Desplegaments PHP

Publicat per Oriol Roselló i Castells [1] el 17/01/2023 - 15:03 | Última modificació: 15/05/2024 - 17:46
Taula de continguts

Per desplegar un projecte PHP utilitzarem preferetment un script bash que haurà de construïr cada projecte segons les seves peculiaritats, el script ha d'estar al repositori de codi GIT del projecte i a totes les branques on calgui utilitzar-lo.

Com a mètode alternatiu es poden fer desplegaments totals amb un .zip de tot el projecte o simplement movent tos els fixers d'un entorn l'altre, però cal conèixer els problemes i les implicacions que te fer-ho així.

Per convenció seguirem els següents procediements preferentment:

1. Repositori GIT

Tot projecte PHP hauria d'utilitzar **GIT** com a repositori de codi i, com a mínim, diposar de dues branques:

- **develop**: es correspon amb el codi que es desplega a l'entorn de **Desenvolupament**.
- **master/main**: es correspon amb el codi que es desplega a l'entorn de **Producció**.

És important que allò que hi hagi al servidor de desenvolupament es correspongui amb una **revisió** del que hi ha a la branca develop del git, i el mateix amb la branca master/main.

Les revisions del git ens donaran una foto d'allò que hi ha desplegat als servidors corporatius exactament, no només del nostre codi sinó també de les versions de les nostres dependències. Si fem, per exemple, un "composer update" directament al servidor sense generar una nova revisió al git del projecte des-sincronitzarem les versions de les dependències, i si desplegem fitxers individuals sense generar revisons no podrem tornar enrrere a revisions concretes si ens cal.

Com a bona pràctica es recomana utlitzar **tags** de tot allò que es desplegui a l'entorn de **Producció**, a ser possible utilitzant un versionat semàntic: X.Y.Z on:

- X: La versió inclou una refactorització important del projecte.
- Y: La versió inclou canvis funcionals.
- Z: La versió només inclou bugfixing.

2. Composer

Els servidors de **Desenvolupament/Integració/Producció** tenen instal·lat de forma global **GIT** i **COMPOSER.**

Git és el repositori de codi que s'utilitzar pels desplegametns del codi, composer és el gestor de dependències que actualitzarà les nostres dependències normalment ubicades a la carpeta de vendors..

Altres utilitats

Si el nostre projecte necesita altres utilitats com drush, yarn, npm, webpack, etc... caldrà que aquestes s'executin localment i les sincronitzem al git o que es puguin executar des d'un vendor del nostre projecte.

Per exemple: drush està disponible a nivell de projecte i no requereix una instal·lació global pot ser executat des del bin o vendor del projecte, així que podem utilitzar drush per netejar la cache d'un projecte Drupal: /usr/bin/php74 -f bin/drush cr o per carregar les configuracions yml de la web: /usr/bin/php74 -f bin/drush cim

En canvi yarn o npm o webpack necesita tenir instal·lat nodejs o altres llibreries a nivell de servidors que no tenim com libsass per compilar SASS.

Així que si el nostre projecte ho necessita ens cal executar-lo localment per generar els fitxers necesaris, en aquest cas normalment es generen els fitxers compilats a public/build i al repositori de codi hauria d'incloure aquests fitxers per poder-ne fer el desplegament, pel que haurem de revisar que la nostre carpeta public/build/ no estigui en el nostre fitxer gitignore.

3. .gitignore

Què ha d'incloure i què no ha d'incloure el nostre fitxer .gitignore.

Si ha d'incloure:

- Les carpetes referents a caches i logs, normalment amb symonfy: /var/
- Les carpetes amb fitxers que puja l'usuari via backoffice, normlamnet amb Symfony: /public/uploads/
- Fitxers i carpets dels ides, tests, logs, etc. Per exemple: .phpunit /.idea
- Variables de configuració del projecte amb informació sensible com paswords de la base de dades, per exemple: /.env.local /.env.local.php /.env.*.local
- Dependències que gestiona el composer, per exemple: /vendor/
- Dependències que gestionem localment amb yarn o npm, per exemple: /node_modules/

No ha d'incloure:

- Fitxers compilats JS/CSS/img que necesitem genrar localment, normalment a Symfony a: /public/build/

3.1. Exemple .gitignore de Symfony

```
/.env.local
/.env.local.php
/.env.*.local
/var/
/vendor/
/.web-server-pid
/node_modules/
npm-debug.log
```

```
yarn-error.log
/.php_cs.cache
/.idea
/.yarn/
.phpunit
/phpunit.xml
.phpunit.result.cache
/public/uploads/
```

4. Fitxers d'instruccions del desplegament

Per convenció crearem un fitxer executable per cada entorn amb les instruccions necesàries que cal realitzar un desplegament a cada un dels entorns amb els següents noms:

/bin/deploy-prd.sh : Desplegament a Producció.
/bin/deploy-pre.sh : Desplegament a Preproducció/Integració
/bin/deploy-dev.sh : Desplegament a Desenvolupament.

