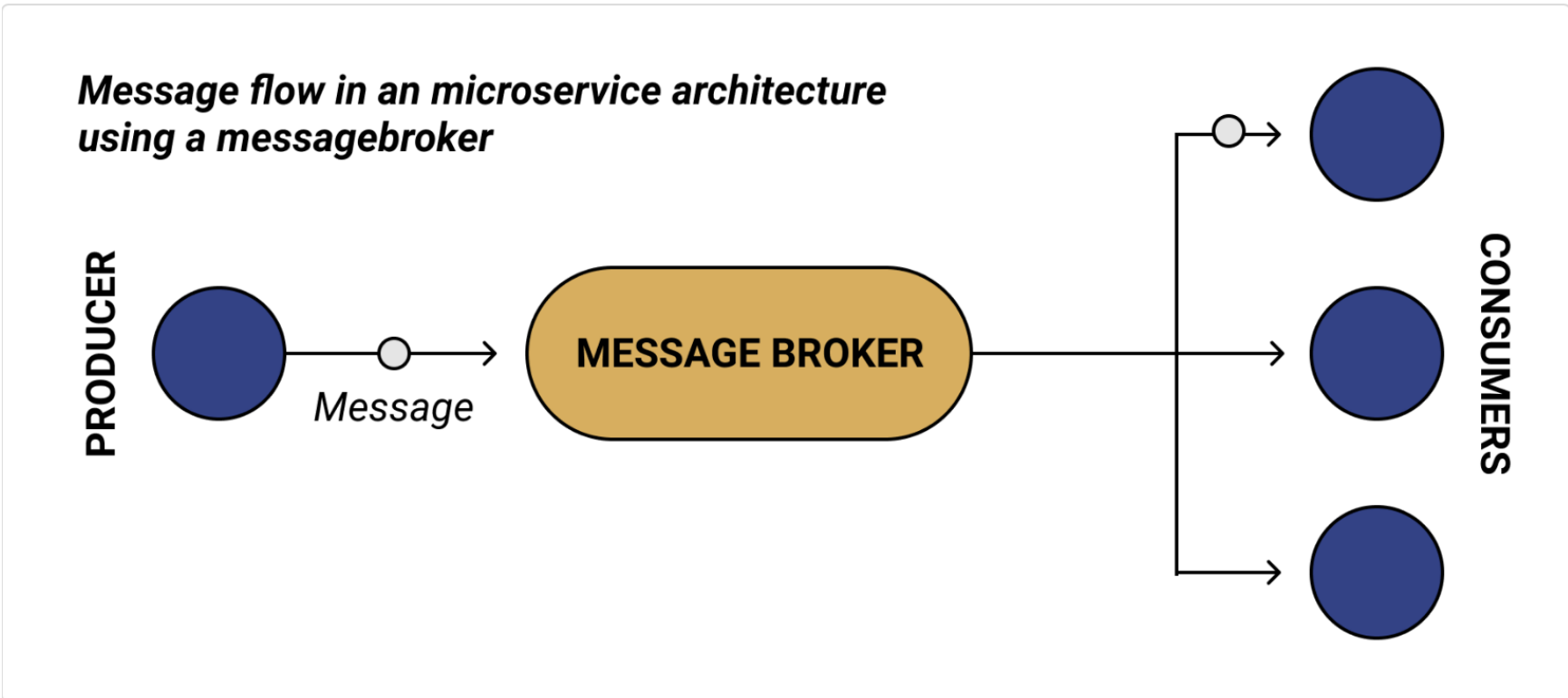
Springboot With Asynchronous Communication

RabbitMQ

kafka

Architectural style



Apache ActiveMQ is an open source message broker written in Java together with a full Java Message Service (JMS) client.

