

— Introduction to *AWS*

Original Author: Tim Book



What is Cloud Computing?

Yet another buzzword in our field. Luckily this one is simple:

Cloud computing is computing on a server that is maintained by someone else.



Why Cloud Computing?

There are two major reasons why you would do this:

1. **Powerful computers are expensive.** It is much cheaper and more efficient to rent them hourly from someone else.

Cloud computing services such as AWS give us access to servers that would cost tens of thousands of dollars if we bought them ourselves. You could even execute code on one of these servers remotely from your laptop!



Why Cloud Computing?

There are two major reasons why you would do this:

2. **Maintaining servers is cumbersome** (and expensive), even for medium-sized companies.

For data scientists, compute times can range from minutes to days. What if the power goes out? What ensures your data won't be lost? Worse: if you maintain a website, your server needs to be **running at all times!** That's an expensive electric bill.

What is a Server?

The word **server** is just another word for **computer** or **computer program** that communicates with, or provides functionality for, a **client**.

Your computer, right now, is a server. Your computer might also be running another server: **Jupyter**.

For us in this lesson, we'll mostly be concerned with **computing servers** supplied by AWS.



Who Supplies Them?

There are many providers of large-scale cloud computing services:

- Amazon Web Services
- Google Cloud Platform
- Microsoft Azure Cloud Computing

If you mainly want a personal always-on server in the cloud, it might be more economical (\$5+/month) to get a virtual private server (VPS):

- Digital Ocean
- Linode



Where are these servers?

Amazon has a few server farms throughout the country. Amazon likes to keep their exact locations as secret as possible for security purposes.

On the east coast, we'll be connecting to **northern Virginia**. There are multiple on the west coast.



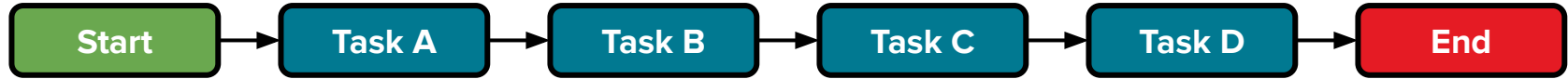
Why do data scientists need cloud computing?

There are many reasons, but today we will spotlight **parallelization**.

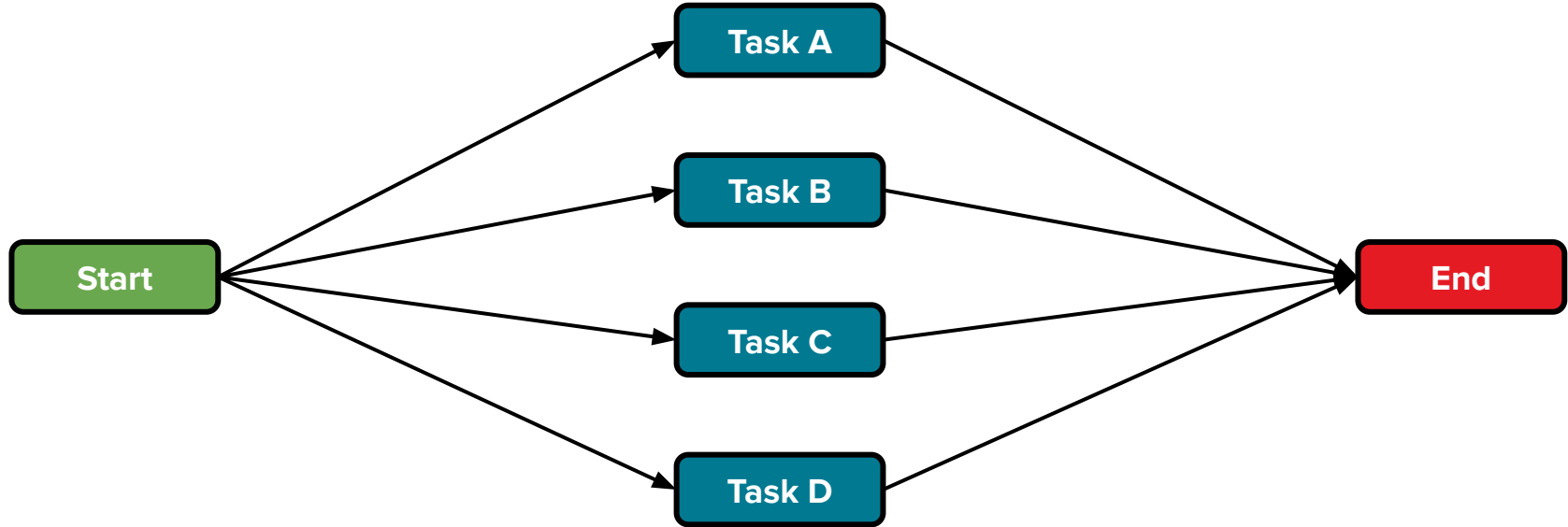
There are many machine learning algorithms that can be parallelized, resulting in much faster computation.



