Message Recall: Deployment Guide

Last Updated: Wed Oct 14, 2015

CONTENTS

Overview

Message Recall is a Google-AppEngine-hosted application designed to run in a **single Google Apps domain**. The following instructions describe a detailed process for deploying the application:

1. Create a Google Apps Domain Super-Admin ‘Role’ account
2. Create the App Engine hosting environment
3. Configure the Google APIs Console Project
4. Configure your Google Apps Domain for the App Engine application
5. Prepare the source code
6. Download the Google App Engine SDK
7. Upload the source code into the App Engine host
8. Test the running application

The end to end process should take about 30 minutes (not including study time to learn about App Engine or applications).

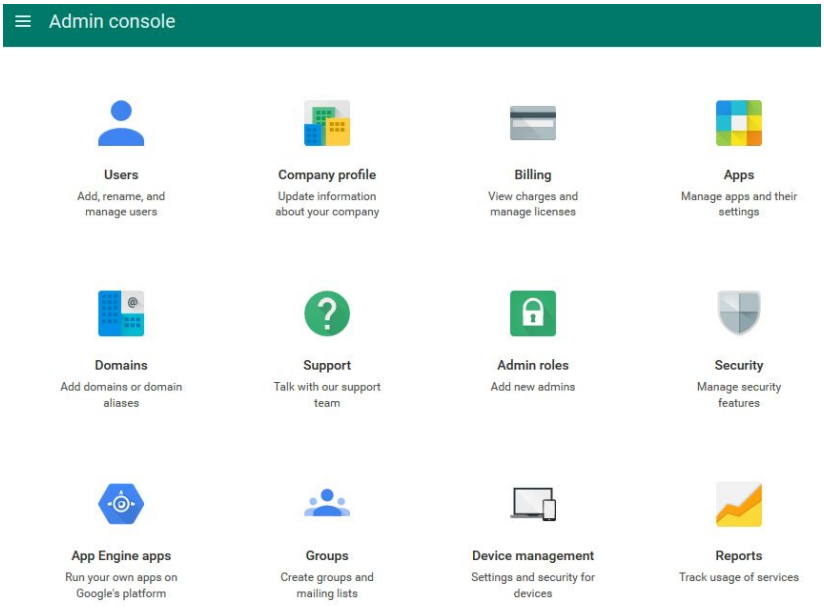
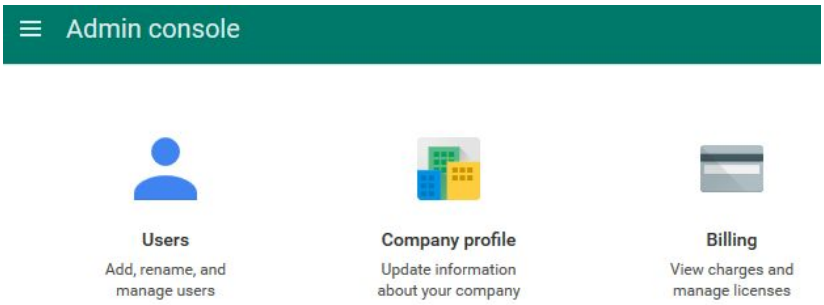
NOTE: We will show examples using a Google Apps Domain named gappslabs.com.

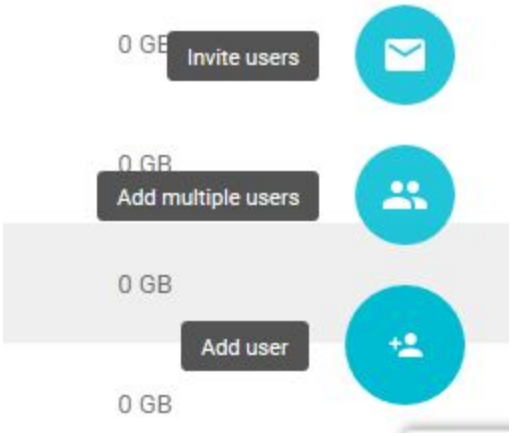
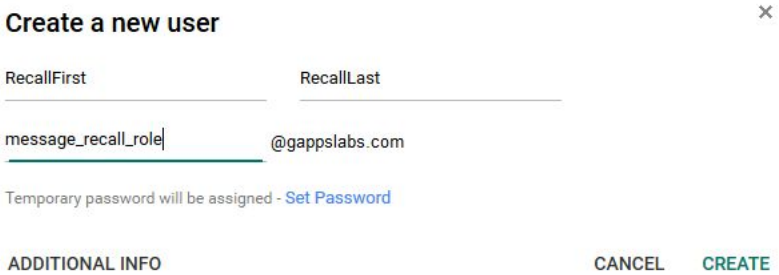
---

