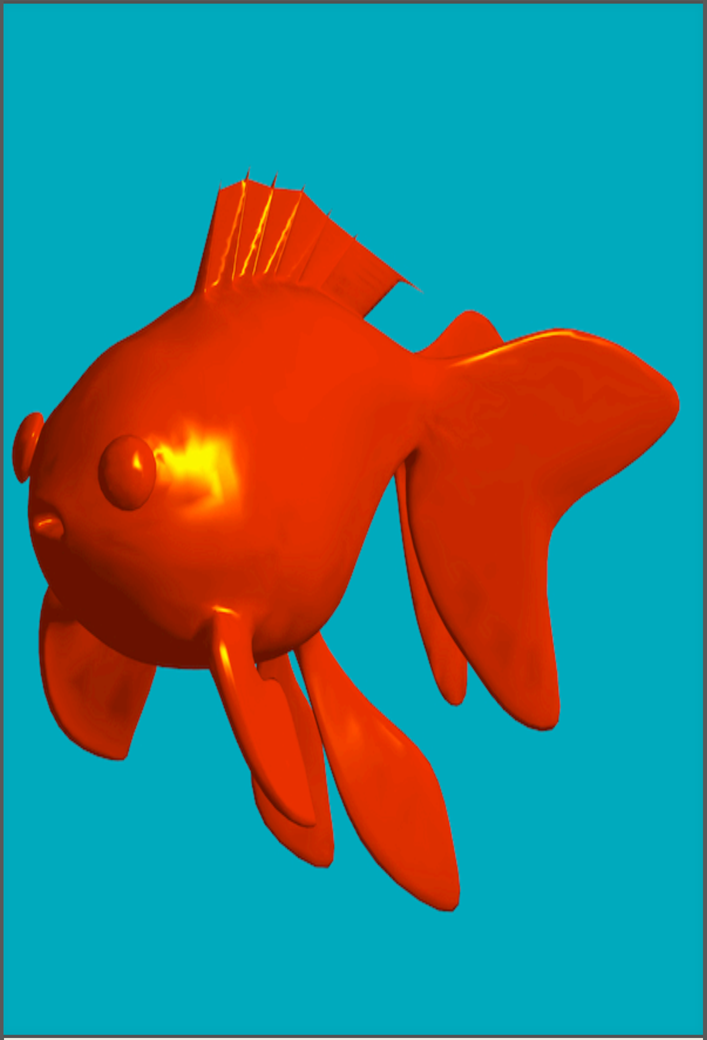
CSCIE-175 Final Project

Project: AWS
Name: Krunal Patel
E-Mail: k5patel@me.com
EC2 AMI (not avail after 1/13/2013): ami-85d35bec
Github: https://github.com/lateralpunk/rgrt
YouTube: http://www.youtube.com/watch?v=9r9xSPlbVDA
Website: http://lateralpunk.blogspot.com/2013/01/cscie-175-final-project-aws-cluster-gpu.html
Website of EC2 Instance (not avail after 1/13/2013): http://ec2-23-20-133-181.compute-1.amazonaws.com/rgrt/
Website RGRT:
http://www.cs264.org/2009/projects/web/Patel_Krunal/index.html



AWS CLUSTER GPU RGRT

Introduction

I've based the last several years of my career focused in on GPU computing. It started off working in CUDA, and now I'm focused in on heterogeneous type computing using OpenCL. Back when I started, one of the major obstacles was obtaining access to compatible computers that can employ GPU computing. Hence I was delighted to learn about Amazon Web Service's offering of their Cluster GPU Instances.

The purpose of this project is to employ AWS Cluster GPU to implement a Regular-Grid Accelerated Ray Tracer. This RGRT will then be hosted on the same GPU instance to provide a web-service for rendering images. To make the project feasible, I decided employ the RGRT CUDA application I had worked on before. This would streamline my application development and also have something practical (and beautiful) to demo. One of the key things this was missing was sort of a service that end-users can employ to try out the ray tracer. I hope to achieve that goal by using AWS, Apache, and PHP.

Regular-Grid Accelerated Ray Tracing

The following explanation is derived from my previous work on this topic (http://www.cs264.org/2009/projects/web/Patel_Krunal/index.html).

Ray-tracing is a method used to draw realistic images on a computer. One can find examples of ray-traced images when viewing Disney/Pixar movies like Toy Story. The underlying algorithm behind ray-tracing is to decompose a picture into pixels, send rays through the pixels, intersect the rays with the objects in the scene, launch any secondary rays, and emit a final color for the original pixel.

Ray-tracing is known to be an embarrassingly parallel problem. Each pixel can be completely calculated independently of any other neighboring pixels. So a mechanism that allows spawning of multiple threads to parallelize the ray-tracing tasks would prove quite useful. Hence we have employed CUDA based technology to exemplify the performance gains to be had with GPU hardware acceleration for ray-tracing.

Our ray-tracer supports spheres, rectangles, and triangles as the basic primitives. Multiple light sources as well as shadows are available. Anti-aliasing via regular sampling is available too. Phong materials are utilized with support for glossy specular highlights. All world objects are composed into build scenes. Support for 3D model meshes by utilizing the PLY file format is available too.

Finally, though we get substantial improvements in rendering time taking a brute approach to ray-tracing, we also employ an acceleration scheme to the whole process. Obviously the brute approach is to traverse the whole scene-graph per pixel and the ray-tracing steps to find any intersections with world objects. A better option does exist. Our regular grid acceleration scheme subdivides the world into axis-aligned cells. These cells contain the world primitive objects that are within it's bounding box. The grid creation is done on the CPU. GPU grid traversal involve shooting a ray and intersecting with the cells and primitives that come in it's path. Once such an intersection is found, it is no longer necessary to continue with finding another intersection as we are guaranteed that the one just found is the closest. Thereafter shading proceeds as normal. More information about regular grid acceleration scheme can be found in the book "Ray Tracing from the Ground Up".

For the purposes of this project, we will treat the CUDA implementation of the RGRT as a black-box. We will assume it just works, and we are more interested in the plumbing involved in getting this to work on AWS as a web-service.

AWS Cluster GPU Setup

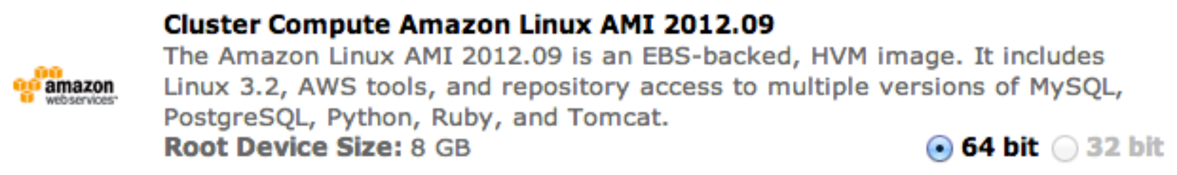
We wish to employ AWS to use the GPU horsepower. As a note, I would use Chrome browser for testing this project. To this extent, we list steps below to show our course of action. We will then go into each step in more detail:

1) Launch a Cluster GPU Instance that is not a spot instance
2) Setup the GPU instance with the right set of CUDA software packages
3) Setup the GPU instance with the right set of web hosting software packages
4) Get the source code for the RGRT off Github, build it and test it
5) Ensure that the PHP code implementing the web-service is up & running
6) Create an AMI of the above for safe-keeping (and so others can get at it quicker)
7) Publicly make the AMI available
8) Destroy the instance from 1)
9) Launch a Cluster GPU Spot Instance using the newly created AMI

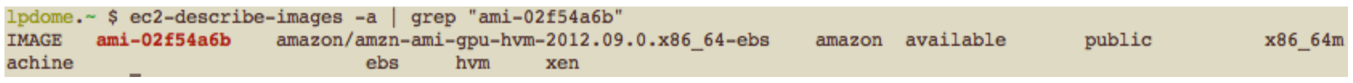
So let the fun begin!!

First we need to launch the Cluster GPU instance that we will use as our setup machine to create the AMI. We decide not to employ a spot instance here because we don't want to have our instance taken from under our noses unexpectedly.

From our investigations, we know we want to employ the AMI: ami-02f54a6b:

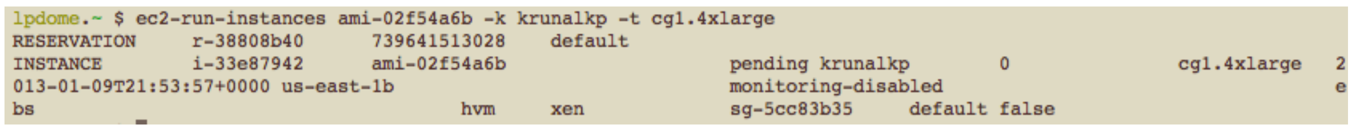


Cluster Compute Amazon Linux AMI 2012.09
The Amazon Linux AMI 2012.09 is an EBS-backed, HVM image. It includes Linux 3.2, AWS tools, and repository access to multiple versions of MySQL, PostgreSQL, Python, Ruby, and Tomcat.
Root Device Size: 8 GB [✓ 64 bit] 32 bit



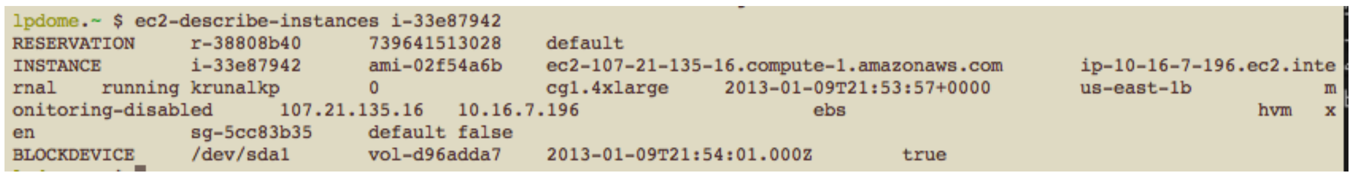
```
lpdome.~ $ ec2-describe-images -a | grep "ami-02f54a6b"
IMAGE   ami-02f54a6b    amazon/amzn-ami-gpu-hvm-2012.09.0.x86_64-ebs    amazon  available       public          x86_64m
achine                          ebs     hvm     xen
```

We then launch the instance from CLI:

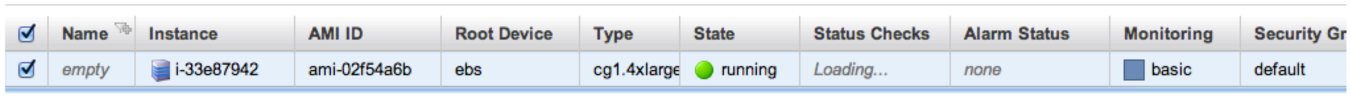


```
lpdome.~ $ ec2-run-instances ami-02f54a6b -k krunalkp -t cg1.4xlarge
RESERVATION     r-38808b40      739641513028    default
INSTANCE        i-33e87942      ami-02f54a6b                    pending krunalkp        0               cg1.4xlarge     2
013-01-09T21:53:57+0000 us-east-1b                              monitoring-disabled                                     e
bs                              hvm     xen                     sg-5cc83b35     default false
```

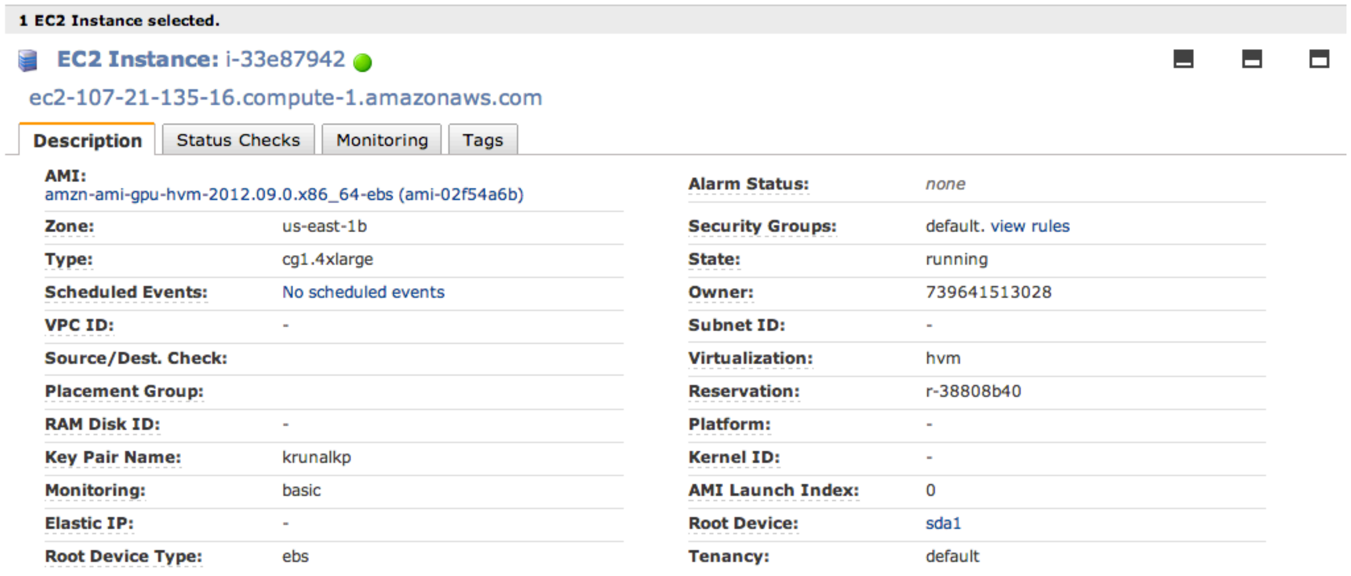
Notice we need to explicitly set the Instance Type (-t cg1.4xlarge).



```
lpdome.~ $ ec2-describe-instances i-33e87942
RESERVATION     r-38808b40      739641513028    default
INSTANCE        i-33e87942      ami-02f54a6b    ec2-107-21-135-16.compute-1.amazonaws.com       ip-10-16-7-196.ec2.inte
rnal    running krunalkp        0               cg1.4xlarge     2013-01-09T21:53:57+0000        us-east-1b              m
onitoring-disabled      107.21.135.16   10.16.7.196                     ebs                                     hvm     x
en              sg-5cc83b35     default false
BLOCKDEVICE     /dev/sda1       vol-d96adda7    2013-01-09T21:54:01.000Z        true
```



✓	Name	Instance	AMI ID	Root Device	Type	State	Status Checks	Alarm Status	Monitoring	Security Gr
✓	*empty*	i-33e87942	ami-02f54a6b	ebs	cg1.4xlarge	● running	*Loading...*	*none*	basic	default



1 EC2 Instance selected.

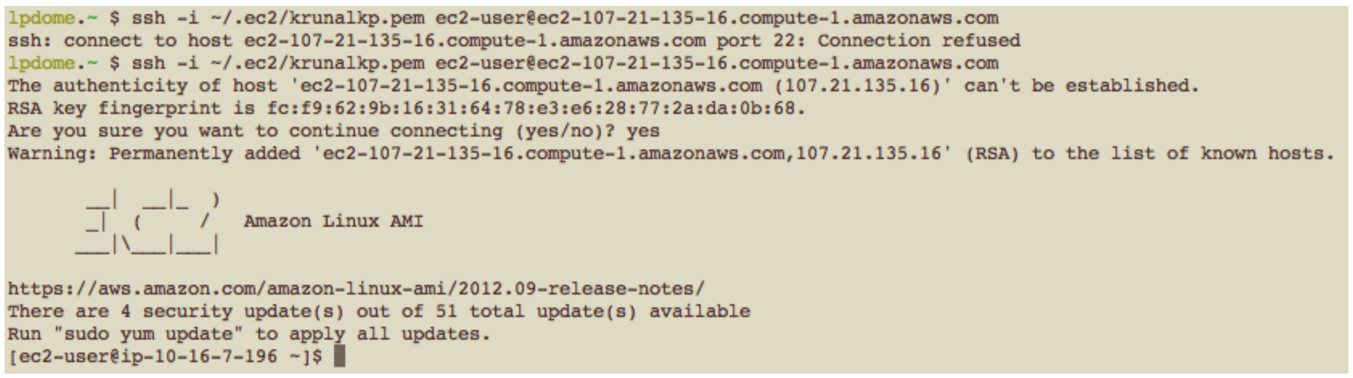
EC2 Instance: i-33e87942 ●

ec2-107-21-135-16.compute-1.amazonaws.com

Description | Status Checks | Monitoring | Tags

AMI:	amzn-ami-gpu-hvm-2012.09.0.x86_64-ebs (ami-02f54a6b)	**Alarm Status:**	*none*
Zone:	us-east-1b	**Security Groups:**	default. view rules
Type:	cg1.4xlarge	**State:**	running
Scheduled Events:	No scheduled events	**Owner:**	739641513028
VPC ID:	-	**Subnet ID:**	-
Source/Dest. Check:		**Virtualization:**	hvm
Placement Group:		**Reservation:**	r-38808b40
RAM Disk ID:	-	**Platform:**	-
Key Pair Name:	krunalkp	**Kernel ID:**	-
Monitoring:	basic	**AMI Launch Index:**	0
Elastic IP:	-	**Root Device:**	sda1
Root Device Type:	ebs	**Tenancy:**	default

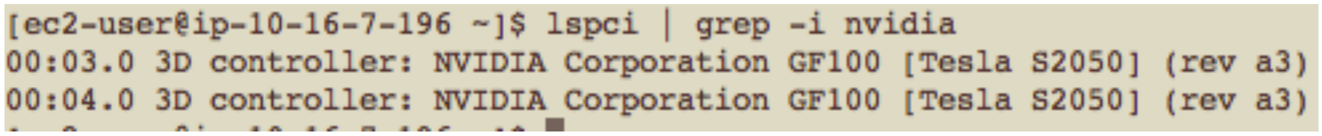
Now we ssh into the machine:



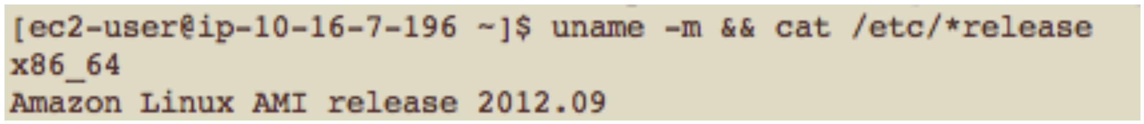
```
lpdome.~ $ ssh -i ~/.ec2/krunalkp.pem ec2-user@ec2-107-21-135-16.compute-1.amazonaws.com
ssh: connect to host ec2-107-21-135-16.compute-1.amazonaws.com port 22: Connection refused
lpdome.~ $ ssh -i ~/.ec2/krunalkp.pem ec2-user@ec2-107-21-135-16.compute-1.amazonaws.com
The authenticity of host 'ec2-107-21-135-16.compute-1.amazonaws.com (107.21.135.16)' can't be established.
RSA key fingerprint is fc:f9:62:9b:16:31:64:78:e3:e6:28:77:2a:da:0b:68.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-107-21-135-16.compute-1.amazonaws.com,107.21.135.16' (RSA) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2012.09-release-notes/
There are 4 security update(s) out of 51 total update(s) available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-16-7-196 ~]$ 
```

We do some updates first (sudo yum update) and reboot the machine. We install the following applications: emacs, git, freeglut-devel (for CUDA samples). I don't show the steps here for simplicity. We reboot (sudo reboot) after all the yum installs.

