CRC Report No. A-103

PREPARING SMOKE-MOVES INPUTS USING THE AWS ENVIRONMENT

Preliminary Report

November 2015

COORDINATING RESEARCH COUNCIL, INC.
5755 NORTH POINT PARKWAY • SUITE 265 • ALPHARETTA, GA 30022

The Coordinating Research Council, Inc. (CRC) is a non-profit corporation supported by the petroleum and automotive equipment industries. CRC operates through the committees made up of technical experts from industry and government who voluntarily participate. The four main areas of research within CRC are: air pollution (atmospheric and engineering studies); aviation fuels, lubricants, and equipment performance, heavy-duty vehicle fuels, lubricants, and equipment performance (e.g., diesel trucks); and light-duty vehicle fuels, lubricants, and equipment performance (e.g., passenger cars). CRC's function is to provide the mechanism for joint research conducted by the two industries that will help in determining the optimum combination of petroleum products and automotive equipment. CRC's work is limited to research that is mutually beneficial to the two industries involved. The final results of the research conducted by, or under the auspices of, CRC are available to the public.

Eastern Research Group, Inc.

Preparing SMOKE-MOVES Inputs Using the AWS Environment

CRC Project A-103

Prepared for:

Coordinating Research Council

Prepared by:

Eastern Research Group, Inc.

November 12, 2015

ERG Project No.: 3987.00.001.001

Preparing SMOKE-MOVES Inputs Using the AWS Environment

CRC Project A-103

PRELIMINARY REPORT

Prepared for:

Coordinating Research Council

Prepared by:

Scott Fincher
Eastern Research Group, Inc.
3508 Far West Blvd., Suite 210
Austin, TX 78731

November 12, 2015

Table of Contents

1.0 Overview

ERG has been using the cloud to prepare on-road emissions for EPA's National Emissions Inventory (NEI) and other national inventories for rulemaking analyses. Many computationally-intensive MOVES runs are required, and the cloud environment provides a relatively inexpensive source of abundant computing resources. The MOVES runs required for SMOKE-MOVES cover many individual counties and months, and are independent of each other, making this application an ideal candidate for parallel computing. As an example, ERG recently ran MOVES in the cloud for an EPA scenario that used 568 separate Amazon Web Services (AWS) computing instances, corresponding to combinations of 284 unique counties for a summer and winter month. This included tens of thousands of individual MOVES runspecs. By running 570 instances in parallel in the cloud, the MOVES runs completed within 60 hours (2.5 days). The total processing time was about 30,000 hours, roughly corresponding to 3.5 years on a single machine.

In its work, ERG has adapted a series of customized scripts originally developed by EPA to handle the mechanics of transferring data into and out of the cloud, organizing data, launching MOVES runs in the cloud, and monitoring their progress including run completion checks. EPA intends to make the scripts public in the future, but they are not currently user friendly. ERG has written this report to describe the full process in a chronological order, beginning with opening an account with a cloud services provider, organizing MOVES input data on a local machine, selecting the desired cloud environment features, transferring data to the cloud, running MOVES, checking results, post-processing, and downloading post-processed results that can then be fed into SMOKE.

As a MOVES end user, should you consider using the cloud environment for executing model runs? For many users, the answer is yes. In the example mentioned above and discussed throughout this document, the time savings for running MOVES in the cloud vs. a single local machine was a factor of 500! In this case, using the cloud took an essentially impossible task and made it reality. While you may not have quite as many runs to do to support your own work, the cloud still presents an opportunity for significant time savings – a scenario requiring only a few hundred MOVES runspecs could still be finished in a single business day, instead of weeks.

In addition, the cloud environment is very cost effective. In the large EPA example above, combined AWS costs for processing time, data storage, and data transfer were approximately $5,000, with more than 90% of that cost for CPU time. (This does not include staff labor costs, which will vary according to familiarity with AWS and MOVES.) Compare this

to the costs of purchasing and running multiple MOVES servers for *years* at a time for *a single scenario!* Smaller workloads on the order of hundreds of runs are also quite affordable; ERG has executed 400 MOVES runspecs to support local emissions inventory works for AWS costs of less than $200.

This report is intended to assist MOVES users with the process of executing model runs in the AWS cloud environment by providing step-by-step instructions, along with screenshots and sample files. Section 2 of this document details initial setup of an AWS account, along with associated security credentials. Section 3 describes the Perl scripts that must be run on a user's local machine to prepare MOVES inputs prior to interaction with the cloud. Section 4 provides instruction on moving files into and out of the cloud, model execution, post-processing, and QA. Finally, Section 5 contains information on frequently encountered issues, troubleshooting tips, and instructions on interacting with individual AWS instances. For user reference, we have also included in an electronic appendix the scripts used to generate SMOKE-MOVES inputs for a specific scenario performed for calendar year 2013 using MOVES2014a.

Supplemented by ERG's experience running several MOVES scenarios in the cloud, this document draws liberally from two existing EPA documents:

- *Running MOVES on Amazon*, Wes Faler, Fluid and Reason LLC. May 2011
- *Documentation of EPA-Side Scripts and Structure for Amazon SMOKE-MOVES*, Harvey Michaels and David Brzezinksi, US EPA OTAQ, November 2012

While much of the process of generating MOVES files for the cloud has largely remained the same since 2012, other aspects have changed – most notably, the Amazon cloud interface itself. We hope this document serves as a useful tool for performing MOVES runs in the cloud in 2015 and beyond.

The author would like to acknowledge the assistance of several individuals in preparing this report:

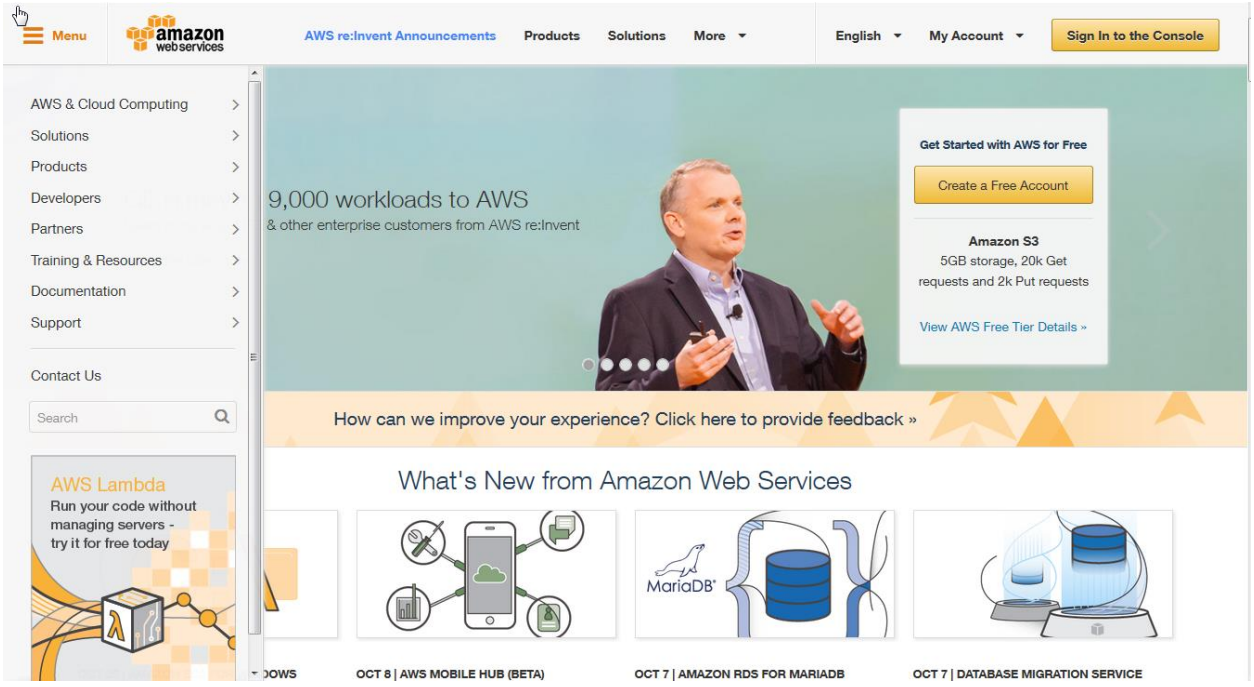
- The authors of the documents above: Wes Faler, Harvey Michaels, and David Brzezinksi,
- Other EPA staff: Alison Eyth, Alexis Zubrow, and David Choi
- ERG Mobile Sources Modeling Team: Sandeep Kishan, John Koupal, Alison DenBleyker, and Doug Jackson

2.0 Initial Amazon Account Setup

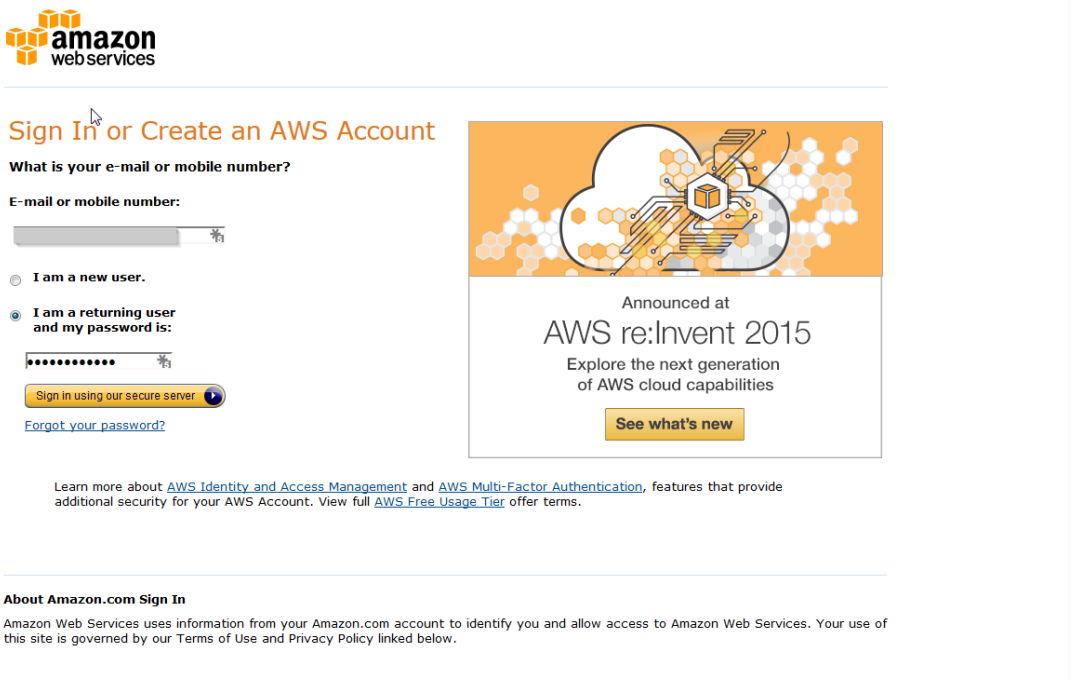
This section describes creation of an Amazon account to use with Amazon Web Services (AWS), as well as associated security credentials. Note that some items in the screenshots that follow are greyed out for security reasons.

2.1 Account Creation

Initial setup of an Amazon account for use with AWS is fairly straightforward: all that is needed is a valid email address, associated contact information, and a valid credit card. Navigate to aws.amazon.com, where you will see the following page:



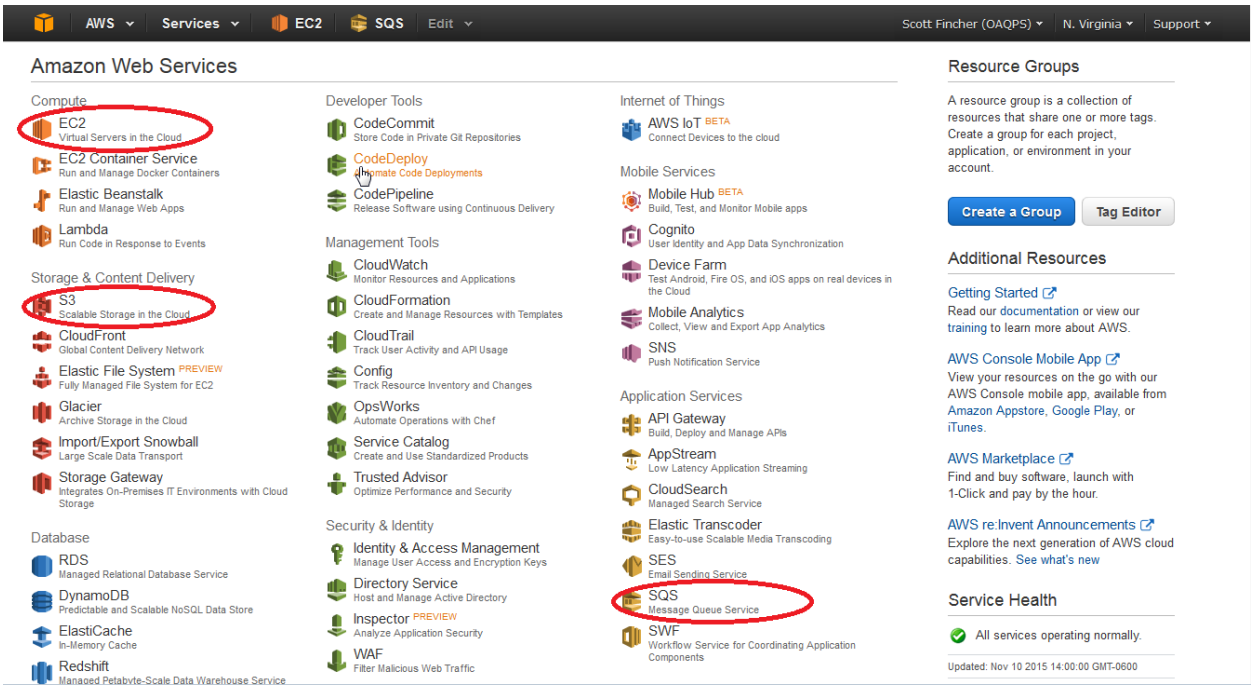
Click on the *Create a Free Account* button and you will be directed to the login page.



Here, enter a valid email address, toggle the *I am a new user button*[1], and click on the *Sign In Using Our Secure Server* button. From there, follow the prompts to provide a password, name, address, phone number, other required contact information, and a credit card number. Amazon will initiate an automated phone call to verify your information. When given the option to choose support, select *Free Support* and continue.

[1] Alternatively, if you already have an existing Amazon account, you can login with those credentials and enable it to use AWS services.

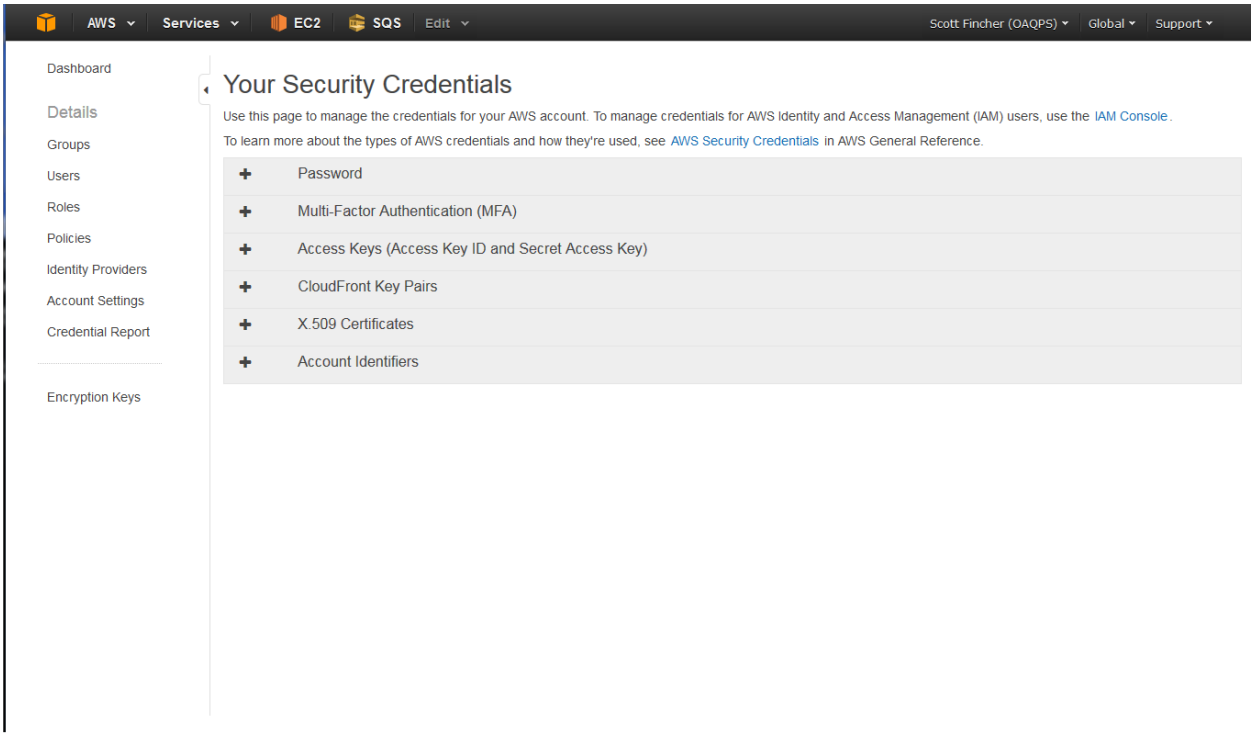
From this point, you should be able to access the AWS management console, which is the jumping off point for accessing various features of the cloud environment.



