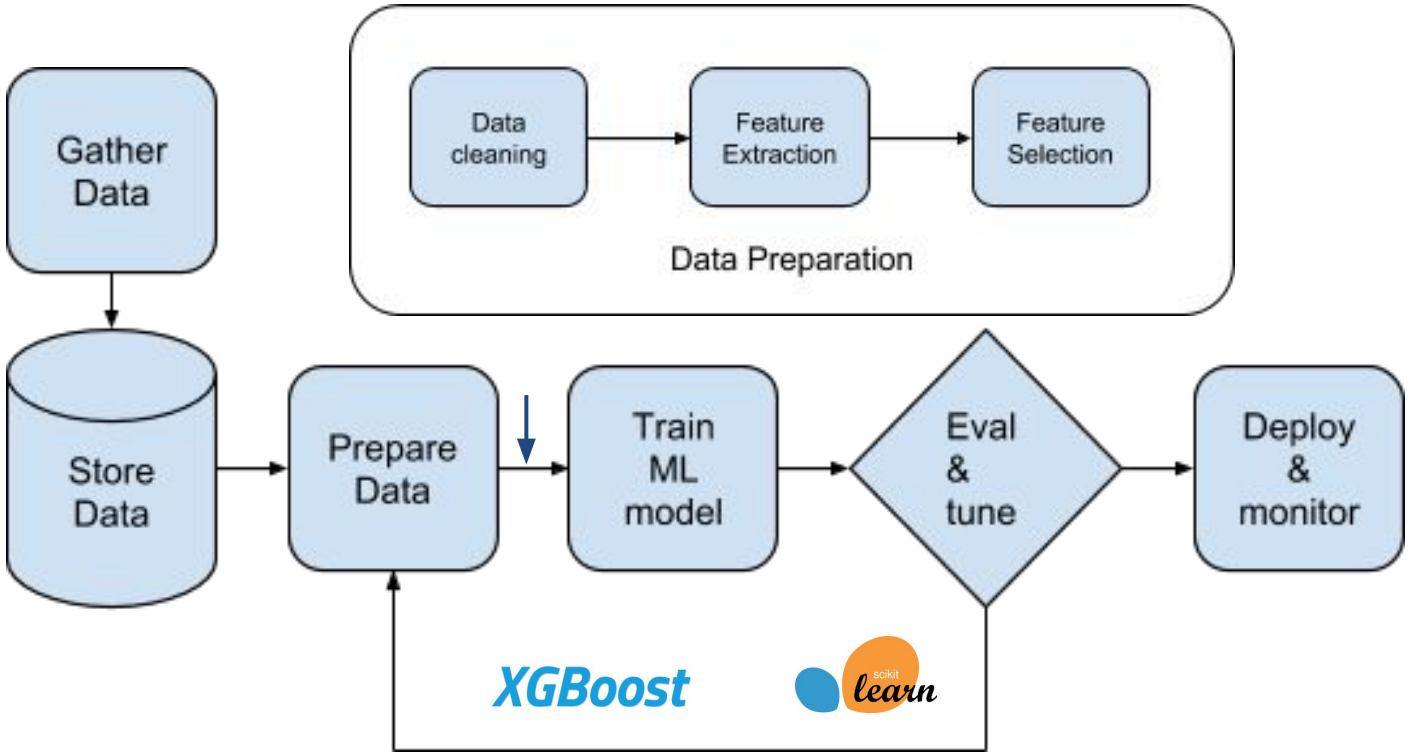
# Overview

- **Last week's reflection**
  - Team Project Phase 2, Live Test
  - Quiz 10

- **This week's schedule**
  - Unit 5 - Modules 21, 22
    - Quiz 11
  - Project 4.2
    - Machine Learning on the cloud
  - **Twitter Analytics: The Team Project**
    - Phase 3
      - Managed Services

