# Expyrimenter Documentation

***Release 0.1.0***

**Carlos Eduardo Moreira dos Santos**

August 06, 2015

Contents

Expyrimenter is an easy-to-use Python tool to help experiment automation. It makes it easy to execute local and remote commands in parallel. With Pushbullet, you can receive messages in your mobile phone if your experiment freezes or right after it finishes. Expyrimenter:

- Is a simple an easy way to run shell and SSH in Python
- Can run shell, SSH and Python code in parallel (it has been used in 200 VMs concurrently)
- Is flexible and extensible
- Has few configurations in one single file
- Compatible with Python >= 3.2 (Ubuntu >= 10.04)
- Is concerned about its source code:
  - Free software available at GiHub
  - Continuous integration
  - Code coverage target is 100%
  - No pep8 or flake8 issues

Contents:

CHAPTER 1

# Quick example

This example shows how to run SSH commands in serial and in parallel.

```
#!/usr/bin/env python3
from expyrimenter import SSH, Executor


# Suppose we have a cluster with hostnames vm0, vm1 and vm2
cluster = ['vm%d' % i for i in range(0, 3)]

# Let's run the command below in all VMs:
cmd = 'echo "$(date +%S) Hello by $(hostname)"'

# Blocking version, serial execution
print('Serial execution\n================')
for vm in cluster:
    output = SSH(vm, cmd, stdout=True).run()
    print(output)

# Create a pool for parallel execution
pool = Executor()

# Non-blocking version, parallel execution
print('\nParallel execution\n==================')
for vm in cluster:
    ssh = SSH(vm, cmd, stdout=True, stderr=False)
    pool.run(ssh)

# Block until all parallel calls are done
# and then print the results
pool.wait()
for result in pool.results:
    print(result)
```

Output (with all clocks synchronized):

```
Serial execution
================
58 Hello by vm0
00 Hello by vm1
01 Hello by vm2

Parallel execution
==================
```

```
02 Hello by vm2
02 Hello by vm0
02 Hello by vm1
```

CHAPTER 2

# Core

## 2.1 Config

Expyrimenter configuration uses ini files. The default one is shown below and can be overriden by a user configuration in `~/.expyrimenter/config.ini`.

```
[executor]
max_concurrency = 100

[pushbullet]
;token = YOUR_ACCESS_TOKEN

[cloudstack]
;url = https://example_cloud/client/api
;key = YOUR_KEY
;secret = YOUR_SECRET
```

**class** `expyrimenter.`**`Config`**(*section*)

`Config.user_ini` is the location of the user ini file and its default value is `~/.expyrimenter/config.ini`. If you change the user config location, it will take effect in every new object.

**`get`**(*key*, *default=None*)

**Parameters**

- **`key`** (*str*) – The key whose value you are looking for
- **`default`** (*any*) – value to return if key is not found

**Returns** *key* value

**Return type** str

## 2.2 Shell

This is one of the innermost classes. The parameters are passed to the constructor and then you can call *`run()`* or use an `Executor` instance to run in parallel.

```
>>> from expyrimenter import Shell
>>> Shell('echo Hello', stdout=True).run()
'Hello'
>>> # You can use try/except
```

```
>>> from subprocess import CalledProcessError
>>> try:
>>>     Shell('wrongcommand').run()
>>> except CalledProcessError as e:
>>>     print("Failed: %s" % e.output)
Failed: /bin/sh: 1: wrongcommand: not found
```

**class** `expyrimenter.`**`Shell`**(*cmd*, *title=None*, *stdout=False*, *stderr=True*)

Bases: `expyrimenter.runnable.Runnable`

**Parameters**

- **`cmd`** (*str*) – Command with arguments to be run.
- **`title`** (*str*) – A title to be displayed in log outputs. If None, `cmd` will be shown.
- **`stdout`** (*bool*) – Whether or not to display standard output. Default is *False*.
- **`stderr`** (*bool*) – Whether or not to display standard error. Default is *True*.

**`command`**

The command to be run in shell.

**Return type** str

**`has_failed`**()

Runs the command and returns whether the return code differs from 0.

**Returns** whether the command has failed.

**Return type** bool

**`run`**()

If the command exits 0, returns 0 or the stdout/stderr output. Otherwise, raises CalledProcessError.

**Returns** shell return code (0) or output.

**Return type** int or str

**Raises CalledProcessError** if return code is not 0.

**`title`**

If the title is not set, returns *`command()`*.

