MacroPower Documentation

Release latest

Aug 20, 2020

Contents

Hello! If you're here, you're probably wondering how my README works. Well, you're in luck.

CHAPTER 1

Badges

- Views counter
- shields.io (everything else)

To add some extra flair, append some query parameters to your request to shields.io. All parameters I've used are well documented in the footer of shields.io.

CHAPTER 2

GitHub Stats

Generate the Stats card via GitHub Readme Stats.

Append `&theme=<your_theme>` with a theme from GitHub Readme Stats Themes.

CHAPTER 3

Dev Icons

All icons are from Devicon as well as cncf-artwork. I used the html table syntax to create the table.

```
<table id="macropower-tech">
  <!-- Row #1 -->
  <tr>
    <td align="center" width="96">
      <a href="#macropower-tech">
        <img src="icon.svg" width="48" height="48" alt="Tech Name" />
      </a>
      <br />Tech Name
    </td>
    <td align="center" width="96">
      {...}
    </td>
  </tr>
  <!-- Row #2 -->
  <tr>
    {...}
  </tr>
</table>
```

CHAPTER 4

Dynamic ASCII Graph

The ASCII Graph uses data from Wakatime. I have developed a set of simple tools that, when combined, can render the graph using this data. Each of the linked repos contains its own more detailed documentation.

4.1 Wakatime Exporter

Wakatime Exporter is a Prometheus Exporter that collects metrics from the Wakatime API on scrape. When added as a Prometheus target, each scrape adds coding metrics to Prometheus.

4.2 Prometheus ASCII

Prometheus ASCII is a Prometheus CLI Client that outputs ASCII graphs. By inputting a PromQL query for the Wakatime data collected via Wakatime Exporter, e.g. `increase(wakatime_seconds_total[24h])`, we can gather data and render the graph using my asciigraph fork, which adds support for bucketing by duration.

4.3 update_graph.sh

update_graph.sh is a very simple shell script that updates this repo's README.md, which is run locally on one of my servers (scheduled via cron). GitHub Actions can not be used unless you plan on exposing your Prometheus server to the internet (or perhaps you could get away with using some clever tunneling).

The only requirement is that you add the two comments to denote the area of README.md to modify. Any content between these comments will be replaced with the rendered graph each time the script is run.

The update_graph.sh script is wrapped with some bash that sets environment-specific things that would be not be appropriate to commit. Also, the script writes some files to the working directory, so you should run in some tmp directory. Example:

```
git config user.email "myemail+bot@gmail.com"
git config user.name "[bot] Username"

export PROMETHEUS_ASCII_SERVER_ENDPOINT="http://<my prometheus instance>:9090"

./update_graph.sh

git add README.md
git commit -m "Update graph : $(date)"

git push "https://<USERNAME:APIKEY>@github.com/<REPO>/<REPO>.git"
```