BAB V
IMPLEMENTASI

Pada bab implementasi, akan dibahas langkah-langkah yang dilakukan dalam pembuatan sistem virtualisasi *server*. Sesuai dengan rancangan pada Bab IV, sistem virtualisasi *server* ini dibangun dengan menggunakan tiga buah komputer HVM dan sebuah komputer *shared storage*. Pada tahap implementasi, langkah-langkah yang akan dilakukan penulis antara lain instalasi, konfigurasi, dan pembuatan berkas program. Dalam hal ini, langkah-langkah tersebut mengacu pada perangkat keras dan perangkat lunak yang akan digunakan.

5.1 Implementasi HVM

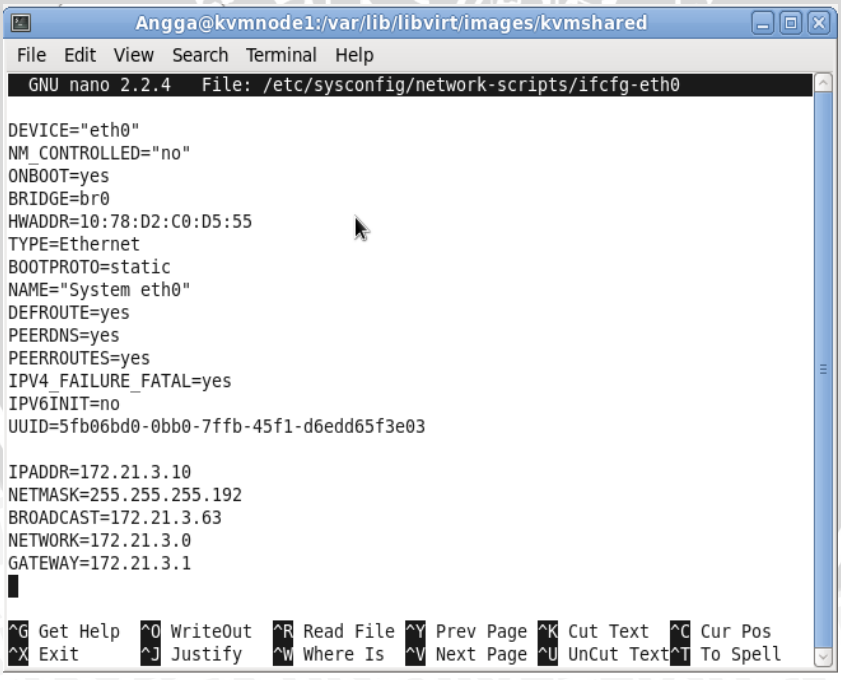
Langkah awal yang perlu dilakukan untuk membangun sistem virtualisasi *server* yaitu menghubungkan seluruh komponen sistem yang ada. Seluruh komponen sistem tersebut dihubungkan dengan menggunakan kabel UTP melalui sebuah *switch*. Langkah selanjutnya adalah proses instalasi perangkat lunak pada HVM satu, HVM dua, dan HVM tiga. Proses instalasi perangkat lunak yang perlu dilakukan pertama kali adalah instalasi sistem operasi Fedora 14-x86_64. Khusus untuk HVM, sistem operasi Fedora yang digunakan yaitu versi *graphical desktop*. Setelah proses instalasi sistem operasi Fedora dilakukan, proses instalasi dilanjutkan dengan perangkat lunak pendukung lainnya seperti KVM *Hypervisor*, NFS, sysstat, dan virt-top.

