

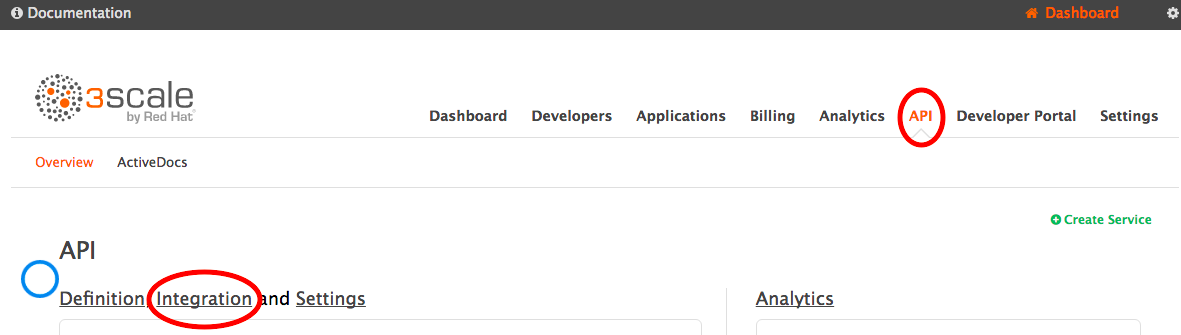
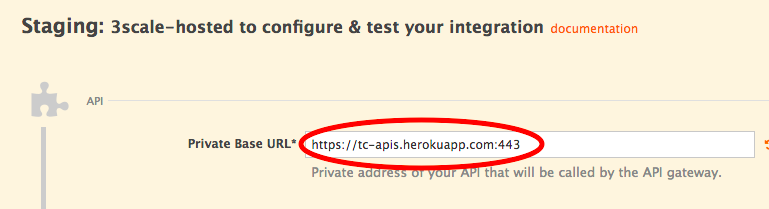
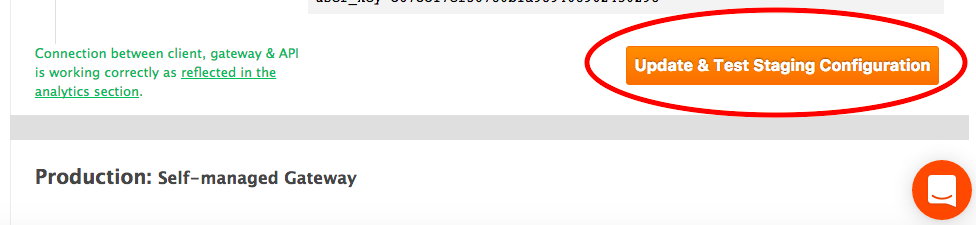
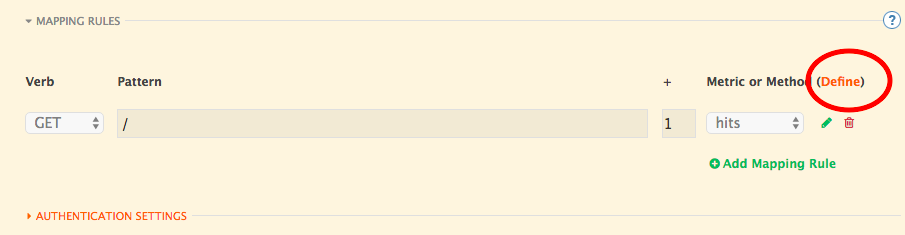
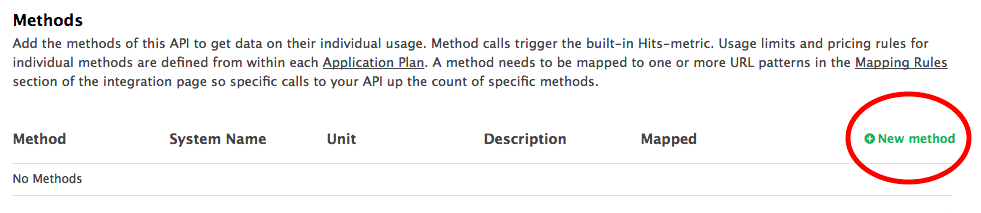
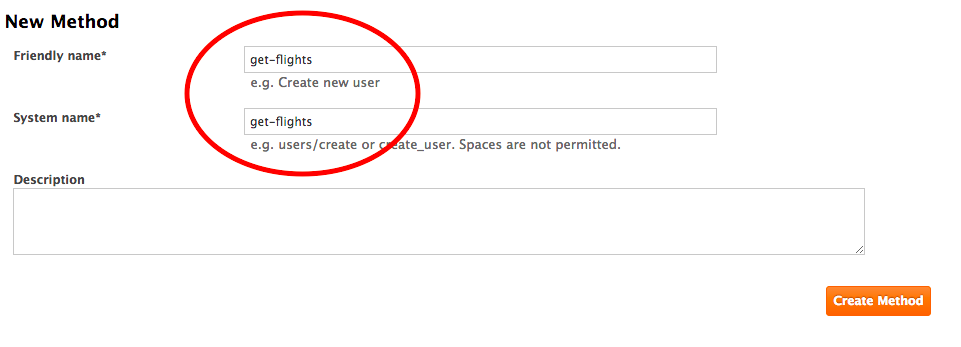
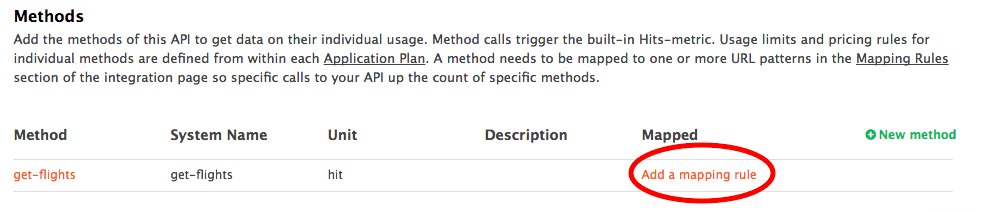
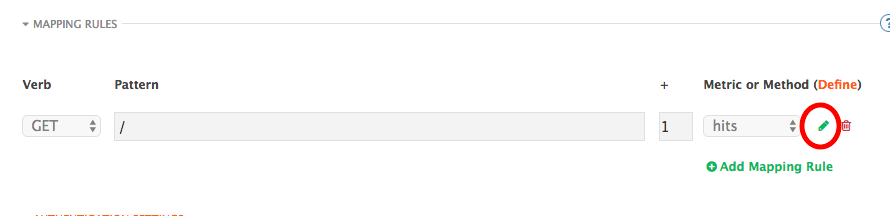
# Introduction

