



## A Guide to Setup a Kafka Environment

📅 20 January 2020

# Kafka (<https://www.mitrais.com/tag/kafka/>), Software Development (<https://www.mitrais.com/tag/software-development/>)

💬 3 Comments

---

### Brief Introduction

Apache Kafka was developed by LinkedIn to handle their log files and handed over to the open source community in early 2011. It became the main Apache project in October 2012. Apache Kafka is a distributed streaming platform, and has three key capabilities:

- Publish and subscribe to streams of records, similar to a message queue or enterprise messaging system
- Store streams of records in a fault-tolerant, durable way
- Process streams of records as they occur

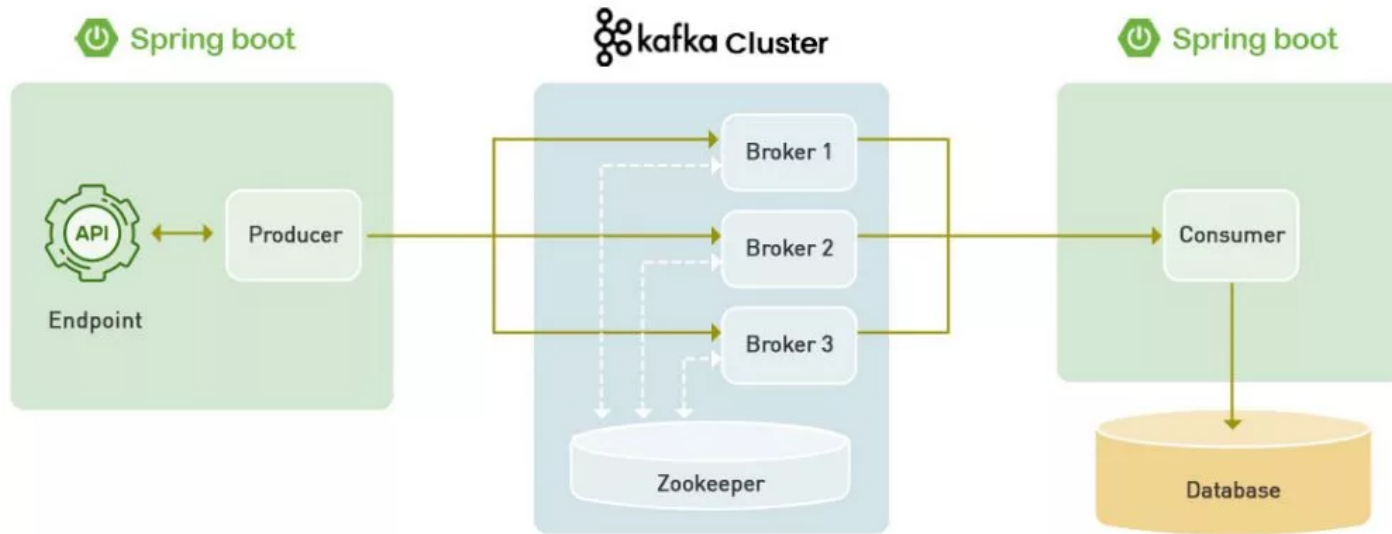
What is it used for?

Apache Kafka can be used for various purposes in an organisation, like as a Messaging service, Real-time stream processing, Log aggregation, Commit log service, Event sourcing, etc.

In order to use this technology, we need to first set up our Kafka environment. In this demonstration we will set up a Kafka environment consisting of 1 zookeeper and 3 brokers. We will create a simple Producer and Consumer using Java (/software-development/java/) Spring boot.

## System Architecture

- **Zookeeper:** is used by Kafka to maintain state between the nodes of the cluster.
- **Brokers:** The “pipes” in our pipeline, which store and emit data
- **Producers:** Send messages to the topic
- **Consumer:** Listen and read messages from topic



# Installation

We will set up our Kafka cluster in a Centos 7 environment.

## I. Prerequisite

- Centos 7 System
- Java 1.8  
If you haven't installed Java 1.8 in your system, please install it first
- kafka\_2.11-2.3.0.tgz  
You can download the file here (<https://kafka.apache.org/downloads>).
- Spring boot application  
If you want to know how to build a spring boot application from scratch, you can refer to this link (<https://spring.io/guides/gs/spring-boot/>), or if you want to just learn how to create a Kafka Producer and Consumer using spring boot please feel free to clone my project (<https://github.com/arifpurwandaru/BootCamp>) instead.

