DigitalLabs

@MMU

DigitalLabs

@MMU

Today

Tasks: Today is a tour of REST APIs and Platforms as a service

Lots to do! *Try and keep to the timings, then we'll have some fun at the end.*

Categories:

- Identity
  - Make sure you are using your development identity. Log into the google account you created previously.
- Editor / IDE
  - We're going to use
    - VSCode + vscode-http-client
  - Sending HTTP requests
- PaaS
  - We're going to use
    - RESTlet cloud
  - Demo a working service
  - Create your own service
    - create an account
    - create a database
    - create a REST API
    - query the API

Before We Start

THE RULES

- Remember: Unless otherwise stated, you should assume your work using on-line resources, like GitHub and Trello will be publicly accessible. It's a fantastic way to start your employment portfolio. *But it can just as easily go the other way.* So:
  - Respect at all times
  - No profanity
  - No self-identification
  - No exposing usernames, passwords, or any other forms of authentication.
- OK? Here we go!

Task 1: Check your basics

(5 mins)

Google Account

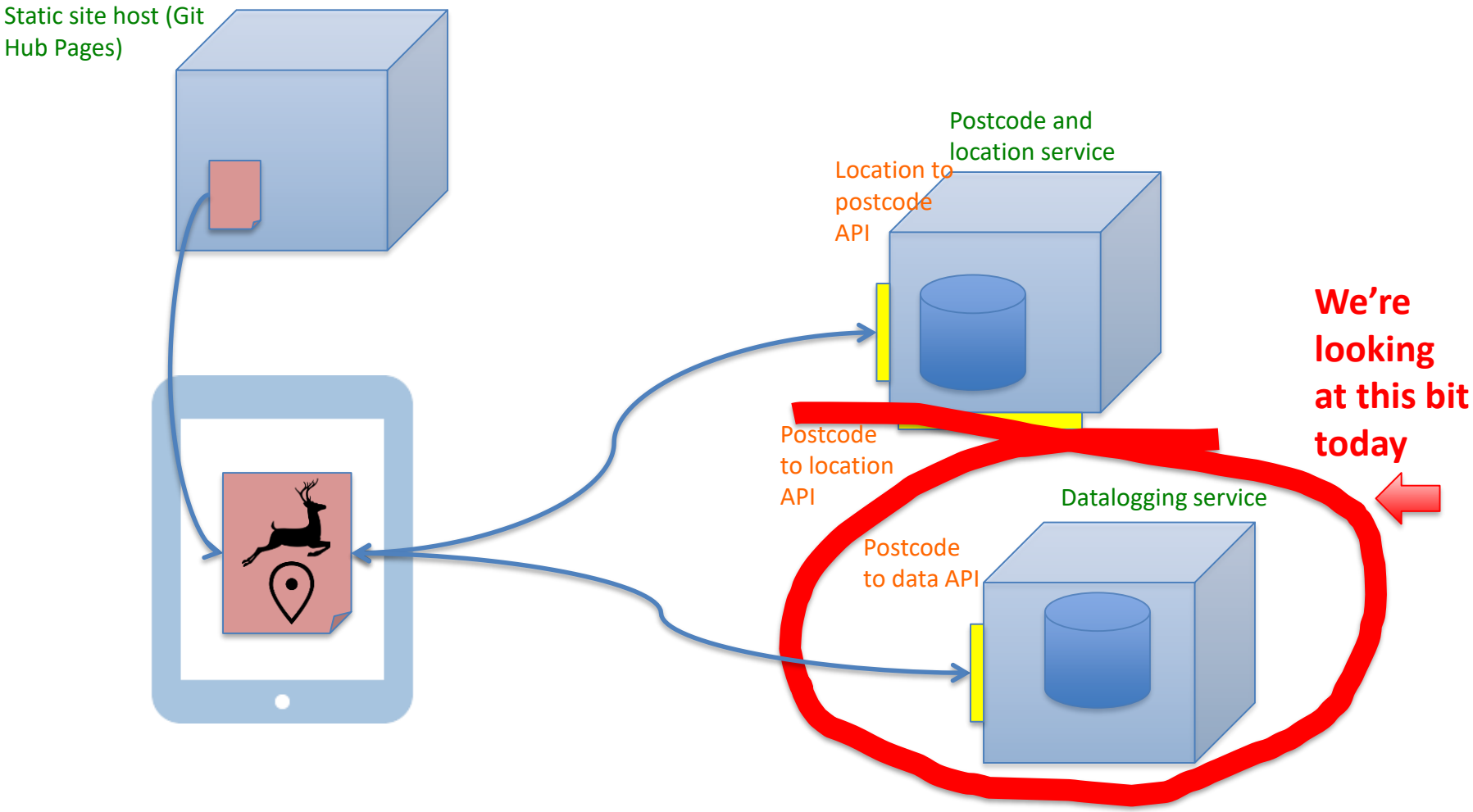
- Make sure you have your 'development account' available
  - *This is the account we created at the beginning of the year*

VSCode

- Make sure VSCode is available, and you have installed the following plugin:
  - vscode-http-client

Task 2: SightingsAndThings API 1

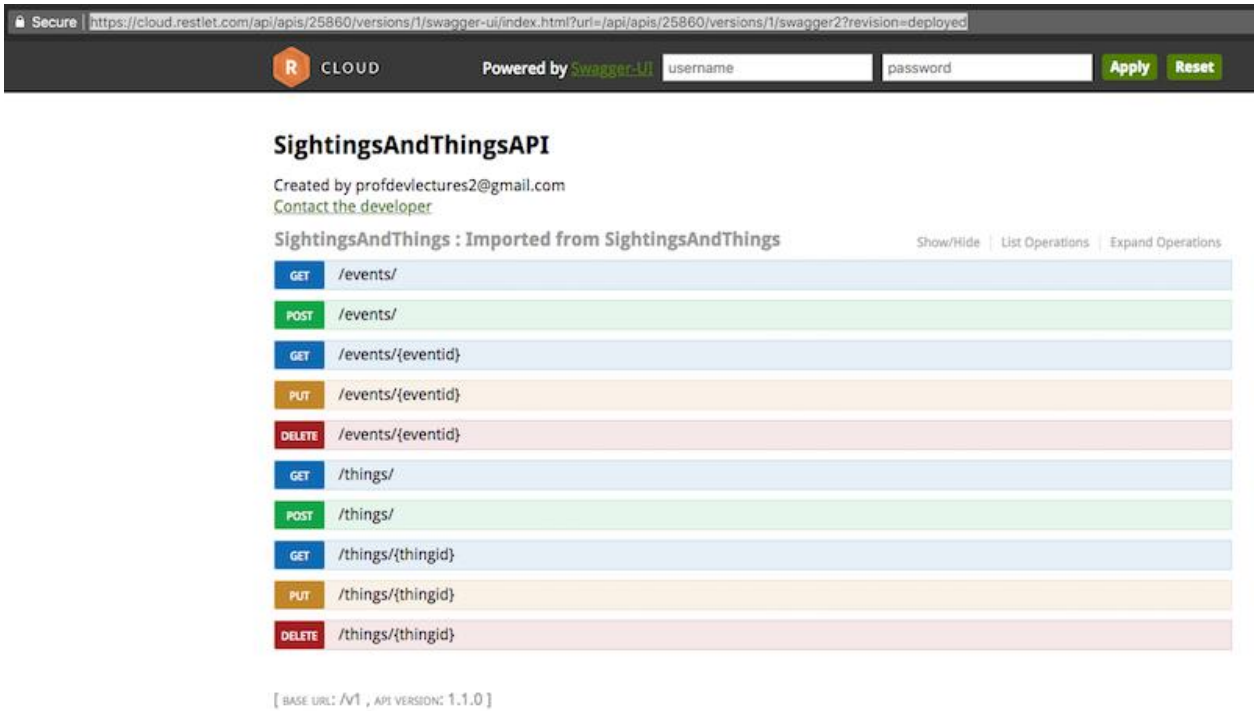
Recap: Remember the Wild Logging App? It logs sightings of wildlife.



