gTech Ads Optimus

Deployment on Google Cloud Platform

krasowiak@google.com

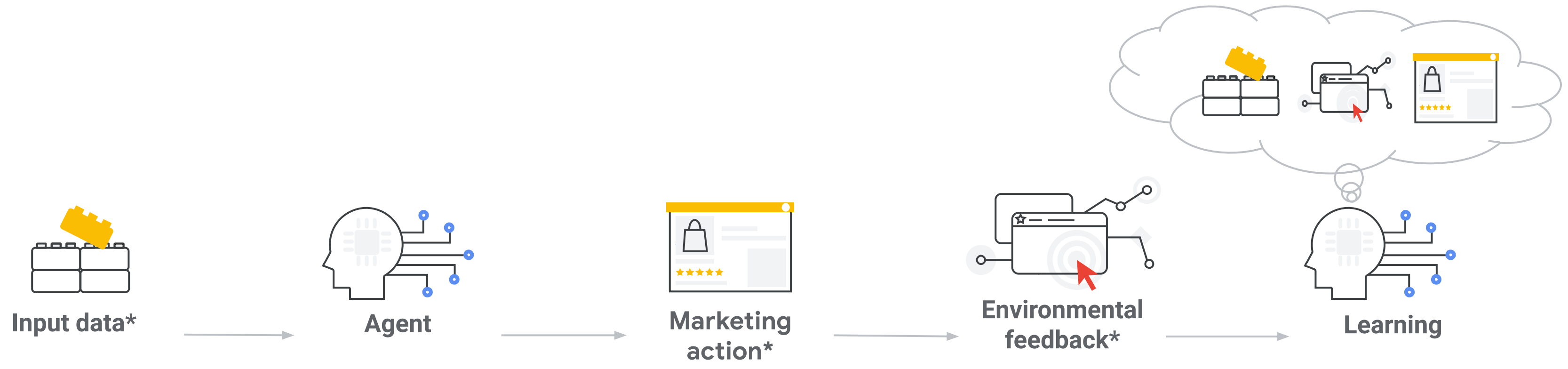
Last updated: 05.02.2024

Agenda

Introduction to Optimus

Optimus helps advertisers optimize their marketing KPIs by dynamically personalizing their front-end applications using Artificial Intelligence and Reinforcement Learning in a privacy-first way.

Optimus learns by finding relationships between input data, predicted marketing action and environmental feedback



* Input data, content building blocks and user interaction types are customized for each advertiser and their optimization goal.

Optimus brings many business-relevant benefits

Can learn patterns in noisy data

These algorithms employ a trial-and-error approach to learn from data, exploring different actions and observing the outcomes to identify patterns and make optimal decisions even in novel situations.

Can adapt to new situations quickly

The algorithms are constantly learning from their experience, and they can use this new information to adjust their behavior.

No need for labeled data

Reinforcement learning eliminates the need for expensive labeled data by enabling agents to learn from the feedback inherent in the consequences of their actions.

*It can autonomously identify and learn patterns in data, evaluate and test various marketing actions, and continuously adapt its approach to evolving market conditions. The autonomous nature of Reinforcement Learning and the high level of automation can significantly reduce the resources and time typically required for optimizing marketing decisions.

The Optimus applications in marketing are vast and diverse

Conversion Rate Optimization (CRO)

