Tidewater Analytics

An informal investigation

Configuring Amazon Web Services (AWS) Elastic Compute Cloud (EC2) for RStudio and R Computing

Author:
Dr. Chuck Cartledge

Founder:
Rick Jones

Saturday 17th October, 2015

Contents

List of Figures

1 Introduction

Amazon Web Services (AWS) and specifically Elastic Cloud Computing (EC2) have been getting a lot of press recently. To quote Amazon documentation:

> *"Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster."*
>
> Amazon Staff [3]

But what does all that mean relative to R and computing on the "cloud?" And even more specifically, how can I run an R script on an AWS EC2 instance?

The following few sections will provide step-by-step directions on how to gain access to the AWS and their EC2 support, on how to use R (specifically RStudio) on an EC2 instance, and then general ideas on how to customize your EC2 instance.

2 Preliminaries

AWS is an Amazon revenue source, meaning that they charge money to access and use their services. EC2 is an AWS service. Amazon also wants to encourage people to use AWS, and so are willing to let people "try out" AWS for free, just so long as they don't use too much of the AWS. Therefore, AWS charges different rates for different level of services.

As of this writing, people that use fewer than 750 hours per month and don't want to store data on the AWS, can have a free account. AWS will ask for a credit card number in case you run over the usage limit. If you require more computing power, then you can request it and you will be charged accordingly.

The rest of the paper assumes you have an AWS account and can access your AWS account starting at this URL:

`http://aws.amazon.com/`

and navigating to the AWS console page (see Figure 1).

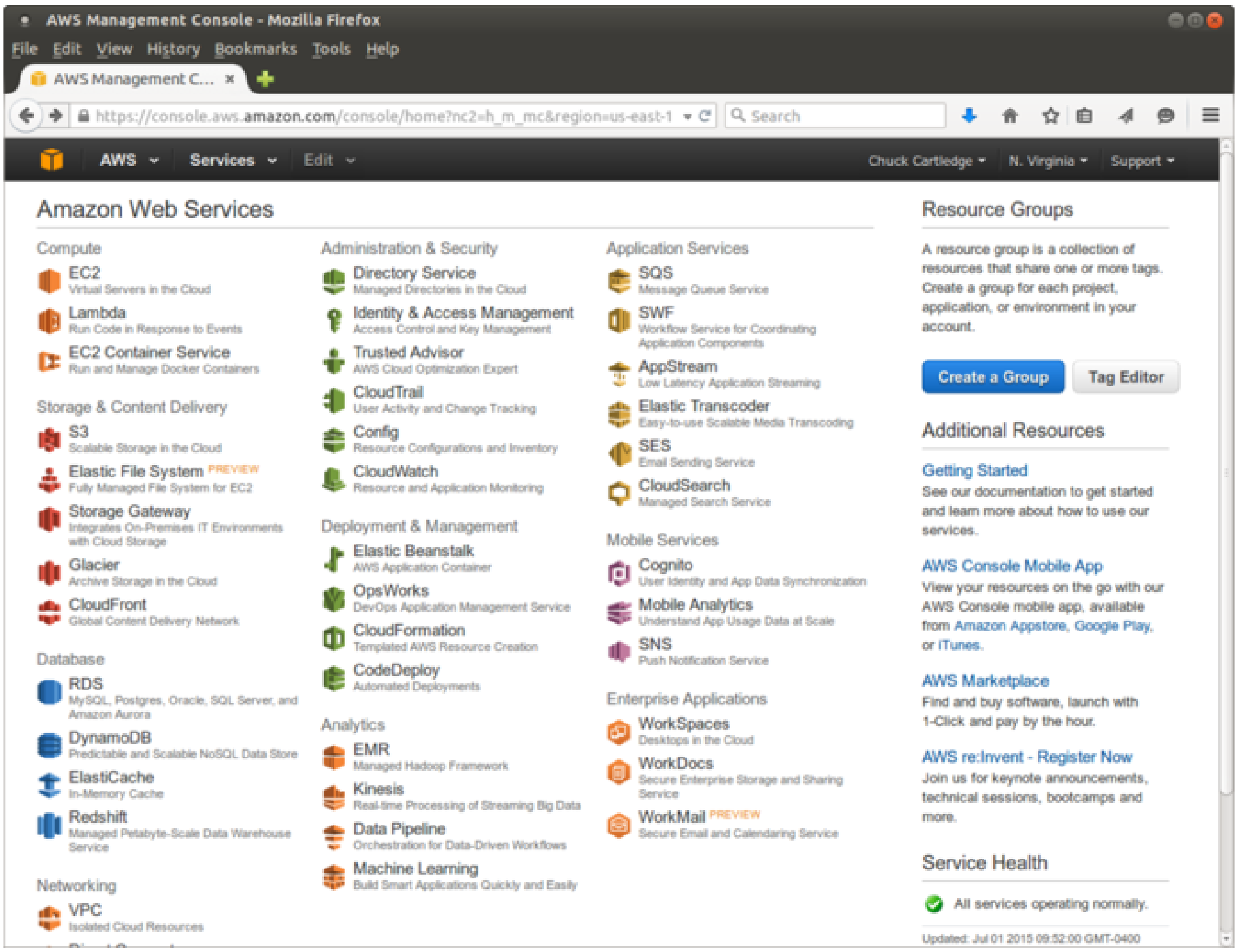

Figure 1: The Amazon Web Services Console. We will be focusing on the EC2 option.

3 An overview of what this document covers

The goal of this document is to list the steps to create a virtual machine in the AWS EC2 environment (see Figure 2), use the virtual machine (see Figure 3), and finally to remove the machine when it is no longer needed or necessary (see Figure 4).

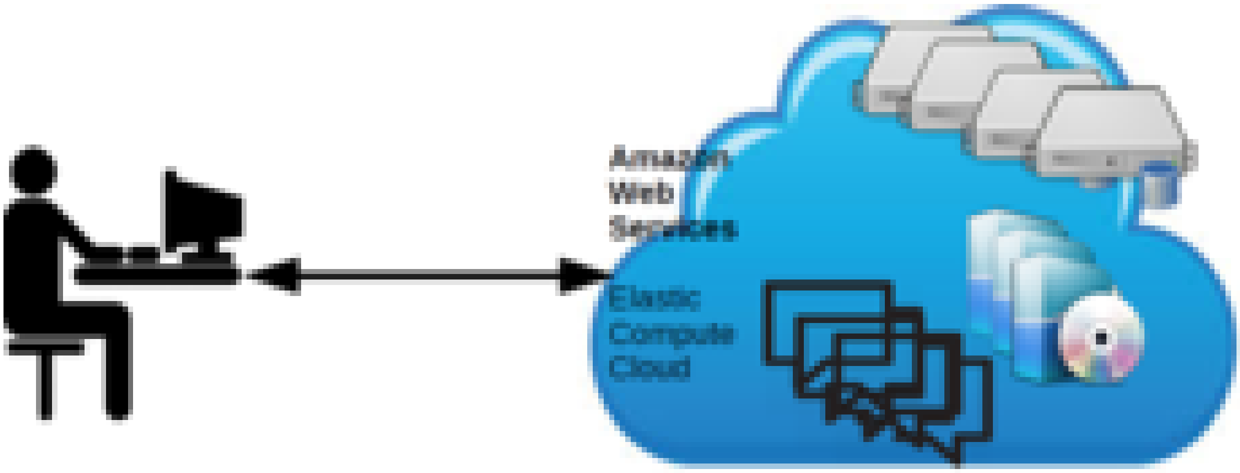

Figure 2: Definition of AWS EC2 virtual machine. The user (you) will connect to the AWS EC2 program and select: *1*) the type of hardware (CPU, RAM, disk storage, etc.) to use, *2*) the operating system and preinstalled software to run on your selected hardware, and *3*) how your local computer will communicate with the virtual machine.

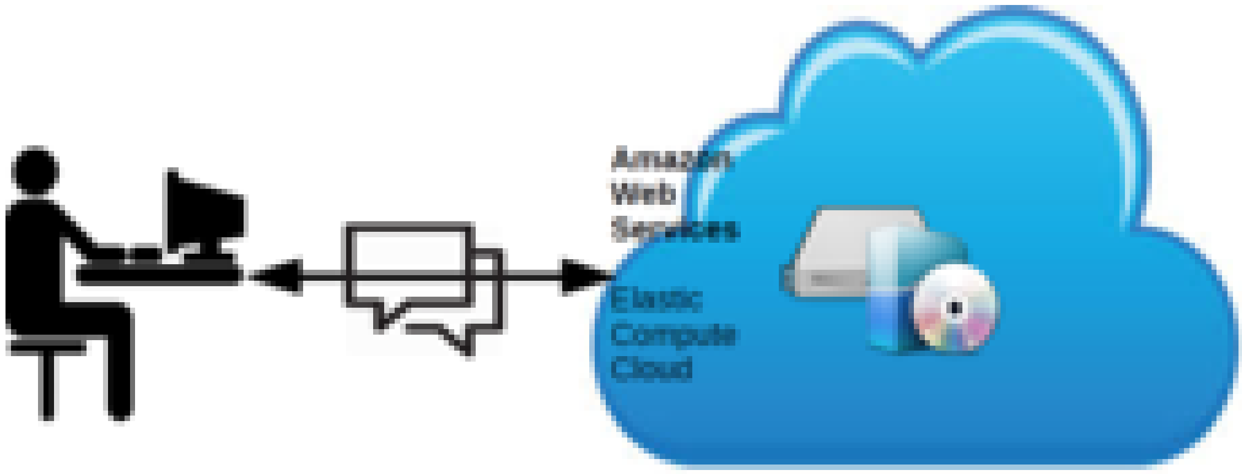

Figure 3: Use the virtual machine. In this document, we will first walk through how to use RStudio via a web interface, and then we will have an example of automating the selection process and executing a program via ssh commands.

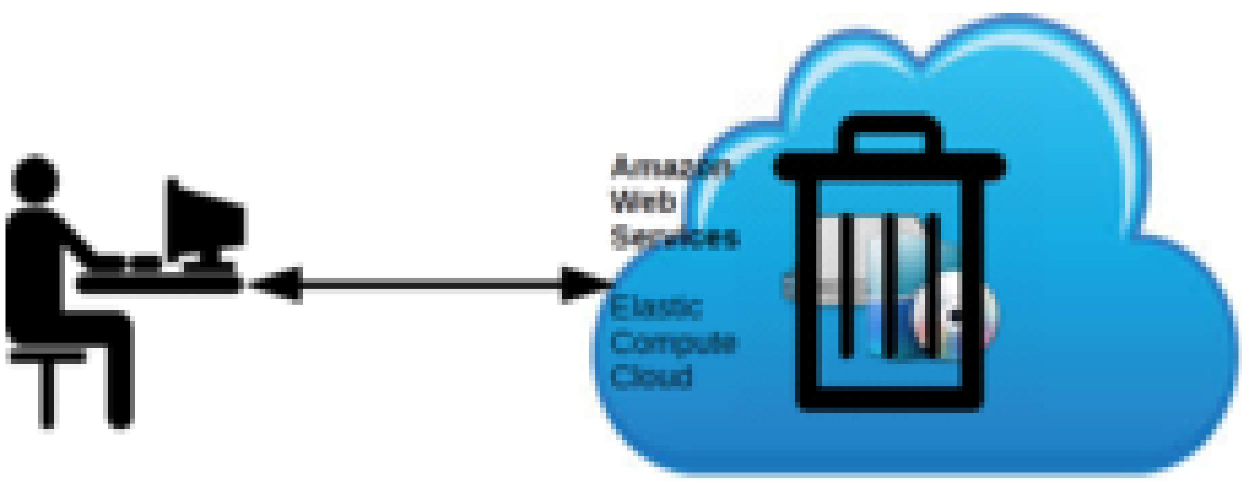

Figure 4: Delete the virtual machine. AWS is a free service, if you use only a limited amount of AWS resources for a short time. If you use a large number of resources, or save your virtual machine then they may charge you.

4 A simple description

AWS provides the capability of creating a set of virtual machines that you can use. Virtual machines don't really exist. A virtual machine is a program that acts appears to be a machine (see Figure 5). At a very macro level, a computer consists of:

- A CPU that does the processing,
- RAM where programs and data live while being executed,
- Disk drive(s) where programs and data live when not being executed, and
- Connections to the outside world (including things like keyboards, mice, displays, LANs, etc.).

Each of these items can be thought of as a layer in a cake, where things flow between the layers (double click for a short virtualization description). Each layer of cake is separated by a layer of frosting that is called an application program interface (API). When one layer of the cake sends data to another layer through the API, the sending layer really has no idea of how things happen on the other side of the API. This "information hiding" allows the layers to be swapped out and replaced by new layers without impacting the way the cake works.

AWS provides the capability to create virtual machines. These virtual machines exist in software running on Amazon hardware in Amazon data centers around the world. The machines are like real machines in that they need an operating system (OS) , and a suite of applications before they are able to do anything. In AWS parlance, the combination of OS and application software is called an Amazon Machine Image (AMI).

To have a functional AWS EC2 instance, you need an AMI.

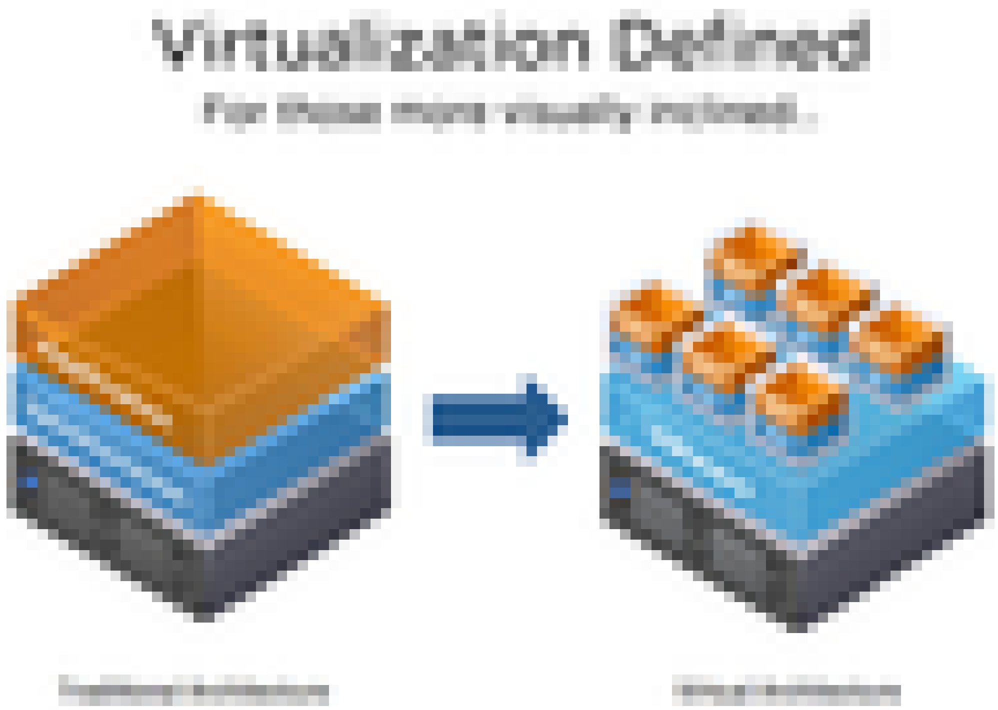

Figure 5: Comparing true machines and virtual machines. A true machine has hardware that you can touch and feel. A virtual machine consists only of software [4].

5 How to select and use an EC2 instance

Now we'll start drilling down into the AWS EC2 world to get to running RStudio. (All the images used in the tutorial were captured using Dr. Cartledge's account, so your screens may be slightly different based on your setting and browser.)

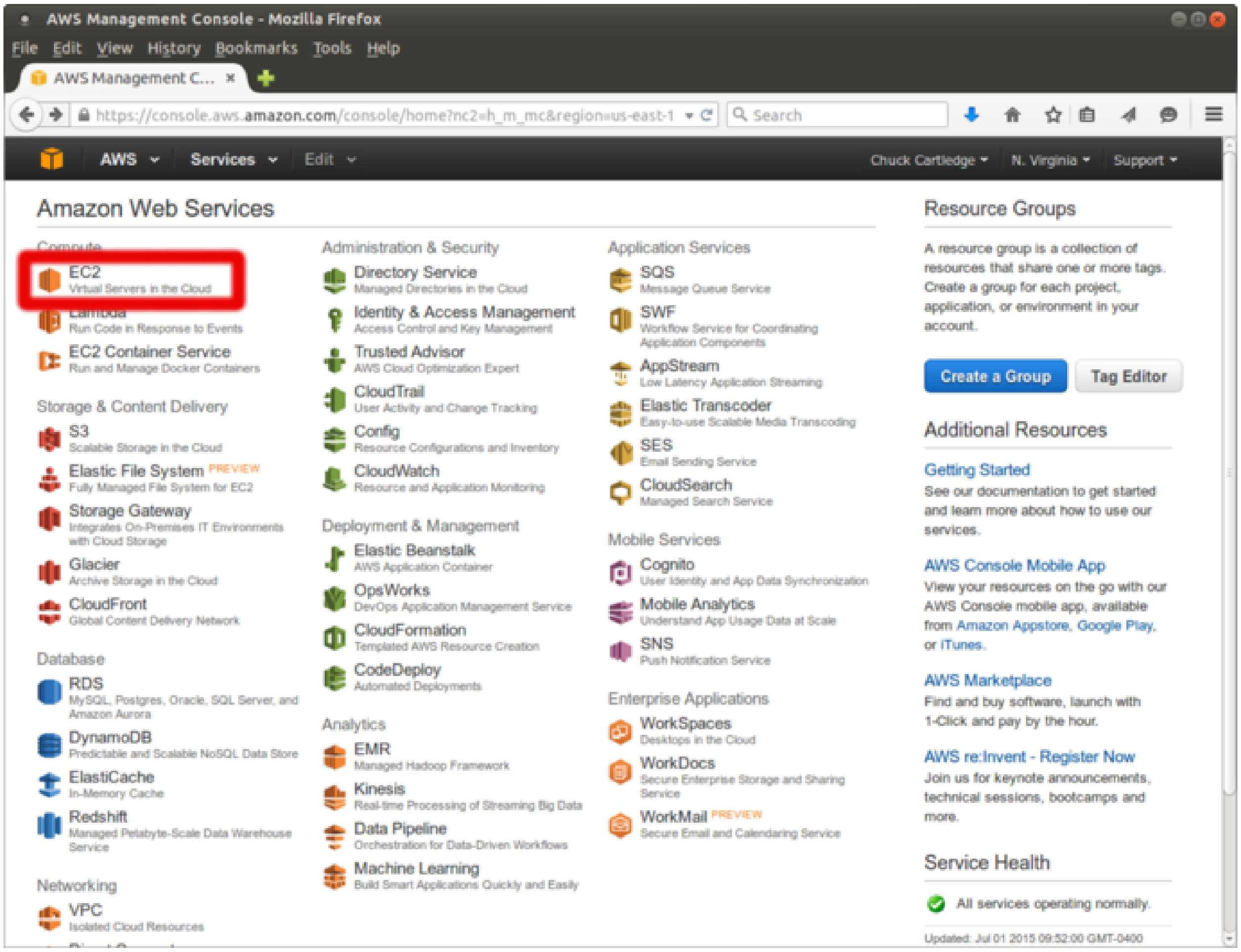

Figure 6: Step 1, follow EC2 link

1. Follow the EC2 link on the AWS console (see Figure 6).

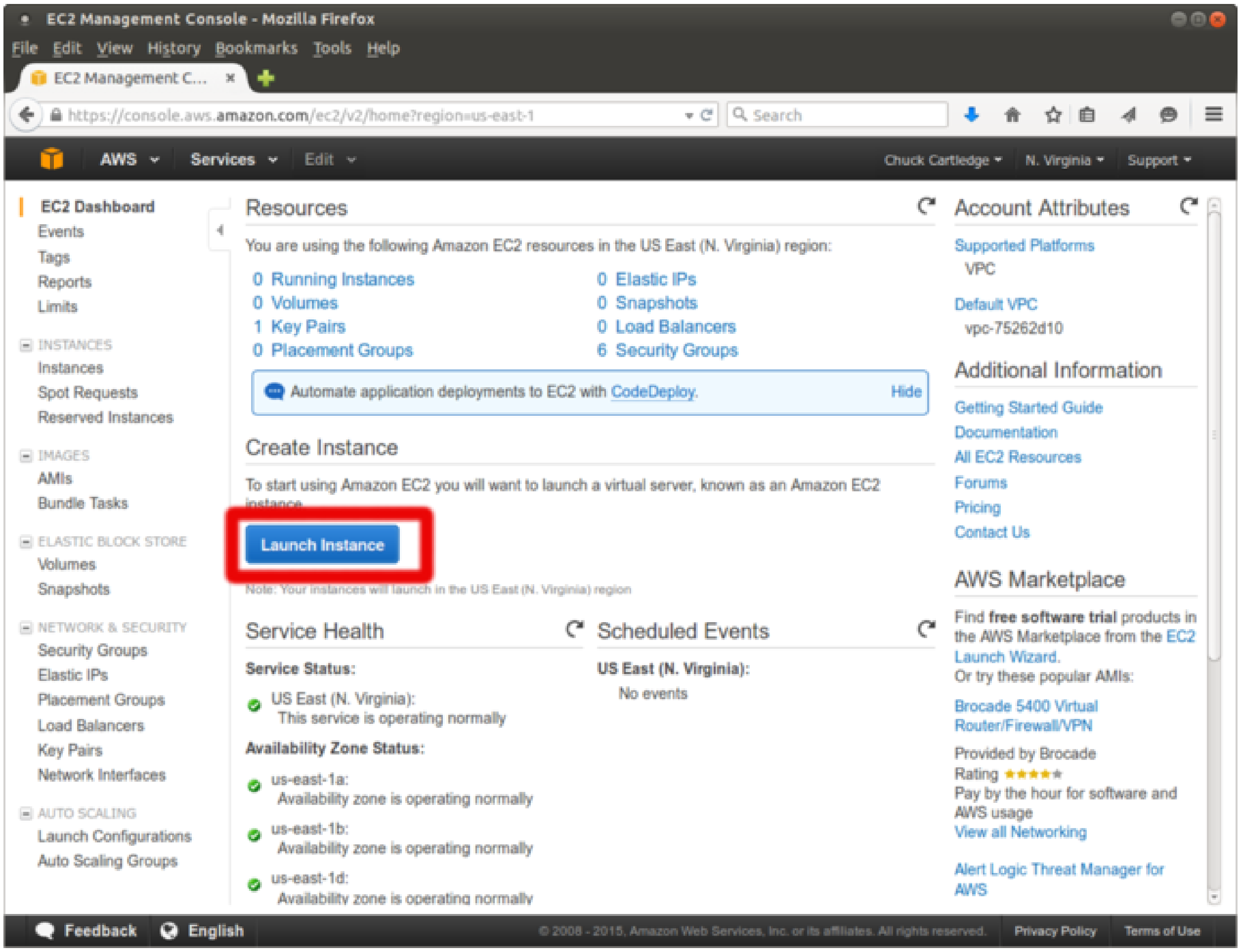

Figure 7: Step 2, click on the “Launch Instance” button This is where we’ll select which AMI to run.

