CSP554—Big Data Technologies

Assignment #9

Worth: 5 points + 5 points extra credit

Exercise 1) 5 points
Read the article "Real-time stream processing for Big Data" available on the blackboard in the 'Articles' section and then answer the following questions:

a) (1.25 points) What is the Kappa architecture and how does it differ from the lambda architecture?
   - In streaming data processing, the Kappa architecture is used. The fundamental concept behind Kappa architecture is that you can conduct both real-time and batch processing with a single technology stack, which is particularly useful for analytics.
   - The Kappa architecture also supports historical analytics, which reads the stored streaming data from the messaging engine in a batch manner later to produce additional analyzable outputs for more types of analysis. The Kappa architecture also allows for real-time analytics since the data is read and converted immediately after being loaded into the messaging system.

b) (1.25 points) What are the advantages and drawbacks of pure streaming versus micro-batch real-time processing systems?
   - Micro-batch processing is a method for gathering and operating on data in small groups. In comparison, conventional batch processing often necessitates working with a broad data set. Micro batch processing is a variation of conventional batch processing in which data is processed more often to deal with smaller groups of new data.
   - The world has changed, and there are many applications where micro-batches are insufficient. Organizations commonly use micro-batch processing to make design decisions that resist stream processing. For example, an Apache Spark shop could be using Spark streaming, which is essentially a micro-batch processing extension of the Spark API with the added benefit of using in-memory computing resources. Stream processing allows systems to respond to new data events as they occur. Rather than grouping and gathering data at preset times, stream processing systems capture and process data as it is produced.
   - In some cases, micro-batch processing has sufficed, but companies are rapidly learning that stream processing based on in-memory technology –

whether in the cloud or on-premises – is the better choice. Current applications are rapidly using stream processing technology. As data has exploded over the past decade, businesses have switched to real-time analysis to respond to data closer to the time it was generated to solve for a range of use cases and applications.

c) (1.25 points) In few sentences describe the data processing pipeline in Storm.
   - The Apache Storm is looking for a Zookeeper cluster. At least two distinct components make up a Storm cluster: Storm Nimbus and Storm Supervisor. Storm Nimbus is a close master node to Hadoop Job Tracker. It is in charge of distributing technology across the cluster, assigning tasks to subordinates, and ensuring that there are no mistakes. Storm Supervisor is a node of the workplace. It executes the code provided to it by Storm Nimbus. Storm clusters are stateless and fail-safe. Since Nimbus distributes the code to supervisors if there is a Zookeeper and at least one supervisor node, a Storm cluster will operate without a Nimbus node.
   - The Storm also provides Storm DRPC, Storm Log reader, and Storm UI. Hadoop and other similar systems process data using the idea of a job, which is a batch process that will ultimately produce a result and end. Topologies, which are execution graphs in which each node includes computing logic and each edge defines how data is distributed between nodes, are used instead by Hurricane.
   - Spouts and bolts are the two streams that make up a storm topology. The source stream is represented by a spout. A bolt is a stream data processing agent with the ability to generate new sources. For eg, a spout could connect to the Twitter API and output a tweet stream, which a bolt could then consume and output as a stream of trending topics.

d) (1.25 points) How does Spark streaming shift the Spark batch processing approach to work on real-time data streams?
   - Modifications to the streaming in order to meet real-time requirements, Spark divides the stream of incoming data objects into small batches, converts them to RDDs, and processes them normally. It also takes care of data flow and distribution on its own.

Exercise 2) 5 points extra credit

Follow the document “Instructions for setting up a VM with Kafka” included with this assignment and execute the demo code. Provide enough screen shots to indicate you have completed the document through section 4. Then remember to terminate your VM.

Instructions for setting up and exploring the operation of a single VM version of Kafka

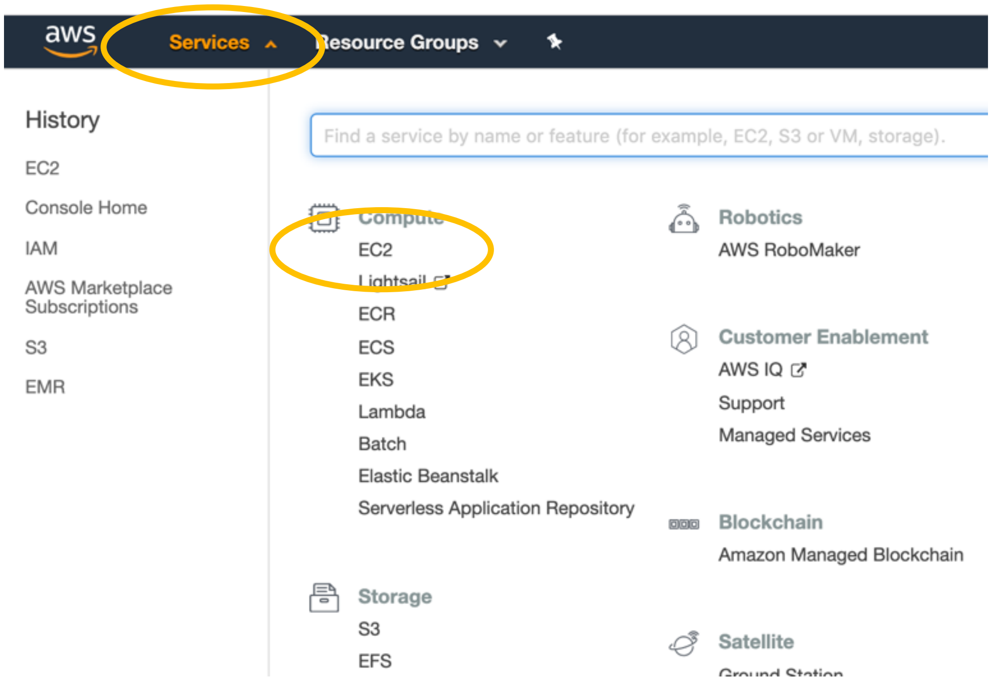
The AWS cloud offers a managed version of Kafka but it is complex to configure and potentially expensive to run. Instead we will use a well-supported open source version of Kafka and install and manage it ourselves on a single virtual machine.

Note, for those of you who might want to include Kafka in a big data processing pipeline, in the future, you might try to install it onto an EMR/Hadoop master node, following the instructions below, but for the following assignment, we will use a separate virtual machine.

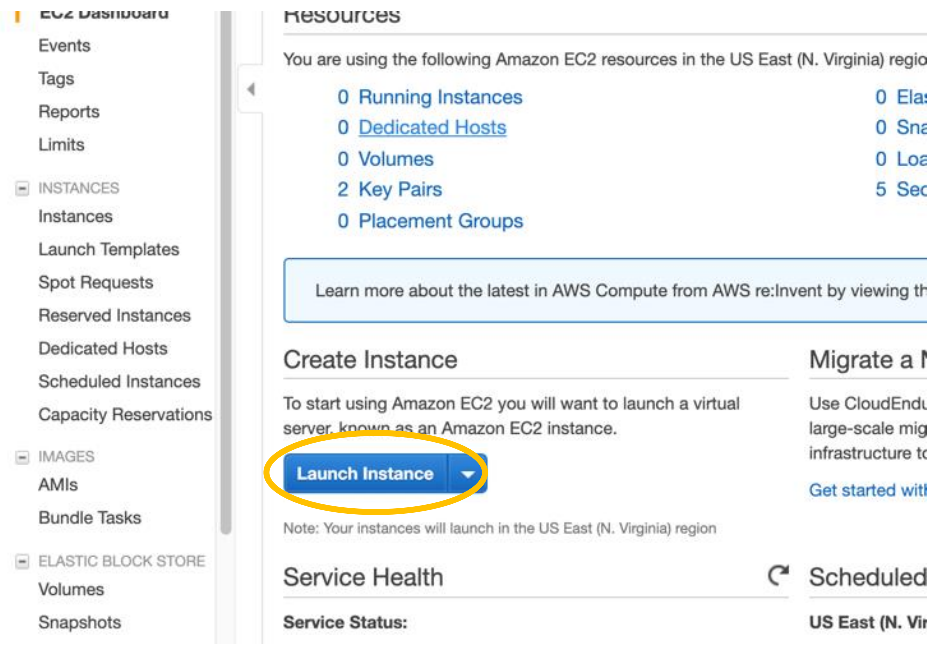
These instructions are divided into five sections. In the first we will set up an AWS EC2 instance (virtual machine). In the second section we will configure the virtual machine with the software and properties necessary to execute Kafka. In the third section we will learn how to start up Kafka and then, in the fourth section explore some of its capabilities. Finally, in the fifth section we describe how to stop, start or terminate your EC2 instance.

Section 1: Instructions for setting up an EC2 instance (virtual machine)

1) Select "Services" on the AWS console. Then select "EC2"



2) Launch a new EC2 instance by clicking on "Launch Instance"



EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Launch Templates
Spot Requests
Reserved Instances
Dedicated Hosts
Scheduled Instances
Capacity Reservations
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots

Resources

You are using the following Amazon EC2 resources in the US East (N. Virginia) regio

0 Running Instances	0 Ela
0 Dedicated Hosts	0 Sna
0 Volumes	0 Loa
2 Key Pairs	5 Sec
0 Placement Groups	

Learn more about the latest in AWS Compute from AWS re:Invent by viewing th

Create Instance

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

Launch Instance

Note: Your instances will launch in the US East (N. Virginia) region

Migrate a M

Use CloudEndu large-scale mig infrastructure to

Get started witl

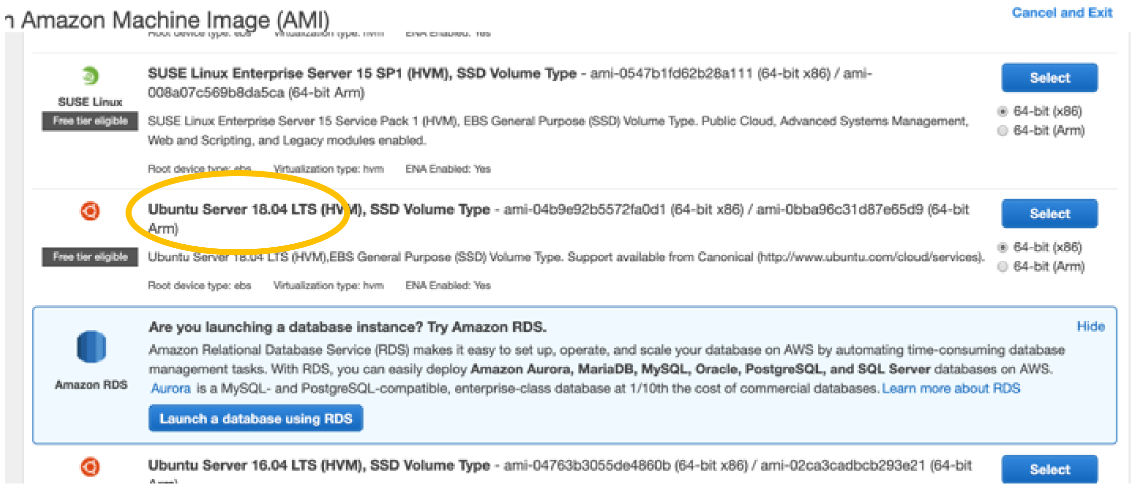
Service Health

Service Status:

Scheduled

US East (N. Vir

3) Select the Ubuntu 18.04 Amazon Machine Image (AMI)



n Amazon Machine Image (AMI)

Cancel and Exit

SUSE Linux
Free tier eligible

SUSE Linux Enterprise Server 15 SP1 (HVM), SSD Volume Type - ami-0547b1fd62b28a111 (64-bit x86) / ami-008a07c569b8da5ca (64-bit Arm)

SUSE Linux Enterprise Server 15 Service Pack 1 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select
● 64-bit (x86)
○ 64-bit (Arm)

Free tier eligible

Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-04b9e92b5572fa0d1 (64-bit x86) / ami-0bba96c31d87e65d9 (64-bit Arm)

Ubuntu Server 18.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select
● 64-bit (x86)
○ 64-bit (Arm)

Amazon RDS

Are you launching a database instance? Try Amazon RDS.

