RabbitMQ with PHP-Direct Exchange

* *Direct Exchange*

The Direct exchange type routes messages with a routing key equal to the routing key declared by the binding queue. This type is useful when you would like to distinguish messages published to the same exchange using a simple string identifier

* *The flow of a message in Direct Exchange*
* A Producer sends a message to an Exchange.
* A Queue binds to an Exchange using a routing key (Usually, there is more than one Queue bind to an Exchange using the same/different routing keys).
* The message sent to the Exchange contains a routing key. Based on the routing key the message is forwarded to one or more Queues.
* Consumers subscribe to the Queue receives the message and processes it.
* *Create your publisher*

<?php

require(\_\_DIR\_\_.'/vendor/autoload.php');

include(\_\_DIR\_\_ . '/config.php');

use PhpAmqpLib\Connection\AMQPStreamConnection;

use PhpAmqpLib\Exchange\AMQPExchangeType;

use PhpAmqpLib\Message\AMQPMessage;

*/\**

*0- setup rabbitMQ connecting*

*1- create channel for new connection*

*2- define variables and routing key*

*3- declare exchange*

*4- declare queue*

*5- bind queue to exchange*

*6- send message*

*7- close channel*

*8- close connection*

*\*/*

$exchange = 'employees';

$meat\_queue = 'meat';

$vegetables\_queue = 'vegetables';

$meat\_routing\_key = 'meat';

$vegetables\_routing\_key = 'vegetables';

$connection = new AMQPStreamConnection(HOST, PORT, USER, PASS, VHOST);

$channel = $connection->channel();

*/\**

*name: $exchange*

*type: direct*

*passive: false //don't check if an exchange with the same name exists*

*durable: true // the exchange will survive server restarts*

*//(if true contine from stored progress else start the begining)*

*//(Metadata of a durable queue is stored on disk, Durable queues will be recovered on node boot, including messages in them published as persistent)*

*auto\_delete: false //the exchange won't be deleted once the channel is closed.*

*\*/*

$channel->exchange\_declare($exchange, AMQPExchangeType::DIRECT, false, true, false);

*/\**

*name: $queue*

*passive: false //don't check if an queue with the same name exists*

*durable: true // the queue will survive server restarts*

*//(if true contine from stored progress else start the begining)*

*//(Metadata of a durable queue is stored on disk, Durable queues will be recovered on node boot, including messages in them published as persistent)*

*exclusive: false // the queue can be accessed in other channels*

*auto\_delete: false //the queue won't be deleted once the channel is closed.*

*\*/*

$channel->queue\_declare($meat\_queue, false, true, false, false);

$channel->queue\_declare($vegetables\_queue, false, true, false, false);

$channel->queue\_bind($meat\_queue, $exchange, $meat\_routing\_key);

$channel->queue\_bind($vegetables\_queue, $exchange, $vegetables\_routing\_key);

*/\* Messages marked as 'persistent' that are delivered to 'durable' queues will be logged to disk. Durable queues are recovered in the event of a crash \*/*

$faker = Faker\Factory::create();

for ($i = 1; $i <= 100; $i++) {

    if($i % 2 == 0){

        $messageBody = json\_encode(['name'=>$faker->name, 'email'=>$faker->email, 'phone'=>$faker->PhoneNumber(),  'meal'=>'meat']);

        $message = new AMQPMessage($messageBody, ['content\_type' => 'application/json', 'delivery\_mode' => AMQPMessage::DELIVERY\_MODE\_PERSISTENT]);

        $channel->basic\_publish($message, $exchange, $meat\_routing\_key);

    }else{

        $messageBody = json\_encode(['name'=>$faker->name, 'email'=>$faker->email, 'phone'=>$faker->PhoneNumber(),  'meal'=>'vegetables']);

        $message = new AMQPMessage($messageBody, ['content\_type' => 'application/json', 'delivery\_mode' => AMQPMessage::DELIVERY\_MODE\_PERSISTENT]);

        $channel->basic\_publish($message, $exchange, $vegetables\_routing\_key);

