緯創IT先進技術實驗室

witlab

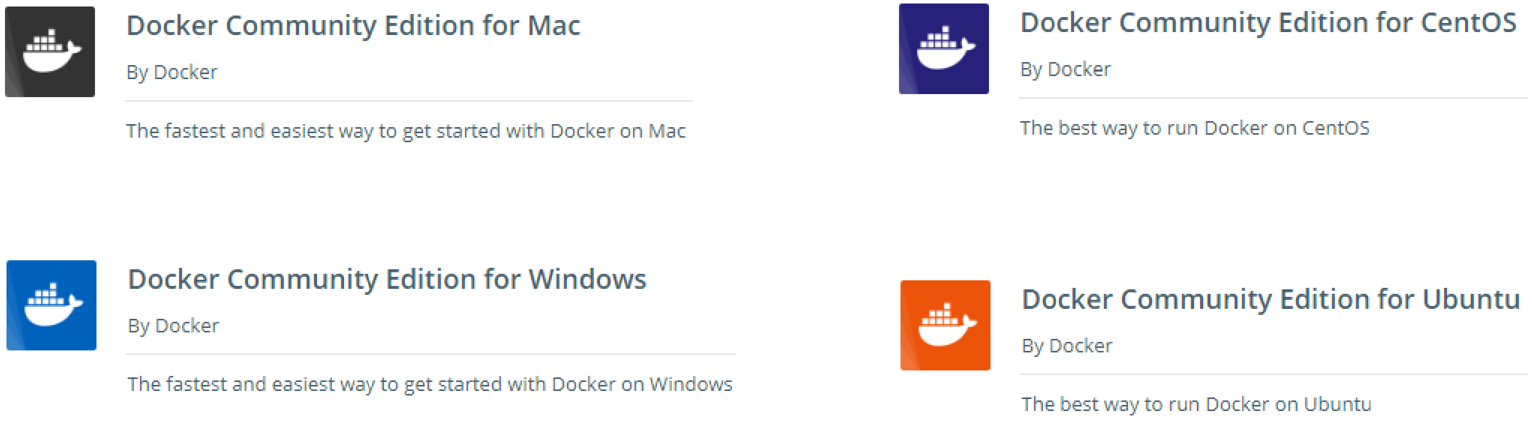
ak系列 – Environment

(Docker Engine)

Install Docker

Step. 1 – Install Docker-CE

- Go to Docker official web site & base on O.S to download and install "Docker Community Edition"
  - https://store.docker.com/search?type=edition&offering=community



Docker Community Edition for Mac
By Docker
The fastest and easiest way to get started with Docker on Mac

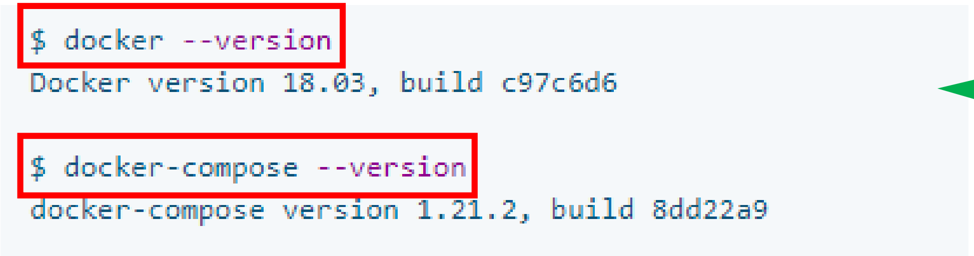
Docker Community Edition for CentOS
By Docker
The best way to run Docker on CentOS

Docker Community Edition for Windows
By Docker
The fastest and easiest way to get started with Docker on Windows

Docker Community Edition for Ubuntu
By Docker
The best way to run Docker on Ubuntu