Task 2: SightingsAndThings API 2

Swagger UI: In this part, we'll use the service documentation to query the API

Click on SightingsAndThings Service



This documentation has been generated automatically.

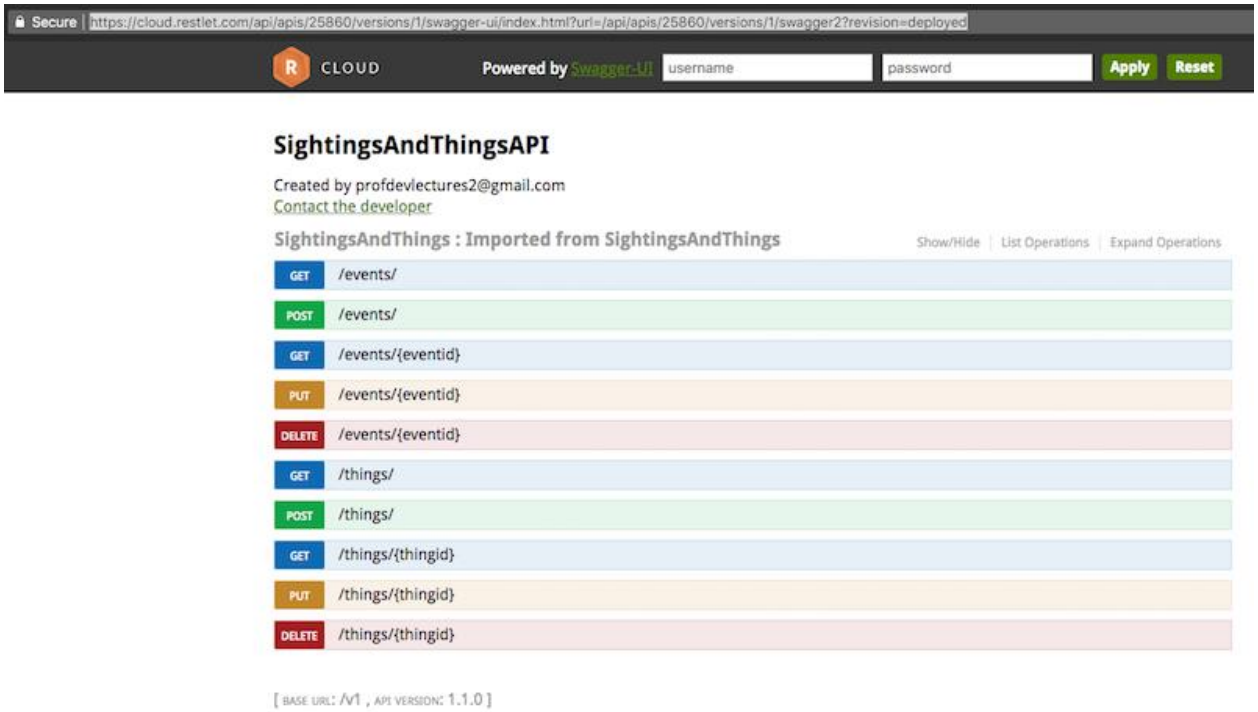
It is served from the same server which exposes the API.

It is a web app, and can be used to query the API.

The API can be registered on API Commons

Task 2: SightingsAndThings API 3 (2 min)

Click on SightingsAndThings Service



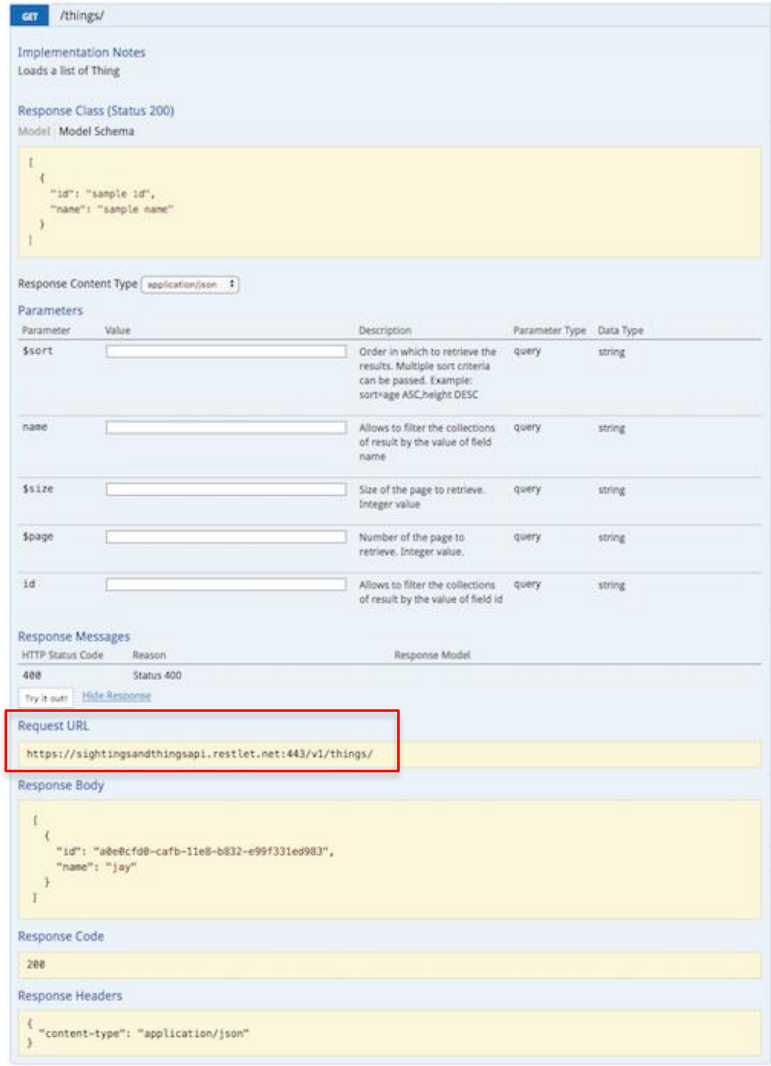
Q: There is no authorisation on this API. Is that a good idea? Why?