    }

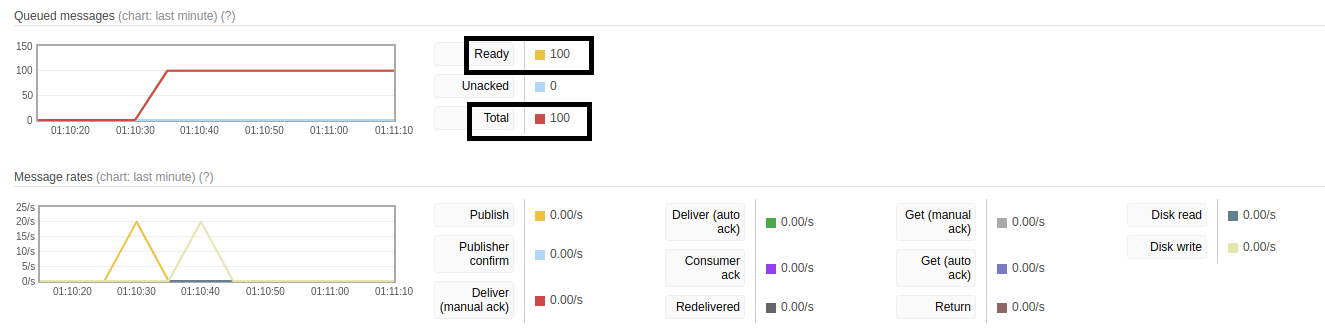
}

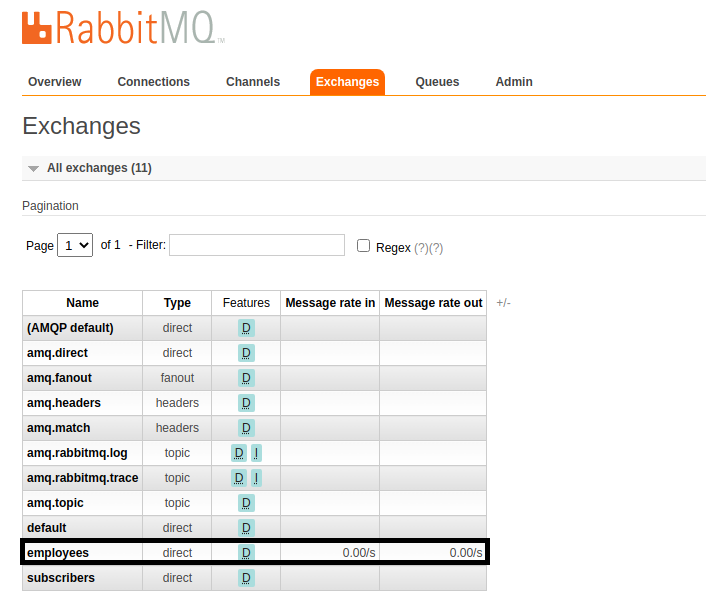
echo 'Finish publishing to queues'.*PHP\_EOL*;

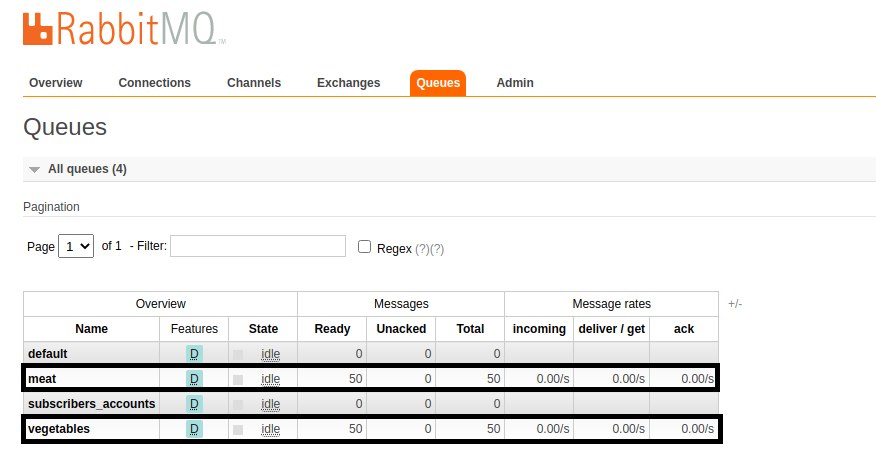
$channel->close();

