

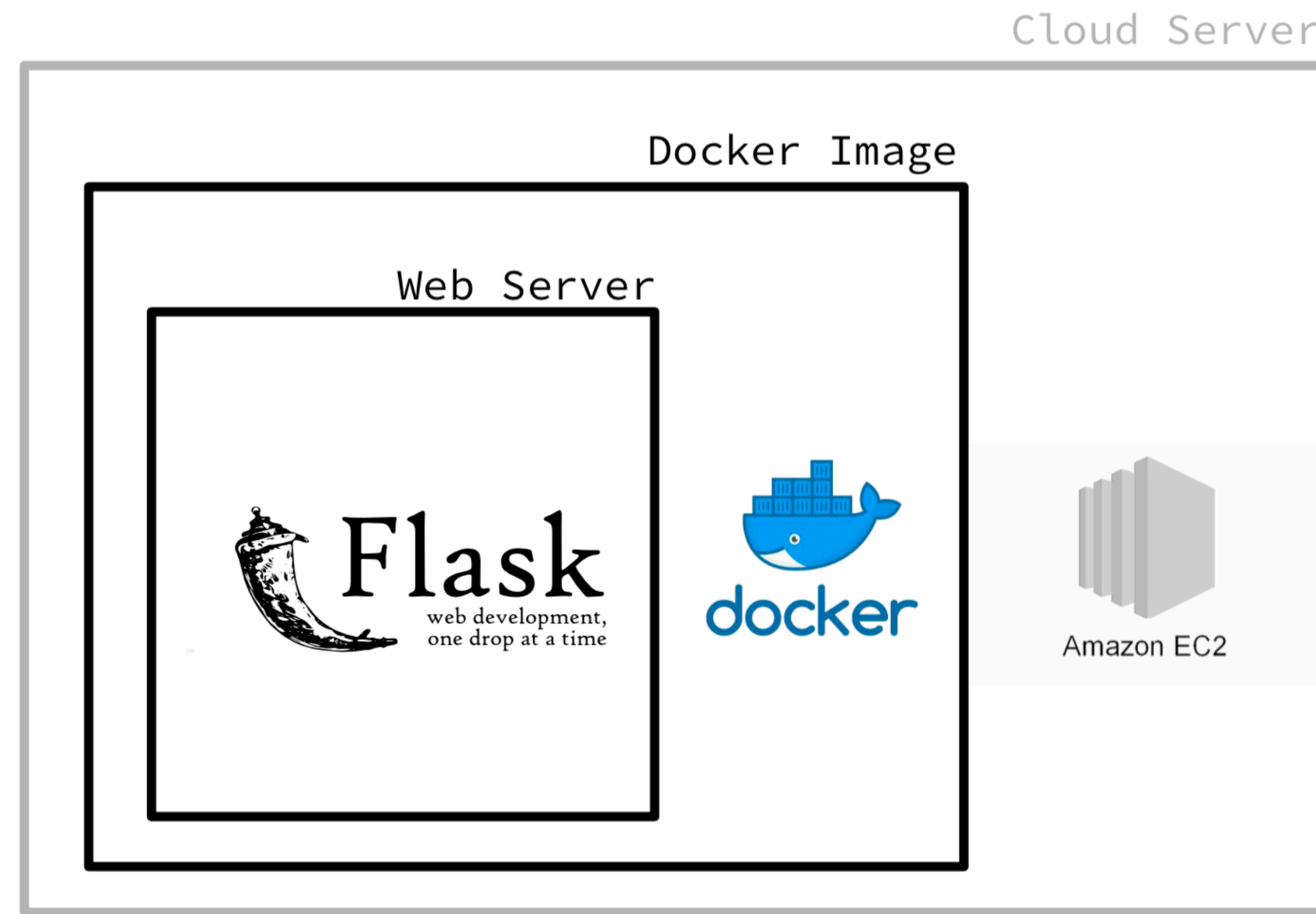
What are we doing here?

Created a machine learning model...now I want someone to use it!

Lets start to think about how we can deploy our model so someone else can access it (without knowing much of what's going on underneath)

Expose a web API (locally for today)

What are we going to build?



