Quiz Sprint 5 :

Can You See ME :

Exploration d'une extension PHP inconnue

Réalisé par :

Sara Akharraz

Encadrants :

M.Abdelmajid Bendrif

M.Hamza Bahlaouane

Entreprise :

Void

Année universitaire 2025-2026

Objectif :

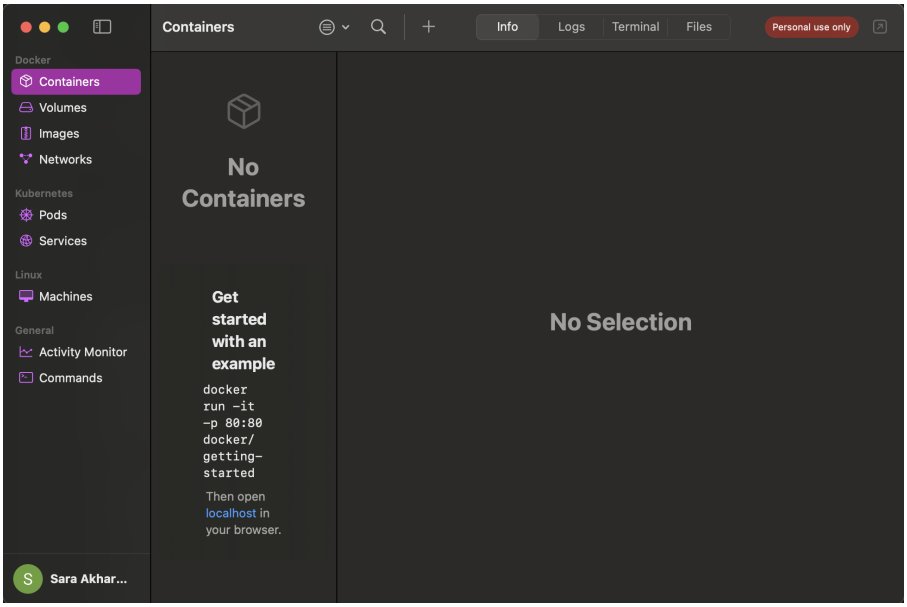
— Découvrir l'écosystème PHP et ses extensions.

Prérequis :

Orbstack :

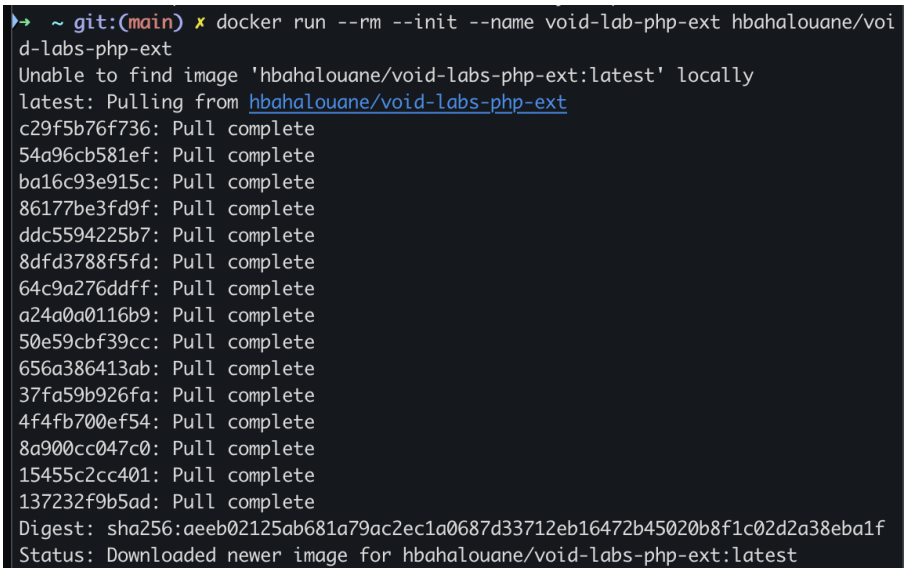
Un outil de virtualisation pour macOS qui permet d'exécuter facilement des conteneurs Docker et des machines virtuelles de manière rapide et légère pour le développement.