2. Press the “Launch Instance” button (see Figure 7).

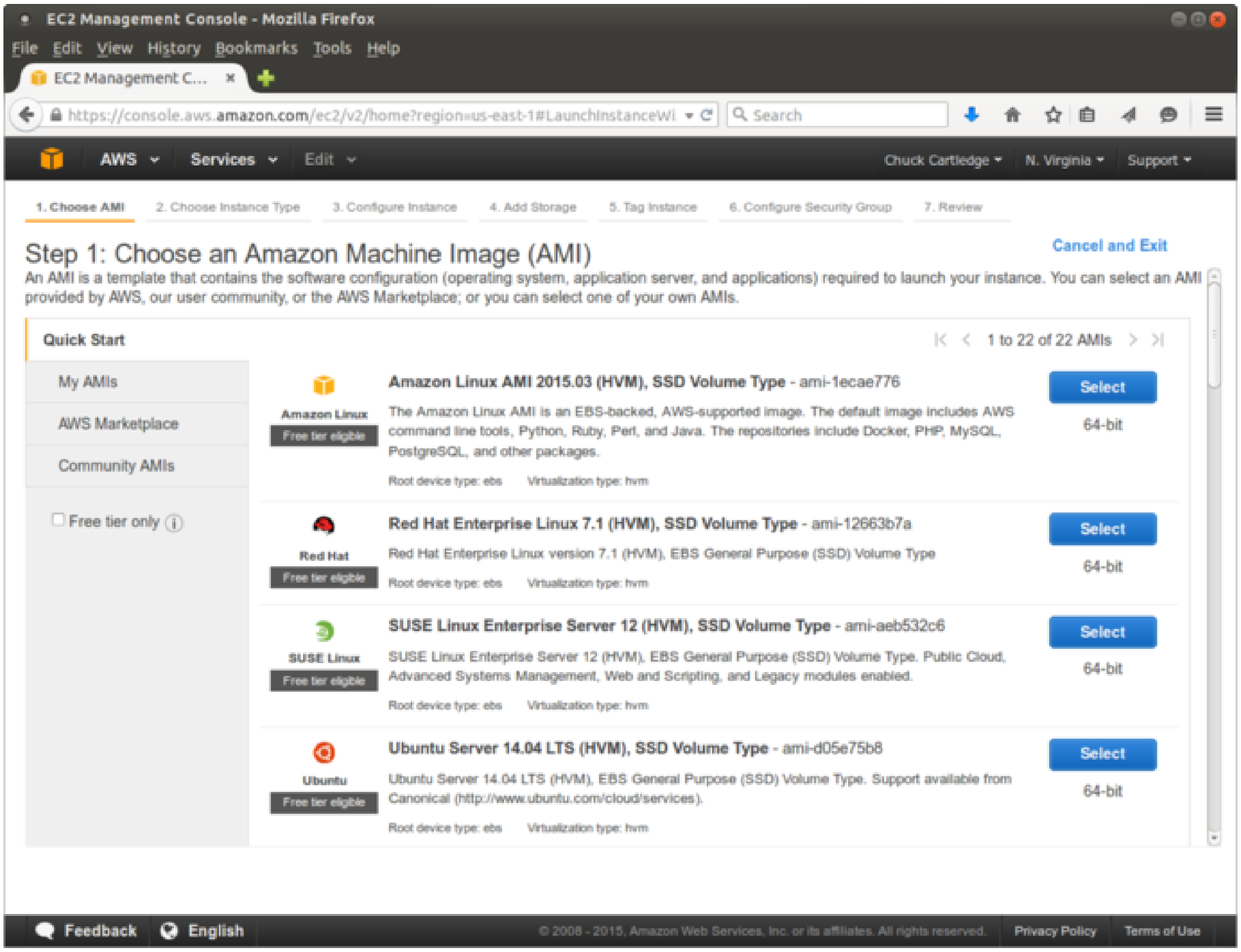

Figure 8: Step 3, choose which AMI to run. As we'll see, there is a plethora of them.

3. Need to select which AMI to run (and there are lots of them) (see Figure 8).

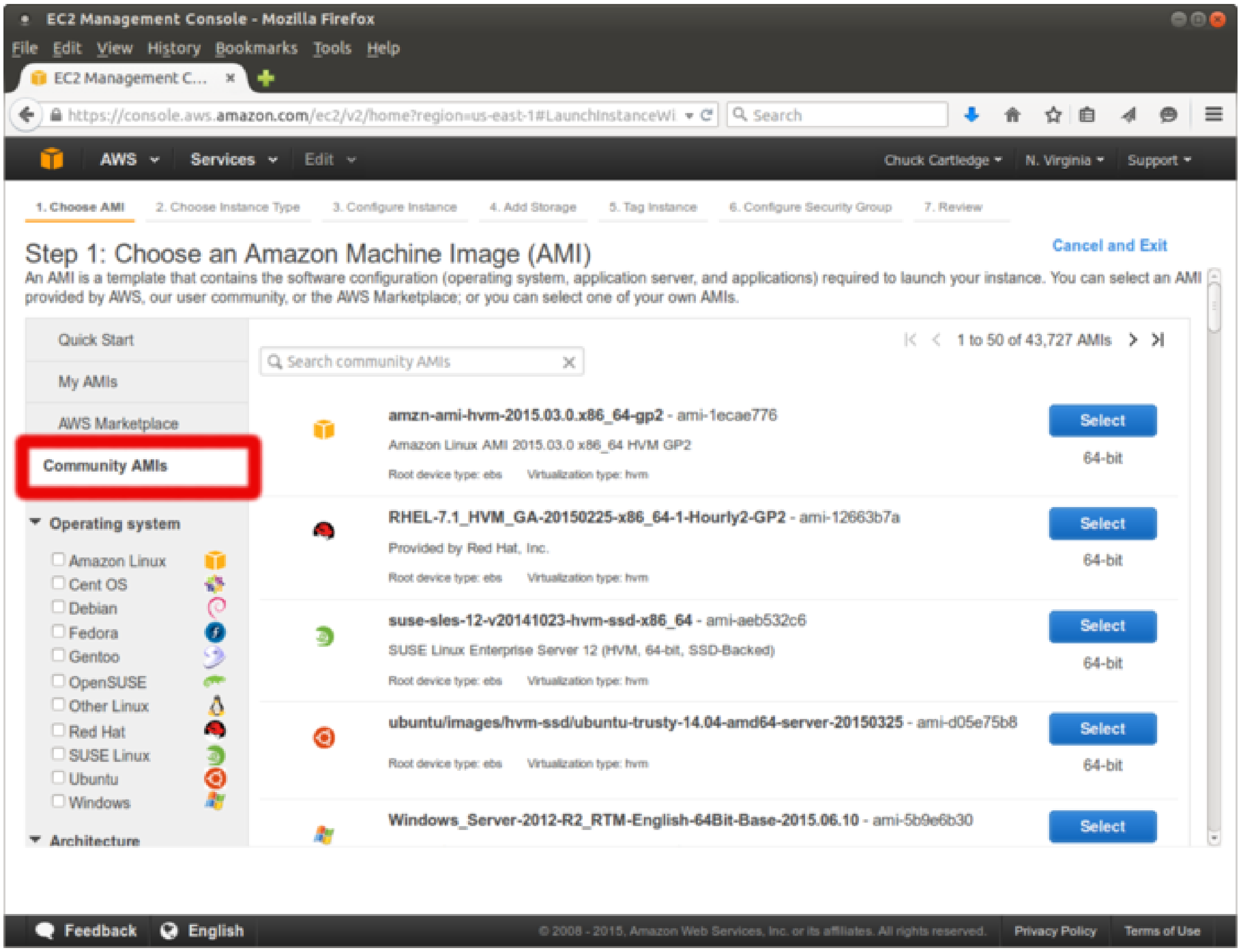

Figure 9: Step 4, choose to select one from the community contributed AMIs.

4. We will choose an AMI that has been contributed by the EC2 community (see Figure 9).

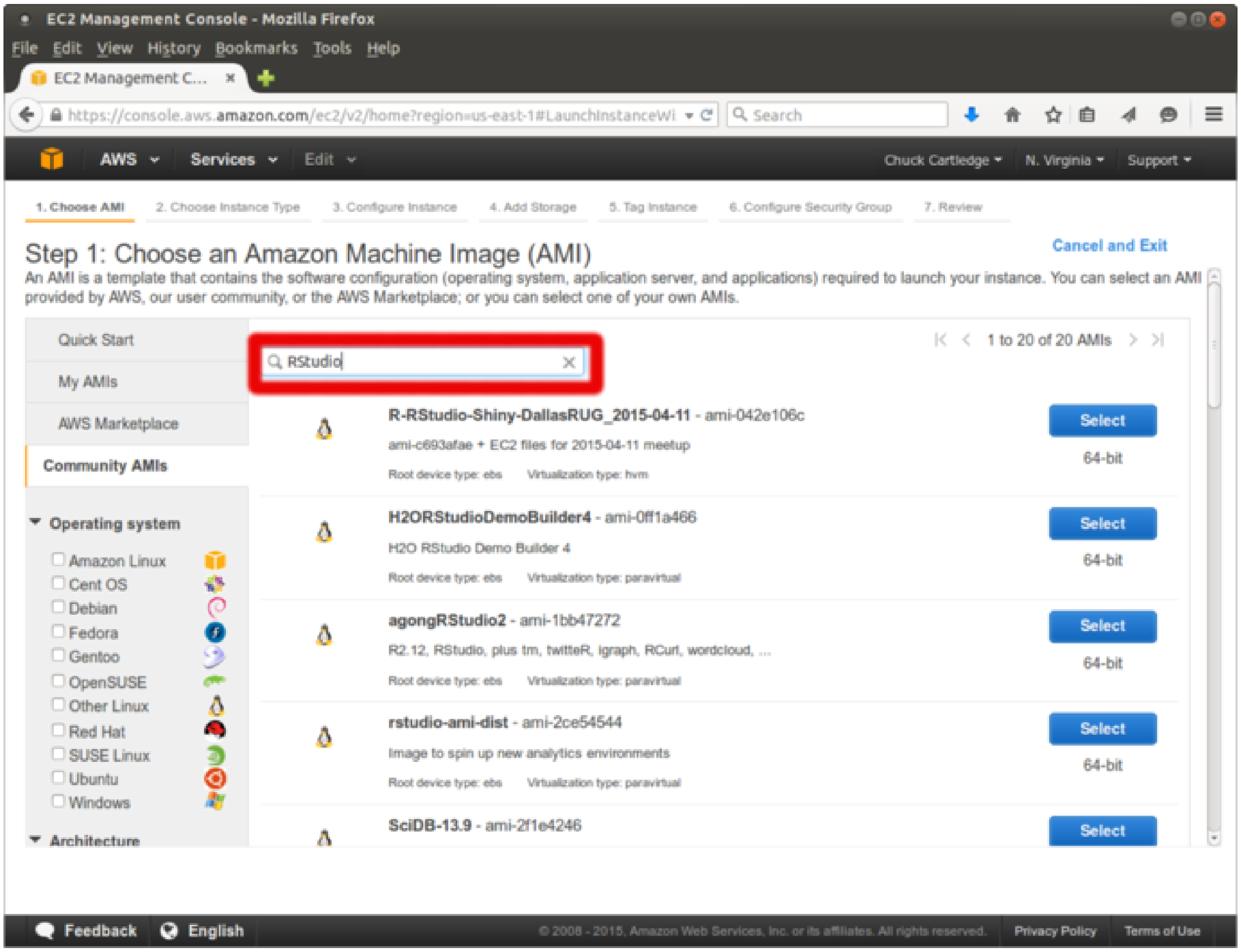

Figure 10: Step 5, search all AMIs tagged with RStudio.

5. Search for all AMIs tagged with RStudio (see Figure 10). An AMI is a combination of an operating system (OS) and possibly some collection of software tailored to perform a specific type of task. In this case we are interested in using the application RStudio, so we are searching for an AMI that was tagged by its creator as having RStudio installed. Normally an AMI creator will tag an AMI with a descriptive set of tags to allow it to be found by a keyword search. For instance, an AMI configured to support Hadoop , would likely be tagged with Hadoop, or HDFS (HaDoop File System).

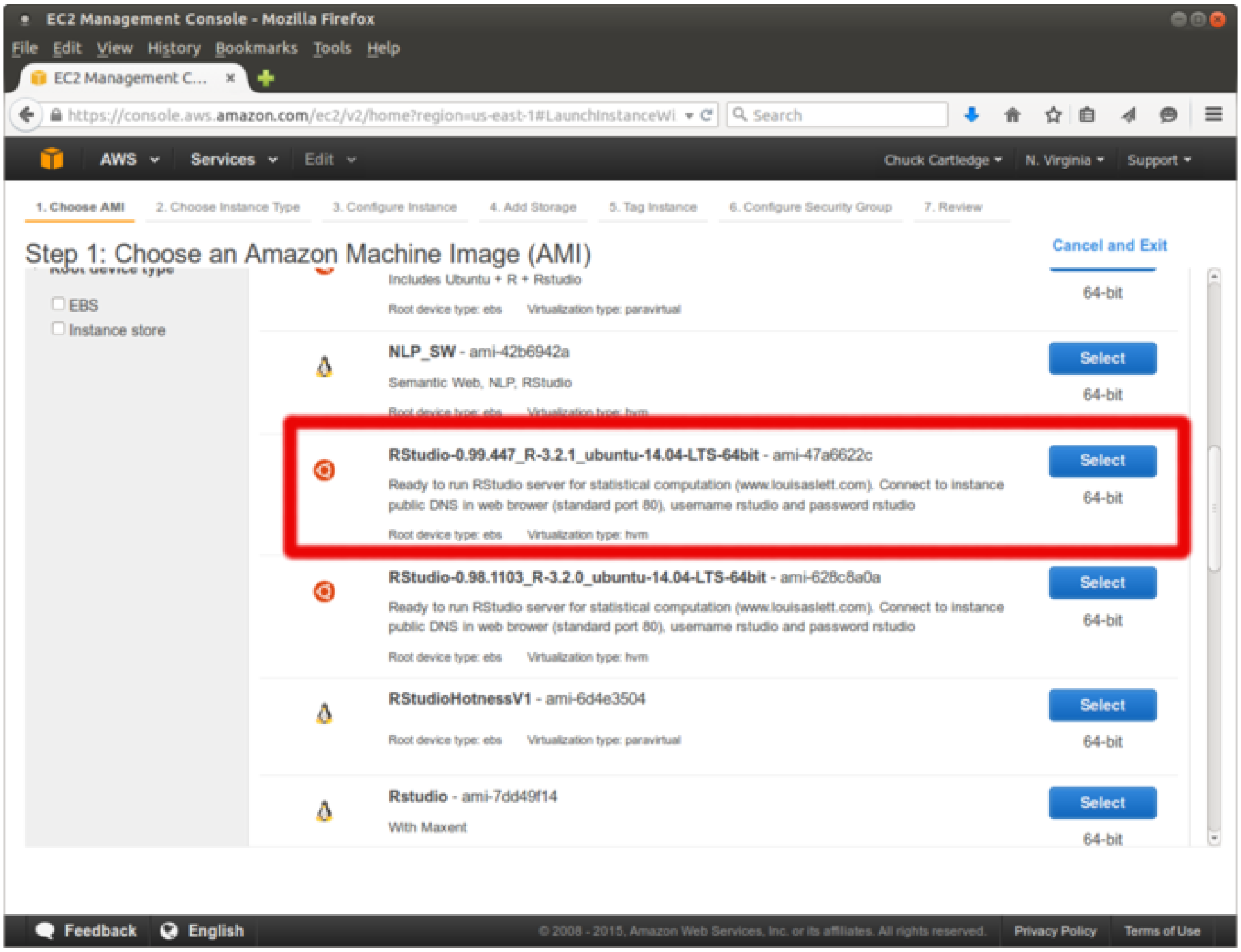

Figure 11: Step 6, select an AMI from Louis Aslett. There are virtually an innumerable number of AMIs available. We have selected this one because of the write-up and list of functions that are available [1].

6. Select an AMI that we've read some things about (see Figure 11).

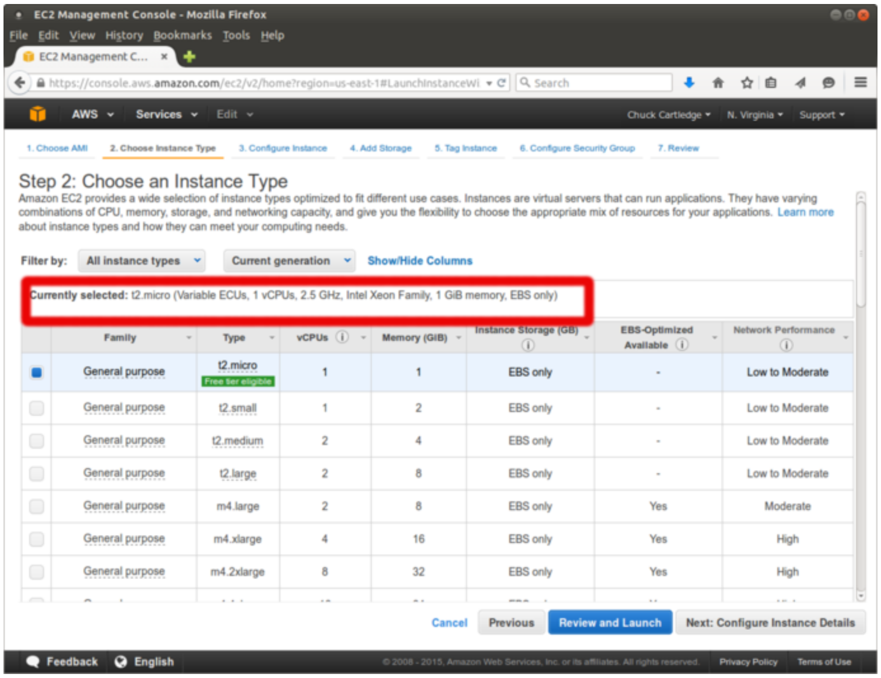

Figure 12: Step 7, verify that the machine has enough "juice."

7. The CPU is limited, but it is free (see Figure 12). We could use the instance as it is, but we want to change how we can access the machine (see Figure 13).

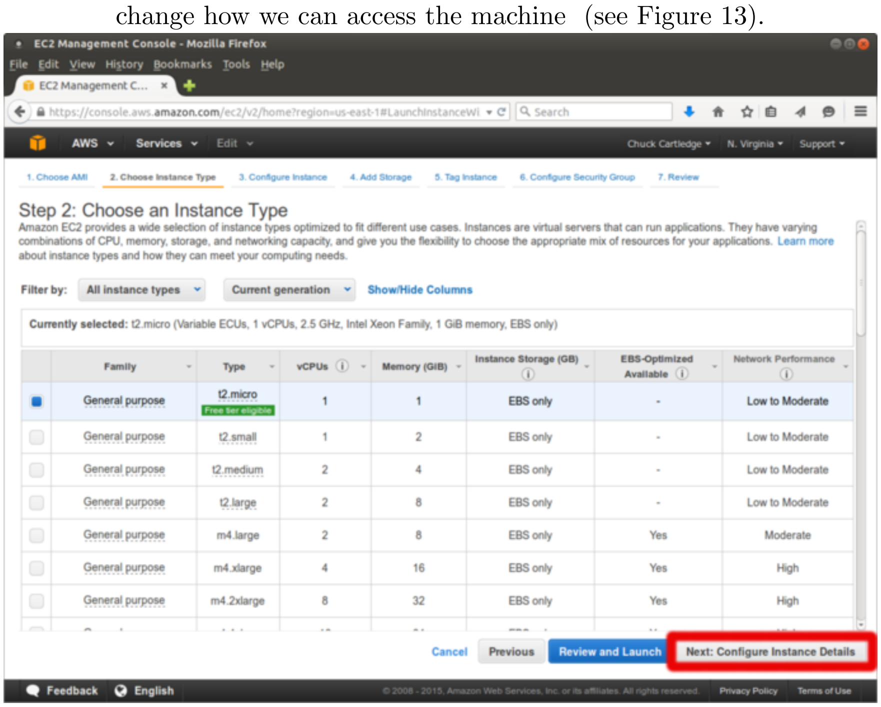

Figure 13: Change the instance's configuration.

