**RabbitMQ**

RabbitMQ, mesaj kuyruğu (message queue) olarak bilinen bir yazılım altyapısıdır. Açık kaynaklıdır ve genellikle dağıtık sistemlerdeki bileşenler arasında mesajların güvenli, asenkron ve yönetilebilir bir şekilde iletilmesi için kullanılır. RabbitMQ, AMQP (Advanced Message Queuing Protocol) başta olmak üzere çeşitli mesajlaşma protokollerini destekler. Earlang Programlama dili ile geliştirilmiştir. Pivotal şirketi satın almıştır.

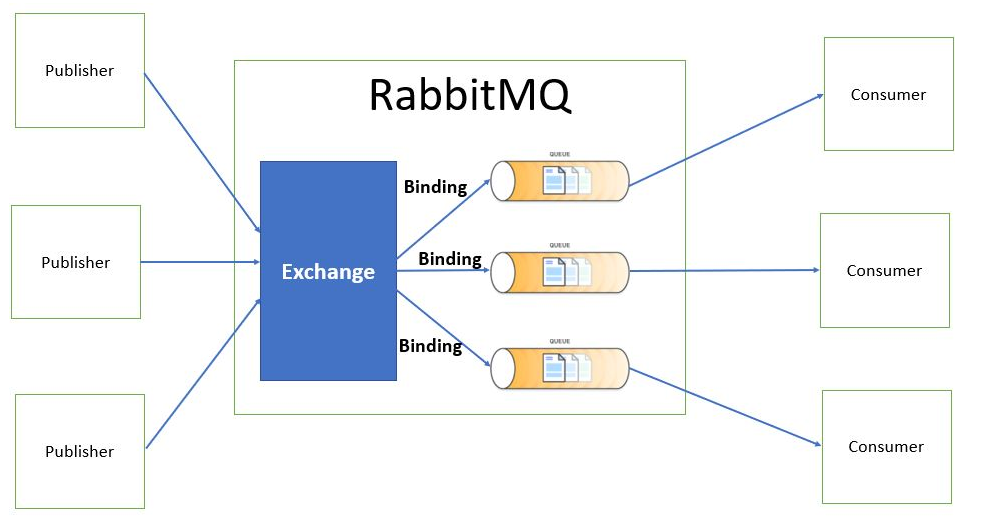
En çok örneği verilen mail gönderim işlemidir. Bir kullanıcıya birden fazla mail gönderiminde bulunacaksınız ve bu mail gönderimi zaman alacağından kullanıcıyı bekletmek doğru olmaz. Bir kuyrukta tutarak gönderme işlemlerini güvenli bir şekilde gerçekleştirebilirsiniz. Bir diğer benim örneğim ise şöyledir; bir ofiste iki bilgisayar (publishers) var ve bir adet yazıcı (consumer) var. İki bilgisayar kullanıcısı birden fazla pdf dosyasını yazdırmak için yazdır tuşlarına bastılar aynı anda veya değil. İlk isteği gönderinin çıktılarını almaya yazıcı başladığını varsayalım. 1000 sayfalık bir çıktı alınırken kullanıcı diğer bilgisayarı kullanan kullanıcı bir çıktı almak istediğinde 1000 sayfanın bitmesini beklemesi ne kadar doğru olur. İşte burada Kuyruk(Que) mantığı devreye giriyor. Kullanıcılar her yazdır butonuna bastığında yazdırma işlemleri beklemeden kuyruğa (yazıcı için yazdırma kuyruğu ) dosyalarını atmış oluyorlar ve yazcının bu kuyruktan dosyaları çekerek yazdırmalarını bekliyorlar. Bu süre zarfında da işlemlerine devam ediyorlar.

**Nerelerde Kullanılabilir?**

* Yoğun E-Mail Gönderimlerinde
* Yoğun veri işlemesinin bulunduğu alanlarda
* Verilerin önemli olduğu yerlerde
* Raporlama sunucularında
* Sensörlerden veri toplanan yerlerde

**Yaşam Döngüsü**

N tane Publisher (İstekte bulunan) bulunabilir ve N tane Consumer (Kuruktaki veriyi işleyen) bulunabilir.  
  
Publisher ve Consumer olarak tanımladığımız yapılar birbirinden farklı programlama diliyle yazılmış programlar olabilir. Önemli olan RabbitMQ desteğinin olması. Ortak dilleri RabbitMQ oluyor.

