Báo cáo bổ sung phần linux

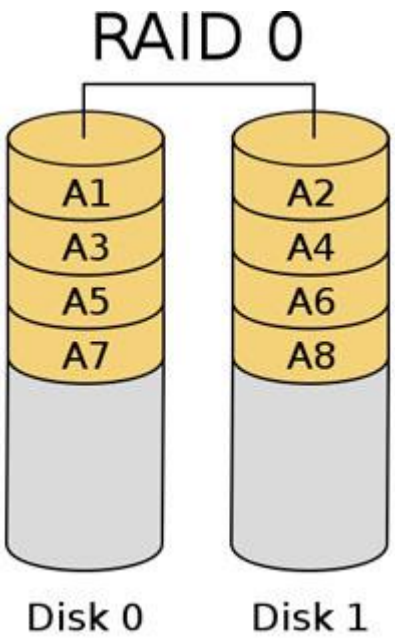
Mục lục

1. RAID....2
   1.1. Cấu hình RAID 0....2
   1.2. Cấu hình RAID 1....3
   1.3. Cấu hình RAID 1-0....4
   1.4. Cấu hình RAID 5....4
2. Sự khác nhau giữa lệnh zip và gzip trong linux?....5
3. File ẩn là gì? Sao lại cần file ẩn....6
4. Quyền trong linux, lệnh chmod....6
5. Lệnh ping, traceroute....8
   5.1. Ping....8
   5.2. Traceroute....8
6. Lệnh route add, del,… trong server linux, tĩnh và động....9
7. Khai báo IP tables, sau khi khai báo xong sx bật lên cùng OS....11
8. Nguyên lí hoạt động của tiến trình....14
9. Các thông tin có trong lệnh top....16
10. Lệnh awk....18
11. Crontab – lặp lịch....19
12. Lệnh scp trong linux....21
13. Ansible....22
    13.1. Inventory....24
    13.2. Playbook....25
    13.3. Loop....26
    13.4. Roles....27
14. Bài tập....28

1. RAID

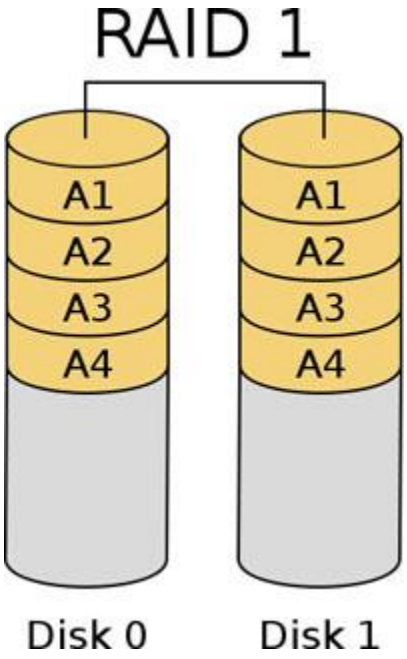
RAID là chữ viết tắt của **Redundant Array of Independent Disks**. Ban đầu, RAID được sử dụng như một giải pháp phòng hộ vì nó cho phép ghi dữ liệu lên nhiều đĩa cứng cùng lúc. Về sau, RAID đã có nhiều biến thể cho phép không chỉ đảm bảo an toàn dữ liệu mà còn giúp gia tăng đáng kể tốc độ truy xuất dữ liệu từ đĩa cứng.

1.1. Cấu hình RAID 0



RAID 0 có cơ chế chia đều các mảnh dữ liệu nhỏ cho các đĩa để tăng tốc độ đọc ổ cứng. Đơn giả hơn là ta có 100MB dữ liệu thì ta cần chia đôi, mỗi ổ 50MB dữ liệu. Ta sẽ sử dụng được dữ liệu của cả 2 ổ và tổng dung lượng sẽ là dung lượng của cả 2 đĩa. Nếu có n đĩa thì ta sẽ chia nhỏ dữ liệu và xếp lần lượt lên các ổ đĩa và dung lượng tổng sẽ là dung lượng của ổ nhỏ nhất * n. Do vậy ta cần tối thiểu 2 ổ cứng để triển khai cấu hình này. Vì được đọc dữ liệu trên nhiều ổ cứng nên tốc độ load dữ liệu sẽ gia tăng đáng kể. Tuy nhiên vấn đề sẽ xảy ra nếu 1 ổ đĩa bị hỏng. Do chia nhỏ dữ liệu nên khi 1 ổ đĩa bị hỏng sẽ kéo theo việc dữ liệu sẽ bị thiếu sót và diễn ra việc mất dữ liệu. Đây là một điều khó có thể chấp nhận đươc.

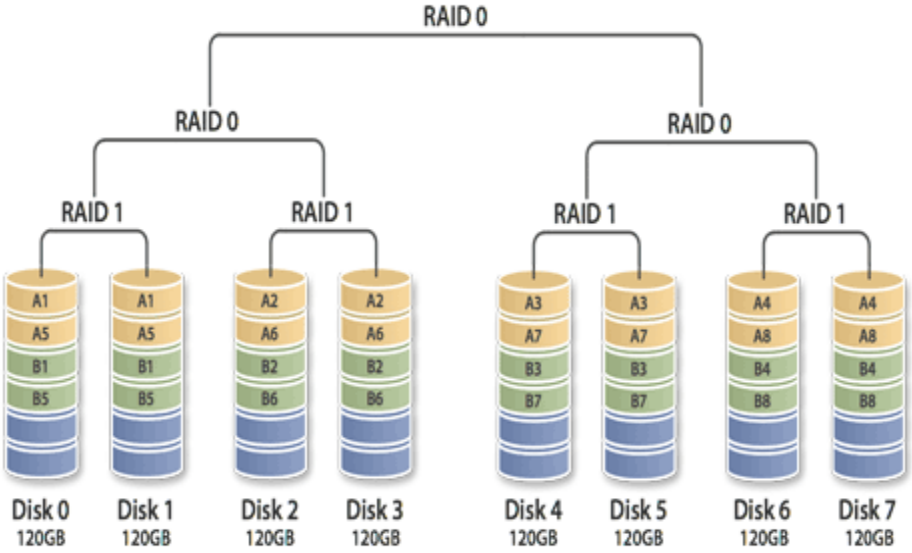
1.2. Cấu hình RAID 1



Đây là cấu hình RAID 1. RAID 1 sử dụng tối thiểu 2 ổ đĩa cứng và dữ liệu của 2 bên ổ cứng là giống hệt nhau. Cần tối thiểu 2 ổ để cài đặt cấu hình RAID 1.

