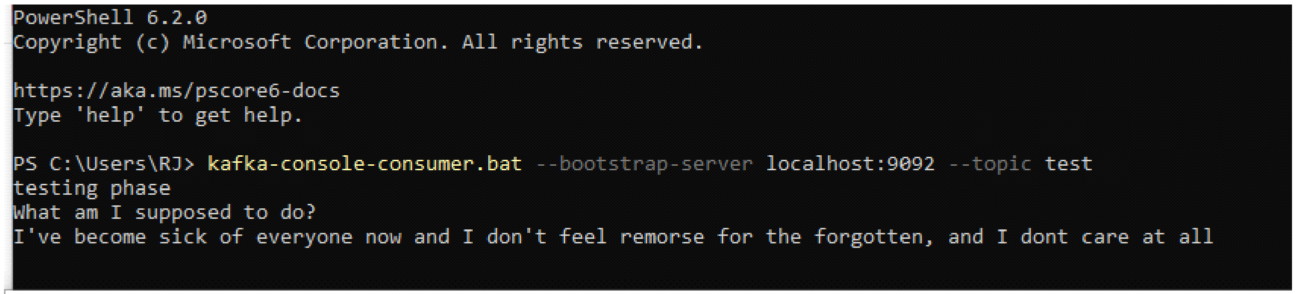
Initials

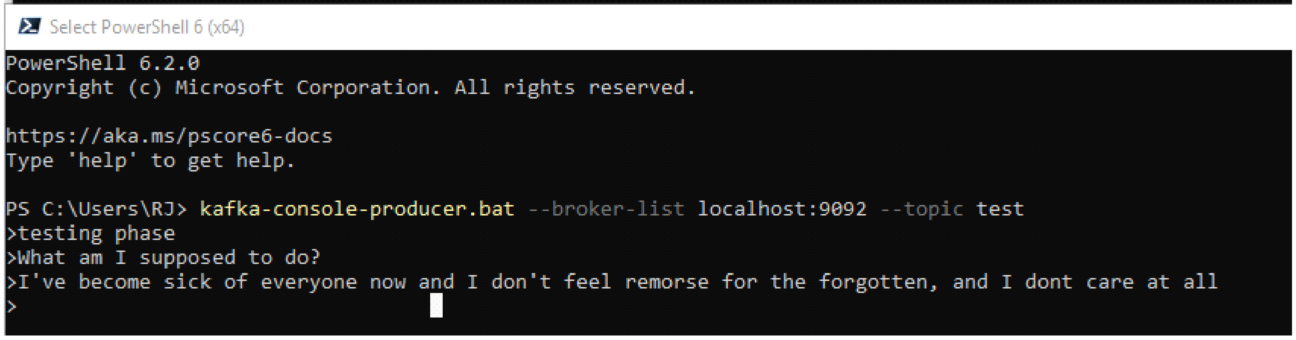
Monday, April 1, 2019 10:49 PM

1. **Zkserver**
2. Cd C:\kafka_2.12-2.2.0\config
   Kafka-server-start.bat server.properties
3. **kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test**

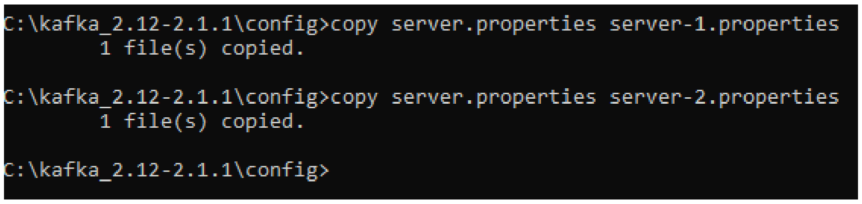
4. **kafka-topics.bat --list --zookeeper localhost:2181**



```
PowerShell 6.2.0
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/pscore6-docs
Type 'help' to get help.

PS C:\Users\RJ> kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic test
testing phase
What am I supposed to do?
I've become sick of everyone now and I don't feel remorse for the forgotten, and I dont care at all
```



```
PowerShell 6.2.0
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/pscore6-docs
Type 'help' to get help.

PS C:\Users\RJ> kafka-console-producer.bat --broker-list localhost:9092 --topic test
>testing phase
>What am I supposed to do?
>I've become sick of everyone now and I don't feel remorse for the forgotten, and I dont care at all
>
```

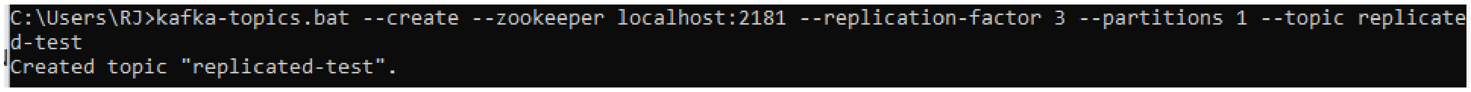


```
C:\kafka_2.12-2.1.1\config>copy server.properties server-1.properties
        1 file(s) copied.

C:\kafka_2.12-2.1.1\config>copy server.properties server-2.properties
        1 file(s) copied.

C:\kafka_2.12-2.1.1\config>
```

```
config/server-1.properties:
    broker.id=1
    listeners=PLAINTEXT://:9093
    log.dirs=/tmp/kafka-logs-1

config/server-2.properties:
    broker.id=2
    listeners=PLAINTEXT://:9094
    log.dirs=/tmp/kafka-logs-2
```

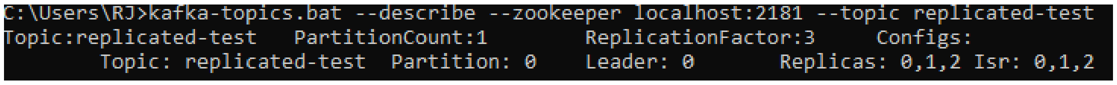
```
bin/kafka-server-start.sh config/server-1.properties &
..
bin/kafka-server-start.sh config/server-2.properties &
..
```

eate a new topic with a replication factor of three:



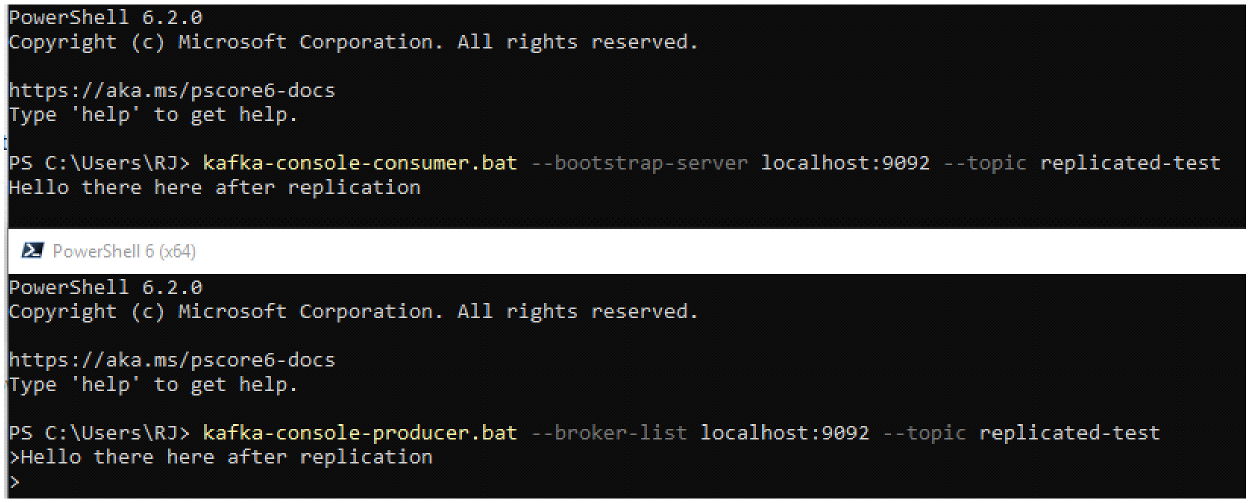
```
C:\Users\RJ>kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 3 --partitions 1 --topic replicate
d-test
Created topic "replicated-test".
```

TO DESCRIBE WHAT THE TOPIC ARE DISTRO INTO


```
C:\Users\RJ>kafka-topics.bat --describe --zookeeper localhost:2181 --topic replicated-test
Topic:replicated-test   PartitionCount:1        ReplicationFactor:3     Configs:
        Topic: replicated-test  Partition: 0    Leader: 0       Replicas: 0,1,2 Isr: 0,1,2
```

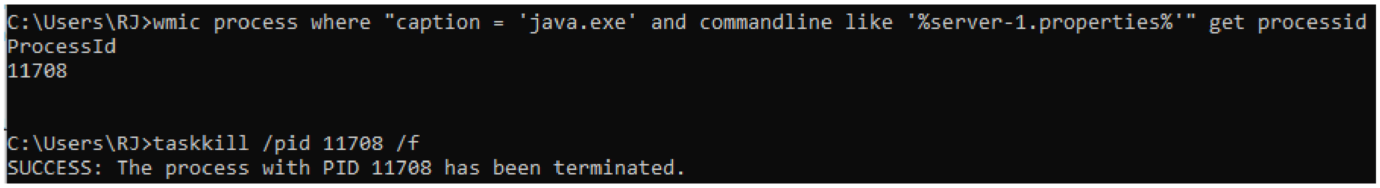
The first line gives a summary of all the partitions, each additional line gives information about one partition. Since we have only one partition for this topic there is only one line.