What is an API anyways?

Technically, API stands for **Application Programming Interface**. At some point or another, most large companies have built APIs for their customers, or for internal use.

When you type www.facebook.com into your browser, a request goes out to Facebook's remote server. Once your browser receives the response, it interprets the code and displays the page.

An API isn't the same as the remote server—rather it is the part of the server that receives requests and sends responses.

When a company offers an API to their customers, it just means that they've built a set of dedicated URLs that return pure data responses—meaning the responses won't contain the kind of presentational overhead that you would expect in a graphical user interface like a **website**. (thanks <https://medium.freecodecamp.org/what-is-an-api-in-english-please-b880a3214a82>)

REST API

A REST API defines a set of functions which developers can perform requests and receive responses via HTTP protocol such as GET and POST.

REST stands for **Representational state transfer** which essentially refers to a style of web architecture that has many underlying characteristics and governs the behavior of clients and servers.

An API can be considered “RESTful” if it has the following features (main features of R):

Client-server – The client handles the front end the server handles the backend and can both be replaced independently of each other.

Stateless – No client data is stored on the server between requests and session state is stored on the client.

Cacheable – Clients can cache response (just like browsers caching static elements of a web page) to improve performance

What is Flask?

A microframework for Python, meaning it has little to no dependencies to external libraries.

It really is a web framework providing tools, libraries and technologies to build web applications. (in our case an API)

Flask can create a REST API that allows you to send data, and receive a prediction as a response.



What are we doing with Flask?

We will create routes for our api that receives different requests (GET and POST), specifically a POSTing json data

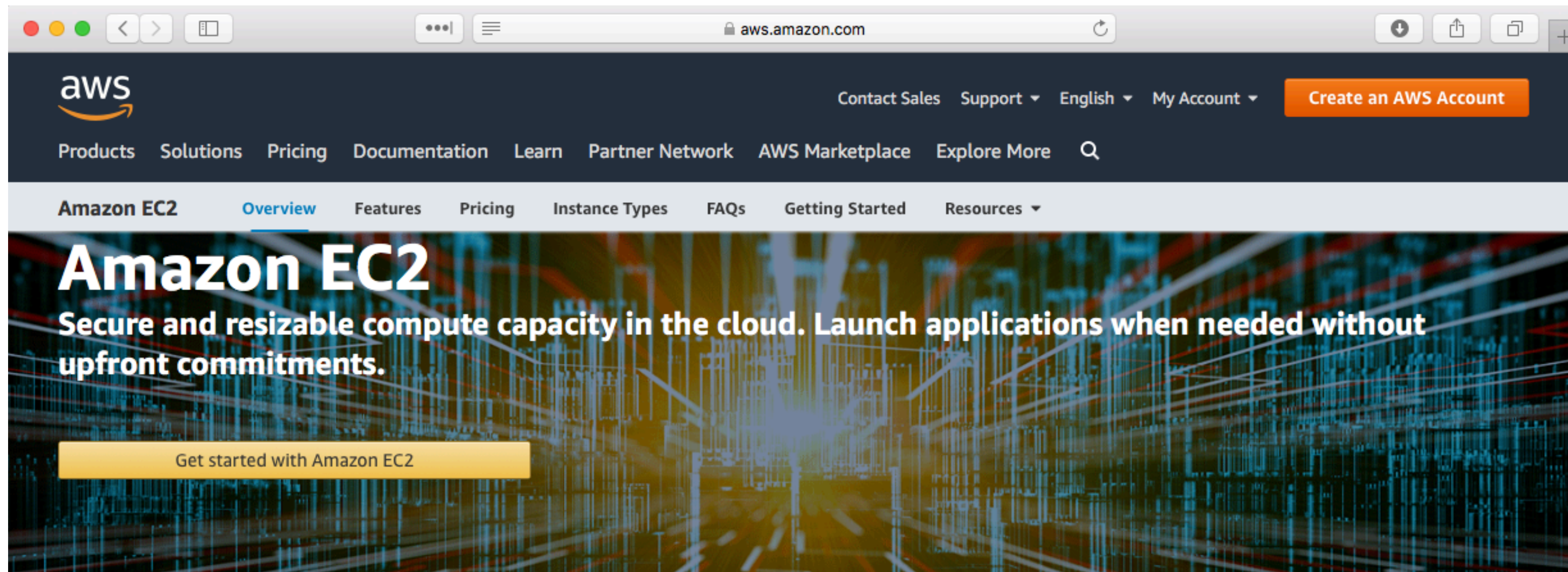
Once our flask server receives json data it will apply a function and then return some value also in json format

What are we doing with Flask?

