Debian

Monday, December 14, 2015 9:32 PM

Basic setup

Non-free repos (reference)

This will let you get things like Firefox instead of IceWeasel.

Other stuff

- Basic setup
  - apt-get update
  - apt-get upgrade
  - Things I install on practically every box:
    - sudo (run "$su" and *then* "$apt-get install sudo")
    - htop (or maybe "glances" if you want network utilization as well)
    - screen
    - curl
    - git
    - vim
    - unzip
    - caca-utils (reference) - this is useful for "cacaview" so that you can view pngs on the console
    - build-essential (which gives make, gcc, g++, etc.)
  - If SSH server isn't already running, then install it with this:
    - apt-get install openssh-server
  - apt-get install sudo (until this point, you'll have to do "su" to switch user to root)
    - After this, do "adduser adam sudo" to put your user into the sudoers group.
    - After modifying groups, you should run "newgrp sudo" to log you in to the new group. Alternatively, you could log out and log back in.
  - apt-get install htop
    - This will let you see a visual representation of how much memory and CPU you're using. My droplet at first was using 41/494 MB and ~0% CPU (which is good!).
  - apt-get install build-essential
    - This gives you everything needed to compile a Debian package: make, gcc, g++, etc
  - df -h - checks free space / disk usage
  - Put my bash_profile on the machine with colorize.py in /usr/local/bin (and chmod +x it)

To get PIP (for Python), do "sudo apt-get install python-pip".

- To forward ports on Debian, you need to modify iptables, which does seem to require a restart (sudo reboot)
  - ~~sudo iptables -t filter -A INPUT -p tcp --dport 3000 -j ACCEPT~~
    - Crossed this off because inputting this manually will get wiped when you reboot.
  - https://wiki.debian.org/iptables
    - This apparently isn't needed

    # Apply everything to the "filter" table. If I didn't put this, I think I would need "-t filter" in my individual rules.
    *filter

    # Append to the INPUT chain
    # -p (protocol) is TCP

```
# Destination port is 3000
# "-j" (jump) - what to do if packet matches this rule
-A INPUT -p tcp --dport 3000 -j ACCEPT

COMMIT
```

Basic commands

Version information

- Check Debian kernel version: uname -a
- less /etc/apt/sources.list
- su - switch user (when none is specified, you'll switch to root)
  - FYI: this probably does not perform a login, meaning if you switch users, you will need to source your shell's profile (e.g. source ~/.bash_profile) or just run your shell again (e.g. bash).
- Stop or disable Gnome
  - sudo systemctl disable gdm (just removes it from autostart)
  - sudo systemctl stop gdm (stops it in the current session)

User management

Deleting a user and their home directory all at once

sudo adduser bob
sudo userdel -r bob

Make it so that "su" doesn't require you to type in a password (reference)

- Open visudo
  - sudo visudo
- Add this line to the end
  - adam ALL=(ALL) NOPASSWD:ALL

Installing NodeJS (reference) (manual instructions here)

Note: if you were to install via apt-get, you would get a super out-of-date version (e.g. 0.10.x).

First, make sure you have "curl", and if not, "sudo apt-get install -y curl".

```
curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
sudo apt-get install -y nodejs
```

PATH (and other environment variables) (reference)

For per-user settings, edit ~/.profile (and put something like export PATH=$PATH:<something else>). For global settings, edit /etc/profile

How to make something run at startup (or shutdown)

Systemd is the preferred Debian way of doing things. One thing that's nice about it for running scripts on startup is that the script itself doesn't have to change. With init.d, I had to modify the comments of the script so that they matched a certain expected header/preamble.

- systemd method (references: 1 2)
  - Make phonehome.service: (note: original file is here: D:\Code\BotLand\botland\packages\ansible\roles\gameservers\files\phonehome.service)
    - sudo vim /etc/systemd/system/phonehome.service

```
[Unit]
Description=Phone home to Overseer
After=ssh.service
Requires=ssh.service

[Service]
Type=oneshot
ExecStart=/usr/local/bin/phonehome

[Install]
WantedBy=multi-user.target
```

        - Note: "requires=" does not imply "after=", so you'll need them both if you want to run after.
        - Note: the text bolded above refers to the startup script that I want to run (that I define below). This is "Type=oneshot" because I don't need a service staying around afterward.
    - sudo vim /usr/local/bin/phonehome
        - Type your script out here (it comes from D:\Code\BotLand\botland\packages\ansible\roles\gameservers\files\phonehome.sh)
    - sudo chmod +x /usr/local/bin/phonehome
    - Test it out first:
        - If you reloaded the file, you'll need to run
            - sudo systemctl daemon-reload
        - sudo systemctl start phonehome
    - Once everything is working, enable the service at startup time by doing
        - sudo systemctl enable phonehome
- init.d method (outdated perhaps) (reference)
    - sudo vim /etc/init.d/test.sh
        - Note: the script doesn't have to live in /etc/init.d, but at least a symlink has to.
    - There is a preamble needed according to the "LSB tags" message that prints out (reference). This preamble can be found in /etc/init.d/skeleton. My final script looked like this:

```
#! /bin/sh
# /etc/init.d/test.sh
#
### BEGIN INIT INFO
# Provides:          skeleton
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Example initscript
# Description:       This file should be used to construct scripts to be
#                    placed in /etc/init.d.
### END INIT INFO

# Some things that run always could go here

# Carry out specific functions when asked to by the system
case "$1" in
  start)
    echo hi > /home/adam/hello_world.txt
    ;;
  stop)
    ;;
  *)
```

```
        echo "Usage: /etc/init.d/blah {start|stop}"
        exit 1
        ;;
    esac

    exit 0
```

- sudo chmod +x /etc/init.d/test.sh
- sudo update-rc.d test.sh defaults
  - Note: you don't need to "/etc/init.d" on the "test.sh" above because it is assumed
- Test it out with "sudo reboot" if you're able to.

