Marathon Documentation

Release 3.1.11

Top Free Games

Feb 23, 2023

Contents

Contents:

CHAPTER 1

Overview

What is Marathon? Marathon is a push notification platform that makes it very easy to send massive push notifications to tens of millions of users for several different apps.

1.1 Features

- **Multi-tenant** - Marathon already works for as many apps as you need, just keep adding new ones;
- **Multi-services** - Marathon supports both gcm and apns services, but plugging a new one shouldn't be difficult;
- **Massive Push Notification** - Send tens of millions of push notifications and keep track of job status;
- **New Relic Support** - Natively support new relic with segments in each API route for easy detection of bottlenecks;
- **Sendgrid Support** - Natively support sendgrid and send emails when jobs are created, scheduled, paused or enter circuit break;
- **Easy to deploy** - Marathon comes with containers already exported to docker hub for every single of our successful builds. Just pick your choice!

1.2 Architecture

Marathon is based on some premises:

- You have a system that reads kafka topics and send pushs to apns and gcms services;
- You have a PostgreSQL Database with user tables containing at least the following information:
  - user_id: the identification of an user, it can be repeated for different tokens (e.g. an user with several devices);
  - token: the device token registered in apns or gcm service;
  - locale: the language of the device (ex: en, fr, pt)

- region: the region of the device (ex: US, FR, BR)
- tz: the timezone of the device (ex: -0400, -0300, +0100)

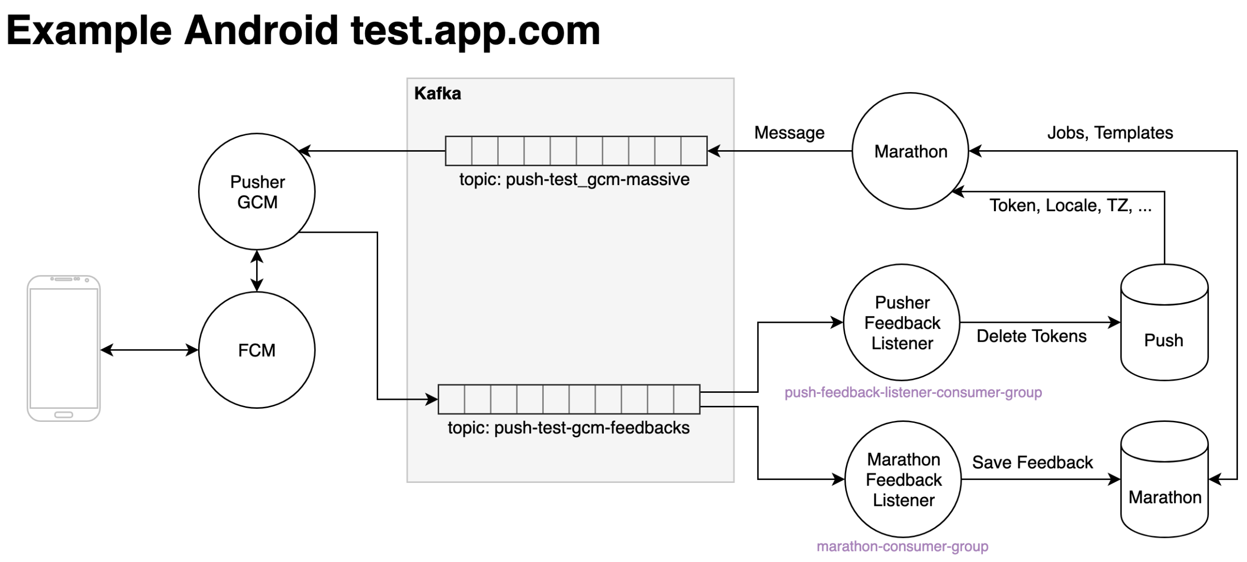
- The apps registered in the Marathon api already have created user tables (in the previous PostgreSQL Database) and Kafka topics for apns and gcm services;

Marathon is composed of three main modules:

- An API responsible for CRUD of apps, templates and jobs;
- A worker composed of several sub-workers:
  - A sub-worker responsible for getting the information from the database, building the message and sending to Kafka.
  - A sub-worker responsible for splitting a CSV and creating small devices ID batches.
  - A sub-worker responsible for scheduling push notifiction for users in a CSV file using each user timezone;
  - A sub-worker responsible for building a message given a template and a context and sending it to the app and service corresponding kafka topic;
- A feedback listener responsible for processing feedbacks of the notifications sent to apns or gcm;

