Google Ads

Reporting & Mutates Micro-Migration

Google Ads API Migration Workshops - 2021

In this session, we will be migrating a script that creates individual components for a new campaign and uses the reporting interface to retrieve data about the entities along the way. At the end, we'll also demonstrate how to retrieve metrics from the newly created campaign.

We will begin with a script that creates an entire campaign hierarchy using the AdWords API, which includes creating one or more:

- Campaign Budgets
- Campaigns
- Ad Groups
- Text Ads
- Keyword Criteria

Then, we will migrate the creation of each resource to use the Google Ads API. We will migrate the script incrementally and execute it along the way so that you can see both APIs working together until the migration is complete and only the Google Ads API is used. The main steps involved in this session are listed below.

This is meant to be an interactive session, and we encourage you to follow along with the demonstration by performing each of the steps below. Please post any questions you have to the Q&A forum, and our team will be standing by to help you out.

Note: There may be some implementation differences if you're working through this Codelab in a language other than Python. We recommend that you review the Campaign Management examples under the Migration directory in your client library's GitHub repository to identify the differences in how methods are organized and how entities are initialized and configured.

The steps and themes we present apply to all client library languages. If you have trouble following the live presentation, you can return to the coding exercises once the recording is available and you can pause the video to give yourself more time to complete each step.

Part 0 - Prerequisites

Before starting this Codelab you will need to complete the steps necessary to set up both an AdWords and Google Ads client library of your choice. If you haven't already, you will need to generate a developer token* and OAuth2 tokens and use them to configure both libraries.

The AdWords client library can be installed with your preferred language's package management system, for example Maven (Java), Nuget (.NET), CPAN (Perl), Composer (PHP), PyPi (Python), or Ruby Gems (Ruby). Similarly for the Google Ads client libraries: Maven (Java), Nuget (.NET), CPAN (Perl), Composer (PHP), PyPi (Python), or Ruby Gems (Ruby).

*Note that as of Apr 29, 2021 any newly created developer token will only have access to the Google Ads API.

Once you've successfully completed the installation steps described above, you're ready to proceed. To verify that your setup is correct, try executing the Get Campaigns example.

Part 1 - Getting Started

Next duplicate the Create Complete Campaign AdWords API Only example from the Migration directory in Google Ads client library you're using, for example:

1.0.0: Directory of the Python example

```
google-ads-python/examples/migration/campaign_management/
```

*Note that this location will be slightly different if you're not using the Python client library.

Lastly you will need the Client customer ID for a Test Google Ads account.

Warning: It's highly recommended that you use a test account in this Codelab because we'll be creating a number of entities that you may not want to use for trafficking ads.

Throughout this Codelab we'll be executing the example, which you can do using the below shell command:

1.0.1: Shell command to run the script

```
python create_complete_campaign_adwords_api_only.py
```

Part 1.1 - Documentation overview

The Google Ads documentation has dedicated migration guides. Of particular interest for updating this script will be:

- Map of AdWords services to Google Ads services
- Map of AdWords reports to Google Ads reports
- AWQL report query migration tool

Part 2 - Script overview

In this part we'll break down parts of the original Create Complete Campaign AdWords API Only script and highlight some key components before we start making changes. This script imports the AdWords API client library and uses it to initialize an AdWords client.

2.0.0: Importing the AdWords API client library

```
# This line imports the AdWords client library
from googleads import adwords
```

When calling `LoadFromStorage` the OAuth2 credentials are loaded from a configuration file that is stored on the local machine.

2.0.1: Initializing an AdWords API client

```
# This line initializes the AdWords client
adwords_client = adwords.AdWordsClient.LoadFromStorage()
```

This script creates a number of entities, one-at-a-time, and uses the ID of each created entity as an input to create the next one. For example, the first step is to create a budget, whose ID is then used to create a campaign in the next step.

2.0.2: Flow of method calls in original script (AdWords API)

```
# Create a Budget and save its ID
budget_id = _create_campaign_budget(adwords_client)
# Use the Budget's ID to create a Campaign
campaign_id = _create_campaign(adwords_client, budget_id)
# Use the Campaign's ID to create and AdGroup
ad_group_id = _create_ad_group(adwords_client, campaign_id)
# Use the AdGroup's ID to create Ads
_create_text_ads(adwords_client, ad_group_id)
# Use the AdGroup's ID to create keyword Criteria
_create_keywords(adwords_client, ad_group_id, KEYWORDS_TO_ADD)
```

Part 3 - Import the Google Ads client library

In this part we'll show how to import the Google Ads client library and initialize a Google Ads client so that it can make requests to the API.

Part 3.1 - Insert the necessary import statement(s):

To import the Google Ads client into the script, add this statement to the top of the file:

3.1.0: Import the Google Ads API client library

```
# imports the Google Ads client from the google-ads package
from google.ads.googleads.client import GoogleAdsClient
```

Part 3.2 - Define your Client customer ID as a global variable.

Where appropriate depending on the language you're using, add your the Client Customer ID of the test account you are using as a global variable in the script:

3.2.0: Insert your Client customer ID (Google Ads API)

```
# Set your test CID
_CUSTOMER_ID = "1234567890"
```

Part 3.3 - Initialize a Google Ads client:

As with the AdWords client library, we'll initialize a Google Ads API client using credentials stored in the local configuration file. Add the highlighted line below to the bottom of the script:

3.3.0: Initialize a Google Ads API client

```
if __name__ == "__main__":
    # Initialize the client object.
    # By default, it will read the config file from the Home Directory.
    adwords_client = adwords.AdWordsClient.LoadFromStorage()
    # Add the googleads_client here alongside the adwords_client
    googleads_client = GoogleAdsClient.load_from_storage()
    budget_id = _create_campaign_budget(adwords_client)
    campaign_id = _create_campaign(adwords_client, budget_id)
    ad_group_id = _create_ad_group(adwords_client, campaign_id)
    _create_text_ads(adwords_client, ad_group_id)
    _create_keywords(adwords_client, ad_group_id, KEYWORDS_TO_ADD)
```

Part 4 - Reporting Overview

In this part we'll go over how reporting works in the Google Ads API, how it differs from reporting in the AdWords API, and how we'll use it in this Codelab.

