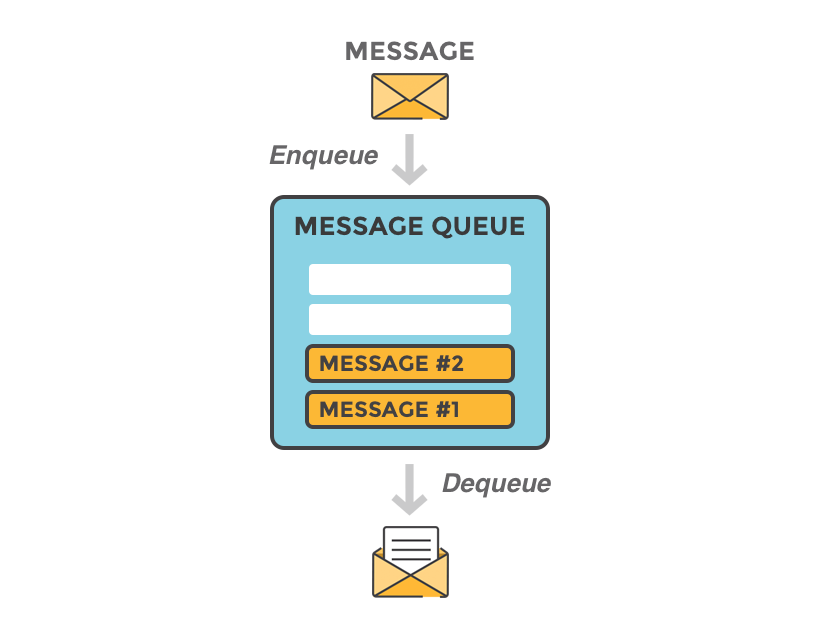
# Integrating RabbitMQ with Dockers:

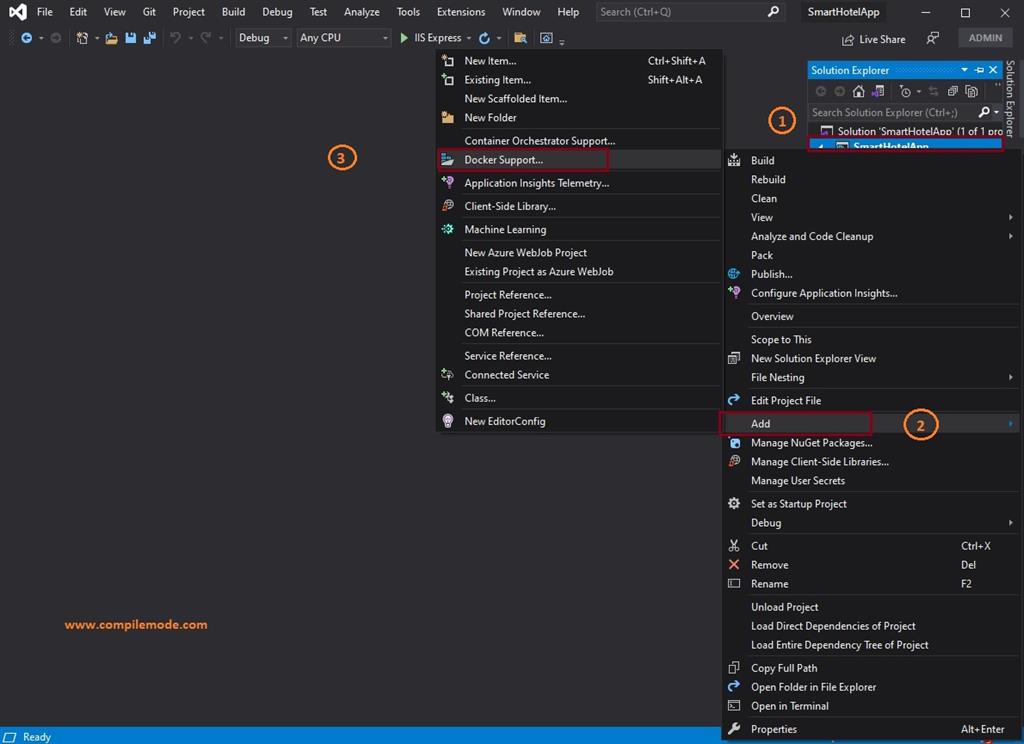
Asynchronous Communication between 2 Dockers uses a third RabbitMQ Docker.

Producer microservices enqueues messages to the Rabbit queue, while Worker/consumer microservice dequeues them.

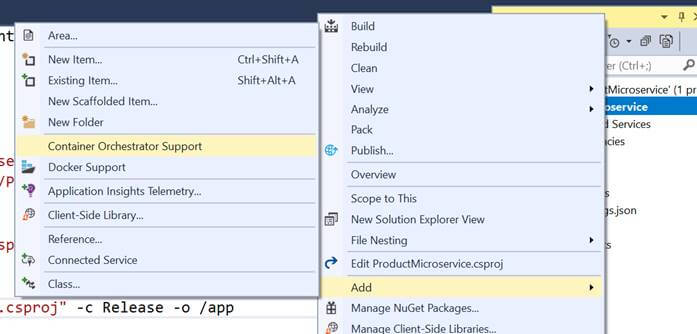
## Adding Rabbit MQ To docker compose Project:

1. **Add Docker Support for each project:**

Right click on the solution explorer of your existing project and follow the steps which are shown in the following image.



1. **Add container orchestrator support for each project:**



1. **Add Rabbit container:**

modify docker-compose.yml by adding RabbitMQ under dervices:

version: '3.4'

services:

productswithrabbit:

image: ${DOCKER\_REGISTRY-}productswithrabbit

build:

context: .

dockerfile: ProductsWithRabbit/Dockerfile

subscriber:

image: ${DOCKER\_REGISTRY-}subscriber

build:

context: .

dockerfile: Subscriber/Dockerfile

depends\_on:

- "productswithrabbit"

- "rabbitmq"

rabbitmq: # login guest:guest

image: rabbitmq:3-management

hostname: "rabbitmq"

labels:

NAME: "rabbitmq"

ports:

- "4369:4369"

- "5671:5671"

- "5672:5672"

- "25672:25672"

- "15671:15671"

- "15672:15672"

What we’ve done above is tell docker-compose to add a new service to our application, named rabbitmq. It pulls an existing image (rabbitmq:3-management) from the public Docker repository.

1. Add RabbitMQ Client Nuget package to each on the projects.
2. On both Producer and receiver change the rabbit factory to create connection to RabbitMQ as follow:

var factory = new ConnectionFactory() { HostName = "rabbitmq", Port = 5672, UserName = "guest", Password = "guest" };