## II. Starting Zookeeper

First, extract `kafka_2.11-2.3.0.tgz` using this command:

```
tar -xvzf kafka_2.11-2.3.0.tgz
```

Inside the extracted `kafka_2.11-2.3.0` folder, you will find a `bin/zookeeper-server-start.sh` file which is used to start the zookeeper and `config/zookeeper.properties` which provides the default configuration for the zookeeper server to run.

Start the zookeeper by running (inside the Kafka root folder):

```
nohup bin/zookeeper-server-start.sh config/zookeeper.properties &>zookeeper_nohup.out&
```

This command uses `nohup` to transfer the output log into a file (`zookeeper_nohup.out`). You can monitor the log by tailing to that file using this command:

```
tail -2000f zookeeper_nohup.out
```

### III. Starting Brokers

In the same way that we started Zookeeper, there are 2 files `config/kafka-server-start.sh` to start the broker and `config/server.properties` to configure the broker server. From the architecture diagram above we want to start 3 brokers, and all you need to do is create 3 configuration properties files. Let's create them:

```
cp config/server.properties config/server1.properties
```

```
cp config/server.properties config/server2.properties
```

```
cp config/server.properties config/server3.properties
```

---

There are, however, 3 properties that must be unique for each broker instance: `broker.id`, `listeners`, and `log.dirs`.

Change the above 3 properties for each copy of the file so that they are all unique.

## server1.properties

```
broker.id=1
```

```
listeners=PLAINTEXT://172.19.16.244:9093
```

```
log.dirs=/tmp/kafka-logs1
```

## server2.properties

```
broker.id=2
```

```
listeners=PLAINTEXT://172.19.16.244:9094
```

```
log.dirs=/tmp/kafka-logs2
```

## server3.properties

```
broker.id=3
```

```
listeners=PLAINTEXT://172.19.16.244:9095
```

```
log.dirs=/tmp/kafka-logs3
```

**Important Note:** 172.19.16.224 is the host IP address (of your centos 7 system). You must set the IP address or domain name in the listener property or your cluster cannot be accessed from outside of your host.

---

Now it's time to run the brokers.

Enter this command to run broker 1

```
nohup bin/kafka-server-start.sh config/server3.properties &> server3_log.out&
```

Enter this command to run broker 2

```
nohup bin/kafka-server-start.sh config/server2.properties &> server2_log.out&
```

Enter this command to run broker 3

```
nohup bin/kafka-server-start.sh config/server3.properties &> server3_log.out&
```

**Warning:** Don't forget to open the brokers port in the centos firewall. You can run the following command to open those 3 ports (9093,9094,9095) in the firewall:

```
firewall-cmd --zone=public --add-port=9093/tcp --permanent
```

```
firewall-cmd --zone=public --add-port=9094/tcp --permanent
```

```
firewall-cmd --zone=public --add-port=9095/tcp --permanent
```

```
firewall-cmd --reload
```

Now we have our brokers running. Before we can start putting data into the cluster, we need to create a topic to which the data will belong.

## Creating a Topic

To create a topic, run this command:

```
bin/kafka-topics.sh --create --topic bootcamp-topic --zookeeper localhost:2181 --partitions 3 --replication-factor 1
```

It will create a Kafka topic named 'bootcamp-topic'. The *--partition* argument describes how many brokers we will use. Since we set up 3 brokers, we can set this option to 3. The *--replication-factor* describes how many copies of your data you require. This is handy – in case

one of the brokers goes down, you still have your data on the others.

## IV. Testing, Connecting and Monitoring our Kafka using Kafka Tools

Now our Kafka is ready to use, we can monitor Kafka traffic using Kafka Tools that can be downloaded from <http://www.kafkatool.com/download.html>  
(<http://www.kafkatool.com/download.html>)

In the Kafka Tools GUI, click **File>>Add New Connection**

**Add Cluster**

**General**

Cluster name: LocalKafka

Kafka Cluster Version: 0.11

Zookeeper Host: 172.19.16.224 Ping

Zookeeper Port: 2181

chroot path: /

**Broker Security**

Type: Plaintext

**Advanced**

Bootstrap servers: 72.19.16.224:9093,172.19.16.224:9094,172.19.16.224:9095  
You can optionally specify broker endpoint(s) to use instead of brokers discovered in Zookeeper. For example: broker1:9092,bro...

