Google Identity And OpenID Connect On Cloud Run Part 1

Dev Thakkar · Follow
4 min read · Dec 12, 2021

Scenario: For pricestats application authenticate user with Google Identity

Tech stack: Identity Platform, OAuth 2.0, Secret Manager, Cloud Run, Python, Flask, Visual Studio Code (VS Code), Cloud Code

Requirements:

(1) Allow user to sign-in with Google Identity before displaying initial page of pricestats application.

(2) Use **Identity Platform** to sign in users with an **OpenID Connect** (OIDC) provider (Google)

(3) Store Oauth2.0 Client ID in Secret Manager

(4) Deploy python application to Google Cloud Run

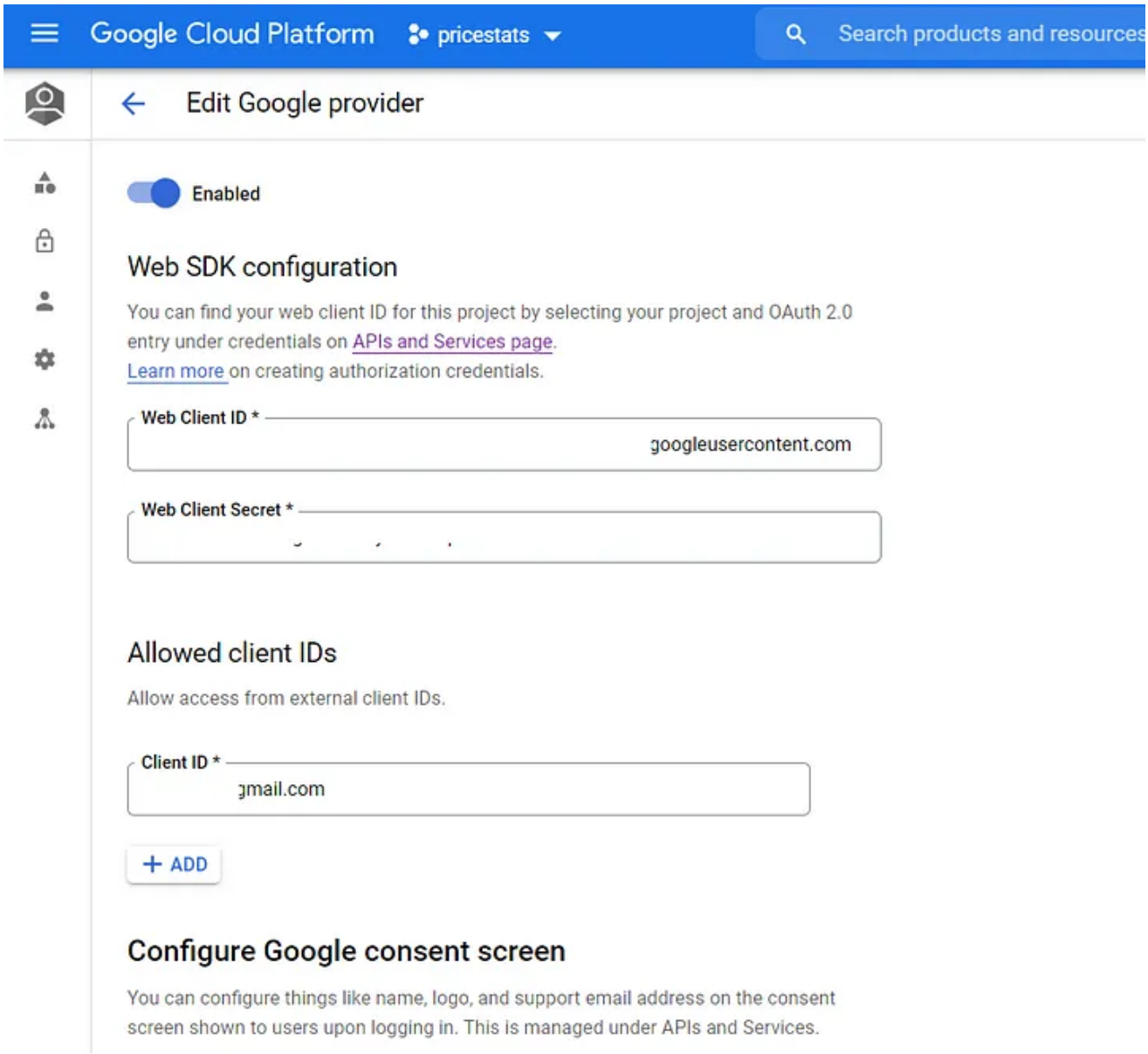
Outcome:

Step 1: Prerequisites

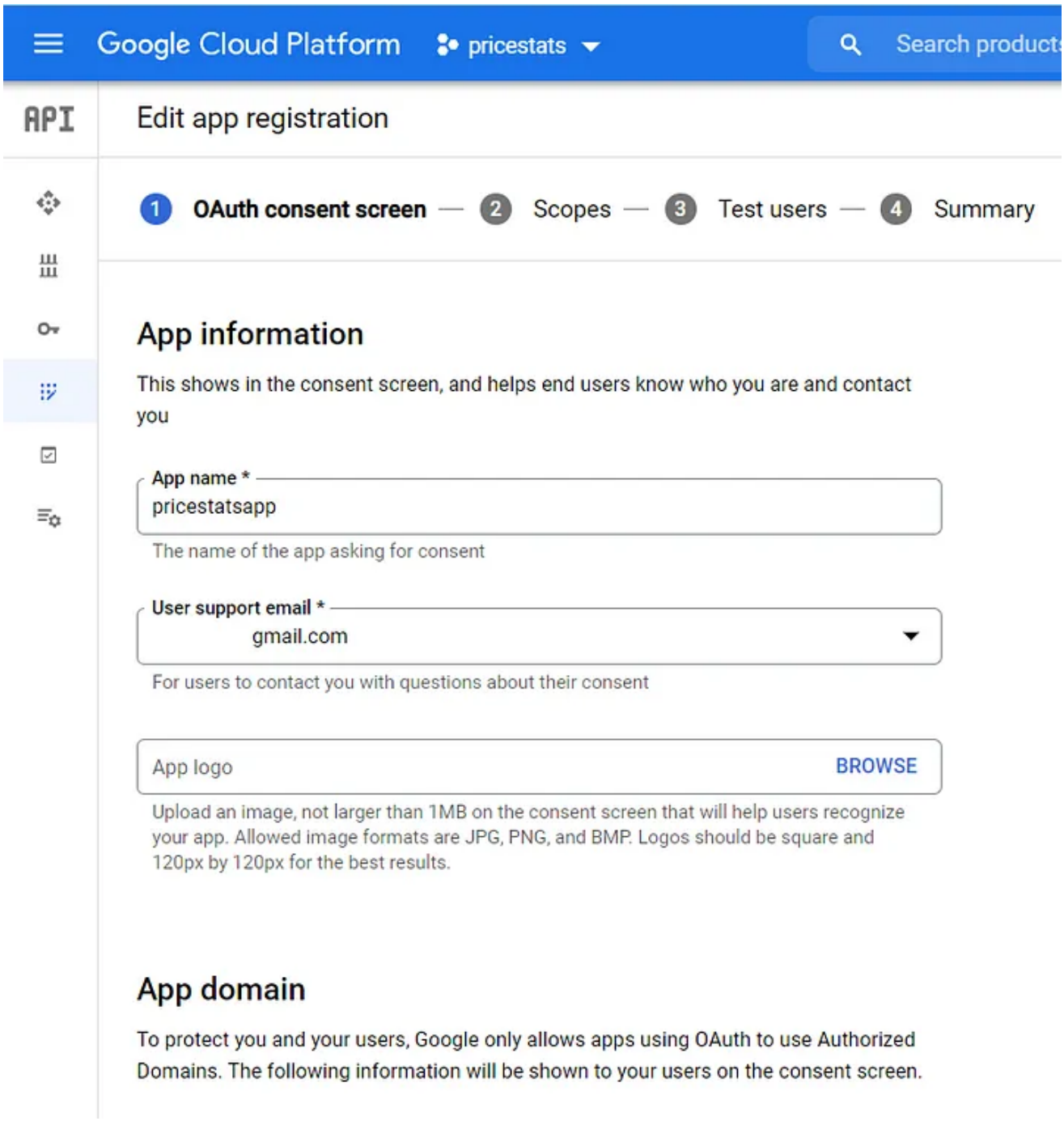
- Setup Google Cloud (GCP) account
- Download VS Code. Install extensions for Python, Cloud Code and Authlib
- Review medium article on pricestats to understand application functionality. This article focus is on user sign-in only

> After completing all relevant steps from pricestats article start from Step 2 below

