Civiparoisse pour les Devops

(Méthodologie, dépôts, opérations courantes)

Union des Églises protestantes d'Alsace et de Lorraine

Méthodologie approximativement employée

- Basée sur gitflow : https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow
- Dépôts git privés
- branche main
- branche développement
- branches de fonctionnalités
- Branches de hotfix
- Spécifique au projet : revue de code obligatoire
- Tags - nom du tag=version : numérotation sémantique : vMajeur.Mineur.Correctif

Git : opérations courantes

- Authentification : la plupart du temps par clef SSH
- Récupérer une copie d'un dépôt : git clone
- Aller sur une branche : git checkout (création de la branche : -b)
- Voir l'état du courant : git status
- Ajouter des fichiers au stage : git add
- Supprimer des fichiers : git rm
- Renommer des fichiers : git mv
- Commiter : git commit -m 'message'
- Pousser les modifications vers la branche d'upstream : git push
- Fusionner des branches : git merge / git mergetool
- Lister/ créer des tags : git tag
- Voir les différences : git log -p / git difftool
- Voir les différences entre branches :
  git log --stat branche1..branche2

Dépôts non Dev

- Dépôts d'exploitation :
  - déploiement continu
  - instances civiparoisses
- Dépôt transversal :
  - la documentation
- Dépôts d'expérimentation

Dépôts Dev

- extension civiparoisse :
  - adaptations principales
  - exemple test unitaire phpunit

- image Docker civiparoisse :
  - génération de l'image Civiparoisse et image de debug (avec nano, less, mc et xdebug)
  - chart Helm (avec fichier de valeur pour un minikube en local)

Dépôt Intégration continue

- fichiers de configuration pour Github Actions
- nécessaire pour une exploitation Dev en docker-compose (avec mapping ports 443 et 3306 sur 127.0.0.1)
- répertoire de secrets pour le dev
- scripts pour arriver à l'état initial "Golden Database"
- certificat SSL pour test (et un script shell pour le regénérer) - nom : civiparoisseci.test
- exécution de tests d'acceptation
- génération de la documentation screenshot

Introduction Docker

- Equivalent à un chroot survitaminé
- Distinction image/container
- Build d'images
- Registre d'images
- Opérations courantes :
  - docker pull <image>
  - docker container ls (-a)
  - docker container rm <container>
  - docker container stop <container>
  - docker image ls
  - docker image rm <image>
  - docker logs (-f) <container>
  - docker build -t <tag> (-f <Dockerfile>) <context reference>

Exemple de Dockerfile

```
FROM uepal_test/civiparoisseci:citest
USER root
RUN apt-get update; \
    apt-get install -y nano less mc ; \
    rm -rf /var/lib/apt/lists/*;
RUN pecl install xdebug
RUN docker-php-ext-enable xdebug
COPY debug/99-xdebug.ini /usr/local/etc/php/conf.d/
USER civiparoisse
```

- Pour builder l'image :
  docker build -t uepal_test/civiparoisseci:citest .
  docker build -t uepal_test/civiparoisseci:debug -f ./Dockerfile_debug .

Extrait de log de build

```
Sending build context to Docker daemon  4.073MB
Step 1/7 : FROM uepal_test/civiparoisseci:citest
 ---> d2accfc8c97a
Step 2/7 : USER root
 ---> Running in ea6c956632f2
Removing intermediate container ea6c956632f2
 ---> 91a273955518
Step 3/7 : RUN apt-get update;     apt-get install -y nano less mc ;     rm -rf /var/lib/apt/lists/*;
 ---> Running in d780982230e7
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
...
Step 5/7 : RUN docker-php-ext-enable xdebug
 ---> Running in ca3a51aee27f
Removing intermediate container ca3a51aee27f
 ---> 6191491858e4
Step 6/7 : COPY debug/99-xdebug.ini /usr/local/etc/php/conf.d/
 ---> 86d852d82e72
Step 7/7 : USER civiparoisse
 ---> Running in e376b2816350
Removing intermediate container e376b2816350
 ---> 51ad9511f0f5
Successfully built 51ad9511f0f5
Successfully tagged uepal_test/civiparoisseci:debug
```

