Real Time Data Streaming with Apache Spark

Arjun Yadav (309939)
Bhomit Kapoor (310007)
Drashti Pandya (310112)
Karan Sethi (309984)

Appendix

Abstract

Data today are being manipulated from multiple resources which are huge in size and complex as it has incremental addition with existing database. Because data are just not only piece of information but with that we do Analytics too, we shall define a unified or defined repository for Data. In today's IT (Information Technology) Age the data created by any business in faster then ever before, creation of new data is tremendous. To be specific this data is known as LIVE DATA. Live Data say for instance, Flight Data, Social Media feeds, News Reports, Sports Score Data, Satellite Information, Shipping or Logistics domain, Stock Exchange, Telecom Industries Information Transformation, Weather Forecasting, Search Engine Search Repository and so on, they are all suppled within an API. Within this Project our team will work on various Tools, Software's, Technologies and Infrastructure to store the collective information in a console and deploy it over cloud as well as on premise. For developing this project on Live Data Streaming we are using Kafka, Docker, Kubernetes Cluster, Spark, Java, Scala, Zookeeper here all the mentioned technologies have its significant role. API is a key resource for this assignment here we are using Alpha-Vantage API, It will allow us to gather the information for Stock Exchange Market. We are using Kafka to rectify the information/data gathered from API as Source API will bring all information but useful information for our project perspective will be then structured with help of Kafka. Kafka will debug the error and organize the data in parameterized manner. Data from Kafka will then be outputted in JSON format. Now that we have data we will create a pathway or pipeline to Stream data which is key objective of assignment, we will be using Apache Spark to stream Realtime Data. This data will be shipped or wrapped inside a Docker Container and it will be Hosted to a cloud Platform.

Steps for Initiating Project.

Start Zookeeper

1. go to zookeeper file
2. C:\zookeeper-3.5.6>zkserver run....

Start Kafka

1. go to kafka file
2. .\bin\windows\kafka-server-start.bat .\config\server.properties run code

Create Topic

1. Open command prompt and go to Apache Kafka installation directory.
2. Go to \bin\windows directory.
3. Run the following command.
4. kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 -- partitions 1 --topic sql-insert

Create producer

C:\kafka_2.11-2.3.1\bin\windows>kafka-console-producer.bat --broker-list localhost:9092 --topic Kafka_Example

Create Consumer

C:\kafka_2.11-2.3.1\bin\windows>kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic Kafka_Example --from-beginning

Create config folder

create KafkaConfig file
- initialize all config in producerFactory
- config bootstrap, key and value

Create model folder

create User class
- define variable
- set get and set method

Create resource folder

create UserResource class
- KafkaTemplate with object
- @Getmapping to get data from JSON
- use Topic name which already created

