**Oracle Cloud Test Drive for PaaS and IaaS**

Lab 1.0

API Platform Cloud Service API First Lab

Contents

[1 Lab Scenario 3](#_Toc508123882)

[1.1 Logging into the Apiary.io 3](#_Toc508123883)

[1.2 The Apiary.io Environment 4](#_Toc508123884)

[1.2.1 Apiary Editor 4](#_Toc508123885)

[1.2.2 Interactive Documentation 4](#_Toc508123886)

[1.2.3 Mock Server 4](#_Toc508123887)

[1.2.4 API Inspector 5](#_Toc508123888)

[2 Desinging and Documenting an API with Apiary 5](#_Toc508123889)

[2.1 Create a New API Project 5](#_Toc508123890)

[2.2 Edit your API definition 7](#_Toc508123891)

[2.2.1 Edit the Sample API definition 7](#_Toc508123892)

[3 Apiary Interactive Documentation 9](#_Toc508123893)

[3.1 Update the API design with additional documentation 9](#_Toc508123894)

[3.2 Review the API Documentation 9](#_Toc508123895)

[4 Apiary Mock Server 10](#_Toc508123896)

[4.1 Using the Mock Server to Test your API 12](#_Toc508123897)

[4.2 Access the Mock Service directly 12](#_Toc508123898)

[4.3 Integrated Code Examples 13](#_Toc508123899)

[5 Collaborating Via Documentation Share 13](#_Toc508123900)

[5.1 Sharing Documentation Via URL 13](#_Toc508123901)

[5.2 Sharing Documentation Via Adding Team Members That Can Edit/View 14](#_Toc508123902)

[5.3 Sharing Documentation Via GitHub Sync 15](#_Toc508123903)

[6 Team Work And Rules 17](#_Toc508123904)

[6.1 Team Overview 17](#_Toc508123905)

[6.2 Rules 18](#_Toc508123906)

[7 Apiary API Inspector 20](#_Toc508123907)

[7.1 Use the API Inspector to review API requests 20](#_Toc508123908)

[8 Apiary designed APIs tested using Dredd 20](#_Toc508123909)

[8.1 Fork the Dredd- project from GitHub 20](#_Toc508123910)

[8.2 Git clone, Install Dredd And Dependancies 21](#_Toc508123911)

[8.3 Actions On Apiary.io 24](#_Toc508123912)

[8.4 Actions on Local Dev Environment 25](#_Toc508123913)

[8.5 Run dredd tests 26](#_Toc508123914)

[8.6 View test results on Apiary.io 27](#_Toc508123915)

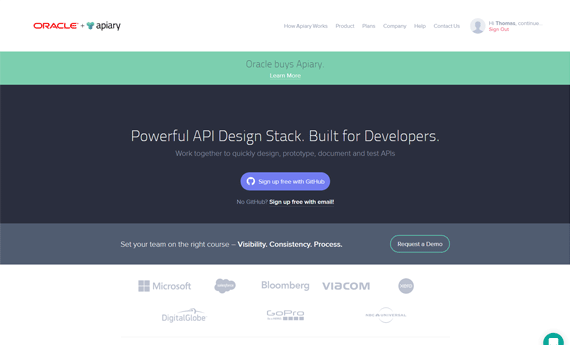
Introduction to the Lab and Logging-in

# Lab Scenario

The objective of this lab is to give you a comprehensive overview of the Oracle Apiary Cloud Service as an API Developer.

Oracle Apiary makes it easy for teams to work together to quickly design, prototype, document and test APIs. API first development, although API developers and consumers to collaborate and validate the design of the API before it is fully developed. This removes the iterative cycle of API development, dissatisfactions from the consumer, re-design while the enterprise adopts and adapts to changing requirements.

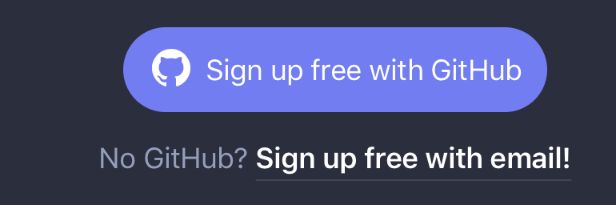
In this lab you will see how easy it is to create, document and test an API using the Oracle Apiary Cloud Service.



**Apiary.io Portal**: This portal is used by App developers, testers, architects, product managers, clients, and partners, allowing effective collaboration, bringing unique perspectives to the design of your APIs.

It provides a dashboard for web-based API Blueprint management, with complete control and visibility to project Role-based access, documentation, API editor, mock server for testing and is pre-integrated with GitHub.

## Logging into the Apiary.io

Access Apiary is free for individual developers but you must signup for an account. (no credit card needed)

Open your browser and navigate to: <http://apiary.io>

Click Signup for free: with GitHub or with email

|  |  |
| --- | --- |
| For GitHub\* | Email Only |
| /var/folders/3c/n2d_jp3x4nn9p8vdr15jsxsh0000gn/T/DIS_357.png/var/folders/3c/n2d_jp3x4nn9p8vdr15jsxsh0000gn/T/DIS_358.png | /var/folders/3c/n2d_jp3x4nn9p8vdr15jsxsh0000gn/T/DIS_354.png |
| Enter a username, email address and password for new accounts. Sinply signing for existing GitHub accounts | |

\*For existing GitHub users, this process will link your GitHub account with Apiary and share your GitHub login credentials for both services.

Check your email for the confirmation email and then complete the registration and login process.

After your first login, the Apiary service will present a short tutorial to get you understand Apiary and get you started, you may review this or continue with the Lab Exercise.

## The Apiary.io Environment

Apiary is focused on solving the fundamental task of improving API development and streamlining the business process of how work gets done.

The API Designer / Developer works with the Apiary tools to create teams who collaborate, design and document APIs, test & inspect mock versions of API services.

The Lab will focus on 4 areas of the Apiary tools:

* Apiary Editor
* Interactive Documentation
* Mock Server
* API Inspector

Let’s go through each component quickly. There is nothing to do here yet.

### Apiary Editor

The Apiary Editor is the foundation of your API design. Apiary Editor supports [API Blueprint](https://apiblueprint.org/) and [Swagger](https://help.apiary.io/api_101/swagger) API Description languages. The Apiary Editor shows a preview of your documentation while providing instant feedback to ensure correct syntax in your API document.

Your API document will be parsed with warnings and errors as you type. These errors will be displayed both in the editor header and on the lines where the warnings and errors exist.

### Interactive Documentation

Apiary interactive documentation is an interactive representation of your API Description for you to not only read and write, but to be a place where you can interact with your API—even before you’ve built it.

### Mock Server

The Mock Server allows you to quickly preview your API and share it with others. It allows you to try out your API as you design it, giving immediate feedback along the way in how it may be used.

The Mock Server accomplishes this by listening for requests as you’ve defined them in your blueprint. When a request is received to your Mock Server for a URL you’ve defined, the corresponding response for that request will be returned.

### API Inspector

The API Inspector to monitors requests to the Mock Server and captures both the requests and responses and allows you to view them.

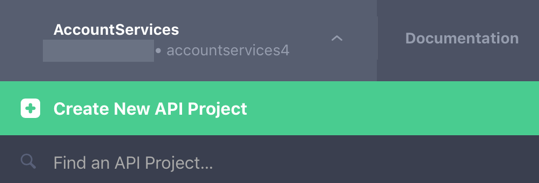
Next, let’s start to implement our scenario.

Lab Execution

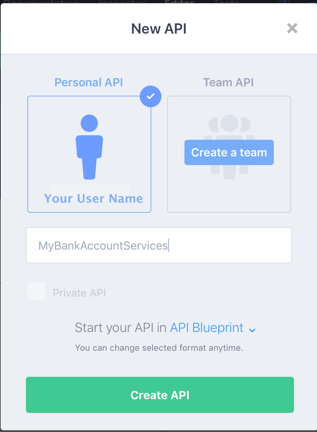
# Desinging and Documenting an API with Apiary

In this first stage, you will design a simple API using the Apiary Editor. You will create a project, which will also create a new GitHub project, if your accounts have been linked. Then you will edit the sample API to design and document your API.

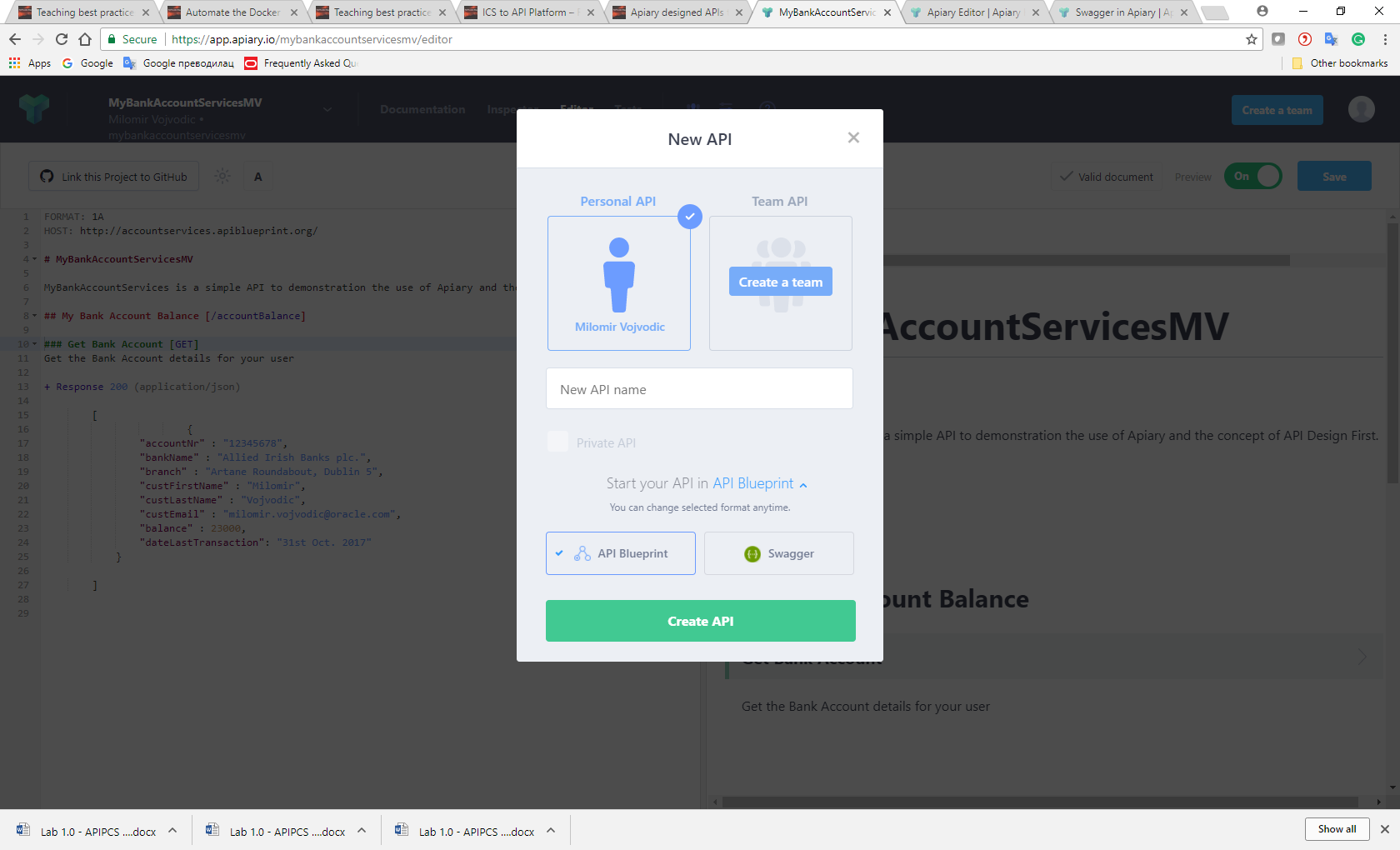
## Create a New API Project

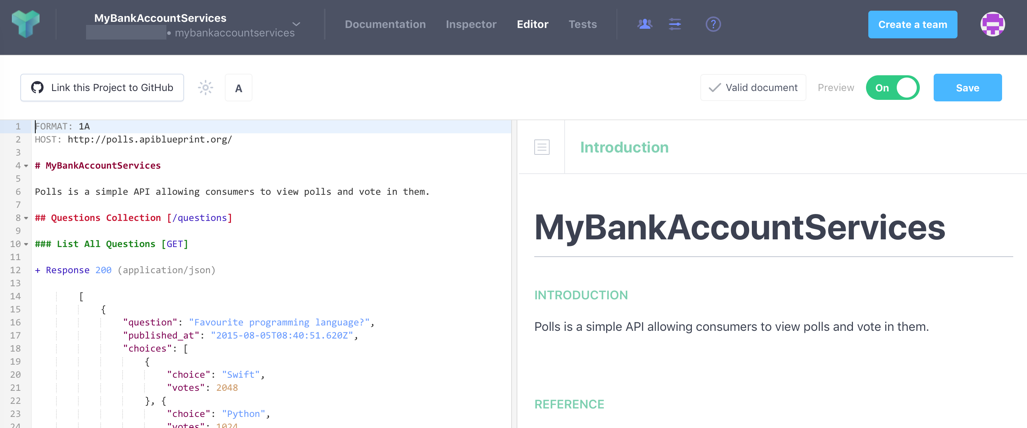
If you’ve already doen so, Log into [Apriary.io](http://apiary.io/).

1. Click on the Account Services menu item in the top left corner
2. Select Create a New API Project



1. Enter **MyBankAccountServices** for the API project name
2. Click Create API
3. Review that you can also select Swagger instead of Blueprint



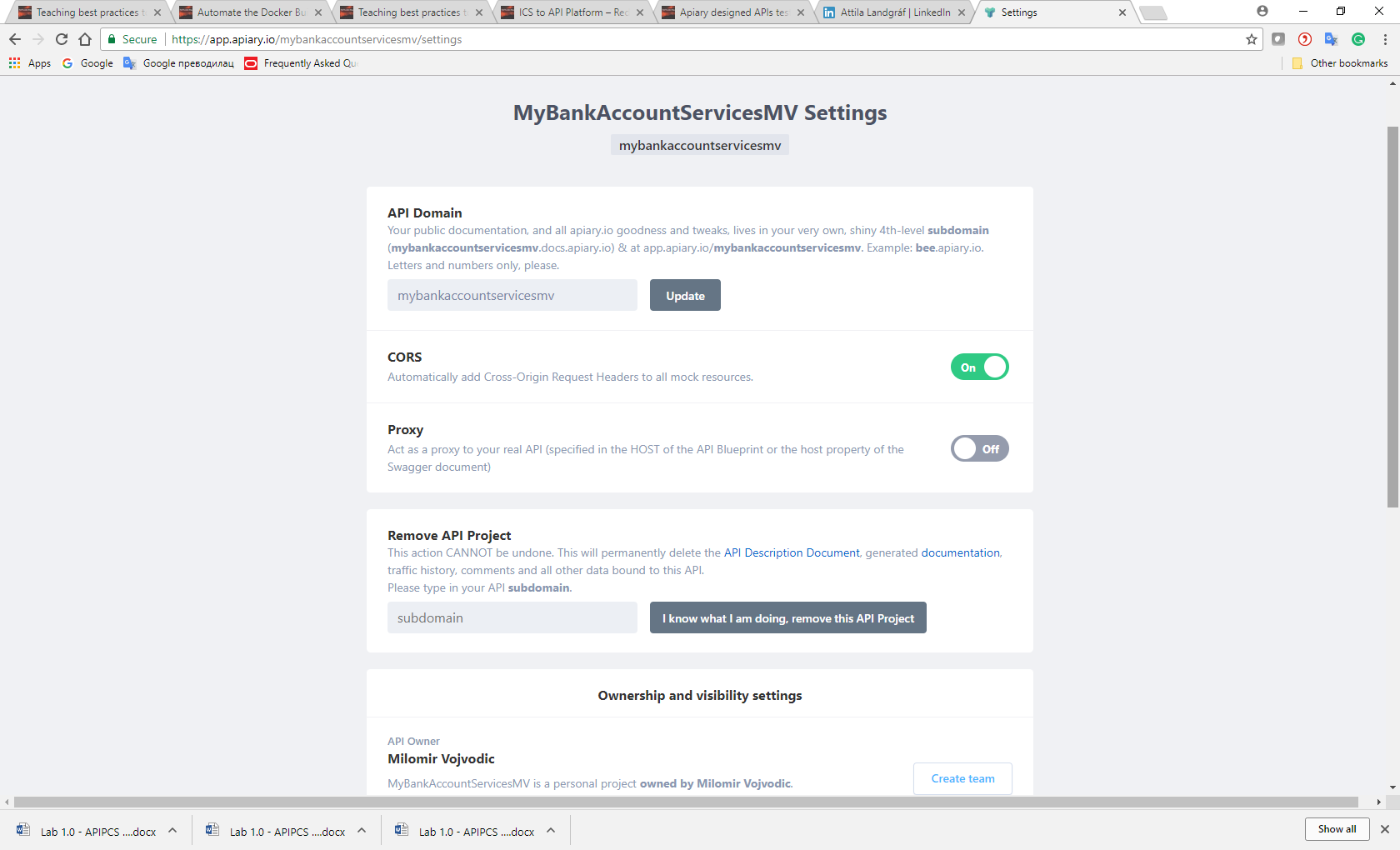
1. You will now see your API project opened in the Apiary Editor with the code editor and documentation preview windows.
2. Click on Settings and review that you can

