Message Queues in DIRAC

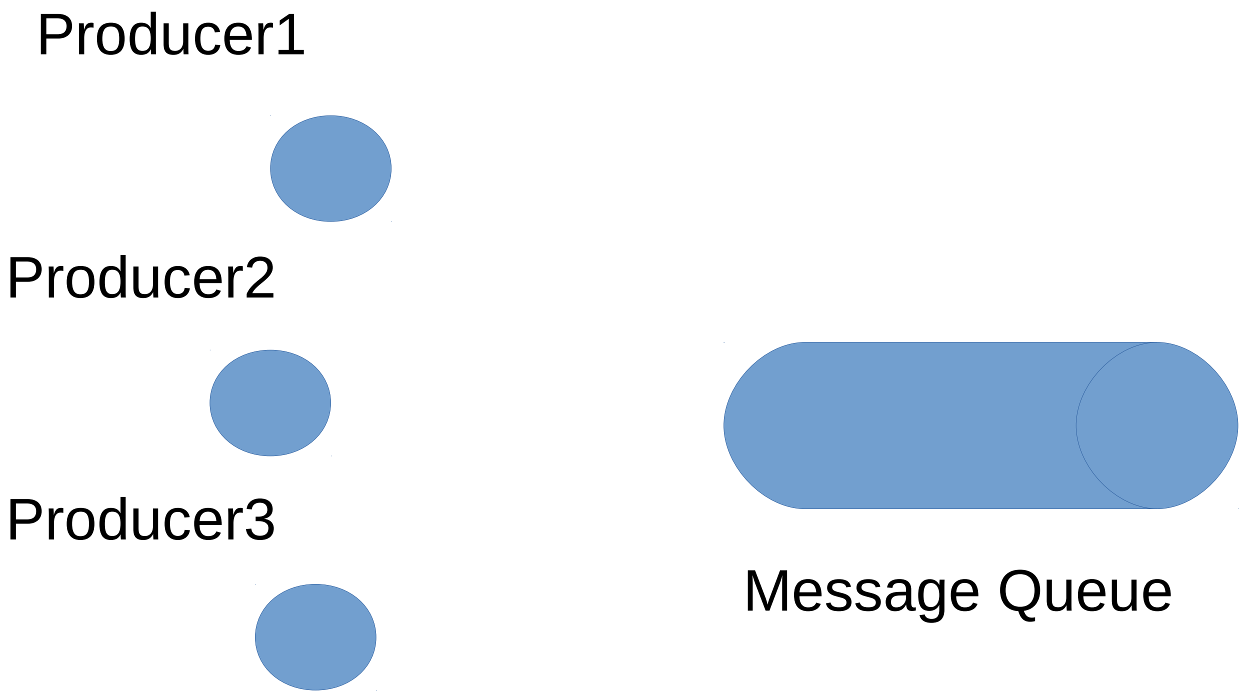
Wojciech Krzemień

The 7th DIRAC Users Workshop

29th of May 2017, Warszawa

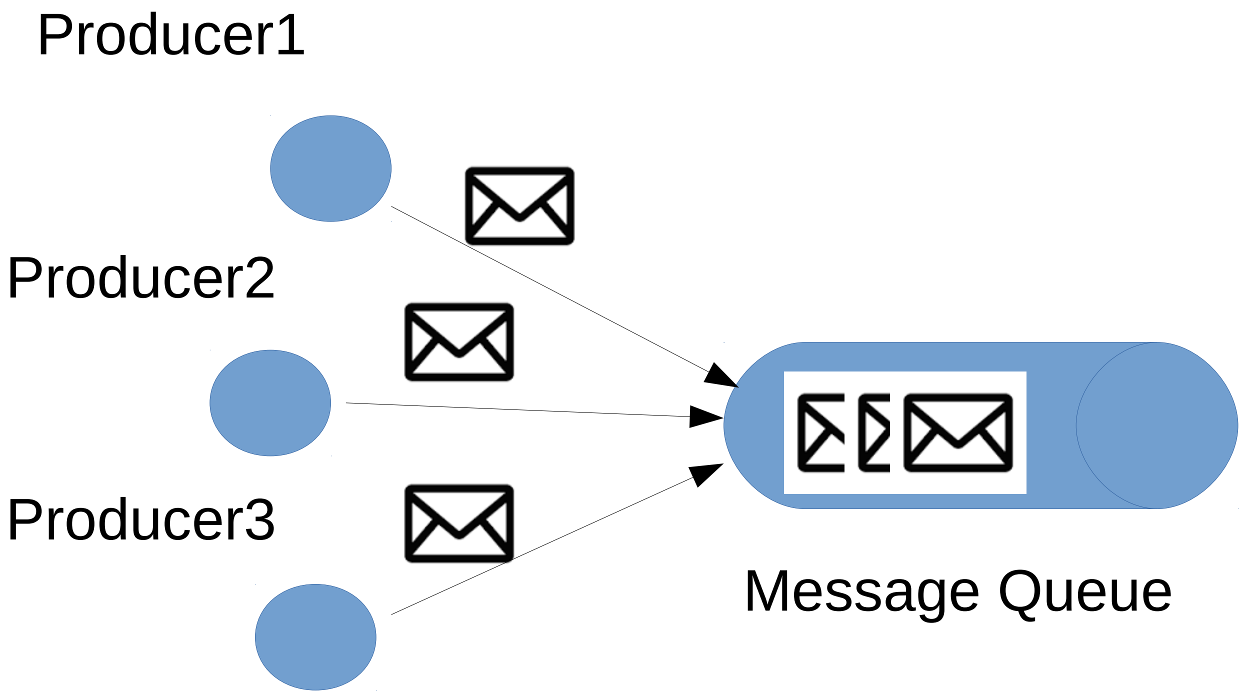
What is Message Queueing?

