

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

Google Cloud Platform pricestats

Search products and resources

← Edit Google provider

Web SDK configuration

You can find your web client ID for this project by selecting your project and OAuth 2.0 entry under credentials on [APIs and Services page](#).
[Learn more](#) on creating authorization credentials.

Web Client ID *

Web Client Secret *

Allowed client IDs

Allow access from external client IDs.

Client ID *

[+ ADD](#)

Configure Google consent screen

You can configure things like name, logo, and support email address on the consent screen shown to users upon logging in. This is managed under APIs and Services.

- Configure consent screen

Google Cloud Platform

pricestats

Search products

API

Edit app registration

1 OAuth consent screen

2 Scopes

3 Test users

4 Summary

App information

This shows in the consent screen, and helps end users know who you are and contact you

App name *

pricestatsapp

The name of the app asking for consent

User support email *

gmail.com

For users to contact you with questions about their consent

App logo

BROWSE

Upload an image, not larger than 1MB on the consent screen that will help users recognize your app. Allowed image formats are JPG, PNG, and BMP. Logos should be square and 120px by 120px for the best results.

App domain

To protect you and your users, Google only allows apps using OAuth to use Authorized Domains. The following information will be shown to your users on the consent screen.

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

Authorized redirect URIs ?

For use with requests from a web server

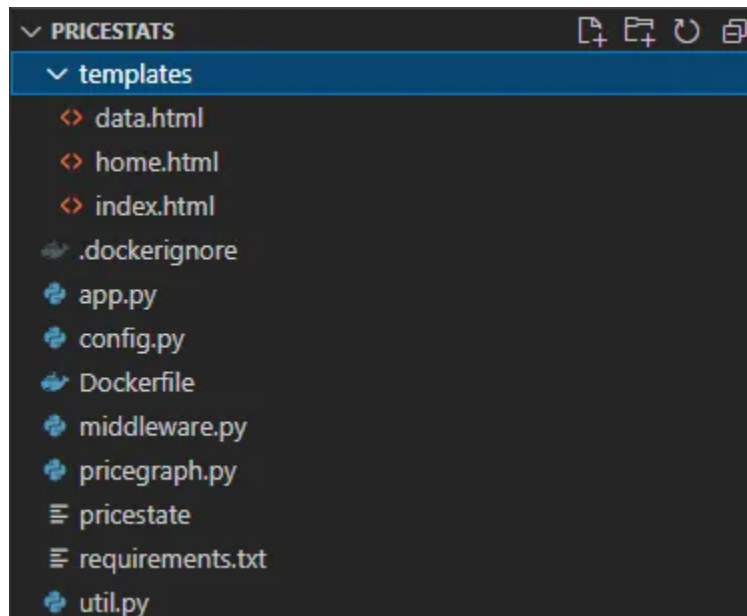
URIs *

http://localhost:8080/auth

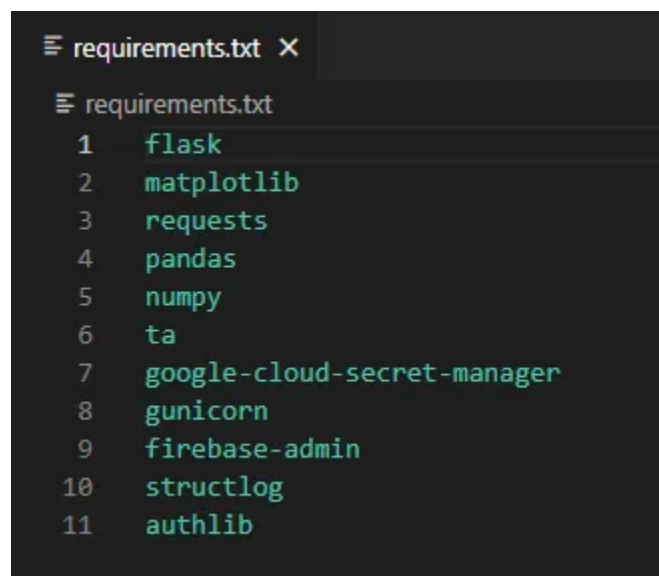
https://pricestats-syoawSrs34s-us.b.run.app/auth

