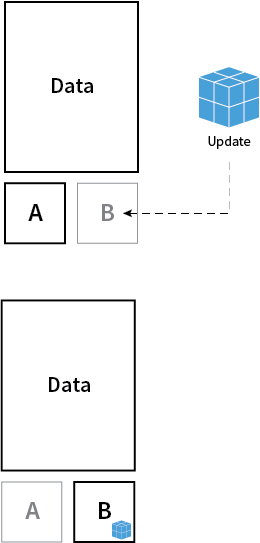
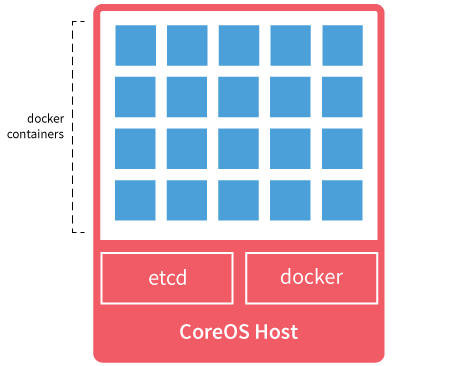
[](http://1.bp.blogspot.com/-awKPsNESkPs/U7c93vHjbSI/AAAAAAAAFGc/5XNoij9nW-E/s1600/coreos-wordmark-horiz-color.png)

CoreOS는 Docker구동에 특화된 리눅스OS를 목표로 탄생한 OS이며 다른 리눅스들과 구분되는 CoreOS의 특징은 크게 다음의 세가지 이다.  
  
최소화된OS : CoreOS는 114M의 메모리만을 사용하며 CoreOS측에 의하면 일반적인 리눅스에 비해 40%적은 메모리 사용량이라 한다.  
  
빠른OS업데이트 : CoreOS는 OS용으로 2개의 부트파티션을 지니며, 이를 이용해 업데이트 작업을 빠르고 안정적으로 수행한다.(이를 CoreUpdate라 부른다.)  
  
이게 무슨 이야기 인가 하면, 아래 그림과 같이 A파티션에서 동작하는 CoreOS는 B파티션에 동일한 내용을 지니고 있다. OS업데이트는 B파티션에서 수행함으로서 현재 운영중인A파티션의 실행에는 영향이 없고, 업데이트가 끝나면 리부트 후 바로 업데이트가 적용된 B파티션 OS로 스위칭 되므로 매우 빠르고 안정적인 OS업데이트가 가능해진다.

[](http://4.bp.blogspot.com/-rrXjVGckDSs/U7dBTH83gXI/AAAAAAAAFG0/b-QSQclF5Nk/s1600/update-diagram.png)

이러한 일련의 OS업데이트 작업은 만약 업데이트에 문제가 발생했을경우 A파티션이 남아 있으므로 손쉬운 롤백이 가능하다는 장점이 있있으며 조작 또한 웹콘솔상에서 원격 제어가  가능하다.  CoreUpdate는 상용 서포트에서만 사용 가능한 기능이다.  
  
Docker구동에 최적화된 아키텍처 채용: CoreOS는 잠시후에 소개할 etcd나 웹인터페이스를 이용한 모니터링 기능등 대규모 클러스터링에서 안정적으로 Docker를 서포트 하는것에 특화된 OS이다.

[](http://2.bp.blogspot.com/-R_FCPYK8ERw/U7c_9CnABaI/AAAAAAAAFGo/TbTiIPgerns/s1600/coreos-host-diagram.png)

|  |
| --- |
| <http://2.bp.blogspot.com/-QDxqY-9Pr2w/U7dDUnpg0aI/AAAAAAAAFHA/HoXmlE7yplE/s1600/Three-Tier-Webapp.png> |
| 컨테이너 로드벨런싱 |

|  |
| --- |
| <http://4.bp.blogspot.com/-05ptm4m3oSY/U7dDaVYZRqI/AAAAAAAAFHI/xcp84rDKKPs/s1600/GroupList-HiDPI.png> |
| 요즘 대세인 미려한 UI의 Web관리콘솔.  CoreUpdate도 이 콘솔을 이용해 수행이 가능하다. 개인적으론 커맨드라인 인터페이스를 선호하지만  요즘 개발자들은 이런게 없으면 어플리케이션이 고급지지 못하다고 생각하는듯 싶다. |

**etcd란?**

etcd는 go언어와 [Raft프레임워크](https://speakerdeck.com/benbjohnson/raft-the-understandable-distributed-consensus-protocol) 이용해 작성된 오픈소스 key-value 저장소로 대규모 Docker 클러스터링에 있어서 컨테이너들을 유기적으로 연동시키고 억세스하기 위한 세련된 아키텍처를 제공한다.  
  
