Red Hat A-MQ – 7.x

A lightweight, high-performance, robust messaging platform

– **Avadhut**

Course Structure

Part – 2 : Core Concepts & Part – 3 : Practical

- Core Concepts (To be covered in subsequent slides)
- A-MQ Producer
- A-MQ Consumer
- A-MQ Topologies
- Master-Slave configuration
- A-MQ DLQ & Message re-delivery
- A-MQ Persistence

Red Hat A-MQ Theory

Core Concepts

A-MQ Producer

- A JMS client that sends messages to queue on Artemis broker

A-MQ Publisher

- A JMS client that publishes messages to topic on Artemis broker

A-MQ Consumer

- A JMS client that receives messages from queue on Artemis broker

A-MQ Subscriber

- A JMS client that subscribes to topic on Artemis broker and receives messages

Red Hat A-MQ

Basic Topologies and Master Slave Config

A-MQ Topologies

HA & FailOver Mechanism

- We define high availability as the ability for the system to continue functioning after failure of one or more of the servers.

- Part of high availability is failover which we define as the ability for client connections to migrate from one server to another in event of server failure so client applications can continue to operate.

Live-Backup Groups

- **A-MQ-7(Artemis) allows servers to be linked together as live - backup groups where each live server can have 1 or more backup servers.**
- **A backup server is owned by only one live server.**
- **Backup servers are not operational until failover occurs, however 1 chosen backup, which will be in passive mode, announces its status and waits to take over the live servers work.**
- **Before failover, only the live server is serving the Apache ActiveMQ Artemis clients while the backup servers remain passive or awaiting to become a backup server.**

Live-Backup Groups

- When a live server crashes or is brought down in the correct mode, the backup server currently in passive mode will become live and another backup server will become passive.
- If a live server restarts after a failover then it will have priority and be the next server to become live when the current live server goes down, if the current live server is configured to allow automatic failback then it will detect the live server coming back up and automatically stop.
- Live-backup strategy also called as master-slave strategy

Red Hat A-MQ

HA Policies

HA Policies

- A-MQ-7 supports two different strategies for backing up a server shared store and replication. Which is configured via the **ha-policy** configuration element.

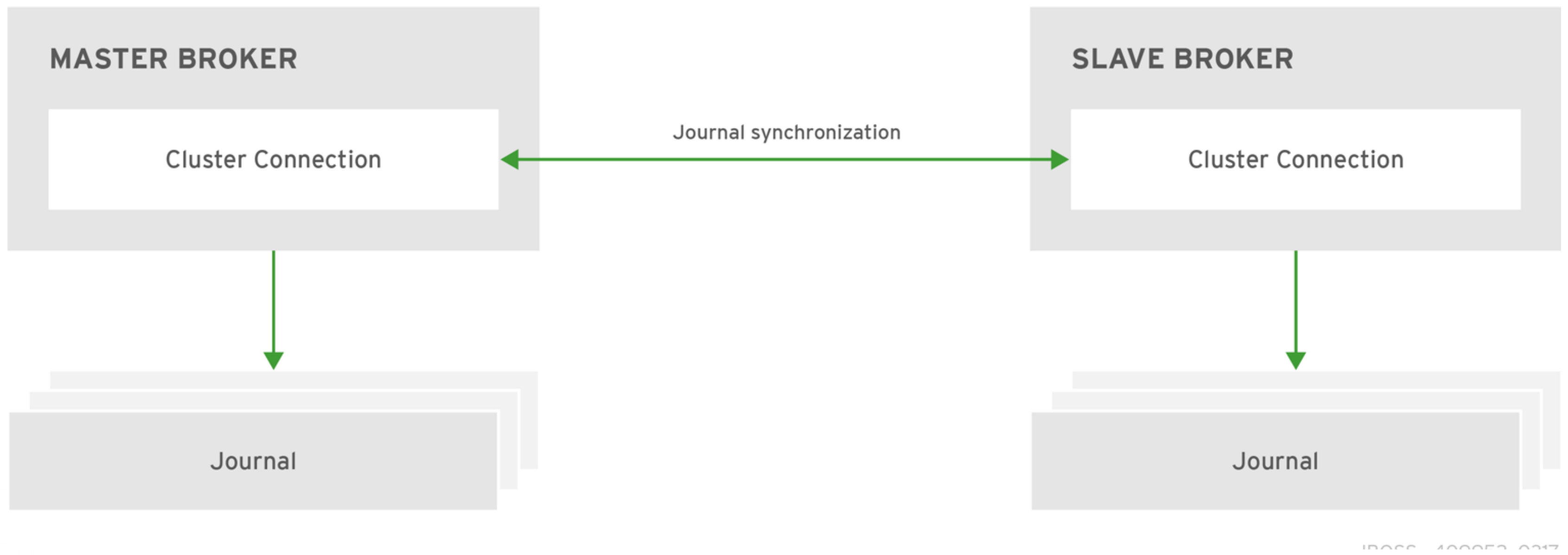
Replication Policy:

```
<ha-policy>
   <replication/>
</ha-policy>
```