# Machine Learning in Production

- A proliferation of tools on the Cloud



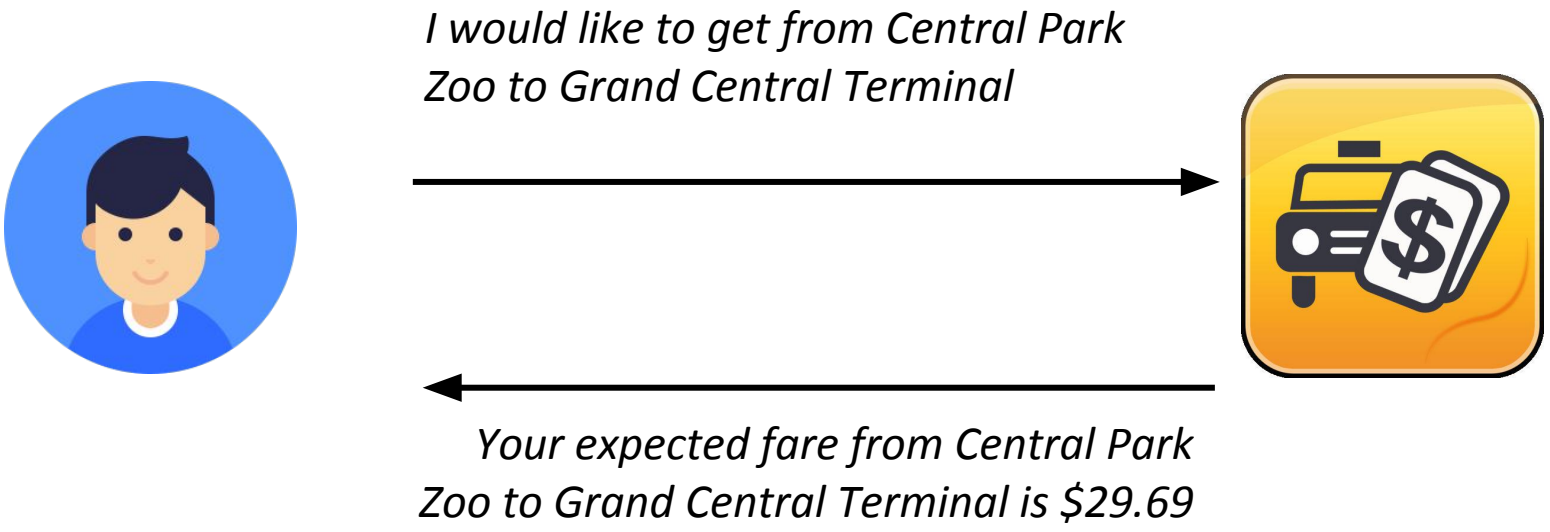


# ML on Managed Services

- Machine learning training on large datasets tend to be computationally intensive
- An increasingly affordable option for users without specialized IT infrastructure is to process ML workloads on the cloud with Managed Services like the Google AI Platform.
- Benefits:
  - No need to provision and configure virtual machines
  - Horizontal and vertical scaling is possible
  - No need to write custom logic to orchestrate multiple workers and achieve parallel training
  - Deploy your model to the cloud

# P4.2 - Taxi Fare Prediction Application

- Accepts speech queries to get the fare estimate to get from point to point (based on historical data), and returns the result as speech