Aquests fitxers inclouran la succeció de tasques que cal executar per realitzar un desplegament, que normalemnt inclou: actualitzar el codi, actualitzar les dependències i netejar caches.

Els fitxers estaran al repositori de codi del nostre projecte i serviran com a executable per desplegar i com a documentació per saber la seqüencia de desplegament que requereix aquell projecte.

Per realitzar un desplegament a producció des de plataformes simplement executaran la comanda sh bin/deploy-prd.sh en cada un dels nodes per activar tota la seqüencia de desplegament.

Els desplegaments a desenvolupament i integració els prodrem realitzar nosaltres mateixos.

Es recomana desar a la carpeta **/bin** tots els scripts necesaris que utilitza el projecte per actualitzar, preparar una compilació, desplegar, etc. i documentar-ho tot en un fitxer README.txt o README.md a l'arrel del projecte.

4.1. Exemple: deploy-prd.sh projecte Symfony:

```
#!/bin/bash

echo "===== Iniciant deploy PRD ====="
echo "===== Connectant al repositori de codi ====="
git checkout master
echo "===== Guardant els canvis locals al stash ====="
echo "===== Es pot revertir i recuperar els canvis locals executant: git stash pop"
git stash clear
git stash
echo "===== Baixan't la darrera versió del codi ====="
git fetch origin
git reset --hard origin/master
echo "===== Actualitzant dependències ====="
export COMPOSER_ALLOW_SUPERUSER=1; sudo php74 -d memory_limit=-1 /usr/bin/composer in
stall --prefer-dist --no-interaction --ansi
echo "===== Netejant cache ====="
php74 -f bin/console cache:clear --no-warmup -e prod
echo "===== Corregeix permisos dels fitxers ====="
chown -R apache:apache ./
```

4.2. Exemple: deploy-dev.sh projecte Symfony:

```
#!/bin/bash

echo "===== Iniciant deploy DEV ====="
echo "===== Connectant al repositori de codi ====="
sudo git checkout develop
echo "===== Guardant els canvis locals al stash ====="
echo "===== Es pot revertir i recuperar els canvis locals executant: git stash pop"
sudo git stash clear
sudo git stash
echo "===== Baixan't la darrera versió del codi ====="
sudo git fetch origin
sudo git reset --hard origin/develop
echo "===== Actualitzant dependències ====="
export COMPOSER_ALLOW_SUPERUSER=1; sudo php74 -d memory_limit=-1 /usr/bin/composer in
stall --prefer-dist --no-interaction --ansi
echo "===== Netejant cache ====="
sudo php74 -f bin/console cache:clear --no-warmup -e dev
echo "===== Corregeix permisos dels fitxers ====="
sudo chown -R apache:apache ./
```

4.3. Exemple: deploy-dev.sh projecte Drupal 8:

```
#!/bin/bash

sudo git checkout develop
sudo git stash clear
sudo git stash
sudo git fetch origin
sudo git reset --hard origin/develop
sudo git pull
export COMPOSER_ALLOW_SUPERUSER=1; sudo php74 -d memory_limit=-1 /usr/bin/composer up
date --prefer-dist --with-dependencies --no-interaction --ansi
sudo php74 -f bin/drush cim
sudo php74 -f bin/drush cr
sudo php74 -f bin/drush updb
sudo chmod -R 777 /var/local/html/comunitats/private
sudo chmod -R 777 /var/local/html/comunitats/web/sites/default/files
sudo chown -R apache:apache ./
```

5. Usuari git genèric/anònim

Per tal de poder fer els desplegaments cal accés al repositori GIT del projecte amb permissos de lectura. Hi ha un usuari anònim que només te permisos de consulta, i que és el mateix sistema que utilitza el jenkins per no haver d'utilitzar usuaris personals.

Cal demanar al redmine que el repositori del projecte se li doni accés de lectura a l'usuari genèric. Aquest mateix usuari també ens permetrà la integració amb el Jenkins.

Exemple:

```
Necessitaríem afegir un accés anònim readonly pels repositoris del projecte PROJECTE
tal com s'ha fet a altres projectes (assumptes #12677, #12770, #48436, etc.).
```

Un cop fet això hi hauran dos accessos al repositori de codi, l'accés habitual amb usuari nominal, per exemple:

```
https://codifont.diba.cat/PROJECTE/PROJECTE.git
```

I l'accés anònim que no requereix usuari de la següent forma:

```
https://codifont.diba.cat/PROJECTE-anonym/PROJECTE.git
```

6. Comandes útils

RECORDEU que el git és case-sensitive! Cal possar **EXACTAMENT** les majúscules i minúscules correctes segons com s'hagi creat el repositori. Si per exemple el repositori és: https://codifont.diba.cat/PROJECTE/PROJECTE.git funcionarà si escrius https://codifont.diba.cat/projecte/projecte.git donarà problemes al fer el git pull.

