Submitted By: Anupriya Goyal and Sravanthi Adibhatla
Group members UBIT Name:
1. anupriya
2. sadibhat
UB Person No. of Anupriya : 50287108
UB Person No. of Sravanthi : 50288587
Date: 11/12/2018

# CSE 586 Distributed Systems Project1 Report
# Project2: Emulating Pub/Sub Distributed System Using Docker Containers

## Introduction:

Pub/Sub is asynchronous service-to-service communication. In a pub/sub model, a message which is published to a particular topic is immediately received by all subscribers who have subscribed to that topic. It enables event driven architecture and decouple applications to increase performance, reliability and scalability.
Advantage of Pub/Sub System:

1. Publishers are loosely coupled to subscribers and need not to even know their existence. Topic being in focus, publishers and subscribers are allowed to remain ignorant of system topology.
2. It provides better scalability than client server through parallel operation, message caching, network-based routing, etc.

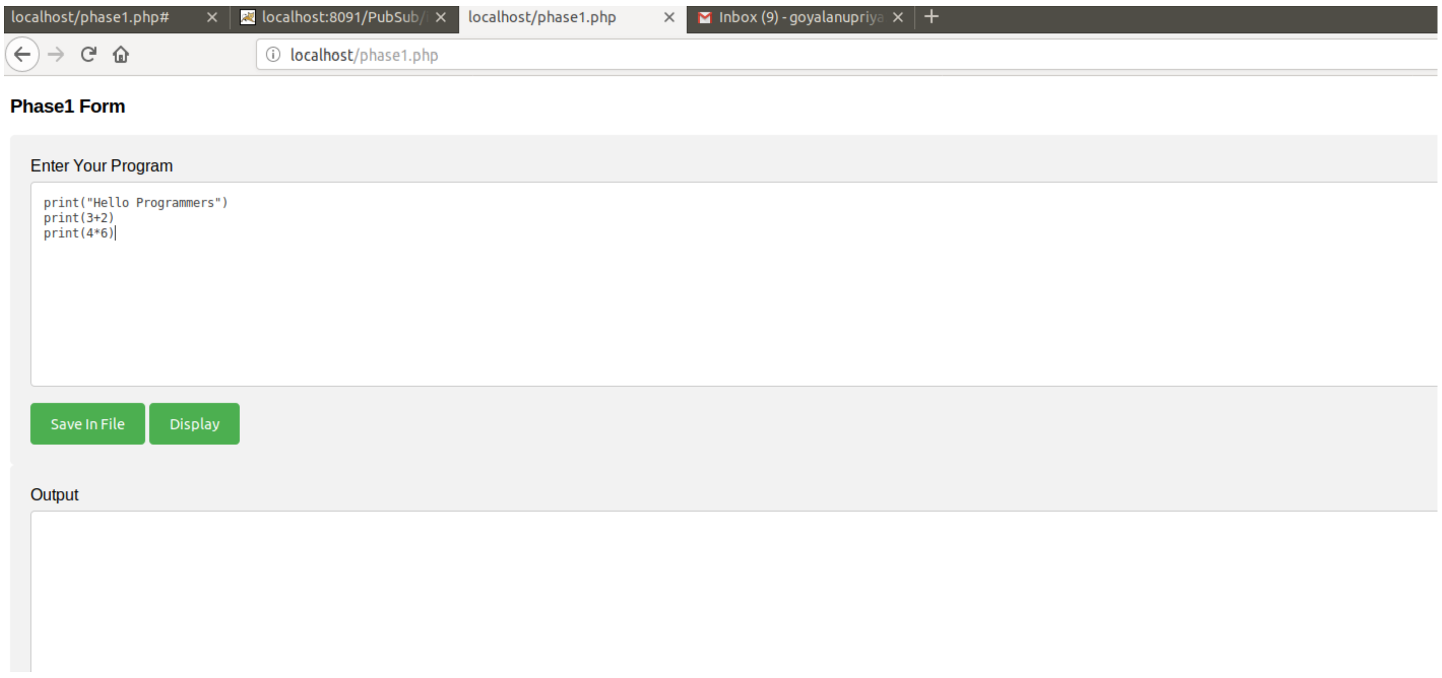
The implementation of Pub/Sub system is done in several phases. These are described as follows:

**Phase1:**
Phase1 consist of installing docker, running a program through it and displaying the execution on web page.
For installing docker in Ubuntu 18.04.1 followed the steps given on the official web page of docker. (URL is added in the references).
Figure1. shows the web page that is developed using technologies php, html and javascript. There is a textbox area where user can input the program of their choice. We have considered language python for the input program in textbox area. Once the user has entered the program in textbox area and clicked on *'Save In File'* button then this program (entered by user) will be saved in python file named as 'my_script.py'.

Since, docker can build images automatically reading instructions from a Dockerfile. A Dockerfile is a text file contains all the commands a user could call on the command line to assemble an image. Therefore, we have created a Dockerfile for python language programs. Along with that there is a shell.sh file which is created to run the docker build image and docker run container command. So that the execution of my_script.py is done through the docker.