Hide

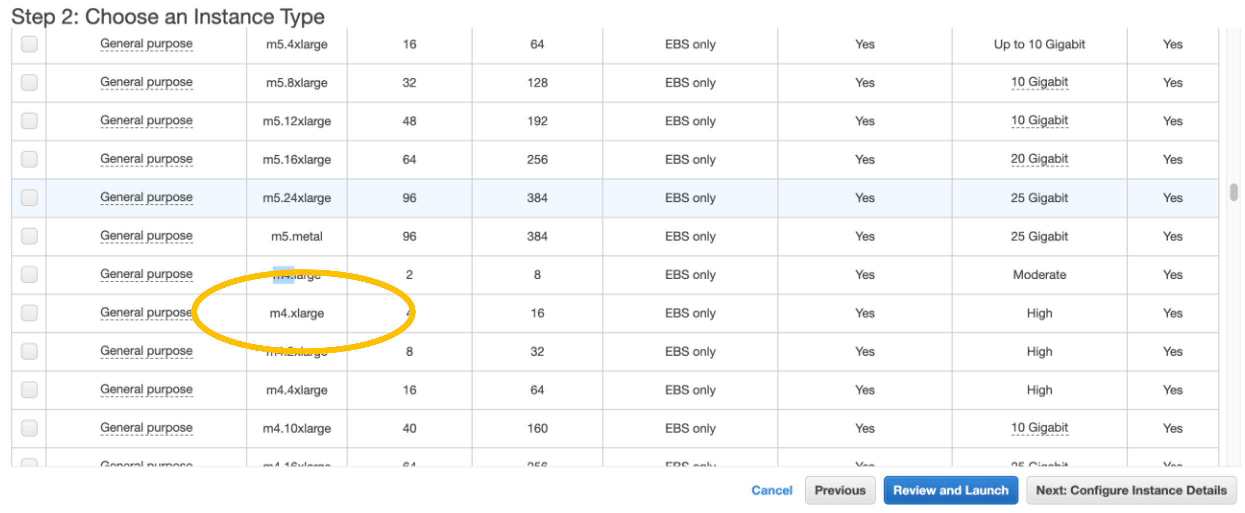
Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale your database on AWS by automating time-consuming database management tasks. With RDS, you can easily deploy **Amazon Aurora, MariaDB, MySQL, Oracle, PostgreSQL, and SQL Server** databases on AWS. Aurora is a MySQL- and PostgreSQL-compatible, enterprise-class database at 1/10th the cost of commercial databases. Learn more about RDS

Launch a database using RDS

Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-04763b3055de4860b (64-bit x86) / ami-02ca3cadbcb293e21 (64-bit

Select

Select an instance (VM) type of m4.xlarge (or if for some reason this is not available to you, try m4.large):

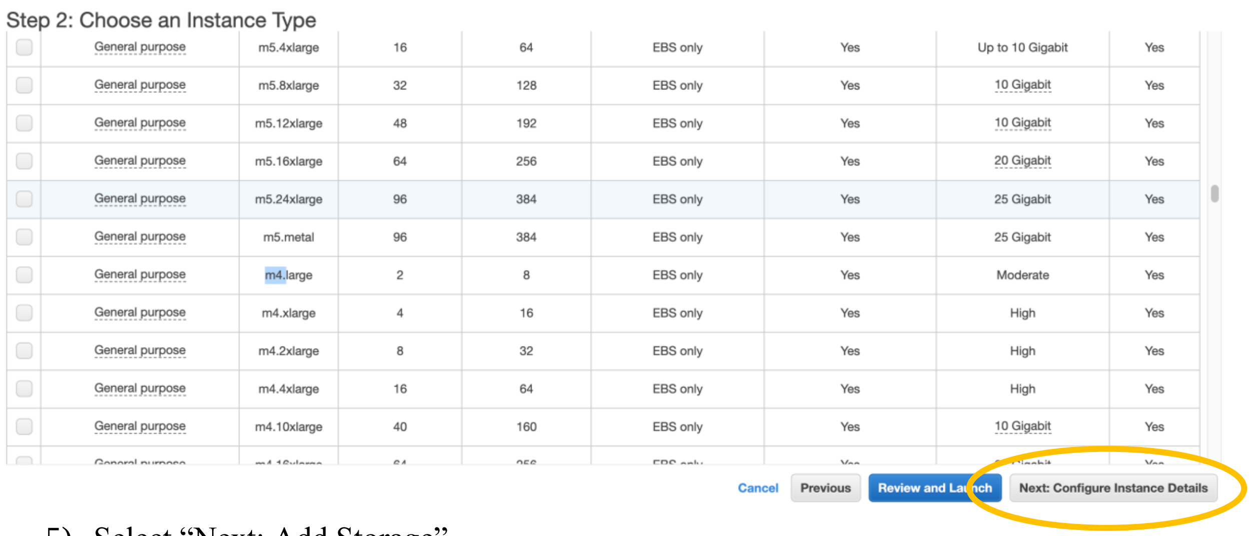


Step 2: Choose an Instance Type

☐	General purpose	m5.4xlarge	16	64	EBS only	Yes	Up to 10 Gigabit	Yes
☐	General purpose	m5.8xlarge	32	128	EBS only	Yes	10 Gigabit	Yes
☐	General purpose	m5.12xlarge	48	192	EBS only	Yes	10 Gigabit	Yes
☐	General purpose	m5.16xlarge	64	256	EBS only	Yes	20 Gigabit	Yes
☐	General purpose	m5.24xlarge	96	384	EBS only	Yes	25 Gigabit	Yes
☐	General purpose	m5.metal	96	384	EBS only	Yes	25 Gigabit	Yes
☐	General purpose	m4.large	2	8	EBS only	Yes	Moderate	Yes
☐	General purpose	m4.xlarge	4	16	EBS only	Yes	High	Yes
☐	General purpose	m4.2xlarge	8	32	EBS only	Yes	High	Yes
☐	General purpose	m4.4xlarge	16	64	EBS only	Yes	High	Yes
☐	General purpose	m4.10xlarge	40	160	EBS only	Yes	10 Gigabit	Yes

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

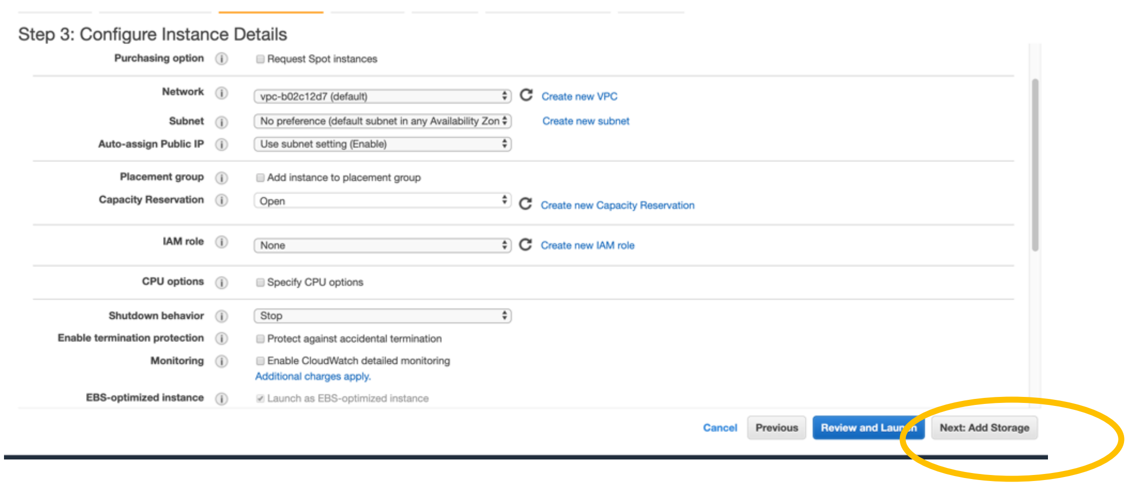
4) Select “Next: Configure Instance Details”



Step 2: Choose an Instance Type

☐	General purpose	m5.4xlarge	16	64	EBS only	Yes	Up to 10 Gigabit	Yes
☐	General purpose	m5.8xlarge	32	128	EBS only	Yes	10 Gigabit	Yes
☐	General purpose	m5.12xlarge	48	192	EBS only	Yes	10 Gigabit	Yes
☐	General purpose	m5.16xlarge	64	256	EBS only	Yes	20 Gigabit	Yes
☐	General purpose	m5.24xlarge	96	384	EBS only	Yes	25 Gigabit	Yes
☐	General purpose	m5.metal	96	384	EBS only	Yes	25 Gigabit	Yes
☐	General purpose	m4.large	2	8	EBS only	Yes	Moderate	Yes
☐	General purpose	m4.xlarge	4	16	EBS only	Yes	High	Yes
☐	General purpose	m4.2xlarge	8	32	EBS only	Yes	High	Yes
☐	General purpose	m4.4xlarge	16	64	EBS only	Yes	High	Yes
☐	General purpose	m4.10xlarge	40	160	EBS only	Yes	10 Gigabit	Yes
☐	General purpose	m4.16xlarge	64	256	EBS only	Yes	[illegible]	Yes

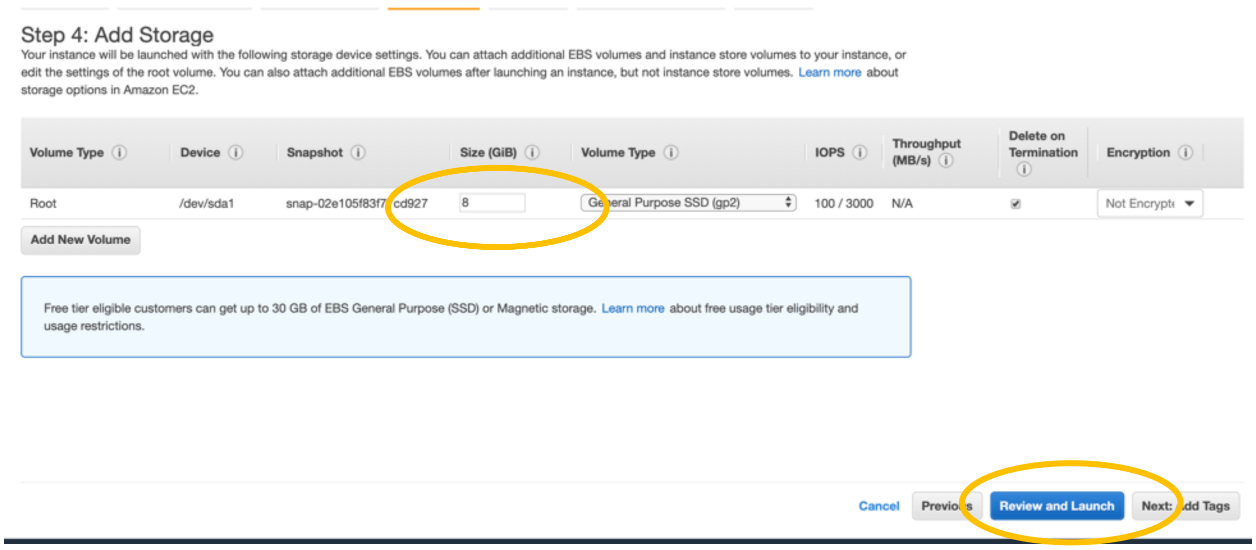
5) Select "Next: Add Storage"



