Pattoo Web Documentation

Peter Harrison

Sep 01, 2020

Introduction:

`pattoo` stores timeseries data in a database and makes it available for users via a GraphQL API. `pattoo-web` provides a web front end to access the data.

Visit the Pattoo Web GitHub site to see the code.

CHAPTER 1

Introduction

General information about the project, including the the prerequisite steps to get it operational on your system.

1.1 About Pattoo

`pattoo` allows you to use your web browser to chart your organization's constantly changing data.

It was inspired by the need to collect and visualize data from various DevOps, network, industrial PLC controllers, electro-mechanical and enterprise systems on a single web dashboard.

This data is collected by `pattoo` agents. There are standard agents for:

- Linux
- SNMP
- Modbus TCP
- Bacnet/IP
- OPC UA

With programming skill, you can create your own custom agents if needed.

1.1.1 Operational Overview

`pattoo` has a number of inter-related components. You can see how they all work together on the pattoo web page.

1.1.2 The Palisadoes Foundation

`pattoo` is based on the original `infoset` code created by the Palisadoes Foundation as part of its annual Calico Challenge program. Calico provides paid summer internships for Jamaican university students to work on selected open source projects. They are mentored by software professionals and receive stipends based on the completion of predefined milestones. Calico was started in 2015.

1.2 Basic Installation

This section covers some key steps to get you started.

1.2.1 Prerequisites

There are some software components that need to be installed prior to starting.

1. `pattoo` only runs on Python 3.6 or higher

Let's install the software.

1.2.2 Installation

Follow these steps.

1. Make sure you have a fully configured `` `pattoo` `` server as this is a `pattoo-web` pre-requisite.

Follow these steps.

1. Install `git` on your system.
2. Select and create the parent directory in which you want to install `pattoo-web`.

```
$ mkdir -p /installation/parent/directory
$ cd /installation/parent/directory
```

3. Clone the repository to the parent directory using the `git clone` command. You can also choose to downloading and unzip the file in the parent directory. The repository can be found at: https://github.com/PalisadoesFoundation/pattoo-web

```
$ cd /installation/parent/directory
$ git clone https://github.com/PalisadoesFoundation/pattoo-web.git
```

4. Enter the `/installation/parent/directory/pattoo-web` directory with the `pattoo-web` files.
5. Install the required packages using the `pip_requirements` document in the `pattoo-web` root directory

```
$ pip3 install --user --requirement pip_requirements.txt
```

6. Use the *Configuration Guide* to create a working configuration.
7. Run the installation script

```
$ setup/install.py
```

8. Start the `bin/pattoo_webd.py` daemon to accept data sent by `pattoo-agents`. *Configuration Guide*

1.2.3 Configuring systemd Daemons

You can also setup all the `pattoo-web` daemons as system daemons by executing the `setup/systemd/bin/install_systemd.py` script.

The script requires you to specify the following parameters. Make sure you have a username and group created for running your `pattoo-web` services.

```
usage: install_systemd.py [-h] -f CONFIG_DIR -i INSTALLATION_DIR -u USERNAME
                          -g GROUP

optional arguments:
  -h, --help            show this help message and exit
  -f CONFIG_DIR, --config_dir CONFIG_DIR
                        Directory where the pattoo configuration files will be
                        located
  -i INSTALLATION_DIR, --installation_dir INSTALLATION_DIR
                        Directory where the pattoo is installed. (Must end
                        with '/pattoo')
  -u USERNAME, --username USERNAME
                        Username that will run the daemon
  -g GROUP, --group GROUP
                        User group to which username belongs
```

Note The daemons are not enabled or started by default. You will have to do this separately using the `systemctl` command after running the script.

```
$ sudo setup/systemd/bin/install_systemd.py --config_dir=~/GitHub/pattoo/etc --user␣
↪pattoo --group pattoo --install ~/GitHub/pattoo

SUCCESS! You are now able to start/stop and enable/disable the following systemd␣
↪services:

pattoo_webd.service

$
```

1.2.4 Testing

You can test whether your `pattoo-web` site is operational by visiting http://pattoo.example.com:20200/pattoo where you substitute `pattoo.example.com` with the IP address or hostname of your server.

Use the *Troubleshooting Pattoo Agents* for further steps to take if you have difficulties.

1.3 Configuration Guide

After installation, you will need to create a configuration file in a directory dedicated to `pattoo`.

