



## Getting Started with Amazon EC2 Management in Eclipse

---

## Table of Contents

Introduction.....	4
Installation.....	4
Prerequisites .....	4
Installing the AWS Toolkit for Eclipse .....	4
Retrieving your AWS Credentials .....	5
Configuring the Toolkit with Your Credentials.....	5
Adding X.509 Certificates .....	6
Deploying a Sample Application to an Amazon EC2 Tomcat Cluster .....	7
Creating a Simple “Hello world” Web Application and Testing it Locally .....	7
Running our Servlet on an Amazon EC2 Tomcat Cluster .....	12
Defining the Cluster .....	13
Securing the Tomcat Amazon EC2 Cluster.....	15
Running the Sample Program on an Amazon EC2 Cluster.....	19
Debugging Applications .....	21
Examining and Monitoring Amazon EC2 Instances .....	23
Examining Amazon EC2 Instances.....	23
Connecting to EC2 Instances.....	23
Enabling HAProxy Monitoring.....	24
Viewing the HAProxy Statistics Dashboard.....	25
Using the EC2 AMIs View.....	27
Opening the EC2 AMIs View .....	27
Finding and Launching an AMI.....	27
Configuring AMI Instances .....	28
Selecting the Number of Instances.....	30
Selecting the Instance Type .....	30
Choosing an Availability Zone.....	31
Configuring Security.....	32
Launching the AMI Instance .....	32
Persisting Amazon EC2 Instance Data .....	33
Bundling an Amazon EC2 Instance .....	34
Creating an Amazon Simple Storage Service Bucket .....	34
Bundling an Amazon EC2 instance.....	35
Testing the Bundle .....	36
Limitations of Bundling .....	36
Using Amazon Elastic Block Storage .....	37
Creating an Amazon EBS Volume .....	37
Creating an Amazon EBS Snapshot.....	39
Using an Amazon EBS Snapshot .....	41

Conclusion .....	42
------------------	----

## Introduction

The Amazon Web Services (AWS) Toolkit for Eclipse integrates with Eclipse's Web Tools Platform (WTP) to enable developers to develop, debug, manage, and deploy Amazon Web Services applications from within the Eclipse integrated development environment.

This guide will help you get started with the EC2 features of the AWS Toolkit for Eclipse.

## Installation

### Prerequisites

The SimpleDB Management feature is just a tool that enables easy manipulation of your SimpleDB content. You will get the most out of it if you understand the fundamental SimpleDB concepts. For more information, see the [Amazon SimpleDB Getting Started Guide](#).

The AWS Toolkit for Eclipse has the following requirements:

- Requires [Java 1.5](#) or higher.
- Requires Eclipse IDE for Java Developers 3.4 or higher. [Eclipse IDE for Java EE Developers 3.5](#) is recommended.
- For Amazon EC2 Management:
  - [Eclipse Web Tools Platform](#) 2.0 or higher, based on the requirements of your Eclipse distribution. The Eclipse IDE for Java EE Developers 3.4 has the Web Tools Platform pre-installed.
  - You must sign up for [Amazon EC2](#).
- For Amazon SimpleDB Management:
  - Requires Eclipse Data Tools Platform 1.7 or higher. If you use Eclipse 3.4, you will need to install it from <http://download.eclipse.org/datatools/updates/>.
  - You must be signed up for [Amazon SimpleDB](#).

### Installing the AWS Toolkit for Eclipse

Download and install the AWS Toolkit for Eclipse using the following Eclipse Update Site:

<http://aws.amazon.com/eclipse/>

#### To install on Eclipse 3.5 (Gallileo)

1. Open **Help** -> **Install New Software....**
2. Click the **Add** button. The Add Site dialog box opens.
3. Enter "AWS Toolkit for Eclipse" in the Name box, and "http://aws.amazon.com/eclipse/" in the Location box, then click **OK**. The update site is now listed in the Available Software Sites list.
4. Select the new "AWS Toolkit for Eclipse" update site from the **Work with:** menu.
5. Select all components of the archive, then click **Next....** Eclipse guides you through the remaining installation steps.

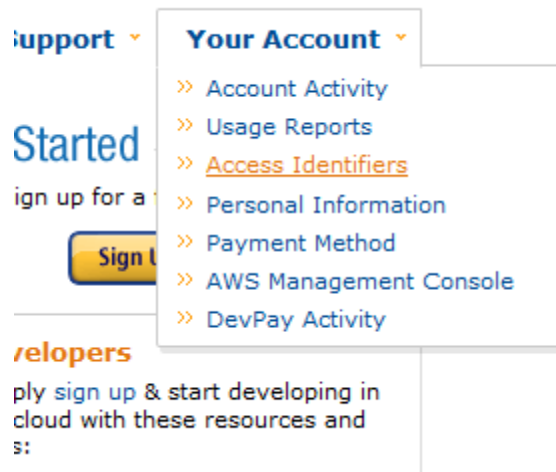
#### To install on Eclipse 3.4 (Ganymede)

Before installing the toolkit with Eclipse 3.4, be sure to install all necessary prerequisites.

1. Open **Help -> Software Updates....**
2. Click the **Available Software** tab.
3. Click **Add Site....** The Add Site dialog box opens.
4. Enter <http://aws.amazon.com/eclipse/> in the Location box, then click **OK**. The update site is now listed in the Available Software list.
5. Select all components of the archive, then click **Install....** Eclipse guides you through the remaining installation steps.

### ***Retrieving your AWS Credentials***

You can retrieve your AWS credentials at any time by either going to the AWS account management page at <http://aws.amazon.com/account> or by going to the main Amazon Web Services page and selecting **Access Identifiers** from the **Your Account** menu in the top right corner of the page.

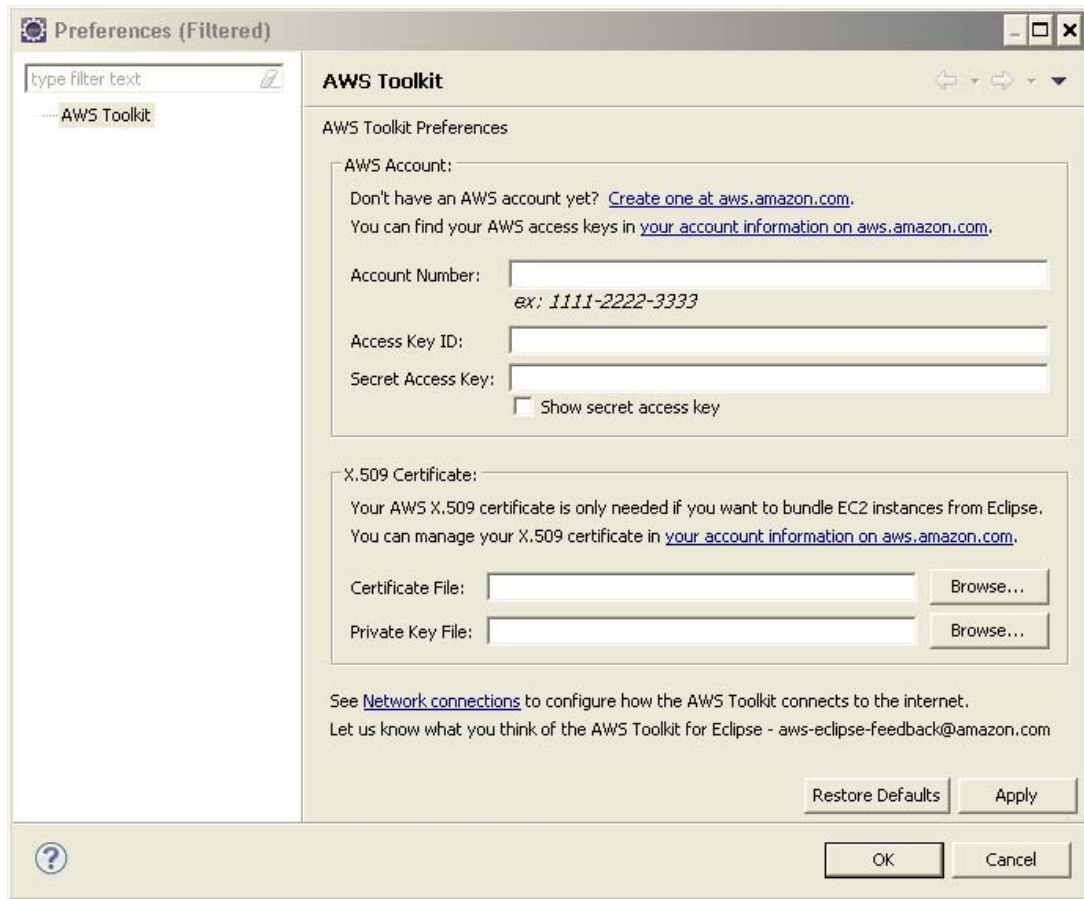


### ***Configuring the Toolkit with Your Credentials***

Once you've retrieved your AWS credentials, you will need to provide them to the AWS Toolkit for Eclipse plug-in.

**To set your global AWS access preferences:**

1. From the Eclipse **Window** menu, select **Preferences**. The preferences dialog appears.



2. Obtain your account number, access key and secret key from your [Account Information Page](#), and use them to complete the connection information. (In the above figure, only the AWS Account information needs to be completed for SimpleDB use. The X.509 section is for EC2 only).
3. Close the Preferences dialog by choosing the OK button. Your AWS credentials are now the global default for connections.

### **Adding X.509 Certificates**

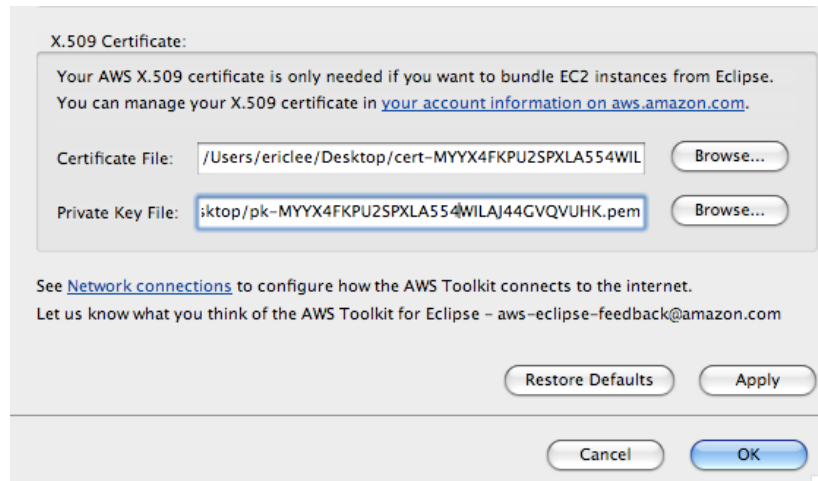
You may also configure the toolkit with your Amazon Web Services X.509 certificate. Your X.509 certificate is only required if you choose to bundle an Amazon Machine Image (AMI) with the AWS Toolkit for Eclipse.

If you do not have an X.509 certificate associated with your account, you can create a new one from the **Amazon Web Services Access Identifiers** page. A link to this page is included on the AWS Toolkit for Eclipse **Preferences** dialog.

Once you have created a new X.509 certificate, you will have an opportunity to download the certificate and private key files and save both to your local file system.

**IMPORTANT:** This is your only opportunity to download your private key.

Once your certificate and private key are downloaded, you can add the files to the toolkit configuration. In the toolkit **Preferences** dialog, use the **Browse** buttons to find the certificate and private key files that you've downloaded.



Click the **OK** button to close the configuration dialog.

With the addition of your AWS credentials, the AWS Toolkit for Eclipse plug-in should now be ready for use.

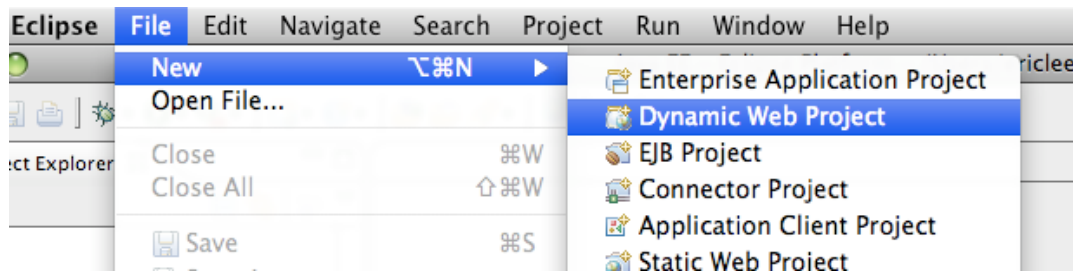
## Deploying a Sample Application to an Amazon EC2 Tomcat Cluster

To help illustrate how the toolkit can help Java developers working with Tomcat, we will create a simple "Hello World" web project and deploy it to a cluster of Tomcat servers hosted in Amazon EC2.