To run Spark

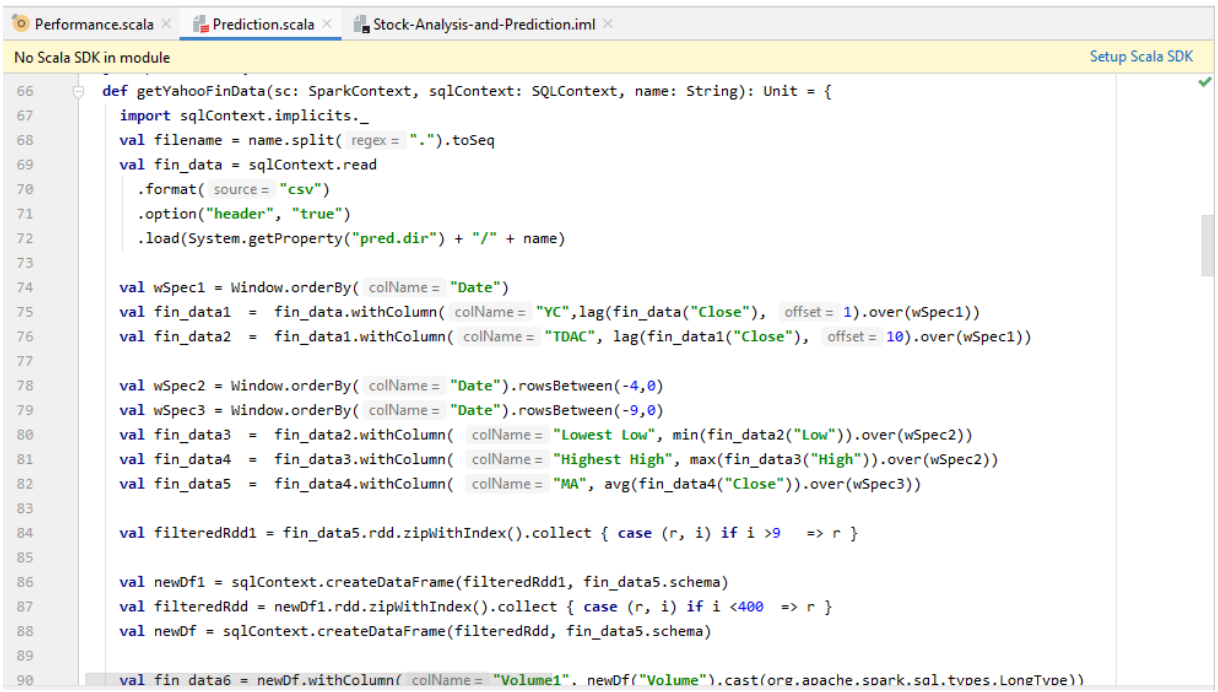
open cmd

run **spark-shell**

create performance.scala

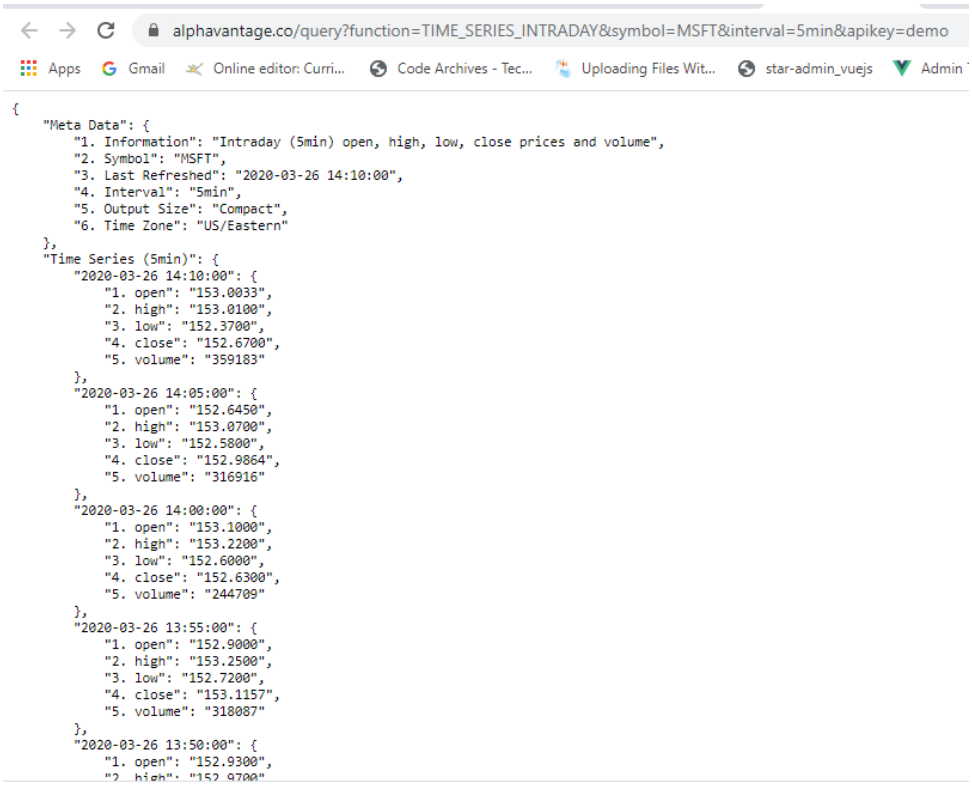
create object file

Create prediction.scala



API data

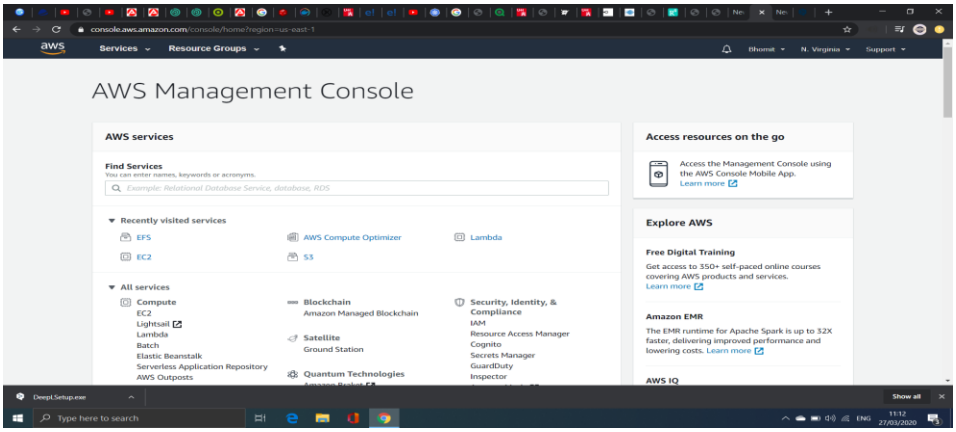
in JSON format

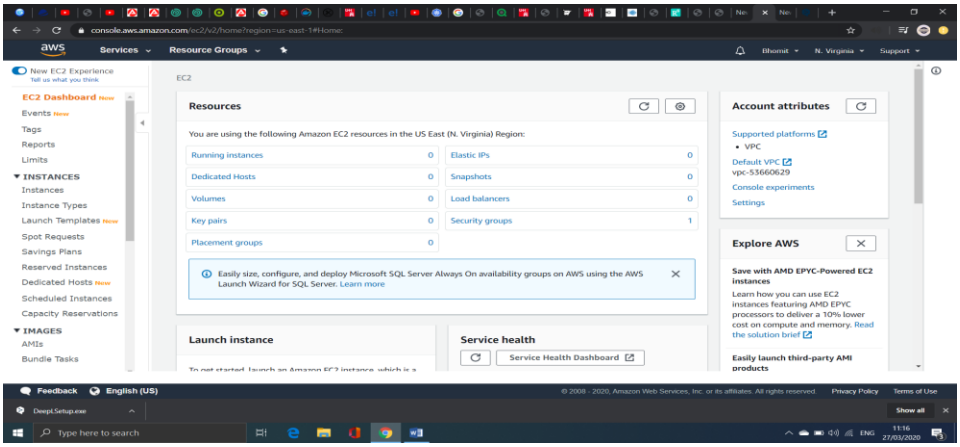


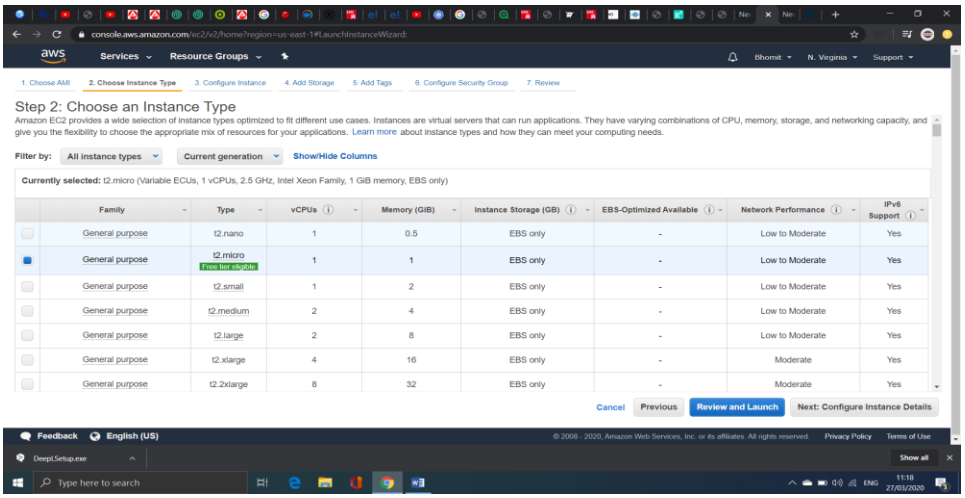
With Alpha-Vantage API we can extract weekly data, monthly data, yearly data, intermediate data etc. apart from this we use some function for 5min, and 1 min duration changes in stock data. Above is an example of 5 min duration of data. This API returns inter day time series (timestamp, high, low, open, close, volume) of the equity specific data. We can get JSON and CSV file to collect and retrieve data from Alphavantage. The most beneficial thing is it gives prices and volume information of the current trading day, update Realtime. We can also use 20+ year historic data to stream and compare with real time.

