*Outsource Your Infrastructure*

# Programming Amazon Web Services

## *S3, EC2, SQS, FPS, and SimpleDB*

O'REILLY®

*James Murty*

# Table of Contents




Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

# Chapter 6. Using EC2 Instances and Images

When you run an instance in Amazon Elastic Compute Cloud (EC2), you gain all the power and the responsibility of a server administrator. You can install any software you want and configure your instances as you please, but you must ensure that they work correctly, are properly secured, are resistant to faults, and do not violate Amazon's acceptable use provisions.

In this chapter we discuss the key features of EC2 instances, as well as the issues you will have to address if you wish to run applications reliably in the EC2 environment. We will discuss how you can customize an instance and permanently save the result as an Amazon Machine Image (AMI), as well as how you can create your own AMIs from scratch to maintain complete control over the content of your images.



This chapter deals with advanced topics, and readers will need to have an in-depth knowledge of Linux systems, or a keen desire to acquire this knowledge, to obtain the full benefit of these powerful EC2 mechanisms.[2] Readers who merely want to get an idea of what is possible, or who have basic requirements, need not feel discouraged; we will take things slowly to start with and progress from simpler examples to more complex ones.

[2] We recommend you find a good book or two on Linux server administration. Two worthy offerings from O'Reilly are *Linux System Administration* by Tom Adelstein and Bill Lubanovic, or *Linux in a Nutshell* by Ellen Siever et al.

## 6.1. EC2 Instances in Detail

### 6.1.1. Linux Kernel

EC2 instances are based on a Xen-compatible Linux kernel compiled with the GNU C Compiler (GCC) version 4.0. The kernel version used in this book was 2.6.16. Unlike standard Linux machines, it is not possible in EC2 to use an alternative kernel to the one provided by Amazon, because the Xen software provides the kernel, rather than reading it from the boot disk. Therefore any software you use in your instance must be compatible with the kernel provided; if you wish to use additional modules, these must be loaded dynamically, rather than compiled into the kernel.

The specific Linux kernel version limitation may change in the future to allow a broader range of operating systems, but for now EC2 users must work within these compatibility boundaries. These boundaries are not too onerous; the Linux operating system provides very powerful server tools, is highly configurable, and most distributions are freely available. You can also create instances using any Linux distribution you prefer, provided


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

it is compatible with the EC2 kernel. Amazon favors the Fedora Linux distribution and uses this version as the basis of the public AMIs it provides.

In this book we also will focus on the Fedora distribution, because it is known to work well in EC2 and many of the instructions, resources, and tools available for EC2 users assume this is the version of Linux you are running. If you wish to use an alternative distribution, feel free; just be aware that every distribution is slightly different, so the instructions contained in this book may not be directly applicable to other versions of Linux. Other distributions that have proven popular with EC2 users are CentOS and Ubuntu. Be sure to check the EC2 discussion forums to see whether others have had success with the distribution you intend to use. With luck, you may even find a public AMI already provided by a member of the EC2 community that you can use as a starting point.

## 6.1.2. Network Addressing

Instances are assigned IP addresses and DNS names when they are launched in the EC2 environment. These addresses are dynamically assigned to an instance for its lifetime. This means that the addresses will refer to the instance for as long as it is running, but as soon as the instance is terminated, the addresses will be released and may be assigned to a new instance.

Instance addresses are divided into two categories: private and public.

### *Private addresses*

The private IP and DNS name addresses allocated to an instance are internal to the EC2 environment and cannot be accessed from outside the service. You should always use these addresses for communications between your instances, because it will allow EC2 to route your network traffic as quickly and efficiently as possible.

### *Public addresses*

The public IP and DNS name addresses allocated to an instance can be used by any device outside the EC2 environment to reach the instance. You can use these addresses directly to access an instance over the Internet, or you can create DNS configurations that translate your own domain names to these addresses.

Public DNS names can also be used by instances inside EC2, because they resolve to the instances' private IP addresses within the system. However, an instance's public IP address *cannot* be used inside EC2, because it will not resolve to an internal address.


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

You can look up the public and private addresses assigned to each instance in the instance's metadata, and we will discuss how to do this below. The public and private DNS names assigned to an instance can also be listed by the DescribeInstances operation, which we covered in Chapter 5 in Section 5.6.3."

#### 6.1.2.1. Dynamic addressing issues

The lack of support for static addressing of instances can cause issues for developers running publicly accessible applications in EC2, and more serious problems may be encountered by developers seeking to provide applications with high availability.

Publicly accessible web applications generally need to be accessible through a constant, unchanging domain name that users can remember and bookmark. Due to the dynamic addressing of EC2 instances, it is not a good idea to use the standard DNS system to assign a domain name to an instance. When the instance is terminated due to a failure or for general maintenance, the domain name will no longer map to a server, and requests against that name will fail. Even if a new EC2 instance is started immediately to replace the terminated one, and the DNS entry is immediately updated, it will take some time for the new instance's IP address to propagate through the DNS system, and the application will remain unavailable until this happens.

A common workaround for this problem is to use a *dynamic DNS* service to assign domain names to your instances, instead of the standard DNS system. For an example, refer to Section 7.1" in Chapter 7. The two main advantages of dynamic DNS services is that they generally provide a mechanism for IP addresses to be updated automatically using simple APIs, and they propagate the new address much more quickly than standard DNS.[*] Dynamic DNS services do not completely solve the addressing problem, however, because it can still take time for your users' computers to become aware of address changes; in the meantime your application may appear to be broken.

[*] Some dynamic DNS services provide additional features, such as basic load balancing for multiple instances, which is performed by providing the IP address of a different instance for each domain name look-up query the service answers.

The lack of static addressing in EC2 is especially problematic for publicly accessible applications that require high availability, because the majority of techniques for ensuring high availability rely on the presence of a public-facing server that has a static IP address. At present, there is no real solution for the issue beyond running an additional, statically addressed server outside EC2 as a frontend for your application.

### 6.1.3. Instance Data

EC2 provides contextual data to each instance running in the EC2 environment. This data includes a set of metadata information about the instance itself, which can include arbitrary custom user data that you provide to an instance when you launch it. This metadata and user data is very useful for configuring the behavior of your instances.