[etcd Overview](http://coreos.com/using-coreos/etcd/)  
  
좀 더 설명하자면 OS에 배당된 IP어드레스에 비해 탑재된 컨테이너들의 수는 엄청나게 많은데 이러한 컨터이너들에 접근하기 위해서는 IP어드레스 이외의 효율적인 어드레싱 수단을 필요로 하게 된다. etcd는 이러한 어드레싱을 HTTP/JSON을 이용해 구현하고 있으며 빠른 성능과 암호화 제공등으로 현재 Docker사용자들에게 주목받고 있다.

|  |
| --- |
| <http://3.bp.blogspot.com/-b8Xp1WdtWag/U7dJtmJ7m9I/AAAAAAAAFHg/vTBwcUqgyuY/s1600/docker-dag.png> |
| Docker컨테이너 실제 구성예. 효율적인 어드레싱 수단이 필요한 이유이다. |

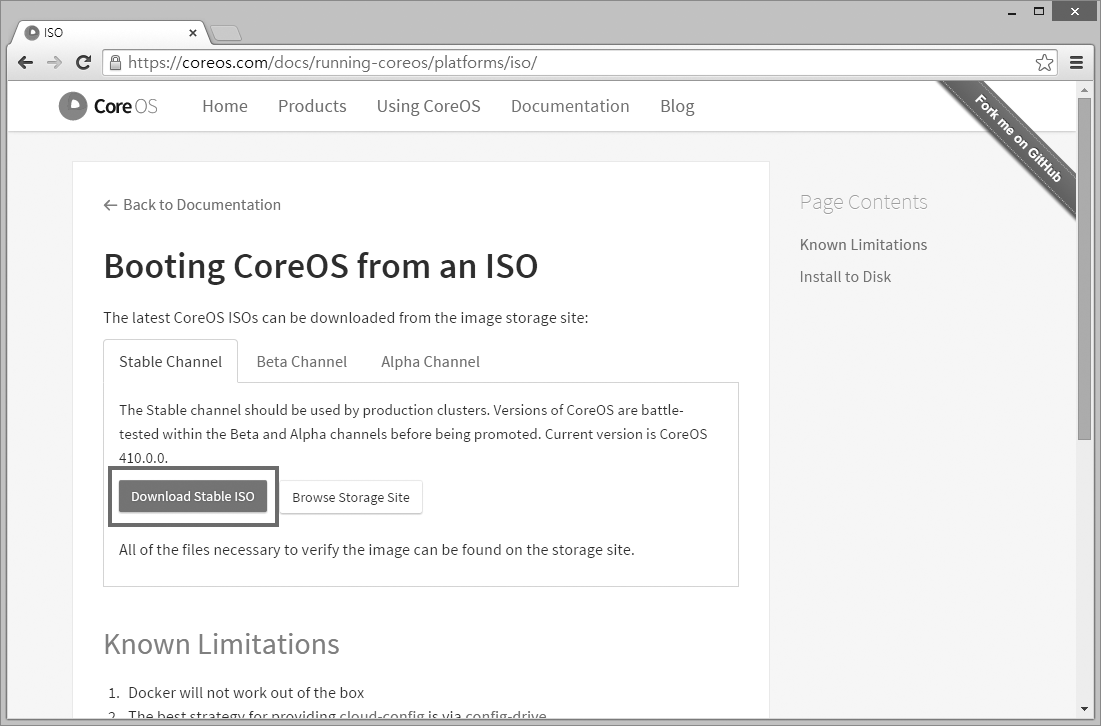
**지원하는 플랫폼**

직접 물리서버에 설치도 가능하고, Vagrant,Amazon EC2, Azure, QEMU/KVM, VMware그리고 OpenStack에 이르기까지 요즘 인기있는 가상화/클라우드 플랫폼은 충실하게 지원하고 있다.   
  
