

## 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  
<http://my-json-server.typicode.com/btierney/threescale/employees>
2. Demonstrate the main 3scale API Management features typically required for a *basic POC* – including
  - a. Access Control
  - b. Policy Enforcement
  - c. Analytics
  - d. Swagger & Basic Developer Portal Configuration
3. Gain familiarity with the 2 Web Portals 3scale exposes
  - a. Admin Portal. Used by: The API Provider – or organization **exposing** the API, e.g. United Airlines.
  - b. 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.



## API > Integration & Configuration

[edit integration settings](#)

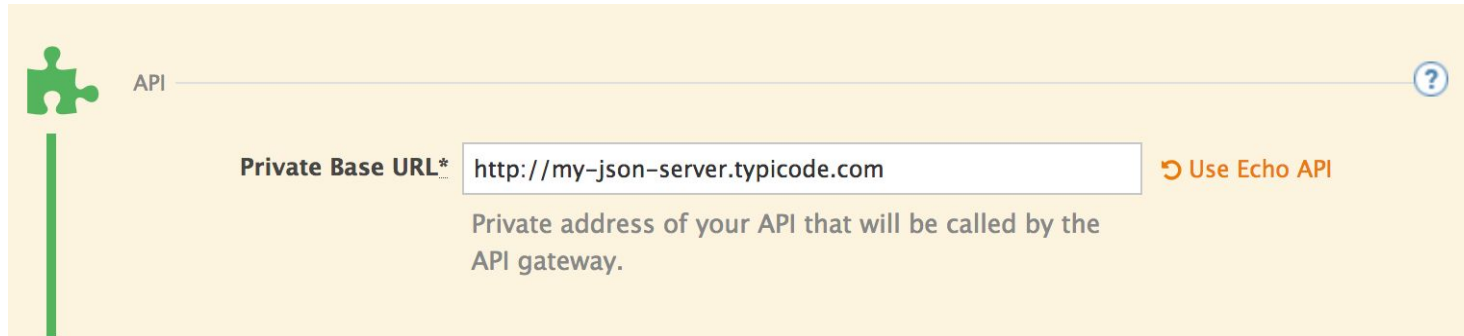
### Integration settings

Deployment Option: APICast

Authentication: API Key (user\_key)

To get started with this service on APICast, **add the base URL of your API and save the configuration.**

- Use the latest APICast
- This is where you configure your gateway which will communicate with the 3scale Platform. Make the following entries:
  - i. Private Base URL\*: <http://my-json-server.typicode.com>

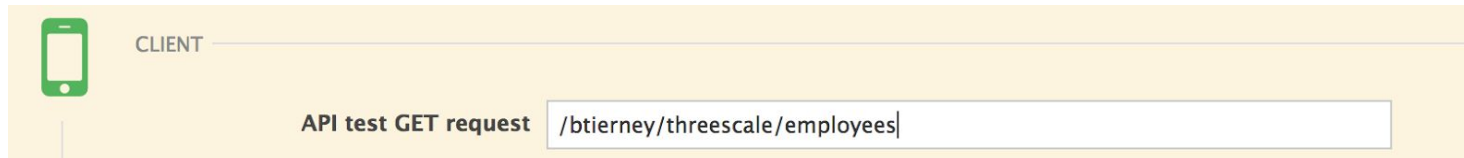


API

Private Base URL\*  [Use Echo API](#)

Private address of your API that will be called by the API gateway.

- ii. In API test GET request, enter “/btierney/threescale/employees”. Then click *Update & Test Staging Configuration*



CLIENT

API test GET request

v

**Note (Gotcha):** If this your second API (aka Service), you will need to create an Application Plan then an Application. Application Plan creation is straightforward. Application creation is done by choosing Account->Applications->Create Application->Choose your new Application Plan and give it a system name and name and create. Then go back to the Integration screen and you will have a user\_key

- iii. 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*.

MAPPING RULES

Verb

Pattern

+

Metric or Method

(Define)

GET

/

1

hits

+

Add Mapping Rule

AUTHENTICATION SETTINGS

- Create a *New Method*
- Name it something like ***get-employees*** (System and Friendly)

### API > New Method

Friendly name\*

get-employees

e.g. Create new user

System name\*

get-employees

e.g. users/create or create\_user. Spaces are not permitted.

Description

Get a list of all employees


Create Method

- Add a Mapping Rule to it back on Integration screen.

