Kafka Fundamentals

SciSpike

Your Instructors: Petter Graff

- Architect, consultant, teacher, and sometime CXX
- Serial entrepreneur and currently the owner and CEO SciSpike and VappusNet
- Main architect of Yaktor, a scalable event-driven, agent based rapid development platform
- Teaches classes on:
  - BigData (Hadoop, Spark, HBase, Cassandra, etc.)
  - Conceptual Computer Science Concepts (Scalability, Architectural Thinking, Design Patterns, OOA&D, etc.)
  - Languages (Java, Scala, C++, JavaScript, ...)
  - And much more...
- Lives in Austin
- O'Reilly author:
  Design Patterns in Java

O'REILLY
Design Patterns in Java
Understand and Apply Analysis, Architecture, Design, and Language Patterns
Petter Graff
VIDEO

SciSpike

- Custom software development: focus on extremely fast development, web scale, cloud, Big Data applications
- Consultant, and mentor to many large organizations worldwide
- Node.js, Scala, Java…
- Big Data and NoSQL systems
- Custom training to leading firms worldwide

- http://www.scispike.com
- Custom software development
- Consulting, mentoring, training
- We are independent

Outline

What is Kafka?

- *"Kafka is a distributed, partitioned, replicated commit log service"*

- Publish-subscribe messaging system
- Fault tolerant
- Scalable
- Durable
- Apache project
- Originally developed by LinkedIn

Design Goals at LinkedIn

- Decouple the producers and consumers by using a push-pull model
- Provide persistence for messages to allow multiple consumers
- Optimize for high throughput of messages
- Allow for horizontal scaling as the data streams grow

Kafka is Born!

- Publish/subscribe system with an interface of a typical messaging system but a storage layer like a log aggregation system
- As of August 2015, LinkedIn produces over one trillion messages and a petabyte of data consumed daily!
- Kafka was released as open source in late 2010 and became an Apache project in 2011

The Name

I thought that since Kafka was a system optimized for writing using a writer's name would make sense. I had taken a lot of lit classes in college and liked Franz Kafka. Plus the name sounded cool for an open source project.

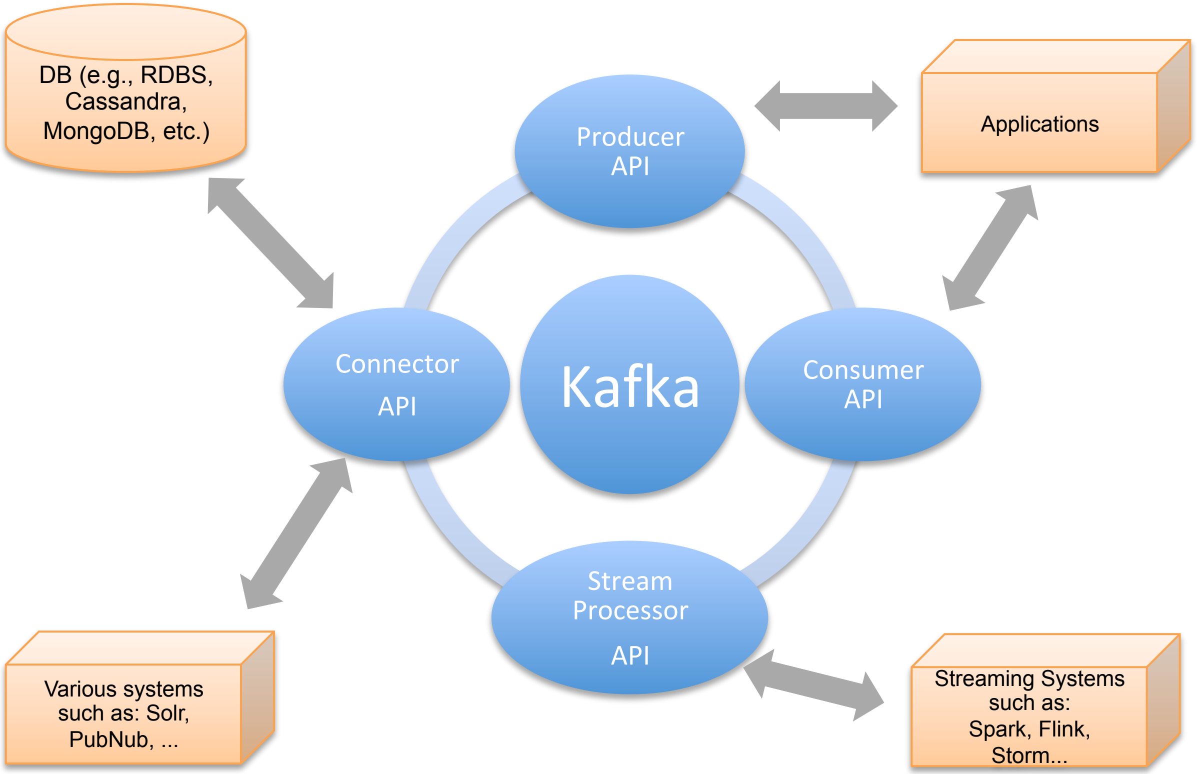
So basically there is not much of a relationship.

Jay Kreps, Lead Developer at LinkedIn

Notable Users

- Cisco Systems
- Netflix
- PayPal
- Spotify
- Uber
- HubSpot
- Betfair
- Shopify

Kafka API's


DB (e.g., RDBS, Cassandra, MongoDB, etc.)
Producer API
Applications
Connector API
Kafka
Consumer API
Stream Processor API
Various systems such as: Solr, PubNub, ...
Streaming Systems such as: Spark, Flink, Storm...

