QEMU & KVM GUIDE #1

Intro & Basic

정정인 (call518@gmail.com)

목차

1. 시작하며

'QEMU/KVM'에 대한 이야기의 첫 시작이다. 이론보다는 "HowTo" 컨셉에 촛점을 맞췄으며, 일단은 아래 목록의 주제를 가진 대략 4개의 문서로 정리가 되지 않을까 예상된다.

- ✓ **QEMU/KVM Guide #1** - Intro & Basic : 소개 및 환경 구축, 기본 사용법 안내.
- ✓ **QEMU/KVM Guide #2** - HowTo Qemu-Monitor : VM 관리/제어 실습.
- ✓ **QEMU/KVM Guide #3** - HowTo Linux-Bridge-Networking : VM 네트워킹의 필수 Bridge 실습.
- ✓ **QEMU/KVM Guide #4** - HowTo Distributed-Storage : 분산파일시스템 Gluster, Sheepdog 실습.

첫 시작인 본 문서는 "Intro & Basic"소제목을 가지며, QEMU/KVM에 대한 간략한 소개, 그리고 실습을 위한 환경 구축과 VM 운영에 필요한 아주 기초적이고 필수적인 사항들을 짚어 보게 될 것이다. 물론 #2 문서에서 집중적으로 다룰 예정이고 가장 핵심적인 내용인 Qemu-Monitor에 대해서도 문서 끝에서 예고편 수준으로 살짝 다뤄볼 것이다.

'시작이 반'이라는 말이 있듯이, 부족한 출발이겠지만, QEMU/KVM에 대해 한번즘 정리가 필요하다는 생각에 만들게 되었으니, 봐주시는 분들에게 감사를 드리며 조금이나 유익한 문서가 되었으면 하는 바램이다.

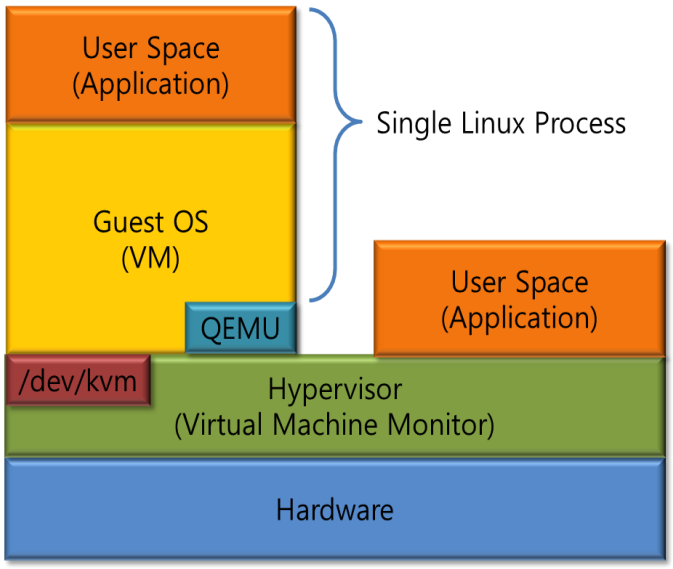
[참고문서] QEMU/KVM
Qumranet : http://en.wikipedia.org/wiki/Qumranet
Hypervisor : http://en.wikipedia.org/wiki/Hypervisor
KVM : http://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine
Qemu-Monitor : http://en.wikibooks.org/wiki/QEMU/Monitor

[참고문서] 완전가상화 / 반가상화
http://call518.tistory.com/103
http://www.ibm.com/developerworks/kr/library/l-linuxvirt/
http://www.slideshare.net/Cameroon45/understanding-full-virtualization-paravirtualization-and

● KVM? Keyboard+Video+Mouse? No~~~ Kernel based Virtual Machine!!!

KVM은 초기에, Xen에 비해 그 성능이나 성능 면에서 매우 열악했다. KVM을 처음에 접했을 때 느낌이, "아니~ 뭐이래? 쓰라는 거야 말라는 거야?"였다. 그때가 아마 2008년도였으니, 6여년이 지난 지금은 그나마 관련 정보가 많이 있지만, 당시에는 PCI장치 하나 붙이는 것도 너무 힘들었다. 맨 처음 접한 KVM Kernel Module Version이 kvm-38 이었다. 다행(?)이었던 것은, 버전업과 패치가 짧게는 2~3일 길어도 1주일만에 제공되었다는 것이다. 거의 매일 버전업하고 수정된 사항들 확인하는 것이 업무가 되어버린 기억이 생생하다. 그만큼 Virtualization분야에서는 신생아였지만, 곧바로 Linux 커널 메인스트림에 포함될 정도라 기대가 큰 놈이었다.

현재는 완전가상화 임에도 반가상화에 뒤지지 않을 만큼 성능 개선도 이룬 듯하고, 무엇보다 Guest OS에 완전한 가상환경을 제공한다는 점은 반가상화에 견주어 가장 큰 장점이라고 생각된다.



KVM의 역사는 길지 않지만 간략히 살펴보면, 출발은 Qumranet에서 시작되었으며, 2008년에 RedHat이 Qumranet을 인수하면서, 든든한 발전 기반을 다질 수 있게 되었다. 최근의 RedHat의 가상화 행보를 보게 되면 RHEL-6 출시 시점부터 Xen은 완전히 제외되고 KVM을 주력 가상화 솔루션으로 채택했다. 그만큼 기능과, 장점 그리고 장래성이 충분하다는 이야기다. 앞으로 검증 사례만 풍부히 확보한다면 가장 강력한 Hypervisor로 손꼽히지 않을까 싶다.

● KVM 가상화 구성 요소

KVM 모듈이 로드 되면(/dev/kvm 노출) Linux Kernel은 Hypervisor역할을 수행할 수 있게 된다. KVM에 특화된 QEMU는 Guest OS에 완전 가상화 된 H/W환경을 에뮬레이터 해주게 되고, Guest OS는 Host OS 입장에서 단일 프로세스로 구동되는 형태다. 그림에서 가장 하단에 Hardware가 있다. (단, 완전가상화를 이용하기 위해서는 Intel-VT나 AMD-V 기술이 지원되는 H/W여야 한다는 조건이 있다.) 그림에는

Guest OS(VM)가 하나만 표현되어 있으나, 수평적으로 다수의 VM이 있을 것이다. 여기서, /dev/kvm의 역할이 핵심이다. /dev/kvm은 캐릭터 장치로서, 가상화된 H/W를 각각의 VM에 노출 시킨다. 이 과정에서 KVM에 맞게 수정된 QEMU프로세스가 KVM모듈과 인터페이싱 한다.

✓ CPU 가상화 기술 및 KVM 모듈 확인

- CPU 가상화 기술 확인
  - 아래와 같이 INTEL, AMD일 경우가 다르나, 어느 경우든, vmx나 svm이라는 CPU Flag가 확인되어야 한다.

```
## INTEL CPU 일 경우,
# cat /proc/cpuinfo | grep vmx

## AMD CPU 일 경우,
# cat /proc/cpuinfo | grep svm
```

- KVM 커널 모듈 확인
  - 역시 INTEL, AMD 어느 경우든 그에 맞는 KVM 커널 모듈이 Load되어야 한다.

```
## INTEL
# modprobe kvm_intel
# lsmod | grep kvm
Kvm_intel               xxxxxx  0
Kvm                     xxxxxx  1 kvm_intel

## AMD
# modprobe kvm_amd
# lsmod | grep kvm
Kvm_amd                 xxxxxx  0
Kvm                     xxxxxx  1 kvm_amd
```

⚠이상의 두 가지 확인 과정에서 문제가 있다면 보유한 H/W로는 QEMU/KVM 가상화 실행이 불가능 하다.

● QEMU는 또 뭔가?

QEMU를 논하기 위해서는 "가상화란?"에 대한 주제에 대해 논하고, QEMU 아키텍처에 대해 다루어야 하나, 그런 세부 내용은 본인으로서도 좀더 깊이 있는 이해와 정리가 필요한 부분이라 여기서는 간단히 짚어보고 넘어 가자. 그래도 자세한 정보가 궁금하다 싶다면 아래 Reference Link 자료나, 웹 검색을 추천한다. 무수히 많은 자료를 접할 수 있을 것이다.

자 그럼 QEMU.... QEMU는 거의 완벽에 가까운 OpenSource 에뮬레이터로서 가장 큰 특징은 Guest CPU를 Host CPU용 명령으로 실행 중에 바꿔주는 동적 변환 기능이다. 즉, 매우 높은 이식성을 바탕으로 대부분의 H/W플랫폼(x86, PowerPC, PowerMac, ARM, SPARC 등등)을 지원한다. 본 문서는 x86을 기본으로 하고 있으며 x86플랫폼 에뮬레이터 대상으로는 CPU, CD-ROM/DVE, FDD, HDD, Video, NIC, Sound, PCI, ISA, PS/2, Parallel-Port, Serial-Interface, USB 등 거의 모든 H/W를 에뮬레이터 해낸다.

이런 장점으로 인해, KVM뿐 만 아니라, Xen(HVM), VirtualBox 등도 각자에 맞게 수정된 QEMU를 사용한다.

본 문서의 주된 내용이기도 한 VM운영에 있어서도 가장 많이 접하게 되고 다루어질 것 같다.

QEMU라는 에뮬레이터의 옵션은 상당히 많다. 모든 옵션에 대해 다루기엔 무리고, 본 문서에서 자주 사용되는 것 위주로 나열하자면 아래와 같다. 보다 자세한 부분은 QEMU사이트를 참조하고, 추가적인 부분은 VM실습 섹션에서 하나씩 새로운 것이 나올 때 마다 그때 그때 설명하는 것으로 일단락 한다. 지금은 그냥 "이런 게 있나 보다~"라고 넘어가도 된다. 실제로 사용되는 것을 보면 "아 이거구나~!"할 것이다.

QEMU 옵션	기능 설명
-boot [order=drives][,once=drives][,menu=on|off]	'drives': floppy (a), hard disk (c), CD-ROM (d), network (n)
-hda/-hdb [file]	use 'file' as IDE hard disk 0/1 image
-hdc/-hdd [file]	use 'file' as IDE hard disk 2/3 image
-cdrom [file]	use 'file' as IDE cdrom image (cdrom is ide1 master)
-drive	[file=file][,if=type][,bus=n][,unit=m][,media=d][,index=i]

	[,cyls=c,heads=h,secs=s[,trans=t]][,snapshot=on|off] [,cache=writethrough|writeback|none|unsafe][,format=f] [,serial=s][,addr=A][,id=name][,aio=threads|native] [,readonly=on|off][,boot=on|off] use 'file' as a drive image
-snapshot	rite to temporary files(/tmp) instead of disk image files
-m [memory size]	set virtual RAM size to megs MB [default=128]
-smp [n]	n[,maxcpus=cpus][,cores=cores][,threads=threads][,sockets=sockets] set the number of CPUs to 'n' [default=1] maxcpus= maximum number of total cpus, including offline CPUs for hotplug, etc cores= number of CPU cores on one socket threads= number of threads on one CPU core sockets= number of discrete sockets in the system
-k [language]	use keyboard layout (for example "fr" for French) ar de-ch es fo fr-ca hu ja mk no pt-br sv da en-gb et fr fr-ch is lt nl pl ru th de en-us fi fr-be hr it lv nl-be pt sl tr The default is en-us.
-name string	set the name of the guest set the name of the guest string1 sets the window title and string2 the process name (on Linux)
-net nic[,vlan=n][,macaddr=mac][,model=type][,name=str][,addr=str][,vectors=v]	create a new Network Interface Card and connect it to VLAN 'n
-net tap[,vlan=n][,name=str][,fd=h][,ifname=name][,script=file][,downscript=dfile][,sndbuf=nbytes][,vnet_hdr=on|off][,vhost=on|off][,vhostfd=h][,vhostforce=on|off]	connect the host TAP network interface to VLAN 'n' and use the network scripts 'file' (default=/etc/qemu-ifup) and 'dfile' (default=/etc/qemu-ifdown) use '[down]script=no' to disable script execution use 'fd=h' to connect to an already opened TAP interface use 'sndbuf=nbytes' to limit the size of the send buffer (the default is disabled 'sndbuf=0' to enable flow control set 'sndbuf=1048576') use vnet_hdr=off to avoid enabling the IFF_VNET_HDR tap flag use vnet_hdr=on to make the lack of IFF_VNET_HDR support an error condition use vhost=on to enable experimental in kernel accelerator (only has effect for virtio guests which use MSIX) use vhostforce=on to force vhost on for non-MSIX virtio guests use 'vhostfd=h' to connect to an already opened vhost net device
-pidfile [file]	Write PID to 'file'
-S	freeze CPU at startup (use 'c' to start execution)
-no-kvm	disable KVM hardware virtualization
-no-acpi	disable ACPI
-loadvm [tag|id]	start right away with a saved state (loadvm in monitor)
-vnc display	start a VNC server on display
-daemonize	daemonize QEMU after initializing

 Base-Image 방식 : 하나의 VDI이미지는 Read-Only상태로 존재(A)하며, 신규로 생성되는 VDI파일(B)은 (A) VDI파일을 출발점으로 한다. 즉 (A)의 모든 데이타를 상속 받으며, 이후 변경되는 모든 데이타는 신규로 생성된 (B)이미지에 기록되게 되는 방식이다. 이 때, 초기 상태를 결정하는 (A)이미지 파일을 Backing-File이라고 한다. 이 방식을 잘 활용한다면 Template 이미지를 사용할 수 있게 된다.

KVM & QEMU Reference Link

http://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine
http://en.wikipedia.org/wiki/Qemu
http://www.ibm.com/developerworks/kr/library/l-qemu/
http://www.ibm.com/developerworks/kr/library/l-linux-kvm/
http://www.ibm.com/developerworks/kr/library/l-linuxvirt/#Resources

용어 정리/약속

앞으로 본 문서에서 사용할 용어에 대한 정리다. 반드시 이 용어들이 정답은 아니나, 내용 흐름상 또 편의상 미리 정해두는 것이다.

- VM (또는 Guest) : 가상머신 (Virtual Machine)
- Host : 가상머신이 구동되는 실제 물리장비 (Physical Machine)
- VDI : 가상머신에 할당된 HDD 이미지 (Virtual Disk Image)

2. 환경 준비 : System & Network

KVM과 QEMU는 대부분의 Linux배포본에서 지원한다. 다만, 앞서 KVM소개에서 잠깐 언급했듯이, 기능 개선이 상대적으로 빠르다 보니, 최신 버전의 패키지들이 상위 또는 고급기능들을 지원한다. 달리 이야기 하면, 본 문서의 배포본 또는, QEMU패키지 버전이 다를 경우, 사용 가능한 명령이 상이 하거나, 제한 될 수 있음을 알아 두자.

따라서, 본 문서에서는 RHEL에서 최신 기능/패키지를 지원하는 Fedora를 사용할 것이다. 배포본 및 장비H/W, Network 구성 상세 내용은 하기 내용 참조.

● Linux 배포본

- Fedora 16 x86_6
- 파티션
  - /boot : 500MB
  - Swap : 1GB
  - / : Free Space All
- Hostname은 편의상 첫 번째 장비는 kvm1.foo.com, 두 번째는 kvm2.foo.com을 정하였다.

⚠ Host OS는 64bit를 사용을 권장한다. 32bit가 불가능한 것은 아니나, 32bit에서는 한번에 엑세스 가능한 메모리 크기가 4GB라는 태생적 한계로 인해, Guest OS에 할당 가능한 최대 Memory 크기가 2GB로 제한되거나, 64bit Guest OS를 운영할 수 없는 등의 제약 사항이 있다.

(https://help.ubuntu.com/community/32bit_and_64bit)

● H/W 스펙 (Physical Host)

- CPU : Intel SandyBridge i3-2120 3.3GHz * 1EA
- MainBoard : Gigabyte GA-H6 1M-DS2V (Micro-ATX)
- Memory : DDR4-10600 4GB * 2EA
- HDD : UTANIA SATA2 1TB (32MB/7200RPM) * 1EA
- NIC : Gigabit(Onboard) * 1EA

⚠ CPU와 MainBoard는 반드시 "가상화 기술"이 '반드시' 지원되어야 한다.

가상화 기술 지원 여부 체크 방법

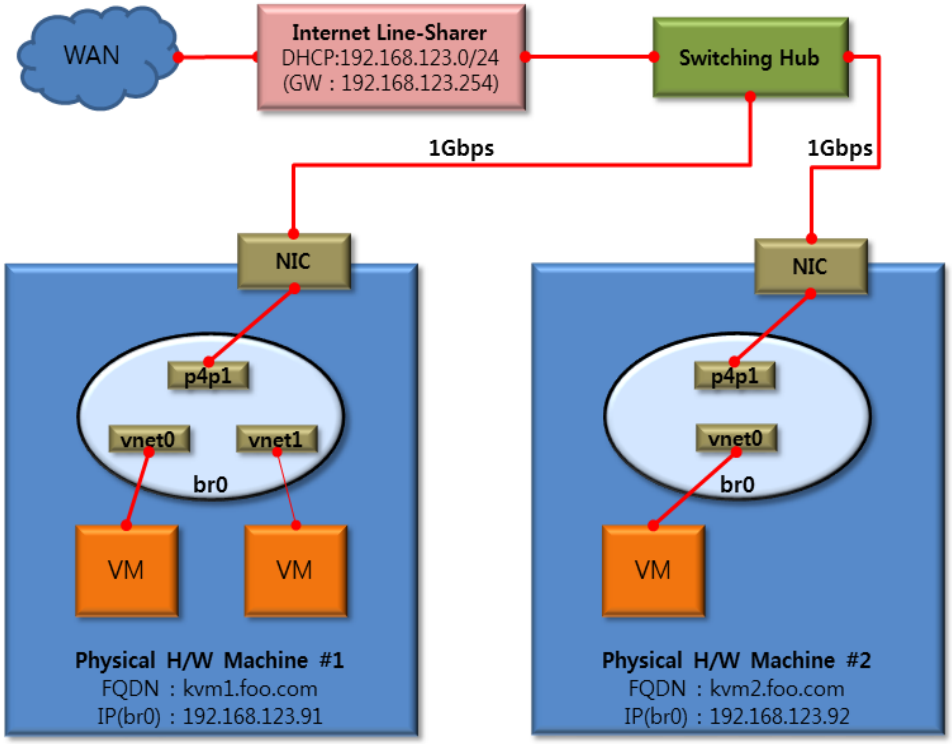
```
cat /proc/cpuinfo | egrep "vmx|svm"
flags                    : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 popcnt tsc_deadline_timer xsave avx lahf_lm arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
```

이상과 같이, CPU Flags에 vmx(Intel계열)나, svm(AMD계열) flag가 발견되어야 한다.

● Networking (Physical)

사실 Virtual-Networking에 대해서는 별도의 꼭지를 통해 이야기를 풀어야 할 분량이다.

본 문서의 주제는 "KVM & QEMU"이므로, 하기 항목들 정도만 인지하고 진행하고, Virtual-Networking에 있어서는 주어진 Networking환경에 맞추거나, 추구하는 모델을 구상하여 잘 적용해보길 바란다. 재미 있을 것이다.



- 공유기(Internet Line-Sharer)의 DHCP기능을 활용.
  - 일반 가정용 IP공유기를 활용하였으며, 공유기의 DHCP 기능을 그대로 이용.
  - 필요하다면 동일 네트워크 안에서 별도의 DHCP 운영 가능.
  - VM들에 할당될 IP-Pool의 범위는 192.168.123.101~192.168.123.250로 설정.
- 내부 통신에 사용되는 Switching-Hub는 포트당 1Gbps를 지원하는 별도 Hub사용
  - p4p1은 통상적인 Linux 배포본에서 인식되는 eth0과 같은 것으로서 Fedora에서 인식되는 명칭.
- 호스트 장비(Physical Machine)에서 br0라는 이름으로 Bridge-Networking 구성
  - 정확한 표현은 아니나, Bridge(br0)는 Linux에서 구동되는 가상의 Switching-Hub로 생각하길 바란다.
  - 각각의 Host에 존재하는, 두개의 br0는 마치 하나의 물리적인 Switching-Hub처럼 동작한다.
  - VM들이 호스트 장비와 같은 대역의 IP사용이 가능.
  - 본 구성 이외에, VM들은 별도의 사설 IP대역을 할당하고 Physical-Host의 IP(br0)로 NAT되어 외부 인터넷과 통신하게끔 하는 구성도 가능하다.
- VM들에 할당되는 가상 NIC
  - br0(브릿지)에 표현되어 있는 vnet0, vnet1등이 VM에 할당 될 가상NIC로서 tun/tap 장치 라고도 한다. 이 장치들이 QEMU에 의해 가상NIC로 에뮬레이팅 되어 VM에서는 실제 Ethernet Card로 인식된다. 자세한 설정은 이후 "OS Installation & Env. Setup" 섹션에서 설명.
- Linux Bridge-Networking 관련 자료 링크
  - http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge
  - http://ebtables.sourceforge.net/br_fw_ia/br_fw_ia.html
  - http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaat/liaatsecurity_pdf.pdf
  - http://williamherry.blogspot.kr/2012/05/bridge-by-linux-foundation.html?m=1

자, 이제부터는 배포본 설치부터, 하나씩 실제로 해보도록 하자!!!

"한번 듣는 것 보다, 한번 보는 것이 백번 낫고, 한번 보는 것보다, 한번 해보는 것이 백번 낫다.!!!"

● OS Installation & Env. Setup

앞선, System & Network 환경에서 설명된 Linux 배포본 OS 설치 과정 요약이다. 몇 단계만 거치면 되므로 크게 어려움은 없으리라 생각된다. 과정별 세부 설명은 내용상 거리가 있어 생략.

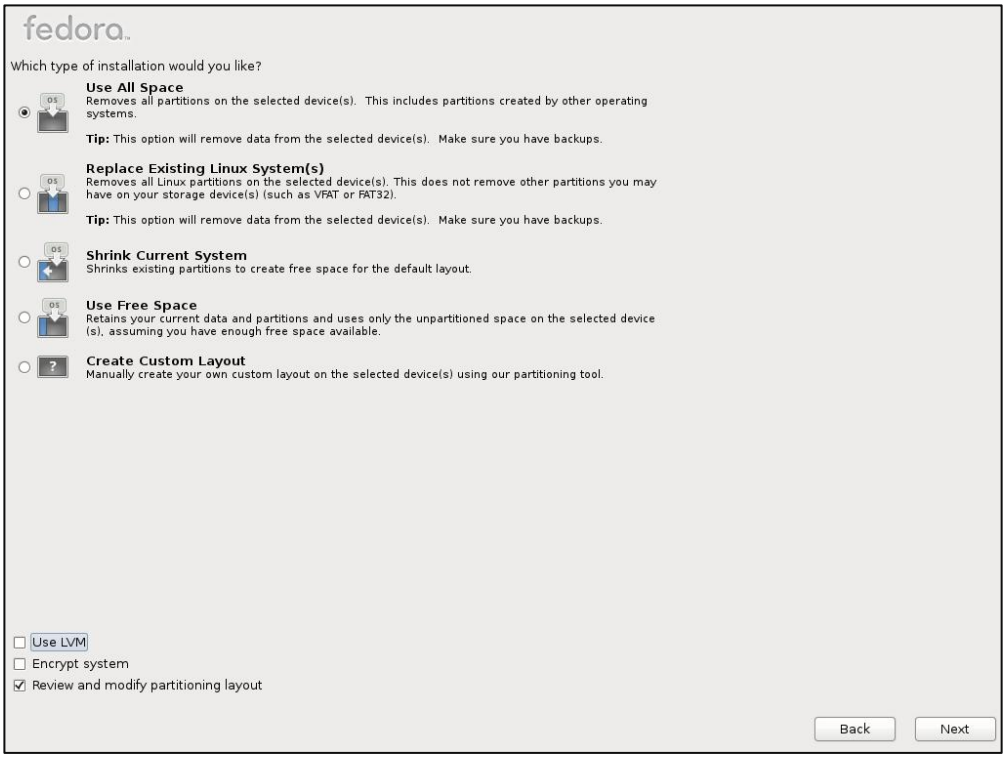
⚠ HDD를 포맷하는 과정이 있으므로, 사용중인 HDD라면 기존 데이터는 모두 유실될 것이다. 따라서, 중요 데이터가 이미 존재 한다면 반드시 별도의 저장장치에 백업을 수행한다.

✓ Installation

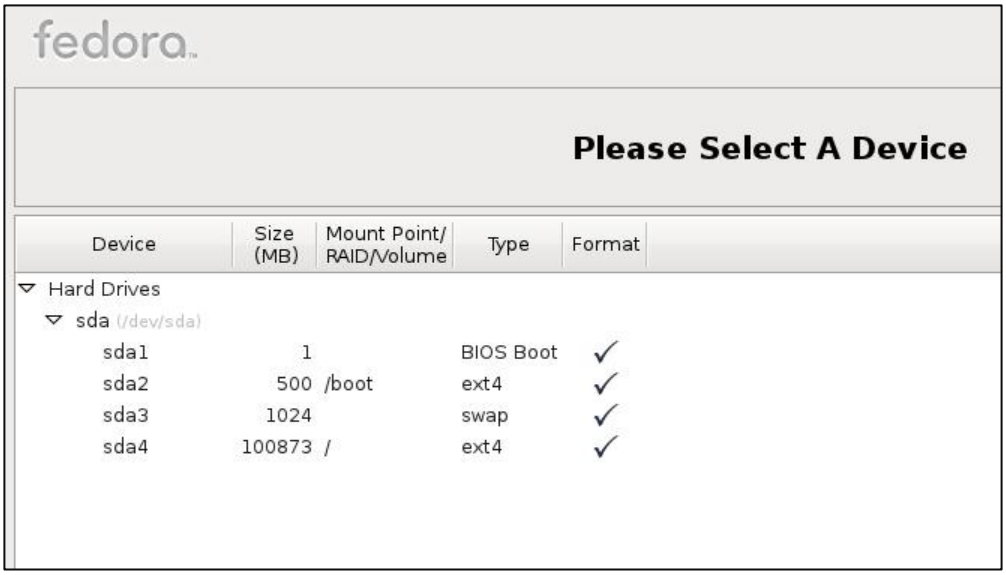
본 문서에 필요한 정보만 ScreenShot으로 안내하고, 나머지 과정은 일반적인 Linux 배포본 설치 과정을 따름.

➢ "kvm1.foo.com" 입력 하고 "Next" (두 번째 장비는 kvm2.foo.com입력)

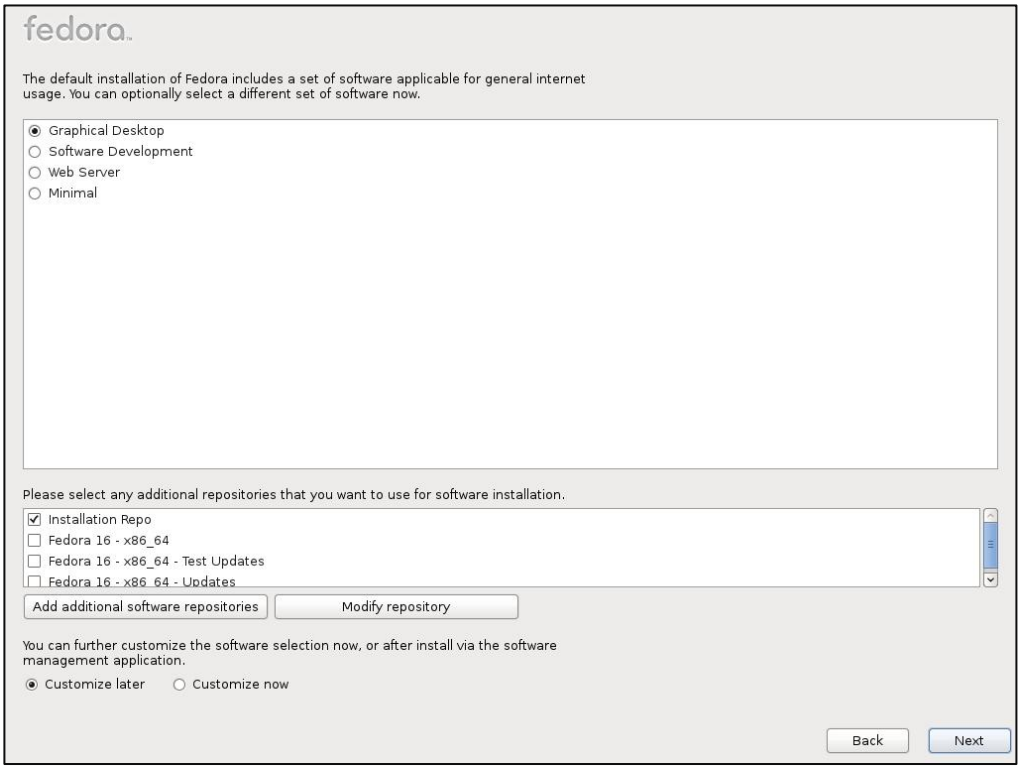
➢ "Use All Space"선택, "Use LVM"체크 해제, "Review and modify partitioning layout"체크 → "Next"



➢ "파티션 분할" → "Next" (반드시 ScreenShot과 동일할 필요는 없다.)



➢ 패키지 선택(Default 선택) "Next"



➢ Reboot 수행 및 초기 설정 과정에서 NTP 클라이언트 활성화
(나중에 다루게 될 Live-Migration등에서 Time 동기화는 매우 중요하다.)

✓ Environment Setup

자~ 설치 과정 짚어 보는 것도 일이군요. 아무튼, 이제 Gnome기반의 GUI환경으로 Login이 가능할 것이다. 물론 SSH터미널로도 정상 접근이 되는지 확인 한다. VNC를 이용하고 싶다면(본 문서에서는 꼭 설치진행), 아래 링크 내용대로 진행한다. 본 문서에서 다룰 내용은 아니라 링크로 대신하며, 부가적으로 root계정이 바로 로그인 되도록 PAM 인증도 약간 수정해야 한다. 링크 참조.....

- http://zeusville.wordpress.com/2012/01/27/setting-up-vncserver-on-fedora-16/
- http://www.linuxquestions.org/questions/linux-desktop-74/enable-root-login-in-fedora-16-security-spin-912929/

➢ udev 장치 관리 정책 변경

```
# cd /lib/udev/rules.d/disabled
# mkdir disabled
# mv 75-presistent-net-generator.rules disabled/
# mv 75-cd-aliases-generator.rules disabled/
# rm -f /etc/udev/rules.d/70-presistent-net.rules
# rm -f /etc/udev/rules.d/70-presistent-cd.rules
```

➢ selinux 비활성 처리

```
# vi /etc/sysconfig/selinux
SELINUX=enforcing
(변경)
SELINUX=disabled
# setenforce 0
```

➢ NetworkManager 제거 (사용자 편의를 위해 Wifi나, VPN, LAN 연결을 관리해주는 패키지 이나, 가상화와 관련된 Networking에서는 문제를 일으킬 가능성이 있다. 아래 명령을 통해 제거 해준다.)

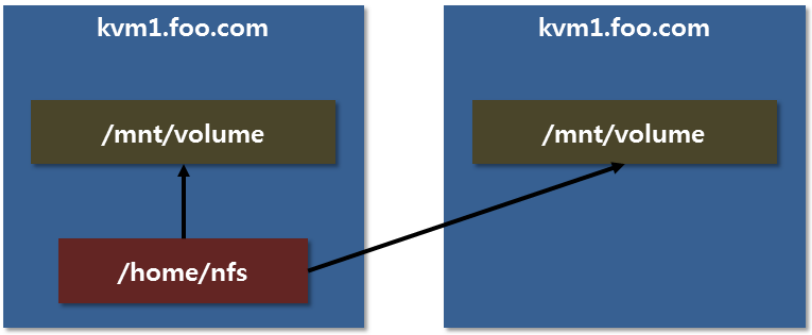
```
# yum -y remove NetworkManager
```

➢ 편의상, 두 장비의 별명(호스트네임)을 정해주기 위해 아래와 같이 /etc/hosts 파일을 작성한다.
(kvm1, kvm2 두 장비 모두에서 동일하게 처리)

```
# echo "
192.168.123.91   kvm1.foo.com      kvm1
192.168.123.92   kvm2.foo.com      kvm2
192.168.123.91nfs.foo.com       nfs" >> /etc/hosts
```

➢ VM들이 사용할 가상의 HDD용 이미지 파일을 저장하고, 더불어 Live-Migration과 같이 두 Host(kvm1, kvm2) 간에 VM을 옮겨

다닐 수 있게, 공유 스토리지 NFS를 설정해보자. 두 장비 중 한대는 NFS-Server를, 나머지 한대는 NFS-Client로 구성한다.
(간단한 실습 수준 목적이라 NFS를 사용했지만, 성능과 안정성 그리고 유용한 기능 활용성을 위해, 분산환경에서 구현되어 있는 Sheepdog이나 Gluster와 같은 파일시스템을을 사용하는 것이 좋다.)



