

Installer, configurer et gérer le sous-domaine "cms"

- **Notes** : ce sous-domaine hébergera le Wordpress temporairement. Il sera à terme basculé sur le domaine principal <domaine-sinp> et le sous-domaine cms.<domaine-sinp> redirigera vers l'interface d'administration de Wordpress.
- **Ressources** :
 - [Wordpress - wp-config.php](#)
 - [How To Install WordPress With Docker Compose](#)
 - [Docker Hub - Wordpress](#) : paramètres et configuration.
 - [Docker Wordpress - Inject configuration using environment variable #142](#)

Installer le domaine

- Créer un fichier de configuration : `vi /etc/nginx/sites-available/cms.conf`
 - Y placer le contenu suivant :

```
server {
    listen 80;
    listen [::]:80;

    server_name cms.<domaine-sinp>;

    location / {
        # ATTENTION : si le header HOST n'est pas renvoyé
        Wordpress utilise l'url 127.0.0.1:50080 pour les fichiers CSS,
        JS...
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $realip_remote_addr;
        proxy_set_header X-Forwarded-Host $host:$server_port;
        proxy_set_header X-Forwarded-Server $host;
        proxy_set_header X-Forwarded-For
        $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        proxy_pass http://127.0.0.1:50080/;# ATTENTION : bien
        mettre un slash final ! Sinon => erreur 404
    }
}
```

- Créer un lien depuis les sites actifs : `cd /etc/nginx/sites-enabled/ ; ln -s ../sites-available/cms.conf cms.conf`
 - Tester la config et relancer Nginx si OK : `nginx-reload` ou `nginx -t && nginx -s reload`
 - Tester l'URL `http://cms.<domaine-sinp>/` qui doit afficher une erreur 502 car nous n'avons pas encore lancé le container Docker.
- En local, sur votre machine, se placer dans le dépôt Github "`docker-paca-sinp`" récupéré

précédemment et si nécessaire resynchroniser le dossier *web-srv* avec le serveur de destination en exécutant la commande *Rsync* indiquée dans le fichier [README.md](#).

- Sur le serveur dans le dossier *docker* de l'utilisateur *admin* :
 - vérifier la présence du réseau Docker spécifique à notre utilisation de type *bridge* nommé *nginx-proxy* (voir fichier *.env*) : `docker network ls`
 - se placer dans le dossier *cms.silene.eu* : `cd ~/docker/cms.silene.eu`
 - exécuter la commande : `docker-compose up`
 - vérifier que tout fonctionne à l'adresse : <http://cms.<domaine-sinp>> (l'installateur Wordpress doit s'afficher)
 - arrêter le container : CTRL+C
 - relancer le container en tant que service : `docker-compose up -d`
 - si besoin de l'arrêter utiliser : `docker compose down`

Activer le SSL

- Installer un certificat SSL via *Certbot (Letsencrypt)* : `certbot --nginx -d cms.<domaine-sinp>`
 - Répondre : 2
 - Tester ensuite la redirection auto de HTTP vers HTTPS : `http://cms.<domaine-sinp>/` → doit rediriger vers HTTPS automatiquement
- Tester la configuration SSL : <https://www.ssllabs.com/ssltest/analyze.html?d=cms.<domaine-sinp>>
- Tester l'URL `https://cms.<domaine-sinp>/`
- La configuration finale :

```
server {
    listen 443 ssl; # managed by Certbot
    listen [::]:443 ssl; # managed by Certbot

    server_name cms.<domaine-sinp>;

    location / {
        # ATTENTION : si le header HOST n'est pas renvoyé Wordpress
        # utilise l'url 127.0.0.1:50080 pour les fichiers CSS, JS...
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Host $host:$server_port;
        proxy_set_header X-Forwarded-Server $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        proxy_pass http://127.0.0.1:50080/; # ATTENTION : bien mettre un
        slash final ! Sinon => erreur 404
    }

    ssl_certificate /etc/letsencrypt/live/cms.<domaine-sinp>/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/cms.<domaine-sinp>/privkey.pem; # managed by Certbot
}
```

```
sinp>/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by
Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {
    listen 80;
    listen [::]:80;

    server_name cms.<domaine-sinp>;

    return 302 https://$host$request_uri;
}
```

