Introduction to Amazon Web Services

Corinne Jones*

April 15, 2019

1 Introduction

These notes provide an introduction to Amazon Web Services (AWS). While there are many things you can do with AWS, I will focus on two services, called EC2 and S3, and show how we can use these to perform a computationally-intensive machine learning task: Running large-scale nearest neighbor computations.

1.1 What is AWS?

Amazon Web Services is a cloud computing provider. That is, it provides on-demand access to computing infrastructure (networks, servers, storage, etc.). There isn't enough time to discuss all of its products, but some important use cases include:

- Renting virtual computers you can run code on
- Building and training machine learning models using canned routines
- Storing and backing up (potentially massive) amounts of data
- Creating a database and running queries on it
- Hosting websites
- Crowd-sourcing human intelligence tasks (tasks computers currently can't perform)

AWS is the most popular cloud computing provider today and is used by companies such as Adobe, Airbnb, Lyft, and Slack. Competitors to AWS include Google Cloud and Microsoft Azure.

1.2 Free credits!

As a student, you are eligible for $100 of free AWS credits, so you shouldn't need to pay out of pocket to use AWS in this class. To apply for these credits, sign up with your UW email address here: `https://aws.amazon.com/education/awseducate/`. Be sure to choose the regular AWS Account and not the AWS Educate Starter Account, as the starter account limits the resources you can use. It might take a few days to receive the credits, so sign up ASAP.

*`cjones6@uw.edu` If you find any typos, feel free to email me.

1.3 Example code used in this tutorial

In this tutorial we will examine how to run code that performs large-scale nearest neighbor computations. The code is from here: `https://github.com/gieseke/bufferkdtree`.

It is not important that you understand the code or the algorithm being used for the purposes of this tutorial. However, it is helpful to know what the code does: The algorithm is given a "training set" of points and a "test set" of points. Then for every point in the test set it computes the 10 nearest neighbors in the training set. If the training data had labels for its points, we could use this code to perform k-nearest neighbor classification.

1.4 If you use Windows...

If you have a Windows operating system you will need to make sure you have a way of SSHing and transferring files to a remote server. If you presently do not have one, you can install the Windows Subsystem for Linux by following the instructions here: `https://msdn.microsoft.com/en-us/commandline/wsl/install_guide`. I recommend choosing Ubuntu in the second part of the instructions. Once this is installed you will be able to connect to AWS in a Bash window. If you have an older version of Windows you should follow the instructions here: `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html`.

2 Getting started

Before using AWS's services, you should know about several things. These include (1) How to set up a budget and billing alerts; and (2) What regions are, and how to choose which one(s) to use.

2.1 Billing alerts/budgets

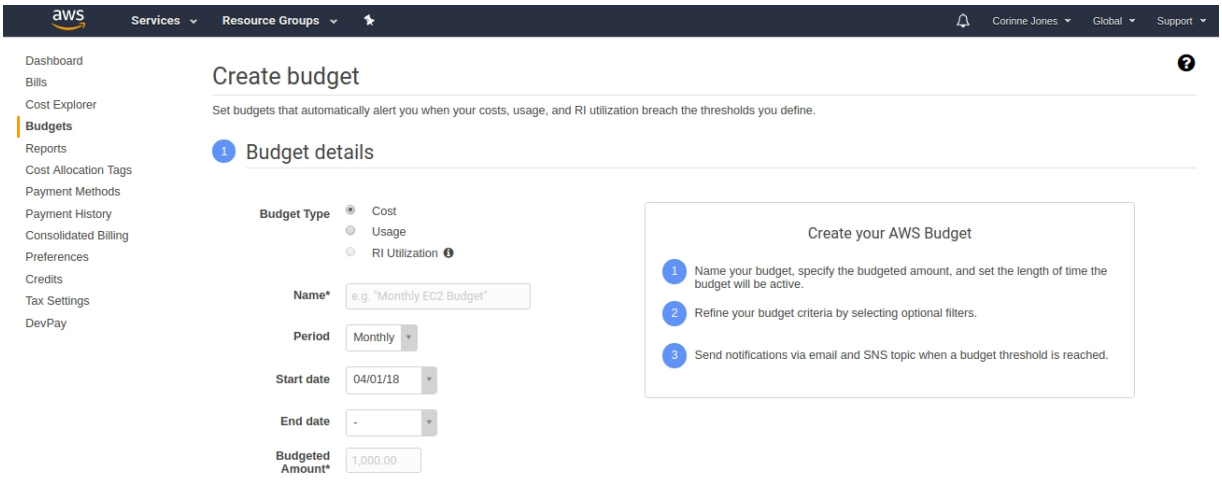
It is a good idea to create a budget and/or billing alerts to ensure you don't unknowingly spend more than you intend to. You can create these by clicking on your name in the upper right and then going to "My Account" and then either "Budgets" or "Preferences".

