Using Amazon Elastic Compute Cloud for 4D Benchmarking and Testing

By Thomas Maul, 4D Germany

Technical Note 09-18

Table of Contents

Abstract

Did you ever have a potential customer demand that your application run fast enough with 20 or 200 users, then go back to your office and realize you only have a few computers to test with? Using cloud computing, you can rent computers for extended periods of time, paying just a few cents per hour. This tech note explains how to setup a test environment using Amazon Elastic Compute Cloud and gives ideas on how to prepare your solution to run automatic and realistic user simulations.

Introduction

In this tech note, we use Amazon's cloud service, Elastic Cloud Compute (EC2), to "rent" multiple machines and demonstrate how this is a cost effective solution for testing a multi-user system without having to buy expensive equipment. Ideas on how to prepare your application for automation and customer demonstration are also provided. First, here is some general information on cloud computing and Amazon's EC2.

Cloud computing

From Wikipedia, the free encyclopedia:

> "Cloud computing is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the internet. Users need not have knowledge of, expertise in, or control over the technology infrastructure "in the cloud" that supports them."

In short, a service like EC2 provides a cloud that customers can tap into to run virtual machines on. The specifics of how the cloud is built and maintained are invisible to customers; all that is seen when logging into these virtual machines are the normal Windows operating system desktop as if running on a single, local machine (or many machines if running many instances).

Amazon Elastic Compute Cloud – EC2

From http://aws.amazon.com/ec2/

> " Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers.
>
> Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale

capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate themselves from common failure scenarios."

Amazon "rents" computers starting at $0.125 up to $1.20 per hour as of April 2009 (see http://aws.amazon.com/ec2/#pricing for more information). These are virtual machines, from small (similar to a 1 GHZ XEON with 1.7 GB RAM) to large (4 cores with 2 GHZ XEON, with 7.5 GB RAM), up to 8 cores with 2.5 GHZ. See http://aws.amazon.com/ec2/instance-types/ for a more in-depth description of how Amazon measures EC2 systems.

The "small" machines are powerful enough to simulate running clients, while using the "large" high-end CPU for the server – which is needed only once – leads to a relatively affordable option for simulating a multi-user system.

Beyond the rates as described, there is no setup fee. You can rent 20 small machines to simulate 20 clients together with a 4 core server machine for a 45 minute test for less than $5! Amazon charges traffic fees if data goes outside of their network area, but currently internal traffic is free, so for a client/server benchmark, you could have little-to-no external data transfer.

While Amazon writes that you can scale to thousands of servers, a new user has a maximum of 20 concurrent instances. This limit can be increased upon written request (with a 3-5 day response time) by visiting:

http://aws.amazon.com/contact-us/ec2-request/

Using EC2 with 4D for benchmarking / user emulation

This tech note shows a way to prepare your application to run automated user emulation with multiple machines using EC2. While professional software for User Interface Testing is available, it is usually quite expensive to control 50 or 200 machines concurrently. By using **POST KEY**, we show a simple-to-install way to control an application using the keyboard.

The next sections focus on the following tasks to accomplish this:

- Define a stress test plan with your customer
- Prepare your application for automated user emulation
- Prepare an Amazon EC2 instance to be used as Client
- Prepare an Amazon EC2 instance to be used as Server
- Run the user test…

Define a stress test plan with your customer

This could easily be the hardest job - do not underestimate this step! Your customer is not really interested in how many records you can create per second, as this number is not really meaningful to them. They are likely more interested in "how many users can you handle", which is a business task matter, depending on the way the company is working and your application is storing data.

As an example, a mail order house does not have 100 people doing accounting, but may have a large call center taking orders by phone and entering new customers. Identifying the parts of your application that are used from a large amount of users is the challenge, not which methods are run once a day.

It helps to visit the user and watch the way they work. Then you need to identify their working speed. Take a stop watch and check how long they need to enter an order or a new customer address.

Finally you need to discuss this list with your customer and write a test plan. What parts of your application with how many users, in what speed. Be realistic here: It makes no sense to say "we have 100 people entering new orders at 1 record per second ", if they do it manually. If they currently have 10 users and they want to be sure they can grow to 100, you need to do the same job in the same speed, just with more clients.

Check how fast they are in one minute. No human being can maintain their fastest speed over an hour, let alone a full day. You could still simulate that, showing the customer that the system is stress resistant enough to run even in worst-case situations.

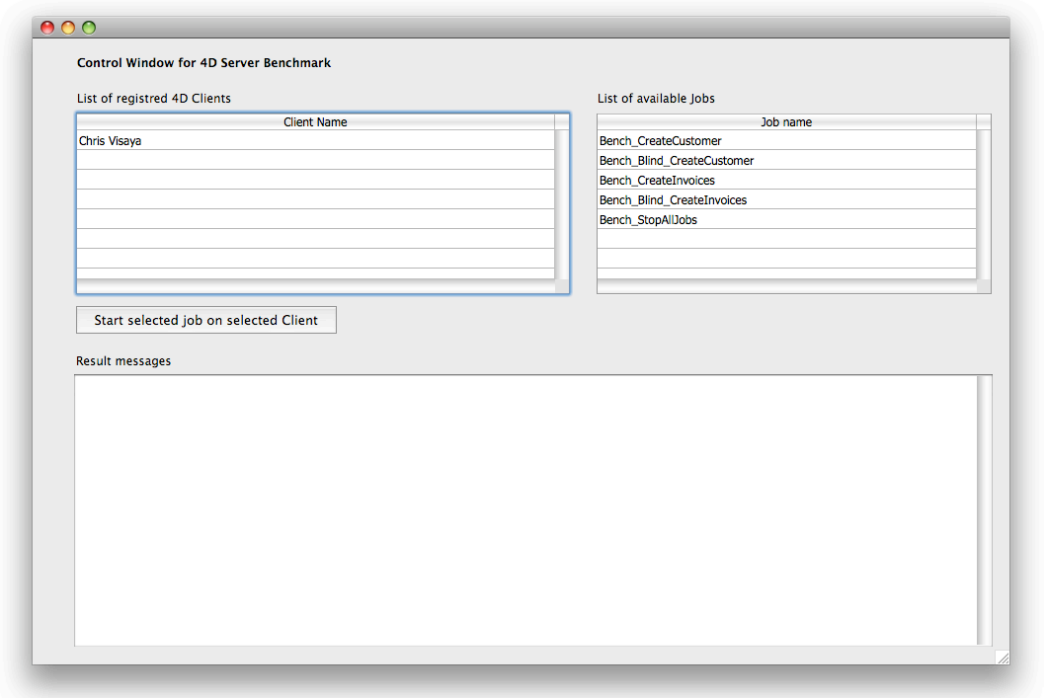
Prepare your application for automated user emulation

Now start to test your application. Check if these jobs can be fully handled using the keyboard, without any mouse usage. If not, you need to assign keyboard short cuts to the buttons to allow it. This is important for automated data entry.