Note sur Docker et Wordpress

- **Ressources :**
 - [Doc PhpMailer](#)
 - [Discussion Github sur Docker Wordpress et la config d'envoi d'email](#)
 - [Site d'exemple pour l'utilisation du Hook phpmailer_init](#)
- Par défaut, le docker Wordpress n'est pas configuré pour envoyé des emails. Pour que l'envoi d'email fonctionne aussi bien dans les extensions que dans le coeur de Wordpress, il a été nécessaire d'installer l'extension **WP Mail SMTP**.
 - Les tentatives suivantes ont été réalisés :
 - configuration de sendmail (installation, lancement auto, modif php.ini) dans l'image via un DockerFile ⇒ fonctionne dans le système du container mais pas dans Wordpress.
 - utilisation du hook `phpmailer_init` dans un plugin pour forcer l'utilisation du SMTP Google ⇒ fonctionne dans les plugins mais pas dans le coeur de Wordpress (les notifications de nouveau utilisateur par exemple ne sont pas envoyée).
 - utilisation d'autres plugin de config SMTP plus simple que *WP Mail SMTP* ⇒ ne fonctionne pas... A priori, *WP Mail SMTP* n'utilise pas le hook mais surcharge le variable globale `$phpmailer` avec son propre code.
 - Les fichiers utilisés pour les différentes tentatives sont dans le dépôt Git SINF-PACA.

Configuration du Wordpress sur le sous-domaine "cms"



ATTENTION : activer l'indexation par les moteurs de recherche juste avant la mise en production

- Installateur Wordpress :
 - *Choisir la langue* : Français
 - *Titre du site* : SILENE
 - *Identifiant* : votre identifiant (ne pas utiliser de compte standard - ex. admin - pour éviter

- les piratages).
- *Mot de passe* : voir Keepass
- *Votre adresse de messagerie* : votre email pro
- *Visibilité pour les moteurs de recherche* : cocher la case pour éviter l'indexation pendant la phase de mise au point...
- "Extensions" > "Extensions installées"
 - Supprimer toutes les extensions par défaut
- "Extensions" > "Ajouter"
 - Ajouter l'extension [WP Mail SMTP](#) et l'activer
 - Cliquer sur "Réglages" :
 - *Adresse e-mail d'envoi* : mailer@cbn-alpin.fr
 - Cocher *Forcer l'e-mail d'expédition*
 - *Nom de l'expéditeur* : Adminsys
 - Autre SMTP:
 - *Hébergeur SMTP* : smtp-relay.gmail.com
 - *Cryptage* : TLS
 - *Port SMTP* : 587
 - *Identifiant SMTP* : mailer@cbn-alpin.fr
 - *Mot de passe SMTP* : générer un mot de passe d'application sur le compte mailer@cbn-alpin.fr et l'utiliser ici.
 - Tester l'envoi d'email avec l'onglet correspondant
 - Ajouter l'extension [WP Matomo](#) et l'activer
 - Cliquer sur "Réglages" :
 - *Se connecter à Matomo* :
 - Mode de Matomo : Auto-hébergé (API HTTP par défaut)
 - URL de Matomo: <https://analytics.<domaine-sinp>/>
 - Jeton d'authentification à l'API : récupérer le jeton sur l'outil [Matomo installé](#), menu "Paramètres > Plateforme > API".
 - Cocher *Auto configuration*
 - Site déterminé : cms-<region>-sinp (<https://cms.<domaine-sinp>>)
 - *Afficher les statistiques* :
 - Date par défaut de Matomo : Aujourd'hui
 - Vue d'ensemble Matomo sur le Tableau de bord : Aujourd'hui
 - Cocher *Graphique Matomo sur le Tableau de bord*
 - Afficher les statistiques pour : Administrator, Editor.
 - Cocher *Afficher les statistiques par article*
 - Cocher *Raccourci Matomo*
 - Nom affiché de WP-Matomo : WP Matomo
 - Cocher *Activer les codes courts*
 - *Activer le suivi* :
 - Ajouter le code de suivi : code de suivi par défaut
 - Position du code JavaScript : entête
- "Réglages" > "Général" :
 - *Slogan* : Système d'Information et de Localisation des Espèces Natives et Envahissantes
 - *Adresse e-mail d'administration* : adminsys@<domaine-sinp>
 - *Rôle par défaut de tout nouvel utilisateur* : contributeur
- "Réglages" > "Discussions" :
 - Décocher la case *Autoriser les notification de lien en provenance d'autres blogs (pings et rétroliens) sur les nouvelles publications*

