**Rasphberry compresor**

**Integrantes:**

1. **José Luis Bueno**
2. **Rubén Suárez Beltrán**
3. **Wellington Andrés Martínez.**

**¿En qué consiste nuestra idea?**

En una aplicación distribuida que comprime archivos usando un cluster formado por 2 raspberrys pi como servidores.

**Funcionamiento de nuestra aplicación:**

La aplicación cliente se comunica con el servicio RabbitMQ el cual debe tener previamente instanciados los workers para realizar el paso de mensajes, el cual está distribuido en 2 nodos, estos cuentan cada uno con una cola que almacena los mensajes enviados por los clientes, luego estos mismos nodos utilizan los recursos de la librería Celery para ejecutar las funciones de comprimir los mensajes.

**Listado de opciones que no estuvieron al alcance:**

No es una aplicación bidireccional, es decir, la computadora que tiene instalado el cliente celery, no puede hacer las veces de servidor.

No se pudo hacer que la compresión de imágenes sea en el sevidor.

Para que el clouster aumente su disponibilidad se necesita de balanceo de cargas, esta requiere de 2 interfaces de red, la primera interfaz del router al balanceador y la segunda interfaz del balanceador al clouster, y no se pudo implementar debido a que faltaron recursos.

Solo se utiliza un nodo para el paso de las tareas.

**¿Qué es RabbitMQ?**

Es un software de negociación de mensajes de código abierto que implementa el estándar AMQP, está considerado como middleware de mensajería. La capacidad para operar de forma distribuida y conmutar ante errores son posibles gracias al framework OTP, su servidor está escrito en Erlang

**¿Por qué escogimos RabbitMQ?**

Porque al ser código abierto, no requerimos de una licencia para trabajar con dicho software, aparte es fácil de configurar y los nodos en la red pueden ser fácilmente escalados, por último, posee varias interfaces de programación debido a que soporta múltiples protocolos de mensajes.

**Librerías usadas:**

**Celery:**

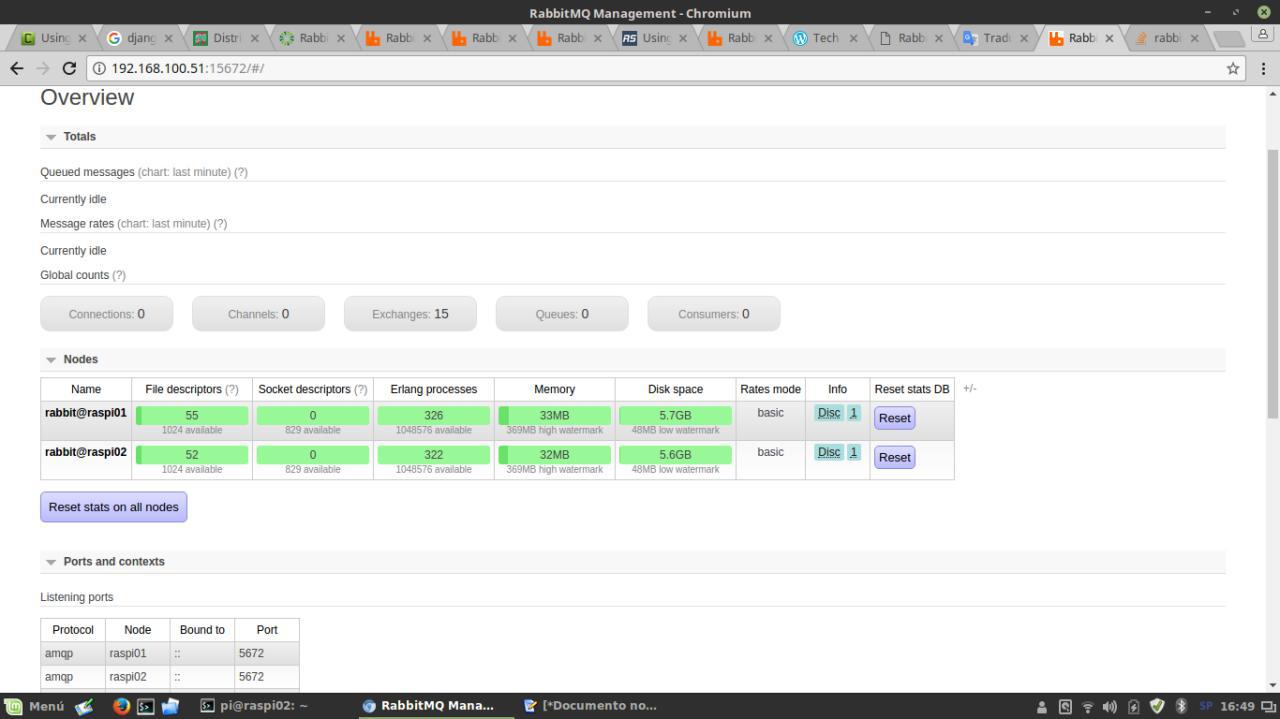
Aplicación que permite crear tareas de trabajo asíncronas gestionadas por un gestor de colas con el fin de enviar mensajes de forma distribuida. Celery puede trabajar con operaciones en tiempo real y con la calendarización de tareas, esto último consiste en trabajar con tareas que se ejecutarán en un momento determinado o de manera periódica. Cada tarea es una unidad de ejecución, y estas se ejecutan de forma concurrente en uno o más nodos de trabajo.

