Αυτοματοποιημένη Διαχείριση Συστημάτων

Διάλεξη 8η

Θωμάς Καμαλάκης

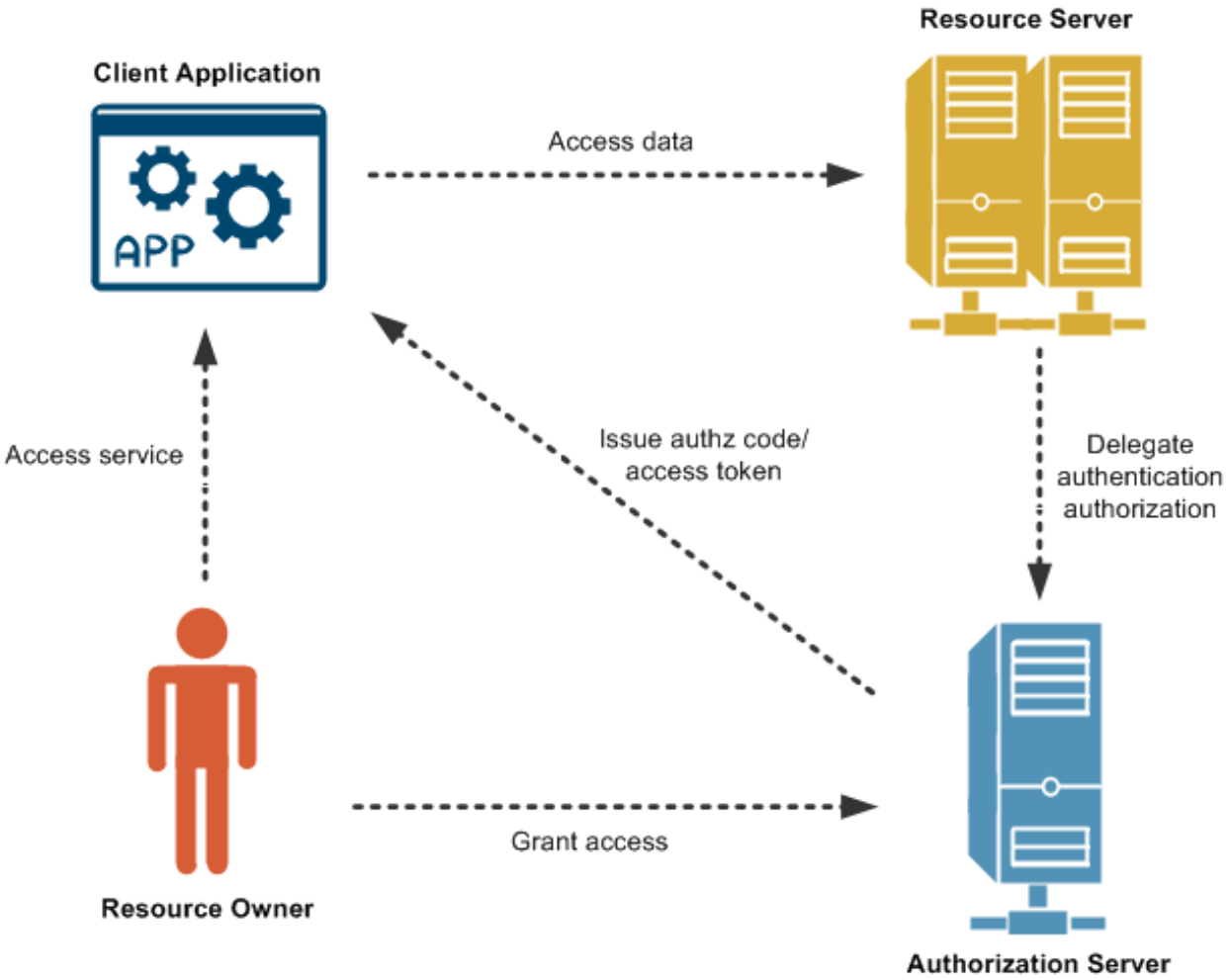
Χαροκόπειο Πανεπιστήμιο Αθηνών

Οκτώβριος 2022

OAuth2.0

- “Open Authorization”
- Επιτρέπει την πρόσβαση σε πόρους από υπηρεσίες εκ μέρους ενός χρήστη από μία εφαρμογή.
- Ο χρήστης θα πρέπει να δώσει την συγκατάθεση του.
- Τότε η εφαρμογή μπορεί να έχει οριοθετημένη πρόσβαση στους πόρους χωρίς να γνωρίζει τους κωδικούς πρόσβασης του χρήστη.

OAuth2.0


Resource Server
Client Application
APP
Access data
Access service
Issue authz code/
access token
Delegate
authentication
authorization
Grant access
Resource Owner
Authorization Server

OAuth2.0

- Resource server: εκεί που υπάρχουν οι πόροι τους οποίους θέλουμε να χρησιμοποιήσουμε.
- Resource owner: αυτός που έχει την εξουσιοδότηση να δώσει πρόσβαση στους πόρους.
- Client application: η εφαρμογή που θέλει να έχει πρόσβαση στους πόρους
- Authorization server: αυτός που εκδίδει κουπόνια που επιτρέπουν την πρόσβαση αφού δώσει την συγκατάθεση του ο owner.
- Scope: καθορίζει ακριβώς τι μπορεί να κάνει η εφαρμογή πάνω στους πόρους που διαχειρίζεται.

OAuth2.0: τα υπέρ

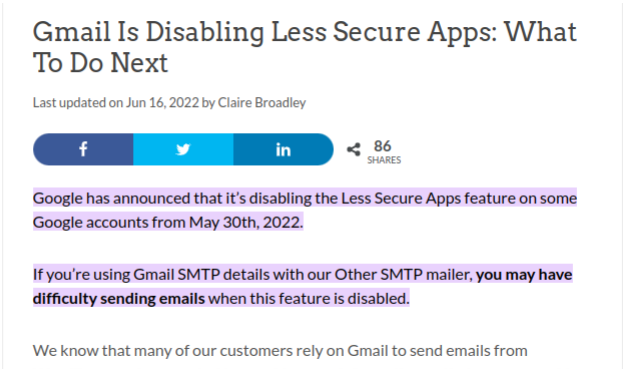
- Χρησιμοποιεί SSL για την κωδικοποίηση της επικοινωνίας
- Δίνει περιορισμένη πρόσβαση (Scope)
- Συνδυάζεται με λύσεις single-sign-on
- Υλοποιείται με https οπότε μπορεί να εφαρμοστεί σε mobile, desktop, web εφαρμογές.
- Αρκετό documentation
- Αρκετές έτοιμες βιβλιοθήκες
- Ήδη χρησιμοποιείται από τους “γίγαντες” (Google, κτλ)
- Δεν εκθέτει τους κωδικούς του χρήστη

Είδη Google API

Τι κάνουν τα Google API

Το API το gmail

- Πως στέλνουμε ένα e-mail από ένα Python script;
- Ο παραδοσιακός τρόπος είναι να χρησιμοποιήσουμε την βιβλιοθήκη smtplib.
- Ωστόσο με τον τρόπο αυτό θα πρέπει να βάλουμε κάπου το κωδικό που έχει ο λογαριασμός στον mail server.
- Η Google (πλέον) δεν υποστηρίζει τον τρόπο αυτό (τουλάχιστον στους standard λογαριασμούς)


Gmail Is Disabling Less Secure Apps: What To Do Next

Last updated on Jun 16, 2022 by Claire Broadley

86 SHARES

Google has announced that it's disabling the Less Secure Apps feature on some Google accounts from May 30th, 2022.

