Using the Galaxy API

While Galaxy's main use case is through an easy-to-use web interface, it also provides a REST API (`https://aws.amazon.com/what-is/restful-api/`) for programmatic access. There are interfaces provided in several languages, for example, Python support is provided by **BioBlend** (`https://bioblend.readthedocs.io`).

BioBlend allows you to automate running analyses on a Galaxy server. It includes both, a standard and object-oriented API.

In this recipe, we will use Galaxy and the BioBlend library to perform a few basic operations and upload data. This will give us a sense of how to access Galaxy programmatically via the API.

Getting ready

We will be working in our Jupyter notebook for this exercise. It is found under the `galaxy` subdirectory in our working directory. Open the `Ch15-2-galaxy-api.ipynb` notebook.

First, we will install the `BioBlend` library:

```
! pip install bioblend
```

Make sure `fastq-dump` is also installed for this exercise. You can check this by typing (from the terminal) the following:

```
fastq-dump -h
```

If it is not already installed, you can install it using `conda`.

Optionally set up channels if not already added:

```
conda config --add channels conda-forge
conda config --add channels bioconda
```

Install the following code using `conda`:

```
conda install -y sra-tools
```

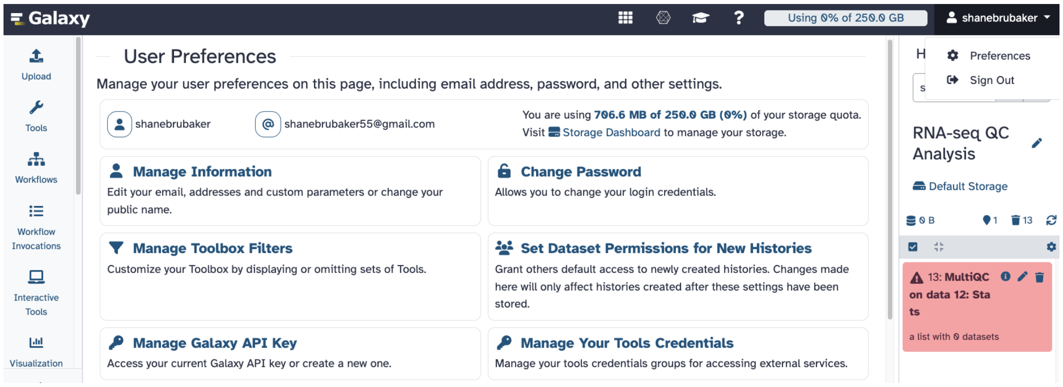
OR – on a Mac, you can install it with Homebrew:

```
brew install sratoolkit
```

Next, we need to get our Galaxy API key:

Go ahead and log in to your `usegalaxy.org` account.

Go to your user name in the upper right, click on **Preferences | Manage API Key** as shown in the following screenshot:



User Preferences

Manage your user preferences on this page, including email address, password, and other settings.

shanebrubaker shanebrubaker55@gmail.com

You are using 706.6 MB of 250.0 GB (0%) of your storage quota.
Visit Storage Dashboard to manage your storage.

Manage Information
Edit your email, addresses and custom parameters or change your public name.

Change Password
Allows you to change your login credentials.

Manage Toolbox Filters
Customize your Toolbox by displaying or omitting sets of Tools.

Set Dataset Permissions for New Histories
Grant others default access to newly created histories. Changes made here will only affect histories created after these settings have been stored.

Manage Galaxy API Key
Access your current Galaxy API key or create a new one.

Manage Your Tools Credentials
Manage your tools credentials groups for accessing external services.

Bonus Figure 15.1 – Managing your API key in the usegalaxy.org interface

Click on **Create New Key**.

Save your key somewhere. In a few moments we will go into the configuration section of the code and replace the section `your_api_key_here` with your actual API key.

Next, we will obtain the FASTQ files for the recipe:

```
! fastq-dump --split-files --gzip -X 25000 SRR1039508
```

Rename the files for ease of use:

```
! mv SRR1039508_1.fastq.gz sample1_R1.fastq.gz
! mv SRR1039508_2.fastq.gz sample1_R2.fastq.gz
```

The code for this recipe can be found in `Ch15/galaxy/Ch15-2-galaxy-api.ipynb`.

How to do it...

Here are the steps to perform the prerequisite steps from the *Getting ready* section:

1. First, let's import our libraries:

```
from bioblend.galaxy import GalaxyInstance
import time
import os
```

2. Next, we will set up our configuration:

```
GALAXY_URL = https://usegalaxy.org
API_KEY = "your_api_key_here"
HISTORY_NAME = "Simple Galaxy Demo"
```

We will set our Galaxy server as `usegalaxy.org`. Note that you could also try this example with your local Docker container if you want by using `http://localhost:8080` - make sure to use that server's API key.