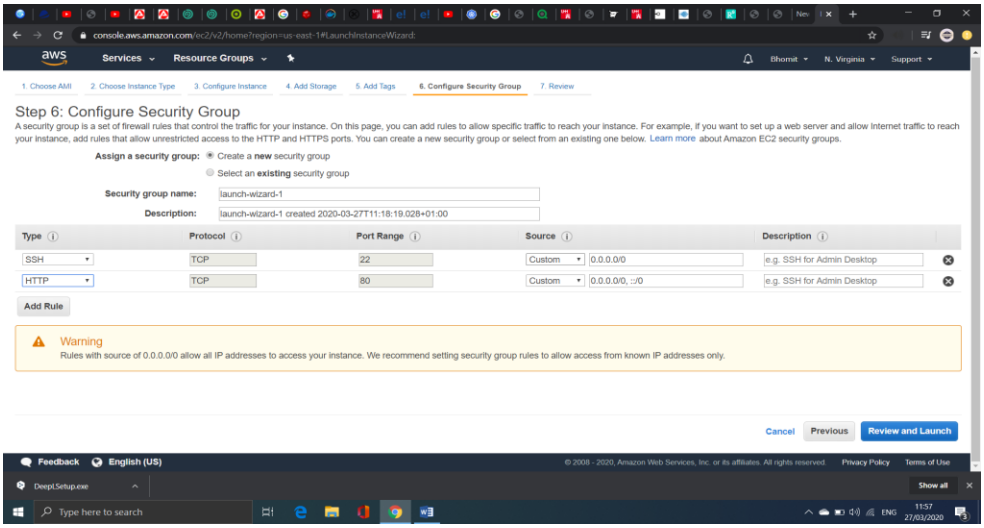
Cloud Deployment

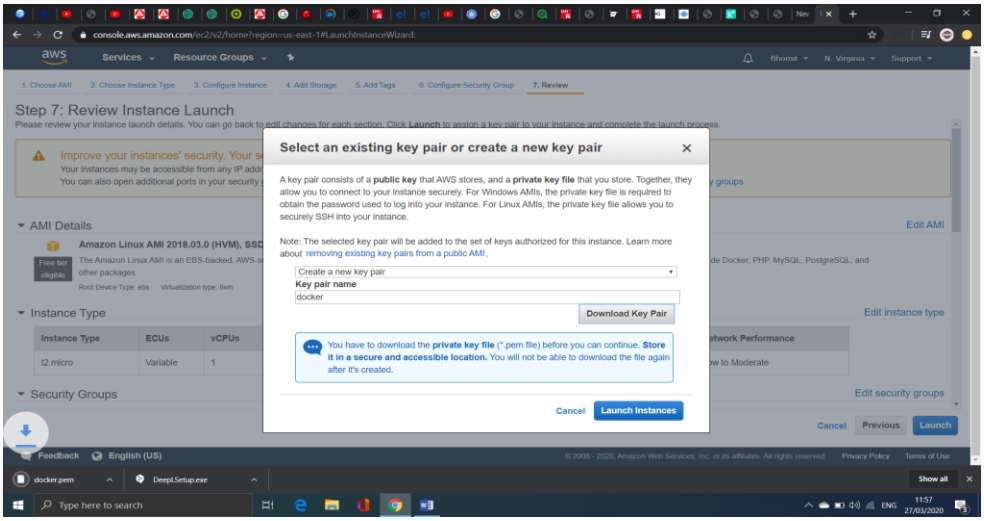
AWS Console Cloud

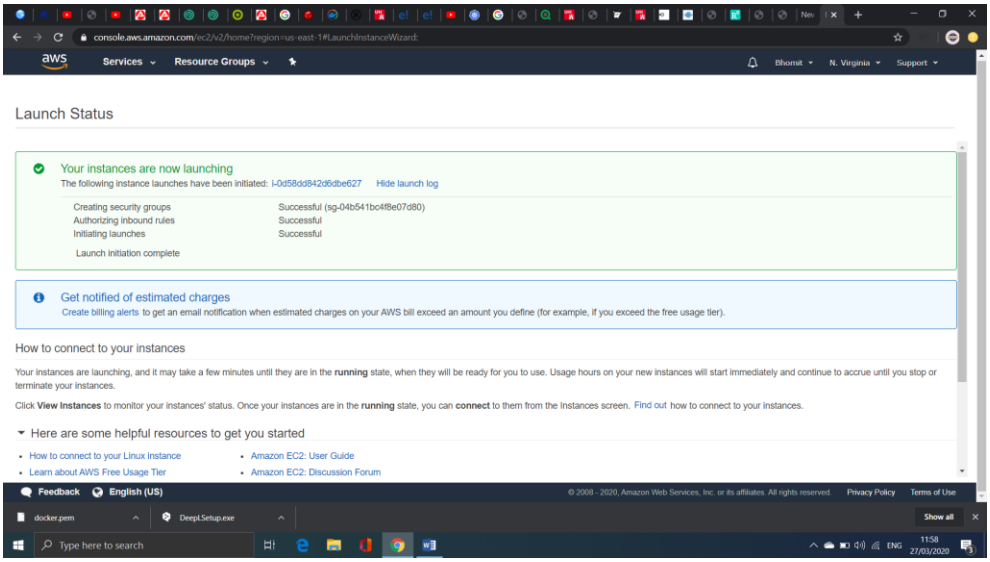


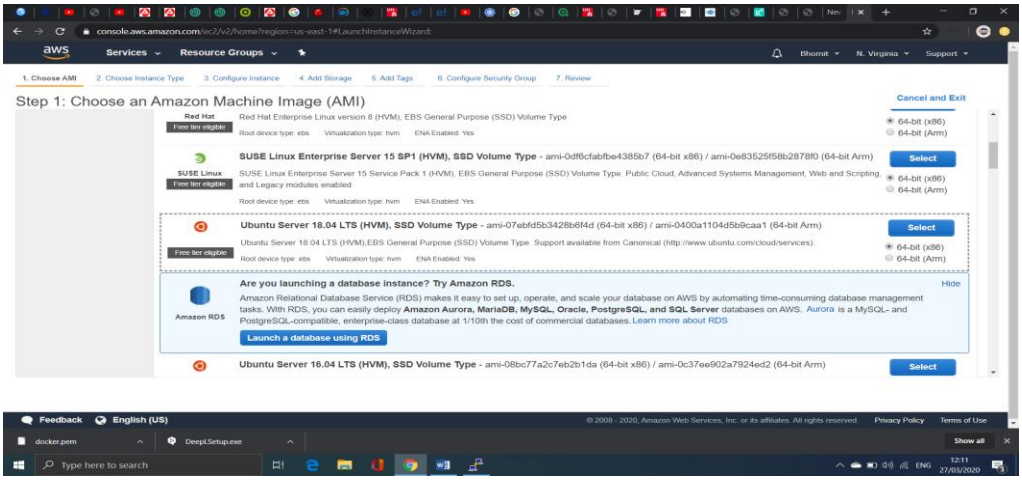


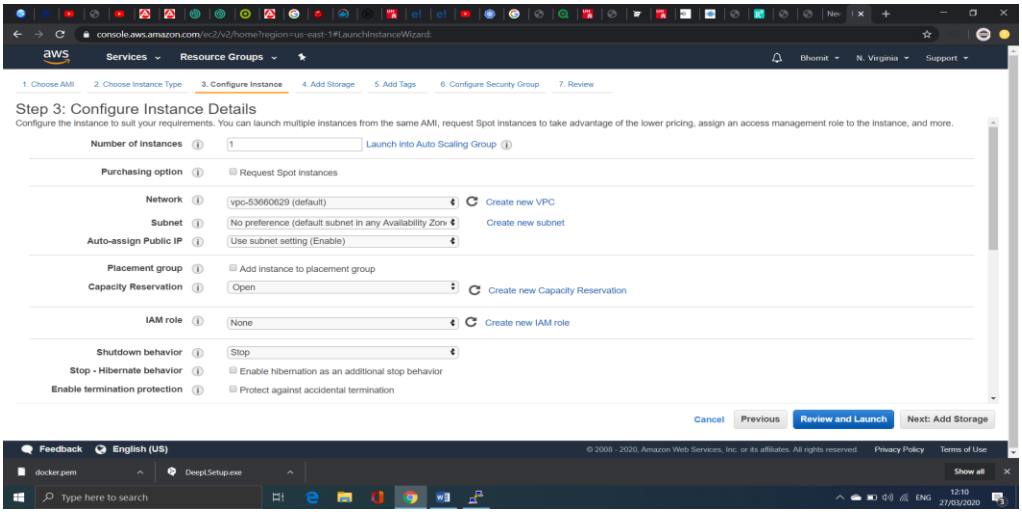


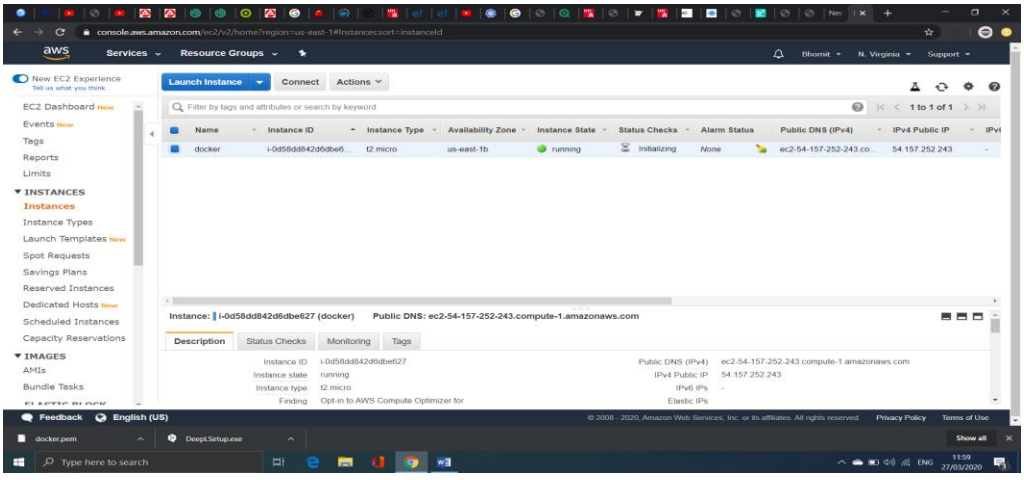

Step 6: Configure Security Group
Assign a security group:
Create a new security group
Select an existing security group
Security group name:
launch-wizard-1
Description:
launch-wizard-1 created 2020-03-27T11:18:19.028+01:00
Type
Protocol
Port Range
Source
Description
SSH
TCP
22
Custom
0.0.0.0/0
e.g. SSH for Admin Desktop
HTTP
TCP
80
Custom
0.0.0.0/0, ::/0
e.g. SSH for Admin Desktop
Add Rule
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.
Cancel
Previous
Review and Launch


