**Hands On Lab 1**

Create Your First OpenAPI Definition with Swagger Editor

OpenAPI definitions, formerly known as Swagger files, allow developers to specify the operations and metadata of their APIs in machine-readable form. This enables them to to automate various processes around the API lifecycle.

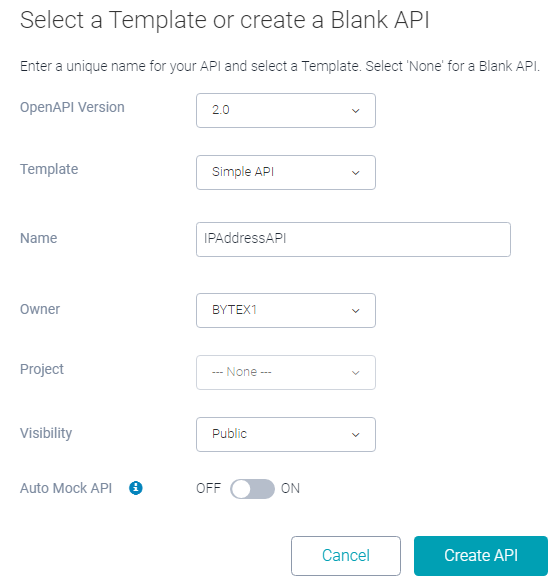
OpenAPI definitions are simple JSON or YAML files that you can create and edit with any text editor. But the right tooling can make your job a lot easier. In this lab, we will showcase one of tools for the API design stage, the Swagger Editor, and be your step-by-step guide for creating your first OpenAPI definition.

## **Swagger Editor Walkthrough**

Swagger Editor is freely available online at <https://app.swaggerhub.com/home.>

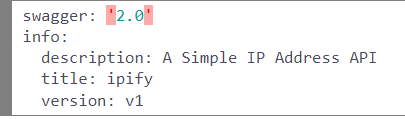
## **Create an OpenAPI definition**

Create a Blank API as shown below:



For the purpose of this lab ,we will use ipify, a simple API that allows software clients to determine their public IP address even behind a NAT. This API is a great example for testing because it is very simple, and also allows unlimited access without an API key. You are free to test with this API or, if you already have your own API, start building the definition for that.

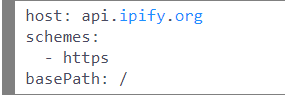
Add the following lines into the editor after clearing it fully.



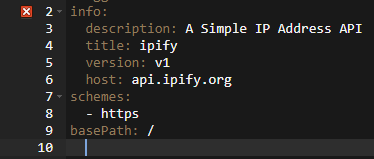
The first line indicates the type and version of the specification. Using info you can set basic human-readable information such as title, description and version.

Check out the right side of the editor now. Your title, version and description have been formatted.

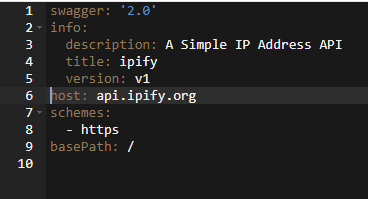
Continue with the information about the API endpoint’s base URL:



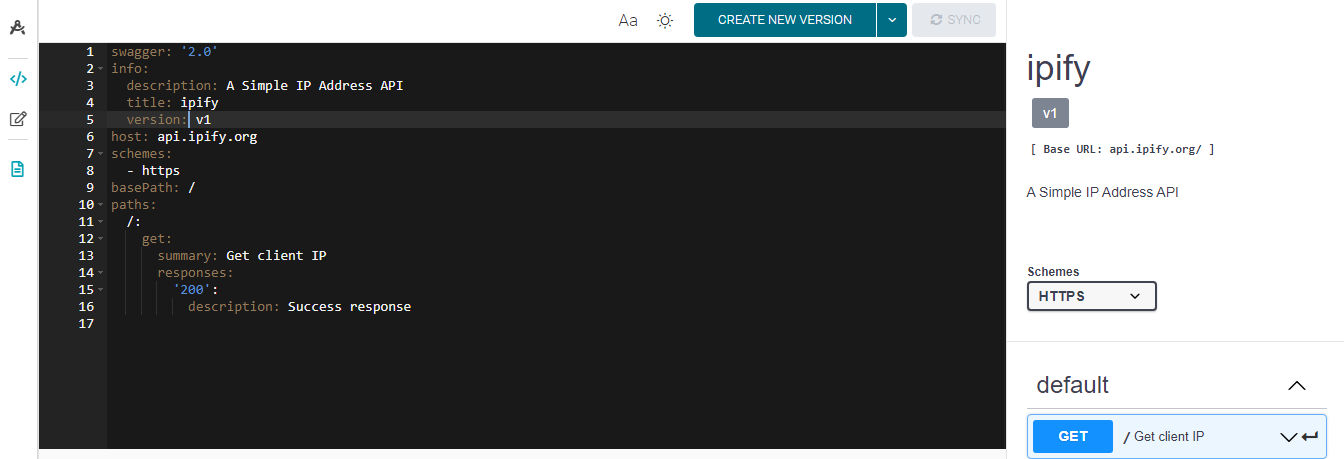
You may occcasionally also see a red box with a ‘x’ inside it. Parser errors, for example if you have malformed YAML, are shown with a red X on the respective line. Whenever you see the red box you know that there is something to fix in your definition. You should only generate code or save your definition and import it into another tool when the red error box has disappeared.



