백업 및 비상상황

victory

백업이란?

데이터가 손상되거나 손실될 경우를 대비해 저장하는 데이터의 사본
→ 당연히 백업해뒀다고 해서 원본을 지우지 않는다.
ex) 사진이 날라갈까봐 무서워서 아이클라우드에 밤마다 동기화 시켜두는것

백업이란?

데이터가 손상되거나 손실될 경우를 대비해 저장하는 데이터의 사본
→ 당연히 백업해뒀다고 해서 원본을 지우지 않는다.
ex) 사진이 날라갈까봐 무서워서 아이클라우드에 밤마다 동기화 시켜두는것

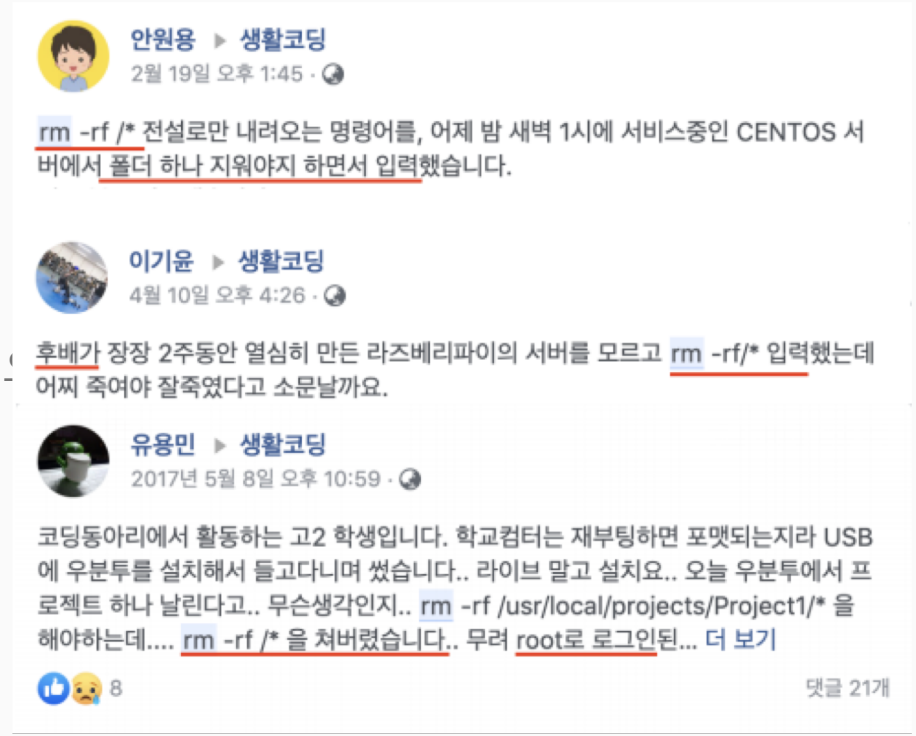
아카이브랑 어떻게 다르지?
아카이브란, 참고용으로 생성한 데이터 사본
→ 꼭 그런건 아니지만, 아카이브를 만든 후에는 원본 데이터를 지울때가 많음

	백업	아카이브
목적	어떤 것을 정상적인 상태로 되돌리는것 → Restore	먼 과거의 데이터에서 일부 데이터를 찾는 것 → Retrieve
무엇을	파일이나 서버, 디비 등	관련 데이터 모음 (원본과 동일한 서버나 동일한 형식으로 저장되어 있지 않을 가능성이 높음)

그러면 백업을 왜할까?

- 하드웨어에 물리적인 문제가 생길수도
- 실수로 파일을 날릴수도 (rm -rf)
- 해커가 서버접근을 막을수도
- 랜섬웨어 같은거에 걸릴수도

- 게임을 악용하는 사례가 생겨서 어느 시점
  롤백을 해야할때

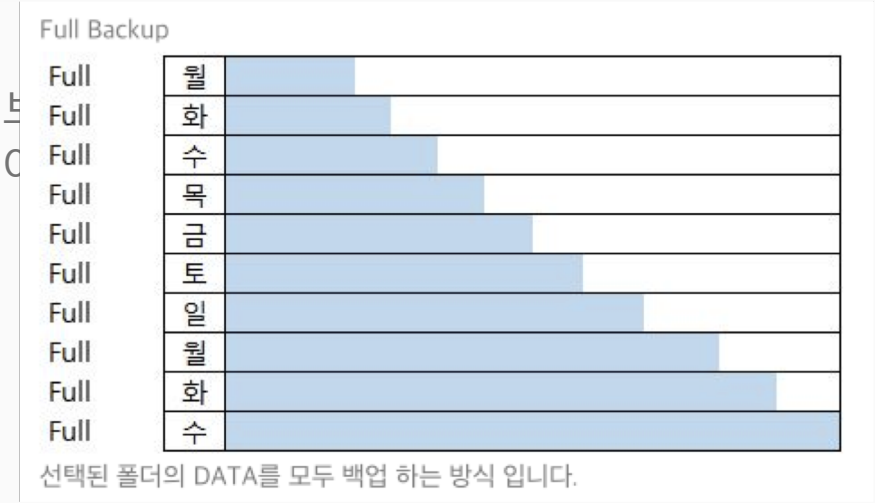


백업의 종류

백업종류	백업되는 데이터	백업시간	복구시간	저장공간
Full	전부	가장 느림	빠름	많이 필요
Incremental	이전 백업 이후로 추가/수정된 내용	빠름	적당	가장 적게 필요
Differential	최근 Full백업 이후로 추가/수정된 내용	적당	빠름	적당
Mirror	이전 백업 이후로 추가/수정된 내용	가장 빠름	가장 빠름	가장 많이 필요

백업의 종류: Full backup

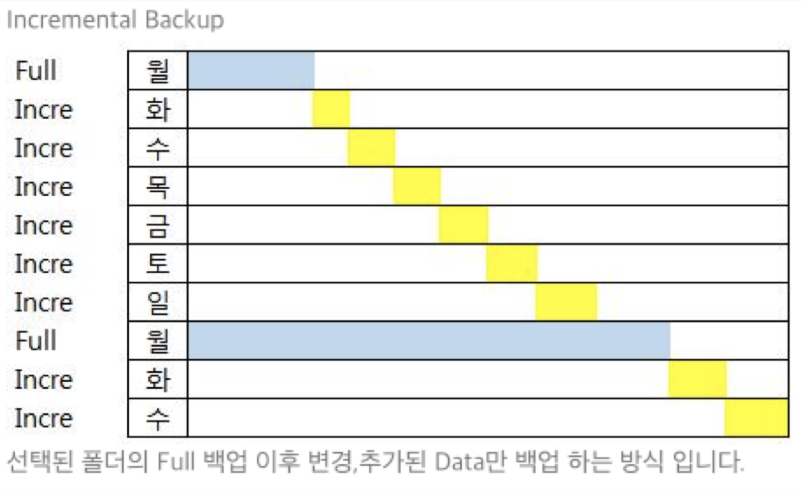
- 할때마다 모든 데이터 백업
- 전부 압축해서 (비유하자면) 하나의 zip 파일로 ㅂ
- 해킹당하면 모든 정보가 보이니까 강력한 보안이
- 그래도 간단하게 복구가 가능!
- 변경사항 유무에도 관계가 없음!



