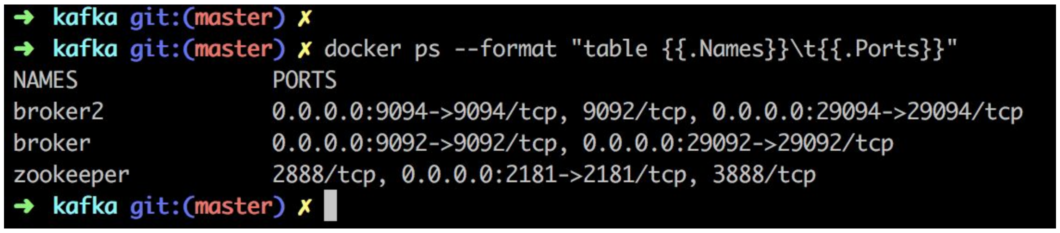
How does Kafka work?

Step 1: Up the zookeeper, brokers in docker

docker-compose up *broker zookeeper broker2*
docker ps



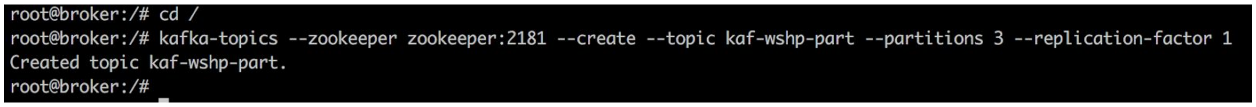
```
➜  kafka git:(master) ✗
➜  kafka git:(master) ✗ docker ps --format "table {{.Names}}\t{{.Ports}}"
NAMES               PORTS
broker2             0.0.0.0:9094->9094/tcp, 9092/tcp, 0.0.0.0:29094->29094/tcp
broker              0.0.0.0:9092->9092/tcp, 0.0.0.0:29092->29092/tcp
zookeeper           2888/tcp, 0.0.0.0:2181->2181/tcp, 3888/tcp
➜  kafka git:(master) ✗
```

Step 2: Create a topic with partitions

You can create a topic from anywhere as long as you give the correct zookeeper address.

Step 2.1: Create a topic with partitions in one of the brokers

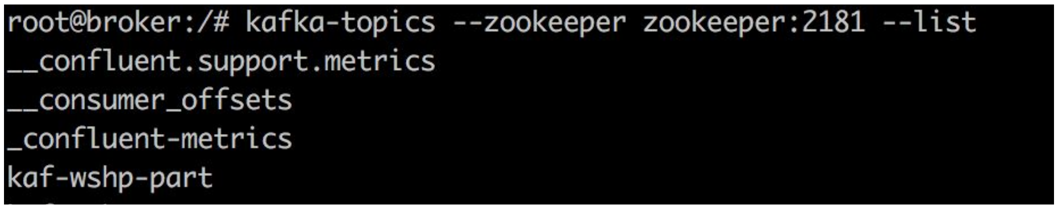
```
docker exec -it broker bash
kafka-topics --zookeeper zookeeper:2181\
 --create\
 --topic kaf-wshp-part\
 --partitions 3\
 --replication-factor 1
```



```
root@broker:/# cd /
root@broker:/# kafka-topics --zookeeper zookeeper:2181 --create --topic kaf-wshp-part --partitions 3 --replication-factor 1
Created topic kaf-wshp-part.
root@broker:/#
```

Check if the topic is listed.

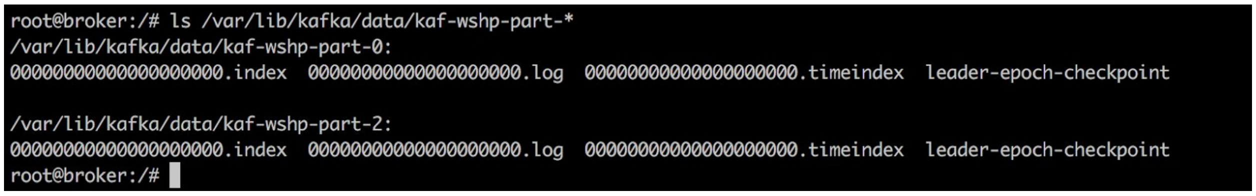
```
kafka-topics --zookeeper zookeeper:2181 --list
```



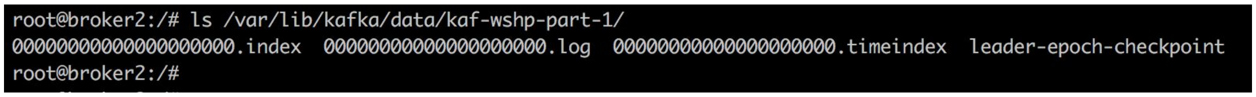
```
root@broker:/# kafka-topics --zookeeper zookeeper:2181 --list
__confluent.support.metrics
__consumer_offsets
_confluent-metrics
kaf-wshp-part
```

Step 2.2: Check the partition files in brokers

ls /var/lib/kafka/data (log.dirs in server.properties file)



```
root@broker:/# ls /var/lib/kafka/data/kaf-wshp-part-*
/var/lib/kafka/data/kaf-wshp-part-0:
00000000000000000000.index  00000000000000000000.log  00000000000000000000.timeindex  leader-epoch-checkpoint

/var/lib/kafka/data/kaf-wshp-part-2:
00000000000000000000.index  00000000000000000000.log  00000000000000000000.timeindex  leader-epoch-checkpoint
root@broker:/#
```

