Requirements

- Unix based OS (Linux, MacOS)
    - The install instructions were written for Ubuntu 18.04 LTS
- 4GB of ram
    - 8GB is recommended, although we have used as little as 2GB for development and 3GB for production deployments
    - In general, the application takes around 330 MB to run properly, but requires more ram when building the docker containers

Note for student installation testers

- Students in the capstone class who are testing these instructions do not need to follow the production installation instructions since they must be executed on a server which is accessible from a top-level domain on the public internet. (We spoke to Ibrahim about this)
- Students testers should follow the oauth setup instructions, however performing the actions is not necessary since oauth is already configured for spt-acas.com
- Students should stil add spt-acas.com to their /etc/hosts file (instructions in development install instructions).

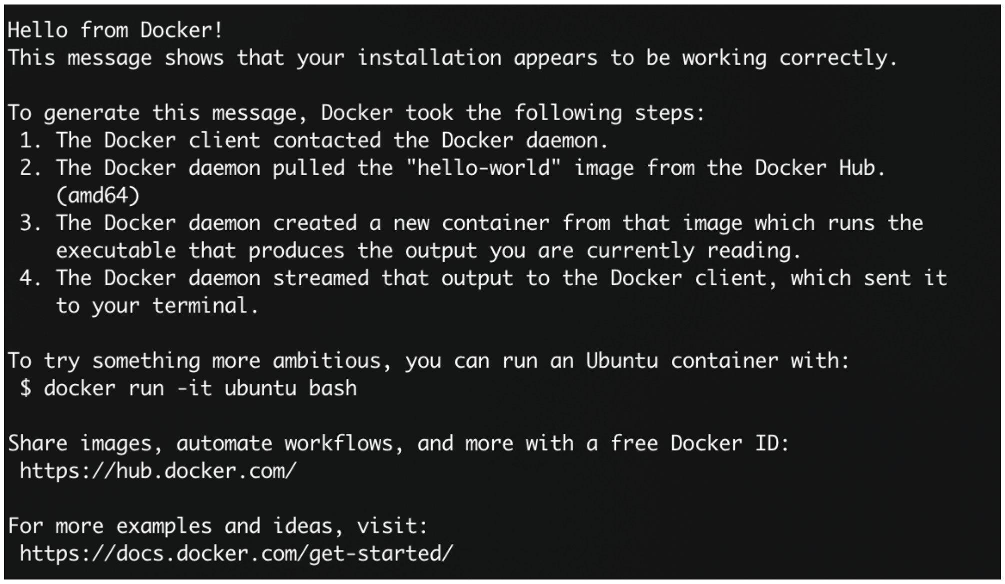
Setup

- Clone the repo
- Download Docker and docker-compose (Docker Install Instructions)
- Setup Google OAuth from the Google API Console (OAuth Setup Instructions)
- To install locally for development, follow the (Development Install Instructions)
- To install on a publically accessible production server, follow the (Production Install Instructions)

Installing Docker CE and Docker-Compose

1. Install Docker (for ubuntu)

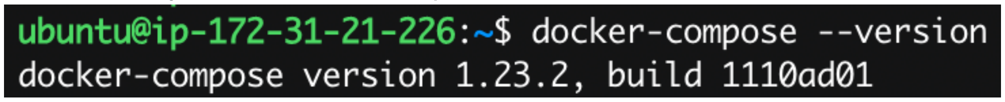
```
sudo apt update
sudo apt install docker.io
```