Step 3: Configure Instance Details

Purchasing option	☐ Request Spot instances
Network	vpc-b02c12d7 (default) — Create new VPC
Subnet	No preference (default subnet in any Availability Zon — Create new subnet
Auto-assign Public IP	Use subnet setting (Enable)
Placement group	☐ Add instance to placement group
Capacity Reservation	Open — Create new Capacity Reservation
IAM role	None — Create new IAM role
CPU options	☐ Specify CPU options
Shutdown behavior	Stop
Enable termination protection	☐ Protect against accidental termination
Monitoring	☐ Enable CloudWatch detailed monitoring Additional charges apply.
EBS-optimized instance	✓ Launch as EBS-optimized instance

6) Change the root storage size from 8 GiB to 30 GiB (if possible, otherwise as much as you can between 8 and 30 GiB). Then select "Review and Launch"



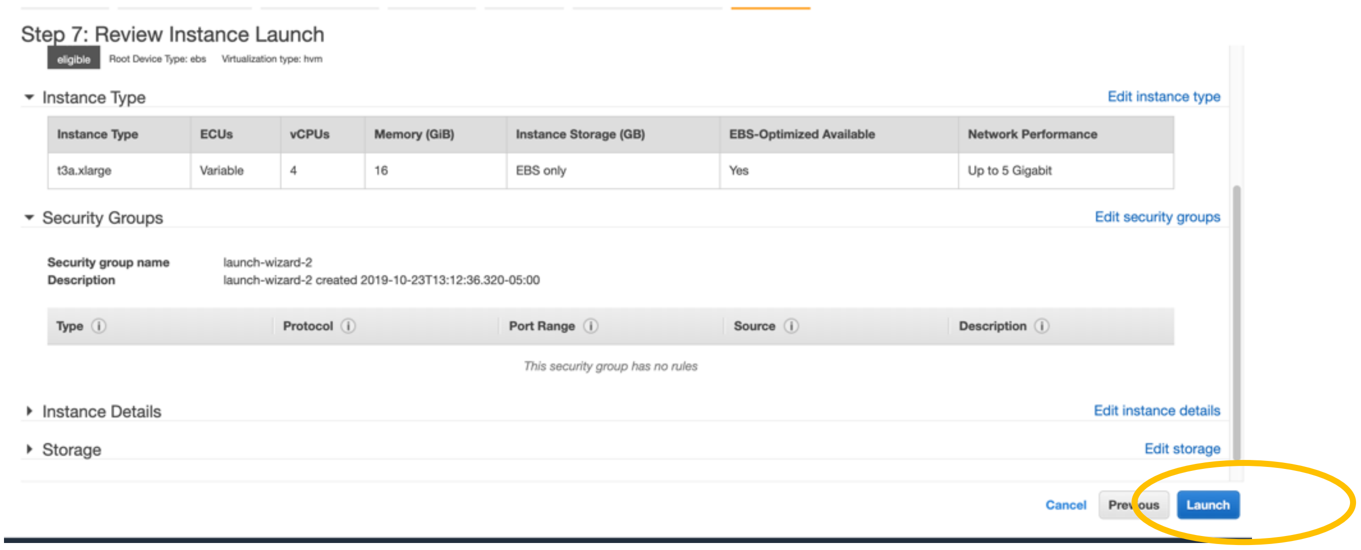
Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-02e105f83f7cd927	8	General Purpose SSD (gp2)	100 / 3000	N/A	✓	Not Encrypte

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.

7 Now "Launch" the instance



Step 7: Review Instance Launch

eligible Root Device Type: ebs Virtualization type: hvm

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t3a.xlarge	Variable	4	16	EBS only	Yes	Up to 5 Gigabit

Security Groups

Security group name	launch-wizard-2
Description	launch-wizard-2 created 2019-10-23T13:12:36.320-05:00

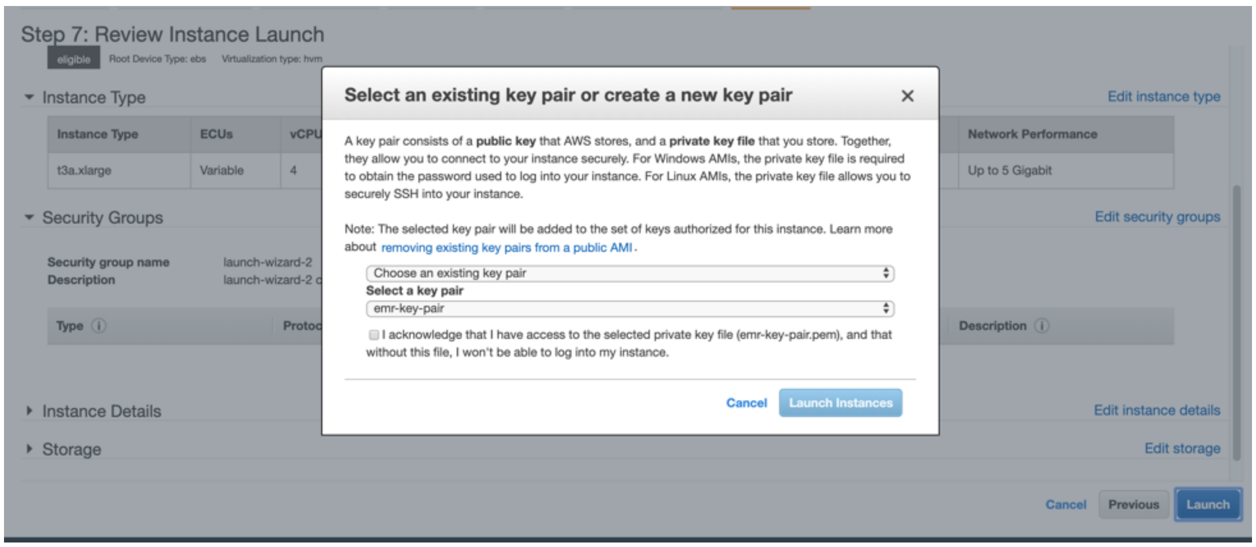
Type	Protocol	Port Range	Source	Description

This security group has no rules

Instance Details

Storage

8) Select an existing key pair (or create a new pair). You can use the same key pair you created for your EMR instances. Make sure to check the “I acknowledge…” checkbox.



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

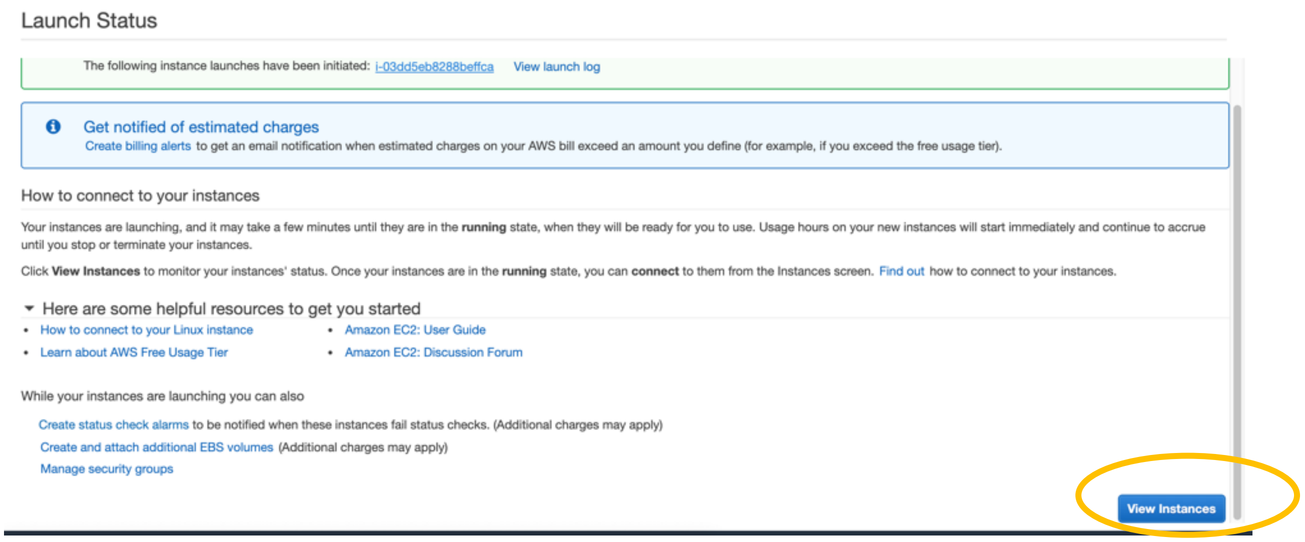
Choose an existing key pair

Select a key pair

emr-key-pair

I acknowledge that I have access to the selected private key file (emr-key-pair.pem), and that without this file, I won't be able to log into my instance.

9) You should see this. Select “View Instances”



Launch Status

The following instance launches have been initiated: i-03dd5eb8288beffca View launch log

Get notified of estimated charges

Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.

Click **View Instances** to monitor your instances' status. Once your instances are in the **running** state, you can **connect** to them from the Instances screen. Find out how to connect to your instances.

Here are some helpful resources to get you started

- How to connect to your Linux instance
- Learn about AWS Free Usage Tier
- Amazon EC2: User Guide
- Amazon EC2: Discussion Forum

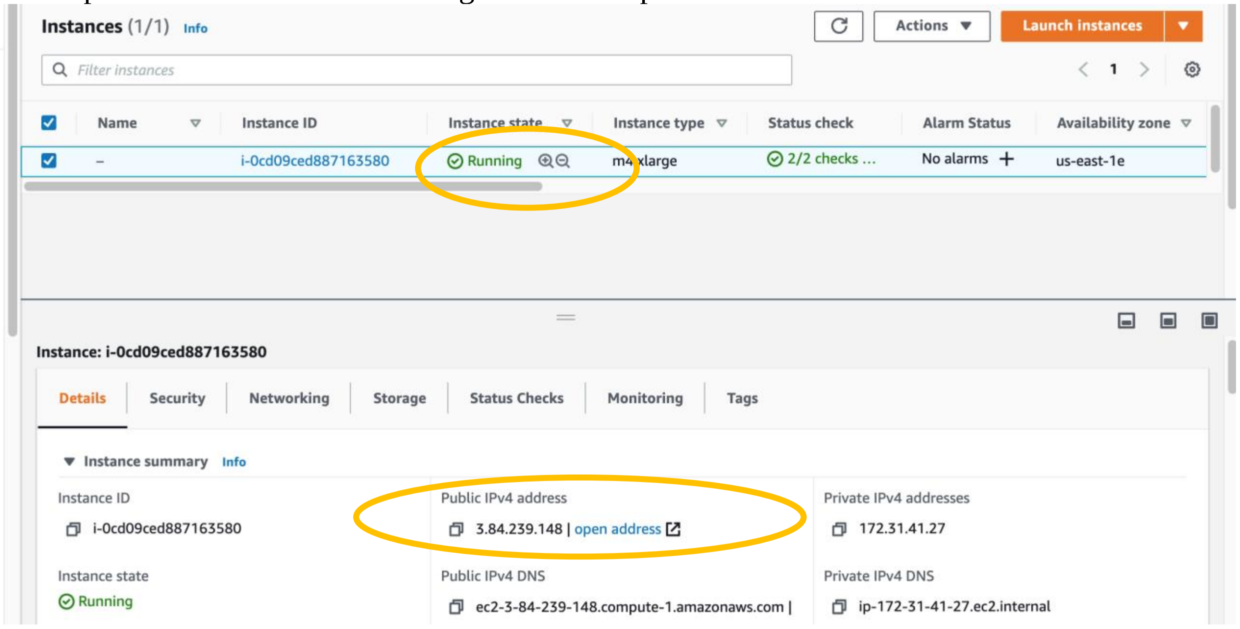
While your instances are launching you can also

Create status check alarms to be notified when these instances fail status checks. (Additional charges may apply)

Create and attach additional EBS volumes (Additional charges may apply)

Manage security groups

10) Wait until the instance state is "running". Note that the public address (Public DNS) is provided for use in connecting via ssh or scp to the instance.