All reports in the Google Ads API go through the SearchStream and Search methods on the GoogleAdsService. These methods accept a GAQL string that specifies which resources should be retrieved, how they should be segmented, filtered, and ordered.

The SearchStream and Search methods are also the recommended way to retrieve individual entities from the API. In the AdWords API you would use a GET method, for example CampaignService.get. There are GET methods in the Google Ads API, but **they are only recommended for testing and debugging**.

The Google Ads API reporting interface is the same whether you want to retrieve resource attributes or performance data for entities, and in most cases you can retrieve this information with a single query.

When migrating a report from the AdWords API, start with the Report migration guide. It contains a query migration tool that converts AWQL queries into GAQL queries that you can pass into a search request. There's even a page that maps each AdWords report Google Ads resources and fields.

Here is a typical AdWords report query:

4.0.0: An AWQL query example (AdWords API)

```
SELECT
  CampaignId,
  AdGroupId,
  Impressions,
  Clicks,
  Cost
FROM ADGROUP_PERFORMANCE_REPORT
DURING LAST_7_DAYS
```

Using the query migration tool we can see that in Google Ads we should use:

4.0.1: A GAQL query example (Google Ads API)

```
SELECT
  campaign.id,
  ad_group.id,
  metrics.impressions,
  metrics.clicks,
  metrics.cost_micros
FROM ad_group
WHERE segments.date DURING LAST_7_DAYS
```

We can then pass that query into a `SearchStream` request as a string:

4.0.2: Reporting example in Google Ads API

```
googleads_service = client.get_type("GoogleAdsService")

query = """
    SELECT
      campaign.id,
      ad_group.id,
      metrics.impressions,
      metrics.clicks,
      metrics.cost_micros
    FROM ad_group
    WHERE segments.date DURING LAST_7_DAYS"""

stream = googleads_service.search_stream(customer_id=_CUSTOMER_ID, query=query)

for response in stream
    for row in response:
        print(
            f"AdGroup with ID {row.ad_group.id} and campaign ID {row.campaign.id} "
            f"has {metrics.impressions} impressions, {metrics.clicks} clicks, "
            f"and has cost {metrics.cost_micros} micros over the past seven days."
        )
```

Part 5 - Write a reporting method

In this part we're going to use the Reporting API in the Google Ads API to retrieve information about the entities created in this example.

This is not necessary in practice because you can use the resource name returned from a mutate response as an input to mutate requests. However, we will use this strategy to demonstrate how resource retrieval works in the Google Ads API.

In this script we'll use the SearchStream method, but it's possible to achieve the same functionality with the Search method. Let's add a new method that we can reuse throughout this script that accepts a GAQL query and returns the first GoogleAdsRow in the response

5.0.0: Method for retrieving entities with reporting (Google Ads API)

```
def _search_google_ads(client, query):
    """Queries the Google Ads API with the given query.

    This method will only return the first row of the query, assuming that
    the query only requests a single entity.

    Args:
        client: a GoogleAdsClient instance.
        query: a GAQL query string.

    Returns:
        A GoogleAdsRow instance or None.
    """
    googleads_service = client.get_service("GoogleAdsService")
    stream = googleads_service.search_stream(customer_id=_CUSTOMER_ID, query=query)
    # Passing the stream to the list constructor forces the stream to process to
    # completion so that we don't have to write a for loop.
    googleads_search_stream_response = list(stream)[0]
    try:
        # Retrieve the first result from the response
        return googleads_search_stream_response.results[0]
    except IndexError:
        # Return None if the response is empty
        return None
```

Note: For a general example of how to use the SearchStream method in any of our supported languages, see the Get Campaigns example under the Basic Operations directory. And for an example of how to use the Search method, see the Get Ad Groups example, also under the Basic Operations directory.

Pause

Part 6 - Create a budget

In this part we're going to rewrite the logic that creates a campaign budget with the AdWords API so that it uses the Google Ads API.

The `_create_campaign_budget` method creates a new budget with a few individual steps:

- Retrieve the necessary service client
- Create an object representing the resource we are creating
- Create and configure a mutate operation object
- Send the operation to the API in a mutate request
- Perform a searchStream request create an object for the newly created entity
- Return the the newly created entity

These steps apply to *all* the mutate methods throughout this script and are the building blocks for most mutate requests in the Google Ads API.

The first step is to swap out the AdWords service client being passed to the `_create_campaign_budget` method with the Google Ads Service client. Because the methods using the Google Ads API return resource objects as opposed to IDs in the AdWords API, let's rename the returned variable name accordingly. The resulting call to `_create_campaign_budget` should look like this:

6.0.0: Updating the call to _create_campaign_budget (Google Ads API)

```
budget = _create_campaign_budget(googleads_client)
```

In order for the following call to `_create_campaign` to work after making these changes, let's update the argument passed to the `_create_campaign` method to pass in the new budget's ID: :

6.0.1: Updating the call to _create_campaign (AdWords API)

```
campaign_id = _create_campaign(adwords_client, budget.id)
```

Part 6.1 - Retrieve the necessary service client:

Here is how the AdWords client library retrieves the BudgetService client:

6.1.0: Retrieving the BudgetService (AdWords API)

```
# Initializes a BudgetService client for the AdWords API.
budget_service = client.GetService("BudgetService", version="v201809")
```

We'll need the Google Ads equivalent of `BudgetService.` According to the map of AdWords services to Google Ads services, we'll need to use the `CampaignBudgetService`:

6.1.1: Retrieving the CampaignBudgetService (Google Ads API)

```
# Initializes a CampaignBudgetService client for the Google Ads API.
campaign_budget_service = client.get_service("CampaignBudgetService")
```

Part 6.2 - Create a mutate operation:

Here the AdWords API operation is a `dict` that's inserted into a `list`:

6.2.0: Initializing a budget operation (AdWords API)

```
budget_operations = [{"operator": "ADD", "operand": budget}]
```

In the Google Ads API these operations have distinct types specific to their respective services. Consult the services documentation for the latest API version to determine which operation class is needed. The `CampaignBudgetService.MutateCampaignBudgets` method requires a `CampaignBudgetOperation`:

6.2.1: Initializing a CampaignBudgetOperation (Google Ads API)

```
budget_operation = client.get_type("CampaignBudgetOperation")
```