Last week we created a locally deployed API to help us understand the process through which we could . Our local API was a Flask app contained in a docker container...the portability of the docker container will come in handy here.

Now let's move this idea to a cloud based service, an Amazon Web Service ec2 instance (we get plenty of free resources!), where people outside our local machines can access our API

It should be noted that this is still simply to give us an idea about the deployment process, but we will be still falling a bit short of full-on productionalization. (no load balancing, no security, no testing, etc.)



Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate them from common failure scenarios.

Try Amazon EC2 for Free

AWS Free Tier includes 750 hours of Linux and Windows t2.micro instances each month for one year. To stay within the Free Tier, use only EC2 Micro instances.

[View AWS Free Tier details >>](#)

(<https://aws.amazon.com/ec2/>)

You will need an *AWS* account to get access to the necessary resources.

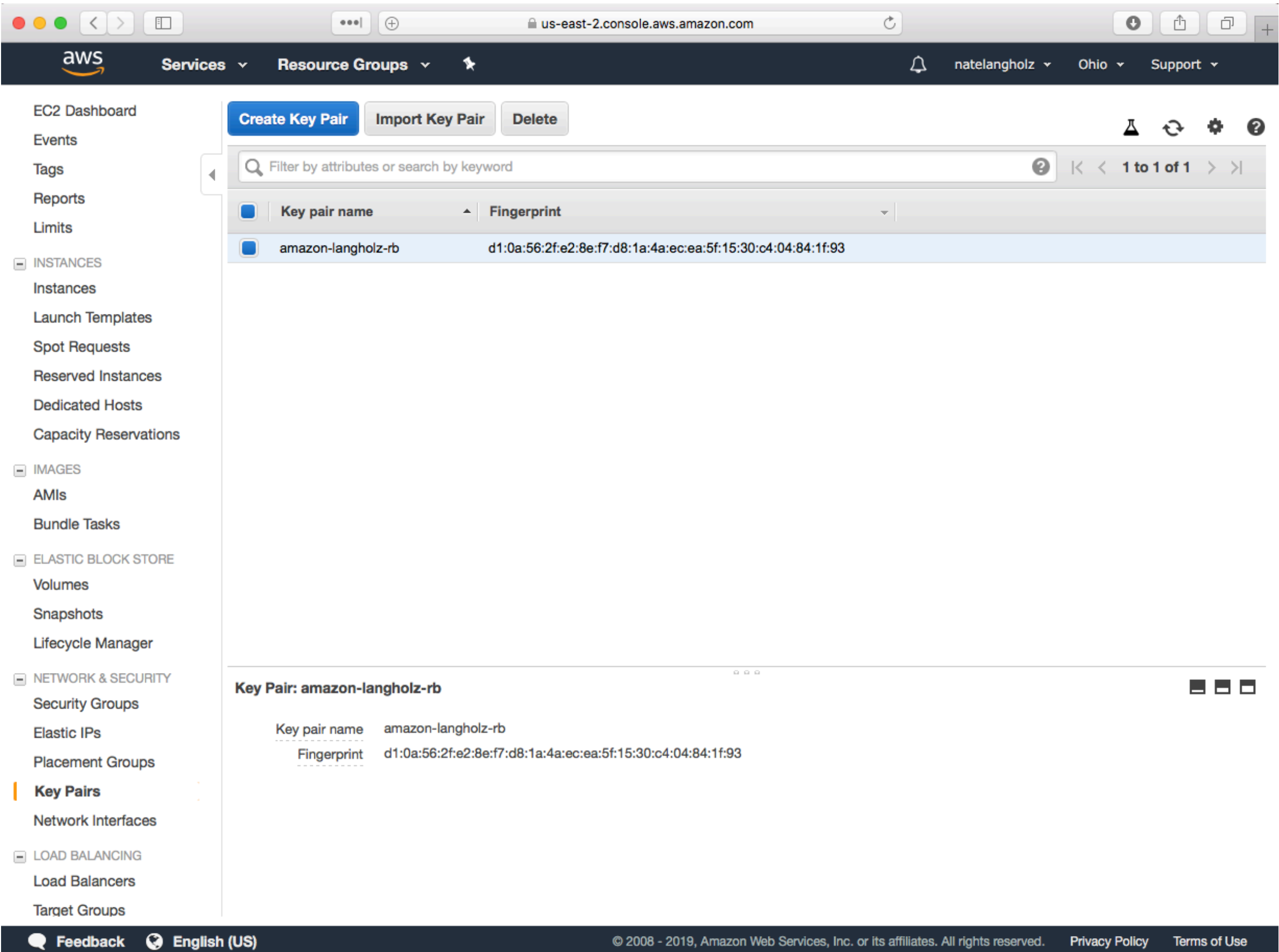
Create one now; make it a personal account.