Instances (1/1) Info

	Name	Instance ID	Instance state	Instance type	Status check	Alarm Status	Availability zone
✓	–	i-0cd09ced887163580	Running	m4.xlarge	2/2 checks ...	No alarms	us-east-1e

Instance: i-0cd09ced887163580

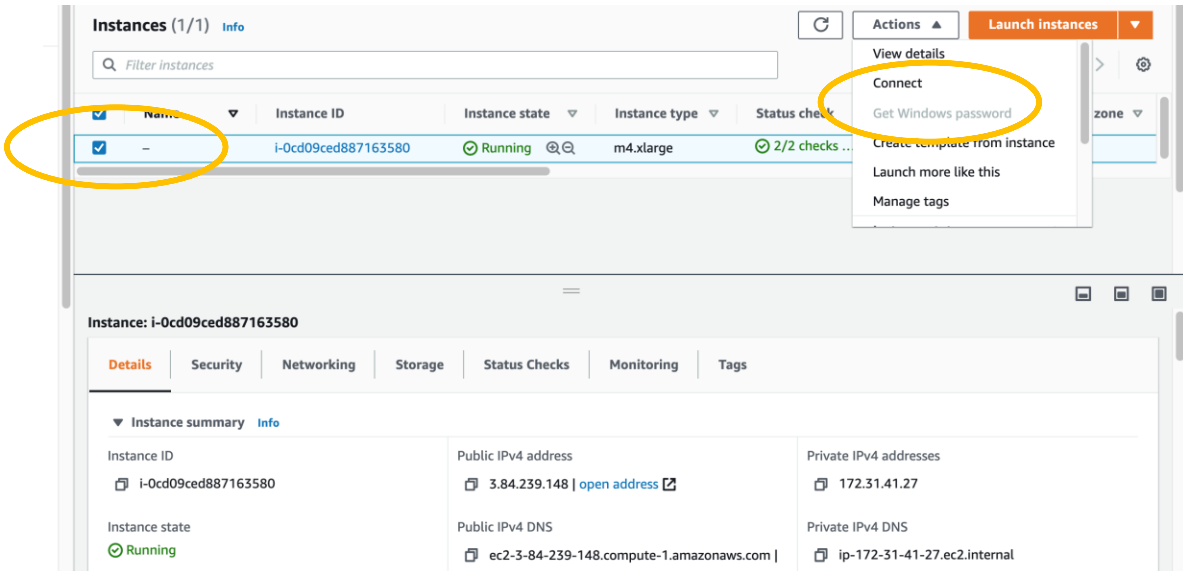
Details | Security | Networking | Storage | Status Checks | Monitoring | Tags

Instance summary Info

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0cd09ced887163580	3.84.239.148 | open address	172.31.41.27
Instance state	Public IPv4 DNS	Private IPv4 DNS
Running	ec2-3-84-239-148.compute-1.amazonaws.com |	ip-172-31-41-27.ec2.internal

10 Now you can ssh to the instance the easy way as follows.

a. Make sure you check the EC2 instance
b. Click on the "Connect" button.



Instances (1/1) Info

	Name	Instance ID	Instance state	Instance type	Status check
✓	–	i-0cd09ced887163580	Running	m4.xlarge	2/2 checks ...

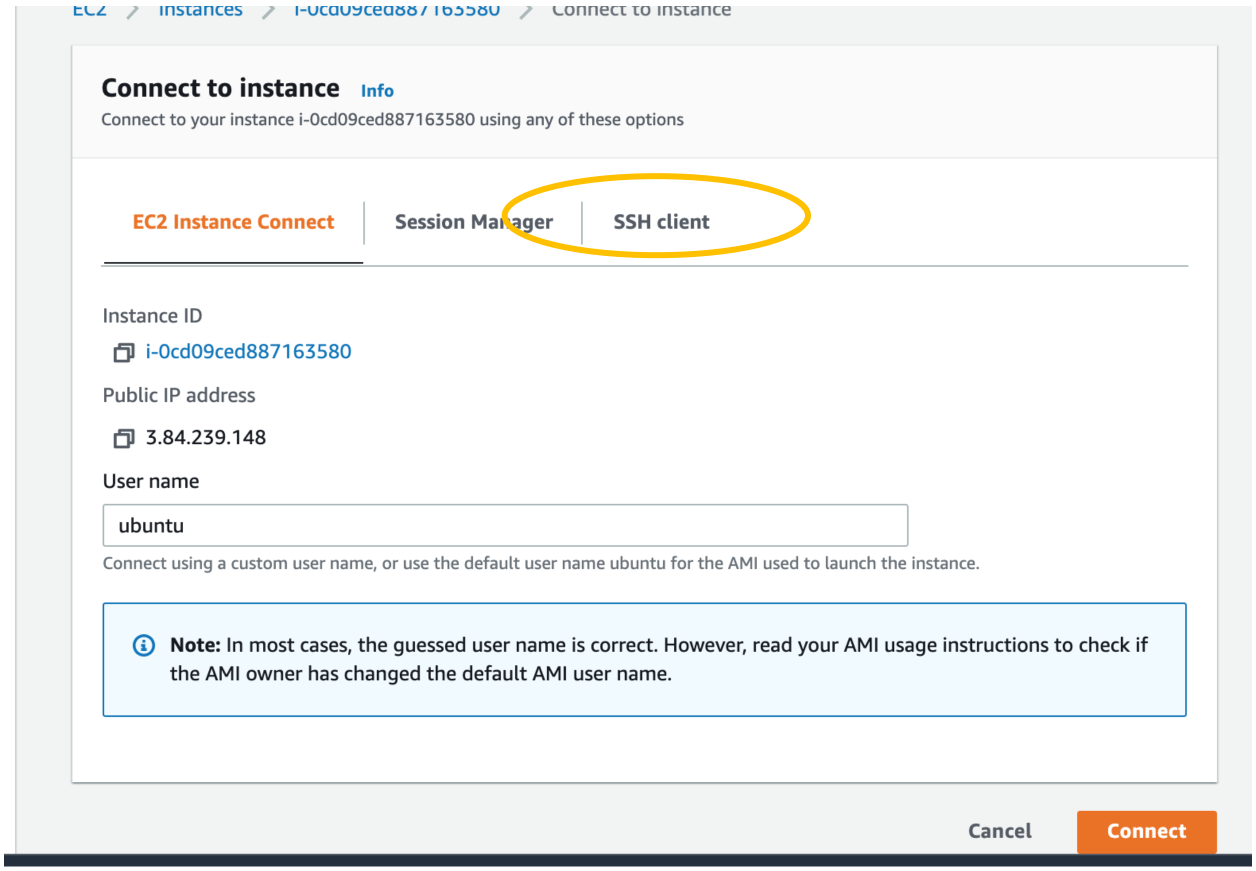
Instance: i-0cd09ced887163580

Details | Security | Networking | Storage | Status Checks | Monitoring | Tags

Instance summary Info

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0cd09ced887163580	3.84.239.148 | open address	172.31.41.27
Instance state	Public IPv4 DNS	Private IPv4 DNS
Running	ec2-3-84-239-148.compute-1.amazonaws.com |	ip-172-31-41-27.ec2.internal

c) Select the "SSH Client" tab


EC2 > Instances > i-0cd09ced887163580 > Connect to instance

Connect to instance Info

Connect to your instance i-0cd09ced887163580 using any of these options

Instance ID

i-0cd09ced887163580

Public IP address

3.84.239.148

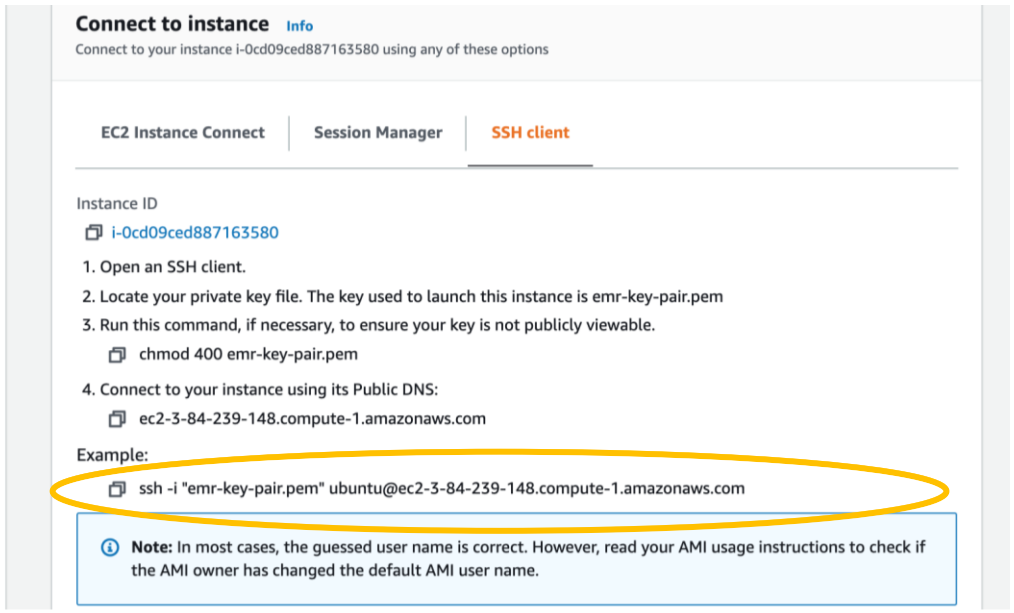
User name

ubuntu

Connect using a custom user name, or use the default user name ubuntu for the AMI used to launch the instance.

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

d) Cut and paste the example ssh command that appears into your terminal to connect to your instance


Connect to instance Info

Connect to your instance i-0cd09ced887163580 using any of these options

Instance ID

i-0cd09ced887163580

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is emr-key-pair.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.

```
chmod 400 emr-key-pair.pem
```

