UGA CSCI4795/6795: Cloud Computing (Spring 2019)

Programming Assignment 3 (PA#3) (out: Mar 19 2019; due: Mar 26 2019 – 11:59 p.m.)

Goals

Gain hands-on experience with Chatbots and "serverless" cloud applications

Introduction

In this assignment, we will create an intelligent nurse appointment system called **TAINA** (**T**he **AI**-based **N**urse **A**gent). Note that this is not a TV series. hmm… Anyway… TAINA is a chatbot – an "intelligent" agent that you interact with via a "chat" client – that a patient would use to get an appointment with a nurse. We're going to use some simulated data from a medical office. Your mission is to [a] create the framework for a running chatbot via Slack, AWS API gateway, and AWS Lambda and then [b] create the ability of your chatbot to answer a set of sample questions (provided below, in this assignment). While we will stop short of truly robust and intelligent processing (aka "AI"), successful completion of this assignment lays the groundwork for such future enhancements.

You are required to do this assignment alone.

Part 1: Create an account/workspace on Slack

Start by creating an account/workspace with Slack (https://slack.com/create#email). Note that you MUST use your UGA email address and you MUST "create a new team". After confirming the email verification, you MUST specify your Slack workspace (question of "what's the name of your company or team") as "UGA_ID_csci4795" (replace "UGA_ID" with your UGA ID, e.g., "ik32805_csci4795"), and use PA3 for the question of "what's a project your team is working on?" Then "SKIP" sending invitations (question of "who else is working on this project").

Part 2: Create 'hello world' from Slack to AWS

We will now create just the basic framework of our chatBot by leveraging an existing AWS Lambda template. All of these are on AWS except where noted:

1. [SLACK] Go to an app config page on your new Slack: http://UGA_ID_csci4795.slack.com/apps Then search "App Directory" for "Slash Commands". Click "Add Configuration". Make the command "**/TAINA**" and then "Add Slash Command Integration". Scroll down to the Token. **Don't do anything here – just start the "AWS" section on a different browser tab.**
2. Make a new encryption key: on the main AWS console web page, search for "kms" or "Key Management Service". Then make a new key ("create key" button) named **TAINAkey** (for alias and tag key), with all values being the default.
3. On the AWS console, select Lambda (under "Compute"), then the "Create function" button (or "Get Started Now"). Select "Use a Blueprint", type in "slack" into the "Filter". Select the "slack-echo-command-python" and hit the "Configure" button.

4. On the BIG next page:
    a. RE: "Basic information": Make "Function name" be "**TAINAfn**", create new role from AWS policy templates (type "**TAINArole**" for "Role name"), no need to add any additional policy templates other than "AWS KMS decryption permissions"
    b. RE: "API Gateway Trigger": Select "Create a new API" and, set Security to be "Open". Expand "Additional settings", name it "**TAINAapi**", and make "Deployment stage" be "**prod**".
    c. RE: Lambda function code: no need to change the code at this time. At the bottom of this section, expand "Encryption configuration", select "Enable helpers for encryption in transit", select "**TAINAkey**". Then go to your slack page and copy the "Token". Paste this token into the "<enter value here>" and hit "Encrypt" (you want it to show "******** …")
    d. Then hit "Create function". (Give it a moment – it might look like it's dead).
    e. On the next screen, select "API Gateway", then at the bottom expand below the "TAINAapi" (next to "Method: ANY" to find the "Invoke URL". Copy that.
5. [SLACK] Paste the URL from the previous step into the "URL" slot, then scroll to the bottom and hit "Save Integration".
6. Open a new browser tab to the top-level of your Slack (e.g., https://ik32805-csci4795.slack.com/). At the bottom of the screen type a new message: "**/taina go dawgs**" (if you get a "timeout" message or other error messages, just try it again). [If you instead see "Explore Slack", click on that first.] If you're successful, you will see the response (with your ID, not mine): "**`ik32805 invoked /taina in general with the following text: go dawgs`**"

If this is not successful, re-check your slack integration configuration. **Cut-and-paste this Slack browser screen (showing success) and enter it into the "What to submit" document.**

Part 3: Add Python Artificial Intelligence Markup Language (AIML)

We will now add an existing Python implementation of the Artificial Intelligence Markup Language (AIML), which will help us with the bot mechanics of getting and generating the dialogue. At this point, launch a micro "Amazon Linux AMI" that you will use as the development environment (note that the user is "ec2-user" instead of "ubuntu"). Then:

7. In the AWS console, create a new S3 bucket called "your-UGA-ID-taina" (e.g., "ik32805taina"). The region should be "US East (N. Virginia)"
8. Cut-and-paste the existing Lambda code (lambda_function.py – you can see this if you click a lambda icon with TAINAfn) into a file on the VM called TAINA.py (you can use `vi` or `emacs` – install via "`sudo yum install emacs`")
9. On the VM, zip this up via "zip -r TAINA.zip TAINA.py"
10. On the VM, upload to your S3 bucket via something like "aws s3 cp TAINA.zip s3://ik32805taina/" (replace the S3 URL with your own bucket). You will need to do "aws configure" to config your security.
11. In the AWS/Lambda console, click on the "configuration" tab. In the "Function code" section [a] change the "code entry type" to "Upload a file from Amazon S3", [b] change the handler to "TAINA.lambda_handler" and [c] specify your ZIP (such as "https://https://s3.amazonaws.com/ik32805taina/TAINA.zip"). Do not select "Enable helpers for encryption in transit", and do not hit "remove" – this security configuration does not need to be changed. Then hit "Save" and re-test via the Slack browser tab.