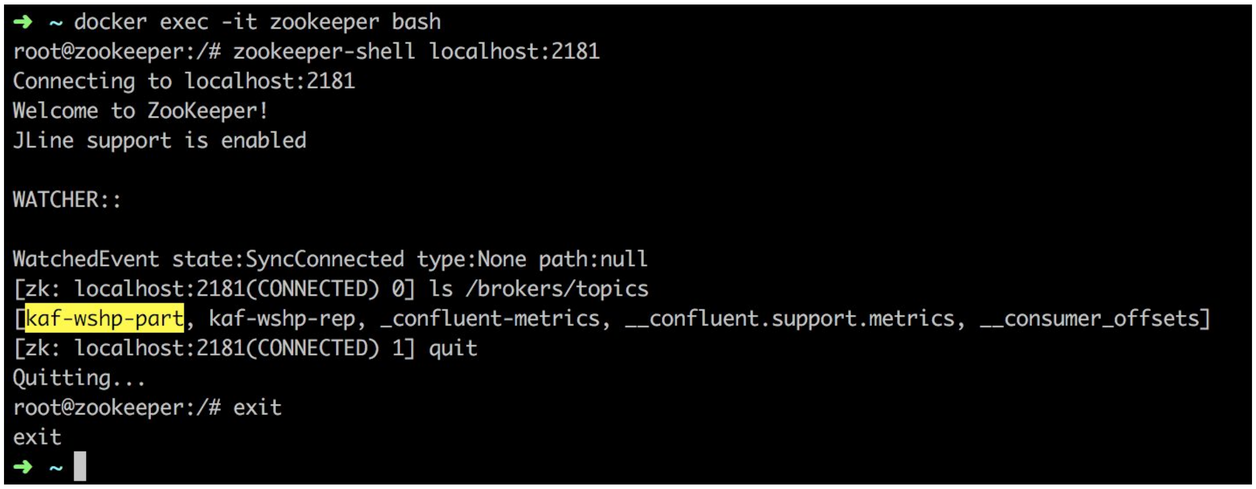


```
root@broker2:/# ls /var/lib/kafka/data/kaf-wshp-part-1/
00000000000000000000.index  00000000000000000000.log  00000000000000000000.timeindex  leader-epoch-checkpoint
root@broker2:/#
```

you can see the partitions created for that topic. All partitions need not be on the same broker.

Step 2.3: Check the topic listed in zookeeper

docker exec -it zookeeper bash
zookeeper-shell localhost:2181
ls /brokers/topics

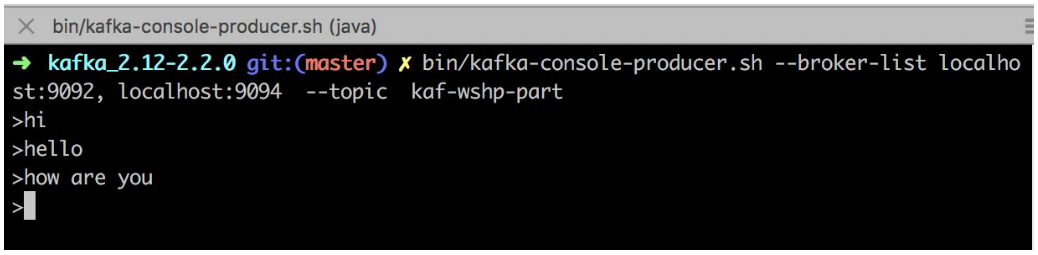


```
➜ ~ docker exec -it zookeeper bash
root@zookeeper:/# zookeeper-shell localhost:2181
Connecting to localhost:2181
Welcome to ZooKeeper!
JLine support is enabled

WATCHER::

WatchedEvent state:SyncConnected type:None path:null
[zk: localhost:2181(CONNECTED) 0] ls /brokers/topics
[kaf-wshp-part, kaf-wshp-rep, _confluent-metrics, __confluent.support.metrics, __consumer_offsets]
[zk: localhost:2181(CONNECTED) 1] quit
Quitting...
root@zookeeper:/# exit
exit
➜ ~
```

Step 3: Produce a message

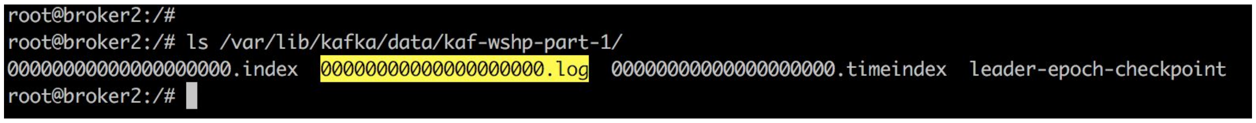
You can produce a message from anywhere as long as you give the correct broker servers list.

```
bin/kafka-console-producer.sh\
 --broker-list localhost:9092, localhost:9094\
 --topic  kaf-wshp-part
```



```
bin/kafka-console-producer.sh (java)
➜ kafka_2.12-2.2.0 git:(master) ✗ bin/kafka-console-producer.sh --broker-list localho
st:9092, localhost:9094  --topic  kaf-wshp-part
>hi
>hello
>how are you
>
```

