R for SysAdmins :D

Author: Kwan Lowe
Date: 2015.07.15

So yesterday I was tasked with getting some information from AWS for later analysis. AWS has a complete and easy-to-use command-line front-end that works well with whatever text processing tools are in your toolkit. I happen to be using R at the moment, and though it was not designed as a general-purpose language for system administration, it works really well for certain tasks such as when your input is any sort of formatted data and you need to automate a process and export it to a format consumable by non-Linux folks (e.g., Excel).

Now, one caveat: Everything I do here could as easily be done in other languages. The AWS CLI can output to text or JSON. It can filter and export in different ways. This output can then be parsed with awk, jq, Perl, Python or ruby using a multitude of libraries.

So let's try to parse some data in R.

In the AWS text format, the output looks something like below, which may be fine for reading but not so easy to machine parse:

```
TAGS    aws:autoscaling:groupName       DIGITALHERMIT-DEV-005
RESERVATIONS    810123459738            r-abcd1234
INSTANCES       0       x86_64  LqjVd1401234567890      False   xen     ami-6e09e006
i-045deb47      m2.xlarge       aki-08ed0761    RCCL    2014-05-20T05:11:54.000Z
ip-192-24-1-101.ec2.internal  192.22.1.101    None    /dev/sda1       ebs     True
None    subnet-df2c0599 paravirtual     vpc-24626b46
BLOCKDEVICEMAPPINGS     /dev/sda1
EBS     2014-05-20T05:11:58.000Z        True    attached        vol-22e21234
MONITORING      disabled
NETWORKINTERFACES       Primary network interface       eni-3a671234    81074942abcd
192.23.1.101    True    in-use  subnet-df2c1234 vpc-2462abcd
```

Since AWS can output to JSON which works well for this sort of data, I just let AWS do the work for me:

```
aws --output json ec2 describe-volumes --query 'Volumes' > /src/R/aws_volumes.json
```

For the purposes of your script you could opt to either save the file locally as we did here or retrieve it each time the script is run. Because my data doesn't change often I saved it. And normally if you're saving the file, Instead of filtering out the volumes at the source, you may opt to pull the entire output from "describe-volumes" and save that.

Like other languages, R has a rich library of functions. Half the job of writing scripts is knowing your library. Check out http://cran.r-project.org/web/packages/ for lots more. For this script I'll pull in the dplyr and jsonlite packages.

```
library(dplyr)
library(jsonlite)
```

Next, read in the output we saved earlier into a data frame, which is an internal data structure that R can access easily:

```
aws_volumes <- data.frame(fromJSON("aws_volumes.json"))
```

Here's where R pays off. We can now create a report based on whichever fields we need. In this case, we can pull the VolumeId, Size, VolumeType and CreateTime. These were needed to determine how much we were spending and how long ago we'd created them:

```
select(aws_volumes, VolumeId, Size,  VolumeType, CreateTime)
```

If I want to generate some summary metrics on the Volume size, it's as easy as:

```
summarize(aws_volumes, sum(Size))
summarize(aws_volumes, mean(Size))
```

And because the finance folks speak in the strange tongue of Spreadsheet, we can easily generate reports for them:

```
write.csv(select(aws_volumes, VolumeId, Size, VolumeType, CreateTime), file =
"outfile.csv" )
```

Or if they prefer, the Excel dialect of Spreadsheet:

```
library(xlsx)
output <- select(aws_volumes, VolumeId, Size, VolumeType, CreateTime)
write.xlsx(output, file = "awsVolumes.xlsx")
```

We could all assemble it into a script using the Rscript executable:

```
#!/usr/bin/Rscript

library(xlsx)
library(dplyr)
library(jsonlite)
aws_volumes <- data.frame(fromJSON("aws_volumes.json"))

output <- select(aws_volumes, VolumeId, Size, VolumeType, CreateTime)
write.xlsx(output, file = "awsVolumes.xlsx")
```

Though Python, Ruby, awk or other scripting languages may be more versatile, R makes quick work of manipulating many types of data. I've found it useful for processing sysstat output, running benchmark analyses, generating graphs, and even running anomaly detections.