Now add the Python implementation of AIML and confirm that it works on the VM and on AWS:

12. On the VM, create a rudimentary "seed" file for the AIML component of your botnet by putting the following into "taina.aiml":

```
<?xml version="1.0" encoding="utf-8"?>
<aiml version="1.0.1">
      <category>
        <pattern>MY NAME IS * </pattern>
        <template>
          Hi <star index="1"/>, I am TAINA!
        </template>
      </category>

      <category>
        <pattern>WHAT TIME IS IT</pattern>
        <template>The local time is: <system>date</system></template>
      </category>
</aiml>
```

13. On the VM execute the following:
    a. `virtualenv ~/taina`
    b. `source ~/taina/bin/activate`
    c. `pip install boto3`
    d. `pip install aiml`
    e. Change TAINA.py (e.g., via `vi` or `emacs`)
        i. Right after the last import statement at the top add:
        ```
        import aiml
        bot = aiml.Kernel()
        bot.learn("taina.aiml")
        ```
        ii. Specify your region-name in `kms = boto3.client(….)` statement by adding
        `region_name='us-east-1'`
        iii. At the bottom of the file add this (and then save the file)
        ```
        sentence = "What time is it?"
        print bot.respond(sentence)
        ```
    f. `export kmsEncryptedToken=`***the_value_from_your_AWS_Lambda***
    g. Confirm that it works via "`python TAINA.py`" (it should show "The local time is:…")
        i. If you have an error, please debug and fix it with error messages
14. Now we can deploy this to AWS Lambda via the following:
    a. Change the "`return respond…`" code (right above the sentence = "What time is it?" line) to return the output of the aiml bot to Slack:

    ```
    return respond(None, "%s : %s" % (command_text, bot.respond(command_text)))
    ```

    b. Package up everything via (Note that we are making a NEW zip file called TAINA2.zip)
    ```
    rm ~/TAINA2.zip; cd $VIRTUAL_ENV/lib/python2.7/site-packages; zip -r9
    ~/TAINA2.zip * ; cd $VIRTUAL_ENV/lib64/python2.7/site-packages; zip -r9
    ~/TAINA2.zip * ; cd ~; zip -g TAINA2.zip *.py; zip -g TAINA2.zip
    taina.aiml
    ```

c. Type "`deactivate`" to get out of the Python virtual env
d. Upload this new ZIP into your S3 bucket ("`aws s3 cp TAINA2.zip s3://ik32805taina/`") and follow the steps from above to deploy it. Now test it via Slack via

"/TAINA what time is it"
"/TAINA My name is In Kee" (replace "In Kee" with your name)

Note: if you do this and you see {"message": "Internal server error"}, click on the "Monitoring" tab of your Lambda function, then "View logs in CloudWatch". Then debug based on what these logs say. **Cut-and-paste this browser screen (showing success) and enter it into the "What to submit" document. To get full credit, the screenshot should have both test cases 1) what time is it and 2) my name is …**

Part 4: Create ChatBot to Help with Scheduling an Appointment with a Nurse

We are now ready to help get an appointment. Note we are only going to show proof-of-concept; you are required to answer two mandatory questions and you are required to answer ONE additional relevant question of your choosing.

The two mandatory questions are:

1) Who is available time-and-date?
2) When is person available on date?

If the user provides an invalid person reply "Sorry, but we don't have a Nurse *nurse_last_name* in this office. Make sure to look up your nurse by last name!" If the user attempts to make an appointment OUTSIDE of 9-5pm Mon-Fri, you MUST reply "Sorry, the clinic is only open from 9am-5pm Mon-Fri."

Additional notes and/or helpful hints:

1. The DBMS (MySQL) is at 52.70.223.35. The username is "`clinicuser`" and the password is "`sparky19`". The DB is `ClinicDB`; there are 2 tables: `nurses` and `nurse_schedule`. This is available for manual (browser) inspection via http://52.70.223.35/phpmyadmin/
2. I recommend that you use PyMYSQL to connect your python code to this database.
3. It is expected that you dev and test ("python TAINA.py") ENTIRELY on the VM. You do NOT have to test new functionality by deploying to Lambda.
4. When starting a new dev session (i.e., when logging on to the VM), be sure to execute steps 13b and 13f.
5. "TAINA.py" probably should only be changed to add some manual testing statements for execution on the VM (at the bottom of the file). The general approach you might consider is to [a] in 'taina.aiml', trigger off of the particular sentence and invoke a custom python program for each (e.g., when_nurse.py), using the AIML "star" functionality, and then [b] in the particular python file, hit the MySQL DB to pull the info out and generate a response. Note that you cannot directly test our Lambda via the Lambda "test" button (because it won't provide the necessary security token).

In your “What to submit” document, provide the SQL commands that you used for your 3 questions and cut-and-paste a browser screen showing sample invocations of your 3 questions. You are required to submit to eLC:

1. A PDF of your “What to submit” document
2. A ZIP of all of your Python code – NOT the zip that you upload into S3!
3. Your taina.aiml file

Do NOT delete your Lambda and API Gateway when you are finished with this assignment!