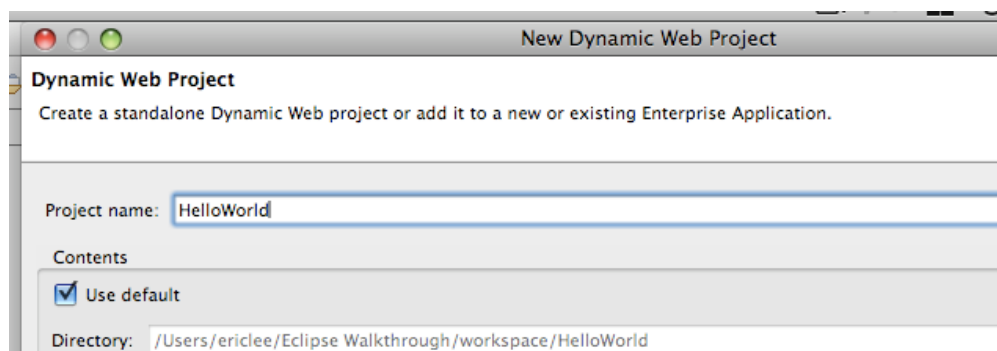
**NOTE:** This section assumes you have Apache Tomcat v6.0 installed. If you do not have Apache Tomcat v6.0 installed, you can download it from [http://Tomcat.apache.org/download-60.cgi](\"http://Tomcat.apache.org/download-60.cgi\").

### *Creating a Simple "Hello world" Web Application and Testing it Locally*

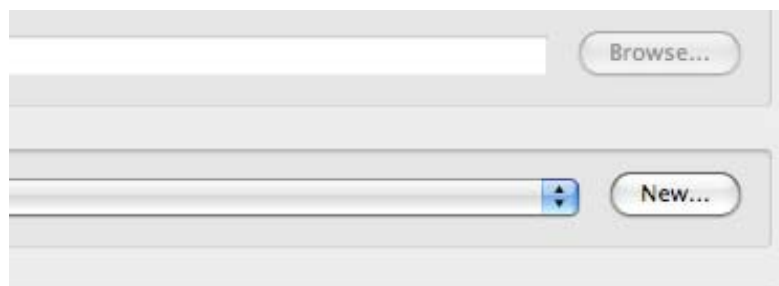
Our first step will be to create a new **Dynamic Web Project**. To do this, select the **File** menu, and then select **Dynamic Web Project** from the **New** menu.



In the **New Dynamic Web Project** dialog, add the name HelloWorld to the **Project Name** text box.

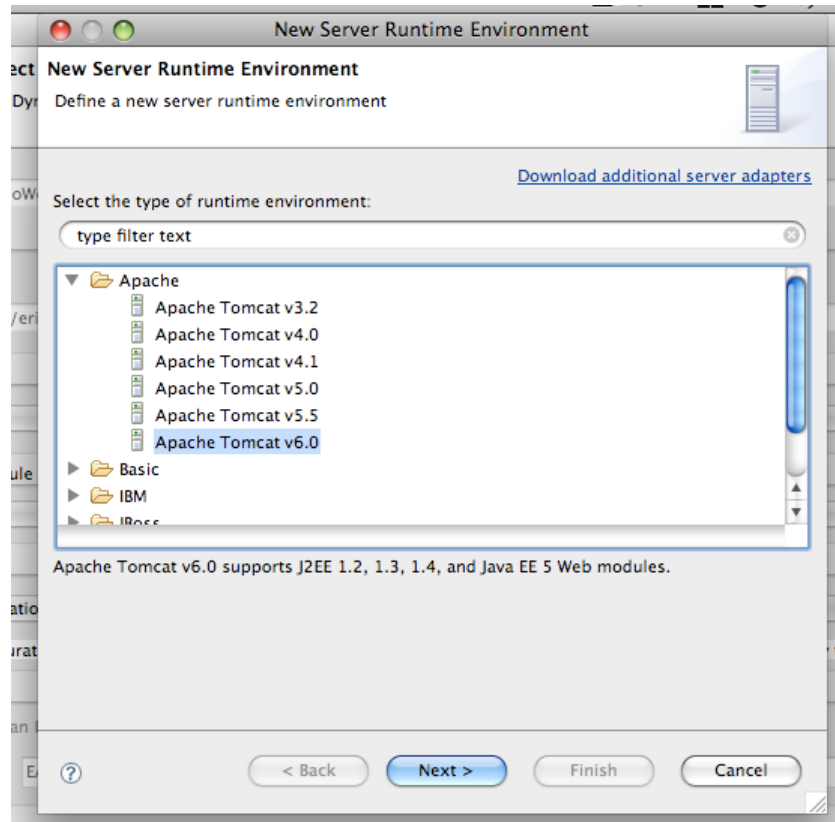


We will first deploy and test our project locally. To do this, click the **New** button next to the **Target Runtime** drop-down list.

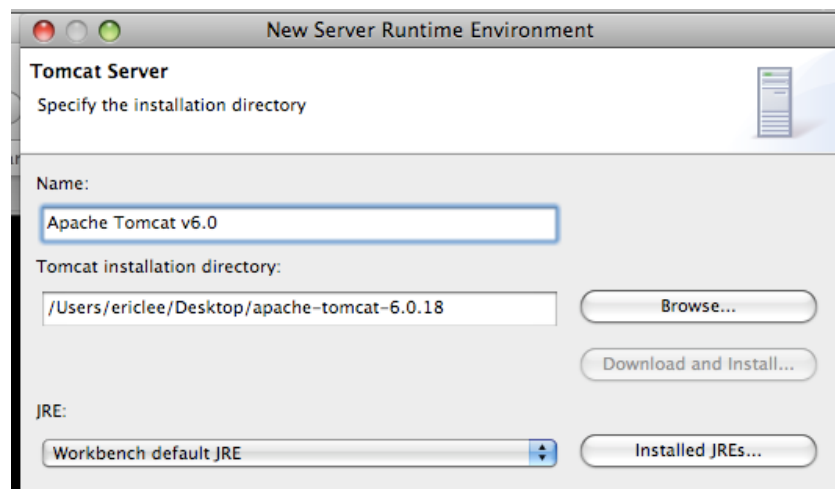


On the **New Server Runtime Environment** dialog box, choose the **Apache Tomcat v6.0** runtime environment from the list, and then click the **Next** button.



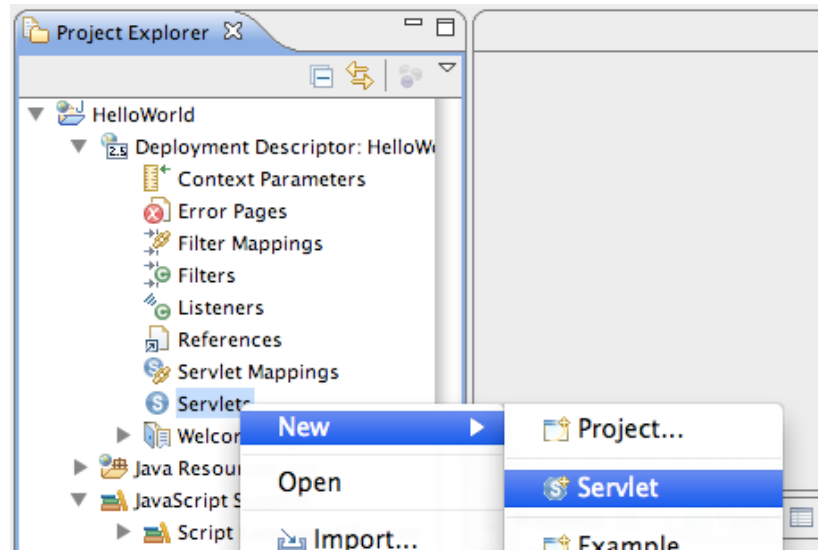


Specify the path to your local Tomcat installation directory in the **New Server Runtime Environment** dialog box, and then click **Finish** to close the dialog and create your project.

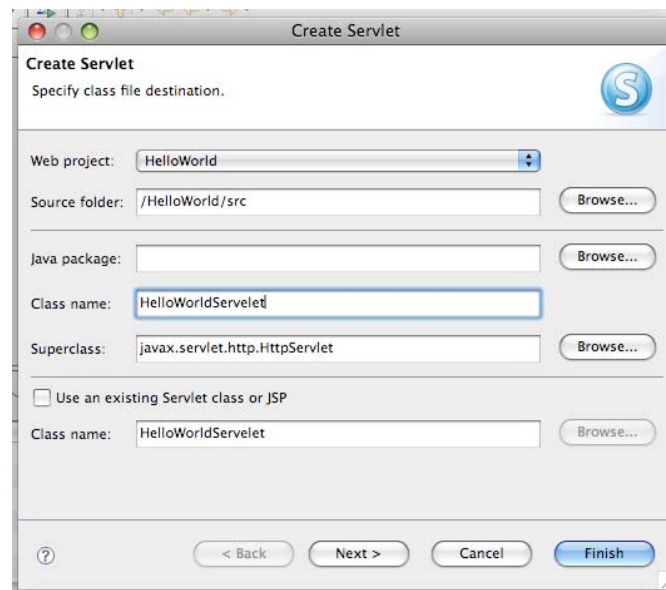


Once your project has been created, open the **Project Explorer** view, select your project, expand the **Deployment Descriptor:** node, and then right-click on the **Servlets** node.

From the context menu, select **New** and then **Servlet**. We will create a basic servlet that we can use to illustrate the remote debugging features in the toolkit.



Type the name `HelloWorldServlet` into the **Class Name** field in the **Create Servlet** dialog box, and then click the **Finish** button to create it.



The skeleton code for a basic servlet will now be created for you. To view this code, expand the **Java Resources: src** node in the **Project Explorer** window, expand the **(default package)** node, and then double-click on the **HelloWorldServlet.java** source code file.

Find the `doGet` method in the source code that was generated for you.

Since we are ultimately going to deploy our servlet to a Tomcat cluster, we will add code to this method that will output the name of the machine that the servlet is currently running on. This will allow us to see that different machines in the Tomcat cluster are running our test servlet.

To show our machine name, add the code shown below to the **doGet** method:

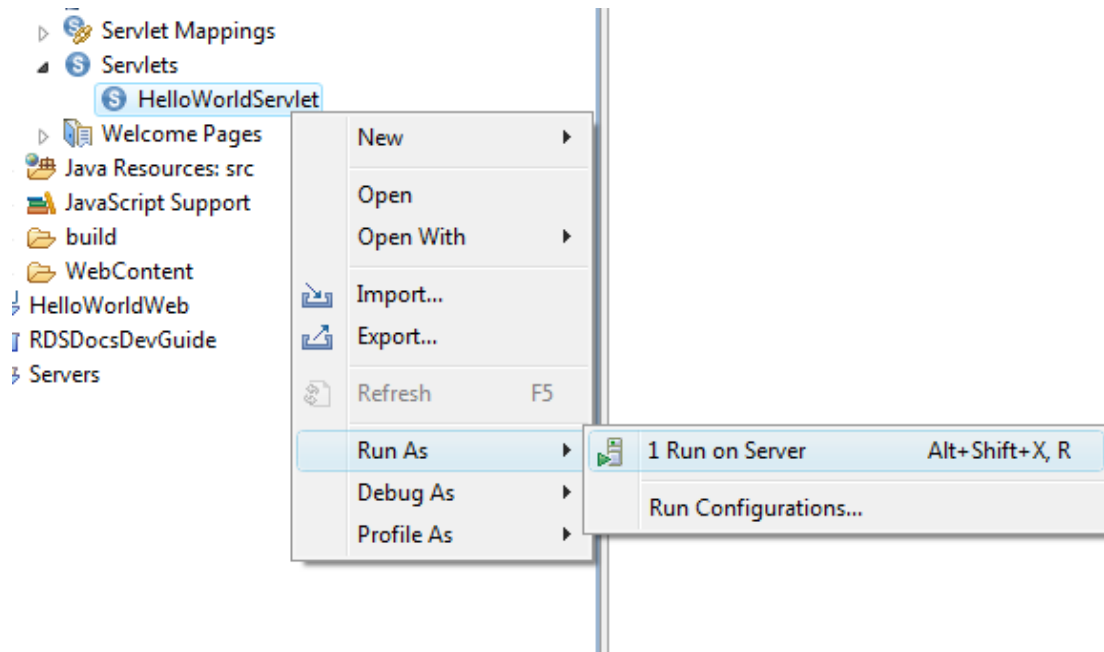
```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {
    // TODO Auto-generated method stub

    response.setContentType("text/html");

    java.io.PrintWriter output = response.getWriter();

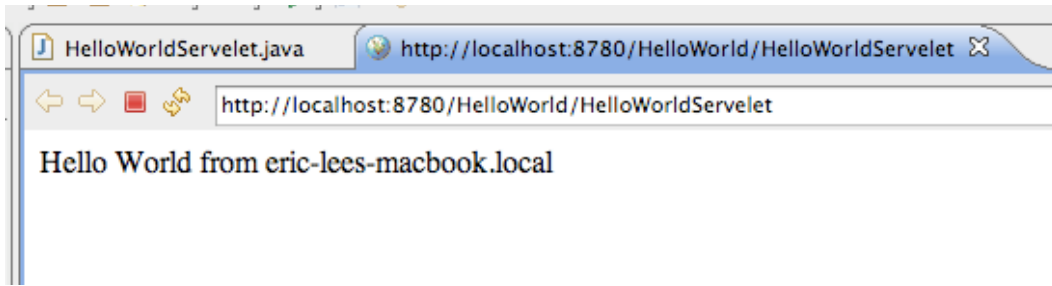
    output.println("Hello World from " + java.net.InetAddress.getLocalHost().getHostName());
    output.close();
}
```

Once your code has been written, save the file. Next, right-click on the **HelloWorldServlet.java** file in the **Project Explorer** and select **Run on Server** from the **Run As** context menu.