4. Connect to your instance using its Public DNS:

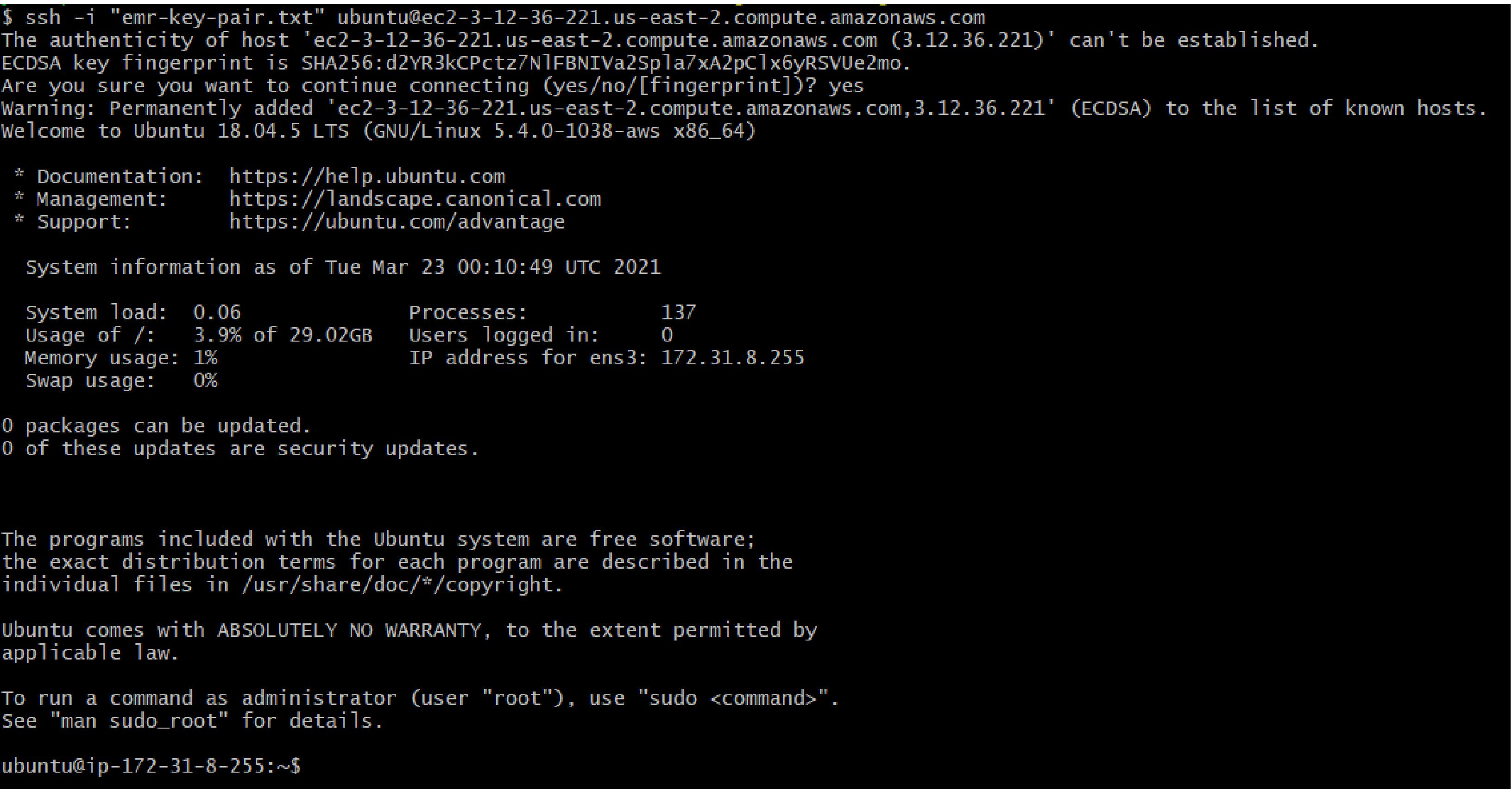
```
ec2-3-84-239-148.compute-1.amazonaws.com
```

Example:

```
ssh -i "emr-key-pair.pem" ubuntu@ec2-3-84-239-148.compute-1.amazonaws.com
```

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Note that in this case the user name of your account is “ubuntu” and the address of your instance <Public DNS (IPv4)> is also used. So the ssh command about is built as follows. Note <Public DNS (IPv4)> is replaced with the actual value listed in the previsouly shown instance information pane.



```
$ ssh -i "emr-key-pair.txt" ubuntu@ec2-3-12-36-221.us-east-2.compute.amazonaws.com
The authenticity of host 'ec2-3-12-36-221.us-east-2.compute.amazonaws.com (3.12.36.221)' can't be established.
ECDSA key fingerprint is SHA256:d2YR3kCPctz7NlFBNIVa2Spla7xA2pClx6yRSVUe2mo.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-12-36-221.us-east-2.compute.amazonaws.com,3.12.36.221' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1038-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Mar 23 00:10:49 UTC 2021

  System load:  0.06              Processes:           137
  Usage of /:   3.9% of 29.02GB   Users logged in:     0
  Memory usage: 1%                IP address for ens3: 172.31.8.255
  Swap usage:   0%

0 packages can be updated.
0 of these updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-8-255:~$
```

11) About terminal windows…

You will need to open up several terminal windows to set up and interact with Kafka. We will refer to these terminal windows by a unique name, which will, we hope, keep things clearer and more organized.

Terminal Window "Name"	Connected to…	Purpose
KT-1	Local computer	Used to scp files from your local computer to the AWS virtual machine
KT-2	AWS virtual machine using ssh	Used to execute the basic commands on the AWS virtual machine necessary to install, configure and start Kafka. Also, used to execute Kafka data producers.
KT-3	AWS virtual machine using ssh	Used to execute Kafka data producers.
KT-4	AWS virtual machine using ssh	Used to execute Kafka data consumers and output the data they consume.

12) Now you can ssh to the EC2 instance as follows using the ssh command you just copied above:

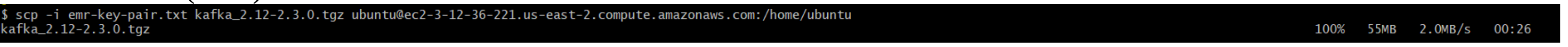
ssh -i emr-key-pair.pem ubuntu@<Public DNS (IPv4)>

Section 2: Installing basic software onto your EC2 instance

1) Download the Kafka software (kafka_2.12-2.3.0.tgz) from the assignment to your PC or MAC

2) In a terminal window (KT-1) "scp" the confluent Kafka software to your EC2 instance:

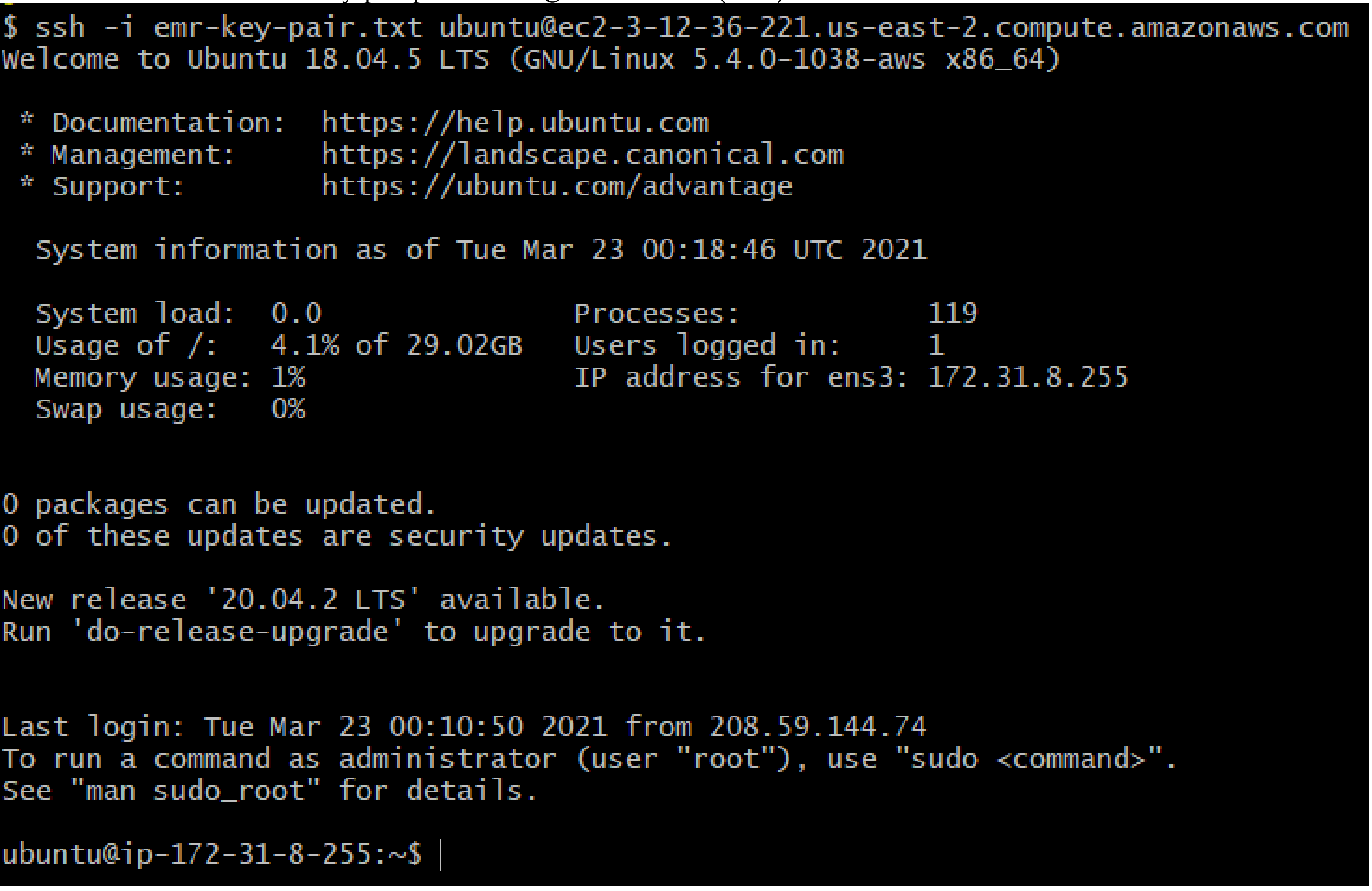
scp -i emr-key-pair.pem /path/to/ kafka_2.12-2.3.0.tgz ubuntu@<Public DNS (IPv4)>:/home/ubuntu



```
$ scp -i emr-key-pair.txt kafka_2.12-2.3.0.tgz ubuntu@ec2-3-12-36-221.us-east-2.compute.amazonaws.com:/home/ubuntu
kafka_2.12-2.3.0.tgz                                                                        100%   55MB   2.0MB/s   00:26
```

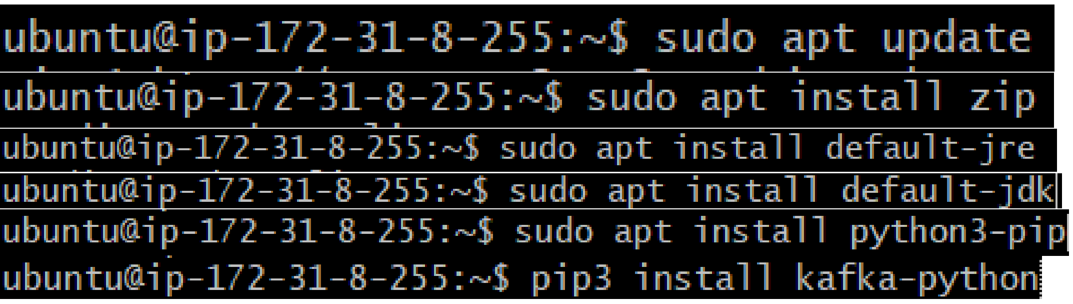
3) If you haven't already done so, open a terminal window and "ssh" to your EC2 instance. We will call this window, now connected to the EC2 instance, the KT-2 terminal window

ssh -i emr-key-pair.pem ubuntu@<Public DNS (IPv4)>



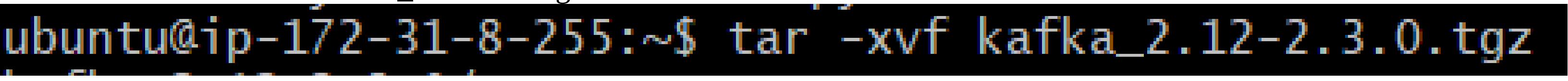
```
$ ssh -i emr-key-pair.txt ubuntu@ec2-3-12-36-221.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1038-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Mar 23 00:18:46 UTC 2021

  System load:  0.0               Processes:           119
  Usage of /:   4.1% of 29.02GB   Users logged in:     1
  Memory usage: 1%                IP address for ens3: 172.31.8.255
  Swap usage:   0%


0 packages can be updated.
0 of these updates are security updates.

New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Tue Mar 23 00:10:50 2021 from 208.59.144.74
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-8-255:~$ |
```

