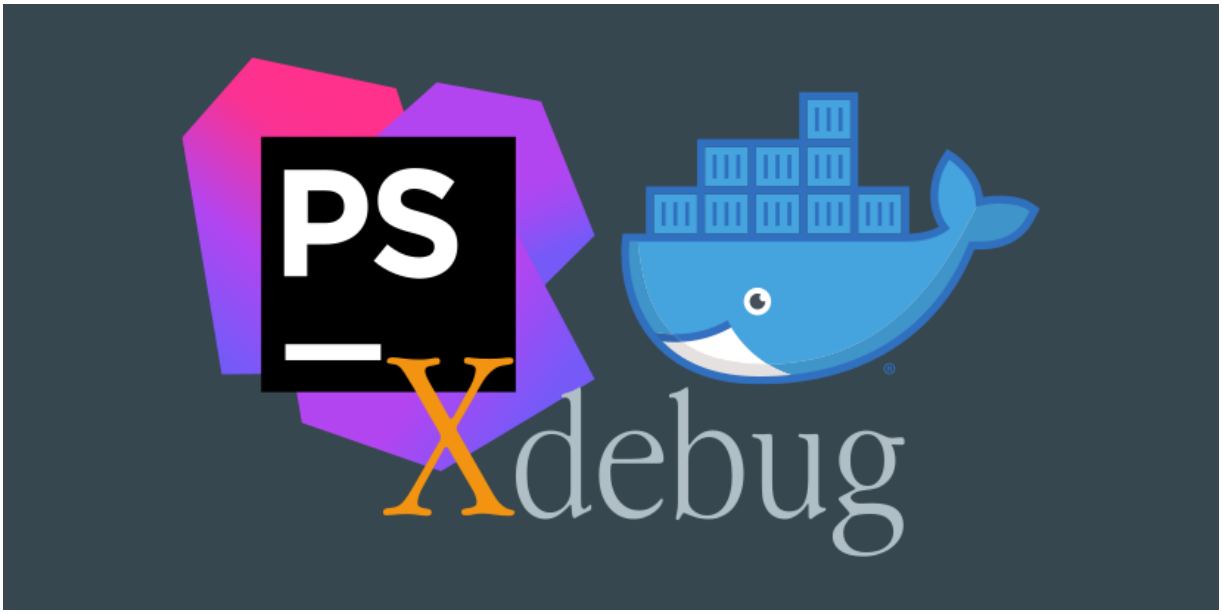
Денис Бондарь

Авторский блог



PhpStorm + Docker + Xdebug

В этой небольшой статье я опишу рабочее решение настройки Xdebug для использования его при отладке в PhpStorm с использованием удаленного интерпретатора PHP, работающего внутри Docker-контейнера.

В итоге получится конфигурация на базе окружения в Docker-контейнерах, которая позволит производить отладку Web-приложения, консольного приложения, тестов, запускаемых из консоли и запускаемых из PhpStorm.

Исходные данные

У вас должны быть установлены Linux, PhpStorm, Docker, Docker-compose. У меня в блоге есть небольшая памятка, как установить Docker и docker-compose в Linux. Если Вы ведете разработку не на Linux, то Вам, наверное, придется чуть сложнее, но раз Вы читаете эту статью, значит Вы уже озадачились вопросом отладки с использованием php в контейнере Docker и у Вас уже всё

давно установлено. Если настройки для MacOS или Windows будут отличаться от приведенных в статье, я обязательно укажу на это.

Статья будет рассмотрена на следующем примере. Каталог на локальной машине (хосте), в котором планируется разработать некоторое приложение: /home/denis/code/docker-xdebug. При этом корень Web-сервера находится в подкаталоге public. Основной каталог проекта отображается внутрь Docker-контейнеров на каталог /var/www. Внутри Docker-контейнера, при этом, получаем путь к корневому каталогу Web-сервера /var/www/public — именно на него настроен Web-сервер nginx.

Пример конфигурации

На самом деле, поставленная задача достаточно тривиальная. Все решения, которые мне удавалось находить ранее, были завязаны на IP-адресе хоста и их невозможно было использовать остальными членами команды, так как в каждом новом случае IP-адрес хоста был случайный и каждому члену команды приходилось бы менять IP-адрес на свой в общем файле конфигурации бандла, что крайне неудобно.

Решение проблемы оказалось достаточно простым. Я просто создал общую сеть для бандла в docker-compose (192.168.220.0/28) и таким образом добился одинакового IP-адреса хоста (192.168.220.1) на всех машинах членов команды разработки.

Если Вы используете MacOS или Windows, то вместо адреса **192.168.220.1** Вам нужно будет указать **host.docker.internal** в приведенном ниже docker-compose.yml.

Ниже приведу подробную инструкцию и все файлы, необходимые для сборки бандла с рабочей отладкой Xdebug.

В файлах отсутствует установка каких либо дополнительных зависимостей и расширений, так как в каждом проекте они свои. Это выходит за рамки данной статьи. Вы должны сами добавить их при необходимости.

docker-compose.yml

```
version: '3'
services:
  php-fpm:
    build:
```

```
      context: docker/php-fpm
    volumes:
      - ./:/var/www
    environment:
      XDEBUG_CONFIG: "remote_host=192.168.220.1 remote_enable=1"
      PHP_IDE_CONFIG: "serverName=Docker"
    networks:
      - internal
  nginx:
    build:
      context: docker/nginx
    volumes:
      - ./:/var/www
    ports:
      - "80:80"
    depends_on:
      - php-fpm
    networks:
      - internal
networks:
  internal:
    driver: bridge
    ipam:
      driver: default
      config:
        - subnet: 192.168.220.0/28
```

Если у вас MacOS или Windows, то вместо **remote_host=192.168.220.1** используйте **remote_host=host.docker.internal**.

docker/php-fpm/Dockerfile

```
FROM php:7.2-fpm

RUN apt-get update && apt-get install -y wget git unzip \
    && pecl install xdebug-2.7.1 \
    && docker-php-ext-enable xdebug
```

```
ADD ./php.ini /usr/local/etc/php/php.ini

RUN wget https://getcomposer.org/installer -O - -q \
    | php -- --install-dir=/bin --filename=composer --quiet

WORKDIR /var/www
```

docker/php-fpm/php.ini

```
max_execution_time = 1000
max_input_time = 1000
```

Это необходимо для того, чтобы Вы успели изучить результаты отладки до истечения таймаута запроса.

docker/nginx/Dockerfile

```
FROM nginx

ADD ./default.conf /etc/nginx/conf.d/default.conf

WORKDIR /var/www
```

docker/nginx/default.conf

```
server {
    listen 80;
    index index.php;
    server_name 127.0.0.1 localhost;
    root /var/www/public;

    location / {
        try_files $uri /index.php?$args;
    }
```

```
        location ~ \.php$ {
            fastcgi_split_path_info ^(.+\.php)(/.+)$;
            fastcgi_pass php-fpm:9000;
            fastcgi_index index.php;
            fastcgi_read_timeout 1000;
            include fastcgi_params;
            fastcgi_param SCRIPT_FILENAME
    $document_root$fastcgi_script_name;
            fastcgi_param PATH_INFO $fastcgi_path_info;
        }
    }
```

Теперь просто запускаем бандл docker-контейнеров:

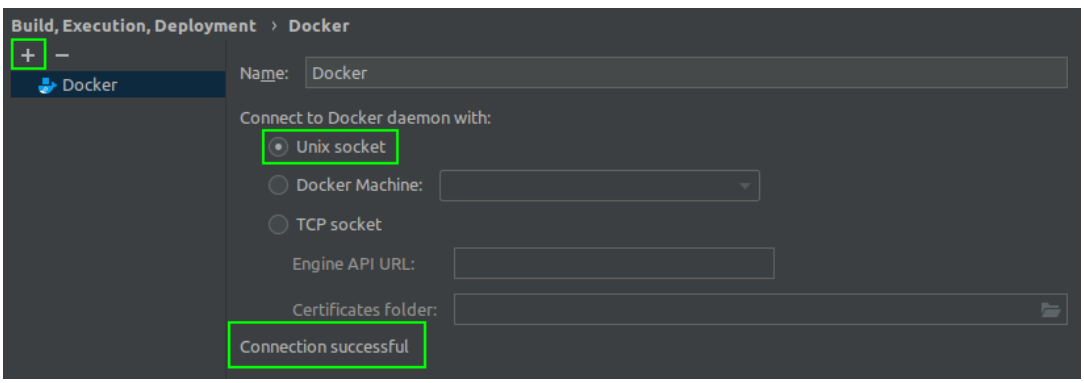
```
$ docker-compose up -d --build
```

Настройка PhpStorm

Настройка отладки в PhpStorm типичная для подобных ситуаций. Необходимо в настройках **(File — Settings)** проделать следующее:

Добавить Docker сервер

Если он у Вас еще не добавлен, конечно. Для этого откройте настройки: **«Build, Execution, Deployment» — Docker**. Затем нажмите плюсик, чтобы добавить новое подключение к докеру. У меня все подключилось сразу же в режиме Unix socket. Нажмите Apply.