5.1.1 Konfigurasi Perangkat Lunak

Setelah proses instalasi seluruh perangkat lunak dilakukan, langkah selanjutnya yaitu mengkonfigurasi beberapa berkas dari sistem operasi dan perangkat lunak. Hal pertama yang dilakukan penulis yaitu mematikan salah satu sistem keamanan sistem operasi berupa *firewall*. Proses tersebut dapat dilakukan dengan cara menonaktifkan *firewall* secara menyeluruh atau dengan hanya membuka beberapa *port* yang diperlukan untuk komunikasi antar HVM dan *shared storage*. Dengan mematikan *firewall*, HVM dapat melakukan pengaksesan *shared storage* dan menjalankan proses *live migration* mesin virtual. Selain itu, diperlukan pula beberapa rsa *key public* agar seluruh HVM dapat saling

berkomunikasi tanpa dibatasi oleh aturan autentikasi sistem operasi. Hal ini sangat membantu sistem untuk dapat memindahkan mesin virtual dari HVM satu ke HVM lainnya tanpa harus memasukkan *password*.

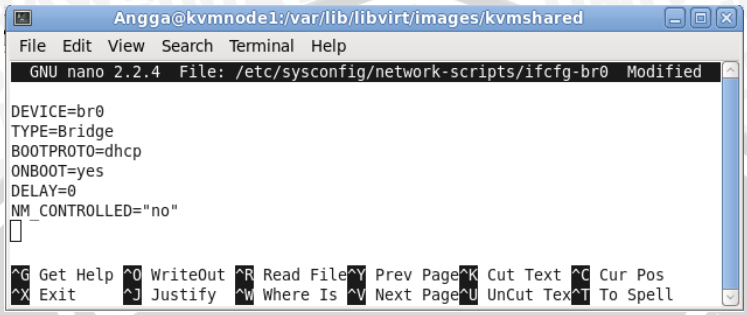
Langkah konfigurasi selanjutnya adalah mengubah *script* jaringan *Ethernet* dan *Bridge* pada seluruh HVM. Konfigurasi *script* ini diperlukan untuk mendapatkan alamat *Internet Protocol* statis pada sebuah segmen jaringan. Dengan segmen jaringan yang sama, seluruh komponen sistem dapat saling berkomunikasi dalam sebuah jaringan lokal. Pada *script Ethernet* tersebut, penulis mengganti baris BOOTPROTO dengan nilai *static* dan kemudian menambahkan beberapa baris beserta alamat IP masing-masing seperti IPADDR, NETMASK, BROADCAST, NETWORK, dan GATEWAY. Selain itu, penulis juga menambahkan satu baris BRIDGE yang memiliki nilai br0. Baris tersebut menjembatani *script Bridge* dan *script Ethernet* untuk mendapatkan alamat IP dalam satu segmen jaringan yang sama. Berikut konfigurasi *script* jaringan *Ethernet* yang dilakukan penulis pada salah satu HVM:



Gambar 5.1 Konfigurasi *Script* Jaringan *Ethernet* Pada HVM

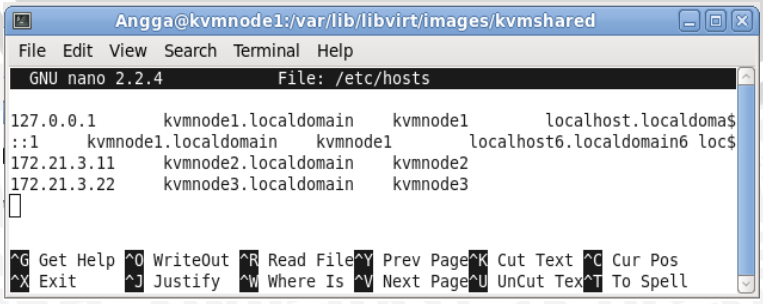
Selain konfigurasi *script* jaringan *Ethernet*, penulis juga membuat sebuah script baru berupa *script* jaringan *Bridge*. *Script* ini diperlukan agar KVM dapat

memberikan alamat *Internet Protocol* mesin virtual dengan segmen jaringan yang sama seperti pada *server* fisik. Konfigurasi *script* jaringan *Bridge* dibuat di dalam berkas yang berbeda namun masih berhubungan dengan *script* jaringan *Ethernet*. Berikut konfigurasi *script* jaringan *Bridge* pada salah satu HVM:

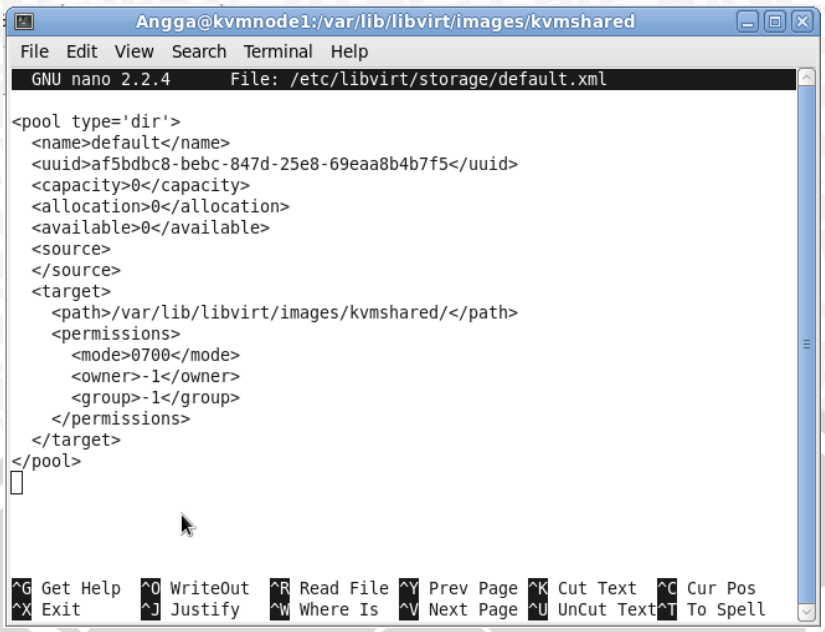


Gambar 5.2 Konfigurasi *Script* Jaringan *Bridge* Pada HVM

Langkah konfigurasi terakhir yaitu merubah dan menambahkan isi dari beberapa berkas yang berhubungan dengan penggunaan perangkat lunak NFS dan KVM. Berkas pertama yang perlu dikonfigurasi adalah berkas hosts pada direktori /etc/. Berkas ini berfungsi sebagai penerjemah nama HVM untuk dirubah ke dalam alamat IP. Dengan begitu, penulis tidak perlu menulis atau menghafal alamat IP masing-masing HVM. Berkas selanjutnya yang perlu dikonfigurasi yaitu berkas default.xml. Berkas ini berada pada direktori /etc/var/libvirt/storage/ yang berisi *pool storage* KVM. Hal ini diperlukan karena penulis membuat sebuah direktori baru untuk menyimpan berkas NFS sehingga tujuan *path pool storage* perlu diarahkan ke direktori /var/lib/libvirt/images/kvmshared/. Berikut hasil konfigurasi dari berkas hosts dan berkas default.xml yang dilakukan pada HVM satu:



Gambar 5.3 Konfigurasi Berkas Hosts



Gambar 5.4 Konfigurasi Berkas *Pool Storage*

5.1.2 Pembuatan Berkas Program

Setelah seluruh instalasi dan konfigurasi selesai dilakukan, proses selanjutnya adalah pembuatan beberapa berkas berupa *Bash Shell* dan teks. Setiap berkas memiliki fungsi tersendiri namun memiliki tujuan yang sama yaitu untuk membangun sistem virtualisasi *server*. Pada HVM satu, seluruh berkas tersebut terdapat pada sebuah direktori /home/Script/. Berikut seluruh daftar berkas yang dibuat penulis pada HVM satu beserta penjelasan singkat dari masing-masing berkas:

Tabel 5.1 Daftar Berkas Sistem Virtualisasi *Server*

No.	Nama Berkas	Fungsi
1	hvm1top.sh	Berkas *shell* ini digunakan untuk mengeksekusi perintah monitoring mesin virtual. Hasil dari eksekusi perintah tersebut ditampung sementara dalam berkas raw.txt.
2	hvm2.txt	Berkas ini berisi nilai 0 dan 1 mengenai informasi apakah HVM dua memiliki VM yang sedang berjalan di dalamnya.