Task 2: SightingsAndThings API 4 (5 min)

1. Expand the GET /things/ entry
2. Click ‘try it out’
   - What do you get?

3. Try some other items
   - What are they doing?

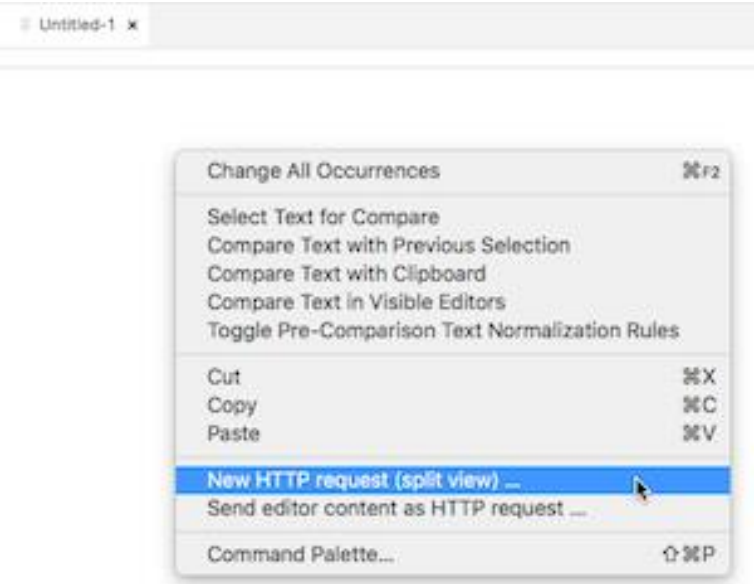
4. Take note of the Request URL
   - (you’ll need it in a bit)



Task 2: SightingsAndThings API 5 (2 min)

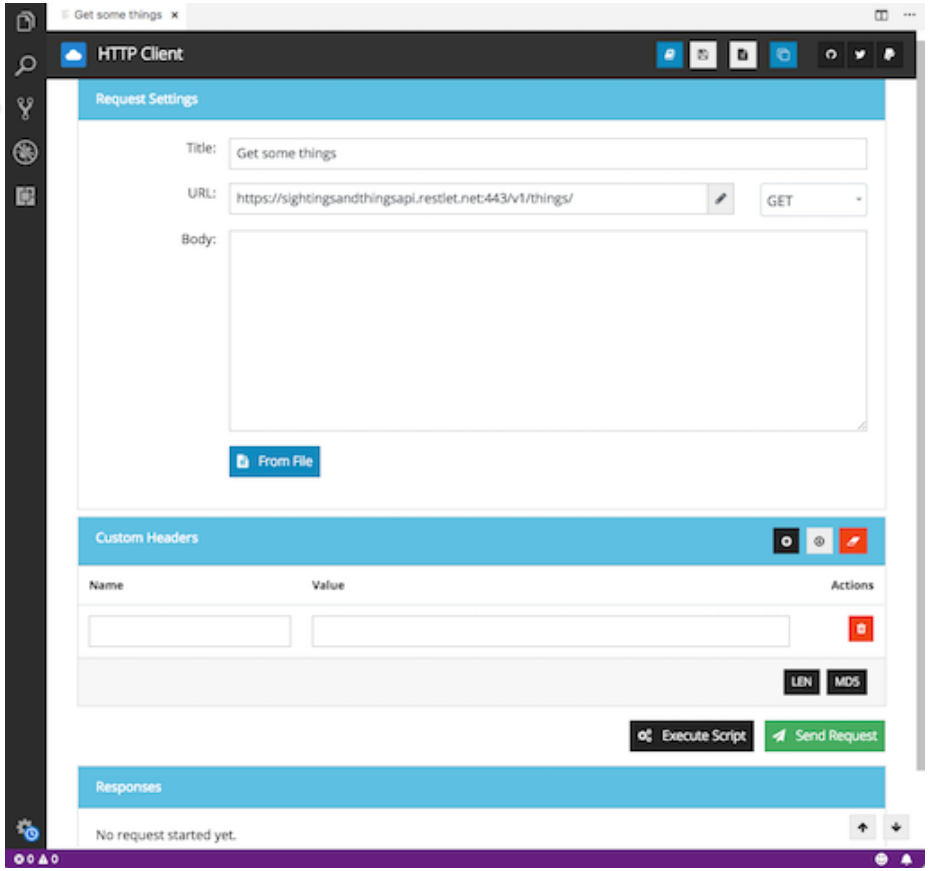
VSCode: In this part, we'll use VSCode and the HTTP plugin to query a REST API

1. Create a new VSCode instance
2. Create a new, untitled file
3. Right-click, in the working area:

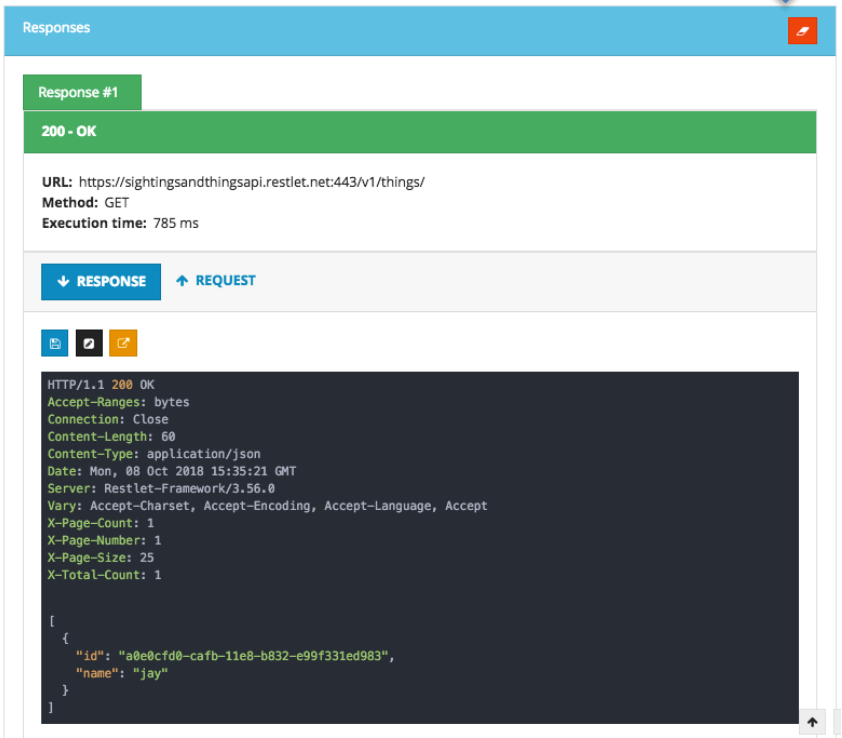


Task 2: SightingsAndThings API 6 (5 min)

1. Fill-in the http-client, like this
2. Click on ‘Send Request’



3. Confirm you can see some data:



Task 3: Create your own RESTful service 1 (5 min)

DigitalLabs @MMU

Service: In this part, we'll create a free account on RESTlet, and use it to re-create the UrbanWild's back-end.