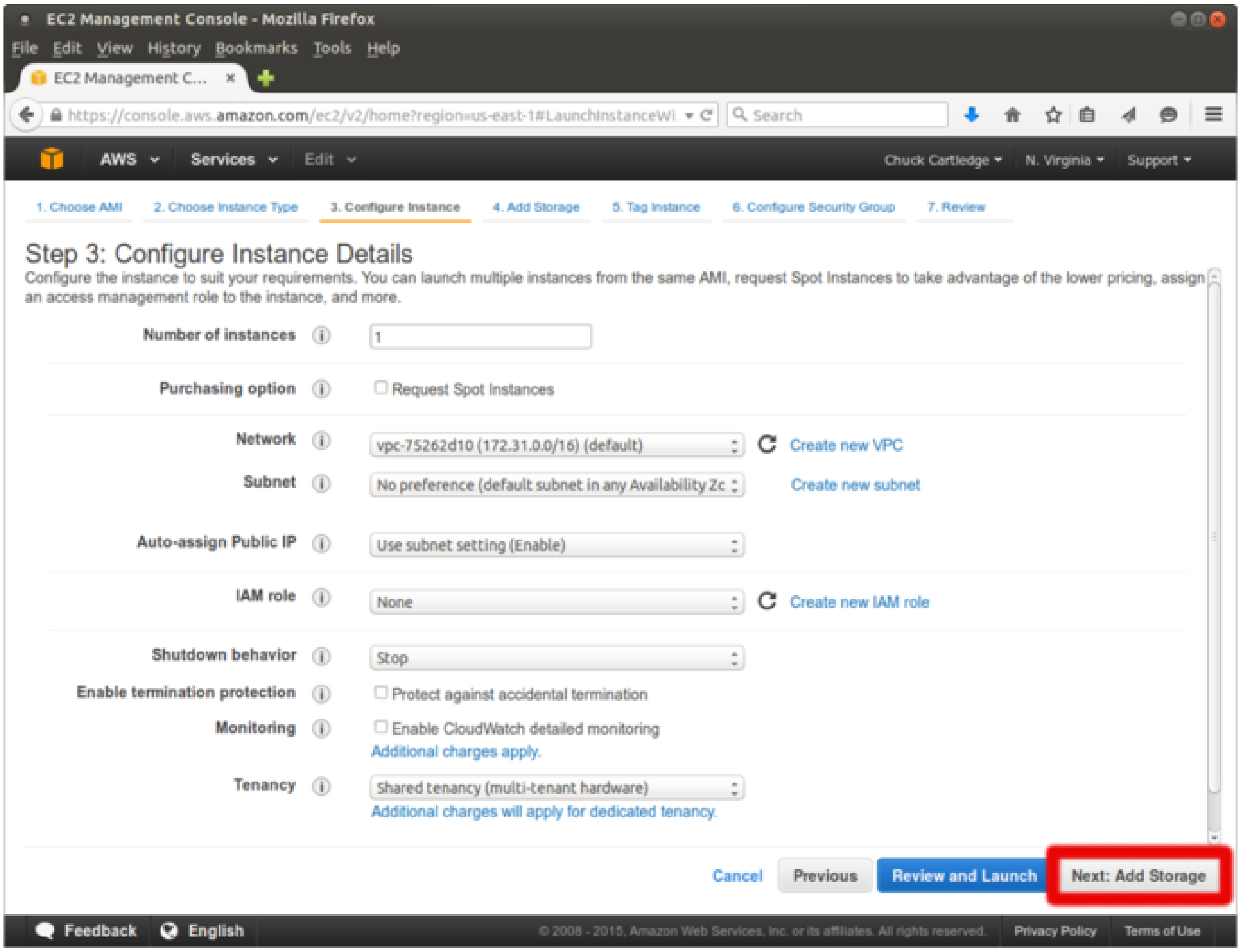

Figure 14: Step 8, accept the basic machine settings.

8. Accept the basic machine settings (see Figure 14).

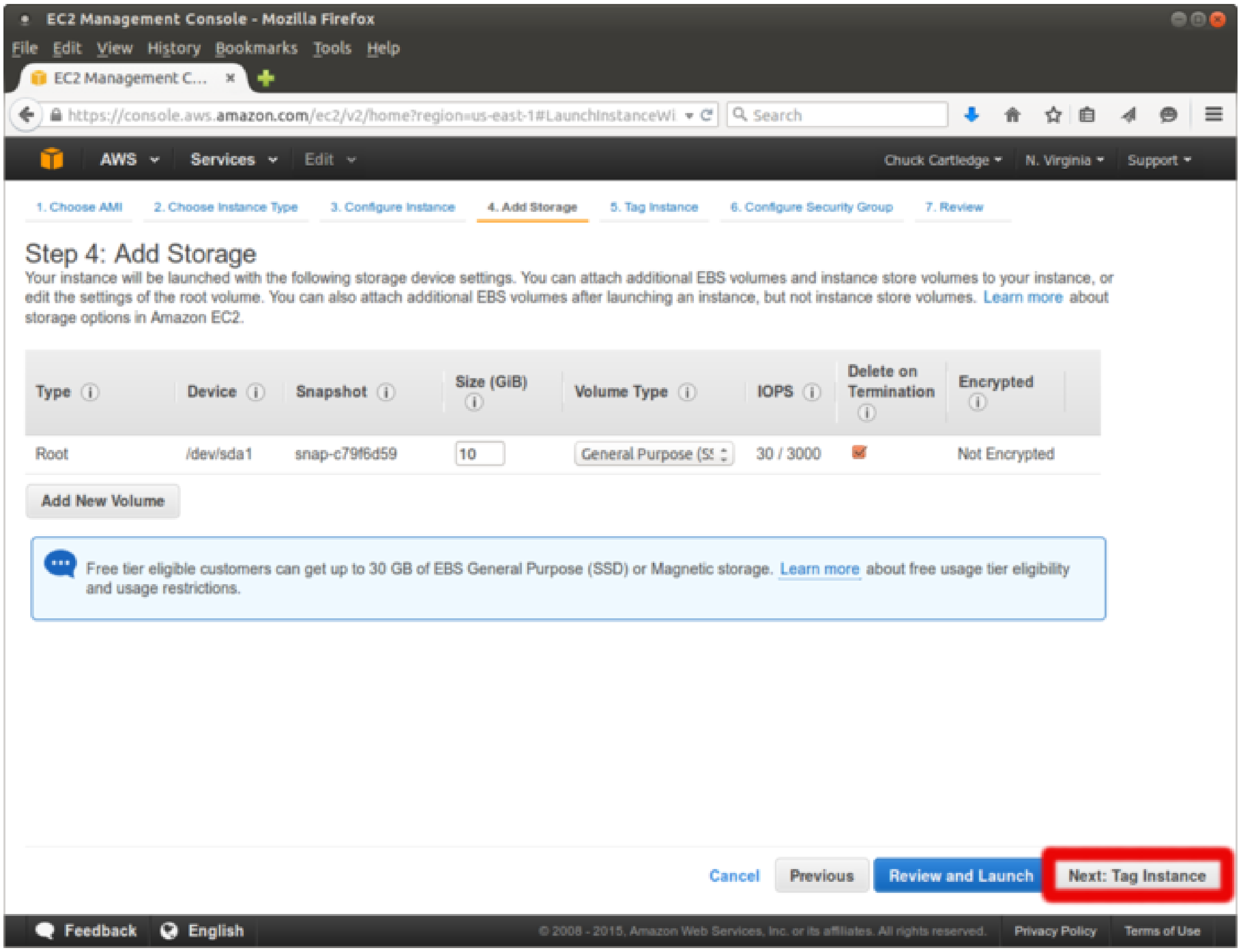

Figure 15: Step 9, accept the default storage settings.

9. Accept the default storage settings (see Figure 15).

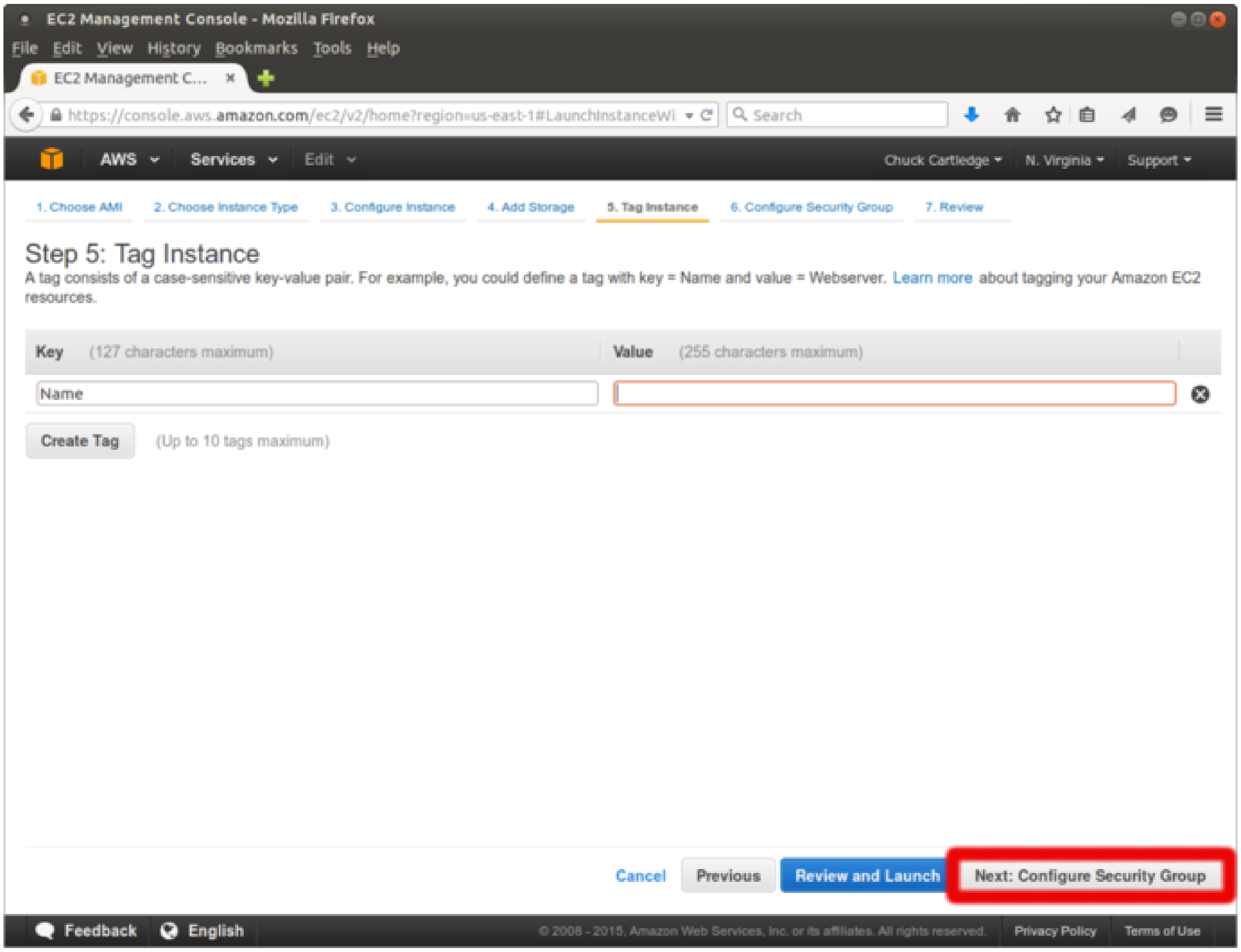

Figure 16: Step 10, accept the tag settings.

10. Accept the default tag settings (see Figure 16).

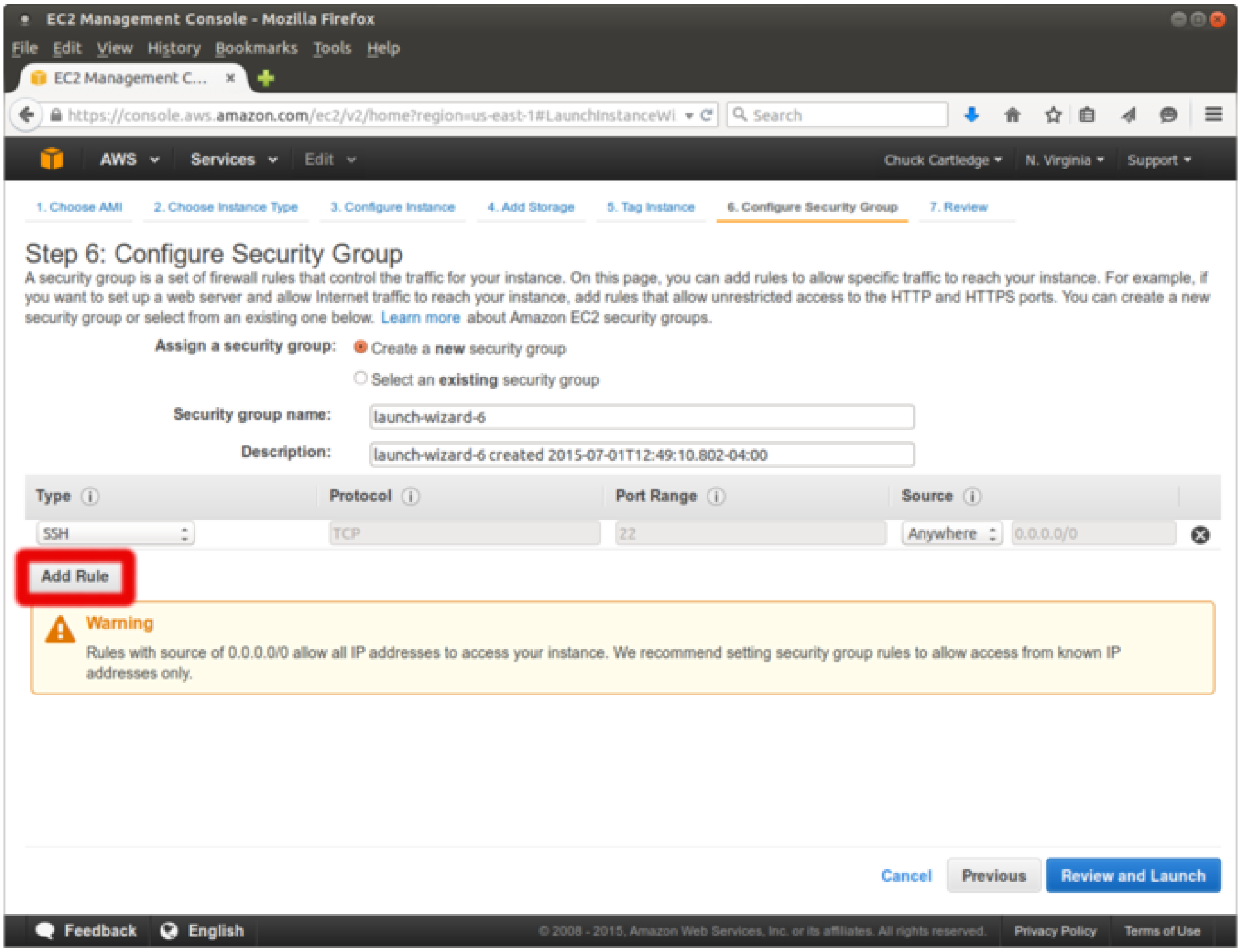

Figure 17: Step 11, tell AWS we want to add a rule to their firewall.

11. Change the Security Group settings to we can access the instance using a browser (see Figure 17).

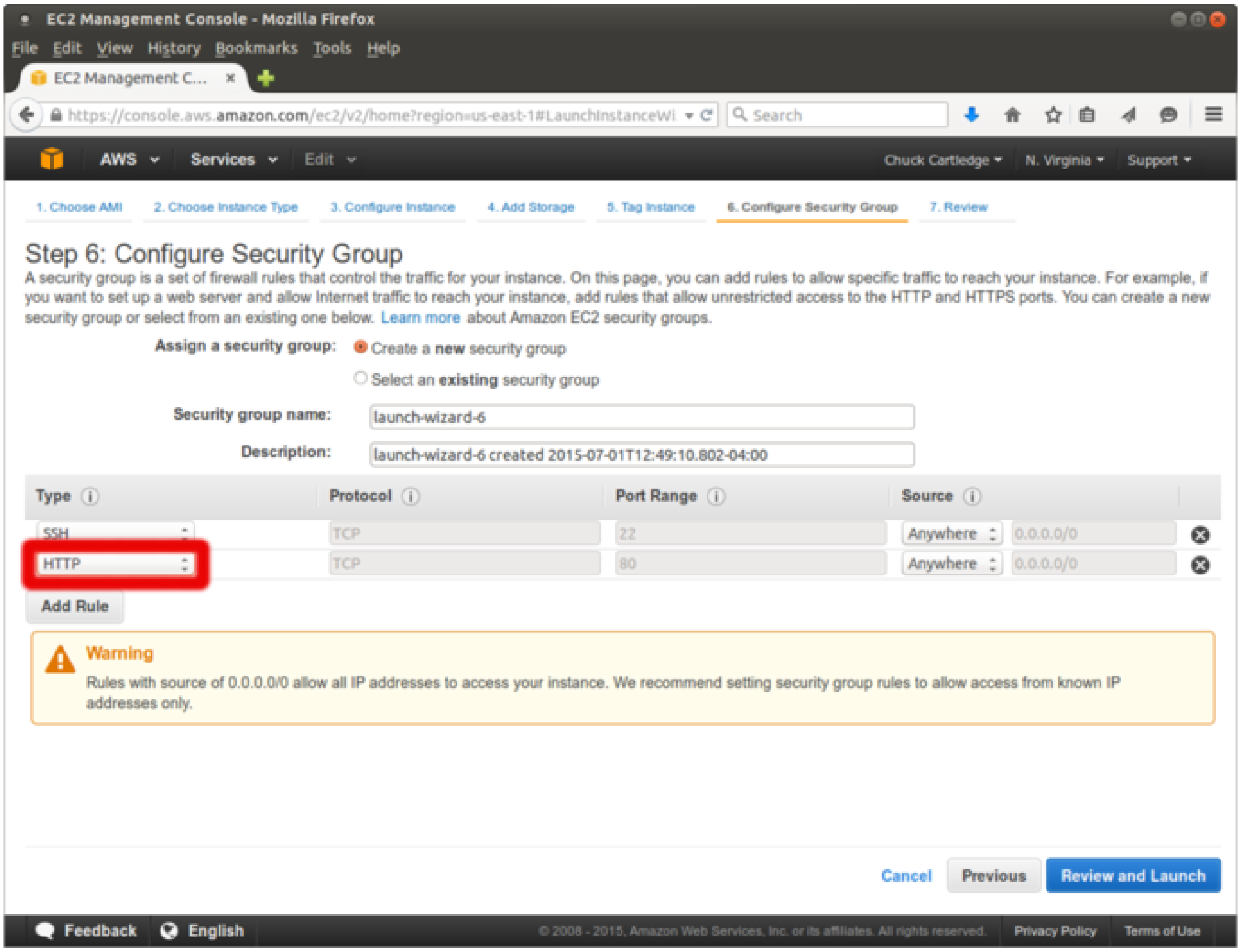

Figure 18: Step 12, configure the AWS firewall to permit HTTP connections to our instance.

12. Add the default HTTP rule (see Figure 18). This will give us two ways to get to the instance: ssh (the default), and HTTP (from our browser). Other protocols are available based on how the AMI is configured. In this case, the AMI has the RStudio server configured for HTTP connections, so we will choose HTTP.

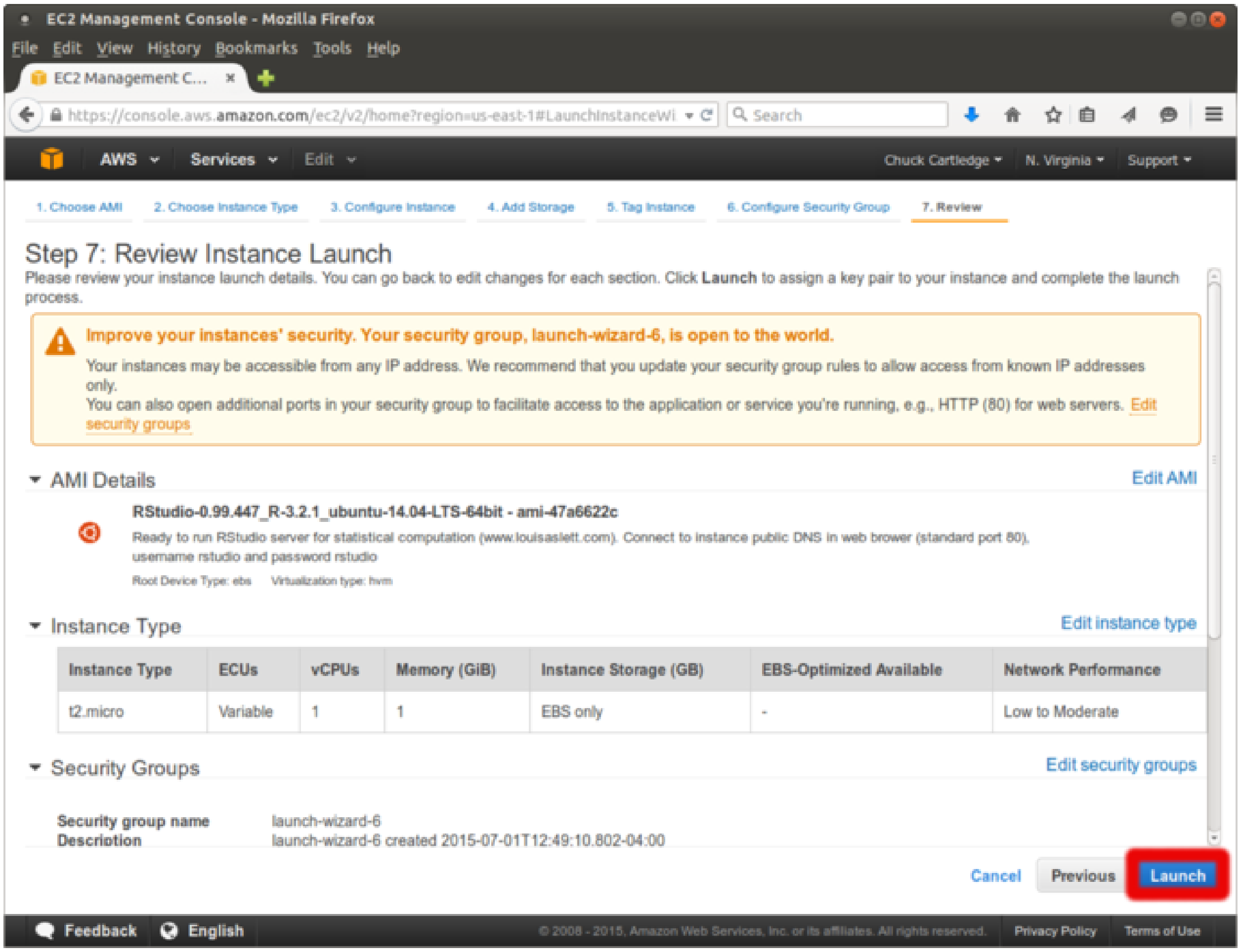

Figure 19: Step 13, verify that things are the way want them.