**Return type** str

**`was_successful`**()

Runs the command and returns whether the return code is 0.

**Returns** whether the command was successful.

**Return type** bool

## 2.3 SSH

Extends *`expyrimenter.Shell`* to run commands in a remote host. One parameter was added (the first one), which is the shell SSH arguments (e.g. `hostname`, `user@hostname`, `-p 2222 user@hostname`, etc).

```
>>> from expyrimenter import SSH
>>> SSH('localhost', 'echo Hello', stdout=True).run()
'Hello'
>>> # You can use try/except
```

```
>>> from subprocess import CalledProcessError
>>> try:
>>>     SSH('localhost','wrongcommand').run()
>>> except CalledProcessError as e:
>>>     print("Failed: %s" % e.output)
Failed: bash: wrongcommand: command not found
```

**class** expyrimenter.**SSH**(*params*, *remote_cmd*, *title=None*, *stdout=False*, *stderr=True*)

Bases: expyrimenter.shell.Shell

**Parameters**

- **params** (*str*) – shell SSH params (at least the hostname).
- **remote_cmd** (*str*) – Command to be run in remote host through SSH.
- **title** (*str*) – A title to be displayed in log outputs. If None, the shell command will be shown.
- **stdout** (*bool*) – Whether or not to display standard output. Default is *False*.
- **stderr** (*bool*) – Whether or not to display standard error. Default is *True*.

**static await_availability**(*params*, *interval=5*, *max_rand=1*)

Periodically tries SSH until it is successful. This function is very useful in cloud environmentes, because there can be a considerable amount of time after a VM is running and before SSH connections are available.

**Parameters**

- **params** (*str*) – shell SSH params (at least the hostname).
- **interval** (*num*) – Time in seconds to wait before new trial.
- **max_rand** (*num*) – A float random number between 0 and max_rand will be added to interval.

CHAPTER 3

# Pushbullet

Experiments may crash, so you probably watch their outputs every few minutes to make sure it is still running or restart it in the worst case. Now, there's no need to keep an eye on the output anymore! By using this code, you will receive a message in your mobile phone and desktop browser if a log file is not updated often enough. You can also send messages at any time, for example, when the experiment finishes.

The code below sends a message if a file is not updated in 5 minutes. The script will keep running until it sends a message or it is killed.

```
from expyrimenter.apps import Pushbullet

pb = Pushbullet()
pb.monitor_file('~/outputs/long_experiment.log', 5 * 60)
```

To send a message:

```
from expyrimenter.apps import Pushbullet

pb = Pushbullet()
# To send a message, inform title and an optional body
pb.send_note('Experiment finished!')
```

Before using it, you must install pushbullet mobile app and/or browser add-on. Then, go to the Pushbullet web site, access your account settings (last time, it was by clicking in the avatar), copy the Access Token and paste it in your `~/.expyrimenter/config.ini` like this (check *expyrimenter.Config* for more details about general configuration):

```
[pushbullet]
token = your_pushbullet_access_token
```

**class** expyrimenter.apps.**Pushbullet**

Requires `[pushbullet]` section with `token` value in config. Check *expyrimenter.Config* for more details about configuration.

**monitor_file**(*filename*, *max_mod_interval*, *title=None*, *body=None*)

Sends a message if *filename* is not modified within *max_mod_interval* seconds. When a message is sent, it stops monitoring the file. Otherwise, it keeps running until it is killed.

**Parameters**

- **filename** (*str*) – The filename to be monitored
- **max_mod_interval** (*int*) – If *filename* is not modified within *max_mod_interval* seconds, a message will be sent
- **title** (*str*) – Optional message title

- **body** (*str*) – Optional message body content

**send_note**(*title=''*, *body=''*)
Sends a message using pushbullet *note* type.

**Parameters**

- **title** (*str*) – Optional message title
- **body** (*str*) – Optional message body content

CHAPTER 4

# CloudStack

## 4.1 CloudStack API

**class** expyrimenter.clouds.cloudstack.**API**

## 4.2 CloudStack

**class** expyrimenter.clouds.cloudstack.**CloudStack**(*s*, *executor=None*, *api=None*)

CHAPTER 5

# Indices and tables

- genindex
- modindex
- search

## e

## A



## C



## E



## G



## H



## M



## P



## R



## S



## T



## W