2. Test the installation of Docker
   - Run `sudo docker run hello-world` in the terminal
   - You will see this window on successful install
     - 

```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

3. Install Docker-Compose
   - Download the latest version
     - Run `sudo curl -L "https://github.com/docker/compose/releases/download/1.23.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose`
   - Apply permissions
     - Run `sudo chmod +x /usr/local/bin/docker-compose`
   - Create a symbolic link
     - Run `sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose`
   - Test the install
     - Run `docker-compose --version`

        - On success you will see an output similar to

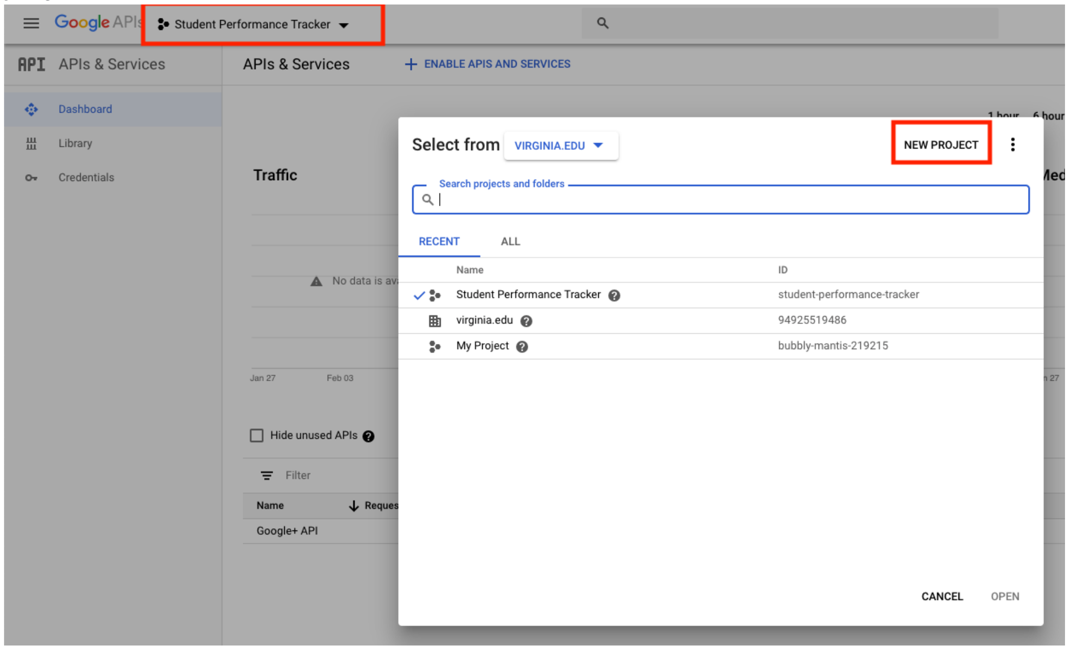