Step 2: Configure Google as a Identity Provider



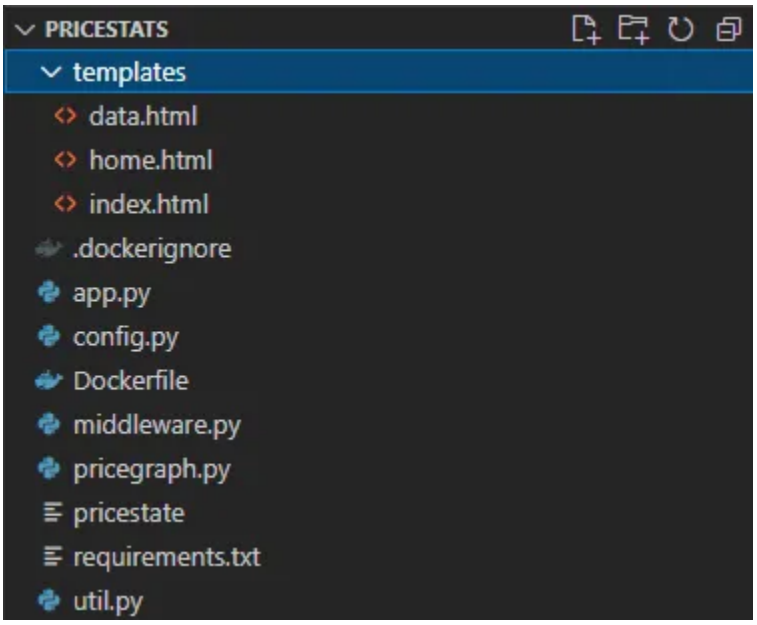
- Configure consent screen



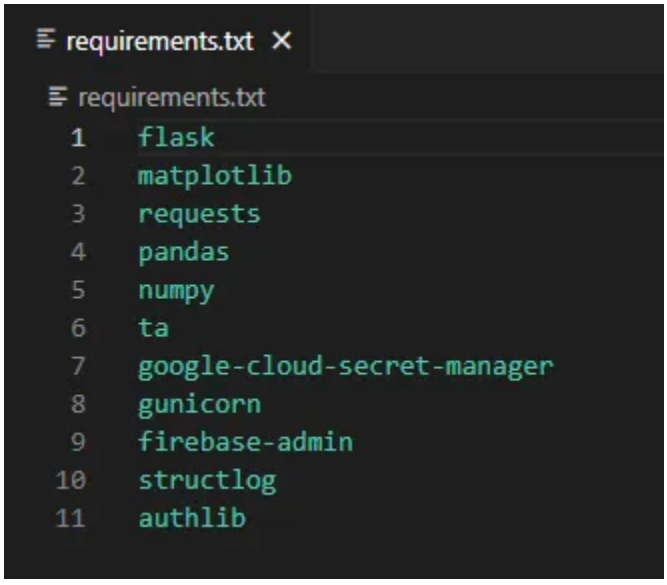
- On consent screen include **Authorized domains** as below

Step 3: Lets code

- In VS Code select empty “pricestats” folder. Below diagram shows files in folder after completing code



- Update **requirements.txt** as below



- At terminal within pricestats folder run below commands

```
> python -m venv env
> env\Scripts\activate.bat
```

```
> pip install -r requirements.txt
```

- Update **.dockerignore** as below

```
.dockerignore
Dockerfile
README.md
*.pyc
*.pyo
*.pyd
__pycache__
.pytest_cache
env
```

- Update **Dockerfile** as below

```
Dockerfile > ...
# Python image to use.
FROM python:3.8

# Set the working directory to /app
WORKDIR /app

# copy the requirements file used for dependencies
COPY requirements.txt .

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Copy the rest of the working directory contents into the container at /app
COPY . .

# Run app.py when the container launches
ENTRYPOINT ["python", "app.py"]
```

- In templates folder create **index.html, home.html** and **data.html**

templates > index.html

```html

<html lang=en xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Price Stats</title>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css">
  <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
  <link rel="icon" href="data:image/svg+xml,<svg xmlns=%22http://www.w3.org/2000/svg%22 viewBox=%220 0
  100 100%22><text y=%22.9em%22 font-size=%2290%22>#</text></svg>">
</head>
<body class="yellow lighten-5">
  <nav class="teal darken-3">
    <div class="nav-wrapper container"><a class="brand-logo center" href="#!">Price Stats</a>
      <ul class="left" id="nav-mobile">
        {% if accesstoken %}
        <a href="/logout">logout</a>
        {% else %}
          <a href="/login">login</a>
        {% endif %}
      </ul>
    </div>
  </nav>
</body>
</html>
```