- **Asynchronous** communication scheme
- Components are **decoupled** by the **queue** in which **messages** are stored



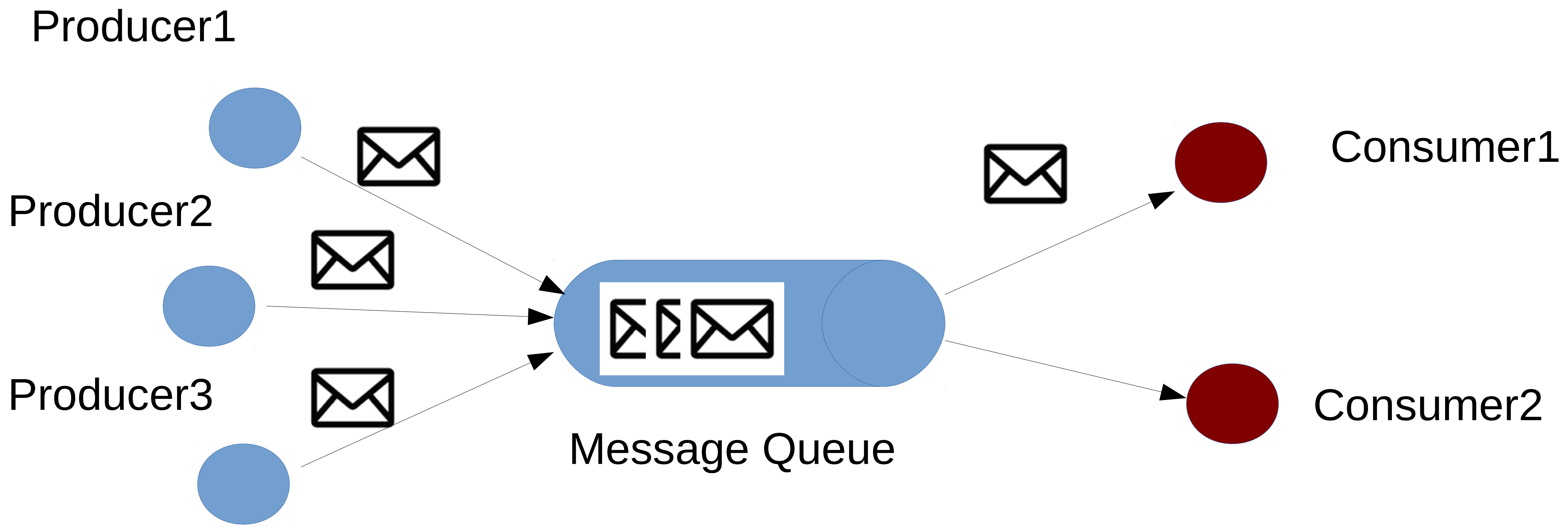
What is Message Queueing?

- **Asynchronous** communication scheme
- Components are **decoupled** by the **queue** in which **messages** are stored



What is Message Queueing?

- **Asynchronous** communication scheme
- Components are **decoupled** by the **queue** in which **messages** are stored



What is Message Queueing?

Advantages:

- **Scalability**
- Performance
- Resilience
- **Connect heterogeneous environments**
- Redundancy
- Delivery Guarantee
- ….

What is Message Queueing?

Advantages:

- **Scalability**
- Performance
- Resilience
- **Connect heterogeneous environments**
- Redundancy
- Delivery Guarantee
- ….

MQ communication protocols:

- Advanced Message Queueing Protocol (AMPQ)
- Streaming Text-Oriented Messaging Protocol (STOMP)
- others

What is Message Queueing?

Advantages:

- **Scalability**
- Performance
- Resilience
- **Connect heterogeneous environments**
- Redundancy
- Delivery Guarantee
- ….

MQ communication protocols:

- Advanced Message Queueing Protocol (AMPQ)
- Streaming Text-Oriented Messaging Protocol (STOMP)
- others

Some open source MQ projects:

ActiveMQ

JBoss™ **Messaging**

Kafka

RabbitMQ

JORAM powered by ScalAgent

Qpid

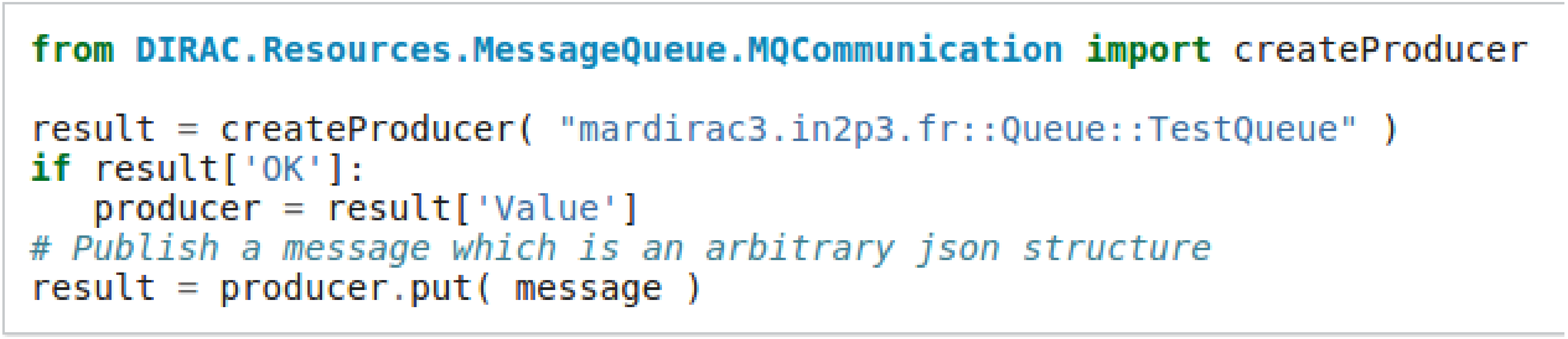
MQ in Dirac

- MQ can be used for sending messages between DIRAC components or to communicate with third-part services
- Generic MQ interface since **DIRAC v6r17**
- **STOMP** protocol handler implementation with **SSL** and **topics** support
- All MQ configuration are loaded from the Configuration Service