You will need to verify your email (and maybe enter a credit card; won't be billed)

Once you have an account let's start the process. Login.

Look for the 'Key Pairs' section under Network & Security. Create one with a simple name which will download a `.pem` file. This is your key.

Save it somewhere with an easy path. For now I've put mine under my week-7 directory.



`Launch` an instance

The screenshot displays the AWS Management Console interface for the US East (Ohio) region. The left-hand navigation pane lists various services, with 'EC2 Dashboard' selected. The main content area is divided into several sections: 'Resources' (showing counts for Running Instances, Elastic IPs, etc.), 'Create Instance' (with the 'Launch Instance' button circled in red), 'Service Health' (showing status for US East (Ohio) and its availability zones), 'Scheduled Events', 'Account Attributes', 'Additional Information', and 'AWS Marketplace'. The bottom of the console features a footer with 'Feedback', 'English (US)', and copyright information.

Resources

You are using the following Amazon EC2 resources in the US East (Ohio) region:

- 0 Running Instances
- 0 Elastic IPs
- 0 Dedicated Hosts
- 0 Snapshots
- 1 Volumes
- 0 Load Balancers
- 1 Key Pairs
- 2 Security Groups
- 0 Placement Groups

Create Instance

To start using Amazon EC2, you will want to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance

Service Health

Service Status:

- US East (Ohio):

Availability Zone Status:

- us-east-2a: Availability zone is operating normally
- us-east-2b: Availability zone is operating normally
- us-east-2c: Availability zone is operating normally

Scheduled Events

US East (Ohio):

- No events

Account Attributes

Supported Platforms

- VPC

Default VPC

- vpc-fd607995

Resource ID length management

Console experiments

Additional Information

- Getting Started Guide
- Documentation
- All EC2 Resources
- Forums
- Pricing
- Contact Us

AWS Marketplace

Find free software trial products in the AWS Marketplace from the [EC2 Launch Wizard](#). Or try these popular AMIs:

- [Barracuda CloudGen Firewall for AWS - PAYG](#)
- By Barracuda Networks, Inc.
- Rating ★★★★★
- Starting from \$0.60/hr or from \$4,599/yr (12% savings) for software + AWS usage fees
- [View all Infrastructure Software](#)
- [Matillion ETL for Amazon Redshift](#)

Feedback **English (US)**

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

Choose an Amazon Machine Instance from the options.

