

What Is Apiary?

Apiary allows us to first create the design of an API and then implement it. Apiary helps in:

1. Creating a mock API framework
2. Generate its documentation fairly quickly

In the API design, we specify:

1. The supported operations
2. The input parameters for the operations
3. The output data model or in other words, JSON output

These mock APIs can be consumed in the applications to see if the API design meets the application needs. With Apiary, the modifications can be quickly modified and can go through a number of iterations very quickly without writing any code.

Why Apiary?

Generally, if we wish to create an API, we first define the schema for the input and the output. Then, we define the operations to return the data; in other words, we implement a minimal solution before it can be integrated.

With Apiary, APIs can be designed (or declared) where the input and the output data can be mocked. Mocked data represents the output the action will return. This mocked API and the data can be integrated and tested to verify that it meets the application requirements. These changes can be made without any coding effort.

This also helps when there are multiple teams working simultaneously. One team works on API implementation and other teams consume the APIs. Once the contract is finalized, both the teams can work independently.

Advantages of Apiary:

API Blueprint

DNA for your API—powerful, open sourced and developer-friendly. The ease of Markdown combined with the power of automated mock servers, tests, validations, proxies, and code samples in your language bindings.

Server Mock

It's often hard to see how an API will be used until you have the chance to code against it. What wireframes are for UI design, a server mock is for API design. A quick way to prototype an API - even before you start writing code.

GitHub Sync

Two clicks will link Apiary to a repository of your choice. It's up to you whether you make the API Blueprint private or public and let community contribute. We update API docs every time you commit, and we push commits to the repo whenever you update your documentation at Apiary. It's a virtuous cycle. We even support on-premise GitHub Enterprise!

Command Line Tools

As nice as our online editor is, we know the command-line is what really counts. The Apiary CLI Gem lets you validate your API Blueprint, preview the documentation, publish it to Apiary—or seamlessly automate Apiary goodness in your own workflow.

Features of Apiary:

Dashboard

Dedicated, web-based team and API Blueprint management dashboard

Access Control

Role-based access control over API documents

Roles

Admin, Editor, and Viewer roles

Provisioning

Add and remove team members from API design projects

Template

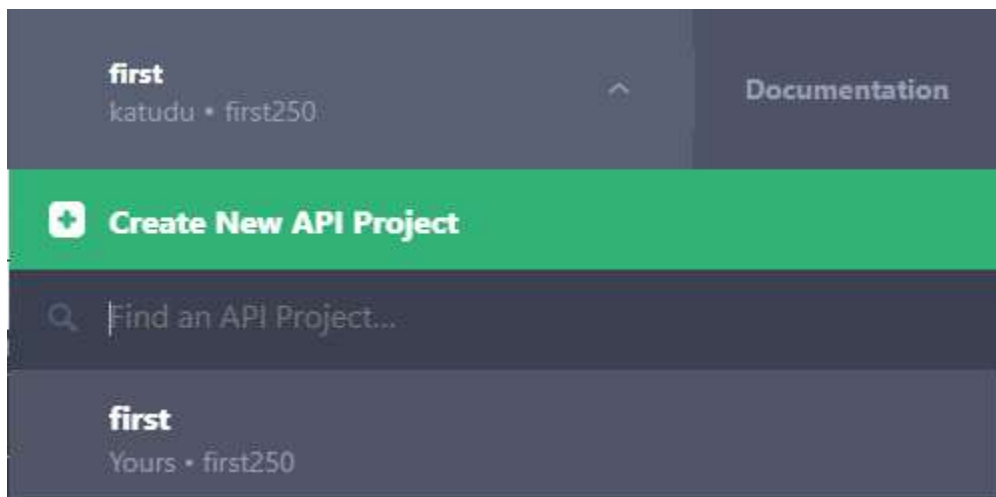
Shared API Blueprint templates to bootstrap new projects

Customization

Default settings for API Blueprint visibility and new team member provisioning

How to get started ?

1. Create a new API by clicking on the green highlighted item "Create New API Project"



2. Give a name to your new API Project. Here we will name it "FirstAPI". You can start you API in either API Blueprint or Swagger. We will go select API Blueprint for this example.