1.3 Marathon and Pusher

Marathon can be used with Pusher. In the following image, you can see a full example of the integration of marathon and pusher. Note that this example doesn't include any routes to send individual messages or to register device tokens (this information must be present the push database, as said in the premises section).



example

1.4 The Stack

For the devs out there, our code is in Go, but more specifically:

- Web Framework - Echo;
- Database - Postgres >= 9.5;

- Workers - go-workers using Redis.
- Kafka and Zookeper;

1.5 Who's Using it

Well, right now, only us at TFG Co, are using it, but it would be great to get a community around the project. Hope to hear from you guys soon!

1.6 How To Contribute?

Just the usual: Fork, Hack, Pull Request. Rinse and Repeat. Also don't forget to include tests and docs (we are very fond of both).

CHAPTER 2

Hosting Marathon

There are two ways to host Marathon: docker or from source.

2.1 Docker

Running Marathon with docker is rather simple. Our docker container image comes bundled with the API binary. All you need to do is load balance all the containers and you're good to go. The API runs at port `8080` in the docker image.

Marathon uses PostgreSQL to store jobs information. The container takes environment variables to specify this connection:

- `MARATHON_DB_HOST` - PostgreSQL host to connect to;
- `MARATHON_DB_PORT` - PostgreSQL port to connect to;
- `MARATHON_DB_DATABASE` - PostgreSQL database to connect to;
- `MARATHON_DB_USER` - Password of the PostgreSQL Server to connect to;

Marathon also uses another PostgreSQL database to read tokens information. The container takes environment variables to specify this connection:

- `MARATHON_PUSH_DB_HOST` - PostgreSQL host to connect to;
- `MARATHON_PUSH_DB_PORT` - PostgreSQL port to connect to;
- `MARATHON_PUSH_DB_DATABASE` - PostgreSQL database to connect to;
- `MARATHON_PUSH_DB_USER` - Password of the PostgreSQL Server to connect to;

For uploading and reading CSV files Marathon uses AWS S3, so you'll need to specify the following environment variables as well:

- `MARATHON_S3_BUCKET` - AWS S3 bucket containing the csv files;
- `MARATHON_S3_FOLDER` - AWS S3 folder containing the csv files;

- MARATHON_S3_ACCESSKEY - AWS S3 access key;
- MARATHON_S3_SECRETACCESSKEY - AWS S3 secret;
- MARATHON_S3_CONTROLGROUPFOLDER - AWS S3 folder containing the control group csv files;

The workers use redis for queueing:

- MARATHON_WORKERS_REDIS_HOST - Redis host to connect to;
- MARATHON_WORKERS_REDIS_PORT - Redis port to connect to;
- MARATHON_WORKERS_REDIS_PASS - Password of the redis server to connect to;

Marathon uses kafka to send push notifications:

- MARATHON_KAFKA_BOOTSTRAPSERVERS - Kafka servers to connect to (comma separated, without spaces);

The workers need a template for sending push notifications:

- MARATHON_WORKERS_TOPICTEMPLATE - Kafka topic template;

Finally, the feedback listener uses kafka for receiving the push notifications' feedbacks from APNS or GCM:

- MARATHON_FEEDBACKLISTENER_KAFKA_BROKERS - Kafka brokers to connect to (comma separated, without spaces);
- MARATHON_FEEDBACKLISTENER_KAFKA_TOPICS - Array of kafka topics to read from;
- MARATHON_FEEDBACKLISTENER_KAFKA_GROUP - Kafka consumer group;
- MARATHON_FEEDBACKLISTENER_FLUSHINTERVAL - Interval during which the feedback listener caches the feedbacks metrics before updating the job feedbacks in PostgreSQL;

Other than that, there are a couple more configurations you can pass using environment variables:

- MARATHON_NEWRELIC_KEY - If you have a New Relic account, you can use this variable to specify your API Key to populate data with New Relic API;
- MARATHON_SENTRY_URL - If you have a sentry server you can use this variable to specify your project's URL to send errors to.
- MARATHON_SENDGRID_KEY - If you have a sendgrid account, you can use this variable to specify your API Key for sending emails when jobs are created, scheduled, paused or enter circuit break;

2.1.1 Example command for running with Docker

```
   $ docker pull tfgco/marathon
   $ docker run -t --rm -e "MARATHON_POSTGRES_HOST=<postgres host>" -e "MARATHON_
↪POSTGRES_PORT=<postgres port>" -p 8080:8080 tfgco/marathon
```

2.2 Source

Left as an exercise to the reader.