백업종류	백업되는 데이터	백업시간	복구시간	저장공간
Full	전부	가장 느림	빠름	많이 필요
Incremental	이전 백업 이후로 추가/수정된 내용	빠름	적당	가장 적게 필요
Differential	최근 Full백업 이후로 추가/수정된 내용	적당	빠름	적당
Mirror	이전 백업 이후로 추가/수정된 내용	가장 빠름	가장 빠름	가장 많이 필요

백업의 종류: Incremental backup

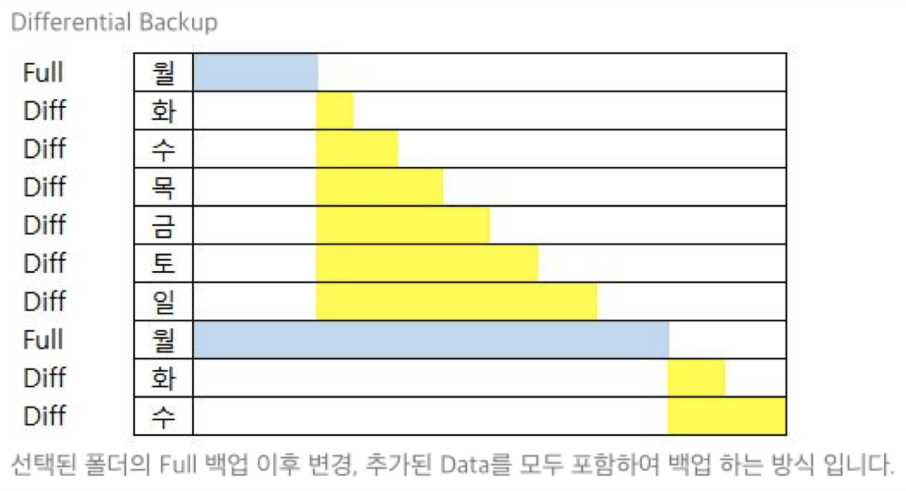
- 적은 내용을 백업하니까 백업시간이 빠르다
- 그렇지만 Full backup에 지나치게 의존적임
- 복구할때 각각 다 Restore해야해서 느림



백업종류	백업되는 데이터	백업시간	복구시간	저장공간
Full	전부	가장 느림	빠름	많이 필요
Incremental	이전 백업 이후로 추가/수정된 내용	빠름	적당	가장 적게 필요
Differential	최근 Full백업 이후로 추가/수정된 내용	적당	빠름	적당
Mirror	이전 백업 이후로 추가/수정된 내용	가장 빠름	가장 빠름	가장 많이 필요

백업의 종류: Differential backup

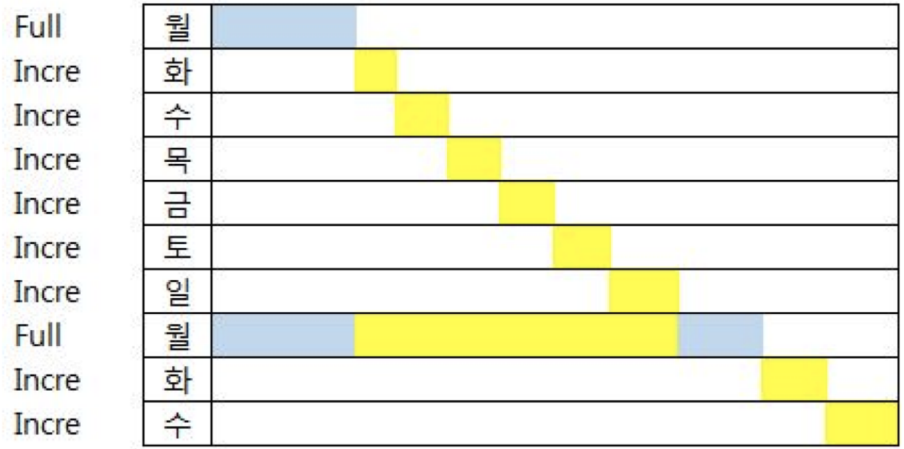
- Full backup 이후 바뀐 내용만 백업



백업종류	백업되는 데이터	백업시간	복구시간	저장공간
Full	전부	가장 느림	빠름	많이 필요
Incremental	이전 백업 이후로 추가/수정된 내용	빠름	적당	가장 적게 필요
Differential	최근 Full백업 이후로 추가/수정된 내용	적당	빠름	적당
Mirror	이전 백업 이후로 추가/수정된 내용	가장 빠름	가장 빠름	가장 많이 필요

실제로는?

적당히 섞어서 함



선택된 폴더의 Full 백업 이후 변경, 추가된 Data를 Incremental Backup 형식으로 저장 후 두번째 Full 백업 작업시 중간에 모아 둔 Incremental bakcup을 이용하여 Full Backup으로 재생성 하는 방식.

백업하기: tar

1. 백업용 디렉토리 만들기

```
$ cd /
$ sudo mkdir backups
$ cd backups
```

--directory 로 백업하고 싶은 디렉토리를 지정
--exclude로 제외할 항목 지정
-c: tar로 묶는다
-v: 압축과정을 터미널에 출력한다
-p: 파일권한을 함께 저장한다
-f 파일 이름을 지정한다

1. 하나의 tar 파일로 모은다

```
$ sudo tar -cvpf /backups/fullbackup.tar --directory=/ --
exclude=proc --exclude=sys --exclude=dev/pts --exclude=backups .
```

백업하기: tar

※ 주의

- ‘tar로 압축해!’ 라는 말을 자주 쓰지만, 보통 말하는 ‘압축'이 아님
  → ‘데이터의 크기를 줄이기 위한 파일 압축'을 수행하지 않음
- 위에서 말했듯이 여러 파일을 하나의 파일로 묶는 용도로 사용될 뿐
- 대신 tar로 하나로 합쳐진 파일을 gzip또는 bzip2 방식을 이용해서 압축(용량을 작게)할 수 있는데
  이런경우는 파일 확장자가 .tar.gz 또는 .tar.bz2 등임

