Séminaire

22-23 Novembre 2006

Réalisation d'agents de surveillance Zephir

Sommaire

Introduction

Besoin de superviser les serveurs Eole :

-> utilisation d'agents de surveillance

-> remonter les informations

Introduction

Un agent :

-> une mesure à intervalle régulier

-> gestion d'un aspect particulier du serveur

Introduction

Publication des données :

- stockage local
  -> publication en local

- envoi des archives des mesures au serveur Zephir
  -> centralisation des informations

Ecriture d'un agent

Arborescence Standard :

Répertoire racine : /usr/share/eole/zephir/monitor/

Sous-répertoires :

- **agents** : fichiers Python contenant le code spécifique des agents
- **configs** : fichiers ".agent" employé pour activer l'agent
- **bin** : scripts shell spécifiques (cas particuliers)

Pour une configuration spécifique à un module, utiliser : configs/module

ex. : configs/amon-1.5

Ecriture d'un agent

Le service **z_stats** :

- charge les agents
- crée les archives
- publie les données localement (port 8090)
- amorce l'envoi des données à Zéphir

On le redémarre grâce à la commande :

```
/etc/init.d/z_stats restart
```

Ecriture d'un agent

Les agents sont des instances de classe :

-> création d'une classe spécifique par agent
-> héritage d'une meta-classe prédéfinie par le framework
- classe : Agent (mesures simples)
- classe : RRDAgent (génération de graphes)

Exemple de classe d'agent :

```
from zephir.monitor.agentmanager.agent import Agent

class Paquet(Agent):
```

Ecriture d'un agent

La classe contient au minimum 2 méthodes :

- measure() : appelée chaque fois que le framework déclenche une mesure

- check_status() : méthode d'évalution de l'état de l'agent (Ok, Warning, Error,...)

Ecriture d'un agent

La méthode measure() :

Elle doit retourner un dictionnaire : {'champ': valeur}

Exemple de mesure :

```
def measure(self):
    """ calcul de la version d'un module """
    out = file('/etc/eole/version')
    version = out.read()
    out.close()
    return {'value': version}
```

Ecriture d'un agent

Mesure avec utilisation de commandes externes :

Le framework utilise TwistedMatrix

-> obligation d'utiliser un appel non bloquant :

```
twisted.internet.utils.getProcessOutput()
```

-> retourne un objet de la classe Deferred
-> le traitement se fait uniquement dans la fonction définie par le CallBack
-> il faut retourner directement l'objet deferred

Ecriture d'un agent

Exemple d'utilisation de getProcessOutput() :

```
from twisted.internet.utils import getProcessOutput
```

```
def measure(self):
    # utilisation d'un appel non blocant
    measure = getProcessOutput('/bin/rpm',
                               args = ['-q', 'zephir-client'],
                               env = {'LC_ALL': 'C'} )
    # déclaration de la fonction de traitement
    measure.addCallback(self.parse_measure)
    # on renvoit directement le resultat
    return measure


def parse_measure(self, result):
    # result contient le résultat de notre commande
    # ici par exemple : "zephir-client-1.1-13eol\n"
    result = result.strip()
    return {'version': result}
```

Ecriture d'un agent

La méthode check_status() :

Retourne un état en fonction des données de l'agent
Les états possibles sont définis dans agentmanager/status.py et sont les suivants :

- status.OK()
- status.Warn()
- status.Error()
- status.Unknown()
- status.Dependant()

Ecriture d'un agent

Exemple de méthode check_status():

```
from zephir.monitor.agentmanager import status
```

```
  def check_status(self):
      if self.last_measure is None:
          # pas de mesure
          return status.Unknown()
      elif self.last_measure.value['version'] != 'zephir-client-1.1-14eol':
          # ancienne version
          return status.Warn("Vous n'avez pas la derniere version")
      else:
          # version OK
          return status.OK()
```

Ecriture d'un agent

Fichier de configuration de l'agent :

- crée une instance (ou plusieurs) de l'agent
- enregistre cette instance comme agent

Paramètres standards de l'instanciation :

- **name** : identification de l'agent
- **period** : durée en secondes de l'intervalle entre deux mesures
- **description** : libellé de l'instance d'agent