특히, docker를 이용한 대규모 클러스터링 구현은 vagrant에 의존하지 않고는 무척 고된작업이 되어버릴 것 이다. 지금부터 docker를 이용한 클러스터링에 관심을 가지고 사용해 보려는 유저는 이 CoreOS와 함께 vagrant를 같이 익혀두면 크게 도움이 될 것이다.

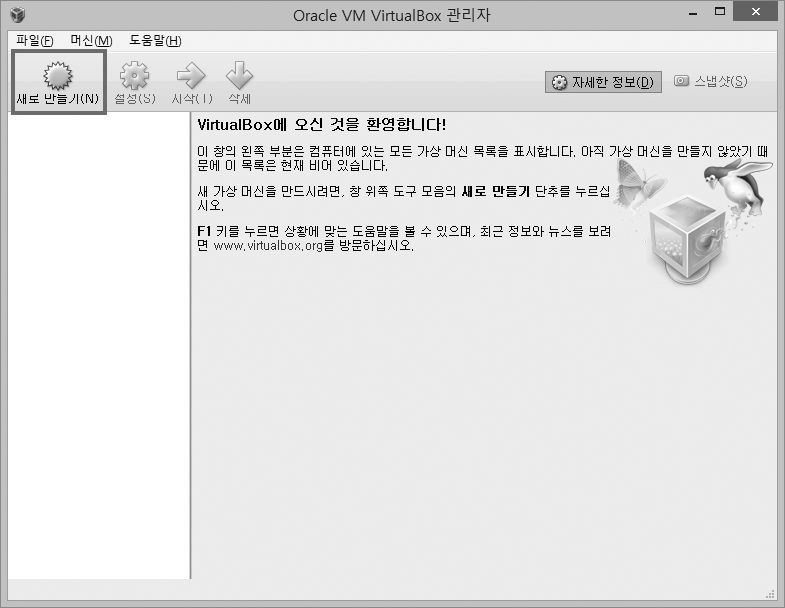
**<coreos 설치>**

먼저 <https://coreos.com/docs/running-coreos/platforms/iso/>에 접속한 뒤 Download Stable ISO 버튼을 클릭하여 ISO 파일을 다운로드 합니다.

* Stable Channel: 많은 테스트를 거친 안정 버전입니다.
* Beta Channel: 알파 채널을 테스트한 뒤 등록된 버전입니다.
* Alpha Channel: 가장 최신 버전입니다. Docker, etcd, fleet 모두 최신 버전이 포함됩니다.

  
**그림 15-3** coreos.com에서 ISO 파일 다운로드

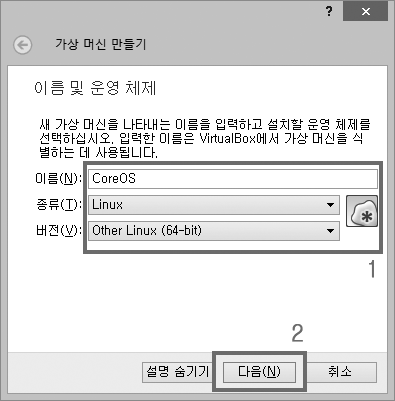
이제 VirtualBox를 실행하고 위쪽 새로 만들기(N) 버튼을 클릭합니다.

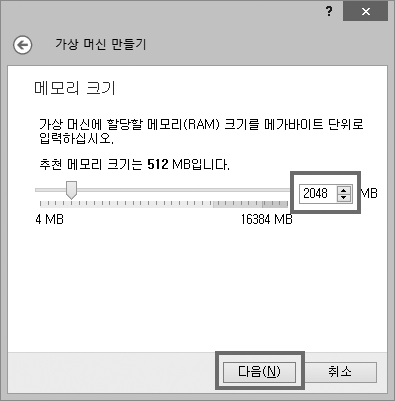
  
**그림 15-4** VirtualBox

VirtualBox 가상 머신을 생성합니다(그림 15-5).

