"Port the Prometheus API to OpenAPI" proposal for Google Summer of Code 2021

Table of contents

Basic Information

Name: Kunal Pawar

Major: Computer Science And Engineering

University: National Institute of Technology, Bhopal

Github: @HelloKunal

Discord handle: HelloKunal #5535

Email: hellokunalpawar@gmail.com

Phone: (+91) 7869541752

Postal Address: 501 D, RB-V Rail Saurabh Colony Jabalpur, 482001 Madhya Pradesh, India

Timezone: Indian Standard Time (UTC +5:30)

Documenting RESTful APIs with Swagger

Abstract

This project aims to implement Swagger (aka OpenAPI) on Prometheus's RESTful API to generate self-documentation and specification files to be used by

third parties. This would enable the use of Prometheus API's with clients from any language easily. It will consist of two independent goals. **First revising API and generating specification files and, second integrating the self-documentation from swagger i.e. swagger UI into the official website suitably.**

Background

Prometheus, a Cloud Native Computing Foundation project, is a systems and service monitoring system. It collects metrics from configured targets at given intervals, evaluates rule expressions, displays the results, and can trigger alerts when specific conditions are observed.

Motivation

A large reason why API documentation is important is to increase API adoption. Comprehensive documentation on all of the functionality, how to effectively use and integrate, and updates on the API lifecycle improves the experience for those using your APIs. **Prometheus offers RESTful API of all sorts but they are not well documented** and thus there are some major problems that are encountered:

- Maintaining documentation, tests and implementation is hard.
- Implementation comes at the cost of user experience
- Communication between services built in different programming languages is not easy

Documentation is the key to a great experience when consuming your API. It not only enables consumer satisfaction, but also allows your API adoption to increase. Popular open source description formats like OpenAPI Specification and commercial platforms like SwaggerHub allow teams to automate the documentation process and work on a great overall experience consuming APIs.

OpenAPI

OpenAPI Specification (formerly Swagger Specification) is an API description format for REST APIs. An OpenAPI file allows you to describe your entire API, including:

- Available endpoints (/users) and operations on each endpoint (GET /users, POST /users)
- Operation parameters Input and output for each operation
- Authentication methods
- Contact information, license, terms of use and other information.

API specifications can be written in YAML or JSON. The format is easy to learn and readable to both humans and machines.

Swagger

Swagger is a set of open-source tools built around the OpenAPI Specification that can help you design, build, document and consume REST APIs. The major Swagger tools include:

- Swagger Editor – browser-based editor where you can write OpenAPI specs.
- Swagger UI – renders OpenAPI specs as interactive API documentation.
- Swagger Codegen – generates server stubs and client libraries from an OpenAPI spec.

Why Do I Want To Contribute?

I was deeply intrigued by this project and am willing to invest my summer in it due to two main reasons.
Firstly, this project has a direct correlation to Swagger and implementation of APIs. Similar to how Git has become the de facto open-source platform for developers, Swagger has emerged as the most popular approach to API design. It has recently been moved to the Linux Foundation under the name "The Open API Initiative", and is backed by companies like Google, Microsoft, Intuit and Atlassian. The initiative is focused on "creating, evolving and promoting a vendor neutral description format." Having a common language for describing APIs will make tooling much easier to use, and will change the way we utilize APIs.
Secondly, **this project will allow me to unfold a lot of new opportunities and provide a taste of implementing concepts in making scalable software.** Implementation of each feature requires getting acquainted with multiple techniques, technologies, libraries and proper knowledge about their compatibility with each other.