Use Cases

- Messaging
- Website Activity Tracking
- Metrics
- Log Aggregation
- Stream Processing
- Event Sourcing
- Commit Log

Website Activity Tracking

- The original use case for Kafka was to be able to rebuild user activity as a set of real-time publish-subscribe feeds
- Site activity such as page views, searches, or other user actions are published to central topics with one topic per activity type
- These feeds are then available for subscription for processing, monitoring, and loading into offline processing/reporting
- Activity tracking can be **very high volume** as many messages are generated for each user page view

SciSpike

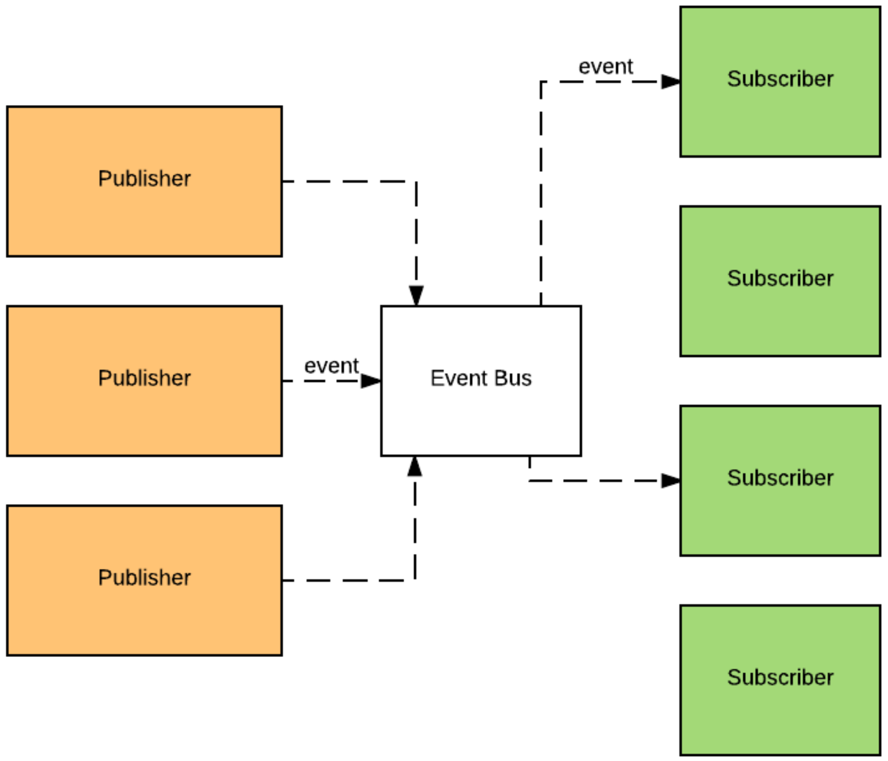
Messaging

- Kafka can be a replacement for a more traditional message broker such as ActiveMQ or RabbitMQ
- Message broker scan be used to decouple processing from data producers, buffer unprocessed messages, etc.
- Kafka has replication, built-in partitioning, and fault-tolerance making it a good solution for large scale applications

Publish-Subscribe Pattern

- Sender (publisher) wants to send a piece of data (message)
- The message is not specifically directed to a receiver (subscriber)
- The sender classifies the message and the receiver subscribes to receive certain classes of messages
- Usually there is a central broker where messages are published

Publish-Subscribe Illustrated



Log Aggregation

- Log aggregation collects physical log files off servers and put them in a central place such as a file server of HDFS for processing
- Kafka abstracts away the details of files and gives a cleaner abstraction of log or event data as a stream of messages

Why Kafka?

- Able to connect a large number of clients (Producers and Consumers)
- Durable
  - Disk based retention
  - Data is replicated across brokers
- Scalable
  - Expansions can be performed while the cluster is online
- High Performance
  - Producers, consumers, and brokers can all be scaled to handle very large message streams
  - Sub-second message latency to consumers

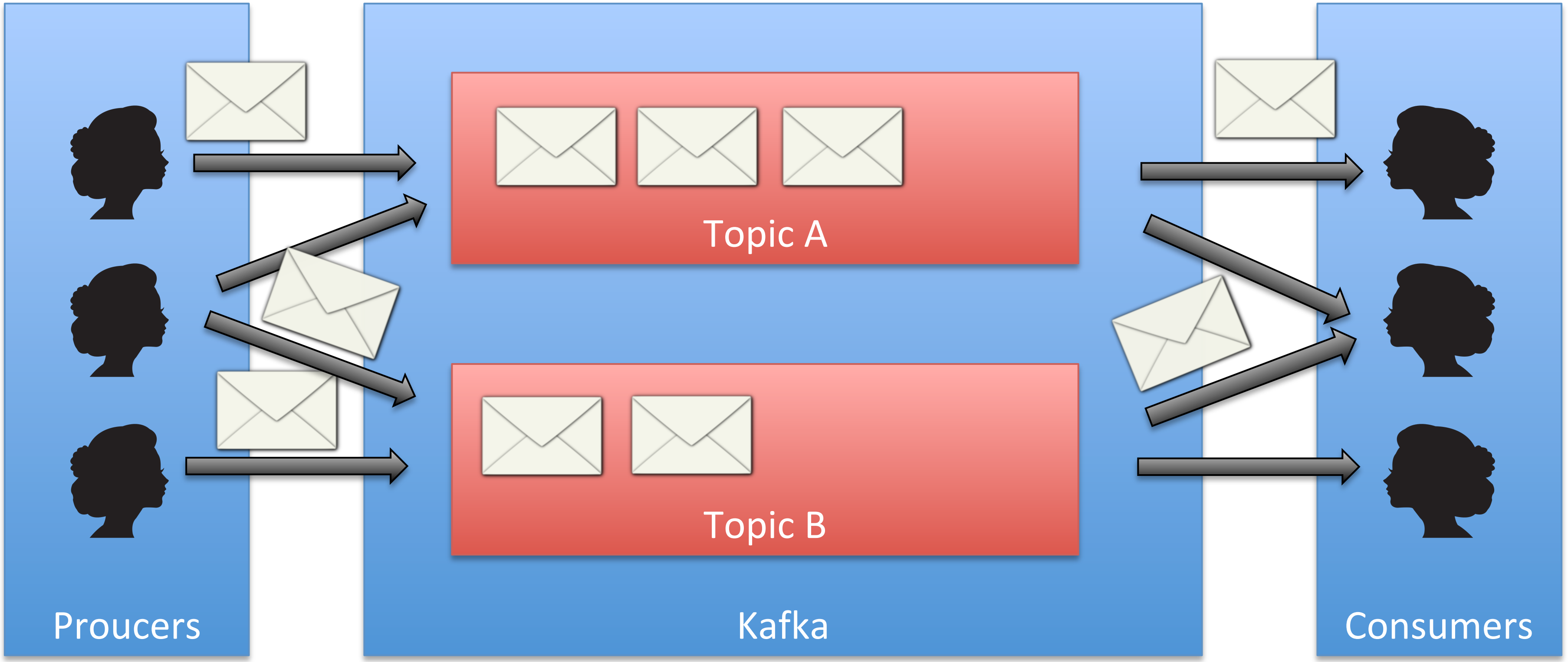
How Big is Big?