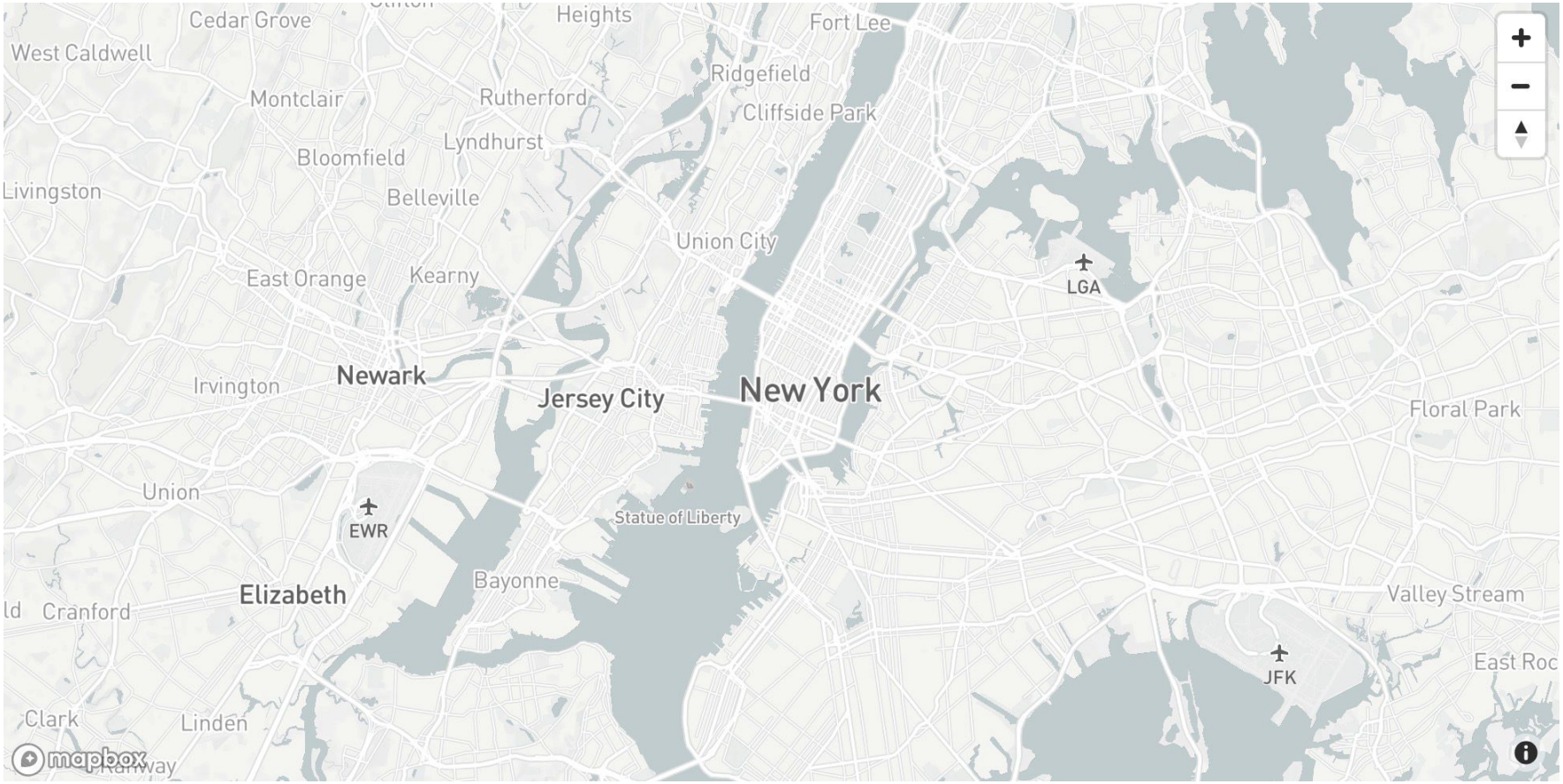
# P4.2 - Overview of Tasks

- Task 1: Data Visualization and Feature Engineering
- Task 2: Training, hyperparam tuning, deploying your model using the Google AI Platform and serving queries.
- Task 3: Stitch together services into a pipeline to build a user-facing interface for fare predictions.
- Bonus:
  - Use Cloud Vision API to identify NYC landmarks
  - Use AutoML to train a custom model that accepts custom landmarks as input for prediction

# Task 1: Feature Engineering - Data Visualization

- You are given a small training dataset containing historical data of fare prices in New York City.
- Steps to perform
  - Data exploration and visualization
  - Understand the data for Feature Engineering with regards to feature construction, data cleaning, etc.