Method	System Name	Unit	Description	Mapped	<a href="#">New method</a>
get-employees	get-employees	hit	Get a list of all employees	<a href="#">Add a mapping rule</a>	





- Replace the default Mapping to *hits* by editing it.

▼ MAPPING RULES

Verb	Pattern		Metric or Method (Define)
GET	/	1	hits 

[Add Mapping Rule](#)

- Replace the Pattern with the URL path to our endpoint (`/flights/intl/flights`) and select our new method *get-flights*. Use the same path (`/btierney/threescale/employees`) in API test GET request then click *Update and Test Staging Configuration*.

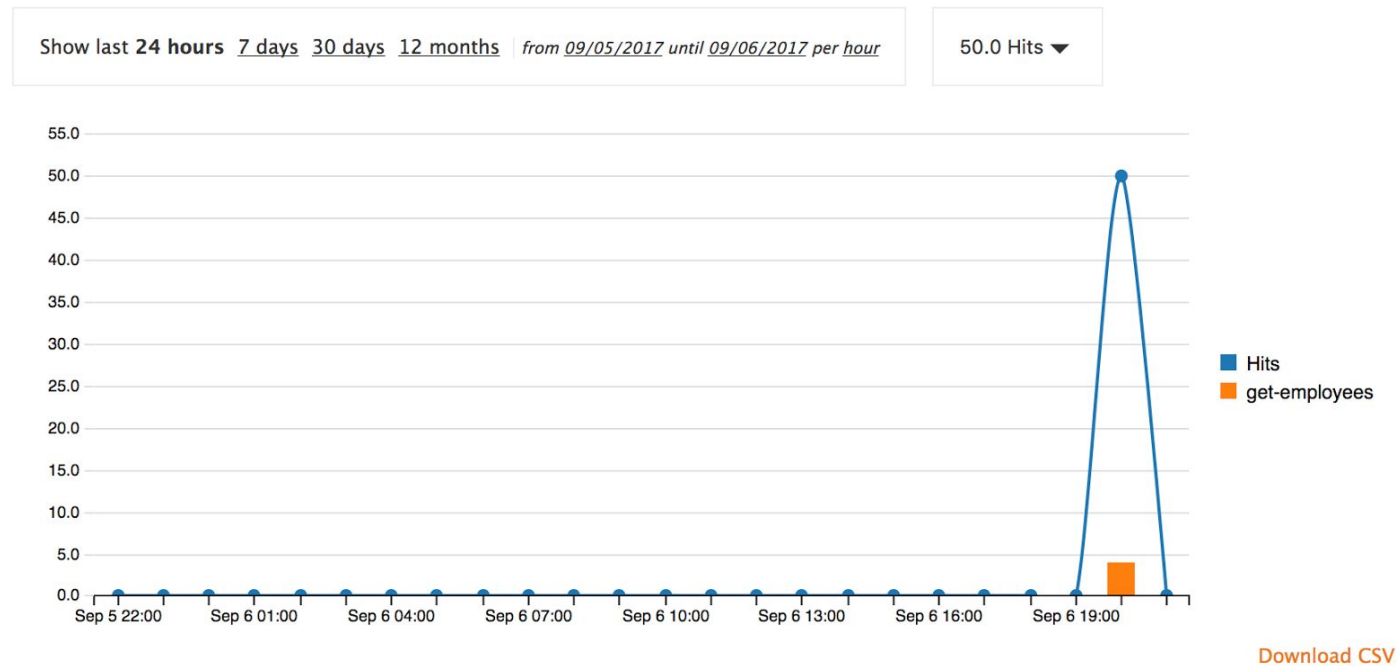
Verb	Pattern	+	Metric or Method (Define)
GET	/btierney/threescale/employees	1	get-emplo  
<a href="#">+ Add Mapping Rule</a>			
<a href="#">▶ AUTHENTICATION SETTINGS</a>			
<a href="#">CLIENT</a> 			
<div>  <div> <p><b>API test GET request</b> /btierney/threescale/employees</p> <p>Optional GET request to a API gateway endpoint. We will use this call to validate your API gateway setup using credentials of the first live application. You can try it yourself by copying the following command into your shell:</p> <pre>curl "https://api-2445582123876.staging.gw.apicast.io:443/btierney/threescale/employees?user_key=2978a250d77be980457a9a6ccb183141"</pre> </div> </div>			
<p>Hit the test button to check the connections between client, gateway &amp; API.</p>			<a href="#">Update &amp; test in Staging Environment</a>

- 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.

