Asynchronous communication

(Demo prep)

- Delete the RabbitMQ image and volume
- Delete the antidote from DB

Sending messages to RabbitMQ via Docker

It's time to stand up our RabbitMQ message broker, and of course we will use docker for that.

1. Open docker-compose.yml

2. **Collapse the navigation pane**

3. Add RabbitMQ to docker compose file:

```
 rabbitmq:
   image: rabbitmq:management
   container_name: rabbitmq
   ports:
     - 5672:5672
     - 15672:15672
   volumes:
     - rabbitmqdata:/var/lib/rabbitmq
   hostname: rabbitmq      # Its important because this is associated to how RabbitMQ stores data and if not specified it gets a random value, and each time we restart the container RabbitMQ would store data in a different place

volumes:
  mongodbdata:
  rabbitmqdata:
```

4. Open Terminal

5. docker-compose up -d

6. Switch to Catalog and **collapse navigation pane**

7. Open Terminal

8. Start Catalog via **dotnet run**

9. Notice the Masstransit info

10. Create an item via Postman:

```
{
   "name": "Antidote",
   "description": "Cures poison",
   "price": 7
}
```

11. Notice debug logs

12. Browse to RabbitMQ portal and show the new **CatalogItemCreated exchange**:

    http://localhost:15672

At this point we are able to publish messages from our Catalog service any time our items database is updated. In the next lesson we will generalize some of the code we added here into our Common library in preparation for having Inventory start consuming the published messages.