RabbitMQ in Docker

Supporting repo is here. (Includes a Celery example)

If you don't already have Docker installed, you can find instructions here.

Setup and Run

Pull the RabbitMQ docker container:

```
docker pull rabbitmq
```

Startup for RabbitMQ docker container:

```
sudo docker run -d --hostname my-rabbit --name some-rabbit -p 5672:5672
rabbitmq:3
```

The port mapping (5672:5672) is not included in the instructions on Docker Hub, but it's required for the Python send/receive scripts to work.

Simple Test in Python

You'll need to install the Pika library before you run the send/receive scripts:

```
sudo pip3 install pika --upgrade
```

Python script to send a message:

send.py

```
    #!/usr/bin/env python3
    import pika

    connection =
pika.BlockingConnection(pika.ConnectionParameters('localhost'))
    channel = connection.channel()

    channel.queue_declare(queue='hello')

    channel.basic_publish(exchange='', routing_key='hello', body='Hello
World!')
```

```
    print(" [x] Sent 'Hello World!'")
    connection.close()
```

Python script to receive messages:

receive.py

```
    #!/usr/bin/env python3
    import pika

    connection =
pika.BlockingConnection(pika.ConnectionParameters('localhost'))
    channel = connection.channel()

    channel.queue_declare(queue='hello')

    def callback(ch, method, properties, body):
        print(" [x] Received %r" % body)

    channel.basic_consume(queue='hello', on_message_callback=callback,
auto_ack=True)

    print(' [*] Waiting for messages. To exit press CTRL+C')
    channel.start_consuming()
```

Simple Test in Go

First, install amqp using go get:

```
go get github.com/streadway/amqp
```

Then, use this to send a message:

send.go

```
    package main

    import (
        "log"

        "github.com/streadway/amqp"
    )

    func failOnError(err error, msg string) {
        if err != nil {
            log.Fatalf("%s: %s", msg, err)
```

```
    }
}

func main() {
    conn, err := amqp.Dial("amqp://guest:guest@localhost:5672/")
    failOnError(err, "Failed to connect to RabbitMQ")
    defer conn.Close()

    ch, err := conn.Channel()
    failOnError(err, "Failed to open a channel")
    defer ch.Close()

    q, err := ch.QueueDeclare(
        "hello", // name
        false,   // durable
        false,   // delete when unused
        false,   // exclusive
        false,   // no-wait
        nil,     // arguments
    )
    failOnError(err, "Failed to declare a queue")

    body := "Hello World!"
    err = ch.Publish(
        "",     // exchange
        q.Name, // routing key
        false,  // mandatory
        false,  // immediate
        amqp.Publishing{
            ContentType: "text/plain",
            Body:        []byte(body),
        })
    log.Printf(" [x] Sent %s", body)
    failOnError(err, "Failed to publish a message")
}
```

And use this to receive messages:

receive.go

```
package main

import (
    "log"

    "github.com/streadway/amqp"
)

func failOnError(err error, msg string) {
    if err != nil {
```

```
            log.Fatalf("%s: %s", msg, err)
        }
    }

    func main() {
        conn, err := amqp.Dial("amqp://guest:guest@localhost:5672/")
        failOnError(err, "Failed to connect to RabbitMQ")
        defer conn.Close()

        ch, err := conn.Channel()
        failOnError(err, "Failed to open a channel")
        defer ch.Close()

        q, err := ch.QueueDeclare(
            "hello", // name
            false,   // durable
            false,   // delete when unused
            false,   // exclusive
            false,   // no-wait
            nil,     // arguments
        )
        failOnError(err, "Failed to declare a queue")

        msgs, err := ch.Consume(
            q.Name, // queue
            "",     // consumer
            true,   // auto-ack
            false,  // exclusive
            false,  // no-local
            false,  // no-wait
            nil,    // args
        )
        failOnError(err, "Failed to register a consumer")

        forever := make(chan bool)

        go func() {
            for d := range msgs {
                log.Printf("Received a message: %s", d.Body)
            }
        }()

        log.Printf(" [*] Waiting for messages. To exit press CTRL+C")
        <-forever
    }
```

docker