Shared Store Policy:

```
<ha-policy>
   <shared-store/>
</ha-policy>
```

Replication



Replication

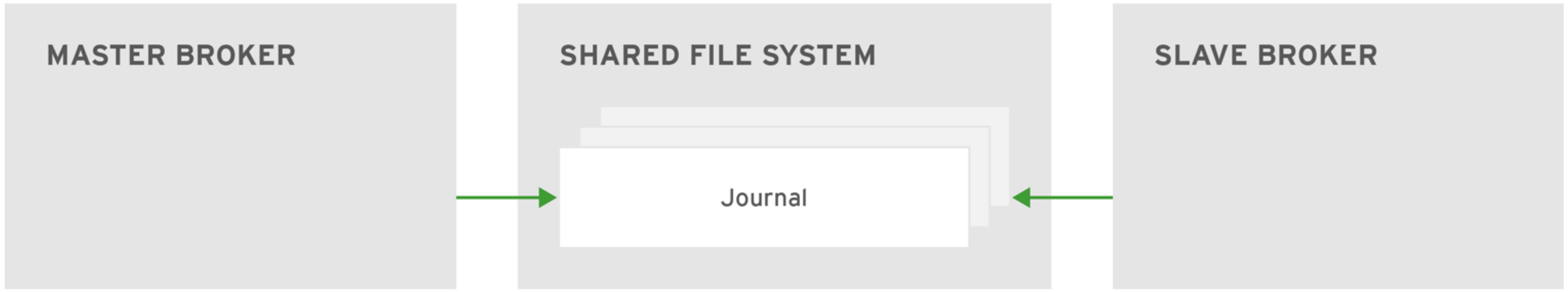
Master:

```
<ha-policy>
  <replication>
    <master>
              <check-for-live-server>true</check-for-live-server>
              <group-name>training-group</group-name>
         </master>
  </replication>
</ha-policy>
<broadcast-groups>
      <broadcast-group name="my-broadcast-group">
        <group-address>${udp-address:231.7.7.7}</group-address>
        <group-port>9876</group-port>
        <broadcast-period>100</broadcast-period>
        <connector-ref>netty-connector</connector-ref>
      </broadcast-group>
    </broadcast-groups>

    <discovery-groups>
      <discovery-group name="my-discovery-group">
        <group-address>${udp-address:231.7.7.7}</group-address>
        <group-port>9876</group-port>
        <refresh-timeout>10000</refresh-timeout>
      </discovery-group>
    </discovery-groups>
    <cluster-connections>
      <cluster-connection name="my-cluster">
        <connector-ref>netty-connector</connector-ref>
        <retry-interval>500</retry-interval>
        <use-duplicate-detection>true</use-duplicate-detection>
        <message-load-balancing>STRICT</message-load-balancing>
        <max-hops>1</max-hops>
        <discovery-group-ref discovery-group-name="my-discovery-
group"/>
      </cluster-connection>
    </cluster-connections>
```

Slave:

```
<ha-policy>
  <replication>
    <slave>
              <allow-failback>true</allow-failback>
              <group-name>training-group</group-name>
         </slave>
  </replication>
</ha-policy>
<broadcast-groups>
      <broadcast-group name="my-broadcast-group">
        <group-address>${udp-address:231.7.7.7}</group-address>
        <group-port>9876</group-port>
        <broadcast-period>100</broadcast-period>
        <connector-ref>netty-connector</connector-ref>
      </broadcast-group>
    </broadcast-groups>

    <discovery-groups>
      <discovery-group name="my-discovery-group">
        <group-address>${udp-address:231.7.7.7}</group-address>
        <group-port>9876</group-port>
        <refresh-timeout>10000</refresh-timeout>
      </discovery-group>
    </discovery-groups>
    <cluster-connections>
      <cluster-connection name="my-cluster">
        <connector-ref>netty-connector</connector-ref>
        <retry-interval>500</retry-interval>
        <use-duplicate-detection>true</use-duplicate-detection>
        <message-load-balancing>STRICT</message-load-balancing>
        <max-hops>1</max-hops>
        <discovery-group-ref discovery-group-name="my-discovery-
group"/>
      </cluster-connection>
    </cluster-connections>
```