Extrait de Docker-compose.yaml

```
services :
  php_webserver:
    container_name: ci_php_webserver
    image: "${SUTIMAGE:-uepal_test/civiparoisseci:citest}"
    entrypoint: ''
    links:
      - db
    environment:
      - SITENAME=NomDuSite
      - BASE_URL=https://civiparoisseci.test
      - MAILADMIN_ADDR=admin@example.org
      - MAILSITE_ADDR=contact@example.org
      - MAILADMIN_DESCR=Addresse mail du compte admin
      - PAROISSE_PHONE=0388000000
    volumes:
      - ./secrets:/var/run/secrets:ro
      - type: volume
        source: ci_filevol
        target: /app/web/sites/default
      - type: volume
        source: ci_privatevol
        target: /app/private
      - ./tls/tls.key:/certificates/tls.key:ro
      - ./tls/tls.crt:/certificates/tls.crt:ro
      - ./system_config:/system_config:ro
    command: ["/bin/sh","-c","apache2-foreground"]
    ports:
      - 127.0.0.1:443:8443
    depends_on:
      - db

  db:
    command: --sql-mode="STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION"
    container_name: ci_db
    image: mysql
    restart: always
    ports:
      - 127.0.0.1:3306:3306
    environment:
      MYSQL_ROOT_PASSWORD: root
    volumes:
      - type: volume
        source: ci_dbvol
        target: /var/lib/mysql
        volume:
          nocopy: true
      - type: bind
        source: ./initdb
        target: /docker-entrypoint-initdb.d
    healthcheck:
      test: ["CMD","mysqladmin","-h","127.0.0.1","ping"]
      interval: 1m
      timeout: 30s
      retries: 3
      start_period: 2m
volumes:
  ci_dbvol:
  ci_privatevol:
  ci_filevol:
```

Local.sh

```
#!/bin/bash
#script pour l'exécution locale (chez un dev)
#il faut avoir fait le taggage de l'image Docker
#et avoir mis la résolution de nom en place
#avant de lancer le script

#on nettoie
docker-compose down -v
rm -Rf vendor
rm -f composer.lock
rm -Rf tests/_output
pkill chromedriver
```

```
#et on lance
docker-compose up -d db && bash isupdb.sh
docker-compose up php_initialize
docker-compose up php_update
docker-compose up php_configure
docker-compose up -d php_webserver
composer upgrade
chromedriver --port=9515&
./vendor/bin/codecept run -vvv --steps
pkill chrom
```

Se préparer un environnement de développement (non debug) avec Docker : partie docker

- récupérer les dépôts de l'image Civiparoisse et de l'intégration continue
- dans le répertoire de l'image civiparoisse : builder l'image :
  docker build -t uepal_test/civiparoisseci:citest .
  - vérifier si l'image est bien présente : docker image ls
  (- supprimer une ancienne image : docker rmi <id image>)
- dans le répertoire de l'intégration continue :
  - créer une copie du fichier docker-compose.yaml : ex : docker-compose.yaml_dev
  - modifie la copie pour rajouter un montage de volume :
    sections "volumes":
    - <chemin local>:<chemin dans l'image>
- exécuter les commandes les unes après les autres :
  docker-compose -f docker-compose.yaml_dev up -d db && bash isupdb.sh
  docker-compose -f docker-compose.yaml_dev up php_initialize
  docker-compose -f docker-compose.yaml_dev up php_update
  docker-compose -f docker-compose.yaml_dev up php_configure
  docker-compose -f docker-compose.yaml_dev up -d php_webserver

- modifier son fichier hosts local pour ajouter la résolution du nom :
  127.0.0.1 civiparoisseci.test

Se préparer un environnement de développement (non debug) avec Docker : partie IDE

- Récupérer depuis l'image le contenu du répertoire /app : docker cp ci_php_webserver:/app .
- Dans la copie locale, supprimer les répertoires correspondants aux extensions que l'on souhaite développer, pour les remplacer par un lien vers le dépôt de travail (ex : /app/ext/fr.uepalparoisse.civiparoisse) ; si nécessaire, ajuster les droits des fichiers
- Utiliser la copie locale dans l'IDE
- Si nécessaire, configurer l'IDE pour indiquer les extensions PHP qui doivent être disponibles
- Optionnel : configurer un accès à la BD pour l'IDE

Xdebug

- Préparé dans la branche SOLR de l'image Civiparoisse
- Nécessite d'utiliser l’image de debug : remplacement de l’image dans les fichiers docker-compose.
- IP "variable" du container, d’où connection issue du container vers l’IDE (cible : port TCP 9003 – Xdebug v3), grâce à l’injection de la résolution du nom de l'hôte docker (nom spécial host-gateway) :

```
extra_hosts:
- "host.docker.internal:host-gateway"
```

- Configurer l'IDE pour Xdebug : configurer le mapping entre le dossier app "local", et le "/app" dans le container, configurer le nom de serveur, lancer l’attente des connexions entrantes sur le port 9003
- Nécessite d’ouvrir le firewall local pour autoriser les connexions

Opérations courantes

- Commandes usuelles : ls, cd, find, grep, cat, more, less, mc, nano, meld
- Se connecter dans le container local (user civiparoisse) :
  docker exec -ti ci_php_webserver bash
- Se connecter dans le container local (user root) :
  docker exec -ti -u 0 ci_php_webserver bash
- Publier les fichiers (à cause du scaffolding) (root)
  composer -d /app civicrm:publish
- Flusher les caches :
  drush –root=/app cr
  cv --cwd=/app -U ciadmin flush
- Exécuter le cron :
  cv -U ciadmin api job.execute --cwd=/app -vv
  drush --root=/app --quiet cron -vv
- Se connecter à la base mysql :
  cv --cwd=/app sql

