CERN Summer Student programme work report

Project ID: 13527

Adrian Alan Pol
adrian.pol@cern.ch

Supervisor: Adriana Telesca
adriana.telesca@cern.ch

ABSTRACT

My summer studentship at CERN lasted a total of 8 weeks from the 10th of June to the 2nd of August. During this time I was involved in several projects:

- my main official project, described in the summer student project list, which was *the automatic configuration of CERN's ALICE DAQ Zabbix monitoring tool*, described in section 1
- my additional official project,*Alice in Wonderland web socket*, described in section 2
- and a project I worked on during my free time, *CERN-maps for Android*, described in section 3

In the following sections I will explain in details the purpose and usage of each software package.

1. AUTO CONFIGURATION OF ZABBIX ALICE DAQ MONITORING TOOL

After the LHC long shutdown a new server performance monitoring will be deployed. The Zabbix[1] system was chosen to replace the current monitoring tool for the ALICE DAQ system. The monitoring system is being configured and deployed in the ALICE DAQ reference system before its installation in the production system at the LHC P2. Even though configuration tasks could be performed manually it is desirable to have an automatic tool in order to change the parameters simpler, on the fly and with possible usage of the cron jobs. The project covered the implementation of automatic tasks to configure the Zabbix system taking ALICE DAQ configuration database as a reference. The technologies used for the purpose of this project were the Zabbix 2.0 API[2], Python 2.6, MySQL, JSON.

[1] http://www.zabbix.com/

[2] https://www.zabbix.com/documentation/2.0/manual/appendix/api/api

Important note before usage: Please make sure that you have python installed together with additional modules eg. `mysql.connector` or `webbrowser`. Also global variables in `mail.py`, `methodsextended.py` and `zabbix-auto-config.py` files have to be correct.

The auto-configuration tool is a command line application, hence it is convenient to create a soft link to the python code in `/usr/bin/`. As I used `argparse`, the tool is used as a casual Linux command. The automatic configuration covers the graph generation and host update. For usage I would recommend to start with reading the –help.

```
$ zabbix-auto-confg --help

usage: zabbix-auto-confg [-h] [-t] [-w] [-q]
{graphhost,update,graphgroup} ...

Automatic configuration of CERN's ALICE DAQ
Zabbix monitoring tool.

positional arguments:
 {graphhost,update,graphgroup}
                commands
  graphhost     Creates graphs of discovered
                items that contain all the
                discovered items in one
                graph
  graphgroup    Creates graph with one item
                for all hosts belonging to
                the same host group
  update        Updates status of hosts

optional arguments:
 -h, --help     shows this help message and
                exits
 -t, --text     grabs names from command
                line
 -w, --wildcard specifies usage of wildcards
 -q, --quiet    invokes quiet mode, no
                prompting for decisions
```

Flags used for the application are `-q`, `-w` and `-t`. Without `-q` flag the tool will ask for any important decision such as deleting the host or displaying the graphs. `-w` flag lets the user insert group names with the use of the regular expressions, hence `zabbix-auto-confg -wt update -s *` will update the status of all the hosts in all the groups.

-t flag, as used above, lets user insert group names in command line, otherwise he has to specify path to file, which includes the groups' names.

There are three positional arguments. First one is the update, responsible for refreshing the information about the hosts taking ALICE DAQ configuration database as a reference:

```
$ zabbix-auto-confg update --help

usage: zabbix-auto-confg update [-h] [-s]
[-c] [-r] [--all] [groups]

positional arguments:
 groups        specifies the source file (or
               text, when [-t] is used) with
               the group entries; separators
               are <RETURN>, <,> or, <;>
optional arguments:
 -h, --help    shows this help message and
               exits
 -s, --status  updates status of hosts
 -c, --create  creates hosts if finds new
               entries in database
 -r, --role    updates the roles
 --all         do all actions: check the
               state, create and change
               host group for each host
```

-s flag checks the current status of the host (active/inactive) and applies changes in the Zabbix system. -c flag will create hosts if the tool finds them in reference database and it is not present in the Zabbix system already. Otherwise it will ask to run the tool with -r flag. This flag is responsible for linking and unlinking the servers from the groups. If the host is not present in reference database, the tool will unlink it from all of the groups but one (Zabbix is not allowing host to belong to no groups) and send an e-mail with notification specified in global variables in mail.py file.

The code understands the difference between the groups and detectors. To operate without errors, make sure, that the there is dash <-> between both eg. *GDC-HLT* in Zabbix system.

In case you are running the command with -all flag, it will perform creation, status and finally role update. This is the best order, that makes sure the update is full and code is optimized.

Important: Some of the groups should be excluded from search all the time eg. *Zabbix server*. They are specified in the global variables in `methodsextended.py` file.

Very important: Sometimes the Zabbix API does not perform the update action, although it returns the *success* message. In most cases running the command twice prevents from such problems. This should be taken into account when setting the cron job.

Finally the groups argument should be either text (when running with [-t] flag) or path to file. Please make sure that there is comma, semicolon or return sign between groups.

