

CSE 586: Distributed Systems (Fall 2021)

Gowtham Rajasekaran
grajasek@buffalo.edu
50363213

Abstract

“Netflix Pub-Sub” is a simple Dockerized and Centralized Python Flask Pub-Sub application. A publish-subscribe system is a system where publishers publish structured events to an event service and subscribers express interest in particular events through subscriptions which can be arbitrary patterns over the structured events. For example, a subscriber could express an interest in all events related to this textbook, such as the availability of a new edition or updates to the related web site.

Architecture

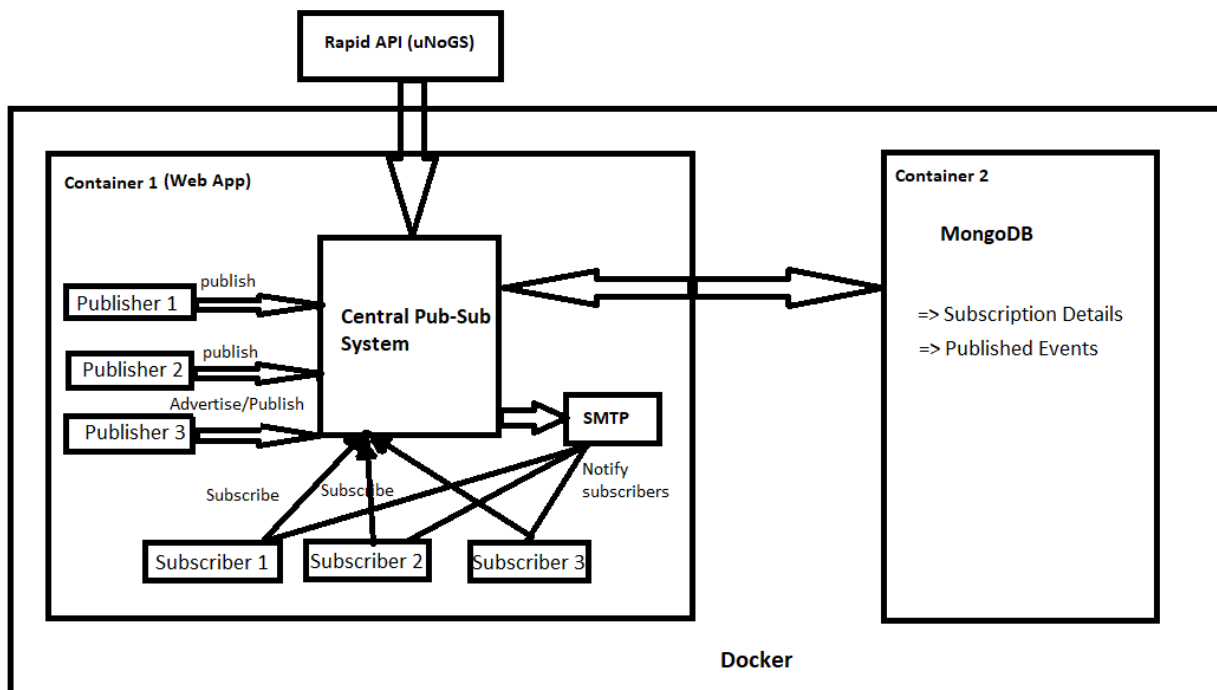


Figure 1: Dockerized “Netflix Pub-Sub” Architecture diagram

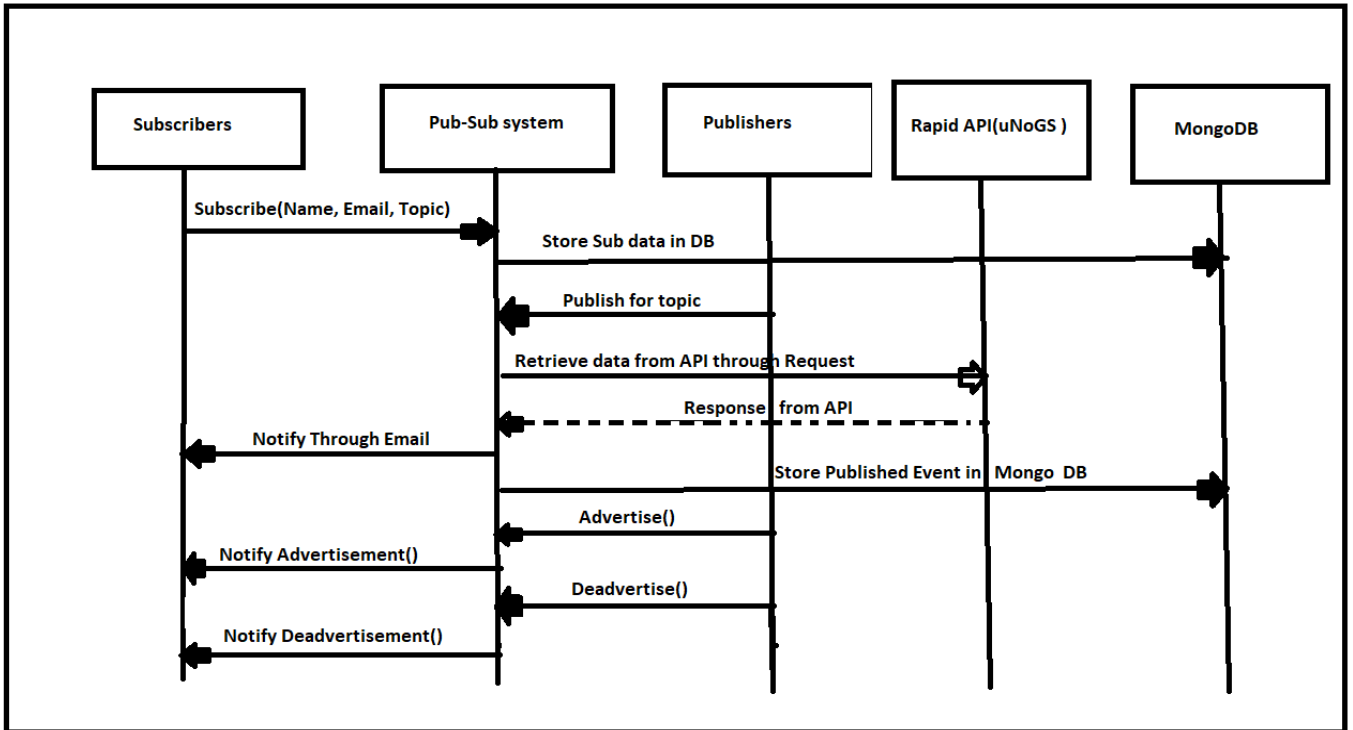


Figure 1: Dockerized “Netflix Pub-Sub” Sequence diagram

Description of Components

Data-Endpoint:

For my pub-sub model I chose to retrieve the Netflix Movies and Tv show information from an unofficial netflix API provided by Rapid API called uNoGS which stands for Unofficial Netflix online Global Search. The data we are currently retrieving from the API through HTTP requests are,

- New Releases - The Top 10 new releases on Netflix.
- Expiring - Expiring from Netflix US this week.
- Deleted Titles - Recent Deleted Titles from Netflix..
- Season Changes - Recently updated Netflix TV shows.

Those are the above data/topics we would be retrieving, furthermore we use the country name, either 'US' or 'UK' to retrieve the data. The country parameter will serve as the filter function during subscribing and publishing events.

Home page

The Web application was built using Python, Flask, MongoDB, Docker, HTML5, CSS3 and a few python libraries. The web application resides on one container and the MongoDB resides on a different container.

The client Facing web application user Interface home page will prompt you to choose between a number of options like publish, subscribe, posts, Advertise. The subscribers use subscribe page for subscribing to events, posts page for viewing the published events, publishers use publish page to publish events from the unofficial netflix end point to the subscribers, publishers(publisher 3 to be exact in our case) use advertise to advertise subscription related deals to subscribers.