13. Verify that things are the way we want. (see Figure 19).

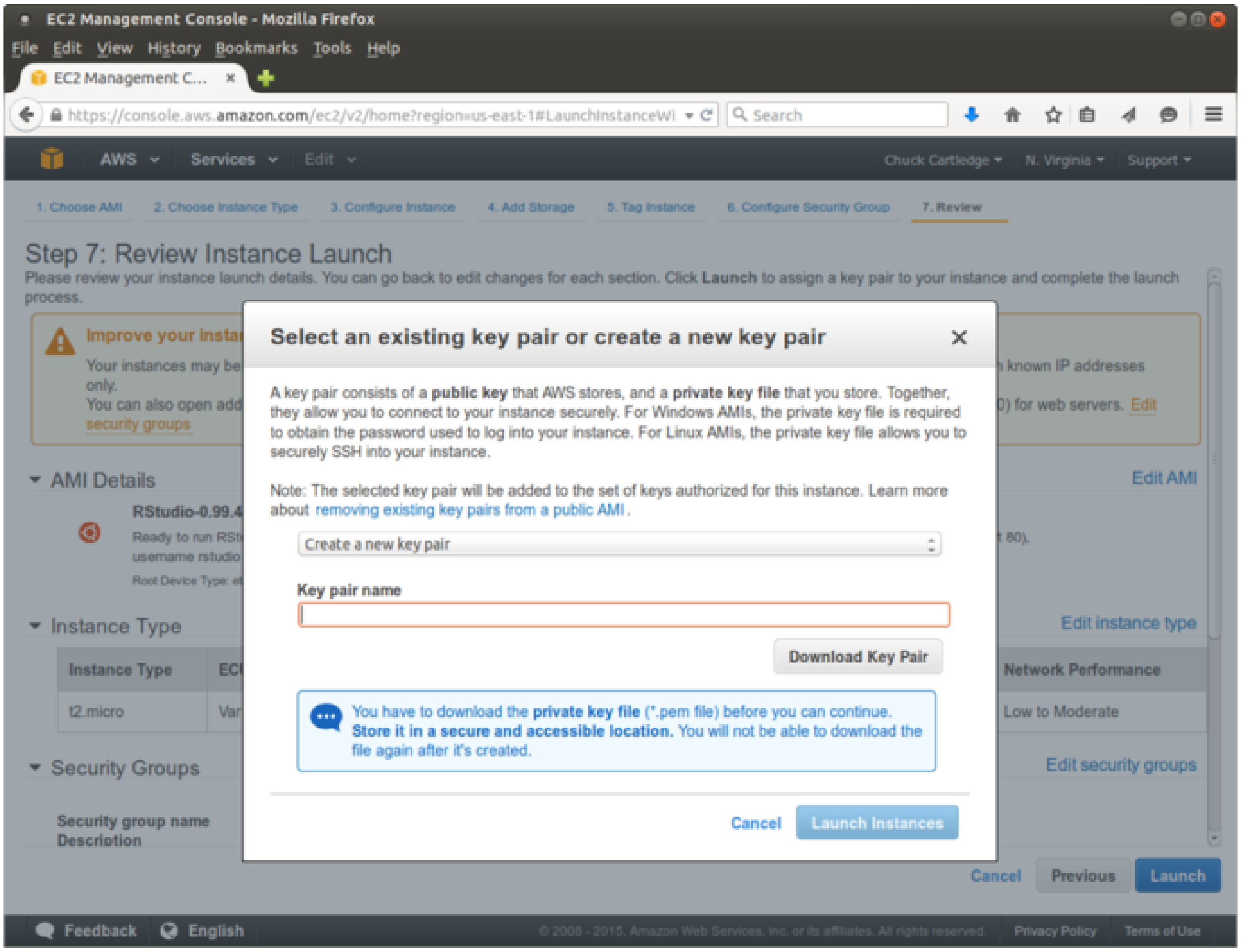

Figure 20: Step 14, a ssh connection is required (and recommended), but may never be used.

14. Create an SSH key privacy enhanced email (PEM) pair to allow ssh connectivity (see Figure 20). The key file name is important and may be used later (see Figure 21).

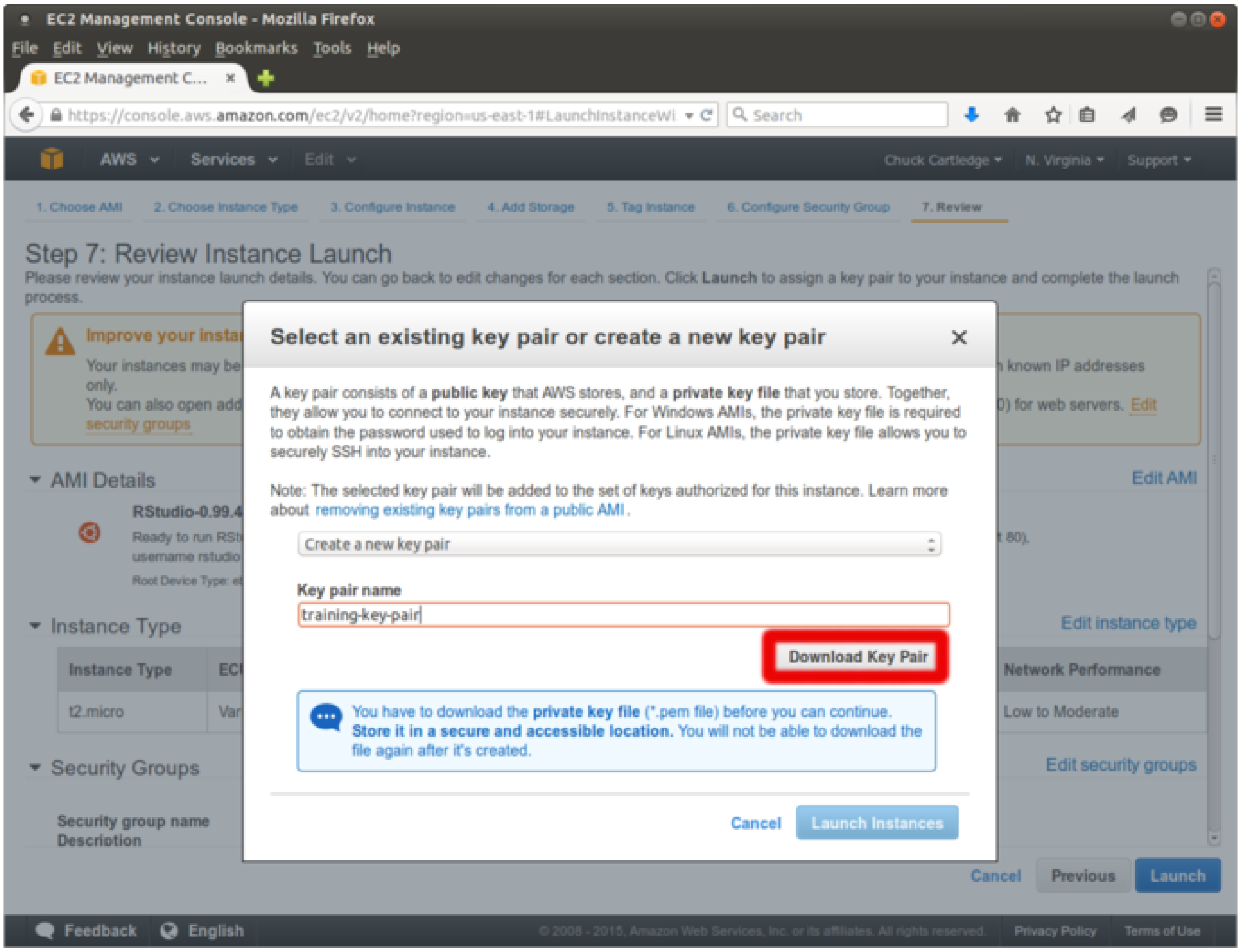

Figure 21: A filled out ssh key pair screen.

15. After the Key Pair is downloaded, it must be put in the correct directory/folder for the ssh program in your operating system to find it (see Figure 21). On *nix distributions, this is ~/.ssh. Windows distributions will have a different location. In addition, it may be necessary to change the protection modes of the downloaded file. A PEM file is used by the secure sockets layer (SSL) protocol. The PEM file may contain a variety of SSL related certificates (public key, private key, root certificates, entire SSL chains, etc.) and is used by AWS EC2 to authenticate your user.

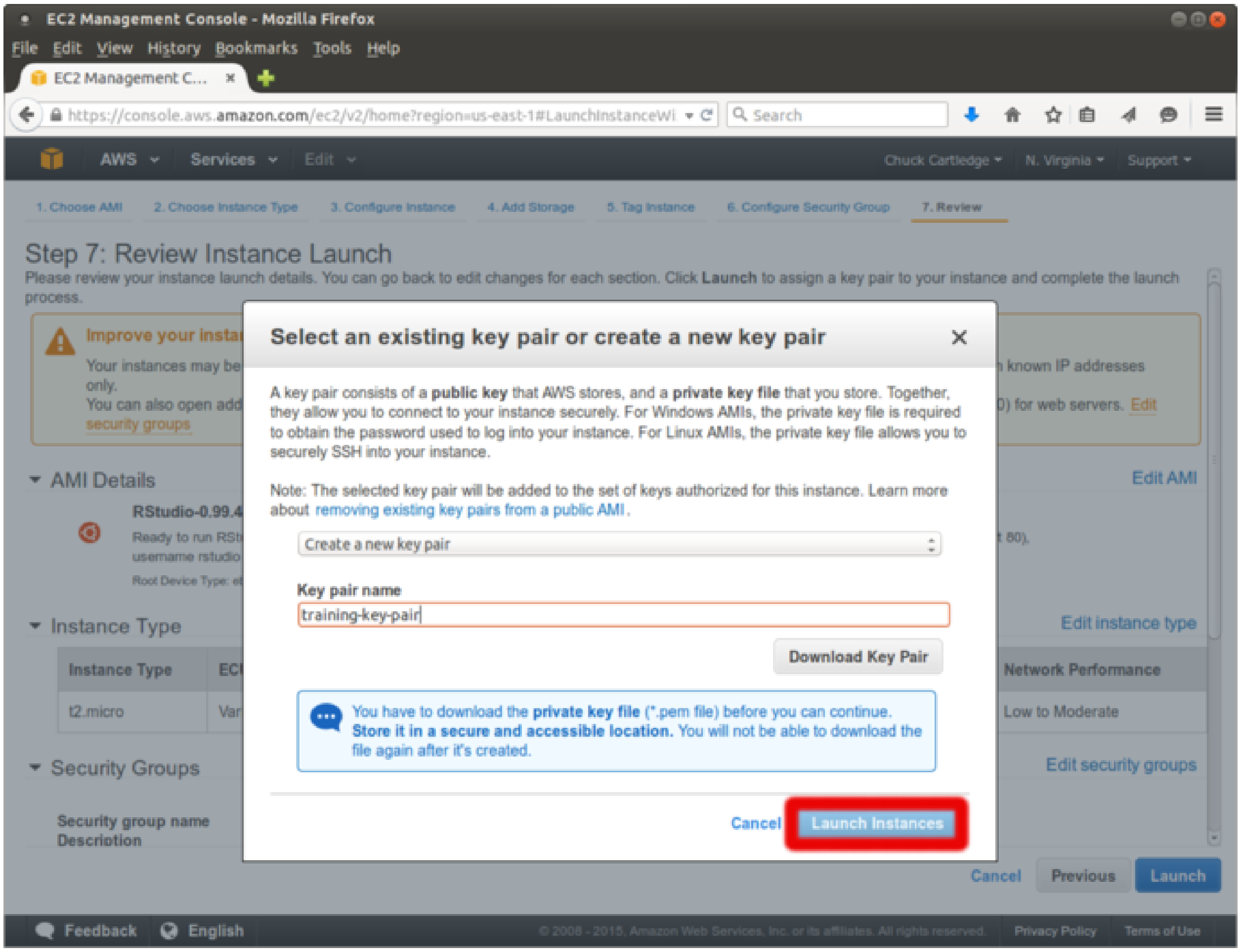

Figure 22: Step 16, launching the instance.

16. Launch the instance (see Figure 22).

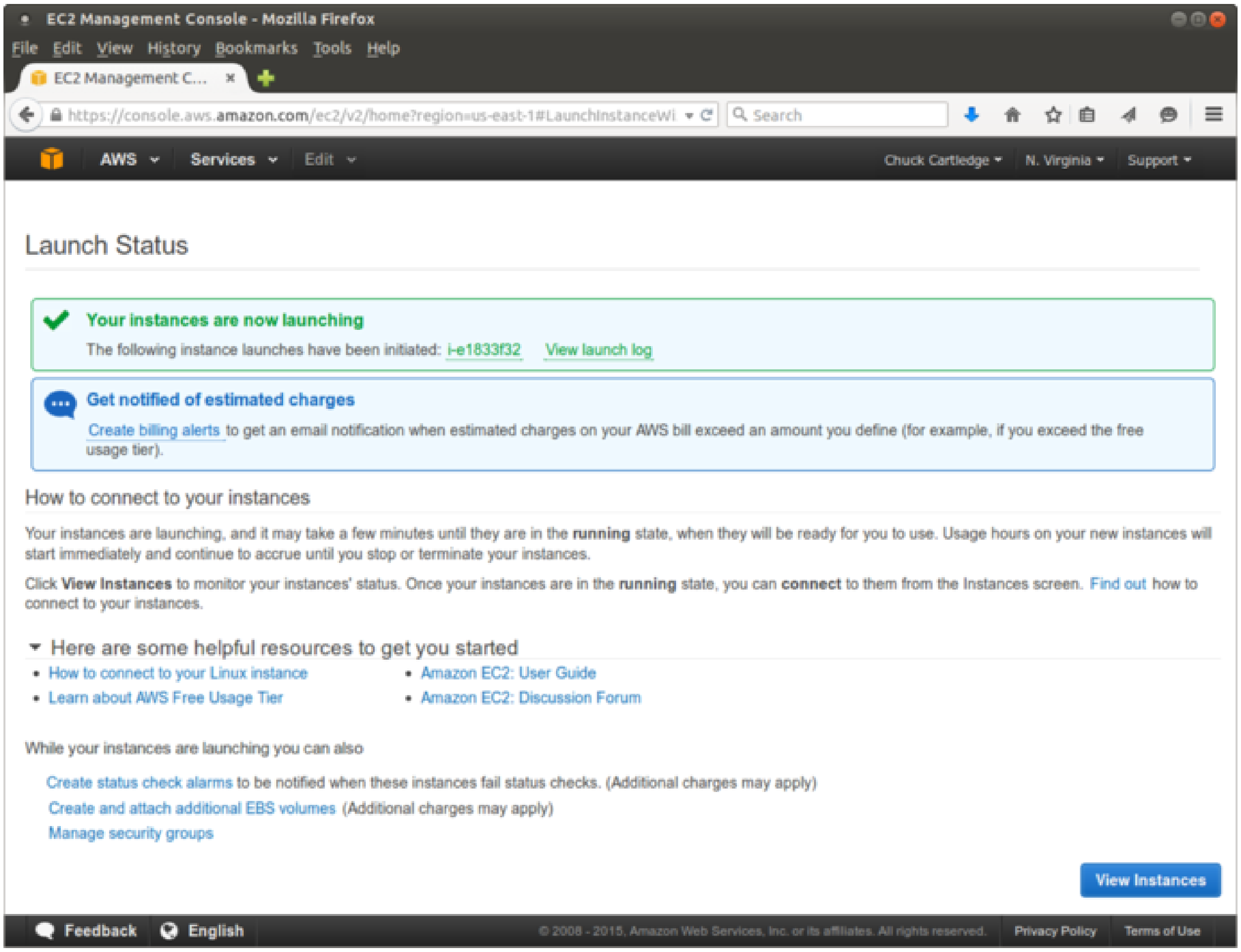

Figure 23: Step 17, viewing confirmation that the instance has been launched.

17. View conformation that the instance has started (see Figure 23).

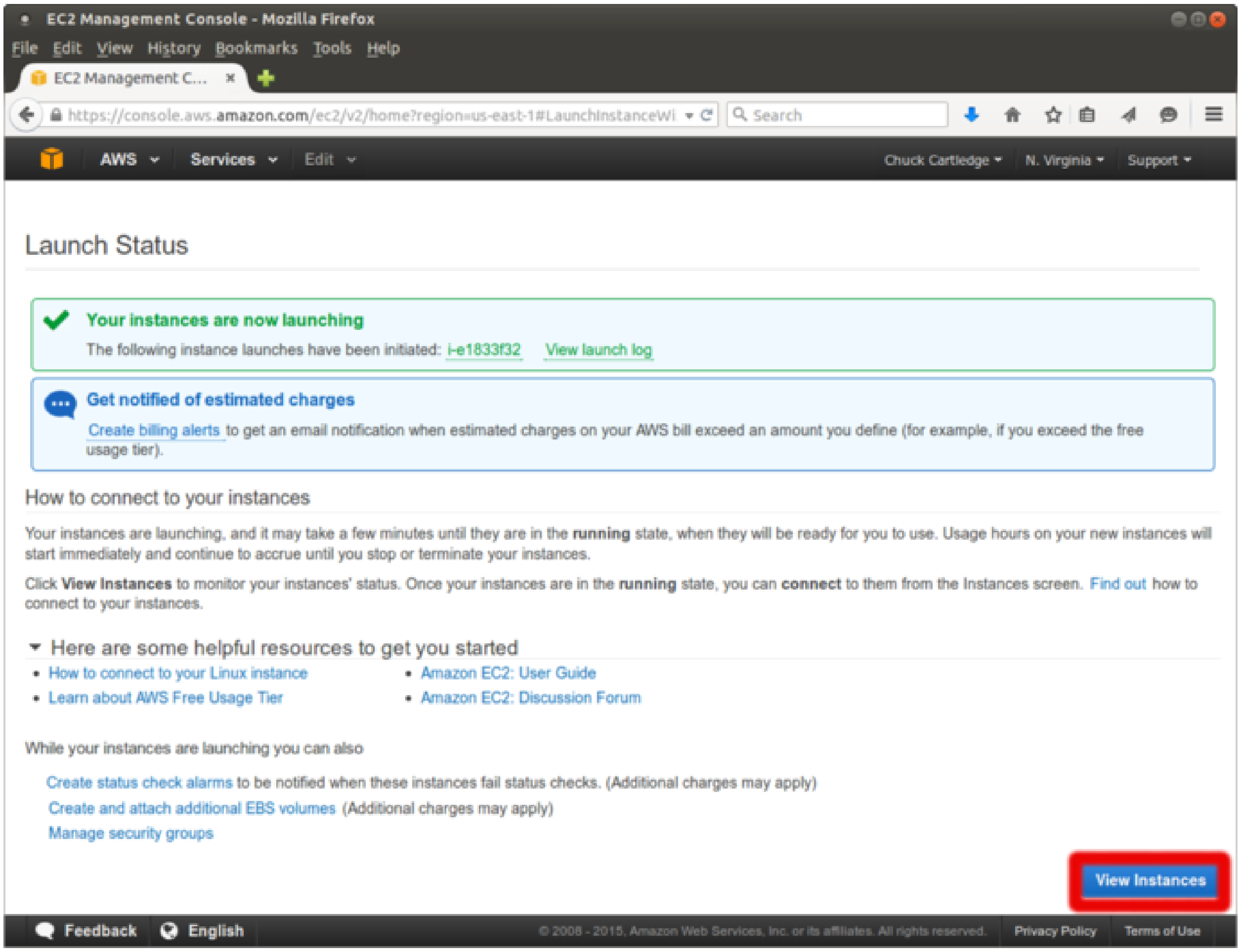

Figure 24: Step 18, view detailed information about the running instance.