Shared Store



Shared Store

Master:

```
<ha-policy>
  <shared-store>
       <master>
         <failover-on-shutdown>true</failover-on-shutdown>
       </master>
     </shared-store>
</ha-policy>
<broadcast-groups>
      <broadcast-group name="my-broadcast-group">
        <group-address>${udp-address:231.7.7.7}</group-address>
        <group-port>9876</group-port>
        <broadcast-period>100</broadcast-period>
        <connector-ref>netty-connector</connector-ref>
      </broadcast-group>
    </broadcast-groups>

    <discovery-groups>
      <discovery-group name="my-discovery-group">
        <group-address>${udp-address:231.7.7.7}</group-address>
        <group-port>9876</group-port>
        <refresh-timeout>10000</refresh-timeout>
      </discovery-group>
    </discovery-groups>
    <cluster-connections>
      <cluster-connection name="my-cluster">
        <connector-ref>netty-connector</connector-ref>
        <retry-interval>500</retry-interval>
        <use-duplicate-detection>true</use-duplicate-detection>
        <message-load-balancing>STRICT</message-load-balancing>
        <max-hops>1</max-hops>
        <discovery-group-ref discovery-group-name="my-discovery-
group"/>
      </cluster-connection>
    </cluster-connections>
```

Slave:

```
<ha-policy>
  <shared-store>
       <slave>
         <failover-on-shutdown>true</failover-on-shutdown>
             <allow-failback>true</allow-failback>
       </slave>
     </shared-store>
</ha-policy>
<broadcast-groups>
      <broadcast-group name="my-broadcast-group">
        <group-address>${udp-address:231.7.7.7}</group-address>
        <group-port>9876</group-port>
        <broadcast-period>100</broadcast-period>
        <connector-ref>netty-connector</connector-ref>
      </broadcast-group>
    </broadcast-groups>

    <discovery-groups>
      <discovery-group name="my-discovery-group">
        <group-address>${udp-address:231.7.7.7}</group-address>
        <group-port>9876</group-port>
        <refresh-timeout>10000</refresh-timeout>
      </discovery-group>
    </discovery-groups>
    <cluster-connections>
      <cluster-connection name="my-cluster">
        <connector-ref>netty-connector</connector-ref>
        <retry-interval>500</retry-interval>
        <use-duplicate-detection>true</use-duplicate-detection>
        <message-load-balancing>STRICT</message-load-balancing>
        <max-hops>1</max-hops>
        <discovery-group-ref discovery-group-name="my-discovery-
group"/>
      </cluster-connection>     </cluster-connections>
```