Part 6.3 - Configure the operation object:

The AdWords client library accepts a `dict` that represents the budget:

6.3.0: Configuring a budget operation (AdWords API)

```
budget = {
    "name": "Interplanetary Cruise Budget #{}".format(uuid.uuid4()),
    "amount": {"microAmount": "50000000"},
    "deliveryMethod": "STANDARD",
}
```

The Google Ads client library includes a class for all resources in the API, and these classes mirror the structure of their protobuf definitions. Here's how to create a campaign budget and add it to the operation:

6.3.1: Configuring a CampaignBudgetOperation (Google Ads API)

```
# Retrieve the campaign_budget from the existing operation.
# Here we use the "create" oneof to specify that this is a
# "create" operation and not an "update" or a "remove". In
# Python you can access this field directly, in other languages
# you may need to set it explicitly.
campaign_budget = budget_operation.create
campaign_budget.name = f"Interplanetary Cruise Budget #{uuid.uuid4()}"
campaign_budget.amount_micros = 50000000
# Enums also have their own classes. The below is special syntax for enums in
# the Python client library and is the equivalent of calling
# client.get_type("BudgetDeliveryMethodEnum").
campaign_budget.delivery_method = client.enums.BudgetDeliveryMethodEnum.STANDARD
```

Part 6.4 - Send the operation to the API in a mutate request:

6.4.0: Submitting a budget mutate request (AdWords API)

```
# Sends the operation to the AdWords API in a mutate request.
results = budget_service.mutate(budget_operations)
```

This step is very similar with the Google Ads client library:

6.4.1: Submitting a budget mutate request (Google Ads API)

```
# Sends the operation to the Google Ads API in a mutate request.
# Note that the Google Ads client library requires the customer ID
# to be passed into the service method, whereas the AdWords client
# library reads the customer ID from configuration.
response = campaign_budget_service.mutate_campaign_budgets(
    customer_id=_CUSTOMER_ID, operations=[budget_operation]
)
```

Part 6.5 - Get information about the new object:

In this script the newly created budget is returned by the response object:

6.5.0: Parsing a budget mutate response (AdWords API)

```
created_budget = results["value"][0]
```

The same is not true by default in the Google Ads API. The default behavior is for mutate requests to *only* return the entity's resource name.

6.5.1: Parsing a budget mutate response (Google Ads API)

```
resource_name = response.results[0].resource_name
```

Note: In the Google Ads API it's possible to configure a mutate request so that the response contains the entire object, not just its resource name. We'll look more closely at this functionality later in Part 10.

Unlike the previous version of this method, we'll return an instance of the campaign budget entity, and will use the reporting method defined in Part 5 to retrieve it.

6.5.2: Using Google Ads API reporting to retrieve the full budget object

```
query = f"""
    SELECT
      campaign_budget.id,
      campaign_budget.name,
      campaign_budget.resource_name
    FROM campaign_budget
    WHERE campaign_budget.resource_name = '{resource_name}'"""

googleads_row = _search_google_ads(client, query)

return googleads_row.campaign_budget
```

Part 6.6 - Finished `_create_campaign_budget` method:

6.6.0: Complete _create_campaign_budget method (Google Ads API)

```
def _create_campaign_budget(client):
    """Creates a new campaign budget and returns a subset of its fields.

    Only the fields selected in the below GAQL query will be present on
    the returned campaign budget.

    Args:
        client: An instance of the GoogleAdsClient class.

    Returns:
        (CampaignBudget) The newly created budget with a subset of its fields.
    """
    budget_service = client.get_service("CampaignBudgetService")

    budget_operation = client.get_type("CampaignBudgetOperation")
    # Retrieve the campaign_budget from the existing operation
    campaign_budget = budget_operation.create
    campaign_budget.name = f"Interplanetary Cruise Budget #{uuid.uuid4()}"
    campaign_budget.amount_micros = 50000000
    # Enums also have their own classes. The below is special syntax for enums in
    # the Python client library and is the equivalent of calling
    # client.get_type("BudgetDeliveryMethodEnum").
    campaign_budget.delivery_method = (
        client.enums.BudgetDeliveryMethodEnum.STANDARD
    )
    # Sends the operation to the Google Ads API in a mutate request.
    # Note that the Google Ads client library requires the customer ID
    # to be passed into the service method, whereas the AdWords client
    # library reads the customer ID from configuration.

    response = campaign_budget_service.mutate_campaign_budgets(
        customer_id=_CUSTOMER_ID, operations=[budget_operation]
    )

    resource_name = response.results[0].resource_name

    query = f"""
        SELECT
          campaign_budget.id,
          campaign_budget.name,
          campaign_budget.resource_name
        FROM campaign_budget
        WHERE campaign_budget.resource_name = '{resource_name}'"""

    googleads_row = _search_google_ads(client, query)

    return googleads_row.campaign_budget
```

Let's run the script to ensure everything is still working as expected:

```
python create_complete_campaign_adwords_api_only.py
```

Pause

Part 7 - Create a campaign

In this part we're going to rewrite the logic that creates a campaign with the AdWords API so that it uses the Google Ads API.

To migrate the `_create_campaign` method we'll follow the same steps as for creating a budget. First let's update the call to `_create_campaign` to pass in the Google Ads client and a budget, and to return a campaign entity:

7.0.0: Updating the call to _create_campaign (Google Ads API)

```
campaign = _create_campaign(googleads_client, budget)
```

Next, let's update the method signature to rename the `budget_id` parameter to just `budget`:

7.0.1: Updating the _create_campaign method signature (Google Ads API)

```
def _create_campaign(client, budget):
```

Lastly, update the call to `_create_ad_group` so that it isn't broken by the changes in this section:

7.0.2: Updating the call to _create_ad_group (AdWords API)

```
ad_group_id = _create_ad_group(adwords_client, campaign.id)
```

Part 7.1 - Retrieve the necessary service client:

The AdWords client library uses the CampaignService, and using the map of AdWords services to Google Ads services we can see that the Google Ads client library should use a service with the same name:

7.1.0: Retrieving the CampaignService (AdWords API)

```
campaign_service = client.GetService("CampaignService", version="v201809")
```

should be changed to:

7.1.1: Retrieving the CampaignService (Google Ads API)

```
campaign_service = client.get_service("CampaignService")
```

Part 7.2 - Create and configure a mutate operation:

7.2.0: Initializing and configuring a campaign operation (AdWords API)

```
campaign = {
    "name": "Interplanetary Cruise #{}".format(uuid.uuid4()),
    "advertisingChannelType": "SEARCH",
    # Recommendation: Set the campaign to PAUSED when creating it to stop the
    # ads from immediately serving. Set to ENABLED once you've added
    # targeting and the ads are ready to serve.
    "status": "PAUSED",
    "biddingStrategyConfiguration": {
        "biddingStrategyType": "MANUAL_CPC",
    },
    "startDate": (datetime.datetime.now() + datetime.timedelta(1)).strftime(
        "%Y%m%d"
    ),
    "endDate": (datetime.datetime.now() + datetime.timedelta(365)).strftime(
        "%Y%m%d"
    ),
    # Budget (required) - note only the budget ID is required.
    "budget": {"budgetId": budget_id},
    "networkSetting": {
        "targetGoogleSearch": "true",
        "targetSearchNetwork": "true",
    },
}
campaign_operations = [{"operator": "ADD", "operand": campaign}]
```

should be changed to:

7.2.1: Initializing and configuring a CampaignOperation (Google Ads API)

```
operation = client.get_type("CampaignOperation")
campaign = operation.create
campaign_service = client.get_service("CampaignService")
campaign.name = f"Interplanetary Cruise#{uuid.uuid4()}"
campaign.advertising_channel_type = (
    client.enums.AdvertisingChannelTypeEnum.SEARCH
)
# Recommendation: Set the campaign to PAUSED when creating it to stop # the ads
from immediately serving. Set to ENABLED once you've added
# targeting and the ads are ready to serve.
campaign.status = client.enums.CampaignStatusEnum.PAUSED
campaign.manual_cpc.enhanced_cpc_enabled = True
campaign.campaign_budget = budget.resource_name
campaign.network_settings.target_google_search = True
campaign.network_settings.target_search_network = True
campaign.network_settings.target_content_network = False
campaign.network_settings.target_partner_search_network = False
# The format used for the below dates evaluates to 2021-08-18.
campaign.start_date = (
    datetime.datetime.now() + datetime.timedelta(1)
).strftime("%Y-%m-%d")
campaign.end_date = (
    datetime.datetime.now() + datetime.timedelta(365)
).strftime("%Y-%m-%d")
```

Part 7.3 - Send the operation to the API in a mutate request:

7.3.0: Submitting a campaign mutate request (AdWords API)

```
results = campaign_service.mutate(campaign_operations)
```

should be changed to:

7.3.1: Submitting a campaign mutate request (Google Ads API)

```
response = campaign_service.mutate_campaigns(
    customer_id=_CUSTOMER_ID, operations=[operation]
)
```

Part 7.4 - Get information about the new object:

Start by retrieving the campaign's resource name:

7.4.0: Parsing a campaign mutate response (AdWords API)

```
created_campaign = results["value"][0]
```

should be changed to:

7.4.1: Parsing a campaign mutate response (Google Ads API)

```
campaign_resource_name = response.results[0].resource_name
```

Then use the `_search_google_ads` method with the below query to retrieve and return the campaign:

7.4.2: Using Google Ads API reporting to retrieve the full campaign object

```
query = f"""
    SELECT
      campaign.id,
      campaign.name,
      campaign.resource_name
    FROM campaign
    WHERE campaign.resource_name = '{resource_name}'"""

googleads_row = _search_google_ads(client, query)

return googleads_row.campaign
```

Part 7.5 - Finished `_create_campaign` method:

7.5.0: Complete _create_campaign method (Google Ads API)

```
def _create_campaign(client, budget):
    """Creates a new campaign and returns a subset of its fields.

    Only the fields selected in the below GAQL query will be present on
    the returned campaign.

    Args:
        client: An instance of the GoogleAdsClient class.
        budget: A CampaignBudget instance.

    Returns:
        (Campaign) The newly created campaign with a subset of its fields.
    """
    campaign_service = client.get_service("CampaignService")

    operation = client.get_type("CampaignOperation")
    campaign = operation.create
    campaign_service = client.get_service("CampaignService")
    campaign.name = f"Interplanetary Cruise#{uuid.uuid4()}"
    campaign.advertising_channel_type = (
        client.enums.AdvertisingChannelTypeEnum.SEARCH
    )
    # Recommendation: Set the campaign to PAUSED when creating it to stop
    # the ads from immediately serving. Set to ENABLED once you've added
    # targeting and the ads are ready to serve.
    campaign.status = client.enums.CampaignStatusEnum.PAUSED
    campaign.manual_cpc.enhanced_cpc_enabled = True
    campaign.campaign_budget = budget.resource_name
    campaign.network_settings.target_google_search = True
    campaign.network_settings.target_search_network = True
    campaign.network_settings.target_content_network = False
    campaign.network_settings.target_partner_search_network = False
    campaign.start_date = (
        datetime.datetime.now() + datetime.timedelta(1)
    ).strftime("%Y%m%d")
    campaign.end_date = (
        datetime.datetime.now() + datetime.timedelta(365)
    ).strftime("%Y%m%d")

    response = campaign_service.mutate_campaigns(
        customer_id=_CUSTOMER_ID, operations=[operation]
    )
```

```
resource_name = response.results[0].resource_name

query = f"""
    SELECT
      campaign.id,
      campaign.name,
      campaign.resource_name
    FROM campaign
    WHERE campaign.resource_name = '{resource_name}'"""

googleads_row = _search_google_ads(client, query)

return googleads_row.campaign
```

Let's run the script to ensure everything is still working as expected:

```
python create_complete_campaign_adwords_api_only.py
```

Pause

Part 8 - Create an ad group

In this part we're going to rewrite the logic that creates an ad group with the AdWords API so that it uses the Google Ads API.

Once again we'll follow the same set of steps that we did in the last two sections. First let's update the calls to `_create_ad_group` and its method signature.