1. In your Google Drive, create a folder:
   `accounts/restlet`

2. Use Dillinger to create a Markdown file:
   `credentials.md`

3. Use your new Google Development Identity to create a new Restlet account.

4. Put your account credentials in the file

5. Choice this option when your account is created:

Remember to check your mailbox! Your account needs to be validated.

Remember to do this with any account you create with this identity.

Task 3: Create your own RESTful service 2 (3 min)

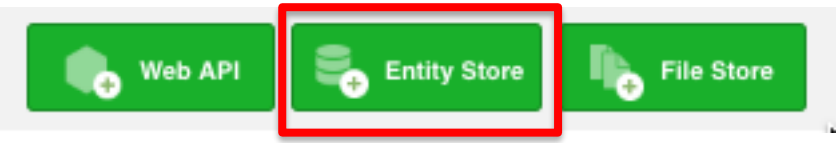
Service: In this part, we'll create an entity store to hold things and events

1. You now have a free account with RESTlet:
   - 1 free api
   - 1 free entity store (DB)

2. RESTlet will ask you for an API and Domain name.
   - Close the dialog; we'll do it later

3. Now we have an empty account!

4. Click the button to create an entity store



5. In the 'create' dialog, specify:
   1. Type: FULL STACK
   2. Name 'Sightings And Things'

Task 3: Create your own RESTful service 3

Service: Here are the entities which UrbanWild will be storing

1. Entities:
   - Thing – the wildlife we saw
   - Event – where and when we saw it

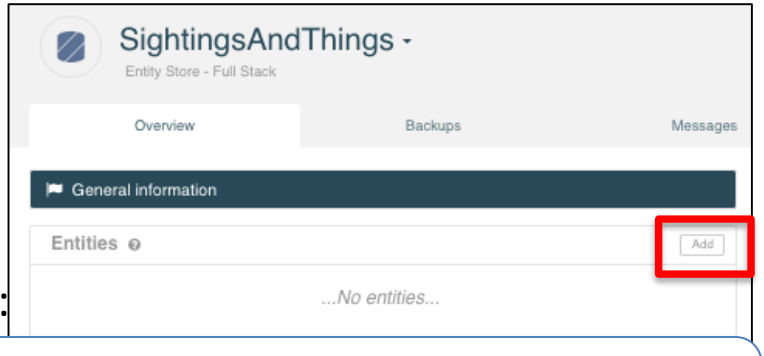
2. Thing:
   - id (String) – used to refer to a unique item (it's a database)
   - name (String) – the species of thing we saw

3. Event:
   - id (String) – used to refer to a unique item
   - thing (Thing) – reference to the thing we saw
   - date (Timestamp) – the data and time it was seen
   - postcode (String) – an easy-to-search location reference
   - lat (Double) – latitude (degrees). A plottable location reference
   - lon (Double) – longitude (degrees).

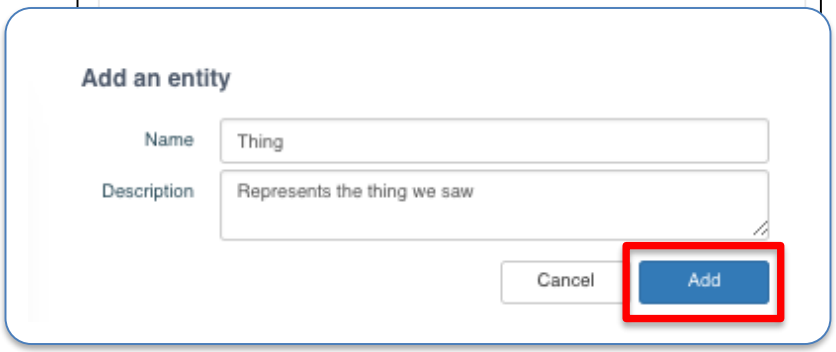
Task 3: Create your own RESTful service 4 (3 min)

Service: In this part, we'll define the Thing entity in RESTlet's entity store

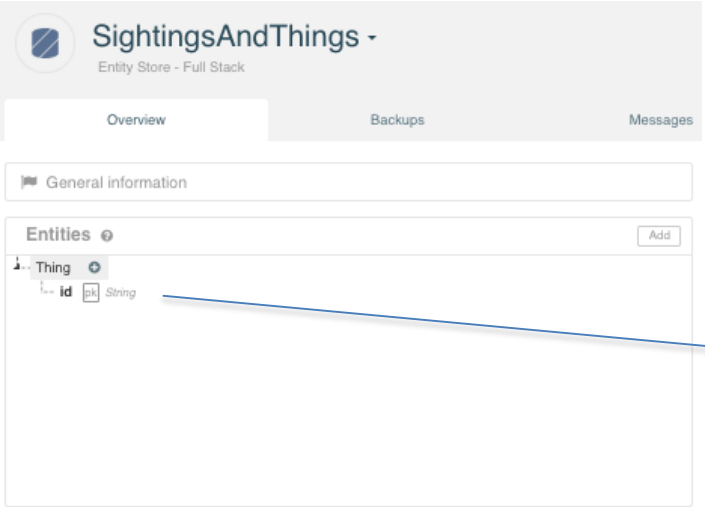
1. We're looking at our new, empty entity store:



2. We're going to add a 'Thing' entity:
   - Click on Add:
   - In the dialog, name & describe the entity:
     - Click on Add.



