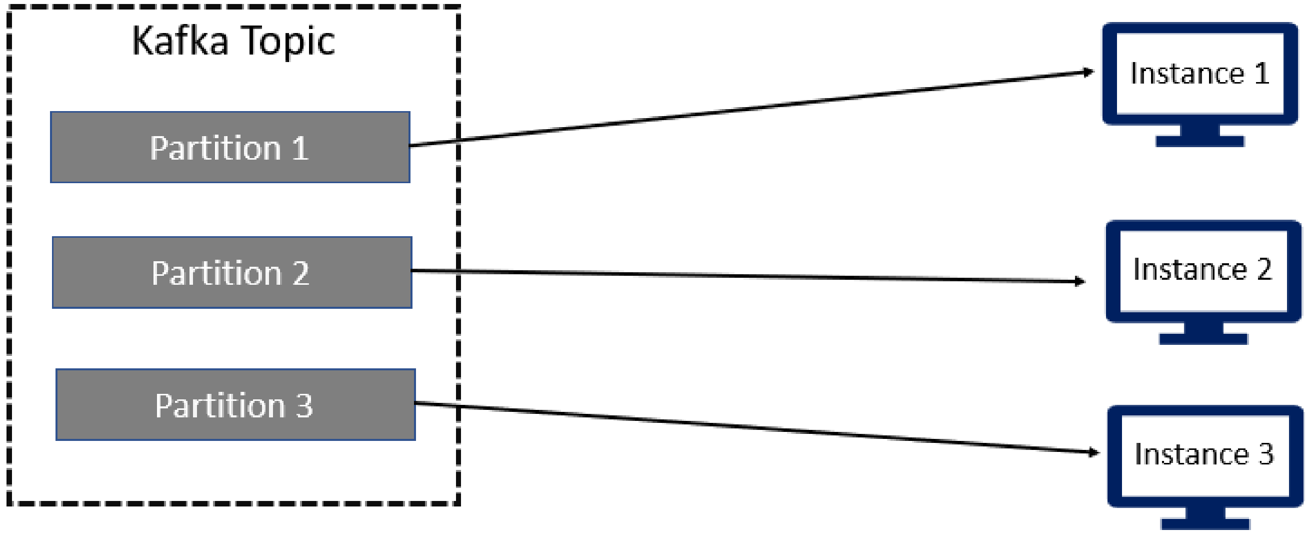
Kafka Streams Architecture

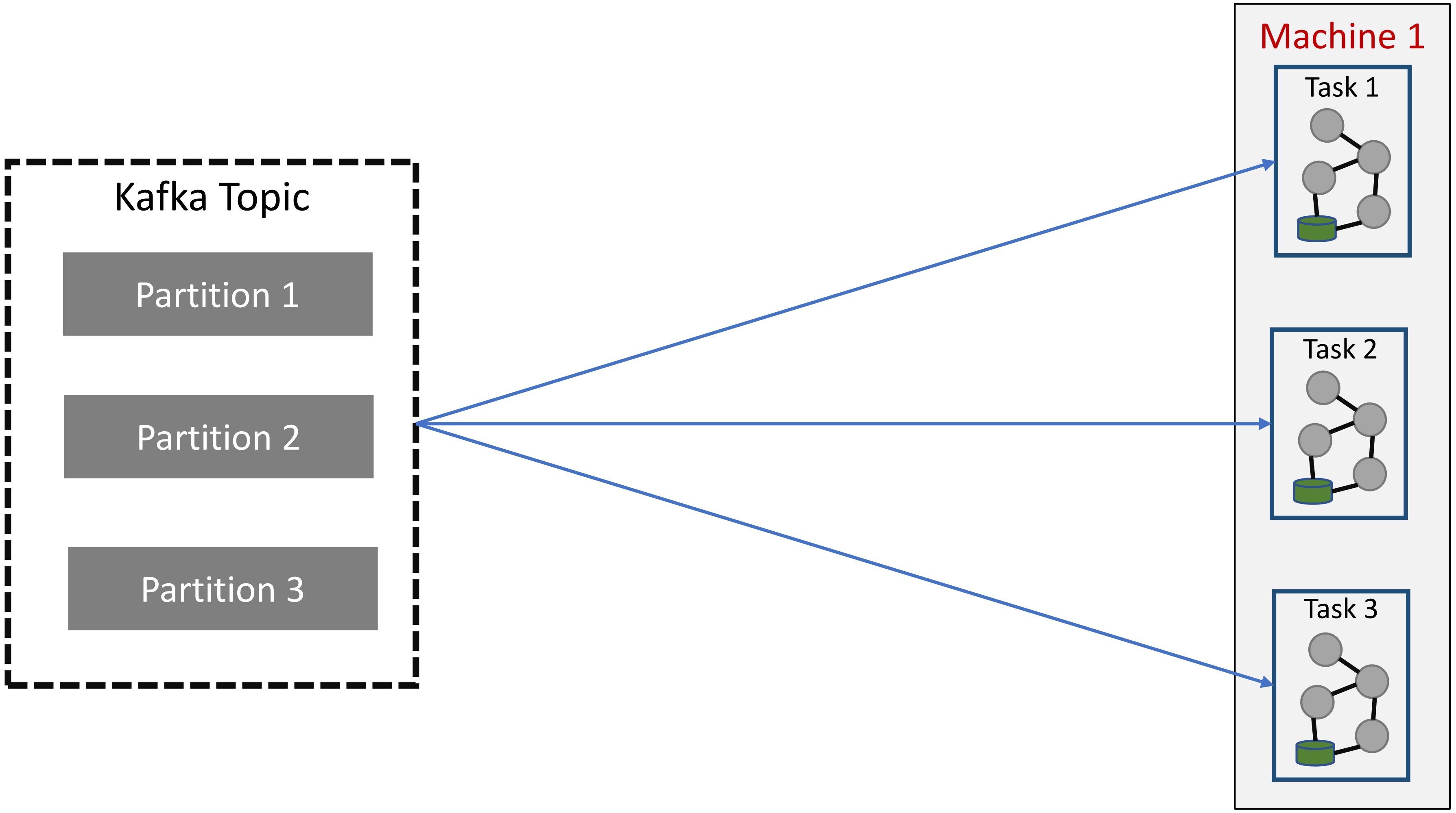
Kafka – Scalability Model