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

The contextual information provided by the EC2 environment is known as *Instance Data*. It is made available to instances through a simple HTTP interface available inside the EC2 environment at the location http://169.254.169.254. Instances can retrieve this information by sending standard GET requests to Universal Resource Identifier (URI) paths at this location; for example, to list the instance data versions available, you can run the `curl` command on your instance as follows:

```
ec2$ curl http://169.254.169.254/
1.0
2007-01-19
2007-03-01
2007-08-29
```

Metadata is returned from the instance data service as plain text strings with the mime type `text/plain`. Metadata values may comprise single items or a set of multiple items. Items are listed as strings that are delimited by ASCII line feed and carriage-return characters. If you provide custom data when you start an instance, that data will be returned as binary data with the mime type `application/x-octet-stream`. If you request a piece of instance data that is not available, the instance data service will return an HTTP `404: Not Found` error response. If you request a multi-value item but leave off the trailing slash in the URI, the service will return a `301: Permanent Redirect` response indicating the correct URI you should use.

**NOTE**

Instance data text values do not include new-line characters at the end of the string. Single-value items return only the item text without a terminating new line character, and multivalue lists include ASCII line-feed and carriage-return characters between each value in the list, but not after the last item.

### 6.1.3.1. Instance data versioning

The instance data service made available in the EC2 environment is versioned, so although the structure and content of the available information may change over time, your scripts will not be affected provided they refer to a specific version of the service.

Table 6-1 lists URI resource paths that retrieve information about the different versions of instance data available from the instance data service. These resource paths are appended to the service's location URI, http://169.254.169.254.


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

**Table 6-1. Resource paths to access versioned instance data**

| Resource Path | Description | Example Response |
|---|---|---|
| / | Returns a list of the instance data versions available | As of January 2008, there were four versions:<br>`1.0`<br>`2007-01-19`<br>`2007-03-01`<br>`2007-08-29` |
| */latest/* | Automatically retrieves the most recent version of instance data, for example this resource path will be an alias for`2007-08-29` while it is the latest version | `user-data`<br>`meta-data/` |
| */2007-08-29/* | Use a specific named version of instance data | `user-data`<br>`meta-data/` |

### 6.1.3.2. Instance Metadata

A set of metadata information is available to every instance running in EC2 to help the instance learn about itself and the environment it is running in.

Table 6-2 lists resource path fragments that can be added to the base URI that refers to the instance data version you wish to access. For example, http://169.254.169.254/2007-08-29/meta-data/ will be the base URI to retrieve metadata information from the 2007-08-29 version.

**Table 6-2. Resource path fragments to access instance metadata**

| Path Fragment | Description | Example Response |
|---|---|---|
| *reservation-id* | The identifier of the reservation set in which the instance was started. | `r-d97e99b0` |
| *ami-id* | The identifier of the AMI from which the instance was launched. | `ami-889075e1` |
| *ami-manifest-path* | The location in S3 of the AMI's manifest file. | *oreilly-aws/ami-fedora7-base.img.manifest.xml* |
| *ami-launch-index* | A zero-based offset value that uniquely identifies each instance in a batch started at the same time. | The first instance launched: 0<br>The third instance launched: 2 |
| *instance-id* | The instance's identifier. | i-b434d4dd |
| *instance-type* | The type of the instance: `m1.small`, `m1.large`, or `m1.xlarge`.<br>This metadata field may not be made available to instances launched from older AMIs. If this value is not present, you can assume that the instances are `m1.small`. | .m1.small |
| *local-hostname* | The instance's private DNS name address. | *domU-12-31-36-00-3C-A2.z-1.compute-1.internal* |
| *public-hostname* | The instance's public DNS name address. | *ec2-72-44-57-154.z-1.compute-1.amazonaws.com* |
| *local-ipv4* | The instance's private IP address. | 10.253.67.80 |
| *public-ipv4* | The instance's public IP address. | 72.44.57.154 |
| *public-keys/* | Lists the keypair names provided to the instance, if any. This listing is the root of a hierarchy described further in the next two items. | 0=ec2-private-key |
| *public-keys/n/* | A list of the formats in which the keypair's public key component is available. | openssh-key |


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

| Path Fragment | Description | Example Response |
|---|---|---|
| *public-keys/n/ openssh-key* | The public key data for the *n*th public key item in the *openssh-key* format. | ssh-rsa AAA...ZEf ec2-private-key |
| *security-groups* | A list of the security groups the instance belongs in. | default |
| *product-codes* | A list of the product codes associated with the instance. | A79EC0DB |

#### 6.1.3.3. Instance user data

You can provide EC2 with up to 16 KB of arbitrary data when you launch an instance. This information is called user data and it is made available to the instance by the instance data service. By passing data to your instance with this mechanism, you can easily supply your instances with custom configuration information or instructions.

The user data you provide is made available to your instances as binary data from the instance data service at the URI resource-path fragment *user-data*. For example, to obtain this data from the *2007-08-29* version of the service, you would use the URI http://169.254.169.254/2007-08-29/user-data. The instance data service returns exactly the same bytes as you supplied to the RunInstances operation, so it is possible to use binary information instead of plain text; for example, you could provide encrypted binary content or images as user data. You must Base-64-encode the user data you supply to the RunInstances operation; EC2 will automatically decode the user data before making it available via the URI.

When you start multiple AMI instances at the same time, each instance receives exactly the same user data. In some circumstances you may want to provide different information to each instance in a set, even though they are launched at the same time. This is possible by taking advantage of the unique launch index offset value that is assigned to each instance you launch in a set, and which is available from the `ami-launch-index` item in the instance data service. If you provide user data that can be interpreted differently by your instance, depending on that instance's launch index number, you can supply slightly different data to each instance.

### 6.1.4. Performance

There are a broad range of factors that will influence the level of performance you can expect from a particular EC2 instance. Some of these factors are consistent and well-known, such as the network bandwidth you can expect within the EC2 environment; others are ill-defined or subject to change between different instances or even on a single instance over time. Overall, you should not rely too much on the stated performance characteristics of instances to indicate how well your own application will perform. The only way to know for sure how your application will perform in EC2 is to test it thoroughly for yourself.


