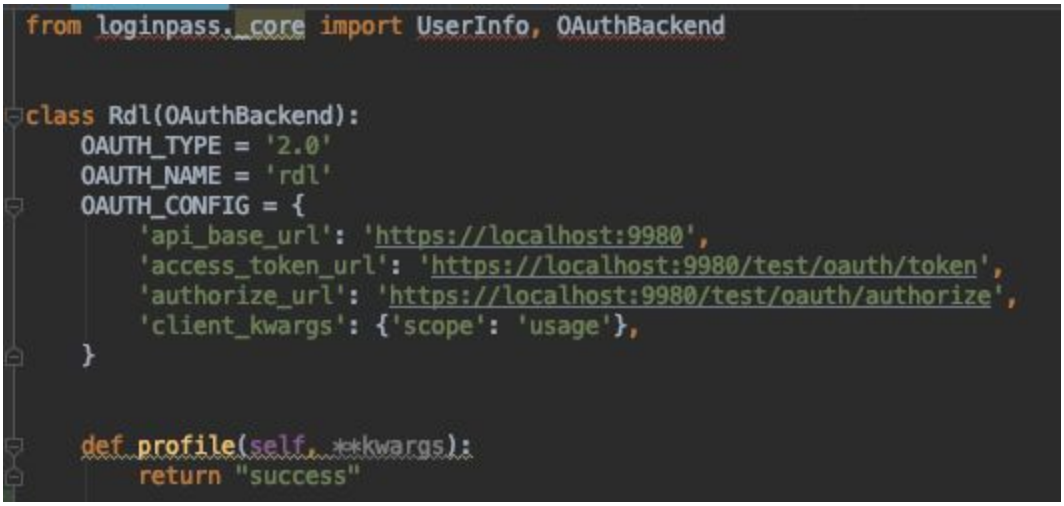
How to Set Up RDL as a Data Provider to Ancile

1. In Ancile's ancile_web/oauth directory, add a new file to store configuration data to help Ancile connect authenticate to RDL
    a. There should be an example file called 'cds.py' that can be copied, renamed, and modified. Here is an example 'rdl.py' file. Make sure class name has first letter capitalized and rest lowercase.



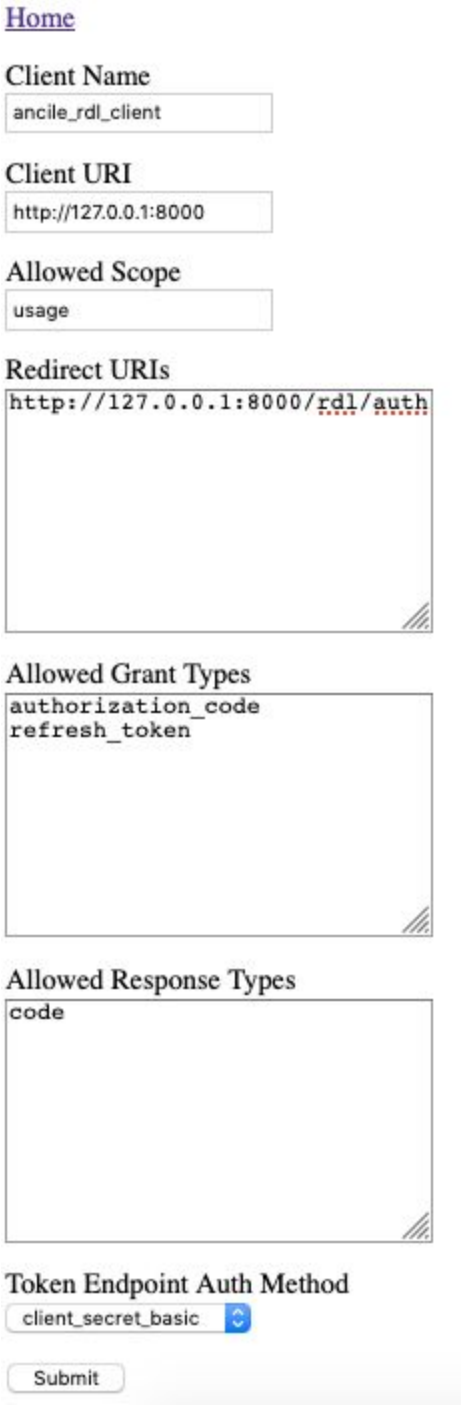
```
from loginpass._core import UserInfo, OAuthBackend


class Rdl(OAuthBackend):
    OAUTH_TYPE = '2.0'
    OAUTH_NAME = 'rdl'
    OAUTH_CONFIG = {
        'api_base_url': 'https://localhost:9980',
        'access_token_url': 'https://localhost:9980/test/oauth/token',
        'authorize_url': 'https://localhost:9980/test/oauth/authorize',
        'client_kwargs': {'scope': 'usage'},
    }


    def profile(self, **kwargs):
        return "success"
```

OAUTH_TYPE - Should be 2.0 since using OAuth 2.0
OAUTH_NAME - Name of data provider in Ancile's UI
api_base_url: Base RDL URL
access_token_url: RDL URL at which Ancile can request an access token
authorize_url: RDL URL at which Ancile can request an auth code (this is also where the user is asked to give Ancile permission to use RDL data)
scope: The name of the access scope that Ancile should request from RDL.

