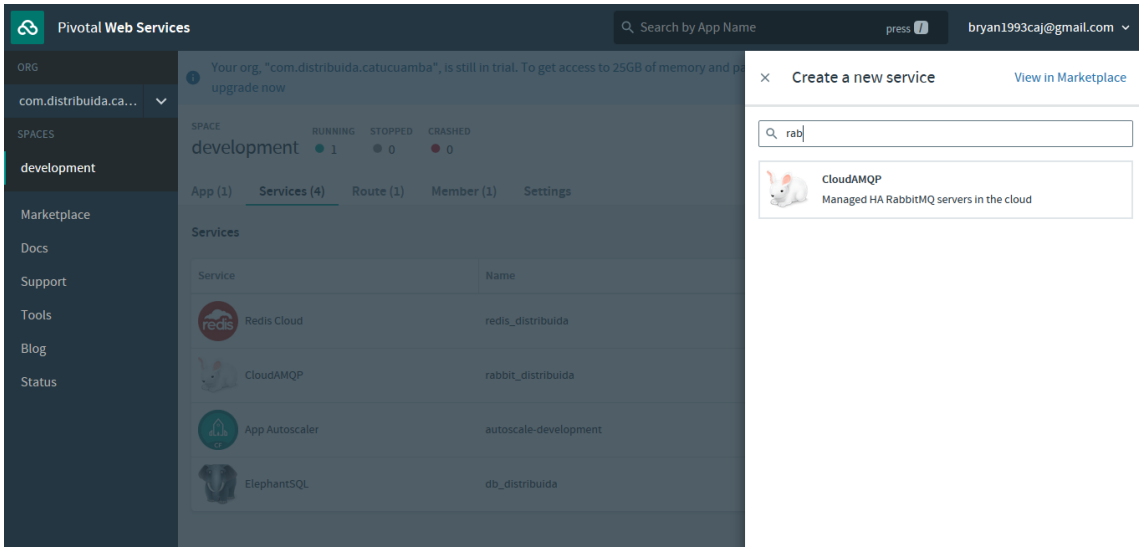
Manual

RabbitMQ en Pivotal

Agregamos el servicio en pivotal



Creamos:

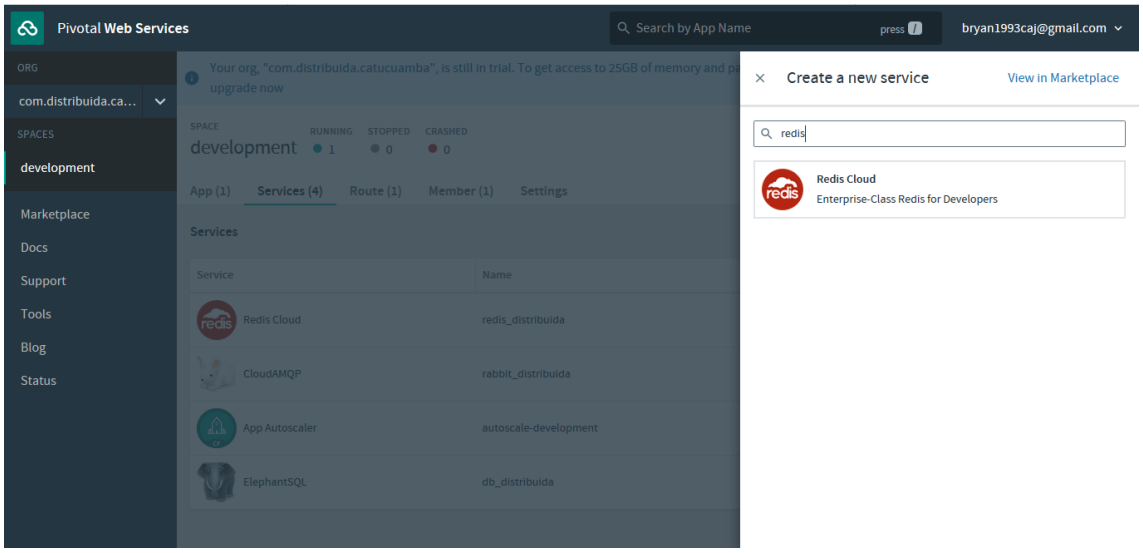
- productor s1 - RabbitTemplate
- consumidor s2 - Listener
- productor web - RabbitTemplate