18. View detailed information about the running instance (see Figure 24).

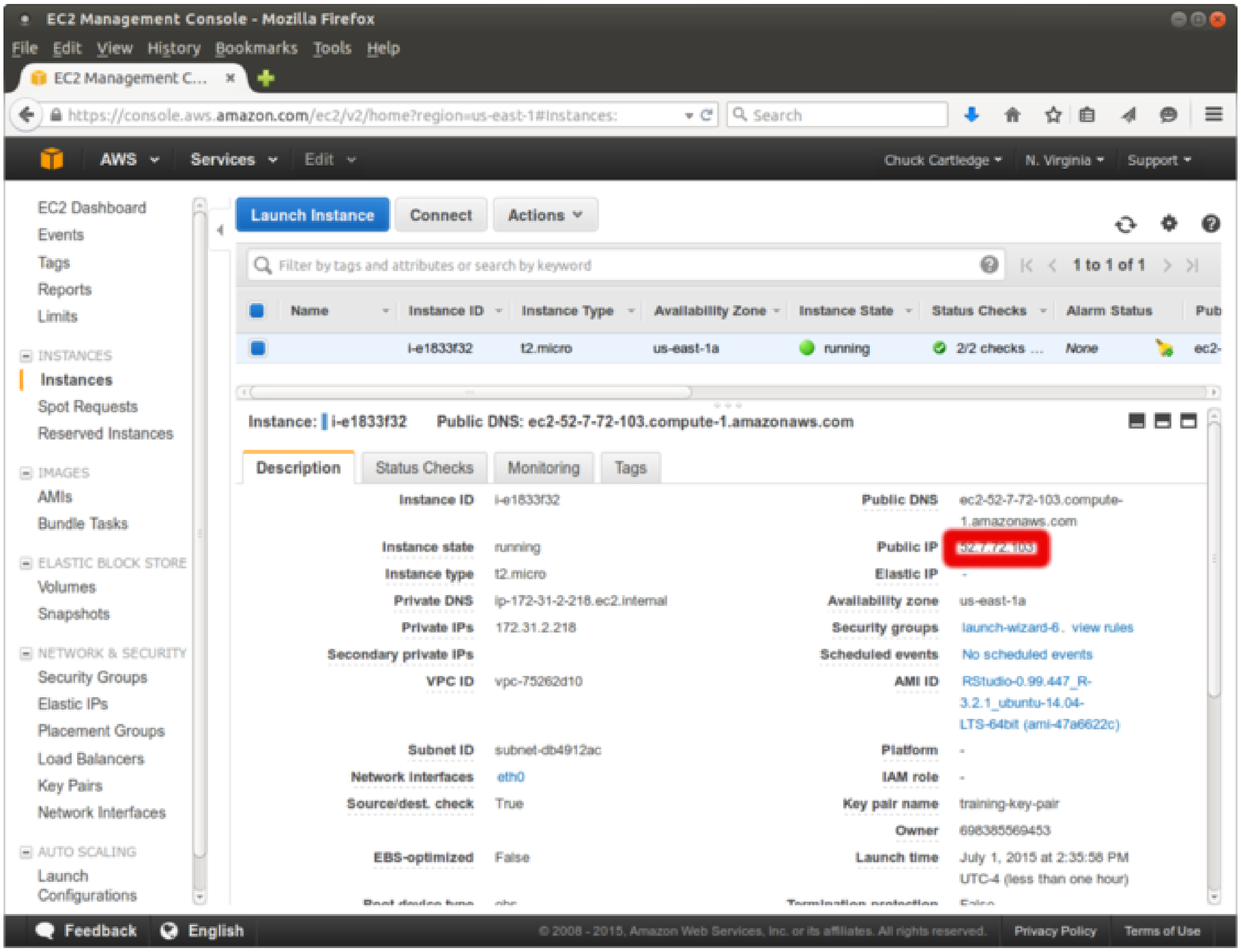

Figure 25: Step 19, detailed information about the running instance. The public IP address is our portal to the instance.

19. Detailed information about the running instance (see Figure 25).

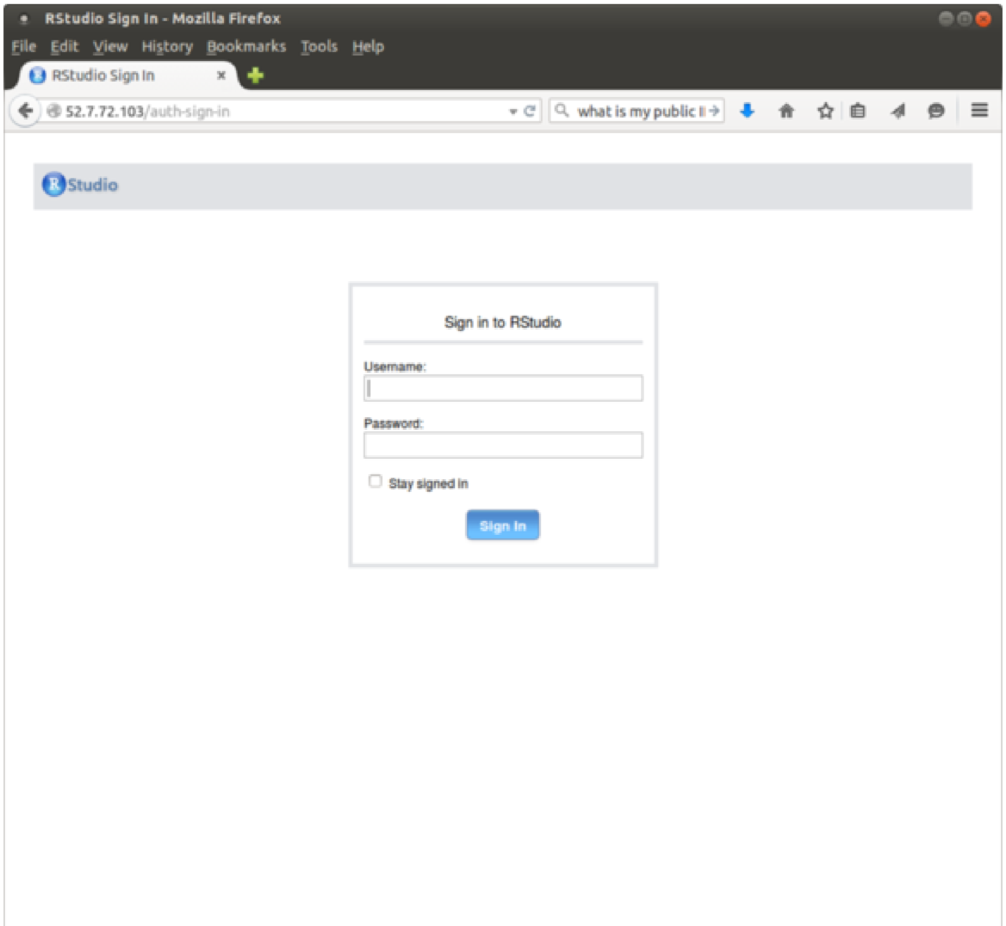

Figure 26: Step 20, viewing the RStudio login prompt. From Aselett's web page we know that the login is: rstudio and password is: rstudio

20. Access the running instance by using the public IP address (see Figure 26).

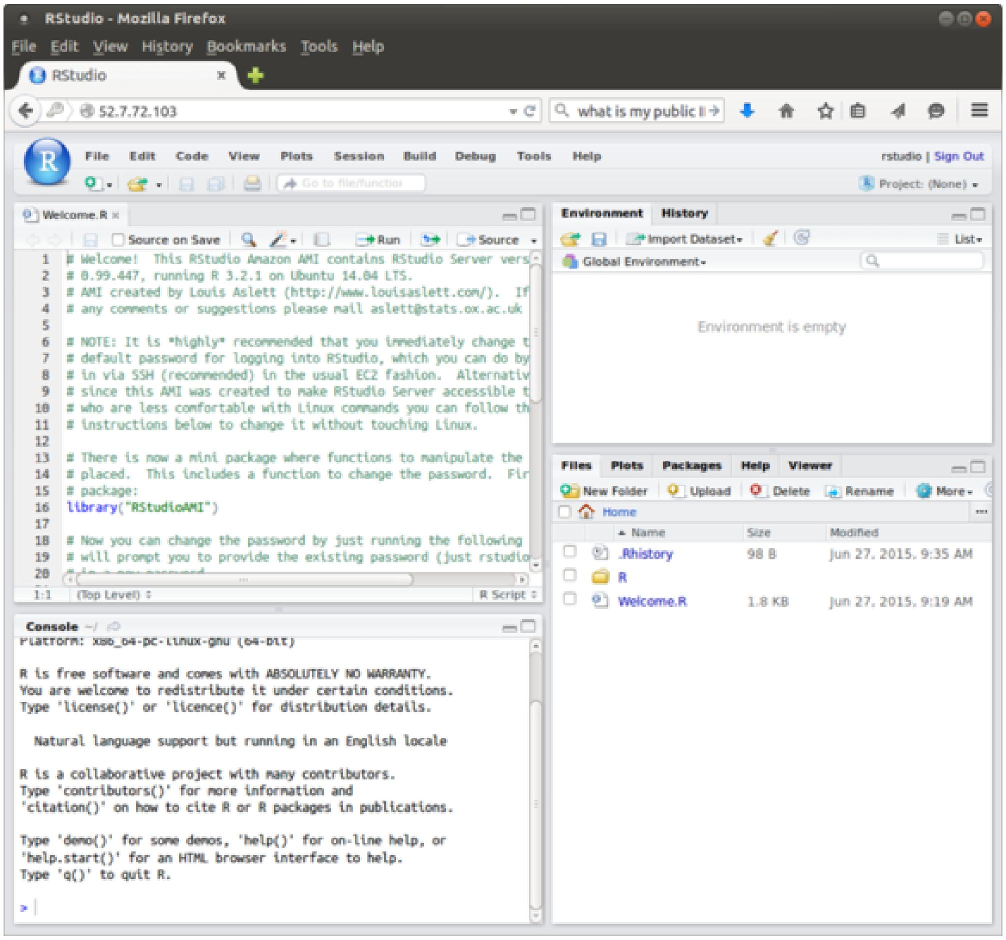

Figure 27: Step 21, RStudio is now available.

21. Use RStudio as you normally would (see Figure 27).

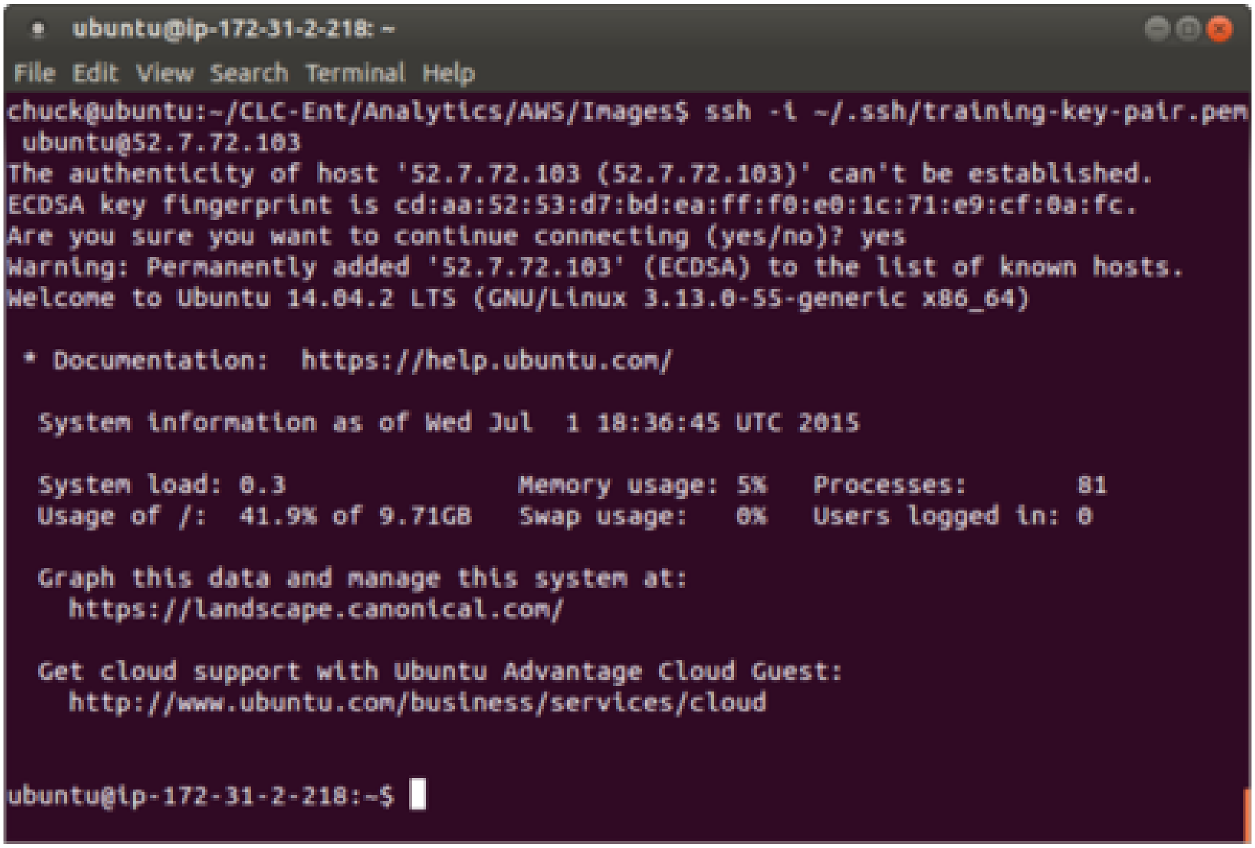

Figure 28: Step 22, use the ssh pair file, the user name, and public IP address to access the instance.

22. Use ssh to access the running instance (see Figure 28). By using ssh you can also scp files to and from the running instance. The general from of the scp command to copy files to the instance is:

```
scp -i identity_file file_to_send user@host:destination
```

In the specific case shown in the figure, the scp command to copy a local data file to the instance would be (line is broken for printing clarity):

```
scp -i ~/.ssh/training-key-pair.pem /tmp/dataFile.csv
      ubuntu@52.7.72.103:/tmp
```

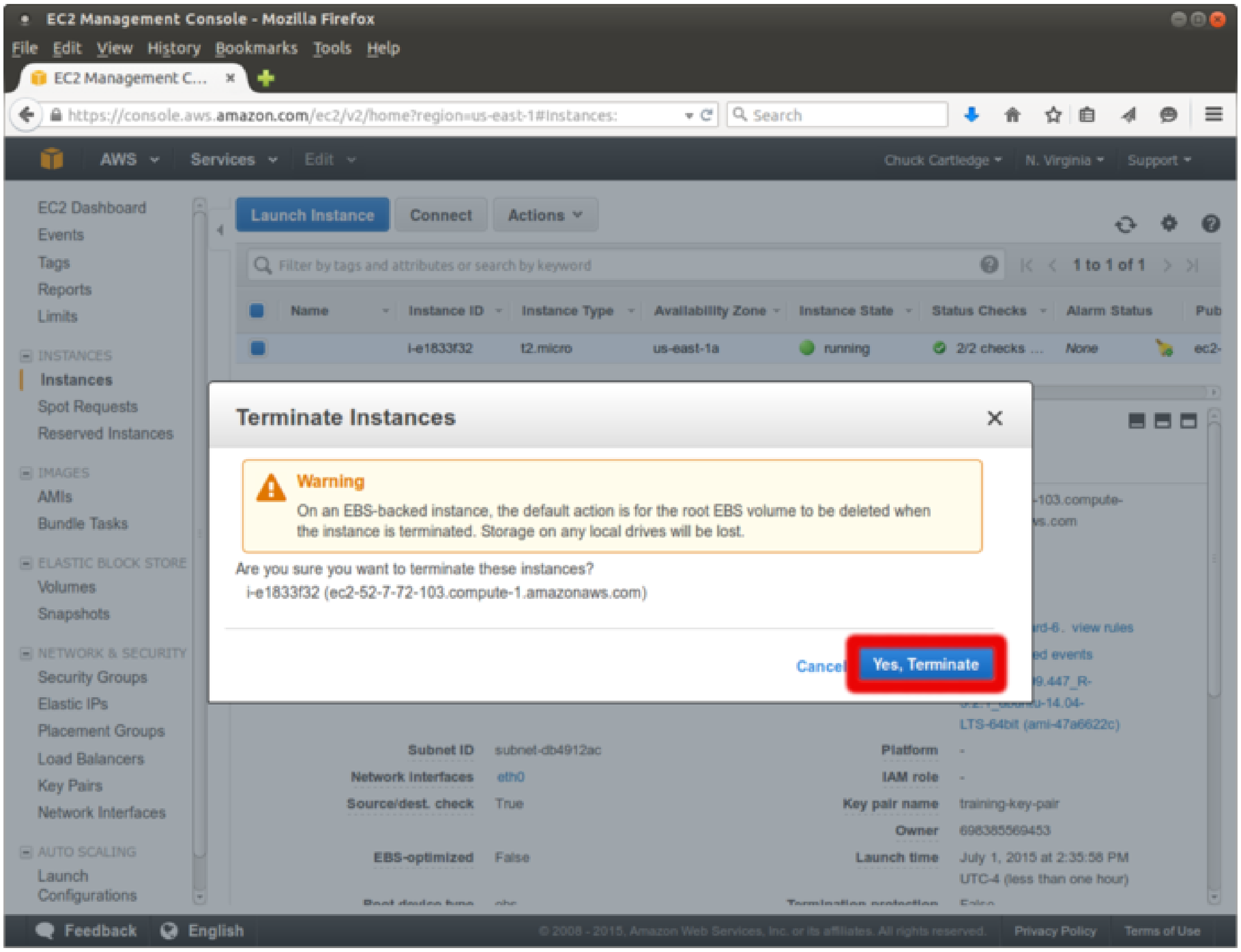

Figure 29: Step 23, terminate the running instance.

23. You can terminate the running instance when you are finished by selecting Actions ⇒ Instance State ⇒ Terminate from the AWS EC2 Management console. You will get a reminder that all data on the instance will be lost (see Figure 29).

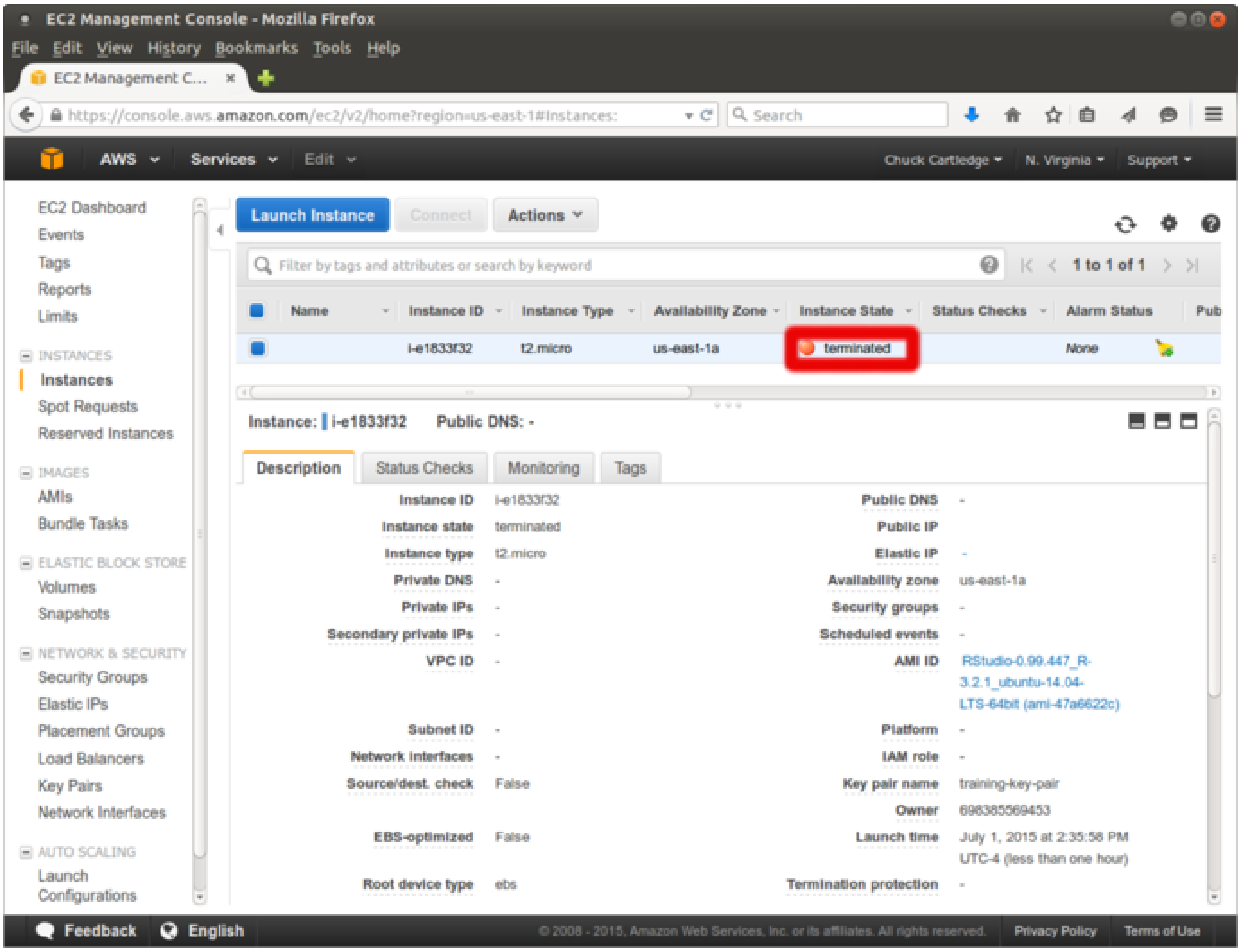

Figure 30: Step 24, observe that the instance has been terminated. The instance will remain on the display for some 10s of minutes, and then will disappear entirely.

24. The status will eventually go to Terminated

6 Ways to tailor your instance

Remembering that the virtual machine can be treated like a "real" machine, almost anything you can do to a real one; you can do the virtual one.

6.1 Adding more programs

You install programs just like you would on a real computer. Naturally, your user has to have permission to add programs. Sometimes this can be a challenge; but depending on how your original AMI was configured, it can be as easy as copying executable files to the correct location.

6.2 Adding more data files

Perhaps the easiest was to add data files is to use scp and copy them from their source to their destination on the virtual machine. *nix has several scp GUI based programs, and the granddaddy of them all, the scp command. Windows has several scp programs. When using scp, you have to use the correct pair file that you created when you created launched the instance, and you have to be the correct user on the virtual machine.

6.3 Saving your tailored instance

After you have modified an existing AMI, you can save an image off for your self. The image will be the original AMI plus any additional files you may have uploaded. (You could also delete files if you wanted.)

