Amazon EC2:

Are you tired of all the hype about cloud computing? **Koen Vervloesem** proves it's not vapourware and shows you how to run Ubuntu in the cloud.

› Our first login to Ubuntu on Amazon EC2.

Our expert

Koen Vervloesem discovered free software in 2000 and has been playing with almost all free operating systems since then.

Amazon's Elastic Compute Cloud (EC2) is a flexible alternative to traditional server hosting. Essentially, it's nothing more than a bunch of servers running virtual machines on the *Xen* hypervisor, but the way you work with it is really flexible. You can launch your own virtual machines in a matter of minutes and shut them down at will, and they're only billed based on three parameters: the number of hours the machine runs, the generated network traffic and the type of instance (from small to large). It's truly on-demand computing: if your server has to handle peak loads, you just allocate more instances as needed. This also solves the problem of long-running hosting contracts, because Amazon doesn't require upfront costs. No wonder many startups and Web 2.0 companies such as Facebook use EC2 for their web applications.

As a user of EC2, you create or download a so-called Amazon Machine Image (AMI). This is a preconfigured package of an operating system and software, which you instantiate as a virtual machine on EC2. Ubuntu has official AMIs for Server Editions 8.04, 8.10 and 9.04. The virtual Ubuntu servers follow the standard Ubuntu maintenance life cycle. This means that a regular release is maintained with security and maintenance updates for 18 months, while a long-term support release such as 8.04 is maintained for five years. Let's see how this works in Ubuntu 9.04.

Official Ubuntu AMIs

Version	Availability Zone	x86	x86_64
8.04 (Hardy)	Europe	ami-30c0e844	ami-3ac0e84e
8.04 (Hardy)	US	ami-5d59be34	ami-2959be40
8.10 (Intrepid)	Europe	ami-80c0e8f4	ami-84c0e8f0
8.10 (Intrepid)	US	ami-5059be39	ami-255bbc4c

Signing up for Amazon EC2

You can use the Ubuntu AMIs freely, but you have to sign up for an Amazon EC2 account and give them your credit card information to be billed for your usage. So, create an AWS account on **http://aws.amazon.com** and go to **http://aws.amazon.com/ec2**, where you choose Sign Up For Amazon EC2. Next, enter your credit card information. After you've registered, the website asks you to create an X.509 certificate or to upload one. If you don't have one yet, let Amazon create one for you and download the private key. Don't lose it, because Amazon deletes it on its servers! You also need to download the certificate file. We saved both in the directory **~/.ec2**. Make a note of your Account Number, which is a number of the form XXXX-XXXX-XXXX that's shown under Welcome, Name if you're logged in to the website.

Now that your account is ready, we have to set some environment variables, because the EC2 tools that we're going to install have to find information such as the private key and the certificate file. This is the reason why we add the following lines to our **~/.bashrc**:

» **Last month** We avoided packet filtering by tunnelling SSH traffic over HTTP.

Up in the cloud

```
export AWS_USER_ID=XXXXXXXXXXXX
export AWS_ACCESS_KEY_ID=XXXXXXXXXXXX
export AWS_SECRET_ACCESS_KEY=XXXXXXXXXXX
export EC2_PRIVATE_KEY=$HOME/.ec2/pk-XXXXXX.pem
export EC2_CERT=$HOME/.ec2/cert-XXXXXXXXXX.pem
export JAVA_HOME=/usr/lib/jvm/java-6-openjdk/
```

You can find the first three numbers on the page called Access Identifiers in your account page of Amazon AWS. **AWS_USER_ID** is your account number but without the hyphens. The other two variables can also be found on your account page. **EC2_PRIVATE_KEY** and **EC2_CERT** refer to your private key and certificate, respectively. Make sure that you adapt the filenames appropriately. After that, open a new terminal or execute **source ~/.bashrc** in the same terminal to reread the environment variables.