- Confirm you can see the 'Thing' entity:

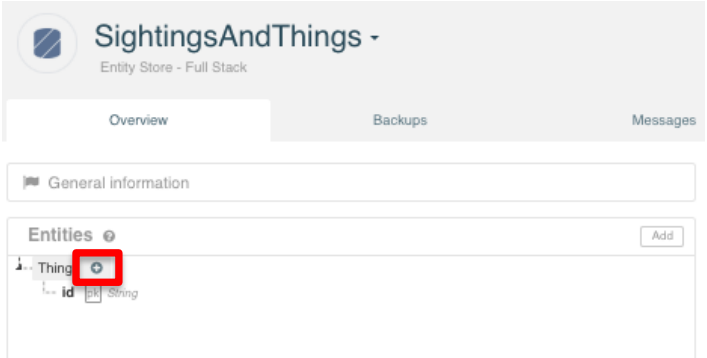


Note: RESTlet has already populated 'Thing' with an id

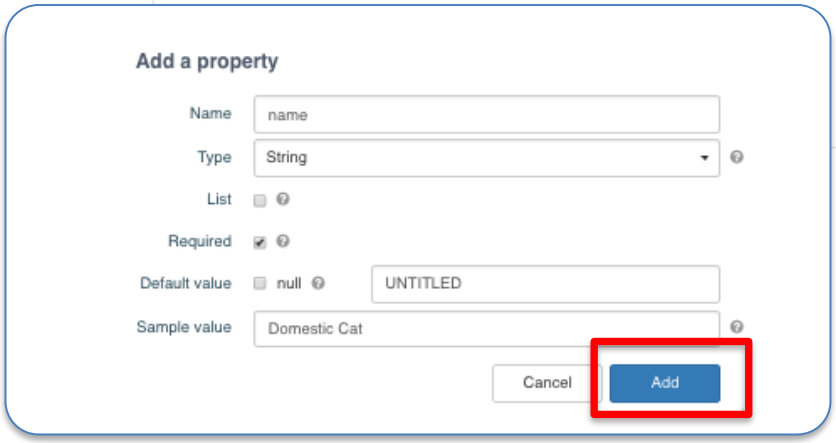
Task 3: Create your own RESTful service 5 (3 min)

Service: In this part, we'll populate the Thing entity with typed properties

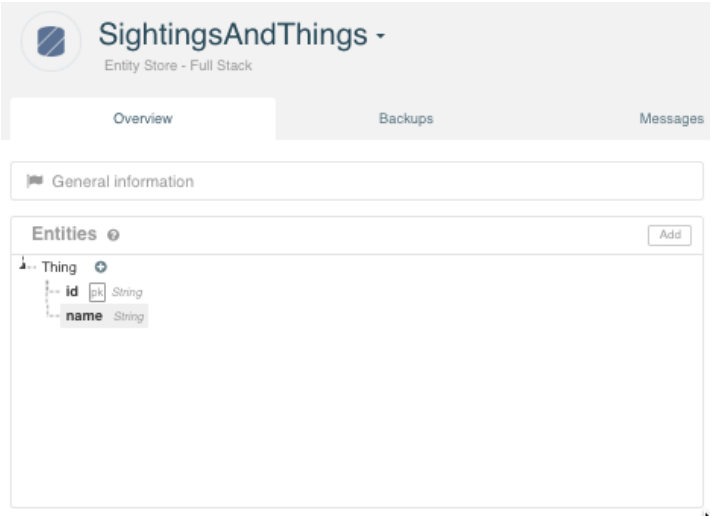
1. We're looking at our entity store:



2. We're going to add a property to the 'Thing':
   - Click on '+':
   - In the dialog, name & describe the property:
     - Click on Add.



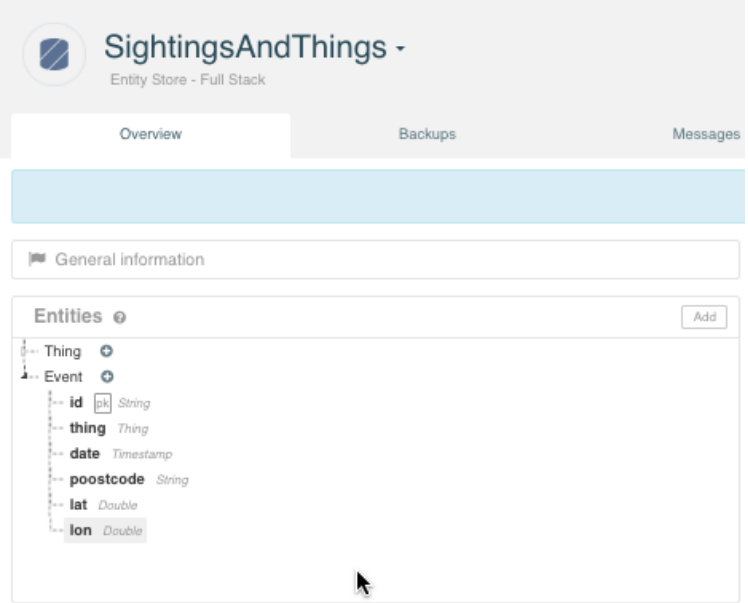
- Confirm you can see the new property:



Task 3: Create your own RESTful service 6 (10 min)

Service: In this part, you'll create and populate the 'Event' property

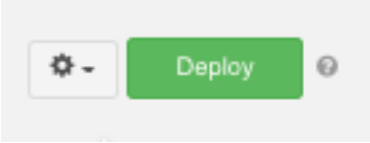
1. As you did for 'Thing', create an entity called 'Event'
2. Define the properties of Event:
   - id (String) – used to refer to a unique item
   - thing (Thing) – reference to the thing we saw
   - date (Timestamp) – the data and time it was seen
   - postcode (String) – an easy-to-search location reference
   - lat (Double) – latitude (degrees). A plottable location reference
   - lon (Double) – longitude (degrees).
3. Here's what you should finish with:



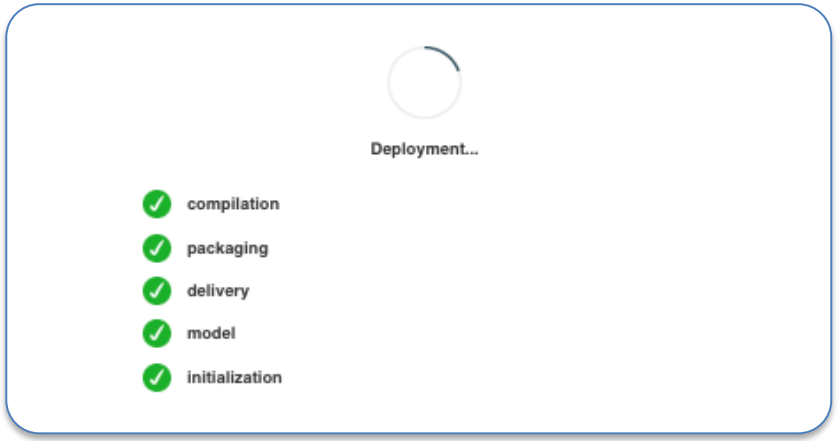
Task 3: Create your own RESTful service 7 (2 min)

Service: In this part, you'll deploy your entity store

1. You've defined your entity store, but to do anything, it needs to be deployed. Click the 'Deploy' button!



2. You'll see Restlet go through the deployment process:



1. Now you have an online database.
2. Each entity is a table in the database.

Task 3: Create your own RESTful service 8 (3 min)