This is a good time to take a look at the enclosed example application. It is based on "4D Invoice", a demo published with an older version of 4D during a time when the Internet was widely unknown and a cloud was something in the sky…

Start the application with 4D Server and connect a client to it. While you can run both on the same computer, it is easier to understand and handle doing that on two machines.

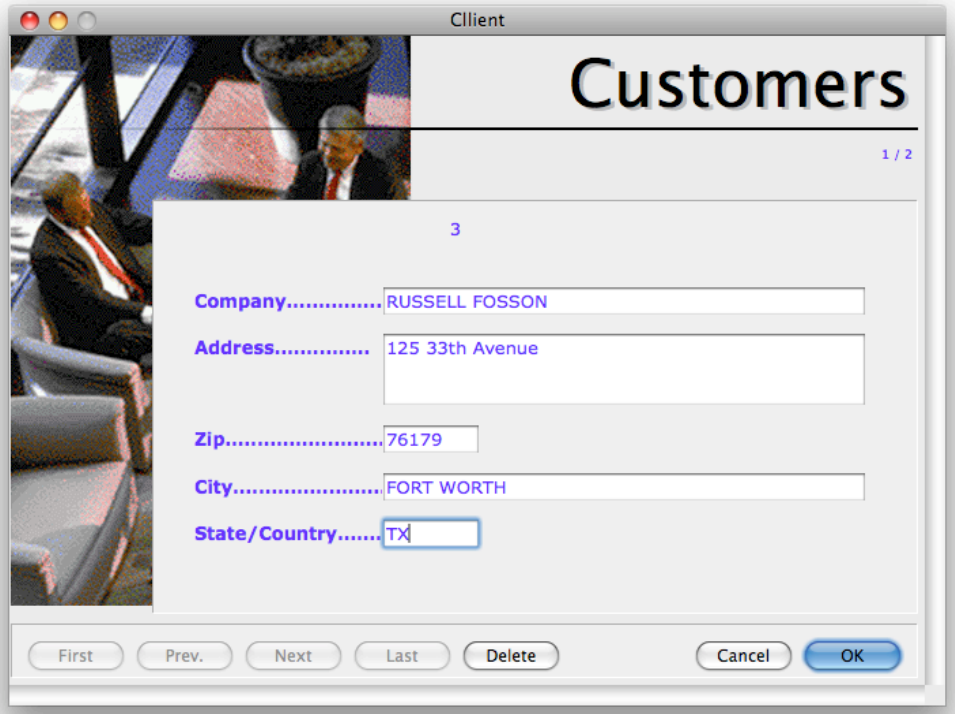
The server opens a control window automatically:



The machine name of every connected client is displayed in the upper left list box. Select the client and select "Bench_CreateCustomer" on the upper right list. Now click on "Start selected job on selected client". You could start this job on one client, or have multiple clients perform the same task with a single click.

The launch needs about 15 seconds, as it reads some test numbers from disk into arrays. The data entry form for our customers is then open. The application automatically enters customer data at a rate of about 4 characters per second. It randomly produces companies by combining typical US first names and last names, a street name and number and finally a city from the US and a zip code. As this is faked information, do not expect realistic addresses.

The following screenshot shows what you see when the system is creating customers:



The folder "resources" contains text files with "Firstnames.txt", "Lastnames.txt" and "Zipcodes". These contain the data described above in text format, which is very useful for such automatic tests, as the data looks much more real than "abc1234". The files contain thousands of first and last names and all cities in the United States.

The data is in Public Domain, loaded from http://www.census.gov.

You may want to show that this is a real demo to your customer by entering some characters using the keyboard to see that they appear on the form. This is the same form a user would be using if they were manually entering data themselves. An additional process is filling random data in text, passing that to another process which uses **POST KEY** to post each character, then waits a few ticks, posts the next character and repeats. When the address is done, the ENTER key is "pressed" to save the record. And last, Ctrl-N is pressed to add a new customer and so on. There is a short delay when the address is fully entered, as a real user would take at least a quick look at the data before saving.

If you have two computers available, start another 4D in remote mode on the server itself and bring the server back to the front to launch a job on the second machine. If not: Let it run for a few more minutes, then stop by clicking "Bench_StopAllJobs" on the server and run it on that client. The client finishes entering the current address and then stops.

There is a second example enclosed, run "Bench_CreateInvoices" to test. Be sure to run Bench_CreateCustomers first as it is necessary to associate invoices to customers. This procedure creates new invoices by randomly selecting a customer. Then it creates 1-3 invoice items with randomly selected products; there are 10 products in the system. Feel free to enter/import more. It does the usual job by calculating the invoice item total and invoice total. This is part of the normal form method, nothing additional added here for the test, as we simulate a user by posting keys as a real user would have entered them.

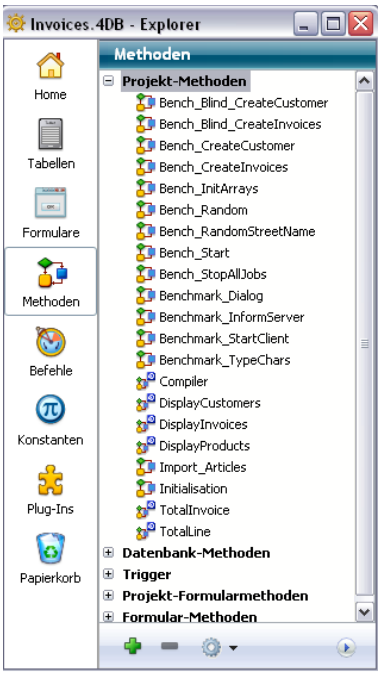
Do not try to press keys while this is running, as both customer ID and product ID must be valid codes. Entering a nonexistent code causes a request to create a new customer or product and the automated data entry is not designed to accommodate this -- it simply keeps entering data and saving records. If you do accidentally enter data while the system is creating invoices, simply stop the job and restart it.

With two machines running, you can see that customers and invoices are created in the application. When you are done, use the server control window and select all clients and run "Bench_StopAllJobs".

Note: *The rest of this section includes tips and ideas for incorporating automated user entry into your own applications. The included sample database is ready to demonstrate all of the steps as listed above both in a standard client-server environment and using Amazon's EC2 service. See "Prepare an Amazon EC2 instance to be used as a Client" if you are interested in trying it out.*

When you are done, quit all clients and the server and run 4D in single user mode and open the invoices demo. Go to design mode and open the explorer.

There are a couple of project methods worth looking at:



You need to install/copy the four methods named "Benchmark_" into your application. They are generic, no need to modify their content.

While you could copy the content manually, the easiest way is to launch a 2nd instance of 4D (duplicate 4D's folder, then simply start the copy to have two versions of 4D running) and open your application with the second instance.