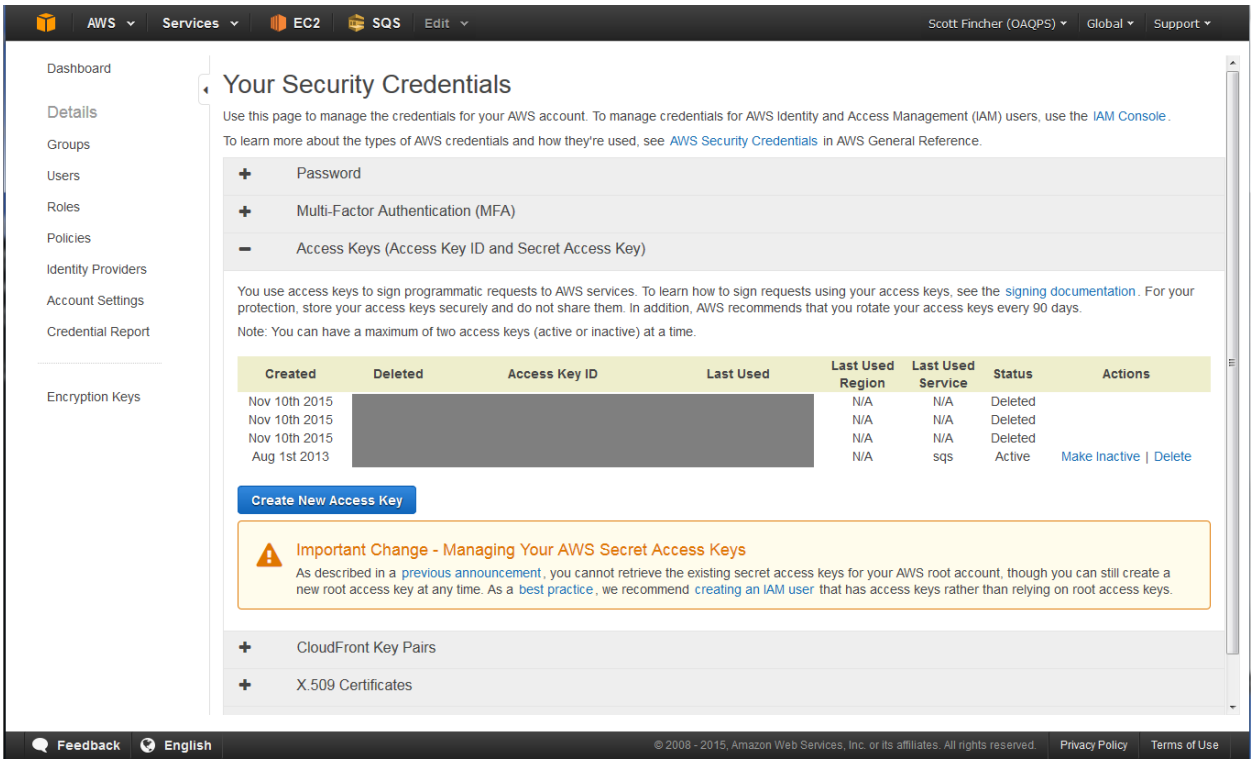
Although there are many such features, for the purposes of MOVES modeling in the cloud, we will use only EC2 (for computing resources), S3 (for data storage), and SQS (for sending instructions to the cloud).

2.2 Security Credentials

Next, create an Access Key that will be needed later. From the AWS Management Console page, select your name in the upper right corner, and click *Security Credentials* in the dropdown menu that appears. You should see a page similar to the following.



Click the plus sign next to *Access Keys* to expand its submenu, and click the blue *Create New Access Key* button. When prompted, click *Show Access Key* to reveal your new Access Key ID and Secret Access Key. Copy these to a file on your local system, and keep them safe for later. Note: these credentials are very important, and must be stored securely! Using them, anyone can start an instance that would be billed directly to your account.



After creating the Access Key, you can move on to setting up modeling scenarios on a local computer using Perl scripts. (You will return to the AWS interface later when it's time to execute model runs.)

2.3 Other Considerations

Instead of credit card billing, you may wish to set up billing via Purchase Order or other mechanism. Unfortunately, this is not possible using the AWS billing interface. If needed, contact Amazon support directly to request alternate billing arrangements.

3.0 Initial Scenario Setup

This section describes setup and QA of required software, scripts, and input data on the user's local machine prior to any processing in the Amazon cloud environment. Software version numbers listed below are current as of December 2015; when installing software, use the latest version available.

3.1 Required Software, Data, and Code

To prepare MOVES runspecs for execution in the cloud, along with other supporting files, the user needs to install the most recent version of MOVES[2]. If not already installed, the MOVES setup program will prompt the user to acquire both MySQL[3] and the Java JDK[4]. Along with these programs, users should download ActivePerl[5] in order to execute many of the included Perl scripts, and the MySQL Workbench[6] to be able to manually view databases and tables during QA. Users can follow default prompts during installation of all software listed above.

In addition to the above, there are a number of other files needed to prepare and execute MOVES runs scripts in the cloud environment. These will be discussed further in the sections that follow. Several have been included in the electronic appendix to this report, but others will need to be obtained from EPA staff. These files include the following:

- Files Provided in Appendix
  - Perl Scripts and their associated input files listed in Section 3.2
- Files Requested from EPA
  - Representative County Databases (CDBs)
  - LEV Databases
  - MOVES-Specific Amazon Machine Image (AMI)
  - JAR files
    - MOVES Code
    - MOVES Databases
    - Postprocessing Code

[2] Available at http://www3.epa.gov/otaq/models/moves/. Current version is MOVES2014a.
[3] Available at https://dev.mysql.com/downloads/mysql/. Current version is 5.7.10.
[4] Available at http://www.oracle.com/technetwork/java/javase/downloads/index.html. Current version is 8u66.
[5] Available at http://www.activestate.com/activeperl/downloads. Current version is 5.22.0.2200.
[6] Available at https://dev.mysql.com/downloads/workbench/. Current version is 6.3.6.

3.2 File System Organization

As defined by EPA staff, in the context of MOVES scenario execution, a **project** is "a set of runs that use the same representative counties and the same met[eorological data]" as input. The runs can involve different calendar years or control strategies so long as each project is connected to a single execution of the Runspec Generator script (discussed below). A **batch** is a subset of a project, and is usually delineated by a combination of county and month. For example, a single batch might involve inputs for Harris County, TX, in the month of July. Further, a **job** is a subset of a batch, and consists of individual MOVES runspecs, which are often separated according to temperature regime or process type. In the included example, you will examine a single project, with 570 batches, each containing between 75 and 150 jobs.

It is often easiest to start create a project by using scripts and inputs from an existing project. Copy the example files from the attached electronic appendix and note both the provided files and the directory structure. Required files as input for MOVES runspec generation include the following:

- Perl Scripts, which are run in the following order and are discussed in more detail in the sections that follow:
  - RepCnties.plx
  - runspec_generator_v0.33_26Mar13.plx
  - LoadZmh.plx
  - CreateandPopulateAmazonDirStructureSmokeMoves.plx (referred to as CreateandPopulate below)
  - CreateBatchFilesForAmazonRunsSmokeMoves.plx (referred to as CreateBatchFiles below)
- Under subdirectory **SampleRunSpecs**, three text files that identify pollutants and processes to be used in creating the RatePerDistance (RD), RatePerProfile(RP), and RatePerVehicle(RV) runspecs, each of which are included as job types in a given batch.
- Under subdirectory **PerlScripts**, two scripts called by other scripts in the process.
- Various inputs required for previously mentioned Perl scripts:
  - RepCounty text file, which contains a list of counties and an 8-digit date of their associated representative county database (CDB)
  - Met4moves input files, obtained from EPA

- Zonemonthhour (zmh) files, created by the runspec generator script and used by LoadZmh
- Empty csv tables, used by the runspec generator and provided here
- RunSpecGenControl, provided here and modified by the user to reflect the project

The Perl scripts and supplemental files described above are used to create the files listed below, which are the direct inputs for MOVES execution in the cloud. They include the following:

- RunSpecGenOutput
- Under subdirectory **InputDatabases**
  - Representative county databases – these 284 counties were chosen by EPA as being representative for the entire US. The databases include many MOVES inputs that are county specific, including IM, fleet distributions, VMT, activity data, and a variety of other data.
  - LEV Databases - required for modeling the effects of LEV in counties where it applies. Includes updates to the MOVES *emissionratebyage* table.
  - Other user supplied databases. This could be modifications to any input desired by the user. In the case of the included example, modifications to the *fuelsupply* table were made and are included.
- Under subdirectory **AmazonStructure**
  - A subdirectory for each project
    - A **0scripts** subdirectory containing batch files, created by the Create Batch File script, for interaction with the cloud environment
    - A subdirectory containing project databases to be uploaded to the cloud
    - A subdirectory for each batch
      - A subdirectory containing batch databases to be uploaded to the cloud
      - A subdirectory for each job associated with a particular batch
        - A runspec and input database for each job

In addition, execution of the scripts requires files in an additional \amazon folder in your MOVES directory. This folder contains commands specific to interacting with the cloud environment, and must be requested from EPA.

3.3 Perl Script Execution

3.3.1 RepCnties Script

Start by modifying the RepCnties.plx script. Open it in a text editor (for example, Notepad++ or UltraEdit). You will need to edit the lines listed below to reflect appropriate project name, calendar year, and directory paths for your project. Variables are denoted in Perl with the $ operand, and five variables of interest in RepCnties.plx are listed in the sample below. Note carefully the double backslash syntax ("\\") within quotes; this is necessary for Perl to interpret the paths correctly. Note that most paths can be on a network drive if desired, so long as it is mapped appropriately.

```
$project='2013-MOVES2014a';
$year=2013;
$filedir="P:\\EPA_MOVES_3-04\\$project\\SmokeMovesRunSpecGenerator_2013-MOVES2014a";
$repcdblist="P:\\EPA_MOVES_3-04\\$project\\285RepCos2013_M2014_20151103.txt";
$output="RunSpecGenRepCnties_$project.txt";
```

Save your changes, and open a command window (to do this, press Windows-R, type **cmd** in the open dialog, and press Enter). Using the Windows **cd** command, navigate to the directory where RepCnties.plx is located, type **perl RepCnties.plx**, and press Enter. The program will run, and produce a RunSpecGenRepCnties text file. This file will be used in execution of the next script.

3.3.2 RunSpecGenerator Script

The runspec_generator_v0.33_26Mar13.plx script itself does not require any modification. Rather, you will provide as input to it the text file produced by RepCnties, and in addition provide a RunSpecGenControl input file. This control file consists of the following lines:

```
DBHOST       = localhost
BATCHRUN     = 2013-MOVES2014a
OUTDIR       = P:\EPA_MOVES_3-04\2013-MOVES2014a\SmokeMovesRunSpecGenerator_2013-
MOVES2014a\RunSpecGenOutput_2013-MOVES2014a\
MOVESHOME    = C:\EPA\MOVES\amazon20130603
MODELYEAR    = 2013
POLLUTANTS   = OZONE,PM,TOXICS,GHG
DAYOFWEEK    = WEEKDAY, WEEKEND
METFILE      = P:\EPA_MOVES_3-04\2013-
MOVES2014a\met4moves\MOVES_RH_DAILY_2013ej_v6_13i_12US2_2013001-2013365.txt
RPMETFILE    = P:\EPA_MOVES_3-04\2013-
MOVES2014a\met4moves\MOVES_DAILY_2013ej_v6_13i_12US2_2013001-2013365.txt
```

Of interest here are the BATCHRUN, which should reflect your project name; the OUTDIR, which should point to a directory for RunSpecGenerator output; the MODELYEAR, which is not actually a model year but should instead reflect a calendar year of interest; and the METFILE and RPMETFILE, which should point to your met4moves data obtained from EPA. Do not modify the other parameters in this example. Edit the file as needed in a text editor and save your changes. To execute the script, use a command window to navigate to its location, type **perl runspec_generator_v0.33_26Mar13.plx [RunSpecGenControl] [RunSpecGenRepCnties]** and press Enter, where the two bracketed values are the names of your particular input files – no brackets are required at the command prompt. This program will take some time to complete, on the order of several hours. In this example 568 batches, containing more than 50,000 runspecs, will be generated. When complete, the program will generate runspecs in the OUTDIR folder, along with associated zonemonthhour files, as well as other XML and batch files that are unused in this example.

3.3.3 LoadZMH Script

Next, open the LoadZMH.plx file in a text editor, and edit the lines listed below to reflect appropriate project name, calendar year, and directory paths for your project. Again, be mindful of the double backslash convention in Perl.

```
$project='2013-MOVES2014a';
$year=2013;
$mysqldata="\"C:\\ProgramData\\MySQL\\MySQL Server 5.6\\data\"";
$homedir="P:\\EPA_MOVES_3-04\\2013-MOVES2014a\\SmokeMovesRunSpecGenerator_2013-
MOVES2014a";
$repcdblist="P:\\EPA_MOVES_3-04\\2013-MOVES2014a\\285RepCos2013_M2014_20151103.txt";
$mvdroutput="$homedir\\RunSpecGenOutput_2013-MOVES2014a";
$mvdroutputlocal_backslash="C:\\a\\${project}_runspecgenoutput_zmh";
$mvdroutputlocal_fwdslash="C:/a/${project}_runspecgenoutput_zmh";
$zmhdbs="$homedir\\2013-MOVES2014a_zmh";
```

Note that the mvdroutputlocal variables denote a directory on your local machine where ZMH files will be copied before input to your local MySQL installation.

Save your changes to the script, and execute it using a command window by navigating to its location, typing **perl LoadZMH.plx**, and pressing Enter. This script takes some time to execute, usually on the order of hours, since the system is creating many thousands of small ZMH databases for each job to be modeled. Per the EPA documentation, "the total number of databases produced should equal the number of zmh.csv files. This number is also the number of jobs that will be produced and run." Do a quick QA check at this point to ensure the number of databases output is what you expect. If not, check your input parameters for typos, and examine and Perl errors that might have occurred.

3.3.4 CreateAndPopulate Script

Initially, you will want to run the CreateAndPopulate script in a modified fashion, to generate a set of jobs for a single batch. In this way, you can carefully QA the created runspecs to make sure they include all of the desired model options, and also execute a runspec locally to catch any errors in our inputs that might otherwise be missed. This is an important step – if it is ignored, you may end up wasting time having to recreate the entire set of batches.

To start, open the CreateAndPopulate script, and edit the lines below to reflect project, years, and paths as appropriate:

```
$project='2013-MOVES2014a';
$year=2013;
$vers=20151106;
$repcdblist="285RepCos2013_M2014_20151103.txt";
$case="$project";  # for when the case does not include a scen & year
$casevers="${case}-${vers}";
$casepath="P:\\EPA_MOVES_3-04\\$project\\AmazonStructure\\$casevers";
$dbdir="P:\\EPA_MOVES_3-04\\$project\\InputDatabases";
$runspecgenyear=2013;  # This can be different from $year
$fuelsdb='M2014a_fuelsupply';
$cdbdir="P:\\EPA_MOVES_3-04\\$project\\InputDatabases\\2013RepCos_20151103";
$zmhoutput="P:\\EPA_MOVES_3-
04\\$project\\SmokeMovesRunSpecGenerator_$project\\${project}_zmh";
$rdrunspectemplate='samplerunspecs\\RDCB05CB6NEI_2013_core.mrs';
$rprunspectemplate='samplerunspecs\\RPCB05CB6NEI_2013_core.mrs';
$rvrunspectemplate='samplerunspecs\\RVCB05CB6NEI_2013_core.mrs';
```

Next, search CreateAndPopulate for the string “diag”; There are three lines in the script containing this string (see below), and each has a leading pound character (#) that denotes a comment. Delete the leading pound character from each of these lines. This will limit the creation of runspecs by the script to a single county, month, and job.