- Décocher la case *Autoriser les commentaires sur les nouvelles publications*
- "Apparence" > "Personnaliser" > "Identité du site" :
 - *Icône du site* : charger le fichier favicon.png
- "Apparence" > "Thèmes" :
 - Supprimer tous les thèmes inutilisés (passer par "Détails" pour accéder à l'action "Supprimer")
- "Outils" > "Santé du site" :
 - Doit afficher : *Bon travail ! Tout fonctionne parfaitement ici.*

Sauvegarde et restauration de Wordpress



Le container blacklabelops/volumerize n'est plus maintenu... voir à terme pour son remplacement si nécessaire

- La sauvegarde et la restauration des fichiers et base de données de Wordpress utilise le container [blacklabelops/volumerize](#).
- Via Docker-Compose le lancement du container ne semble pas générer les fichiers nécessaire pour mettre en place le système Cron utilisant [Jobber](#). Pour l'activer, il faut :
 - Se connecter au container : `docker exec -it cms-volumerize /bin/bash`
 - Se placer dans le dossier suivant : `cd /opt/volumerize`
 - Ajouter la ligne : `CUR_DIR='/opt/volumerize` après la première ligne du fichier : `vi /opt/volumerize/create_jobber.sh`
 - Lancer le script : `./create_jobber.sh`
 - Vérifier la présence du fichier : `vi /root/.jobber`
 - Recharger les jobs de tous les utilisateurs : `jobber reload -a`
 - Vérifier la présence des jobs : `jobber list`
 - Tester un job : `jobber test <nom-du-job>`
- Les sauvegardes sont :
 - exécutées automatiquement chaque jour à 2 heures du matin
 - glissantes sur 7 jours
 - incrémentales sur 6 jours
 - complètes tous les 7 jours
 - stockées dans le dossier `/home/admin/docker/cms.<domaine-sinp>/backup`
- Backend de sauvegarde Dropbox : <https://github.com/blacklabelops/volumerize/tree/master/backends/Dropbox>
- Le fichier `docker-compose.yml` principal contient le service `cms-volumerize`. Ce service lance le container [blacklabelops/volumerize](#) qui contient l'outil Duplicity. Ce dernier se charge des sauvegardes et restauration.
- Pour réaliser une restauration, il faut utiliser le fichier `docker-compose.restore.yml` en suivant les étapes suivante :
 - Arrêter le service `cms-volumerize` : `docker-compose stop cms-volumerize`
 - Vérifier les différences entre les données actuelles et le dernier point de restauration : `docker-compose -f docker-compose.restore.yml run cms-volumerize-restore-from-local verify`
 - Pour vérifier avec un point plus ancien, utiliser l'option `-t` et [un temps](#)
 - Exécuter une restauration avec le dernier point de restauration local : `docker-compose -f docker-compose.restore.yml run cms-volumerize-restore-from-local restore`

- Pour restaurer à un point plus ancien, utiliser l'option `-t` et [un temps](#). Exemple pour restaurer l'état 3 jours avant : `docker-compose -f docker-compose.restore.yml run cms-volumerize-restore-from-local restore -t 3D`
- Si nécessaire, redémarrer les services restaurer : `docker-compose restart cms-nginx cms-wordpress cms-mariadb cms-adminer cms-volumerize`
- Vérifier que la restauration est bonne
- Voir [la documentation pour restaurer depuis Dropbox](#)
 - Ex. : `docker-compose -f docker-compose.restore.yml run cms-volumerize-restore-from-dropbox restore`

From:
<https://wiki-sinp.cbn-alpin.fr/> - **CBNA SINP**

Permanent link:
<https://wiki-sinp.cbn-alpin.fr/serveurs/installation/web-srv/docker-wordpress?rev=1582894977>

Last update: **2020/02/28 13:02**