- Ưu: Điều này sẽ làm cho RAID 1 rất tốt trong việc bảo vệ dữ liệu. Nếu 1 ổ bị hỏng thì ổ còn lại vẫn hoạt động bình thường, ta có thể thay thế ổ khác mà không cần quan tâm đến vấn đề mất mát dữ liệu.
- Nhược điểm : chậm hơn so với RAID 0 do việc không được đọc dữ liệu trên nhiều ổ. Ngoài ra, ta chỉ có thể sử dụng dung lượng 50% của tổng dung lượng của các ổ đĩa

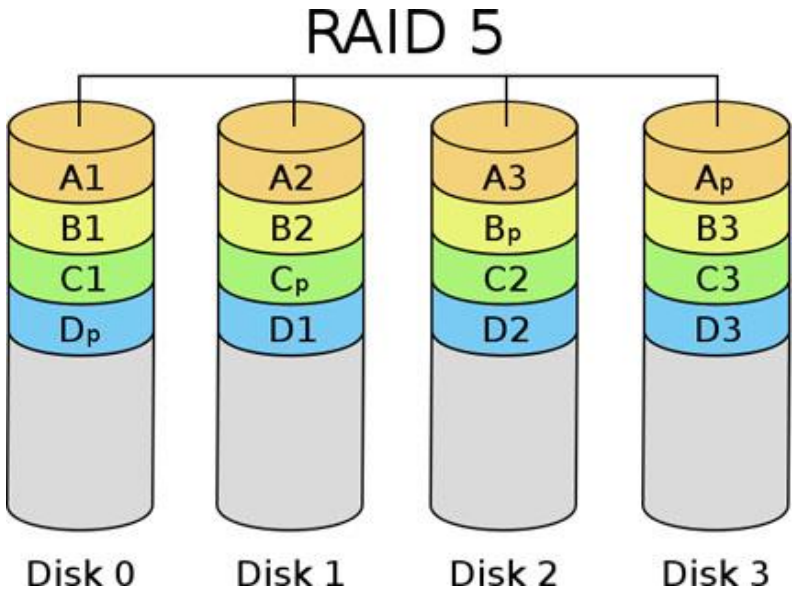
1.3. Cấu hình RAID 1-0



Đây là kiểu cấu hình kết hợp cả RAID 0 và RAID 1. Ta sử dụng RAID 0 để đọc nhanh dữ liệu từ ổ đĩa và RAID 1 để tăng tính toàn vẹn dữ liệu. Ta cần tối thiểu 4 ổ đĩa để cài đặt được cấu hình RAID 1-0

- Ưu điểm: tốc độ đọc dữ liệu nhanh hơn do dữ liệu được chia nhỏ lên nhiều ổ đĩa. Ngoài ra, nhờ có dự phòng nên dữ liệu được an toàn và toàn vẹn hơn.
- Nhược điểm: chỉ sử dụng được 50% dung lượng của bộ nhớ

1.4. Cấu hình RAID 5



Cấu hình RAID 5 cũng gần giống so với cấu hình RAID 1 và RAID 0, tức là có thể tách ra lưu trữ các ổ cứng riêng biệt và vẫn có phương án dự phòng khi sự cố phát sinh. Nhưng RAID 5 hơi khác một chút so với 2 cái trên. Tối thiểu cần 3 ổ đĩa để cấu hình RAID 5. Cơ chế của nó như sau:

- Ta sẽ chia nhỏ dữ liệu ra và sắp xếp nó vào các ổ giống như RAID 0 tuy nhiên ta sẽ có 1 phần dự phòng cho những dữ liệu vừa được ghi. Ví dụ như trong hình, ta ghi dữ liệu A1,A2,A3 vào các ổ và ghi dự phòng Ap cho 3 mảnh dữ liệu trước A1,A2,A3 vào ổ đĩa còn lại. Tương tự ta ghi dữ liệu B1, B2, B3 và ghi dự phòng Bp cho các mảnh B1, B2, B3 đã lưu vào ổ còn lại
- Vậy tại sao 1 dự ô dự phòng lại có thể dự phòng cho 3 mảnh. Các mảnh dữ liệu khi được lưu sẽ chuyển sang dạng nhị phân như 00110101. Ta có 3 mảnh A1 đổi thành 1001, A2 đổi thành 0101, A3 đổi thành 0011. Ta có một số thuật toán để tạo ra Ap như ta thực hiện phép toán AND các bit trong chuỗi bit. Ap sẽ bằng 1111. Giả dụ nếu mất 1 ổ dữ liệu, ta có thể suy ra được mảnh dữ liệu bị mất từ dự phòng Ap và ta vẫn bảo toàn được tính toàn vẹn của dữ liệu. Nhưng khi 2 ổ đĩa bị lỗi cùng 1 lúc thì RAID 5 cũng không thể làm gì được để bảo toàn tính toàn vẹn dữ liệu

2. Sự khác nhau giữa lệnh zip và gzip trong linux?

Zip và gzip là 2 lệnh dùng để nén tập tin hoặc thư mục trong linux. Tuy nhiên có những điểm khác nhau giữa 2 lệnh trên như sau:

- Zip: có khả năng nén kết hợp, có thể nén nhiều file và thư mục lại thành 1 file nén .zip duy nhất. Nó cũng tự động nén các nội dung bên trong thư mục nén. Zip nén mà không cần sử dụng thêm các công cụ hỗ trợ nào nữa cả
- Gzip: chỉ nén nội dung của từng file riêng lẻ, không kết hợp nhiều file thành một. Khi cần nén nhiều file thành một file duy nhất, ta cần sử dụng công cụ TAR để gom nhiều file, thư mục thành 1 file .tar sau đó mới bắt đầu nén bằng gzip. Đuôi của file nén sẽ có tên là .gz hoặc .tar.gz nếu ta sử dụng cả những công cụ TAR để gom file, thư mục

Ví dụ về lệnh zip:

- Tạo tệp nén: **zip archive.zip file1 file2 folder1**

- Giải nén tệp: **unzip archive.zip**
- Thêm tệp vào tệp nén hiện tại: **zip archive.zip newfile**
- Xóa tệp từ tệp nén: **zip -d archive.zip file1**

Ví dụ về gzip:

- Nén tệp: **gzip file**
- Giải nén tệp: **gunzip file.gz** hoặc **gzip -d file.gz**
- Đổi tên tệp nén: **gzip -c file > file.gz**
- Nén tệp và giữ tệp gốc: **gzip -k file**

3. File ẩn là gì? Sao lại cần file ẩn

File ẩn trong linux là các file có tên bắt đầu bằng dấu (.) ví dụ như file **.config**, **.bashrc**, **.gitignore**

Các file này thường không hiển thị khi ta sử dụng lệnh ls thông thường mà cần phải thêm option –a vào mới hiện lên. File ẩn thường được sử dụng để lưu trữ các tệp cấu hình và dữ liệu liên quan đến hệ thống hoặc chương trình, nhưng không cần hiển thị cho người dùng bởi một số lí do sau:

- Giảm sự lộn xộn trong hệ thống: giấu đi các file này sẽ làm cho giao diện người dùng trở nên sạch sẽ và tập trung hơn vào các tệp mà người dùng sẽ trực tiếp làm việc
- Bảo vệ các file cấu hình quan trọng: các file ẩn chứa các cài đặt quan trọng cho hệ điều hành cà ứng dụng. Bằng cách ẩn chúng đi, người dùng khả năng người dùng vô tình xóa hoặc chỉnh sửa các file cấu hình sẽ giảm đi, giúp an toàn hệ thống
- Lưu giữ thông tin cá nhân và cấu hình người dùng: có những file chứa thông tin riêng của từng cá nhân và bảo mật không được cho người khác nhìn nên cần ẩn đi

4. Quyền trong linux, lệnh chmod

Mỗi tập tin có 3 quyền cơ bản:

- Read (quyền đươc đọc file, kí hiệu là r)
- Write (quyền được ghi file, kí hiệu là w)
- Execute (quyền thực thi file, kí hiệu là x)

Nếu quyền theo người dùng thì ta có 3 quyền truy cập như sau:

- Owner (người sử hữu file, kí hiệu là u)
- Group (Nhóm mà chủ sở hữu thuộc về, kí hiệu là g)
- Others (những người khác, kí hiệu là o)

Từ những điều trên, ta có thể chỉnh sửa các quyền cho những nhóm người dùng khác nhau bằng lệnh chmod. Sau đây là cú pháp của các lệnh.