**Repositorio de GitHub:**

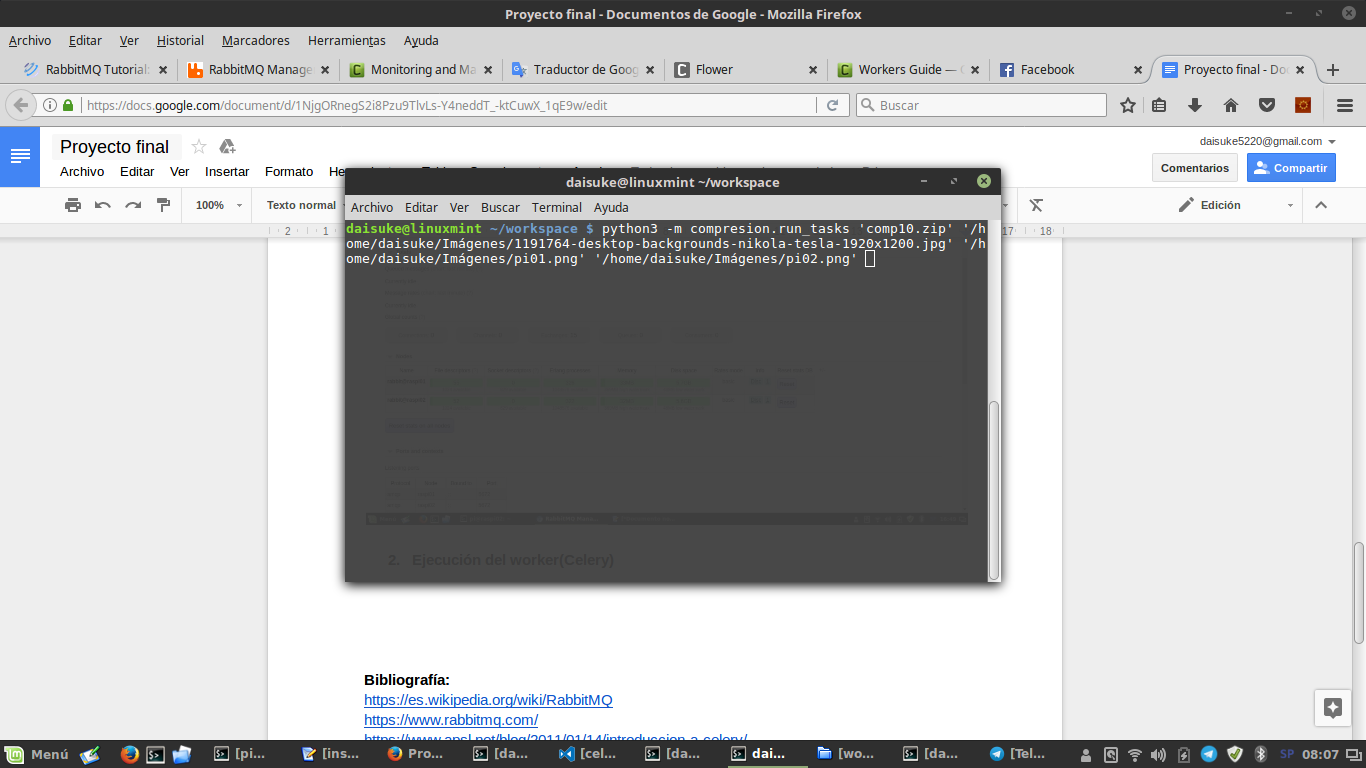
[https://github.com/daisuke245/Multimedia-Compressor](https://l.facebook.com/l.php?u=https%3A%2F%2Fgithub.com%2Fdaisuke245%2FMultimedia-Compressor&h=ATNrmVXtUGDyz-aStvQiwVNSRRel3dhTuq8R3jMUfoTTv8xz4h_ihz8owBpH7gYta0NZX7Oy3_C5MyZ0jQi16CJoT_l6_HY5GGCqXTQOEsW83LQLcUAOr1yi0qdQD7mcIgR0yHk-psNR9MY)