First, update the method signature to accept the Google Ads client and a campaign (not just a campaign ID), and to return an ad group:

8.0.0: Updating the call to _create_ad_group (Google Ads API)

```
ad_group = _create_ad_group(googleads_client, campaign)
```

Update the method signature so it accepts a campaign:

8.0.1: Updating the _create_ad_group method signature (Google Ads API)

```
def _create_ad_group(client, campaign):
```

Lastly update the calls to `_create_text_ads` and `_create_keywords` to pass the ad group's ID as the second parameter:

8.0.2: Updating the calls to _create_text_ads and _create_keywords (AdWords API)

```
_create_text_ads(adwords_client, ad_group.id)
_create_keywords(adwords_client, ad_group.id, KEYWORDS_TO_ADD)
```

Part 8.1 - Retrieve the necessary service client:

According to the map of AdWords services to Google Ads services we will need to use the AdGroupService in the Google Ads API. So:

8.1.0: Retrieving the AdGroupService (AdWords API)

```
ad_group_service = client.GetService("AdGroupService", "v201809")
```

should be changed to:

8.1.1: Retrieving the AdGroupService (Google Ads API)

```
ad_group_service = client.get_type("AdGroupService")
```

Part 8.2 - Create and configure a mutate operation:

8.2.0: Initializing and configuring an ad group operation (AdWords API)

```
ad_group = {
    "name": "Earth to Mars Cruise #{}".format(uuid.uuid4()),
    "campaignId": campaign_id,
    "status": "ENABLED",
    "biddingStrategyConfiguration": {
        "bids": [
            {
                # The 'xsi_type' field allows you to specify the xsi:type of the
                # object being created. It's only necessary when you must provide
                # an explicit type that the client library can't infer.
                "xsi_type": "CpcBid",
                "bid": {"microAmount": 10000000},
            }
        ]
    },
    "adGroupAdRotationMode": "OPTIMIZE",
}

adgroup_operations = [{"operator": "ADD", "operand": ad_group}]
```

should be changed to:

8.2.1: Initializing and configure an ad group operation (Google Ads API)

```
operation = client.get_type("AdGroupOperation")
ad_group = operation.create
ad_group.name = f"Earth to Mars Cruises #{uuid.uuid4()}"
ad_group.campaign = campaign.resource_name
ad_group.status = client.enums.AdGroupStatusEnum.ENABLED
ad_group.type = client.enums.AdGroupTypeEnum.SEARCH_STANDARD
ad_group.cpc_bid_micros = 10000000
```

Part 8.3 - Send the operation to the API in a mutate request:

8.3.0: Submitting an ad group mutate request (AdWords API)

```
results = ad_group_service.mutate(adgroup_operations)
```

should be changed to:

8.3.1: Submitting an ad group mutate request (Google Ads API)

```
response = adgroup_service.mutate_ad_groups(
    customer_id=_CUSTOMER_ID, operations=[operation]
)
```

Part 8.4 - Get information about the new object:

8.4.0: Parsing an ad group mutate response (AdWords API)

```
created_ad_group = results["value"][0]
```

should be changed to:

8.4.1: Parsing an ad group mutate response (Google Ads API)

```
ad_group_resource_name = response.results[0].resource_name
```

Then use the `_search_google_ads` method with the below query to retrieve and return the campaign:

8.4.2: Using Google Ads API reporting to retrieve the full ad group object

```
query = f"""
    SELECT
      ad_group.id,
      ad_group.name,
      ad_group.resource_name
    FROM ad_group
    WHERE ad_group.resource_name = '{resource_name}'"""

googleads_row = _search_google_ads(client, query)

return googleads_row.ad_group
```

Part 8.5 - Finished `_create_ad_group` method:

8.5.0: Complete _create_ad_group method (Google Ads API)

```
def _create_ad_group(client, campaign):
    """Creates a new ad group and returns a subset of its fields.

    Only the fields selected in the below GAQL query will be present on
    the returned campaign.

    Args:
        client: An instance of the GoogleAdsClient class.
        campaign: A Campaign instance.

    Returns:
        (AdGroup) The newly created ad group with a subset of its fields.
    """
    ad_group_service = client.get_type("AdGroupService")

    operation = client.get_type("AdGroupOperation")
    ad_group = operation.create
    ad_group.name = f"Earth to Mars Cruises #{uuid.uuid4()}"
    ad_group.campaign = campaign.resource_name
    ad_group.status = client.enums.AdGroupStatusEnum.ENABLED
    ad_group.type = client.enums.AdGroupTypeEnum.SEARCH_STANDARD
    ad_group.cpc_bid_micros = 10000000

    response = adgroup_service.mutate_ad_groups(
        customer_id=_CUSTOMER_ID, operations=[operation]
    )

    ad_group_resource_name = response.results[0].resource_name

    query = f"""
    SELECT
      ad_group.id,
      ad_group.name,
      ad_group.resource_name
    FROM ad_group
    WHERE ad_group.resource_name = '{resource_name}'"""

    googleads_row = _search_google_ads(client, query)
    return googleads_row.ad_group
```

Let's run the script to ensure everything is still working as expected:

```
python create_complete_campaign_adwords_api_only.py
```

Pause

Part 9 - Create text ads

In this part we're going to rewrite the logic that creates ad group ads with the AdWords API so that it uses the Google Ads API.

Again, we'll follow the same set of steps that we did in the last few sections. The main differences here are that the `_create_text_ads` method sends multiple mutate operations instead of just one, and our methods will not return anything since the entities created aren't used in subsequent operations. This is a good opportunity to demonstrate the use of the `response_content_type` request header.