Dynamic pricing

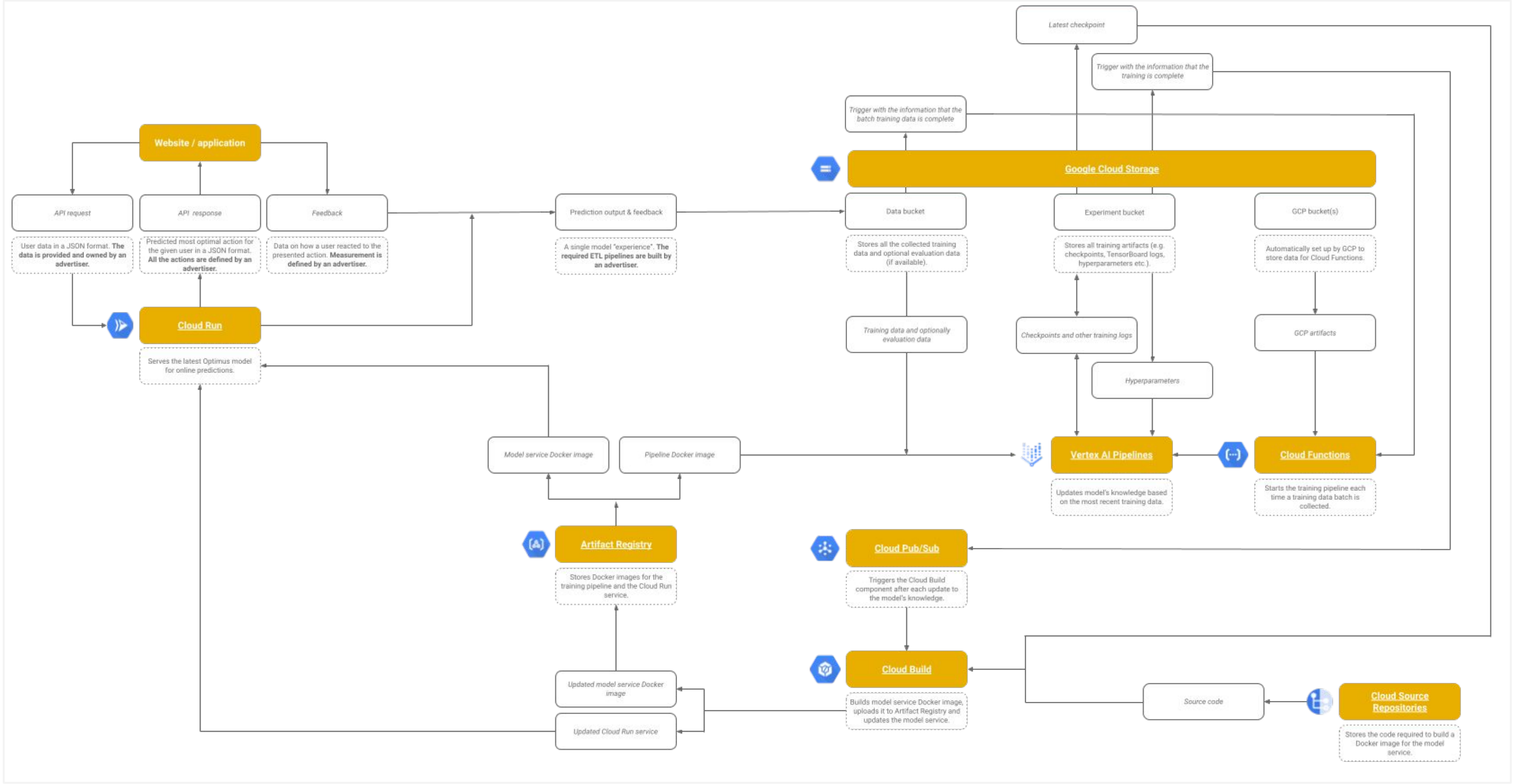
Marketing automation

Support for ad bidding

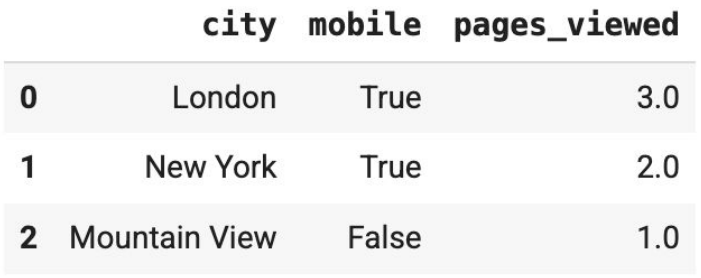
These are just use case examples and many more can be identified.