Step 7: Review Instance Launch
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.
Create a new key pair
Key pair name
docker
Download Key Pair
You have to download the private key file (*.pem file) before you can continue. Store it in a secure and accessible location. You will not be able to download the file again after it's created.
Cancel
Launch Instances
AMI Details
Instance Type
Security Groups
Edit AMI
Edit instance type
Edit security groups
Cancel
Previous
Launch

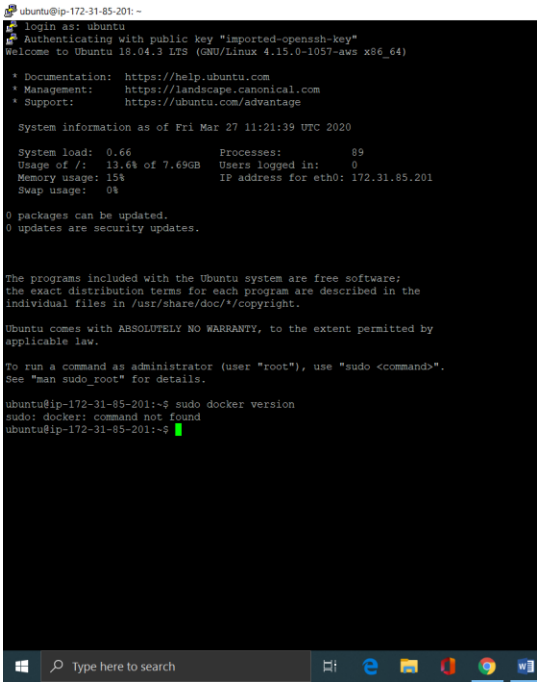

Launch Status
Your instances are now launching
The following instance launches have been initiated: i-0d58dd842d6dbe627
Hide launch log
Creating security groups
Successful (sg-04b541bc4f8e07d80)
Authorizing inbound rules
Successful
Initiating launches
Successful
Launch initiation complete
Get notified 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

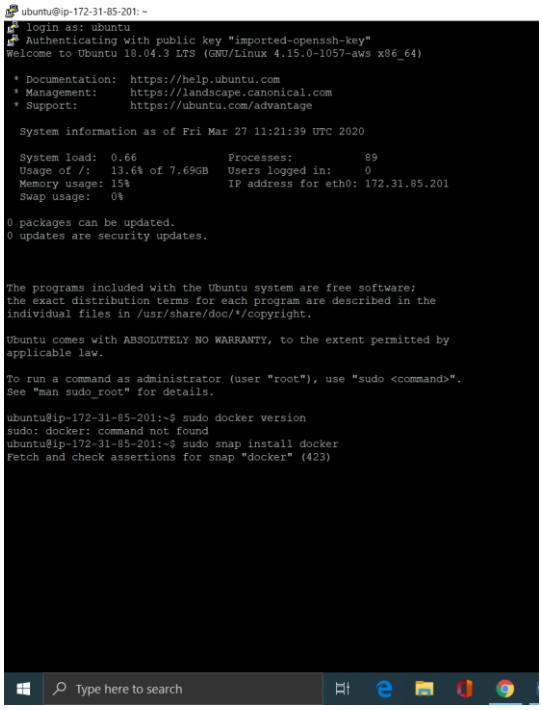

Step 1: Choose an Amazon Machine Image (AMI)
Cancel and Exit
Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type
SUSE Linux Enterprise Server 15 SP1 (HVM), SSD Volume Type - ami-0df6cfabfbe4385b7 (64-bit x86) / ami-0e83525f58b2878f0 (64-bit Arm)
SUSE Linux Enterprise Server 15 Service Pack 1 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting and Legacy modules enabled.
Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-07ebfd5b3428b6f4d (64-bit x86) / ami-0400a1104d5b9caa1 (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).
Are you launching a database instance? Try Amazon RDS.
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-08bc77a2c7eb2b1da (64-bit x86) / ami-0c37ee902a7924ed2 (64-bit Arm)
Select
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

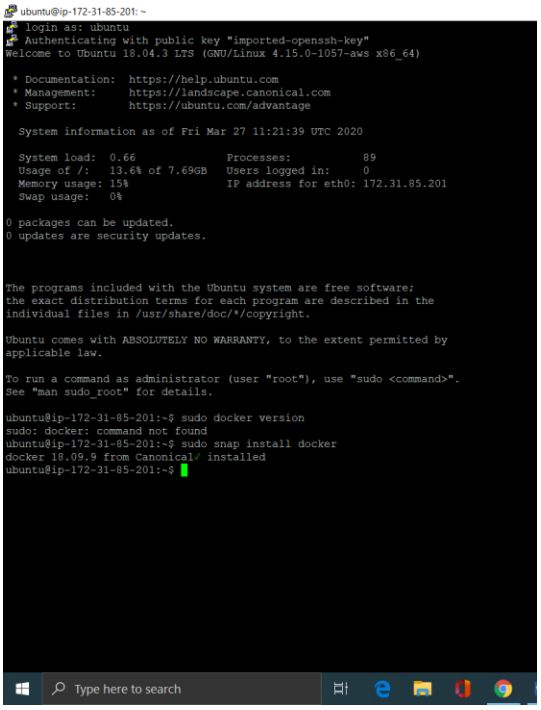

Step 3: Configure Instance Details
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.
Number of instances
Launch into Auto Scaling Group
Purchasing option
Request Spot instances
Network
vpc-53660629 (default)
Create new VPC
Subnet
No preference (default subnet in any Availability Zone
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
Shutdown behavior
Stop
Stop - Hibernate behavior
Enable hibernation as an additional stop behavior
Enable termination protection
Protect against accidental termination
Cancel
Previous
Review and Launch
Next: Add Storage

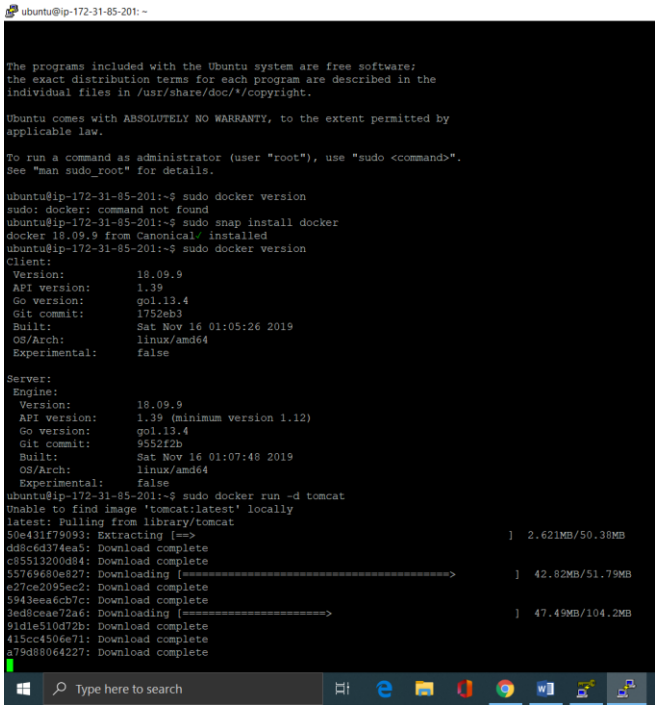

Launch Instance
Connect
Actions
docker
i-0d58dd842d6dbe627
t2.micro
us-east-1b
running
Initializing
None
ec2-54-157-252-243.compute-1.amazonaws.com
54.157.252.243
Instance: i-0d58dd842d6dbe627 (docker)
Public DNS: ec2-54-157-252-243.compute-1.amazonaws.com
Description
Status Checks
Monitoring
Tags

