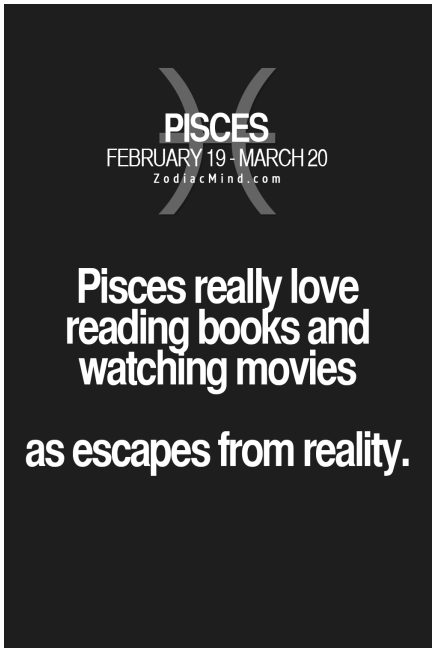
General Description

What we have today
Today we have a horoscopes api service which is hosted on Heroku, and using AWS API Gateway as gateway. The horoscopes api has 1 endpoint: **GET /v1/{date}** which returns all horoscopes for the requested date (all signs x standard / love / career / weekly / monthly).

What we want to add
We want to add an additional endpoint to the existing API for zodiac facts and traits: **GET /v1/facts?sign={sign}&factnumber={factnumber}**.
This endpoint returns an image which described a random fact about **{sign}**. For example, **GET /v1/facts?sign=pisces&factnumber=1** will return a url (AWS S3) to this image:



All Project Parts

Excel doc with raw data
An excel doc with the following rows will be sent to you:

Sign	Line1	Line2	SerialNumber

Explanation about each column meaning
SerialNumber - This is the serial number of the sentence per sign (counted separately per sign, so there will be 90 lines with Aries as the sign and rows with SerialNumbers 1 until 90. For Taurus there should also be 90 rows with SerialNumber 1 until 90, same logic applies for all signs).
Line1 + **Line2** - If the sentence is short, you should put it inside Line1 and leave Line2 empty. If the sentence is long you should split it into 2 lines in a way that it makes sense. For example, right before a "but" in the sentence... The reason is because it is going to be turned into an image with 2 sections of text one above the other (Line1 above Line2).
Sign - just type in the sign name, please make sure it is always lowercase and that you don't misspell the zodiac sign

Our writers will fill this in with relevant data. This should be loaded into our API's database.

Image Generator
Today we have a PHP based image generator which you will use. It received Sign, Line1, Line2 parameters and generated an image with these texts and sign:

```
let imageUrl =
`http://www.xxxxxxxx.com/imagemakersign.php?sign=${signName}&text1=${preRenderedData.TextLine1}&text2=${preRenderedData.TextLine2}&${renderer.getRandomInt(1000000, 9999999).toString()}`;
```

getRandomInt here is meant to avoid Facebook's caching system per URL (we might be posting the image on facebook). It should be built this way incase we update the design and want to regenerate a certain image. More details on this further down the document.

Horoscopes API
We are hosting a NodeJS horoscopes API in Heroku. It is using Postgres database with Sequelize data access layer.

Here you will need to add a few features (more details about these points appear further down the document):

1. New endpoint according to the structure described in the **General Description**
2. New table to hold all the data and url's of the zodiac fact images
3. A function to generate all images, save them in an AWS S3 bucket and link the corresponding table row to that specific image url.

AWS API Gateway
An API gateway serves as a proxy and volume control for all API requests.
Here you will be required to add an additional endpoint for the new route in **Horoscopes API** Heroku code.

AWS S3 Bucket
Our existing S3 bucket will hold all the images which the Horoscopes API points to.
The images should be uploaded to and held in /generated-zodiac-facts folder:

More details about required changes in Horoscopes API Heroku App

1. You will need to create a new endpoint both on the Heroku API project and on the AWS API Gateway, and eventually test it end to end with the public URL. This way you can make sure it goes through the AWS API Gateway like it should:
   https://api.xxxxxxxx.com/v1/facts?sign={...}&factnumber={...}

2. You will need to add a new Table to the Heroku Horoscopes API project. This table should be named **ZodiacSignFacts** should be in the following structure (open for discussion):

   id - unique row id
   createdOn - the datetime when row was created
   generatedOn - nullable, it holds the timestamp when imageUrl was generated
   sign - the zodiac sign of the fact represented in this row
   imageUrl - nullable, the generated image Url (in our AWS S3 bucket)
   line1 - holds the original line from the excel, make sure to escape all required characters
   line2 - nullable, holds the second line if such exists (Line2 is optional in the excel)
   factNumber - the number of the fact per sign (we will have 90 per each sign for starters)

   The entire excel we receive from our writers should be loaded to this table before we have an imageUrl.

   Make sure to use Sequelize model definitions when creating this table.

3. Every time a fact image URL is returned from the Horoscopes API /fact endpoint, it should have a random param added to it as in this line of code:
   `{imageUrl}?getRandomInt(1000000, 9999999).toString()`
   The reason is to avoid caching of the image. Here is the full function used:

```
static getRandomInt(min, max) {
        let ceilMin = Math.ceil(min);
        let floorMax = Math.floor(max);
        return Math.floor(Math.random() * (floorMax - ceilMin)) + ceilMin;
    }
```

4. Once all the data from the excel is loaded to **ZodiacSignFacts** Table, there should be a .js script which generated all the images by calling our Image Generator url with the relevant parameters and saving the output image in the S3 bucket. In addition, all relevant columns in the table should be updated when the image is saved to S3. This script should run only over rows which have **imageUrl is Null = true** unless it is called

with **-override** parameter. In this case, it will run over all rows, regenerate the images and replace existing **imageUrl**s in the table.