Programming in Serial



Programming in Parallel



Ok, cool. How do we do this?

We're going to go through several steps to get up and running with AWS:

1. Set up our keys for security.
2. “Spin up” a server.
3. Connect to the server.
4. Install Anaconda
5. Push data to the server.
6. Execute code on the server (either via the command line or Jupyter).
7. Pull results down from the server.



Step 1: Security

Since we'll be hosting potentially sensitive/proprietary data on the web, it's important that no one else can hack into our server!

In order to access our own servers, we'll need two things:

1. A security key (a .pem file)
2. Permissions from the admin (you)

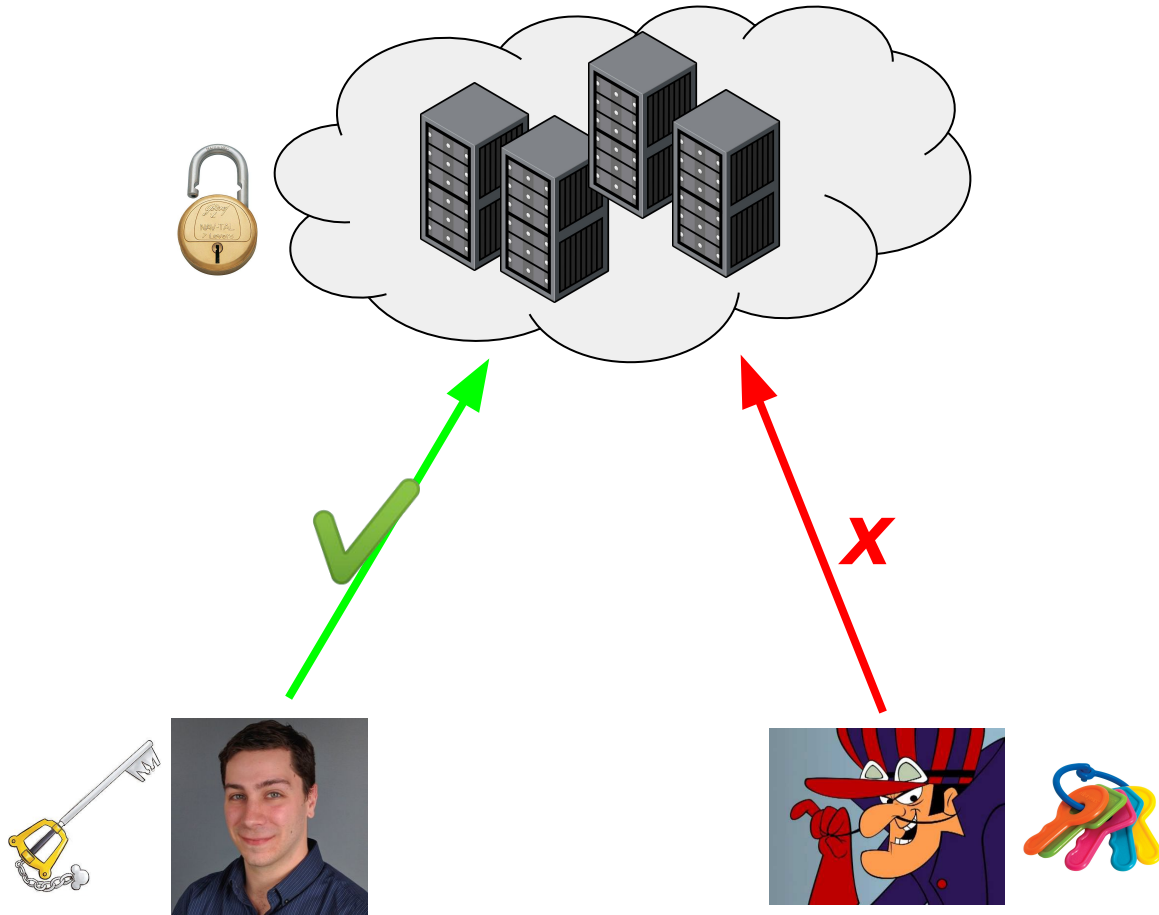


Step 1: Security

You'll need a key to get into your own server.

The **private version** of your **key pair** will be a **.pem** file.

(Side note: AWS will demand your .pem file has certain security settings. You may need to chmod it.)



Step 1: Security

You'll also need to be allowed by permissions of a **security group**.

In most cases, this will be a range of allowable IP addresses you may connect from.

Only allow:
12.34.56.*



Connecting from:
12.34.56.78



Connecting from:
11.22.33.44



Step 1: Security

Let's configure this now!



Step 2: Spinning up a server

You'll have a seemingly overwhelming amount of choices to make here. Let's break it down one at a time:



Choice 1: Operating System

AWS offers several flavors of Linux, as well as some versions of Windows.

We're choosing Linux. Windows is not as popular in cloud computing, and it is often more expensive.

A beginner- and user-friendly flavor of Linux is **Ubuntu**. It's also the most widely used, which makes things easier.