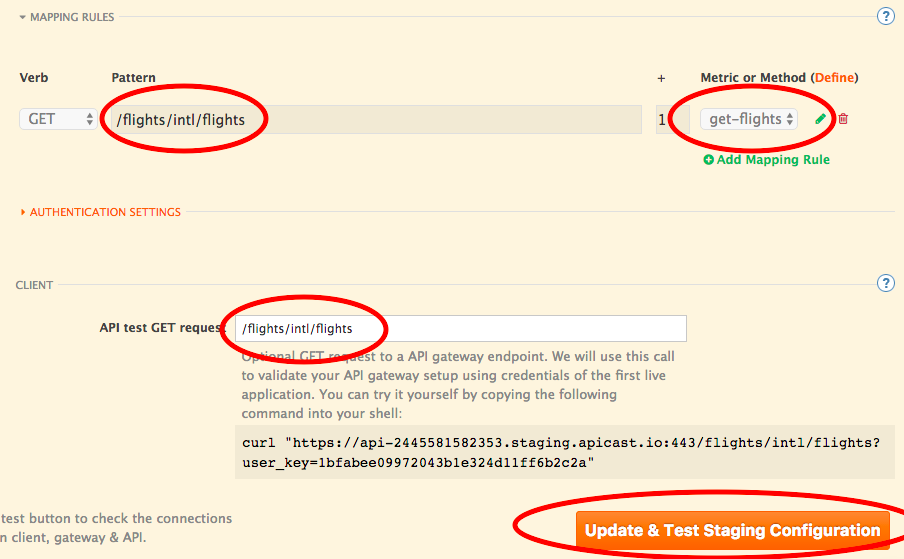
This workshop introduces you to some basic configuration of 3scale briefly touching on the main areas of functionality. We will loosely base it on a very basic POC we would normally roll out.