-see that you can API Domain Your public documentation, and all apiary.io have their subdomain(mybankaccountservicesmv.docs.apiary.io)

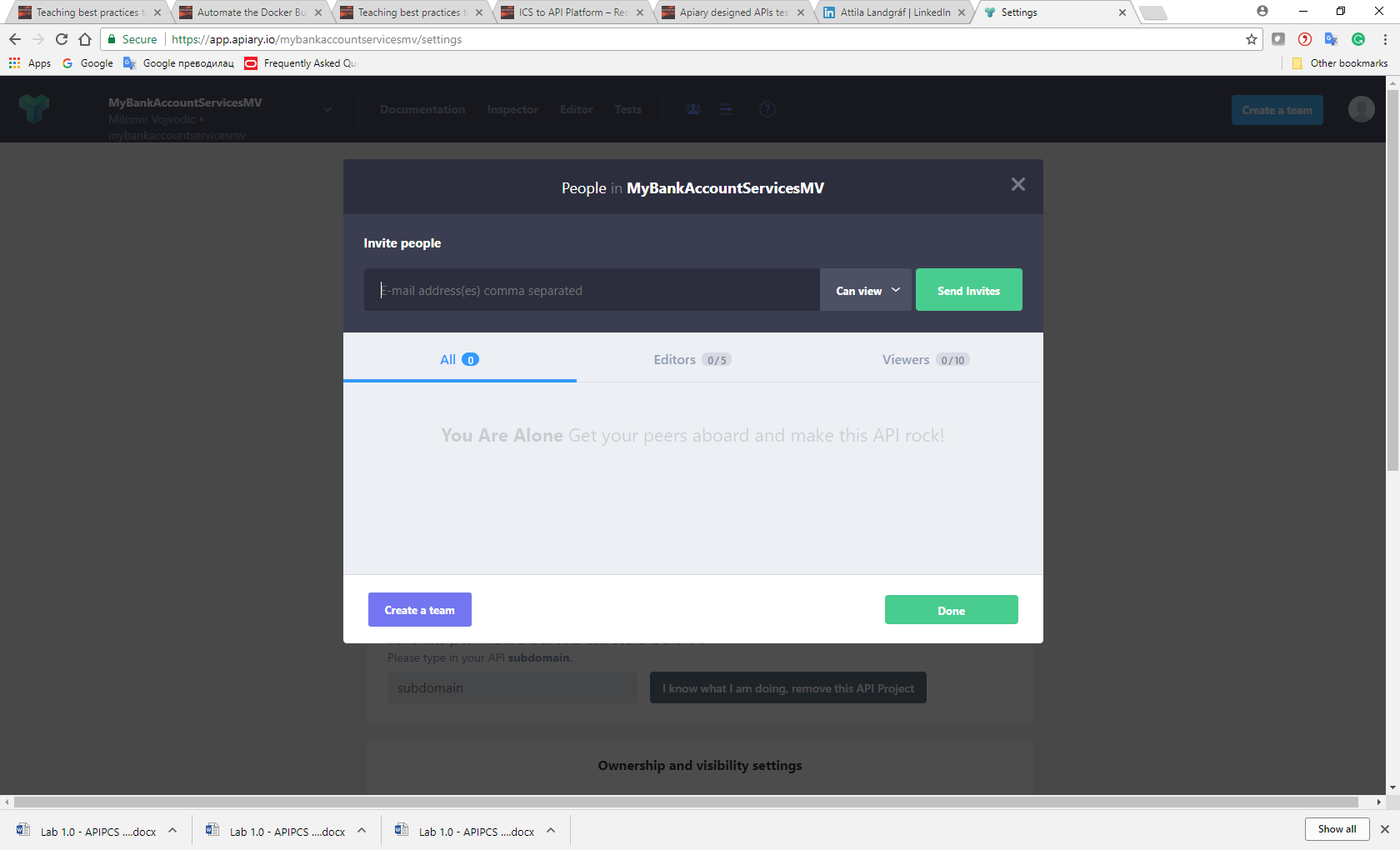
-there is CORS (Cross-Origin Request Headers) to all mock resources-

-there is Proxy (Act as a proxy to your real API (specified in the HOST of the API Blueprint or the host property of the Swagger document))

–you can remove API Project



1. Go to People and see that you can invite people and provide them ‘can view’ and ‘can edit’ roles



## Edit your API definition

We have selected to use the [API Blueprint](https://apiblueprint.org/) format, which is a open source project, to describe the API. This format and the Apiary editor will use the metadata and resource description to document and build the mock service of your API.

The API Blueprint uses levels to document your API, the first level uses one '#' which will become the API Name. For example:

# Polls

A headings section is used for Resources with two '##', a description and URI. For example:

## Question Collection [/questions]

Subheadings are used for actions with three '###', a description and HTTP method, followed by usage text and the Response(s)

You can refer to the [URI Template](https://help.apiary.io/api_101/uri-templates/) for additional details on structuring your API.

### Edit the Sample API definition

In the Code Editor

1. Edit the Host: and change the URI
   1. From: **http://polls.apiblueprint.org/**
   2. To: **http://accountservices.apiblueprint.org/**
2. Edit the API resouce
   1. Change the Resourse:
      1. From: ## Questions Collection [/questions]
      2. To: **## My Bank Account Balance [/accountBalance]**
   2. Notice the documentation preview is being updated automatically as you change the design
3. Edit the API description as follows:
   1. From: **Polls is a simple API allowing consumers to view polls and vote in them.**
   2. To: **MyBankAccountServices is a simple API to demonstration the use of Apiary and the concept of API Design First.**
4. Edit the API subheading and Response as follows:
   1. Change the subheading:
      1. From: ### List All Questions [GET]
      2. To: **### Get Bank Account** **[**GET**]**
   2. Change the json of the Response as following:
   3. From:

{

"question": "Favourite programming language?",

"published\_at": "2015-08-05T08:40:51.620Z",

"choices": [

{

"choice": "Shift",

"votes": 2048

}, {

"choice": "Python",

"votes":1024

}, {

"choice": "Objective-C",

"votes": 512

}, {

"choice": "Ruby",

"votes": 256

}

]

}

* 1. To:

{

"accountNr" : "12345678",

"bankName" : "Allied Irish Banks plc.",

"branch" : "Artane Roundabout, Dublin 5",

"custFirstName" : "YourFirstName",

"custLastName" : "YourLastName",

"custEmail" : "YourEmailAddress",

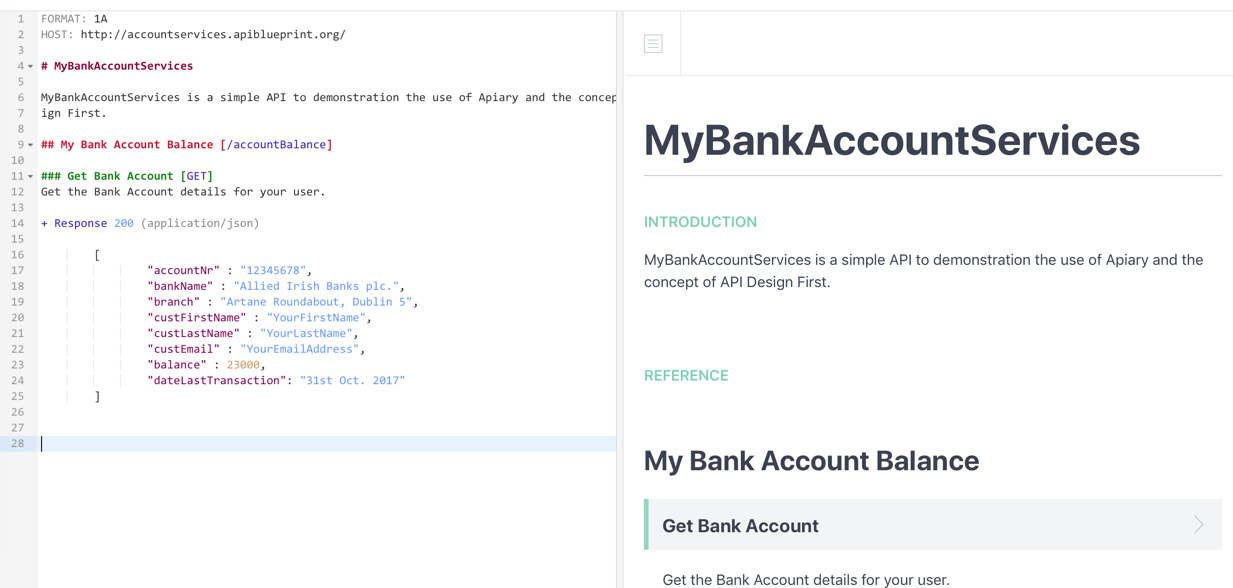
"balance" : 23000,

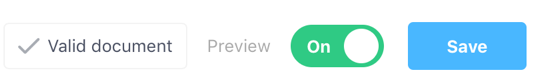
"dateLastTransaction": "31st Oct. 2017"

}

You may copy and paste this section but don’t forget to update the custFirstName, custLastName, and custEmail values with your information.

1. Delete the remaining Resource and Response sections. Your API code and documentation should look as follows, without any Warnings or Errors from the validator





1. Click on Save

# Apiary Interactive Documentation

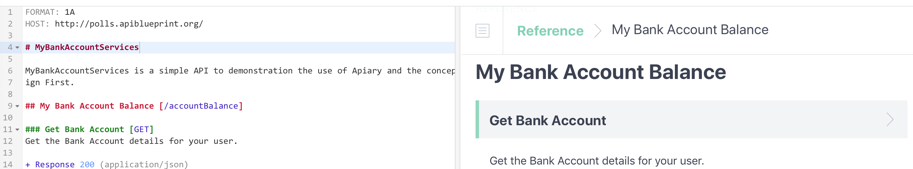
As you have seen the is generated automatically as the API is being designed based on the tagging of sections and the provided descriptions.

The interactive documentation contains two main columns: the human and machine columns. These two columns provide the separation that is important for reading the documentation and actually using tooling to interact with it.

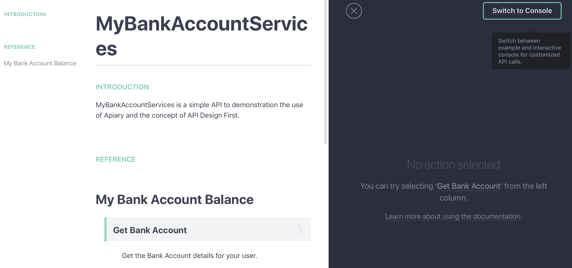
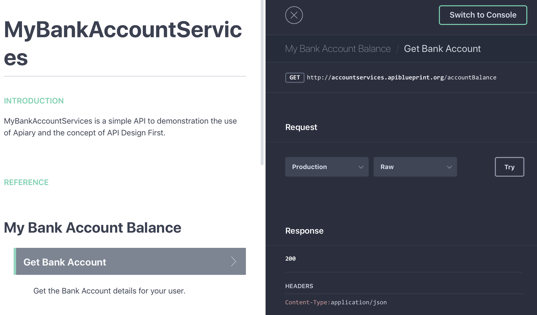
The Human Column provides visibility into the long-form description of your API. When you write your API Description, the goal is to write it as you would any other document you mean for other humans to read and consume.

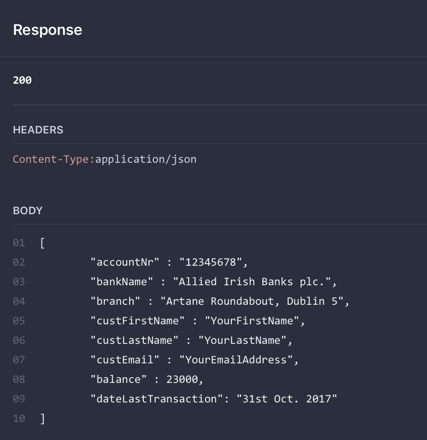
The Machine Column displays the information that clients and servers will be interested in when interacting with your API or Apiary’s mock and debugging servers.

## Update the API design with additional documentation

1. Update the API design and add a description for the Get Bank Account resource [GET] action
2. Type the following under the [GET ] action
   1. Get the Bank Account details for your user.
3. Preview the updated documentation
4. Click on Save

## Review the API Documentation

1. Navigate to the API Documentation
   1. Click on Documentation from the menu
   2. If shown, click leave page
2. Review the Introduction and Reference sections in the Human Column
3. Select the Get Bank Account action this will update the Machine column with the actual Resource, Request and Response for the API.
   1. Scroll down to preview the Response

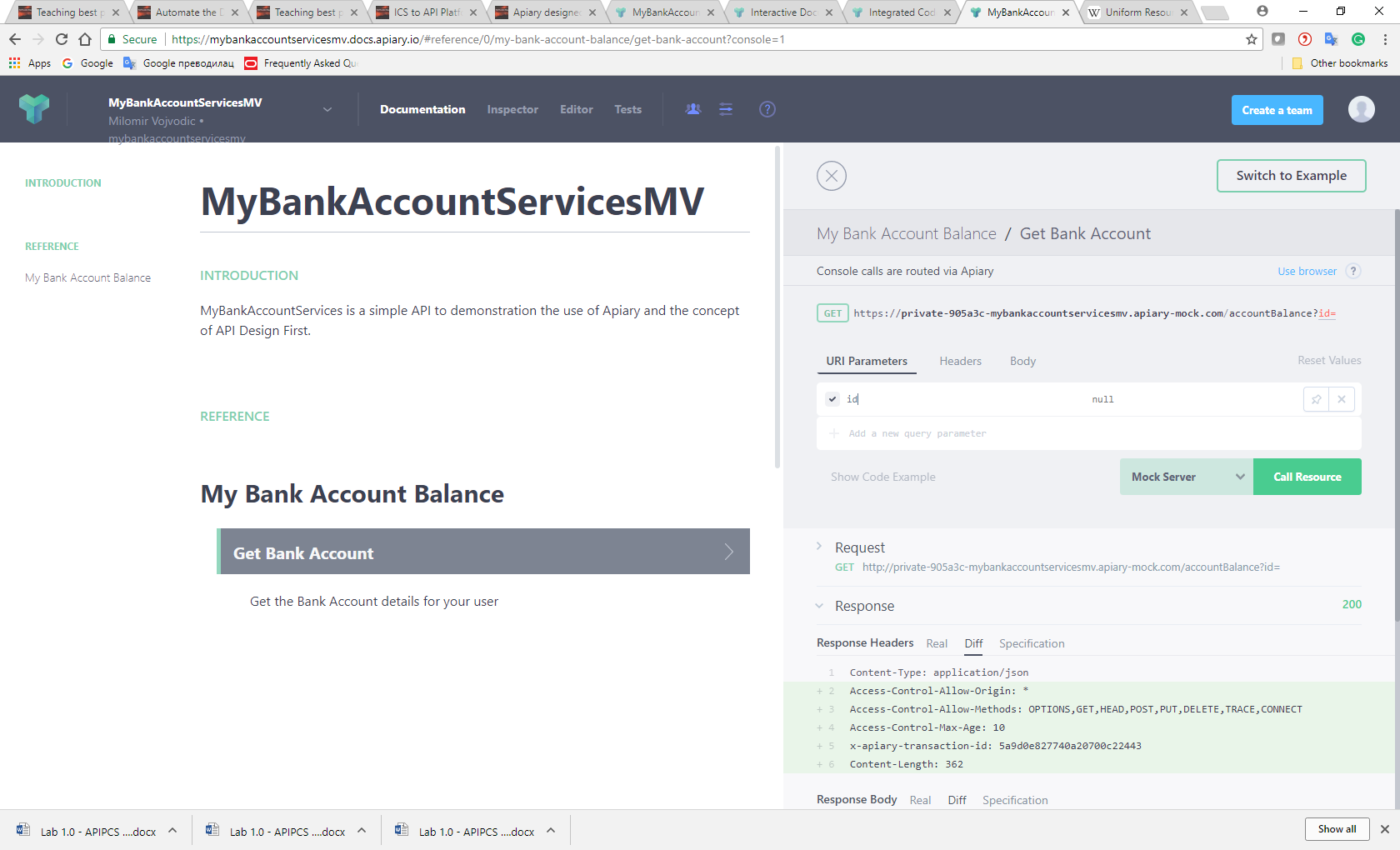


1. Review that you can set header and parameters in console view

The console in the machine column is used for creating requests to your production server, mock server, and debugging server. You can get to this by clicking the “Switch to Console” link in the machine column.