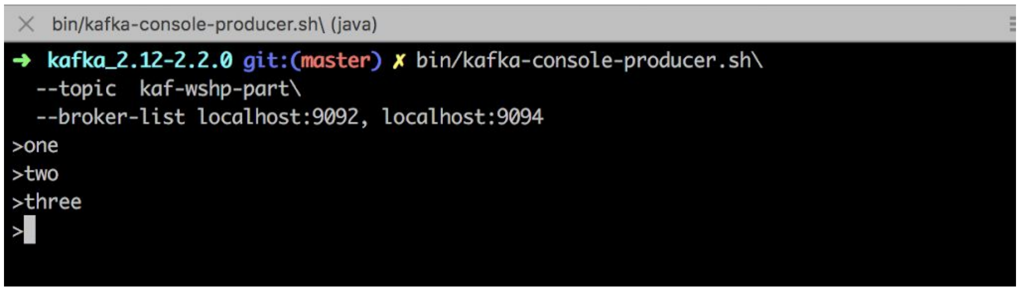
Now you can check if your message is in one of these partition log files.



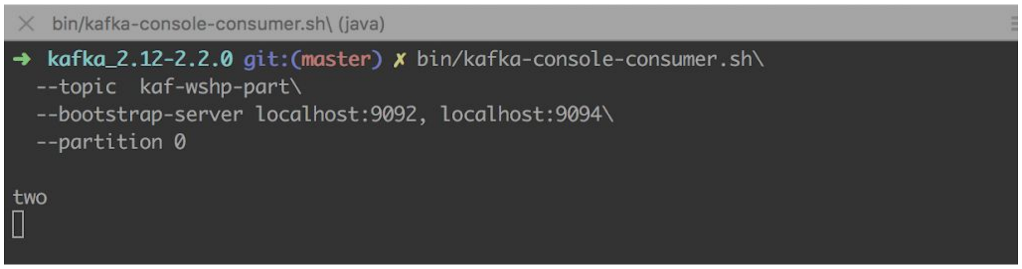
```
root@broker2:/#
root@broker2:/# ls /var/lib/kafka/data/kaf-wshp-part-1/
00000000000000000000.index  00000000000000000000.log  00000000000000000000.timeindex  leader-epoch-checkpoint
root@broker2:/#
```

Step 4: Consume from different partitions

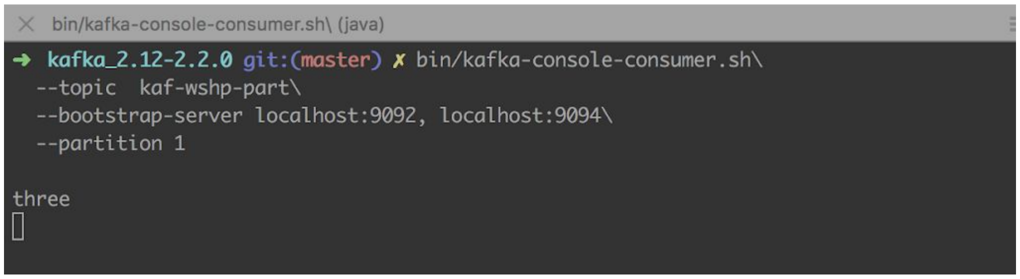
```
bin/kafka-console-consumer.sh\
  --topic  kaf-wshp-part\
  --bootstrap-server localhost:9092, localhost:9094\
  --partition <partition>
```



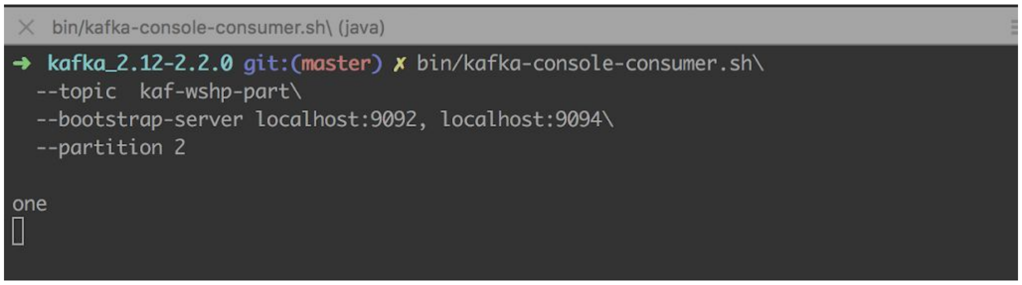
```
bin/kafka-console-producer.sh\ (java)
➜ kafka_2.12-2.2.0 git:(master) ✗ bin/kafka-console-producer.sh\
  --topic  kaf-wshp-part\
  --broker-list localhost:9092, localhost:9094
>one
>two
>three
>
```



```
bin/kafka-console-consumer.sh\ (java)
➜ kafka_2.12-2.2.0 git:(master) ✗ bin/kafka-console-consumer.sh\
  --topic  kaf-wshp-part\
  --bootstrap-server localhost:9092, localhost:9094\
  --partition 0

two
```



```
bin/kafka-console-consumer.sh\ (java)
➜ kafka_2.12-2.2.0 git:(master) ✗ bin/kafka-console-consumer.sh\
  --topic  kaf-wshp-part\
  --bootstrap-server localhost:9092, localhost:9094\
  --partition 1

three
```

