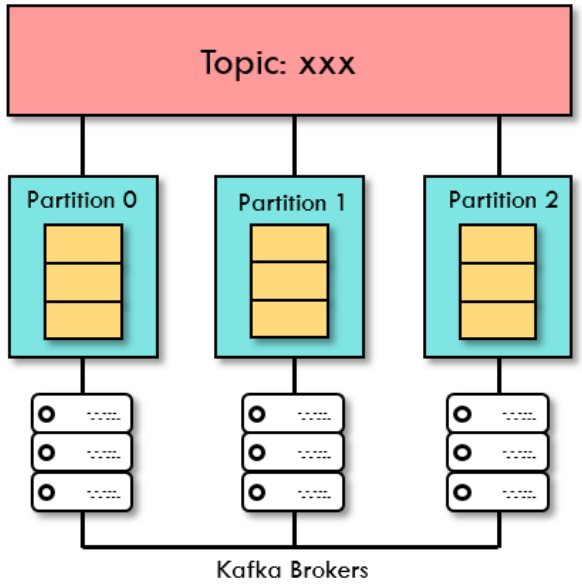
ak系列 – ak01
(Hands-on)

Apache Kafka: Hands-on Practice

Kafka Topic Operations

Kafka Topic operations
create, list, delete, describe



- Create
  - kafka-topics --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test
- List
  - kafka-topics --list --zookeeper localhost:2181
- Delete
  - kafka-topics --delete --zookeeper localhost:2181 --topic test
- Describe
  - kafka-topics --describe --zookeeper localhost:2181 --topic test

Kafka Topic: create

記得先用docker-compose把Kafka與Zookeeper啟動起來!!

```
$ kafka-topics
    --create
    --zookeeper localhost:2181
    --replication-factor 1 --partitions 1
    --topic test
```

```
root@kafka:/# kafka-topics --create \
>    --zookeeper zookeeper:2181 \
>    --replication-factor 1 \
>    --partitions 1 \
>    --topic test
Created topic "test".
```

在container裡頭的zookeeper服務的Hostname

Topic “test” is created!

Kafka Topic: list

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

```
root@kafka:/# kafka-topics --list --zookeeper zookeeper:2181
__confluent.support.metrics
test
```

The command will list out all existing “topics”

Kafka Topic: describe

```
$ kafka-topics
    --describe
    --zookeeper localhost:2181
    --topic test
```

```
root@kafka:/# kafka-topics --describe --zookeeper zookeeper:2181 --topic test
Topic:test      PartitionCount:1        ReplicationFactor:1     Configs:
        Topic: test     Partition: 0    Leader: 1       Replicas: 1     Isr: 1
```

The command tells many details of a "topic"

Kafka Topic: delete

```
$ bin/kafka-topics
    --delete --zookeeper localhost:2181
    --topic test
```

```
root@kafka:/# kafka-topics --delete --zookeeper zookeeper:2181 --topic test
Topic test is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
```

The command will has the "topic" marked for deletion!

