Remaining:

Final smoke test of setup script

A thing to parallelize ssh-ing

A thing to create the users as needed

Download the code into my git repo

Initialize directories in my git repo

Auto-emailing login information

Get information on ssh port forwarding with jupyter

Test all of this with the class’s resources

For parallel ssh-ing:

<https://codereview.stackexchange.com/questions/168859/running-ssh-commands-on-multiple-servers>

Email information:

Machinelearning.uruguay@gmail.com

support\_vector\_machine

AWS Stuff

Account ID 999458389316

IAM user name: julian.a.alverio

pw: Standard secure

Otherwise:

Julian.a.alverio

Standard insecure

Linux 2 AMI

ami-00068cd7555f543d5

Linux AMI

ami-00eb20669e0990cb4

ssh -i /Users/julianalverio/Desktop/ec2-keypair.pem [ec2-user@](mailto:ec2-user@ec2-3-95-189-184.compute-1.amazonaws.com)ec2-34-207-167-29*.compute-1.amazonaws.com*

Here is documentation on what everything means in client.describe-instance():

<https://docs.aws.amazon.com/cli/latest/reference/ec2/describe-instances.html>

Setting up credentials with my personal account

1. Follow the instructions here on how to create a user with the right permissions, download the credentials in a csv, and transfer the credentials to the AWS CLI <https://stackabuse.com/automating-aws-ec2-management-with-python-and-boto3/> I believe those same instructions are here <https://blog.ipswitch.com/how-to-create-an-ec2-instance-with-python>
2. Use the second link to create a keypair and then run chmod 400 on it
3. Those two links are enough to write the API that I wrote to manage instances.
4. If you can’t ssh in initially, you will need to modify the default security rule so that it will accept ssh connections from anywhere, not just a particular IP address
5. To find the ubuntu AMI for a particular region, search “bionic” here <https://cloud-images.ubuntu.com/locator/ec2/>
6. The root user is ubuntu for ubuntu AMIs, and ec2-user for linux. Check that here <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/connection-prereqs.html>
7. Ssh in with:
   1. ssh -i /Users/julianalverio/code/machine\_learning\_aws/ec2-keypair.pem [ubuntu@3.91.191.74](mailto:ubuntu@3.91.191.74)
   2. ssh -i /Users/julianalverio/code/machine\_learning\_aws/ec2-keypair.pem ubuntu@ec2-34-207-167-29.compute-1.amazonaws.com

GSL Uruguay Stuff

1. Our previous plan of giving them all constant GPU access puts us way over budget
2. First 2 weeks don’t require GPU access. Use CPU-only machines. To avoid being wasteful, keep these on 16 hours a day.
3. Even then, the smallest GPU machine is too expensive for the second 2 weeks
4. Let’s keep GPU machines on only 16 hours a day – the other 8 hours are wasteful
5. Let’s put multiple students on big GPU machines. This both saves cost and provides more resources per student. It’s also easier for us to manage fewer machines.
6. While we’re at it, let’s provide bigger cpu-only machines for the students. The cost per student and resources per student are still the same, but students can then use each other’s underutilized resources.
7. At the lower bound, we can actually run the whole program for about $3,685.06 + memory storage. This gives plenty of resources per student (4.2 cpu, 18.25GB, 1GPU), but not tons of resources for the first two weeks (1 cpu, 1.14GB memory)
8. Since we have money left over, let’s allocate more cpu-machine resources per student for the first two weeks. With beefier machines we can get 4 cpus and 8.6GB of memory per student, at a total of $4,713.22 🡪 these are more resources than a lot of people’s laptops have. This is about what a 2015 macbook pro has. The students can also use each other’s unused computational resources.
9. This does not account for data storage costs while the machines are turned down, but I will come back to this.
10. For access, we make a username and default password for each student.
11. For dependencies, we make each of them a conda environment and give each of them an identical working directory.
12. For backups, switching between machines, and avoiding data storage costs, we copy their folders into a giant folder and push it to github every night, and shut down their machines. In the morning it’s trivial to spin them up.
13. Data storage costs are tricky to compute, and add up quickly.
14. The scripts for this are easy to write, but require a lot of documentation reading. So we can just write this over Christmas break.

**Other stuff**

1. We need the code notebooks online

**The Math**

$0.0255 the most basic a1.medium machine

For 60 students over 14 days, 16 hours a day 🡪 $342.72

This is for 1 vcpu and 2 GiB of memory

a1.4xlarge – 16 vcpu and 32 GiB, 4 students per machine 🡪 3.3

0.408 per Hour per machine \* 15 machines \* 16 hours/day \* 14 days = 1,096.71

g4dn.16xlarge -- $4.352 per hour

For 4 machines (15 students per machine) for 16 hours per day over 12 days: $3,342.336

4.2 cores per student, 17 GiB 🡪 18.25 GB per student, 1 GPU per student

AMI for vanilla linux

ami-00068cd7555f543d5

Launched EC2 Instance i-0b5e383b6d8e01b25

INFO: 2019-11-23 22:21:30,449: VPC ID: vpc-2a964c53

INFO: 2019-11-23 22:21:30,449: Private IP Address: 172.31.64.249