

Okta integration guide: Google Cloud endpoints

November, 2017

Overview

Google Cloud Endpoints (GCE) integrates with Okta for user authentication via OAuth. You can set up Okta as a token (jwt) provider for your API, and GCE will enforce token validation in front of your API.

Prerequisites

- An API running on GCE. If you don't yet have one up and running, you can follow the instructions here to set up a sample API:
 - <https://cloud.google.com/endpoints/docs/openapi/get-started-app-engine>
 - Before proceeding, you should be able to get a "hello world" response from the API deployed on GCE
 - This setup guide will walk you through the process of setting up Okta as an authentication provider for GCE.
 - The following Google page is also a helpful reference:
 - <https://cloud.google.com/endpoints/docs/openapi/authenticating-users>
 - This page contains more detail about the token validation process
- An Okta tenant
 - Set up an OIDC client (described below)
- A web app/server
 - An application that can accept a token coming back from Okta

Setup

Set up an OIDC application in your Okta tenant

1. Set up an OIDC client
 - a. For Okta Enterprise edition: Applications->Add Application->Create New App->OpenID Connect

Create a New Application Integration



Platform

Web

Sign on method

- Secure Web Authentication (SWA)
Uses credentials to sign in. This integration works with most apps.
- SAML 2.0
Uses the SAML protocol to log users into the app. This is a better option than SWA, if the app supports it.
- OpenID Connect
Uses the OpenID Connect protocol to log users into an app you've built.

Create

Cancel

Click "create"


Create OpenID Connect Integration


GENERAL SETTINGS

Application name

Application logo (Optional) 

CONFIGURE OPENID CONNECT

Login redirect URIs 

Logout redirect URIs 

The Login redirect URI should be an application hosted on a web server that can accept an incoming access token from Okta. A barebones page running on localhost is fine.

Click "save" and you will have your client_id and client_secret.

On the General Settings screen, check the box for Implicit grant type, and check the boxes for allowing both the id token and the access token.

General Sign On Assignments Registration

General Settings Cancel

APPLICATION

Application label

Application type Web

Allowed grant types

- Client Credentials
- Authorization Code
- Refresh Token
- Implicit (Hybrid)
 - Allow ID Token with implicit grant type
 - Allow Access Token with implicit grant type

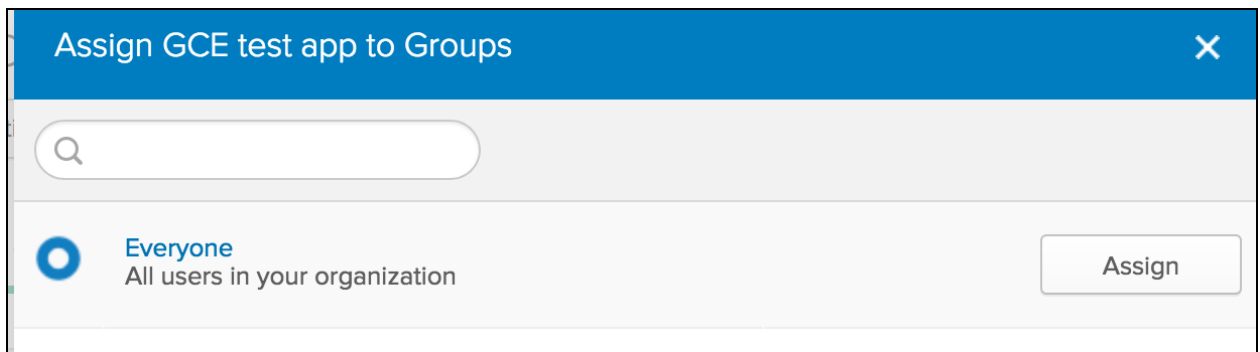
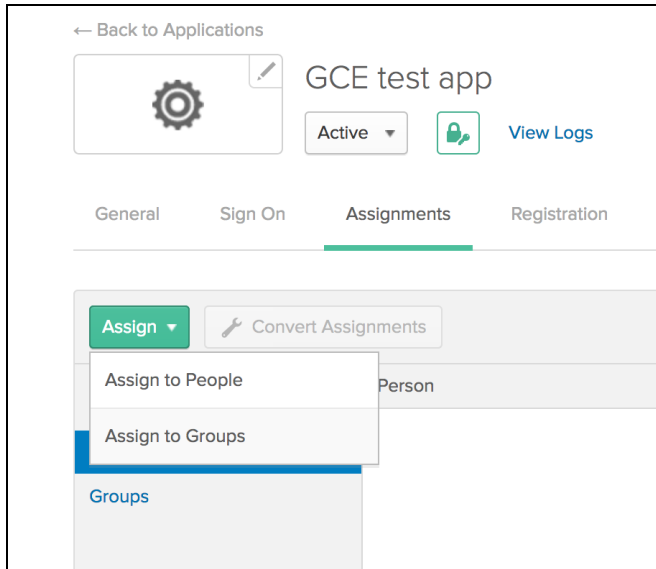
LOGIN

Login redirect URIs ? ×

+ Add URI

Save the application. Copy your client id to a convenient place.