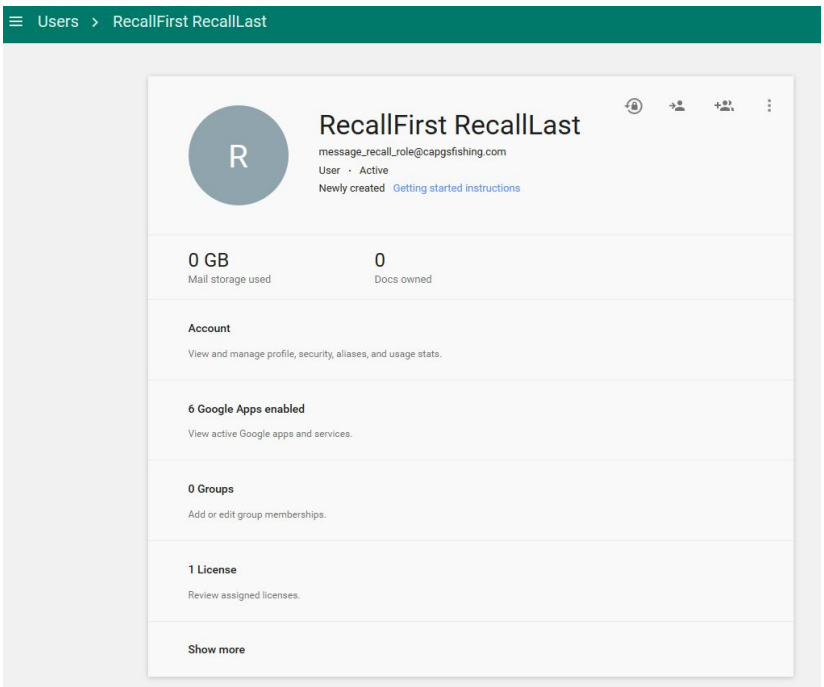
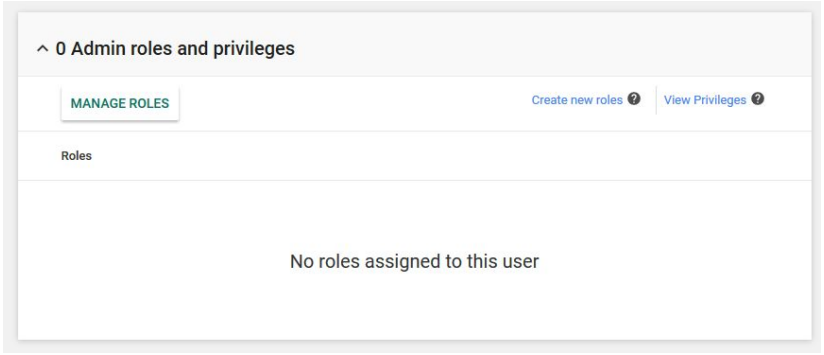
1. Create a Google Apps Domain Super-Admin ‘Role’ account

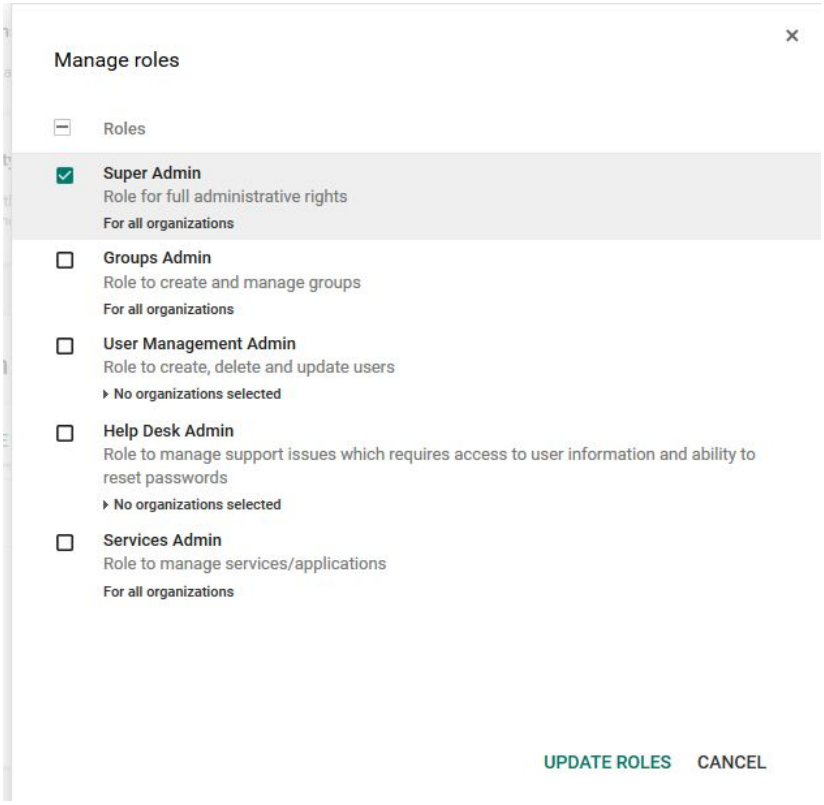
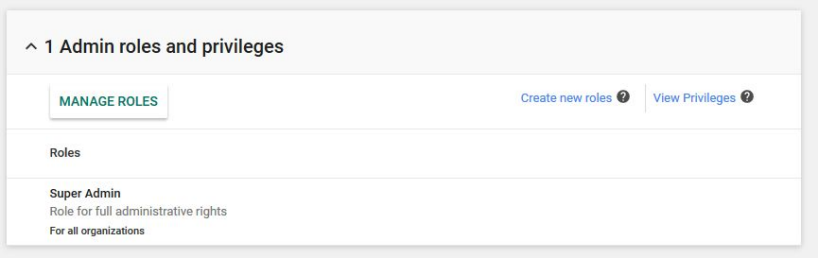
For additional reference, the following articles from the Google Support center discuss accounts and Admin roles:

- Adding users individually
- Pre-built administrator roles
- Assigning Administrator roles to users

In your Google Apps Domain, while logged in as a Domain Admin, navigate to the Admin console at: https://admin.google.com.	
Select ‘Users’ to create the Role account.	