Select the 4 methods starting with "Benchmark_" and drag&drop them into the explorer of your application.

Also be sure to copy the Project form "Benchmark_ServerControl" into your app.

The methods assist in running the automated tasks, but it is up to the developer to prepare their data and application as necessary.

The following is an example of how to prepare an automated task:

```
$job:="N"+Char(1)+"hello"+Char(Enter)
ARRAY LONGINT($specialkey;1)
$specialkey{1}:=Command key mask
Benchmark_TypeChars($customerprocess;$job;->$specialkey;10)
```

This example posts the keys: N [1] h e l l o [ENTER]. The array "$specialkey" allows us to pass control keys, such as Command/Ctrl or Option/Alt, adding support for your button shortcuts. The array is automatically reformatted to fit the size of the string, so we do not need to bother here with the correct size. The method Benchmark_TypeChars does all of the keyboard entry work. The first parameter is a process ID, which receives the Post keys. As you need to control your application, the user interface is run in another process.

The second parameter, $job, is text containing all characters to post.

The third parameter is a pointer to the array with the command keys, the 4th the delay time between each character in ticks (here 10 ticks = 1/6 second).

Char(1) has a special meaning: It is interpreted as a 1 second pause. This allows opening the dialog after "pressing" the button. In real life, a user would most likely wait untill the dialog fully appears. You may need to add an extended delay for complex screens.

Now let's take a look at the whole method running the simulation to enter invoices:

```
  ` benchmark demo Bench_CreateInvoices
  ` create new customers

C_LONGINT($maxcust;$keydelay)
$maxcust:=100  ` create 100 customers, then stop!
$keydelay:=10  ` 10 tick = type 6 character/second.
C_BOOLEAN(<>runTest)
<>runTest:=True

$customerprocess:=New process("DisplayInvoices";512000;"DisplayInvoices")

READ ONLY([Customers])
ARRAY TEXT($product_IDs;0)
READ ONLY([Products])
ALL RECORDS([Products])
SELECTION TO ARRAY([Products]ID_Product;$product_IDs)
$maxproducts:=Size of array($product_IDs)

$time:=Current time
$counter:=0
While (($counter<$maxcust) & (Not(Process aborted)) & (<>runTest))
   $counter:=$counter+1

   $countcust:=Records in table([Customers])
   $cust:=Bench_Random ($countcust)-1
```

```
   GOTO RECORD([Customers];$cust)  ` this assumes that no record is deleted!
   ` if you cannot assure this, use goto selected record or another way
   $company:=[Customers]Company
   $itemcount:=Bench_Random (3)
   ARRAY LONGINT($item_id;$itemcount)
   ARRAY LONGINT($item_vol;$itemcount)
   For ($i;1;$itemcount)
         $item_id{$i}:=Bench_Random ($maxproducts)
         $item_vol{$i}:=Bench_Random (10)  ` sell 1 - 10 pieces
   End for

     ` prepare the next job
     ` add Ctrl+N to open a new "create customer" window
     ` then enter the fields
     ` press "Enter" to finalize
   ARRAY LONGINT($specialkey;1)
   $specialkey{1}:=Command key mask
     ` wait 1 second before we start, so the dialog appears. Wait 1 seconds
after the last char to simulate user, then enter (goto back to list), wait 3
seconds
   $job:="N"+Char(1)+$company+Char(9)
   For ($i;1;$itemcount)
         $job:=$job+"+"
         ARRAY LONGINT($specialkey;Length($job))  ` resize array to fit...
         $specialkey{Size of array($specialkey)}:=Command key mask
         $job:=$job+$product_IDs{$item_id{$i}}+Char(9)+String($item_vol{$i})
   End for

   $job:=$job+Char(9)+Char(1)+Char(Enter )+Char(1)+Char(1)+Char(1)
   ARRAY LONGINT($specialkey;Length($job))  ` resize array to fit...

   Benchmark_TypeChars ($customerprocess;$job;->$specialkey;$keydelay)
End while

  ` close last window! Send escape to do that...
ARRAY LONGINT($specialkey;1)
$specialkey{1}:=0
Benchmark_TypeChars ($customerprocess;Char(27);->$specialkey;$keydelay)

  ` send finalize message to server
$time2:=Current time
$message:=String($counter)+" Invoices created in "+Time string($time2-$time)
$p:=Execute on server("Benchmark_InformServer";512000;"Bench Inform";$message)
```

This method handles simulating invoice data entry. It first starts the DisplayInvoices method, normally called by the user using the menu, as a new process. Depending on your application, it could be easier to start specific tasks like this – or to control the user interface and start jobs using menu short cuts.

We fill an array with all available product ID's, as we assume products to normally be a static list. For customers, we always use the current existing records. However, more customers can be added while we are creating invoices, so the amount can grow.

The method loops until the preset amount of records are created - or the process is aborted (because somebody killed the process or requested to shutdown the computer) or till the "end the test" flag is set. The method "Bench_StopAllJobs" sets the flag variable <>runTest to false.

In the loop, we randomly select a customer. The method Bench_Random is a simple extension to the 4D function Random, which returns only numbers between 0-32767. We may have more customers, so this range may not be big enough, but for testing purposes it is sufficient.

Two arrays are then filled with a random number of elements consisting of random product codes and numbers.

Now we can fill the $job variable with all of the characters to emulate user input. Here the Ctrl+"+" combination is a shortcut for the "add subrecord" button. Finally we call Benchmark_TypeChars to start the typing.

When all is done, we use Execute on Server to start Benchmark_InformServer passing a message, which is displayed in the lower list box on the server control window.

Write your own method – and test it. You could start a single user, executing the method directly. This speeds up debugging. When you think it works fine, try with 2 or 3 clients in your local network.

As soon this runs fine, add more jobs if needed.

Depending on the number of requested users, you may want to add some "blind" jobs to simulate more users than computers available. Even though Amazon machines are cheap and cost a few cents per hour, they still cost money. Even the small version is fast enough to run more jobs than typing 6 characters per second.

As soon you have convinced the customer with real user entry simulations, you can run additional jobs in the background, doing the same job in the same speed, just without user interface. Of course you need to write this jobs from scratch. See "Bench_Blind_CreateInvoices" as an example.

A good possible ratio is to run on each user machine 2-4 blind jobs, to multiply the effective number of users per machine.

Prepare an Amazon EC2 instance to be used as a Client

If you are happy that your application is ready to be tested, the next step is to sign up for Amazon's web services, EC2 and Simple Storage Service (S3). From the outset this may appear to be an involving process, but rest assured that getting set up can be done relatively quickly and, if done correctly, only needs to be performed once per instance type. Here is a short list of requirements that you should have to start:

Requirements

- Elasticfox and Firefox
- Sign up for Amazon Web Services (EC2 and S3)
- Microsoft Remote Desktop
- Application for testing (Invoices.4DB to test)

Setting up Elasticfox and Amazon Web Services

There are several ways to create EC2 instances such as by using a terminal session, or using Web Services. However we are going to avoid these by using a nice GUI, Elasticfox. Elasticfox is a Firefox plugin, so if you do not already use it, first step is to install Firefox. (http://www.mozilla.com).