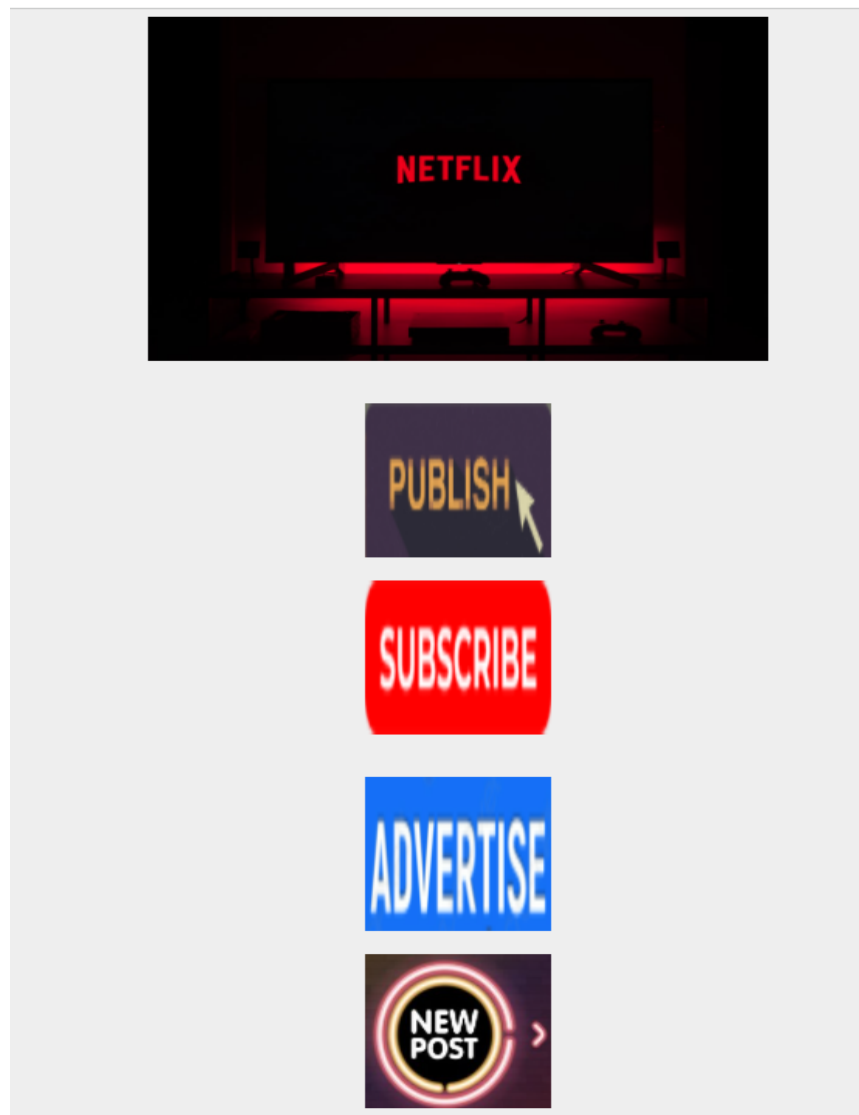
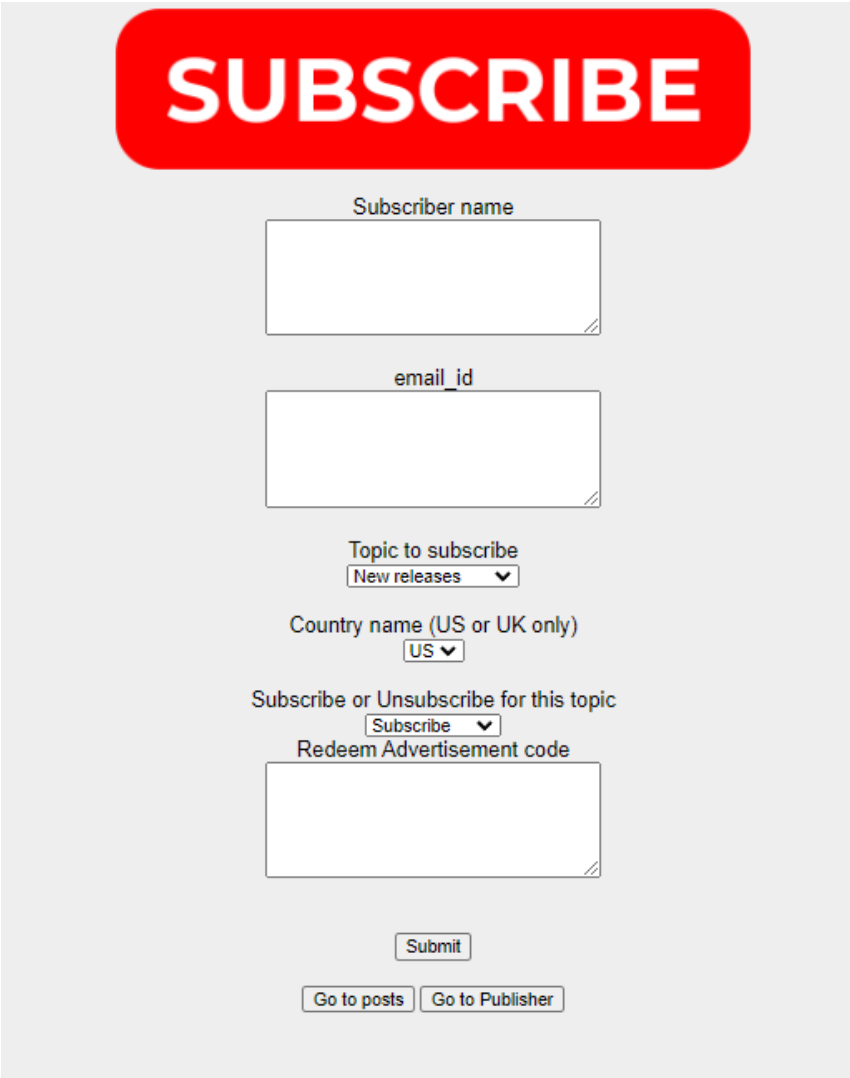


Figure 3: Homepage of the Web app

Subscribe

The first module/component in discussion is the subscribe module which lets the subscribers to subscribe to topics, the subscriber enters their name, Email ID and choses the topic from the drop down list and also selects the country for the topic, the country option acts as a filter so the user can subscribe to new releases from US, Expiring movies from UK, etc. Then there is an input text area for redeem advertisement code which would be discussed in detail in the Advertise section. Finally the user can select whether they want to subscribe for that selected topic or unsubscribe if they are already subscribed.