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

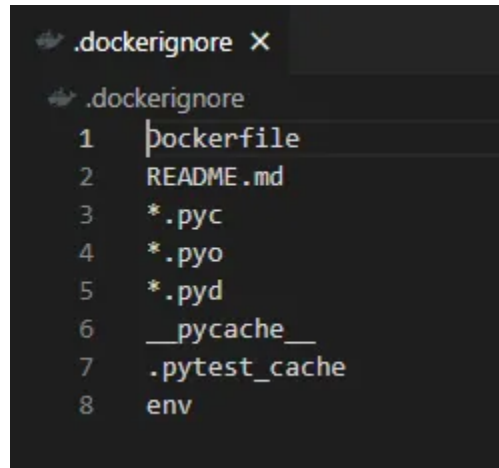


- At terminal within pricestats folder run below commands

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

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

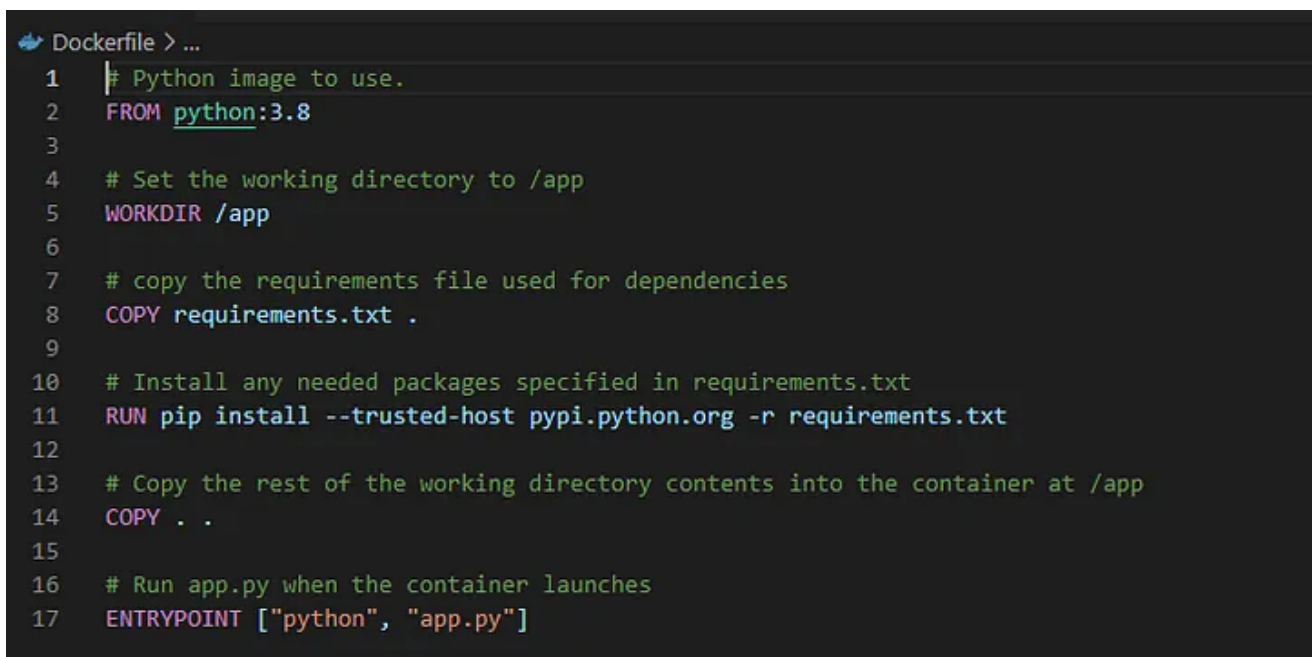
- Update **.dockerignore** as below



A screenshot of a code editor showing the contents of a `.dockerignore` file. The file lists several patterns to ignore: `dockerfile`, `README.md`, `*.pyc`, `*.pyo`, `*.pyd`, `__pycache__`, `.pytest_cache`, and `env`. The editor has a dark theme and a tab labeled `.dockerignore` at the top.

```
.dockerignore
1  dockerfile
2  README.md
3  *.pyc
4  *.pyo
5  *.pyd
6  __pycache__
7  .pytest_cache
8  env
```