2. Launch RDL, sign in, and create a client for Ancile to use at the /test/ directory. Fill out the client creation form as below. After submitting the form, the client info - including client id and secret - will be displayed.



Client Name: Arbitrary name for the client that Ancile will use to set up OAuth with RDL.
Client URI: Ancile's URI. See note below in "Running Locally" about 127.0.0.1 vs localhost.
Allowed Scope: The name of the scope that Ancile is allowed to request. API endpoints on RDL will be configured to only send data when the request is coming from a client with specific scope(s).
Redirect URIs: Set to {Ancile URI}/rdl/auth

Allowed Grant Types: 'authorization_code' and 'refresh_token' need to be specified to allow Ancile to make these necessary requests.

Allowed Response Types: 'code' must be entered here so that RDL responds with an authorization code at the appropriate time in the OAuth process.

Token Endpoint Auth Method: client_secret_basic should be selected here as the other option is not fully supported by RDLs OAuth Library.

3. In Ancile's config/secret.yaml, add an OAuth client id and secret for Ancile to use to authenticate to RDL. These should be displayed at {RDL_URI}/test/ after completing the previous step.
    b. For example:

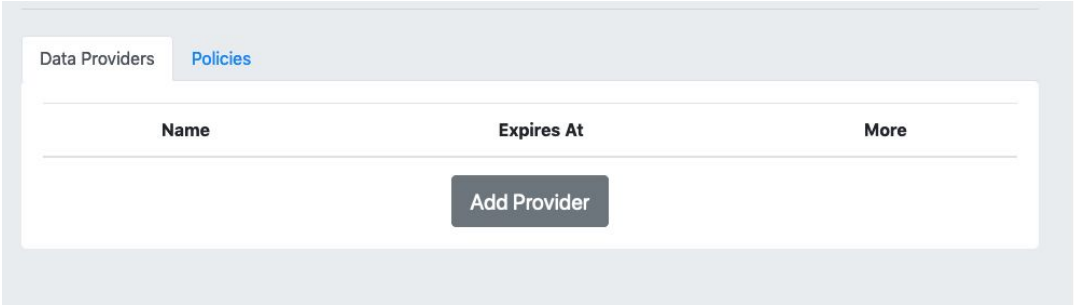
```
RDL_CLIENT_ID: dLImBdsbEln94
RDL_CLIENT_SECRET: jnrnSM                                    I6
```

4. In RDL, make sure the API endpoints that Ancile needs are configured with the scope entered in steps (1) and (2).
    a. API endpoints are in takeoutfilter/oauth_routes.py. Here is an example:

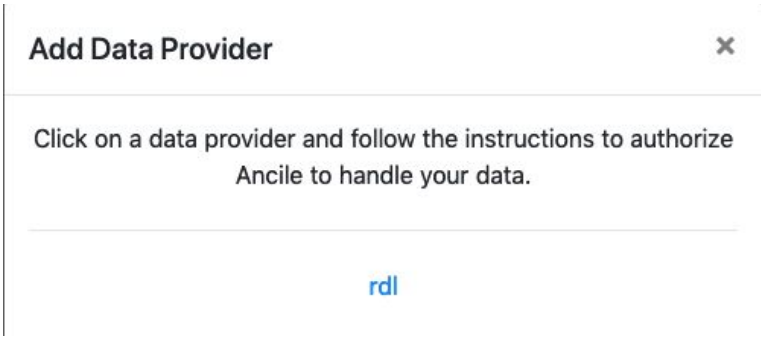
```
@bp.route('/api/usage')
@require_oauth('usage')
def usage_stats():
    user = current_token.user
    return usage(user)
```

    This code creates an endpoint at {RDL URI}/api/usage and requires clients to have the usage scope. It returns the output of the usage() function to the client.

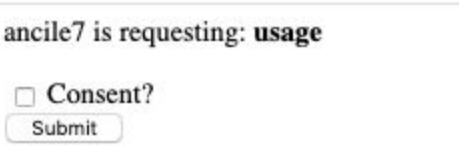
5. Restart RDL
    a. 'circusctl restart || circusd circus.ini'
6. Start Ancile
    a. 'sh scripts/start_server.sh'
7. Login to Ancile, and add RDL as a data provider
    a. Press "Add Provider" in the User Control Panel