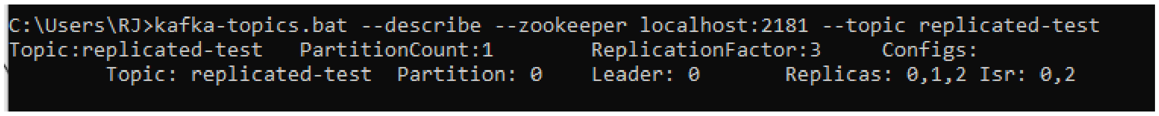
- "leader" is the node responsible for all reads and writes for the given partition. Each node will be the leader for a randomly selected portion of the partitions.
- "replicas" is the list of nodes that replicate the log for this partition regardless of whether they are the leader or even if they are currently alive.
- "isr" is the set of "in-sync" replicas. This is the subset of the replicas list that is currently alive and caught-up to the leader.nr
- Each partition has one server which acts as the "leader" and zero or more servers which act as "followers". The leader handles all read and write requests for the partition while the followers passively replicate the leader. If the leader fails, one of the followers will automatically become the new leader. Each server acts as a leader for some of its partitions and a follower for others so load is well balanced within the cluster.


```
PowerShell 6.2.0
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/pscore6-docs
Type 'help' to get help.

PS C:\Users\RJ> kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic replicated-test
Hello there here after replication
```

```
PowerShell 6.2.0
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/pscore6-docs
Type 'help' to get help.

PS C:\Users\RJ> kafka-console-producer.bat --broker-list localhost:9092 --topic replicated-test
>Hello there here after replication
>
```

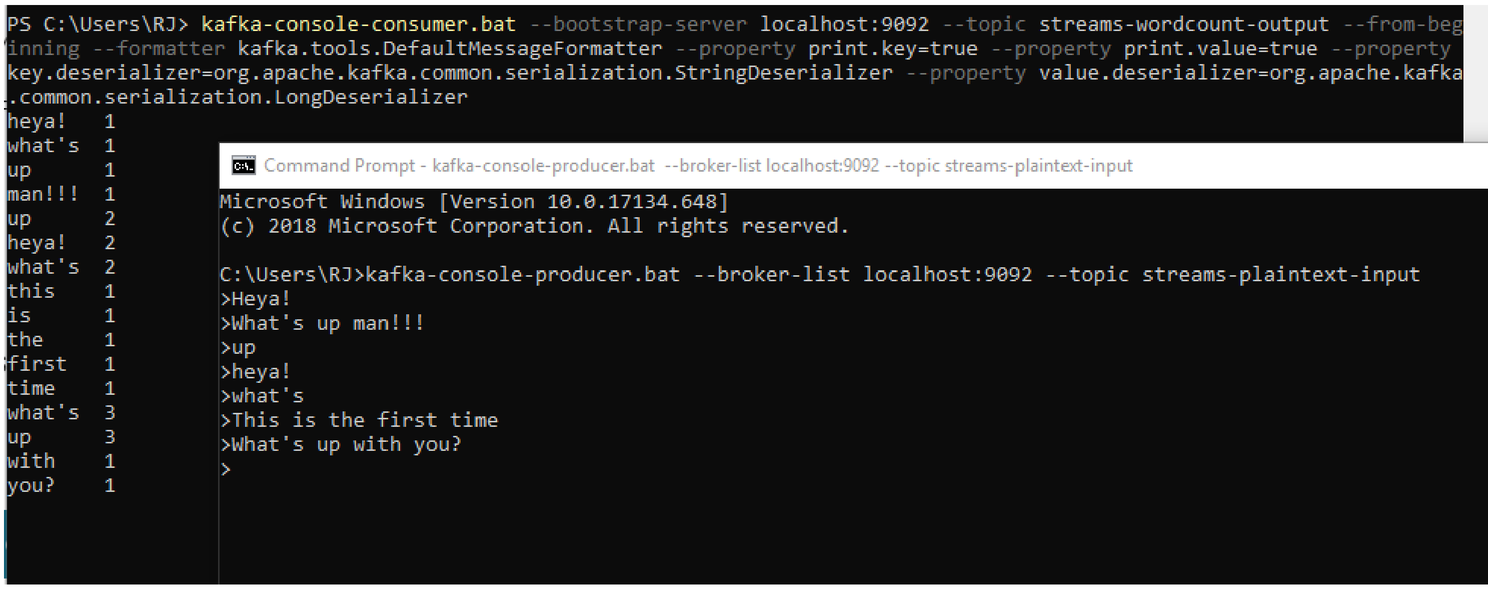
Killing the leader node


```
C:\Users\RJ>wmic process where "caption = 'java.exe' and commandline like '%server-1.properties%'" get processid
ProcessId
11708


C:\Users\RJ>taskkill /pid 11708 /f
SUCCESS: The process with PID 11708 has been terminated.
```


```
C:\Users\RJ>kafka-topics.bat --describe --zookeeper localhost:2181 --topic replicated-test
Topic:replicated-test   PartitionCount:1        ReplicationFactor:3     Configs:
        Topic: replicated-test  Partition: 0    Leader: 0       Replicas: 0,1,2 Isr: 0,2
```

LEADERSHIP CHANGED AFTER THE LEADER NODE IS TERMINATED HENCE, FAULT TOLERANCE

- **WordCount Program**


```
PS C:\Users\RJ> kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic streams-wordcount-output --from-beg
inning --formatter kafka.tools.DefaultMessageFormatter --property print.key=true --property print.value=true --property
key.deserializer=org.apache.kafka.common.serialization.StringDeserializer --property value.deserializer=org.apache.kafka
.common.serialization.LongDeserializer
heya!   1
what's  1
up      1
man!!!  1
up      2
heya!   2
what's  2
this    1
is      1
the     1
first   1
time    1
what's  3
up      3
with    1
you?    1
```

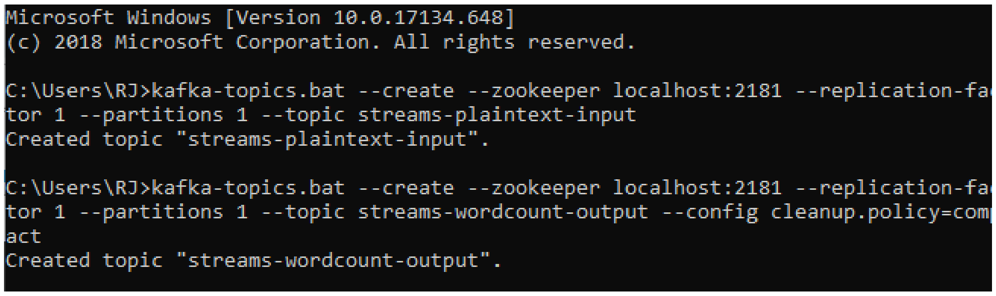
Command Prompt - kafka-console-producer.bat --broker-list localhost:9092 --topic streams-plaintext-input

```
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\RJ>kafka-console-producer.bat --broker-list localhost:9092 --topic streams-plaintext-input
>Heya!
>What's up man!!!
>up
>heya!
>what's
>This is the first time
>What's up with you?
>
```

Create topic streams-plaintext-input and streams-wordcount-output

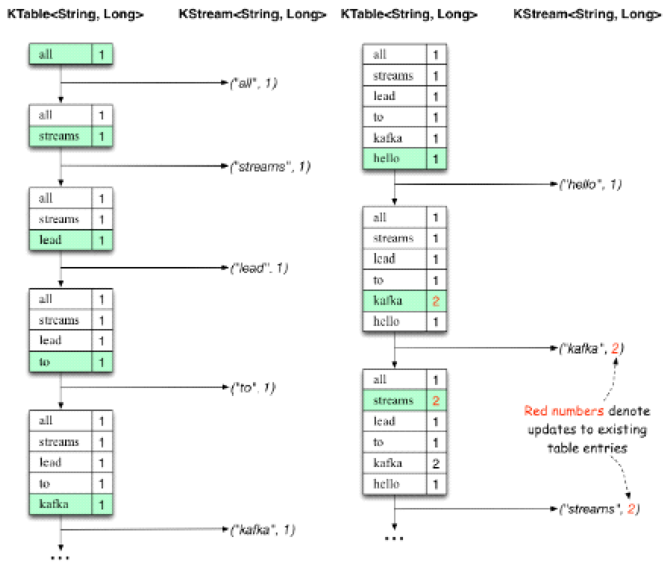
kafka-console-producer.bat --broker-list localhost:9092 --topic streams-plaintext-input

kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic streams-wordcount-output --from-beginning --formatter kafka.tools.DefaultMessageFormatter --property print.key=true --property print.value=true --property key.deserializer=org.apache.kafka.common.serialization.StringDeserializer --property value.deserializer=org.apache.kafka.common.serialization.LongDeserializer


```
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\RJ>kafka-topics.bat --create --zookeeper localhost:2181 --replication-fa
tor 1 --partitions 1 --topic streams-plaintext-input
Created topic "streams-plaintext-input".

C:\Users\RJ>kafka-topics.bat --create --zookeeper localhost:2181 --replication-fa
tor 1 --partitions 1 --topic streams-wordcount-output --config cleanup.policy=com
act
Created topic "streams-wordcount-output".
```

- **Here, the first column is the Kafka message key in java.lang.String format and represents a word that is being counted, and the second column is the message value in java.lang.Longformat, representing the word's latest count.**

The two diagrams below illustrate what is essentially happening behind the scenes. The first column shows the evolution of the current state of the `KTable<String, Long>` that is counting word occurrences for `count`. The second column shows the change records that result from state updates to the KTable and that are being sent to the output Kafka topic **streams-wordcount-output**.



First the text line "all streams lead to kafka" is being processed. The `KTable` is being built up as each new word results in a new table entry (highlighted with a green background), and a corresponding change record is sent to the downstream `KStream`.

When the second text line "hello kafka streams" is processed, we observe, for the first time, that existing entries in the `KTable` are being updated (here: for the words "kafka" and for "streams"). And again, change records are being sent to the output topic.

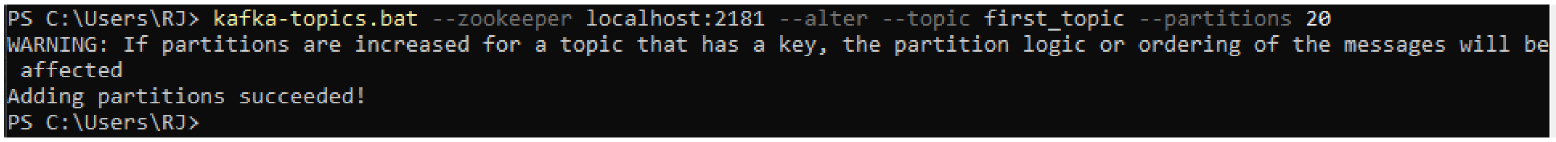
And so on (we skip the illustration of how the third line is being processed). This explains why the output topic has the contents we showed above, because it contains the full record of changes.

Looking beyond the scope of this concrete example, what Kafka Streams is doing here is to leverage the duality between a table and a changelog stream (here: table = the KTable, changelog stream = the downstream KStream): you can publish every change of the table to a stream, and if you consume the entire changelog stream from beginning to end, you can reconstruct the contents of the table.