Deploying Optimus on GCP

Optimus deployment architecture on GCP


Website / application
API request
User data in a JSON format. The data is provided and owned by an advertiser.
API response
Predicted most optimal action for the given user in a JSON format. All the actions are defined by an advertiser.
Feedback
Data on how a user reacted to the presented action. Measurement is defined by an advertiser.
Prediction output & feedback
A single model "experience". The required ETL pipelines are built by an advertiser.
Cloud Run
Serves the latest Optimus model for online predictions.
Latest checkpoint
Trigger with the information that the training is complete
Trigger with the information that the batch training data is complete
Google Cloud Storage
Data bucket
Stores all the collected training data and optional evaluation data (if available).
Experiment bucket
Stores all training artifacts (e.g. checkpoints, TensorBoard logs, hyperparameters etc.).
GCP bucket(s)
Automatically set up by GCP to store data for Cloud Functions.
Training data and optionally evaluation data
Checkpoints and other training logs
Hyperparameters
GCP artifacts
Model service Docker image
Pipeline Docker image
Vertex AI Pipelines
Updates model's knowledge based on the most recent training data.
Cloud Functions
Starts the training pipeline each time a training data batch is collected.
Artifact Registry
Stores Docker images for the training pipeline and the Cloud Run service.
Cloud Pub/Sub
Triggers the Cloud Build component after each update to the model's knowledge.
Cloud Build
Builds model service Docker image, uploads it to Artifact Registry and updates the model service.
Updated model service Docker image
Updated Cloud Run service
Source code
Cloud Source Repositories
Stores the code required to build a Docker image for the model service.

Currently Optimus supports only a tabular data input that can be passed as a JSON file via an API request.



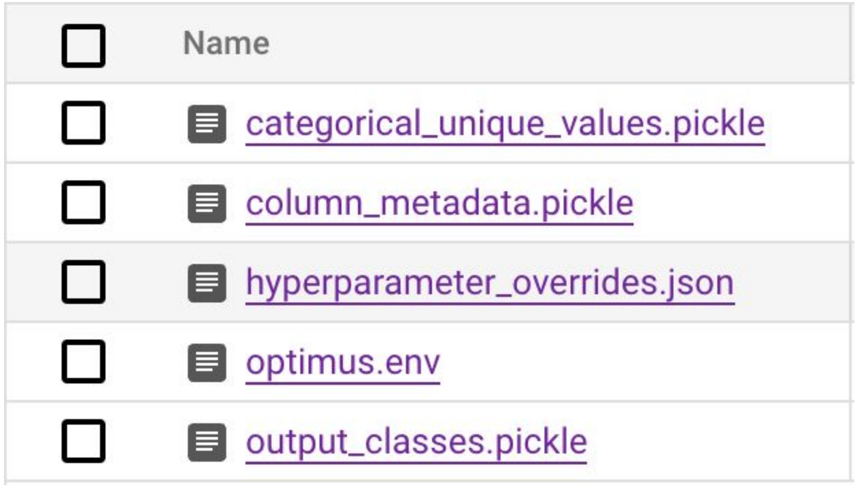
	city	mobile	pages_viewed
0	London	True	3.0
1	New York	True	2.0
2	Mountain View	False	1.0

```
{'instances': [{'city': 'London', 'mobile': True, 'pages_viewed': 3.0},
  {'city': 'New York', 'mobile': True, 'pages_viewed': 2.0},
  {'city': 'Mountain View', 'mobile': False, 'pages_viewed': 1.0}]}
```

Sample input data for Optimus to determine the best marketing action.

Sample input data format for an API request.

There are 5 files you need to have ready to deploy Optimus



☐	Name
☐	categorical_unique_values.pickle
☐	column_metadata.pickle
☐	hyperparameter_overrides.json
☐	optimus.env
☐	output_classes.pickle

The categorical unique values pickle* file contains a mapping between categorical column names to respective lists of unique column values

```
{'city': ['London', 'New York', 'Mountain View'], 'mobile': [True, False]}
```

Example of the categorical unique values mapping. You will need to know this information beforehand from your Customer if there are any categorical values in the input data.