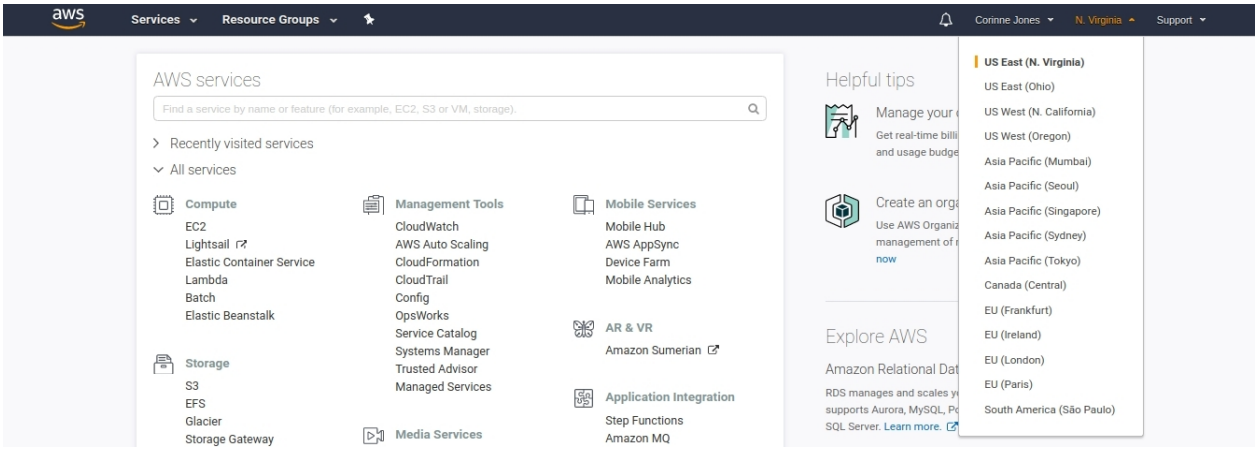
2.2 Regions

For many things you will do in AWS you will need to select a geographical region. A region is an area where Amazon has data centers. Each region is independent of the others for fault tolerance and stability. Moreover, each region has what are called "availability zones"– independent data centers in the same vicinity, which are used for redundancy and data replication. Some regions have three availability zones, while others only have two.

So, how should you choose a region? There are several factors you should consider:

- Proximity to you/your customers: The closer a data center is to the end user, the lower the latency and the faster the response will be.
- Cost: Costs vary by region, sometimes by a lot.





- Available services: Some services are only available in certain regions. E.g., AWS Machine Learning, a service consisting of a few off-the-shelf machine learning routines, is only available in N. Virginia and Ireland.

- Other regions you currently use: Not everything you use in one region is readily accessible from another region. On the other hand, using multiple regions could give you more fault tolerance, if that's what you need.

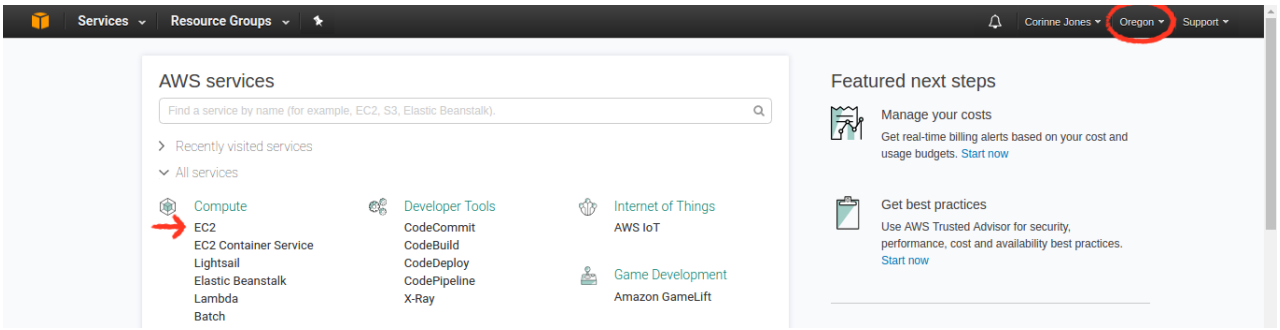
3 Launching an Elastic Compute Cloud (EC2) instance

Those who want to run their own own machine learning algorithms on AWS (as opposed to using off-the-shelf tools provided in Amazon's Machine Learning service) use the Elastic Compute Cloud (EC2). EC2 allows you to configure your own virtual computing environments, which they call "instances". This entails choosing a configuration of CPUs, GPUs, memory, storage, and networking capacity to meet your needs.

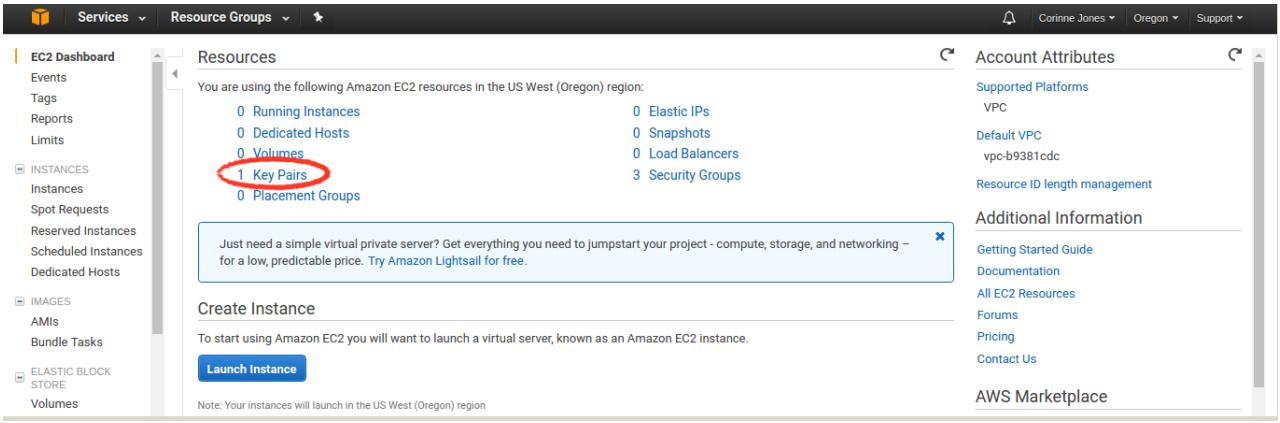
The most important thing to keep in mind when using EC2 is the following: **Remember**

to terminate your instances when you are done with them! Otherwise, you will rack up a very large bill!