Initials

Thursday, April 4, 2019 11:21 AM

1. **Modifying partitions**



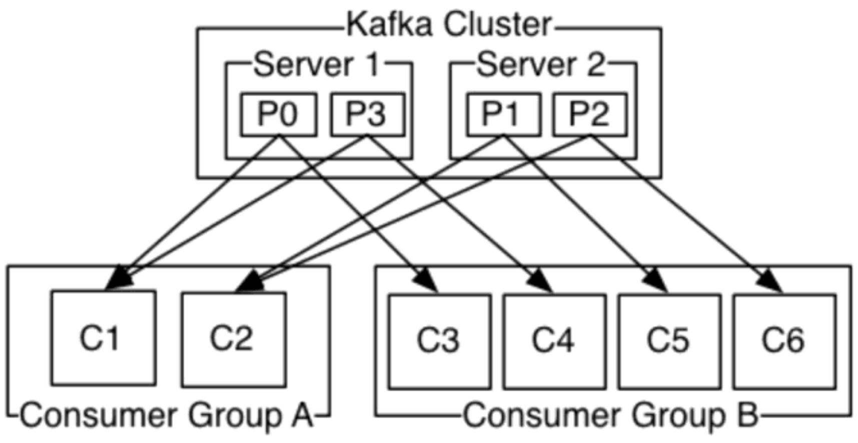
```
PS C:\Users\RJ> kafka-topics.bat --zookeeper localhost:2181 --alter --topic first_topic --partitions 20
WARNING: If partitions are increased for a topic that has a key, the partition logic or ordering of the messages will be
 affected
Adding partitions succeeded!
PS C:\Users\RJ>
```

Keynote

04 April 2019 08:09

Geo-Replication

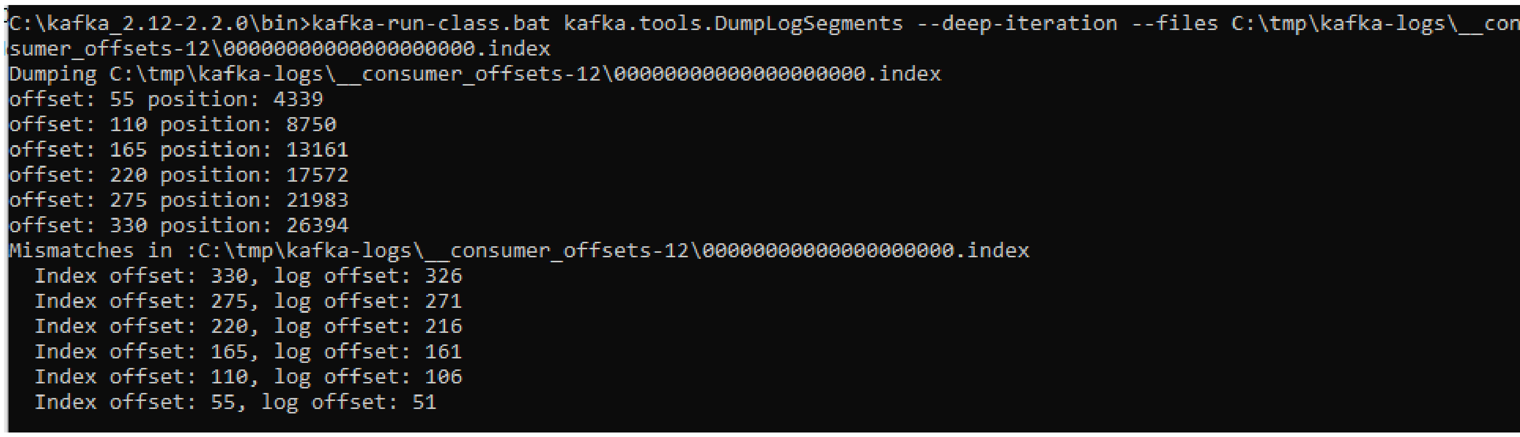
Kafka MirrorMaker provides geo-replication support for your clusters. With MirrorMaker, messages are replicated across multiple datacenters or cloud regions. You can use this in active/passive scenarios for backup and recovery; or in active/active scenarios to place data closer to your users, or support data locality requirements.



A two server Kafka cluster hosting four partitions (P0-P3) with two consumer groups. Consumer group A has two consumer instances and group B has four.

Extras

Sunday, April 28, 2019 3:58 PM

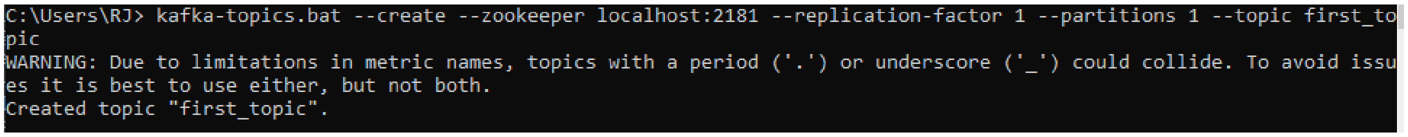


```
C:\kafka_2.12-2.2.0\bin>kafka-run-class.bat kafka.tools.DumpLogSegments --deep-iteration --files C:\tmp\kafka-logs\__con
sumer_offsets-12\00000000000000000000.index
Dumping C:\tmp\kafka-logs\__consumer_offsets-12\00000000000000000000.index
offset: 55 position: 4339
offset: 110 position: 8750
offset: 165 position: 13161
offset: 220 position: 17572
offset: 275 position: 21983
offset: 330 position: 26394
Mismatches in :C:\tmp\kafka-logs\__consumer_offsets-12\00000000000000000000.index
  Index offset: 330, log offset: 326
  Index offset: 275, log offset: 271
  Index offset: 220, log offset: 216
  Index offset: 165, log offset: 161
  Index offset: 110, log offset: 106
  Index offset: 55, log offset: 51
```

Lab 1

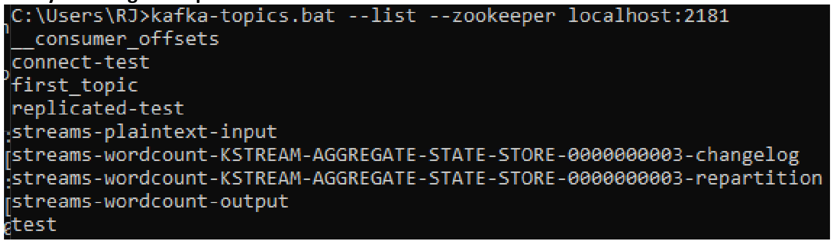
Wednesday, April 3, 2019   10:10 PM

1. **kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic first_topic**



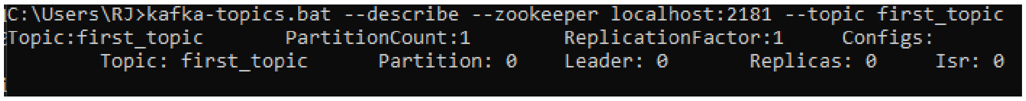
```
C:\Users\RJ> kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic first_to
pic
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issu
es it is best to use either, but not both.
Created topic "first_topic".
```

2. **Verify or listing the topics created**



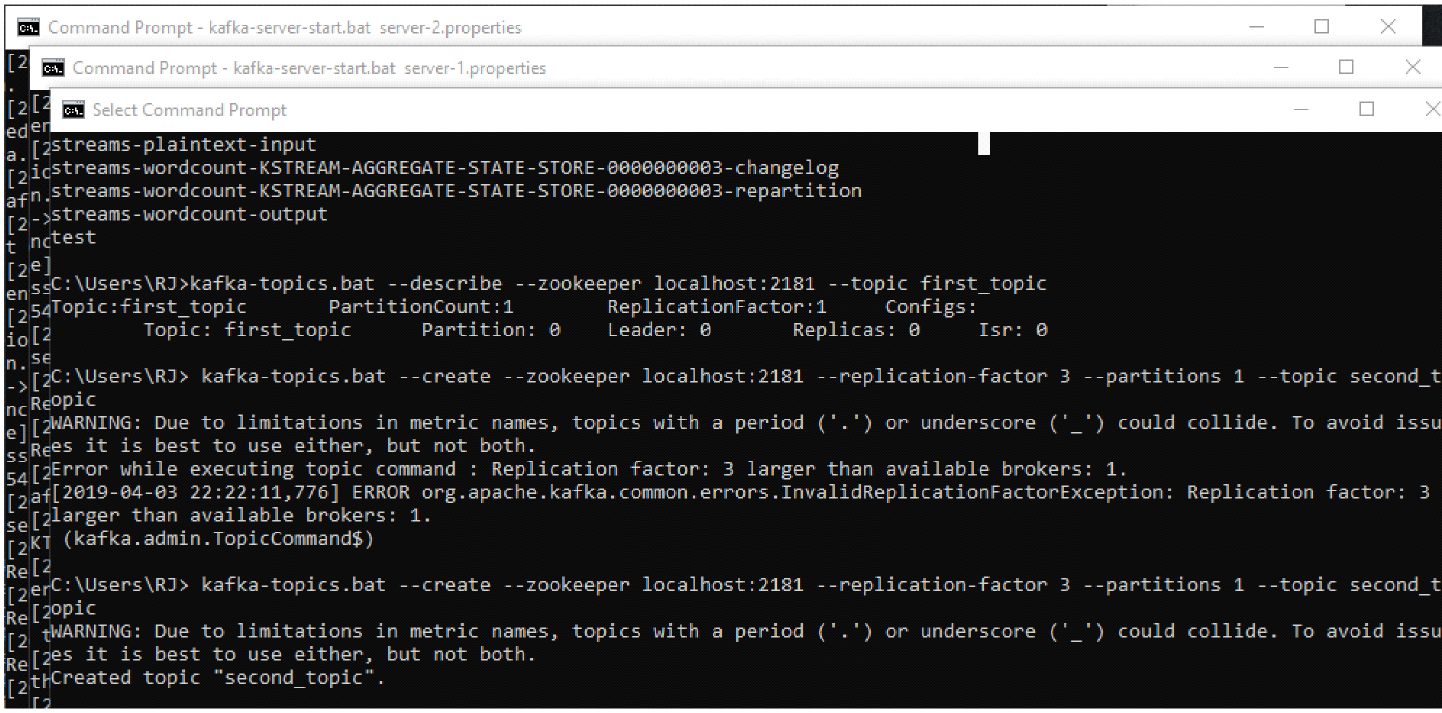
```
C:\Users\RJ>kafka-topics.bat --list --zookeeper localhost:2181
__consumer_offsets
connect-test
first_topic
replicated-test
streams-plaintext-input
streams-wordcount-KSTREAM-AGGREGATE-STATE-STORE-0000000003-changelog
streams-wordcount-KSTREAM-AGGREGATE-STATE-STORE-0000000003-repartition
streams-wordcount-output
test
```

3.



