緯創IT先進技術實驗室

witlab

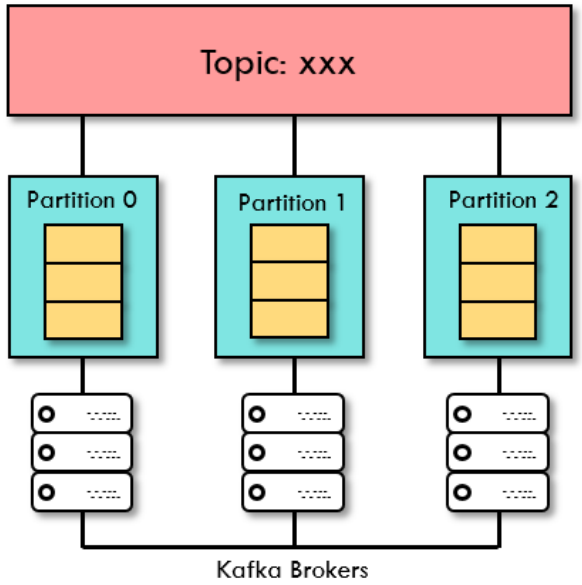
ak系列 – ak01

(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

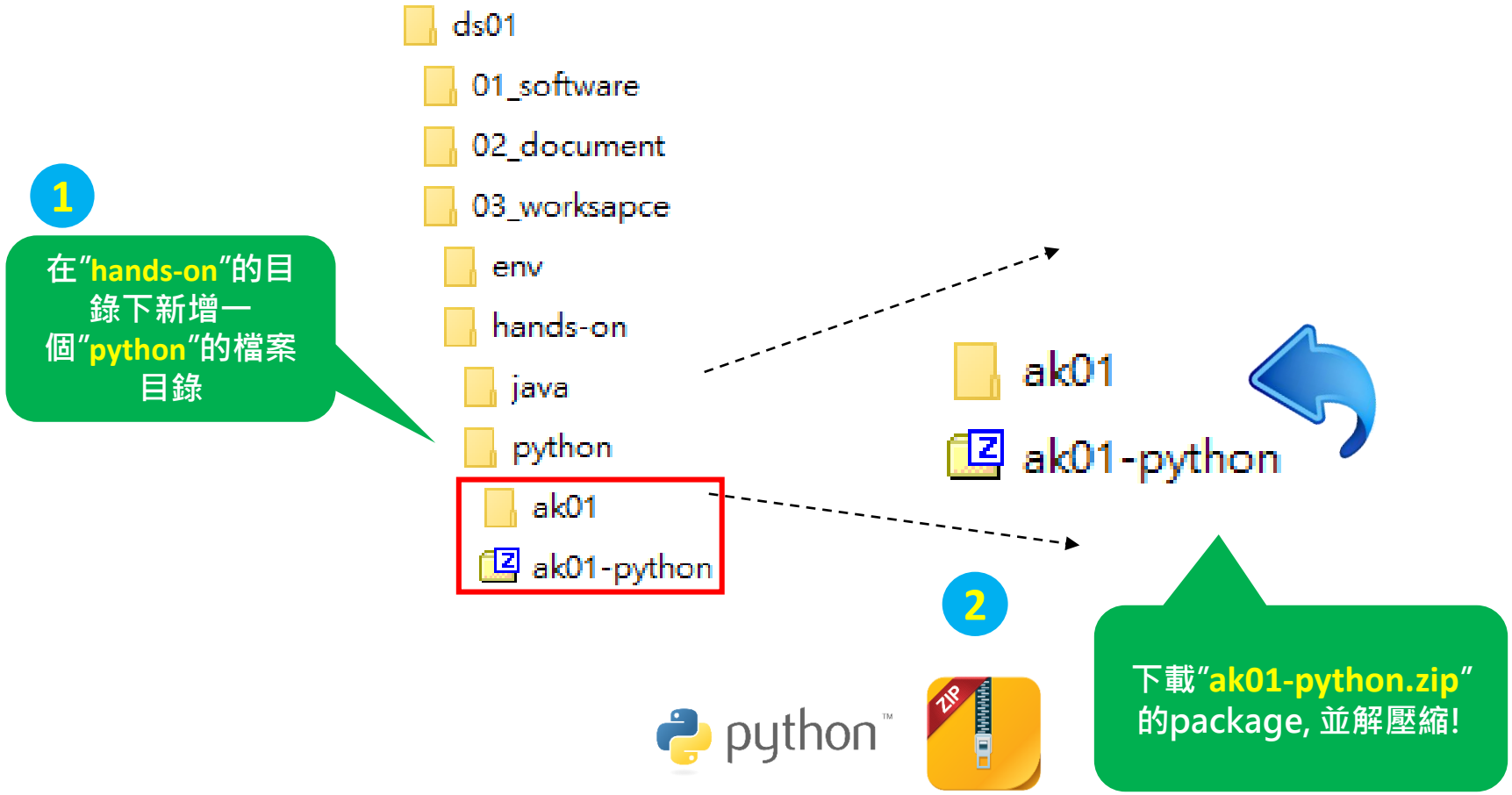
Java Producer & Consumer

Download Sample Python Project Code

- Get Demo source code for each training session

Open Demo Project using **PyCharm**

Step.1

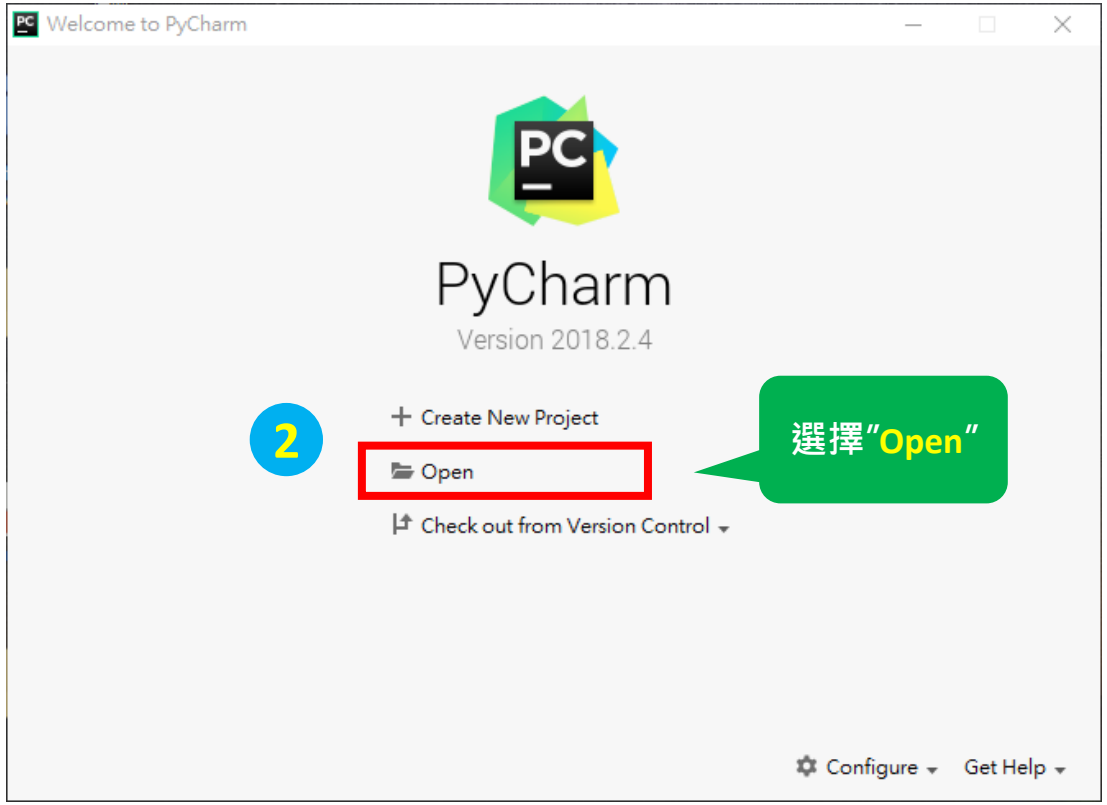


Open Demo Java Project using **IntelliJ**

Step.2

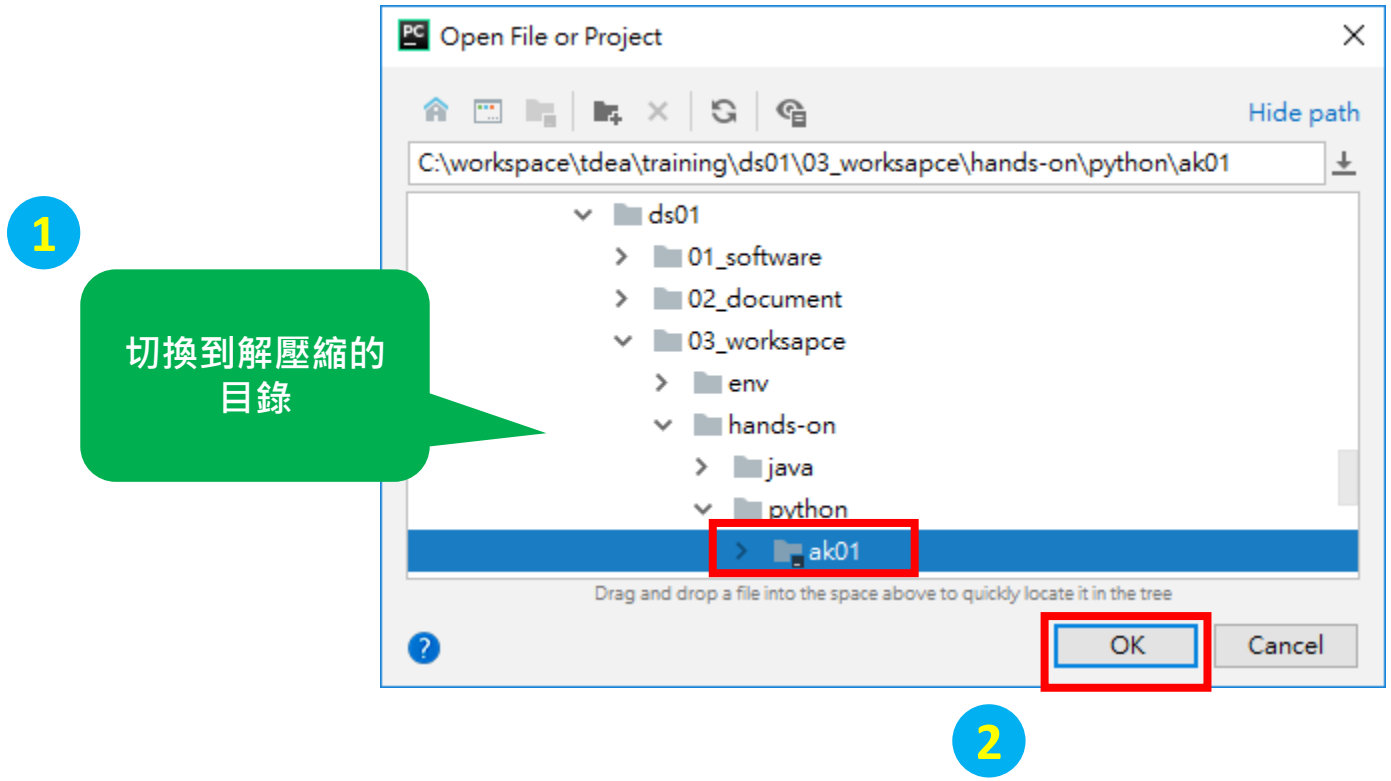
1

啟動PyCharm