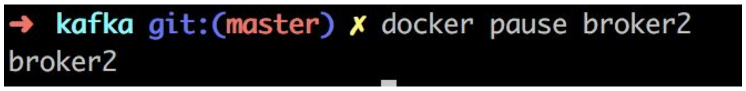


```
bin/kafka-console-consumer.sh\ (java)
➜ kafka_2.12-2.2.0 git:(master) ✗ bin/kafka-console-consumer.sh\
  --topic  kaf-wshp-part\
  --bootstrap-server localhost:9092, localhost:9094\
  --partition 2

one
```

Step 5: Bring down broker2 and observe

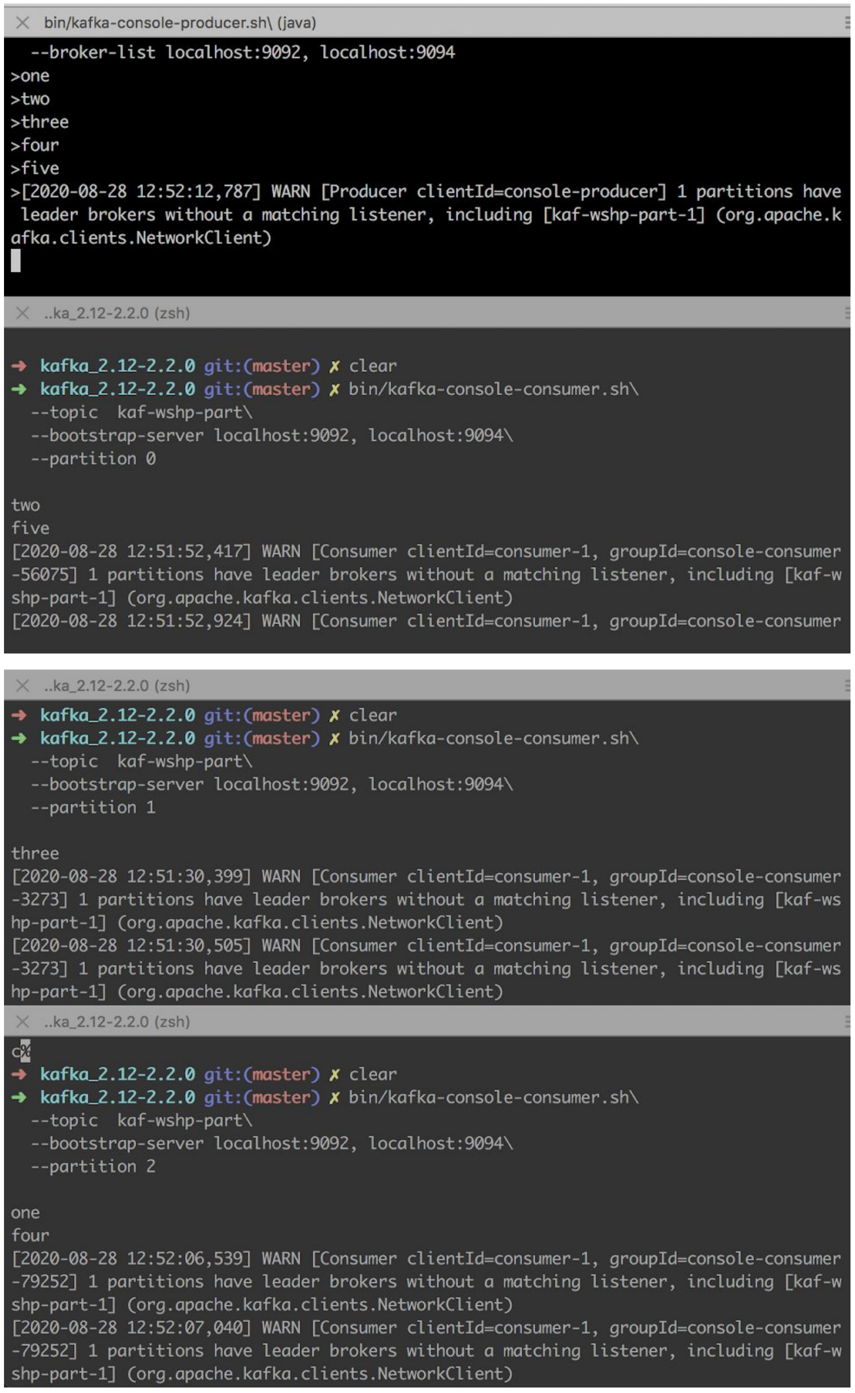
Step 5.1: Bring down broker2

docker pause broker2



```
➜  kafka git:(master) ✗ docker pause broker2
broker2
```

Step 5.2: You will observe exceptions regarding the partition connectivity.



```
bin/kafka-console-producer.sh\ (java)
  --broker-list localhost:9092, localhost:9094
>one
>two
>three
>four
>five
>[2020-08-28 12:52:12,787] WARN [Producer clientId=console-producer] 1 partitions have
 leader brokers without a matching listener, including [kaf-wshp-part-1] (org.apache.k
afka.clients.NetworkClient)
```