- kvm1.foo.com (Server + Clinet 두 역할을 모두 한다.)

```
# yum -y install nfs-utils
# cat > /etc/exports << EOF
/home/nfs            192.168.123.0/24(rw,async,no_root_squash)
EOF
# exportfs -avr
# mkdir /mnt/volume
# echo " nfs.foo.com:/home/nfs   /mnt/volume       nfs       defaults,_netdev 0 0" >> /etc/fstab
# systemctl start nfs-server.service
# mount -a
```

- kvm2.foo.com (Client 역할만 한다.)

```
# mkdir /mnt/volume
# echo "nfs.foo.com:/home/nfs   /mnt/volume       nfs       defaults,_netdev 0 0" >> /etc/fstab
# systemctl start nfs-server.service
# mount -a
```

➢ VM으로 설치할 OS이미지를 다운로드 해둔다.
- 위치는 양쪽에서 공유되어야 하므로, /mnt/volume/iso로 정했다.
- Linux로는 CentOS를, Windows로는 WindowsXP를 예제로 삼았다.

```
# mkdir /mnt/volume/iso
# cd /mnt/volume/iso
# ls -al
-rwxrwxrwx 1 qemu     qemu     4423129088 Mar  7 22:02 CentOS-6.2-x86_64-bin-DVD1.is
-rwxrwxrwx 1 call518 call518  680570880 Mar  7 22:14 Windows_XP_Pro_SP3_Original.iso
```

➢ 이제 KVM 과 QEMU 관련 패키지들을 설치 해보자. 간단하다.

```
# yum -y groupinstall "Virtualization"
```

➢ Bridge(br0) & NIC 셋업
- Bridge 는 앞서 "Networking (Physical)" 섹션에서도 잠깐 언급 했으며, "Linux Bridge Networking" 이라는 파트가 따로 있을 정도로 다루어지는 범위가 넓다. 여기서는 Linux 라는 시스템 안에서 구동되는 L2 레벨의 가상의 Switch 정도로 이해하면 되겠다. 관심이 있다면 "Linux Bridge Networking"에 대해 별도로 공부를 해두는 것도 강력 권장한다. VM 을 비롯한 가상화 인프라(IaaS)에서는 매우~!매우~! 중요한 파트이기 때문이다.
- Bridge 셋업은 하기와 같이 한다.
- kvm1, kvm2 두 대 모두 각각의 IP 주소와 Hostname, Mac 주소 정보를 바탕으로 적절하게 설정한다.

⚠ Default 로 사용되던 eth0 을 변경하는 작업이니 Networking 이 단절 될 수 있다.

```
# yum -y install bridge-utils
(Bridge 네트워킹 사용을 위해 패키지 설치)

# vi /etc/sysconfig/network
(각각의 호스트네임 등록 한다.)

# vi /etc/sysconfig/network-scripts/ifcfg-p4p1
(각자 서버에 따라 p4p1은 다르게 보일 수 있다. ig : eth0)
================================
DEVICE="p4p1"
HWADDR="50:E5:49:CC:9A:00"
ONBOOT="yes"
BRIDGE="br0"
================================

# vi /etc/sysconfig/network-scripts/ifcfg-br0
================================
DEVICE="br0"
BOOTPROTO="static"
ONBOOT="yes"
TYPE="Bridge"
IPADDR="192.168.123.91"
NETMASK="255.255.255.0"
GATEWAY="192.168.123.254"
================================

# reboot
(변경된 Network 및 Hostname 적용을 위해 Reboot)

[Bridge 확인]
# brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.50e549cc9a00       no              p4p1

[Network 인터페이스 확인]
# ifconfig
br0       Link encap:Ethernet  HWaddr 50:E5:49:CC:9A:00
          inet addr:192.168.123.91  Bcast:192.168.123.255  Mask:255.255.255.0
          inet6 addr: fe80::52e5:49ff:fecc:9a00/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:43126 errors:0 dropped:0 overruns:0 frame:0
          TX packets:26316 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:65201630 (62.1 MiB)  TX bytes:1988737 (1.8 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:118 errors:0 dropped:0 overruns:0 frame:0
```

```
          TX packets:118 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:15704 (15.3 KiB)  TX bytes:15704 (15.3 KiB)

p4p1      Link encap:Ethernet  HWaddr 50:E5:49:CC:9A:00
          inet6 addr: fe80::52e5:49ff:fecc:9a00/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:50972 errors:0 dropped:0 overruns:0 frame:0
          TX packets:26246 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:66851951 (63.7 MiB)  TX bytes:1982573 (1.8 MiB)
          Interrupt:41
```

✓ VNC 클라이언트 실행 및 접속

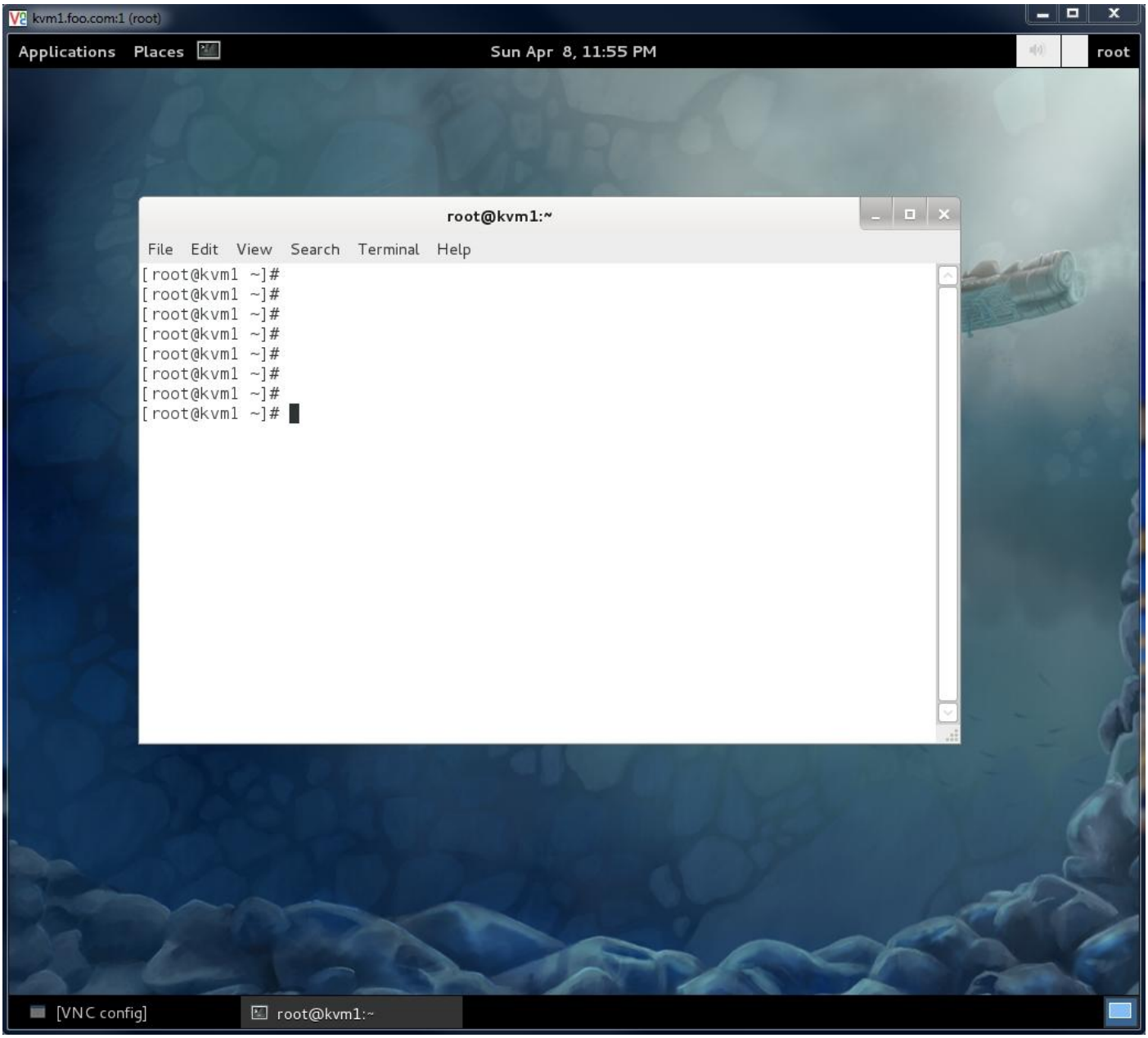
VM생성을 위해서는 GUI환경이 필요하다. (아니, 꼭 필요한 건 아니다.....)

QEMU와 KVM를 가지고, 마치 Server나 PC에 OS를 설치해보고 여러 가지 유용한 기능을 테스트 해보는 것이 목적이므로, 앞서 환경설정에서 잠시 언급했지만, 아래 두 링크 내용대로 Fedora16에서 VNC서버 설정을 해둔다.

- http://zeusville.wordpress.com/2012/01/27/setting-up-vncserver-on-fedora-16/
- http://www.linuxquestions.org/questions/linux-desktop-74/enable-root-login-in-fedora-16-security-spin-912929/

그리고, VNC 클라이언트로는 공개S/W로 RealVNC라는 도구를 추천.

RealVNC의 접속 화면. (참고로, kvm1의 VNC는 tcp/5901, kvm2의 VNC는 tcp/5902로 임의 설정했다.)

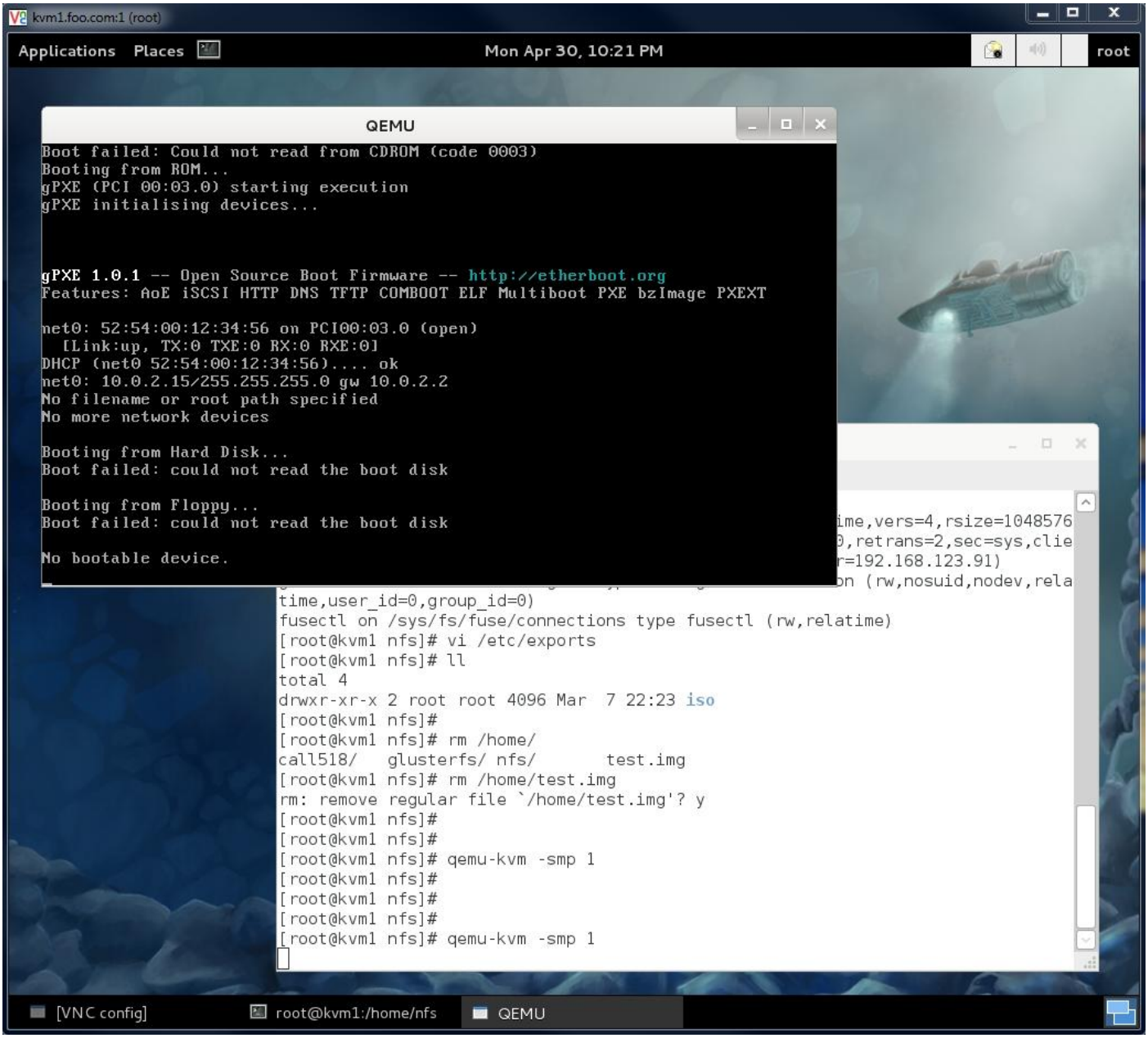


VNC를 이용해, kvm1호스트의 데스크탑 GUI화면에 접속하여 터미널 창을 하나 열어둔 화면.
이제부터 작업은 이 VNC화면에서 대부분 진행될 것이다.

3. VM 기본기 익히기

자~ 약간은 지루했을 초기 환경 설정을 어느 정도 마무리 되었다. 이제 QEMU와 KVM을 가지고, 우리가 원했던 VM을 생성해 보도록 하자.

이후의 모든 VM 생성은 오직 QEMU 에뮬레이터 명령만 가지고 컨트롤 할 것이다. Libvirt라는 하이퍼바이져 컨트롤러의 API를 활용하는 virt-manager등의 GUI툴도 있지만, 이 문서의 취지는 QEMU가 실제로 어떻게 VM을 구동시키고, 컨트롤 하는지에 맞춰있기에 모든 실행은 명령라인(쉘)을 통해 이루어 질 것이다. 그렇다고 어렵거나 하지는 않을 것이다. 간단한 것부터, 또 되도록 자세한 주석과 설명을 붙일 것이므로....



정말 간단히 VM을 실행한 화면이다. 단지 CPU만 1개(-smp 1)라는 옵션만 부여한 뒤, VM을 Booting시킨 결과다. 당연히 Booting될리가 없다. 하드디스크(HDD)가 없으니, OS가 설치되어 있을 리도 없다. 그렇다고 CD-ROM과 같은 ODD도 없으니, Booting같은게 될 리 없다. 이제부터 VM이 부팅될 수 있는 조건에는 무엇이 필요한지 하나씩 알아가면서 갖추어 나가도록 하자.

● VDI 관리 도구 qemu-img

VM생성에 앞서, 먼저 알아두어야 하고, 앞으로의 과정에서도 매우 중요한 도구이다. qemu-img라는 도구는 VM에 사용될 가상의 하드디스크 이미지(이하 VDI)를 생성/삭제/스냅샷/변환 등의 작업을 가능케 해주는 필수 도구이다. 일반적인 사용법은 아래와 같다.

⚠ VDI = Virtual Disk Image의 약자로 VM에서 사용될 가상의 하드디스크 이미지 파일을 의미.

✓ qemu-img 사용법

10GB 용량의 VDI 생성/확인 예제

(앞서 작업했던 NFS 공유 스토리지 마운트 위치로 이동)
cd /mnt/volume

(편의상 images 디렉토리를 하나 만들고 진입한다. 이후 모든 VDI이미지는 이곳에 작성한다.)
mkdir images
cd images

(이제 VDI를 생성해본다.)
qemu-img create test.img 10G
Formatting 'test.img', **fmt=raw** size=10737418240
ls -alh test.img
-rw-r--r-- 1 root root **10G** Apr 30 22:18 **test.img**
(10GB 사이즈로 test.img라는 파일이 생성된 것을 확인. 이 파일이 VM의 OS등이 설치 될 가상의 하드디스크 VDI파일이다.)

 VDI에는 "Format"이라는 것이 있다. 위 예제와 같이 특별히 포맷을 지정하지 않으면, "raw"타입의 포맷으로 VDI가 생성된다. raw타입은 단순하고, 그로 인해 빠르다는 장점이 있으나, test.img 파일과 같이 지정한 사이즈 10GB를 처음부터 잡아 먹고 시작한다. 혹, Thin-Privisioning라는 단어를 들어 보았는지 모르겠으나, "qcow2"라는 포맷을 지정하여 사용하면, 전체 크기 10GB는 잡혀 있으나, 초기 생성시에는 VDI파일이 약 200kB정도의 사이즈로 생성된다. 이후, OS를 설치 한다던가 하는 작업이 발생하면 그때 VDI파일의 사이즈가 사용한 만큼씩 증가 되는 것이다. 이것을 Thin-Provisioning라고 하며, 한정된 물리적인 Disk 저장 공간을 다수의 VM들이 효율적으로 사용하는 것이 가능하다. 본 문서에서는 이 "qcow2" 포맷을 기본으로 삼을 것이다.

(qcow2 포맷으로 생성)
qemu-img create test2.img 10G -f qcow2
Formatting 'test2.img', **fmt=qcow2** size=10737418240 encryption=off cluster_size=65536

ls -alh test2.img
-rw-r--r-- 1 root root **193K** Apr 30 22:28 **test2.img**

qemu-img info teste2.img
image: test2.img
file format: qcow2
virtual size: 10G (10737418240 bytes)
disk size: 136K
cluster_size: 65536

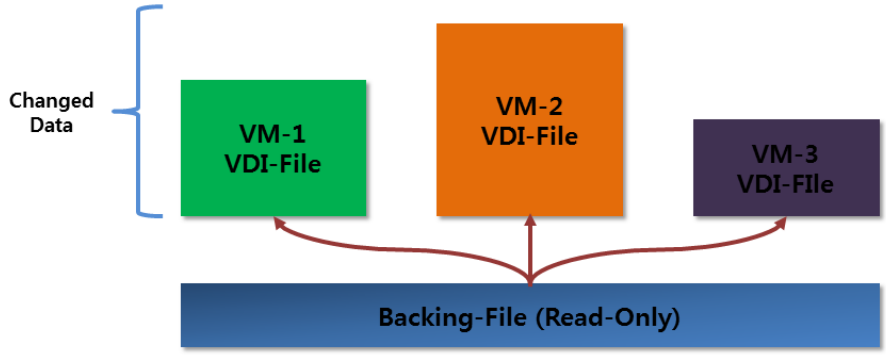
✓ qemu-img 옵션 설명

주로 사용되는 옵션은 하기와 같다. 이 외에도 몇 가지 옵션들이 더 있으나, 사용할 수 있는 조건이 제한적이며 또 매우 특별한 경우가 아니면 거의 이용이 되지 않으니, 우선은 하기 내용만 완벽히 숙지 하도록 한다.

옵션	설명
check	VDI의 일관성 체크.
create	신규 VDI 생성.
commit	**Base-Image 방식**으로 생성되었을 경우, VDI의 내용을 Backing-Image파일에 병합.
convert	qemu-img에서 지원하는 Format간의 변환 처리.
info	VDI의 메타 정보를 확인.
snapshot	VDI에 대한 스냅샷을 관리 (생성/삭제/확인)
resize	VDI의 전체 크기를 재조정. (증가는 상관 없으나, 축소시에는 주의가 필요하다.)

✓ Base-Image 방식 (Backing-File)

qemu-img도구는 Base-Image라는 기능을 제공한다. 무엇이냐고? 음... 간단히 말하면 "가지 치기" 정도? 그냥 아래 그림을 보자.



아래쪽 파란색으로 표시된 VDI파일(Backing-File)이 Base-Image로서, VM-1 VM-2, VM-3라는 Guest머신들의 VDI파일들의 모태격인 역할을 한다. 즉, 각각의 VM들의 초기 상태는 Backing-File로부터 파생되어 모두 동일하게 출발하게 되나, 사용되면서 변경이(삭제/추가) 가해지는 모든 사항은 VM각자의 VDI파일에 기록되게 되는 것이다.

⚠ Base-Image 방식에서 주의 해야 하는 것은, 눈치 챘겠지만, 윗 그림에서 파란색인 Backing-File이 손상되거나 삭제되게 되면, 그 파일을 Backing-File로 사용하는 모든 VM들의 VDI는 연쇄적으로 손상된다. 따라서, 편리하고 효율적인 방법이지만, 이런 상대적인 단점도 있으니, VDI파일들의 연관관계에 항상 주의를 기울여야 한다.

자 그럼 실제로, qemu-img를 이용해, Base-Image 타입의 VDI를 만들어 보자.

Backing-File 지정은 "-b" 옵션을 사용하며, 신규로 생성(사실은 Backing-File에 링크 형태)되는 VDI에 대한 용량 지정은 없다. 이유는? 당연히 Backing-File의 용량까지 그대로 상속받게 되기 때문이다. "qemu-img info" 명령을 통해 VDI정보를 보면 앞서 단독으로 생성했던 VDI 정보와는 다른 것을 확인 할 수 있으며, 이렇게 생성된 VDI의 실제 파일 사이즈는 136KB정도이나, VM구동시 Backing-File에 있는 정보를 곧바로 이용이 가능하다. 만일 OS라면 즉시 부팅 및 서비스가 가능하다는 이야기다.

```
# cd /mnt/volume/images

# qemu-img create -b /mnt/volume/images/Default-CentOS.img -f qcow2 Test-Base-Image.img
Formatting 'Test-Base-Image.img', fmt=qcow2 size=53687091200 backing_file='/mnt/volume/images/Default-CentOS.img' encryption=off cluster_size=65536

# qemu-img info Test-Base-Image.img
image: Test-Base-Image.img
file format: qcow2
virtual size: 50G (53687091200 bytes)
disk size: 136K
cluster_size: 65536
backing file: /mnt/volume/images/Default-CentOS.img (actual path: /mnt/volume/images/Default-CentOS.img)
```

이렇게 Backing-File을 뒷 단에 "링크" 형식으로 걸고 생성하게 되면, OS가 이미 설치된 VDI를 아주 빠르게 복제 해 낼 수 있는 것이다. 다만, Backing-File에 의존적이라는 단점이 있지만, 실제 대규모 VM인프라 운영 시에 아주 유용한 방식이기도 하다. (반대로 Disk 사용량 효율성이나 복제 소요시간 면에서는 전체복제가 불리하나 단독 운영이라는 측면에서 보면 안정성은 더 높게 보는 시각도 있다.)

다음 내용에 나올 VM OS설치 과정에서 만들어 보게 될 Template 파일인 Default-CentOS.img라는 이름의 VDI를 Backing-File로 두고 새로운 VM을 생성해 보도록..... 그리고 부팅까지 시켜 보면 "아~ 이거구나~!" 할 것이다.

⚠ 본 문서에서는 Template파일을 전체복제 방식과 Base-Image 방식이 내용상 필요에 의해 혼용 될 것이다. Default는 Base-Image방식을 우선시 함.

✓ VDI 이미지 변환

Qemu-img는 여러가지 VDI 포맷을 지원한다. 그리고 각각의 포맷간 변환(Convert)도 가능하다. 본 문서에서는 별도로 다루진 않을 것이나, 간단히 변환하는 테스트만 해보고 넘어 가도록 하겠다.

음, Linux KVM 기반의 VM이외에 대중적으로 많이 쓰는 것이, VMWare일 것이다. VMWare에서 사용하던 이미지(vmdk포맷)을 가져와 KVM

QEMU/KVM으로 구동시키기 위해 qcow2포맷으로 변환하거나, 또는 그 반대도 있을 수 있겠다.

여기서는 앞서 만들어둔 이미지도 만들어 봤으니, qcow2를 vmdk포맷으로 변환 테스트를 해보도록 하자.

```
# qemu-img create -f qcow2 qcow2.img 10G
Formatting 'qcow2.img', fmt=qcow2 size=10737418240 encryption=off cluster_size=65536

# qemu-img convert -f qcow2 -O vmdk qcow2.img vmdk.img

# qemu-img info vmdk.img
image: vmdk.img
file format: vmdk
virtual size: 10G (10737418240 bytes)
disk size: 12K
```

먼저, 10G 크기, 그리고 qcow2포맷으로 qcow2.img라는 이름의 VDI파일을 생성했다.

이후, vmdk 포맷으로 변환 처리하고, info 결과에서 정상적으로 변환되었음을 확인 할 수 있다.

지금은 아무 데이터가 없으므로, 빠르게 변환되나, VDI내에 많은 데이터가 이미 기록되어 있다면, 변환에 많은 시간과 시스템 부하가 걸릴 수 있음을 알아 두자.

참고로 아래는 현재 Fedora16 버전에 포함된 qemu-img가 지원하는 VDI 포맷 목록이다.

```
Supported formats: rbd tftp ftps ftp https http host_cdrom host_floppy host_device file blkverify sheepdog blkdebug nbd parallels qed qcow2 vvfat vpc bochs dmg cloop vmdk vdi qcow cow raw
```

● VM 생성 및 OS설치

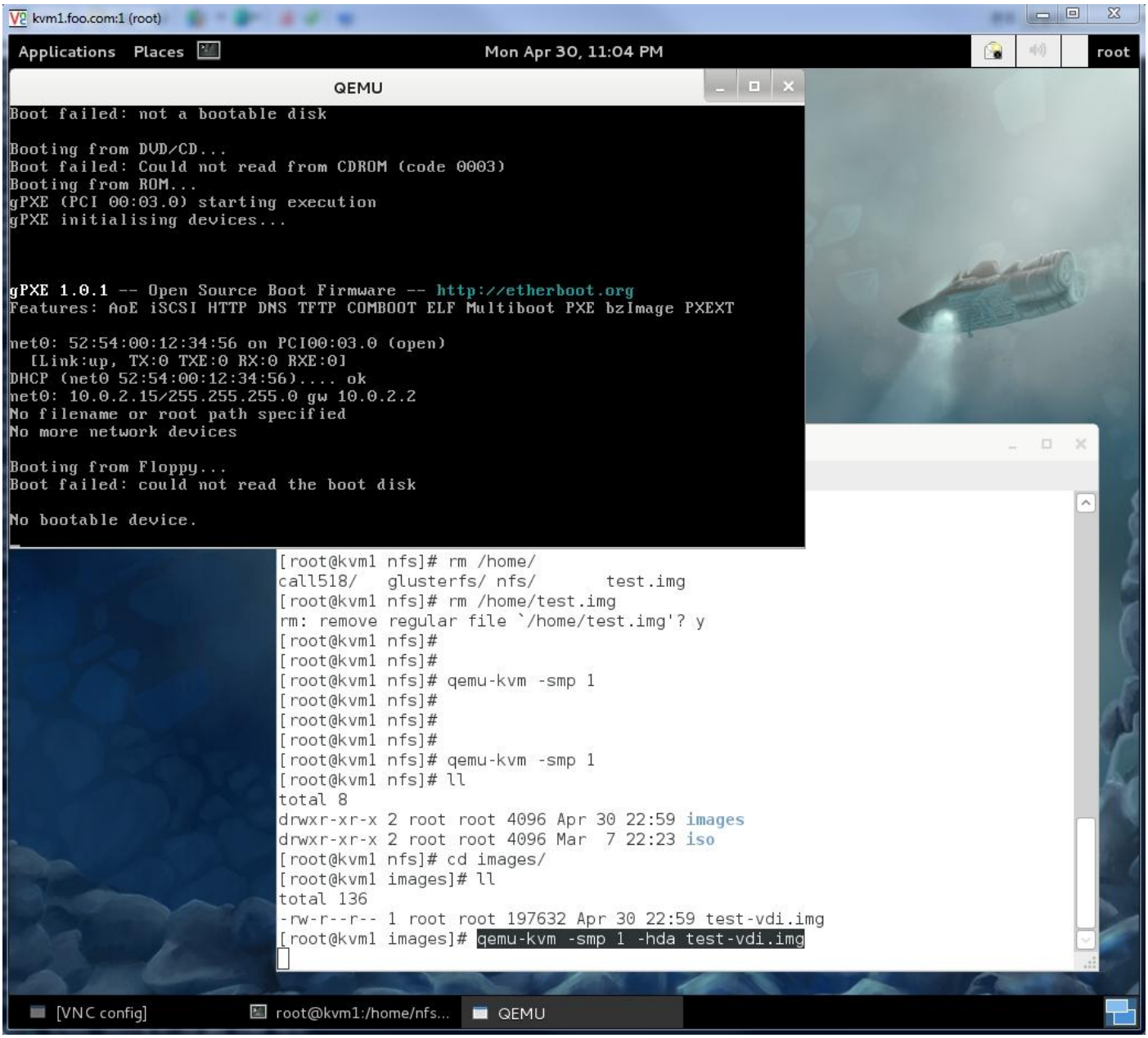
✓ 가상 하드디스크 드라이브(VDI) 연결

자 앞서 생성했던 VM에는 HDD가 없었다. 우리는 이제 qemu-img를 알았으니, 적당한 크기의 HDD를 만들어 붙여주자. 사이즈는 적당히 50GB짜리, qcow2 포맷으로 만들자.

```
# cd /mnt/volume/images
# qemu-img create -f qcow2 test-vdi.img 50G
# ls -al
total 144
drwxr-xr-x 2 root root     4096 Apr 30 22:59 .
drwxr-xr-x 4 root root     4096 Apr 30 22:17 ..
-rw-r--r-- 1 root root 197632 Apr 30 22:59 test-vdi.img
```

VDI도 만들었으니, 이제 VM을 구동 시켜 보자.

```
# qemu-kvm -smp 1 -hda test-vdi.img
```



그래도 여전히 "No bootable device"라는 메시지와 함께 부팅되는 화면을 볼 수 없다. 당연하다. 방금 장착해준 test-vid.img라는 HDD는 PC로 치면 이제 막 구입한 HDD를 장착한 상태와 같기 때문이다. 그럼 어떻게 해야 하나? 당연히 PC에 OS를 설치 하듯이 진행하면 되겠다. 당장 쉽게 떠오르는 것이 무엇? 그렇다. CD-ROM에 원하는 설치용 CD를 넣고 OS를 설치 해보는 것이다.

✓ VM에 CentOS설치 (with CD-ROM)

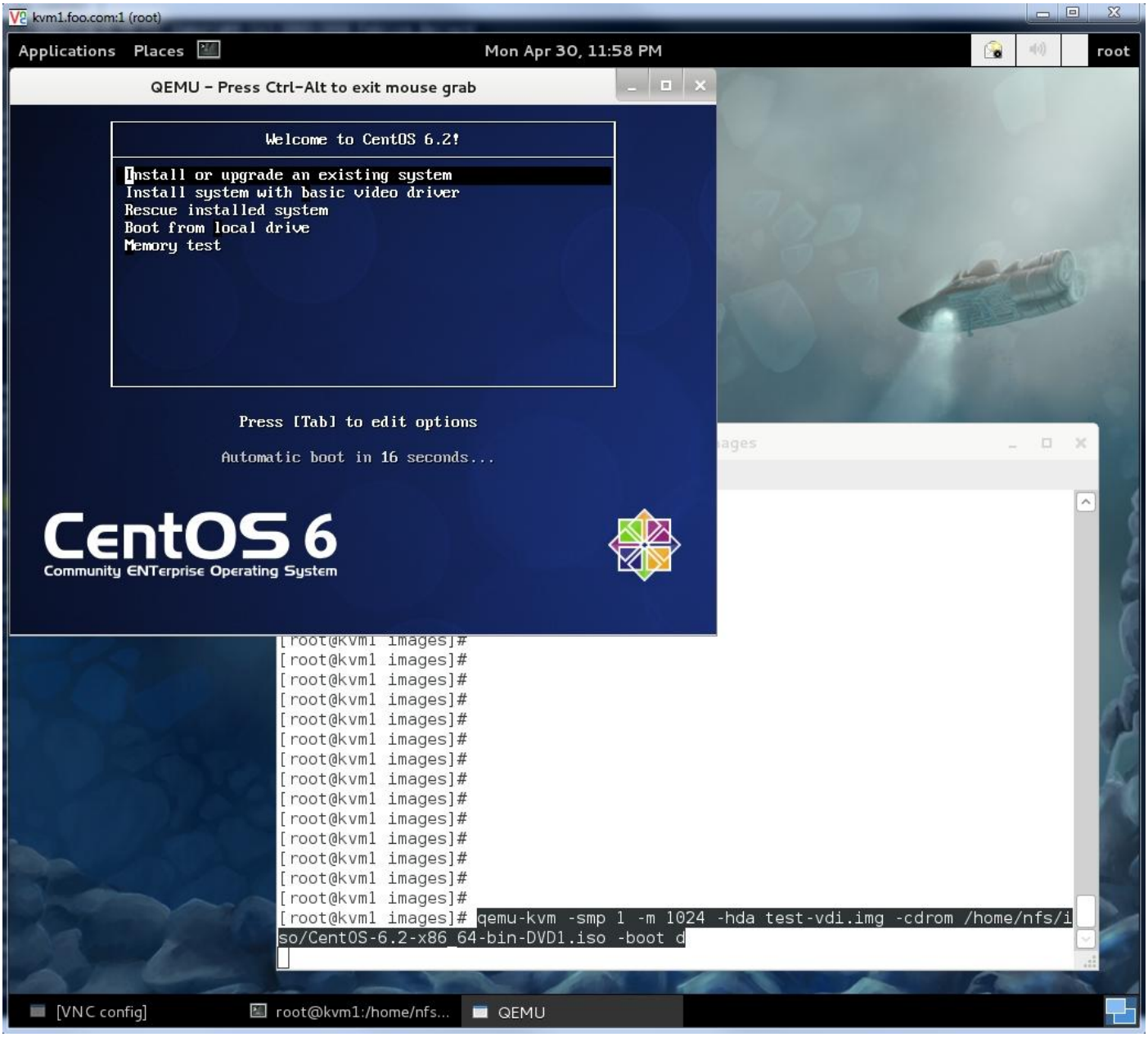
자, CD-ROM도 달아주고, 설치 CD(iso 파일이다.)도 넣어주고 부팅시켜 보자.