In the resulting dialog box, select the local Tomcat server that you created earlier, and then click the **Finish** button to run your project.

In a few moments, the local Eclipse browser should appear with the output of your servlet. That output should resemble the illustration below.



Now that we have verified our servlet works locally, we can deploy it to a Tomcat cluster running in Amazon EC2.

### ***Running our Servlet on an Amazon EC2 Tomcat Cluster***

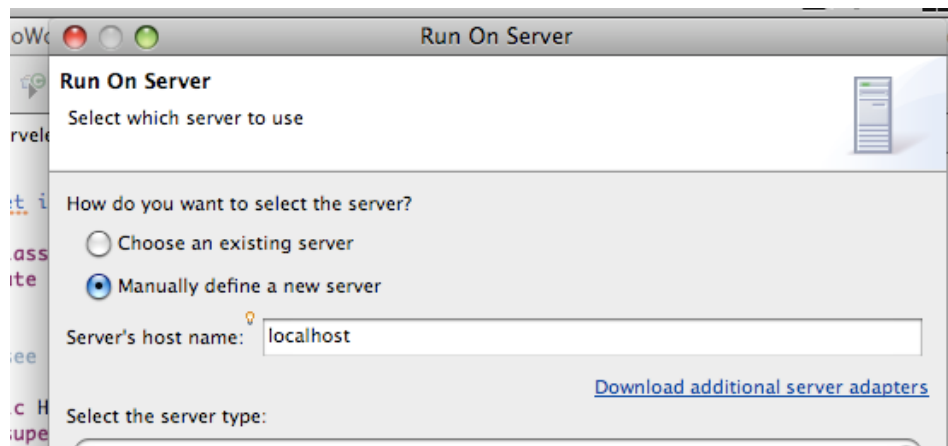
Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable, *virtual* computing capacity in the cloud. Amazon EC2 enables you to use web service interfaces to launch machine instances, load them with your custom application environment, and run your applications using as many or few systems as required.

The AWS Toolkit for Eclipse uses the Amazon EC2 web service in a manner that is specifically beneficial to Java developers deploying to Apache Tomcat. The toolkit integrates with the existing Eclipse Web Tools Platform and provides an easy-to-use interface to manage Amazon EC2 instances.

In this section, we will use the Amazon Web Services Toolkit for Eclipse to define and configure an Apache Tomcat cluster hosted on Amazon EC2, and then deploy and run our sample servlet on the cluster.

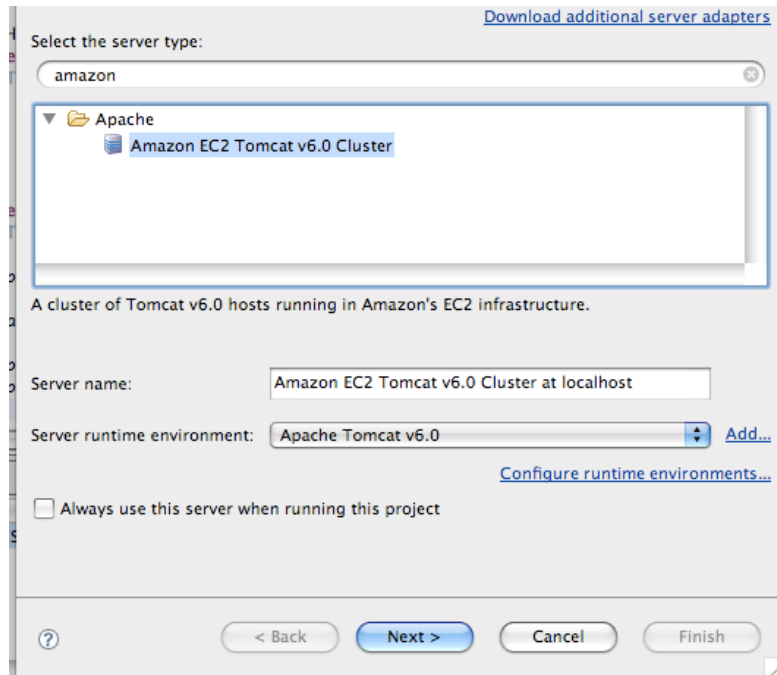
To get started, right-click on the `HelloWorldServlet.java` file in the Project explorer, and then select **Run on Server** from the **Run As** context menu.

In this case, we will define a new server definition – this new server will represent our Amazon EC2 instances. To do this, click on the **Manually define a new server** option on the **Run On Server** dialog box.



Select the **Amazon EC2 Tomcat v6.0 Cluster** option from the **Select the server type** list.

The **Server name** text box is automatically populated with the name “Amazon EC2 Tomcat v6.0 Cluster on localhost”. You may optionally change this name to one that’s easier to remember. This name will appear as an option on the **Run As** context menu once our cluster is created.



Click the **Next** button.

The next page to appear will be the **Configure Your Cluster Setting** dialog.

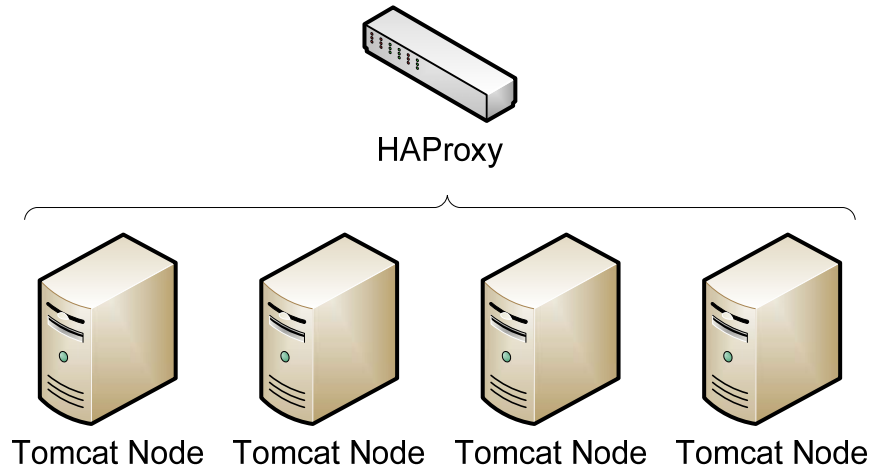
## Defining the Cluster

The **Configure Cluster Settings** dialog box is where we can define the characteristics of our Tomcat Cluster.

## Specifying Cluster Size

The **Cluster Size** setting specifies how many Amazon EC2 instances will be instantiated for the cluster. The toolkit will instantiate one instance of the image configured with HAProxy, and the remaining instances configured with Tomcat.

For example, specifying 5 in the **Cluster Size** field would result in the following 4-node Tomcat cluster:



**NOTE:** One advantage of an Amazon EC2 cluster is *flexibility*. Servers can be added to or removed from an Amazon EC2 cluster at any time as needed to appropriately meet demand.

## Choosing a Server AMI

The **Server AMI** setting defines which Amazon Machine Image (AMI) will be used for the Amazon EC2 instances.

The AWS Toolkit for Eclipse deployment support is designed to work with two specific types of Amazon Machine Images (AMIs):

- The **ami-11ca2d78** image is configured as a single server instance running Apache Tomcat v6.0. This is the default image for the toolkit.
- The **ami-11ca2d78** image is configured with *HAProxy*, an open-source HTTP/TCP load balancer. Choosing this image will instantiate a Tomcat cluster.

For this sample, we will choose the `ami-d1ca2vb8` image so that we are running our test application in an Amazon EC2 Tomcat Cluster.

To select this image, click the **Custom** radio button on the dialog box, and then type (or paste) the image name into the text field. Note that the toolkit will validate your selection as you type it.

## Choosing an Instance Type

Our next choice is the **Instance Type**. The table below shows an example of the types of instances that are available as of this writing.

Type	Description	Cost
------	-------------	------

<b>Small</b>	• 1.7 GB of memory	\$0.10/hour (US)
	• 1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit)	\$0.11/hour (Europe)
	• 160 GB of instance storage	
	• 32-bit platform	
<b>High-CPU Medium</b>	• 1.7 GB of memory	\$0.20/hour (US)
	• 5 EC2 Compute Units (2 virtual cores with 2.5 EC2 Compute Units each)	\$0.22/hour (Europe)
	• 350 GB of instance storage	
	• 32-bit platform	

**NOTE:** The toolkit currently only supports 32-bit Amazon EC2 instances.

In the **Configure Cluster Settings** dialog, select **Small** as the instance type.

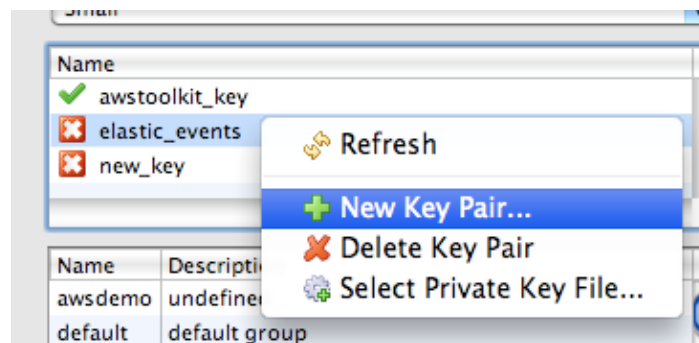
### Securing the Tomcat Amazon EC2 Cluster

Amazon EC2 makes security a top priority. In order to access your running Amazon EC2 instances, you must validate your identity with an X.509 certificate key pair. This pair is comprised of a X.509 public certificate and its matching private key.

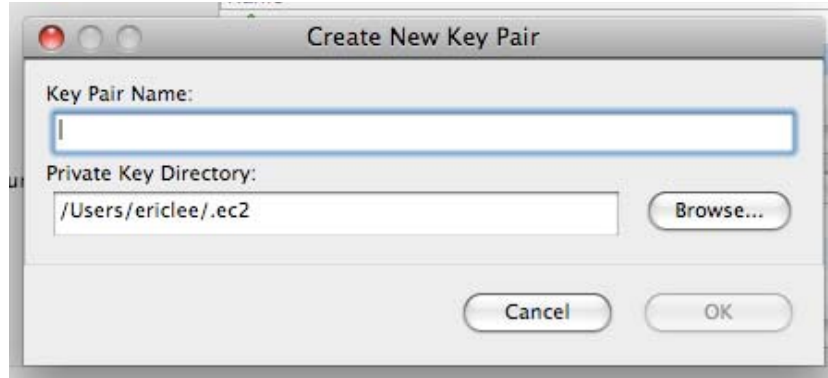
### Creating a Key Pair

This key-pair is structurally the same as the X.509 certificate that is associated with our Amazon Web Services account. But, we always create separate, new key-pairs for accessing our instances. Unlike the single X.509 certificate that is associated with our account, we can have as many key-pairs for accessing instances as we want to.

To create a new key-pair, right-click and choose the **New Key Pair** menu option.



In the resulting dialog, choose a name of your new key-pair (any name of reasonable length is fine). Press the **OK** button to close this dialog.



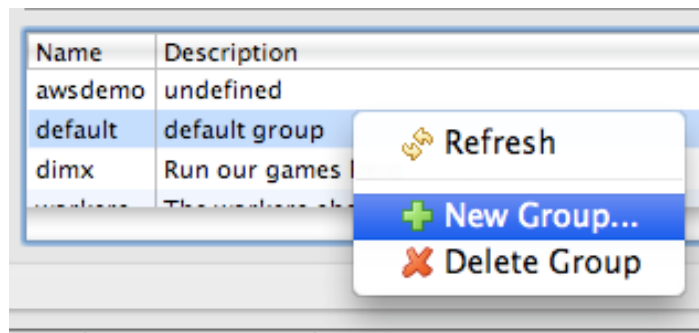
You may also use existing key-pairs. Existing key pair files should be placed in the `~/ .ec2` folder if you are using Linux, or the directory specified by the `HOME` environment variable if you are using Windows.

## Creating Security Groups

The next section of the configuration settings dialog enables us to select or create security groups.

Amazon EC2 security groups function much like firewalls; they can be configured to only allow specific types of requests.

To create a new security group, right-click in the **Security Group** list and choose **New Group...** from the context menu to create a new security group. A dialog box will appear, prompting you to enter a name and description for your security group.



Enter a name and description for your security group.

**NOTE:** A best practice is to name the security group in a way that reflects the machines that will be part of that group. For example, a group that will contain database servers might be called 'Data Tier'.