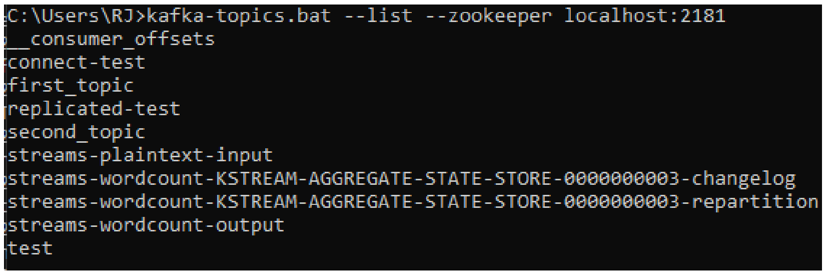
```
C:\Users\RJ>kafka-topics.bat --describe --zookeeper localhost:2181 --topic first_topic
Topic:first_topic       PartitionCount:1        ReplicationFactor:1     Configs:
        Topic: first_topic      Partition: 0    Leader: 0       Replicas: 0     Isr: 0
```

4.


Command Prompt - kafka-server-start.bat server-2.properties
Command Prompt - kafka-server-start.bat server-1.properties
Select Command Prompt

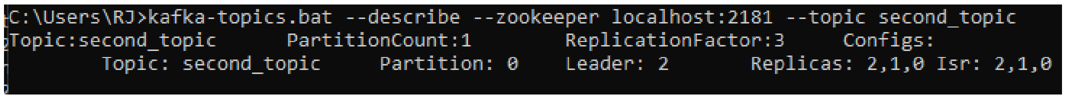
```
streams-plaintext-input
streams-wordcount-KSTREAM-AGGREGATE-STATE-STORE-0000000003-changelog
streams-wordcount-KSTREAM-AGGREGATE-STATE-STORE-0000000003-repartition
streams-wordcount-output
test

C:\Users\RJ>kafka-topics.bat --describe --zookeeper localhost:2181 --topic first_topic
Topic:first_topic       PartitionCount:1        ReplicationFactor:1     Configs:
        Topic: first_topic      Partition: 0    Leader: 0       Replicas: 0     Isr: 0

C:\Users\RJ> kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 3 --partitions 1 --topic second_t
opic
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issu
es it is best to use either, but not both.
Error while executing topic command : Replication factor: 3 larger than available brokers: 1.
[2019-04-03 22:22:11,776] ERROR org.apache.kafka.common.errors.InvalidReplicationFactorException: Replication factor: 3
larger than available brokers: 1.
 (kafka.admin.TopicCommand$)

C:\Users\RJ> kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 3 --partitions 1 --topic second_t
opic
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issu
es it is best to use either, but not both.
Created topic "second_topic".
```

5.



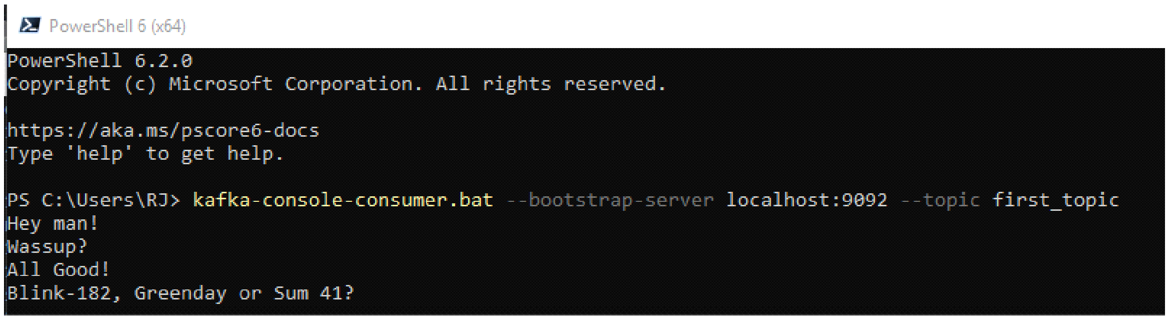
```
C:\Users\RJ>kafka-topics.bat --list --zookeeper localhost:2181
__consumer_offsets
connect-test
first_topic
replicated-test
second_topic
streams-plaintext-input
streams-wordcount-KSTREAM-AGGREGATE-STATE-STORE-0000000003-changelog
streams-wordcount-KSTREAM-AGGREGATE-STATE-STORE-0000000003-repartition
streams-wordcount-output
test
```

6.

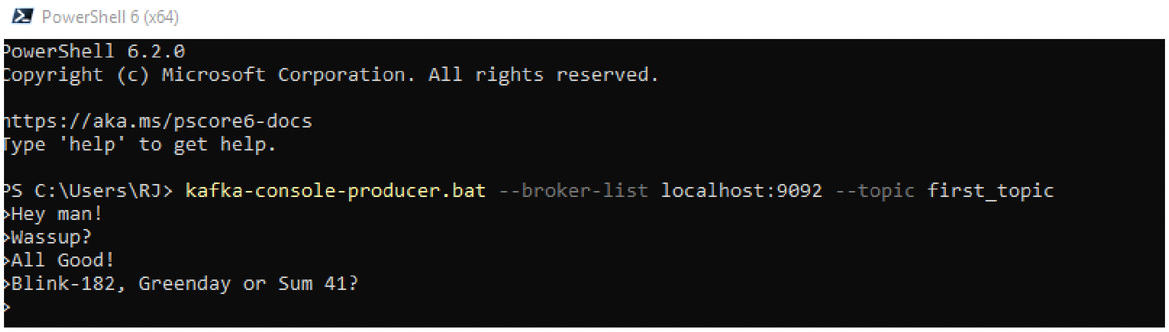


```
C:\Users\RJ>kafka-topics.bat --describe --zookeeper localhost:2181 --topic second_topic
Topic:second_topic      PartitionCount:1        ReplicationFactor:3     Configs:
        Topic: second_topic     Partition: 0    Leader: 2       Replicas: 2,1,0 Isr: 2,1,0
```

7,8.

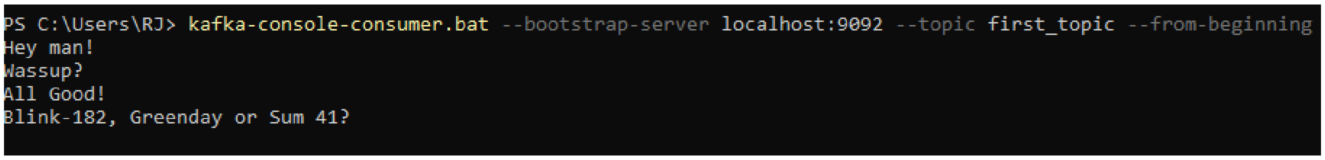


```
PowerShell 6.2.0
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/pscore6-docs
Type 'help' to get help.

PS C:\Users\RJ> kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic first_topic
Hey man!
Wassup?
All Good!
Blink-182, Greenday or Sum 41?
```



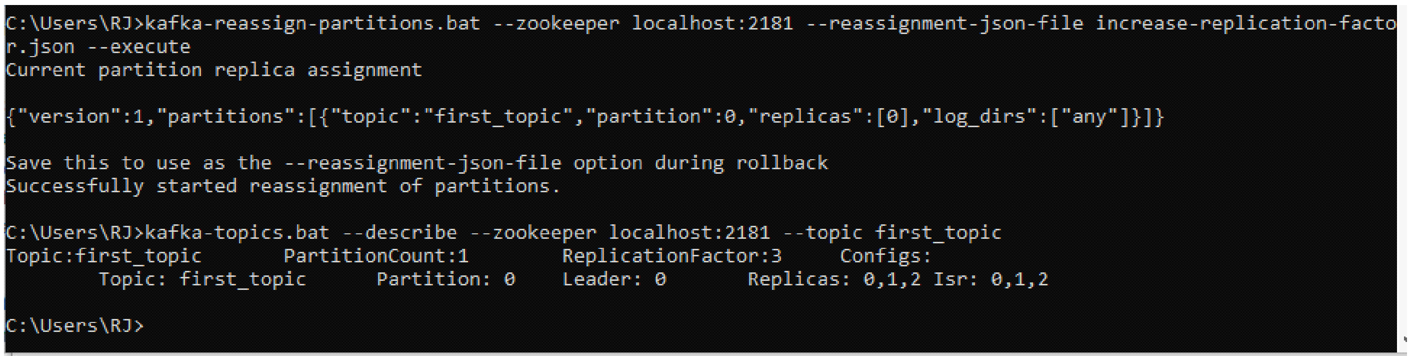
```
owerShell 6.2.0
opyright (c) Microsoft Corporation. All rights reserved.

ttps://aka.ms/pscore6-docs
ype 'help' to get help.

S C:\Users\RJ> kafka-console-producer.bat --broker-list localhost:9092 --topic first_topic
>Hey man!
>Wassup?
>All Good!
>Blink-182, Greenday or Sum 41?
>
```

9.

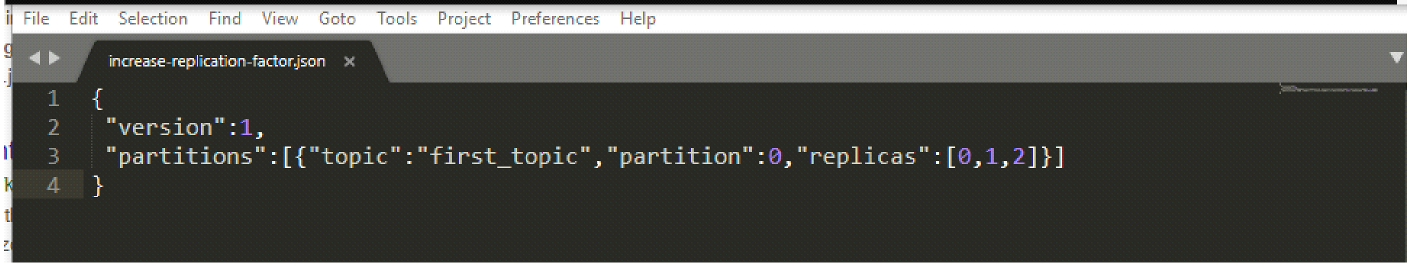


```
PS C:\Users\RJ> kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic first_topic --from-beginning
Hey man!
Wassup?
All Good!
Blink-182, Greenday or Sum 41?
```

10.