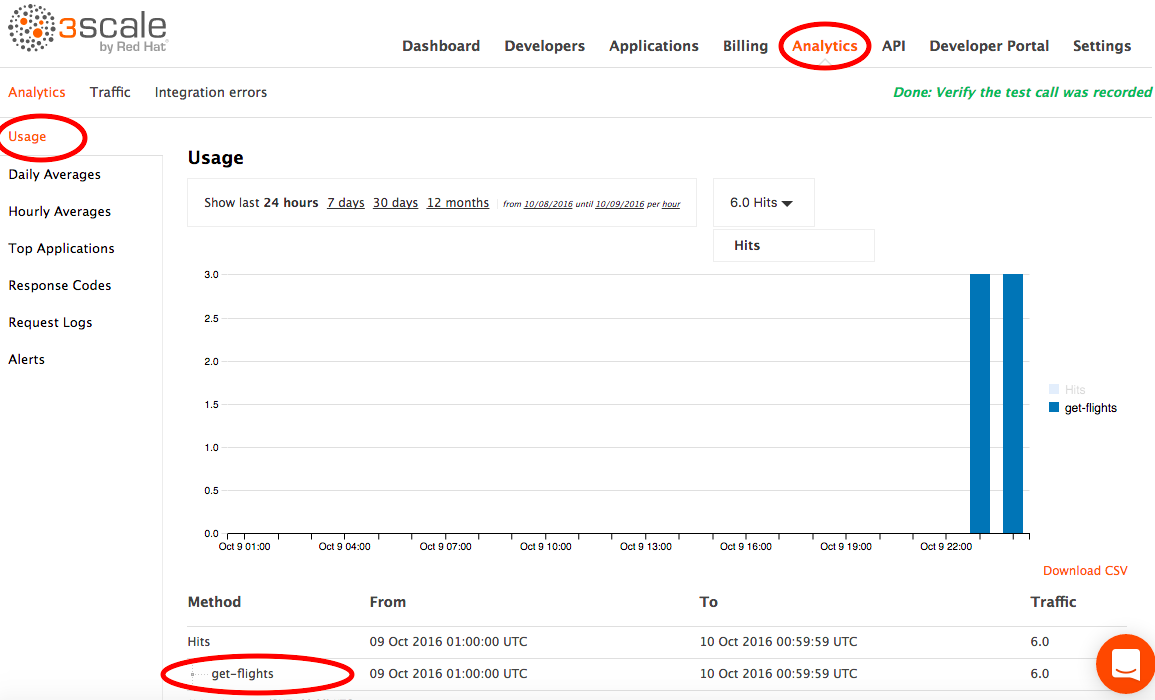
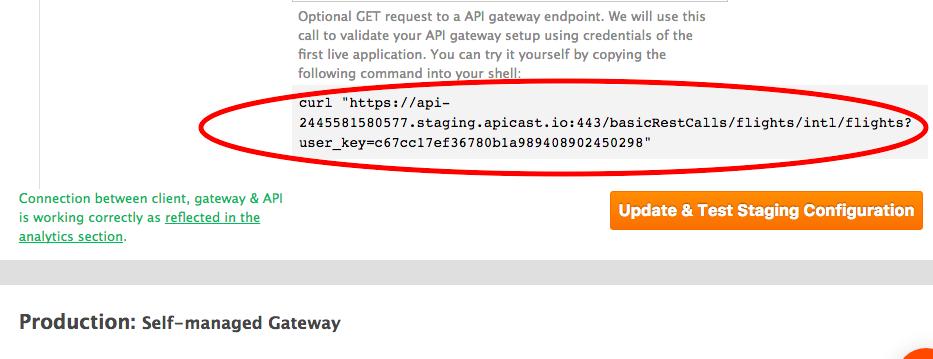
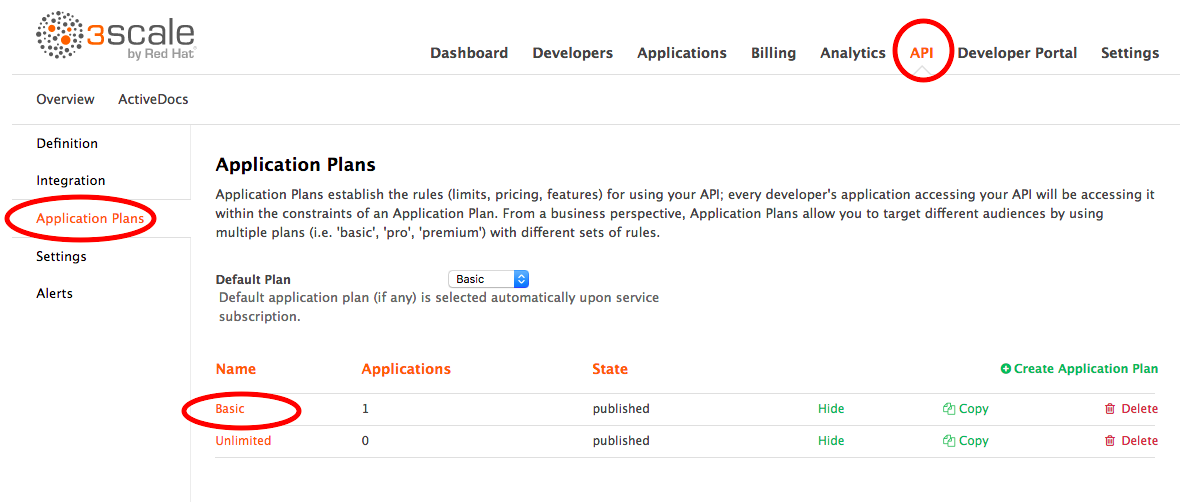
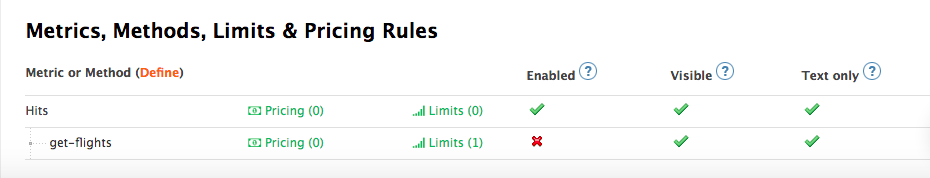
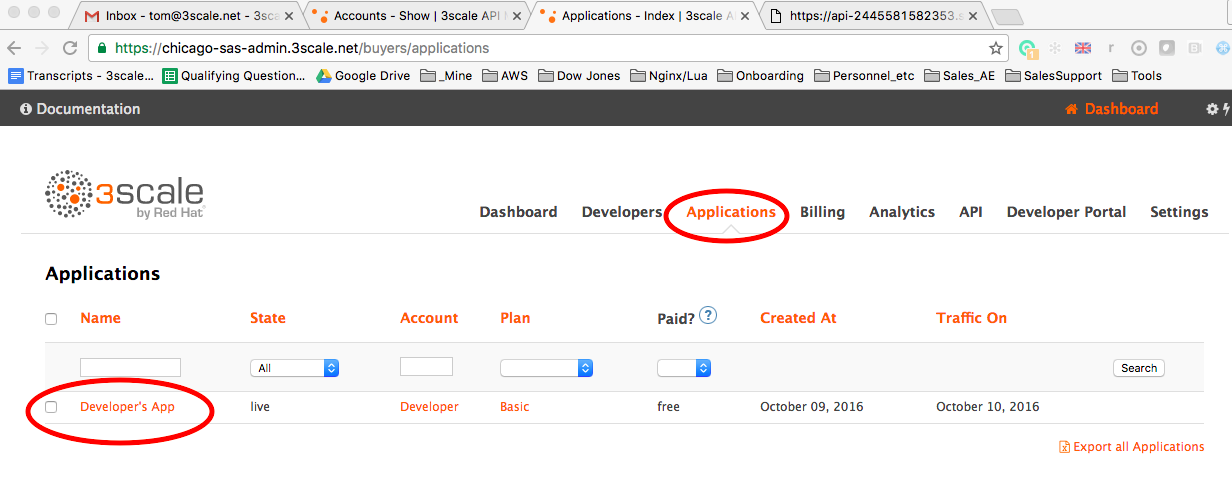