- Per day:
  - 800 billion messages
  - 175 TB of data
  - 650 TB of consumed data
- Per second:
  - 13 million messages
  - 2.75 GB of data
- Configuration
  - 1100 Kafka brokers
  - 60 clusters

$\left.\begin{matrix}\\\\\\\\\end{matrix}\right\}$ Numbers from LinkedIn

SciSpike

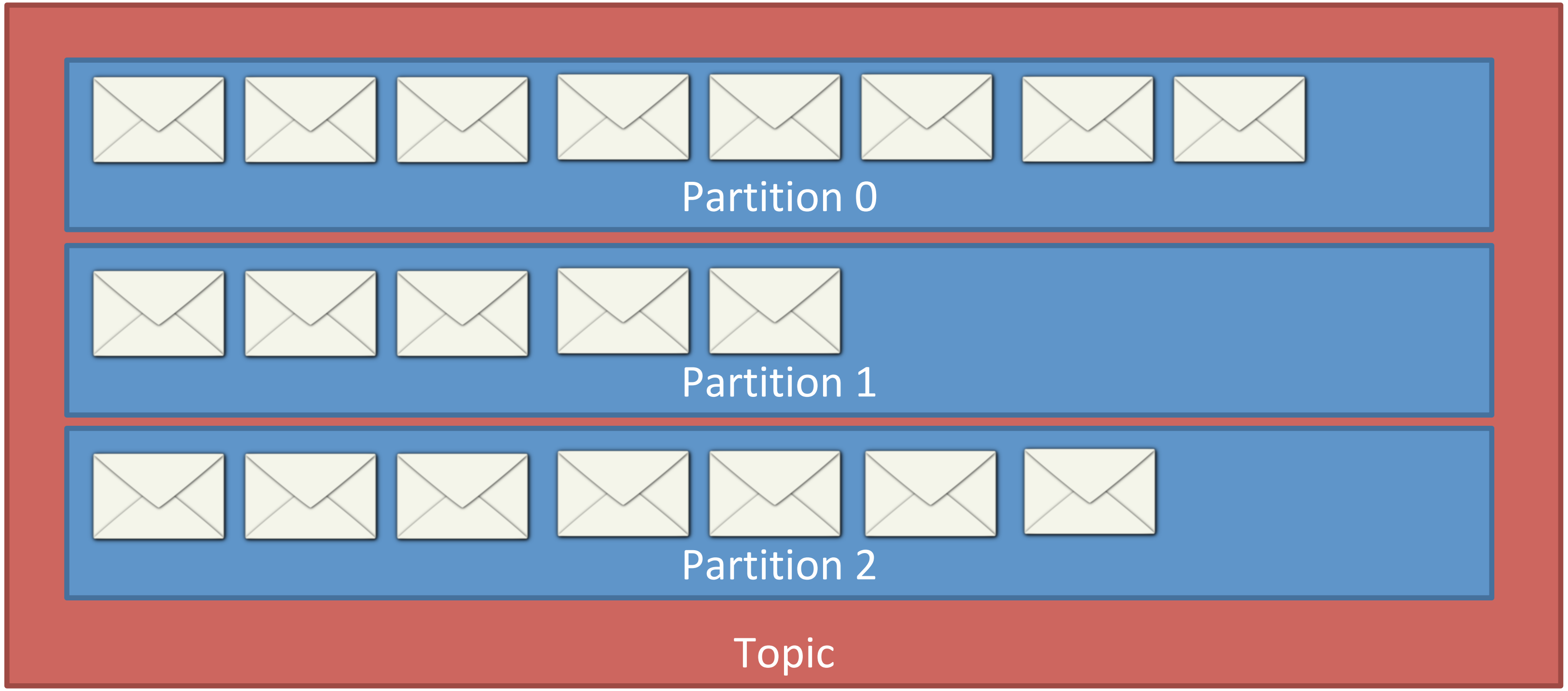
Kafka Main Concepts: Topics



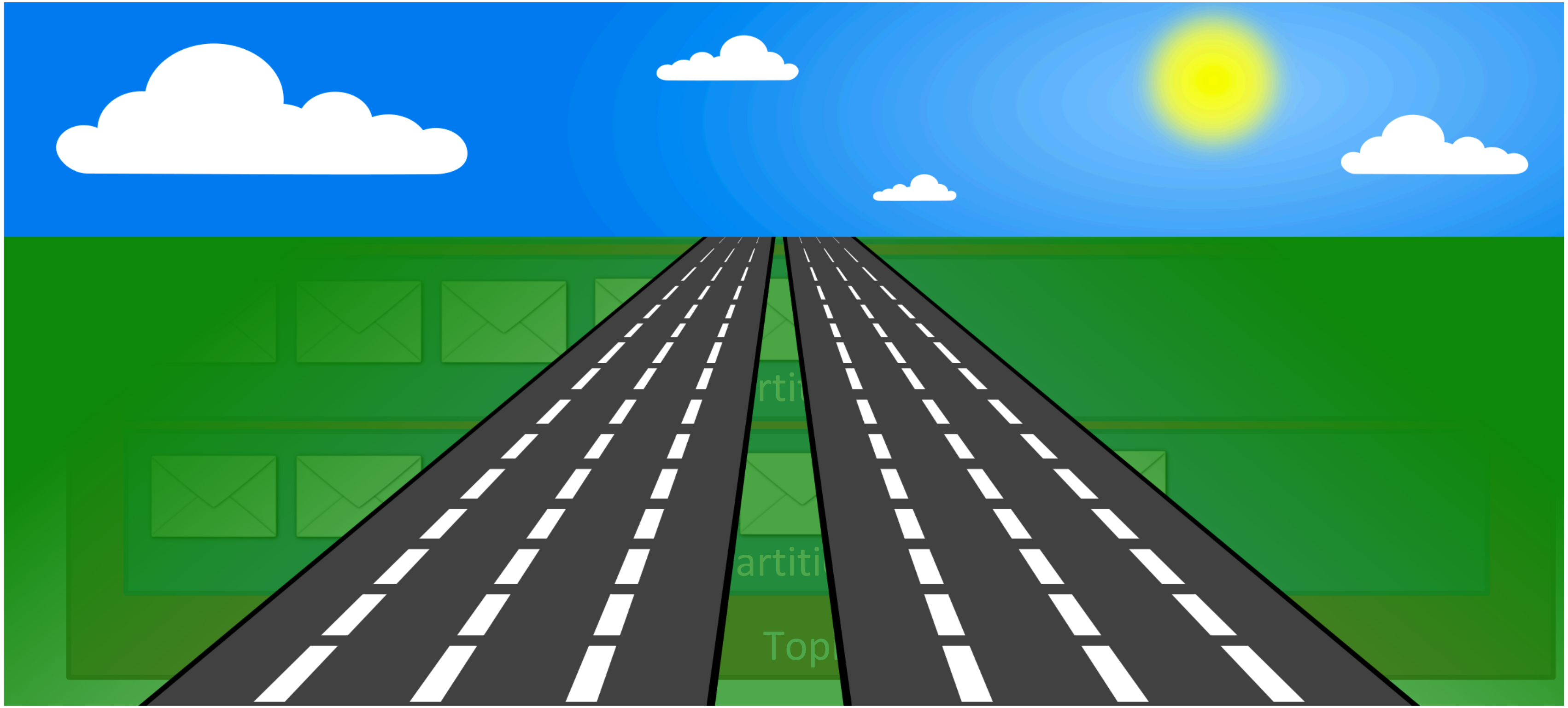
Kafka Main Concepts: Message

- Kafka looks at every message as a sequence of bytes
- The bytes are separated into:
  - Key
    - Used to determine partition (more about that later)
  - Value
    - The actual payload of the message