CHAPTER 3

Marathon API

Every request other than GET /healthcheck must pass a `x-forwarded-email` header, otherwise it will return 401 Unauthorized.

3.1 Healthcheck Routes

3.1.1 Healthcheck

`GET /healthcheck`

Validates that the app is still up, including the database connection.

- Success Response
  - Code: `200`
  - Content:

    ```
    {
      "healthy": true
    }
    ```

  - Headers:

    It will add an `MARATHON-VERSION` header with the current marathon module version.
- Error Response

  It will return an error if it failed to connect to the database.
  - Code: `500`
  - Content:

```
{
  "healthy": false
}
```

3.2 App Routes

3.2.1 List Apps

`GET /apps`

List all apps in Marathon DB.

- Success Response
  - Code: `200`
  - Content:

```
[
  {
    id:        [uuid],
    name:      [string],
    bundleId:  [string],
    createdBy: [string], // email
    createdAt: [int64],  // nanoseconds since epoch
    updatedAt: [int64]   // nanoseconds since epoch
  },
  {
    id:        [uuid],
    name:      [string],
    bundleId:  [string],
    createdBy: [string], // email
    createdAt: [int64],  // nanoseconds since epoch
    updatedAt: [int64]   // nanoseconds since epoch
  },
  ...
]
```

- Error Response

  It will return an error if no `x-forwarded-email` header is specified
  - Code: `401`
  - Code: `500`
  - Content:

```
{
  "reason": [string]
}
```

3.2.2 Create App

`POST /apps`

Creates a new app with the given parameters.

- Payload

```
{
  "name":                            [string],   // 255 characters max
  "bundleId":                        [string]    // matching ^[a-z0-9]+\\.[a-z0-
↪9]+(\\.[a-z0-9]+)+$
}
```

- Success Response
  - Code: `201`
  - Content:

```
{
  id:         [uuid],    // generated by marathon
  name:       [string],
  bundleId:   [string],
  createdBy:  [string],  // email of the authenticated user
  createdAt:  [int64],   // nanoseconds since epoch
  updatedAt:  [int64]    // nanoseconds since epoch
}
```

- Error Response

  It will return an error if no `x-forwarded-email` header is specified

  - Code: `401`

  It will return an error if there is already an app with the same bundleId.

  - Code: `409`
  - Content:

```
{
  "reason": [string]
}
```

  It will return an error if there are missing or invalid parameters.

  - Code: `422`
  - Content:

```
{
  "reason": [string]
}
```

  - Code: `500`
  - Content:

```
{
  "reason": [string]
}
```

3.2.3 Retrieve App

`GET /apps/:appId`

Retrieves the app that has id `appId`.

- Success Response
  - Code: `200`
  - Content:

```
{
  id:        [uuid],
  name:      [string],
  bundleId:  [string],
  createdBy: [string], // email
  createdAt: [int64],  // nanoseconds since epoch
  updatedAt: [int64]   // nanoseconds since epoch
}
```

- Error Response

  It will return an error if no `x-forwarded-email` header is specified

  - Code: `401`
  - Code: `500`
  - Content:

```
{
  "reason": [string]
}
```

3.2.4 Update App

`PUT /apps/:appId`

Updates the app that has id `appId`.

- Payload

```
{
  "name":                             [string],   // 255 characters max
  "bundleId":                         [string]    // matching ^[a-z0-9]+\\.[a-z0-
↪9]+(\\.[a-z0-9]+)+$
}
```

- Success Response
  - Code: `201`
  - Content:

```
{
  id:        [uuid],    // generated by marathon
  name:      [string],
  bundleId:  [string],
  createdBy: [string], // email of the authenticated user
  createdAt: [int64],  // nanoseconds since epoch
  updatedAt: [int64]   // nanoseconds since epoch
}
```

- Error Response

  It will return an error if no `x-forwarded-email` header is specified

  - Code: `401`

  It will return an error if there is already an app with the same bundleId.