```
ubuntu@ip-172-31-21-226:~$ docker-compose --version
docker-compose version 1.23.2, build 1110ad01
```

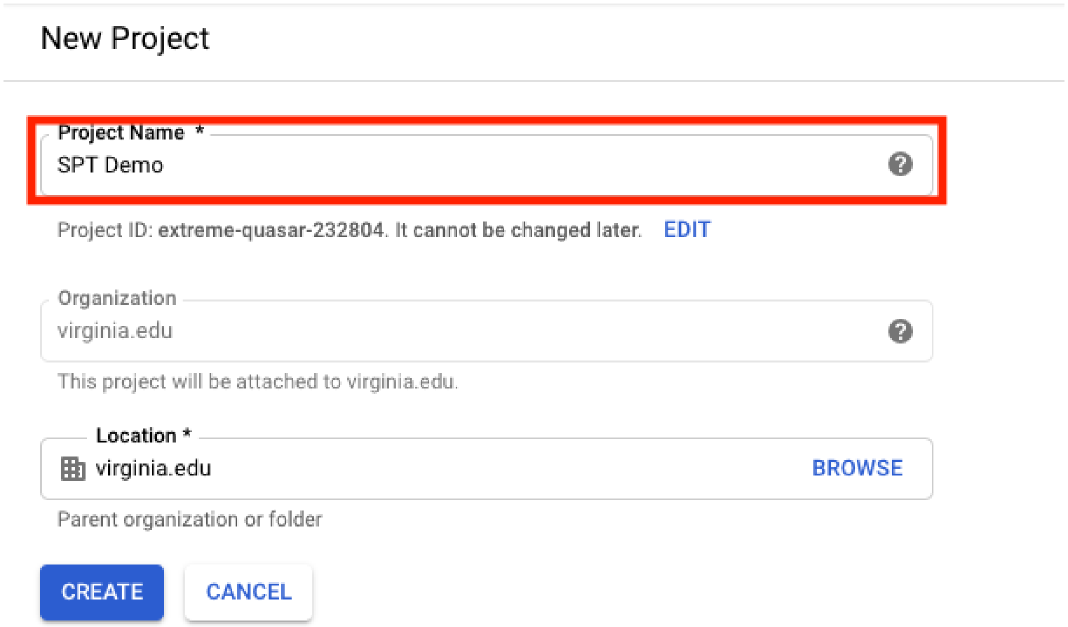
4. Configure Docker to run on boot
    - Run `sudo systemctl enable docker` in the terminal
5. Configure Docker to run as non-admin
    - Run `sudo usermod -aG docker ${USER}`
    - Logout and log back in

Setting up Google OAuth

For OAuth in development, use the domain `spt-acas.com` and add it to your /etc/hosts file so that it directs to localhost (instructions in development install instructions). For OAuth in production, use the domain that your server is accessible at.

1. Navigate to the Google API Console
   - Login using your gmail
2. Create a new project by clicking on the box next to *Google APIs* then selecting *new project*
   - 

Name	ID
Student Performance Tracker	student-performance-tracker
virginia.edu	94925519486
My Project	bubbly-mantis-219215
3. Name your new project and click create


New Project

Project Name *
SPT Demo

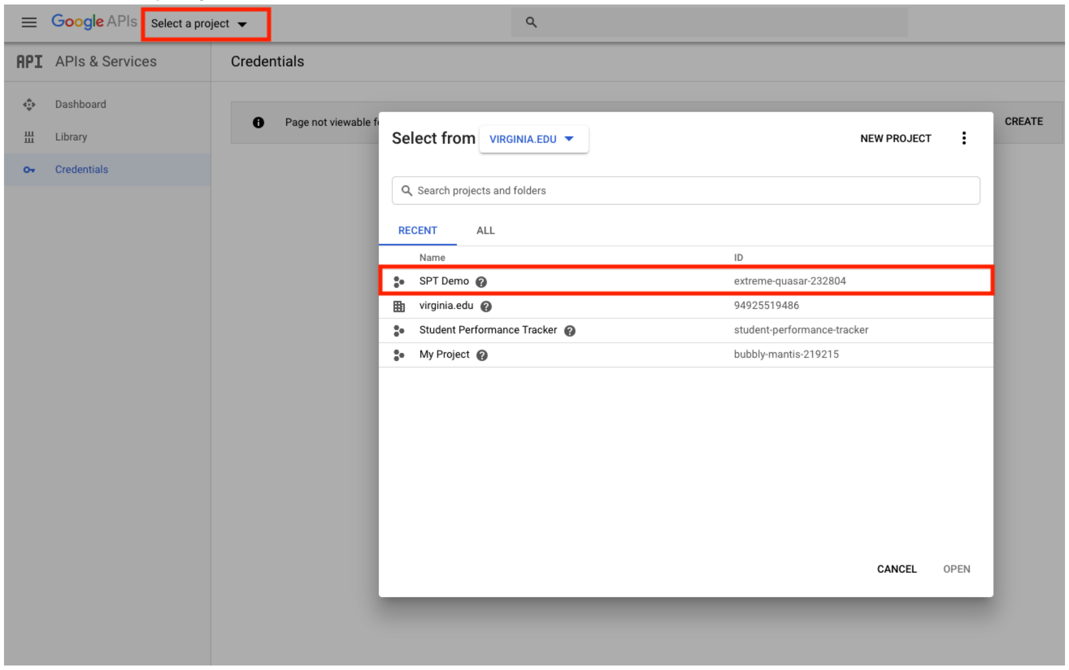
Project ID: extreme-quasar-232804. It cannot be changed later. EDIT