어쩌다 tar를 자주 쓰게 되었을까?
: tar로 묶여지기 전 파일들의 속성, 심볼릭 링크, 디렉토리 구조 등을 그대로 가져갈 수 있어서

백업하기: tar

※ 명령어

tar 사용 예	명령어 옵션
현재 디렉토리의 모든 파일과 디렉토리를 tar로 묶기	`tar cvf T.tar *`
대상 디렉토리를 포함한 모든 파일과 디렉토리를 tar로 묶기	`tar cvf T.tar [PATH]`
파일을 지정하여 tar 아카이브로 묶기	`tar cvf T.tar [FILE_1] [FILE_2]`
tar 아카이브를 현재 디렉토리에 풀기	`tar xvf T.tar`
tar 아카이브를 지정된 디렉토리에 풀기	`tar xvf T.tar -C [PATH]`
tar 아카이브의 내용 확인하기	`tar tvf T.tar`
현재 디렉토리를 tar로 묶고 gzip으로 압축하기	`tar zcvf T.tar.gz *`
gzip으로 압축된 tar 아카이브를 현재 디렉토리에 풀기	`tar zxvf T.tar.gz`
현재 디렉토리를 tar로 묶고 bzip2로 압축하기	`tar jcvf T.tar.bz2 *`
bzip2로 압축된 tar 아카이브를 현재 디렉토리에 풀기	`tar jxvf T.tar.bz2`
tar 아카이브 묶거나 풀 때 파일 별 진행 여부 확인하기	`tar cvfw T.tar *`

원격 서버에 백업하기

나의 소중한 old서버 자료들~
날라갈지 모르니까 /backup 에 tar gzip 으로 매일매일 full backup~
이히히히 역시 나는 똑똑한 휠~

원격 서버에 백업하기

나의 소중한 old서버 자료들~
날라갈지 모르니까 /backup 에 tar gzip 으로 매일매일 full backup~
이히히히 역시 나는 똑똑한 휠~

만약 서버가 통째로 날라간다면?

원격 서버에 백업하기

나의 소중한 old서버 자료들~
날라갈지 모르니까 /backup 에 tar gzip 으로 매일매일 full backup~
이히히히 역시 나는 똑똑한 휠~

만약 서버가 통째로 날라간다면?
→ 다른 서버로 백업이 필요하다.

원격 서버에 백업하기: rsync

remote sync의 줄임말

```
$ rsync {options} {source} {destination}
```

주요 옵션

-v: --verbose 자세한 정보 출력
-q: --quiet 정보 출력 억제
-a: --archive 아카이브 (-rlptgoD 옵션을 다 준것과 동일)
-r: --recursive 디렉토리 재귀
-l: --links 심볼릭 링크 복사
-z: --compress 압축
--progress: 동기화 진행률 표시

원격 서버에 백업하기: rsync

참고: 소스에 / 가 있고 없음의 차이

```
ubuntu@ip-172-31-45-109:~$ tree test
test
└── a.txt

0 directories, 1 file
ubuntu@ip-172-31-45-109:~$ ls
test
ubuntu@ip-172-31-45-109:~$ tree test
test
└── a.txt

0 directories, 1 file
ubuntu@ip-172-31-45-109:~$ rsync -a test test_dest
ubuntu@ip-172-31-45-109:~$ tree test_dest
test_dest
└── test
    └── a.txt

1 directory, 1 file
ubuntu@ip-172-31-45-109:~$ rsync -a test/ test_dest_wanted
ubuntu@ip-172-31-45-109:~$ tree test_dest_wanted/
test_dest_wanted/
└── a.txt

0 directories, 1 file
```

원격 서버에 백업하기: rsync

pem file을 이용해서 ec2 instance에 rsync 하기

```
$ rsync -e 'ssh -i ~/.ssh/victory.pem' -avz test/ ubuntu@3.34.97.20:~/test_rsync
```

원격 서버에 백업하기: rsync

pem file을 이용해서 ec2 instance에 rsync 하기

```
$ rsync -e 'ssh -i ~/.ssh/victory.pem' -avz test/ ubuntu@3.34.97.20:~/test_rsync
```

다른 방법도 있음
→ 비밀번호 (pem file) 을 매번 넣어주지 말고,
private key, public key를 통해 instance가 믿을 수 있다는걸 확정지어 주자

원격 서버에 백업하기: rsync

pem file을 이용해서 ec2 instance에 rsync 하기

```
$ rsync -e 'ssh -i ~/.ssh/victory.pem' -avz test/ ubuntu@3.34.97.20:~/test_rsync
```

다른 방법도 있음
→ 비밀번호 (pem file) 을 매번 넣어주지 말고,
private key, public key를 통해 instance가 믿을 수 있다는걸 확정지어 주자 (Secure Shell)

어떻게?

1. public key 생성
   ```
   $ ssh-keygen -t rsa
   ```
2. public key를 instance에 넣어줌
   ```
   $ scp -i ~/.ssh/victory.pem ~/.ssh/id_rsa.pub ubuntu@3.34.97.20:~/.ssh/authorized_keys
   ```
3. 그러면 내 local의 private key와 instance의 public key가 한쌍인지 확인되고
4. 그 key를 이용해 암호화 복호화하며 데이터를 주고 받는다.
   ```
   rsync -avz test/ ubuntu@3.34.97.20:~/test_rsync
   ```

주기적으로 백업하기

'주기적'이면 뭐다? cron 이다.
backup.sh 같은거 하나 잘 만들어서 매일 밤 자정 실행시키거나 하면 된다.

backup.sh 예시:

```
#!/bin/sh
FROM_BACKUP_DIR="/path/to/source/dir"
TO_BACKUP_DIR="/path/to/backup/dir"
TODAY=`date +%Y%m%d`
BACKUP_FILENAME=backup_${TODAY}.tar.gz
tar -zcvpf ${TO_BACKUP_DIR}/${BACKUP_FILENAME} ${FROM_BACKUP_DIR}
rsync -av --delete ${TO_BACKUP_DIR}/${BACKUP_FILENAME} {사용자}@{원격서버}:{destination}
```

cron에 추가 (매일 밤 10시 쉘스크립트 실행)

```
$ crontab -e
0 22 * * * /home/ubuntu/backup.sh
```

비상상황

비상상황을 만드는 세가지가 뭘까

1. 소프트웨어
2. 하드웨어
3. 사람

Software Failure

- 내부
  - 파일 시스템 에러
  - 장치 설정 에러
  - 부팅 에러
  - 기타 프로그램 에러
  - 커널 패닉 (블루스크린)
  - 메모리 오버플로우
  - etc

- 외부
  - 해킹
  - 악성 코드, 바이러스
  - 접속자 폭주
  - etc

Hardware Failure

- 내부
  - 랜선 고장
  - 전원장치 고장
  - 파워 이상
  - 냉각 이상
  - 특정 부품/부분 망가짐
  - etc

- 외부
  - 먼지
  - 물 쏟음
  - 케이블 절단
  - etc

사람의 문제

- 내부
  - 관리자 실수
  - 악의적 내부자
  - 잘못된 입력 혹은 오타
  - etc
- 외부
  - 도둑
  - 해커
  - 악의적 사용자
  - etc

파일 시스템 명령어

df (디스크 남은 용량 보기)

```
ubuntu@ip-172-31-45-109:~$ df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
udev           devtmpfs  488M     0  488M   0% /dev
tmpfs          tmpfs     100M  4.5M   95M   5% /run
/dev/xvda1     ext4      7.7G  1.6G  6.2G  20% /
tmpfs          tmpfs     496M     0  496M   0% /dev/shm
tmpfs          tmpfs     5.0M     0  5.0M   0% /run/lock
tmpfs          tmpfs     496M     0  496M   0% /sys/fs/cgroup
/dev/loop0     squashfs   98M   98M     0 100% /snap/core/9289
/dev/loop1     squashfs   18M   18M     0 100% /snap/amazon-ssm-agent/1566
/dev/loop2     squashfs   97M   97M     0 100% /snap/core/9665
/dev/loop3     squashfs   29M   29M     0 100% /snap/amazon-ssm-agent/2012
tmpfs          tmpfs     100M     0  100M   0% /run/user/1000
```