Save your changes to the script, and execute it using a command window by navigating to its location, typing **perl CreateAndPopulate.plx**, and pressing Enter. Verify that the program has generated output for a single job in the AmazonStructure folder; if it hasn’t, make corrections to the script and try again. Once you’re successful creating a single runspec, once again edit the CreateAndPopulate script, this time re-inserting a leading pound character on the following line of code, like so:

```
#       if($jobcount>1){last;}  # Limit to one job for diagnostics
```

Re-run the script. This time, all of the RD, RP, RV runs for a single batch will be created. Navigate to the batch output directory and verify the runspecs exist. Select one runspec each from RD, RP, and RV and open them in a text editor for QA. Examine them carefully to ensure the inputs are correct, especially the calendar year, list of sourcetypes, pollutants/processes/fuels, and calls to any external databases.

3.4 Local QA

Next, verify that the runspecs generated by the script can be correctly interpreted and processed by MOVES itself. Select a single RP runspec and copy it to your local machine (if it's not already there), along with its required input databases. Open the MOVES GUI and load the runspec. All of the checkmarks in the GUI should be green; if not, examine them further by drilling down into the inputs to determine the cause of the problem.

Execute the RP runspec and wait for it to complete, which should take a few minutes. When it is finished, take a look at the output database. Does the *moveserror* table contain any entries? Is the *rateperprofile* table populated with data? Does the *movestablesused* table correctly list databases that were used during the run?

If desired, you can also perform similar QA on the RV and RD runspecs, although if the RP passes QA, it is likely the others will as well. Generally, it is best to start with an RP run since its execution time is relatively short, especially compared to RD which can take several hours.

Once local QA is complete, edit the CreateAndPopulate script a final time, this time re-inserting leading pound characters on the following two lines of code, like so:

```
#if($cntycount>1){last;}  # Limit to one county for diagnostics
#if($monthcount>1){last;}  # Diagnostic limit to one month
```

This will enable the script to generate output for all county and month combinations.

3.5 Full Runspec and Batch File Generation

Now you are prepared to generate all of the runspecs necessary to support output for SMOKE MOVES. As before, run the CreateAndPopulate script you just edited above. In addition to creating runspecs, the script also compresses them, along with their associated input databases, into .jar files (which are functionally the same as .zip files, and can be examined with any archive software[7]). In the included example, this process takes about a day of processing time.

Finally, you must create a series of batch files that will allow our local file system to more easily interact with the AWS cloud environment. Open the CreateBatchFiles script, and edit the lines below to reflect project, years, and paths as appropriate. Many of these variables will be similar, or even identical, to variables from CreateAndPopulate.

[7] The freely available 7zip file archiver, downloadable at http://www.7-zip.org/, is recommended.

```
$project='2013-MOVES2014a';
$year=2013;
$vers=20151106;
$movescode='20151028';
$mddb="movesdb20151028";
$codebucket="moves_code_bucket";
$dbbucket="moves_db_bucket";
$repcdblist="285RepCos2013_M2014_20151103.txt";
$casepath="P:\\EPA_MOVES_3-04\\$project\\AmazonStructure\\$casevers";
$casepathp="P:\\\\EPA_MOVES_3-04\\\\$project\\\\AmazonStructure\\\\$casevers";
$accesskey='ABCDEFGHIJKLMNOPQRS';
$secretkey='Abcdefg1234567!@#$%^&Abcdefg1234567!@#$%';
```

Of particular interest are the bolded variables above. **movescode** and **mddb** will reflect the version of the MOVES code and MOVES databases provided by EPA, and will be discussed further in the following section. **codebucket** and **dbbucket** are names for the Amazon buckets where the MOVES code and database, will be stored, respectively. Note carefully that these buckets must have names that are unique across all of AWS. The **accesskey** and **secretkey** are text strings generated during the creation of your account. And allow for direction connection to AWS via batchfiles.

Once editing is complete, save changes and execute CreateBatchFiles using a command window by navigating to its location, typing **perl CreateBatchFiles.plx**, and pressing Enter. This script runs very quickly, and generates a number of different batch files in the AmazonStructure/[Project]/0scripts directory (not all of which you will use). Batch files of interest to this example are discussed in the following section.

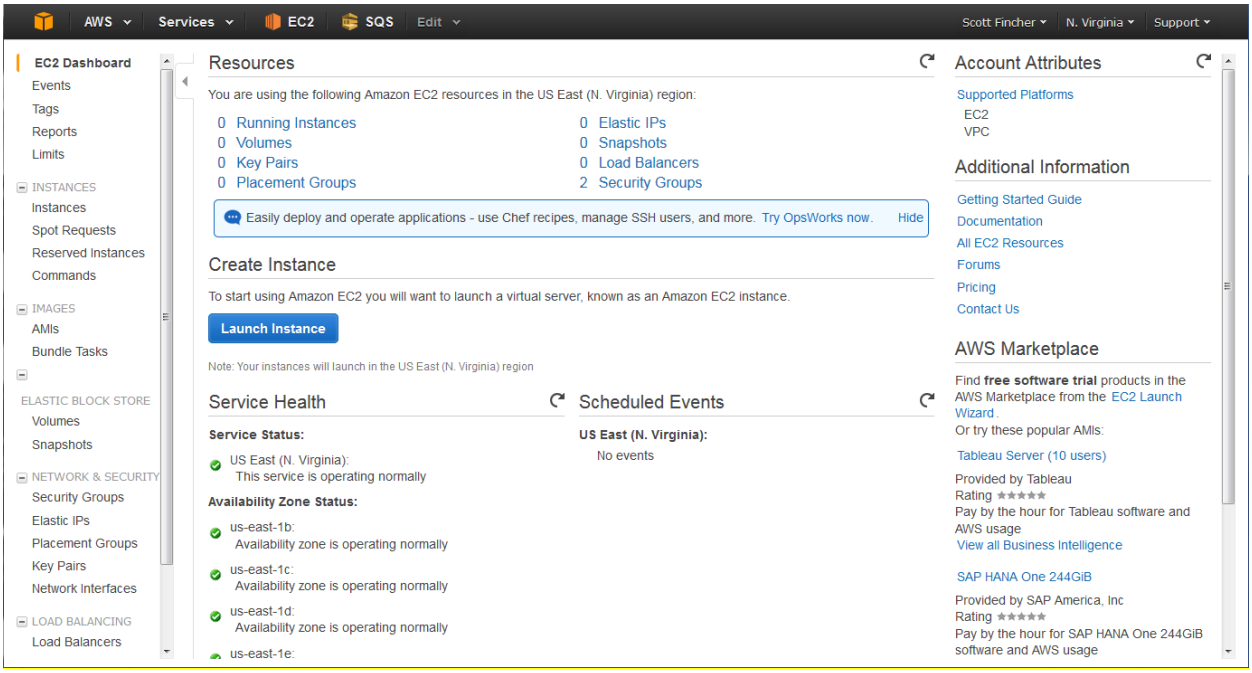
4.0 Execution of MOVES in the AWS Cloud Environment

This section describe the process of uploading and executing MOVES batches generated locally, along with QA, post-processing, and downloading of model results.

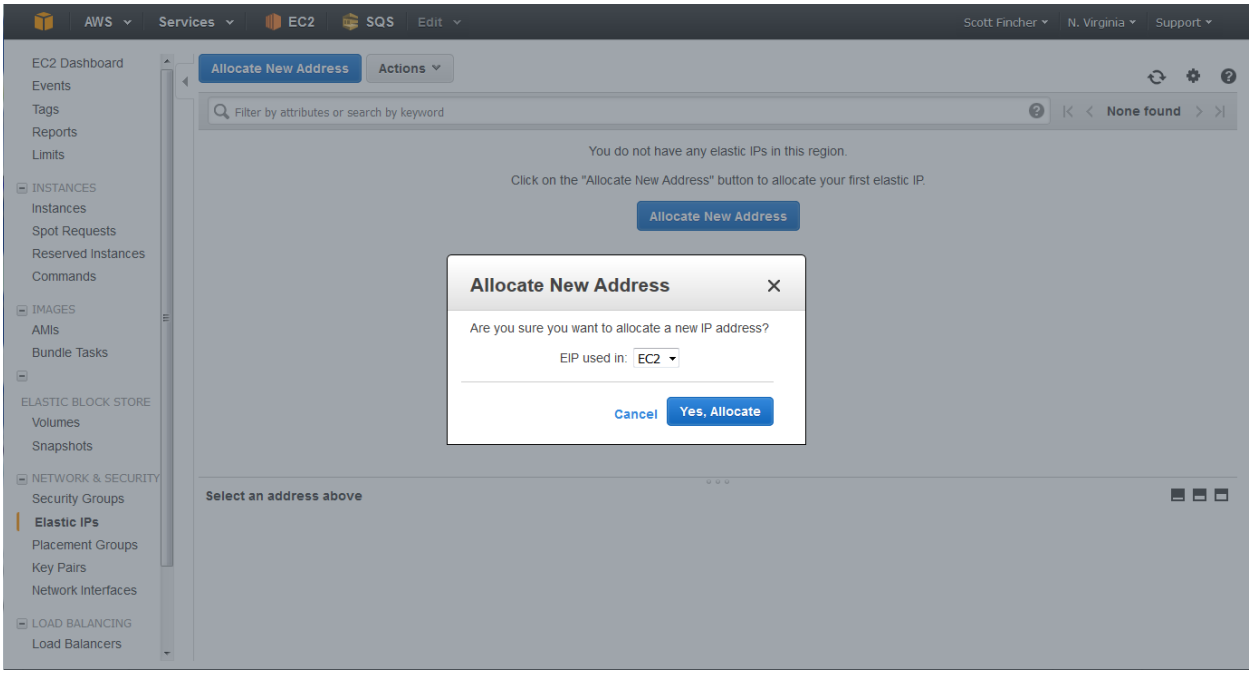
4.1 Amazon AWS Options and Setup

Section 2 above discusses creation of security credentials associated with your Amazon account. Beyond that, there is some additional one-time setup that must take place prior to proceeding with uploading and executing MOVES runs.

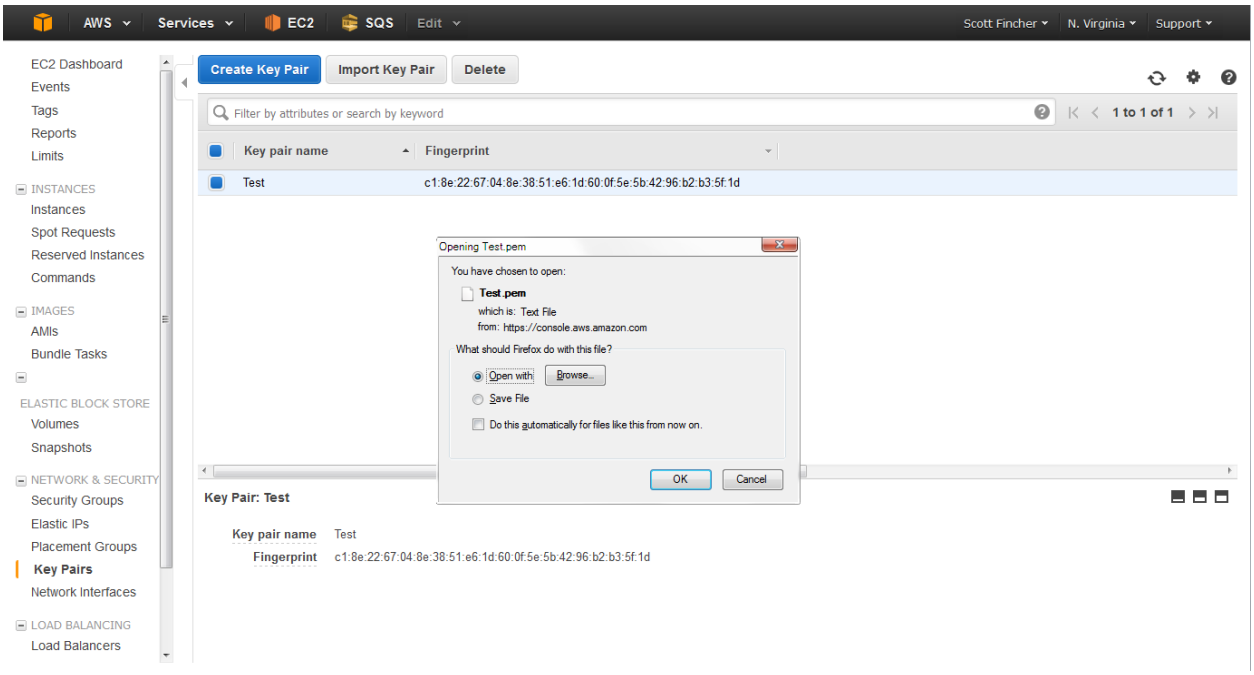
First, login to your account. You should be presented with the AWS management console. Click on *EC2* to switch to the EC2 management console. You should see a screen similar to this:



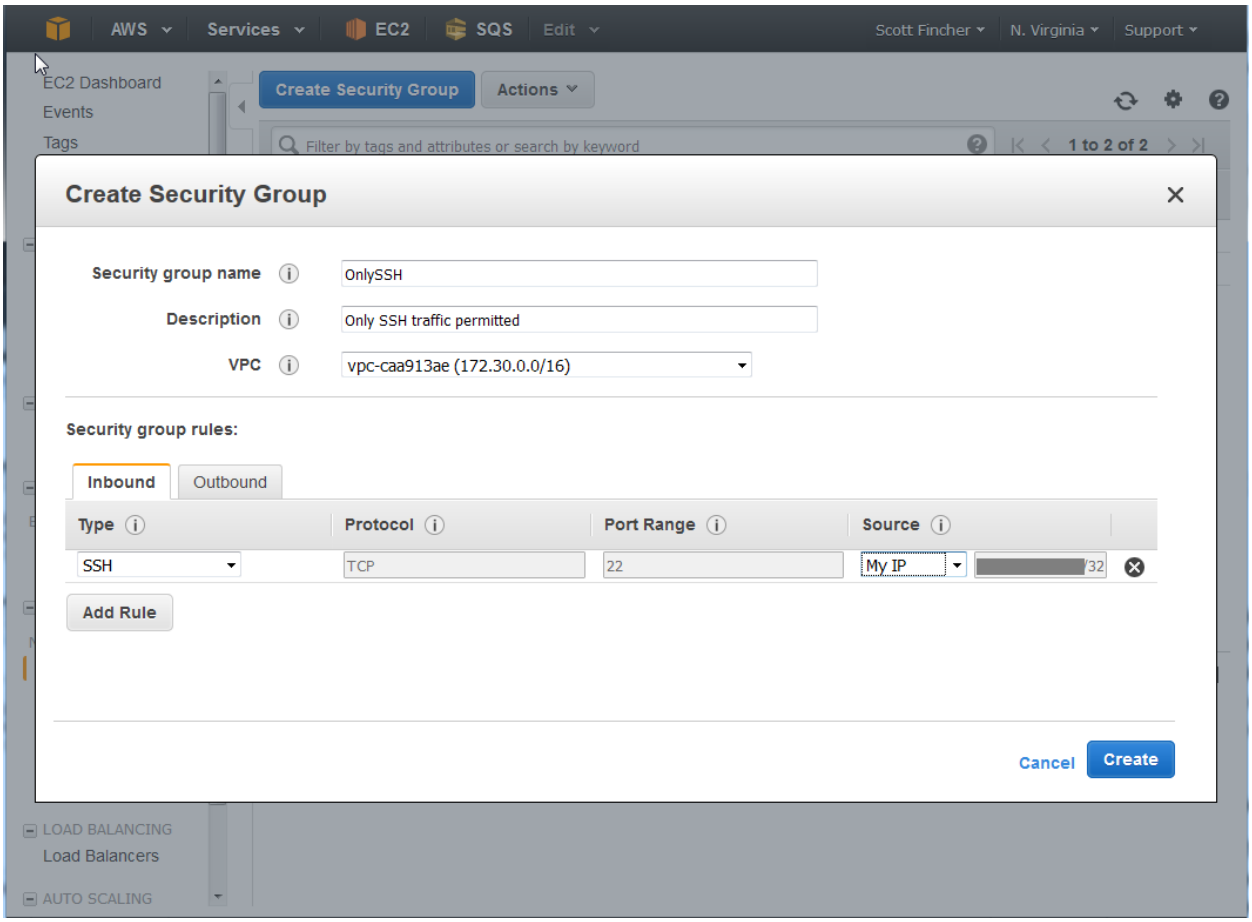
In the EC2 dashboard on the left hand side of the screen, click *Elastic IPs*, then the *Allocate New Address* button. Make sure the drop down menu reads EC2, and click *Yes, Allocate*, as pictured below. When the confirmation popup window appears, click *Close*. This will create a new IP address for you to access your instances.