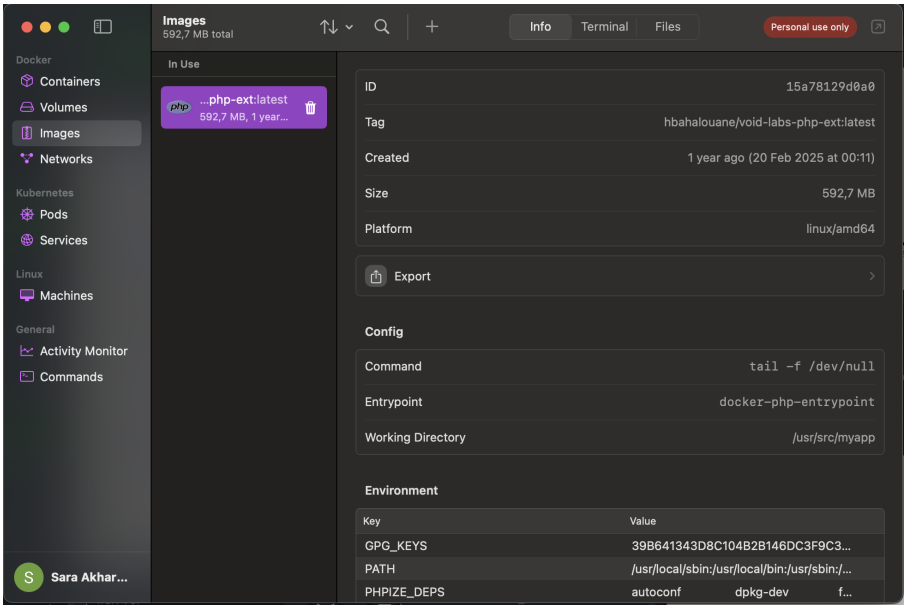
Déjà installé lors du premier sprint (setup).



Lancement du conteneur Docker :

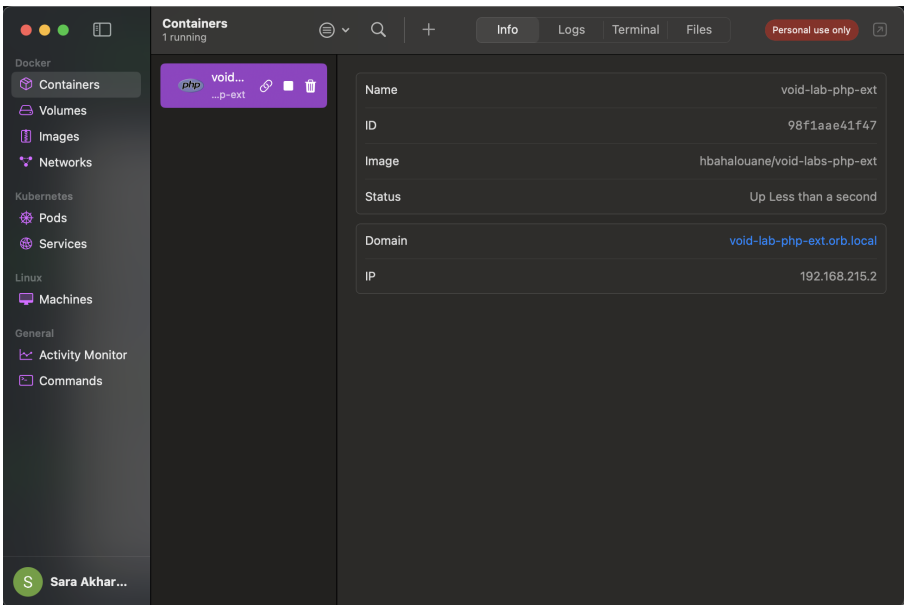
```
docker run --rm --init --name void-lab-php-ext hbahalouane/void-labs-php-ext
```

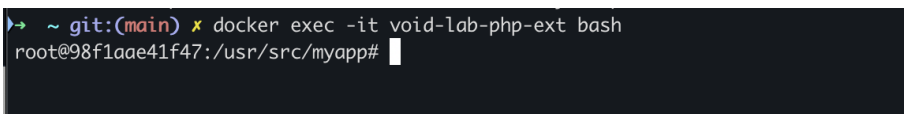




Connexion via SSH :

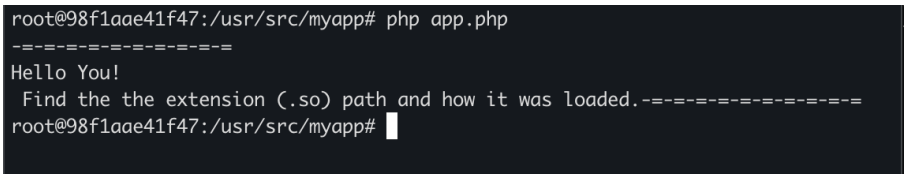
```
docker exec -it void-lab-php-ext bash
```





Exécutons le script PHP :

```
php app.php
```



Analyse du comportement du script :

```
docker exec -it void-lab-php-ext bash
<?php

echo str_repeat("-=", 10) . "\n";

\VoidLabs\canYouSeeMe();

echo str_repeat("-=", 10) . "\n";
~
~
~
~
```

La première question à poser est : D'où vient `\VoidLabs\canYouSeeMe()` ?

