緯創IT先進技術實驗室

ak系列 – Environment

(Docker Engine)

Install Docker (Centos 7)

Step. 1 – Install docker engine (Centos 7)

- su root

```
$ su root
```

- Uninstall old versions

```
$ yum remove docker docker-client docker-client-latest docker-common docker-latest docker-latest-
  logrotate docker-logrotate docker-engine
```

- Install using the repository

```
$ yum install -y yum-utils device-mapper-persistent-data lvm2

$ yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

- Install Docker Engine – Community

```
$ yum install -y docker-ce docker-ce-cli containerd.io
```

這一個步驟如果遇到要求 **container-selinux > 2.x** 的問題，請先執行下列指令，再重新執行這一個步驟

```
$ yum install wget
$ wget http://mirror.centos.org/centos/7/extras/x86_64/Packages/container-
  selinux-2.107-3.el7.noarch.rpm
$ yum install -y policycoreutils-python
$ rpm -ivh container-selinux-2.107-3.el7.noarch.rpm
```

- Enable & Start Docker

```
$ systemctl enable docker
$ systemctl start docker
```

緯創IT先進技術實驗室 Witlab

Step. 1 – Known Issue (Centos 7)

- **container-selinux > 2.x**

```
$ yum install wget
$ wget http://mirror.centos.org/centos/7/extras/x86_64/Packages/container-selinux-2.107-3.el7.noarch.rpm
$ yum install -y policycoreutils-python
$ rpm -ivh container-selinux-2.107-3.el7.noarch.rpm
```

Step. 1 – Install docker compose (Centos 7)

- Install Compose

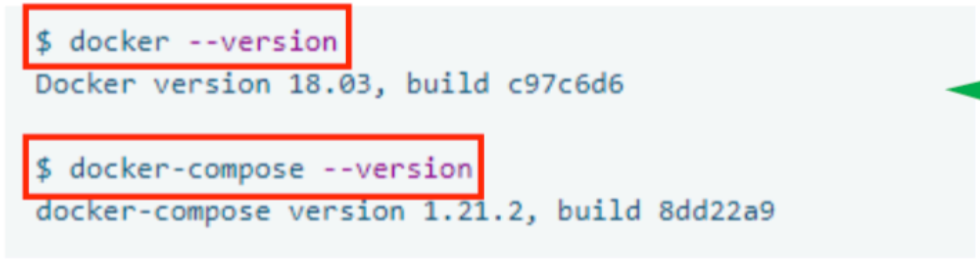
```
$ curl -L "https://github.com/docker/compose/releases/download/1.25.3/docker-compose-$(uname -s)-$(uname -m)" -o
  /usr/local/bin/docker-compose