Headers -The REST headers and parameters contain a wealth of information that can help you track down issues when you encounter them. HTTP Headers are an important part of the API request and response as they represent the meta-data associated with the API request and response.

Parameters-REST parameters specify the variable parts of your resources: the data that you are working with.



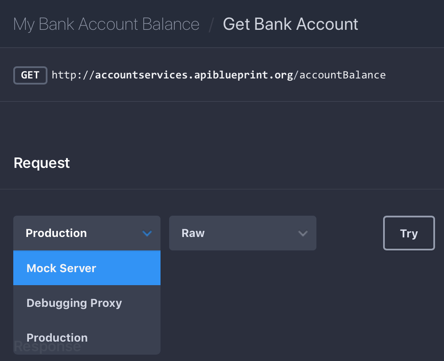
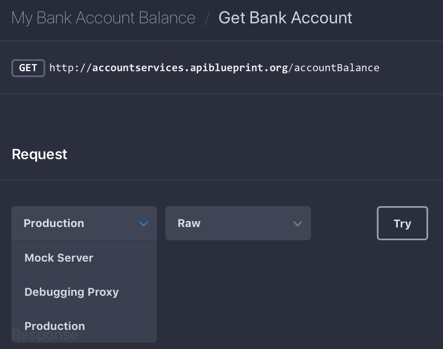
1. Click on <https://pollsapi.docs.apiary.io/#reference/question/question/view-a-question-detail?console=1>
2. Review click on ‘View on question detail’
3. Review click on ‘View on a Choice'

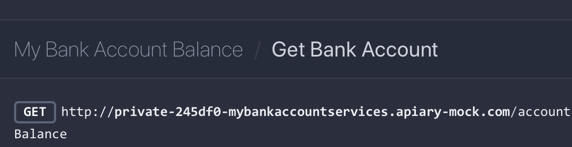
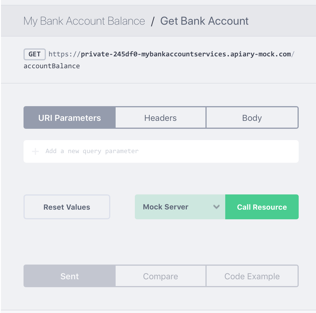
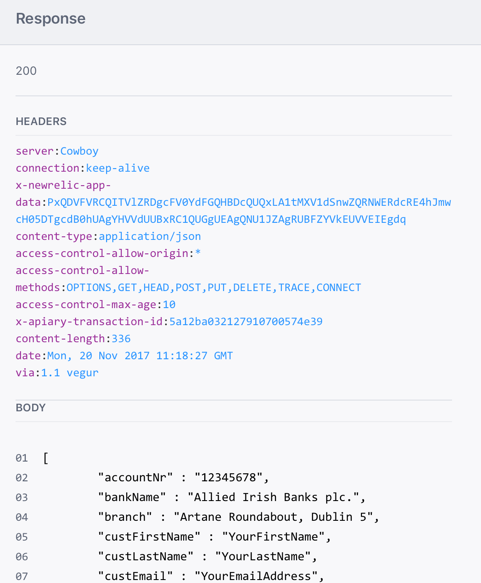
# Apiary Mock Server

In the design phase of an API, it is helpful to experiment with how you feel your API should work and iterate on this process quickly. In the past, this involved writing code as a prototype and setting up a server to run it. This takes time and directly effects the creativity behind the design process.

The Mock Server allows you to try out your API as you design it, giving immediate feedback along the way in how it may be used.

## Using the Mock Server to Test your API

The Mock Server is automatically created each time you publish your API Description. This means the only thing you have to do to get started is to write your API Description, include specific requests and responses for your resources, and click publish.

1. From the Documentation menu, ensure your request action is selected.
2. Change the Request from Production to Mock Server
3. Notice that the request URI has changed to the Mock server
4. Click Try
5. This will change from the Example to the Console View where you may test the API. As this API does not require any input parameters, we can simple invoke it without any parameters.
6. Click **Call Resource**
7. The Mock Server will return the first response from the API
   1. Scroll down to review the response

## Access the Mock Service directly

You may invoke the Mock Service directly in your browser as well by using the API URI for the Mock Server and your API service resource.

1. Copy the service URL for the GET http request:
   1. Sample - https://private-245df0-mybankaccountservices.apiary-mock.com/accountBalance



1. Paste this in your Browser and submit the request
2. Review the API response

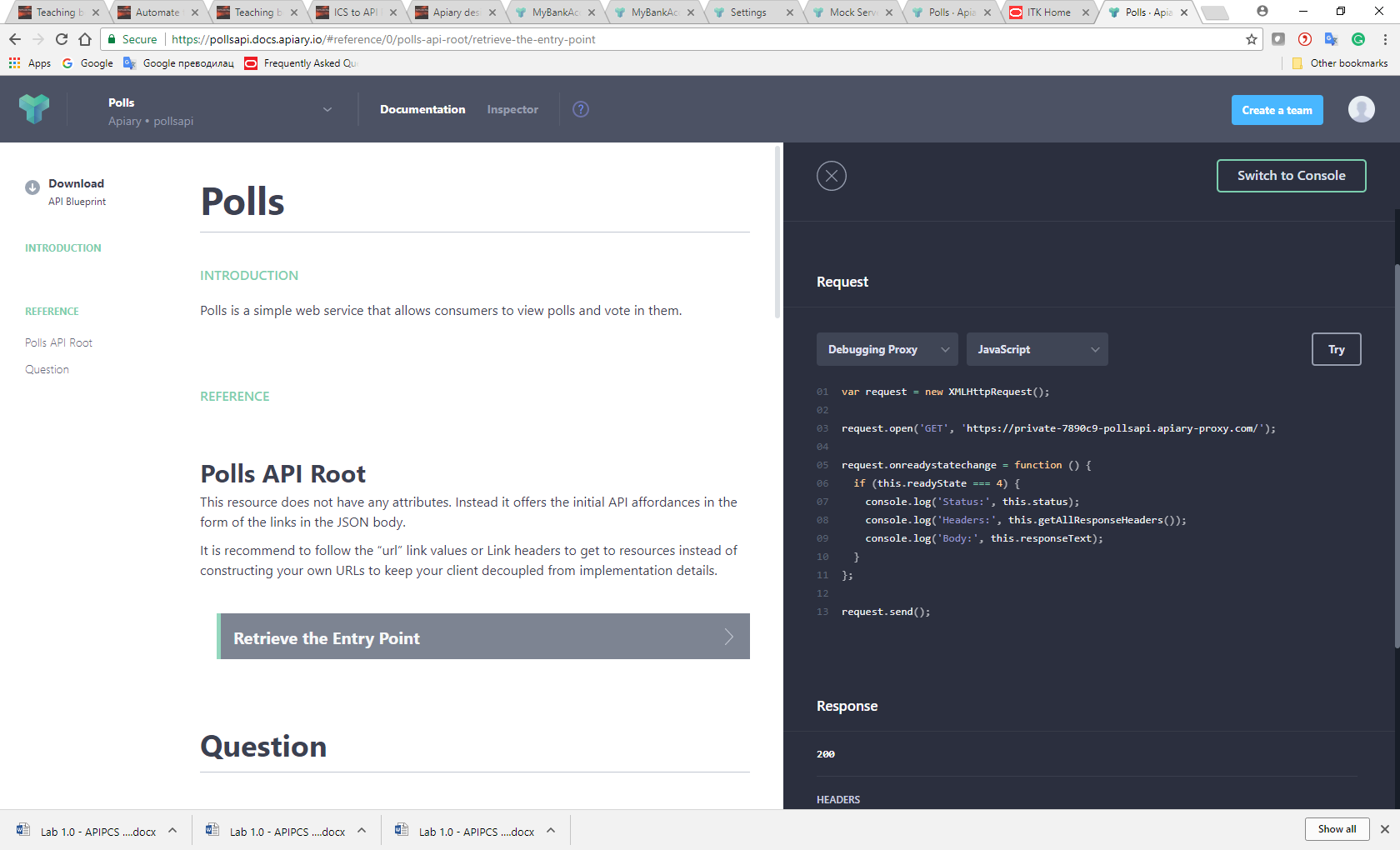
## Integrated Code Examples

We encourage a contract-first approach to designing your API. Write your API Description first before writing any server or client code.

To make this process possible, Apiary provides tools that allow you to try out your API as you design it. Code examples are one tool provided that can be dropped into your existing code or used for prototyping.

Apiary has auto-generated tools that can create API clients in dozens of programming languages, including Python, PHP, Nodejs, Java, Go, C#, etc.

Finding and Using Code Examples-You can find the code examples in the machine column of the documentation. Below is a screenshot of the JavaScript code example you would see in the Polls API documentation.



1. Click on <https://pollsapi.docs.apiary.io/#reference/question/question/view-a-question-detail?console=1>
2. Click on ‘Retreive the Entry Point’
3. Click on ‘Debuggting Proxy and JavaScript’ in request

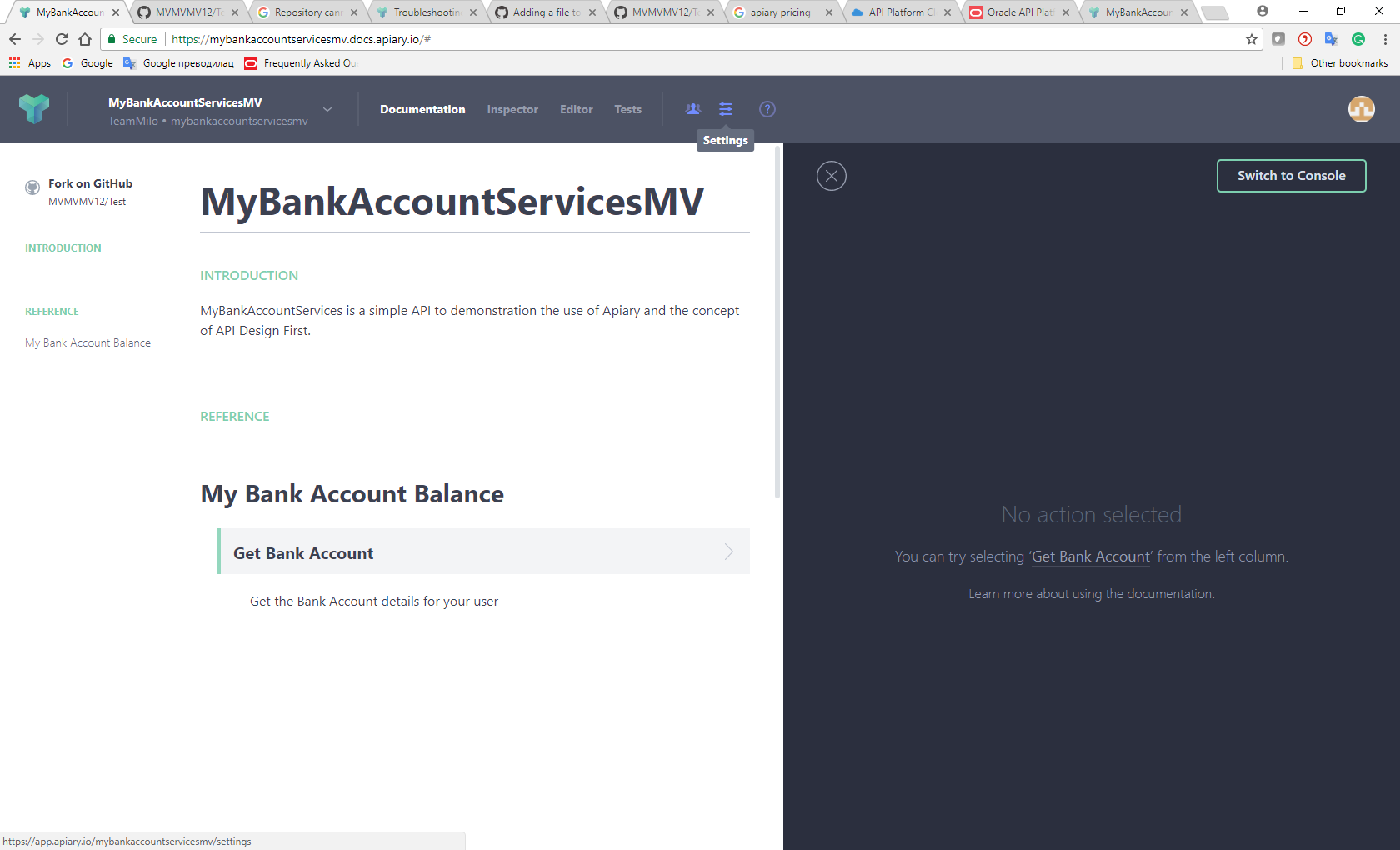
Contributing Code Examples-Our code examples are open source, and you can contribute by improving our existing examples or adding new ones. You can find these examples templates in this GitHub repository <https://github.com/apiaryio/language-templates>

# Collaborating Via Documentation Share

GitHub Integration is third way of sharing APIs (first one is domain URL, second one is inviting team members