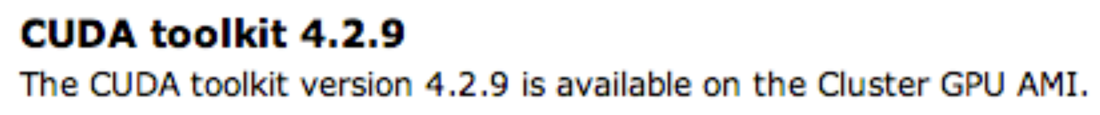
Next, by default the AMI is using CUDA 4.2. I wish to upgrade to the latest CUDA 5 since my code uses some utility functions only available in the later version of CUDA. But as a sanity check, let's ensure that we actually have CUDA compatible GPUs:


```
[ec2-user@ip-10-16-7-196 ~]$ lspci | grep -i nvidia
00:03.0 3D controller: NVIDIA Corporation GF100 [Tesla S2050] (rev a3)
00:04.0 3D controller: NVIDIA Corporation GF100 [Tesla S2050] (rev a3)
```

So as promised by AWS, we do have CUDA capable devices (2 for that matter). Next we verify that we have a compatible Linux distro (which we should ofcourse):


```
[ec2-user@ip-10-16-7-196 ~]$ uname -m && cat /etc/*release
x86_64
Amazon Linux AMI release 2012.09
```

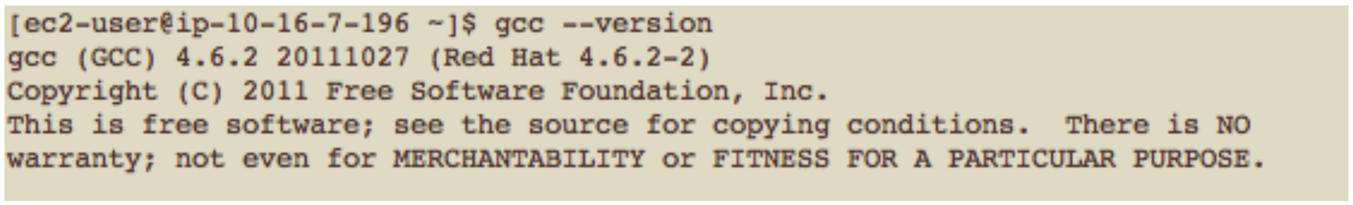
Ok, well this doesn't tell us much unfortunately. Doing a simple Google search gives us http://aws.amazon.com/amazon-linux-ami/2012.09-release-notes/


CUDA toolkit 4.2.9

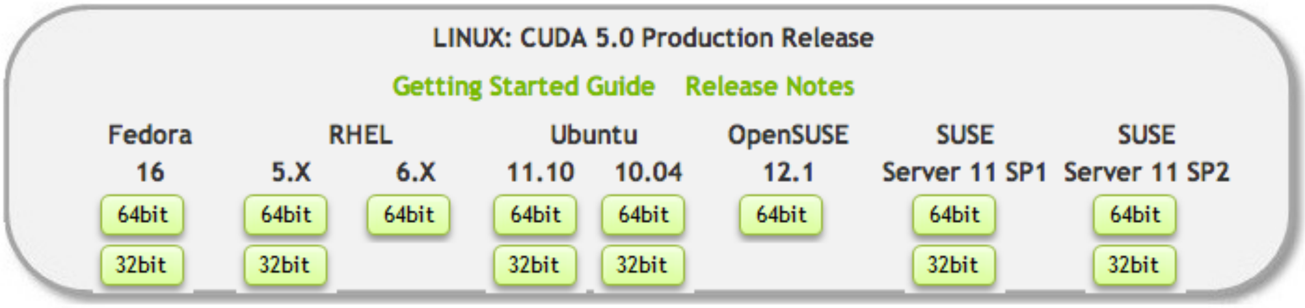
The CUDA toolkit version 4.2.9 is available on the Cluster GPU AMI.

Ok well this proves that we don't have the right toolkit (and we will update), but still doesn't tell us the distro. This link http://ivan.manida.com/2012/11/installing-ffmpeg-on-amazon-ami-centos-6.html gives us the information we want: the AMI is a rebranded CentOS.

Ok then I need to verify gcc installed on the system:


```
[ec2-user@ip-10-16-7-196 ~]$ gcc --version
gcc (GCC) 4.6.2 20111027 (Red Hat 4.6.2-2)
Copyright (C) 2011 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

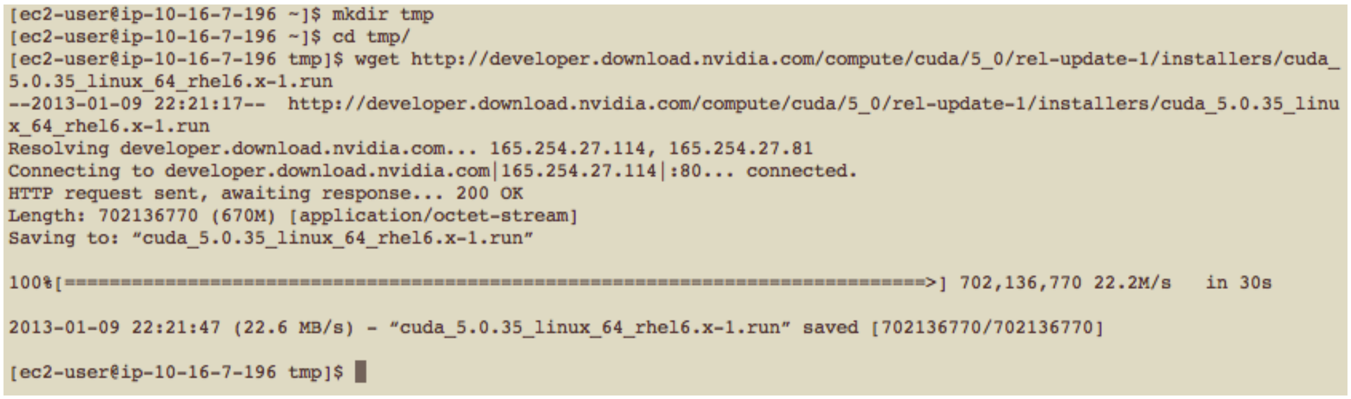
Oh nice, here we find out that the distro is Red Hat 4.6.2-2. We do have a good enough version of GCC as well. Finally, we go to the Nvidia CUDA page and download version 5:


LINUX: CUDA 5.0 Production Release

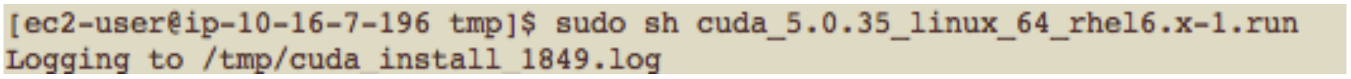
Getting Started Guide   Release Notes

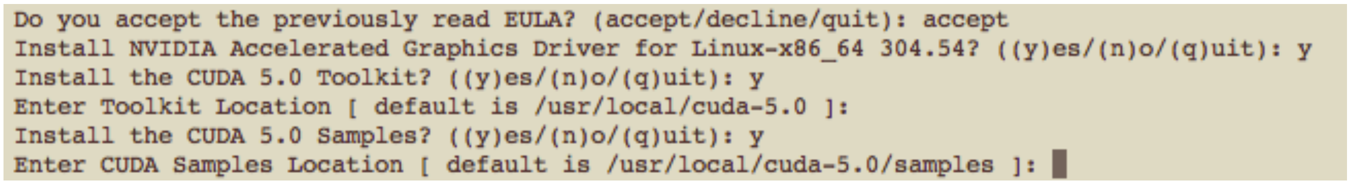
Fedora	RHEL		Ubuntu		OpenSUSE	SUSE	SUSE
16	5.X	6.X	11.10	10.04	12.1	Server 11 SP1	Server 11 SP2
64bit	64bit	64bit	64bit	64bit	64bit	64bit	64bit
32bit	32bit		32bit	32bit		32bit	32bit

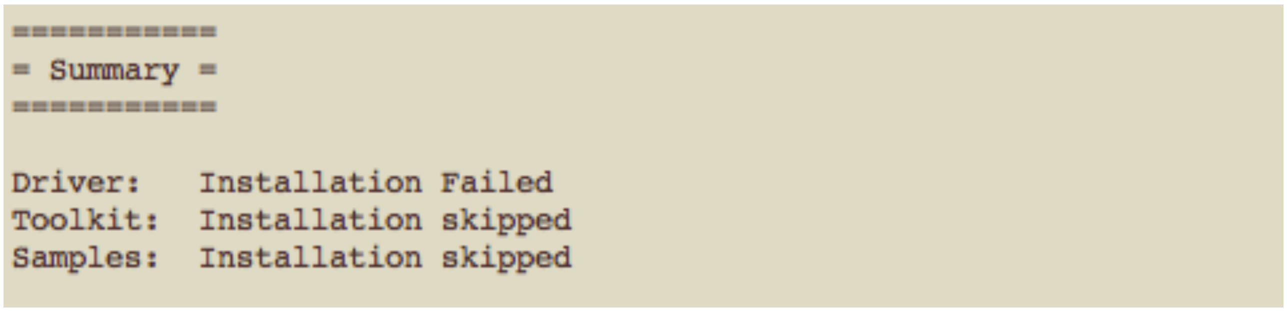
Namely, we download RHEL 6.x 64-bit: http://developer.download.nvidia.com/compute/cuda/5_0/rel-update-1/installers/cuda_5.0.35_linux_64_rhel6.x-1.run


```
[ec2-user@ip-10-16-7-196 ~]$ mkdir tmp
[ec2-user@ip-10-16-7-196 ~]$ cd tmp/
[ec2-user@ip-10-16-7-196 tmp]$ wget http://developer.download.nvidia.com/compute/cuda/5_0/rel-update-1/installers/cuda_
5.0.35_linux_64_rhel6.x-1.run
--2013-01-09 22:21:17--  http://developer.download.nvidia.com/compute/cuda/5_0/rel-update-1/installers/cuda_5.0.35_linu
x_64_rhel6.x-1.run
Resolving developer.download.nvidia.com... 165.254.27.114, 165.254.27.81
Connecting to developer.download.nvidia.com|165.254.27.114|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 702136770 (670M) [application/octet-stream]
Saving to: "cuda_5.0.35_linux_64_rhel6.x-1.run"

100%[==========================================================================>] 702,136,770 22.2M/s   in 30s

2013-01-09 22:21:47 (22.6 MB/s) - "cuda_5.0.35_linux_64_rhel6.x-1.run" saved [702136770/702136770]

[ec2-user@ip-10-16-7-196 tmp]$ █
```

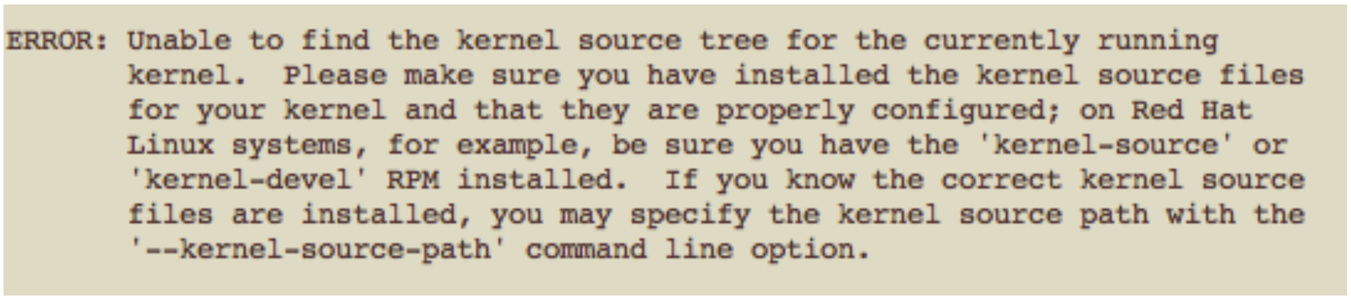
And then we install it:


```
[ec2-user@ip-10-16-7-196 tmp]$ sudo sh cuda_5.0.35_linux_64_rhel6.x-1.run
Logging to /tmp/cuda_install_1849.log
```


```
Do you accept the previously read EULA? (accept/decline/quit): accept
Install NVIDIA Accelerated Graphics Driver for Linux-x86_64 304.54? ((y)es/(n)o/(q)uit): y
Install the CUDA 5.0 Toolkit? ((y)es/(n)o/(q)uit): y
Enter Toolkit Location [ default is /usr/local/cuda-5.0 ]:
Install the CUDA 5.0 Samples? ((y)es/(n)o/(q)uit): y
Enter CUDA Samples Location [ default is /usr/local/cuda-5.0/samples ]: █
```

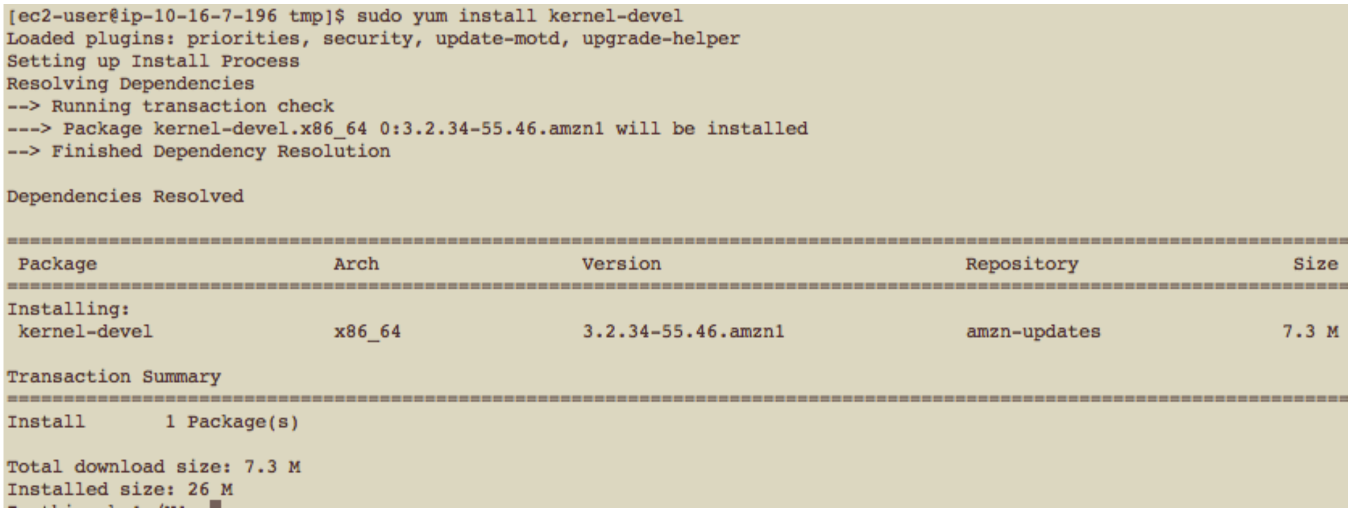