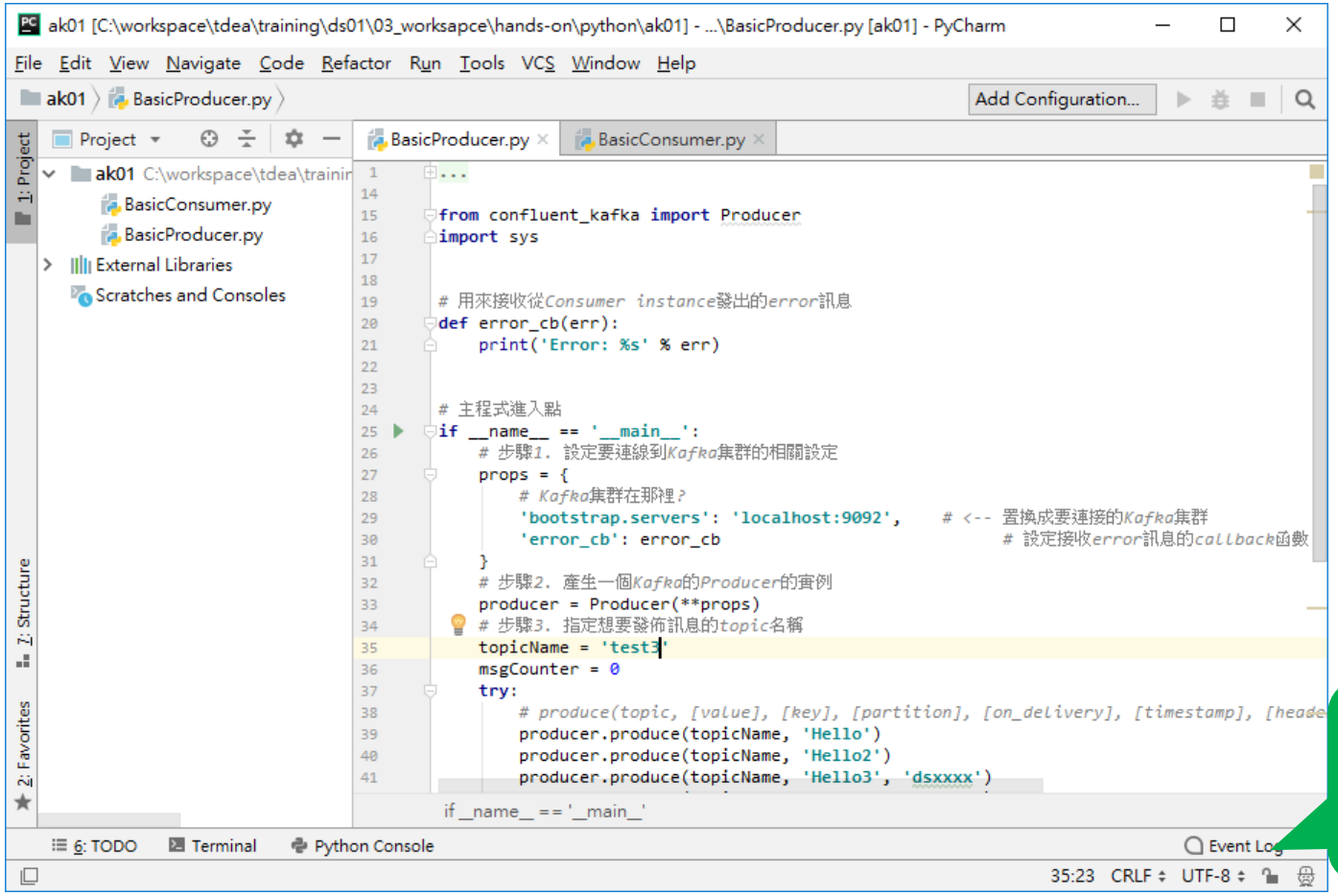
Open Demo Project using **PyCharm**

Step.3



Open Demo Project using **PyCharm**

Step.4



Kafka Python client: BasicProducer

```python
from confluent_kafka import Producer
import sys


# 用來接收從Consumer instance發出的error訊息
def error_cb(err):
    print('Error: %s' % err)


# 主程式進入點
if __name__ == '__main__':
    # 步驟1. 設定要連線到Kafka集群的相關設定
    props = {
        # Kafka集群在那裡?
        'bootstrap.servers': 'localhost:9092',    # 置換成要連接的Kafka集群
        'error_cb': error_cb                      # 設定接收error訊息的callback函數
    }
    # 步驟2. 產生一個Kafka的Producer的實例
    producer = Producer(**props)
    # 步驟3. 指定想要發佈訊息的topic名稱
    topicName = 'test3'
    msgCounter = 0
    try:
        # produce(topic, [value], [key], [partition], [on_delivery], [timestamp], [headers])
        producer.produce(topicName, 'Hello')
        producer.produce(topicName, 'Hello2')
        producer.produce(topicName, 'Hello3', 'dsxxxx')
        producer.produce(topicName, 'Hello4', 'dsxxxx')

        producer.poll(0)  # 呼叫poll來讓client程式去檢查內部的Buffer, 並觸發callback
        msgCounter+=4
        print('Send ' + str(msgCounter) + ' messages to Kafka')
    except BufferError as e:
        # 錯誤處理
        sys.stderr.write('%% Local producer queue is full (%d messages awaiting delivery): try again\n' % len(producer))
    except Exception as e:
        print(e)
    # 步驟5. 確認所在Buffer的訊息都已經送出去給Kafka了
    producer.flush()
```