1. Multithreading – Vertical Scaling
2. Multiple Instances – Horizontal Scaling
3. Streams Topology
4. Streams Task



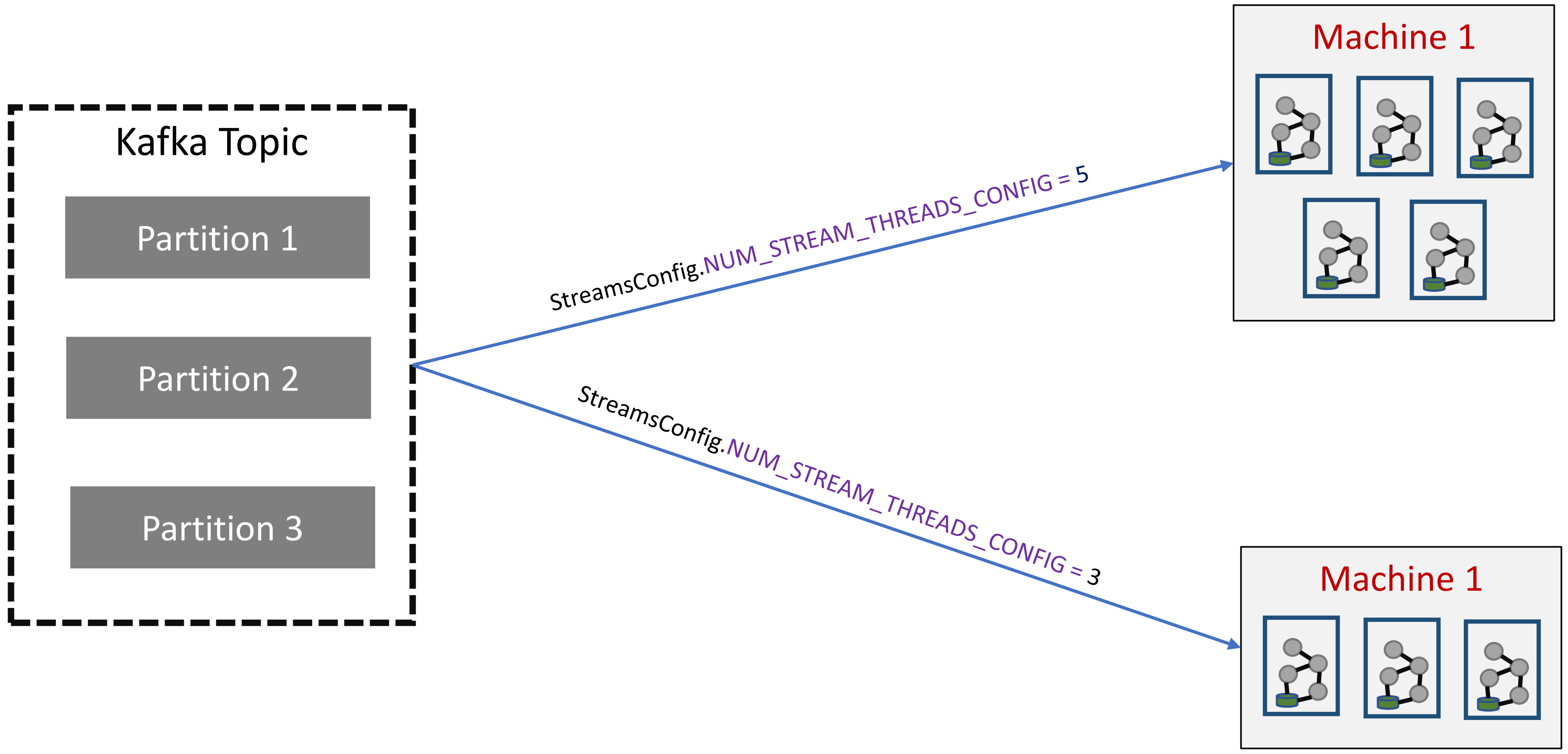
LEARNING JOURNAL

Kafka Streams – Vertical Scaling



```
props.put( StreamsConfig.NUM_STREAM_THREADS_CONFIG, 3 );
```