4) Now into the EC2 console accessible through your "KT-2" terminal window enter the following commands one at a time. As each executes, they may ask you to respond with "Yes" or "Y". Do so each time. On your instance console enter the following:
   I. sudo apt update
   II. sudo apt install zip
   III. sudo apt install default-jre
   IV. sudo apt install default-jdk
   V. sudo apt install python3-pip
   VI. pip3 install kafka-python


```
ubuntu@ip-172-31-8-255:~$ sudo apt update
ubuntu@ip-172-31-8-255:~$ sudo apt install zip
ubuntu@ip-172-31-8-255:~$ sudo apt install default-jre
ubuntu@ip-172-31-8-255:~$ sudo apt install default-jdk
ubuntu@ip-172-31-8-255:~$ sudo apt install python3-pip
ubuntu@ip-172-31-8-255:~$ pip3 install kafka-python
```

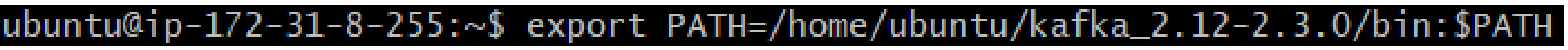
5) Then through your "KT-2" terminal window enter the following commands

tar -xvf kafka_2.12-2.3.0.tgz


```
ubuntu@ip-172-31-8-255:~$ tar -xvf kafka_2.12-2.3.0.tgz
```

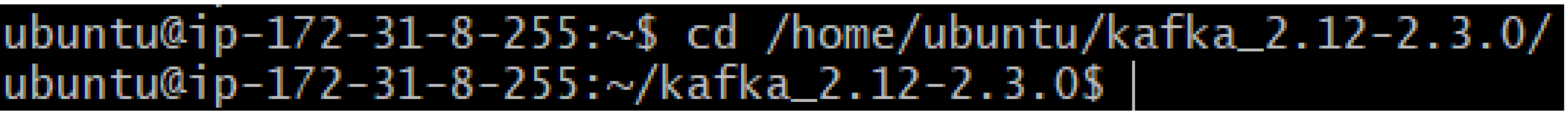
6) Add the install location of the Kafka "bin" directory to your PATH in the KT-2 terminal window as follows:

   export PATH=/home/ubuntu/kafka_2.12-2.3.0/bin:$PATH


```
ubuntu@ip-172-31-8-255:~$ export PATH=/home/ubuntu/kafka_2.12-2.3.0/bin:$PATH
```

   Then change to the /home/ubuntu/kafka_2.12-2.3.0 directory:

   cd /home/ubuntu/kafka_2.12-2.3.0

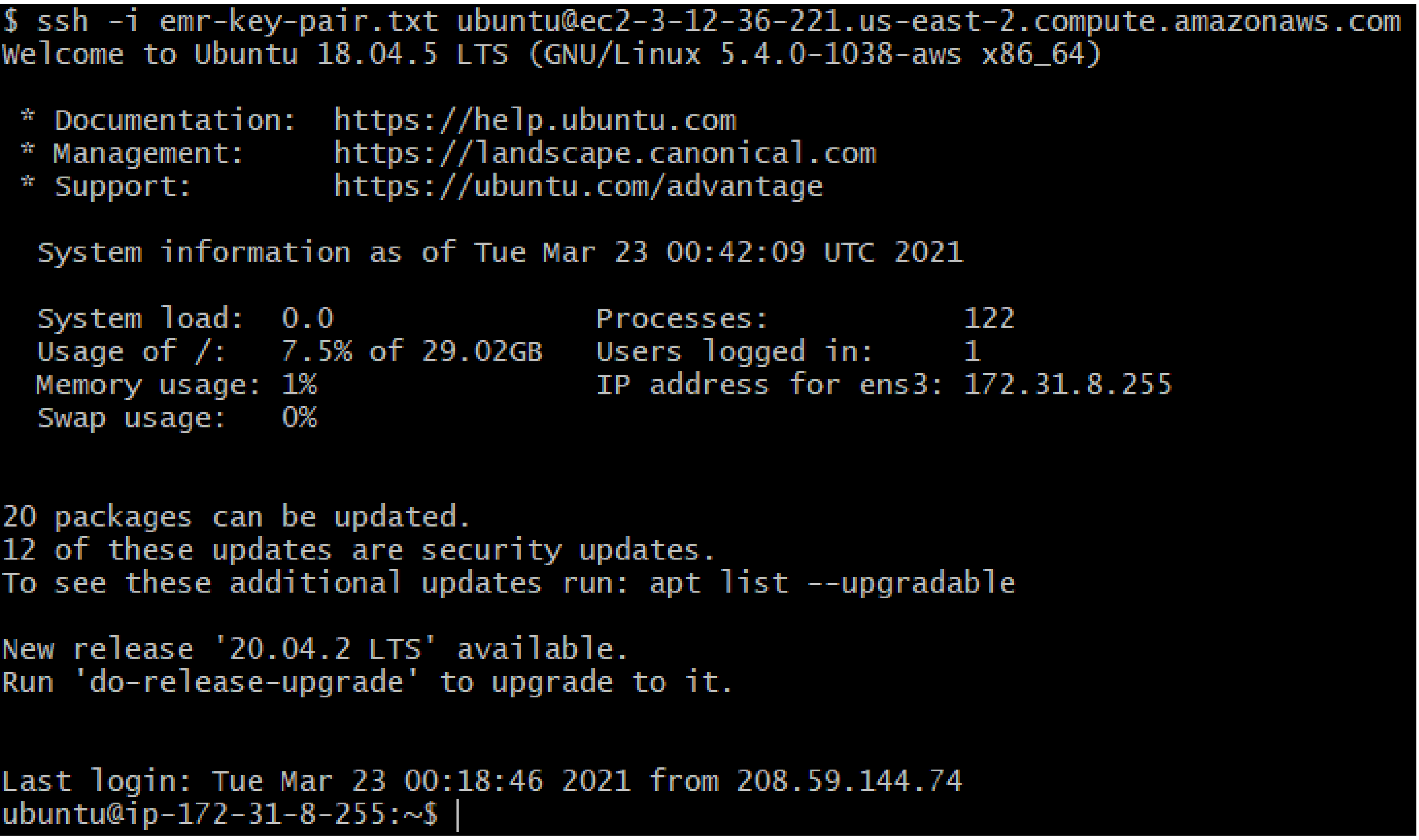

```
ubuntu@ip-172-31-8-255:~$ cd /home/ubuntu/kafka_2.12-2.3.0/
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$
```

Section 3: Starting Kafka

We need to have three windows connected to the EC2 instance for this section. If you have one already recall we refer to it as the "KT-2" terminal but we will need to open and connect two more windows to the EC2 instance.

1) Open a third terminal window and enter the following. We will refer to this as the "KT-3" terminal window.

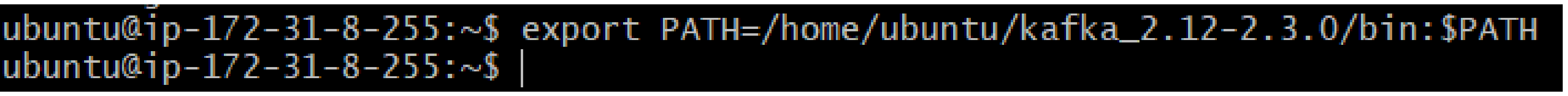
   ssh -i emr-key-pair.pem ubuntu@<Public DNS (IPv4)>



```
$ ssh -i emr-key-pair.txt ubuntu@ec2-3-12-36-221.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1038-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Mar 23 00:42:09 UTC 2021

  System load:  0.0               Processes:           122
  Usage of /:   7.5% of 29.02GB   Users logged in:     1
  Memory usage: 1%                IP address for ens3: 172.31.8.255
  Swap usage:   0%


20 packages can be updated.
12 of these updates are security updates.
To see these additional updates run: apt list --upgradable

New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Tue Mar 23 00:18:46 2021 from 208.59.144.74
ubuntu@ip-172-31-8-255:~$ |
```

Add the install location of the Kafka “bin” directory to your PATH in the KT-3 terminal window as follows:

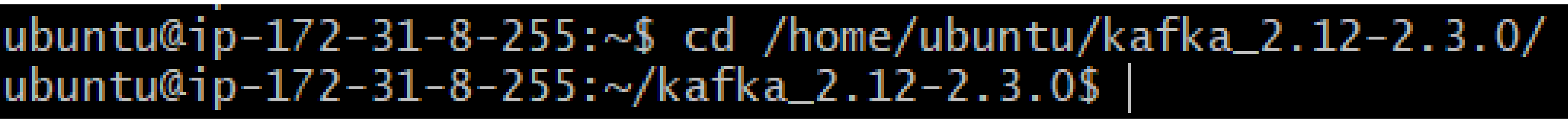
export PATH=/home/ubuntu/kafka_2.12-2.3.0/bin:$PATH



```
ubuntu@ip-172-31-8-255:~$ export PATH=/home/ubuntu/kafka_2.12-2.3.0/bin:$PATH
ubuntu@ip-172-31-8-255:~$ |
```

Then change to the /home/ubuntu/kafka_2.12-2.3.0 directory:

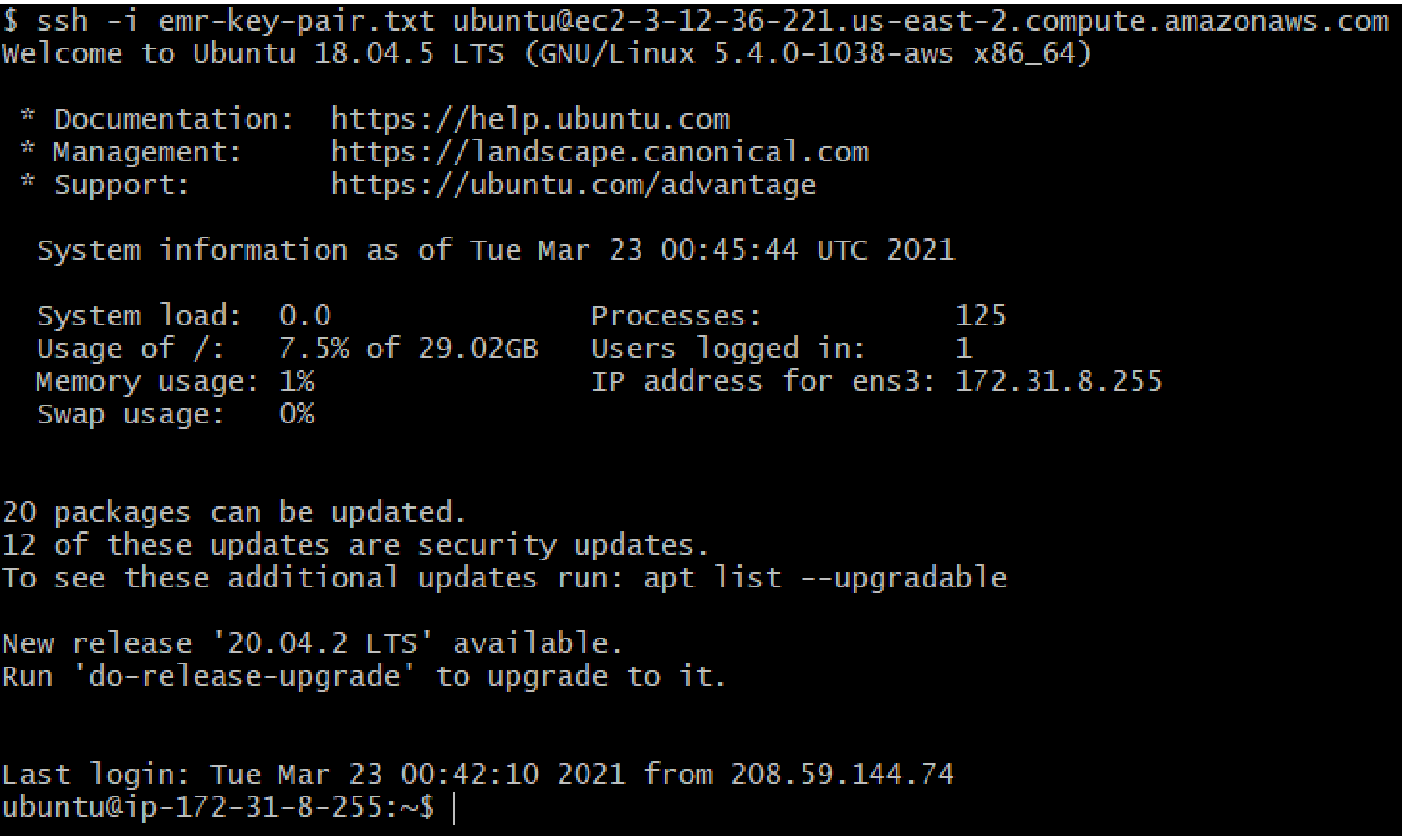
cd /home/ubuntu/kafka_2.12-2.3.0



```
ubuntu@ip-172-31-8-255:~$ cd /home/ubuntu/kafka_2.12-2.3.0/
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ |
```

