

# File de messages

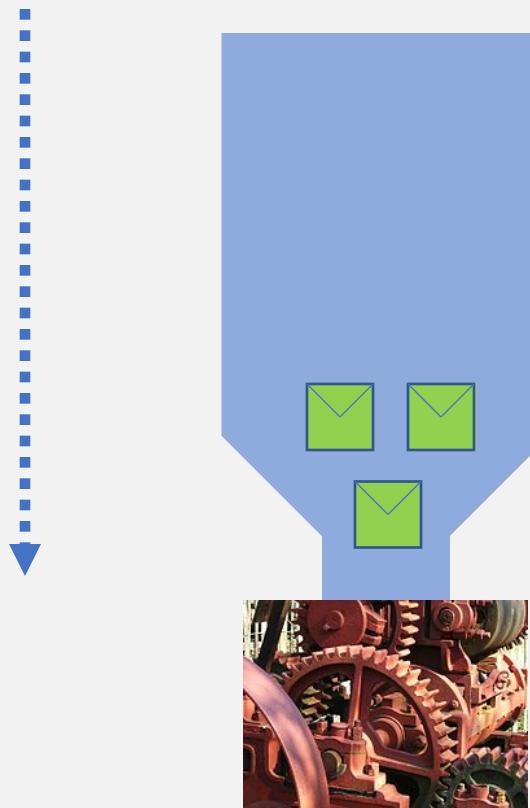


# Objectifs

- Problème du mode requête / réponse
- File de messages dans les échanges de données
- Exemple d'implantation en C# avec RabbitMQ

# Problème du modèle requête / réponse

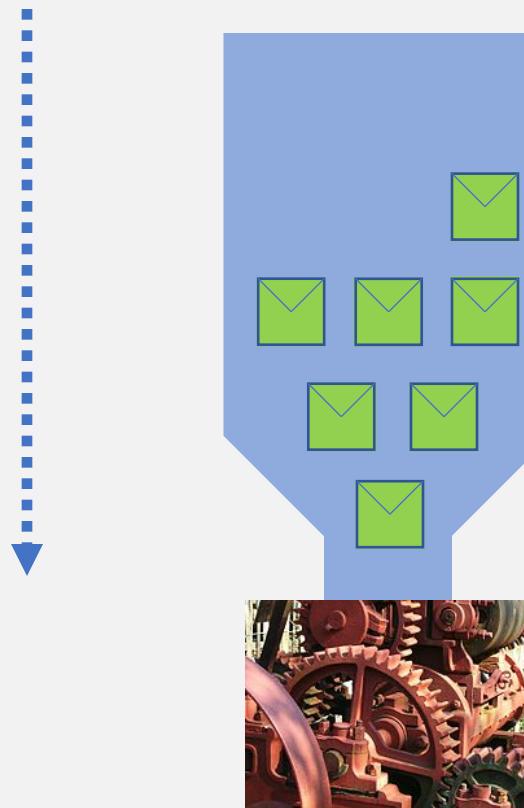
- Dans le modèle requête / réponse (request / reply), le service ne maîtrise pas le nombre de requêtes qu'il reçoit. Il peut être débordé et pourrait saturer !



Les messages sont  
**poussés** sans que le  
serveur ait le  
contrôle.

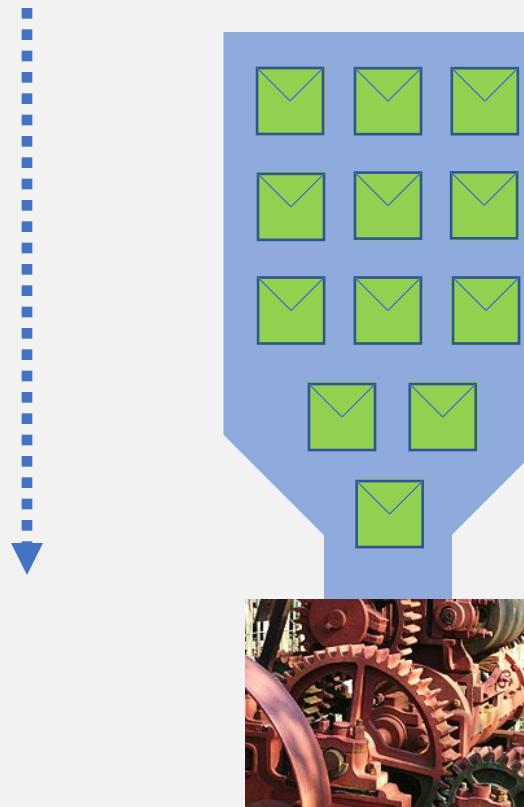
# Problème du modèle requête / réponse

- Dans le modèle requête / réponse (request / reply), le service ne maîtrise pas le nombre de requêtes qu'il reçoit. Il peut être débordé et pourrait saturer !