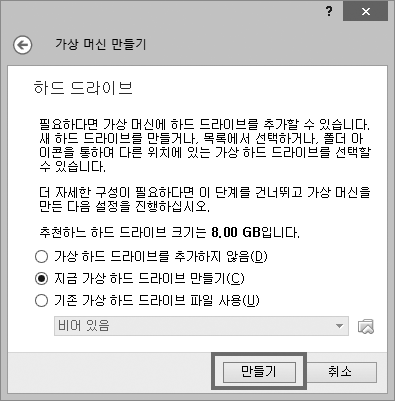
* 이름: 가상 머신 이름입니다. CoreOS를 입력합니다.
* 종류: 가상 머신에 설치될 운영체제 종류입니다. Linux를 선택합니다.
* 버전: 가상 머신에 설치될 운영체제의 버전입니다. Other Linux (64-bit)를 선택합니다.

설정이 완료되었으면 다음(N) 버튼을 클릭합니다.

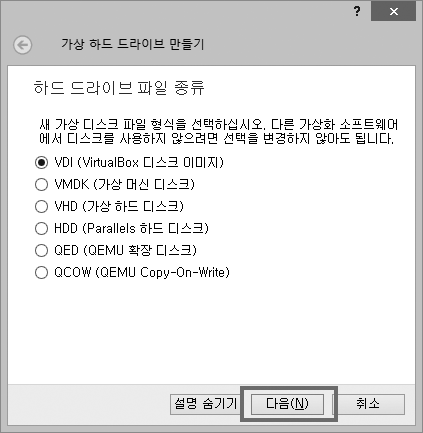
  
**그림 15-5** VirtualBox 가상 머신 이름 및 운영체제 설정

가상 머신의 메모리 크기를 설정합니다. 저는 2048 MB를 설정하였습니다. 각자 알아서 적절한 크기로 설정하고 다음(N) 버튼을 클릭합니다(그림 15-6).  
  
**그림 15-6** VirtualBox 가상 머신 메모리 크기 설정

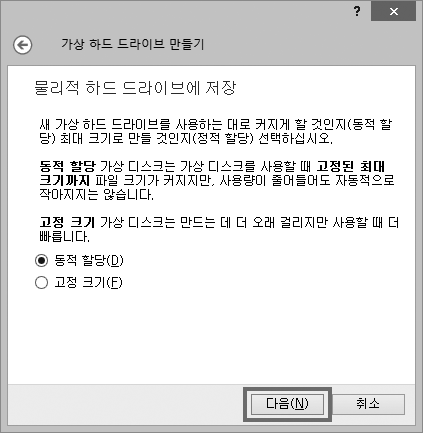
지금 가상 하드 드라이브 만들기(C)를 선택하고 만들기 버튼을 클릭합니다(그림 15-7).

  
**그림 15-7** VirtualBox 하드 드라이브 생성

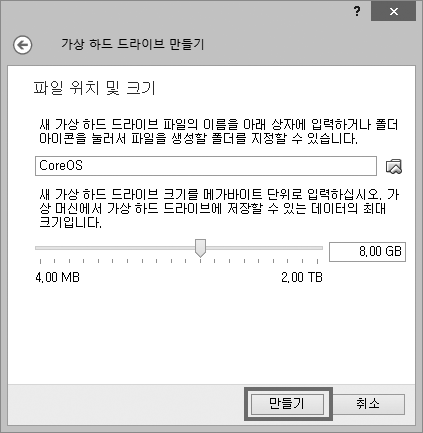
기본 값 그대로 VDI(VirtualBox 디스크 이미지)를 선택하고 다음(N) 버튼을 클릭합니다(그림 15-8).