2) Open a fourth terminal window and enter the following. We will refer to this as the “KT-4” terminal

ssh -i emr-key-pair.pem ubuntu@<Public DNS (IPv4)>



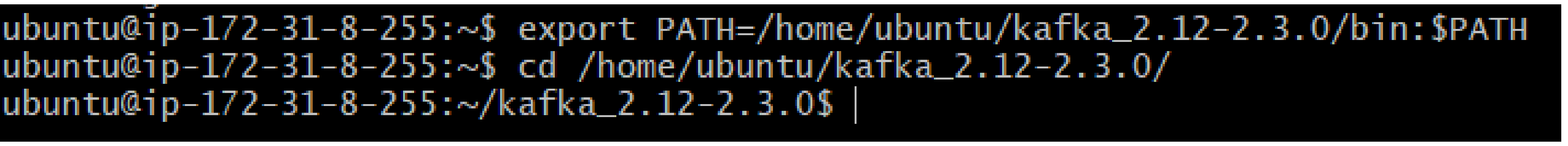
```
$ ssh -i emr-key-pair.txt ubuntu@ec2-3-12-36-221.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1038-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Mar 23 00:45:44 UTC 2021

  System load:  0.0               Processes:           125
  Usage of /:   7.5% of 29.02GB   Users logged in:     1
  Memory usage: 1%                IP address for ens3: 172.31.8.255
  Swap usage:   0%


20 packages can be updated.
12 of these updates are security updates.
To see these additional updates run: apt list --upgradable

New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Tue Mar 23 00:42:10 2021 from 208.59.144.74
ubuntu@ip-172-31-8-255:~$ |
```

Add the install location of the Kafka “bin” directory to your PATH in the KT-1 window as follows:

```
export PATH=/home/ubuntu/kafka_2.12-2.3.0/bin:$PATH
```

Then change to the /home/ubuntu/kafka_2.12-2.3.0 directory:

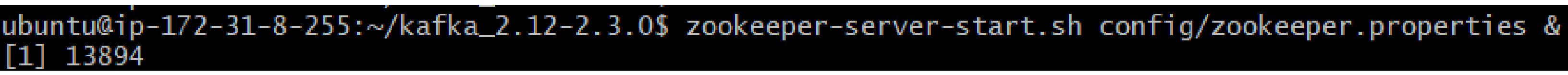
```
cd /home/ubuntu/kafka_2.12-2.3.0
```



```
ubuntu@ip-172-31-8-255:~$ export PATH=/home/ubuntu/kafka_2.12-2.3.0/bin:$PATH
ubuntu@ip-172-31-8-255:~$ cd /home/ubuntu/kafka_2.12-2.3.0/
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ |
```

3) Kafka uses Zookeeper so you need to first start a local ZooKeeper server in the KT-2 terminal window. You use the following command packaged with kafka to get a quick-and-dirty single-node Zookeeper instance. Notice the “&” at the end of the command. It instructs the command shell to run this command in the background. This will allow you to enter further commands into the KT-2 window:

```
zookeeper-server-start.sh config/zookeeper.properties &
```

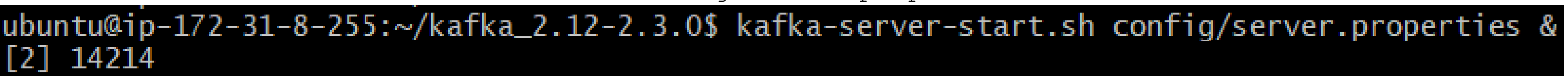


```
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ zookeeper-server-start.sh config/zookeeper.properties &
[1] 13894
```

Note: you may need to press “Enter” a time or two for the Linux prompt to reappear.

4) Now start the Kafka server itself in the KT-2 terminal window as follows. Again, notice the "&" at the end of the command:

```
kafka-server-start.sh config/server.properties &
```



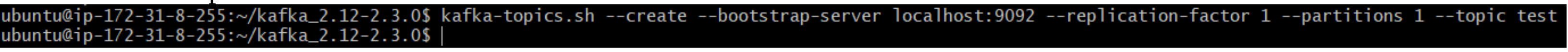
```
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ kafka-server-start.sh config/server.properties &
[2] 14214
```

Note: you may need to press "Enter" a time or two for the Linux prompt to reappear.

Section 4: Working with Kafka

1) Let's create a topic named "test" with a single partition and only one replica. Do so in the KT-3 terminal window:

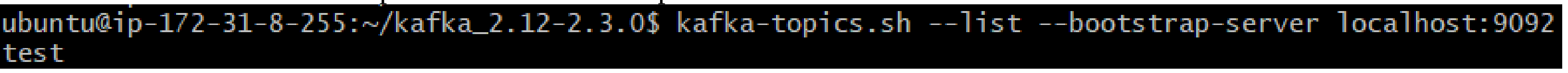
kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic test



```
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic test
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ |
```

2) We can now see that topic if we run the list topic command in the KT-3 terminal window:

kafka-topics.sh --list --bootstrap-server localhost:9092

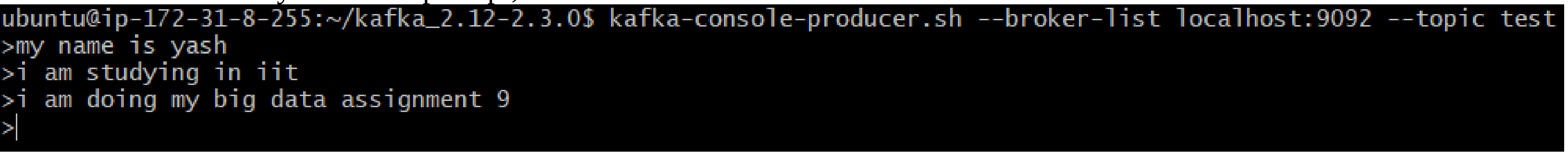


```
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ kafka-topics.sh --list --bootstrap-server localhost:9092
test
```

3) Kafka comes with a command line client that will take input from a file or standard input and send it out as messages to the Kafka cluster. By default, each line will be sent as a separate message. Run the producer in the KT-3 terminal window and then type a few messages into the console to send to the server:

kafka-console-producer.sh --broker-list localhost:9092 --topic test

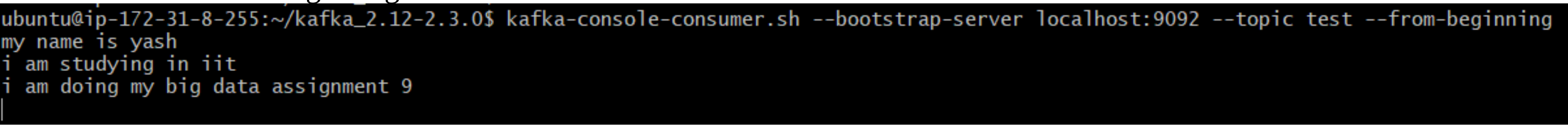
Upon execution of this command you should see a prompt (">") in a few moments. When you see the prompt, enter two or three lines of text.



```
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ kafka-console-producer.sh --broker-list localhost:9092 --topic test
>my name is yash
>i am studying in iit
>i am doing my big data assignment 9
>|
```

4) Kafka also has a command line consumer that will dump out messages to standard output. Run the consumer in the KT-4 terminal window as follows:

kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning



```
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning
my name is yash
i am studying in iit
i am doing my big data assignment 9
|
```

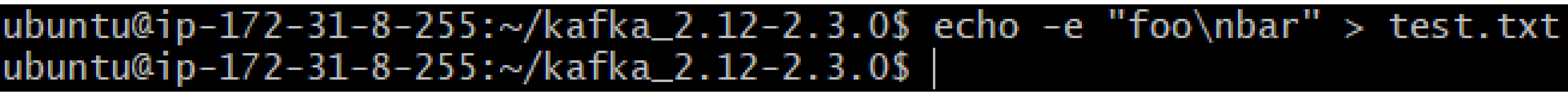
If you have each of the above commands running in a different terminal then you should now be able to see messages you previously typed in and also type new messages into the producer terminal and see them appear in the consumer terminal.

5) To end this part of the demo type control-c into the KT-3 terminal window and KT-4 terminal window to stop the producer and consumer. But keep the KT-3 terminal window and KT-4 terminal window connected to your kafka VM.

6) Writing data from the console and writing it back to the console is a convenient place to start, but you'll probably want to use data from other sources or export data from Kafka to other systems. For many systems, instead of writing custom integration code you can use Kafka Connect to import or export data. Kafka Connect is a tool included with Kafka that imports and exports data to Kafka. It is an extensible tool that runs *connectors*, which implement the custom logic for interacting with an external system. Here we will see how to run Kafka Connect with simple connectors that import data from a file to a Kafka topic and export data from a Kafka topic to a file.

7) Now, we'll start by quickly creating a file to test with using the KT-3 terminal window:

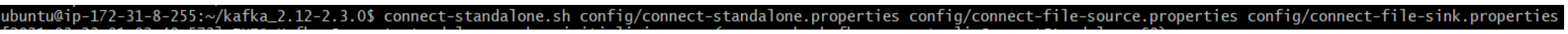
   echo -e "foo\nbar" > test.txt



```
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ echo -e "foo\nbar" > test.txt
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ |
```

8) Next, we'll start two connectors running in standalone mode, using the following command below in the KT-3 window, which means they run in a single, local, dedicated process. We provide three configuration files as parameters. The first is always the configuration for the Kafka Connect process, containing common configuration such as the Kafka brokers to connect to and the serialization format for data. The remaining configuration files each specify a connector to create. These files include a unique connector name, the connector class to instantiate, and any other configuration required by the connect

   connect-standalone.sh config/connect-standalone.properties config/connect-file-source.properties config/connect-file-sink.properties



```
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ connect-standalone.sh config/connect-standalone.properties config/connect-file-source.properties config/connect-file-sink.properties
```

   These sample configuration files, included with Kafka, use the default local cluster configuration you started earlier and create two connectors: the first is a source connector that reads lines from an input file and produces each to a Kafka topic and the second is a sink connector that reads messages from a Kafka topic and produces each as a line in an output file.