Note: Personal API is covered under the free terms of service, but for Team API, one will need to upgrade to Apiary Standard.

New API

Personal API ☒

Team API

katudu

Create a team

New API name

☐ Private API

Start your API in [API Blueprint](#) ▼

You can change selected format anytime.

Create API

3. Once you create your API, you will see two columns on your screen. The left column contains the Apiary editor. This editor provides instant validation and live preview enables you to describe, test, share, and collaborate on an API in minutes.

This is how API Blueprint looks like:

```

1  FORMAT: 1A
2  HOST: http://polls.apibluprint.org/
3
4  # FirstAPI
5
6  Polls is a simple API allowing consumers to view polls and vote in them.
7
8  ## Questions Collection [/questions]
9
10 ### List All Questions [GET]
11
12 + Response 200 (application/json)
13
14     [
15     {
16         "question": "Favourite programming language?",
17         "published_at": "2015-08-05T08:40:51.620Z",
18         "choices": [
19             {
20                 "choice": "Swift",
21                 "votes": 2048
22             }, {
23                 "choice": "Python",
24                 "votes": 1024
25             }, {
26                 "choice": "Objective-C",
27                 "votes": 512
28             }, {
29                 "choice": "Ruby",
30                 "votes": 256
31             }
32         ]
33     }
34     ]
35
36 ### Create a New Question [POST]
37
38 You may create your own question using this action. It takes a JSON
39 object containing a question and a collection of answers in the
40 form of choices.
41
42 + Request (application/json)

```

4. Metadata, API Name & Description:

The first step for creating a Blueprint is to specify the API Name and metadata:

```

1  FORMAT: 1A
2  HOST: http://polls.apibluprint.org/
3
4  # FirstAPI
5
6  Creating our first API.

```

The Blueprint starts with a metadata section. The FORMAT keyword is required and denotes that document is API Blueprint.

First level heading # FirstAPI will become API name.

5. Resource:

API consists of *resources* specified by their URIs. In this example we have a resource called **Question Collection**, which allows you to view a list of questions. The heading specifies its URI inside of square brackets [/questions].

```
8 ▾ ## Questions Collection [/questions]
```

6. Actions:

You should specify each action you may make on a resource. An action is specified with a sub-heading with the name of the action followed by the HTTP method.

```
8 ▾ ## Questions Collection [/questions]
9
10 ▾ ### List All Questions [GET]
11
12 + Response 200 (application/json)
13
14 [
15   {
16     "question": "Favourite programming language?",
17     "published_at": "2015-08-05T08:40:51.620Z",
18     "choices": [
19       {
20         "choice": "Swift",
21         "votes": 2048
22       }, {
23         "choice": "Python",
24         "votes": 1024
25       }, {
26         "choice": "Objective-C",
27         "votes": 512
28       }, {
29         "choice": "Ruby",
30         "votes": 256
31       }
32     ]
33   }
34 ]
```

DOCUMENTATION TAB:

7. Under the "Documentation" tab, you can test your API on a mock server by selecting one of the questions.

Questions Collection

List All Questions




Create a New Question



You may create your own question using this action. It takes a JSON object containing a question and a collection of answers in the form of choices.

8. By selecting "Mock Server" and clicking on "Call Resource", we can see the response of our API on the mock server.



Switch to Example

Questions Collection / List All Questions

Console calls are routed via Apiary [Use browser](#) ?

GET

https://private-8c013-firstapi530.apiary-mock.com/questions

URI Parameters

Headers

Body

Reset Values

+

Add a new query parameter

Show Code Example

Mock Server

▼

Call Resource

9. The response will be displayed in the following format displayed below. We are actually hitting a live mock server to get our response and if we change our API documentation, it changes in real time as well.