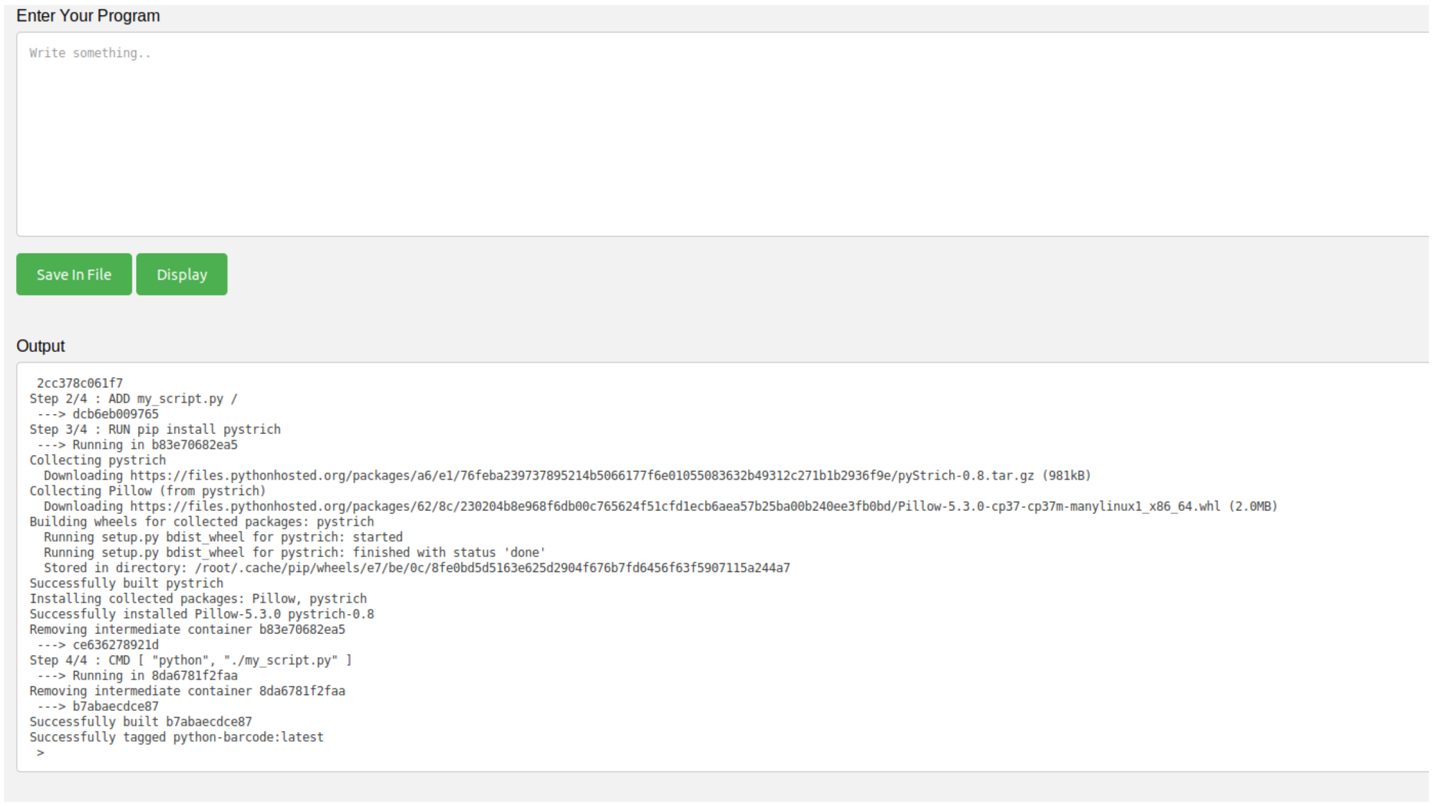
Figure2. shows the output in output box area of the docker execution. It is displayed to the user once the user clicks on '*Display'* button. On clicking Display button shell.sh file will be executed through php script and the output of docker execution is stored in one text file. The result of text file on each execution is displayed to the user in Output textbox area.

In this way the program entered by user is executed through docker.



```
print("Hello Programmers")
print(3+2)
print(4*6)
```