Agregamos las configuraciones respectivamente de RabbitMQ en el application.properties

```
spring.rabbitmq.username=gvnuppap
spring.rabbitmq.password=E3FvSwV2t5zFG52vdkPbrLE_xcpP4gUT
spring.rabbitmq.host=lion.rmq.cloudamqp.com
spring.rabbitmq.virtual-host=gvnuppap
```

REDIS en Pivotal

Agregamos el servicio en pivotal



Agregamos las configuraciones respectivamente de Redis en el application.properties

```
#redis
spring.redis.host=redis-10830.c52.us-east-1-4.ec2.cloud.redislabs.com
spring.redis.port=10830
spring.redis.password=Av7edTjTVkXnjdhYULFimqQNVhQ3IVbS
```

El manejo lo hacemos en s2 espcificamente

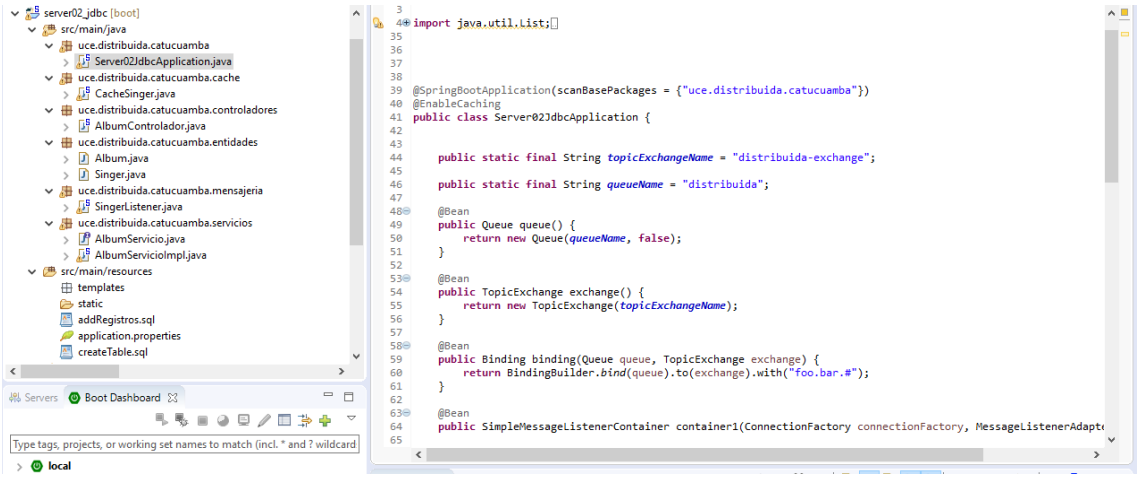
Se implementó la clase CacheSinger, en donde tenemos la lista de los Singer:

```
import java.util.List;

@Component
public class CacheSinger {

    @Autowired
    RestTemplate rt;

    @Cacheable("listaCantantes")
    public List<Singer> listarSingers() {
        System.out.println("Guardando datos en cache");
        List<Singer> singers;
        singers = rt.getForObject("http://localhost:9090/api/server01jdbc/singer/singers" , List.class);
        System.out.println(singers);
        return singers;
    }

}
```

En la clase SingerListener tenemos el método para refrescar el cache, cuando recibe la notificación

```java

@Component
public class SingerListener {


    @Autowired
    RestTemplate rt;

    @CachePut(cacheNames="listaCantantes")
    public List<Singer> receiveMessage(Singer singer) {
        System.out.println(">>>>> actualizando lista de cantantes");
        System.out.println("Refrescando datos en cache");
        List<Singer> singers;
        singers = rt.getForObject("http://localhost:9090/api/server01jdbc/singer/singers" , List.class);
        System.out.println(singers);
        return singers;
    }

}

```

La clase principal, clase boot debe tener la anotación: @EnableCatching



Además de las declaraciones para poder implementar el servicio cache

```java
@Bean
JedisConnectionFactory jedisConnectionFactory() {
    return new JedisConnectionFactory();
}

@Bean
public RedisTemplate<String, Object> redisTemplate() {
    RedisTemplate<String, Object> template = new RedisTemplate<>();
    template.setConnectionFactory(jedisConnectionFactory());
    return template;
}
```

No olvidemos las dependencias necesarias

```
    // https://mvnrepository.com/artifact/org.springframework.data/spring-data-redis
compile group: 'org.springframework.data', name: 'spring-data-redis', version: '2.0.9.RELEASE'
    // https://mvnrepository.com/artifact/redis.clients/jedis
compile group: 'redis.clients', name: 'jedis', version: '2.9.0'
```