## Sharing Documentation Via URL

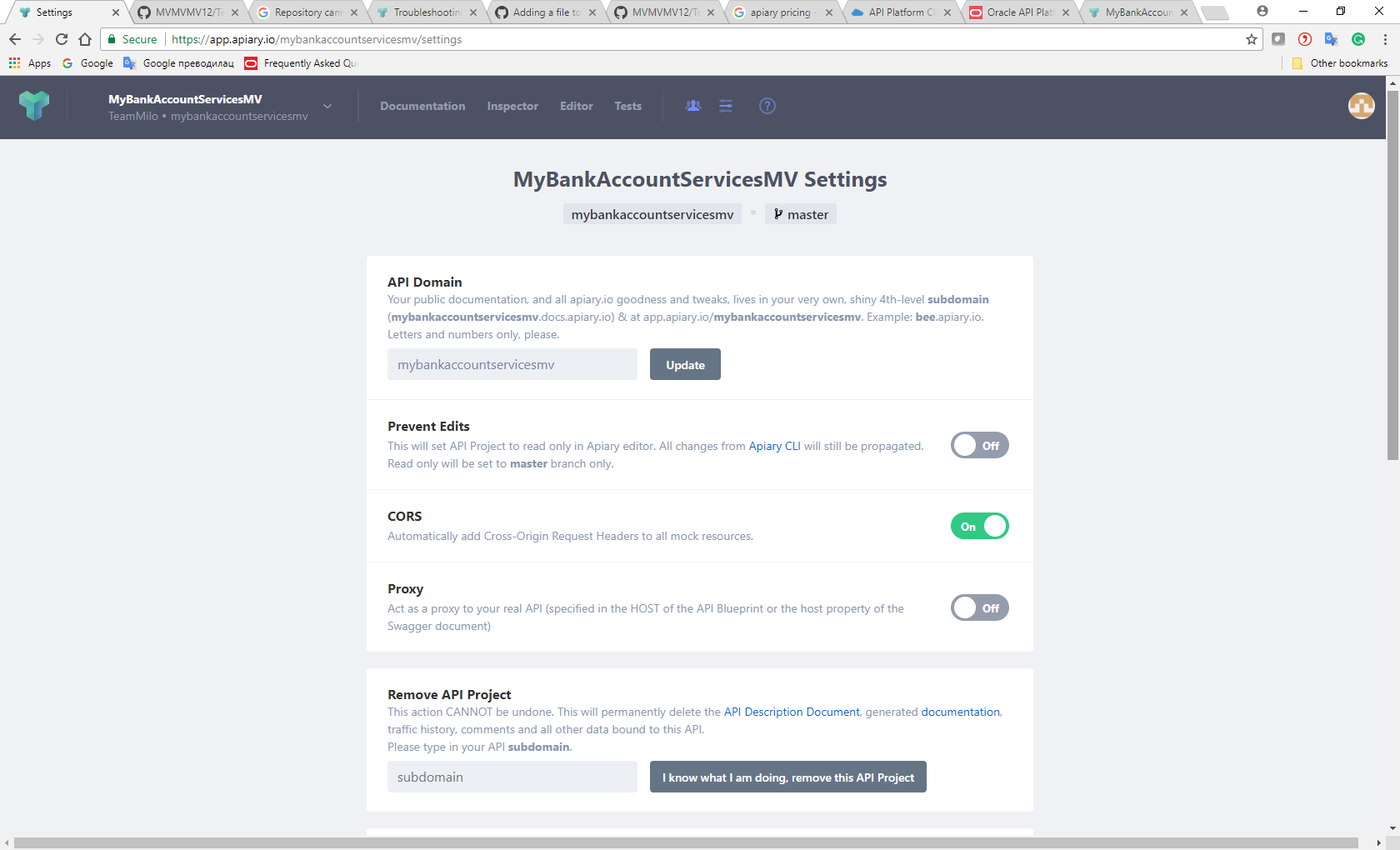
1. Click on the Settings



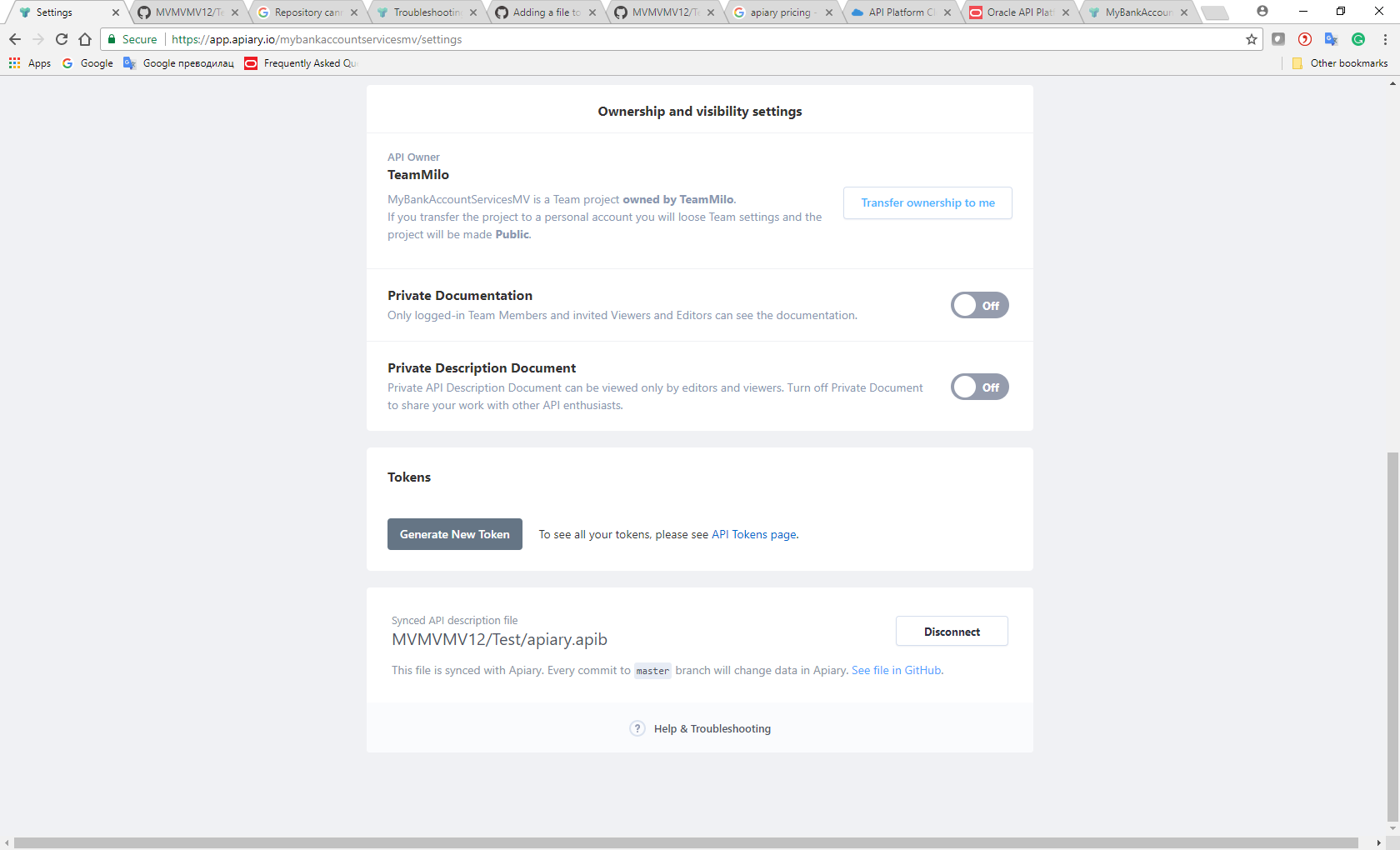
1. Review API domain (URL that you can copy paste and share with everyone who do not even have Apiary)