Step. 2 – Verify Docker readiness

- Check version

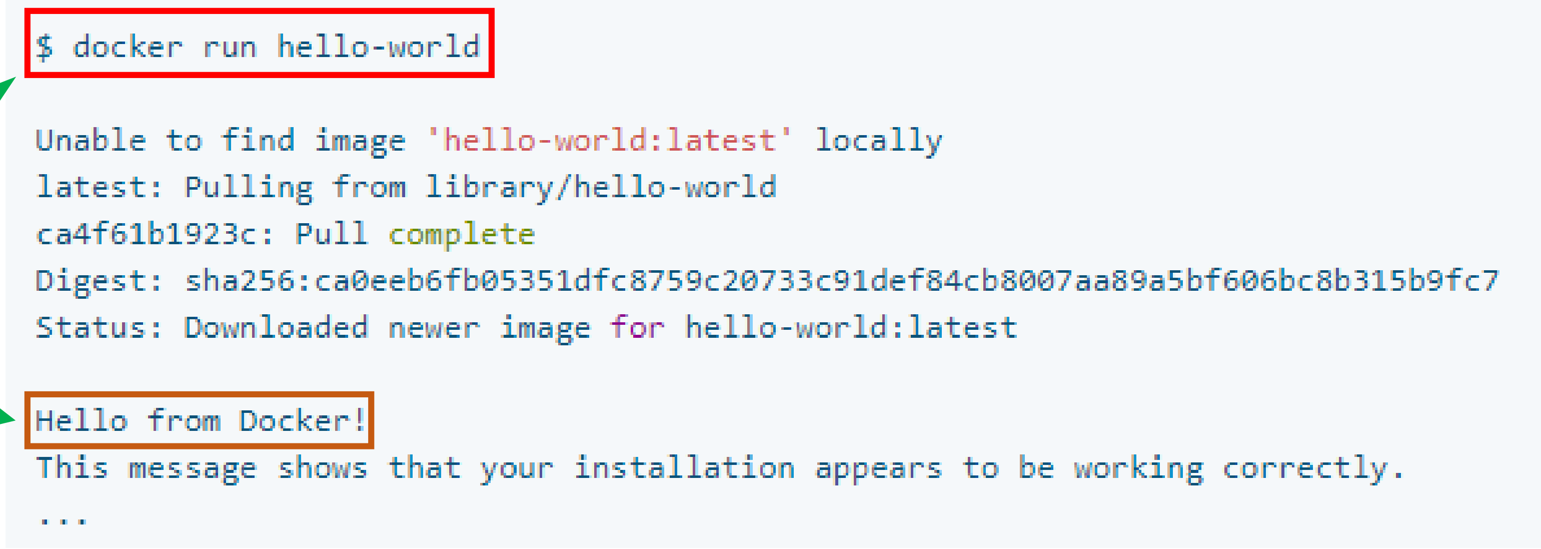


```
$ docker --version
Docker version 18.03, build c97c6d6

$ docker-compose --version
docker-compose version 1.21.2, build 8dd22a9
```

檢查看是否
有正確安裝！

- Explore the application



```
$ docker run hello-world

Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
ca4f61b1923c: Pull complete
Digest: sha256:ca0eeb6fb05351dfc8759c20733c91def84cb8007aa89a5bf606bc8b315b9fc7
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
...
```

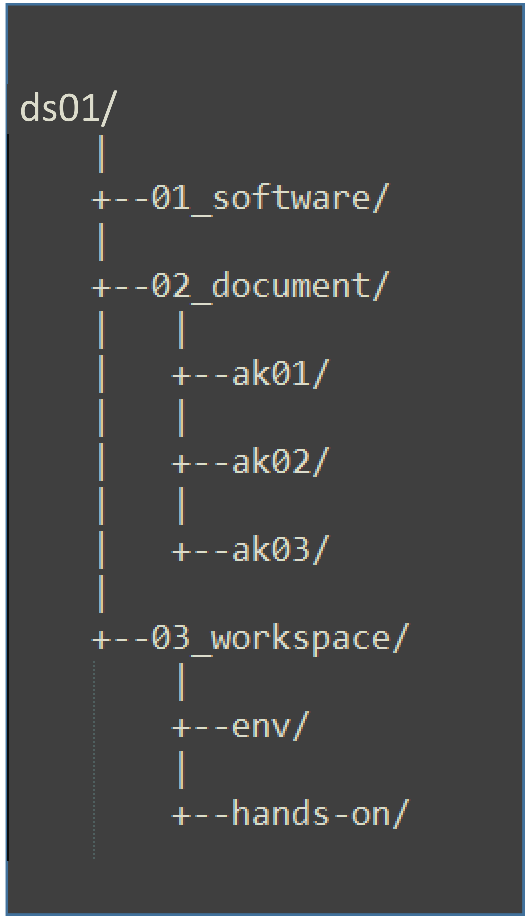
跑一個hello-world的container來驗證

要看到這個訊息才代表Docker是正常運行喔!