The screenshot shows the AWS Management Console interface for selecting an Amazon Machine Image (AMI). The page is titled "Step 1: Choose an Amazon Machine Image (AMI)" and includes a "Cancel and Exit" link. Below the title, there is a search bar and a list of AMIs. The AMIs listed are:

- Amazon Linux 2 AMI (HVM), SSD Volume Type** - ami-02bcbb802e03574ba (64-bit x86) / ami-06a134062219ad132 (64-bit Arm). This AMI is marked as "Free tier eligible".
- Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type** - ami-0cd3dfa4e37921605. This AMI is marked as "Free tier eligible" and is highlighted by a red oval.
- Red Hat Enterprise Linux 8 (HVM), SSD Volume Type** - ami-05220a0e7fce3d1 (64-bit x86) / ami-080877849cddac64b (64-bit Arm). This AMI is marked as "Free tier eligible".
- SUSE Linux Enterprise Server 15 (HVM), SSD Volume Type** - ami-0eb9f58db22854f8f (64-bit x86) / ami-064a69af69b77fa05 (64-bit Arm). This AMI is marked as "Free tier eligible".

The sidebar on the left shows navigation options: "My AMIs", "AWS Marketplace", "Community AMIs", and a "Free tier only" filter. The bottom of the page includes a footer with "Feedback", "English (US)", and copyright information.

Using the Amazon Linux AMI because its Free tier eligible and the repos include Docker. (I believe the Linux 2 AMI would also work)

Configure Security Group; click `add rule`; the new rule will allow inbound requests

The screenshot shows the AWS Management Console interface for configuring a security group. The breadcrumb trail indicates the current step is "6. Configure Security Group". The page title is "Step 6: Configure Security Group". Below the title, there is a description of security groups and a link to "Learn more" about Amazon EC2 security groups. The "Assign a security group" section has two radio buttons: "Create a new security group" (selected) and "Select an existing security group". The "Security group name" field is "launch-wizard-2" and the "Description" field is "launch-wizard-2 created 2019-05-14T07:56:38.015-07:00". Below this is a table for adding rules. The table has columns: Type, Protocol, Port Range, Source, and Description. There are two rows of rules. The first row is "SSH" with protocol "TCP" and port range "22". The second row is "Custom TCP Rule" with protocol "TCP" and port range "5000". Both rules have a source of "Anywhere" (0.0.0.0/0, ::/0) and a description of "e.g. SSH for Admin Desktop". A red circle highlights the "Add Rule" button and the rule configuration table. Below the table is a warning message: "Warning: Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only." At the bottom right, there are three buttons: "Cancel", "Previous", and "Review and Launch".

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group ☐ Select an **existing** security group

Security group name:

Description:

Type <small>i</small>	Protocol <small>i</small>	Port Range <small>i</small>	Source <small>i</small>	Description <small>i</small>
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
Custom TCP Rule	TCP	5000	Anywhere 0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

Warning

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

[Feedback](#) [English \(US\)](#) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

Pick `Custom TCP Rule`
Enter 5000 into the Port Range (match whatever port you have used in your local app)
Change Source to `Anywhere`

Review and Launch

You'll notice the warning about improving security. For what we're doing this should be okay not to worry about.

us-east-2.console.aws.amazon.com

aws

Services

Resource Groups

natelangholz

Ohio

Support

1. Choose AMI

2. Choose Instance Type

3. Configure Instance

4. Add Storage

5. Add Tags

6. Configure Security Group

7. Review

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

Improve your instances' security. Your security group, launch-wizard-2, is open to the world.

Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.

You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. [Edit security groups](#)

AMI Details

Free tier eligible

Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-0cd3dfa4e37921605

The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.

Root Device Type: ebs Virtualization type: hvm

[Edit AMI](#)

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

[Edit instance type](#)

Security Groups

Security group name

launch-wizard-2

Description

launch-wizard-2 created 2019-05-14T08:01:05.030-07:00

Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ	Description ⓘ
SSH	TCP	22	0.0.0.0/0	
Custom TCP Rule	TCP	5000	0.0.0.0/0	

[Edit security groups](#)

Cancel

Previous

Launch

Feedback

English (US)

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

Should be okay to Launch at this point

After clicking launch you should see a pop-up looking for confirmation of your
`key-pair`