Organization
virginia.edu
This project will be attached to virginia.edu.

Location *
virginia.edu BROWSE
Parent organization or folder

◦

4. Select the new project

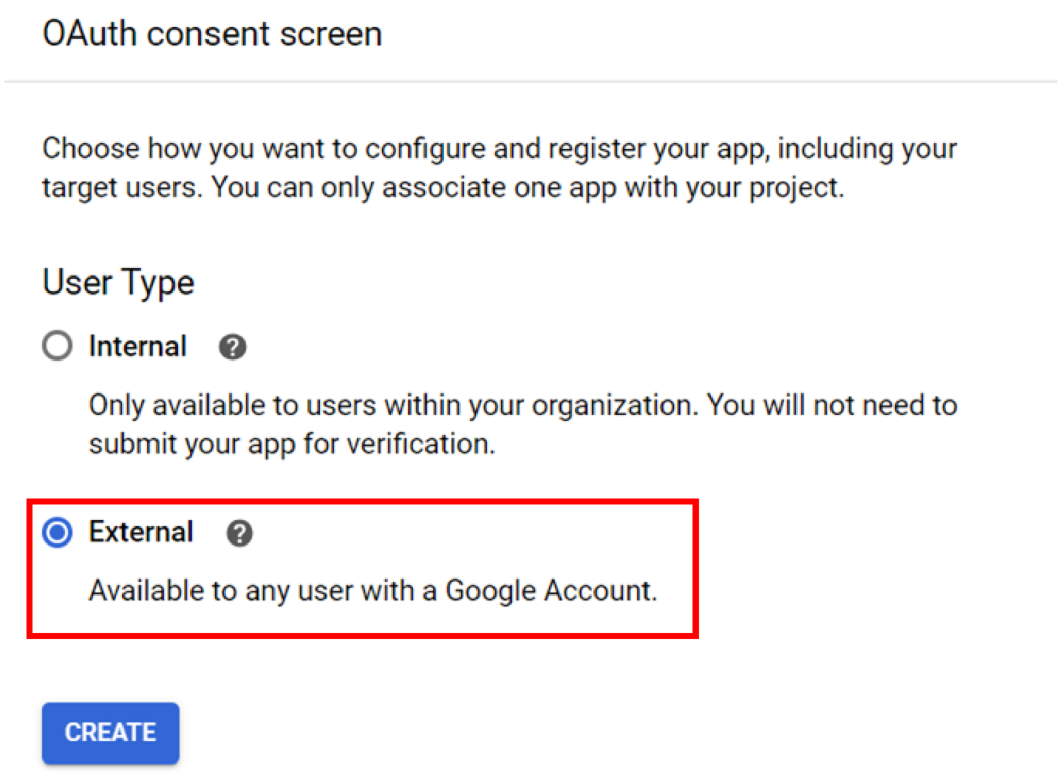


Name	ID
SPT Demo	extreme-quasar-232804
virginia.edu	94925519486
Student Performance Tracker	student-performance-tracker
My Project	bubbly-mantis-219215

◦

5. Select the *OAuth Consent Screen* tab on the left menu

- Select *External*.


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.

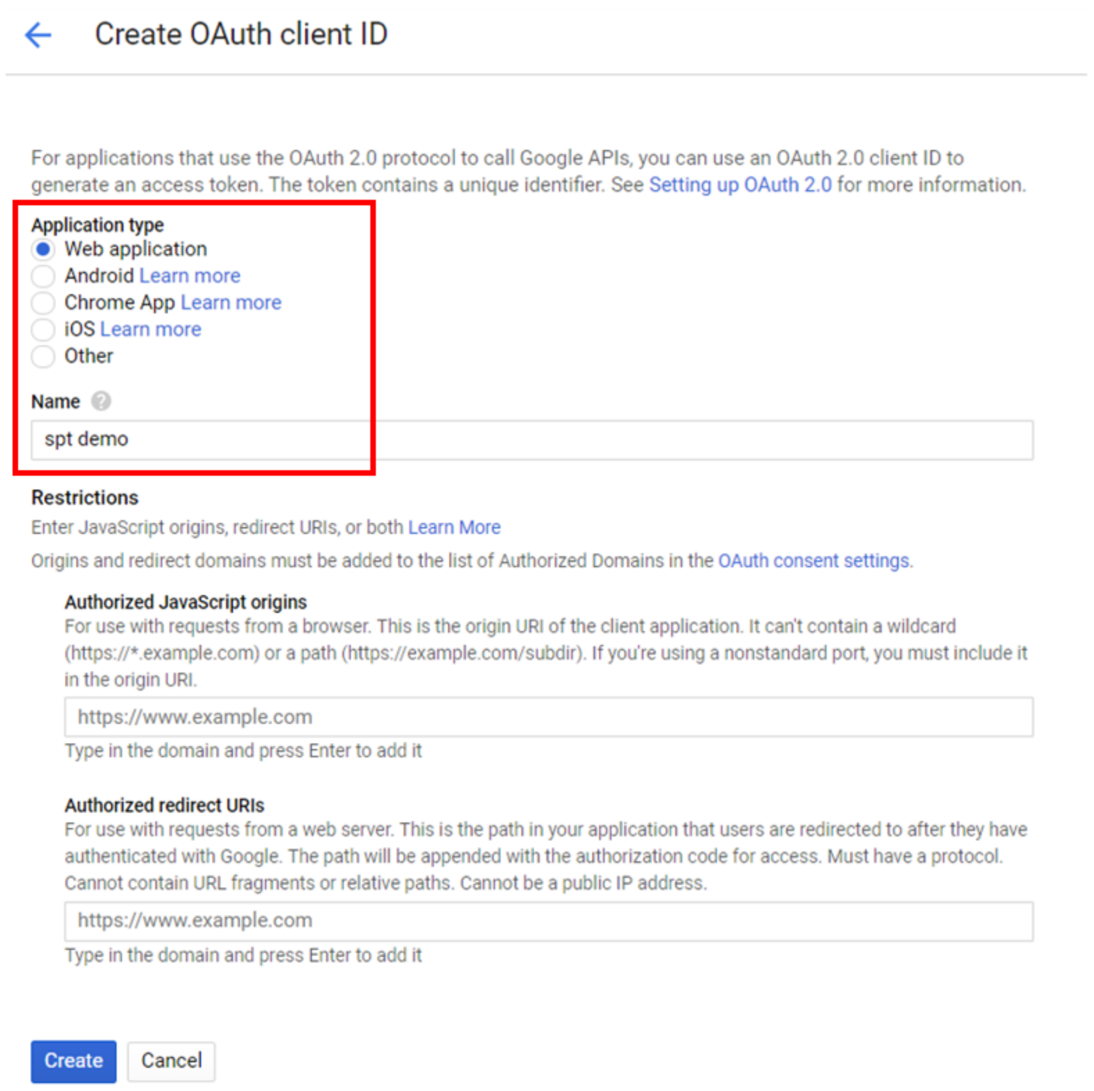
[✓ External]

Available to any user with a Google Account.

- Give the application a name
- Add the domain to the authorized domains list.
- Click save

7. Select the *Credentials* tab

- Click on *Create Credentials*
  - Select *OAuth client ID*
  - Select *Web Application*
    - Name it *SPT* or another name if desired



Create OAuth client ID

For applications that use the OAuth 2.0 protocol to call Google APIs, you can use an OAuth 2.0 client ID to generate an access token. The token contains a unique identifier. See Setting up OAuth 2.0 for more information.