NB :
Une fonction en PHP vient de 3 endroits seulement :

1. Code PHP (cœur du langage et code applicatif)
   - `Fonctions natives intégrées au cœur de PHP (ex : isset(), empty(), etc.)`
   - `Code écrit dans des fichiers .php (app.php, src/, etc.)`
2. Dépendances Composer (`composer require`)
   - `Bibliothèques tierces installées dans le dossier vendor/`
3. Extensions PHP compilées
   - `Modules chargés automatiquement au démarrage de PHP (ex : PDO, MySQLi...)`

1- Est-ce dans le code PHP ?
On test par les commandes suivantes :

```
find / -name "*VoidLabs*"
find / -name "*canYouSeeMe*"
grep -r "canYouSeeMe" .
```

```
root@98f1aae41f47:/usr/src/myapp# find / -name "*VoidLabs*"
root@98f1aae41f47:/usr/src/myapp# find / -name "*canYouSeeMe*"
root@98f1aae41f47:/usr/src/myapp# grep -r "canYouSeeMe" .
./app.php:\VoidLabs\canYouSeeMe();
root@98f1aae41f47:/usr/src/myapp#
```

Résultat : rien trouvé !

2- Est-ce une dépendance Composer ?

```
ls -la vendor/
```

```
root@98f1aae41f47:/usr/src/myapp# ls -la vendor/
ls: cannot access 'vendor/': No such file or directory
root@98f1aae41f47:/usr/src/myapp#
```

Résultat : /vendor n'existe meme pas !

Puisque ce n'est ni du code PHP ni une dépendance Composer, il s'agit d'**une extension PHP**.

Une extension PHP est du code compilé en C/C++ qui étend les capacités de PHP.

Approche 01 : php -m

NB :

Où les extensions sont activées ? :

- PHP charge toujours en priorité les extensions compilées dans son cœur (date,json ...).
- Les extensions supplémentaires sont activées dans le fichier php.ini.

check this (Extensions PHP : Activer une extension)

Dans notre cas :

```
php --ini
```

```
root@98f1aae41f47:/usr/src/myapp# php --ini
Configuration File (php.ini) Path: /usr/local/etc/php
Loaded Configuration File:         (none)
Scan for additional .ini files in: /usr/local/etc/php/conf.d
Additional .ini files parsed:      /usr/local/etc/php/conf.d/docker-php-ext-sample.ini,
/usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
```

Le fichier php.ini n'existe pas , donc les extensions actives proviennent du cœur de PHP et des fichiers **.ini** .

```
ls /usr/local/etc/php/conf.d/
cat /usr/local/etc/php/conf.d/docker-php-ext-sample.ini
cat /usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
```

```
root@98f1aae41f47:/usr/src/myapp# ls /usr/local/etc/php/conf.d/
docker-php-ext-sample.ini  docker-php-ext-sodium.ini
root@98f1aae41f47:/usr/src/myapp# cat /usr/local/etc/php/conf.d/docker-php-ext-sample.ini
extension=sample
root@98f1aae41f47:/usr/src/myapp# cat /usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
extension=sodium
root@98f1aae41f47:/usr/src/myapp#
```

On liste toutes les extensions chargées par PHP au démarrage :

check this : How do I see the extensions loaded by PHP?

On utilise la commande suivante :

```
php -m
```

```
root@98f1aae41f47:/usr/src/myapp# php -m
[PHP Modules]
Core
ctype
curl
date
dom
fileinfo
filter
hash
iconv
json
libxml
mbstring
mysqlnd
openssl
pcre
PDO
pdo_sqlite
Phar
posix
random
readline
Reflection
sample
session
SimpleXML
sodium
SPL
sqlite3
standard
tokenizer
xml
xmlreader
xmlwriter
zlib

[Zend Modules]

root@98f1aae41f47:/usr/src/myapp#
```

NB :

Les extensions compilés dans le coeur PHP sont :

Core, ctype, curl, date, dom, fileinfo, filter, hash, iconv, json, libxml, mbstring, mysqlnd, openssl, pcre, Phar, posix, random, readline, Reflection, SPL, standard, tokenizer, xml, xmlreader, xmlwriter, zlib

celles du fichier .ini : sample,sodium

Or D'où vient PDO ,pdo_sqlite ,sqlite3 ?

Ces extensions sont incluses par défaut dans l'image Docker PHP.

```
php -i | grep -i pdo
```

Puisque on a bien :

```
PDO
PDO support => enabled
PDO drivers => sqlite
pdo_sqlite
```

Et il n'y a aucune référence à un fichier .ini : c'est compilé dans le binaire de l'image.

Au contraire :

```
php -i | grep -i sample
```

Les extensions peuvent donc provenir des extensions compilées dans PHP, des fichiers .ini, ou être incluses directement dans l'image Docker (dans le cas d'un container).