- Update **Dockerfile** as below



A screenshot of a code editor showing the contents of a `Dockerfile`. The file contains instructions to build a Docker image based on `python:3.8`, set the working directory to `/app`, copy the `requirements.txt` file, install dependencies using `pip install --trusted-host pypi.python.org -r requirements.txt`, copy the rest of the working directory contents into the container, and run `app.py` when the container launches. The editor has a dark theme and a tab labeled `Dockerfile` at the top.

```
Dockerfile > ...
1  # Python image to use.
2  FROM python:3.8
3
4  # Set the working directory to /app
5  WORKDIR /app
6
7  # copy the requirements file used for dependencies
8  COPY requirements.txt .
9
10 # Install any needed packages specified in requirements.txt
11 RUN pip install --trusted-host pypi.python.org -r requirements.txt
12
13 # Copy the rest of the working directory contents into the container at /app
14 COPY . .
15
16 # Run app.py when the container launches
17 ENTRYPOINT ["python", "app.py"]
```

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

```

templates > <> index.html
1
2 <html lang=en xmlns:th="http://www.thymeleaf.org">
3 <head>
4   <title>Price Stats</title>
5   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css">
6   <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
7   <link rel="icon" href="data:image/svg+xml,<svg xmlns=%22http://www.w3.org/2000/svg%22 viewBox=%220 0
8     100 100%22><text y=%22.9em%22 font-size=%2290%22>#</text></svg>">
9 </head>
10 <body class="yellow lighten-5">
11   <nav class="teal darken-3">
12     <div class="nav-wrapper container"><a class="brand-logo center" href="#!">Price Stats</a>
13     <ul class="left id="nav-mobile">
14       .....{% if accesstoken %}
15       .....<a href="/logout">logout</a>
16       .....{% else %}
17       .....<a href="/login">login</a>
18       .....{% endif %}
19     </ul>
20   </div>
21 </nav>
22 </body>
23 </html>

```

Show "Login" link when accesstoken not present (else statement)

```

templates > <> home.html
9 <body class="yellow lighten-5">
10 <nav class="teal darken-3">
11   <div class="nav-wrapper container"><a class="brand-logo center" href="#!">Price Stats</a>
12   <ul class="left id="nav-mobile">
13     .....{% if accesstoken %}
14     .....</pre>
15     .....<a href="/logout">logout</a>
16   .....{% else %}
17   .....<a href="/login">login</a>
18   .....{% endif %}
19   </ul>
20 </div>
21 </nav>
22 <form action="/data" method = "POST">
23   <label for="symbol">Symbol:</label>
24   <select style="display: block; width:120px" name="symbol" id="symbol">
25     <option value="SPY">SPY</option>
26     <option value="AAPL">AAPL</option>
27     <option value="MSFT">MSFT</option>
28     <option value="PFE">PFE</option>
29   </select>
30   <label for="days">time (in days):</label>
31   <select style="display: block; width:120px" name="days" id="days">
32     <option value="5">5</option>
33     <option value="10">10</option>
34     <option value="15">15</option>
35   </select>
36   <p><input id="submitauth" type = "submit" value = "Submit" /></p>
37 </form>
38 </body>
39 </html>