Application type
- ● Web application
- ○ Android Learn more
- ○ Chrome App Learn more
- ○ iOS Learn more
- ○ Other

Name: spt demo

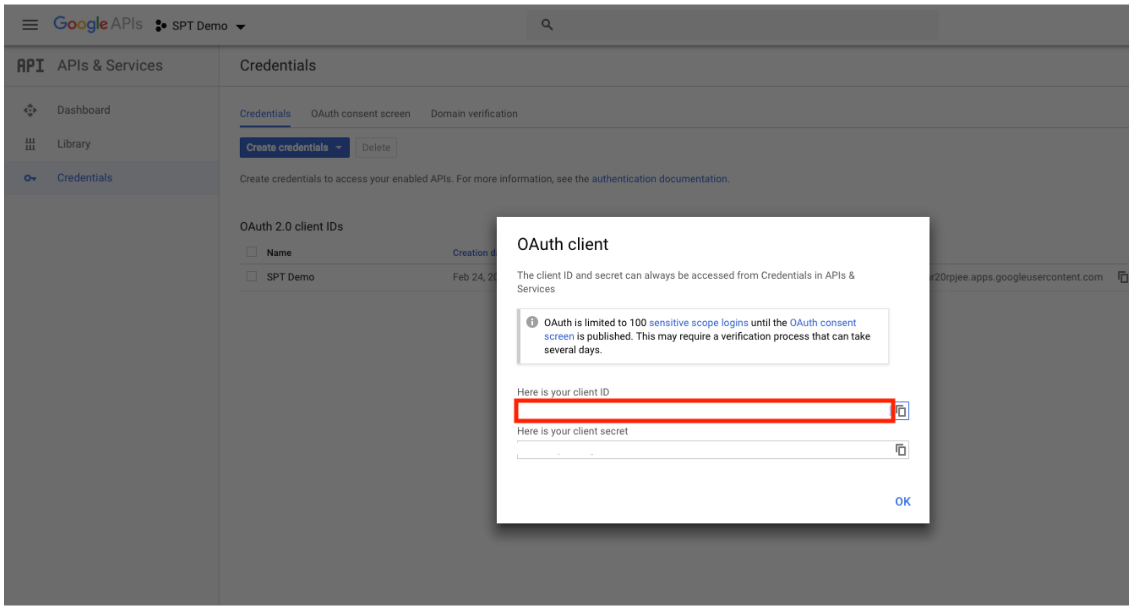
Restrictions
Enter JavaScript origins, redirect URIs, or both Learn More
Origins and redirect domains must be added to the list of Authorized Domains in the OAuth consent settings.

Authorized JavaScript origins
For use with requests from a browser. This is the origin URI of the client application. It can't contain a wildcard (https://*.example.com) or a path (https://example.com/subdir). If you're using a nonstandard port, you must include it in the origin URI.
https://www.example.com
Type in the domain and press Enter to add it

Authorized redirect URIs
For use with requests from a web server. This is the path in your application that users are redirected to after they have authenticated with Google. The path will be appended with the authorization code for access. Must have a protocol. Cannot contain URL fragments or relative paths. Cannot be a public IP address.
https://www.example.com
Type in the domain and press Enter to add it