SASL Mechanism:   
Enter a value to override the default sasl.mechanism value [GSSAPI]

**Offset Topic**

☐ Use background thread to read from \_\_consumer\_offsets topic  
If disabled, viewing the consumer offsets will be considerably slower

Test Add Cancel

Fill in the following fields:

Cluster Name: You can input anything – it's just a connection name

Zookeeper Host: IP address or domain name where we installed the Kafka cluster

Bootstrap server: your broker's instance [IP,port] separated by comma

Click the Test button to test the connection. If it's a success, you can add the connection by pressing the Yes button on the popup dialog.



per Host

per Port

path

Security

ed

ap servers


n optionally specify broker endpoint(s) to use instead of brokers discovered in Zookeeper. For example: broker1:9092,broker2:9092

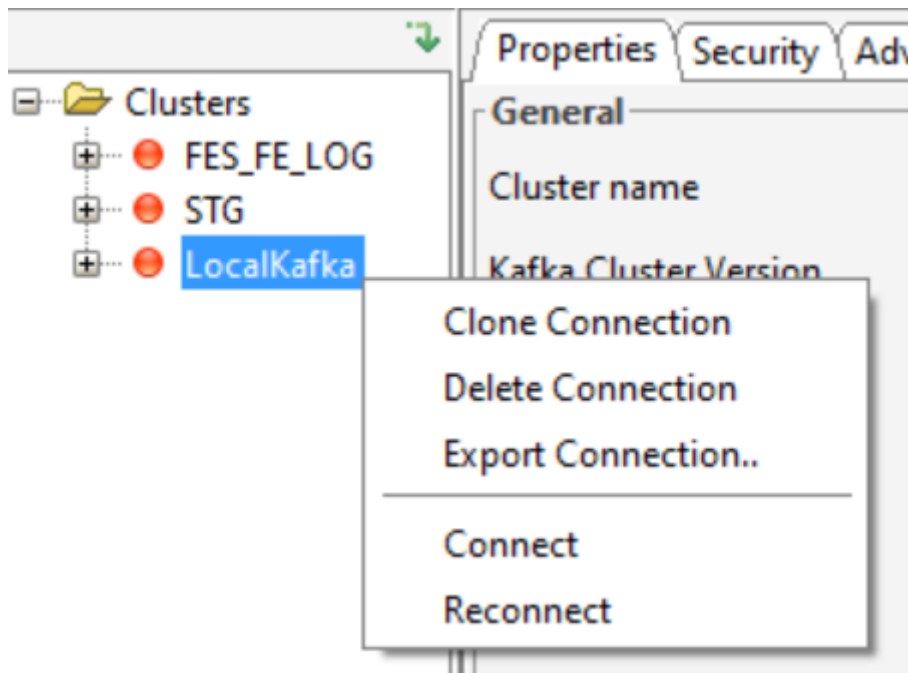
mechanism

value to override the default sasl.mechanism value [GSSAPI]