Apache Kafka: Hands-on Practice

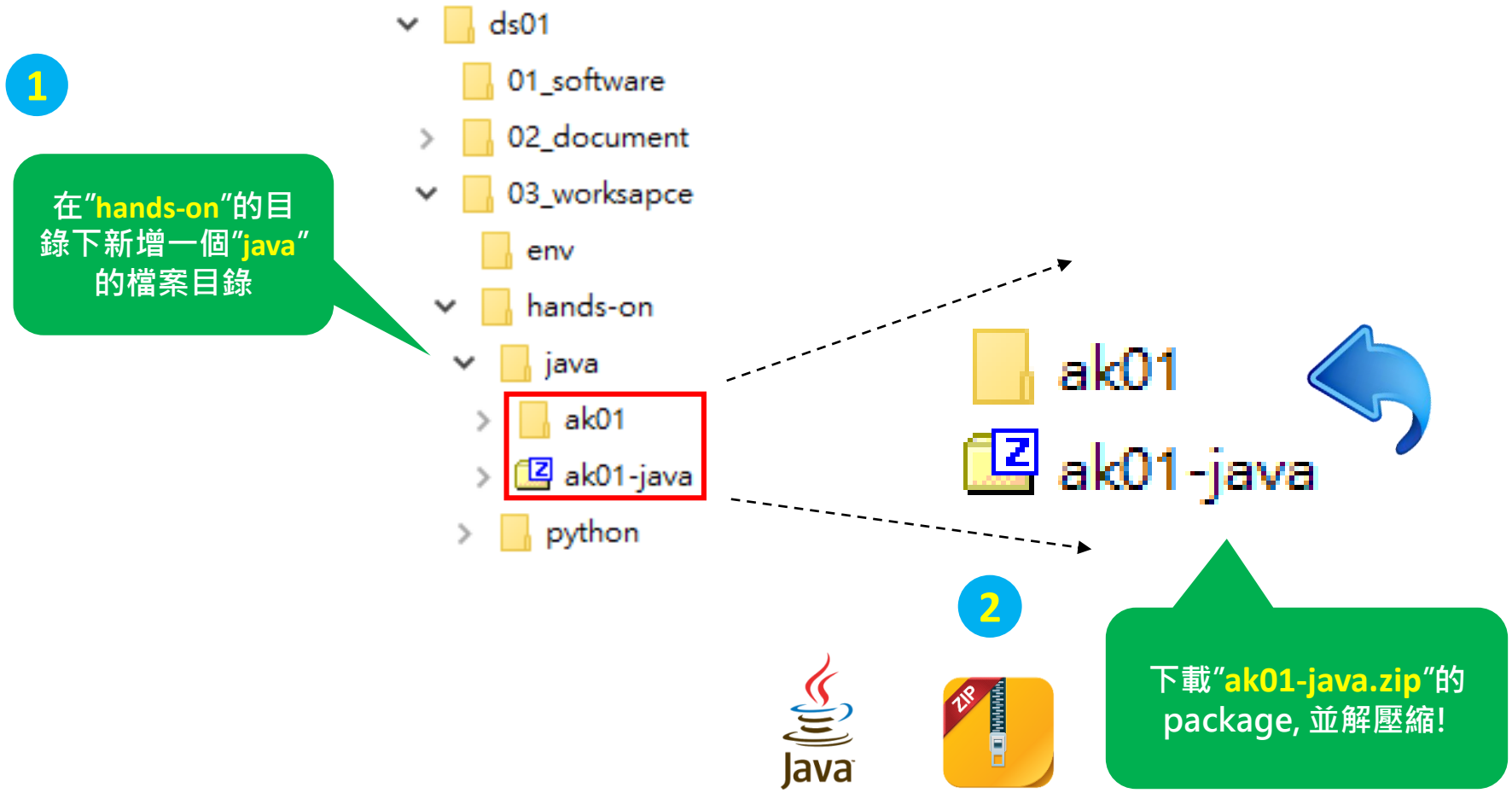
Java Producer & Consumer

Download Sample Java Project Code

- Get Java Demo source code for each training session

Open Demo Java Project using **IntelliJ**

Step.1

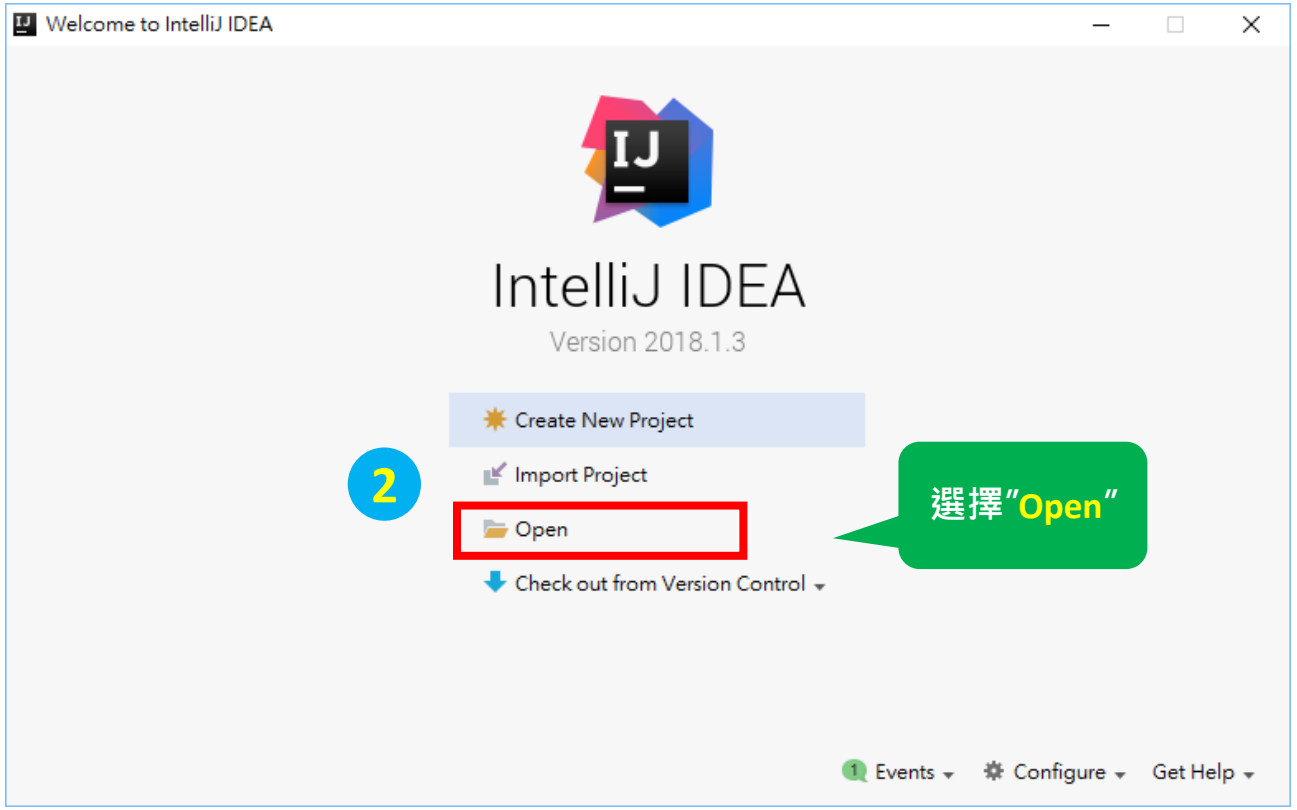


Open Demo Java Project using **IntelliJ**

Step.2

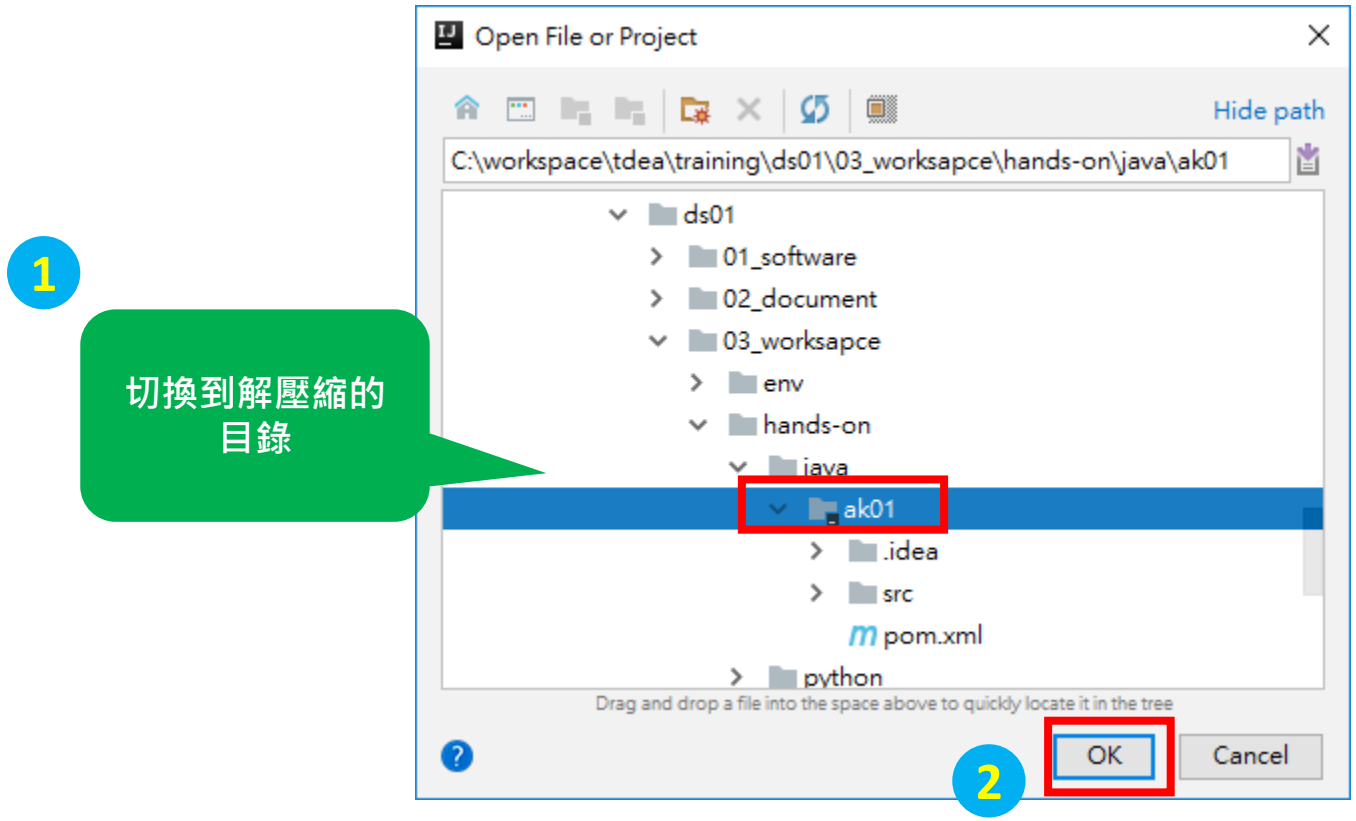
1

啟動IntelliJ IDEA



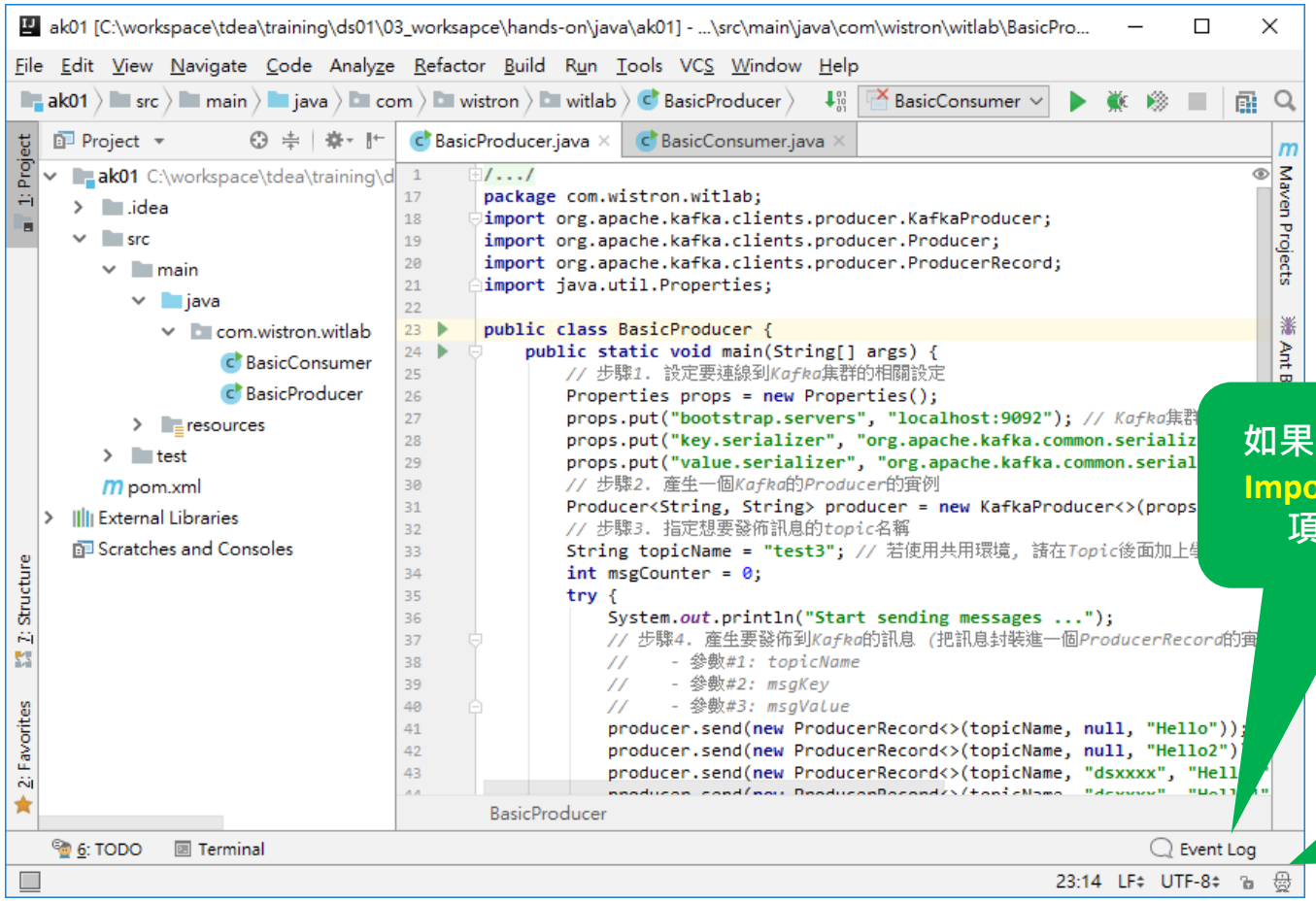
Open Demo Java Project using **IntelliJ**

Step.3



Open Demo Java Project using **IntelliJ**