Remember to place the API key you generated in the area marked `your_api_key_here`.

We will also set a name for our history. In Galaxy a **History** is a workspace that saves your uploaded data, analyses, and resulting outputs. It includes the parameters, intermediate files, and other details of the programs you ran.

3. Now we will connect to Galaxy:

```
def main():
    print(f"Connecting to Galaxy at {GALAXY_URL}")
    gi = GalaxyInstance(url=GALAXY_URL, key=API_KEY)
    try:
        version = gi.config.get_version()
        print(f"Connected to Galaxy version: {version}")
    except Exception as e:
        print(f"Connection failed: {e}")
        return
```

This is our main function. We first connect to Galaxy using our URL and API key through the `GalaxyInstance()` function. We then retrieve and print the server version.

4. This next section of code will create a **History** to save our work:

```
    print(f"Creating new history: {HISTORY_NAME}")
    history = gi.histories.create_history(name=HISTORY_NAME)
    history_id = history['id']
    print(f"History created with ID: {history_id}")
```

Please note that we are still within the `main()` function so the indentation must reflect that. We use our Galaxy instance and the `histories.create_history()` function to create our history workspace.

5. Great! Let's perform a simple operation with the Galaxy API and list the available tools:

   ```
       print("\nListing some available tools:")
       try:
           tools = gi.tools.get_tools()
           print(f"Total tools available: {len(tools)}")
           for i, tool in enumerate(tools[:10]):
               print(f"  {tool['id']} - {tool['name']}")
           common_tools = ['upload1', 'cat1', 'Cut1',
                           'sort1', 'Grep1']
           available_common = []
           for tool in tools:
               if any(
                   common in tool['id'] for common in common_tools
               ):
                   available_common.append(tool['id'])
           print(f"\nCommon tools found: {available_common}")
       except Exception as e:
           print(f"Could not list tools: {e}")
   ```

   This code uses our Galaxy instance to run the `tools.get_tools()` function and get a list of available tools. We list the top 10 tools and also check for the presence of some common tools like `cat` or `grep`.

6. Let's try uploading some sample files as follows:

   I. Take a look at section 6 of the code. In this code we check for the existence of our sample FASTQ files (obtained in the *Getting ready* section) and then create a Galaxy dataset using the `tools.upload_file()` function.

   II. If we are unable to upload the data, we create a simple test file and upload the content directly using the `tools.paste_content()` function.

   III. We next enter a loop to poll for the status of the uploaded files and check that the upload has completed successfully.

7. Now we try a simple, basic tool as follows:

   I. Now review this section of code in the notebook. Here, we check for the existence of a basic tool.

   II. We then perform a basic action on the first uploaded file using the `tools.run_tool()` function.

   III. We include a loop to check for completion of the task.

8. Review the contents of your **History**:

```
    print(f"\nFinal history contents:")
    try:
        history_contents = gi.histories.show_history(
            history_id, contents=True)
        datasets = history_contents.get('contents', [])

        for dataset in datasets:
            print(
                f" {dataset['name']} - {dataset['state']} - "
                f" {dataset['extension']}"
            )

    except Exception as e:
        print(f"Could not list history contents: {e}")

    print("\n" + "="*50)
    print("BASIC WORKFLOW COMPLETED")
    print("="*50)
    print(f"History: {HISTORY_NAME}")
    print(f"Files processed: {len(uploaded_datasets)}")
    print(f"Galaxy URL:{GALAXY_URL}/history/view/{history_id}")
    print("\nThis demonstrates basic Galaxy API functionality.")
    print("For more complex workflows, "
          "ensure required tools are installed.")
```