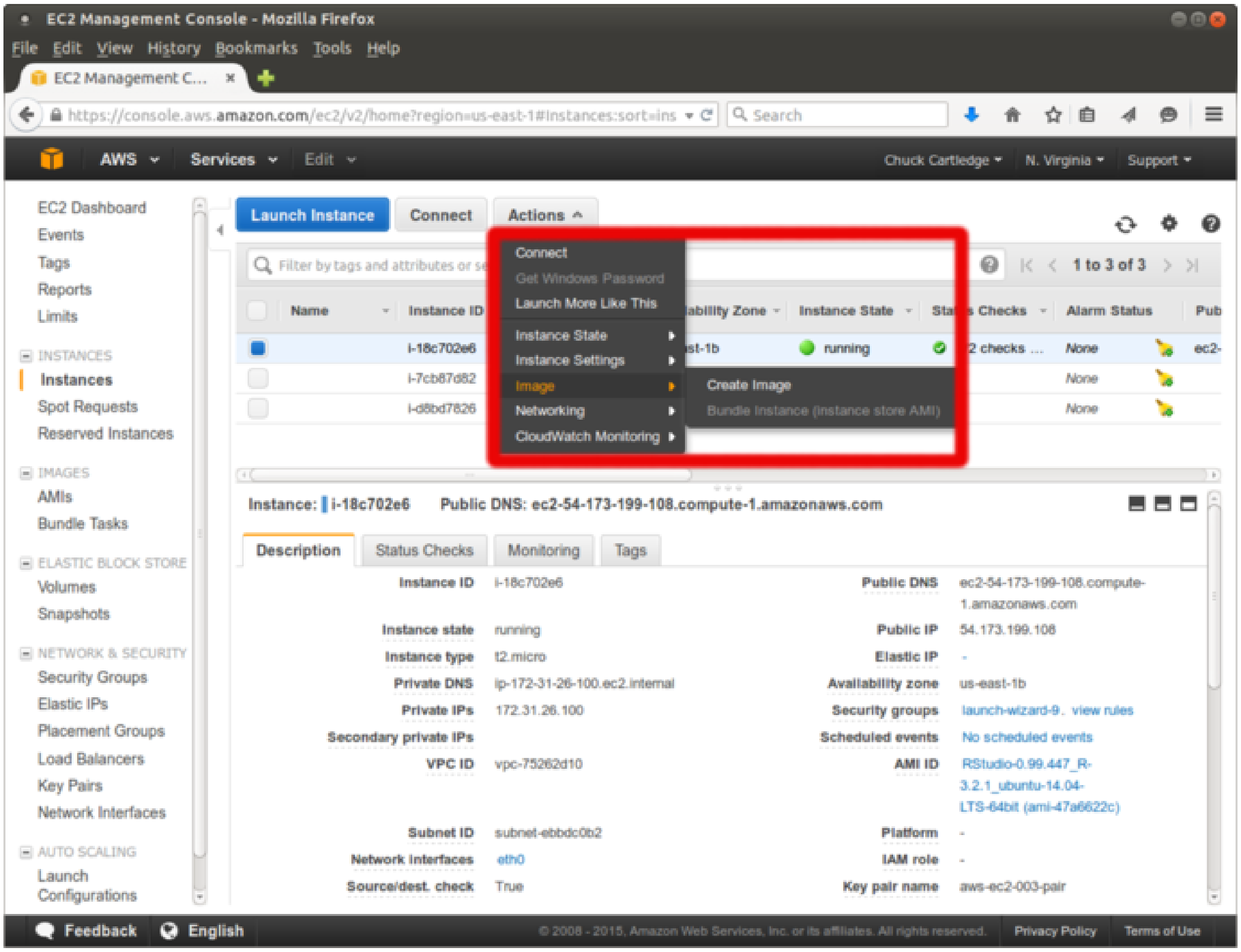

Figure 31: Step 1, start creating an image.

To create an image:

1. On the EC2 Management Console, Instance display, navigate Actions ⇒ Image ⇒ Create Image (see Figure 31).

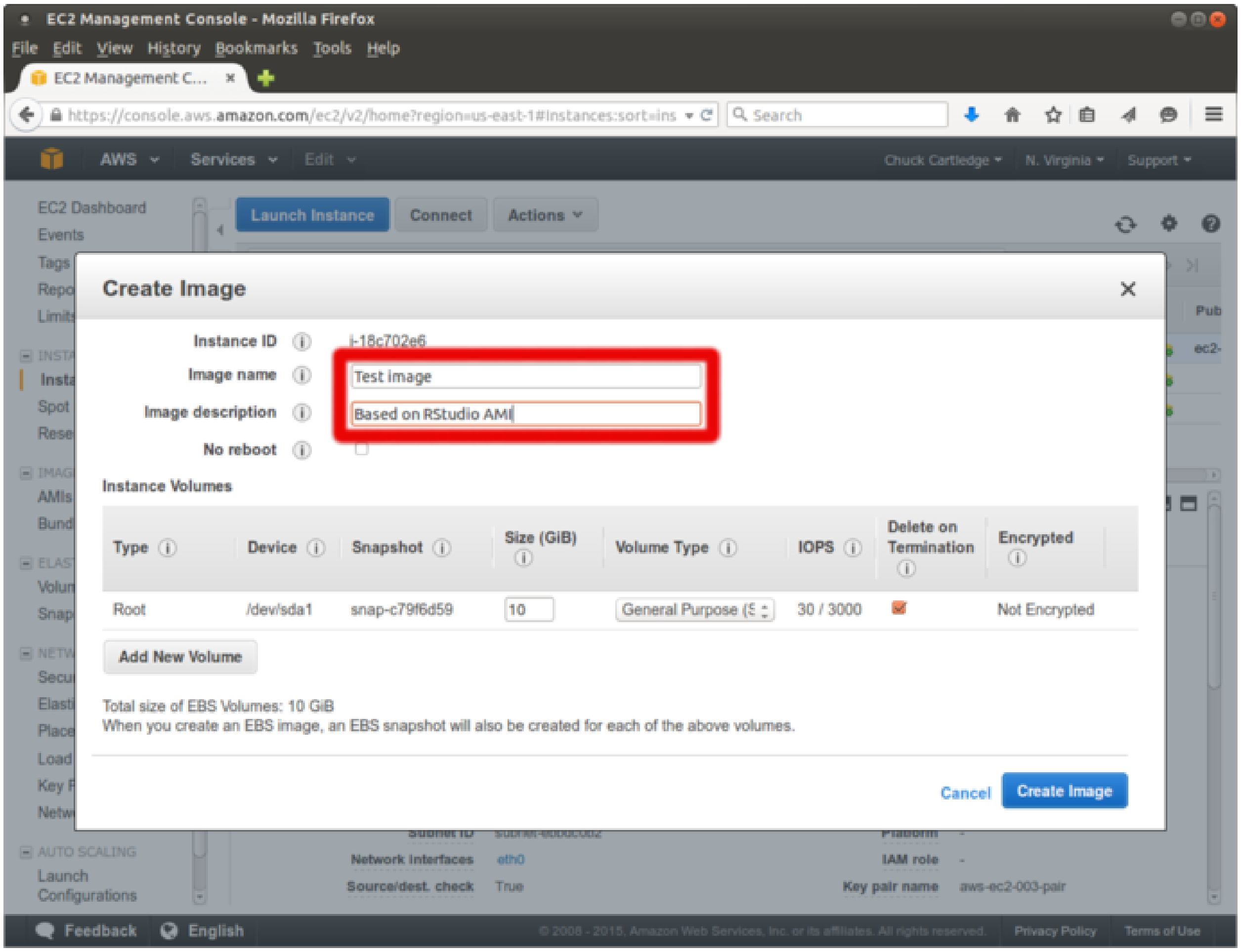

Figure 32: Step 2, name the image.

2. Name the image something meaningful and descriptive (see Figure 32).

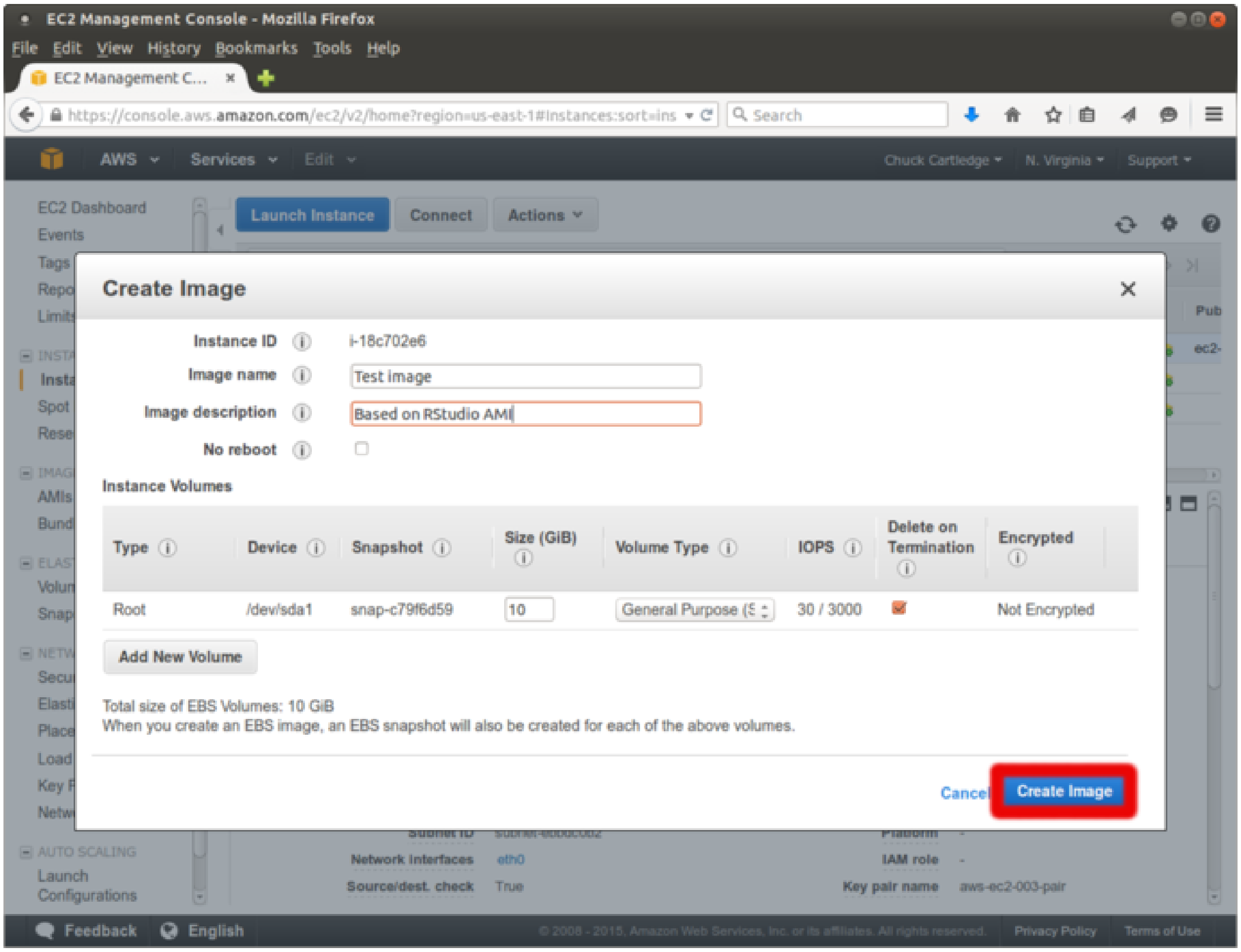

Figure 33: Step 3, create the image.

3. Tell AWS to create the image (see Figure 33).

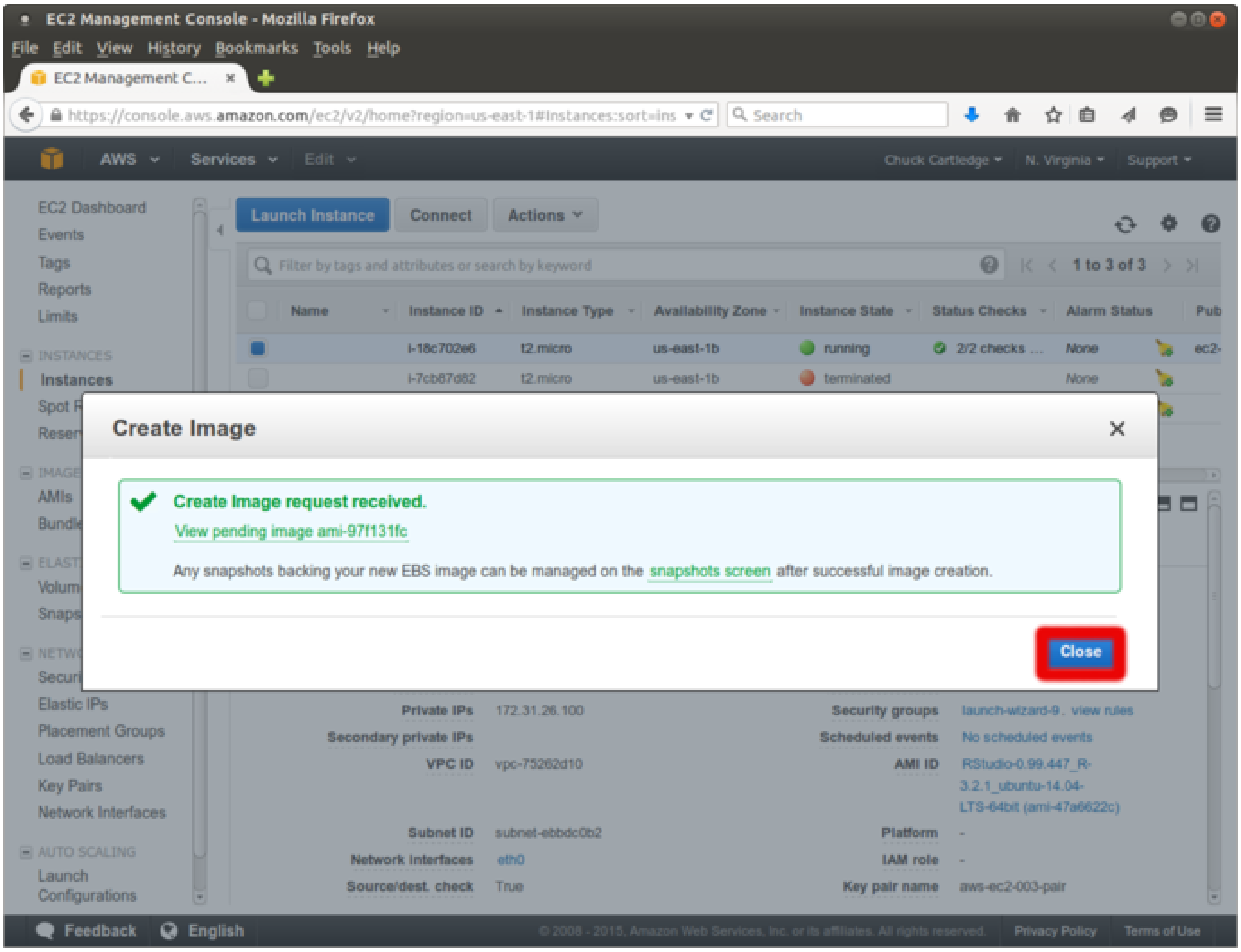

Figure 34: Step 4, save the image.

4. Tell AWS to save the image (see Figure 34).

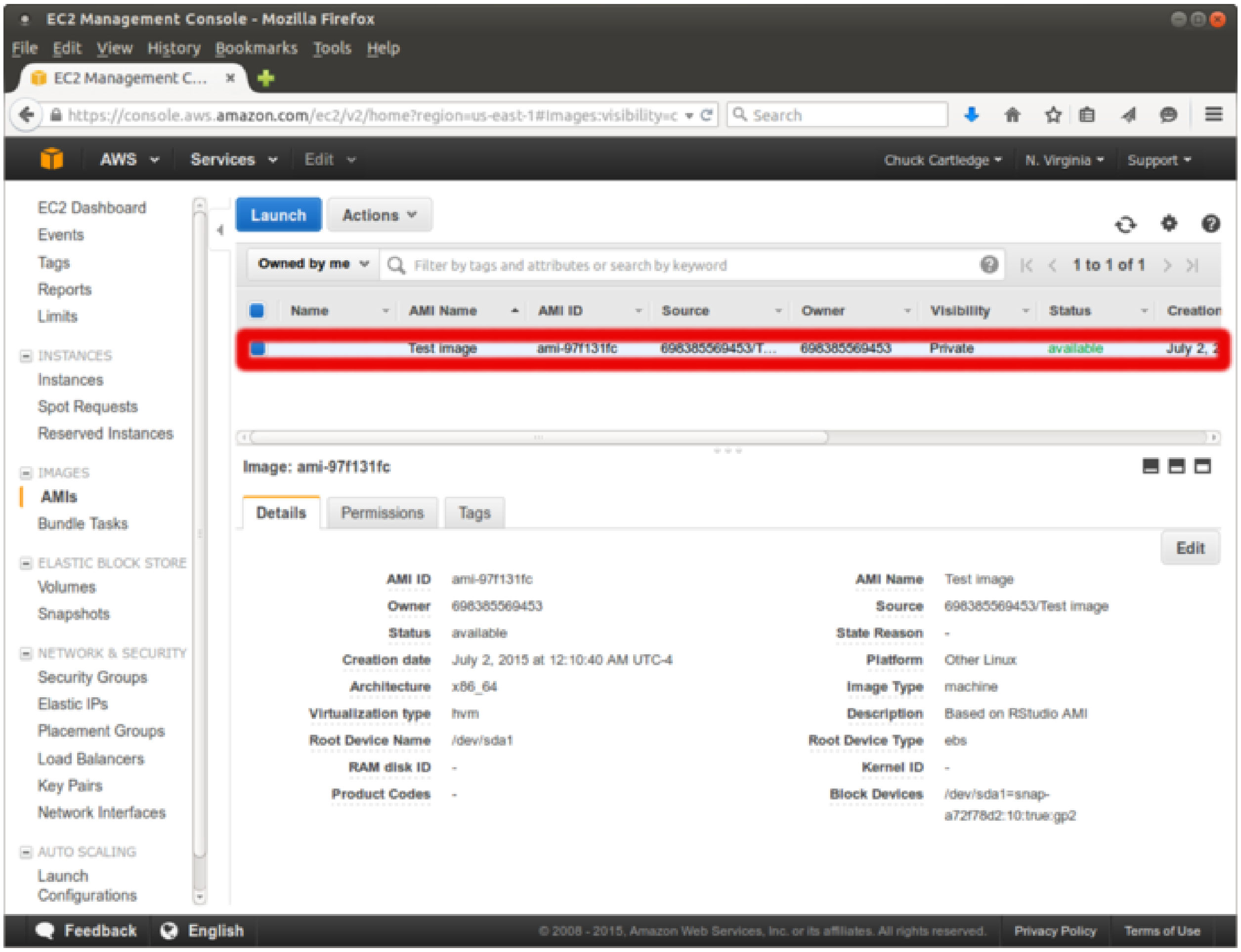

Figure 35: Step 5, verify the image was saved.

5. Verify that the image was saved by AWS (see Figure 35).

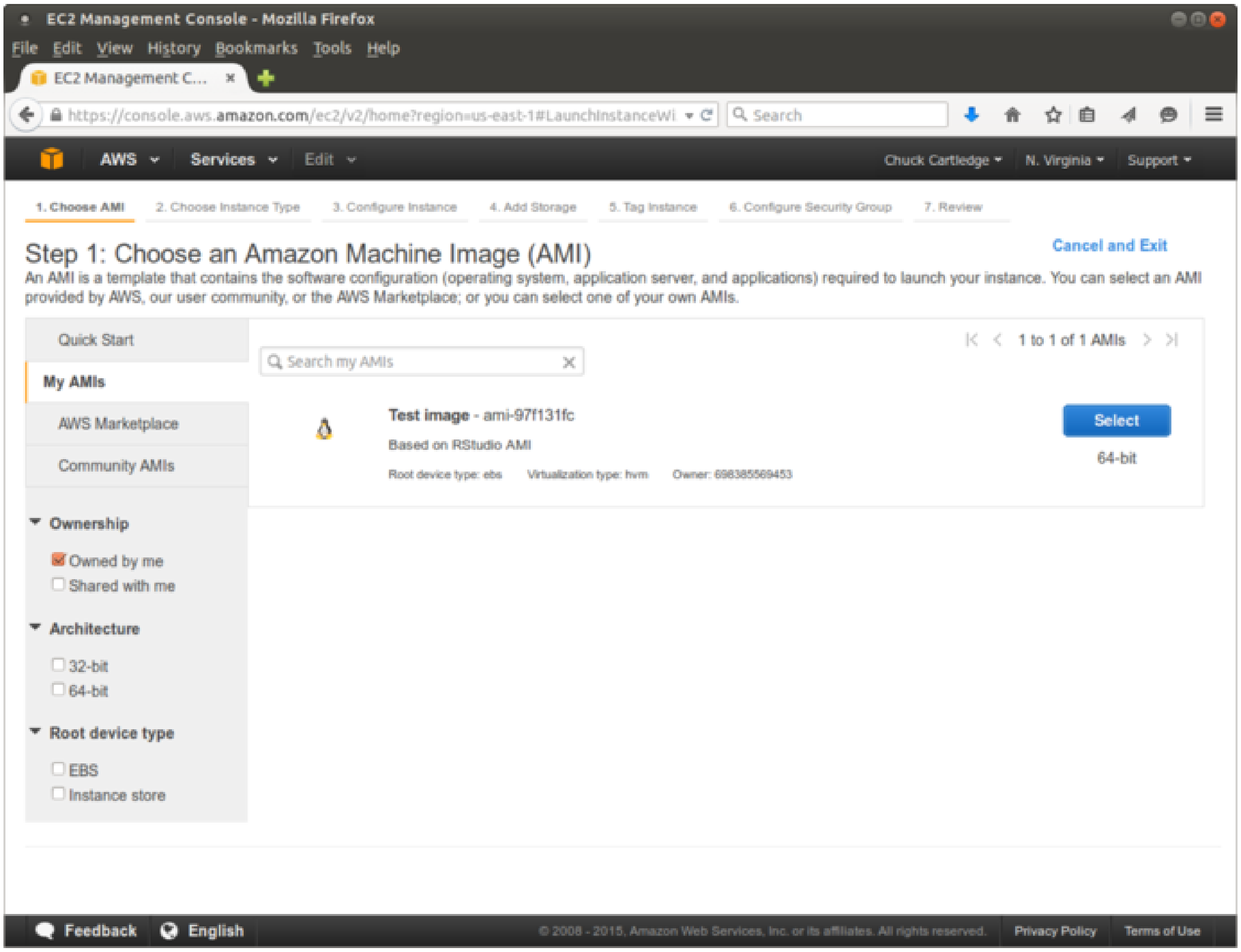

Figure 36: Step 6, see that the image is available.