Step.4



Kafka Java client: BasicProducer

```java
public class BasicProducer {
    public static void main(String[] args) {
        // 步驟1. 設定要連線到Kafka集群的相關設定
        Properties props = new Properties();
        // Kafka集群在那裡?
        props.put("bootstrap.servers", "localhost:9092");
        // 指定msgKey的序列化器
        props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        // 指定msgValue的序列化器
        props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        // 步驟2. 產生一個Kafka的Producer的實例
        Producer<String, String> producer = new KafkaProducer<>(props);
        // 步驟3. 指定想要發佈訊息的topic名稱
        String topicName = "test";
        int msgCounter = 0;
        try {
            System.out.println("Start sending messages ...");
            // 步驟4. 產生要發佈到Kafka的訊息 (把訊息封裝進一個ProducerRecord的實例中)
            //     - 參數#1: topicName
            //     - 參數#2: msgKey
            //     - 參數#3: msgValue
            producer.send(new ProducerRecord<>(topicName,  key: null,  value: "Hello"));
            producer.send(new ProducerRecord<>(topicName,  key: null,  value: "Hello2"));
            producer.send(new ProducerRecord<>(topicName,  key: "sg0000",  value: "Hello3"));
            producer.send(new ProducerRecord<>(topicName,  key: "sg0000",  value: "Hello4"));
            msgCounter+=4;
            System.out.println("Send " + msgCounter + " messages to Kafka");
        } catch (Exception e) {
            // 錯誤處理
            e.printStackTrace();
        }
        // 步驟5. 關掉Producer實例的連線
        producer.close();
        System.out.println("Message sending completed!");
    }
}
```