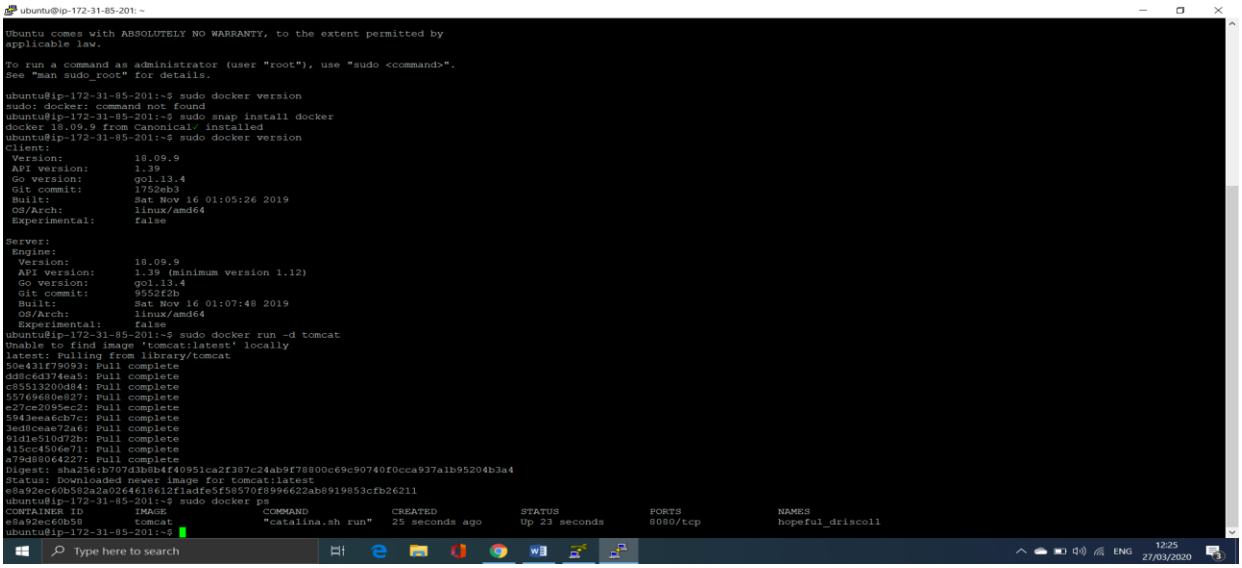
Output window for Ubuntu instance

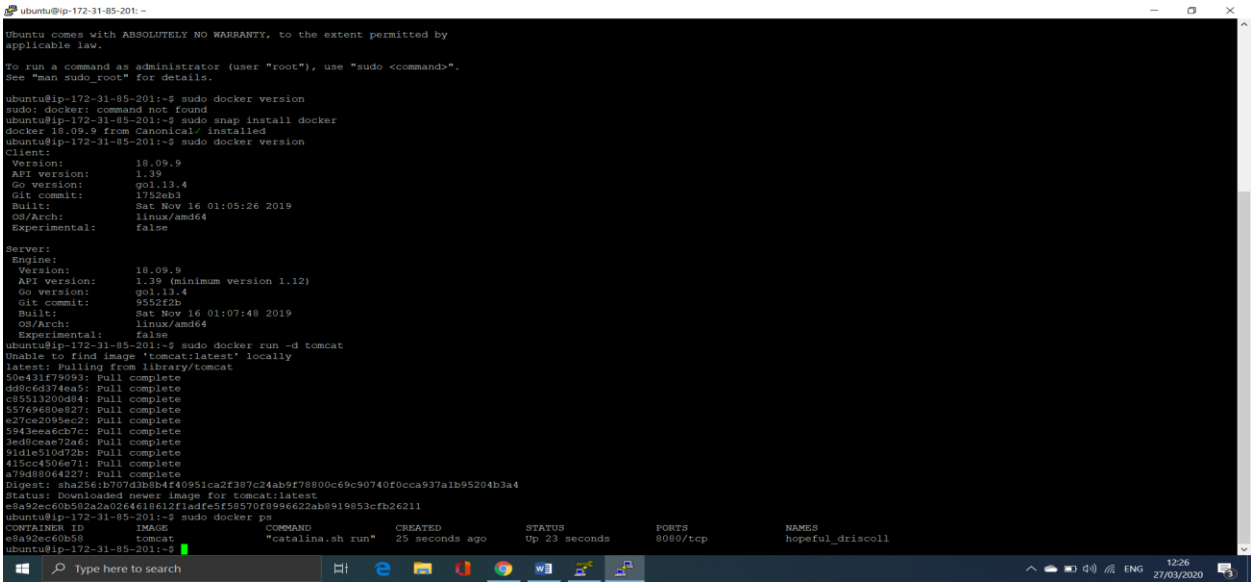


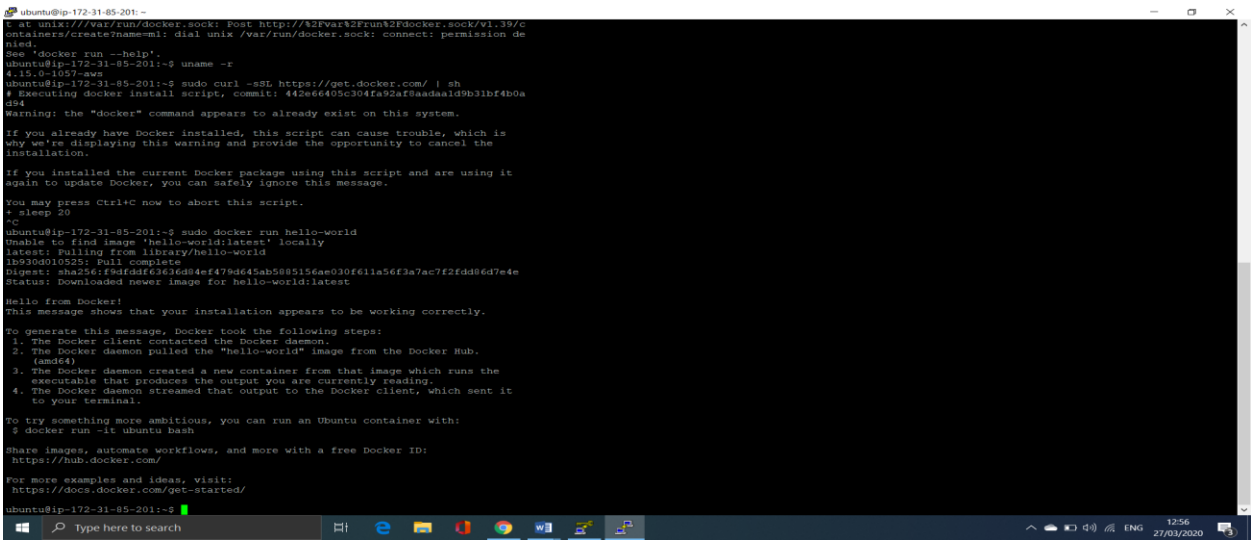


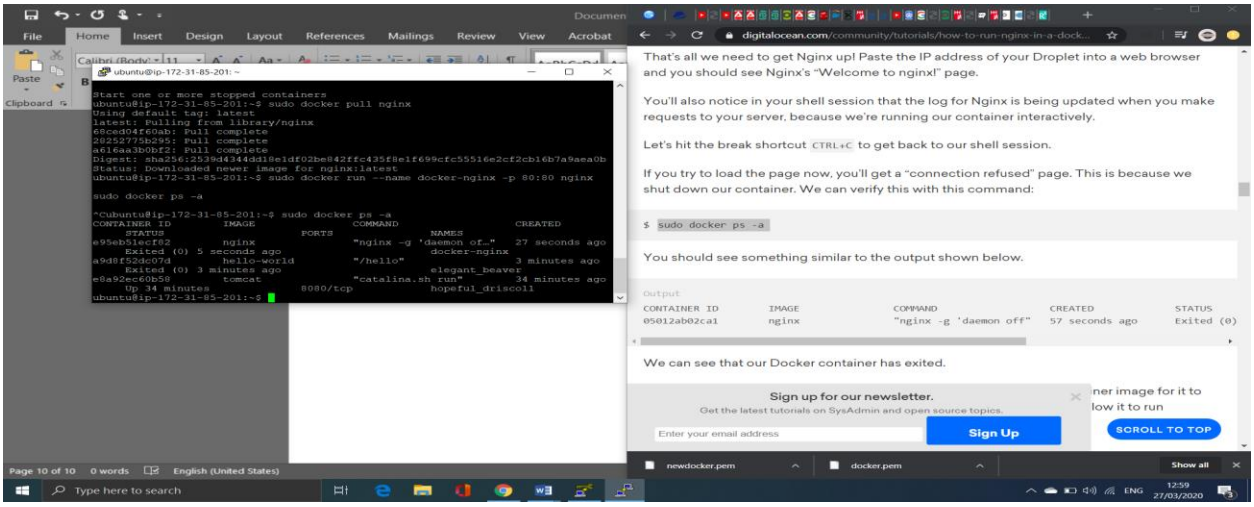


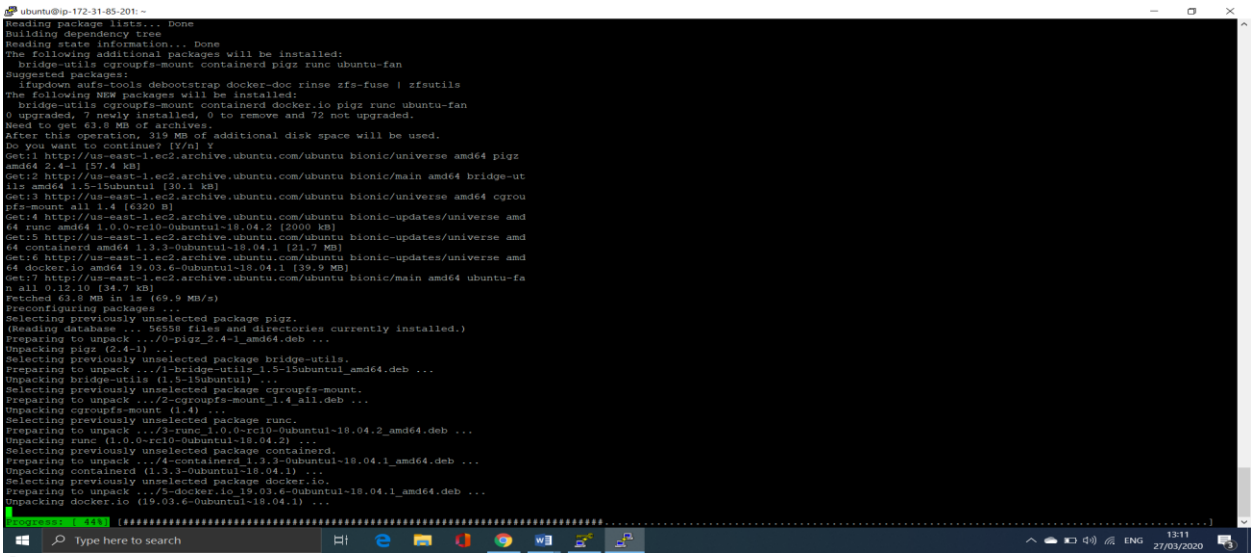


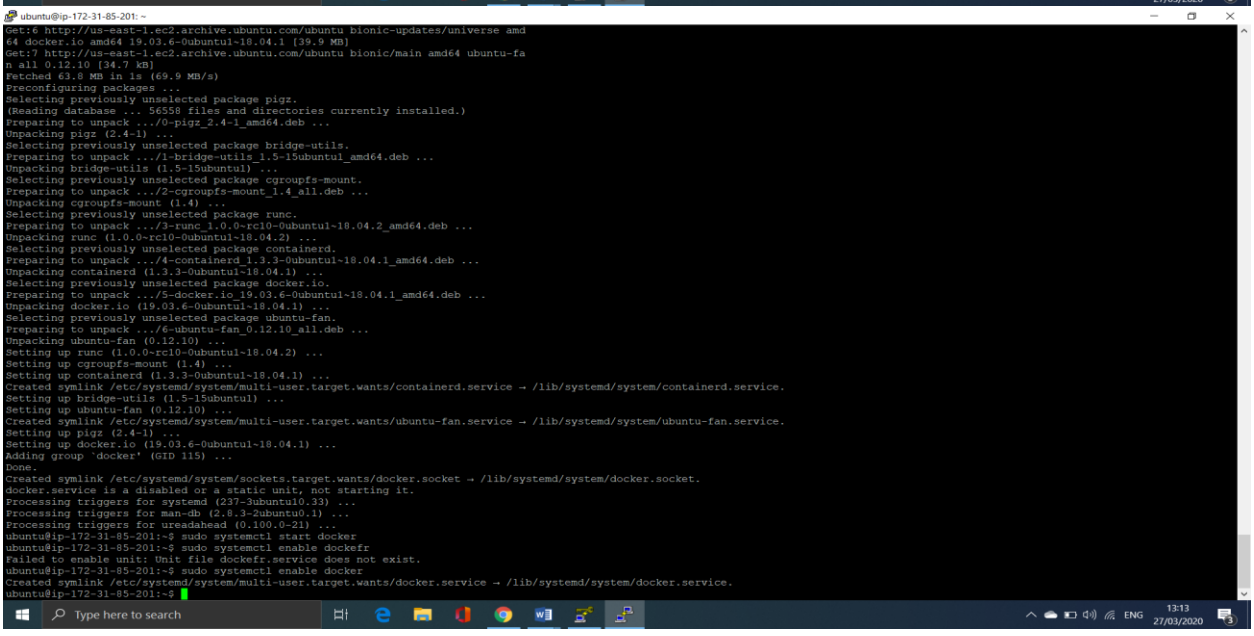

ubuntu@ip-172-31-85-201: ~
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
ubuntu@ip-172-31-85-201:~$ sudo docker version
sudo: docker: command not found
ubuntu@ip-172-31-85-201:~$ sudo snap install docker
docker 18.09.9 from Canonical✓ installed
ubuntu@ip-172-31-85-201:~$ sudo docker version
Client:
Version: 18.09.9
API version: 1.39
Go version: go1.13.4
Git commit: 1752eb3
Built: Sat Nov 16 01:05:26 2019
OS/Arch: linux/amd64
Experimental: false
Server:
Engine:
Version: 18.09.9
API version: 1.39 (minimum version 1.12)
Go version: go1.13.4
Git commit: 9552f2b
Built: Sat Nov 16 01:07:48 2019
OS/Arch: linux/amd64
Experimental: false
ubuntu@ip-172-31-85-201:~$ sudo docker run -d tomcat
Unable to find image 'tomcat:latest' locally
latest: Pulling from library/tomcat
50e431f79093: Pull complete
dd8c6d374ea5: Pull complete
c85513200d84: Pull complete
55769680e827: Pull complete
e27ce2095ec2: Pull complete
5943eea6cb7c: Pull complete
3ed8ceae72a6: Pull complete
91d1e510d72b: Pull complete
415cc4506e71: Pull complete
a79d88064227: Pull complete
Digest: sha256:b707d3b8b4f40951ca2f387c24ab9f78800c69c90740f0cca937a1b95204b3a4
Status: Downloaded newer image for tomcat:latest
e8a92ec60b582a2a026461861 2f1adfe5f585570f8996622ab8919853cfb26211
ubuntu@ip-172-31-85-201:~$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
e8a92ec60b58 tomcat "catalina.sh run" 25 seconds ago Up 23 seconds 8080/tcp hopeful_driscoll
ubuntu@ip-172-31-85-201:~$
Type here to search
ENG
12:25
27/03/2020