Đối với cú pháp kí hiệu, lệnh như sau:

Chmod [who] [+/-] [permision] file

- Who : u (owner), g (group), o (other), a (all)
- + để thêm quyền, - để xóa quyền
- Permision : r (read), w (write), x (execute)

Ví dụ như **chmod a+x test.txt** : thêm quyền thực thi file test.txt cho tất cả

chmod g-w file : xóa quyền ghi file của nhóm

chmod o+r file : thêm quyền đọc cho other còn lại

Đối với cú pháp số như sau, quyền được chia làm 3 số:

- Read là 4
- Write là 2
- Execute là 1

Các quyền cộng lại để thành các số. Ví dụ quyền đọc và ghi sẽ là 4+2 = 6. Quyền đọc và thực thi sẽ là 4+1 = 5

Cú pháp lệnh khi này sẽ là chmod [permission] file

Chmod 777 test.txt: đổi quyền file test.txt cho owner là cả 3 quyền, group là cả 3 quyền và other là cả 3 quyền …

Ngoài ra ta có thể sử dụng lệnh **ls –l** để xem quyền của các file và thư mục

5. Lệnh ping, traceroute

5.1. Ping

Lệnh ping trong linux để kiểm tra kết nối từ máy hiện tại đến các ứng dụng khác. Cú pháp lệnh như sau:

Ping [option] [ip đích]

Ví dụ **ping 192.168.1.1**

Một số option lệnh ping như sau:

-c (count) xác định số lượng cụ thể gói tin ping

ping –c 40 192.168.1.1 : gửi 40gói tin ping đến địa chỉ

-i (interval) : xác định khoảng thời gian giữa các gói tin ping

ping –i 2 google.com : gửi gói tin ping mỗi 2s

-s (size) : xác định kích thước gói tin ping

ping –s 100 example.com : gửi gói tin ping có kích thước 100 byte

-t (TTL) : kiểm soát số lượng bước nhảy gói tin có thể đi qua

ping -t 64 example.com : gói tin đi qua tối đa 64 hope

-f (flood) : gửi nhiều gói tin nhất có thể đến địa chỉ ip để kiểm tra server xử lí gói tin như thế nào

ping –f exampledomain.com

…

5.2. Traceroute

Là lệnh dùng để theo dõi tuyến đường đi từ đi của các gói tin từ hệ thống đến đích qua các router và thiết bị mạng khác. Công cụ này xác định các bước nhảy (hops) mà gói tin phải đi qua, từ đó xác định vị trí các vấn đề mạng và hoặc đánh giá hiệu năng mạng

Cú pháp: traceroute [option] destination

-m (max hope) : xác định số bước nhảy tối đa mà tracerouter sẽ theo dõi trước khi dừng

traceroute -m 20 google.com: theo dõi tối đa 20 hope

-p (port) : xác định cụ thể cổng sẽ ping tới, Mặc định sẽ gói tin UDP với port = 33434

traceroute -p 80 google.com (gửi tới port = 80)

-q (number queries) : xác định số lượng gói tin sẽ chuyển đến mỗi bước nhảy

traceroute -q 5 google.com (gửi 5 gói tin đến mỗi bước nhảy)

-I (ICMP) gửi gói tin ICMP

traceroute -I google.com (gửi gói tin ICMP)

-T (TCP)

traceroute -T google.com (gửi bằng giao thức TCP)

6. Lệnh route add, del,… trong server linux, tĩnh và động

Lệnh route trong Linux là một tiện ích dòng lệnh mạnh mẽ, được sử dụng để quản lý bảng định tuyến IP trong hệ thống Linux. Bảng định tuyến lưu trữ thông tin về cách định tuyến lưu lượng truy cập mạng đến các điểm đến khác nhau. Lệnh route cho phép bạn xem, thêm, sửa và xóa các mục trong bảng định tuyến.

Các chức năng chính trong lệnh route linux:

- Hiển thị bảng định tuyến IP hiện tại của hệ thống.
- Thêm tuyến tĩnh đến các mạng hoặc máy chủ cụ thể.
- Sửa các tuyến tĩnh hiện có trong bảng định tuyến.
- Xóa các tuyến tĩnh khỏi bảng định tuyến.
- Xem thông tin chi tiết về một tuyến cụ thể.

Cú pháp lệnh:

Route [option] [command] [netAddress] [netMask] gw [gateway] [metric]

Trong đó

Option: là những tùy chọn cho lệnh route

Command là các lệnh có thể sử dụng như add, change, del, get

netAdd là địa chỉ mạng

netmask : địa chỉ mask

gateway: địa chỉ IP của router để chuyển tiếp gói tin đến đích

metric: ưu tiên trong mạng

ví dụ:

route : hiển thị bảng định tuyến

route –n : hiển thị bảng định tuyến theo dạng số

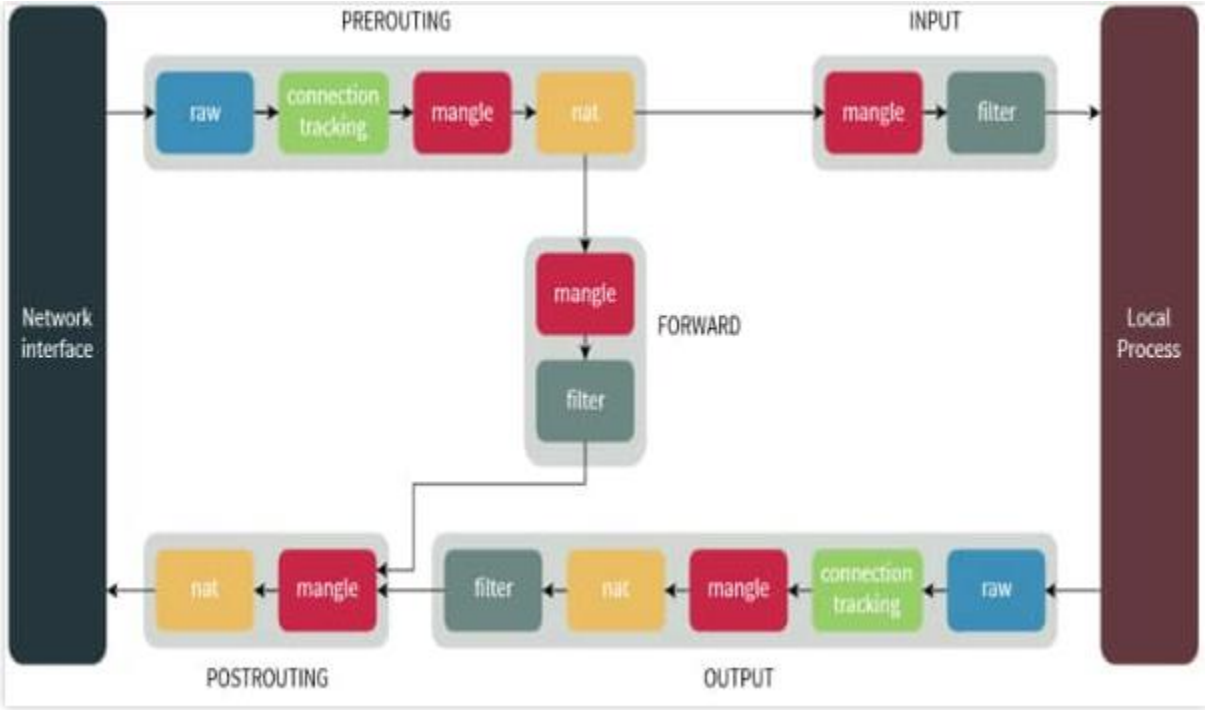
route add default gw 192.168.1.1 : add thêm tuyến đường mặc định đến gateway có IP 19.168.1.1

route add -host 192.168.1.51 reject : từ chối 1 host không cho truy cập

route del default: xóa tuyến default

7. Khai báo IP tables, sau khi khai báo xong sx bật lên cùng OS

Sử dụng tường lửa (Firewall) hầu như là việc bắt buộc phải làm khi chúng ta vận hành máy chủ, để chống lại các truy cập bất hợp pháp bằng cách tự thiết lập ra các quy tắc chặn truy cập của riêng chúng ta. Trong các bản phân phối của Linux như CentOS, Ubuntu, Fedora, … có tích hợp sẵn hay cài đặt 1 công cụ để quản lý, cấu hình các quy tắc firewall gọi là IPtables được sử dụng rộng rãi nhất hiện nay nhưng thật khó cấu hình cho các bạn vừa làm quen hệ thống Linux.



