Budgie Documentation

Release 0.1.0-dev.0

Vahid Mardani

Oct 29, 2016

CONTENTS

- How can send an ssh command on windows? cygwin, or new ssh implementation from microsoft? at all, I assume all workstations are linux.
- Observation processes must be asynchronous. for example 20 workers will waiting for client's result at the same time, simultaneously.
- Because we are going to save the encryption key on the clients machine and everyone can access them, we should use asymmetric encryption. but at all, i cannot understand if, the connection is made by ssh between the client & server, why data encryption was requested? So I'll ignore the encryption phase, because I believe the all exchanged data between server & client is hardly encrypted using 4096 bits ssh key.
- The database object will be created using ORM in code-first manner.
- For the configuration I prefer to use `YAML` instead of the `XML`. it's so clear and readable. so my own open-source configuration library will be used.
- If time available in the future. the PSExec and or winexec will be tested for windows clients. A few years ago, I was used them for an ITIL(Help-Desk) project, to executing a command on the remote machine using Win-IPC, as the file & printer sharing is working. But it's very un-secure.
- Using github's kanban, to manage project.

CHAPTER

ONE

LIMITATIONS

- Only works with linux
- Require Python3.5 or higher
- Does not encrypting data from client script, because the data will be sent over the SSH

CHAPTER

TWO

DEPENDENCIES

The main dependencies of the project is listed below:

- https://github.com/pylover/pymlconf
- http://sqlalchemy.org/
- https://github.com/paramiko/paramiko

Requirements for running tests:

- https://github.com/carletes/mock-ssh-server
- http://nose.readthedocs.io/en/latest/

Requirements for building documents:

- sphinx
- texlive-latex-base
- texlive-latex-recommended
- texlive-latex-extra
- texlive-fonts-recommended

CHAPTER

THREE

API

API Reference

cli Module

The unique command line interface of this application.

```
usage: budgie [-h] [-c CONFIG_FILE] [-V] {setup-db,run} ...

positional arguments:
{setup-db,run}        Available commands:

optional arguments:
-h, --help            show this help message and exit
-c CONFIG_FILE, --config-file CONFIG_FILE
                      The server configuration file, default:
                      development.yaml
-V, --version         Show the version.
```

budgie.cli.**init**()

Initialize the CLI Arguments

Returns The parsed CLI arguments.

configuration Module

A high level and flexible YAML configuration library which provided by me as GPLv3 is used here.

The goal of this module is to allow the other modules import `settings` before initializing the configuration. A proxy is used to achieve this.

budgie.configuration.**init**(*config_file=None*, *context=None*, ***kw*)

Initialize the global configuration instance: `settings`.

Parameters

- **config_file** – A config file to merge with the instance.
- **context** – A dictionary containing pre-provided values, like `here`.
- **kw** – Additional key-value pairs to pass to the `pymlconf.ConfigManager` constructor.

models Module

All database stuff will be done here.

I use the code-first method, for using ORM's

class budgie.models.**AgentLog**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

The only model of this applcation. it holds the log of observation agent.

budgie.models.**create_database_objects**(*cli_arguments*)
Generates the database objects.

budgie.models.**init**()
Initialize the mapper and binding configurations.

observer Module

HelpdeskObserver

class budgie.observer.**HelpdeskObserver**(*max_clients=1000*)
Bases: object

The main object of the server. I uses an event-pool to execute jobs concurrently.

start()
Starts the workers.

worker Module

HelpDeskWorker

class budgie.worker.**HelpDeskWorker**(*username*, *host*, *port=22*, *key_file=None*, *password=None*, *agent_script=None*)
Bases: object

check_for_alerts(*alerts_config*, *target*)
Checking for reaching limitations and sending e-mail if any error or limitation reach is detected.

Parameters

- **alerts_config** – A list of alert config object.
- **target** – Destination email address.

smtp Module

SMTPClient

class budgie.smtp.**SMTPClient**(*startls=False*, *auth=False*)
Bases: object

The SMTP client to send email when any incident is detected.

send(*from_*, *to*, *subject*, *body*, *cc=None*, *bcc=None*)

Parameters

- **from** – From address
- **to** – Target address
- **subject** – E-mail subject.
- **body** – The email body in plain/unicode text.
- **cc** –
- **bcc** –

exceptions Module

exception budgie.observer.**MaximumClientsReached**(*maximum*)
Bases: Exception

Raised when maximum allowed clients to observation is reached.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

b

A

B

C

H

I

M

S