We have now finished configuring our Tomcat cluster. The **Configure Cluster Settings** dialog box should look similar to the illustration below:

**Configure Cluster Settings**  
Configure your Amazon EC2 cluster

Cluster Settings:

EC2 Region: us-east-1

Cluster Size: 5

Server AMI: ☐ Default (ami-11ca2d78) ☒ Custom ami-11ca2d78

Instance Type: Small

Key Pair:


Name
✓ Test1
✓ Test2
✓ Test3

Security Group:

Name	Description
My Security Group	My new security group
default	default group

Optional Settings:

Elastic IP	Attached Instance

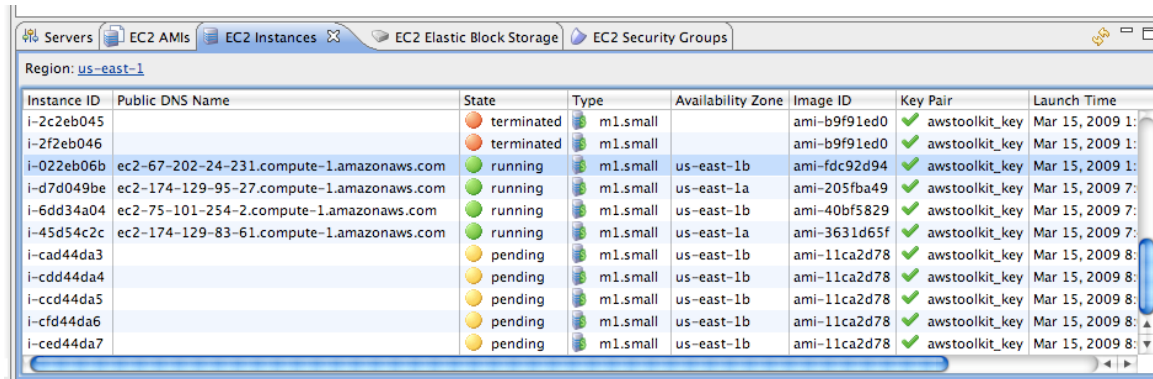
 You will be charged the hourly rate for any instances you launch until you successfully shut them down.

Press the **Finish** button to close this dialog box. Your cluster has been created and will now be listed in the **Servers** view.

## Starting the Amazon EC2 Tomcat Cluster

Right-click on your server and choose **Start**. This is exactly what we would do to start a local Tomcat server; the difference now is that we are actually starting a cluster of Tomcat servers running in Amazon EC2.

After a few moments, switch to the **EC2 Instances** view; you should see a set of instances in the *pending* state.



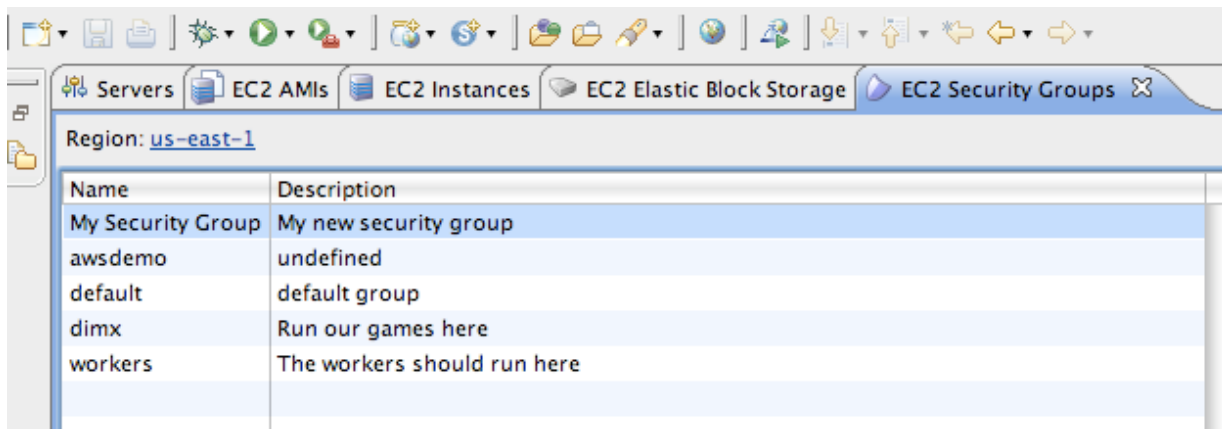
Region: us-east-1

Instance ID	Public DNS Name	State	Type	Availability Zone	Image ID	Key Pair	Launch Time
i-2c2eb045		terminated	m1.small		ami-b9f91ed0	awstoolkit_key	Mar 15, 2009 1:
i-2f2eb046		terminated	m1.small		ami-b9f91ed0	awstoolkit_key	Mar 15, 2009 1:
i-022eb06b	ec2-67-202-24-231.compute-1.amazonaws.com	running	m1.small	us-east-1b	ami-fdc92d94	awstoolkit_key	Mar 15, 2009 1:
i-d7d049be	ec2-174-129-95-27.compute-1.amazonaws.com	running	m1.small	us-east-1a	ami-205fba49	awstoolkit_key	Mar 15, 2009 7:
i-6dd34a04	ec2-75-101-254-2.compute-1.amazonaws.com	running	m1.small	us-east-1b	ami-40bf5829	awstoolkit_key	Mar 15, 2009 7:
i-45d54c2c	ec2-174-129-83-61.compute-1.amazonaws.com	running	m1.small	us-east-1a	ami-3631d65f	awstoolkit_key	Mar 15, 2009 7:
i-cad44da3		pending	m1.small	us-east-1b	ami-11ca2d78	awstoolkit_key	Mar 15, 2009 8:
i-cdd44da4		pending	m1.small	us-east-1b	ami-11ca2d78	awstoolkit_key	Mar 15, 2009 8:
i-ccd44da5		pending	m1.small	us-east-1b	ami-11ca2d78	awstoolkit_key	Mar 15, 2009 8:
i-cfd44da6		pending	m1.small	us-east-1b	ami-11ca2d78	awstoolkit_key	Mar 15, 2009 8:
i-ced44da7		pending	m1.small	us-east-1b	ami-11ca2d78	awstoolkit_key	Mar 15, 2009 8:

While we wait for our instances to start, we will configure our security group.

## Configuring Amazon EC2 Security Groups

Switch to the **EC2 Security Groups** view and select the group you just created.

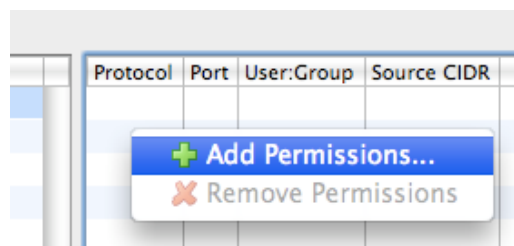


Region: us-east-1

Name	Description
My Security Group	My new security group
awsdemo	undefined
default	default group
dimx	Run our games here
workers	The workers should run here

By default, security groups are configured to block traffic on all ports. So, right now, our instances are not reachable at all. This is ideal from a security point of view because nobody has access to our Amazon EC2 machines until we decide to allow it.

In the section on the left, right-click and choose **Add Permissions...**

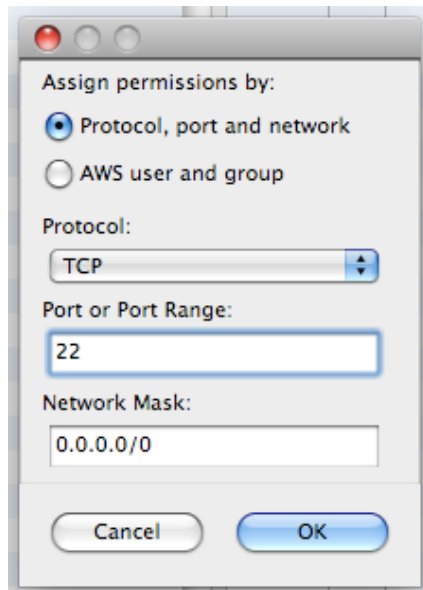


There is an additional level of security; besides specifying which ports are accessible, we can specify what addresses are allowed to access these ports.

This is commonly done when we want to model security groups that are not publicly accessible, but need to communicate with one another. Also, it is common to only allow certain hosts to access administrative functions on a production site; the network mask in a security group is the way to do this.

For our example, we will just keep the default value, which allows any host to access our ports.

Enter **22** in the dialog; this will enable us to SSH into our instances. Press the **OK** button to enable access to this port.

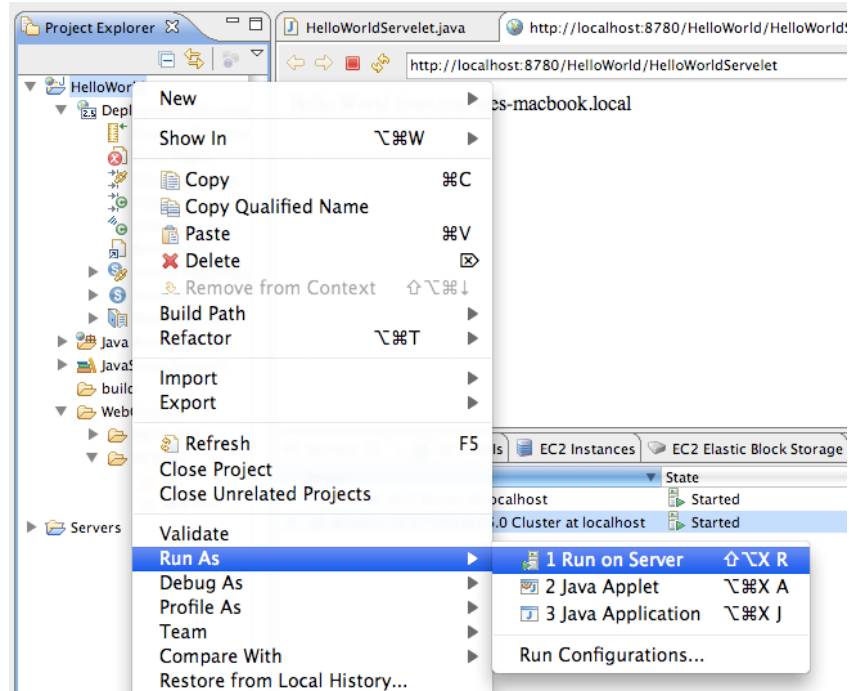


Repeat the same process for ports 80, 8080 and 443. Your security group should look as shown below.

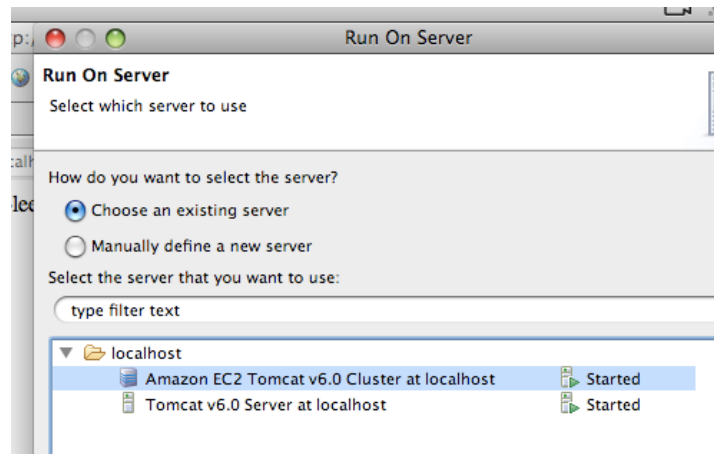
Protocol	Port	User:Group	Source CIDR
tcp	22		0.0.0.0/0
tcp	80		0.0.0.0/0
tcp	443		0.0.0.0/0
tcp	8080		0.0.0.0/0

### Running the Sample Program on an Amazon EC2 Cluster

Now we are ready to give our cluster a try. To get started, right-click on the `HelloWorldServlet.java` file in the Project explorer, and then select **Run on Server** from the **Run As** context menu.

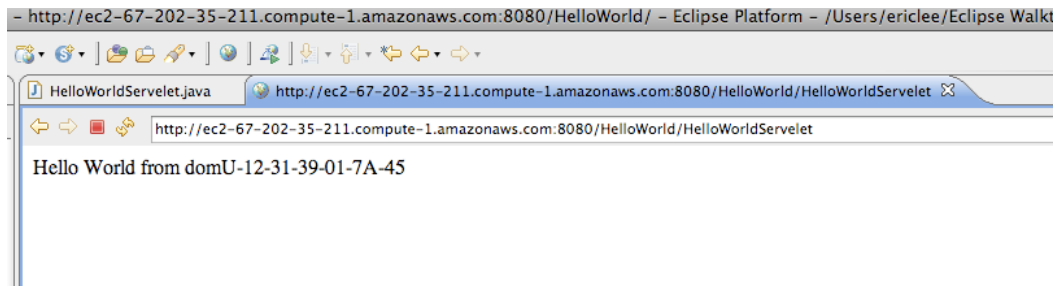


This time, instead of choosing your local server, choose the **Amazon EC2 Tomcat v6.0 Cluster** that you just created and press the **Finish** button.

