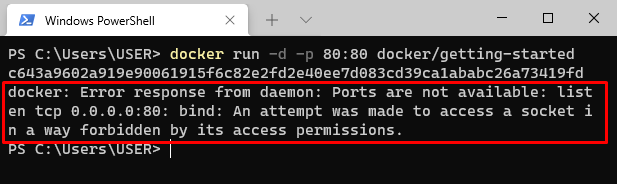
**Docker Documentation**

**Getting started:**

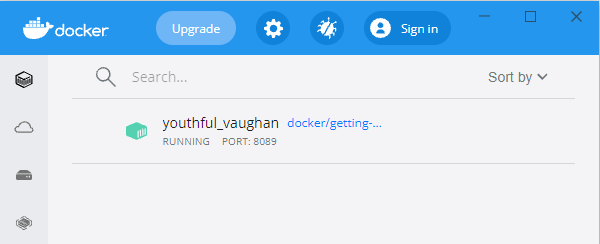
* Download and install docker desktop from [Docker.com](https://www.docker.com/)
* Test application
* Run this command on PowerShell

$ docker run -d -p 80:80 docker/getting-started

* -d - run the container in detached mode (in the background)
* -p 80:80 - map port 80 of the host to port 80 in the container
* docker/getting-started - the image to use

If it shows error like this, please change host port and try again.

* See the container list from Docker desktop



* Open your browser and try to access <http://localhost:8089>
* Change the port [8089] according to what you have entered

**Kafka & Zookeeper compose file without password:**

version: '3'

services:

  zookeeper:

    image: confluentinc/cp-zookeeper:7.0.1

    container\_name: zookeeper

    environment:

      ZOOKEEPER\_CLIENT\_PORT: 2181

      ZOOKEEPER\_TICK\_TIME: 2000

  broker:

    image: confluentinc/cp-kafka:7.0.1

    container\_name: broker

    ports:

      - "9092:9092"

    depends\_on:

      - zookeeper

    environment:

      KAFKA\_BROKER\_ID: 1

      KAFKA\_ZOOKEEPER\_CONNECT: 'zookeeper:2181'

      KAFKA\_LISTENER\_SECURITY\_PROTOCOL\_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT\_INTERNAL:PLAINTEXT

      KAFKA\_ADVERTISED\_LISTENERS: PLAINTEXT://localhost:9092,PLAINTEXT\_INTERNAL://broker:29092

      KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR: 1

      KAFKA\_TRANSACTION\_STATE\_LOG\_MIN\_ISR: 1

      KAFKA\_TRANSACTION\_STATE\_LOG\_REPLICATION\_FACTOR: 1

* Create a file named ‘docker-compose.yml’ and past the yml codes on it.
* Open terminal in same location
* Type ‘docker-compose up –d’ to create and run container for kafka & zookeeper

**Kafka & Zookeeper compose file with password:**

version: '3'

services:

  zookeeper:

    image: confluentinc/cp-zookeeper:7.0.1

    ports:

      - 2181:2181

    environment:

      ZOOKEEPER\_SERVER\_ID: 1

      ZOOKEEPER\_CLIENT\_PORT: 2181

      ZOO\_MY\_ID: 1

      ZOO\_PORT: 2181

      ZOOKEEPER\_SERVERS: server.1=zookeeper:2888:3888

      ZOOKEEPER\_SASL\_ENABLED: "false"

    volumes:

      - ./zoo1/data:/data

      - ./zoo1/datalog:/datalog

  kafka:

    image: confluentinc/cp-kafka:7.0.1

    ports:

      - "9092:9092"

    environment:

      KAFKA\_LISTENERS: SASL\_PLAINTEXT://:9092

      KAFKA\_ADVERTISED\_LISTENERS: SASL\_PLAINTEXT://localhost:9092

      KAFKA\_ZOOKEEPER\_CONNECT: zookeeper:2181

      ZOOKEEPER\_SASL\_ENABLED: "false"

      KAFKA\_OPTS: "-Djava.security.auth.login.config=/etc/kafka/kafka\_server\_jaas.conf"

      KAFKA\_INTER\_BROKER\_LISTENER\_NAME: SASL\_PLAINTEXT

      KAFKA\_SASL\_ENABLED\_MECHANISMS: PLAIN

      KAFKA\_SASL\_MECHANISM\_INTER\_BROKER\_PROTOCOL: PLAIN

      KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR: 1

    volumes:

      - /var/run/docker.sock:/var/run/docker.sock

      - ./kafka\_server\_jaas.conf:/etc/kafka/kafka\_server\_jaas.conf

      - ./data:/var/lib/kafka/data

    links:

      - zookeeper

* Create a file named ‘docker-compose.yml’ and past the yml codes on it.