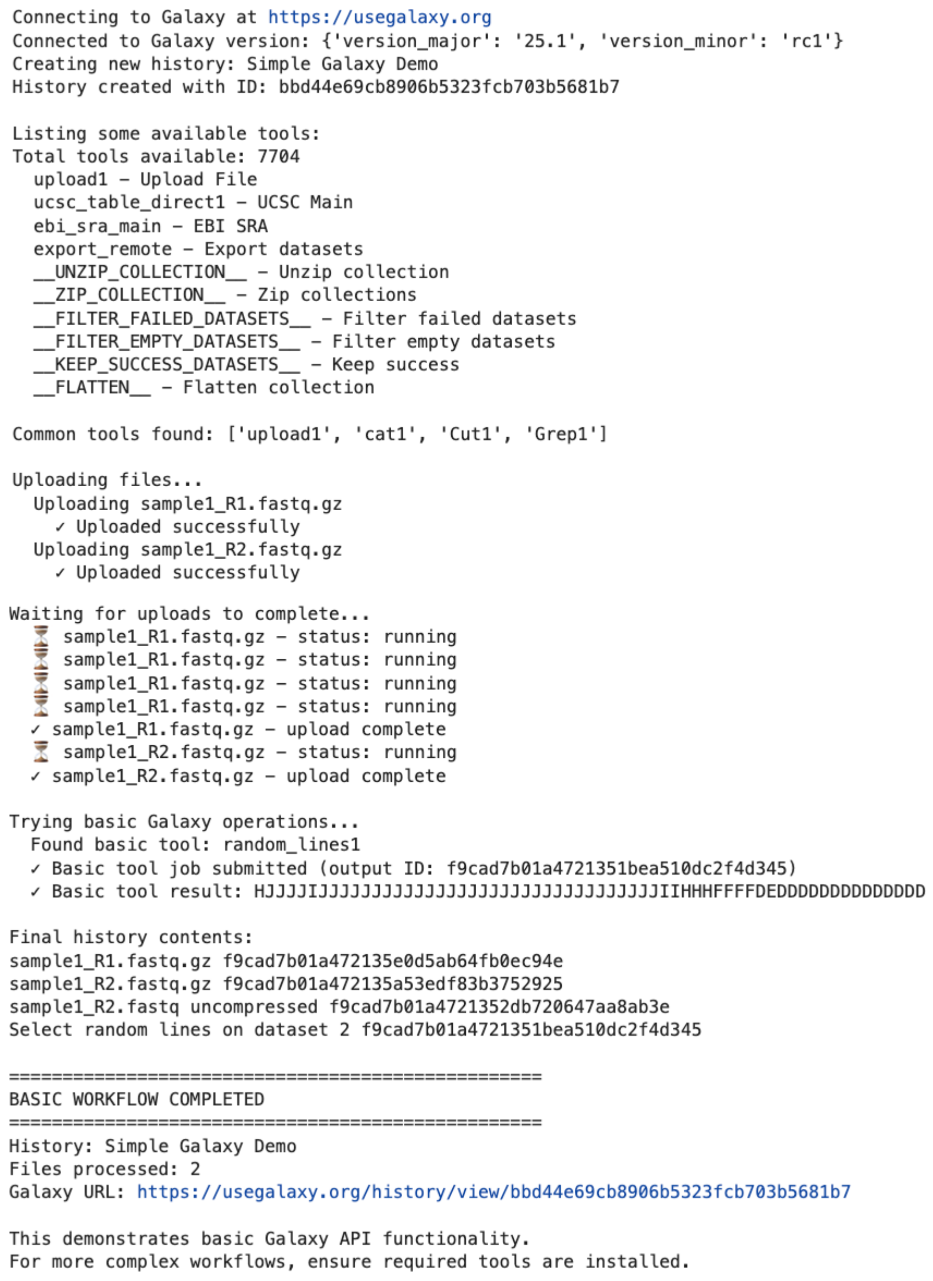
This code lists out the history of our workflow and gets its contents. We print detailed information about each dataset. We also print a summary of the overall workflow execution.

9. Great! Let's run our code:

```
if name == "main":
    if API_KEY == "your_api_key_here":
        print("Please configure your API key first!")
        print("Go to Galaxy -> User -> Preferences -> Manage API 
Key")
    else:
        main()
```

We check that the API key has been set and then run our `main()` function.

Here is what our output looks like:

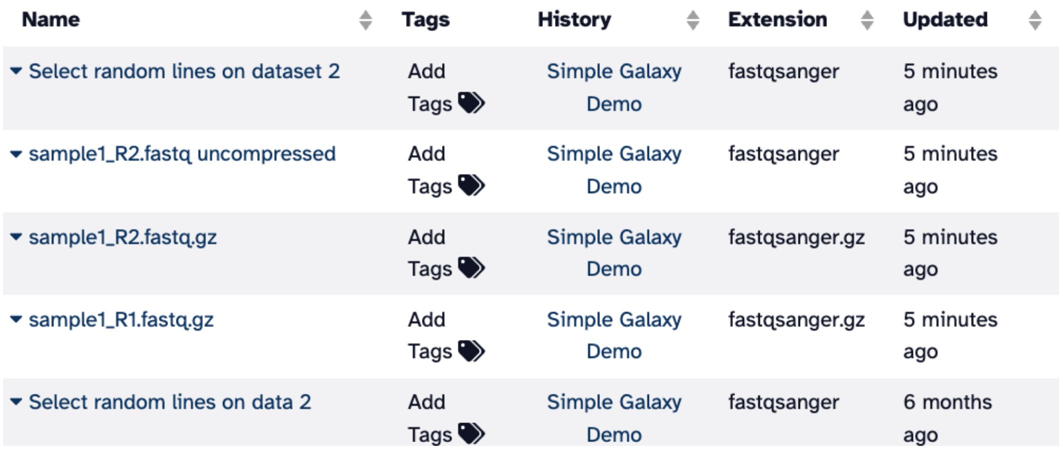


```
Connecting to Galaxy at https://usegalaxy.org
Connected to Galaxy version: {'version_major': '25.1', 'version_minor': 'rc1'}
Creating new history: Simple Galaxy Demo
History created with ID: bbd44e69cb8906b5323fcb703b5681b7

Listing some available tools:
Total tools available: 7704
  upload1 - Upload File
  ucsc_table_direct1 - UCSC Main
  ebi_sra_main - EBI SRA
  export_remote - Export datasets
  __UNZIP_COLLECTION__ - Unzip collection
  __ZIP_COLLECTION__ - Zip collections
  __FILTER_FAILED_DATASETS__ - Filter failed datasets
  __FILTER_EMPTY_DATASETS__ - Filter empty datasets
  __KEEP_SUCCESS_DATASETS__ - Keep success
  __FLATTEN__ - Flatten collection

Common tools found: ['upload1', 'cat1', 'Cut1', 'Grep1']

Uploading files...
  Uploading sample1_R1.fastq.gz
    ✓ Uploaded successfully
  Uploading sample1_R2.fastq.gz
    ✓ Uploaded successfully

Waiting for uploads to complete...
  ⏳ sample1_R1.fastq.gz - status: running
  ⏳ sample1_R1.fastq.gz - status: running
  ⏳ sample1_R1.fastq.gz - status: running
  ⏳ sample1_R1.fastq.gz - status: running
  ✓ sample1_R1.fastq.gz - upload complete
  ⏳ sample1_R2.fastq.gz - status: running
  ✓ sample1_R2.fastq.gz - upload complete

Trying basic Galaxy operations...
  Found basic tool: random_lines1
  ✓ Basic tool job submitted (output ID: f9cad7b01a4721351bea510dc2f4d345)
  ✓ Basic tool result: HJJJJIJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJIIHHHFFFFDEDDDDDDDDDDDDDD

Final history contents:
sample1_R1.fastq.gz f9cad7b01a472135e0d5ab64fb0ec94e
sample1_R2.fastq.gz f9cad7b01a472135a53edf83b3752925
sample1_R2.fastq uncompressed f9cad7b01a4721352db720647aa8ab3e
Select random lines on dataset 2 f9cad7b01a4721351bea510dc2f4d345

==================================================
BASIC WORKFLOW COMPLETED
==================================================
History: Simple Galaxy Demo
Files processed: 2
Galaxy URL: https://usegalaxy.org/history/view/bbd44e69cb8906b5323fcb703b5681b7

This demonstrates basic Galaxy API functionality.
For more complex workflows, ensure required tools are installed.
```

Bonus Figure 15.2 – Output of the Galaxy API workflow

We see information on our connection to Galaxy online, list some common tools, check for expected basic tools, and then upload our files. We then enter our loop to check for completion of file upload. We perform a run of a basic tool, look at our **History** contents, and provide a summary. In this case the basic tool was `random_lines()` so we see the output returned is a random line from the file.

Now go into the Galaxy website and look at the **Datasets** tab. This is what our work looks like in the **Datasets** tab:



Name	Tags	History	Extension	Updated
Select random lines on dataset 2	Add Tags	Simple Galaxy Demo	fastqsanger	5 minutes ago
sample1_R2.fastq uncompressed	Add Tags	Simple Galaxy Demo	fastqsanger	5 minutes ago
sample1_R2.fastq.gz	Add Tags	Simple Galaxy Demo	fastqsanger.gz	5 minutes ago
sample1_R1.fastq.gz	Add Tags	Simple Galaxy Demo	fastqsanger.gz	5 minutes ago
Select random lines on data 2	Add Tags	Simple Galaxy Demo	fastqsanger	6 months ago

Bonus Figure 15.3 – Output of the Galaxy workflow in the Datasets tab

There's more...

This is just a quick introduction to the Galaxy API. You are encouraged to explore further in the documentation available here: `https://galaxyproject.org/develop/api/`

See Also

- Read more about BioBlend here: Sloggett et al, *BioBlend: automating pipeline analyses within Galaxy and CloudMan*, BioInformatics, Jul 2013 - `https://academic.oup.com/bioinformatics/article/29/13/1685/185761`
- This tutorial covers a range of interesting aspects of Galaxy including AI development on GPUs - `https://training.galaxyproject.org/training-material/topics/statistics/tutorials/gpu_jupyter_lab/tutorial.html`