Create Training File Directory

Training File Directory

- Create recommended file directory tree
  - **01_software**
    - Use to put necessary softwares
  - **02_document**
    - Use to put each training sessions documents, scripts or sample codes
  - **03_workspace**
    - **env**
      - Put docker-compose file for environment setup
    - **hands-on**
      - Sample demo codes



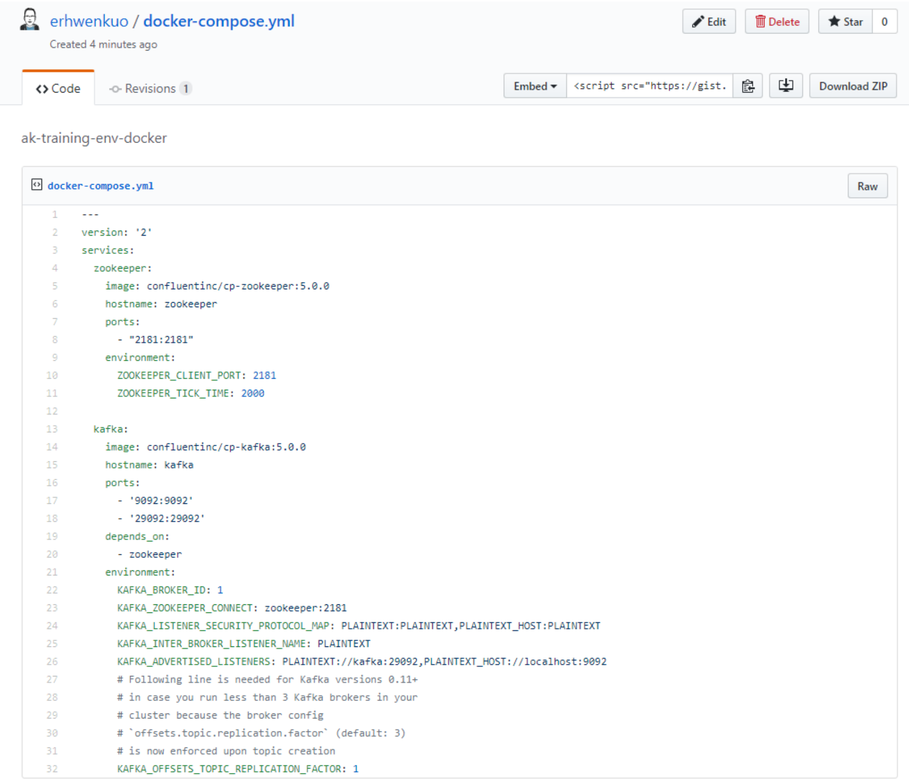
```
ds01/
     |
     +--01_software/
     |
     +--02_document/
     |    |
     |    +--ak01/
     |    |
     |    +--ak02/
     |    |
     |    +--ak03/
     |
     +--03_workspace/
          |
          +--env/
          |
          +--hands-on/
```

Setup Kafka & Zookeeper

using Docker Compose

Step. 1 – Download Environment Docker-Compose Setting File

- Use browser to download “**docker-compose.yml**” from gist
  - https://gist.github.com/erhwenkuo/7b72c2464419ab5806a7332005ae41e8
- Extract “**docker-compose.yml**” to specific folder, for example:
  - ~/ds01/03_workspace/env/