6. Verify that the image was saved by AWS and is shown in your list of AMIs (see Figure 36).

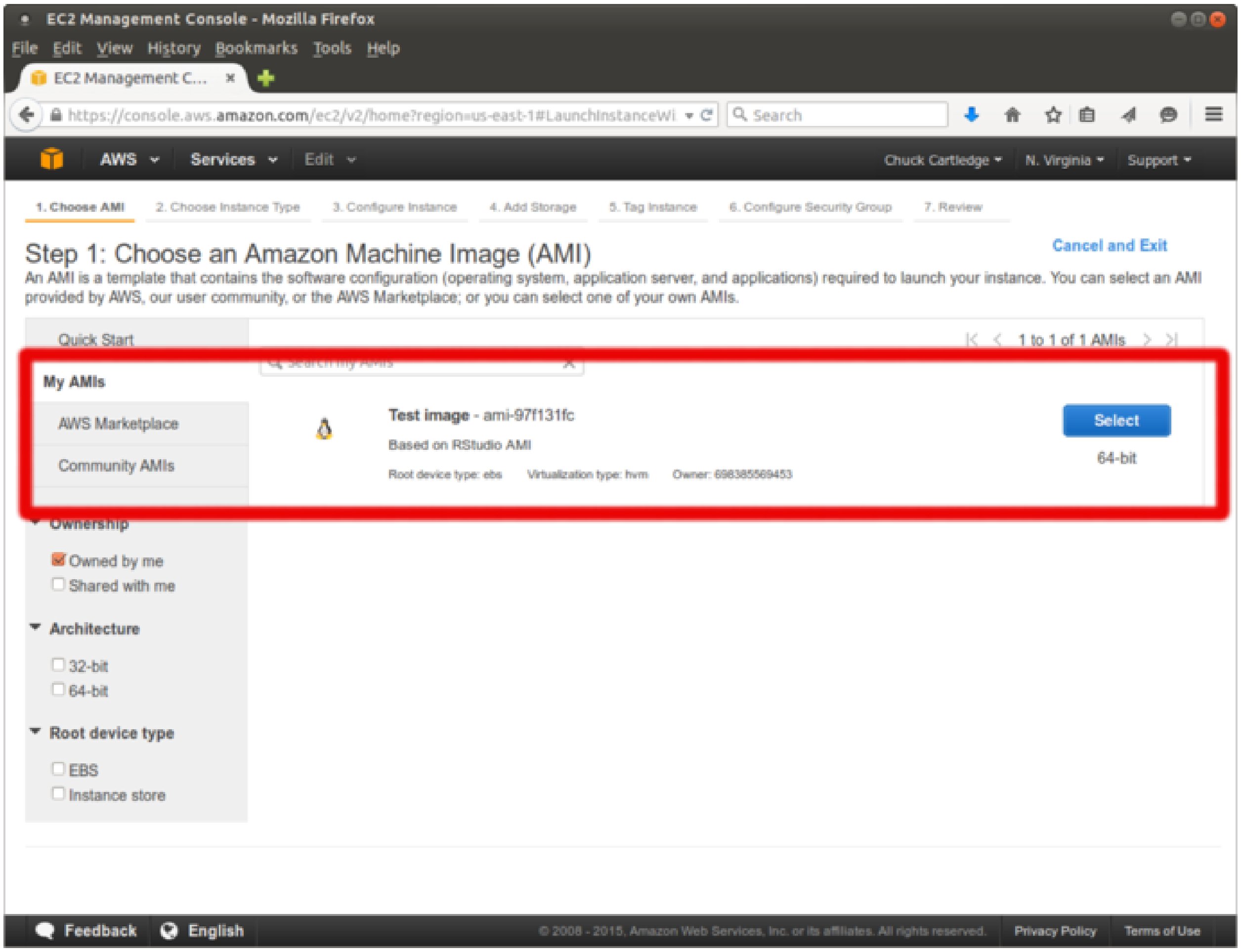

Figure 37: Step 7, see that the AMI is available.

7. Verify that the image is now available for use (see Figure 37).

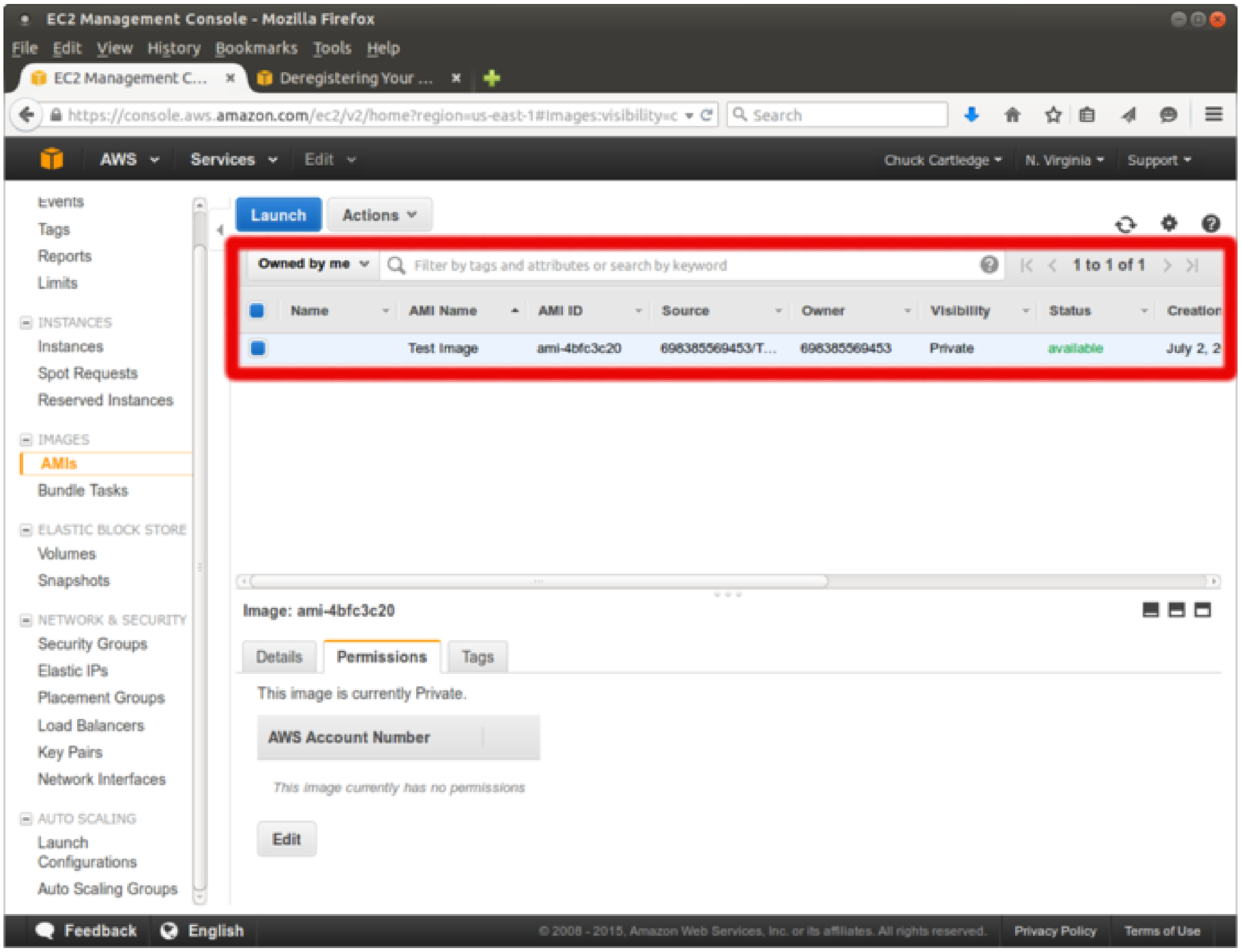

Figure 38: See that the AMI is available.

8. The newly created AMI now belongs to you and can be used like any other marketplace or community AMI (see Figure 38).

9. There are a few things you have to do to remove your AMI. They are:
    (a) Ensure that the AMI is not running (i.e., it must not be on the Instances console, or if it is then its status must be terminated).
    (b) The AMI must be deregistered on the Images ⇒ AMIs console.
    (c) Any file system snapshot must be deleted on the Elastic Block Store ⇒ Snapshots console.

7 Accessing AWS via a command line interface (CLI)

AWS has a command line interface (CLI) that is usable and accessible.

> *"The AWS Command Line Interface (CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts."*
>
> Amazon Staff [2]

The AWS CLI provides a way to use AWS EC2 resources without the need for human intervention or action. For instance, an AWS CLI script could be initiated via a cron table entry to execute at a particular day and time, or as a series of background activities when necessary to support human investigation (things like offline background processes, and so on).

AWS CLI requires the following (assuming a *nix distribution):

1. An AWS account. (The details of setting up an AWS account are beyond the scope of this simple report.)
2. Establishing Identity and Access Management (IAM) credentials.
3. The AWS CLI python script (see Section 7.1).
4. Configure your AWS CLI installation (see Section 7.2).
5. Execute commands on your AWS instance via the command line (see Section 7.3)

7.1 Installing the AWS CLI python script

AWS CLI is a python script. It requires:

- Python 2 version 2.6.5+, or
- Python 3 version 3.3+.

and the python pip package management program.

If you don't have python installed, follow these steps for python 3.4.1

```
sudo apt-get install libssl-dev openssl

wget https://www.python.org/ftp/python/3.4.1/Python-3.4.1.tgz

tar -xvf Python-3.4.1.tgz

cd Python-3.4.1/

./configure

make

sudo make install

./python

quit()
```

If you don't have pip installed, follow these steps:

1. Download the get-pip.py script from:

   `https://bootstrap.pypa.io/get-pip.py`

2. Run the get-pip.py:

   `python get-pip.py`

3. Upgrade pip:

   `pip install -U pip`

To check your the version of your installed python application, execute the following:
`python --version`
To install AWSCLI, execute the following:
`sudo pip install awscli`
To test that you have installed the AWS CLI correctly, execute the following:
`aws --version`

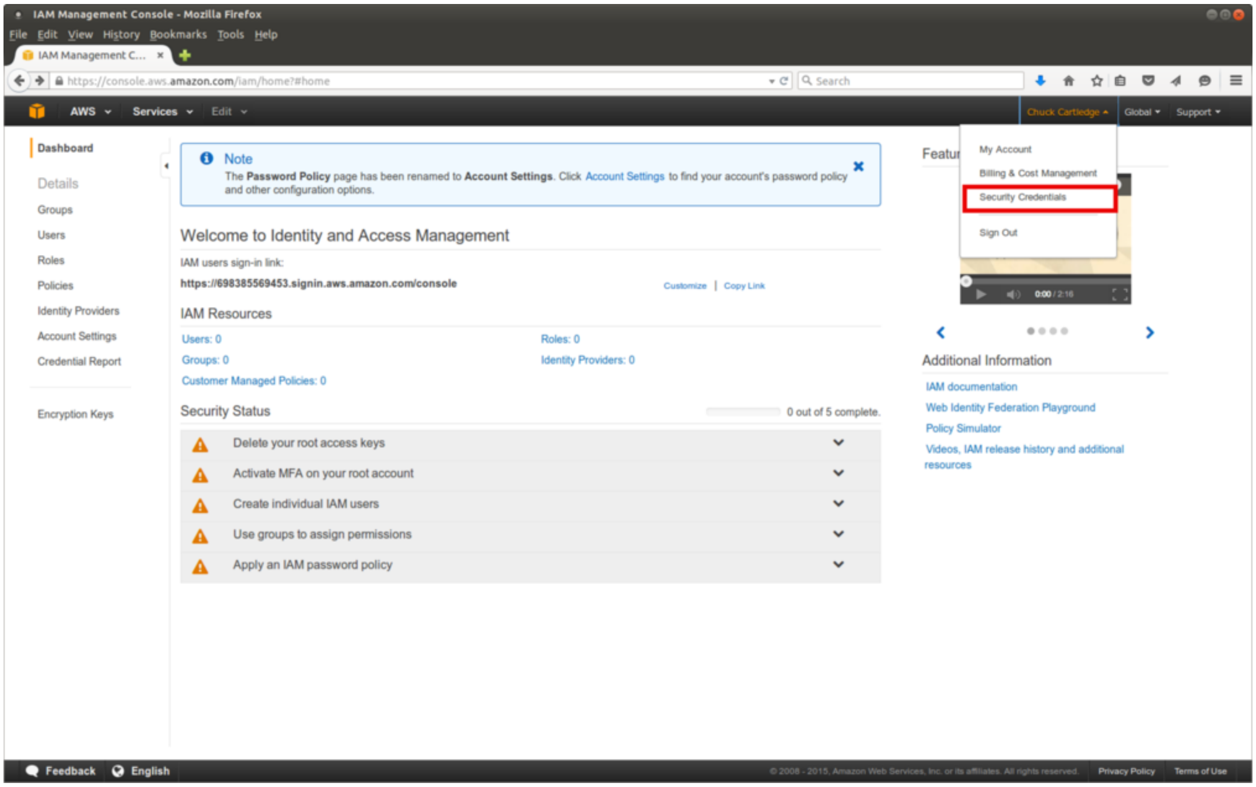

Figure 39: Step 1, update security credentials.

7.2 Configuring your AWS CLI installation

AWS CLI will sign into the AWS and execute commands on your behalf. The AWS CLI requires security credentials (an AWS Access Key ID, an AWS Secret Access Key, and a PEM file).

To get the AWS Access Key ID, and AWS Secret Access Key, follow these steps:

1. Open the AWS management console and select Security Credentials (see Figure 39).

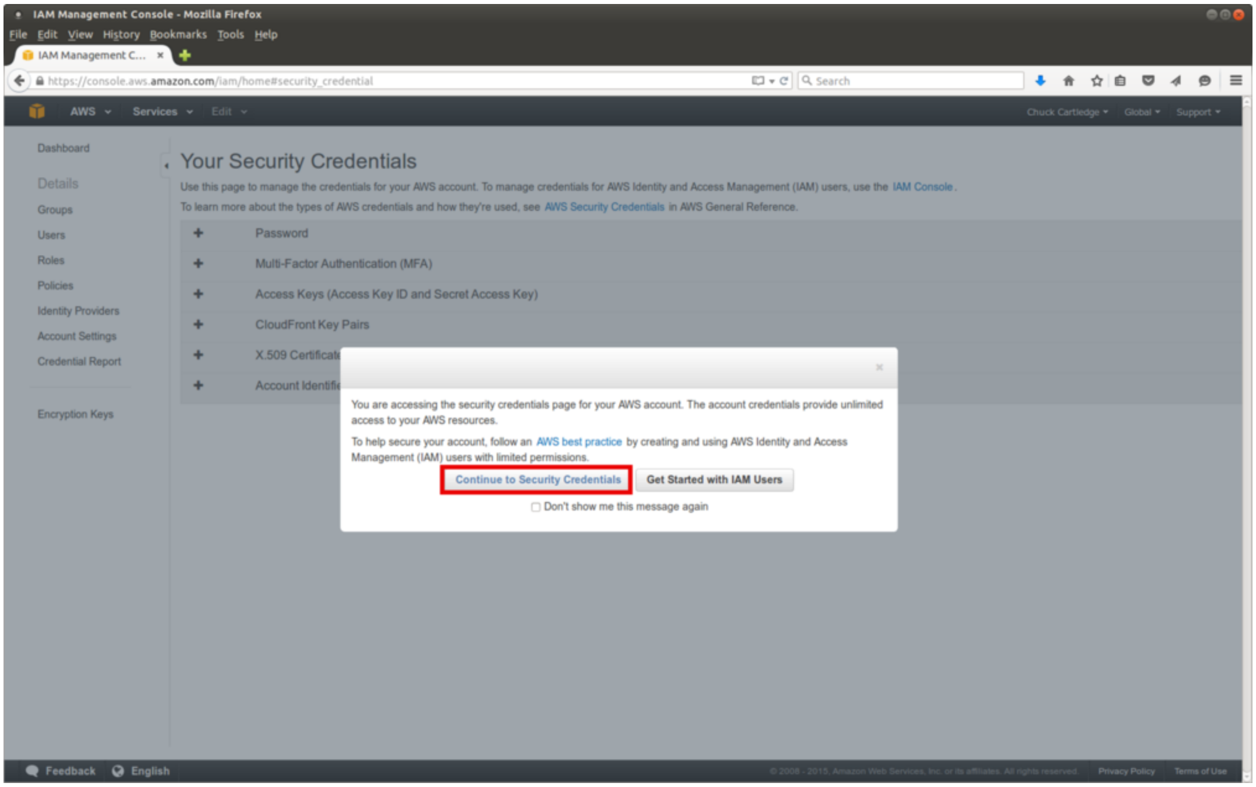

Figure 40: Step 2, accept the security warning.

2. Accept the security warning (see Figure 40).

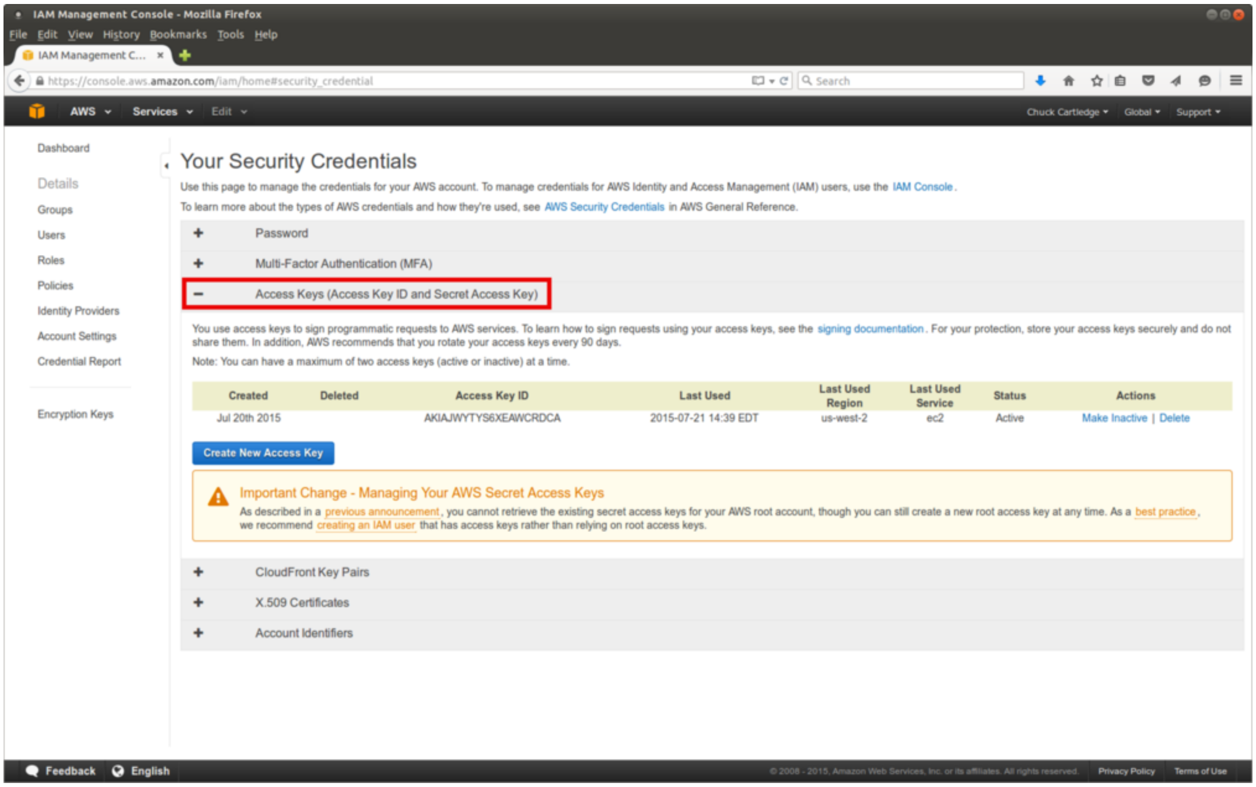

Figure 41: Step 3, choose to create access keys. Data from a previous creation is shown. We will create a new set of keys.