### 3. Access Control and Analytics

- Keep the Integration tab open and open another tab on Analytics -> Usage. Notice your method, *get-flights*

#### API > Usage



- Test Authentication.
  - In a terminal (or browser), hit the URL in the curl statement on the Integration screen you worked on (above) - something like:



CLIENT



API test GET request

/btierney/threescale/employees

Optional GET request to a API gateway endpoint. We will use this call to validate your API gateway setup using credentials of the first live application. You can try it yourself by copying the following command into your shell:

```
curl "https://api-2445582123876.staging.gw.apicast.io:443/btierney/threescale/employees?user_key=bf289a532f14ec46c033cb99b74e0770"
```

It should succeed.

ii. Enter an incorrect user\_key and retry, it should fail.

- Switch to your Analytics tab and refresh. The count for *get-employees* should have incremented

4. 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.



## Application Plans

Application Plans establish the rules (limits, pricing, features) for using your API; every developer's application accessing your API will be accessing it within the constraints of an Application Plan. From a business perspective, Application Plans allow you to target different audiences by using multiple plans (i.e. 'basic', 'pro', 'premium') with different sets of rules.

### Default Plan

Default application plan (if any) is selected automatically upon service subscription.

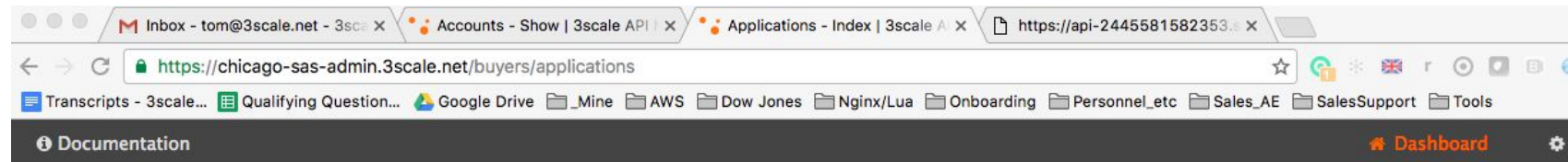
Name	Applications	State			<a href="#">Create Application Plan</a>
Basic	1	published	<a href="#">Hide</a>	<a href="#">Copy</a>	<a href="#">Delete</a>
Unlimited	0	published	<a href="#">Hide</a>	<a href="#">Copy</a>	<a href="#">Delete</a>

- Disable method *get-employees* by clicking its *Enabled* check box

## Metrics, Methods & Limits

Metric or Method (Define)		Enabled ?	Visible ?	Text only ?
Hits	<a href="#">Limits (0)</a>	✓	✓	✓
get-employees	<a href="#">Limits (1)</a>	✗	✓	✓

- 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 2<sup>nd</sup> 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.



Dashboard Developers **Applications** Billing Analytics API Developer Portal Settings

## Applications

<input type="checkbox"/>	Name	State	Account	Plan	Paid? <sup>?</sup>	Created At	Traffic On
<input type="checkbox"/>	Developer's App	live	Developer	Basic	free	October 09, 2016	October 10, 2016

[Export all Applications](#)

- 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-employees* – 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*

Documentation

3scale by Red Hat

Dashboard Developers Applications Billing

Account 'Developer' > Application 'Developer's App' > Analytics | API Request Log

### Developer's App

[Edit](#) [Delete](#)

Description Description of your default application

Service API

**State**

✓ Live suspend

**API Credentials**

**User Key**

1bfabee09972043b1e324d11ff6b2c2a

[Regenerate](#)

[Set Custom Key](#)

**Application Plan: Basic**

**FEATURES**

Unlimited Greetings	✓
24/7 support	✗
Unlimited calls	✗

**LIMITS**

get-flights	0 hit / eternity
Hits	✓

[Customize](#)

**Change Plan**

✓ Unlimited [Change](#)

- 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-employees*
5. 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.
    - Open API->Active Docs menu item on the top left

[Create Service](#)

## API

### Definition, Integration and Settings

Integrated through APIcast Cloud Gateway  
 Authenticated by API key  
 ID for API calls is 2555417736054 and system name is api  
 Users can manage application keys  
 Users can manage applications  
 Users can request plan change  
 Users cannot select a plan when creating an application  
 Users cannot see API log requests

### Analytics



### Latest alerts

There are no alerts.

ii. Open Echo link and the left and Delete it.