****

**RabbitMQ Yapısı Nedir?**

Exhange ve Que lardan oluşur.Exchange, Publisherdan gelen istekleri etiketlere göre böler ve etiket olarak Que larda turar.

**Exchange Type**

1. **Direct Exchange**

* **Tanım:** Mesajların belirli bir routing key (yönlendirme anahtarı) ile yönlendirildiği exchange türüdür. Direct exchange, gönderilen mesajın routing key'i ile kuyruğun bağlandığı routing key eşleşirse mesaj o kuyruklara yönlendirilir.
* **Kullanım Durumu:** Belirli bir tüketiciye mesaj göndermek istediğinizde kullanılır. Örneğin, belirli bir hizmete veya işleme özel mesajlar göndermek istediğinizde.
* **Örnek:**
  + Exchange: direct\_logs
  + Routing Key: error
  + Queue: error\_queue

Bu durumda, sadece error routing key'ine sahip mesajlar error\_queue'ya yönlendirilir.

**2. Fanout Exchange**

* **Tanım:** Fanout exchange, gelen tüm mesajları bağlı olduğu tüm kuyruklara yönlendirir, routing key'i dikkate almaz.
* **Kullanım Durumu:** Bir mesajın birden fazla tüketiciye iletilmesi gereken durumlarda kullanılır. Yayın tipi mesajlaşma senaryoları için idealdir.
* **Örnek:**
  + Exchange: logs
  + Queue: log\_queue\_1, log\_queue\_2

Bu durumda, logs exchange'ine gelen her mesaj log\_queue\_1 ve log\_queue\_2 kuyruklarına yönlendirilir.

**3. Topic Exchange**

* **Tanım:** Topic exchange, routing key'e dayalı olarak mesajları yönlendirir ancak burada routing key'in bir şablon (pattern) ile eşleşmesi gerekir. Routing key noktalarla ayrılmış kelimelerden oluşur (örn. stock.update). Kuyruklar bu şablonlara göre exchange'e bağlanır.
* **Kullanım Durumu:** Mesajların belirli bir desenle eşleşen anahtarlara yönlendirilmesi gerektiğinde kullanılır. Bu, çok yönlü mesaj yönlendirme için kullanışlıdır.
* **Örnek:**
  + Exchange: topic\_logs
  + Routing Key: \*.error, stock.#
  + Queue: error\_queue, stock\_queue

Bu durumda, \*.error pattern'ine sahip tüm routing key'ler error\_queue'ya, stock. ile başlayan tüm routing key'ler stock\_queue'ya yönlendirilir.

**4. Headers Exchange**

* **Tanım:** Headers exchange, routing key yerine mesaj başlıklarına (headers) dayanarak mesajları yönlendirir. Mesaj başlıkları, belirli bir koşul ile eşleştirildikten sonra ilgili kuyruğa yönlendirilir.
* **Kullanım Durumu:** Routing key yerine mesaj başlıklarına göre yönlendirme yapmak istediğinizde kullanılır. Esneklik sağlar, ancak performans açısından daha ağır olabilir.
* **Örnek:**
  + Exchange: headers\_logs
  + Headers: format=pdf ve x-match=all
  + Queue: pdf\_queue

Bu durumda, format başlığı pdf olan ve x-match=all koşulunu sağlayan mesajlar pdf\_queue'ya yönlendirilir.

**Sonuç**

* **Direct**: Tam eşleşme gerektiren basit yönlendirme.
* **Fanout**: Yayımlama, her kuyruk için bir mesaj.
* **Topic**: Desen eşleşmesine dayalı esnek yönlendirme.
* **Headers**: Başlıklara göre koşul bazlı yönlendirme.

Bu exchange türleri, RabbitMQ'nun esnek ve güçlü bir mesaj yönlendirme altyapısı sunmasını sağlar. Hangi türün kullanılacağı, mesajlaşma gereksinimlerinize ve sistem mimarinize bağlıdır.

RabbitMQ'daki **Topic Exchange** türünde kullanılan \* (yıldız) ve # (kare) sembolleri, mesajların routing key'lerine (yönlendirme anahtarlarına) göre kuyruklara nasıl yönlendirileceğini belirler. Bu iki sembol, routing key'lerin farklı desenlerle eşleşmesini sağlar.