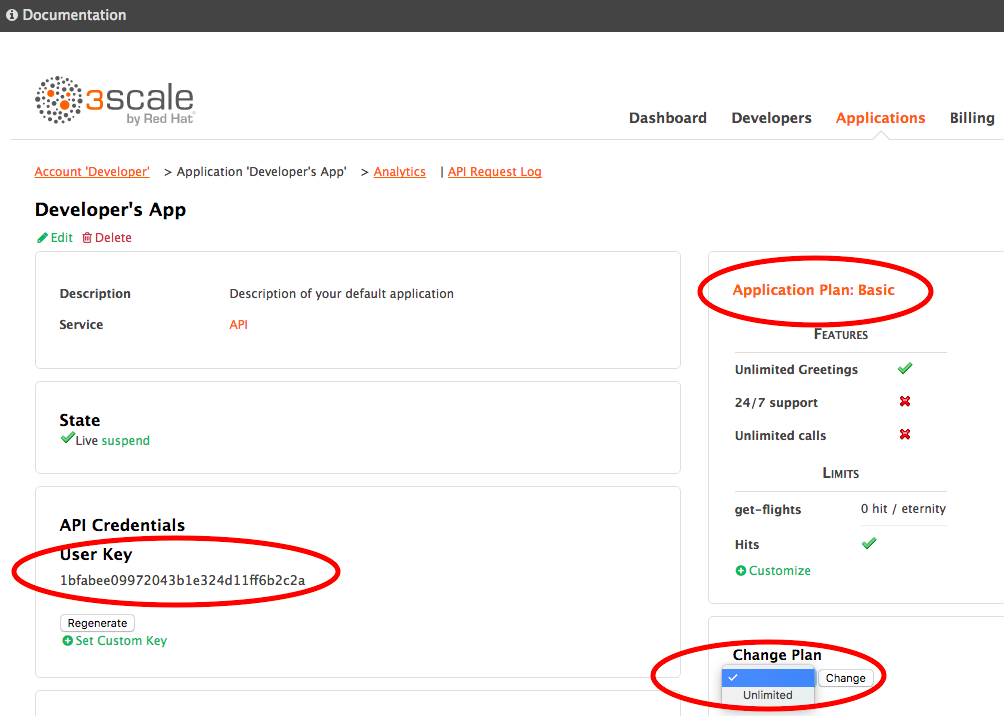
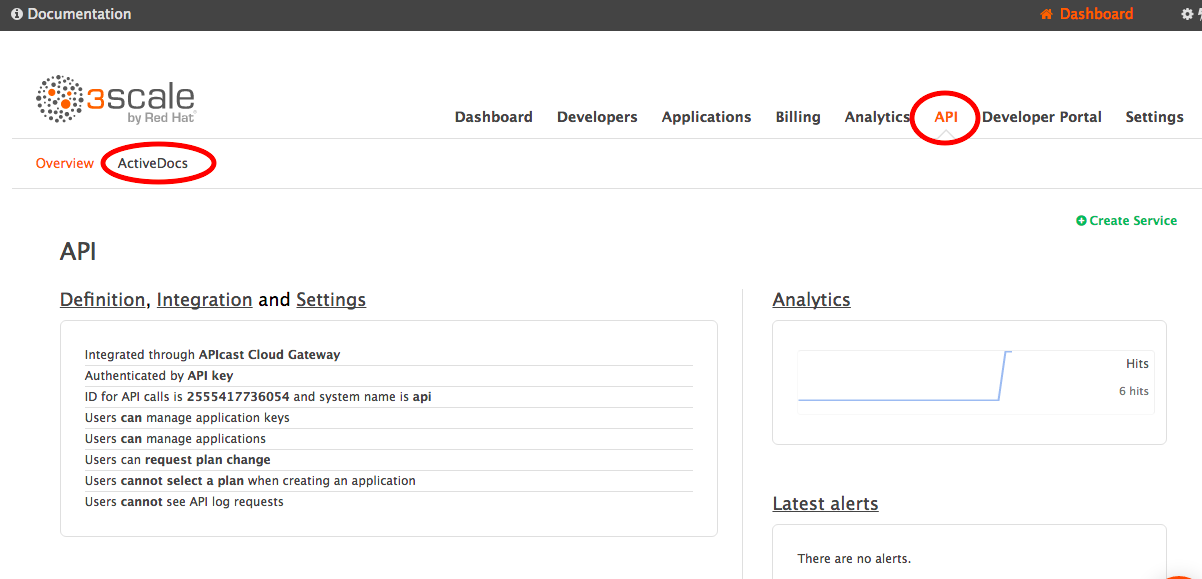
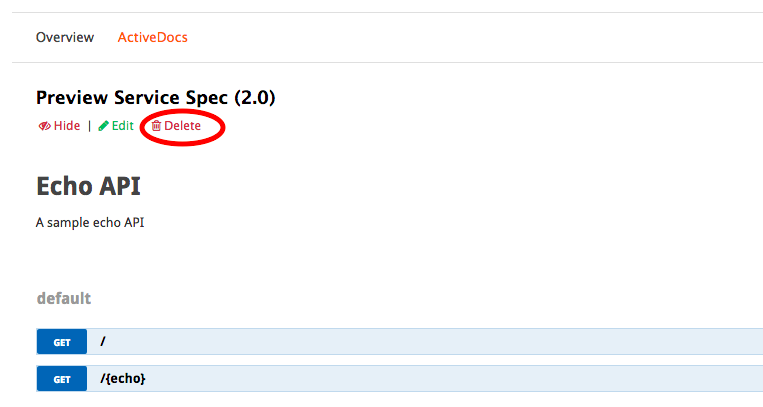
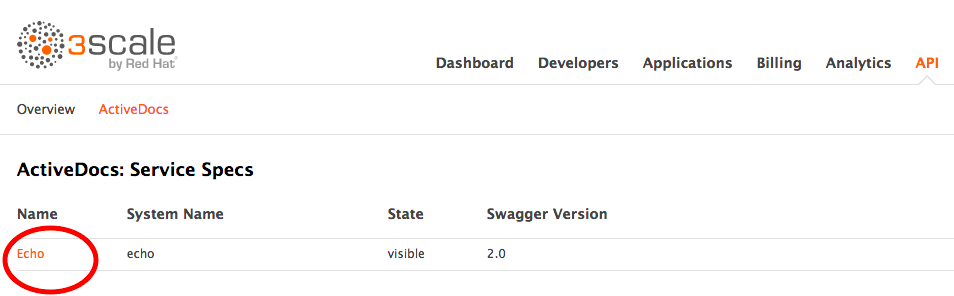