파일 시스템 명령어

du (내가 쓴 용량 보기)

```
ubuntu@ip-172-31-45-109:~$ du -h
8.0K    ./test_dest_wanted
4.0K    ./.cache
16K     ./.ssh
8.0K    ./test_rsync
8.0K    ./test
8.0K    ./test_dest/test
12K     ./test_dest
84K     .
```

```
ubuntu@ip-172-31-45-109:~$ du -ah
4.0K    ./test_dest_wanted/a.txt
8.0K    ./test_dest_wanted
0       ./.cache/motd.legal-displayed
4.0K    ./.cache
4.0K    ./.ssh/authorized_keys
4.0K    ./.ssh/rsa.pem
4.0K    ./.ssh/rsa.pem.pub
16K     ./.ssh
0       ./.sudo_as_admin_successful
4.0K    ./.bash_history
4.0K    ./.bash_logout
4.0K    ./.viminfo
4.0K    ./.bashrc
4.0K    ./test_rsync/rsync.txt
8.0K    ./test_rsync
4.0K    ./.profile
4.0K    ./test/a.txt
8.0K    ./test
4.0K    ./.selected_editor
4.0K    ./test_dest/test/a.txt
8.0K    ./test_dest/test
12K     ./test_dest
84K     .
```

파일시스템은 언제 잘못되나

1) 시스템이 갑자기 shutdown (서버가 중지됐을 때), 전원이 나가버렸을 때
   → 근데 요새는 journaling file system 덕분에 문제가 잘 안생기긴함
2) 두 프로세스가 서로를 인지하지 못한채 구조나 내용을 변경했을 때 (lock이 제대로 안먹혔을 때)
3) 시스템 디버거가 잘못 쓰였을 때

   등등

파일 시스템 명령어

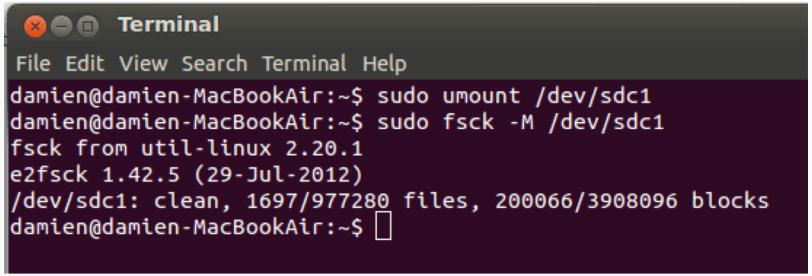
fsck (리눅스 파일 시스템 점검)

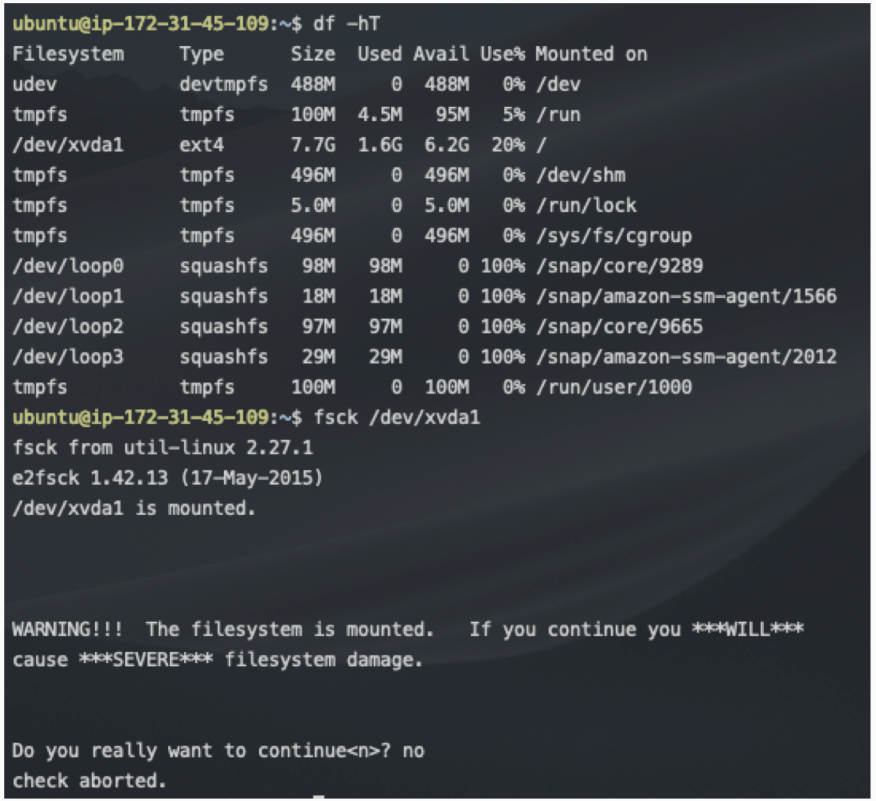
```
$ man fsck.ext4
```

일단 점검하고자 하는 디스크를 unmount 시키고 (umount 명령어 이용)

```
$ fsck {option} {장치명}
```

-v: 자세한 출력
-f: 파일시스템 이상 유무와 상관없이 강제 체크





파일 시스템 명령어

fsck (리눅스 파일 시스템 점검)

`$ fsck {option} {장치명}`

-v: 자세한 출력

-f: 파일시스템 이상 유무와 상관없이 강제 체크

fsck Available options

Fsck command needs to be run with superuser privileges or **root**. You can use it with different arguments. Their usage depend on your specific case. Below you will see some of the more important options:

- `-A` – Used for checking all filesystems. The list is taken from `/etc/fstab`.
- `-C` – Show progress bar.
- `-l` – Locks the device to guarantee no other program will try to use the partition during the check.
- `-M` – Do not check mounted filesystems.
- `-N` – Only show what would be done – no actual changes are made.
- `-P` – If you want to check filesystems in parallel, including root.
- `-R` – Do not check root filesystem. This is useful only with '`-A`'.
- `-r` – Provide statistics for each device that is being checked.
- `-T` – Does not show the title.
- `-t` – Exclusively specify the filesystem types to be checked. Types can be comma separated list.
- `-V` – Provide description what is being done.

정전

카이스트는 정전이 잦다.

정전 발생시 당황하지 말자.

서버실에는 UPS (무정전 전원 공급장치)가 있어서 어느정도 시간동안은 서버가 멈추지 않고 작동 가능

서버를 종료하기 전에 SPARCS로 서비스 중단 공고(페이스북페이지, 트위터, 이메일)를 하고, 서버들을 차례로 종료시켜두자.