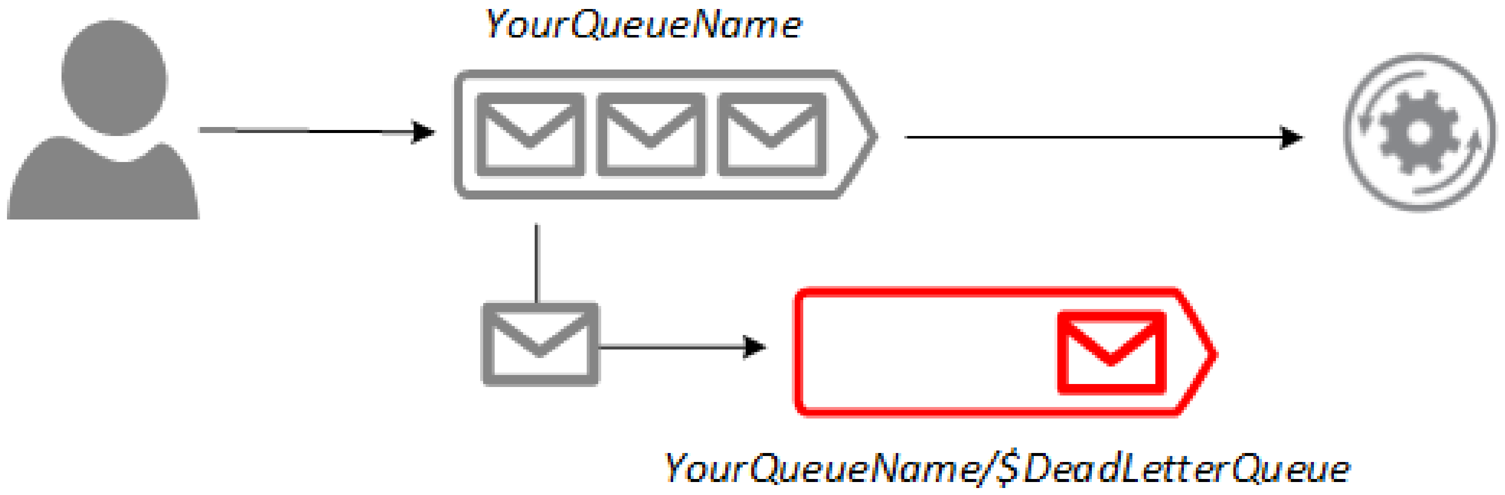
Red Hat A-MQ

HA Practicals

Red Hat A-MQ

DLQ and Message Re-delivery

Dead Letter Queue



Dead Letter Queue

In messaging, the **dead letter** queue is a service implementation to store messages that meet one or more of the following criteria:

- Message that is sent to a queue that does not exist
- Message length limit exceeded
- Message is rejected by another queue exchange
- Queue length limit exceeded
- Message reaches a threshold read counter number, because it is not consumed. Sometimes this is called a "back out queue"

Dead letter queue storing of these messages allows developers to look for common patterns and potential software problems.

Messaging systems that incorporate DLQs include Amazon Simple Queue Service, Apache ActiveMQ, HornetQ, Microsoft MQ,WebSphere MQ and Rabbit MQ.

Message Re-delivery

Messages can be delivered unsuccessfully (e.g. if the transacted session used to consume them is rolled back).

Such a message goes back to its queue ready to be redelivered. However, this means it is possible for a message to be delivered again and again without success thus remaining in the queue indefinitely, clogging the system.

**** 2 ways to deal with these undelivered messages:**

Delayed redelivery: It is possible to delay messages redelivery. This gives the client some time to recover from any transient failures and to prevent overloading its network or CPU resources.

Dead Letter Address: It is also possible to configure a dead letter address so that after a specified number of unsuccessful deliveries, messages are removed from their queue and sent to the dead letter address. These messages will not be delivered again from this queue.

Both options can be combined for maximum flexibility.

Delayed Re-delivery

Delaying re-delivery is very useful in cases where clients regularly fail or rollback.

Without a delayed re-delivery, the system can get into a "thrashing" state, with delivery being attempted, the client rolling back, and delivery being re-attempted and infinitum in quick succession, consuming valuable CPU and network resources.

By default, there is no redelivery delay (redelivery-delayis set to 0).

```
<!-- delay redelivery of messages for 5s -->
<address-setting match="jms.queue.exampleQueue">
<!-- default is 1.0 -->
<redelivery-delay-multiplier>1.5</redelivery-delay-multiplier>
<!-- default is 0 (no delay) -->
<redelivery-delay>5000</redelivery-delay>
<!-- default is redelivery-delay * 10 -->
<max-redelivery-delay>50000</max-redelivery-delay>

</address-setting>
```

Dead Letter Address

If a dead-letter-address is not specified, messages will removed after max-delivery-attempts unsuccessful attempts.

By default, messages are redelivered 10 times at the maximum. Set max-delivery-attempts to -1 for infinite redeliveries.

```
<!-- undelivered messages in exampleQueue will be sent to the dead letter address
deadLetterQueue after 3 unsuccessful delivery attempts →

<address-setting match="jms.queue.exampleQueue">

<dead-letter-address>jms.queue.deadLetterQueue</dead-letter-address>

<max-delivery-attempts>3</max-delivery-attempts>

</address-setting>
```

Red Hat A-MQ

DLQ & Message Re-delivery Practicals

Red Hat A-MQ

Persistence

A-MQ Persistence

- **Apache ActiveMQ Artemis ships with a high performance journal.**
- **Since Apache ActiveMQ Artemis handles its own persistence, rather than relying on a database or other 3rd party persistence engine it is very highly optimised for the specific messaging use cases.**