If you're using Gmail SMTP details with our Other SMTP mailer, **you may have difficulty sending emails** when this feature is disabled.

We know that many of our customers rely on Gmail to send emails from

Πως στέλνουμε e-mail με το Google API

1. Θα πρέπει να ξεκινήσουμε ένα νέο project στο Google Cloud Platform (GCP)
2. Ανοίγουμε εκεί το Gmail API
3. Φτιάχνουμε τα διαπιστευτήρια για την εφαρμογή μας (το script μας δηλαδή)
4. Ενσωματώνουμε στο script μας τη ροή του Oauth2.0
5. Την πρώτη φορά που τρέχει το script μας ζητάει την συγκατάθεση μας να χρησιμοποιήσει το email μας.
6. Η εφαρμογή μας λαμβάνει ένα token το οποίο μπορεί να ανανεώνει και πλέον μπορεί να στένλει e-mail εκ μέρους μας.

Disclaimer

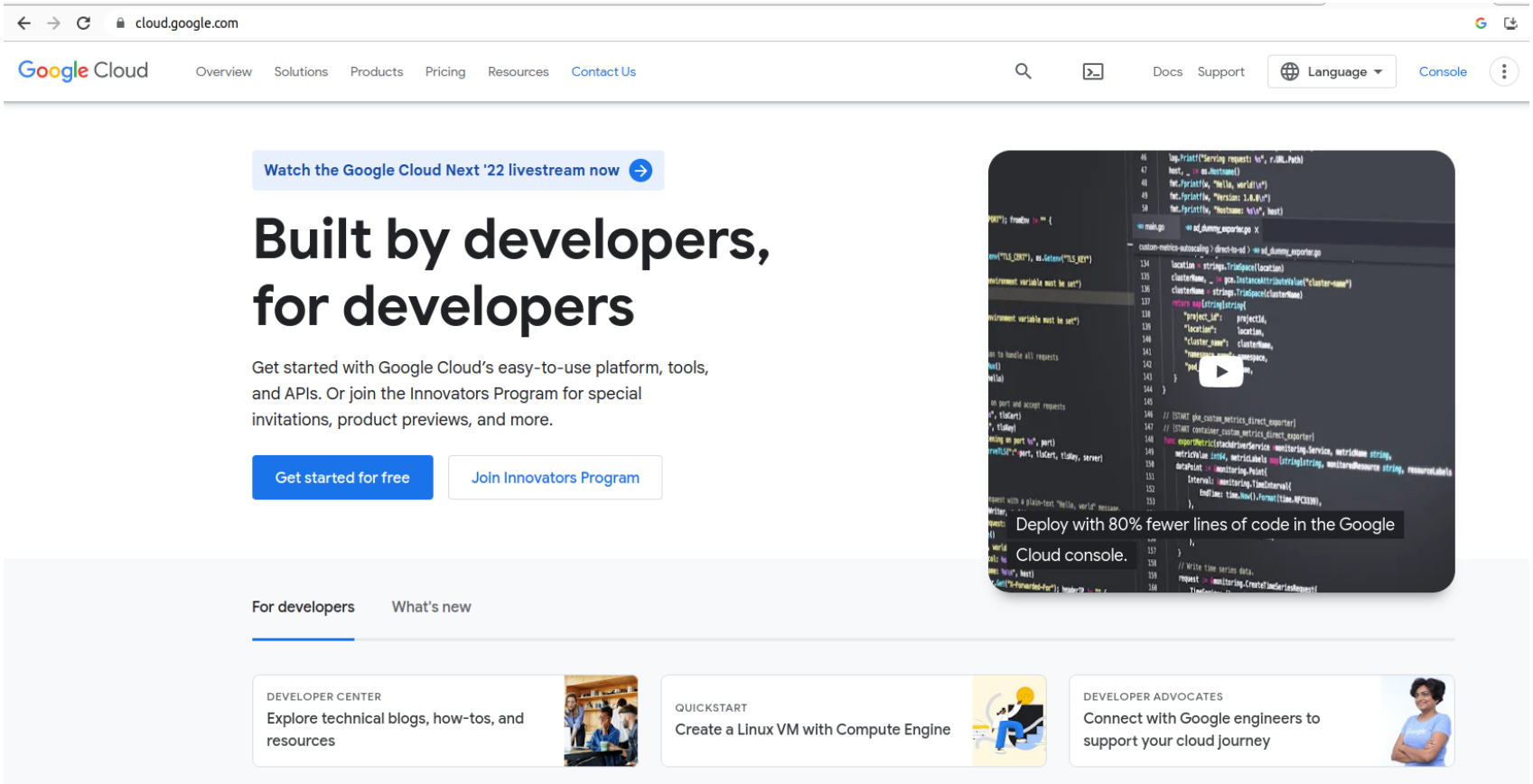
- Προσοχή!
- Με τα παρακάτω παραδείγματα θα δώσουμε στον κώδικα μας πλήρη έλεγχο πάνω στο @gmail.com λογαριασμό μας.
- Όπως είμαστε προσεκτικοί με τα κλειδιά μας έτσι είμαστε και με τα κουπόνια μας.
- Π.χ. δεν τα ανεβάζουμε σε repos, δεν τα αφήνουμε σε server που δεν ελέγχουμε πλήρως κτλ.
- Καλύτερα να χρησιμοποιήσετε ένα δοκιμαστικό λογαριασμό αν το περιεχόμενο του γραμματοκιβωτίου είναι σημαντικό για εσάς.