Về cơ bản, IPtables chia làm 3 thành phần: Tables, Chains, Targets

Các Tables có trong IPtables chủ yếu là:

- Filter table : quyết định xem gói tin có được chuyển đến đích không
- NAT table: chỉnh sửa IP khi gửi gói tin đi

Các chains như sau:

- Chain INPUT : các luồng từ bên ngoài vào máy linux chúng ta
- Chain OUTPUT: các luồng từ máy linux đi ra
- Chain FORWARD : các luồng chuyển tiếp qua máy chúng ta

Các target bao gồm:

- ACCEPT: chấp nhận và cho phép gói tin đi vào hệ thống.
- DROP: loại gói tin, không có gói tin trả lời.
- REJECT: loại gói tin những có trả lời table gói tin khác. Ví dụ: trả lời table 1 gói tin "connection reset" đối với gói TCP hoặc "destination host unreachable" đối với gói UDP và ICMP.
- LOG: chấp nhận gói tin nhưng có ghi lại log.

Chúng ta có thể xem các rules như sau:

IPtables -L –v

```
IPtables -L -v

TARGET    PROT OPT  IN OUT SOURCE DESTINATION

ACCEPT    all --   lo any anywhere   anywhere

ACCEPT    all --   any any anywhere   anywhere ctstate  RELATED,ESTABLISHED

ACCEPT    tcp --   any any anywhere   anywhere tcp dpt:ssh

ACCEPT    tcp --   any any anywhere   anywhere tcp dpt:http

ACCEPT    tcp --   any any anywhere   anywhere tcp dpt:https

DROP      all --   any any anywhere   anywhere
```

sau khi gõ lệnh sẽ hiển thị như trên, các trường trong bảng có ý nghĩa như sau:

Target : hành động sẽ thực thi (accept drop)

PROT : giao thức truyền qua

IN: chỉ ra rule sẽ áp dụng cho các gói tin đi vào từ interface nào, ví dụ lo, eth0, eth1 hoặc any là áp dụng cho tất cả interface

OUT: tương tự IN, chỉ ra rule sẽ áp dụng cho các gói tin đi ra từ interface nào.

DESTINATION: địa chỉ của lượt truy cập được phép áp dụng quy tắc.

Một số option có trong lệnh:

- Chỉ định tên table: -t ,
- Chỉ định loại giao thức: -p ,
- Chỉ định card mạng vào: -i ,
- Chỉ định card mạng ra: -o ,
- Chỉ định địa chỉ IP nguồn: -s <địa_chỉ_ip_nguồn>,
- Chỉ định địa chỉ IP đích: -d <địa_chỉ_ip_đích>, tương tự như –s.
- Chỉ định cổng nguồn: –sport ,
- Chỉ định cổng đích: –dport , tương tự như –sport

Các tùy chọn với chain :

- Tạo chain mới : IPtables –N
- Xóa hết rule : IPtables –X
- Liệt kê các rule có trong chain: IPtables -L
- Xóa các rule có trong chain (flush chain): IPtables -F

Một số tùy chọn thao tác với rule:

- Thêm rule : -A (append)
- Xóa rule : -D (delete)
- Thay thế rule –R(replace)
- Chèn thêm rule –I (insert)

Một số ví dụ về lệnh:

IPtables -A INPUT -i lo -j ACCEPT : tạo một rule mới trong chain INPUT cho phép interface lo được accept dữ liệu vào máy linux

Sau khi thay đổi hoặc thêm sửa xóa thì phải lưu và khởi động lại:

service IPtables save

service IPtables restart

IPtables -A INPUT -p tcp --dport 22 -j ACCEPT: cho phép các gói tin tcp đi qua port 22 đi vào máy chủ (cho phép ssh)

IPtables -D INPUT 4 : xóa rule tại vị trí thứ 4 trong bảng

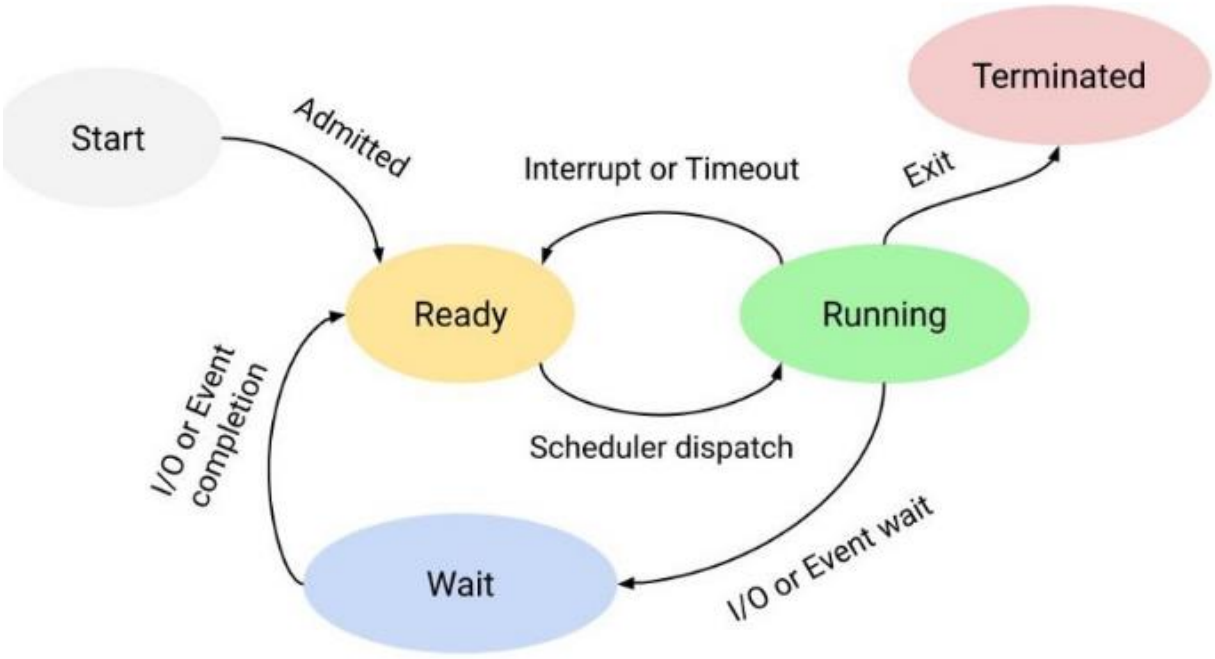
IPtables -D INPUT -j DROP : xóa toàn bộ rule