要發佈到那個 Kafka集群？

要發佈到那個 Topic？

訊息的Key與 Value是？

Kafka Java client: BasicConsumer

給一個 ConsumerGroup 名字

要訂閱那個 Topic的訊息？

在迴圈中持續拉取Topic的訊息及處理訊息

```java
public class BasicConsumer {
    public static void main(String[] args) {
        // 步驟1. 設定要連線到Kafka集群的相關設定
        Properties props = new Properties();
        // Kafka集群在那裡?
        props.put("bootstrap.servers", "localhost:9092");
        // ConsumerGroup的名稱
        props.put("group.id", "my-group");
        // 指定msgKey的反序列化器
        props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
        // 指定msgValue的反序列化器
        props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
        props.put("auto.offset.reset", "earliest"); // 是否從這個ConsumerGroup尚未讀取的partition/offset開始讀
        // 步驟2. 產生一個Kafka的Consumer的實例
        Consumer<String, String> consumer = new KafkaConsumer<>(props);
        // 步驟3. 指定想要訂閱訊息的topic名稱
        String topicName = "test";
        // 步驟4. 讓Consumer向Kafka集群訂閱指定的topic
        consumer.subscribe(Arrays.asList(topicName));
        // 步驟5. 持續的拉取Kafka有進來的訊息
        try {
            System.out.println("Start listen incoming messages ...");
            while (true) {
                // 請求Kafka把新的訊息吐出來
                ConsumerRecords<String, String> records = consumer.poll(Duration.ofMillis(1000));
                // 如果有任何新的訊息就會進到下面的迭代
                for (ConsumerRecord<String, String> record : records){
                    // ** 在這裡進行商業邏輯與訊息處理 **
                    // 取出相關的metadata
                    String topic = record.topic();
                    int partition = record.partition();
                    long offset = record.offset();
                    TimestampType timestampType = record.timestampType();
                    long timestamp = record.timestamp();
                    // 取出msgKey與msgValue
                    String msgKey = record.key();
                    String msgValue = record.value();
                    // 秀出metadata與msgKey & msgValue訊息
                    System.out.println(topic + "-" + partition + "-" + offset + " : (" + record.key() + ", " + record.value() + ")");
                }
            }
        } finally {
            // 步驟6. 如果收到結束程式的訊號時關掉Consumer實例的連線
            consumer.close();
            System.out.println("Stop listen incoming messages");
        }
    }
}
```

Apache Kafka: Hands-on Practice

Publish using Console Tools & Consume using Java Consumer

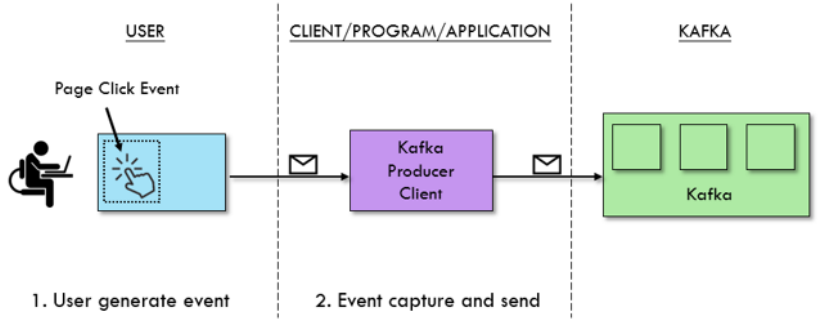
Create Kafka Topic for Practice

```
$ kafka-topics
    --create
    --zookeeper localhost:2181
    --replication-factor 1 --partitions 1
    --topic test2
```

```
root@kafka:/# kafka-topics --create --zookeeper zookeeper:2181 \
>    --replication-factor 1 --partitions 1 \
>    --topic test2
Created topic "test2".
```