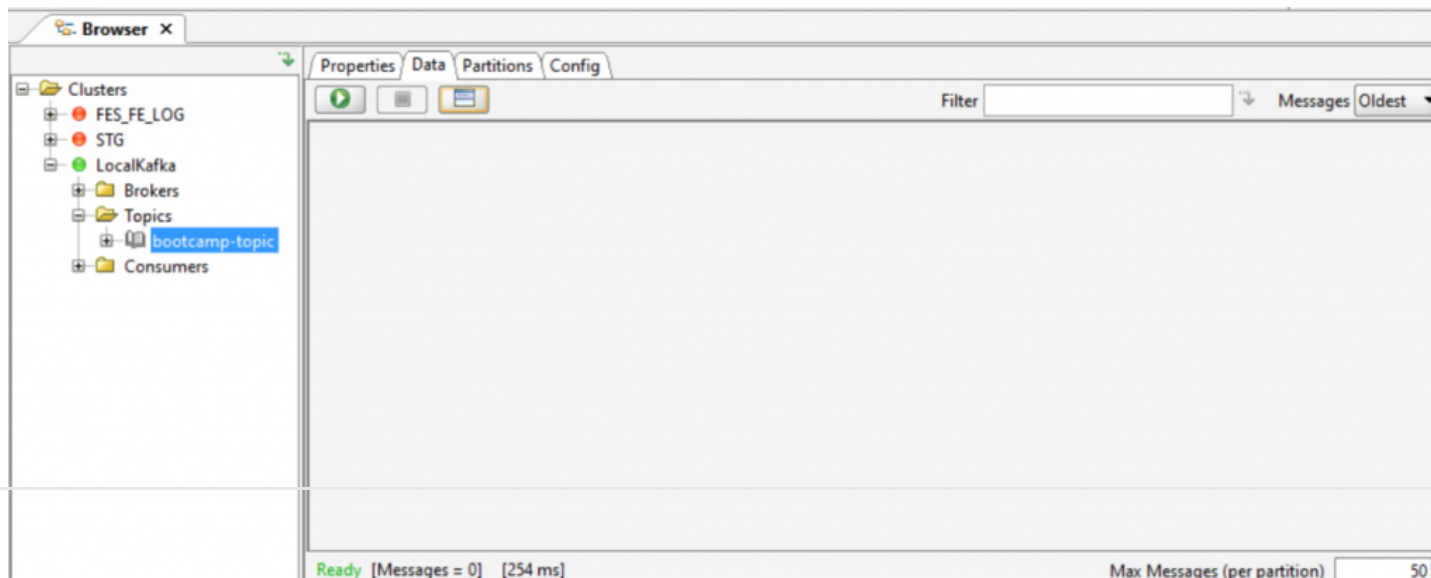
Topic

Connection Test Successful

 Connection successful, do you want to add this connection?



Right click your newly added connection and click Connect.



This is how it should look. There is no data in the topic yet – we will add it later.

## V. Creating a Producer

In this section, we will create the producer within our spring-boot application. You can create and use your own spring boot application from scratch, or clone my spring boot project (<https://github.com/arifpurwandaru/BootCamp>) from github.

First, we need to add the *spring-Kafka* library. Put this in your *build.gradle* :

```
compile("org.springframework.kafka:spring-kafka")
```

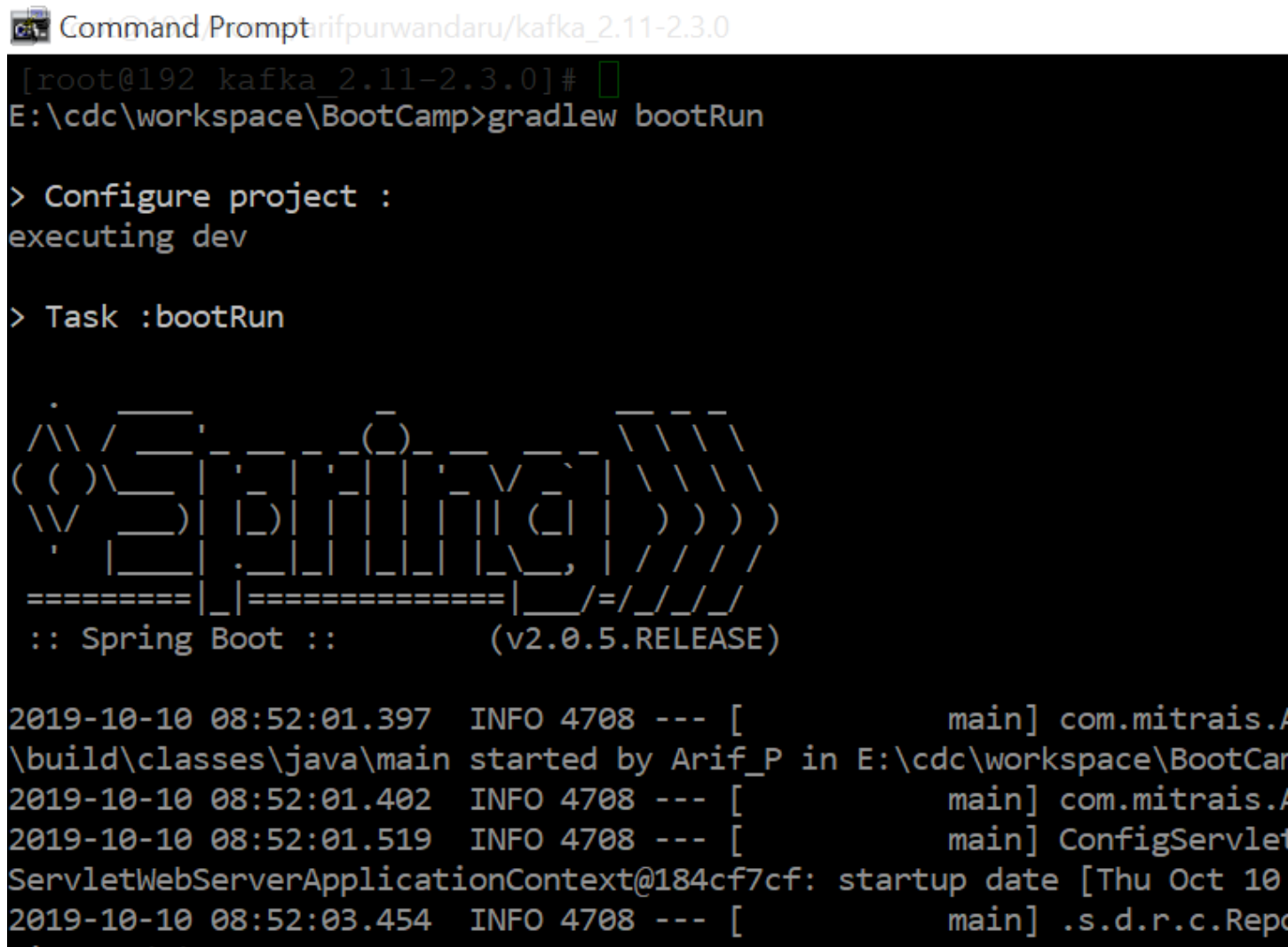
Add the following properties in *application.properties*:

```
spring.kafka.bootstrap-servers=172.19.16.224:9093,172.19.16.224:9094,172.19.16.224:9095
kafka.retries=3
kafka.session.timeout=15000
kafka.my.topic=bootcamp-topic
kafka.auto.commit=true
kafka.offset.reset=latest
kafka.security.protocol=PLAINTEXT
spring.kafka.consumer.group-id=jcg-group
```

