

How to send an email using Gmail API?

Step-1) Create an account on Google Cloud Platform

Try Google Cloud for free

Step 1 of 2 Account Information

Anubhav Patrick
anubhav.patrick@jindia.com [SWITCH ACCOUNT](#)

Good news! You're eligible for an additional \$100.00 in Free Trial credits for a total of \$400.00. You'll receive these credits within 24 hours of completing signup.

Country
India

What best describes your organization or needs?
Please select
Startup

Terms of Service
☒ I have read and agree to the [Google Cloud Platform Terms of Service](#), [Supplemental Free Trial Terms of Service](#), and the terms of service of [any applicable services and APIs](#).
Required to continue

[CONTINUE](#)

Verify your card to get started
Your card is used to verify you're not a robot. Don't worry, it won't be charged until you manually upgrade to a paid account.

Access all Google Cloud products
Get everything you need to build and run your apps, websites and services, including Firebase and the Google Maps API.

\$300 credit for free
Put Google Cloud to work with \$300 in credit to spend over the next 90 days.

Enter the relevant details regarding to your account and complete the Google Cloud Account using your organization's email id.

Note: You do not have to provide your credit/debit card details. If it shows to provide the card details again and again, cut the window and retry.

Step-2) Create a new project called Gmail API Project

New Project

You have 19 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)
[MANAGE QUOTAS](#)

Project name *
Gmail API Project

Project ID: gmail-api-project-376506. It cannot be changed later. [EDIT](#)

Organization *
abesit.in

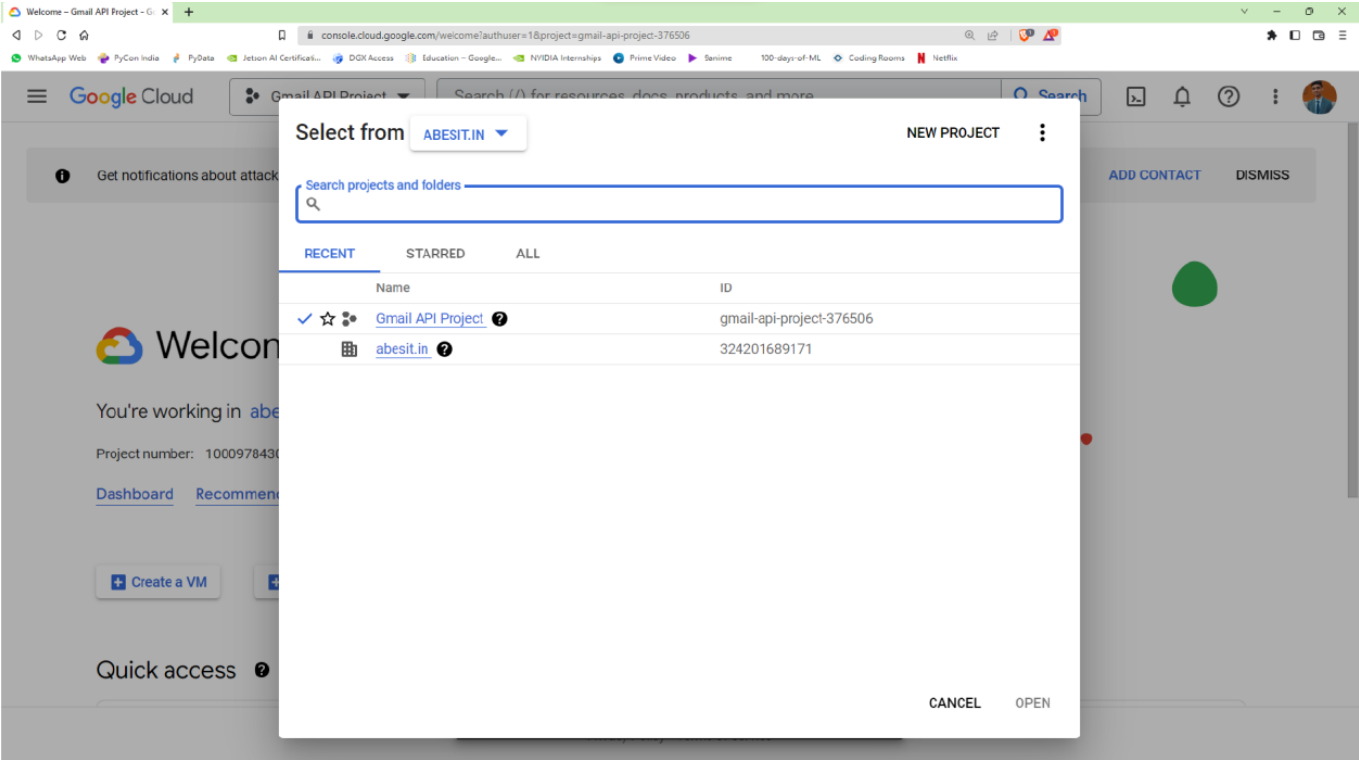
Select an organization to attach it to a project. This selection can't be changed later.