Networking (TCP)

Reference

- Update SYN retries - you may want to do this when you expect TCP to timeout after a shorter amount of time than 127 seconds when the other endpoint isn't even online.
  - First, you can see what the current value is by doing this command
    - cat /proc/sys/net/ipv4/tcp_syn_retries
  - Modify /etc/sysctl.d/99-sysctl.conf
    - Type this as the first line:
      - net.ipv4.tcp_syn_retries=1
    - You don't want to set that to 0 since this is OS-wide.
  - sysctl --system
    - This will reload all of the sysctl files.

tcp_syn_retries - INTEGER
Number of times initial SYNs for an active TCP connection attempt
will be retransmitted. Should not be higher than 127. Default value
is 6, which corresponds to 63seconds till the last retransmission
with the current initial RTO of 1second. With this the final timeout
for an active TCP connection attempt will happen after 127seconds.

tcp_retries1 - INTEGER
This value influences the time, after which TCP decides, that
something is wrong due to unacknowledged RTO retransmissions,
and reports this suspicion to the network layer.
See tcp_retries2 for more details.

RFC 1122 recommends at least 3 retransmissions, which is the
default.

tcp_retries2 - INTEGER
This value influences the timeout of an alive TCP connection,
when RTO retransmissions remain unacknowledged.
Given a value of N, a hypothetical TCP connection following
exponential backoff with an initial RTO of TCP_RTO_MIN would
retransmit N times before killing the connection at the (N+1)th RTO.

The default value of 15 yields a hypothetical timeout of 924.6
seconds and is a lower bound for the effective timeout.
TCP will effectively time out at the first RTO which exceeds the

hypothetical timeout.

RFC 1122 recommends at least 100 seconds for the timeout,
which corresponds to a value of at least 8.

Setting up SSH

- On the client, use "ssh-keygen" to generate a keypair. Ideally you should call it "id_rsa", which means don't change the name of the key (although it's okay if you do; you'll just need to specify the "-i" argument for all commands later)
- Copy the public certificate as a file or onto your clipboard.
- On the server, append the public key's text to ~/.ssh/authorized_keys (or create the file if it doesn't exist):
  - cat **public key text** >> ~/.ssh/authorized_keys
    - Warning: make SURE you have two angle brackets, otherwise you may overwrite the file by only using one.
  - Note: you can do this step more easily by using "ssh-copy-id", which will SSH into the server and copy your specified key (or id_rsa.pub) into the server's authorized_keys
- On the client, you should now be able to do this:
  - ssh -i <path to private key> user@host
  - e.g. ssh -i ~/.ssh/test_key adam@192.168.1.26

If you want to be able to SSH without specifying your key location every time, then either name it "id_rsa" and put it in "~/.ssh/" or set up an agent:

- ssh-agent bash <-- this launches bash underneath the ssh-agent
- ssh-add <path to private key>
- You're good to SSH into things just via "ssh user@host"
- Alternatively, you can specify your private key location in ~/.ssh/config

Not showing "Last login" message

Typically, when you log in via SSH, you'll see a message like this:

Last login: Wed Feb 15 08:46:20 2017 from 192.168.1.12

When streaming, this shows my public IP address when logging into cloud machines like AWS or DigitalOcean, so I wanted a way to disable it. It's actually pretty easy:

1. sudo vim /etc/ssh/sshd_config
2. Find a line that says "PrintLastLog yes"
3. Change it to "PrintLastLog no"
4. Restart the SSHD service
   a. sudo service sshd reload

Two factor authentication with SSH (reference)

15:24 HiDeoo: Adam13531 https://www.sysorchestra.com/2017/11/24/add-two-factor-authentication-to-ssh-on-debian-wheezy-and-jessie/
15:28 hoeindoe: Just make sure your server has a ntp service running to keep time in sync. or you can get locked out :)

Setting up FTP/SFTP

Note: you should not use regular FTP on any box that is open to the Internet.

Just do "sudo apt-get install vsftpd"

After doing that, you should be able to FTP (*not* SFTP) normally with just host/user/password/port=21.

If you want to use SFTP, then read this link.

Checking bandwidth usage

If it's for a single session, you can just check "ifconfig" under RX bytes. This may only keep track up to a certain number of GB though. Alternatively, use nload or munin or iftop.

sudo apt-get install nload

Then simply run "nload" and you'll see a nice graph any time network traffic occurs.

Setting up NTP (network time protocol)

For setting it up outside of AWS, look at these instructions. However, for AWS, follow the Ubuntu instructions on this page (and note: I first uninstalled ntp by doing "sudo apt-get remove ntp").