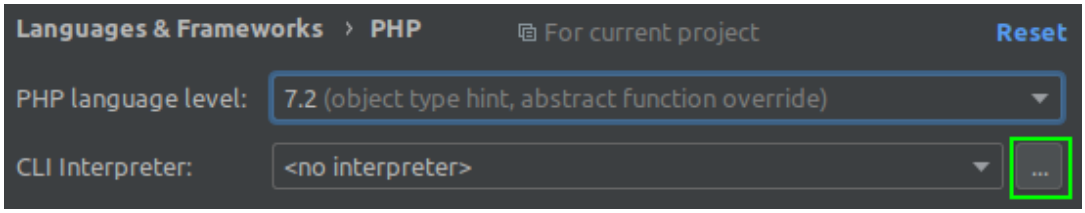


Добавление нового Docker сервера

Добавить внешний интерпретатор

Откройте настройки: **«Languages & Frameworks» — PHP**

Справа от CLI Interpreter нажмите кнопку с тремя точками.

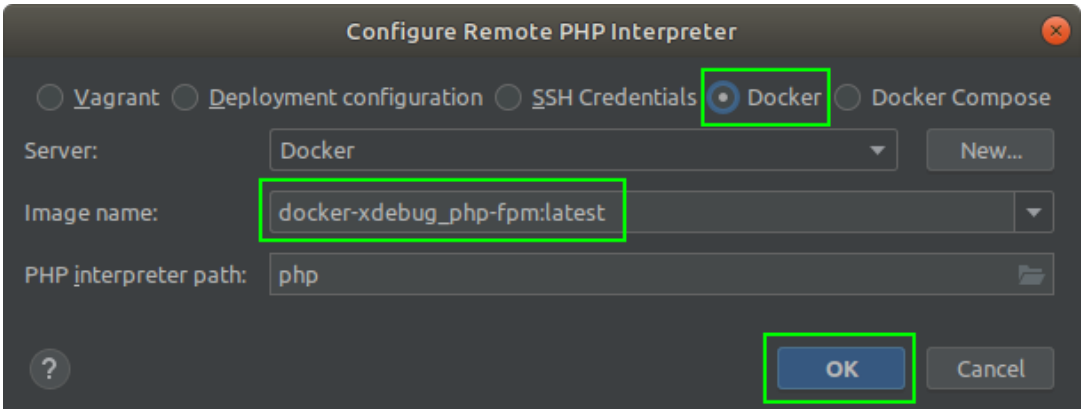


Выбор версии PHP и добавление интерпретатора

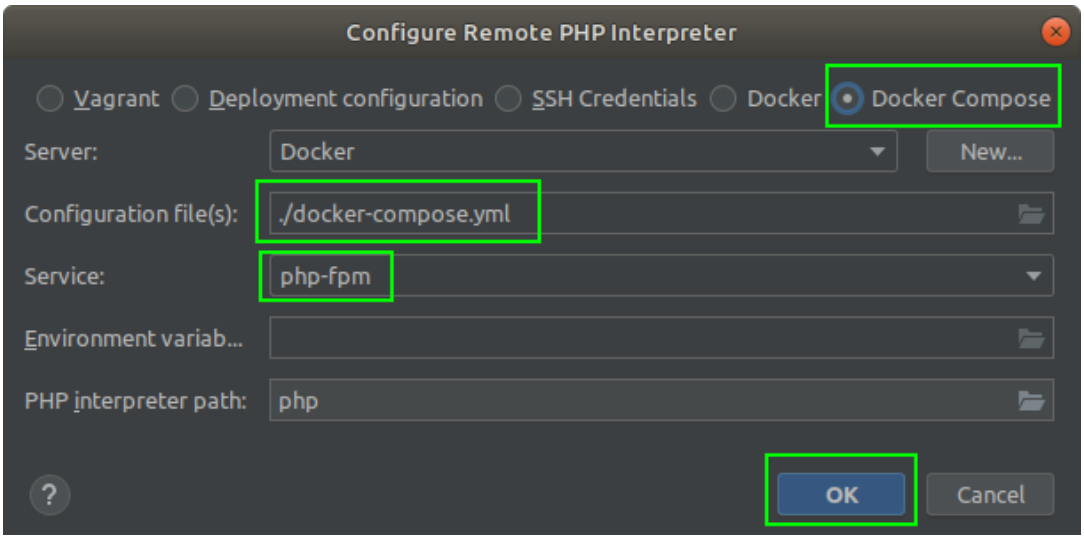
В открывшемся окне CLI Interpreters нажмите плюс слева вверху и выберите там **From Docker...**.

Далее нужно выбрать **Docker** (либо в новых версиях PhpStorm можно также выбрать **Docker Compose**). Второй вариант мне показался удобней, так как в этом случае я вижу контейнеры только данного проекта, а не все существующие на хосте (у меня их довольно много).

Два варианта на выбор:



Docker

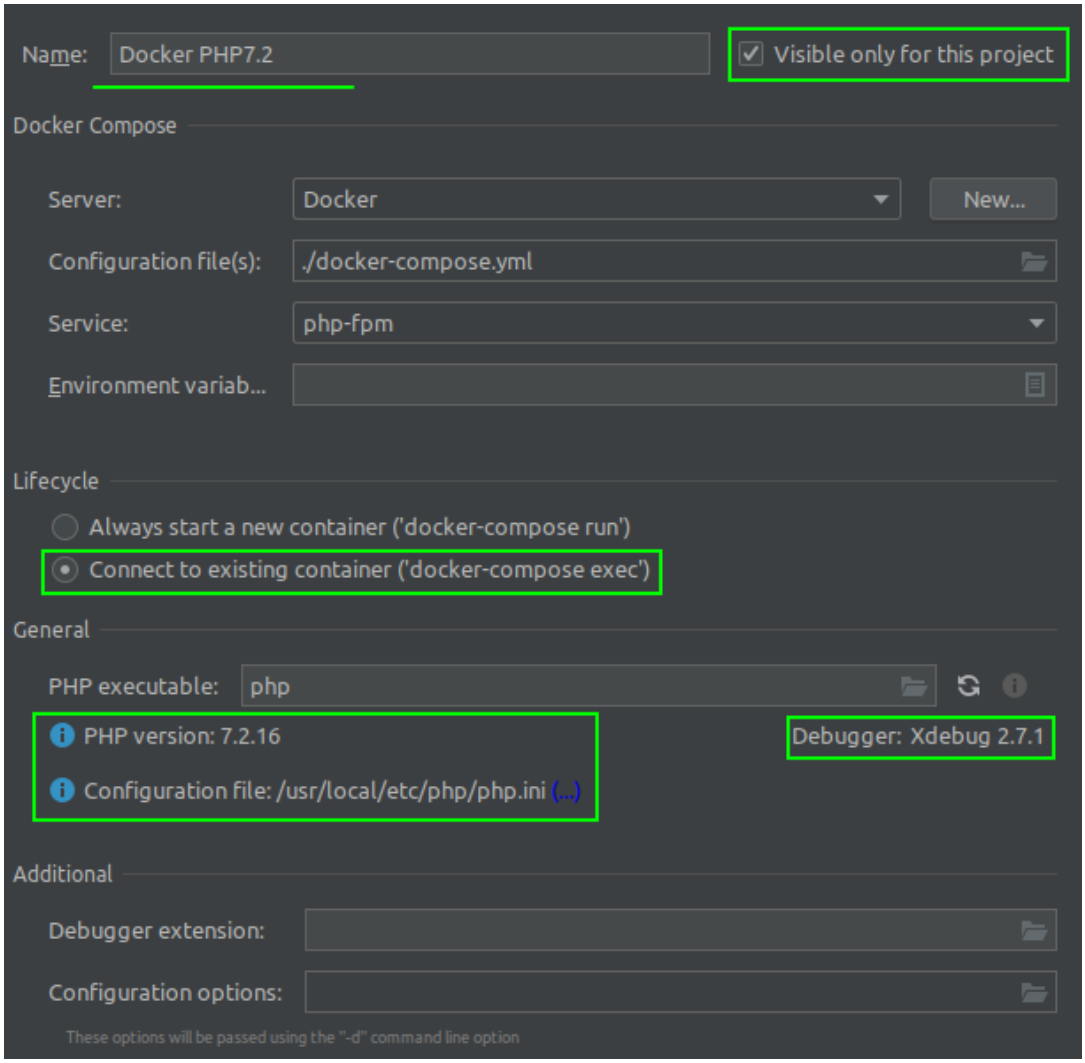


Docker Compose

Не забывайте правильно указывать наименование образа или сервиса. Нажмите OK, после чего будет произведена попытка добавления выбранного интерпретатора из Docker.

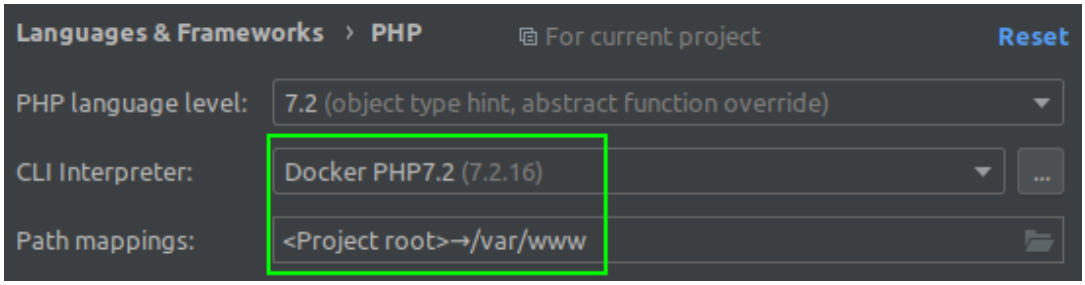
В случае удачи, вы увидите следующее окно (здесь видно, что была определена версия PHP, конфигурационный файл PHP, а также, что очень важно, версия Xdebug. Переименуйте интерпретатор в нечто более понятное, например, как на скриншоте ниже.

Начиная с версии PhpStorm 2019.1 появилась возможность выбора, каким образом запускать тесты в контейнере. Если у вас контейнер с постоянно работающей службой, такой как **php-fpm**, то выбирайте **Connect to existent container**. Если же у вас в качестве интерпретатора используется **php-cli**, который не выполняется постоянно, то выберите опцию **Always start a new container**. Нажмите OK.



Настройка интерпретатора PHP

Теперь Вы должны увидеть следующее:

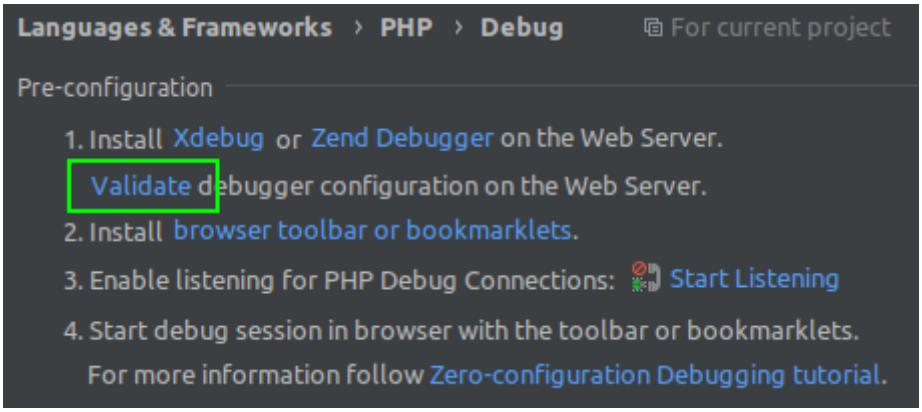


Path mappings берется из docker-compose.yml

Нажмите Apply.

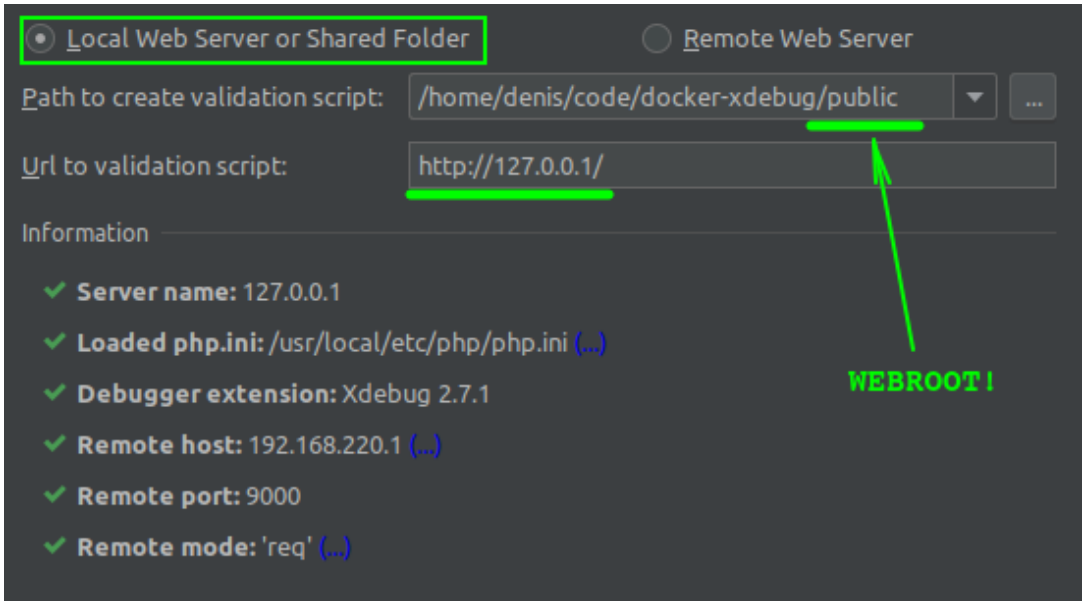
Проверить конфигурацию Xdebug

Откройте настройки: **«Languages & Frameworks» — PHP — Debug.** И нажмите на ссылку **Validate**.



Нажмите Validate для проверки настройки

Откроется окно, в котором Вы должны указать полный путь к web-корню (в нашем примере это подкаталог проекта **public**) и адрес сервера (в нашем примере порт 80 контейнера nginx транслируется в порт 80 хоста, поэтому в адресе порт не указывается. Если Вы транслировали в другой порт хоста — укажите обязательно его, например, http://127.0.0.1:8080). Нажмите Validate.



Результаты проверки Xdebug

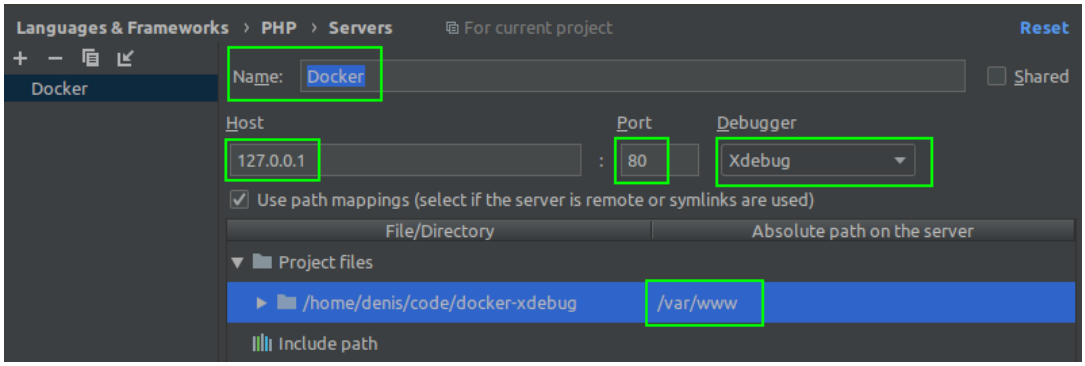
Если у Вас результат проверки выглядит так же, то Вы все настроили правильно. Закройте окно Validate Debugger Configuration.

Если у Вас проблема с валидацией имени сервера, значит вы в настройках сервера nginx не указали 127.0.0.1 в качестве одного из server_name. Если проблема с загрузкой php.ini, значит вы использовали другой способ конфигурирования PHP вместо загрузки php.ini в каталог контейнера /usr/local/etc/php/ — это не страшно, если Вы понимаете, что делаете.

Добавить PHP сервер

Откройте настройки: **«Languages & Frameworks» — PHP — Servers** и нажмите плюсик, чтобы добавить новый.

Укажите имя сервера **Docker** (должно совпадать с переменной окружения PHP_IDE_CONFIG в docker-compose.yml) и хост 127.0.0.1. Затем ниже включите опцию Use path mappings и укажите соответствие между локальным каталогом проекта и этим же каталогом проекта внутри Docker-контейнера. Это соответствие изначально настраивается в docker-composer.yml для службы php-fpm в разделе volumes. Затем нажмите OK.

