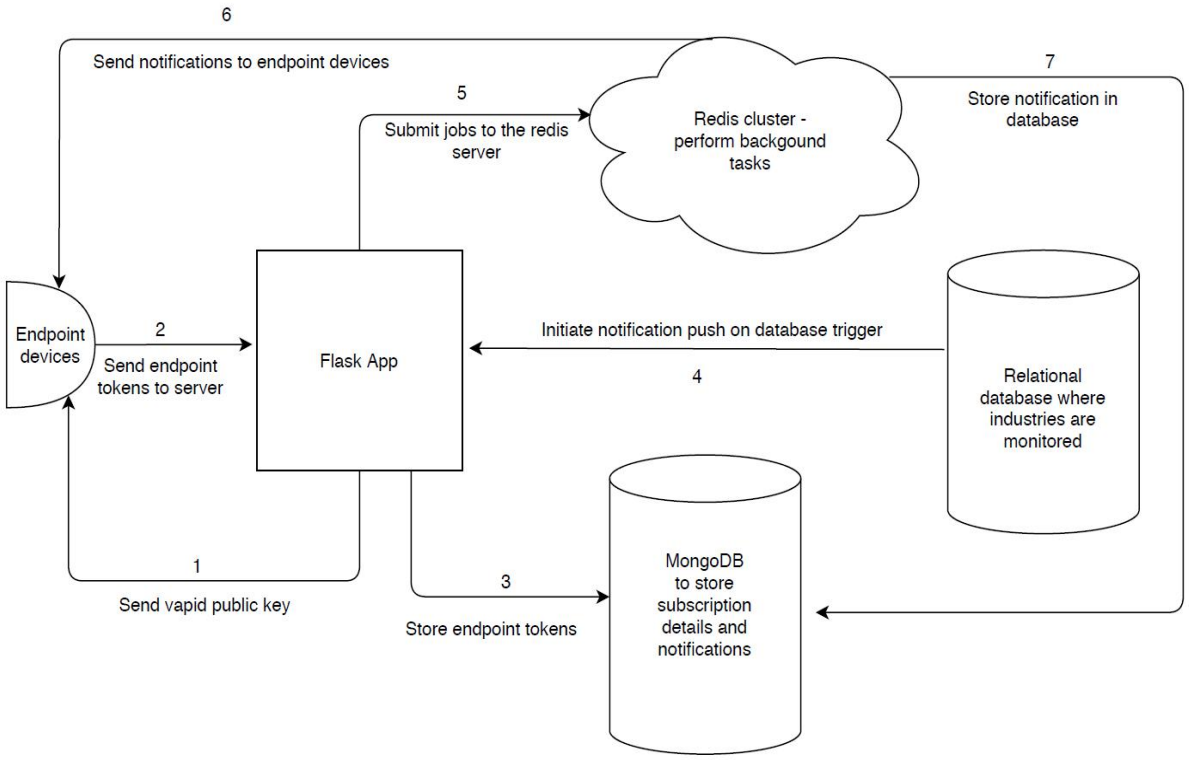
Simple Notification Service

Overview:

This project focuses on building a scalable notification service to send push notifications to users based on their subscriptions to a given topic.

System Design :



1) The Flask application sends it's vapid public key to the endpoint devices.

2) Using the vapid public key of the web server, the endpoint device generates a subscription token which is unique for every browser and sends it to the web server.

3) Then that endpoint device is added as a subscriber to a particular topic (in our case topic means industry)

4) Whenever a topic(ie industry) is created, database triggers are set on that industry. As soon as a database trigger occurs, the notification publish is initiated.

5) The flask app receives the notification publish api call and it submits that job to the redis cluster. Redis cluster may have many workers listening on to 3 queues - db-jobs, webpush-jobs and publish-jobs. Any worker which is free can accept the job.

6) The redis worker is going to publish the notification to all the subscriber of a particular topic.

7) The redis worker stores the notification in the database

API Endpoints :

Get vapid public key of the web server

Route : /api/v1/vapid/public/key
HTTP Request method : GET
Relevant HTTP Response code : 200, 404
Request : None
Response : {'public_key': <vapid_public_key_of_server>}
Description : The browser should fetch the public key of the server to generate a subscription token unique for itself.