**Figure1. Screenshot of the Web page for phase1 where the user enters the program**



```
 2cc378c061f7
Step 2/4 : ADD my_script.py /
 ---> dcb6eb009765
Step 3/4 : RUN pip install pystrich
 ---> Running in b83e70682ea5
Collecting pystrich
  Downloading https://files.pythonhosted.org/packages/a6/e1/76feba239737895214b5066177f6e01055083632b49312c271b1b2936f9e/pyStrich-0.8.tar.gz (981kB)
Collecting Pillow (from pystrich)
  Downloading https://files.pythonhosted.org/packages/62/8c/230204b8e968f6db00c765624f51cfd1ecb6aea57b25ba00b240ee3fb0bd/Pillow-5.3.0-cp37-cp37m-manylinux1_x86_64.whl (2.0MB)
Building wheels for collected packages: pystrich
  Running setup.py bdist_wheel for pystrich: started
  Running setup.py bdist_wheel for pystrich: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/e7/be/0c/8fe0bd5d5163e625d2904f676b7fd6456f63f5907115a244a7
Successfully built pystrich
Installing collected packages: Pillow, pystrich
Successfully installed Pillow-5.3.0 pystrich-0.8
Removing intermediate container b83e70682ea5
 ---> ce636278921d
Step 4/4 : CMD [ "python", "./my_script.py" ]
 ---> Running in 8da6781f2faa
Removing intermediate container 8da6781f2faa
 ---> b7abaecdce87
Successfully built b7abaecdce87
Successfully tagged python-barcode:latest
 >
```

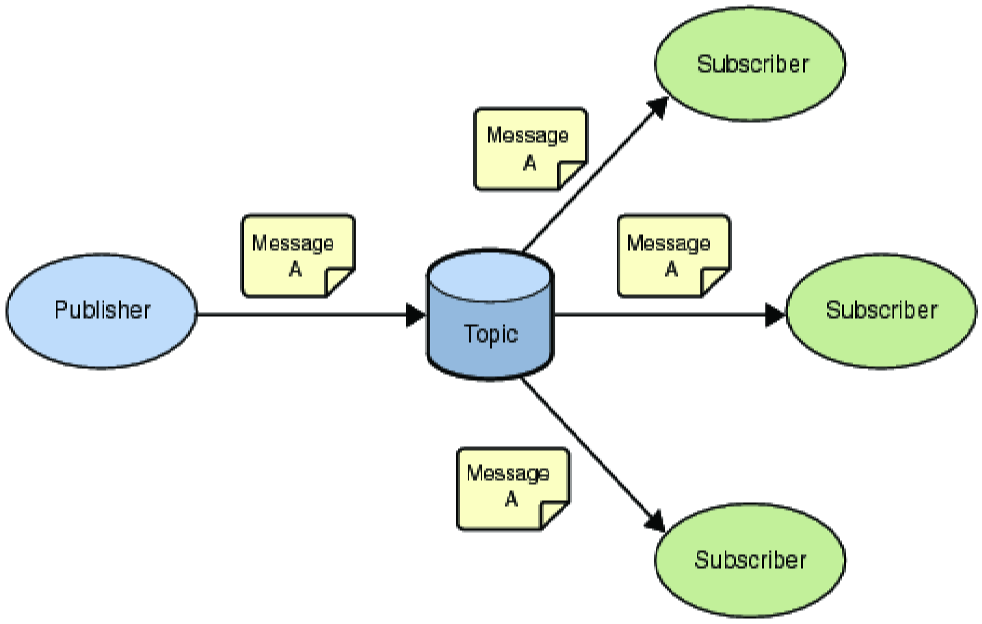
**<u>Figure2. Screenshot of the ouput box displayed after user clicks on Display button</u>**

## Phase 2

In phase 2, we are implementing pub-sub centralized model and deploying that application through Docker. A centralized model means it has only one Docker container which is used to communicate between publisher and subscriber.

In this phase, we create publisher and subscriber and establish communication between them. We are implementing topic- based pub/sub model.

- Publisher has the functionality to publish new content with respect to any topic.
- Subscriber has the functionality to subscribe to receive any info regarding the topic he/she has subscribed to.
- Notify is a functionality that notifies the subscriber whenever publisher publishes any data for the subscribed topic.



In our design implementation, we have a publisher box, subscriber box and a conversation box. We can write in any content in text and publish it under any topic.

Every publish/subscribe operation is displayed in conversation box. Whenever there are any publish content events, either it is notified globally or notifies to the subscribers of that topic.

Technologies used to develop phase 2 are: J2EE, Java and Docker

After the application development. We are using Docker to deploy the application.

We create Dockerfile in which we mention the commands to deploy the **.war file** of the application.

```
FROM tomcat:8.5.11-jre8

COPY /PubSub.war /usr/local/tomcat/webapps/PubSub.war
```

Once, we run the following commands we can access the application through localhost.

```
docker build -t pubsub.
docker run -it --rm -p 8091:8080 pubsub
```

Below are the screen shots which will help in understanding the implementation in a better way.