```
===========
= Summary =
===========

Driver:   Installation Failed
Toolkit:  Installation skipped
Samples:  Installation skipped
```

Problem, there was a failure in installing the drivers. Let's check the error log:



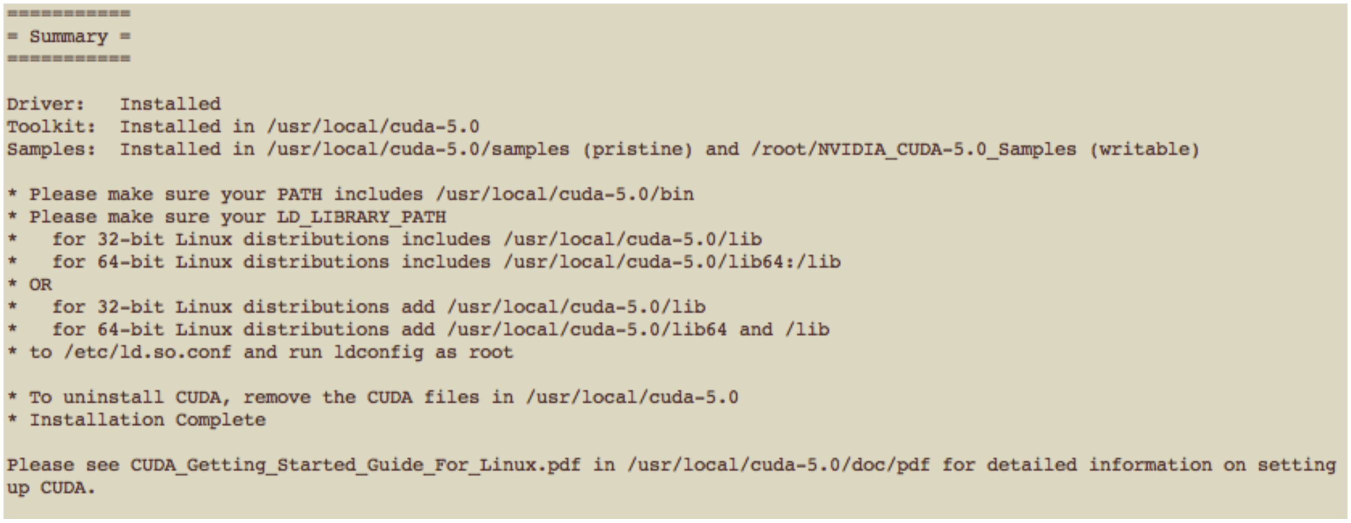
```
ERROR: Unable to find the kernel source tree for the currently running
       kernel.  Please make sure you have installed the kernel source files
       for your kernel and that they are properly configured; on Red Hat
       Linux systems, for example, be sure you have the 'kernel-source' or
       'kernel-devel' RPM installed.  If you know the correct kernel source
       files are installed, you may specify the kernel source path with the
       '--kernel-source-path' command line option.
```

Ok, so let's yum install kernel-devel:



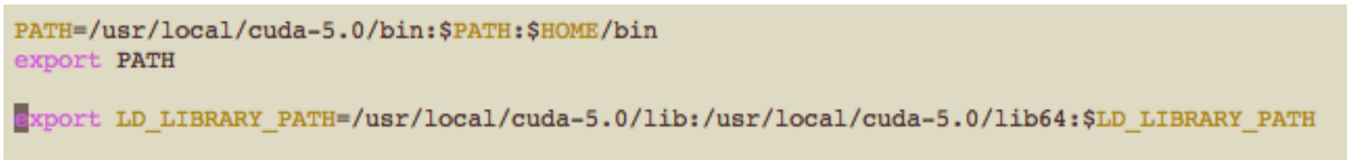
```
[ec2-user@ip-10-16-7-196 tmp]$ sudo yum install kernel-devel
Loaded plugins: priorities, security, update-motd, upgrade-helper
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package kernel-devel.x86_64 0:3.2.34-55.46.amzn1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

========================================================================================================================
 Package                    Arch                 Version                              Repository                  Size
========================================================================================================================
Installing:
 kernel-devel               x86_64               3.2.34-55.46.amzn1                   amzn-updates               7.3 M

Transaction Summary
========================================================================================================================
Install       1 Package(s)

Total download size: 7.3 M
Installed size: 26 M
```

And now we retry CUDA 5 installer:



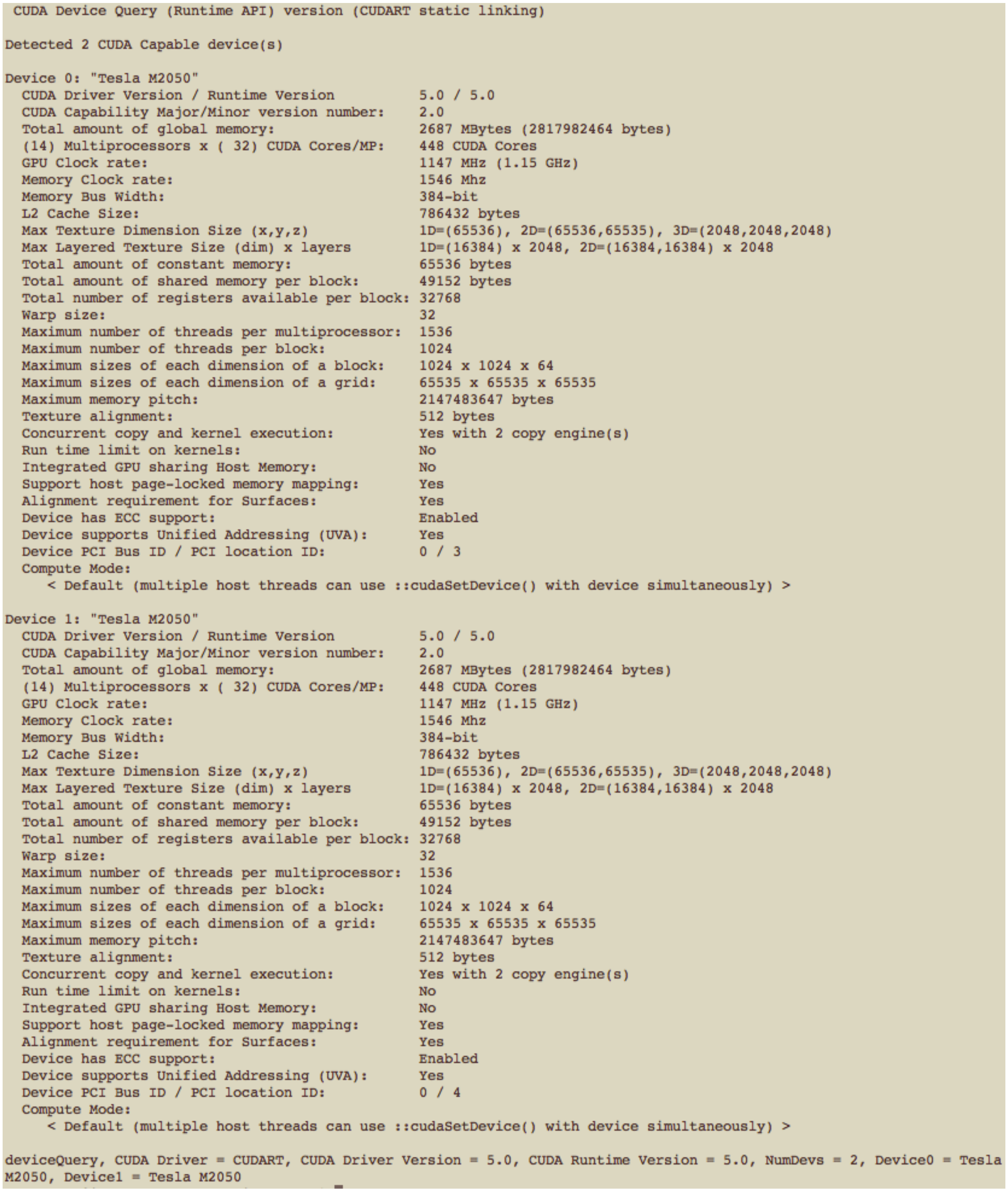
```
===========
= Summary =
===========

Driver:   Installed
Toolkit:  Installed in /usr/local/cuda-5.0
Samples:  Installed in /usr/local/cuda-5.0/samples (pristine) and /root/NVIDIA_CUDA-5.0_Samples (writable)

* Please make sure your PATH includes /usr/local/cuda-5.0/bin
* Please make sure your LD_LIBRARY_PATH
*   for 32-bit Linux distributions includes /usr/local/cuda-5.0/lib
*   for 64-bit Linux distributions includes /usr/local/cuda-5.0/lib64:/lib
* OR
*   for 32-bit Linux distributions add /usr/local/cuda-5.0/lib
*   for 64-bit Linux distributions add /usr/local/cuda-5.0/lib64 and /lib
* to /etc/ld.so.conf and run ldconfig as root

* To uninstall CUDA, remove the CUDA files in /usr/local/cuda-5.0
* Installation Complete

Please see CUDA_Getting_Started_Guide_For_Linux.pdf in /usr/local/cuda-5.0/doc/pdf for detailed information on setting
up CUDA.
```

Good, we got it. Let’s update our .bash_profile file and then we reboot.



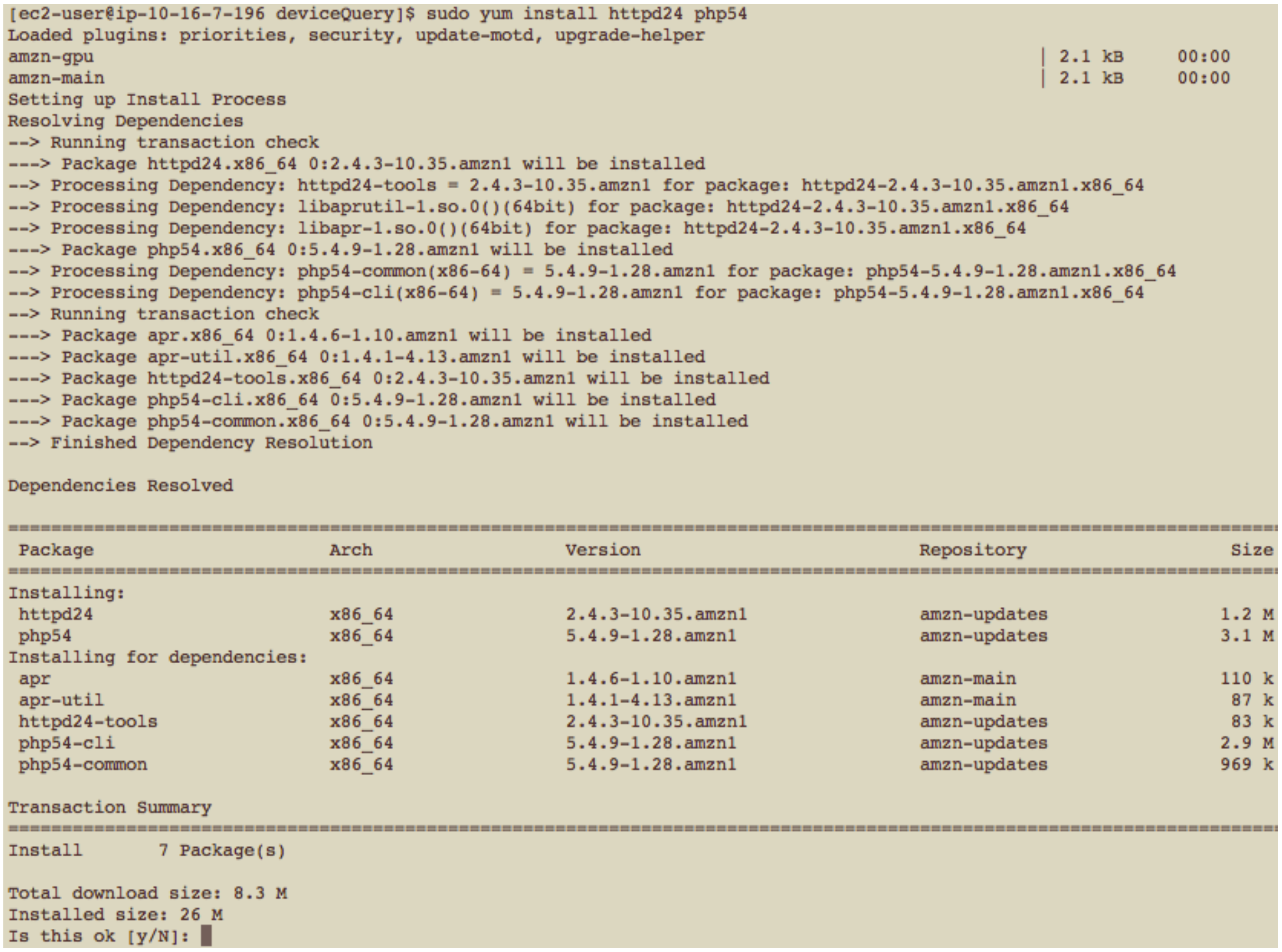
```
PATH=/usr/local/cuda-5.0/bin:$PATH:$HOME/bin
export PATH

export LD_LIBRARY_PATH=/usr/local/cuda-5.0/lib:/usr/local/cuda-5.0/lib64:$LD_LIBRARY_PATH
```

After reboot, we verify that CUDA is working by using the deviceQuery application (after running sudo make in the samples directory):



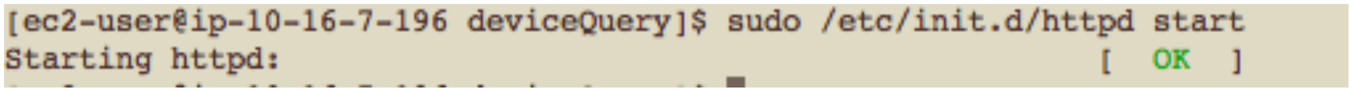
```
 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 2 CUDA Capable device(s)

Device 0: "Tesla M2050"
  CUDA Driver Version / Runtime Version          5.0 / 5.0
  CUDA Capability Major/Minor version number:    2.0
  Total amount of global memory:                 2687 MBytes (2817982464 bytes)
  (14) Multiprocessors x ( 32) CUDA Cores/MP:    448 CUDA Cores
  GPU Clock rate:                                1147 MHz (1.15 GHz)
  Memory Clock rate:                             1546 Mhz
  Memory Bus Width:                              384-bit
  L2 Cache Size:                                 786432 bytes
  Max Texture Dimension Size (x,y,z)             1D=(65536), 2D=(65536,65535), 3D=(2048,2048,2048)
  Max Layered Texture Size (dim) x layers        1D=(16384) x 2048, 2D=(16384,16384) x 2048
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 32768
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  1536
  Maximum number of threads per block:           1024
  Maximum sizes of each dimension of a block:    1024 x 1024 x 64
  Maximum sizes of each dimension of a grid:     65535 x 65535 x 65535
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 2 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Enabled
  Device supports Unified Addressing (UVA):      Yes
  Device PCI Bus ID / PCI location ID:           0 / 3
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

Device 1: "Tesla M2050"
  CUDA Driver Version / Runtime Version          5.0 / 5.0
  CUDA Capability Major/Minor version number:    2.0
  Total amount of global memory:                 2687 MBytes (2817982464 bytes)
  (14) Multiprocessors x ( 32) CUDA Cores/MP:    448 CUDA Cores
  GPU Clock rate:                                1147 MHz (1.15 GHz)
  Memory Clock rate:                             1546 Mhz
  Memory Bus Width:                              384-bit
  L2 Cache Size:                                 786432 bytes
  Max Texture Dimension Size (x,y,z)             1D=(65536), 2D=(65536,65535), 3D=(2048,2048,2048)
  Max Layered Texture Size (dim) x layers        1D=(16384) x 2048, 2D=(16384,16384) x 2048
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 32768
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  1536
  Maximum number of threads per block:           1024
  Maximum sizes of each dimension of a block:    1024 x 1024 x 64
  Maximum sizes of each dimension of a grid:     65535 x 65535 x 65535
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 2 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Enabled
  Device supports Unified Addressing (UVA):      Yes
  Device PCI Bus ID / PCI location ID:           0 / 4
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 5.0, CUDA Runtime Version = 5.0, NumDevs = 2, Device0 = Tesla
M2050, Device1 = Tesla M2050
```

Good. We are done with the CUDA installation. Now lets go onto installing Apache & PHP:



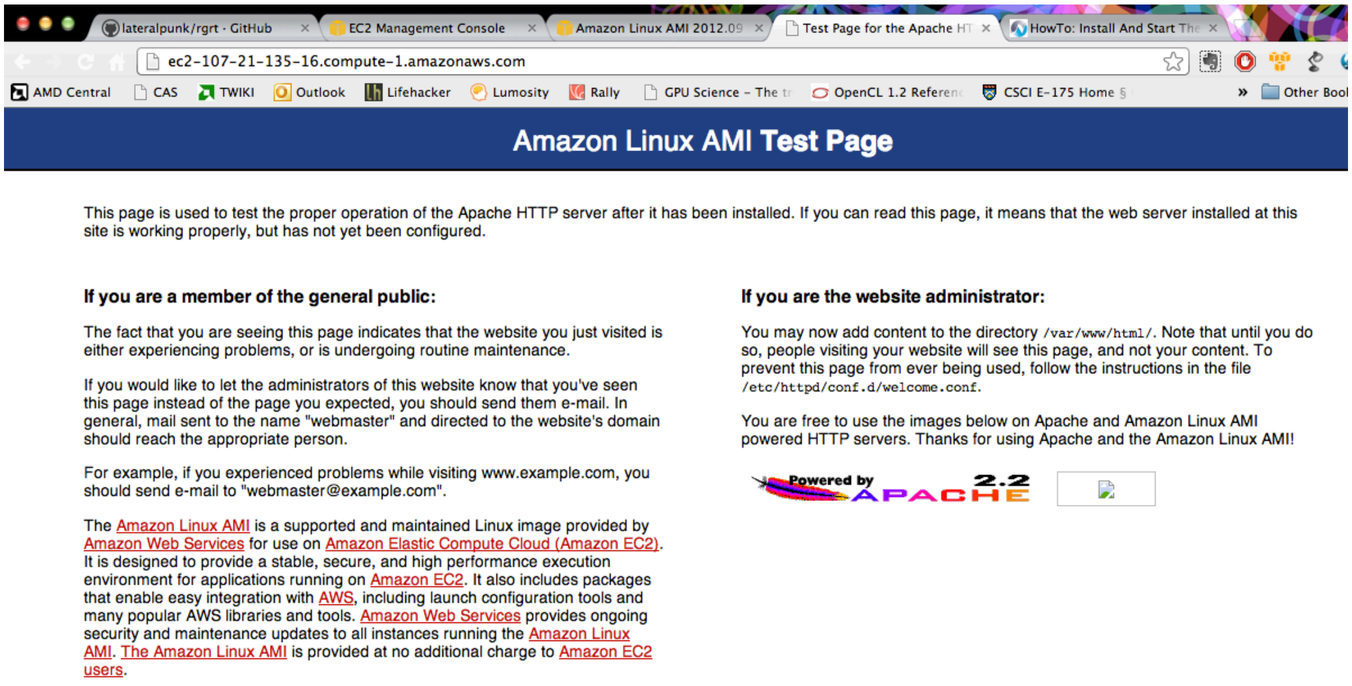
```
[ec2-user@ip-10-16-7-196 deviceQuery]$ sudo yum install httpd24 php54
Loaded plugins: priorities, security, update-motd, upgrade-helper
amzn-gpu                                                                                    | 2.1 kB     00:00
amzn-main                                                                                   | 2.1 kB     00:00
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package httpd24.x86_64 0:2.4.3-10.35.amzn1 will be installed
--> Processing Dependency: httpd24-tools = 2.4.3-10.35.amzn1 for package: httpd24-2.4.3-10.35.amzn1.x86_64
--> Processing Dependency: libaprutil-1.so.0()(64bit) for package: httpd24-2.4.3-10.35.amzn1.x86_64
--> Processing Dependency: libapr-1.so.0()(64bit) for package: httpd24-2.4.3-10.35.amzn1.x86_64
---> Package php54.x86_64 0:5.4.9-1.28.amzn1 will be installed
--> Processing Dependency: php54-common(x86-64) = 5.4.9-1.28.amzn1 for package: php54-5.4.9-1.28.amzn1.x86_64
--> Processing Dependency: php54-cli(x86-64) = 5.4.9-1.28.amzn1 for package: php54-5.4.9-1.28.amzn1.x86_64
--> Running transaction check
---> Package apr.x86_64 0:1.4.6-1.10.amzn1 will be installed
---> Package apr-util.x86_64 0:1.4.1-4.13.amzn1 will be installed
---> Package httpd24-tools.x86_64 0:2.4.3-10.35.amzn1 will be installed
---> Package php54-cli.x86_64 0:5.4.9-1.28.amzn1 will be installed
---> Package php54-common.x86_64 0:5.4.9-1.28.amzn1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====================================================================================================================
 Package                        Arch                   Version                         Repository                Size
=====================================================================================================================
Installing:
 httpd24                        x86_64                 2.4.3-10.35.amzn1               amzn-updates             1.2 M
 php54                          x86_64                 5.4.9-1.28.amzn1                amzn-updates             3.1 M
Installing for dependencies:
 apr                            x86_64                 1.4.6-1.10.amzn1                amzn-main                110 k
 apr-util                       x86_64                 1.4.1-4.13.amzn1                amzn-main                 87 k
 httpd24-tools                  x86_64                 2.4.3-10.35.amzn1               amzn-updates              83 k
 php54-cli                      x86_64                 5.4.9-1.28.amzn1                amzn-updates             2.9 M
 php54-common                   x86_64                 5.4.9-1.28.amzn1                amzn-updates             969 k