**** Artemis ships 2 journal implementations:**

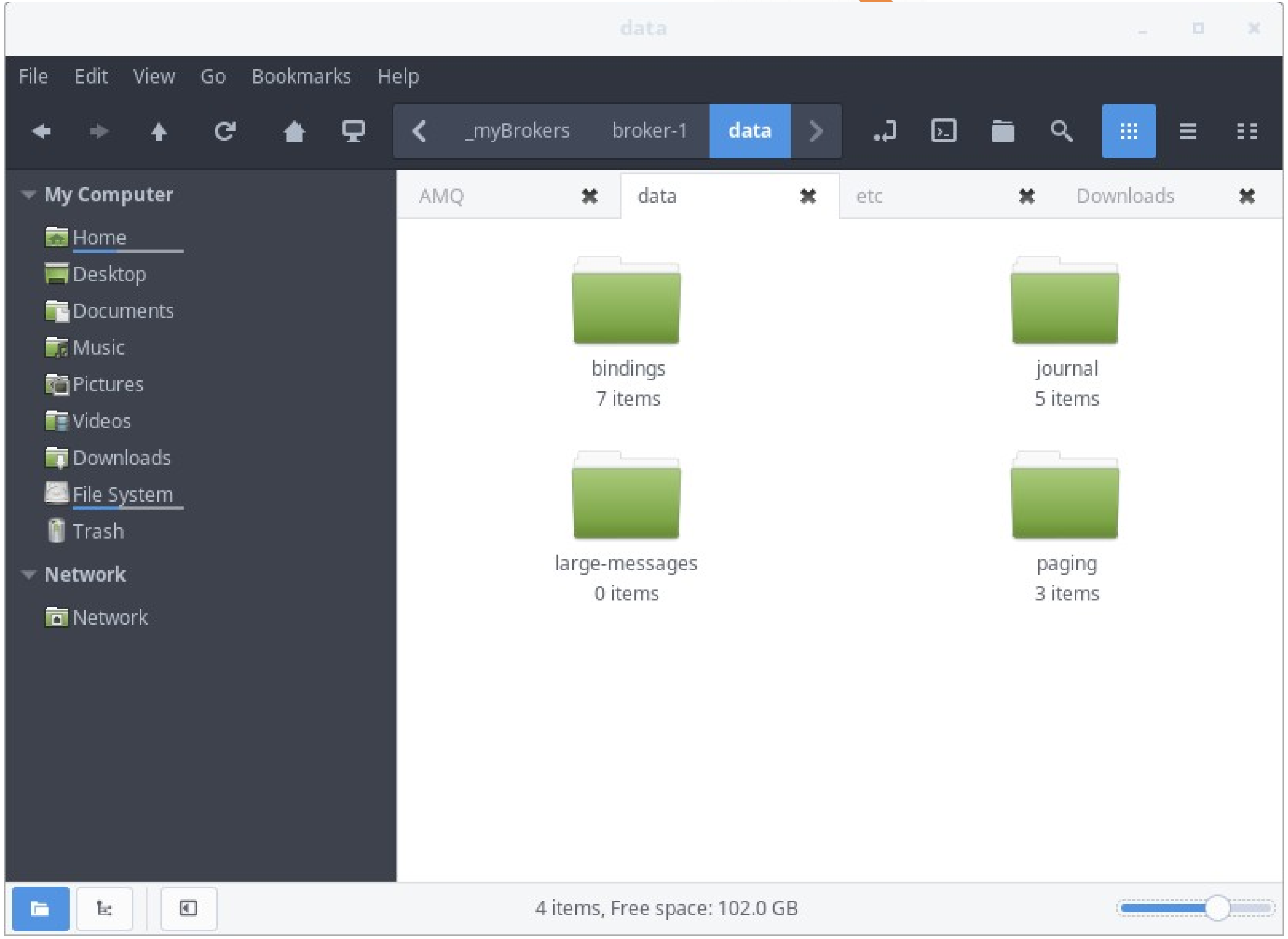
- NIO
- AIO

A-MQ Persistence

The standard Apache ActiveMQ Artemis core server uses two instances of the journal:

- Bindings journal : Stores bindings related data. That includes the set of queues that are deployed on the server and their attributes. It also stores data such as id sequence counters
- JMS journal : Stores all JMS related data, This is basically any JMS Queues, Topics and Connection Factories and any JNDI bindings for these resources
- Message journal : Stores all message related data, including the message themselves and also duplicate-id caches