서버실 과열

서버실 온도는 여름철 33~37도, 겨울철에도 26도에 이르기까지 좀 높은 편.
서버실에 온도계가 있으니 그걸로 확인.

에어컨을 늘 확인하자. 가끔 멀티탭에 문제가 있는 경우가 있으니 멀티탭도 확인.
에어컨이 고장났으면 바로 시설팀에 연락하고 수리 요청
서버실 문을 열고, 선풍기 등을 이용해 열을 빼내자. (단, 보안에 신경써야함)

비상시, 중요하지 않은 서버들을 종료하여 최대한 발열량을 줄이자.

해킹

주로 명령어를 못쓰게 하거나, 중요 파일 삭제/변조, 로그 삭제/변조, 비밀번호 에러가 뜨도록 변조

미리 백업해 놓은 시스템 코어로 대체 후 작업

데미지가 클 수도 있는 대응책:

- 해킹 감지 즉시 시스템 싱글 유저 모드 부팅
- 랜선 등 기타 네트워크 강제 차단
- 로그 보존을 위한 하드의 즉각적인 unmount

해킹 대비하기

- 안쓰는 port 닫기
  - ec2 instance 에서 '위치무관'을 막 쓰지 맙시다.
- 의심 가는 process는 죽이기
- 모의 해킹 등으로 보안 점검
- 주기적으로 프로그램 update
- root로 로그인하고 자리 비우지 않기
- 비밀번호 적어놓기 or 숨기지 않은 파일에 중요한 것들 적어놓기 **절대 X**
- 서버실 물리적 보안

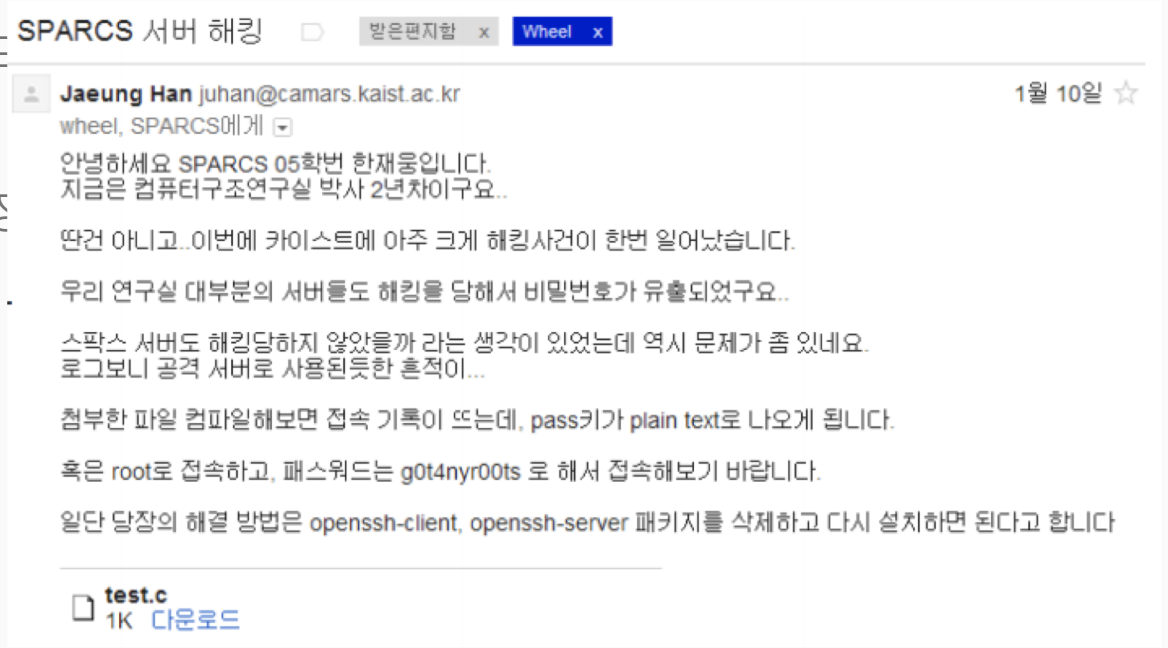
2012 KAIST/SPARCS 해킹 사건

2012년 1월 10일

KAIST 연구실들이 해킹당함. 스팍스도

/var/run/sshd.sync/에
사용자들 비밀번호가 plain text로 저장

root 아이디에 g0t4nyr00ts 를 치면
접속이 된다.



SPARCS 서버 해킹 받은편지함 x Wheel x

Jaeung Han juhan@camars.kaist.ac.kr 1월 10일

wheel, SPARCS에게

안녕하세요 SPARCS 05학번 한재웅입니다.
지금은 컴퓨터구조연구실 박사 2년차이구요..

딴건 아니고..이번에 카이스트에 아주 크게 해킹사건이 한번 일어났습니다.

우리 연구실 대부분의 서버들도 해킹을 당해서 비밀번호가 유출되었구요..

스팍스 서버도 해킹당하지 않았을까 라는 생각이 있었는데 역시 문제가 좀 있네요.
로그보니 공격 서버로 사용된듯한 흔적이...

첨부한 파일 컴파일해보면 접속 기록이 뜨는데, pass키가 plain text로 나오게 됩니다.

혹은 root로 접속하고, 패스워드는 g0t4nyr00ts 로 해서 접속해보기 바랍니다.

일단 당장의 해결 방법은 openssh-client, openssh-server 패키지를 삭제하고 다시 설치하면 된다고 합니다

test.c
1K 다운로드

2012 KAIST/SPARCS 해킹 사건

<elaborate> 아.....

<elaborate> 패닉패닉패닉

* pcpenpal (pcpenpal@125.7.192.138) 님께서 대화방 #sparcs에 참여하셨습니다.

<YUI> lol

<YUI> elaborate panic ? :D

<elaborate> YUI, 안녕하세요, 누구신가요?

<YUI> talk english please

<elaborate> ?? I am sorry but who are you?

<YUI> your biggest nightmare

<elaborate> ;;;;;

<elaborate> What do you do?

<YUI> I hack

<softdie_> aha...

<softdie_> You attacked sparcs?

<YUI> I didn't attack anything

<YUI> I only gained root

<YUI> ah k

<elaborate> Well,,, I would like to know how you know thw password

<elaborate> I just want to know the process

<YUI> [illegible]

<elaborate> F__k you!

<YUI> elaborate you need to study more

YUI란 아이디의 해커가 스팍스 IRC에 접속

2012 KAIST/SPARCS 해킹 사건

〈YUI〉 elaborate you need to study more

〈softdie_〉 First time

〈YUI〉 ever heard of zeroboard

〈softdie_〉 foolish Web application is the problem.

〈softdie_〉 use zeroboard?

〈YUI〉 i used an exploit of 2005

〈YUI〉 sshd backdoor :)

〈YUI〉 not only that

〈YUI〉 ssh client is backdoored aswell

해킹 루트 분석:

zeroboard4 취약점으로 root 획득 → 획득한 서버에 백도어 설치 → 획득한 서버의 sshd를 변조 → ssh 사용자의 id와 pw를 획득 → 다른 서버에 같은 작업 반복

2012 KAIST/SPARCS 해킹 사건

Zeroboard?