### ActiveDocs: Service Specs

Name	System Name	State	Swagger Version
Echo	echo	visible	2.0

Overview **ActiveDocs**

---

**Preview Service Spec (2.0)**

[Hide](#) | [Edit](#) | [Delete](#)

**Echo API**

A sample echo API

default


GET	/
GET	/echo

iii.

iv. Create new Spec.

Documentation Dashboard

---

 Dashboard Developers Applications Billing Analytics **API** Developer Portal Settings

Overview **ActiveDocs** ActiveDocs Spec was successfully deleted.

---

**ActiveDocs: Service Specs**

[Create your first spec](#)

- Name it *employee-swagger* (both Name and System Name) and **click Publish**.

## ActiveDocs: Edit Service Spec

Name\*

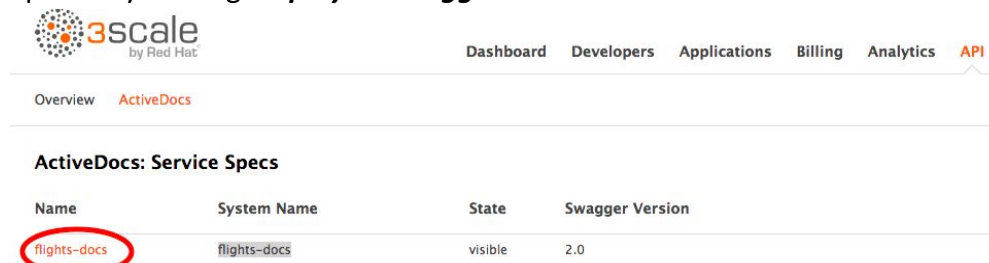
System name\*

Only ASCII letters, numbers, dashes and underscores are allowed.  
Warning: With ActiveDocs 1.2 the API will be described in your developer portal as *System name: Description*

☒ Publish?

- Paste in the **emp-swagger.json** file from <https://github.com/btierney/threescale> into *API JSON Spec*\*
- Replace the **host** you've been using in your curls, e.g. something like **api-2445581074662.staging.apicast.io**
- Create Service

- Open it by clicking **employee-swagger**

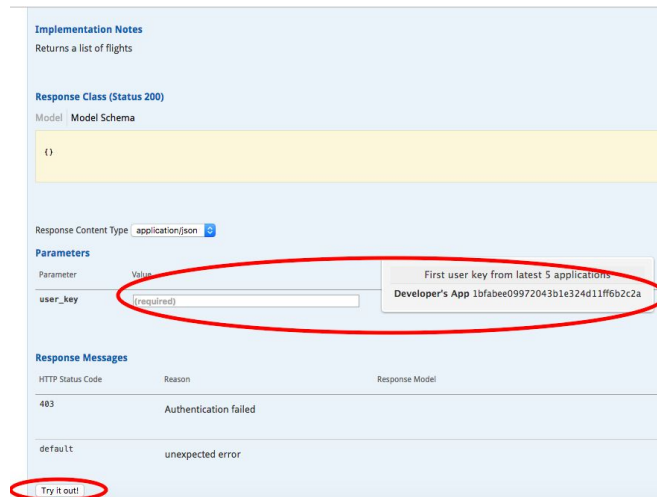


The screenshot shows the 3scale by Red Hat interface. At the top, there's a navigation bar with links: Dashboard, Developers, Applications, Billing, Analytics, and API (highlighted). Below this, there's a sub-navigation bar with 'Overview' and 'ActiveDocs' (highlighted). The main content area is titled 'ActiveDocs: Service Specs' and contains a table with the following data:

Name	System Name	State	Swagger Version
flights-docs	flights-docs	visible	2.0

The 'flights-docs' entry in the 'Name' column is circled in red.

- click in the user\_key field (select pop up) and Try it out!



Implementation Notes  
Returns a list of flights

Response Class (Status 200)  
Model | Model Schema

```
{}
```

Response Content Type: application/json

Parameters

Parameter	Value
user_key	(required)

