Exercise - Polly

This an optional exercise for those interested in Amazon Polly for lifelike text-to-speech.

This course has covered a lot of the fundamentals of APIs, SDKs, the AWS CLI, and authentication. You can now start applying this knowledge to learn a new service. In order to challenge your knowledge a little, this exercise is less prescriptive than the others. The answers to the challenges are hidden in expandable sections. You may

Loading [a11y]/explorer.js answer the challenges before revealing the answer.

Amazon Polly is a service that turns text into lifelike speech, allowing you to create applications that talk and build entirely new categories of speech-enabled products. Amazon Polly is a text-to-speech service that uses advanced deep learning technologies to synthesize speech that sounds like a human voice.

1. Create Amazon Polly synthesized speech with the AWS CLI.

- In your AWS Cloud9 instance terminal, you can find the Amazon Polly API operations by running the following command.

```
aws polly help
```

- The help is scrollable. Press `spacebar` to scroll though the text and press `Q` to close the help.
- You can also refer to the AWS CLI Command Reference for Polly.
- To create synthesized speech from Amazon Polly, consider using the API operation `synthesize-speech`. You can get more details with the CLI command below.

```
aws polly synthesize-speech help
```

- The mandatory parameters for the operation are:

```
synthesize-speech
--output-format <value>
--text <value>
--voice-id <value>
outfile <value>
```

- Scrolling through the help reveals the values for `output-format` and `voice-id`.

- As an experiment, create an MP3 file using `synthesize-speech`.
- In your AWS Cloud9 instance terminal, return to your `~/environment` folder and try the command below.

  ```
  aws polly synthesize-speech --output-format mp3 --text 'Any text you want' --voice-id Matthew output.mp3
  ```

  Are you signed in as the **edXProjectUser** user? Did you receive an error message similar to the one below?

  ```
  An error occurred (AccessDeniedException) when calling the SynthesizeSpeech operation: User: arn:aws:iam::012345678912:user/edXProjectUser is not authorized to perform: polly:SynthesizeSpeech
  ```

Challenge: Do you know how to fix the `AccessDeniedException` exception? Where is *authorization* for your **edXProjectUser** user defined?

▼ Expand for challenge solution.

Your goal is to add a policy to the **edXProjectUser** that will allow Amazon Polly operations. There is more than one way to do this. To add the AWS managed policy **AmazonPollyReadOnlyAccess**, follow the instructions below.

- In the AWS Console, open the **IAM dashboard**.
- In the left navigation menu, click **Users**.
- Click **edXProjectUser**.

- On the **Permissions** tab, click **Add permissions**.
- Click **Attach existing policies directly**.
- In the search box below, type **AmazonPollyReadOnlyAccess**
- In the search results, click the check box next to **AmazonPollyReadOnlyAccess**.

  You can also click on the **AmazonPollyReadOnlyAccess** hyperlink. That opens a new window with all the details of the managed policy, including the JSON policy document.

- Click **Next: Review**.
- Click **Add Permissions**.
- Return to your Cloud9 IDE and rerun the `aws polly synthesize-speech` command above. It may take a minute or two for the permissions update to propagate.

- In the Cloud9 IDE, you will see the **output.mp3** file appear in the **Environment** list of files on the left side.
- Double-click the **output.mp3** file. The file will play inside of Cloud9. This screen also has a download arrow icon. Click it to download the file.

2. Explore the Amazon Polly voices.

Your previous Polly command used the voice-id **Matthew**. The voice-id determines the language, accent, and gender of the synthesized voice.

Challenge: Can you find the details of the **Matthew** voice?

Challenge: Find the male Australian English voice. First, visit the AWS CLI Command Reference for Polly.

▼ Expand for challenge solution.

- In your AWS Cloud9 instance terminal, type `aws polly describe-voice`
- The Australian English voice is named **Russell**. Conincidently, that's the name of your Australian English-accented presenter!

Challenge: Add Amazon Polly to the photos application

The Boto 3 SDK allows Python developers to write software that uses Amazon services. We can make some small changes to the Photos application to add a feature to greet users with their nickname.

Again, this section will not be 100% prescriptive. It'll require a little Python knowledge and debugging.

- First, add a route that returns the MP3 contents of a user greeting.
- Open **/exercise-cognito/FlaskApp/application.py** from your previous exercise.

```
@application.route("/members_voice")
@flask_login.login_required
def members_voice():
    """A polly synthesized voice"""
    polly = boto3.client("polly")
    message = "hello %s welcome back" %
flask_login.current_user.nickname
    response =
polly.synthesize_speech(VoiceId='Nicole',
Text=message, OutputFormat='mp3')

    polly_bytes = response['AudioStream'].read()
    return send_file(
        io.BytesIO(polly_bytes),
        mimetype='audio/mpeg',
        cache_timeout=-1
    )
```

- Review the code. What do you expect this code to do when a user logs in and accesses the route? What do you expect it to do when a user who is *not* logged in accesses the route?
- The Cloud9 IDE will display some warnings for `io` and `send_file`. Scroll up to the top of the code and add `io` (this is a new line), and `send_file` (this is an update to an existing line).

```
import io

from flask import Flask, render_template_string,
session, redirect, request, url_for, send_file
```

- Save **application.py**.
- Test the new **members_voice** route:
    - Point the **Python3RunConfiguration** to **/exercise-**

cognito/FlaskApp/application.py and click **Run**.

- To test the application, click **Preview -> Preview Running Application** on the top menu bar of the Cloud9 environment.
- Pop out the application in a new window by clicking the **Pop Out** button.
- You should see the application load. Log in to the application.
- In your browser add **members_voice** to the end of the current location to access the new route, e.g. **https://0123456789abcdef0123456789abcdef.vfs.cloud9.us-west-2.amazonaws.com/members_voice**

Now you have a route into your application that returns a customized welcome message. Next you can update the HTML template to include an audio player.

- Open **/exercise-cognito/FlaskApp/templates/main.html** from your previous exercise.
- Find a location in the HTML template where the user is greeted. It will look like this:

```
<li><p class="navbar-text">Hello,
{{current_user.nickname}}!</p></li>
```

- Replace all of this code with the version below that includes an audio player.

```
<li><p class="navbar-text">Hello, 
{{current_user.nickname}}!
<span class="glyphicon glyphicon-volume-up" 
style="cursor: pointer;" 
onclick="document.getElementById('audio').load();docu
ment.getElementById('audio').play();"></span>
<audio id="audio">
  <source src="{{ url_for('members_voice') }}" 
type="audio/mpeg">
</audio></p>
</li>
```

- Save **main.html**.
- Ensure that your Run Configuration is still running. Return to your application in the Cloud9 preview.
- Log in to your application, if you are not already.
- Click the speaker icon beside your name. You now have Amazon Polly text integrated into your application!