Location *
abesit.in [BROWSE](#)

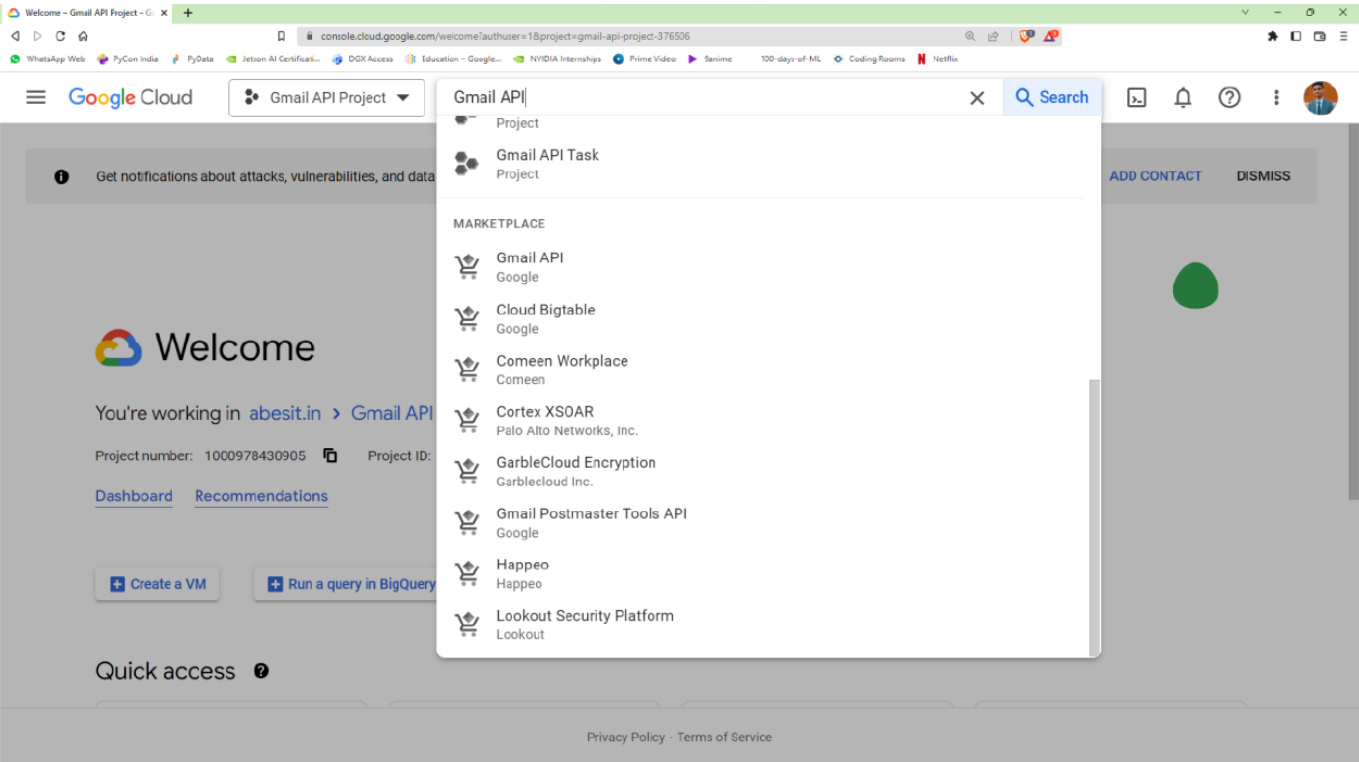
Parent organization or folder

[CREATE](#) [CANCEL](#)