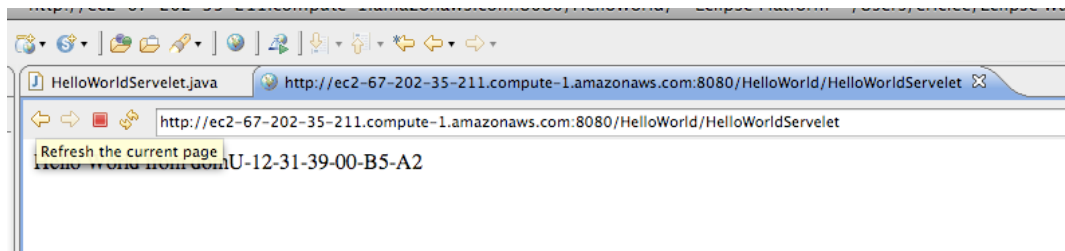


This will result in a dialog box that shows the progress of your project being deployed to Amazon EC2.

After a few moments, you should see your application being served from an instance in our Amazon EC2 cluster.



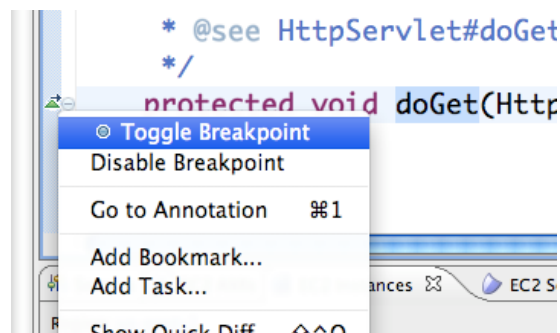
Refresh your browser a few times and you should see the machine name change. The toolkit configured a cluster of Tomcat servers, so each request may be served by a different node in that cluster.



### Debugging Applications

The toolkit enables you to debug applications deployed to an Amazon EC2 cluster. The toolkit's integration with the Eclipse Web Tool Platform makes debugging applications deployed to a cluster as familiar as debugging locally deployed applications. In this section, we will debug the sample application that we've already deployed to our Amazon EC2 cluster.

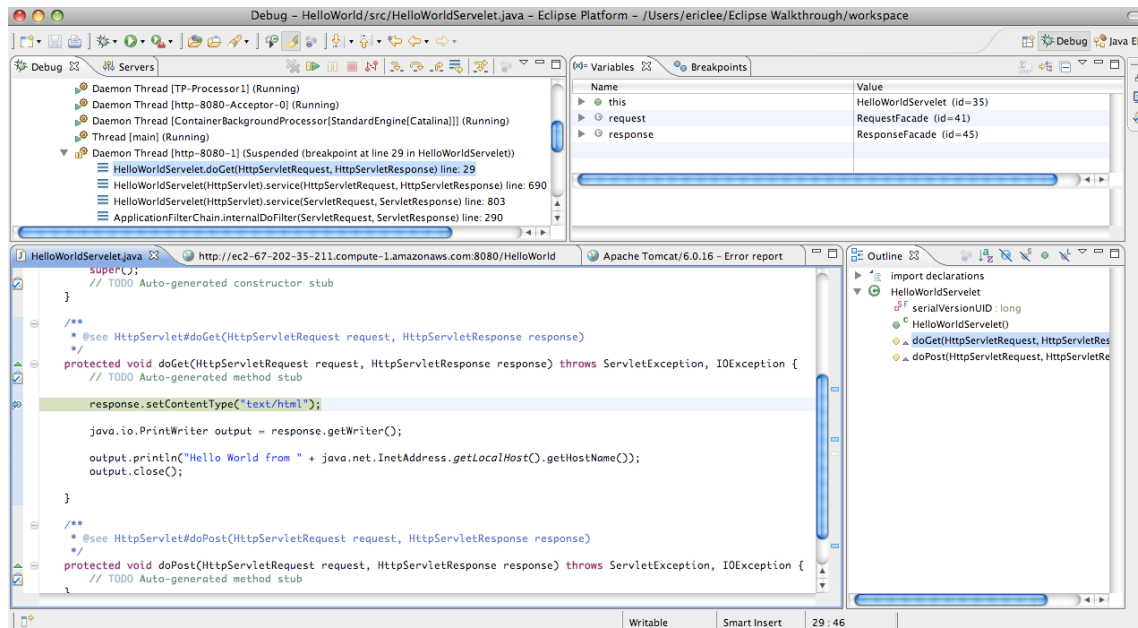
First, set a breakpoint in the code. To do this, open the `HelloWorldServlet.java` source file, find the `doGet` method, and then right-click and choose **Toggle Breakpoint** from the context menu to activate a breakpoint for this method.



To start debugging, right-click on the `HelloWorldJava` servlet and select **Debug on Server** from the **Run As** context menu.

After a few moments, we should hit our breakpoint in the `doGet` method.

## Getting Started with Amazon EC2 Management in Eclipse



Even though we are executing our code in a cluster of Tomcat servers running in the cloud, we can still debug just as though we were developing locally. The toolkit's integration with the Web Tools Platform makes this possible.

## Examining and Monitoring Amazon EC2 Instances

Now that we have our sample application running in Amazon EC2, we can take a closer look at how our cluster has been set up.

### Examining Amazon EC2 Instances

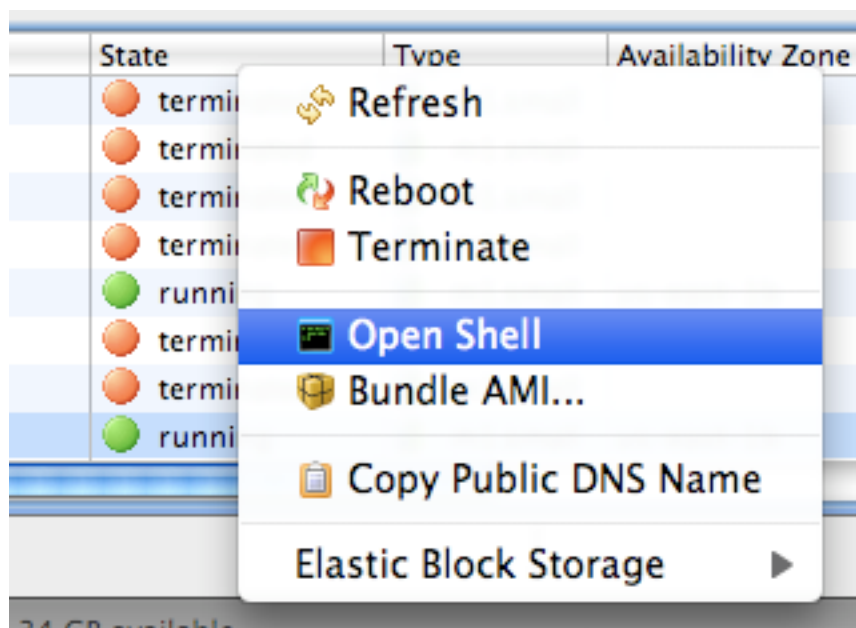
Select the **EC2 Instances** tab. Find the running instances and look at the **Image ID** column for these instances. There should be one instance with an Image ID value of **ami-d1ca2db8**. This instance is running the HAProxy that was automatically configured when we created our cluster. The HAProxy accepts all of application requests and hands them off to nodes in the cluster. The **Public DNS Name** value for this instance is the URL for our sample application.

	Availability Zone	Image ID		Key Pair
11.small	us-east-1b	ami-11ca2d78	✓	awsto
11.small	us-east-1b	ami-11ca2d78	✓	awsto
11.small	us-east-1b	ami-11ca2d78	✓	awsto
11.small	us-east-1b	ami-d1ca2db8	✓	awsto

### Connecting to EC2 Instances

The toolkit has a convenient feature that enables us to SSH directly into an instance. We will use this feature connect to the HAProxy server and enable monitoring.

In the **EC2 Instances** tab, right-click the server running HAProxy and choose **Open Shell** from the context menu.



In a few moments, we will have a terminal connected to our instance running a command shell.

### Enabling HAProxy Monitoring

In the command shell, open the `/etc/haproxy.cfg` file in a text editor. In the HAProxy configuration file, navigate towards the bottom of the file, and add the following line under the `listen tomcat-proxy` entry:

```
stats enable
```

This setting enables HAProxy to collect and report statistics for the proxy.

Notice each server listed in this configuration file. These servers are the nodes in our Tomcat cluster; the network address listed for each server is its private DNS name.

```

    redispatch
    maxconn      2000
    contimeout    5000
    clitimeout    50000
    srvtimeout    50000

listen tomcat-proxy
    bind          :8080
    stats enable
    balance       roundrobin
    server s1     domU-12-31-39-02-F0-B5.compute-1.internal:8080
    server s2     domU-12-31-39-03-45-33.compute-1.internal:8080
    server s3     domU-12-31-39-02-64-84.compute-1.internal:8080

-- INSERT --

```

Each Amazon EC2 instance has a public and private DNS name.

**NOTE:** Using the private DNS name is recommended for communicating between Amazon EC2 instances for cost and performance benefits.

After each private DNS address, add **check port 8080 inter 1000**

**NOTE:** The port number may be different depending on your server configuration.

```

listen tomcat-proxy
    bind          :8080
    stats enable
    balance       roundrobin
    server s1     domU-12-31-39-02-F0-B5.compute-1.internal:8080 check port 8080 inter 1000
    server s2     domU-12-31-39-03-45-33.compute-1.internal:8080 check port 8080 inter 1000
    server s3     domU-12-31-39-02-64-84.compute-1.internal:8080 check port 8080 inter 1000

```

Save the changes to the configuration file and exit the text editor.

The HAProxy will need to be restarted before the configuration changes become effective.



To restart the HAProxy, type `ps -e -f | grep haproxy` from the command shell prompt to show a list of all processes running on this instance and note the PID (Process ID) for the running instance of HAProxy. Copy the complete command-line for **haproxy** from the process listing to your clipboard to make restarting the haproxy process easier later on. For example:

```
/env/haproxy/haproxy -D -f /etc/haproxy.cfg -p /var/run/haproxy
```

To shut down the HAProxy, at the command prompt, type **kill** followed by the PID for the running instance of HAProxy.

To restart HAProxy, paste the command line to run **haproxy** into your terminal. Run **ps -e -f** again to ensure this process is running.

HAProxy is now collecting statistics for servers running in the cluster.

### Viewing the HAProxy Statistics Dashboard

To view the dashboard, appending `/haproxy?stats` to the public DNS of our proxy server and paste it into a browser. You can find the public DNS of the proxy server by right-clicking on the server in the EC2 Instances tab and selecting **Copy Public DNS Name** from the context menu. You can also copy the URL of the HelloWorld servlet from the browser and append `/haproxy?stats`. For example:

```
http://ec2-12-123-234-56.compute-1.amazonaws.com/haproxy?stats
```

The dashboard view should show all of the servers in the cluster as UP.

Edit.jsp
Statistics Report for HAProxy

<http://ec2-72-44-46-89.compute-1.amazonaws.com:8080/haproxy?stats>

# HAProxy

## Statistics Report for pid 1110

### > General process information

pid = 1110 (nbproc = 1)

uptime = 0d 0h00m13s

system limits : memmax = unlimited ; ulimit-n = 8206

maxsock = 8206

maxconn = 4096 (current conns = 1)

active UP

active UP, going down

active DOWN, going up

active or backup DOWN

backup UP

backup UP, going down

backup DOWN, going up

not checked

### > Proxy instance tomcat-proxy : 1 conns (maxconn=2000), 0 queued (0 unassigned), 6 total conns

Server		Queue		Sessions			Errors								
Name	Weight	Status	Act.	Bck.	Curr.	Max.	Curr.	Max.	Limit	Cumul.	Conn.	Resp.	Sec.	Check	Down
s1	1	UP	Y	-	0	0	0	0	-	0	0	0	0	0	0
s2	1	UP	Y	-	0	0	0	0	-	0	0	0	0	0	0
s3	1	UP	Y	-	0	0	0	0	-	0	0	0	0	0	0
Dispatcher	-	UP	-	-	0	0	1	1	2000	6	0	0	0	-	-
Total	-	UP	3	0	0	0	1	1	2000	6	0	0	0	0	0

HAProxy will monitor the status of the servers in the cluster and only delegate requests to healthy servers.

To observe this, terminate or reboot some servers by selecting the **EC2 Instances** tab and locating instances running with the **ami-11ca2d78** ImageID. Terminate one or more of these instances by right-clicking and selecting **Terminate** from the context menu.

In this example, we will terminate all but one instance in our cluster and then view the HAProxy dashboard . The terminated instances will be shown on the dashboard view.