  
**그림 15-8** VirtualBox 하드 드라이브 파일 종류 선택

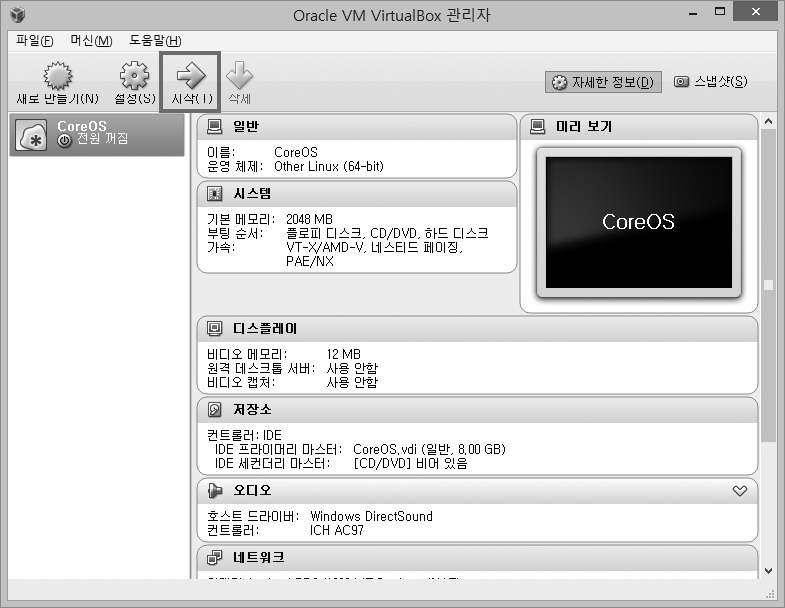
VirtualBox 하드 드라이브 파일의 저장 방식을 선택합니다. 기본 값 그대로 동적 할당(D)을 선택하고 다음(N) 버튼을 클릭합니다(그림 15-9).

  
**그림 15-9** VirtualBox 하드 드라이브 파일 저장 방식 선택

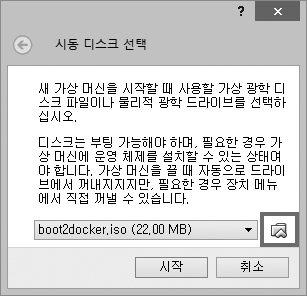
VirtualBox 하드 드라이브 파일의 위치와 크기를 설정합니다. 여기서는 기본 값 그대로 8.00 GB를 설정하였습니다. 각자 알아서 적절한 크기를 설정하고 만들기 버튼을 클릭합니다(그림 15-10).

  
**그림 15-10** VirtualBox 하드 드라이브 파일 위치 및 크기 설정

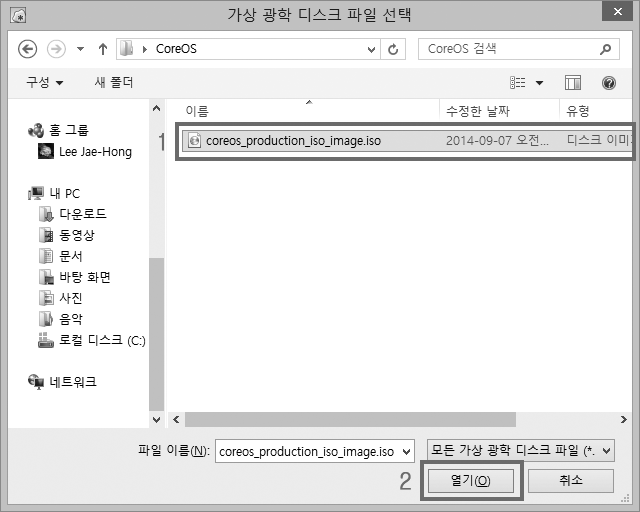
CoreOS용 가상 머신이 생성되었습니다. 위쪽 시작(T) 버튼을 클릭합니다.

  
**그림 15-11** VirtualBox 가상 머신 생성 완료

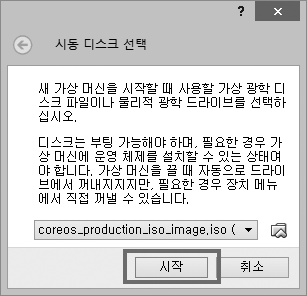
가상 머신을 실행하면 그림 15-12처럼 시동 디스크 선택 창이 표시됩니다. 아래쪽 폴더 버튼을 클릭합니다.