Copy-Paste URL in another browser to check

1. Review Preventing Edits (On/Off)
2. Review CORS (On/Off)
3. Revew Proxy (On/Off)
4. Review Removing API Project

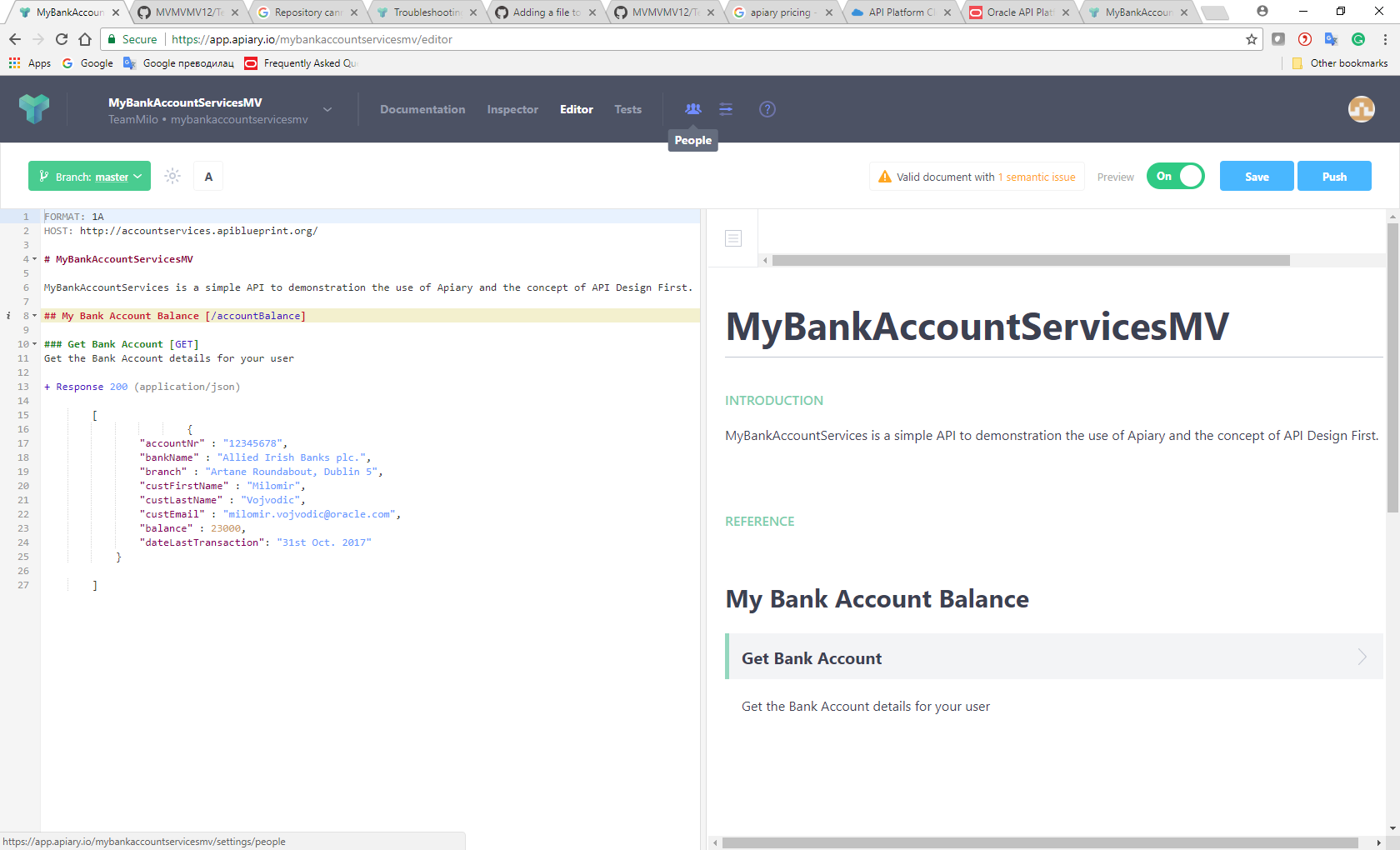


1. Review Request for Transfering Ownernship
2. Review Private Documentation (On/Off)

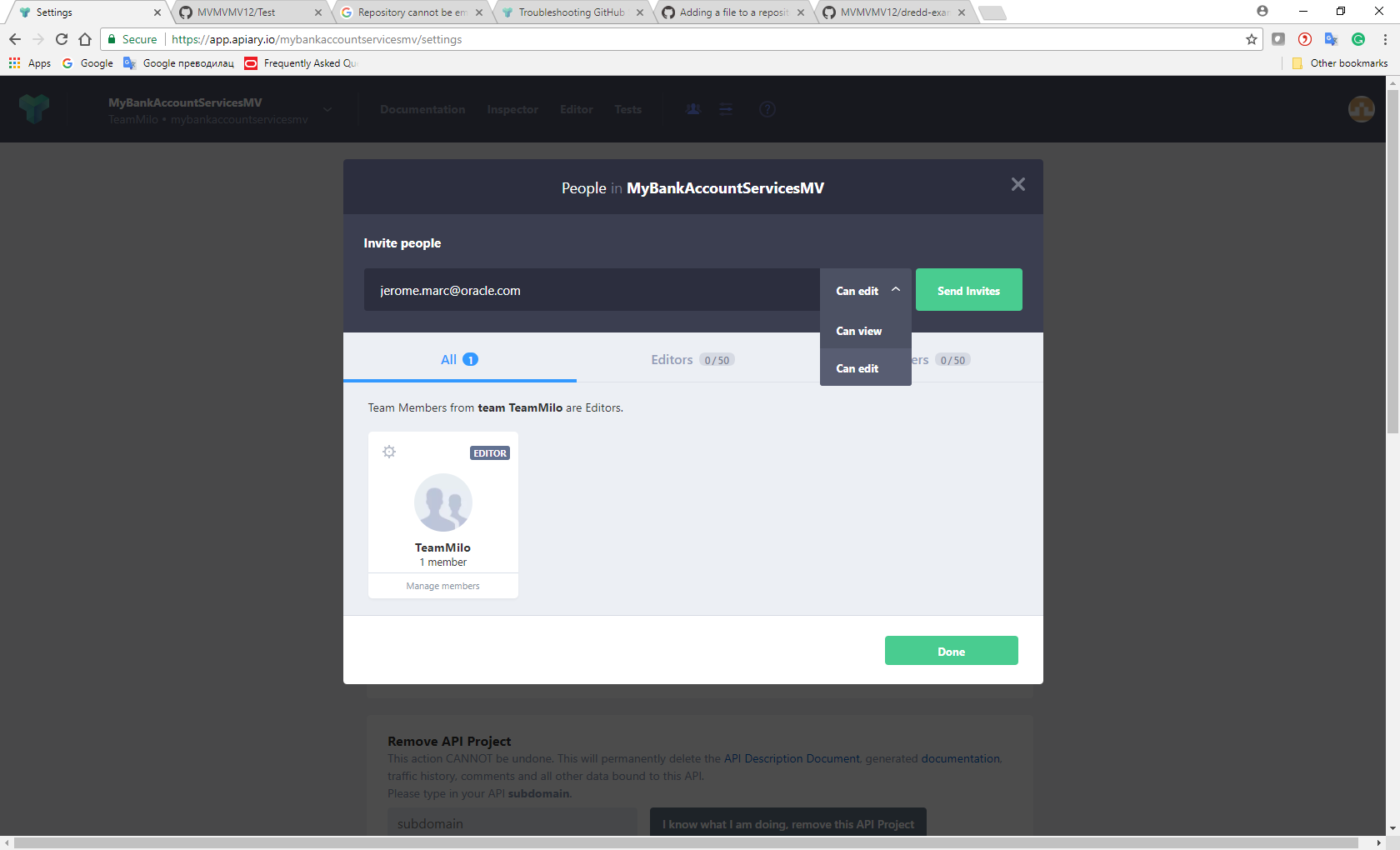


## Sharing Documentation Via Adding Team Members That Can Edit/View

1. Click on People Tab

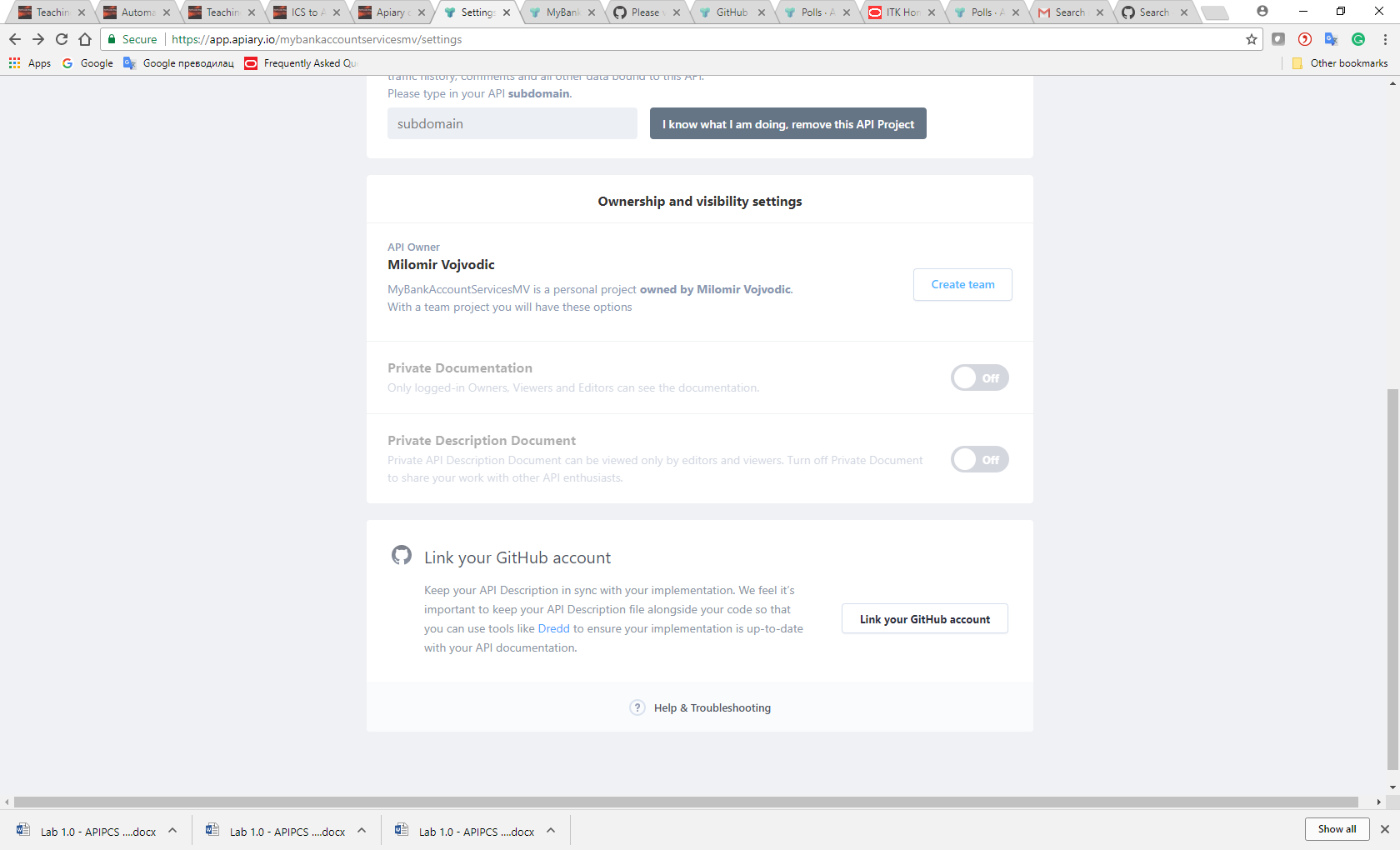


1. Review Inviting People and Can View/Edit roles

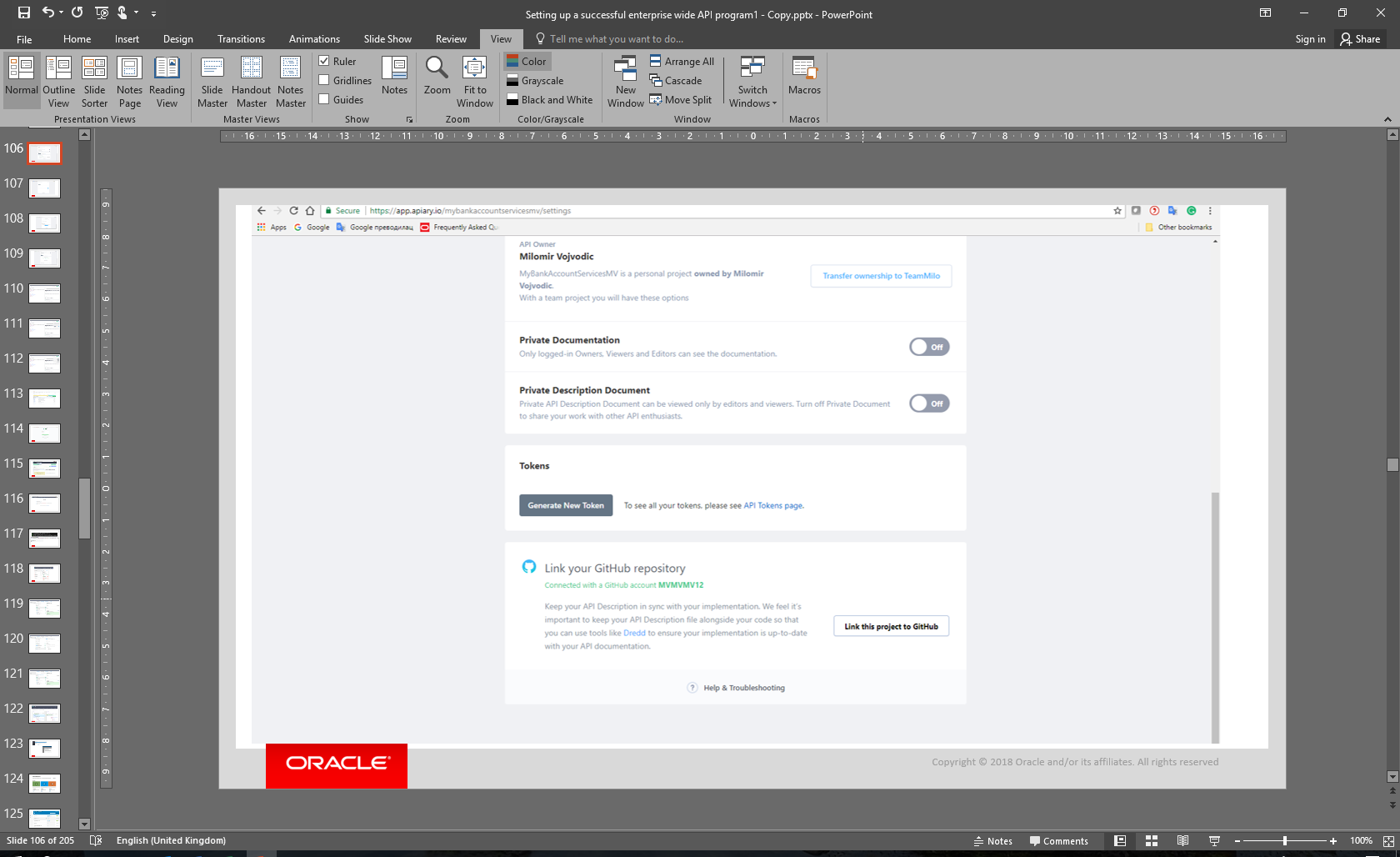


## Sharing Documentation Via GitHub Sync

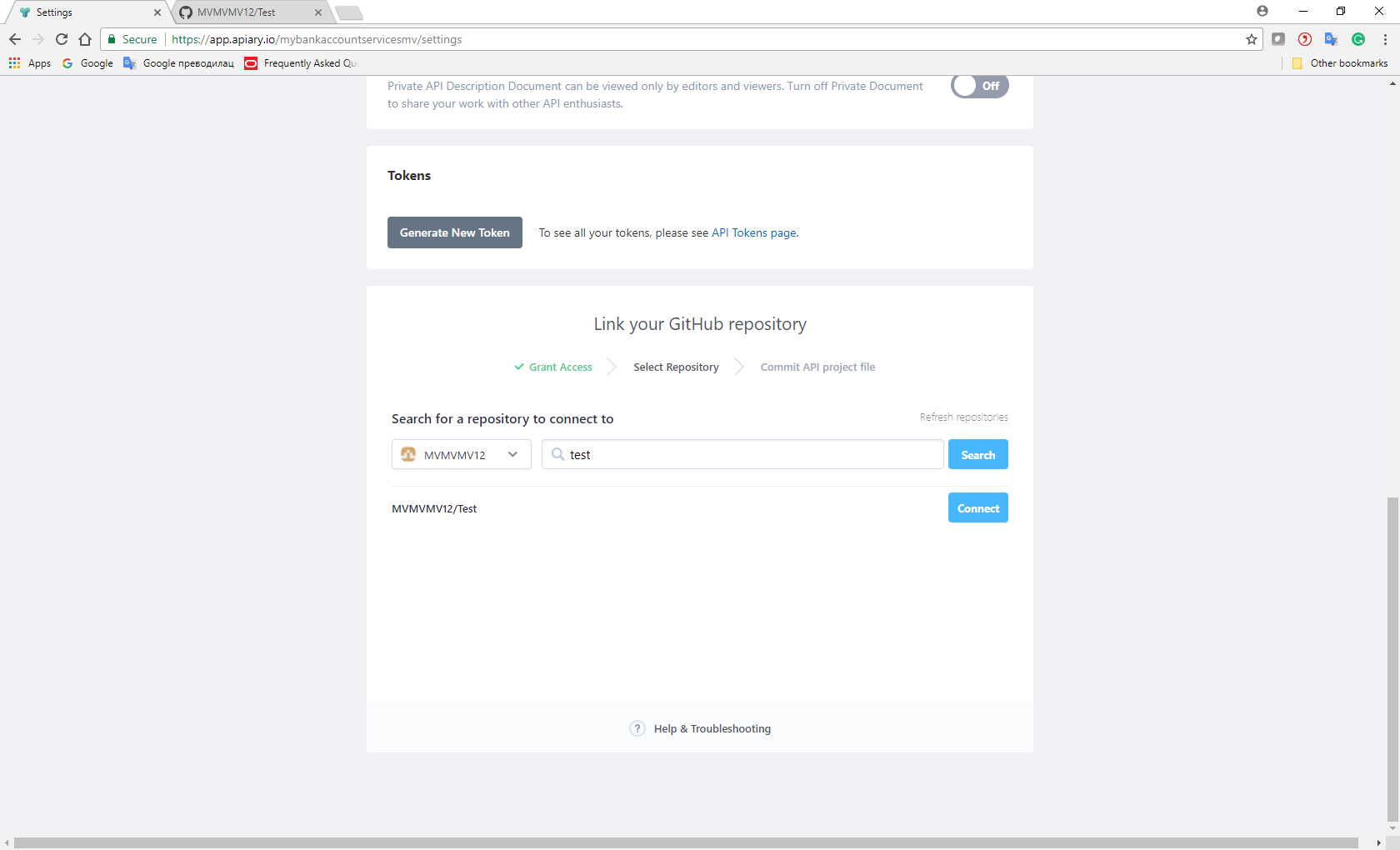
1. Click on Settings
2. Review Link to your GitHub account in the bottom of page



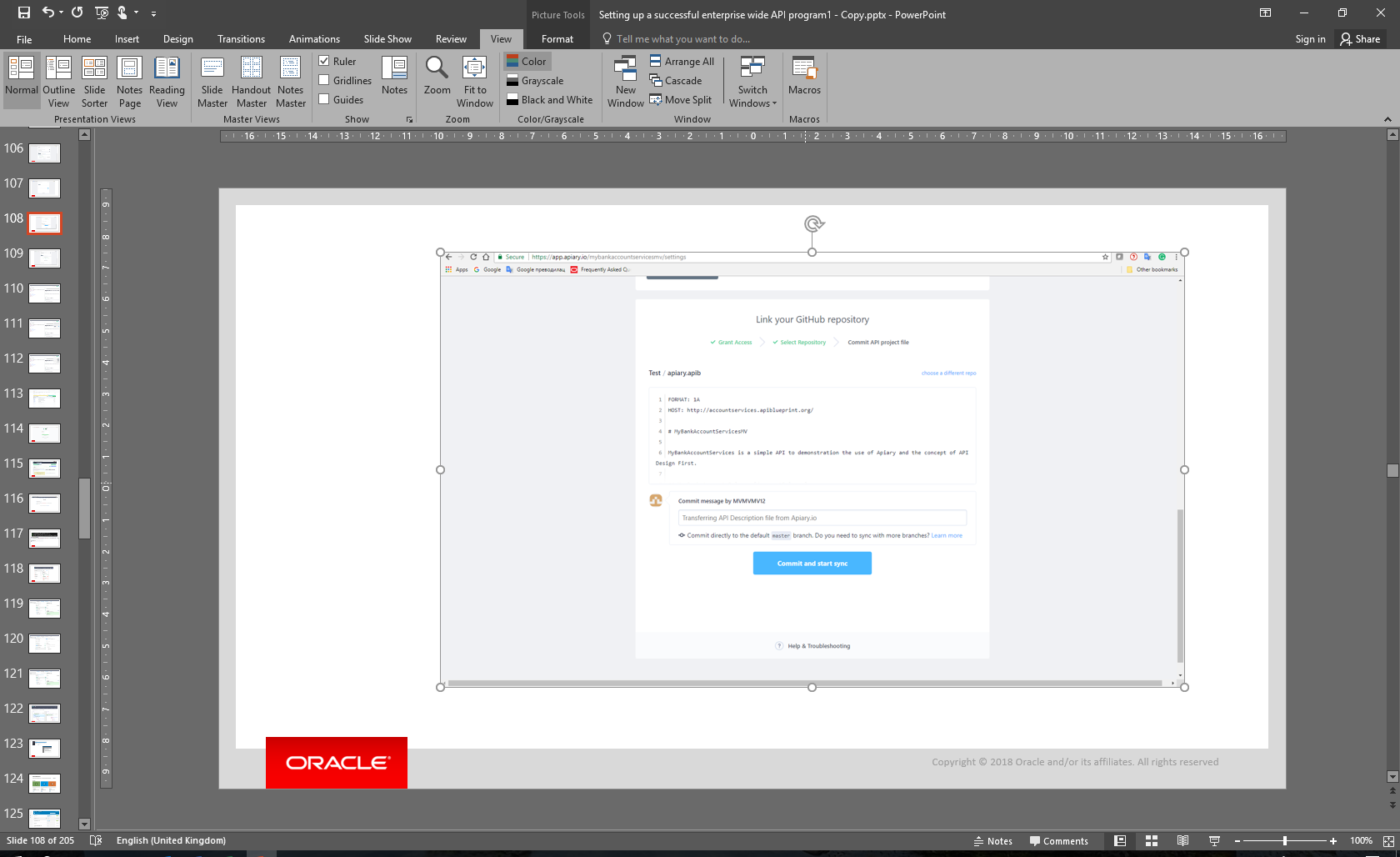
1. Enter your GitHub account and click Link Project



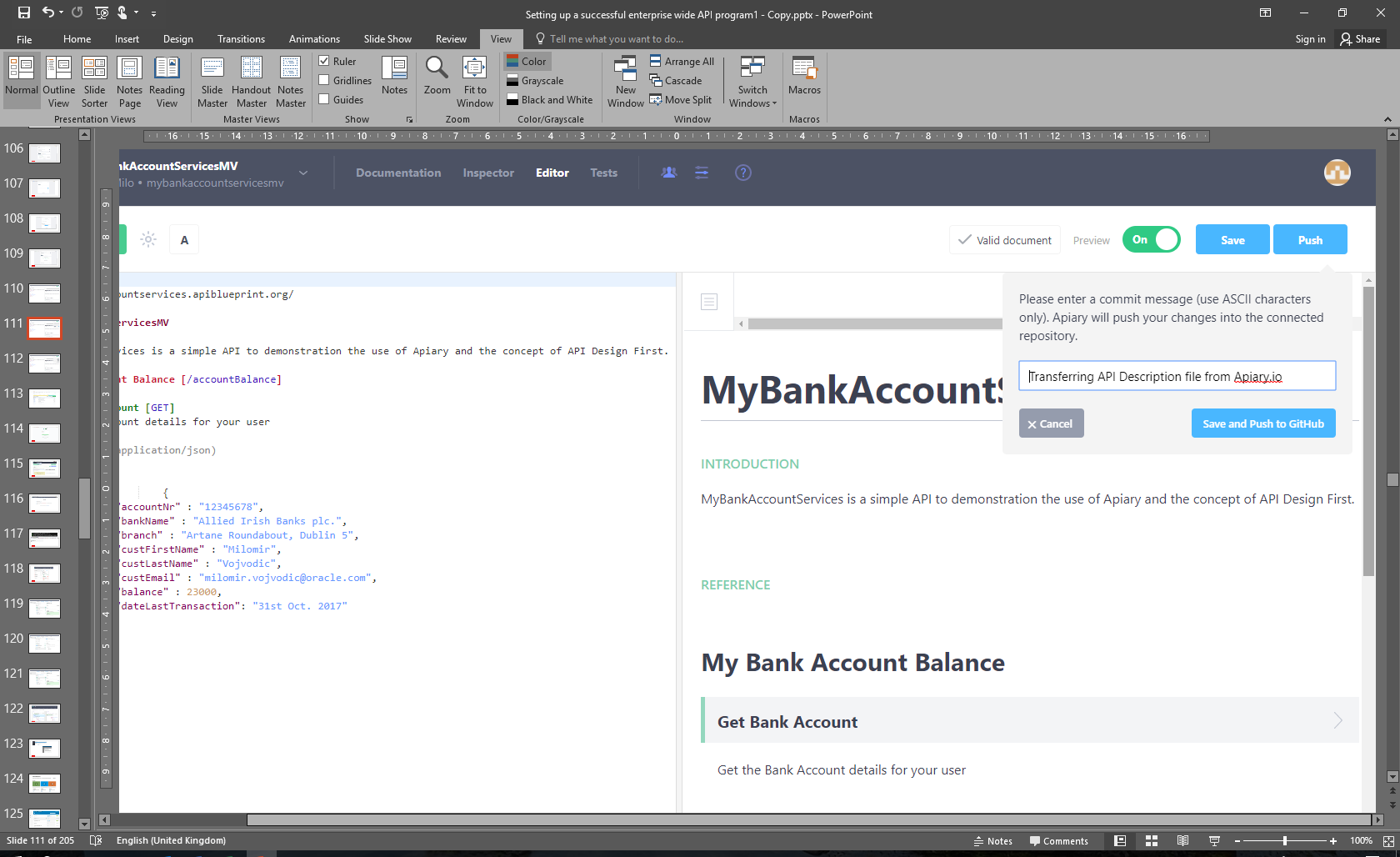
1. Find available repository on your GitHub account and click Connect