Edit.jsp

Statistics Report for HAProxy

Peter's Page JSPWiki: Main

[⏮](#)
[⏪](#)
[🔍](#)
[🔖](#)

http://ec2-72-44-46-89.compute-1.amazonaws.com:8080/haproxy?stats

# HAProxy

## Statistics Report for pid 1110

> General process information

pid = 1110 (nbproc = 1)

uptime = 0d 0h02m43s

system limits : memmax = unlimited ; ulimit-n = 8206

maxsock = 8206

maxconn = 4096 (current conns = 1)

active UP

active UP, going down

active DOWN, going up

active or backup DOWN

backup UP

backup UP, going down

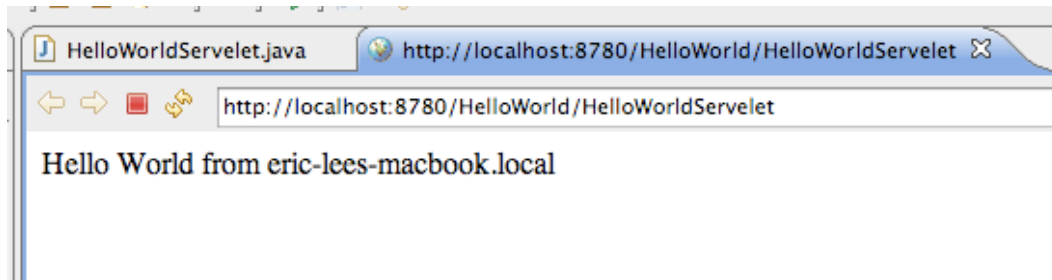
backup DOWN, going up

not checked

> Proxy instance tomcat-proxy : 1 conns (maxconn=2000), 0 queued (0 unassigned), 10 total conn:

Server		Queue		Sessions				Errors								
Name	Weight	Status	Act.	Bck.	Curr.	Max.	Curr.	Max.	Limit	Cumul.	Conn.	Resp.	Sec.	Check	Down	
s1	1	DOWN	Y	-	0	0	0	0	1	-	1	0	0	0	2	1
s2	1	DOWN	Y	-	0	0	0	0	0	-	0	0	0	0	2	1
s3	1	UP	Y	-	0	0	0	0	0	-	0	0	0	0	0	0
Dispatcher	-	UP	-	-	0	0	1	1	2000		9	0	0	0	-	-
Total	-	UP	1	0	0	0	1	1	2000	10	0	0	0	0	4	2

Even though all but one of the nodes in our cluster is down, our application should still be available as long as one node is available:



The toolkit allows you to quickly and easily realize the reliability and scalability benefits of an Amazon EC2 cluster from within the Eclipse development environment.

## Using the EC2 AMIs View

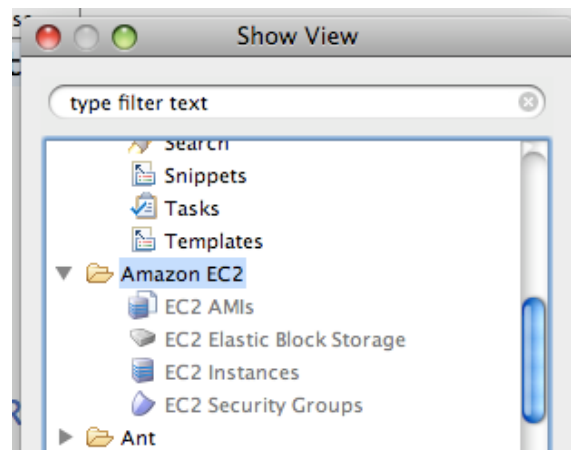
The toolkit is tightly integrated with the Eclipse platform. Besides making it easy to deploy applications to Amazon EC2 clusters, it also provides a number of views that can be used to manage Amazon EC2 instances. These views are intended to give developers quick access to the most crucial aspects of managing Amazon EC2.

**NOTE:** For more comprehensive management capabilities use the AWS Toolkit for Eclipse in conjunction with the AWS Management Console. The AWS Management Console is available at <https://console.aws.amazon.com>.

In this section, we will work with the toolkit's EC2 AMIs view. Access this view by selecting the Eclipse **Window** menu. Select **Show View** and then **Other** to display the Eclipse **Show View** dialog box.

### Opening the EC2 AMIs View

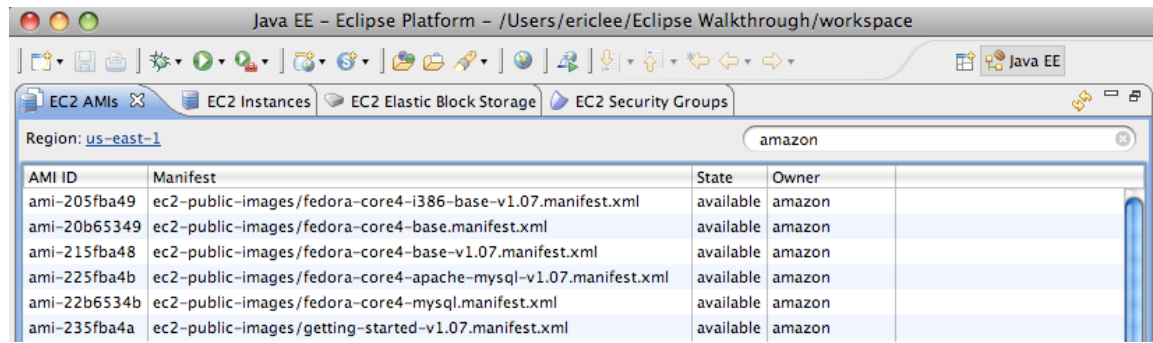
Open the **Amazon EC2** node and select the **EC2 AMIs** view, and then click the **OK** button.



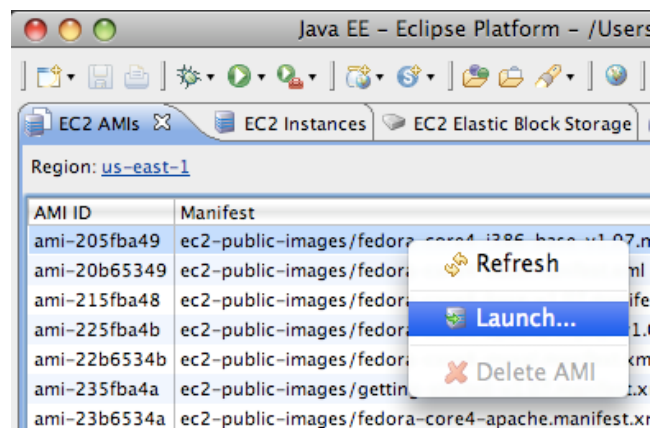
The EC2 AMIs view displays a searchable list of the Amazon Machine Images (AMIs) that are available for use.

### Finding and Launching an AMI

To find a list of all images created by Amazon, type “amazon” in the search box at the top left of the EC2 AMIs view, and press the **ENTER** key.



Amazon Machine Images are often referenced by just their ID. For our example, find the image with an ID of **ami-205fba49**. Right-click this image and select **Launch...** from the context menu.



### Configuring AMI Instances

Launching an AMI instance will display the **Launch Amazon EC2 Instances** dialog. This dialog allows us to configure our EC2 instances. The dialog will look similar to the one shown below.

## Launch Amazon EC2 Instances

Configure the options for launching your Amazon EC2 instances



AMI: ami-205fba49 (i386)

Number of Hosts: 1

Instance Type: Small

Memory: 1.7 GB      Virtual Cores: 1  
Disk Capacity: 160 GB      Architecture: 32 bits

Availability Zone: us-east-1a

Key Pair:

Name
✓ Test1
✓ Test2
✓ Test3

Security Group:

Name	Description
My Security Group	My new security group
default	default group

User Data:

You will be charged the hourly rate for any instances you launch until you successfully shut them down.

Finish Cancel

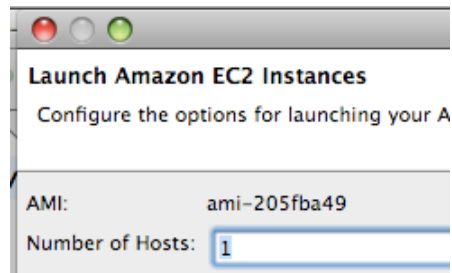
There are a number of options available when we launch an AMI, including:

- The number of instances
- The configuration of our instances
- The physical location of our instances

## Selecting the Number of Instances

Because Amazon EC2 focuses on *elastic* computing, the number of running instances can shrink or grow based on demand. You are only charged for the instances actually running.

In this example, we will launch a single instance by typing 1 into the **Number of Hosts** text box:

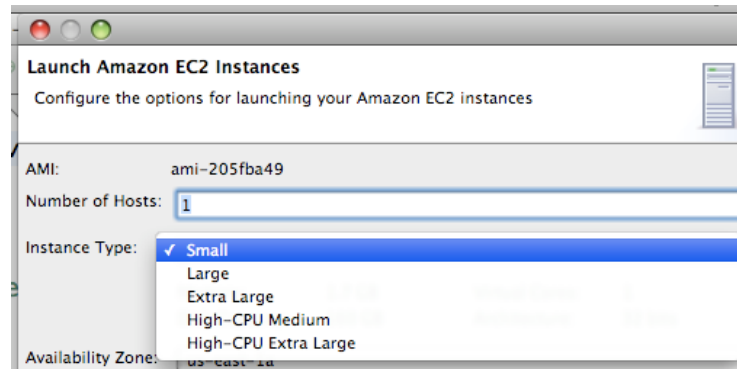


## Selecting the Instance Type

The **Instance Type** setting is used to indicate the type of hardware our instance will run on. As of this writing, the instance types described in the following table are available:

Type	CPU	Memory	Storage	Platform	I/O
<b>Small</b>	1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit)	1.7 GB	160 GB instance storage (150 GB plus 10 GB root partition)	32-bit	Moderate
<b>Large</b>	4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each)	7.5 GB	850 GB instance storage (2 x 420 GB plus 10 GB root partition)	64-bit	High
<b>Extra Large</b>	8 EC2 Compute Units (4 virtual cores with 2 EC2 Compute Units each)	15 GB	1690 GB instance storage (4 x 420 GB plus 10 GB root partition)	64-bit	High
<b>High-CPU Medium</b>	5 EC2 Compute Units (2 virtual cores with 2.5 EC2 Compute Units each)	1.7 GB	350 GB instance storage (340 GB plus 10 GB root partition)	32-bit	Moderate
<b>High-CPU Extra Large</b>	20 EC2 Compute Units (8 virtual cores with 2.5 EC2 Compute Units each)	7 GB	1,690 GB instance storage (4 x 420 GB plus 10 GB root partition)	64-bit	High

For our example, we will choose a **Small** instance type.



### Choosing an Availability Zone

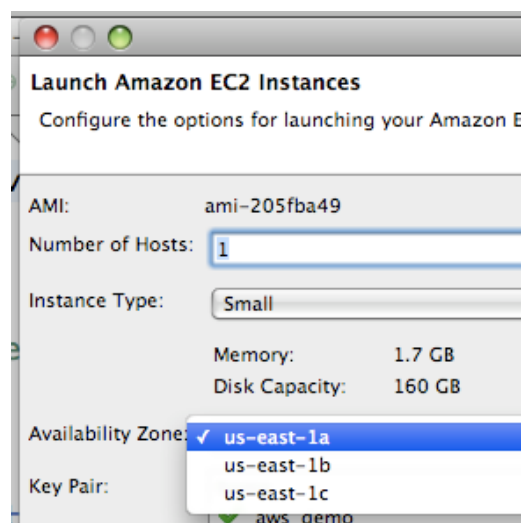
Amazon EC2 provides the ability to place instances in multiple locations. Amazon EC2 locations are composed of *Availability Zones* and *regions*.

Regions are located in separate geographic areas (for example, the United States and Europe). Availability Zones are distinct locations within a region that are engineered to be insulated from failures in other Availability Zones and to provide inexpensive, low latency network connectivity to other Availability Zones within the same region.

You can leverage regions in your application design to allow your application be deployed closer to specific customers or to meet legal or other requirements. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location.

The region used by the toolkit is automatically selected when the toolkit is installed, but may be changed at any time using the AWS Toolkit for Eclipse **Preferences** dialog.

In this example, we are configured for the US region.



### **Configuring Security**

You can now configure security settings for your Amazon EC2 instances. For more information, please see the [Securing the Tomcat Amazon EC2 cluster](#) section of this document.

### **Launching the AMI Instance**

Once the configuration for the AMI instance is complete, click the **Finish** button. Your Amazon EC2 instance will now be launched and available for use. You can use the **EC2 Instances** view to check the state of your new instance and begin working with it.

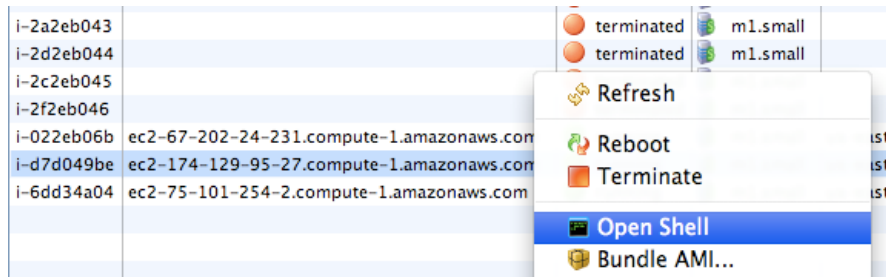


## Persisting Amazon EC2 Instance Data

By default, Amazon EC2 instances do not save data when terminated. You can save data for your instances by either *bundling* your instance, which creates a snapshot of your instance state as an Amazon Machine Image (AMI), or by using Amazon Elastic Block Storage (EBS).