# Task 1: Feature Engineering - Data Visualization



# Task 1: Feature Engineering

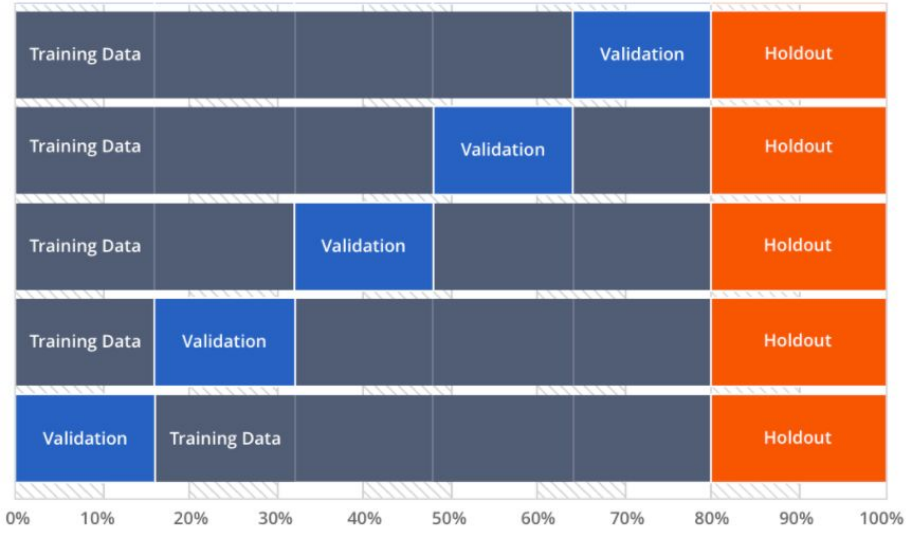
- You are given a small training dataset containing historical data of fare prices in New York City
- Steps to perform
  - Clean the data and remove outliers
    - Consider what you learned from the data visualization task
  - Extract or construct meaningful features that will improve performance over the baseline model (which uses raw features with no transformations)

# Task 1: Feature Engineering

- Feature engineering = transforming domain knowledge into better features
- Some ideas for feature engineering
  - Calculate distance from the geo-coordinates
  - Calculate distance to landmarks
  - What are good proxies for traffic conditions?

# Task 1: Feature Engineering

- Evaluating your model
  - Metric: Root Mean Squared Error (RMSE)
  - K-fold Cross-Validation
    - Used to assess the predictive performance of the model outside the training sample on unseen data




  - Plot feature importance

# Task 2: Training, Tuning & Deploying

- Train and tune the model on complete dataset on Google AI Platform.
- Deploy the trained model to AI Platform.
- Develop a Flask application that accepts web requests and returns fare predictions.
  - Transform raw features from web requests using the feature engineering solution developed in Task 1.
  - Make API calls to the model hosted on AI Platform
  - Format and return a web response
- Deploy the Flask application on Google App Engine.

# Task 2: Tuning with Google AI Platform

- **Hyperparameter Tuning**
- Parameters v/s. Hyperparameters
  - Parameters: internal, often not set by the practitioners
  - Hyperparameters: external, often set by the practitioners before training
    - Basically, configuration parameters that impact the training process
- Finding optimal hyperparameters with an exhaustive Grid Search is expensive