Disclaimer

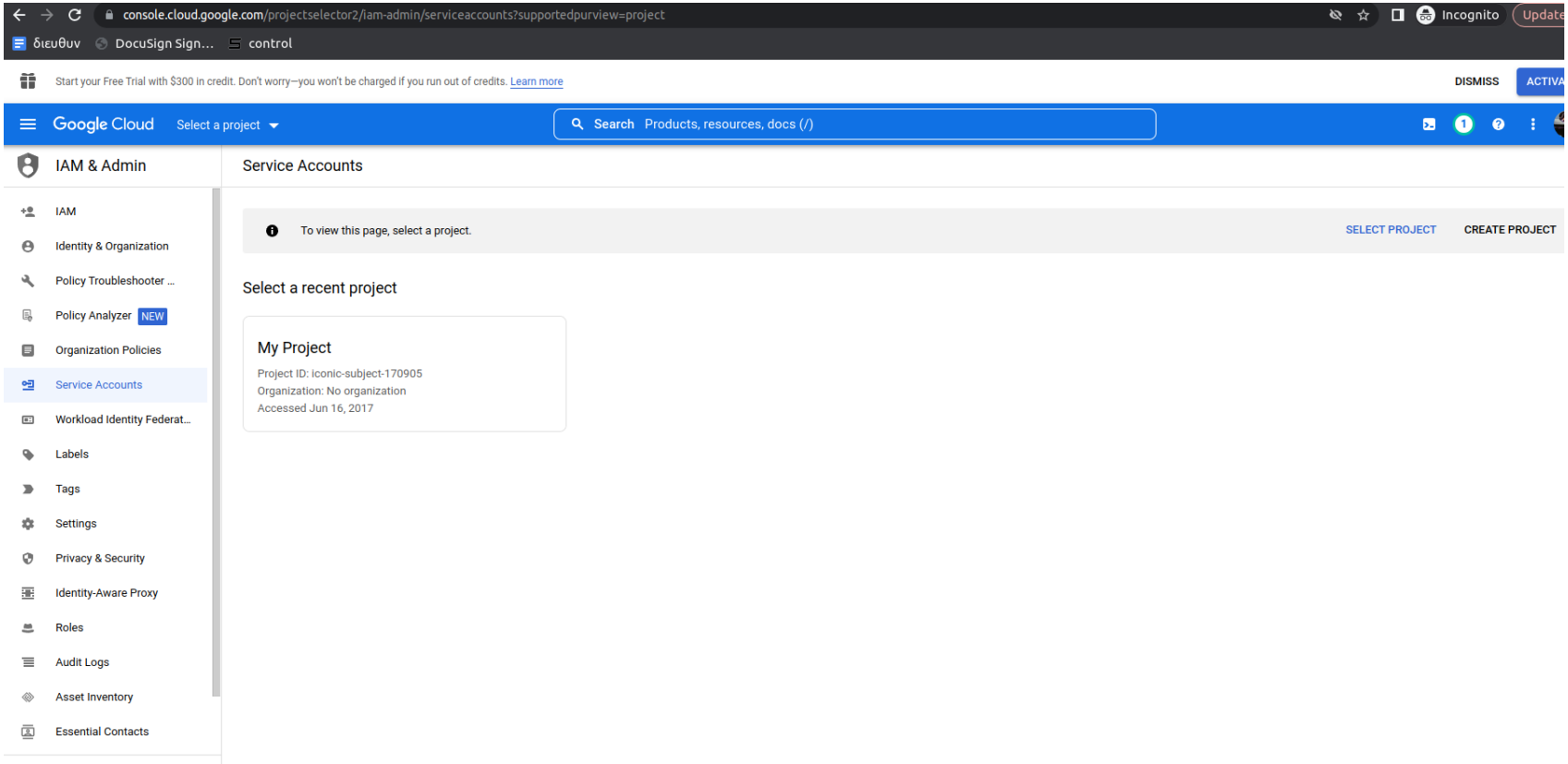

WHAT ARE YOU DOING ON THE DARK WEB??
I FORGOT MY PASSWORD, SO I'M LOOKING IT UP.
CYBERSECURITY VENTURES
CYBERSECURITYVENTURES.COM
Bathstein/MORGAN 2021