  
**그림 15-12** VirtualBox 시동 디스크 선택

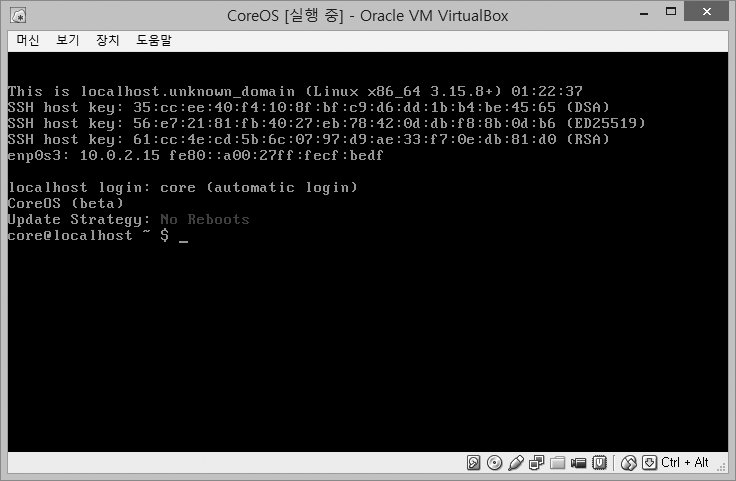
앞에서 다운로드한 coreos\_production\_iso\_image.iso 파일을 선택하고 열기 버튼을 클릭합니다(그림 15-13).

  
**그림 15-13** CoreOS ISO 열기

CoreOS ISO 파일을 열었으면 시작 버튼을 클릭합니다(그림 15-14).

  
**그림 15-14** VirtualBox 시동 디스크 선택

CoreOS ISO(CD) 파일로 부팅했습니다(그림 15-15). CoreOS는 따로 설치 화면이 없고 ISO 파일로 부팅한 coreos-install 명령을 사용하여 하드디스크에 설치하는 방식입니다.

  
**그림 15-15** CoreOS ISO 부팅

간편하게 테스트하기 위해 SSH 키 설정 대신 비밀번호 방식을 사용하겠습니다. VirtualBox 안에서 다음 명령을 입력하여 비밀번호를 생성한 뒤 **cloud-config.yaml**로 저장합니다.

* [dockerbook/Chapter15/cloud-config.yaml](https://github.com/pyrasis/dockerbook/blob/master)

$ openssl passwd -1 > cloud-config.yaml Password: <사용할 비밀번호 입력> Verifying - Password: <사용할 비밀번호 다시 입력>

이제 vim으로 **cloud-config.yaml** 파일을 열고 다음과 같이 작성합니다.

**cloud-config.yaml**

#cloud-config users: - name: exampleuser passwd: $1$qeiJttft8$vs8v2Rnlw1iNkeAFnPwQ00 groups: - sudo - docker

* **#cloud-config**는 반드시 첫째 줄에 입력합니다.
* users: 사용자 설정입니다.
  + name: 사용자 계정 이름입니다. 콘솔이나 SSH로 로그인할 때 이 사용자 이름을 사용하게 됩니다. **exampleuser**를 입력합니다.
  + passwd: 사용자 계정의 비밀번호 해시 값입니다. 앞에서 openssl 명령으로 생성한 비밀번호 해시 값을 그대로 사용합니다.
  + group: 사용자 계정의 그룹 설정입니다. root 권한으로 명령을 실행할 수 있도록 **sudo** 그룹을 설정하고, docker를 sudo 명령없이 실행할 수 있도록 **docker** 그룹을 설정합니다.

**참고**

SSH 키를 설정하려면 다음과 같이 작성합니다.

**cloud-config-ssh.yaml**

#cloud-config users: - name: exampleuser ssh-authorized-keys: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQ<생략> groups: - sudo - docker

만약 GitHub 계정에 SSH 키를 등록해놓았다면 다음과 같이 작성하여 GitHub에서 SSH 키를 가져올 수 있습니다.