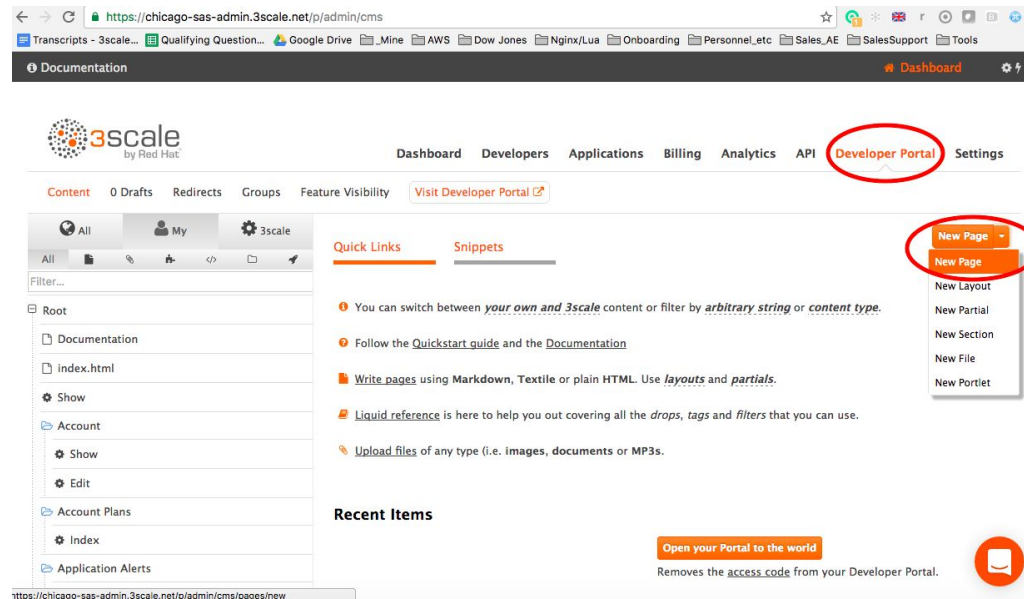
First user key from latest 5 applications  
Developer's App 1bfabee09972043b1e324d11ff6b2c2a

Response Messages

HTTP Status Code	Reason	Response Model
403	Authentication failed	
default	unexpected error	

Try it out!

- **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.
  - i. 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: /testdocs  
 Expand *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



Dashboard Developers Applications Billing Analytics API **Developer Portal** Settings

[Visit Developer Portal](#)

### New page

[New Page](#)

Title\*

Section\*

Path\*

Does not depend on a selected section.

Layout

▼ Advanced options

System name

Content type

Can be HTML, CSS, Javascript or an arbitrary MIME type.

☒ Liquid enable\* Process Liquid tags and drops?

Handler\*

Do you use any markup language?

Tag list

```

1 <h3>Active Docs/Swagger 2.0 Documentation</h3>
2
3
4 {% active_docs version: "2.0" services: "flights-docs" %}
5
6 <script type="text/javascript">
7   $(function () {
8     window.swaggerUi.load();
9   });
10 </script>
11

```

[Create Page](#)

- After Create Page, Publish it. Note there are 2 modes you can save in: Draft (clicking Save) and Published- obviously clicking Publish.

- Set your Dev Portal credentials (should you wish to override default credentials of <developer email>/123456)
  - Open Developers menu item, then open the Developer Account.

Documentation [Dashboard](#)

3scale BY RED HAT

[Dashboard](#) **Developers** [Applications](#) [Billing](#) [Analytics](#) [API](#) [Developer Portal](#) [Settings](#)

[Accounts](#) [Messages](#) [Forum](#)

### Accounts

☐ **Group/Org.** **Admin** **Signup Date** **Apps** **State** [Create](#)

☐ **Developer** John Doe 9 Oct, 2016 1 Approved [Export all Accounts](#)

Documentation [Dashboard](#)

3scale BY RED HAT

[Dashboard](#) **Developers** [Applications](#) [Billing](#) [Analytics](#) [API](#) [Developer Portal](#) [Settings](#)

[Accounts](#) [Messages](#) [Forum](#)

[Accounts](#) > [Account 'Developer'](#) > [1 Application](#) **1 User** [0 Invitations](#) [0 Group Memberships](#) [0 Invoices](#)

### Users of Developer

Name	Email	Created on	Role	State	
<b>John Doe</b>	tcorcora+test@radhat.com	October 9, 2016	admin	active	<a href="#">Edit</a> <a href="#">Suspend</a>

- Click on the 1 User, click the user.
- Edit then add a password, confirm it and Update User

Documentation [Dashboard](#)

3scale BY RED HAT

[Dashboard](#) **Developers** [Applications](#) [Billing](#) [Analytics](#) [API](#) [Developer Portal](#) [Settings](#)

[Accounts](#) [Messages](#) [Forum](#)