Brief Tentative Working

Major Dependencies

- Swagger (Apache License Version 2.0)
- go-swagger (Apache License Version 2.0)
- Redocly/redoc (MIT License) (Optional)
- Prometheus (Apache License Version 2.0)

Generating Swagger Spec

When it comes to creating the OAS definition, two important schools of thoughts have emerged: The “Design First” and the “Code First” approach to API development.
Here the API is already built so we continue with the “Code First” approach.
The Code First approach is a more traditional approach to building APIs, with development of code happening after the business requirements are laid out, then the documentation of the API is done from the code. There are 3 ways

- Generating an OAS definition with Inspector (Paid)
- OAS Generation During Runtime (Not Usable for Go APIs)
- **Swagger Generation During Build Time**

In this method, the OAS contract is generated when preprocessing the API, that is, before runtime. **Comments against various resources, methods and functions within the API help generate the OAS definition.** These comments are usually in a predefined, special syntax, based on the type of tool we use to generate the contract. The tool scans the API code for these special comments and produces the OAS contract as an output. Cakephp-swagger and grape-swagger are prominent examples of tools that generate the OAS contract during build time.

Integrating Swagger UI with the rest of the docs

- **Embed Swagger UI in docs**

  When using Swagger UI, it's not necessary for the Swagger UI output to be a standalone site. We can also embed Swagger into an existing web page. It’s pretty easy to embed Swagger into an HTML page — just copy the code from the index.html file from the dist folder into your doc page (more or less). However, the effect is still kind of clunky and is obvious that the content is embedded from some other document generator. It’s not a seamlessly branded experience.

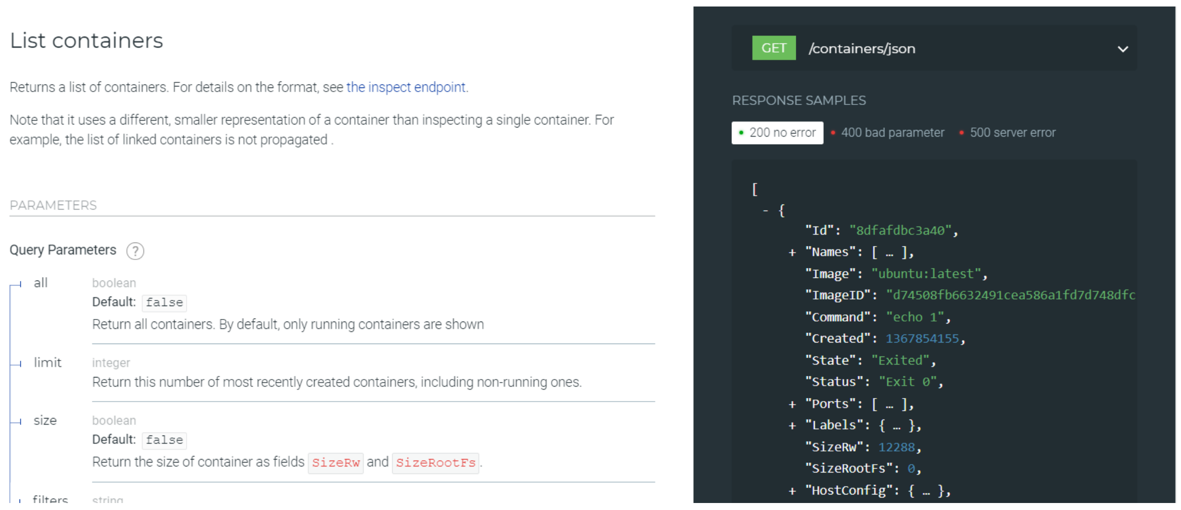
- Using a tool that imports Swagger and allows additional docs

  Another approach is to use a tool like Readme.com, Redoc or Stoplight that allows us to both import your OpenAPI spec and also add our own separate documentation pages.

Related Implementation

- Docker API Doc

  Created using Swagger and ReDoc



List containers

Returns a list of containers. For details on the format, see the inspect endpoint.

Note that it uses a different, smaller representation of a container than inspecting a single container. For example, the list of linked containers is not propagated .

