RabbitMQ with PHP-Fanout Exchange

* *Fanout Exchange*

The Fanout exchange type routes messages to all bound queues indiscriminately. If a routing key is provided, it will simply be ignored.

* *The flow of a message in Fanout Exchange*
* One or more Queues bind to the Fanout exchange with no routing keys.
* A publisher sends the Exchange a message.
* The Exchange then forwards the message to the Queues unconditionally.
* *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 = 'fanout\_exchange';

$orders\_queue = 'orders';

$casher\_queue = 'casher';

$delivery\_queue = 'delivery';

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

$channel = $connection->channel();

*/\**

*name: $exchange*

*type: fanout*

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

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

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

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

*auto\_delete: true //the exchange will be deleted once the channel is closed.*

*\*/*

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

*/\**

*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.*

*\*/*

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

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

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

$channel->queue\_bind($orders\_queue, $exchange);

$channel->queue\_bind($casher\_queue, $exchange);

$channel->queue\_bind($delivery\_queue, $exchange);

$faker = Faker\Factory::create();

for ($i = 0; $i < 100; $i++) {

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

    $message = new AMQPMessage($messageBody, array('content\_type' => 'application/json'));

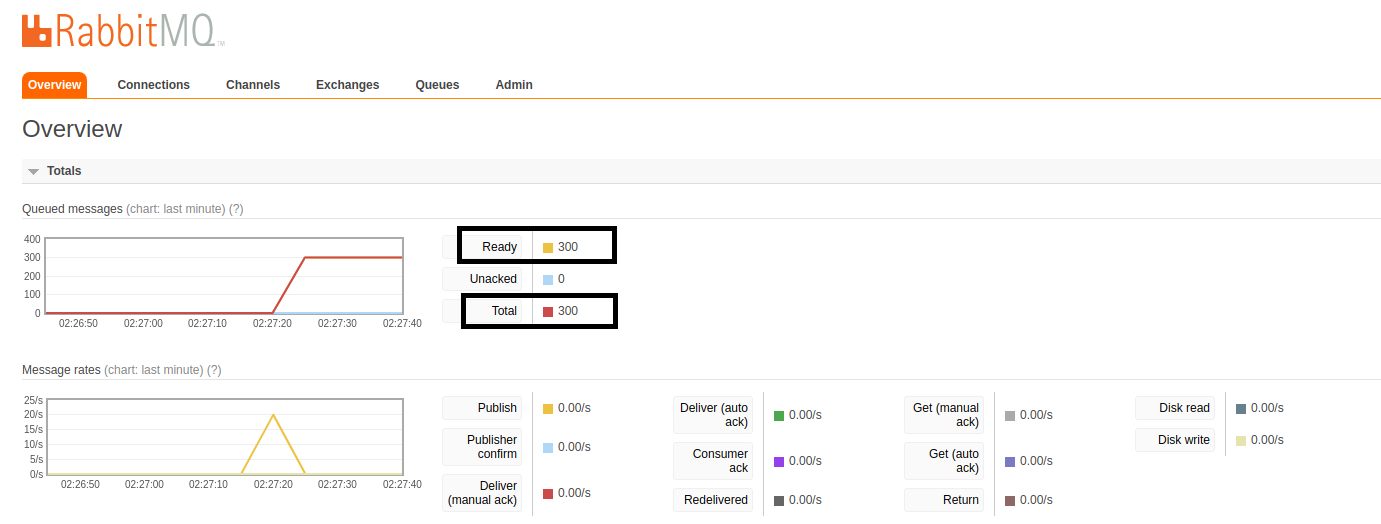
    $channel->basic\_publish($message, $exchange);