[Accounts](#) > [Account 'Developer'](#) > [1 Application](#) [1 User](#) [0 Invitations](#) [0 Group Memberships](#) [0 Invoices](#)

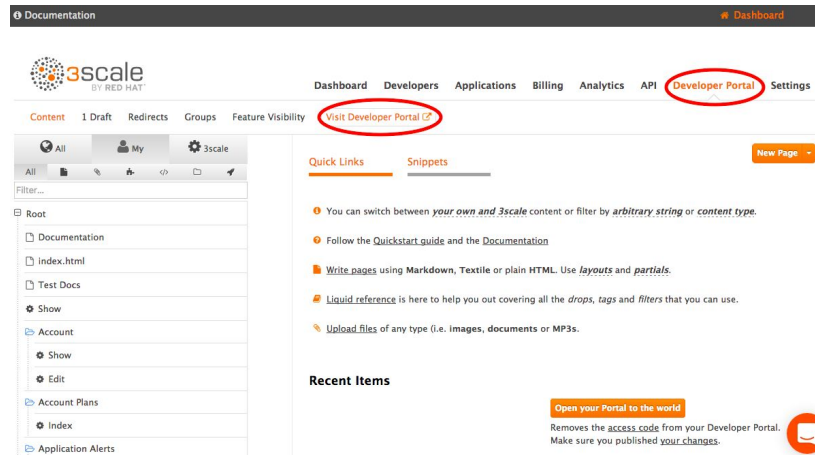
**User John Doe of buyer account Developer**

[Edit](#)

Name John Doe  
Created on October 09, 2016 22:52  
Role admin  
State [active](#) [Suspend](#)  
Username john  
Email tcorcora+test@radhat.com

iii. 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.

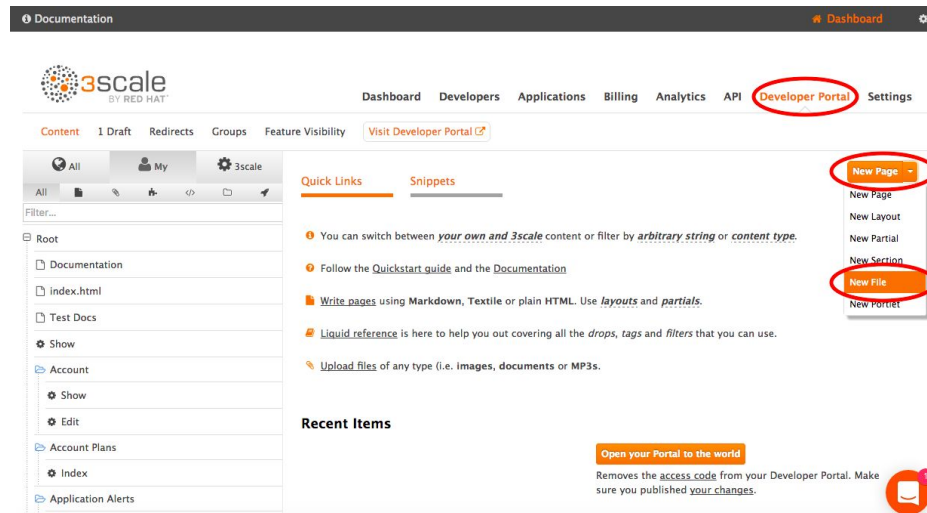
- On Developer Portal Menu, click Visit Developer Portal. This is the out of the box Developer Portal.



- 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://<your host>.3scale.net/testdocs> Swagger page appears.
- You can TRY IT OUT! As you did previously inside the Admin Portal.

iv. Add custom Logo – example of very basic Developer Portal customization.

- Back in the Admin console, in the Developer Portal section, add a new File



## Upload File

Section\*

images

Path\*

/images/threescale-logo.png

☐ Downloadable Checked sets the content-disposition to 'attachment.'

Attachment

Choose File threescale-logo.png

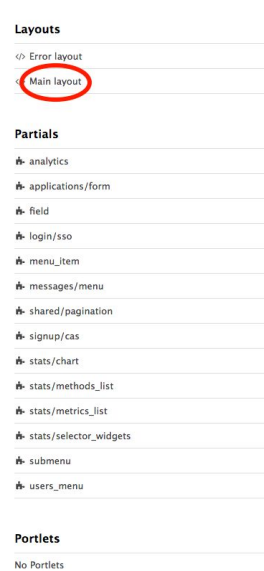
Tag list

New File

Create 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="#">
    
  </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>
```