ZeroBoard Remote Command Execution (Exploit, preg_replace) 31 May. 2005

Summary

Zeroboard is one of the most popular PHP web boards in Korea. The following exploit uses a recently reported vulnerability in Zeroboard to inject and execute PHP script on vulnerable system.

제로보드 **preg_replace** 보안 취약점

강 성훈, 2005년 1월 23일[1] (수정 #2)

- 대상: **제로보드** 4.1 pl2 - 4.1 pl5 및 비공식 패치 버전
- 환경: php4/php5 전 버전, register_globals/magic_quotes_gpc 설정과 무관
- 요약: HTML 태그를 포함해서 글을 쓸 수 있는 권한이 있는 공격자가 임의의 php 코드를 실행할 수 있습니다.
- 상황: 4.1 pl6에서 수정되었습니다.

해킹 루트 분석:

zeroboard4 취약점으로 root 획득 → 획득한 서버에 백도어 설치 → 획득한 서버의 sshd를 변조 → ssh 사용자의 id와 pw를 획득 → 다른 서버에 같은 작업 반복

2012 KAIST/SPARCS 해킹 사건

https://s3.ap-northeast-2.amazonaws.com/sparcs.home/seminars/chocho-20140806_1-0.pdf

KAIST 해킹 사건에 대해 긴급하게 공지합니다. Kuss (ajmbell)

안녕하세요, SPARCS 회장 안재만입니다.
2011년 말과 2012년 초에 카이스트 네트워크에 있는 유닉스 계열 서버에 대한 공격이 있었습니다. 현재 유닉스 계열의 연구실 홈페이지, 개인 컴퓨터, 각종 서버 등 굉장히 많은 서버의 관리자 계정이 탈취 된 것으로 파악되고 있습니다. root가 탈취된 서버에서는 ssh 데몬이 변조되어 /var/run/sshd.sync에 접속한 사용자의 기록이 남겨져 있으며, 해커가 지정한 특정 아이디와 비밀번호에 대해서 접속이 가능합니다. 몇몇 주요 서버에 대해서는 또 다른 rootkit이 발견되고 있습니다.

따라서 서버 관리자 분들 께서는 반드시 /var/run/sshd.sync 파일이 있나 확인해주시고, 만약 있을 경우 서버를 완전히 새로 세팅하는 것을 강력하게 권고드립니다. 만약 서버를 완전히 세팅하는 것이 불가능 할 경우, 파일 삭제 후 openssh-server, openssh-client에 대한 재설치를 해주시기 바랍니다. debsums 등을 이용하여 각종 알려진 패키지에 대한 변조 검사 역시 해주시기 바랍니다. 또한, /__sh에 숨겨져 저장된 또다른 rootkit 역시 발견되었으므로, debugfs로 반드시 루트 디렉토리 하에 __sh 디렉토리가 있는지 확인해 주시기 바랍니다. 만약 있는 경우, __sh디렉토리 안의 모든 파일을 삭제하시고, 삭제 한 후에는 제대로 부팅이 되지 않을 수 있으므로 live usb (http://www.sysresccd.org/Main_Page) 를 준비하시어 /etc/ld.so.preload 파일의 삭제 및 fsck 역시 수행해 주시기 바랍니다.

지금까지 밝혀진 rootkit인 이정도이나, 이후 새로운 사실이 밝혀질 때 마다 공지하도록 하겠습니다.
추가적인 문의사항이 있을 경우에 wheel@sparcs.org 로 문의 주십시

[2012년 1월 13일 추가]

현재 해킹사건이 어느정도 정리된것으로 보입니다.
첫 서버의 해킹은 제로보드 취약점을 이용하여 이루어 진것으로 보입
http://cosmic.mearie.org/2005/01/zeroboard-4.1-remote-exe
곳에서 수집한 아이디와 비밀번호를 바탕으로 다른 서버를 해킹하는
을하는 홈페이지는 반드시 버전업그레이드를 진행해 주시기 바랍니다
또한, 해커가 아이디와 비밀번호를 가지고 다시 해킹을 시도할 가능성
인을 자주 진행해 주시면 감사하겠습니다.

< 제3회 본엔젤스 매드(MAD, Mobile Application Developer) 컴프

신조 pereven (10개월 1주 전) 아울러, /etc/ssh/sshd_config 파일을 확
시기 바랍니다. 또한 제로보드 4를 사용하
므로 다른 게시판 사용을 적극적으로 권유

bit OS재설치 및 mir 패키지 업그레이드 안내

받은편지함 x

정창제 changjej@gmail.com 1월 12일

wheel, sparcs에게

안녕하십니까? 휠장 rodumani 정창제 입니다.

동아리 개발서버 bit의 OS를 재설치하는 작업과 mir의 패키지 업그레이드를 진행하였습니다.

bit의 경우 이번 해킹사태로 인하여 피해를 입었던 만큼, OS를 새로 설치하여 잠재적 위험을 없애고자 하였습니다.

Dutch Cyberpolice Arrested 17 year old boy suspected of hacking KPN and KAIST

Submitted by siavash on Mon, 03/26/2012 - 17:53

The High Tech Crime Team of the National Crime Squad in Barendrecht city close To Rotterdam arrested 17 year old boy who is suspected of hacking into KPN(The leading telecommunications and ICT service provider in The Netherlands) on 16 January 2012. The boy was arrested last Tuesday at home when he was online on the Internet.Dutch National Office Press Release

The investigators of the National Police (National Police Agency) has seized an encrypted computer, two laptops and data carriers, such as external hard drives, USB sticks and DVDs.

The computer system of KPN is probably hacked because there was a vulnerability. The hacker gained a few hundred servers the highest access rights. These servers were used for internet services and storage of (customer) information.

며 설치된 모든 패키지는 최신판으로 업그레

되어있으며, 이는 일부 외부로 비밀번호가 유해 또다시 피해를 입을 수 있기 때문입니다.

원들에 대한 조치는 조만간 다시 알려드리겠습it, mir 모두 접속을 하실 수 없습니다.

적 피해를 입지는 않았지만 보안의 우려가 있급하게 시행하였습니다.

docker, aws..
(feat. 우리는 발전했다.)

앗 뉴아라 백엔드 서버가 그냥 다 안돼! 뒤졌어!

1. 평소 중요한 파일을 S3에 저장한다.
   S3에 new-ara-api의 docker-compose.yml 파일이 저장되어있다.
2. 새로운 인스턴스 생성 + 거기에 docker-compose.yml 파일만 복사
3. docker-compose up -d 하면 뽀로롱 생성
4. 그 후 찬찬히 왜 뒤졌나 디버깅해보자

어떻게 가능하냐고?