Click on the Assignments tab to assign this app to users. For PoC purposes, it is easiest to assign the app to the “everyone” group.



Now that you have assigned the app to the Everyone group, add the redirect uri to the list of trusted origins for this tenant:

Security->API->Trusted Origins->Add Origin

Add Origin

Name

Origin URL ?

Type

CORS Selecting 'CORS' enables the origin URL to access Okta APIs from Javascript.

Redirect Selecting 'Redirect' points users to the origin URL during logout.

This completes the setup for your Okta tenant.

Skip to the next step.

- b. For Okta Developer edition: Applications->Add Application->Web->Next

APPLICATION SETTINGS

Name

Base URIs
Optional

The domains where your application runs. Trusted Origins will be created for these URIs, and will be the only domains Okta accepts API calls from. [Docs](#)

Login redirect URIs

Okta will send OAuth authorization response to these URIs. Add your application's callback endpoint. [Docs](#)

Group assignments
Optional

Users can only sign in to apps that they are assigned to. Group assignments are easier to manage than individual users.

Grant type allowed

- Authorization Code
- Implicit
- Refresh Token

Okta can authorize your native app's requests with these OAuth 2.0 grant types. Limit the allowed grant types to minimize security risks [Docs](#)

Click "done" to finish the OIDC client setup.

You will now have your client_id. Copy your client id to a convenient place.

Set up GCE to accept JSON web tokens (JWT) from Okta

To add Okta to GCE API, you need to edit the OpenAPI configuration file for your API. In Google's sample application for node.js, for example, this file is:

nodejs-docs-samples/endpoints/getting-started/openapi-appengine.yaml

By default, there is an entry for "api_key" in the security section of the OpenAPI configuration file:

```
security:  
- api_key: []
```

Add Okta to this section to allow either an Okta access token or a Google API key for all API endpoints:

```
security:  
- api_key: []  
- okta_jwt: []
```

Add the definition for okta_jwt to the securityDefinitions section:

```
securityDefinitions:  
  okta_jwt:  
    authorizationUrl: ""  
    flow: "implicit"  
    type: "oauth2"  
    x-google-issuer: "https://partnerpoc.oktapreview.com"  
    x-google-jwks_uri:  
"https://partnerpoc.oktapreview.com/oauth2/v1/keys"  
    x-google-audiences: "0oac7c9f6goG9x3FY0h7"
```

As indicated, you need to update only three values:

x-google-issuer: this is your okta tenant name

x-google-jwks_uri: substitute your okta tenant name in this url

x-google-audiences: this is the client_id from the client you created in your Okta tenant

That's it. Save the OpenAPI configuration file and deploy a new version your API.

Getting an access token from Okta

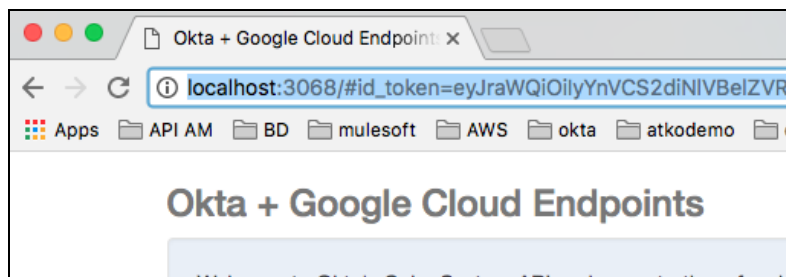
Now you can get an access token from Okta to test against your GC endpoint. First, make sure you have an active user in your Okta tenant, and that this user has been assigned to the OIDC client you created earlier. (If you assigned the OIDC app to the Everyone group you are fine.)

Use the following URL as a template, substituting your own values:

```
https://partnerpoc.oktapreview.com/oauth2/v1/authorize?response_type=id_token&client_id=0oac7c9f6goG9x3FY0h7&redirect_uri=http://localhost:3068&state=someState&nonce=someNonce&prompt=login&scope=openid
```

Note: in a production environment, you should provide valid values for *state* and *nonce*.

When you successfully authenticate, you will be redirected to the `redirect_uri` and an access token will be appended to the URL:



Copy the `id_token` from the URL.

Access the protected GC endpoint

Now that you have an access token, you can try to access your protected Google Cloud endpoint. Just append the access token to the url of your API endpoint and POST to the url. If you are using the example GCE application, post a message in the body and it will be echoed back:

The screenshot shows a REST client interface with the following details:

- Request Method:** POST
- URL:** `https://gcpandokta.appspot.com/echo?access_token=eyJraWQiOiIybnVCS2diNlVBelZVRzFaZmt4...`
- Params:** A table with the following entries:

Key	Value	Description
<input checked="" type="checkbox"/> access_token	eyJraWQiOiIybnVCS2diNlVBelZVRzFaZmt4MVVpaEM..	
New key	Value	Description
- Body:** The request body is a JSON object: `{ "message": "hello world" }`. The format is set to `JSON (application/json)`.
- Response:** The response status is 200. The response body is a JSON object: `{ "message": "hello world" }`.

Note: for production, Okta and Google recommend including the access token in the authorization header of the HTTP request.

That completes the process of setting up Okta as an access token provider for Google Cloud Endpoints.