There's another broker under the ActiveMQ umbrella code-named Artemis. It is based on the HornetQ code-base which was donated[4] from the JBoss community to the Apache ActiveMQ community in 2015. Artemis is the "next generation" broker from ActiveMQ and will ultimately become the next major version of ActiveMQ.[5]

Code Deep Dive

Producer-
consumer

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-activemq</artifactId>
</dependency>
```

```java
@Configuration
@EnableRetry
public class ArtemisConfig {
    private static final Logger LOGGER = LoggerFactory.getLogger(ArtemisConfig.class);
    @Value("${artemis.enable.ssl}")
    private String sslEnabled;
    @Value("${artemis.brokerUrl}")
    private String brokerUrl;
    @Value("${artemis.user}")
    private String user;
    @Value("${artemis.password}")
    private String password;
    @Value("${artemis.connection.cache.size}")
    private int sessionCacheSize;

    @Bean(name = "jmsConnectionFactory")
    public CachingConnectionFactory cachingConnectionFactory() {
        ActiveMQConnectionFactory activeMQConnectionFactory = new ActiveMQConnectionFactory();
        activeMQConnectionFactory.setBrokerURL(brokerUrl);
        activeMQConnectionFactory.setUserName(user);
        activeMQConnectionFactory.setPassword(password);
        activeMQConnectionFactory.setTrustedPackages(Collections.singletonList("com.rama.artemis"));
        CachingConnectionFactory cachingConnectionFactory = new CachingConnectionFactory(activeMQConnectionFactory);
        cachingConnectionFactory.setSessionCacheSize(sessionCacheSize);
        return cachingConnectionFactory;
    }
}
```

```
spring.artemis.mode=native
#spring.artemis.host=localhost
#spring.artemis.port=61618
artemis.user=admin
artemis.password=admin
jms.queue.destination=myqueue
server.port=1220
artemis.brokerUrl=tcp://localhost:61618
artemis.enable.ssl=false
artemis.connection.cache.size=5
```

Retry Producer Important!

```
########################################
#########Artemis Retry###################
########################################
artemis.retry.maxattempt= 4
artemis.retry.delay= 1000
artemis.retry.maxdelay= 5000
artemis.retry.multiplier=2
```

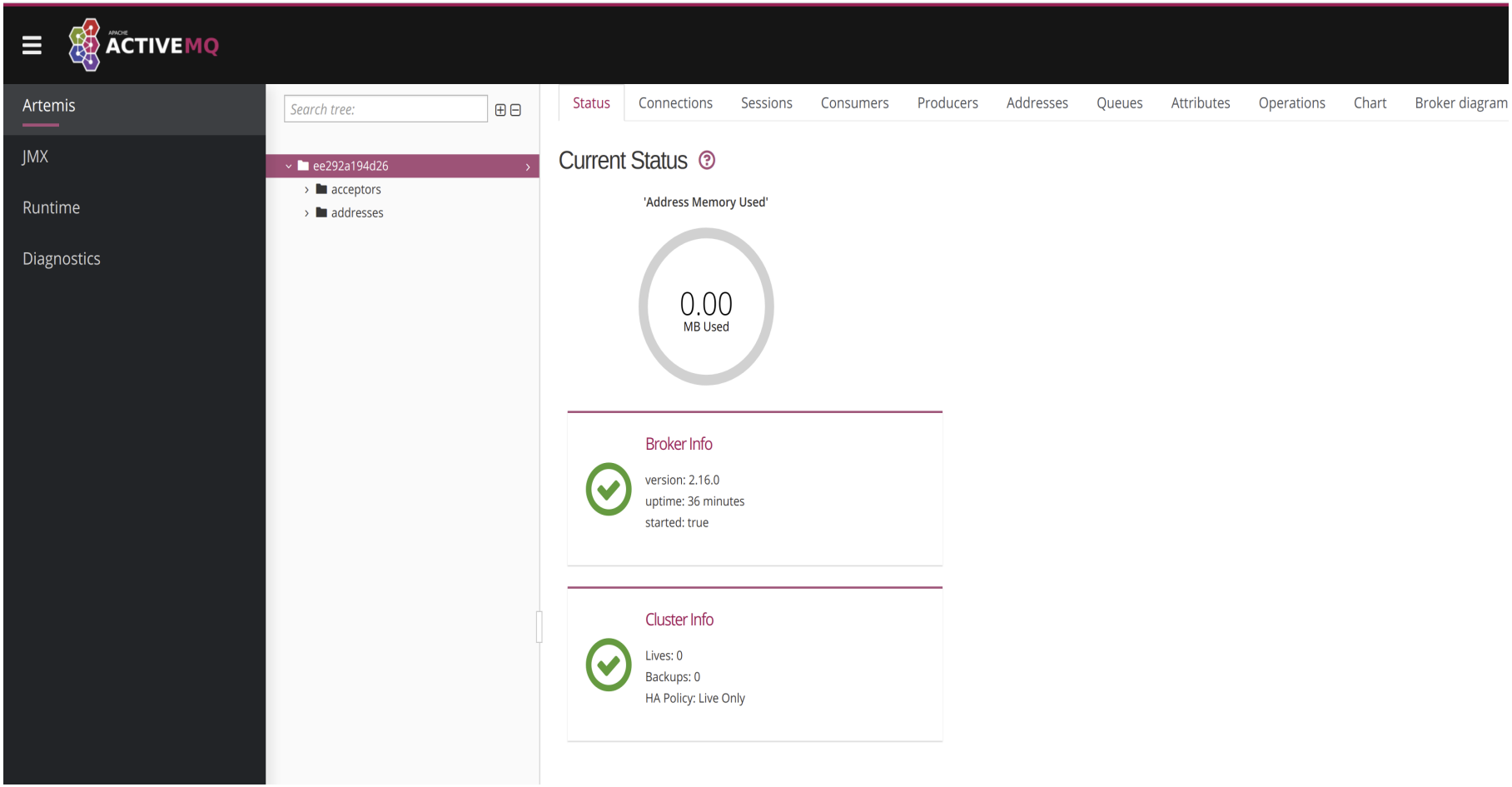
```
public ArtemisProducer(JmsTemplate jmsTemplate) { this.jmsTemplate = jmsTemplate; }