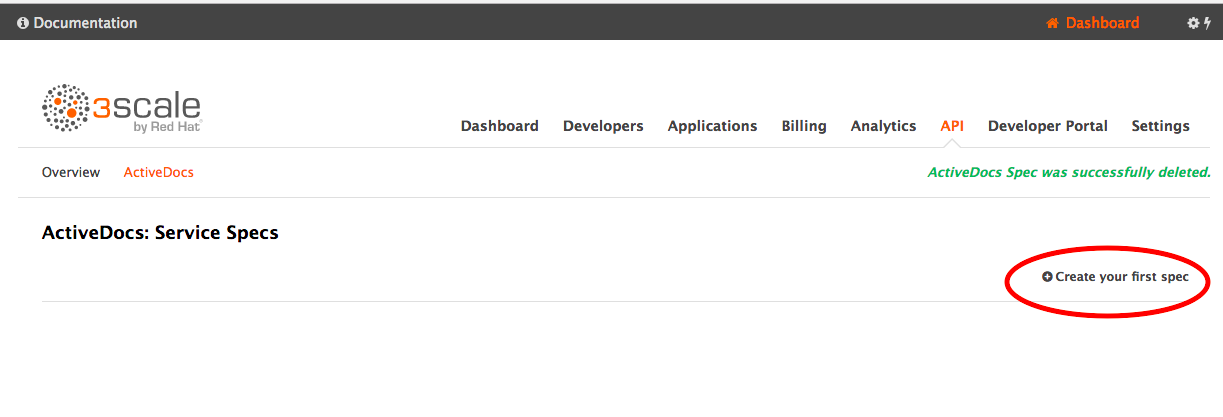
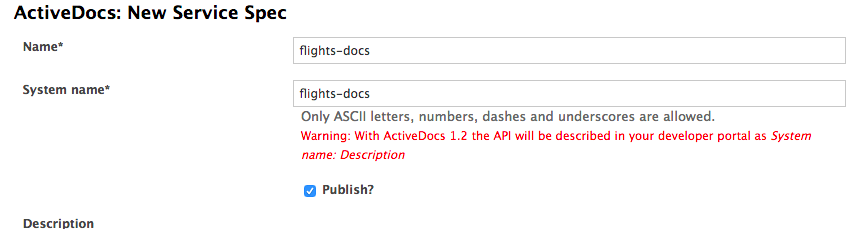
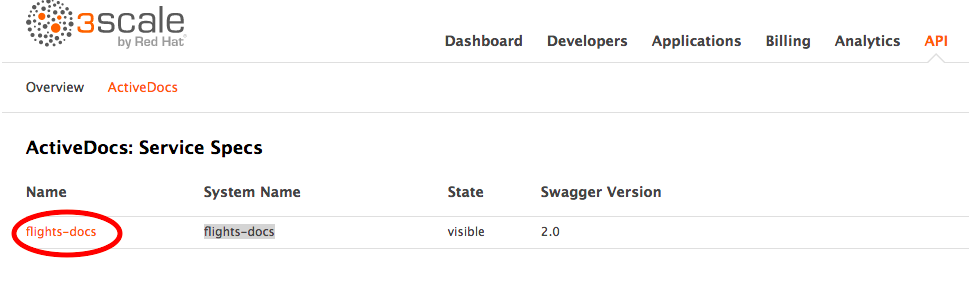
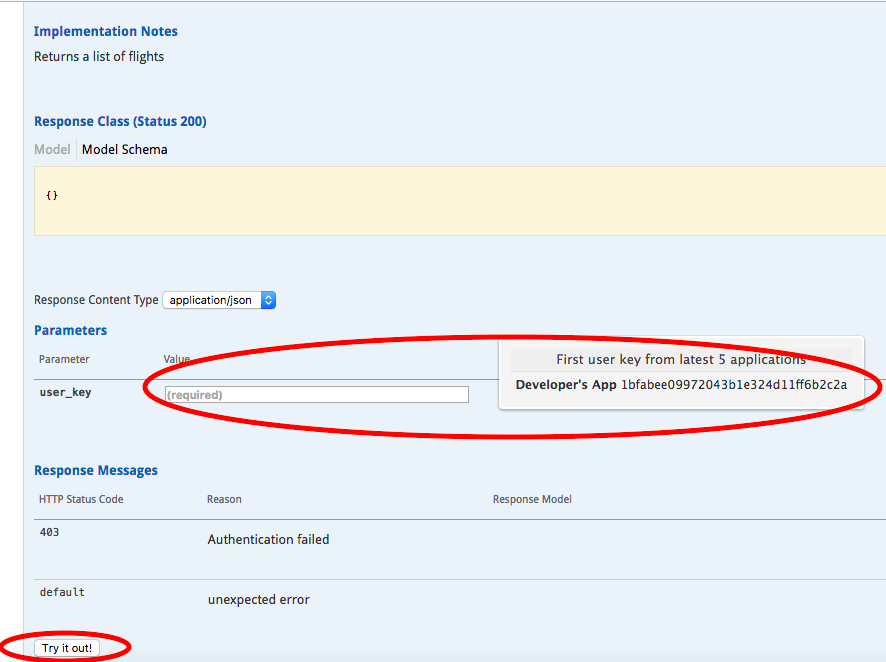
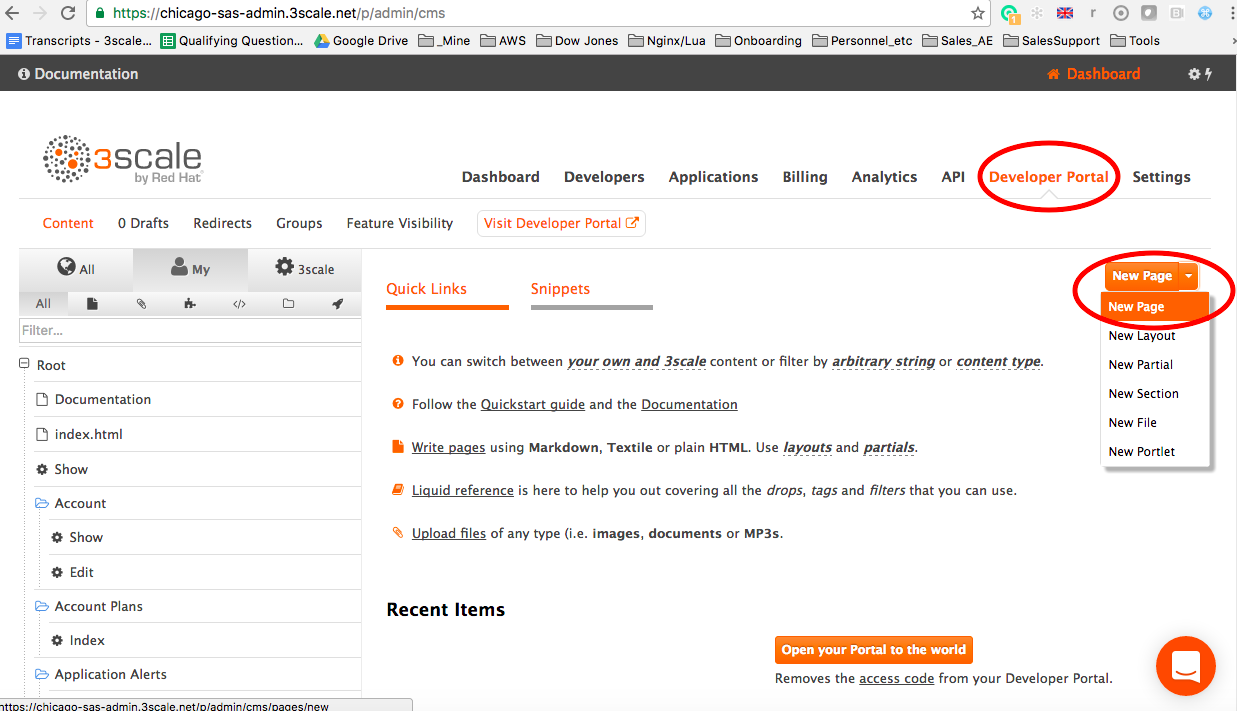
# Goals