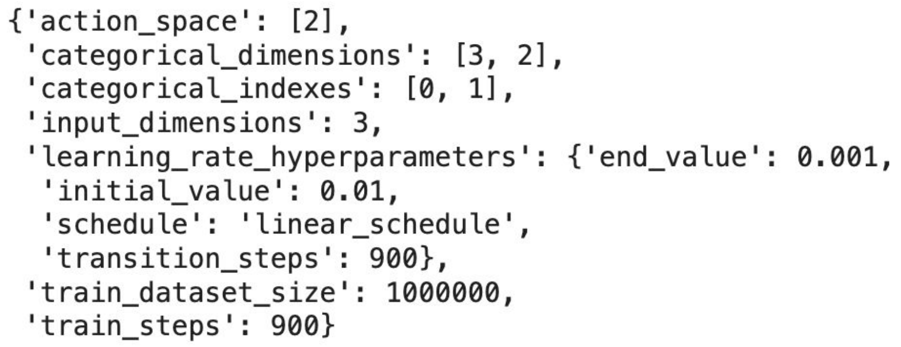
*We use the PICKLE format, because JSON automatically converts all keys into strings. And some column names might not be strings.

The column metadata must list all the columns, any categorical columns and any columns to skip

```
{'all_columns': ['city', 'mobile', 'pages_viewed', 'session_timespan'],
 'categorical_columns': ['city', 'mobile'],
 'skipped_columns': ['session_timespan']}
```

Example of the column metadata mapping. The list of all columns in order as they appear in the input data must always be present. Provide empty lists if there are no categorical columns or no columns to skip. The mapping keys must be as indicated.

Hyperparameter overrides is a JSON file to specify any required hyperparameters and, optionally, modify the default ones.



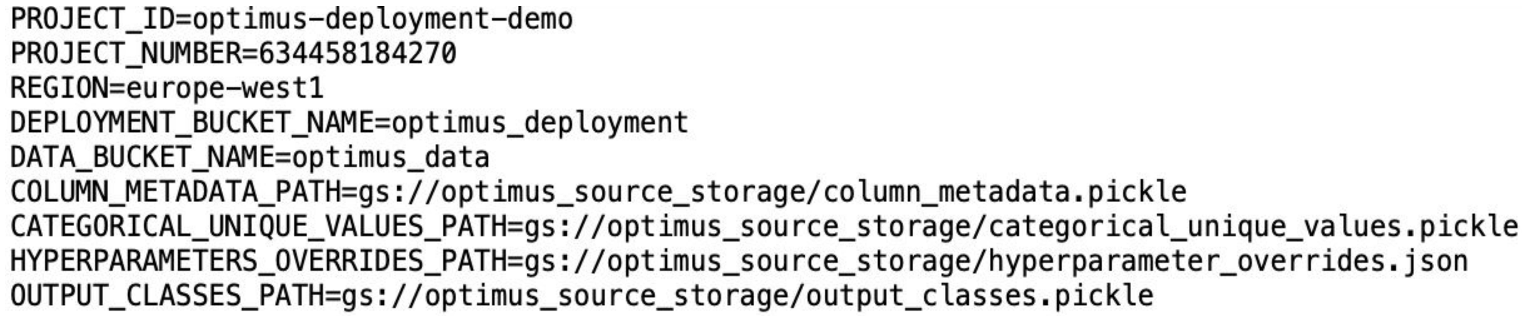
```
{'action_space': [2],
 'categorical_dimensions': [3, 2],
 'categorical_indexes': [0, 1],
 'input_dimensions': 3,
 'learning_rate_hyperparameters': {'end_value': 0.001,
  'initial_value': 0.01,
  'schedule': 'linear_schedule',
  'transition_steps': 900},
 'train_dataset_size': 1000000,
 'train_steps': 900}
```

\`action_space\`, \` categorical_dimensions\`, \`categorical_indexes\`, \`input_dimensions\`, \`learning_rate_hyperparameters\`, \`train_dataset_size\` and \`train_steps\` are the hyperparameters that need to be specified by a user.

Required hyperparameter	Description
action_space	The number of actions Optimus has to choose from.
categorical_indexes	A sequence with indexes of categorical columns in the input data.*
categorical_dimensions	A sequence with categorical column dimensions in the input data.*
input_dimensions	The number of features (i.e. columns) in the input data.
learning_rate_hyperparameters	A mapping between learning rate hyperparameters and their values. \`transition_steps\` is needs to be amended. It's recommended to be the same as \`train_steps\`.**
train_dataset_size	The number of observations that are estimated to be passed through Optimus.***
train_steps	The number of Optimus retraining sessions.***

*It can be easily determined with the base_preprocessing module from the Optimus package.
** All the other \`learning_rate_hyperparameters\` must be included in the mapping, not just the amended \`transition_steps\`
***The total number of traffic, the time for deployment and other factors must be taken under consideration to determine it.

