Node.js/ express/ git/ déploiement *cloud* Heroku / jQuery Ajax

Vous êtes informaticien (nne) dans une société de service. Une firme de recherche en génétique vous demande de réaliser un service Web pour donner accès à un service d'analyse de similarité d'ADN. Service sans hébergement : il n'y a pas d'informaticien ou de machines serveur dans l'entreprise.

Vous allez réaliser un site Web d'accès à des pages Web et à un service de calcul et vous allez l'héberger sur le cloud.

Les brins d'ADN sont des marqueurs très spécifiques d'une personne. Leur codage peut (en simplifiant) se représenter par une chaine de lettres A, C, G et T. Quand on retrouve des traces de sang sur les lieux d'un crime, on peut, à partir de ces traces de sang, retrouver la chaîne d'ADN et ensuite, en comparant cette chaîne à d'autres chaînes, déterminer de quelle personnes l'ADN retrouvée est la plus proche.

Le problème, c'est que, aussi bien pour le sang trouvé que pour les échantillons déjà disponibles, il est quasiment impossible d'obtenir une chaîne parfaite et sans erreur. Donc, comparer deux chaînes n'est pas si simple. Quand on a deux chaînes, C1 et C2, on essaye de trouver le nombre d'opérations élémentaires qui permettent de passer d'une chaîne à l'autre.

Par exemple, rajouter un caractère : distance(ACG, AC) = 1, mais distance(ACG, AG) = 1

Ou enlever un caractère : distance(AG, ACG) = 1

Ou substituer un caractère : distance(ATG, ACG) = 1

La distance de deux chaînes est la somme minimale des opérations élémentaires à réaliser pour passer d'une chaîne à l'autre.

La firme a réalisé un algorithme pour ce travail : cet algorithme est breveté, pas question de donner l'accès au code à qui que ce soit. Par contre, la firme souhaite permettre gratuitement aux services judiciaires d'utiliser ce système. Gratuitement : pour l'instant.

Vous allez donc, dans un premier temps, réaliser un site web (en utilisant Node.js, express et pug/jade) qui donne accès à la page index.html (présentation de la firme), mais qui donne également accès au service de calcul de distance de chaînes.

Pour calculer la distance entre deux chaines d'ADN, le client devra lancer une requête GET, en passant le chemin /**gibbs4567**/distance/ et les variables A (avec comme contenu la première chaine) et la variable B avec comme paramètre la seconde chaine.

gibbs4567 est une clé (fixée dans un premier temps) qui permet d'utiliser ce service. Vous ne pouvez pas interroger le système si vous n'avez pas une clé valide.

Le serveur répond en envoyant une chaine JSON :

{ "utilisateur" : "gibbs4567",

"date" :"3 mars 2018 10h12",

"A" :"ACGTGCAGTACGATGCGTAGC",

"B" :" ACGTGCTGTATGATGCGTAG ",

"distance" : "3",

"temps de calcul (ms)" : "150",

"interrogations minute" : 4

}

Le serveur contient une liste des clés valides (inventez 10 clés valides : elles se terminent par 4 chiffres obligatoirement). Pour ne pas surcharger le service, chaque utilisateur n'a le droit qu'à 5 requêtes par minute. Quand le nombre est dépassé, la réponse du serveur est :

{

"utilisateur" : "gibbs4567",

"erreur" : "nombre de requêtes dépassé, attendez une minute"

}

Les erreurs possibles sont également :

"vous n'avez pas les autorisations pour utiliser ce service"

"une des deux chaînes est trop longue (gardez des chaînes inférieures à 50)"

"une des chaînes ne code pas de l'ADN"

"la requête est mal formée"

Pour le calcul (secret) de distances entre chaînes, vous utiliserez le package **js-levenshtein**. Vous inventerez une première page d'index (genre :

ou

A vous de voir – utilisez PUG/JADE

)

Cette page indique également et de manière dynamique, le nombre de requêtes réalisées par tous les utilisateurs (mise à jour toutes les 30 s).

Première partie : réalisez le serveur express qui réalise ces fonctionnalités. Testez-le localement. Utilisez obligatoirement le mécanisme **package.json** pour décrire votre projet (qui doit pouvoir être déployé par **npm install**)

…

Seconde partie : déployez votre serveur sur le cloud.