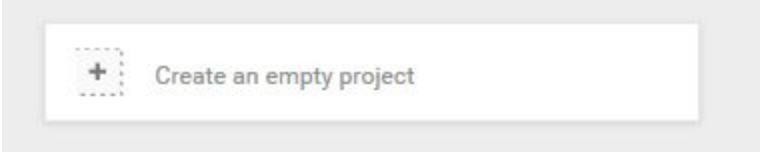
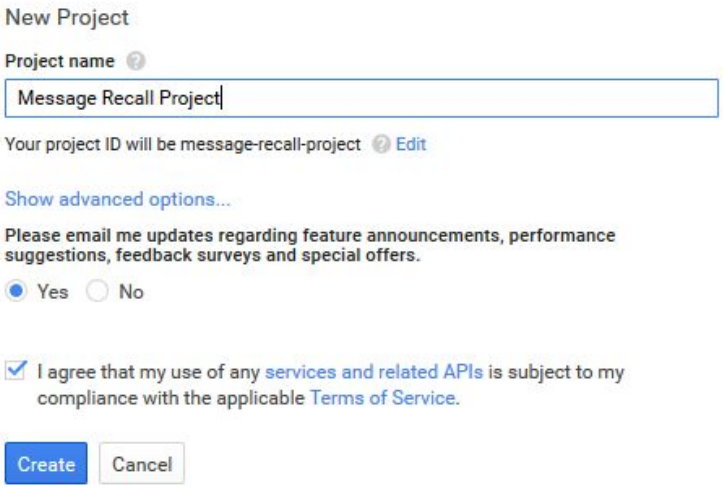
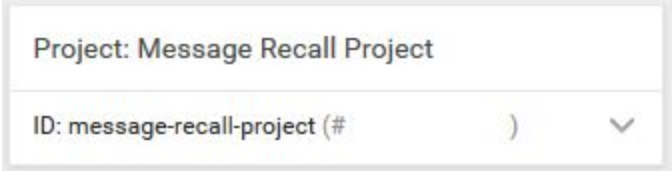
Add a new user by clicking ‘Add user’ that appears after clicking the + graphic. 	
Enter ‘First name’, ‘Last name’ and ‘Primary email address’ (we used message_recall_role for this example) then choose ‘Create’. On the next confirmation page, note the Password generated for the new account. You may choose to send email or print details. Choose Done.	

Select the new user, choose ‘Show more’ details.	
Choose: ‘Admin roles and its privileges’.	
Choose: ‘Manage roles’.	

Select: ‘Super Admin’ and choose ‘Update Roles’.	
You should notice your user now has ‘1’ next to ‘Admin roles and its privileges’.	
Sign out. Sign back in as message_recall_role @yourdomain.com.	

2. Create the App Engine hosting environment

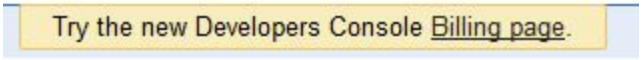
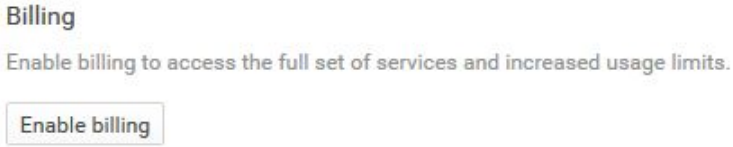
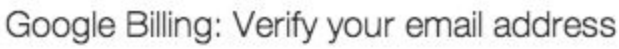
2.1 Create the application in App Engine

While logged in as message_recall_role, navigate to 'Google App Engine' at https://appengine.google.com.	
Choose 'Create an empty project'. (You may be requested to verify your account via text message or voice call.)	
Fill out the new project form. A Project name like 'Message Recall <yourdomainname>' is recommended. This maps to an application named message-recall-<yourdomain>. Select 'I agree...' Choose 'Create'.	
You should see the new Project.	
If you return to 'Google App Engine' at https://appengine.google.com you should see your new application project.	

2.2 Enable billing for your application

This application uses the IMAP mail API which uses sockets. To use sockets we must 'Enable Billing' in the application. Our use of the sockets is very small so any actual billing is very small.

However, Message Recall's use of backends is not free. It is **highly recommended** that you track your Billing Status on the application dashboard frequently. Running out of daily billing quota will cause your application to show a **OverQuotaError**.

