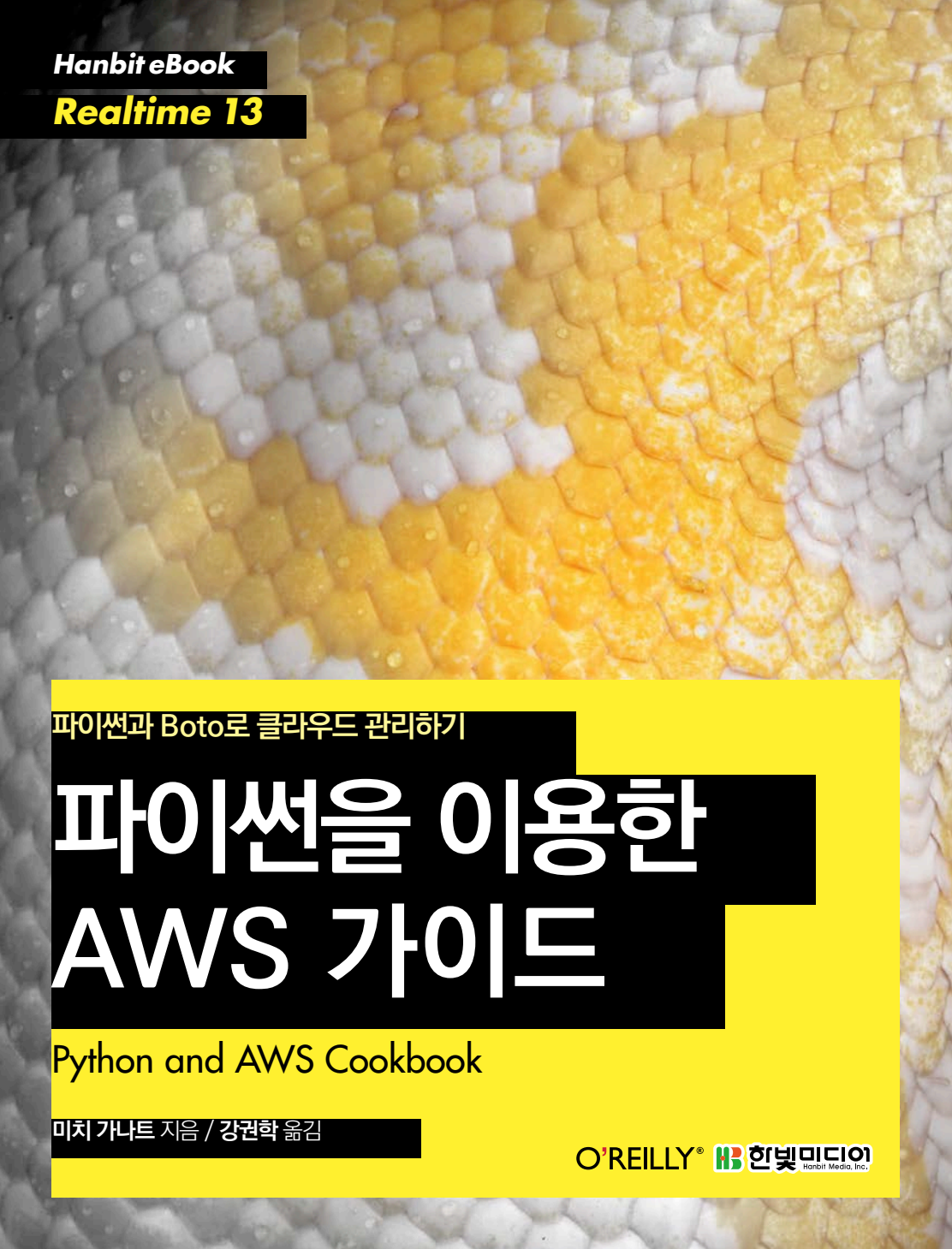

Hanbit eBook
Realtime 13
파이썬과 Boto로 클라우드 관리하기
파이썬을 이용한
AWS 가이드
Python and AWS Cookbook
미치 가나트 지음 / 강권학 옮김
O'REILLY®
한빛미디어
Hanbit Media, Inc.

Managing Your Cloud with Python and Boto

Python & AWS Cookbook

O'REILLY®

Mitch Garnaat

이 도서는 O'REILLY의
Python & AWS Cookbook의
번역서입니다.

파이썬과 Boto로 클라우드 관리하기

파이썬을 이용한 AWS 가이드

파이썬과 Boto로 클라우드 관리하기 **파이썬을 이용한 AWS 가이드**

초판발행 2013년 1월 31일

지은이 미치 가나트 / **옮긴이** 강권학 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-7914-368-3 15560 / **정가** 9,900원

책임편집 배용석 / **기획·편집** 김창수
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 김현미
영업 김형진, 김진불, 조유미 / **마케팅** 박상용, 박주훈, 정민하

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanb.co.kr / **이메일** ask@hanb.co.kr

Published by HANBIT Media, Inc. Printed in Korea
Copyright ⓒ 2013 HANBIT Media, Inc.
Authorized Korean translation of the English edition of *Python and AWS Cookbook, ISBN 9781449305444* ⓒ 2011 Mitch Garnaat. This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.
이 책의 저작권은 오라일리 사와 한빛미디어(주)에 있습니다.
저작권법에 의해 보호를 받는 저작물이므로 무단 복제 및 무단 전재를 금합니다.

