



Lamine
DABO
Cohorte 3 SA





Heroku ?

Heroku est une plateforme en tant que service (PaaS) créant des logiciels pour serveur qui permettent le déploiement d'applications web.

Nous allons voir ici comment déployer une application Symfony utilisant Api platform (ou pas)

Regardez la vidéo détaillée ici → <https://youtu.be/yLKr1A5-XeY>



❖ Prérequis

Installer Git → Git installation instructions

Installer Heroku Cli → Heroku CLI installation instructions

Avoir un compte heroku → signup is free and instant

Regardez la vidéo détaillée ici →

<https://youtu.be/yLKr1A5-XeY>

Initialisation du projet

- ❖ Créer une nouvelle application sur

<https://dashboard.heroku.com/apps>

- ❖ Après création de l'app taper ces commandes dans la racine du projet à déployer :

`$ git init` // s'il s'agit d'un nouveau projet

`$ heroku login -i` // et entrez vos infos de connexion heroku

`$ heroku git:remote -a nom_du_projet` // crée tout a l'heure

Configuration de la base de données

❖ Postgres

Heroku Postgres est un service de base de données SQL géré fourni directement par Heroku.

Créons une base de données postgres avec la commande :

```
$ heroku addons:create heroku-postgresql:hobby-dev
```

Après l'exécution de cette commande notre DATABASE_URL est créée. On peut le voir en tapant la commande :

```
$ heroku config
```

Et mettre à jour le DATABASE_URL dans le fichier **.env**

Apache pack

Symfony 4 et les applications plus récentes ne contiennent plus de **.htaccess** ou autre configuration pour permettre la réécriture des URL de telle manière qu'elles ne nécessitent pas le script `index.php` dans le chemin.

Afin d'activer rapidement la réécriture pour le serveur Web Apache, vous pouvez installer le pack **symfony/apache-pack**, qui place un fichier `.htaccess` approprié dans votre répertoire public /:

On tape la commande suivante pour l'installer :

```
$ composer require symfony/apache-pack
```

Le fichier Procfile

Pour déployer votre application sur Heroku, vous devez souvent d'abord créer un fichier Procfile, qui indique à Heroku la commande à utiliser pour lancer le serveur Web avec les paramètres corrects. Par défaut, Heroku lancera un serveur Web Apache avec PHP pour servir les applications.

Certaines anciennes versions de Symfony ont parfois créé ce fichier automatiquement, mais vous devez explicitement créer ce fichier Procfile dans le cadre de votre application, ce qui deviendra de toute façon nécessaire une fois que vous aurez commencé à utiliser des types de processus supplémentaires. Il s'agit d'un fichier texte que vous créez dans le répertoire racine de votre application.

Le contenu du Procfile ressemble à ceci:

```
web: vendor/bin/heroku-php-apache2 public/
```

Ou tapez simplement la commande :

```
$ echo 'web: heroku-php-apache2 public/' > Procfile
```

Environnement

Si vous ne configurez pas explicitement l'environnement (dev, prod, etc.) à utiliser, Symfony utilisera, par défaut, l'environnement dev dans les commandes de la console et lors de l'exécution.

Pour que Symfony sache qu'il doit utiliser l'environnement prod à tout moment, il lit à partir de la variable d'environnement `APP_ENV`. Vous pouvez définir des variables d'environnement à l'aide de la fonction de configuration heroku.



Environnement

D'autres variables d'environnement telles que `CORS_ALLOW_ORIGIN`, `APP_SECRET` etc. sont configurables

Vous pouvez déclarer les variables d'environnement en ligne dans la section Settings et aussi en local avec la ligne de commande Heroku.

Exemple :

```
$ heroku config:set APP_ENV=prod
```



Autres paramètres

❖ Pour simplifier les choses et éviter d'éventuelles erreurs fatales, Symfony fournit un outil permettant de vérifier rapidement si votre système répond à certaines exigences. De plus, l'outil fournit des recommandations le cas échéant.

Exécutez cette commande pour installer l'outil:

```
$ composer require symfony/requirements-checker
```

❖ Si vous utilisez jwt dans votre projet ou que des en-têtes http (http headers) sont susceptibles d'être injectées dans vos requêtes, effectuez les modifications suivantes:

Ajoutez ces 3 lignes ci-dessous dans votre fichier **.htaccess** dans la balise `<IfModule mod_rewrite.c>` :

```
SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
```

```
SetEnvIf Content-Type "(.*)" HTTP_CONTENT_TYPE=$1
```

```
SetEnvIf Accept "(.*)" HTTP_ACCEPT=$1
```

Autres paramètres

❖ Dans le cas où l'authentification se fait par JWT, enlevez les extensions **.pem** dans le fichier **.gitignore** pour qu'elles ne soient pas ignorées lors du déploiement. Dans ce cas l'authentification ne marchera pas. Pour cela commentez la ligne suivante :

Comme ceci → `#/config/jwt/*.pem`

❖ Fixtures post-déploiement

Pour pouvoir lancer les fixtures après le déploiement, installer doctrine-fixtures-bundle en enlevant l'option `-dev` :

`$ composer require doctrine/doctrine-fixtures-bundle`

NB : Écrivez vos fixtures de sortes qu'il y'ait au moins un utilisateur disposant de tous les droits et surtout pour l'authentification.

Autres paramètres

❖ Script à exécuter après le déploiement

Vous pouvez créer des scripts qui seront exécutés une fois le déploiement se termine avec succès. Pour cela insérez vos scripts dans le fichier **composer.json** dans la section **post-install-cmd** de la section **scripts** comme ceci :

```
"scripts": {  
    "auto-scripts": {  
        "cache:clear": "symfony-cmd",  
        "assets:install %PUBLIC_DIR%": "symfony-cmd"  
    },  
    "post-install-cmd": [  
        "@auto-scripts",  
        "php bin/console doctrine:schema:update --force",  
        "php bin/console doctrine:fixtures:load --append"  
    ],  
    "post-update-cmd": [  
        "@auto-scripts"  
    ]  
}
```


Déploiement et Tests

❖ Déploiement

Tapez les commandes suivantes pour déployer le projet sur Heroku :

```
$ git add .
```

```
$ git commit -am "your commit message"
```

```
$ git push heroku master
```

Si tout se passe bien vous verrez l'URL de votre application que vous pouvez cliquer directement(ctrl+click)



Déploiement et Tests

❖ Heroku CLI

L'interface de ligne de commande Heroku(CLI) facilite la création et la gestion de vos applications Heroku directement à partir du terminal. C'est une partie essentielle de l'utilisation d'Heroku. Exécutez simplement à la racine du projet :

Pour utiliser le bash, tapez la commande :*

```
$ heroku run bash
```

Et commencez à taper vos commandes

❖ Références :

<https://symfony.com/doc/current/deployment.html#c-install-update-your-vendors>

<https://devcenter.heroku.com/articles/getting-started-with-php>

<https://devcenter.heroku.com/articles/nodejs-support#heroku-specific-build-steps>

<https://api-platform.com/docs/deployment/heroku/>



MERCI...

<https://www.linkedin.com/in/lamine-dabo-39769a1b7/>

https://twitter.com/Ldab_developer

https://www.instagram.com/ldab_dev/