  - Code: `409`
  - Content:

  ```
  {
    "reason": [string]
  }
  ```

  It will return an error if there are missing or invalid parameters.

  - Code: `422`
  - Content:

  ```
  {
    "reason": [string]
  }
  ```

  - Code: `500`
  - Content:

  ```
  {
    "reason": [string]
  }
  ```

3.2.5 Delete App

`DELETE /apps/:appId`

Deletes the app that has id `appId`.

- Success Response

  - Code: `204`

- Error Response

  It will return an error if no `x-forwarded-email` header is specified

  - Code: `401`

  It will return an error if there are missing or invalid parameters.

  - Code: `422`
  - Content:

  ```
  {
    "reason": [string]
  }
  ```

  - Code: `500`
  - Content:

```
{
  "reason": [string]
}
```

3.3 Template Routes

3.3.1 List app templates

`GET /apps/:appId/templates`

List all templates for the app with the given id.

- Success Response
  - Code: `200`
  - Content:

```
[
  {
    id:        [uuid],
    name:      [string],
    locale:    [string],
    defaults:  [json],
    body:      [json],
    appId:     [uuid],
    createdBy: [string], // email
    createdAt: [int64],  // nanoseconds since epoch
    updatedAt: [int64]   // nanoseconds since epoch
  },
  {
    id:        [uuid],
    name:      [string],
    locale:    [string],
    defaults:  [json],
    body:      [json],
    appId:     [uuid],
    createdBy: [string], // email
    createdAt: [int64],  // nanoseconds since epoch
    updatedAt: [int64]   // nanoseconds since epoch
  },
  ...
]
```

- Error Response

  It will return an error if no `x-forwarded-email` header is specified

  - Code: `401`
  - Code: `500`
  - Content:

```
{
  "reason": [string]
}
```

3.3.2 Create Template

`POST /apps/:appId/templates`

Creates a new template with the given parameters.

- Payload

```
{
  name:      [string],
  locale:    [string],
  defaults:  [json],   // cannot be empty
  body:      [json]   // cannot be empty
}
```

- Success Response
  - Code: `201`
  - Content:

```
{
  id:        [uuid],
  name:      [string],
  locale:    [string],
  defaults:  [json],   // cannot be empty
  body:      [json],   // cannot be empty
  appId:     [uuid],
  createdBy: [string], // email
  createdAt: [int64],  // nanoseconds since epoch
  updatedAt: [int64]   // nanoseconds since epoch
}
```

- Error Response

  It will return an error if no `x-forwarded-email` header is specified

  - Code: `401`

  It will return an error if there is already a template with the same appId, name and locale.

  - Code: `409`
  - Content:

```
{
  "reason": [string]
}
```

  It will return an error if there are missing or invalid parameters.

  - Code: `422`
  - Content:

```
{
  "reason": [string]
}
```

  - Code: `500`
  - Content:

```
{
  "reason": [string]
}
```

3.3.3 Retrieve Template

`GET /apps/:appId/templates/:templateId`

Retrieves the app that has id `templateId`.

- Success Response
  - Code: `200`
  - Content:

```
{
  id:         [uuid],
  name:       [string],
  locale:     [string],
  defaults:   [json],
  body:       [json],
  appId:      [uuid],
  createdBy:  [string]
  createdAt:  [int64],
  updatedAt:  [int64]
}
```

- Error Response

  It will return an error if no `x-forwarded-email` header is specified

  - Code: `401`
  - Code: `500`
  - Content:

```
{
  "reason": [string]
}
```

3.3.4 Update Template

`PUT /apps/:appId/templates/:templateId`

Updates the template that has id `templateId`.

- Payload

```
{
  name:       [string],
  locale:     [string],
  defaults:   [json],    // cannot be empty
  body:       [json]   // cannot be empty
}
```

- Success Response

  - Code: `201`
  - Content:

```
{
  id:        [uuid],
  name:      [string],
  locale:    [string],
  defaults:  [json],
  body:      [json],
  appId:     [uuid],
  createdBy: [string],
  createdAt: [int64],
  updatedAt: [int64]
}
```

- Error Response

  It will return an error if no `x-forwarded-email` header is specified

  - Code: `401`

  It will return an error if there is already a template with the same appId, name and locale.

  - Code: `409`
  - Content:

```
{
  "reason": [string]
}
```

  It will return an error if there are missing or invalid parameters.

