

Jetty on Amazon Elastic Compute Cloud (Amazon EC2) Tutorial

Jetty on Amazon Elastic Compute Cloud (Amazon EC2) Tutorial

Prerequisites

- EC2 and S3 Account - if you do not have EC2 and S3 account. [click here](#)
- Java Runtime Environment - This document assumes that you have successfully setup Java Runtime Environment with JAVA_HOME set and your PATH includes <java_home>\bin
- SSH Client - you may use putty and winscp for windows users, and openssh for linux and unix users

EC2 Environment Setup and Simple Instance Startup

The video tutorial below is all you need to know in starting to use EC2

<http://developer.amazonwebservices.com/connect/entry.jspa?entryID=583>

Flash Version from amazon site: [Watch Now](#)

Windows Media format Version from amazon site: [Download Now](#)

Creating A Custom Amazon AMI Image

Here are the steps you need to create your FC6 image. Two notes before getting started: 1) I am using an FC6 box to run the following commands on so your luck may vary with older system and 2) Some of these can be done as a non-root user but you might as well be root for all of them.

If you are in a hurry you may download all of the following steps in a single script that will generate the custom bootable AMI.

1) Create the image file and initialize the filesystem on it (note that I'm only making giving myself 1G of space for this install, if you think you will need more room you should create a larger file by changing the seek value):

```
dd if=/dev/zero of=fc6-i386.img bs=1M
count=1 seek=2048
/sbin/mke2fs -F -j fc6-i386.img
```

2) Mount the file with a loopback device:

```
mount -o loop fc6-i386.img /mnt
```

3) Create base directories and device files:

```
mkdir /mnt/dev
mkdir /mnt/proc
mkdir /mnt/etc
for i in console null zero ; do
/sbin/MAKEDEV -d /mnt/dev -x $i ; done
```

4) Create the initial fstab [file:

]

```
cat <<EOL > /mnt/etc/fstab
/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 1 2
/dev/sda3 swap swap defaults 0 0
EOL
```

5) Mount the proc under the new root filesystem so yum will work correctly:

```
mount -t proc none /mnt/proc
```

6) Create your a yum configuration file:

```
cat <<EOL > /tmp/yumec2.conf
[main]
cachedir=/var/cache/yum
debuglevel=2
logfile=/var/log/yum.log
exclude=*-debuginfo
gpgcheck=0
obsoletes=1
reposdir=/dev/null

[base]
name=Fedora Core 6 - i386 - Base
mirrorlist=http://fedora.redhat.com/download
/mirrors/fedora-core-6
enabled=1

[updates-released]
name=Fedora Core 6 - i386 - Released Updates
mirrorlist=http://fedora.redhat.com/download
/mirrors/updates-released-fc6
enabled=1
EOL
```

7) Run yum to install the base group of packages to your root filesystem (this may take some time but you should see it progress, I have had all kinds of trouble with yum in the past so if it hangs you may want to kill it and try again):

```
yum -c /tmp/yumec2.conf --installroot=/mnt  
-y groupinstall Base
```

8) Clean the yum cache:

```
yum -c /tmp/yumec2.conf --installroot=/mnt  
-y clean packages
```

9) Move the TLS directory out of the way:

```
mv /mnt/lib/tls /mnt/lib/tls-disabled
```

10) Modify the boot script to download your SSH key and stick it in root's directory:

```
cat <<EOL >> /mnt/etc/rc.local
if [ ! -d /root/.ssh ] ; then
mkdir -p /root/.ssh
chmod 700 /root/.ssh
fi
# Fetch public key using HTTP
curl
http://169.254.169.254/1.0//meta-data/public
-keys/0/openssl >
/tmp/my-key
if [ $? -eq 0 ] ; then
cat /tmp/my-key >>
/root/.ssh/authorized_keys
chmod 600 /root/.ssh/authorized_keys
rm /tmp/my-key
fi
# or fetch public key using the file in the
ephemeral store:
if [ -e /mnt/openssh_id.pub ] ; then
cat /mnt/openssh_id.pub >>
/root/.ssh/authorized_keys
chmod 600 /root/.ssh/authorized_keys
fi
EOL
```