ubuntu@ip-172-31-85-201: ~
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-172-31-85-201:~$ sudo docker version
sudo: docker: command not found
ubuntu@ip-172-31-85-201:~$ sudo snap install docker
docker 18.09.9 from Canonical✓ installed
ubuntu@ip-172-31-85-201:~$ sudo docker version
Client:
Version: 18.09.9
API version: 1.39
Go version: go1.13.4
Git commit: 1752eb3
Built: Sat Nov 16 01:05:26 2019
OS/Arch: linux/amd64
Experimental: false
Server:
Engine:
Version: 18.09.9
API version: 1.39 (minimum version 1.12)
Go version: go1.13.4
Git commit: 9552f2b
Built: Sat Nov 16 01:07:48 2019
OS/Arch: linux/amd64
Experimental: false
ubuntu@ip-172-31-85-201:~$ sudo docker run -d tomcat
Unable to find image 'tomcat:latest' locally
latest: Pulling from library/tomcat
50e431f79093: Pull complete
dd8c6d374ea5: Pull complete
c85513200d84: Pull complete
55769680e827: Pull complete
e27ce2095ec2: Pull complete
5943eea6cb7c: Pull complete
3ed8ceae72a6: Pull complete
91d1e510d72b: Pull complete
415cc4506e71: Pull complete
a79d88064227: Pull complete
Digest: sha256:b707d3b8b4f40951ca2f387c24ab9f78800c69c90740f0cca937a1b95204b3a4
Status: Downloaded newer image for tomcat:latest
e8a92ec60b582a2a026461861 2f1adfe5f585570f8996622ab8919853cfb26211
ubuntu@ip-172-31-85-201:~$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
e8a92ec60b58 tomcat "catalina.sh run" 25 seconds ago Up 23 seconds 8080/tcp hopeful_driscoll
ubuntu@ip-172-31-85-201:~$
Type here to search
ENG
12:26
27/03/2020

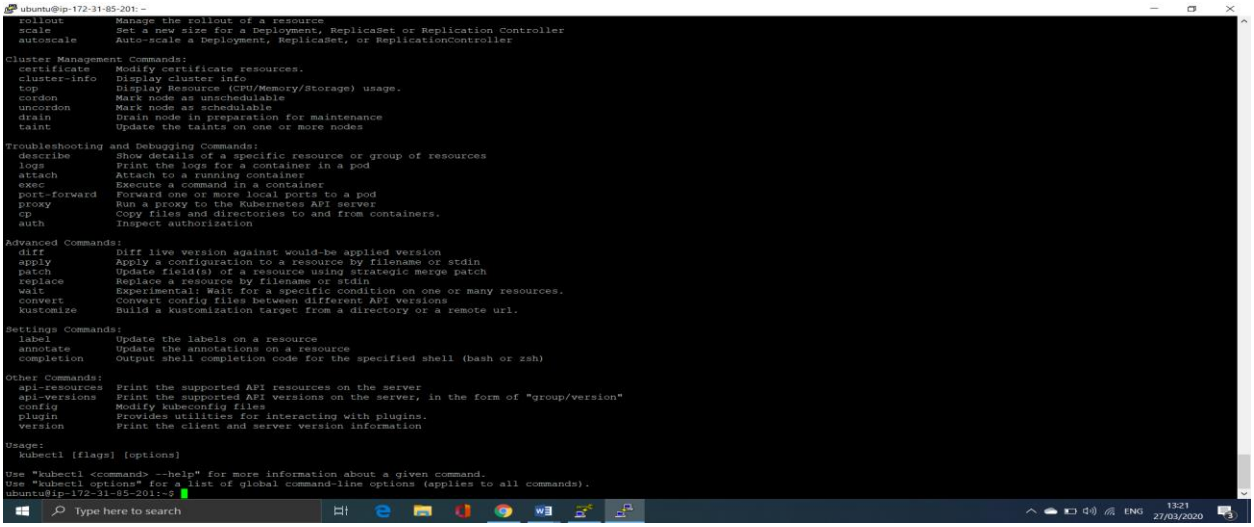

ubuntu@ip-172-31-85-201: ~
t at unix:///var/run/docker.sock: Post http://%2Fvar%2Frun%2Fdocker.sock/v1.39/c
ontainers/create?name=ml: dial unix /var/run/docker.sock: connect: permission de
nied.
See 'docker run --help'.
ubuntu@ip-172-31-85-201:~$ uname -r
4.15.0-1057-aws
ubuntu@ip-172-31-85-201:~$ sudo curl -sSL https://get.docker.com/ | sh
Executing docker install script, commit: 442e66405c304fa92af8aadaa1d9b31bf4b0a
d9d
Warning: the "docker" command appears to already exist on this system.
If you already have Docker installed, this script can cause trouble, which is
why we're displaying this warning and provide the opportunity to cancel the
installation.
If you installed the current Docker package using this script and are using it
again to update Docker, you can safely ignore this message.
You may press Ctrl+C now to abort this script.
+ sleep 20
^C
ubuntu@ip-172-31-85-201:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:f9dfddf63636d84ef479d645ab5885156ae030f611a56f3a7ac7f2fdd86d7e4e
Status: Downloaded newer image for hello-world:latest
Hello from Docker!
This message shows that your installation appears to be working correctly.
To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the
executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
to your terminal.
To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash
Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/
For more examples and ideas, visit:
https://docs.docker.com/get-started/
ubuntu@ip-172-31-85-201:~$
Type here to search
ENG
12:56
27/03/2020

