Run this command .\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties to start Zookeeper

Run this command .\bin\windows\kafka-server-start.bat .\config\server.properties to start Kafka

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

==============

Note: edit

Add in path : C:\sw\kafka\_2.12-2.3.0\bin\windows

Run zookeeper

zookeeper.properties:-dataDir=C:/sw/kafka\_2.12-2.3.0/data/zookeeper

C:\sw\kafka\_2.12-2.3.0>zookeeper-server-start.bat config\zookeeper.properties

Run kafka

Server.properties

log.dirs=C:/sw/kafka\_2.12-2.3.0/data/kafka

C:\sw\kafka\_2.12-2.3.0>kafka-server-start.bat config\server.properties

**Kafka is a distributed system**

Machine generated alternative text:
Why Apache Kafka: 
Decoupling of data streams & systems 
Website Events 
natahacn 
Financial 
Pricing Data 
Transactions 
Apache Kafka 
A hø•throuøvut distrbuted messa$;g systen. 
User 
interactions 

Machine generated alternative text:
Apache Kafka Series - Learn Apache Kafka for Beginners v2 
Why Apache Kafka 
• Created by Linkedln, now Open Source Project mainly maintained by Confluent 
• Distributed, resilient architecture, fault tolerant 
• Horizontal scalability: 
• Can scale to IOOs of brokers 
• Can scale to millions of messages per second 
• High performance (latency of less than 10ms) — real time 
• Used by the 2000+ firms, 35% of the Fortune 500: 
airbnb 
NETFLIX 
UBER 
Linked ff 
Walmart 

Machine generated alternative text:
Apache Kana: Use cases 
• Messaging System 
• Activity Tracking 
• Gather metrics from many different locations 
• Application Logs gathering 
• Stream processing (with the Kafka Streams API or Spark for example) 
• De-coupling of system dependencies 
• Integration with Spark, Flink, Storm, Hadoop, and many other Big Data 
technologies 

Machine generated alternative text:
For example... 
• Netflix uses Kafka to apply recommendations in real-time while 
you're watching TV shows 
• Uber uses Kafka to gather user, taxi and trip data in real-time to 
compute and forecast demand, and compute surge pricing in real-time 
• Linkedln uses Kafka to prevent spam, collect user interactions to 
make better connection recommendations in real time. 
• Remember that Kafka is only used as a transportation mechanism! 

Machine generated alternative text:
Course Structure 
Part 3 - Advanced & 
Annexes 
Advanced Topic 
Configuration 
Annexes 
Part 1 
Fundamentals — 
Kafka Theory 
Starting Kafka 
Kafka CLI 
Kafka & Java 101 
3h 
Part 2 
Real World - 3h30 
Twitter Producer 
ElasticSearch 
Consumer 
Extended API Intro 
Case Studies 
Kafka in the Enterprise 

Machine generated alternative text:
Apache Kafta Series 
Welcome! 
2. 
3. 
4. 
5. 
6. 
7. 
8. 
Kafka For Beginners: get a strong base for Kafka, basic 
operations, write your first producers and consumers 
Kafka Connect API: Understand how to import / export data to / from Kafka 
Kafka Streams API: Learn how to process and transform data within Kafka 
Kafka Cluster Setup & Administration: Get a deep understanding of how Kafka 
& Zookeeper works, how to Setup Kafka and various administration tasks 
Confluent Components: REST Proxy and Schema Registry 
Kafka Security: Setup Kafka security in a Cluster and Integrate your applications 
with Kafka Security 
Kafka Monitoring and Operations: use Prometheus and Grafana to monitor 
Kafka, learn operations 
And more courses in the pipeline! 

Machine generated alternative text:
Topics, partitions and offsets 
• Topics: a particular stream of data 
• Similar to a table in a database (without all the constraints) 
• You can have as many topics as you want 
• A topic is identified by its name 
• Topics are split in partitions 
• Each partition is ordered 
• Each message within a partition gets an incremental id, called offset 
45 
Kafka Topic 
Partition O 
Partition I 
Partition 2 
o 
o 
o 
1 
1 
1 
2 
2 
2 
3 
3 
3 
4 
4 
5 
5 
6 
6 
1 
789 
2 
8 
1 
789 
writes 

