



# AWS Immersion Day

## Compute Hands-On Lab

*Getting Started with Linux on Amazon EC2, Notebooks,  
and Batch*

---

## Lab Overview

Amazon Elastic Compute Cloud ([Amazon EC2](#)) is a web service that provides resizable compute capacity in the cloud. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. 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.

An [Amazon SageMaker notebook instance](#) is a machine learning (ML) compute instance running the [Jupyter](#) Notebook App. Amazon SageMaker manages creating the instance and related resources. Use Jupyter notebooks in your notebook instance to prepare and process data, write code to train models, deploy models to Amazon SageMaker hosting, and test or validate your models.

Scientists and engineers frequently use Jupyter notebooks for data wrangling, machine learning, and ETL using Spark. Notebooks foster reproducibility in data experiments, and enable sharing among colleagues.

[AWS Batch](#) enables developers, scientists, and engineers to easily and efficiently run hundreds of thousands of batch computing jobs on AWS. AWS Batch dynamically provisions the optimal quantity and type of compute resources (e.g., CPU or memory optimized instances) based on the volume and specific resource requirements of the batch jobs submitted.

We will take a look at a several of ways of exercising compute resources at AWS in this lab. You will launch a cloud-hosted Jupyter notebook, and create a traditional compute host using Amazon EC2. Those who are adventurous, and have time remaining will also submit a batch job using AWS Batch.

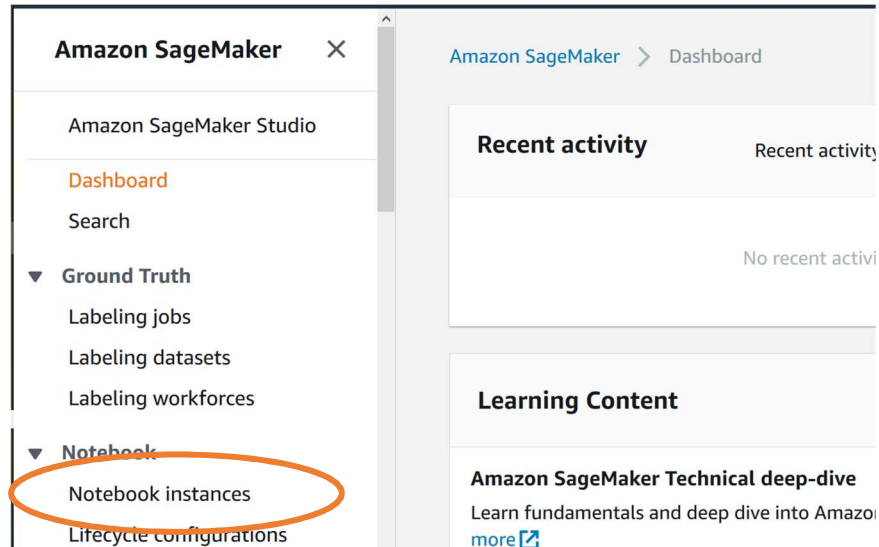
## Important - Clean up when complete

In these labs, we will create resources in your AWS account. While these are running, you are accruing charges. When you are finished, be sure to delete these resources, as described at the end of the lab.

## Lab One: Amazon SageMaker Notebook Instances

SageMaker, Glue, and Elastic Map Reduce services provide Jupyter managed notebooks in the AWS cloud. In this lab, you will launch a SageMaker Notebook, and associate it with a Github repository.

1. Sign into the AWS Management Console and open the Amazon SageMaker console at <https://console.aws.amazon.com/sagemaker>.
2. In the left columns, select "Notebook Instances"



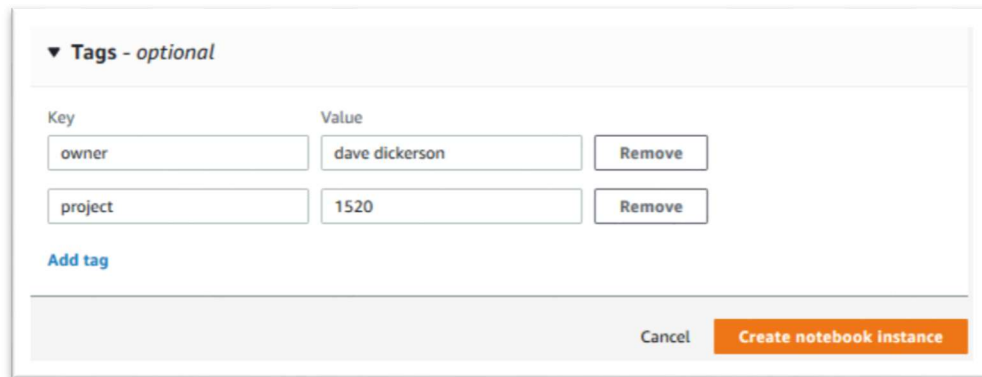
3. Click "Create Notebook Instance" button
4. Give the notebook a name ... perhaps, labtest-yourname
5. Select an ml.t2.medium instance type. While you are here, take a look at the other instance types available. They come in different sizes (CPU, RAM) depending on your needs. There are instances balanced, heavy in CPU, RAM heavy, or some with GPUs (or fractions of GPUs). Because we are running a simple notebook, we're picking the smallest available instance.
6. We will talk about IAM Roles in a later lab. For now, allow the service to create a role.
7. Allow the notebook to clone a public Git repository. For the git repository URL enter:

`https://github.com/jrjohansson/scientific-python-lectures.git`

The image shows a section of the Amazon SageMaker Studio interface titled 'Git repositories - optional'. Below this title is a section for 'Default repository'. It includes a 'Repository' label and a description: 'Jupyter will start in this repository. Repositories are added to your home directory.' There is a dropdown menu with the selected option 'Clone a public Git repository to this notebook instance only'. Below the dropdown is a 'Git repository URL' label and a description: 'Clone a repository to use for this notebook instance only.' A text input field contains the URL 'https://github.com/jrjohansson/scientific-python-lectures.git'.

