Back End Documentation

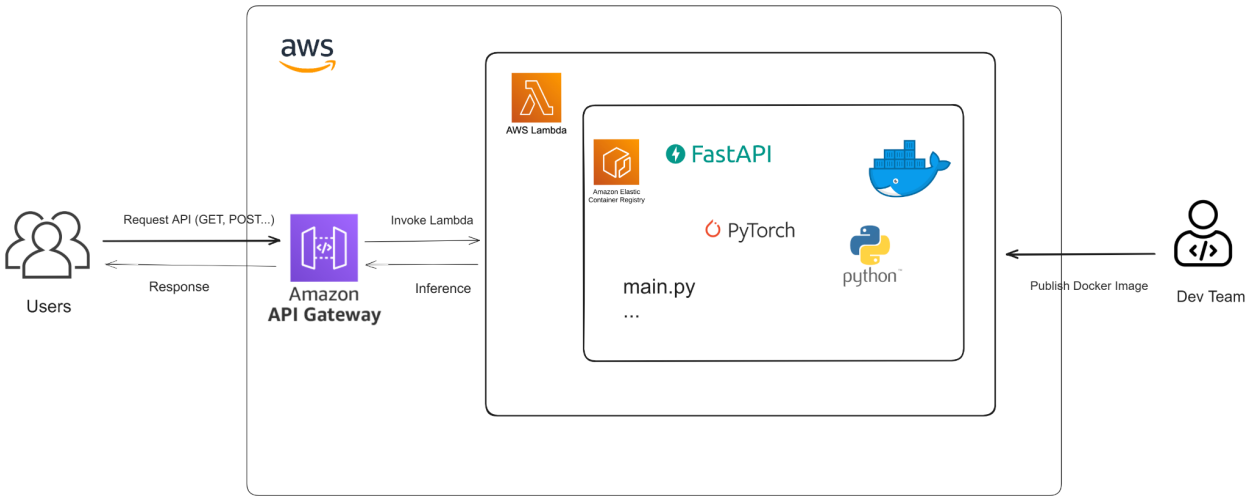
TeethSeg: Deploying a Machine Learning Model

Team:

ESSABIR Mohamed, EL BELGHITI Hamza, AKHMIM Abdelilah, EL QESSOUAR Tariq, ROCHDI Yahya

I. Back End Structure

The backend of TeethSeg is a RESTful API developed using Python FastAPI, ensuring robust functionality and reliability. For deployment we used the AWS Cloud using the Amazon Elastic Container Registry (ECR), Lambda and API Gateway services. Here's the architecture of the backend in the AWS Cloud.



Backend deployment in aws lambda

Note: You can deploy the backend on your own server by following the installation instructions provided below **(IV)** or to the **AWS Cloud (VI).**

II. Project Structure

.

├── model/ : Stores pre-trained MeshSegNet models.

├── temp/ : Temporary file storage during processing.

├── output/ : Stores files generated by the model for API responses.

├── config.py: Configuration and model loading.

├── helper.py: Utility functions for APIs.

├── main.py: Defines API endpoints.

├── meshsegnet.py: Encapsulates MeshSegNet's architecture and methods.

├── model.py: Contains `predict` and `predict_alpha` functions for 3D segmentation.

├── setup.py:Installs required packages and configures the workspace.

├── install-requirements.ps1: Configures the project on Windows.
├── install-requirements.sh: Configures the project on Linux/macOS.
├── Dockerfile: Used for deployment to Render hosting.
├── requirements.txt: Lists backend package dependencies.
├── README: Provides installation, configuration, and project information.
├── venv: Isolated environment for Python projects.
└── License: Project licensed under MIT License.

IV. Local Installation

To set up and run the backend locally:

1. You can run the setup.py script to configure your project.

```
$ python setup.py
```

Or by following these instruction:

2. Create a new venv.

```
$ py -m venv venv
```

3. Activate venv (Windows)

```
$ ./venv/Scripts/activate
```

4. Install dependencies

```
$ pip install -r requirements.txt
$ pip install pygco
```

5. Run server

After configuring your project now you can run the app using uvicorn.

```
$ uvicorn main:app --reload
```

V. API Endpoints

Here's the available endpoints and the methods to call them:

- "/": Description of the API and its routes
- "/api/v1/predict": Without post-proccessing (this will not give a good result)
- "/api/v1/predict/post_processing": With post-proccessing (The segmentation with post proccessing is more precise)

Go to http://localhost:8000/ if you see a message like {"message": "Hi 3DSF Interns!"} everything is working correctly.

Make sure to call api/v1/predict and api/v1/predict/post_processing endpoints with the POST method.

