April 2024 -- Deploy on Jelastic

Nouvel environnement

Pour déployer budgetApp, j'ai choisi un environnement nommé `budgetapp-prod-a` contenant :

- un 'application server' `apache-ruby` (`nginx-ruby` plus cher !) avec une IP publique (pas d'accès SLB)

Et un autre (!) environnement nommé `app-data1` contenant :

- un 'noeud sql' `postgre 15.5` avec accès SLB.

Je n'ai pas modifié les configuration cloudlets par défaut.

Avoir un environnement séparé pour la base de donnée permet :

- de pouvoir déployer une mise à jour dans un autre environnement que `budgetapp-prod` (par exemple `budgetapp-prod-b`), avec `budgetapp-prod-b` pointant sur la même base de donnée. Ainsi le basculement de version peut se faire sans délai pour les usagers, par exemple en changeant l'IP vers lequel pointe le gestionnaire DNS. Cela permet un retour en arrière simple en cas de dysfonctionnement.
- d'utiliser le même gestionnaire de base de données pour plusieurs applications. Si jamais mettre tous ces oeufs dans le même panier n'est pas opportun alors je créerai `app-data2` (and so on; raison de la numérotation initiée.)

Note : des deux environnements `budgetapp-prod-a` et `budgetapp-prod-b` , stopper celui vers lequel le DNS record pointe. Ainsi, seul l'environnement utile est facturé. Les deux environnements ne devraient être actifs tous les deux qu'en situation de test précédent une mise à jour.. ATTENTION toutefois, bien s'assurer de mettre à jour le bon environnement !

Déploiement de l'application

1. Dans 'Gestionnaire de déploiement', renseigner un nouveau projet avec un token tout frais créé sur github. Bien faire attention à ne pas introduire de 'typo' dans l'url : `https://github.com/MaTsou/budgetApp.git` .

   > Le git token doit avoir les checkbox `repo` et `admin:repo_hook` cochées.

   Ce gestionnaire conserve les données du déploiement. Ainsi, si je change de containeur (environnement), je peux recréer le déploiement initial depuis le gestionnaire. Les update suivants sont faits depuis le conteneur (on ne déploie qu'une fois)

   Il reste à tester comment la mise à jour d'un token peut être opérée (on peut le changer dans gestionnaire de déploiement : cela suffit-il ?)

Dès lors, les mises à jour sur github peuvent être déployées, soit manuellement (clic sur la flèche verte...) soit automatiquement (configuration du déploiement). Je préfère manuellement (à rapprocher du choix fait plus haut de travailler avec 2 environnements).

À tester : doit-on systématiquement `bundle install` (je pense que non; le usecase est un ajout de gem) ; idem pour `rails assets:precompile` . Note : `rails assets:precompile` doit être suivi d'un relancement du serveur.

Lien avec la base de données

À la création de l'environnement, on reçoit un mail de confirmation avec les identifiants du gestionnaire de base de données -- qui est accessible en cliquant sur 'open in browser' (pour le noeud bdd) :

```
host: nodexxxx-app-data1.jcloud-ver-jpe.ik-server.com
id: webadmin
pwd: PQKmln28921 (obviously, here is a fake pwd !)
```

Mettre à jour `config/database.yml' (si pas déjà fait) :

```
production:
  <<: *default
  database: budgetApp_production
  username: webadmin
  password: <%= ENV["BUDGETAPP_DATABASE_PASSWORD"] %>
```

Variables d'environnement

Dans Application Server, cliquer sur la roue dentée puis 'variables'. À cet endroit, on a accès aux variables d'environnement et on peut en créer.

Créer les variables suivantes :

- RAILS_ENV : production
- DATABASE_URL : postgresql://nodexxxxxx (le host du mail)
- RENTAPP_DATABASE_PASSWORD : PQKmln28921

Bien redémarrer le serveur après cela. Vérifier dans /webroot/ROOT que les variables définies existent `printenv | grep DATABASE` .

Cryptage des infos sensibles

Dans le web-ssh, `/webroot/ROOT` , créer `config/master.key` avec un contenu identique à celui que j'ai en local (sur ma machine). Attention, ce fichier ne doit pas se retrouver sur github car il contient la clé en clair. Cette clé ne doit en aucun cas être perdue ! Elle permet de décrypter le fichier `config/credentials.yml.enc` qui, lui, passe par github (et est généré à la création d'une app. ou avec `rails credentials:edit` .

Lancement de l'application

```
bundle update
bundle install
rails db:create
rails db:migrate
rails assets:precompile
```

J'ai parfois eu besoin de lancer 2 fois `bundle install` (notamment pour le gem `pg`); je n'ai pas compris pourquoi..

Idem pour `rails assets:precompile` .

Ensuite, `open in browser` fonctionne..

Configure the public IP and SSL certificate to access the app.

Note the public IP (ipv4) and go to infomaniak DNS page. Add a new DNS A-record pointing to this IP with url `budgetapp.logicore.fr` or whatever (for testing site, I choose `budgetapp-testingzone.logicore.fr` .

Certificat SSL (un certificat par sous-domaine !):

- 1e option : certificat gratuit basique (Let's Encrypt). Le faire directement dans l'environnement jelastic. Tout en haut, onglet `MarketPlace` puis `Add-ons` .
- 2e option : certificat payant (avec assurance). Le faire depuis Infomaniak puis télécharger le certificat, l'extraire et retourner sur jelastic pour l'introduire dans la configuration :

  > 'Paramètres de l'environnement principal' -> 'SSL personnalisé'.
  > Télécharger les fichiers :
  >
  > - logicore.key
  > - ca_bundle.crt
  > - logicore.crt
  >
  > puis installer.