There are a large number of Jupyter Notebook samples on [Jupyter's Wiki](#).

8. Tags are always a good idea. They are used to keep track of who created resources, which project it is associated, and other metadata. Add two tags:
  1. Key: owner Value: YourName
  2. Key: project Value: AWSlab



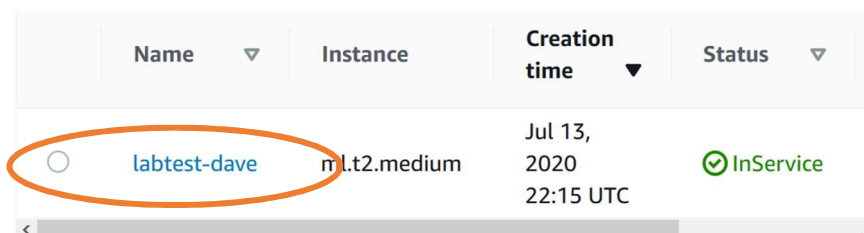
▼ Tags - optional

Key	Value	
owner	dave dickerson	Remove
project	1520	Remove

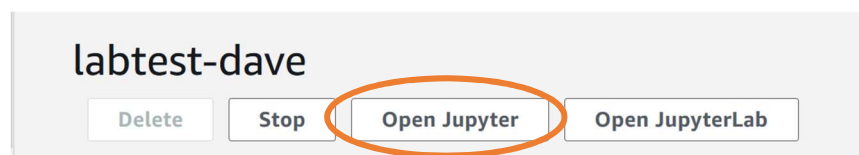
[Add tag](#)

Cancel [Create notebook instance](#)

9. Click **"Create notebook instance"**. It takes over 5 minutes to create a notebook. Please move on to the next lab, EC2 instances. **Check back in a few minutes to verify that the instance completed.** When the notebook becomes available continue with step 10, below.
10. **Welcome back.** Now that the notebook is running, click the notebook name to open it. Then click "Open Jupyter".



	Name ▼	Instance	Creation time ▼	Status ▼
<input type="radio"/>	labtest-dave	m1.t2.medium	Jul 13, 2020 22:15 UTC	InService



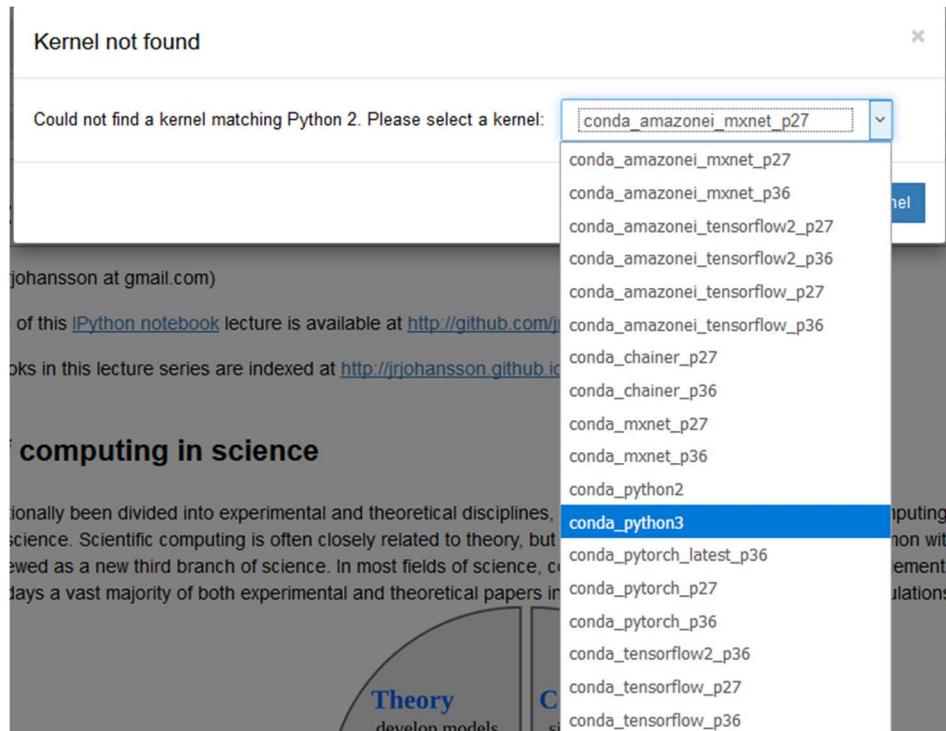
labtest-dave

[Delete](#) [Stop](#) [Open Jupyter](#) [Open JupyterLab](#)

You should see the content from the public github.com repo. If you're familiar with Jupyter, take a run through the code. Just take a couple of minutes - we have two more labs to go ...