## Results:

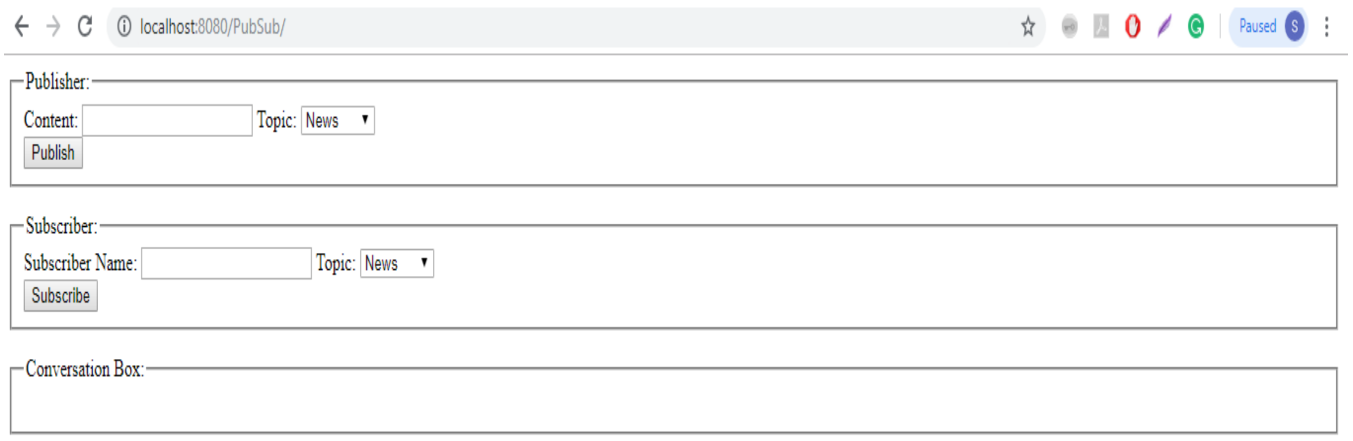
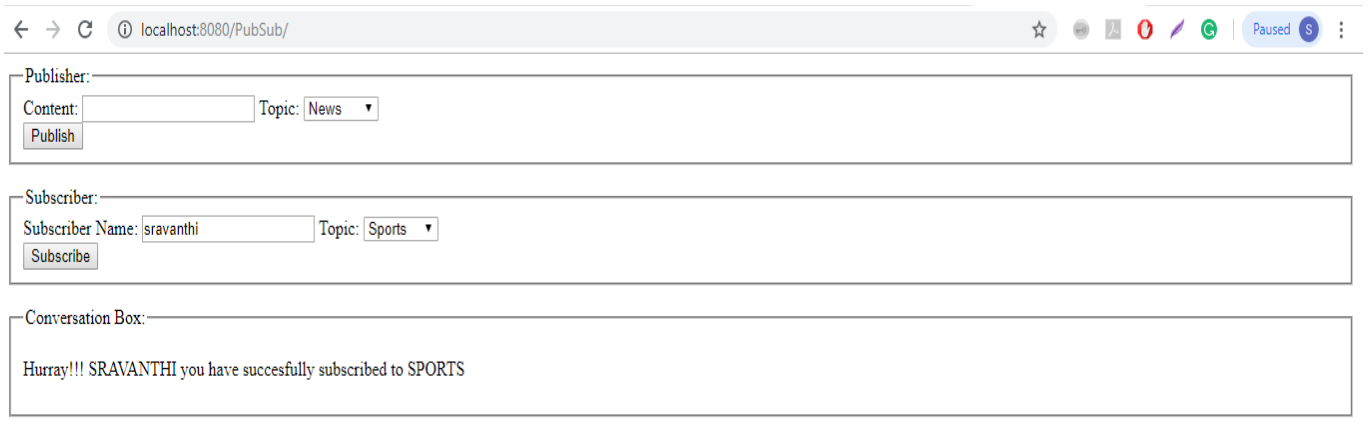


