

OPENSOLARIS™ OPERATING SYSTEM GETTING STARTED GUIDE FOR AMAZON EC2

August 2008

Table of Contents

OpenSolaris™ Operating System Getting Started Guide for Amazon EC2	1
About the AMI/Application Programming Interface (API) and command line tools . .	1
Prerequisites	1
Downloading and using the AMI/API tools	2
About the Solaris AMI	3
Launching Instances	3
Rebundling and Uploading OpenSolaris Images on Amazon EC2	5
Rebundling an OpenSolaris 2008.05 based AMI	6
Setting up the Environment for EC2 Re-Bundling	8
Bundling the Image	9
Uploading the Bundle to an EC2 Bucket	9
Registering an AMI	10
Verifying the Image Attributes	10
Launching AMI and Verifying the Re-bundled Machine Image	11
Rebundling a SXCE Build 79 based AMI	11
Basic steps	11
Example	12
Known Limitations and Important Notes	15
References	16

OpenSolaris™ Operating System Getting Started Guide for Amazon EC2

Sun and Amazon Web Services are opening a private beta program starting on May 5, 2008. Approved beta users get access to OpenSolaris™ operating system (OS) at <http://www.opensolaris.org/> on Amazon Elastic Compute Cloud (EC2). OpenSolaris on Amazon EC2 is an Amazon Web service that incorporates hardware virtualization technologies based on the Sun™ xVM software and the Xen open source community work. Information about Amazon EC2 is located at: <http://aws.amazon.com/ec2>.

About the AMI/Application Programming Interface (API) and command line tools

The Amazon EC2 AMI tools are command-line utilities to help bundle an Amazon Machine Image (AMI), create an AMI from an existing machine or installed volume, and upload a bundled AMI to Amazon S3 (Simple Storage Service).

AMI Tools can be downloaded from: <http://developer.amazonwebservices.com/connect/entry.jspa?externalID=368&categoryID=88>.

The Amazon EC2 API command line tools serve as the client interface to the Amazon EC2 Web service. These tools are employed to register and launch instances, manipulate security groups, and more. The tools are the primary interface to Amazon EC2 services.

Amazon EC2 API command line tools can be downloaded from:
<http://developer.amazonwebservices.com/connect/entry.jspa?externalID=351&categoryID=88>.

Amazon EC2 API documentation is available from:
<http://developer.amazonwebservices.com/connect/entry.jspa?externalID=1344&categoryID=118>.

Prerequisites

This section describes how to setup the necessary tools/utility with the appropriate version, including environment variables that are needed for establishing a connection to Amazon EC2 services.

In order to access the Amazon EC2 environment, it is required to have an SSH client and appropriate Java™ Runtime Environment (JRE) software. This guide was developed using Solaris clients - for other clients (for example, Windows or Linux), refer to the Amazon EC2 Web site. If using a Solaris client to access the Amazon EC2 Web Service, an SSH

client is already installed. If using a Solaris 10 system as client, the appropriate Java runtime is already installed as well.

Set the `JAVA_HOME` environment variable as shown below (which might vary depending on the shell).

```
bash # export JAVA_HOME=/usr/jdk/latest
```

Test the `JAVA_HOME` path by running following command:

```
bash # $JAVA_HOME/bin/java -version
```

The output should be something similar to the following:

```
java version "1.6.0_04"  
Java(TM) SE Runtime Environment (build 1.6.0_04-b12)  
Java HotSpot(TM) Server VM (build 10.0-b19, mixed mode)
```

If planning to use any other client to connect to Amazon EC2 services, follow the instructions in the Amazon Getting Started Guide available from:

<http://docs.amazonwebservices.com/AWSEC2/2008-02-01/GettingStartedGuide/prerequisites.html>.

Downloading and using the AMI/API tools

The AMI/API command line tools can be downloaded from the following location:

http://www.amazon.com/gp/redirect.html/ref=aws_rc_ec2tools?location=http://s3.amazonaws.com/ec2-downloads/ec2-api-tools.zip&token=A80325AA4DAB186C80828ED5138633E3F49160D9.