⚠ CD-ROM에 삽입할 CD이미지 파일(ISO)는 미리 /mnt/volume/iso 디렉토리에 다운로드 해두었다. 이번 테스트에서는 CentOS라는 리눅스를 설치 하는 것을 테스트 해본다.

```
# qemu-kvm -smp 1 -hda test-vdi.img -cdrom /mnt/volume/iso/CentOS-6.2-x86_64-bin-DVD1.iso -boot d
```

⚠ 옵션 해설)

-smp	CPU(Core)수를 지정. 1개의 CPU를 할당했다.
-m	메모리가 필요하다. 1024MB(1GB)를 할당했다.
-cdrom	CD-ROM 장치를 장착하며, 사용할 CD ISO파일 위치를 지정한다.
-boot	VM의 Booting 우선순서를 지정한다. 여기서 사용된 (d)는 CD-ROM이다.



자, CentOS6 버전의 Linux설치 화면이 로딩되었다. 간단하지만, OS설치를 위한 VM구동 예제를 해본 것이다. 이후, CentOS설치 과정은 일반적인 설치 과정과 같으므로, 생략하고 설치 완료로 넘어 가자.

⚠ (주의) : Host장비 설치 때 수행했던 udev 및 selinux 비활성 처리를, VM의 OS도 설치 후, 동일하게 처리 해둔다. selinux 보안 정책이, 이 후 실습해 볼 사항들에 방해될 수 있고, udev의 경우 VM의 Template을 통한 복제시 NIC장치 인식에 문제를 일으킬 수 있기 때문이다.

✓ Daemon으로 실행하기

앞서 실습한 모든 것은 Foreground로 실행 한 것이다. qemu-kvm을 실행한 쉘에서 Ctrl+C를 입력하거나, VM창을 닫게 되면 VM도 종료되버린다. 이때, Daemon형태로 QEMU를 실행할수 있는데, 방법은 아래와 같이 **-daemonize** 옵션을 추가한다. VM을 웹서버와 같은 지속적인 서비스로 구동할 경우에는 Daemon형태로 실행하는 것이 유리할 것이다.

```
# qemu-kvm -daemonize -smp 1 -hda test-vdi.img -cdrom /mnt/volume/iso/CentOS-6.2-x86_64-bin-DVD1.iso -boot d
```

● CentOS VM 부팅 (from VDI)

CentOS 설치를 완료 했다. 이 과정에서 우리는 VDI파일로 test-vdi.img라는 파일을 qcow2포맷으로 사용했었다. 잠깐 qcow2포맷의 정보를 살펴보자.

```
# ls -alh test-vdi.img
```

```
-rw-r--r-- 1 root root 2.9G May  1 15:26 test-vdi.img

# qemu-img info test-vdi.img
image: test-vdi.img
file format: qcow2
virtual size: 50G (53687091200 bytes)
disk size: 2.8G
cluster_size: 65536
```

처음에 200kB정도 였던 것이, CentOS를 설치 하고나니, 약 3GB정도로 증가 되었다. 앞서도 설명했던 Thin-Provisioning의 특징이다.
자, 이제, Booting가능한 OS도 설치 했으니, VDI를 통해 부팅을 해보자. (Windows설치는 부록 형태로 다루겠다.)

⚠ test-vm.img는 CentOS 초기 설치상태이므로, 이 파일을 Default-CentOS.img라는 이름으로, 파일명을 변경하자. 추후 사용할 일이 많다.

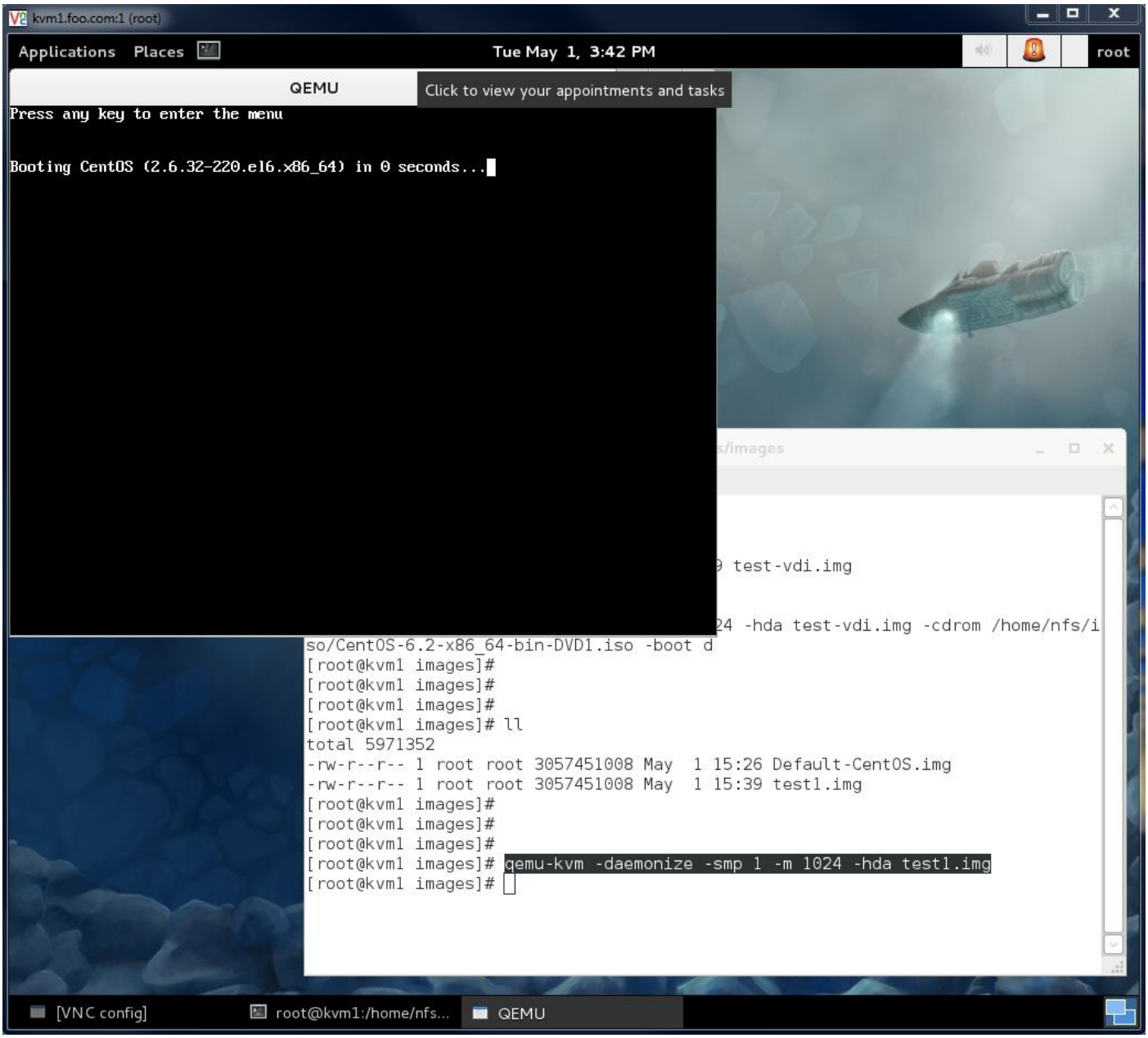
```
# pwd
/mnt/volume/images
# mv test-vdi.img Default-CentOS.img
# ls -al
total 2985684
drwxr-xr-x 2 root root          4096 May  1 15:38 .
drwxr-xr-x 4 root root          4096 Apr 30 22:17 ..
-rw-r--r-- 1 root root 3057451008 May  1 15:26 Default-CentOS.img
```

이제, 실습할 VDI를 Default-CentOS.img로부터 하나 복제 한다. 적당히 test1.img라고 했다.

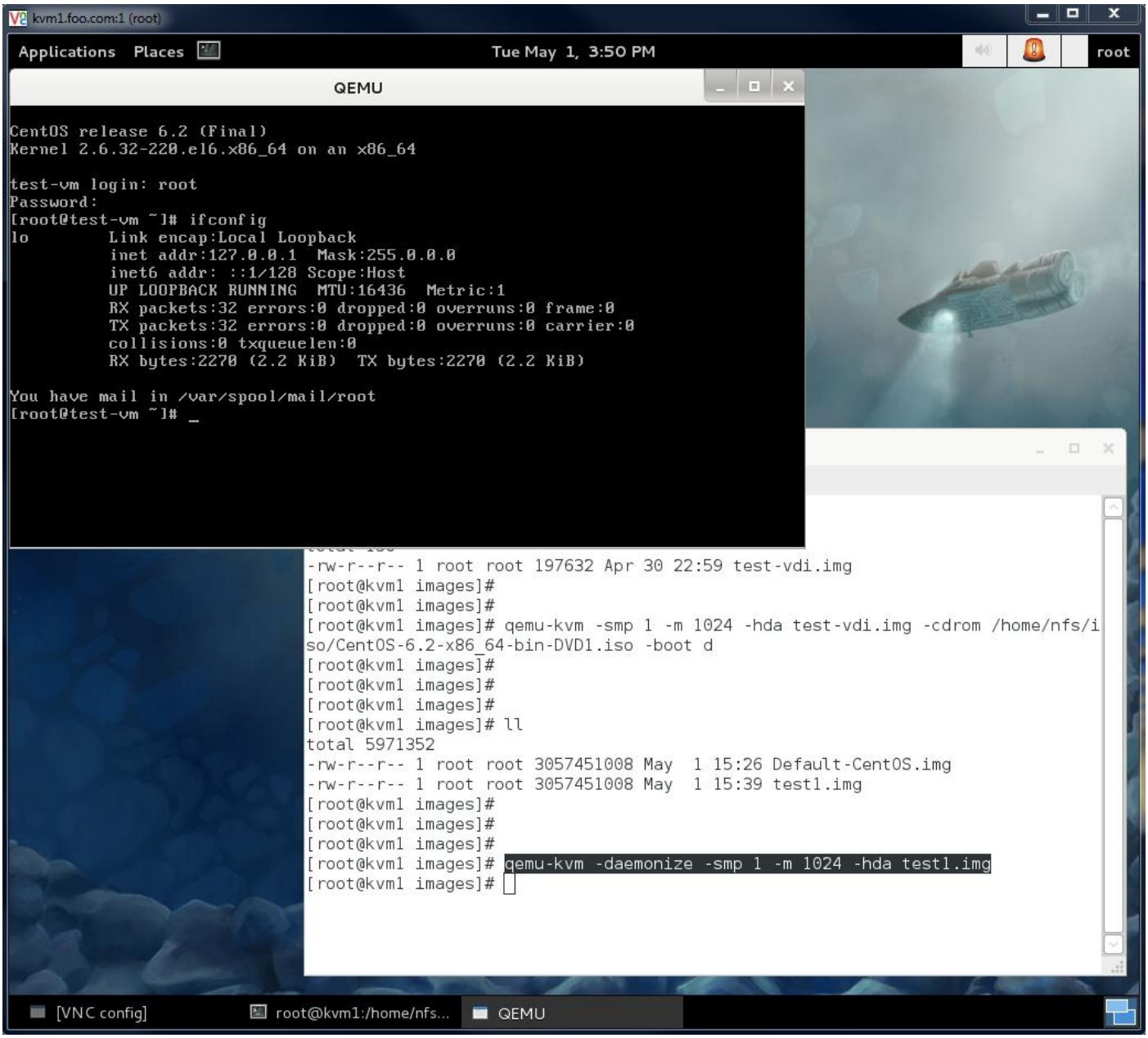
```
# cp Default-CentOS.img test1.img
# ls -al
total 5971360
drwxr-xr-x 2 root root          4096 May  1 15:39 .
drwxr-xr-x 4 root root          4096 Apr 30 22:17 ..
-rw-r--r-- 1 root root 3057451008 May  1 15:26 Default-CentOS.img
-rw-r--r-- 1 root root 3057451008 May  1 15:39 test1.img
```

이제, 이 test1.img를 이용해 부팅을 시도 해보자. 당연히 CD-ROM은 필요 없다.
(실습 겸, Daemon 형태로 실행 해보자. 이렇게 실행하게 되면 명령 프롬프트가 다시 떨어진다.)

```
# qemu-kvm -daemonize -smp 1 -m 1024 -hda /mnt/volume/images/test1.img
```



Linux 부팅 시, 자주 볼 수 있는, Grub 부트로더 화면이다.



CentOS Linux가 Text모드로 부팅되었으며, 최고관리자인 root로 Login도 성공한 스크린샷이다.

자, 여기까지는 아주 기본적인 VM 생성 실습이었다. 눈치 챘는지 모르겠으나, 아주 중요한 Network장치는 전혀 부여하지 않은 상태라 인터넷 연결 같은 Networking작업은 불가능한 VM이다. 스크린샷에서 볼 수 있듯이, lo라는 loopback장치만이 덜렁 있을 뿐이다.

여기까지 실습의 목적은 qemu-img의 용법과, qemu-kvm을 통해, VM이 어떻게 구동되며, OS설치는 어떻게 하는지를 알아보기 위함이었으니, 이 두 가지에 대해서만 이해가 되었으면 다음으로 넘어 가자.

● VM Networking

OS도 설치 했고, 부팅도 되었지만, Networking이 안되면 말 그대로 "무엇에 쓰는 물건인고?"이다.

Networking을 하기 위해서는, 사전작업이 좀 필요하다. 환경준비-Networking에서 수행했던, Bridge(가상 L2-Switch)에 연결될 Tap/Tun 장치를 만들어 줘야 하고, 그 장치를 VM이 NIC로 인식하게끔 Mac Address정보와 함께 QEMU를 통해 연결 해주어야 한다. 아래 과정을 통해 세부 과정을 살펴보고, 이 후 부터는 Networking관련 작업을 자동화 하기 위해 Script도 만들어 보도록 하자.

(도입부에서도 언급했지만, Linux Bridge Networking은 별도의 문서로 다루어야 할 정도의 기능과 범위를 가진다. 특히 대규모 VM운영 환경에서는 성능과 직결되는 중요한 요소이다. 이 파트는 추후 별도의 문서에서 다시 다뤄보도록 하자.)

✓ tun/tap 인터페스 생성

tunctl이라는 명령어가 있다. 사용법은 아래와 같으며, 앞서 설명한대로 VM에 NIC로 사용되며, 그와 동시에 Bridge에 연결될 장치이다.

```
# tunctl -u root -t vnet0
# ifconfig
br0       Link encap:Ethernet  HWaddr 50:E5:49:CC:9A:00
          inet addr:192.168.123.91  Bcast:192.168.123.255  Mask:255.255.255.0
          inet6 addr: fe80::52e5:49ff:fecc:9a00/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:66934 errors:0 dropped:0 overruns:0 frame:0
          TX packets:77701 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:72878432 (69.5 MiB)  TX bytes:59775235 (57.0 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:150 errors:0 dropped:0 overruns:0 frame:0
          TX packets:150 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:18556 (18.1 KiB)  TX bytes:18556 (18.1 KiB)

p4p1      Link encap:Ethernet  HWaddr 50:E5:49:CC:9A:00
          inet6 addr: fe80::52e5:49ff:fecc:9a00/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:77431 errors:0 dropped:0 overruns:0 frame:0
          TX packets:77625 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:75259457 (71.7 MiB)  TX bytes:59768599 (56.9 MiB)
          Interrupt:42 Base address:0x2000

vnet0     Link encap:Ethernet  HWaddr 3E:C0:4C:C4:EF:CF
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

굵게 표시된 장치가 새로 추가되었다. 이 장치를 이제 가상스위치인 Bridge에 추가해보자.

```
# brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.50e549cc9a00       no              p4p1

# brctl addif br0 vnet0

# brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.3ec04cc4efcf       no              p4p1
                                                        vnet0
```

자, br0라는 이름으로 만들었던 Bridge에 방금 만들었던 tun/tap장치 vnet0가 정상적으로 추가 되었다. 이제 이 vnet0라는 NIC를 VM에 장착해서 실행해야 하는데, 그전에 NIC에는 각각 고유의 Mac-Address라는 정보가 필요하다. 아래와 같이 간단히 생성할 수 있다.

⚠ Mac-Address는 동일 L2 영역에서 중복이 없어야 한다. (VM마다 다른 Mac-Address를 생성하여 부여 하여야 한다.)

```
# python -c 'import random; r=random.randint; print "00:16:3E:%02X:%02X:%02X" % (r(0, 0x7f), r(0, 0xff), r(0, 0xff))'
```

00:16:3E:79:A2:D8

위와 같이, **00:16:3E:79:A2:D8** Mac-Address가 Random하게 생성되었다.

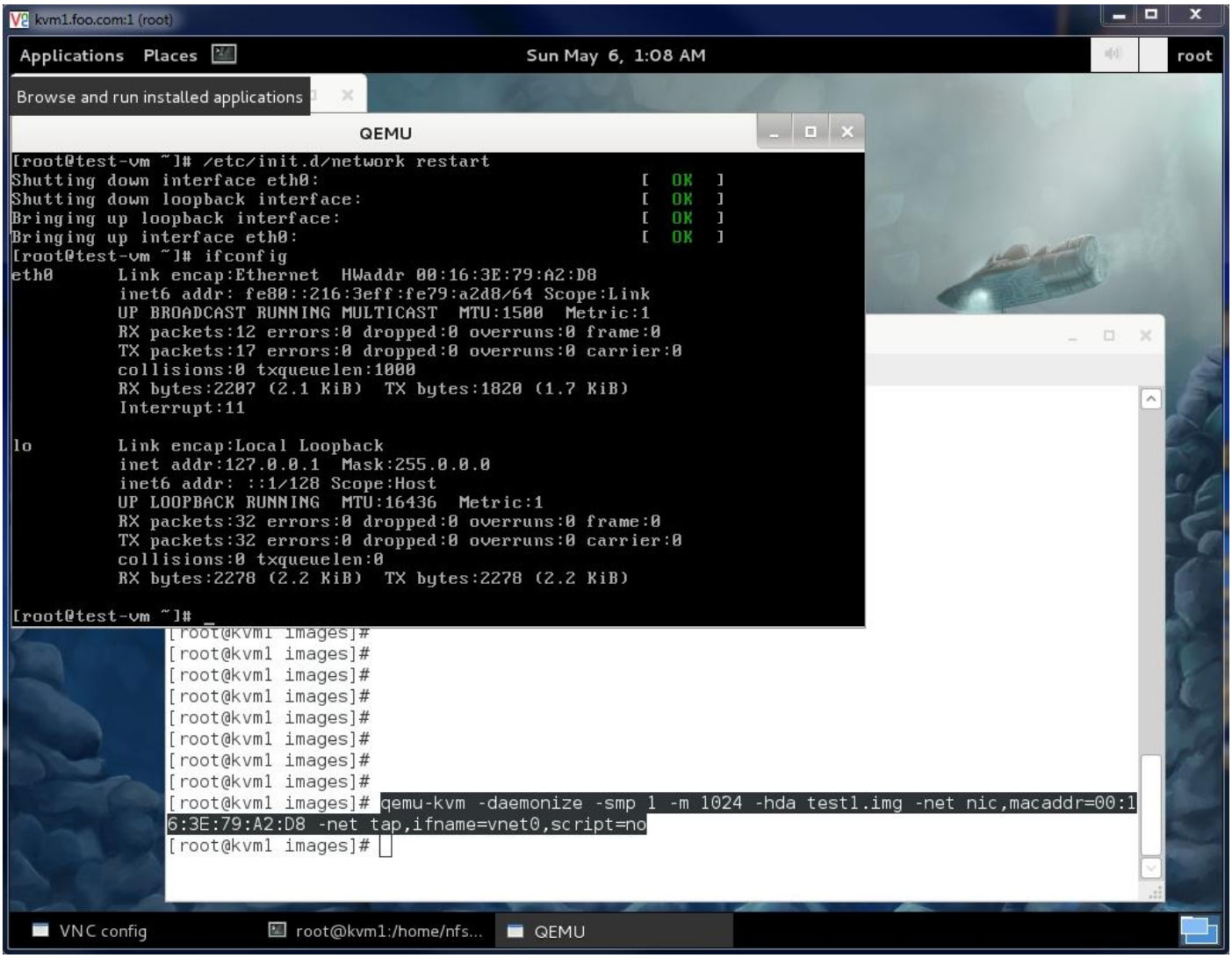
이제, 이 Mac-Address로 VM을 구동한다.

```
# qemu-kvm -daemonize -smp 1 -m 1024 -hda /mnt/volume/images/test1.img -net nic,macaddr=00:16:3E:79:A2:D8 -net tap,ifname=vnet0,script=no
```

(옵션 설명)

-net	Networking 장치 정보를 정의할 때 사용하는 옵션
macaddr={Mac-Address값}	사용할 Mac-Address 정의.
tap	NIC 장치 타입 정의
ifname={tun/tap장치명}	NIC 장치명 정의.
script={장치 자동생성 Script}	tunctl 및 brctl 명령의 수작업을 자동화 시킨 script 사용 여부 정의.

자, 아래 스샷처럼 eth0이라는 NIC(Network Interface Card)가 정상적으로 인식되어 있다.



이제, 이 eth0에 IP를 할당할 차례다. 우리는 IP공유기의 DHCP가 자동으로 IP를 주도록 환경 준비를 했으므로, , VM내에서 하기와 같이 eth0의 설정파일을 정의/생성해주고 Network를 재시작 하면 된다.

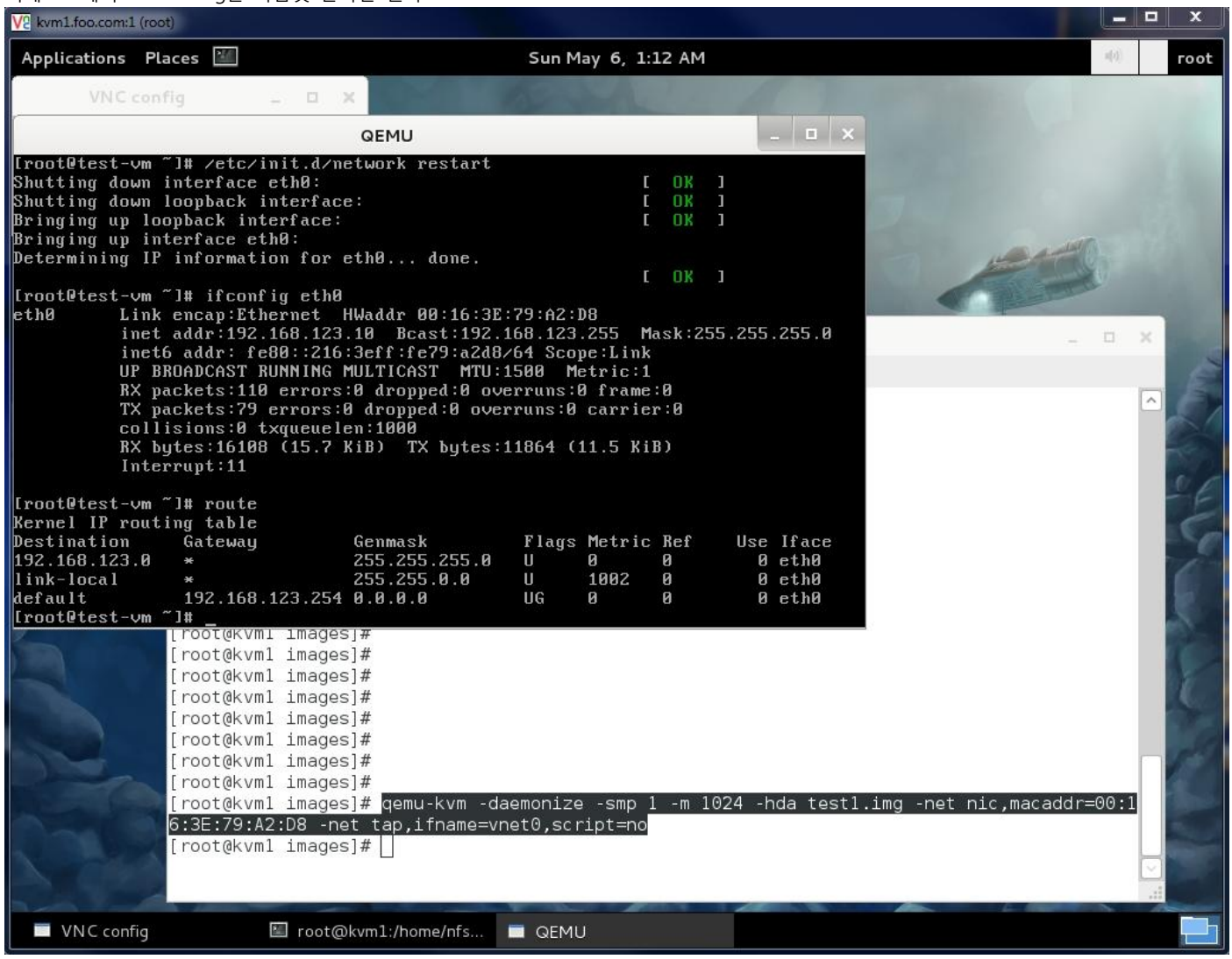
```
# vi /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth1"
HWADDR="00:16:3E:79:A2:D8"
NM_CONTROLLED="no"
```

```
ONBOOT="yes"
BOOTPROTO="dhcp"

# /etc/init.d/network restart

# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:16:3E:79:A2:D8
          inet addr:192.168.123.10  Bcast:192.168.123.255  Mask:255.255.255.0
          inet6 addr: fe80::216:3eff:fe79:a2d8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:90 errors:0 dropped:0 overruns:0 frame:0
          TX packets:68 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11940 (11.6 KiB)  TX bytes:9582 (9.3 KiB)
          Interrupt:11
```

192.168.123.10이라는 IP주소를 확보했고, Route정보를 보면 Gateway(192.168.123.254)인 IP공유기의 IP주소가 정상적으로 할당되어 있다.
이제 VM에서 Networking을 마음껏 즐기면 된다.



✓ NIC 스크립트 (tun/tap 자동설정 스크립트)

앞의 예제에서는, tunctl을 이용해, vnet0라는 장치를 만들고, brctl을 이용해, vnet0를 br0에 추가하는 작업을 수동으로 처리 하였다. 그러나, Qemu에서는 -net 옵션 인자에 script라는 인자를 이용해, 이 과정을 간단히 자동화 할 수 있다.

스크립트는 가상NIC의 생성/삭제에 따른 2가지가 필요하며 아래와 같이 작성 한다. (kvm1, kvm02 두대 모두 작성)

/etc/kvm-ifup (VM 구동시 사용될 Script)

```
# vi /etc/kvm-ifup

#!/bin/bash

if [ -x /sbin/brctl ]; then
    BRCTL="/sbin/brctl"
elif [ -x /usr/sbin/brctl ]; then
  BRCTL="/usr/sbin/brctl"
else
  echo "no bridge utils installed"
  exit 1
fi
if [ -x /sbin/ip ]; then
  switch=$(/sbin/ip route list | awk '/^default / { sub(/.* dev /, ""); print $1 }')
  /sbin/ip link set $1 up
else
  switch=$(/bin/netstat -rn | awk '/^0₩.0₩.0₩.0/ { print $NF }')
  /sbin/ifconfig $1 0.0.0.0 up
fi
${BRCTL} addif ${switch} $1

# chmod 755 /etc/kvm-ifup
```

/etc/kvm-ifdown (VM 구동시 사용될 Script)

```
# vi /etc/kvm-ifdown

#!/bin/bash

if [ -x /sbin/brctl ]; then
  BRCTL="/sbin/brctl"
  elif [ -x /usr/sbin/brctl ]; then
  BRCTL="/usr/sbin/brctl"
else
  echo "no bridge utils installed"
  exit 1
fi
if [ -x /sbin/ip ]; then
  switch=$(/sbin/ip route list | awk '/^default / { sub(/.* dev /, ""); print $1 }')
  ${BRCTL} delif ${switch} $1
  /sbin/ip link set $1 down
else
  switch=$(/bin/netstat -rn | awk '/^0₩.0₩.0₩.0/ { print $NF }')
  ${BRCTL} delif ${switch} $1
  /sbin/ifconfig $1 down
fi

# chmod 755 /etc/kvm-ifdown
```

자, 이제 이 Script사용법을 살펴보자. VM은 앞서 -net 옵션을 수동으로 주었을 때와 동일한 OS이나, NIC를 할당하는 방식을 Script를 이용하는 것으로 대체 한 것이다.

⚠ (앞서 만들었던, 아래와 같은 명령으로 vnet0 장치는 먼저 삭제 한다.)