The .env file specifies 8 environmental variables for Optimus deployment on GCP



```
PROJECT_ID=optimus-deployment-demo
PROJECT_NUMBER=634458184270
REGION=europe-west1
DEPLOYMENT_BUCKET_NAME=optimus_deployment
DATA_BUCKET_NAME=optimus_data
COLUMN_METADATA_PATH=gs://optimus_source_storage/column_metadata.pickle
CATEGORICAL_UNIQUE_VALUES_PATH=gs://optimus_source_storage/categorical_unique_values.pickle
HYPERPARAMETERS_OVERRIDES_PATH=gs://optimus_source_storage/hyperparameter_overrides.json
OUTPUT_CLASSES_PATH=gs://optimus_source_storage/output_classes.pickle
```

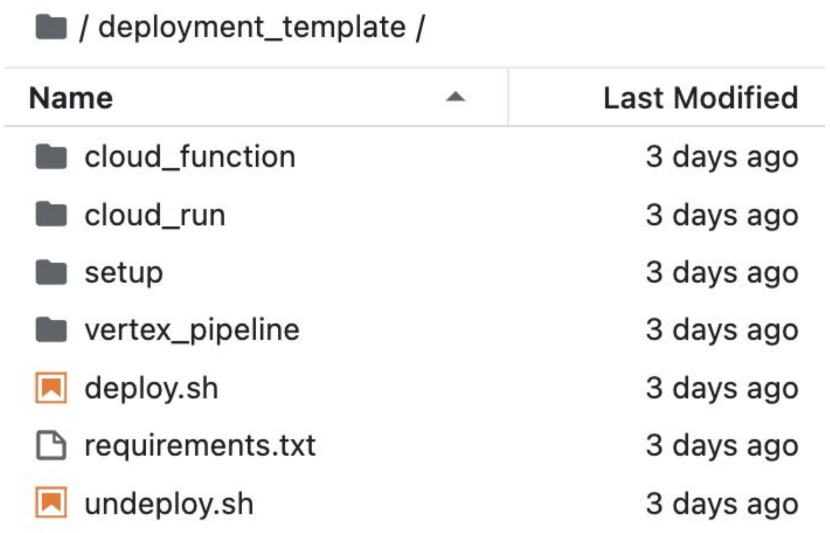
All these variables are required and the keys must be as indicated.

The output classes contains a sequence with all possible actions, i.e. choices for the model

```
['No', 'Yes']
```

Example of the output classes sequence. The sequence is required only when the action is a categorical choice rather than a numerical prediction. The number of actions can span across hundreds of choices.

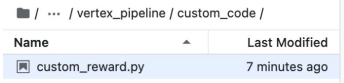
Sync to the Optimus GitHub repository and make the deployment repository your working directory



/ deployment_template /

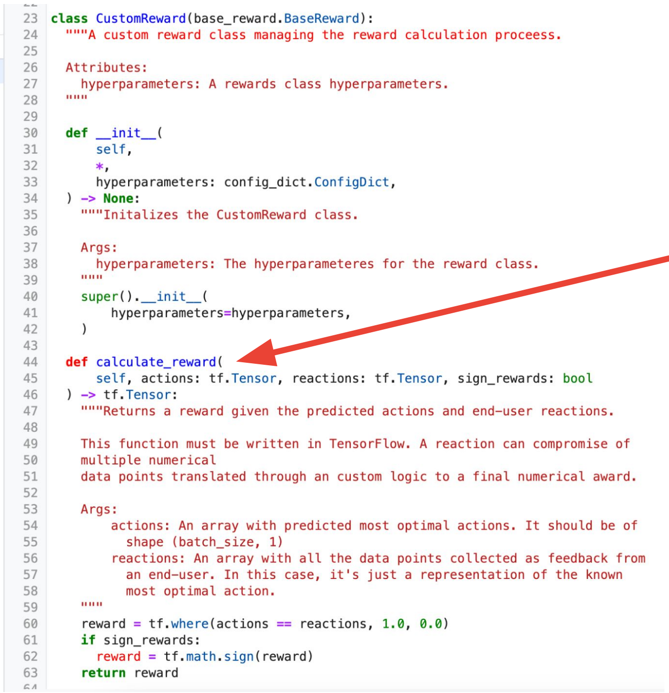
Name	Last Modified
cloud_function	3 days ago
cloud_run	3 days ago
setup	3 days ago
vertex_pipeline	3 days ago
deploy.sh	3 days ago
requirements.txt	3 days ago
undeploy.sh	3 days ago