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

Bearing that in mind, here is a list of the main factors that affect the performance of EC2 instances.

### *Instance type*

There are three different instance types available in EC2 with different performance characteristics and platforms; these were summarized in Table 5-1. These instance types determine the amount of CPU processing power, RAM, and storage space allocated to an instance.

### *Shared subsystems*

Each instance runs as a virtual machine on underlying physical hardware, which may run multiple instances at a time. Although CPU, RAM, and storage resources are dedicated to each instance and perform at a constant rate, the networking and disk I/O subsystems are shared between all the instances running on an underlying machine, and their performance may fluctuate. Fortunately, this fluctuation will always be to your benefit.

The networking and I/O resources of the underlying hardware are shared equitably, such that each instance is guaranteed at least a baseline performance level; however, if an instance is fortunate enough to be running on a host where the other instances are idle, it can access more of these resources than the guaranteed baseline. This means that some of your instances may occasionally perform better than expected, though this benefit is entirely unpredictable.

### *Network bandwidth*

Instances are allowed 250 Mbps of network bandwidth within the EC2 network. Bandwidth outside of the EC2 environment will vary.

### *Storage space initialization*

The first write operation to any location on the virtualized disks used by EC2 instances will be slow. Only after an initial write has been performed to a specific disk location will subsequent writes to that location run at full speed. For applications where write operations must occur at top speed from the very beginning, you will have to initialize your instance's storage space by writing once to every location. This task can take hours, depending on the instance type you are using and how many partitions you must initialize.


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

*RAID*

To improve the performance of software-based RAID on an instance, you can increase the default minimum reconstruction speed from 1MBps to 30MBps. This will help to overcome the initial write-speed penalty imposed by the virtualized disks.

# 6.2. Data Management in EC2

EC2 instances do not store data in the same way as physical servers. It is vital to understand how data storage works in your instances to manage your data efficiently and keep it as safe as possible.

## 6.2.1. Storage Locations

Data storage resources are made available to your instances as standard Linux partitions. These partitions are virtual disk drives that exist for the lifetime of your instance. Table 6-3 lists the storage partitions available to EC2 instances of different types.

**Table 6-3. EC2 storage partitions by instance type**

| Location | Mount Point | Small Instance | Large Instance | Extra-Large Instance |
|---|---|---|---|---|
| /dev/sda1 | root (/) | 10 GB | 10 GB | 10 GB |
| /dev/sda2 | /mnt | 150 GB | - | - |
| /dev/sda3 | /swap | 0.9 GB | - | - |
| /dev/sdb | /mnt | - | 420 GB | 420 GB |
| /dev/sdc | - | - | 420 GB | 420 GB |
| /dev/sdd | - | - | - | 420 GB |
| /dev/sde | - | - | - | 420 GB |

Partitions automatically associated with a mount point by EC2 are formatted with the `ext3` filesystem. Partitions that are not mounted, such as */dev/sdc* for the large and extra-large instance types, are not preformatted. These partitions are available for use in a RAID configuration, or you can manually format and mount them yourself.

The root partition plays a vital role in EC2 instances. When you launch an AMI, the image's data files are copied from S3 to this partition, mounted as the instance's root partition, and run as the instance's boot volume. When your instance starts, the root partition will have the contents originally saved to the AMI. The root volume has a maximum size of 10 GB in all instance types, though the formatted size of the mounted volume will match the size of the AMI, which may be less than this. Partitions other than the root are empty when your instance starts.


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

All the data storage partitions made available to your instance are ephemeral. It is your responsibility to store your data somewhere outside the EC2 instance if you want it to outlive your instances.

## 6.2.2. Ephemeral Storage

Although instances running in EC2 appear to have storage partitions like any standard computer system, it is vital to understand that data on these partitions are not actually stored outside the instance. All the storage resources are ephemeral, and the data are only "stored" for as long as the instance is running. When the instance is rebooted, the data remains intact; but if you terminate the instance, or if it fails for reasons beyond your control, any data created since the instance first booted will be lost. To store data beyond the lifetime of an instance, you must save it to a location outside the EC2 environment.

Because EC2 instances do not automatically save their data outside the EC2 environment, it is your responsibility as the owner of the instance to ensure that your data are persisted to an external resource.

This notion can be a daunting one for anyone accustomed to working with physical hardware, where power cycling the server has no effect on data stored on the server's disk. However, rather than focusing on this added responsibility as a burden, it is worth pointing out that EC2's ephemeral storage merely forces you to adopt the regular, rigorous, and well-considered back-up processes that you would put in place eventually anyway. The real difference is that you must put your back-up processes in place from the very beginning when working with EC2 instances; unlike physical servers, there is no option to defer the task and simply hope that your disk drives do not fail.

Amazon intends for EC2 users to rely primarily on S3 to handle their persistent storage requirements. After all, S3 is the storage service component of the AWS services architecture; EC2 is designed only to provide computing resources, and these services were designed to be complementary. A key benefit of using S3 to reliably store your instance's data is that there is no bandwidth cost for transferring data between EC2 and S3 buckets located in the United States.[*]

[*] You will incur data transfer fees if you transfer data between EC2 and S3 buckets located anywhere other than the United States.

There are a number of strategies you can use for working with the ephemeral storage limitation of EC2. Here are some solutions that use S3 for long-term data storage.

### *Treat EC2 storage as a cache*

> Design your instances to obtain all their data from an external storage system, and treat the data in your instance as a local cache that need not survive longer than the


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

instance. This could be an option if you have a simple system, like a web server, where the pages and images are stored in S3, and your instance merely performs processing tasks.

### Perform scheduled backups

Perform regular backups of important data into S3 and have a mechanism to recover from those backups if necessary. In situations where your instance's data is relatively static, a regular back-up regime could be sufficient to minimize the chance of losing data. This approach can also work for database-driven applications if you could afford to risk losing data. In the case of databases, your back-up process could involve occasionally storing a backup of the entire database and performing more frequent backups of your transaction logs.

#### *Perform scheduled bundling of an entire instance to an AMI*

With the instance-bundling feature of EC2, you can make backups of your instance's entire root partition and create a new registered AMI that you can start from if things go wrong. This approach will make sense if you have made system-level changes in your instance, such as installing new software or changing your software's configuration.