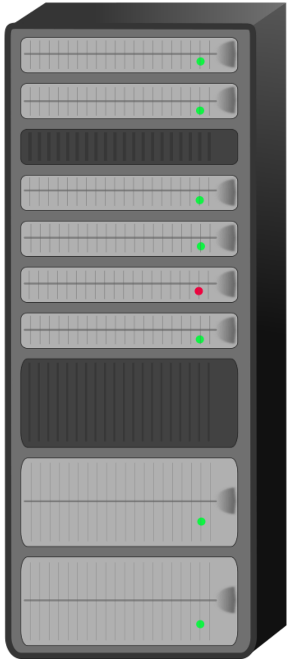
Kafka Key Concepts: Partitions



Kafka Key Concepts: Partitions

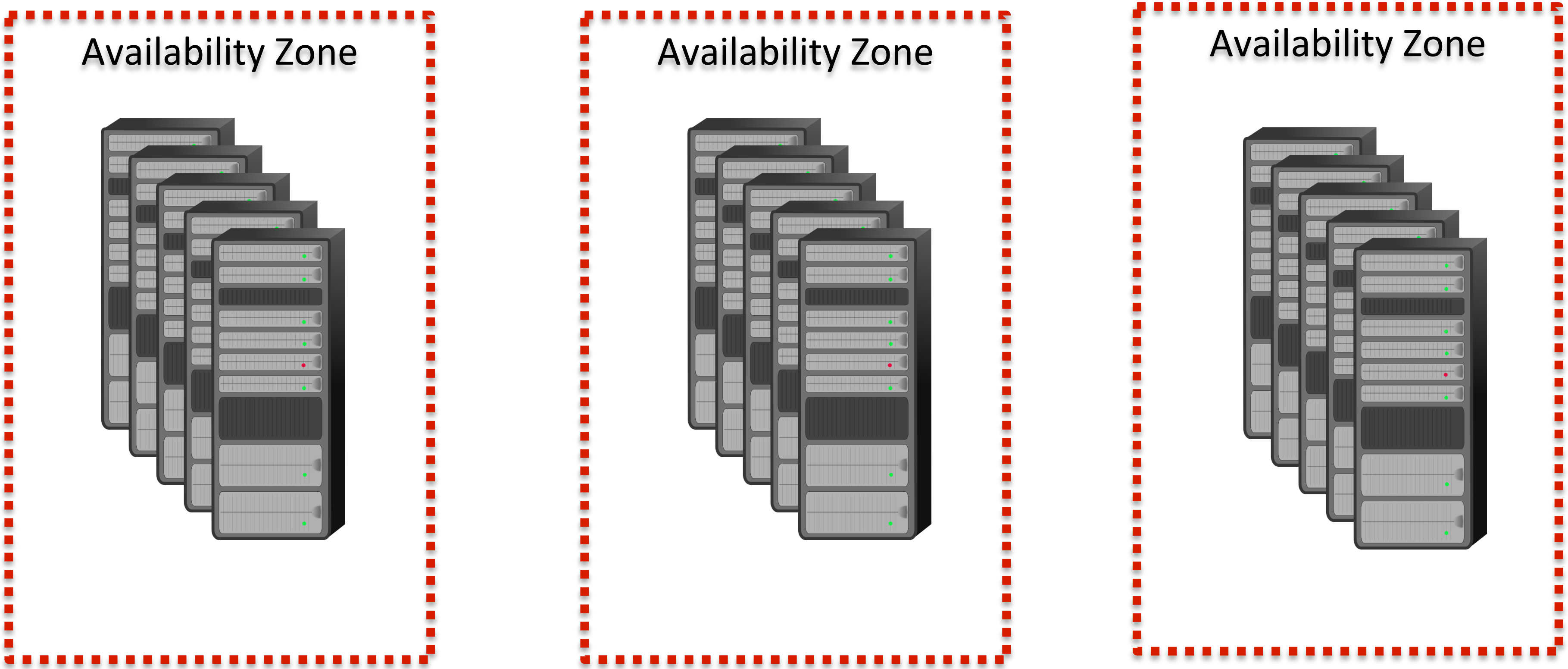


Kafka Main Concepts: Broker

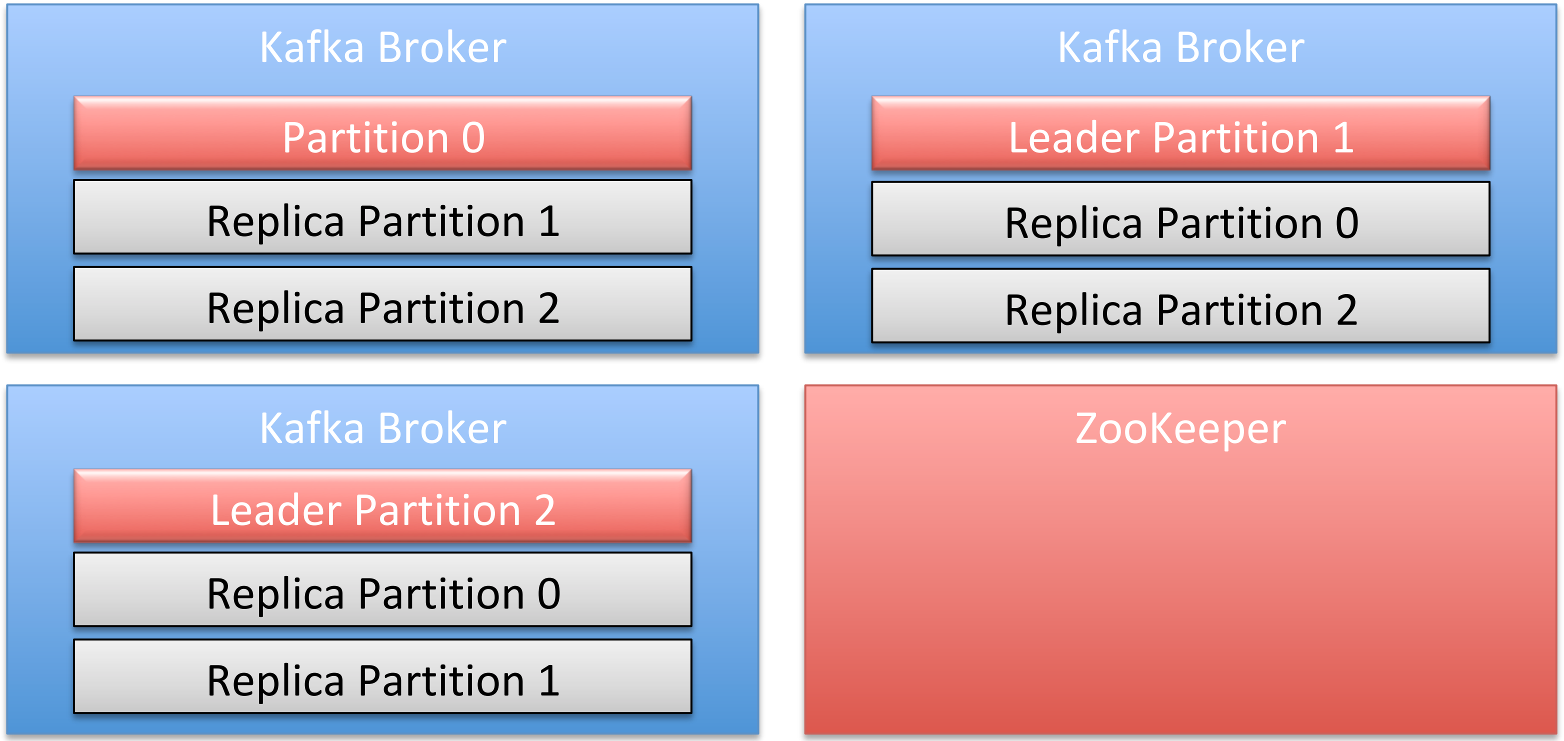


- A server or process running Kafka
- Holds multiple partitions across multiple topics
- The physical manifestation of Kafka

Kafka Main Concepts: Cluster



Kafka Main Concepts: Partitions and Clusters



Summary

- Apache Kafka is an open source, distributed, partitioned, and replicated commit-log based publish-subscribe messaging system
  - Scalable
  - High Performance
  - Multiple Consumers
  - Multiple Producers
  - Disk-based Retention

Start of Exercise

- Do you have Docker installed?

- We will be using Docker to run Kafka
- If you have not done so already, please install Docker
  - https://docs.docker.com/engine/installation/

- Check if Docker works by running

SciSpike

Check installation of Docker

```
$ docker -v
Docker version 1.13.0, build 49bf474
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
78445dd45222: Already exists
Digest: sha256:c5515758d4c5e1e838e9cd307f6c6a0d620b5e07e6f927b07d05f6d12a1ac8d7
Status: Downloaded newer image for hello-world:latest

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

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/
```

docker-compose.yml

```
version: '2'
services:
  zookeeper:
    image: wurstmeister/zookeeper:3.4.6
    ports:
     - 2181:2181
  kafka:
    image: wurstmeister/kafka:1.1.0
    environment:
      KAFKA_LISTENERS: PLAINTEXT://kafka:9092
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
    ports:
      - 9092:9092
```

Run Zookeeper

Run Kafka

Essential Kafka CLI Commands (used in exercise)

- Kafka can be configured via the command
- Key commands
  - kafka-topics.sh
    - Allows us to manipulate and view topics
  - kafka-console-producer.sh
    - Allows us to produce data from using stdin
  - kafka-console-consumer.sh
    - Allows us to consume messages from the console
- We're running all tools in docker, so we have to 'reach into' the docker instance, hence our commands look like this:
  - docker-compose exec kafka /opt/kafka/bin/CMD
  - You may want to alias these commands to reduce typing or simply run a bash shell inside the docker image
    - docker-compose exec kafka /bin/bash

Lab

- In this lab we'll simply ensure that we can run Kafka
- We'll run Kafka using Docker
  - Easy configuration
  - Runs Kafka and Zookeeper
  - Flexible setup that allows us to setup a cluster of machines for later exercises
- The lab simply:

1. Starts the docker images using docker-compose
2. Runs a simple producer
3. Runs a simple consumer
4. Allows you to type messages in the producer and see them consumed by the consumer

