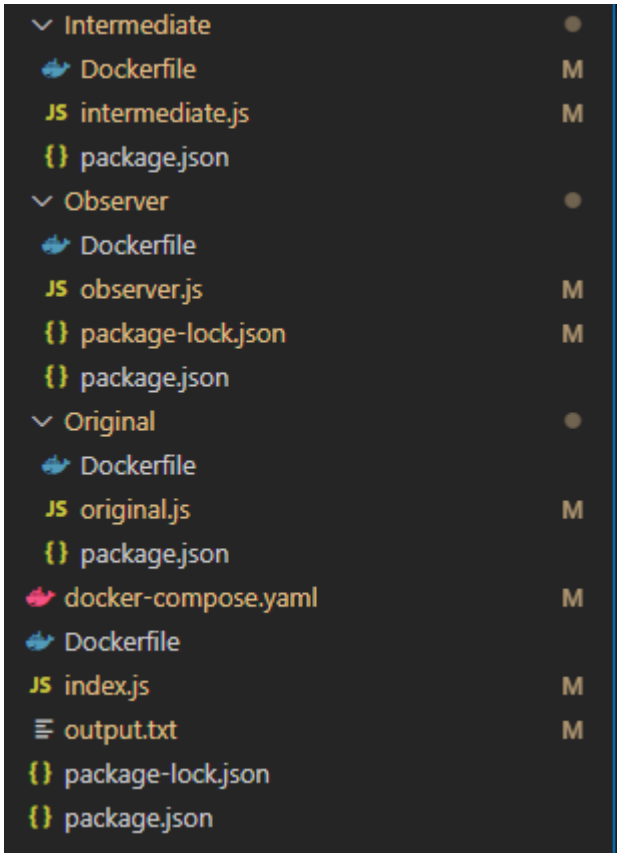
The application contains the following file structure:



original.js will publish three message, intermediate.js will subscribe topic my.o and publish modified message to topic my.i. The observer.js will subscribe all topics with "#" and the http server read the log from a saved file.

It is a typical model of communication on the network of http based request-response. In this model, the client send a request to the server and the server will response to the request.

For example in this application HTTPSERV container, the http library used to create server and listen on port 8080. When user open localhost with port 8080, it will send http request to the server and server will response with the content inside saved file (log of message).

A different way for devices to communicate on the network is using the publish-subsribe or topic-based. In this model, a broker receives and distributes the data, and clients can publish data to the broker or subscribe to the broker in order to get the data form it.

The client that publishes the data only when data changed, and client will altomatically receive changed data from broker. Here the broker doesnt store the data, it just move from the publisher to the subscriber. When the data get from the publisher, it will immediately forward to any client that subscribes to the data.

The benefits of topic-based communication in my mind is that this model can decoupling many subsystems and each subsystem can be managed independently. Even if some subsystem go offline, it will not affect the overall system messages. It also provides separate concerns for the application. like here how we used

*, which represent a word and #, which represent for zero or many words, each application can forcus on its own function or topic.

The topic-based model also improve reliability. The async messaging helps application continue to run smoothly under increased work load and can handle failures more effectively.

For this exercise, the topic model we used I feel more like add support of keys (* and hash #) based on routing mode in order to meet more complex message distrubution scenarios.