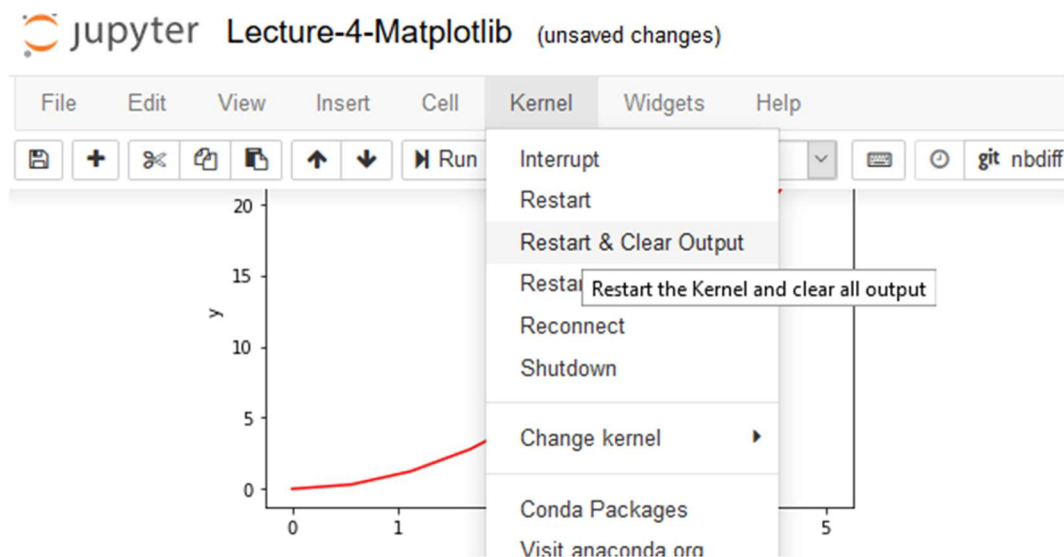
11. Select a notebook from the list, perhaps starting with **Lecture-4-Matplotlib.ipynb**.

It will likely ask you to select a Python Kernel. If it does, chose `conda_python3`.



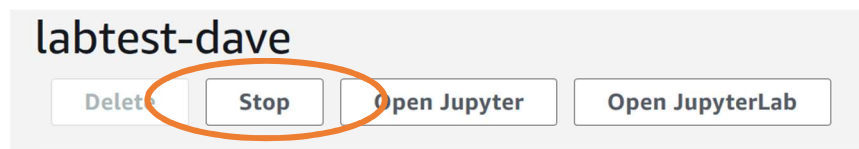
Navigating the notebook is as easy as hitting the “Run” button, or highlighting a cell and typing SHIFT/ENTER.

It's often convenient to restart the kernel and start with a fresh notebook. From the Kernel menu, clear the notebook.

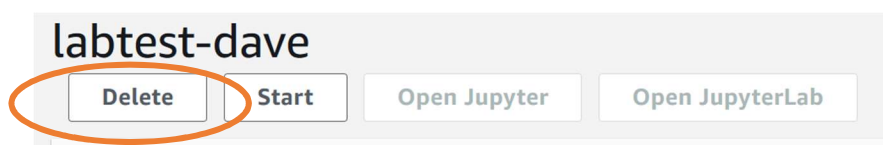


With Jupyter notebooks you use [Markdown](#) to merge notes and your code. Notebooks are a convenient tool for reproducible research and for sharing your work with others.

12. An important feature of the cloud is you only pay for what you use. In the case of notebooks, this means that while the notebook is running, you are [paying](#) for it. With notebooks, and most resources, it is best practice to turn off or delete the notebook when not in use. Navigate to the [SageMaker](#) console and **stop, and delete the notebook**. Best practice is to stop notebooks evenings and weekend when you're not actively using them.



13. Now delete the notebook, and continue to the EC2 lab.

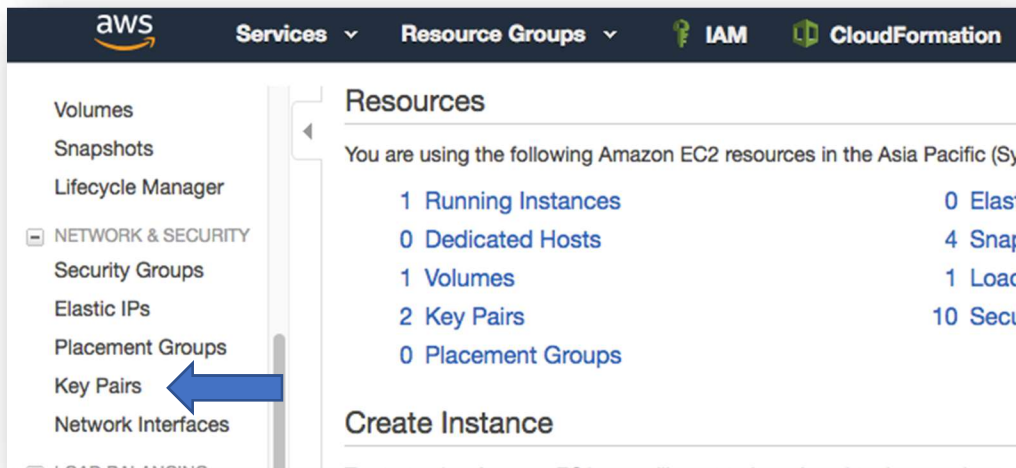


## Lab Two: AWS Elastic Compute Cloud - EC2

### Create a new Key Pair

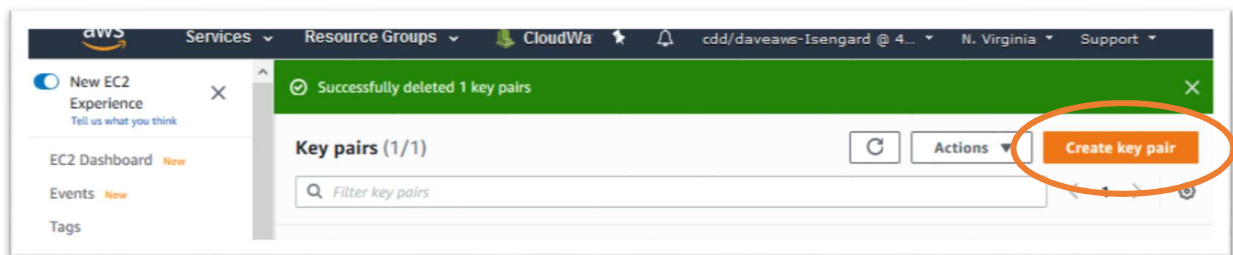
In this lab, you will create an EC2 instance as a web server. To manage the instance, you need to be able to connect to it via SSH. The following steps outline how to create a unique SSH keypair for this purpose.