**\* (Yıldız)**

* **Anlamı:** Yıldız, routing key'deki **bir tek kelimeyi** temsil eder. Yani, bir yıldız, routing key'deki bir kelimenin yerini alabilir. Her bir kelime, routing key'deki nokta (.) ile ayrılmış segmentlerden oluşur.
* **Kullanım Durumu:** Belirli bir segmentte herhangi bir kelimenin kabul edilebileceği durumlarda kullanılır.
* **Örnek:**
  + Routing Key: stock.update
  + Pattern: stock.\*

Bu durumda, stock.\* pattern'i stock.update, stock.delete, stock.create gibi iki segmentli routing key'lerle eşleşir, ancak stock.price.update gibi üç segmentli bir key ile eşleşmez.

**# (Kare)**

* **Anlamı:** Kare, routing key'deki **sıfır veya daha fazla kelimeyi** temsil eder. Yani, kare bir veya daha fazla segmentin yerini alabilir, hatta hiç segment olmamasını bile kabul eder.
* **Kullanım Durumu:** Routing key'in daha genel bir şekilde eşleştirilmesi gerektiğinde kullanılır.
* **Örnek:**
  + Routing Key: stock.update
  + Pattern: stock.#

Bu durumda, stock.# pattern'i stock.update, stock.price.update, stock, stock.price.daily.update gibi stock. ile başlayan tüm routing key'lerle eşleşir.

**Kısaca:**

* \*: Sadece **bir kelime** ile eşleşir.
* #: **Sıfır veya daha fazla kelime** ile eşleşir.

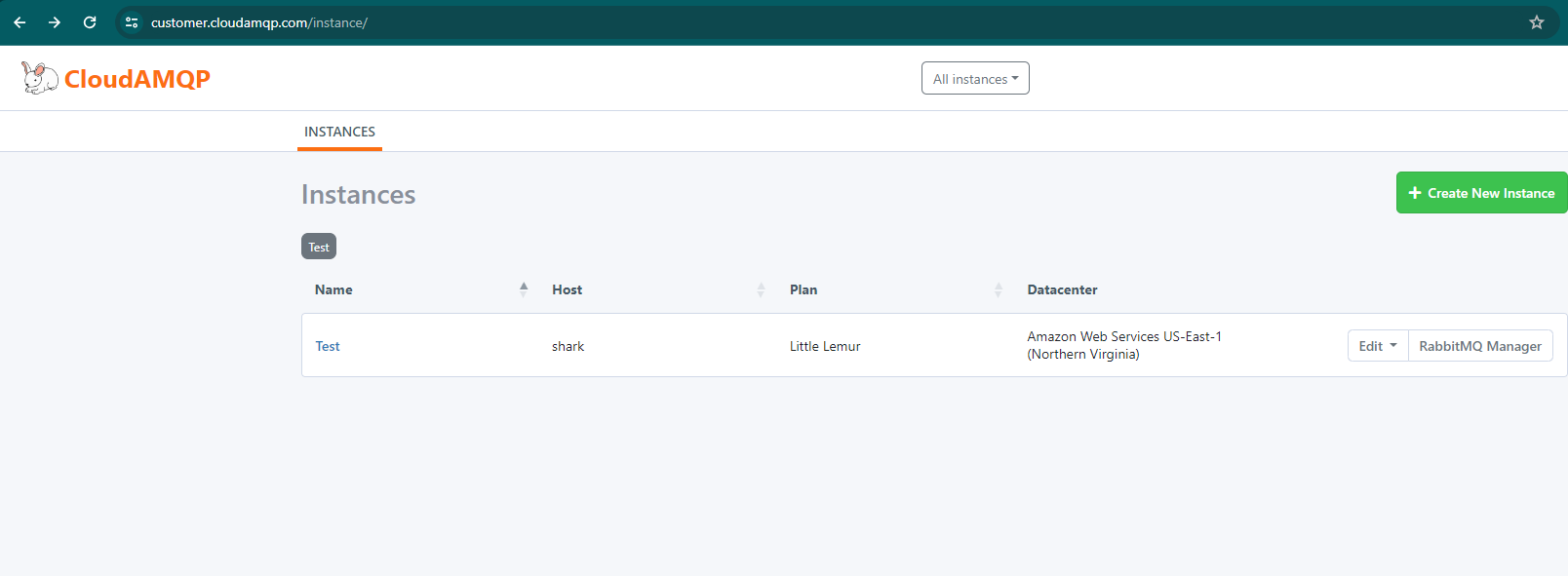
Bu sayede \* ve # sembolleri, RabbitMQ'da daha esnek ve güçlü bir mesaj yönlendirme mekanizması oluşturmanızı sağlar.