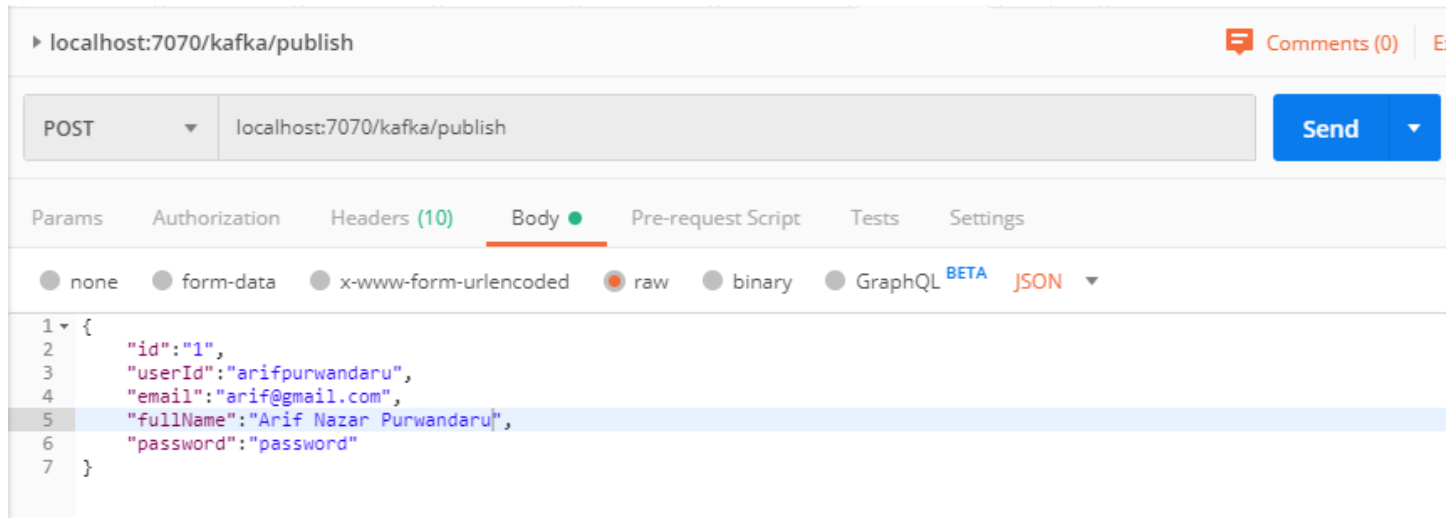
Now you can create a simple Restful API that will publish posted request body from the client to Kafka.

```
SimpleKafkaController.java
1 package com.mitrais.controller;
2 import org.springframework.beans.factory.annotation.Autowired;
3 import org.springframework.beans.factory.annotation.Value;
4 import org.springframework.http.MediaType;
5 import org.springframework.kafka.core.KafkaTemplate;
6 import org.springframework.web.bind.annotation.PostMapping;
7 import org.springframework.web.bind.annotation.RequestBody;
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.RestController;
10 import com.mitrais.persistence.entity.MUser;
11 import com.mitrais.utils.CommonResponse;
12 import com.mitrais.utils.JsonUtil;
13
14 @RestController
15 @RequestMapping("kafka")
16 public class SimpleKafkaController extends CommonController{
17
18     @Value("${kafka.my.topic}")
19     private String topic = "";
20
21     @Autowired
22     private KafkaTemplate<String, String> kafkaStringTemplate;
23
24     @PostMapping(path="/publish", produces=MediaType.APPLICATION_JSON_VALUE)
25     public String publishKafka(@RequestBody MUser user) throws Exception {
26         kafkaStringTemplate.send(topic, "String", JsonUtil.generateJson(user));
27         CommonResponse<String> resp = new CommonResponse<>();
28         return JsonUtil.generateJson(resp);
29     }
30
31 }
32
```