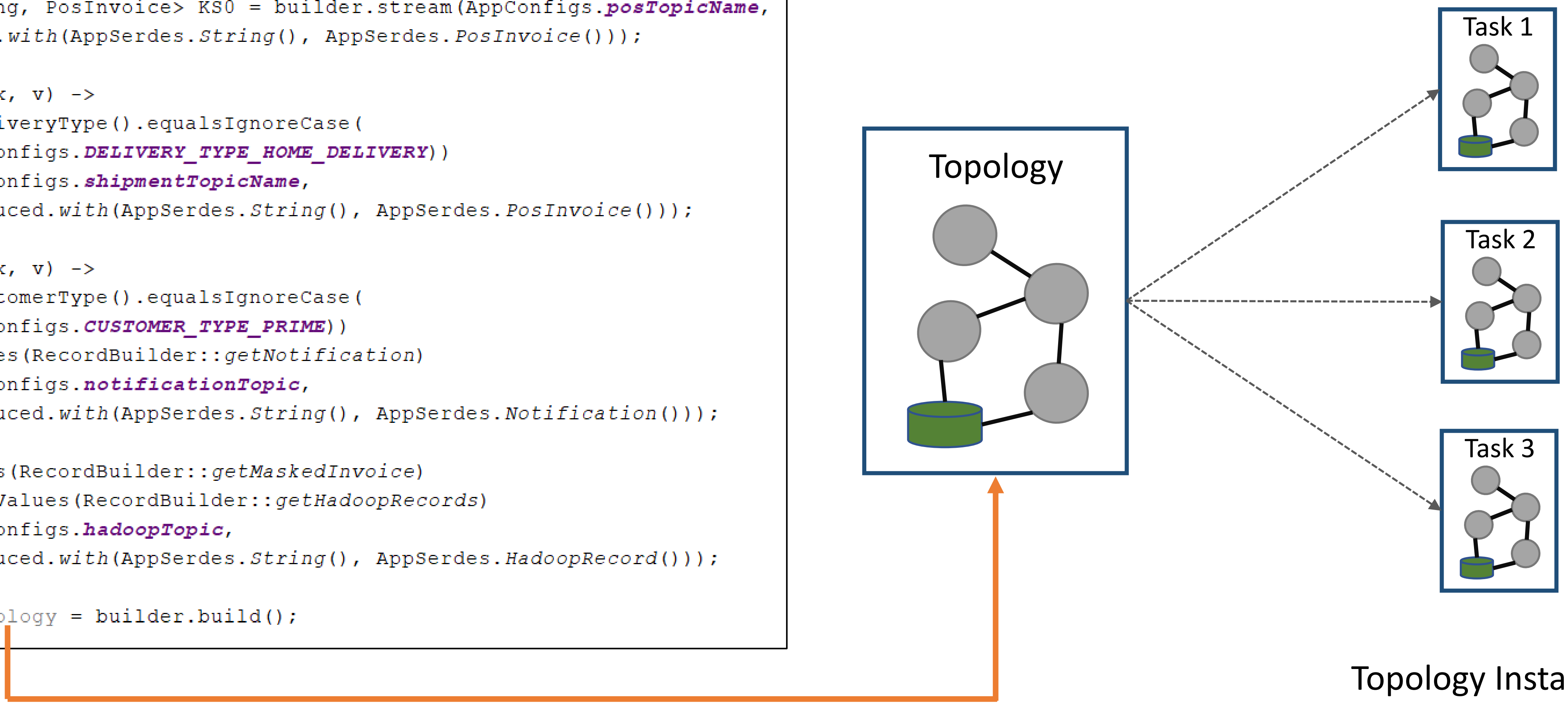
Kafka Streams – Horizontal Scaling


Kafka Topic
Partition 1
Partition 2
Partition 3
StreamsConfig.NUM_STREAM_THREADS_CONFIG = 5
StreamsConfig.NUM_STREAM_THREADS_CONFIG = 3
Machine 1
Machine 1

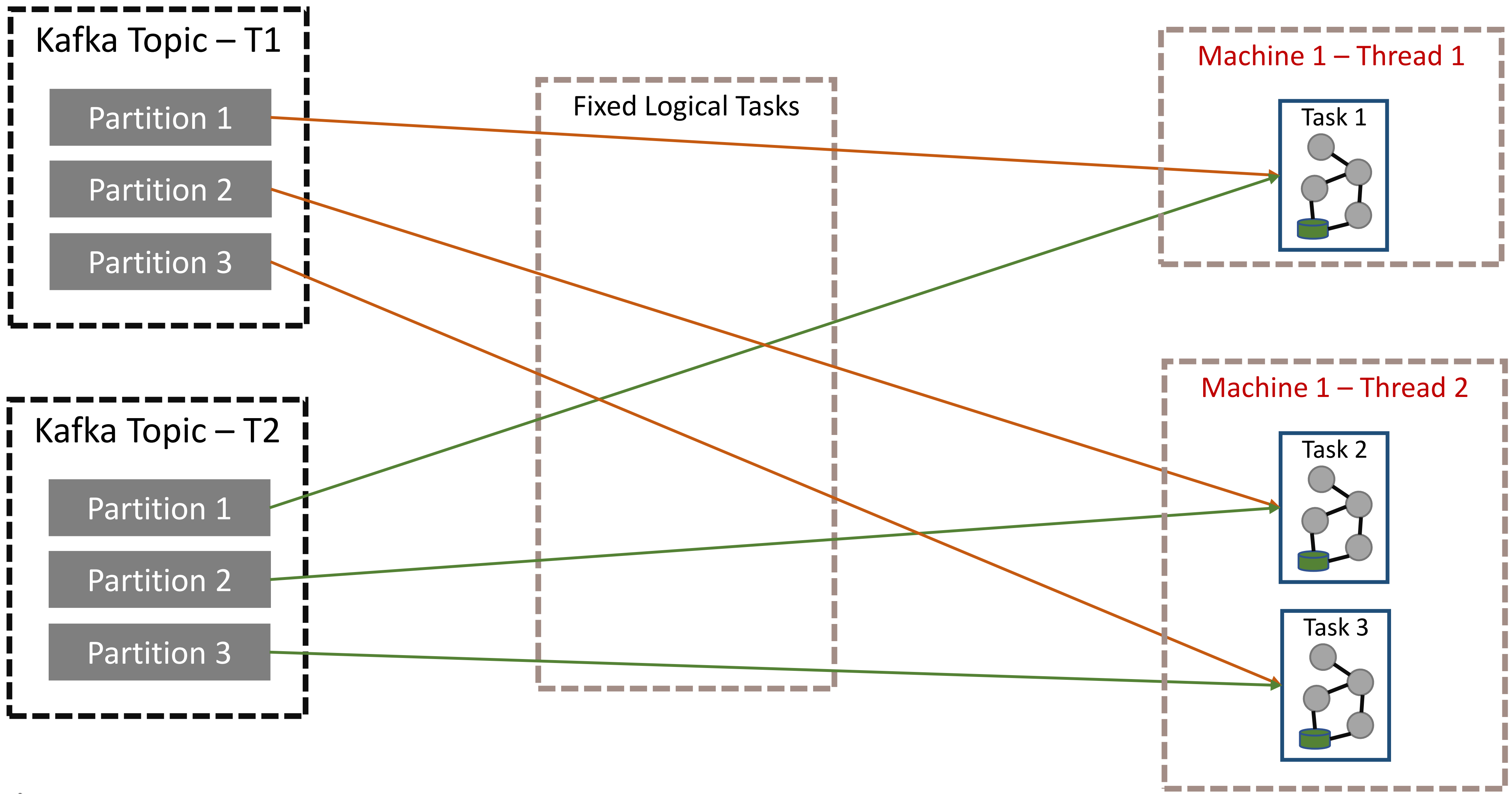
LEARNING JOURNAL

Kafka Streams – Architecture

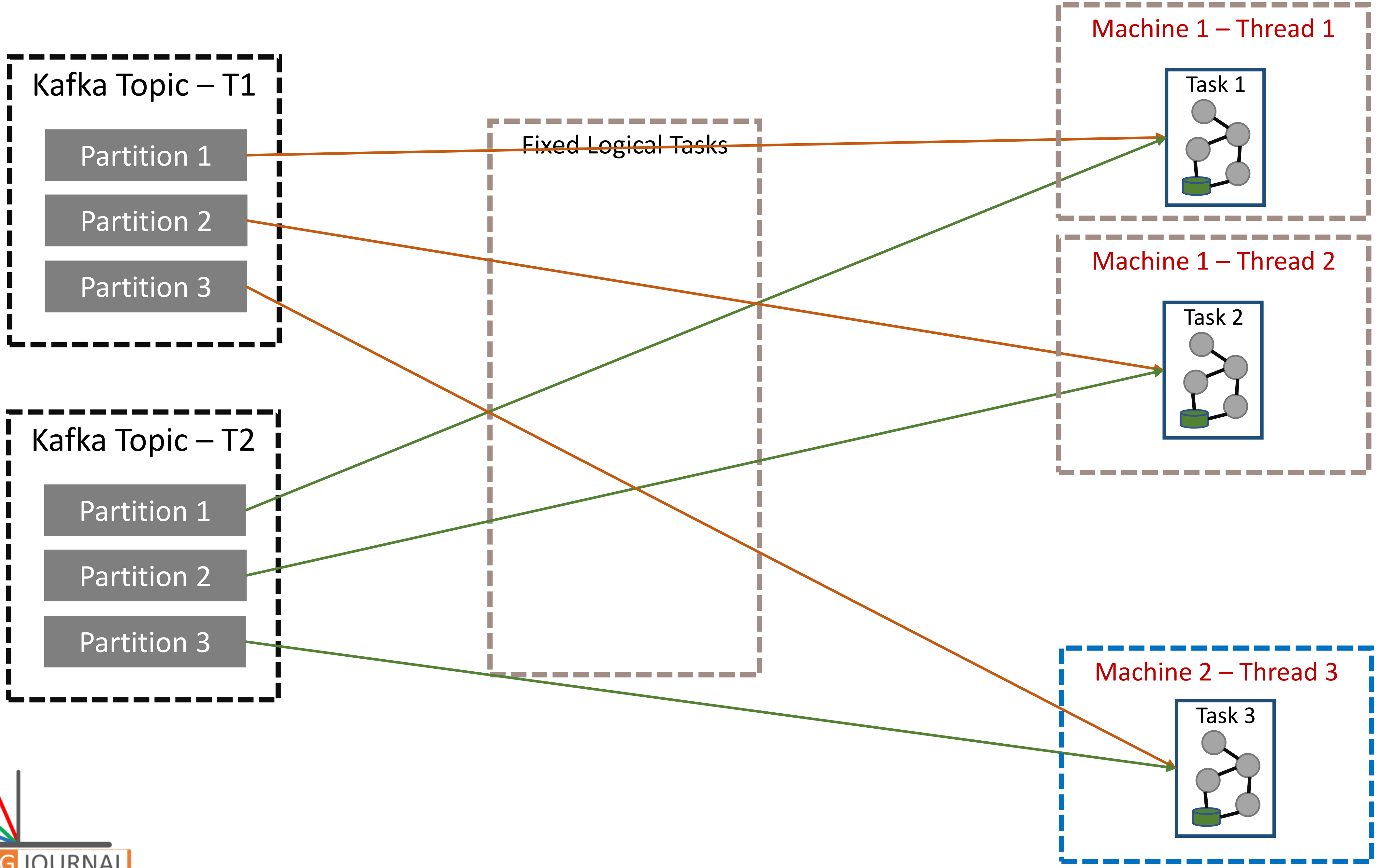
```java
StreamsBuilder builder = new StreamsBuilder();

KStream<String, PosInvoice> KS0 = builder.stream(AppConfigs.posTopicName,
    Consumed.with(AppSerdes.String(), AppSerdes.PosInvoice()));

KS0.filter((k, v) ->
    v.getDeliveryType().equalsIgnoreCase(
        AppConfigs.DELIVERY_TYPE_HOME_DELIVERY))
    .to(AppConfigs.shipmentTopicName,
        Produced.with(AppSerdes.String(), AppSerdes.PosInvoice()));

KS0.filter((k, v) ->
    v.getCustomerType().equalsIgnoreCase(
        AppConfigs.CUSTOMER_TYPE_PRIME))
    .mapValues(RecordBuilder::getNotification)
    .to(AppConfigs.notificationTopic,
        Produced.with(AppSerdes.String(), AppSerdes.Notification()));

KS0.mapValues(RecordBuilder::getMaskedInvoice)
    .flatMapValues(RecordBuilder::getHadoopRecords)
    .to(AppConfigs.hadoopTopic,
        Produced.with(AppSerdes.String(), AppSerdes.HadoopRecord()));

Topology topology = builder.build();
```



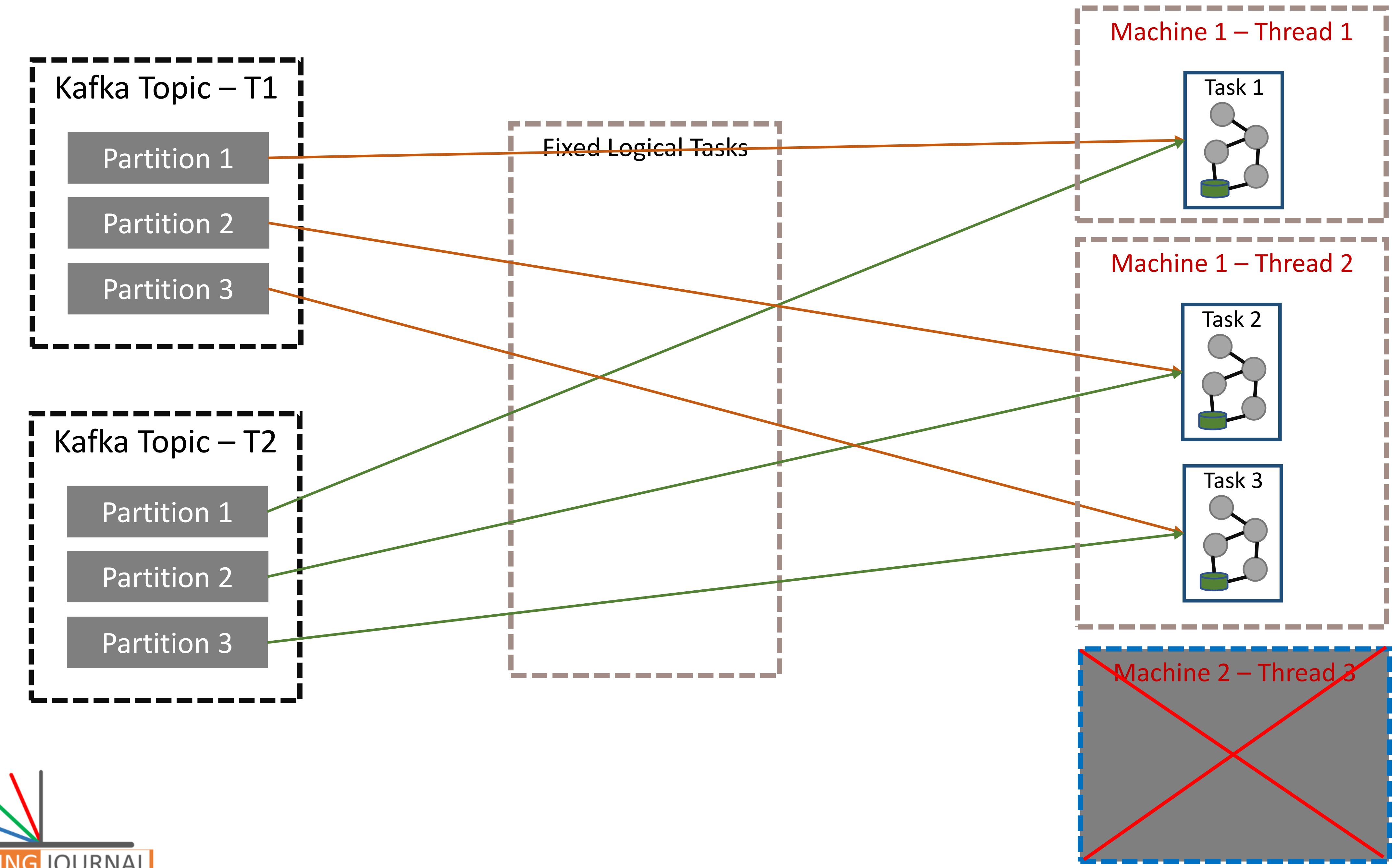
Topology Instances



LEARNING JOURNAL



LEARNING JOURNAL



LEARNING JOURNAL