$connection->close();

* *Navigate RabbitMQ Management Plugin*

**

**

**

* First Queue Consumer (Meat Queue)

<?php

require(\_\_DIR\_\_.'/vendor/autoload.php');

include(\_\_DIR\_\_ . '/config.php');

use PhpAmqpLib\Connection\AMQPStreamConnection;

use PhpAmqpLib\Exchange\AMQPExchangeType;

$exchange = 'employees';

$meat\_queue = 'meat';

$meat\_routing\_key = 'meat';

$consumerTag = '';

$connection = new AMQPStreamConnection(HOST, PORT, USER, PASS, VHOST);

$channel = $connection->channel();

*/\**

*name: $exchange*

*type: direct*

*passive: false //don't check if an exchange with the same name exists*

*durable: true // the exchange will survive server restarts*

*//(if true contine from stored progress else start the begining)*

*//(Metadata of a durable queue is stored on disk, Durable queues will be recovered on node boot, including messages in them published as persistent)*

*auto\_delete: false //the exchange won't be deleted once the channel is closed.*

*\*/*

$channel->exchange\_declare($exchange, AMQPExchangeType::DIRECT, false, true, false);

*/\**

*name: $queue*

*passive: false //don't check if an queue with the same name exists*

*durable: true // the queue will survive server restarts*

*//(if true contine from stored progress else start the begining)*

*//(Metadata of a durable queue is stored on disk, Durable queues will be recovered on node boot, including messages in them published as persistent)*

*exclusive: false // the queue can be accessed in other channels*

*auto\_delete: false //the queue won't be deleted once the channel is closed.*

*\*/*

$channel->queue\_declare($meat\_queue, false, true, false, false);

$channel->queue\_bind($meat\_queue, $exchange, $meat\_routing\_key);

*/\*\**

*\* @param \PhpAmqpLib\Message\AMQPMessage $message*

*\*/*

function process\_message($message)

{

    $message\_body = json\_decode($message->body);

    writeDataToFile($message\_body);

    $message->ack();

*// Send a message with the string "quit" to cancel the consumer.*

    if ($message->body === 'quit') {

        $message->getChannel()->basic\_cancel($message->getConsumerTag());

    }

}

*/\**

*queue: Queue from where to get the messages*

*consumer\_tag: Consumer identifier*

*no\_local: Don't receive messages published by this consumer.*

*no\_ack: If set to true, automatic acknowledgement mode will be used by this consumer. See https://www.rabbitmq.com/confirms.html for details.*

*exclusive: Request exclusive consumer access, meaning only this consumer can access the queue*

*nowait:*

*callback: A PHP Callback*

*\*/*

$channel->basic\_consume($meat\_queue, $consumerTag, false, false, false, false, 'process\_message');

*/\*\**

*\* @param \PhpAmqpLib\Channel\AMQPChannel $channel*

*\* @param \PhpAmqpLib\Connection\AbstractConnection $connection*

*\*/*

function shutdown($channel, $connection)

{

    $channel->close();

    $connection->close();

}

register\_shutdown\_function('shutdown', $channel, $connection);

*// Loop as long as the channel has callbacks registered*

while ($channel->is\_consuming()) {

    $channel->wait();

}

function writeDataToFile($message)

{

    echo $message->name . "\n";

    $folder\_location = 'uploads/';

    $file\_path = 'uploads/' . strtolower(trim(str\_replace(' ', '\_', 'meat\_'.$message->name)));

*//check path*

    if (!file\_exists($folder\_location)) {

        mkdir($folder\_location, 0777, true);

    }

*//check file to open*

    $file\_object = fopen($file\_path, 'a+');