Show “Login” link when accesstoken not present (else statement)

templates > home.html

```html
<body class="yellow lighten-5">
  <nav class="teal darken-3">
    <div class="nav-wrapper container"><a class="brand-logo center" href="#!">Price Stats</a>
      <ul class="left" id="nav-mobile">
        {% if accesstoken %}
        </pre>
        <a href="/logout">logout</a>
        {% else %}
          <a href="/login">login</a>
        {% endif %}
      </ul>
    </div>
  </nav>
  <form action="/data" method = "POST">
    <label for="symbol">Symbol:</label>
    <select style="display: block; width:120px" name="symbol" id="symbol">
      <option value="SPY">SPY</option>
      <option value="AAPL">AAPL</option>
      <option value="MSFT">MSFT</option>
      <option value="PFE">PFE</option>
    </select>
    <label for="days">time (in days):</label>
    <select style="display: block; width:120px" name="days" id="days">
        <option value="5">5</option>
        <option value="10">10</option>
        <option value="15">15</option>
      </select>
    <p><input id="submitauth" type = "submit" value = "Submit" /></p>
</form>
</body>
</html>
```

After valid Google login the home.html page is displayed with pricestats dropdowns (symbol and days)

templates > data.html

```html
<body class="yellow lighten-5">
  <nav class="teal darken-3">
    <div class="nav-wrapper container"><a class="brand-logo center" href="#!">Price Stats</a>
      <ul class="left" id="nav-mobile">
        {% if accesstoken %}
        </pre>
        <a href="/logout">logout</a>
        {% else %}
          <a href="/login">login</a>
          {% endif %}
      </ul>
    </div>
  </nav>
   {% for key,value in form_data.items() %}
   <p> {{key}}</p>
   <p> {{value}}</p>
   {% endfor %}
    <img src="/imageshow.jpeg" alt="Image Placeholder" height="400">
</body>
</html>
```

data.html will display graph after submit from home.html

Explanation: (1) **home.html** will display Login link (2) After login through Google Identity the user is directed to home.html page (3) **home.html** will display two drop downs (a) Symbol and (b) time in days (2) On submit the home.html *formdata* will route through "*data*" method in **app.py** (given below) (3) **data.html** shows selected (a) symbol (b) time in days and generated graph in *img* tag

- **util.py** and and **pricegraph.py** remain same as provided in pricestats article
- Update **middleware.py** as below

```python
middleware.py ×
middleware.py > validatetoken > decorated_function
from functools import wraps
from typing import Any, Callable
from flask import request, Response, session


# [START cloudrun_user_validatetoken]
def validatetoken(func: Callable[..., int]) -> Callable[..., int]:
    @wraps(func)
    def decorated_function(*args: Any, **kwargs: Any) -> Any:
        ipaddr = request.remote_addr
        tokenvalue = request.cookies.get('access_token')

        ###***PROD consideration***### - in production deny access for ipaddr with localhost address or IP's in a given range

        # below code uses flask session to store and validate token
        useridsession = session.get(ipaddr)
        if useridsession == None:
            return Response('Access denied', status=401)

        accesstokensession = session.get(useridsession)
        if tokenvalue != accesstokensession:
            return Response('Access denied', status=401)

        ###***PROD consideration***### - in production validate user using database session management
        ###***PROD consideration***### - refresh access token token in db/Flask session after every call
        ###***PROD consideration***### - create mechanism to expire access token due to inactvity after few minutes

        return func(*args, **kwargs)

    return decorated_function

# [END cloudrun_user_validatetoken]
```

Explanation: validatetoken is a decorator function that verifies authenticated user

Note: Current design has session management within flask session only. In a production scenario each user session will be managed with a data store (eg Firestore) with token refreshed to prevent random user attacks

Review example python app session management using Firestore

```
From link https://developers.google.com/identity/sign-in/web/backend-auth

Create an account or session
After you have verified the token, check if the user is already in your user database. If so, establish an authenticated session for the user. If the user isn't yet in your user database, create a new user record from the information in the ID token payload, and establish a session for the user. You can prompt the user for any additional profile information you require when you detect a newly created user in your app.
```

- Update **config.py** as below

```python
config.py > ...
import os
from util import access_secret_version

# obtain ClientId and Client Secret stored in Secret Manager
GOOGLE_CLIENT_ID = access_secret_version('pricestats', 'googleclientkey', '1')
GOOGLE_CLIENT_SECRET = access_secret_version('pricestats', 'googleclientsecret', '1')
```