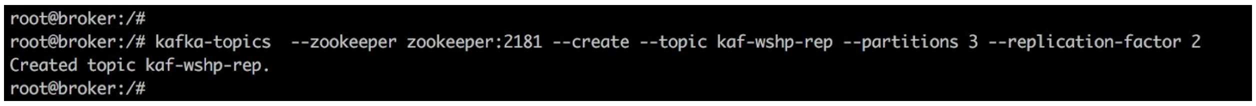
```
..ka_2.12-2.2.0 (zsh)

➜  kafka_2.12-2.2.0 git:(master) ✗ clear
➜  kafka_2.12-2.2.0 git:(master) ✗ bin/kafka-console-consumer.sh\
  --topic  kaf-wshp-part\
  --bootstrap-server localhost:9092, localhost:9094\
  --partition 0

two
five
[2020-08-28 12:51:52,417] WARN [Consumer clientId=consumer-1, groupId=console-consumer
-56075] 1 partitions have leader brokers without a matching listener, including [kaf-w
shp-part-1] (org.apache.kafka.clients.NetworkClient)
[2020-08-28 12:51:52,924] WARN [Consumer clientId=consumer-1, groupId=console-consumer
```

```
..ka_2.12-2.2.0 (zsh)
➜  kafka_2.12-2.2.0 git:(master) ✗ clear
➜  kafka_2.12-2.2.0 git:(master) ✗ bin/kafka-console-consumer.sh\
  --topic  kaf-wshp-part\
  --bootstrap-server localhost:9092, localhost:9094\
  --partition 1

three
[2020-08-28 12:51:30,399] WARN [Consumer clientId=consumer-1, groupId=console-consumer
-3273] 1 partitions have leader brokers without a matching listener, including [kaf-ws
hp-part-1] (org.apache.kafka.clients.NetworkClient)
[2020-08-28 12:51:30,505] WARN [Consumer clientId=consumer-1, groupId=console-consumer
-3273] 1 partitions have leader brokers without a matching listener, including [kaf-ws
hp-part-1] (org.apache.kafka.clients.NetworkClient)
```

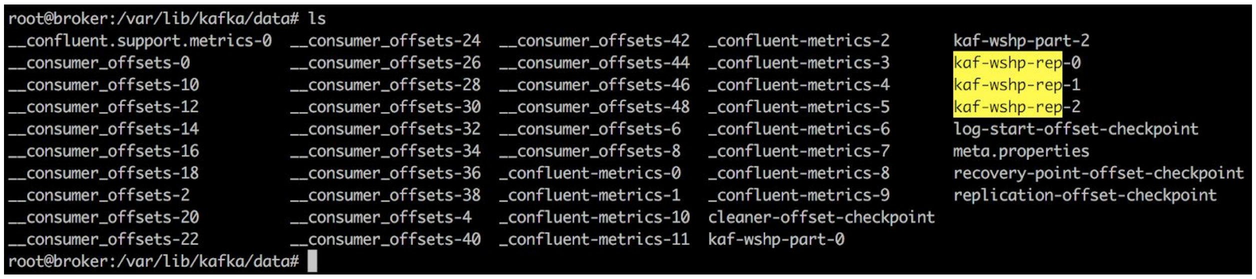
```
..ka_2.12-2.2.0 (zsh)
c%
➜  kafka_2.12-2.2.0 git:(master) ✗ clear
➜  kafka_2.12-2.2.0 git:(master) ✗ bin/kafka-console-consumer.sh\
  --topic  kaf-wshp-part\
  --bootstrap-server localhost:9092, localhost:9094\
  --partition 2

one
four
[2020-08-28 12:52:06,539] WARN [Consumer clientId=consumer-1, groupId=console-consumer
-79252] 1 partitions have leader brokers without a matching listener, including [kaf-w
shp-part-1] (org.apache.kafka.clients.NetworkClient)
[2020-08-28 12:52:07,040] WARN [Consumer clientId=consumer-1, groupId=console-consumer
-79252] 1 partitions have leader brokers without a matching listener, including [kaf-w
shp-part-1] (org.apache.kafka.clients.NetworkClient)
```