Contributors: S. Balbuena, H. Giemza, W. Krzemien, Z. Mathe, F. Stagni

Code snippet

Create Producer and send a message to the queue



```python
from DIRAC.Resources.MessageQueue.MQCommunication import createProducer

result = createProducer( "mardirac3.in2p3.fr::Queue::TestQueue" )
if result['OK']:
   producer = result['Value']
# Publish a message which is an arbitrary json structure
result = producer.put( message )
```

Code snippet

Create Producer and send a message to the queue

```python
from DIRAC.Resources.MessageQueue.MQCommunication import createProducer

result = createProducer( "mardirac3.in2p3.fr::Queue::TestQueue" )
if result['OK']:
   producer = result['Value']
# Publish a message which is an arbitrary json structure
result = producer.put( message )
```

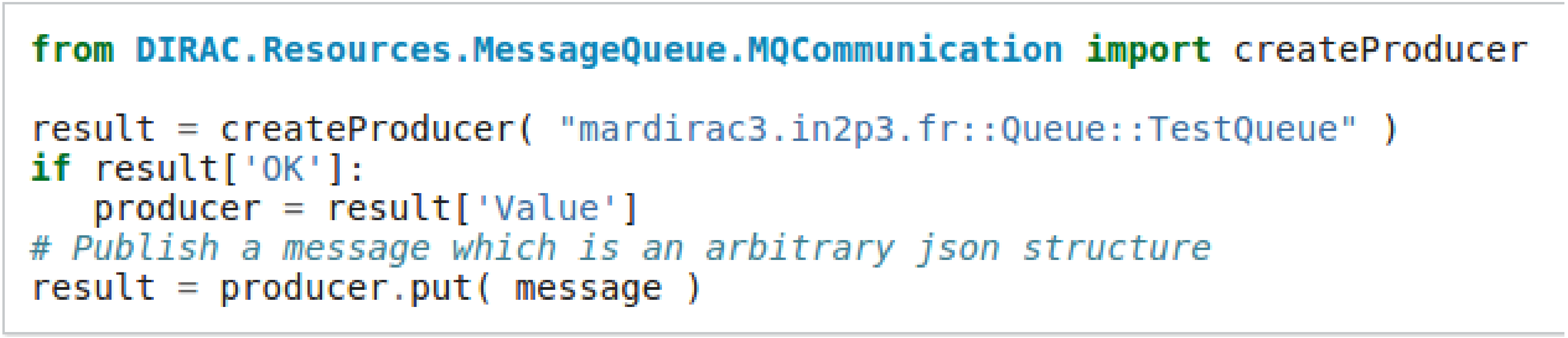
Create Consumer and read a message from the queue

```python
from DIRAC.Resources.MessageQueue.MQCommunication import createConsumer

result = createConsumer( "mardirac3.in2p3.fr::Queue::TestQueue" )
if result['OK']:
   consumer = result['Value']
result = consumer.get( message )
if result['OK']:
  message = result['Value']
```

Code snippet

Create Producer and send a message to the queue

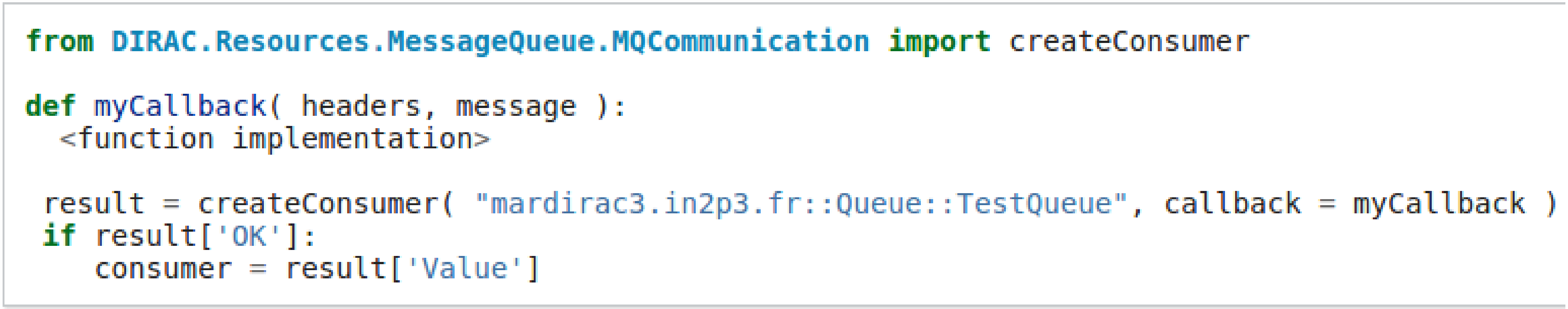


```python
from DIRAC.Resources.MessageQueue.MQCommunication import createProducer

result = createProducer( "mardirac3.in2p3.fr::Queue::TestQueue" )
if result['OK']:
   producer = result['Value']
# Publish a message which is an arbitrary json structure
result = producer.put( message )
```

Create Consumer and use a callback function

to handle messages



```python
from DIRAC.Resources.MessageQueue.MQCommunication import createConsumer

def myCallback( headers, message ):
  <function implementation>

 result = createConsumer( "mardirac3.in2p3.fr::Queue::TestQueue", callback = myCallback )
 if result['OK']:
    consumer = result['Value']
```

A bit of details

MQCommunication

createConsumer()
createProducer()