Let's get started. On the AWS home page first change your region to N. Virginia in the upper right corner (The instances we are going to use tend to be cheaper there). Next, click on "EC2" in the upper left corner of the list of services.



3.1 Creating a key pair



The first thing you need to do is to create a key pair if you don't already have one. This allows you to securely log in to your instances. To create a new key, click on "Key Pairs" under "Resources". Then click on "Create Key Pair", enter a name for your key, and click "Create". A file with the extension .pem should automatically download. Save this somewhere where you will be able to find it, e.g., ~/.ssh/my-key-name.pem (on Linux/-Mac/Windows 10[1]) or C:\keys\my-key-name.pem (on old versions of Windows). You also need to change the file permissions on the private key file you just saved. You can do this in the terminal via

[1]If you're using Windows 10, move the key into that directory in the Bash window, not using file explorer. Assuming the key is in your Downloads folder and is called my-key-name.pem, you can do this via

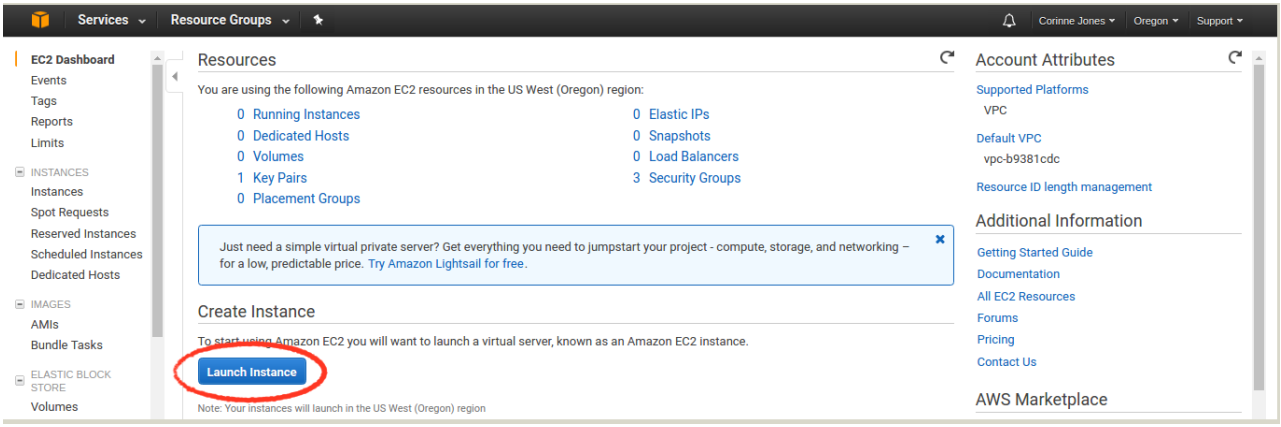
```
mv /mnt/c/Users/YOUR_USERNAME/Downloads/my-key-name.pem ~/.ssh/my-key-name.pem
```

where YOUR_USERNAME is replaced by your Windows username.

```
chmod 400 my-key-name.pem
```

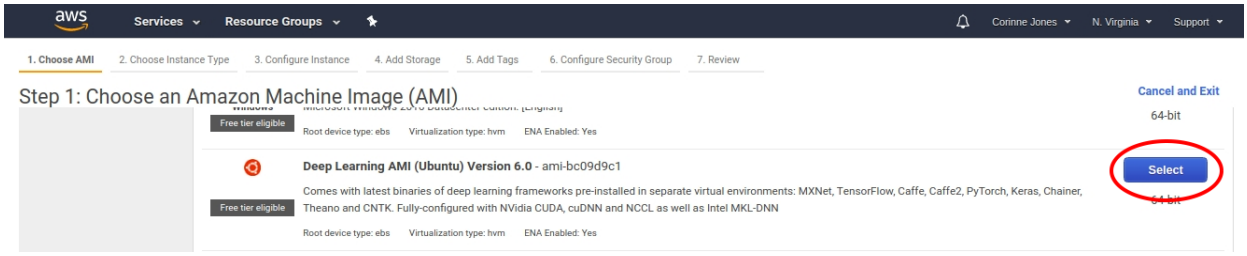
Now we are ready to launch an instance.

3.2 Amazon Machine Images



In the same window where you just were, click on "Launch Instance". The first thing we need to do is to select a machine image. Amazon Machine Images (AMIs) are preconfigured templates that have operating systems and additional software you might need. Beware of the fact that some AMIs are free, whereas others cost money to use (on top of the EC2 cost).

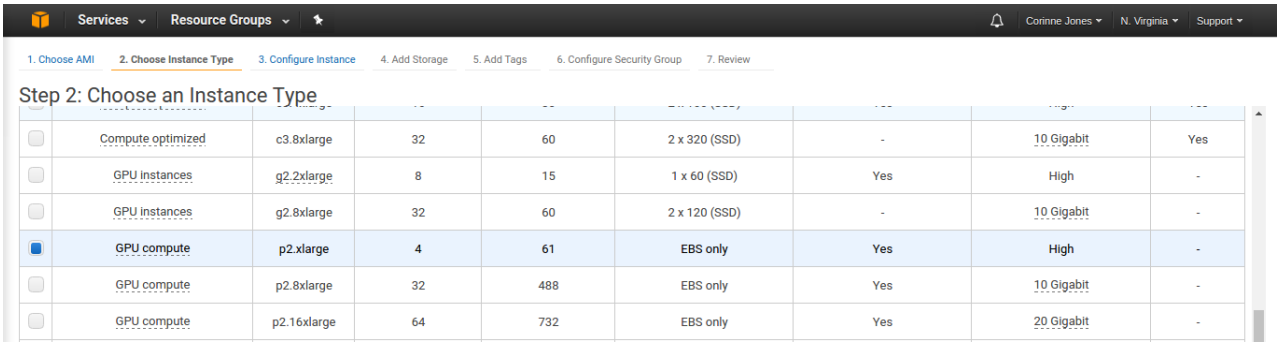
For our purposes we want to be able to run Python code on GPUs and use something called OpenCL. One AMI that has these pre-installed is called "Deep Learning AMI (Ubuntu) Version 6.0". You can scroll down in the Quick Start list to find and select it.