상태 확인에 실패한 인스턴스 문제 해결

https://docs.aws.amazon.com/ko_kr/AWSEC2/latest/UserGuide/TroubleshootingInstances.html

커널패닉으로 뒤져버린 인스턴스라면..

https://www.rootusers.com/how-to-repair-an-aws-ec2-instance-without-console/

볼륨에 대한 스냅샷 생성이 가능

앗 뉴아라 백엔드 서버가 그냥 다 안돼! 뒤졌어!

```yaml
# Check S3 'sparcs-newara' bucket for the complete file
version: '3'

services:
  api:
    container_name: newara
    image: newara:dev
    tty: false
    ports:
      - 9000:9000
    environment:
      - DJANGO_ENV=development
      - SECRET_KEY=
      - AWS_BUCKET_NAME=
      - AWS_BUCKET_NAME_STATIC=
      - AWS_ACCESS_KEY_ID=
      - AWS_SECRET_ACCESS_KEY=
      - SSO_CLIENT_ID=
      - SSO_SECRET_KEY=
      - NEWARA_DB_HOST=
      - NEWARA_DB_PORT=
      - NEWARA_DB_USER=
      - NEWARA_DB_PASSWORD=
      - NEWARA_DB_NAME=
    entrypoint: /newara/www/entrypoint.sh

```

어떻게 가능하냐고?

1. 도커 이미지로 바로 띄워요 (이미지는 도커허브나 AWS ECR에 올라가 있음)
2. 파일관리를 AWS S3로 해요
3. DB는 AWS RDS에 있어요

앗 뉴아라 백엔드 서버가 그냥 다 안돼! 뒤졌어!

```yaml
# Check S3 'sparcs-newara' bucket for the complete file
version: '3'

services:
  api:
    container_name: newara
    image: newara:dev
    tty: false
    ports:
      - 9000:9000
    environment:
      - DJANGO_ENV=development
      - SECRET_KEY=
      - AWS_BUCKET_NAME=
      - AWS_BUCKET_NAME_STATIC=
      - AWS_ACCESS_KEY_ID=
      - AWS_SECRET_ACCESS_KEY=
      - SSO_CLIENT_ID=
      - SSO_SECRET_KEY=
      - NEWARA_DB_HOST=
      - NEWARA_DB_PORT=
      - NEWARA_DB_USER=
      - NEWARA_DB_PASSWORD=
      - NEWARA_DB_NAME=
    entrypoint: /newara/www/entrypoint.sh

```

어떻게 가능하냐고?

1. 도커 이미지로 바로 띄워요 (이미지는 도커허브나 AWS ECR에 올라가 있음)
2. 파일관리를 AWS S3로 해요
3. DB는 AWS RDS에 있어요

아니 위의 S3나 RDS가 잘못될수도 있잖아요!!! 빼애액

AWS라고 다 믿을수있는거 아니잖아요오오오

→ 맞아요

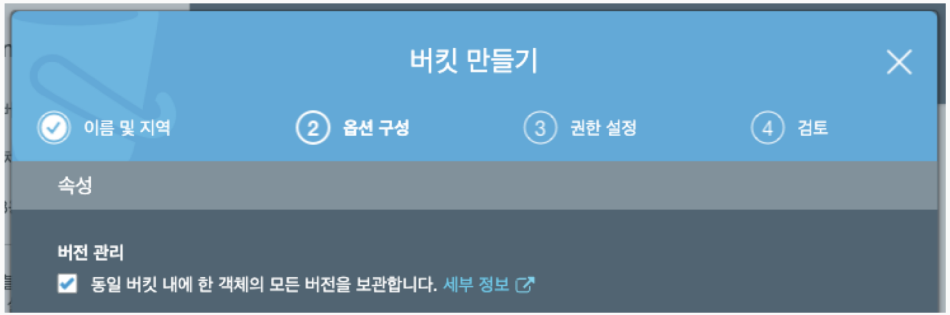
→ 그래도 우리보다는 믿을만해요..

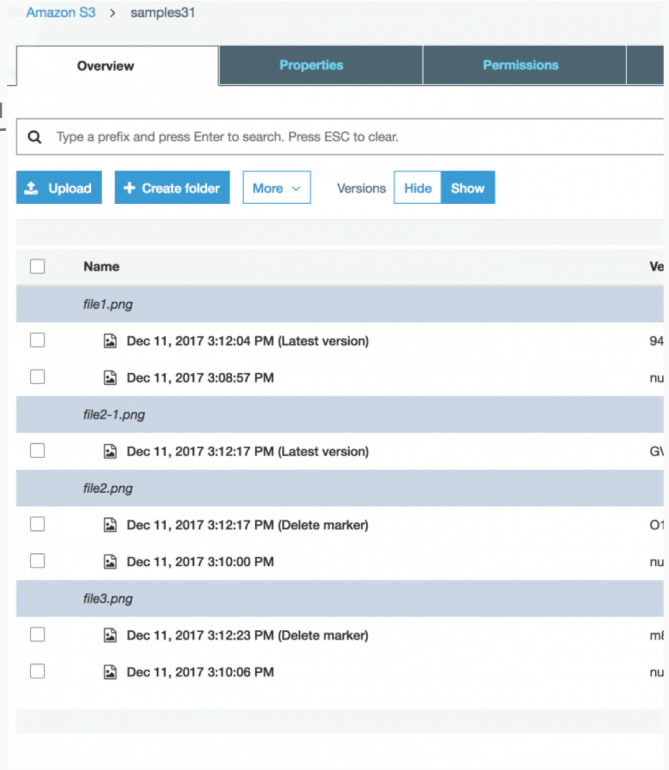
→ 그리고 다양한 백업을 지원해준답니다!

AWS에서 우리 서비스를 최대한 안전하게 하려면? - S3

S3

- S3의 버전을 보관하자
- S3의 내용도 주기적으로 S3 glacier로 보내자
  (glacier는 보통 아카이빙 목적으로 쓰임 - 싸고 느리
  * 참고: https://bluese05.tistory.com/39





AWS에서 우리 서비스를 최대한 안전하게 하려면? - RDS

RDS

- 스냅샷을 생성하고 관리하자
  * snapshot 생성:
  https://docs.aws.amazon.com/ko_kr/AmazonRDS/latest/UserGuide/USER_CreateSnapshot.html
  * snapshot 이용해서 복원:
  https://docs.aws.amazon.com/ko_kr/AmazonRDS/latest/UserGuide/USER_RestoreFromSnapshot.html

서버복구라는건 그렇게 간단한 일일까?

앗 뉴아라 백엔드 서버가 그냥 다 안돼! 뒤졌어!

1. 평소 중요한 파일을 S3에 저장한다.
   S3에 new-ara-api의 docker-compose.yml 파일이 저장되어있다.
2. 새로운 인스턴스 생성 + 거기에 docker-compose.yml 파일만 복사
3. docker-compose up -d 하면 뽀로롱 생성

어떻게 가능하냐고?

서버복구라는건 그렇게 간단한 일일까?

앗 뉴아라 백엔드 서버가 그냥 다 안돼! 뒤졌어!

