

Analytics Reporter

A lightweight system for publishing analytics data from the Digital Analytics Program (DAP) Google Analytics 4 government-wide property. This project uses the Google Analytics Data API v1 to acquire analytics data and then processes it into a flat data structure.

The project previously used the Google Analytics Core Reporting API v3 and the Google Analytics Real Time API v3, also known as Universal Analytics, which has slightly different data points. See [Upgrading from Universal Analytics](#) for more details. The Google Analytics v3 API will be deprecated on July 1, 2024.

This is used in combination with [analytics-reporter-api](#) to power the government analytics website, [analytics.usa.gov](#).

Available reports are named and described in [api.json](#) and [usa.json](#). For now, they're hard-coded into the repository.

Local development setup

Prerequisites

- NodeJS > v20.x
- A postgres DB running and/or docker installed

Install dependencies

```
1 npm install
```

Linting

This repo uses ESLint and Prettier for code static analysis and formatting. Run the linter with:

```
1 npm run lint
```

Automatically fix lint issues with:

```
1 npm run lint:fix
```

Install git hooks

There are some git hooks provided in the `./hooks` directory to help with common development tasks. These will checkout current NPM packages on branch change events, and run the linter on pre-commit.

Install the provided hooks with the following command:

```
1 npm run install-git-hooks
```

Running the unit tests

The unit tests for this repo require a local PostgreSQL database. You can run a local DB server or create a docker container using the provided test compose file. (Requires docker and docker-compose to be installed)

Starting a docker test DB:

```
1 docker-compose -f docker-compose.test.yml up
```

Once you have a PostgreSQL DB running locally, you can run the tests. The test DB connection in `knexfile.js` has some default connection config which can be overridden with environment variables. If using the provided docker-compose DB then you can avoid setting the connection details.

Run the tests (pre-test hook runs DB migrations):

```
1 npm test
```

Running the unit tests with code coverage reporting If you wish to see a code coverage report after running the tests, use the following command. This runs the DB migrations, tests, and the NYC code coverage tool:

```
1 npm run coverage
```

Running the integration tests

The integration tests for this repo require the google analytics credentials to be set in the environment. This can be setup with the `dotenv-cli` package as described in “Setup Environment” section above.

Note that these tests make real requests to google analytics APIs and should be run sparingly to avoid being rate limited in our live apps which use the same account credentials.

```
1 # Run cucumber integration tests
2 dotenv -e .env npm run cucumber
3
4 # Run cucumber integration tests with node debugging enabled
5 dotenv -e .env npm run cucumber:debug
```

The cucumber features and support files can be found in the `features` directory

Running the application as a npm package

- To run the utility on your computer, install it through npm:

```
1 npm install -g analytics-reporter
```

Running the application locally

To run the application locally with database reporting, you'll need a postgres database running on port 5432. There is a docker-compose file provided in the repo so that you can start an empty database with the command:

```
1 docker-compose up
```

Setup environment See “Configuration and Google Analytics Setup” below for the required environment variables and other setup for Google Analytics auth.

It may be easiest to use the `dotenv-cli` package to configure the environment for the application.

Create a `.env` file using `env.example` as a template, with the correct credentials and other config values. This file is ignored in the `.gitignore` file and should not be checked in to the repository.

Run the application

```
1 # running the app with no config
2 npm start
3
4 # running the app with dotenv-cli
5 dotenv -e .env npm start
```

Configuration and Google Analytics Setup

- Enable Google Analytics API for your project in the Google developer dashboard.

-
- Create a service account for API access in the Google developer dashboard.
 - Go to the “KEYS” tab for your service account, create new key using “ADD KEY” button, and download the **JSON** private key file it gives you.
 - Grab the generated client email address (ends with `gserviceaccount.com`) from the contents of the `.json` file.
 - Grant that email address `Read`, `Analyze` & `Collaborate` permissions on the Google Analytics profile(s) whose data you wish to publish.
 - Set environment variables for `analytics-reporter`. It needs email address of service account, and view ID in the profile you authorized it to:

```
1 export ANALYTICS_REPORT_EMAIL="YYYYYYY@developer.gserviceaccount.com"
2 export ANALYTICS_REPORT_IDS="XXXXXX"
```

You may wish to manage these using `autoenv`. If you do, there is an `example.env` file you can copy to `.env` to get started.