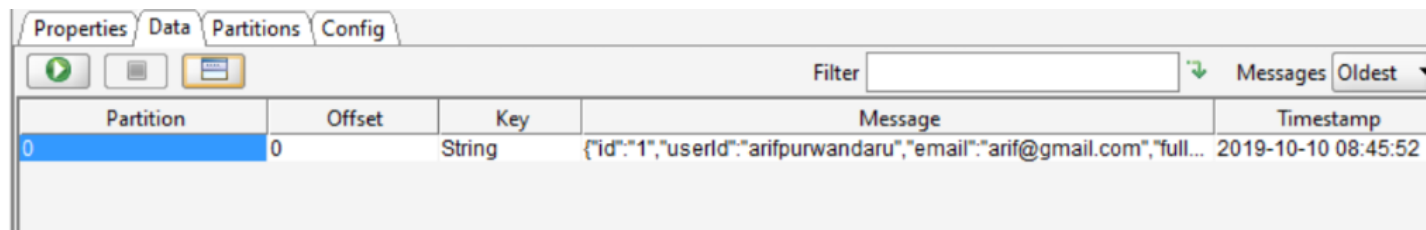
Run spring boot application by using *gradlewbootRun* command, or you can build the jar file and execute it directly with the *java -jar* command.



Let's try to hit our newly created Restful API service using the postman:



Let's check our message in Kafka tools:



Now we have 1 message in our topic.

## VI. Creating a Consumer

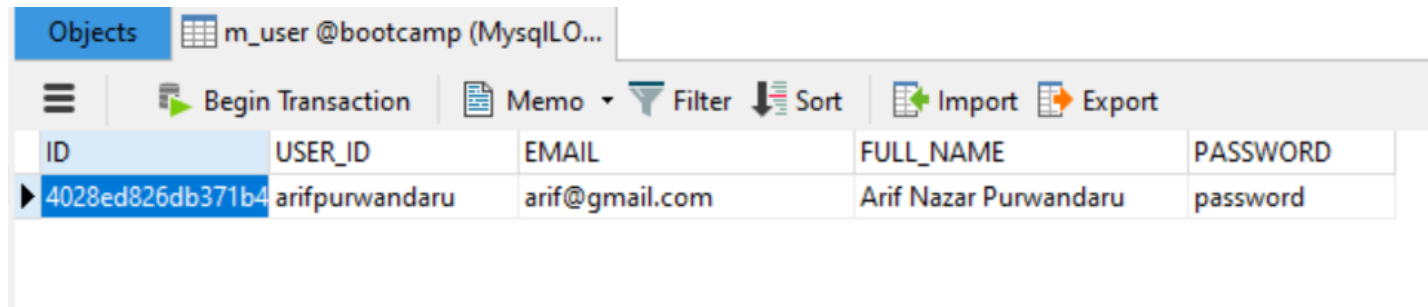
We will create the consumer within the spring boot application.

The consumer is a simple spring service that listens to the Kafka topic. Whenever a new message is sent to the topic, the listener captures it and saves to the database.

```
SimpleKafkaController.java  MessageListenerImpl.java ✕
1 package com.mitrais.kafka.service;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.kafka.annotation.KafkaListener;
5 import org.springframework.stereotype.Service;
6
7 import com.mitrais.persistence.entity.MUser;
8 import com.mitrais.persistence.service.UserDAO;
9 import com.mitrais.utils.JsonUtil;
10
11 @Service
12 public class MessageListenerImpl implements MessageListener{
13
14     @Autowired
15     private UserDAO dao;
16
17     @Override
18     @KafkaListener(topics="${kafka.my.topic}")
19     public void listen(String jsonStr) throws Exception {
20         MUser user = JsonUtil.parseJson(jsonStr, MUser.class);
21         dao.save(user);
22         System.out.println("=====> Consumed Message: "+jsonStr);
23     }
24
25 }
```

```
application.properties
2019-10-10 09:14:27.411 INFO 5888 --- [qtp313869647-16] o.a.kafka.common.utils.AppInfoParser : Kafka version : 1.0.2
2019-10-10 09:14:27.411 INFO 5888 --- [qtp313869647-16] o.a.kafka.common.utils.AppInfoParser : Kafka commitId : 2a121f7b1d402825
=====> Consumed Message: {"userId":"arifpurwandaru","email":"arif@gmail.com","fullName":"Arif Nazar Purwandaru","password":"password"}
```

If we run the spring boot, and monitor the log, we will have something like this whenever there is a message sent to the Kafka topic.