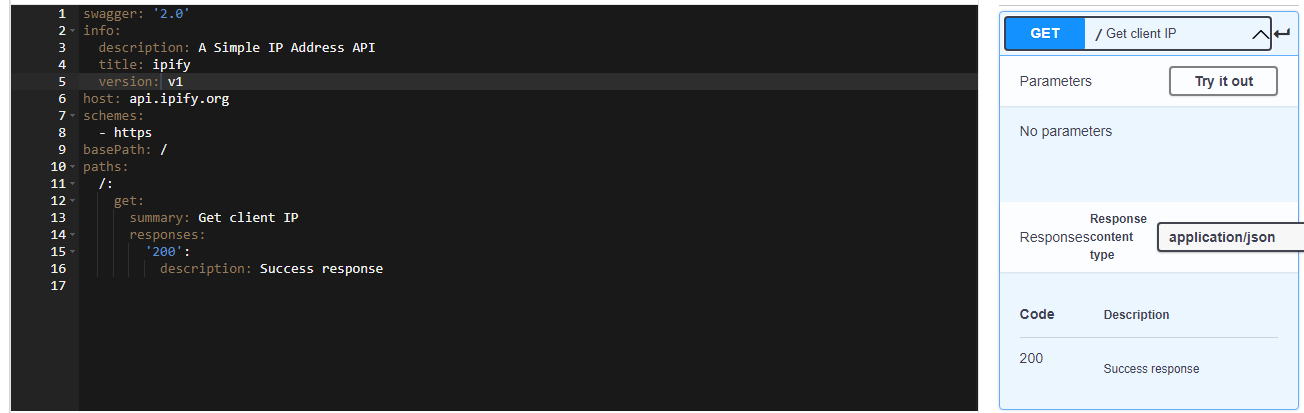
Fix the error as below:



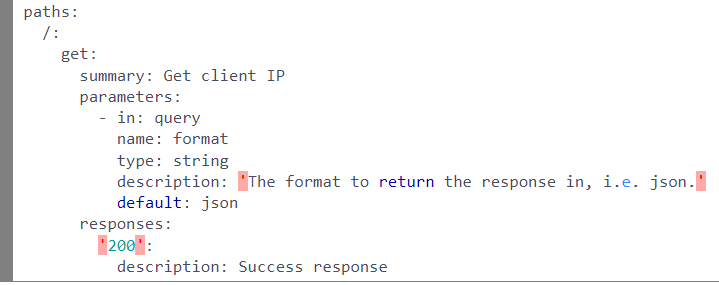
 adding an operation to our API:



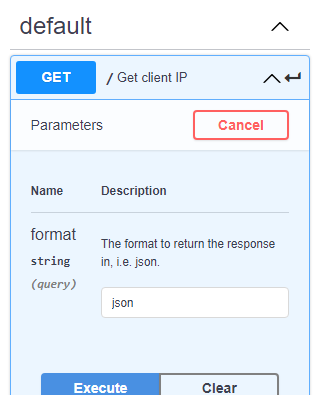
So what have we done now? We’ve added a single path at the root and an operation with the HTTP GET verb. The summary, which is basically the name of the operation, is set to “Get client IP”, and there’s one possible response defined for the 200 status code under responses. This is the bare minimum to have an operation listed in the documentation on the right side. You can already try the operation by clicking the ****Try it out**** button inside the operation’s box first (if you don’t see this box it might be collapsed, click on the operation’s name to open it), and then ****Execute****. The request is executed from your browser directly to the API and the response is shown. Note that this requires the API to support CORS (cross origin resource sharing), which ipify does.



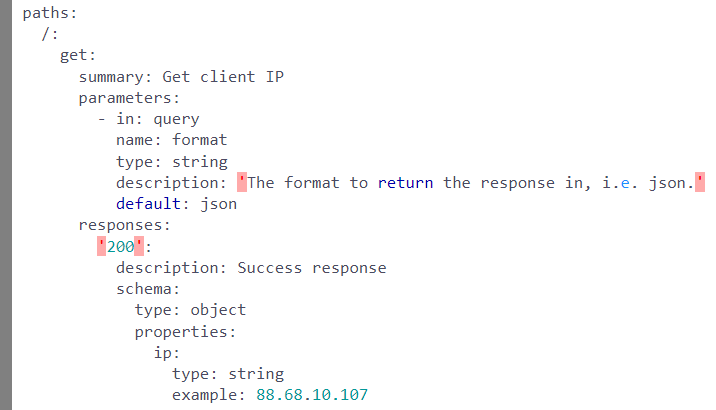
API operations typically require parameters. While ipify works without parameters, they do support a parameter to modify the response format. Let’s add this parameter! Here’s the extended snippet for the paths section:



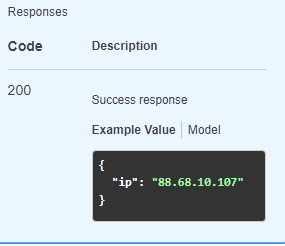
As you can see, parameters use in to declare where they are added to the request. In this example it’s a query parameter. It is possible to specify the name and type and also add a description and default value. If you look at the generated API documentation now you can see the list of parameters. Each parameter contains an editable text field so you can test your API operation with different inputs.



Finally, let’s add a description of the response, so the readers of our documentation can expect what the output of the API will be even before sending their request. Once again, here goes the full snippet for the paths section:



As you can see, we’ve added a schema property to the response. The response type is given as an object with a string-typed property called ip and an example value. In the generated API documentation, you can toggle between viewing this example or a description of the model for the response. Note that the OpenAPI specification relies on another specification called [JSON Schema](http://json-schema.org/) for modeling JSON objects for requests and responses.



Congratulations, you now have created your first OpenAPI file and observed how such a machine-readable definition can easily and automagically turn into interactive API documentation. Feel free to play around with it. You could import some examples from the web or go back to the Swagger Petstore example by clearing the editor and then refreshing the page.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*