Utiliser Deployer dans la CD

C'est quoi Deployer ?

- C'est un outil écrait en PHP
- Il permet de déployer **automatiquement** des projets PHP (mais pas que !)
- Deployer propose des **recettes** pour les frameworks populaire, dont **Symfony** par exemple

Installer Deployer son poste ?

Avant d'intégrer l'outil dans nos pipelines, testons le en local.

- Installons le produit **globalement** sur notre poste :

```
curl -LO https://deployer.org/deployer.phar
sudo mv deployer.phar /usr/local/bin/dep
sudo chmod +x /usr/local/bin/dep
```

Avec GitLab ou GitHub

- Une grande partie de la configuration de **deployer** est commune à GitLab et GitHub.
- Les différences sont indiquées par l'utilisation de **[GitLab]** ou **[GitHub]** dans le titres des lecçons.

Initialiser Deployer sur le projet

-
```
dep init
  [0 ] Common
  Repository > OK
  Do you confirm? (yes/no) [yes]: no
```
- Nous n'utilisons pas la recette **Symfony**, nous allons écrire la notre 👊
- Un fichier `deploy.php` est créer à la racine de votre projet !

Customiser le déploiement

Nous allons modifier le fichier `deploy.php`

```
// ...
set('git_tty', false);
host('your.host.com')
  ->user('user')
  ->port(22)
  ->set('deploy_path', '/var/www');
// ...
task('deploy', [
    // ...
    //'deploy:vendors', <= on supprime
    'sf:vendors',
    'sf:clear_cache',
    //'sf:migrate', // <= Pour notre exemple, on desactive
]);
```

Customiser le déploiement

```
// (...)
task('sf:vendors', function () {
    run(('cd {{release_path}} && composer install'));
});

task('sf:clear_cache', function () {
    run(('php {{release_path}}/bin/console cache:clear --env=prod'));
});

task('sf:migrate', function () {
    run(('php {{release_path}}/bin/console doctrine:migrations:migrate --env=prod'));
});
```

Mais il est ou mon projet ?

- **Deployer** c'est de la magie 🪄 mais il ce passe quoi ?
  - Création d'un repertoire dans

```
/var/deployer/releases/1 //(puis2, 3, etc ...)
```

  - Création d'un lien **lien symbolique** vers

```
/var/deployer/current/
```

  - Avantages : Préparation de l'app (vendor, cache) **avant mise en production**

Le point configuration ⚠️

Pour que votre serveur web suive le **SymLink**, un peu de configuration à prévoir (exemple pour **Apache**) :

```
<VirtualHost *:80>
    ServerName localhost
    DocumentRoot /var/deployer/current/public
    DirectoryIndex /index.php
    <Directory /var/www/deployer/public>
        Options +Indexes +FollowSymLinks
        AllowOverride None
        Require all granted
        FallbackResource /index.php
    </Directory>
</VirtualHost>
```

Garder des fichiers d'une release à l'autre

- Par defaut, tout le repertoire de l'app est renouvelé, mais il est souaitable de **partager** des repertoires ou des fichiers d'une version à l'autre !

```
set('shared_dirs', ['var/log']);
```

[GitLab] Utiliser deployer

```
deployer:
    stage: deploy
    script:
        - 'which ssh-agent || ( apt-get update -y && apt-get install openssh-client git -y )'
        - eval $(ssh-agent -s)
        - echo "$SSH_PRIVATE_KEY" | tr -d '\r' | ssh-add - > /dev/null
        - mkdir -p ~/.ssh
        - chmod 700 ~/.ssh
        - '[[ -f /.dockerenv ]] && echo -e "Host *\n\tStrictHostKeyChecking no\n\n" >> ~/.ssh/config'
        - |
          curl -LO https://deployer.org/deployer.phar
          mv deployer.phar /usr/local/bin/dep
          chmod +x /usr/local/bin/dep
        - dep deploy -vvv
```

[GitHub] Utiliser deployer

```yaml
name: deploy
on:
  push:
    branches: [ main ]
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Setup PHP
      uses: shivammathur/setup-php@v2
      with:
        php-version: '8.0'
    - name: Deploy
      uses: deployphp/action@v1
      with:
        private-key: ${{ secrets.SSH_PRIVATE_KEY }}
```