```

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


```

templates > <> data.html
8  <body class="yellow lighten-5">
9  <nav class="teal darken-3">
10 <div class="nav-wrapper container"><a class="brand-logo center" href="#">Price Stats</a>
11 <ul class="left" id="nav-mobile">
12   {% if accesstoken %}
13   </pre>
14   <a href="/logout">logout</a>
15   {% else %}
16   <a href="/login">login</a>
17   {% endif %}
18 </ul>
19 </div>
20 </nav>
21 {% for key,value in form_data.items() %}
22 <p> {{key}}</p>
23 <p> {{value}}</p>
24 {% endfor %}
25 
26 </body>
27 </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 **pricegraph.py** remain same as provided in [pricestats](#) article
- Update **middleware.py** as below

```

middleware.py X
middleware.py > validate_token > decorated_function
1  from functools import wraps
2  from typing import Any, Callable
3  from flask import request, Response, session
4
5
6  # [START cloudrun_user_validate_token]
7  def validate_token(func: Callable[..., int]) -> Callable[..., int]:
8      @wraps(func)
9      def decorated_function(*args: Any, **kwargs: Any) -> Any:
10         ipaddr = request.remote_addr
11         tokenvalue = request.cookies.get('access_token')
12
13         #####PROD consideration##### - in production deny access for ipaddr with localhost address or IP's in a given range
14
15         # below code uses flask session to store and validate token
16         useridsession = session.get(ipaddr)
17         if useridsession == None:
18             return Response('Access denied', status=401)
19
20         accesstokensession = session.get(useridsession)
21         if tokenvalue != accesstokensession:
22             return Response('Access denied', status=401)
23
24         #####PROD consideration##### - in production validate user using database session management
25         #####PROD consideration##### - refresh access token token in db/Flask session after every call
26         #####PROD consideration##### - create mechanism to expire access token due to inactivity after few minutes
27
28         return func(*args, **kwargs)
29
30     return decorated_function
31
32 # [END cloudrun_user_validate_token]
33
34

```

Explanation: validate_token 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](https://developers.google.com/identity/sign-in/web/backend-auth)

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


```

config.py > ...
1  import os
2  from util import access_secret_version
3
4  # obtain ClientId and Client Secret stored in Secret Manager
5  GOOGLE_CLIENT_ID = access_secret_version(['pricestats', 'googleclientkey', '1'])
6  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

```

app.py > ...
1  from flask import Flask, url_for, Response, request
2  from flask import render_template, session, redirect
3  from flask.helpers import make_response
4  from matplotlib.figure import Figure
5  from pricegraph import plotgraph
6  from util import access_secret_version
7  import os
8  from middleware import validate_token
9  from authlib.integrations.flask_client import OAuth
10
11  app = Flask(__name__, static_folder="static", static_url_path="")
12
13  # get sessionkey from Secret Manager
14  sessionKey = access_secret_version('pricestats', 'sessionkey', '1')
15
16  # assign session key for managing Flask session
17  app.config['SECRET_KEY'] = sessionKey
18
19  app.config.from_object('config')
20
21  #register OAuth
22  CONF_URL = 'https://accounts.google.com/.well-known/openid-configuration'
23  oauth = OAuth(app)
24  oauth.register(
25      name='google',
26      server_metadata_url=CONF_URL,
27      client_kwargs={
28          'scope': 'openid email profile'
29      }
30  )

```

```

app.py x
app.py > logout

31
32 @app.route('/')
33 def index():
34     """ renders index.html page
35     """
36     return render_template('index.html')
37
38
39 @app.route('/login')
40 def login():
41     redirect_uri = url_for('auth', _external=True)
42     return oauth.google.authorize_redirect(redirect_uri)
43
44
45 @app.route('/auth')
46 def auth():
47     token = oauth.google.authorize_access_token()
48     idinfo = oauth.google.parse_id_token(token)
49
50     if idinfo['iss'] != 'https://accounts.google.com':
51         return f"wrong issuer"
52
53     userid = idinfo['sub']
54
55     accesstoken = token.get('access_token')
56     if accesstoken:
57         session[userid] = accesstoken
58         ipaddr = request.remote_addr
59         session[ipaddr] = userid
60         resp = make_response(redirect('/home')) # instead of return redirect('/home')
61         resp.set_cookie('access_token', accesstoken)
62         #resp.set_cookie('refresh_token', 'YOUR_REFRESH_TOKEN') TODO
63         return resp
64
65 @app.route('/home')
66 @validatetoken
67 def homepage():
68     sessiontoken = pullaccesstoken()
69     if sessiontoken == 'Error':
70         return f'Access Denied'
71
72     return render_template('home.html', accesstoken= sessiontoken)
73

