



Exercise 1: AWS Setup & Docker

Each assignment is graded out of 5 points. The topic for this assignment is getting set up with AWS and running your first Docker application on AWS.

Question 1: Create Accounts at Cloud Services (1.0 points)

If you have not already done so, create accounts at the following cloud services we will be using in the course:

1. GitHub: www.github.com
2. Amazon Web Services: <https://aws.amazon.com/>
3. (Optional) DockerHub: <https://hub.docker.com/>

Submit:

1. your username for GitHub and AWS
2. a screenshot of the “profile” page of your GitHub account and the AWS management console

Example Submission:

GitHub username = memmanuel

Personal settings

Profile

Account

Security

Security log

Emails

Notifications

Billing

SSH and GPG keys

Blocked users

Repositories

Organizations

Saved replies

Applications

Developer settings

Public profile

Name

Michael S. Emanuel

Your name may appear around GitHub where you contribute or are mentioned. You can remove it at any time.

Public email

mse999@g.harvard.edu

X Remove

You can manage verified email addresses in your [email settings](#).

Bio

After 17 years in finance, I started a second career and am currently in the Masters of Data Science program at Harvard.

You can @mention other users and organizations to link to them.

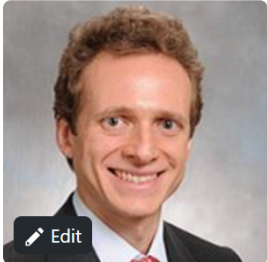
URL

Company

Elliptic Enterprises

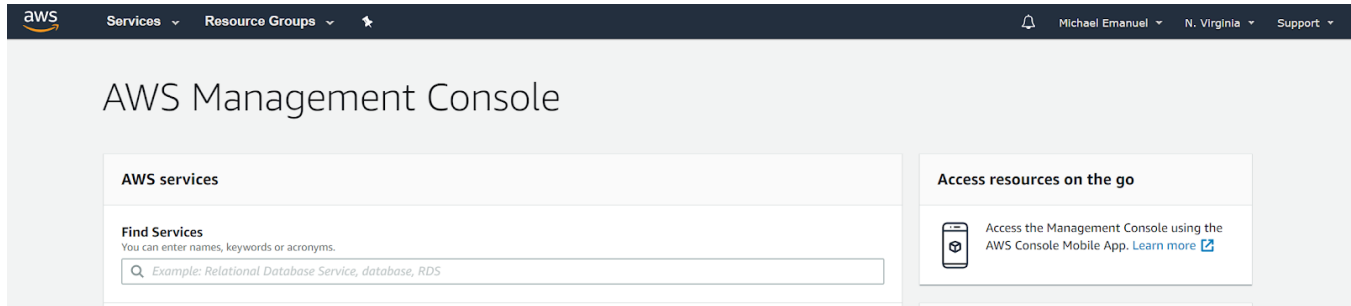
You can @mention your company's GitHub organization to link it.

Profile picture



Edit

AWS username = michael.s.emanuel@gmail.com



Question 2: Activate Two Factor Authentication (0.5 points)

Good security is critical to practical data science.

If you have not already done so, please activate two factor authentication for your accounts at GitHub, AWS, and DockerHub, and test that it's working by logging out and logging back in with 2FA.

We recommend an app such as Google Authenticator (or LastPass authenticator) as a more secure alternative to SMS messaging.

Submit:

1. a screenshot from your settings page showing 2FA configured for GitHub

Note: we do not require that you use 2FA for this course. If you really don't like it, you can turn it back off after you do this question. We're trying to help you develop good habits, not be your annoying big brother.



Example Submission:

Change password

Old password

New password

Confirm new password

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

Update password

[I forgot my password](#)

Two-factor authentication

Enabled

Two-factor authentication adds an additional layer of security to your account by requiring more than just a password to log in. [Learn more.](#)

Two-factor methods		
Authenticator app	Configured	Edit
Security keys i	2 security keys	Edit
SMS number	Not configured	Edit
Recovery options		
Recovery codes i	Viewed	Show
Fallback SMS number i	+1 9175922163	Edit
Recovery tokens i	Account recovery enabled	Edit



Question 3: Set up SSH Keys (0.5 points)

Many tasks in deploying apps to the cloud are done at the command line. SSH is an essential tool for secure use of command line interfaces to remote systems. SSH is also more convenient than password authentication once you've set it up.

If you have not already done so, create a default SSH key on your main computer.

Configure GitHub so that you can access your repositories using SSH. You can follow this tutorial:

<https://help.github.com/en/enterprise/2.18/user/github/authenticating-to-github/adding-a-new-ssh-key-to-your-github-account>

Try to clone one of your repositories at the command line using the SSH interface.

Submit:

1. a screenshot of your GitHub account showing at least one SSH key.
2. a screenshot of a terminal where you clone any repo with the SSH.