產生一個用來練習用的topic

Publishing data to a topic
console producer



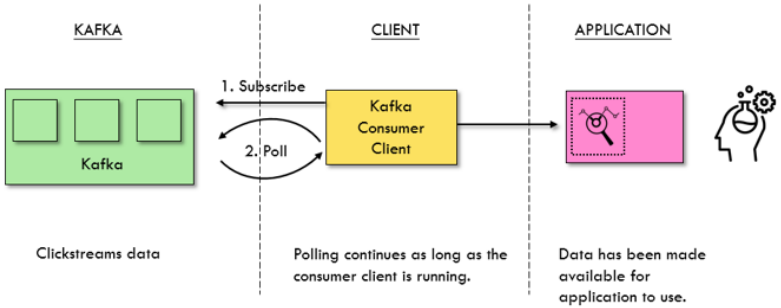

```
$ kafka-console-producer --broker-list localhost:9092 --topic test2
>Hello
>Hello2
```

```
root@kafka:/# kafka-console-producer --broker-list kafka:9092 --topic test2
>Hello
>Hello2
```

我們推送了兩個event的訊息

Consuming data from a topic

java consumer



Apache Kafka: Hands-on Practice

Publish using Java Consumer

/ Subscribe using Console Tools

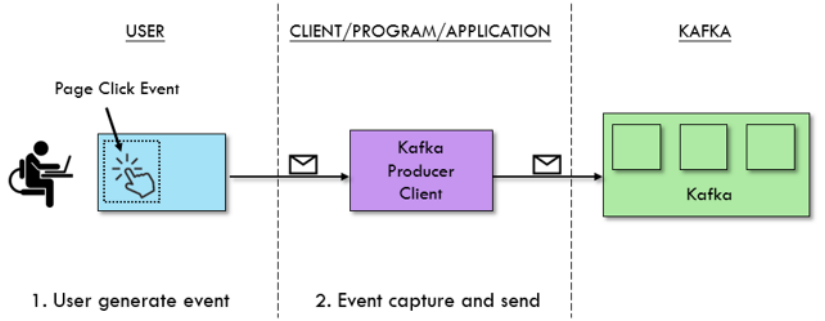
Create Kafka Topic for Practice

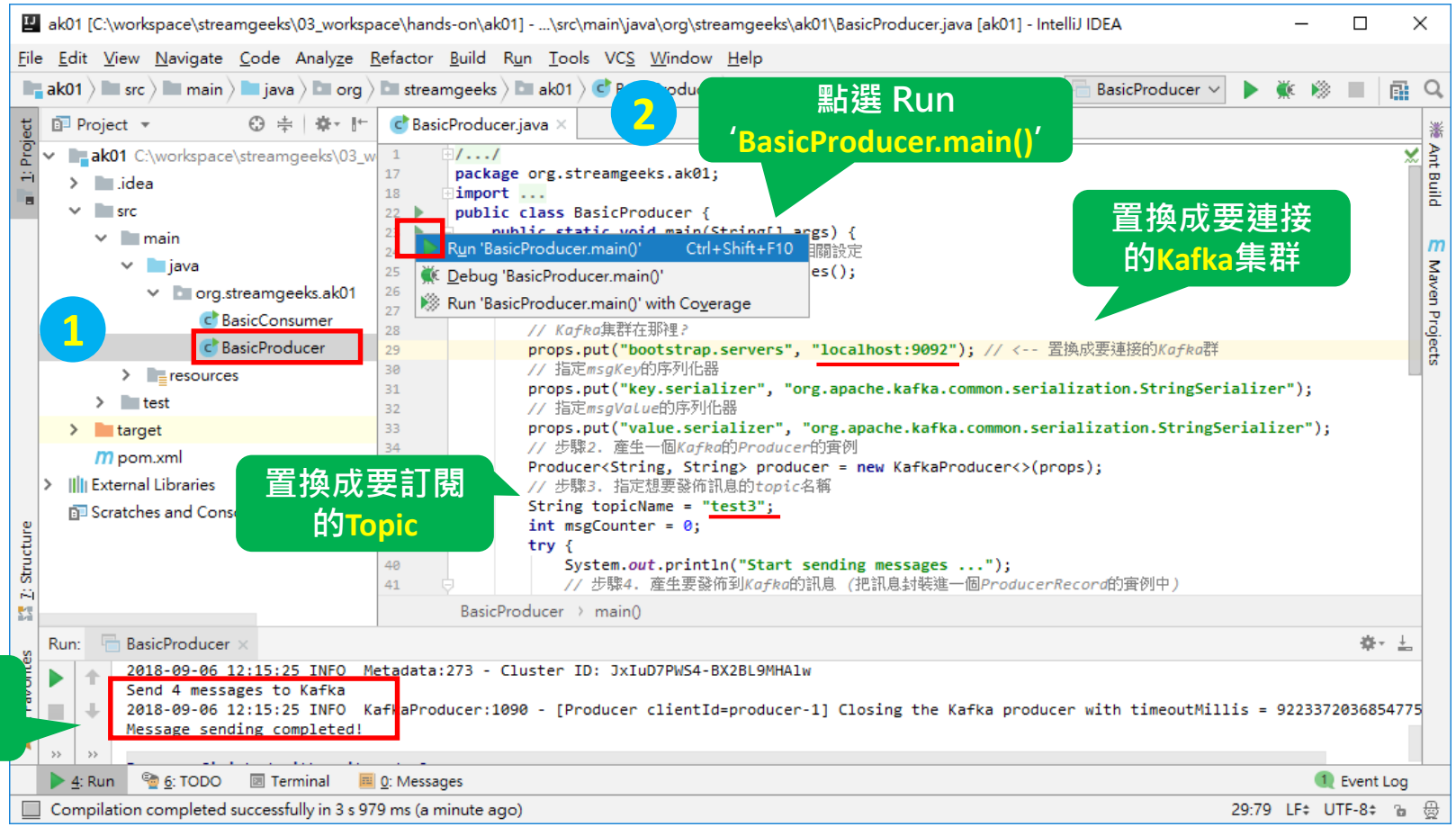
```
$ kafka-topics
    --create
    --zookeeper localhost:2181