#### *Mount S3 storage as a local partition*

There are software tools emerging that can make your S3 resources appear as just another partition attached to your instance, in which case you can back up your data to S3 by simply saving it to the appropriate partition on your instance using standard file management programs. Such tools are still fairly new, and there are questions as to whether they can really provide the reliability and fault tolerance you may need, but they provide an attractive option. In Section 4.3.2" in Chapter 4, we gave a brief introduction to one of these tools.

#### *Take your chances*

Of course, you can always ignore the risks and hope like crazy that your instances run forever without failing. This approach is even more reckless when using EC2 than when using physical hardware, because there is no possibility of sending a failed disk off to the experts to have the data recovered. This is only an option if you do not care about your data.

It should go without saying that you will have to choose a backup strategy that works for you, based on the kind of system you run in EC2, the frequency of data changes in your system, the importance of your data, and the effort it will take to replace data if they are


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

lost. When you are planning your backup strategy, we recommend you check the S3 forums where the merits and disadvantages of various approaches are discussed in great detail.

## 6.3. Modifying an AMI

The easiest way to create an EC2 instance perfectly suited to your purpose is to start from an existing AMI that is already close to what you need, modify it to do what you need, then save it as your very own image by bundling the running instance into a new AMI. When you bundle a running instance, the entire root filesystem is stored in a new AMI, along with the software and configuration changes you have applied.

In Section 5.5" in Chapter 5, we demonstrated how to find, launch, and log in to one of the public AMIs made available by Amazon. Amazon's public AMIs include only a small set of tools and applications, which makes them a good starting point if you want to customize your own image from a minimal base.

Alternatively, you can choose from many public AMIs provided by other EC2 users that already include the software and configuration settings to fulfill common needs. There are many preprepared images available that include ready-to-run environments for things such as cms or web application server environments for Ruby on Rails and Java. Unless you have unusual requirements, there is a good chance you will be able to find a public AMI that already includes everything, or almost everything, that you need. The vast majority of the public images are free.

Find more information about the public AMIs provided by Amazon and others here: http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=101.

### 6.3.1. AMI Security Auditing

When you use a public AMI, or any other AMI you did not create yourself, you must consider the security implications of running a system created by a third party. Although it is convenient to not have to go through the process of creating an AMI system from scratch, it is vital that you audit any public AMI thoroughly to check for unacceptable security risks involved in running it.

The following is a list of the basic steps you should take to audit an AMI instance prior to using it for important tasks or deploying it into a network of production EC2 instances. These steps alone will not be sufficient to ensure an instance contains no malevolent code or potentially damaging security mistakes, but they are a start. If you use a public AMI, it is entirely up to you to audit the instance to ensure it is suitable for your intended use.


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

**NOTE**

Even if you have created an AMI from scratch, such as by following the instructions in Section 6.5" later in this chapter, it is worthwhile to perform these auditing steps to make sure you have not accidentally made any security blunders yourself.

### *Research the AMI provider*

Ideally you should only use AMIs provided by individuals or organizations you trust. This may mean using Amazon's own AMIs in preference to those provided by other EC2 users, or checking the EC2 forums to find out whether any of the other public AMIs have been thoroughly vetted by other forum members.

This approach only gets you so far, because even a trusted source could accidentally produce a flawed system. Regardless of the source of the AMI, you should perform certain auditing steps.

### *Firewall untrusted instances*

Create a security group specifically for running untrusted instances, so they cannot communicate with any other instances you have running. If an untrusted instance cannot connect to your other instances, the chances of it causing any serious trouble are greatly reduced.

### *Secure Shell login permissions*

Make sure there are no backdoor Secure Shell login accounts for the AMI. In particular, obtain a listing of all the users configured in the system, and check the *~/.ssh/authorized_keys* file belonging to every account to ensure there are no unexpected entries. The only public key entries included in these files should belong to you or to another trusted entity.

If you use the keypair–based login method to provide login credentials to your instances dynamically, you can increase your security further by disallowing the use of plain passwords as login credentials. Do this by setting the following configuration options in */etc/ssh/sshd_config*:

```
PasswordAuthentication no
UsePAM no
PermitRootLogin without-password
```


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

### *System services*

Use the following command to check the services running on the system to ensure there are no unrecognized, potentially malevolent services. It is also wise to turn off any services you do not need.

```
ec2$ /sbin/service --status-all | grep running
```

### *Check for open ports*

Ensure that only legitimate ports are open for network communications. The following command lists the open ports:

```
ec2$ netstat -a | grep LISTEN
```

### *Scheduled jobs*

Confirm there are no malevolent or risky scheduled jobs on the system by checking all *crontab* files, both at the system level and for all user accounts on the system.

### *Software audit*

This item is left until last, because it is the hardest one to achieve. To be absolutely confident that a system is safe for use, it might be necessary to check that all the software installed on the system is legitimate by comparing installed executables, libraries, and the like with official versions.

If your security requirements are so strict that this process is necessary, you should probably build your own system from scratch instead.

## 6.3.2. Install Amazon's Bundling Tools

To save an instance to your own AMI, you will need to have Amazon's instance bundling tools installed on the instance. If you started from one of Amazon's public AMIs, these tools will already be installed; most other public AMIs will include them too. Check for the presence of the `ec2-bundle-vol` and `ec2-upload-bundle` commands.