Please note that it is always safe to share the **public** part of your SSH key as in submission 1, but you should never share a private RSA key with anyone!

As an example of cloning by SSH, when I type:

```
$ git clone git@github.com:memmanuel/AC-295-MSE.git
```

this command works “by magic” on a computer that has one of the SSH keys I’ve configured on GitHub in the default key location `~/.ssh/id_rsa` (linux and Mac) or `C:\Users\Michael\.ssh\id_rsa` (Windows 10)

Example Submission:

Personal settings

Profile

Account

Security

Security log

Emails

Notifications

Billing

SSH and GPG keys

Blocked users

Repositories

Organizations

Saved replies





Applications

Developer settings

SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

 SSH	Michael-Tablet (Microsoft Surface Pro 4) / Git BASH 9c:f8:16:97:bf:f2:2a:7a:e3:5d:ad:bd:fc:c6:27:38 Added on Sep 4, 2018 Last used within the last 4 months — Read/write	Delete
 SSH	michael@Loki f3:57:31:bc:6f:d8:47:68:48:fe:c8:62:95:d0:d4:01 Added on Oct 5, 2018 Last used within the last 2 weeks — Read/write	Delete
 SSH	Michael-MacBookPro ac:d9:3b:4e:8c:14:b0:21:0f:c6:e4:f2:06:aa:dc:0c Added on Feb 26, 2019 Last used within the last week — Read/write	Delete
 SSH	michael@Grimnir 63:77:de:92:22:27:15:b2:cc:53:47:b7:80:95:91:16 Added on Dec 20, 2019 Last used within the last 2 weeks — Read/write	Delete



```
D:\Temp
λ git clone git@github.com:memmanuel/AC-295-MSE.git
Cloning into 'AC-295-MSE'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 85 (delta 0), reused 3 (delta 0), pack-reused 80
Receiving objects: 100% (85/85), 60.57 MiB | 6.61 MiB/s, done.
Resolving deltas: 100% (17/17), done.
```

Question 4: Your First Docker App on AWS (3.0 points)

Work through the tutorial from the reading on containers:

<https://towardsdatascience.com/data-science-for-startups-containers-d1d785bfe5b>

Get the echo application working on AWS.

Submit:

1. a screenshot from your computer showing the echoed response from the public IP address of the server at AWS.
2. a text file with a log of all the commands you typed at the command line along with enough comments that you could easily recreate your steps in the future.

Maintaining a text file with all the commands required to perform a task at the command line is a good practice. The next time you need to do this or a similar task, you will be happy to have it. This shouldn't be the raw log with every command you typed including those that failed. Edit it down to just those that accomplished your goals. Include comments to remind yourself of anything that was non obvious at the time.

For those of you who attended the optional lecture, this question is asking you to do exactly what I demonstrated on my MacBook.

If you can't get the end to end version working, don't despair. Describe the steps that worked, and tell us where things broke down, for very generous partial credit.

Also, don't hesitate to post questions on the Ed class discussion board. One of your classmates or the teaching staff can help.

Example Submission (text log):

Run on AWS Ubuntu 18.04

- 1) Go to console.aws.amazon.com. Spin up a tiny instance. Select Ubuntu 18.04 LTS.
- 2) Assign the default security policy to it (not the throwaway policy!). This will allow connections.
- 3) Find the public name and IP address. Connect by SSH
\$ ssh ubuntu@ec2-54-166-117.compute-1.amazonaws.com
(don't need to input SSH key because EllepticEnterprises.pem has been added to SSH agent.)
- 4) Install first batch of software
\$ sudo apt install python3-pip python3 python3-setuptools
- 5) Install Docker; regular APT install doesn't appear to work



```
$ curl -fsSL https://get.docker.com -o get-docker.sh
$ sudo sh get-docker.sh
$ sudo usermod -aG docker ubuntu
$ sudo service docker start
$ sudo docker ps
$ pip3 install flask
```

copy GitHub personal access token from Michael-PC:

```
$ scp -i AWS\EllipticEnterprises.pem GitHub\GitHub_PersonalAccessToken ubuntu@35.173.232.130:~
```

clone repo with exercises

```
$ git clone https://github.com/memanuel/AC-295-MSE.git
```

Because I'm using 2FA, password must be contents of GitHub_PeronsalAccesToken (NOT usual password!)

```
$ python3 echo.py
```

in browser on Michael-PC, enter:

<http://ec2-35-173-232-130.compute-1.amazonaws.com:5000/predict?msg=HelloWorld>

now build Docker container version:

```
$ sudo docker image build -t "echo_service" .
```

```
$ sudo docker images
```

```
$ sudo docker run -d -p 80:5000 echo_service
```

```
$ sudo docker ps
```

refresh browser window. fails; refuses to connect.

solution: don't include port 5000 anymore! this time, want to connect on the default http port (80)

<http://ec2-35-173-232-130.compute-1.amazonaws.com/predict?msg=HelloWorld>

--> success!