Ecriture d'un agent

Exemple de fichier de configuration :

```
# -*- mode: Python; coding: iso-8859-1 -*-
"""
Configuration de l'agent paquet
"""

# on importe la classe d'agent
from zephir.monitor.agents.paquet import Paquet

# on crée l'instance
p = Paquet('paquet',
           period=10,
           description='Version du paquet zephir-client'
          )

# on enregistre l'agent
AGENTS = [p]
```

Ecriture d'un agent

Prise en compte de l'agent :

* Si le fichier de configuration est présent dans "config" ou dans le sous-répertoire du module, les mesures sont réalisées

* il faut ajouter le nom de l'instance d'agent dans le fichier spécial
"**site.cfg**" pour qu'il apparaîsse dans l'interface Web.

Ecriture d'un agent

Test de l'exemple avec les fichiers :

- agents/paquet.py

et

- configs/paquet.agent

Ecriture d'un agent

Affichage des données dans la page Web de l'agent :

il faut utiliser des instances des classes :

- `HTMLData` sert à insérer du code Html directement dans la page
- `TableData` sert à présenter les mesures sous forme de tableau

puis les stocker sous forme de liste dans la variable : `self.data` au niveau de la méthode `__init()__`

Ecriture d'un agent

Exemple avec HTMLData et TableData :

```
from zephir.monitor.agentmanager.data import HTMLData,TableData

class Paquet(Agent):

    def __init__(self, name, **params):
        Agent.__init__(self, name, **params)
        self.last_measure = None

        # affichage du résultat
        titre = HTMLData("<h2>Exemple d'agent</h2>")
        self.table = TableData([
            ('version', 'Version', {'align':'center'}, None),
        ])
        self.data = [titre, self.table]
```

Ecriture d'un agent

Les données de notre TableData doivent être mises à jour par la méthode write_data() :

```
def write_data(self):
    Agent.write_data(self)
    if self.last_measure is not None:
        self.table.table_data = [self.last_measure.value]
```

Ecriture d'un agent

Test de l'exemple avec les fichiers :

- agents/paquet2.py

et

- configs/paquet2.agent

Ecriture d'un agent

Les agents RRD :

RRDtool (Round Robin Database tool) : est un outil de stockage et d'affichage de statistiques.

=> possibilité de générer des graphiques à partir de code Python
=> réalisation d'une classe "RRDAgent"

Ecriture d'un agent

Exemple d'agent RRD :

```
#!/usr/bin/env python
# -*- coding: iso-8859-1 -*-

from zephir.monitor.agentmanager.agent import RRDAgent
from random import randint

class Random(RRDAgent):

    def __init__(self, name, **params):
        RRDAgent.__init__(self, name, **params)
        self.last_measure = None

    def init_data(self, archive_dir):
        RRDAgent.init_data(self,archive_dir)

    def measure(self):
        return {'rand' : randint(0,10)}

    def write_data(self):
        RRDAgent.write_data(self)

    def save_measure(self, measure):
        RRDAgent.save_measure(self, measure)
```

Ecriture d'un agent

Configuration d'un agent RRD :

```
# -*- mode: Python; coding: iso-8859-1 -*-

from zephir.monitor.agents.rand import Random

# périodicité des mesures (secondes)
period = 30

r = Random( "rand", period=period,
            description = """Nombres aléatoires""",
            datasources = [{'name': "rand", 'min_bound': 0, 'max_bound': 10}],
            # sur 1 heures => toutes les 30 secondes
            # sur 24 heures => toutes les 15 minutes (900 secondes)
            archives = [{'rows':120,  'steps':1},
                        {'rows':24*4, 'steps':900/period}],
            graphs = [
              { 'pngname': "random.png",
                'vnamedefs': {"pan": ("rand", 'AVERAGE')},
                'options': ["-l0", "-s end-1hour", "-e now",
                            "-t Nombres aléatoires", "AREA:pan#FF8C00"] } ]
    )
AGENTS = [r]
```

Ecriture d'un agent

Test de l'exemple avec les fichiers :

- agents/rand.py

et

- configs/rand.agent

Merci de votre attention.