This can be done on either Windows or Mac OS. Even though Amazon EC2 only launches Windows and Linux/UNIX operating systems, you can still control the virtual machines using Elasticfox on a Mac.

We suggest to take a look at the Elasticfox "Getting Started Guide":
http://developer.amazonwebservices.com/connect/entry.jspa?externalID=1797

Install Elastixfox by clicking this link in Firefox:
http://s3.amazonaws.com/ec2-downloads/elasticfox.xpi

The next step is to signup for two Amazon Web Services (AWS), EC2 and S3. EC2 is the cloud service that Amazon provides and is what we use to launch virtual machines. S3 is their storage solution and can be used in tandem with an EC2 instance to access data that you want to persist outside of specific instances. Go to the AWS homepage (aws.amazon.com) and see the Getting Started Guide for more details. Signing up for these services requires a credit card that will be billed on a monthly basis instead of a per-usage basis.

Once you are signed up, you need to enter the account data in the "Credentials" dialog at the top of the Elasticfox window. Note that you do not need to enter your password here, but the Access Keys, which can be retrieved from the "Your Account" page from aws.amazon.com.

The last step in this section is to use Elasticfox to create a KeyPair. Use the "Key Pairs" tab in Elasticfox to do so, following the Getting Started Guide.

Note that you need to decide now which region to use. Currently, Amazon offers "us-east-1" and "eu-west-1". Whatever you choose, you must use it for all following items. If you decide to run server and client in different regions it may become expensive, so make sure you use the same for all.

For European users,"eu-west-1" may have better latency times, but the prices are higher. As long as client and server run in the same region and you connect

via Remote Desktop, the region is not an important issue. Therefore, from a financial standpoint, it is suggested to use "us-east-1" even from Europe.

The information in this section – as well as the information in the Getting Started Guide - should be enough to get Elasticfox and your AWS account set up.

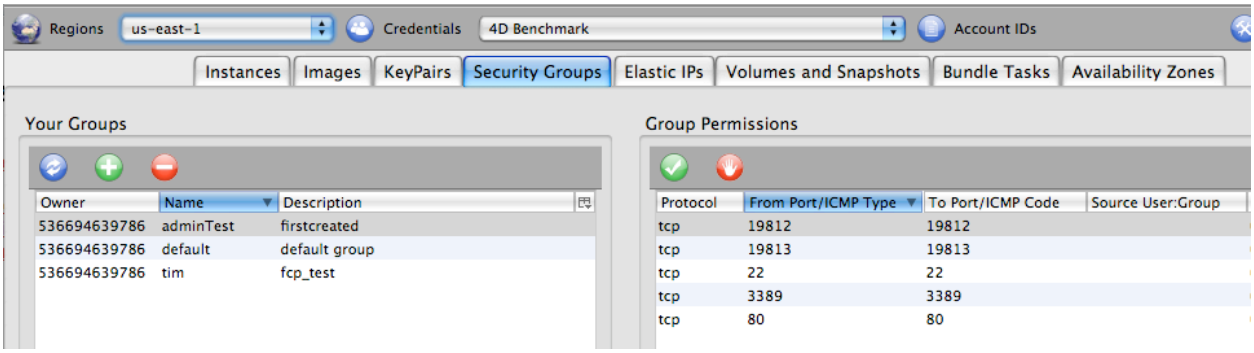
Create an Instance to be used with 4D as Remote Client

We must now set up an instance to be launched with as our clients. The following steps are based off of the "Tutorial #1: Running an Instance" section from the Getting Started Guide with some 4D-specific modifications. It is suggested to keep the guide and this tech note handy and place them beside each other for quick cross-referencing.

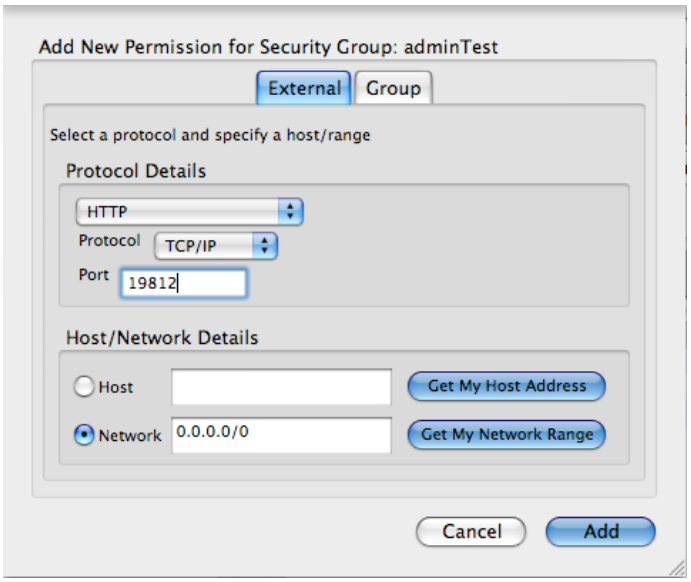
Step 1: Setting up a Security Group

Follow the steps 1-6 from the guide. They are abbreviated here for reference:

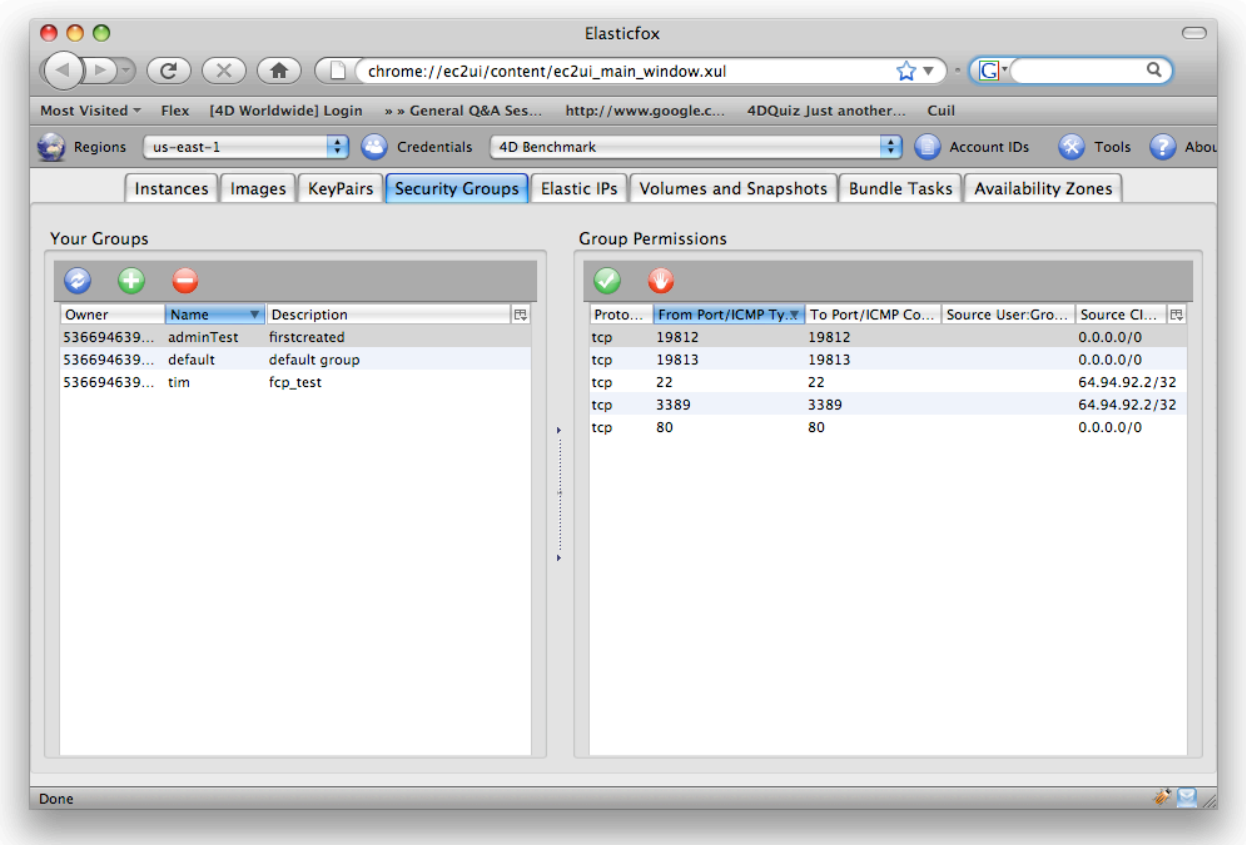
1. Launch Elasticfox by selecting Tools->Elasticfox
2. Navigate to the Security Groups tab



3. Click the "+" icon in the "Your Groups" section to create a group
4. Select a name and use the default options then confirm
5. With that highlighted, click the check mark icon in the "Group Permissions" section on the right to change port access
6. Select HTTP protocol, network radio button with value "0.0.0.0/0" for port 19812. Repeat steps 5 and 6 for ports 80 and 3389



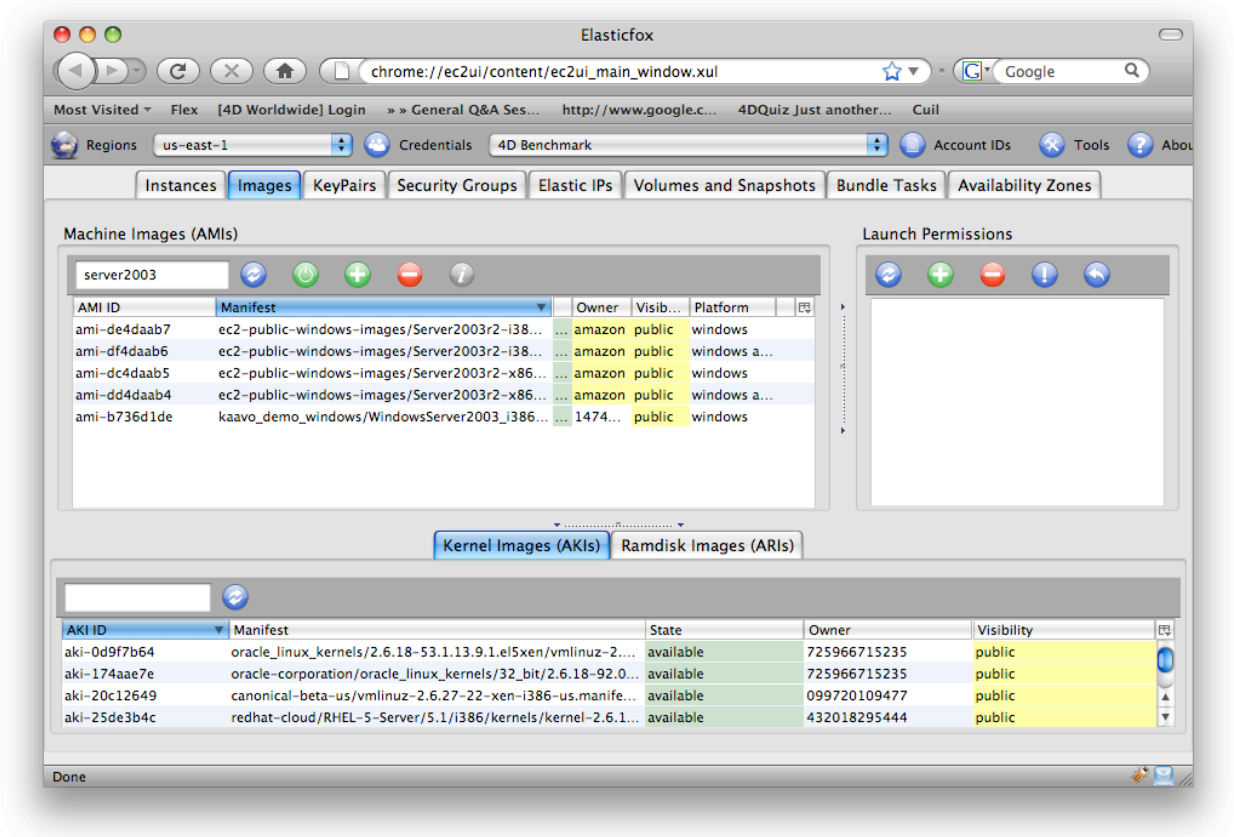
As the guide states, we need to allow port 3389. Allowing the default 4D Application Server port, 19813, to be available is not necessary for the client, but in the interest of simplicity, we are going to use the same security group for both client and server. We also opened port 80 in case you want to run 4D's web server. In short, you may open up ports that you are going to use for your tests but for this demonstration, ports 3389 and 19813 are sufficient. The final setup should look similar to this screen:



Step 2: Choosing an Amazon Machine Image

There are many Amazon Machine Images (AMI) for you to select as your "base". Elasticfox displays this in the Images tab. First, narrow the field by querying for an appropriate machine. As of this tech note, Amazon offers only Windows 2003 Server images. Enter "Server2003" to get 4 results: two with 32 bit, two with 64 bit, each either with or without authentication. These results may be different in the future. Keep in mind that images offered by Amazon start with "ec2-public", as there are also images offered from 3rd parties.