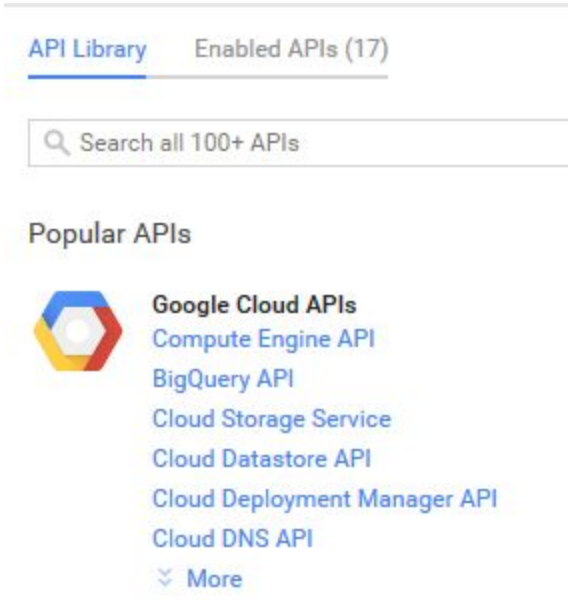
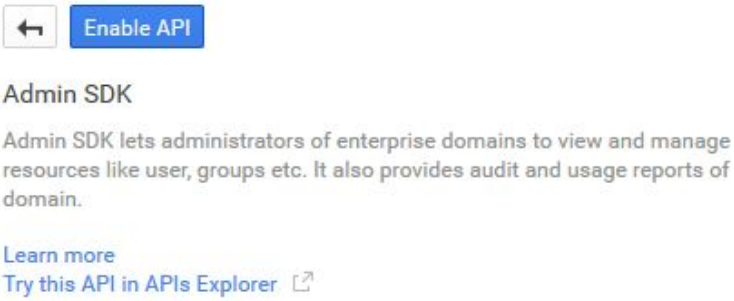
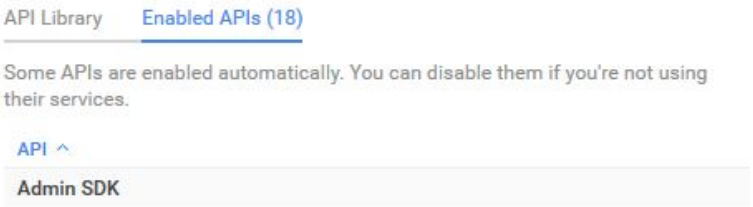
Choose 'message-recall-<yourdomain>' to see Application details.	
Now, under Billing, choose: 'Billing Status'.	
Now jump to the new Developers Console Billing page.	
Now choose 'Enable Billing'. You will need to establish a payment instrument (credit card) with a billing address. The default budgets are fine to start.	
You will need to approve the billing setup by clicking a link in the email inbox you use when you setup the payment instrument.	

The subject of the email should be ‘Google Billing: Verify your email address’ Click the link in the message body to verify your email.	

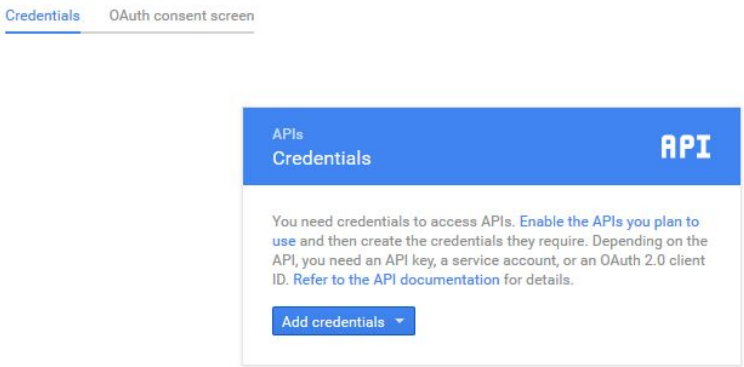
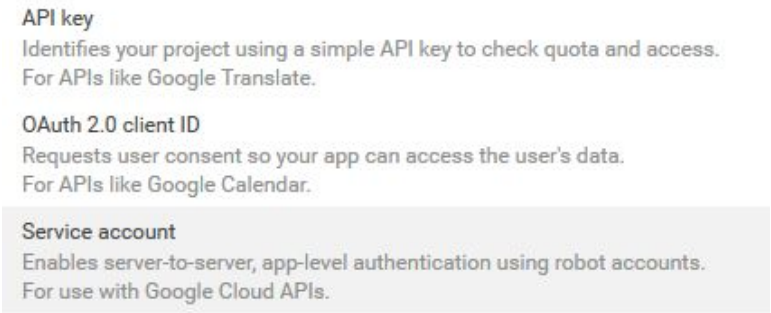
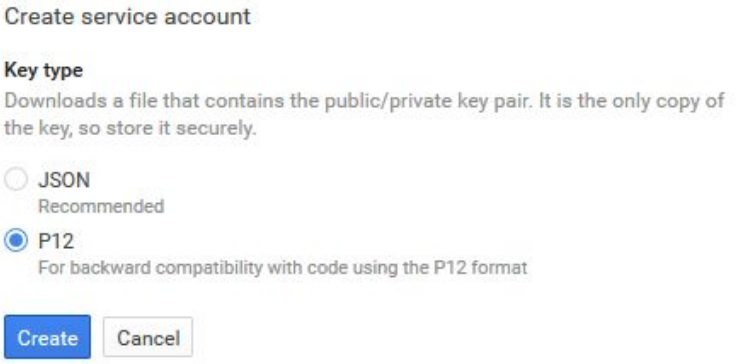
---

3. Configure the Google APIs Console Project

3.1 Enable the Admin SDK API

In the Developer Console for your project, choose: 'APIs & auth'. Then choose APIs. This will drop you into the APIs panel.	
Type 'Admin SDK' in the Search box. Choose 'Admin SDK' in the results. Choose 'Enable API' to enable this API.	
The 'Admin SDK' should now be listed in the 'Enabled APIs' list.	