Client ID and Client Secret were created in Step 2 of this article. Optional to store the keys in Google Secret Manager

- Update **app.py** as below

```python
app.py > ...
from flask import Flask, url_for, Response, request
from flask import render_template, session, redirect
from flask.helpers import make_response
from matplotlib.figure import Figure
from pricegraph import plotgraph
from util import access_secret_version
import os
from middleware import validatetoken
from authlib.integrations.flask_client import OAuth

app = Flask(__name__, static_folder="static", static_url_path="")

# get sessionkey from Secret Manager
sessionKey = access_secret_version('pricestats', 'sessionkey', '1')

# assign session key for managing Flask session
app.config['SECRET_KEY'] = sessionKey

app.config.from_object('config')

#register OAuth
CONF_URL = 'https://accounts.google.com/.well-known/openid-configuration'
oauth = OAuth(app)
oauth.register(
    name='google',
    server_metadata_url=CONF_URL,
    client_kwargs={
        'scope': 'openid email profile'
    }
)
```

```python
@app.route('/')
def index():
    """ renders index.html page
    """
    return render_template('index.html')


@app.route('/login')
def login():
    redirect_uri = url_for('auth', _external=True)
    return oauth.google.authorize_redirect(redirect_uri)


@app.route('/auth')
def auth():
    token = oauth.google.authorize_access_token()
    idinfo = oauth.google.parse_id_token(token)

    if idinfo['iss'] != 'https://accounts.google.com':
        return f"wrong issuer"

    userid = idinfo['sub']

    accesstoken = token.get('access_token')
    if accesstoken:
        session[userid] = accesstoken
        ipaddr = request.remote_addr
        session[ipaddr] = userid
        resp = make_response(redirect('/home')) # instead of return redirect('/home')
        resp.set_cookie('access_token', accesstoken)
        #resp.set_cookie('refresh_token', 'YOUR_REFRESH_TOKEN') TODO
        return resp

@app.route('/home')
@validatetoken
def homepage():
    sessiontoken = pullaccesstoken()
    if sessiontoken == 'Error':
        return f'Access Denied'

    return render_template('home.html', accesstoken= sessiontoken)
```

Explanation: (1) initial page load shows the index.html page (2) User clicks on “Login” link that directs call to *login* method (3) From *login* method the user is redirected to *auth* method (4) Google Logon screen will be presented for user (in case user is already logged in then the screen will not pop) (5)After valid authentication the *homepage* method is called using route ‘/home’ (6) *homepage* method includes decorator *validatetoken that verifies access token before allowing user to proceed*

app.py > ...

```python

@app.route('/data', methods = ['POST', 'GET'])
@validatetoken
def data():
    """ renders data.html page. /data action called on form submit within index.html
    """
    if request.method == 'GET':
        return f"Access Denied"  #deny access if get request directly to data.html

    if request.method == 'POST':
        formdata = request.form
        session['formvar'] = formdata #store formdata in session
        sessiontoken = pullaccesstoken()
        if sessiontoken == 'Error':
            return f"Access Denied"

        return render_template('data.html', form_data = formdata, accesstoken = sessiontoken )

@app.route("/imageshow.jpeg")
@validatetoken
def plot_jpeg():
    """ renders the image. /imageshow.jpeg action is invoked from img tag of data.html
    """
    formvar = session.pop('formvar', None) #fetch formdata from session
    output = plotgraph(formvar) #call to generate price chart
    return Response(output.getvalue(), mimetype="image/jpeg")
```

methods **data** and **plot_jpeg** provide similar pricestats functionality

app.py > pullaccesstoken

```python
def pullaccesstoken():
    ipaddr = request.remote_addr

    #In production deny access for ipaddr with localhost address or IP's in a
    # given range TODO
    useridsession = session.get(ipaddr)
    if useridsession == None:
        return 'Error'

    accesstokensession = session.get(useridsession)
    if accesstokensession == None:
        return 'Error'

    return accesstokensession

@app.route('/logout')
def logout():
    ipaddr = request.remote_addr
    useridsession = session.get(ipaddr)
    # check for useridsession.
    # useridsession key  maps to accesstoken. If exists then remove key from session
    if useridsession != None:
            session.pop(useridsession)

    #remove ipaddr key that maps to userid
    session.pop(ipaddr)
    return redirect('/')

if __name__ == '__main__':
    server_port = os.environ.get('PORT', '8080')
    app.run(debug=False, port=server_port, host='0.0.0.0')
```

clean up user from session during logout

Test Local => Continue to Part 2

Openid Connect

Google Cloud Run

Written by Dev Thakkar

3 Followers · 3 Following

No responses yet

What are your thoughts?