Disclaimer

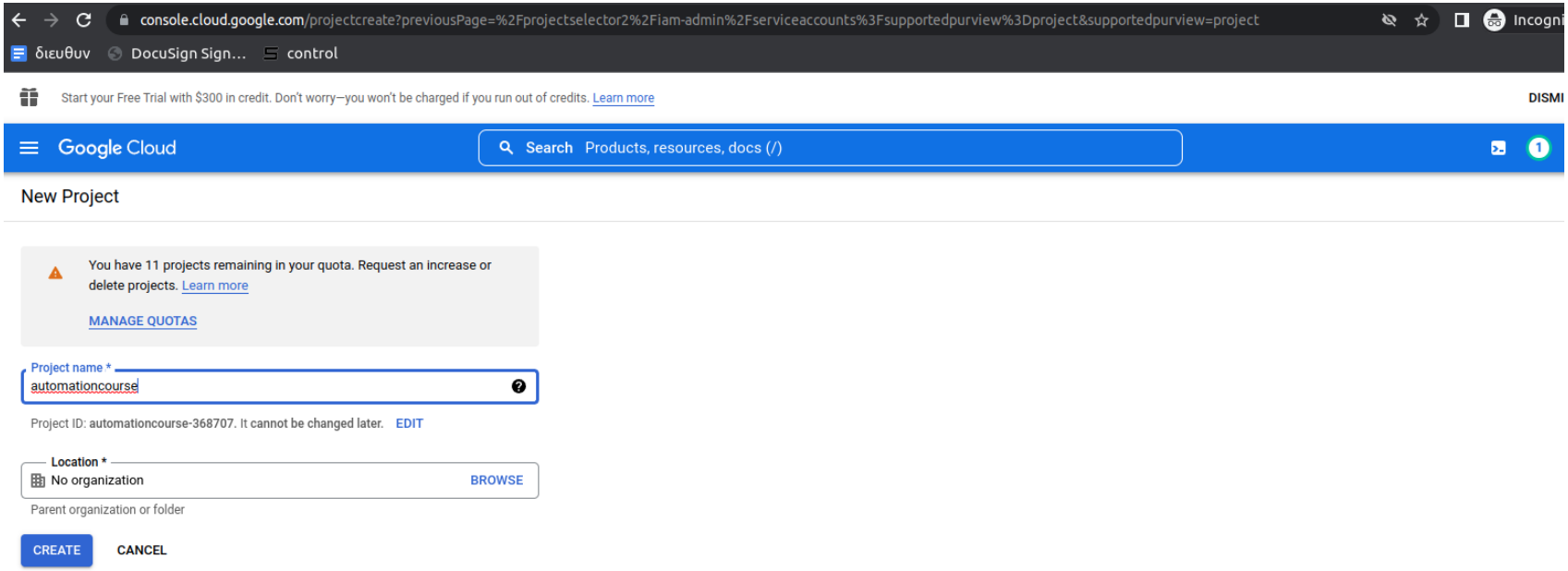
Ξεκινάμε από το GCP



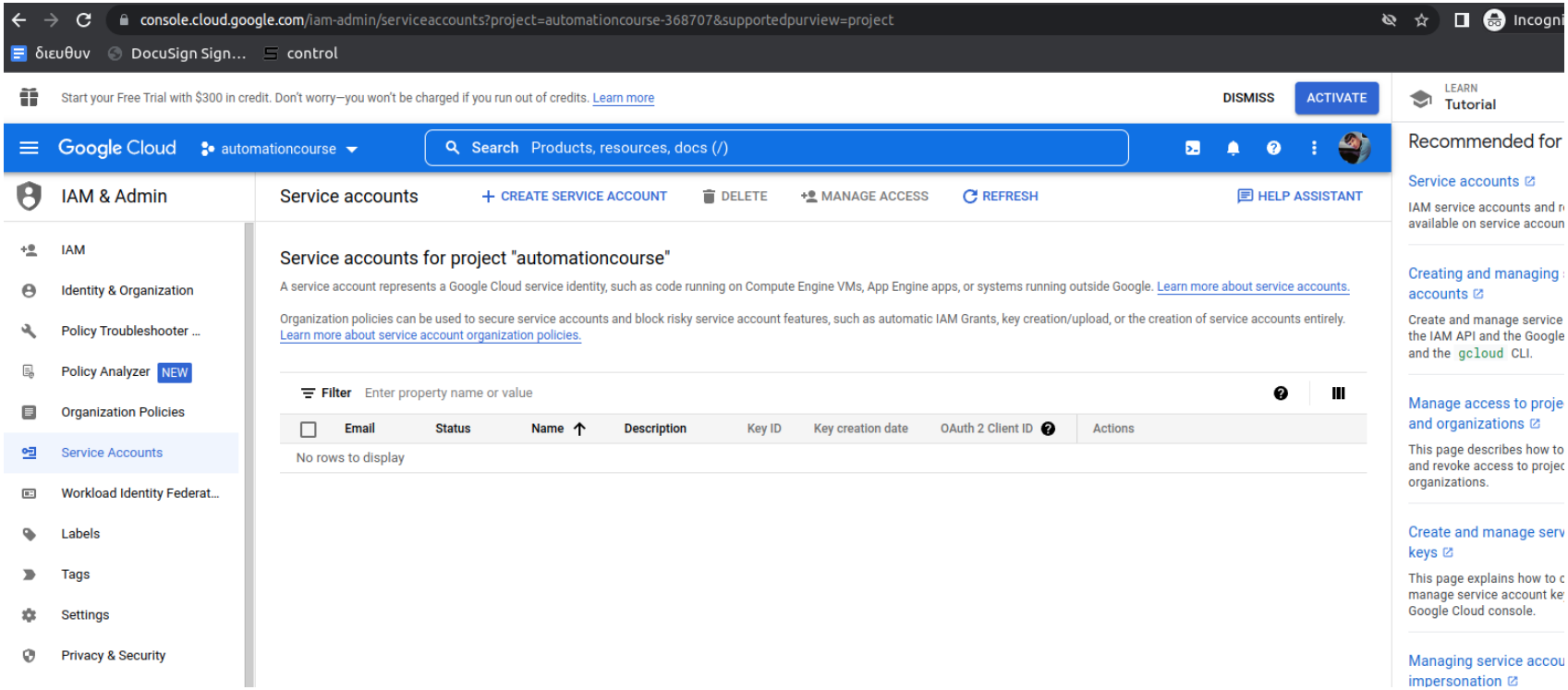
Δημιουργούμε έναν client



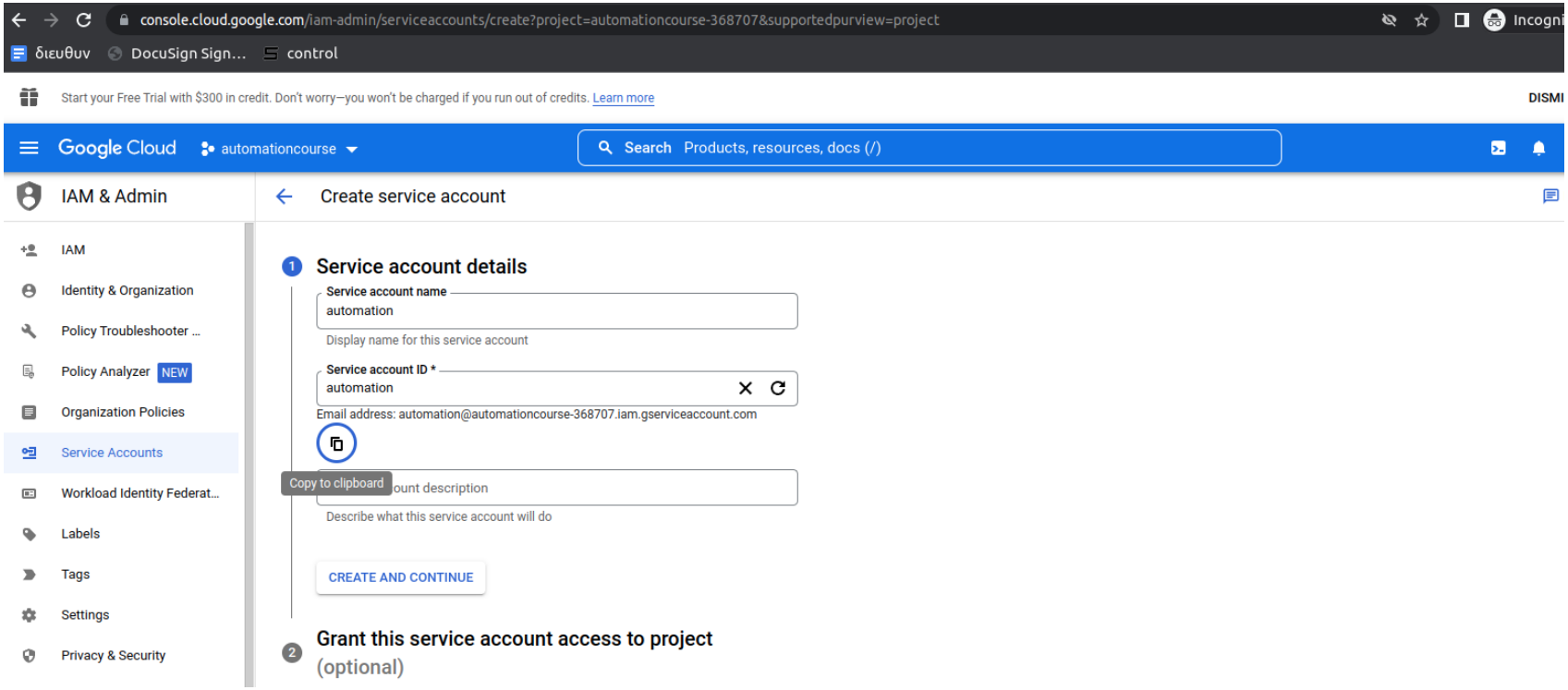
Δημιουργούμε έναν client



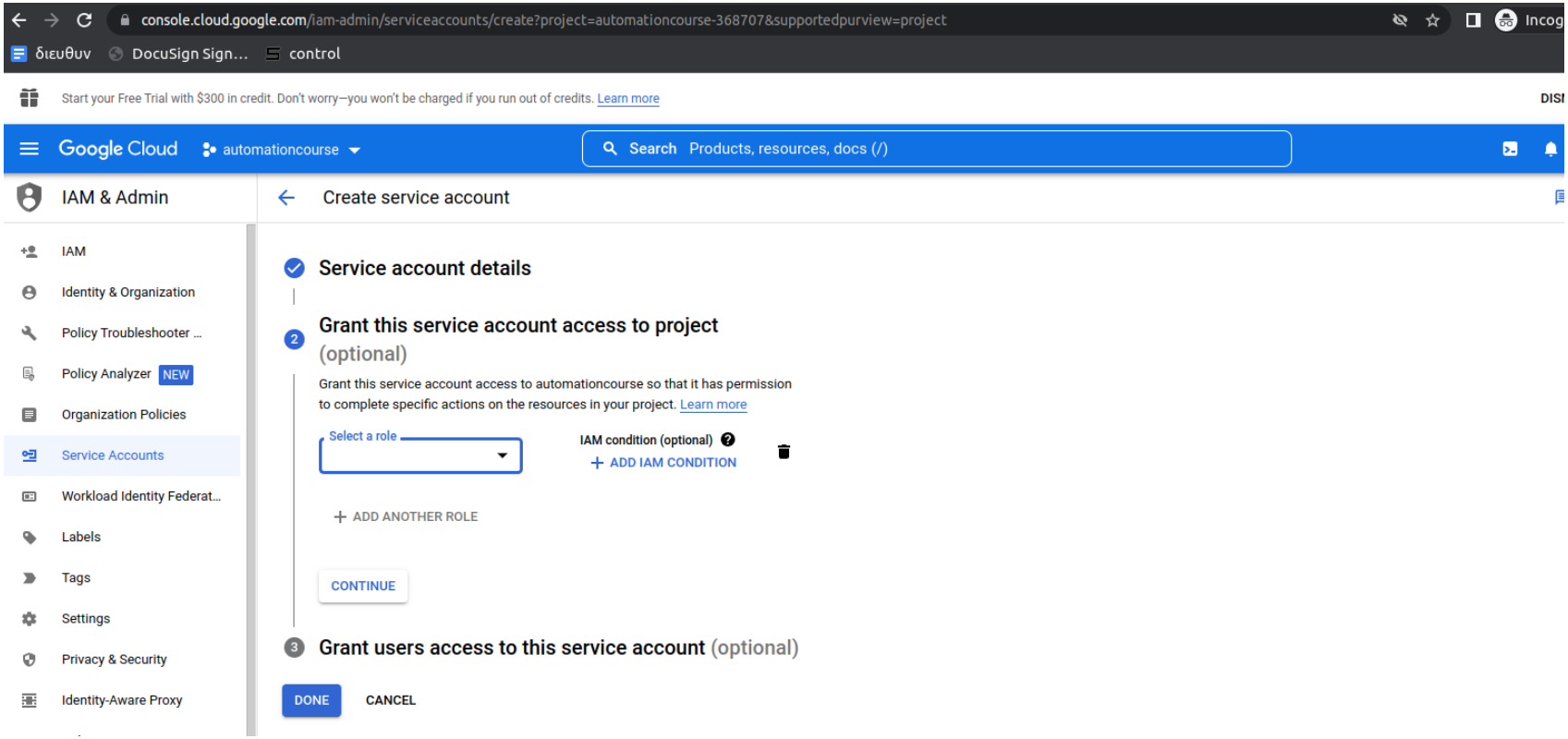
Δημιουργούμε έναν client



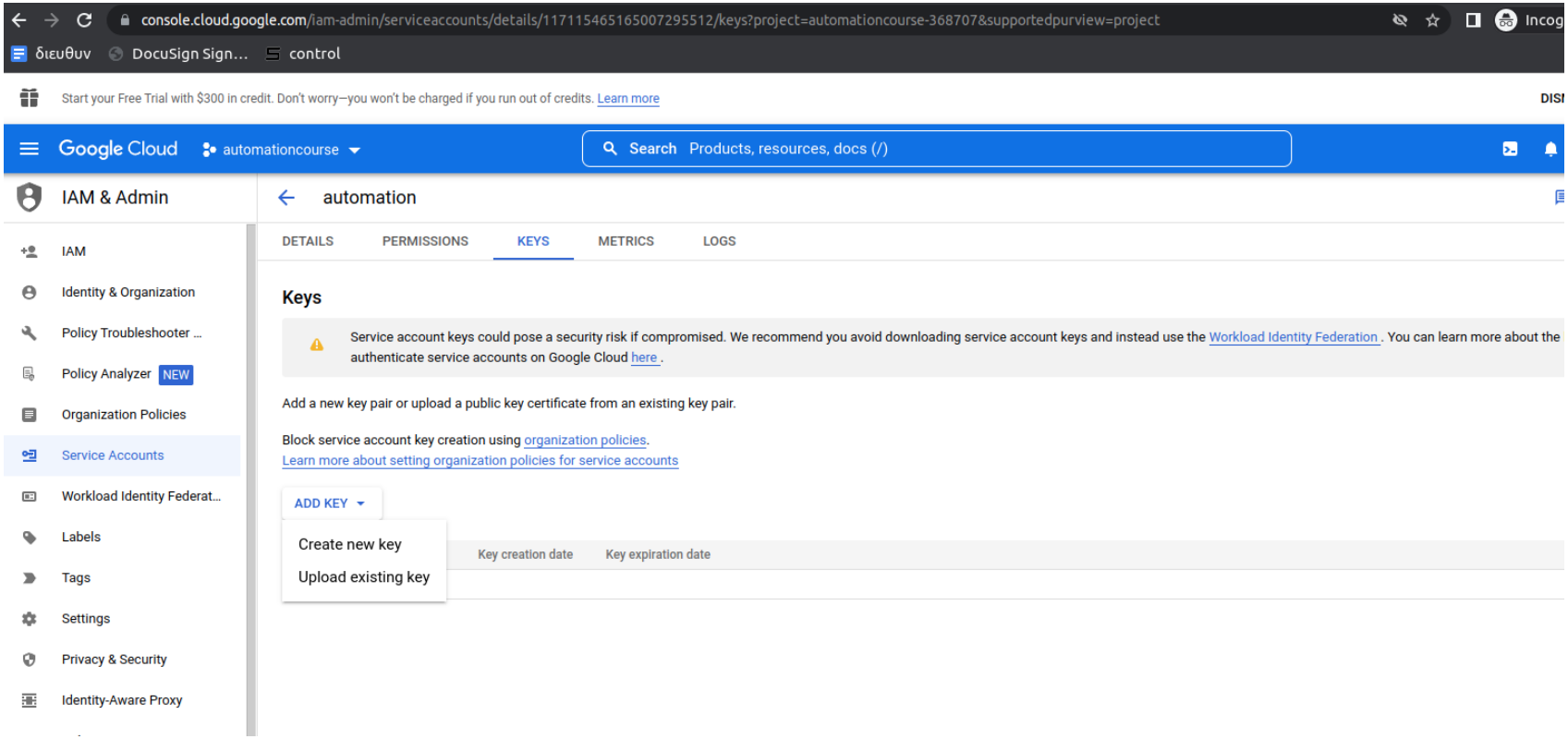
Δημιουργούμε έναν client



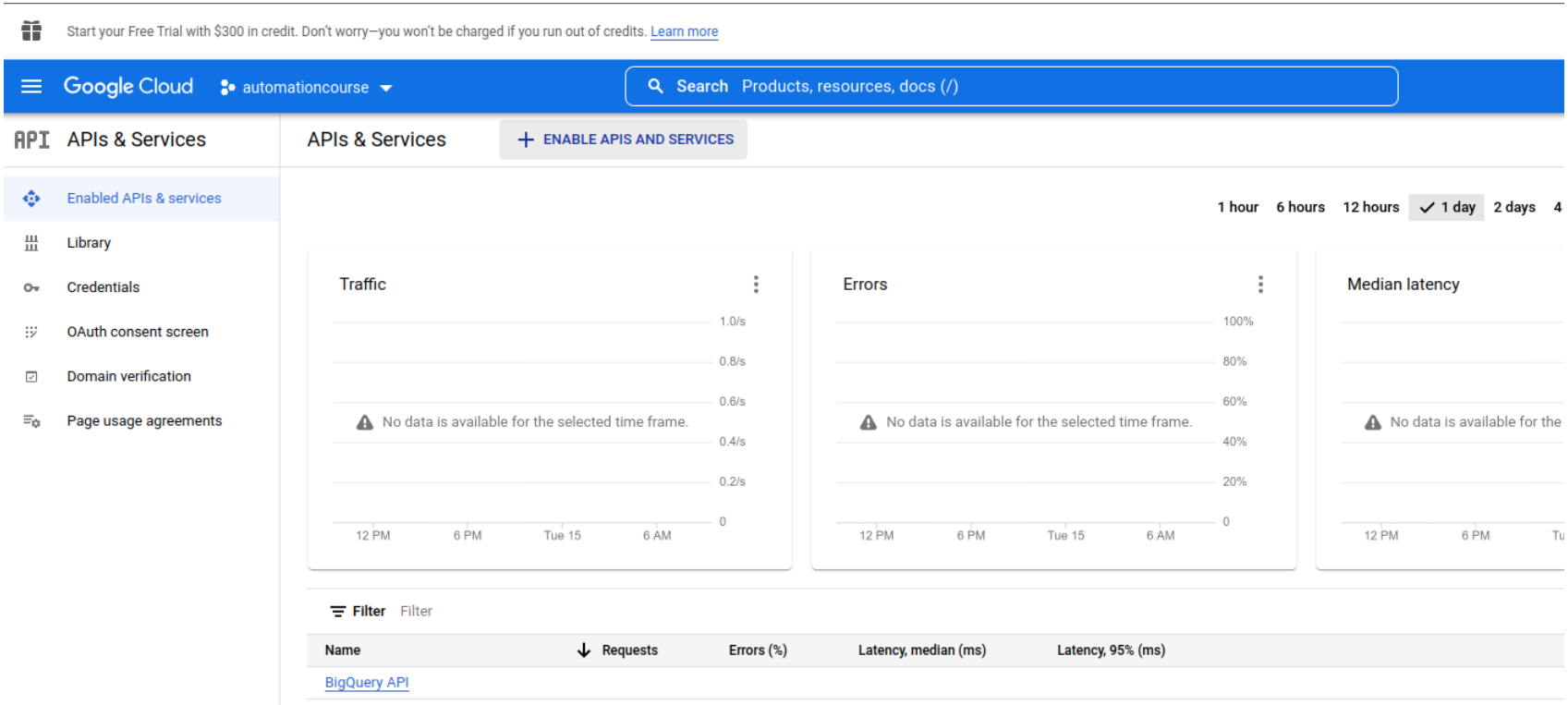
Δημιουργούμε έναν client



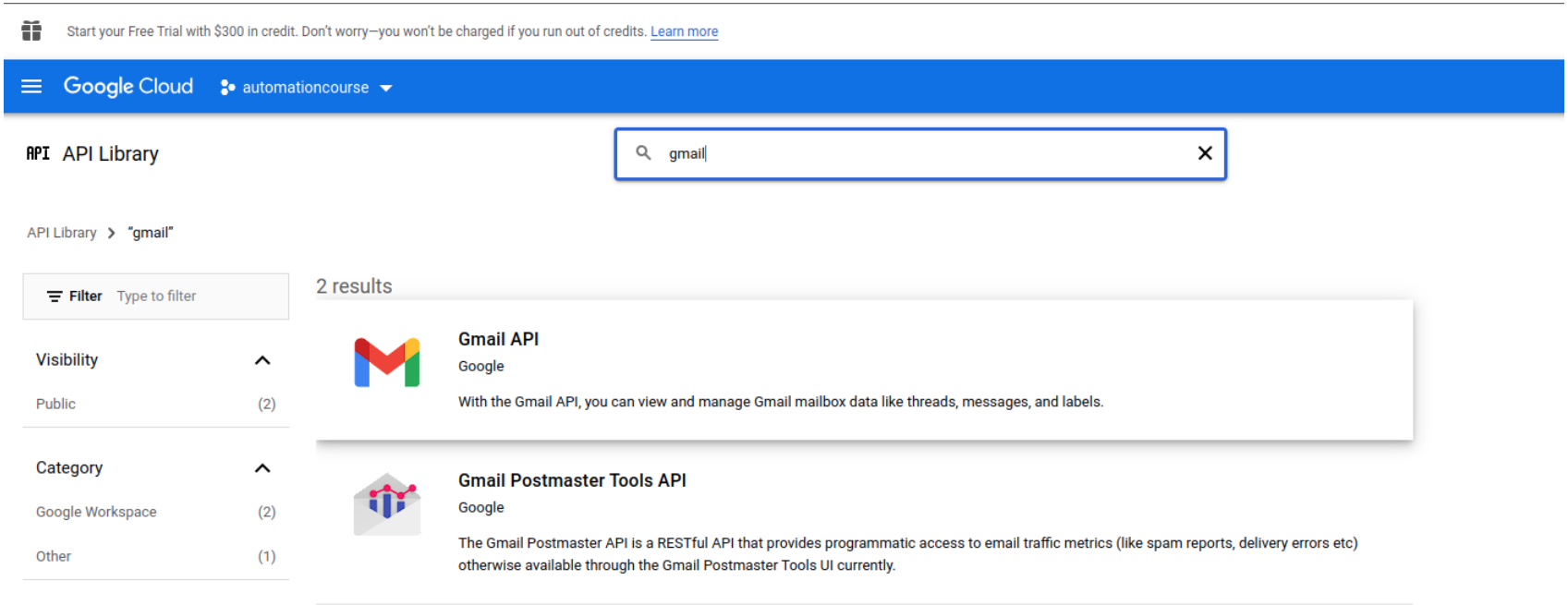
Δημιουργούμε έναν client



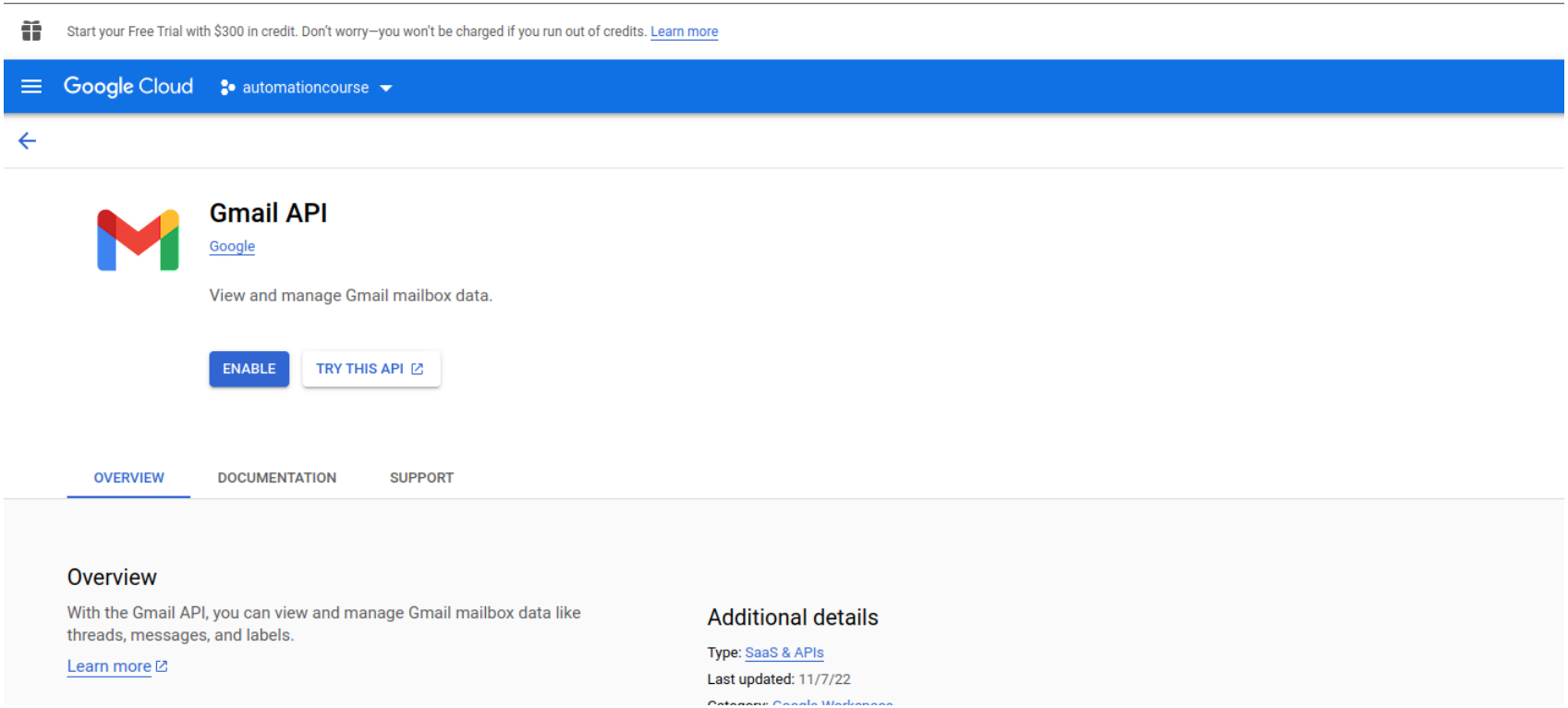
Δημιουργούμε έναν client



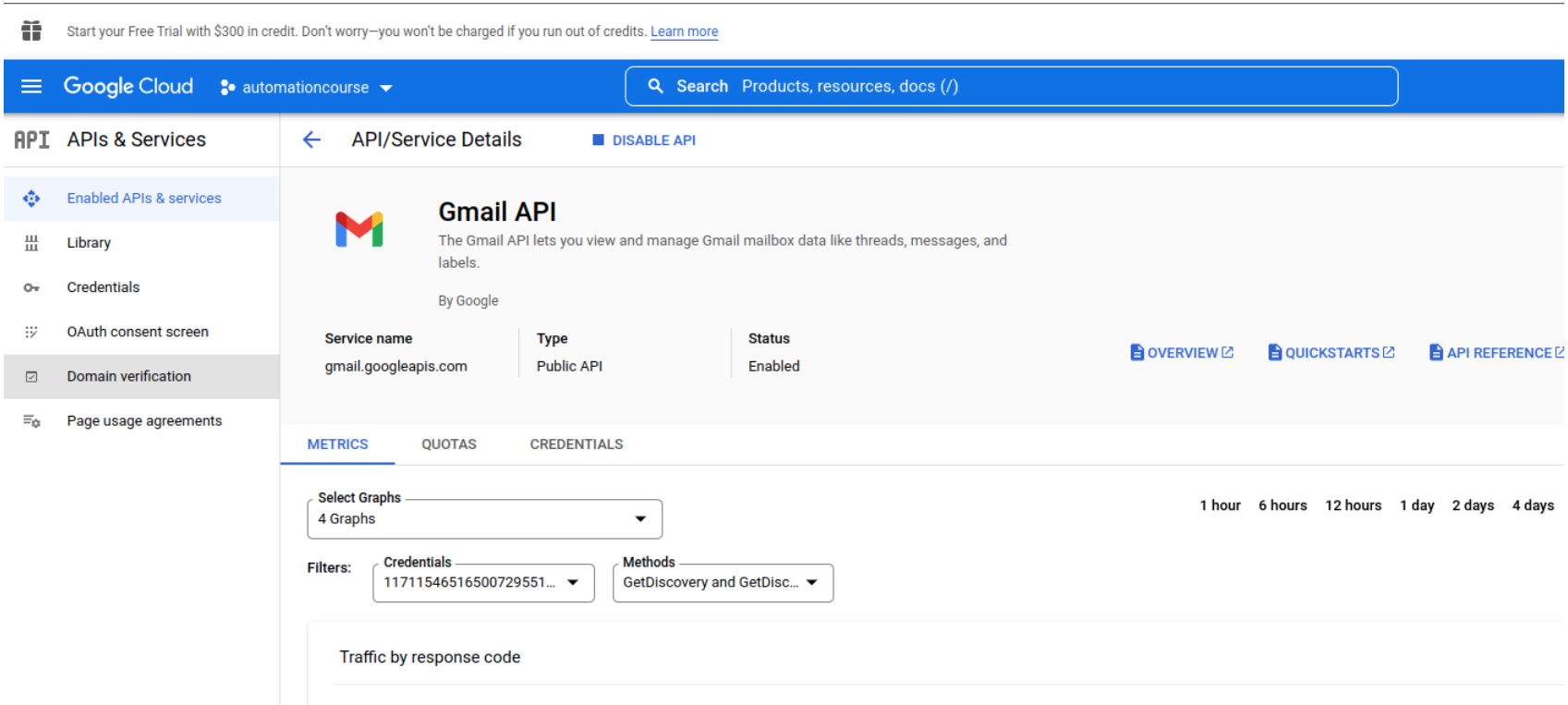
Ανοίγουμε το Gmail API



Ανοίγουμε το Gmail API



Ανοίγουμε το Gmail API



Εγκαθιστούμε τις απαραίτητες βιβλιοθήκες

```
root@debian:~# apt-get install python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential cpp