First update the method call and signature to accept an ad group instead of just an ad group ID:

9.0.0: Updating the call to create_text_ads (Google Ads API)

```
_create_text_ads(googleads_client, ad_group)
```

9.0.1: Updating the _create_text_ads method signature (Google Ads API)

```
def _create_text_ads(client, ad_group):
```

Part 9.1 - Retrieve the necessary service client:

According to the map of AdWords services to Google Ads services we will need to use the AdGroupAdService in the Google Ads API. So:

9.1.0: Retrieving the AdGroupAdService (AdWords API)

```
ad_group_ad_service = client.GetService("AdGroupAdService", "v201809")
```

should be changed to:

9.1.1: Retrieving the AdGroupAdService (Google Ads API)

```
ad_group_ad_service = client.get_type("AdGroupAdService")
```

Part 9.2 - Create and configure a mutate operation:

Here, a loop creates multiple operations. We'll update the entire loop at once:

9.2.0: Initializing and configuring ad group ad operations (AdWords API)

```
operations = []
for i in range(NUMBER_OF_ADS):
    operation = {
        "xsi_type": "AdGroupAd",
        "adGroupId": ad_group_id,
        # Additional properties (non-required).
        "status": "PAUSED",
        "ad": {
            "xsi_type": "ExpandedTextAd",
            "headlinePart1": "Cruise #{} to Mars".format(
                str(uuid.uuid4())[:8]
            ),
            "headlinePart2": "Best Space Cruise Line",
            "headlinePart3": "For Your Loved Ones",
            "description": "Buy your tickets now!",
            "description2": "Discount ends soon",
            "finalUrls": ["http://www.example.com/"],
        },
    }
    adgroup_operations = {"operator": "ADD", "operand": operation}
    operations.append(adgroup_operations)
```

should be changed to:

9.2.1: Initializing and configuring ad group ad operations (Google Ads API)

```
operations = []
for i in range(0, NUMBER_OF_ADS):
    operation = client.get_type("AdGroupAdOperation")
    ad_group_ad_operation = operation.create
    ad_group_ad_operation.ad_group = ad_group.resource_name
    ad_group_ad_operation.status = client.enums.AdGroupAdStatusEnum.PAUSED
    ad_group_ad_operation.ad.expanded_text_ad.headline_part1 = (
        f"Cruise to Mars #{str(uuid.uuid4())[:4]}"
    )
    ad_group_ad_operation.ad.expanded_text_ad.headline_part2 = (
        "Best Space Cruise Line"
    )
    ad_group_ad_operation.ad.expanded_text_ad.description = (
        "Buy your tickets now!"
    )
    ad_group_ad_operation.ad.final_urls.append("http://www.example.com")
    operations.append(operation)
```

Part 9.3 - Send the operation to the API in a mutate request:

9.3.0: Submitting an ad group ad mutate request (AdWords API)

```
results = ad_group_service.mutate(operations)
```

should be changed to the following. Notice that we construct the request object outside of the mutate call. This is necessary in order to set optional header fields, namely the `response_content_type` header. We determine the name of the request object class by referring to the AdGroupAdService.MutateAdGroupAds method documentation.

9.3.1: Submitting an ad group ad mutate request (Google Ads API)

```
request = client.get_type("MutateAdGroupAdsRequest")
request.customer_id = _CUSTOMER_ID
request.operations = operations
request.response_content_type = (
    client.enums.ResponseContentTypeEnum.MUTABLE_RESOURCE
)

response = ad_group_ad_service.mutate_ad_group_ads(request=request)
```

Part 9.4 - Print information about the new objects:

This method only prints details about the newly created entities, so there is no need to return anything:

9.4.0: Parsing an ad group ad mutate response (AdWords API)

```
for result in results["value"]:
    print(
        "Expanded text ad with ID {} and "
        "headline {}-{} {} was created".format(
            result["ad"]["id"],
            result["ad"]["headlinePart1"],
            result["ad"]["headlinePart2"],
            result["ad"]["headlinePart3"],
        )
    )
```

should be changed to the below:

9.4.1: Parsing an ad group ad mutate response (Google Ads API)

```
for ad_group_ad in response.results:
    print(
        f"Created expanded text ad with ID {ad_group_ad.ad.id}, "
        f"status {ad_group_ad.ad.status} and headline "
        f"{ag_group_ad.ad.expanded_text_ad.headline_part1}."
        f"{ad_group_ad.ad.expanded_text_ad.headline_part2}"
    )
```

Part 9.5 - Finished `_create_text_ads` method:

9.5.0: Complete _create_text_ads method (Google Ads API)

```
def _create_text_ads(client, ad_group):
    """Creates a set of new text ads and prints information about them.

    Args:
        client: An instance of the GoogleAdsClient class.
        ad_group: An AdGroup instance.
    """
    ad_group_ad_service = client.get_type("AdGroupAdService")

    operations = []
    for i in range(0, NUMBER_OF_ADS):
        operation = client.get_type("AdGroupAdOperation")
        ad_group_operation = operation.create
        ad_group_operation.ad_group = ad_group.resource_name
        ad_group_operation.status = client.enums.AdGroupAdStatusEnum.PAUSED
        ad_group_operation.ad.expanded_text_ad.headline_part1 = (
            f"Cruise to Mars #{str(uuid.uuid4())[:4]}"
        )
        ad_group_operation.ad.expanded_text_ad.headline_part2 = (
            "Best Space Cruise Line"
        )
        ad_group_operation.ad.expanded_text_ad.description = (
            "Buy your tickets now!"
        )
        ad_group_operation.ad.final_urls.append("http://www.example.com")
        operations.append(operation)

    request = client.get_type("MutateAdGroupAdsRequest")
    request.customer_id = _CUSTOMER_ID
    request.operations = operations
    request.response_content_type = (
        client.enums.ResponseContentTypeEnum.MUTABLE_RESOURCE
    )

    response = ad_group_ad_service.mutate_ad_group_ads(request=request)

    for ad_group_ad in response.results:
    print(
        f"Created expanded text ad with ID {ad_group_ad.ad.id}, "
        f"status {ad_group_ad.ad.status} and headline "
        f"{ag_group_ad.ad.expanded_text_ad.headline_part1}."
        f"{ad_group_ad.ad.expanded_text_ad.headline_part2}"
    )
```

Let's run the script to ensure everything is still working as expected:

```
python create_complete_campaign_adwords_api_only.py
```

Pause

Part 10 - Create keyword criteria

In this part we're going to rewrite the logic that creates ad group criterion with the AdWords API so that it uses the Google Ads API.

The `_create_keywords` method is very similar to the `_create_text_ads` method in that it builds multiple mutate operations and prints the details of the newly created entities.