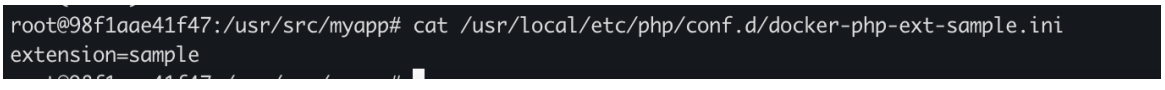
Quelle est l'extension responsable de l'ajout de \VoidLabs \canYouSeeMe() ?

On vérifie l'extension sample :

Pourquoi sample ?

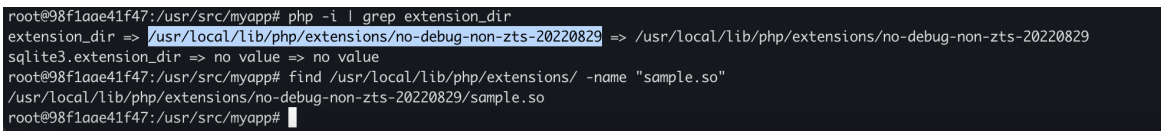
Elle n'est pas standard (une extension custom que Docker a ajoutée)

```
cat /usr/local/etc/php/conf.d/docker-php-ext-sample.ini
```



Pour trouver l'emplacement du fichier **sample.so**, on utilise **extension_dir** (c'est le répertoire qui indique à PHP où trouver les extensions).

```
php -i | grep extension_dir
find /usr/local/lib/php/extensions/ -name "sample.so"
```



La commande strings permet d'extraire le texte lisible d'un fichier binaire :

```
strings /usr/local/lib/php/extensions/no-debug-non-zts-20220829/sample.so | grep VoidLabs
```

```
root@98f1aae41f47:/usr/src/myapp# strings /usr/local/lib/php/extensions/no-debug-non-zts-20220829/sample.so | grep VoidLabs
VoidLabs\canYouSeeMe
root@98f1aae41f47:/usr/src/myapp#
```

Approche 02 : Réflexion

La Réflexion PHP permet d'inspecter le code PHP à l'exécution.

check this :How to find out where a function is defined?
J'ai crée un fichier test_reflection.php dans le container

```
touch test_reflection.php
```

Puis, j'ai utilisé la même fonction que celle présentée dans le lien ci-dessus, en y ajoutant des logs :

```php
<?php

// Test Reflection sur la fonction VoidLabs

$reflFunc = new ReflectionFunction('\VoidLabs\canYouSeeMe');

echo "=== Analyse avec Reflection ===\n";
echo "Fichier: " . $reflFunc->getFileName() . "\n";
echo "Ligne: " . $reflFunc->getStartLine() . "\n";
echo "Extension: " . $reflFunc->getExtensionName() . "\n";
echo "=== Fin ===\n";
~
~
```

on exécute le fichier :

```
php test_reflection.php
```

```
root@98f1aae41f47:/usr/src/myapp# php test_reflection.php
=== Analyse avec Reflection ===
Fichier:
Ligne:
Extension: sample
=== Fin ===
```

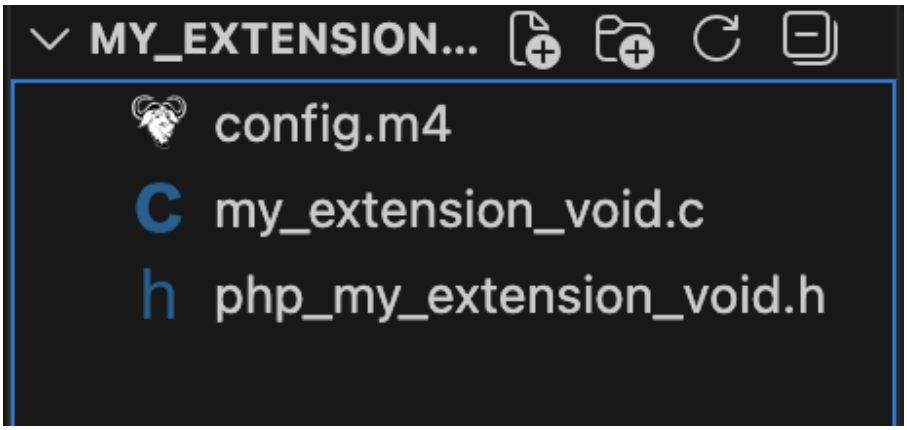
Bonus

Si vous deviez créer une extension PHP similaire, comment procéderiez-vous ?