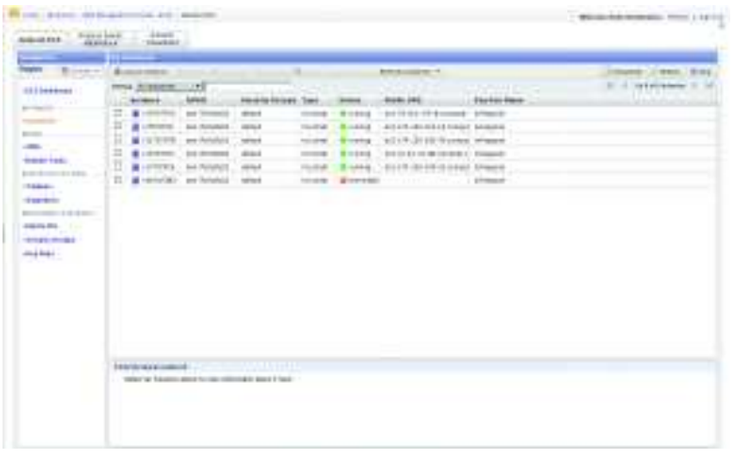
Installing the tools

Now that your account is ready and the environment variables are in place, we only have to install the EC2 tools. If you have version 9.04 (Jaunty) installed, this is simple. Just enable the Multiverse repository (because Amazon's EC2 tools are not open source) and install the tools – for example, with **sudo apt-get install ec2-api-tools**.

To see if the tools are set up correctly, simply find out whether the following command works:

```
ec2-describe-images -o self -o amazon
```

If all goes well, you get a long list with all of the AMIs Amazon offers to launch on EC2. If, however, you're greeted with a 'Client.AuthFailure' error message, then there's something wrong with your Amazon Web Services account (have you provided valid credit card information, for instance?), or it could mean you don't refer to the correct private keys or certificate file in **~/.bashrc**.



Now you have to generate an SSH keypair to log in to your EC2 instance. This goes as follows:

```
ec2-add-keypair lxf-keypair > ~/.ec2/lxf-keypair.pem
```

And then we authorise SSH access, because otherwise we can't log in to our instances (if you install a web server on the instance, you also have to authorise access to port 80):

```
ec2-authorize default -p 22
```

As a last step, we ensure that the permissions of all key files are correct. You don't want them to be accessible by others:

```
chmod 600 ~/.ec2/*.pem
```

Launching our first AMI on EC2

Now we have all our mechanics ready to run an Ubuntu instance on EC2. The only thing we need to know is the AMI ID of a public Ubuntu image (we've made a note of a few in the Official Ubuntu AMIs box over the page). For example, if »

› **Use the AWS Management Console to check the status of your EC2 instances.**

Quick tip

If you want to run another instance type than the default (which is m1.small), then use the option **-t** in the command **ec2-run-instances**.

Should you migrate to cloud servers?

Ubuntu Server 9.04 has added useful profiles to GNU screen, distributed under the name *screen-profiles*. This makes it possible to show some system information in the status bar of the screen. One of the things *screen-profiles* can show is the approximate cost of your system running on Amazon EC2. It takes into account the system's uptime, the number of processors and the network activity. You can also use the tool on servers that don't run on EC2. In this way, you can see how much it would cost to run an existing physical or virtual server on EC2. You can also call the *ec2-cost* script from the command line with **/var/lib/screen-profiles/ec2-cost --detail**, which provides a more detailed computation.

› **Ubuntu's *screen-profiles* show some interesting information about your server, including the cost on Amazon EC2.**

› **Know what you're paying for on EC2. Use *screen-profiles* on servers that don't run on EC2 to see if it's worth migrating them.**

you want to run a 32-bit Ubuntu Hardy in the US availability zone (you can choose between the US and Europe), you execute this as follows:

```
ec2-run-instances ami-5d59be34 -k lxf-keypair
```

Make sure you use the name of the keypair after the **-k** parameter, and not the local filename with the path. If everything is OK, then the command **ec2-describe-instances** should list a new instance with the status 'pending'.

Quick tip

You can list all of the AMIs everyone's allowed to start by using the command **ec2-describe-images -x all**.

Up and running

After a time (this bit could take a couple of minutes to complete, so you'll just have to be patient!) the status changes to 'running' and the instance is attached to an IP address and a hostname, as follows:

```
$ ec2-describe-instances
RESERVATION r-c5a1f2ac 181921386298 default
INSTANCE i-afa7a8c6 ami-5d59be34 ec2-174-129-86-66.compute-1.amazonaws.com ip-10-244-125-177.ec2.internal running lxf-keypair 0 m1.small 2009-07-29T13:45:18+0000 us-east-1d aki-6e709707 ari-6c709705
```

And then you can log into the virtual Ubuntu server you started, using the username 'ubuntu'. Just refer to the correct SSH key and the correct hostname (which begins with **ec2-** and ends on **amazonaws.com**):

```
ssh -i ~/.ec2/lxf-keypair.pem ubuntu@ec2-174-129-86-66.compute-1.amazonaws.com
```

Because you're using the keypair to log in, you don't have to enter a password. Your Ubuntu virtual machine is greeting you and now you can do all you want on your Ubuntu server in the cloud! You can create users or install packages such as a web server or database. The simplest way to do the latter is with the **sudo tasksel** command.