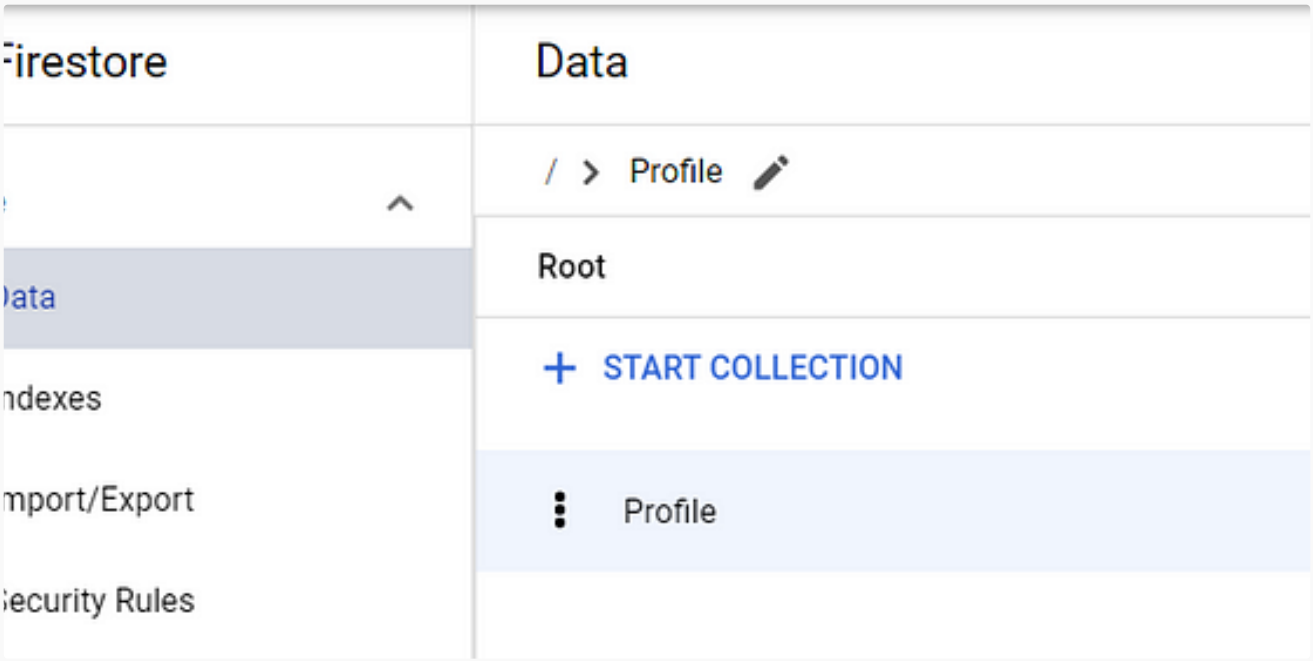
Respond

More from Dev Thakkar

```
aml > ...
Fact Registry - create repository kmtestrepo
d and push to repository kmtestrepo

e: 'gcr.io/cloud-builders/docker'
s: [ 'build', '-t', '${_LOCATION}-docker.pkg.dev/$PROJECT_ID/${_REPOSITORY}/${_IMAGE}

s:
_LOCATION}-docker.pkg.dev/$PROJECT_ID/${_REPOSITORY}/${_IMAGE}'

itutions:
CATION: "us-east1"
POSITORY: "kmtestrepo"
AGE: "my-image"
```

Dev Thakkar

Dec 29, 2021 5 1



Dev Thakkar

Nov 19, 2021

You have 20 projects remaining in your quota. Request an increase or delete projects. Learn more

MANAGE QUOTAS

Project name *
KMSTESTPROJ

Project ID *
kmstestproj

Project ID can have lowercase letters, digits, or hyphens. It must start with a lowercase letter and end with a letter or number.