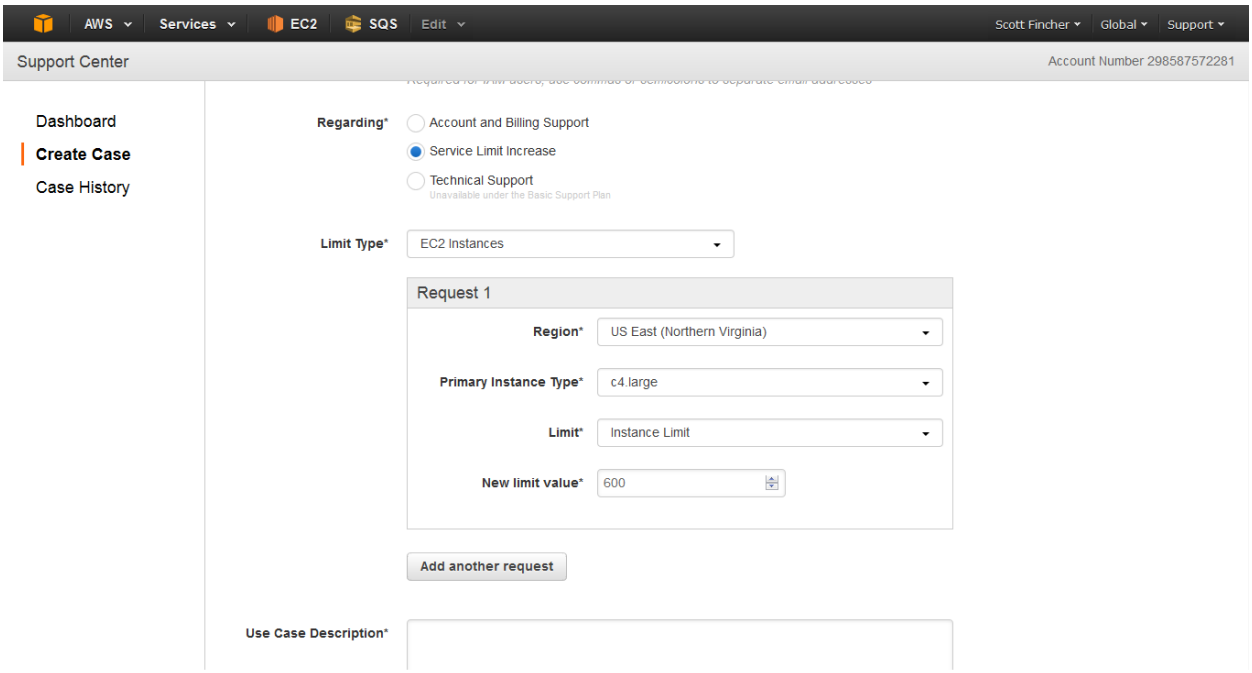
Next, click *Key Pairs* in the EC2 dashboard, and click the blue *Create Key Pair* button. Give the key a name, and click Create. The system will prompt you to download and save a .pem file to your local system. This file is used to directly login (via SSH) to instances you've created, usually during QA or troubleshooting. Without it, you will be able to start instances, but not login to them, so be sure to store it securely.



Next, click Security Groups in the EC2 Dashboard, and click the blue Create Security Group button at the top of the page. Provide a security group name and description, and select the VPC option. Under the inbound tab, click the Add Rule button, and select SSH under the Type dropdown menu. Under Source, select My IP, and the field will be automatically populated. When finished, the dialog should appear similar to the figure below. Click the blue Create button to finish. These settings will ensure that only someone using your IP address, with the SSH protocol, will be able to connect to the instances you create later.



Finally, click *Limits* in the EC2 Dashboard. There are a few limit increase requests to make here, in order to run a sufficient number of simultaneous instances. EPA generally selects the c4.large instance type, since it provides sufficient computational power and RAM to complete batches in a few days' time[8]. As an example, scroll down under the Instance Limits subheading, find the "Running On-Demand c4.large instances" item, and click *Request Limit Increase*. This will open a new browser window under the AWS Support Dashboard. Scroll down the page and fill in the drop down menus as needed. An example of a request for an increase in the limit of c4.large instances to 600 is shown below.



In addition to the above, you will probably want to increase both Provisioned IOPS (SSD) volume storage and General Purpose (SSD) volume storage, listed under the EBS Limits subheading, from 20 TiB to 50 TiB, since files associated with SMOKE-MOVES input generation can be quite large. You may wish to make limit increase requests for other type of instances to meet your specific needs.

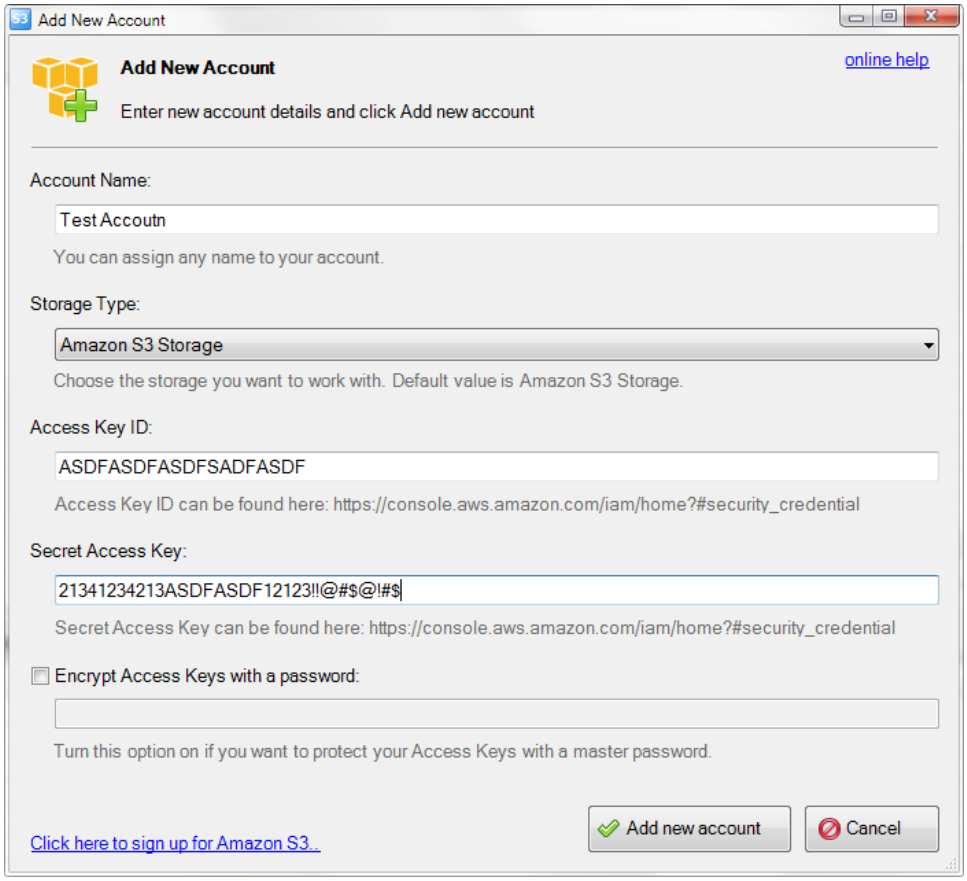
Finally, you will need access to a specific Amazon Machine Image (AMI) to run properly run MOVES in a cloud environment. An AMI is analogous to the operating system on your local machine. As mentioned in the previous section, you will also need .jar files containing MOVES code, its associated database, and post-processing code. All of these items will need to be obtained from EPA by request.

[8] For more information on AWS instance types, including available CPUs, RAM, and costs, see https://aws.amazon.com/ec2/pricing/

4.2 Uploading and Adding Jobs

Before any files can be uploaded to your AWS account, you need to create three buckets for storing data – one for the MOVES code, one for the database, and one for the batch input and output files. Although this can be done via the AWS web interface, it is easier to accomplish using the freely available S3 Browser[9]. Several screenshots in the section that follows will display the S3 Browser GUI.

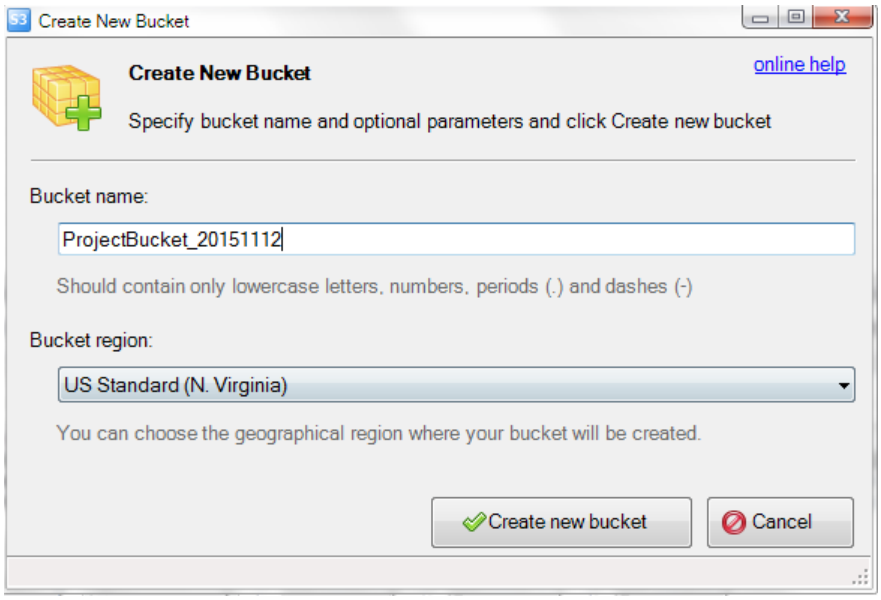
To create the buckets, open the S3 Browser and first add your AWS account. Under the *Accounts* menu, click *Add New Account*, populate the fields shown, and click the *Add New Account* button.



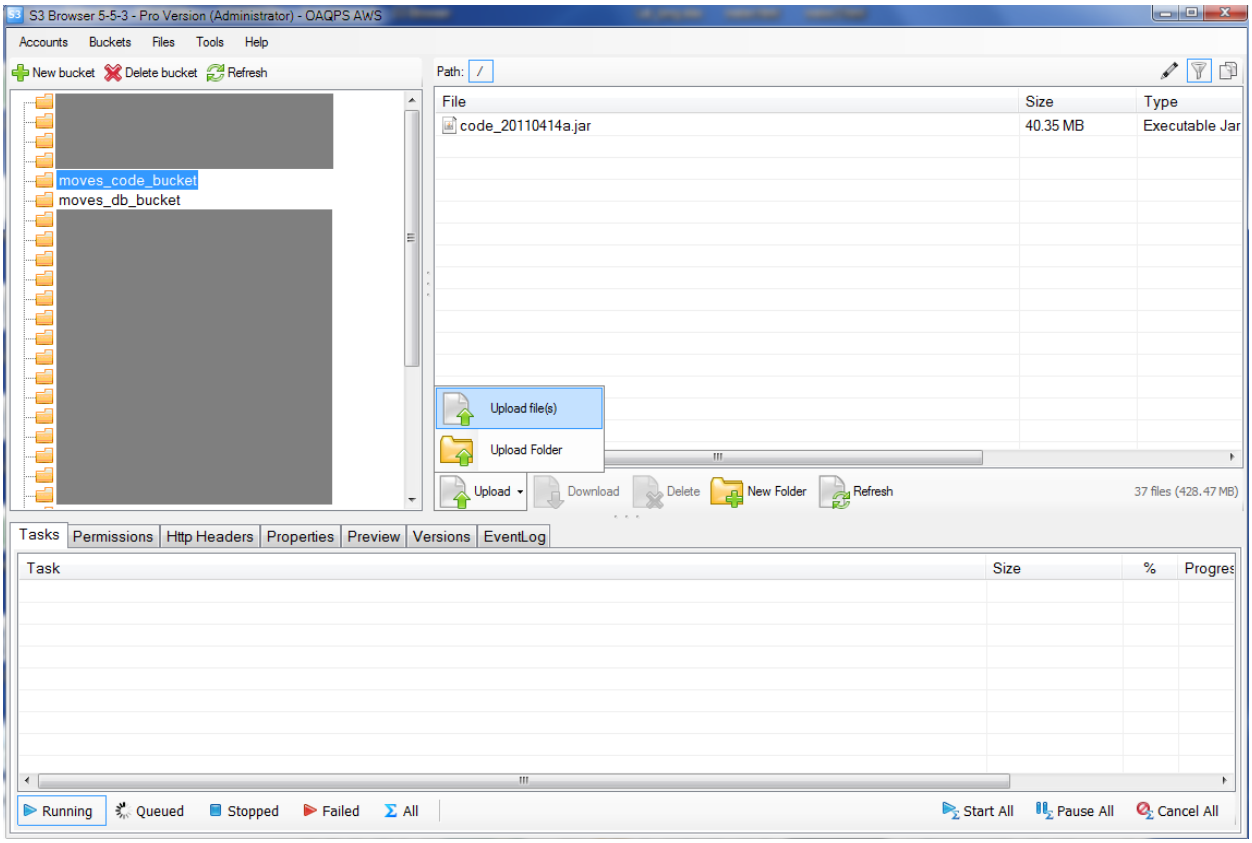
Next, click the New Bucket button and in the prompt that follows, give your bucket a name that matches the project name specified in the Perl scripts earlier, and also ensure that the region matches the one where you created your account. Do this again for your MOVES code

[9] Download from http://s3browser.com/download.php

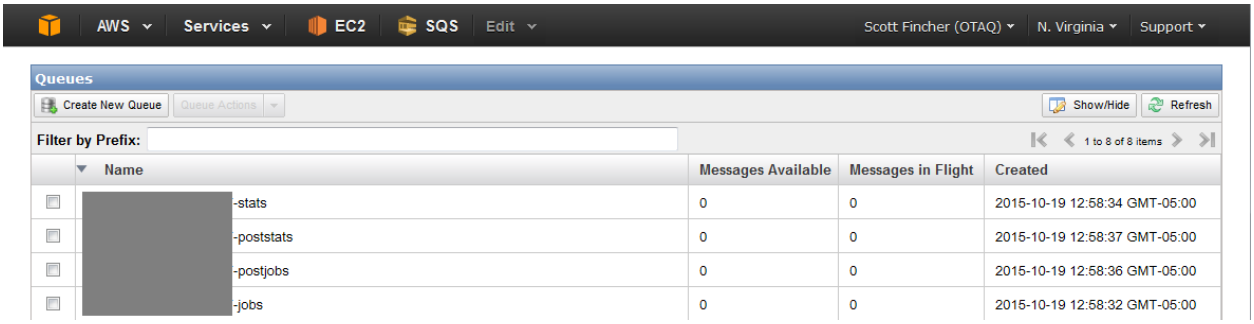
bucket and database bucket, again being careful to provide names that match those in the previous Perl scripts.



Now upload the MOVES code and database jars obtained from EPA to your newly created buckets via the S3 Browser GUI.



Having created buckets in the Amazon account, you are almost ready to begin uploading batches. Before you can do that, however, you must create queues using Amazon's Simple Query Service (SQS). These queues allow you to pass commands to the Amazon environment, including upload, download, execution, and several other operations. Do this by navigating to the \0scripts directory generated by CreateBatchFiles.plx on your local machine, and double clicking the CreateQueues.bat file. A DOS window will appear with status messages while your local machine communicates with Amazon. Don't close this window until the "Press Any Key to Continue" message appears, which should take less than a minute. Once created, you should be able to see the empty queues via the AWS web interface under the SQS Management Console, as presented in the example below.