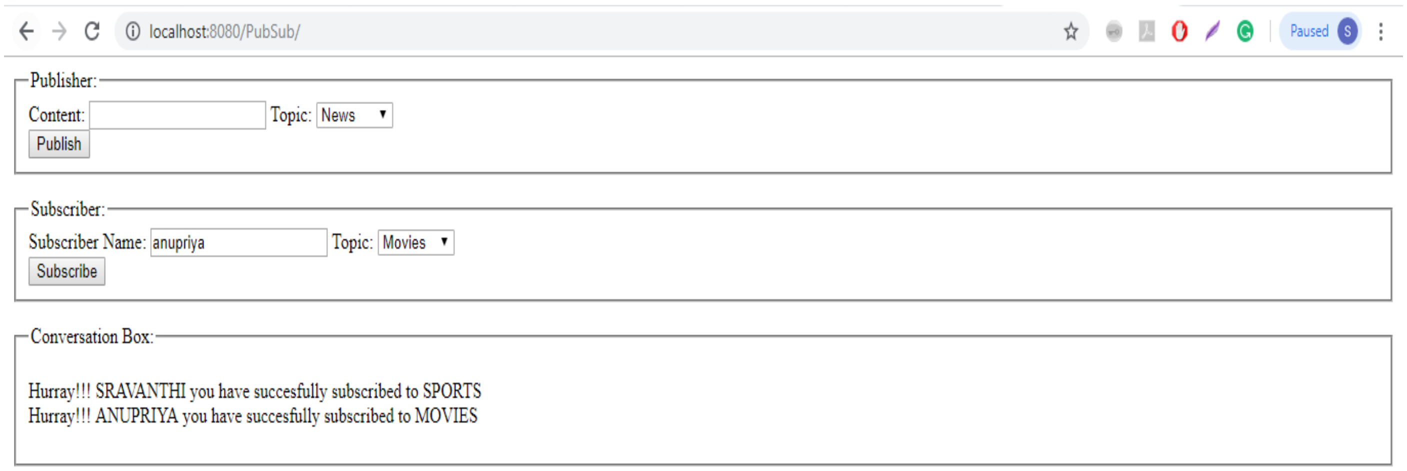
**Figure3. Home screen of Phase 2**



Subscriber Name: sravanthi Topic: Sports

Hurray!!! SRAVANTHI you have succesfully subscribed to SPORTS

**Figure4. Add Subscriber screenshot of Phase 2**



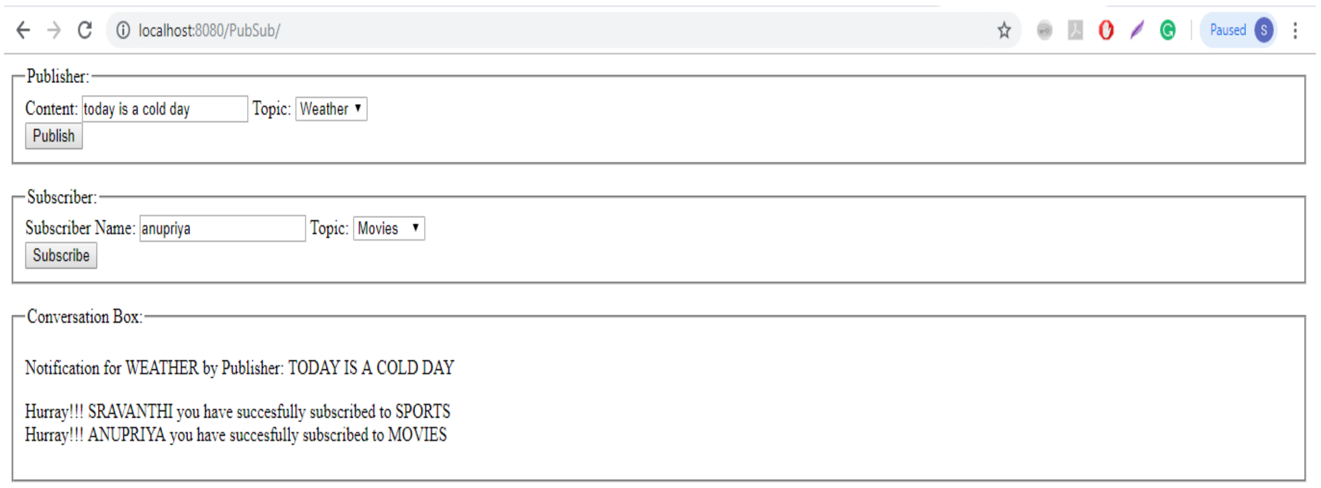
Publisher:
Content: Topic: News

Subscriber:
Subscriber Name: anupriya Topic: Movies

Conversation Box:

```
Hurray!!! SRAVANTHI you have succesfully subscribed to SPORTS
Hurray!!! ANUPRIYA you have succesfully subscribed to MOVIES
```

**Figure 5. Multiple subscriber screenshot for Phase 2**



Publisher:
Content: today is a cold day Topic: Weather

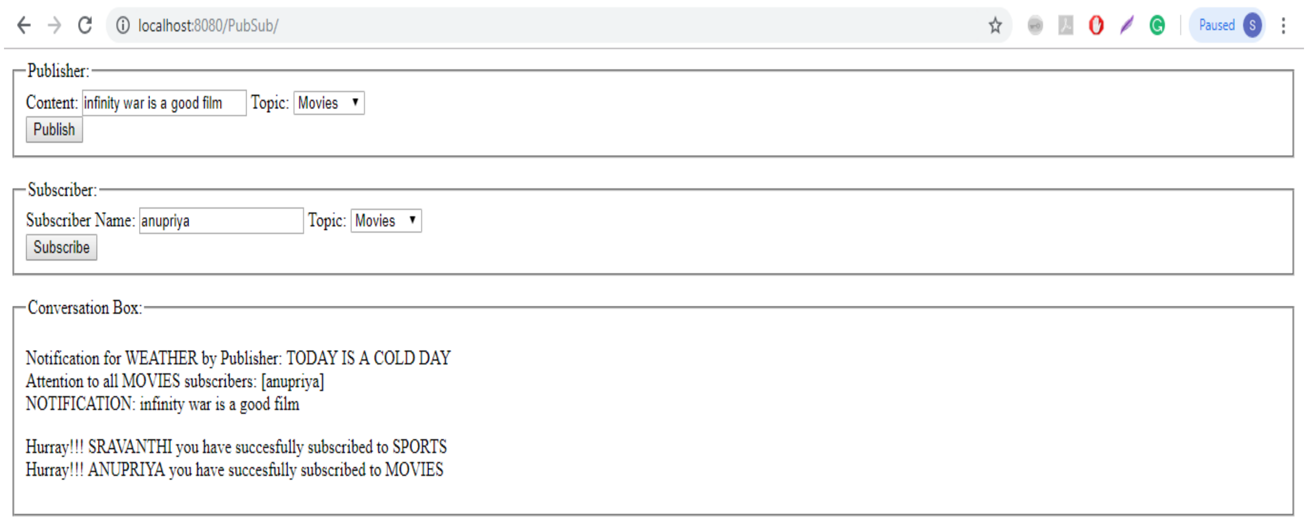
Subscriber:
Subscriber Name: anupriya Topic: Movies