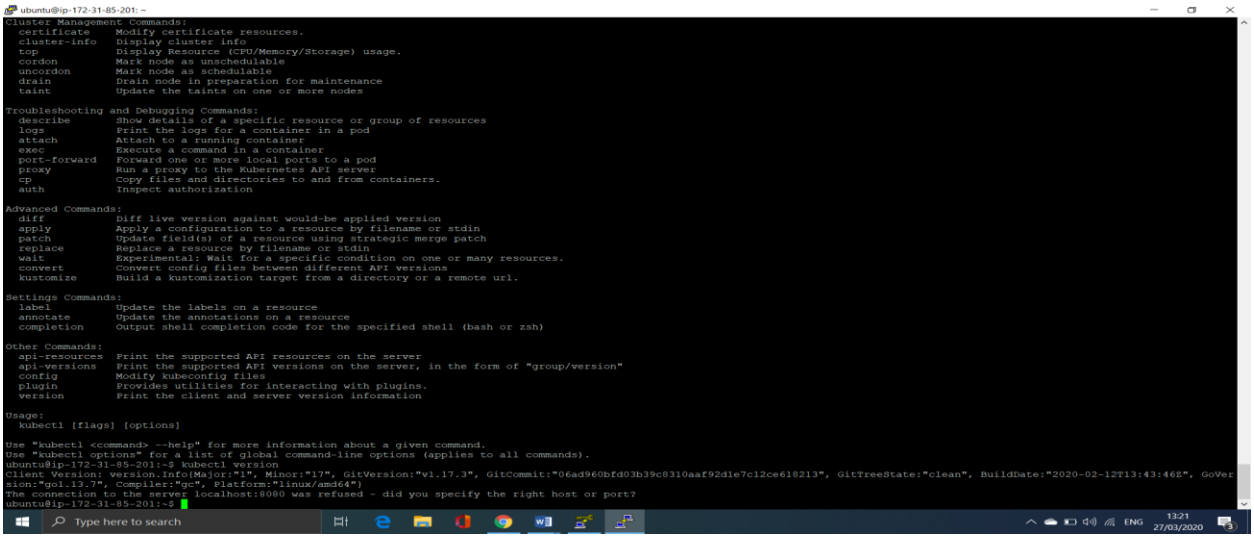

ubuntu@ip-172-31-85-201: ~
Start one or more stopped containers
ubuntu@ip-172-31-85-201:~$ sudo docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
68ced04f60ab: Pull complete
26252775b295: Pull complete
a616aa3b0bf2: Pull complete
Digest: sha256:2539d4344dd18e1df02be842ffc435f8e1f699cfc55516e2cf2cb16b7a9aea0b
Status: Downloaded newer image for nginx:latest
ubuntu@ip-172-31-85-201:~$ sudo docker run --name docker-nginx -p 80:80 nginx
sudo docker ps -a
^Cubuntu@ip-172-31-85-201:~$ sudo docker ps -a
CONTAINER ID IMAGE COMMAND CREATED
STATUS PORTS NAMES
e95eb51ecf82 nginx "nginx -g 'daemon of…" 27 seconds ago
Exited (0) 5 seconds ago docker-nginx
a9d8f52dc07d hello-world "/hello" 3 minutes ago
Exited (0) 3 minutes ago elegant_beaver
e8a92ec60b58 tomcat "catalina.sh run" 34 minutes ago
Up 34 minutes 8080/tcp hopeful_driscoll
ubuntu@ip-172-31-85-201:~$
That's all we need to get Nginx up! Paste the IP address of your Droplet into a web browser and you should see Nginx's "Welcome to nginx!" page.
You'll also notice in your shell session that the log for Nginx is being updated when you make requests to your server, because we're running our container interactively.
Let's hit the break shortcut CTRL+C to get back to our shell session.
If you try to load the page now, you'll get a "connection refused" page. This is because we shut down our container. We can verify this with this command:
$ sudo docker ps -a
You should see something similar to the output shown below.
Output
CONTAINER ID IMAGE COMMAND CREATED STATUS
05012ab02ca1 nginx "nginx -g 'daemon off" 57 seconds ago Exited (0)
We can see that our Docker container has exited.
Sign up for our newsletter.
Get the latest tutorials on SysAdmin and open source topics.
Enter your email address
Sign Up
SCROLL TO TOP
newdocker.pem
docker.pem
Show all


ubuntu@ip-172-31-85-201: ~
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
bridge-utils cgroupfs-mount containerd pigz runc ubuntu-fan
Suggested packages:
ifupdown aufs-tools debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
bridge-utils cgroupfs-mount containerd docker.io pigz runc ubuntu-fan
0 upgraded, 7 newly installed, 0 to remove and 72 not upgraded.
Need to get 63.8 MB of archives.
After this operation, 319 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic/universe amd64 pigz amd64 2.4-1 [57.4 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 bridge-utils amd64 1.5-15ubuntu1 [30.1 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic/universe amd64 cgroupfs-mount all 1.4 [6320 B]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 runc amd64 1.0.0~rc10-0ubuntu1~18.04.2 [2000 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 containerd amd64 1.3.3-0ubuntu1~18.04.1 [21.7 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 docker.io amd64 19.03.6-0ubuntu1~18.04.1 [39.9 MB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 ubuntu-fan all 0.12.10 [34.7 kB]
Fetched 63.8 MB in 1s (69.9 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 56558 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.4-1_amd64.deb ...
Unpacking pigz (2.4-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.5-15ubuntu1_amd64.deb ...
Unpacking bridge-utils (1.5-15ubuntu1) ...
Selecting previously unselected package cgroupfs-mount.
Preparing to unpack .../2-cgroupfs-mount_1.4_all.deb ...
Unpacking cgroupfs-mount (1.4) ...
Selecting previously unselected package runc.
Preparing to unpack .../3-runc_1.0.0~rc10-0ubuntu1~18.04.2_amd64.deb ...
Unpacking runc (1.0.0~rc10-0ubuntu1~18.04.2) ...
Selecting previously unselected package containerd.
Preparing to unpack .../4-containerd_1.3.3-0ubuntu1~18.04.1_amd64.deb ...
Unpacking containerd (1.3.3-0ubuntu1~18.04.1) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../5-docker.io_19.03.6-0ubuntu1~18.04.1_amd64.deb ...
Unpacking docker.io (19.03.6-0ubuntu1~18.04.1) ...
Progress: (44%) [


ubuntu@ip-172-31-85-201: ~
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 docker.io amd64 19.03.6-0ubuntu1~18.04.1 [39.9 MB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 ubuntu-fan all 0.12.10 [34.7 kB]
Fetched 63.8 MB in 1s (69.9 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 56558 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.4-1_amd64.deb ...
Unpacking pigz (2.4-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.5-15ubuntu1_amd64.deb ...
Unpacking bridge-utils (1.5-15ubuntu1) ...
Selecting previously unselected package cgroupfs-mount.
Preparing to unpack .../2-cgroupfs-mount_1.4_all.deb ...
Unpacking cgroupfs-mount (1.4) ...
Selecting previously unselected package runc.
Preparing to unpack .../3-runc_1.0.0~rc10-0ubuntu1~18.04.2_amd64.deb ...
Unpacking runc (1.0.0~rc10-0ubuntu1~18.04.2) ...
Selecting previously unselected package containerd.
Preparing to unpack .../4-containerd_1.3.3-0ubuntu1~18.04.1_amd64.deb ...
Unpacking containerd (1.3.3-0ubuntu1~18.04.1) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../5-docker.io_19.03.6-0ubuntu1~18.04.1_amd64.deb ...
Unpacking docker.io (19.03.6-0ubuntu1~18.04.1) ...
Selecting previously unselected package ubuntu-fan.
Preparing to unpack .../6-ubuntu-fan_0.12.10_all.deb ...
Unpacking ubuntu-fan (0.12.10) ...
Setting up runc (1.0.0~rc10-0ubuntu1~18.04.2) ...
Setting up cgroupfs-mount (1.4) ...
Setting up containerd (1.3.3-0ubuntu1~18.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/containerd.service.
Setting up bridge-utils (1.5-15ubuntu1) ...
Setting up ubuntu-fan (0.12.10) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /lib/systemd/system/ubuntu-fan.service.
Setting up pigz (2.4-1) ...
Setting up docker.io (19.03.6-0ubuntu1~18.04.1) ...
Adding group `docker' (GID 115) ...
Done.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
docker.service is a disabled or a static unit, not starting it.
Processing triggers for systemd (237-3ubuntu10.33) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
ubuntu@ip-172-31-85-201:~$ sudo systemctl start docker
ubuntu@ip-172-31-85-201:~$ sudo systemctl enable dockefr
Failed to enable unit: Unit file dockefr.service does not exist.
ubuntu@ip-172-31-85-201:~$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
ubuntu@ip-172-31-85-201:~$

Enable Docker





Conclusion

By utilizing the technologies such as Kafka, Spark, Docker, Kuberneties. We were able to achieve the key objective of Streaming Live Data. The data we got was from Alpha-Vantage API for real time as well as historical data of stock market. The output is getting stored to the docker container which than will be orchestrated by kuberneties clusters.