**Hadi Bazı Testler Yapalım**

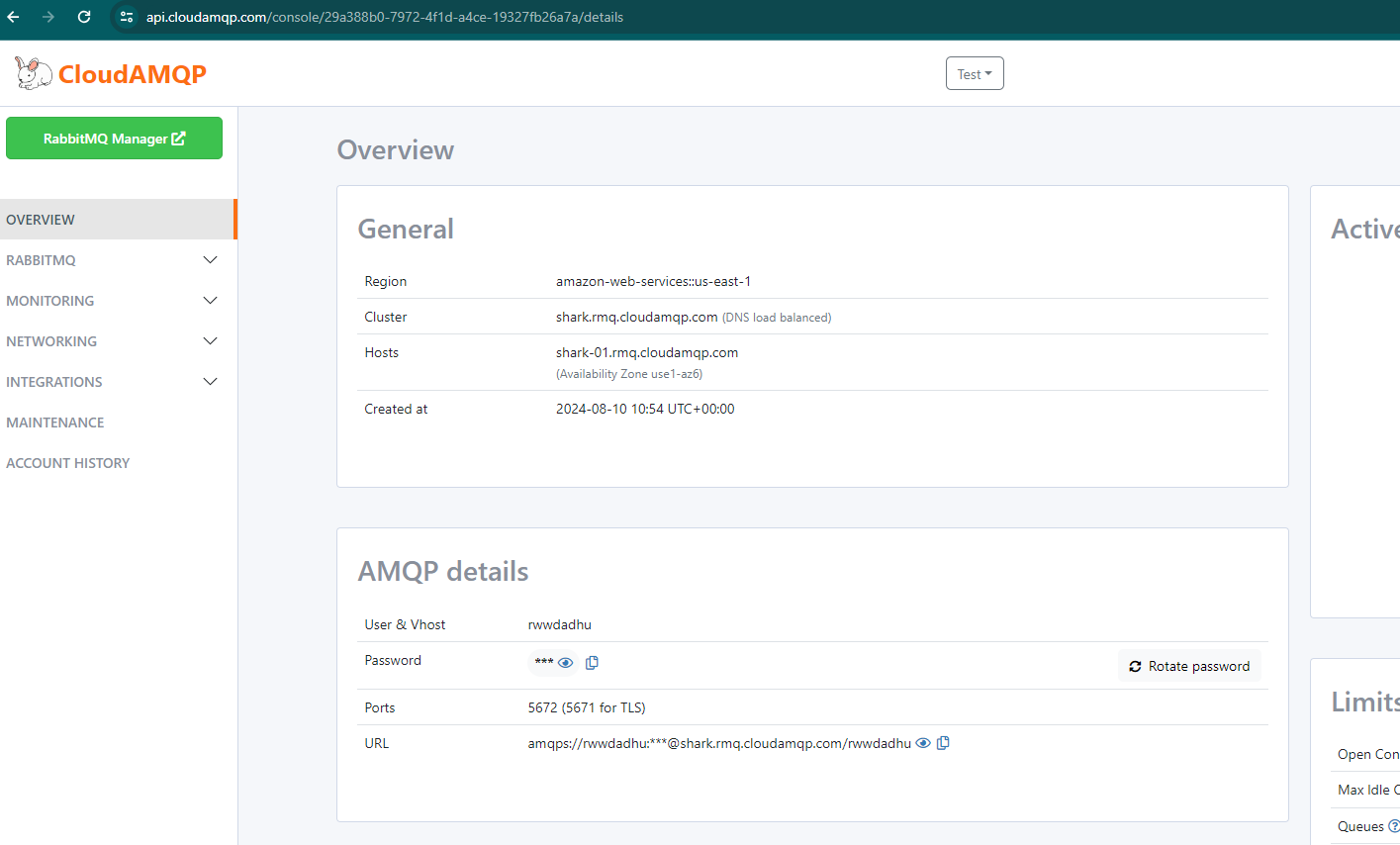
RabbitMQ nun ücretsiz cloud denemesini kullandım. Sizde bilgisayarına kurmak istemiyorsanız. Aşağıdaki linkten deneme hesabı açabilirsiniz.

