UDACITY

‹ Return to Classroom

Engineering Full Stack Apps in the Cloud

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Congratulation 🎉

You have completed the "Engineering Full Stack Apps in the Cloud" project in this Nanodegree. You've done an excellent job with the implementation of the endpoint functionality and correctly used typescript best practices.

To further improve or learn more about Elastic Beanstalk, I would recommend to you reading the following articles:

- Security best practices for Elastic Beanstalk
- 6 Things to Know about AWS Elastic Beanstalk
- AWS Elastic Beanstalk Cheat Sheet

.
Hope the knowledge and techniques you learn throughout this course will help you in your career.
Keep up the great work

Engineering Process and Quality

All project code is stored in a GitHub repository and this link has been submitted for review. There are at least two branches - one for development (dev, development) and one master. Master should contain the most up-to-date, stable code at the time of submission.

Awesome

Great job! you've correctly added more than one branch. Adding an extra branch will allow you to make changes safely without breaking the production code on the master branch.

You could refer to the below link for some git commands 👇

- Github: Git Cheat Sheet

Any variables use typescript typing wherever possible, variable and function names are clear, endpoints are logically named. Good coding practices are followed.

Awesome

The goal of TypeScript is to be a static type-checker for JavaScript programs - in other words, a tool that runs before your code runs (static) and ensures that the types of the program are correct (type-checked). That's why is important to set the type of variable or parameters during their definitions.

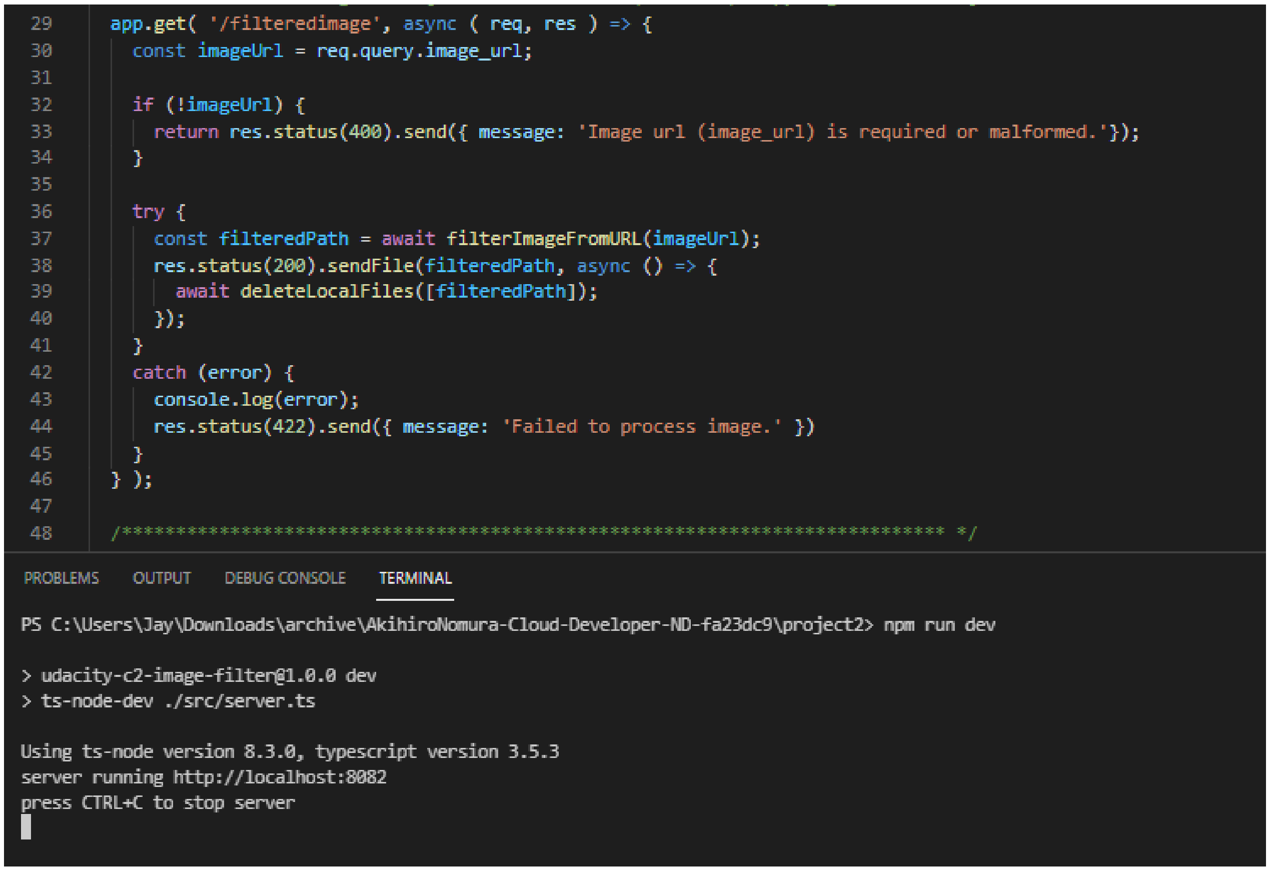
You could refer to the links below to familiarize yourself with more of the typescript best practices