Conversation Box:

```
Notification for WEATHER by Publisher: TODAY IS A COLD DAY

Hurray!!! SRAVANTHI you have succesfully subscribed to SPORTS
Hurray!!! ANUPRIYA you have succesfully subscribed to MOVIES
```

**Figure 6. Publish Content if  no subscribers screen shot for Phase 2**

Since, there are no subscribers for topic WEATHER, only the content is shown.



Publisher:

Content: infinity war is a good film Topic: Movies

Publish

Subscriber:

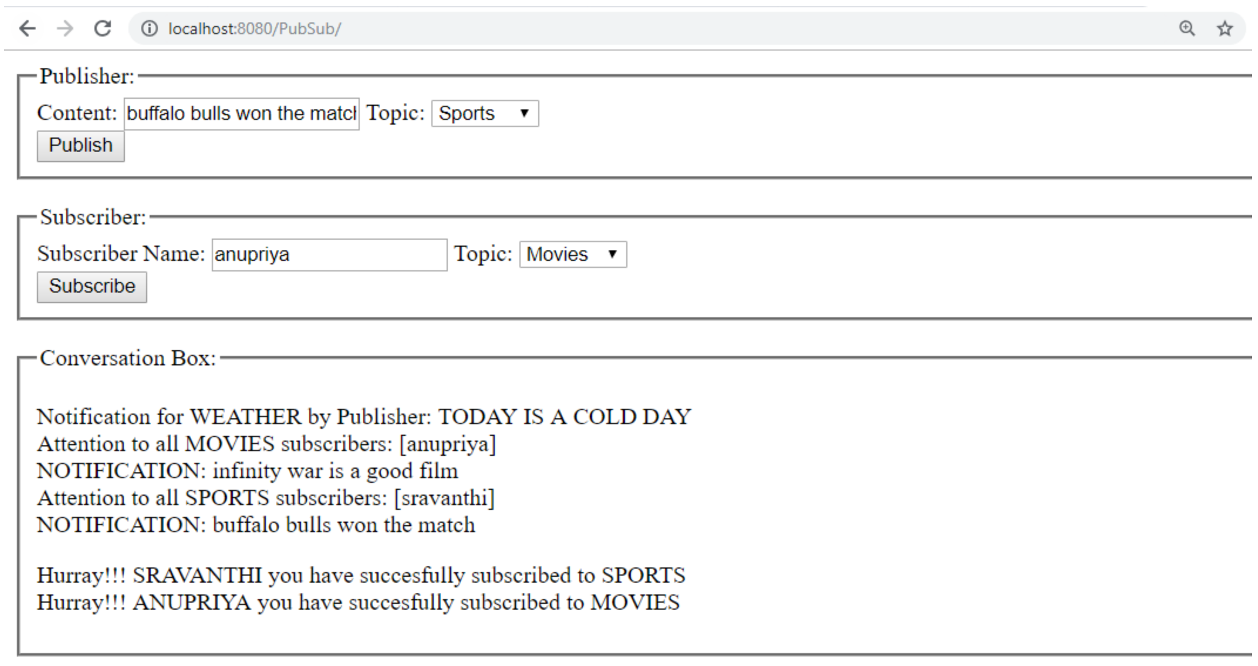
Subscriber Name: anupriya Topic: Movies

Subscribe

Conversation Box:

```
Notification for WEATHER by Publisher: TODAY IS A COLD DAY
Attention to all MOVIES subscribers: [anupriya]
NOTIFICATION: infinity war is a good film

Hurray!!! SRAVANTHI you have succesfully subscribed to SPORTS
Hurray!!! ANUPRIYA you have succesfully subscribed to MOVIES
```

**Figure 7.Notifying to subscriber screen shot for Phase2**



Publisher:

Content: buffalo bulls won the matcl Topic: Sports

Publish

Subscriber:

Subscriber Name: anupriya Topic: Movies

Subscribe

Conversation Box:

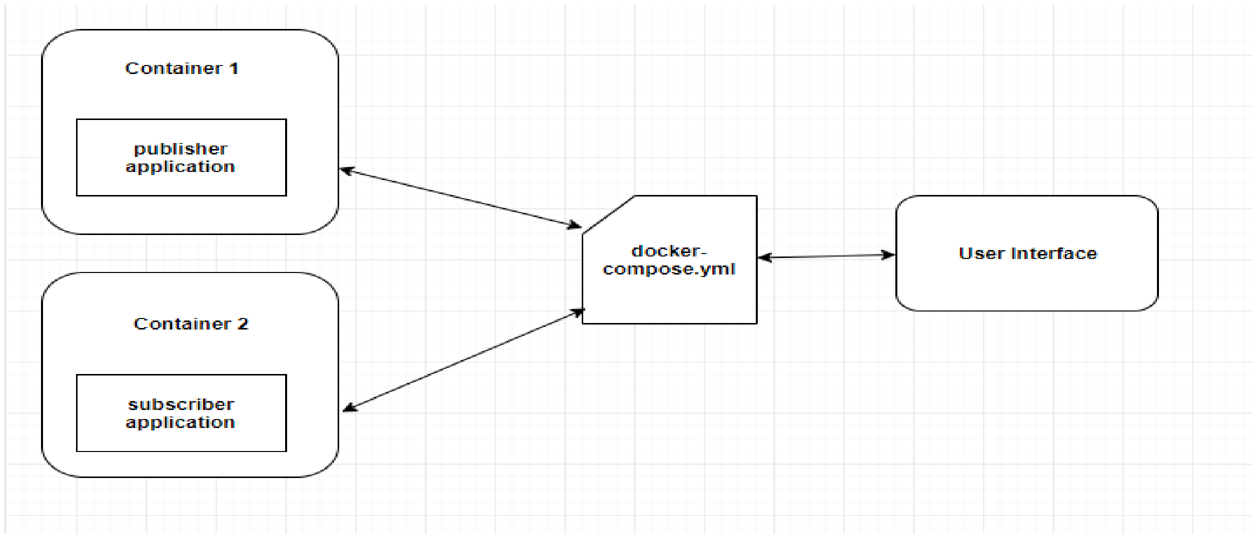
```
Notification for WEATHER by Publisher: TODAY IS A COLD DAY
Attention to all MOVIES subscribers: [anupriya]
NOTIFICATION: infinity war is a good film
Attention to all SPORTS subscribers: [sravanthi]
NOTIFICATION: buffalo bulls won the match

Hurray!!! SRAVANTHI you have succesfully subscribed to SPORTS
Hurray!!! ANUPRIYA you have succesfully subscribed to MOVIES
```

**Figure 8: Notifying to subscriber screen shot for Phase2**

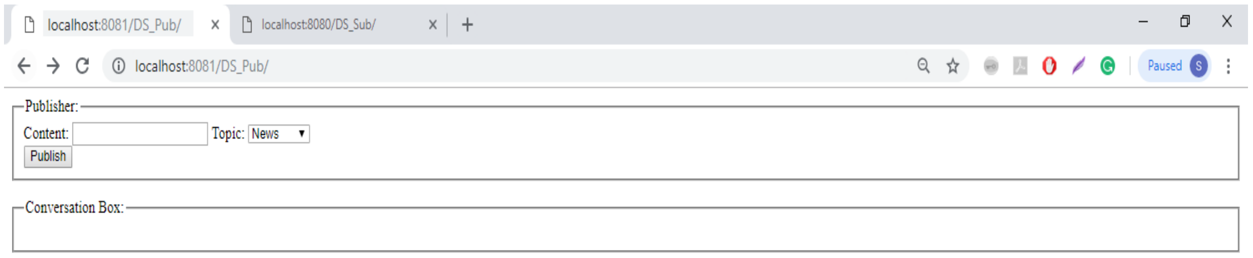
## Phase3:

In this phase we are implementing distributed version of Pub/Sub system.We created individual web-applications for publishers, subscribers and broker. In Phase2 we were having publisher and subscriber in one war file that is communicating through Docker to provide functionality of Pub/Sub system. For Phase3 following steps are implemented:

1. We deployed these applications in different Docker containers by making use of docker compose
2. These applications in different containers communicate with each other to emulate a distributed publish-subscribe system.
3. We generated the following containers for Publisher, Subscriber and Broker (MySQL DB).
4. Compose is a tool for defining and running multi-container Docker applications. We are using YAML file to configure our application so that we create and start all the services from our configuration.
5. Using Compose involves  three-step process:
    - Define app's environment with a Dockerfile so it can be reproduced anywhere.
    - Define the services that make up app and provide both .war files  in docker-compose.yml so that separate containers which is connected  together in an isolated environment.
    - Run docker-compose up and Compose starts and runs your entire app.
6. The MySQL database application contains a table 'pubsub_Table' which is used by both publisher and subscriber applications to emulate a distributed publish-subscribe system