```
# brctl delif br0 vnet0
# tunctl -d vnet0
```

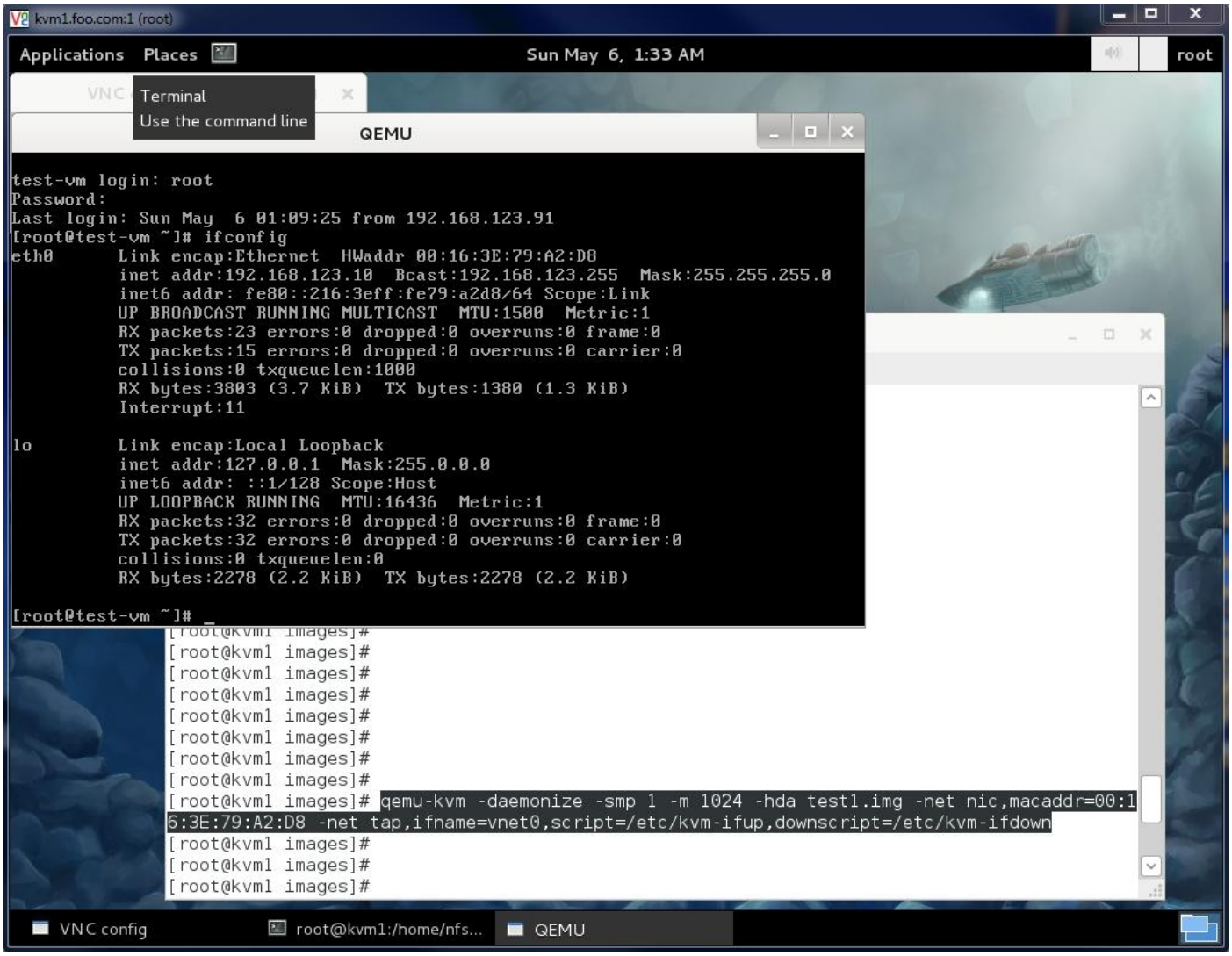
[VM 실행]

```
# qemu-kvm -daemonize -smp 1 -m 1024 -hda /mnt/volume/images/test1.img -net nic,macaddr=00:16:3E:79:A2:D8 -net tap,ifname=vnet0,script=/etc/kvm-ifup,downscript=/etc/kvm-ifdown
```

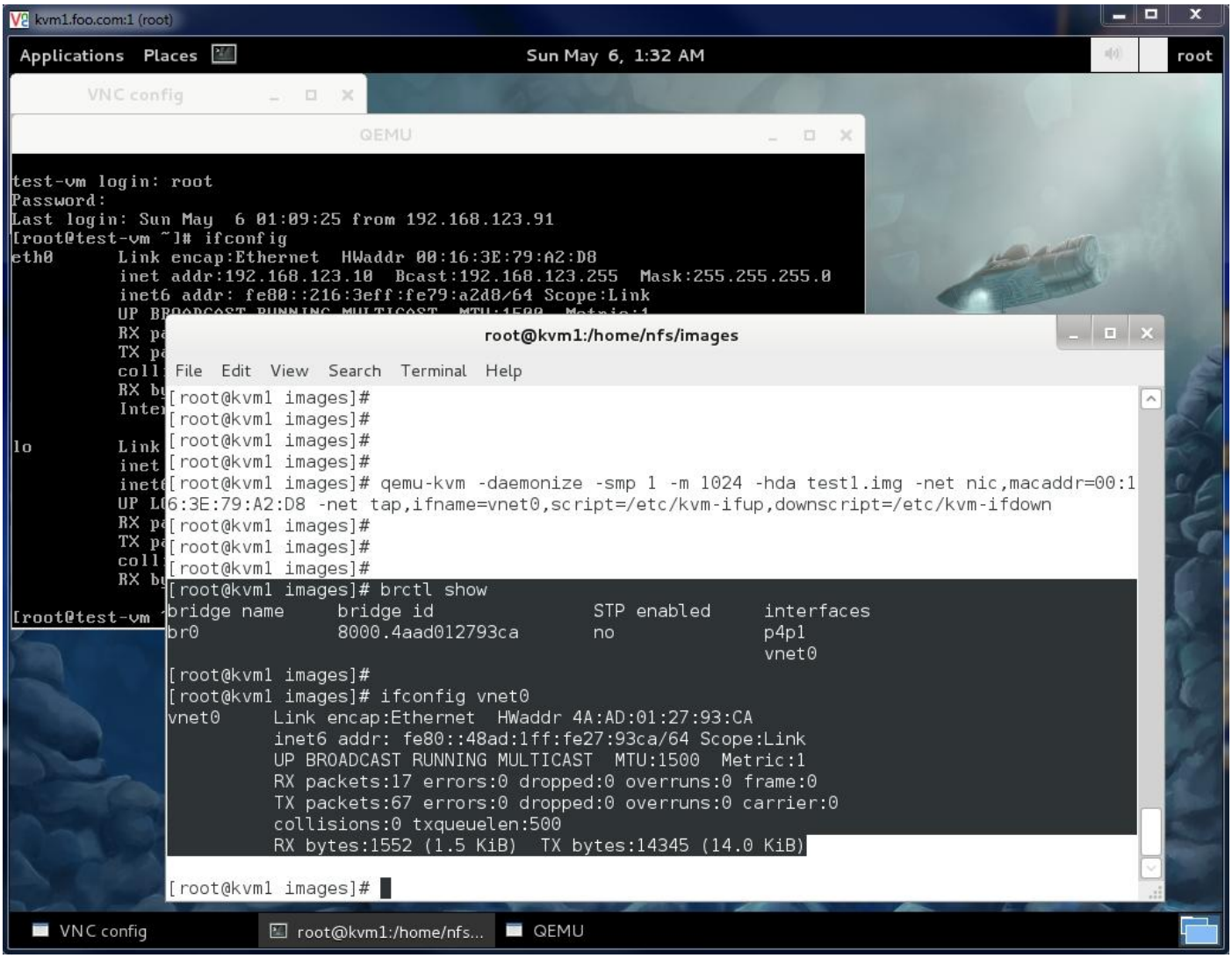
앞서 수동으로 처리 했을 땐는, script=no 였으나, 이제 우리는 tunctl/brctl 작업을 자동으로 해주는 Script를 작성하였으므로, 해당 스크립트를 위와 같이 QEMU에 알려주기만 하면 된다.

(추가 설명)

script=	VM이 Start될 때, 실행. (가상 NIC 생성)
downscript=	VM이 Stop될 때, 실행. (가상 NIC 삭제)

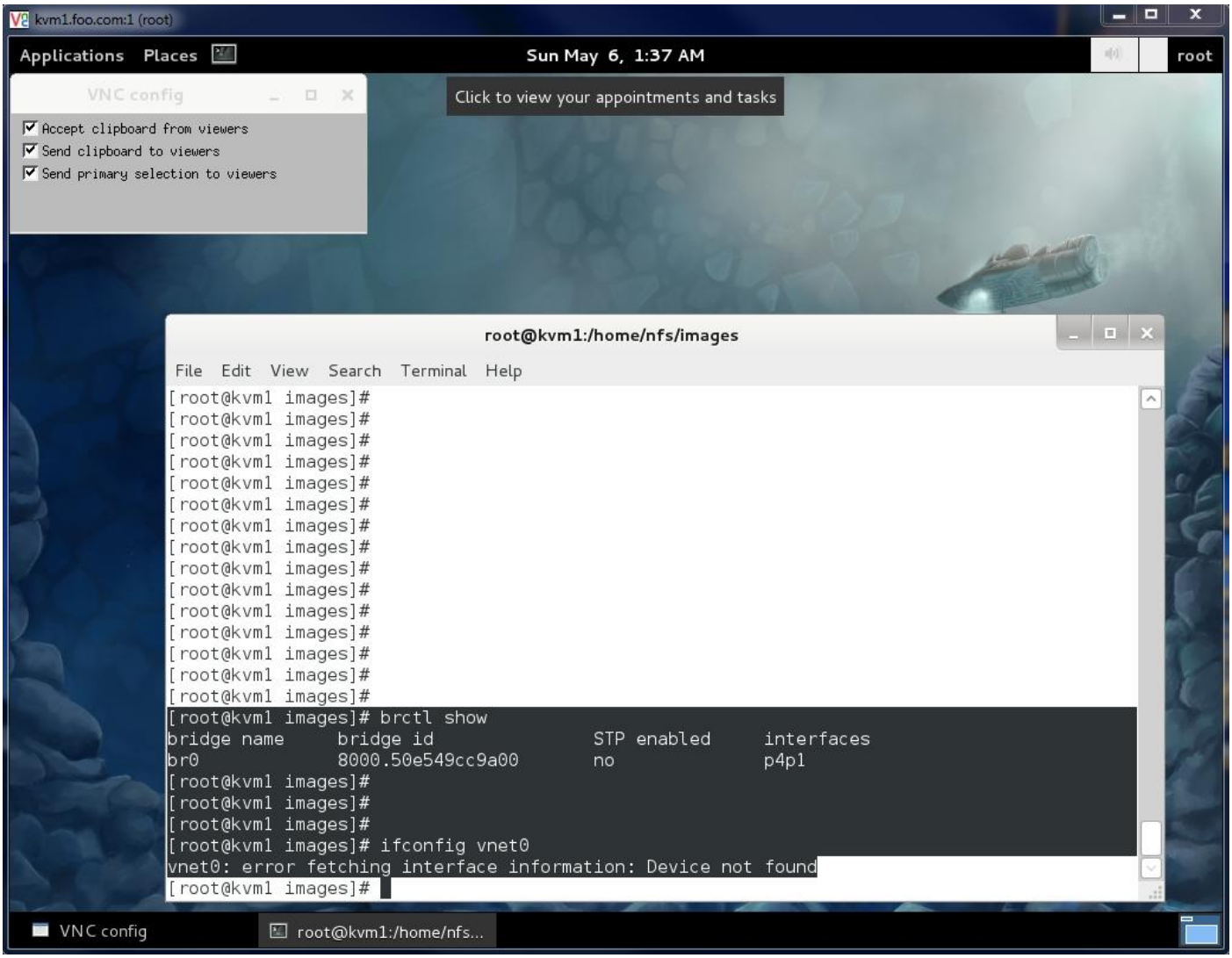


VM에서 NIC정상 인식 확인.



Host에서의 brctl정보와 ifconfig정보에, -net옵션에서 지정한 ifname값이 었던 vnet0라는 장치가 자동으로 생성되어서 Brdige에도 정상적으로 연결된 것을 확인 할 수 있다.

그럼 이제, VM을 종료 하고 다시 Host에서 brctl과 ifconfig정보를 확인해보자.



VM을 Stop(프로세스 종료)한 것만으로, 사용되었던 tun/tap장치(vnet0)의 Bridge에서의 제거 그리고 장치 제거까지 자동으로 처리 된 것을 확인.

이상과 같이 VM에서의 Networking을 사용하는 방법을 알아 보았다. 앞으로 모든 VM에 대한 Networking은 이 과정을 따를 것이다.

● VNC로 VM콘솔화면 접근

지금껏 VM의 로컬콘솔화면은 모두 Host의 VNC화면을 통해 로컬에서만 보고 있다. 이것을 외부의 원격지에서도 볼 수 있다.
간단하며 기본인 VNC를 비롯해, 화면 전송 효율과 품질을 향상시킨 Spice라는 것도 지원한다.
본 문서에서는 간단히 VNC만 다뤄보고, Spice는 별도의 문서로 다루도록 하겠다.

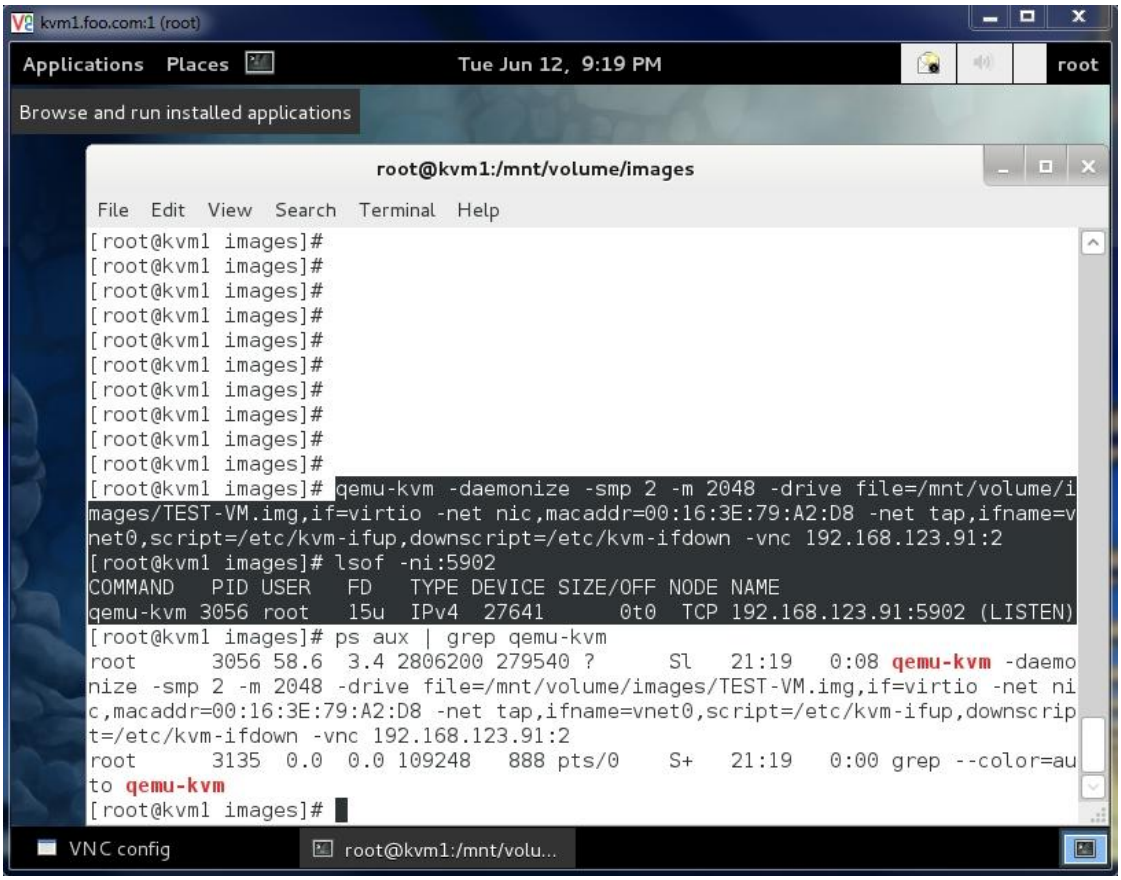
✓ VNC콘솔 활성

VM의 콘솔을 VNC로 리다이렉트(Redirect)하는 방법은 매우 간단하다. 단지 qemu-kvm실행 라인에 아래와 같이 "-vnc :번호"를 사용하면 된다. (모든 인터페이스를 통해 VNC접근을 허용하려면 "-vnc :2"형태로 IP부분을 공란 처리한다.)

```
# qemu-kvm -daemonize -smp 2 -m 2048 -drive file=/mnt/volume/images/TEST-VM.img,if=virtio -net nic,macaddr=00:16:3E:79:A2:D8 -net tap,ifname=vnet0,script=/etc/kvm-ifup,downscript=/etc/kvm-ifdown -vnc 192.168.123.91:2
```

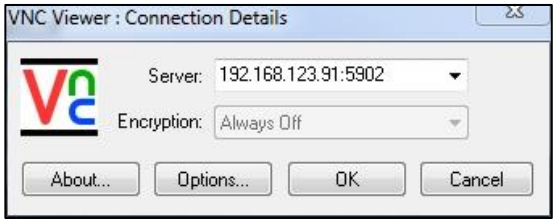
⚠ VNC로 콘솔을 활성화면, 실행 시 자동으로 열리던 로컬콘솔 화면은 더 이상 볼 수 없다.

⚠ 또한, 실제 VNC로 접근 가능한 IP/Port는 VM의 IP가 아니라, VM이 구동되는 Host의 IP/Port이다. 특히, Port는 -vnc 옵션을 통해 지정한 숫자에 5900(VNC기본 포트값)을 더한 값이 된다.
아래 결과를 보면 쉽게 이해가 될 것이다.



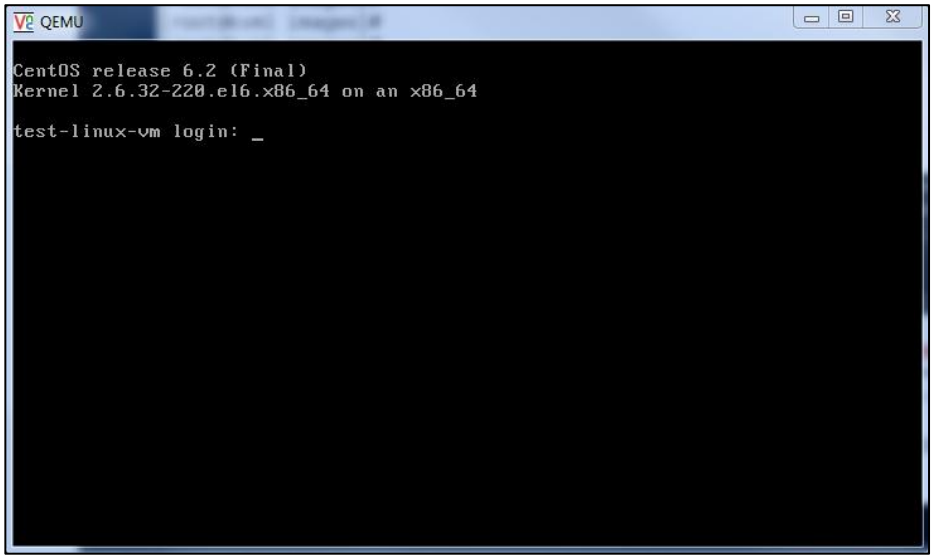
로컬 콘솔은 더 이상 열리지 않으며, Host-IP인 192.168.123.91에 TCP/5902 포트에 VNC 서버가 LISTEN 중임을 확인 할 수 있다.

아래는 외부에서 이 VNC를 통해 VM의 콘솔에 접속하는 과정 및 결과 화면.



원격에서 VM의 콘솔 접속이 가능하다.

⚠ 접근 허가 인증(Auth)은 적용하지 않았다. VNC에 접근암호(Password)설정도 가능하다. http://www.linux-kvm.com/content/securing-your-vnc-headless-guest-simple-passwords 링크 내용을 참조해서 Password도 설정해 보라.



4. Virtio로 I/O 성능 향상 (Disk/Network)

● Virtio (Para-Virtualized drivers) 소개

앞서, KVM은 완전가상화(Full-Virtualization)이라고 했다. 완전가상화에서 VM은 자신이 '가상화'중이라는 것을 인식하지 못하며, 모든 H/W가 에뮬레이트 한다는 것을 의미한다. Computing이나 Memory성능은 크게 문제가 안되나, I/O(입출력) 부분에서는 VM이 '가상화'중임을 인식하지 못하기에, 반가상화(Para-Virtualization)에 비해 상대적으로, 요청처리 단계가 많고 변환과정이 필요하다. 이런 Overhead로 인해, 성능 저하가 불가피하다. 그래서, Disk와, Network I/O에 대해서, Para의 장점을 도입하게 된 것이 Virtio이다.

Virtio는 Para-Virtualized Drivers로서, 완전가상화의 성능 향상을 목적으로 한다. VM의 I/O Latency(지연)을 감소시키고, Host와 비슷한 수준의 성능까지 Throughput(처리량)을 증가 시켜준다. 앞으로는 virtio타입을 기본 사용하는 것으로 한다.

⚠ (주의) Para-Virtualization 기능이므로, Guest OS(VM)에서 Virtio를 인식할 수 있는 OS여야 한다. Linux최신 배포본들은 기본적으로 지원하며, Windows계열은 별도의 HDD Driver 형태로 지원이 된다.

혹, Virtio에 대해 보다 상세한 자료가 필요하다면, 아래 링크 문서를 추천한다.

Virtio: Linux를 위한 I/O 가상화 프레임워크 : http://www.ibm.com/developerworks/kr/library/l-virtio/index.html

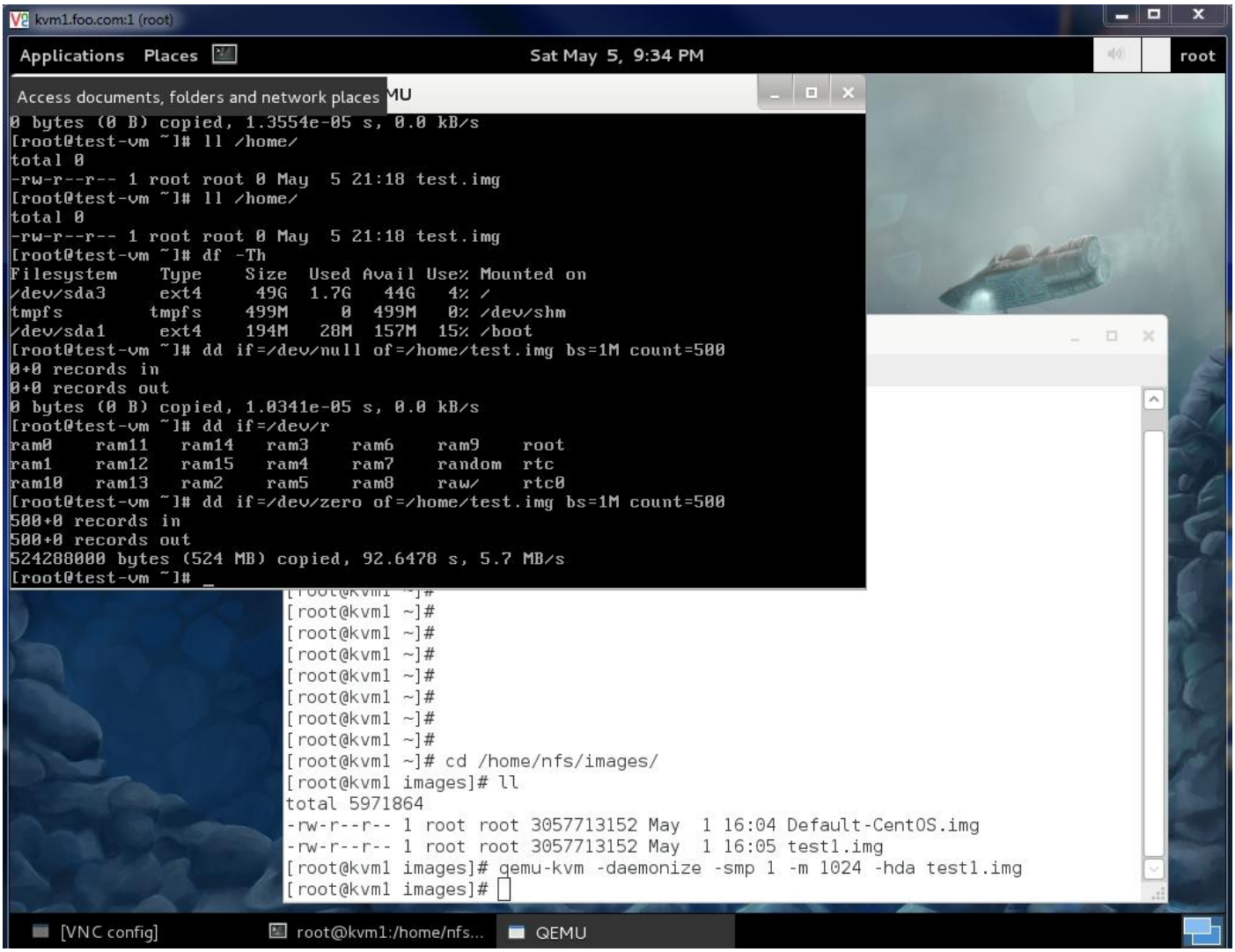
● Disk : Non-Virtio vs Virtio

앞서 VM 간단 실습에서 해봤던 VDI타입(-had)이 Non-Virtio로서, VM에서 인식 타입과 성능을 간단히 살펴보자.

먼저 fdisk -l 실행결과를 보면, VM에서 인식한 HDD가 /dev/sda로서, IDE타입으로 잡혀 있다.

✓ **TEST - Non-Virtio Disk**

그리고, 간단한 방법으로, dd명령을 통해, 500MB짜리 파일 생성테스트를 해보니 5.7MB/s라는 수치가 나왔다. 너무나 느린 값이다.



✓ TEST - Virtio Disk

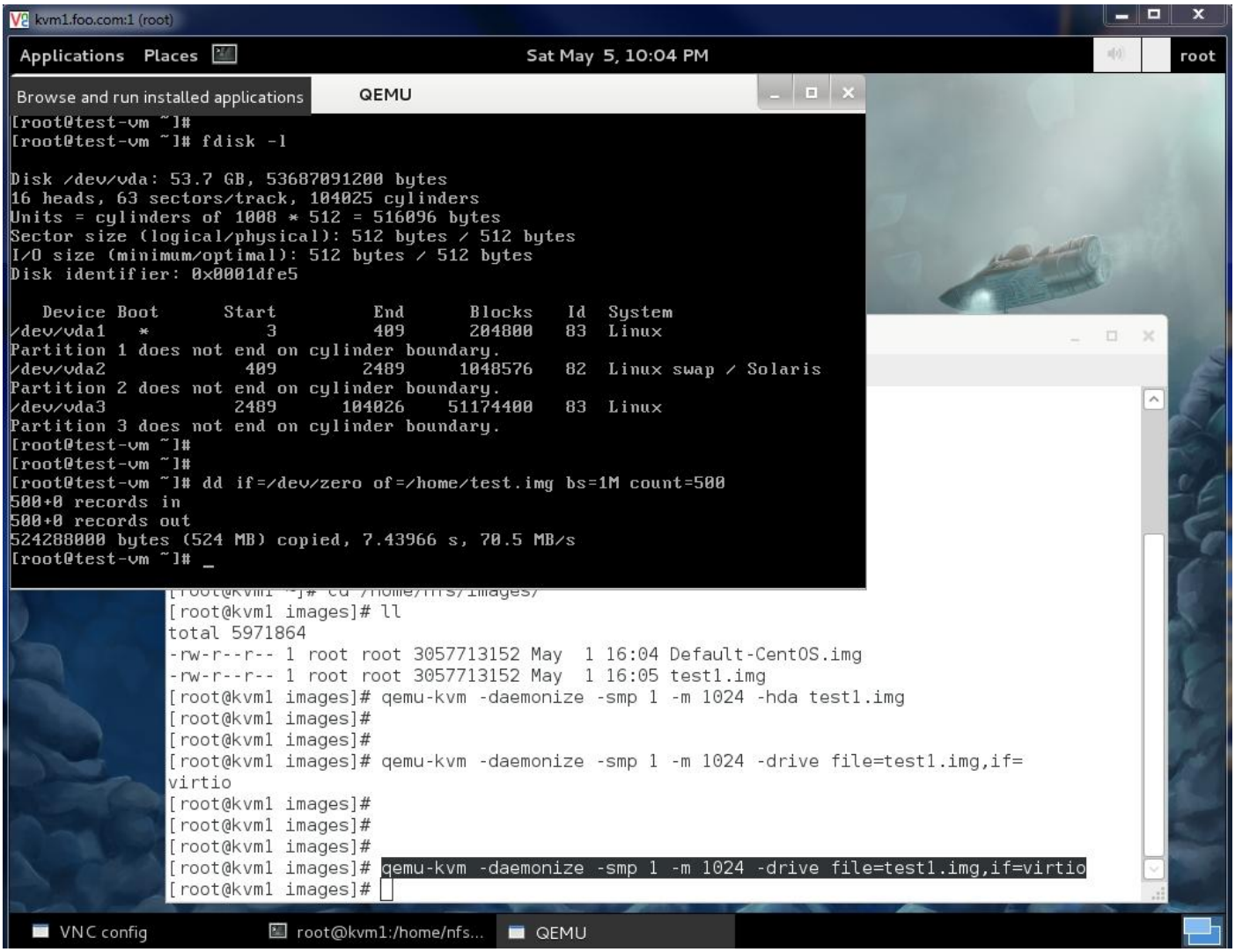
이번에는 동일 VDI를 가지고 Virtio타입으로 VM을 구동시켜서 동일한 테스트를 해보자.

아, 실행 옵션이 하기와 같이 달라진다.

```
# qemu-kvm -daemonize -smp 1 -m 1024 -drive file=/mnt/volume/images/test1.img,if=virtio
```

자, 어떤가? 성능부터 보자, 70.5MB/s가 나왔다. Non-Virtio일 때는 5.7MB/s라는 아주 저조한 수치에 비해, 비약적인 성능 향상을 확인 할 수 있다.

또한, 중요한 부분이, VDI인식 장치명이 /dev/vda이다. 즉, QEMU/KVM에서 Virtio타입으로 장착된 VDI는 vda, vdb, vdc...등등과 같이 "v"로 시작하는 장치명을 같게 된다는 것을 기억해 두자.



● NIC : Non-Virtio vs Virto

이번에는 NIC장치에 대해 Virtio를 적용해보자.

None-Virtio의 경우는, VM Networking섹션에서 이미 다루어 보았으며. 그 때 가상 NIC와 관련된 옵션이 아래와 같았다.

```
-net nic,macaddr=00:16:3E:79:A2:D8 -net tap,ifname=vnet0,script=/etc/kvm-ifup,downscript=/etc/kvm-ifdown
```

Virtio타입으로 가상 NIC를 할당 하기 위해서는 아래와 같이 옵션을 변경해 준다.

```
-netdev type=tap,ifname=vnet0,script=/etc/kvm-ifup,downscript=/etc/kvm-ifdown,id=net0 -device virtio-net-pci,mac=00:16:3E:79:A2:D8,netdev=net0
```

-net대신 -netdev옵션으로 NIC정의 해주고, -device옵션을 이용해 정의한 NIC의 속성정보를 부여한다. 이때, virtio-net-pci타입으로 정의 해주면 되는 것이다.

✓ TEST - Non-Virtio NIC

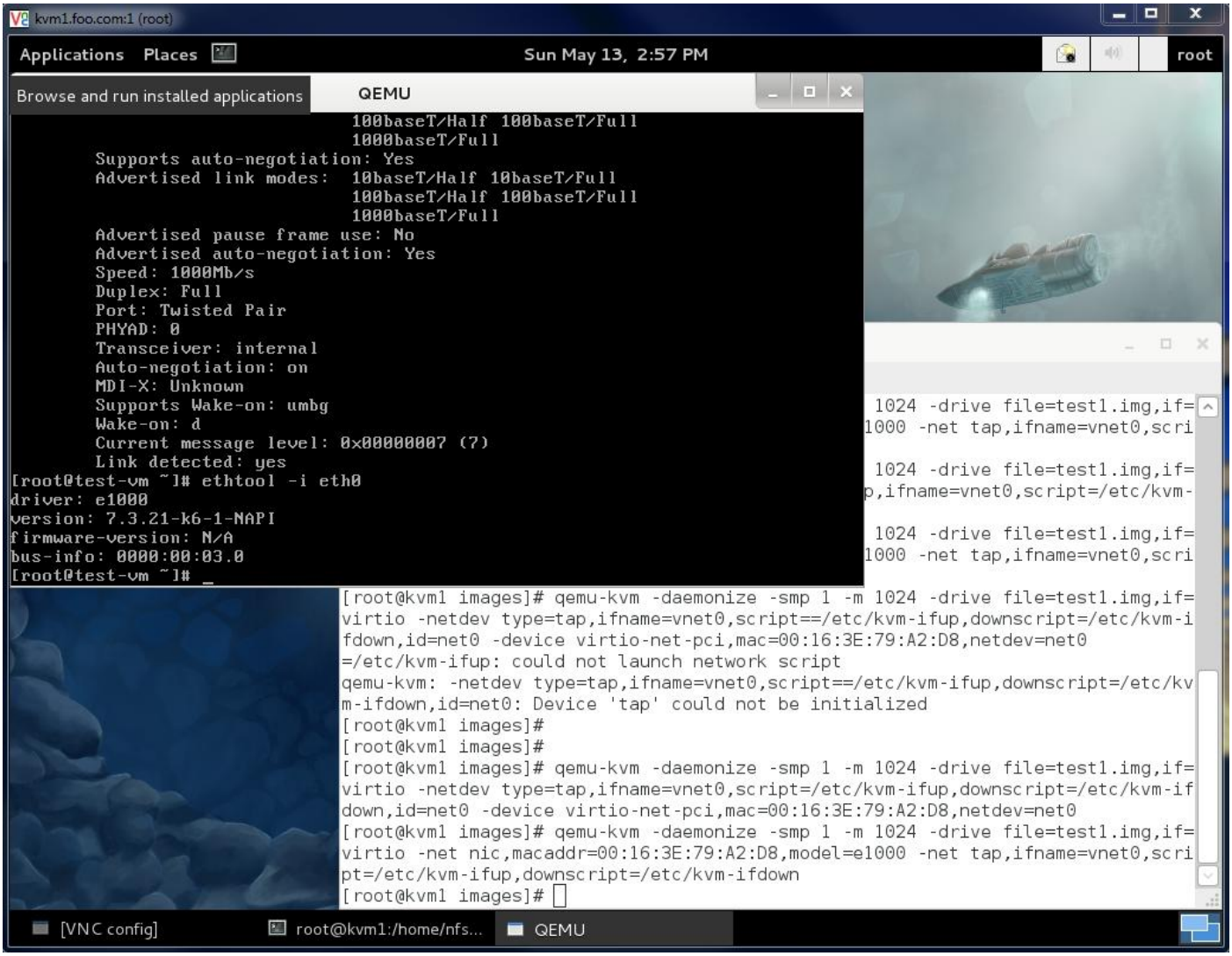
자, 그럼 None-Virtio와, Virtio의 경우에서, 어떻게 차이가 나는지 실제 VM을 구동시켜 확인 해보자.

먼저, Non-Virtio로.....(Disk는 Virtio로 한다. 괜히 살펴 본게 아니니~)

⚠ model=e1000이라는 옵션은 여기서 처음 나온 것이다. model옵션을 사용하지 않게 되면, Default로 RTL8139 NIC로 에뮬레이팅 된다. RTL8139는 100Mbits/sec타입이다. Non-Virtio타입으로 사용하더라도 e1000(Intel NIC로서, 1000Mbits/sec)로 사용할 것을 권장.