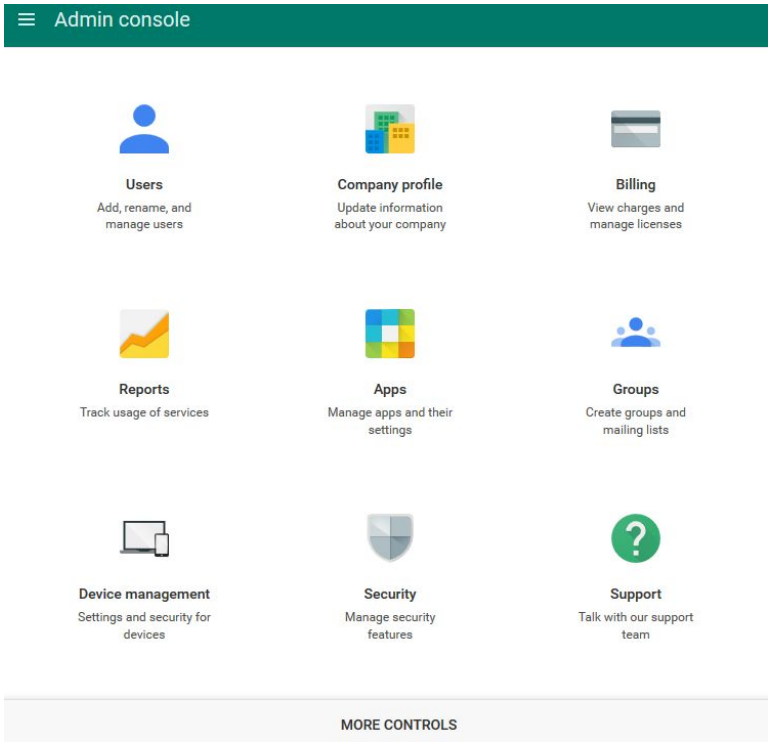
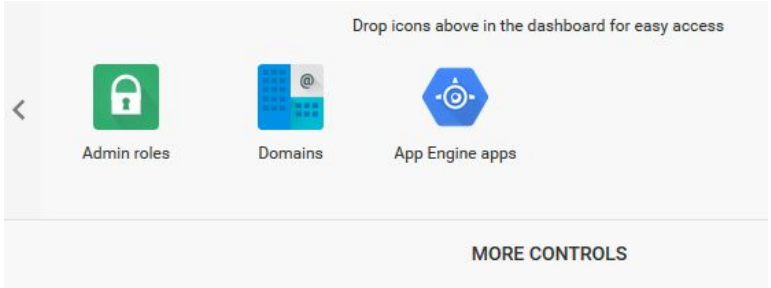
3.2 Create a Project Service Account

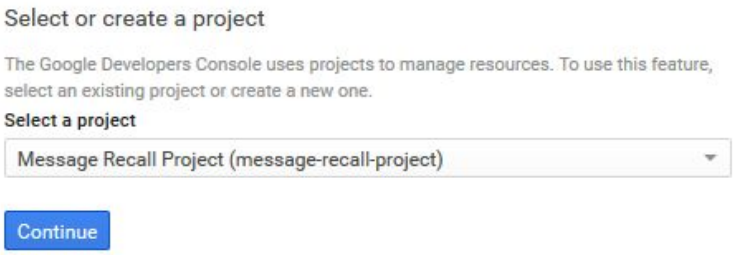
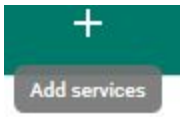
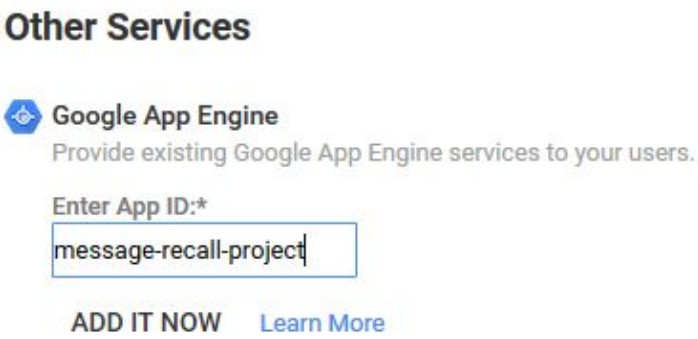
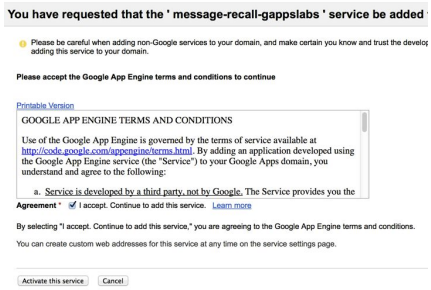
Choose ‘Credentials’ from the API Console Project. Then, choose ‘Add credentials’ to create your new OAuth 2.0 API application client.	
Choose ‘Service account’.	
Choose ‘P12’ for backward compatibility. Click ‘Create’.	

A file is generated and downloaded to your local system with your private key material. The file has a name of the form 'XXXX-privatekey.p12'. Choose 'Close' to proceed.	New public/private key pair **Message Recall Project-7768782e41c7.p12** has been downloaded to your computer and serves as the only copy of this key. Store it securely. **This is the private key's password. It will not be shown again. You must present this password to use the private key.** Learn more `notasecret` Close
Your OAuth 2.0 client information is now displayed in the Developer Console. Click the email address to see all details. Take special note of the 'Client ID' and 'Email address' as both will be needed later.	Service account Client ID: **xxxx.apps.googleusercontent.com** Email address: **xxxx@developer.gserviceaccount.com** Certificate fingerprints: **########** Done