SUBSCRIBE

Subscriber name

email_id

Topic to subscribe
New releases ▼

Country name (US or UK only)
US ▼

Subscribe or Unsubscribe for this topic
Subscribe ▼

Redeem Advertisement code

Submit

Go to posts Go to Publisher

Figure 4- Subscriber Page.

Publish

Publish module is used by the publishers to publish posts/events the the subscribers, the publishers are asked to enter their names, Topic name and Country name(Note: This field only applies to publisher 3, publisher 1 and 2 will have inbuilt country types which will be discussed below. When the publisher publishes the event, the subscriptions list is retrieved from the MongoDB collection which provides the subscriber name and email and a notify method is called to notify the subscriber through Email. The below table represents the topics each publisher can/is allowed to publish.

Publisher Name	Topics
Pub1	new_releasesUS, expiringUS.
pub2	new_releasesUK, expiringUK
pub3	deleted_titlesUS, deleted_titlesUK, season_changesUS and season_changesUK.

Pub1/pub2 don't require a country name to be selected from the dropdown in the publisher page, because pub1 only publishes for the US, pub2 only for the UK.



Figure 5- Publisher Page.

Advertise

The advertise Module allows publisher3 (pub3) to advertise and de-advertise to the subscribers, When the advertise option is selected all the current subscribers of deleted_titlesUS will receive an email notification stating that

"Advertisement: Subscribe to Deleted titles from UK and get free subscription to Season changes from US and UK. Enter the code PUB_SUB."

So if the subscribers subscribe to 'deleted_titlesUK' and enter the code "PUB_SUB" in the redeem advertisement code, the system automatically subscribes the user to season_changesUS and season_changesUK.

The De-advertise option (ie: 'Cancelled' on web page) notifies the subscribers through Email that the offer that was advertised is no longer available and the code will not work.

To test the advertise functionality

- Create a subscription for delted_titles from us, because the publisher 3 advertises (through Email) only to the subscribers of the Deleted titles US
- Then proceed to create a subscription for deleted titles UK with the advertised code entered as 'PUB_SUB'. Doing this will automatically subscribe the user to the topics season_changesUS and season_changesUK.



Figure 6- Advertisement Page.

Notify

The Notify method notifies all the subscribers in the list with the value sent by the publishers to the central system through Email. The Email module is implemented with the help of SMTP protocol which is leveraged through the smtplib library from python.

Posts

The posts module allows the subscribers to view their posts history retrieved from the database. When the publisher publishes the topic, the event is stored into MongoDB for each user which would be retrieved when the user types their name and submits.

NEW POST

Enter Subscriber name
Jack

Submit

POSTS

POST#1

-->The Top 10 new releases in US are:
My Name
Adventure Beast
Bright: Samurai Soul
Convergence: Courage in a Crisis
Fever Dream
The Forgotten Battle
The Four of Us
Flip a Coin -ONE OK ROCK Documentary-
Found

Figure 7- Posts Page.

How to Run?

Running the web application is simple

- 1) Navigate to the project folder in the terminal.
- 2) Run command “docker-compose up”

Note:

- Each publisher can only publish to specific topics refer to the publishers section to correctly publish.
- Currently Mailing notifications Works using a hard coded dummy email and password and it was tested to work on different systems, If it doesn't work, it might be due to a critical alert by google, in that case please email me at grajasek@buffalo.edu so i can fix it or login to the gmail ID with the credentials given in notify function and switch off the critical alert.
- Advertise and De-advertise are just images which act as buttons. Just clicking them once is sufficient, there is no interaction on that page.
- When testing advertise please enter the code 'PUB_SUB' during the subscription phase.

Project File Structure

Static:

style.css => Utilize styles for the web page

Images => contains images for rendering the web page

templates:

home.html => Renders the Home page

subscribe.html => Renders the subscriber page

publish.html => Renders the publisher page

posts.html => Renders the posts page for subscribers.

advertise.html => Renders the advertisement page for publisher3.

app.py => Server side Application which handles the input and processes the output to render to the user.

init-db.js => Javascript file to initialize and populate the MongoDB with records.

Dockerfile => A text document that contains all the commands a user could call on the command line to assemble an image

Docker-compose.yml => Allows to deploy, combine and configure multiple docker-containers at the same time.

References

The Images and Emojis used in the HTML pages were used from google images and emojitranslate.com.

Project built from references from

- 1) <https://docs.docker.com/>
- 2) <https://flask-doc.readthedocs.io/en/latest/>
- 3) <https://www.w3schools.com/w3css/defaultT.asp/>
- 4) <https://rapidapi.com/unogs/api/unogs/>