Examples of usage

Full update for groups included in file.txt:

```
zabbix-auto-confg update -all ./file.txt
```

Role update for GDC detectors groups (this will not update *GDC* group):

```
zabbix-auto-confg -wt update -r "GDC-*"
```

Full update for GDC and LDC groups:

```
zabbix-auto-confg -wt update -all "GDC*;LDC*"
```

Apart from updating the hosts, the zabbix-auto-config tool generates the graphs as well. The graphgroup and graphhost parameters are used in a similar way. The command needs two lists: one with groups and one with items. The last parameter is graph name (for graphgroup used as prefix in a "<prefix><item name> for group <group name>" way). In Zabbix the name of the graph has to be unique, hence the application will ask if the already existing graph should be deleted. As with update there is possibility to use the wildcards and include the text in command line instead of reading from the file.

```
$ zabbix-auto-confg graphhost --help

usage: zabbix-auto-confg graphhost [-h]
group items gname

positional arguments:
 group      specifies the source file (or
            text, when [-t] is used) with
            the group names; separators
            are <RETURN>, <,> or <;>
 items      specifies the source file (or
            text, when [-t] is used) with
            the group names; separators
            are <RETURN>, <,> or <;>
 gname      specifies name for the graph;
            note that the name must be
            unique for the every host in
            a group!

optional arguments:
 -h, --help  shows this help message and
             exits
```

The graphhost parameter creates graph for each host in the group with the items listed in the file. Thus if there are five hosts in group GDC, command zabbix-auto-confg -t graphhost GDC "item1, item2" graph1 will create five graphs and each will contain two items: item1 and item2.

The graphgroup parameter will always generate the same

amount of graphs as the number of supplied groups times number of items in each group. If there are still five hosts in GDC, command `zabbix-auto-confg -t graphgroup GDC "item1, item2"` will create two graphs, each with five datasets. However, it is possible have multiple items on the graph i.e. with the above example creating one graph with ten datasets. This can be achieved by running the command with wildcards, like `zabbix-auto-confg -wt graphgroup "GDC" "item*"`.

Summing up, the application has number of features that are listed below:

- automatic creation of host graph;
- automatic creation of group graph with one item;
- automatic creation of group graph with multiple items;
- automatic update of hosts with ALICE DAQ configuration database as a reference;
- command line input;
- argparse implemented;
- reading names from files (and checking file existence);
- reading names from command line;
- understanding <,> <;> and <RETURN> separators;
- discarding duplicates in supplied list of names;
- discarding empty values in supplied list of names;
- possibility of using wildcards in lists;
- if graph name exists, possibility of deleting and creating a new one;
- opening graphs in webbrowser after finishing job;
- memory taken into account while opening browser;
- prompting for decisions, understanding y or yes;
- handling the key_ problem (Zabbix issue) by prompting user for feedback;
- updating screens according to changes of graphs;
- auto-generating colors for graphs if needed;

2. ALICE IN WONDERLAND WEB SOCKET

This project involved prototyping a websocket-based tool to be used in the context of the dataflow monitoring project in the ALICE DAQ. The dataflow monitoring tool that will be developed during LS1 will need to display performance data that are collected by Zabbix and stored in the Zabbix database. The Alice in wonderland web socket package represents a proof of concept to understand if the exporting of Zabbix data to other visualization tool or a web browser is possible and efficient. The technologies used for the purpose of this project were Python, JavaScript, MySQL, JSON, jQuery, HTML.

The front-end is written in JavaScript. It uses the jqplot library for displaying the plots. This is very powerful and esthetic tool for presenting web-based graphs. Most important, all new versions of main browsers are compatible with it. This was crucial aspect as most stations at CERN use Internet Explorer or Mozilla Firefox.

The back-end is written in Python. It uses the `autobahn`[3] for creating the sockets. This package has to be installed (eg. with `easy_install`). The server uses JSON format to communicate with client. The connection is initialized by client and requests for data transfer are being sent by the client as well. Due to time constraints it was impossible to script one important part of the socket, the messaging queue handling. At the moment there is one MySQL query for each request.

3. CERNMAPS FOR ANDROID

At the beginning of my stay I tried the maps system provided by CERN. The paper maps are good, however not as usable as the electronic ones. The GIS department is providing only native iPhone/iPad application and a mobile website for other phones. I did not find the mobile website usable, hence I dedicated my free time in coding a native app for Android. Project is finished but not published. Currently I am in contact with representatives of the GS and IT departments in order to get more information on what I can publish and how. At the time of writing, these details are unknown. The technologies used for the purpose of this project were the Android API, Java, JSON, Bash, Python. Below the list of the app's features:

Figure 1: Screenshot from the application

- native app for Android;
- building search working with voice recognition;
- pinpoint user location (GPS)
- tram schedule at the CERN stop
- app is working without internet

[3] http://autobahn.ws/pythonlibrary

- Android UI guidelines are kept;
- usage of Open Street Maps;

The app was tested on low and high resolution screens (also on phones with weak computing power).