```
# qemu-kvm -daemonize -smp 1 -m 1024 -drive file=/mnt/volume/images/test1.img,if=virtio -net nic,macaddr=00:16:3E:79:A2:D8,model=e1000 -net tap,ifname=vnet0,script=/etc/kvm-ifup,downscript=/etc/kvm-ifdown
```

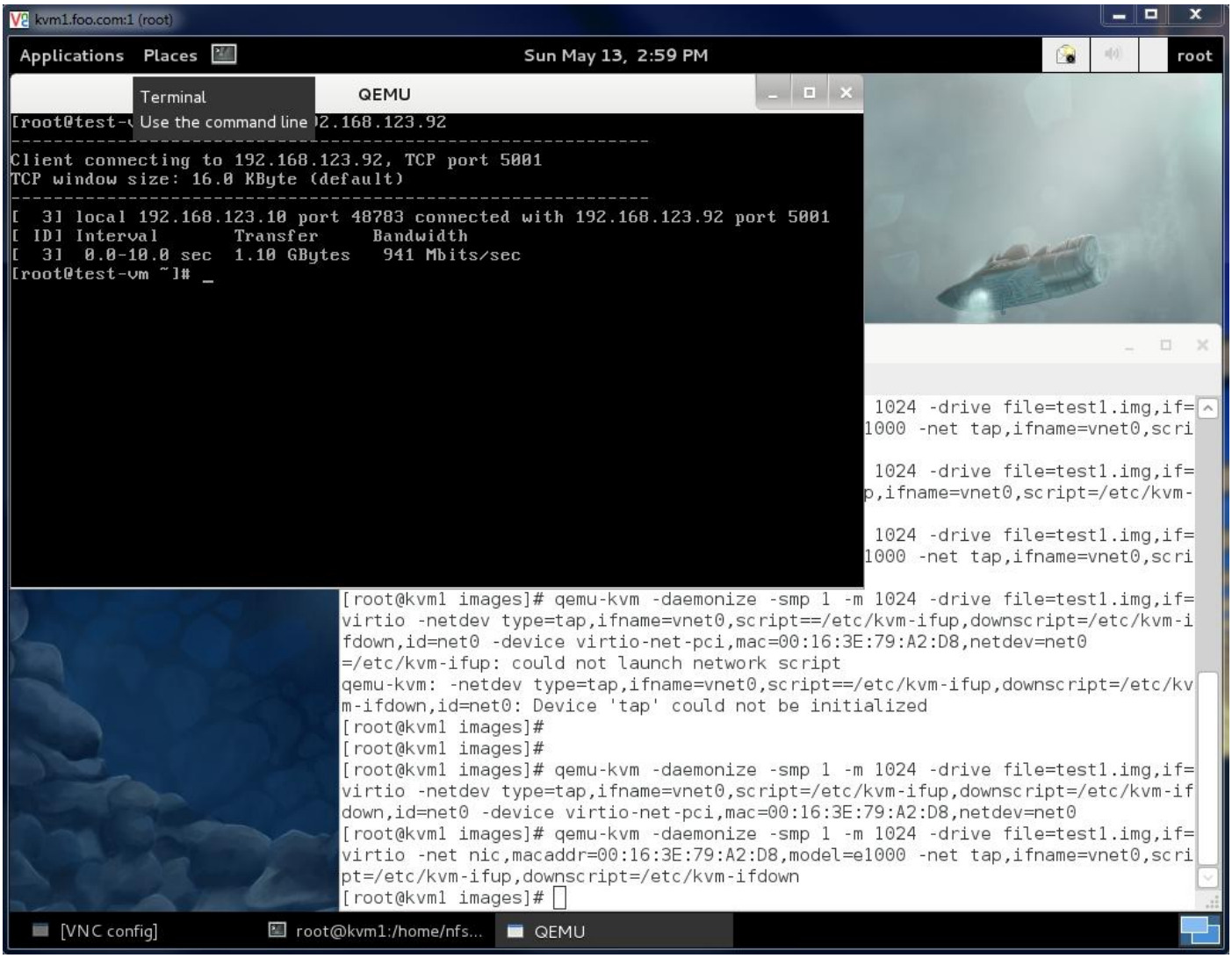
자~ 아래 스샷이 None-Virtio에서 e1000 타입으로 장착된 Ethernet 장치에 대한 세부 정보를 보여주고 있다.
Speed: 1000Mb/s이고, Full Duplex를 정상적으로 지원하고 있다. 그리고 NIC 드라이버는 지정한대로 e1000이 담당.



⚠현재 실습하고 있는 VM인, CentOS에서는 iperf 패키지가 없을 것이다. 아래와 같이 설치 해준다.

```
# rpm -Uvh http://download.fedoraproject.org/pub/epel/6/i386/epel-release-6-6.noarch.rpm
# yum -y install iperf
```

이제, NIC 성능 테스트를 해보자. 도구로는 iperf를 사용할 것이다. iperf 서버는 kvm2.foo.com에 구동(iperf -s 실행)시켜 두고, VM에서 iperf 클라이언트를 실행시켜 어느 정도 전송 성능이 나오는지 테스트한 결과를 보자.



전송 대역폭이 941Mbits/sec가 나왔다. e1000의 보장 대역폭인 1Gbits/sec(=1000Mbits/sec)에 근접한 수치가 나왔다. 일단 Non-Virtio 테스트는 완료

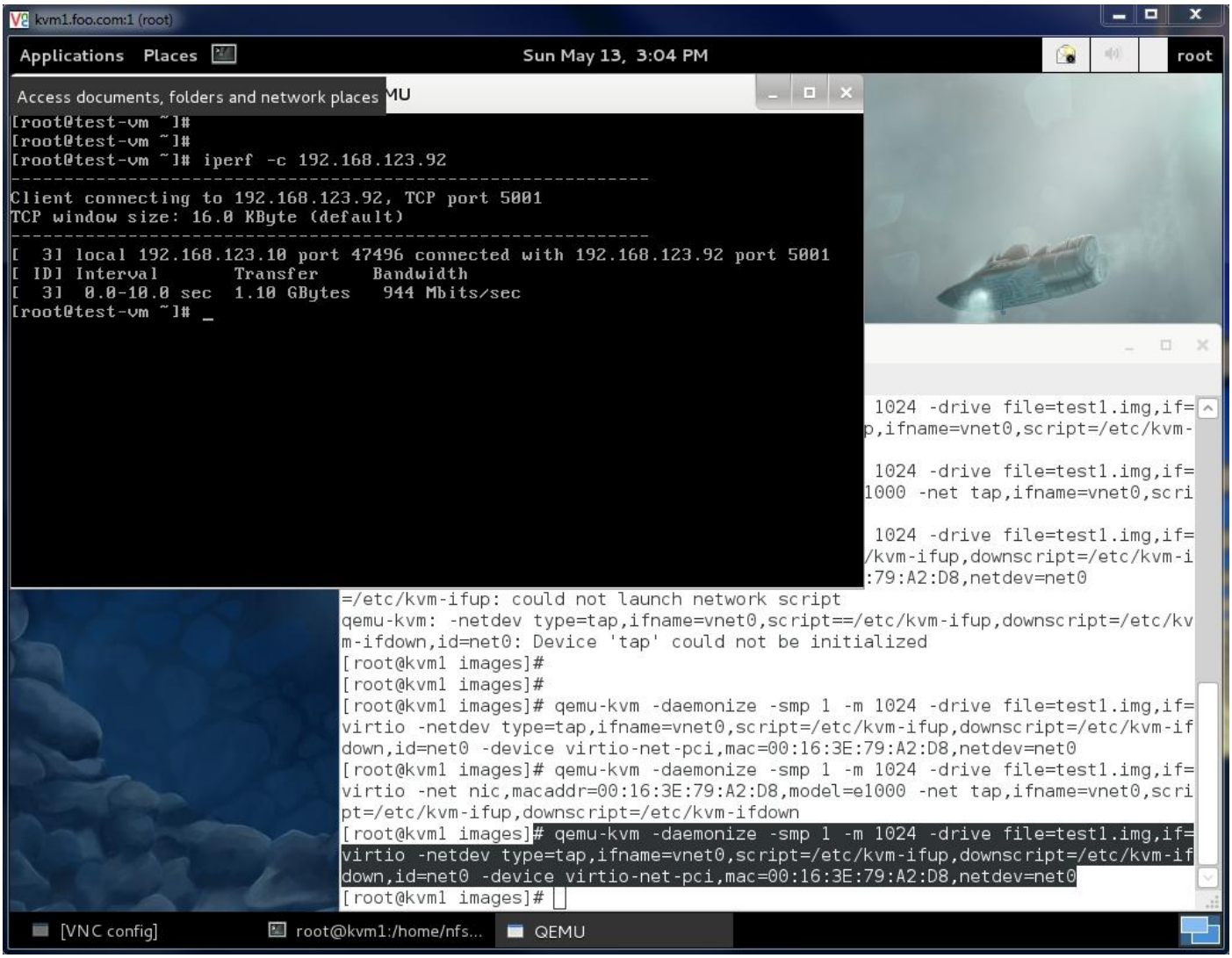
✓ TEST - Virtio NIC

다음으로 Virtio모드에서 테스트를 해보자.

VM 구동 옵션은 아래와 같이 한다.

```
# qemu-kvm -daemonize -smp 1 -m 1024 -drive file=/mnt/volume/images/test1.img,if=virtio -netdev type=tap,ifname=vnet0,script=/etc/kvm-ifup,downscript=/etc/kvm-ifdown,id=net0 -device virtio-net-pci,mac=00:16:3E:79:A2:D8,netdev=net0
```

결과 스샷이다.



Virtio모드에서의 NIC 전송 대역폭 결과로 944Mbits/sec가 나왔다. 성능 향상이 없자나!! 뭐지!!!????? ~라고 할 것이다. 941이나 944나 약간의 편차만 존재하는 값이지 거의 대등한 성능 값이 맞다.

이유는, 단일 VM하나로 테스트를 진행 했기 때문이다. 다수의 VM들에서 NIC패킷 부하가 동시에, 그리고 다량 발생할 경우, Non-Virtio에서는 극심한 성능 저하가 발생하나, Virtiio에서는 Non-Virito대비 월등한 성능을 보여줄 것이다.

⚠ 이상에서 살펴본 바와 같이, Hardware가 지원되는 환경이라면 가능한 Disk나 NIC에서는 Virtio(Para Driver)타입을 사용할 것을 권장한다. 그리고 본 문서에서 앞으로의 모든 테스트에서도 특별한 언급이 없으면, Virtio모드를 기본으로 할 것이다.

● Windows VM 설치 및 Virtio적용 실습

이왕 시작한 거~ Virtio타입의 Windows도 간단히 테스트 해보고 Template도 만들어 두자. 나중에 혹시 쓰일 일이 있을지도...

순서는 대략 아래와 같다.

1. NIC없이 HDD만(Non-Virtio)으로 심플하게 VM을 생성해서 Windows XP 설치
2. Virtio-Win 드라이버 ISO를 CD-ROM형태로 붙여 VM 부팅 및 Disk와 NIC 드라이버 설치
3. VM Shutdown 및 Virtio타입으로 HDD변경, 그리고 NIC 역시 Virtio모드로 추가하여 부팅
4. Windows에서 Disk와, NIC가 Virtio타입으로 연결되었는지 확인.

그리고 Windows Virtio 설치와 관련된 추천 문서는 아래와 같으니, 시간 있을 때 참조하면 도움이 될 것이다.

http://www.linux-kvm.com/content/redhat-54-windows-virtio-drivers-part-2-block-drivers
http://www.linux-kvm.com/content/block-driver-updates-install-drivers-during-windows-installation
http://www.linux-kvm.com/content/tip-how-setup-windows-guest-paravirtual-network-drivers

자, 그럼 먼저 Windows XP를 노멀하게 설치 해보자. 누구나 집에서 많이들 해본 과정이라 시작과 끝 부분 정도만 설명 한다.

✓ Windows VM에 사용될 이미지 파일 생성

앞서, 공부 했듯이 qemu-img를 통해 VM이 사용할 VDI를 생성해 주어야 한다.

```
# cd /mnt/volume/images

# qemu-img create -f qcow2 Default-Windows-XP.img 50G
Formatting 'Default-Windows-XP.img', fmt=qcow2 size=53687091200 encryption=off cluster_size=65536