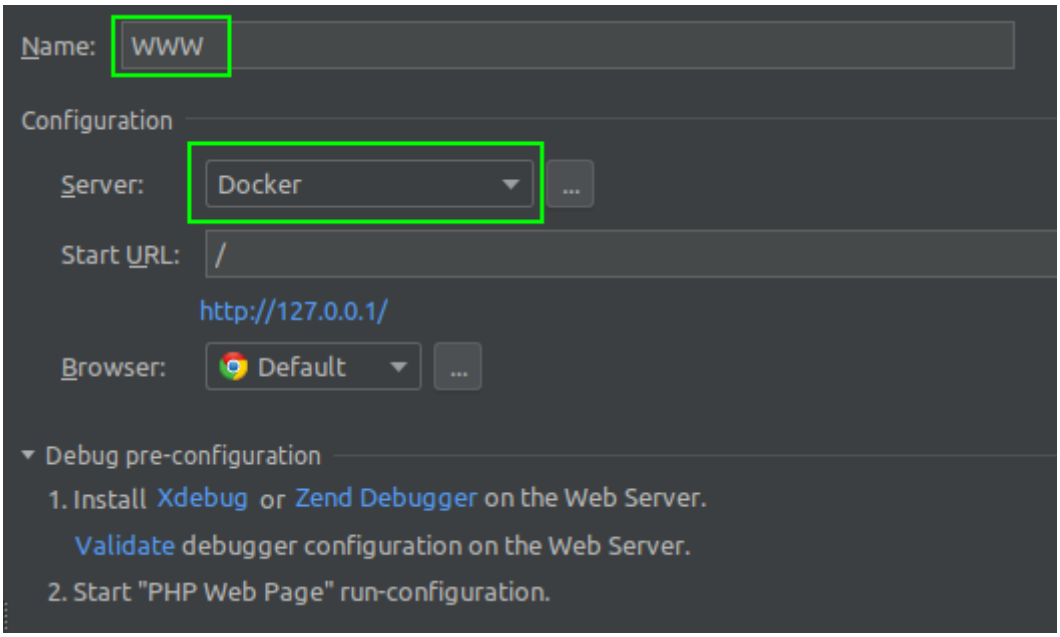


Добавить конфигурацию для запуска

Остался последний штрих — добавление конфигурации запуска. Мы добавим простую Web-страницу.

В главном меню **Run** перейдите в **Edit configurations...**

В открывшемся окне нажимайте плюсик вверху слева и выбирайте PHP Web page. Укажите имя для данной страницы, например, WWW, затем укажите сервер, с которым связана эта страница (мы создали его на предыдущем шаге) и нажмите OK.



Конфигурация Web сервера для отладки

Упрощенный способ взаимодействия с контейнерами

Для управления контейнерами, вам придется использовать длинные команды вроде:

```
docker-compose up -d
docker-compose down -v --remove-orphans
docker-compose logs -f
и так далее...
```

Но это не самое сложное. Самое сложное заключается в том, что теперь для запуска тестов вам придется вызывать сценарии внутри контейнеров. Например, чтобы запустить конкретный набор тестов, вам придется ввести следующую команду:

```
docker-compose run --rm php-fpm vendor/bin/phpunit --testsuite
unit
```

Постоянный ввод таких команд — вырабатывает хорошую привычку, но быстро надоест. Ведь это можно упростить!

Для этого создайте в корне проекта файл (например, с именем **alias.bash**), содержащий псевдонимы команд, а затем настройте PhpStorm таким образом, чтобы он подключал его при открытии терминала. В Git-репозитории, ссылка на который приведена в конце статьи, вы найдете все необходимые файлы из этой статьи, включая файл alias.bash, который уже содержит некоторые самые необходимые команды в более развернутом виде, который будет удобней для редактирования и поддержки.

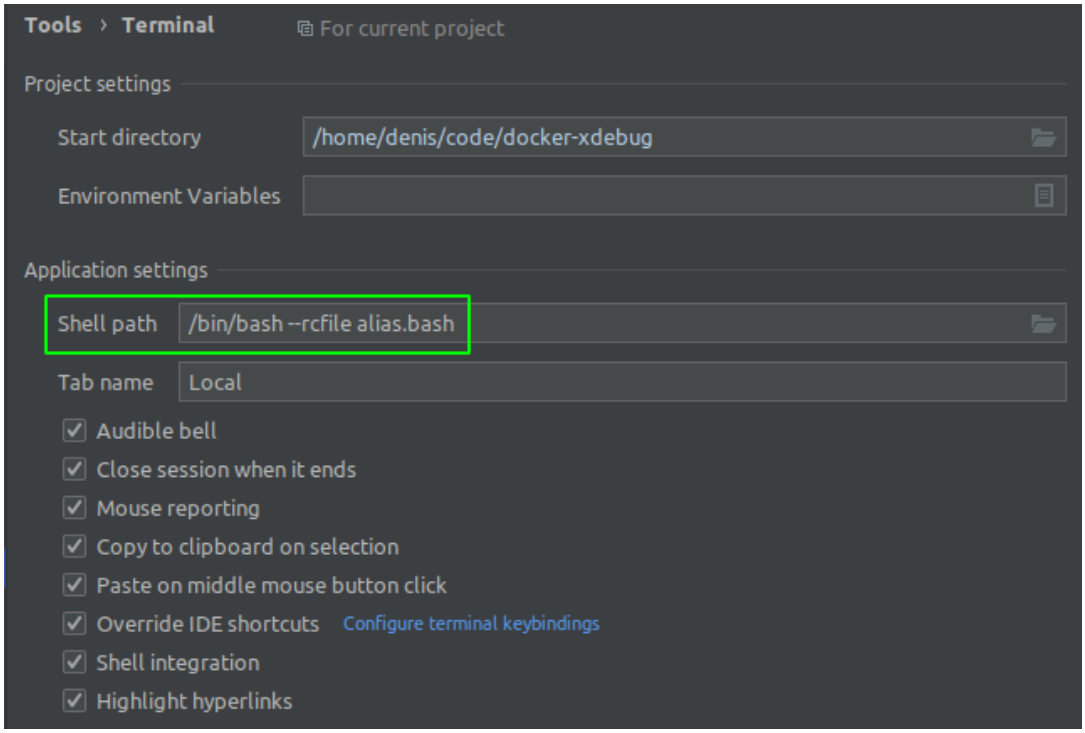
Ваш файл alias.bash может, например, выглядеть следующим, упрощенным образом:

```
alias env-up='docker-compose up -d'
alias env-stop='docker-compose stop'
alias phpunit='docker-compose run --rm php-fpm
vendor/bin/phpunit'
```

Псевдонимы могут работать с параметрами. Это видно в последнем псевдониме — сценарию phpunit передаются все параметры, с которыми был вызван псевдоним. Такие псевдонимы можно назвать своего рода прокси-командами.

Чтобы файл с псевдонимами применялся при открытии сессии терминала в PhpStorm, откройте настройки: **Tools — Terminal** и в поле **Shell path** введите:

```
/bin/bash --rcfile alias.bash
```



Настройка загрузки файла с псевдонимами при старте сессии оболочки в PhpStorm

Теперь откройте терминал в PhpStorm (Alt+F12). Если он уже был открыт, то нужно закрыть и открыть снова. Теперь попробуйте просто выполнить команду env-up и Вы увидите, что фактически выполнилась команда docker-compose up -d.

Вы можете составлять псевдонимы любой сложности, включая составные, состоящие из других псевдонимов. Это позволяет сосредоточиться на разработке, не отвлекаясь на составление команд для запуска чего либо внутри контейнера вручную.

Запуск отладки

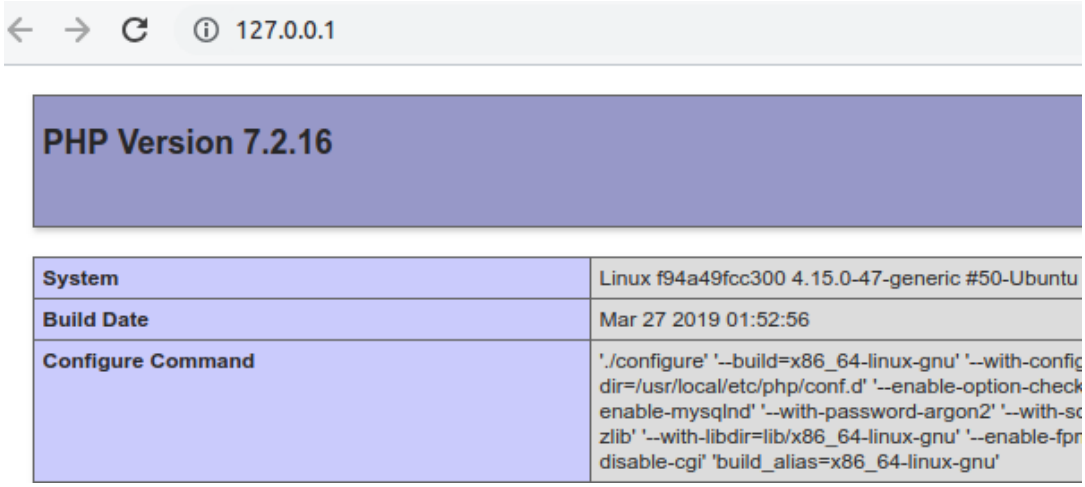
Теперь пришло время приступить к самой отладке.