Use the name of your key-pair that we generated earlier

Now launch the AMI and view instances

ec2 Dashboard

EC2 Dashboard

Events

Tags

Reports

Limits

INSTANCES

Instances

Launch Templates

Spot Requests

Reserved Instances

Dedicated Hosts

Capacity Reservations

IMAGES

AMIs

Bundle Tasks

ELASTIC BLOCK STORE

Volumes

Snapshots

Lifecycle Manager

NETWORK & SECURITY

Security Groups

Elastic IPs

Placement Groups

Key Pairs

Network Interfaces

LOAD BALANCING

Load Balancers

Target Groups

Launch Instance

Connect

Actions

Filter by tags and attributes or search by keyword

	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
<input type="checkbox"/>		i-04d5e340fa0323cc7	t2.micro	us-east-2c	running	Initializing	None	ec2-3-16-214-255.us-east-2.compute.amazonaws.com

Instance: i-04d5e340fa0323cc7

Public DNS: ec2-3-16-214-255.us-east-2.compute.amazonaws.com

Description

Status Checks

Monitoring

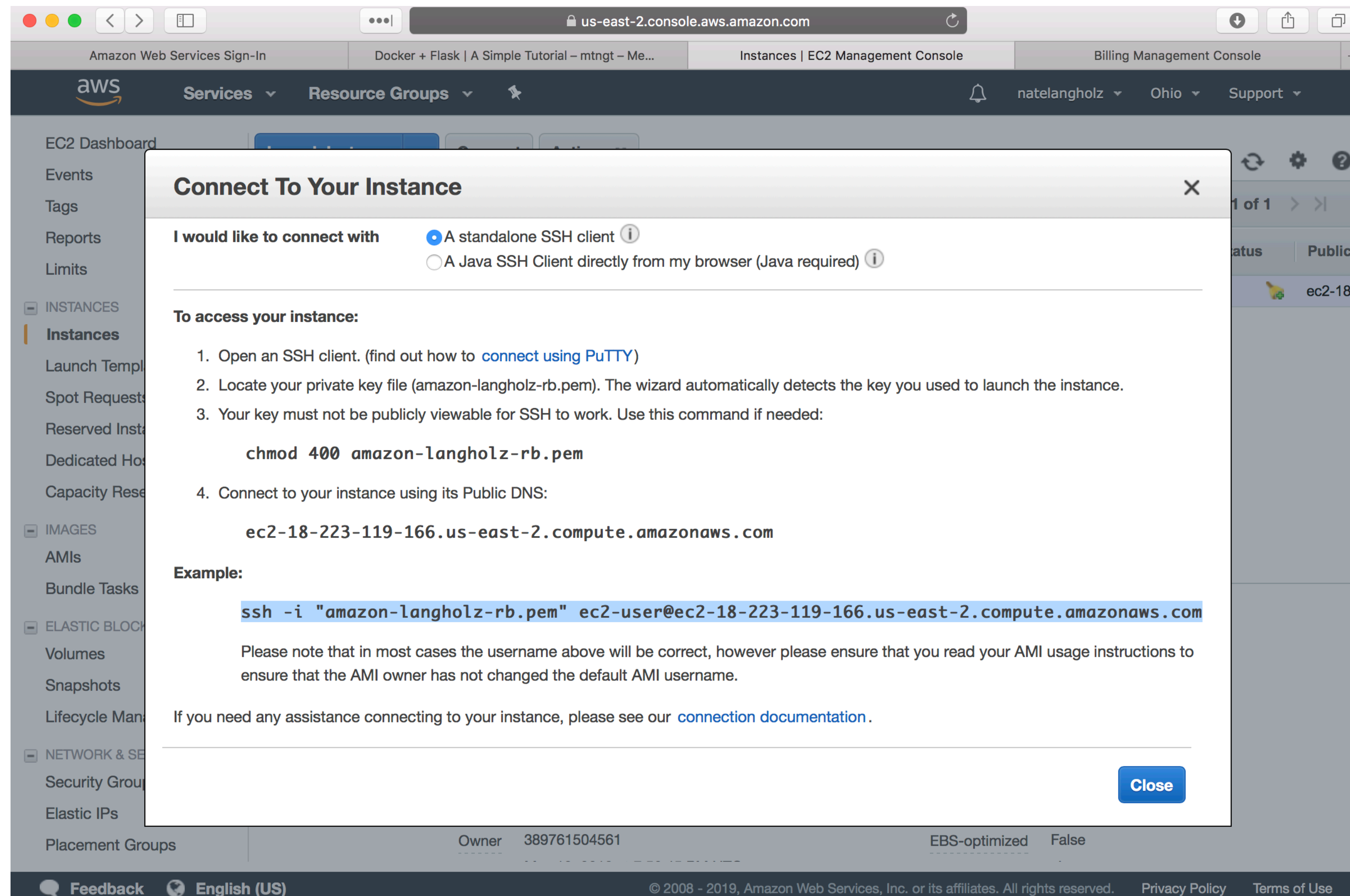
Tags

Instance ID	i-04d5e340fa0323cc7	Public DNS (IPv4)	ec2-3-16-214-255.us-east-2.compute.amazonaws.com
Instance state	running	IPv4 Public IP	3.16.214.255
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs		Private DNS	ip-172-31-33-194.us-east-2.compute.internal
Availability zone	us-east-2c	Private IPs	172.31.33.194

Observe that your Instance state is running

Also note your Public DNS, as this is where you will access this instance

Connect to your instance; this is an overview of the instructions. I'll try to detail them a bit more. (See these by clicking `Connect` in your dashboard)



Remember our Key?

PEM or Privacy Enhanced Mail is a Base64 encoded DER certificate. PEM certificates are frequently used for web servers as they can easily be translated into readable data using a simple text editor. Generally when a PEM encoded file is opened in a text editor, it contains very distinct headers and footers. (<https://support.quovadisglobal.com/kb/a37/what-is-pem-format.aspx>)

We need to to change the permissions on the key to private

For Mac and Linux users

```
chmod 400 key-file.pem
```


For Windows (if above doesn't work)...

Windows solutions to chmod and ssh and scp

Let's ssh into our AMI