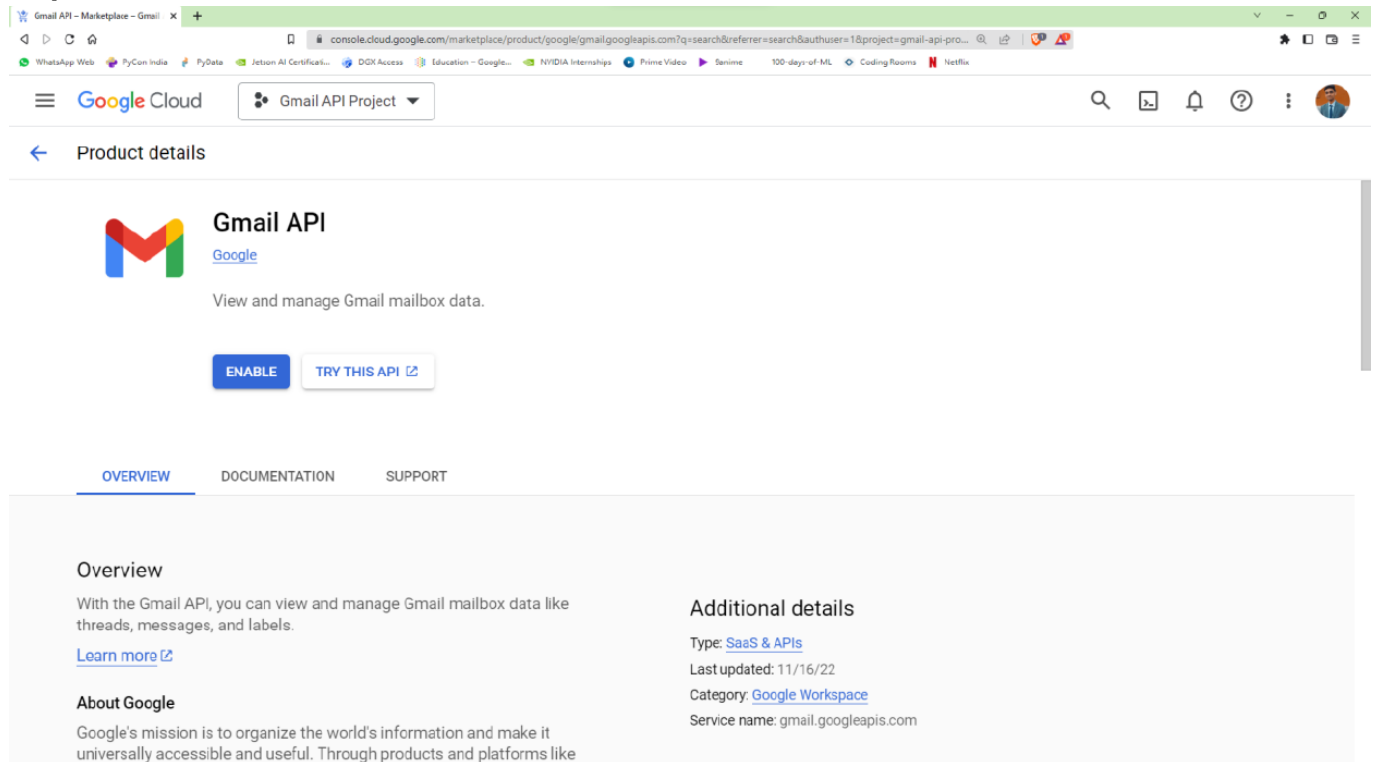
Step-3) Select the created project from the top pane.



Step-4) Type Gmail API in the search prompt and select Gmail API from Marketplace.



Step-5) Enable Gmail API



The screenshot shows the Google Cloud Marketplace page for the Gmail API. The page header includes the Google Cloud logo and a search bar. The main content area features the Gmail API logo and a description: "View and manage Gmail mailbox data." Below this, there are two buttons: "ENABLE" and "TRY THIS API". The page is divided into sections: "OVERVIEW", "DOCUMENTATION", and "SUPPORT". The "OVERVIEW" section contains an "Overview" paragraph, a "Learn more" link, and an "About Google" paragraph. The "Additional details" section lists the type as "SaaS & APIs", the last updated date as "11/16/22", the category as "Google Workspace", and the service name as "gmail.googleapis.com".

Gmail API
Google

View and manage Gmail mailbox data.

[ENABLE](#) [TRY THIS API](#)

OVERVIEW DOCUMENTATION SUPPORT

Overview

With the Gmail API, you can view and manage Gmail mailbox data like threads, messages, and labels.

