Qwiklabs Assessment:

Automate Updating Catalog Information

# Introduction

You work for an online fruits store, and you need to develop a system that will update the catalog information with data provided by your suppliers. The suppliers send the data as large images with an associated description of the products in two files (.TIF for the image and .txt for the description). The images need to be converted to smaller jpeg images and the text needs to be turned into an HTML file that shows the image and the product description. The contents of the HTML file need to be uploaded to a web service that is already running using Django. You also need to gather the name and weight of all fruits from the .txt files and use a Python request to upload it to your Django server.

You will create a Python script that will process the images and descriptions and then update your company's online website to add the new products.

Once the task is complete, the supplier should be notified with an email that indicates the total weight of fruit (in lbs) that were uploaded. The email should have a PDF attached with the name of the fruit and its total weight (in lbs).

Finally, in parallel to the automation running, we want to check the health of the system and send an email if something goes wrong.

# What you’ll do

* Write a script that summarizes and processes sales data into different categories
* Generate a PDF using Python
* Automatically send a PDF by email
* Write a script to check the health status of the system

# Fetching Supplier Data

You'll first need to get the information from the supplier that is currently stored in a Google Drive file. The supplier has sent data as large images with an associated description of the products in two files (.TIF for the image and .txt for the description).

Here, you'll find two script files download\_drive\_file.sh and the example\_upload.py files. You can view it by using the following command.

ls ~/

Output:



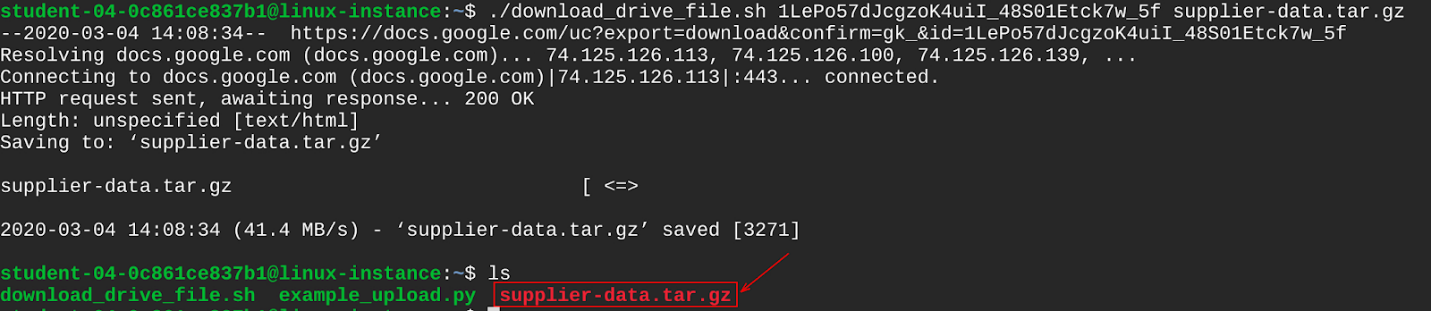
To download the file from the supplier onto our linux-instance virtual machine we will first grant executable permission to the download\_drive\_file.sh script.

sudo chmod +x ~/download\_drive\_file.sh

Run the download\_drive\_file.sh shell script with the following arguments:

./download\_drive\_file.sh 1LePo57dJcgzoK4uiI\_48S01Etck7w\_5f supplier-data.tar.gz

Output:



You have now downloaded a file named supplier-data.tar.gz containing the supplier's data. Let's extract the contents from this file using the following command:

tar xf ~/supplier-data.tar.gz

This creates a directory named supplier-data, that contains subdirectories named images and descriptions.



List contents of the supplier-data directory using the following command:

ls ~/supplier-data

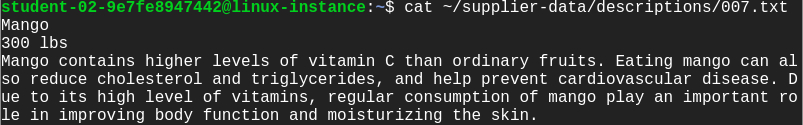
Output:



The subdirectory images contain images of various fruits, while the descriptions subdirectory has text files containing the description of each fruit. You can have a look at any of these text files using cat command.

cat ~/supplier-data/descriptions/007.txt

Output:



The first line contains the name of the fruit followed by the weight of the fruit and finally the description of the fruit.

