Python for System Admistration

Python brings the full convenience of a programming language to solve problems in system administration. We will look at some libraries in the `python standard library`.

By Michael Atambo

Interface with the OS:

These modules provide various interfaces to os resources, like file manipulation, searching, reading and writting, executing other programs, obtaining the ‘os’ state information

1. `"os" module`
2. `"subprocess" modules & psutil & stat`

Excercise on OS modules

In the provided code, please complete the provided code named 1_os* - 4_os* to do the following:

- 1_ospath.py : Print the absolute path to a file on the system (already done)
- 2_os_creating_dirs.py : This should create a directory with the name passed as a parameter
- 3_os_creating_files.py : same as above, for a normal file
- 4_os_search_dirs.py : This should search for a file in a location, both `file` and `location` are passed in as parameters

Provided code is at : Github repo: Link

Excercise fingerprinting an OS:

Use the `sys`, `psutil`, and (or) `platform` modules to characterise the systems specific information:

Complete these:

- 5_psutil.py : Use psutil to get these system parameters normally reported in ganglia:
- cpu_percent (per CPU), number of physical CPUs,
- available virtual memory, and total swap size
- print out all the disk partitions and their mountpoints, the disk space usage on the first partition reported
- 7_platfm.py : Use the platform module to find out this information on the system you are logged into:
- machine type (i386, x86_64), node name, processor , python_version, platform version,

Lower level access can be provided using pyudev

Standard Library utilities:

Exercise: Common compressed files can be handled by the tarfile and zipfile utilities:

Download the python source in zip and tar.gz format, we will use those here:

- `9_handle_tarfile.py`: List all the files, in the tar file, find the number of files
- `10_handle_zipfile.py`: Without extracting the zipfile, take out the installation instruction text file.

Log parsing:

Text processing in python is extensively supported, from reading and writting to perl compatible regular expressions, and dedicated modules.

12_parsing_logs.py : By reading the provided log file, find out the error the application is having.

virtualenv, pip,

A python virtualenv is an isolated copy of python on your system that gives you the ability to work with a particular environement without interference with the systemwide python installation. `Live Demo`: we will go through the process of setting up isolated environments.

- Creating isolated python environements with virtualenv.py

User Management:

User management can be achieved from the library libuser (in need root priviledges), we will skip this for now.

Ganglia module: Excercise

- Ganglia is used extensively to monitor distributed environments,
- It provides a way to extend the metrics that are collected using python
- The api consists of just about three methods that NEED to be implemented.
- Create a module that reports the:
- cpu temperature
- disk usage
- uptime
- logins

A refrence can be found here: ref

Backup and restore:

We always need backups, of configuration, user data... We can write modules to periodically take care of backups (compress if neccessary..) and restore them again. Or we can use readymade utilities like bakthat which will by default back up to s3.

Tools for managing machines:

These tools support command excution and automation as well as configuration management in the case of salt.

1. `Fabric`
2. `salt`
3. `ansible`

Fabric

What is Fabric? in the authors words:

> *Fabric is a Python (2.5-2.7) library and command-line tool for streamlining the use of SSH for application deployment or systems administration tasks.*

What can you do with it?

- Automating tasks
- Local and remote execution.

Fabric Excercise:

Starting simple: Helloworld! in Fabric: Place the following in a file named `fabfile.py`

```
def hello(name="world"):
    print("Hello world")
```

and run it with:

```
$ fab hello
 Hello world
```

Fabric will import the function and run the command you instruct.

Fabric Exercise II.

Create a file named `fabfile.py` and in it, create functions to

- Copy the `/var/log/messages` log to the current directory
- Querry and save the uptime in a file in the current directory
- do this for both the head and the slave nodes.