After completing the download, follow the instructions in the Amazon Getting Started Guide for setting up the tools to make it available or usable. The guide is available from: <http://docs.amazonwebservices.com/AWSEC2/2008-02-01/GettingStartedGuide/setting-up-your-tools.html>.

The following is an example of setting up the tools.

Assume that the API tools (command line tools) were downloaded to the `/export/home/ec2user` directory:

```
bash # unzip ec2-api-tools.zip  
bash # export EC2_HOME=/export/home/ec2user/ec2-api-tools-1.3-19403  
bash # export PATH=$PATH:$EC2_HOME/bin
```

The following variables point to the Amazon supplied private key and certificate files:

```
bash # export EC2_PRIVATE_KEY=~/.ec2/<YOUR-PK-PEM-FILE>.pem
bash # export EC2_CERT=~/.ec2/<YOUR-CERT-PEM-FILE>.pem
```

Set the EC2 URL to the right server

```
bash # export EC2_URL=http://ec2.amazonaws.com/
```

If you are using a proxy server for Internet connection, then add the following environment variable:

```
bash # export EC2_JVM_ARGS="-Dhttps.proxyHost=web-proxy -
Dhttps.proxyPort=8080"
```

Amazon provides a Firefox extension “ElasticFox” for interacting with Amazon EC2 using the Firefox browser. You can find more information about this extension below:

<http://developer.amazonwebservices.com/connect/entry.jspa?externalID=609>.

About the Solaris AMI

The OpenSolaris AMI is a prepackaged image for running OpenSolaris on Amazon EC2. The AMI includes the tools and utilities required to bring up a Solaris instance.

To obtain the correct AKI/ARI/AMI ID and for general or technical questions, email: ec2-solaris-support@sun.com.

Launching Instances

Details on running an instance can be found in the Amazon Getting Started Guide at: <http://docs.amazonwebservices.com/AWSEC2/2008-02-01/GettingStartedGuide/running-an-instance.html>.

After creating an account, performing the key pair setup, and setting up the correct version of the JRE and the SSH client, the next step is to launch an OpenSolaris instance on Amazon EC2.

A typical session involving performing the steps above would be on an OpenSolaris system, as illustrated below. The command prompt might be different depending on the system. Note that `grep (1M)` is used to reduce the output.

1. Find a suitable AMI.

```
bash # ec2-describe-images | grep b79alatest.img
IMAGE ami-8f36d3e6 ec2sun/b79alatest.img.manifest.xml <999999999999>
available private i386 machine
```

2. Generate a key pair.

```
bash # ec2-add-keypair mykeypair
```

3. Save the private key returned by the command in a local file so that it can be used later. Using a text editor, create a file named *mykeypair* and paste everything between (and including) the "`—BEGIN RSA PRIVATE KEY—`" and "`—END RSA PRIVATE KEY—`" lines into it. The file can be saved in any directory, but if it is not saved in the current directory, it is necessary to specify the full path when using the SSH command below.

4. Change the permissions on the file as follows.

```
bash # chmod 600 mykeypair
```

5. Run an instance.

```
bash # ec2-run-instances ami-8f36d3e6 -k mykeypair
RESERVATION r-e75bad8e <999999999999> default
INSTANCE i-a338c6ca ami-8f36d3e6 pending mykeypair 0 m1.small 2008-
02-27T03:55:24+0000

-- Wait a few minutes for startup --

bash # ec2-describe-instances | grep i-a338c6ca
INSTANCE i-a338c6ca ami-8f36d3e6 ec2-72-44-33-244.compute-
1.amazonaws.com ip-10-251-73-175.ec2.internal running mykeypair 0
m1.small 2008-02-27T03:55:24+0000 aki-d636d3bf ari-a936d3c0

-- Wait a few minutes for boot --
```