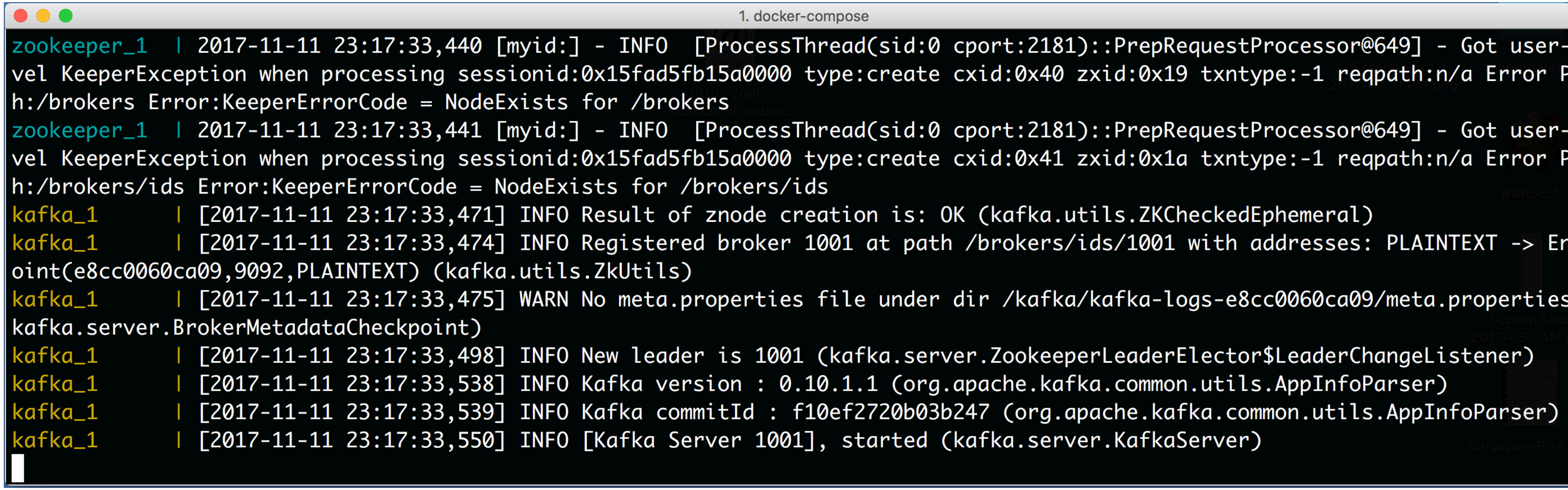
Step 1: Start Kafka and ZooKeeper

a) In a terminal, go to:

```
kafka-labs/labs/01-Verify-Installation
```

b) Run:

```
docker-compose up
```



```
zookeeper_1  | 2017-11-11 23:17:33,440 [myid:] - INFO  [ProcessThread(sid:0 cport:2181)::PrepRequestProcessor@649] - Got user-
vel KeeperException when processing sessionid:0x15fad5fb15a0000 type:create cxid:0x40 zxid:0x19 txntype:-1 reqpath:n/a Error P
h:/brokers Error:KeeperErrorCode = NodeExists for /brokers
zookeeper_1  | 2017-11-11 23:17:33,441 [myid:] - INFO  [ProcessThread(sid:0 cport:2181)::PrepRequestProcessor@649] - Got user-
vel KeeperException when processing sessionid:0x15fad5fb15a0000 type:create cxid:0x41 zxid:0x1a txntype:-1 reqpath:n/a Error P
h:/brokers/ids Error:KeeperErrorCode = NodeExists for /brokers/ids
kafka_1      | [2017-11-11 23:17:33,471] INFO Result of znode creation is: OK (kafka.utils.ZKCheckedEphemeral)
kafka_1      | [2017-11-11 23:17:33,474] INFO Registered broker 1001 at path /brokers/ids/1001 with addresses: PLAINTEXT -> En
oint(e8cc0060ca09,9092,PLAINTEXT) (kafka.utils.ZkUtils)
kafka_1      | [2017-11-11 23:17:33,475] WARN No meta.properties file under dir /kafka/kafka-logs-e8cc0060ca09/meta.properties
kafka.server.BrokerMetadataCheckpoint)
kafka_1      | [2017-11-11 23:17:33,498] INFO New leader is 1001 (kafka.server.ZookeeperLeaderElector$LeaderChangeListener)
kafka_1      | [2017-11-11 23:17:33,538] INFO Kafka version : 0.10.1.1 (org.apache.kafka.common.utils.AppInfoParser)
kafka_1      | [2017-11-11 23:17:33,539] INFO Kafka commitId : f10ef2720b03b247 (org.apache.kafka.common.utils.AppInfoParser)
kafka_1      | [2017-11-11 23:17:33,550] INFO [Kafka Server 1001], started (kafka.server.KafkaServer)
```

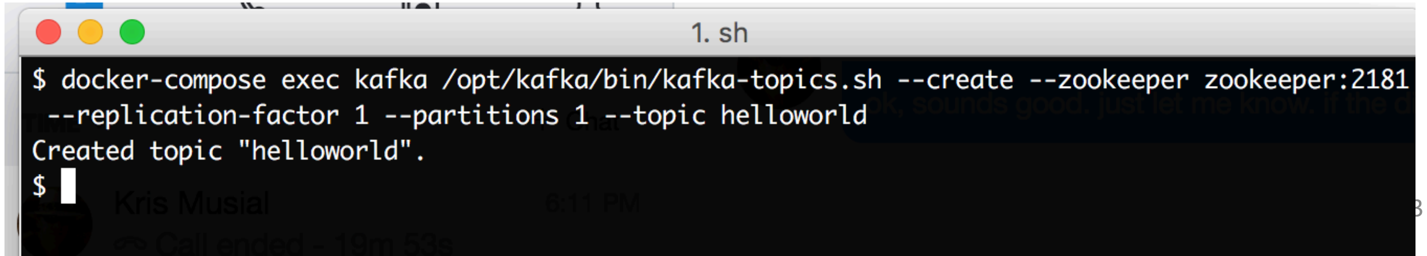
Step 2: Create a Topic

a) Run a second terminal and go to the directory

```
kafka-labs/labs/01-Verify-Installation
```

b) In this directory run:

```
docker-compose exec kafka
/opt/kafka/bin/kafka-topics.sh
--create
--zookeeper zookeeper:2181
--replication-factor 1
--partitions 1
--topic helloworld
```



