オープンソースなシステム管理フレームワークFUNC

第4回KLab勉強会
(株)paperboy&co.
宮下　剛輔
http://mizzy.org/
2008/03/28

自己紹介

- (株)paperboy&co.（ペパボ）で働いてます
  - レンタルサーバ のロリポップ！の会社です
- 技術責任者というポジションで色々やってます
  - システムアーキテクト的なこととか
  - システム障害調査とか
  - ウェブアプリケーションやAPIの開発とか
  - 部署宛の郵便物を部員に配ったりとか
- ドクターペッパー大好きです


$5.87
$5.87
Shiner
Shiner
Shiner

Funcとは？

Funcとは？

- Fedora Unified Network Controller
- システム管理フレームワーク
  - と勝手に呼んでます
  - システム管理用のプログラムを開発するためのフレームワーク
- **gihyo.jpで連載中です**
- **You Can Do Almost Anything With It**
  - フレームワークなので「具体的な何かができる」というものではない
  - 逆にいえば何でもできます

Funcとは？

- 複数のサーバに対して，何らかの処理を一括でまとめて実行して結果が取得できる
- 「何らかの処理」の部分は，モジュールで拡張できる
- 「何らかの処理」はコマンドラインから実行して単に結果を表示，ということもできるし，Python API でプログラマブルに処理することもできる

サンプルその1

```
# func target.example.org
  call hardware info
# func "*.example.org" call
  yum update
# func "*" call
  moduleyouwrote methodname
  1234 5678 firetruck
  acmelabs hike!
```

サンプルその2

```
import func.overlord.client as fc
results =
  fc.Client("*").service.status("httpd
  ")

for (host, rc) in results.iteritems():
  if rc != 0:
  fc.Client(host).service.start("httpd
  ")
```

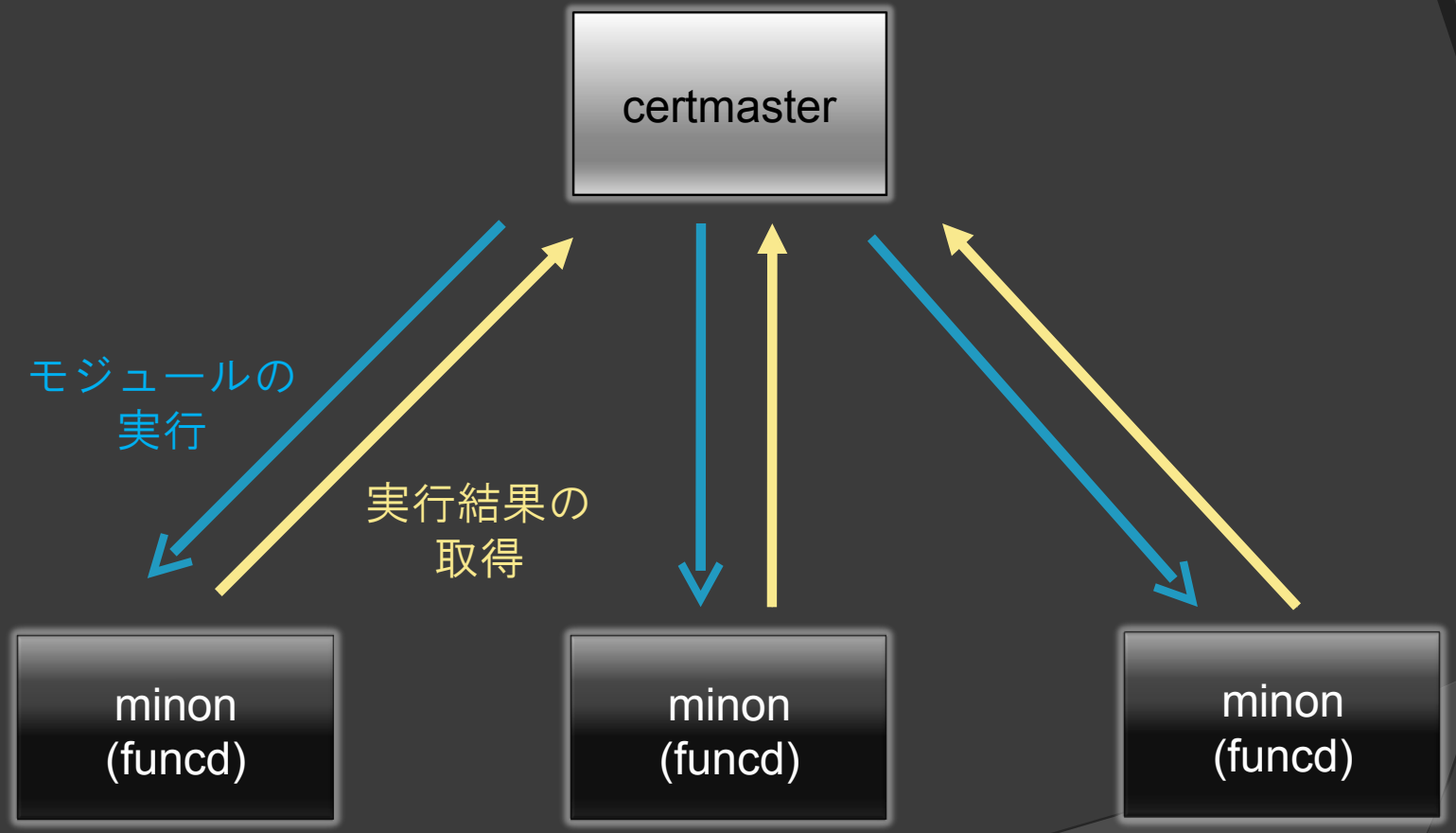
同じことをFuncなしでやろうとすると？

- 対象となるホスト情報はどうやって管理する？
- 対象ホストとどうやって通信して処理を実行する？SSH？HTTP？
- セキュリティはどうやって確保する？
- 実行結果の取得とパースはどうする？
- といったことを考えないといけません