ID	USER_ID	EMAIL	FULL_NAME	PASSWORD
4028ed826db371b4	arifpurwandaru	arif@gmail.com	Arif Nazar Purwandaru	password

The message is then saved to the database.

## Conclusion

Now you have Apache Kafka running on your CentOS server with 3 brokers and 1 zookeeper. You can also produce and consume messages from your spring boot application to your Kafka topic. This makes it much easier to implement asynchronous processes in your next project. To learn more about Kafka, have a look at its documentation (<http://kafka.apache.org/documentation.html>).

Author:


Arif Nazar Purwandaru – Analyst Programmer

---

Share

 ([https://www.facebook.com/sharer/sharer.php?](https://www.facebook.com/sharer/sharer.php?u=https%3A%2F%2Fwww.mitrais.com%2Fnews-updates%2Fa-guide-to-setup-a-kafka-environment%2F)

[u=https%3A%2F%2Fwww.mitrais.com%2Fnews-updates%2Fa-guide-to-setup-a-kafka-environment%2F](https://www.facebook.com/sharer/sharer.php?u=https%3A%2F%2Fwww.mitrais.com%2Fnews-updates%2Fa-guide-to-setup-a-kafka-environment%2F))

 (<http://twitter.com/intent/tweet?url=https%3A%2F%2Fwww.mitrais.com%2Fnews-updates%2Fa-guide-to-setup-a-kafka-environment%2F>)



**in** ([http://www.linkedin.com/shareArticle?](http://www.linkedin.com/shareArticle?url=https%3A%2F%2Fwww.mitrais.com%2Fnews-updates%2Fa-guide-to-setup-a-kafka-environment%2F)

[url=https%3A%2F%2Fwww.mitrais.com%2Fnews-updates%2Fa-guide-to-setup-a-kafka-environment%2F\)](https%3A%2F%2Fwww.mitrais.com%2Fnews-updates%2Fa-guide-to-setup-a-kafka-environment%2F)

✉ ([mailto:?body=https%3A%2F%2Fwww.mitrais.com%2Fnews-updates%2Fa-guide-to-setup-a-kafka-environment%2F\)](mailto:?body=https%3A%2F%2Fwww.mitrais.com%2Fnews-updates%2Fa-guide-to-setup-a-kafka-environment%2F)

Get the latest news from us to your inbox

Email address \*

**SUBSCRIBE**

(Weekly newsletter)

## Related Articles

(<https://www.mitrais.com/news-updates/mitrais-high-quality-code/>)

(<https://www.mitrais.com/news-updates/the-rise-of-blockchain-technologies-at-blockbali-conference-2017/>)

(<https://www.mitrais.com/case-studies/mule-esb-in-ksatria-medical-system/>)

(<https://www.mitrais.com/news-updates/mitrais-high-quality-code/>)

(<https://www.mitrais.com/news-updates/the-rise-of->

(<https://www.mitrais.com/case-studies/mule-esb-in-ksatria-medical-system/>)

Mitrais High Quality  
Code  
(<https://www.mitrais.com/news-updates/mitrais-high-quality-code/>)

blockchain-technologies-at-  
blockbali-conference-2017/)  
The Rise of  
Blockchain  
Technologies at  
BlockBali Conference  
2017  
(<https://www.mitrais.com/news-updates/the-rise-of-blockchain-technologies-at-blockbali-conference-2017/>)

Mule ESB in Ksatria  
Medical Systems  
(<https://www.mitrais.com/case-studies/mule-esb-in-ksatria-medical-system/>)

## Leave a comment

Message \*

Full Name \*

E-mail \*

**Submit****Dadin**

4 months ago

Nice work Arif, keep it a good work. I have small problem with the code I got error when saving to database.

The code

```
@Override
```

```
@Transactional
```

```
public Object save(Object o) throws DataIntegrityViolationException{  
    em.persist(o);  
    return o;  
}
```

here is the error

```
org.springframework.kafka.listener.ListenerExecutionFailedException: Listener  
method `public void  
com.mitrais.kafka.service.MessageListenerImpl.listen(java.lang.String) throws  
java.lang.Exception` threw exception; nested exception is  
java.lang.IllegalArgumentException: Unknown entity: java.lang.String
```

table column name  
(id,userId,email,fullName,password)

I am new in Spring..