A bit of details

MQCommunication

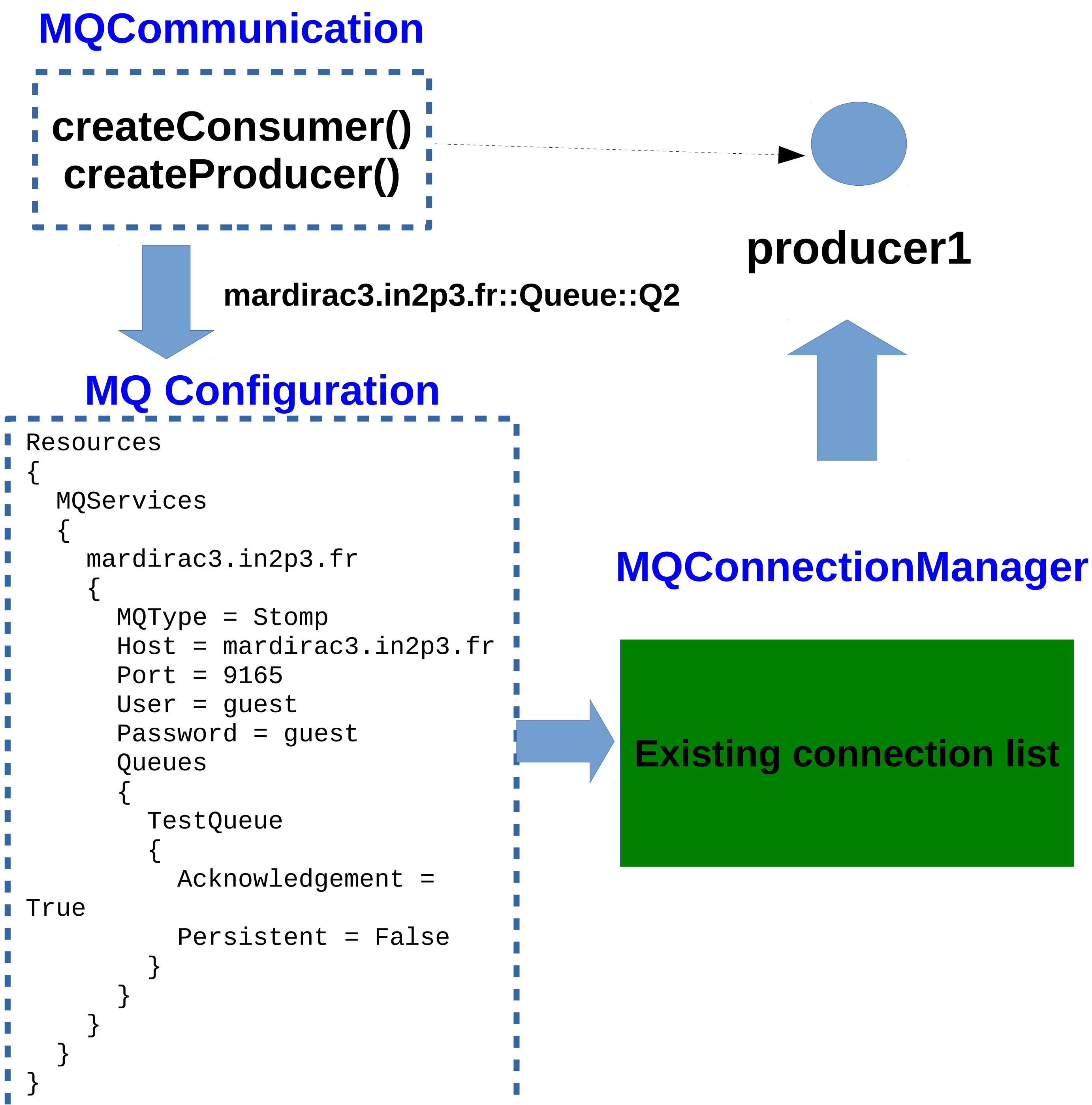
createConsumer()
createProducer()