  - Code: `422`
  - Content:

```
{
  "reason": [string]
}
```

  - Code: `500`
  - Content:

```
{
  "reason": [string]
}
```

3.3.5 Delete Template

`DELETE /apps/:appId/templates/:templateId`

Deletes the templaye that has id `templateId`.

- Success Response
  - Code: `204`

- Error Response

  It will return an error if no `x-forwarded-email` header is specified

  - Code: `401`

  It will return an error if there are missing or invalid parameters.

  - Code: `422`
  - Content:

```
{
  "reason": [string]
}
```

  - Code: `500`
  - Content:

```
{
  "reason": [string]
}
```

3.4 Job Routes

3.4.1 List app jobs

`GET /apps/:appId/jobs?template=<optional-template-name>`

List all jobs for the app with the given id. If the `template` query string parameter is sent only jobs for the templates with this name will be returned.

- Success Response
  - Code: `200`
  - Content:

```
[
  {
    id:                    [uuid],
    totalBatches:          [null|int], // if null the total batches that will␣
↪be sent was not calculated yet
    completedBatches:      [int],
    totalUsers:            [null|int], // if null the total users that will␣
↪receive the push was not calculated yet
    totalTokens:           [null|int], // if null the total tokens that will␣
↪receive the push was not calculated yet
    completedTokens:       [int],
    dbPageSize:            [int],     // page size that will be used for␣
↪retrieving tokens from the database
    localized:             [boolean],
    completedAt:           [int64],   // nanoseconds since epoch,
    expiresAt:             [int64],   // nanoseconds since epoch, optional but␣
↪if > 0 push will no longer be sent after this timestamp,
    startsAt:              [int64],   // nanoseconds since epoch, optional but␣
↪if > 0 job was scheduled,
```

(continues on next page)

(continued from previous page)

```
    context:              [json],   // optional
    service:              [gcm|apns],
    filters:              [json],   // optional
    metadata:             [json],   // optional
    csvPath:              [string], // full path of the S3 file with the csv␣
↪containing users ids for this job,
    templateName:         [string], // can also be several strings separated␣
↪by commas
    pastTimeStrategy:     [null|string], // null if job is not localized or␣
↪one of [skip, nextDay]
    status:               [null|string], // null if job is running or one of␣
↪[paused, stopped, circuitbreak]
    appId:                [uuid],
    createdBy:            [string], // email
    createdAt:            [int64],  // nanoseconds since epoch
    updatedAt:            [int64],  // nanoseconds since epoch
    controlGroup:         [float],  // float between 0-1, represents the % of␣
↪users that won't receive notifications
    controlGroupCsvPath:  [string]  // full path of the S3 file with the csv␣
↪containing users ids of users in the control group
  },
  {
    id:                   [uuid],
    totalBatches:         [null|int],
    completedBatches:     [int],
    totalUsers:           [null|int],
    totalTokens:          [null|int],
    completedTokens:      [int],
    dbPageSize:           [int],
    localized:            [boolean],
    completedAt:          [int64],
    expiresAt:            [int64],
    startsAt:             [int64],
    context:              [json],
    service:              [gcm|apns],
    filters:              [json],
    metadata:             [json],
    csvPath:              [string],
    templateName:         [string],
    pastTimeStrategy:     [null|string],
    status:               [null|string],
    appId:                [uuid],
    createdBy:            [string],
    createdAt:            [int64],
    updatedAt:            [int64],
    controlGroup:         [float],
    controlGroupCsvPath:  [string]
  },
  ...
]
```

- Error Response

  It will return an error if no `x-forwarded-email` header is specified

  - Code: `401`
  - Code: `500`

– Content:

```
{
  "reason": [string]
}
```

3.4.2 Create Job

`POST /apps/:appId/jobs?template=<mandatory-template-name>`

Creates a new job with the given parameters and template name. The template name can be composed of several template names separated by commas. Example `POST /apps/:appId/jobs?template=tpl1,tpl2,tpl3,tpl4`. In this case the template messages will be randomly chosen for each user using a uniform distribution.