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanb.co.kr)로 보내주세요.
한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.지금 하지 않으면 할 수 없는 일이 있습니다.

지은이_ **미치 가나트(Mitch Garnaat)**

유칼립투스 시스템의 소프트웨어 개발자이며, 클라우드 컴퓨팅 리소스를 더욱 쉽게 생성하고 관리하는 도구 개발을 담당하고 있다. 이전에는 제록스 파크Xerox PARC에서 수석 엔지니어로서, 제록스의 웹 기반 문서 관리 시스템인 DocuShare의 공동 개발자이자 선임 개발자였다. 클라우드 컴퓨팅을 위한 오픈소스 파이썬 라이브러리인 boto보토를 창작했다.

옮긴이_ **강권학**

중앙대학교 컴퓨터공학과에서 학사와 석사 학위를 받았다. 국방과학연구소, 퓨쳐시스템, 안철수연구소에서 13년간 개발자, 보안전문가, 프로젝트 관리자로 근무했으며, 2009년 4월 호주 멜번에 iGonagi Pty. Ltd.를 설립하고 아이폰 앱을 개발하고 있다. 『만들면서 배우는 아이폰 게임 프로그래밍』을 공동으로 저술했고, 『Head First iPhone Development』, 『Head First Programming』, 『Head First Python』, 『iPhone Programming 제대로 배우기』, 『iPhone 3D Programming: using OpenGL ES』(이상 한빛미디어)를 번역했다.

저자 서문

2006년 3월 14일에 처음으로 아마존 웹 서비스Amazon Web Services, AWS에 접하게 되었다. 언론을 통해 심플 스토리지 서비스Simple Storage Service, S3라고 부르는 웹 기반 스토리지 서비스를 제공한다는 소식을 들었으며, 아마존이 그런 서비스를 제공한다는 점이 다소 생소하게 느껴졌던 것으로 기억된다. 그렇지만, 가입해 계정을 만들고 문서를 읽기 시작했다.

S3는 매우 신선한 충격이었다. 간단하고도 적절한 가격의 모델, 멋진 RESTful API, 거의 무제한의 스토리지 능력, 이 모든 것이 놀라웠다. 그 당시 생각하기엔, 파이썬 인터페이스만 제공되면 더할 나위 없을 것이라고 생각했다. 그 때부터 boto(https://github.com/boto/boto) 라이브러리의 초기 버전을 구현하기 시작했다. 책에서는 아마존 웹 서비스에 연결하기 위해 boto 라이브러리를 사용한다.

AWS 및 기타 클라우드 서비스와 연동하는 데 파이썬이 최고의 언어라고 생각한다. 파이썬에 기본적으로 설치되는 환상적인 라이브러리, 파이썬 치즈 가게Python Cheese Shop(http://pypi.python.org/pypi)를 통해 설치할 수 있는 수많은 모듈, 요청을 보내고 결과를 바로 확인함으로써 클라우드 서비스를 대화형으로 처리할 수 있는 능력이 결합되어, 애플리케이션을 개발하고 클라우드 기반 구조를 제어할 수 있는 강력하고도 즐거운 개발 환경을 제공한다.

새로운 것을 배우는 최고의 방법은 예제 코드를 많이 보는 것이라고 생각한다. 이 책은 파이썬과 boto를 사용해 EC2Amazon Elastic Compute Cloud와 S3Simple Storage Service에 흔히 발생하는 문제를 해결하는 많은 예제 코드를 보여준다. 이 책이 도움이 되길 바란다!

지은이 **미치 가나트(Mitch Garnaat)**

대상 독자 및 도서 구성

초급 | 초중급 | **중급** | 중고급 | 고급

원격 컴퓨팅과 데이터 저장을 위해 아마존 웹 서비스Amazon Web Service, AWS를 사용할 계획이라면, 애플리케이션을 개발하고 클라우드 기반 기반구조를 관리하는 데 파이썬이 최고의 개발 언어이다. 이 책은 지은이가 개발한 boto 라이브러리와 파이썬 언어로 AWS를 사용하기 위한 20개가 넘는 비법을 제공한다.

EC2Amazon Elastic Compute Cloud 뿐만 아니라 S3Simple Storage Service를 사용하기 위한 자세한 비법을 제공한다. EC2를 사용하면 클라우드 애플리케이션을 설계하고 구현할 수 있다. 각 비법에는 바로 사용할 수 있는 코드가 제공되며, 이 비법이 작동하는 과정을 자세히 설명한다. AWS 및 여러 클라우드 서비스에 boto를 사용할 때 고려할 사항도 설명한다.

예제 파일

예제 파일은 파일은 “http://examples.oreilly.com/0636920020202”에서 받을 수 있습니다.

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook 입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. **eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.**

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내시는 선배, 전문가, 고수분에게는 보다 쉽게 집필하실 기회가 되리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 준비 중이며, 조만간 선보일 예정입니다.