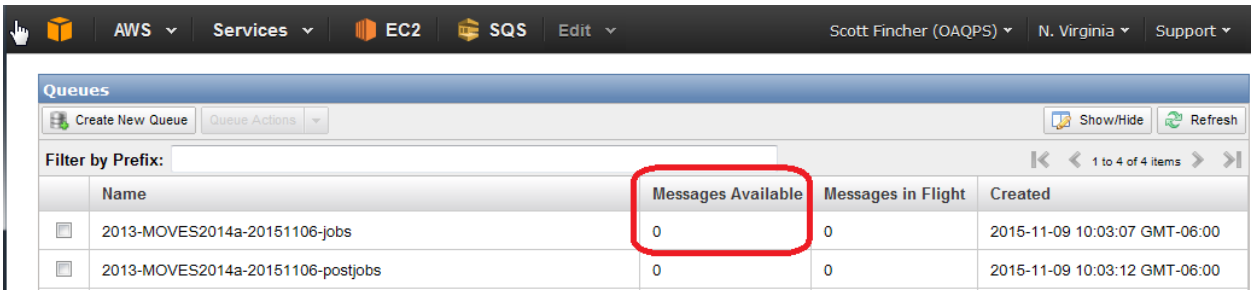
Now that the queues are created, you can upload the jobs from your local machine to the cloud. Do this by double clicking the UploadJobs.bat file in the \0scripts directory. Again, a DOS window will appear, but this operation will usually take several hours to complete. You can monitor the status of uploaded files via the log that UploadJobs.bat creates in the 0scripts directory, or by opening S3 Browser and refreshing the view of the project bucket.

Another way to monitor the progress of the upload process (as well as adding, executing, and downloading jobs later on) is to use the BatchStatus.bat file. This program takes about 15 minutes to run, and checks status files created by the Amazon batch files during execution. When complete, it creates a batchstatus.csv file that can be opened in Microsoft Excel, as in the example below. (Note how batches are usually identified by combinations of county and month, for example, 01073_1, where jobs are identified by additional RD/RV/RP text and associated temperature regime.)

A job that has been successfully uploaded will be marked as such in the appropriate column (with similar results for the other functions tracked by the program). When checking for batches or jobs that may have failed during upload, you can filter the spreadsheet for blank

values as shown. Any blank values reflect that the job has either not yet been processed, or has failed; in the latter case, more investigation is required to determine the cause of the failure.

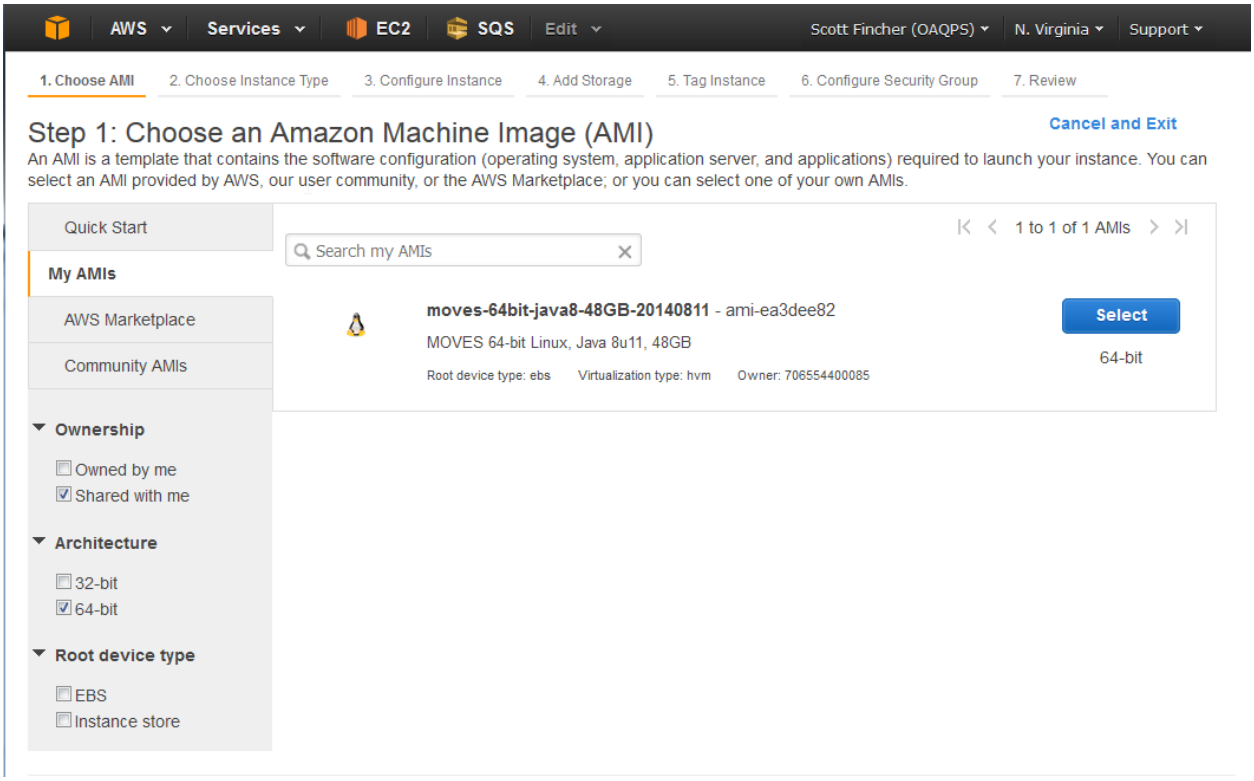
Once your batches have been successfully uploaded to Amazon S3, the next step is to add those batches to the queue for processing. Do this by navigating to the \0scripts directory and double clicking the AddJobs.bat file. This script takes a few hours to run, depending upon the number of batches you have prepared and the speed of your local computer. You can check its progress by monitoring the AddJobs.log file, or by opening the SQS management console and reviewing the Messages Available column in the jobs queue (which you can refresh in real time). Adding jobs is complete when the DOS window disappears, and the number of messages available is equal to the number of batches you have prepared. You can also use BatchStatus.bat, as described above, to monitor progress.



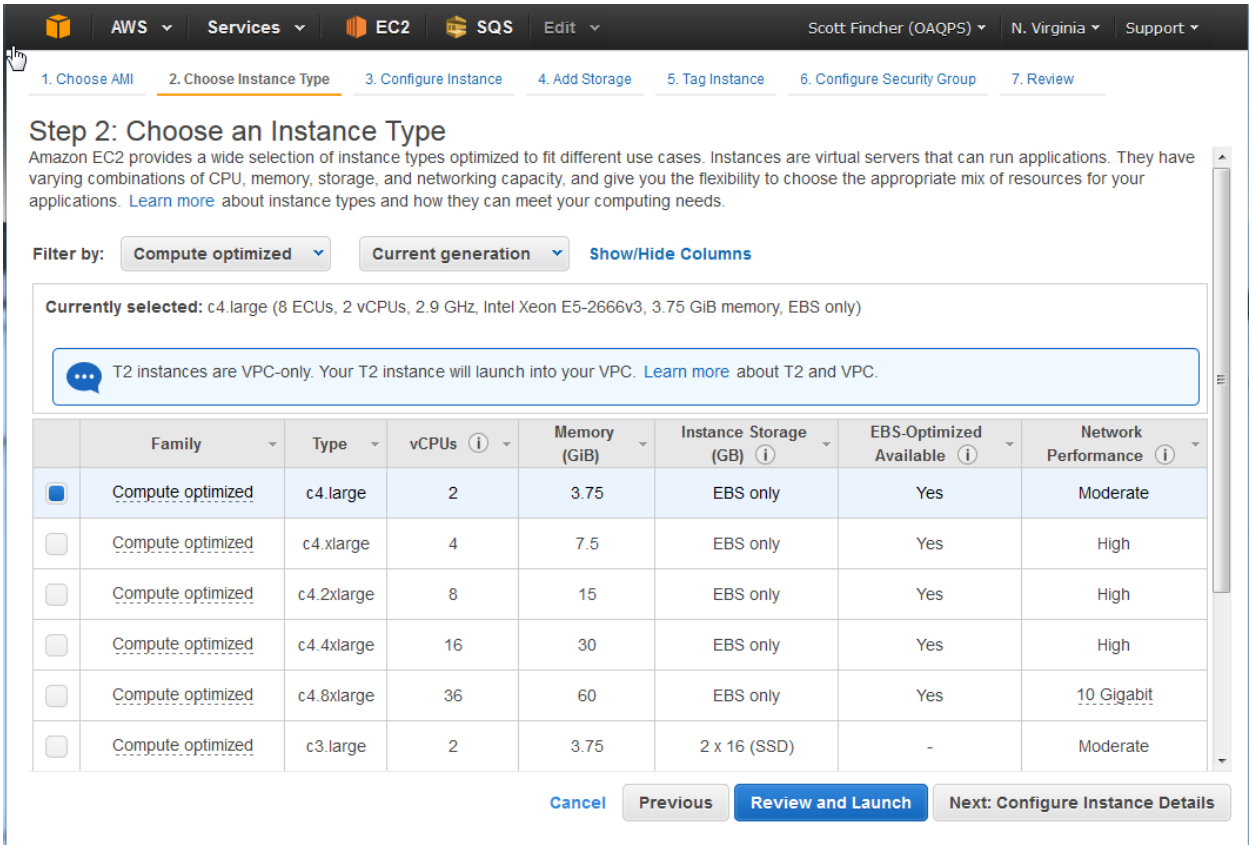
Once batches are uploaded and queued, you are ready to start Amazon instances to execute MOVES runs in the cloud environment.

4.3 Creating Instances

To process your batches, start at the EC2 management console page and click the *Create Instance* button. You will be directed to Step 1 of the instance creation page. In the left-side frame, click *My AMIs*, and toggle the checkboxes as shown below, to include Ownership: *Shared with me* and Architecture: *64-bit*. This will filter the available AMIs to the particular MOVES AMI shared with you by EPA. Click the blue *Select* button.

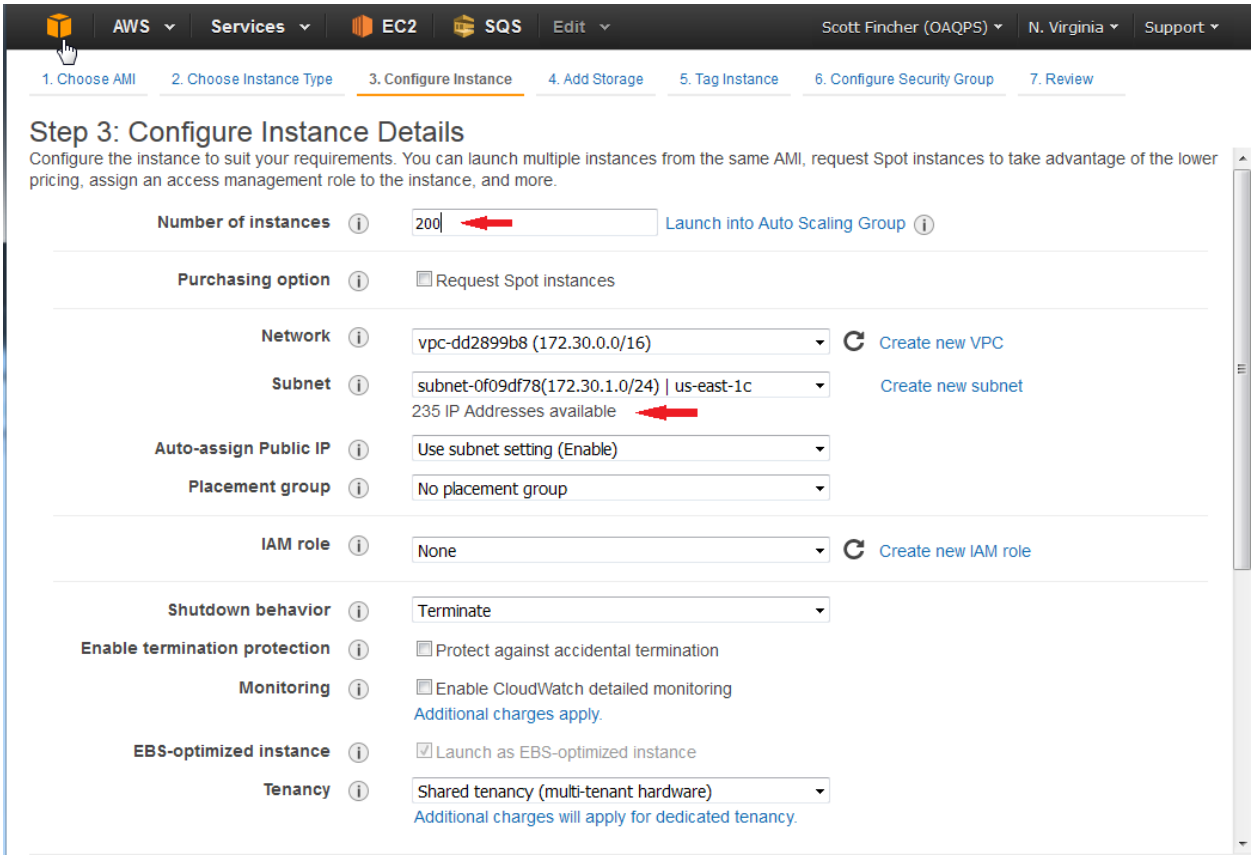


At Step 2, filter the instances by *Compute Optimized*, and click the box corresponding to the c4.large instance type. (This is for the purposes of the included example; you may of course choose another instance type as needs dictate.) Click the grey *Next: Configure Instance Details* button in the lower right corner.

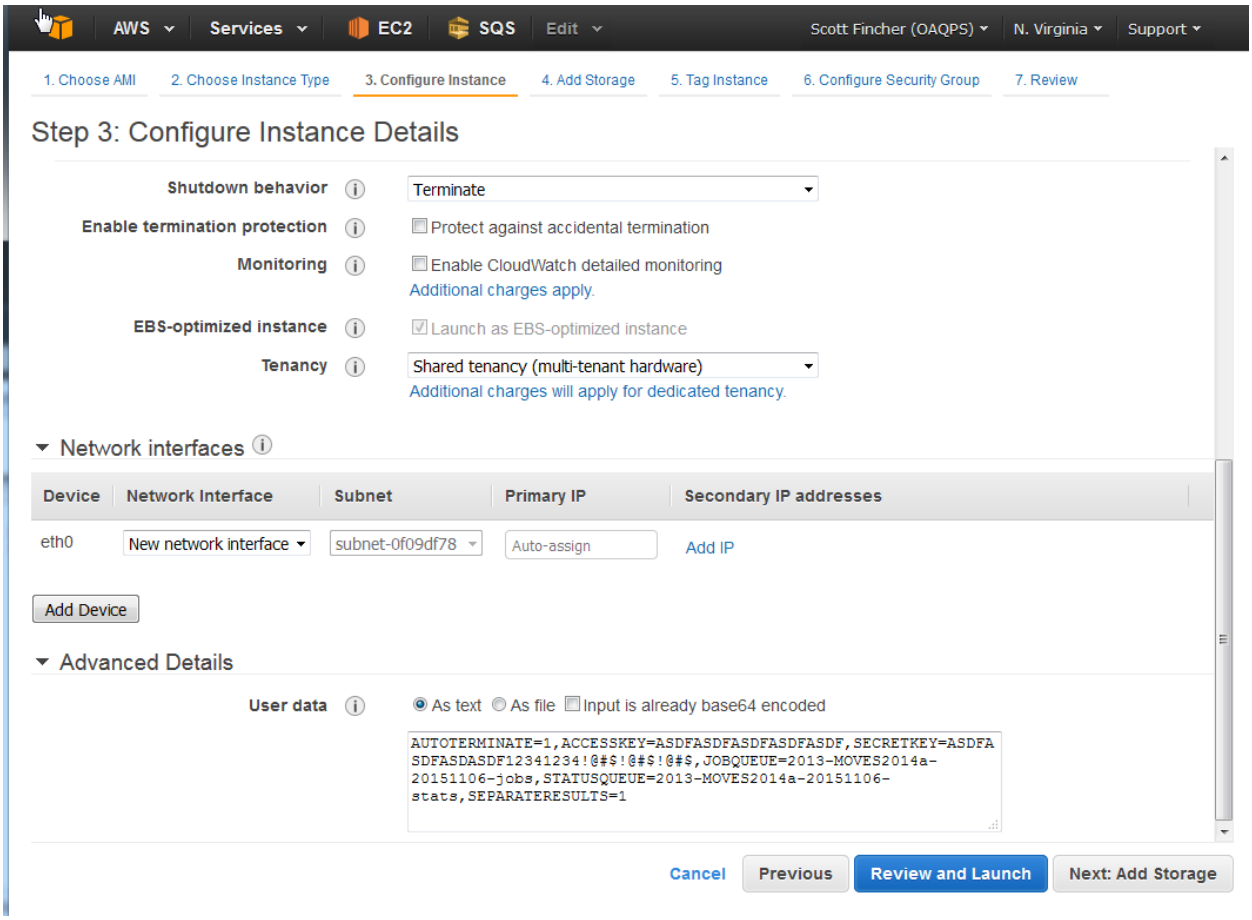


At Step 3, there are several important parameters to configure. The number of instances you can would like to execute in a subnet net at once (first red arrow) is limited by the number of IP addresses available, so your choice must be less than or equal to the limit. This number is usually 251 if no other instances are currently running; if you need to run more than 251 instances concurrently, as EPA does, you will need to repeat this process a few more times, creating groups of instances in different subnets. (Eventually, you need to generate as many instances as you have messages in the jobs queue.)