3	hvm2vm.txt	Berkas ini berisi informasi nama VM yang sedang berjalan pada HVM dua.
4	hvm3.txt	Berkas ini berisi nilai 0 dan 1 mengenai informasi apakah HVM tiga memiliki VM yang sedang berjalan di dalamnya.
5	hvm3vm.txt	Berkas ini berisi informasi nama VM yang sedang berjalan pada HVM tiga
6	hvmload.txt	Berkas ini berisi informasi mengenai beban kerja HVM satu yang ditampung sementara dari hasil eksekusi hvm1top.sh.
7	hvmon.txt	Berkas ini berisi informasi HVM yang sedang memiliki VM yang berjalan di dalamnya.
8	kalkul.sh	Berkas *shell* ini digunakan untuk menghitung rata-rata dari data beban kerja hasil eksekusi hvm1top.sh.
9	log.txt	Berkas ini berisi informasi lengkap dari hasil eksekusi program main_script.sh yang dieksekusi di setiap menit. Berkas ini berisi waktu eksekusi, proses perpindahan yang terjadi, beban CPU HVM satu, beban *memory* HVM satu, beban CPU VM satu, VM dua, dan VM tiga.
10	main_script.sh	Berkas ini merupakan *Bash Shell* utama yang mengatur keseluruhan proses yang terjadi pada sistem virtualisasi *server*.
11	node1load.sh	Berkas *shell* ini digunakan untuk mengeksekusi perintah monitoring beban HVM. Isi dari berkas ini adalah perintah mpstat yang dieksekusi selama satu menit. Data yang diambil dari berkas ini adalah rata-rata beban HVM dari satu menit.
12	raw.txt	Berkas ini berisikan data mentah dari hasil eksekusi

		virt-top yang dijalankan oleh berkas hvm1top.sh.
13	totalvm.txt	Berkas ini berisi informasi jumlah total mesin virtual yang berjalan pada HVM satu. Informasi tersebut didapat dengan melakukan eksekusi perintah virsh list.
14	virttop1.txt	Berkas ini berisi infomasi akhir dari hasil kalkulasi berkas kalkul.sh. Informasi yang disimpan dalam berkas ini yaitu beban kerja CPU VM beserta nama VM tersebut.
15	vmmigrate2.txt	Berkas ini berisi informasi VM yang dipindahkan kedua kali.
16	vmmigrate.txt	Berkas ini berisi informasi VM yang dipindahkan pertama kali.

Dari tabel di atas, terdapat empat berkas yang merupakan berkas *Bash Shell*. Berkas *shell* tersebut meliputi hvm1top.sh, kalkul.sh, node1load.sh, dan main_script.sh. Berkas main_script.sh merupakan program *Bash Shell* utama yang berfungsi untuk mengatur keseluruhan proses yang dilakukan pada sistem ini. Penulisan secara lengkap kode program *Bash Shell* tersebut penulis sertakan dalam Lampiran di akhir penulisan penelitian. Berikut penjelasan dari masing-masing berkas *shell* pada HVM satu:

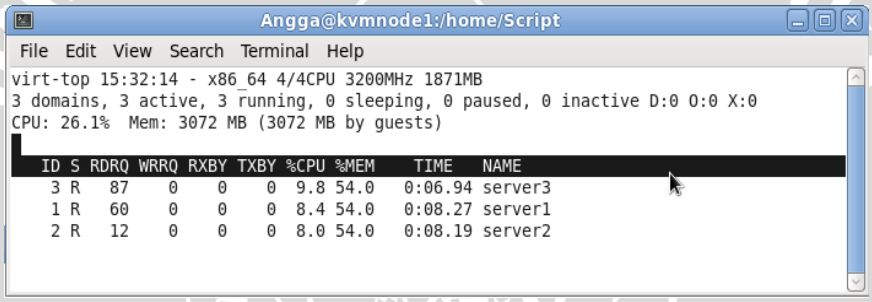
- **hvm1top.sh**

  Berkas *shell* ini digunakan untuk mengeksekusi perintah monitoring mesin virtual. Hasil eksekusi dari perintah tersebut kemudian ditampung sementara di dalam berkas raw.txt. Berikut kode program dari berkas hvm1top.sh:

```
#!/bin/bash

sleep 2
virt-top -d 2 -n 27 --script --csv
/home/Script/raw.txt
```