Transaction Summary
=====================================================================================================================
Install       7 Package(s)

Total download size: 8.3 M
Installed size: 26 M
Is this ok [y/N]:
```

Then we manually startup the httpd server:



```
[ec2-user@ip-10-16-7-196 deviceQuery]$ sudo /etc/init.d/httpd start
Starting httpd:                                            [  OK  ]
```

And check that Apache is running through our web-browser:



Amazon Linux AMI **Test Page**

This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page, it means that the web server installed at this site is working properly, but has not yet been configured.

If you are a member of the general public:

The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting www.example.com, you should send e-mail to "webmaster@example.com".

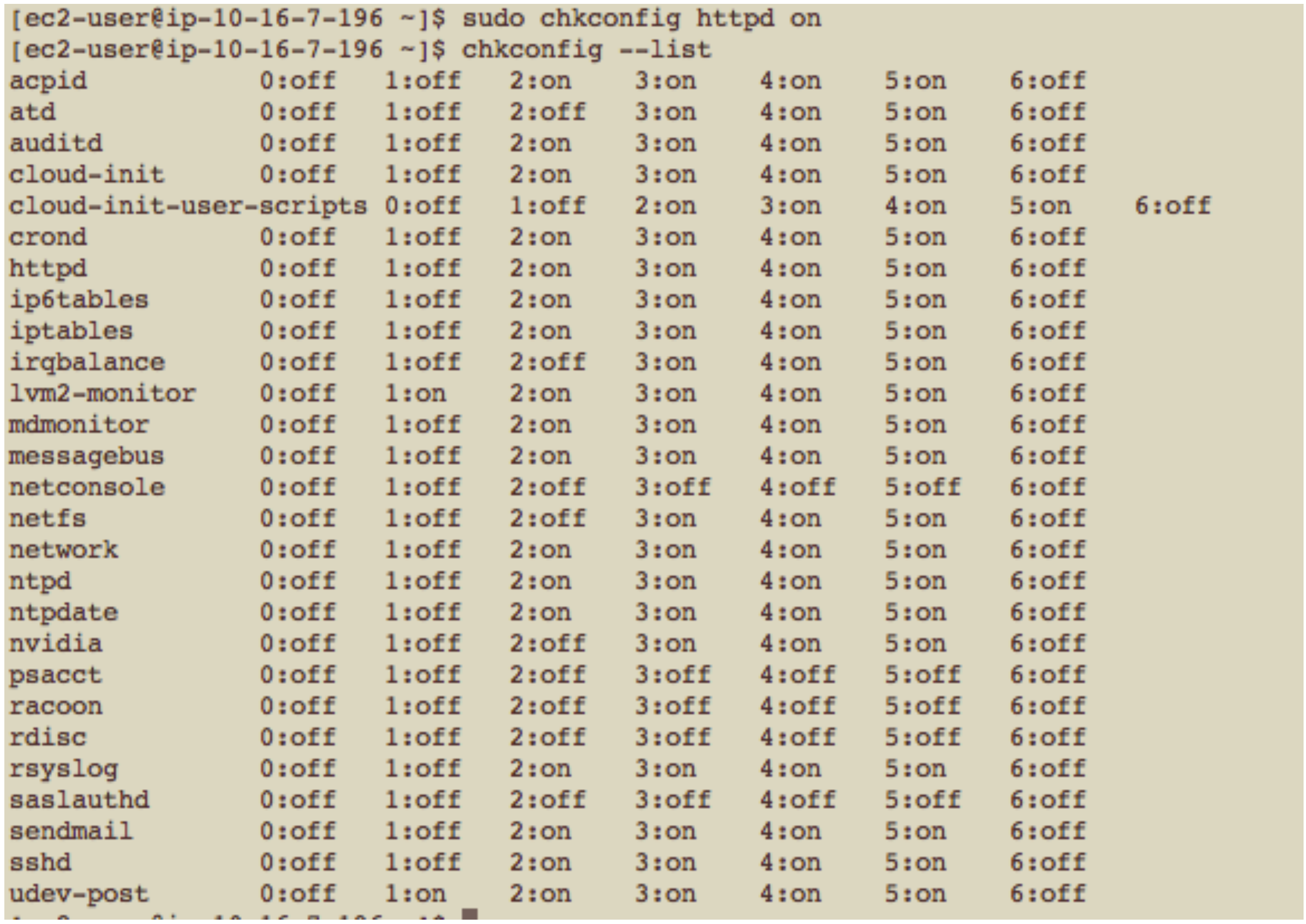
The Amazon Linux AMI is a supported and maintained Linux image provided by Amazon Web Services for use on Amazon Elastic Compute Cloud (Amazon EC2). It is designed to provide a stable, secure, and high performance execution environment for applications running on Amazon EC2. It also includes packages that enable easy integration with AWS, including launch configuration tools and many popular AWS libraries and tools. Amazon Web Services provides ongoing security and maintenance updates to all instances running the Amazon Linux AMI. The Amazon Linux AMI is provided at no additional charge to Amazon EC2 users.

If you are the website administrator:

You may now add content to the directory `/var/www/html/`. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file `/etc/httpd/conf.d/welcome.conf`.

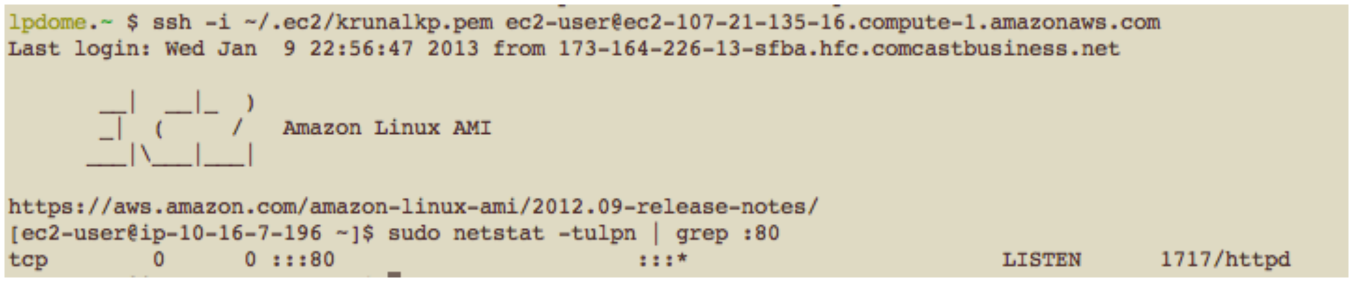
You are free to use the images below on Apache and Amazon Linux AMI powered HTTP servers. Thanks for using Apache and the Amazon Linux AMI!

We want to ensure that the httpd service starts up on boot:



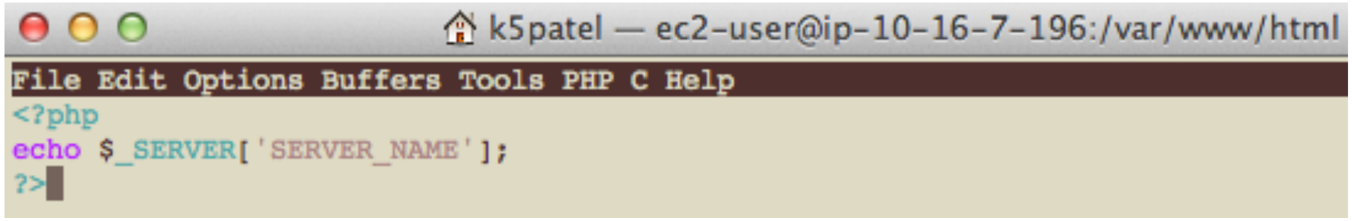
```
[ec2-user@ip-10-16-7-196 ~]$ sudo chkconfig httpd on
[ec2-user@ip-10-16-7-196 ~]$ chkconfig --list
acpid           0:off   1:off   2:on    3:on    4:on    5:on    6:off
atd             0:off   1:off   2:off   3:on    4:on    5:on    6:off
auditd          0:off   1:off   2:on    3:on    4:on    5:on    6:off
cloud-init      0:off   1:off   2:on    3:on    4:on    5:on    6:off
cloud-init-user-scripts 0:off   1:off   2:on    3:on    4:on    5:on    6:off
crond           0:off   1:off   2:on    3:on    4:on    5:on    6:off
httpd           0:off   1:off   2:on    3:on    4:on    5:on    6:off
ip6tables       0:off   1:off   2:on    3:on    4:on    5:on    6:off
iptables        0:off   1:off   2:on    3:on    4:on    5:on    6:off
irqbalance      0:off   1:off   2:off   3:on    4:on    5:on    6:off
lvm2-monitor    0:off   1:on    2:on    3:on    4:on    5:on    6:off
mdmonitor       0:off   1:off   2:on    3:on    4:on    5:on    6:off
messagebus      0:off   1:off   2:on    3:on    4:on    5:on    6:off
netconsole      0:off   1:off   2:off   3:off   4:off   5:off   6:off
netfs           0:off   1:off   2:off   3:on    4:on    5:on    6:off
network         0:off   1:off   2:on    3:on    4:on    5:on    6:off
ntpd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
ntpdate         0:off   1:off   2:on    3:on    4:on    5:on    6:off
nvidia          0:off   1:off   2:off   3:on    4:on    5:on    6:off
psacct          0:off   1:off   2:off   3:off   4:off   5:off   6:off
racoon          0:off   1:off   2:off   3:off   4:off   5:off   6:off
rdisc           0:off   1:off   2:off   3:off   4:off   5:off   6:off
rsyslog         0:off   1:off   2:on    3:on    4:on    5:on    6:off
saslauthd       0:off   1:off   2:off   3:off   4:off   5:off   6:off
sendmail        0:off   1:off   2:on    3:on    4:on    5:on    6:off
sshd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
udev-post       0:off   1:on    2:on    3:on    4:on    5:on    6:off
```

After reboot:



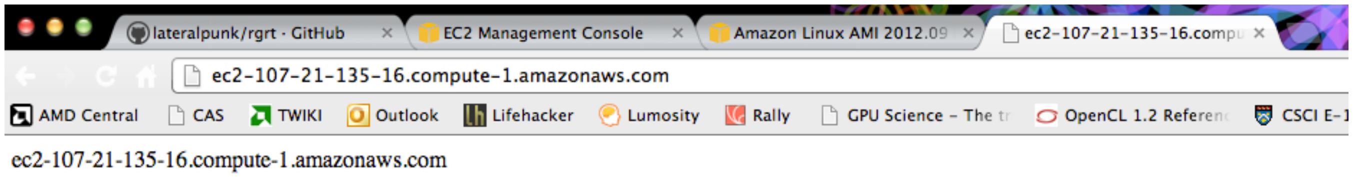
```
lpdome.~ $ ssh -i ~/.ec2/krunalkp.pem ec2-user@ec2-107-21-135-16.compute-1.amazonaws.com
Last login: Wed Jan  9 22:56:47 2013 from 173-164-226-13-sfba.hfc.comcastbusiness.net

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2012.09-release-notes/
[ec2-user@ip-10-16-7-196 ~]$ sudo netstat -tulpn | grep :80
tcp        0      0 :::80                       :::*                        LISTEN      1717/httpd
```

Now, let's remove the error page, and have just a single index.php file that will host our logic:



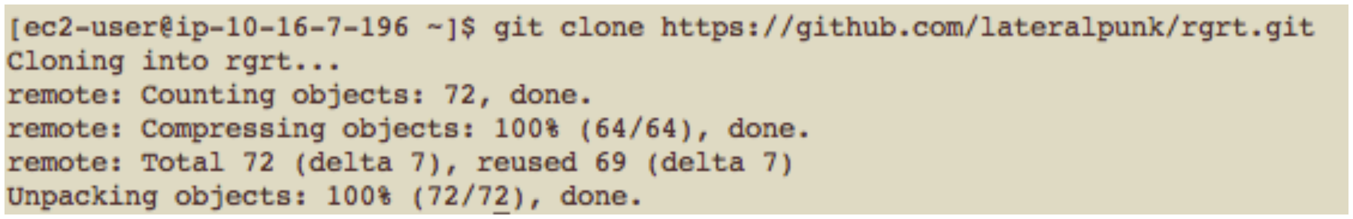
k5patel — ec2-user@ip-10-16-7-196:/var/www/html

```
File Edit Options Buffers Tools PHP C Help
<?php
echo $_SERVER['SERVER_NAME'];
?>
```

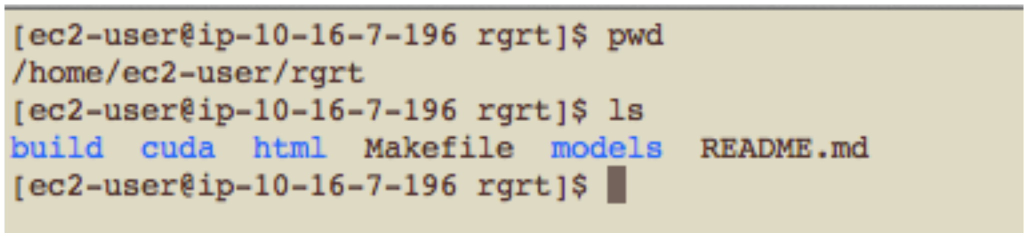


ec2-107-21-135-16.compute-1.amazonaws.com

Ok, now let's go onto getting the source code for the RGRT from github where I posted it publicly. Here we use the https protocol so that we can provide our username & password. If we instead used the SSH protocol that the AMI image that I would eventually make would have my credentials!

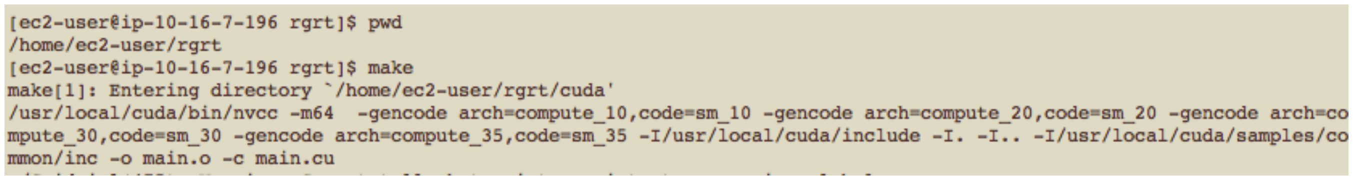


```
[ec2-user@ip-10-16-7-196 ~]$ git clone https://github.com/lateralpunk/rgrt.git
Cloning into rgrt...
remote: Counting objects: 72, done.
remote: Compressing objects: 100% (64/64), done.
remote: Total 72 (delta 7), reused 69 (delta 7)
Unpacking objects: 100% (72/72), done.
```

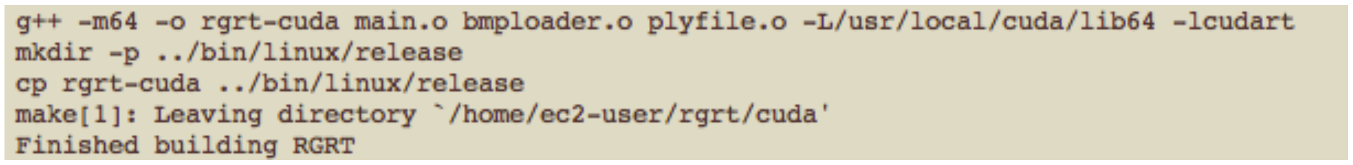