Thanks for sharing

Reply (<https://www.mitrais.com/news-updates/a-guide-to-setup-a-kafka-environment/?replytocom=86#respond>)



Dadin

4 months ago

I solve the problem i have wrong column name

Thanks

Reply (<https://www.mitrais.com/news-updates/a-guide-to-setup-a-kafka-environment/?replytocom=87#respond>)



Arif Nazar Purwandaru

4 months ago

Hi Dadin, Thank you for your comment. Sorry for late response, good to know you have solved the problem. Cheers :)

Reply (<https://www.mitrais.com/news-updates/a-guide-to-setup-a-kafka-environment/?replytocom=88#respond>)

## #JoinMitrais (/careers/)

(<https://www.mitrais.com/careers/windows-app-developer/>)

Software Engineer

**Microsoft Windows Application (Desktop) Developer**

📍 Bali, Bandung, Jakarta, Jogja

(<https://www.mitrais.com/careers/senior-it-engineer/>)

**Senior IT Engineer**

📍 Bali, Bandung, Jakarta, Jogja

(<https://www.mitrais.com/careers/web-front-end-developer/>)

Software Engineer

### Web Front End Developer

📍 Bali, Bandung, Jakarta, Jogja

**See All Vacancies (<https://apply.mitrais.com/>)**

English ▼

(/)

**A World Class Software Development Company**  
(<https://www.cac-holdings.com/eng/>)

#### USEFUL LINKS

> [About Us \(/about-us/\)](/about-us/)

- › [Careers \(/careers/\)](/careers/)
- › [Our Team \(/our-team/\)](/our-team/)
- › [Our Offices \(/our-offices/\)](/our-offices/)
- › [Our Training \(/training/\)](/training/)
- › [Testimonials \(/testimonials/\)](/testimonials/)

## **SOFTWARE DEVELOPMENT (/SOFTWAREDEVELOPMENT/)**

- › [Windows App \(/software-development/windows-app/\)](/software-development/windows-app/)
- › [Microsoft Web \(/software-development/web-solutions/\)](/software-development/web-solutions/)
- › [Java Development \(/software-development/java/\)](/software-development/java/)
- › [Testing & QA \(/software-development/quality-testing/\)](/software-development/quality-testing/)
- › [Web Front End \(/software-development/front-end/\)](/software-development/front-end/)
- › [PHP Development \(/software-development/php/\)](/software-development/php/)
- › [Enterprise Architect \(/software-development/enterprise-architect/\)](/software-development/enterprise-architect/)
- › [JavaScript \(/software-development/javascript/\)](/software-development/javascript/)
- › [Ruby on Rails \(/software-development/ruby-rails/\)](/software-development/ruby-rails/)
- › [UI/UX Design \(/software-development/ui-ux-design/\)](/software-development/ui-ux-design/)
- › [OutSystems \(/software-development/outsystems/\)](/software-development/outsystems/)
- › [DevOps Consulting \(/software-development/devops-consulting/\)](/software-development/devops-consulting/)
- › [Cloud Computing \(/software-development/cloud-computing/\)](/software-development/cloud-computing/)
- › [Agile Development \(/software-development/agile-development/\)](/software-development/agile-development/)
- › [HEALTHCARE \(/customhealthcare/\)](/customhealthcare/)
- › [FINANCE SECTOR \(/finance-sector-software/\)](/finance-sector-software/)

---

## **MINING SOFTWARE (/MININGSOFTWARE/)**

- › Ellipse (/mining-software/ellipse/)
- › LinkOne (/mining-software/linkone/)
- › Spry Scheduler (/mining-software/spry-scheduler/)
- › MCat & Cataloguing (/mining-software/mcat-services/)

Join the conversation



(<https://www.facebook.com/Mitrais/>)



(<https://twitter.com/Mitrais/>)



(<https://www.linkedin.com/company/mitrais/>)



(<https://www.instagram.com/mitrais/>)



(<https://www.youtube.com/mitraisbali>)

---

**CONTACT US**

*from Indonesia:*

*from Australia:*

*from New Zealand:*

0800-755-025 (tel:0800-755-025)

0361-849-7952 (tel:0361-849-7952)

*from Singapore:*

8311-1374 (tel:8311-1374)

1800-755-025 (tel:1800-755-025)

*from other countries:*

+65-6407-1331 (tel:+65-6407-1331)

© Copyright 1991 - 2020 Mitrais

[Terms & Conditions \(/terms-and-condition/\)](/terms-and-condition/) | [Privacy Policy \(/privacy-policy/\)](/privacy-policy/)