IPtables – A INPUT –s 192.168.1.1 -j DROP : chặn truy cập từ IP 192.168.1.1

8. Nguyên lí hoạt động của tiến trình

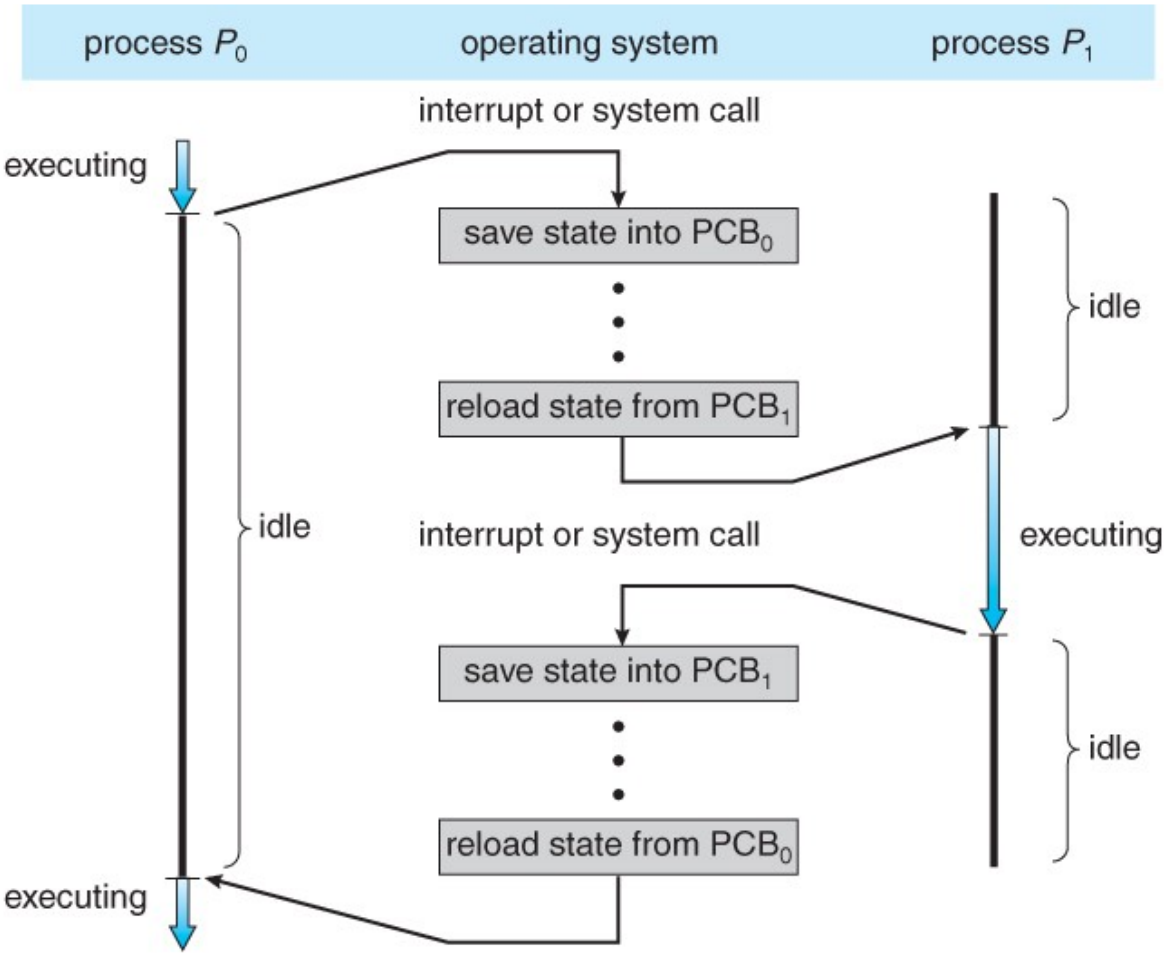
Tiến trình là một chương trình đang được chạy trong một hệ điều hành. Nó có một PCB(process control block) lưu trữ các thông tin về tiến trình như :PID, status, CPU-schedule, con trỏ, I/O status, memory,…

Thông thường CPU không thực hiện cùng lúc nhiều tiến trình cùng một lúc. Thay vào đó, CPU thực hiện nhiều tiến trình nhưng mỗi tiến trình được CPU xử lí trong một khoảng thời gian ngắn. Do vậy CPU cần PCB để theo dõi tiến trình đang được thực hiện gì rồi và cần gì tiếp theo.



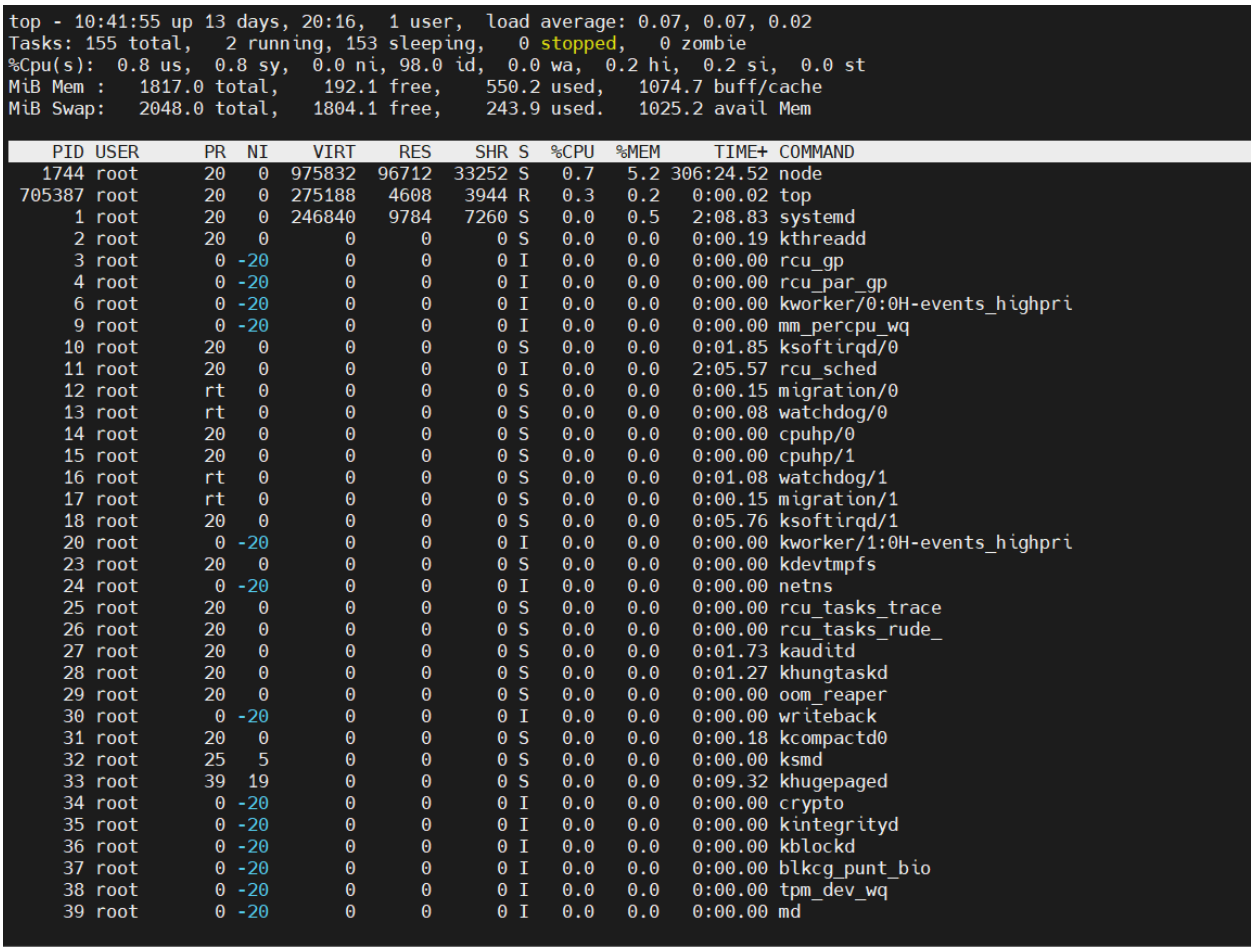
Các trạng thái của tiến trình được thể hiện như hình vẽ:

Ban đầu, khi tiến trình được khởi tạo, nó ở trang thái **start** và sau đó được chuyển sang trạng thái **Ready.** Trạng thái này sẽ duy được duy trì đến khi CPU bắt đầu chuyển sang tính toán tiến trình đó. Lúc này sẽ chuyển sang trạng thái **Running.** Trạng thái này sẽ được duy trì trong một khoảng thời gian sau đó sẽ chuyển sang trạng thái **Wait** nếu tiến trình đang có một sự kiện I/O nào đó đang diễn ra trong tiến trình đó hoặc chuyển sang trạng thái Ready nếu CPU chuyển sang tính toán những tiến trình khác do hết thời gian do thuật toán của hệ điều hành cung cấp. Trạng thái Wait sau khi hết thời gian I/O sẽ chuyển sang trạng thái **Ready**. Trạng thái **Running** sau khi thực hiện xong tác vụ của tiến trình sẽ tiến vào trạng thái **Terminated** và kết thúc tiến trình



Sau khi tiến trình P0 hết thời gian CPU, các thông tin của tiến trình sẽ được lưu vào PCB0. Sau đó, CPU sẽ load PCB1 của tiến trình 1 và thực thi tiến trình đó trong 1 khoảng thời gian thực thi hệ điều hành cung cấp. Tiếp theo sau khi hết thời gian CPU, thông tin của tiến trình 1 sẽ được lưu vào PCB1 và quá trình lại lặp lại, CPU sẽ loadd thông tin của PCB0 và thực thi tiến trình đó.

9. Các thông tin có trong lệnh top



Sau khi gõ lệnh top, ma hình sẽ hiển thị như hình. Nó được chia làm 2 phần:

- Thông tin trên
- Bảng chính

Phần thông tin trên có 4 dòng với nội dung như sau:

- Dòng 1: thời gian hệ thống – thời gian uptime – số lượng người dùng – tải trung bình 1p 5p 15p (tải trung bình là nói về số lượng công việc CPU phải xử lí chiếm bao nhiêu % CPU)
- Dòng 2: tổng số lượng tiến trình – đang chạy – đang ngủ - đang dừng – đang zombie
- Dòng 3: nói về phần trăm CPU người dùng sử dụng – hệ thống sử dụng – phần trăm do các tiến trình có mức độ ưu tiên thấp sử dụng – đang rảnh – IO – phần trăm xử lí gián đoạn phần cứng – phần trăm xử lí gián đoạn phần mềm – phần trăm do máy ảo dùng

- Dòng 4: Tổng bộ nhớ hệ thống(tính bằng Kb) – bộ nhớ trống – bộ nhớ đã sử dụng – bộ nhớ cache
- Dòng 5: tổng bộ nhớ swap(Kb) – bộ nhớ trống – bộ nhớ đã sử dụng – bộ nhớ khả dụng

Bảng chính có các trường sau:

- PID : id của tiến trình
- USER : người dùng
- PR : ưu tiên
- NI :
- VIRT: bộ nhớ ảo được sử dụng bởi tiến trình
- RES: bộ nhớ mà tiến trình sử dụng
- SHR : bộ nhớ có thể chia sẻ
- %CPU : mức độ phần trăm CPU đang được sử dụng bởi tiến trình
- %MEM : mức độ bộ nhớ đang được sử dụng
- TIME+ : thời gian tiến trình đã được chạy
- COMMAND : lệnh

10. Lệnh awk

awk là một ngôn ngữ kịch bản được sử dụng để thao tác dữ liệu và tạo báo cáo. Ngôn ngữ lập trình lệnh awk không yêu cầu biên dịch và cho phép người dùng sử dụng các biến, hàm số, hàm chuỗi và toán tử logic.

awk chủ yếu được sử dụng để quét và xử lý mẫu. Nó tìm kiếm một hoặc nhiều tệp để xem liệu chúng có chứa các dòng phù hợp với các mẫu được chỉ định hay không và sau đó thực hiện các hành động liên quan.

Cách awk hành động:

- Đọc file awk đầu vào theo từng dòng
- Đối với mỗi dòng, nó sẽ khớp lần lượt với các pattern, nếu khớp thì hành động action sẽ được thực hiện, nếu không khớp thì không có hành động action nào được thực hiện

- Cú pháp cơ bản làm việc với lệnh awk thì pattern hoặc action phải có 1 trong 2 không thể thiếu cà 2.
- Mỗi câu lệnh trong phần action được phân tách nhau bởi dấu chấm phẩy.

Cú pháp: **awk '/search-pattern/ {action-take on match; another action;}' file**

Trong đó:

- /search action/ : mẫu tìm kiếm
- Action-take-on match : hành động khi có mẫu trùng
- Another action: các hành động tiếp theo nếu có nhiều hành động
- File: tên file muốn tìm kiếm

Một số ví dụ:

1. In hết tất cả file (tương đương lệnh cat)

Awk '{print}' file : in ra tất cả file

2. In ra cột thay vì in hết

Awk '{print $1, $2, $NF}' file : in ra cột 1, cột 2 và cột cuối cùng của file

3. In ra các trường theo kí tự

Awk '/abc/ && /def/ {print}' file : in ra các bản ghi có chữ abc trong file

4. So sánh

Awk '$2 > 500 {print $1, $2 >> "output.txt"}' file : xét điều kiện nếu cột 2 của file > 500 thì sẽ in ra cột 1 và cột 2 vào file output.txt

…

11. Crontab – lặp lịch

Đối với những yêu cầu công giống nhau tại một thời điểm và lặp đi lặp lại như báo thức vào một thời gian cụ thể nào đó, backup dữ liệu đều đặn ,…. Thì ta có thể sử dụng lặp lịch thay cho việc sẽ phải tự tay làm những công việc đó. Lặp lịch sẽ canh

đến đúng thời điểm đó để thực hiện công việc đó cho chúng ta mà chúng ta không cần lo lắng về điều đó.