- TypeScript Best Practices 2021 - Medium
- Do's and Don'ts

Development Server

Starting the server with `npm run dev` runs a local instance of the server with no errors

> npm run dev initiates a local server without errors. Kudos!!



```
app.get( '/filteredimage', async ( req, res ) => {
  const imageUrl = req.query.image_url;

  if (!imageUrl) {
    return res.status(400).send({ message: 'Image url (image_url) is required or malformed.'});
  }

  try {
    const filteredPath = await filterImageFromURL(imageUrl);
    res.status(200).sendFile(filteredPath, async () => {
      await deleteLocalFiles([filteredPath]);
    });
  }
  catch (error) {
    console.log(error);
    res.status(422).send({ message: 'Failed to process image.' })
  }
} );

/******************************************************************************* */
```

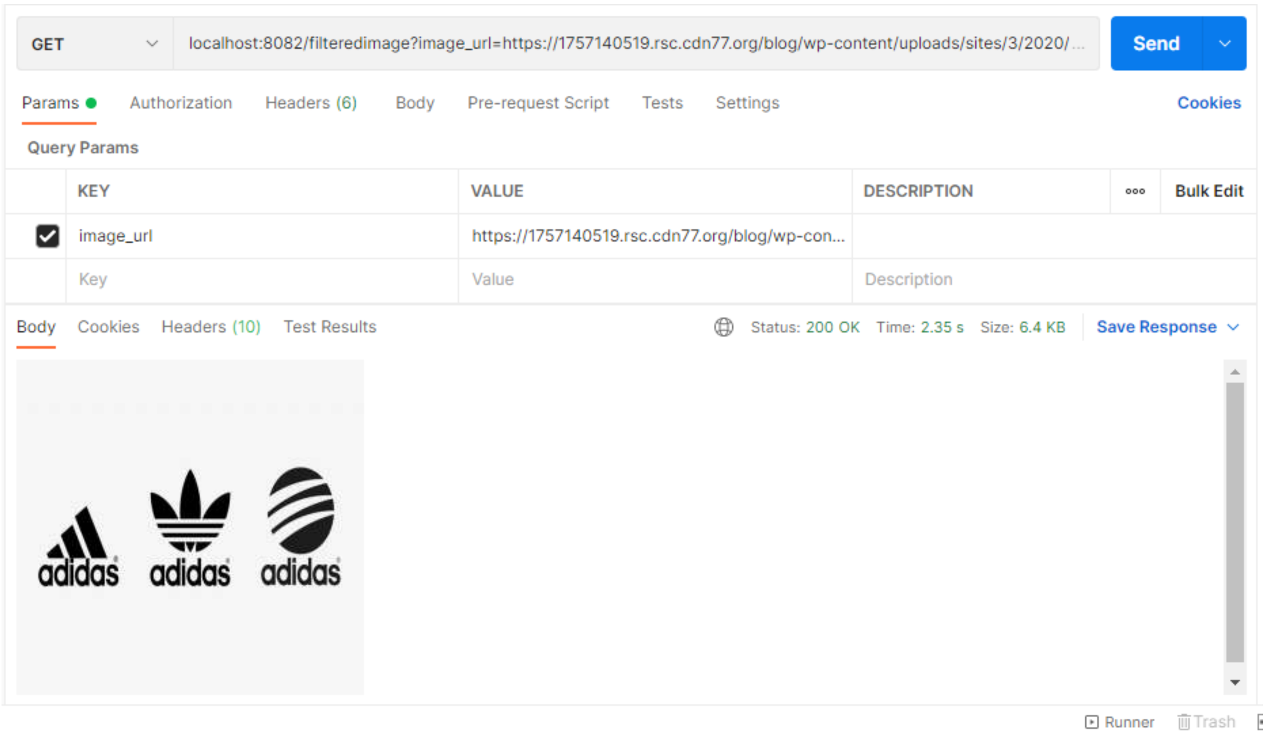
```
PS C:\Users\Jay\Downloads\archive\AkihiroNomura-Cloud-Developer-ND-fa23dc9\project2> npm run dev

> udacity-c2-image-filter@1.0.0 dev
> ts-node-dev ./src/server.ts

Using ts-node version 8.3.0, typescript version 3.5.3
server running http://localhost:8082
press CTRL+C to stop server
```

The stubbed `@TODO1` endpoint in `src/server.ts` is completed and accepts valid requests including: http://localhost:{{PORT}}/filteredimage?image_url=https://upload.wikimedia.org/wikipedia/commons/b/bd/Golden_tabby_and_white_kitten_n01.jpg

> I like the image filter algorithm you came up with. It correctly filters the image


GET localhost:8082/filteredimage?image_url=https://1757140519.rsc.cdn77.org/blog/wp-content/uploads/sites/3/2020/...

Query Params

	KEY	VALUE	DESCRIPTION
✓	image_url	https://1757140519.rsc.cdn77.org/blog/wp-con...	
	Key	Value	Description

Status: 200 OK Time: 2.35 s Size: 6.4 KB

Successful responses have a 200 code, at least one error code for caught errors (i.e. 422)

> Well done! You used status codes correctly.

HINT

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Refer to the below link to learn more about status codes

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

Elastic Beanstalk Deployment

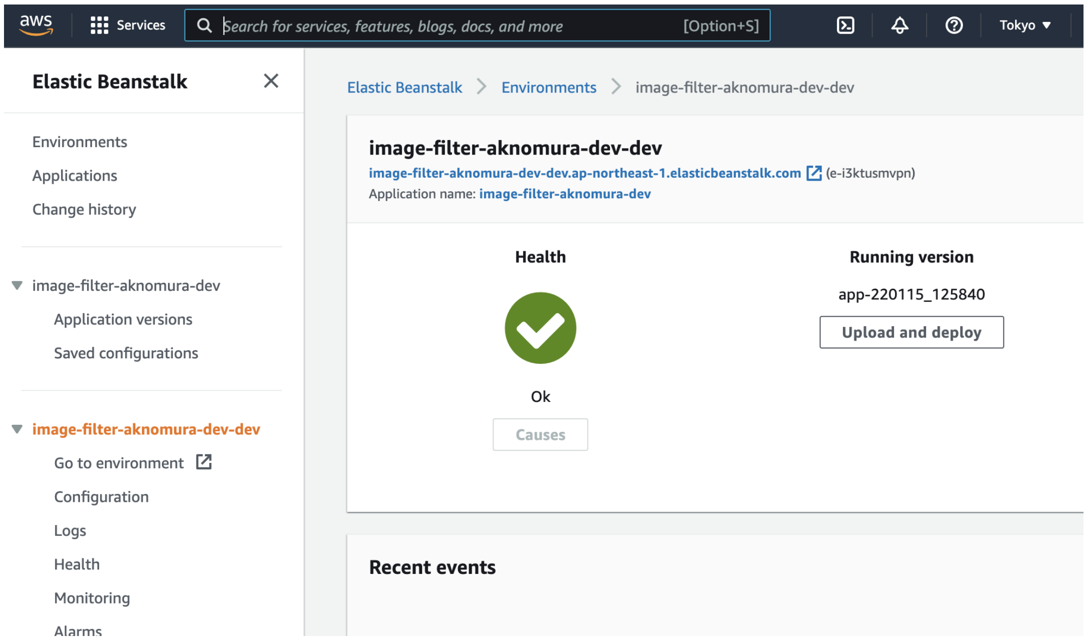
An endpoint URL for a running elastic beanstalk deployment (EB_URL) has been submitted along with the project submission. This endpoint responds to valid GET requests including:
http://{{EB_URL}}/filteredimage?
image_url=https://upload.wikimedia.org/wikipedia/commons/b/bd/Golden_tabby_and_white_kitten_n01.jpg

image_url=https://upload.wikimedia.org/wikipedia/commons/b/bd/Golden_tabby_and_white_kitten_n01.jpg

> The elastic beanstalk endpoint you submitted correctly responds to a valid GET request 👍

The project was deployed using the AWS Elastic Beanstalk CLI `eb init`, `eb create`, and `eb deploy` commands.

A screenshot of the elastic beanstalk application dashboard is included in a `deployment_screenshot` directory.

> It is evident you successfully deployed to Elastic Beanstalk as indicated by the screenshots you provided
>
> - Also, the URL you provided matches the EB endpoint present in the screenshot.
>
> 
>
>
> image-filter-aknomura-dev-dev
> image-filter-aknomura-dev-dev.ap-northeast-1.elasticbeanstalk.com (e-i3ktusmvpn)
> Application name: image-filter-aknomura-dev
> app-220115_125840

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START