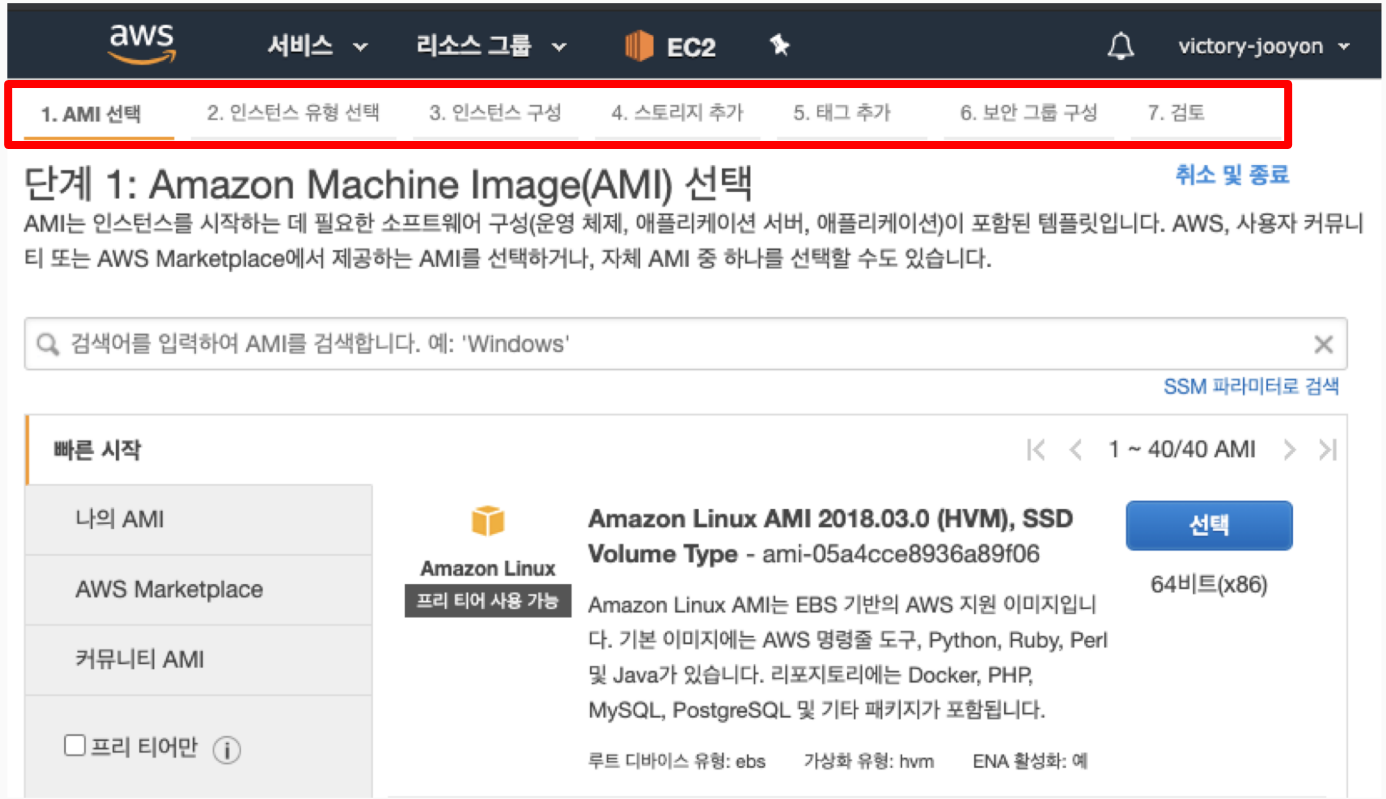
1. 평소 중요한 파일을 S3에 저장한다.
   S3에 new-ara-api의 docker-compose.yml 파일이 저장되어있다.
2. 새로운 인스턴스 생성 + 거기에 docker-compose.yml 파일만 복사
3. docker-compose up -d 하면 뽀로롱 생성

어떻게 가능하냐고?

그래요 인생은 그렇게 쉽지 않아요...

Infrastructure as Code 의 필요성

단계가 무려 7개! 어떻게 똑같이 생성할건가요?



Infrastructure as Code 의 필요성

단계가 무려 7개! 어떻게 똑같이 생성할건가요?

당신의 선택은?

1. 하하 나는 기억력이 좋다고! 슈슈슉 똑같이 눌러주면 되지!

Infrastructure as Code 의 필요성

단계가 무려 7개! 어떻게 똑같이 생성할건가요?

당신의 선택은?

1. 하하 나는 기억력이 좋다고! 슈슈슉 똑같이 눌러주면 되지!
   → 시간이 급박해요. 서비스가 죽은거라고요. 누르는데 시간걸리잖아요.
   → 그래요, 당신 기억력도 좋고 손도 엄청 빠르다고 합시다. 너님이 휴가가면? 다른팀원이 살려야하요?

Infrastructure as Code 의 필요성

단계가 무려 7개! 어떻게 똑같이 생성할건가요?

당신의 선택은?

1. 하하 나는 기억력이 좋다고! 슈슈슉 똑같이 눌러주면 되지!
   → 시간이 급박해요. 서비스가 죽은거라고요. 누르는데 시간걸리잖아요.
   → 그래요, 당신 기억력도 좋고 손도 엄청 빠르다고 합시다. 너님이 휴가가면? 다른팀원이 살려야해요?

2. 후후 우리팀은 노션을 쓰지! 거기에 다 어떻게 눌러야하는지 기록해놨어!

Infrastructure as Code 의 필요성

단계가 무려 7개! 어떻게 똑같이 생성할건가요?

당신의 선택은?

1. 하하 나는 기억력이 좋다고! 슈슈슉 똑같이 눌러주면 되지!
   → 시간이 급박해요. 서비스가 죽은거라고요. 누르는데 시간걸리잖아요.
   → 그래요, 당신 기억력도 좋고 손도 엄청 빠르다고 합시다. 너님이 휴가가면? 다른팀원이 살려야해요?

2. 후후 우리팀은 노션을 쓰지! 거기에 다 어떻게 눌러야하는지 기록해놨어!
   → 오늘 새벽 3시에 팀원B가 사실 인스턴스 크기 늘려놓음. 그리고 노션 아직 업데이트 안함..

Infrastructure as Code: Terraform

꼭 이런 이유가 아니더라도..

같은 '설정'을 가진 mysql DB, VPN, server 등등을 만들고 싶을 수도 있고,
언제 어떻게 변경을 했는지 기록하고 싶을 수도 있고,

뭐 그런 갖가지 이유가 있어서.. Infrastructure as Code 의 등장!

인프라도 코드로 관리한다!
깃헙처럼 버전관리도 된다!

그 코드 마지막 버전에서 예를 들어 make up! 같은 명령어 한줄이면
알아서 실제 지금 떠있는 서버랑 코드에 있는데 없는 서버 등등을 비교해서
뽀로롱 하고 설정 똑같이 빈걸 띄우는거지~

Infrastructure as Code: Terraform

한번의 백업이
하룻밤 삽질, 10번의 복구작업,
100번의 후회와 해고를 막습니다...

by. 10학번 sillo 선배님