OAuth integration

Deleted: Integration

Carina supports OAuth v2 for integration with external applications. OAuth is an authentication protocol that allows users to grant limited access to their account without sharing their password. With Carina OAuth, you can authenticate users, download their cluster credentials, and create clusters on their behalf.

Deleted: a

Deleted: or

Deleted: a

- OAuth dance
- Register your application
- Configure your application
- Interact with Carina

OAuth dance

Comment [KH1]: First-level headings should be at the H3 level (###). Revise all of the heading levels accordingly.

Deleted: Dance

The flow that a user experiences when authorizing an application is sometimes called the "OAuth dance" because of the back-and-forth communication between the external application and OAuth provider. The following process describes that dance with Carina.

Deleted: , due to

1. The external application requests permission to access the user's Carina account. Usually this request happens when a user clicks a login or authorize button on the application's website.

Deleted: will be

Deleted:

Deleted: their

2. The user is redirected to the Carina OAuth website. If the user is not logged in, the user is prompted for Rackspace credentials. These credentials are used to authenticate the user against Carina and are not shared with the external application.

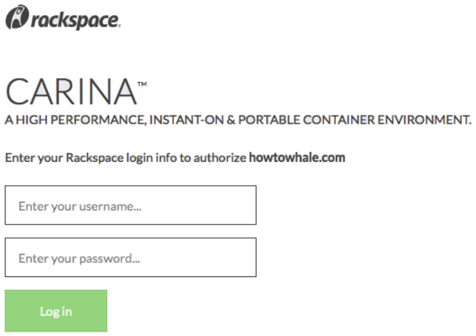
Deleted: where, i

Deleted: they

Deleted: are

Deleted: their

Deleted: ,



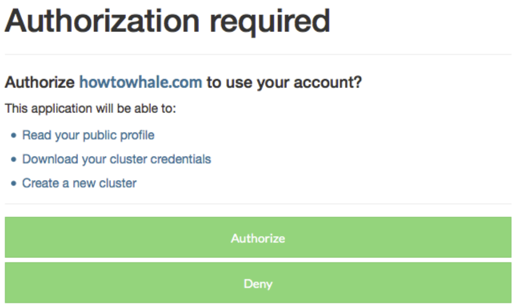
CARINA™

A HIGH PERFORMANCE, INSTANT-ON & PORTABLE CONTAINER ENVIRONMENT.

Enter your Rackspace login info to authorize **howtowhale.com**

3. The user is presented with an authorization form that lists the permissions requested by the application.

Deleted: , listing



Authorization required

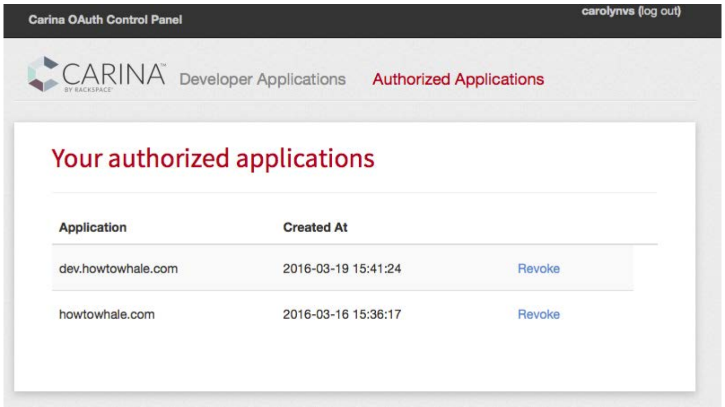
Authorize howtowhale.com to use your account?

This application will be able to:

- Read your public profile
- Download your cluster credentials
- Create a new cluster

4. After the application is authorized, the user is redirected back to the application.

Authorized applications can act on behalf of the user until that authorization is revoked. You can view a list of authorized applications on the Carina OAuth website and revoke an application's access.



Your authorized applications

Application	Created At	
dev.howtowhale.com	2016-03-19 15:41:24	Revoke
howtowhale.com	2016-03-16 15:36:17	Revoke

Register your application

Before your application can integrate with Carina, you must first register the application.