Choice 2: Instance Type

You'll also be presented with a buffet of instance choices. Your decision will largely be based on two things:

1. How much memory do you need?
 - The amount of RAM you have determines how much data you can hold in memory at once. Larger data = more RAM needed.
2. How many cores do you need?
 - The amount of cores you have determines the number of threads you may parallelize with (i.e., **n_jobs**). More threads = faster compute time.



Choice 2: Instance Type

Bigger instances are more expensive. Don't fall into the following pitfalls:

- **Estimate the resources you need before you buy.** Otherwise, this is a waste of money for no benefit!
- **Don't choose a high-powered instance if your model is not parallelizable.** You probably won't see any improvement, and you'll just be wasting money.

[Here is AWS's EC2 pricing guide!](#)



Step 2: Spinning up a server

Finally, you'll need to tell AWS your security settings. In the tab labeled **6. Configure Security Group**, you'll specify your security group. After you hit "Launch" from **7. Review**, you'll be prompted for the key you want to use.

Let's do all of this now!



Step 3: Connect to the Server

Using our command line, we'll connect to our server using a new protocol: **SSH**, the **secure shell** protocol. As long as you've properly **chmod**ed your key, this is easy:

```
ssh -i /path/to/key.pem ubuntu@ec2-ip-address.amazonaws.com
```

Flag telling ssh
you're using a
key file.

Path to .pem file.

Default username on
Ubuntu is "ubuntu"

Public DNS (IPv4) for
this instance

Step 4: Install Anaconda

This is as simple as a few CL commands:

(Pull Anaconda install from internet)

```
wget http://repo.continuum.io/archive/Anaconda3-2020.07-Linux-x86_64.sh
```

(Execute installer - make sure to say "yes" - add to PATH)

```
bash Anaconda3-2020.07-Linux-x86_64.sh
```

(Restart Bash configuration)

```
source ~/.bashrc
```

(Check to see if it all worked!)

```
which python3
```

```
which conda
```

Step 4½: Something something Jupyter Notebooks

Yes, the rumors are true, you can actually run a Jupyter Notebook in your EC2 instance and interact with it via your local web browser. Setting this up requires an additional setting in your security group, and the manual editing of a few config files in your AWS instance. I'm not going to do this here because it's very error-prone and is just a few more directions to follow, but here are two very good tutorials:

[Tutorial from the legendary Chris Albon](#)

[Another tutorial from Alex Sanchez](#)

Step 5: Uploading files to the instance

In order to save time, money, and energy, **you should be building your code locally and then uploading**. You'll also need to upload your data. You can transfer files back and forth with yet another protocol: **scp** (secure copy protocol).



Step 5: SCP

Easy mode: Using your mouse

There are GUI-based SCP clients that make this very easy:

For Windows: **WinSCP** is best

For Everyone: **Filezilla** is popular

Step 5: SCP

Expert mode: Using the command line. (Reminder: you are an expert)

(Copy file.txt from local to remote)

```
scp -i key.pem file.txt ubuntu@ip-address:~/target_directory
```

Note!

(Copy my_folder directory from local to remote)

```
scp -i key.pem -r my_folder ubuntu@ip-address:~/target_directory
```

Note!

(Copy file.txt from remote to local)

```
scp -i key.pem ubuntu@ip-address:~/data/file.txt ~/local_dir
```

(General structure)

```
scp -i key.pem from to
```

Step 5: SCP

Let's bring **model1.py** into our instance!



Step 6: Execute code!

Hopefully, the easiest part! Let's run:

```
python3 model.py
```

Note that because of Linux, we'll be executing Python scripts with **python3** instead of **python**.



This is a miniature example, but:

When I run a RandomForestClassifier on an arbitrary classification problem with:

$n = 300,000$

$p = 20$

$C = 5$

Computer	Cores	Time	Improvement
Local	1	13m17s	1x
c5.2xlarge	8	2m37s	5x
c5.9xlarge	36	40s	20x

Step 7: Pulling the results back down

Remember to use your favorite **scp** tool!

Let's bring our output.csv down...



Cliffsnotes

1. Have security features ready (security key, security group)
2. Instantiate EC2 instance (Slides 14-17)
3. SSH into instance (Slide 18)
4. Install Anaconda (Slide 19)
5. SCP relevant files to instance (Slides 21-23)
6. Execute code
7. SCP results back down (Slide 23)
- 8. TERMINATE INSTANCE!**



How to “memorize” this:

1. ***Understand*** what is happening in each step, and ***why*** it is being done.
2. Go through this process over and over until you feel like you can do it with minimal reference to this guide.



Summary of Important Notes

1. Don't forget to terminate instances when you're done! Easy way to run your company into the ground.
2. Don't be a doofus - keep your security keys secure.
3. Understand if using AWS is really what you need!
 - a. Is your problem parallelizable?
 - b. Are you actually parallelizing?
 - c. Can your problem take advantage of more storage, CPU power, or RAM?

