Interstellar backend coding challenge

In geospatial information systems, a "granule" is an image of a particular location at a particular point in time.

Each granule includes information from a wide spectrum of light bands, but the instruments on the satellite that capture this data do so as separate images, one for each band. Typically each band is provided as a single `.jp2` file (JPEG 2000 format (https://en.wikipedia.org/wiki/JPEG_2000)).

Your company, Future Construction GmbH, would like to start using satellite data to perform advanced location scouting for new infrastructure projects. They have decided to start with the publicly available data from the Sentinel-2 dataset ((https://cloud.google.com/storage/docs/public-datasets/sentinel-2).).

The team lead at
Future Construction has asked you to build a prototype service to fetch and process this satellite data, with the goal of the initial spike being to turn it into standard RGB images that a human surveyor can look over.

Even though this is just a prototype, at Future Construction your team places a lot of emphasis on clean code and good tests. You might also want to package everything into a nice Docker container, so that you can deploy it on the existing infrastructure.

Features

We'd like you to build a backend service that does the following:

1. Accesses a GCP bucket (https://cloud.google.com/compute/docs/disks/gcs-buckets) which contains a set of granule images in `.jp2` format
2. Expose an endpoint that fetches the images by band, and maps them to the RGB channels of
   a JPG, and returns that JPG
3. Your project should operate as a standalone service

Data source details

The data you will be accessing is accessible via a GCP bucket.

The bucket can be found here: https://console.cloud.google.com/storage/browser/gcp-public-data-sentinel-2/tiles/

Individual granules are stored at URLs that provide details about when and where the data was captured. For the purposes of this challenge, you don't need to parse the URLs.

The important thing is that within each granule folder, there is a set of images that [corresponds to the different bands (https://en.wikipedia.org/wiki/Sentinel-2#Instruments)](https://en.wikipedia.org/wiki/Sentinel-2#Instruments) that the instruments on the satellite capture.

For example, the images for one tile over eastern Germany can be found here:

```
https://console.cloud.google.com/storage/browser/gcp-public-data-sentinel-2/tiles/33/U/UP/S2A_MSIL1C_20150704T101337_N0202_R022_T33UUP_20160606T205155.SAFE/GRANULE/S2A_OPER_MSI_L1C_TL_EPA__20160605T113933_A000162_T33UUP_N02.02/IMG_DATA
```

The image files ending in `B02.jp2`, `B03.jp2`, and `B04.jp2` correspond to the standard blue, green, and red bands of visible light.

Endpoint specification

Your application should have one endpoint, `/generate-image`, that accepts a POST request with a JSON payload, which has the two following parameters:

- `granuleImages` - The path within the bucket to the images of a particular granule
- `channelMap`- A string naming a channel mapping to use. Include at least the following:
    - `visible` - Map channels 2, 3, and 4 to B, G, and R
    - `vegetation` - Map channels 5, 6, and 7 to R, G, and B
    - `waterVapor` - Map channel 9 to B only

The response should be an image with the mime-type `image/jpeg`.

Example request

```
POST /images
{
    "granuleImages":
"/33/U/UP/S2A_MSIL1C_20150704T101337_N0202_R022_T33UUP_20160606T205155.SAFE/GRANULE/S2A_OPER_MSI_L1C_TL_EPA__20160605T113933_A000162_T33UUP_N02.02/IMG_DATA",
    "channelMap": "visible"
}
```

Requirements

- You should spend no more than four hours on this challenge
- You can use any framework or language, so select one you are comfortable working in and know well
- Provide a README with instructions on how to setup and run the project

Helpful links

- Sentinel-2 dataset documentation – https://cloud.google.com/storage/docs/public-datasets/sentinel-2
- GCP bucket documentation – https://cloud.google.com/compute/docs/disks/gcs-buckets
- List of Sentinel-2 imaging bands – https://en.wikipedia.org/wiki/Sentinel-2#Instruments

Working with JPEG2000

Java

- ImageMagick – http://im4java.sourceforge.net/
- JAI ImageIO plugin for JPEG2000 – https://github.com/jai-imageio/jai-imageio-jpeg2000

Python

- Pillow (PIL) – https://pillow.readthedocs.io/en/5.3.x/
- Wand (ImageMagick bindings) – http://docs.wand-py.org/en/0.4.4/

Submission format

Please provide a link to an online Git repository, or a ZIP file which includes the local `.git` folder for the project.

We are looking forward to your creative solution!
The Engineering Team at *Interstellar*