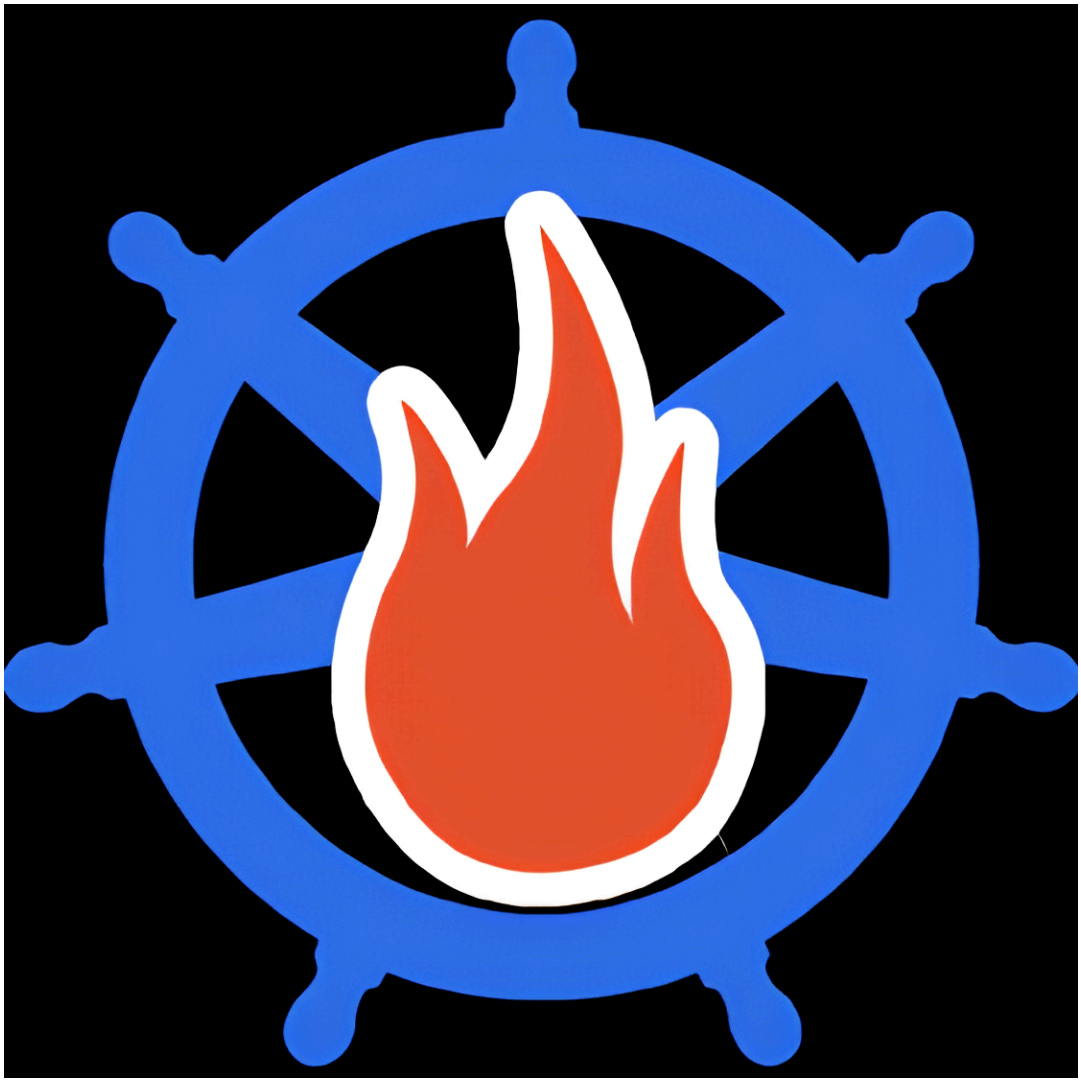

Prometheus Operator - Website / Documentation Revamp

GSoC - 2024 Proposal

Project Size : Medium

Table of Contents

About Me

Name	S Ashwin
Date of Birth	11/01/2003
Country	India
Email ID	ashwinsriram11@gmail.com
Github ID	AshwinSriram11
University	Indian Institute of Technology (BHU), Varanasi
Contact Number	+91 63074 81473
Timezone	UTC+05:30 (Asia/Kolkata)
Preferred Language	English

Past Experiences

1. **Tech Team, Games & Sports Council, IIT BHU (Github)**
   I have been part of the tech team for developing the official website of the Games and Sports Council. It utilizes ReactJs and TailwindCSS for its responsive UI.

2. **Music Player Website (Github)**

BeatBridge is a music player website with a recommendation engine system. I worked on building the responsive UI for the website and also integrated the Spotify REST API to search songs, albums, podcasts, etc.

3. Completed Hacktoberfest 2023

4. Other Web Development Projects

Past Contributions

From my time of joining i.e 15th March, 2024, I have made the following contributions -

Issue Raised

- #6428: 404 error in website

PR

- #6449

Abstract

Prometheus Operator serves as an important tool for deployment, configuration and management of Prometheus instances within a Kubernetes cluster. However, it is still difficult to leverage the functionalities of Prometheus Operator to its full potential. The main reason for this drawback is documentation being outdated and ambiguous. This is due to documentation not being updated in a timely and agile manner. The documentation hosted on https://prometheus-operator.dev/ is not currently aligned with the latest updates of the project. This results in wasted time, redundant efforts, communication gaps among team members working on the project, and a lack of clarity for users trying to utilize the tool.