# ls -al
drwxr-xr-x 2 root root          4096 Jun   9 22:55 .
drwxr-xr-x 4 root root          4096 Apr 30 22:17 ..
-r--r--r-- 1 root root 3057713152 May 13 22:03 Default-CentOS.img
-rw-r--r-- 1 root root       197632 Jun   9 22:55 Default-Windows-XP.img
-rw-r--r-- 1 root root 3662610944 May 13 22:58 test1.img
```

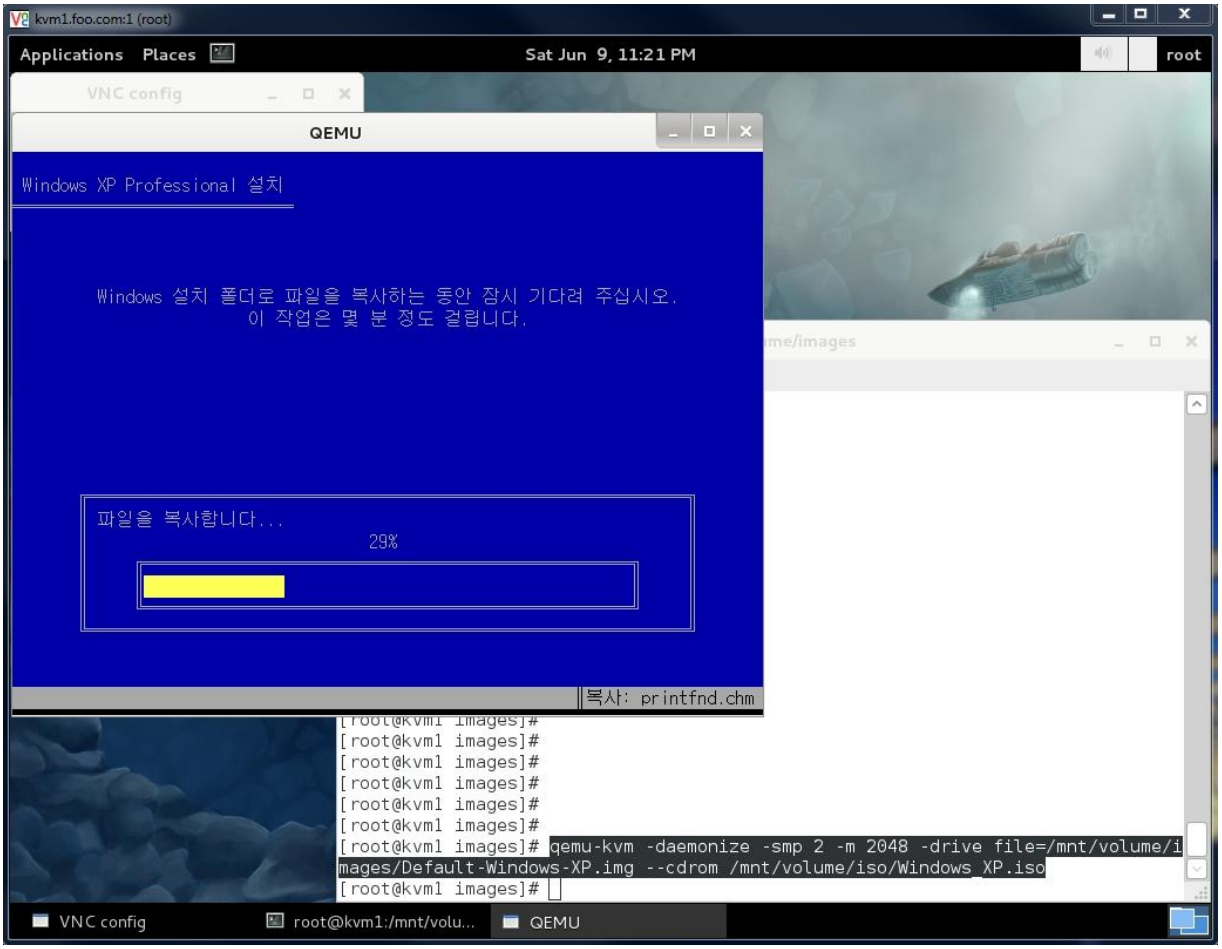
✓ Windows XP 초기 설치 진행

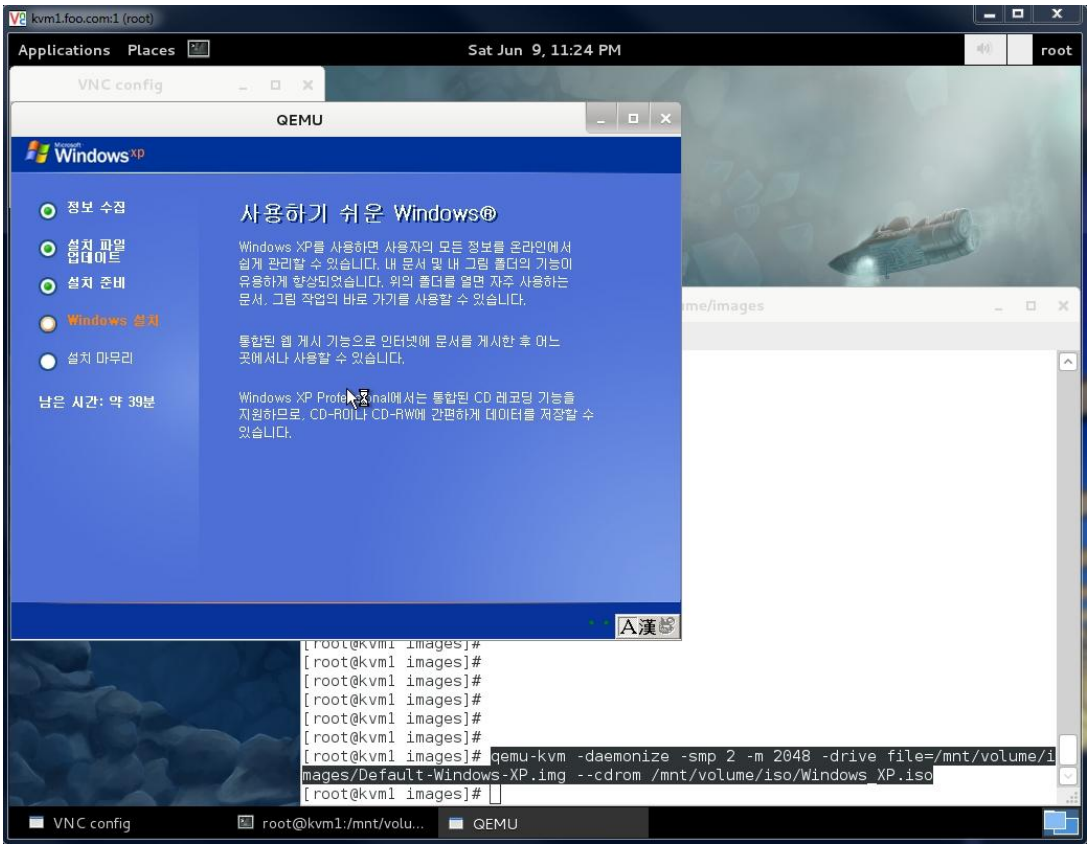
Windows이기도 하고, 변화를 좀 줘 봤다. CPU는 2개(-smp 2), 메모리는 2GB(-m 2048)로 하였다.

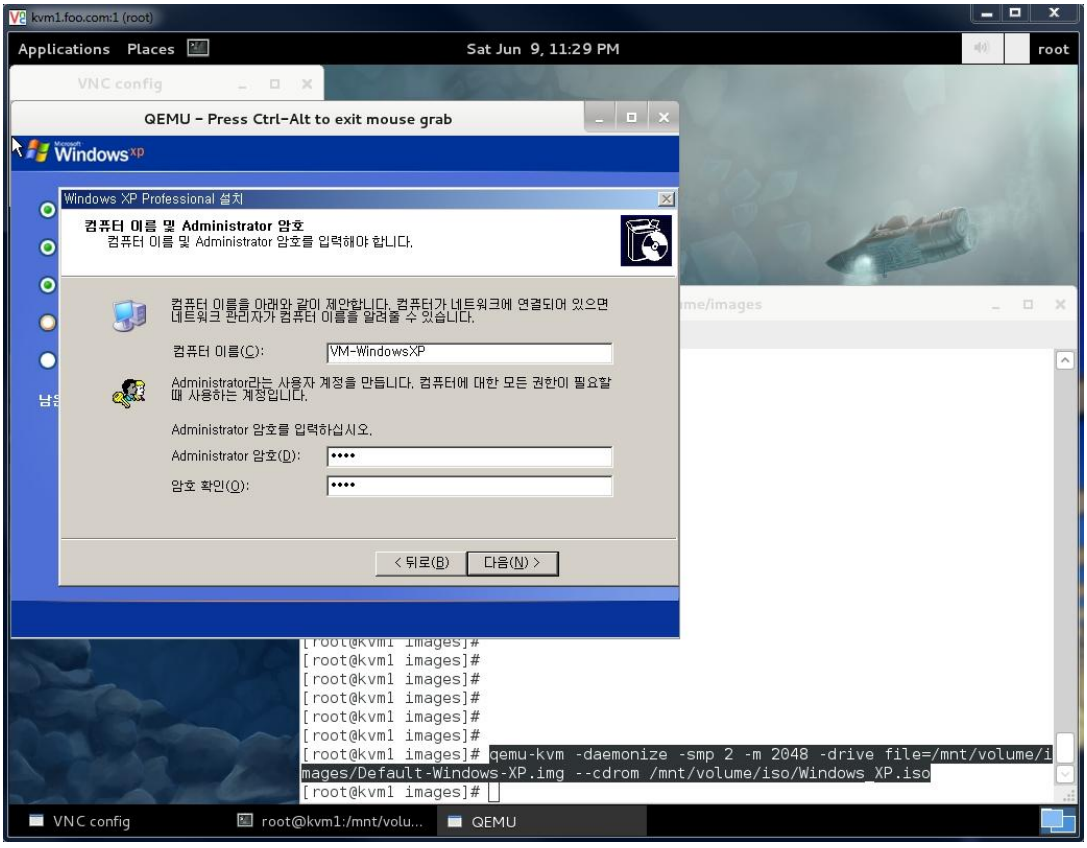
```
# qemu-kvm -daemonize -smp 2 -m 2048 -drive file=/mnt/volume/images/Default-Windows-XP.img --cdrom /mnt/volume/images/Windows_XP.iso
```

⚠ Windows XP 설치 CD이미지 파일(ISO)는 미리 /mnt/volume/iso/Windows_XP.iso라는 파일로 준비해 두었다.

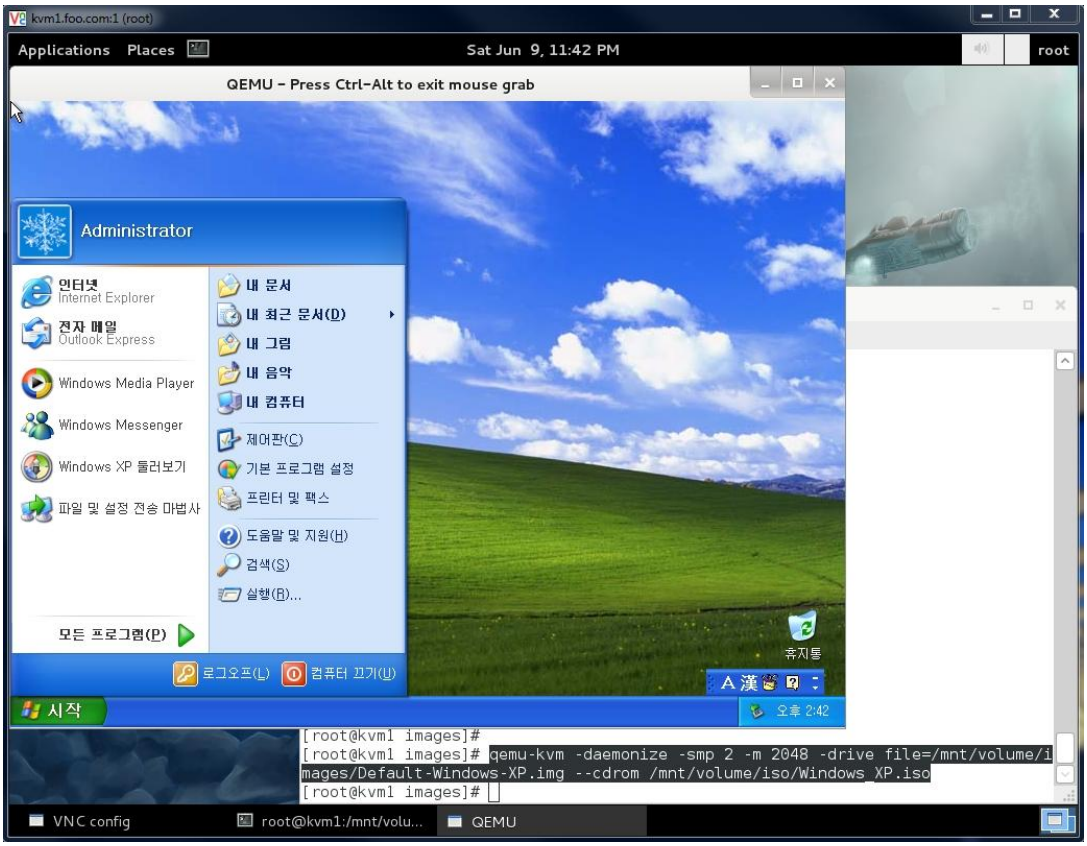
설치 시작....




kvm1.foo.com:1 (root)
Applications Places
Sat Jun 9, 11:24 PM
root
VNC config
QEMU
Windows XP
정보 수집
설치 파일 업데이트
설치 준비
Windows 설치
설치 마무리
남은 시간: 약 39분
사용하기 쉬운 Windows®
Windows XP를 사용하면 사용자의 모든 정보를 온라인에서 쉽게 관리할 수 있습니다. 내 문서 및 내 그림 폴더의 기능이 유용하게 향상되었습니다. 위의 폴더를 열면 자주 사용하는 문서, 그림 작업의 바로 가기를 사용할 수 있습니다.
통합된 웹 게시 기능으로 인터넷에 문서를 게시한 후 어느 곳에서나 사용할 수 있습니다.
Windows XP Professional에서는 통합된 CD 레코딩 기능을 지원하므로, CD-R이나 CD-RW에 간편하게 데이터를 저장할 수 있습니다.
[root@kvm1 images]#
[root@kvm1 images]# qemu-kvm -daemonize -smp 2 -m 2048 -drive file=/mnt/volume/images/Default-Windows-XP.img --cdrom /mnt/volume/iso/Windows_XP.iso
[root@kvm1 images]#
VNC config
root@kvm1:/mnt/volu...
QEMU


kvm1.foo.com:1 (root)
Applications Places
Sat Jun 9, 11:29 PM
root
VNC config
QEMU - Press Ctrl-Alt to exit mouse grab
Windows XP
Windows XP Professional 설치
컴퓨터 이름 및 Administrator 암호
컴퓨터 이름 및 Administrator 암호를 입력해야 합니다.
컴퓨터 이름을 아래와 같이 제안합니다. 컴퓨터가 네트워크에 연결되어 있으면 네트워크 관리자가 컴퓨터 이름을 알려줄 수 있습니다.
컴퓨터 이름(C): VM-WindowsXP
Administrator라는 사용자 계정을 만듭니다. 컴퓨터에 대한 모든 권한이 필요할 때 사용하는 계정입니다.
Administrator 암호를 입력하십시오.
Administrator 암호(D):
암호 확인(O):
< 뒤로(B) 다음(N) >
[root@kvm1 images]#
[root@kvm1 images]# qemu-kvm -daemonize -smp 2 -m 2048 -drive file=/mnt/volume/images/Default-Windows-XP.img --cdrom /mnt/volume/iso/Windows_XP.iso
[root@kvm1 images]#
VNC config
root@kvm1:/mnt/volu...
QEMU

초기 설치 완료...



✓ Virtio 드라이브 설치 (Disk)

설치는 완료 했으니, Virtio 드라이버들을 설치할 차례다.

VM을 종료하고, Windows 설치CD를 분리한다. 그리고 Virtio Windows 드라이버 ISO를 다운로드.

다운로드 전에 아래 두곳중에서 에서 최신 버전을 먼저 확인 한다.

(XP의 경우, 현재 시점에서는 잘 지원되지 않는 듯.. XP에서 정상 동작하는 virtio 드라이버 찾기가 좀 난해함...)

http://www.linux-kvm.com/content/redhat-54-windows-virtio-drivers-part-2-block-drivers

http://alt.fedoraproject.org/pub/alt/virtio-win/latest/images/bin/

http://www.windowsservercatalog.com/results.aspx?text=Red+Hat&bCatID=1282&avc=10&ava=0&OR=5&=Go&chtext=&cstext=&csttext=&chbtext=

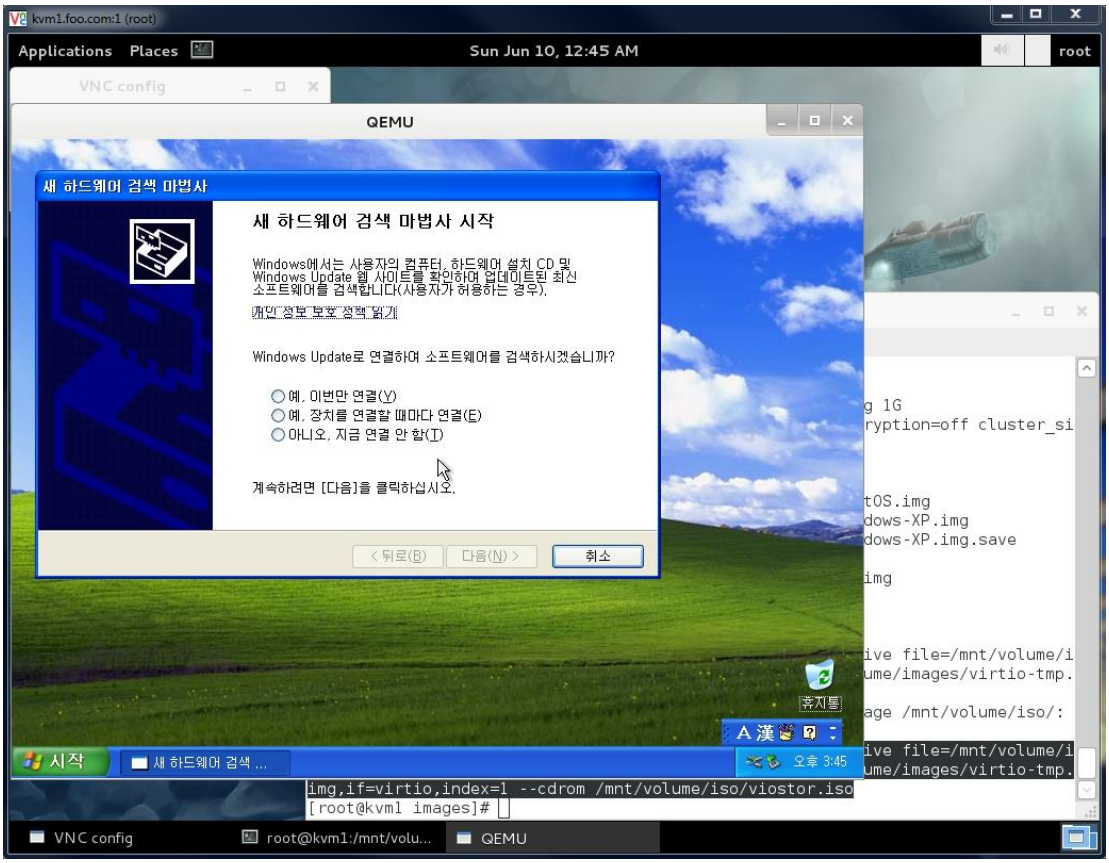
```
# cd /mnt/volume/iso
# http://alt.fedoraproject.org/pub/alt/virtio-win/latest/images/bin/virtio-win-0.1-22.iso
```

이제, Windows VM에 조금전 다운로드 받은 Virtio 드라이버 CD를 VM에 연결하여 VM을 부팅한다. 단, virtio 모드로 Temp Disk를 하나 만들어 임시로 연결한다. 이유는 Virtio 드라이버를 Window에 인식시키기 위함이다. 잠시 사용하고 삭제할 것이므로 적당히 1만으로 생성한다. 그리고 Disk가 2개 이상이 되었으므로, 장착 순서를 index라는 옵션을 이용해 지정 해준다.

VM 구동 명령은 아래와 같을 것이다.

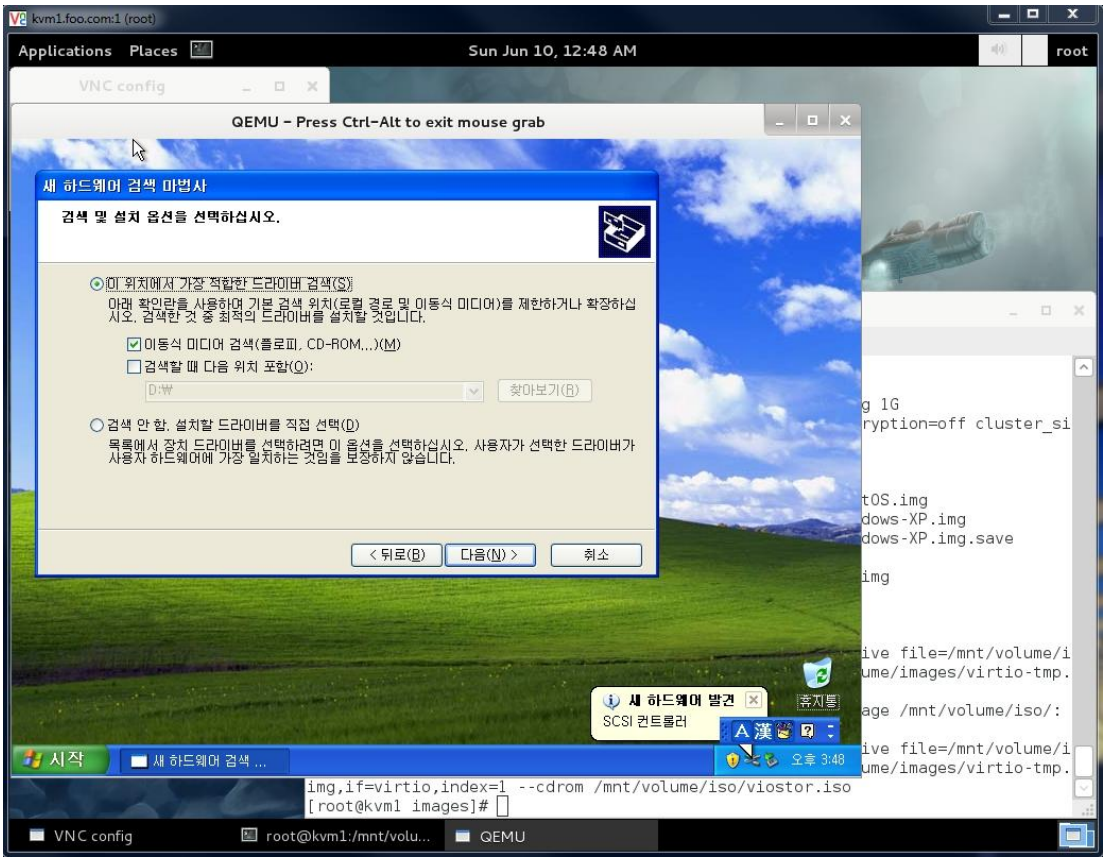
```
# cd /mnt/volume/images
# qemu-img create -f qcow2 virtio-tmp.img 1G
# qemu-kvm -daemonize -smp 2 -m 2048 -drive file=/mnt/volume/images/Default-Windows-XP.img,index=0 --drive file=/mnt/volume/images/virtio-tmp.img,if=virtio,index=1 --cdrom /mnt/volume/iso/ virtio-win-0.1-22.iso
```

부팅이 완료 되면, 아래와 같이 virtio타입의 디스크 장치를 인식하고 드라이버를 요구한다.

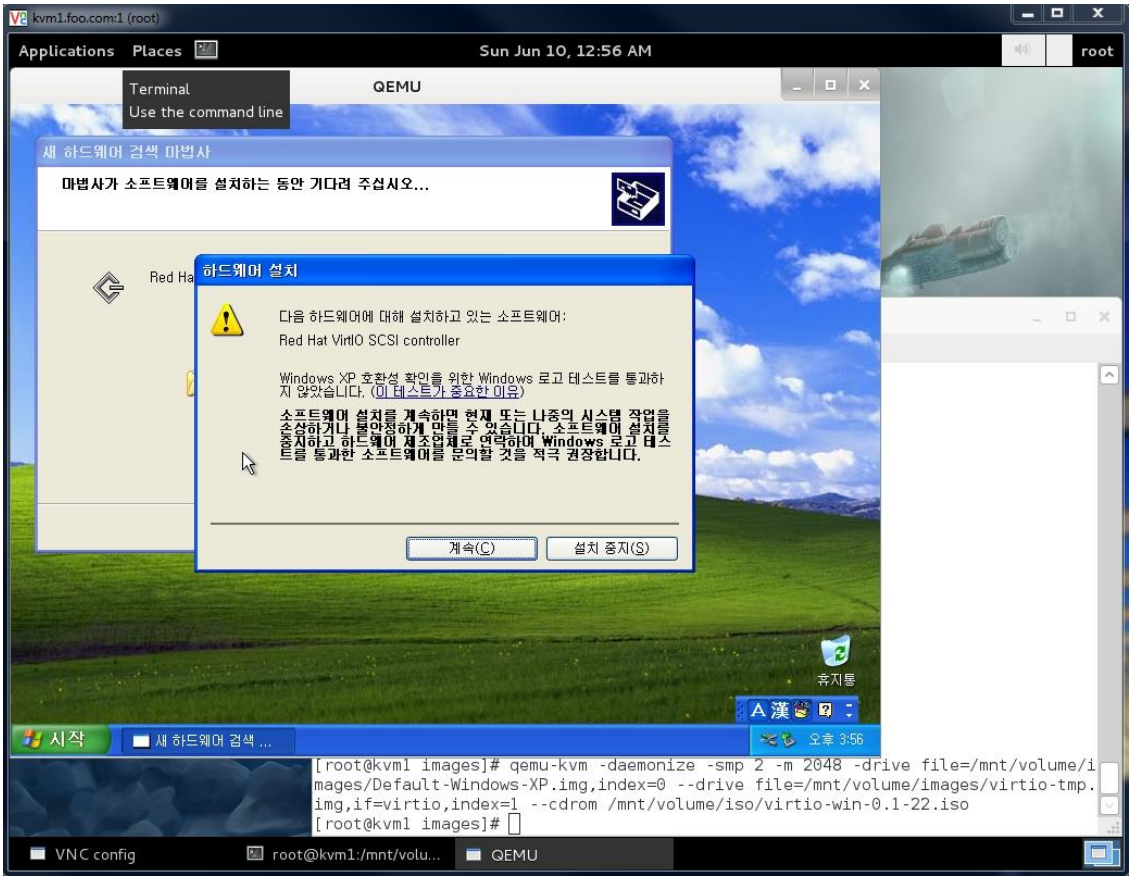


viostor.iso 이미지를 CD-ROM에 장착 해두었으므로, "목록 또는 특정 위치에서 설치(고급)"을 통해, CD-ROM에서 드라이버를 설치 해준다.

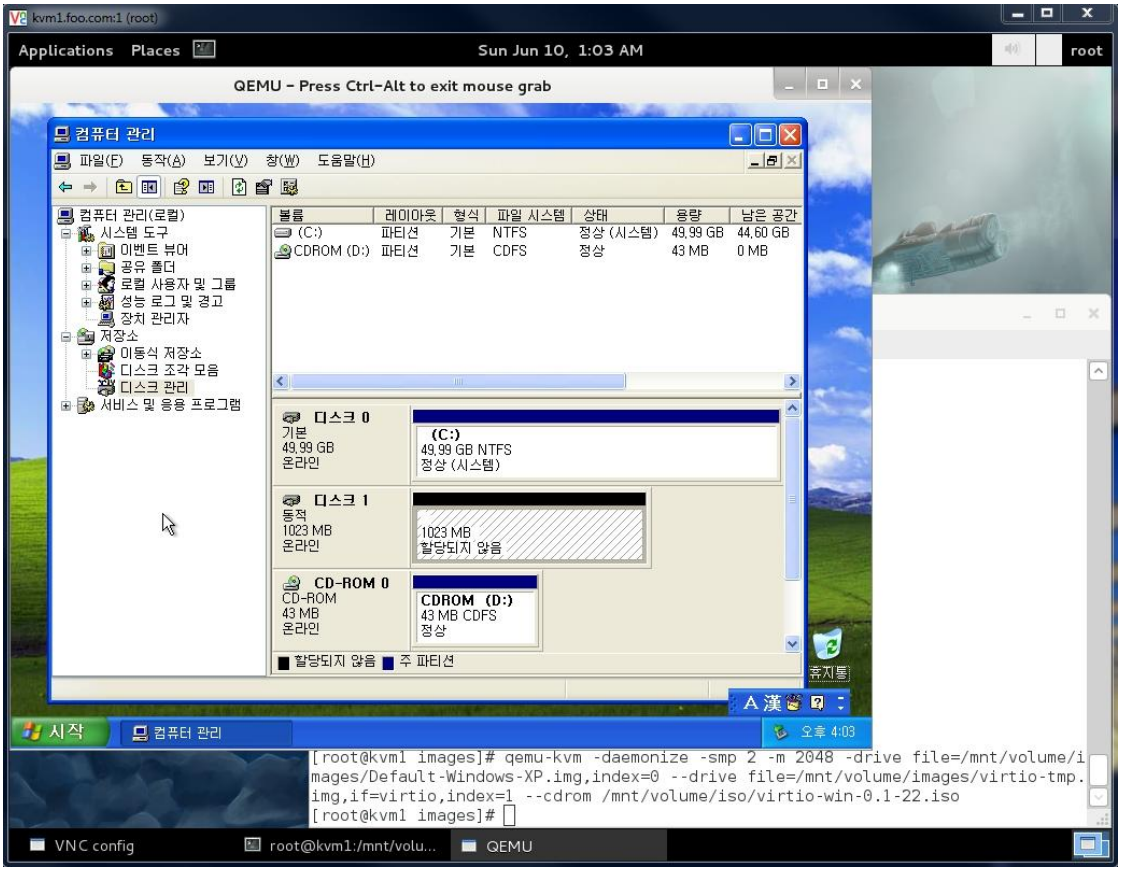
자동 검색으로 설치 했을 때 정상 설치가 되지 않을 경우, "내 컴퓨터-장치관리자"에서 느낌표가 붙은 Red Hat드라이버를 수동 업데이트를 진행한다. 이후, "설치할 드라이버를 직접 선택"을 하고, "디스크 있음"을 이용해 "CD-ROM₩WXp₩x86₩viostor"을 선택 한다. XP의 경우 현재에는 구형 OS라 지원이 미미한 듯 하다. Windows7의 경우에는 자동 검색으로 바로 설치 가능.)



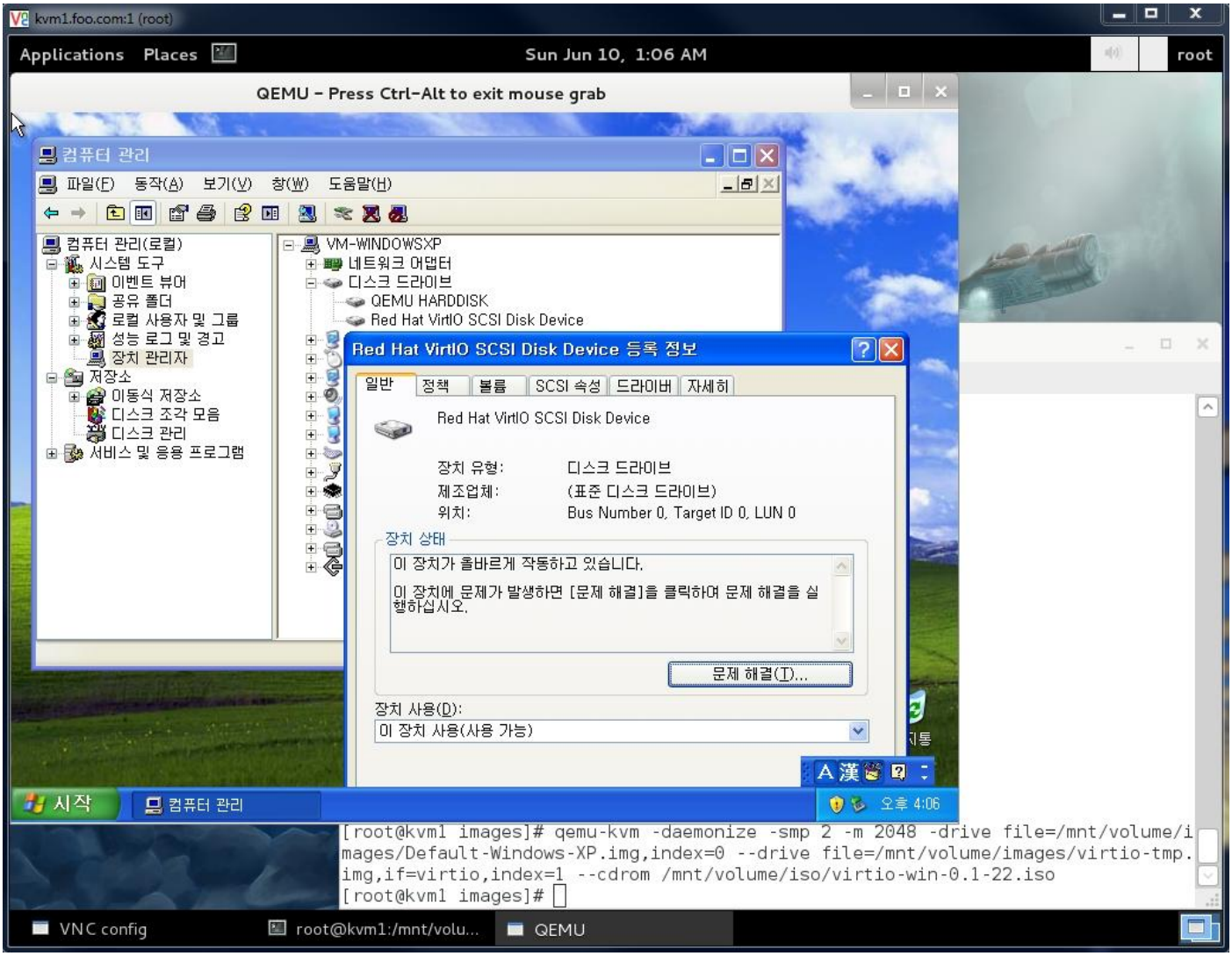
드라이버 설치



적용을 위해 재부팅을 수행하고, Virtio타입의 Disk 인식 확인.
"디스크 1"이라는 이름으로 정상 인식을 확인 하였다.



또한, 장치관리자에서 아래와 같이 "Red Hat Virtio SCSI Disk Device" 드라이버가 느낌표 없이 잘 작동 되고 있음을 확인 하여야 한다.



✓ Virtio 드라이브 설치 (NIC)

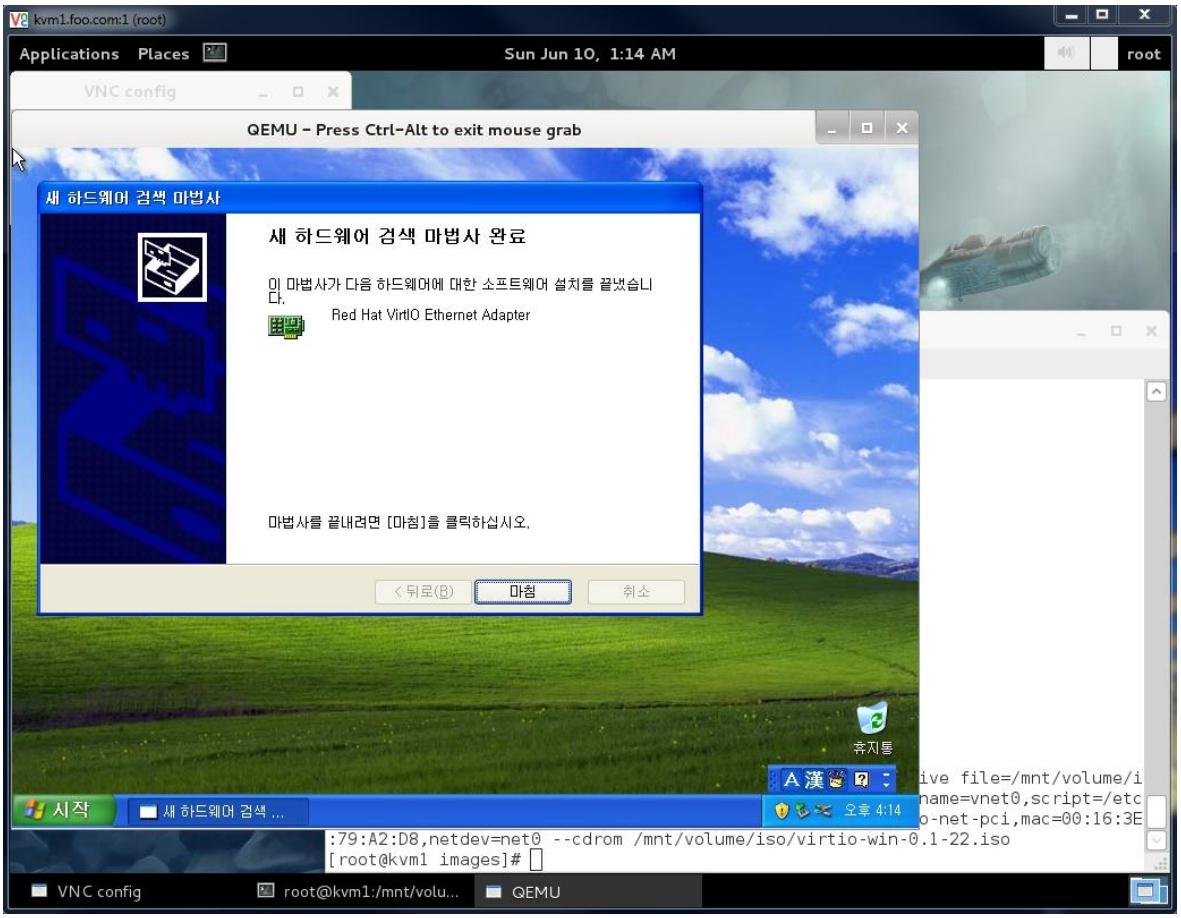
자 이제, 드라이버 인식을 위해 임시로 연결 했던 virtio-tmp.img는 연결 해제 및 삭제 해버리고, 이제 Windows가 설치된 기본 HDD를 Virtio타입으로 변경하고 NIC까지 Virtio로 연결 하면 된다.

(virtio-win-0.1-22.iso 드라이버 CD는 NIC 드라이버 설치를 위해 조금더 연결 해두어야 한다.)

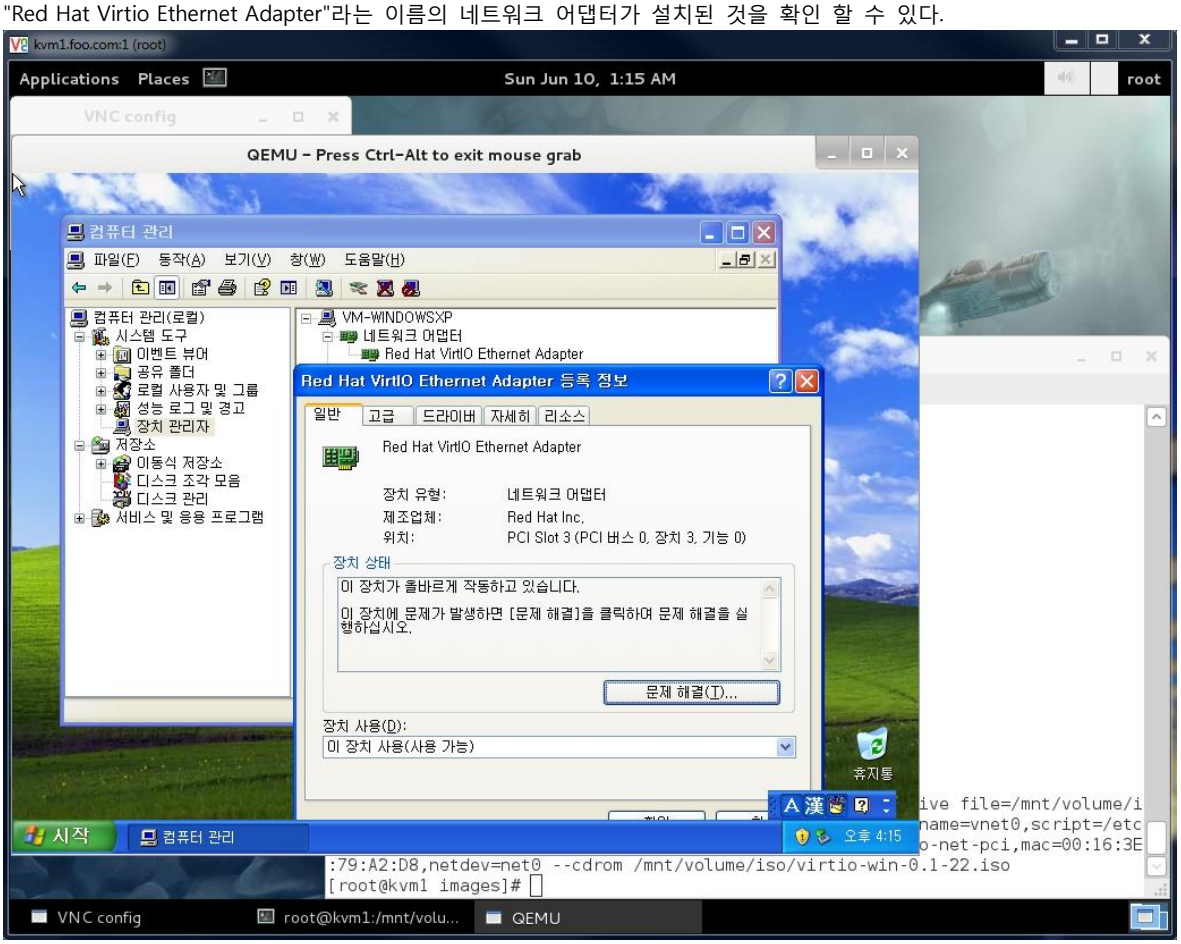
⚠ Mac Address는 이전 CentOS VM에서 사용하던 값을 그대로 사용하였다. 만일, 다른 VM에서 동일 Mac값을 사용 중일 경우에는, 반드시 다른 Mac값을 생성해서 진행한다.

```
# cd /mnt/volume/images
# rm virtio-tmp.img
# qemu-kvm -daemonize -smp 1 -m 1024 -drive file=/mnt/volume/images/Default-Windows-XP.img,if=virtio -netdev type=tap,ifname=vnet0,script=/etc/kvm-ifup,downscript=/etc/kvm-ifdown,id=net0 -device virtio-net-pci,mac=00:16:3E:79:A2:D8,netdev=net0 --cdrom /mnt/volume/iso/virtio-win-0.1-22.iso
```

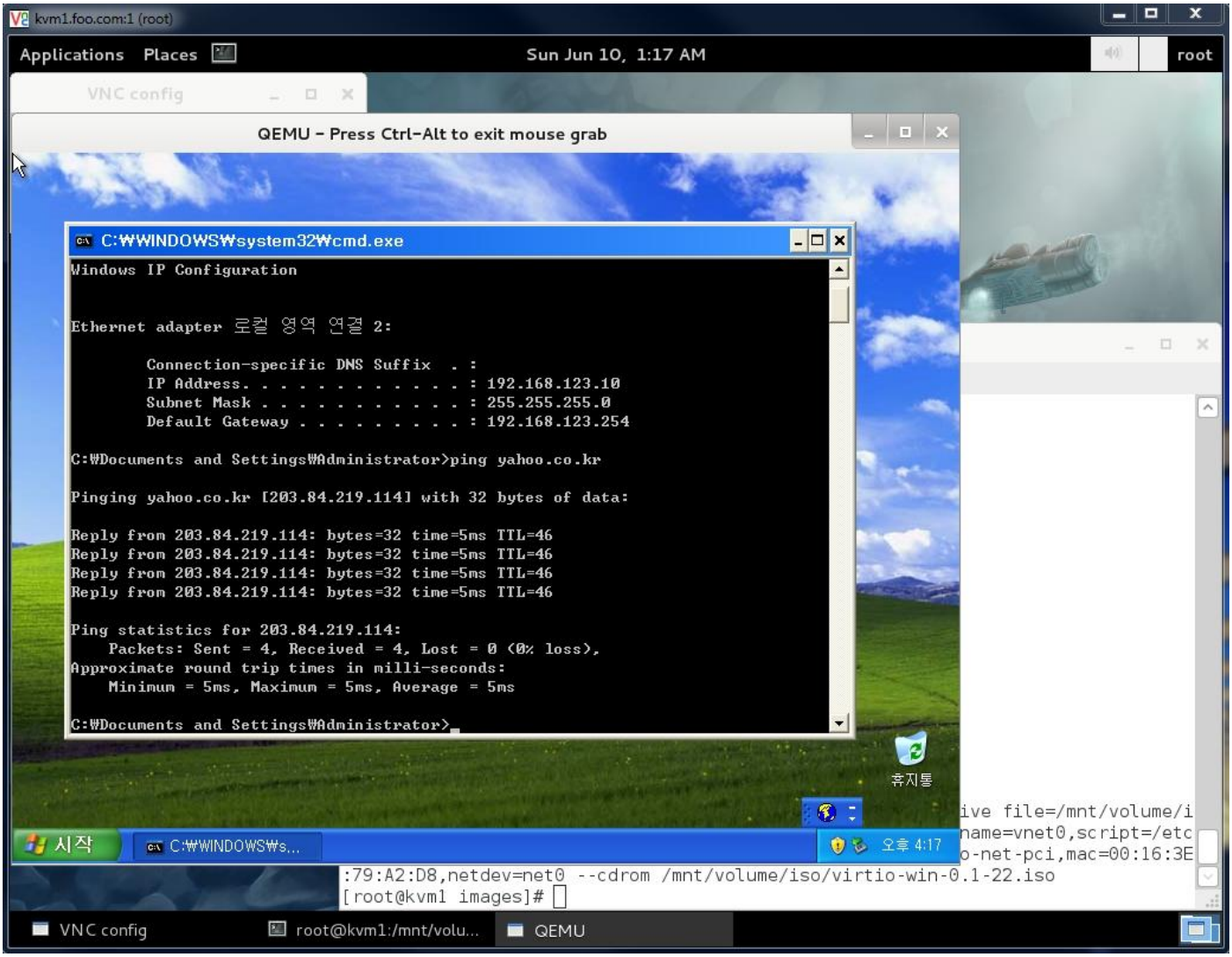
부팅이 완료되면 Windows 는 곧바로 Virtio 타입의 NIC 를 인식하고 드라이버를 요구하게 된다. NIC 의 경우에는 그냥 CD-ROM 장치에서 자동 검색을 통해 설치 되도록 하면 된다.



"Red Hat Virtio Ethernet Adapter"라는 이름의 네트워크 어댑터가 설치된 것을 확인 할 수 있다.



최종적으로 Networking 테스트를 아래와 같이 진행한다.



이제 virtio-win-0.1.22.iso CD 이미지는 필요 없으니, 아래와 같은 명령만으로 Windows 를 실행 하면 될 것이다.

```
# qemu-kvm -daemonize -smp 1 -m 1024 -drive file=/mnt/volume/images/Default-Windows-XP.img,if=virtio -netdev type=tap,ifname=vnet0,script=/etc/kvm-ifup,downscript=/etc/kvm-ifdown,id=net0 -device virtio-net-pci,mac=00:16:3E:79:A2:D8,netdev=net0
```

자, 이로써 Windows XP + Virtio NIC/Disk 설치하기를 진행 해보았다. 다소 복잡한듯한 인상을 받았을지도 모르나, 초기 한번만 수행하고 Template 로 만들어 두면 같은 수고는 반복하지 않아도 된다. 그럼 Windows VM 설치는 이정도로 마무리.....

⚠ Windows 의 Template 생성의 경우, Linux 때의 단순 파일 복제와는 달리 Sysprep 이라는 도구를 통해 '봉인' 작업을 거쳐야 한다. Windows 의 고유 특성(OS 마다 SSID 라는 고유 보안 ID 부여)이 중복되어서는 안되기 때문이다. Sysprep 에 대한 설명은 분량상 생략 한다. 관심이 있다면 아래 링크나 인터넷 문서를 참조. (주의 할 것은 Virtio 타입의 Disk/NIC 상태를 봉인할때는 "3 사 드라이버 포함" 모드로 봉인을 수행 해야 한다.)

http://www.google.co.kr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&ved=0CE0QFjAD&url=http%3A%2F%2Fwww.answersthatwork.com%2FDownload_Area%2FATW_Library%2FWinXP_Professional%2FWinXP__3-Setup-How_to_SYSPREP_a_Windows_XP_PC_setup.pdf&ei=3nnTT9CnFqjLmAXjncGGAw&usg=AFQjCNGnrd8ZEwxGIvJfppl47VD8LSXw9w&sig2=p6fXtxVcIADVqML-d7whug

http://technet.microsoft.com/ko-kr/library/cc721940(v=ws.10).aspx

5. Qemu-Monitor

자, 드디어 이제부터 QEMU의 숨겨진(?) 세상! 바로 Qemu-Monitor 이다.

Qemu-Monitor는 QEMU 에뮬레이터가 실행 중일 때, 에뮬레팅 상태를 체크하거나, 변경 등의 모든 제어를 할 수 있는 모니터 콘솔이라고 생각하면 이해하기 쉬울 것이다. 특정 Host에서 작동중인 VM을 다른 Host로 서비스 다운타임 없이 Live 이전을 한다거나, 스냅샷을 찍어 두었다가 원할 경우 해당 시점으로 되돌리거나, 심지어 Running상태에서 메모리 양을 조절한다거나 등등을 제어 할 수 있다.

이 모니터콘솔은 매우 많은 기능과 정보를 제공하나, "Intro & Basic"라는 소제목이 의미하듯이, 아래와 같은 자주 이용되고 유용한 기능들을 통해 Qemu-Monitor에 대해 이해하는 정도로 해두고, 활용 테크닉이나 상세 기능 파악은 다음 문서에서 집중적으로 다뤄보도록 할 예정이다.

- Info : 각종 VM 운영 상태 정보 확인.
- Live-Migration : VM Shutdown없이, Running상태 그대로를 다른 Host로 이전.
- Snapshot : VM의 특정시점을 저장해 두었다가, 필요 시 해당 시점으로 롤백.

[Qemu-Monitor 관련 추가 자료]
http://en.wikibooks.org/wiki/QEMU/Monitor
http://wiki.qemu.org/download/qemu-doc.html#pcsys_005fmonitor

● Qemu-Monitor 콘솔 접근

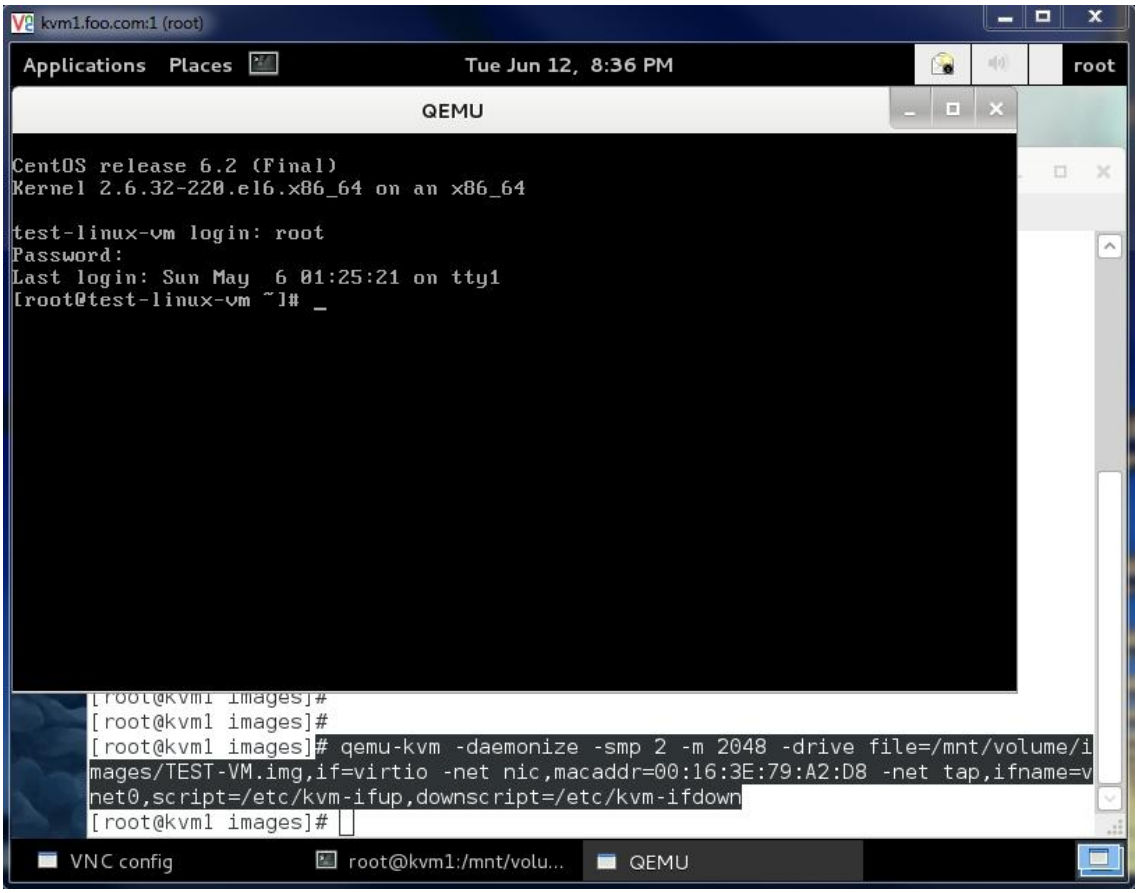
먼저, 앞서 여러 번 수행했던, CentOS VM을 하나 생성해보자. (Base-Image타입으로 생성해보자.)