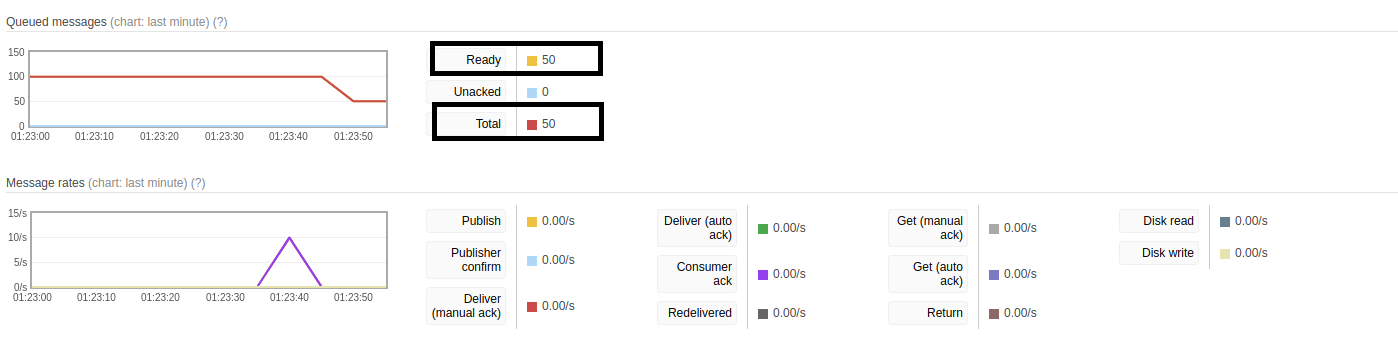
*//write data to file*

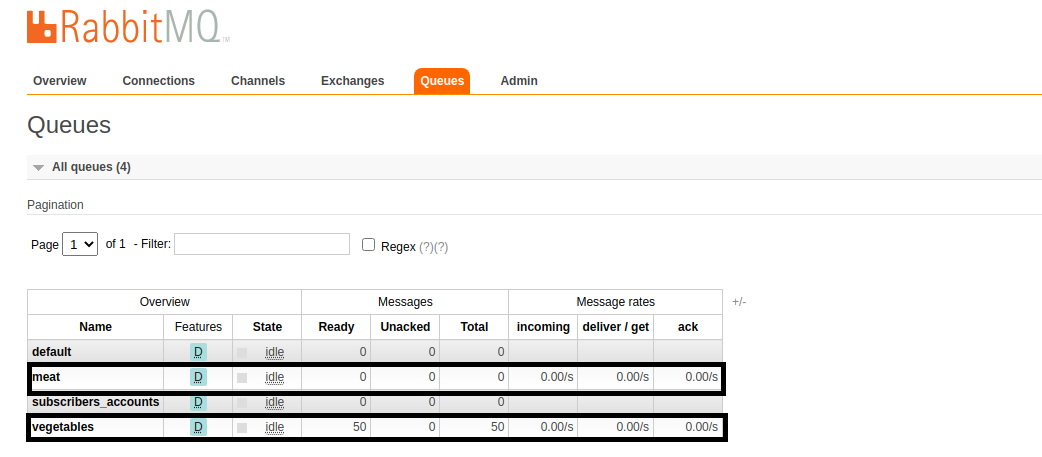
    fwrite($file\_object, $message->name . "\n" .$message->email. "\n" .$message->phone);

    fclose($file\_object);

}

* *Navigate RabbitMQ Management Plugin*





* Second Queue Consumer (vegetables Queue)

<?php

require(\_\_DIR\_\_.'/vendor/autoload.php');

include(\_\_DIR\_\_ . '/config.php');

use PhpAmqpLib\Connection\AMQPStreamConnection;

use PhpAmqpLib\Exchange\AMQPExchangeType;

$exchange = 'employees';

$vegetables\_queue = 'vegetables';

$vegetables\_routing\_key = 'vegetables';

$consumerTag = '';

$connection = new AMQPStreamConnection(HOST, PORT, USER, PASS, VHOST);

$channel = $connection->channel();

*/\**

*name: $exchange*

*type: direct*

*passive: false //don't check if an exchange with the same name exists*

*durable: true // the exchange will survive server restarts*

*//(if true contine from stored progress else start the begining)*

*//(Metadata of a durable queue is stored on disk, Durable queues will be recovered on node boot, including messages in them published as persistent)*