> Request

GET http://private-8c013-firstapi530.apiary-mock.com/questions

✓ Response

200

Response Headers Real Diff Specification

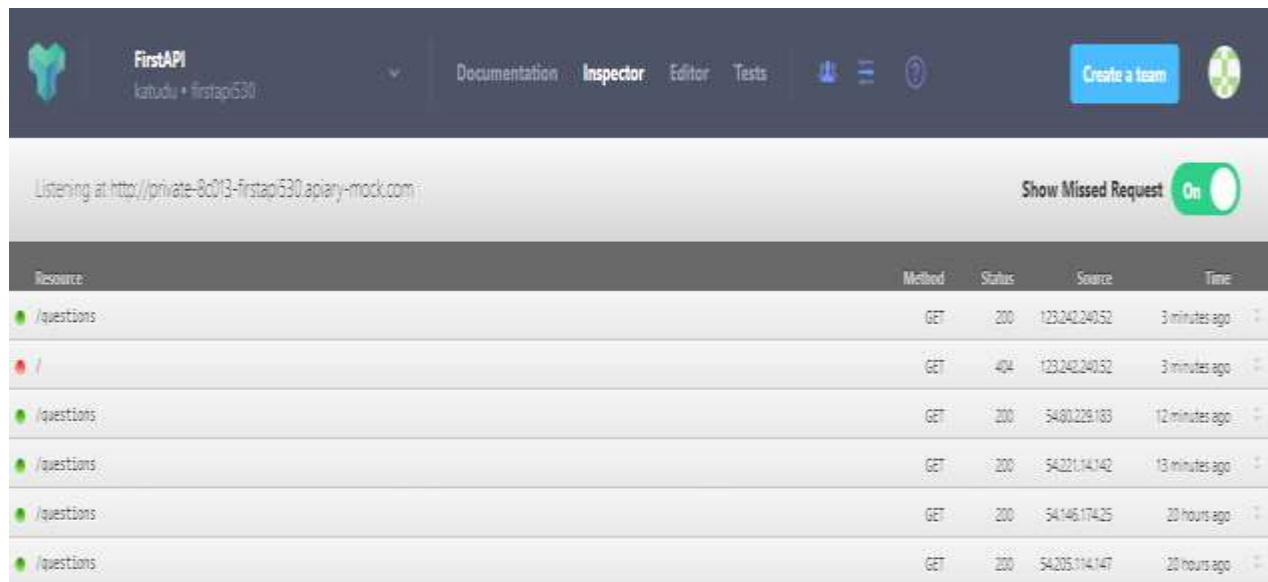
```
1 Content-Type: application/json
+ 2 Access-Control-Allow-Origin: *
+ 3 Access-Control-Allow-Methods: OPTIONS,GET,HEAD,POST,PUT,DELETE,T
+ 4 Access-Control-Max-Age: 10
+ 5 x-apiary-transaction-id: 5c665b3f4246cf0a00da80f5
+ 6 Content-Length: 496
```

Response Body Real Diff Specification







```
1  [
2    {
3      "question": "Favourite programming language?",
4      "published_at": "2015-08-05T08:40:51.620Z",
5      "choices": [
6        {
7          "choice": "Swift",
8          "votes": 2048
9        },
10       {
11         "choice": "Python",
12         "votes": 1024
13       },
14       {
15         "choice": "Objective-C",
16         "votes": 512
```

INSPECTOR TAB:

10. Under the "Inspector" tab, we can see the resources that we have called and displays the URL of the mock server.



The screenshot shows the FirstAPI interface with the 'Inspector' tab selected. At the top, it indicates 'Listening at: http://private-8c013-firstap530.apify-mock.com'. A 'Show Missed Request' toggle is set to 'On'. Below this is a table of API requests.