3.3 Choosing an instance type

Many different kinds of instances exist, and you can find the details on them here: `https://aws.amazon.com/ec2/instance-types/`. You are going to have to determine which one is most appropriate for the task you need to perform.

Let's choose the smallest GPU instance for general-purpose GPU compute applications: p2.xlarge. Then click on "Next: Configure Instance Details". Note that we are doing this because we are going to be running code on a GPU. If you do not intend to run code on a GPU you should choose a CPU instance instead.



Compute optimized	c3.8xlarge	32	60	2 x 320 (SSD)	-	10 Gigabit	Yes
GPU instances	g2.2xlarge	8	15	1 x 60 (SSD)	Yes	High	-
GPU instances	g2.8xlarge	32	60	2 x 120 (SSD)	-	10 Gigabit	-
GPU compute	p2.xlarge	4	61	EBS only	Yes	High	-
GPU compute	p2.8xlarge	32	488	EBS only	Yes	10 Gigabit	-
GPU compute	p2.16xlarge	64	732	EBS only	Yes	20 Gigabit	-

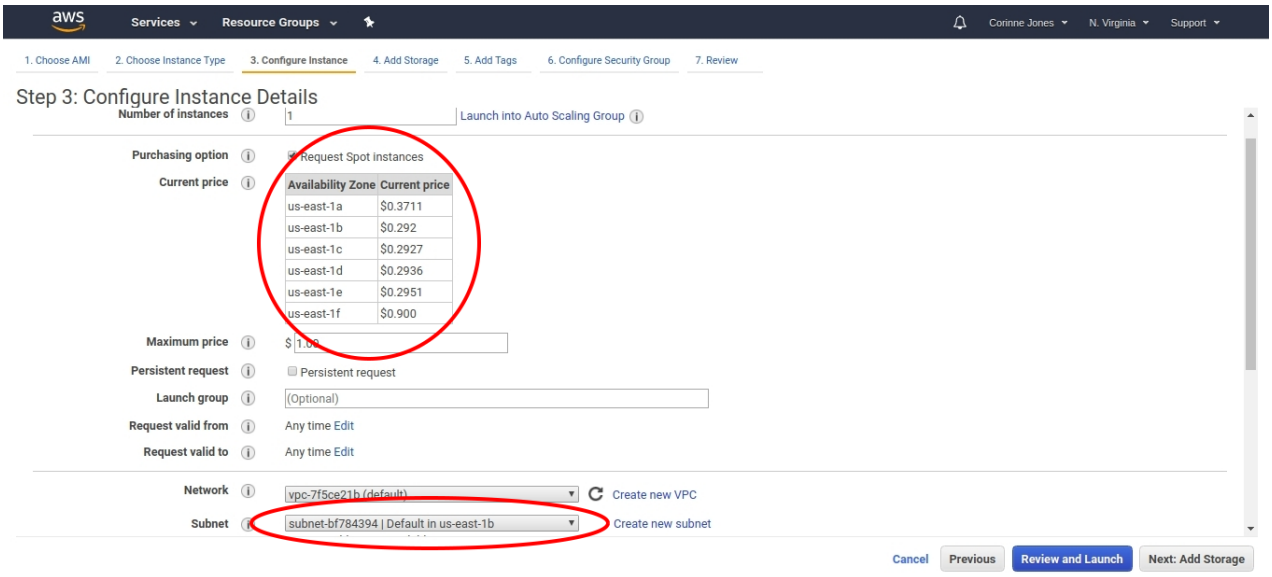
3.4 Configure instance details

Now we can specify various things like the number of instances we want and whether we want to request on-demand instances (the default option) or spot instances.

With an on-demand instance you pay by the hour and your use won't be interrupted. Moreover, you can increase and decrease your compute capacity at will. However, this tends to be significantly more expensive than spot instances.

If you choose a spot instance, you bid on Amazon's spare computing capacity. When you bid, you set the maximum price you are willing to pay per instance per hour. However, you will pay the current spot price, not your bid price. If at any point the current spot price exceeds your bid price while your instance is running, your instance is terminated. Your spot instance can also be terminated due to a lack of capacity or due to constraints. EC2 determines the interruption order for capacity interruptions.

Let's try the spot pricing. Choose a price above the current spot prices. Then you can change the subnet from "No preference" to the currently cheapest subnet. Leave everything else as-is and click "Next: Add Storage" at the bottom right.



3.5 Add storage

There are many different storage options when using EC2, and these are outlined in Appendix A. For our purposes, we will not be needing any additional storage, as we will be able to store everything we temporarily need in the root device storage and everything we want to permanently save in the S3 storage. I will discuss S3 later.