1. Sign into the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2>.
2. In the upper-right corner of the AWS Management Console, confirm you are in the desired AWS region (e.g., N. Virginia).
3. Click on **Key Pairs** in the NETWORK & SECURITY section near the bottom of the leftmost menu. This will display a page to manage your SSH key pairs.

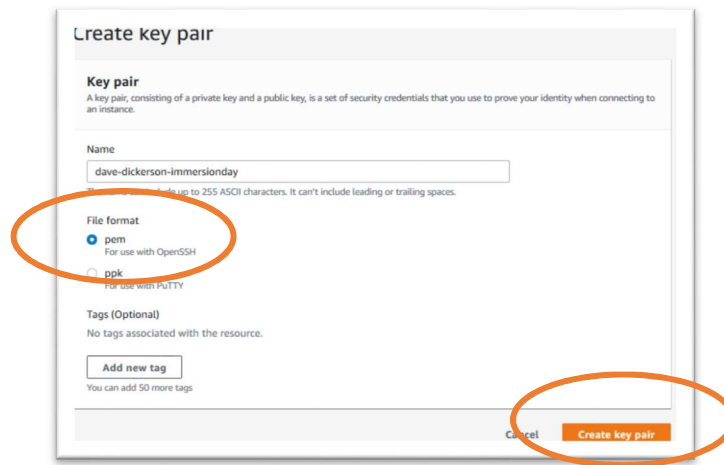


4. To create a new SSH key pair, click the **Create Key Pair** button at the top of the browser window.

Note - another option is to create a key pair using ssh-keygen and upload the public key to AWS. If you'd prefer to generate your own keys, see [the documentation](#).



5. In the resulting pop up window, type *[First Name]-[Last Name]-ImmersionDay* into the **Key Pair Name:** text box and click **Create**. Windows users who use the Putty terminal should select ppk, MacOS, WSL, and Linux users should select pem format.



6. The page prompts you to download the file “[First Name]-[Last Name]-ImmersionDay.pem” to the local drive. Follow your browser instructions to save the file to the default download location.
7. Remember the full path to this .pem file you just downloaded. This file contains your private key for future SSH connections.

## Launch a Web Server Instance

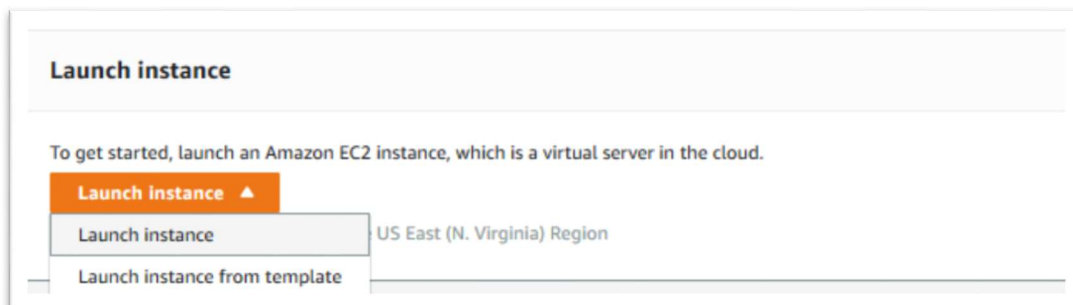
In this example, we will launch an Amazon Linux 2 instance, bootstrap Apache/PHP, and install a basic web page that will display information about our instance.

Sign into your AWS Management Console and choose EC2 from the Services menu.



*Upon logging into your AWS Console, you should ALWAYS check which region you are operating in. This can be found in the top right of your Console window.*

8. Click on Launch Instance





- In the **Quick Start** section, select the first Amazon Linux 2 AMI for 64-bit (x86) architecture and click **Select**. Note that the ami-xxxxxxx label and specific versions of the installed package may be different than in the image below.



- In the Step 2. *Choose an Instance Type*, select the **t2.micro** instance size and click **Next: Configure Instance Details**.

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	General purpose	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes

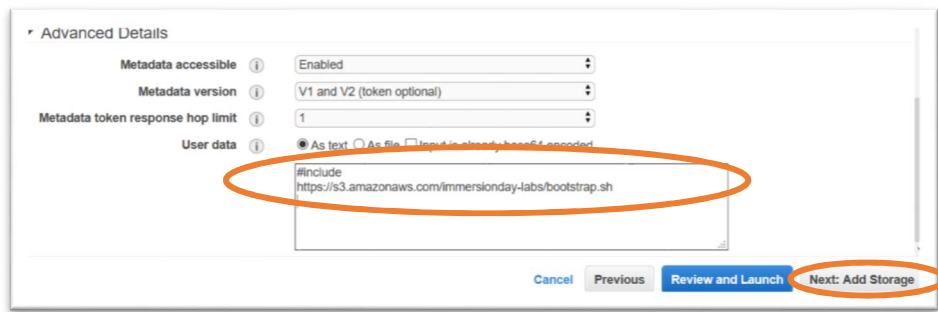
Cancel Previous **Review and Launch** Next: Configure Instance Details

- On Step 3. **Configure Instance Details** page, expand the **Advanced Details** section located at the bottom of the page, then, copy/paste the script below into the **User Data** field. This shell script will install Apache & PHP, start the web service, and deploy a simple web page. Click **Next: Add Storage**.