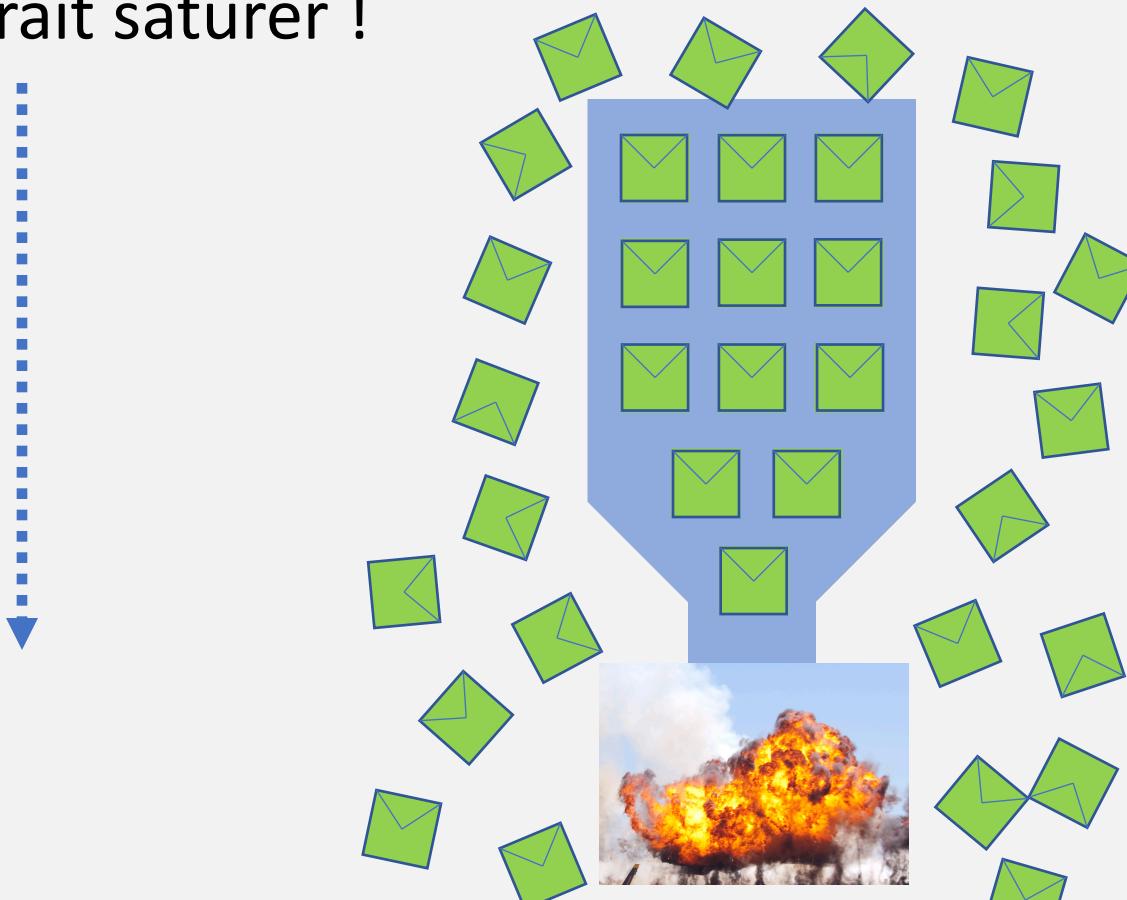
# Problème du modèle requête / réponse

- Dans le modèle requête / réponse (request / reply), le service ne maîtrise pas le nombre de requêtes qu'il reçoit. Il peut être débordé et pourrait saturer !



# Problème du modèle requête / réponse

- Dans le modèle requête / réponse (request / reply), le service ne maîtrise pas le nombre de requêtes qu'il reçoit. Il peut être débordé et pourrait saturer !