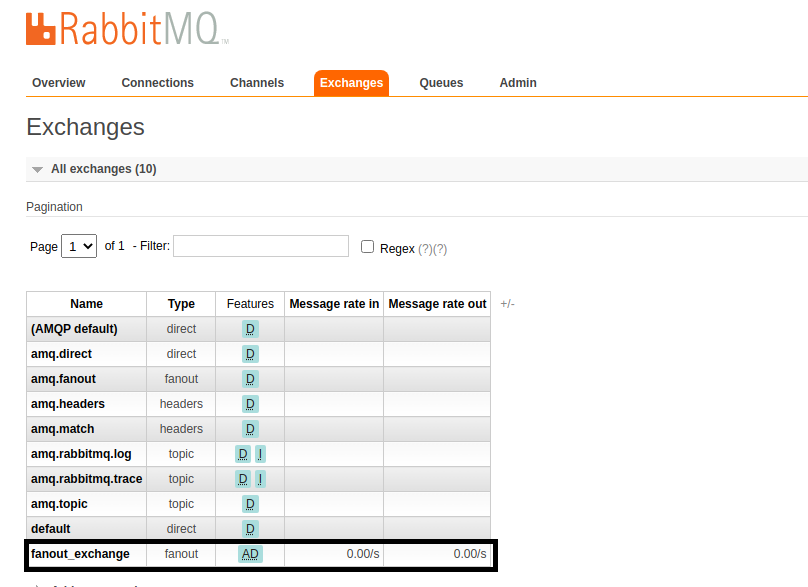
}

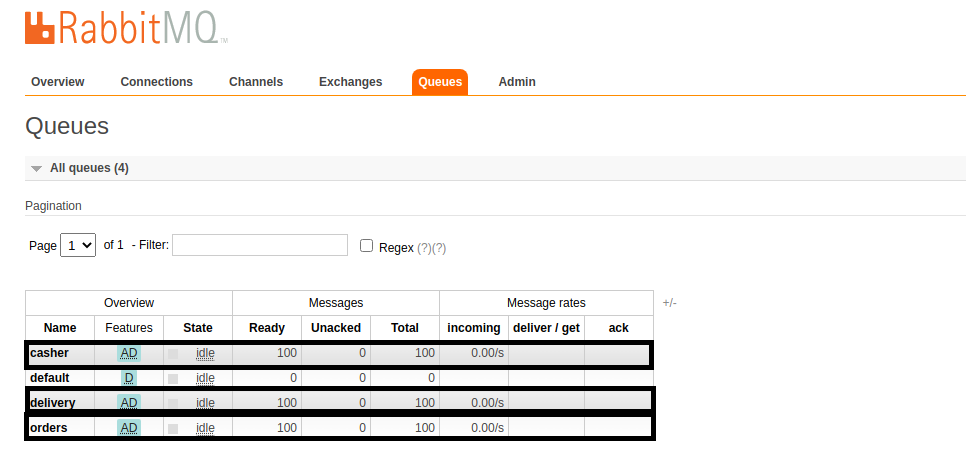
$channel->close();

$connection->close();

* *Navigate RabbitMQ Management Plugin*







* First Queue Consumer (Casher Queue)

<?php

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

include(\_\_DIR\_\_ . '/config.php');

use PhpAmqpLib\Connection\AMQPStreamConnection;

use PhpAmqpLib\Exchange\AMQPExchangeType;

$exchange = 'fanout\_exchange';

$queue = 'casher';

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

$channel = $connection->channel();

*/\**

*name: $exchange*

*type: fanout*

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

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

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

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

*auto\_delete: true //the exchange will be deleted once the channel is closed.*

*\*/*

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

*/\**

*name: $queue    // should be unique in fanout exchange.*

*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 might be accessed by other channels*

*auto\_delete: true //the queue will be deleted once the channel is closed.*

*\*/*

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

$channel->queue\_bind($queue, $exchange);

*/\*\**

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

*\*/*

function process\_message($message)

{

    echo "\n--------\n";

    echo $message->body;

    echo "\n--------\n";

    $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: don't wait for a server response. In case of error the server will raise a channel*

*exception*

*callback: A PHP Callback*

*\*/*

$channel->basic\_consume($queue, '', 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);