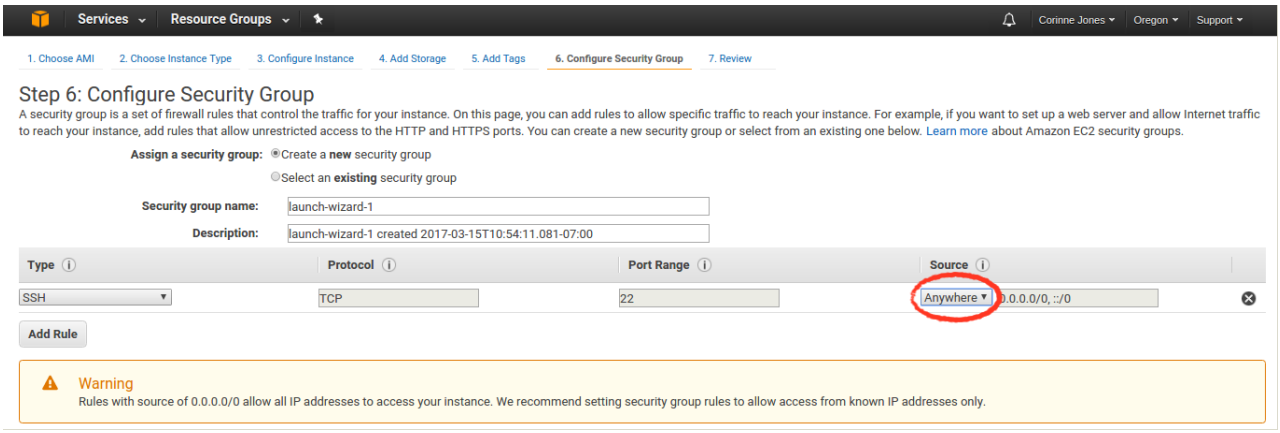
Therefore, leave everything as-is on this page. Note that the box "Delete on Termination" is checked for Root, meaning that everything we save to the root device storage will be deleted once we terminate the instance.

3.6 Add tags

If you wanted to, you could label your instance. This is useful if you have multiple instances, but as we don't here I'm going to skip this.

3.7 Configure Security Group

I recommend you change the security settings so that you only allow SSHing from your IP address. This makes it harder for your EC2 instance to be attacked. Then click "Review and Launch".



At this point you might be asked what you want to use as the default boot volume. If you are, I suggest you choose the first option.

3.8 Review and Launch

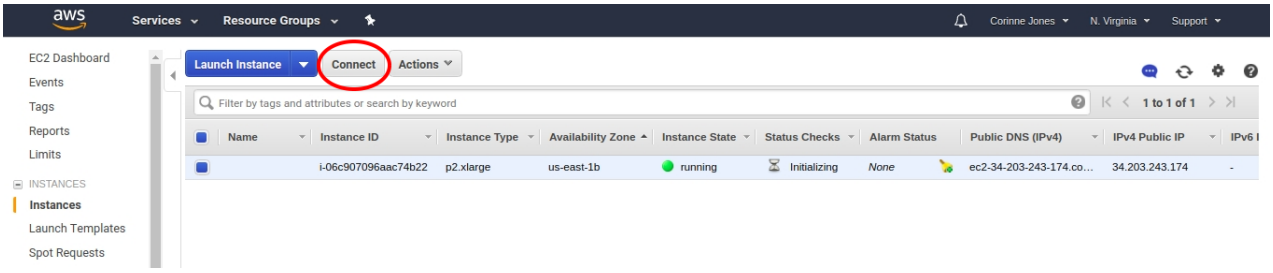
Click "Launch". You will be prompted to specify a key pair or create a new one. You can use the one you created before. Once you officially request your instances you can click the button that shows up in order to view their status.

If you encounter the error "Spot Request Failed. Max spot instance count exceeded" then you may need to request an increase by creating a case: `https://console.aws.amazon.com/support/home#/case/create?issueType=service-limit-increase&limitType=service-code-ec2-instances` It may take AWS several days to increase your limit. If you know that

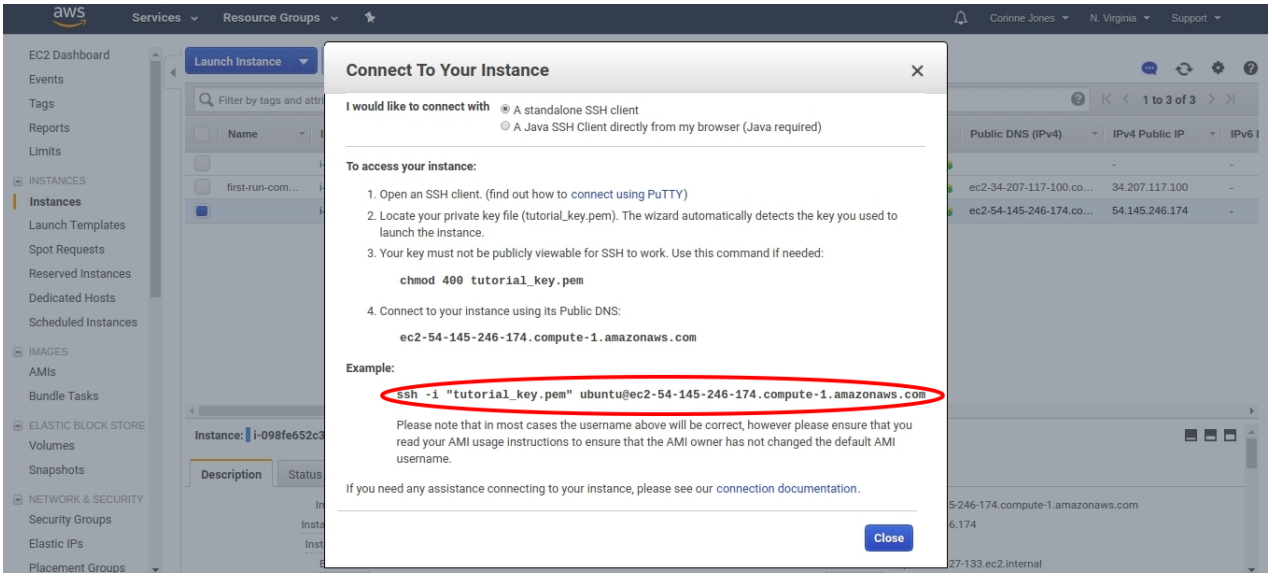
you did not actually exceed your limit then it may work if you wait several hours and try again.