Để sử dụng được crontab, trước tiên cần cài đặt trước sau đó mới sử dụng. Ta sử dụng câu lệnh sau: **sudo apt-get install cron**

Cấu trúc của crontab như sau:

```
*     *     *     *     *     lệnh hoặc script được chạy
-     -     -     -     -
|     |     |     |     |
|     |     |     |     +----- Thứ (0 - 6) (Chủ Nhật=0)
|     |     |     +------- Tháng (1 - 12)
|     |     +--------- Ngày (1 - 31)
|     +----------- giờ (0 - 23)
+------------- phút (0 - 59)
```

Ta điền thời gian mong muốn rồi sau đó thêm script muốn chạy vào. Sau đó, cứ đến thời gian khoảng thời gian đó, hệ thống sẽ tự động chạy thực thi các lệnh ta mong muốn

Một số lệnh thường dùng trong crontab:

crontab -e: tạo, chỉnh sửa các crontab

crontab -l: Xem các Crontab đã tạo

crontab -r: xóa file crontab

một số ví dụ về crontab

30 * * * * command : chạy script 30p một lần

0 3 * * * command : chạy 3h sang mỗi ngày

30 3 5 7 * command : chạy vào lúc 3h30p sáng ngày 5 tháng 7

0 7,21 * * * command : chạy vào lúc 7h sáng và 21h mỗi ngày

0 4 * * 1-5 command : chạy từ thứ 2 đến thứ 6 lúc 4h sáng

12. Lệnh scp trong linux

Lệnh này sẽ copy file hoặc thư mục từ máy chủ từ xa đến máy chủ cục bộ của chúng ta. Vì lệnh này chuyển tệp qua mạng đến một số máy chủ lưu trữ khác, nên cần có quyền truy cập SSH. Scp thường được ưu tiên hơn các phương pháp truyền tệp khác vì trong quá trình truyền, kết nối giữa hai máy chủ được mã hóa. Giao thức SSH chịu trách nhiệm mã hóa các tệp, mật khẩu và bất kỳ chi tiết nhạy cảm nào khác. Có 3 trường hợp thường được sử dụng là:

- Máy chủ cục bộ đến máy chủ từ xa
- Máy chủ từ xa đến máy chủ cục bộ
- 2 máy chủ từ xa

Cú pháp lệnh:

Scp [option] [source] [destination]

Nếu ta muốn chuyển 1 file từ máy chủ từ xa đến máy chủ cục bộ, ta có thể dùng lệnh như sau: **scp user@remote-host:/home/document.txt /home :** lệnh này sẽ chuyển file document.txt từ máy chủ từ xa đến thư mục home của máy chủ cục bộ. Lệnh này sẽ yêu cầu ta nhập password của máy chủ từ xa

Nếu ta muốn chuyển 1 file từ máy chủ cục bộ đến máy chủ từ xa, ta có thể dùng lệnh như sau: **scp /home/document.txt user@remote-host:/home** lệnh này sẽ chuyển file document.txt từ máy chủ cục bộ đến thư mục home của máy chủ từ xa. Lệnh này sẽ yêu cầu ta nhập password của máy chủ từ xa

Nếu ta muốn chuyển thư mục, file từ 2 máy từ xa với nhau thì ta có thể dùng lệnh sau: **scp user1@remote-host1:/home/document.txt user2@remote-host2:/home/folder**. Lệnh này sẽ yêu cầu nhập mật khẩu của 2 máy chủ từ xa

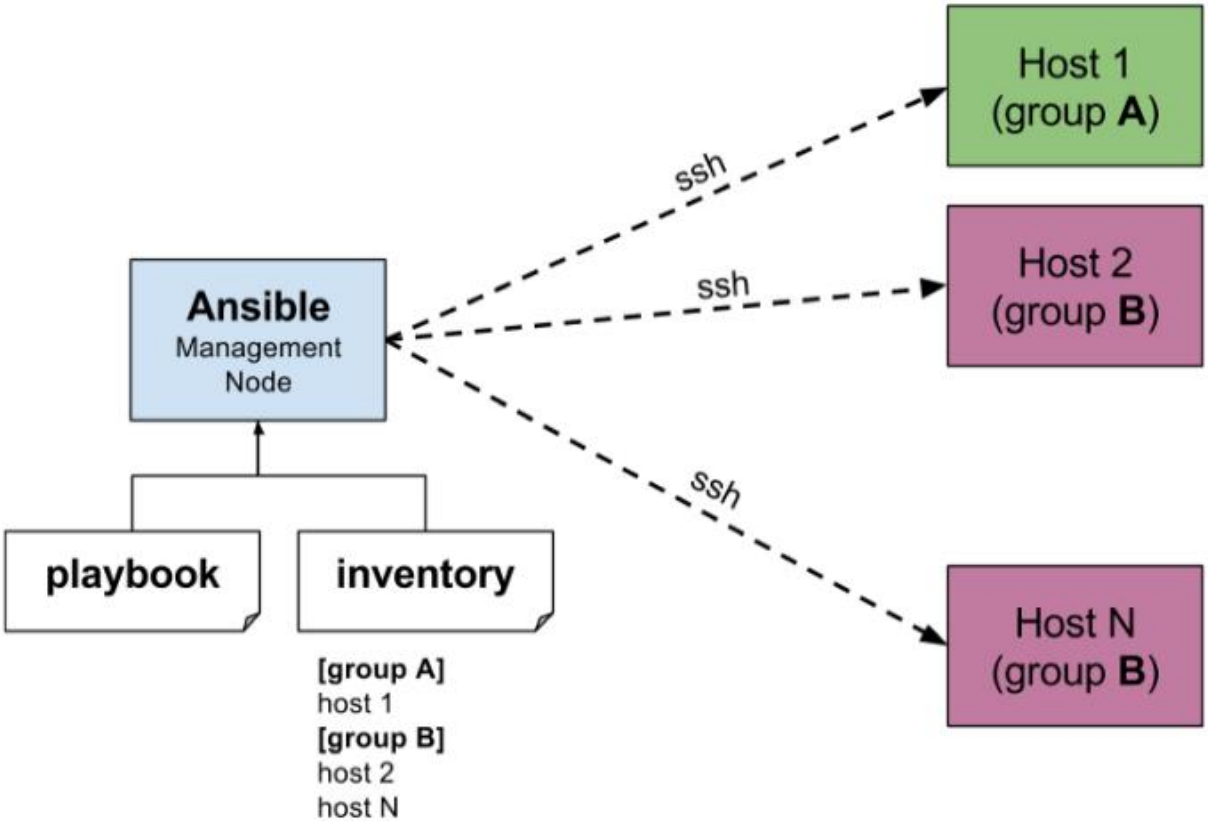
Một số option có thể kể đến như sau:

- –P port xác định chính xác cổng vào của server (port mặc định của TCP là 22)
- –r là recursive copying, có nghĩa là bao gồm tất cả thư mục con
- –u sẽ xóa file nguồn sau khi chuyển đổi hoàn tất.

…

13. Ansible

Thông thường để cấu hình các máy chủ, ta cần phải cấu hình từng máy một và đọc lại các câu lệnh từng chút một. Điều này gây mệt mỏi cho các quản trị viên khi phải làm theo từng bước ở mỗi thiết lập mới và điều này có thể gây ra sai lầm và thiếu sót. Khi đó Ansible sinh ra để chúng ta xử lí việc đó. Ansible giúp chúng ta cấu hình nhiều server cùng một lúc giúp tăng độ chính xác và giảm thiểu thời gian phải thao tác cài đặt trên nhiều server khác nhau



Để cài đặt Ansible, ta có thể sử dụng lệnh sau:

apt-add-repository -y ppa:ansible/ansible

apt-get update

apt-get install -y ansible

Trong ansible có một số key concept trong ansible như sau:

- Inventory : chứa danh sách các máy chủ mà ta muốn quản lí
- Playbook: chứa các tập lệnh mà ta muốn server thực thi
- Task : một công việc nhỏ trong số những tập lệnh của server
- Roles
- Variables

13.1. Inventory

Có dạng như sau:

vim /etc/ansible/hosts

[local]

127.0.0.1

[apiserver]

192.168.88.13

192.168.88.14

[jobserver]

192.168.89.100

192.168.89.101

[dbservers]

192.168.90.200

192.168.90.201

[group_server1:childrens]

apiserver

jobserver

chứa thông tin của các server và nhóm của chúng nếu có

cú pháp lệnh của Ansible

ansible –i [tên host cần gọi] -m [tên module] -a [tham số truyền vào module]

-i : inventory host. Trỏ thư viện group_host cần gọi, mặc định nếu không có -i thì sẽ gọi /etc/ansible/hosts

-m : gọi module của ansible

-a : command_argument gửi kèm theo module mà ta đang gọi

-u : user

-vvvv : debug option

ansible apiserver -m ping (giải thích: gọi ping toàn bộ các hosts trong inventory)

ansible apiserver -m command -a uptime

ansible apiserver -a uptime (Default, ansible sẽ cho module = "command")

ansible dbserver -m service -a "name=mysql state=restarted

Ansible sử dụng giao thức ssh để liên lạc với các server khác nên ta cần biết mật khẩu của các server đó trước khi sử dụng ansible để liên lạc.