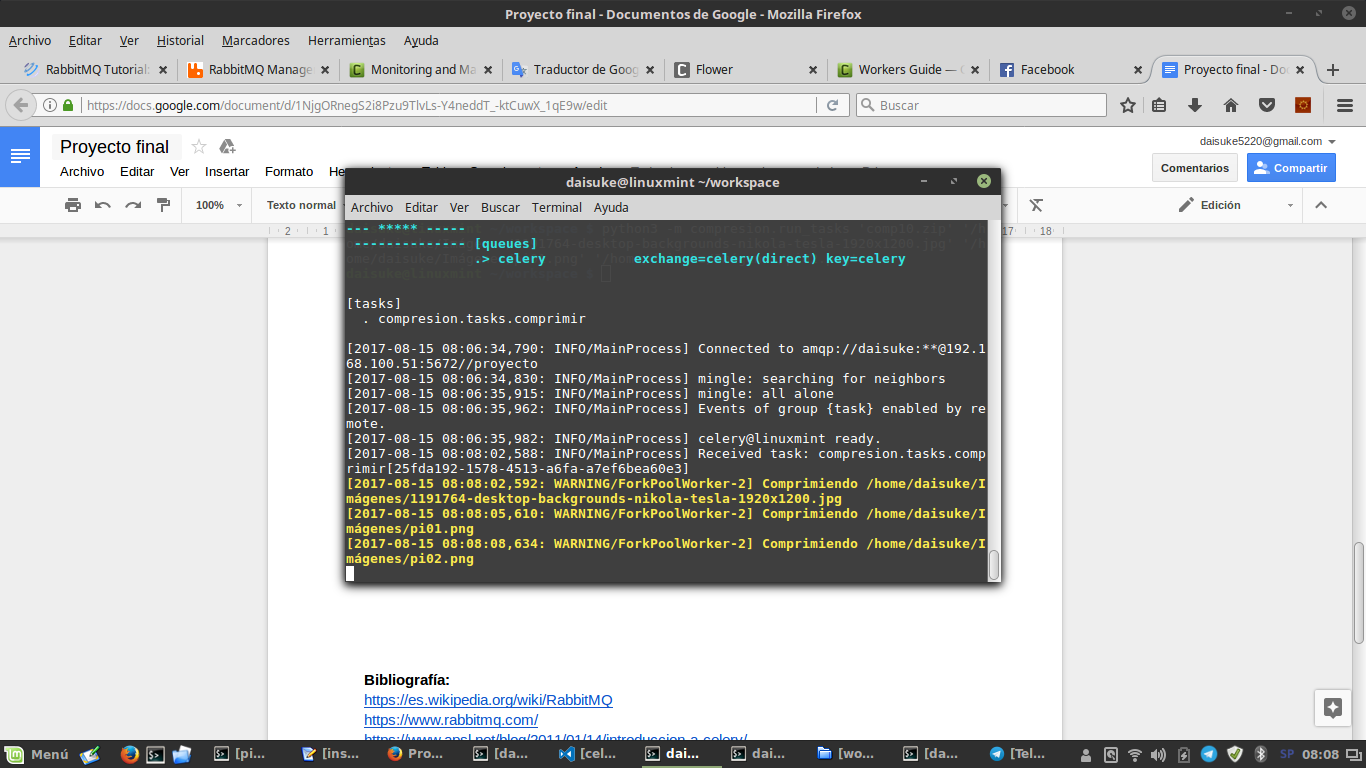
**Pruebas de funcionamiento**

1. **RabbitMQ ejecutándose en todos los nodos.**

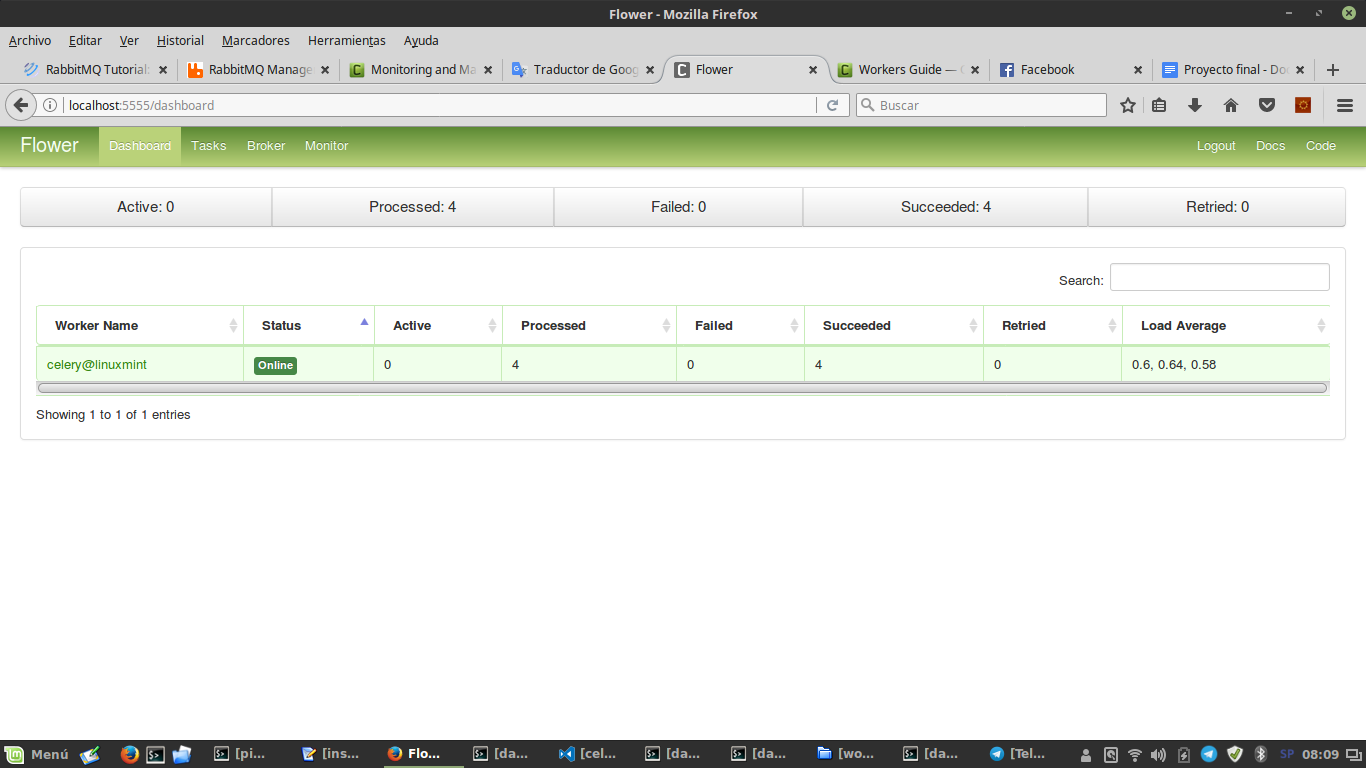