# Working with Supplier Images

In this section, you will write a Python script named changeImage.py to process the supplier images. You will be using the PIL library to update all images within ~/supplier-data/images directory to the following specifications:

* **Size**: Change image resolution from **3000x2000** to **600x400** pixel
* **Format**: Change image format from **.TIFF** to **.JPEG**

Create and open the file using nano editor.

nano ~/changeImage.py

Add a shebang line in the first line.

#!/usr/bin/env python3

This is the challenge section, where you will be writing a script that satisfies the above objectives.

**Note:** The raw images from images subdirectory contains alpha transparency layers. So, it is better to first convert RGBA 4-channel format to RGB 3-channel format before processing the images. Use convert("RGB") method for converting RGBA to RGB image.

After processing the images, save them in the same path ~/supplier-data/images, with a JPEG extension.

Once you have completed editing the changeImage.py script, save the file by clicking **Ctrl-o**, **Enter** key, and **Ctrl-x**.

Grant executable permissions to the changeImage.py script.

sudo chmod +x ~/changeImage.py

Now run the changeImage.py script:

./changeImage.py

Now, let's check the specifications of the images you just updated. Open any image using the following command:

file ~/supplier-data/images/003.jpeg

Output:



Click *Check my progress* to verify the objective.

Update image specifications

Check my progress

# Uploading Images to Web Server

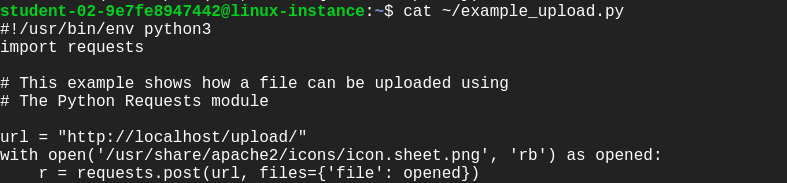
You have modified the fruit images through changeImage.py script. Now, you will have to upload these modified images to the web server that is handling the fruit catalog. To do that, you'll have to use the Python requests module to send the file contents to the [linux-instance-IP-Address]/upload URL.

Copy the external IP address of your instance from the Connection Details Panel on the left side and enter the IP address in a new web browser tab. This opens a web page displaying the text "Fruit Catalog".

In the home directory, you'll have a script named example\_upload.py to upload images to the running fruit catalog web server. To view the example\_upload.py script use the cat command.

cat ~/example\_upload.py

Output:



In this script, we are going to upload a sample image named icon.sheet.png.

Grant executable permission to the example\_upload.py script.

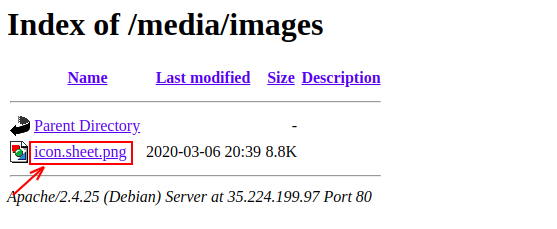
sudo chmod +x ~/example\_upload.py

Execute the example\_upload.py script, which will upload the images.

./example\_upload.py

Now check out that the file icon.sheet.png was uploaded to the web server by visiting the URL [linux-instance-IP-Address]/media/images/, followed by clicking on the file name.

Output:



In a similar way, you are going to write a script named supplier\_image\_upload.py that takes the **jpeg** images from the supplier-data/images directory that you've processed previously and uploads them to the web server fruit catalog.

Use the nano editor to create a file named supplier\_image\_upload.py:

nano ~/supplier\_image\_upload.py

Complete the script with the same technique as used in the file example\_upload.py.

Once you have completed editing the supplier\_image\_upload.py script, save the file by typing **Ctrl-o**, **Enter** key, and **Ctrl-x**.

Grant executable permission to the changeImage.py script.

sudo chmod +x ~/supplier\_image\_upload.py

Run the changeImage.py script.

./supplier\_image\_upload.py

Refresh the URL opened earlier, and now you should find all the images uploaded successfully.

Output:

