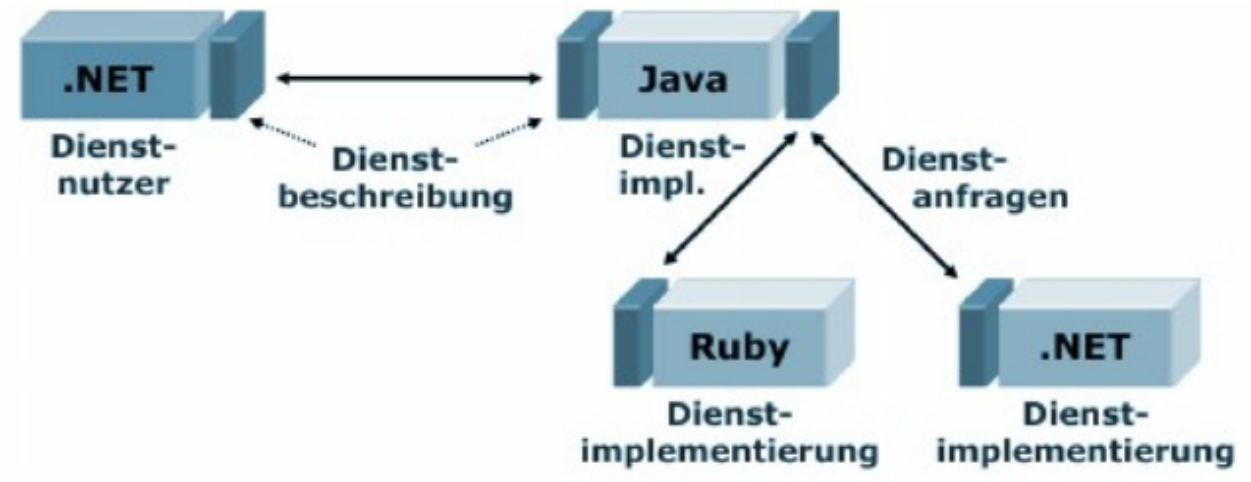
RabbitMQ Präsentation

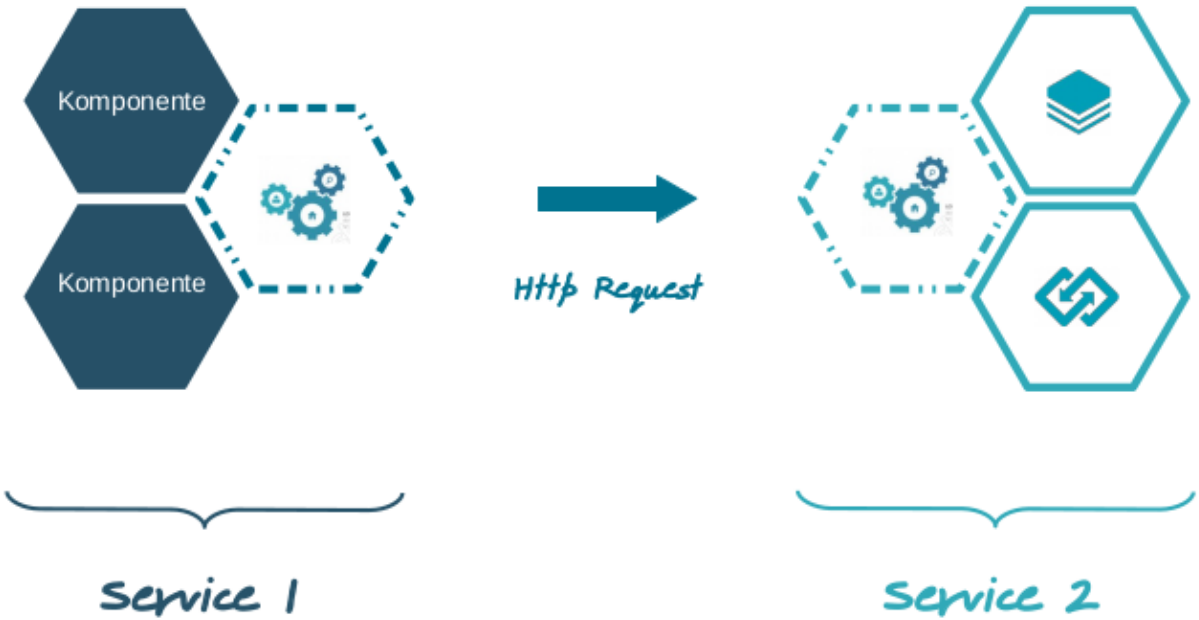
Dipl.-Ing. Paul Panhofer Bsc.

1. Messagebroker
   - MOM - Message Oriented Middleware
   - MOM Protokolle
2. RabbitMQ - AMQP
   - AMQP Artefakte
3. Spring Boot - Producer, Consumer
   - RabbitTemplate, RabbitListener

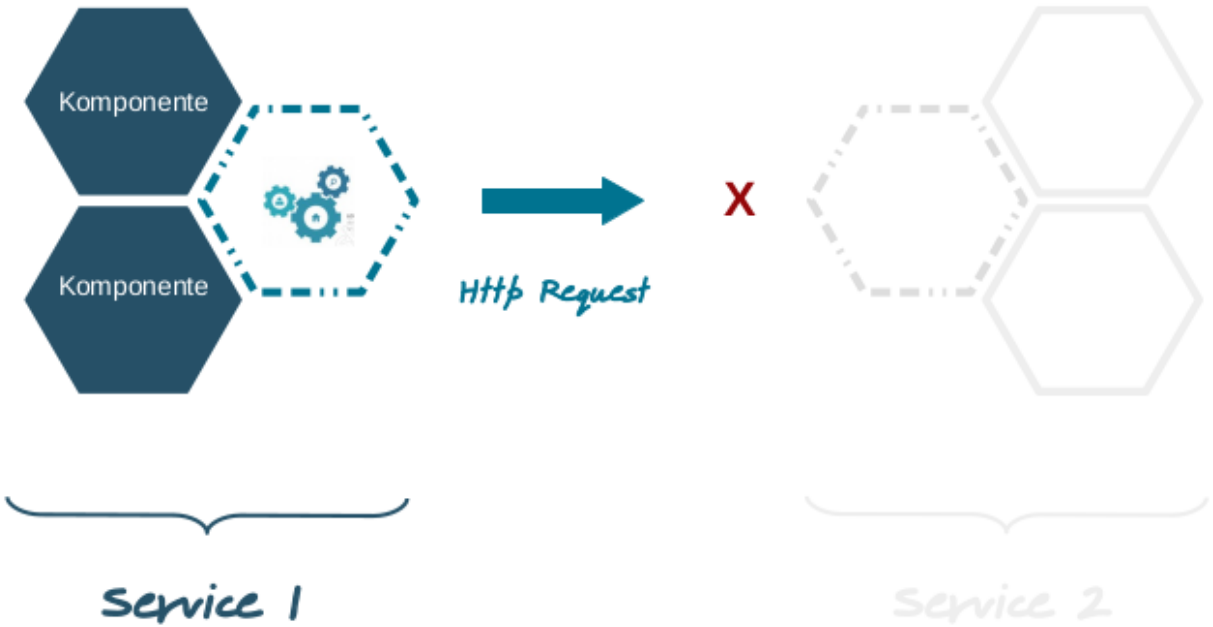
SOA - Rest Kommunikation



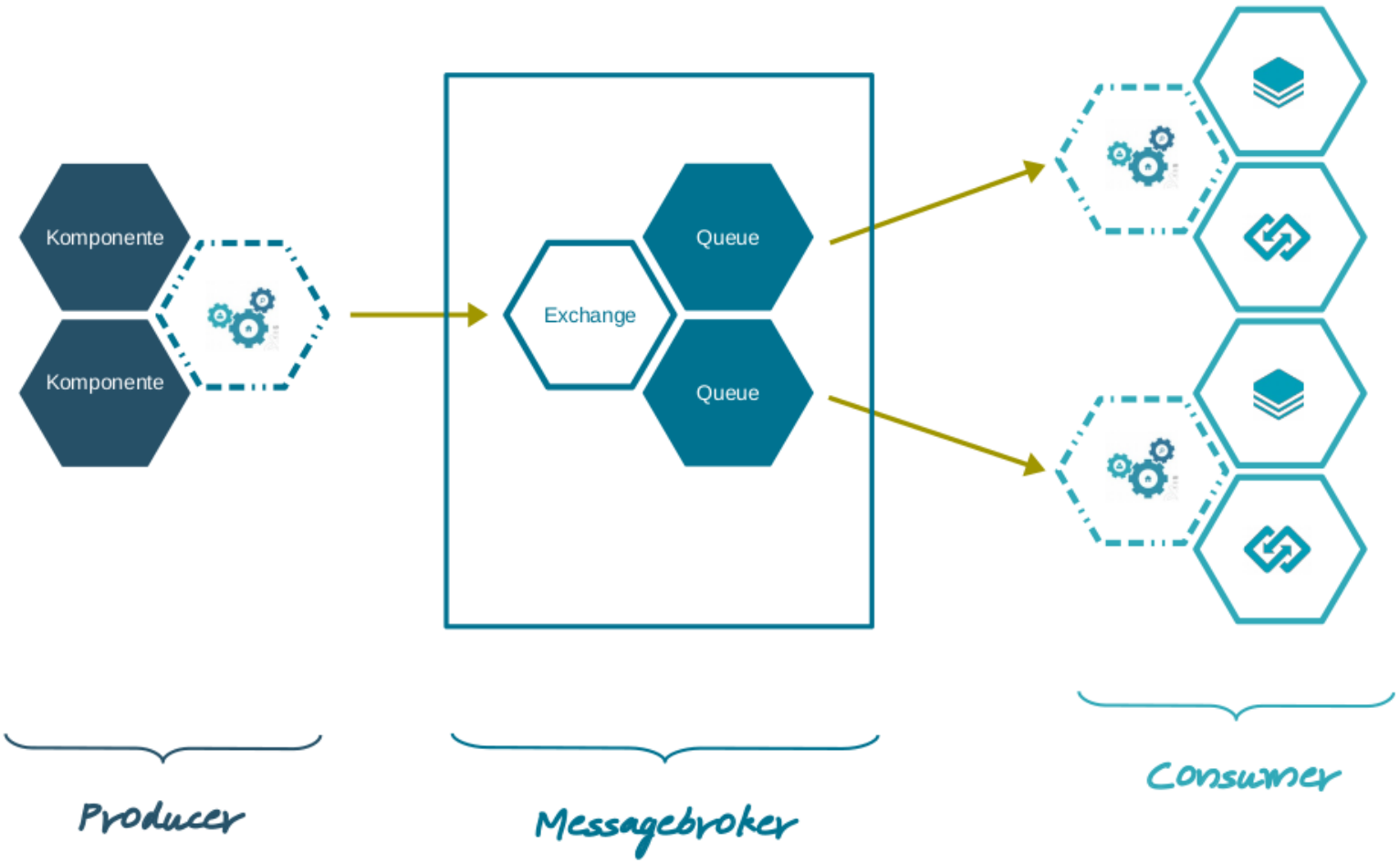
SOA - Rest Kommunikation



SOA - Rest Kommunikation



Messagebroker



MOM Kommunikation

Messagebroker

Ein Messagebroker ist ein Architekturmuster, das beschreibt, wie der **Nachrichtenaustausch** zwischen den Teilen einer Anwendung erfolgt.

- Mit dem Messagebroker erfolgt eine Entkoppelung des Produzenten einer Nachricht von ihrem Empfänger.

MOM Kommunikation

Die Verarbeitung von Nachrichten erfolgt in einem MOM Sytem **asynchron**.

- Kommt es zu einem Ausfall einer der beiden Kommunikationspartner wird die Nachricht für die Dauer des Ausfalls zwischengespeichert. Sobald die Kommunikation wieder hergestellt ist kann die Nachricht verschickt werden.

Eigenschaften von MOM

Die Kommunikation über MOM zeichnet sich durch eine Reihe von Eigenschaften gegenüber anderen Kommunikationsprotokollen aus:

- asynchrone Kommunikation
- Schwache Koppelung
- Zuverläßigkeit

Eigenschaften von MOM

Asynchrone Kommunikation

Asynchrone Kommunikation: Das Verschicken einer Nachricht **blockiert** den Sender **nicht**.

Eigenschaften von MOM

Schwache Koppelung

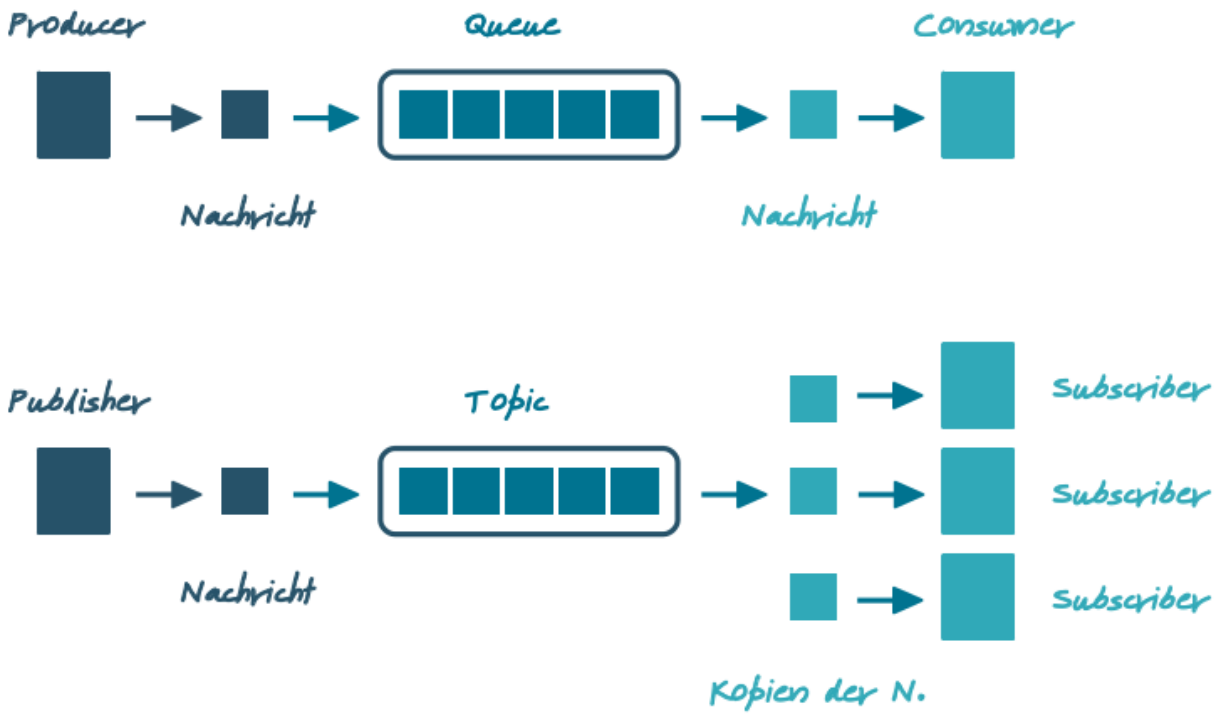
Schwache Koppelung: Der Sender **kennt** den Empfänger der Nachricht **nicht**.

Eigenschaften von MOM

Zuverläßigkeit

Zuverläßigkeit: Nachrichten überleben auch den Ausfall des Netzwerks. Der Messagebroker **speichert** die Nachrichten und stellt sie zu, wenn das Netzwerk wieder zur Verfügung steht.

Kommunikationsmuster



Usecases MOM

- Banken und Versicherung verwenden Messagebroker um Millionen von Nachrichten der Börsen in kurzer Zeit verläßlich verarbeiten zu können.
- Wetterstationen schicken ihre Daten an Messagebroker zur weiteren Verarbeitung.
- Bestellsysteme (Black Friday) leiten Kunden-bestellungen an Messagebroker weiter.

1. Messagebroker
   - MOM - Message Oriented Middleware
   - MOM Protokolle
2. RabbitMQ - AMQP
   - AMQP Artefakte
3. Spring Boot - Producer, Consumer
   - RabbitTemplate, RabbitListener

MOM Protokolle

MOM Systeme unterstützen eine Vielzahl verschiedener Protokolle. Die 2 bedeutensten sind:

- STOMP
- AMQP

MOM Protokolle

STOMP - Simple Text Oriented Protocol

STOMP wird in erster Linie zum Verschicken einfacher textbasierter Nachrichten verwendet.

- STOMP basiert auf einem einfachen textbasierten Format zur Kommunikation.
- Das Protokoll unterstütz **keine komplexen** Szenarien für das Verschicken von Nachrichten.

MOM Protokolle

AMQP - Advanced Message Queue Protocol

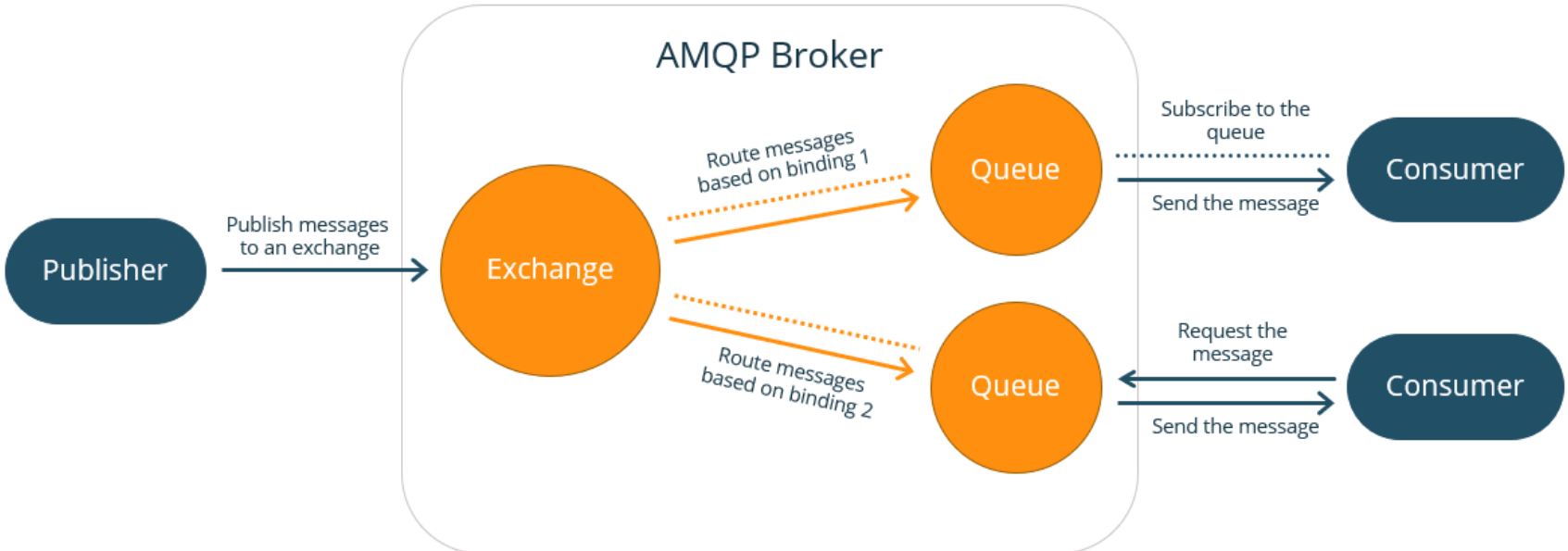
AMQP wird in Szenarien eingesetzt die komplexe Kommunikationsvorgänge abbilden.

- Im AMPQ Model wird die Nachricht nicht direkt an eine Queue sondern an einen sogenannten Exchange geschickt.
- Der **Exchange** routet die Nachricht dann an eine oder mehrere Queues weiter.
- Das Protokoll unterstützt dabei unterschiedliche Formen des Routings für Exchanges.

1. Messagebroker
   - MOM - Message Oriented Middleware
   - MOM Protokolle
2. RabbitMQ - AMQP
   - AMQP Artefakte
3. Spring Boot - Producer, Consumer
   - RabbitTemplate, RabbitListener


spring
RabbitMQ

AMQP Artefakte



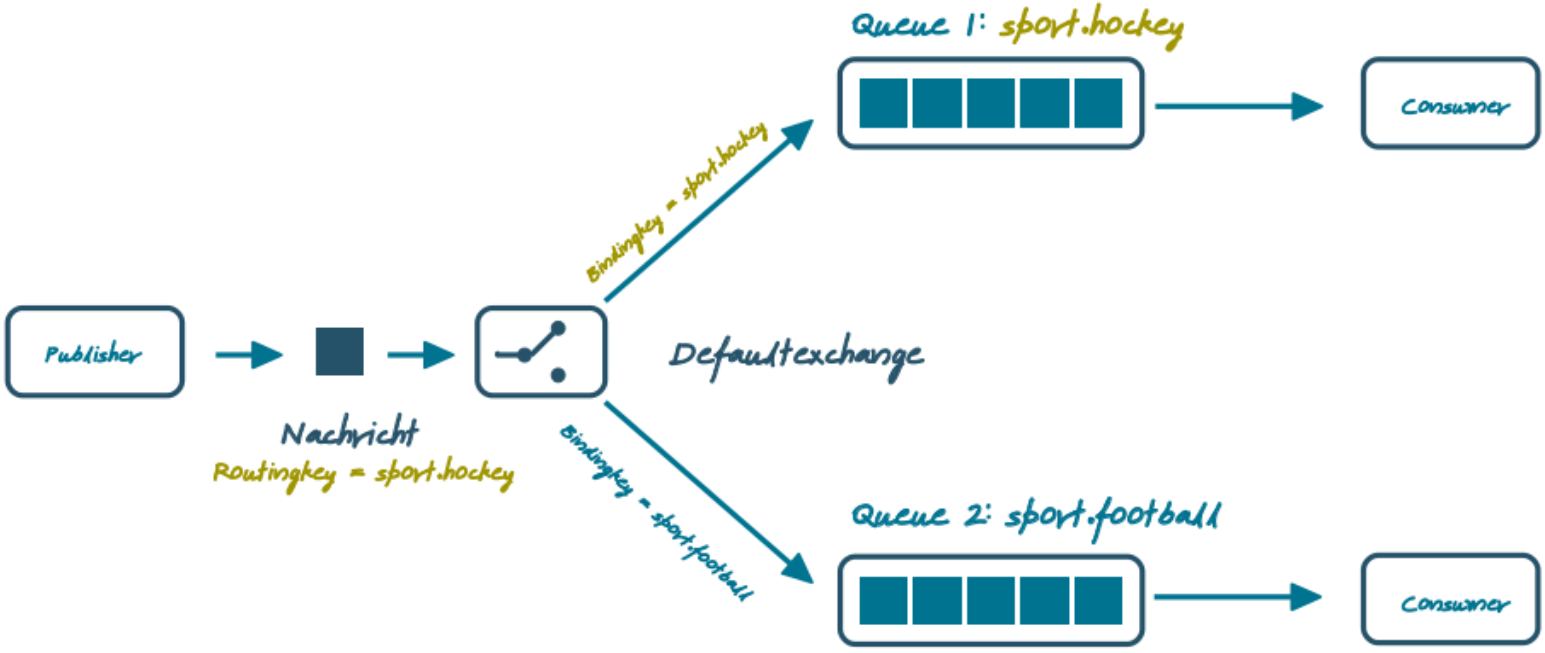
AMQP Artefakt: Exchange

Exchangetypen

Das AMQP Protokoll unterstützt mehrere Exchangetypen. Je nach Exchangetyp wird ein unterschiedliches **Routing** der Nachrichten durchgeführt.

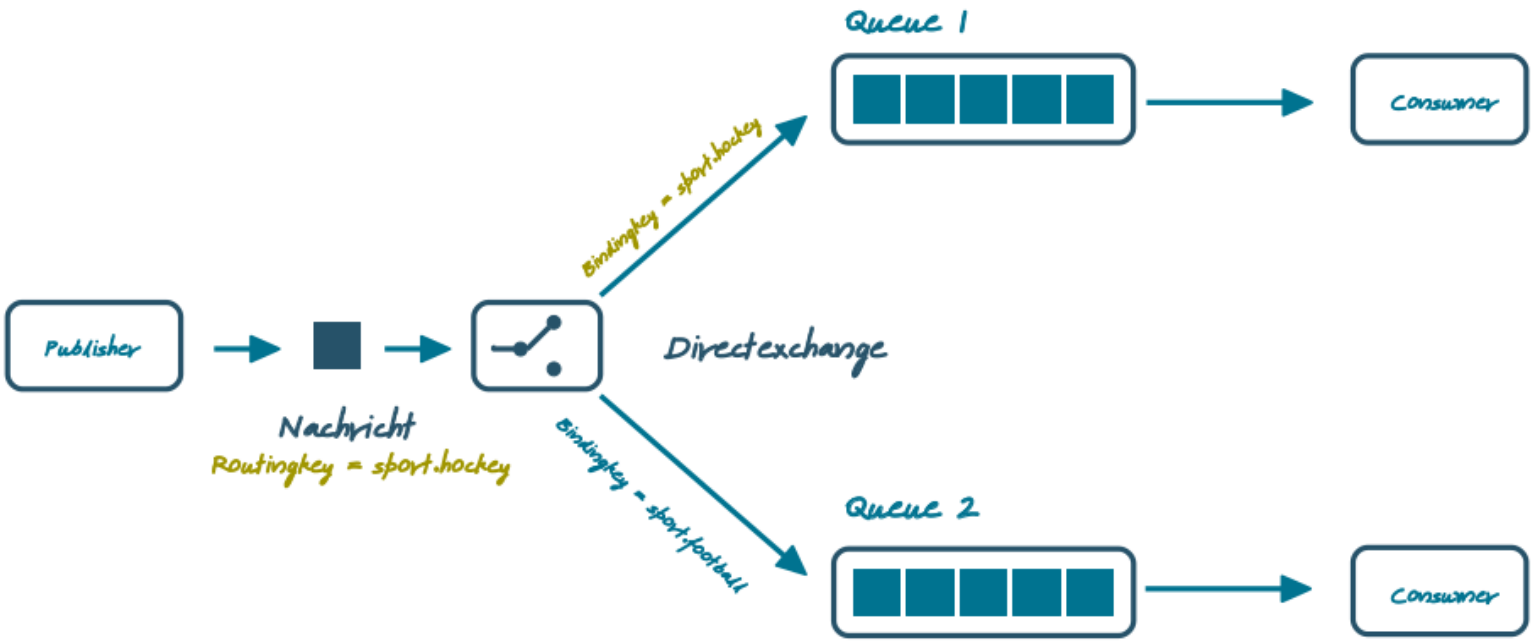
Exchangetypen

Defaultexchange



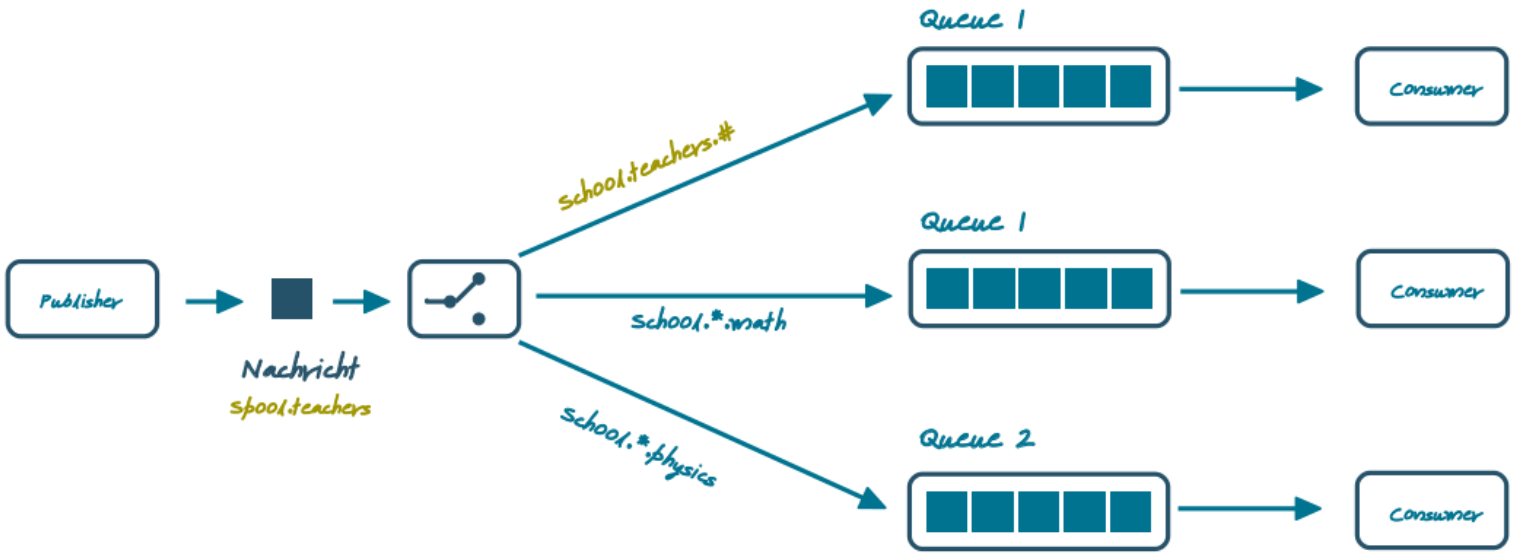
Exchangetypen

Directexchange



Exchangetypen

Topicexchange

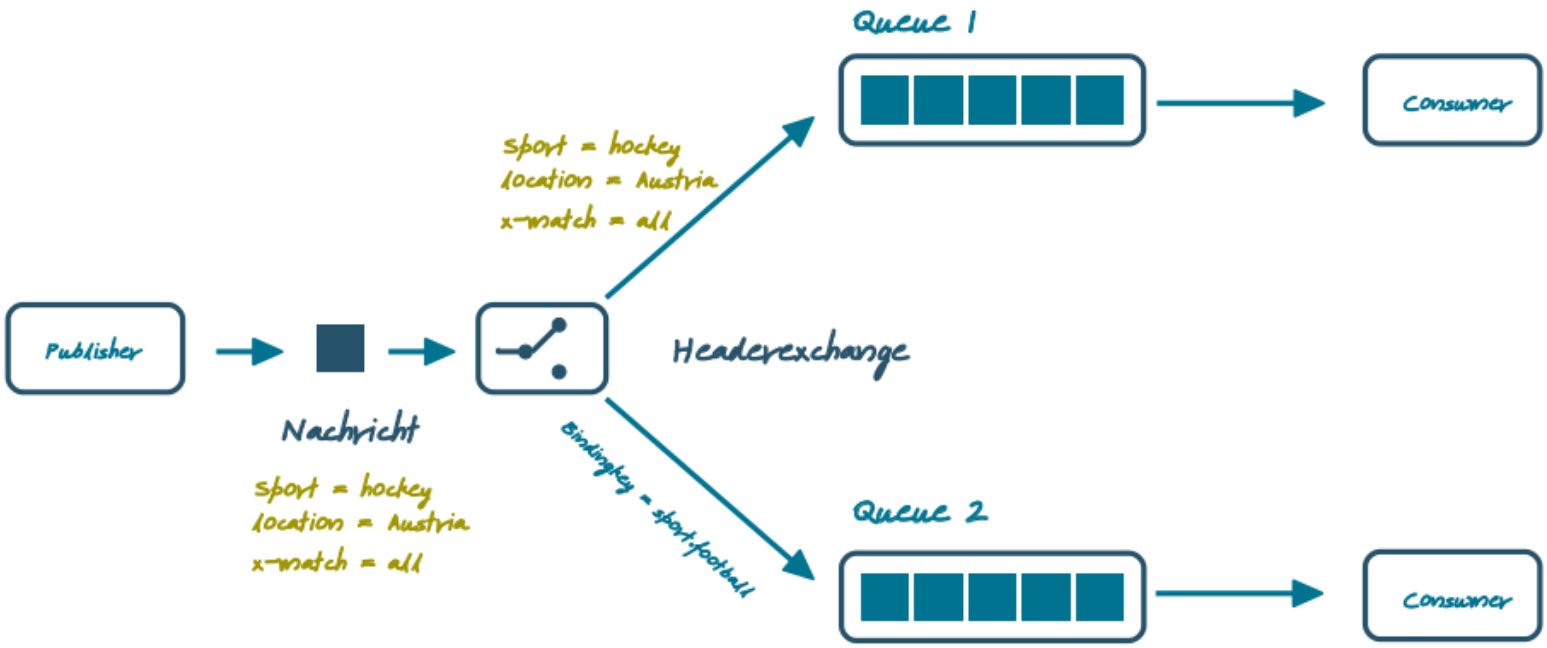


... substistute for one word or more words

* ... substistute for exactly one word

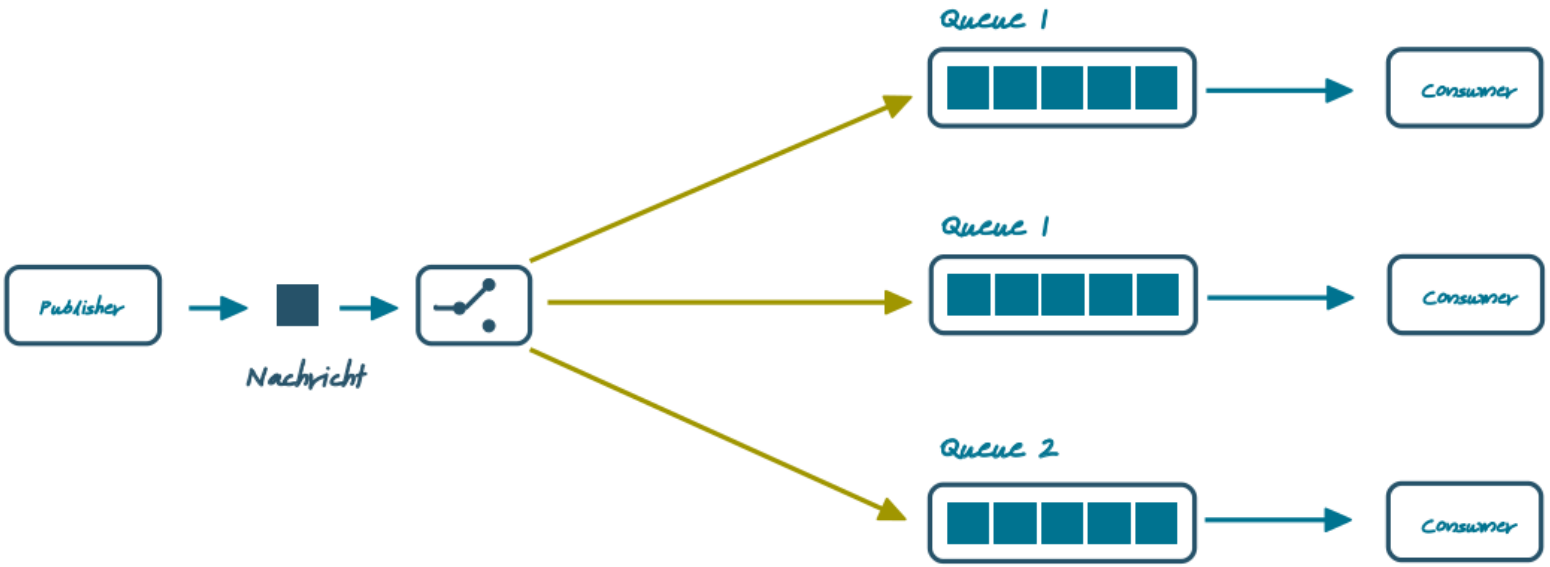
Exchangetypen

Headerexchange



Exchangetypen

Fanoutexchange



1. Messagebroker
   - MOM - Message Oriented Middleware
   - MOM Protokolle
2. RabbitMQ - AMQP
   - AMQP Artefakte
3. Spring Boot - Producer, Consumer
   - RabbitTemplate, RabbitListener

Spring Boot Messaging

Spring Boot **Messagingartefakte**:

- Configuration: application.properties
- Producer: RabbitTemplate
- Consumer: RabbitListener

Spring Boot Messaging

Configuration

```
spring.rabbitmq.host=127.0.0.1
spring.rabbitmq.port=5672
spring.rabbitmq.username=htl
spring.rabbitmq.password=htl