Resource	Method	Status	Source	Time
 /questions	GET	200	123.242.240.52	3 minutes ago
 /	GET	404	123.242.240.52	3 minutes ago
 /questions	GET	200	54.80.229.183	12 minutes ago
 /questions	GET	200	54.221.14.142	13 minutes ago
 /questions	GET	200	54.146.174.25	20 hours ago
 /questions	GET	200	54.205.114.147	20 hours ago


11. Clicking on one of the successful API requests will display the Call Header, Response Header and the Response Body.




People in **FirstAPI**


Invite people


E-mail address(es) comma separated

Can view 

Send Invites

All 

Editors 

Viewers 

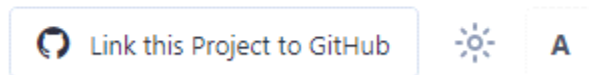
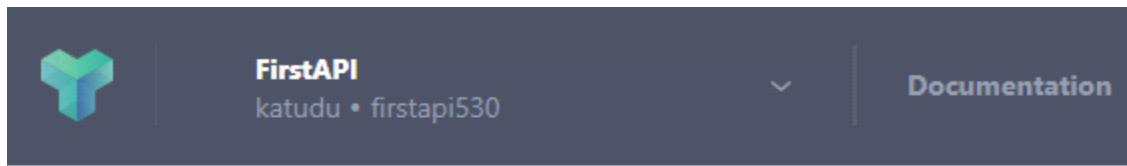
You Are Alone

Get your peers aboard and make this API rock!

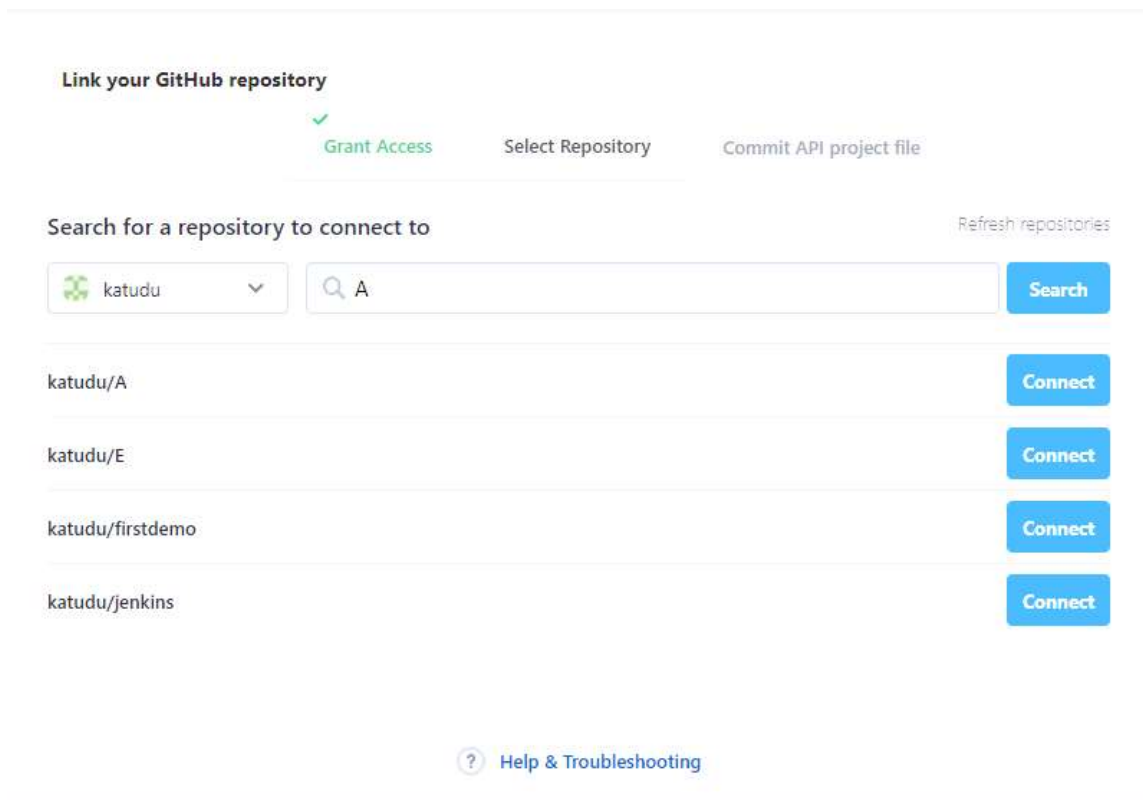
Create a team

Done

13. Another feature of Apiary is to link the API to GitHub by clicking on the button shown below.



We are then given the choice to select the repository where we want to connect our API.