```

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 > ...
74
75 @app.route('/data', methods = ['POST', 'GET'])
76 @validatetoken
77 def data():
78     """ renders data.html page. /data action called on form submit within index.html
79     """
80     if request.method == 'GET':
81         return f"Access Denied" #deny access if get request directly to data.html
82
83     if request.method == 'POST':
84         formdata = request.form
85         session['formvar'] = formdata #store formdata in session
86         sessiontoken = pullaccesstoken()
87         if sessiontoken == 'Error':
88             return f"Access Denied"
89
90         return render_template('data.html', form_data = formdata, accesstoken = sessiontoken )
91
92 @app.route("/imageshow.jpeg")
93 @validatetoken
94 def plot_jpeg():
95     """ renders the image. /imageshow.jpeg action is invoked from img tag of data.html
96     """
97     formvar = session.pop('formvar', None) #fetch formdata from session
98     output = plotgraph(formvar) #call to generate price chart
99     return Response(output.getvalue(), mimetype="image/jpeg")
100

```

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

```

app.py > pullaccesstoken
100
101 def pullaccesstoken():
102     ipaddr = request.remote_addr
103
104     #In production deny access for ipaddr with localhost address or IP's in a
105     # given range TODO
106     useridsession = session.get(ipaddr)
107     if useridsession == None:
108         return 'Error'
109
110     accesstokensession = session.get(useridsession)
111     if accesstokensession == None:
112         return 'Error'
113
114     return accesstokensession
115
116 @app.route('/logout')
117 def logout():
118     ipaddr = request.remote_addr
119     useridsession = session.get(ipaddr)
120     # check for useridsession.
121     # useridsession key maps to accesstoken. If exists then remove key from session
122     if useridsession != None:
123         session.pop(useridsession)
124
125     #remove ipaddr key that maps to userid
126     session.pop(ipaddr)