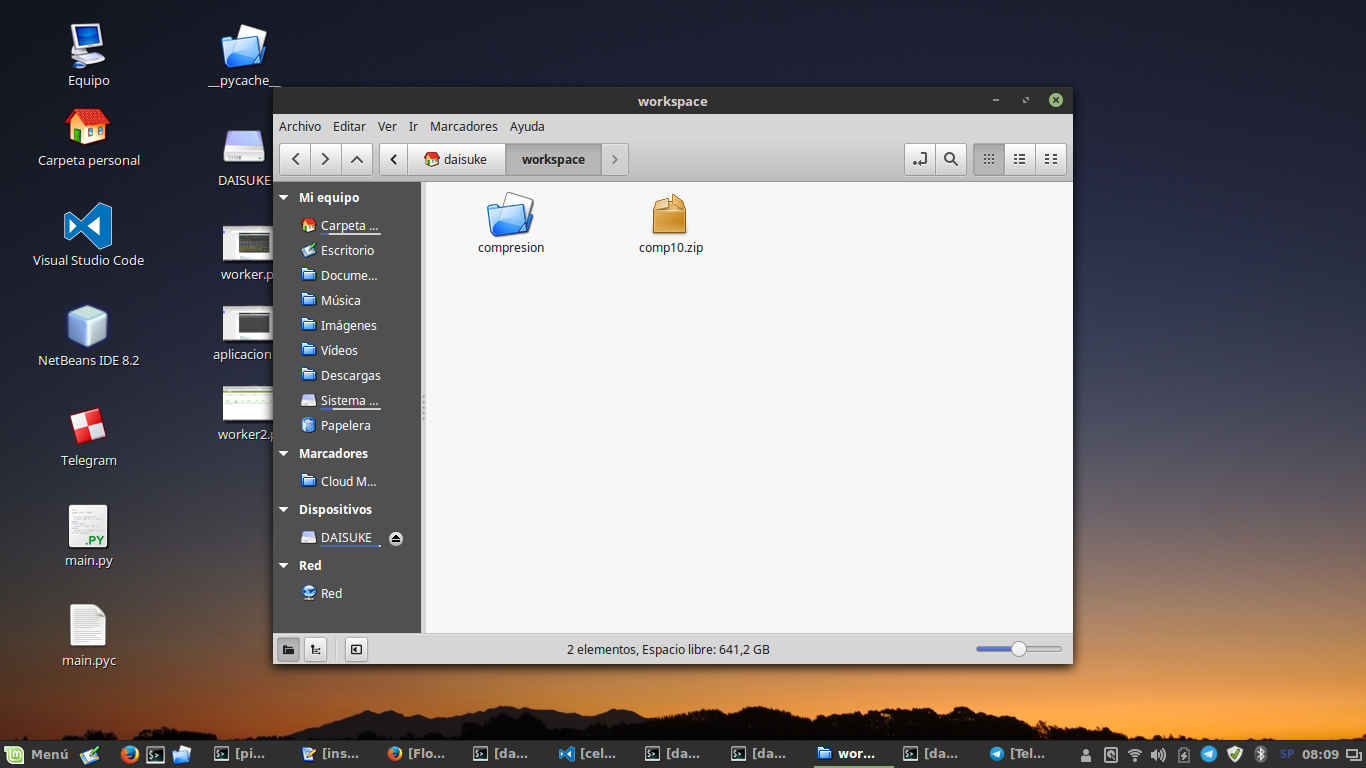
1. **Ejecución del worker(Celery) y de la aplicación.**

****

****

1. **Resultados.**

****

****

**Bibliografía:**

<https://es.wikipedia.org/wiki/RabbitMQ>

<https://www.rabbitmq.com/>

<https://www.apsl.net/blog/2011/01/14/introduccion-a-celery/>