*'User data' is a method for bootstrapping your instance - Any code placed here will be executed the first time an instance is launched.  
The user data in this lab installs a PHP-based web server, with HTML content.*

```
#include
https://s3.amazonaws.com/immersionday-labs/bootstrap.sh
```



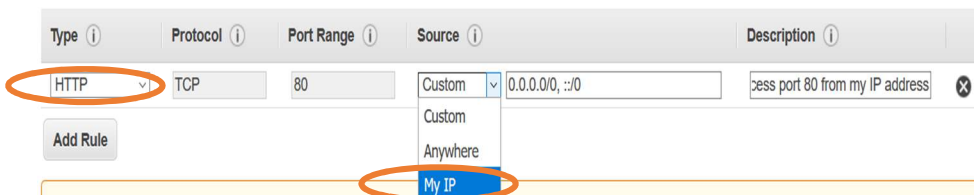
If you are curious about the bootstrap script, you can review the script.

wget <https://s3.amazonaws.com/immersionday-labs/bootstrap.sh> -or- browse to that link.

12. On this page you have the ability to modify or add storage and disk drives to the instance. For this lab, we will simply accept the storage defaults and click **Next: Add Tags**.
13. Here, you can choose a “friendly name” for your instance by clicking ‘Add Tag’, and entering “Name” for the Key part and “[Your Name] Web Server” for the Value part. This Name key, more correctly known as a **tag**, will appear in the console once the instance launches. It makes it easy to keep track of running machines in a complex environment. Click **Next: Configure Security Group**.
14. You will be prompted to create a new security group, which will be your firewall rules. On the assumption that we are building out a Web server, name your new security group “[Your Name] Web Tier”. Click **Add Rule**.
15. For this lab, we will not be accessing the instance using SSH, so you can delete that rule. Click the X to delete the SSH Rule.

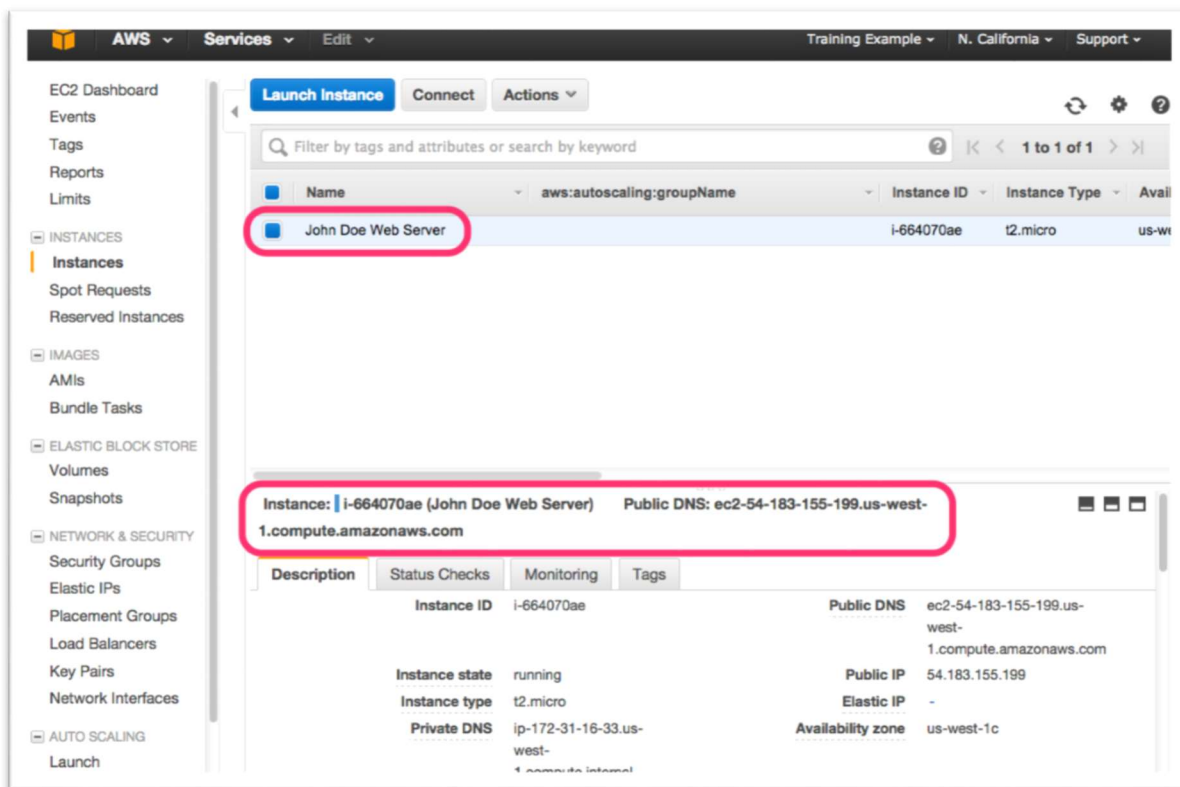


16. We will be using HTTP, to access the web server. Add an HTTP Type from source “My IP”.



17. Click the **Review and Launch** button after configuring the security group.

18. Review your configuration and choices, and then click **Launch**.
19. Select the key pair that you created in the beginning of this lab from the drop-down and check the "I acknowledge" checkbox. Then click the **Launch Instances** button. Your instance will now be starting, which may take a moment.
20. Click the **View Instances** button in the lower right hand portion of the screen to view the list of EC2 instances. Once your instance has launched, you will see your Web Server as well as the Availability Zone the instance is in, and the publicly routable DNS name.
21. Click the checkbox next to your web server to view details about this EC2 instance.



## Browse the Web Server

1. Wait for the instance to pass the Status Checks to finish loading.

<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
<input type="checkbox"/>	John Doe Web Server	i-664070ae	t2.micro	us-west-1c	<span style="color: green;">●</span> running	Initializing

Finished initializing

<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
<input type="checkbox"/>	John Doe Web Server	i-664070ae	t2.micro	us-west-1c	<span style="color: green;">●</span> running	<span style="color: green;">✓</span> 2/2 checks passed

Open a new browser tab and browse the Web Server by entering the EC2 instance's Public DNS name into the browser. The EC2 instance's Public DNS name can be found in the console by reviewing the "Public DNS" name line highlighted above.