Here are the key parameters or options in the example command above:

```
Amazon account no: <999999999999>
AMI ID: ami-8f36d3e6
AKI ID: aki-d636d3bf
ARI ID: ari-a936d3c0
KeyPair: mykeypair
```

6. Authorize network access to the instances.

```
bash # ec2-authorize default -p 22
PERMISSION    default  ALLOWS  tcp    22    22    FROM    CIDR
0.0.0.0/0
bash # ec2-authorize default -p 80
PERMISSION    default  ALLOWS  tcp    80    80    FROM    CIDR
0.0.0.0/0
```

7. Connect to the Amazon EC2 OpenSolaris instance.

```
bash # ssh -i $HOME/.ssh/mykeypair -l root ec2-72-44-33-244.compute-
1.amazonaws.com
The authenticity of host 'ec2-72-44-33-244.compute-1.amazonaws.com
(72.44.33.244)' can't be established.
RSA key fingerprint is 19:d3:d5:52:b4:08:77:00:d0:12:41:6a:5c:bd:cb:d2.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-72-44-33-244.compute-
1.amazonaws.com,72.44.33.244' (RSA) to the list of known hosts.
Last login: Mon Jan 21 04:50:15 2008
Sun Microsystems Inc. SunOS 5.11 snv_79 January 2008

or the same can be done as below :
bash # ssh -i $HOME/.ssh/mykeypair root@ec2-72-44-33-244.compute-
1.amazonaws.com
```

8. Terminate an instance when finished using it.

```
bash # ec2-terminate-instances i-a338c6ca
INSTANCE i-a338c6ca running shutting-down
bash #
```

Rebundling and Uploading OpenSolaris Images on Amazon EC2

For generic instructions on Amazon EC2 bundling instructions refer to the Amazon Getting Started Guide at: <http://docs.amazonwebservices.com/AWSEC2/2008-02-01/GettingStartedGuide/creating-an-image.html>.

Rebundling is a process of creating a customized image based on generic Amazon EC2 offering. This might include additional software packages and any configuration changes to the running instance of a unique Amazon EC2 environment. This image can be used as a prototype image, enabling the user to launch an environment later based on this image.

Important Note: Rebundling instructions are different for SXCE.79 and OpenSolaris 2008.05-based AMIs. Please refer to their respective sections for more information.

Rebundling an OpenSolaris 2008.05 based AMI

Rebundling a running instance on Amazon EC2 can be achieved using a number of OpenSolaris 2008.05 utilities to create a working ZFS based file system bundle. The rebundling process takes a large amount of space and time and must be done with caution. The **ec2-bundle-image** command creates a set of files that consume additional space.

1. Create an empty image file that will serve as a destination or output file for the re-silvering process. Use a suitable name for this image.

```
bash # dd if=/dev/zero of=/mnt/your-bundle.img bs=1k seek=10000k count=1
```

2. Mount this empty image file as lofi device.

```
bash # lofiadm -a /mnt/your-bundle.img
```

Set the appropriate variable to the right values needed for rebundling:

3. Set the `rpool_addr` variable

```
bash # rpool_addr=$(echo "::spa " | mdb -k | grep "rpool" | awk '{print $1}')
```

4. Set the `offset` variable

```
bash # offset=$(echo "::offsetof struct spa spa_is_root" | mdb -k | awk '{print $NF}')
```

5. Now, verify that the values of the `rpool_addr` and `offset` variables are set correctly - here is a sample output

```
bash # echo $rpool_addr
d1095b00
bash # echo $offset
0x3d0
```

6. Set the `is_root` value for this pool to the correct value as follows:

```
bash # echo "$rpool_addr+$offset/Z 0" | mdb -kw
```

7. After all the variables have been set, initialize the re-silvering process by attaching the storage pool to the correct lofi device that you created earlier

```
bash # zpool attach rpool c4d0s0 /dev/lofi/1
```

Since the re-silvering process might take some time, it is important to check the pool status after every few minutes. The following command can be executed repeatedly to check the rpool status and verify if the re-silvering is completed. It is followed by a sample output. Notice the “to go” field, which estimates the time this process may take. The time may fluctuate but the overall process should take approximately fifteen minutes and can vary based on the content or size of your domU.

