Linux command line (sh/Perl/Python) interface to Google groups

Mentor: Rick McGowan Contributor: Sanju Raj

Abstract

This project aims to develop a suite of command-line interface (CLI) modules tailored for Linux environments, enabling automation of common Google Groups operations. These modules, predominantly in sh, Perl, and Python, will leverage various Google APIs to facilitate tasks seamlessly. Primarily, the tool will serve as a bridge between Little Green Light (LGL), the unicode's central contact management system, and Google Groups. By integrating with LG's daily reports, the tool will enable automatic updates to Google Groups based on constituent data changes. This ensures that Google Groups remain synchronised with the single source of truth provided by LGL. Expected outcomes encompass a robust CLI toolset capable of:

1. Adding members to groups.
2. Removing members from groups.
3. Modifying member roles within groups.
4. Validating membership status based on email addresses.
5. Efficiently removing members from all groups.
6. Listing groups in the organisation linked to specific individuals.

By automating these tasks, the project aims to save significant time and effort previously spent on manual group management processes. The tool's modular design will facilitate easy integration into existing workflows, enabling seamless execution via cron jobs and shell scripts.

Project Goals

This project simplifies the management of Google Groups by automating common tasks. It uses LGL reports as the primary data source to ensure accuracy. Through a user-friendly console interface, the tool efficiently synchronises groups, reducing manual effort.

Implementation

I plan to approach the implementation in two phases. Firstly, I'll develop a functional solution using Perl, Bash, Python, or a combination of these languages. In the second phase, I'll focus on refining the solution. This involves setting it up as a cron job for automatic execution, ensuring its robustness through rigorous testing, improving the console interface for clarity and ease of use, and exploring opportunities for optimization. Along the way, I'll also document the process, providing a reference guide for extending the tool's functionality in the future.

Technical details

Part 1: getting data from linux processes parsing the LGL reports.

As Rick mentioned, there are existing processes that parse the LGL reports. We can access the data from these processes directly using interprocess communication. Alternatively, we can write the data to a file and then read it back, but this method is less efficient because it involves frequent writing and reading from the disk.

Please note that the implementation might change later if we discover a better way to access the data from these processes or if we learn more about the specific Linux process and its output format.

Part 2: Using the data to perform the required operation.

After obtaining the parsed data we will use the google APIs listed below to complete the required operations.

Functionality required and relevant google APIs:

Add a person (email address) to a group.	POST `https://admin.googleapis.com/admin/directory/v1/groups/{groupKey}/members`
Remove a person from a group.	DELETE https://admin.googleapis.com/admin/directory/v1/groups/groupKey/members/memberKey
Change a person's status, e.g. to make them owner/manager of a group.	PUT https://admin.googleapis.com/admin/directory/v1/groups/groupKey/members/memberKey
Check to see if a specific email address is in a group.	GET https://admin.googleapis.com/admin/directory/v1/groups/groupKey/members/memberKey

Remove a person from all groups.	We will first find all the groups whose user is a member and then we can remove him from all groups one by one.
List all groups in the org for which specific person is a member.	GET https://admin.googleapis.com/admin/directory/v1/groups/groupKey/members/memberKey

Note: Some users have reported difficulties in interacting with these APIs. If they don't work as expected, we can explore alternatives like the Google Admin SDK Directory API.During the community bonding period (May 1 to May 26), I will connect with Rick and other community members to discuss potential additional operations and features.

Testing

Before attempting to execute this script on the actual Google Workspace, we'll first run it in a similar environment to the original one. This approach will allow us to identify any potential bugs without affecting the original Google Workspace. For generating test data, I plan to utilise one of the dummy data generators available online. Here are the tests I'm tentatively planning to conduct, although they may change based on requirements:

- Unit testing to ensure that all the operations mentioned above work as intended.
- Functional testing to verify that the application functions correctly as a whole.
- Performance testing to identify opportunities for optimization.