A-MQ Persistence



bindings 7 items
journal 5 items
large-messages 0 items
paging 3 items

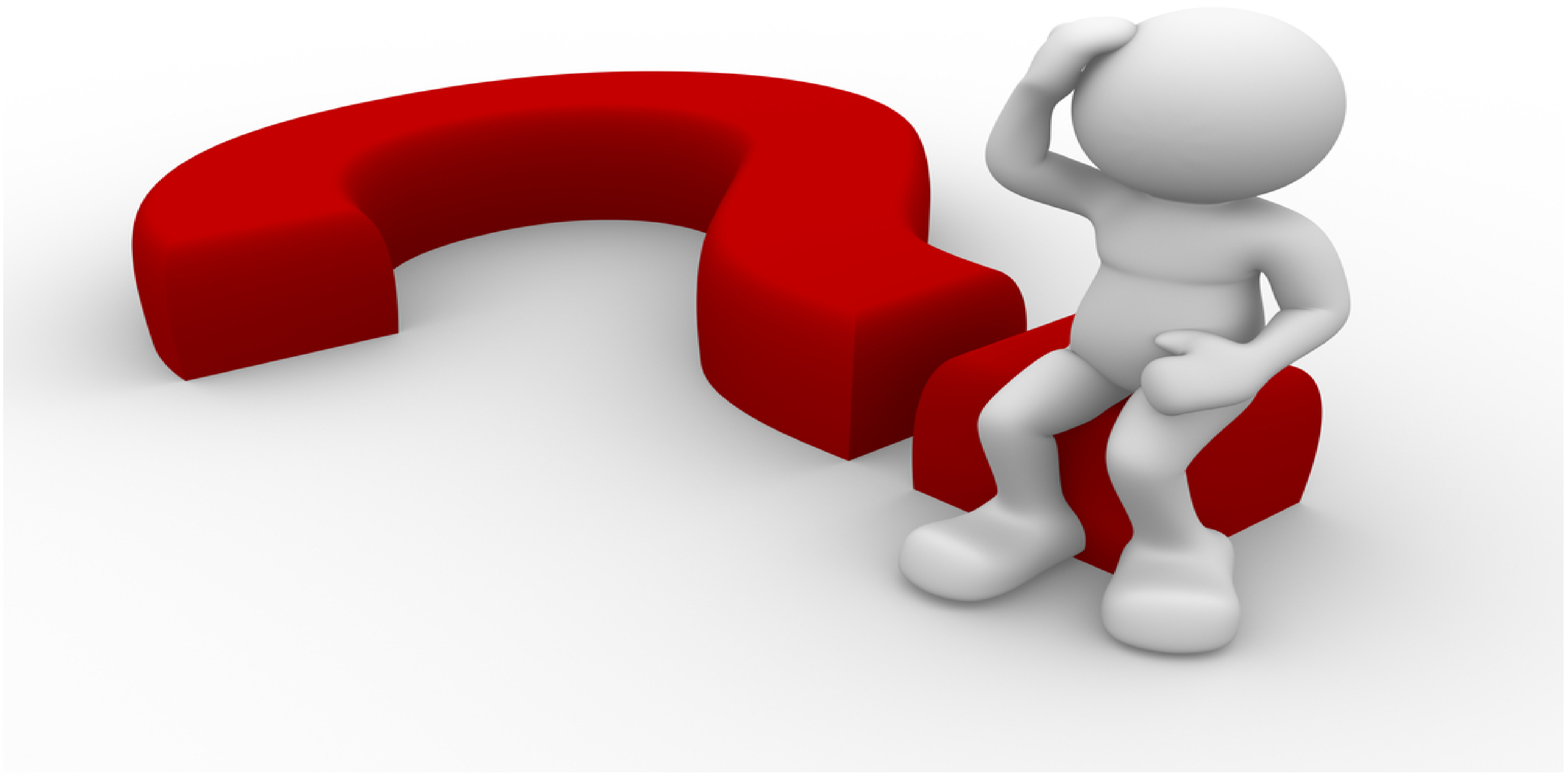
4 items, Free space: 102.0 GB

A-MQ Part-2 revision

Part – 2 : Core Concepts & Part – 3 : Practical

- Core Concepts (To be covered in subsequent slides)
- A-MQ Producer
- A-MQ Consumer
- A-MQ Topologies
- Master-Slave configuration
- A-MQ DLQ & Message re-delievery
- A-MQ Persistence

Questions?



kodtodya-talks

Thank you..!!!

LinkedIn, GitHub, GitLab, Twitter: @kodtodya