Following is sample output displaying the resilvering process progress:

```
bash # zpool status rpool
pool: rpool
state: ONLINE
status: One or more devices is currently being resilvered. The pool
will continue to function, possibly in a degraded state.
action: Wait for the resilver to complete.
scrub: resilver in progress for 0h1m, 9.65% done, 0h15m to go
config:

    NAME                STATE      READ WRITE CKSUM
    rpool                ONLINE    0     0     0
      mirror            ONLINE    0     0     0
        c4d0s0          ONLINE    0     0     0
        /dev/lofi/1     ONLINE    0     0     0

errors: No known data errors
```

```
bash # zpool status rpool

pool: rpool
state: ONLINE
scrub: resilver completed after 0h9m with 0 errors on Tue Jul 15
09:55:02 2008
config:

    NAME                STATE      READ WRITE CKSUM
    rpool                ONLINE    0     0     0
      mirror            ONLINE    0     0     0
        c4d0s0          ONLINE    0     0     0
        /dev/lofi/1     ONLINE    0     0     0

errors: No known data errors
```

You must wait for this to complete for successful re-bundling. For any reason, if you interrupt or continue before its completed it will result into an unusable image.

8. After the re-silvering is completed, we can offline the mounted image with the following command:

```
bash # zpool offline rpool /dev/lofi/1
```

9. Next, duplicate the label on the newly create image as follows

```
bash # /opt/ec2/sbin/ec2-zdb.32 -D /dev/dsk/c4d0s0 /mnt/your-bundle.img
```

10. Detach the lofi device /dev/lofi/1 from rpool since it is no longer needed

```
bash # zpool detach rpool /dev/lofi/1
```

11. Finally, unmount the image with the following command:

```
bash # lofiadm -d /dev/lofi/1
```

Setting up the Environment for EC2 Re-Bundling

The following environment variables assist in the bundling and upload process with the API tools:

```
export BUCKET=<BUCKET NAME>
export JAVA_HOME=/usr/jdk/latest
export EC2_HOME=/opt/ec2
export PATH=$PATH:$EC2_HOME/bin
export RUBYLIB=$EC2_HOME/lib
export EC2_URL=https://ec2.amazonaws.com
export EC2_PRIVATE_KEY=/mnt/keys/<PRIVATE KEY FILE>
export EC2_CERT=/mnt/keys/<CERT FILE>
export EC2_KEYID=<KEY ID>
export EC2_KEY=<KEY>
export DIRECTORY=/mnt
```

Create the `parts` and `keys` directories inside the `/mnt` directory.

```
bash # mkdir $DIRECTORY/parts
bash # mkdir $DIRECTORY/keys
```

Secure copy the keys on `/mnt/keys`

```
bash # scp -i <PATH TO your-keypair file> <PATH TO .pem FILES>/*.pem
<your instance public dns name>:/mnt/keys/
```

Note: It is important that the key and certificate files are uploaded into `/mnt/keys` to prevent them being bundled with the new AMI.

Once you have completed the previous steps, a system snapshot needs to be created and packaged into an AMI by using the **ec2-bundle-image** utility.

ec2-bundle-image encrypts and signs the image to ensure it cannot be tampered with and that only you and Amazon EC2 can decrypt it.

Bundling the Image

At this point, the OpenSolaris 2008.05 machine image has been modified and the private key and X.509 certificates are uploaded. The AMI can now be bundled, using your *AWS account ID* as your username (*not* your AWS Access Key ID).