Dari kode program hvm1top.sh, terdapat perintah virt-top yang digunakan penulis untuk melakukan monitoring beban kerja mesin virtual. Pemberian tanda –d atau *delay*, menandakan bahwa proses eksekusi dilakukan dalam selang waktu dua detik. Sedangkan tanda –n adalah jumlah iterasi, yaitu sebanyak 27 kali eksekusi dalam rentang waktu satu menit. Untuk dapat mengambil data beban kerja CPU mesin virtual dari hasil eksekusi virt-top, penulis perlu merubah perintah virt-top ke dalam bentuk lain yaitu script csv. Hasil dari eksekusi perintah virt-top tersebut yang kemudian ditulis kembali ke dalam berkas raw.txt. Berikut contoh dari monitoring beban kerja CPU mesin virtual dengan menggunakan perintah virt-top sebelum diubah ke dalam bentuk script csv:



Gambar 5.5 Keluaran Dari Perintah Virt-top

- **kalkul.sh**

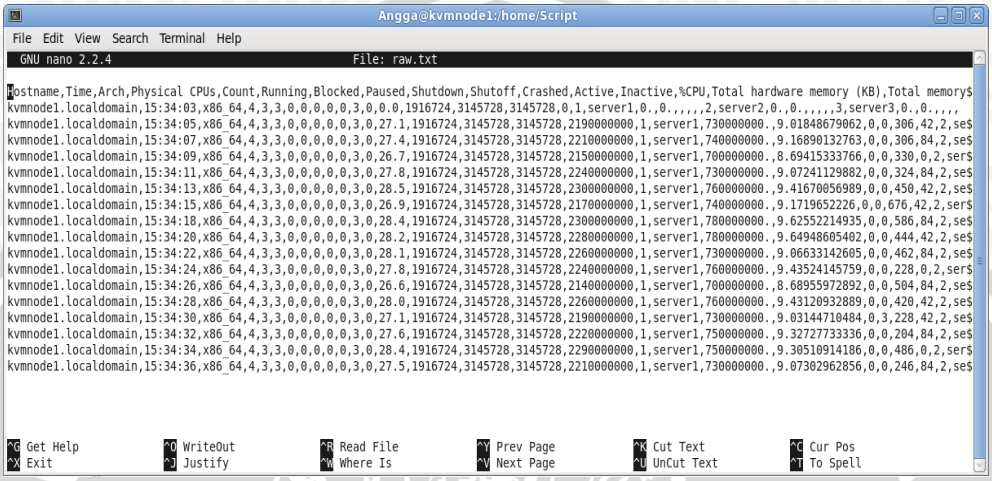
Berkas *shell* ini digunakan untuk menghitung rata-rata beban kerja CPU mesin virtual dari hasil eksekusi berkas hvm1top.sh. Setelah eksekusi dilakukan, hasil perhitungan berupa angka ditulis ke dalam berkas lain yaitu virttop1.txt. berikut kode program dari berkas kalkul.sh:

```
#!/bin/bash

awk -F "\"*,\"*" ' {sum1+=$22} {sum2+=$30} {sum3+=$38}
END                                                {print
$20"\n"sum1/26"\n"$28"\n"sum2/26"\n"$36"\n"sum3/26}'
/home/Script/raw.txt > /home/Script/virttop1.txt
```

Berdasarkan kode program di atas, dapat dilihat bahwa penulis melakukan penghitungan beban kerja CPU dari berkas raw.txt secara

manual. Langkah yang dilakukan adalah dengan mengambil data-data berdasarkan kolom yang berisikan beban kerja CPU mesin virtual. Berikut contoh data-data kasar beban kerja CPU dari berkas raw.txt yang diambil melalui berkas program kalkul.sh:



Gambar 5.6 Berkas Raw.txt

Berkas *shell* kalkul.sh dibuat dengan menggunakan bantuan perintah awk. Dengan perintah awk, penulis dapat mengambil data secara spesifik berdasarkan kolom atau baris dari sebuah berkas. Dari kode program kalkul.sh dan berkas raw.txt, dapat dilihat bahwa kolom 22, 30, dan 38 merupakan data beban kerja CPU untuk *server* 1, *server* 2, *server* 3. Penghitungan rata-rata dilakukan dengan menjumlahkan nilai di tiap kolom data dan kemudian membaginya dengan total jumlah iterasi yang dilakukan pada berkas hvm1.sh. Setelah itu, berkas kalkul.sh menulis hasil peng-hitungan rata-rata beserta nama mesin virtual pada sebuah berkas lain yaitu virttop1.txt. Berkas ini yang kemudian digunakan oleh main_script.sh untuk mendapatkan informasi beban kerja CPU mesin virtual yang berada pada HVM satu.

- **node1load.sh**

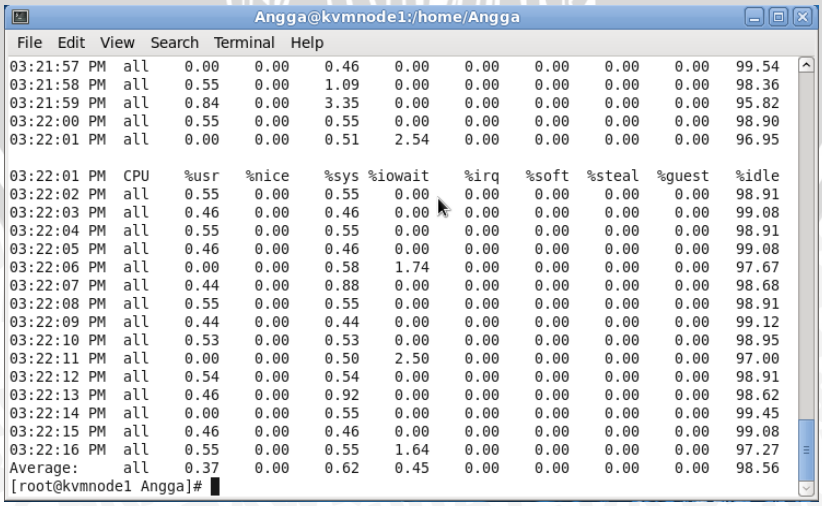
  Berkas *shell* ini digunakan untuk mengeksekusi perintah monitoring beban kerja CPU HVM satu. Isi dari berkas ini adalah perintah mpstat yang dieksekusi dalam rentang waktu satu menit. Data yang diambil dari berkas

ini adalah hasil rata-rata beban kerja CPU HVM satu selama satu menit tersebut. Berikut kode program dari berkas node1load.sh:

```
#!/bin/bash

loadfloat=`mpstat 1 59 | awk 'FNR==63 {print 100-
$11}'`
node1load=${loadfloat/.*}
echo $node1load > '/home/Script/hvmload.txt'
exit
```

Informasi beban kerja CPU HVM satu diperlukan sebagai parameter untuk melakukan pemindahan mesin virtual. Untuk mendapatkan informasi tersebut, penulis menggunakan perintah mpstat yang dieksekusi pada HVM satu. Dalam program ini, data yang diambil adalah hasil rata-rata beban kerja CPU dari 59 kali iterasi dalam satu menit. Tidak seperti perintah virt-top sebelumnya, perintah mpstat menyediakan hasil penghitungan rata-rata dari setiap eksekusi perintah. Oleh karena itu, perintah awk juga diperlukan penulis dalam berkas ini untuk mengambil data spesifik pada baris dan kolom tertentu. Data yang diambil berasal dari baris terakhir yaitu 63 dan kolom 11. Hasil akhir eksekusi program ini kemudian dikonversi menjadi bilangan bulat dan ditulis dalam berkas hvmload.txt yang kemudian digunakan oleh berkas main_script.sh sebagai parameter batas beban kerja CPU HVM satu. Berikut contoh dari eksekusi perintah mspstat yang dilakukan dengan 59 kali iterasi dalam satu menit:



Gambar 5.7 Keluaran Dari Perintah Mpstat

- **main_script.sh**

  Berkas main_script.sh merupakan program *Bash Shell* utama yang mengatur keseluruhan proses yang terjadi pada sistem virtualisasi *server*. Program *shell* ini merupakan representasi dari algoritma program *Bash Shell* yang telah dirancang pada Bab IV. Berkas ini berisi perintah pengambilan data, operasi penghitungan, seleksi kondisi, migrasi atau remigrasi, dan juga pencatatan *log* sistem virtualisasi *server*. Berikut salah satu contoh kode program untuk perintah pengambilan data dan operasi penghitungan:

```
cpuload=`cat /home/Script/hvmload.txt | head -n 1`
bufcach=`free | awk 'FNR==3 {print $3}'`
memtot=`free | awk 'FNR==2 {print $2}'`
let memload=$bufcach*100/$memtot
```

  Berikut salah satu contoh kode program untuk perintah remigrasi mesin virtual:

```
virsh --connect qemu+ssh://kvmnode2.localdomain/system
\ migrate --live server$vmmigrate2 qemu+ssh:///system
```

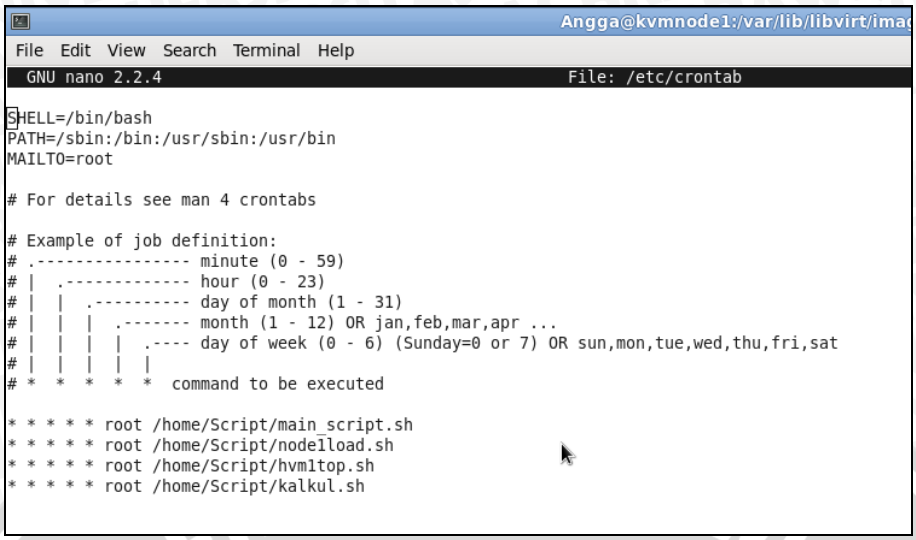
  Berikut salah satu contoh kode program untuk perintah seleksi kondisi:

```
maxhvm=80
if [ $cpuload -gt $maxhvm ];then
```

  Berikut salah satu contoh kode program untuk perintah pencacatan *log* sistem virtualisasi *server* ke berkas log.txt:

```
echo -en $date "\tMIGRASI (VM"$vmtarget" KE
HVM2)\t\tHVM1 CPU = "$cpuload"%\tHVM1 MEM =
"$memload"%\t\tVM1 LOAD = "$vm1load"%\tVM2 LOAD =
"$vm2load"%\tVM3 LOAD = "$vm3load"%\n" >>
'/home/Script/log.txt'
```