# Task 2: Tuning with GCP HyperTune

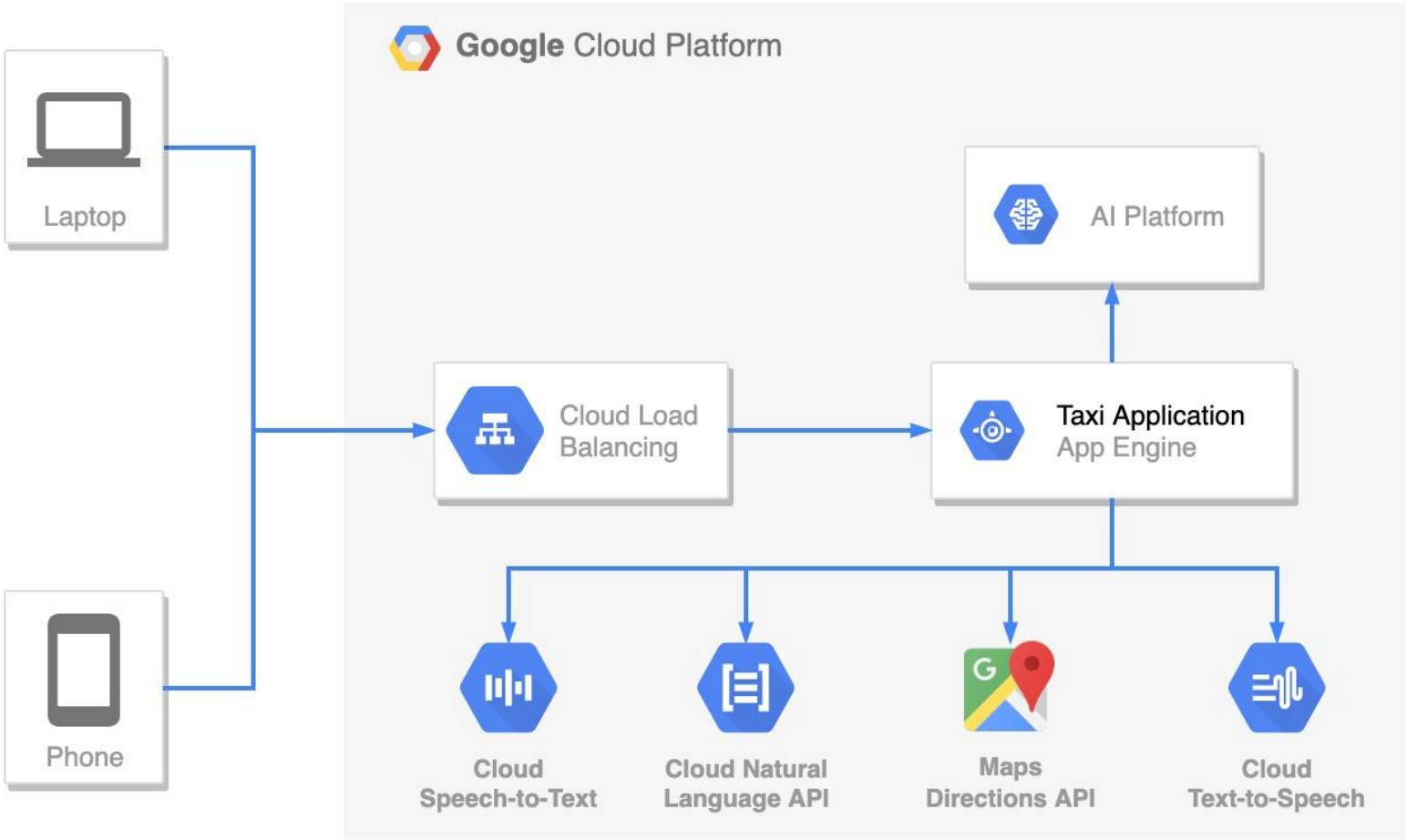
- Black box optimization service (does not need access to the underlying model)
- Need to specify a config yaml file that describes which hyperparameters to tune
- Uses a method called Bayesian Optimization to efficiently search through different combinations of hyperparameters
- An example of a HyperTune configuration file: hptuning_config.yaml

# Task 2: Deploying Model to AI Platform

- **To get a full score in this task, you need to:**
  - Enable HyperTune, add at least 3 additional parameters to tune, run the hypertuning job and create a model on Google AI Platform.
  - Deploy the fare prediction application to GAE that uses the model you created and serves web requests correctly.
  - The predictions should achieve a target accuracy, measured by RMSE.

# Task 3: ML Application Pipeline

- Build an end-to-end application pipeline to predict car fare requests using the following architecture.



# Task 3: ML Application Pipeline

- Your application will include multiple APIs
  - Functional APIs to be implemented
    - **/predict** - Generate fare predictions for a JSON array of rides
    - **/speechToText** - Convert WAV audio to text string
    - **/textToSpeech** - Convert text string to WAV audio
    - **/namedEntities** - Identify landmarks in a given sentence
    - **/directions** - For two given NYC landmarks, determine the latitude / longitude for each pickup and drop off pair

# Task 3: ML Application Pipeline

Putting it together:

- **/farePrediction** - Given a WAV audio ride request, determine the predicted fare
  - **Response**
    - { "**predicted_fare**": "23.78",
      "**entities**": ["Charging Bull", "Carnegie Hall"],
      "**text**": "Your expected fare from Charging Bull to Carnegie Hall is $23.78",
      "**speech**": <BASE64 ENCODED AUDIO> }