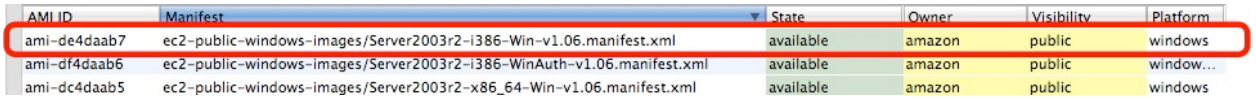
For using 4D in remote mode, you may use a 32bit version of Windows, to be able to use their small machine (cheapest). For the server you could select a 64 bit image, though this example works fine with both a 32 bit machine for client and server. The image below displays what is shown when searching for AMIs that contain "Server2003":



It is not necessary to use the AMIs with "Auth" in their names, as this feature is unneeded.

This example uses the image "ec2-public-windows-images/Server2003r2-i386-Win-v1.06.manifest.xml".

Click on the image to select it.



Step 3: Launching an Instance

The following steps go over how to launch the instance and the screenshot at the end of this section can be used for reference.

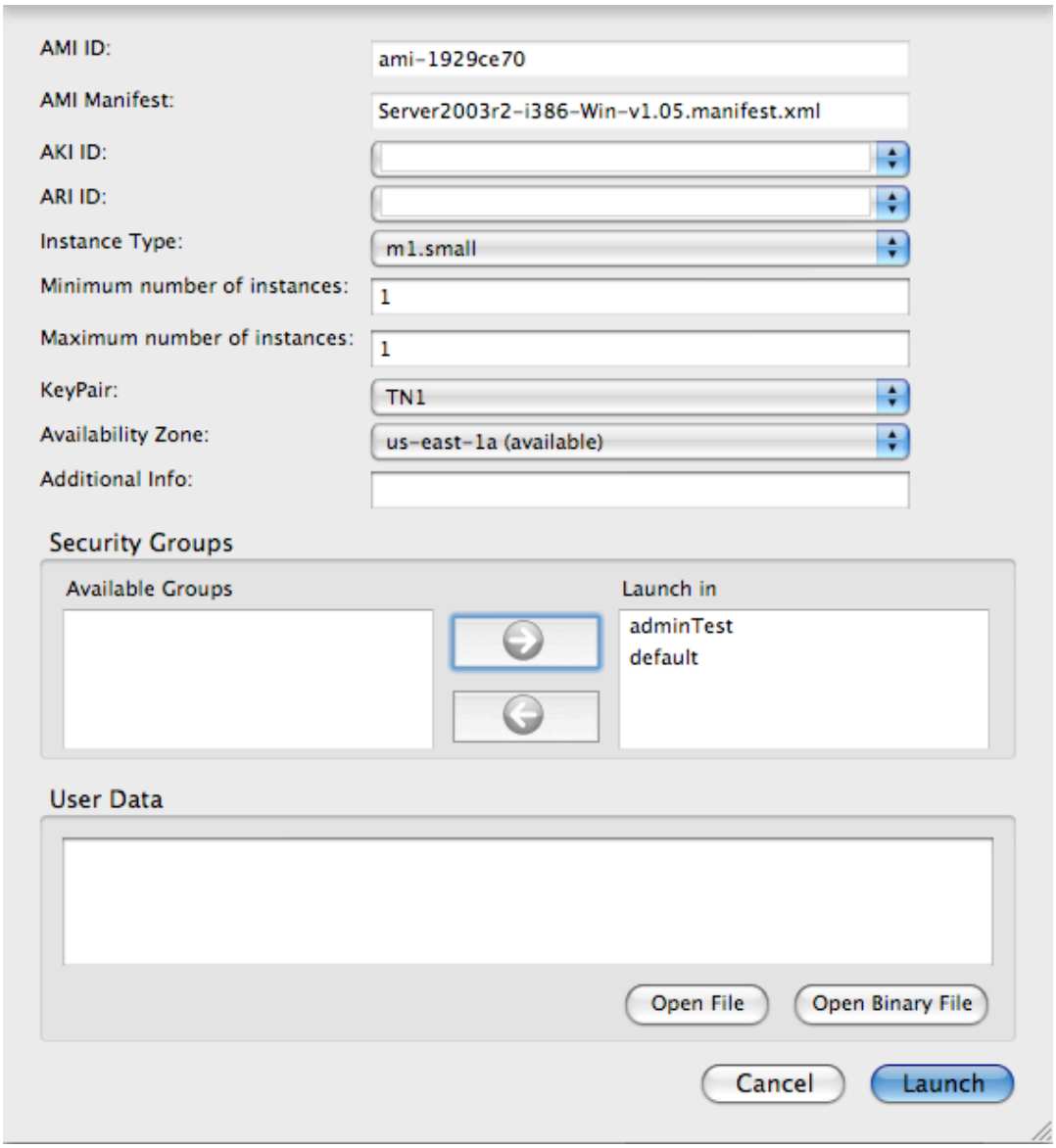
1. Right click the selected image and click on "Launch instance(s) of this AMI".

2. Select the KeyPair that was generated in the “Setting up Elasticfox and Amazon Web Services” section.

3. Select your created Security Group (in the left list box) and click the -> arrow button to move it to "Launch in". This is a very important step, if you miss that, you cannot use Remote Desktop to connect to your machine!

    a. Select the Instance Type. For 4D as a remote client, "m1.small" is sufficient. For the setup we need only one instance running, so keep the 1 for maximum number of instances. This field allows you later to run 20 instances of 4D with one single click.
    b. Select a specified zone, but use the same zone for all started instances.

4. Click Launch



5. The instance eventually shows up in the Instances tab with a status of "pending". You must wait until the status is displayed as “running”. This may take up to several minutes. In that time, a virtual machine is created for you and your operating system is launched.

Step 4: Connecting to the instance (departure from guide)

Once the instance is displayed as running, right click it and select "Show console output". When the text "Windows is ready to use" appears, your instance is fully running.

Right click the running instance again and select "Get Administrator Password". This copies a randomly generated password to the clipboard, as this image is a public image to be used from everyone. It would be wise to paste it to a static document on your local machine for future reference.

Note that this step fails if the instance is not fully running.

Step 5: Launching an Instance

Start Microsoft Remote Desktop to connect to your instance.

On Windows XP/Vista, as this tool is preinstalled, simply right click in ElasticFox the image and select "Connect to Public DNS Name". This launches launch Remote Desktop with the needed data to connect.

On Mac you have to install Remote Desktop first:

http://www.microsoft.com/mac/products/remote-desktop/default.mspx

Right click the instance in Elasticfox the and select "Copy Public DNS Name to clipboard". Then launch Microsoft Remote Desktop and paste the DNS Name from the clipboard to connect to the instance.

Login as "Administrator" and use the password from the previous step.

At this point you should have successfully gained access to the virtual computer. If this is your first time with an Amazon instance you may want to browse around a little bit. You will see that it behaves like any other computer running Windows Server 2003. This is an English version of the OS; there are no localized versions available, but this does not matter for testing/running 4D.