  cpp-10 dpkg-dev fakeroot fontconfig-config fonts-dejavu-core g++ g++-10 gcc
  gcc-10 javascript-common libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libasan6 libatomic1 libbinutils libc-dev-bin
  libc-devtools libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0
  libdeflate0 libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl
  libfontconfig1 libgcc-10-dev libgd3 libgomp1 libisl23 libitm1 libjbig0
  libjpeg62-turbo libjs-jquery libjs-sphinxdoc libjs-underscore liblsan0
  libmpc3 libmpfr6 libnsl-dev libpython3-dev libpython3.9 libpython3.9-dev
```

Εγκαθιστούμε τις απαραίτητες βιβλιοθήκες

```
                              Don't periodically check PyPI to determine
                              whether a new version of pip is available for
                              download. Implied with --no-index.
  --no-color                  Suppress colored output.
  --no-python-version-warning
                              Silence deprecation warnings for upcoming
                              unsupported Pythons.
  --use-feature <feature>     Enable new functionality, that may be backward
                              incompatible.
  --use-deprecated <feature>  Enable deprecated functionality, that will be
                              removed in the future.
root@debian:~# pip install --upgrade google-api-python-client google-auth-httpli
b2 google-auth-oauthlib
```

Gmail Quickstart

```
2  import os.path
3  from google.auth.transport.requests import Request
4  from google.oauth2.credentials import Credentials
5  from google_auth_oauthlib.flow import InstalledAppFlow
6  from googleapiclient.discovery import build
7  from googleapiclient.errors import HttpError
8
9  # API scope
10 SCOPES = ['https://mail.google.com/']
11 creds = None
```

Listing: testgmail.py

Gmail Quickstart

```
13 # Do we already have a token?
14 if os.path.exists('token.json'):
15     # If so get credentials from token file
16         creds = Credentials.from_authorized_user_file('token.json', SCOPES)
17 # Now check whether we need to refresh the token or obtain a new one
18 if not creds or not creds.valid:
19
20     # If the token has expired, refresh it
21     if creds and creds.expired and creds.refresh_token:
22         creds.refresh(Request())
23
24     # otherwise we must login
25     else:
26         flow = InstalledAppFlow.from_client_secrets_file(
27             'credentials.json', SCOPES)
28         creds = flow.run_local_server(port=0)
29
30     # Save the token for the next run
31     with open('token.json', 'w') as token:
32         token.write(creds.to_json())
```