- General solution flow
  - Speech to text ride request (/speechToText)
  - Extract entities from text ride request (/namedEntities)
  - Get the coordinates of the pickup and drop off locations (/directions)
  - Query the AI Platform model to get the predicted fare (/predict)
  - Convert the text response to speech (/textToSpeech)

# Bonus: Landmark Recognition

- (5 points) Use Cloud Vision to identify NYC landmarks
- (5 points) Add unique destinations using AutoML

- **/farePredictionVision**
  - Unlike **/farePrediction**, the ride request will not be sent as WAV audio
  - The API will accept the source and destination as images of NYC landmarks
  - Must query the Cloud Vision API and custom AutoML model to determine the landmark names
  - Continue with the same request as **/farePrediction**

# Bonus: Landmark Recognition



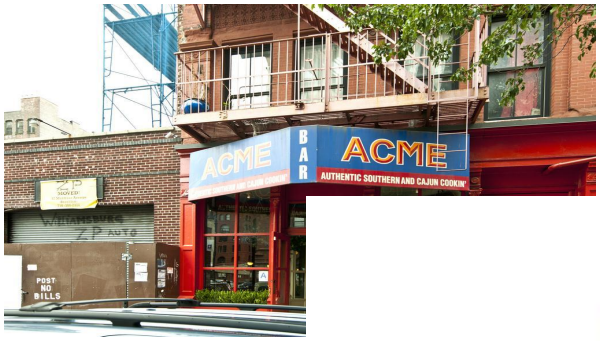







. . .  . . .  . . .

# Bonus: Landmark Recognition







Cloud AutoML Vision





. . .  . . .  . . .

# Hints

- Task 1: Feature transformation
    - The exact same feature transformations must be applied to the training and the test set
    - Cannot share code if stateful functions are used, for example:
        - get_dummies()
        - df.qcut()
    - Store state like bin ranges and categorical values to apply the transformation consistently
    - Jupyter: command not found (use virtualenv)

# Hints

- Task 2: HyperTune
  - Read the XGBoost hyperparameter doc to understand which hyperparameters can help most.
  - You can change the number of workers for the AI Platform training job to parallelize the training process.
  - Learn to make good estimates for the cost for each run
    - Cost = Consumed ML Units * $0.49

# Issues to Consider

- Overfitting
  - RMSE on training data is much lower than test data
  - You should not filter outliers just because it makes your cross validation scores look better, since some of these records may be representative of the patterns in the real world.
    - Students who do this may pass Task 1, but will fail Task 2.
    - You should make sure you have good features first, before trying to play around with filtering outliers.

# TEAM PROJECT
# Twitter Data Analytics

# Team Project Phase 2 Live Test
# Top Q1 Teams

| | |
|---|---|
| Pikachu | 47656.00 |
| WhiteGiving | 43577.52 |
| RedChillies | 42597.50 |
| randomGroup | 40378.63 |
| WuhanJiayou | 39302.70 |

Q1H

| | |
|---|---|
| WhiteGiving | 62070.91 |
| FollowMe | 54065.30 |
| RedChillies | 48608.60 |
| Pikachu | 47317.40 |
| 14 Principles | 44390.81 |

Q1M

Congrats to **WhiteGiving, Pikachu, RedChillies** for top performance for both HBase and MySQL tests.

# Team Project Phase 2 Live Test
# Top Q2 Teams

| | |
|---|---|
| WhiteGiving | 22791.10 |
| FollowMe | 11775.96 |
| Frantic Horizon | 10988.26 |
| Best_In_Adelaide | 10367.73 |
| infinity | 10253.90 |

Q2H

| | |
|---|---|
| FollowMe | 28206.20 |
| WhiteGiving | 27394.40 |
| Frantic Horizon | 24737.30 |
| SV No.1 | 23095.02 |
| RedChillies | 19509.22 |

Q2M

Congrats to **WhiteGiving, FollowMe, Frantic Horizon** for top performance for both HBase and MySQL tests.