Machine generated alternative text:
Topic example: truck 
Truck GPS data 
00 0 
Every 20 seconds 
00 0 
1 
o 
Location dashboard 
Notification service 
• Say you have a fleet of trucks, each truck reports its GPS position to Kafka. 
• You can have a topic trucks_gps that contains the position of all trucks. 
• Each truck will send a message to Kafka every 20 seconds, each message 
will contain the truck ID and the truck position (latitude and longitude) 
• We choose to create that topic with 10 partitions (arbitrary number) 

Machine generated alternative text:
Topics, partitions and offsets 
12 
writes 
Kafka Topic 
Partition O 
Partition I 
Partition 2 
o 
o 
o 
1 
1 
2 
2 
1 
3456789 
3 4 5 67 
1 
3456 
789 
• Offset only have a meaning for a specific partition. 
• E.g. offset 3 in partition 0 doesn't represent the same data as offset 3 in partition I 
• Order is guaranteed only within a partition (not across partitions) 
Data is kept only for a limited time (default is one week) 
• 
Once the data is written to a partition, it can't be changed (immutability) 
• 
Data is assigned randomly to a partition unless a key is provided (more on this later) 
• 

Machine generated alternative text:
Brokers 
• A Kafka cluster is composed of multiple brokers (servers) 
• Each broker is identified with its ID (integer) 
• Each broker contains certain topic partitions 
• After connecting to any broker (called a bootstrap broker), you will be 
connected to the entire cluster 
• A good number to get started is 3 brokers, but some big clusters have over 
100 brokers 
• In these examples we choose to number brokers starting at 100 (arbitrary) 
Broker 101 
Broker 102 
Broker 103 

Machine generated alternative text:
Brokers and topics 
• Example of 
Topic-A 
with 3 partitions 
• Example of 
with 2 partitions 
Topic-B 
Broker 103 
Topic-A 
Partition 1 
Broker 101 
Topic-A 
Partition O 
Partition : 
Broker 102 
Topic-A 
Partition 2 
Partition 
• Note: Data is distributed and Broker 103 doesn't have any Topic B data 

Machine generated alternative text:
Topic replication factor 
• Topics should have a replication factor I (usually between 2 and 3) 
• This way if a broker is down, another broker can serve the data 
• Example: Topic-A with 2 partitions and replication factor of 2 
Broker 101 
Partition O 
Topic-A 
Broker 102 
Partition 
opic- 
Partition O 
Broker 103 
Partition 
opic- 

Machine generated alternative text:
Topic replication factor 
• Example: we lost Broker 102 
• Result: Broker 101 and 103 can still serve the data 
Broker 103 
Partition 1 
Broker 101 
Partition O 
Topic-A 
Broker 102 

Machine generated alternative text:
Concept of Leader for a Partition 
• At any time only ONE broker can be a leader for a given partition 
• The other brokers will synchronize the data 
• Therefore each partition has one leader and multiple ISR (in-sync replica) 
Broker 101 
Partition O 
Topic-A (Leader) 
replication 
Broker 102 
Partition 1 
opic-A (Leader 
Partition O 
Broker 103 
replication 
Partition 1 
Topic-A (ISR) 

Load balance in kafka

Machine generated alternative text:
Producers 
• Producers write data to topics (which is made of partitions) 
• Producers automatically know to which broker and partition to write to 
• In case of Broker failures, Producers will automatically recover 
12 
45 
Producer 
The load is balanced to many brokers 
thanks to the number of partitions 
Send data 
Broker 101 
o 
Topic-A / Partition O 
Broker 102 
o 
Topic-A / Partition 1 
Broker 103 
Topic-A / Partition 2 
1 
2 
3 
3 
4 
5 
6 
789 
8 
1 
2 
writes 

High level :

Concept of key ,acknowledgement and round robin

Machine generated alternative text:
Producers 
• Producers can choose to receive acknowledgment of data writes: 
• acks:O: Producer won't wait for acknowledgment (possible data loss) 
• agkE_L: Producer will wait for leader acknowledgment (limited data loss) 
• agk'äll.• Leader + replicas acknowledgment (no data loss) 
789 
8 
1 
2 
writes 
Producer 
Automatic load balance by broker 
Send data 
Broker 101 
o 
Topic-A / Partition O 
Broker 102 
o 
Topic-A / Partition 1 
Broker 103 
Topic-A / Partition 2 
1 
1 
2 
2 
3 
3 
4 
4 
5 
5 
6 