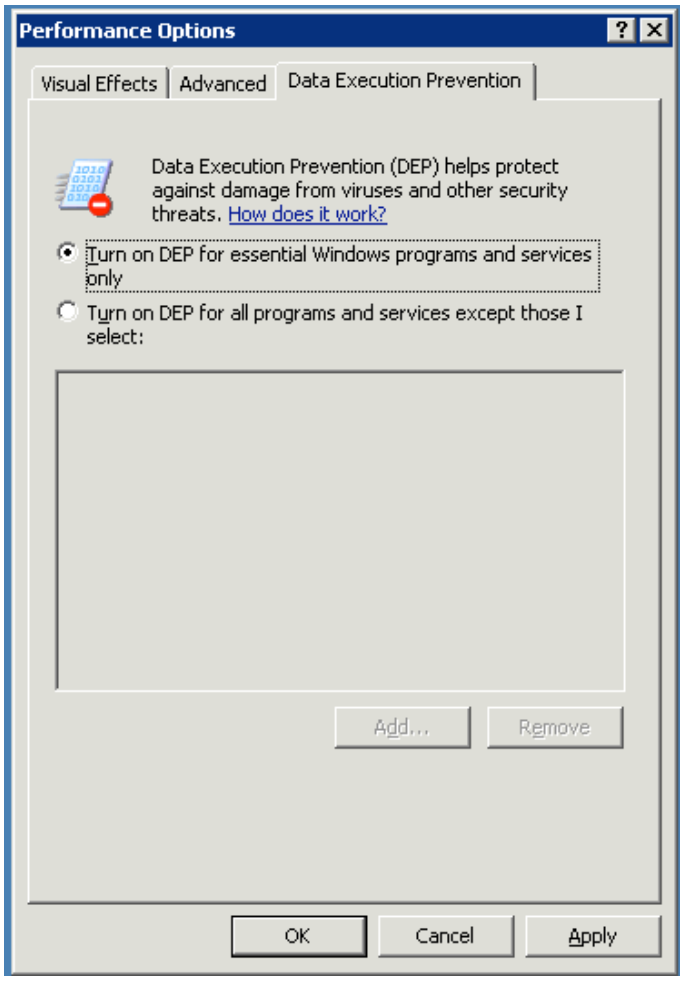
Step 6: Prepare the Instance for 4D

This step is 4D-specific and is not part of the Amazon guide.

1. A key step is to change the Administrator password or at least remember the default password that was assigned. Needless to say, without the account password, you can not log in to the instance. We suggest to note the chosen password together with the other Amazon account data. This is a nice trap - if you miss that, you will end up with your own image, perfectly created, but unable to login.

   To change password via Remote Desktop, press "CTRL+ALT+End". Note: The 'end' key is used, not 'del', as on a Windows computer ctrl+alt+del opens your local security dialog, while ctrl+alt+end is forwarded to the remote machine.

2. If you are running an 4D v11 SQL Release 3 or earlier, this step is necessary. If using Release 4 or later, you can skip this step. We are going to disable DEP to allow running compiled databases.
   - Click Start, click Run, type sysdm.cpl, and then click OK.
   - On the Advanced tab, under Performance, click Settings.
   - On the Data Execution Prevention tab.
   - Click Turn on DEP for essential Windows programs and services only to select the OptIn policy.



If you miss this step, interpreted structures work, but compiled ones may not run correctly. See http://kb.4d.com/search/assetid=51535.

3. Install 4D Client. You may use the installer from www.4d.com, copy it from a connected drive using on Windows Remote Desktop or via your own FTP site. In our testing, an FTP site was about 4-8 times faster than remote desktop, so the additional preparation time may be worth spending.

4. As this machine is running as client, there is no need to care for license numbers, as 4D runs without it in remote mode.

5. We need to find a way to tell this client which IP address to connect to at startup. There are several ways to do that. For example, the "User data" feature from EC2, an "Elastic IP" address which increase your Amazon bill, or use a DNS entry which we can control. The last method is a simple and cost-effective solution and is described here.

   Select a DNS Service, such as dyndns.org and create an account and then a host entry. For the initial setup of the host, the IP address you enter is irrelevant as this is going to change once we acquire the server's IP address. Select a host name and take note of that. In this example we use "test4DServer".

6. Next we are going to create a 4Dlink file to specify the server to connect to. There are a couple of ways to do this. Copy a 4D Favorites file from your local hard disk to the remote machine or use a text editor such as Edit.exe to create one on the remote machine. The contents of this file should be:

```
<?xml version="1.0" encoding="UTF-8"?>
<database_shortcut is_remote="true" server_database_name="mytest.4DB"
server_path="test4dServer.dyndns.org"/>
```

   Note that the server_path is set to the host name that we created from the previous step. Save it as "Autostart.4DLink" on the desktop or directly in the C:\ volume. To test it, drag and drop the 4DLink file onto 4D.exe. This launches 4D and automatically tries to connect to a server, which is not yet running, so you will fail with a -10002 at that point. The point is that if it is *trying* to connect, the 4Dlink file is being applied correctly.

7. Use Edit.exe to create a batch file with the following content. It is one single line with a blank between the two paths. As the paths contain blanks, you must use quotes around them. The first path shows the position of the 4D.exe, in this example it was placed in a folder named "4D" on the desktop. The second path shows the folder to the Autostart.4DLink file created from the previous step. Note that this new file will look different for you if your installation directory and path to the 4DLink file are different.

```
"C:\Documents and Settings\Administrator\Desktop\Windows\4D\4D.exe"
"C:\Documents and
Settings\Administrator\Desktop\Windows\4D\Autostart.4DLink"
change path to point to exe and 4d.link
```

8. Save it as "Autostart.bat" on the desktop or in the C:\ drive.

9. Double click "Autostart.bat" to test if everything works fine. This should launch 4D and have it try to connect to a server but fail.

10. The last step is to setup the computer to automatically launch the batch file.

    a. Start/All Programs/Accessories/System Tools/Scheduled Tasks
    b. Click "Add Scheduled Task".
    c. Click "Browse…" and select the created batch (.bat) file.
    d. Click OK
    e. Select "When my computer starts", enter the Administrator password
    f. Click Finish
    g. Right click the task, Use "Run" to test the created task, just to be sure everything works (as in step 9 of this section).

While these 10 steps looks complicated, remember, you need to do them only once. Even if you want to bench several databases, you always can use the same client image, just change the Dyndns host IP address to try with another application.

Step 7: Save/Bundle the instance to create an image

We now again follow the Getting Starting Guide, with "Tutorial#2: Bundling an instance into an AMI" with "Step 2: Bundle the Image."