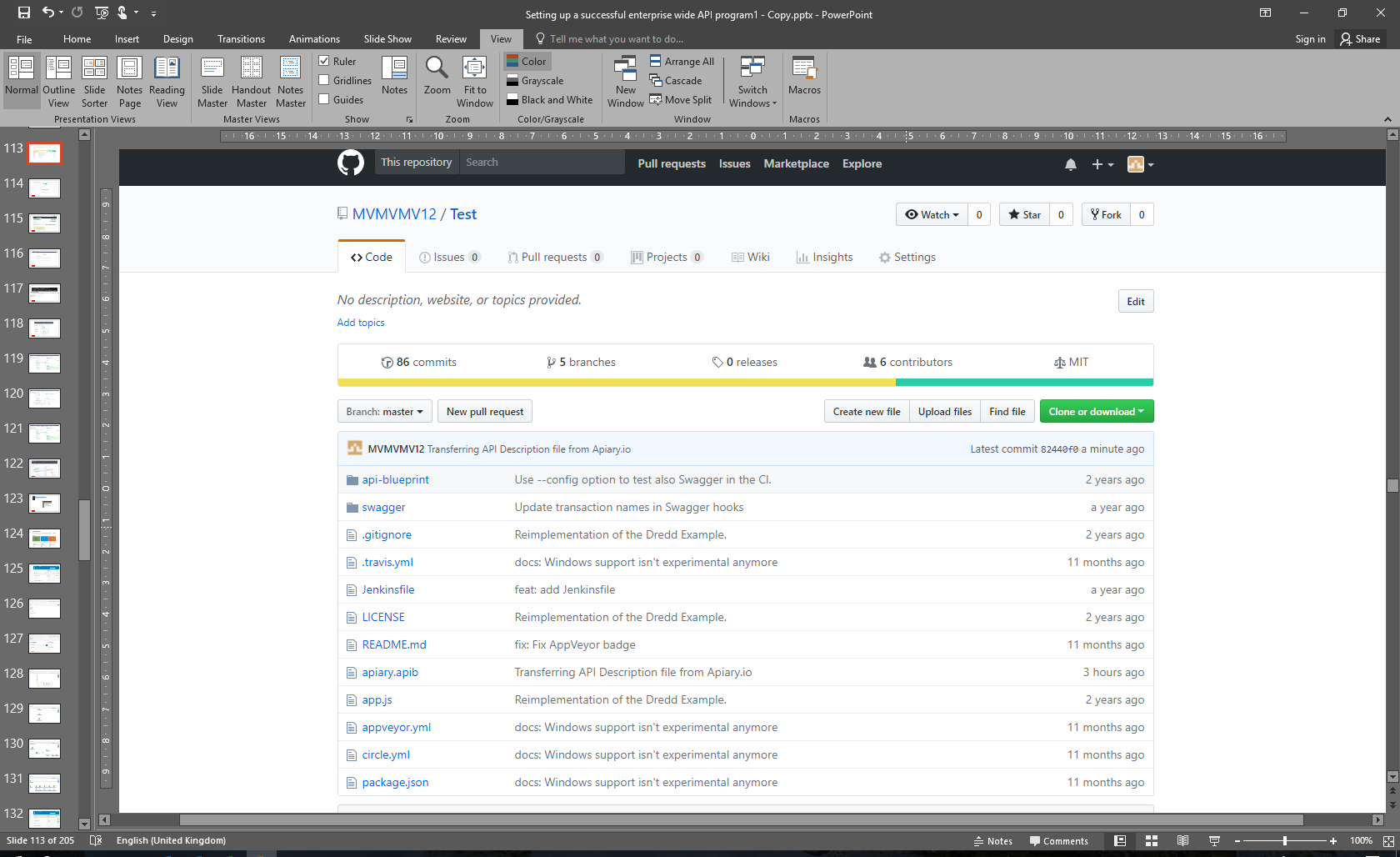
If you don’t have available repository and you create new repository, pls note that repository can not be empty and needs to be initialized (please initialize it with at least 1 file) details on <https://help.github.com/articles/create-a-repo/>



1. Click Connect and Start syncronizing
2. Go back to the Editor Page and notice Push button on the top right corner and press it

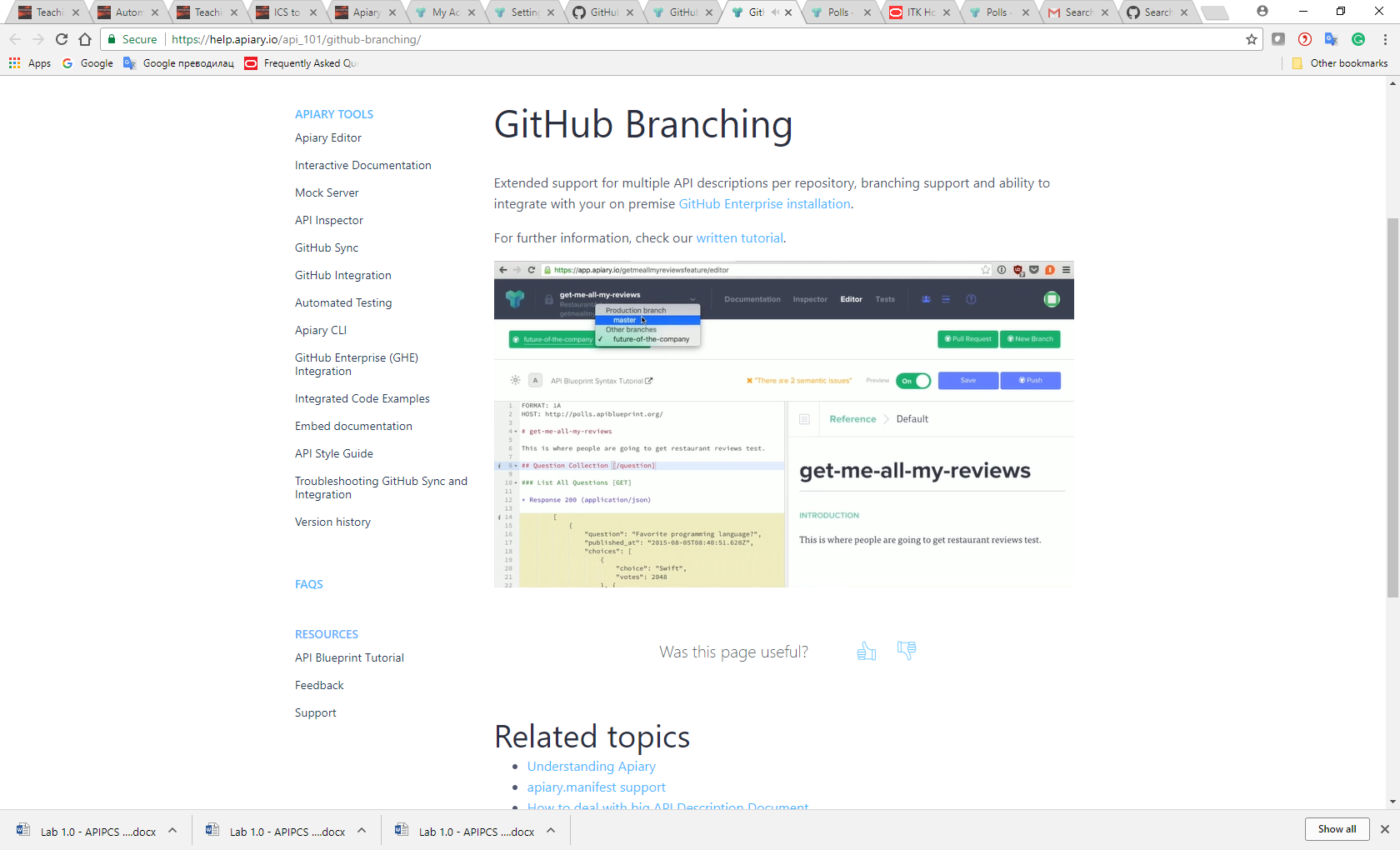


1. Click on Save and Push to GitHub
2. Go on GitHub and notice the Sync that just happened



1. Go on the Editor page and notice in the top left corner Branch Tab

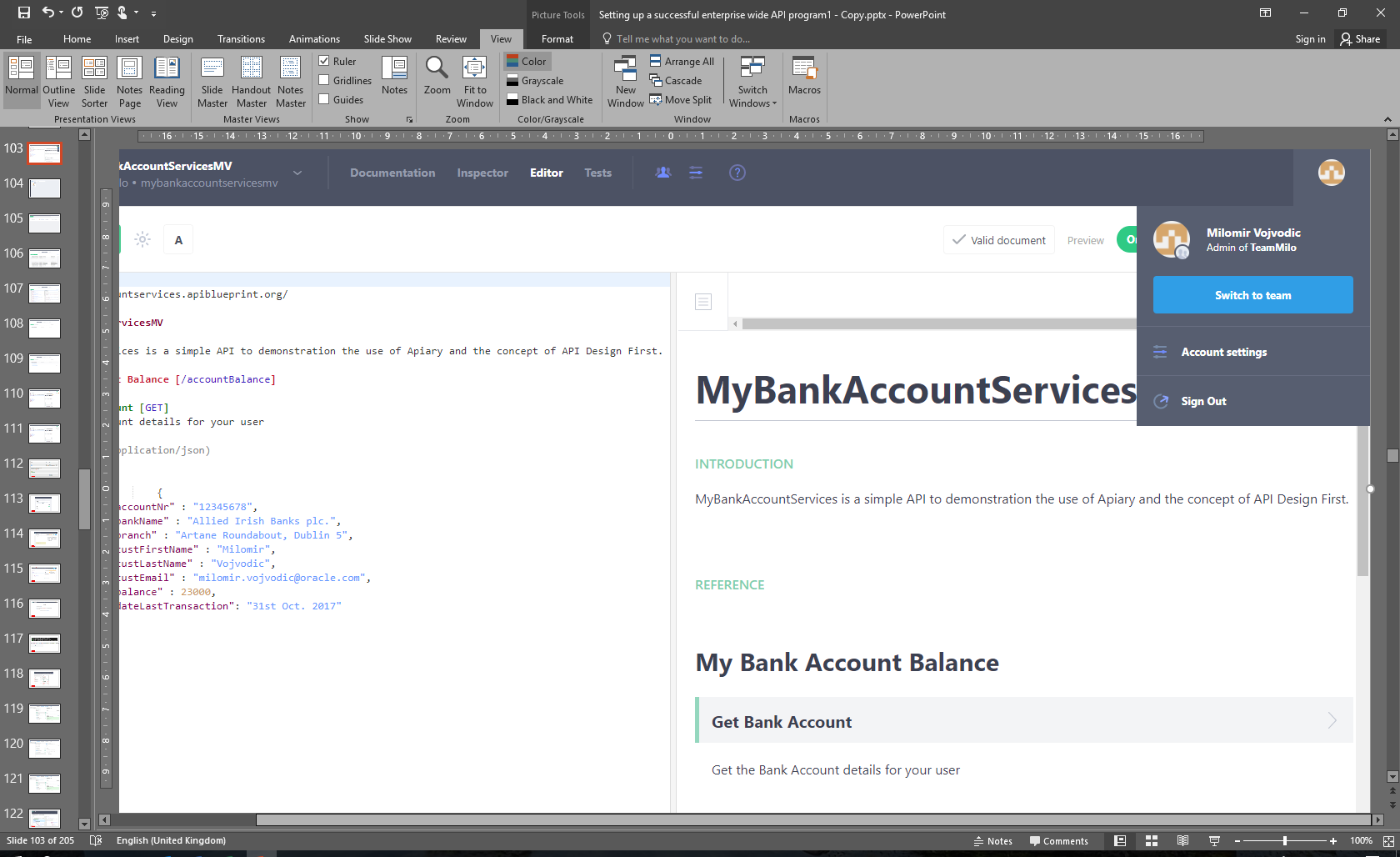
The GitHub Integration feature allows you to create branches and utilize workflows like GitHub Flow for managing your API design process. This feature is available with Apiary Pro. With Free and Standard plans you can sync default branch with GitHub. After this is complete, Apiary will now be pushing to and pulling from that specific branch in your synced repository.



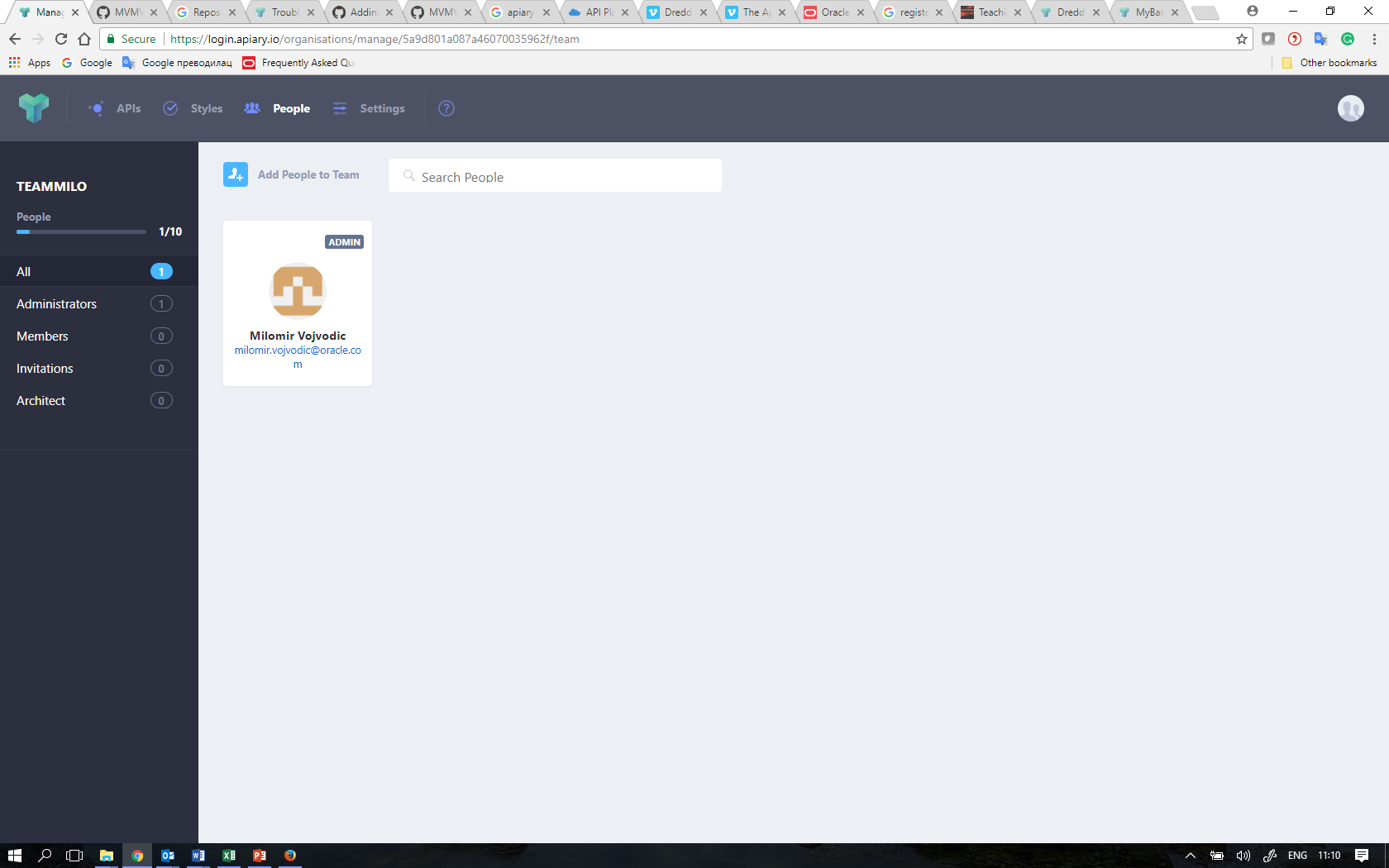
# Team Work And Rules

## Team Overview

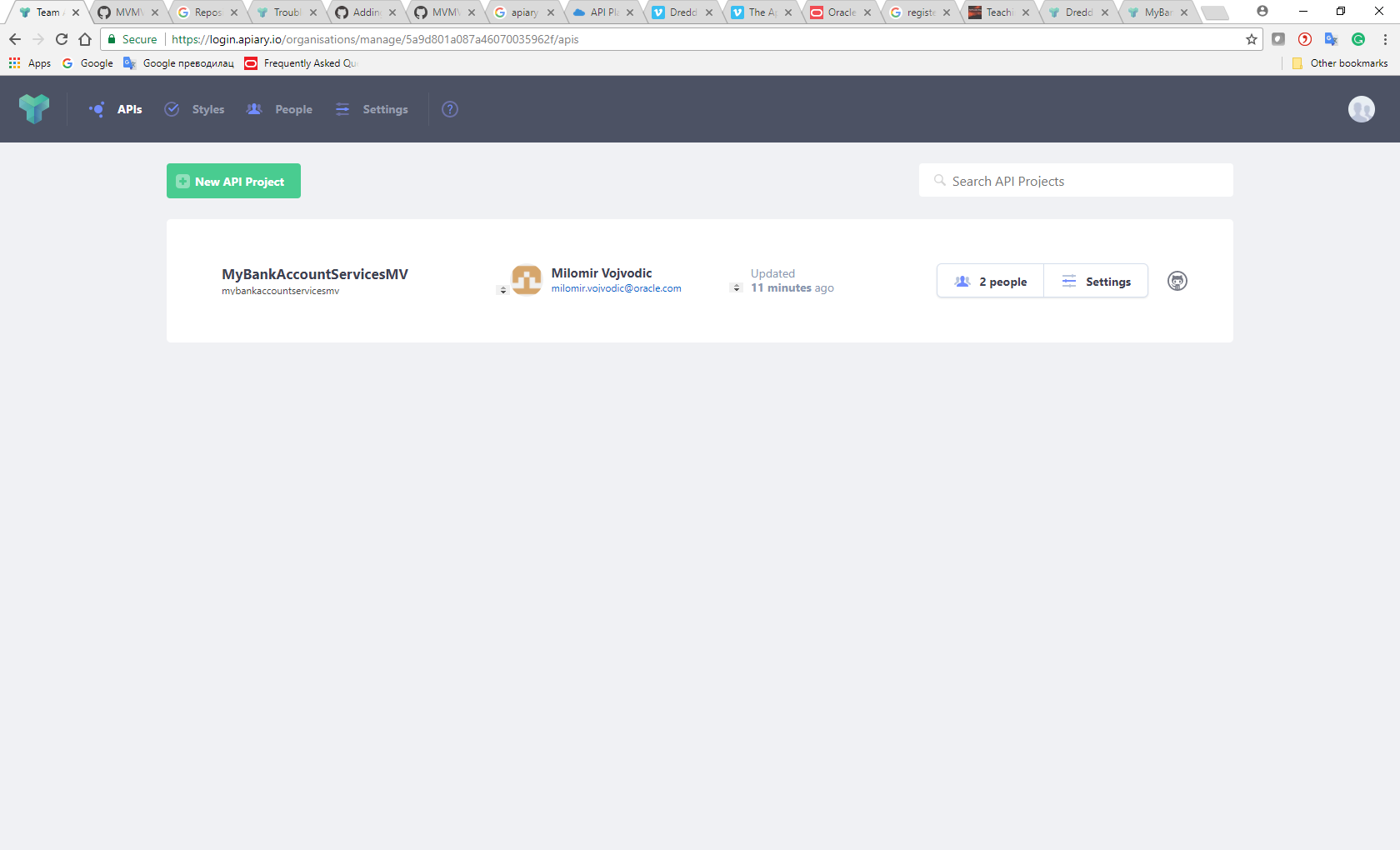
1. Click on the The Icon Top Right Corner And Click Switch to Team



