Suppose the following scenario:

We want to notify a number of different employees of our organization about temperature and humidity changes in some critical industrial process of our factory. To do this, we already placed several temperature and humidity sensors in the main reactors. These sensors are connected (via a magical link) to a PostgreSQL database. The database has a table named 'measurements' where the sensors store the information. The table looks like this:

| Date | Sensor ID | Temperature | Humidity |
|---|---|---|---|
| 12:00:15+3 | 1000 | 25 | 50 |
| 12:10:15+3 | 1001 | 30 | 45 |
| ... | ... | ... | ... |

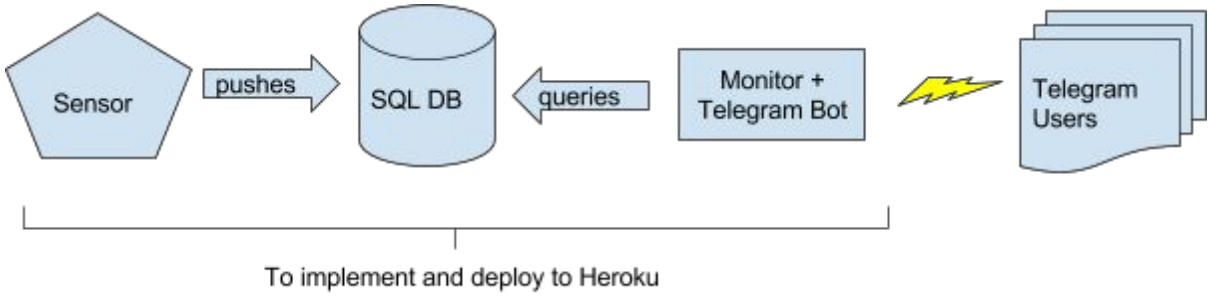
We need a Monitoring process that reads this table looking for new data and then checks the values of the sensors against a certain criteria. If an alarm condition is triggered, then an alarm should be generated and send using a Telegram Bot to all the employees registered to that Bot.

The normal conditions for the plant are:

- The temperature must be between 40 and 45 degrees
- The humidity must be between 60 and 80 %

If any of those conditions is not met, then an alarm is triggered. The alarm message should contain the moment the condition was measured and the ID of the sensor that produced it.

The following picture describes this situation:




You would need to write an application that:

- Periodically generates 'fake sensor data' and store it in a Postgres SQL database
- Periodically makes a query to get the new measurements from the database
- Compares the values against the defined thresholds
- If a problematic condition is detected, an alarm is sent to all the users that have started a conversation to the Telegram Bot.
- The application (and the database) must be deployed to Heroku

- The application code must be available on a public github repository

Tips:

- You can make a simple loop that:
  - generates a random measurement ( N(42, 5) for T and N(70, 20) for H for example) and inserts it into the database.
  - makes a query based on the last measurement time, in other words each query asks for a new dataset (remember that the insert part is just a mock we create for this test and that in reality it would be the sensors pushing to it)
  - checks against the thresholds
  - sends alarms.
- Telegram Bot:
  - You will need to create a bot and have a telegram regular user.
  - When a user starts a conversation with the bot, a start message is sent to the bot.
  - The bot should remember the ids of the users that initiated the chat with the bot.
  - Each time an alarm is generated, send the message to all the subscribed users.
  - https://core.telegram.org/bots
- Take into account that the code is much better when you are able to test it before deploying to heroku.
  - How are you testing it? Integration tests? Unittests?
- There is no need to have a Web UI for the application (unless you want to use as a debugging tool)
- You can use any of the supported languages in Heroku (Python, Node.js, …)
- You are encourage to use as much third parties modules as possible to reduce the custom code needed.