1. **Étape 1 :** Préparer la structure de l'extension

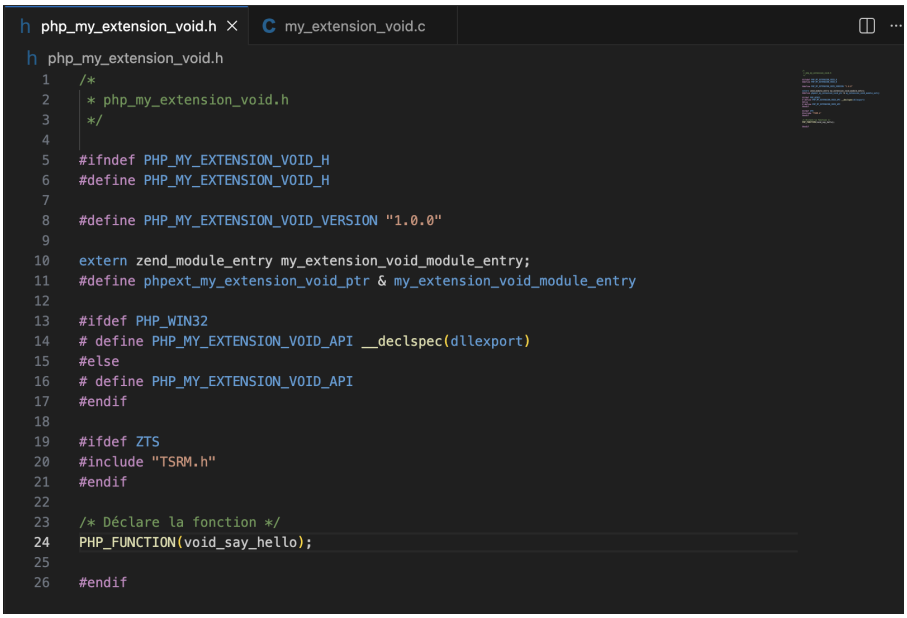
   On crée le répertoire et les fichiers de base pour l'extension :

```
➜  ~ git:(main) ✗ mkdir -p ~/Sprint5/my_extension_void
➜  ~ git:(main) ✗ cd ~/Sprint5/my_extension_void
➜  my_extension_void git:(master) touch config.m4
➜  my_extension_void git:(master) ✗ touch php_my_extension_void.h
➜  my_extension_void git:(master) ✗ touch my_extension_void.c
➜  my_extension_void git:(master) ✗ ls -la
total 0
drwxr-xr-x@  5 void  staff  160 22 Feb 16:21 .
drwxr-xr-x@ 12 void  staff  384 22 Feb 16:19 ..
-rw-r--r--@  1 void  staff    0 22 Feb 16:20 config.m4
-rw-r--r--@  1 void  staff    0 22 Feb 16:21 my_extension_void.c
-rw-r--r--@  1 void  staff    0 22 Feb 16:21 php_my_extension_void.h
➜  my_extension_void git:(master) ✗ '
```



2. **Étape 2 :** Écrire le code en C :

   Le fichier **php_my_extension_void.h** :



   Le fichier **my_extension_void.c** :

my_extension_void.c

```c
/*
 * my_extension_void.c
 * Extension PHP: my_extension_void
 */

#ifdef HAVE_CONFIG_H
#include "config.h"
#endif

#include "php.h"
#include "php_ini.h"
#include "ext/standard/info.h"
#include "php_my_extension_void.h"

ZEND_BEGIN_ARG_INFO(arginfo_void_say_hello, 0)
ZEND_END_ARG_INFO()

/* Liste des fonctions disponibles en PHP */
static const zend_function_entry my_extension_void_functions[] = {
    PHP_FE(void_say_hello, arginfo_void_say_hello)
    PHP_FE_END
};

/* Définition du module */
zend_module_entry my_extension_void_module_entry = {
    STANDARD_MODULE_HEADER,
    "my_extension_void",                    /* Nom de l'extension */
    my_extension_void_functions,            /* Fonctions */
    NULL,                                    /* Module init */
    NULL,                                    /* Module shutdown */
    NULL,                                    /* Request init */
    NULL,                                    /* Request shutdown */
    NULL,                                    /* Info function */
    PHP_MY_EXTENSION_VOID_VERSION,          /* Version */
    STANDARD_MODULE_PROPERTIES
};

#ifdef COMPILE_DL_MY_EXTENSION_VOID
#ifdef ZTS
ZEND_TSRMLS_CACHE_DEFINE()
#endif
ZEND_GET_MODULE(my_extension_void)
#endif


PHP_FUNCTION(void_say_hello)
{
    php_printf("Hello from my_extension_void sara!\n");
    RETURN_TRUE;
}
```

3. **Étape 3 :** Créer les fichiers de configuration

config.m4

```
dnl config.m4 for extension my_extension_void

PHP_ARG_ENABLE(my_extension_void, whether to enable my_extension_void support,
[  --enable-my_extension_void           Enable my_extension_void support])

if test "$PHP_MY_EXTENSION_VOID" = "yes"; then
  AC_DEFINE(HAVE_MY_EXTENSION_VOID, 1, [Whether you have my_extension_void])
  PHP_NEW_EXTENSION(my_extension_void, my_extension_void.c, $ext_shared)
fi
```