[Learn more](#)

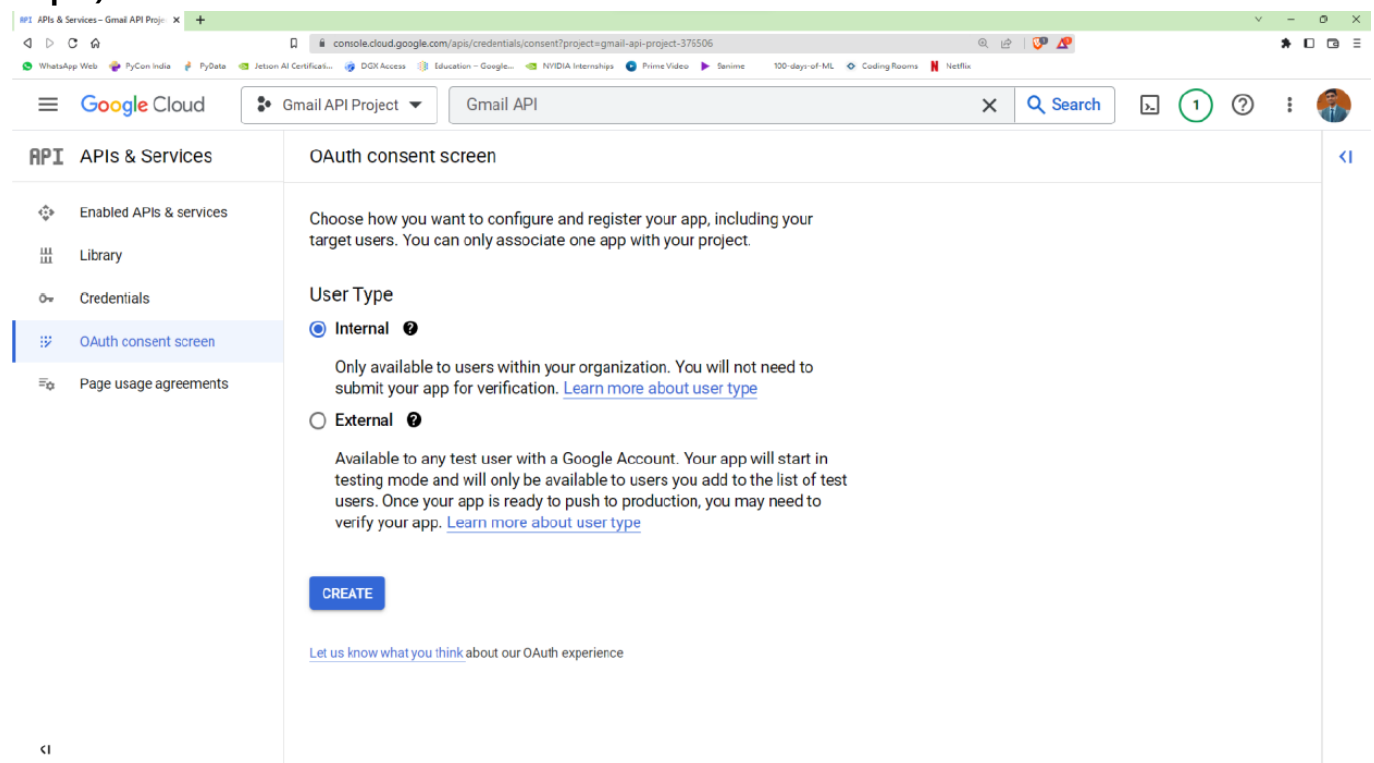
About Google

Google's mission is to organize the world's information and make it universally accessible and useful. Through products and platforms like

Additional details

Type: [SaaS & APIs](#)
Last updated: 11/16/22
Category: [Google Workspace](#)
Service name: gmail.googleapis.com

Step-6) Click OAuth Consent Screenshot from the side menu bar and select Internal.



The screenshot shows the Google Cloud OAuth consent screen configuration page. The left sidebar contains a menu with "APIs & Services" expanded, showing "Enabled APIs & services", "Library", "Credentials", "OAuth consent screen" (selected), and "Page usage agreements". The main content area is titled "OAuth consent screen" and contains instructions: "Choose how you want to configure and register your app, including your target users. You can only associate one app with your project." Below this, the "User Type" section has two radio buttons: "Internal" (selected) and "External". The "Internal" option is described as "Only available to users within your organization. You will not need to submit your app for verification. [Learn more about user type](#)". The "External" option is described as "Available to any test user with a Google Account. Your app will start in testing mode and will only be available to users you add to the list of test users. Once your app is ready to push to production, you may need to verify your app. [Learn more about user type](#)". At the bottom, there is a "CREATE" button and a link: "Let us know what you think about our OAuth experience".

APIs & Services

- Enabled APIs & services
- Library
- Credentials
- OAuth consent screen**
- Page usage agreements

OAuth consent screen

Choose how you want to configure and register your app, including your target users. You can only associate one app with your project.