First update the method call and signature to accept and ad group instead of just an ad group ID:

10.0.0: Updating the call to _create_keywords (Google Ads API)

```
_create_keywords(googleads_client, ad_group, KEYWORDS_TO_ADD)
```

10.0.1: Updating the _create_keywords method signature (Google Ads API)

```
def _create_keywords(client, ad_group, keywords_to_add):
```

Part 10.1 - Retrieve the necessary service client:

According to the map of AdWords services to Google Ads services we will need to use the AdGroupCriterionService in the Google Ads API. So:

11.1.0: Retrieving the AdGroupCriterionService (AdWords API)

```
ad_group_criterion_service = client.GetService(
    "AdGroupCriterionService", "v201809"
)
```

should be changed to:

10.1.1: Retrieving the AdGroupCriterionService (Google Ads API)

```
ad_group_criterion_service = client.get_type("AdGroupCriterionService")
```

Part 10.2 - Create and configure a mutate operation:

Here, a loop creates multiple operations. We'll update the entire loop at once:

10.2.0: Initializing and configuring ad group criterion operations (AdWords API)

```
operations = []
for keyword in keywords_to_add:
    operation = {
        "xsi_type": "BiddableAdGroupCriterion",
        "adGroupId": ad_group_id,
        "criterion": {
            "xsi_type": "Keyword",
            "text": keyword,
            "matchType": "BROAD",
        },
        "userStatus": "PAUSED",
        "finalUrls": [
            "http://www.example.com/mars/cruise/?kw={}".format(
                urllib.parse.quote(keyword)
            )
        ],
    }
    create_keyword = {"operator": "ADD", "operand": operation}
    operations.append(create_keyword)
```

should be changed to:

10.2.1: Initializing and configuring AdGroupCriterionOperations (Google Ads API)

```
operations = []
for keyword in keywords_to_add:
    operation = client.get_type("AdGroupCriterionOperation")
    ad_group_criterion = operation.create
    ad_group_criterion.ad_group = ad_group.resource_name
    ad_group_criterion.status = (
        client.enums.AdGroupCriterionStatusEnum.ENABLED
    )
    ad_group_criterion.keyword.text = keyword
    ad_group_criterion.keyword.match_type = (
        client.enums.KeywordMatchTypeEnum.EXACT
    )
    operations.append(operation)
```

Part 10.3 - Send the operation to the API in a mutate request:

10.3.0: Submitting an ad group criterion mutate request (AdWords API)

```
results = ad_group_criterion_service.mutate(operations)
```

should be changed to the following:

10.3.1: Submitting an ad group criterion mutate request (Google Ads API)

```
request = client.get_type("MutateAdGroupCriteriaRequest")
request.customer_id = _CUSTOMER_ID
request.operations = operations
request.response_content_type = (
    client.enums.ResponseContentTypeEnum.MUTABLE_RESOURCE
)

response = ad_group_criterion_service.mutate_ad_group_criteria(request=request)
```

Notice that we construct the request object outside of the mutate call. This is necessary in Python in order to set optional header fields, namely the `response_content_type` header. We determine the name of the request object class by referring to the AdGroupCriterionService.MutateAdGroupCriteria method documentation.

Part 10.4 - Print information about the new objects:

This method only prints details about the newly created entities, so there is no need to return anything:

10.4.0: Parsing an ad group criterion mutate mutate response (AdWords API)

```
for result in results["value"]:
    print(
        "Keyword with ad group ID {}, keyword ID {}, text {} and match"
        "type {} was created".format(
            result["adGroupId"],
            result["criterion"]["id"],
            result["criterion"]["text"],
            result["criterion"]["matchType"],
        )
    )
```

should be changed to the following:

10.4.1: Parsing an ad group criterion mutate mutate response (Google Ads API)

```
for criterion in response.results:
    print(
        f"Keyword with text {criterion.keyword.text}, id="
        f"{criterion.criterion_id} and match type "
        f"{criterion.keyword.match_type} was created"
    )
```

Part 10.5 - Finished `_create_keywords` method:

10.5.0: Complete _create_keywords method (Google Ads API)

```
def _create_keywords(client, ad_group, keywords_to_add):
    """Creates a set of new keyword criteria and prints information about them.

    Args:
        client: An instance of the GoogleAdsClient class.
        ad_group: An AdGroup instance.
         Keywords_to_add: a list of keyword strings
    """
    ad_group_criterion_service = client.get_type("AdGroupCriterionService")

    operations = []
    for keyword in keywords_to_add:
        operation = client.get_type("AdGroupCriterionOperation")
        ad_group_criterion_operation = operation.create
        ad_group_criterion_operation.ad_group = ad_group.resource_name
        ad_group_criterion_operation.status = (
            client.enums.AdGroupCriterionStatusEnum.ENABLED
        )
        ad_group_criterion_operation.keyword.text = keyword
        ad_group_criterion_operation.keyword.match_type = (
            client.enums.KeywordMatchTypeEnum.EXACT
        )
        operations.append(operation)

    request = client.get_type("MutateAdGroupCriteriaRequest")
    request.customer_id = _CUSTOMER_ID
    request.operations = operations
    request.response_content_type = (
        client.enums.ResponseContentTypeEnum.MUTABLE_RESOURCE
    )

    response = ad_group_criterion_service.mutate_ad_group_criteria(request=request)

    for criterion in response.results:
        print(
            f"Keyword with text {criterion.keyword.text}, id="
            f"{criterion.criterion_id} and match type "
            f"{criterion.keyword.match_type} was created"
        )
```

Let’s run the script to ensure everything is still working as expected:

10.5.1: Shell command to run the script

```
python create_complete_campaign_adwords_api_only.py
```

Appendix

Logs examples

Here's an example of how the logs should appear when running this script using only AdWords API, and with the logging level set to INFO:

A0: Example logs using AdWords API Only

```
[2021-08-19 08:58:30,020 - googleads.soap - INFO] Request made: Service:
"BudgetService" Method: "mutate" URL:
"https://adwords.google.com/api/adwords/cm/v201809/BudgetService"
Budget with ID 9312798435 and name Interplanetary Cruise Budget
#72ff2cce-592f-4f1a-8371-0cf3c5d95aa5 was created
[2021-08-19 08:58:30,333 - googleads.soap - INFO] Request made: Service:
"CampaignService" Method: "mutate" URL:
"https://adwords.google.com/api/adwords/cm/v201809/CampaignService"
CreatedCampign with ID 14331639971 and name Interplanetary Cruise
#ba24d9ab-2a0e-4eb4-b5b5-3ab8a9f9666a was created
[2021-08-19 08:58:30,678 - googleads.soap - INFO] Request made: Service:
"AdGroupService" Method: "mutate" URL:
"https://adwords.google.com/api/adwords/cm/v201809/AdGroupService"
Ad group with ID 130020093990 and name Earth to Mars Cruise
#75c50df3-710f-48e7-9baf-e16b421e2e49 was created
[2021-08-19 08:58:31,017 - googleads.soap - INFO] Request made: Service:
"AdGroupAdService" Method: "mutate" URL:
"https://adwords.google.com/api/adwords/cm/v201809/AdGroupAdService"
Expanded text ad with ID 540256098380 and headline Cruise #33b609fd to Mars-Best
Space Cruise Line For Your Loved Ones was created
Expanded text ad with ID 540256098383 and headline Cruise #131745fe to Mars-Best
Space Cruise Line For Your Loved Ones was created
Expanded text ad with ID 540256098386 and headline Cruise #f4065b21 to Mars-Best
Space Cruise Line For Your Loved Ones was created
Expanded text ad with ID 540256098389 and headline Cruise #5c8c9f19 to Mars-Best
Space Cruise Line For Your Loved Ones was created
Expanded text ad with ID 540256098392 and headline Cruise #d28e7da7 to Mars-Best
Space Cruise Line For Your Loved Ones was created
[2021-08-19 08:58:31,722 - googleads.soap - INFO] Request made: Service:
"AdGroupCriterionService" Method: "mutate" URL:
"https://adwords.google.com/api/adwords/cm/v201809/AdGroupCriterionService"
Keyword with ad group ID 130020093990, keyword ID 376765347, text mars cruise and
matchtype BROAD was created
Keyword with ad group ID 130020093990, keyword ID 142242236, text space hotel and
matchtype BROAD was created
```

Here’s an example of how the logs should appear when running this script using only Google Ads API, and with the logging level set to INFO:

A1: Example logs using Google Ads API Only

```
[2021-08-19 09:02:40 - INFO] Request made: ClientCustomerId: 1234567890, Host:
qa-prod-googleads.sandbox.googleapis.com, Method:
/google.ads.googleads.v8.services.CampaignBudgetService/MutateCampaignBudgets,
RequestId: None, IsFault: False, FaultMessage: None
[2021-08-19 09:02:41 - INFO] Request made: ClientCustomerId: 1234567890, Host:
qa-prod-googleads.sandbox.googleapis.com, Method:
/google.ads.googleads.v8.services.GoogleAdsService/Search, RequestId: None,
IsFault: False, FaultMessage: None
Added budget named Interplanetary Cruise Budget
c3ac6ec0-b228-4741-8c16-d877fe91e0d1
[2021-08-19 09:02:41 - INFO] Request made: ClientCustomerId: 1234567890, Host:
qa-prod-googleads.sandbox.googleapis.com, Method:
/google.ads.googleads.v8.services.CampaignService/MutateCampaigns, RequestId: None,
IsFault: False, FaultMessage: None
[2021-08-19 09:02:42 - INFO] Request made: ClientCustomerId: 1234567890, Host:
qa-prod-googleads.sandbox.googleapis.com, Method:
/google.ads.googleads.v8.services.GoogleAdsService/Search, RequestId: None,
IsFault: False, FaultMessage: None
Added campaign named Interplanetary Cruise#ee5ec035-53f4-482a-991c-1ad1d1f0923a
[2021-08-19 09:02:42 - INFO] Request made: ClientCustomerId: 1234567890, Host:
qa-prod-googleads.sandbox.googleapis.com, Method:
/google.ads.googleads.v8.services.AdGroupService/MutateAdGroups, RequestId: None,
IsFault: False, FaultMessage: None
[2021-08-19 09:02:43 - INFO] Request made: ClientCustomerId: 1234567890, Host:
qa-prod-googleads.sandbox.googleapis.com, Method:
/google.ads.googleads.v8.services.GoogleAdsService/Search, RequestId: None,
IsFault: False, FaultMessage: None
Added AdGroup named Earth to Mars Cruises #acea416a-053a-40b7-bf23-7c2f76be9591
[2021-08-19 09:02:43 - INFO] Request made: ClientCustomerId: 1234567890, Host:
qa-prod-googleads.sandbox.googleapis.com, Method:
/google.ads.googleads.v8.services.AdGroupAdService/MutateAdGroupAds, RequestId:
None, IsFault: False, FaultMessage: None
[2021-08-19 09:02:44 - INFO] Request made: ClientCustomerId: 1234567890, Host:
qa-prod-googleads.sandbox.googleapis.com, Method:
/google.ads.googleads.v8.services.GoogleAdsService/Search, RequestId: None,
IsFault: False, FaultMessage: None
Created expanded text ad with ID 540192815485, status 3 and headline Cruise to Mars
#d253.Best Space Cruise Line
Created expanded text ad with ID 540192815488, status 3 and headline Cruise to Mars
```

```
#2318.Best Space Cruise Line
Created expanded text ad with ID 540192815491, status 3 and headline Cruise to Mars
#959a.Best Space Cruise Line
Created expanded text ad with ID 540192815494, status 3 and headline Cruise to Mars
#5406.Best Space Cruise Line
Created expanded text ad with ID 540192815497, status 3 and headline Cruise to Mars
#c83f.Best Space Cruise Line
[2021-08-19 09:02:44 - INFO] Request made: ClientCustomerId: 1234567890, Host:
qa-prod-googleads.sandbox.googleapis.com, Method:
/google.ads.googleads.v8.services.AdGroupCriterionService/MutateAdGroupCriteria,
RequestId: None, IsFault: False, FaultMessage: None
[2021-08-19 09:02:45 - INFO] Request made: ClientCustomerId: 1234567890, Host:
qa-prod-googleads.sandbox.googleapis.com, Method:
/google.ads.googleads.v8.services.GoogleAdsService/Search, RequestId: None,
IsFault: False, FaultMessage: None
Keyword with text space hotel, id=140068236 and match type 2 was created
Keyword with text mars cruise, id=302272685926 and match type 2 was created
```