127     return redirect('/')
128
129 if __name__ == '__main__':
130     server_port = os.environ.get('PORT', '8080')
131     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

```

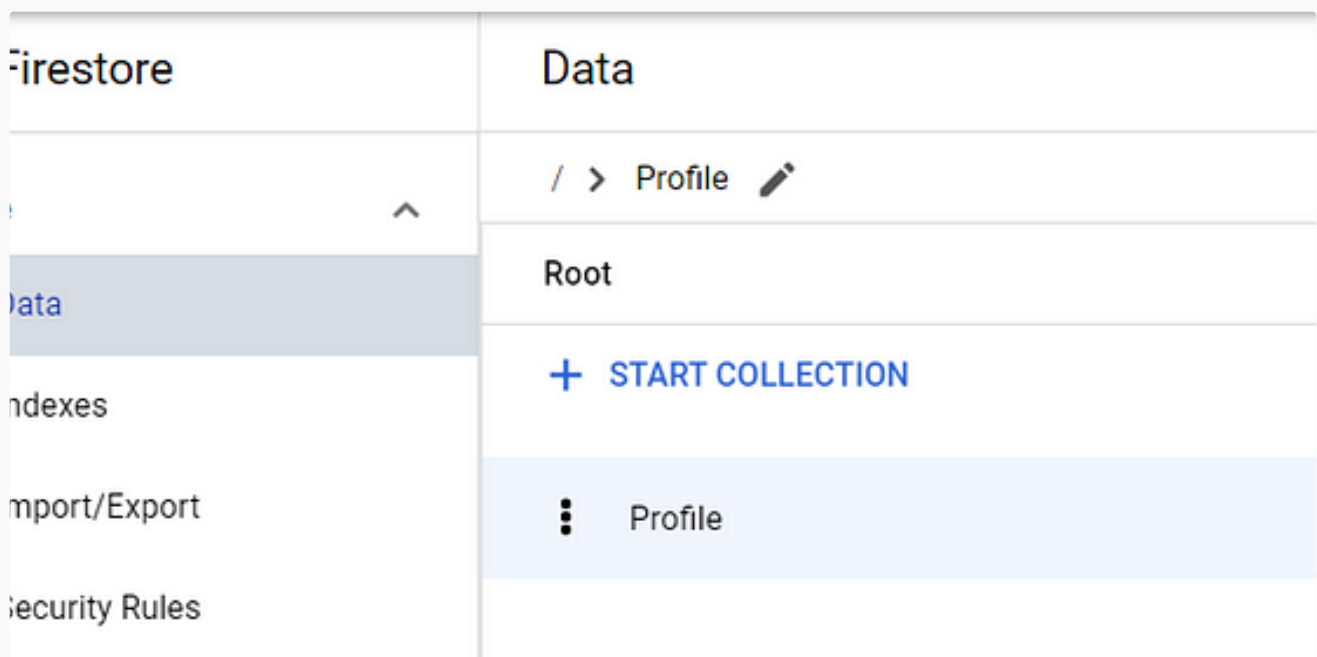
aml > ...
Fact| Registry - create repository kmtestrepo
d and push to repository kmtestrepo
:
e: 'gcr.io/cloud-builders/docker'
s: [ 'build', '-t', '${_LOCATION}-docker.pkg.dev/$PROJECT_ID/${_REPOSITORY}/${_IMAGE}'
s:
LOCATION}-docker.pkg.dev/$PROJECT_ID/${_REPOSITORY}/${_IMAGE}'

stitutions:
LOCATION: "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

Always Free
24 GB RAM + 4 CPU + 200 GB



 @harendraverma2  @harendra21  @harendra21



Harendra



Oct 26



8.3K



132



In Stackademic by Crafting-Code



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



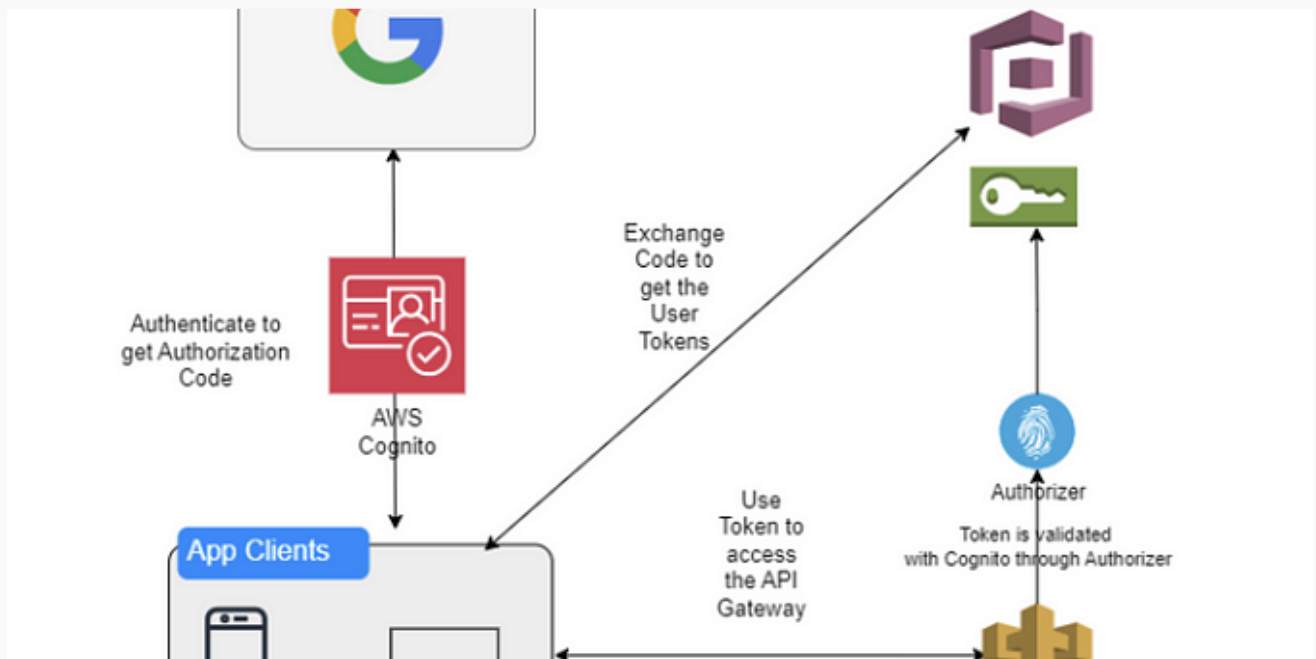
 Jessica Stillman






 Vinod Pal

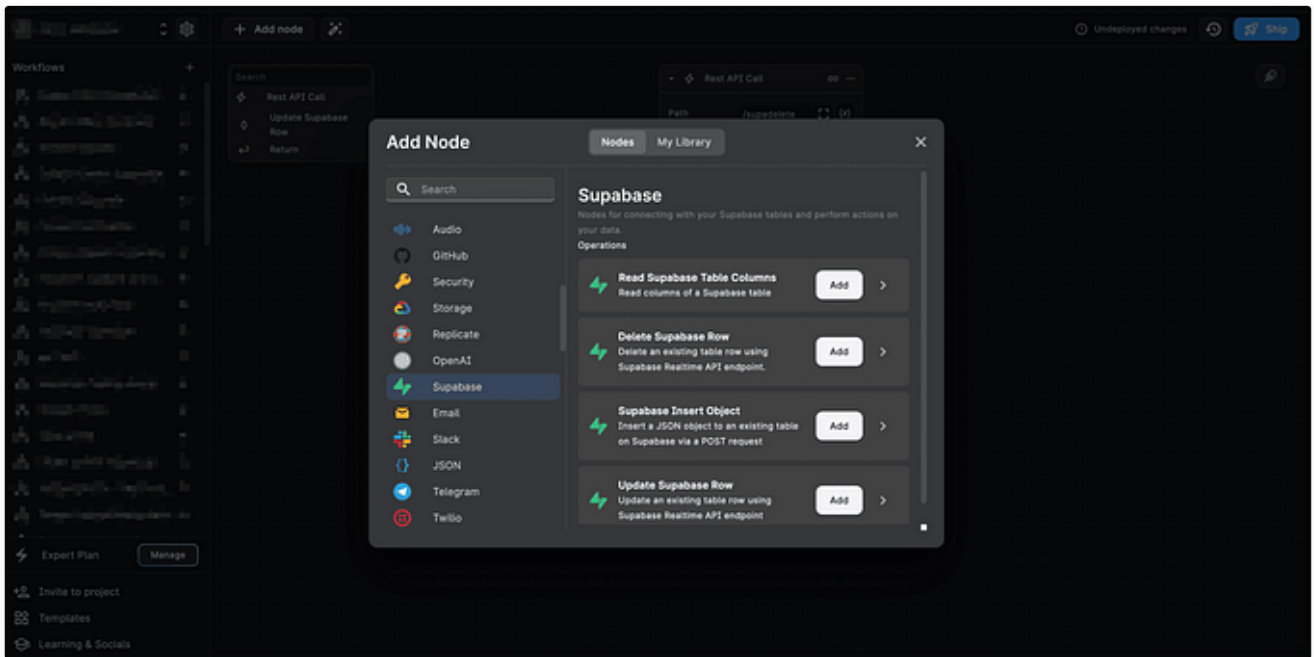
★ Dec 6



 Shiv Pal Singh Kaundal

Oct 18 🖐 1





 Girff

★ Nov 15



See more recommendations