Barcelona GSE Data Science Center

Introduction to Computing

September, 2018

Barcelona GSE Data Science Center

Goals

This module will provide the tools you need to start being productive using Linux servers.

You will:

- Launch a cloud virtual machine
- Connect via SSH.
- Use the command line
- Write a shell script
- Install programs
- Use Docker

TCP

TCP is a protocol, a “transport layer” representation that abstracts the two-way communication between two computers. It ensures all the “packets” get delivered in the proper order.

HTTP, the protocol of the web, is built on top of TCP.

SSH, “secure shell”, is also built on top of TCP .

SSH can be used to login to another computer and control it via a “shell”, particularly via the a CLI shell.

Barcelona GSE Data Science Center

Launching a Server

Let's launch a virtual machine on AWS.

We will use SSH to login and control the machine via a CLI.

We will run a Jupyter notebook on this server as well.

The Jupyter notebook creates a web server, which we access over HTTP.

So we will create two TCP "channels" to communicate with our virtual machine both via SSH and via HTTP.

Key pairs

The first TCP channel we will create is to SSH into the server.

Logging in via SSH requires a password or RSA keypair.

For AWS instances, you must use a keypair!

A keypair consists of a private key and a public key.

You can create this keypair or AWS can create this keypair.

Launching a Server

Demo:

(create a key)

(configure: specs, volumes, security groups)

(connect via SSH)

Shell

There are many shells in the world.

Luckily, most use similar commands.

“Bash” is a very common shell that is available in most Linux distros, as well as in Mac OSX.

All shell CLI’s come with builtin commands. For example:

```
which
pwd
cd
echo
```

Barcelona GSE Data Science Center

Navigating the File System

You can use `cd`, `ls`, and `pwd` to move and look around your file system!

The file system is a tree. The root is `/`.

Let's look around.

Practise:

(Navigating and tab-complete (creating, moving, copying, viewing files)

`ls`

`touch`

`mv`

`cp`

`cat`

`head`

`tail`

Barcelona GSE Data Science Center

$HOME

Home contains per-user settings and files.

It’s alias’d to ~ and available in environment variable: $HOME

Environment Variables

These are just variables, but they live in the “process” (the shell, or whatever is started from the shell)

`$HOME` is an example of an environment variable.

You can create environment variables, which can be a way to pass information to your program.

```
echo $USER
```

Environment Variables

Set an environment variable with the `export` command:

```
export FOO="Hello I'm Foo."
```

Users and Permissions

Unix systems have “users” which belong to “groups.”

All files and folders belong to a user and a group.

Your laptop, or this VM, might only have one human user. But there are still many users and groups to deal with the permissions that different applications have.

Inspecting File Ownership

`ls -l` will let you see the owner and group, as well as the "permissions" of the file.

All files have three permission types:

`r: read`
`w: write`
`x: execute`

For each of the following: "owner", "group", and "others".

Barcelona GSE Data Science Center

File Ownership

You can change the ownership of a file with `chown` and the permissions with `chmod`.

Let's play around with permissions and see how it effects us.

Programs

What if you want to create a program?

```
echo "date > foo" > write-date-to-foo.sh
chmod +x write-date-to-foo.sh
./write-date-to-foo.sh
```

You've created a program.

Barcelona GSE Data Science Center

Executables

Shell scripting is one way to create programs.

The other way is to create an executable (a.k.a. binary)!

An executable is machine code: instructions that your computer can run.

Machine code is created by compiling source code.

Source code is for humans, machine code is for machines.

Compilers are translators, human > machine!

Shell as a program

Shells, such as bash, are executables that runs on your computer!

```
stdin >  bash  > stdout
       ---^v---
       computer
```

Practise

For more practise with basic shell commands:

https://github.com/veltman/clmystery.git

Barcelona GSE Data Science Center

$PATH

Your contains an environment variable called PATH:

```
echo $PATH
```

PATH contains a list of paths, separated by colons:

```
usr/local/bin:/usr/bin:/bin
```

When you enter a command into your shell, (for example “python”) it will look for any executables that are in any of the folders listed in PATH.

Installing

Installing a program, therefore, is often nothing more than putting an executable in a folder that is listen in your PATH.

Instead of moving it to one of those folders, you could also create a symbolic link:

```
sudo ln -s /your/program/executable /usr/local/bin
```

Or, conversely, you could modify your PATH to add the folder that holds the executable:

```
export PATH="your/program/:$PATH"
```

Compiling from source

Demo:

(Installing various programs)

https://github.com/bgsedatascience/module-computing/tree/master/compiling