*auto\_delete: false //the exchange won't be deleted once the channel is closed.*

*\*/*

$channel->exchange\_declare($exchange, AMQPExchangeType::DIRECT, false, true, false);

*/\**

*name: $queue*

*passive: false //don't check if an queue with the same name exists*

*durable: true // the queue will survive server restarts*

*//(if true contine from stored progress else start the begining)*

*//(Metadata of a durable queue is stored on disk, Durable queues will be recovered on node boot, including messages in them published as persistent)*

*exclusive: false // the queue can be accessed in other channels*

*auto\_delete: false //the queue won't be deleted once the channel is closed.*

*\*/*

$channel->queue\_declare($vegetables\_queue, false, true, false, false);

$channel->queue\_bind($vegetables\_queue, $exchange, $vegetables\_routing\_key);

*/\*\**

*\* @param \PhpAmqpLib\Message\AMQPMessage $message*

*\*/*

function process\_message($message)

{

    $message\_body = json\_decode($message->body);

    writeDataToFile($message\_body);

    $message->ack();

*// Send a message with the string "quit" to cancel the consumer.*

    if ($message->body === 'quit') {

        $message->getChannel()->basic\_cancel($message->getConsumerTag());

    }

}

*/\**

*queue: Queue from where to get the messages*

*consumer\_tag: Consumer identifier*

*no\_local: Don't receive messages published by this consumer.*

*no\_ack: If set to true, automatic acknowledgement mode will be used by this consumer. See https://www.rabbitmq.com/confirms.html for details.*

*exclusive: Request exclusive consumer access, meaning only this consumer can access the queue*

*nowait:*

*callback: A PHP Callback*

*\*/*

$channel->basic\_consume($vegetables\_queue, $consumerTag, false, false, false, false, 'process\_message');

*/\*\**

*\* @param \PhpAmqpLib\Channel\AMQPChannel $channel*

*\* @param \PhpAmqpLib\Connection\AbstractConnection $connection*

*\*/*

function shutdown($channel, $connection)

{

    $channel->close();

    $connection->close();

}

register\_shutdown\_function('shutdown', $channel, $connection);

*// Loop as long as the channel has callbacks registered*

while ($channel->is\_consuming()) {

    $channel->wait();

}

function writeDataToFile($message)

{

    echo $message->name . "\n";

    $folder\_location = 'uploads/';

    $file\_path = 'uploads/' . strtolower(trim(str\_replace(' ', '\_', 'vegetables\_'.$message->name)));

*//check path*

    if (!file\_exists($folder\_location)) {

        mkdir($folder\_location, 0777, true);

    }

*//check file to open*

    $file\_object = fopen($file\_path, 'a+');

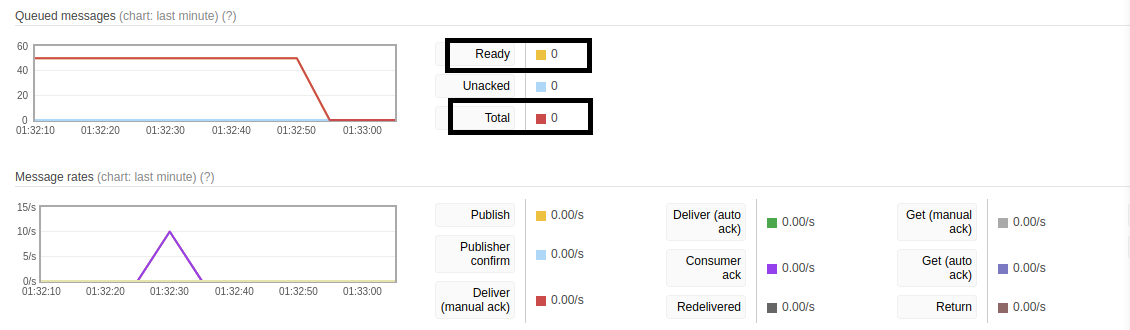
*//write data to file*

    fwrite($file\_object, $message->name . "\n" .$message->email. "\n" .$message->phone);

    fclose($file\_object);

}

* *Navigate RabbitMQ Management Plugin*

**