mardirac3.in2p3.fr::Queue::Q2

MQ Configuration

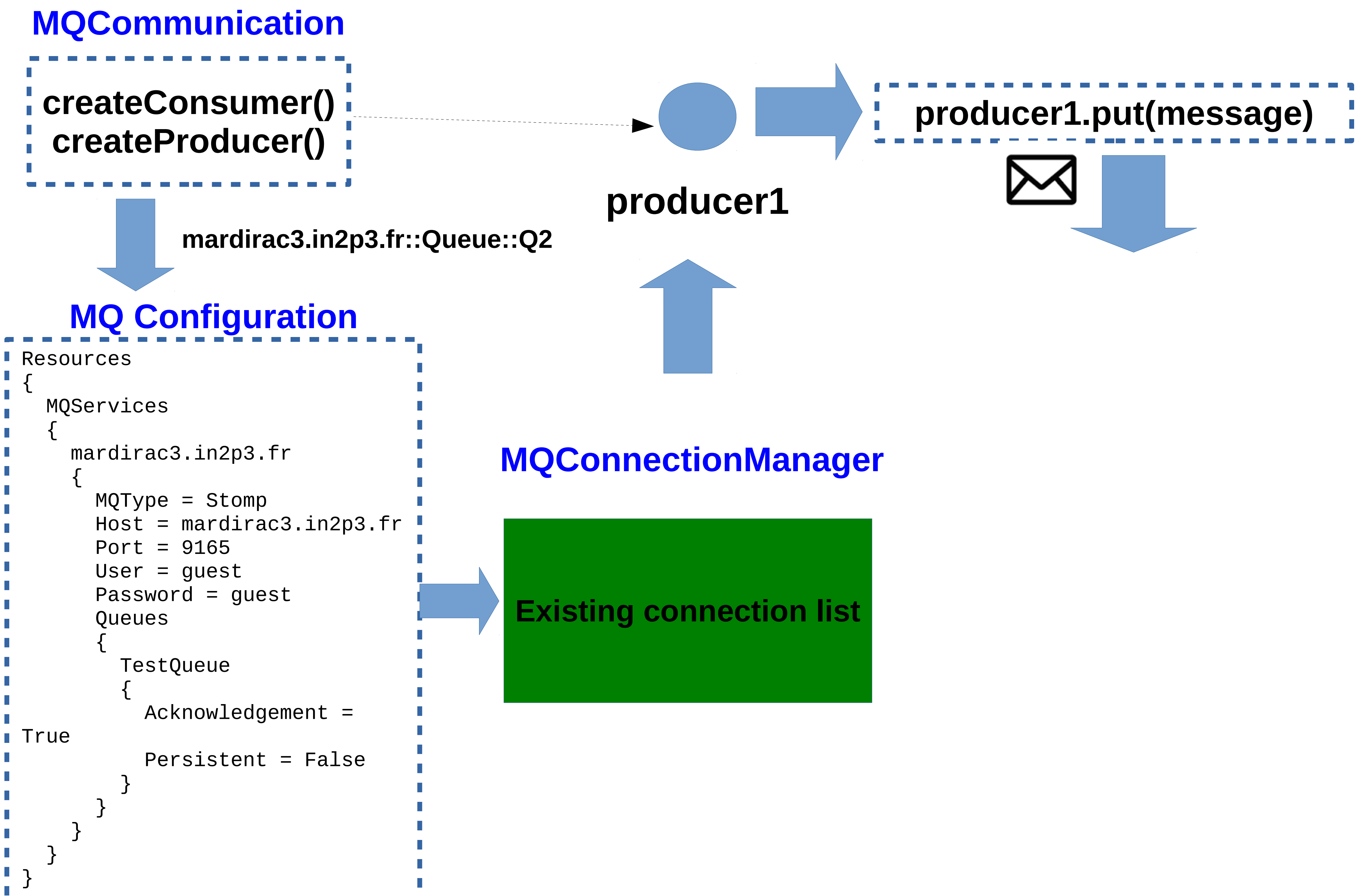
```
Resources
{
  MQServices
  {
    mardirac3.in2p3.fr
    {
      MQType = Stomp
      Host = mardirac3.in2p3.fr
      Port = 9165
      User = guest
      Password = guest
      Queues
      {
        TestQueue
        {
          Acknowledgement =
True
          Persistent = False
        }
      }
    }
  }
}
```

A bit of details



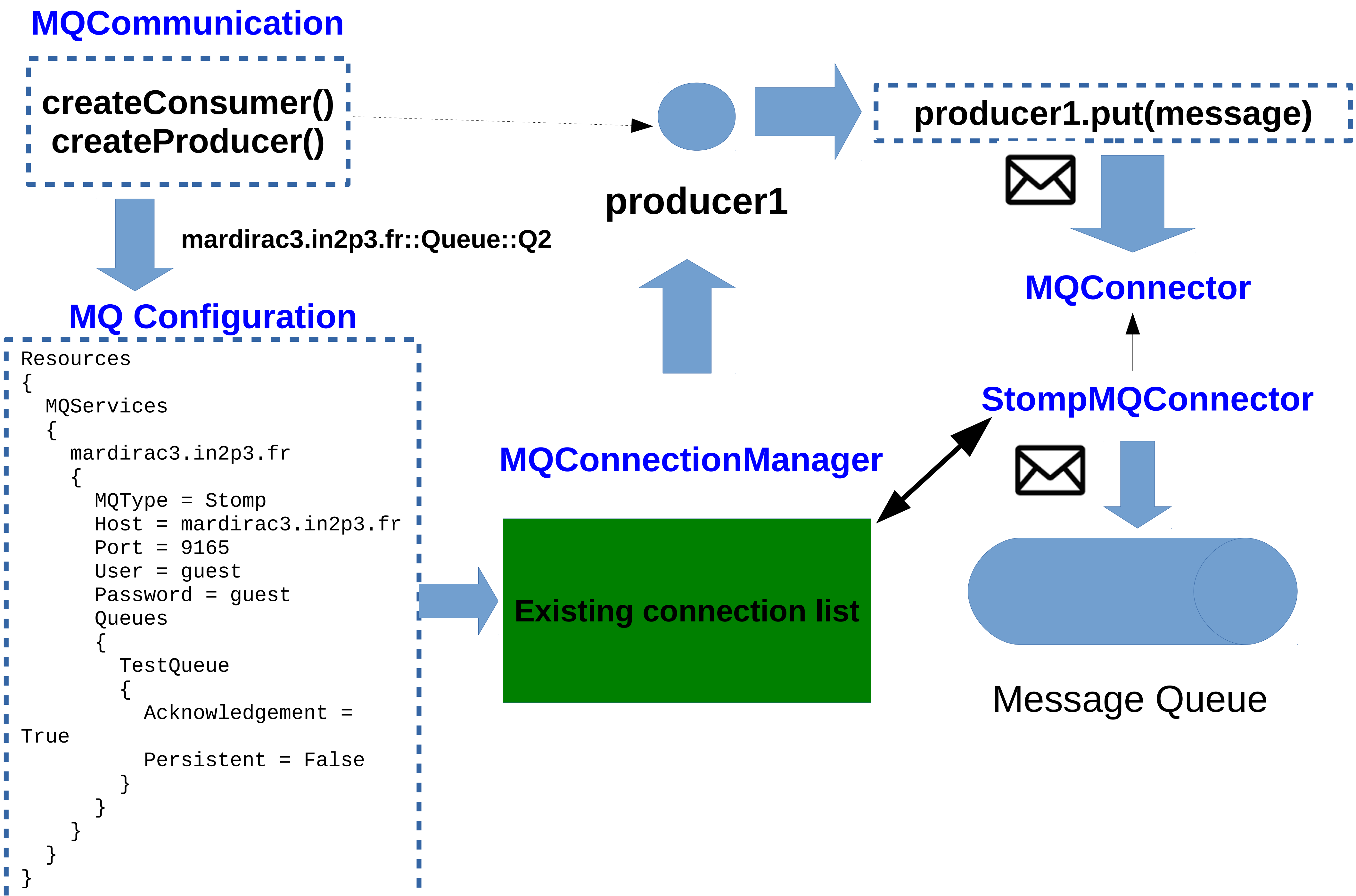
```
Resources
{
  MQServices
  {
    mardirac3.in2p3.fr
    {
      MQType = Stomp
      Host = mardirac3.in2p3.fr
      Port = 9165
      User = guest
      Password = guest
      Queues
      {
        TestQueue
        {
          Acknowledgement =
True
          Persistent = False
        }
      }
    }
  }
}
```