6.1. Vincular una carpeta existent amb el repositori de codi: branca master

```
cd carpeta-projecte
git init
git remote add origin https://codifont.diba.cat/PROJECTE/projecte.git [2]
git pull origin master
git fetch
git checkout master -f
git branch --set-upstream-to origin/master master
```

6.2. Vincular una carpeta existent amb el repositori de codi: branca develop

```
sudo git init
sudo git remote add origin https://codifont.diba.cat/PROJECTE/projecte.git [2]
sudo git pull origin develop
sudo git fetch
sudo git checkout develop -f
sudo git branch --set-upstream-to origin/develop develop
```

6.3. Canviar el repositori d'una carpeta que ja te un vincle git

```
git remote set-url origin https://codifont.diba.cat/PROJECTE-Anonym/projecte.git [3]
```

6.4. Desvincular una carpeta del repositori git sense perdre el contingut

```
rm -rf .git
```

6.5. Esborrar o incorporar els fitxers/carpetes que no estiguin al git

Un cop has vinculada una carpeta amb el repositori de codi pots consultar la llista de fitxers i carpetes diferents entre la carpeta i el repositori de codi amb un:

```
git status
```

Pots decidir esborrar els fitxers/carpetes locals que no estan al git amb:

```
git clean -fd
```

O pots decidir incorporar aquests fixers/carpetes/canvis al teu repositori de codi:

```
git add .
git commit -m "Text dels canvis"
git push
```

6.6. Crear un milestone per desplegar a Producció

Primer cal fer tots els canvis necessaris a la branca "develop". Un cop aprovats i validats tots els canvis preparar la versió per desplegar a Producció. Utilitzarem el --squash si no volem arossegar tots els commit intermitjos realitzats a Desenvolupament o sense aquest si volem incorporar tots els passos intermitjos a la branca.

```
git checkout master
git merge --squash develop
git add .
git commit -m "Versió de producció X.Y.Z"
git push
git tag X.Y.Z
git push origin tag X.Y.Z
```

6.7. Exemple de la preparació d'un desplegament per Producció del Delfos en local: Symfony 5

En aquest projecte s'inclou la instalació d'una llibreria de pagament que està en un zip, la compilació del sass i js amb dependències de llibreries javascript gestiodes pel npm, i generació dels fixers de traducció.

Tota aquesta llista de tasques especials que necesita el projecte per generar una versió desplegable es poden possar, per exemple, en un script de prepare-deploy-prd.sh per no haver-les d'escriure cada cop i documentar-ho en el nostre fitxer README.

```
git checkout master
git merge --squash develop

COMPOSER_MEMORY_LIMIT=-1 composer install --no-interaction --ansi
node node_modules/datatables.net-editor/install.js ./assets/libs/Editor-1.9.7.zip
npm rebuild node-sass
yarn encore production
php bin/console translation:update --dump-messages --prefix="" ca
php bin/console cache:clear --no-warmup -e prod

git add .
git commit -m "Versió de producció X.Y.Z"
git push
git tag X.Y.Z
git push origin tag X.Y.Z
```

Un cop preparat el desplegament amb totes les compilacions locals que es fan amb npm, node, yarn, etc... es pot demanar el desplegament a producció amb l'script de desplegament .

6.8. Actualitzar el servidor de Desenvolupament/Producció/Integració

Per actualitzar, si ja tenim fet el vincle amb el git simplement executarem des de l'arrel del nostre projecte:

```
sh bin/deploy-prd.sh
```

El mateix per la resta d'entorns, etc.

```
sh bin/deploy-dev.sh
```

Si el nostre desplegament inclou canvis en el script de desplegament caldrà executar abans un git pull per actualitzar el fitxer de deploy, o simplement executar el deploy dues vegades. La primera farà el deploy i actualitzarà el script de deploy i la segona executarà el nou script de deploy.

El millor en aquests casos sempre serà fer un git pull abans del deploy per actualitzar el script de deploy i després fer el deploy.

```
git pull
sh bin/deploy-prd.sh
```

- [4]

URL d'origen: https://comunitatdstsc.diba.cat/wiki/desplegaments-php

Enllaços:
[1] https://comunitatdstsc.diba.cat/members/admindiba
[2] https://codifont.diba.cat/PROJECTE/projecte.git
[3] https://codifont.diba.cat/PROJECTE-Anonym/projecte.git
[4] https://comunitatdstsc.diba.cat/node/1408