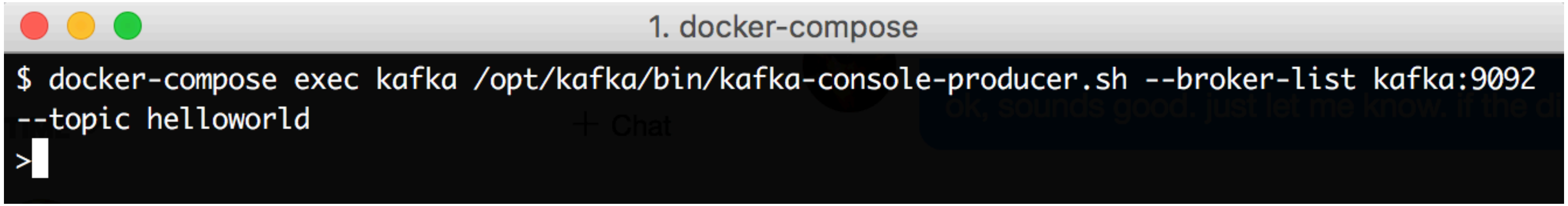
```
$ docker-compose exec kafka /opt/kafka/bin/kafka-topics.sh --create --zookeeper zookeeper:2181
 --replication-factor 1 --partitions 1 --topic helloworld
Created topic "helloworld".
$ 
```

Step 3: Run the Kafka Producer

a) In the same terminal as you created the topic, in the directory:
`kafka-labs/labs/01-Verify-Installation`

b) In this directory run:

```
docker-compose exec kafka
/opt/kafka/bin/kafka-console-producer.sh --broker-list
kafka:9092
--topic helloworld
```

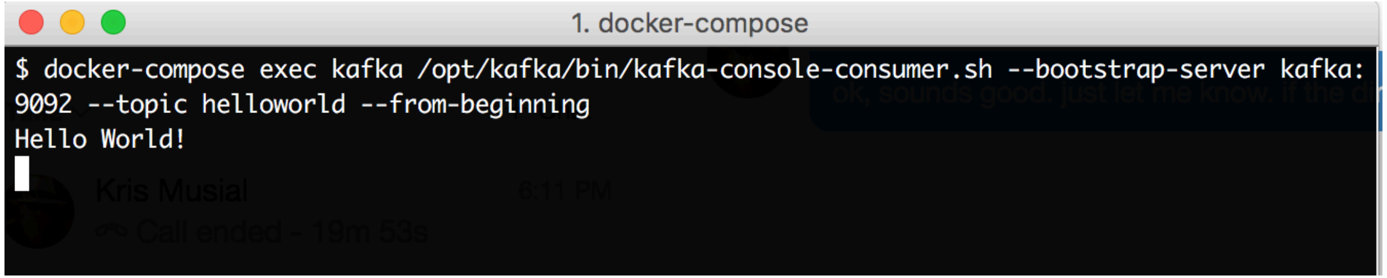


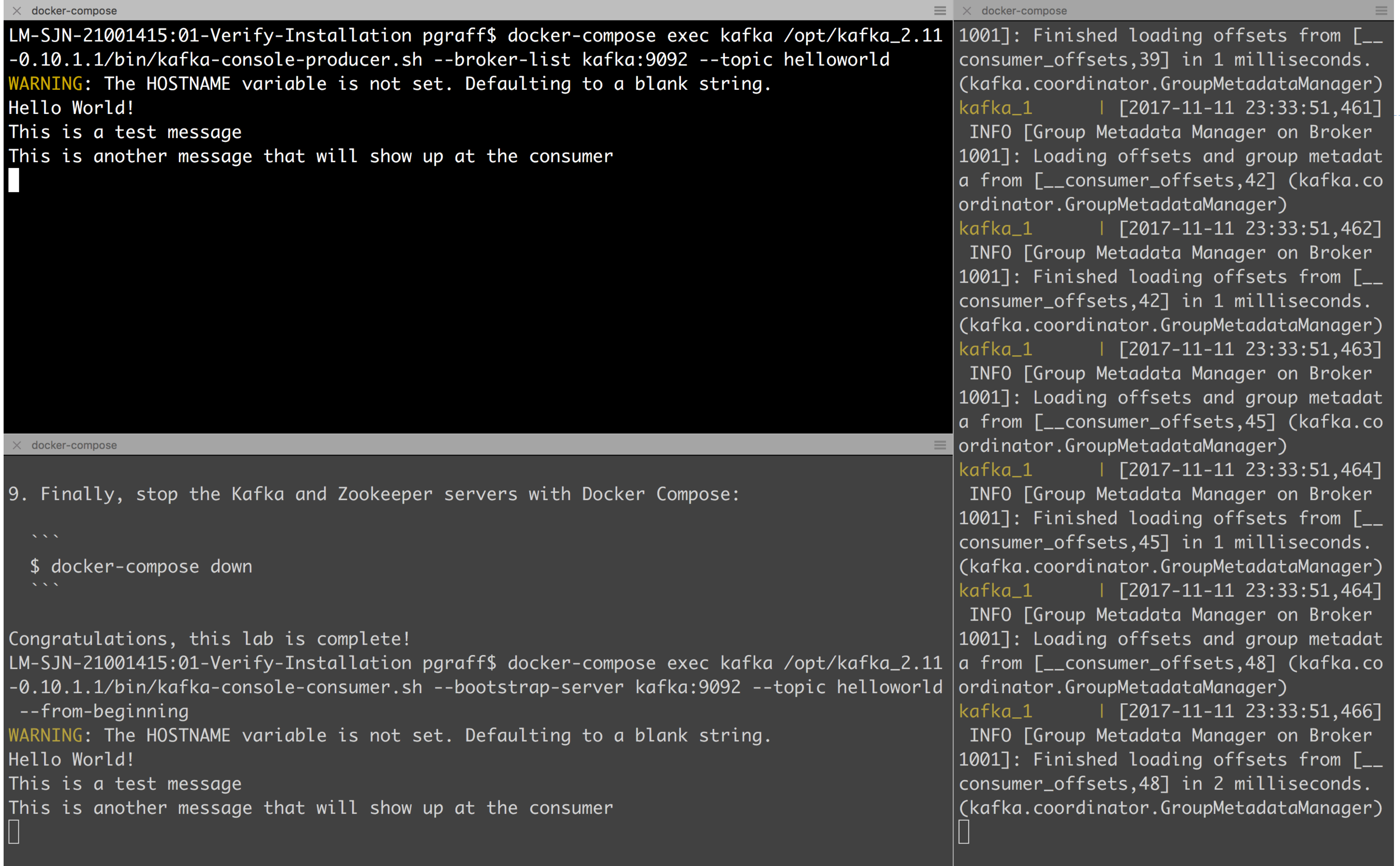
```
$ docker-compose exec kafka /opt/kafka/bin/kafka-console-producer.sh --broker-list kafka:9092
--topic helloworld
>
```