1.3.1 Setting the Configuration Directory Location

You must first set the location of the configuration directory by using the `PATTOO_CONFIGDIR` environmental variable. Here is how to do this from the Linux command line:

```
$ export PATTOO_CONFIGDIR=/path/to/configuration/directory
```

`pattoo` applications will read the configuration files located in this directory when `PATTOO_CONFIGDIR` is set.

You can automatically set this variable each time you log in by adding these lines to your `~/.bash_profile` file.

```
export PATTOO_CONFIGDIR=/path/to/configuration/directory
```

Make sure that files in this directory are readable by the user that will be running `pattoo` agent daemons or scripts.

1.3.2 Configuration Options

There are two ways to configure pattoo. These are the:

1. Quick Method
2. Expert Method

Quick Method

Use the quick method if you are new to pattoo.

Run the setup/configure.py script. It will prompt you for all configuration parameters. The defaults should be sufficient in most cases.

Here's the command to run:

```
setup/configure.py
```

Run the installation script next as outlined in the *Basic Installation* guide.

Expert Method

This section goes into configuration parameters in great detail.

Copy the Templates to Your Configuration Directory

Copy the template files in the examples/etc directory to the PATTOO_CONFIGDIR location.

NOTE: If a /path/to/configuration/directory/pattoo_web.yaml or /path/to/configuration/directory/pattoo.yaml file already exists in the directory then skip this step and edit the file according to the steps in following sections.

```
$ cp examples/etc/pattoo_web.yaml.template \
    /path/to/configuration/directory/pattoo_web.yaml

$ cp examples/etc/pattoo.yaml.template \
    /path/to/configuration/directory/pattoo.yaml
```

The next step is to edit the contents of both files.

Edit Your Configuration Files

The pattoo server uses two configuration files:

1. pattoo.yaml: Provides general configuration information for all pattoo related applications. pattoo.yaml also defines how pattoo agents should connect to the pattoo server APIs.
2. pattoo_webd.yaml: Provides configuration details for all the pattoo-web server's API daemons.

Take some time to read up on YAML formatted files if you are not familiar with them. A background knowledge is always helpful.

Edit Your pattoo Communication Configuration

The pattoo.yaml file created from the template will have sections that you will need to edit with custom values. Don't worry, these sections are easily identifiable as they all start with PATTOO_

NOTE: The indentations in the YAML configuration are important. Make sure indentations line up. Dashes '-' indicate one item in a list of items (if applicable).

```
pattoo:

    log_level: debug
    log_directory: PATTOO_LOG_DIRECTORY
    cache_directory: PATTOO_CACHE_DIRECTORY
    daemon_directory: PATTOO_DAEMON_DIRECTORY
    system_daemon_directory: PATTOO_SYSTEM_DAEMON_DIRECTORY
    language: en

pattoo_web_api:

   ip_address: 0.0.0.0
   ip_bind_port: 20200
```

pattoo Communication Configuration Explanation

This table outlines the purpose of each configuration parameter.

Section	Config Options	Description
pattoo		
	log_directory	Path to logging directory. Make sure the username running the daemons have RW access to files there.
	log_level	Default level of logging. debug is best for troubleshooting.
	cache_directory	Directory that will temporarily store data data from agents prior to be added to the pattoo database.
	daemon_directory	Directory used to store daemon related data that needs to be maintained between reboots
	system_daemon_directory	Directory used to store daemon related data that should be deleted between reboots. This should only be configured if you are running pattoo daemons as systemd daemons. The systemd daemon installation procedure automatically adjusts this configuration. This parameter defaults to the daemon_directory value if it is not configured.
	language	Language spoken by the human users of pattoo. Defaults to en (English)
pattoo_web_api		
	ip_address	IP address of the pattoo server to which the pattoo_webd daemon will use to transmit and receive data.
	ip_bind_port	TCP port of used by the pattoo server for this purpose

Edit Your Web Configuration

The pattoo_webd.yaml file created from the template will have sections that you will need to edit with custom values. Don't worry, these sections are easily identifiable as they all start with PATTOO_

NOTE: The indentations in the YAML configuration are important. Make sure indentations line up. Dashes '-' indicate one item in a list of items (if applicable).

```
pattoo_webd:

   ip_listen_address: 0.0.0.0
   ip_bind_port: 20200
```

Web Configuration Explanation

This table outlines the purpose of each configuration parameter.