Reference Projects

These are the list of projects which we can refer to while implementation.
1) **GNU Mailman:**
https://www.gnu.org/software/mailman/ (Suggested by rick, I had a look at it and it is mostly similar except for the part that it is for locally hosted groups).
2) **Hubspot:**
https://github.com/HubSpot/python-app-google-groups?tab=readme-ov-file (This is a slack app for managing google groups, It is doing mostly similar thing except that it is hosted as slack app)
3) **blackbaud-to-google-group-sync:**
https://github.com/groton-school/blackbaud-to-google-group-sync (Sync membership of Blackbaud LMS community groups to Google Groups, we can look for implementation of role based groups here)

Timeline

My university final exams conclude by the first week of May, and I don't have any other commitments during the summer, so I'll be available to work on Unicode full-time if selected. I can dedicate 7-8 hours per day, totaling over 40 hours per week.

Since I've already developed a project for managing Google Groups in a school environment, I'm well-acquainted with Google APIs and won't require much time to understand them. Additionally, I

have a strong background in Python and C. I'll use my free time in April to learn Perl and Bash further.

I am planning to complete this project in four phases,

Here is the detailed timeline:

Phase 0 (Pre GSOC till May 1):

1) Spend some time expanding my knowledge on Perl and Bash.
2) Go through the projects mentioned in the reference part above and get a good understanding of them.

Phase 1 (May 1 to May 20):

1) Get in touch with Rick and other community members, spend time in finalising the design (Exact feature, Implementation language).
2) Start the implementation of the first part of the project, a working set of scripts to manage the google groups.

This will roughly require 2-3 weeks of time.

Phase 2 (May 20 to June 5):

1) Complete the implementation of the first part and test it out on dummy groups.
2) Present the current Implementation to the Rick/community members to get feedback on console interface and potential optimisations.
3) Finalise the console interface.

This will roughly require 2 weeks of time.

Phase 3 (June 5 to July 5):

1) Implement a more clean and intuitive and clean console interface.
2) Do performance testing and see the possibility for optimisations.
3) Add documentation

This will require 3-4 weeks of time.

Phase 4 (July 5 to July 30):

1) Implement the possible optimisations
2) Do final testing
3) Fix out the possible bugs
4) Add it as cron job
5) Add some more documentation

This will require 2-3 weeks of time.

I have kept 2-3 weeks of buffer time if things go south or maybe Implementing specific portion takes a lot of time.

About me

- Name - Sanju Raj
- University - Indian Institute of Technology BHU
- Email - sanjuraj57421@gmail.com
- Time Zone - IST (GMT +5:30)
- Preferred Language For Communication: English

I'm a first-year undergraduate student pursuing a bachelor's degree in Electronics and Communication Engineering. I've been programming in Python, C, and C++ for a year now. Ubuntu is my daily operating system. This summer and beyond, I'm eager to work with Unicode.

With my background in Python and Google APIs, I believe I'm well-equipped to handle this project. I'm also open to learning new things along the way.

Over the past one year, I've completed some programming projects and assignments in Python and c.

Post GSOC

As I've mentioned before, I'm truly passionate about contributing to open source projects. I'm particularly intrigued by Unicode because it's used by the vast majority of internet users. Contributing to Unicode would be an exciting experience because it impacts billions of people worldwide. Currently, I'm exploring new languages related to the project, and I've found the Unicode/icu4x repository fascinating. I plan to contribute to it in the near future. Additionally, I'll always be available for any future changes or extensions to this project.

References

1. https://docs.google.com/document/d/e/2PACX-1vQwQhjcjYENWYTfA4_FEsyVPZzqhp-QAyeVfPcYR9x0RFqL4mhby1hWKatjiG6Itwkuz8Q4ZXSF_IZz/pub
2. https://www.gnu.org/software/mailman/
3. https://github.com/HubSpot/python-app-google-groups?tab=readme-ov-file
4. https://github.com/HubSpot/python-app-google-groups?tab=readme-ov-file
5. https://developers.google.com/open-source/gsoc/timeline
6. https://google.github.io/gsocguides/student/writing-a-proposal
7. https://github.com/unicode-org/icu4x/discussions/4727#discussioncomment-8955865