PARAMETERS

Query Parameters

all	boolean Default: `false` Return all containers. By default, only running containers are shown
limit	integer Return this number of most recently created containers, including non-running ones.
size	boolean Default: `false` Return the size of container as fields `SizeRw` and `SizeRootFs`.
filters	string

GET /containers/json

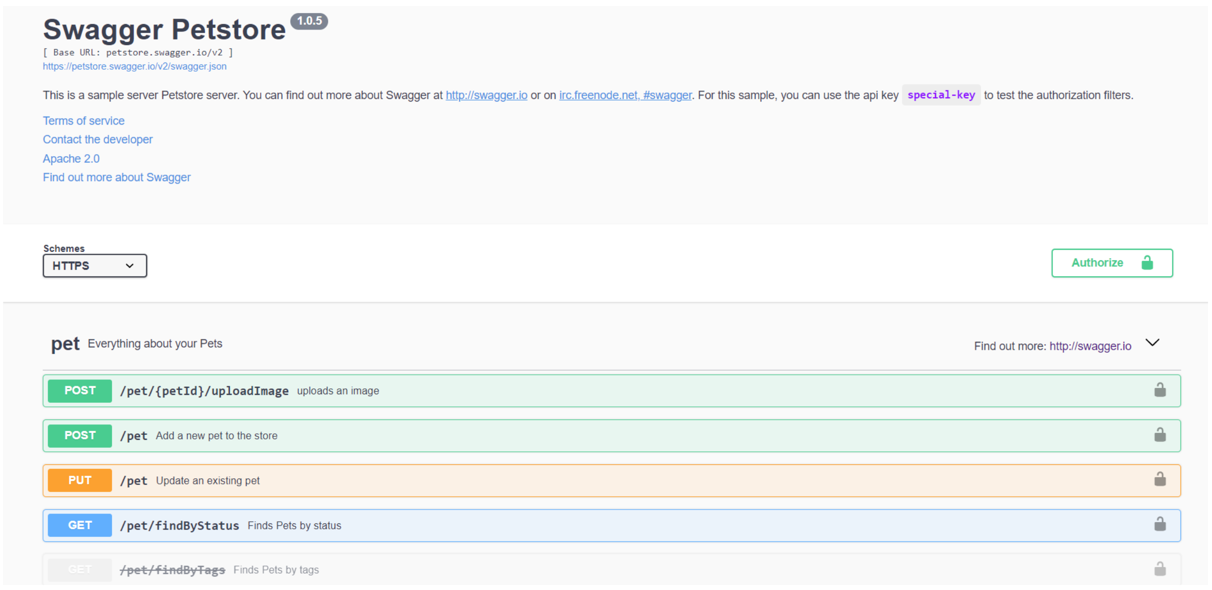
RESPONSE SAMPLES

● 200 no error ● 400 bad parameter ● 500 server error

```
[
  - {
        "Id": "8dfafdbc3a40",
      + "Names": [ … ],
        "Image": "ubuntu:latest",
        "ImageID": "d74508fb6632491cea586a1fd7d748dfc
        "Command": "echo 1",
        "Created": 1367854155,
        "State": "Exited",
        "Status": "Exit 0",
      + "Ports": [ … ],
      + "Labels": { … },
        "SizeRw": 12288,
        "SizeRootFs": 0,
      + "HostConfig": { … },
```

- **Swagger Petstore**

  Created only using OpenAPI. Though this website differs from yours because Prometheus documentation contains many more important links apart from HTTP API. Therefore we need to integrate this Swagger UI within the website.



Swagger Petstore 1.0.5

[Base URL: petstore.swagger.io/v2]
https://petstore.swagger.io/v2/swagger.json

This is a sample server Petstore server. You can find out more about Swagger at http://swagger.io or on irc.freenode.net, #swagger. For this sample, you can use the api key `special-key` to test the authorization filters.

Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger

Schemes: HTTPS

Authorize

pet Everything about your Pets — Find out more: http://swagger.io

POST	`/pet/{petId}/uploadImage`	uploads an image
POST	`/pet`	Add a new pet to the store
PUT	`/pet`	Update an existing pet
GET	`/pet/findByStatus`	Finds Pets by status
GET	~~`/pet/findByTags`~~	Finds Pets by tags

Proposed Deliverables

1. Prometheus API v2
   a. Added method for generating Swagger Spec (Most probably Swagger Generation During Build Time)
   b. Swagger.yaml is also autogenerated as a specification file.
   c. Added docs.go as readme and for generating docs
   d. Added makefile and .bat file to execute swagger

2. (Optional) Updated Prometheus HTTP API Documentation
    a. Integrated Swagger UI with rest of the docs

Schedule

Phase 0 [Pre-GSoC Period]

- 4 Weeks (13 April - 17 May)

  During this period, I'll try to solve more tasks and also focus on solving some of the GSoC qualification tasks. Along with that, I'll carry on my experiments and demos with necessary technologies being used in this project to gain a deeper understanding of the same.