After adding a commit message, we can successfully commit our API description file to GitHub.

Link your GitHub repository



Grant Access



Select Repository

Commit API project file

A / apiary.apib


[choose a different repo](#)

```
1 FORMAT: 1A
2 HOST: http://polls.apibblueprint.org/
3
4 # FirstAPI
5
6 Creating our first API.
7
8 ## Questions Collection [/questions]
```



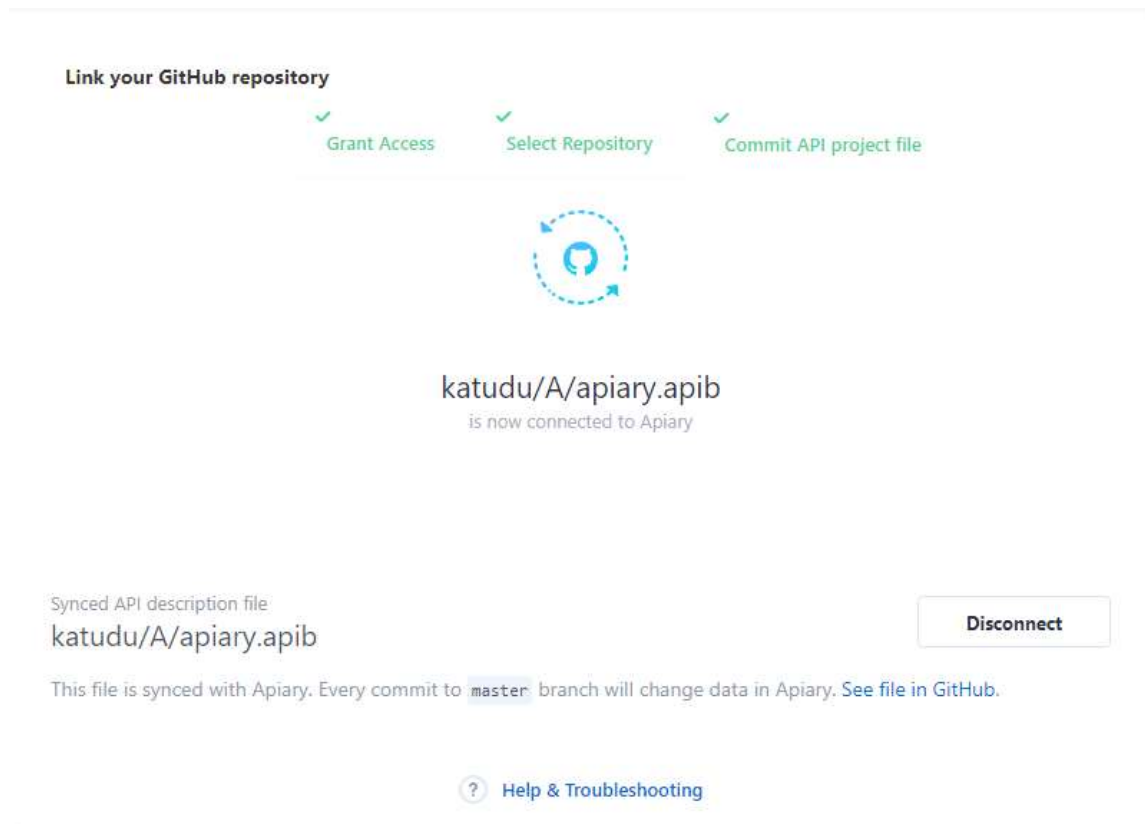
Commit message by katudu

Transferring API Description file from apiary.io

 Commit directly to the default `master` branch. Do you need to sync with more branches? [Learn more](#)

Commit and start sync

A confirmation message is subsequently displayed with an option to disconnect the file from GitHub as well.



14. After connecting the API description file to GitHub, our editor window has a couple of new changes. We can change our branch from the default "master" but only if we have a "Pro" apiary account. There is also a new Push button to push the file directly to GitHub.