› You can keep tabs on the console output of an EC2 instance from your web browser.

By the way, Amazon is billing you from the moment your EC2 instance is running, so if you don't need it any more, don't forget to terminate it! This goes as follows:

```
ec2-terminate-instances i-afa7a8c6
```

You can find the instance ID, which is **i-afa7a8c6** in the example above, in the output of **ec2-describe-instances**. However, should you terminate an instance all of your changes to the image will be lost.

By default, each new virtual machine we instantiate gets a random IP address. A server running in the cloud is useful enough, but if you want it to host a service for other users, then it has to be available on a fixed IP address. Do we have to pay for this? No – luckily Amazon enables each user to reserve five IP addresses for free. This goes as follows:

1. Reserve an IP address with **ec2-allocate-address**. This returns a line with the address Amazon registered for you.
2. Bind a running instance to a reserved IP address with **ec2-associate-address**. This needs two parameters: the identifier of the instance and the IP address.

```
$ ec2-associate-address 174.129.11.115 -i i-15a3ac7c
ADDRESS 174.129.11.115 i-15a3ac7c
```

The process needs a couple of minutes, during which the system is not reachable on the old nor the new IP address. After that, you have only one IP address to remember, and you could set up your own domain name referring to that IP address. You can list all the addresses you've reserved with the command **ec2-describe-addresses**.

Creating your own AMIs

EC2 is awesome, but – as we mentioned earlier – terminating an instance destroys all your changes. So why not build your own custom AMIs? Fortunately, Eric Hammond has written a script that builds, bundles and uploads an Ubuntu or Debian AMI for Amazon EC2 (Hammond has also built some desktop AMIs – see the box, left). To get started, just download it from **http://ec2ubuntu-build-ami.notlong.com**.

Now we'll create an Ubuntu Jaunty AMI with the **ec2ubuntu-build-ami** command. This is a long command with many options. An example would be:

```
sudo bash ec2ubuntu-build-ami --codename jaunty --bucket lxf-bucket --user $AWS_USER_ID --access-key $AWS_ACCESS_KEY_ID --secret-key $AWS_SECRET_ACCESS_
```

A desktop in the cloud

You can also run an Ubuntu desktop on Amazon EC2 and Eric Hammond has built some publicly available desktop AMIs. Look up the AMI ID on **http://alestic.com** and launch it on EC2. For example, for the 32-bit Jaunty AMI:

```
ec2-run-instances ami-0b729462 -k lxf-keypair
```

This image will take longer to boot than the server types. Log into the AMI, update the system (**apt-get update && apt-get upgrade -y**) and create a user with **user-setup**. Now install and run an NX client on your local computer and point it to the external hostname of your Ubuntu desktop EC2 instance. Enter the username and password you supplied for your new user and choose Gnome as the desktop type.

Now you're running your Ubuntu desktop on Amazon's servers. Thanks to NX, it's as if it's running locally!

› Look! We've managed to connect to our Ubuntu desktop in the cloud from a machine running Fedora. The possibilities are endless...

Available instance types

Type	CPU	Memory	Storage	Platform	Name	Price/hour US	Price/hour EU
Small	1 EC2 Compute Unit	1.7GB	160GB	32-bit	m1.small	$0.10	$0.11
Large	4 EC2 Compute Units	7.5GB	850GB	64-bit	m1.large	$0.40	$0.44
Extra Large	8 EC2 Compute Units	15GB	1,690GB	64-bit	m1.xlarge	$0.80	$0.88
High-CPU Medium	5 EC2 Compute Units	1.7GB	350GB	32-bit	c1.medium	$0.20	$0.22
High-CPU Extra Large	20 EC2 Compute Units	7GB	1,690GB	64-bit	c1.xlarge	$0.80	$0.88

```
KEY --private-key ~/.ec2/pk-XXXXXXXXX.pem --cert ~/.ec2/
cert-XXXXXXXX.pem --bundle-opts --no-inherit
```

Here, **lxf-bucket** is the name of the S3 bucket where the AMI will end up. For the codename, we can use 'dapper', 'hardy', 'intrepid', 'jaunty' and 'karmic' for Ubuntu images, and 'etch', 'lenny' and 'squeeze' for Debian images. You can also run this command on an AMI instance running on EC2, but then you have to drop the **--bundle-opts --no-inherit** options.