- 1 Week (22 April - 29 April)

  End semester exams begin at my institute, and I will not be contributing actively during this period. Nonetheless, I'll be actively following the progress and participate in conversations over github.

Phase 1 [Community Bonding Period]

- 2 Weeks (May 17 -- June 7)

  During the community bonding, the main focus will be to frame a roadmap for the project with the guidance of the mentor (along with improving bonding, which is what the period is for). My main priority will be to clarify integration details related to the documentation website. This period will also be used to study the structure of the Prometheus API to understand all the swagger components required.

Phase 2 [Coding Period 1]

- Week 1--3 (June 7 -- June 27)

  I'll initialize makefile, bat file and docs.go to execute swagger. Then I start adding commented syntax to the API which will help generate the spec file. Since the API is large, this process will take a long time. Therefore I will prioritize certain sections first, which will generate a decent swagger UI for Phase 1 evaluations.

- Week 4--5 (June 28 -- July 11)

  During this week, I'll work on integrating a new swagger UI to the documentation website. I may start 1 week early since the initialization of this step may take significantly more time depending on the method used.

Phase 3 [GSoC Phase 1 Evaluations]

This period will be used to write a detailed report on the work done in Coding Period 1. The presented work will include the swagger spec file and the autogenerated swagger UI.

Phase 4 [Coding Period 2]

- Week 7--9 (July 19 -- August 8)

  I will spend most of the time completing the commented syntax for the entire API which will complete the spec file. I will also add minor adjustments to swagger UI and the website.

- Week 10 (August 8 -- August 15)

  This week will be spent bug fixing, completing the requirements given by the mentor, and completing the final evaluation report.

Phase 5 [Final Evaluation]

All the deliverables promised for GSoC will be provided by this stage along with a detailed report.

Personal Information

Personal Details

I am Kunal Pawar, an undergraduate student at National Institute of Technology Bhopal . My time zone is UTC+05:30.
I have always believed that the internet has the potential to bring positive changes to human life. From a very young age I started developing an interest towards programming. I am fascinated by the way it impacts various services like ticket reservations in India, which prompted me to pursue a career in programming.
I initially started programming in C/C++, Java in my final year of schooling, where I created a Netbeans Java project that went beyond the requirements. Later I moved to Python, HTML and **CSS/JavaScript** where I created various projects to

understand the language better. In order to expand my toolset, I started learning '**Go**' while undergoing an online course and gained working proficiency in it.
I aspire to contribute more towards the Open Source community.
I hold a good experience at competitive coding and my codechef handle is ''kupnoodle.''

Working Environment And Schedule

I'll be initially working from home where I've constant internet connection. Mostly working full-time on the code on weekdays (Monday to Friday). **My awake hours would usually be between 7 AM IST (1:30 AM UTC) to 12 midnight IST (6:30 PM UTC)** and I'm comfortable working anytime during this period. Except for a few days of traveling (which I'll be informing in advance to my mentor), I'll be having no other absences. Anyhow, in cases of emergency, I'll responsibly notify my mentor of the same with enough detailing.

Communication

I'm very flexible with my schedule and already have the habit of working at night and hence timezone variation (with my mentor) won't be an issue. I'm comfortable with any form of communication that suits my mentor. Below are the various options available:

- Email: hellokunalpawar@gmail.com
- Phone (Call, WhatsApp and Telegram): (+91) 7869541752
- Microsoft Teams: 191112004@manit.ac.in
- Discord handle: HelloKunal #5535

"Data is the new oil. And if data is indeed equivalent to oil, then APIs are the pipelines, and API backends are like the refineries"