This document will serve as a solution to this problem by explaining about -

1. The current issues in the documentation and the development process
2. What could be done to revamp the documentation to its full potential

With this being implemented, more emphasis should also be made on the steps for the future to avoid facing this difficulty.

Motivation

Walking in as a newcomer, it is intimidating to pick up any issue and start working on it. Seeing all the project ideas, this project seemed to be a good starting point for me to learn about Prometheus Operator and to dive deep into the concepts it brings to the table. As I am fairly new to this project, I think that taking small steps at a time would be better to get the best educational experience I can out of the project. My goal in this project would be to take the documentation to a level where a newcomer can find this project easy to work with and appeal to them about the importance of Prometheus Operator.

Project Plan

As discussed above, we will walk through the plan in two stages given below-

Challenges in Current Model

As the website for documentation is maintained in a different repository than the main repository, contributors are finding it difficult to update the changes in the documentation. The process of separately dealing with the documentation makes it a repetitive and a tedious process. Looking at the current documentation, there are many discrepancies that can be found and are listed below -

- **Concerns with Documentation Structure**
  If we look into the structure of the documentation, we can find that some topics are not belonging where they should actually belong. For example, the Prologue should not contain Contributing as a sub-section. There is no clear link between these two topics.

  We can also see that there are subsections of Getting-started and Quick-Start which introduces us to two different ways to set up Prometheus Operator locally. I believe all the installation methods should

be listed on a single page. Also, I believe there is no section on how to set up a local dev environment.

- **Insufficient Instructions for Contributing**
  The CONTRIBUTING file currently contains very basic instructions. As Prometheus Operator contains different CRDs to deal with, it is confusing to understand for a developer to start working as a contributor and be able to know the specifications required for a particular CRD or the changes they need to make for contributing to the project. This increases the workload for maintainers as they need to keep specifying how and where the changes are needed in the project. Also the changes made while contributing should also be simultaneously described in the documentation of the project making it an agile process which is currently not implemented.

  Let us take an example from the issues present in the repository. If we looked at the following issues #5859, #6360 and #6387, these three issues are of the same nature(related to Alertmanager CRD) and you can also observe that PR is used to explain how to contribute to this type of issue.

- **Ambiguous Separation of Concerns for Users and Developers**
  Although there is a separate section called User-Guide for a user to work with Prometheus Operator, it is still misleading whether it is enough for a user to simply follow the guide and get the idea of all use-cases. There is still no separate guide for developers. Developers have to go through the whole documentation to look for the relevant information.

- **Absence of References in Pre-Requisites**
  While mentioning the pre-requisites, there should also be a link or some reference to where a user can find the correct resources to learn about it. Absence of resources sometimes makes it less appealing to a user. Searching for a reliable resource to understand the topic can be time consuming as well as misleading.

Adjustments Needed in the Model

To improve the documentation, I believe implementing the following ideas might help in optimizing the current model -

- **Remodeling the Content**
  As the current structure presents many discrepancies for readability, I believe a new structure which incorporates existing docs, with changes to the current structure should be implemented to enhance the flow of content. Observing other documentations, I believe a basic structure could look like following -

    1. Introduction
    2. Getting Started
    3. User's Guide
    4. Developer's Guide
        a. Prometheus Operator
        b. Kube Prometheus
    5. Troubleshooting
    6. Contributing
    7. Community
    8. Blogs

  I can formulate a proper structure for the website by discussion with the maintainers and through careful research from other organizations(KEDA, Perses).

- **Establishing Consistent Documentation Standards**
  The goal is to create documentation that is easy to understand, navigate and ensure uniformity throughout. For this, each page should have uniformity in the way it is presented. A sample flow of a page can be as follows:

    1. What is X ?
    2. What does X do ?
    3. Pre-requisites
    4. Why is it needed?

5. Examples
6. Subtopics
7. Use Cases
8. Summary
9. FAQs

This will be beneficial for a developer to organize the content as well as beneficial for the user as the flow being similar on each page will be easily understood by the user.

- **Synchronizing Development with Documentation Maintenance**
  As the documentation is not going hand-in-hand with the development process, I think that we need to incorporate the idea of simultaneous maintenance of documentation. Whenever a PR is raised for a new feature implementation, or a bug being fixed, we can add the "**docs-needed**" label with the PR so that the contributor can also update the docs before merging the PR and the whole process becomes agile and streamlined and everyone puts more effort towards keeping the documentation as accurate as possible.

- **Version-Control in Documentation**
  Developers encounter difficulties understanding changes between versions, debugging issues, and maintaining consistency across the codebase due to absence of version-specific documentation. Also, there are many cases where some features from the previous version might be needed in future versions but are not present in the current version. So, implementing version-specific documentation brings various benefits to the development process.