key-file.pem

ec2-user@public-dns



```
ssh -i "amazon-langholz-rb.pem" ec2-user@ec2-18-223-119-166.us-east-2.compute.amazonaws.com
```

Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. Typical applications include remote command-line login and remote command execution, but any network service can be secured with SSH. (https://en.wikipedia.org/wiki/Secure_Shell)

For windows if you are having trouble with the ssh command here are resources to help with connecting (<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstances.html>)

What does it look like if correctly connected?

```

week-7 — ec2-user@ip-172-31-33-194:~ — ssh -i amazon-langholz-rb.pem ec2-user@ec2-3-16-214-255.us-east-2.compute.amazonaws.com — 167x49
NA-nlanghol-A02:week-7 nlangholz$ ssh -i "amazon-langholz-rb.pem" ec2-user@ec2-3-16-214-255.us-east-2.compute.amazonaws.com
The authenticity of host 'ec2-3-16-214-255.us-east-2.compute.amazonaws.com (3.16.214.255)' can't be established.
ECDSA key fingerprint is SHA256:f1B8nriZ0952jcSudPhOF8SiRser92GnVJrYBb6P8u0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-3-16-214-255.us-east-2.compute.amazonaws.com,3.16.214.255' (ECDSA) to the list of known hosts.
Last login: Tue May 14 03:38:27 2019 from cpe-172-91-122-53.socal.res.rr.com

    __|__ | )
   _|_ ( /   Amazon Linux AMI
  ___\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
[ec2-user@ip-172-31-33-194 ~]$ ls
flask-app-small
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
total 36
drwx----- 4 ec2-user ec2-user 4096 May 14 04:46 .
drwxr-xr-x 3 root      root     4096 May 13 16:49 ..
-rw----- 1 ec2-user ec2-user 2197 May 14 04:59 .bash_history
-rw-r--r-- 1 ec2-user ec2-user   18 Aug 30  2017 .bash_logout
-rw-r--r-- 1 ec2-user ec2-user  193 Aug 30  2017 .bash_profile
-rw-r--r-- 1 ec2-user ec2-user  124 Aug 30  2017 .bashrc
drwxrwxr-x 4 ec2-user ec2-user 4096 May 14 04:17 flask-app-small
drwx----- 2 ec2-user ec2-user 4096 May 13 16:49 .ssh
-rw-rw-r-- 1 ec2-user ec2-user  209 May 14 04:23 .wget-hsts
[ec2-user@ip-172-31-33-194 ~]$ 
[ec2-user@ip-172-31-33-194 ~]$ 
```

olic-dns

Once inside, this is a Linux machine so everyone should be operating under the same build