A bit of details



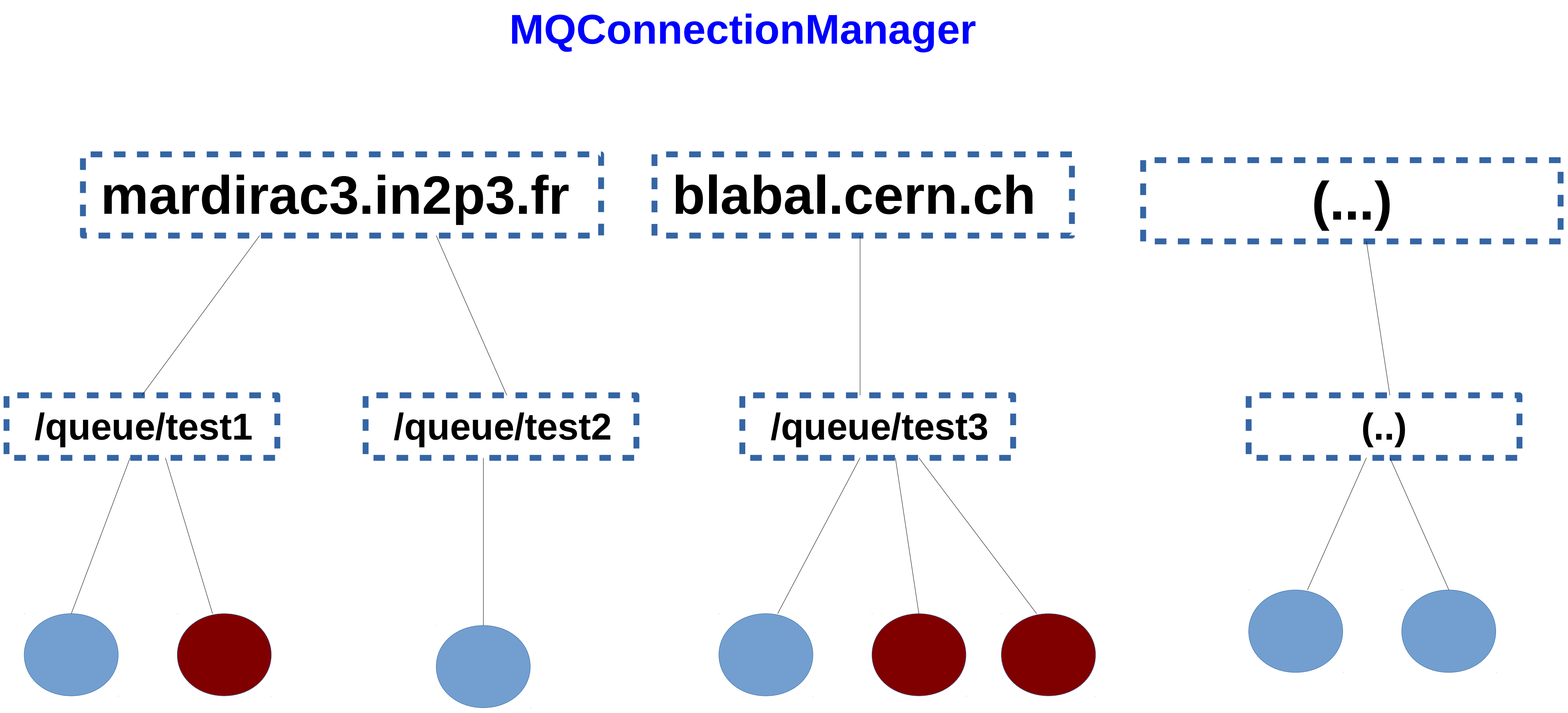
```
Resources
{
  MQServices
  {
    mardirac3.in2p3.fr
    {
      MQType = Stomp
      Host = mardirac3.in2p3.fr
      Port = 9165
      User = guest
      Password = guest
      Queues
      {
        TestQueue
        {
          Acknowledgement =
True
          Persistent = False
        }
      }
    }
  }
}
```

A bit of details



```
Resources
{
  MQServices
  {
    mardirac3.in2p3.fr
    {
      MQType = Stomp
      Host = mardirac3.in2p3.fr
      Port = 9165
      User = guest
      Password = guest
      Queues
      {
        TestQueue
        {
          Acknowledgement =
True
          Persistent = False
        }
      }
    }
  }
}
```