# Team Project Phase 2 Live Test
# Top Q3 Teams

| | |
|---|---|
| WhiteGiving | 5140.80 |
| infinity | 3151.50 |
| FollowMe | 2030.20 |
| Frantic Horizon | 1775.30 |
| PepeFTW | 1667.61 |

Q3H

| | |
|---|---|
| WhiteGiving | 16616.00 |
| SV No.1 | 9683.10 |
| infinity | 8484.20 |
| PepeFTW | 7675.00 |
| Frantic Horizon | 5986.30 |

Q3M

Congrats to **WhiteGiving, infinity, Frantic Horizon, PepeFTW** for top performance for both HBase and MySQL tests.

# Team Project - Phase 3

- Use only **AWS managed services** for all queries
- Development budget: $120
  - Penalty for lavishness: >$150
- Live test:
  - Per-hour-budget: $1.28 (included in $100)
- Perform ETL on your beloved GCP and Azure

# Cloud Managed Services

- Managed services remove the burden from having to operate the provisioned cloud resources.

- Management tools such as monitoring, patching, security, backup are offered as part of the service.

# Team Project - Phase 3

- RPS targets have **not** been changed →
  - Q1: 32000
  - Q2: 10000
  - Q3: 1500
- For web-tier, you could use either ECS or EKS.
- For storage-tier, you could use any managed services we gives in the writeup.
- Rule of thumb:
  - If you want to start any resources on EC2 dashboard, **stop**
  - If you are doing `sudo apt install mysql-server`, **stop**
- Teams should explore the managed services provided by AWS to come up with a solution
- Teams are required to use Terraform (unless Terraform does not have support for your particular managed service)

# Team Project General Hints

- No EC2 instances and EBS volumes in the live test!
  - Nonetheless, you can use those to do verification or comparison to the hosted service you built before in the development process.
  - Note that ECS and EKS are backed up by EC2 instances, so it’s perfectly normal to see EC2 instances in dashboard after you fire up your containers.
- Don’t use T-family instances for ECS or EKS, as their performance are not consistent.

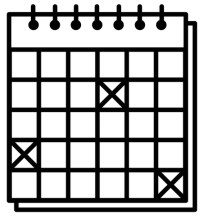
# Team Project General Hints

- One option would be to split the services into web-tier and storage-tier and choose different managed services.
  - If so, the compatibility of these two services should be taken into account.

- Consider the different characteristics of queries to decide what kind of managed services to use.

- High performance/cost ratio is valued.
  - Try your best to achieve the highest possible ratio.

# Team Project Time Table



| Phase (and query due) | Start | Deadlines | Code and Report Due |
|---|---|---|---|
| Phase 1<br>• Q1, Q2 | Monday 10/12/2020 00:00:00 ET | **Q1 Checkpoint: Sunday 10/18/2020 23:59:59 ET**<br>**Q1: Sunday 10/25/2020 23:59:59 ET**<br>**Q2 Checkpoint: Monday 10/25/2020 23:59:59 ET**<br>**Q2: Sunday 11/01/2020 23:59:59 ET** | Tuesday 11/03/2020 23:59:59 ET |
| Phase 2<br>• Q1, Q2,Q3 | Monday 11/02/2020 00:00:00 ET | **Sunday 11/15/2020 18:00:00 ET** | |
| Phase 2 Live Test (Hbase **AND** MySQL)<br>• Q1, Q2, Q3 | **Sunday 11/15/2020 18:00:00 ET** | **Sunday 11/15/2020 23:59:59 ET** | **Tuesday 11/17/2020 23:59:59 ET** |
| Phase 3<br>• Q1, Q2, Q3 (Managed services) | Monday 11/16/2020 00:00:00 ET | Sunday 12/06/2020 15:59:59 ET | |
| Phase 3 Live Test<br>• Q1, Q2, Q3 (Managed services) | Sunday 12/06/2020 18:00:00 ET | Sunday 12/06/2020 23:59:59 ET | Tuesday 12/08/2020 23:59:59 ET |

# Upcoming Deadlines

- Team Project: Phase 2
  - Code and report due:
    - Tuesday, Nov 17, 2020 11:59 PM ET
- Project 4.2: Machine Learning on the Cloud
  - Due Sunday, Nov 22, 2020, 11:59 PM ET
- Team Project: Phase 3
  - Live-test:
    - Sunday, Dec 06, 2020 6:00 PM ET
  - Code and report due:
    - Tuesday, Dec 08, 2020 11:59 PM ET