2. **무료로 업데이트되는, 전자책 전용 서비스입니다.**

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정한 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나, 저자(역자)와 독자가 소통하면서 보완되고 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위하여, DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT기기에서 자유롭게 활용하실 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해, 독자 여러분이 언제 어디서 어떤 기기를 사용하시더라도 편리하게 전자책을 보실 수 있도록 하기 위함입니다.

4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

지금 보시는 전자책에 소유권한을 표시한 문구가 없거나 타인의 소유권한을 표시한 문구가 있다면 위법하게 사용하고 계실 가능성이 높습니다. 이 경우 저작권법에 의해 불이익을 받으실 수 있습니다.

다양한 기기에 사용할 수 있습니다. 또한 한빛미디어 사이트에서 구입하신 후에는 횟수에 관계없이 다운받으실 수 있습니다.

한빛미디어 전자책은 인쇄, 검색, 복사하여 붙이기가 가능합니다.

전자책은 오탈자 교정이나 내용의 수정보완이 이뤄지면 업데이트 관련 공지를 이메일로 알려드리며, 구매하신 전자책의 수정본은 무료로 내려받으실 수 있습니다.

이런 특별한 권한은 한빛미디어 사이트에서 구입하신 독자에게만 제공되며, 다른 사람에게 양도나 이전되지 않습니다.

차례

1 | 개요

1.1 파이썬 버전

이 책의 예제는 파이썬 2.7.1을 사용하고 있지만, 2.5.x에서 2.7.x의 모든 버전의 파이썬에서도 정상적으로 작동할 것이다. boto 라이브러리는 아직 파이썬 3.x 버전으로 포팅하고 테스트하지 않았지만, 조만간 포팅될 예정이다.

파이썬 홈페이지(http://python.org)에서는 소스코드 또는 다양한 플랫폼에 맞게 이미 컴파일된 모든 버전의 파이썬을 내려받을 수 있다.

1.2 boto 설치

책의 예제를 실행하려면 boto 2.1 이상 버전을 설치해야 한다. 다음에서는 boto를 설치하는 다양한 방법을 설명한다.

1.2.1 github.com에서 내려받고 설치하기

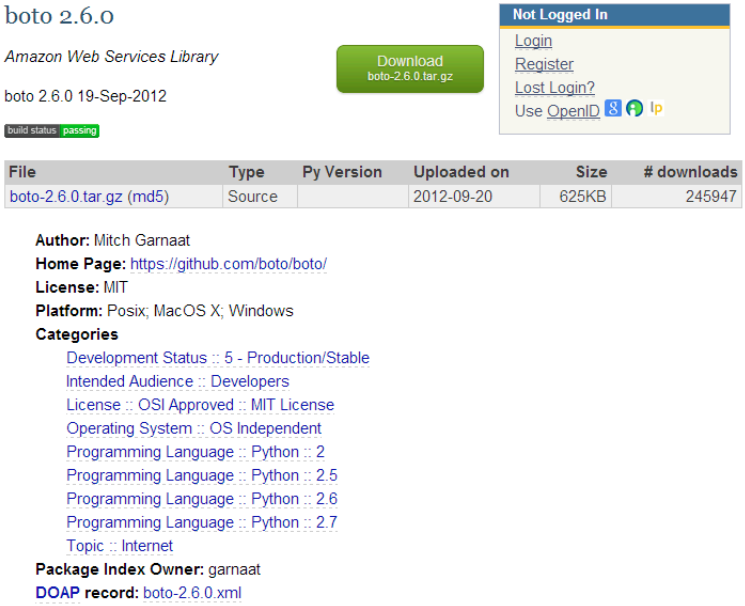
boto 프로젝트는 기트허브github를 소스코드 저장소로 사용한다. 기트허브 저장소를 복제하고, 복제한 배포본으로 boto를 설치할 수 있다. 기트허브 저장소로 설치하면 언제든 최신의 boto 소스코드에 접근할 수 있다. 최신 소스코드를 사용하면 최신 기능을 사용할 수 있지만, 동시에 버그가 있을 수도 있으므로, 기트허브를 사용하는 방법이 자신의 상황에 맞는지 판단해야 한다. 다음의 명령으로 저장소를 복제하고 boto 라이브러리를 설치하면 된다.

```
% git clone https://github.com/boto/boto
% cd boto
% sudo python setup.py install
```

1.2.2 boto를 직접 내려받아 설치하기

파이썬 치즈 가게(http://pypi.python.org/pypi)는 파이썬 패키지의 공식 저장소이다. 치즈 가게(파이파이PyPI라고도 알려져 있다)에 접속해 boto를 검색해 boto 페이지의 화면 아래로 스크롤하면 [그림 1-1]과 같은 페이지를 볼 수 있다.