Service: In this part, you'll create some data entries

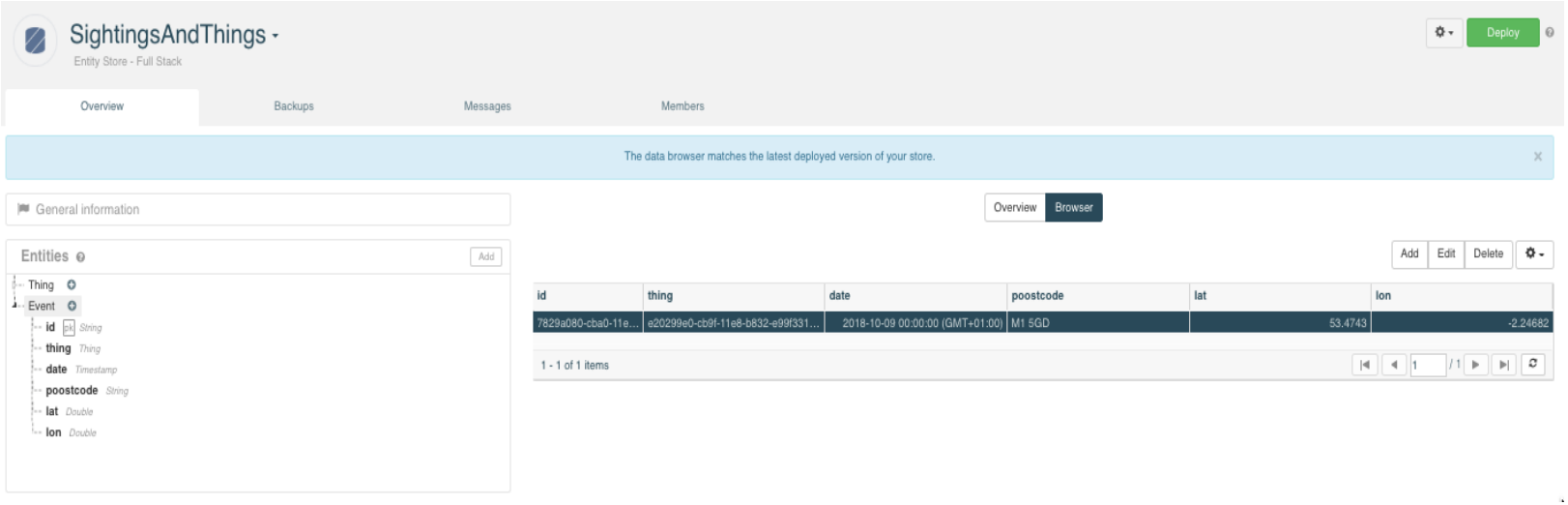
1. Add an entry for a Thing:

1. Click on the Entity you want to add
2. Click on 'Browser'
3. Click on 'Add'
4. Restlet gives you a dialog to add the properties
5. I've added 'Domestic Cat'

Task 3: Create your own RESTful service 8 (3 min)

Service: In this part, you'll create some data entries

1. Add an entry for an Event:



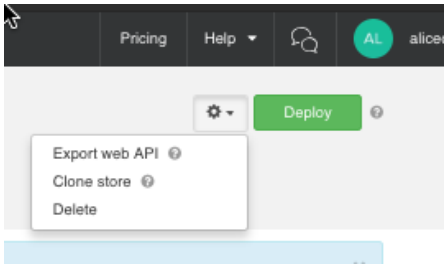
1. Copy the Id of the Thing you want to reference
2. Add a new event
3. Give it the post code of the The Shed (M1 5GD)
4. Give it the approximate location:
   - lat: 53.474300
   - lon: -2.246820

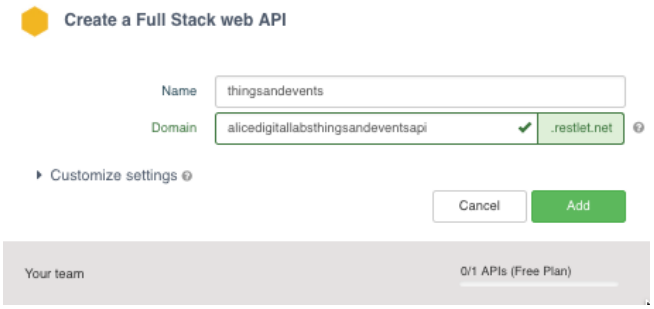
Task 3: Create your own RESTful service 9 (3 min)

Service: In this part, you'll create an interface definition from the Entity Store

You have entered a definition which Restlet has used to create a database schema and populate a database for you.
Restlet will use the same definition to create a general-purpose CRUD API for you:

1. We're looking at our Entity Store:
   - Click the 'gear'
   - Choose 'Export Web API'
   - Specify:
     - a name for the API
     - a unique domain
   - Click on 'Add'





2. Restlet will create an interface definition for you.

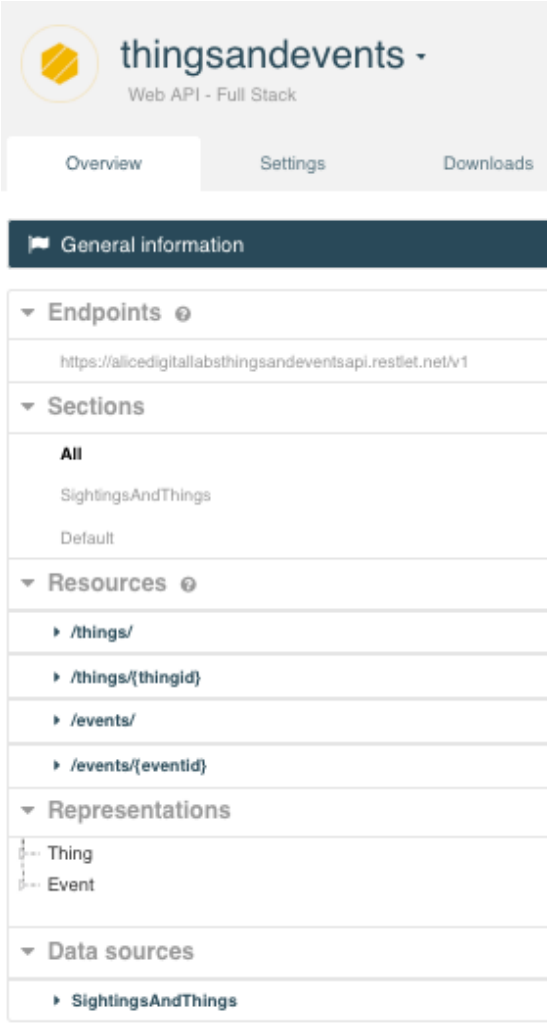
Task 3: Create your own RESTful service 10

Service: In this part, you'll confirm your API

Here's the API on my account:

How did yours go?