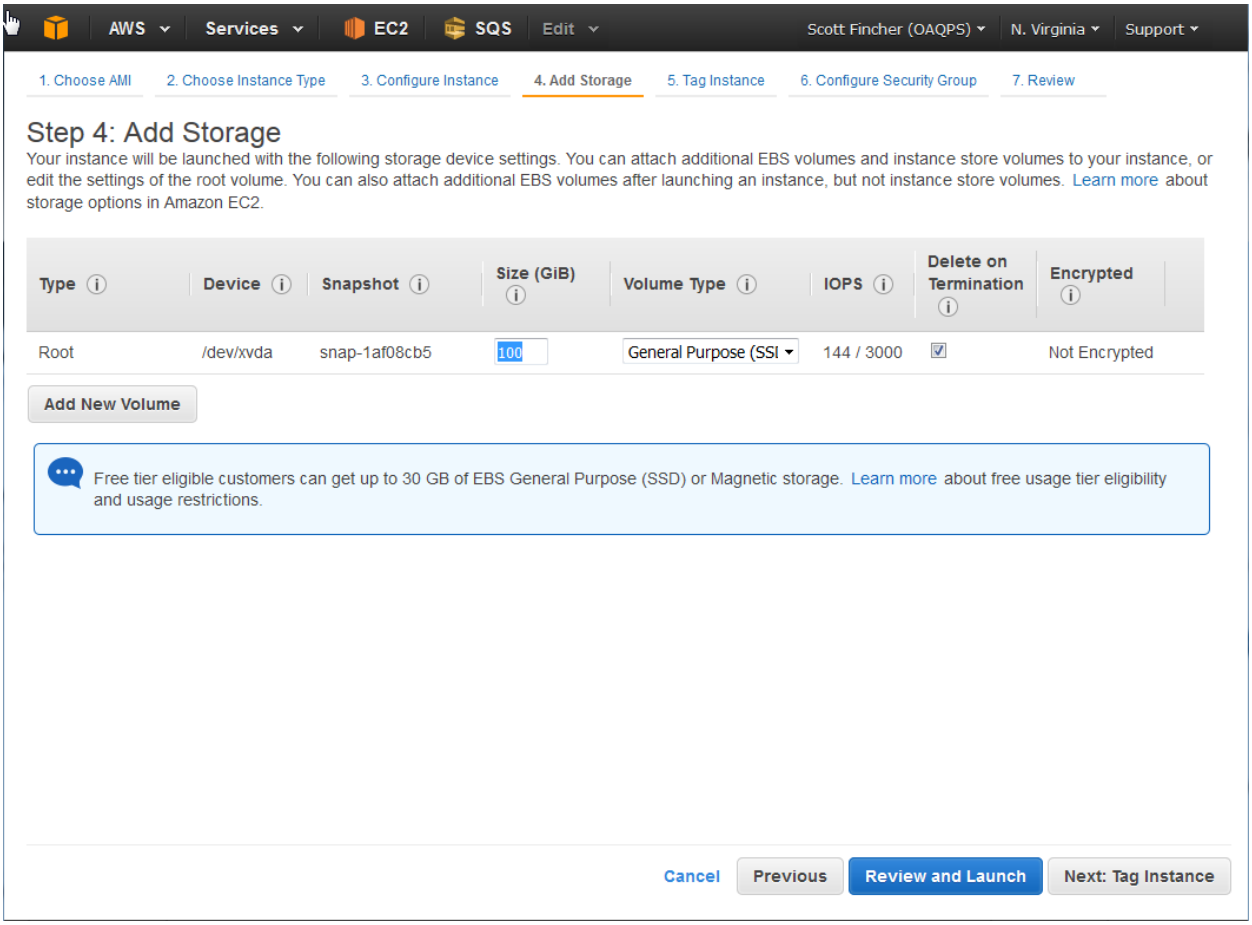
Populate the other options as shown. Many of these values are defaults, with the exception of *Shutdown Behavior*, which should be changed from "Stop" to "Terminate".



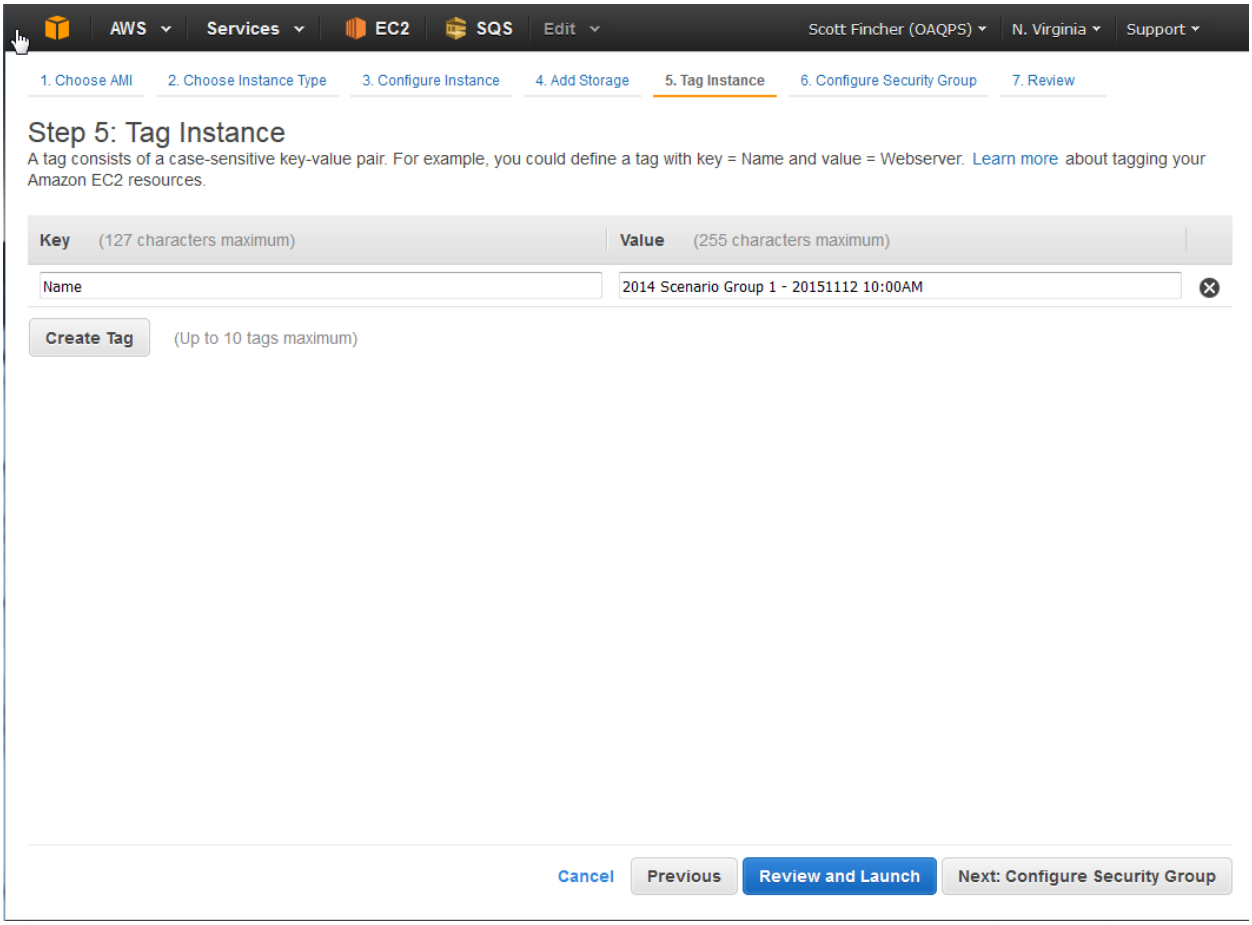
Next, scroll down to the bottom of the page and click the arrow next to *Advanced Detail* to expand its submenu. You need to paste some configuration text into the text box shown; this text is available in the \0scripts directory in the file **TextToStartInstance-MOVES.txt**. Open this file in a text editor, and copy its entire contents to the text box at the bottom of the Step 3 page. It should look similar to the figure below. Click the grey *Next: Add Storage* button.



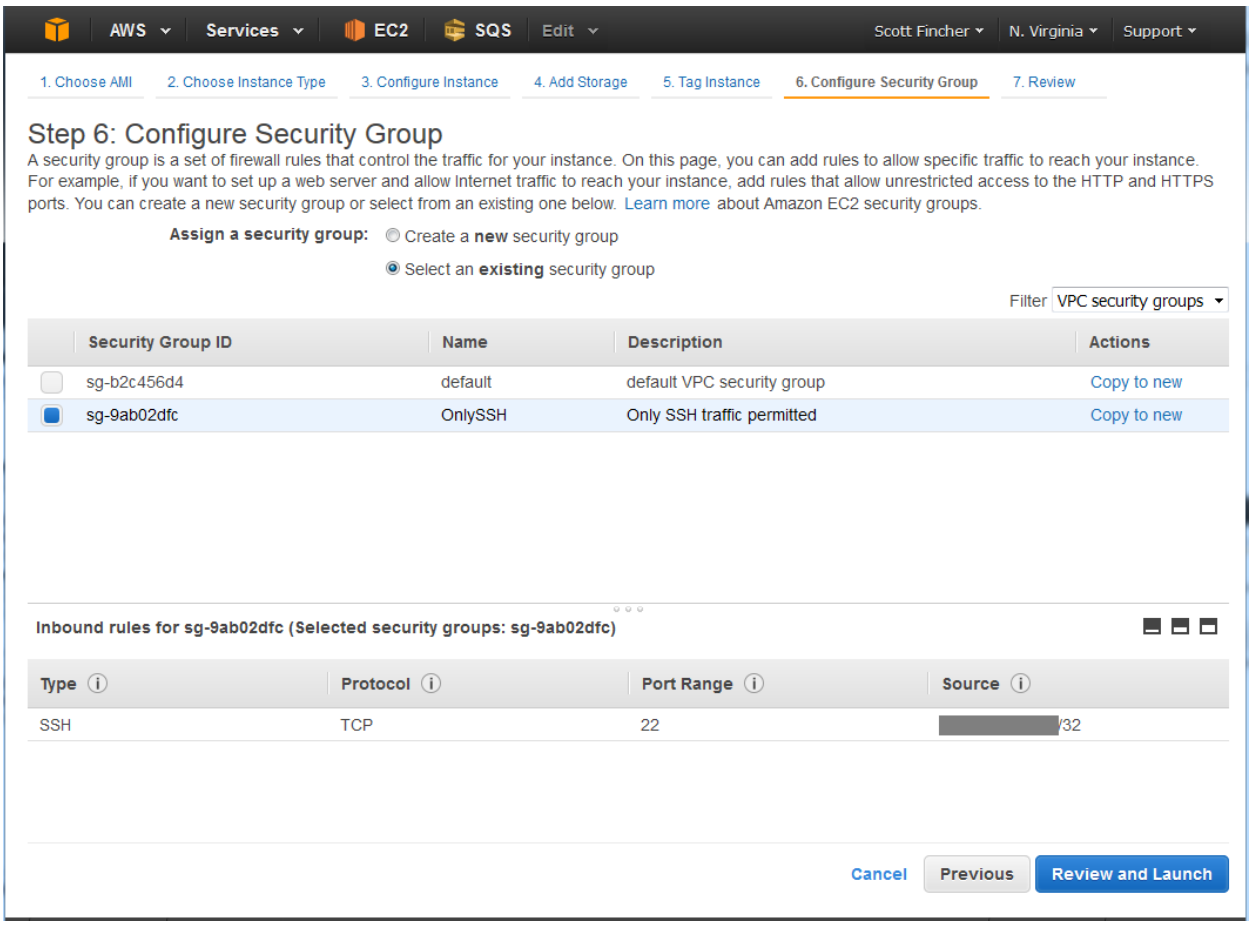
On the Step 4 page, you need to increase the default size in GiB from 48 to 100. If you don't, MOVES may get hung up during its activity generator process and fail to proceed – although your instance will not terminate, so you will continue to incur charges! Click the grey *Next: Tag Instance* button.



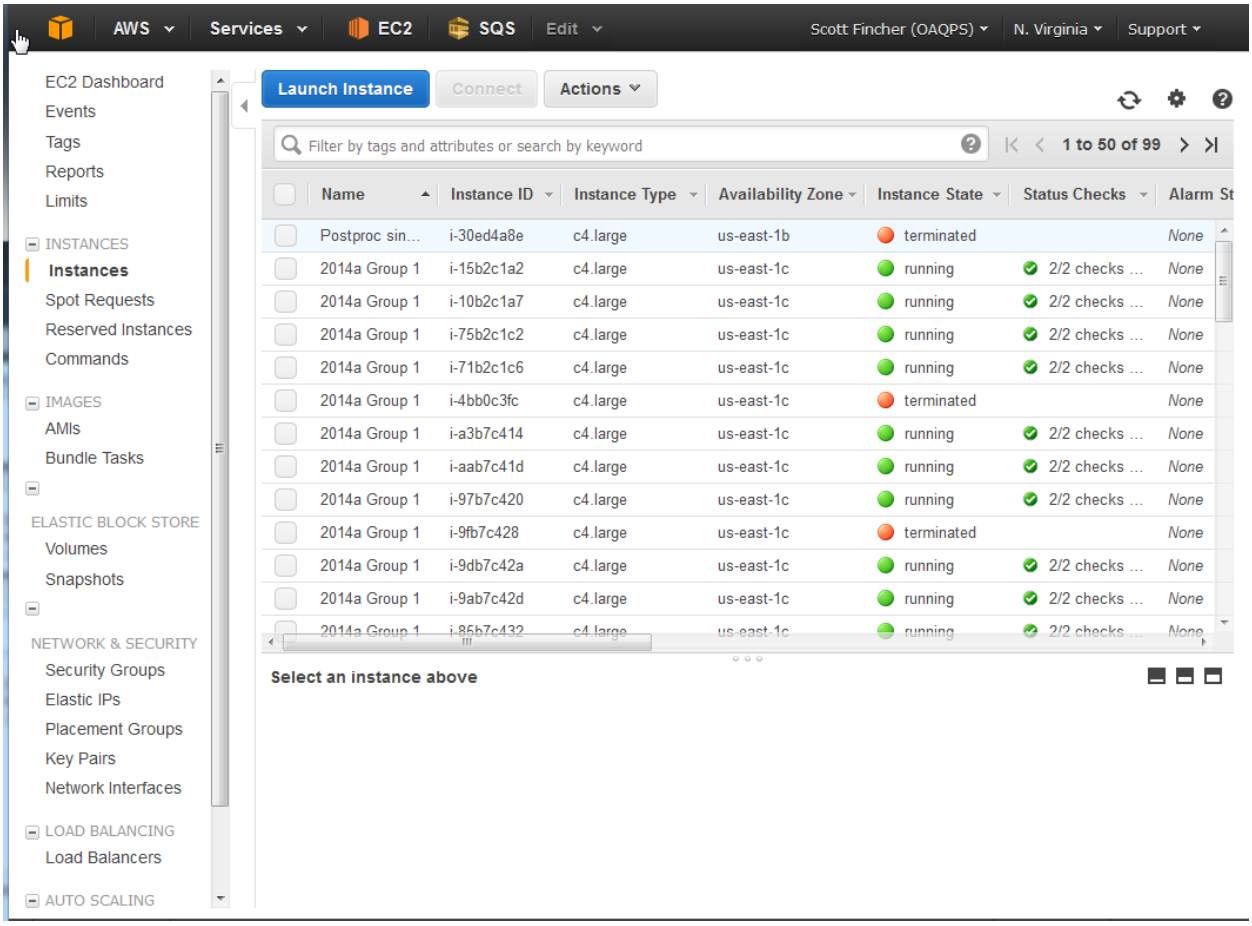
At Step 5, populate the Value field to assign a name to the instances generated. It can be useful to provide a fair amount of descriptive information here, including date and time, for the purposes of QA later. Click the grey *Next: Configure Security Group* button.