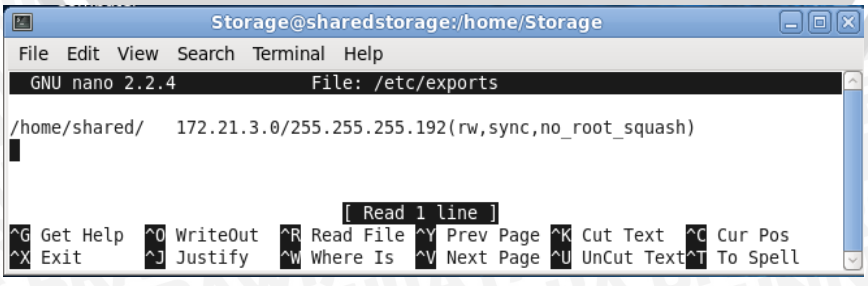
Setelah proses pembuatan seluruh berkas selesai dilakukan, langkah terakhir yaitu mengatur waktu eksekusi program *Bash Shell*. Seluruh berkas *Bash Shell* dieksekusi secara berkala di setiap menit dengan menggunakan bantuan Linux Cron. Berdasarkan tabel 5.1, terdapat empat buah program *Bash Shell* yang dieksekusi di setiap menit dengan hak pengguna sebagai *root*. Berikut hasil pengaturan eksekusi Crontab yang dilakukan pada HVM satu:



Gambar 5.8 Pengaturan Eksekusi Crontab

5.2 Implementasi *Shared Storage*

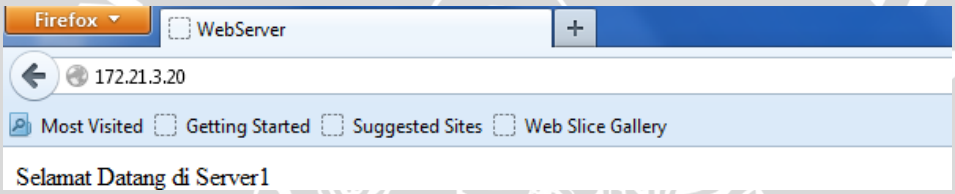
Seperti halnya pada implementasi HVM, proses instalasi yang perlu dilakukan pertama adalah instalasi sistem operasi. Sistem operasi yang digunakan yaitu Fedora 14-x86_64 dengan *graphical desktop*. Khusus pada *shared storage*, perangkat lunak tambahan yang perlu dipasang hanyalah NFS. Setelah itu, diperlukan konfigurasi berkas NFS yang terdapat pada direktori /etc/ untuk memberikan hak akses NFS kepada segmen jaringan tertentu. Pada penelitian ini, hak akses baca tulis ditujukan untuk segmen jaringan 172.21.3.0/26. Setelah konfigurasi NFS tersebut dilakukan, langkah selanjutnya yaitu proses mount direktori NFS pada seluruh HVM agar berkas yang berada di dalam *shared storage* dapat diakses. Selain itu, penulis juga mematikan sistem keamanan dari sistem operasi Fedora yaitu berupa *firewall*. Berikut hasil dari konfigurasi NFS pada *shared storage*:



Gambar 5.9 Konfigurasi NFS *Shared Storage*

5.3 Implementasi Mesin Virtual

Langkah pertama yang perlu dilakukan penulis adalah mengalokasikan sumber daya komputasi mesin virtual sesuai dengan rancangan pada Bab VI melalui perangkat lunak KVM. Setelah itu, proses instalasi sistem operasi Fedora14-x86_64 versi minimal dilakukan. Mesin virtual ini tidak memerlukan konfigurasi jaringan khusus karena alamat *Internet Protocol* akan diberikan secara otomatis oleh jaringan *Bridge*. Selain sistem operasi, perangkat lunak lainnya yang perlu dipasang pada mesin virtual ini yaitu Apache Web *server*. Perangkat lunak ini berfungsi sebagai media untuk melakukan pengujian sistem virtualisasi *server*. Berikut salah satu hasil pemasangan *Web server* virtual yang diakses melalui komputer klien:



Gambar 5.10 Web *Server* VM Satu