**cloud-config-github.yaml**

#cloud-config users: - name: exampleuser coreos-ssh-import-github: <GitHub 계정> groups: - sudo - docker

다음 명령을 실행하여 VirtualBox 가상 머신의 하드디스크(/dev/sda)에 CoreOS를 설치합니다.

$ sudo coreos-install -d /dev/sda -C stable -c cloud-config.yaml

* -d: 설치할 하드디스크 장치입니다. **/dev/sda**를 설정합니다.
* -C: 릴리스 채널입니다. **stable**을 설정합니다. 알파, 베타 채널을 사용하려면 **alpha**, **beta**를 설정합니다.
* -c: 설정 파일 경로입니다. 앞에서 작성한 **cloud-config.yaml**을 설정합니다.

잠시 기다리면 CoreOS 설치가 완료됩니다. 다음과 같이 출력되면 설치가 끝난 것입니다.

Success! CoreOS stable <버전> is installed on /dev/sda

VirtualBox 가상 머신 화면에서 장치 → CD/DVD 장치 → 가상 드라이브에서 디스크 꺼내기를 클릭합니다(그림 15-16).

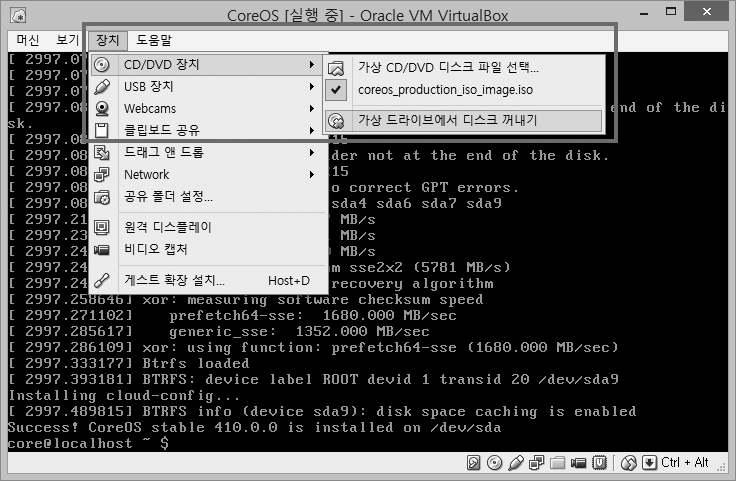
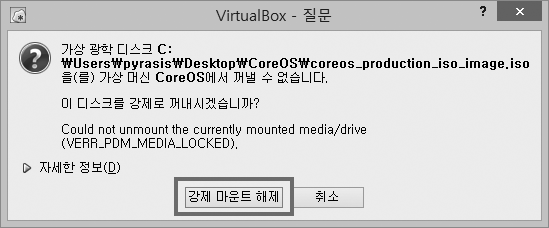
  
**그림 15-16** VirtualBox 가상 머신에서 CoreOS ISO 파일 꺼내기

그림 15-17과 같은 창이 표시되면 강제 마운트 해제 버튼을 클릭합니다.

  
**그림 15-17** VirtualBox 강제 마운트 해제

ISO 파일을 꺼냈으면 다음 명령을 입력하여 가상 머신을 재부팅합니다.

$ sudo reboot

VirtualBox 가상 머신이 재부팅된 뒤에 login에 **exampleuser**, Password에 앞에서 설정한 비밀번호를 입력하면 CoreOS에 로그인이 됩니다. VirtualBox 대신 SSH로 접속해도 됩니다.

docker를 활용하여 서비스를 올려보자.

$ docker images    => 등록된 이미지를 확인(물론, 아직은 아무것도 없다)

$ docker search nginx => 저장소에 어떤 버전들이있는지 검색  
$ docker pull nginx  => 간단히 nginx를 설치한다.  
$ docker run --name ngix -d -p 80:80 -v /home/core:/usr/share/nginx/html:ro -d nginx   => 서비스를 올려보자

$ docker ps -a => 실행 상태 확인

참, 뭔가 화면에 보여야하니까. /home/core/index.htm파일을 만들어보자. body 내용은 "Nginx start"이라고 입력한다.

$ curl http://hostip:80