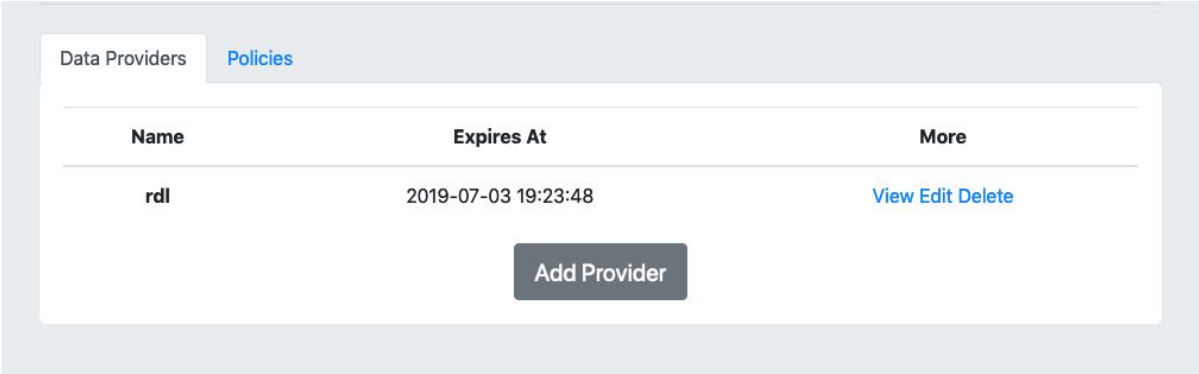
    b. Select 'rdl' from the list



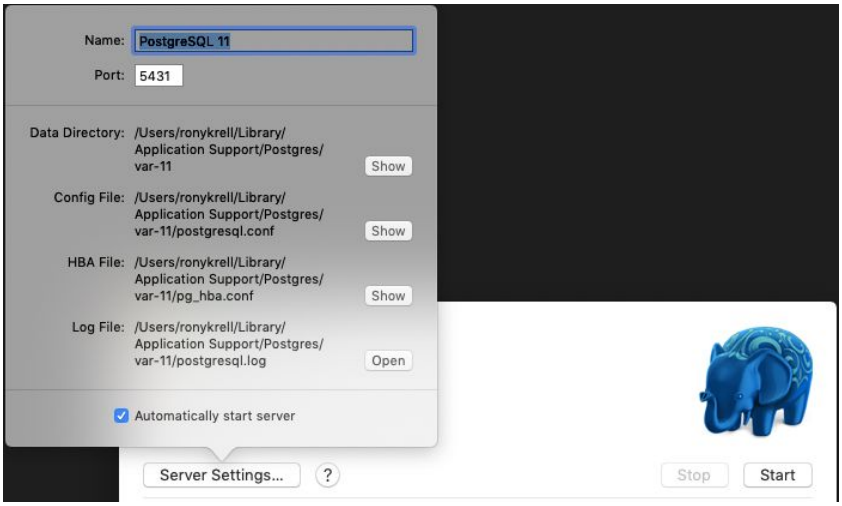
c. Login to RDL via Google OAuth and grant RDL permission to access Google Account data. Make sure to use the same Google Account that was logged into RDL earlier when registering Ancile's OAuth client.
d. Check the "Consent?" checkbox and press Submit



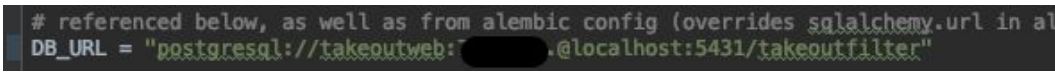
e. RDL added as Ancile Data Provider



8. Running Locally
   If running Ancile and RDL locally, additional adjustments need to be made
   a. Both Ancile and RDL need to run Postgres and Redis. Configure either one to run Postgres and Redis on non-default ports to prevent conflicts. Below is an example for modifying RDL services to run on alternate ports.
      i. Redis
         1. Start redis on a different port 'redis-server --port 6378'
         2. In takeoutfilter/celery_init.py, change the port number for the CELERY_BROKER_URL and CELERY_RESULT_BACKEND properties
      ii. Postgres
         1. If running the Postgres app on OSX, change the Postgres port on the app UI



2. In takeoutfilter/models/database.py, add the alternate port number to the DB_URL property



b. Run one of the apps on localhost and the other one on 127.0.0.1. Even though these are equivalent, this configuration change prevents one app from overwriting the other's cookies. Without this change, Ancile sessions will be lost and OAuth will not complete successfully
   i. To have Ancile run on 127.0.0.1, modify the command in scripts/start_server.sh

```
#!/usr/bin/bash

source .env/bin/activate;
gunicorn runner:app -b 127.0.0.1:8000 -w 1
```

c. If RDL is running locally with https using a self-signed certificate, Ancile will throw an INVALID_CERT error since the certificate is not signed by a trusted CA. To get around this, add the certificate to Ancile's CA certificate store in the virutal env. For example, using python 3.6.8, the file is located at: .env/lib/python3.6/site-packages/certifi/cacert.pem

d. Debugging tip - to debug using PyCharm,
   i. Run Ancile/RDL
   ii. In PyCharm, open up the appropriate project
   iii. Go to "Run" -> "Attach to Process".
   iv. Select the process for app to debug, and PyCharm will then break on breakpoints set in the IDE

v. If debugging Ancile, it might be helpful to limit the number of gunicorn workers to 1 in order to know which one will be picking up our requests. This can be done in the scripts/start_server.sh file as shown above.