Section	Config Options	Description
pattoo_webd		
	ip_listen_address	IP address used by the pattoo_webd daemon for accepting data from web browser users. Default of '0.0.0.0' which indicates listening on all available network interfaces. You can also use IPv6 nomenclature such as ::. The pattoo APIs don't support IPv6 and IPv4 at the same time.
	ip_bind_port	TCP port of used by the pattoo_webd daemon for accepting data from remote pattoo agents. Default of 20202.

1.4 Configuring systemd Daemons

You can also setup all the pattoo related daemons located in this GitHub repository as system daemons by executing the setup/systemd/bin/install_systemd.py script.

The script requires you to specify the following parameters. Make sure you have a username and group created for running your pattoo services.

```
usage: install_systemd.py [-h] -f CONFIG_DIR -u USERNAME -g GROUP

optional arguments:
  -h, --help            show this help message and exit
  -f CONFIG_DIR, --config_dir CONFIG_DIR
                        Directory where the pattoo configuration files will be located
  -u USERNAME, --username USERNAME
                        Username that will run the daemon
  -g GROUP, --group GROUP
                        User group to which username belongs
```

Note The daemons are not enabled or started by default. You will have to do this separately using the systemctl command after running the script.

```
$ sudo setup/systemd/bin/install_systemd.py --user pattoo --group pattoo --config_dir␣
↪/etc/pattoo

SUCCESS! You are now able to start/stop and enable/disable the following systemd␣
↪services:

pattoo_api_agentd.service
pattoo_apid.service
pattoo_ingesterd.service
```

(continues on next page)

(continued from previous page)

```
$
```

1.5 Pattoo Web API Daemon

`pattoo_webd` makes `pattoo` agent data available to web users.

1.5.1 Installation

Follow these steps.

1. Follow the installation steps in the *Basic Installation* file.
2. Configure the main section of the configuration file following the steps in *Configuration Guide* file.
3. Start the desired daemons using the commands below. You may want to make these `systemd` daemons, if so follow the steps in the *Basic Installation* file.

1.5.2 Usage

`pattoo_webd` has a simple command structure.

The daemon will require a configuration file in the `etc/` directory. See the configuration section for details.

```
$ bin/pattoo_webd.py --help
usage: pattoo_webd.py [-h] [--start] [--stop] [--status] [--restart]
                      [--force]

optional arguments:
  -h, --help  show this help message and exit
  --start     Start the agent daemon.
  --stop      Stop the agent daemon.
  --status    Get daemon daemon status.
  --restart   Restart the agent daemon.
  --force     Stops or restarts the agent daemon ungracefully when used with --stop or
              --restart.
$
```

1.5.3 Configuration

No additional configuration steps beyond that in the *Configuration Guide* file are required.

1.5.4 Testing

There are a number of steps you can take to make sure everything is OK.

1. If you have setup the daemon for `systemd` then you can use the `systemctl` command to get the status of the daemon.
2. The daemon should be running on the port configured with the `ip_bind_port` parameter. Use the `netstat` command to verify this.

3. Visit the URL `http://localhost:20202/pattoo/api/v1/web/status` to get the status page.
4. Use the *Troubleshooting Pattoo Agents* for further steps to take

1.6 Troubleshooting Pattoo Agents

Troubleshooting steps can be found in the PattooShared troubleshooting documentation

1.7 Backup and Restoration

Always take precautions. Backup your data as you'll never know when you'll need to restore it.

1.7.1 Backup

It is strongly advised that you backup your agents to protect you in the event of catastrophe.

The following directories need to be saved periodically.

1. The `PATTOO_CONFIGDIR` directory which contains your configuration
2. The `daemon_directory` location defined in your configuration. This area stores important authentication information.
3. The `pattoo-agents` directory which contains your source code.

We'll discuss data restoration next.

1.7.2 Restoration

It's important to follow these steps in this order when restoring `pattoo-agents` after a disaster.

1. FIRST make sure all the `pattoo` agents are stopped.
2. SECOND restore the contents of the `daemon_directory` location defined in your configuration. This area stores important authentication information.
3. Restore the `PATTOO_CONFIGDIR` directory which contains your configuration
4. Restore `pattoo-agents` directory which contains your source code.

You should now be able to restart your agents without issue.

1.8 Periodic Jobs

You will need to configure some jobs to improve `pattoo` performance and troubleshooting.

1.8.1 Logrotate Configuration

The default `pattoo` debug logging mode can quickly create large logging files. The logrotate utility can automatically compress and archive them.

1. Copy the the `examples/logrotate.d/pattoo` file to the `/etc/logrotate.d` directory.

2. Edit the file path accordingly.

Read up on the logrotate utility if you are not familiar with it. The documentation is easy to follow.