$ chmod +x /usr/local/bin/docker-compose

$ ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

Step. 2 – Verify Docker readiness

- Check version



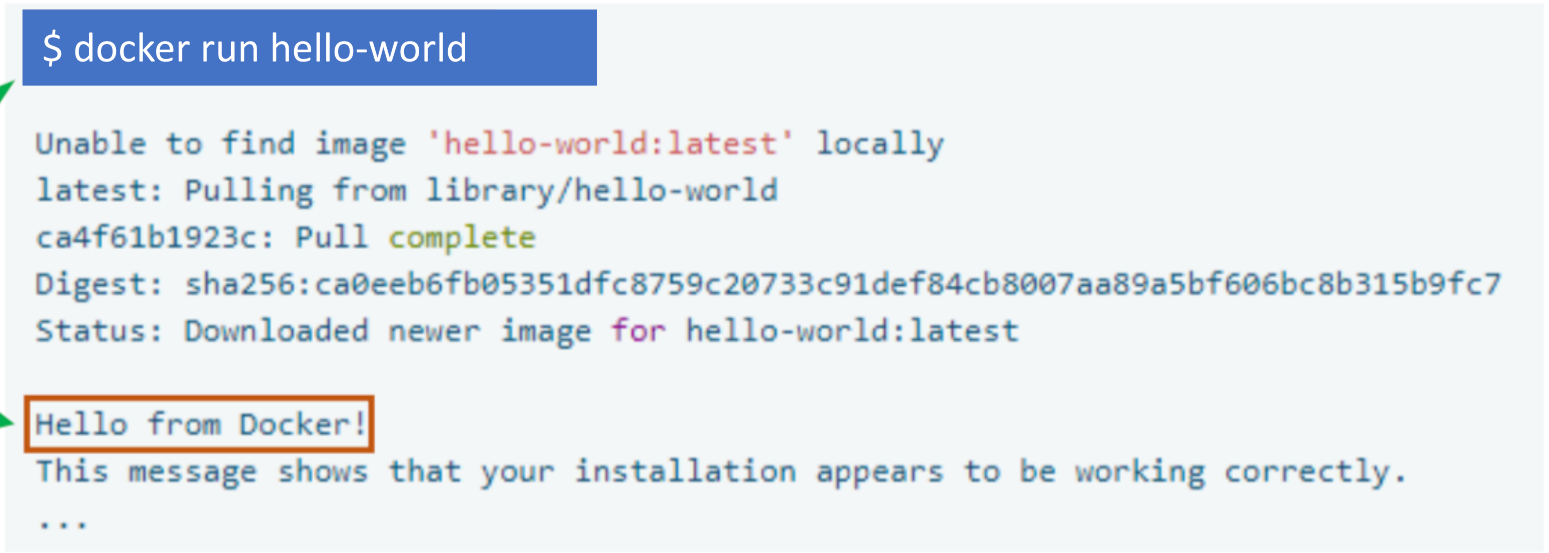
```
$ docker --version
Docker version 18.03, build c97c6d6

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

檢查看是否有正確安裝！

- Explore the application

跑一個hello-world的container來驗證



```
$ 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.
...
```

要看到這個訊息才代表Docker是正常運行喔!

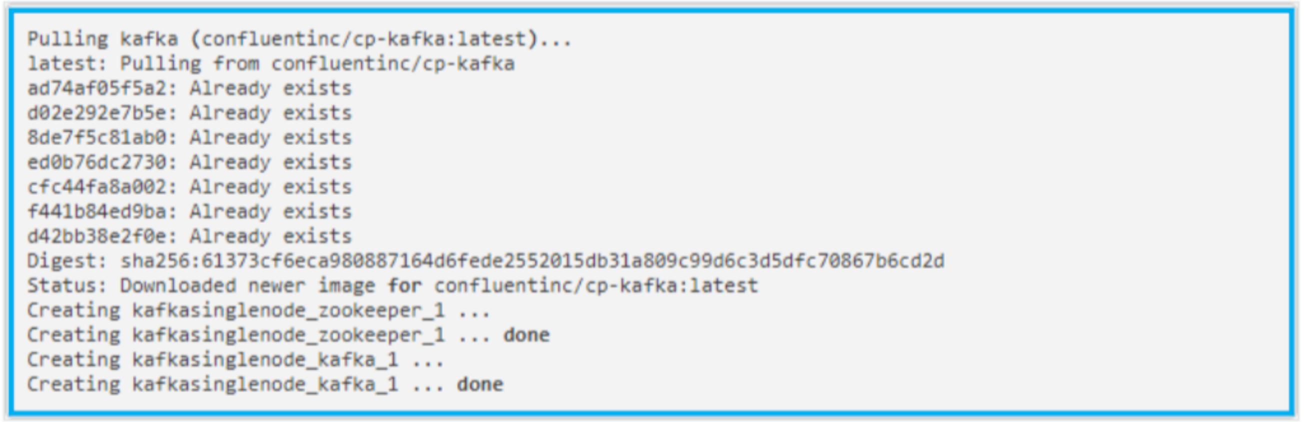
Setup Kafka & Zookeeper

using Docker Compose

Step. 1 – Start Zookeeper & Kafka

- Run below command which make a directory that contains a "**docker-compose.yml**" file.