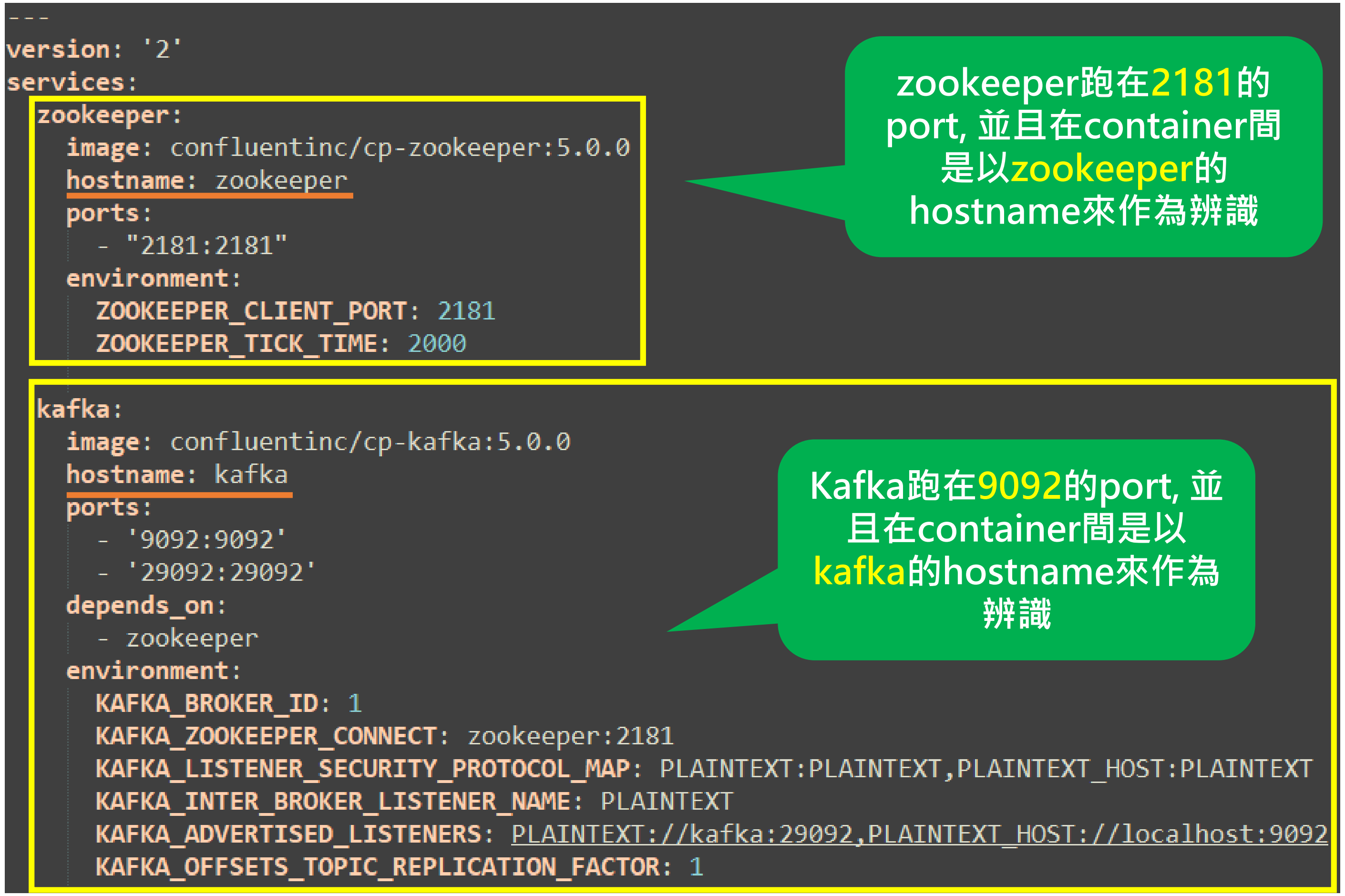


```
---
version: '2'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:5.0.0
    hostname: zookeeper
    ports:
      - "2181:2181"
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000

  kafka:
    image: confluentinc/cp-kafka:5.0.0
    hostname: kafka
    ports:
      - '9092:9092'
      - '29092:29092'
    depends_on:
      - zookeeper
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:29092,PLAINTEXT_HOST://localhost:9092
      # Following line is needed for Kafka versions 0.11+
      # in case you run less than 3 Kafka brokers in your
      # cluster because the broker config
      # `offsets.topic.replication.factor` (default: 3)
      # is now enforced upon topic creation
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
```