Opérations moins courantes

- Voir les extensions locales :
  cv -U ciadmin ext:list -L

- Installer civix :
```
cd /tmp
mkdir CIVIX
cd CIVIX
echo
'{"require":{"civicrm/civix":"*"},"minim
um-stability":"dev","prefer-
stable":true }' >composer.json
composer upgrade
cd vendor/bin
export PATH=$PATH:`pwd`
cd /app
civix list
civix civicrm:ping
```

- Installer civistrings :
```
cd /tmp
mkdir CIVISTRINGS
cd CIVISTRINGS
echo
'{"require":{"civicrm/strings":"*"},"min
imum-stability":"dev","prefer-
stable":true }' >composer.json
composer upgrade
cd vendor/bin
export PATH=$PATH:`pwd`
cd /app
```

Intérêt de Minikube

- Intérêt de Minikube :
  - capable de fournir un cluster Kubernetes dans un container Docker
  - Notion de profil : configuration par container
  - bon environnement pour apprendre
  - dispose de plugins intéressants (dont le dashboard)

- Remontage en local : https://uepal-civiparoisse-documentation.github.io/TECHNIQUE/POC/remontage_local.html

Prise en main de Minikube

- disposer de Docker (pour utiliser Minikube en container)
- disposer d'une machine avec beaucoup de RAM
- disposer de suffisamment de stockage rapide
- récupération depuis https://github.com/kubernetes/minikube/releases
- minikube start [-p <nom profil>] [--cpus= <#cpu>][--memory= <#ram>]
- minikube stop [-p <nom profil>]
- minikube delete [-p <nom profil>]
- minikube addons list
- minikube addons enable dashboard
- minikube dashboard
- minikube ip
- minikube docker-env

Minikube : quelques petits points problématiques

- stockage persistant provisionné : faire le ménage après utilisation (minikube ssh, et nettoyer /var/hostpath-provisionner)
- drivers de stockage de hostpath et métriques d'espace disque
- des alarmes peuvent survenir avec Kubestack : configuration spéciale lors de la création du cluster

Prise en main de kubectl

- gestion des manifestes (déclaration de ressources en yaml)
- minikube [-p <nom profil>] kubectl -- <arguments kubectl>
- kubectl api-resources
- kubectl get
- kubectl describe
- kubectl apply -f
- kubectl delete -f
- kubectl create
- kubectl logs
- kubectl exec

Conseils pour la préparation des manifestes

- utiliser un IDE qui a un support de Kubernetes
- récupérer tous les CRDs spécifiques via du get -o yaml, et les mettre à disposition de l'IDE
- utiliser l'autocomplétion
- considérer les infobulles
- valider progressivement le code : indentation YAML, contenu de l'objet grâce au CRD, exécution en dry-run

Helm : description

- packaging de manifestes K8S sous forme de charts
- fonctions utilisées pour le templating ; variables (.Values),constantes (.Chart), infos de déploiement (.Release)
- fichiers de valeurs (par défaut et spécifique)
- plusieurs charts spécifiques sur le projet, dont Civiparoisse
- dépendance de Civiparoisse sur Traefik et Cert-manager.

Helm : opérations courantes

- helm repo add <url du dépôt>
- helm repo update
- helm lint <chemin>
- helm (un)install [--values <fichier de valeurs>] [-n <namespace>] [--create-namespace] <nom-release> <identification-chart>
- helm upgrade [--values <fichier de valeurs>] [-n <namespace>] <nom-release> <identification-chart>
- helm get all [-n <namespace>] <nom-release>

Interactions entre Docker et Minikube : principe

- But : pouvoir accéder aux ressources Kubernetes qui sont dans le container minikube depuis un autre container Docker (ex d'utilisation : développement avec un SOLR embarqué dans K8S).
- Minikube crée un réseau Docker qui a pour nom le nom du profil (minikube par défaut) ; il prend l'IP en .2 pour le container, et la .1 pour l'hôte.
- Il faut connecter le container Docker à ce réseau, et prendre une IP en .3 dans le réseau
- Il faut ajouter dans la résolution de noms du container les noms avec l'IP du .2

Interactions entre Docker et Minikube : pratique

- Ajout du réseau minikube au niveau top level :

```
networks:
  minikube:
    driver: bridge
    external: true
```

- Ajout du réseau minikube dans le service php_webserver : ex : pour un réseau 192.168.58.0/24 (extraits) :

```
  php_webserver:
    extra_hosts:
     - "civisolr.solr:192.168.58.2"
     - "host.docker.internal:host-gateway"
    networks:
      default: {}
      minikube:
        ipv4_address: 192.168.58.3
    links:
      - db
```

Et la suite ?

- Documentation : https://uepal-civiparoisse-documentation.github.io/
- Projet d'équipe, qui doit s'étoffer pour que le projet perdure
- **Venez mouiller le maillot avec nous !**

Union des Églises protestantes d'Alsace et de Lorraine