1. Review on the left side All, Administrators, Members, Invitations, Architect roles and Adding People function



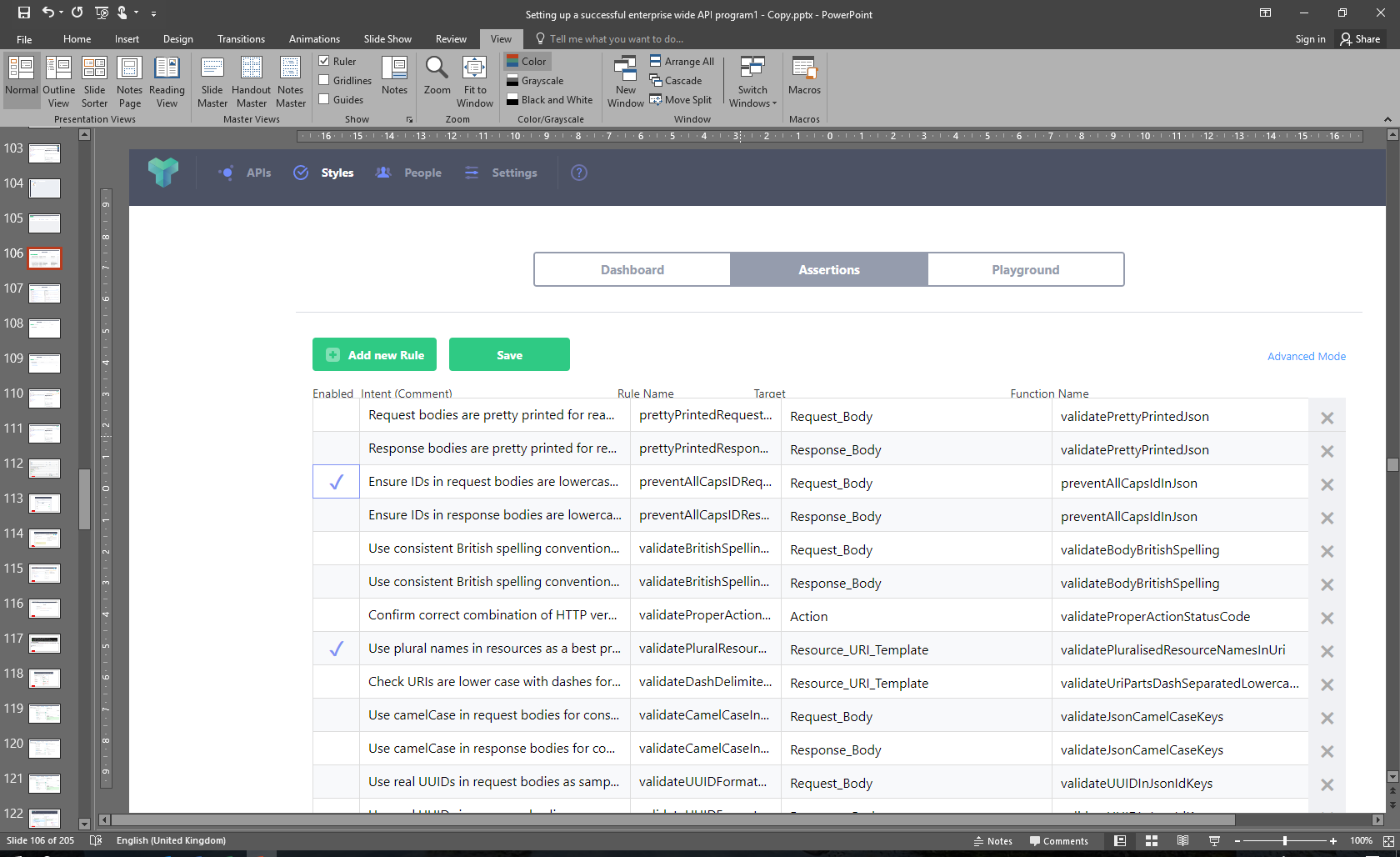
1. Review on the top APIs, Styles, People, Settings Function and click on APIs function



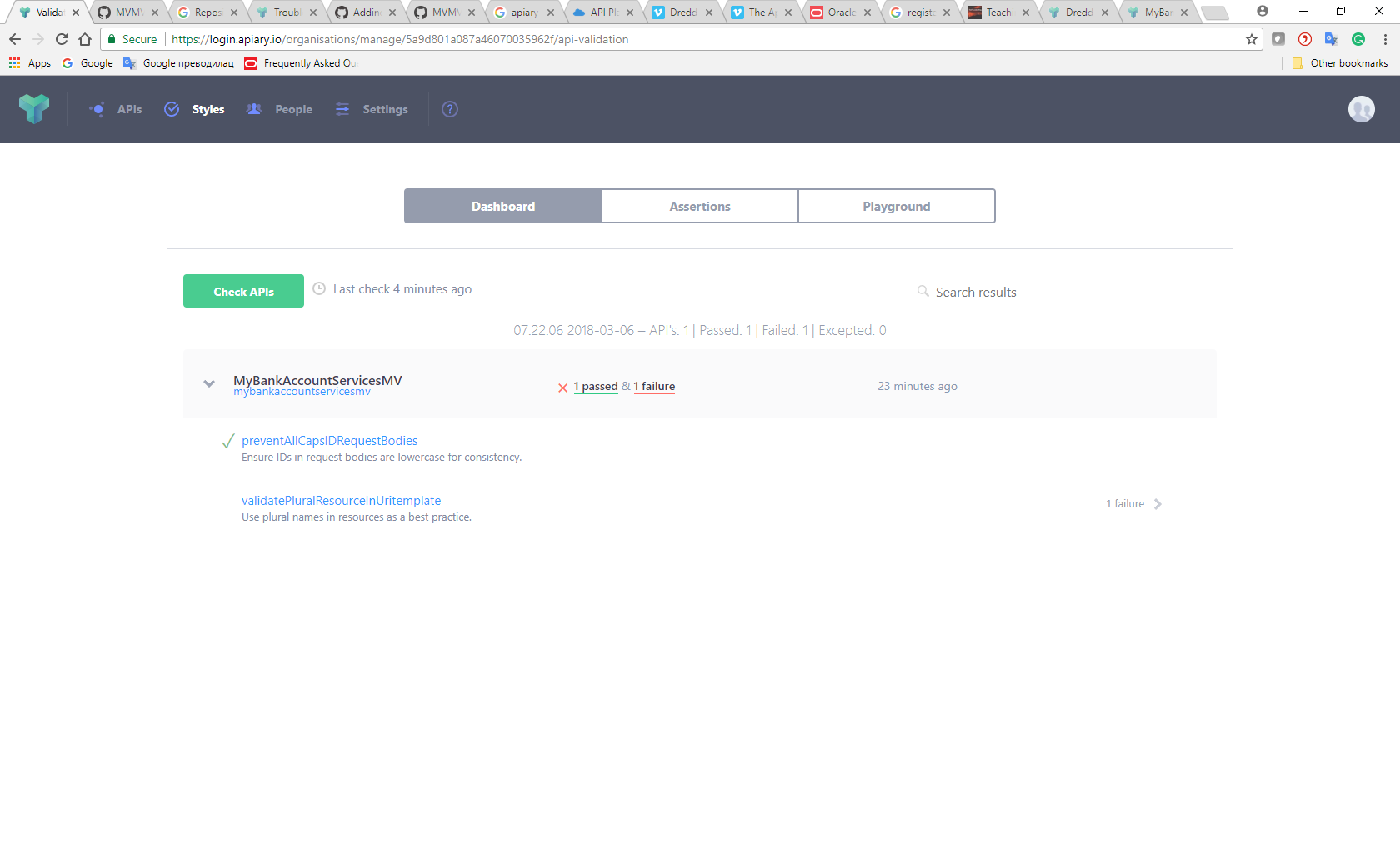
1. Review People and Settings Buttons Within API

## Rules

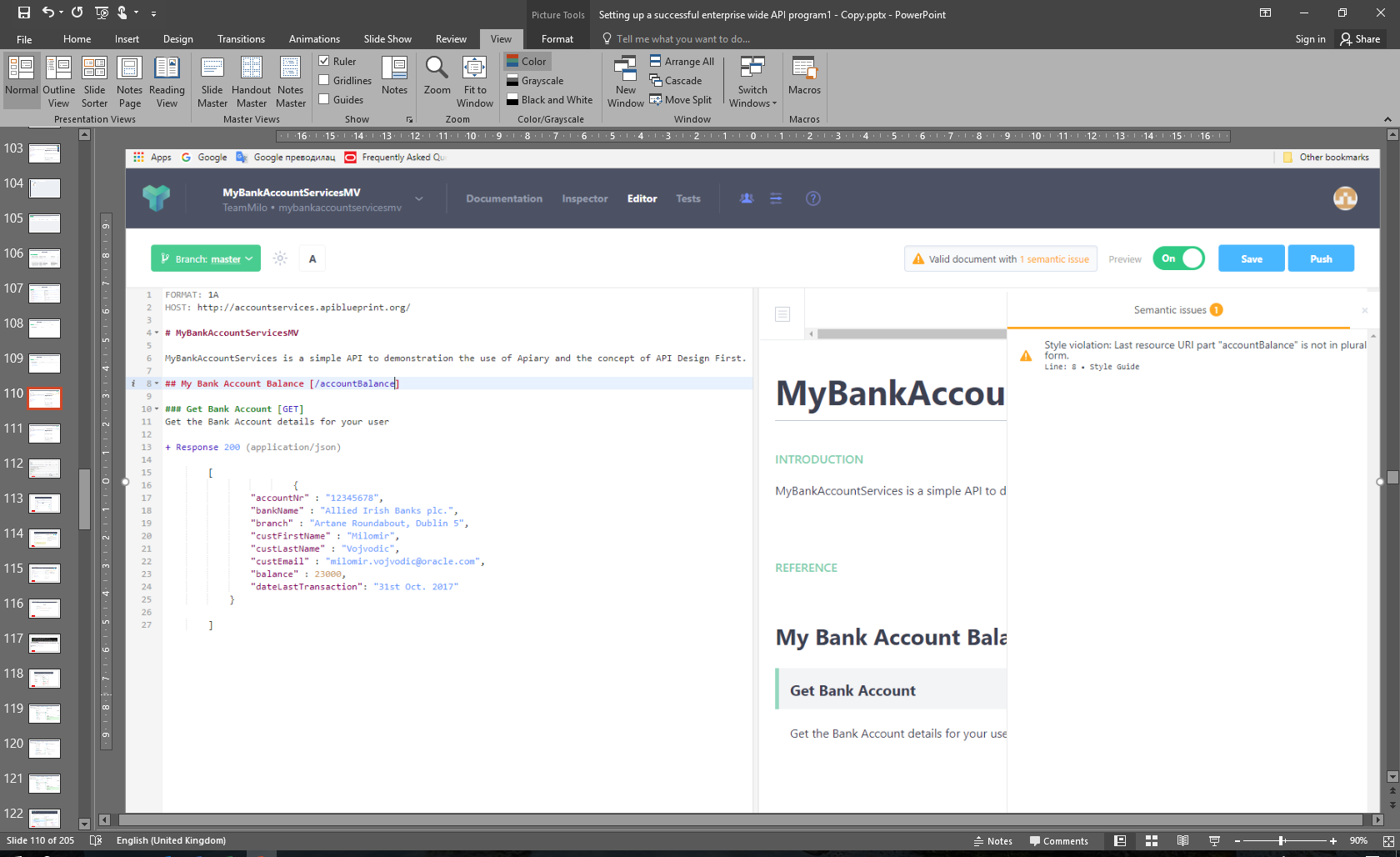
1. Click on Styles
2. Click on Assertions

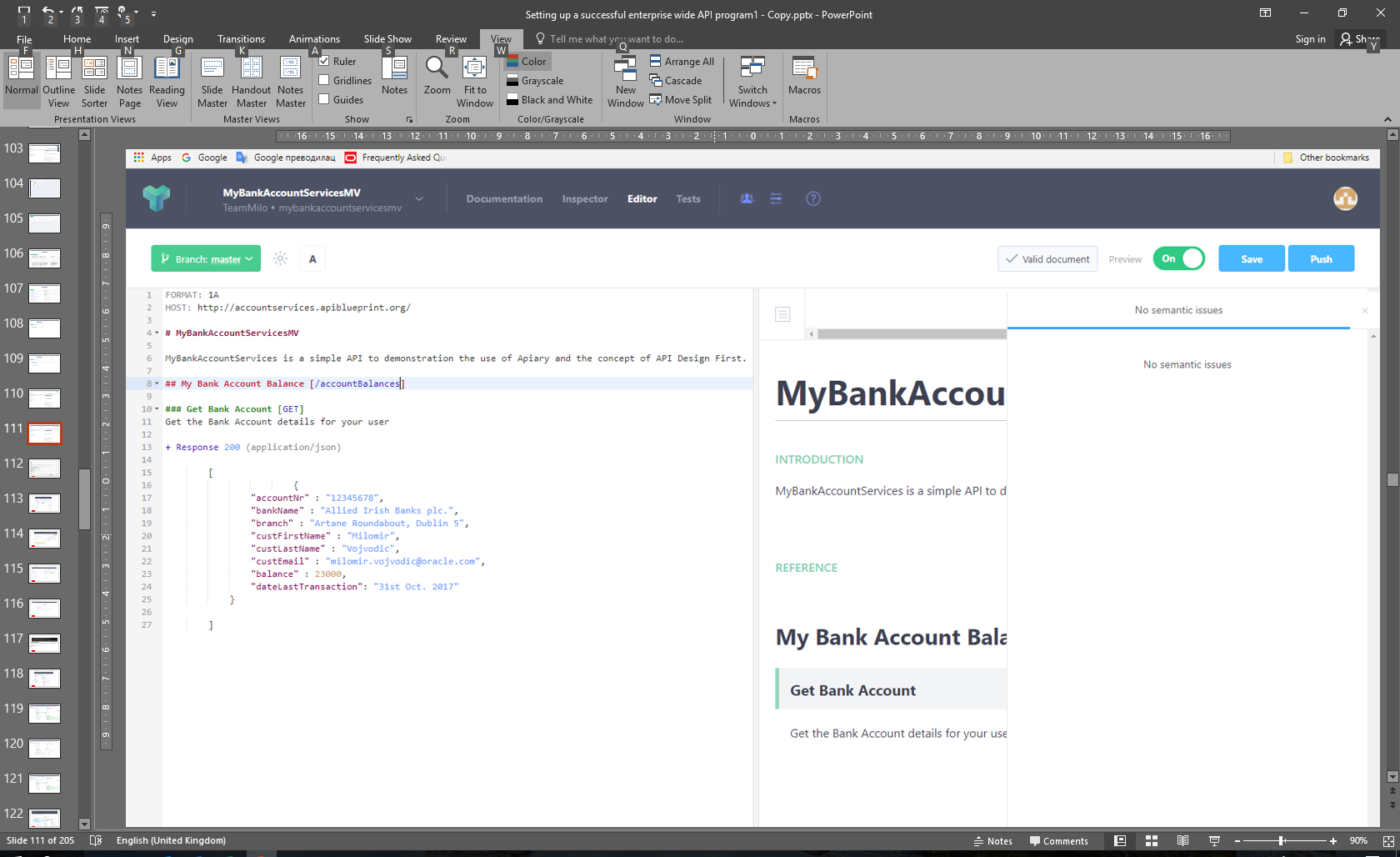


1. Select for example 3rd and 8th Rule
2. Click on Advanced View
3. Click on Mouse Click View
4. Go to the Dashboard
5. Click Check APIs
6. Click on Passed and Failed Rules to see details of one passed and one failed rule