Step 6: Create topic with replication factor

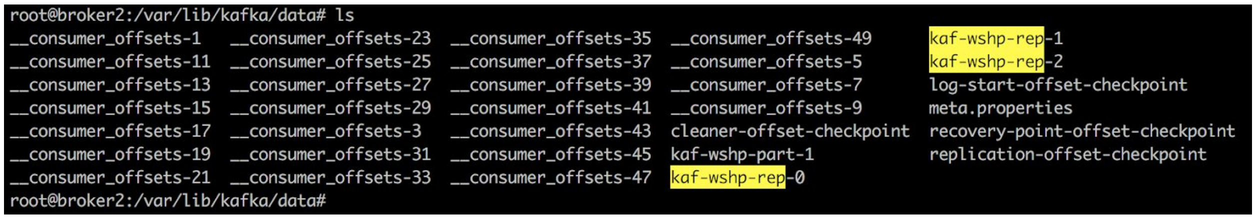
kafka-topics --zookeeper zookeeper:2181 --create --topic kaf-wshp-rep --partitions 3 --replication-factor 2



```
root@broker:/#
root@broker:/# kafka-topics  --zookeeper zookeeper:2181 --create --topic kaf-wshp-rep --partitions 3 --replication-factor 2
Created topic kaf-wshp-rep.
root@broker:/#
```



```
root@broker:/var/lib/kafka/data# ls
__confluent.support.metrics-0  __consumer_offsets-24  __consumer_offsets-42  _confluent-metrics-2      kaf-wshp-part-2
__consumer_offsets-0           __consumer_offsets-26  __consumer_offsets-44  _confluent-metrics-3      kaf-wshp-rep-0
__consumer_offsets-10          __consumer_offsets-28  __consumer_offsets-46  _confluent-metrics-4      kaf-wshp-rep-1
__consumer_offsets-12          __consumer_offsets-30  __consumer_offsets-48  _confluent-metrics-5      kaf-wshp-rep-2
__consumer_offsets-14          __consumer_offsets-32  __consumer_offsets-6   _confluent-metrics-6      log-start-offset-checkpoint
__consumer_offsets-16          __consumer_offsets-34  __consumer_offsets-8   _confluent-metrics-7      meta.properties
__consumer_offsets-18          __consumer_offsets-36  _confluent-metrics-0   _confluent-metrics-8      recovery-point-offset-checkpoint
__consumer_offsets-2           __consumer_offsets-38  _confluent-metrics-1   _confluent-metrics-9      replication-offset-checkpoint
__consumer_offsets-20          __consumer_offsets-4   _confluent-metrics-10  cleaner-offset-checkpoint
__consumer_offsets-22          __consumer_offsets-40  _confluent-metrics-11  kaf-wshp-part-0
root@broker:/var/lib/kafka/data#
```



```
root@broker2:/var/lib/kafka/data# ls
__consumer_offsets-1   __consumer_offsets-23  __consumer_offsets-35  __consumer_offsets-49      kaf-wshp-rep-1
__consumer_offsets-11  __consumer_offsets-25  __consumer_offsets-37  __consumer_offsets-5       kaf-wshp-rep-2
__consumer_offsets-13  __consumer_offsets-27  __consumer_offsets-39  __consumer_offsets-7       log-start-offset-checkpoint
__consumer_offsets-15  __consumer_offsets-29  __consumer_offsets-41  __consumer_offsets-9       meta.properties
__consumer_offsets-17  __consumer_offsets-3   __consumer_offsets-43  cleaner-offset-checkpoint  recovery-point-offset-checkpoint
__consumer_offsets-19  __consumer_offsets-31  __consumer_offsets-45  kaf-wshp-part-1            replication-offset-checkpoint
__consumer_offsets-21  __consumer_offsets-33  __consumer_offsets-47  kaf-wshp-rep-0
root@broker2:/var/lib/kafka/data#
```

Repeat the above step for kaf-wshp-rep topic

Start console producer and produce messages

```
bin/kafka-console-producer.sh\
 --broker-list localhost:9092, localhost:9094\
 --topic  kaf-wshp-part
```

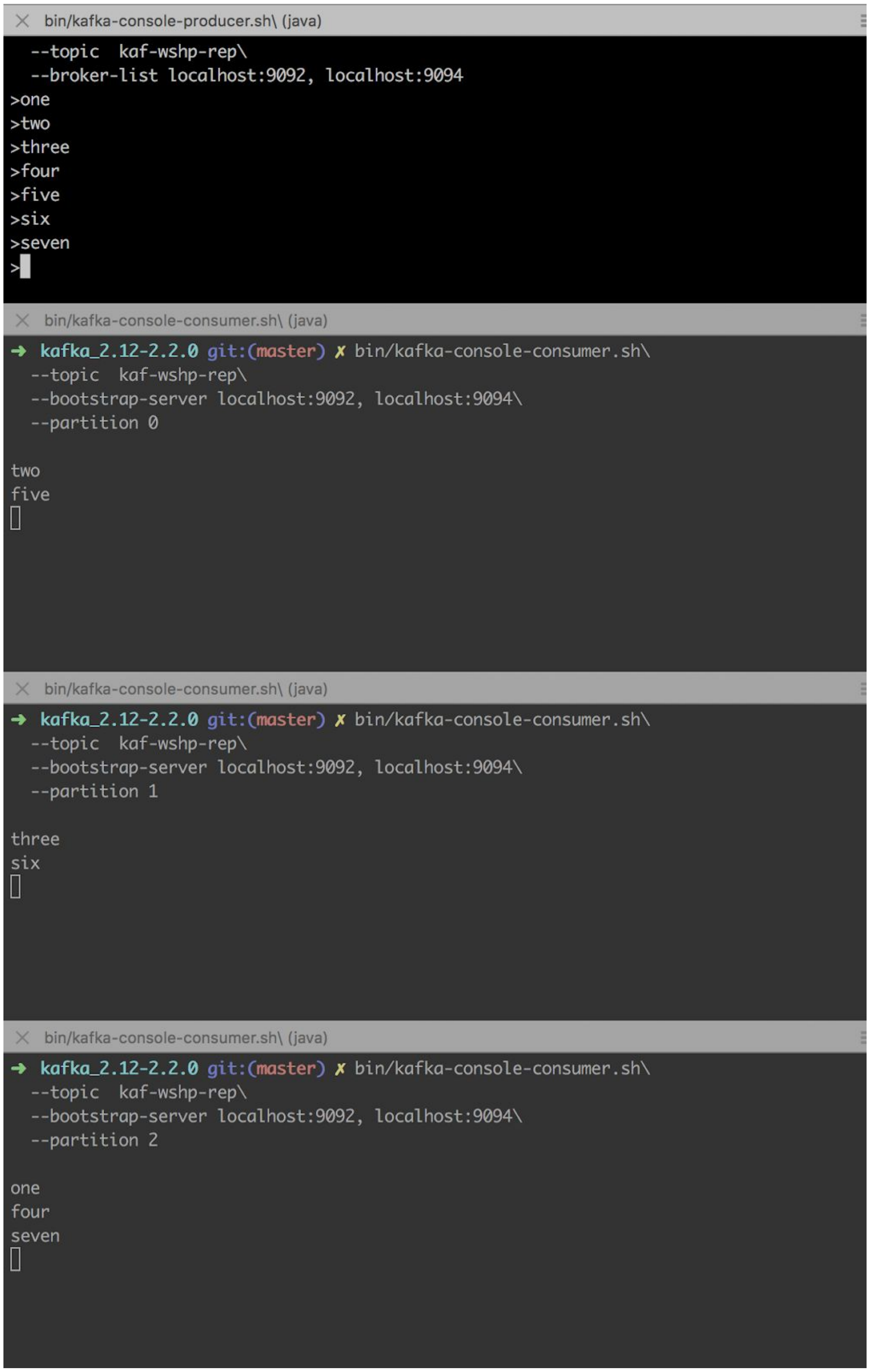
Start console consumer and consume messages from different partitions

```
bin/kafka-console-consumer.sh\
  --topic  kaf-wshp-part\
  --bootstrap-server localhost:9092, localhost:9094\
  --partition <partition>
```