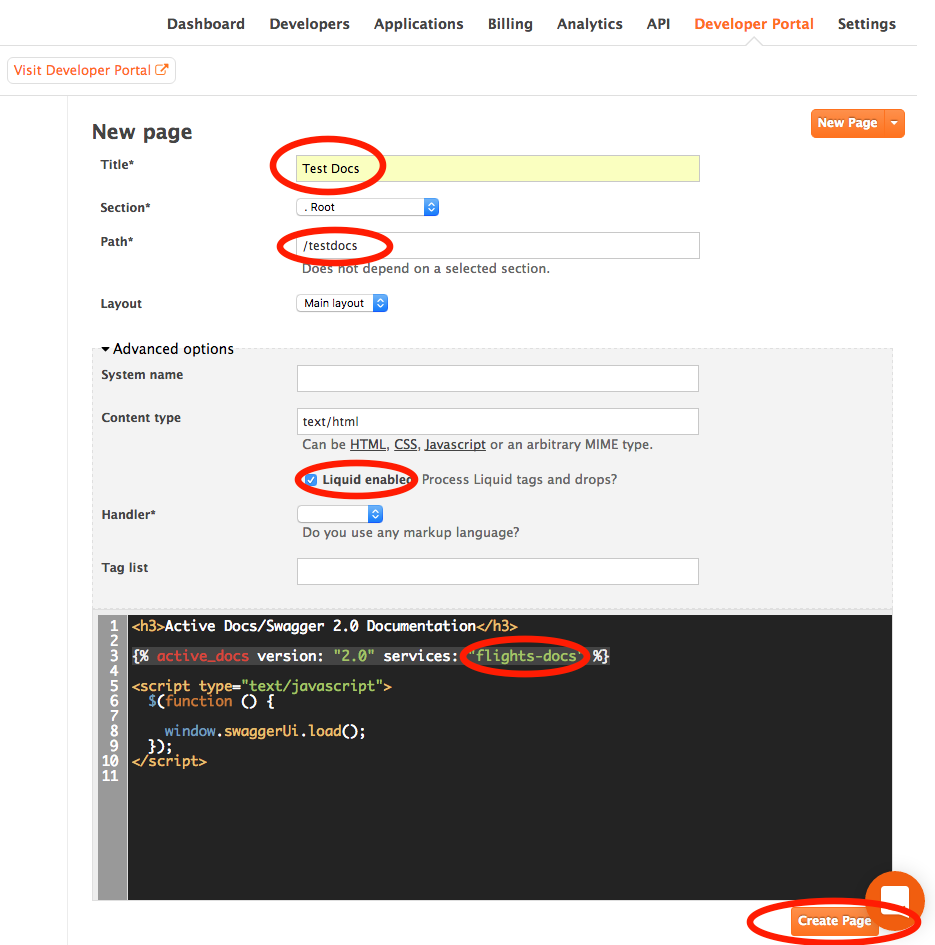
1. Integrate this simple ***unmanaged*** API Endpoint into 3scale   
   <https://tc-apis.herokuapp.com/flights/intl/flights>   
   (\*\*Note, this is a test API on Heroku. The first time it is hit every hour is slow)
2. Demonstrate the main 3scale API Management features typically required for a *basic POC* – including
   1. Access Control
   2. Policy Enforcement
   3. Analytics
   4. Swagger & Basic Developer Portal Configuration
3. Gain familiarity with the 2 Web Portals 3scale exposes
   1. Admin Portal. Used by: The API Provider – or organization ***exposing*** the API, e.g. United Airlines.
   2. Developer Portal. Used by: API consumers, e.g. Sites and mobile Apps ***using*** the APIs – e.g. Expedia, CheapOAir etc.