그림 1-1 치즈 가게 웹사이트의 boto 페이지



boto-2.6.0.tar.gz 링크를 클릭하면 boto 소스코드를 갖고 있는 압축된 타볼Tarball 파일을 내려받는다. 이 파일을 컴퓨터에 저장하고, 다음의 명령을 수행해 소스코드 패키지로 설치하면 된다.

```
% tar xfz boto-2.1.tar.gz
% cd boto-2.1
% sudo python setup.py install
```

1.2.3 easy_install로 boto 설치하기

당연한 이야기겠지만, easy_install 프로그램을 사용하면 파이썬 패키지를 쉽게 검색, 설치, 갱신, 제거할 수 있다. easy_install의 멋진 기능을 사용하려면, 먼저 easy_install을 설치해야 한다. setuptools 패키지(setuptools 패키지에 easy_install이 포함되어 있다)를 설치하는 방법은 "http://pypi.python.org/pypi/setuptools"에서 볼 수 있다. 리눅스 배포판에 따라 다양한 방법으로 setuptools를 설치할 수 있다. 예를 들어 우분투Ubuntu에서는 다음과 같이 설치할 수 있다.

```
% sudo apt-get install python-setuptools
```

페도라Fedora나 센트OSCentOS와 같이 윰yum을 사용하는 배포판에서는 다음과 같이 setuptools를 설치할 수 있다.

```
% sudo yum install python-setuptools
```

일단 여러분의 컴퓨터에 easy_install을 설치한 후에는 다음의 명령으로 간단히 boto를 설치할 수 있다.

```
% sudo easy_install boto
```

1.2.4 pip로 boto 설치하기

파이썬 패키지를 검색, 설치, 갱신, 제거할 수 있는 pip라는 프로그램도 있다. 나의 경험으로는 pip나 setuptools 모두 잘 작동했으므로, 어느 프로그램을 사용할지는 여러분이 직접 조사하고 결정하기를 바란다. 아니면 둘 다 사용해도 좋다! pip를 설치하는 자세한 방법은 "http://pypi.python.org/pypi/pip"에서 볼 수 있

다. 일단 여러분의 컴퓨터에 pip를 설치한 후에는 다음의 명령으로 간단히 boto를 설치할 수 있다.

```
% sudo pip install boto
```

1.2.5 virtualenv로 boto 설치하기

설치를 쉽게 해주는 방법 중 마지막 도구는 사실 설치 도구는 아니며, 격리된 파이썬 가상 환경을 만들어 주는 virtualenv라는 도구다. virtualenv를 사용하면 서로 영향을 주지 않고 독립적으로 실행할 수 있는 가상 환경을 한 컴퓨터 안에 무한히 만들 수 있다. 가상 환경에서 설치된 패키지는 시스템에 설치된 파이썬 패키지나 다른 가상 환경에 설치된 패키지에 전혀 영향을 주지 않는다. 필자의 랩탑 컴퓨터에서 여러 프로젝트를 동시에 진행하면서, 한 프로젝트에서 실수로 변경한 설정이 다른 프로젝트나 시스템 전역 설정에 영향을 미치지 못하게 하는 이 도구는 정말 유용하게 사용되었다. 게다가 virtualenv를 설치해 가상 환경을 만들면, easy_install이나 pip를 따로 설치하지 않고도 바로 사용할 수 있으며, 가상 환경에서는 수퍼 유저나 관리자 권한을 갖고 있지 않아도 소프트웨어를 설치할 수 있다.

자세한 virtualenv 설치 방법은 "http://pypi.python.org/pypi/virtualenv"에서 볼 수 있다. 일단 virtualenv를 설치한 후에는 다음의 명령으로 가상 환경을 설정할 수 있다.

```
% virtualenv paws
```

가상 환경 이름은 어떤 이름이든 사용할 수 있다. 그리고 다음과 같이 가상 환경을 활성화한 후에 boto를 설치할 수 있다.

```
% cd paws
% source bin/activate
% pip install boto
```

1.2.6 paramiko 설치하기

paramiko 패키지는 원격 컴퓨터에 안전하게 연결하기 위한 SSH2 프로토콜을 구현한다. paramiko가 없어도 boto를 사용할 수 있지만, 책에서 설명하는 EC2 비법 중 일부는 paramiko 패키지를 사용하므로 지금 설치하는 편이 좋다. "http://www.lag.net/paramiko/"의 설명을 따라 직접 설치할 수도 있지만, easy_install을 설치했다면 easy_install paramiko 명령으로, pip를 설치했다면 pip install paramiko 명령으로 간단히 설치할 수 있다.

1.2.7 euca2ools 설치하기

여러분이 설치할 마지막 패키지는 euca2ools이다. 이 패키지는 유칼립투스 Eucalyptus 팀에 의해 개발되었으며, AWS가 제공하는 도구와 호환되는 명령행 도구를 제공한다. euca2ools는 파이썬으로 구현되었으며 boto 위에서 실행된다. euca2ools는 클라우드 기반 구조를 관리하는 훌륭한 도구들을 제공할 뿐만 아니라, 공부할 가치가 있는 좋은 예제 코드이기도 하다.

euca2ools는 여러 리눅스 배포판에 패키지로 제공된다. 우분투에서는 sudo apt-get -y euca2ools 명령으로 설치할 수 있다. 윰 기반의 배포판에서는 sudo yum install euca2ools 명령으로 설치하면 된다. 혹은 "https://launchpad.net/euca2ools"에서 최신 소스코드를 내려받거나, "http://open.eucalyptus.com/downloads"에서 소스코드 패키지를 내려받을 수 있다.