   During startup you'll see a number of log messages, including some indicating that the connectors are being instantiated. Once the Kafka Connect process has started, the source connector should start reading lines from test.txt and producing them to the topic connect-test, and the sink connector should start reading messages from the topic connect-test and write them to the file test.sink.txt. We can verify the data has been delivered through the entire pipeline by examining the contents of the output file entering the following command in the KT-4 window

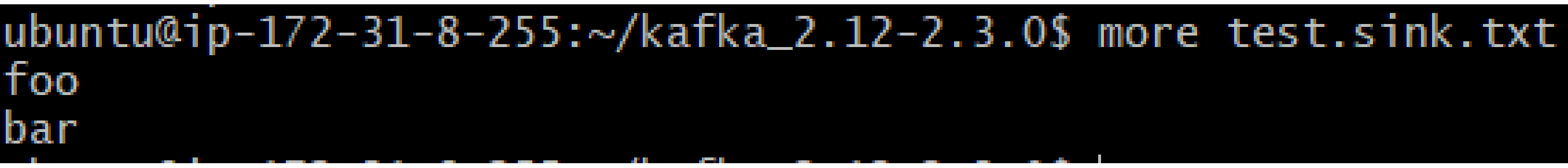
      more test.sink.txt

   And you should see the following:

foo

bar

Now type control-c into the KT-4 terminal window to stop the “more”



```
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ more test.sink.txt
foo
bar
```

9) Note that the data is being stored in the Kafka topic connect-test, so we can also run the following console consumer command in the KT-4 terminal window to see the data in the topic (or use custom consumer code to process it):

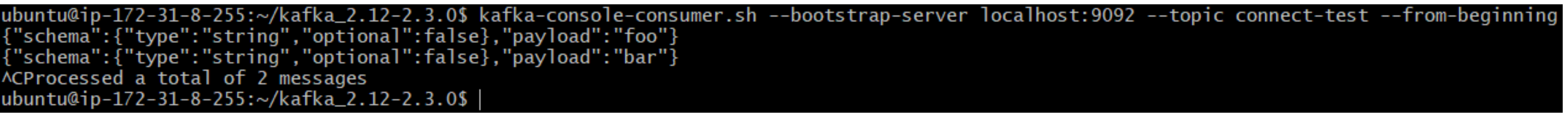
   kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic connect-test --from-beginning

   And you should see the following:

   {"schema":{"type":"string","optional":false},"payload":"foo"}

   {"schema":{"type":"string","optional":false},"payload":"bar"}

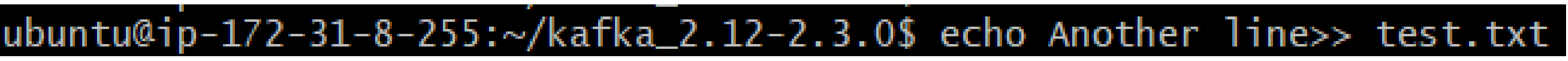
   Now type control-c into the KT-4 window to stop the consumer.



```
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic connect-test --from-beginning
{"schema":{"type":"string","optional":false},"payload":"foo"}
{"schema":{"type":"string","optional":false},"payload":"bar"}
^CProcessed a total of 2 messages
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ |
```

10) The connectors continue to process data, so we can add data to the file and see it move through the pipeline. Enter the following into the KT-3 terminal window

   echo Another line>> test.txt



```
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ echo Another line>> test.txt
```

   You should see the line appear in the console consumer output and in the sink file if you then do the following:

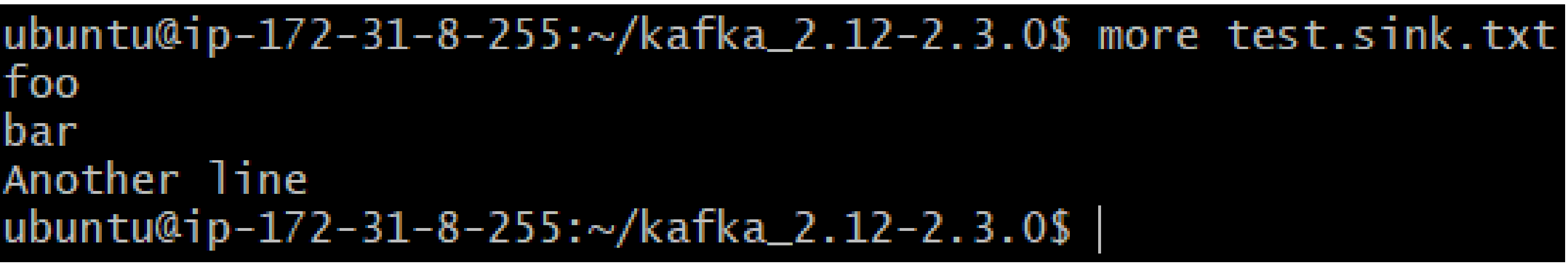
   more test.sink.txt

   And you should see the following:

   foo

   bar

   Another line



```
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ more test.sink.txt
foo
bar
Another line
ubuntu@ip-172-31-8-255:~/kafka_2.12-2.3.0$ |
```

Section 5: Managing your EC2 instance

Your instance:

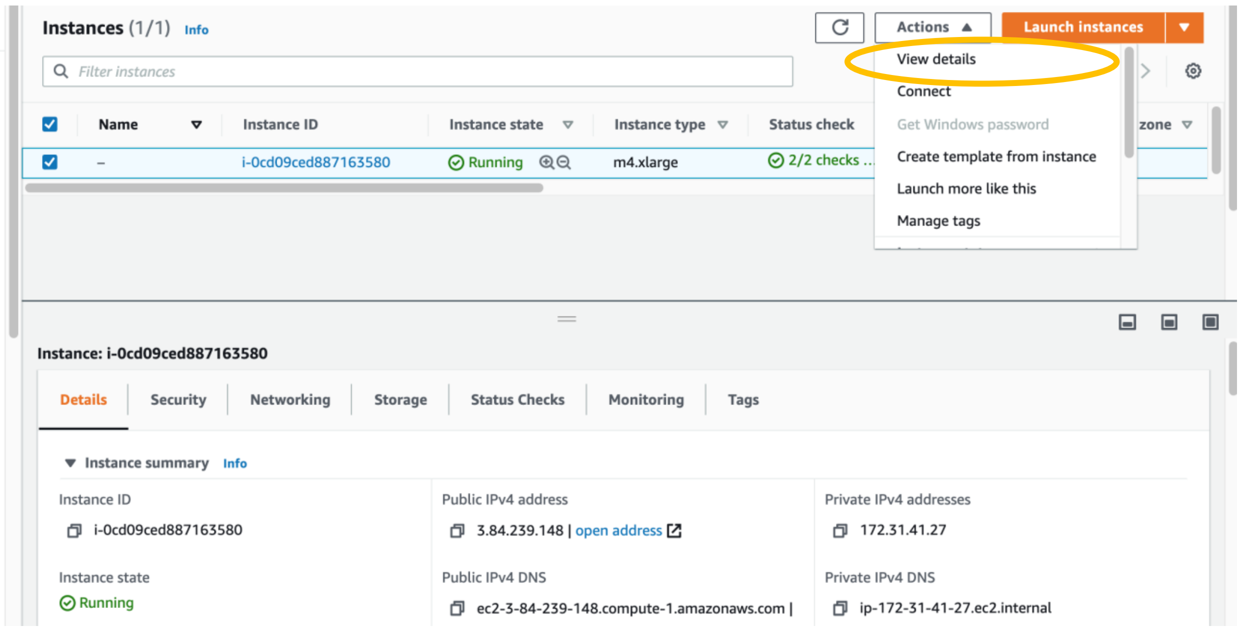
- Some basic facts about managing your instance. If you stop the instance you do not lose any of the software you installed but you will lose all topics and their contents. You will also stop the execution of zookeeper, kafka and any connectors and other related processes. So, when you start the EC2 instance again, you will need to restart these processes and redefine any topics.

- When you terminate the instance you completely lose everything, including the software you installed.

- When you (re)start an instance it will get a new public DNS address, so make sure to find out what it is.

To manage an EC2 instance

a) Click on the actions button
b) Click "View Details"

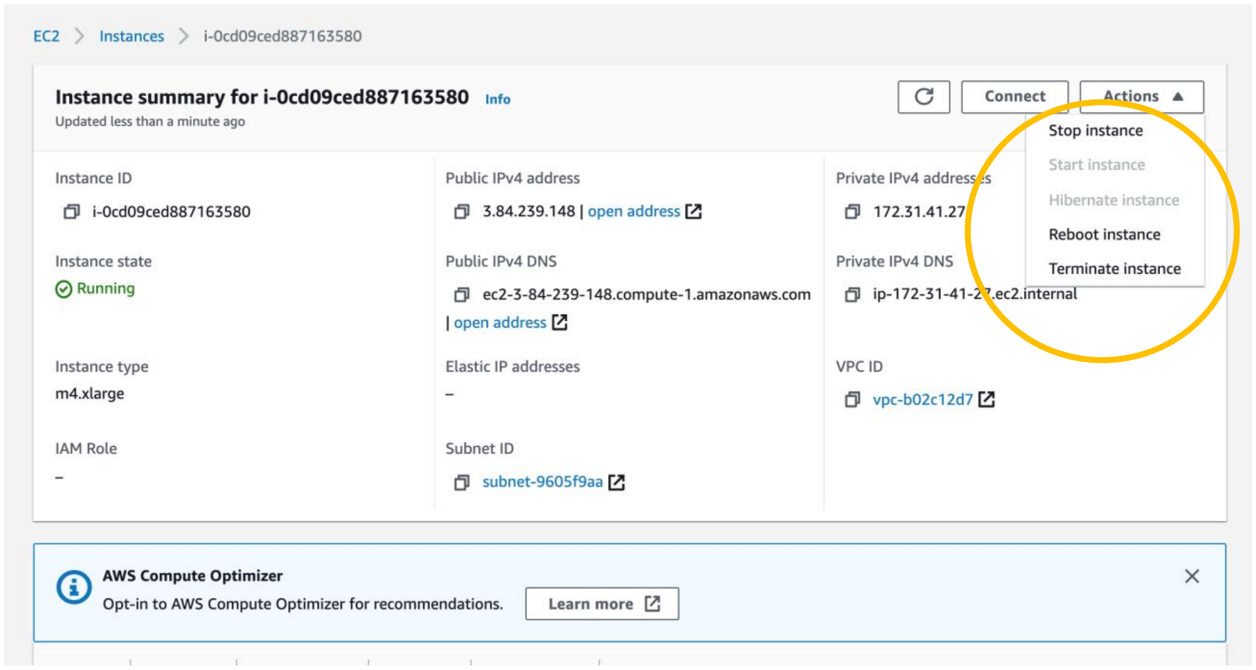


Name	Instance ID	Instance state	Instance type	Status check	zone
–	i-0cd09ced887163580	Running	m4.xlarge	2/2 checks ...	

Instance: i-0cd09ced887163580

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0cd09ced887163580	3.84.239.148 | open address	172.31.41.27
Instance state	**Public IPv4 DNS**	**Private IPv4 DNS**
Running	ec2-3-84-239-148.compute-1.amazonaws.com |	ip-172-31-41-27.ec2.internal

Then select the "Actions" button and a list of actions will be displayed. Now select "Stop instance" "Start instance" or "Terminate instance" as required.



EC2 > Instances > i-0cd09ced887163580

Instance summary for i-0cd09ced887163580 Info

Updated less than a minute ago

Connect | Actions ▲

Instance ID: i-0cd09ced887163580

Public IPv4 address: 3.84.239.148 | open address

Private IPv4 addresses: 172.31.41.27

Instance state: Running

Public IPv4 DNS: ec2-3-84-239-148.compute-1.amazonaws.com | open address

Private IPv4 DNS: ip-172-31-41-27.ec2.internal

Instance type: m4.xlarge

Elastic IP addresses: –

VPC ID: vpc-b02c12d7

IAM Role: –

Subnet ID: subnet-9605f9aa

AWS Compute Optimizer

Opt-in to AWS Compute Optimizer for recommendations. Learn more