11) Set sshd to allow remote root connections and now hang on DNS problems:

```
cat <<EOL >> /mnt/etc/ssh/sshd_config
UseDNS no
PermitRootLogin without-password
EOL
```

12) Create the networking scripts:

```
cat <<EOL > /mnt/etc/sysconfig/network
NETWORKING=yes
HOSTNAME=localhost.localdomain
EOL

cat <<EOL >
/mnt/etc/sysconfig/network-scripts/ifcfg-eth
0
ONBOOT=yes
DEVICE=eth0
BOOTPROTO=dhcp
EOL
```

13) Sync and umount your root filesystem:

```
sync
umount /mnt/proc
umount /mnt
```

You have now created your very own bootable AMI. If you want to fiddle with it from this point you may continue to use the yum command as in the above examples or you can also remount the filesystem and chroot to it using a command like this:

```
mount -o loop fc6-i386.img /mnt
mount -t proc none /mnt/proc
chroot /mnt /bin/sh
```

now you can install java, svn, cvs, jetty and etc

One thing to remember if you use chroot like this is that everything is local now. You will want to mount the proc filesystem and probably add entries to /etc/resolve.conf so any hostnames you try to resolve will work.

The next step is to get the AMI to S3 so that it can be booted.

Bundling an AMI

A root file system image needs to be bundled as an AMI in order to be used with the Amazon EC2 service. The bundling process first compresses the image to minimize bandwidth usage and storage requirements. The compressed image is then encrypted and signed to ensure confidentiality of the data, and authentication against the creator. The encrypted image is finally split into manageable parts for upload. A manifest file is created containing a list of the image parts with their checksums. This chapter provides an overview of the AMI tools that automate this process and some examples of their use.

The AMI tools are three command-line utilities:

1. `ec2-bundle-image` bundles an existing AMI
2. `ec2-bundle-vol` creates an AMI from an existing machine or installed volume
3. `ec2-upload-bundle` uploads a bundled AMI to S3 storage

Installing the AMI Tools

The AMI tools are packaged as an RPM suitable for running on Fedora Core 3/4 with Ruby 1.8.2 (or greater) installed. On Fedora Core 4 Ruby can be installed by following the steps below. You will need root privileges to install the software. You can find the AMI tools RPM from our public S3 downloads bucket.

First install Ruby using the yum package manager.

```
yum install ruby
```

Install the AMI tools RPM.

```
rpm -i ec2-ami-tools-x.x-xxxx.i386.rpm
```

Installation Issues

The AMI tools libraries install under `/usr/lib/site_ruby`. Ruby should pick up this path automatically, but if you see a load error when running one of the AMI utilities, it may be because Ruby isn't looking there. To fix this, add `/usr/lib/site_ruby` to Ruby's library path, which is set in the `RUBYLIB` environment variable.

Documentation

The manual describing the operation of each utility can be displayed by invoking it with the `--manual` parameter. For example:

```
ec2-bundle-image --manual
```

Invoking a utility with the `--help` parameter displays a summary and list of command line parameters. For example:

```
ec2-bundle-image --help
```

Using the AMI Tools in Bundling an Image

Once a machine image has been created it must be bundled as an AMI for use with Amazon EC2, as follows. Use `ec2-bundle-image` to bundle an image that you have prepared in a loopback file, as described in the previous section.

```
ec2-bundle-image -i my-image.img -k  
private_key.pem -c certificate.pem -u  
12345678
```

This will create the bundle files:

```
image.part.00  
image.part.01  
...  
image.part.NN  
image.manifest.xml
```

Uploading a Bundled AMI

The bundled AMI needs to be uploaded for storage in Amazon S3 before it

can be accessed by Amazon EC2. Use `ec2-upload-bundle` to upload the bundled AMI that you created as described above. S3 stores data objects in buckets, which are similar in concept to directories. Buckets must have globally unique names. The `ec2-upload-bundle` utility will upload the bundled AMI to a specified bucket. If the specified bucket does not exist it will be created. However, if the specified bucket already exists, and belongs to another user, then `ec2-upload-bundle` will fail.

```
ec2-upload-bundle -b my-bucket -m  
image.manifest.xml -a my-aws-access-key-id  
-s my-secret-key-id
```

And Register as an AMI Bundle

```
ec2-register my-bucket/image.manifest.xml
```

your bundle is now ready to use