At Step 6, click the radio button next to *Select an existing security group*. The security group that you created during initial account setup will appear (in this example, OnlySSH). Click the box next to that group, and then click the blue *Review and Launch* button in the bottom right of the page.



At this point, you're almost ready to launch the instances[10]. On the Step 7 screen, review all of the choices you've made during steps 1 through 6. If you need to make changes, you can go back and do so. After you've carefully check the instance options, click the blue *Launch* button. Amazon will begin creating instances, and you will be presented with a wait screen during this process. After creation, you can return to the EC2 management console, where all of the instances and their status will be displayed. Below is an example of what to expect once instances have been running for 1-2 days. You can see that some instances are still ongoing, while a few have completed. All told, the MOVES batches typically take between 36 and 72 hours to complete using the c4.large instance type.



You can also verify, via the SQS management console, that messages in the queue are being "picked up" by your newly created instances. If you repeatedly click the *Refresh* button on

[10] If this is the first time you're starting instances associated with a particular project or scenario, you may want to consider launching only a single instance to see whether it will execute successfully, before executing all available batches. Doing so can avoid excess EC2 charges.

this page, you should be able to see the “Messages Available” transition to “Messages in Flight” in real time; the number of messages in the jobs queue should decrease to 0, while the number of messages in the stats queue increases rapidly.

4.4 Downloading Initial Results and QA

When each batch is processing in the cloud environment, there are multiple RD, RV, and RP runspecs that are executed in sequence. This means that although a particular instance may not have finished yet, after several hours it will begin to generate output files. You can monitor the status of your batches by viewing outputs generated to date using the S3 Browser. You can also run the **DownloadResults.bat** file in the \0scripts folder, which will download the results*.jar files associated with each job, which contain compressed MOVES logs. After running DownloadResults, which can take several hours in some cases, BatchStatus can then be run to view how many jobs in each batches have been complete. In fact, EPA recommends running the **RepeatDownloadResults.bat** program a few hours after starting instances; this program repeats the download process 100 times, and can help to save time by accessing job outputs as they complete, rather than waiting to download the results all at once.

After the instances have completed execution and results have been downloaded, run BatchStatus a final time to verify that all batches are complete. If any are not, make note of them. Then, open a DOS command window, navigate to the \0scripts directory and execute the **QA1MovesOslogLogqueue.plx** perl script at the command prompt. This script takes a few minutes to run. When it completes, read the output file it generates into Excel, and use the Text to Columns wizard to split the text based on spaces. EPA recommends checking that the total count of jobs is correct, and that the number of generated bundles is equal to retrieved bundles for all jobs. If not, you will need to check your inputs to determine where the problem lies. (Note that it is also possible for batches to fail due to Amazon errors, and not because of errors in the inputs themselves!)

Next, execute the **QA2MovesOslogLogqueue.plx** perl script at the command prompt. Wait a few minutes for it to complete, and review its output. This script searches for the string "RUN_ERROR:" in the MOVES logs, but excludes the string "RUN_ERROR: Warning:", and is intended to detect failures that don't cause a job to stop. MOVES output can pass QA1, but fail QA2, so it is important to run both.

At this point, if any batches have failed, you have likely detected them by use of BatchStatus or either one of the QA scripts above. These batches will need to be rerun, but many times a batch will not fail completely; rather, a few jobs will succeed before the instance

encounters a problem. In order rerun only the jobs that have failed, and not the entire batch, you can use the **Readdjobs.bat** file. This file contains by default all batches, but you only want to re-add those batches that have failed. Open the file in a text editor, and either "comment out" lines corresponding to batches you don't wish to re-run by adding the text "rem " at the beginning of the line, or delete the line entirely. Save the Readdjobs with a slightly different name so as not to overwrite the original, and execute your newly edited batch file by double clicking it. At this point, you will repeat the process described above of monitoring jobs added to the SQS queue, adding instances to process them, downloading their logs and otherwise monitoring their progress.

4.5 Post-processing MOVES results into SMOKE-MOVES format

Once all of the MOVES batches are complete and have been quality assured, the final step is to post-process MOVES outputs into a format useable by SMOKE-MOVES. This process is similar to that described above in sections 4.2 and 4.3, so it will not be repeated in detail here. Rather, significant differences in the processing will be highlighted below.

Post-processing consists of the following steps:

- Use AddPostProcess.bat to add messages to the SQS post queue, which will use the MOVES post-processing code (provided by EPA in a .jar file, which you uploaded previously). No uploading of data is required here, since the program will be using MOVES output already present in the S3 project bucket.

- Create and launch instances as before, using similar options. One important exception is that you should use the text from the **TextToStartInstances-post.txt** file during Step 3 of the instance configuration. These batches typically take only about 30-60 minutes to complete.

- If any post-processing batches fail, edit and run Readdpostproc.bat as necessary.

As the post-processing batches complete, you will notice output files in the S3 Browser. These files are usually between 100 and 300 MB in size. The outputs can be downloaded automatically using Downloadpostresults.bat, or manually using the S3 browser.

5.0 Troubleshooting

This section presents additional information that may be helpful when troubleshooting instances that have failed or are non-responsive.

5.1 Tips and Tricks

There are a few other items to keep in mind as you work through the modeling process.

- Amazon has set a hard limit of 100 S3 buckets per account. To create more, you will have to delete existing buckets.
- If you don't see your instances or any other settings when logging in from a new system, make sure you have chosen the proper region (e.g., N. Virginia).
- Be careful not to run multiple Perl scripts or batch files at once, as this can cause undesirable behavior. For example, one script may log out of your account before the other script completes, causing errors. Proceed systematically through execution of scripts and batch files.
- Become familiar with the format of the batch files. Once you are comfortable with how they work, it is often useful to edit them during the QA process to allow for uploading, adding, and running single batches.

Note that it is not uncommon for AWS instances to fail. In fact, you can expect a failure rate of about 5% in general. There are a number of possible reasons for these failures, including the following:

- Sometimes batches may not be picked up from the SQS queue at all. In this case, create a new instance for each job remaining.
- Some batches may fail midway through. In this case, use the Readdjobs batch file to process only the jobs that remain instead of starting the entire batch over from the beginning.
- Instances may immediately fail without any warning or error messages. If this happens, check to see if your instance and/or storage limits have been exceeded.

- Amazon may not have enough capacity for the instance type you have selected. You can either wait for Amazon to free up additional capacity, try a different instance type, or create instances in a different subnet.

- On occasion, during post-processing, instances may complete their calculations but fail to write out data, and "freeze up" without terminating. One work-around for this problem is to create a new bucket specifically for post processing, copy the MOVES database output to that bucket, edit your batch files to point to the new bucket, and re-run the post-processing there.

5.2 Lifecycle

The following is taken directly from Section 5.2 of Faler's *Running MOVES on Amazon*, and presents the sequential lifecycle of a project in the cloud environment. It's a useful summary of

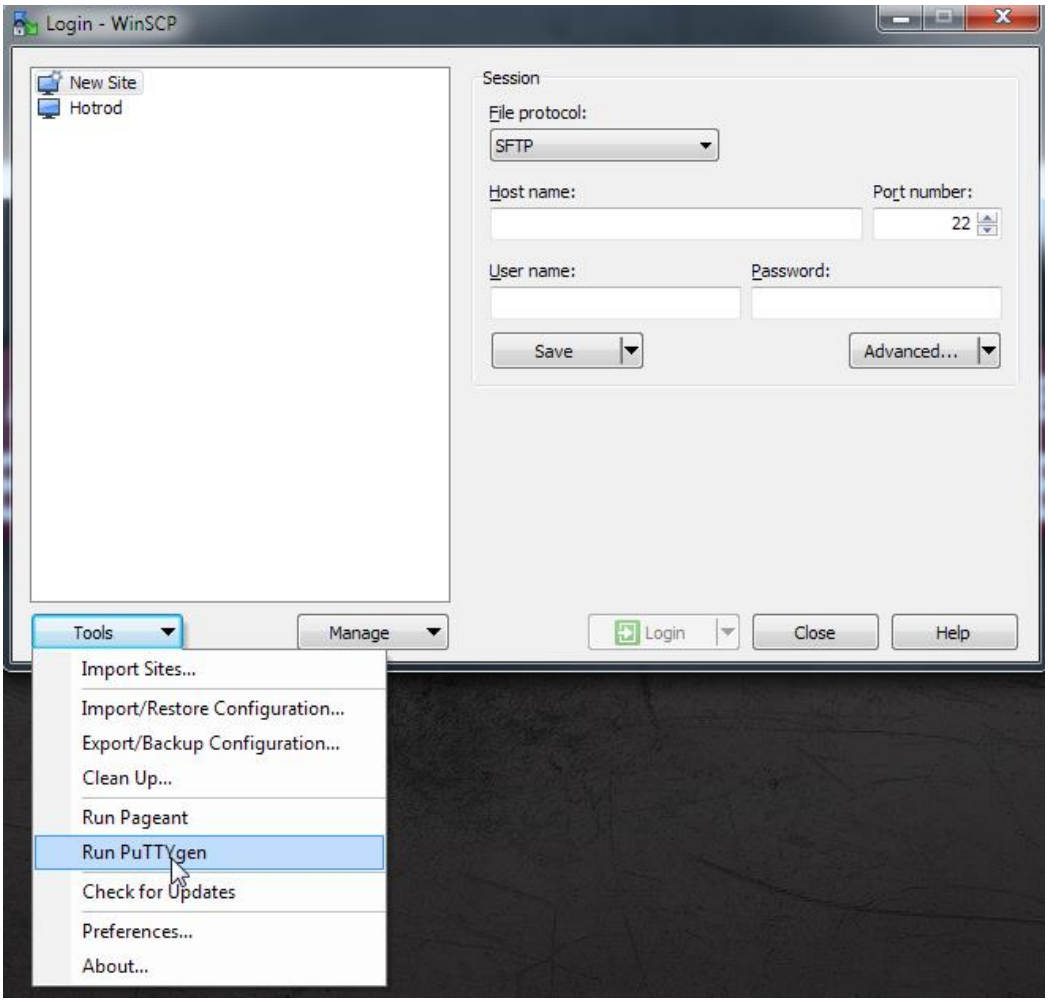
> The overall steps for processing MOVES jobs on Amazon that use post processing are:
>
> - An Amazon SQS queue is created to hold job processing commands.
> - An Amazon SQS queue is created to hold status messages.
> - MOVES code is placed into a JAR file.
> - The MOVES code JAR is placed into an Amazon S3 bucket.
> - A MOVES default database is placed into a JAR file.
> - The default database JAR file is placed into an Amazon S3 bucket.
> - JAR files are created for each job.
>   - A JAR file is created for each job. This file contains the job's runspec (.mrs file) and input databases, if any.
>   - A JAR file is created for the batch-level input databases, if any.
>   - A JAR file is created for the scenario-level input databases, if any.
> - The JAR files are uploaded to a single Amazon S3 bucket.
> - After all JAR files for jobs in a batch have been uploaded, a command to process all jobs in a batch is placed into the command queue.
> - One or more Amazon EC2 instances are started to process the commands. These instances are given the SEPARATERESULTS=1 flag in their instance data.
> - The status queue is polled for messages originating from the EC2 instances.
> - Job result JAR files are downloaded from an Amazon S3 bucket. These JAR files contain only the log files.
> - Result JAR files and job JAR files are deleted from the bucket. Database result JAR files remain undownloaded in the bucket.
> - The result JAR file's contents are extracted, including only log files.
> - Operating system log files are duplicated and placed into the batch's logqueue directory for automated scanning.
> - Amazon EC2 instances shutdown automatically after processing all jobs in a batch.
> - An Amazon SQS queue is created to hold post processing commands.
> - An Amazon SQS queue is created to hold post processing status messages.
> - Post processing code and required databases are placed into a JAR file.
> - The post processing code JAR is placed into an Amazon S3 bucket.
> - A command to post process all jobs in a batch is placed into the post processing command queue.
> - One or more Amazon EC2 instances are started to process the commands. These instances are given the JOBCOMMAND=batchpostprocess flag in their instance data.
> - The status queue is polled for messages originating from the EC2 instances.
> - Batch result JAR files are downloaded from an Amazon S3 bucket. These JAR files contain batch-level post processing results and log files.
> - No result JAR files are deleted from the bucket.
> - The result JAR file's contents are extracted.
> - Operating system log files are duplicated and placed into the batch's logqueue directory for automated scanning.
> - Amazon EC2 instances shutdown automatically after processing all jobs in a batch.

the entire process that can be used as a quick reference.

5.3 Logging into Instances

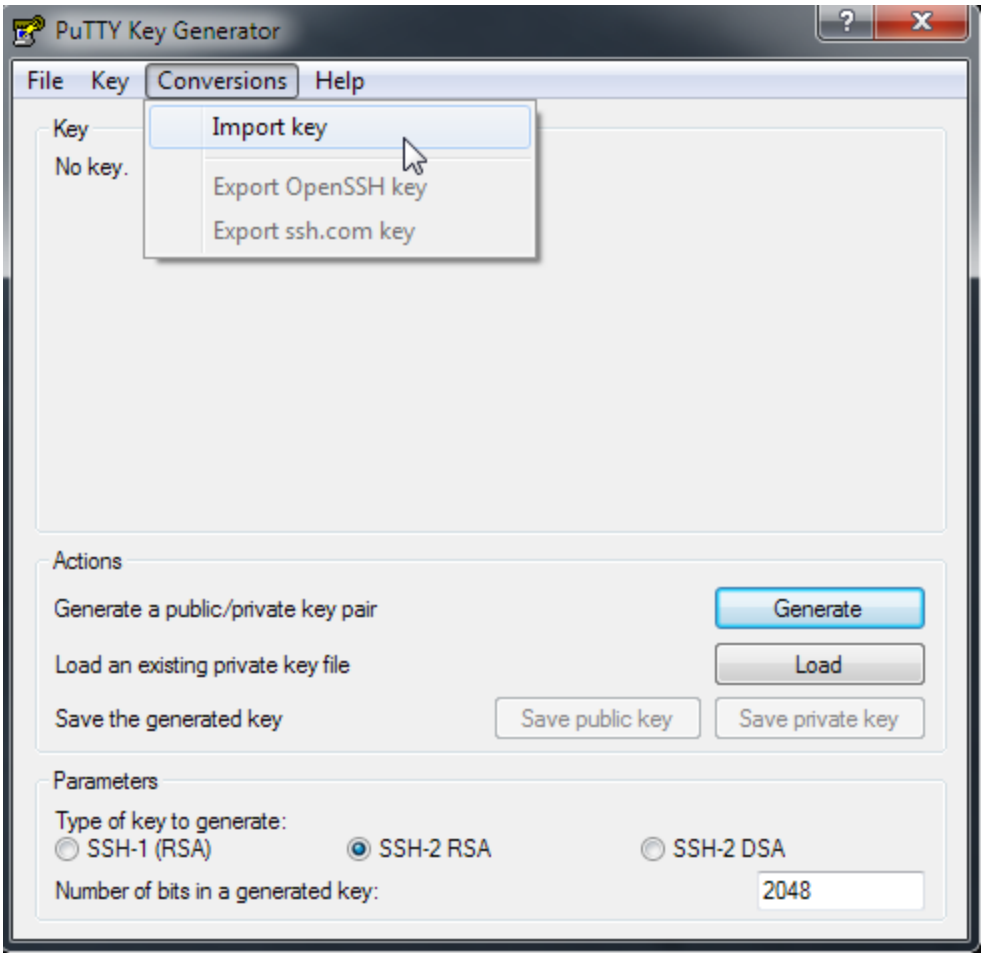
Sometimes the only way to diagnose problems with a particular instance is to login to it directly. This can be done by using the free WinSCP[11] software package, which allows you to connect to an Amazon instance directly using the private key credentials (in .pem format) created earlier.