https://gist.github.com/erhwenkuo/7b72c2464419ab5806a7332005ae41e8

Step. 2 – Modify docker-compose.yml



```
---
version: '2'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:5.0.0
    hostname: zookeeper
    ports:
      - "2181:2181"
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000

  kafka:
    image: confluentinc/cp-kafka:5.0.0
    hostname: kafka
    ports:
      - '9092:9092'
      - '29092:29092'
    depends_on:
      - zookeeper
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:29092,PLAINTEXT_HOST://localhost:9092
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
```

zookeeper跑在2181的port, 並且在container間是以zookeeper的hostname來作為辨識

Kafka跑在9092的port, 並且在container間是以kafka的hostname來作為辨識

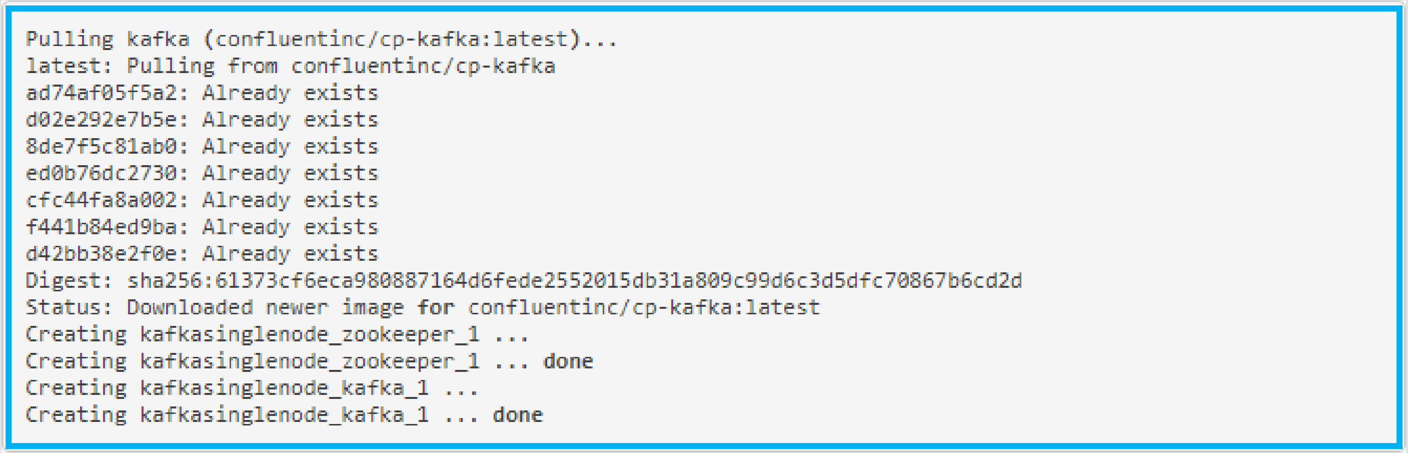
Step. 2 – Start Zookeeper & Kafka

- Run below command from the directory that contains the “**docker-compose.yml**” file.

```
$ docker-compose up -d
```

記得要切換到放置docker-compose.yml的目錄底下, 目錄名為：env

- You should see the following:



```
Pulling kafka (confluentinc/cp-kafka:latest)...
latest: Pulling from confluentinc/cp-kafka
ad74af05f5a2: Already exists
d02e292e7b5e: Already exists
8de7f5c81ab0: Already exists
ed0b76dc2730: Already exists
cfc44fa8a002: Already exists
f441b84ed9ba: Already exists
d42bb38e2f0e: Already exists
Digest: sha256:61373cf6eca980887164d6fede2552015db31a809c99d6c3d5dfc70867b6cd2d
Status: Downloaded newer image for confluentinc/cp-kafka:latest
Creating kafkasinglenode_zookeeper_1 ...
Creating kafkasinglenode_zookeeper_1 ... done
Creating kafkasinglenode_kafka_1 ...
Creating kafkasinglenode_kafka_1 ... done
```