In the Instances tab Elasticfox, right click your still-running instance and select "Bundle into an AMI". Enter an S3 Bucket Name and image name. The bucket name has to be unique and lowercase characters only. You may use a "_" sign. To be sure to have unique names, it is suggested to use a combination of your company/domain name and your name or a test, such as "com4d_test1". If the chosen name is already in use, this step fails (without a clear error text). Avoid diacritical characters outside a-z.

If you already own software supporting Amazon S3, you may want to use that first to create a "bucket", to avoid this issue. Several FTP clients do support S3, such as Interarchy or Cyberduck.

Now click OK to start the process. This process needs several minutes, depending on the hard disk content and fragmentation. As we only copied some files on the disk, it is not necessary to be concerned so much about "zeroing" as described in the Getting Starting Guide, but this may be an issue with the server instance.

The process of bundling appears in the "Bundle Tasks" tab. In a few minutes, the progress should be displayed as "completed". When this happens, right click and choose "Register a new AMI". Return to the Images tab and run a search for your AMI using the S3 Bucket Name that you assigned.

Done – you now have your own image, ready for use. You can launch multiple instances of this image with a single click 1, 5, 20, 100 instances of this image that automatically logs into Windows, launches 4D and attempts to connect to your server.

While all of that works automatically, of course you still can use MS Remote Desktop to connect manually to each instance and show the customer what is happening. Using several Remote Desktop windows you can show several clients at once!

Prepare an Amazon EC2 instance to be used as Server

This step is very similar to the previous one, except we are going to install 4D Server. Follow the steps in the previous section up until installation of the client (step 5.2). If you want to run the server on a high speed machine, select the 64 bit image to support more than 3 GB of RAM for the computer, allowing 4D Server v11 SQL to use up to 4 GB of RAM though this is not necessary.

Install 4D Server. As with the client, you can install it from 4D's web site or copy the folder via your own FTP site, which is drastically faster than using Remote Desktop copy's feature.

Launch the server and activate it, assign Client licenses depending on the number of clients you want to bench. For a one time test you can use the evaluation license (valid for 30 days).

While you could install your application, including a test datafile, on the image, it may make sense to use an "external" hard disk for this, as the hard disk space of volume "C" is very limited. Also remember that every change on the internal hard drives from the virtual machines are lost (they are not saved) when you shut down Windows.

A solution is to create an "EBS Volume". See the Elasticfox Getting Started Guide, Tutorial #3: Creating an Elastic Block Store (EBS) Volume. An EBS Voume allows us the use of online storage that persists without having to rebundle an instance after changes are made to it. It also costs a few cents per GB. Attach it to your running 4D Server instance and format it. Note that when you attach it to an instance you see an additional disk on the machine, typically with hard drive letter "F:\". Copy your application (4D structure, test datafile, plugins, resources, extra, etc) on that disk.

Note: it is usually faster if you zip all files on your local machine, upload that and then expand it. Windows 2003 Server can expand .zip files without additional software.

When the EBS Volume is fully prepared, take a snapshot (as described in the Guide Tutorial 3, Step 4). This allows you do store all that data on Amazon's network and very quickly access that.

Last, make a bundle of the server instance. This is done exactly the same way as we created the client instance from the previous section.

Run the user test

Everything is prepared now – we can launch a test anytime now with a few clicks!

1. Launch the server AMI. Locate your AMI in the Instances tab, right click it and launch it. Make sure you specify in which zone to launch and to assign the correct security group. For the first test take a cheap machine (32 bit or 64 bit, depending of the image)
2. Wait until the instance is fully launched (remember the console says Windows is ready)
3. Login via Remote Desktop, using your modified Administrator password
4. If used, assign the EBS Volume with your own data by right clicking the instance and selecting "Attach an EBS volume"
5. As Disk C:\ is to small, copy your application from Drive C or from the EBS Volume to drive D:\ which is completely empty at this point; you can use all disk storage on that drive.
6. Launch 4D Server!
7. Use the monitor window to note the local IP address used from the server.
8. Connect to your DynDNS account (or whatever service you used) and modify the IP address for the used hostname to match the local address of the server
9. Launch the client AMI. Locate your AMI in the Instances tab. For the first test, launch 1 to verify it works, but later – as many as you wish! You can request more than 20 instances by filling out a form at http://aws.amazon.com/contact-us/ec2-request/.
10. On the Server instance, you will see clients appear in the benchmark control window as soon they have fully booted and 4D is launched and connected to the server.
11. If the clients do not appear but their instances are running, log into a client using Remote Desktop to see if the scheduled task is properly running the batch file, which in turn is properly launching 4D and applying the 4DLink file. Correct the problem and create a new "bundle" and retry.
12. As soon as all clients have appeared in the Control Window, run your tests as you would have on a local network!

13. When you are happy, you may abort the clients from the Control Window (in our example database by calling "Bench_StopAllJobs"). This shows some statistics in the Control Window.
14. Terminate all Client instances using ElasticFox. No need to shut them down, simply select all in ElasticFox, right click "Terminate". As soon they are terminated, billing for CPU time ends.
15. If the last status from the test application or data file is needed in the future, you can copy them to an EBS Volume – don't forget to make a Snapshot to save that!
16. Terminate the Server instance.

Cleaning up

If you followed these steps, you will have two or more saved images, usage of S3 buckets and optionally an EBS Volume, all producing monthly costs. Take a look at your Amazon AWS account info to see your current bill. The images created for this tech note produces costs of about $0.50/month, so there was no real need to hurry. But be sure that you terminate all running instances, as they are charged per hour!

If you do not need the EBS Volume anymore, simply select it, right click and choose "Delete this volume".

If you do not need a created image anymore, right click it and choose "Deregister this AMI"

Note: even after deregistering an AIM, it still uses the storage space in your S3 bucket and Amazon will continue to charge you a monthly fee. Accessing an S3 bucket using the Amazon tools is not very user friendly, but there are several tools (shareware, etc) and even web pages to make that easy. If you continue to use Amazon S3, it is strongly recommended to install one of the available tools, such as Cyberduck (http://cyberduck.ch/) on Mac. For Windows, do an internet search for "windows ftp s3" to find several tools.

Remember that you need to use your account name and the account keys (as in Setting up Elasticfox) to connect to your S3 bucket. Delete unused files in the bucket.

If you do not want to install software for that, there are several web sites offering this service and you need to upload your private AWS key for that. It is recommended to change the AWS key directly after using these sites to empty the bucket (or delete the files).

If you decide not to use Amazon EC2 anymore at all, you may also disable your account (remember, you need to disable all three accounts for EC2, S3 and AWS) to be sure that billing ends.

Conclusion

This technical note described how to setup an Amazon EC2 image to be used for testing a 4D client-server environment. It also described how to enhance existing applications to run automated user emulation.

Amazon EC2 allows control over many computers with a few clicks (as soon they are prepared) at a very low rate of cost. Amazon virtual machines can be rented per hour without any setup fee and is a very interesting option to run benchmarks and stress tests with.