**Ücretsiz Deneme (For Cloud)**

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

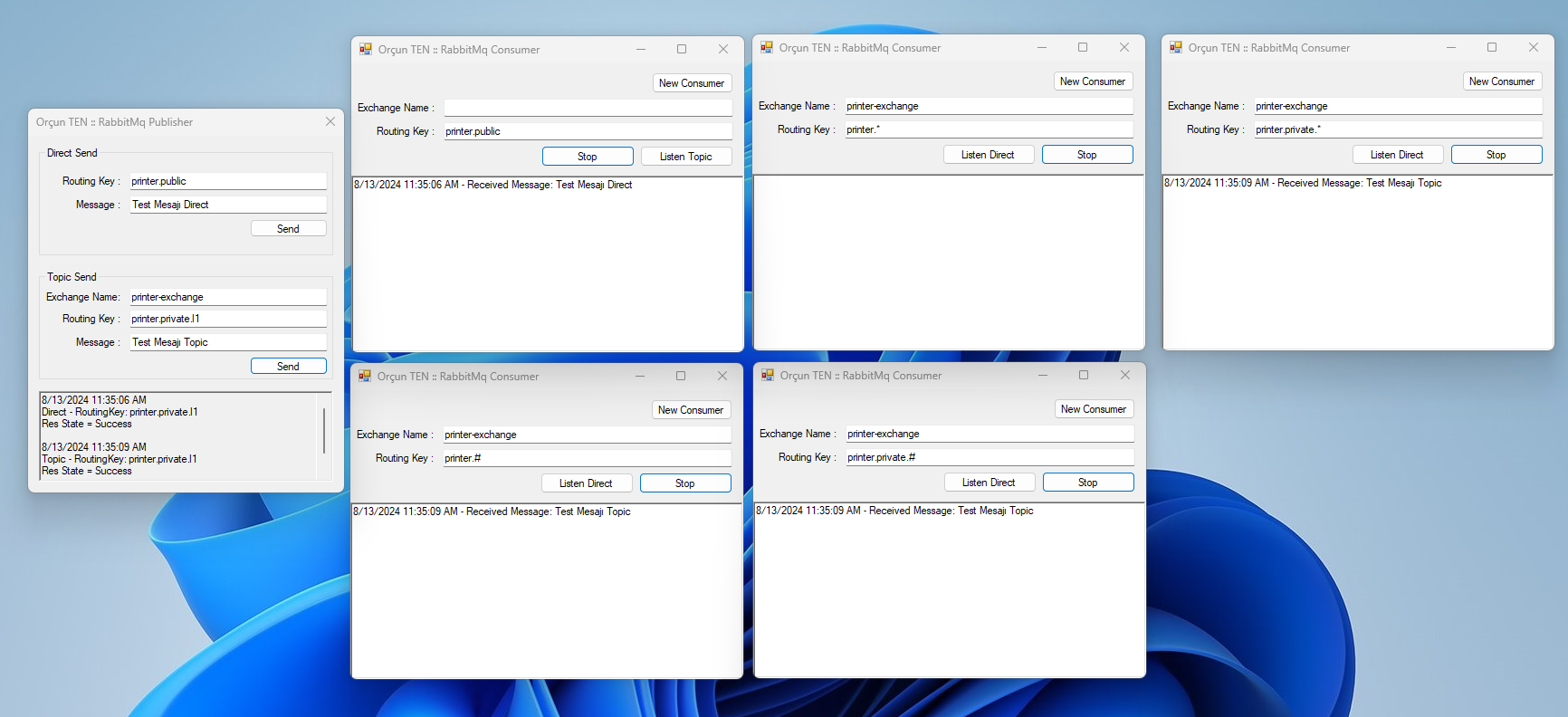


Yukarı daki resimde Gösterildiği gibi bir Test Instance’I oluşturdum va ödeme planını Little Lamur seçtim (Free). Test linkine tıkladığınızda bağlantı bilgilerini ve diğer birçok ayarları (Monitoring, Network Durumu, Loglar vb.) gözlemleyebilirsiniz.



**.Net Nuget Paketleri**

Nuget Package : RabbitMQ.Client.6.8.1



Yukarıda bir direct ve bir topic örneğini görmüş oluyoruz. Direct de birebir RoutingKey girilmelidir. Topic de ise Escape karakterler ile gönderilen bir mesaj birden fazla kuyruğa girmektedir.

**Git Repo**

https://github.com/oten52/RabbitMQBlog/tree/main