```
[ec2-user@ip-10-16-7-196 rgrt]$ pwd
/home/ec2-user/rgrt
[ec2-user@ip-10-16-7-196 rgrt]$ ls
build  cuda  html  Makefile  models  README.md
[ec2-user@ip-10-16-7-196 rgrt]$ 
```

Let's ensure that we can build our application:

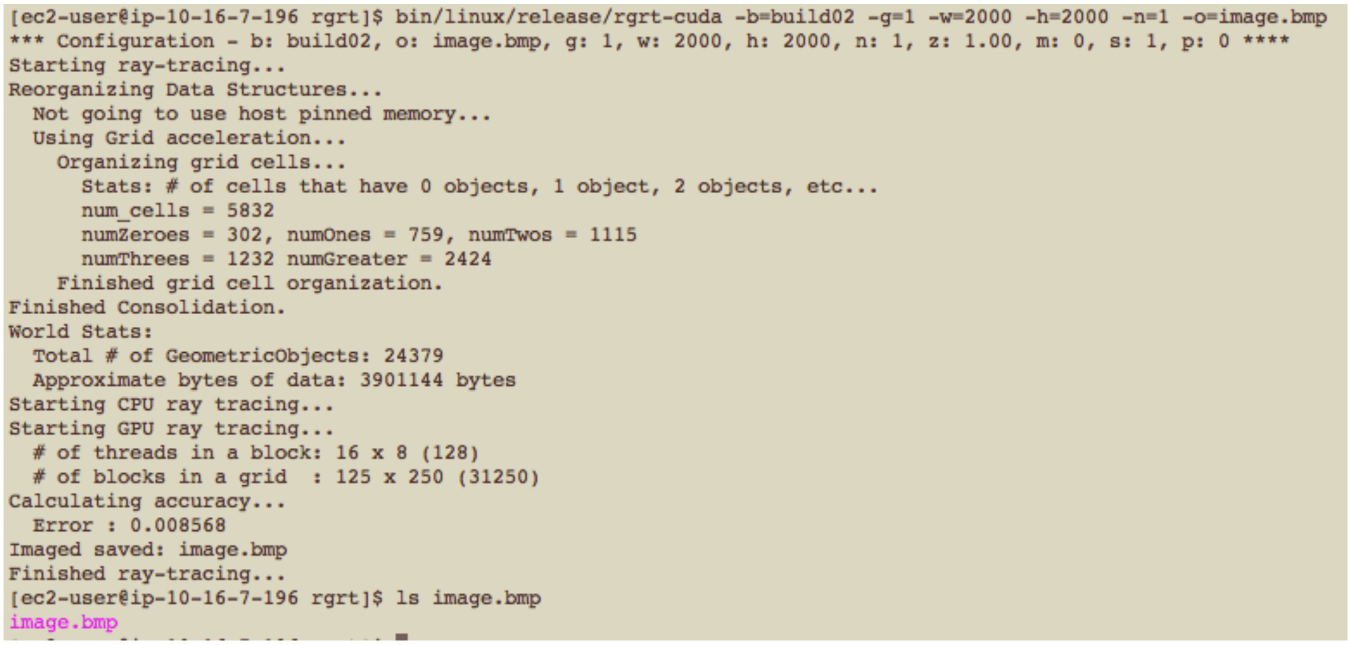


```
[ec2-user@ip-10-16-7-196 rgrt]$ pwd
/home/ec2-user/rgrt
[ec2-user@ip-10-16-7-196 rgrt]$ make
make[1]: Entering directory `/home/ec2-user/rgrt/cuda'
/usr/local/cuda/bin/nvcc -m64  -gencode arch=compute_10,code=sm_10 -gencode arch=compute_20,code=sm_20 -gencode arch=co
mpute_30,code=sm_30 -gencode arch=compute_35,code=sm_35 -I/usr/local/cuda/include -I. -I.. -I/usr/local/cuda/samples/co
mmon/inc -o main.o -c main.cu
```



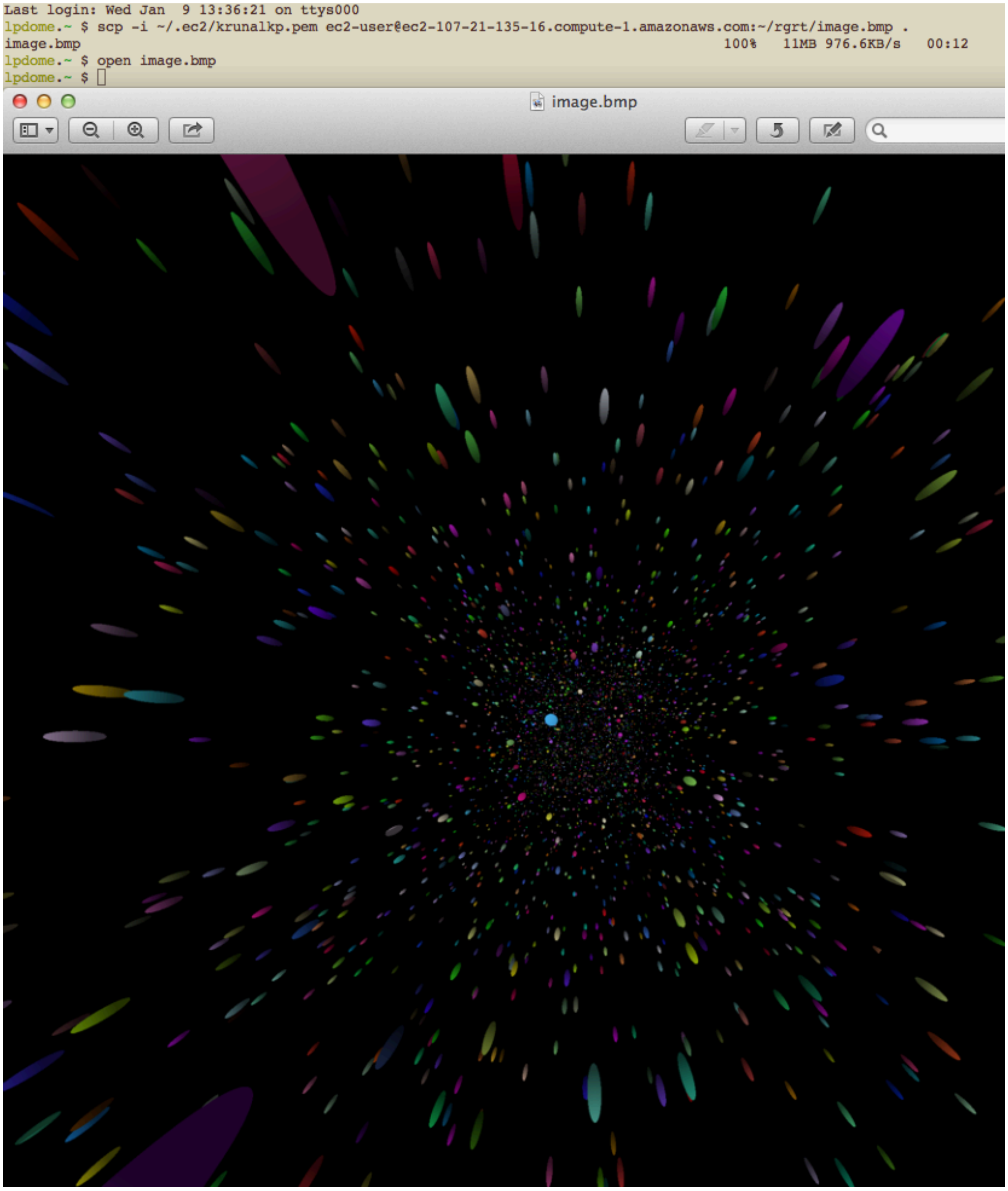
```
g++ -m64 -o rgrt-cuda main.o bmploader.o plyfile.o -L/usr/local/cuda/lib64 -lcudart
mkdir -p ../bin/linux/release
cp rgrt-cuda ../bin/linux/release
make[1]: Leaving directory `/home/ec2-user/rgrt/cuda'
Finished building RGRT
```

Ok, good, now launch it:



```
[ec2-user@ip-10-16-7-196 rgrt]$ bin/linux/release/rgrt-cuda -b=build02 -g=1 -w=2000 -h=2000 -n=1 -o=image.bmp
*** Configuration - b: build02, o: image.bmp, g: 1, w: 2000, h: 2000, n: 1, z: 1.00, m: 0, s: 1, p: 0 ****
Starting ray-tracing...
Reorganizing Data Structures...
  Not going to use host pinned memory...
  Using Grid acceleration...
    Organizing grid cells...
      Stats: # of cells that have 0 objects, 1 object, 2 objects, etc...
      num_cells = 5832
      numZeroes = 302, numOnes = 759, numTwos = 1115
      numThrees = 1232 numGreater = 2424
    Finished grid cell organization.
Finished Consolidation.
World Stats:
  Total # of GeometricObjects: 24379
  Approximate bytes of data: 3901144 bytes
Starting CPU ray tracing...
Starting GPU ray tracing...
  # of threads in a block: 16 x 8 (128)
  # of blocks in a grid  : 125 x 250 (31250)
Calculating accuracy...
  Error : 0.008568
Imaged saved: image.bmp
Finished ray-tracing...
[ec2-user@ip-10-16-7-196 rgrt]$ ls image.bmp
image.bmp
```

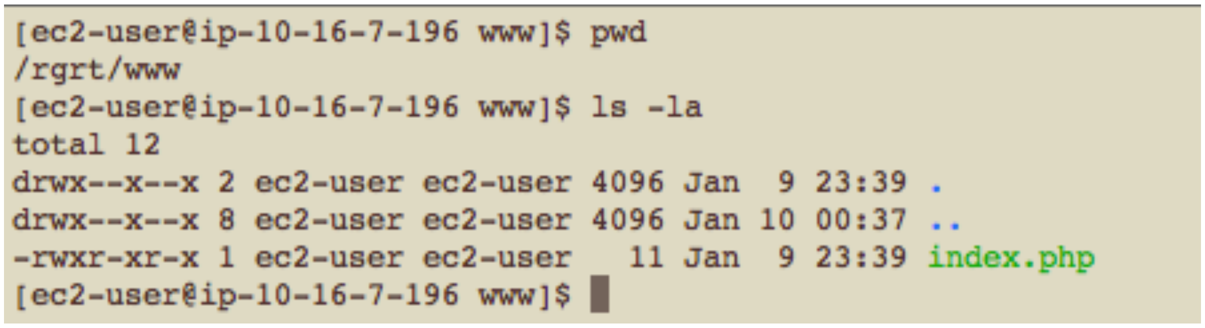
Ok, well it look like it ran just fine, but we need to verify the output image.bmp, let’s scp it to look at it:



```
Last login: Wed Jan  9 13:36:21 on ttys000
lpdome.~ $ scp -i ~/.ec2/krunalkp.pem ec2-user@ec2-107-21-135-16.compute-1.amazonaws.com:~/rgrt/image.bmp .
image.bmp                                                                          100%   11MB 976.6KB/s   00:12
lpdome.~ $ open image.bmp
lpdome.~ $
```

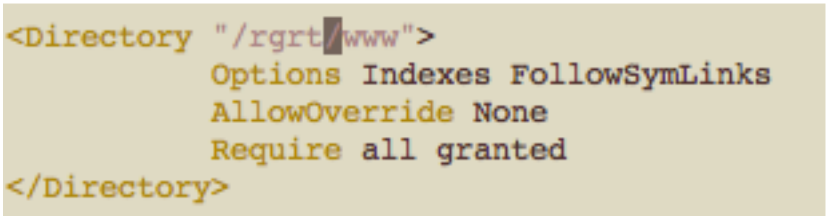
Cool, looks like all the plumbing is in effect.

Now 1 more area of concern. Since we are employing Apache, we need to make a alias to a directory that is web server accessible to our html pages. Instead of copying the files over to /var/www/html, I instead decided to use mod_alias. The first thing I did was copy the rgrt git repo over to the root directory. Then I ensured that /rgrt/www is world accessible:

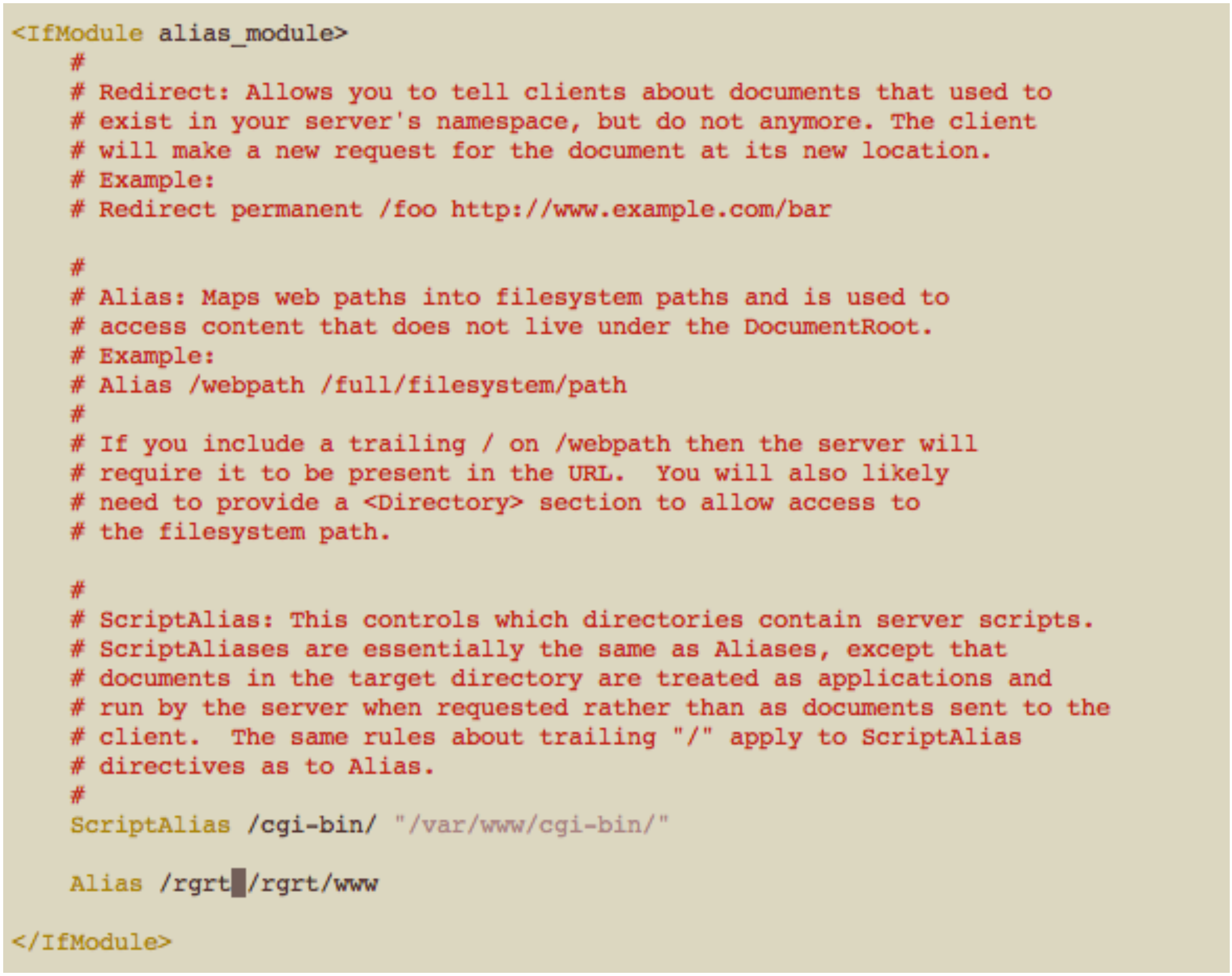


```
[ec2-user@ip-10-16-7-196 www]$ pwd
/rgrt/www
[ec2-user@ip-10-16-7-196 www]$ ls -la
total 12
drwx--x--x 2 ec2-user ec2-user 4096 Jan  9 23:39 .
drwx--x--x 8 ec2-user ec2-user 4096 Jan 10 00:37 ..
-rwxr-xr-x 1 ec2-user ec2-user   11 Jan  9 23:39 index.php
[ec2-user@ip-10-16-7-196 www]$
```

Then we update the httpd.conf file:

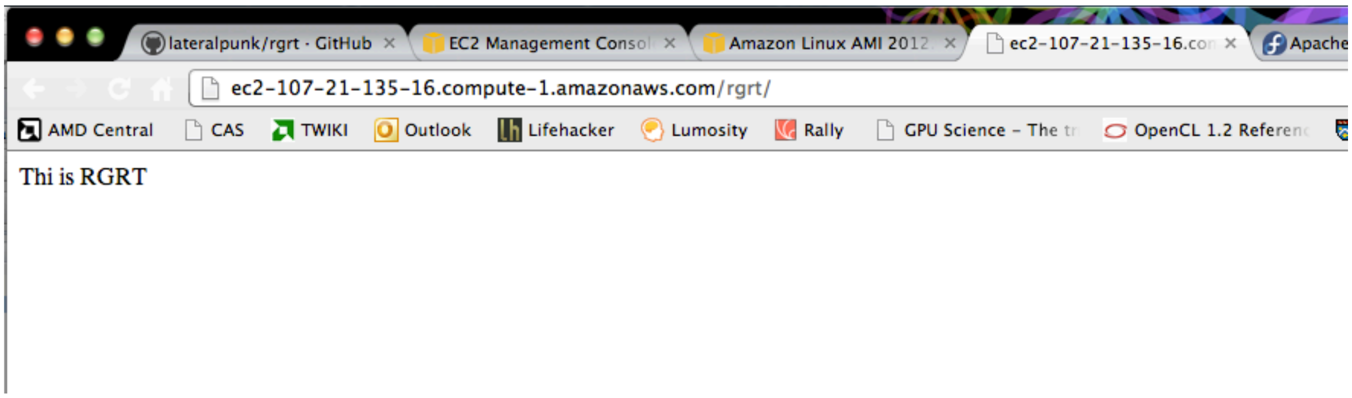


```
<Directory "/rgrt/www">
           Options Indexes FollowSymLinks
           AllowOverride None
           Require all granted
</Directory>
```



```
<IfModule alias_module>
    #
    # Redirect: Allows you to tell clients about documents that used to
    # exist in your server's namespace, but do not anymore. The client
    # will make a new request for the document at its new location.
    # Example:
    # Redirect permanent /foo http://www.example.com/bar

    #
    # Alias: Maps web paths into filesystem paths and is used to
    # access content that does not live under the DocumentRoot.
    # Example:
    # Alias /webpath /full/filesystem/path
    #
    # If you include a trailing / on /webpath then the server will
    # require it to be present in the URL.  You will also likely
    # need to provide a <Directory> section to allow access to
    # the filesystem path.