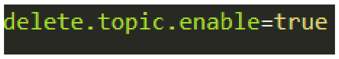


```
C:\Users\RJ>kafka-reassign-partitions.bat --zookeeper localhost:2181 --reassignment-json-file increase-replication-facto
r.json --execute
Current partition replica assignment

{"version":1,"partitions":[{"topic":"first_topic","partition":0,"replicas":[0],"log_dirs":["any"]}]}

Save this to use as the --reassignment-json-file option during rollback
Successfully started reassignment of partitions.

C:\Users\RJ>kafka-topics.bat --describe --zookeeper localhost:2181 --topic first_topic
Topic:first_topic       PartitionCount:1        ReplicationFactor:3     Configs:
        Topic: first_topic      Partition: 0    Leader: 0       Replicas: 0,1,2 Isr: 0,1,2

C:\Users\RJ>
```


File Edit Selection Find View Goto Tools Project Preferences Help
increase-replication-factor.json

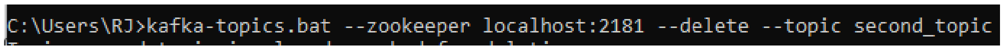
```
{
 "version":1,
 "partitions":[{"topic":"first_topic","partition":0,"replicas":[0,1,2]}]
}
```

11.



```
delete.topic.enable=true
```

In the config file add above line.

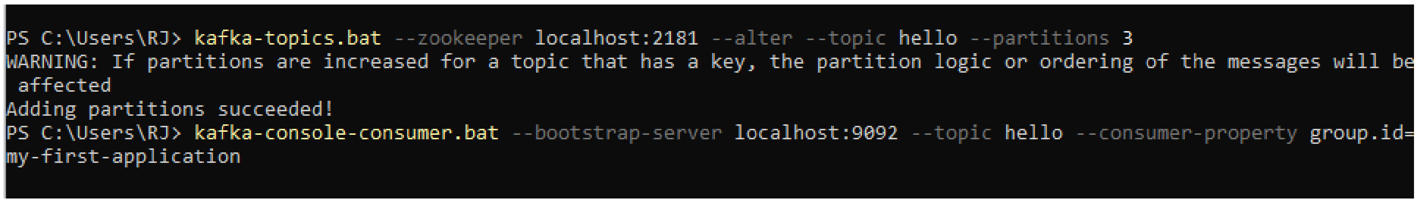


```
C:\Users\RJ>kafka-topics.bat --zookeeper localhost:2181 --delete --topic second_topic
```

Lab 2

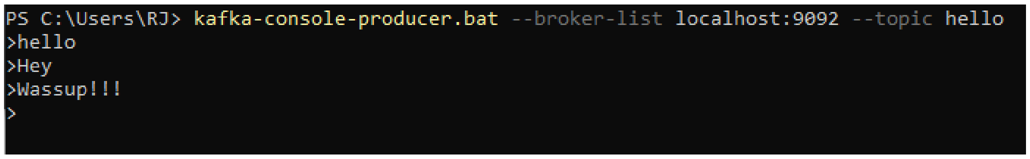
Sunday, April 14, 2019 9:45 PM

1. **Creating a topic named hello and consumer group named my-first application**



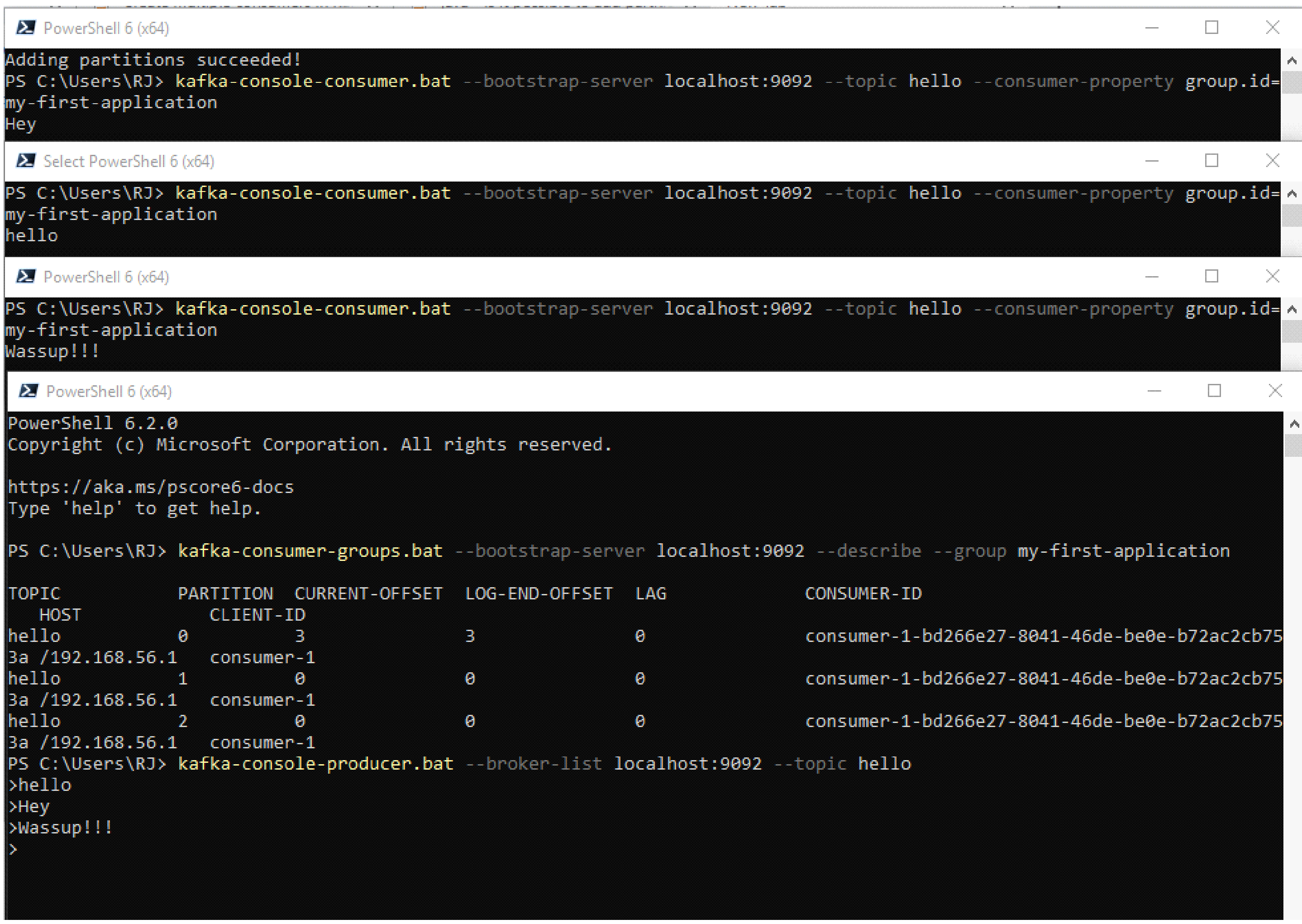
```
PS C:\Users\RJ> kafka-topics.bat --zookeeper localhost:2181 --alter --topic hello --partitions 3
WARNING: If partitions are increased for a topic that has a key, the partition logic or ordering of the messages will be
 affected
Adding partitions succeeded!
PS C:\Users\RJ> kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic hello --consumer-property group.id=
my-first-application
```

2. **Describing our newly created consumer**



```
PS C:\Users\RJ> kafka-console-producer.bat --broker-list localhost:9092 --topic hello
>hello
>Hey
>Wassup!!!
>
```

3. **Messages being spread to the consumer groups**



PowerShell 6 (x64)

```
Adding partitions succeeded!
PS C:\Users\RJ> kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic hello --consumer-property group.id=
my-first-application
Hey
```

Select PowerShell 6 (x64)

```
PS C:\Users\RJ> kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic hello --consumer-property group.id=
my-first-application
hello
```

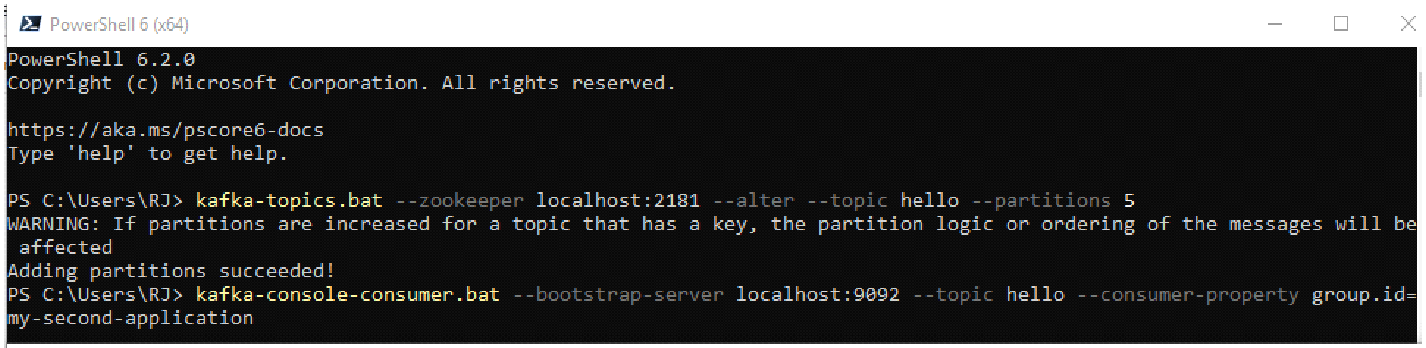
PowerShell 6 (x64)

```
PS C:\Users\RJ> kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic hello --consumer-property group.id=
my-first-application
Wassup!!!
```