1.3 아마존 웹 서비스 시작하기

1.3.1 AWS 계정 만들기

아마존 웹 서비스를 사용하려면 먼저 계정을 만들어야 한다. "http://aws.amazon.com/"에 접속해 [Sign Up Now] 버튼을 클릭한다. 이미 amazon.com에 계정을 갖고 있고 그 계정을 AWS에 연동하려면, amazon.com 계정으로 로그인하면 된다. 원하면 AWS 작업에만 사용할 계정을 따로 만들 수도 있다.

AWS 계정을 만드는 자세한 방법은 RightScale(http://rightscale.com/)이 제공하는 튜토리얼(http://support.rightscale.com/1._Tutorials/01-RightScale/1.5_Sign-up_for_AWS)을 참조한다.

이 때 계정이 최신의 EC2와 S3 서비스를 사용하도록 활성화되었는지 확인한다. 위에 제공한 링크의 튜토리얼은 서비스에 가입하는 과정을 자세히 설명한다.

일단 계정을 만든 후에, 계정에는 다음과 같은 다양한 자격증명이 연결된다.

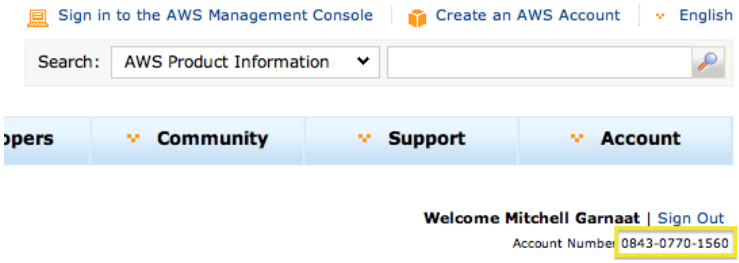
AWS 계정 아이디와 패스워드

AWS 웹 포털과 AWS 관리 콘솔Management Console에 로그인하려면 이 아이디와 패스워드를 사용한다. 아이디는 이메일 주소 형식으로 되어 있다. 아이디와 패스워드가 아래에 설명하는 다른 자격증명에 대한 접근을 제어하므로, 강력한 패스워드를 사용하고 주기적으로 변경하는 편이 좋다.

AWS 계정 번호

이 번호는 AWS 계정에 연결된 고유한 12자리 숫자이다. 여기에서 설명하는 다른 자격증명과 달리 계정 번호는 비밀 정보가 아니다. AWS 포털에 로그인한 후에 웹 페이지 오른쪽 상단을 보면 [그림 1.2]와 같이 계정 번호를 쉽게 볼 수 있다.

그림 1-2 AWS 계정 번호 알아내기



계정 번호는 공개 식별자로서 AWS 안에서 주로 리소스를 공유하기 위해 사용된다. 예를 들어 EC2 안에 AMI(Amazon Machine Image)를 생성한 후에 일반에 공개하지는 않고 어떤 사용자와 공유하고 싶다면, 공유할 사용자의 계정 번호를 AMI에 연결된 사용자 계정 번호 목록에 추가하면 된다. 여기서 주의할 점이 있다. 비록 화면에는 계정 번호가 네자리 숫자가 하이픈으로 연결된 세 개의 그룹으로 나뉘어져 보이지만, API에서 계정 번호를 사용할 때에는 이 하이픈을 제거해야 한다.

접근키 ID(AccessKeyID)와 비밀 접근키(SecretAccessKey)

이 접근 식별자는 AWS의 모든 API를 사용할 때 매우 중요하다. AWS를 요청하는 모든 REST 요청이나 쿼리 API를 호출할 때에는 여러분의 AccessKeyID를 전달해 여러분이 누구인지 알려줘야 한다. 또한 API를 요청할 때 시그너처도 계산해서 포함해야 한다.

시그너처는 요청할 때 보내는 여러 항목들(예를 들어, 타임스탬프, 요청 이름, 인자 등)을 연결해 StringToSign에 저장한 후에, SecretAccessKey를 키로 사용해 StringToSign에 저장된 값의 HMAC(Keyed-Hash Message Authentication Code, '에이치맥'이라고 부른다)을 계산해서 구한다.

AWS가 요청을 받으면, 요청에 들어있는 항목들을 똑같이 연결해 StringToSign에 저장하고, 요청에 들어 있는 AccessKeyID와 연결된 SecretAccessKey를 이용해 시그너처를 계산한다. 계산된 시그너처와 받은 시그너처가 동일하면 요청이 인증된다. 그러나 두 시그너처가 다르면 요청을 거부한다.

계정에 연결된 AccessKeyID는 바꿀 수 없지만, SecretAccessKey는 언제든지 AWS 포털에서 다시 생성할 수 있다. SecretAccessKey는 서버와 클라이언트 간에만 공유된 비밀이며 이 비밀에 기반해 모든 인증 메커니즘이 작동하므로, SecretAccessKey가 노출되었다고 의심되면, 반드시 다시 생성해야 한다.

