This tutorial shows you how to retrieve, verify, and store third-party credentials using Identity Platform, the App Engine standard environment, and Datastore.

This document walks you through a simple note-taking application called Firenotes that stores users' notes in their own personal notebooks. Notebooks are stored per user, and identified by each user's unique Identity Platform ID. The application has the following components:

- The frontend configures the sign-in user interface and retrieves the Identity Platform ID. It also handles authentication state changes and lets users see their notes.
- FirebaseUI (https://github.com/firebase/FirebaseUi-Web) is an open-source, drop-in solution thatworks with Identity Platform to simplify implementation of complex authentication and UI tasks. The SDK handles user login, linking multiple providers to one account, recovering passwords, and more. It implements authentication best practices for a smooth and secure sign-in experience.
- The backend verifies the user's authentication state and returns user profile information as well as the user's notes.

The application stores user credentials in Datastore by using the NDB client library (/appengine/docs/standard/python/ndb/), but you can store the credentials in a database of your choice.

Firenotes is based on the Flask (http://flask.pocoo.org/) web application framework. The sample app uses Flask because of its simplicity and ease of use, but the concepts and technologies explored are applicable regardless of which framework you use.

By completing this tutorial, you'll accomplish the following:

- Configure the user interface with FirebaseUI for Identity Platform.
- Obtain a Identity Platform ID token and verify it using server-side authentication.
- Store user credentials and associated data in Datastore.
- Query a database using the NDB client library.
- Deploy an app to App Engine.

This tutorial uses billable components of Google Cloud, including:

- Datastore
- Identity Platform

Use the Pricing Calculator (/products/calculator/) to generate a cost estimate based on your projected usage. New Google Cloud users might be eligible for a free trial (/free-trial).

1. Install Git (https://git-scm.com/book/en/v2/Getting-Started-Installing-Git/), Python 2.7 (https://www.python.org/), and virtualenv (/python/setup#installing_and_using_virtualenv). For more information on setting up your Python development environment, such as installing the latest version of Python, refer to Setting Up a Python Development Environment (/python/setup) for Google Cloud.
2. Sign in (https://accounts.google.com/Login) to your Google Account.

   If you don't already have one, sign up for a new account (https://accounts.google.com/SignUp).

3. In the Cloud Console, on the project selector page, select or create a Cloud project.

★ **Note**: If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

Go to the project selector page (https://console.cloud.google.com/projectselector2/home/dashboard)

4. Install and initialize the Cloud SDK (/sdk/docs/).

If you have already installed and initialized the SDK to a different project, set the `gcloud` project to the App Engine project ID you're using for Firenotes. See Managing Cloud SDK Configurations (/sdk/docs/managing-configurations/) for specific commands to update a project with the `gcloud` tool.

To download the sample to your local machine:

1. Clone the sample application repository to your local machine:

   Alternatively, you can download the sample (https://github.com/GoogleCloudPlatform/python-docs-samples/archive/master.zip) as a zip file and extract it.

2. Navigate to the directory that contains the sample code:

To configure FirebaseUI for Identity Platform and enable identity providers:

1. Add Identity Platform to your app by following these steps:

   a. Go to the Cloud Console (https://console.cloud.google.com/).
   Go to the Cloud Console (https://console.cloud.google.com/)

   b. Select the Google Cloud project you want to use:

   - If you have an existing project, select it on the **Select organization** drop-down list at the top of the page.
   - If you don't have an existing Google Cloud project, create a new project (https://console.cloud.google.com/projectcreate) in the Cloud Console.

   c. Go to the Identity Platform Marketplace (https://console.cloud.google.com/marketplace/details/google-cloud-platform/customer-identity) page in the Cloud Console.
   Go to the Identity Platform Marketplace page (https://console.cloud.google.com/marketplace/details/google-cloud-platform/customer-identity)

   d. On the Identity Platform Marketplace page, click **Enable Customer Identity**.

   e. Go to the Customer Identity Users (https://console.cloud.google.com/customer-identity/users) page in the Cloud Console.
   Go to the Users page (https://console.cloud.google.com/customer-identity/users)

f. On the top right, click application **setup details**.

g. Copy the application setup details into your Web Application.

appengine/standard/firebase/firenotes/frontend/main.js
(https://github.com/GoogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/frontend/main.js)

oogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/frontend/main.js)

2. Edit the `backend/app.yaml` file and enter your Google Cloud project ID in the environment variables:

appengine/standard/firebase/firenotes/backend/app.yaml
(https://github.com/GoogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/backend/app.yaml)

ogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/backend/app.yaml)

3. In the `frontend/main.js` file, configure the FirebaseUI login widget (https://github.com/firebase/FirebaseUI-Web) by selecting which providers you want to offer your users.

appengine/standard/firebase/firenotes/frontend/main.js
(https://github.com/GoogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/frontend/main.js)

oogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/frontend/main.js)

4. In the Cloud Console, enable the providers you chose to keep:

    a. Go to the Customer Identity Providers (https://console.cloud.google.com//customer-identity/providers) page in the Cloud Console.
    Go to the Providers page (https://console.cloud.google.com/customer-identity/providers)

    b. Click **Add A Provider**.

    c. On the **Select a provider** drop-down list, select the providers you want to use.

    d. Next to **Enabled**, click the button to enable the provider.

    - For third-party identity providers, enter the provider ID and secret from the provider's developer site. The Firebase docs give specific instructions in the "Before you begin" sections of the Facebook (https://firebase.google.com/docs/auth/web/facebook-login), Twitter (https://firebase.google.com/docs/auth/web/twitter-login), and GitHub (https://firebase.google.com/docs/auth/web/github-auth) guides.
    - For SAML and OIDC integrations, refer to the configuration at your IdP.