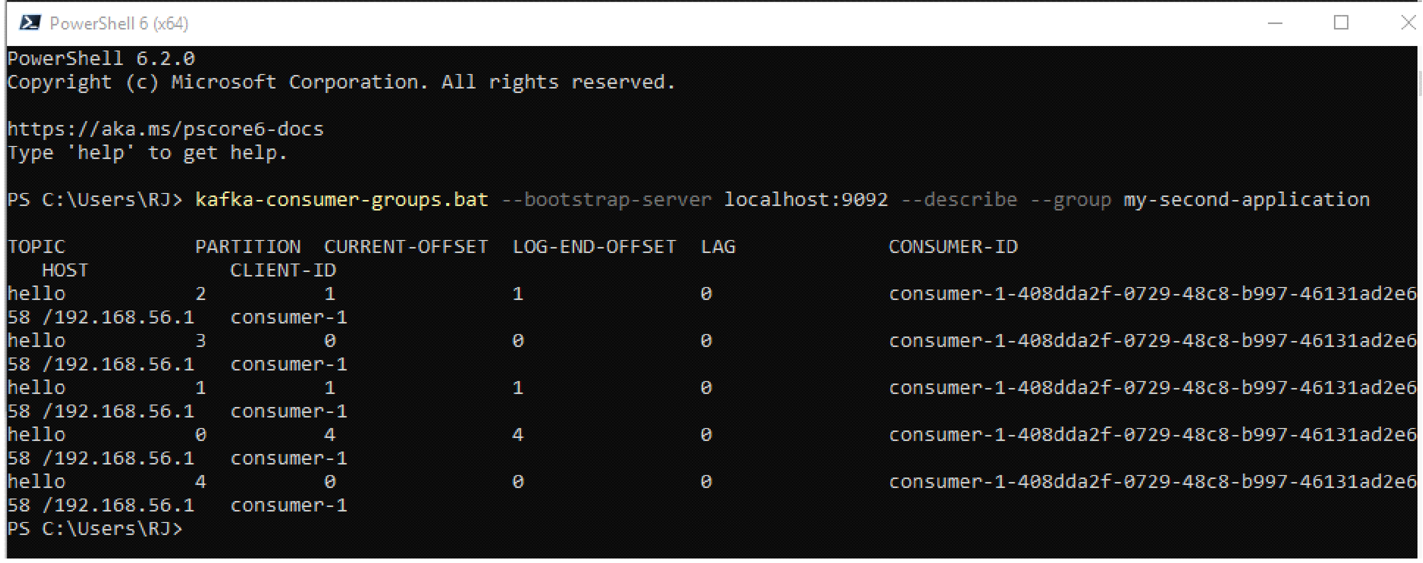
PowerShell 6 (x64)

```
PowerShell 6.2.0
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/pscore6-docs
Type 'help' to get help.

PS C:\Users\RJ> kafka-consumer-groups.bat --bootstrap-server localhost:9092 --describe --group my-first-application

TOPIC           PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG             CONSUMER-ID
   HOST            CLIENT-ID
hello           0          3               3               0               consumer-1-bd266e27-8041-46de-be0e-b72ac2cb75
3a /192.168.56.1   consumer-1
hello           1          0               0               0               consumer-1-bd266e27-8041-46de-be0e-b72ac2cb75
3a /192.168.56.1   consumer-1
hello           2          0               0               0               consumer-1-bd266e27-8041-46de-be0e-b72ac2cb75
3a /192.168.56.1   consumer-1
PS C:\Users\RJ> kafka-console-producer.bat --broker-list localhost:9092 --topic hello
>hello
>Hey
>Wassup!!!
>
```

4. **Creating a second consumer group named my-second-application**

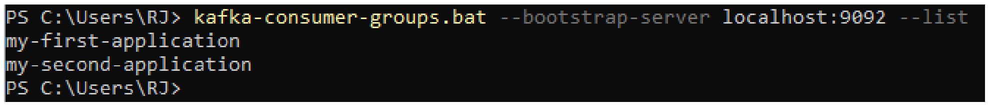


```
PowerShell 6.2.0
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/pscore6-docs
Type 'help' to get help.

PS C:\Users\RJ> kafka-topics.bat --zookeeper localhost:2181 --alter --topic hello --partitions 5
WARNING: If partitions are increased for a topic that has a key, the partition logic or ordering of the messages will be
 affected
Adding partitions succeeded!
PS C:\Users\RJ> kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic hello --consumer-property group.id=
my-second-application
```



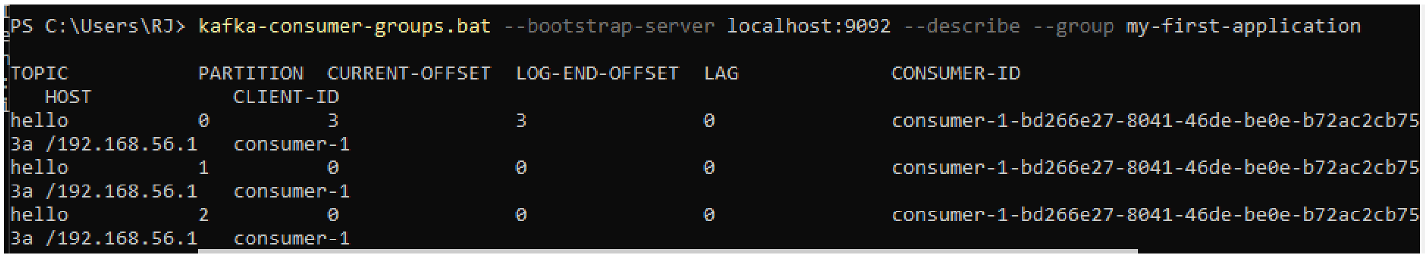
```
PowerShell 6.2.0
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/pscore6-docs
Type 'help' to get help.

PS C:\Users\RJ> kafka-consumer-groups.bat --bootstrap-server localhost:9092 --describe --group my-second-application

TOPIC           PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG             CONSUMER-ID
   HOST            CLIENT-ID
hello           2          1               1               0               consumer-1-408dda2f-0729-48c8-b997-46131ad2e6
58 /192.168.56.1   consumer-1
hello           3          0               0               0               consumer-1-408dda2f-0729-48c8-b997-46131ad2e6
58 /192.168.56.1   consumer-1
hello           1          1               1               0               consumer-1-408dda2f-0729-48c8-b997-46131ad2e6
58 /192.168.56.1   consumer-1
hello           0          4               4               0               consumer-1-408dda2f-0729-48c8-b997-46131ad2e6
58 /192.168.56.1   consumer-1
hello           4          0               0               0               consumer-1-408dda2f-0729-48c8-b997-46131ad2e6
58 /192.168.56.1   consumer-1
PS C:\Users\RJ>
```

5. **Listing all consumer groups**



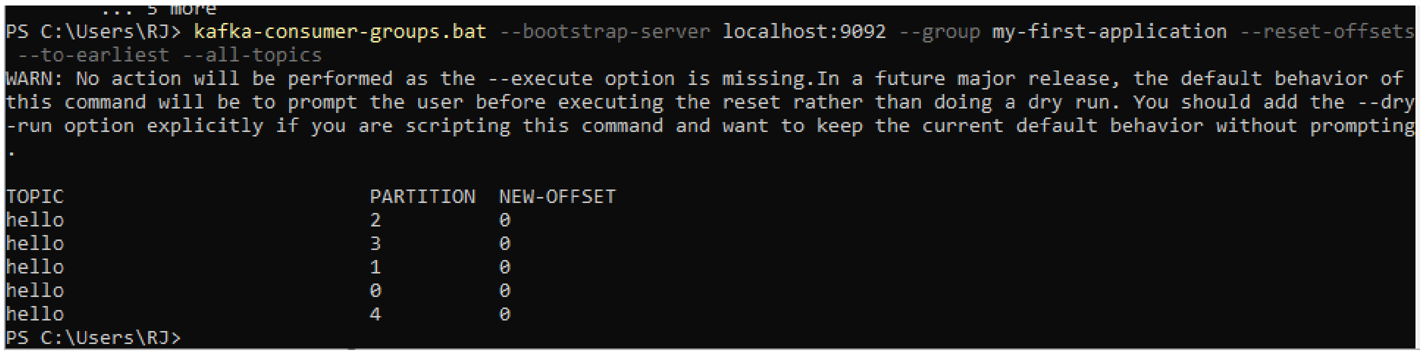
```
PS C:\Users\RJ> kafka-consumer-groups.bat --bootstrap-server localhost:9092 --list
my-first-application
my-second-application
PS C:\Users\RJ>
```

6. **Describe the first consumer group**

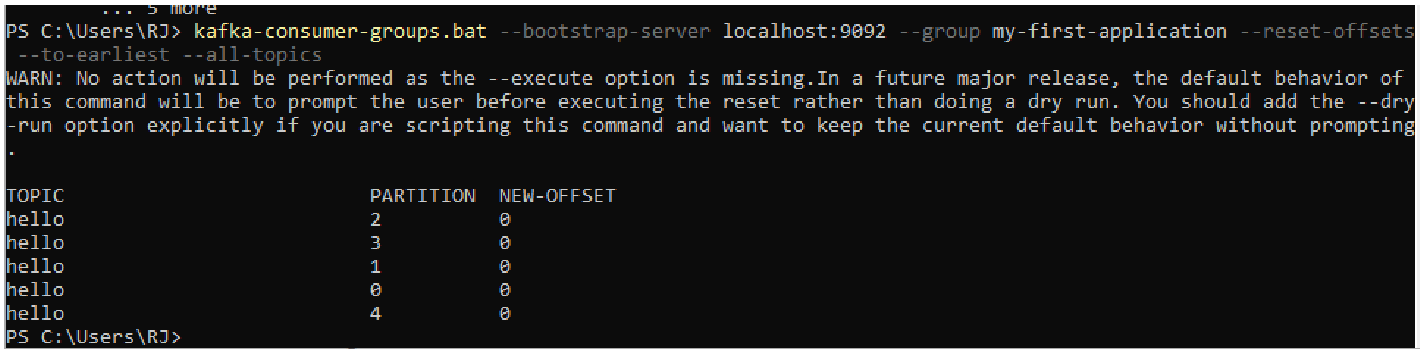


```
PS C:\Users\RJ> kafka-consumer-groups.bat --bootstrap-server localhost:9092 --describe --group my-first-application

TOPIC           PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG             CONSUMER-ID
   HOST            CLIENT-ID
hello           0          3               3               0               consumer-1-bd266e27-8041-46de-be0e-b72ac2cb75
3a /192.168.56.1   consumer-1
hello           1          0               0               0               consumer-1-bd266e27-8041-46de-be0e-b72ac2cb75
3a /192.168.56.1   consumer-1
hello           2          0               0               0               consumer-1-bd266e27-8041-46de-be0e-b72ac2cb75
3a /192.168.56.1   consumer-1
```

7. **Reset all offsets to earliest**



```
        ... 5 more
PS C:\Users\RJ> kafka-consumer-groups.bat --bootstrap-server localhost:9092 --group my-first-application --reset-offsets
 --to-earliest --all-topics
WARN: No action will be performed as the --execute option is missing.In a future major release, the default behavior of
this command will be to prompt the user before executing the reset rather than doing a dry run. You should add the --dry
-run option explicitly if you are scripting this command and want to keep the current default behavior without prompting
.

TOPIC                          PARTITION  NEW-OFFSET
hello                          2          0
hello                          3          0
hello                          1          0
hello                          0          0
hello                          4          0
PS C:\Users\RJ>
```