To find your Google Analytics view ID:

1. Sign in to your Analytics account.
 2. Select the Admin tab.
 3. Select an account from the dropdown in the ACCOUNT column.
 4. Select a property from the dropdown in the PROPERTY column.
 5. Select a view from the dropdown in the VIEW column.
 6. Click “View Settings”
 7. Copy the view ID. You’ll need to enter it with `ga:` as a prefix.
- You can specify your private key through environment variables either as a file path, or the contents of the key (helpful for Heroku and Heroku-like systems).

To specify a file path (useful in development or Linux server environments):

```
1 export ANALYTICS_KEY_PATH="/path/to/secret_key.json"
```

Alternatively, to specify the key directly (useful in a PaaS environment), paste in the contents of the JSON file’s `private_key` field **directly and exactly**, in quotes, and **rendering actual line breaks** (not `\n`’s) (below example has been sanitized):

```
1 export ANALYTICS_KEY="-----BEGIN PRIVATE KEY-----
2 [contents of key]
3 -----END PRIVATE KEY-----
4 "
```

If you have multiple accounts for a profile, you can set the `ANALYTICS_CREDENTIALS` variable with a JSON encoded array of those credentials and they'll be used to authorize API requests in a round-robin style.

```
1 export ANALYTICS_CREDENTIALS='[
2   {
3     "key": "-----BEGIN PRIVATE KEY-----\n[contents of key]\n-----END
        PRIVATE KEY-----",
4     "email": "email_1@example.com"
5   },
6   {
7     "key": "-----BEGIN PRIVATE KEY-----\n[contents of key]\n-----END
        PRIVATE KEY-----",
8     "email": "email_2@example.com"
9   }
10 ]'
```

- Make sure your computer or server is syncing its time with the world over NTP. Your computer's time will need to match those on Google's servers for the authentication to work.
- Test your configuration by printing a report to STDOUT:

```
1 ./bin/analytics --only users
```

If you see a nicely formatted JSON file, you are all set.

- (Optional) Authorize yourself for S3 publishing.

If you plan to use this project's lightweight S3 publishing system, you'll need to add 6 more environment variables:

```
1 export AWS_REGION=us-east-1
2 export AWS_ACCESS_KEY_ID=[your-key]
3 export AWS_SECRET_ACCESS_KEY=[your-secret-key]
4
5 export AWS_BUCKET=[your-bucket]
6 export AWS_BUCKET_PATH=[your-path]
7 export AWS_CACHE_TIME=0
```

There are cases where you want to use a custom object storage server compatible with Amazon S3 APIs, like minio, in that specific case you should set an extra env variable:

```
1 export AWS_S3_ENDPOINT=http://your-storage-server:port
```

Other configuration

If you use a **single domain** for all of your analytics data, then your profile is likely set to return relative paths (e.g. `/faq`) and not absolute paths when accessing real-time reports.

You can set a default domain, to be returned as data in all real-time data point:

```
1 export ANALYTICS_HOSTNAME=https://konklone.com
```

This will produce points similar to the following:

```
1 {
2   "page": "/post/why-google-is-hurrying-the-web-to-kill-sha-1",
3   "page_title": "Why Google is Hurrying the Web to Kill SHA-1",
4   "active_visitors": "1",
5   "domain": "https://konklone.com"
6 }
```

Use

Reports are created and published using `npm start` or `./bin/analytics`

```
1 # using npm scripts
2 npm start
3
4 # running the app directly
5 ./bin/analytics
```

This will run every report, in sequence, and print out the resulting JSON to STDOUT.