Deleted: you

Deleted: your

1. Log in to the Carina OAuth website.

2. On the Developer Applications page, click New Application.

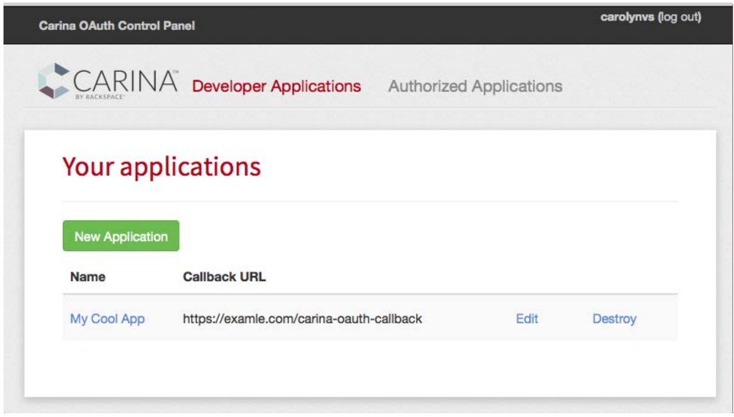
Comment [KH2]: When I performed these steps, the sequence was somewhat different:

1. Go to the Carina OAuth website.

2. Click the **Developer Applications** link and log in with your Carina credentials.

3. On the Developer Applications page, click …

If this is actually the sequence (for users who are not already logged in), then revise the steps as shown.



Your applications

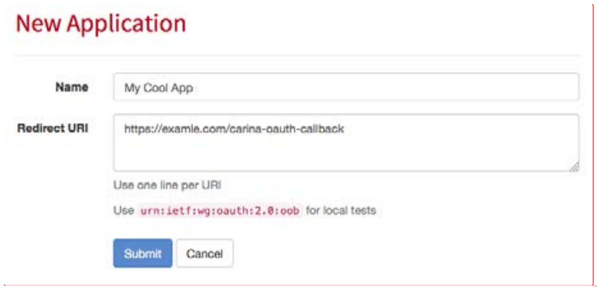
Name	Callback URL		
My Cool App	https://examle.com/carina-oauth-callback	Edit	Destroy

Comment [KH3]: I wonder if it would be at all confusing that the URL that you are registering in the next step is the same one that is shown as already created here. If a user is presumably doing this task for the first time, perhaps this screenshot should not show any existing registrations.

3. Complete the registration form. The Redirect URI specifies the URL to which users are directed after they authorize your application. It must be a secure web page, starting with `https`.

Deleted: the user is

Deleted: webpage



New Application

Name: My Cool App

Redirect URI: https://examle.com/carina-oauth-callback

Use one line per URI

Use `urn:ietf:wg:oauth:2.0:oob` for local tests

Comment [KH4]: The URL includes "examle." Just to be nitpicky, consider changing it to "example." ☺ Oh, if you change it here, you'll have to redo the next screenshot as well.

On the confirmation page, the application ID and secret are displayed. Do not share the secret publicly, and do not check it in to source control.

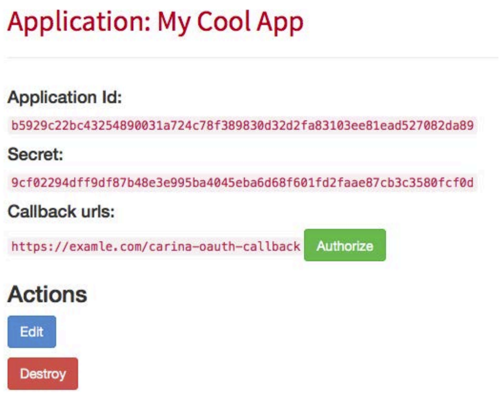
Deleted: The

Formatted: No bullets or numbering

Deleted: your Application Id and Secret

Deleted: must not be shared

Deleted: should not be checked into



Application: My Cool App

Application Id:

`b5929c22bc43254890031a724c78f389830d32d2fa83103ee81ead527082da89`

Secret:

`9cf02294dff9df87b48e3e995ba4045eba6d68f601fd2faae87cb3c3580fcf0d`

Callback urls:

`https://examle.com/carina-oauth-callback`

Actions

Configure your application

Select an OAuth2 library for the programming language used by your application. Depending on the library, some or all of the following configuration is handled by the library. The following guidelines are generic; see the documentation for your particular library for additional details.

Deleted: ,

1. In your application, configure the library to use the application ID, secret, and callback URL assigned to your registered application in the Carina OAuth Control Panel.