4 Accessing your instance

Go to the status page for your instance (If you're not already there, click on Instances on the EC2 page). Then click on connect.



Clicking on connect will bring up a window telling you how to connect to your instance. Copy the example in that window. Then open a terminal window and paste what you just copied. Be sure to replace the key name (e.g., "tutorial-key.pem") with the path to your key (e.g., ".ssh/tutorial_key.pem"). Hit `enter` to connect. It will ask you if you are sure you want to continue connecting. Type `yes`. It might take a few minutes for it to set itself up, but now you should be on your instance!



At the top of the window it tells you how you can enter the Anaconda environment of our choice. We will choose the "base Python 2 (CUDA 9)" option, so type

```
source activate python2
```

There are a couple of things we need to install ourselves to run the example we're going to use: (1) swig; and (2) bufferkdtree. You can do this by running the following commands in the terminal once you are on your instance:

```
sudo apt-get install swig
pip install bufferkdtree
```

If it throws an error saying "E: Could not get lock", then wait a couple of minutes and try again.

Now we just need to transfer the example we want to run to our instance. The example may be found at `https://github.com/gieseke/bufferkdtree/blob/master/examples/artificial.py`. There are several ways you can do this, including the following three:

1. Using wget:

   ```
   wget https://raw.githubusercontent.com/gieseke/bufferkdtree/master/examples/artificial.py
   ```

2. Uploading the file (after downloading it to your computer): Install the AWS Command Line Interface (CLI) via the instructions in Appendix B. Then open a new terminal (without connecting to your instance) and run the command below (all on one line) after replacing "~/.ssh/tutorial_key.pem" with the path to your private key and the "DNS" with your instance's DNS address:

   ```
   scp -i ~/.ssh/tutorial_key.pem artificial.py ubuntu@DNS:~/
   ```

   Be sure to include and/or modify the last three characters :~/. This tells it to put the file in the home directory.

3. Copying and pasting: Copy the code from the website above. Then in the terminal where you want to store it on your instance, type

   ```
   vim artificial.py
   ```

   Right click and paste the code. Make sure the top didn't get cut off. If it did, type `i` (for insert) and then fix the top line. Then hit the `esc` key and afterward type `:wq` and hit `enter` to save and exit.

Now we can run the example! Move to the location the example is in on your instance and type the following in the terminal:

```
python artificial.py
```

The artificial example should run and output the ten nearest neighbors to one point from a sample of 10,000 points.

If you wanted to output the results to a text file instead, you could do something like

```
python artificial.py > output.txt
```

Now suppose we want to permanently save this. Then we could move it to S3 after properly installing and configuring the AWS CLI (See Appendix B) via

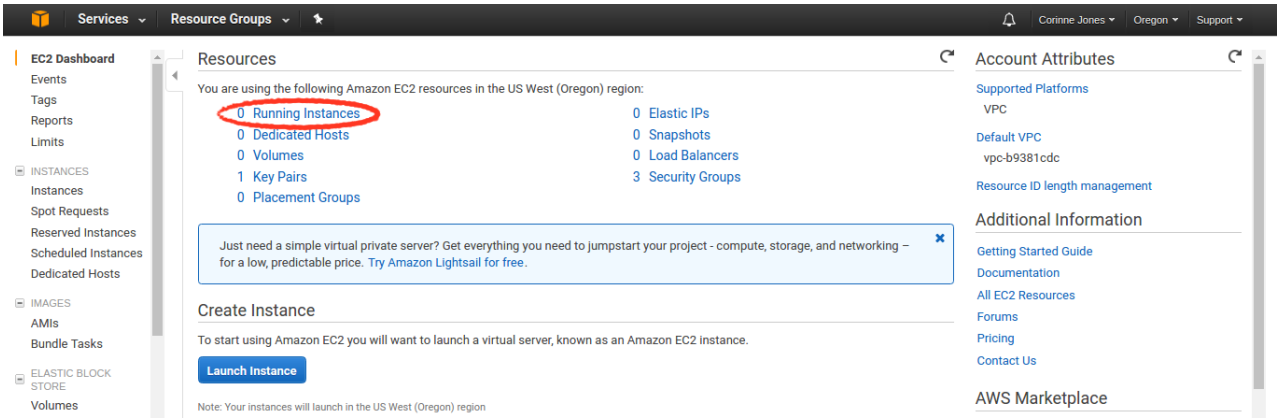
```
aws s3 cp output.txt s3://my-bucket-name/output.txt
```

where "my-bucket-name" is the name of a bucket you have already created in S3.

5 Terminating your instance

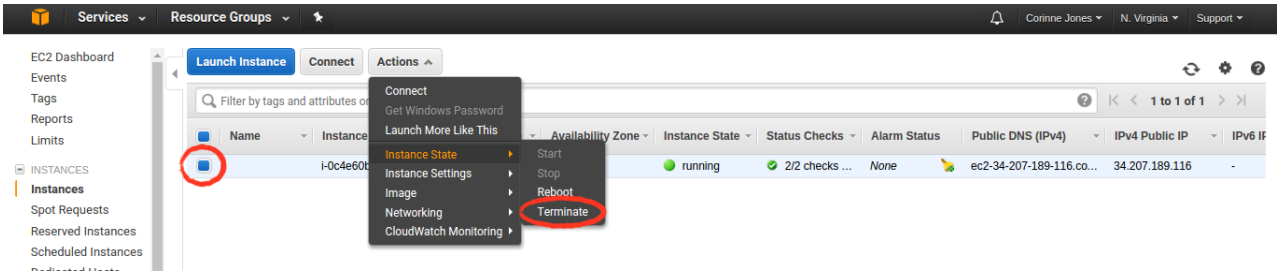
Once again, **remember to terminate your instance when you are done with it!** This entails doing the following:

1. Logging out of your instance in the terminal (e.g., by typing `exit`)
2. Returning to the EC2 page and clicking on "Running Instances" under Resources



3. Selecting the instance you want to terminate, clicking "Actions", "Instance State", and then "Terminate"

It might take a few minutes for the instance to shut down.