If these tools are not available on the instance, you can install them yourself. Here are the commands you can use on an instance running Fedora to download and install Amazon's tools. The commands will be different for alternative distributions.

```
# Install the ruby library (required by Amazon's tools)
ec2$ yum -y install ruby
ec2$ yum -y install rubygems
```


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

```
# Install the Java library (required by Amazon's tools)
# Note: This command installs the open-source GNU version of Java, not Sun's
# version. Not all Java software is compatible with the GNU version.
ec2$ yum -y install java

# Amazon's EC2 API tools (Java-based)
ec2$ wget http://s3.amazonaws.com/ec2-downloads/ec2-api-tools.zip
ec2$ unzip ec2-api-tools.zip

# Amazon's EC2 AMI tools (ruby-based)
ec2$ wget http://s3.amazonaws.com/ec2-downloads/ec2-ami-tools.noarch.rpm
ec2$ rpm -i ec2-ami-tools.noarch.rpm
```

### 6.3.3. Configure System Services

For security and performance reasons, you should reduce the number of system services (daemons) running on your instance to the bare minimum. The Fedora distribution provides the `chkconfig` command to control which services start when the system boots, and the `service` command to list services currently running.

```
# List all services that will start on boot
ec2$ /sbin/chkconfig --list | grep 3:on

# List all services currently running
ec2$ /sbin/service --status-all | grep running
```

Here is the command you would run to to prevent the `yum-updatesd` automated system update service from starting when an instance boots and the command to terminate the service if it is already running.

```
# Disable service start-up on boot
ec2$ /sbin/chkconfig yum-updatesd off

# Stop currently running service
ec2$ /sbin/service yum-updatesd stop
```

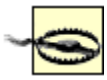
We recommend that you disable any services you do not specifically need, including items such as the following, which may run by default in a Fedora instance:

- `yum-updatesd`: Automatic updates
- `cups` or `cupsd`: Printing
- `pcscd`: Smart cards and smart card readers
- `sendmail`: Mail transport agent
- `kudzu`: Automatic detection and configuration of hardware

Make sure that you do enable the following services:

- `ntpd` (Network Time Protocol service): Ensures that an instance's internal clock is set to the correct time, which is very important when using AWS (see http://www.ntp.org).
- `dhcdbd` (Dynamic Host Configuration Protocol interface): Allows an instance to discover the network addresses assigned to it in the EC2 environment.


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25



Never disable the *sshd* service; it is necessary to log in via a Secure Shell.

## 6.3.4. Bundling an Instance into an AMI

Once you have your EC2 instance running the way you like, you must bundle it into a new AMI to keep your changes. The bundling process takes the entire root partition of the instance and packages it into a series of files ready to store in S3.

To bundle your system, use the `ec2-bundle-vol` tool available in Amazon's AMI toolset. Before you can run the bundling tool, you need to make sure you have all the information the tool will need to run.

### *AWS public and private certificate files*

Your AWS account has a public and private keypair associated with it. These keypairs are used as AWS credentials by the bundling tool, and they are required to encrypt and sign your AMI image so that only you and the EC2 service can use it. The public key is available via the AWS Credentials page on the AWS web site, and your private certificate file was made available to you when you first created your EC2 account. You stored this file somewhere safe, right? If not, you can generate new AWS credential keys from the credentials page.

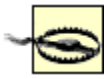
**NOTE**

Your AWS public and private key credentials are completely different from any keypair credentials you created in the EC2 service. Your AWS credentials are associated with your AWS account as a whole, but your EC2 keypair resources are only useful within EC2.

Your AWS credential key files must be copied to your instance to make them available to the bundling tool. Copy your public and private certificate files from your local computer to your EC2 instance using the `scp` (secure copy) command.

```
local$ scp -i ec2-private-key.enc *.pem \
          root@ec2-67-202-23-170.compute-1.amazonaws.com:/mnt
```


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25



If you intend to share your AMI, *do not* store your AWS credential key files anywhere on the instance's root partition, because the files will be included in the AMI. Your AWS key credential files are not encrypted, so if you accidently store them in an AMI, anyone who launches an instance from this AMI can use these keys and pretend to be you. To be safe, always copy these files to the */mnt* ephemeral storage location which is not included in the AMI.

### *AWS account number*

The AMI bundling tool requires a user identifier value associated with your AWS account. The identifier the tool needs is your AWS account number, *not* your AWS Access Key or Secret Key. If you use the wrong identifier value, you will be able to bundle the instance to an AMI, but you will never be able to launch it; your mistake will only become apparent after you have gone through the effort of storing and registering the AMI.

Your AWS account number is a 12-digit number in the format `1234-5678-9012`. You can find this number at the top of your Account Activity statement on the AWS web site. Do not include the hyphens in this number when you run the bundling command.

When you have all the necessary information, you are ready to run the bundling tool from inside your running instance. To list the arguments the tool understands, have it print a help message by running the command: *ec2-bundle-vol --help*. Here is a command that will bundle the root volume of a 32-bit (i386) instance into a set of image files, where each file name will start with the prefix *my-ami-filename.img*. These statements are all part of a single command that should be typed on a single line.

```
ec2$ ec2-bundle-vol
       --volume /
       --prefix my-ami-filename.img
       --destination /mnt
       --arch i386
       --size 10240
       --cert /mnt/cert-ABCDEFGHIJK123ABCDEFGHIJK123ABCD.pem
       --privatekey /mnt/pk-ABCDEFGHIJK123ABCDEFGHIJK123ABCD.pem
       --user 123456789012
```

**NOTE**


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

By default, the EC2 volume bundling tool creates image files in the temporary directory */tmp*. The temporary partition in EC2 instances may have a limited size that will not fit large images, so you should always use the `--destination` option to have these files written to the roomier */mnt* location.

Once you have bundled the image, you will be left with a numbered sequence of files stored in the */mnt* directory. The files will be named using the prefix name you provided and with a suffix of *.part.x*, where *x* is the part number. For example, the preceding command would create files with the names *my-ami-filename.img.part.1*, *my-ami-filename.img.part.2*, and so on. You must upload these files to S3 and register the image to make it available in EC2.

**NOTE**

If you have edited the filesystem table configuration file */etc/fstab* in your instance, you must use the `--fstab` option to use your version of the file instead of the default *fstab* file provided by Amazon.

## 6.3.5. Uploading AMI Files to S3

Your bundled AMI files must be stored in S3 to be available to the EC2 service. To upload these files, you can use the *ec2-upload-bundle* tool. This tool creates a bucket in S3 if necessary, updates the ACL settings of the bucket to grant the EC2 service read access to the bucket's contents, and uploads each image file into S3.