要發佈到那個 Kafka集群？

要發佈到那個 Topic？

訊息的Key與 Value是？

Kafka Python 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

```
if __name__ == '__main__':
    # 步驟1.設定要連線到Kafka集群的相關設定
    # Consumer configuration
    # See https://github.com/edenhill/librdkafka/blob/master/CONFIGURATION.md
    props = {
        'bootstrap.servers': 'localhost:9092',          # Kafka集群在那裡? (置換成要連接的Kafka集群)
        'group.id': 'STUDENT_ID',                       # ConsumerGroup的名稱 (置換成你/妳的學員ID)
        'auto.offset.reset': 'earliest',                # Offset從最前面開始
        'session.timeout.ms': 6000,
        'error_cb': error_cb                            # 設定接收error訊息的callback函數
    }
    # 步驟2. 產生一個Kafka的Consumer的實例
    consumer = Consumer(props)
    # 步驟3. 指定想要訂閱訊息的topic名稱
    topicName = "test";
    # 步驟4. 讓Consumer向Kafka集群訂閱指定的topic
    consumer.subscribe([topicName])
    # 步驟5. 持續的拉取Kafka有進來的訊息
    try:
        while True:
            # 請求Kafka把新的訊息吐出來
            records = consumer.consume(num_messages=500, timeout=1.0)  # 批次讀取
            if records is None:
                continue
            for record in records:
                # 檢查是否有錯誤
                if record is None:
                    continue
                if record.error():
                    # Error or event
                    if record.error().code() == KafkaError._PARTITION_EOF:
                        # End of partition event
                        sys.stderr.write('%% %s [%d] reached end at offset %d\n' %
                                         (record.topic(), record.partition(), record.offset()))
                    else:
                        # Error
                        raise KafkaException(record.error())
                else:
                    # ** 在這裡進行商業邏輯與訊息處理 **
                    # 取出相關的metadata
                    topic = record.topic()
                    partition = record.partition()
                    offset = record.offset()
                    timestamp = record.timestamp()
                    # 取出msgKey與msgValue
                    msgKey = try_decode_utf8(record.key())
                    msgValue = try_decode_utf8(record.value())
                    # 秀出metadata與msgKey & msgValue訊息
                    print('%s-%d-%d : (%s , %s)' % (topic, partition, offset, msgKey, msgValue))
```

Apache Kafka: Hands-on Practice

Publish using Console Tools & Consume using Python Consumer

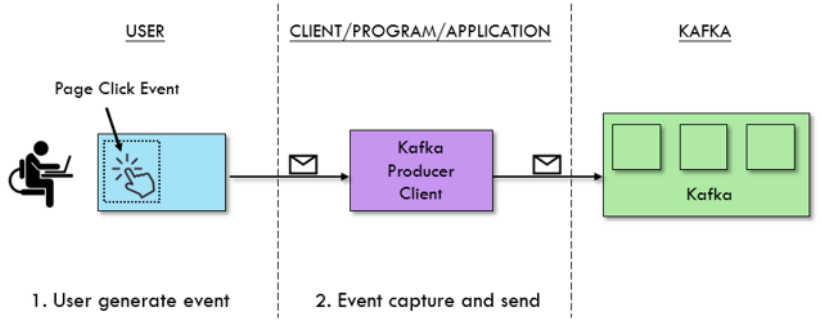
Create Kafka Topic for Practice

```
$ 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

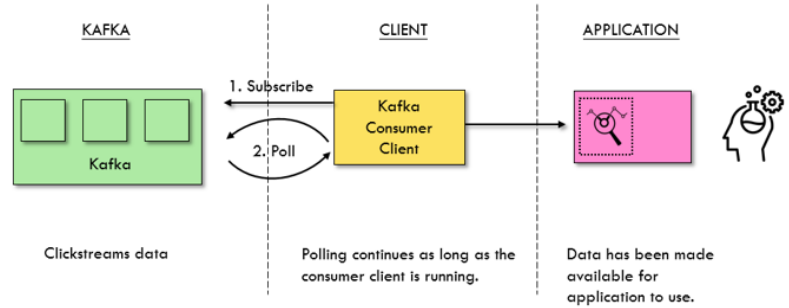