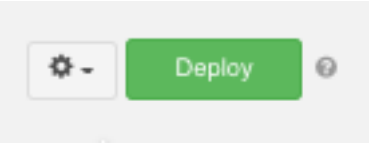
We're not quite ready yet – we need to deploy.

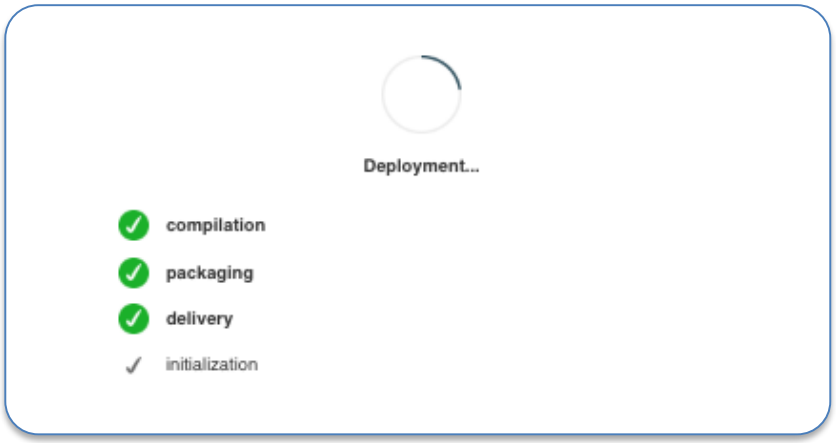


Task 3: Create your own RESTful service 11 (2 min)

Service: In this part, you'll deploy your API

Deploy the API in the same way as for the Entity Store: Click the 'Deploy' button!



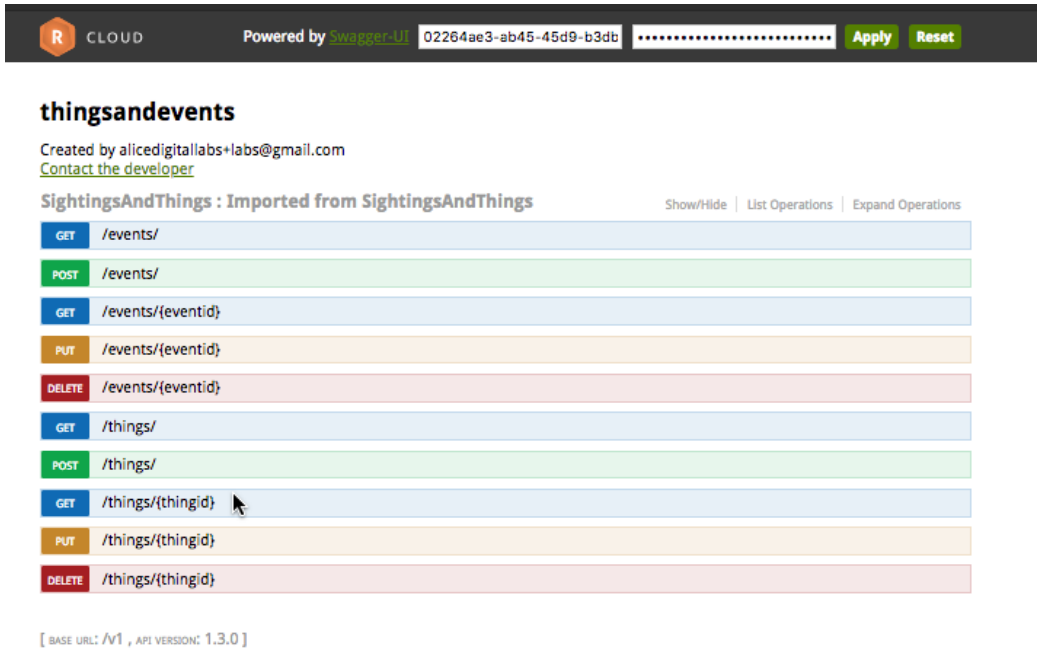


The API is now deployed, and ready to use.

Task 4: Query your RESTful service 1 (3 min)

Service: In this part, you'll query your API

Take a look the the Swagger Interface:



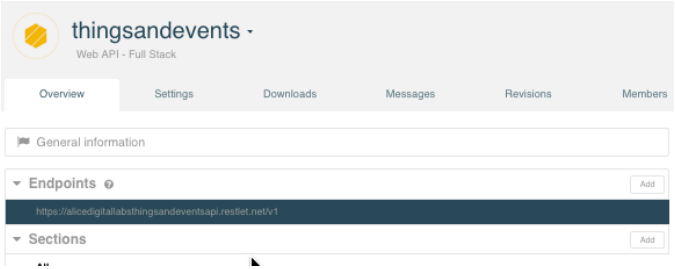
- Try a query to find the Thing you defined, by the name you gave it.
- Try a query to find the Event associated with the Thing
- Compare with the SightingsAndThings service. What do you notice?

Task 4: Query your RESTful service 2 (2 min)

Service: In this part, you'll find access control credentials for your API

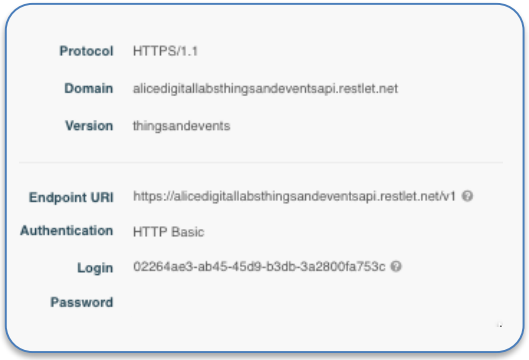
We're looking at the API Definition:

- Note, the endpoint is selected.



Take note of the detail for the endpoint:

- Authentication:
  - HTTP Basic
  - Login
  - password

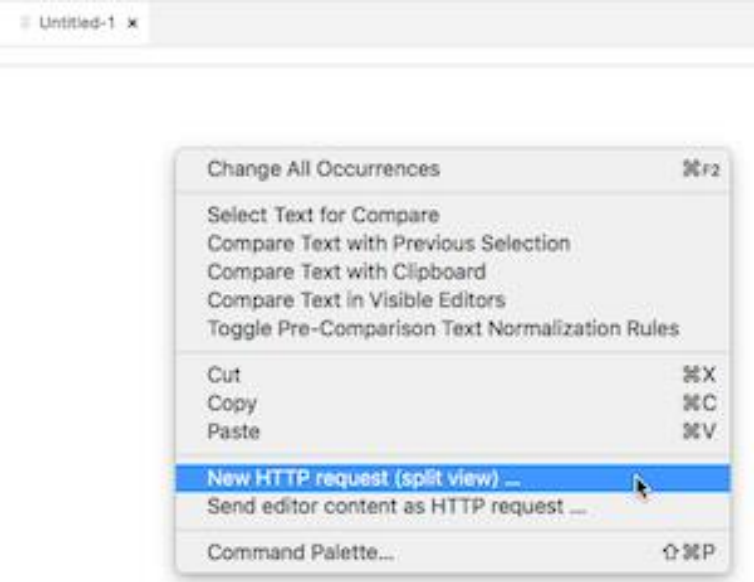


This is the access control for the API. It is a 'Shared Secret'
Clients must use this each time they make an HTTP request to the API.
There are many different types of Authentication. It's up to you which you apply to a service.
Shared Secret is unsuitable to secure data if the client is unsecured. For instance, a Javascript client. In this case, it's OK to use it to discourage casual use.