Machine generated alternative text:
Producers: Message keys 
• Producers can choose to send a key with the message (string, number, etc..) 
• If key—null, data is sent round robin (broker 101 then 102 then 103...) 
• If a key is sent, then all messages for that key will always go to the same partition 
• A key is basically sent if you need message ordering for a specific field (ex: truck_id) 
example; 
Producer 
Broker 101 
Topic-A / Partition O 
Send data 
data will always be in partition O 
data will always be in partition O 
345 data will always be in partition I 
456 data will always be in partition 1 
Broker 102 
Topic-A / Partition 1 
truck id 
truck id 
Key is truck_id 

Machine generated alternative text:
Consumers 
• Consumers read data from a topic (identified by name) 
• Consumers know which broker to read from 
• In case of broker failures, consumers know how to recover 
• Data is read in order within each partitions 
Consumer 
Consumer 
Broker 101 
Topic-A/ Partition O 
Broker 102 
Topic-A / Partition 1 
Broker 103 
Topic-A / Partition 2 
o 
o 
1 
1 
2 
2 
3 
3 
4 
4 
Read in order 
11 
56789 
Read in order 

Machine generated alternative text:
Consumer Groups 
• Consumers read data in consumer groups 
• Each consumer within a group reads from exclusive partitions 
• If you have more consumers than partitions, some consumers will be inactive 
Consumer 1 
Topic-A 
Partition O 
Consumer 2 
Topic-A 
Partition 1 
Consumer 1 
Topic-A 
Partition 2 
Consumer 2 
Consumer 3 
Consumer : 
consumer-group-S 

Machine generated alternative text:
Consumer Groups 
What if too many consumers? 
• If you have more consumers than partitions, some consumers will be inactive 
Topic-A 
Partition O 
Consumer 1 
Topic-A 
Partition 1 
Consumer 2 
Topic-A 
Partition 2 
Consumer 4 
Consumer 3 
(inative) 
consumer-group-application 

Machine generated alternative text:
Consumer Offsets 
• Kafka 
stores the offsets at which a consumer group has been reading 
• The offsets committed live in a Kafka topic named 
consumer offsets 
• When a consumer in a group has processed data received from Kafka, it 
should be committing the offsets 
• If a consumer dies, it will be able to read back from where it left off 
thanks to the committed consumer offsets! 
Consumer from 
Consumer Group 
4 
2 
5 
8 
4 
2 
5 
9 
4 
2 
6 
o 
4 
2 
6 
1 
4 
2 
6 
2 
4 
2 
6 
3 
4 
2 
6 
4 
4 
2 
6 
5 
4 
2 
6 
6 
4 
2 
6 
7 
4 
2 
6 
8 
Committed offset 
4 
2 
6 
9 
Reads 

Machine generated alternative text:
Delivery semantics for consumers 
• Consumers choose when to commit offsets. 
• There are 3 delivery semantics: 
• At most once: 
• offsets are committed as soon as the message is received. 
• If the processing goes wrong, the message will be lost (it won't be read again). 
• At least once (usually preferred): 
• offsets are committed after the message is processed. 
• If the processing goes wrong, the message will be read again. 
• This can result in duplicate processing of messages. Make sure your processing is 
idempotent (i.e. processing again the messages won't impact your systems) 
• Exactl once: 
• Can be achieve 
a wor ows using a 
treams 

Machine generated alternative text:
Kafka Broker Discovery 
• Every Kafka broker is also called a "bootstrap server" 
• That means that you only need to connect to one broker, 
and you will be connected to the entire cluster. 
• Each broker knows about all brokers, topics and partitions (metadata) 
1. Connection + 
Metadata request 
Kafka Client 
2. List of all broke 
3. Can connect to the needed bro 
Broker 101 
(bootstrap) 
Broker 102 
(bootstrap) 
Kafka Cluster 
Broker 103 
(bootstrap) 
Broker 104 
(bootstrap) 
Broker 105 
(bootstrap) 