```
# cd /mnt/volume/images/

# qemu-img create -b Default-CentOS.img -f qcow2 TEST-VM.img
Formatting 'TEST-VM.img', fmt=qcow2 size=53687091200 backing_file='Default-CentOS.img' encryption=off cluster_size=65536
```

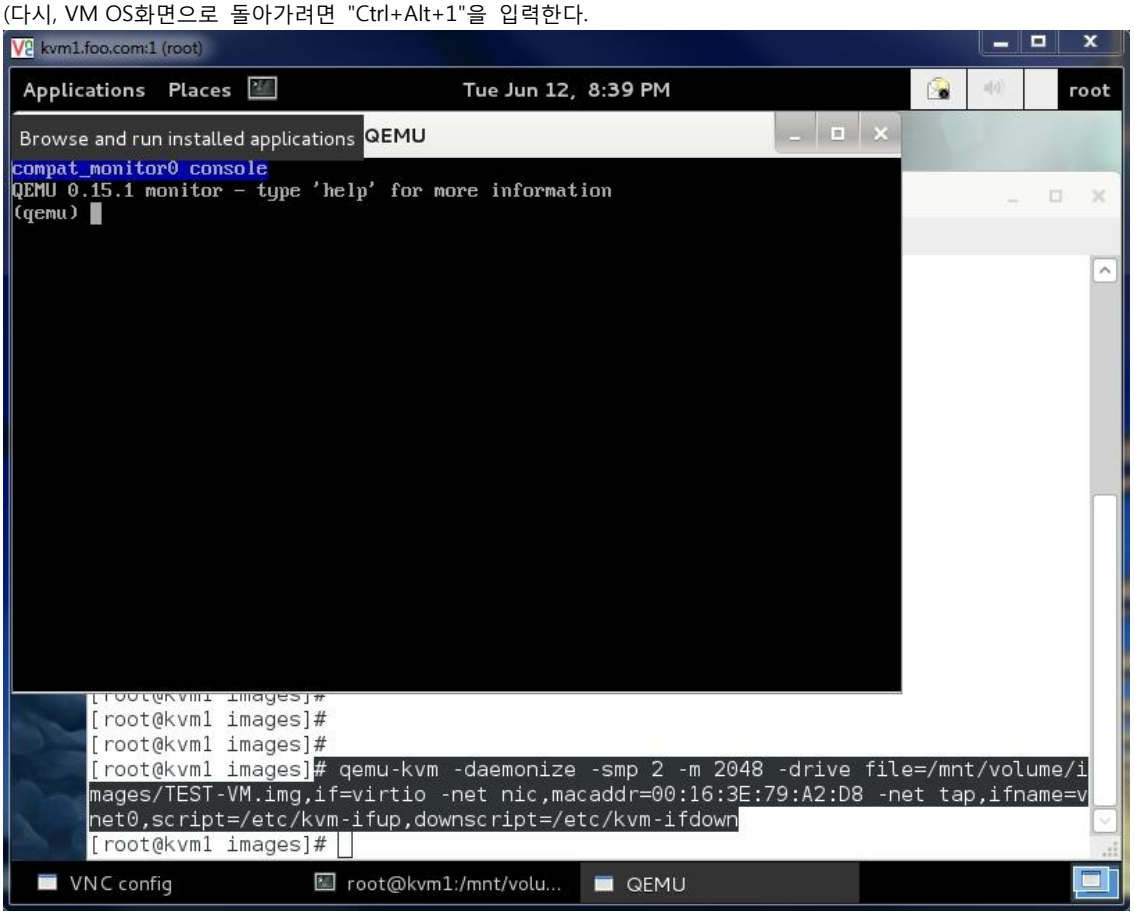
자, Default-CentOS.img 초기 이미지로부터 TEST-VM.img라는 새로운 VDI를 만들었다. 이것으로 부팅을 해보자. 물론 NIC등등 앞서 실습했던 것들 모두 적용해본다.

```
# qemu-kvm -daemonize -smp 2 -m 2048 -drive file=/mnt/volume/images/TEST-VM.img,if=virtio -net nic,macaddr=00:16:3E:79:A2:D8 -net tap,ifname=vnet0,script=/etc/kvm-ifup,downscript=/etc/kvm-ifdown
```



정상적으로 Booting 되었다.

이런 류의 모든 QEMU는 콘솔화면에서 "Ctrl+Alt+2"키를 입력하면 Qemu-Monitor 콘솔로 전환이 된다. 아래 그림과 같이...
(다시, VM OS화면으로 돌아가려면 "Ctrl+Alt+1"을 입력한다.



프롬프트가 (qemu)로 나타나며 이 콘솔이 Qemu-Monitor이다. 단, 로컬콘솔에서만 접근이 가능하다는 단점이 있다. 원격에서 접근을 하기 위해서는 QEMU에서는 -monitor 라는 옵션을 사용할 수 있다.

사용은 qemu 실행에 "-monitor dev"형태로 옵션을 주는데, 여기서 dev가 Qemu-Monitor을 노출 시킬 장치를 정의 해주어야 한다.

자주 사용될 수 있는 예를 통해 살펴보자.

✓ tcp/telnet 소켓 접근

TCP를 통해, Qemu-Monitor를 노출 시키는 방법이다.

형식) -monitor tcp:{인터페이스IP}:{PORT},server,nowait

```
# qemu-kvm {기타옵션} -monitor tcp:192.168.1.1:4444,server,nowait
```

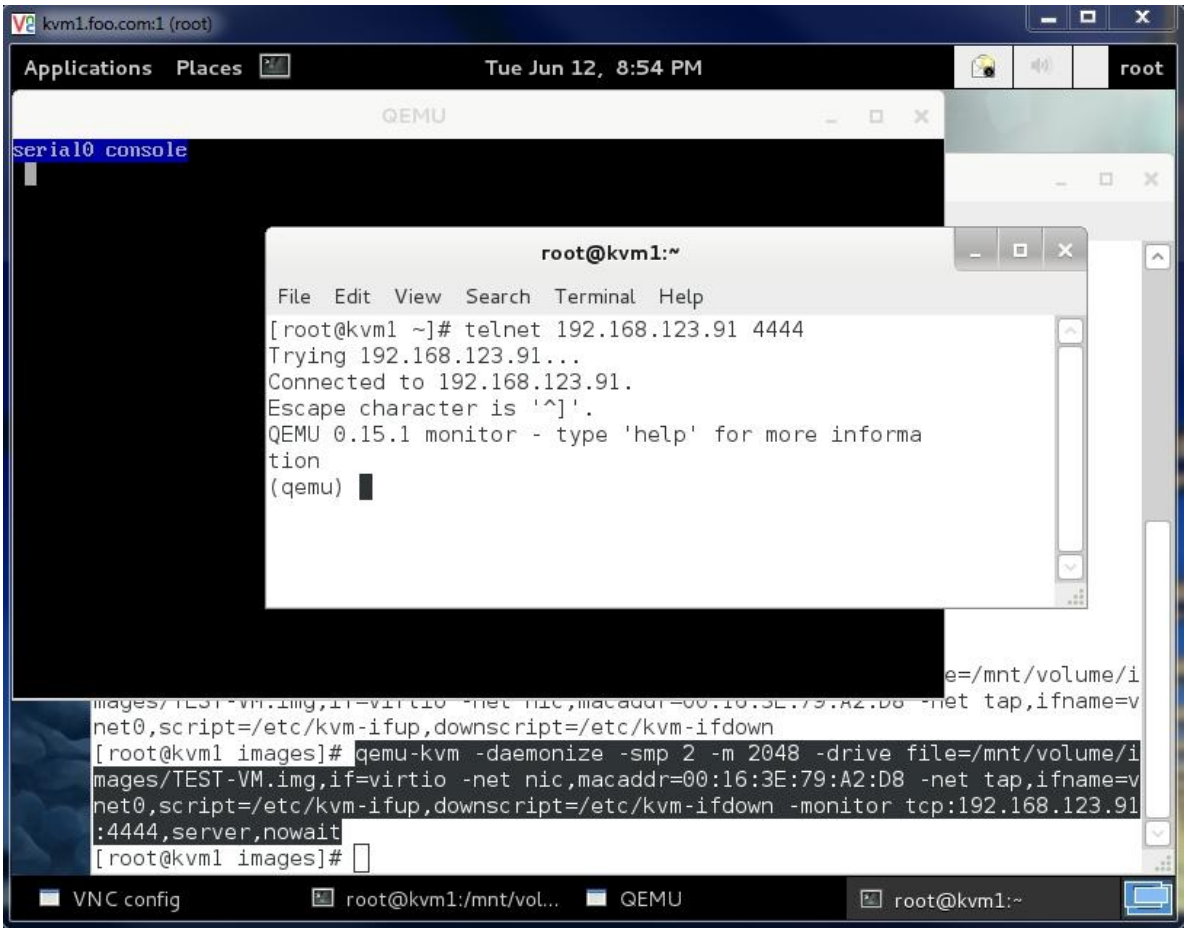
(옵션 항목)

tcp/telnet	TCP 소켓 또는 Telnet 프로토콜 타입을 지정.
192.168.123.91	Listening 할 IP(인터페이스) 지정. 모든 IP에 대해 지정하려면, Blank처리 한다. (TCP::4444,server.nowaiit)
4444	Listening 할 Port지정.
server	Server모드로 Listening.
nowait	클라이언트의 Socket 어플리케이션을 기다리지 않는다.

실제로, VM 실행시 옵션을 주고 테스트 해보자.

```
# qemu-kvm -daemonize -smp 2 -m 2048 -drive file=/mnt/volume/images/TEST-VM.img,if=virtio -net nic,macaddr=00:16:3E:79:A2:D8 -net tap,ifname=vnet0,script=/etc/kvm-ifup,downscript=/etc/kvm-ifdown -monitor tcp:192.168.123.91:4444,server,nowait
```

VM 콘솔에서 "Ctrl+Alt+2"를 입력해 두고, 터미널 창에서 Qemu-Monitor로 접근한 스크린샷이다. 특이할 점은 -Monitor 접근은 유일해야 한다는 것이다. 다중 접근을 허용치 않는다. 그림에서 콘솔에서 "Ctrl+Alt+2"를 입력한 결과를 보라. (qemu)프롬프트가 아닌 "serial0 console" 이라는 콘솔 이름만 나타나며 Qemu-Monitor접근이 안된다. 여기서는 "-monitor"옵션으로 지정한 TCP/4444 포트로만, 그것도 192.168.123.91 이라는 인터페이스로만 접근이 가능하다는 이야기가 된다.



● info 명령

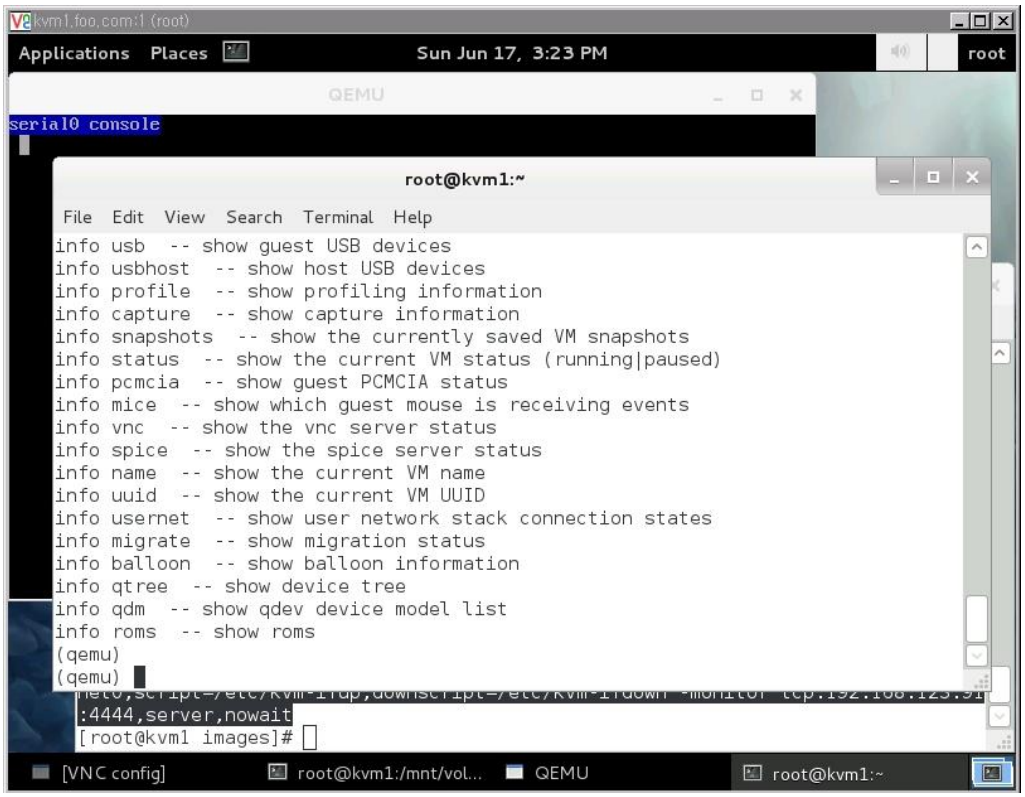
Qemu-Monitor콘솔 명령어 가운데 하나인 info는, VM이 구동되는 모든 H/W상태 및 Snapshot, Migration 상태를 포함해 Guest System에 대한 모든 정보를 확인 할 수 있는 도구이다.

앞서, TCP접근이 가능한 형태의 VM을 이용해 이어서 확인 해보자.

콘솔에서 'help'를 실행하면 그림과 같이 무수히 많은 명령어들이 수 페이지에 걸쳐 뿌려질 것이다.

(이번 문서에서는 대표적인 몇가지에 대해 맛보기 정도로 살펴보고, 차기 문서에서 이 명령들에 대해 상세히 다뤄볼 것이다.)

자, 다시 'help info'를 실행해보자. info명령 하위에 속한 서브명령어(subcommand)들이 나열된다.



✓ Info 하위명령 목록

"info help"의 결과를 표로 정리한 것이다.

info [subcommand]	기능 설명
info version	Show the version of QEMU
info network	Show the network state
info chardev	Show the character devices
info block	Information about block devices, such as hard drives, floppy drives, or CD-ROMs
info blockstats	Read and write statistics on block devices
info registers	Show the cpu registers
info cpus	Shows information about available CPUs
info history	Show the command line history
info irq	Show the interrupts statistics (if available)
info pic	Show i8259 (PIC) state
info pci	Show PCI info
info tlb	Show virtual to physical memory mappings
info mem	Show the active virtual memory mappings
info jit	Show dynamic compiler info
info kvm	Show KVM information
info numa	Show NUMA information
info usb	Show guest USB devices
info usbhost	Show host USB devices
info profile	Show profiling information
info capture	Shows the capture (audio grab) information
info snapshots	Show the currently saved VM snapshots
info status	Show the current VM status (running|paused)
info pcmcia	Show guest PCMCIA status
info mice	Show which guest mouse is receiving events
info vnc	Show the vnc server status

info spice	Show the spice server status
info name	Show the current VM name
info uuid	Show the current VM UUID
info usernet	Show user network stack connection states
info migrate	Show migration status
info balloon	Shows the balloon device information
info qtree	Show device tree
info qdm	Show qdev device model list
info roms	Show roms

✓ **info 사용 예제**

본 문서에서 상기 명령들 모두를 다루진 못하며, 대표적인 것들 몇가지를 실행해 보는 것으로 마무리 하자. 실습에 사용된 Guest OS의 실행 옵션은 아래와 같다.

```
# qemu-kvm -daemonize -smp 2 -m 2048 -drive file=/mnt/volume/images/TEST-VM.img,if=virtio -net nic,macaddr=00:16:3E:79:A2:D8 -net tap,ifname=vnet0,script=/etc/kvm-ifup,downscript=/etc/kvm-ifdown -monitor tcp:192.168.123.91:4444,server,nowait
```

➢ KVM 모듈 정보

```
(qemu) info kvm
info kvm
kvm support: enabled
```

➢ Network 정보

```
(qemu) info network
info network
VLAN 0 devices:
  tap.0: type=tap,ifname=vnet0,script=/etc/kvm-ifup,downscript=/etc/kvm-ifdown
  rtl8139.0: type=nic,model=rtl8139,macaddr=00:16:3e:79:a2:d8
Devices not on any VLAN:
```

➢ CPU 정보

```
(qemu) info cpus
info cpus
* CPU #0: pc=0xffffffff810375ab (halted) thread_id=2001
  CPU #1: pc=0xffffffff810375ab (halted) thread_id=2002
```

➢ Memory 정보

```
(qemu) info mem
info mem
00007f4090000000-00007f4090001000 0000000000001000 urw
00007f4094000000-00007f4094001000 0000000000001000 urw
00007f4098000000-00007f4098002000 0000000000002000 urw
00007f4098002000-00007f4098003000 0000000000001000 ur-
00007f409c000000-00007f409c001000 0000000000001000 urw
(중략)
```

➢ Block 장치 정보

```
(qemu) info block
info block
virtio0: removable=0 file=/mnt/volume/images/TEST-VM.img backing_file=Default-CentOS.img ro=0 drv=qcow2 encrypted=0
```

```
ide1-cd0: removable=1 locked=0 [not inserted]
floppy0: removable=1 locked=0 [not inserted]
sd0: removable=1 locked=0 [not inserted]
```

- Block 장치 사용 정보

```
(qemu) info blockstats
info blockstats
virtio0: rd_bytes=80046080 wr_bytes=3474432 rd_operations=3657 wr_operations=264
ide1-cd0: rd_bytes=0 wr_bytes=0 rd_operations=0 wr_operations=0
floppy0: rd_bytes=0 wr_bytes=0 rd_operations=0 wr_operations=0
sd0: rd_bytes=0 wr_bytes=0 rd_operations=0 wr_operations=0
```

- Snapshot 정보 (스냅샷 생성 내역이 현재는 없다. 뒤에 Snapshot 설명 파트에서 다시 다룰 것이다.)

```
(qemu) info snapshots
info snapshots
There is no snapshot available
```

- Guest(VM)의 상태 정보

```
(qemu) info status
info status
VM status: running
```

- Migrate 정보

```
(qemu) info migrate
info migrate
```

- PCI 장치 정보

```
(qemu) info pci
info pci
  Bus  0, device   0, function 0:
    Host bridge: PCI device 8086:1237
      id ""
  Bus  0, device   1, function 0:
    ISA bridge: PCI device 8086:7000
      id ""
  Bus  0, device   1, function 1:
    IDE controller: PCI device 8086:7010
      BAR4: I/O at 0xc000 [0xc00f].
      id ""
  Bus  0, device   1, function 3:
    Bridge: PCI device 8086:7113
      IRQ 9.
      id ""
  Bus  0, device   2, function 0:
    VGA controller: PCI device 1013:00b8
      BAR0: 32 bit prefetchable memory at 0xf0000000 [0xf1ffffff].
      BAR1: 32 bit memory at 0xf2000000 [0xf2000fff].
      BAR6: 32 bit memory at 0xffffffffffffffff [0x0000fffe].
      id ""
  Bus  0, device   3, function 0:
    Ethernet controller: PCI device 10ec:8139
```

```
        IRQ 11.
        BAR0: I/O at 0xc100 [0xc1ff].
        BAR1: 32 bit memory at 0xf2020000 [0xf20200ff].
        BAR6: 32 bit memory at 0xffffffffffffffff [0x0000fffe].
        id ""
  Bus  0, device   4, function 0:
    SCSI controller: PCI device 1af4:1001
        IRQ 11.
        BAR0: I/O at 0xc200 [0xc23f].
        BAR1: 32 bit memory at 0xf2040000 [0xf2040fff].
        id ""
```

그 외의 info 하위명령들도 한번씩 실행해 보길 바란다.

● Live-Migration

Live-Migration은 물리적으로 다른 Host장비 사이에서 Running상태의 Guest-VM을 Shutdown이나 서비스 중단 없이(메모리 상태까지 포함) 이전 시키는 작업을 말한다. (단순 Migration은 Image공유를 통해, 다른 Host장비에서 다시 Booting하는 수준을 보통 일컫는다.) 본 문서에서는 Qemu-Monitor콘솔에서 Live-Migration이 어떤 과정을 거쳐 수행되는지 살펴 보고자 한다. 물론, Live-Migration이 가능하기 위한 조건들이 몇가지 있으며 그 중 가장 중요한 포인트는 아래와 같다.

이 장에서는 맛보기 정도만 하고, 상세한 내용은 Qemu-Monitor에 대해 본격적으로 다뤄볼 예정인 다음 편 문서에서 살펴 볼 예정이다.

① 두 장비(이전 되기 전 장비와 이전 되어 갈 장비)의 CPU벤더가 동일해야 한다.
② VDI 파일이 외부의 공유 스토리지에 존재 해야 하며, 두 장비에서 동일한 절대 경로를 통해 접근도 가능해야 한다.

Live-Migration을 위한 이유이기도 했던, 환경 준비과정에서 안내된 두 장비 kvm1, kvm2모두 모든 사양이 동일한 장비이고, NFS를 통해 양쪽 모두 동일한 경로 값으로 VDI를 공유 할 수 있게 해둔 것이다. 따라서 Live-Migration을 위한 준비도 이미 되어 있는 것으로 보면 된다.

QEMU/KVM에서 Live-Migration의 처리 과정은 아래와 같다. 상황은 (A)장비에서 운영중인 Guest-VM을 (B)장비로 서비스 DownTime없이 이전.

① (A)장비에서 구동중인 qemu-kvm 옵션과 동일한 상태에서 아래 형태의 옵션을 추가 하여 (B)장비에서 실행 해둔다.
 # qemu-kvm (중략) -incoming [tcp/udp]:[Listening IP]:[Port Number]
② (A)장비에서 구동되고 있는 VM의 Qemu-Monitor콘솔로 접속 후, 아래 명령을 통해, (B)장비로의 Live-Migration을 시작.
(qemu) migrate -d [tcp/udp]:[Listening IP]:[Port Number]
③ 정상적으로 이전이 되었는지 확인 후, (A)장비의 VM을 종료/제거 한다.

자, 실제로 테스트 해보자.
상황은 앞의 상황과 동일하다. kvm1 장비에서 하기와 같은 옵션으로 실행된 Guest-VM을 kvm2 장비로 Live-Migration을 수행.

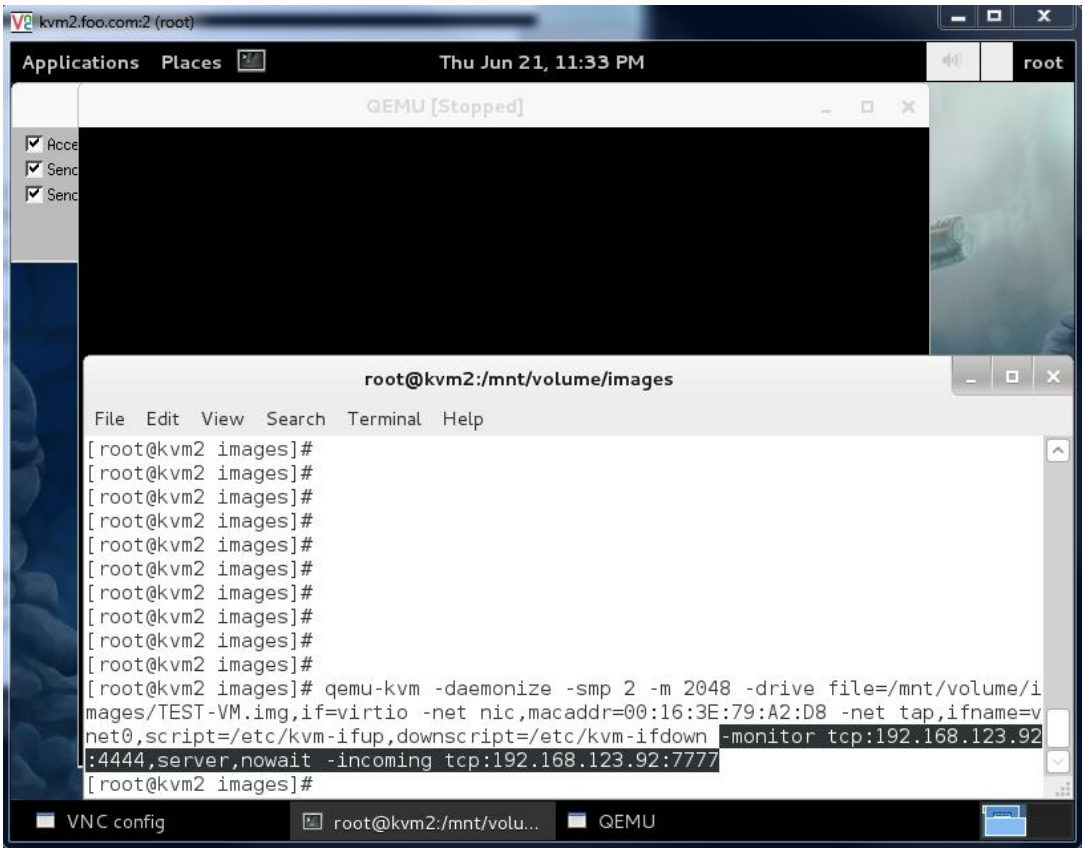
Guest-VM의 실행 옵션은 하기와 같다.
(Qemu-Minitor은 편의상 Ctrl+Alt+2로 사용할 것이므로 제거 되었다. kvm2로 Live-Migration이후에도Monitor을 TCP나 Telnet으로 연결 하고 싶으면, kvm2 장비 IP로 -monitor 옵션 정보를 수정하던지, 공란으로 비워두면 된다.)

```
[root@kvm1 ~]# qemu-kvm -daemonize -smp 2 -m 2048 -drive file=/mnt/volume/images/TEST-VM.img,if=virtio -net nic,macaddr=00:16:3E:79:A2:D8 -net tap,ifname=vnet0,script=/etc/kvm-ifup,downscript=/etc/kvm-ifdown
```

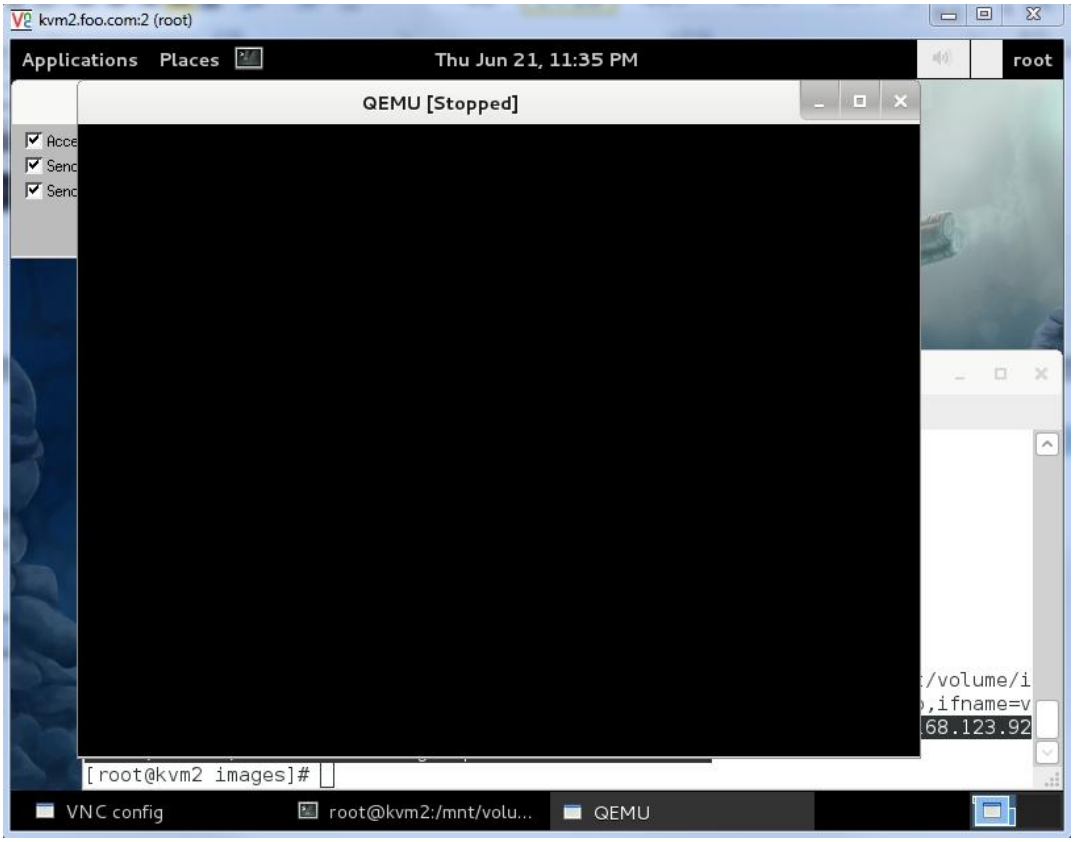
✓ Migratioin 예제

➢ kvm2 장비에 이전 대상 Guest-VM과 동일한 구동 옵션에 -incoming옵션 추가하여 Live-Migration을 받아들일 수 있게끔 실행.