Here is the command that will upload the instance bundle we created earlier:

```
ec2$ ec2-upload-bundle
       --manifest /mnt/my-ami-filename.img.manifest.xml
       --bucket TargetBucketName
       --access-key AWS_ACCESS_KEY
       --secret-key AWS_SECRET_KEY
```

Notice that this command requires you to provide your AWS Access Key and Secret Key credentials on the command line. This practice can be risky if you are creating an AMI that will be shared, because your shell history may store the credentials you have entered and make them visible to whoever launches the AMI. To avoid the risk of accidentally leaving your credentials in the command history, we recommend that you load these credentials from a file, rather than typing them into the command prompt.


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

To load your credentials from a file, use a text-editing program on your instance to create two files, */mnt/access-key.txt* and */mnt/secret-key.txt*, and enter your AWS Access and Secret Key credentials respectively into these files. Then use the following variation of the earlier command, which reads your credentials from these files without making them visible in your command history.

```
ec2$ ec2-upload-bundle
       --manifest /mnt/my-ami-filename.img.manifest.xml
       --bucket TargetBucketName
       --access-key `cat /mnt/access-key.txt`
       --secret-key `cat /mnt/secret-key.txt`
```

**NOTE**

You may use any S3 client tool to upload your AMI files to S3, but you'll have to manually update the ACL of your target bucket to allow the EC2 service to read your AMI. The bucket's ACL must grant READ permission to the `za-team` EC2 user with the following canonical ID: `6aa5a366c34c1cbe25dc49211496e913e0351eb0e8c37aa3477e40942ec6b97c`.

# 6.4. Registering an AMI

Once your AMI files have been uploaded to S3, the last step necessary to make your own AMI ready for use is to register it. The registration step tells the EC2 service where to find the image's manifest file and, through this, all the other image files.

You can register your AMI using the RegisterImage API method described in Section 5.9" in Chapter 5, or you can use the `ec2-register` command provided with Amazon's tools. To use the API method, you would run the following commands in interactive Ruby:

```
irb> require 'EC2'
irb> ec2 = EC2.new
irb> ec2.register_image('TargetBucketName/my-ami-filename.img.manifest.xml')
```

When you register your image, the service will respond with the identifier assigned to your AMI. This identifier will also appear in the list provided by the DescribeImages API action. To make it easier to find your AMI in the images listing, use the value `self` in the image owners filter to display only your own images.

```
irb> ec2.describe_images(:owners => 'self')
```

Once your AMI is registered, you will be able to start instances based on this image, complete with all the changes you have applied.


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

# 6.5. Create an AMI from Scratch

Creating an AMI from scratch will give you the most control over exactly what the image will contain and how it will run. This process takes more effort than specializing an existing AMI, but you have the advantage of knowing exactly what the image contains.

The process for creating a new image requires access to a Linux system, on which you will create your image volume. Ideally, the system on which you create the AMI should have the same Linux distribution as you will use for the image; this will make the process much easier.

The AMI itself will be stored as a single file on the underlying system. This single file will be mounted, using the loop-back file mechanism, to look like a separate partition on your underlying system; once it is mounted, you can create the image's filesystem using standard Linux tools.

In this section we will guide you through the process of creating an AMI based on the Fedora 7 Linux distribution. The instructions are written with the assumption that the underlying Linux system is also Fedora version 7. We chose Fedora because it is known to be compatible with EC2, and many of the instructional guides available concerning EC2 instances use this distribution, so it should not be difficult to find help if you need it. Version 7 was the latest version of Fedora available when this book was written.

Before you proceed, please ensure that the Fedora system on which you will create your AMI meets the following requirements:

- The yum package manager is installed
- Ruby version 1.8 is installed
- Amazon's EC2 AMI tools are installed (see Section 6.3.2")
- Your AWS public and private key credential files are available (see Section 6.3.4")
- You have root access on the underlying system

The commands in the following sections are run as the root user on your underlying Linux system.

## 6.5.1. Prepare the AMI Filesystem

Create an empty file with at least 1.5 GB of space to host the AMI using the `dd` command. We will call this image file *ami-fedora7-base.img* and save it in our home directory.

```
$ dd if=/dev/zero of=~/ami-fedora7-base.img bs=1M count=1536
```


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

**NOTE**

The amount of space available on your image's root filesystem will be limited by the size of the image file. Because the image file is compressed during the bundling process, there is a minimal penalty for making the image larger than you need, so you may wish to make your image's root filesystem as large as possible by creating a 10 GB image file with a value of `10240` for the `count` parameter.

Create an `ext3` filesystem inside the image file.

```
$ /sbin/mke2fs -F -j ~/ami-fedora7-base.img
```

Mount the image file using the loop-back option. This allows your system to treat the image file as if it were a standard disk drive.

```
$ mkdir /mnt/ami-fedora7-base
$ mount -o loop ~/ami-fedora7-base.img /mnt/ami-fedora7-base
```

Prepare the empty image filesystem with paths for system devices and configuration files, which must be present before you can install an operating system on the image.

```
$ mkdir /mnt/ami-fedora7-base/proc
$ mkdir /mnt/ami-fedora7-base/etc
$ mkdir /mnt/ami-fedora7-base/dev

$ mkdir /mnt/ami-fedora7-base/var
$ mkdir /mnt/ami-fedora7-base/var/cache
$ mkdir /mnt/ami-fedora7-base/var/log

$ /sbin/MAKEDEV -d /mnt/ami-fedora7-base/dev -x console
$ /sbin/MAKEDEV -d /mnt/ami-fedora7-base/dev -x null
$ /sbin/MAKEDEV -d /mnt/ami-fedora7-base/dev -x zero
```

Create the text file */mnt/ami-fedora7-base/etc/fstab* to store the filesystem configuration information for your instance. Edit this file to contain the following entries for the appropriate instance type:

```
# Small Instance Type
/dev/sda1 /        ext3   defaults       1   1
none      /dev/pts devpts gid=5,mode=620 0   0
none      /dev/shm tmpfs  defaults       0   0
none      /proc    proc   defaults       0   0
none      /sys     sysfs  defaults       0   0
/dev/sda2 /mnt     ext3   defaults       0   0
/dev/sda3 swap     swap   defaults       0   0

# Large or Extra-Large Instance Type
/dev/sda1 /     ext3    defaults 1 1
/dev/sdb  /mnt  ext3    defaults 0 0
none      /proc proc    defaults 0 0
none      /sys  sysfs   defaults 0 0
```


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

**NOTE**

Additional partitions are available for large and extra-large instance types, which are not formatted or mounted by default. If you wish to mount these partitions, you must format them yourself when you first start the instance.