X.509 인증서

계정과 관련된 또 다른 접근 식별자는 X.509 인증서이다. 직접 자신의 인증서를 제공하거나 AWS가 인증서를 생성하게 할 수 있다. 이 인증서는 SOAP 버전의 AWS API를 사용할 때 요청을 인증하기 위해 사용할 수 있으며, EC2에서 S3 기반의 AMI를 생성할 때에도 사용한다. 본질적으로 AMI 번들을 만들 때 생성된 파일은 계정에 연결된 X.509 인증서를 사용해 암호 알고리즘을 적용해 서명된다. 따라서 AMI 번들을 조작하면, 시그너처가 올바르지 않게 되고 번들이 조작되었음을 쉽게 알 수 있다.

SOAP API를 사용할 때, 보안 관점에서 X.509 인증서는 앞에서 설명한 SecretAccessKey와 마찬가지로 매우 중요하다. 따라서 조심해서 관리해야 한다. 여러분이 SOAP을 사용하지 않더라도, 해커가 사용할 수 있음에 주의한다.

SSH 키

마지막으로 설명할 자격증명은 EC2 인스턴스에 SSH로 접근할 때 사용하는 공개키/개인키 쌍이다. 기본적으로 EC2 인스턴스는 SSH로 접근할 때 공개키 기반 인증만을 허용한다. 여러분이 직접 생성한 AMI 인스턴스에서는 이 정책을

지키는 편이 좋다. SSH 키 쌍은 AWS 콘솔이나 API로 생성할 수 있지만, 기존 키 쌍을 임포트할 수도 있다. 실행되는 인스턴스에 접근해야 하는 사람들은 모두 각자 고유한 키 쌍을 생성하고, 생성된 SSH 키는 엄격히 보호해야 한다.

1.3.2 boto에서 AWS 자격증명 관리하기

일단 AWS 계정을 만들고 자격증명들을 받았다면 boto에게 이 자격증명을 알려줘야 하는데, 다음과 같은 다양한 방법을 사용할 수 있다.

명시적으로 boto에게 자격증명을 전달하기

AWS 서비스에 접근할 때마다 boto를 사용해 해당 서비스에 연결해야 한다. 서비스에 연결할 때, 다음과 같이 접근 키 ID와 비밀 접근 키를 명시적으로 알려주면 boto가 서비스와 통신할 때 이 자격증명들을 사용한다.

```
% python
Python 2.7.1 (r271:86882M, Nov 30 2010, 10:35:34)
[GCC 4.2.1 (Apple Inc. build 5664)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import boto
>>> ec2 = boto.connect_ec2(aws_access_key_id='<접근키 ID>',
aws_secret_ access_key='<비밀 접근키>')
```

대부분의 사람들은 곧 이 방법이 지루하다고 생각하게 된다.

자격증명을 환경 변수에 저장하기

앞 예제와는 달리 연결할 때 boto에게 자격증명을 명시적으로 전달하지 않으면, boto는 AWS_ACCESS_KEY_ID와 AWS_SECRET_ACCESS_KEY 환경 변수가 정의되어 있는지 확인한다. 이 환경 변수가 정의되어 있으면, boto는 환경 변수에 정의된 값을 접근 키와 비밀 키로 사용한다.

boto 설정 파일에 자격증명 저장하기

명시적으로 전달받지 못하고 사용자 환경 변수에도 정의되어 있지 않다면, boto는 설정 파일을 검사한다. boto는 기본적으로 /etc/boto.cfg와 ~/.boto 설정 파일을 참조한다. 다른 파일에 환경 설정을 저장하려면, 설정 파일 경로를 BOTO_CONFIG 환경 변수에 저장해야 한다. boto 환경 설정 파일에 자격증명을 추가하려면, 다음과 같은 섹션을 추가해야 한다.

```
[Credentials]
aws_access_key_id = <접근 키 ID>
aws_secret_access_key = <비밀 접근 키>
```

일단 boto 설정 파일에 자격증명을 저장하면, API를 호출할 때마다 boto가 자동으로 설정 파일에 저장된 정보를 사용한다. AWS 인증서를 처리하는 데 이 방법이 가장 편리한 방법이다. 그러나 AWS 자격증명을 boto 설정 파일에 저장할 때에는, 오직 여러분만이 이 설정 파일에 접근할 수 있도록 접근 권한을 설정해야 한다.

1.3.3 간단한 테스트

이제 boto를 설치하고, AWS 계정을 만들고, 자격증명을 환경 변수나 boto 설정 파일에 저장하는 준비 과정을 마쳤다. 더 진행하기 전에 제대로 작동하는지 다음과 같이 확인한다.

```
% python
Python 2.7.1 (r271:86882M, Nov 30 2010, 10:35:34)
[GCC 4.2.1 (Apple Inc. build 5664)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import boto
>>> ec2 = boto.connect_ec2()
```

```
>>> ec2.get_all_zones()
[Zone:us-east-1a, Zone:us-east-1b, Zone:us-east-1c, Zone:us-east-1d]
>>>
```

앞의 예제에 나온 간단한 테스트를 실행해 비슷하게 실행되면, 이 책에서 설명하는 비법을 실행할 준비가 된 것이다. 그렇지 않다면, 앞에서 설명한 단계를 다시 한 번 확인한다. 진행하다 막히는 부분이 있다면, boto 사용자 그룹(http://groups.google.com/group/boto-users)에 질문하고 도움을 받을 수 있다.