Dev Thakkar

Nov 19, 2021

Dev Thakkar

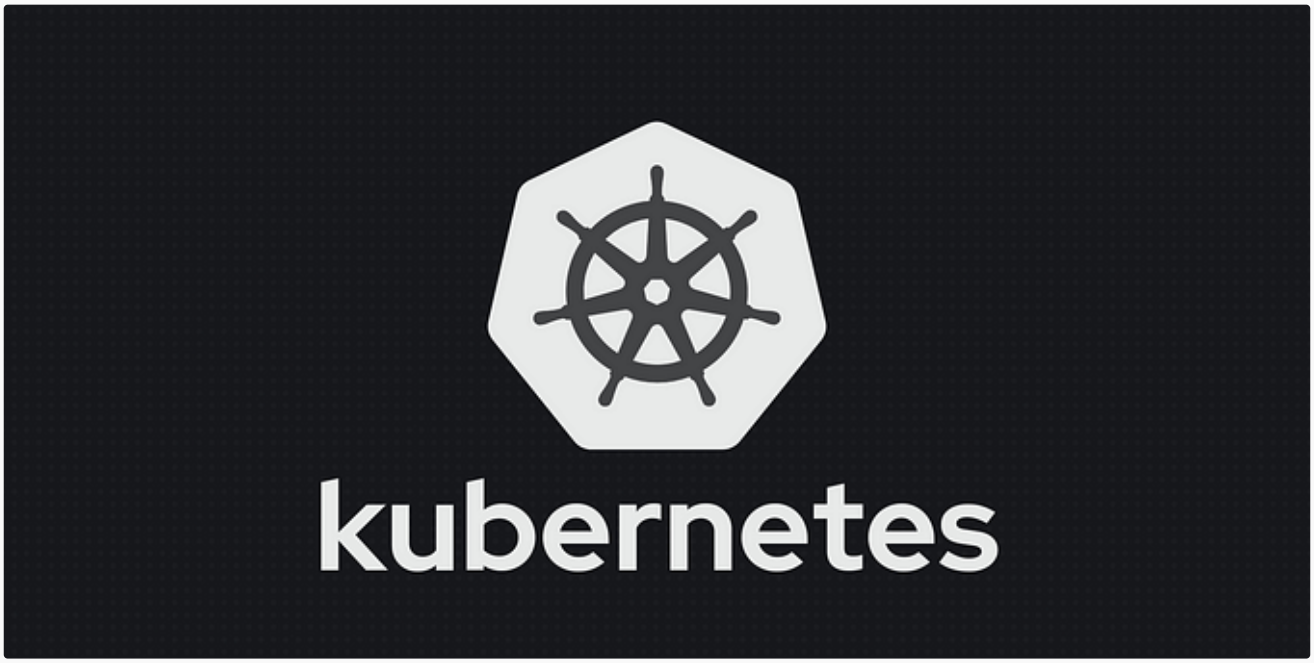
Mar 15

See all from Dev Thakkar

Recommended from Medium

Harendra

Oct 26 8.3K 132



In Stackademic by Crafting-Code

Nov 19 4.6K 137

Lists

Staff picks

791 stories · 1532 saves

Stories to Help You Level-Up at Work