Bring down broker2

```
docker pause broker2
```

Now produce messages and see that the messages are still consumed properly without any exception.



```
bin/kafka-console-producer.sh\ (java)
  --topic  kaf-wshp-rep\
  --broker-list localhost:9092, localhost:9094
>one
>two
>three
>four
>five
>six
>seven
>
```

```
bin/kafka-console-consumer.sh\ (java)
➜ kafka_2.12-2.2.0 git:(master) ✗ bin/kafka-console-consumer.sh\
  --topic  kaf-wshp-rep\
  --bootstrap-server localhost:9092, localhost:9094\
  --partition 0

two
five
```

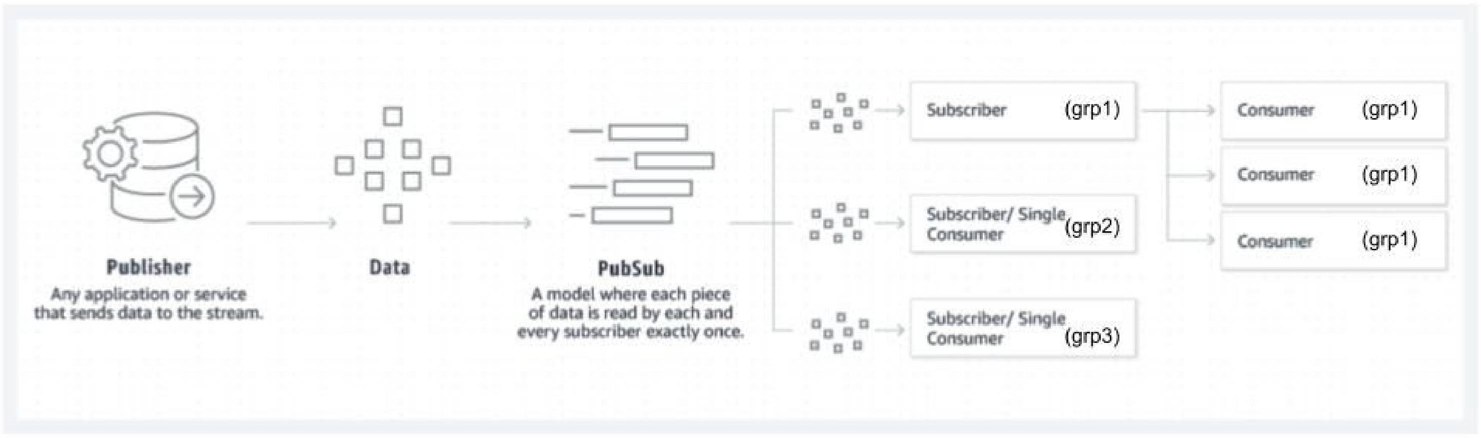
```
bin/kafka-console-consumer.sh\ (java)
➜ kafka_2.12-2.2.0 git:(master) ✗ bin/kafka-console-consumer.sh\
  --topic  kaf-wshp-rep\
  --bootstrap-server localhost:9092, localhost:9094\
  --partition 1

three
six
```

```
bin/kafka-console-consumer.sh\ (java)
➜ kafka_2.12-2.2.0 git:(master) ✗ bin/kafka-console-consumer.sh\
  --topic  kaf-wshp-rep\
  --bootstrap-server localhost:9092, localhost:9094\
  --partition 2

one
four
seven
```

Step 7: Consume the message from different subscribers.