1. Click on API in oder to get back to Editor View
2. Review The Semantic Errors

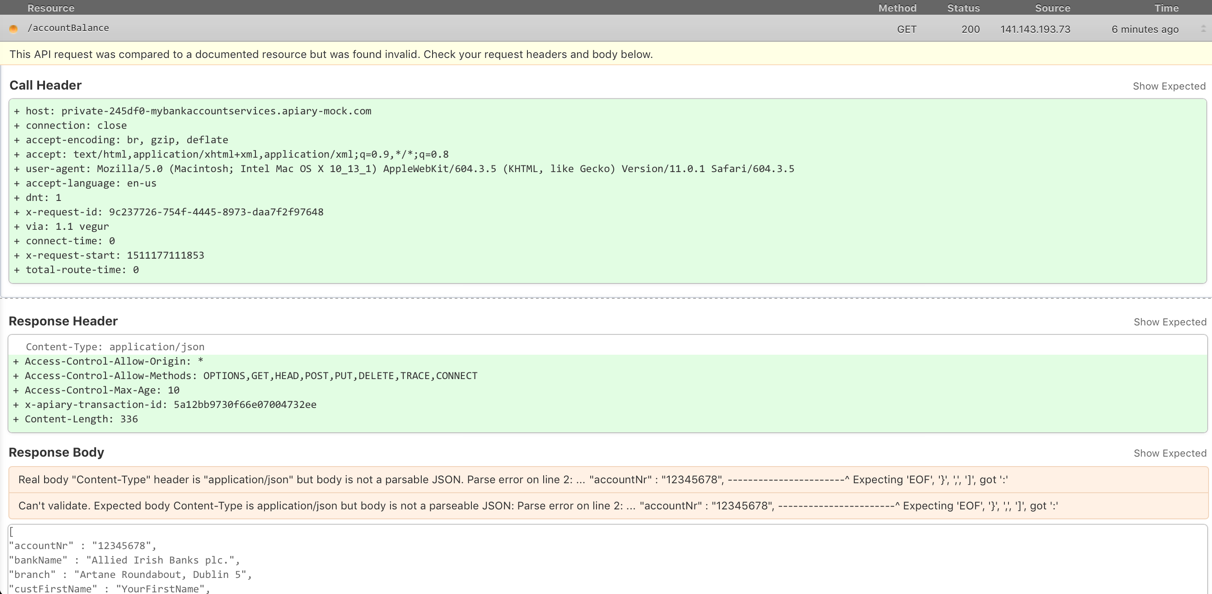


1. Make a change in ## My Bank Account Balance from singular to pluraal, from [/accountBalance] to [/accountBalances]
2. Review that there no more semantic issues
3. Go back and delete in Assertions Rules that were selected

# Apiary API Inspector

Any request to or response from the Mock Server will be logged by the Apiary Debugging Server and made available in the API Inspector. This means that you can use your favorite tool for interacting with your API (such as curl or Paw) and Apiary will log that traffic.

## Use the API Inspector to review API requests

1. Navaigate to the API Inspector
   1. Click on Inspector from the menu
   2. If shown, click leave page
2. You should see atleast two requests for your API resource
3. Select one of the resource request to view the Request and Response

Congratulations you have completed the APIPCS API Desigin First lab.

# Apiary designed APIs tested using Dredd

Lab copied and simplified from this location : <https://redthunder.blog/2017/09/20/apiary-designed-apis-tested-using-dredd/>

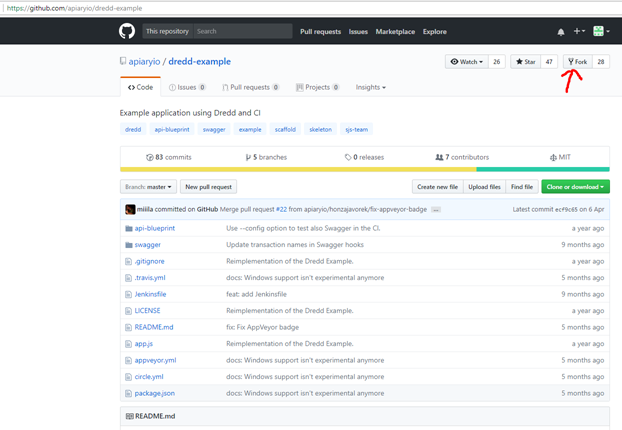
Pls follow the original lab for all steps if simplified version has some non-clarified steps

## Fork the Dredd- project from GitHub

After logging into GitHub, search for the project <https://github.com/apiaryio/dredd-example>

Once you have opened this project, click the **Fork** button in the top right.

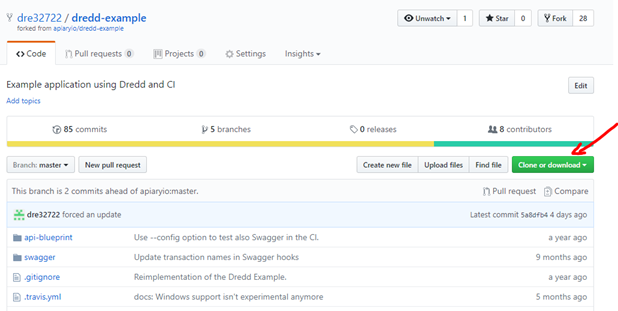
Taking a closer look at the dredd-example project, you will discover that it is a Node.js project that has an implementation of a few APIs relating to Gists.  
It provides a really great way to get started learning about how Apiary, GitHub and the Dredd testing framework can work together as part of the API Lifecycle.



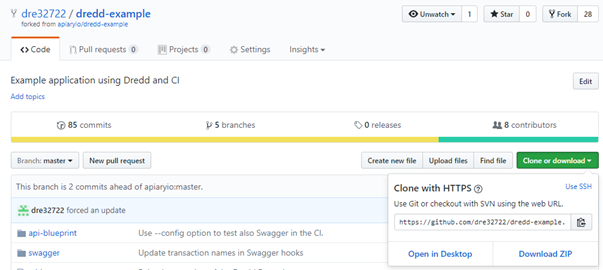
## Git clone, Install Dredd And Dependancies

Sign into GitHub account and access the forked copy of the Dredd-Example project. This project will be visible under the Repositories tab.

Click the **dredd-example** link.



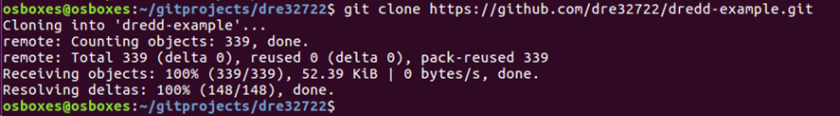
Copy the web URL for the dre32722/dredd-example repository.



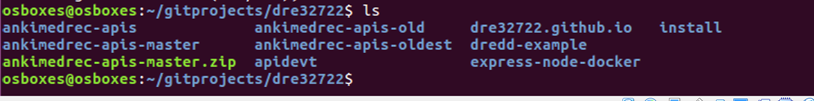
In local dev env, **cd** (change directory command-line) to directory locally like in this case  **/home/osboxes/gitprojects** directory.

**Git clone** fork of the dredd-example project.

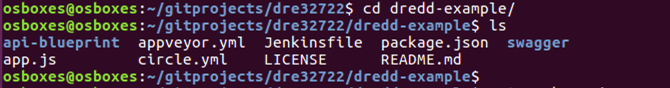
https://solutionsanz.files.wordpress.com/2017/09/092017_1346_apiarydesig11.png?w=840



The above step creates a directory locally like in this case /home/osboxes/gitprojects/dre32722 directory.



**cd** (change directory command-line) into the newly created **dredd-example** directory.



Install Dredd by doing the following; **npm install -g dredd**

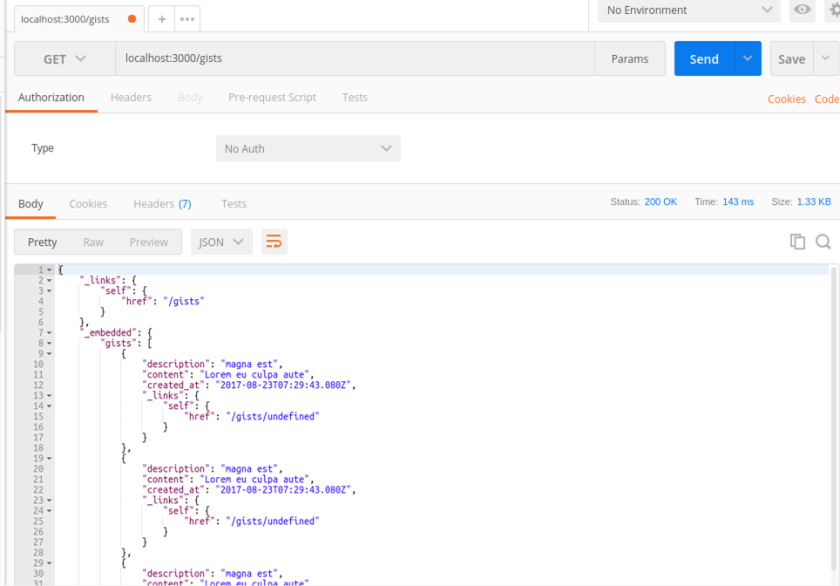
The install takes a few minutes.

The dredd-example is a node.js project and there are a number of dependencies specified in the package.json file that are required in order for it to run.

**npm install**The above command will install all the required modules for the dredd-example project.  
These dependencies include mongodb, express, body\_parser and dredd.  
Confirmed that you could successfully run the app and access the APIs using Postman.

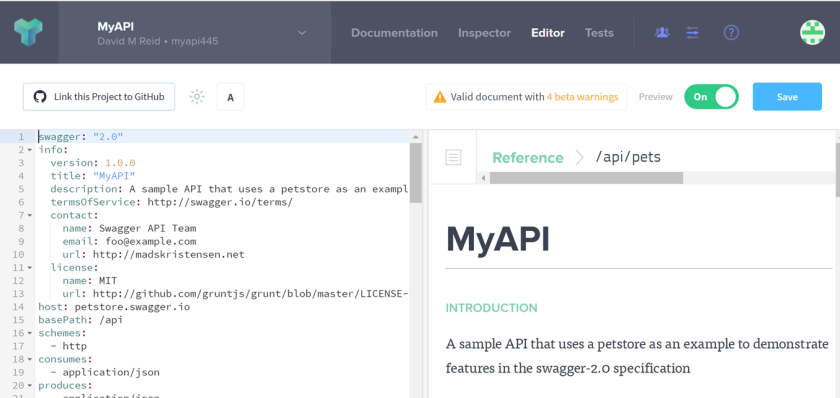
https://solutionsanz.files.wordpress.com/2017/09/092017_1346_apiarydesig19.png?w=840

Launch REST client -postman and execute an HTTP GET of localhost:3000/gists as per screenshot below.



## Actions On Apiary.io

Create an API project.   
You will notice that when the API project is created, that the PetStore API sample is automatically populated as your project API design, as per “MyAPI” project screenshot below.

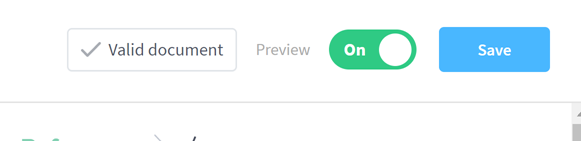


Copy the swagger.yml file contents into the Apiary project

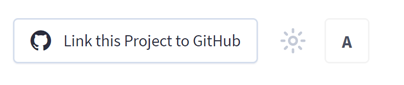
From the forked project dre32722/dredd-example copy the contents of the swagger.yml file.  
Than paste this into Apiary project, overwriting the PetStore API definitions.

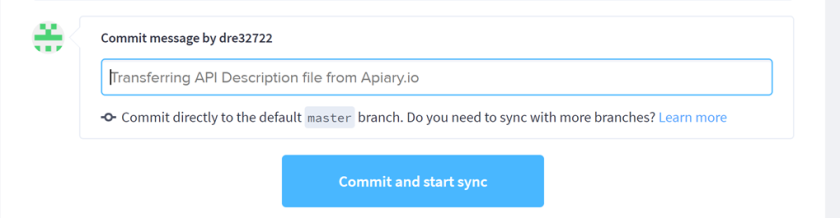
Click the Raw Button and then copy the content. Paste the Gist Fox API content from the api-description.yml file into API project.

You should see that your document is Valid. Press the Save button to save the changes made to your project API design.



Click the button in order to link the Apiary project to the forked GitHub project. In my case dre32722/dredd-example.





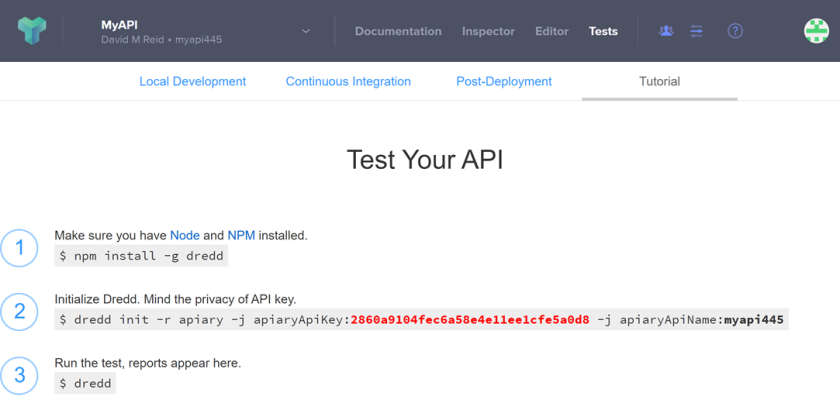
Provide a Commit Message or Accept the default message and then press the Commit and start sync button.

## Actions on Local Dev Environment

Initialise the dredd-example project

Initialise project so that Dredd tests and Apairy project were linked. To do this find out the API key for Apiary API project.

To get this information Click the Tests Tab and this provids details describing how to initialise dredd to support Test Driven Development.

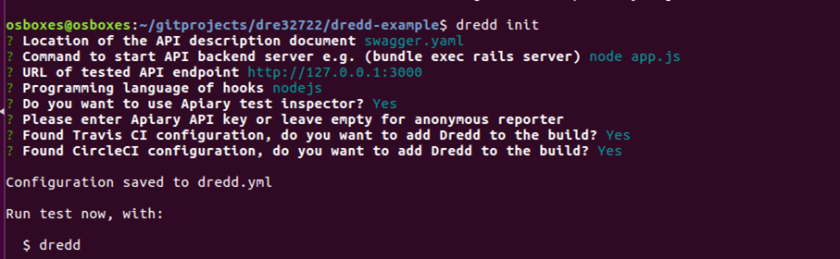


**Cd** (change directory command-line) into directory locally like in this case  **/home/osboxes/gitprojects/dre32722/dredd-example** project directory.  
Execute the dredd init command as follows.

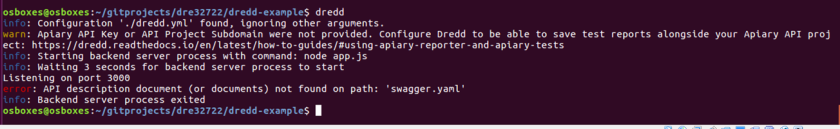
https://solutionsanz.files.wordpress.com/2017/09/092017_1346_apiarydesig39.png?w=840

## Run dredd tests

In order to run dredd to test your APIs against the Apiary design, simply do the following;



If you haven’t initialised dredd with your Apiary project you will get a warning message – Apiary API Key or API Project Subdomain were not provided.



When properly initialised, when you run dredd command, it will start the node application and run the tests.

Each test result will be displayed on the screen but will also be pushed into Apiary.

Please note below the link to the apiary project and the test results.  
Copy the URL specified at the end of the test run / output and pasted into a browser in order to view the test results in Apiary.

## View test results on Apiary.io

Copy and paste the URL from the dredd console output web browser and it resolved to the following;



The above changes should mean that several of the failed tests in the first dredd run will succeed when you run the tests again**.**