    #
    # ScriptAlias: This controls which directories contain server scripts.
    # ScriptAliases are essentially the same as Aliases, except that
    # documents in the target directory are treated as applications and
    # run by the server when requested rather than as documents sent to the
    # client.  The same rules about trailing "/" apply to ScriptAlias
    # directives as to Alias.
    #
    ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"

    Alias /rgrt /rgrt/www

</IfModule>
```

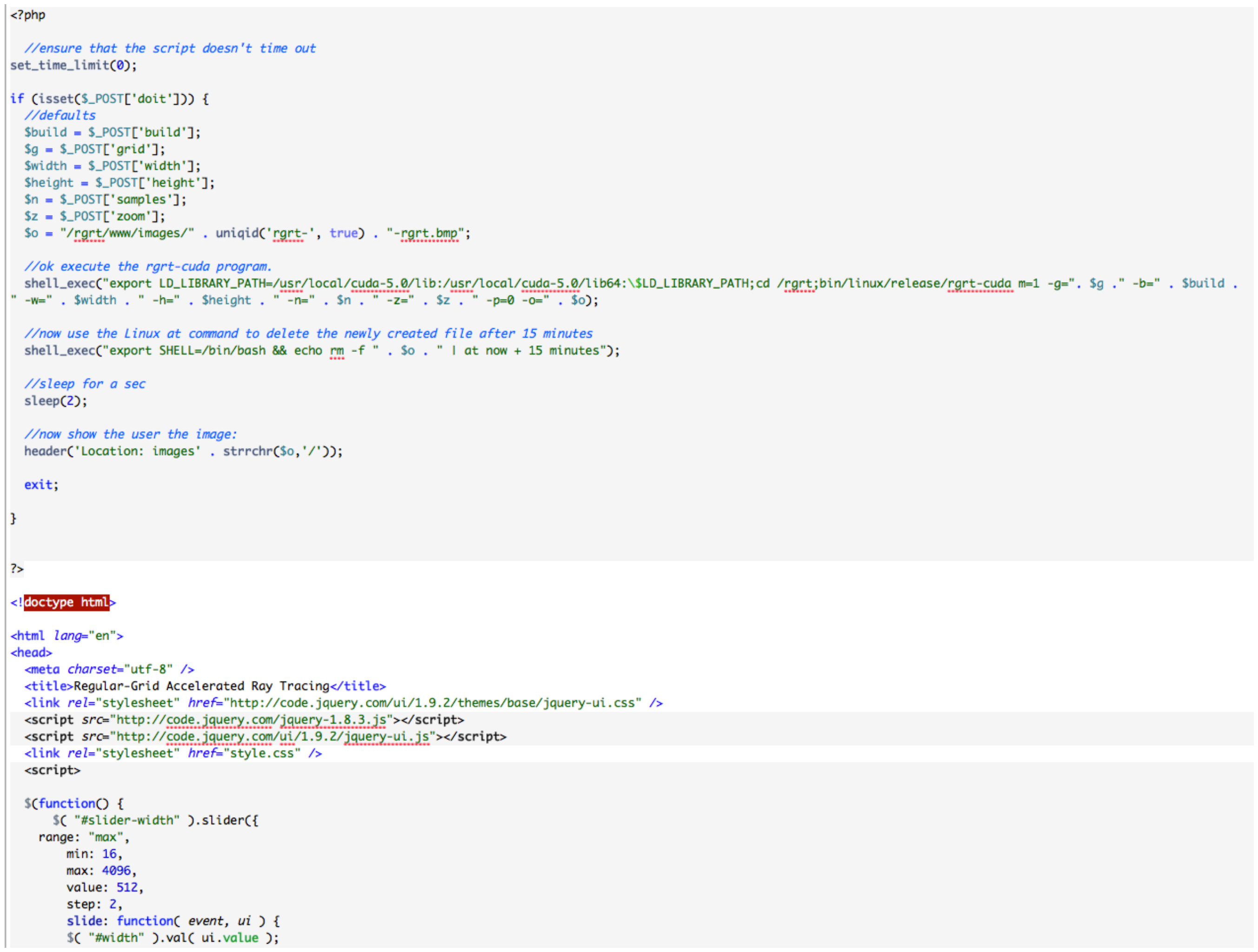
Then we can finally access our website (there's nothing there right now):



ec2-107-21-135-16.compute-1.amazonaws.com/rgrt/

Thi is RGRT

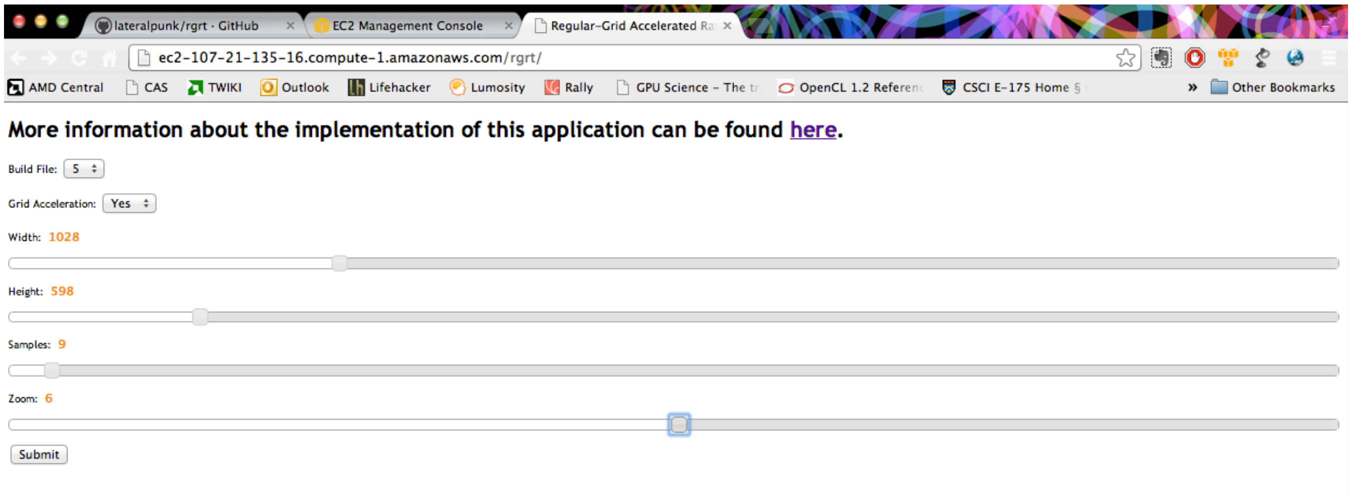
Now, we implement the web service in PHP. This service will feature a form that allows a user to select a build scene and adjust some paramaters:



```
<?php

  //ensure that the script doesn't time out
set_time_limit(0);

if (isset($_POST['doit'])) {
  //defaults
  $build = $_POST['build'];
  $g = $_POST['grid'];
  $width = $_POST['width'];
  $height = $_POST['height'];
  $n = $_POST['samples'];
  $z = $_POST['zoom'];
  $o = "/rgrt/www/images/" . uniqid('rgrt-', true) . "-rgrt.bmp";

  //ok execute the rgrt-cuda program.
  shell_exec("export LD_LIBRARY_PATH=/usr/local/cuda-5.0/lib:/usr/local/cuda-5.0/lib64:\$LD_LIBRARY_PATH;cd /rgrt;bin/linux/release/rgrt-cuda m=1 -g=". $g ." -b=" . $build .
" -w=" . $width . " -h=" . $height . " -n=" . $n . " -z=" . $z . " -p=0 -o=" . $o);

  //now use the Linux at command to delete the newly created file after 15 minutes
  shell_exec("export SHELL=/bin/bash && echo rm -f " . $o . " | at now + 15 minutes");

  //sleep for a sec
  sleep(2);

  //now show the user the image:
  header('Location: images' . strrchr($o,'/'));

  exit;

}

?>

<!doctype html>

<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>Regular-Grid Accelerated Ray Tracing</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.9.2/themes/base/jquery-ui.css" />
  <script src="http://code.jquery.com/jquery-1.8.3.js"></script>
  <script src="http://code.jquery.com/ui/1.9.2/jquery-ui.js"></script>
  <link rel="stylesheet" href="style.css" />
  <script>

  $(function() {
      $( "#slider-width" ).slider({
    range: "max",
        min: 16,
        max: 4096,
        value: 512,
        step: 2,
        slide: function( event, ui ) {
        $( "#width" ).val( ui.value );
      }
    });
      $( "#width" ).val( $( "#slider-width" ).slider( "value" ) );
    });

  $(function() {
      $( "#slider-height" ).slider({
    range: "max",
        min: 16,
        max: 4096,
        value: 512,
        step: 2,
        slide: function( event, ui ) {
        $( "#height" ).val( ui.value );
      }
    });
      $( "#height" ).val( $( "#slider-height" ).slider( "value" ) );
    });

  $(function() {
      $( "#slider-samples" ).slider({
    range: "max",
        min: 1,
        max: 256,
        value: 1,
        step: 1,
        slide: function( event, ui ) {
        $( "#samples" ).val( ui.value );
      }
    });
      $( "#samples" ).val( $( "#slider-samples" ).slider( "value" ) );
    });

  $(function() {
      $( "#slider-zoom" ).slider({
    range: "max",
        min: -1024,
        max: 1024,
        value: 1,
        step: 1,
        slide: function( event, ui ) {
        $( "#zoom" ).val( ui.value );
      }
    });
      $( "#zoom" ).val( $( "#slider-zoom" ).slider( "value" ) );
    });


  </script>
</head>
<body>

<form action="index.php" method="post">
<input type="hidden" name="doit" value="doit">
```

```
<p>
<label>Build File:</label>
<select name="build">
<option value="build01">1</option>
<option value="build02">2</option>
<option value="build03">3</option>
<option value="build04">4</option>
<option value="build05">5</option>
<option value="build06">6</option>
<option value="build07">7</option>
<option value="build08">8</option>
<option value="build09">9</option>
</select>
</p>

<p>
<label>Grid Acceleration:</label>
<select name="grid">
<option value="1">Yes</option>
<option value="0">No</option>
</select>
</p>

<p>
  <label>Width:</label>
  <input type="text" id="width" name="width" style="border: 0; color: #f6931f; font-weight: bold;" />
</p>
<div id="slider-width"></div>

<p>
  <label>Height:</label>
  <input type="text" id="height" name="height" style="border: 0; color: #f6931f; font-weight: bold;" />
</p>
<div id="slider-height"></div>

<p>
  <label>Samples:</label>
  <input type="text" id="samples" name="samples" style="border: 0; color: #f6931f; font-weight: bold;" />
</p>
<div id="slider-samples"></div>

<p>
  <label>Zoom:</label>
  <input type="text" id="zoom" name="zoom" style="border: 0; color: #f6931f; font-weight: bold;" />
</p>
<div id="slider-zoom"></div>

<br/>
<input type="submit" value="Submit">

</form>


</body>
</html>
```

Now let's test it out:



More information about the implementation of this application can be found here.

Build File: 5

Grid Acceleration: Yes

Width: 1028

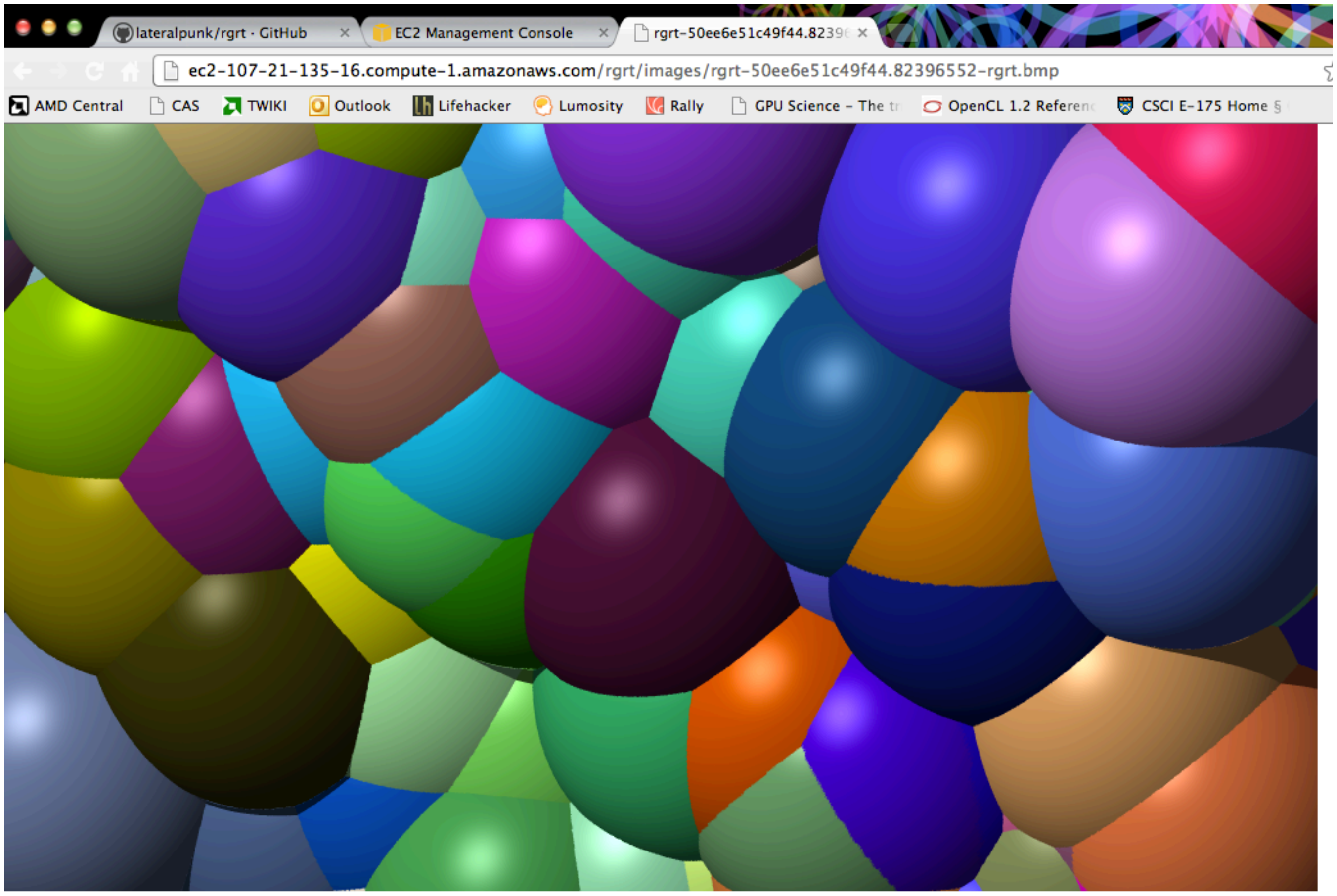
Height: 598

Samples: 9

Zoom: 6

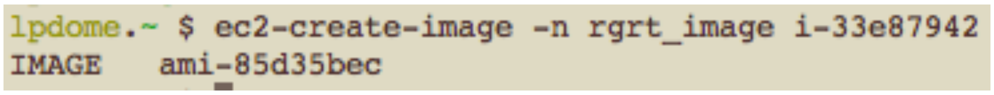
Submit

And after a little while:



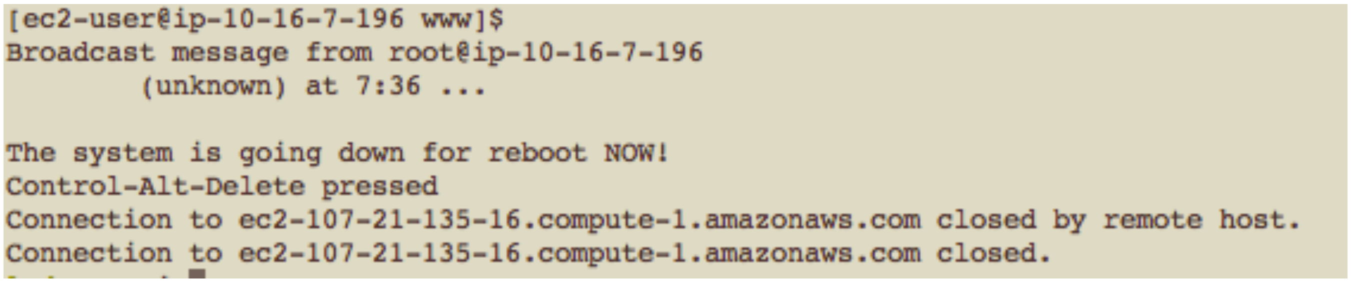
After 15 minutes the above file will automatically delete itself off the server. Try different paramaters and you'll see that everything is dynamic.

Ok, now that the application is complete, let's create an AMI out of this:



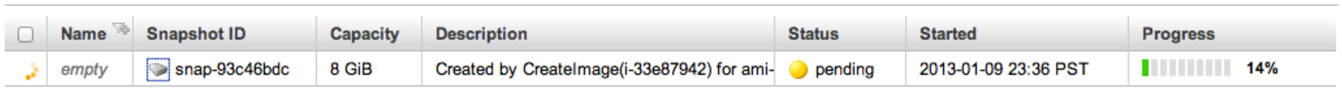
```
lpdome.~ $ ec2-create-image -n rgrt_image i-33e87942
IMAGE    ami-85d35bec
```

We see that the ssh that we were connected to before gets disconnected:



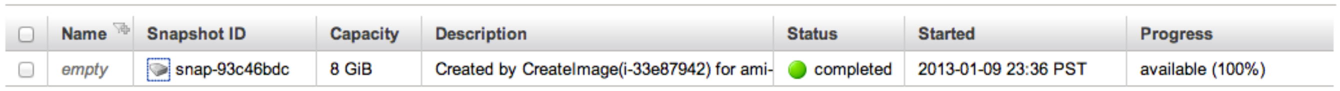
```
[ec2-user@ip-10-16-7-196 www]$
Broadcast message from root@ip-10-16-7-196
        (unknown) at 7:36 ...