Step 3: Run the Kafka Producer

a) Run a third terminal and go to the directory
kafka-labs/labs/01-Verify-Installation

b) In this directory run:
```
docker-compose exec kafka
/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-
server kafka:9092
--topic helloworld
--from-beginning
```


1. docker-compose
```
$ docker-compose exec kafka /opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server kafka:
9092 --topic helloworld --from-beginning
Hello World!
```



docker-compose

```
LM-SJN-21001415:01-Verify-Installation pgraff$ docker-compose exec kafka /opt/kafka_2.11
-0.10.1.1/bin/kafka-console-producer.sh --broker-list kafka:9092 --topic helloworld
WARNING: The HOSTNAME variable is not set. Defaulting to a blank string.
Hello World!
This is a test message
This is another message that will show up at the consumer
```

docker-compose

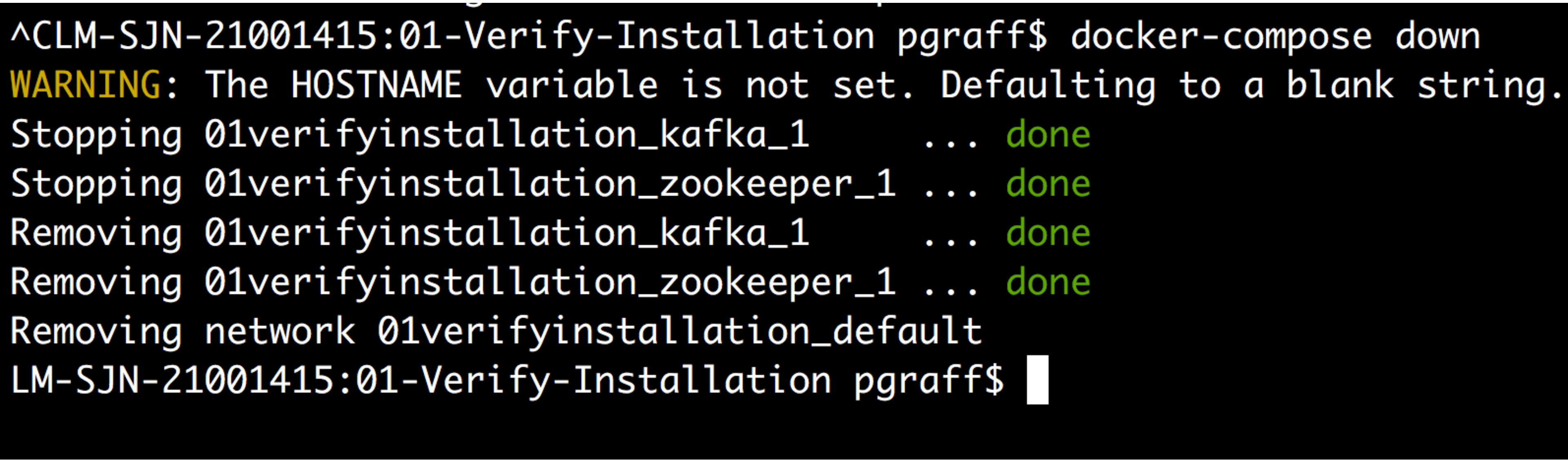
```

9. Finally, stop the Kafka and Zookeeper servers with Docker Compose:

  ```
  $ docker-compose down
  ```

Congratulations, this lab is complete!
LM-SJN-21001415:01-Verify-Installation pgraff$ docker-compose exec kafka /opt/kafka_2.11
-0.10.1.1/bin/kafka-console-consumer.sh --bootstrap-server kafka:9092 --topic helloworld
 --from-beginning
WARNING: The HOSTNAME variable is not set. Defaulting to a blank string.
Hello World!
This is a test message
This is another message that will show up at the consumer
```

docker-compose

```
1001]: Finished loading offsets from [__
consumer_offsets,39] in 1 milliseconds.
(kafka.coordinator.GroupMetadataManager)
kafka_1        | [2017-11-11 23:33:51,461]
 INFO [Group Metadata Manager on Broker
1001]: Loading offsets and group metadat
a from [__consumer_offsets,42] (kafka.co
ordinator.GroupMetadataManager)
kafka_1        | [2017-11-11 23:33:51,462]
 INFO [Group Metadata Manager on Broker
1001]: Finished loading offsets from [__
consumer_offsets,42] in 1 milliseconds.
(kafka.coordinator.GroupMetadataManager)
kafka_1        | [2017-11-11 23:33:51,463]
 INFO [Group Metadata Manager on Broker
1001]: Loading offsets and group metadat
a from [__consumer_offsets,45] (kafka.co
ordinator.GroupMetadataManager)
kafka_1        | [2017-11-11 23:33:51,464]
 INFO [Group Metadata Manager on Broker
1001]: Finished loading offsets from [__
consumer_offsets,45] in 1 milliseconds.
(kafka.coordinator.GroupMetadataManager)
kafka_1        | [2017-11-11 23:33:51,464]
 INFO [Group Metadata Manager on Broker
1001]: Loading offsets and group metadat
a from [__consumer_offsets,48] (kafka.co
ordinator.GroupMetadataManager)
kafka_1        | [2017-11-11 23:33:51,466]
 INFO [Group Metadata Manager on Broker
1001]: Finished loading offsets from [__
consumer_offsets,48] in 2 milliseconds.
(kafka.coordinator.GroupMetadataManager)
```

Final Step: Shutdown Kafka and Zookeeper

- Final step is to shutodwn the experiment
  1. Control-C out of the Producer and the Consumer
  2. In one of the terminals (e.g., Producer) run:
     docker-compose down



```
^CLM-SJN-21001415:01-Verify-Installation pgraff$ docker-compose down
WARNING: The HOSTNAME variable is not set. Defaulting to a blank string.
Stopping 01verifyinstallation_kafka_1     ... done
Stopping 01verifyinstallation_zookeeper_1 ... done
Removing 01verifyinstallation_kafka_1     ... done
Removing 01verifyinstallation_zookeeper_1 ... done
Removing network 01verifyinstallation_default
LM-SJN-21001415:01-Verify-Installation pgraff$ 
```