Also, note that `$EC2_CERT` and `$EC2_PRIVATE_KEY` are environment variables set to point at the Amazon EC2 certificate and private key file.

```
bash # ec2-bundle-image -c $EC2_CERT -k $EC2_PRIVATE_KEY \
--kernel aki-b78367de --ramdisk ari-b68367df \
--block-device-mapping "root=rpool/114@0,ami=0,ephemeral0=1" \
--user <userid> --arch i386 \
-i $DIRECTORY/your-bundle.img -d $DIRECTORY/parts
```

Uploading the Bundle to an EC2 Bucket

It is important to make sure that you select the right bucket to where you will be uploading the image. This will ensure that you don't accidentally overwrite the contents of a previously uploaded image.

```
bash # echo $BUCKET
```

All AMIs are loaded from Amazon S3 storage. The newly bundled AMI needs to be uploaded to an existing account on Amazon S3.

Amazon S3 stores data objects in buckets, which are similar in concept to directories. You'll need to specify a bucket name in the following example as `<your-s3-bucket>`. Buckets have globally unique names and are owned by unique users. If you have used S3 before, you can use any of your existing buckets or just give **ec2-upload-bundle** any

name that makes sense to you. The **ec2-upload-bundle** utility will upload the bundled AMI to a specified bucket. If the specified bucket does not exist it will create it. If the specified bucket belongs to another user, **ec2-upload-bundle** will fail, and you will have to try a different name.

For this step you'll need your AWS Access Key ID (<aws-access-key-id>) and AWS Secret Access Key (<aws-secret-access-key>). For information on how to find these keys, see [Signing up for Amazon S3](http://docs.amazonwebservices.com/AmazonEC2/gsg/2006-06-26/) (<http://docs.amazonwebservices.com/AmazonEC2/gsg/2006-06-26/>).

EC2_KEYID and EC2_KEY variables are the Access Key ID and Secret Access Key respectively used with S3.

The upload process can be quite lengthy, but you will get continuous feedback until the upload has completed as shown in the following example.

```
bash # ec2-upload-bundle -b $BUCKET -m your-bundle.img.manifest.xml \  
> --url http://s3.amazonaws.com \  
> --retry -a $EC2_KEYID -s $EC2_KEY
```

Registering an AMI

Your image must be registered with Amazon EC2, so you can locate it and run instances based on it. Moreover, if you make any changes to the source image stored in Amazon S3, you must re-register the image.

Execute the following command to register the AMI you uploaded to Amazon S3:

```
bash # ec2reg -C $EC2_CERT -K $EC2_PRIVATE_KEY $BUCKET/your-  
bundle.img.manifest.xml
```

Note: The name of the bundle image manifest file will change in your case.

Verifying the Image Attributes

ec2-describe-image-attribute can be used to get more details about the kernel, ramdisk, launch permissions and other details of an AMI.

```
bash # ec2-describe-image-attribute [--kernel or --ramdisk or -l] <ami-  
id>
```

Launching AMI and Verifying the Re-bundled Machine Image

You can now run an instance of the modified AMI by specifying the image identifier you received when you registered the image.

```
bash # ec2-run-instances <ami from ec2reg> -k your-keypair
```

The following command should list the details of the AMI that you just launched. It will list the instance id, public and private DNS names, your key-pair, instance state, etc.

```
bash # ec2-describe-instances | grep your-keypair
```

Rebundling a SXCE Build 79 based AMI

Rebundling a running instance on Amazon EC2 can be achieved using the `vbdcfg` script in `/opt/ec2/bin` in the Solaris AMI. The script uses a number of OpenSolaris utilities to create a working file system bundle. The rebundling process takes a large amount of space and time. Two disk images are created and a system flash archive (known as a *flar archive*) are generated. On top of that, the `ec2-bundle-image` command creates a set of files that consume additional space.

The default OpenSolaris instance has an additional disk available in the configuration. It is recommended to use this for the additional space.

Basic steps

1. On a fresh OpenSolaris instance, apply all of the changes required for the new image. For example, install packages and configure any required files and services.
2. On the running instance, create a *flar archive* using the OpenSolaris `flar` command. Sun recommends performing all of the rebundling operations in a directory on an additional filesystem.
Direct `flar` to omit that directory from archiving.

```
flar create -n <name> -x <build directory> <build directory>/sys.flar
```

The `-x` omits the *<build directory>* from being added to the flar. The `flar` command must be run by root.