A report might look something like this:

```
1 {
2   "name": "devices",
3   "frequency": "daily",
4   "slim": true,
5   "query": {
6     "dimensions": [
7       {
8         "name": "date"
9       },
10      {
11        "name": "deviceCategory"
12      }
13    ],
14    "metrics": [
15      {
16        "name": "sessions"
```

```
17     }
18   ],
19   "dateRanges": [
20     {
21       "startDate": "30daysAgo",
22       "endDate": "yesterday"
23     }
24   ],
25   "orderBys": [
26     {
27       "dimension": {
28         "dimensionName": "date"
29       },
30       "desc": true
31     }
32   ]
33 },
34 "meta": {
35   "name": "Devices",
36   "description": "30 days of desktop/mobile/tablet visits for all
37   sites."
38 }
39 "data": [
40   {
41     "date": "2023-12-25",
42     "device": "mobile",
43     "visits": "13681896"
44   },
45   {
46     "date": "2023-12-25",
47     "device": "desktop",
48     "visits": "5775002"
49   },
50   {
51     "date": "2023-12-25",
52     "device": "tablet",
53     "visits": "367039"
54   },
55   ...
56 ],
57 "totals": {
58   "visits": 3584551745,
59   "devices": {
60     "mobile": 2012722956,
61     "desktop": 1513968883,
62     "tablet": 52313579,
63     "smart tv": 5546327
64   }
65 },
66 "taken_at": "2023-12-26T20:52:50.062Z"
67 }
```

Options

- `--output` - write the report result to a provided directory. Report files will be named with the name in the report configuration.

```
1 ./bin/analytics --output /path/to/data
```

- `--publish` - Publish to an S3 bucket. Requires AWS environment variables set as described above.

```
1 ./bin/analytics --publish
```

- `--write-to-database` - write data to a database. Requires a postgres configuration to be set in environment variables as described below.
- `--only` - only run one or more specific reports. Multiple reports are comma separated.

```
1 ./bin/analytics --only devices
2 ./bin/analytics --only devices,today
```

- `--slim` - Where supported, use totals only (omit the `data` array). Only applies to JSON, and reports where `"slim": true`.

```
1 ./bin/analytics --only devices --slim
```

- `--csv` - Formats reports as CSV instead of the default JSON format.

```
1 ./bin/analytics --csv
```

- `--frequency` - Run only reports with 'frequency' value matching the provided value.

```
1 ./bin/analytics --frequency=realtime
```

- `--debug` - Print debug details on STDOUT.

```
1 ./bin/analytics --publish --debug
```

Saving data to postgres

The analytics reporter can write data it pulls from Google Analytics to a Postgres database. The postgres configuration can be set using environment variables:

```
1 export POSTGRES_HOST = "my.db.host.com"
2 export POSTGRES_USER = "postgres"
3 export POSTGRES_PASSWORD = "123abc"
4 export POSTGRES_DATABASE = "analytics"
```

The database expects a particular schema which will be described in the API server that consumes and publishes this data.

To write reports to a database, use the `--write-to-database` option when starting the reporter.

Upgrading from Universal Analytics

Background

This project previously acquired data from Google Analytics V3, also known as Universal Analytics (UA).

Google is retiring UA and is encouraging users to move to their new version Google Analytics V4 (GA4). UA will be deprecated on July 1st 2024.

Migration details

Some data points have been removed or added by Google as part of the move to GA4.

Deprecated fields

- browser_version
- has_social_referral
- exits
- exit_page

New fields

bounce_rate The percentage of sessions that were not engaged. GA4 defines engaged as a session that lasts longer than 10 seconds or has multiple pageviews.

file_name The page path of a downloaded file.

language_code The ISO639 language setting of the user's device. e.g. 'en-us'

session_default_channel_group An enum which describes the session. Possible values:

'Direct', 'Organic Search', 'Paid Social', 'Organic Social', 'Email', 'Affiliates', 'Referral', 'Paid Search', 'Video', and 'Display'

Deploying to Cloud.gov

The analytics reporter runs on :cloud:.gov. Please refer to the `manifest.yml` file at the root of the repository for application information.

Ensure you're targeting the proper `org` and `space`.

```
1 cf target
```

Deploy the application with the following command.

```
1 cf push -f manifest.yml
```

Set the environmental variables based on local `.env` file.

```
1 cf set-env analytics-reporter AWS_ACCESS_KEY_ID 123abc
2 cf set-env analytics-reporter AWS_SECRET_ACCESS_KEY 456def
3 # ...
```

Restage the application to use the environment variables.

```
1 cf restage analytics-reporter
```

Public domain

This project is in the worldwide public domain. As stated in CONTRIBUTING:

This project is in the public domain within the United States, and copyright and related rights in the work worldwide are waived through the CC0 1.0 Universal public domain dedication.

All contributions to this project will be released under the CC0 dedication. By submitting a pull request, you are agreeing to comply with this waiver of copyright interest.