1.4 유칼립투스와 함께 boto 사용하기

1.4.1 유칼립투스 개요

"http://open.eucalyptus.com/learn/what-is-eucalyptus"는 다음과 같이 유칼립투스Eucalyptus에 대해 설명하고 있다.

> 유칼립투스는 회사가 갖고 있는 기존 IT 기반 구조의 도구를 바꾸거나 특별한 하드웨어를 추가할 필요 없이 프라이빗 클라우드Private cloud를 만들 수 있게 한다. 유칼립투스는 IT 자원을 서비스Infrastructure as a Service, IaaS로 제공하는 프라이빗 클라우드를 구현하며, 아마존 EC2와 아마존 S3와 호환되는 API로 접근할 수 있다. 이 호환성 덕분에 유칼립투스 클라우드는 퍼블릭 클라우드Public cloud로부터 컴퓨팅 리소스를 가져올 수 있게 하며, 하이브리드 클라우드Hybrid cloud로 변환할 수 있다. 그리고 유칼립투스에서는 업계 표준인 EC2와 S3를 따르는 많은 도구와 애플리케이션과 호환된다.

간단히 말하면, 유칼립투스를 사용해 AWS와 호환되는 고유한 소형 클라우드(또는 충분한 하드웨어를 갖고 있다면 대형 클라우드)를 만들 수 있게 해준다. 이 책에서 설명하는 AWS를 위한 거의 모든 비법은 유칼립투스에서도 작동하므로, 어느 정도 하드웨어를 갖고 있다면 모든 비법을 여러분의 안전하고 편안한 네트워크 안에서 실험해볼 수 있다.

1.4.2 유칼립투스 설치하기

유칼립투스를 설치하고 싶다면 유칼립투스 패스트스타트Eucalyptus FastStart를 사용하는 편이 좋다(http://open.eucalyptus.com/try/faststart). 패스트스타트를 사용하면 번거롭지 않고 빠르게 유칼립투스를 설치하고 실행할 수 있다.

1.4.3 유칼립투스 커뮤니티 클라우드 사용하기

유칼립투스는 소프트웨어를 설치하지 않고도 사용할 수 있다. 유칼립투스는 유칼립투스 커뮤니티 클라우드Eucalyptus Community Cloud라는 샌드박스에 놓인 가상 환경도 제공한다. 이 가상 환경에서 유칼립투스 클라우드 소프트웨어를 시험해볼 수 있다. 자세한 정보는 "http://open.eucalyptus.com/try/community-cloud"를 참조한다.

1.4.4 boto에서 유칼립투스 인증서 관리하기

일단 유칼립투스 서비스를 사용할 수 있게 준비해 놓으면, boto로 시스템에 쉽게 접근할 수 있다. 먼저 유칼립투스 서버 이름과 자격증명을 boto 설정 파일에 다음과 같이 추가한다.

```
[Credentials]
euca_access_key_id = <유칼립투스 접근 키>
euca_secret_access_key = <유칼립투스 비밀 접근 키>

[Boto]
eucalyptus_host = "<유칼립투스 CLC의 도메인명이나 IP 주소>"
walrus_host = "<월러스의 도메인명이나 IP 주소>"
```

boto 설정 파일을 수정한 후에는, 다음과 같이 유칼립투스EC2 호환 서비스와 월러스Walrus, S3 호환 서비스에 연결할 수 있다.

```
% python
>>> import boto
>>> euca = boto.connect_euca()
>>> walrus = boto.connect_walrus()
```

1.5 구글 클라우드 스토리지와 boto 사용하기

1.5.1 구글 클라우드 스토리지 개요

개발자용 구글 클라우드 스토리지Google Cloud Storage for Developers (http://code.google.com/apis/storage/)는 구글의 기반 구조에 여러분의 데이터를 저장하고 접근하기 위해 사용하는 RESTful 서비스이다. 구글 클라우드 스토리지와 S3는 각기 고유한 기능을 제공하지만, 많은 기능과 기본 API가 서로 유사하다. 구글 개발자들이 boto에 많은 기여를 해준 덕분에 boto로 구글 클라우드 스토리지의 모든 기능에 접근할 수 있다. 이 책에서 설명하는 많은 비법은 구글 클라우드 스토리지에서도 작동한다.