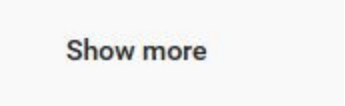
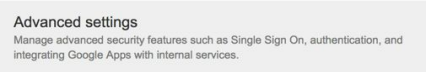
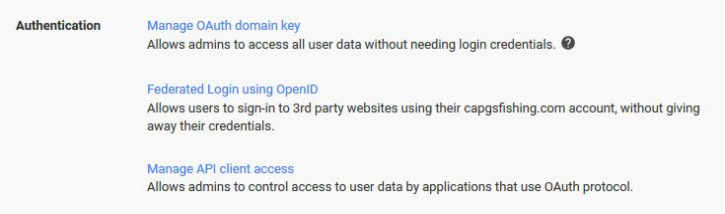
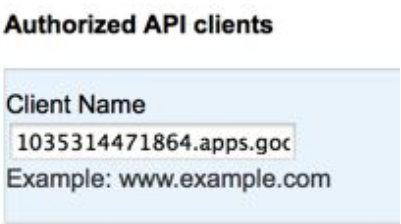
4. Configure your Google Apps Domain for the App Engine application

4.1 Add the App Engine application to the domain

Navigate back to the ‘Admin console’ for your domain at https://admin.google.com. Choose ‘More Controls’ at the bottom to view ‘App Engine Apps’.	
Choose ‘App Engine apps’ to view a list of registered applications.	
Jump to the ‘Google Developers Console’ to continue.	

Notice the project is ‘Message Recall Project’ and choose ‘Continue’.	
Click the ‘Add services’ + graphic on the top-right.	
In ‘Other Services’, enter the name of the app engine application created: ‘message-recall-<yourdomain>’. Choose ‘Add It now’.	
Select the box to activate the service ‘Agreement’. Choose ‘Activate this service’.	

4.2 Authorize your application in the Google Apps Domain

Return to the ‘Admin console’ and choose ‘Security’	
Choose ‘Show more’.	
Choose ‘Advanced settings’.	
Under ‘Authentication’ choose ‘Manage API client access’.	
In the ‘Client Name’ field, enter the Service Account ‘Client ID’ from step 3.2 earlier. It should be of the form: ##.apps.googleusercontent.com	
In the ‘One or More API Scopes’ box, enter the following scopes string: https://mail.google.com/ Choose ‘Authorize’	
You should notice an entry for your service account now listed.	**Email (Read/Write/Send)** https://mail.google.com/ **View users on your domain** https://www.googleapis.com/auth/admin.directory.user.readonly

5. Prepare the source code

5.1 Get the source code

Browse to the github.com page for [googleapps-message-recall]().	google / **googleapps-message-recall**
Click the 'Download ZIP' button.	Download ZIP
Create an empty folder for the project source code. Copy the downloaded googleapps-message-recall-master.zip file to the empty folder. Unzip the project source code.	`$ unzip ./googleapps-message-recall-master.zip`

Notice the source files under a new folder: googleapps-message-recall-master.	`$ ls -l ./googleapps-message-recall-master/`
	`-rw-r--r--@  1 adminuser  xxxxx  1697 Mar 18 11:29 README.md`
	`drwxr-xr-x@ 28 adminuser  xxxxx  952 Mar 18 11:29 message_recall`

5.2 Make local updates

In the extracted source, edit the app.yaml file.
 	
Change the 'application' from 'message_recall' to your app engine 'Application Identifier'.
 	
Save your new app.yaml file.
	From:
	`application: message-recall`
	
	To:
	`application: message-recall-gappslabs`

In the extracted source, edit the service_account.py file.
 	
Change the SERVICE_ACCOUNT_NAME to reflect the 'Service Account Email address' discovered in step 3.3.
	From:
	`SERVICE_ACCOUNT_NAME =`
	`('000000000000-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'`
	`'@developer.gserviceaccount.com')`
	
	To:
	`<your Service Account Email address>`

5.3 Create a service account certificate .pem file

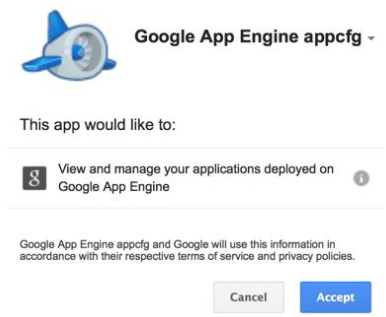
Find the file downloaded 'XXXX-privatekey.p12' file and copy it to your application folder.	`googleapps-message-recall-master/message_recall/`
Use the openssl tool to convert the PKCS12 certificate file to PEM format. Note: openssl is available on Linux and Windows Note the output filename of 'messagerecall_privatekey.pem' is required.	`$ openssl pkcs12 -in xxxx-privatekey.p12 -out messagerecall_privatekey.pem -nodes -nocerts`
When prompted for a password, supply the one presented earlier: *notasecret*.	`Enter the password: notasecret`
You should see the following output response and a new file created: messagerecall_privatekey.pem.	`MAC verified OK`
You can now remove the privatekey.p12 file.	`$ rm xxxx-privatekey.p12`
Edit the new messagerecall_privatekey.pem file. Delete the Bag Attributes lines. Delete the Key Attributes line. The .pem file should begin with the 'BEGIN PRIVATE KEY' line and end with the 'END PRIVATE KEY' line.	**Bag Attributes** **friendlyName: privatekey** **localKeyID: 54 69 6D 65 20 31 33 38 31 37 37 36 30 34 33 31 37 34** **Key Attributes: <No Attributes>** -----BEGIN PRIVATE KEY----- MIICdwIB…… -----END PRIVATE KEY-----
The edited file must be placed in the root source folder next to the app.yaml file.	`googleapps-message-recall-master/message_recall/`

6. Download the Google App Engine SDK

Find your platform, download and install the 'Google App Engine SDK for Python'	https://developers.google.com/appengine/downloads#Google_App_Engine_SDK_for_Python
Your installation process may be a single unzip or an executable installation program.	