You should see a website that looks like the following:



LOAD TEST
RDS

Meta-Data	Value
InstanceId	i-664070ae
Availability Zone	us-west-1c

Current CPU Load: 0%



*If you don't see the web page (and you've waited a sufficient time for the instance to boot), try rebooting the instance via the console. Can you find it??*

**Great Job! You have deployed a server and launched a web site in a matter of minutes!!**

## Lab Three: AWS Batch

This lab is extra credit. If you have time and are interested in batch processing in the cloud, please proceed.

AWS Batch enables developers, scientists, and engineers to easily and efficiently run hundreds of thousands of batch computing jobs on AWS. Batch dynamically provisions the optimal quantity and type of compute resources (e.g., CPU or memory optimized instances) based on the volume and specific resource requirements of the batch jobs submitted. With Batch, there is no need to install and manage batch computing software or server clusters that you use to run your jobs, allowing you to focus on analyzing results and solving problems. Batch plans, schedules, and executes your batch computing workloads across the full range of AWS compute services and features, such as Amazon EC2 and Spot Instances.

In AWS Batch, you create a container with your workload and submit the container to a Job Queue. Batch manages compute resources for you, leveraging AWS Elastic Container Service (ECS).

In this lab, we will deploy a CloudFormation template which establishes a job queue, compute resources, job definitions and IAM roles. We will then submit and run a Batch Job.

## Deploy Batch Infrastructure

1. From your EC2 instance, download the CloudFormation code:

```
cd $HOME
mkdir lab
cd lab
git clone https://github.com/dotstar/batchlab.git
```

The batchlab/cfn directory contains our CloudFormation. While this is often deployed from the console, today let's use the CLI.

Two parameters in the template are unique to your account.

VPCid	This represents your VPC
SubnetId	For today's lab, select one of the public subnets from your VPC

You will find the VPC and Subnet information for your account in the [AWS console VPC](#) service.

Edit the file batch\_parameters.json, which looks like:

```
[
  { "ParameterKey": "EnvironmentName", "ParameterValue": "Batch" },
  { "ParameterKey": "VPCId", "ParameterValue": "vpc-0e77ef725616198b6" },
  { "ParameterKey": "SubnetId", "ParameterValue": "subnet-012f088c2337443ed" }
]
```

2. From the console, navigate to the VPC service.
  1. Determine your VPC-id. Edit batch\_parameters.json to reference your VPC.
  2. Determine a public subnet in your VPC and update the SubnetId parameter in batch\_parameters.json
3. Make sure you are in the cloud formation (cfn) directory, then create the stack.

#### Linux Environments

If you are running Linux, Windows Subsystem for Linux (WSL), or AppleOS you can use the Makefile to simplify typing.

```
cd $HOME/lab/batchlab/cfn
make create
```

#### Windows Environment

If you are running Windows (not WSL) and do not have access to the *make* utility, you can run this AWS CLI command to create the CloudFormation Stack. [Alternatively the stack can be run from the [AWS Console](#) ]

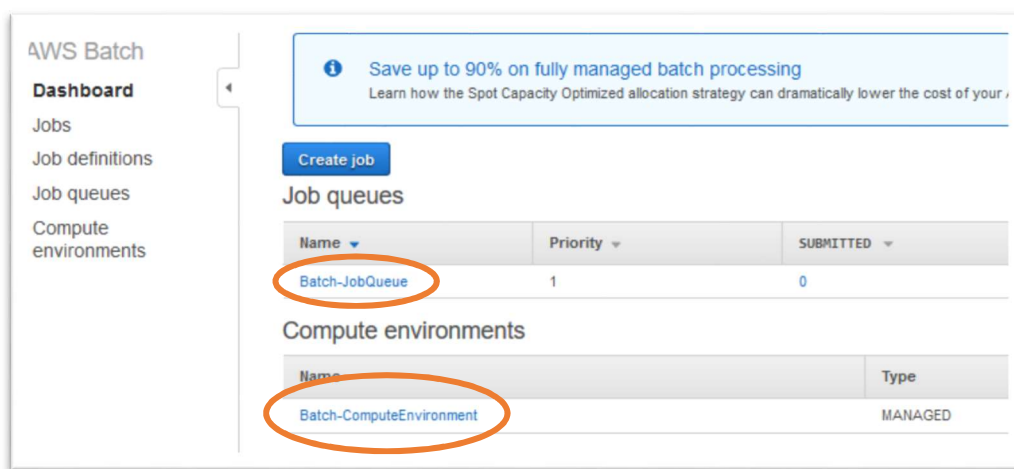
```
cd lab\batchlab\cfn

aws cloudformation create-stack --stack-name lab --
parameters file://batch_parameters.json --template-body
file://batch.yaml --capabilities CAPABILITY_NAMED_IAM --
region us-east-1
```

The stack creates several IAM roles, a Batch Queue, Compute Environment, and Description. Creating the stack requires approximately 5 minutes. You can watch the creation progress on the [CloudFormation console](#).

## Running your Batch Job

To save you some time, we've prepopulated the Batch job. Navigate to [AWS Batch](#) on the console. The console displays the queue and compute environment which we created in CloudFormation.



For this lab, we configured Batch to take advantage of AWS Spot instances. Amazon EC2 Spot Instances let you take advantage of unused EC2 capacity in the AWS cloud. This capacity is discounted at up to 90% as compared to on-demand pricing. When there is a demand surge, AWS may reclaim this capacity, giving you two minutes of notice. You have the option to hibernate, stop or terminate your Spot Instances when EC2 reclaims the capacity back with two-minutes of notice.