```

Spring Boot Messaging

Producer: DirectExchange

```
@Component
public class DirectRabbitMQProducer {

    @Autowired
    private RabbitTemplate rabbitTemplate;

    public void sendDirectMessage(String message){
        rabbitTemplate.convertAndSend(
            "exchange_name",
            "routing_key",
            message
        );
    }

}
```

Spring Boot Messaging

Consumer: DirectExchange

```
@Service
public class DirectRabbitMQConsumer {

    private static final Logger log =
        LoggerFactory.getLogger(DirectRabbitMQConsumer.class);

    @RabbitListener(queues = "htl.sytd", concurrency =
        "2")
    public void listen(String message){
        log.info("message from direct exchange:
            {}",message);
    }

}
```

Spring Boot Messaging

Producer: DirectExchange with JSON

```
@ToString
@Data
@AllArgsConstructor
public class Employee implements Serializable {

    private String firstName;

    private String lastName;

}
```

Spring Boot Messaging

Producer: DirectExchange with JSON

```
@Component
public class DirectRabbitMQProducer {

    @Autowired
    private ObjectMapper objectMapper;

    public void sendDirectMessage(Employee employee)
        throws JsonProcessingException {
        String message =
            objectMapper.writeValueAsString(employee);

        rabbitTemplate.convertAndSend("exchange","routingKey",
            message);
    }

}
```

Spring Boot Messaging

Consumer: DirectExchange with JSON

```
@Service
public class DirectRabbitMQConsumer {

    @Autowired
    private ObjectMapper objectMapper;

    @RabbitListener(queues = "htl.sytd")
    public void listen(String message) {
        Employee employee =
            objectMapper.readValue(message,
            Employee.class);
        ...
    }

}
```