---

7. Upload the source code into the AppEngine host

Upload the front-end code. From your application source root folder (the folder that contains app.yaml), run the appcfg.py command that was installed with the 'Google App Engine SDK for Python'. Authenticate as your message_recall_role@yourdomain.com user.	`$ appcfg.py --noauth_local_webserver update .`
When prompted, choose 'Accept' to allow the appcfg program to upload the source code to your app engine hosting environment.	

You should notice the following progress on your console confirming the initial application upload (deployment).	`Compilation starting.`
	`Compilation completed.`
	`Starting deployment.`
	`Checking if deployment succeeded.`
	`Deployment successful.`
	`Checking if updated app version is serving.`
	`Completed update of app:`
	`  message-recall-gappslabs, version: 1`
	`Uploading index definitions.`
	`Uploading task queue entries.`

Now update the queue setup.	`$ appcfg.py update_queues .`
Now upload the back-end code.	`$ appcfg.py backends . update`

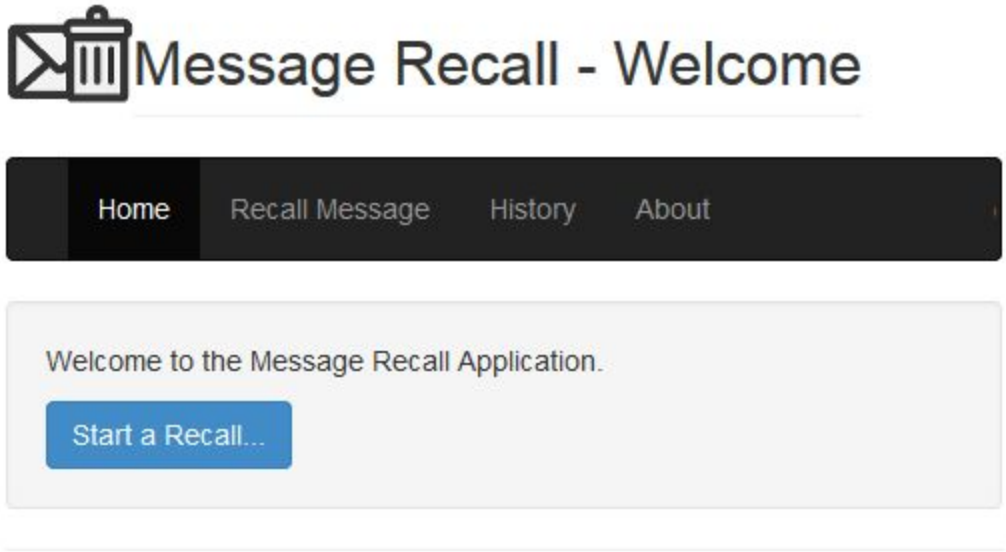
You should notice the following progress on your console confirming the deployment.	`Starting update of app: message-recall-gappslabs, backend: recall-backend`
	`Getting current resource limits.`
	`Scanning files on local disk.`
	`...`
	`Compilation starting.`
	`Compilation completed.`
	`Starting deployment.`

	Checking if deployment succeeded. Deployment successful. Checking if updated app version is serving. Completed update of app: message-recall-gappslabs, backend: recall-backend

---

8. Test the running application

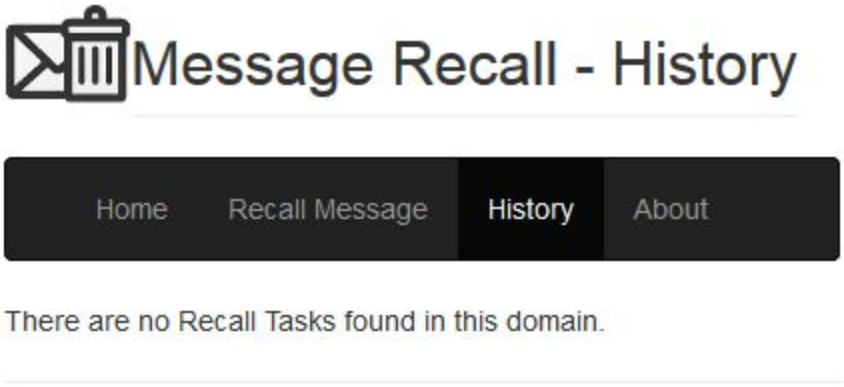
A. Navigate to the application and view the landing page:
https://message-recall-gappslabs.appspot.com