AWS Batch, which can be configured to resubmit failed jobs, is a very good use case for Spot instances.

In the lab, you will run a pre-created container, from Docker Hub. The container runs several matrix multiplies, using the Numpy framework. The source is in the git repo, it looks like this:

```

from numpy import random
from time import time
import os

# Initialize parameters from environment, or use defaults.
loops = int(os.environ.get('LOOPS',10))
matrix_size=int(os.environ.get('MATRIX_SIZE',2048))
seed=int(os.environ.get('SEED',2020))

if __name__ == "__main__":
    x = y = matrix_size
    random.seed(seed)

    starttime = time()
    for i in range(loops):
        m1 = random.rand(x,y)
        m2 = random.rand(x,y)
        m3 = m1 * m2
    stoptime = time()
    runtime = stoptime - starttime
    print(f'calculated {loops} loops of {matrix_size}x{matrix_size} multiplies in {runtime} seconds')

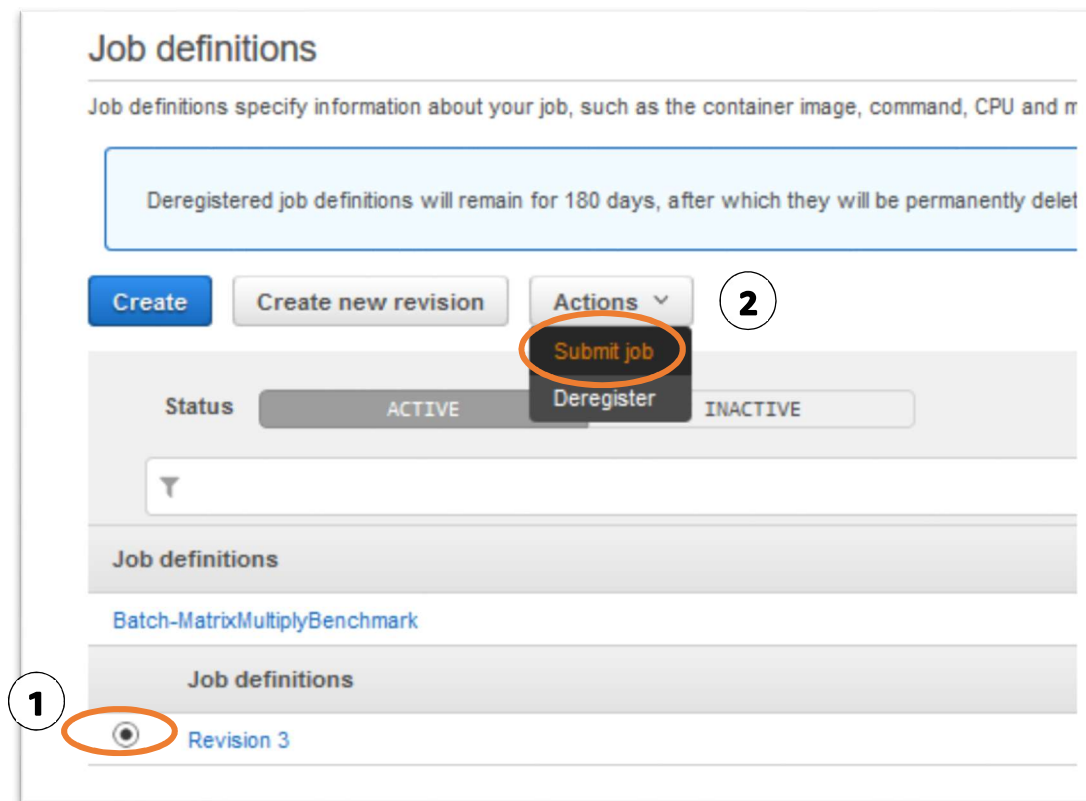
```

One way to change the inputs in a container is via environment variables. Notice that this code uses the variables LOOPS, MATRIX\_SIZE, and SEED as inputs.

### Submit a Batch job

1. Navigate to **Job Definitions** (on the left side of the console).
2. Click on the **Batch-MatrixMultiplyBenchmark** Job Definition
3. Select the most recent version, Then select **Submit Job** from the **Actions** button





4. Give your job a name
5. Select a Job queue
6. Change the environment variable LOOPS to 800
7. Click Submit Job

AWS Batch will now schedule your container to run, as resources become available in the AWS Batch Compute Environment which we built.

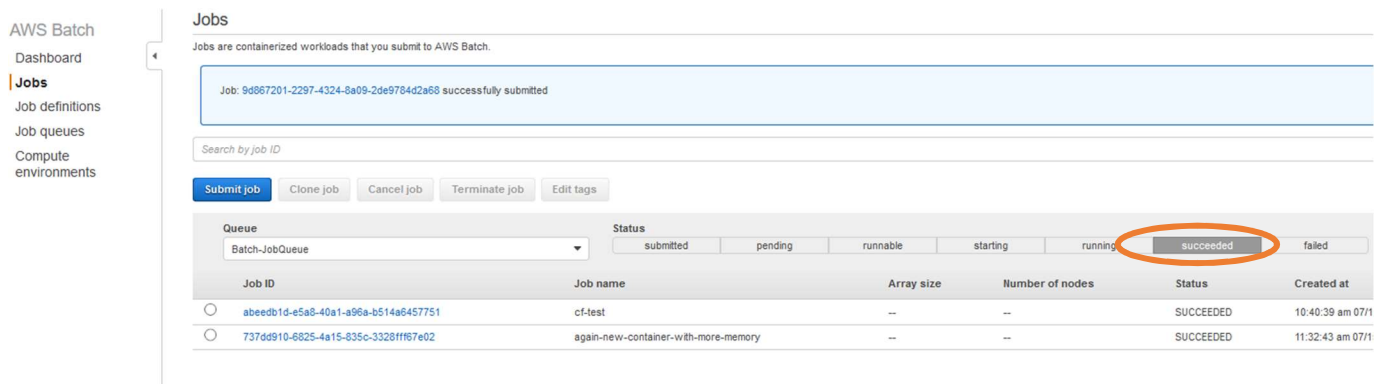
Now would be a good time to explore the Batch console.