    --replication-factor 1 --partitions 1
    --topic test3
```

```
root@kafka:/# kafka-topics --create --zookeeper zookeeper:2181 \
>    --replication-factor 1 --partitions 1 \
>    --topic test3
Created topic "test3".
```

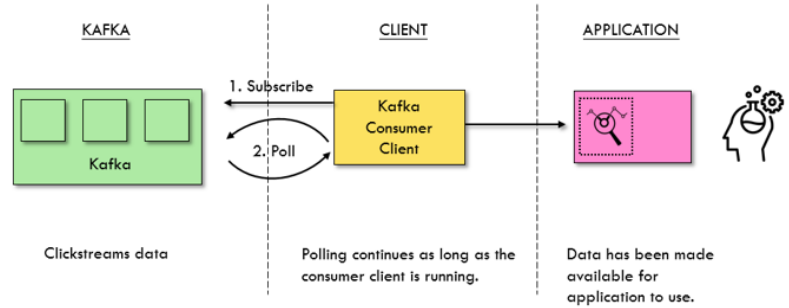
產生一個用來練習用的topic

Publish data to a topic
java Producer





Consuming data from a topic
console consumer




```
$ kafka-console-consumer  --bootstrap-server localhost:9092
    --topic test3
```

```
root@kafka:/# kafka-console-consumer \
>    --bootstrap-server kafka:9092 \
>    --from-beginning \
>    --topic test3
Hello
Hello2
Hello3
Hello4
```

我們從Kafka中收到發佈的訊息

36