User Type

☒ **Internal** ?

Only available to users within your organization. You will not need to submit your app for verification. [Learn more about user type](#)

☐ **External** ?

Available to any test user with a Google Account. Your app will start in testing mode and will only be available to users you add to the list of test users. Once your app is ready to push to production, you may need to verify your app. [Learn more about user type](#)

[CREATE](#)

[Let us know what you think about our OAuth experience](#)

Step-7) Click on the Create button. Then, fill the required fields as per your project concerns.

The screenshot shows the 'Edit app registration' page for the Gmail API in the Google Cloud console. The left sidebar shows the 'APIs & Services' menu with 'OAuth consent screen' selected. The main content area is titled 'App information' and contains the following fields:

- App name ***: gmail-app
- User support email ***: hamza2019cs148@abesit.edu.in
- App logo**: A section with a 'BROWSE' button to upload a logo file.

Step-8) Click Save and Continue, and finally Back to Dashboard. You will see a screen something like this.

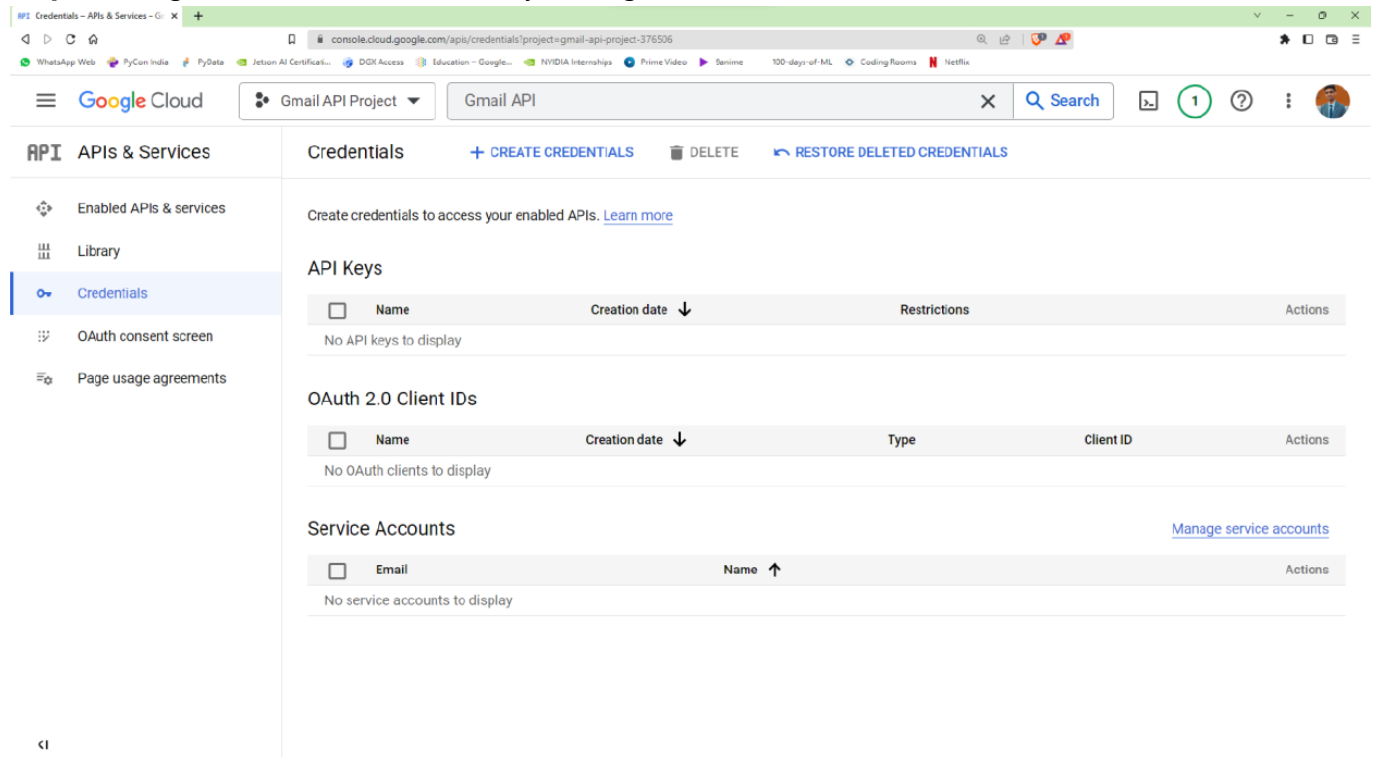
The screenshot shows the 'OAuth consent screen' page for the Gmail API in the Google Cloud console. The left sidebar shows the 'APIs & Services' menu with 'OAuth consent screen' selected. The main content area is titled 'hamza' and contains the following information:

- Publishing status**: Testing
- User type**: External
- OAuth user cap**: While publishing status is set to "Testing", only test users are able to access the app. Allowed user cap prior to app verification is 100, and is counted over the entire lifetime of the app. [Learn more](#)

The right sidebar lists various OAuth consent screen topics:

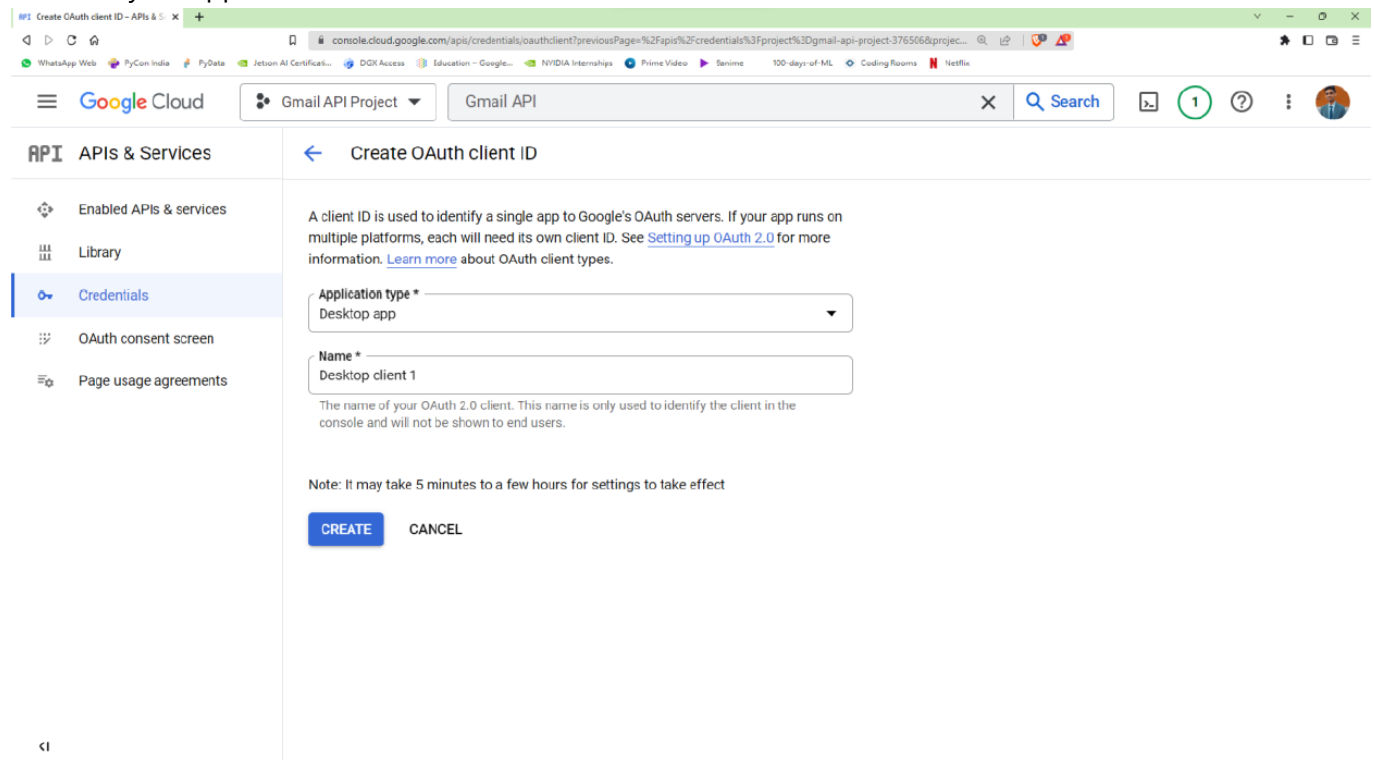
- Google OAuth consent screen
- What is the OAuth consent screen?
- What are OAuth consent scopes?
- What are sensitive API scopes?
- What are restricted API scopes?
- The app registration process
- What information do I need?
- Will my app need to be verified by Google?
- What if I don't verify my app?
- How long does the verification process take?

Step-9) Navigate to credentials window by clicking on Credentials menu from the sidebar.