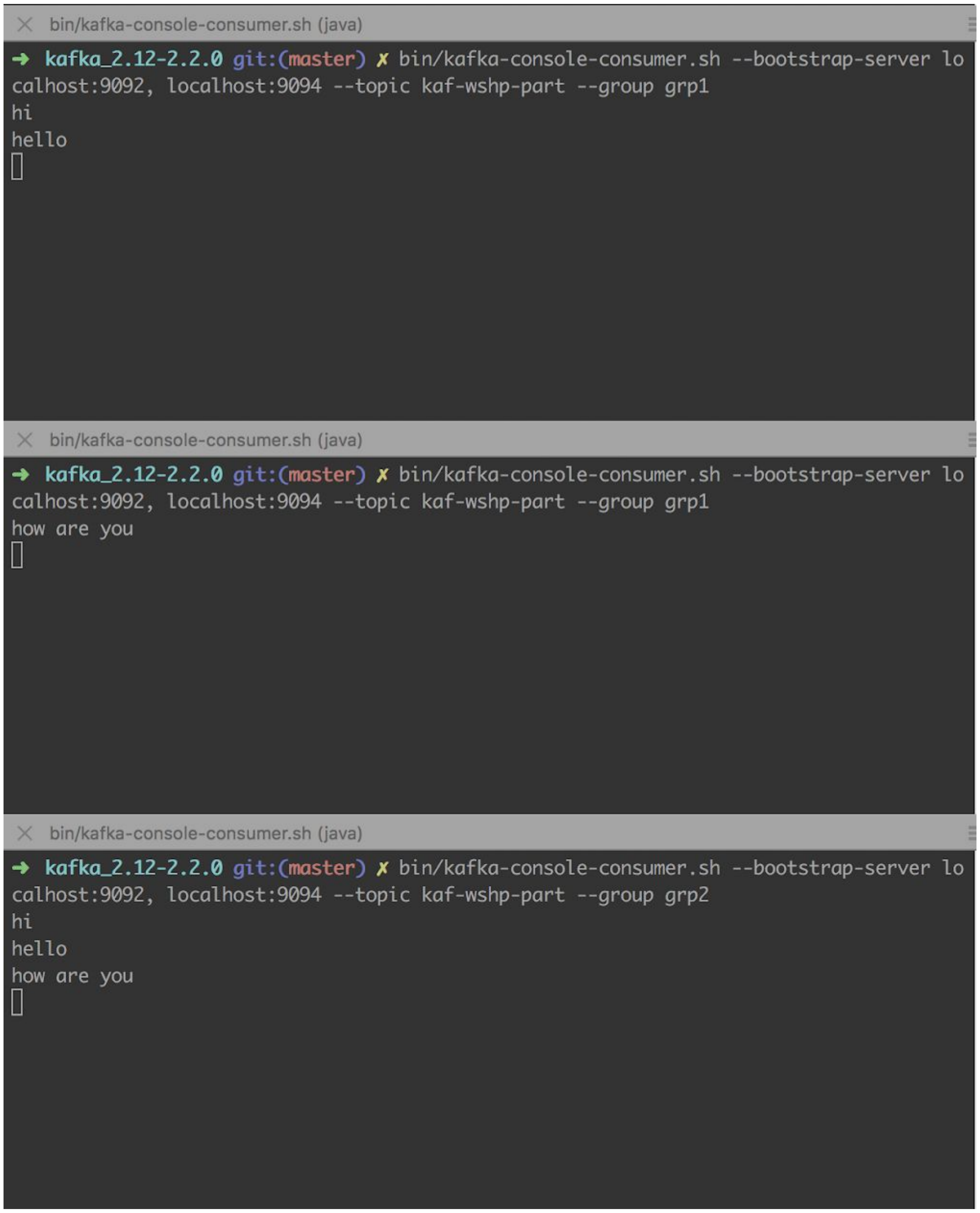
Kafka allows multiple subscriptions using consumer groups. Each group id maps to a subscription. Multiple consumers can use the same group id to form into a consumer group. This allows effective processing of messages parallelly.

Each consumer in a consumer group listens to one or more partitions dedicatedly.
No of consumers is proportional to no of partitions.
More no of partitions , more no of consumers.
More no of consumer groups, More no of subscribers


Publisher
Any application or service that sends data to the stream.
Data
PubSub
A model where each piece of data is read by each and every subscriber exactly once.
Subscriber (grp1)
Subscriber/ Single Consumer (grp2)
Subscriber/ Single Consumer (grp3)
Consumer (grp1)
Consumer (grp1)
Consumer (grp1)

You can consume a message from anywhere as long as you give the correct broker servers list.

```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092, localhost:9094 --topic
kaf-wshp-part --group <group>
```



```
bin/kafka-console-consumer.sh (java)
➜ kafka_2.12-2.2.0 git:(master) ✗ bin/kafka-console-consumer.sh --bootstrap-server lo
calhost:9092, localhost:9094 --topic kaf-wshp-part --group grp1
hi
hello
```

```
bin/kafka-console-consumer.sh (java)
➜ kafka_2.12-2.2.0 git:(master) ✗ bin/kafka-console-consumer.sh --bootstrap-server lo
calhost:9092, localhost:9094 --topic kaf-wshp-part --group grp1
how are you
```

```
bin/kafka-console-consumer.sh (java)
➜ kafka_2.12-2.2.0 git:(master) ✗ bin/kafka-console-consumer.sh --bootstrap-server lo
calhost:9092, localhost:9094 --topic kaf-wshp-part --group grp2
hi
hello
how are you
```

From the above example, we can see that one copy of messages are distributed among consumers in the same consumer group grp1(subscription with multiple consumers) and a copy of the same messages are sent to another consumer group grp2 (subscription with single consumer).

Exercise:

Yet order of messages is not guaranteed in kafka with partitions. If you want to maintain order to a certain level you can configure key based partition instead of default round robin fashion.

So all messages produced from a producer with the same key will go to the same partition where order can be maintained.

```
kafka-console-producer --topic example-topic --broker-list broker:9092\
  --property parse.key=true\
  --property key.separator=":"
```

Notes:

1. Brokers have one of them as a leader given the distributed nature.
2. Topic partitions are distributed among brokers. So these partitions have one of the brokers where partitions are created as leader for that topic.