- Payload

```
{
  localized:         [boolean],
  expiresAt:         [int64],  // nanoseconds since epoch, optional but if > 0␣
↪push will no longer be sent after this timestamp,
  startsAt:          [int64],  // nanoseconds since epoch, optional but if > 0 job␣
↪was scheduled,
  context:           [json],   // optional
  service:           [gcm|apns],
  filters:           [json],   // optional
  metadata:          [json],   // optional
  csvPath:           [string], // full path of the S3 file with the csv containing␣
↪users ids for this job,
  pastTimeStrategy: [null|string], // null if job is not localized or one of␣
↪[skip, nextDay]
  controlGroup:      [float]   // float between 0-1, represents the % of users␣
↪that won't receive notifications
}
```

- Success Response
  - Code: `201`
  - Content:

```
{
  id:                  [uuid],
  totalBatches:        [null|int],
  completedBatches:    [int],
  totalUsers:          [null|int],
  completedUsers:      [int],
  completedTokens:     [int],
  dbPageSize:          [int],
  localized:           [boolean],
  completedAt:         [int64],
  expiresAt:           [int64],
  startsAt:            [int64],
  context:             [json],
  service:             [gcm|apns],
  filters:             [json],
  metadata:            [json],
  csvPath:             [string],
  templateName:        [string],
```

(continues on next page)

(continued from previous page)

```
  pastTimeStrategy: [null|string],
  status:           [null|string],
  appId:            [uuid],
  createdBy:        [string],
  createdAt:        [int64],
  updatedAt:        [int64],
  controlGroup:        [float],
  controlGroupCsvPath: [string]
}
```

- Error Response

  It will return an error if no `x-forwarded-email` header is specified

  - Code: `401`

  It will return an error if there are missing or invalid parameters.

  - Code: `422`
  - Content:

```
{
  "reason": [string]
}
```

  - Code: `500`
  - Content:

```
{
  "reason": [string]
}
```

3.4.3 Retrieve Job

`GET /apps/:appId/jobs/:jobId`

Retrieves the app that has id `jobId`.

- Success Response
  - Code: `200`
  - Content:

```
{
  id:               [uuid],
  totalBatches:     [null|int],
  completedBatches: [int],
  totalUsers:       [null|int],
  completedUsers:   [int],
  completedTokens:  [int],
  dbPageSize:       [int],
  localized:        [boolean],
  completedAt:      [int64],
  expiresAt:        [int64],
  startsAt:         [int64],
```

(continues on next page)

(continued from previous page)

```
  context:           [json],
  service:           [gcm|apns],
  filters:           [json],
  metadata:          [json],
  csvPath:           [string],
  templateName:      [string],
  pastTimeStrategy: [null|string],
  status:            [null|string],
  appId:             [uuid],
  createdBy:         [string],
  createdAt:         [int64],
  updatedAt:         [int64],
  controlGroup:         [float],
  controlGroupCsvPath: [string]
}
```

- Error Response

  It will return an error if no `x-forwarded-email` header is specified

  - Code: `401`
  - Code: `500`
  - Content:

```
{
  "reason": [string]
}
```

3.4.4 Pause Job

`PUT /apps/:appId/jobs/:jobId/pause`

Pauses the job that has id `jobId`.

- Payload

```
{}
```

- Success Response
  - Code: `201`
  - Content:

```
{
  id:                [uuid],
  totalBatches:      [null|int],
  completedBatches: [int],
  totalUsers:        [null|int],
  completedUsers:    [int],
  completedTokens:   [int],
  dbPageSize:        [int],
  localized:         [boolean],
  completedAt:       [int64],
  expiresAt:         [int64],
  startsAt:          [int64],
```

(continues on next page)

(continued from previous page)

```
  context:           [json],
  service:           [gcm|apns],
  filters:           [json],
  metadata:          [json],
  csvPath:           [string],
  templateName:      [string],
  pastTimeStrategy: [null|string],
  status:            "paused",
  appId:             [uuid],
  createdBy:         [string],
  createdAt:         [int64],
  updatedAt:         [int64],
  controlGroup:         [float],
  controlGroupCsvPath: [string]
}
```

- Error Response

  It will return an error if no `x-forwarded-email` header is specified

  - Code: `401`

  It will return an error if there are missing or invalid parameters or if the job previous status was not null.

  - Code: `422`
  - Content:

```
{
  "reason": [string]
}
```

  - Code: `500`
  - Content:

```
{
  "reason": [string]
}
```

3.4.5 Stop Job

`PUT /apps/:appId/jobs/:jobId/pause`

Stops the job that has id `jobId`.