3. Choose to create Access Keys (see Figure 41).

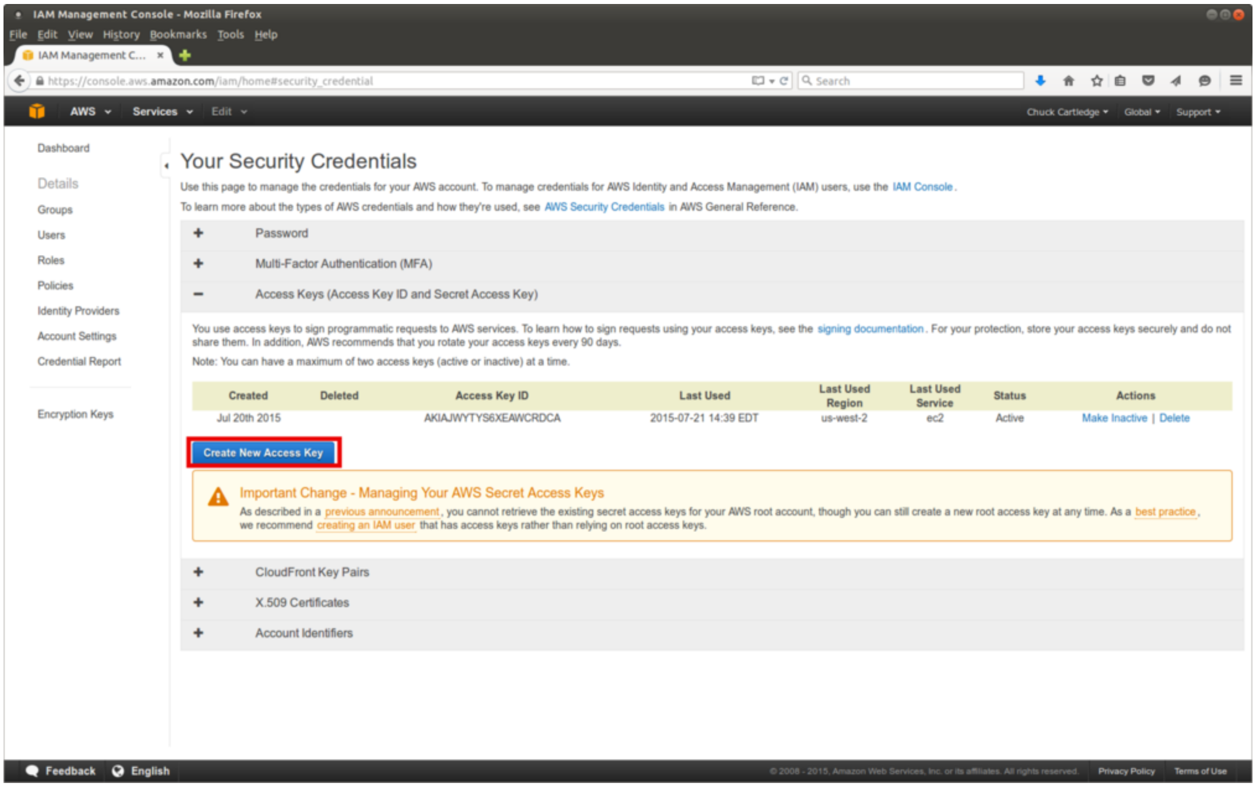

Figure 42: Step 4, create new access key.

4. Create new access key (see Figure 42).

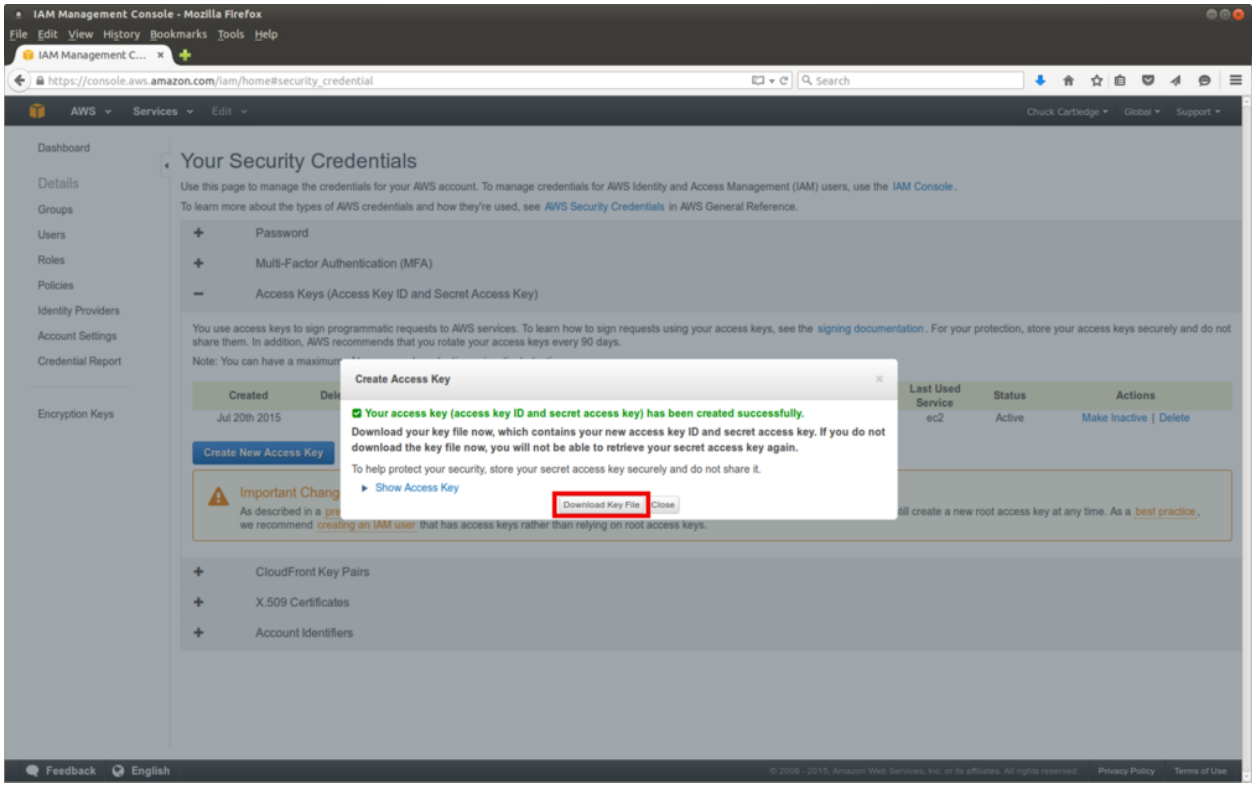

Figure 43: Step 5, download the new access key.

5. Download the new access key file (see Figure 43).

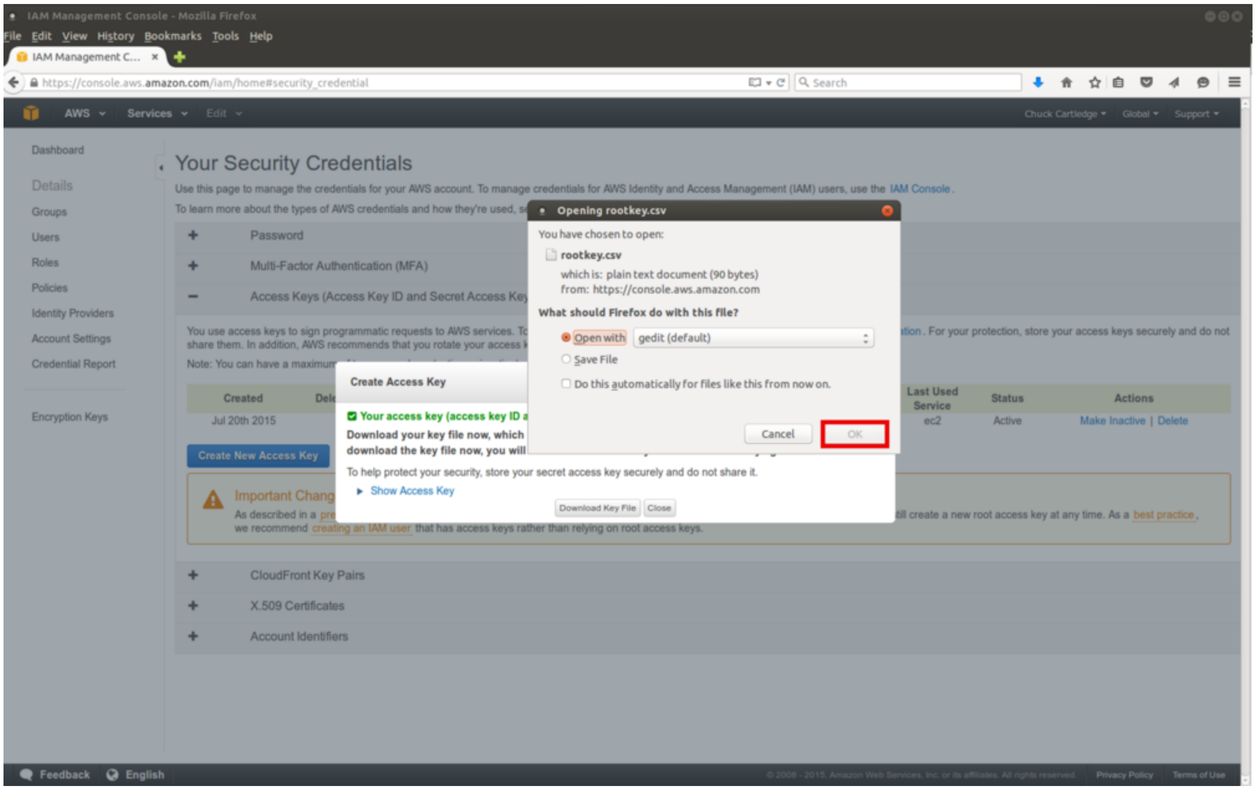

Figure 44: Step 6, open the key file (part one).

6. Open the file (part one) (see Figure 44).

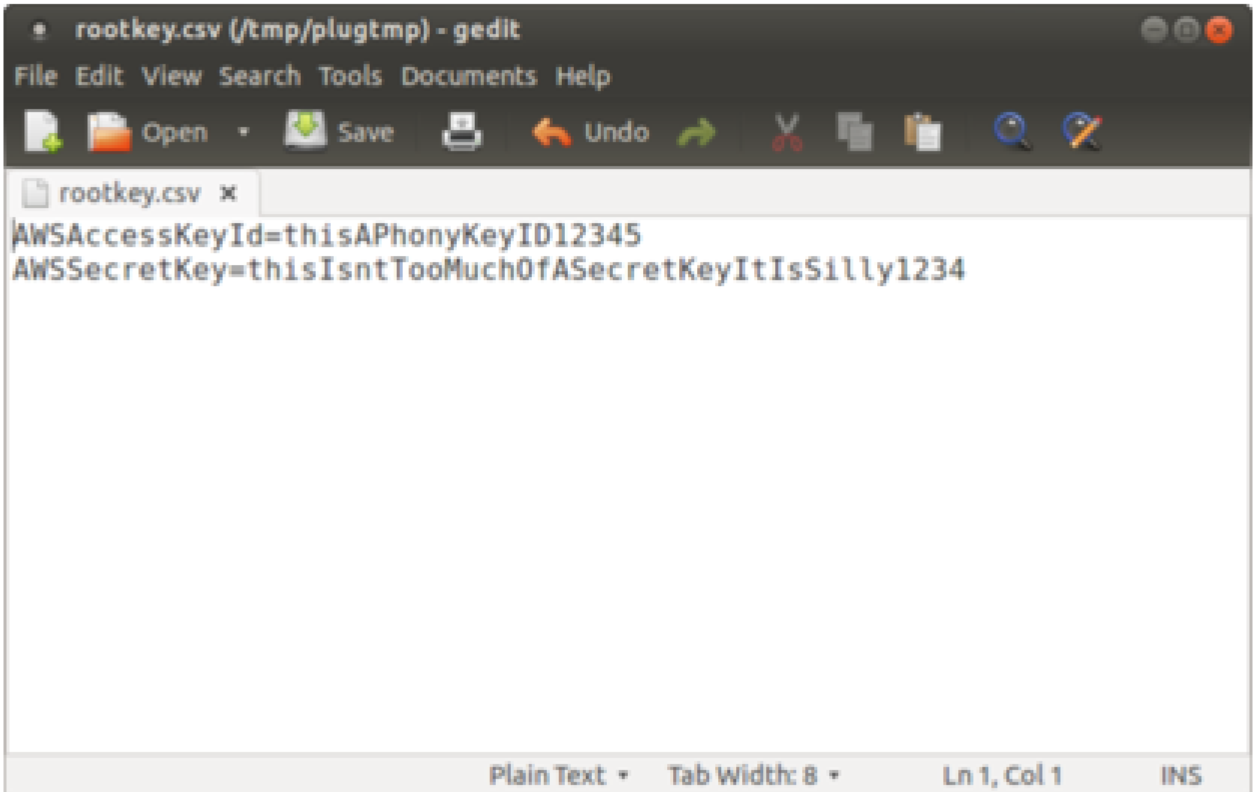

Figure 45: Step 7, open the key file (part two).

7. Open the file (part two) (see Figure 45).

8. Use the AWS CLI program to configure your local data (see Figure 46).

Execute these steps to configure the local AWS data.

```
# start the configuration AWS program
aws configure

# Copy and paste the key ID as shown on the previous page
AWS Access Key ID[]:

# Copy and past the secret key as show on the previous page
AWS Secret Access Key []:

# This is the AWS datat center where the AMI will run
Default region name []:

# This is how data will be returned to your local machine
Default output format []:
```

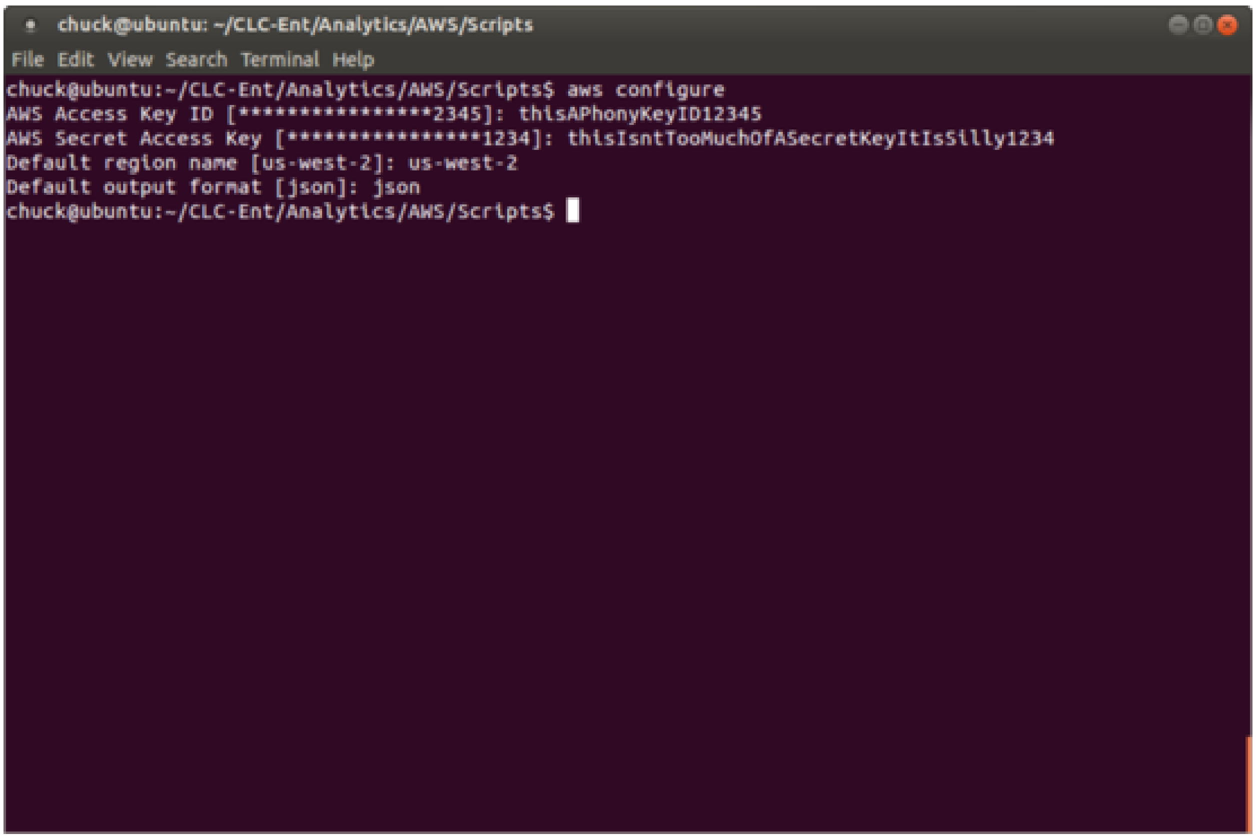

Figure 46: Step 8, use the AWS CLI program to configure you local data. The AWS Access Key ID and AWS Secret Access Key were downloaded from AWS. The local data needs to be configured only when the access data changes. AWS recommends that the keys be changed monthly.

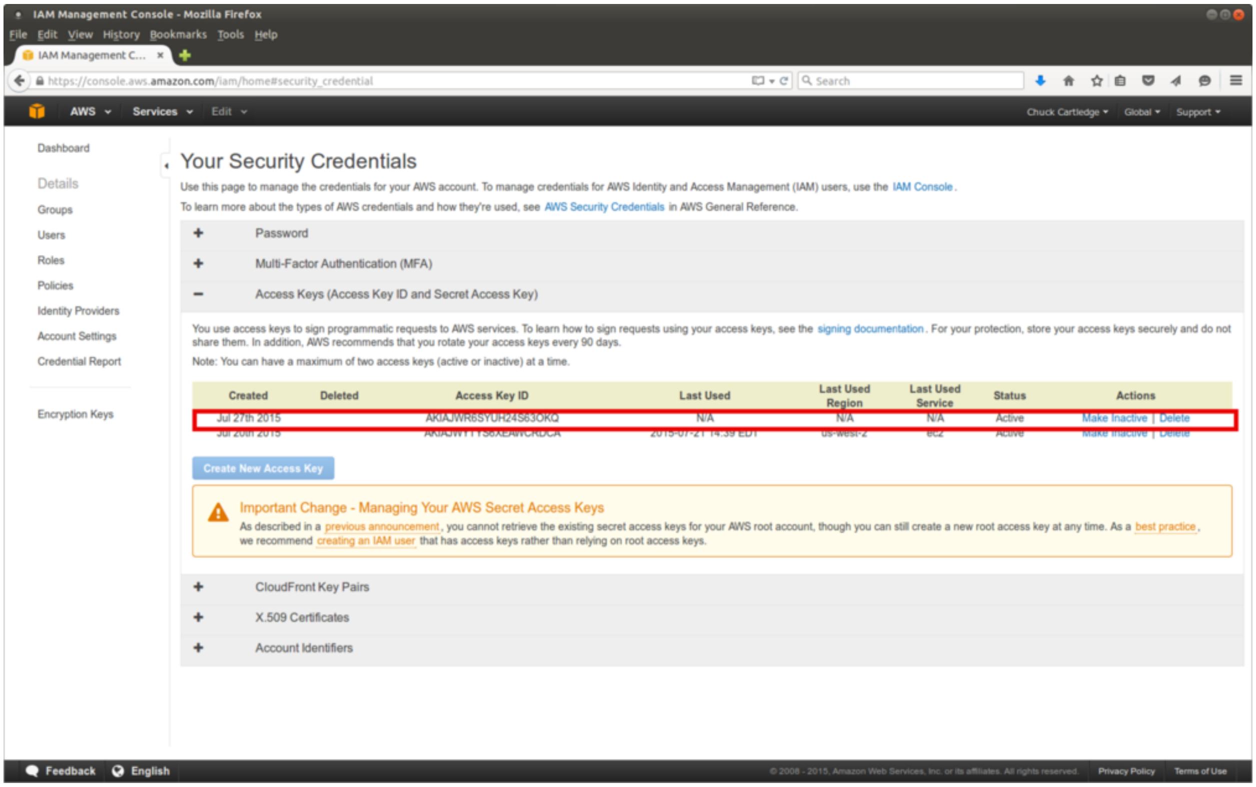

Figure 47: Step 9, verify that your new access keys are available.

9. Verify that your new access keys are available (see Figure 46).

10. Create an AWS EC2 security group using this command:

```
aws ec2 create-security-group --group-name documentation-sg
   --description ''used for training and documentation''
```

11. Authorize the security group using this command:

```
aws ec2 authorize-security-group-ingress
   --group-name documentation-sg
   --protocol tcp --port 22
   --cidr 0.0.0.0/0
```

12. Create a PEM key pair file using this command:

```
aws ec2 create-key-pair --key-name documentation-key
   --query 'KeyMaterial' --output text
   > documentation-key.pem
```

You have now configured your AWS CLI to access the AWS EC2 ecosystem.

7.3 Executing AWS EC2 commands via the AWS CLI

By this point you have installed awscli on your local machine, you have enabled all the appropriate security credentials to allow "hands free" access to AWS EC2, and you are ready to start using the system. There are an infinite number of ways that your system could be exercised, so I've chosen a trivial example that has all the necessary bits and pieces needed to do some work for you.

Attached to this report are a small number of files. They are:

- input.csv – a file containing two numbers to be added together
- readAndAdd.R – an R script file that will be run on the AWS instance
- awsLogin03.sh – a bash script file that will execute the aws commands, cause the readAndAdd.R file to be executed, return the answer to your computer, and terminate the AWS instance.

There are a few things worthy of note in the awsLogin03.sh file:

1. The variable AMI is set to an AWS image that runs only on the US-West region. Like any other image, you will have to choose one that is available in your region and that is configured correctly for your needs.
2. The path to Rscript in the variable remoteCommand was found by executing an ssh to the instance and then locating where the Rscript command exists. The full path name to the executable is required. When you ssh into an instance (or execute a command via ssh), execution starts at your home directory, and you cannot assume that environment variables are set.
3. There is a sleep command in the script that is needed for the AWS ecosystem to settle out. Most of the time the sleep command is long enough, but it might fail from time to time. It would be better to check the status of the describe-instances command, but not worth the trouble for a simple report like this.

4. The AWS instance user is ubuntu.
5. The scp and ssh commands have a lot of optional arguments to allow them to execute and not require a human to accept the new IP addresses of the machines they are connecting to.
6. The bash script secure copys the answer file down from the instance to the local machine.
7. The bash script terminates the instance and cleans up after itself.

If you extract this files from this report you should be able to use AWS CLI to do useful work.

8 Conclusion

The Amazon Web Service (AWS) Elastic Cloud Computing (EC2) ecosystem provides a rich array of virtual computing resources. At times the access to the virtual resources may seem arcane and obtuse, but each step of the process has meaning and purpose. The AWS EC2 environment is a relatively straight forward and understandable environment (the second time around).

References

[1] Louis Aslett, *Rstudio server amazon machine image (ami)*, `http://www.louisaslett.com/RStudio_AMI/`, 2015.

[2] Amazon Staff, *Aws command line interface*, `http://aws.amazon.com/cli/`.

[3] ———, *What is amazon ec2?*, `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html`, 2015.

[4] VMWare Staff, *Server virtualization*, `https://www.vmware.com/se/virtualization/virtualization-basics/how-virtualization-works`, 2015.

Index