Before you can access an instance, you must first import your credentials into WinSCP. Download, install and open WinSCP. At the bottom right, click the *Tools* button and Select *Run PuttyGen*.

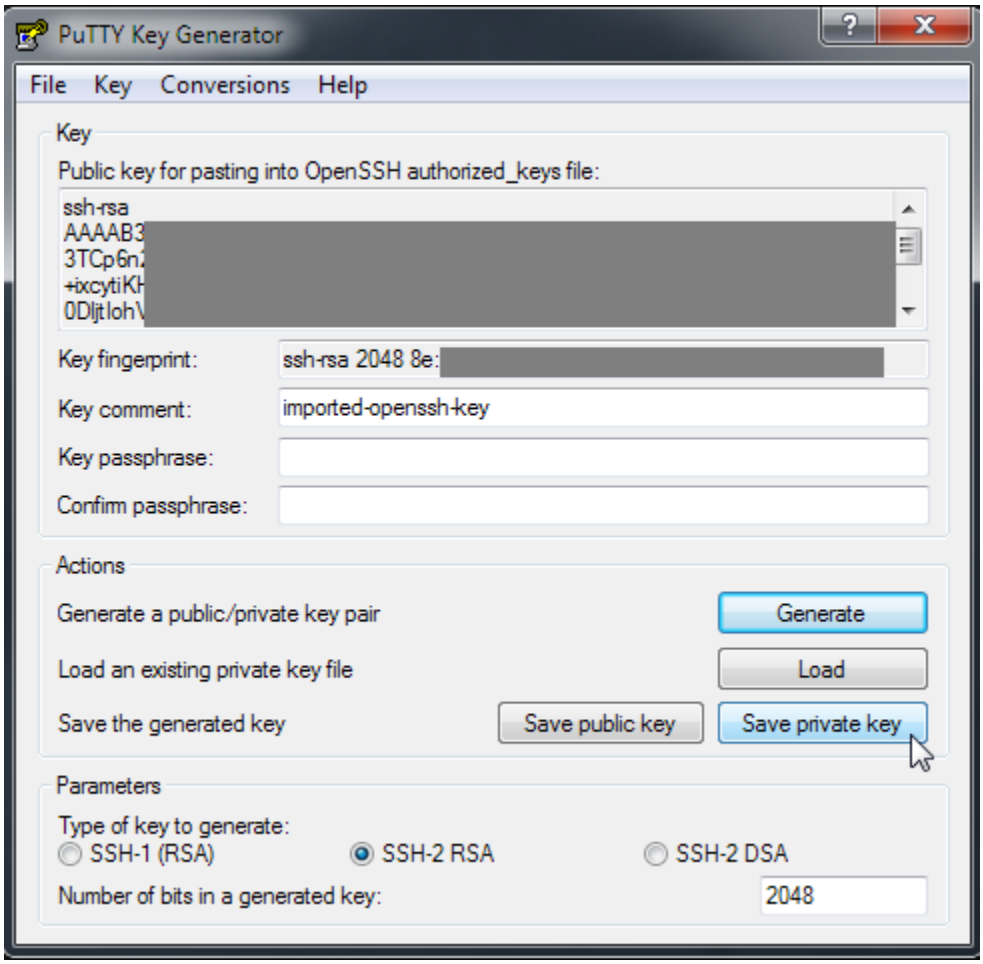


In the new window that appears click *Conversions* and select *Import Key*. You will be presented with a Windows open file dialog; navigate to the .pem file you created earlier during account setup and click *Open*.

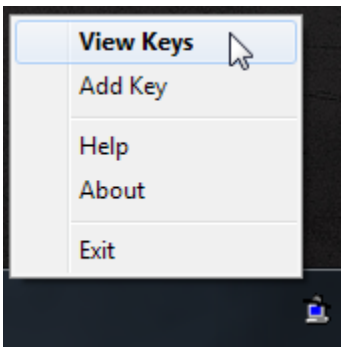
[11] Available at https://winscp.net/eng/download.php. Current version is 5.7.6.


PuTTY Key Generator
File
Key
Conversions
Help
Import key
Export OpenSSH key
Export ssh.com key
Key
No key.
Actions
Generate a public/private key pair
Generate
Load an existing private key file
Load
Save the generated key
Save public key
Save private key
Parameters
Type of key to generate:
SSH-1 (RSA)
SSH-2 RSA
SSH-2 DSA
Number of bits in a generated key:
2048

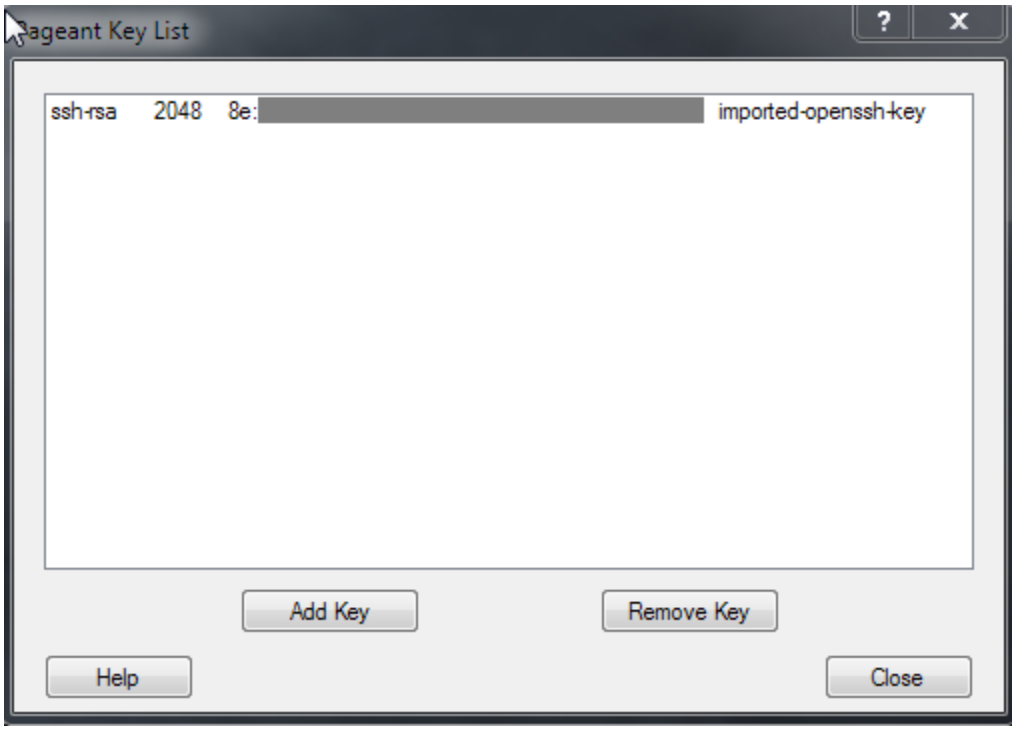
The window will populate with information on your key. Click *Save Private Key*, give the key a name in the file dialog, and close the window.



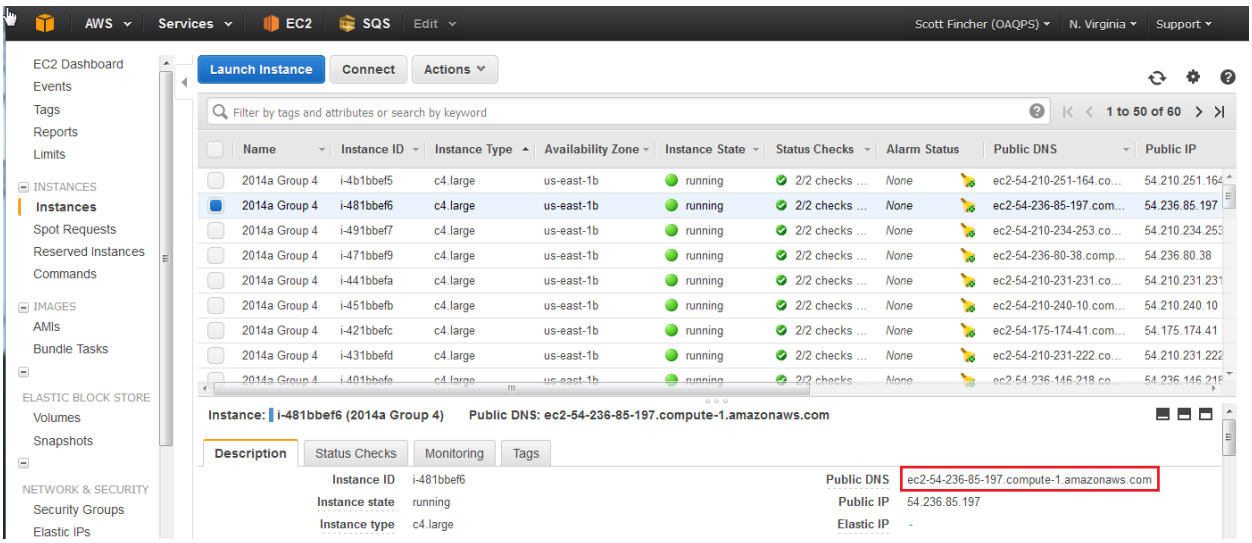
Next, at the main WinSCP window, again click the *Tools* button, but this time select *Run Pageant*. A small blue computer icon will appear in your system tray. Right click this icon and select *View Keys*.



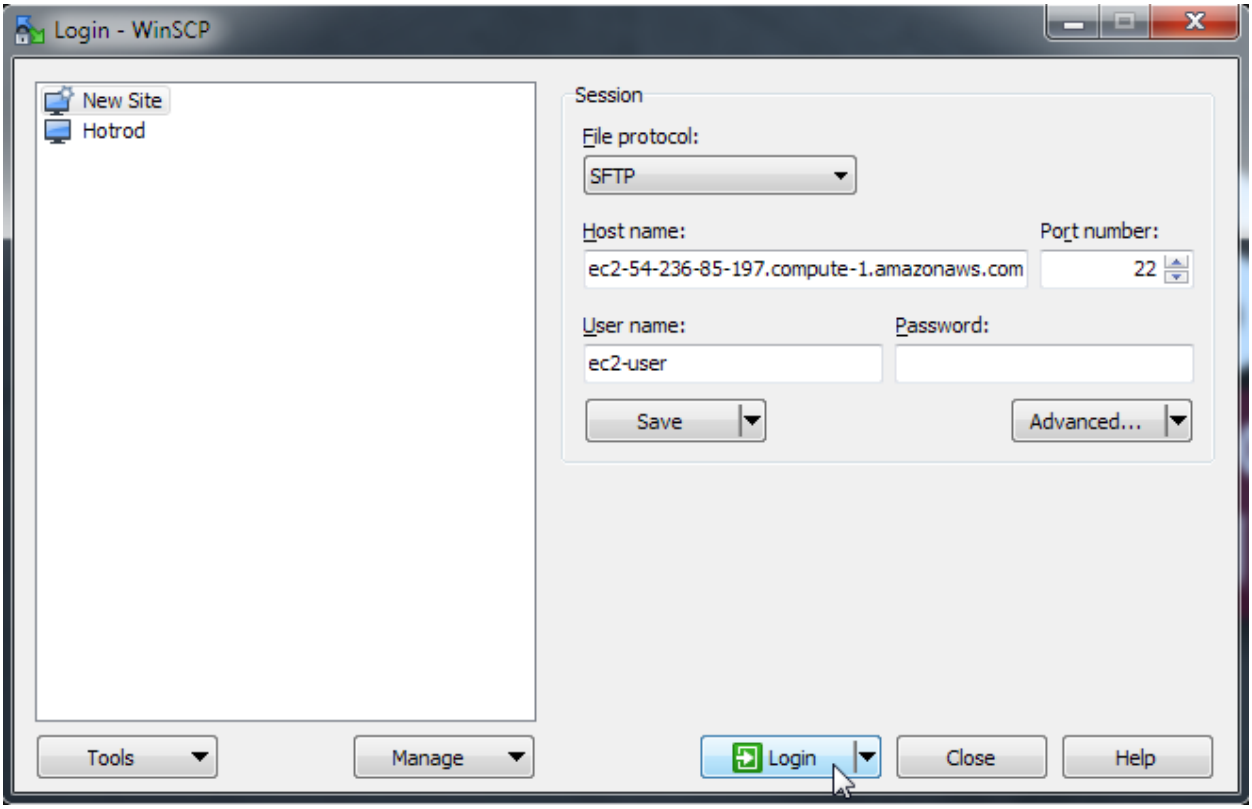
At the next window, click the Add Key button and navigate to the .ppk file you just created in the previous step. You should see a screen similar to this one.



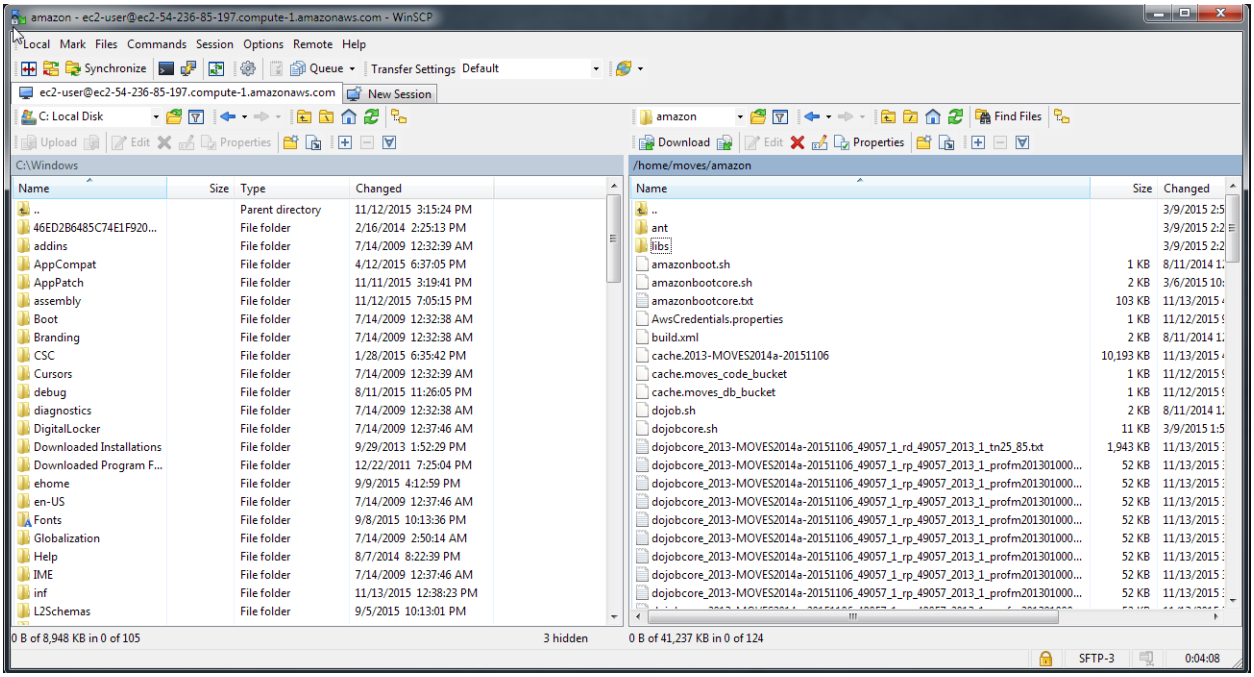
Having imported your credentials, you can now prepare to login to an Amazon instance. To determine the IP address of the instance of interest, open the EC2 management console and click the blue button next to the chosen instance. Descriptive information about the instance will appear; find the Public DNS value, highlight it, and copy it.



Now return to the main WinSCP window and paste the public DNS into the *Host Name* field. For the *User Name*, enter "ec2-user". No password is required here since you have already provided credentials via private key. Click *Login*.



When prompted to add an unknown server's host key to a cache, click yes. Next you'll be presented with an FTP environment, with your local machine on the left side of the screen and the Amazon instance on the right. Use the icons to navigate to the /home/moves/amazon directory in the instance as shown.



From here, you can download several different log files to assist you in diagnosing instance failures by double clicking on their file names. Logs of interest include:

- **amazonbootcore.txt**, which lists jobs that have been executed
- **dojobcore** files, which are logs of MOVES progress
- **movesamazon.log**, which lists files retrieved and stored by the instance
- **toplog.txt**, which is a periodic dump of the Linux top command, which displays CPU usage, memory usage, and running processes with the most recent at the bottom.

From this point forward, you're on your own. Please review the electronic appendix carefully for further details and example files. Good luck!

APPENDIX A: Scripts and Examples

Provided electronically