Mount the image's *proc* device in advance to avoid problems with the yum installation manager.

```
$ mount -t proc none /mnt/ami-fedora7-base/proc
```

## 6.5.2. Install Fedora 7 Base

In this section we will install the base Fedora operating system and updates using the `yum` package installer utility on your underlying system. Create a text file called *yum-ami.conf* and add the following configuration settings:

```
[main]
cachedir=/var/cache/yum
logfile=/var/log/yum.log
debuglevel=2
exclude=*-debuginfo
gpgcheck=0
obsoletes=1
reposdir=/dev/null

[base]
name=Fedora 7 - i386 - Base
mirrorlist=http://mirrors.fedoraproject.org/mirrorlist?repo=fedora-7&arch=i386
enabled=1

[updates-released]
name=Fedora 7 - i386 - Released Updates
mirrorlist=http://mirrors.fedoraproject.org/mirrorlist?repo=updates-released-f7&arch=i386
enabled=1
```

**NOTE**

If you are creating an AMI to run with the large or extra-large instance types, adjust this yum configuration file to install the 64-bit versions of software by setting the `arch` parameter to the value `x86_64` instead of `i386`.

We will now perform a base operating system installation by invoking yum. The following command runs yum with the configuration script we defined above and instructs it to install the software in the image mount point */mnt/ami-fedora7-base*. This process could take a long time, because it involves downloading and installing hundreds of packages. Be patient.


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

```
$ yum -c yum-ami.conf --installroot=/mnt/ami-fedora7-base -y groupinstall Base
```

When the base installation completes, copy the *yum-ami.conf* file to the image, so you can reuse it in your instances.

```
$ cp yum-ami.conf /mnt/ami-fedora7-base/etc/yum.conf
```

With the base install of Fedora 7 in place, it is time to add kernel modules to the system. Because EC2 instances all run on the `2.6.16-xenU` kernel provided by the EC2 environment, rather than the local kernel, we must replace the kernel modules included with the Fedora 7 base installation with versions that will work properly with the EC2 kernel.

A set of precompiled modules has been made available as a tar archive by Amazon. To install these EC2-specific modules, download the module pack and extract it to the root of the AMI image. Because the kernel modules installed by yum are not appropriate for the EC2 environment, it is worthwhile to remove these to save space.

```
# Download 32-bit modules compatible with the EC2 Linux kernel
$ wget http://s3.amazonaws.com/ec2-downloads/modules-2.6.16-ec2.tgz

# Extract the modules to the root image directory
$ tar xvzf modules-2.6.16-ec2.tgz -C /mnt/ami-fedora7-base/

# Remove the default Fedora modules that are incompatible with EC2
$ rm -fR /mnt/ami-fedora7-base/lib/modules/*.fc7
```

**NOTE**

If you are building a 64-bit AMI instead of a 32-bit one, the 64-bit modules are available in this same S3 bucket with the archive name *ec2-modules-2.6.16.33-xenU-x86_64.tgz*

The module archive provided by Amazon does not include the module dependency mapping files, so you should generate these with the following command. After running the command the image's kernel modules directory, */mnt/ami-fedora7-base/lib/modules/2.6.16-xenU*, should contain a *modules.dep* file among others.

```
$ /usr/sbin/chroot /mnt/ami-fedora7-base /sbin/depmod 2.6.16-xenU
```

By default the base installation of Fedora 7 is not configured to load any modules on boot. If you wish to load modules in your instance, you should add entries referring to these modules to the file */mnt/ami-fedora7-base/etc/modules*.


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

## 6.5.3. Configure Networking

To ensure that images play nicely with the EC2 environment's network, we must change some settings. First, we will tighten the default Secure Shell daemon settings so that only keypair-based logins are allowed, and standard passwords cannot be used to log in to your instance. Second, we will turn off DNS lookups in the daemon to ensure DNS hiccups will not prevent us from logging in. Edit the Secure Shell daemon configuration file */mnt/ami-fedora7-base/etc/ssh/sshd_config*, and set the `UseDNS` and `PermitRootLogin` variables as follows:

```
UseDNS  no
PermitRootLogin without-password
```

Configure the image's networking system to be active and to use DHCP for DNS resolution by creating the file */mnt/ami-fedora7-base/etc/sysconfig/network* with the following content:

```
NETWORKING=yes
HOSTNAME=localhost.localdomain
```

Next, create the file */mnt/ami-fedora7-base/etc/sysconfig/network-scripts/ifcfg-eth0* with the following content:

```
ONBOOT=yes
DEVICE=eth0
BOOTPROTO=dhcp
```

## 6.5.4. Startup Scripts

We will configure your AMI to allow keypair-based logins, so when the instance starts you can log in using your private key, and you will not have to store hard-coded login credentials in your AMI. The public key information required to authenticate keypair logins is made available to an instance via the instance data service in the metadata item *meta-data/public-keys/0/openssh-key*. This public key data must be added to the root user's *~/.ssh/authorized_keys* file used by `OpenSSH` to store login credentials.

Create a new startup script file in your image at */mnt/ami-fedora7-base/usr/local/sbin/get-credentials.sh* and add the following content to the file:

```
#!/bin/bash
# Retrieve the keypair public key data from the Instance Data service and
# add it to the root user's authorized public keys file.

PUB_KEY_URI=http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key
PUB_KEY_FROM_HTTP=/mnt/openssh_id.pub
ROOT_AUTHORIZED_KEYS=/root/.ssh/authorized_keys

# Make sure the root user's home directory has an .ssh directory for
# the authorized_keys file.
if [ ! -d /root/.ssh ]
then
  mkdir -p /root/.ssh
  chmod 700 /root/.ssh
```