В подкаталоге проекта public создайте файл index.php с следующим содержимым:

```
<?php
$s = $_SERVER;
phpinfo();
```

Вверху справа на панели запуска выберите созданную конфигурацию и нажмите зеленый треугольник.

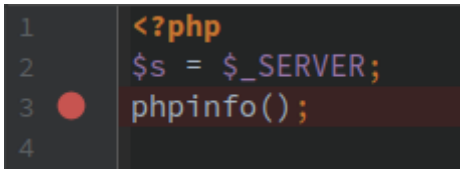
Это действие должно запустить сервер, который откроет в браузере файл index.php, в результате чего в браузере должен быть отображен результат работы phpinfo().



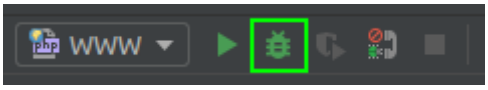
Результат работы index.php через браузер

Отладка WEB-приложения

Устанавливаем точку останова на строку **phpinfo();**



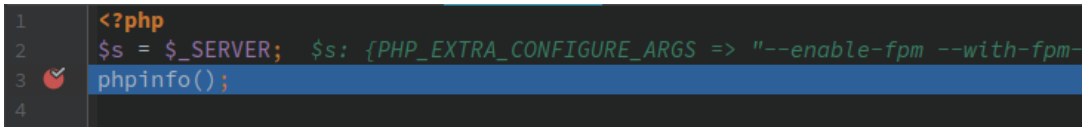
И вместо зеленого треугольника нажмем кнопку с зеленым жуком — это запустит сервер в режиме отладки.



При первом запуске отладки в браузере вам возможно понадобится специальное расширение **JetBrains IDE Support**, которое обычно предлагается к установке автоматически. Установите его.

При запуске отладки, откроется браузер с URL, к которому будет добавлен параметр с идентификатором сессии отладки, что-то вроде: http://127.0.0.1/?XDEBUG_SESSION_START=11223

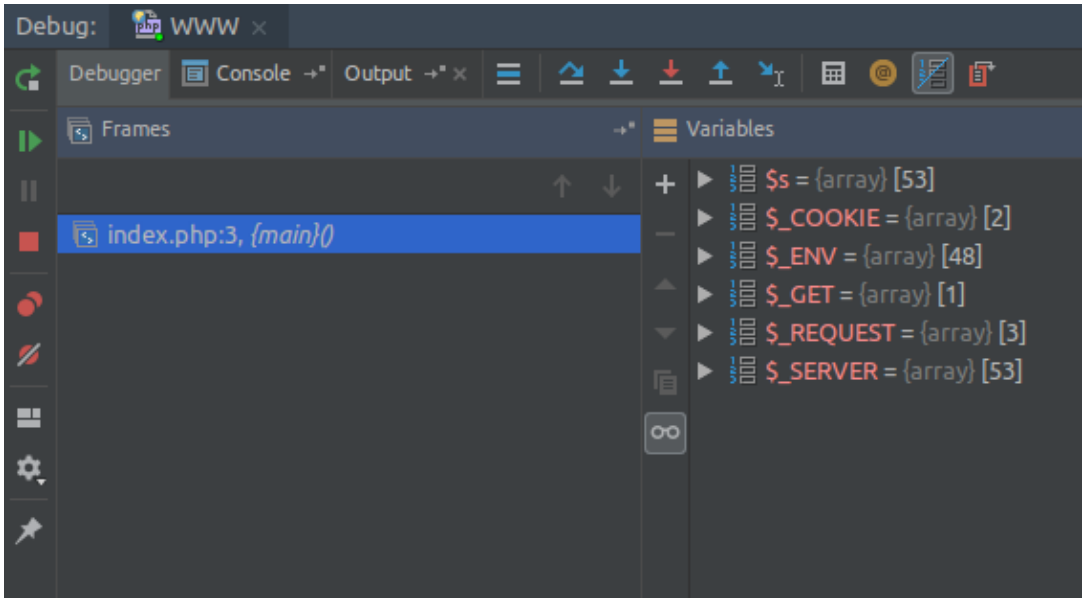
При этом вывод в браузер не будет выполнен, так как сработает точка останова. Вернитесь в окно PhpStorm. Активная точка останова выглядит вот так:



Активная точка останова

В данном случае птичкой обозначается та точка (а их может быть несколько), на которой произошел останов. Все переменные выше будут подсвечены их значениями для удобства.

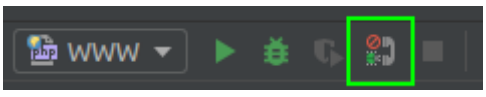
Теперь переместимся ниже — на панель отладки.



Панель отладки PhpStorm

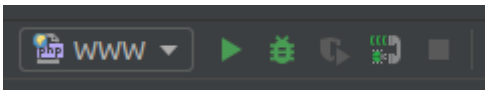
На ней видна вся отладочная информация. Справа — значения всех переменных в текущей области видимости (здесь также видна наша переменная $s), слева виден стек вызовов (сейчас он пуст, так как у нас примитивный пример). Имеются кнопки на левой панели, при помощи которых можно перезапустить отладку, прервать отладки продолжить выполнение с этого места, остановить отладку и другие. На верхней панели имеются кнопки, при помощи которых можно управлять отладкой: проходить пошагово каждое действие, начиная с точки останова, перепрыгивать функции, не заходя во внутрь и т.д.

Также существует другой способ отладки — при помощи «прослушивания». Нажмите кнопку с телефонной трубкой, чтобы активировать режим прослушивания.



После того, как кнопка изменит вид (с красного запрещающего знака на зеленые полоски), Вы можете просто открыть веб-приложение в браузере и,

если в коде проставлены точки останова, режим отладки запустится автоматически.



Отладка консольного приложения (CLI)

Для отладки консольного приложения можно нажать кнопку с трубкой, чтобы включить режим прослушивания, как и в описанном выше способе.

Теперь можно запустить консольный скрипт следующей командой:

```
$ docker-compose exec php-fpm php ./public/index.php
```

Также можно создать псевдоним для вызова php-сценариев и поместить его в файл alias.bash. Например, вот такой:

```
alias php='docker-compose exec php-fpm php'
```

И теперь тоже самое можно будет сделать выполнив следующую команду:

```
php ./public/index.php
```

Если при этом отладка не сработала, убедитесь, чтобы имя сервера, которое создавали выше, совпадало с именем сервера в файле docker-compose.yml в переменной окружения **PHP_IDE_CONFIG**.

После окончания отладки не забудьте отключить режим прослушивания.