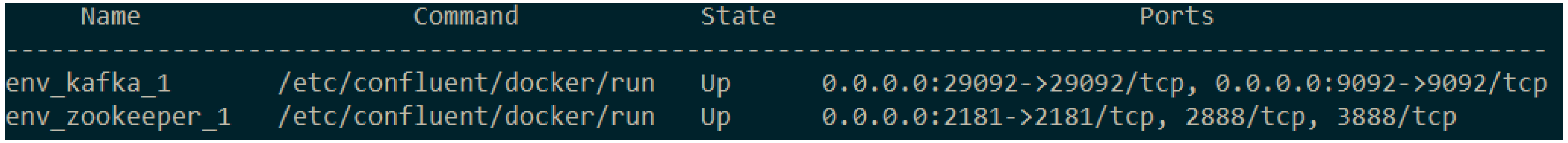
第一次啟動的時候, 會花一點時間從網路上下載Docker的images檔案

Step. 3 – Verify Zookeeper & Kafka services

- Run below command

```
$ docker-compose ps
```

- You should see the following:



```
     Name                     Command             State                     Ports
------------------------------------------------------------------------------------------------------
env_kafka_1         /etc/confluent/docker/run    Up      0.0.0.0:29092->29092/tcp, 0.0.0.0:9092->9092/tcp
env_zookeeper_1     /etc/confluent/docker/run    Up      0.0.0.0:2181->2181/tcp, 2888/tcp, 3888/tcp
```

如果正確啟動, 會看到本機上會有兩個container在跑

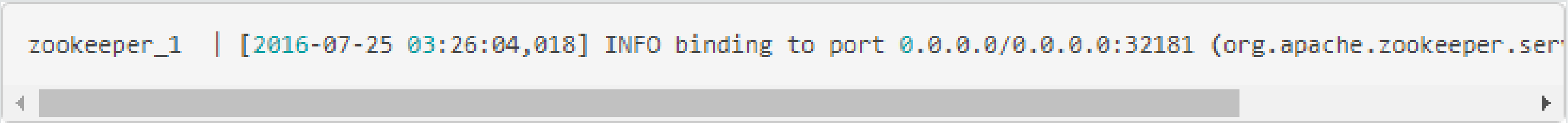
Step. 4 – Verify Zookeeper is healthy

- Run below command (non-windows)

```
$ docker-compose logs zookeeper | grep  -i binding
```

這個command只
有在非Windows
上才跑的出來

- You should see the following:



```
zookeeper_1   | [2016-07-25 03:26:04,018] INFO binding to port 0.0.0.0/0.0.0.0:32181 (org.apache.zookeeper.ser
```

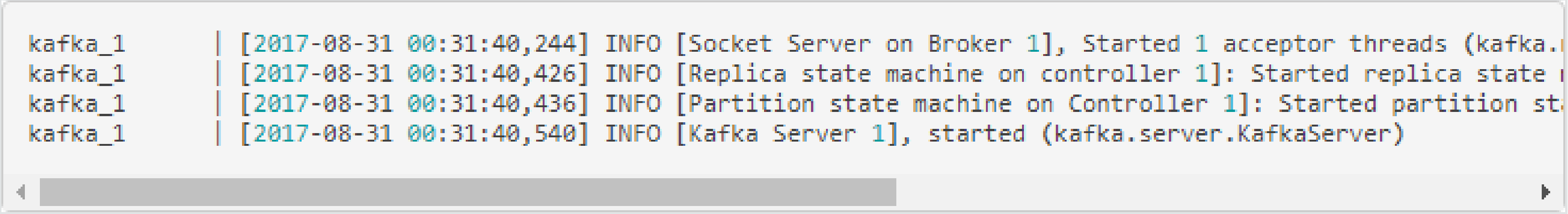
Step. 5 – Verify Kafka is healthy

- Run below command (Non-Windows)

```
$ docker-compose logs kafka | grep  -i started
```