3. The Amazon AMI and API tools are located in `/opt/ec2/bin`. Add that directory to `PATH`.

```
bash # PATH=$PATH:/opt/ec2/bin
bash # export PATH
```

4. Use the customized program, `vbdcfg` (located in `/opt/ex2/bin`), which assists in creating AMIs for use on Amazon EC2.

```
bash # vbdcfg mkproto -f -P <build directory>/xc/proto \  
-a <build directory>/sys.flar <proto name>
```

This creates the proto in the build directory using the files from the flar archive.

5. Use `vbdcfg` to build the image for Amazon EC2.

```
bash # vbdcfg mkdomU -f -P <build directory>/xc/proto \  
-R <build directory>/xc/xvm <proto name> <image name>
```

6. Bundle the image file that is located in:

```
<build directory>/xc/xvm/<image name>
```

7. Upload and run the AMI. See the example below for workarounds to known issues with bundling and uploading images to EC2.

Example

Here is an example that follows the steps above. The shell variable `BUILDDIR` is used to make the directory structure clear. In the instance, an additional disk exists. The `codiso` partition should be available to build a filesystem.

```
bash # newfs /dev/dsk/c0d1s0
```

If that fails, `format` will need to be run to `fdisk` the virtual disk and to partition the disk.

In this example, `/mnt` is used as the mount point.

```
bash # mount /dev/dsk/c0d1s0 /mnt
```

Create the required directories.

```
bash # BUILDDIR=/mnt/build  
bash # mkdir $BUILDDIR  
bash # mkdir $BUILDDIR/myecimage  
bash # mkdir $BUILDDIR/xc  
bash # mkdir $BUILDDIR/xc/proto  
bash # mkdir $BUILDDIR/xc/xvm
```

Adjust the environment for access to the AMI tools.

```
bash # PATH=$PATH:/usr/sfw/bin:/opt/ec2/bin
bash # JAVA_HOME=/usr/jdk/latest
bash # EC2_HOME=/opt/ec2
bash # RUBYLIB=$EC2_HOME/lib
bash # export JAVA_HOME EC2_HOME RUBYLIB
```

The following environment variables assist in the bundling and upload process with the API tools.

```
EC2_URL=https://ec2.amazonaws.com
EC2_PRIVATE_KEY=<path to>/<keyfile>.pem
EC2_CERT=<path to>/<certfile>.pem
EC2_KEYID=<keyid>
EC2_KEY="<private key>"
```

Create the flar archive omitting the directory used to build.

```
bash # flar create -n myFlar -x $BUILDDIR $BUILDDIR/sys.flar
```

Use `vbdcfg` to create the `root.file` archive.

```
bash # vbdcfg mkproto -f -P $BUILDDIR/xc/proto -a $BUILDDIR/sys.flar \
myProto
bash # vbdcfg mkdomU -f -P $BUILDDIR/xc/proto \
-R $BUILDDIR/xc/xvm myProto myImage
```

The `root.file` archive is now in the directory structure.

```
bash # $BUILDDIR/xc/xvm/myImage/root.file
```

Bundling the image can be performed using a command similar to the following, which uses the assigned kernel and ramdisk (*aki-d636d3bf* and *ari-a936d3c0*) for 32-bit.

```
bash # ec2-bundle-image -c $EC2_CERT -k $EC2_PRIVATE_KEY \
--kernel aki-d636d3bf --ramdisk ari-a936d3c0 \
--block-device-mapping "ami=0,ephemeral0=1,root=/dev/dsk/c0d0s0" \
--user <userid> -i root.file -d $BUILDDIR/myecimage
```

Note that `$EC2_CERT` and `$EC2_PRIVATE_KEY` are environment variables set to point at the Amazon EC2 certificate and private key files. Also note that there will be a tar error, which can be ignored.