Machine generated alternative text:
Zookeeper 
• Zookeeper manages brokers (keeps a list of them) 
• Zookeeper helps in performing leader election for partitions 
• Zookeeper sends notifications to Kafka in case of changes (e.g. new 
topic, broker dies, broker comes up, delete topics, etc.. 
• Kafka can't work without Zookeeper 
• Zookeeper by design operates with an odd number of servers (3, 5, 7) 
• Zookeeper has a leader (handle writes) the rest of the servers are 
followers (handle reads) 
• (Zookeeper does NOT store consumer offsets with Kafka vO. 10) 

Machine generated alternative text:
Zookeeper 
Zookeeper 
Server 2 
(Leader) 
Broker 3 
Zookeeper 
(Follower) 
Kafka 
Broker 4 
Kafka 
Broker S 
Kafka 
Broker 1 
Zookeeper 
Server 1 
(Follower) 
Kafka 
Broker 2 

Machine generated alternative text:
Kafka Guarantees 
• Messages are appended to a topic-partition in the order they are sent 
• Consumers read messages in the order stored in a topic-partition 
• With a replication factor of N, producers and consumers can tolerate 
up to N-l brokers being down 
• This is why a replication factor of 3 is a good idea: 
• Allows for one broker to be taken down for maintenance 
• Allows for another broker to be taken down unexpectedly 
• As long as the number of partitions remains constant for a topic (no 
new partitions), the same key will always go to the same partition 

Machine generated alternative text:
Theory Roundup 
We've looked at all the Kafka concepts 
Kafka Cluster 
+ topics 
Source 
Producers 
Systems 
+ round robin 
+ key based ordering 
+ acks strategy 
Broker 101 
Broker 102 
Broker 109 
Zookeeper 
+ partitions 
+ replication 
+ partition leader & in-sync-replicas (ISR) 
+ offsets topic 
Target 
Consumers 
Systems 
+ consumer offsets 
+ consumer groups 
+ at least once 
+ at most once 
+ leader follower 
+ broker management 

**Very important**: you only need to connect to one broker (any broker) and just provide the topic name you want to write to. Kafka Clients will route your data to the appropriate brokers and partitions for you!

**Creating topic**

Machine generated alternative text:
kafka—topics —zookeeper 127.Ø.Ø.1:2181 —topic first_topic —create —partitions 3 —replication—factor 2 
RAFWING•. Due to limitations in etric noes, topics with a period or underscore ( ) could collide. To avoid issues it is best to 
use either, but not both. 
Error *fiile executing topic : ieptication factor: 2 larger than available brokers: 1. 
org.apache.kafka.comon.errors.InvalidRepucationFactorException: Replication factor: 2 larger than avai 
lable brokers: 1. 
( kafka. admin. TopicCcmvand$) 

**Note** : we cannot have replication factor larger than available broker

**Command to create Topic:**

C:\sw\kafka\_2.12-2.3.0>kafka-topics --zookeeper 127.0.0.1:2181 --topic first\_topic --create --partitions 3 --replication-factor 1

**How u know kafka topic is created**

C:\sw\kafka\_2.12-2.3.0>kafka-topics --zookeeper 127.0.0.1:2181 --list

first\_topic

**command to know more about topic:**

C:\sw\kafka\_2.12-2.3.0>kafka-topics --zookeeper 127.0.0.1:2181 --topic first\_topic --describe

Topic:first\_topic PartitionCount:3 ReplicationFactor:1 Configs:

Topic: first\_topic Partition: 0 Leader: 0 Replicas: 0 Isr: 0

Topic: first\_topic Partition: 1 Leader: 0 Replicas: 0 Isr: 0

Topic: first\_topic Partition: 2 Leader: 0 Replicas: 0 Isr: 0

**Kafka-console-producer**

C:\sw\kafka\_2.12-2.3.0>kafka-console-producer --broker-list 127.0.0.1:9092 --topic first\_topic

>new course

>learning kafka

>Terminate batch job (Y/N)? Y

**adding property:**

C:\sw\kafka\_2.12-2.3.0>kafka-console-producer --broker-list 127.0.0.1:9092 --topic first\_topic --producer-property acks=all

