Firstly, your answer may differ from the model answer in many ways. For example, you may have chosen to encode your images differently and packaged your Lambda function differently, too.

Below are some justifications for the structure of the model answer.

**ZIP folder**

- You'll notice that the packages and files within the ZIP folder follow a flat structure. This approach is recommended for Lambda functions and can be shown by the additional resource provided for deploying Python code as a ZIP folder.
- The ZIP includes a lambda_function.py file, which holds our encoding code.
- It includes an XML file, which the encoding code uses.
- It also includes a requirements.txt file, which the security lead requested.

**lambda_function.py**

- The function name lambda_handler is what will be run when invoking the lambda function.
- First, the boto3 client is instantiated. Boto3 is the package you can use to read data files from S3.
- Then the face cascader is instantiated. The encoding code will use this to identify faces within the image.
- Data from the lambda request is then read, and an image file from S3 is loaded.
- This file is converted to an image format using OpenCV.
- The file is converted into colour using OpenCV.
- Then, the file is converted into grayscale using OpenCV as this is used by the facial detection algorithm.
- After this, the face from the images is cropped to remove noise around the outside of the image.
- The cropped image of a face is then cropped and resized for the HOG descriptor.
- The HOG descriptor computes features using histograms of gradients, which is just one of many ways you can compute features from an image.
- Finally, the HOG feature descriptor is computed. This is the output vector computed by the features extracted using HOG.
- The results of the hog descriptor are returned.