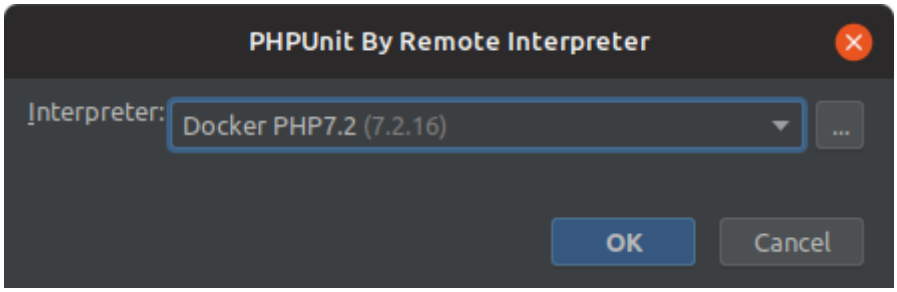
Отладка тестов (PhpUnit)

С отладкой тестов дела обстоят точно так же, как и с консольным приложением. Включайте прослушивание (кнопку с трубкой) и запускайте тесты следующим псевдонимом (прокси-командой):

```
$ phpunit
```

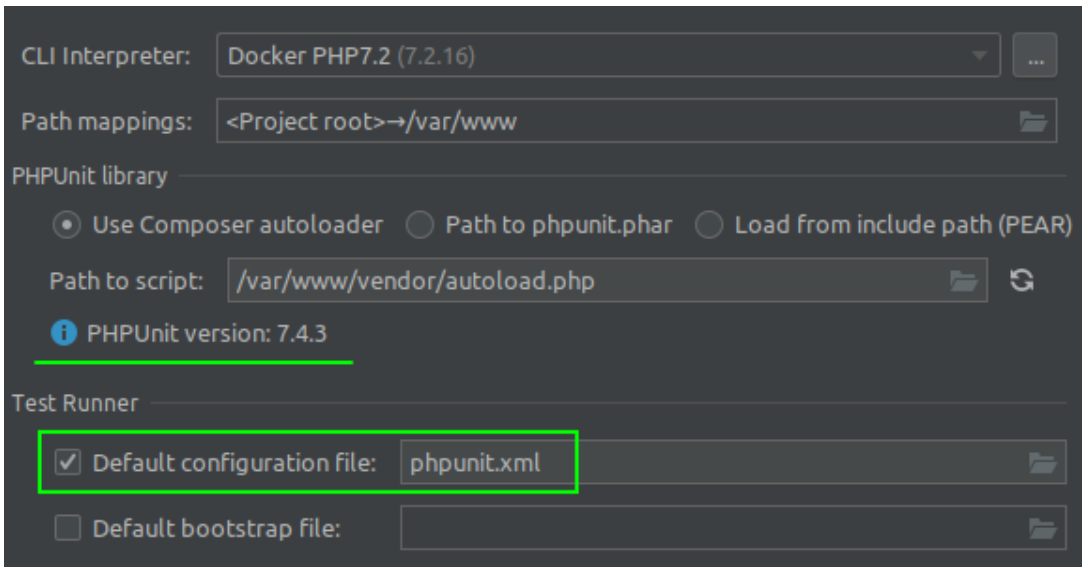
Но используя PhpStorm намного удобней и практичней использовать встроенные средства запуска тестов, которые предоставляют удобную навигацию по тестам, а также множество других полезных функций. Для того чтобы запустить тестирование средствами PhpStorm, необходимо настроить тестовый фреймворк.

Откройте настройки: **«Languages & Frameworks | PHP | Test Frameworks»**. И добавьте новый фреймворк (нажмите кнопку с плюсиком) — PHPUnit by Remote Interpreter. В открывшимся окне выберите наш интерпретатор и нажмите OK.



Настройка интерпретатора для **PHPUnit**

Затем необходимо указать путь к phpunit.xml, убедиться, что путь к Composer autoloader автоматически добавился (добавить вручную, если нет) и нажать OK.



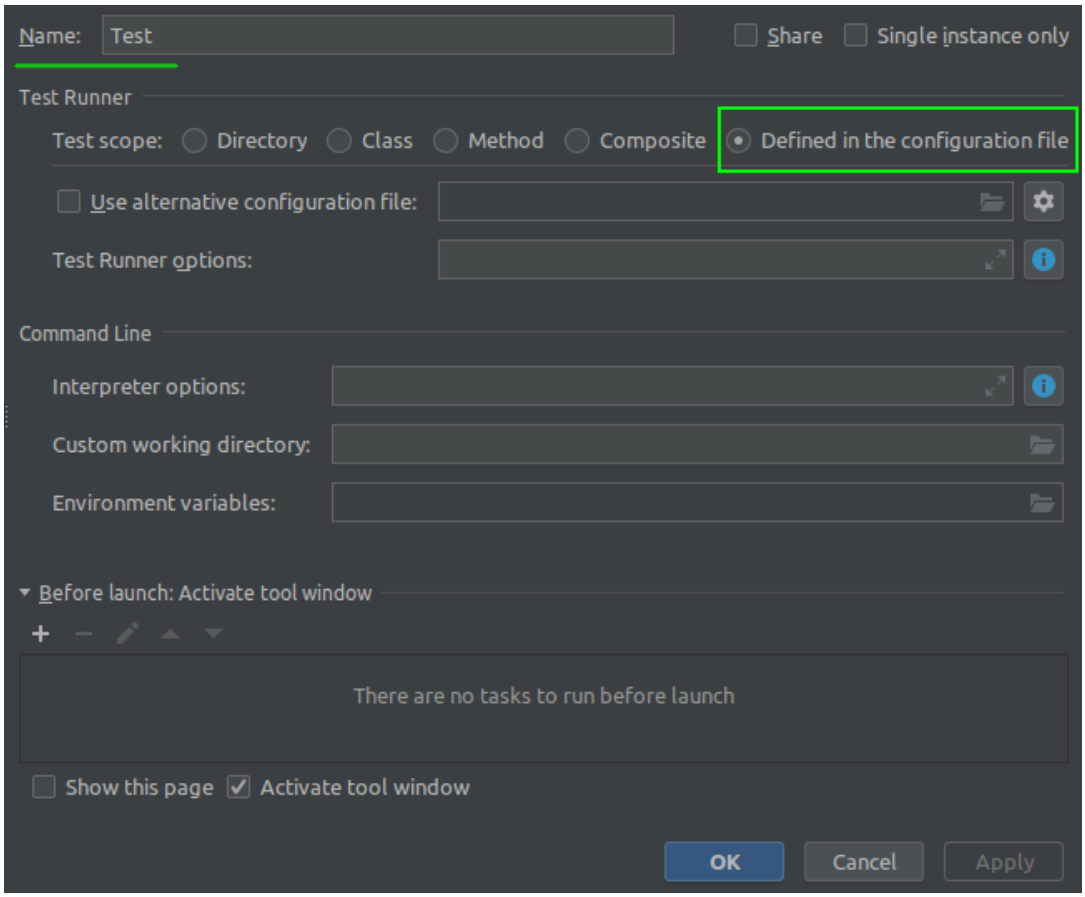
Настройка **PHPUnit**

Тестовый фреймворк настроен. Теперь необходимо добавить **конфигурацию запуска** тестов, подобно той, которую мы добавляли чуть выше для запуска веб-приложения.

Для этого в главном меню **Run** перейдите в **Edit configurations...**

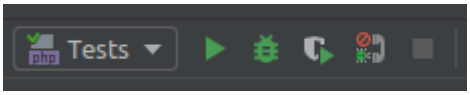
Нажмите плюсик и добавьте **PhpUnit**. Укажите удобное Вам имя, например *Tests* и установите переключатель запуска в Defined in the configuration file.

Теперь Вы можете настраивать тесты через конфигурационный файл phpunit.xml.



Конфигурация запуска тестов PHPUnit

Теперь точно так же кнопкой с зеленым треугольником на панели запуска Вы можете запускать тесты в обычном режиме, кнопкой с жуком — запускать тесты в режиме отладки и кнопки с щитом запускать тесты в режиме расчета покрытия тестами (для расчета покрытия Вы должны настроить блок whitelist в конфигурационном файле phpunit.xml).



При этом активировать режим прослушивания (кнопка с трубкой) уже не нужно.

Резюме

В данной статье был описан простой способ настройки и использования отладчика Xdebug в PhpStorm, если используется удаленный интерпретатор в виде Docker-контейнера.

Все файлы можно скачать на GitHub:
https://github.com/denisbondar/docker_php-fpm_xdebug

После того, как Вы склонируете себе репозиторий, не забудьте выполнить composer install для установки зависимостей и создания автозагрузчика. Для этого можно воспользоваться уже входящим в набор псевдонимов псевдонимом composer:

```
$ composer install
```

18.10.2018 Денис Бондарь Разработка, Статьи docker, docker-compose, php-fpm, phpstorm, xdebug

PhpStorm + Docker + Xdebug: 53 комментария

Андрей

17.08.2019 в 17:45:49

Добрый день! Сделал все точно также, но команда docker-compose up -d —build не отрабатывает и выдает ошибку ‘Unsupported config option for services.networks: ‘internal»
Можете подсказать в чем может быть проблема?

Денис Бондарь

17.08.2019 в 18:57:43

Добрый день. Очень похоже на ошибку в форматировании файла docker-compose.yml, а именно на то, что блок networks у вас попал внутрь блока services

Сергей Штурнев

30.07.2019 в 17:15:17

👍

Г Л

11.07.2019 в 07:13:07

Отличная статья. Большое спасибо!

Денис Бондарь

11.07.2019 в 08:28:43

Пожалуйста!