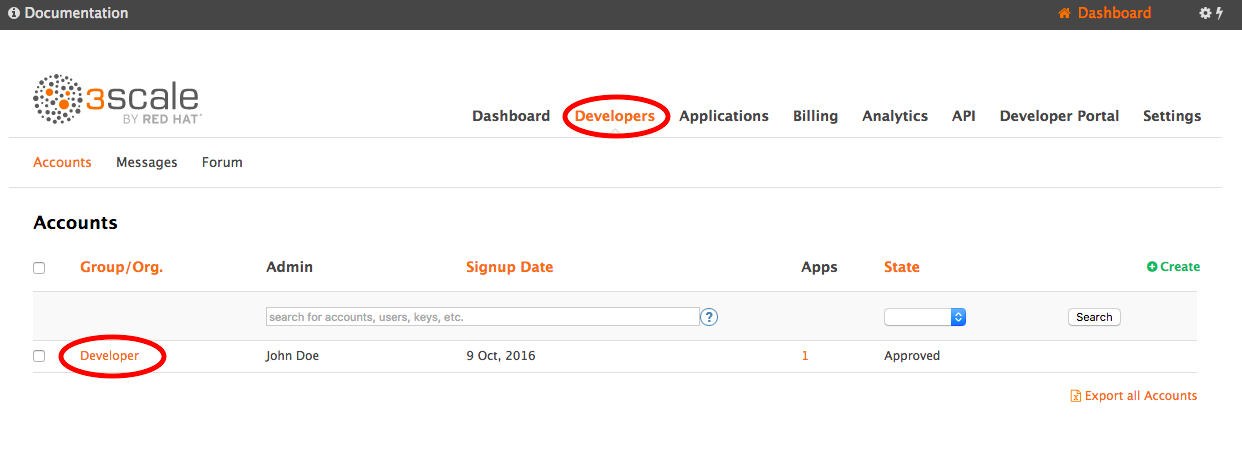
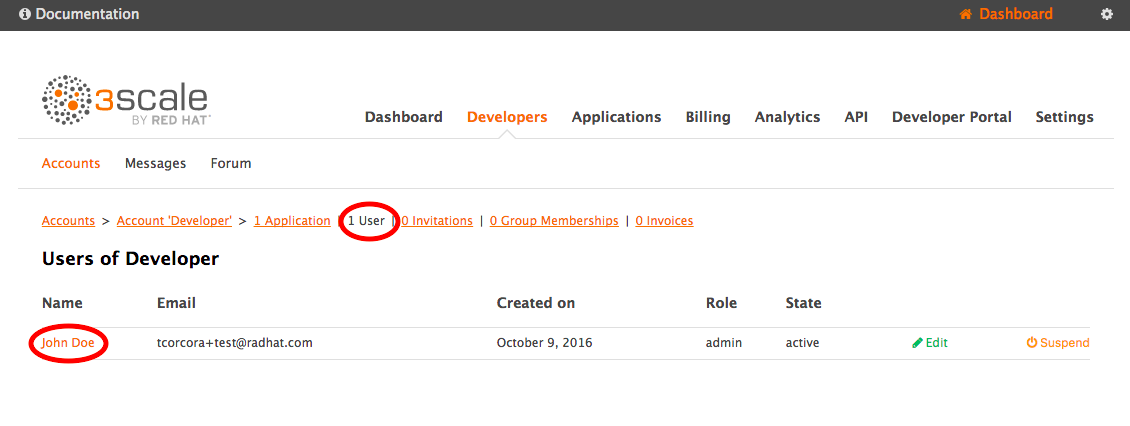
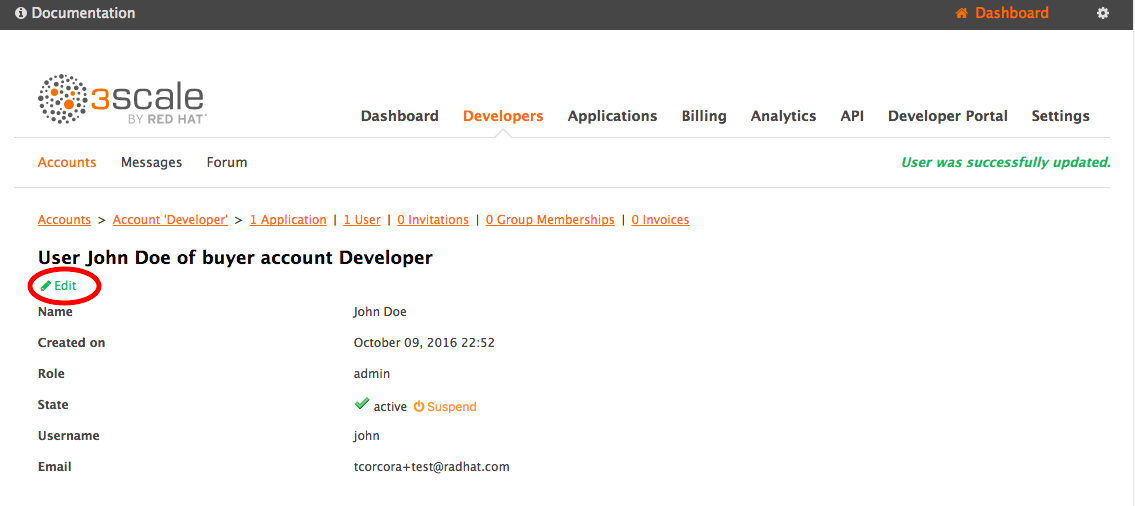
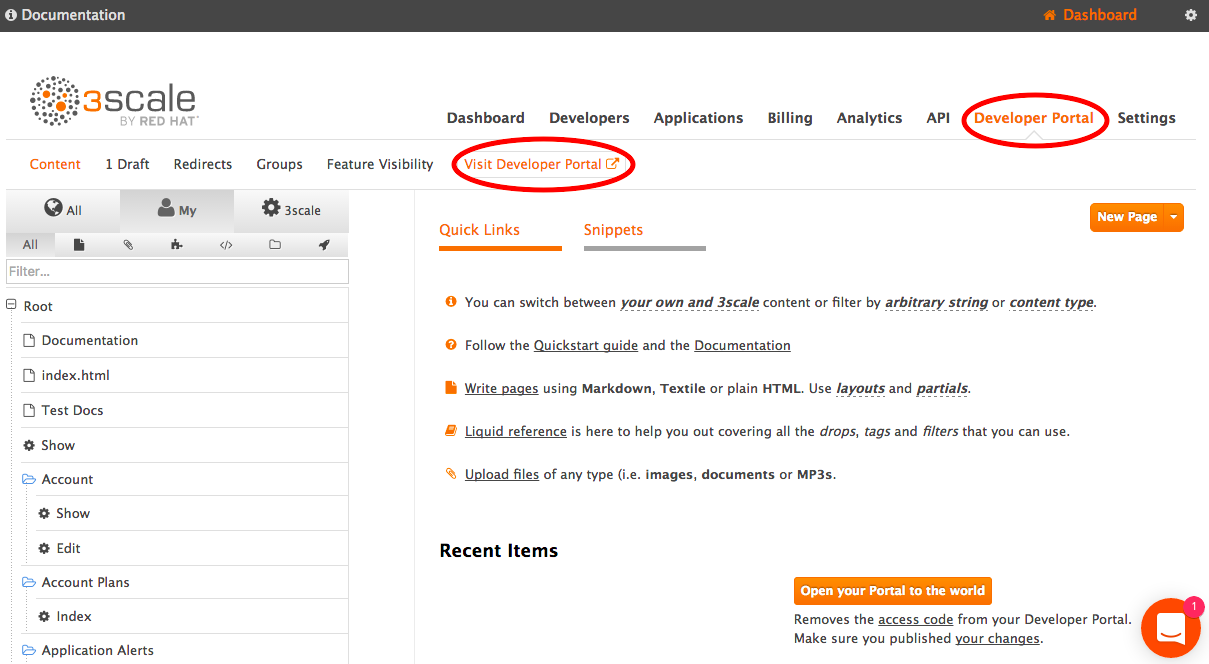
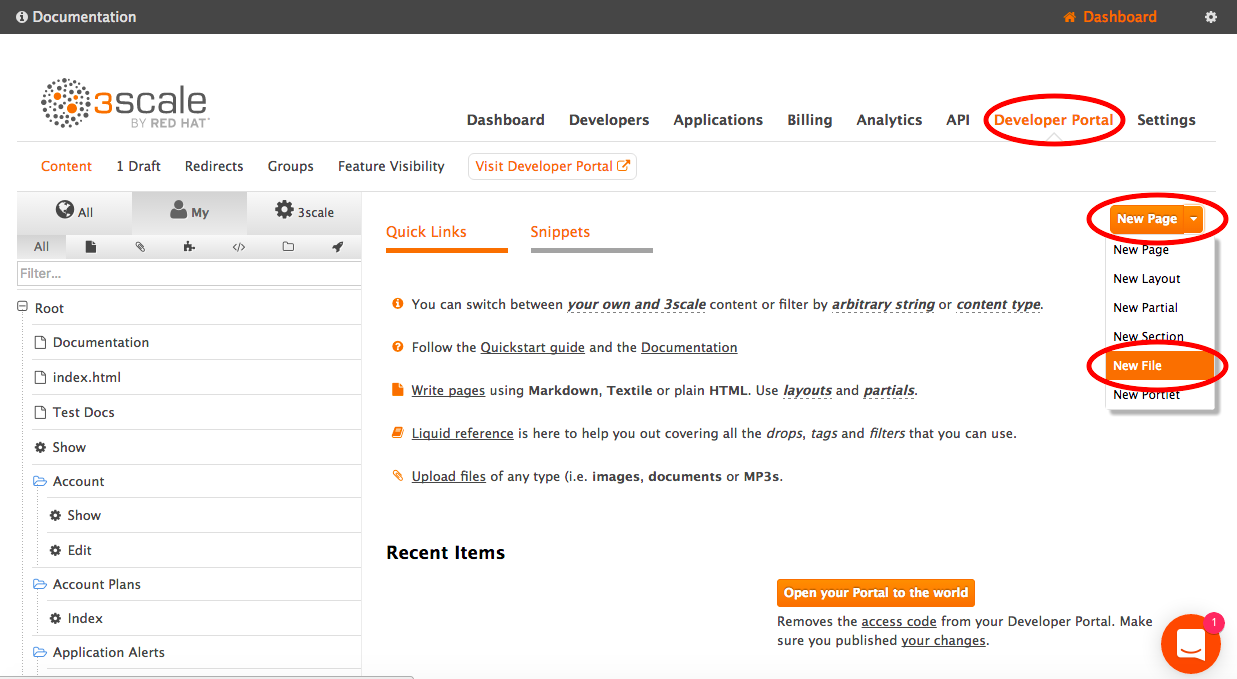
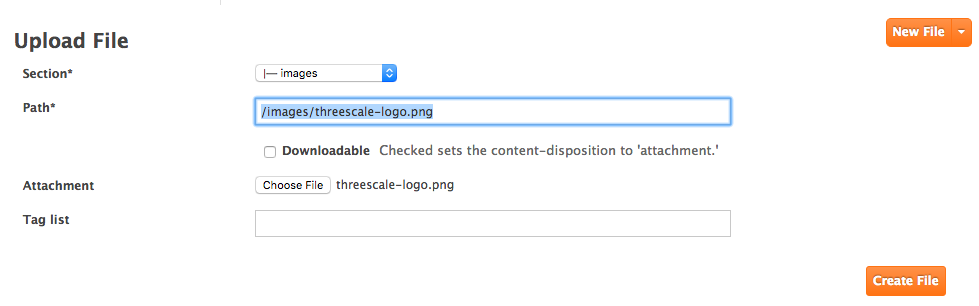
# Steps