Time to install Docker and Docker-compose

Update the environment installed packages and cache

```
sudo yum update -y
```

Install Docker

```
sudo yum install -y docker
```

Add the ec2-user to the docker group so you can execute Docker commands without sudo

```
sudo usermod -a -G docker ec2-user
```

this pulls docker-compose from GitHub

```
sudo curl -L https://github.com/docker/compose/releases/download/1.21.0/docker-compose-`uname -s`-`uname -m` | sudo tee /usr/local/bin/docker-compose > /dev/null
```

```
sudo chmod +x /usr/local/bin/docker-compose
```

(honestly not sure if below is needed; try anyways)

```
ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```


Install docker-compose

```
docker-compose
```

Start the docker service

```
sudo service docker start  
sudo chkconfig docker on
```

Check docker-compose version

```
docker-compose --version
```

To leave your AMI, simply type ``exit``

Let's add some files from our local machine through secure copy

```
scp -i "amazon-langholz-rb.pem" -r flask-api-small/ ec2-  
user@ec2-3-14-13-135.us-east-2.compute.amazonaws.com:/  
home/ec2-user
```

SSH back into your AMI instance and run `ls`. You should see your directory; proceed as usual with docker.

Change to the docker directory, run `docker-compose up -d`

Then test the connection there in your instance using a curl command

To stop your API, within your AMI stop your docker container as usual.

Either

`Docker-compose stop`

or

`Docker container ls`

`Docker container kill <container-name>`

Then exit your AMI by simply typing ``exit``

Another easy way to create an easy deployment is through GitHub.

Remember this is a linux environment with bash setup so we can use these commands.

We have a nice completed example of homework 3 in a standalone repository.
We can get a zip file through

```
wget https://github.com/tannerkoscinski/MTCars-Flask-API/archive/master.zip
```

Unzip using

```
unzip *.zip
```

And then remove the zip file as we no longer need it

```
rm master.zip
```

Let's take a look at making predictions on my api from your machines

```
rm -rf MTCars-Flask-API-master/  
sudo rm -rf prediction.cpython-37.pyc
```

Lastly to remove everything from that repository

```
sudo rm -rf MTCars-Flask-API-master/
```

STOP your instance from continuing to run when not in use!!!

EC2 Dashboard

Events

Tags

Reports

Limits

INSTANCES

Instances

Launch Templates

Spot Requests

Reserved Instances

Dedicated Hosts

Capacity Reservations

IMAGES

AMIs

Bundle Tasks

ELASTIC BLOCK STORE

Volumes

Snapshots

Lifecycle Manager

NETWORK & SECURITY

Security Groups

Elastic IPs

Placement Groups

Launch Instance

Connect

Actions

Filter by tags and attributes or search

	Name	Instance ID	Availability Zone	Instance State	Status Checks	Alarm Status	Public
<input type="checkbox"/>		i-04d5e340fa032cc7	us-east-2c	running	2/2 checks ...	None	ec2-18-223-119-166.us-east-2.compute.amazonaws.com

Instance: i-04d5e340fa032cc7

Public DNS: ec2-18-223-119-166.us-east-2.compute.amazonaws.com

Description

Status Checks

Monitoring

Tags

Instance ID	i-04d5e340fa032cc7	Public DNS (IPv4)	ec2-18-223-119-166.us-east-2.compute.amazonaws.com
Instance state	running	IPv4 Public IP	18.223.119.166
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs		Private DNS	ip-172-31-33-194.us-east-2.compute.internal
Availability zone	us-east-2c	Private IPs	172.31.33.194

Feedback

English (US)

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use