Павел

24.06.2019 в 14:21:08

Здравствуйте!
Скажите, пожалуйста, а если мне нужно в тестах использовать базу данных(например отдельный контейнер с postgresql), то как в этом случае действовать?
ПОтому что, я, к примеру, даю название контейнеру postgresql-test, а потом в настройках symfony указываю нечто вроде DATABASE_URL=postgresql://postgres:password@postgresql-test:5432/database_test. И именно для тестов не срабатывает(то есть если обычные запросы на сервер выполнять, то все работает, а если с помощью докера запускать тесты — то ничего не выходит).
«could not translate host name «postgresql-test» to address: Name or service not known» — получается, что при подключении к докеру через сокеты, в сети не видно необходимого мне контейнера.
Никогда не сталкивались с подобной ситуацией?

Денис Бондарь

24.06.2019 в 16:45:46

Добрый день.
Но тесты вы ведь из докера запускаете? Не на локальной машине.
Внутри набора (бандла) контейнеров, который вы разворачиваете при помощи docker-compose, организовывается локальная сеть с резолвером имен хостов.
Этот резолвер работает только внутри этой сети.
То есть только запустив процесс внутри одного из контейнеров этого набора, Вы сможете обратиться по сети к другим контейнерам этого же набора при помощи указания имени хоста.
При этом, если не указано явно, то имя хоста будет совпадать с именем сервиса.
Получить ошибку Name or service not known Вы могли в следующих случаях:
1. Вы пытаетесь обратиться к контейнеру postgresql-test из другого набора docker-compose

2. Вы пытаетесь обратиться к контейнеру postgresql-test с хоста (локально), что, в принципе, тоже самое, что и п.1, то есть, не из внутреннего набора
3. У вас дополнительно прописано имя хоста для этого контейнера в бандле. В этом случае нужно обращаться по указанному имени хоста или удалить эту настройку из docker-compose.yml
4. Может быть что-то еще, о чем я не смог сразу вспомнить…

Павел

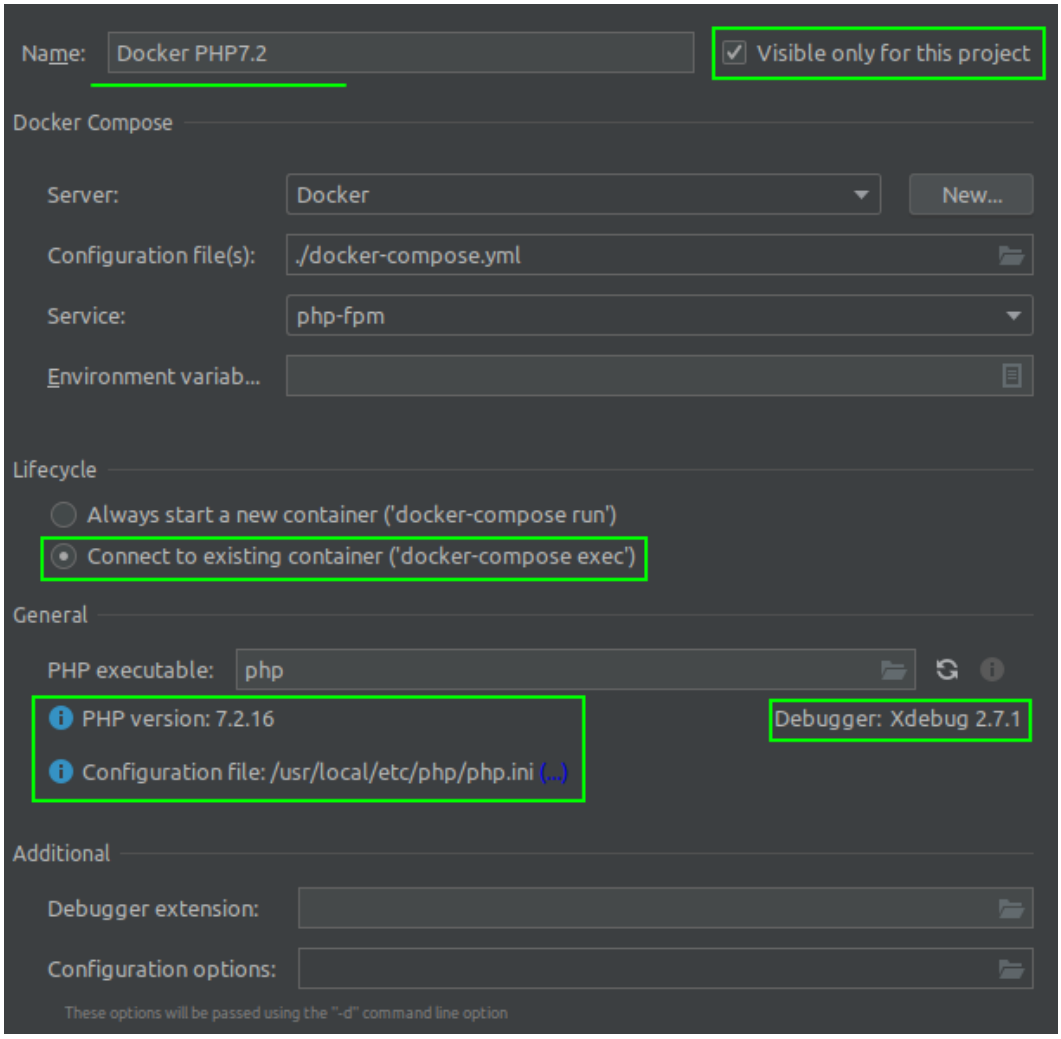
26.06.2019 в 12:06:20

Спасибо за ответ.
Сегодня, понял в чем дело: когда phpstorm запускает тесты через докер, он создает новый контейнер, который ничего не знает про конкретную сеть, которую, я, к примеру, определил в docker-compose.yml. Соответственно, нужно было зайти в Settings>Languages>PHP>Docker Container>Network mode и прописать туда название созданной сети.

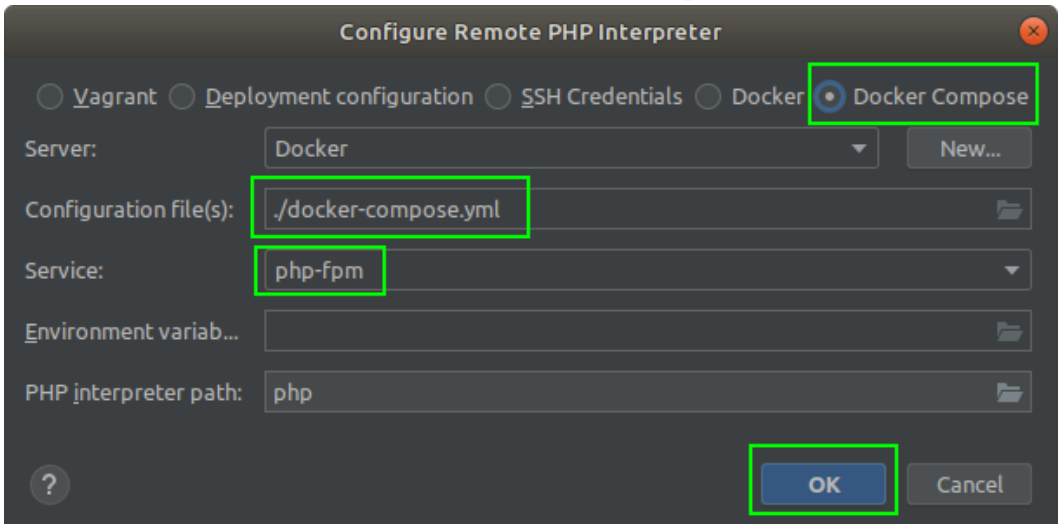
Денис Бондарь ♟

26.06.2019 в 12:17:13

Кажется, я понимаю, что произошло.
Вам достаточно было в настройках интерпретатора PHP указать, что нужно использовать уже существующий контейнер, а не создавать новый.



Также указать, что вы используете бандл через компоуз вот здесь.



Andrey Zausaylov

10.06.2019 в 17:20:39

спасибо! очень помогло

Денис Бондарь ♟

10.06.2019 в 17:25:22

Пожалуйста!

Александр

18.05.2019 в 13:33:07

Денис, спасибо за статью.

У меня появился вопрос: после подключения xdebug, локальный хостинг начал работать в 3-4 раза медленнее. Скажите, есть какой-нибудь вариант, когда опционально можно его включать/выключать на уровне команды запуска(docker-compose up -d)? Или может еще есть какой-нибудь быстрый путь его отключения?

Денис Бондарь ♟

18.05.2019 в 17:45:10

Пожалуйста.

Но вы же xdebug установили внутри контейнера, а не на локальном хостинге. Он работает только внутри контейнера с PHP, в котором вы ведете разработку и не влияет абсолютно ни на что за пределами этого контейнера.

Или что имеется ввиду под «локальный хостинг»?