```
$ kafka-console-producer --broker-list localhost:9092 --topic test2
>Hello
>Hello2
```

```
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
python consumer



1

2

點選 Run 'BasicConsumer.main()'

ak01 [C:\workspace\tdea\training\ds01\03_w... cConsumer.py [ak01] - PyCharm

File Edit View Navigate Code Refactor

ak01 > BasicConsumer.py

BasicConsumer

1: Project

Proj...

ak01 C:\workspace\tdea\

BasicConsumer.py

BasicProducer.py

External Libraries

Scratches and Consoles

BasicConsumer.py

```
if __name__ == '__main__':
    # 步驟1.設定要連線到Kafka集群的相關設定
    # Consumer configuration
    # See https://github.com/edenhill/librd...ATION.md
    props = {
        'bootstrap.servers': 'localhost:9092',          # Kafka集群在那裡? (置換成要連接的
        'group.id': 'STUDENT_ID',                       # ConsumerGroup的名稱 (置換成你/妳
        'auto.offset.reset': 'earliest',                # Offset從最前面開始
        'session.timeout.ms': 6000,
        'error_cb': error_cb                            # error訊息的callback函數
    }
    # 步驟2. 產生一個Kafka的Consumer的實例
    consumer = Consumer(props)
    # 步驟3. 指定想要訂閱訊息的topic名稱
    topicName = "test2";
    # 步驟4. 讓Consumer向Kafka集群訂閱指定的topic
    consumer.subscribe([topicName])
    # 步驟5. 持續的拉取Kafka有進來的訊息
    try:
        if __name__ == '__main__'
```

置換成要連接的Kafka集群

置換成你/妳的學員ID

置換成要訂閱的Topic

3

Run: BasicConsumer

```
C:\Users\8703147\AppData\Local\Continuum\anaconda3\envs\tdea\python.exe C:/workspace/tdea/training/ds01/03_worksa
% test2 [0] reached end at offset 6
test2-0-4 : (None , Hello)
test2-0-5 : (None , Hello2)
```

會看到已發佈的訊息!