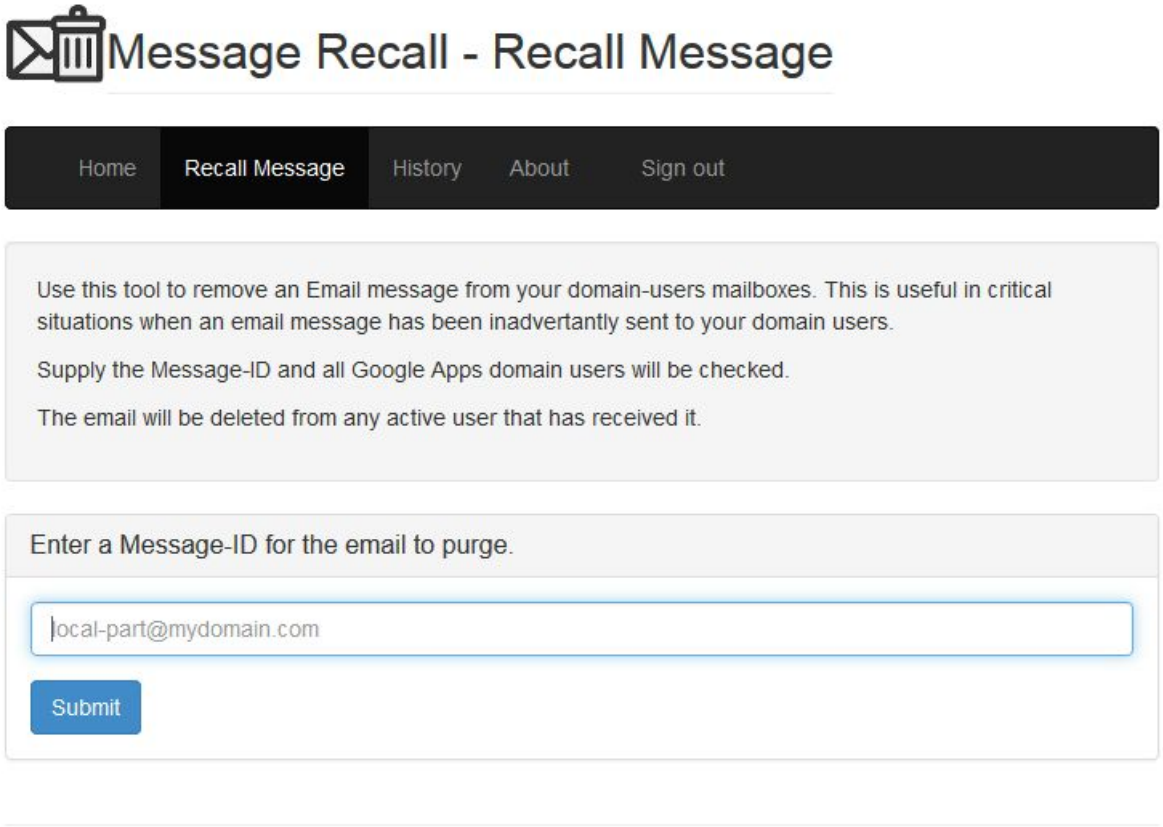
B. Navigate to the History page:

NOTE: The initial application data store indices need to be constructed. If you see the following message on the History page, initial index construction has not yet completed.

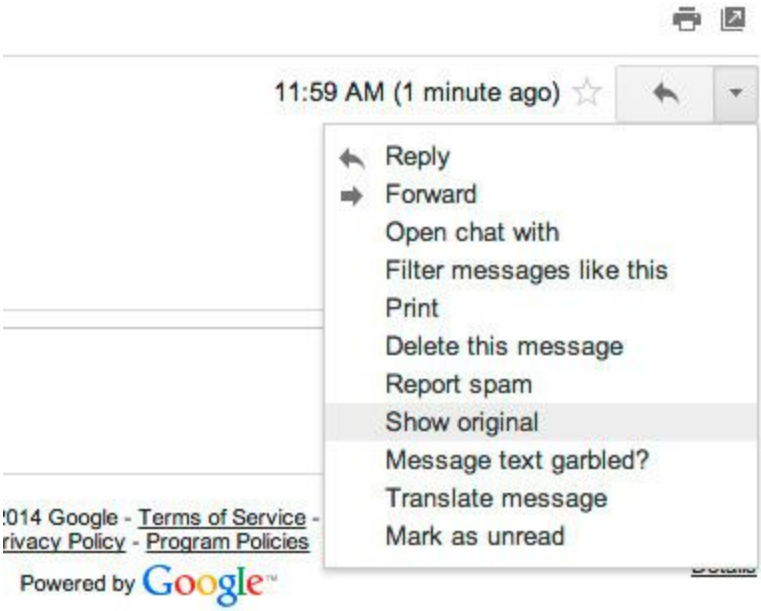
You can also check progress on initial index creation from the AppEngine administrator page under 'Datastore Indexes'.



C. Try recalling a message:



1. From Gmail, send a message to any domain user.
2. From your Gmail ‘Sent Mail’ folder, open the sent message.
3. You can see detailed information about your message by clicking the ‘Show original’.



4. Note the Message-ID of your sent message (it is the text inside of the <> brackets).
5. Navigate to the 'Recall Messages' page of your application.
6. Enter the Message-ID of your sent message in the textbox.
7. Click 'Submit'.
8. You are dropped into the Message Recall 'Task' page. Refresh it frequently to see status of your recall task.
9. You may choose the 'View Report' page to see a different view of progress.
10. The Task state starts at '*Getting Users*' and progresses through *'Recalling'* to *'Done'*.

11. When the application completes, you should notice the following on the 'Task' page if the message was only sent to 1 user. This indicates the user was identified, is not suspended, and the message was able to be deleted.

Message Recall - Task

Home Recall Message History About Sign out

Tip: Refresh frequently to see updates.

State	
Owner	
Message-ID	
Start (UTC)	20151024 10:20:49
Stop (UTC)	20151024 10:21:06
Elapsed (m:s)	
Total Users	
Users Processed	
Users with Messages Recalled	1

View Users View Report Debug Task

Appendices

Getting Help

If you need additional assistance setting up and/or configuring your application, please try the following resources:

Help with Google App Engine
The Google App Engine SDK for Python
Google Apps Documentation & Support: Sign in to your Admin console
Google APIs Console Help
The Google Apps Admin SDK
The googleapps-message-recall Google Group