>some message is acked

>just for fun learning

>Terminate batch job (Y/N)? y

Note: if topic is not created and producing message

Machine generated alternative text:
—broker—Ust 127. 1:9092 —topic 
Ryey this topic does not exist! 
(2ø18—ø8—24 0921 (Producer clientldzconsote—producerl Error *lite fetching ætadata with correlation id 1 : Inw_topiczLEA 
ien? 
*another essage 
kafka-topics —zookeeper 127.ø.ø.1:2181 —Ust 
f irst_topic 
new_topic 
kafka-topics —zookeeper 127.ø.ø.1:2181 
ReplicationFactor: 1 
PartitionCount:1 
—topic new_topic —describe 
Con f igs: 
Replicas: 
Isr: ø 
TOOic: nev_topic 
Partition:ø Leader: 

5)**Kafka Consumer command:**

bootstrap server is kafka server only

kafka-console-consumer --bootstrap-server 127.0.0.1:9092 --topic first\_topic

**NOTE: how to read all message at a time from consumer**

C:\sw\kafka\_2.12-2.3.0>kafka-console-consumer --bootstrap-server 127.0.0.1:9092 --topic first\_topic **--from-beginning**

just for fun learning

hi

hello

new course

**Kafka consumer in group**

Machine generated alternative text:
Grc•up 

A **consumer group** is a **group** of related **consumers**that perform a task, like putting data into Hadoop or sending messages to a service. **Consumer groups**each have unique offsets per partition. Different**consumer groups** can read from different locations in a partition.

C:\sw\kafka\_2.12-2.3.0>kafka-console-consumer --bootstrap-server 127.0.0.1:9092 --topic first\_topic --group my-first-application

blue

Machine generated alternative text:
C: XswXkafka 2.12-2.3. Dkafka-console-producer 
) red 
)blue 
) green 
)orange 
--broker-list 127.e.e.1:9e92 
- -topic first _ topic 

Machine generated alternative text:
C: XswXkafka 2.12-2.3. Dkafka-console-consumer 
blue 
--bootstrap-server 127.e.e.1:9B92 
- -topic first _ topic 
- -group my-first-application 

Machine generated alternative text:
C: XswXkafka 2.12-2.3. Dkafka-console-consumer 
red 
green 
orange 
--bootstrap-server 127.e.e.1:9B92 
- -topic first _ topic 
- -group my-first-application 

**Kafka-consumer-group**

**Note:- my-first-application is the name of group**

**To show list of group**

C:\sw\kafka\_2.12-2.3.0>kafka-consumer-groups --bootstrap-server localhost:9092 --list

my-first-application

Machine generated alternative text:
—bootstrap—server Vocalhost:9092 
irst—app lication 
console—con s LME r—1Ø824 
my-second—app licat ion 
—bootstrap—server tocalhost:9ø92 —describe 
Note:As of Kafka 2. l, the "console-consumer-" groups 
will not appear in this command anymore. 

Machine generated alternative text:
C:XswXkafka 2.12-2.3. 
4ca6-9baa- 
/1e.e.75.1 
4ca6-9baa- 
/1e.e.75.1 
/1e.e.75.1 
Dkafka-consumer-groups 
- -bootstrap- server 
CURRENT-OFFSET 
localhost : 9B92 - -describe 
LOG-END-OFFSET 
4 
4 
3 
LAG 
e 
e 
e 
- -group my-first-application 
4B9c 
85acd77c1c99 
85acd77c1c99 
-9a18- aeebce4826eb 
HOST 
CLIENT-ID 
consumer- 1 
consumer- 1 
consumer- 1 
GROUP 
my-first-application 
my-first-application 
my-first-application 
TOPIC 
first 
first 
first 
topic 
topic 
topic 
PARTITION 
e 
1 
2 
CONSUMER-ID 
consumer- 
consumer- 
consumer- 
1-77b3e1ad-de98- 
1-77b3e1ad-de98- 
1-e3594a31- 
59 ff_ 
4 
4 
3 

**Resetting offsets**

C:\sw\kafka\_2.12-2.3.0>kafka-consumer-groups --bootstrap-server localhost:9092 --group my-second-application --reset-offsets --to-earliest --execute --topic second\_topic

Machine generated alternative text:
C: XswXkafka 2.12-2.3. Dkafka-consumer-groups 
fsets --to-earliest - -execute --topic 
second 
topic 
topic 
topic 
- -bootstrap-server localhost : 9B92 
topic 
-group my-second- application 
-reset-o 
GROUP 
my-second-application 
my-second-application 
my-second-application 
TOPIC 
second 
second 
second 
PARTITION NEW-OFFSET 
e 
2 
1 
e 
e 
e 