-
- Add your domain to the *Authorized JavaScript Origins* and the *Redirect URLs*
  - If using the production deployment, use https, otherwise use http
- Click *Save*

8. You will be given a pop-up dialog, copy the *Client ID*

- 
Google APIs SPT Demo

APIs & Services — Credentials

Dashboard
Library
Credentials

Credentials | OAuth consent screen | Domain verification

Create credentials | Delete

Create credentials to access your enabled APIs. For more information, see the authentication documentation.

OAuth 2.0 client IDs

Name	Creation d
SPT Demo	Feb 24, 20

r20rpjee.apps.googleusercontent.com

OAuth client

The client ID and secret can always be accessed from Credentials in APIs & Services

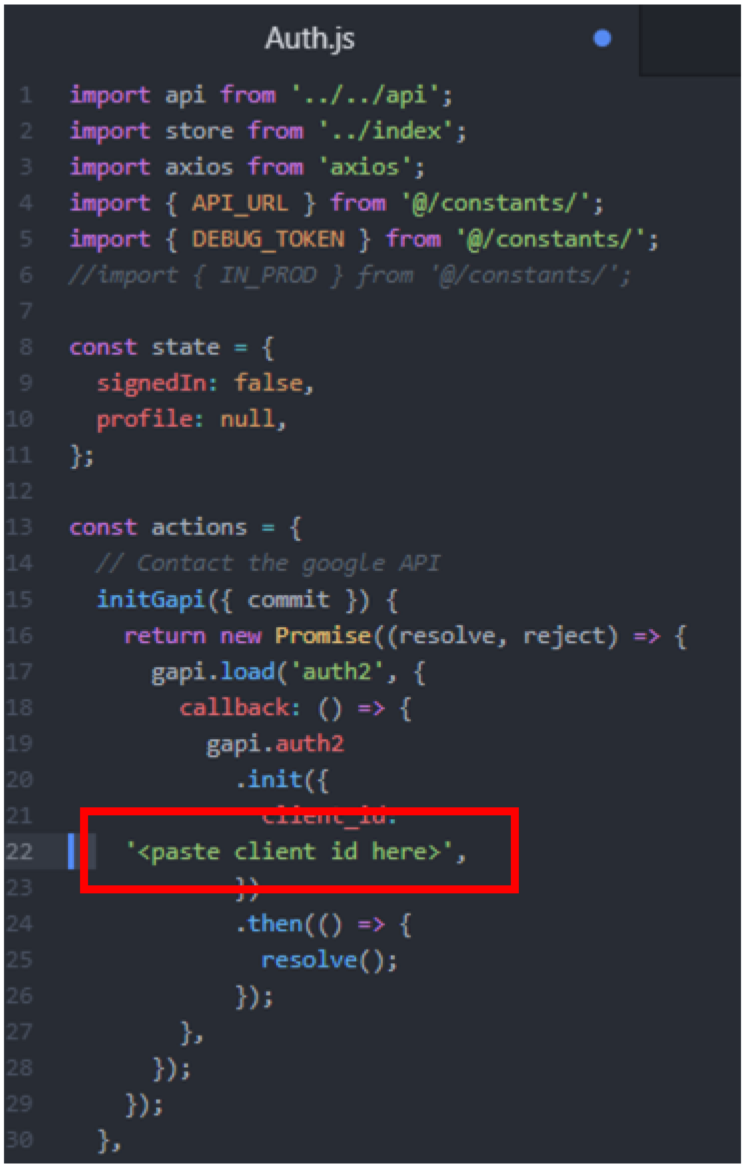
OAuth is limited to 100 sensitive scope logins until the OAuth consent screen is published. This may require a verification process that can take several days.

Here is your client ID

Here is your client secret

OK

- Open the project in an IDE
  - navigate to Auth.js
    - `src/frontend/src/vuex/modules/Auth.js`
    - Edit the show line to include your Client ID