2: Favorites 7: Structure

4: Run 6: TODO Terminal Python Console Event Log

48:36 CRLF UTF-8

Apache Kafka: Hands-on Practice

Publish using Python Producer

/ Subscribe using Console Tools

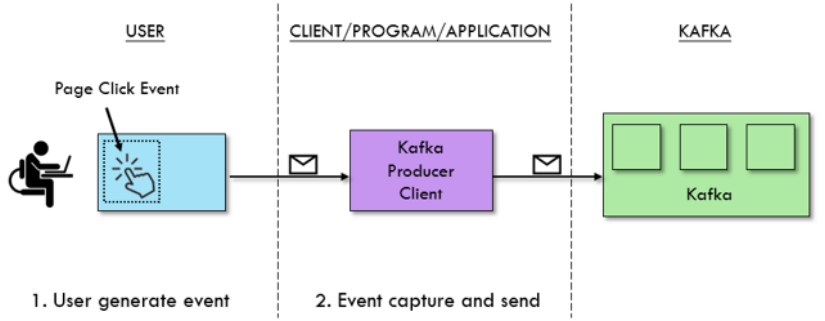
Create Kafka Topic for Practice

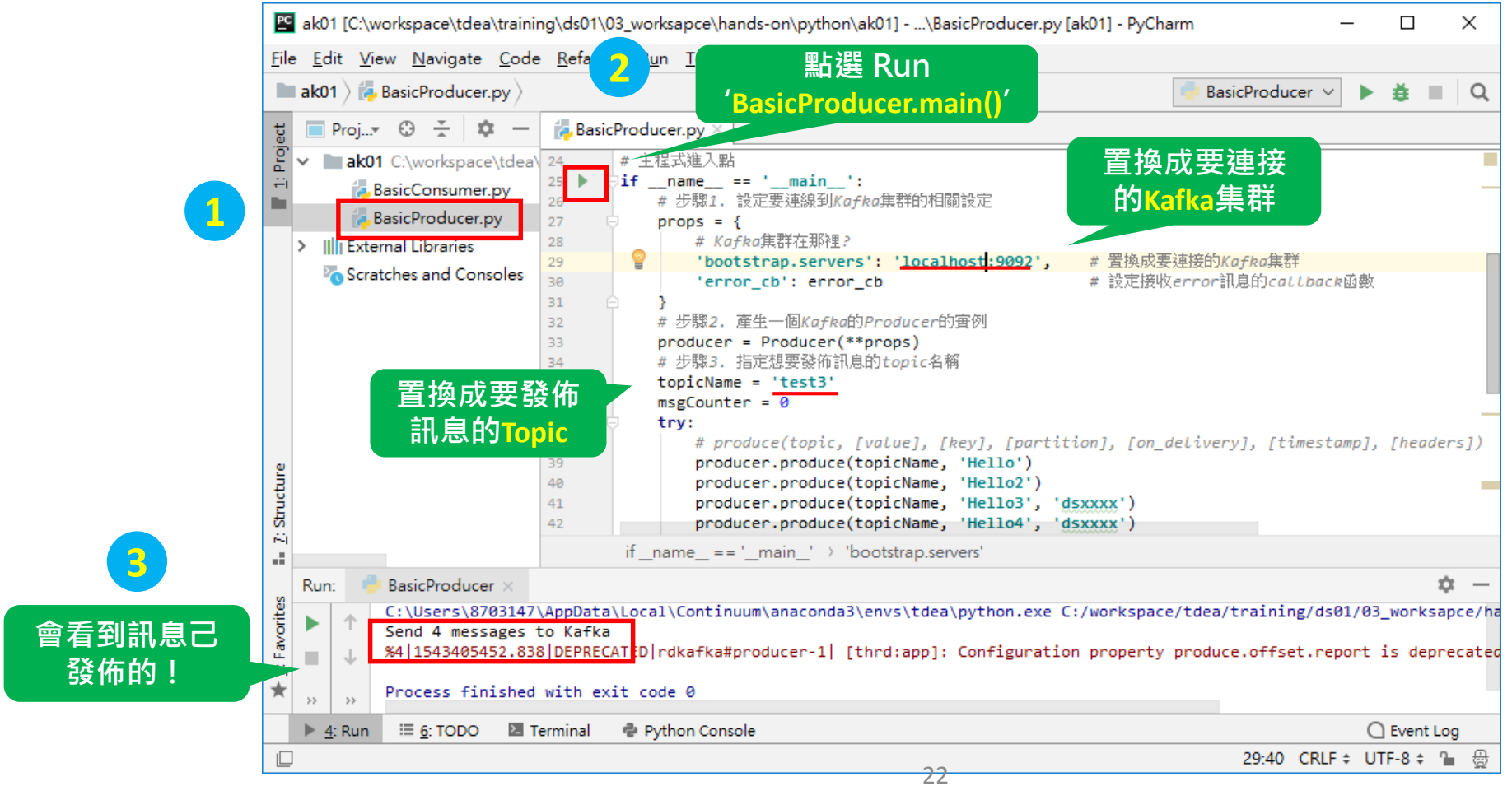
```
$ kafka-topics
    --create
    --zookeeper localhost:2181
    --replication-factor 1 --partitions 1
    --topic test3
```

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

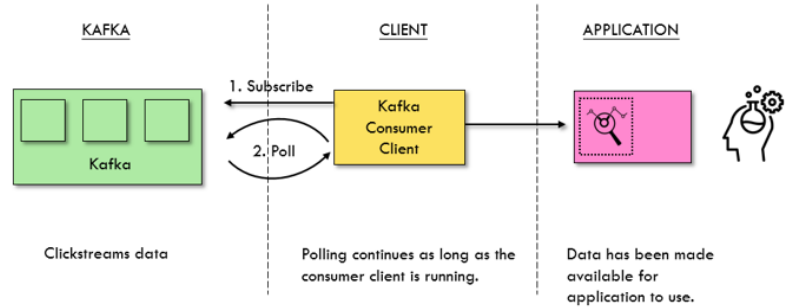
產生一個用來練習用的topic

Publish data to a topic
python 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