4. **Étape 4 :** Compiler l'extension

```
phpize
./configure --enable-my_extension_void
make
make install
```

(a) `phpize` Prépare l'environnement PHP pour compiler une extension.

```
➜  my_extension_void git:(master) ✗ phpize
Configuring for:
PHP Version:             8.5
PHP Api Version:         20250925
Zend Module Api No:      20250925
Zend Extension Api No:   420250925
➜  my_extension_void git:(master) ✗ ls -la
total 1224
drwxr-xr-x@ 11 void  staff     352 22 Feb 16:29 .
drwxr-xr-x@ 12 void  staff     384 22 Feb 16:19 ..
drwxr-xr-x@  7 void  staff     224 22 Feb 16:29 autom4te.cache
drwxr-xr-x@ 14 void  staff     448 22 Feb 16:29 build
-rw-r--r--@  1 void  staff    1775 22 Feb 16:29 config.h.in
-rw-r--r--@  1 void  staff     403 22 Feb 16:26 config.m4
-rwxr-xr-x@  1 void  staff  458165 22 Feb 16:29 configure
-rw-r--r--@  1 void  staff    5124 22 Feb 16:29 configure.ac
-rw-r--r--@  1 void  staff    1391 22 Feb 16:25 my_extension_void.c
-rw-r--r--@  1 void  staff     499 22 Feb 16:29 php_my_extension_void.h
-r-xr-xr-x@  1 void  staff  142978 22 Feb 16:29 run-tests.php
➜  my_extension_void git:(master) ✗
```

(b) `./configure -enable-my_extension_void` Configure la compilation pour l'extension.

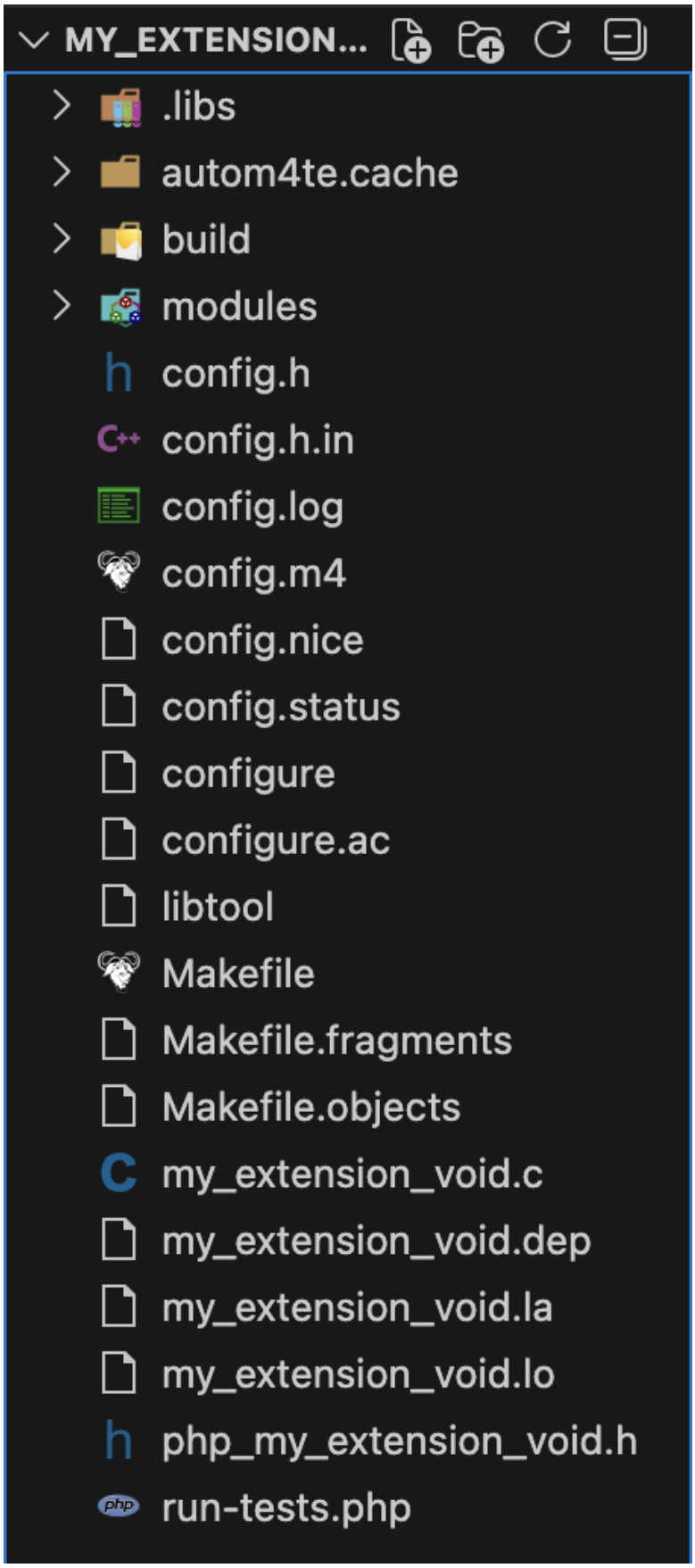
```
➜  my_extension_void git:(master) ✗ ./configure --enable-my_extension_void
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for a sed that does not truncate output... /usr/bin/sed
checking build system type... x86_64-apple-darwin23.5.0
checking host system type... x86_64-apple-darwin23.5.0
checking target system type... x86_64-apple-darwin23.5.0
checking for gawk... no
checking for nawk... no
checking for awk... awk
checking if awk is broken... no
checking for pkg-config... /usr/local/bin/pkg-config
checking pkg-config is at least version 0.9.0... yes
```