- Payload

```
{}
```

- Success Response
  - Code: `201`
  - Content:

```
{
  id:                [uuid],
```

(continues on next page)

(continued from previous page)

```
  totalBatches:      [null|int],
  completedBatches: [int],
  totalUsers:        [null|int],
  completedUsers:    [int],
  completedTokens:   [int],
  dbPageSize:        [int],
  localized:         [boolean],
  completedAt:       [int64],
  expiresAt:         [int64],
  startsAt:          [int64],
  context:           [json],
  service:           [gcm|apns],
  filters:           [json],
  metadata:          [json],
  csvPath:           [string],
  templateName:      [string],
  pastTimeStrategy: [null|string],
  status:            "stopped",
  appId:             [uuid],
  createdBy:         [string],
  createdAt:         [int64],
  updatedAt:         [int64],
  controlGroup:         [float],
  controlGroupCsvPath: [string]
}
```

- Error Response

  It will return an error if no `x-forwarded-email` header is specified

  – Code: `401`

  It will return an error if there are missing or invalid parameters.

  – Code: `422`

  – Content:

```
{
  "reason": [string]
}
```

  – Code: `500`

  – Content:

```
{
  "reason": [string]
}
```

3.4.6 Resume Job

`PUT /apps/:appId/jobs/:jobId/resume`

Resumes the job that has id `jobId`.

- Payload

```
{}
```

- Success Response
  - Code: `201`
  - Content:

```
{
  id:                 [uuid],
  totalBatches:       [null|int],
  completedBatches: [int],
  totalUsers:         [null|int],
  completedUsers:    [int],
  completedTokens:   [int],
  dbPageSize:         [int],
  localized:          [boolean],
  completedAt:        [int64],
  expiresAt:          [int64],
  startsAt:           [int64],
  context:            [json],
  service:            [gcm|apns],
  filters:            [json],
  metadata:           [json],
  csvPath:            [string],
  templateName:       [string],
  pastTimeStrategy: [null|string],
  status:             null,
  appId:              [uuid],
  createdBy:          [string],
  createdAt:          [int64],
  updatedAt:          [int64],
  controlGroup:          [float],
  controlGroupCsvPath: [string]
}
```

- Error Response

  It will return an error if no `x-forwarded-email` header is specified

  - Code: `401`

  It will return an error if there are missing or invalid parameters or if the job previous status was not [paused|circuitbreak].

  - Code: `422`
  - Content:

```
{
  "reason": [string]
}
```

  - Code: `500`
  - Content:

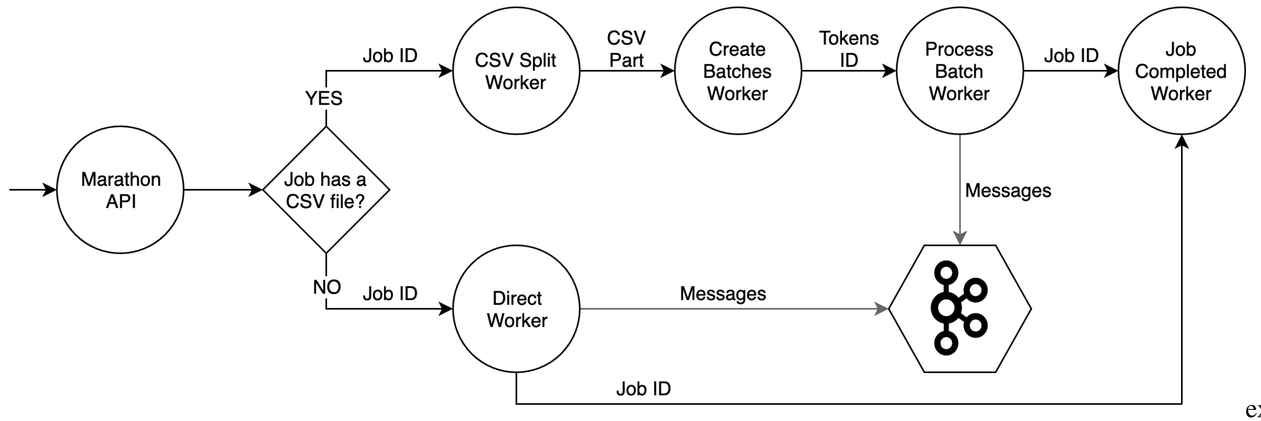
```
{
  "reason": [string]
}
```

CHAPTER 4

Marathon Workers

4.1 Overview

Depending on the job type, different workers can be called. See the image below:



example

This image shows the workers execution order. Also, you can see each data that go between the workers.