1.5.2 boto에서 구글 클라우드 스토리지 자격증명 관리하기

구글 클라우드 스토리지가 별도의 서비스이므로, 이 서비스를 위한 별도의 자격증명들이 있다. 이 자격증명들은 boto 설정 파일에 다음과 같이 추가하면 된다.

```
[Credentials]
gs_access_key_id = <구글 클라우드 스토리지 접근 키 ID>
gs_secret_access_key = <구글 클라우드 스토리지 비밀 접근 키>
```

설정 파일에 추가한 다음에는 다음과 같이 구글 클라우드 스토리지에 연결할 수 있다.

```
% python
Python 2.7.1 (r271:86882M, Nov 30 2010, 10:35:34)
[GCC 4.2.1 (Apple Inc. build 5664)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import boto
>>> gs = boto.connect_gs()
```

1.6 AWS를 사용할 지역 찾기

기본적으로 boto에서 EC2 서비스에 연결할 때, 라이브러리는 US-EAST-1 지역에 접속한다. 원래는 이 지역만 사용할 수 있었지만, AWS가 상당히 확장되어 이 책을 쓰고 있는 현재 다음과 같은 지역의 EC2를 사용할 수 있다.

- us-east-1 [미국 동부 (북 버니지니아 주)]
- us-west-1 [미국 서부 (북 캘리포니아 주)]
- us-west-2 [미국 서부 (오레곤 주)
- eu-west-1 [EU (아일랜드)]
- ap-southeast-1 [아태 (싱가포르)]
- ap-southeast-2 [아태 (시드니)]
- ap-northeast-1 [아태 (동경)]
- sa-east-1 [남미 (상파울로)]

boto는 이 지역을 찾아내고 연결할 수 있는 많은 방법을 제공한다. 예를 들어 다음 코드는 해당 서비스(여기서는 EC2)에 대한 RegionInfo 객체의 목록을 반환한다. RegionInfo 객체는 connect( ) 메서드를 갖고 있으며, 이 메서드를 호출하면 그 지역에 대한 Connection 객체를 반환한다.

```
$ python
```

```
Python 2.7.1 (r271:86882M, Nov 30 2010, 10:35:34)
[GCC 4.2.1 (Apple Inc. build 5664)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import boto.ec2
>>> boto.ec2.regions()
[RegionInfo:eu-west-1, RegionInfo:us-east-1, RegionInfo:ap-northeast-1,
RegionInfo:us-west-1, RegionInfo:ap-southeast-1]
>>> eu_conn = _[0].connect()
```

연결할 지역의 이름이 미리 지정되어 있다면, 다음과 같이 코드에 명시할 수도 있다.

```
$ python
Python 2.7.1 (r271:86882M, Nov 30 2010, 10:35:34)
[GCC 4.2.1 (Apple Inc. build 5664)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import boto.ec2
>>> eu_conn = boto.ec2.connect_to_region('eu-west-1')
>>>
```

그리고 EC2에 연결할 때 boto가 사용할 기본 지역을 변경하려면, boto 설정 파일에 다음의 항목을 추가하면 된다.

```
[Boto]
ec2_region_name = eu-west-1
```

앞의 예와 같이 boto 설정 파일에 추가한 후에는, 인자 없이 boto.connect_ec2() 메서드를 호출하면 기본적으로 eu-west-1 지역에 대한 연결 객체를 생성한다.

1.7 boto의 디버그 출력 활성화하기

때로는 코드가 원하는 대로 작동하지 않을 수도 있다. HTTP 기반의 API로 원격 서비스에 연결해 처리할 때에는, 실제 전송된 HTTP 요청과 원격 서버가 보내온 응답을 로그에 자세히 남기는 방법이 가장 좋은 디버깅 도구가 된다.

boto는 파이썬 로깅 모듈로 매우 자세히 로그를 남길 수 있다. 파이썬 로깅 모듈에 대한 자세한 설명은 "http://docs.python.org/library/logging.html"에서 볼 수 있지만, 다음 예제와 같이 명령하면 boto를 사용할 때 콘솔에 자세한 디버그 메시지를 출력할 수 있다.

```
% python
>>> import boto
>>> boto.set_stream_logger('paws')
>>> ec2 = boto.connect_ec2(debug=2)
>>> s3 = boto.connect_s3(debug=2)
>>>
```

set_stream_logger() 메서드에는 원하는 어떤 문자열도 전달할 수 있으며, 이 문자열은 모든 로그 항목의 제일 앞에 나타난다. 이제부터는 EC2나 S3 연결에 수행하는 모든 연산은 대화형 파이썬 셸에 모든 디버그 로그를 출력한다.

1.8 boto에서 소켓 시간 제한 설정하기

boto가 클라우드 서비스와 통신하기 위해 사용하는 모든 API는 HTTP에 기반하고 있다. 다시 말하면 내부적으로는 소켓을 사용해 분산 서비스와 통신하고 있다는 의미이다. 때로는 서비스의 응답이 없거나 애플리케이션과 서비스 간의 통신 계층이 불안정할 수 있다. 이런 상황을 적절히 처리하려면 시간 제한을 설정하는 일이 중요하다. 시간 제한을 통해 애플리케이션은 네트워크나 서비스에 대한 문제를 감

지하고, 문제를 직접 해결하거나 적어도 사용자에게 상황을 알려줄 수 있기 때문이다.

boto 설정 파일에 다음과 같은 항목을 추가하면 소켓 수준의 시간 제한을 명시한다. 시간 제한 값은 초 단위로 지정한다.

```
[Boto]
http_socket_timeout = 5
```