Google Application Default Credentials

The Application Default Credentials provide a simple way to get authorization credentials for use in calling Google APIs.

They are best suited for cases when the call needs to have the same identity and authorization level for the application independent of the user. This is the recommended approach to authorize calls to Google Cloud APIs, particularly when you're building an application that is deployed to Google App Engine (https://cloud.google.com/appengine/docs/) or Google Compute Engine (https://cloud.google.com/compute/docs/) virtual machines.

When to use the Application Default Credentials

We recommend using Application Default Credentials in any of the following circumstances:

- Your code runs on Google App Engine or Google Compute Engine. The Application Default Credentials provide access to the built-in service accounts when deployed, but also provide a way of using alternative credentials when testing out the application prior to deployment.
- You wish to avoid embedding authentication information in application source code. It is generally a best practice to avoid including secrets related to authentication in source code, and it is also sometimes desirable to use different credentials in different contexts, such as testing and production. Credentials can be defined outside the application using an environment variable.
- You are accessing APIs with data associated with a cloud project or otherwise scoped to the whole application rather than personal user data. For calls involving user data, it is instead best to use an authorization flow where the end user gives explicit consent for access (see Using OAuth 2.0 to Access Google APIs (https://developers.google.com/identity/protocols/OAuth2)).

How the Application Default Credentials work

You can get Application Default Credentials by making a single client library call. The credentials returned are determined by the environment the code is running in. Conditions are checked in the following order:

1. The environment variable `GOOGLE_APPLICATION_CREDENTIALS` is checked. If this variable is specified it should point to a file that defines the credentials. The simplest way to get a credential for this purpose is to create a Service account key in the Google API Console:
    a. Go to the API Console Credentials page (https://console.developers.google.com/project/_/apis/credentials).
    b. From the project drop-down, select your project.
    c. On the Credentials page, select the **Create credentials** drop-down, then select **Service account key**.
    d. From the **Service account** drop-down, select an existing service account or create a new one.
    e. For **Key type**, select the **JSON** key option, then select **Create**. The file automatically downloads to your computer.
    f. Put the *.json file you just downloaded in a directory of your choosing. This directory must be private (you can't let anyone get access to this), but accessible to your web server code.
    g. Set the environment variable `GOOGLE_APPLICATION_CREDENTIALS` to the path of the JSON file downloaded.
2. If you have installed the Google Cloud SDK on your machine and have run the command `gcloud auth application-default login` (https://cloud.google.com/sdk/gcloud/reference/beta/auth/application-default/login), your identity can be used as a proxy to test code calling APIs from that machine.
3. If you are running in Google App Engine production, the built-in service account associated with the application will be used.
4. If you are running in Google Compute Engine production, the built-in service account associated with the virtual machine instance will be used.
5. If none of these conditions is true, an error will occur.

Calling the Application Default Credentials in application code

The default credentials are provided in Google API client libraries. Supported environments are Google Compute Engine VMs, and installed applications on Linux, Windows, and Mac OS.

Java

The default credentials are provided as of version 1.19 of the Google APIs Client Library for Java (https://developers.google.com/api-client-library/java/).

```
import com.google.api.client.googleapis.auth.oauth2.GoogleCredential;

...

GoogleCredential credential = GoogleCredential.getApplicationDefault();
```

It can be used to access an API service as follows:

```
Compute compute = new Compute.Builder
    (transport, jsonFactory, credential).build();
```

The credentials you retrieved might require you to specify the scopes you need explicitly. Check for this case, and inject the requested scopes if required.

```
Collection COMPUTE_SCOPES =
    Collections.singletonList(ComputeScopes.COMPUTE);
if (credential.createScopedRequired()) {
    credential = credential.createScoped(COMPUTE_SCOPES);
}
```

See also: Google APIs Client Library for Java documentation (https://developers.google.com/api-client-library/java/google-api-java-client/reference).

Python

The default credentials are provided as of version 1.3 of the Google APIs Client Library for Python (https://developers.google.com/api-client-library/python/).

```
from oauth2client.client import GoogleCredentials
credentials = GoogleCredentials.get_application_default()
```

This can be used by a service like this:

```
from googleapiclient.discovery import build
```

```
...

service = build('compute', 'v1', credentials=credentials)
```

The `build()` method takes care of injecting the proper scopes for the given service, although the method `create_scoped` can be used to do this explicitly.

See also: Google APIs Client Library for Python documentation (https://oauth2client.readthedocs.org/en/latest/source/oauth2client.client.html#oauth2client.client.GoogleCredentials).

Go

Default credentials are provided by the `golang.org/x/oauth2/google` package. To use them, add the following import:

```
import "golang.org/x/oauth2/google"
```

The credentials you retrieved might require you to specify the scopes you need explicitly. Check for this case, and inject the requested scopes if required.

```
import (
        "golang.org/x/net/context"
        "golang.org/x/oauth2/google"
        "google.golang.org/api/compute/v1"
)

// Use oauth2.NoContext if there isn't a good context to pass in.
ctx := context.TODO()

client, err := google.DefaultClient(ctx, compute.ComputeScope)
if err != nil {
  // Handle error.
}
computeService, err := compute.New(client)
if err != nil {
  // Handle error.
}
```

If you need a `oauth2.TokenSource`, use the `DefaultTokenSource` function:

```
ts, err := google.DefaultTokenSource(ctx, scope1, scope2, ...)
if err != nil {
  // Handle error.
}
httpClient := oauth2.NewClient(ctx, ts)
```

See also: **`golang.org/x/oauth2/google`** (https://godoc.org/golang.org/x/oauth2/google) package documentation.