L'entreprise génétique n'a pas de service informatique. Vous allez lui proposer un hébergement de type cloud parce que le serveur doit réaliser des calculs et pas seulement fournir des pages web. En fonction de la montée en charge de l'application et en fonction du nombre de clients, il sera possible d'adapter la puissance et la mémoire de l'hébergement (normalement, d'ailleurs, le service cloud est facturé de manière dynamique). A terme, le projet de l'entreprise est de faire payer ses clients (et financer l'accès cloud). Pour l'instant, les clients 'gratuits' vont faire de la publicité.

Heroku est une entreprise qui fournit des services de type cloud. Il est possible d'en utiliser certains de manière gratuite qui offrent un minimum de ressources. Heroku permet de déployer des solutions écrites en Ruby, **Scala**, Java, Python, Go, Clojure, Php et évidemment **Node.js**. **Heroku** n'est pas le seul fournisseur de services cloud (mais il fait partie des plus importants).

Lisez l'énoncé complètement avant de commencer.

1. Commencez par créer un compte **Heroku** (sur Heroku.com).

2. Vous devez ensuite importer les outils de déploiement **Heroku** qui vont vous permettre d'envoyer vos applications sur le cloud en utilisant **npm** :

Si vous n'êtes pas admin sur votre machine, *modifiez le prefix d'installation npm* (voir le cours) pour permettre d'installer les packages global *npm i - - g* (et avoir accès aux commandes).

Vérifiez que les paramètres proxy de **npm** sont ok.

Si nécessaire :

npm config set proxy "http://squidva.univ-ubs.fr:3128"

npm config set https-proxy "http://squidva.univ-ubs.fr:3128"

npm config set strict-ssl true

et

export HTTPS_PROXY=http://squidva.univ-ubs.fr:3128
export HTTP_PROXY=http://squidva.univ-ubs.fr:3128

puis installez le *cli* (command line interface) par :

npm install -g heroku-cli

Attention, vous devez avoir une version de **node** récente (au moins 8.3.0).

Vérifiez l'installation par ***heroku --version*** : normalement, le script est accessible directement (si ce n'est pas le cas, c'est que vous n'avez rien installé en global – repartez à l'étape de modification de prefix etc..

3. Supposons votre projet dans le répertoire ***monprojet***. Il vous faut un espace de projet avec un dépôt local (git) pour gérer votre travail. Si vous n'êtes pas familier avec git (**froncement de sourcils**), vous pouvez rapidement vous mettre à jour avec la page : http://rogerdudler.github.io/git-guide/index.fr.html

4. Si ce n'est pas déjà le cas, équipez votre projet d'un espace de dépôt (*init*, *add*, *commit*).

5. Faites attention de ne pas ajouter *n'importe quoi* dans votre dépôt (par exemple, pas le répertoire *node_module*) (**froncement de sourcils**). Vérifiez par :

   git ls-tree - -full-tree –r HEAD

6. Ne passez pas au déploiement si votre programme ne fonctionne pas en local (et est testé correctement)

7. Vous allez maintenant créer un projet cloud en utilisant le système Heroku.
   heroku login

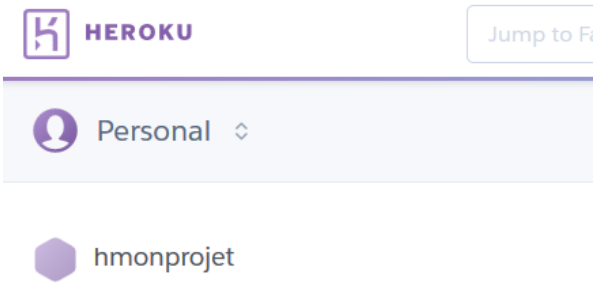
   Entrez vos paramètres de connexion (ceux créés pour votre compte Heroku)