/* @Retryable(
         value = {UncategorizedJmsException.class},
         maxAttempts = 10,
         backoff = @Backoff(random = true, delay = 1000, maxDelay = 8000,multiplier = 2)
 )*/
@Retryable(value = { UncategorizedJmsException.class }, maxAttemptsExpression = "${artemis.retry.maxattempt}",
           backoff = @Backoff(random = true, delayExpression = "${artemis.retry.delay}",
                              maxDelayExpression = "${artemis.retry.maxdelay}", multiplierExpression = "${artemis.retry.multiplier}"))
public void send(String msg) {
    LOGGER.info("Sending Data:");
    jmsTemplate.convertAndSend(destinationQueue, msg);
    LOGGER.info("Data Sent:");
}
```

Consumer

```java
@Component
public class ArtemisConsumer {
    @JmsListener(destination = "${jms.queue.destination}")
    public void receive(String msg) { System.out.println("Got Message: " + msg); }
}
```

Start Artemis in docker:

```yaml
version: '3.2'
services:
  artemis:
    image: vromero/activemq-artemis:latest
    ports:
      - "61618:61616"
    environment:
      ARTEMIS_USERNAME: admin
      ARTEMIS_PASSWORD: admin
      ARTEMIS_MIN_MEMORY: 512M
      ARTEMIS_MAX_MEMORY: 2048M
```


ACTIVEMQ
Artemis
JMX
Runtime
Diagnostics
Search tree:
ee292a194d26
acceptors
addresses
Status
Connections
Sessions
Consumers
Producers
Addresses
Queues
Attributes
Operations
Chart
Broker diagram
Current Status
'Address Memory Used'
0.00
MB Used
Broker Info
version: 2.16.0
uptime: 36 minutes
started: true
Cluster Info
Lives: 0
Backups: 0
HA Policy: Live Only

Lab4:Create Your first artemis project

Create a springboot project which sends the message artemis with retry mechanism(Try message with request body both format json, xml)

Input:
Auto message communication :

Message:
to,from,content,correlationid(uuid)

Operation:
Send message
Receive message

Run and test through curl and postman

RabbitMQ is an open-source message-broker software (sometimes called message-oriented middleware) that originally implemented the Advanced Message Queuing Protocol (AMQP) and has since been extended with a plug-in architecture to support Streaming Text Oriented Messaging Protocol (STOMP), MQ Telemetry Transport (MQTT), and other protocols.

Code Deep Dive

Producer-
consumer

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
```