Deleted: your

Deleted: Application Id, Secret and Callback URL

Deleted: which

2. Add a link to your application that requests access to the user's Carina account. Replace `<applicationId>` and `<redirectURI>` with the values from your application registration.

```
https://oauth.getcarina.com/oauth/authorize?client_id=<applicationId>&redirect_uri=<redirectURI>&response_type=code
```

The `scope` query string parameter is optional and should be a space separated list of the

Comment [KH5]: So, it looks like the *callback URL* and the *redirect URI* are the same thing? If so, that seems a little confusing. After the user registers the application, use the label that is shown in the registration details. If that is *callback URL*, then use `<callbackURL>` for this placeholder text, instead of `<redirectURI>`. If that is *redirect URL*, then in step 1, replace *callback URL* with *redirect URI*. Be consistent throughout the article, reflecting as closely as possibly what the user sees in the control panel.

Deleted: ,

requested scopes.

```
&scope=identity+cluster_credentials+create_cluster
```

3. Add a web page to your application that handles requests to your Redirect URI.

Comment [KH6]: redirect URI *or* callback URL ?

Deleted: webpage

Formatted: Indent: Left: 0.25", No bullets or numbering

After the user authorizes your application, the user is redirected to this page with the authorization `code` included in the query string.

Deleted: web

```
https://example.com/carina-oauth-callback?code=abc123
```

Comment [KH7]: If you don't change the earlier screenshots that show "examle," then change this to "examle."

The callback web page should use the authorization code to request a token, and then store the returned tokens. The `access_token` value has an expiration date, after which the `refresh_token` value can be used to request a new token.

Formatted: Indent: Left: 0.25", No bullets or numbering

Deleted: webpage

Example request

Deleted: Request

```
POST https://oauth.getcarina.com/oauth/token
```

```
{
  "client_id": "<applicationId>",
  "client_secret": "<secret>",
  "redirect_uri": "<redirectURI>",
  "code": "<authorizationCode>",
  "grant_type": "authorization_code"
}
```

Example response

Deleted: Response

```
{
  "access_token": "09ba487fc3df...",
  "token_type": "bearer",
  "expires_in": 7200,
  "refresh_token": "8c488ab5f75d61..."
}
```

Interact with Carina

The following OAuth scopes are available, granting your application varying levels of access to a user's account.

- Identity
- Cluster credentials
- Create cluster

All requests must include an authorization header, `Authorization: bearer <access_token>`, replacing `<access_token>` with the OAuth token returned from the `/oauth/token` endpoint.

Deleted: ,

Identity scope

Deleted: Scope

The `identity` scope enables an application to read a user's profile. Because the amount of information in a user’s profile is limited, this scope is mainly useful for delegating your application's authentication to Carina.

Deleted: Due to the limited

Deleted: users'

Example request

Deleted: Request

```
GET https://oauth.getcarina.com/me
```

Example response

Deleted: Response

```
{
  "id": 1,
  "username": "carolynvs",
  "email": "carolyn.vanslyck@rackspace.com",
  "full_name": "carolynvs"
}
```

Cluster credentials scope

Deleted: Credentials

Deleted: Scope

The `cluster_credentials` scope enables an application to download the credentials zip file for a user’s Carina cluster. Replace `<clusterName>` with name of the cluster.

Deleted: users's

Example request

Deleted: R

```
GET https://oauth.getcarina.com/clusters/<clusterName>
```

Example responses

Deleted: R

The body of the response is the user's cluster credentials zip file, provided as an `application/zip` encoded binary attachment.

If the cluster is not yet active, a `404 NOT FOUND` response is returned. After a new cluster is created, it can take 2-3 minutes for the cluster to become active. Poll the cluster credentials endpoint at a reasonable interval, such as every 30 seconds, until the cluster is active.

Deleted: creating

```
{
  "message": "The cluster is not yet active. Retry this request later."
}
```

Create cluster scope

Deleted: Cluster

Deleted: Scope

The `create_cluster` scope enables an application to create a cluster on the user’s Carina account. If the cluster already exists, the request succeeds and the cluster information is returned. Replace `<clusterName>` with the name of the cluster to create.

Deleted: users'

Comment [KH8]: But I thought the point was to create a cluster? If the cluster already exists, how can the request to create it succeed? Is this wording correct?

Example request

Deleted: Request

```
PUT https://oauth.getcarina.com/clusters/<clusterName>
```

Example response

Deleted: Response

```
{
  "cluster_name": "<clusterName>",
  "status": "building",
  "autoscale": false,
  "flavor": "container1-4G",
  "nodes": 1
}
```

Resources

Comment [KH9]: This heading should remain tagged as an H3 (###).

- OAuth2 libraries
- OAuth2 authorization code grant workflow