In this example, we will create a test file on our instance.

In the **EC2 Instances** view, right-click the EC2 instance and click **Open Shell**.



A terminal will display showing a command shell for our Amazon EC2 instance.

```

root@ip-10-251-90-223 ~ ssh - 111x2
Last login: Sun Mar 15 19:30:08 on ttys001
eric-lee-macbook:~ ericlee$ /usr/bin/ssh -o CheckHostIP=no -o TCPK
ServerAliveInterval=120 -o ServerAliveCountMax=100 -i /Users/ericl
95-27.compute-1.amazonaws.com
Last login: Sun Mar 15 22:25:59 2009 from 68.166.180.21

  _ | _ | _ ) Rev: 3
 _ | ( _ | /
 _ | \ _ | _ |

Welcome to an EC2 Public Image
:-)

Base

[root@ip-10-251-90-223 ~]#

```

Create a simple file by typing the following at the command prompt:

```
echo 'AWS Toolkit Rocks!' >helloworld.txt
```

```

Welcome to an EC2 Public Image
:-)

Base

[root@ip-10-251-90-223 ~]# echo 'AWS Toolkit Rocks!' > helloworld.txt

```

Without creating an instance bundle or attaching elastic block storage to our Amazon EC2 instance, this file will be lost when the instance is terminated.

## Bundling an Amazon EC2 Instance

The first technique for persisting instance data is to bundle the instance into an Amazon Machine Instance (AMI). The AWS Toolkit for Eclipse makes this very simple to do.

**NOTE:** This section assumes that you've signed up for Amazon's Simple Storage Service (Amazon S3). To do so, go to the Amazon S3 website at <http://aws.amazon.com/s3/>

## Creating an Amazon Simple Storage Service Bucket

AMIs are stored in Amazon Simple Storage Service *buckets*.

To create an Amazon S3 bucket for this example, we will use the Amazon S3 Firefox Organizer tool add-on for the Mozilla Firefox browser (S3Fox). The Amazon S3 Firefox Organizer can be downloaded from <http://addons.mozilla.org/en-US/firefox/addon/3247>.

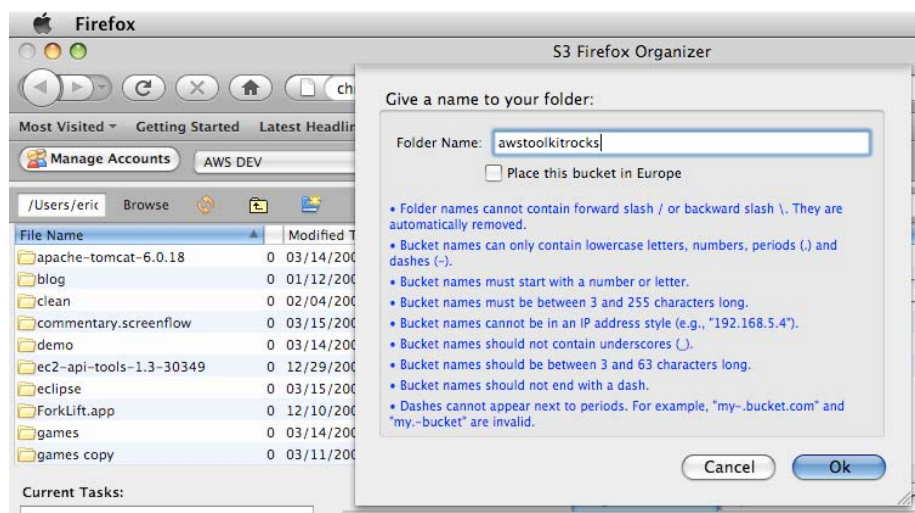
Once the S3 Organizer is installed, run it by opening the Firefox **Tools** menu and selecting **S3 Firefox Organizer**.

The S3 Organizer shows local files in a list view on the left, and Amazon S3 buckets, folders, and files in a list view on the right.

Top-level folders in your Amazon S3 account are called *buckets*. To create a new bucket, right-click in the Amazon S3 view on the right and select **Create Directory**. The **Upload options** dialog will appear, prompting you to enter a name for your bucket in the **Folder name** field.

**NOTE:** Bucket names must be globally unique.

Type a unique bucket name into the **Folder name** field, and then click the **OK** button. An Amazon S3 bucket with the specified name will be created.

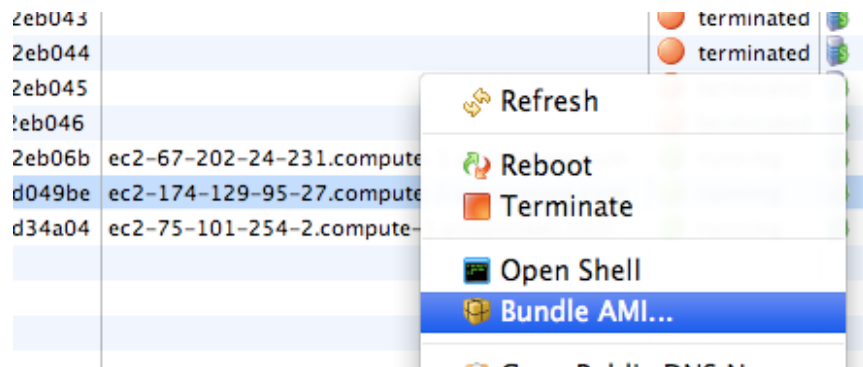


## Bundling an Amazon EC2 instance

Bundling an instance into an AMI with the AWS Toolkit for Eclipse is simple.

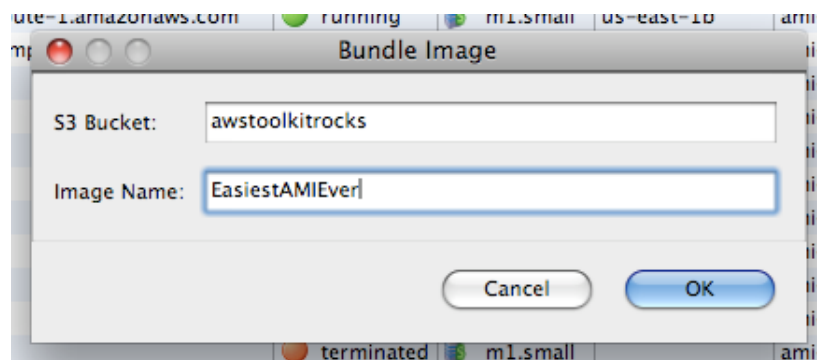
**NOTE:** Bundling an AMI instance requires that you have an X.509 certificate created and configure in the AWS Toolkit for Eclipse configuration. For more information, please see the [Adding X.509 certificates to the AWS Toolkit Configuration](#) section of this document.

To bundle an Amazon EC2 instance into an AMI, go to the **EC2 Instances** view in the toolkit, right-clicking the server instance that you wish to bundle, and click **Bundle AMI...** from the context menu.

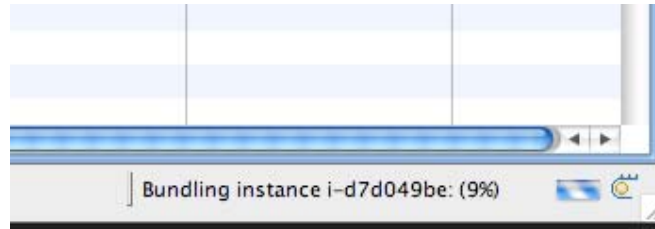


The **Bundle Instance** dialog will appear. Enter the name of the Amazon S3 bucket we created in the step above in the **S3 Bucket** text box, and then enter a name for your image in the **Image Name** text box.

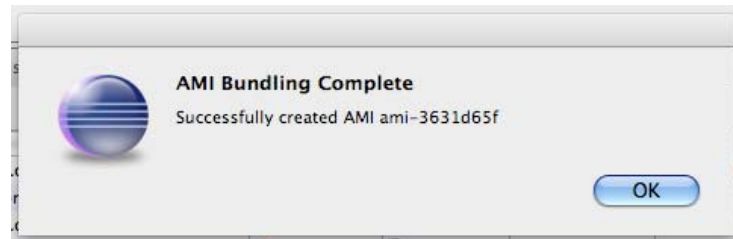
Press the **OK** button to start the bundling process.



Bundling usually takes from 10 to 15 minutes. You can monitor the bundling process in status bar of the Eclipse IDE.

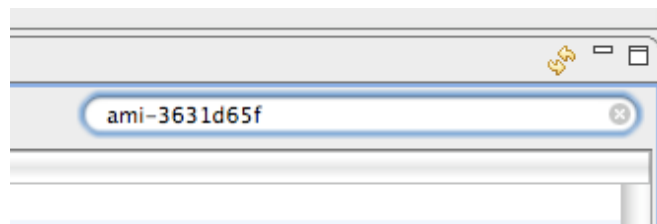


When your bundle has been created, the toolkit will provide you with a unique identifier for the AMI. Make note of this ID - we will use it to launch an instance of our new AMI.



### Testing the Bundle

Once the bundle has been created, enter the ID of the AMI we just created into the search field of the EC2 Instances view and press **ENTER**.



The AMI we just created should now be listed in the search results.

Connect to this AMI by right-clicking it and selecting **Open Shell** from the context menu.

The file that we created at the beginning of this section should still be in the instance.

### Limitations of Bundling

Bundling provides a simple way to persist instance data, but has a number of limitations:

- Amazon Machine Images are limited to 10GB in size
- You must create a new AMI every time you change data in your instance.

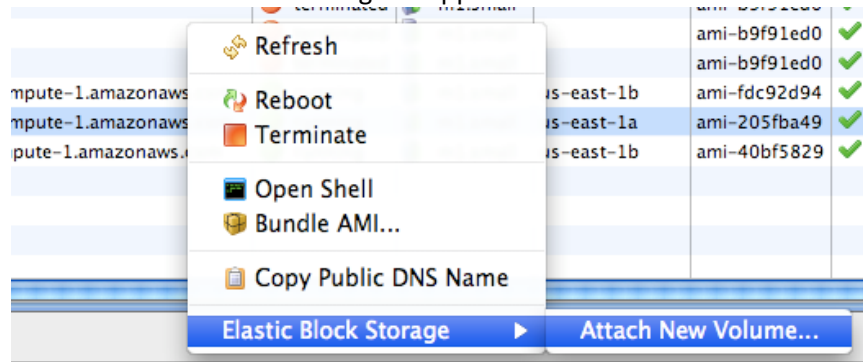
Because of these limitations, AMI images are intended mainly for storing server configurations. A more dynamic method to persisting instance data that is often used in conjunction with bundling an AMI is *Amazon Elastic Block Store* (Amazon EBS).

## Using Amazon Elastic Block Storage

Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with Amazon EC2 instances. Amazon EBS volumes are “off-instance” storage that persists independently from the life of an Amazon EC2 instance. Amazon Elastic Block Store provides highly available, highly reliable storage volumes that can be attached to a running Amazon EC2 instance and exposed as a device within the instance. Amazon EBS is particularly well suited to applications that require a database, file system, or access to raw block level storage.

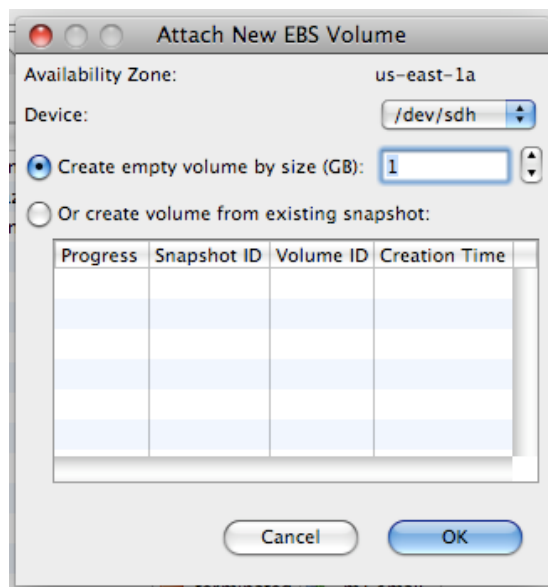
### Creating an Amazon EBS Volume

To create an Amazon EBS volume, right-click a running server instance in the **EC2 Instances** view, and then select **Elastic Block Storage** and click **Attach New Volume...** from the context menu. The **Attach New EBS Volume** dialog will appear.