Each advertiser will be collecting different user feedback and optimize against their own KPI, it requires to customize the reward function



/ ··· / vertex_pipeline / custom_code /

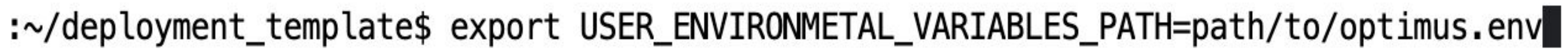
Name	Last Modified
custom_reward.py	7 minutes ago



```python
class CustomReward(base_reward.BaseReward):
  """A custom reward class managing the reward calculation proceess.

  Attributes:
    hyperparameters: A rewards class hyperparameters.
  """

  def __init__(
      self,
      *,
      hyperparameters: config_dict.ConfigDict,
  ) -> None:
    """Initalizes the CustomReward class.

    Args:
      hyperparameters: The hyperparameteres for the reward class.
    """
    super().__init__(
        hyperparameters=hyperparameters,
    )

  def calculate_reward(
      self, actions: tf.Tensor, reactions: tf.Tensor, sign_rewards: bool
  ) -> tf.Tensor:
    """Returns a reward given the predicted actions and end-user reactions.

    This function must be written in TensorFlow. A reaction can compromise of
    multiple numerical
    data points translated through an custom logic to a final numerical award.

    Args:
        actions: An array with predicted most optimal actions. It should be of
          shape (batch_size, 1)
        reactions: An array with all the data points collected as feedback from
          an end-user. In this case, it's just a representation of the known
          most optimal action.
    """
    reward = tf.where(actions == reactions, 1.0, 0.0)
    if sign_rewards:
      reward = tf.math.sign(reward)
    return reward
```

You can then, optionally, modify any other file in the deployment template if your use case requires more customization

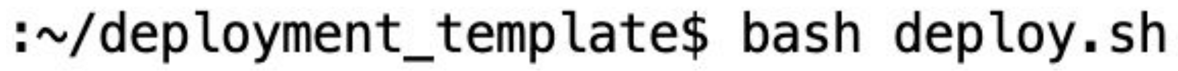
USER_ENVIRONMETAL_VARIABLES_PATH variable needs to be set before deploying Optimus on GCP



```
:~/deployment_template$ export USER_ENVIRONMETAL_VARIABLES_PATH=path/to/optimus.env
```

The variable is the path to the optimus.env file created earlier.