13.2. Playbook

Chứa tập lệnh mà ta muốn server thực thi.

```
---
- hosts: local
  tasks:
    - name: Ping check host
      ping: ~
    - name: Install Apache2
      apt: name=apache2 update_cache=yes
```

ví dụ như ở trên có 2 task, nhiệm vụ đầu tiên là ping đến localhost và lệnh thứ 2 là install apache2 cho máy localhost. Tương tự như vậy ta có thể viết thêm lệnh và sau đó lưu vào file rồi nhiệm vụ tiếp thep là apply file đó là ta đã thành công cài đặt các lệnh cần thiết cho server

13.3. Loop

Thay vì viết module cho từng nhóm, giờ ta nhóm lại để chạy 1 lần luôn

Ta có inventory như sau:

cat /etc/ansible/hosts

[allone]

192.168.88.88

Giờ để cài đặt môi trường cho server, ta có thể làm như sau:

```
---
- hosts: allone
  become: yes

  tasks:
    - name: Install Apache.
      apt:
        name: "{{ item }}"
        state: present
      loop:
        - apache2
        - mysql-server
```

```
    - php
    - php-mysql
- name: Restart Apache and Mysql
  service:
    name: "{{item}}"
    state:  running
  loop:
     - apache2
     - mysql
```

Ansible sẽ loop qua từng Item và cài đặt 4 ứng dụng apache2, mysql-server, php, php-mysql

13.4. Roles

Đổi với những task có những play nhỏ hơn giống nhau thì ta có thể viết chung chúng trong roles. Ngoài những task ra, ta có thể viết các files, template, khai báo các biến chung vào đây rồi sau đó sử dụng trong các file hoặc thư mục ansible khác. Mặc định, ansible sẽ nhìn vào những file main.yml nên ta sẽ viết vào những file này

```
roles/
    common/               # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the template resource
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for this role
        meta/             #
            main.yml      #  <-- role dependencies
        library/          # roles can also include custom modules
        module_utils/     # roles can also include custom module_utils
        lookup_plugins/   # or other types of plugins, like lookup in this case

    webtier/              # same kind of structure as "common" was above, done for the webtier role
    monitoring/           # ""
    fooapp/               # ""
```

14. Bài tập

Đề bài: cho file statistic.log. Sử dụng các lệnh trong linux, awk,.. để thực hiện task sau:

1. Hiển thị thông tin QuerySucess, QueryFail, UpdateSuccess, UpdateFail, Avg Res Time, tps
2. Vừa hiển thị thông tin như câu 1 vừa ghi kết quả ra file output.txt
3. Hiển thị thời gian,tps cao nhất trong file log theo thứ tự giảm dần, giới hạn 10 mốc
4. Hiển thị thời gian, tps cao nhất trong file log theo thứ tự tăng dần, giới hạn 10 mốc

Giải

Câu 1:

Ta dùng câu lệnh sau:

```
awk –F ‘[| ]’ ‘{
for(i=1;i<=$NF;i++){
        if($i ~ / QuerySucess:/) {qSuccess = $(i+1)}
        if($i ~ / QueryFail:/) {qFail = $(i+1)}
        if($i ~ / UpdateSucess:/) {uSuccess = $(i+1)}
        if($i ~ / UpdateFail:/) {uFail = $(i+1)}
        if($i ~ /Avg/ && $(i+1) ~ /Res/ &&  $(i+2) ~ /Time:/) {avgResTime = $(i+3)}
        if($i ~ /tps/) {tps = $(i+1)}
}
print qSuccess, qFail, uSuccess, uFail, avgResTime, tps
}’ statistic.log
```

Giải thích:

Ta dùng lệnh awk để lọc. Option –F để chia nhỏ dòng log ra thành nhiều cột khác nhau và chia theo 2 kí tự gạch dọc (|) và dấu cách ( ) mà ta để trong ngoặc vuông. Tiếp theo ta đi lần lượt từ cột đầu tiên (i=1) đến cuối cùng (i = $NF – đây là một biến được gán mặc định trong awk chỉ định số cuối cùng sau khi chia nhỏ bằng option -F). Nếu tìm được cột có thông tin như yêu cầu (**QuerySucess, QueryFail,…**) thì ta sẽ lấy cột tiếp theo (cột tiếp theo là cột chứa dữ liệu của tham số đó, ta cần lấy dữ liệu) và gán vào biến. Sau đó ta in biến đó ra là xong.

Câu 2 :

Ta dùng câu lệnh :

```
awk –F ‘[| ]’ ‘{
for(i=1;i<=$NF;i++){
        if($i ~ / QuerySucess:/) {qSuccess = $(i+1)}
        if($i ~ / QueryFail:/) {qFail = $(i+1)}
        if($i ~ / UpdateSucess:/) {uSuccess = $(i+1)}
        if($i ~ / UpdateFail:/) {uFail = $(i+1)}
        if($i ~ /Avg/ && $(i+1) ~ /Res/ &&  $(i+2) ~ /Time:/) {avgResTime = $(i+3)}
        if($i ~ /tps/) {tps = $(i+1)}
}
print qSuccess, qFail, uSuccess, uFail, avgResTime, tps
}’ statistic.log >> output.txt
```

Tương tự như câu trên tuy nhiên ta thêm phần ghi ra file output.txt ở cuối

Câu 3:

Ta dùng lệnh sau:

```
awk –F ‘[| ]’ ‘{
for(i=1;i<=$NF;i++){
        if($i ~ /tps/) {tps = $(i+1)}
}
Print $1, $2, tps
}’ statistic.log  | sort -r –k 3 | head -10
```

Giải thích:

Tương tự ta dùng câu lệnh awk để lọc. Lần này hiển thị ra ngày, giờ và thông số tps (giờ ở $1 và ngày ở $2). Sau đó ta lấy dữ liệu vừa lọc ra được cho vào lệnh sort để sắp xếp dữ liệu. Option **–r** để sắp xếp giảm dần, option **–k 3** là để sắp xếp theo cột thứ 3 là cột tps (cột 1 là giờ, cột 2 là ngày, cột 3 là tps). Sau khi đã sắp xếp xong dữ liệu trên, ta lại dùng nó để lấy ra 10 cái đầu tiên bằng lệnh **head -10**

câu 4:

ta dùng lệnh sau

```
awk –F ‘[| ]’ ‘{
for(i=1;i<=$NF;i++){
        if($i ~ /tps/) {tps = $(i+1)}
}
Print $1, $2, tps
}’ statistic.log  | sort –k 3 | head -10
```

Tương tự như câu trên nhưng lần này ta sắp xếp theo chiều tăng dần và in ra 10 giá trị đầu tiên.