```
[root@kvm2 ~]# qemu-kvm (중략) -incoming tcp:192.168.123.92:7777
```

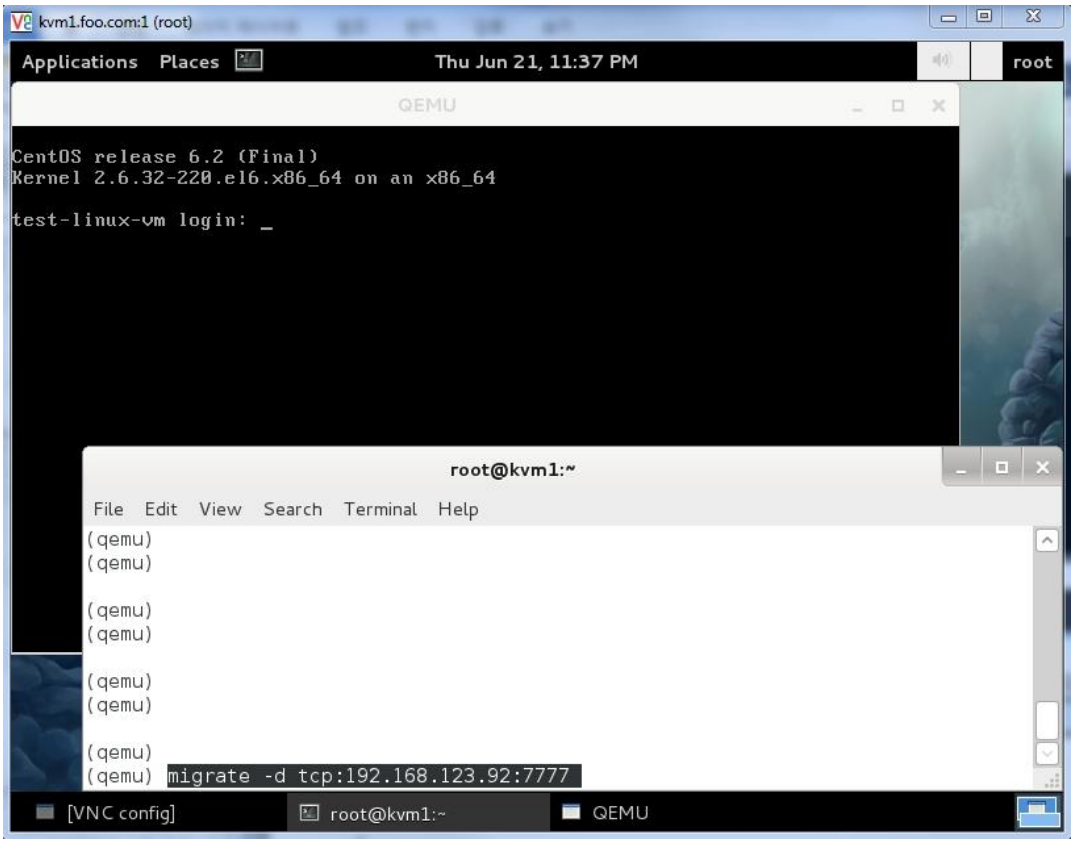


"-incoming" 옵션을 사용하게 되면, Migration에 대해서 Listening 모드로 QEMU/KVM Guest OS가 구동되게 된다. 다시 말해, 어떤 OS의 Booting이나, BIOS로딩이 없다. 단지 대기 모드다. 스크린샷 상에서도 뒤에 검은 화면만 덩그러니 있을 뿐이다.

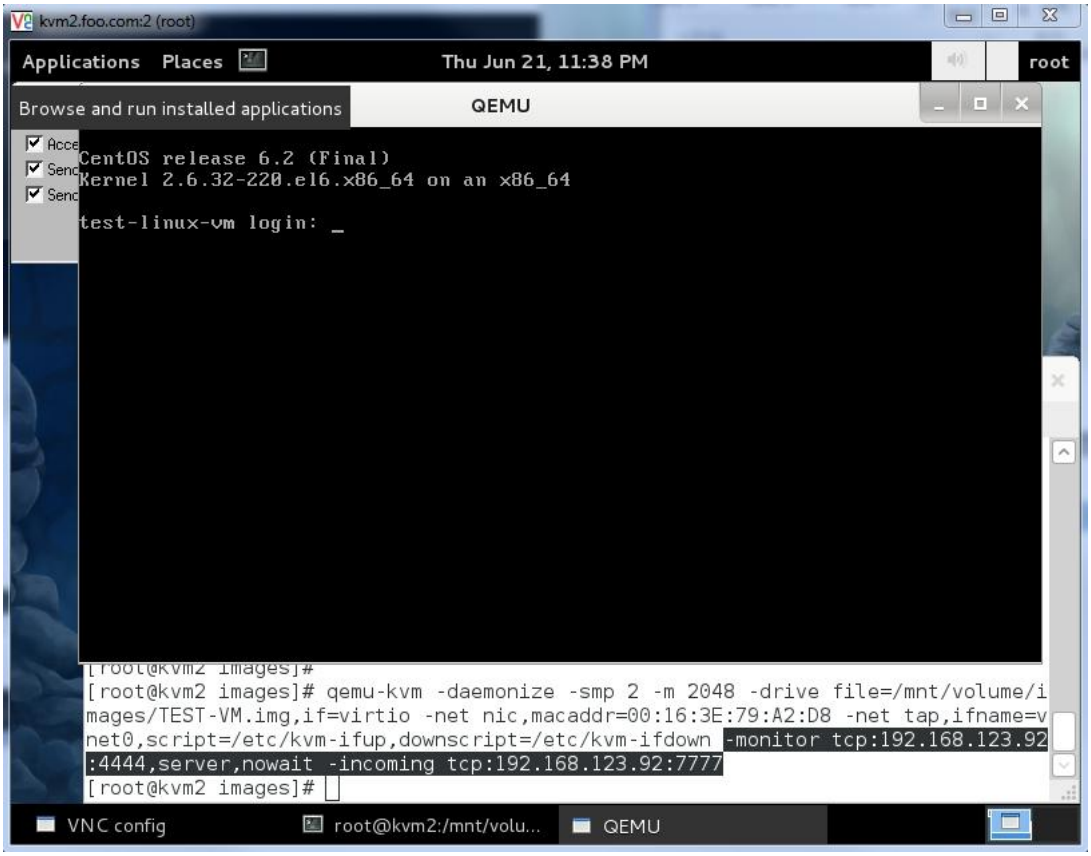


- kvm1 장비에서 구동중인 VM에서 Ctrl+Alt+2를 눌러 Qemu-Monitor콘솔로 진입하여, Migration을 시작.

```
(qemu) migrate -d tcp:192.168.123.92:7777
```



- kvm2. 장비에 Listening 모드로 대기중인 VM에 정상적으로 Live-Migration이 됨을 확인하고, kvm1에서는 구동되었던 VM은 아무 반응이 없이 멈춰 있음을 확인할 수 있으면 정상적으로 수행된 것이다.



- 이제 정상적으로 원하던 Live-Migration 수행이 완료 되었으니, kvm1에 남아 있는 VM은 종료(kill) 처리 하면 모든 Live-Migration 작업은 완료.

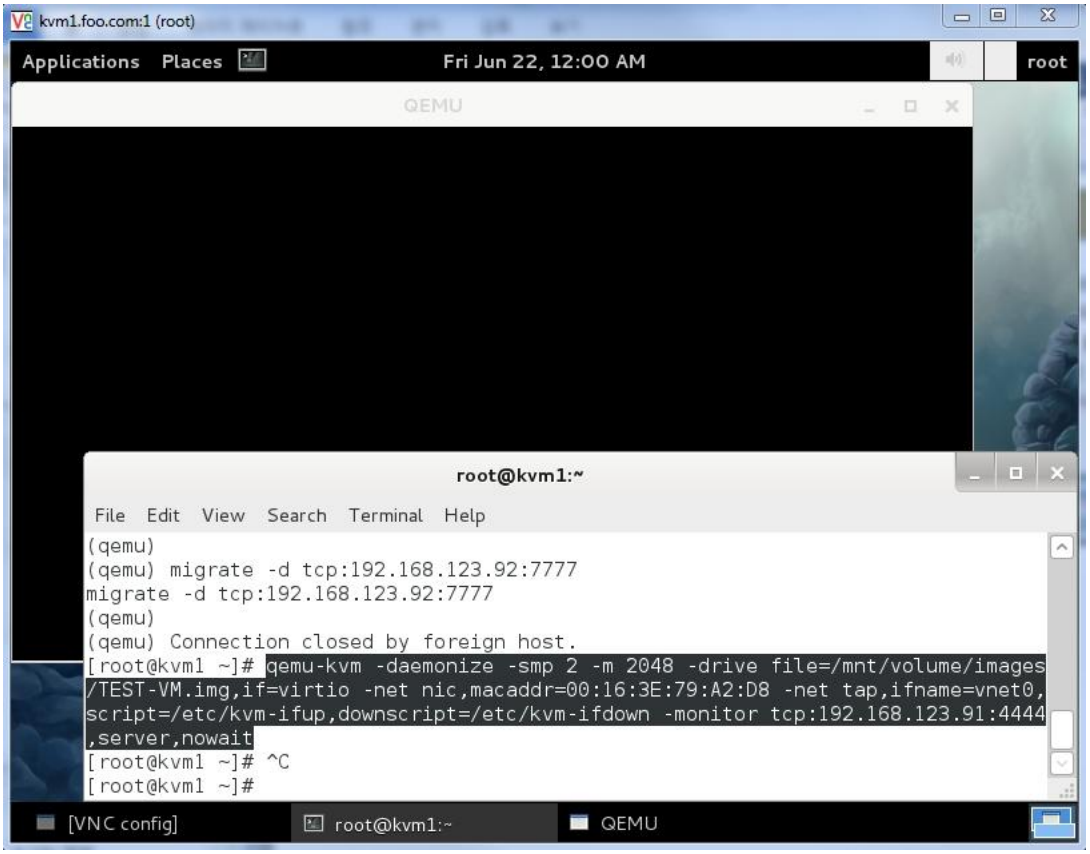
QEMU/KVM에서 Migration/Live-Migration 참조 자료
[URL] http://www.linux-kvm.org/page/Migration#savevm.2Floadvm_to_an_external_state_file_.28using_pseudo-migration.29
[PDF] http://www.linux-kvm.org/wiki/images/5/5a/KvmForum2007$Kvm_Live_Migration_Forum_2007.pdf

● Snapshot

Snapshot.... 사실... 개념이 어떤 파트냐에 따라 상당히 격차가 심한 단어다. 다른 파트의 내용은 거두 절미하고, 본 장에서 다룰 Snapshot에 대해 간단히 설명 하자면, 원하는 시점의 Guest OS의 상태를 Snapshot(정지 사진)을 찍어 두었다가, 차후에 필요 시, 해당 Snapshot을 찍었던 시점으로 Guest OS의 상태(Memory, VDI-Data)를 되돌려 주는 기능이라고 말할 수 있다. 물론 세부적으로 상세히 다루자면 다양한 기능들로 나누어 질 수 있다. 이를테면 Memory상태는 제외한다던지... 어쨌든, 본 문서에서는 앞선 Live-Migration과 마찬가지로 Snapshot이 뭐다~라는 정도의 맛보기만 해보고, Qemu-Monitor을 타겠으로 다룰 다음 문서에서 깊이 있게 다시 다룰 예정이니 그 때 다시 공부해보도록 하자.

자, 그럼 임의의 VM을 아래와 같이 하나 실행 해둔다.

```
qemu-kvm -daemonize -smp 2 -m 2048 -drive file=/mnt/volume/images/TEST-VM.img,if=virtio -net nic,macaddr=00:16:3E:79:A2:D8 -net tap,ifname=vnet0,script=/etc/kvm-ifup,downscript=/etc/kvm-ifdown -monitor tcp:192.168.123.91:4444,server,nowait
```



Snapshot과 관련된 Qemu-Monitor 명령들은 아래 표를 참조.

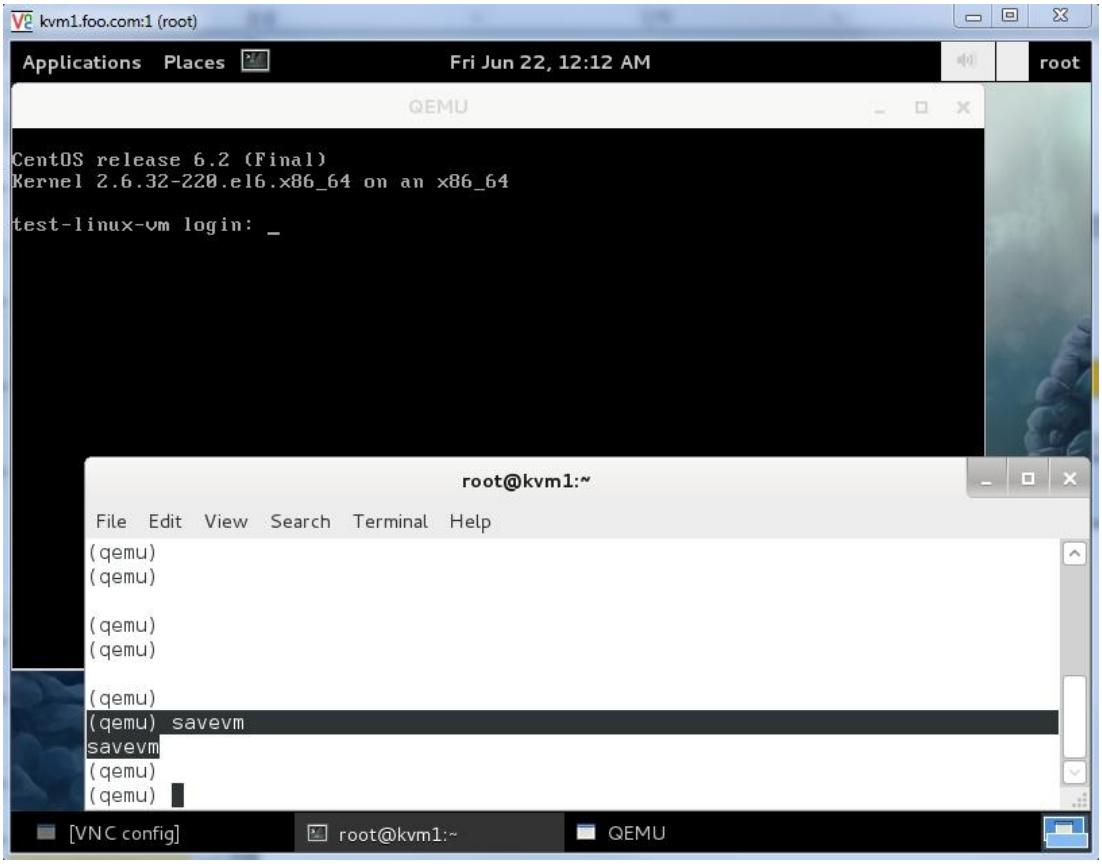
Snapshot Command (Qemu-Monitor)	기능 설명
savevm [tag|id]	save a VM snapshot. If no tag or id are provided, a new snapshot is created
loadvm tag|id	restore a VM snapshot from its tag or id
delvm tag|id	delete a VM snapshot from its tag or id

✓ Snapshot 생성/확인

Qemu-Monitor로 접근해서, 하기와 같은 형태로 Snapshot을 생성한다.

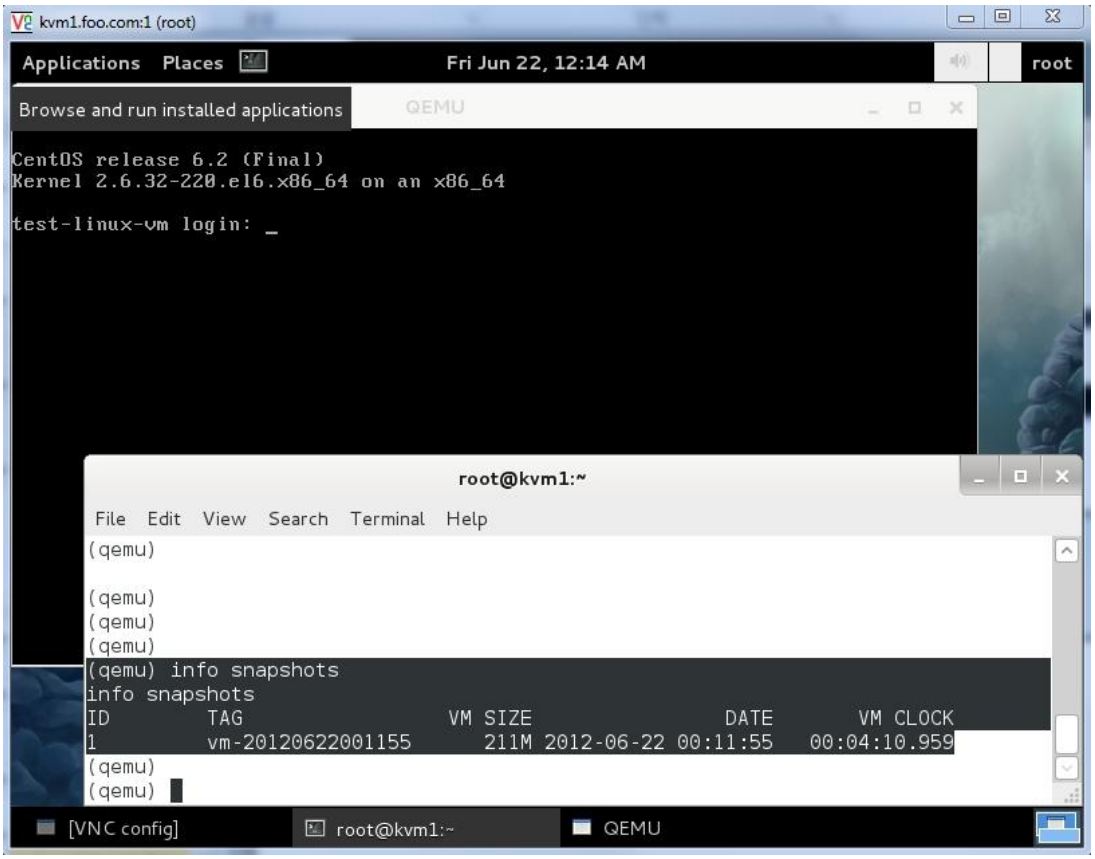
(앞선 표에서 살짝 설명 되었듯이, savevm 명령에서는 추후 식별의 용이성을 위해 tag나 id값을 별도로 부여 할 수 있다.)

```
(qemu) savevm
savevm
```



생성된 Snapshot 확인은, 앞서 살펴봤던 info 명령을 통해 가능하다.
(여러 개의 Snapshot도 가능하며, 목록으로 나타날 것이다.)

```
(qemu) info snapshots
info snapshots
ID        TAG                 VM SIZE                DATE       VM CLOCK
1         vm-20120622001155      211M 2012-06-22 00:11:55   00:04:10.959
```



자 정보가 어떤가? TAG나 ID는 savevm 실행시 별도로 주지 않았던 관계로, 자동으로 태깅/넘버링 되었다. 기타 정보로, Snapshot 크기(SIZE), Snapshot이 수행된 시간(DATE), Guest OS내부에서 인식되는 시간(VM CLOCK) 정보가 기록되어 있다. 추후, 이 정보를 바탕으로 Guest OS를 롤백시키면 되는 것이다.

⚠ 현재 시점을 별도로 Snapshot으로 저장하지 않고 과거 시점의 Snapshot으로 롤백 하게 되면, 다시 현재로 돌아 올 수 없다. 명심해야 한다.

✓ Snapshot 으로 시스템 복구

자, 그럼 극단적(?)인 실습을 해보도록 하자.

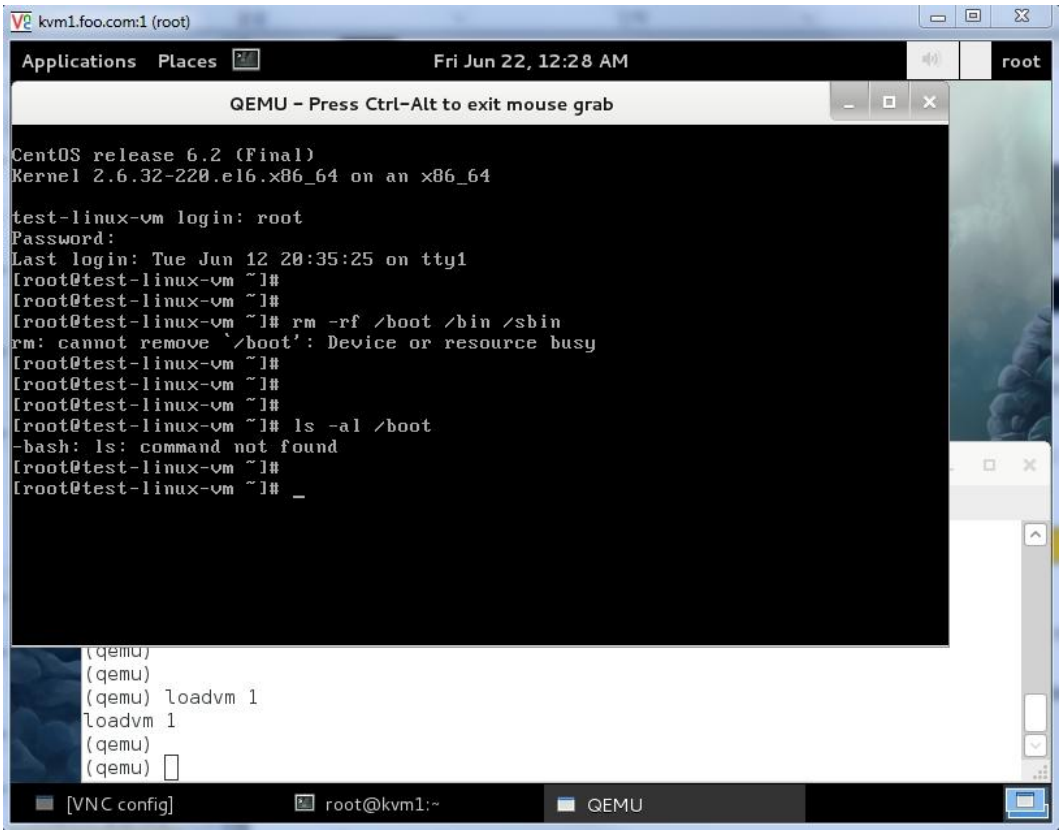
Guest OS내에서 어떤한 명령도 실행 할 수 없고, 심지어 부팅도 불가능한 상황에 빠졌다는 가정하에, /boot, /bin, /sbin 디렉토리를 모두 삭제 할 것이다. (업그레이드 이후 오작동 상황을 가정)

이런 상황에서 앞서 생성해두 었던 Snapshot을 이용해 시스템을 복구 해보는 과정을 진행해보자.

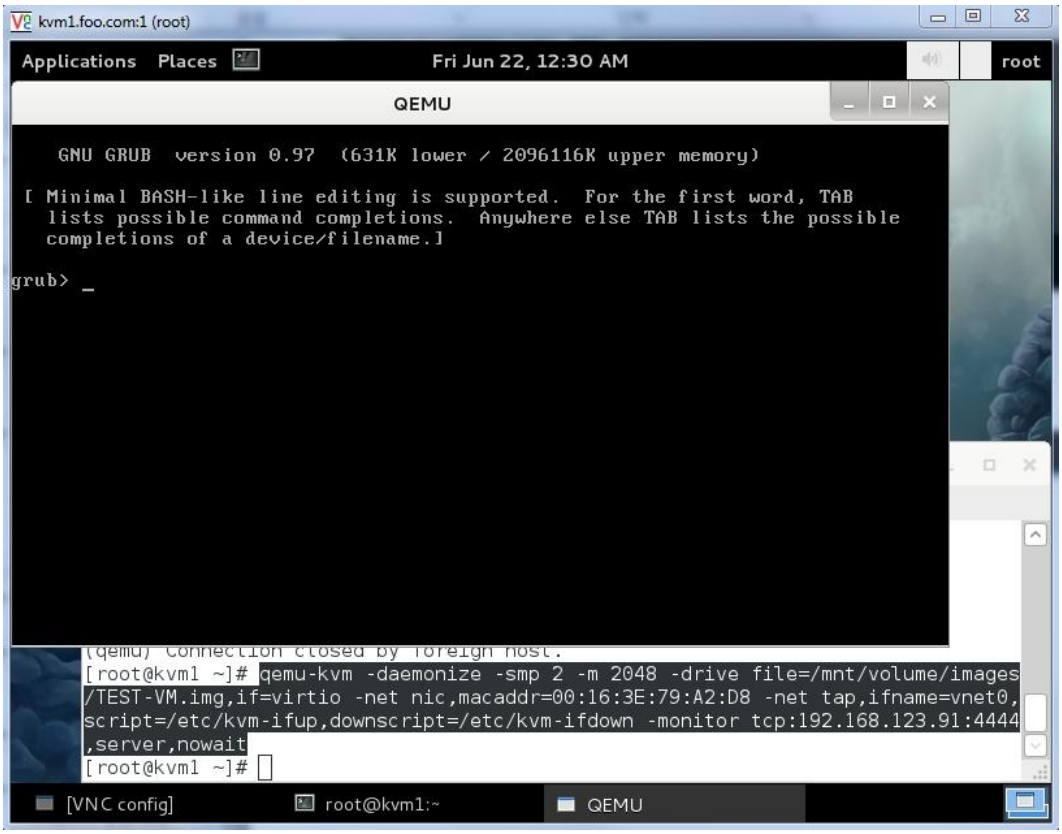
- 시스템 훼손

```
# rm -rf /boot /bin /sbin
```

아래 스크린샷과 같이 삭제 이후, 잘되는 ls명령어도 실행 할 수 없게 되었다.

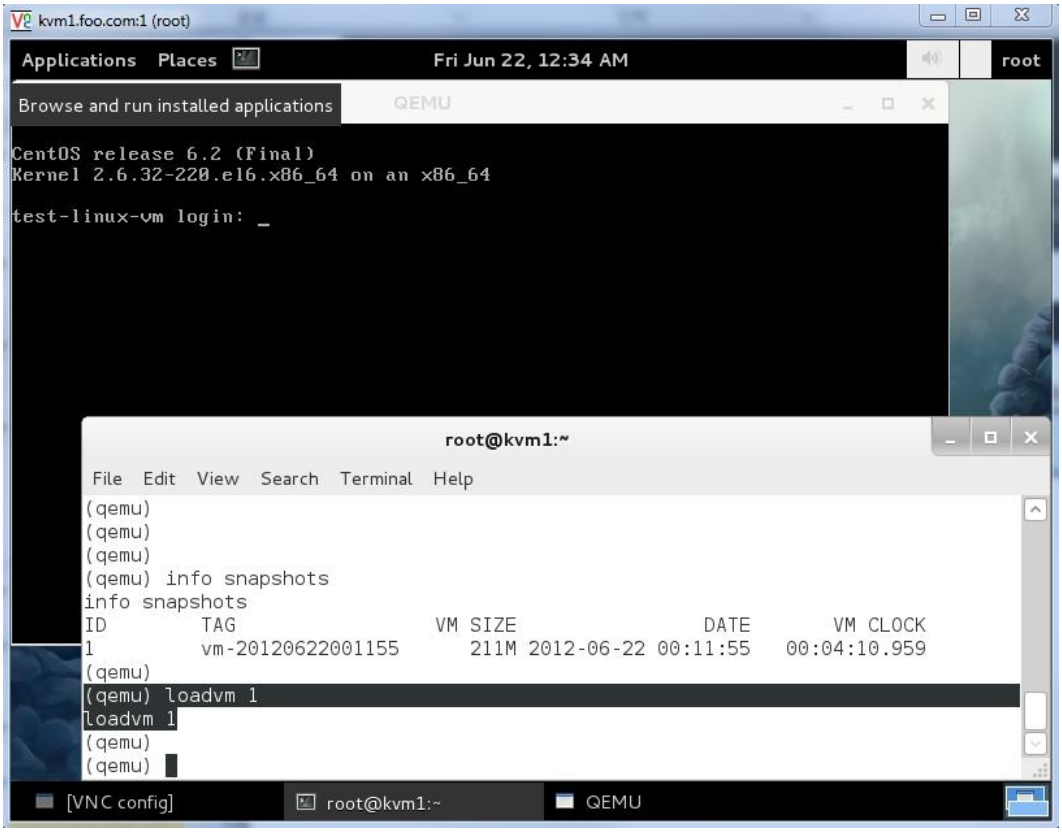


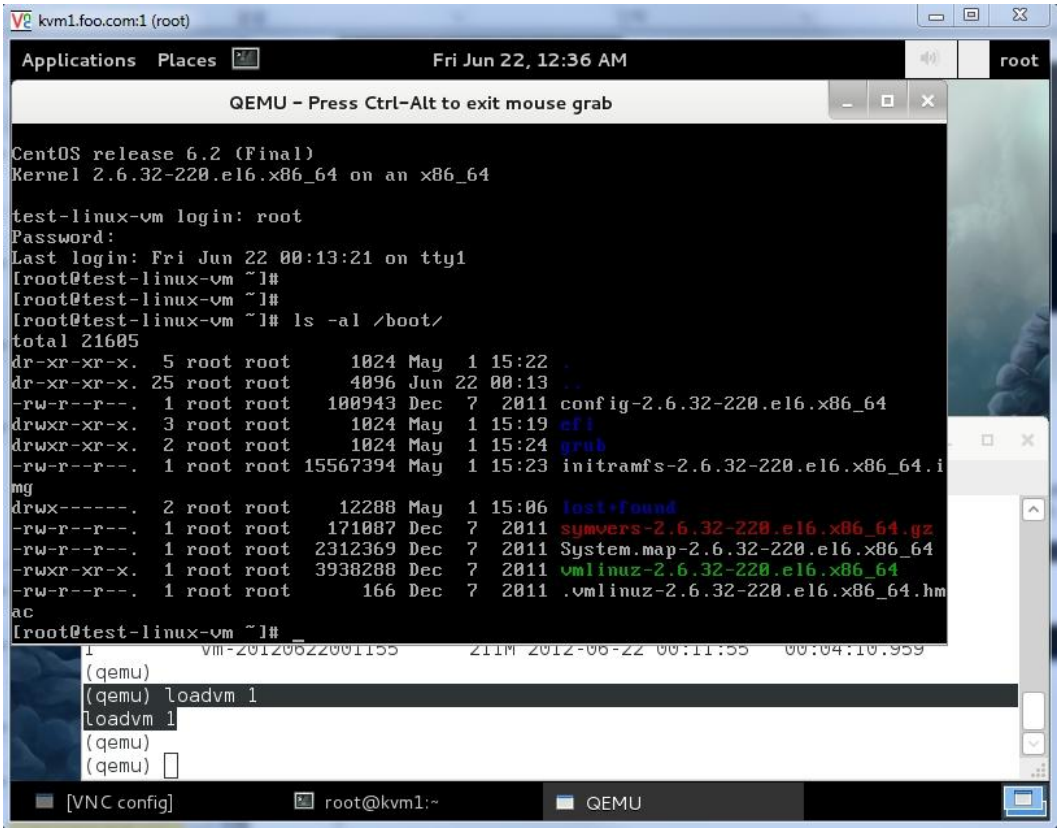
Reboot명령도 실행되지 않을 테니, qemu-kvm을 강제 종료(Kill)하고, 다시 실행 해도 아래와 같이 부팅이 불가능하다. 부팅 할 수 있는 Kernel과 램디스크파일들이 보두 삭제(/boot 디렉토리) 되었기 때문이다.



- 이제, 앞서 savevm으로 생성해 두었던 Snapshot으로 시스템을 순식간에 복구(엄밀히 롤백) 해보자.

```
(qemu) loadvm 1
loadvm 1
```





유후~ 모든게 정상화 되었다. 그것도 순식간에~ 얼마나 유용한가~ Snapshot에 대해 대충 감이 올 것이다.

⚠ (주의) 위에서 안내된 Snapshot 방식에 대해 오해하는 사람들을 가끔 만난다. 위 Snapshot을 절대로 통상적인 Backup 개념으로 사용해서는 안된다. Snapshot 시점 관리와 그 이후에 변화되는 데이터 관리의 오버헤드가 존재하기 때문이다. 여기서 안내된 Snapshot 방식의 용도는 어떤 패키지의 업데이트, 또는 OS전체에 대한 업그레이드로 인한 오작동이 예상될 경우를 대비해 생성해두는 보험(?)정도로 사용해야

한다. 따라서, 해당 작업이 완료되었을 경우, 가능하다면, Snapshot은 삭제 처리 하는 것이 성능상 이롭다. (반드시 이렇게 해야 한다는 것은 아니니, 각자가 판단할 몫이다.)

6. '다음'을 기약하며....

정리를 한번 해야지.. 해야지... 하는 생각은 늘 있었다. 그런데, 이런 핑계 저런 핑계로 미루다 시작 했지만, 하나의 문서로 마무리를 못하고 겨우, 맛보기 수준의 "#1" 이라는 우스꽝스러운 꼬리표를 달고 1막을 내리게 되는 것 같다.

막상 시작해보니, "이 부분도 중요한데, 저건?? 저 내용이 빠지면 앞뒤가 안맞잖아~!!" 옥신 각신 혼자 고민 아닌 고민도 좀 하게 되었던 문서 였다. 한마디로, 분량 조절 실패... 다 다뤄 볼 심산이었는데, 욕심이었다....(—.—);;;;;

어쨌든, 개인적으로 머릿속에서만 맴돌고 정리되지 않던 내용들에 대해서, 가볍지만 전체적으로 살펴본 것으로 족하다.

"시작이 반이다"라는 말이 있듯이, 이제부터 시작이다라는 생각으로, 다음 "HowTO Qemu-Monitor" 문서도 틈틈히 정리해 봐야 겠다.

언제가 될지 모르겠으나, 그 때 다시.....

감사합니다.

2012/06/21