Listing: testgmail.py

Gmail Quickstart

```
34 try:
35     # Call the Gmail API
36     service = build('gmail', 'v1', credentials=creds)
37     results = service.users().labels().list(userId='me').execute()
38     labels = results.get('labels', [])
39 
40     if not labels:
41         print('No labels found.')
42     else:
43         print('Labels:')
44         for label in labels:
45             print(label['name'])
46 
47 except HttpError as error:
48     print(f'An error occurred: {error}')
```

Listing: testgmail.py

Gmail Quickstart

```
34 try:
35     # Call the Gmail API
36     service = build('gmail', 'v1', credentials=creds)
37     results = service.users().labels().list(userId='me').execute()
38     labels = results.get('labels', [])
39 
40     if not labels:
41         print('No labels found.')
42     else:
43         print('Labels:')
44         for label in labels:
45             print(label['name'])
46 
47 except HttpError as error:
48     print(f'An error occurred: {error}')
```

Listing: testgmail.py

Βασικές Λειτουργίες - libgmail.py

```
2  import mimetypes
3  import os.path
4  from google.auth.transport.requests import Request
5  from google.oauth2.credentials import Credentials
6  from google_auth_oauthlib.flow import InstalledAppFlow
7  from googleapiclient.discovery import build
8  from googleapiclient.errors import HttpError
9  import base64
10 from email.message import EmailMessage
11
12 SCOPES = ['https://mail.google.com/']
```

Listing: testgmail.py

Βασικές Λειτουργίες - libgmail.py

```
14 class gmailapi:
15
16     def __init__(self, token_file = 'token.json', credentials_file = 'credentials.json', from_ad = '
       thomaskamalakis@gmail.com'):
17         self.from_ad = from_ad
18         try:
19             creds = None
20             if os.path.exists(token_file):
21                 creds = Credentials.from_authorized_user_file(token_file, SCOPES)
22
23             if not creds or not creds.valid:
24                 if creds and creds.expired and creds.refresh_token:
25                     creds.refresh(Request())
26                 else:
27                     flow = InstalledAppFlow.from_client_secrets_file(
28                         credentials_file, SCOPES)
29                     creds = flow.run_local_server(port = 0)
30                 with open(token_file, 'w') as token:
31                     token.write(creds.to_json())
32
33             self.service = build('gmail', 'v1', credentials = creds)
34
35         except HttpError as error:
36             print(f'An error occurred: {error}')
```

Listing: testgmail.py

Βασικές Λειτουργίες - libgmail.py

```
38      def build_message(self, to, subject, body):
39          message = EmailMessage()
40
41          message['To'] = to
42          message['From'] = self.from_ad
43          message['Subject'] = subject
44          message.set_content( body )
45          return message
```

Listing: libmail.py

Βασικές Λειτουργίες - libgmail.py

```
47      def build_draft(self, message):
48          encoded_message = base64.urlsafe_b64encode(message.as_bytes()).decode()
49          create_message = {
50              'message': {
51                  'raw': encoded_message
52              }
53          }
54          draft = self.service.users().drafts().create(userId = "me",
55                                                       body = create_message).execute()
56          return draft
```

Listing: libmail.py

Βασικές Λειτουργίες - libgmail.py

```
58      def create_draft(self, to, subject, body):
59          message = self.build_message(to, subject, body)
60          return self.build_draft(message)
61
62      def create_draft_with_attachments(self, to, subject, body, attachments = []):
63          message = self.build_message(to, subject, body)
64
65          for attachment in attachments:
66              type_subtype, _ = mimetypes.guess_type(attachment)
67              maintype, subtype = type_subtype.split('/')
68
69              with open(attachment, 'rb') as f:
70                  data = f.read()
71                  filename = os.path.basename( attachment )
72              message.add_attachment(data, maintype, subtype, filename = filename)
73          return self.build_draft( message )
```

Listing: libmail.py

Δημιουργία προσωρινού e-mail

```
75      def send(self, to, subject, body, attachments = []):
76          draft_dict = self.create_draft_with_attachments(to, subject, body, attachments = attachments)
77          id = draft_dict['id']
78          draft = self.service.users().drafts().send(body = {'id': id}, userId = 'me').execute()
```

Listing: libmail.py

```
1  from libgmail import gmailapi
2
3  subject = 'Test e-mail'
4  to = 'thkam@hua.gr'
5  body = \
6  """
7  This is a test email
8  using the gmail API
9  """
10
11 g = gmailapi()
12 g.create_draft(to, subject, body)
```

Listing: testdraft.py

Αποστολή e-mail

```
1  from libgmail import gmailapi
2
3  subject = 'Test e-mail'
4  to = 'thkam@hua.gr'
5  body = \
6  """
7  This is a test email
8  using the gmail API
9  """
10
11 attachment = 'pic.png'
12 g = gmailapi()
13 g.send(to, subject, body, attachments = [attachment])
```

Listing: testsend.py

Calendar API

1. Με τον ίδιο τρόπο ενεργοποιούμε το Calendar API στο GCP
2. Φτιάνουμε ένα project
3. Φτιάχνουμε τα διαπιστευτήρια
4. Ενσωματώνουμε στη ροή του Oauth2.0
5. Δίνουμε συγκατάθεση
6. Η εφαρμογή μας λαμβάνει ένα token που μπορεί να χρησιμοποιήσει το calendar μας.

Disclaimer

Calendar Quickstart

```
2  import datetime
3  import os.path
4
5  from google.auth.transport.requests import Request
6  from google.oauth2.credentials import Credentials
7  from google_auth_oauthlib.flow import InstalledAppFlow
8  from googleapiclient.discovery import build
9  from googleapiclient.errors import HttpError
10
11 SCOPES = ['https://www.googleapis.com/auth/calendar']
```

Listing: testcalendar.py

Calendar Quickstart

```
13 creds = None
14 if os.path.exists('cal_token.json'):
15     creds = Credentials.from_authorized_user_file('cal_token.json', SCOPES)
16 if not creds or not creds.valid:
17     if creds and creds.expired and creds.refresh_token:
18         creds.refresh(Request())
19     else:
20         flow = InstalledAppFlow.from_client_secrets_file(
21             'calendar_creds.json', SCOPES)
22         creds = flow.run_local_server(port=0)
23     with open('cal_token.json', 'w') as token:
24         token.write(creds.to_json())
```

Listing: testcalendar.py

Calendar Quickstart

```
26 try:
27     service = build('calendar', 'v3', credentials=creds)
28
29     # Call the Calendar API
30     now = datetime.datetime.utcnow().isoformat() + 'Z'  # 'Z' indicates UTC time
31     print('Getting the upcoming 10 events')
32     events_result = service.events().list(calendarId='primary', timeMin=now,
33                                           maxResults=10, singleEvents=True,
34                                           orderBy='startTime').execute()
35     events = events_result.get('items', [])
```

Listing: testcalendar.py

Calendar Quickstart

```
37      if not events:
38          print('No upcoming events found.')
39      else:
40
41          # Prints the start and name of the next 10 events
42          for event in events:
43              start = event['start'].get('dateTime', event['start'].get('date'))
44              print(start, event['summary'])
45
46  except HttpError as error:
47      print('An error occurred: %s' % error)
```

Listing: testcalendar.py