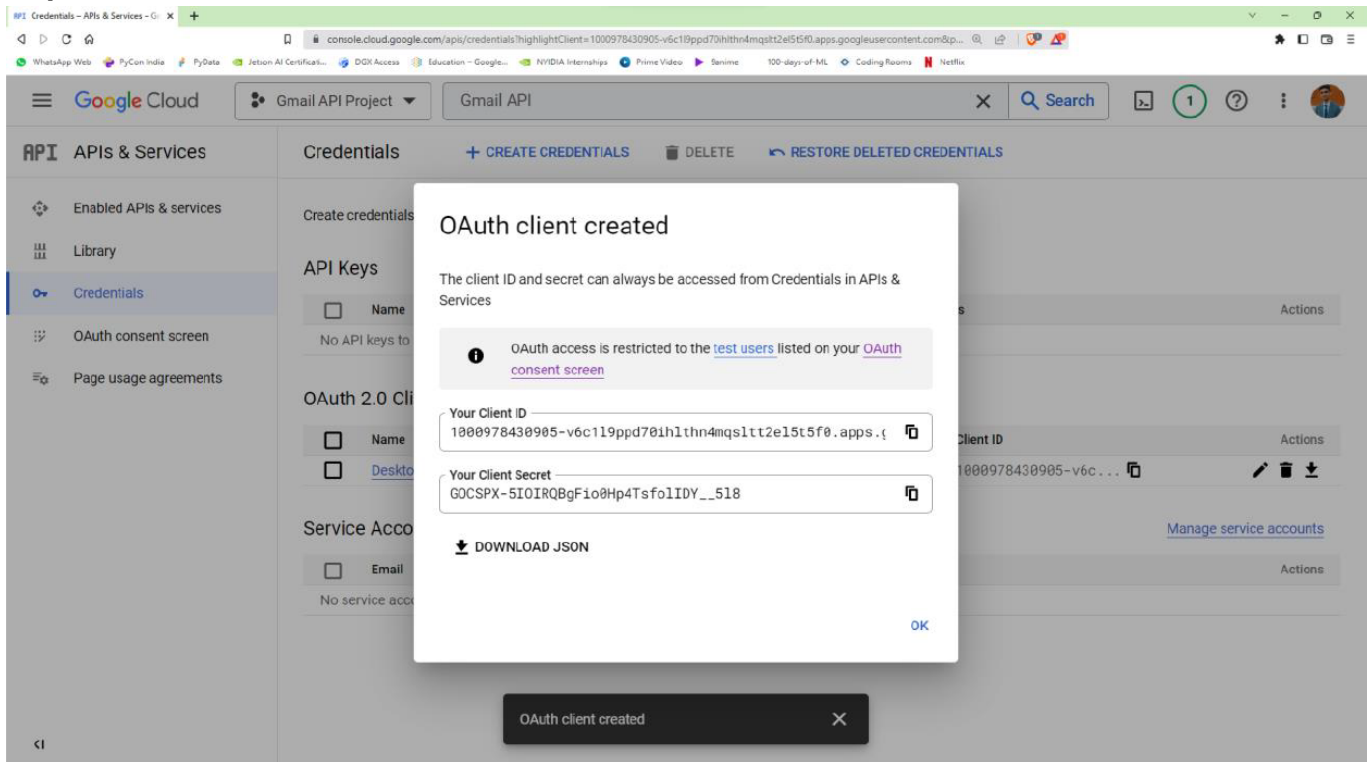


The screenshot shows the Google Cloud console interface. The left sidebar is expanded, showing the 'Credentials' menu item. The main content area is titled 'Credentials' and includes a search bar, a 'Gmail API' filter, and buttons for '+ CREATE CREDENTIALS', 'DELETE', and 'RESTORE DELETED CREDENTIALS'. Below these are three sections: 'API Keys' (showing 'No API keys to display'), 'OAuth 2.0 Client IDs' (showing 'No OAuth clients to display'), and 'Service Accounts' (showing 'No service accounts to display').

Step-10) Click on + CREATE CREDENTIALS and select OAuth Client ID. Select your application type and input name for your application.



The screenshot shows the 'Create OAuth client ID' form in the Google Cloud console. The left sidebar is expanded, showing the 'Credentials' menu item. The main content area is titled 'Create OAuth client ID' and includes a back arrow, a description of client IDs, and a form with two fields: 'Application type *' (set to 'Desktop app') and 'Name *' (set to 'Desktop client 1'). Below the form is a note: 'Note: It may take 5 minutes to a few hours for settings to take effect'. At the bottom are 'CREATE' and 'CANCEL' buttons.