8. Placez-vous dans le répertoire de votre projet (équipé d'une espace de dépôt), lancez
   heroku create hmonProjet

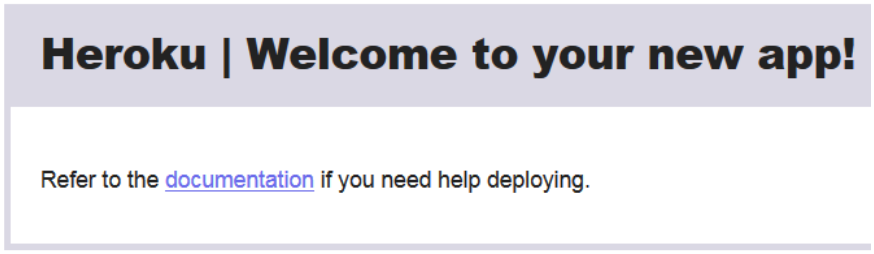
   Heroku fabrique un espace projet. Attention, il vous faut trouver un nom non déjà utilisé (car ce nom sert ensuite dans le nommage d'une URL)

   (Au cas où, vous pouvez effacer un projet Heroku de la manière suivante :
   heroku apps:destroy --app hmonprojet)

   Connectez-vous sur votre compte web **Heroku** : https://dashboard.heroku.com/apps
   Vous pourrez consulter l'état de votre travail ainsi que son déploiement Cloud.



Consultez la page https://hmonprojet.herokuapp.com/ (évidemment remplacez par le nom de votre projet).



Un espace git a également été créé : https://git.heroku.com/hmonprojet.git (avec votre nom de projet)

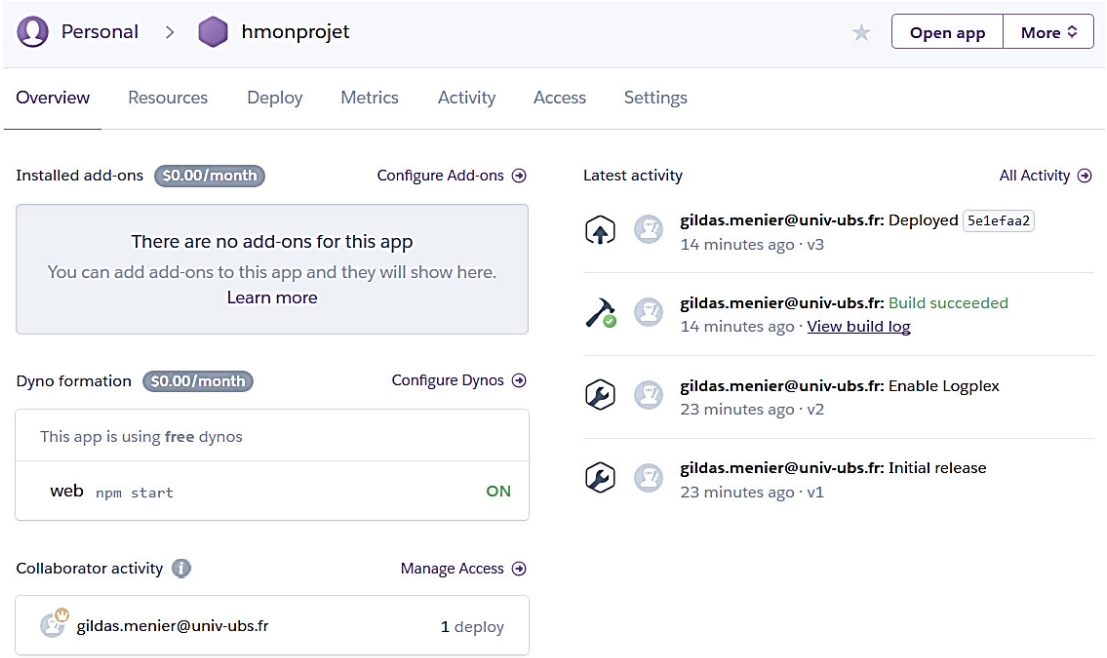
9. Vous allez maintenant publier votre projet vers **Heroku** en utilisant votre dépôt git local :
   git push heroku –u master

```
remote:        Skipping because npm 5.5.1 sometimes fails when runni
e' due to a known issue
remote:        https://github.com/npm/npm/issues/19356
remote:
remote:        You can silence this warning by updating to at least
 your package.json
remote:        https://devcenter.heroku.com/articles/nodejs-support#
n-npm-version
remote:
remote: -----> Build succeeded!
remote: -----> Discovering process types
remote:        Procfile declares types     -> (none)
remote:        Default types for buildpack -> web
remote:
remote: -----> Compressing...
remote:        Done: 17.3M
remote: -----> Launching...
remote:        Released v3
remote:        https://hmonprojet.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/hmonprojet.git
 * [new branch]      master -> master

D:\users\menier\Documents\_Doc_\_Doc_Prog\heroku\monprojet>
```