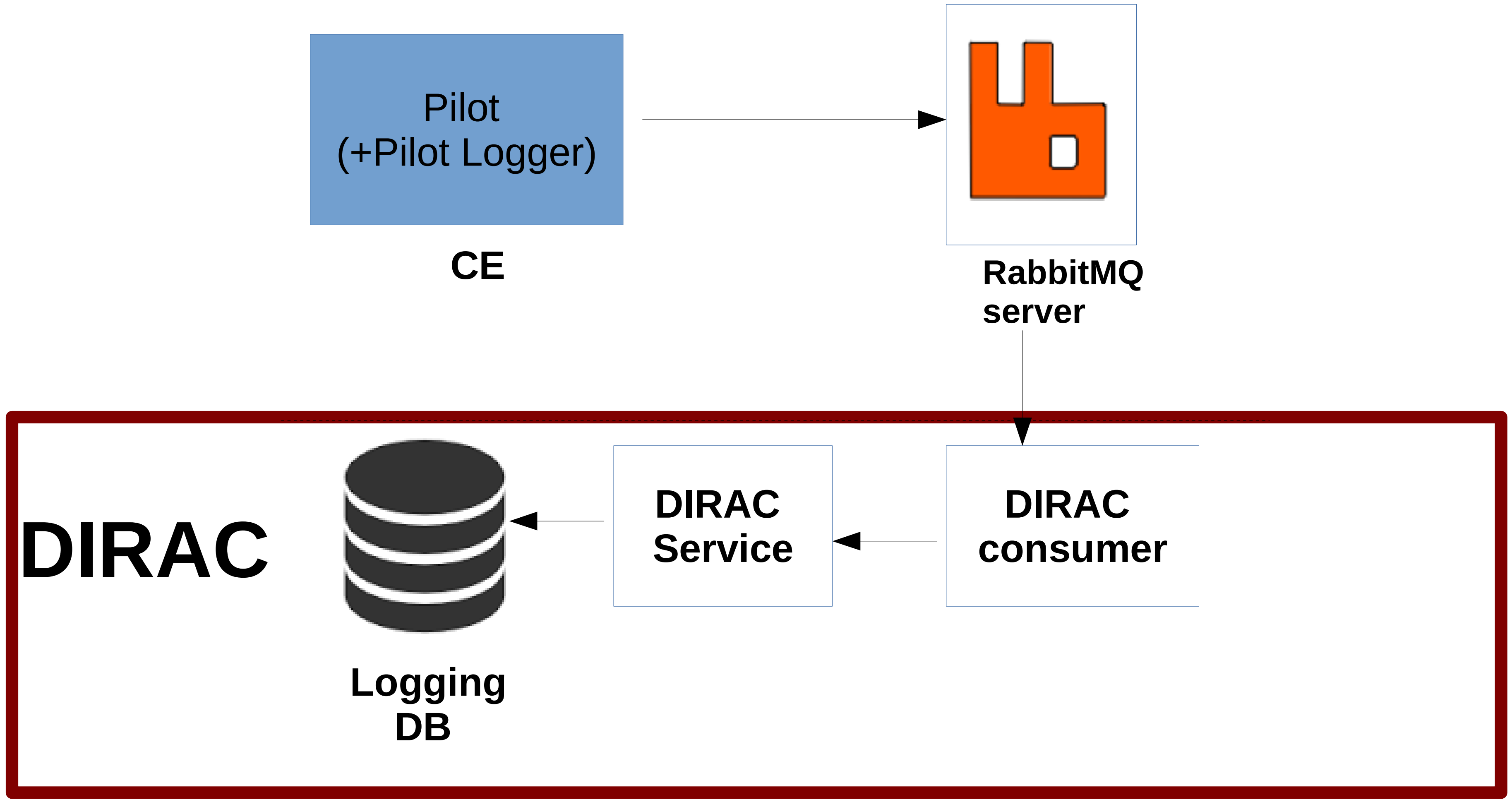
A bit of details II



- Connections to MQ servers can be reused
- **MQConnectionManager** internally manages connections
- Thread-safety is assured

MQ usage example

MQ can be used as a part of **Pilot Logger** architecture



Summary

- Message Queueing as established communication scheme for scalable, distributed computing
- General MQ interface included since **DIRAC v6r17**
- STOMP implementation available as ***technology preview*** since **DIRAC v6r17**
- RabbitMQ administration API included
- Tests performed with RabbitMQ server with SSL support