這個command只
有在非Windows
上才跑的出來

- You should see the following:



```
kafka_1      | [2017-08-31 00:31:40,244] INFO [Socket Server on Broker 1], Started 1 acceptor threads (kafka.
kafka_1      | [2017-08-31 00:31:40,426] INFO [Replica state machine on controller 1]: Started replica state
kafka_1      | [2017-08-31 00:31:40,436] INFO [Partition state machine on Controller 1]: Started partition sta
kafka_1      | [2017-08-31 00:31:40,540] INFO [Kafka Server 1], started (kafka.server.KafkaServer)
```

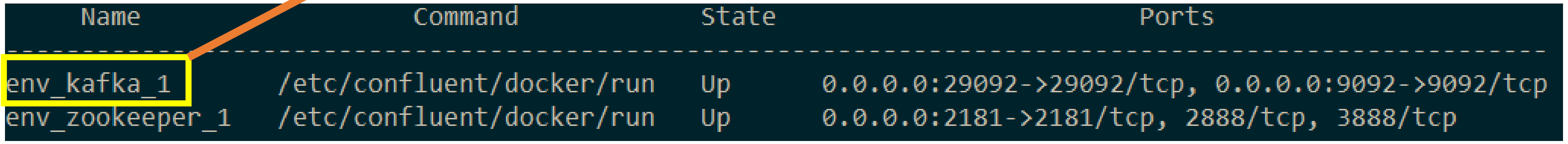
Test Kafka & Zookeeper Env.

using Docker Compose

Get into Docker container

- Run below command

```
$ docker exec -it env_kafka_1 bash
```



```
      Name                     Command             State                         Ports
------------------------------------------------------------------------------------------------------
env_kafka_1       /etc/confluent/docker/run    Up      0.0.0.0:29092->29092/tcp, 0.0.0.0:9092->9092/tcp
env_zookeeper_1   /etc/confluent/docker/run    Up      0.0.0.0:2181->2181/tcp, 2888/tcp, 3888/tcp
```

使用docker container的名稱來登入到container中

```
root@kafka:/#
```

Step. 1 – Create a topic

- Run below command (inside-container)

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

由於這個container的基底是Linux的OS, 所以必需使用 "\"來進行換行

```
Created topic "test".
```

Step. 2 – Publish data

- Run below command (inside-container)

```
$ kafka-console-producer \
   --broker-list kafka:9092 \
   --topic test
```

我們使用“kafka”
(黃色)是因為在
docker-compose時
我們設定
的”hostname”
(見第7頁)

```
>hello
>hello2
```

Step. 3 – Subscribe data

- Run below command (inside-container)

```
$ kafka-console-consumer \
   --bootstrap-server kafka:9092 \
   --topic test \
   --from-beginning
```

我們使用“kafka”是因為在docker-compose時我們設定的”hostname”(見第7頁)

```
hello
hello2
```

Shutdown Kafka & Zookeeper

using Docker Compose

Step. 1 – Shutdown Zookeeper & Kafka

- Run below command from the directory that contains the “**docker-compose.yml**” file.

```
$ docker-compose stop
```

記得要切換到放置docker-compose.yml的目錄底下,
建議目錄名為：env

- You should see the following:

```
Stopping env_kafka_1            ... done
Stopping env_zookeeper_1     ... done
```

Step. 2 – Start exiting Zookeeper & Kafka

- Run below command from the directory that contains the “**docker-compose.yml**” file.

```
$ docker-compose start
```

記得要切換到放置docker-compose.yml的目錄底下,建議目錄名為：env

- You should see the following:

```
Starting env_kafka_1          ... done
Starting env_zookeeper_1     ... done
```

這個指令是把之前暫時停掉的Containers再重新跑起來（以前的資料都還在)。

Step. 3 – Remove Zookeeper & Kafka container/data

- Run below command from the directory that contains the “**docker-compose.yml**” file.

```
$ docker-compose down
```

這個指令會把
containers的資料
都清除掉!

- You should see the following:

```
Stopping  env_kafka_1            ... done
Stopping  env_zookeeper_1    ... done
Removing env_kafka_1             ... done
Removing env_zookeeper_1     ... done
```

The End