Here is an example for 64-bit.

```
bash # ec2-bundle-image -c $EC2_CERT -k $EC2_PRIVATE_KEY \  
  --block-device-mapping  
"ami=0,ephemeral0=1,ephemeral1=2,ephemeral2=3,ephemeral3=4,root=/dev/  
dsk/c0d0s0" \  
  --kernel aki-ab3cd9c2 --ramdisk ari-2838dd41 \  
  --user <userid> -i root.file -d $BUILDDIR/myecimage
```

The rest of the steps for running the instance are as follows.

```
bash # cd $BUILDDIR/myecimage  
  
bash # ec2-upload-bundle -b <bucket> -m root.file.manifest.xml \  
  --url http://s3.amazonaws.com --retry \  
  -a $EC2_KEYID -s $EC2_KEY
```

In case of an upload bundle failure due to time differences, it might be necessary to properly set the system clock. The `date` command or the following can be used to reset the clock using an open ntp server.

```
bash # ntpdate 0.north-america.pool.ntp.org
```

Start the instance.

```
bash # ec2reg <bucket>/root.file.manifest.xml  
bash # ec2-run-instances --key <keypair> --ramdisk ari-a936d3c0 \  
  --kernel aki-d636d3bf <ami from ec2reg>
```

Known Limitations and Important Notes

Following are some of the known limitations of OpenSolaris 2008.05 based AMI:

1. `pkg image-update` - This command is currently not supported on Amazon EC2 since it modifies the kernel and ramdisk files resulting in non-bootable AMI. As we know, in the EC2 environment modifying the kernel and ramdisk is not permitted. In certain cases, if the user wants to enable this command, then the user can edit the `/usr/bin/pkg` file appropriately.
2. `shutdown(1M)` - Issuing “`shutdown -i6`” on an OpenSolaris instance on EC2 will not reboot but terminate the instance. However, the `reboot(1M)` command can be used to reboot the running instance.
3. Automatic mount of second disk - Due to rebundling requirements and easier management of second disk, the second disk will be mounted on `/mnt`. This can be verified using the “`zpool list`” command.
4. Timezone - When you launch an instance based on OpenSolaris, the timezone, by default, is set to PST (Pacific Standard Time). To change the timezone, edit `/etc/TIMEZONE` with the proper TZ entry. See the `timezone(4)` man page for pointers to valid TZ entries and `TIMEZONE(4)` manpage for more information on the `/etc/TIMEZONE` file.
5. Operating System version - The output of `uname(1)` command or content of `/etc/release` will show the operating system version as “OpenSolaris 2008.11 snv_91 X 86”. This is due to the fact that the AMI has been updated to include bug fixes and feature enhancements.

References

OpenSolaris Community Documentation: <http://opensolaris.org/os/community/documentation>

Amazon EC2 Web Services: http://www.amazon.com/EC2-AWS-Service-Pricing/b/ref=sc_fe_L_2?ie=UTF8&node=201590011&no=3435361&me=A36L942TSJ2AJA

Solaris 10 Reference Manual Collection: <http://docs.sun.com/app/docs/coll/40.10>

Commands References:

- [grep\(1\)](#)
- [export\(1\)](#)
- [flar\(1M\)](#)
- [chmod\(1\)](#)
- [newfs\(1M\)](#)
- [mount\(1M\)](#)
- [mkdir\(1\)](#)
- [zpool\(1M\)](#)
- [lofiadm\(1M\)](#)
- [mdb\(1\)](#)
- [awk\(1\)](#)
- [dd\(1M\)](#)
- [scp\(1\)](#)
- [zdb\(1M\)](#)

Video: Setting up and Running Amazon EC2 from Windows:

<http://developer.amazonwebservices.com/connect/entry.jspa?externalID=583&categoryID=100>

Video: Create Your Own Customized AMI: <http://developer.amazonwebservices.com/connect/entry.jspa?externalID=938&categoryID=100>