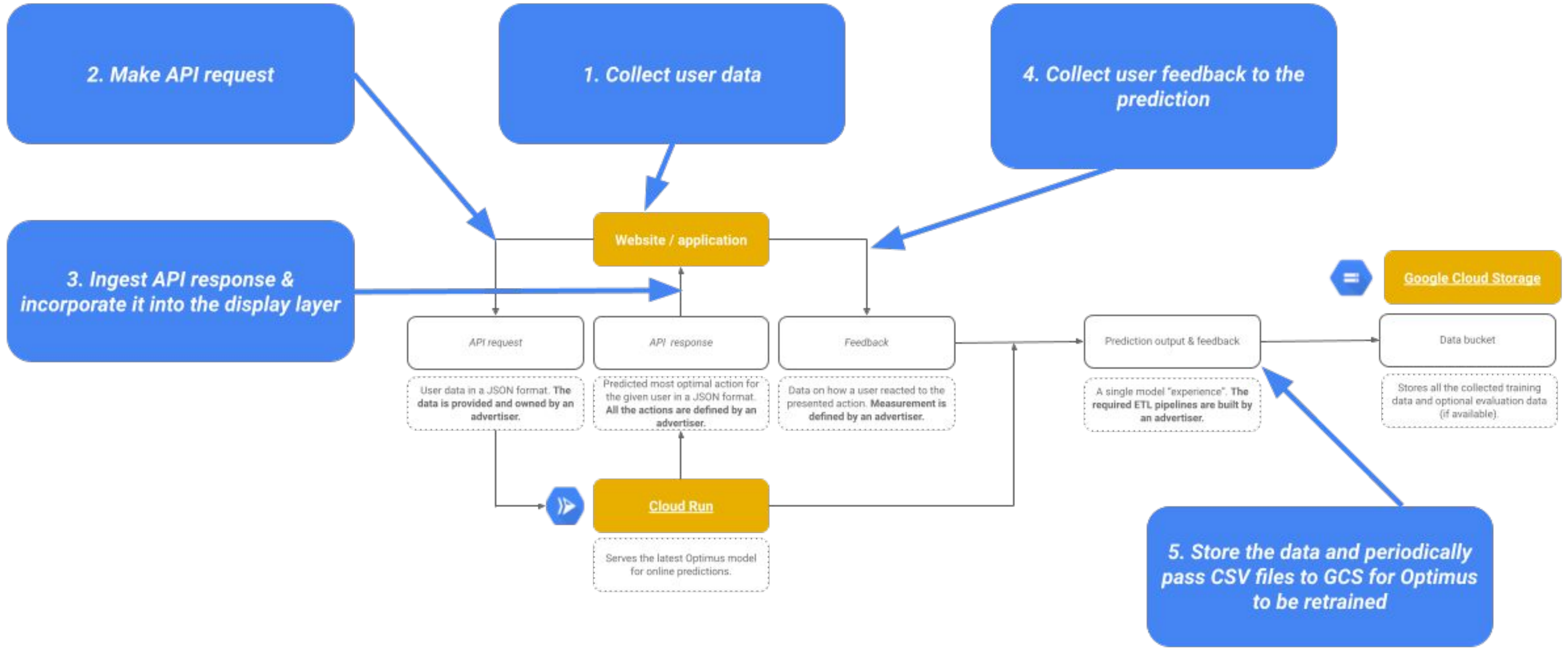
Run the deploy.sh file to deploy Optimus



```
:~/deployment_template$ bash deploy.sh
```

Advertiser responsibilities

An advertiser would typically own 5 tasks


2. Make API request
1. Collect user data
4. Collect user feedback to the prediction
Website / application
3. Ingest API response & incorporate it into the display layer
Google Cloud Storage
API request
API response
Feedback
Prediction output & feedback
Data bucket
User data in a JSON format. The data is provided and owned by an advertiser.
Predicted most optimal action for the given user in a JSON format. All the actions are defined by an advertiser.
Data on how a user reacted to the presented action. Measurement is defined by an advertiser.
A single model "experience". The required ETL pipelines are built by an advertiser.
Stores all the collected training data and optional evaluation data (if available).
Cloud Run
Serves the latest Optimus model for online predictions.
5. Store the data and periodically pass CSV files to GCS for Optimus to be retrained

Files with most recent experiences need to be uploaded to gs://$DATA_BUCKET_NAME/training_experiences by the advertiser

	city	mobile	pages_viewed	action	value	log_probability	done	attentive_transformer_loss	feedback
0	London	True	3.0	discount	0.1	0.1	True	0.1	1
1	New York	True	2.0	discount	0.2	0.2	True	0.2	1
2	Mountain View	False	1.0	no_discount	0.3	0.3	True	0.3	0

All columns would be part of the prediction output, only feedback must be collected by an advertiser. The two datasets must be joined together by a unique id defined by the advertiser. An agreed number of such records must be periodically saved as a CSV file in gs://$DATA_BUCKET_NAME/training_experiences to trigger the retraining of the Optimus model.

Thank you!

gTech
‹professional services/›

Appendices

Marketing use case selection criteria for Reinforcement Learning

Access to rewards

Agents must receive feedback to learn optimal actions.
You always need to ensure that there is conceptual and/or technical ability for the agent's performance measurement.

Data velocity and volume

The more decisions the agent makes the quicker it learns.
Best use cases are where the agent would need make decisions frequently.

Exploration-exploitation trade-off

Exploration is essential for an agent to learn about its environment.

It means that at certain times, most noticeably at the beginning of its lifespan, the agent will make mistakes.