(c) `make` Compile le code C et génère le fichier `my_extension_void.so`.

```
➜  my_extension_void git:(master) ✗ make
/bin/sh /Users/void/Sprint5/my_extension_void/libtool --tag=CC --mode=compile cc -I. -I/Users/void/Sprint5/my_extension_void -I/usr/local/Cellar/php/8.5.2/include/php -I/usr/local/Cellar/php/8.5.2/include
/php/main -I/usr/local/Cellar/php/8.5.2/include/php/TSRM -I/usr/local/Cellar/php/8.5.2/include/php/Zend -I/usr/local/Cellar/php/8.5.2/include/php/ext -I/usr/local/Cellar/php/8.5.2/include/php/ext/date/lib
  -DHAVE_CONFIG_H  -g -O2 -D_GNU_SOURCE    -DZEND_COMPILE_DL_EXT=1 -c /Users/void/Sprint5/my_extension_void/my_extension_void.c -o my_extension_void.lo  -MMD -MF my_extension_void.dep -MT my_extension_voi
d.lo
mkdir .libs
 cc -I. -I/Users/void/Sprint5/my_extension_void -I/usr/local/Cellar/php/8.5.2/include/php -I/usr/local/Cellar/php/8.5.2/include/php/main -I/usr/local/Cellar/php/8.5.2/include/php/TSRM -I/usr/local/Cellar/
php/8.5.2/include/php/Zend -I/usr/local/Cellar/php/8.5.2/include/php/ext -I/usr/local/Cellar/php/8.5.2/include/php/ext/date/lib -DHAVE_CONFIG_H -g -O2 -D_GNU_SOURCE -DZEND_COMPILE_DL_EXT=1 -c /Users/void/
Sprint5/my_extension_void/my_extension_void.c -MMD -MF my_extension_void.dep -MT my_extension_void.lo  -fno-common -DPIC -o .libs/my_extension_void.o
/bin/sh /Users/void/Sprint5/my_extension_void/libtool --tag=CC --mode=link cc -shared -I/usr/local/Cellar/php/8.5.2/include/php -I/usr/local/Cellar/php/8.5.2/include/php/main -I/usr/local/Cellar/php/8.5.2
/include/php/TSRM -I/usr/local/Cellar/php/8.5.2/include/php/Zend -I/usr/local/Cellar/php/8.5.2/include/php/ext -I/usr/local/Cellar/php/8.5.2/include/php/ext/date/lib  -DHAVE_CONFIG_H  -g -O2 -D_GNU_SOURCE
    -o my_extension_void.la -export-dynamic -avoid-version -prefer-pic -module -rpath /Users/void/Sprint5/my_extension_void/modules  my_extension_void.lo
cc ${wl}-flat_namespace ${wl}-undefined ${wl}suppress -o .libs/my_extension_void.so -bundle  .libs/my_extension_void.o
ld: warning: -undefined suppress is deprecated
dsymutil .libs/my_extension_void.so || :
creating my_extension_void.la
(cd .libs && rm -f my_extension_void.la && ln -s ../my_extension_void.la my_extension_void.la)
/bin/sh /Users/void/Sprint5/my_extension_void/libtool --tag=CC --mode=install cp ./my_extension_void.la /Users/void/Sprint5/my_extension_void/modules
cp ./.libs/my_extension_void.so /Users/void/Sprint5/my_extension_void/modules/my_extension_void.so
cp ./.libs/my_extension_void.lai /Users/void/Sprint5/my_extension_void/modules/my_extension_void.la
----------------------------------------------------------------------
Libraries have been installed in:
   /Users/void/Sprint5/my_extension_void/modules

If you ever happen to want to link against installed libraries
in a given directory, LIBDIR, you must either use libtool, and
specify the full pathname of the library, or use the `-LLIBDIR'
flag during linking and do at least one of the following:
   - add LIBDIR to the `DYLD_LIBRARY_PATH' environment variable
     during execution

See any operating system documentation about shared libraries for
more information, such as the ld(1) and ld.so(8) manual pages.
----------------------------------------------------------------------

Build complete.
Don't forget to run 'make test'.
```