  To implement this:
  1. Proper analysis across different versions is needed.
  2. Across each version, we need to keep track of all the new features, updates and bugs - fixes present in that version.
  3. Convey the key differences between two versions.
  4. Summarize all the changes from each version and implement it.

- **Integrating Flow Charts and Diagrams**
  Conveying the architecture, intricacies and flow of code becomes much easier if there is a visual aid for it. Incorporating flowcharts, diagrams can provide better support and may be able to clear the misconceptions much effectively in a concise manner, enabling developers to grasp the concepts efficiently.

  For this, proper identification of key components, processes and dependencies is necessary. After proper identification, I will use appropriate tools (Miro, Lucidchart, etc.) to create visual representations to convey code flow and logic.

- **Improving the Troubleshooting Page**
  By addressing the common issues faced, users will be able to find the relevant solutions more easily. Adding the solutions of commonly faced issues to the troubleshooting page can save time for users and provide a better user experience, as well as reduce the number of support queries faced by the developer team. This will save time also for developers as they will be less engaged in solving the issues faced by users and will help the development process.

  Let us take an example from the **kind/support** issues, we can see that there are many similar issues(#4200, #4292, #4830, #4892, #5744, #5765 and #6248) for setting up the basic auth to Prometheus and Alertmanager web pages. As users do not check for previously solved or discussed issues, they open up a duplicate issue and take up the time of maintainers. Having information specifically on the Troubleshooting Page can solve this problem.

  To implement this, first proper analysis of the common problems faced by users is needed and then proper summarization of these issues with solutions is needed in a comprehensive manner. To implement this, I will:

  1. Utilize the existing support issues (open as well as closed) and find the solutions to those problems.

2. Look for the most commented issues as a starting point as it has the highest frequency of being an issue faced by most of the users.
3. Highlight the crux of the issue with their solution and compile these issues concisely.
4. Thorough testing as well as feedback will also be needed to refine the troubleshooting page.

- **Improving the Contributing Page**
  As discussed above, there needs to be more detailed instructions for the contributors as to how they can contribute to a particular section and where and how the changes need to be made. We can use previously solved issues and use them as examples in the Contributing Page which makes it easier for contributors to contribute and reduce the workload of a maintainer. Mentioning the specific contribution details related to a particular CRD(like Alertmanager as seen above) in the Contributing Page will be much better than having a maintainer explaining what to do in that particular situation.

- **Addition of Blog Page**
  Integrating a blog page/section can be advantageous in many ways-

1. Can serve as a platform for addressing user queries, troubleshooting queries, and other problems not addressed in the core documentation.
2. Explore the topics that are not discussed in the core documentation. That may include discussing best practices, tutorials, industrial importance, use cases, etc.
3. To communicate about updates, new features, releases, announcements to the users.
4. To take feedback, suggestions and insights from the community.

Timeline

➢ **Week 1 - 2 : Project Planning**

Analyze the existing documentation for strengths, weaknesses, areas of interests.
Specifying the desired improvements.
Taking feedback from the team.

➢ **Week 3 - 4 : Remodeling and Establishing Guidelines**
Revaluate the structure and content of documentation.
Restructure the content into logical sections and subsections.
Define clear standards for formatting, language, structure and visual aid.
Include documentation tasks in the development process.

➢ **Week 5 - 6 : Testing and Solving Existing Documentation Issues**
Review the repository and make a priority order based on the importance of issues.
Take input and solve the issues accordingly.
Test the revised documentation for uniformity.

➢ **Week 7 - 8 : Compiling and Improvising Troubleshooting Page**
Gather the data from existing support issues and analyze it.
Summarize the issues with solutions and key points.
Incorporate the issues in the Troubleshooting page.
Conduct thorough testing on the Troubleshooting page.

➢ **Week 9 - 10 : Analysis for Version - Specific Documentation**
Assess the changes and updates in each version.
Focus on explaining new features, improvements and bug - fixes in different versions.
Add logs to explain the changes from the previous version in a concise manner.

➢ **Week 11 - 12 : Implementing Version - Specific Documentation**
Implement the changes and updates in the documentation.
Provide references or links across different versions for easy navigation.
Review documentation for each version.
Take feedback from developers, users for further improvements.

Additional Information

As a newcomer to cloud native computing, I wanted to try my hands in a tool that is of utmost importance. While searching, I found out about Prometheus Operator and found it to be very interesting. Even though I have been part of this community for less than a month, I have learnt many things which have increased my knowledge in the domain. I felt that contributing to this project would give me a chance to further expand my knowledge. I felt that the community here provides a lot of support and guidance. I believe I would be able to make a positive impact as a contributor.

At the start of the GSoC period, I will be having my end term examinations but after that, I will be on my summer vacation. So, I would mostly be available for the project. Since I don't have any other engagements, I would be able to give most of my time to the project. Personally, I will also look to study related to the devops field and would like to work on some personal projects.

After I complete my GSoC program, I would like to work on other projects in Prometheus Operator and continue to keep contributing to the organization. I believe that working in a project like this will provide me with various insights, new opportunities and industrial experience to deliver quality code. I hope to be a part of this dynamic team in the future and to keep learning whatever I can from all the members.