Building and uploading

The **ec2ubuntu-build-ami** command takes a lot of time and spits out a lot of messages. Go out for a relaxing walk in the trees, make yourself a cup of tea or other caffeinated beverage and wait until the building and uploading is complete. When you come back and see the command has finished, it will suggest that you run an **ec2-register** command to register the AMI you uploaded to Amazon S3. This will return an AMI ID, like this:

```
$ ec2-register lxf-bucket/ubuntu-9.04-jaunty-
custom-20090730.manifest.xml
IMAGE   ami-7b6c8d12
```

You can also look up this AMI by using the **ec2-describe-images** command, as follows:

```
$ ec2-describe-images
IMAGE   ami-7b6c8d12   lxf-bucket/ubuntu-9.04-jaunty-
custom-20090730.manifest.xml 181921386298   available
private      i386   machine
```

Swyzzle user forum

The Atlanta-based company Swyzzle offers a website to create your own multimedia show. On the server side, it's using Ubuntu on Amazon EC2. Laurie Hall, vice president of products, explains why her company is such a big fan of this combination: "We moved from another cloud platform to Amazon EC2 due to its ability to support many different platforms. We selected Ubuntu because our media server software, *Wowza*, supported it and we wanted to use a Unix-like platform. We had no prior experience with Ubuntu, although we had several years' experience with many flavours of Unix. I'd suggest selecting one of the instances built by **http://alestic.com** since these are clean and well supported. The user forum (ec2ubuntu at **http://groups.google.com/group/ec2ubuntu**) has been extremely valuable in our transition to the platform. I'd highly recommend that anyone trying to implement EC2 on Ubuntu subscribes to this group."

This shows that we now have a private image available, identified by **ami-7b6c8d12**.

This instance can be run like the other instances we launched. So, in this example:

```
ec2-run-instances ami-7b6c8d12 -k lxf-keypair
```

Now wait a bit and SSH into your own Ubuntu Jaunty server on Amazon EC2 (only a root user is created by default). If the machine doesn't boot up, you can look at the kernel's console output with **ec2-get-console-output** and the instance ID.

Once that's done, you probably want to customise this image further. You can add packages with **--package NAME**. You can also run an external script before bundling of the AMI, with the option **--script FILE**. If you want to create a desktop AMI, then use **--desktop nx**. This also installs an NX server on the AMI, so that you can access the Linux desktop in the cloud from an NX client.

Besides the command-line tools, Amazon offers a web-based interface to manage your EC2 instances. Just go to **http://console.aws.amazon.com/ec2** and log into the website with your AWS account. Now you can see the status of your EC2 activity in the EC2 dashboard. If you click on Instances, you see all your running EC2 instances, and can terminate and reboot them. You can also look at the console output of an existing instance or launch a new instance.

If you click on AMIs, you can see all of the AMIs and select which ones you'd like to view. You can register a new AMI, deregister an existing one or change the permissions of your AMIs. For example, by default an AMI is made private, so only the owner is able to launch it. However, you can make it public to share it with other users. You can also add other users if you have their AWS account number. All these tasks can be done with the command-line tools too, but the web-based interface gives you a better overview.

Uses for Ubuntu in the cloud

Why would you use Ubuntu on Amazon EC2? If you're hosting a web application and have to handle peak loads regularly, then it can be vital – but it also has other uses. For example, if you sometimes need a lot of computing power but you don't have any fast hardware available, you can use the High-CPU Extra Large type, which is roughly equivalent to eight virtual cores with 2.5GHz Opteron or Xeon processors (see Available Instance Types, above, for all the options you can choose from). This is nice virtual hardware if you'd like to speed up your kernel building when you only have old hardware. And remember: you're only billed by the hour, so it costs you virtually nothing! LXF

Quick tip

The first command you have to execute when you launch an Ubuntu instance on EC2 is **sudo apt-get update && sudo apt-get upgrade -y**. This ensures the system is up to date.

» **Next month** We'll use Ubuntu's UEC technology to run our own cloud.