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

```
fi

# Fetch the public key data and save it to a file
echo "Attempting to fetch public keypair"
curl --retry 3 --retry-delay 0 --silent --fail -o $PUB_KEY_FROM_HTTP $PUB_KEY_URI

# Add the public key data to the root user's authorized_keys file if it
# is not already present in this file.
if ! grep -q -f $PUB_KEY_FROM_HTTP $ROOT_AUTHORIZED_KEYS; then
  cat $PUB_KEY_FROM_HTTP >> $ROOT_AUTHORIZED_KEYS
  echo "New KeyPair added to authorized_keys file" | logger -t "ec2"
fi

# Ensure the authorized_keys file has the right permissions
chmod 600 $ROOT_AUTHORIZED_KEYS

# Remove the public key data file
rm -f $PUB_KEY_FROM_HTTP
```

We will also create a script to automatically update the Amazon AMI and API tools to the latest available version. Create the script */mnt/ami-fedora7-base/usr/local/sbin/update-tools.sh* and add the following content:

```
#!/bin/bash

# Update ec2-ami-tools to the latest version.
[ -f ec2-ami-tools.noarch.rpm ] && rm -f ec2-ami-tools.noarch.rpm
echo "Attempting ami-utils update from S3"
(wget http://s3.amazonaws.com/ec2-downloads/ec2-ami-tools.noarch.rpm \
      && echo "Retrieved ec2-ami-tools from S3" \
      || echo "Unable to retreive ec2-ami-tools from S3") \
  | logger -s -t "ec2"
(rpm -Uvh ec2-ami-tools.noarch.rpm \
      && echo "Updated ec2-ami-tools from S3" \
      || echo "ec2-ami-tools already up to date") \
  | logger -s -t "ec2"

# Update EC2 API tools to the latest version
[ -f ec2-api-tools.zip ] && rm -f ec2-api-tools.zip
wget http://s3.amazonaws.com/ec2-downloads/ec2-api-tools.zip -P /root
(cksum ec2-api-tools.zip | diff -q - .ec2-api_latest \
      && echo "API tools are up to date" \
      || (echo "Downloaded latest API tools"; unzip ec2-api-tools.zip; \
      cksum ec2-api-tools.zip > .ec2-api_latest)) \
  | logger -s -t "ec2"
```

Make both the startup scripts executable by the root user as follows:

```
$ chmod u+x /mnt/ami-fedora7-base/usr/local/sbin/*.sh
```

Now make sure that both of these scripts are called when the instance starts by adding the following script fragment to the end of the image's startup file, */mnt/ami-fedora7-base/etc/rc.local*:

```
/usr/local/sbin/get-credentials.sh
/usr/local/sbin/update-tools.sh
```


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

**NOTE**

To avoid having to type these scripts yourself, refer to the online resources for this book to download the script files.

## 6.5.5. Install Additional Software

You may now wish to install additional software so it will be available when you launch instances from your AMI. As a minimum, you should install the Ruby and Java packages; these are required by Amazon's AMI and API tools, respectively.

You have to take special steps to install software on your instance's image file instead of on the underlying system.

### *Installation using yum*

Use the yum configuration file *yum-ami.conf* we defined earlier, and be sure to always include the `--installroot` option pointing to the loop-back mount point */mnt/ami-fedora7-base*.

Here are the commands to install the Ruby and Java packages on your image:

```
$ yum -c yum-ami.conf --installroot=/mnt/ami-fedora7-base install ruby
$ yum -c yum-ami.conf --installroot=/mnt/ami-fedora7-base install java
```

### *Installing by unarchiving*

Be sure to unarchive files into an appropriate location inside the loop-back image, such as the root user's home directory, */mnt/ami-fedora7-base/root/*

### *Installing using RPM*

Run RPM installations using the `chroot` command so the installation will occur in the image instead of in the underlying filesystem. For example, execute the following to manually install an rpm called *example.rpm*:

```
$ /usr/sbin/chroot /mnt/ami-fedora7-base rpm -i example.rpm
```


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

### 6.5.6. Cleanup

When you have finished installing your yum applications, be sure to perform a `clean` operation on the yum cache to minimize the size of your final AMI image file.

```
$ yum -c yum-ami.conf --installroot=/mnt/ami-fedora7-base clean all
```

You have now finished installing and configuring the image. Synchronize the filesystem to ensure all pending data is written to the image file, then unmount the image's *proc* and loop-back mount points.

```
$ sync
$ umount /mnt/ami-fedora7-base/proc
$ umount /mnt/ami-fedora7-base
```

### 6.5.7. Bundle and Register Your AMI

By this stage you will have a disk image file, *ami-fedora7-base.img*, which can be bundled for EC2 using Amazon's image bundling tool, `ec2-bundle-image`. Refer to the prior sections Section 6.3.4" and Section 6.3.5" for more information on the bundling and registration processes.

The only real difference between bundling an image file and an EC2 instance is that you use Amazon's `ec2-bundle-image` tool instead of `ec2-bundle-instance`. Here is a command that will bundle a 32-bit image file, assuming your AWS credential certificate files are available in your home directory.

```
$ ec2-bundle-image \
       --image ~/ami-fedora7-base.img \
       --prefix ami-fedora7-base-i386 \
       --cert ~/cert-ABCDEFGHIJK123ABCDEFGHIJK123ABCD.pem \
       --privatekey ~/pk-ABCDEFGHIJK123ABCDEFGHIJK123ABCD.pem \
       --user 123456789012 \
       --destination /tmp \
       --arch i386
```

Once you have bundled your image file into a sequence of AMI files using this command, upload the files to S3 with the `ec2-upload-bundle` tool, and register the new image with the `ec2-register` command available in Amazon's API tools.

Test your AMI by launching a new instance from it to ensure that you can log in with your keypair and that the instance works as expected. Unless something drastic has gone wrong in the image-creation process, you should be able to start the instance and log in just as you would for the publicly available images.

It is worth putting the instance through its paces to ensure there are no minor problems that will cause difficulties later on. In particular look out for any error messages that occur as the instance loads. If your instance fails to start, or if you are unable to log in once it has started, you can take advantage of the ConsoleOutput API operation to obtain a listing of


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25

the console messages generated while your instance was starting; this may help you to correct the problems.

Good luck!


Programming Amazon Web Services By James Murty ISBN: 9780596515812 Publisher: O'Reilly Media, Inc.
Print Publication Date: 2008/03/25