The system is going down for reboot NOW!
Control-Alt-Delete pressed
Connection to ec2-107-21-135-16.compute-1.amazonaws.com closed by remote host.
Connection to ec2-107-21-135-16.compute-1.amazonaws.com closed.
```

Checking AWS EC2 we get the following:

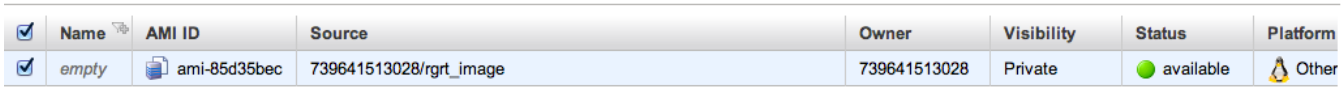


☐	Name	Snapshot ID	Capacity	Description	Status	Started	Progress
	empty	snap-93c46bdc	8 GiB	Created by CreateImage(i-33e87942) for ami-	● pending	2013-01-09 23:36 PST	14%

After waiting for a little while, our AMI has been created:

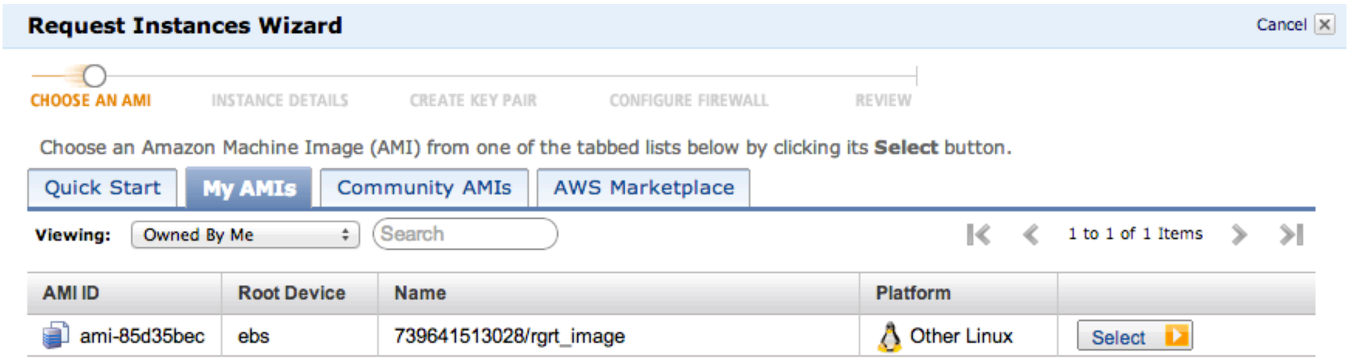


☐	Name	Snapshot ID	Capacity	Description	Status	Started	Progress
☐	*empty*	snap-93c46bdc	8 GiB	Created by CreateImage(i-33e87942) for ami-	● completed	2013-01-09 23:36 PST	available (100%)



☑	Name	AMI ID	Source	Owner	Visibility	Status	Platform
☑	*empty*	ami-85d35bec	739641513028/rgrt_image	739641513028	Private	● available	Other

Ok good. Now let's launch a spot instance of the AMI we just created. We are going to leave this 1 running so that you can evaluate it. We'll shut down the original instance so that we don't get charged crazy. **NOTE: change of plan, I wanted to document all this down right now, but I'll start up an instance that is running right before I submit my assignment to you so that I don't incur unnecessary charges.**



Request Instances Wizard Cancel

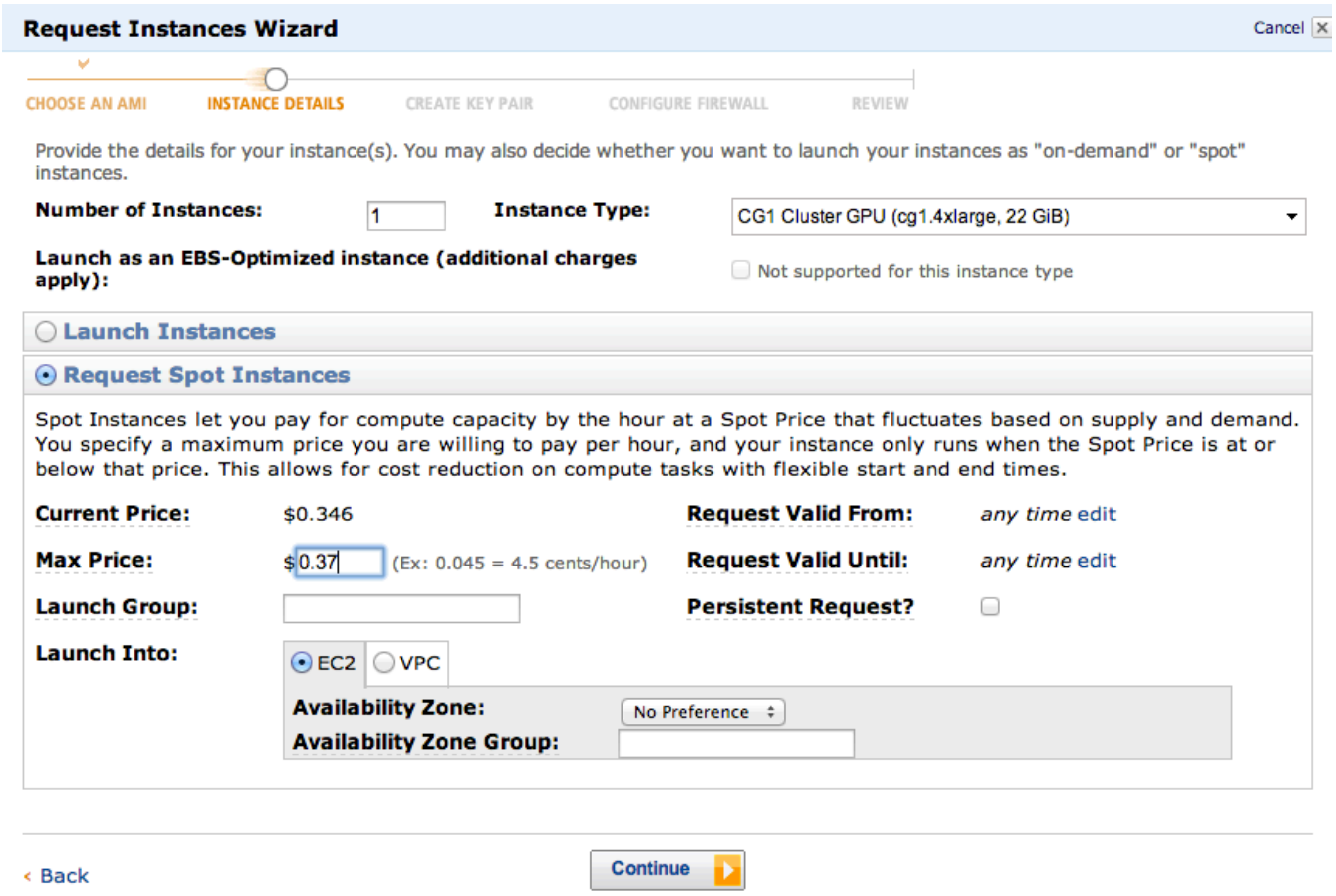
CHOOSE AN AMI | INSTANCE DETAILS | CREATE KEY PAIR | CONFIGURE FIREWALL | REVIEW

Choose an Amazon Machine Image (AMI) from one of the tabbed lists below by clicking its **Select** button.

Quick Start | **My AMIs** | Community AMIs | AWS Marketplace

Viewing: Owned By Me | Search | 1 to 1 of 1 Items

AMI ID	Root Device	Name	Platform	
ami-85d35bec	ebs	739641513028/rgrt_image	Other Linux	Select



Request Instances Wizard Cancel

CHOOSE AN AMI | INSTANCE DETAILS | CREATE KEY PAIR | CONFIGURE FIREWALL | REVIEW

Provide the details for your instance(s). You may also decide whether you want to launch your instances as "on-demand" or "spot" instances.

Number of Instances: 1 **Instance Type:** CG1 Cluster GPU (cg1.4xlarge, 22 GiB)

Launch as an EBS-Optimized instance (additional charges apply): ☐ Not supported for this instance type

○ Launch Instances

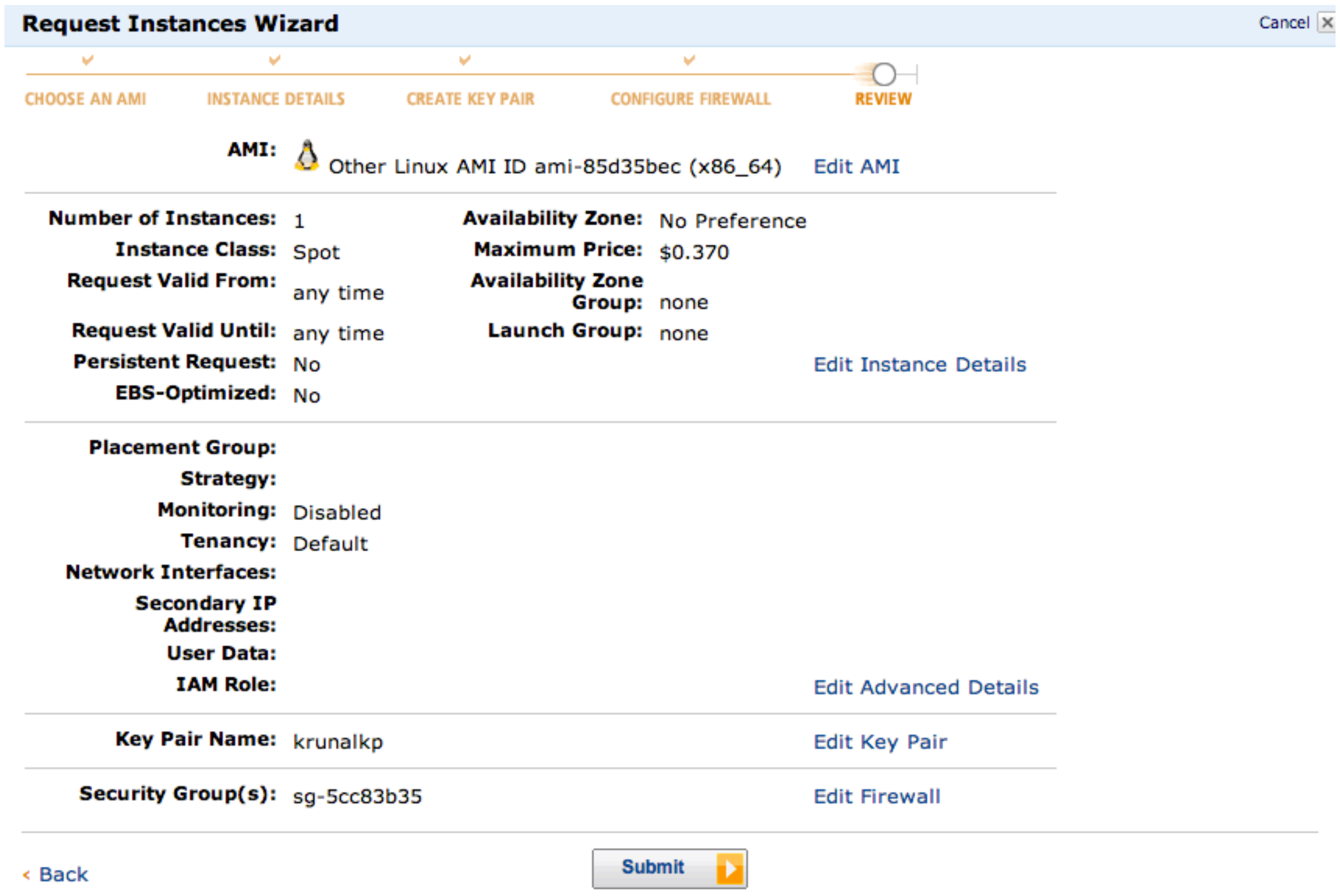
● Request Spot Instances

Spot Instances let you pay for compute capacity by the hour at a Spot Price that fluctuates based on supply and demand. You specify a maximum price you are willing to pay per hour, and your instance only runs when the Spot Price is at or below that price. This allows for cost reduction on compute tasks with flexible start and end times.

Current Price:	$0.346	**Request Valid From:**	*any time* edit
Max Price:	$0.37 (Ex: 0.045 = 4.5 cents/hour)	**Request Valid Until:**	*any time* edit
Launch Group:		**Persistent Request?**	☐
Launch Into:	● EC2 ○ VPC		

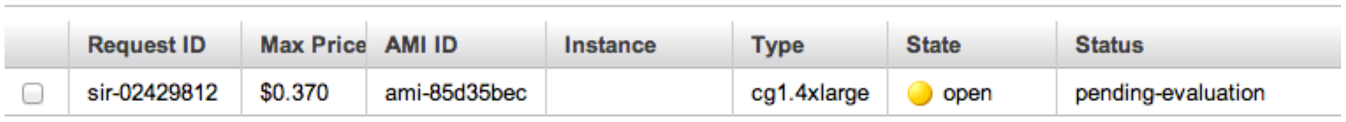
Availability Zone: No Preference
Availability Zone Group:

‹ Back | Continue



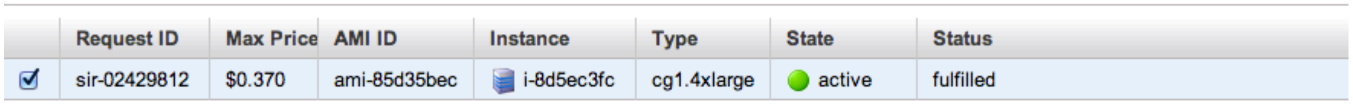
AMI:	Other Linux AMI ID ami-85d35bec (x86_64)		
Number of Instances:	1	Availability Zone:	No Preference
Instance Class:	Spot	Maximum Price:	$0.370
Request Valid From:	any time	Availability Zone Group:	none
Request Valid Until:	any time	Launch Group:	none
Persistent Request:	No		
EBS-Optimized:	No		
Placement Group:			
Strategy:			
Monitoring:	Disabled		
Tenancy:	Default		
Network Interfaces:			
Secondary IP Addresses:			
User Data:			
IAM Role:			
Key Pair Name:	krunalkp		
Security Group(s):	sg-5cc83b35		

And then we wait for a spot to appear:

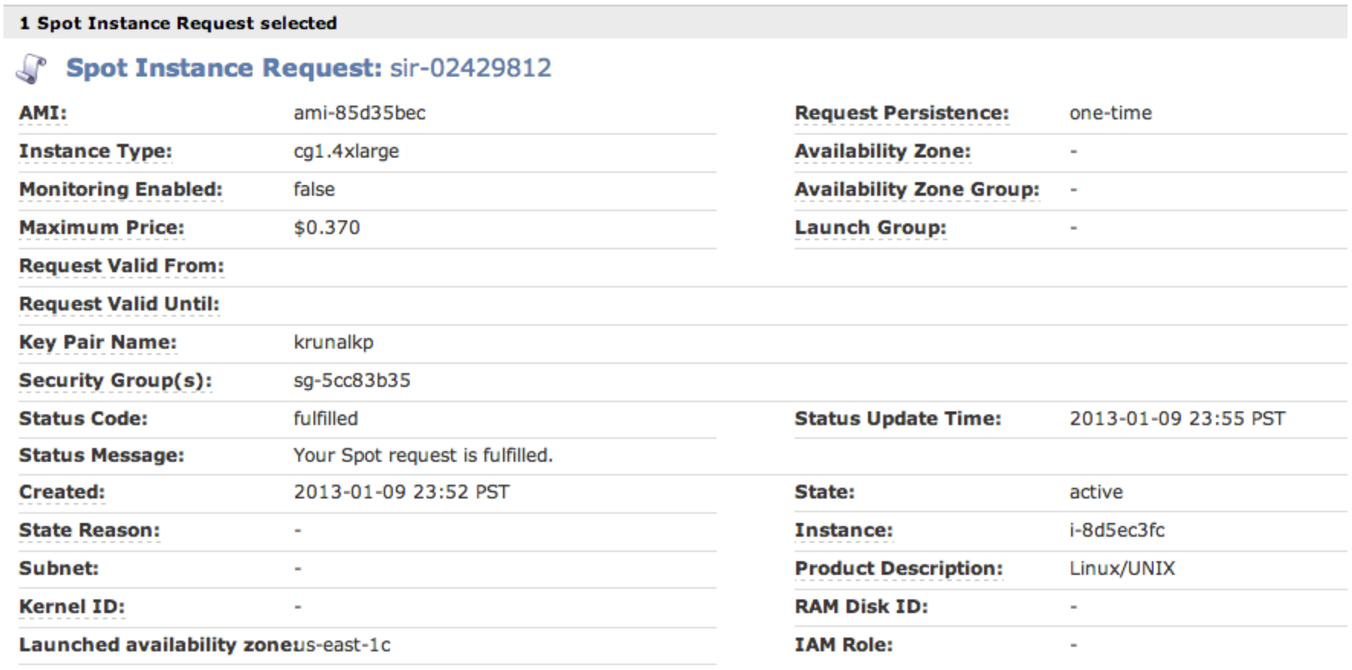


	Request ID	Max Price	AMI ID	Instance	Type	State	Status