- You should find the Job either in the queue, or running, or complete
- Explore the Job Definition. Notice that we're pulling a publically available container from [hub.docker.com](https://hub.docker.com/r/dotstar/matrixmath:v1) ( dotstar/matrixmath:v1 ). In production, you are likely to store containers in a private repo, such as AWS Elastic Container Registry (ECR).
- If you are familiar with AWS Elastic Container Service (ECS) take a peek at that console too. You'll see that Batch created a cluster to run your workload. If the job is still running, it will appear in ECS as an ECS task.

If you are familiar with [Amazon Elastic Container Service](https://aws.amazon.com/ecs/) (ECS), you might also want to look at

that [console](#). Behind the scenes, Batch creates an ECS cluster with your compute resources then submits your task queue jobs to that cluster.

- When the batch job completes, navigate to the Jobs window, where you can find additional information. There will be a Job ID, under the "Succeeded" tab. Click that to explore the log from this job.

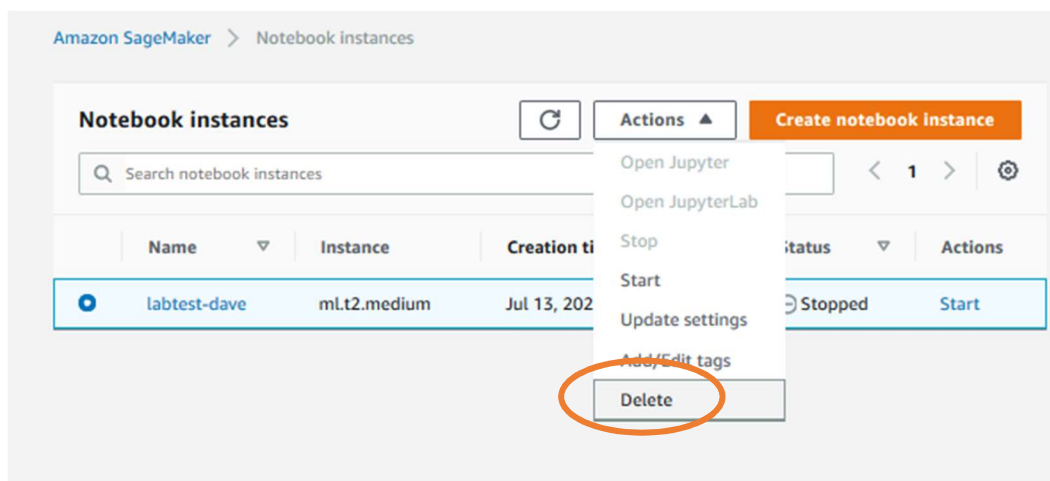


## Post Lab Cleanup

We have created running compute resources during this lab. To **ensure that you don't accumulate charges for unused resources, it is important to delete them.**

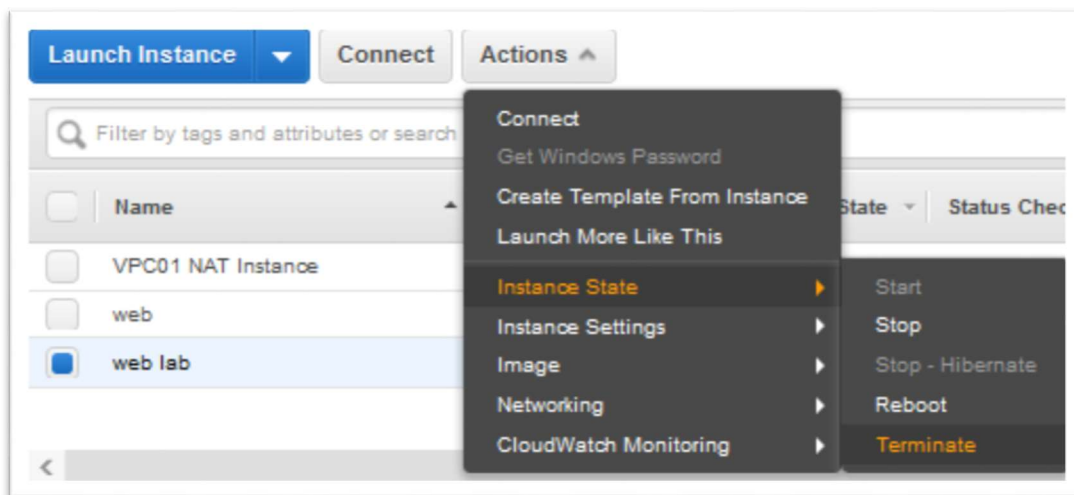
### Delete the notebook

From the SageMaker Console, **stop** and **delete** the notebook.



### Delete the EC2 instance

From the [EC2 Console](#), terminate the instance. Select the instance, then Actions -> Instance State -> Terminate



## Delete the AWS Batch infrastructure

```
cd $HOME/lab/batchlab/cfn  
make delete
```

This runs the CLI to delete the stack. One way to watch it delete, and verify that the resources are completely removed is via the console. Navigate to <https://console.aws.amazon.com/cloudformation> and make sure the stack deletes.

CloudFormation displays “DELETE\_IN\_PROGRESS” while the stack is being removed. It takes around 5 minutes to completely delete these resources.