Machine generated alternative text:
on —reset—offsets 
(2Ø18-Ø8-24 11:39: lø, 
2. Value will be set 
(2018-ø8-24 
1. Value will be set 
(2ø18-øg-24 
e. Value wilt be set 
TOPIC 
f i rst_topic 
f irst—topic 
f irst_topic 
—bootstrap—server localhost:9Ø92 —group 
—shift—by 2 —execute —topic first_topic 
7911 WAR-I New offset (15) is higher than latest offset for topic partition first_topic— 
to 13 (kafka.admin. 
7921 WR New offset (12) is higher than latest Offset for topic partition first_topic— 
to lø (kafka.admin.ConsuærGroupComvandS) 
7921 WAR4 offset (13) is higher than latest offset for topic partition first_topic— 
to 11 (kafka. ConsuerGroupC•'and$) 
PARTITIm NEW-OFFSET 
2 
1 
13 
lø 
11 
—bootstrap—server locathost:9ø92 —group nFfirst—appucation — 
-reset—offsets —shift—by —2 —execute —topic first_topic 
TOPIC 
first—topic 
f irst_topic 
f irst_topic 
PARTITICN NEW-OFFSET 
2 
1 
9 

**Kafka-manager:**

<https://github.com/yahoo/kafka-manager>

For ui kafka tool:-

<http://www.kafkatool.com/>

============================================================================================

**Kafka java programming**

download  **install** **IntelliJ Community Edition**:

<https://www.jetbrains.com/idea/download/>

**Kafka dependency :-**

<dependency>

<groupId>org.apache.kafka</groupId>

<artifactId>kafka-clients</artifactId>

<version>2.0.0</version>

</dependency>

1. **Creating producer**

**Note: it is asynchronous so we have to use producer.flush() and producer.close()**

Machine generated alternative text:
String bootstrapServers "127.ø.ø.I:9092"; 
// create Producer properties 
Properties properties : nw Properties(); 
properties. bootstrapservers) 
StringSeriaUzer.ctass.getName()); 
StringSeriaUzer.ctass.getName()); 
// create the producer 
KafkaProducer«String, String» producer nw ; 
// create a producer record 
ProducerRecord«String, String» record : 
topic: "first_topic", value: "hello 
// send data — asynchronous 
producer. send( record) ; 
// flu 
producer. flush( ) ; 
producer. close ( ) ; 

Machine generated alternative text:
38 
import org.apache . kafka . clients . 
9 
public class 
11 
129 
13 
14 
15 
16 
17 
18 
19 
21 
22 
23 
24 
25 
26 
28 
29 
31 
32 
33 
349 
359 
36 
38 
39 
41 
42 
43 
45 
46 
48 
public static void main(StringLJ args) 
final Logger logger LoggerFactory class) 
String bootstrapServers 
"127.ø.ø.1:9Ø92"; 
// create Producer properties 
o; 
Properties 
properties 
new Properties 
properties . setProperty(ProducerConfig.BOOTSTRAP SERVERS CONFIG, bootstrapServers) 
properties. setProperty(ProducerConfig.KEY SERIALIZER CLASS CONFIG, StringSeria1izer. class . getName 
properties. setProperty(ProducerConfig. VALUE_SERZALZZER CLASS CONFIG, StringSeria1izer. class . getName 
// create the producer 
new String) (properties) 
KafkaProducer(String, String) producer 
for (int i:ø; i(1Ø; i++ ) 
// create a producer record 
ProducerRecord(String, String) record 
new "first topic' 
// send data 
asynchronous 
producer. send(record, new Callback() 
"hello world 
+ Integer. tostring(i)); 
public void onComp1etion(RecordMetadata recordMetadata, Exception e) 
// executes every time a record is successfully sent or an exception is thrown 
if (e 
null) 
// the 
record was successfully sent 
logger. info( "Received new metadata. Xn" + 
+ recordMetadata . topic() 
" Topic:' 
" Partition : 
+ recordMetadata . partition 
+ recordMetadata. offset() + "Xn" + 
"Offset : 
+ recordMetadata . timestamp()); 
" Timestamp: 
else 
logger. "Error while producing", e 

**Imp Note: key always goes to same partition for fixed number of partition**