octopus|labs

Octopus Labs
33 Holborn, London EC1N 2HT
octopuslabs.com

t: +44 (0)800 294 6848
e: hello@octopuslabs.com

As promised and agreed, please see the code challenge. I would like the code delivered to me as a Github repo that I can access, with a clear README file with install instructions. We will run *docker-compose up* to see the software working, so please ensure we can have a local server running just that command.

Please complete it before the end of tomorrow. Just send me an email when you are done, letting me know how many hours you took.

Octopus is keen to develop its technology as a series of cloud based, RESTful micro-services, with good security measures in place to protect customer data. The code-challenge thus includes the requirement to build some application-level encryption.

1) Use *docker-compose* for the different parts of the application (web server, DB)
2) Create a *Python* web application using Tornado web server.
3) The project should have a single page with a form where I can enter a URL to any website (e.g. Wikipedia or BBCNews)
4) The application should fetch that URL and:
    a. Build a dictionary that contains the frequency of use of each word on that page.
5) Use this dictionary to display, on the client’s browser, a “word cloud” of the top 100 words, where the font size is largest for the words used most frequently, and gets progressively smaller for words used less often.
6) Each time a URL is fetched, it should save the top 100 words to a MySQL DB, with the following three columns:
    a. The primary key for the word is a salted hash of the word.
    b. The word itself is saved in a column that has asymmetrical encryption, and you are saving the encrypted version of the word.
    c. The total frequency count of the word.

   Each time a new URL is fetched, you should INSERT or UPDATE the word rows.
7) An “admin” page, that will:
    a. List all words entered into the DB, ordered by frequency of usage, visible in decrypted form.
8) **Optional**
    a. Use wit.ai to perform sentiment analysis (positive, negative) on the text of the document. Feel free to train the system in the way you wish.
    b. each time a URL is fetched, it should save the sentiment analysis to a MySQL DB, with the following columns:
        i. The primary key as salted hash of the URL
        ii. The primary key as the URL
        iii. The result of sentiment analysis (positive, negative)
    c. In the admin, list all the URLs with their negative/positive qualification

We are looking for:
- Good project layout/structure
- Clear README
- Clean, well-documented code

Extra points for:
- Displaying just nouns and verbs (no prepositions or articles)
- In the README, describe the best way to safely store and manage the encryption keys
- Elegant front-end layout
- Unit tests

Feel free to let me know if you have any questions.

Good luck!