10. L’environnement de programmation décrit par votre **package.json** est déployé : il est donc TRES important de bien utiliser et de bien spécifier **package.json**.

11. Vérifiez sur le dashboard :



12. En théorie, vous devriez pouvoir ouvrir la page https://hmonprojet.herokuapp.com/

    (ou par la commande '**heroku open'**)

    Normalement à ce stade, votre application est déployée. Cependant, si votre application est un serveur web, il vous faut indiquer dans votre code le port à utiliser par **Heroku**. Vous pouvez déterminer le numéro de port de la manière suivante :

    const port = process.env.PORT || 5000

    process.env.PORT est une variable d'environnement qui permet au fournisseur d'accès cloud de vous offrir un numéro de port pour communiquer avec l'extérieur. Par défaut, en local, cette ligne utiliserait 5000 (pour les tests), mais cette valeur de port pourrait ne pas être disponible pour le redéploiement du code cloud : il faut demander au fournisseur une valeur acceptée par les ressources utilisées.

    N'oubliez pas de lancer un **commit** (si vous avez un message qui indique la non prise en compte des modifications, utilisez l'option **–a** (ou –am) de **commit**. Puis refaites un **push heroku –u master**. Vérifiez que vous avez compris ce qu'est l'option –a.

Votre application est maintenant hébergée sur le cloud. **Heroku** (comme les autres fournisseurs de cloud) vous offre des services payants de métrologie, de mémoire, ainsi que de processus/conteneurs (*dynos*). Dans la version gratuite, vous n'avez qu'un accès à peu de mémoire et peu de puissance de calcul.

Notez bien la mise à disposition d'une valeur de port par le fournisseur de solution cloud. C'est exactement cette stratégie qui est utilisée par d'autres fournisseurs comme **Nodejitsu**, **AWS** (Amazon Web Service), **Amazon's Elastic Beanstalk** etc..

Dans le cas de ce TD, vous déployez un serveur Web – mais vous pouvez déployer absolument n'importe quelle application. Il est évidemment possible de l'associer à un nom de domaine que vous avez enregistré (voir les cours réseau).

Un des gros avantages de cette solution d'hébergement cloud est de permettre de modifier à la demande la puissance et la mémoire nécessaire (en fonction des clients et travaux à effectuer) avec une administration minimale et un déploiement simple.

Refaites un passage sur ce TP : vérifiez que vous avez bien compris ce qui est hébergé où, où se trouve votre projet, à quoi sert GIT, comment se fait le lien du port cloud et de celui de votre serveur, comment s'administre votre déploiement etc.

Troisième partie : page d'accueil et démonstration en ligne (utilisez jQuery + Pug/Jade)

On veut donner la possibilité aux clients de tester la recherche. Vous allez donc réaliser une autre page Web (accessible à partir de la page d'index par un lien). Cette page contiendra un

formulaire qui permettra à l'utilisateur de rentrer sa clé (mot de passe caché) puis les deux chaînes de recherche et enfin un bouton d'interrogation.

En cas d'erreur, un **alert** indique le problème. Sinon, une liste UL/LI se forme en bas de l'écran avec les requêtes + la réponse + l'heure d'interrogation. Un bouton permet d'effacer cette liste.

A rendre :

Un rar (ou zip) avec :

Un readme (qui indique également l'adresse web de votre application déployée)
Le source minimal pour le déploiement complet.

Remarque :

Ce TD décrit une chaîne possible d'outils (très) simplifiée de développement web : GIT, npm et package.json sont ***toujours très importants*** pour la pile javascript. Il vous manque encore la base de données (voir avec MongoDB) pour arriver à ce qu'on appelle développeur ***fullstack***.