Task 4: Query your RESTful service 3 (5 min)

Service: In this part, you'll query your service using VSCode's HTTP client plugin

1. Create a new VSCode instance
2. Create a new, untitled file
3. Right-click, in the working area:

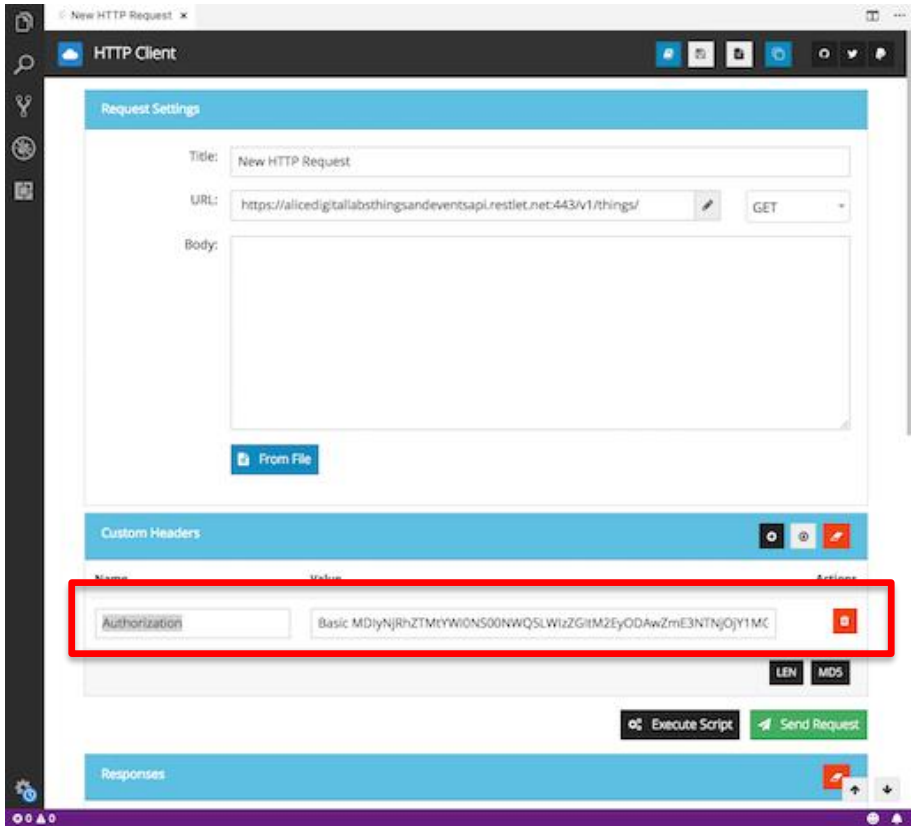


To access the API from a client, you will need to use Basic Access Authentication, for each HTTP request that you make.
To do this, you need to add the 'Authorization' property to the HTTP header.
Construct the value string like this:

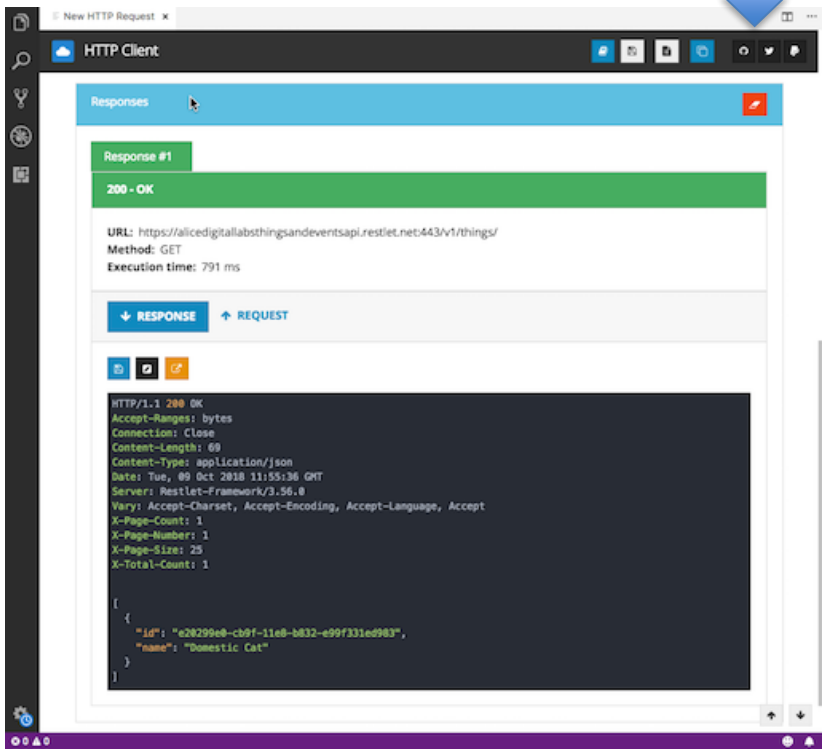
"Basic" + " " + base64encode(login + ":" + password)

Task 4: Query your RESTful service 4 (10 min)

1. Fill-in the http-client, like this
2. Click on ‘Send Request’



3. Confirm you can see some data:



Congratuations!

This week you have:

1. Figured out what the JAM stack is
2. Looked at what goes to make up an API
3. Queried an API using a tool
4. Created your own data service with its own API
5. Secured the API (Just..!) with a shared secret
6. Made authenticated queries to the API

Did you know..?

1. The interface you have created is the same as that used by the Urban Wild application.
2. To have your own Urban Wild application, clone the Urban Wild UI project, and point it at your Restlet web service instead.

Another thing…

1. Restlet will export the API as IDL in the form of YAML.
2. You can import the YAML into the Swagger toolchain.
3. Swagger will export a skeleton Nodejs server which exposes your API.
4. Why not play with it as localhost?

Yet Another thing…

1. Your skeleton server isn't connected to anything!
2. Find out how to connect a PostgreSQL DB
   - Here
3. Your server has no protection!
4. Find out how to add an Authentication layer
   - Here
5. Your client doesn't work anymore!
6. Find out how to handle authentication with your client
   - Here

DigitalLabs

@MMU

DigitalLabs

@MMU

SustainableCreativeAmbitiousSuccessful