CHAPTER 2

Developers

2.1 How To Contribute

Start contributing today!

2.1.1 Introduction

Below is the workflow for having your contribution accepted into the `pattoo-web` repository.

1. Create an Issue or comment on an existing issue to discuss the feature
2. If the feature is approved, assign the issue to yourself
3. Fork the project
4. Clone the fork to your local machine
5. Add the original project as a remote (git remote add upstream https://github.com/PalisadoesFoundation/pattoo-web, check with: git remote -v)
6. Create a topic branch for your change (git checkout -b `BranchName`)
7. you may create additional branches if modifying multiple parts of the code
8. Write code and Commit your changes locally. An example of a proper `git commit` message can be seen below:

```
Make the example in CONTRIBUTING imperative and concrete ...

Without this patch applied the example commit message in the CONTRIBUTING
document is not a concrete example.  This is a problem because the
contributor is left to imagine what the commit message should look like
based on a description rather than an example.  This patch fixes the
problem by making the example concrete and imperative.
```

(continues on next page)

(continued from previous page)

```
The first line is a real life imperative statement with a ticket number
from our issue tracker.  The body describes the behavior without the␣
↪patch,
why this is a problem, and how the patch fixes the problem when applied.

Resolves Issue: #123
See also: #456, #789
```

9. When you need to synch with upstream (pull the latest changes from main repo into your current branch), do:
    1. `git fetch upstream`
    2. `git merge upstream/master`
10. Check for unnecessary white space with `git diff --check`.
11. Write the necessary unit tests for your changes.
12. Run all the tests to assure nothing else was accidentally broken
13. Push your changes to your forked repository (git push origin `branch`)
14. Perform a pull request on GitHub
15. Your code will be reviewed
16. If your code passes review, your pull request will be accepted

2.1.2 Code Style Guide

For ease of readability and maintainability code for all `pattoo` projects must follow these guidelines. Code that does not comply will not be added to the `master` branch.

1. All `pattoo` projects use the Google Python Style Guide for general style requirements
2. All `pattoo` python projects use the The Chromium Projects Python Style Guidelines for docstrings.
3. Indentations must be multiples of 4 blank spaces. No tabs.
4. All strings must be enclosed in single quotes
5. In addition too being `pylint` compliant, the code must be PEP8 and PEP257 compliant too.
6. There should be no trailing spaces in files

Guidelines to remember

- Always opt for the most pythonic solution to a problem
- Avoid applying idioms from other programming languages
- Import each module with its full path name. ie: from pack.subpack import module
- Use exceptions where appropriate
- Use doc strings
- Try not to have returns at multiple points in a function unless they are failure state returns.
- If you are in the middle of a development session and have to interrupt your work, it is a good idea to write a broken unit test about what you want to develop next. When coming back to work, you will have a pointer to where you were and get back on track faster.

Commits

The `pattoo` projects strive to maintain a proper log of development through well structured git commits. The links below offer insight and advice on the topic of commit messages:

1. https://robots.thoughtbot.com/5-useful-tips-for-a-better-commit-message
2. http://chris.beams.io/posts/git-commit/

Sample .vimrc File for Compliance

You can use this sample .vimrc file to help meet our style requirements

```
" Activate syntax
syntax on
" set number

" Disable automatic comment insertion
autocmd FileType * setlocal formatoptions-=c formatoptions-=r formatoptions-=o

" Delete trailing whitespace
autocmd BufWritePre * :%s/\s\+$//e

" Convert tabs to spaces
set expandtab

" Set tabs to 4 spaces
set tabstop=4

" Set the number of spaces for indentation
set shiftwidth=4

" Switch on highlighting the last used search pattern when the terminal has colors
if &t_Co > 2 || has("gui_running")
  set hlsearch
endif

" Tell vim to remember certain things when we exit
"  '10  :  marks will be remembered for up to 10 previously edited files
"  "100 :  will save up to 100 lines for each register
"  :20  :  up to 20 lines of command-line history will be remembered
"  %    :  saves and restores the buffer list
"  n... :  where to save the viminfo files
set viminfo='10,\"100,:20,%,n~/.viminfo

" Function for viminfo to work
function! ResCur()
  if line("'\"") <= line("$")
    normal! g`"
    return 1
  endif
endfunction

" Function for viminfo to work
augroup resCur
  autocmd!
  autocmd BufWinEnter * call ResCur()
augroup END
```