Authenticating end users

Most applications handle requests from end users, and it's a best practice to restrict access to only the allowed end users. In order to accomplish this, you can integrate Google Sign-In and grant users (https://cloud.google.com/run/docs/securing/managing-access) the `roles/run.invoker` (https://cloud.google.com/run/docs/reference/iam/roles) IAM role, or implement Firebase Authentication and manually validate their credentials.

Note that Cloud Run does not assist in the sharing of sessions between container instances and does not guarantee session affinity to a specific container instance.

Using Google Sign-In

First, you'll need to enable Google Sign-In in your project:

1. Create an OAuth 2.0 client ID for your app in the same project as the service you want to secure:
    a. Go to the Credentials (https://console.cloud.google.com/apis/credentials) page.

    GO TO THE CREDENTIALS PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/APIS/CREDENTIA

    b. Select the project with the service you want to secure.
    c. Click **Create credentials**, then select **OAuth Client ID**.
        i. You may be required to configure your OAuth consent screen before creating a client ID. If necessary, do so in order to continue.
    d. Select the **Application type** for which you want to create credentials.
    e. Add a **Name** and **Restrictions** if appropriate, then click **Create**.
2. Re-deploy the service you want to secure. This will ensure that the correct client ID is set on the service.

If you have multiple OAuth client IDs (for example, one each for Android, iOS, and web), you must re-deploy your service(s) after adding each one to ensure the service picks up the change.

Similarly, if you delete a client ID, you must re-deploy your service(s) to remove that client ID and deny requests. All client IDs within a project will be accepted.

In your web or mobile app, you'll need to:

1. Get an ID token for the OAuth client ID:
    - **Android**: Use Google Sign-In (https://developers.google.com/identity/sign-in/android/) to get an ID token.
    - **iOS**: Use Google Sign-In (https://developers.google.com/identity/sign-in/ios/backend-auth) to get an ID token.
    - **Web**: Use Google Sign-In (https://developers.google.com/identity/sign-in/web/) to get an ID token.
2. Include the ID token in an `Authorization: Bearer ID_TOKEN` header in the request to the service.

Cloud Run will validate the auth token and allow the request, or reject the request before the service starts up. If a request is rejected, you won't be billed for that request.

Getting user profile information

If you want to access user profile information, you can pull the token out of the `Authorization` header and make a request to the `Validate Token` endpoint (https://developers.google.com/identity/sign-in/web/backend-auth#verify-the-integrity-of-the-id-token).

The body of the ID token should be returned with the following information:

```
{
 // These six fields are included in all Google ID Tokens.
 "iss": "https://accounts.google.com",
 "sub": "110169484474386276334",
 "azp": "1008719970978-hb24n2dstb40o45d4feuo2ukqmcc6381.apps.googleusercontent.com",
 "aud": "1008719970978-hb24n2dstb40o45d4feuo2ukqmcc6381.apps.googleusercontent.com",
 "iat": "1433978353",
 "exp": "1433981953",

 // These seven fields are only included when the user has granted the "profile"
 // and "email" OAuth scopes to the application.
 "email": "testuser@gmail.com",
 "email_verified": "true",
 "name" : "Test User",
```

```
 "picture": "https://lh4.googleusercontent.com/-kYgzyAWpZzJ/ABCDEFGHI/AAAJKLMNOP/tIX
 "given_name": "Test",
 "family_name": "User",
 "locale": "en"
}
```

Troubleshooting

If user requests are being rejected and you believe they should be allowed, ensure that users have been granted (https://cloud.google.com/run/docs/securing/managing-access) the `roles/run.invoker` role, or have the `run.routes.invoke` permission. Learn more about these in the Cloud Run IAM reference (https://cloud.google.com/run/docs/reference/iam).

Using Identity Platform or Firebase Authentication

If you want to authenticate users using email/password, phone number, social providers like Facebook or GitHub, or a custom authentication mechanism, you can use Firebase Authentication (https://firebase.google.com/docs/auth) or Identity Platform (https://cloud.google.com/identity-platform/docs/).

Note: Unlike Google Sign-in above, your service is doing the authentication; therefore, you will be billed for unauthenticated requests since the service must do work to validate the token.

We'll show how to use Firebase Authentication, but the steps are similar for Identity Platform.

First you'll need to set up Firebase Authentication in your project and service:

1. Set up Firebase Authentication in the Firebase Console.

   GO TO THE FIREBASE CONSOLE (HTTPS://CONSOLE.FIREBASE.GOOGLE.COM/PROJECT/_/AUTHEN

2. Import the appropriate Firebase Admin SDK (https://firebase.google.com/docs/admin/setup) and configure it properly.
3. Add middleware to your code to verify Firebase ID tokens (https://firebase.google.com/docs/auth/admin/verify-id-tokens#verify_id_tokens_using_the_firebase_admin_sdk)

   .

4. Deploy your service publicly.

In your web or mobile app, you need to:

1. Use the appropriate Firebase Auth client library to get an ID token:
    - **Android**: Use the `GetTokenResult().getToken()` (https://firebase.google.com/docs/reference/android/com/google/firebase/auth/GetTokenResult) method.
    - **iOS**: Use the `User.getIDTokenResult(completion:)` (https://firebase.google.com/docs/reference/swift/firebaseauth/api/reference/Classes/User#getidtokenresultcompletion) method.
    - **Web**: Use the `firebase.User.getIdToken()` (https://firebase.google.com/docs/reference/js/firebase.User#getIdToken) method.
2. Include the ID token in an `Authorization: Bearer ID_TOKEN` header in the request to the service.

Getting user profile information

If you want to access user profile information, you can use the Firebase Admin SDK to retrieve user data (https://firebase.google.com/docs/auth/admin/manage-users#retrieve_user_data).

Sample code tutorial for Cloud Run for Anthos on Google Cloud

For a tutorial on authenticating end users for Cloud Run for Anthos on Google Cloud, refer to the tutorial Authenticating end users on Cloud Run on GKE (https://cloud.google.com/solutions/authenticating-cloud-run-on-gke-end-users-using-istio-and-identity-platform)

.

Last updated November 14, 2019.