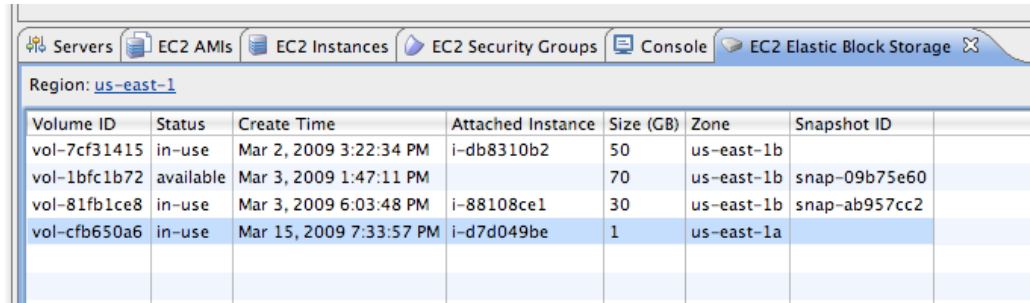
We can now configure our new Amazon EBS volume. You can create an Amazon EBS volume of up to 1 TB in size. For our example, we will just specify the minimum size of 1 GB.

Click the **OK** button to create your volume.



Open the toolkit's **EC2 Elastic Block Storage** view (if this is not already open, you can open it by clicking the Eclipse **Window** menu, selecting **Show View**, clicking **Other**, and then selecting the **EC2 Elastic Block Storage** view from the **Amazon EC2** node).

In the **EC2 Elastic Block Storage** view, find the instance ID of the server we used to attach an Amazon EBS volume. In a few moments you should see that the new volume is attached to your instance.

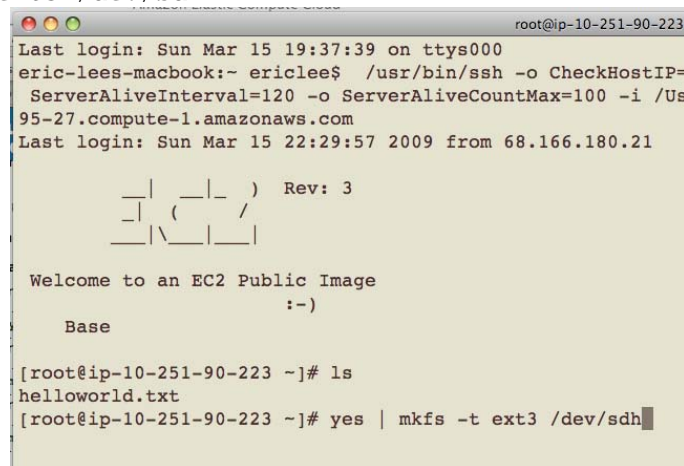


Volume ID	Status	Create Time	Attached Instance	Size (GB)	Zone	Snapshot ID
vol-7cf31415	in-use	Mar 2, 2009 3:22:34 PM	i-db8310b2	50	us-east-1b	
vol-1bfc1b72	available	Mar 3, 2009 1:47:11 PM		70	us-east-1b	snap-09b75e60
vol-81fb1ce8	in-use	Mar 3, 2009 6:03:48 PM	i-88108ce1	30	us-east-1b	snap-ab957cc2
vol-cfb650a6	in-use	Mar 15, 2009 7:33:57 PM	i-d7d049be	1	us-east-1a	

### Creating and mounting the file system

Switch back to the **EC2 Instances** view and connect to your instance. Since we attached a raw volume to our instance, we will need to create a file system on that volume before we can use it. To do this, type in the at the shell command prompt:

```
yes | mkfs -t ext3 /dev/sdh
```



```

root@ip-10-251-90-223:~
Last login: Sun Mar 15 19:37:39 on ttys000
eric-lees-macbook:~ ericlee$ /usr/bin/ssh -o CheckHostIP=
ServerAliveInterval=120 -o ServerAliveCountMax=100 -i /Us
95-27.compute-1.amazonaws.com
Last login: Sun Mar 15 22:29:57 2009 from 68.166.180.21

 _ | _ | _ ) Rev: 3
 _ | ( _ /
 _ | \ _ | _ |

Welcome to an EC2 Public Image
      :-)

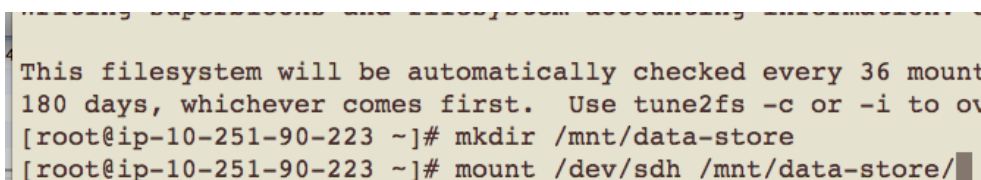
Base

[root@ip-10-251-90-223 ~]# ls
helloworld.txt
[root@ip-10-251-90-223 ~]# yes | mkfs -t ext3 /dev/sdh

```

After a few moments, your file system will be created. Type in the following two commands to mount your volume:

```
mkdir /mnt/data-store
mount /dev/sdh /mnt/data-store
```



```

This filesystem will be automatically checked every 36 mount
180 days, whichever comes first. Use tune2fs -c or -i to ov
[root@ip-10-251-90-223 ~]# mkdir /mnt/data-store
[root@ip-10-251-90-223 ~]# mount /dev/sdh /mnt/data-store/

```

Navigate into your new volume and create a new text file by typing the following commands at the shell prompt:

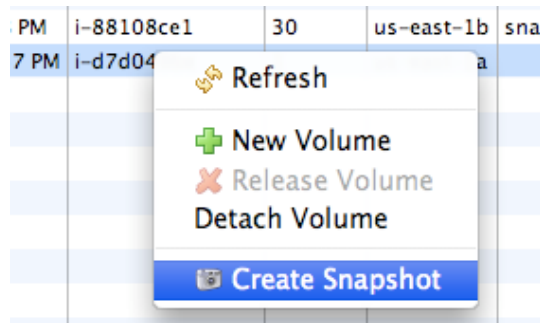
```
cd /mnt/data-store
echo 'EBS Rocks!' >helloworld.txt
```

```
[root@ip-10-251-90-223 ~]# cd /mnt/data-store/
[root@ip-10-251-90-223 data-store]# ls
lost+found
[root@ip-10-251-90-223 data-store]# echo 'EBS Rocks!' > helloworld.txt
```

Because we've created the new text file on the Amazon EBS volume, the file will persist even if the Amazon EC2 instance is terminated; just mount the Amazon EBS volume in a newly created instance to access the data again.

## Creating an Amazon EBS Snapshot

Amazon EBS provides the ability to create point-in-time snapshots of your data and save them to Amazon S3. Amazon EBS snapshots are *differential* backups; only the blocks on the device that have changed since the last snapshot was created will be incrementally saved. For example, if you have a device with 100 gigabytes of data, but only 5 gigabytes of data has been changed since your last snapshot, only the 5 additional GBs of snapshot data will be stored to Amazon S3. To create a snapshot of a volume, right-click the volume in the **EC2 Elastic Block Storage** view and select **Create Snapshot** from the context menu.



We will see our snapshot in the lower portion of the **EC2 Elastic Block Storage** tab.

Java EE - Eclipse

EC2 AMIs | EC2 Instances | EC2 Elastic Block Storage | EC2 Security Groups

Region: us-east-1

Volume ID	Status	Create Time	Attached Instance	Size (GB)	Zoned
vol-7cf31415	in-use	Mar 2, 2009 3:22:34 PM	i-db8310b2	50	us-east-1a
vol-1bfc1b72	available	Mar 3, 2009 1:47:11 PM		70	us-east-1a
vol-81fb1ce8	in-use	Mar 3, 2009 6:03:48 PM	i-88108ce1	30	us-east-1a
vol-cfb650a6	in-use	Mar 15, 2009 7:33:57 PM	i-d7d049be	1	us-east-1a

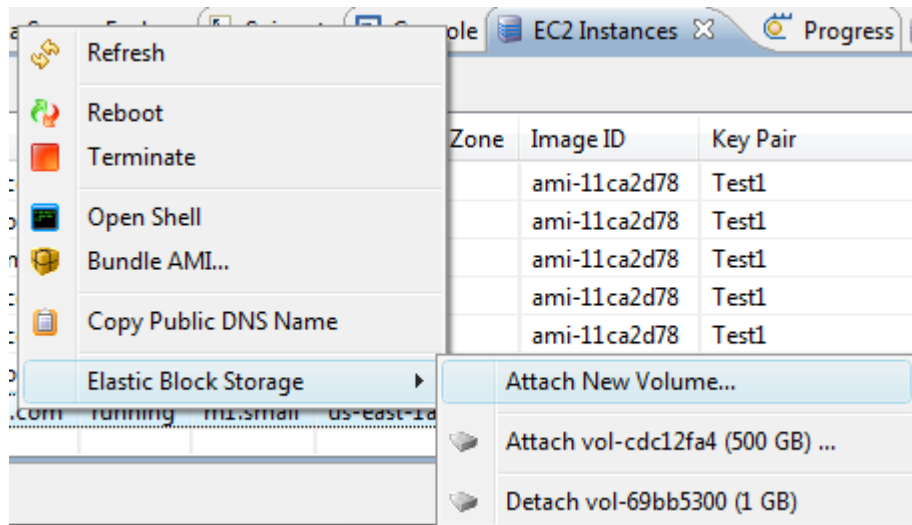
  

Progress	Snapshot ID	Volume ID	Creation Time
completed	snap-a545b0cc	vol-cfb650a6	Mar 15, 2009 7:39:15 PM

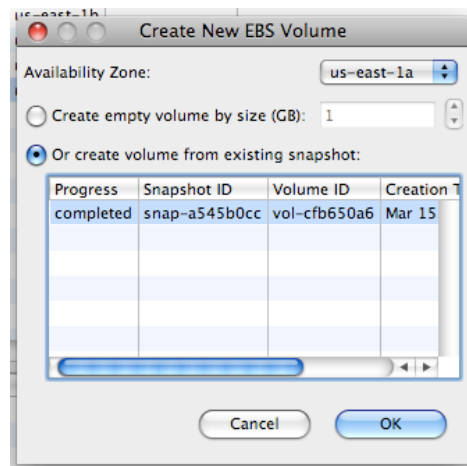


## Using an Amazon EBS Snapshot

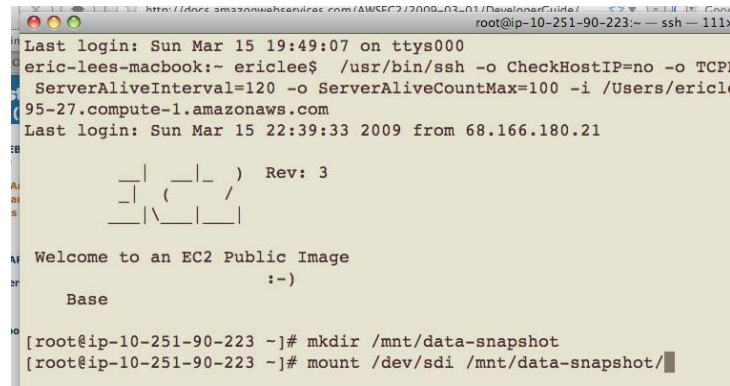
To use a snapshot, create a new volume and attach it to a running instance. While you can create an Amazon EBS volume from the **EC2 Elastic Block Storage** tab and then attach it to a running instance using the **EC2 Instances view**, you can complete both tasks by right-clicking on an instance in the **EC2 Instances** view and selecting **Elastic Block Storage** and then **Attach New Volume...** from the context menu.



In the **Create New EBS Volume** dialog, select the option to create a volume from an existing snapshot and then choose a snapshot, and then press **OK** to create the volume.



We can mount this volume in the same manner as described in the [Creating an Amazon EBS volume](#) section above. Enter the two commands shown below at the command shell prompt to map the new volume created from the snapshot to **/mnt/data-snapshot**.



```
http://docs.amazonwebservices.com/AWSEC2/2009-07-01/DeveloperGuide/...
root@ip-10-251-90-223:~# ssh - 111x
Last login: Sun Mar 15 19:49:07 on ttys000
eric-lees-macbook:~ ericlee$ /usr/bin/ssh -o CheckHostIP=no -o TCP
ServerAliveInterval=120 -o ServerAliveCountMax=100 -i /Users/ericle
95-27.compute-1.amazonaws.com
Last login: Sun Mar 15 22:39:33 2009 from 68.166.180.21

  _|  _|_ ) Rev: 3
  _| (  _| /
  _|\_||_|

Welcome to an EC2 Public Image
      :-)

Base

[root@ip-10-251-90-223 ~]# mkdir /mnt/data-snapshot
[root@ip-10-251-90-223 ~]# mount /dev/sdi /mnt/data-snapshot/
```

Navigate to the newly mounted volume by entering `cd /mnt/data-store` at the command shell, and then type `ls` to list the directory. This new volume will have the same text file as the one we created in the previous example.

## Conclusion

Amazon Elastic Cloud Computing enables you to increase or decrease computing capacity within minutes, not hours or days, and only pay for what you need, when you need it. The AWS Toolkit for Eclipse allows developers to harness this elastic computing power more easily than ever.