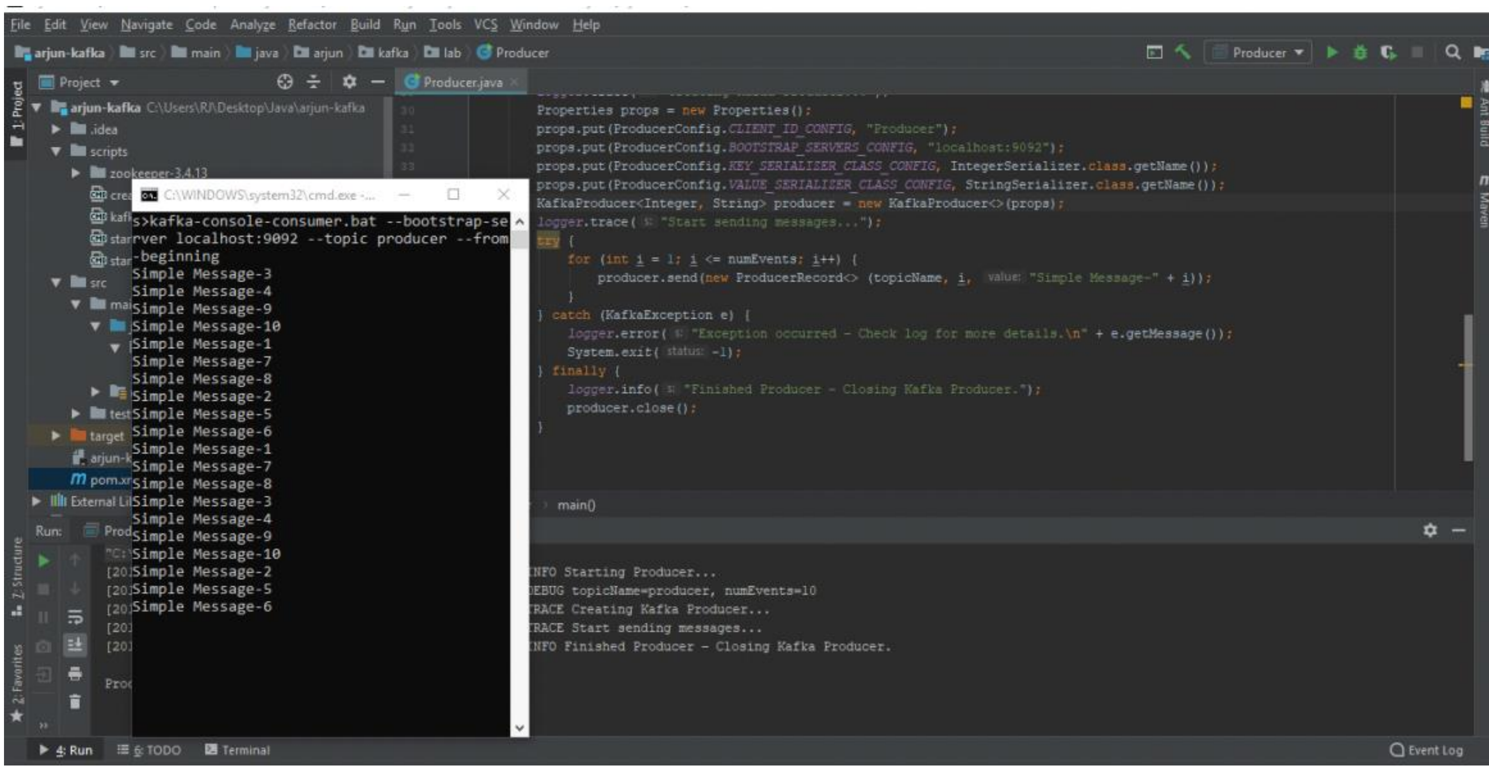


```
        ... 5 more
PS C:\Users\RJ> kafka-consumer-groups.bat --bootstrap-server localhost:9092 --group my-first-application --reset-offsets
 --to-earliest --all-topics
WARN: No action will be performed as the --execute option is missing.In a future major release, the default behavior of
this command will be to prompt the user before executing the reset rather than doing a dry run. You should add the --dry
-run option explicitly if you are scripting this command and want to keep the current default behavior without prompting
.

TOPIC                          PARTITION  NEW-OFFSET
hello                          2          0
hello                          3          0
hello                          1          0
hello                          0          0
hello                          4          0
PS C:\Users\RJ>
```

Kafka thru Java

Sunday, April 21, 2019 3:10 PM

1. **Fire and Forget**



```
Properties props = new Properties();
props.put(ProducerConfig.CLIENT_ID_CONFIG, "Producer");
props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, IntegerSerializer.class.getName());
props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, StringSerializer.class.getName());
KafkaProducer<Integer, String> producer = new KafkaProducer<>(props);
logger.trace( "Start sending messages...");
try {
    for (int i = 1; i <= numEvents; i++) {
        producer.send(new ProducerRecord<> (topicName, i, value: "Simple Message-" + i));
    }
} catch (KafkaException e) {
    logger.error( "Exception occurred - Check log for more details.\n" + e.getMessage());
    System.exit( status: -1);
} finally {
    logger.info( "Finished Producer - Closing Kafka Producer.");
    producer.close();
}
```

```
s>kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic producer --from-beginning
Simple Message-3
Simple Message-4
Simple Message-9
Simple Message-10
Simple Message-1
Simple Message-7
Simple Message-8
Simple Message-2
Simple Message-5
Simple Message-6
Simple Message-1
Simple Message-7
Simple Message-8
Simple Message-3
Simple Message-4
Simple Message-9
Simple Message-10
Simple Message-2
Simple Message-5
Simple Message-6
```

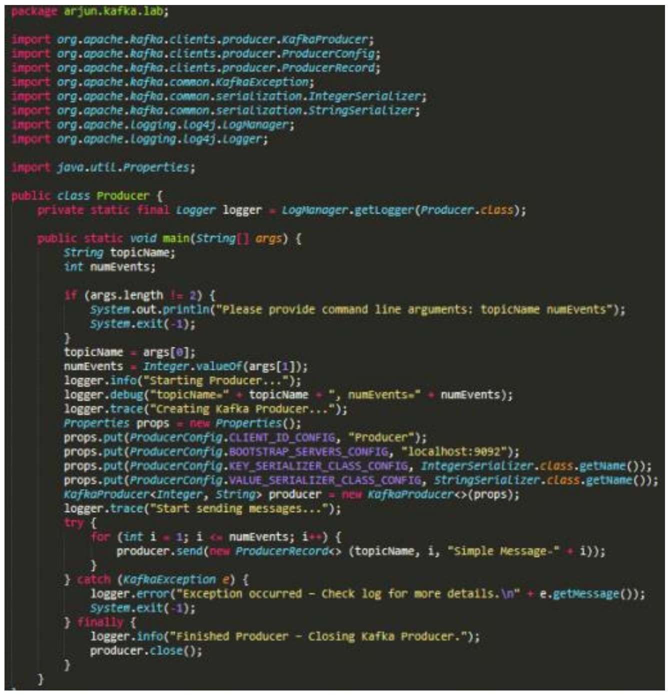
```
INFO Starting Producer...
DEBUG topicName=producer, numEvents=10
TRACE Creating Kafka Producer...
TRACE Start sending messages...
INFO Finished Producer - Closing Kafka Producer.
```

Code -

Producer (Simple String Producer)

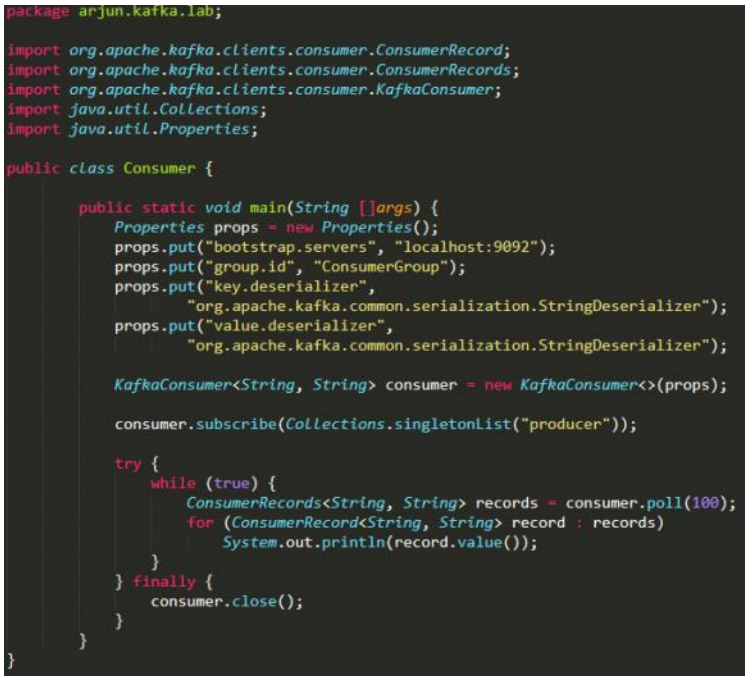
```java
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerRecord;

import java.util.Properties;

public class Producers
{
    public static void main(String[] args) {

        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");

        Producer<String, String> producer = new KafkaProducer<>(props);

        ProducerRecord<String, String> record = new ProducerRecord<>("producer", "String Key", "Hello,Fire and Forget");

        try {
            producer.send(record);
        } catch (Exception e) {
            e.printStackTrace();
        }

        producer.close();
    }
}
```

Producer (Loop Producer)

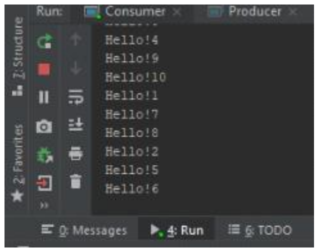


```java
package arjun.kafka.lab;

import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.clients.producer.ProducerRecord;
import org.apache.kafka.common.KafkaException;
import org.apache.kafka.common.serialization.IntegerSerializer;
import org.apache.kafka.common.serialization.StringSerializer;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

import java.util.Properties;

public class Producer {
    private static final Logger logger = LogManager.getLogger(Producer.class);

    public static void main(String[] args) {
        String topicName;
        int numEvents;

        if (args.length != 2) {
            System.out.println("Please provide command line arguments: topicName numEvents");
            System.exit(-1);
        }
        topicName = args[0];
        numEvents = Integer.valueOf(args[1]);
        logger.info("Starting Producer...");
        logger.debug("topicName=" + topicName + ", numEvents=" + numEvents);
        logger.trace("Creating Kafka Producer...");
        Properties props = new Properties();
        props.put(ProducerConfig.CLIENT_ID_CONFIG, "Producer");
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, IntegerSerializer.class.getName());
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, StringSerializer.class.getName());
        KafkaProducer<Integer, String> producer = new KafkaProducer<>(props);
        logger.trace("Start sending messages...");
        try {
            for (int i = 1; i <= numEvents; i++) {
                producer.send(new ProducerRecord<> (topicName, i, "Simple Message-" + i));
            }
        } catch (KafkaException e) {
            logger.error("Exception occurred - Check log for more details.\n" + e.getMessage());
            System.exit(-1);
        } finally {
            logger.info("Finished Producer - Closing Kafka Producer.");
            producer.close();
        }
    }
```