19 stories · 901 saves

Self-Improvement 101

20 stories · 3167 saves

Productivity 101

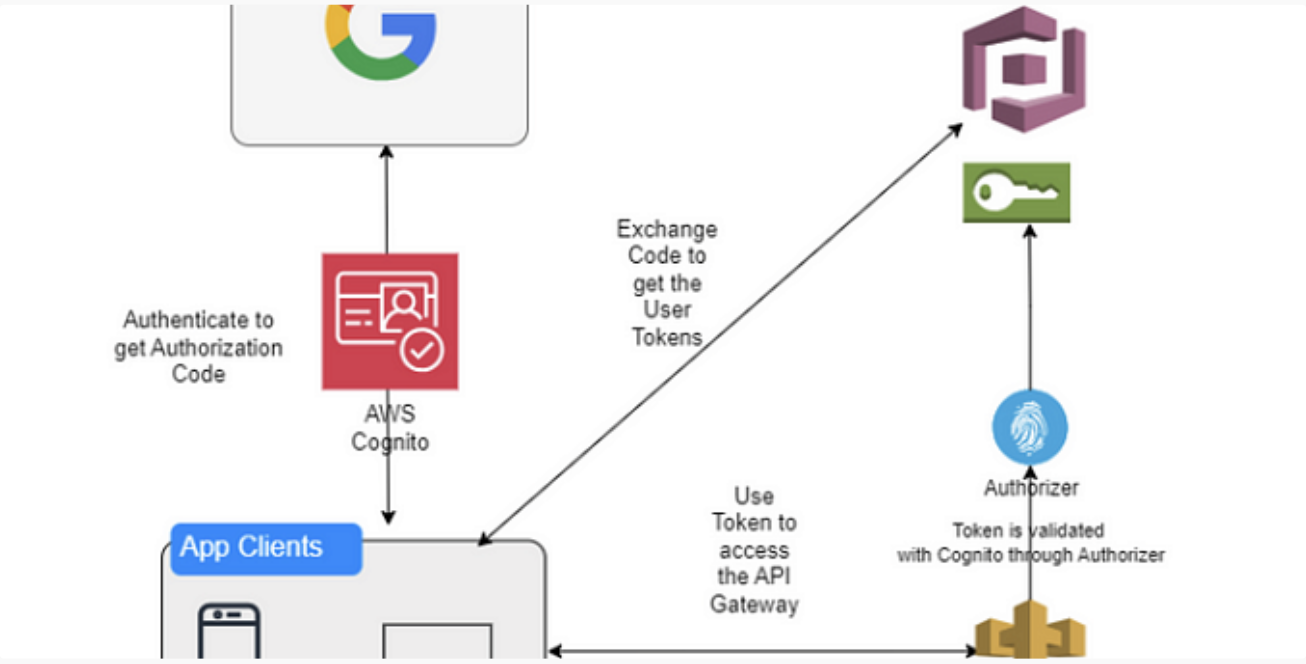
20 stories · 2681 saves

Jessica Stillman

Oct 30 18.3K 478

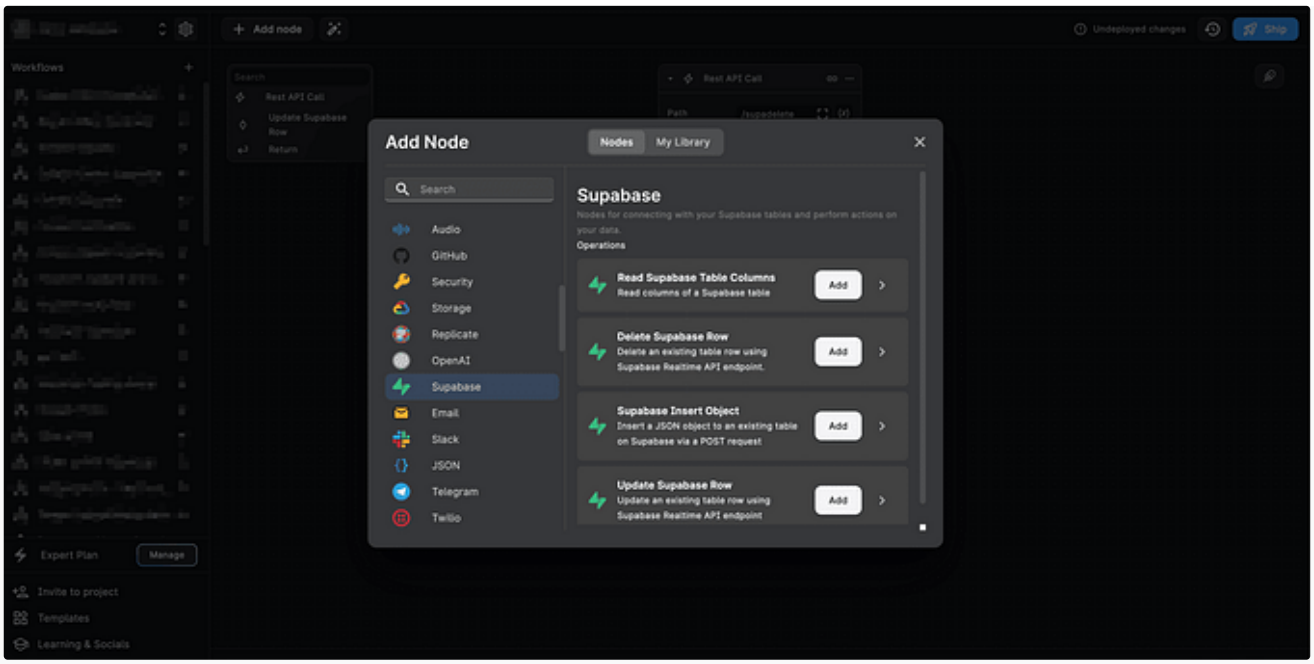
Vinod Pal

Dec 6



Shiv Pal Singh Kaundal

Oct 18 1



Girff

Nov 15

See more recommendations