**Architecture model for phase 3**



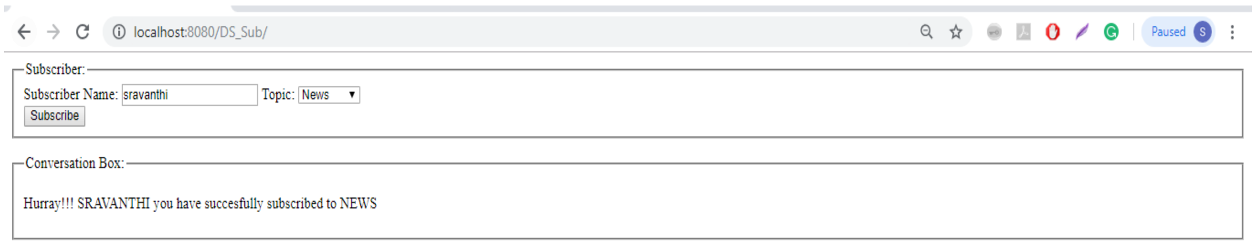
Publisher:

Content: Topic: News

Publish

Conversation Box:

**Figure 9. Home Screen for Phase3**



Subscriber:

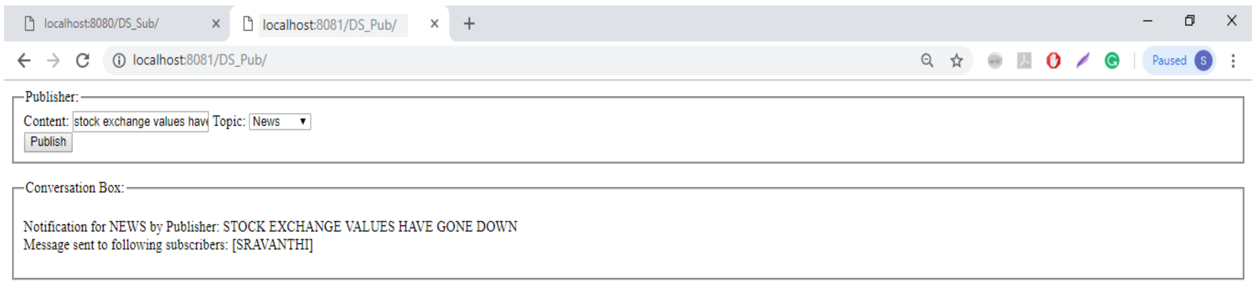
Subscriber Name: sravanthi Topic: News

Subscribe

Conversation Box:

Hurray!!! SRAVANTHI you have succesfully subscribed to NEWS

**Figure 10.Adding subscriber to topic screenshot for phase3**

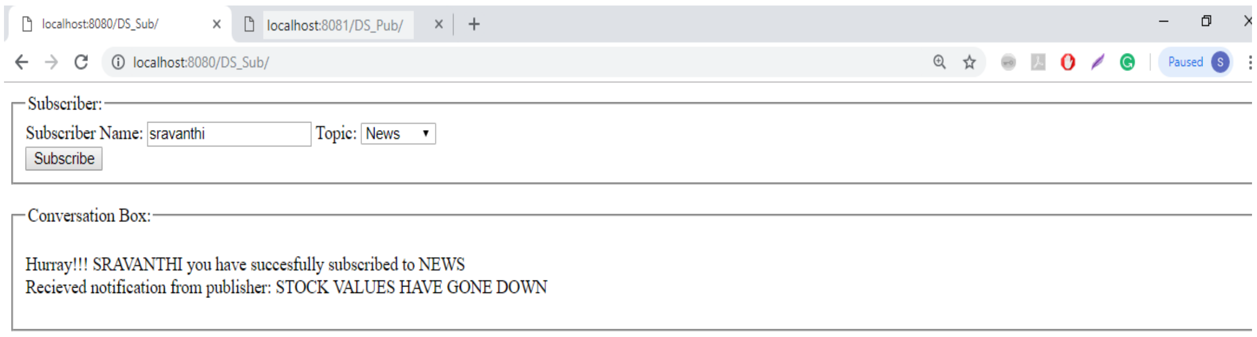


Publisher:
Content: stock exchange values hav Topic: News
Publish

Conversation Box:

```
Notification for NEWS by Publisher: STOCK EXCHANGE VALUES HAVE GONE DOWN
Message sent to following subscribers: [SRAVANTHI]
```

**Figure 11. Notified message from publisher screenshot for phase 3**



Subscriber:
Subscriber Name: sravanthi Topic: News
Subscribe

Conversation Box:

```
Hurray!!! SRAVANTHI you have succesfully subscribed to NEWS
Recieved notification from publisher: STOCK VALUES HAVE GONE DOWN
```

**Figure 12. Subscriber received notification screenshot phase3**

## References:

1. https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern
2. https://aws.amazon.com/pub-sub-messaging/
3. https://docs.docker.com/engine/reference/builder/
4. https://docs.docker.com/compose/