Producer

```java
@Configuration
public class RabbitMQConfig {

  @Value("${rabbitmq.queue.name}")
  String queueName;

  @Value("${rabbitmq.exchange.name}")
  String exchange;

  @Value("${rabbitmq.routingkey.name}")
  private String routingkey;

  @Bean
  Queue queue() { return new Queue(queueName,  durable: false); }

  @Bean
  DirectExchange exchange() { return new DirectExchange(exchange); }

  @Bean
  Binding binding(Queue queue, DirectExchange exchange) { return BindingBuilder.bind(queue).to(exchange).with(routingkey); }

  @Bean
  public MessageConverter jsonMessageConverter() { return new Jackson2JsonMessageConverter(); }


  @Bean
  public AmqpTemplate rabbitTemplate(ConnectionFactory connectionFactory) {
    final RabbitTemplate rabbitTemplate = new RabbitTemplate(connectionFactory);
    rabbitTemplate.setMessageConverter(jsonMessageConverter());
    return rabbitTemplate;
  }
}
```

```java
@Service
public class RabbitMQSender {
  @Autowired
  private AmqpTemplate rabbitTemplate;

  @Value("${rabbitmq.exchange.name}")
  private String exchange;

  @Value("${rabbitmq.routingkey.name}")
  private String routingkey;

  public void send(Employee company) {
    rabbitTemplate.convertAndSend(exchange, routingkey, company);
    System.out.println("Send msg = " + company);

  }
}
```

```
spring.rabbitmq.host=localhost
spring.rabbitmq.port=5672
spring.rabbitmq.username=guest
spring.rabbitmq.password=guest
rabbitmq.exchange.name=test.exchange
rabbitmq.queue.name=test.queue
rabbitmq.routingkey.name=test.routingkey
spring.main.allow-bean-definition-overriding=true
```

Consumer

```
spring.rabbitmq.host=localhost
spring.rabbitmq.port=5672
spring.rabbitmq.username=guest
spring.rabbitmq.password=guest
rabbitmq.queue.name=test.queue
server.port=8082
```

```java
import com.rama.rabbitmq.model.Employee;
import org.springframework.amqp.rabbit.annotation.RabbitListener;
import org.springframework.stereotype.Component;

@Component
public class RabbitMQConsumer {

  @RabbitListener(queues = "${rabbitmq.queue.name}")
  public void recievedMessage(Employee employee) { System.out.println("Recieved Message From RabbitMQ: " + employee); }
}
```

```yaml
version: "3"
services:
  rabbitmq:
    image: "rabbitmq:3-management"
    ports:
      - "5672:5672"
      - "15672:15672"
    volumes:
      - 'rabbitmq_data:/data'

volumes:
  rabbitmq_data:
```

localhost:15672/#/queues/%2F/test.queue

Apps Work Study car school

RabbitMQ™ RabbitMQ 3.9.5 Erlang 24.0.5

Overview Connections Channels Exchanges **Queues** Admin

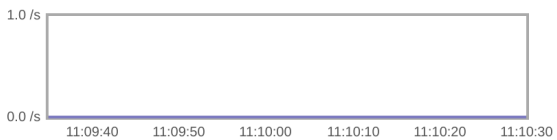
Queue test.queue

Overview

Queued messages last minute ?

Ready	1
Unacked	0
Total	1

Message rates last minute ?



Publish	0.00/s	Consumer ack	0.00/s	Get (auto ack)	0.00/s
Deliver (manual ack)	0.00/s	Redelivered	0.00/s	Get (empty)	0.00/s
Deliver (auto ack)	0.00/s	Get (manual ack)	0.00/s		

Details

Features	State	idle	
Policy	Consumers	0	
Operator policy	Consumer capacity ?	0%	
Effective policy definition			

	Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Messages ?	1	1	0	0	0	
Message body bytes ?	38 B	38 B	0 B	38 B	0 B	0 B
Process memory ?	11 kiB					

Consumers

Lab4:Create Your first Rabbit project

Create a springboot project which sends the message rabbitmq with retry mechanism(Try message with request body both format json, xml)

Input:
Auto message communication :

Message:
to,from,content,correlationid(uuid)

Operation:
Send message
Receive message

Run and test through curl and postman

THANKS !