Create a topic

Route : /api/v1/topics/
HTTP Request method : POST
Relevant HTTP Response code : 201, 400
Request body: {'topic': <topic_name>, 'description': <description_about_topic>}
Response body: None
Description : In our case topic means industry. For example, if Google is a customer of Fluxgen, then we should create {'topic': 'google', 'description':<optional>}

List all topics
Route : /api/v1/topics/list
HTTP Request method : GET
Relevant HTTP Response code : 200, 204
Request body : None
Response body : [<topic1>, <topic2>, <topic3>, … <topicN>]

Delete a topic
Route : /api/v1/topics/<topic_name>
HTTP Request method : DELETE
Relevant HTTP Response code : 200, 400
Request body : {'topic': <topic_name>}
Response body : None

Subscribe endpoint device to a topic
Route : /api/v1/topics/subscribe
HTTP Request method : POST
Relevant HTTP Response code : 200, 400
Request body : {'topic': <topic_name>, 'token': <endpoint_token>}
Response body : None
Description : The browser should send it's subscription token to the server in order to subscribe to a topic

Unsubscribe endpoint device from a topic
Route : /api/v1/topics/unsubscribe
HTTP Request method : POST
Relevant HTTP Response code : 200, 400
Request body : {'topic': <topic_name>, 'token': <endpoint_token>}
Response body : None
Description : In order to unsubscribe from the topic, the browser should send it's subscription token

Push notification to single endpoint device
Route : /api/v1/notifications/push
HTTP Request method : POST
Relevant HTTP Response code : 200, 400
Request body : {'token' : <endpoint_token>, 'notification' : <message_in_json_format>}
Response body : None

Publish notifications to subscribers
Route : /api/v1/notifications/publish
HTTP Request method : POST
Relevant HTTP Response code : 200, 400
Request body : {'topic': <topic_name>, 'notification' : <message_in_json_format>}
Response body : None
Description : Using the topic name, it will fetch the endpoint tokens of all the subscribers of that topic and publishes notifications to all of them.

List notifications
Route :/api/v1/notifications/list?topic=<topic_name>&start=<list_from>&end=<list_until>
HTTP Request method : GET
Relevant HTTP Response code : 200, 204, 400
Request body : None
Response body : [<notification_start>, <notification_start+1>, notification_start+2>, …<notification_end>]
Description : List notifications in decreasing order of their timestamps. For example if start = 0 and end = 10, it will list the latest 10 notifications. If start = 10 and end = 20, it will list the next 10 notifications. And so on.

MongoDB document structure :

- Each topic is a document in a collection called ‘topics’
- Structure of each document :

{
‘topic’ : <topic_name>,
‘description’ : <brief_description_about_topic>,
‘subscribers’ : [
<subscriber1_token>,
<subscriber2_token>,
.
.
.
<subscriberN_token>
],
‘notifications’ : [
<notification1_json>,
<notification2_json>,
.
.
.
<notificationN_json>
]
}

Redis Queues :

- Initially 3 redis workers (i.e.; docker containers) are started.

1) **webpush-jobs** - pushes notification to a single endpoint device. *'redis-worker-webpush'* listens on this queue.

2) **db-jobs** - stores the notification in the database. '*redis-worker-db*' listens on this queue.

3) **publish-jobs** - assigns jobs to the *webpush-jobs* queue and *db-jobs* queue. '*redis-worker-publish*' listens on this queue

Whenever, we need to scale up/down we could just increase/decrease the number of redis workers respectively.

Docker containers :

1) ***app*** - the REST application runs here

2) ***mymongo*** - `*app*` interacts with this container for database operations

3) ***redis-server*** - used to perform background jobs

4) ***redis-worker-webpush*** - performs the job of pushing notification to a single endpoint device

5) ***redis-worker-db*** - performs the job of storing the notification for the appropriate topic in `*mongo*`

6) ***redis-worker-publish*** - performs the job of assigning jobs to `*redis-worker-db*` and `*redis-worker-webpush*`

Installation and setup :

- Generate vapid public key and vapid private key.
  $ sudo npm install -g web-push
  $ web-push generate-vapid-keys

- Update the .flaskenv file with newly generated public and private keys.

- Add an environment variable ‘SECRET_KEY’ in .flaskenv file which is used by flask for security purposes
  $ echo “SECRET_KEY=<secret_key_here>” >> .flaskenv

- Install docker
  $ sudo apt install docker docker.io docker-compose

- Build all docker containers for the first time
  $ cd Notification-Service
  $ sudo docker-compose build .

- Start the notification service
  $ sudo docker-compose up