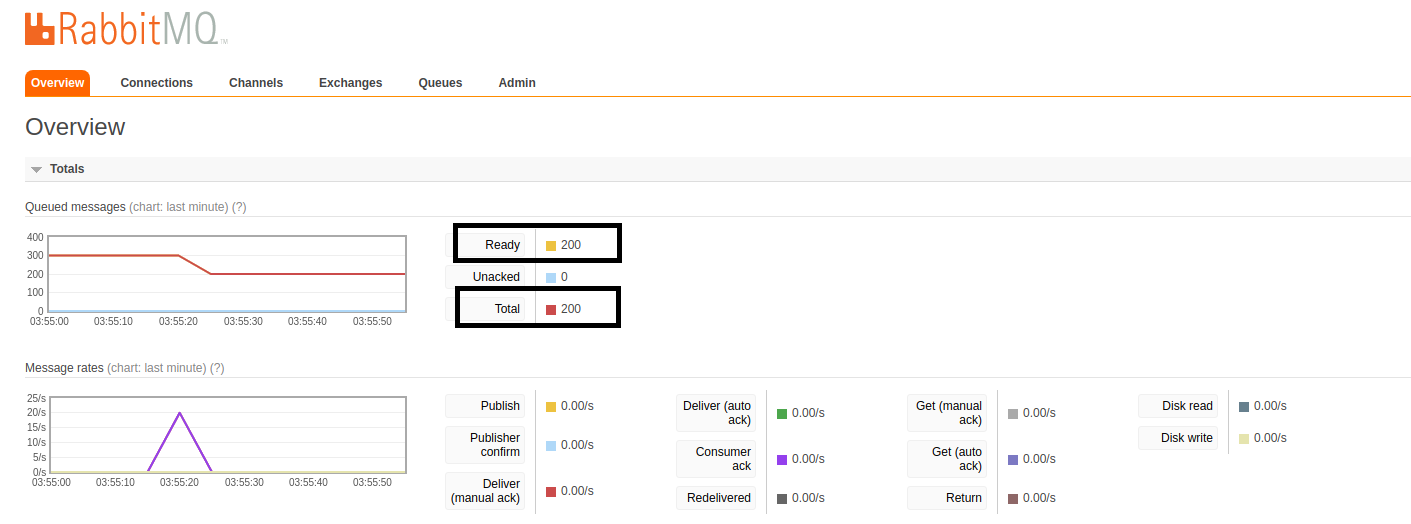
*// Loop as long as the channel has callbacks registered*

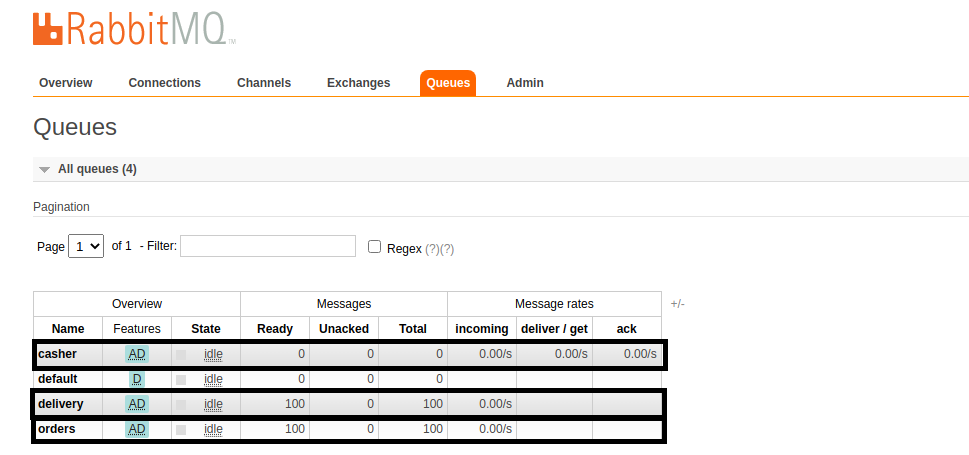
while ($channel->is\_consuming()) {

    $channel->wait();

}

* *Navigate RabbitMQ Management Plugin*





* Second Queue Consumer (delivery Queue)

<?php

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

include(\_\_DIR\_\_ . '/config.php');

use PhpAmqpLib\Connection\AMQPStreamConnection;

use PhpAmqpLib\Exchange\AMQPExchangeType;

$exchange = 'fanout\_exchange';

$queue = 'delivery';

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

$channel = $connection->channel();

*/\**

*name: $exchange*

*type: fanout*

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

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

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

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

*auto\_delete: true //the exchange will be deleted once the channel is closed.*

*\*/*

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

*/\**

*name: $queue    // should be unique in fanout exchange.*

*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 might be accessed by other channels*

*auto\_delete: true //the queue will be deleted once the channel is closed.*

*\*/*

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

$channel->queue\_bind($queue, $exchange);