I also recommend you check your balance at this point. To do this, go to the cost and management dashboard: `https://console.aws.amazon.com/billing/home?#/`. To see costs below $1 you may need to click on "Bill Details". Also remember that you will incur storage charges for as long as you store anything on S3, so you will not want to leave files there indefinitely.

6 AWS Batch

If you want to submit batch jobs, you can do this via AWS Batch. With this you can still specify the EC2 instance type you want to use and even use spot instances. If you've read through and understood the rest of this document I'm confident that you can figure out how to use AWS Batch on your own.

Appendices

A Storage

When using AWS you will need to store your data, code, and/or output. There are several storage options on AWS, and which one(s) you will want to use will depend on your application and budget.

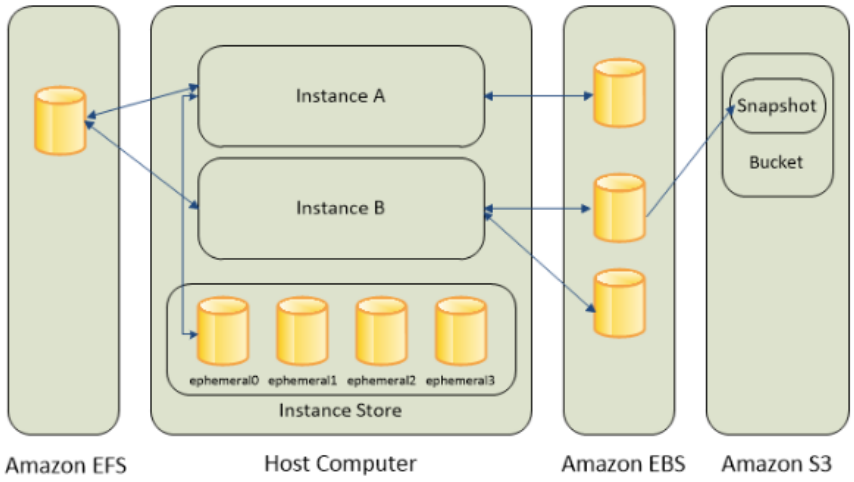


Figure 1: Storage options when using EC2. Source: `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Storage.html`

- Instance store: The instance store is temporary block-level storage that comes with an EC2 instance. The data stored on here is lost if the instance is stopped or terminated, or if the disk drive fails. However, the data is retained if the instance reboots. The disks could be HDDs or SSDs, depending on the instance you choose.
- Elastic Block Store (EBS): EBS is block-level storage that can be attached to any one instance in the same availability zone. Multiple volumes can be attached to the same instance. The data is quickly accessible and persistent. There are four different subtypes that vary on IOPS (input/output operations per second), throughput, and cost. Unlike with EFS, S3, and Glacier, you need to predefine the amount of storage

you want. This means you'll tend to overpay for it since you'll request more than you need.

- Elastic File System (EFS): Unlike EBS, EFS can be attached to multiple EC2 instances at once and can be accessed from multiple availability zones. It has a higher throughput than EBS, but also a slightly higher latency (See Figure 1). Currently you can only use these with AWS Linux instances.
- Simple Storage Service (S3): Unlike the storage types discussed above, S3 is an object store, not a file system. The data is accessible from every region, although you have to pay to move data from one region to another. Copying files from S3 to and from EC2 in the same region is free. S3 allows concurrent read/write access, but read/write operations are slower than with the other storage types above. Lastly, you should keep in mind that you can't modify a file–you can only delete and re-upload it, which slows things down.
- S3-Infrequent Access (S3-IA): This storage is for data accessed less frequently than that in S3, but that requires rapid access when needed.
- Glacier: Glacier storage is long-term storage for archiving data. It has slow retrieval rates (1-5 minutes for expedited retrieval) and the retrieval costs are more expensive.

		Amazon EFS	Amazon EBS PIOPS
Performance	Per-operation latency	Low, consistent	Lowest, consistent
	Throughput scale	Multiple GBs per second	Single GB per second
Characteristics	Data Availability/Durability	Stored redundantly across multiple AZs	Stored redundantly in a single AZ
	Access	1 to 1000s of EC2 instances, from multiple AZs, concurrently	Single EC2 instance in a single AZ
	Use Cases	Big Data and analytics, media processing workflows, content management, web serving, home directories	Boot volumes, transactional and NoSQL databases, data warehousing & ETL

Figure 2: EFS vs EBS. Table taken from `https://aws.amazon.com/efs/details/`

Storage costs for 1 GB per month in the N Virginia region as of 4/1/18 (prices vary by region) :

- EBS: \$0.025-\$0.125 (+ IOPS cost if you choose provisioned IOPS SSDs)
- EFS: \$0.30
- S3: \$0.023

- S3-IA: $0.0125 (+ $0.01 per GB retrieval)
- Glacier: $0.004 (+ retrieval cost)

If cost is an issue, then your best choice may be to store your data that is used relatively frequently in S3 and then move it to EBS or EFS when you need to use it on EC2.

B Transferring data

It is important to know how to move data to/from your computer to the various storage services. Here I discuss how to transfer data between your computer, S3, and EC2.