1. Signup to 3scale
   * <https://www.3scale.net/signup/>
   * Activate Emailed Link.
2. Login & Integrate an API endpoint.
   * Dismiss Greeting Message
   * Navigate to: API > Integration.   
     
   * This is where you configure your gateway which will communicate with the 3scale Platform. Make the following entries:
     1. Private Base URL\*: <https://tc-apis.herokuapp.com:443>  
          
          
        After entering this, click *Update & Test Staging Configuration* towards the bottom of the page (you are about to leave this page momentarily so this is necessary to save your Private Base URL)  
        
     2. Mapping Rules. This is where you map URL patterns of endpoints to ***Methods*** and ***Metrics*** (enabling you to track usage and control access)
        + Define a Method (logical representation of an Endpoint) by clicking *Define*.  
          
        + Create a *New Method*  
          
        + Name it something like *get-flights* (System and Friendly)  
          
        + Add a Mapping Rule to it back on Integration screen.  
          
        + Replace the default Mapping to *hits* by editing it.  
          

* + - * Replace the Pattern with the URL path to our endpoint (/flights/intl/flights) and select our new method *get-flights.* Use the same path (/flights/intl/flights) in *API test GET request* then click *Update and Test Staging Configuration*.
      * If all goes well, status will turn green – indicating your configuration has been deployed to the 3scale Nginx on AWS. You can now test your managed API endpoint using the curl in the screenshot above – or just pasting the URL into a browser.   
        (\*\*Note, this is a test API on Heroku. The first time it is hit every hour is slow)

1. Access Control and Analytics
   * Keep the Integration tab open and open another tab on Analytics -> Usage. Notice your method, *get*-*flights*
   * Test Authentication.
     1. In a terminal (or browser), hit the URL in the curl statement on the Integration screen you worked on (above) - something like:   
          
        It should succeed.
     2. Enter an incorrect user\_key and retry, it should fail.
   * Switch to your Analytics tab and refresh. The count for *get*-*flights* should have incremented
2. Policy Enforcement. This illustrates how you can give different access rights to different classes of API Consumer.
   * Go to API -> Application Plans -> click the Basic Plan which opens it.   
     
   * Disable method get-*flights* by clicking its *Enabled* check box
   * Go back to your terminal window and run the curl twice\*
   * Access will be denied  
     \*requires 2 as policy information in the cache will show success from the previous call and allow the first through. The 2nd will use freshly populated data and block access.
   * Now we’re going to change the access rights this API consumer has. Open Applications -> then click on the Developer’s App.   
     
   * Next we further illustrate Access Policy enforcement  
     This *application* represents our API Consumer. You can see the User Key on the left – the one we’ve been attaching to our 3scale managed API calls. The access rights this consumer gets are defined by the Plan this Application subscribes to: ***Basic*** on the top right. The Basic Plan has disabled access to *get-flights* – hence we’re blocked as you’ve seen.   
     Now we’re going to change the level of access this Developer App has. Change the Plan to *Unlimited* – which does have access to *get-flights*  
     
   * Go back to terminal and hit the API again – this time it succeeds because the application subscribes to a plan (i.e., Unlimited) that allows access to *get*-*flights*
3. Swagger & Basic Developer Portal Configuration. 3scale uses the popular API Documentation and testing tool Swagger. We’re now going to configure it then display it in the Developer Portal
   * Swagger.
     1. Open Active Docs menu item on the top left  
        
     2. Open Echo link and the left and Delete it.  
        
     3. Create new Spec.  
           
        + Name it *flights-docs* (both Name and System Name) and **click Publish**.   
          
        + Paste in the JSON in Appendix 1 – Swagger below into *API JSON Spec\**
        + Replace the string ***<yours>*** with the host you've been using in your curls, e.g. api-2445581074662.staging.apicast.io
        + Create Service
        + Open it by clicking *flights-docs*
        + click in the user\_key field (select pop up) and Try it out! (\*\*Note, this is a test API on Heroku. The first time it is hit every hour is slow)  
          
   * Developer Portal. Now we’re going to configure a page and a logo that will be ***viewable by an API consumer*** – in our example could be someone like Expedia or CheapOAir  
     2 very common items needed for POCs: 1) Swagger in Portal, 2) Custom Logo for API Provider – a very basic customization.
     1. Add Swagger Page
        + Click on Developer Portal menu item then New Page ->New Page  
          
        + Fill it in as follows.

Add a name and Add Path: /testdocsExpand *Advanced options* and click *Liquid Enabled*Paste in Code snippet ( below in *Appendix 2 – Dev Portal Swagger JavaScript)*   
Replace "<system-name>" with your Swagger System Name above (5.a.iii.1), *flights-docs*Create Page  


* + - * After Create Page, Publish it. Note there are 2 modes you can save in: Draft (clicking Save) and Published- obviously clicking Publish.
    1. Set your Dev Portal credentials.
       - Open Developers menu item, then open the Developer Account.  
         
       - Click on the 1 User, click the user.  
         
       - Edit then add a password and Update User  
         
    2. Login to the Dev Portal. Note be aware we are now switching the other web console – the one we setup for API Consumers, where ***users*** of the API (e.g. Expedia or CheapOAir) onboard and learn about the API.  
       - On Developer Portal Menu, click Visit Developer Portal. This is the out of the box Developer Portal.  
         
       - Sign in – using email and password of the user you just edited.
       - Inside Developer Portal, choose the Documentation menu or append the path you added to your Portal page you defined above and hit that URL, e.g. https://atlanta-training.3scale.net**/testdocs**  
         Swagger page appears.
       - You can TRY IT OUT! As you did previously inside the Admin Portal.
    3. Add custom Logo – example of very basic Developer Portal customization.
       - Back in the Admin console, in the Developer Portal section, add a new File  
           
           
           
           
           
           
           
         
       - Set the **Section** to be *images* and the **Path** to be */images/threescale-logo.png*, **Choose File** and browse to threescale-logo.png that accompanies this document (You’ll need to download it off the google drive and save it  
         locally on your hard drive first)   
         Create File then Save
       - Now Add this file to the main layout page.  
         Scroll down and click the Main Layout – Left Hand Side near the bottom.  
         

Line 46. Replace line: <a class="navbar-brand" href="/">{{ provider.name }}</a>

With:

<div class="logo">

<a href="#">

<img src="/images/threescale-logo.png " alt="" style="height:50px; width:150px;">

</a>

</div> ***Publish*** it.

* + - * Go Back to the Dev Portal and refresh.
      * ADJUST THE WIDTH/HEIGHT (in red above) TO LOOK RIGHT

You’re done! You’ve completed the steps to do a minimal 3scale POC!

# **Appendix 1 – Swagger**

{

"swagger": "2.0",

"info": {

"version": "1.0.0",

"title": "Flights API",

"description": "Flights Demo API",

"termsOfService": "http://swagger.io/terms/",

"contact": {

"name": "The Flights Demo Team",

"email": "demo-flights@redhat.com",

"url": "http://demo-flights.redhat.com"

},

"license": {

"name": "MIT",

"url": "http://github.com/3scale/demo-flights/LICENSE-MIT"

}

},

"host": "<yours>",

"basePath": "/",

"schemes": [

"https"

],

"paths": {

"/flights/intl/flights": {

"get": {

"description": "Returns a list of flights",

"operationId": "Returns a list of flights",

"parameters": [

{

"name": "user\_key",

"in": "query",

"description": "API/User Key",

"required": true,

"type": "string",

"x-data-threescale-name": "user\_keys"

}

],

"responses": {

"200": {

"description": " Flights response",

"schema": {

"$ref": "#/definitions/flights"

}

},

"403": {

"description": "Authentication failed",

"schema": {

"$ref": "#/definitions/AuthenticationFailed"

}

},

"default": {

"description": "unexpected error",

"schema": {

"$ref": "#/definitions/Error"

}

}

}

}

}

},

"definitions": {

"flights": {

"type": "object"

}

}

}

# **Appendix 2 – Dev Portal Swagger JavaScript**

<h3>Active Docs/Swagger 2.0 Documentation</h3>

{% active\_docs version: "2.0" services: "<system-name>" %}

<script type="text/javascript">

$(function () {

window.swaggerUi.load();

});

</script>