.NET

The default credentials are provided by the Google APIs Auth Client Library for .Net (https://www.nuget.org/packages/Google.Apis.Auth/), version 1.9.3 or newer. To use them, import the OAuth 2.0 library and retrieve the credentials:

```
using Google.Apis.Auth.OAuth2;
...
GoogleCredential credential = await GoogleCredential.GetApplicationDefaultAsync(
```

Then, use the credentials to access an API service as follows:

```
var compute = new ComputeService(new BaseClientService.Initializer()
{
        HttpClientInitializer = credential
});
```

The credentials you retrieved might require you to specify the scopes you need explicitly. Check for this case, and inject the requested scopes if required.

```
if (credential.IsCreateScopedRequired)
{
        credential = credential.CreateScoped(new[] { ComputeService.Scope.CloudP
}
```

Node

Default credentials are provided by the Google APIs Client Library for Node.js (https://github.com/google/google-api-nodejs-client), versions 2.0.1 and newer. To use them, call **`getApplicationDefault`**:

```
google.auth.getApplicationDefault(function(err, authClient) {
  if (err) {
    return cb(err);
  }});
```

The credentials you retrieved might require you to specify the scopes you need explicitly. Check for this case, and inject the requested scopes if required.

```
if (authClient.createScopedRequired &&
    authClient.createScopedRequired()) {
  authClient = authClient.createScoped(
      ['https://www.googleapis.com/auth/devstorage.read_write']);
}
```

Then, use the credentials to access an API service as follows:

```
var storage = google.storage('v1');
storage.buckets.list({
  auth: authClient,
  project: projectId
}, cb);
```

Ruby

Default credentials are provided by the Google APIs Client Library for Ruby (https://github.com/google/google-auth-library-ruby), versions 0.1.0 and newer. To use them, import the **`googleauth`** gem, then pass the scopes you need to access to **`Google::Auth.get_application_default`**:

```
require 'googleauth'
scopes =  ['https://www.googleapis.com/auth/cloud-platform', 'https://www.go
authorization = Google::Auth.get_application_default(scopes)
```

Then, use the credentials to access an API service as follows:

```
require "google/apis/storage_v1"
storage = Google::Apis::StorageV1::StorageService.new
storage.authorization = authorization
```

PHP

Default credentials are provided by the Google APIs Client Library for PHP (https://github.com/google/google-api-php-client), versions 2.0.0 and newer. To use them, call `useApplicationDefaultCredentials`:

```
$client = new Google_Client();
$client->useApplicationDefaultCredentials();
```

Then, use the credentials to access an API service as follows:

```
$client->setScopes(['https://www.googleapis.com/auth/books']);
$service = new Google_Service_Books($client);
$results = $service->volumes->listVolumes('Henry David Thoreau');
```

Tool Support

Google Cloud SDK

As of version 98.0.0 of the Cloud SDK, running the command **`gcloud beta auth application-default login`** (https://cloud.google.com/sdk/gcloud/reference/beta/auth/application-default/login) will make your identity available to the Application Default Credentials as proxy when testing application code locally. An exception is that this does not happen when running the command from a gcloud workspace directory.

This can be a fast and convenient way to verify code locally. One caveat is that your personal identity usually has a lot of permissions so you may find situations where an API call works locally but not when deployed. It also only works for permission scopes for Cloud APIs. If these issues become a problem, consider using the environment variable environment variable (#howtheywork) with a downloaded service account key.

Google App Engine SDK

As of version 1.9.18 of the Google App Engine SDK, the Application Default Credentials can also be used when running the local development application server. Note, this is only supported in the server provided by the command **`gcloud preview app run`**

(https://cloud.google.com/sdk/gcloud/reference/preview/app/run) and is not supported in the older language-specific development servers.

Troubleshooting

The Application Default Credentials allows you to use different identities depending on the context. This sometimes means adjustments are needed to ensure all the identities being used have the right capabilities.

Permissions

Code using Application Default Credentials can run as user or service account identities, including the built-in service account identities. If using more than one identity, they must all have permissions for the calls being made. To configure permissions, go to the Google API Console Permissions page (https://console.developers.google.com/project/_/permissions/serviceaccounts).

Scopes

OAuth2 permission scopes are a means of defining which API calls can be used in a given context. They take the form of URIs that name a set of capabilities for a given API, e.g. "https://www.googleapis.com/auth/compute.readonly". The different identity types handle scopes differently.

For downloaded service account keys and the Google App Engine built-in service account, scopes must be specified in code. API wrappers may do this by default but if not you will get an error when using the credentials. See Calling the Application Default Credentials in application code (#calling) for information on how to inject scopes for your language.

For the Google Compute Engine virtual machine service accounts, the supported scopes must be specified at the time the virtual machines are created. If using the Cloud SDK do this with the **--scopes** parameter for the command `gcloud compute instances create` (https://cloud.google.com/sdk/gcloud/reference/compute/instances/create).

If using the Google Cloud SDK `gcloud beta auth application-default login` (https://cloud.google.com/sdk/gcloud/reference/beta/auth/application-default/login) command to provide your own identity for use locally, the set of scopes is currently fixed, although it

includes all Google Cloud API scopes. If you need a scope outside this set the recommendation is to use a downloaded service account key.

Last updated October 31, 2016.