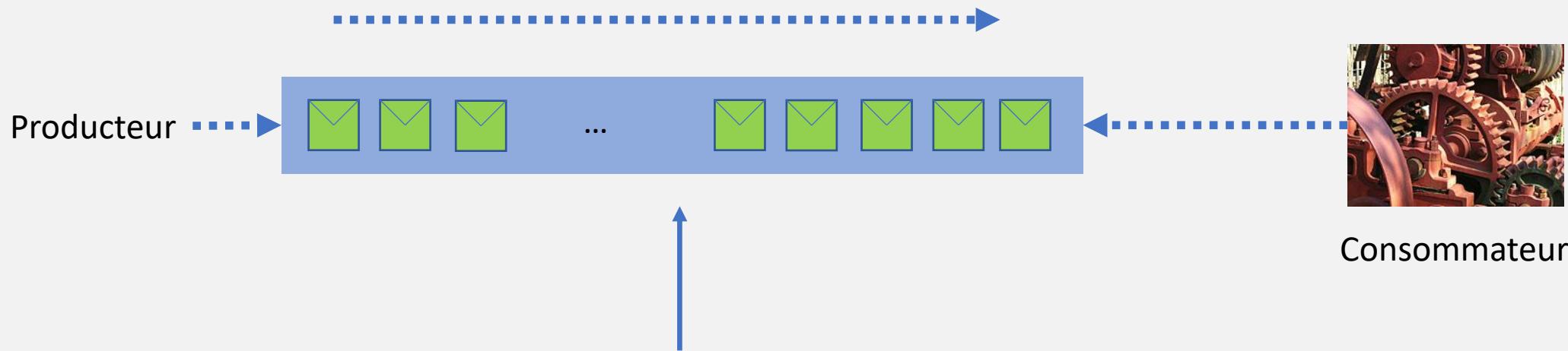


# Problème du modèle requête / réponse – Solutions ?

- Créer plusieurs instances du service
  - Mais combien ?
  - Comment être élastique ? (Création au besoin et libération : Kubernetes ?)
- Passer dans un mode envoyer et oublier (fire and forget)
  - Idée :
    - Utiliser un intermédiaire qui reçoit le message
    - Le message est traité quand les instances du services ont le temps : le message est **tiré** contrairement au mode **poussé** du **synchrone**
  - Exemples :
    - Traitement des vidéos dans YouTube, Facebook, etc.
    - Compression d'un répertoire dans Google Drive, One Drive, etc.
    - Transactions financières

# Modèle asynchrone

Les messages sont **tirés** le serveur a le contrôle



**Courtier de messages** : agit comme une boîte aux lettres, un bureau de poste et un facteur

# Rabbit MQ

- Rabbit MQ est un courtier de messages (Broker)
- Un producteur envoie des messages et le consommateur va le chercher
- Il permet d'utiliser d'autres modèles que nous verrons dans d'autres modules
- Pour le producteur et consommateur, il faut ajouter le package Nuget : RabbitMQ.Client

# RabbitMQ – Producteur

```
ConnectionFactory factory = new ConnectionFactory() { HostName = "localhost" };
using (IConnection connexion = factory.CreateConnection())
{
    using (IModel channel = connexion.CreateModel())
    {
        channel.QueueDeclare(queue: "hello",
            durable: false,
            exclusive: false,
            autoDelete: false,
            arguments: null
        );

        for (int i = 0; i < 10; i++)
        {
            string message = "mon message - " + Guid.NewGuid();
            byte[] body = Encoding.UTF8.GetBytes(message);

            channel.BasicPublish(exchange: "", routingKey: "hello", body: body);
        }
    }
}
```

# RabbitMQ – Consommateur

```
private static ManualResetEvent waitHandle = new ManualResetEvent(false);
[...]
ConnectionFactory factory = new ConnectionFactory() { HostName = "localhost" };
using (IConnection connexion = factory.CreateConnection())
{
    using (IModel channel = connexion.CreateModel())
    {
        channel.QueueDeclare(queue: "hello", durable: false, exclusive: false,
                             autoDelete: false, arguments: null
        );

        EventingBasicConsumer consommateur = new EventingBasicConsumer(channel);
        consommateur.Received += (model, ea) =>
        {
            byte[] donnees = ea.Body.ToArray();
            string message = Encoding.UTF8.GetString(donnees);
            Console.Out.WriteLine(message);
            channel.BasicAck(ea.DeliveryTag, false);
        };
        channel.BasicConsume(queue: "hello",
                             autoAck: false,
                             consumer: consommateur
        );
        waitHandle.WaitOne();
    }
}
```

# Références

- Documentation RabbitMQ : <https://www.rabbitmq.com>
- Image RabbitMQ : [https://hub.docker.com/\\_/rabbitmq](https://hub.docker.com/_/rabbitmq)