In this section we will make use of the command line interface (CLI). If using Conda, you can install it by running the following command:

```
conda install -c conda-forge awscli
```

If not using Conda, you can install it with pip:

```
pip install --upgrade --user awscli
```

(Note that the above command has two dashes before upgrade and two dashes before user.) If that fails, read the instructions here: `http://docs.aws.amazon.com/cli/latest/userguide/installing.html`.

You will also need to configure the CLI if you want to use it to move files to or from S3. To do so, follow the instructions here: `http://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started_create-admin-group.html` to create an IAM profile for yourself as an administrator. This will generate an AWS Access Key ID and AWS Secret Access Key for you. Then follow the instructions here: `http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html` to configure the CLI.

B.1 Transferring data between your computer and S3

The easiest way to transfer data to and from S3 is via the graphical user interface. To access it, go to the main AWS page by logging into AWS and/or clicking the orange picture of a cube in the upper left corner. Then click on S3.

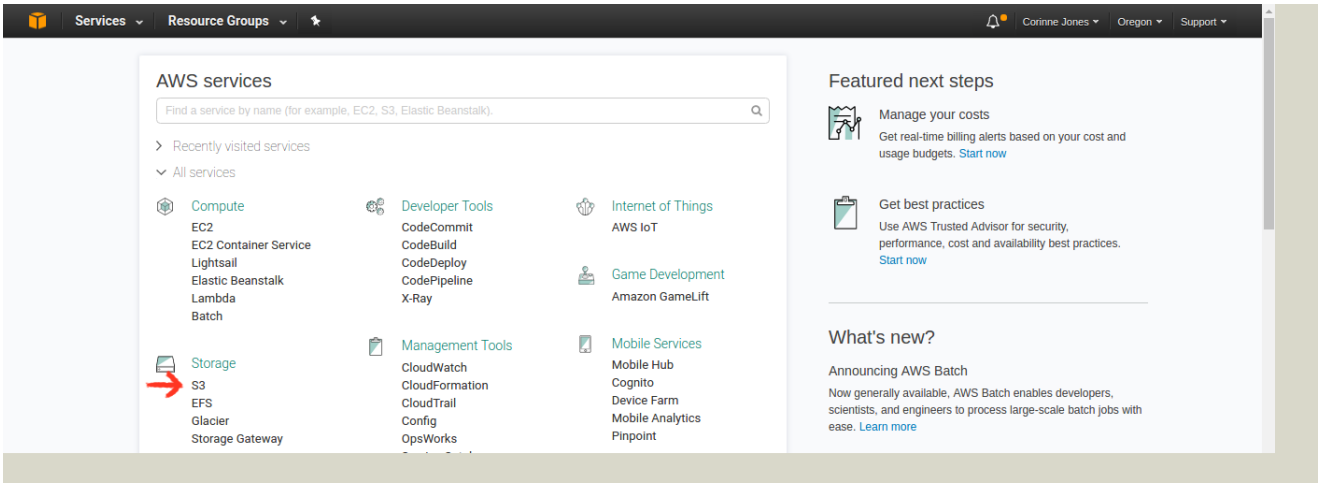
Here you can create "buckets" where you can upload files and also download and delete existing files. In addition, you can perform other actions like set permissions and define "Lifecycle" rules that automatically transition your files to other forms of cheaper storage after a certain period of time. Note that bucket names are global, meaning that each of your bucket names must be different from every other AWS users' bucket names.

Alternatively, you can upload the data using the CLI. For example, if I want to move a file called artificial.py from my computer to S3, I can first run the command

```
aws s3 mb s3://stat558-cjones6
```

to create a bucket called "stat558_cjones6" and then run the command

```
aws s3 cp artificial.py s3://stat558-cjones6/artificial.py
```



to move the file "artificial.py" to the newly created bucket. To copy the same file from S3 onto my computer, I can just reverse the order of the file paths:

```
aws s3 cp s3://stat558_cjones6/artificial.py artificial.py
```

B.2 Transferring data between S3 and EC2

Transferring data between S3 and EC2 is pretty much the same as transferring data between your computer and S3. The only difference is that you run the CLI from your instance rather than from your computer; the commands are the same.

B.3 Transferring data between your computer and EC2

To use the terminal to transfer a file called "artificial.py" from your computer to your EC2 instance, you supply the path to your private key, the location of your file on your computer, and where you want it to go:

```
scp -i ~/.ssh/tutorial_key.pem artificial.py ubuntu@DNS:~/
```

In other words, you should replace " /.ssh/tutorial_key.pem" above with the path to your key and "DNS" with the DNS address of your EC2 instance. To transfer a file from your instance to your computer, just reverse the order of the commands:

```
scp -i ~/.ssh/tutorial_key.pem ubuntu@DNS:~/ artificial.py
```

C Sources

Regions:

- `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html`

- https://www.concurrencylabs.com/blog/choose-your-aws-region-wisely/

EC2:

- http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html
- https://aws.amazon.com/ec2/pricing/
- http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html

Storage:

- http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Storage.html
- https://aws.amazon.com/efs/details/
- http://stackoverflow.com/questions/29575877/aws-efs-vs-ebs-vs-s3-differences-when-to-use
- https://aws.amazon.com/efs/pricing/
- https://aws.amazon.com/s3/pricing/
- https://aws.amazon.com/ebs/pricing/

CLI:

- http://docs.aws.amazon.com/cli/latest/userguide/installing.html
- http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html
- http://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_create.html?icmpid=docs_iam_console
- http://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started_create-admin-group.html

Code:

- Fabian Gieseke, Justin Heinermann, Cosmin Oancea, and Christian Igel. *Buffer k-d Trees: Processing Massive Nearest Neighbor Queries on GPUs. In Proceedings of the 31st International Conference on Machine Learning (ICML)* 32(1), 2014, 172-180.