**Pipeline Data Management Scripts With Dropbox API Documentation**

**Author**: Yaniv Malach  
**Date**: 5.9.2023

**Introduction:**

**User Perspective:**

* **Verifying Files:** Ensures all required files for each sample are present and correct.
* **Metadata Update:** Consults the metadata from Emory (stored in Macaque R24\ /subject\_metadata/metadata.xlsx) and creates a spreadsheet listing missing metadata for any new samples and subjects (stored in Macaque R24/results /missing.xlsx).
* **Notification:** Sends daily summary reports to a dedicated Slack channel named **#** **r24\_macaque\_samples.**

**Operational Details:**

* Frequency and Time: The script runs at 00.00 US EST.
* Target Directory: The script scans /macaque r24/sequencing in Dropbox. This can be changed in app.py under the variable DROPBOX\_FOLDER\_PATH.
* Metadata File: The script checks metadata against a file named **config.json**, located in the same target directory.

**Slack Channel:**

The script sends notifications and summary tables to a Slack channel named **#** **r24\_macaque\_samples**.

Tables Produced:

1. **Slack Table 1**: Contains summary data about AIRR-Seq and Genomic samples.
   * Columns: Date, Total Subjects, Total Samples, Samples added in past 24 hours, Samples missing files.
2. **Slack Table 2**: Contains information about subjects missing metadata.
   * Columns: Date, Subjects missing metadata.
3. **Excel Table**: This table keeps track of each file path and whether it has been run in the pipeline.

**How it is Installed:**

**Server Details:**

The script runs on **Jenkinx**, in the **/work/jenkins/dropbox\_scripts/** directory.

**Configuration File:**

The configuration file named **secrets.json** is located in the same directory as the script and contains settings Slack Webhook URL, Drobox: Refresh token, APP\_KEY, APP\_SECRET**.**

The script runs under the **igguest** account.

**Setup**

1. **Create a Dropbox App**: Go to the [Dropbox Developers Console](https://www.dropbox.com/developers/apps) and create a new app to get your **APP\_KEY** and **APP\_SECRET**.
2. **Get Authorization Code**: Visit this URL to get an authorization code.
3. **Set** SOURCE\_PATH and other constants
4. The script should be placed in the **/work/jenkins/dropbox\_scripts/** directory and runs on Jenkins.

**Secrets File**:

Create a **secrets.json** file and fill in the **APP\_KEY**, **APP\_SECRET**, **REFRESH\_TOKEN** and Slack Webhook url.

תמונה שמכילה טקסט, צילום מסך, תצוגה, תוכנה

התיאור נוצר באופן אוטומטי

**Notes**

* The **REFRESH\_TOKEN** does not expire, so you typically only need to run **generate\_refresh\_token()** function once to get it.
  + For generating the Slack **Webhook URL**, navigate to Workspace -> Administration -> Manage apps -> Build -> Select your App -> Add Features and Functionality -> Incoming Webhooks -> Create a new Webhook URL.

**Classes:**

**App.py:**

* The program uses a cursor to keep track of the last state it was in to ensure that it picks up from where it left off during the next run.
* **start\_new\_check(server\_dropbox, cursor, folder\_monitor):** The function that does the heavy lifting. It gets the Dropbox files, downloads new files if any, and finally updates some sort of summary and table. It returns the result containing the new cursor.

**FolderMonitor.py:**

* includes various functions for reading and processing metadata files, performing filesystem checks, sending messages to Slack, and writing information to Excel files.
* **check\_new\_subject(subject\_path)** the function that start the checks of the folder

**ServerDropbox**: allows you to interact with Dropbox to perform several tasks:

* Refreshing the Dropbox access token.
* Connecting to Dropbox.
* Listing files in a Dropbox folder.
* Downloading files and folders from Dropbox to a local directory.

**How it Works:**

**Repository:**

The code is stored in a Git repository named <https://github.com/yanivm96/Macaque-R24-FolderMonitoring> .

**Code Functionality:**

The script is written in Python and uses the following libraries:

1. **requests**: For sending HTTP requests, particularly to Slack.
2. **pandas**: For handling and updating the Excel table.
3. **tabulate**: For formatting tables sent to Slack.
4. **gzip and shutil**: For unzipping **.gz** files.

The script follows these steps:

1. It reads from the **secrets.json** to get initial setup details.
2. Checks the Dropbox directory for new files.
3. Validates the new files based on **metadata.json**.
4. Manages file unzipping and renaming.
5. Updates the Excel metadata table.
6. Sends a summary to the Slack channel.

Python Library for Dropbox:

The script uses the Dropbox API's Python SDK for accessing Dropbox directories. You can find the documentation [here](https://www.dropbox.com/developers/documentation/python).

For any further details or issues, feel free to reach out to the author.