VI. Deploy to AWS Cloud

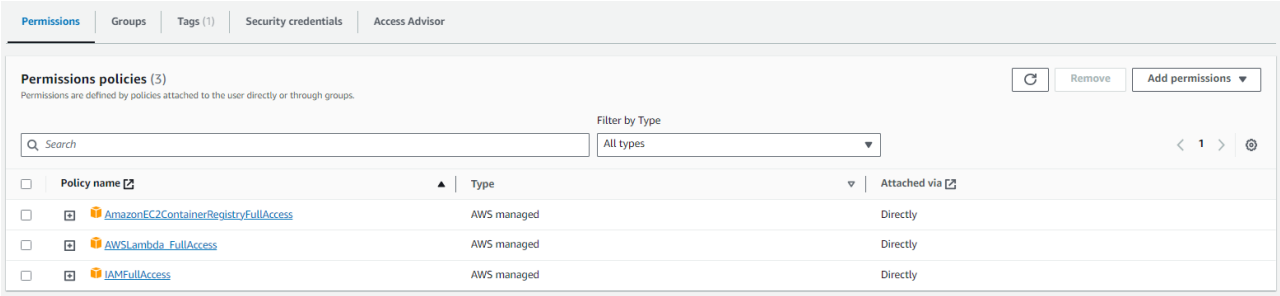
1. First, go ahead and clone the Back End repository from Github:

```
$ git clone https://github.com/3DSF-Internship/TeethSeg.git
```

2. Switch to the aws_lambda branch:

```
$ git checkout aws_lambda
```

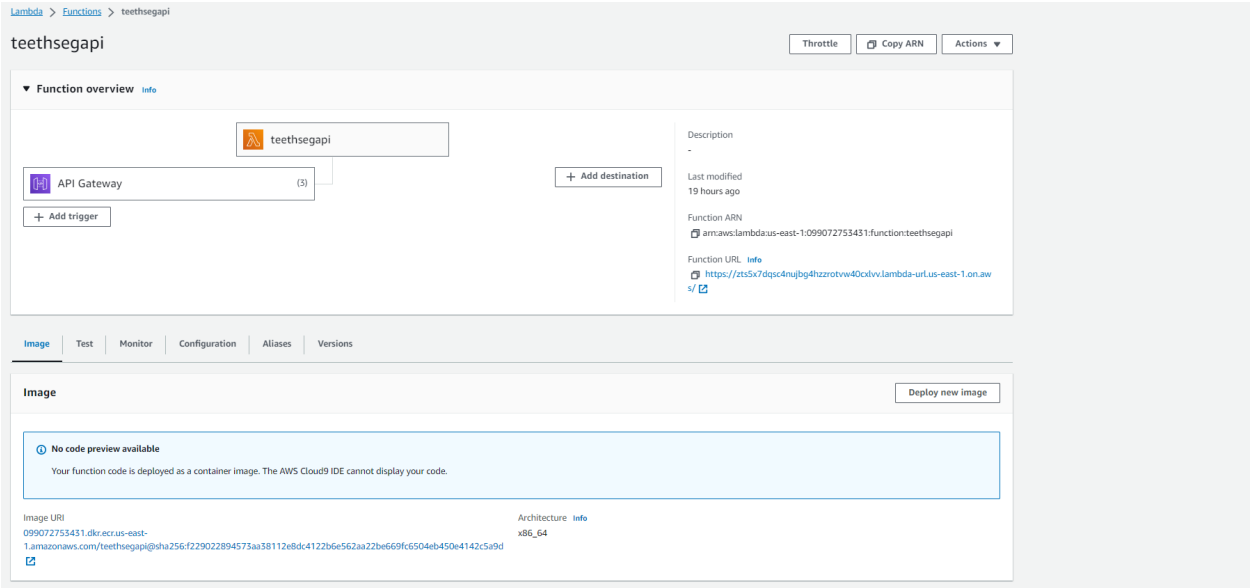
3. Now you need to create an AWS IAM User, then you should give the user those permissions:



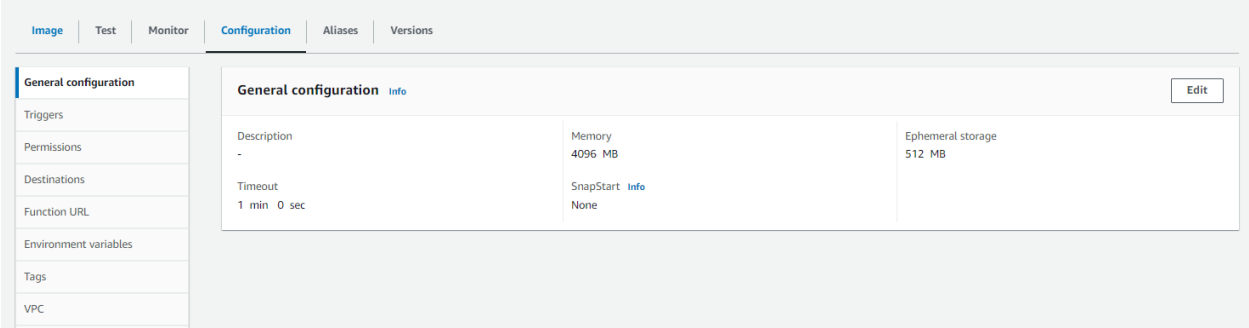
4. Now go ahead and connect to the IAM User, then create an ECR repository and push the Dockerfile in the folder you just cloned to the newly created ECR repository (See steps in the video below):

Deploy Machine Learning Model to AWS Lambda using Docker and FastAPI

5. After pushing the Docker Image to ECR, go ahead and create a new Lambda Function using that container Image.

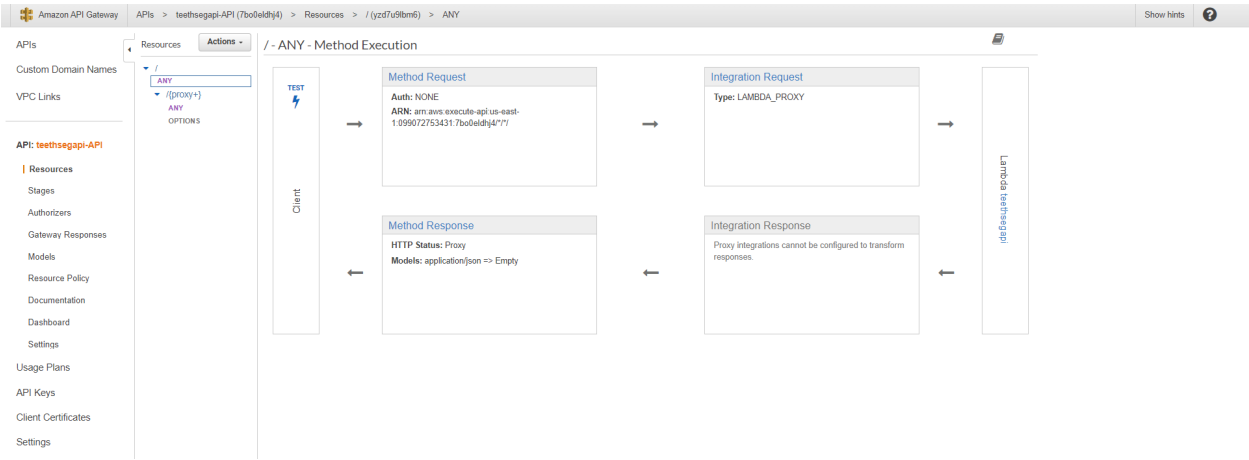


6. Now we need to configure our Lambda function. Change the memory allocated to 4096MB (4GB) (If you're not able to increase above 3GB, go ahead and contact the Support), and the set the timeout to 1 minute (normally our prediction is done within 15/20 seconds and the maximum memory use is 4GB).



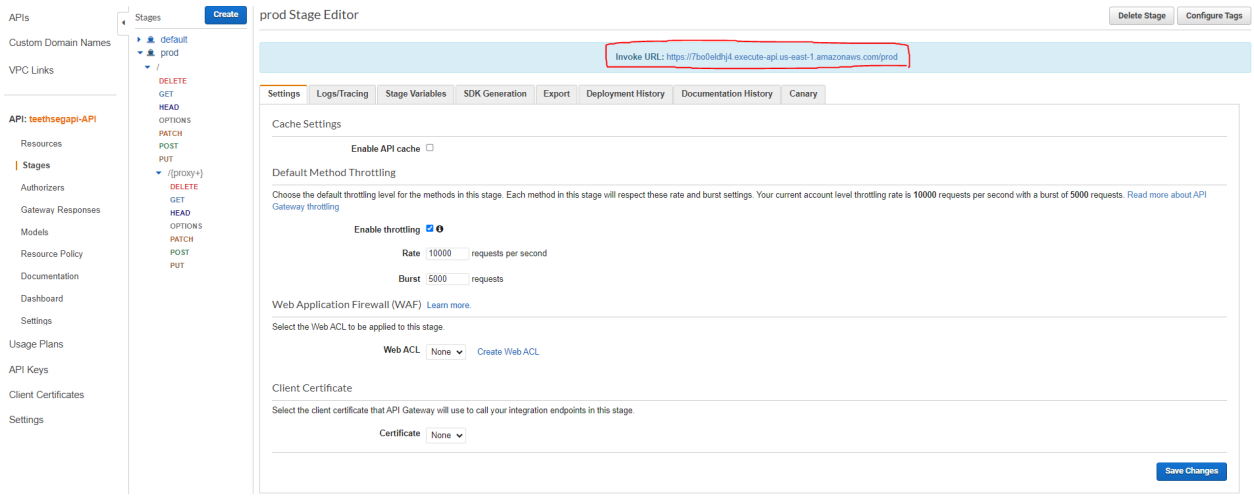
7. Now we need to add an API Endpoint that we're gonna use to communicate with our lambda function. For that we need to add an API Gateway to our Lambda function. See the video below for steps to take:

Connect AWS Lambda to API Gateway for REST APIs

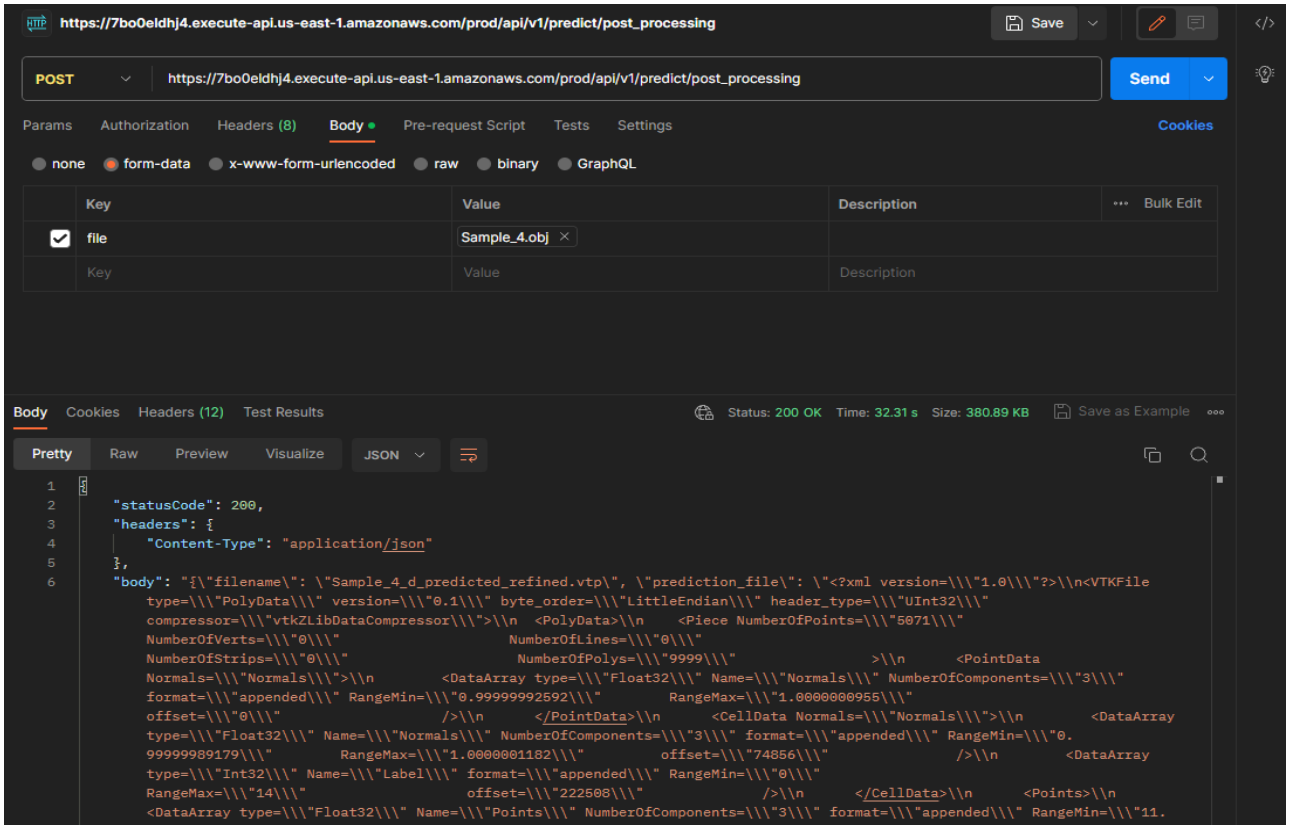


(NOTE: Configure the routes like this! You should have a "/" ressources and "ANY" for the methods, and a "/{proxy+}" ressource with "ANY" as method.) After doing this **remember to click "Actions" and "Deploy API" to have your url endpoint.**

You should see your deployed API in the "Stages" page.



8. Finally go ahead and test your API using the endpoint in Postman and check the response:



If you have any problem or question, please send an email to: ***hamza.lbelghiti@gmail.com***