When we have a scheduled job with multiple timezones, multiple jobs will be created and scheduled. Each job will be processed within 1 hour intervals, for example, if a user is at the +1215 timezone he will receive a message when the +1200 is processed.

4.2 Workers

4.2.1 Direct Worker

The API is responsible to create batches using an estimative of the database size. This worker process these batches. It will query the PUSH_DB using the job filters, creates the messages and send to Kafka. This worker is really fast and can handle big amount of tokens (tested with 1.5x10^8 tokens).

If a control group is set, it will be saved on Redis. The completed job worker will pull this data and create a CSV with the control group ids.

It will not generate a CSV of the sent messages.

This worker will produce two metrics:

- `starting_direct_part`: represents when the worker starts;
- `get_from_pg` represent the spent time on retrieving data from the database.

After processing all parts, the `Job Completed Worker` is called. To know if all batches are completed, a counter is the Redis is used.

4.2.2 CSV Split Worker

This worker downloads a CSV file from AWS S3, reads it size and splits it in small batches. **Only one worker will do this job**.

After creating each batch, it will send `csv_job_part` metric.

4.2.3 Create Batches Worker

This worker downloads a part of the CSV file from AWS S3, reads it and creates batches of user information (locale, token, tz).

4.2.4 Process Batch Worker

This worker receives a batch of user information (locale and token), builds the template for each user using the locale information and the job template name and send the message to the Kafka topic corresponding to the job app and service. If the error rate is more than a threshold this job enters circuit break state. When the job is paused or in circuit break, the batches are stored in a paused job list in Redis with an expiration of one week.

When this work starts, it sends the `starting_process_batch_worker` metric.

The metric send_message_return is reported when messages are delivered to Kafka, either successfully or with errors.

The batches are defined by a position in the CSV and the number of bytes to read from that position. Because of that, during the batches creation, some IDs can be divided and have it beginning in one batch and the end in other. To recovery this IDs, each process batch worker will search for the first `\n` in the file and will considerate all the bytes before this marker has part of one split ID. The method is similar at the end of the file, it will considerate all bytes since the las `\n` as part of another split ID. It will save this information in the Redis and after all, batches are processed, the last worker will retrieve this information, join the splits IDs and send the messages.

To know if all batches are completed, a counter is the Redis is used.

4.2.5 Job Completed Worker

When all `Process Batch Workers` or all `Direct Workers` is completed, they will call this worker.

This worker will send one email saying the job is completed.

Also, it reads the Redis and creates the control group CSV.

4.2.6 Resume Job Worker

This worker handles jobs that are paused or in circuit break state. It removes a batch from the paused job list and calls the process batch worker for each one of them until are has no more paused batches.

4.2.7 Metrics

Each one of the workers has metrics indicating the start (e.g. `starting_create_batches_worker`), the completion (e.g. `completed_create_batches_worker`) and possible execution errors (e.g. `error_create_batches_worker`). The error metrics are only generated for internal errors or input validation errors, for instance when no valid IDs are present in the CSV file. In every other case, the workers will generate the completed metric.

CHAPTER 5

Marathon Feedbacks

After a message is sent to the kafka topic corresponding to the job app and service, another system will attempt to send the push notification to APNS or GCM and write a feedback of this successful or failed push notification in another queue, the feedback kafka.

Marathon has another command, besides the API and the workers, that starts a feedback listener that reads from this kafka's topics and update the job's feedback column in the PostgreSQL database. The messages in this queue contain all metadata sent by marathon including the job id.

5.1 Feedbacks column

The feedbacks column contains a JSON in the following format:

```
{
  "ack":        <int>,  // count
  "error-key1": <int>,  // count
  "error-key2": <int>   // count
  ...
}
```

In the case of successful push notifications the key is `ack`. For failed push notifications the key will be the error reason received from APNS or GCM, for example `BAD_REGISTRATION`, `unregistered`, etc.

To avoid updating the job entry in the PostgreSQL database for every message received in the feedbacks kafka, we update the database periodically (defaults to every 5 seconds) by using a local cache to store all feedbacks received in the mean time.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search