Funcを使うと？

- 対象となるホスト情報はどうやって管理する？
  - Funcが情報持ってます
- 対象ホストとどうやって通信して処理を実行する？SSH？HTTP？
  - XMLRPC over HTTPS
  - 利用者は意識しなくて良い
- セキュリティはどうやって確保する？
  - SSL証明書による相互認証
- 実行結果の取得とパースはどうする？
  - Pythonの内部データ形式で結果取得

Funcのシステム構成



Funcが目指すもの

- 超わかりやすい
- 超シンプルに使える
- メンテと設定が最小限
- Kick ass

でも対応はRed Hat系Linuxのみ

- FC6以降、EL4以降
- 起動スクリプトがRed Hat 依存
- モジュールがRed Hat依存
- といっても、ピュアPythonなので、他のOSやディストリビューションへの対応はそれほど難しくなさそう
- 今後に期待

funcコマンド

管理対象minion一覧表示

```
# func “*” list_minions
['https://func02.example.org:51234',
  'https://func01.example.org:51234']
func02.example.org
func01.example.org
```

利用可能モジュール一覧表示

```
# func "func01*" call system list_modules
on https://func01.example.org:51234 running
   system list_modules ()
['certmaster', 'command', 'copyfile',
'filetracker', 'func_module', 'hardware',
'jobs', 'mount', 'nagios-check',
'netapp.snap', 'netapp.vol',
'netapp.vol.clone', 'networktest',
'process', 'reboot', 'rpms', 'service',
'smart', 'snmp', 'sysctl', 'test',
'yumcmd']
```

モジュールメソッドの表示

```
# func "func01*" call service
  list_methods
on https://func01.example.org:51234
  running service list_methods ()
['status', 'reload', 'get_running',
  'stop', 'start', 'inventory',
  'get_enabled', 'restart',
  'module_description',
  'module_version',
  'module_api_version', 'list_methods']
```

モジュール説明の表示

```
# func "func01*" call service
  module_description
on https://func01.example.org:51234
  running service module_description ()
Allows for service control via func.
```

ntpdの起動

```
# func "func01*" call service start
  ntpd
on https://func01.example.org:51234
  running service start (ntpd)
0
```

ntpdのステータス確認

```
# func "func01*" call service status
  ntpd
on https://func01.example.org:51234
  running service status (ntpd)
0
```

ntpdのステータス確認(停止してる場合)

```
# func "func01*" call service status
  ntpd
on https://func01.example.org:51234
  running service status (ntpd)
3
```

Python APIを利用したプログラミング

基本的なコード

```
import func.overlord.client as fc

results =
  fc.Client("*").module.method(args)

for ( host, result ) in
  results.iteritems():
    # 何か処理
```

virtモジュールの利用例

```
results = fc.Client("*").virt.state()

for ( host, vms ) in results.iteritems():
  if vms[0] == 'REMOTE_ERROR':
    continue

  for vm in vms:
    ( domain, state ) = vm.split(' ')
    if state == 'shutdown':
      fc.Client(host).virt.create(domain)
```

smartモジュールの利用例

```
info = fc.Client("*").smart.info()

for (host,details) in info.iteritems():
  if type(details) != list:
    print "%s had an error : %s" %
   (host,str(details))
    break

  (rc, list_of_output) = details

  if rc != 0:
    print "Host %s may have problems" % host
    print "\n".join(list_of_output[3:])
```

デフォルトで使えるモジュール

- command
- copyfile
- filetracker
- hardware
- jobs
- mount
- nagios-check
- netapp.options
- netapp.snap
- netapp.vol
- netapp.vol.clone
- networktest
- process
- reboot
- rpms
- service
- smart
- snmp
- sysctl
- test
- yumcmd

モジュールの拡張

モジュールのパスレイアウト

```
$PYTHONPATH/func/minion/modules/
  mymodule.py
(func “*” call mymodule method)

$PYTHONPATH/func/minion/modules/
  foo/
    __init__.py
    bar.py
(func “*” call foo.bar method)
```

モジュールコード

```
import func_module

class Test(func_module.FuncModule):

  version = "11.11.11"
  api_version = "0.0.1"
  description = "Just a very simple example module"

  def add(self, num1, num2):
    return num1 + num2

  def exception(self):
    raise exceptions.Exception("khhhhhhaaaaaan!!!!!!")
```

func-create-module

```
# func-create-module
Module Name: example
Description: An example module.
Author: Gosuke Miyashita
Email: gosukenator@gmail.com

Leave blank to finish.
Method: mymethod
Method:
Your module is ready to be hacked on. Wrote
  out to example.py.
```

func-create-module で生成された雛型

```
import func_module

class Example(func_module.FuncModule):

    # Update these if need be.
    version = "0.0.1"
    api_version = "0.0.1"
    description = "An example module."

    def mymethod(self):
        """
        TODO: Document me ...
        """
        pass
```

モジュールの配布

- やりかたわかりません＞＜
- copyfile使えばいい？

Funcの課題

Funcの課題

- 様々なOSへの対応
  - PuppetのようなOSの違いを吸収するための仕組みが必要
- モジュールの充実
  - モジュール自体も色んなOSへの対応が必要
- Func上で動作するアプリケーションの充実
  - func-inventoryぐらいしか今はない
- ホスト管理のバリエーション
  - DBとかYAMLとかLDAPとか

Q&A