- 

```
Auth.js
import api from '../../api';
import store from '../index';
import axios from 'axios';
import { API_URL } from '@/constants/';
import { DEBUG_TOKEN } from '@/constants/';
//import { IN_PROD } from '@/constants/';

const state = {
  signedIn: false,
  profile: null,
};

const actions = {
  // Contact the google API
  initGapi({ commit }) {
    return new Promise((resolve, reject) => {
      gapi.load('auth2', {
        callback: () => {
          gapi.auth2
            .init({
              client_id:
  '<paste client id here>',
            })
            .then(() => {
              resolve();
            });
        },
      });
    });
  },
```

- navigate to (../src/backend/sptApp/auth.py)
  - Edit the SERVER_CLIENT_ID to also match your Client ID

Development mode installation instructions

- To enable Google oauth locally in develop mode for sign-up and sign-in, add the following line to your /etc/hosts file (on linux or mac) `127.0.0.1 spt-acas.com`
- Naviage to src and run `docker-compose up`
  - Note: you may have to use sudo when executing docker-compose commmands
- Load the debug users into the database. While docker-compose is up, execute the following command in another terminal:

```
docker exec backend python3 manage.py loaddata debug_users.json
```

- Create an admin account with the following command:

```
docker exec -it backend python3 manage.py createsuperuser
```

The application is available at spt-acas.com:8080. The api is routed to spt-acas:8000. The django admin page is at spt-acas.com:8000/admin

The command `docker-compose down` will stop the containers but not delete the database. If you wish to delete the database, execute `docker-compose down --volumes`

Note: Accessing the application from spt-acas.com is done to allow oauth to work locally since Google oauth requires accessing the application from authorized top level domains. Accessing from spt-acas.com is not required for logging into the debug users. You can access the application from localhost:8080, but oauth will not work for regular sign in and account creation.

Running tests

Tests can be executed with the following command:

```
docker-compose -f docker-compose.test.yml up --exit-code-from backend
```

Additionally, you may execute the command ./test.sh if it has executable permissions. Note: The application must not be running before running tests.

Production installation instructions

- Note: The following commands must be ran on the production server itself which is accessable by a public domain on the internet
- Set DOMAIN in src/config/deployment_vars to the domain that you will be deploying on
- Setting the EMAIL variable will associate the certificates with your email, which is recommended by letsencrypt
- Inside the src folder, give executable permissions to .sh files with the command `chmod +x *.sh`
- Apply your configuration by executing the command `./apply_deployment_vars.sh`
- Build the production files with `docker-compose -f docker-compose.prod.yml build`
- If you wish to apply a different configuration, you must first run `./reset_deployment_vars.sh` before applying the new configuration. The build will need to be ran again.
    - Optionally, you can reset the files with `git checkout *`. Warning: This will lose all local changes to the src files.
- Collect certificates with the command `sudo ./init-letsencrypt.sh`
    - IMPORTANT: You must execute this command on a public server accessible at the domain specified in /src/config/deployment_vars
    - Port 80 must be open and not in use by another program
- Start the server with `docker-compose -f docker-compose.prod.yml up`