Consumer-

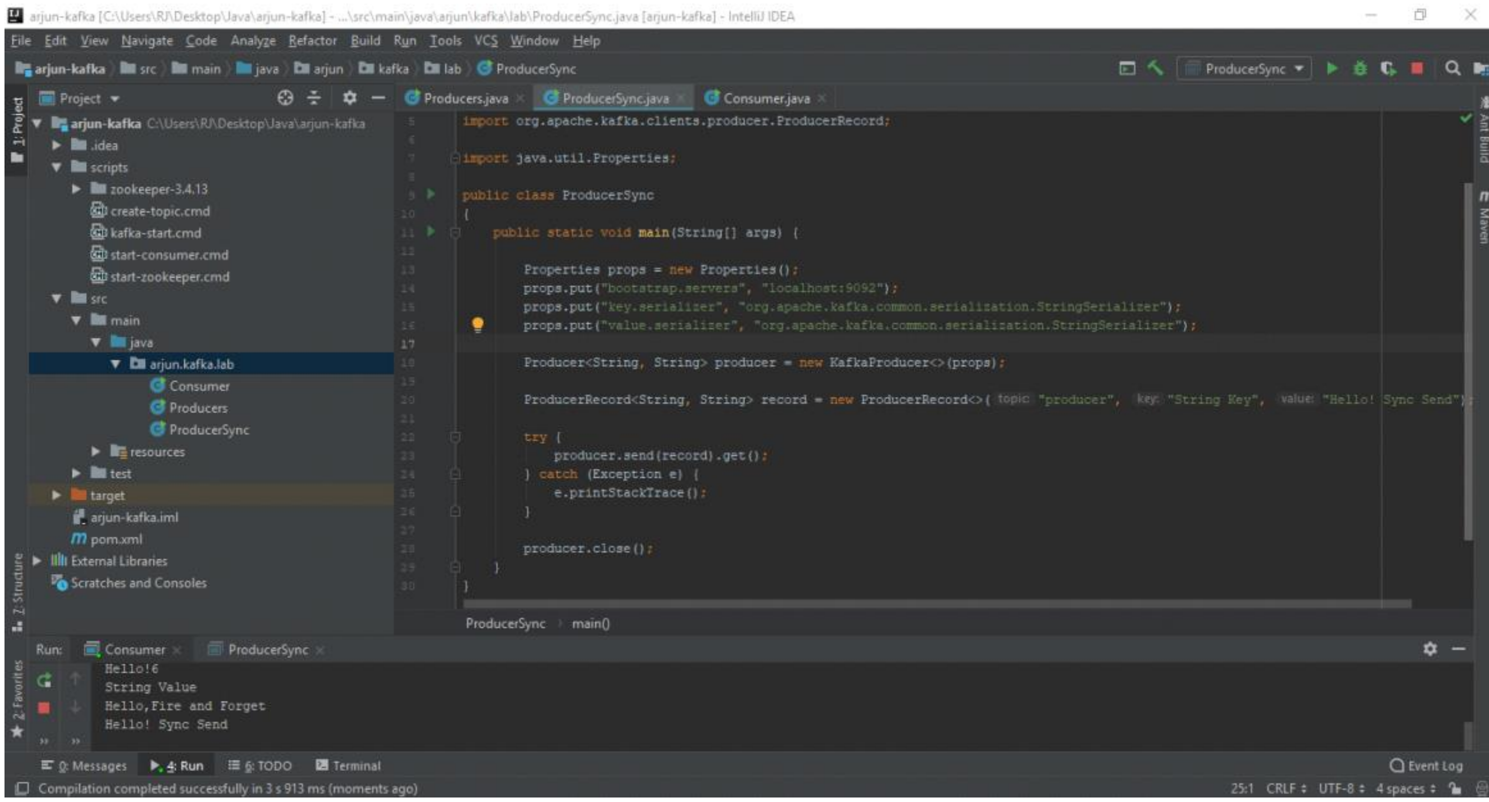


```java
package arjun.kafka.lab;

import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import java.util.Collections;
import java.util.Properties;

public class Consumer {

        public static void main(String []args) {
            Properties props = new Properties();
            props.put("bootstrap.servers", "localhost:9092");
            props.put("group.id", "ConsumerGroup");
            props.put("key.deserializer",
                    "org.apache.kafka.common.serialization.StringDeserializer");
            props.put("value.deserializer",
                    "org.apache.kafka.common.serialization.StringDeserializer");

            KafkaConsumer<String, String> consumer = new KafkaConsumer<>(props);

            consumer.subscribe(Collections.singletonList("producer"));

            try {
                while (true) {
                    ConsumerRecords<String, String> records = consumer.poll(100);
                    for (ConsumerRecord<String, String> record : records)
                        System.out.println(record.value());
                }
            } finally {
                consumer.close();
            }
        }
}
```



```
Hello!4
Hello!9
Hello!10
Hello!1
Hello!7
Hello!8
Hello!2
Hello!5
Hello!6
```

2. **Synchronous Send**

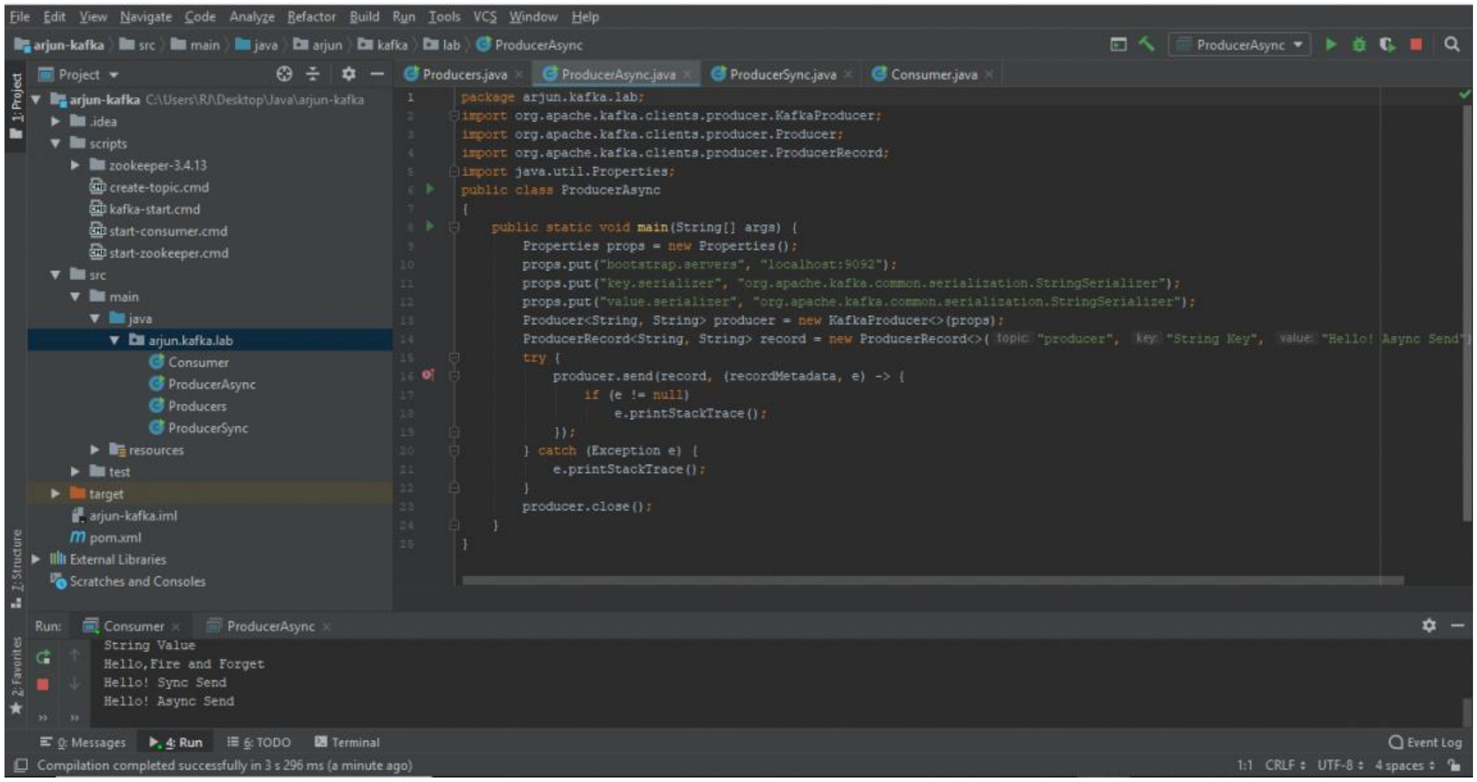


```
import org.apache.kafka.clients.producer.ProducerRecord;

import java.util.Properties;

public class ProducerSync
{
    public static void main(String[] args) {

        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");

        Producer<String, String> producer = new KafkaProducer<>(props);

        ProducerRecord<String, String> record = new ProducerRecord<>( topic: "producer",  key: "String Key",  value: "Hello! Sync Send"

        try {
            producer.send(record).get();
        } catch (Exception e) {
            e.printStackTrace();
        }

        producer.close();
    }
}
```

```
Hello!6
String Value
Hello,Fire and Forget
Hello! Sync Send
```

Compilation completed successfully in 3 s 913 ms (moments ago)

3. Asynchronous Send



```
package arjun.kafka.lab;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerRecord;
import java.util.Properties;
public class ProducerAsync
{
    public static void main(String[] args) {
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        Producer<String, String> producer = new KafkaProducer<>(props);
        ProducerRecord<String, String> record = new ProducerRecord<>( topic: "producer",  key: "String Key",  value: "Hello! Async Send"
        try {
            producer.send(record, (recordMetadata, e) -> {
                if (e != null)
                    e.printStackTrace();
            });
        } catch (Exception e) {
            e.printStackTrace();
        }
        producer.close();
    }
}
```

```
String Value
Hello,Fire and Forget
Hello! Sync Send
Hello! Async Send
```

Compilation completed successfully in 3 s 296 ms (a minute ago)

.