☐	sir-02429812	$0.370	ami-85d35bec		cg1.4xlarge	● open	pending-evaluation

Got it:



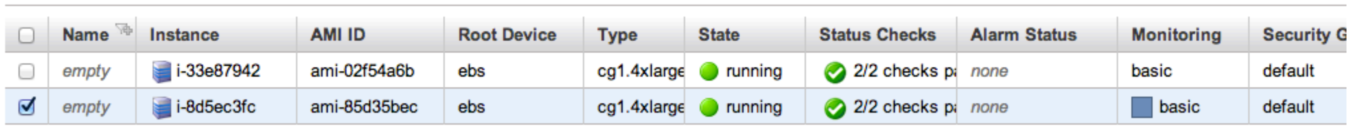
	Request ID	Max Price	AMI ID	Instance	Type	State	Status
✓	sir-02429812	$0.370	ami-85d35bec	i-8d5ec3fc	cg1.4xlarge	● active	fulfilled



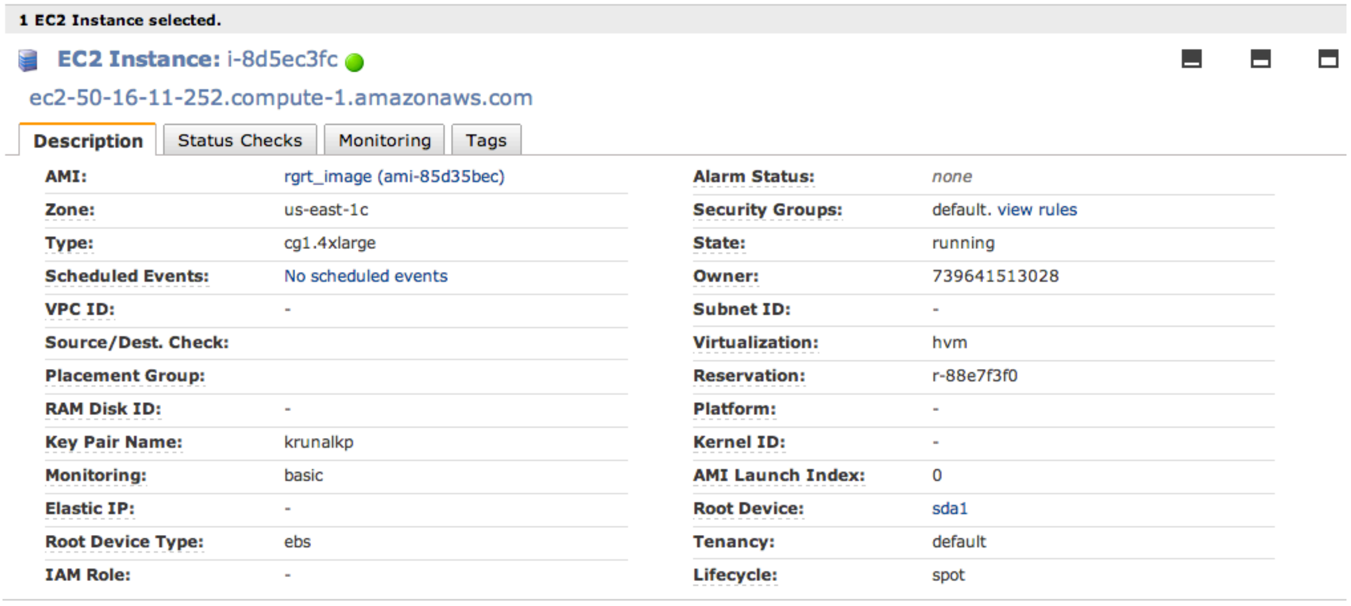
1 Spot Instance Request selected

Spot Instance Request: sir-02429812

AMI:	ami-85d35bec	Request Persistence:	one-time
Instance Type:	cg1.4xlarge	Availability Zone:	-
Monitoring Enabled:	false	Availability Zone Group:	-
Maximum Price:	$0.370	Launch Group:	-
Request Valid From:			
Request Valid Until:			
Key Pair Name:	krunalkp		
Security Group(s):	sg-5cc83b35		
Status Code:	fulfilled	Status Update Time:	2013-01-09 23:55 PST
Status Message:	Your Spot request is fulfilled.		
Created:	2013-01-09 23:52 PST	State:	active
State Reason:	-	Instance:	i-8d5ec3fc
Subnet:	-	Product Description:	Linux/UNIX
Kernel ID:	-	RAM Disk ID:	-
Launched availability zone	us-east-1c	IAM Role:	-



	Name	Instance	AMI ID	Root Device	Type	State	Status Checks	Alarm Status	Monitoring	Security G
☐	*empty*	i-33e87942	ami-02f54a6b	ebs	cg1.4xlarge	● running	✓ 2/2 checks pa	*none*	basic	default
☑	*empty*	i-8d5ec3fc	ami-85d35bec	ebs	cg1.4xlarge	● running	✓ 2/2 checks pa	*none*	basic	default



1 EC2 Instance selected.

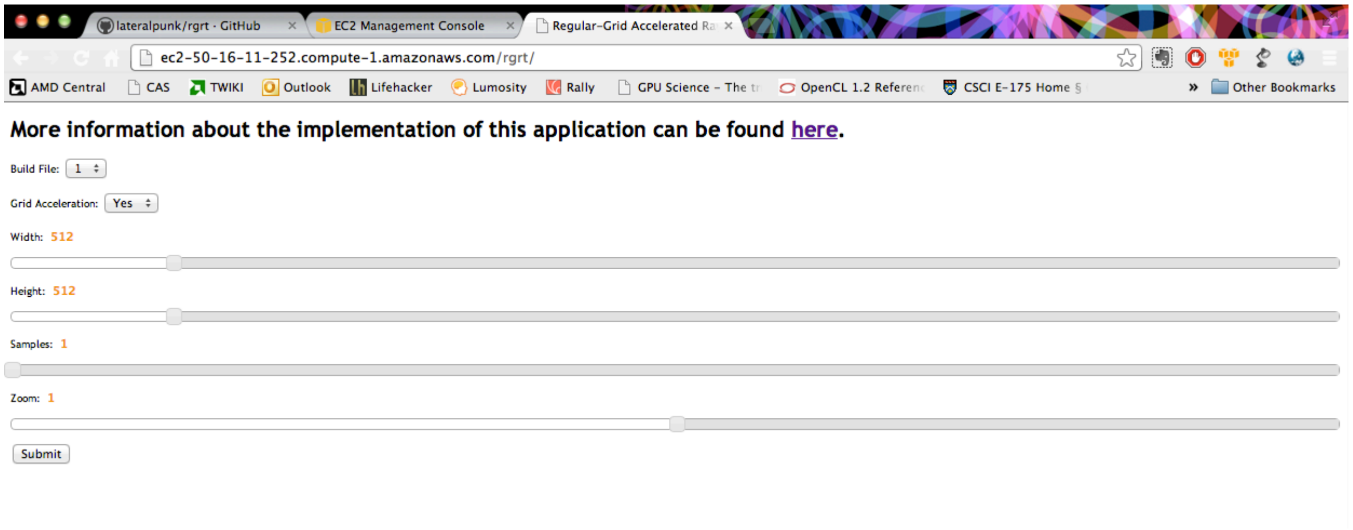
EC2 Instance: i-8d5ec3fc ●

ec2-50-16-11-252.compute-1.amazonaws.com

Description | Status Checks | Monitoring | Tags

AMI:	rgrt_image (ami-85d35bec)	**Alarm Status:**	*none*
Zone:	us-east-1c	**Security Groups:**	default. view rules
Type:	cg1.4xlarge	**State:**	running
Scheduled Events:	No scheduled events	**Owner:**	739641513028
VPC ID:	-	**Subnet ID:**	-
Source/Dest. Check:		**Virtualization:**	hvm
Placement Group:		**Reservation:**	r-88e7f3f0
RAM Disk ID:	-	**Platform:**	-
Key Pair Name:	krunalkp	**Kernel ID:**	-
Monitoring:	basic	**AMI Launch Index:**	0
Elastic IP:	-	**Root Device:**	sda1
Root Device Type:	ebs	**Tenancy:**	default
IAM Role:	-	**Lifecycle:**	spot

At this point, it should be a simple matter of point our browser to the new URL:



ec2-50-16-11-252.compute-1.amazonaws.com/rgrt/

More information about the implementation of this application can be found here.

Build File: 1

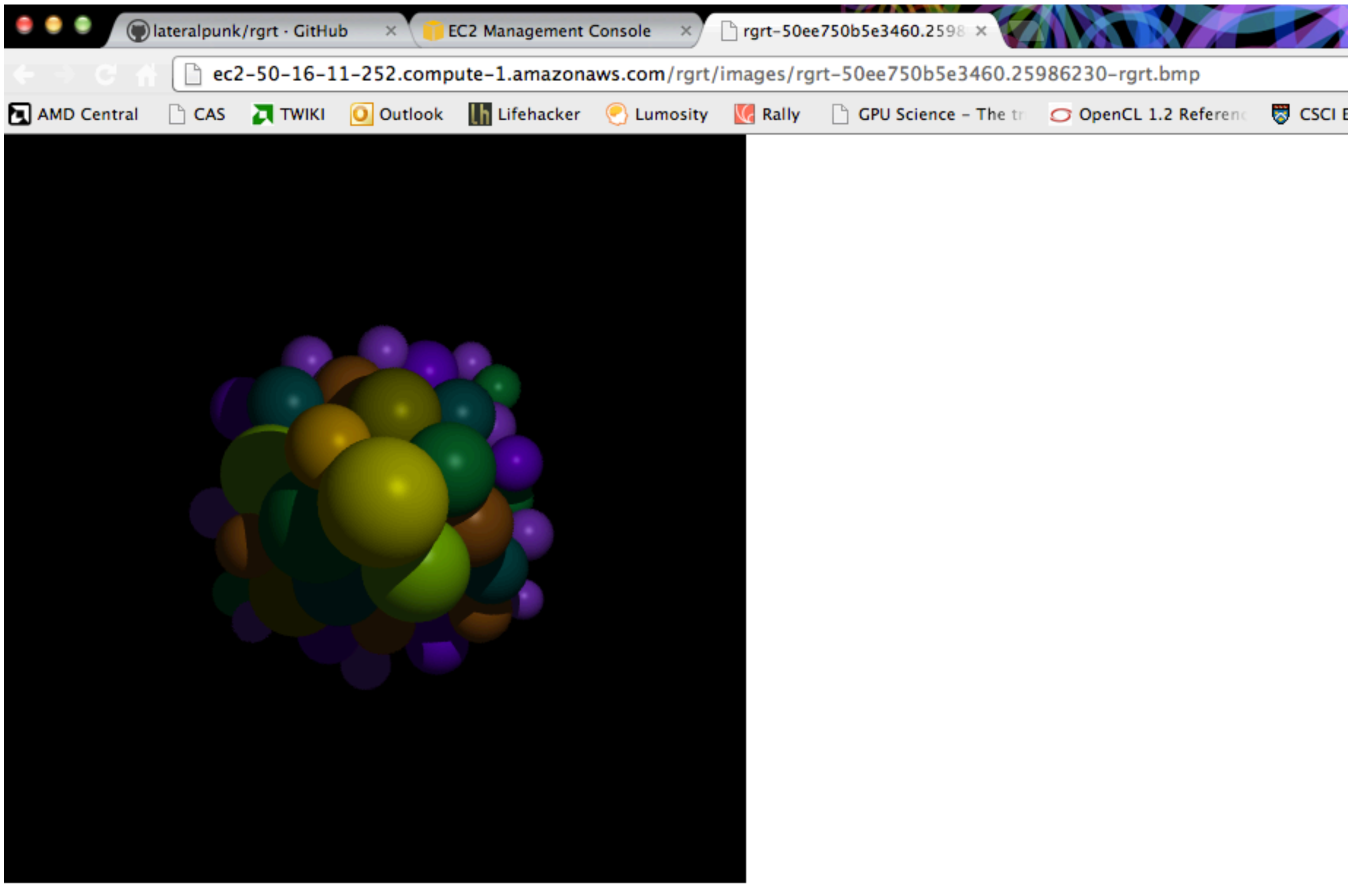
Grid Acceleration: Yes

Width: 512

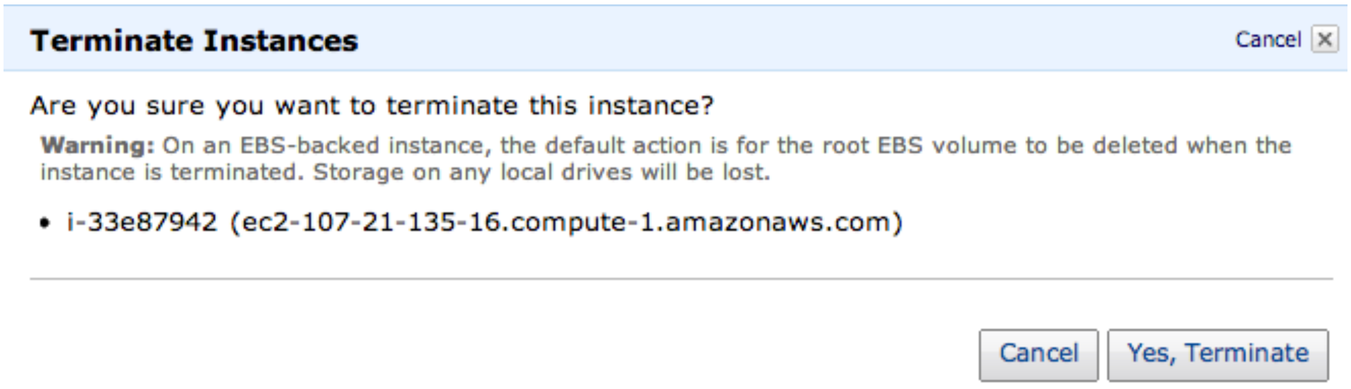
Height: 512

Samples: 1

Zoom: 1



And everything just magically worked! Awesome. Now let's terminate our original instance since it's costing us too much:

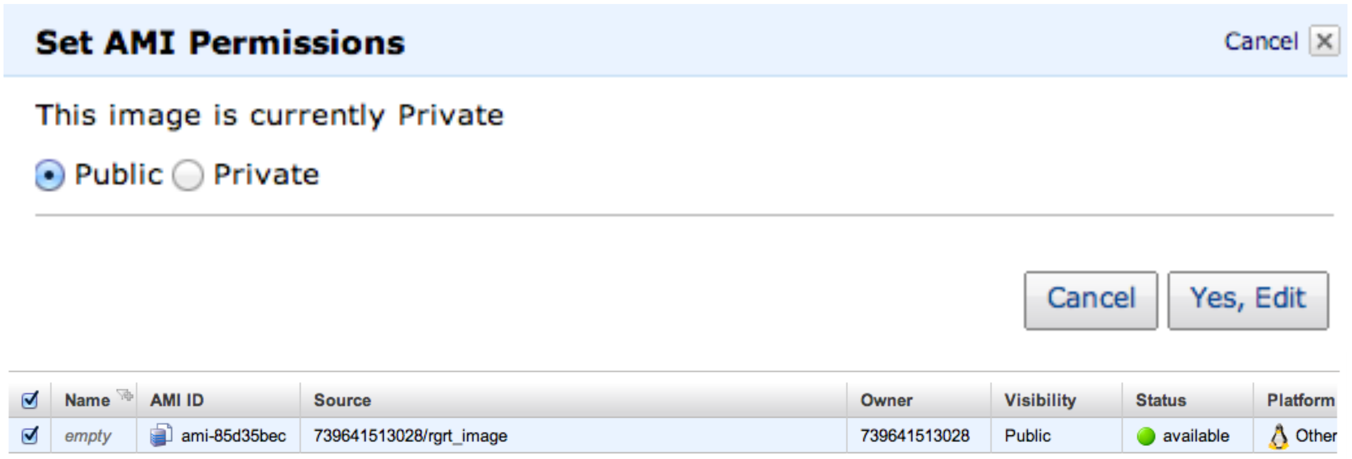

Terminate Instances

Are you sure you want to terminate this instance?

Warning: On an EBS-backed instance, the default action is for the root EBS volume to be deleted when the instance is terminated. Storage on any local drives will be lost.

- i-33e87942 (ec2-107-21-135-16.compute-1.amazonaws.com)

Finally, so that you can evaluate the AMI, we make it's visibility Public:

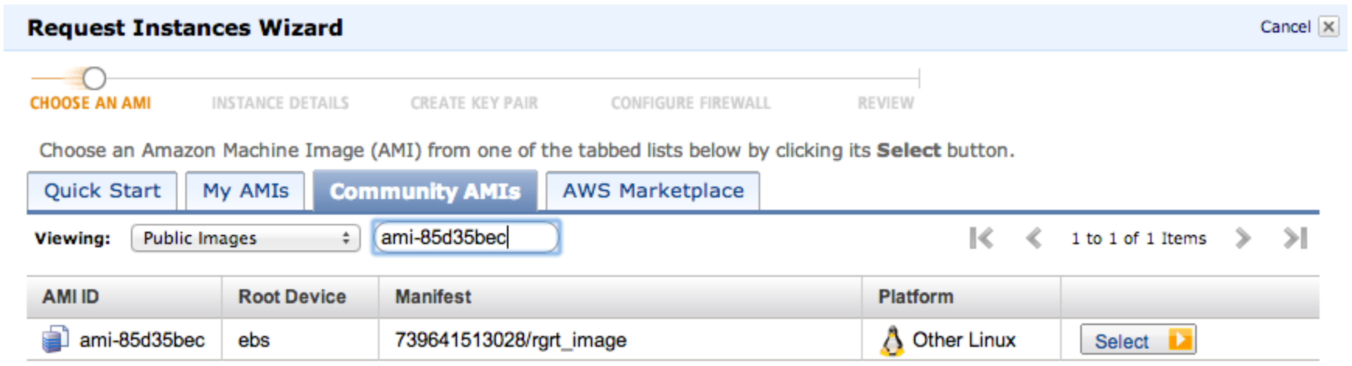

Set AMI Permissions

This image is currently Private

[✓ Public] Private

✓	Name	AMI ID	Source	Owner	Visibility	Status	Platform
✓	empty	ami-85d35bec	739641513028/rgrt_image	739641513028	Public	● available	Other

And you can confirm it is public by:


Request Instances Wizard

Choose an Amazon Machine Image (AMI) from one of the tabbed lists below by clicking its **Select** button.

Viewing: Public Images ami-85d35bec

1 to 1 of 1 Items

AMI ID	Root Device	Manifest	Platform	
ami-85d35bec	ebs	739641513028/rgrt_image	Other Linux	Select

So you can just use the above (make sure instance type is cg1.4xlarge) and everything should work just nicely.

Conclusion

Hence we've shown complete steps on how to setup an AWS EC2 GPU Instance and support our RGRT.