Step-11) Click on Create button and Download JSON.**Step-12)** Now create a project directory.

```
mkdir gmail_api
cd gmail_api
```

Step-13) Create a virtual environment and activate it.

```
python3 -m venv venv/
source venv/bin/activate
```

Step-14) Install the Google client library for Python.

```
pip3 install --upgrade google-api-python-client google-auth-httpplib2 google-auth-
oauthlib
```

Step-15) Move the client secret (`client_secrets.json`) downloaded in step 8 to `gmail_api/` directory and do not share this file anywhere else since your Gmail account can be accessed by anyone using these credentials.

Step-16) Create a new python file inside `gmail_api/` titled `send_mail.py`

```
"""
This module sends emails with attachments to the participants
Reference - https://developers.google.com/gmail/api/quickstart/python
```

```

"""

import base64
import mimetypes
import os
from email.message import EmailMessage

from google.auth.transport.requests import Request
from google.oauth2.credentials import Credentials
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError

# If modifying these scopes, delete the file token.json.
SCOPES = ['https://www.googleapis.com/auth/gmail.send']

def authentication():
    credentials = None

    # The file token.json stores the user's access and refresh tokens, and is
    # created automatically when the authorization flow completes for the time.
    if os.path.exists('token.json'):
        credentials = Credentials.from_authorized_user_file(
            'token.json',
            SCOPES
        )

    # If there are no valid credentials available, let the user log in.
    if not credentials or not credentials.valid:
        if credentials and credentials.expired and credentials.refresh_token:
            credentials.refresh(Request())
        else:
            flow =
InstalledAppFlow.from_client_secrets_file('client_secrets.json', SCOPES)
            credentials = flow.run_local_server(port=0)

        # Save the credentials for the next run
        with open('token.json', 'w') as token:
            token.write(credentials.to_json())

    return credentials

def prepare_and_send_email(recipient, subject, message_text, attachment):
    """
    Prepares and send email with attachment to the participants.
    :param attachment: file path of the attachment that you want to send.
    :param recipient: email-id of the recipient
    :param subject: subject of the email
    :param message_text: body of the email
    """

    credentials = authentication()

```

```

try:
    # Call the Gmail API
    service = build(serviceName='gmail', version='v1',
credentials=credentials)

    # create message
    message = create_message('hamza2019cs148@abesit.edu.in', recipient,
subject, message_text, attachment)
    send_message(service, 'me', message)
except HttpError as error:
    # TODO(developer) - Handle errors from gmail API
    print(f"An error occurred: {error}")

def create_message(sender, to, subject, message_text, attachment):
    """
    Create a message for an email.
    :param attachment: file path of the attachment that you want to send.
    :param sender: Email address of the sender.
    :param to: Email address of the receiver.
    :param subject: The subject of the email message.
    :param message_text: The text of the email message.
    :return: An object containing a base64url encoded email object.
    """

    # create gmail api client
    mime_message = EmailMessage()

    # headers
    mime_message['From'] = sender
    mime_message['To'] = to
    mime_message['Subject'] = subject

    # text
    mime_message.set_content(message_text)

    # attachment
    attachment_filename = attachment
    # guessing the MIME type
    type_subtype, _ = mimetypes.guess_type(attachment_filename)
    maintype, subtype = type_subtype.split('/')

    with open(attachment_filename, 'rb') as fp:
        attachment_data = fp.read()
    mime_message.add_attachment(attachment_data, maintype, subtype,
filename=attachment_filename)

    return {'raw': base64.urlsafe_b64encode(mime_message.as_bytes()).decode()}

def send_message(service, user_id, message):
    """
    Send an email message.

```



```

:param service: Authorized Gmail API service instance.
:param user_id: User's email address. The special value 'me' can be used to
indicate the authenticated user.
:param message: Message to be sent.
:return: Sent Message.
"""

try:
    message = (service.users().messages().send(userId=user_id,
body=message).execute())
    print(f"Message Id: {message['id']}")
    return message
except HttpError as error:
    # TODO(developer) - Handle errors from gmail API
    print(f"An error occurred: {error}")

if __name__ == "__main__":
    prepare_and_send_email('hamzaaziz822@gmail.com', 'Greeting from Hamza Aziz',
'This is a test email', 'photo.jpg')

```

Step-17) Run the python file.

```
python3 send_mail.py
```

At the first time you will be prompted to enter you Google account credentials. Use the same credentials that you used to sign up in Google Cloud.