*/\*\**

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

*\*/*

function process\_message($message)

{

    echo "\n--------\n";

    echo $message->body;

    echo "\n--------\n";

    $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: don't wait for a server response. In case of error the server will raise a channel*

*exception*

*callback: A PHP Callback*

*\*/*

$channel->basic\_consume($queue, '', 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);

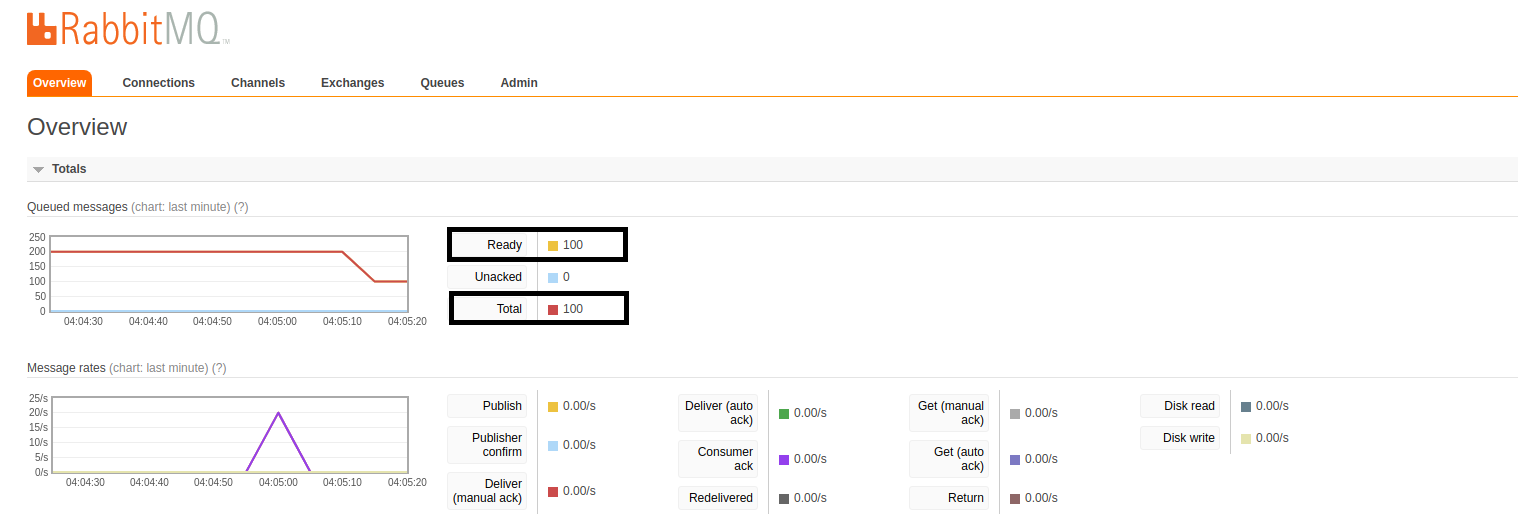
*// Loop as long as the channel has callbacks registered*

while ($channel->is\_consuming()) {

    $channel->wait();

}

* *Navigate RabbitMQ Management Plugin*



* Third Queue Consumer (orders Queue)

<?php

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

include(\_\_DIR\_\_ . '/config.php');

use PhpAmqpLib\Connection\AMQPStreamConnection;

use PhpAmqpLib\Exchange\AMQPExchangeType;

$exchange = 'fanout\_exchange';

$queue = 'orders';

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

$channel = $connection->channel();

*/\**

*name: $exchange*

*type: fanout*

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

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

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

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

*auto\_delete: true //the exchange will be deleted once the channel is closed.*

*\*/*

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

*/\**

*name: $queue    // should be unique in fanout exchange.*

*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 might be accessed by other channels*

*auto\_delete: true //the queue will be deleted once the channel is closed.*

*\*/*

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

$channel->queue\_bind($queue, $exchange);

*/\*\**

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

*\*/*

function process\_message($message)

{

    echo "\n--------\n";

    echo $message->body;

    echo "\n--------\n";

    $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: don't wait for a server response. In case of error the server will raise a channel*

*exception*

*callback: A PHP Callback*

*\*/*

$channel->basic\_consume($queue, '', 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();

}

* *Navigate RabbitMQ Management Plugin*

