Project Setup Instructions

This brief document will guide you through the steps required to install the AWS Comprehend Demo project. Please make sure you have the project files at hand; the .jar and .py files are required and the latter can be found in the "pythonLambda" folder. Review the next point for the .jar file. Please note that this Project was designed in the "eu-central-1" region, if you plan on operating in a different region please review the "free tier" offerings to make sure they apply to your region, too. To avoid any cost or functional problems, please refrain from distributing the elements of this project across different regions.

Building the .jar File

Please note that the project was built with maven. To manually build the project, navigate to the SpringBoot root folder "awsComprehendService" with the console and enter:
"mvnw repackage -DskipTests"
A new .jar file will be built in the "target" folder.

AWS Account

If you don't already have one, sign up with the AWS Cloud. The features used in this guide were all covered by the "free tier" at the time of writing (12/03/2020). Once logged in, services can be accessed via the search bar near the top.

Creation of an IAM Role

IAM is AWS's mechanism for controlling who gets to see or use which resources. As we want to use AWS Comprehend we're going to need to setup an entity with the permission to do so. To do this, open the IAM service, navigate to the user pane and create a new user with a name of your choice. Make sure the "Programmatic Access" box is ticked. On the next screen, "Attach existing policies" and add "ComprehendReadOnly". Click next until the user has been created, make sure to **write down the keys**. The "secret" keys (passwords, if you will) are only ever displayed once in AWS at creation.

Creation of the Inference Lambda

The Lambda will be responsible for making an inference call to Comprehend and then passing the answer back to the caller. Navigate to the Lambda service and "Create a Function". Then, "Author from scratch", enter a name, select the lastest Python runtime (3.7 currently) and then create the function. Make a note of the Lambda name. Once this has completed, scroll down to the Python code and replace it with the contents of the "comprehendInference.py" file. If you've setup in a different region you will need to change the "region_name" parameter. Refer to
https://docs.aws.amazon.com/general/latest/gr/rande.html for a list of valid endpoints.
Press save and now scroll down to "Environment variables". Edit them and add 2 new ones: "awsIDkey" and "awsSecretKey". For their values, enter the IAM keys you copied down in the previous step. Save and done!

Creation of the API Gateway

Navigate to the API Gateway console and create a new one. Select "REST API", ensuring that you select the one that **isn't** bound to a VPC. The settings can be left as they are, just enter a name for the Gateway. Then, press "Actions" and then create a resource. Enter a name and a custom resource path if desired (the resource path becomes a part of the URL). Create and then press "Actions" and create a method. Select "post" from the drop down menu that just appeared. Here, tick the "Use Lambda Proxy Integration" box, enter the name of the Lambda you created and press save. Allow gateway access to the Lambda. Press "Actions" again and deploy the API by creating a new stage in the popup window. Once this is done, expand the menu point (it has the name of your stage!) and navigate to the "POST" subheading. Press it and make a note of the "Invoke Url".

Creation of an RDS(PostgreSQL)

Navigate to the RDS console and create a database. Use the "Standard Create" option, select PostgreSQL and from the templates select "Free tier". Select a name for your database and the credentials. Make a note of them! Under the "Connectivity" heading, select "Create new VPC". Then select "Create new DB Subnet Group". Make the database publically accessible and finally create a new VPC security group. The other settings can be left as they are, create the database! This creation will take a few moments, in the meantime, start up the database querying tool of your choice, I recommend DBeaver.
Once the database is created, click on it and make a note of the "Endpoint". It should look something like this:
"postydb.cnb2slyihq9c.eu-central-1.rds.amazonaws.com"
Make a note of it. Now you can connect to the database with the afore mentioned tool (if the connection fails for whatever reason, view the next task). Once here, create a table with the following command:

```
CREATE TABLE text_sentiment (
        id serial primary KEY,
        feedback text,
        sentiment varchar(15),
        positive numeric(5,4),
        negative numeric(5,4),
        neutral numeric(5,4),
        mixed numeric(5,4),
        detectedLanguage varchar(4)
        );
```

Configuring RDS Connection Properties

Currently, the RDS database will not allow public access the way we want it to. To fix this, go to the RDS console and select the postgres database you created. In the information summary there will be a "VPC security groups" heading, click on the link. This will redirect you to another page where you will find a table with one entry. Click on the blue link (it will have the format "sg-" followed by numbers and letters). At the bottom, navigate to "Inbound Rules" and then edit them. Remove whatever IP address is there and from the dropdown menu select "Anywhere". After saving our RDS database can now be accessed from anywhere, including the Beanstalk!

Creation of the Beanstalk App

Navigate to the Elastic Beanstalk console. Create a new Application. Once this is complete, press "Actions" and create a new environment. Select "Web server environment". Here, set the "Platform" to Java. Under "Application code" select "Upload your code" and select the .jar file. Once the upload is complete, create the environment.
Once this is done, the app will likely "degrade"; this is because we haven't setup up required environmental variables yet! From the menu on the left, select "Configuration". Then "modify" the subheading "Software". You can find the environmental variables at the bottom, add the following to them:

Table 1: Beanstalk App Environment Variables

Variable Name	**Value**
db_username	database master username
db_password	database password
db_endpoint	database endpoint
gateway_uri	API Gateway invoke url
server_port	5000

Once you've entered them save and wait for the application to reboot. At the top, the endpoint of the application should be displayed. Copy this and add "/app/feedbackEntry" to the end of it. An exemplary full URL could look like this:
http://comptest-env.eba-m9yvrdup.eu-central-1.elasticbeanstalk.com/app/feedbackEntry
Enter this in the browser of your choice and everything should have worked!