5. Add your domain to the list of authorized domains in Identity Platform:

    a. Go to the Customer Identity Settings (https://console.cloud.google.com//customer-identity/settings) page in the Cloud Console.
    Go to the Settings page (https://console.cloud.google.com/customer-identity/settings)

    b. Under **Authorized Domains**, click **Add Domain**.

    c. Enter the domain of your app in the following format:

    Don't include `http://` before the domain name.

1. Navigate to the `backend` directory and complete the application setup:

2. Install the dependencies into a `lib` directory in your project:

3. In `appengine_config.py`, the `vendor.add()` method registers the libraries in the `lib` directory.

To run the application locally, use the App Engine local development server:

1. Add the following URL as the `backendHostURL` in `main.js`:

   `http://localhost:8081`

2. Navigate to the root directory of the application. Then, start the development server:

3. Visit http://localhost:8080/ (http://localhost:8080/) in a web browser.

Now that you have set up a project and initialized an application for development, you can walk through the code to understand how to retrieve and verify Identity Platform ID tokens on the server.

The first step in server-side authentication is retrieving an access token to verify. Authentication requests are handled with the `onAuthStateChanged()` listener from Identity Platform:

appengine/standard/firebase/firenotes/frontend/main.js
(https://github.com/GoogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/frontend/main.js)

oogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/frontend/main.js)

When a user is signed in, the Identity Platform `getToken()` method in the callback returns a Identity Platform ID token in the form of a JSON Web Token (JWT).

After a user signs in, the frontend service fetches any existing notes in the user's notebook through an AJAX `GET` request. This requires authorization to access the user's data, so the JWT is sent in the `Authorization` header of the request using the `Bearer` schema:

appengine/standard/firebase/firenotes/frontend/main.js (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/frontend/main.js)

oogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/frontend/main.js)

Before the client can access server data, your server must verify the token is signed by Identity Platform. You can verify this token using the Google Authentication Library for Python (http://google-auth.readthedocs.io/). Use the authentication library's `verify_firebase_token` (http://google-auth.readthedocs.io/en/stable/reference/google.oauth2.id_token.html#google.oauth2.id_token.verify_firebase_token) function to verify the bearer token and extract the claims:

appengine/standard/firebase/firenotes/backend/main.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/backend/main.py)

oogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/backend/main.py)

Each identity provider sends a different set of claims, but each has at least a `sub` claim with a unique user ID and a claim that provides some profile information, such as `name` or `email`, that you can use to personalize the user experience on your app.

After authenticating a user, you need to store their data for it to persist after a signed-in session has ended. The following sections explain how to store a note as a Datastore *entity* and segregate entities by user ID.

You can create an entity in Datastore by declaring an NDB model class (/appengine/docs/standard/python/ndb/creating-entity-models/) with certain properties such as integers or strings. Datastore indexes entities by *kind*; in the case of Firenotes, the kind of each entity is `Note`. For

querying purposes, each `Note` is stored with a *key name*, which is the user ID obtained from the `sub` claim in the previous section.

The following code demonstrates how to set properties of an entity, both with the constructor method for the model class when the entity is created and through assignment of individual properties after creation:

appengine/standard/firebase/firenotes/backend/main.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/backend/main.py)

oogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/backend/main.py)

To write the newly created `Note` to Datastore, call the `put()` method on the `note` object.

To retrieve user data associated with a particular user ID, use the NDB `query()` method to search the database for notes in the same entity group. Entities in the same group, or *ancestor path* (/appengine/docs/standard/python/datastore/entities#Python_Ancestor_paths), share a common key name, which in this case is the user ID.

appengine/standard/firebase/firenotes/backend/main.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/backend/main.py)

oogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/backend/main.py)

You can then fetch the query data and display the notes in the client:

appengine/standard/firebase/firenotes/frontend/main.js
(https://github.com/GoogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/frontend/main.js)

oogleCloudPlatform/python-docs-samples/blob/6f5f3bcb81779679a24e0964a6c57c0c7deabfac/appengine/standard/firebase/firenotes/frontend/main.js)

You have successfully integrated Identity Platform with your App Engine application. To see your application running in a live production environment:

1. Change the backend host URL in `main.js` to `https://backend-dot-[PROJECT_ID].appspot.com`. Replace `[PROJECT_ID]` with your project ID.
2. Deploy the application using the Cloud SDK command-line interface:
3. View the application live at `https://[PROJECT_ID].appspot.com`.

★ **Note:** Datastore indexes take a few minutes to update, so the application might not be fully functional immediately after deployment.

To avoid incurring charges to your Google Cloud account for the resources used in this tutorial, delete your App Engine project:

The easiest way to eliminate billing is to delete the project that you created for the tutorial.

To delete the project:

Caution: Deleting a project has the following effects:

- **Everything in the project is deleted.** If you used an existing project for this tutorial, when you delete it, you also delete any other work you've done in the project.
- **Custom project IDs are lost.** When you created this project, you might have created a custom project ID that you want to use in the future. To preserve the URLs that use the project ID, such as an **`appspot.com`** URL, delete selected resources inside the project instead of deleting the whole project.

If you plan to explore multiple tutorials and quickstarts, reusing projects can help you avoid exceeding project quota limits.

1. In the Cloud Console, go to the **Manage resources** page.

   Go to the Manage resources page (https://console.cloud.google.com/iam-admin/projects)

2. In the project list, select the project you want to delete and click **Delete** .
3. In the dialog, type the project ID, and then click **Shut down** to delete the project.

- Set up Eclipse for development on App Engine (/appengine/docs/standard/python/tools/setting-up-eclipse)
- Try out other Google Cloud features for yourself. Have a look at our tutorials (/docs/tutorials).