(d) `make install` Copie le fichier `.so` dans le répertoire `extension_dir`.

```
➜  my_extension_void git:(master) ✗ sudo make install
Password:
Installing shared extensions:     /usr/local/Cellar/php/8.5.2/pecl/20250925/
```

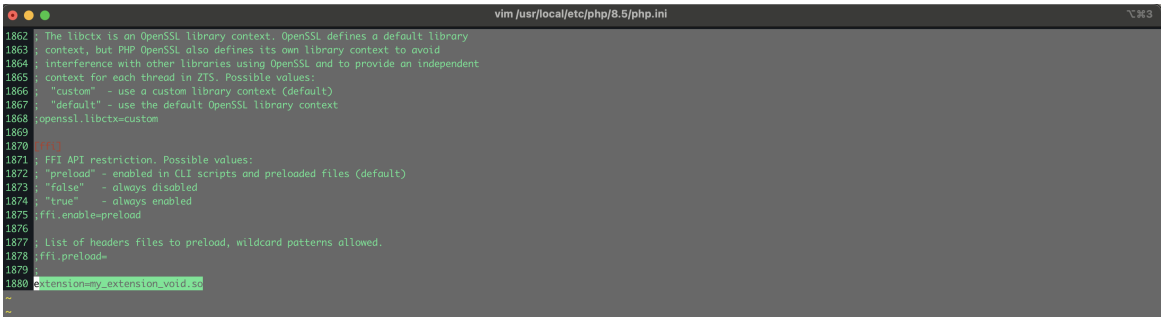


5. **Étape 5 :** Installer et configurer PHP

```
➜  my_extension_void git:(master) ✗ php --ini
Configuration File (php.ini) Path: "/usr/local/etc/php/8.5"
Loaded Configuration File:         "/usr/local/etc/php/8.5/php.ini"
Scan for additional .ini files in: "/usr/local/etc/php/8.5/conf.d"
Additional .ini files parsed:      (none)
➜  my_extension_void git:(master) ✗
```

On ajoute au php.ini :

`extension=my_extension_void.so`



6. **Étape 6 :** Tester l'extension

On teste par :

```
php -m | grep my_extension_void
php -i | grep extension_dir
```

```
➜  my_extension_void git:(master) ✗ php -m | grep my_extension_void
PHP Warning:  Missing arginfo for void_say_hello() in Unknown on line 0
my_extension_void
➜  my_extension_void git:(master) ✗ php -i | grep extension_dir
PHP Warning:  Missing arginfo for void_say_hello() in Unknown on line 0
extension_dir => /usr/local/lib/php/pecl/20250925 => /usr/local/lib/php/pecl/20250925
sqlite3.extension_dir => no value => no value
```

L'extension est bien chargée .

NB :
pour l'erreur :Depuis PHP 8, chaque fonction d'extension doit déclarer ses arginfo (informations sur les paramètres).
après correction du code c :

```
➜  my_extension_void git:(master) ✗ php -m | grep my_extension_void
my_extension_void
➜  my_extension_void git:(master) ✗
```

Pour tester la fonction :

`php -r "void_say_hello();"`

```
➜  my_extension_void git:(master) ✗ php -r "void_say_hello();"
Hello from my_extension_void sara!
➜  my_extension_void git:(master) ✗
```

Le lien vers le dépot GitHub

check this : **extension-tutorial**

Quel intérêt peut avoir une extension PHP en termes de performance et de fonctionnalités ?

— **Vitesse d'exécution :** Les extensions PHP sont significativement plus rapides que le code PHP pur interprété grâce à la compilation en C/C++.

— **Utilisation mémoire :** Les extensions consomment beaucoup moins de RAM grâce à une gestion mémoire optimisée au niveau système.

— **Opérations mathématiques :** Les calculs complexes s'exécutent de manière beaucoup plus efficace en code compilé.

— **Traitement de fichiers volumineux :** La lecture et le traitement de gros fichiers consomment considérablement moins de mémoire.

— **Traitement de données structurées :** Le parsing JSON, XML et autres formats est nettement plus rapide avec les extensions.

— **Utilisation du processeur :** Les extensions réduisent significativement la charge CPU comparé au code PHP pur.

— **Accès aux bases de données :** Les extensions fournissent des fonctionnalités impossibles à implémenter en PHP pur.

— **Traitement d'images :** Les manipulations d'images sont exécutées de manière beaucoup plus efficace et rapide.

— **Compression de données :** Les extensions permettent une compression efficace des données avec une faible consommation de ressources.

— **Accès aux ressources système :** Les extensions offrent un accès direct aux capacités du système d'exploitation.