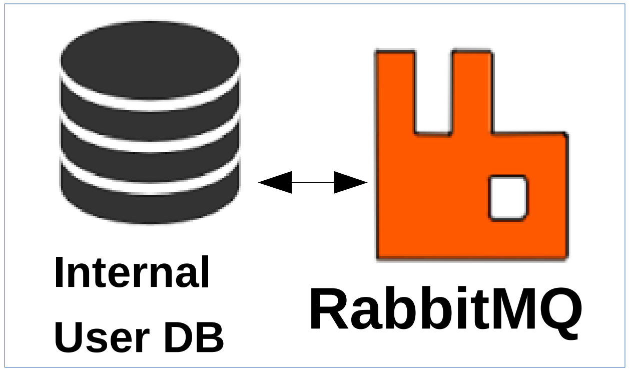
Example usage

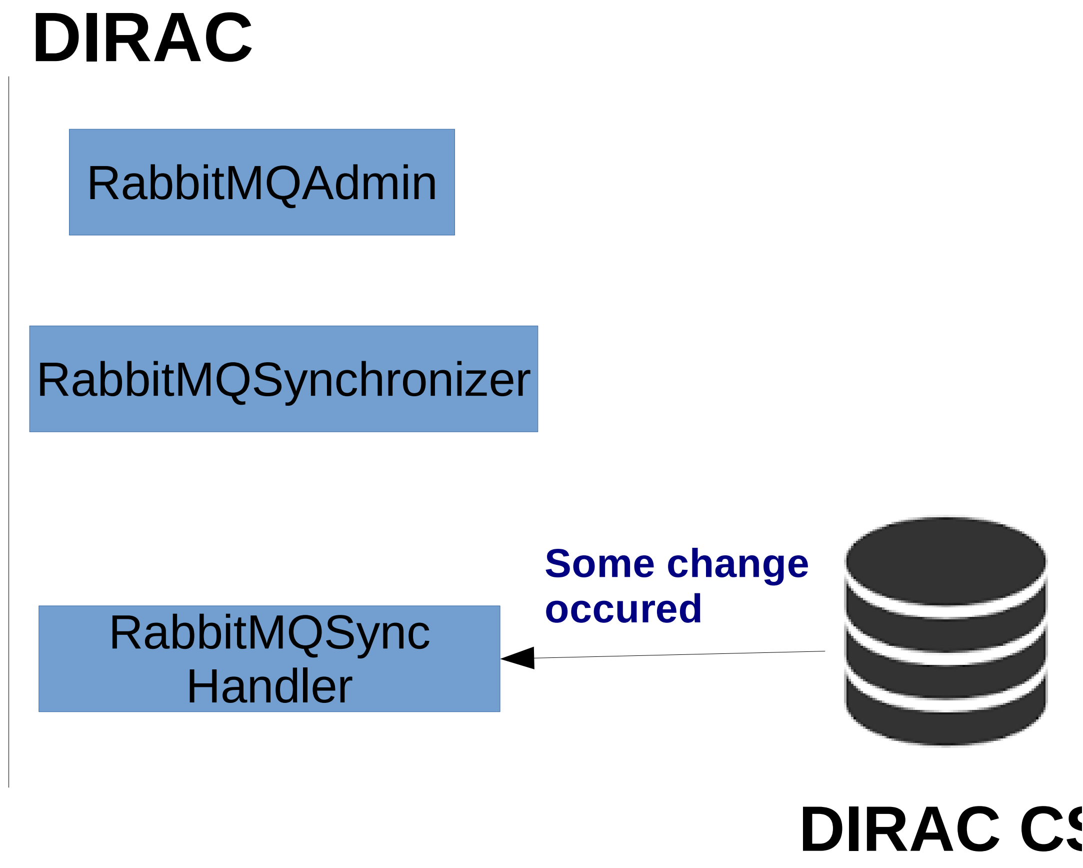
- MQ as a part of **Pilot Logger** architecture
- MQ used as a compontent in the **perfSONAR-DIRAC** bridge

Thank you

Sync between DIRAC CS and RabbitMQ

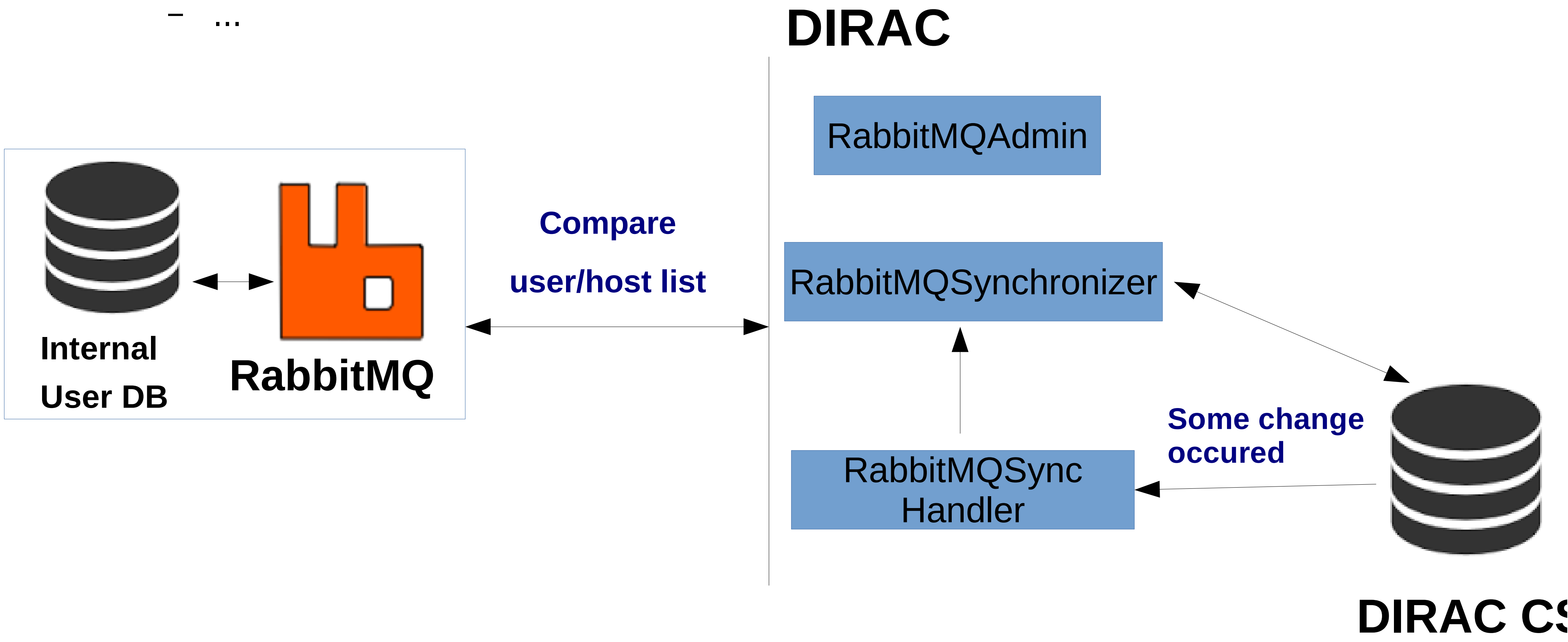
- RabbitMQ has internal User DB, with user loggins and authentication rights
- It should be synchronized with DIRAC Configuration Service (CS)
- RabbitMQAdmin module:
  - AddUser()
  - SetUserPermission()
  - DeleteUser()
  - ...





Sync between DIRAC CS and RabbitMQ

- RabbitMQ has internal User DB, with user loggins and authentication rights,
- It should be synchronized with DIRAC Configuration Service (CS)
- RabbitMQAdmin module:
  - AddUser()
  - SetUserPermission()
  - DeleteUser()
  - ...


DIRAC
RabbitMQAdmin
Compare
user/host list
RabbitMQSynchronizer
Internal
User DB
RabbitMQ
Some change
occured
RabbitMQSync
Handler
DIRAC CS