KafkaServer {

  org.apache.kafka.common.security.plain.PlainLoginModule required

  username="admin"

  password="pass123"

  user\_admin="pass123"

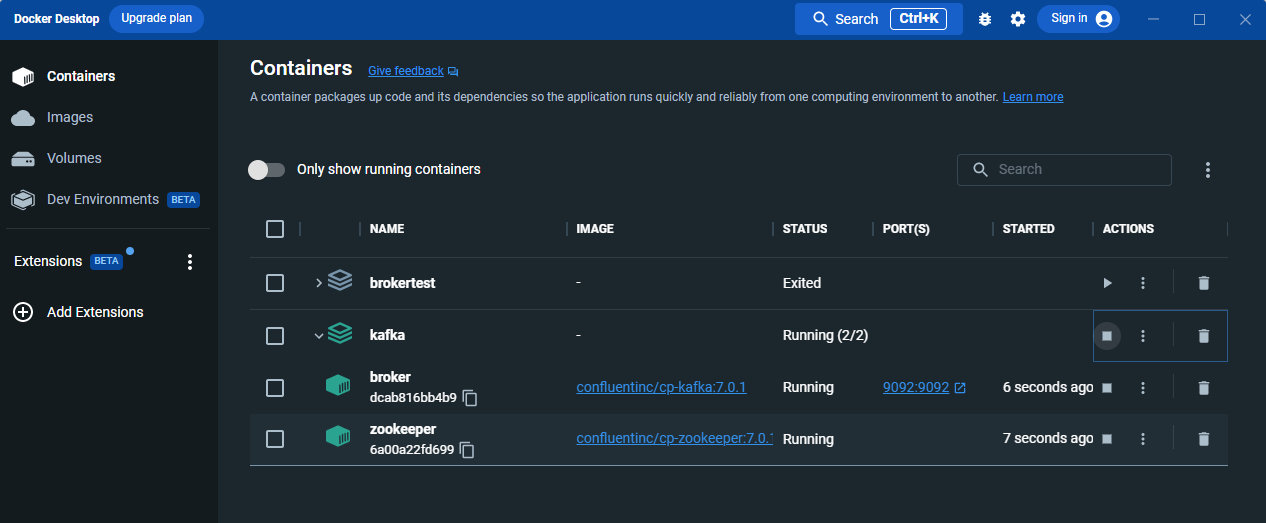
  user\_alice="alice-pass123";

};

Client{};

* Create a file named ‘kafka\_server\_jaas.conf’ and past the codes on it.
* Open terminal in same location
* Type ‘docker-compose up –d’ to create and run container for kafka & zookeeper

After successfully run the containers you will see docker desktop like this:



**Add Topic on brocker:**

docker exec broker kafka-topics --bootstrap-server broker:9092 --create --topic TopicName

**Asp.net application:**

* Open your asp.net core application and go NuGet package manager and add a package named ‘Confluent.Kafka’

**Create an interface for Produce message in kafka topic:**

**IProducer.cs:**

public interface IProducer

{

Task AddMsgOnTopic(string topicName, string msg);

}

**Producer.cs :**

public class Producer : IProducer

{

private readonly ProducerConfig \_producerConfig;

public Producer()

{

\_producerConfig = new ProducerConfig

{

#region add those lines for kafka SASL security

SaslMechanism = SaslMechanism.Plain,

SecurityProtocol = SecurityProtocol.SaslPlaintext,

SaslUsername = "admin",

SaslPassword = "pass123",

#region add those lines for kafka SASL security

BootstrapServers = $"127.0.0.1:9092",

EnableDeliveryReports = true,

ClientId = Dns.GetHostName(),

Debug = "msg",

// retry settings:

// Receive acknowledgement from all sync replicas

Acks = Acks.All,

// Number of times to retry before giving up

MessageSendMaxRetries = 3,

// Duration to retry before next attempt

RetryBackoffMs = 1000,

// Set to true if you don't want to reorder messages on retry

EnableIdempotence = true

};

}

public async Task AddMsgOnTopic(string topicName, string msg)

{

var producer = new ProducerBuilder<long, string>(\_producerConfig)

.SetKeySerializer(Serializers.Int64)

.SetValueSerializer(Serializers.Utf8)

.SetLogHandler((\_, message) =>

Console.WriteLine($"Facility: {message.Facility}-{message.Level} Message: {message.Message}"))

.SetErrorHandler((\_, e) => { })

.Build();

try

{

var deliveryReport = await producer.ProduceAsync(topicName,

new Message<long, string>

{

Key = DateTime.UtcNow.Ticks,

Value = msg

});

if (deliveryReport.Status != PersistenceStatus.Persisted)

{

// delivery might have failed after retries. This message requires manual processing.

// Console.WriteLine($"ERROR: Message not ack'd by all brokers (value: '{message}'). Delivery status: {deliveryReport.Status}");

}

}

catch (ProduceException<long, string> e)

{

// Log this message for manual processing.

Console.WriteLine($"Permanent error: {e.Message} for message (value: '{e.DeliveryResult.Value}')");

Console.WriteLine("Exiting producer...");

}

}

}

**Add scope service in Startup.cs:**

services.AddScoped<IProducer, Producer>();

**Add message from manager:**

\_producer.AddMsgOnTopic("TopicName", "message");

**Create an interface for Consume message from kafka topic:**

**IConsumer.cs:**

public interface IConsumer

{

void StartReceivingMessages(CancellationToken cancellationToken, string topicName);

}

**Consumer.cs:**

public class Consumer : BackgroundService, IConsumer

{

private readonly ConsumerConfig \_consumerConfig;

public IServiceProvider Services { get; }

public Consumer(IServiceProvider services)

{

Services = services;

\_consumerConfig = new ConsumerConfig

{

#region add those lines for kafka SASL security

SaslMechanism = SaslMechanism.Plain,

SecurityProtocol = SecurityProtocol.SaslPlaintext,

SaslUsername = "admin",

SaslPassword = "pass123",

#endregion add those lines for kafka SASL security

BootstrapServers = $"127.0.0.1:9092",

EnableAutoCommit = false,

EnableAutoOffsetStore = false,

MaxPollIntervalMs = 300000,

GroupId = "default",

// Read messages from start if no commit exists.

AutoOffsetReset = AutoOffsetReset.Earliest

};

}

public async void StartReceivingMessages(CancellationToken cancellationToken, string topicName)

{

var consumer = new ConsumerBuilder<long, string>(\_consumerConfig)

.SetKeyDeserializer(Deserializers.Int64)

.SetValueDeserializer(Deserializers.Utf8)

.SetErrorHandler((\_, e) => { })

.Build();

try

{

consumer.Subscribe(topicName);

while (!cancellationToken.IsCancellationRequested)

{

var result = consumer.Consume(TimeSpan.FromMilliseconds(\_consumerConfig.MaxPollIntervalMs - 1000 ?? 250000));

var message = result?.Message?.Value;

if (string.IsNullOrEmpty(message))

continue;

using (var scope = Services.CreateScope())

{

var manager = scope.ServiceProvider.GetRequiredService<IEmailManager>();

manager.SendMail(message);

}

consumer.Commit(result);

consumer.StoreOffset(result);

}

}

catch (KafkaException e)

{

}

finally

{

consumer.Close();

}

}

protected override Task ExecuteAsync(CancellationToken stoppingToken)

{

Task.Run(() => StartReceivingMessages(stoppingToken, "TopicName"), stoppingToken);

return Task.CompletedTask;

}

}

**Add scope service in Startup.cs:**

services.AddHostedService<Consumer>();

services.AddScoped<IConsumer, Consumer>();