```
$ su root
$ yum install -y git
$ cd ~
$ git clone https://github.com/semicolon1709/kafka-tutorial-docker-env.git
$ cd kafka-tutorial-docker-env
$ docker-compose up -d
```



```
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
```

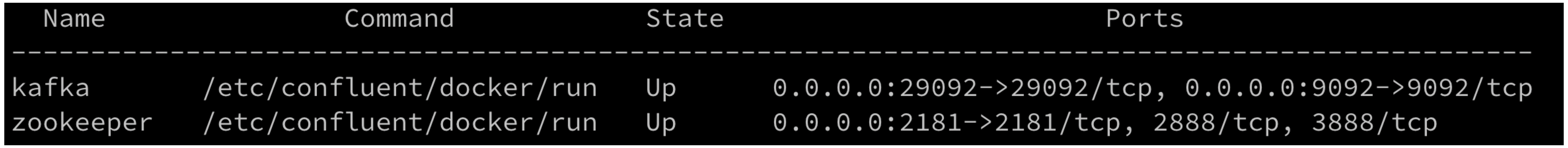
- 第一次啟動的時候，會花一點時間從網路下載 Docker 的 image 檔案

Step. 2 – Verify Zookeeper & Kafka services

- Run below command.

```
$ docker-compose ps
```

- You should see the following:



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

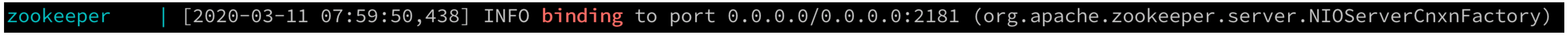
如果正確啟動，會看到本機
上有兩個 container 在跑

Step. 3 – Verify Zookeeper is healthy

- Run below command

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

- You should see the following:



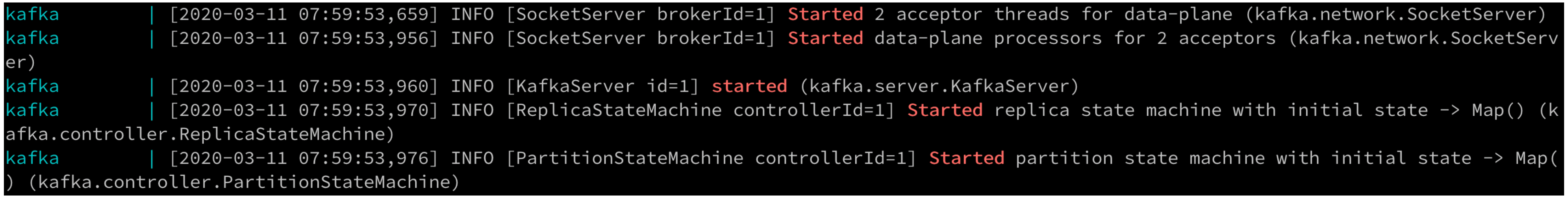
```
zookeeper     | [2020-03-11 07:59:50,438] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)
```

Step. 4 – Verify Kafka is healthy

- Run below command

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

- You should see the following:



```
kafka         | [2020-03-11 07:59:53,659] INFO [SocketServer brokerId=1] Started 2 acceptor threads for data-plane (kafka.network.SocketServer)
kafka         | [2020-03-11 07:59:53,956] INFO [SocketServer brokerId=1] Started data-plane processors for 2 acceptors (kafka.network.SocketServ
er)
kafka         | [2020-03-11 07:59:53,960] INFO [KafkaServer id=1] started (kafka.server.KafkaServer)
kafka         | [2020-03-11 07:59:53,970] INFO [ReplicaStateMachine controllerId=1] Started replica state machine with initial state -> Map() (k
afka.controller.ReplicaStateMachine)
kafka         | [2020-03-11 07:59:53,976] INFO [PartitionStateMachine controllerId=1] Started partition state machine with initial state -> Map(
) (kafka.controller.PartitionStateMachine)
```

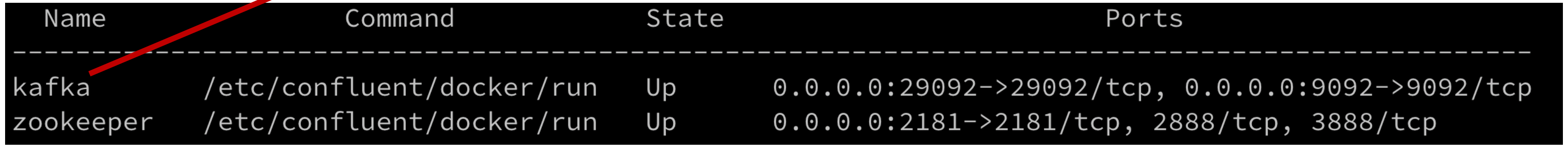
Test Kafka & Zookeeper Env.

using Docker Compose

Get into Docker container

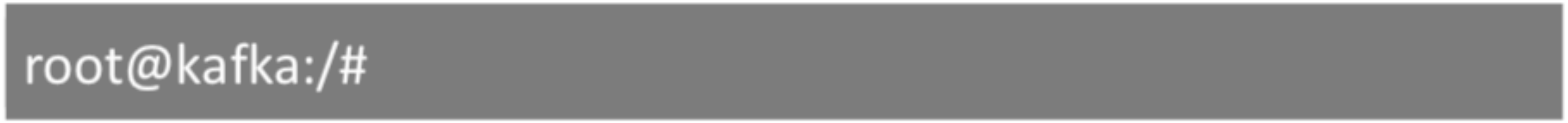
- Run below command

```
$ docker exec -it kafka bash
```



```
  Name                Command            State                      Ports
---------------------------------------------------------------------------------------------------
kafka        /etc/confluent/docker/run   Up      0.0.0.0:29092->29092/tcp, 0.0.0.0:9092->9092/tcp
zookeeper    /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:/#
```

Create a topic

- Run below command (inside-container)

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

```
Created topic “test”.
```

Exit from Docker container

- Run below command (inside-container)

```
$ exit
```

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
```

先切換到放置 docker-compose.yml 的目錄底下(Kafka-Tutorial/ 03_workspace/env), 再執行這個 command

- You should see the following:

```
Stopping kafka     ... done
Stopping zookeeper ... done
```

Step. 2 – Start exiting Zookeeper & Kafka

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

```
$ docker-compose start
```

先切換到放置 docker-compose.yml 的目錄底下(Kafka-Tutorial/ 03_workspace/env), 再執行這個 command

- You should see the following:

```
Starting zookeeper ... done
Starting kafka     ... 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)

緯創IT先進技術實驗室 Witlab

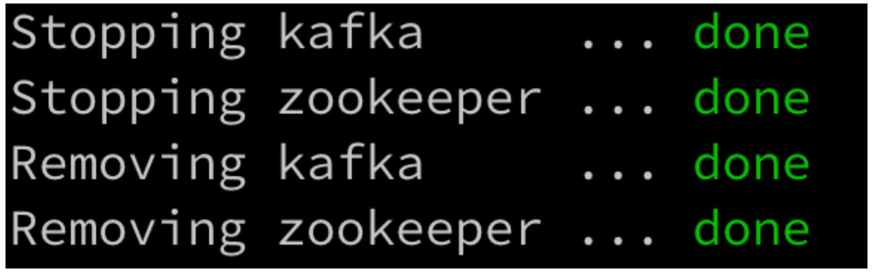
Step. 3 – Remove Zookeeper & Kafka container/data

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

```
$ docker-compose down
```

這個指令會把 containers 的資料全部清除掉!

- You should see the following:



```
Stopping kafka      ... done
Stopping zookeeper  ... done
Removing kafka      ... done
Removing zookeeper  ... done
```

The End