Александр

18.05.2019 в 18:06:45

У меня операционная система macOS. Внутри докера развертываю веб-сервер, очень похожий на ваш.
Вот собственно после успешного добавления xdebug в контейнер с php, этот сервер начал работать ооооочень мееееедленно. В целом все и так не шустро бегало(я так понимаю потому что не linux на машине), а после добавления расширения совсем «грусть-печаль», wordpress сайты по 10 секунд страницы генерирует и тд.

Пробовал :cache прописывать для volumes — совсем небольшой прирост производительности. Пробовал ресурсов докеру давать по-больше, даже не понял — повлияло ли.

Собственно вопроса 2:

1) Что еще можно попробовать для ускорения работы аналогичного вашему веб-серверу развернутого в docker? docker-sync или еще чего?

2) Если простых/понятных способов нет, то как на уровне правок только лишь docker-compose.yml отключать xdebug из запускаемого контейнера php?
Чтобы завести 2 yml-файла, и по ключу -f запускать сервер без xdebug, когда в работе он не нужен.

Денис Бондарь ♟

18.05.2019 в 22:29:33

Да, это проблема синхронизации файлов через Volumes в MacOS. Серебряной пули в этом случае не существует, насколько я знаю. Но знаю, что в сети достаточно информации про попытки изменить эту ситуацию к лучшему. Все они кажутся в разной степени уродливыми и не решают проблему до конца, но, тем не менее, есть что есть.

1. Я не могу подсказать про MacOS ничего. Я не пользуюсь этой системой и никогда не приходилось сталкиваться с поиском решения подобных проблем.
2. В docker-compose.yml для сервиса php-fpm (или php-cli) передается переменная окружения XDEBUG_CONFIG, значение которой — параметры=значения расширения xdebug через пробел. Отключить работу xdebug можно заменив эту строку на следующую:
   `XDEBUG_CONFIG: "profiler_enable=0 remote_autostart=0 remote_enable=0"`
   В принципе, может быть будет достаточно даже без замены всей строки установить в приведенной в статье строке значение параметра remote_enable=0.

Вы можете также запустить два fpm контейнера, один с дебагом, другой без и настроить в nginx два сервера — каждый на свой fpm и в зависимости от того, нужен xdebug или нет, обращаться к конкретному url.

Александр

19.05.2019 в 00:00:19

Денис, спасибо за ответ!
Последний вариант подходит наилучшим образом.

Денис Бондарь

20.05.2019 в 21:18:49

Пожалуйста. Успехов в разработке!

grts aws

14.05.2019 в 20:29:59

Денис, на одной машине у меня следующая <смертельная> связка: Win7, в котором на vmware установлена Ubuntu. На виртуалке Ubuntu установлен docker. Phpstorm находится на хосте Win7. Как настроить xdebug? Подскажите пожалуйста куда копать. Убил уйму времени а решения не нашел. Спасибо.

Денис Бондарь

14.05.2019 в 20:43:13

xdebug работает следующим образом. Когда запускается на выполнение какой либо php-скрипт, xdebug подключается к удалённому клиенту отладки на порт 9000.
Клиентом, в данном случае, является PHPStorm. Он принимает подключение от xdebug.
Именно для этого важно заранее знать, какой IP-адрес будет назначен хосту. Ведь на этом хосте по этому IP-адресу работает PHPStorm.
Как это работает, если у вас докер установлен на хосте:
`[containter:192.168.222.25 -----> host:192.168.222.1:9000]` (мы заранее знаем, какой адрес докер присвоит хосту в пределах бандла)

В вашей же схеме, когда есть еще один слой (vmware), xdebug должен полдключаться не к хосту (это будет гостевая ОС), а к тому хосту, где работает PHPStorm.
Его IP-адрес вы можете посмотреть в настройках сети на Win7. Это будет IP-адрес бриджевого соединения или тунельного соединения. Смотря, как Вы настроили виртуальную машину vmware.
Когда Вы узнаете этот адрес, попробуйте указать его в docker-compose.yml в параметре `remote_host=xxx.xxx.xxx.xxx` .
Если никаких фаерволов и блокировок нет и если маршрутизация работает правильно (так должно быть по умолчанию), то xdebug без

проблем подключится через всю эту цепочку к вашему PHPStorm. `[containter:192.168.222.25 -----> host_guest_os:192.168.222.1 -----> host:192.168.1.1:9000]` (в данном примере вы должны использовать `remote_host=192.168.1.1`)

---

grts aws

14.05.2019 в 15:27:19

Денис, огромное спасибо.
Тысячу плюсов.
Все завелось с первого раза.

Денис Бондарь

14.05.2019 в 16:32:52

Огромное пожалуйста!
Успехов в разработке!

Александр Кравцов

06.05.2019 в 01:15:02

Супер статья! Спасибо

Денис Бондарь

06.05.2019 в 07:51:34

Пожалуйста ;)

Анатолий Крекотенко

04.04.2019 в 13:30:47

Ребята у кого ни в какую не работает проверьте версию xdebug еще позавчера у меня все работало потом перестали работать brekpoint два дня не спал не мог понять в чем может быть проблема два идентичных проекта, на одном работает корректно на втором нет, уже не знал что делать, по сто раз перенастраивал все заново ничего не помогало. Оказалось docker подтянул последнюю версию xdebug 2.7.0 а у другого проекта осталась старая закешированная версия. Оказывается новая версия xdebug выдает ошибку в phpstorm https://intellij-support.jetbrains.com/hc/en-us/community/posts/360001498520-xdebug-

works-only-with-first-line (последние три комментария). Сменил версию на 2.6.1 и все завелось.

Денис Бондарь

29.04.2019 в 10:13:13

У вас видимо устаревшая версия PHPStorm, не поддерживающая работу с Xdebug 2.7

Андрей

20.03.2019 в 17:37:08

Очень подробная и наглядная статья! Большое спасибо

Денис Бондарь

20.03.2019 в 17:45:22

Пожалуйста. Рад, что статья помогла.

Nikolay

20.03.2019 в 15:33:55

Добрый день, не получается... Прменял хост на .10
http://joxi.ru/a2XbKw7cw08xlr

Nikolay

20.03.2019 в 14:52:58

Подскажите пожалуйста, сделал как в гайде, за исключением того что сменил remote_host на другой. Вот такая штука получается, на ендпоинты не реагирует:
http://joxi.ru/a2XbKw7cw08xlr

Денис Бондарь

20.03.2019 в 15:22:30

Ну, судя из скриншота, Xdebug подключается к шторму нормально.

Nikolay

20.03.2019 в 15:35:29

А в чем может быть проблема не подскажете? Прошу прощения за коммент выше

Денис Бондарь

20.03.2019 в 15:50:41

Честно говоря, даже не знаю в чем может быть проблема.

Nikolay

20.03.2019 в 15:43:28

В логах статус постоянно stopping, а в логах контейнера [20-Mar-2019 13:29:04] WARNING: [pool www] child 6 said into stderr: «There was a problem sending 178 bytes on socket 6: Broken pipe»

Егор

01.03.2019 в 12:58:37

Добрый день. Спасибо за статью.
Вроде бы все настроено правильно. Xdebug работает если запускать php-скрипт.
Но не работает в режиме прослушки как бы я его не крутил с расширением и без, с параметром XDEBUG_SESSION_START, такое чувство что вообще не хочет слушать сайт или ему кто-то мешает это делать.
Куда копать не понимаю

Денис Бондарь

01.03.2019 в 13:07:26

Добрый день. Пожалуйста.
Добавьте логгирование в xdebug.
Для этого в файл docker-compose.yml в значение параметра XDEBUG_CONFIG дополнительно добавьте через пробел:

```
remote_log=/var/www/xdebug.log
```

И пересоберите контейнеры:

```
docker-compose up --build -d
```

В корне проекта должен будет появиться файл xdebug.log, в котором, скорее всего, и найдется ответ.

Егор

01.03.2019 в 14:57:25

Ответа как такого я не вижу.
Опять как я не пробую включить прослушивание , в логи пишется одно и тоже и дебаг молчит
https://i.imgur.com/i9CaCyD.png

Денис Бондарь

01.03.2019 в 16:24:05

Он у вас коннектится на localhost:9000, а должен на хост, указанный в remote_host

---

Егор

01.03.2019 в 16:16:46

Не очень понятно , но отладка сайта заработала после того как посмотрел ip докера через ipconfig и дописал его в docker-compose.yml remote_host=172.17.0.1.
Но теперь не могу настроить маппинг, хотя сопоставления правильное

Денис Бондарь

01.03.2019 в 16:22:50

remote_host должен быть первым из диапазона хостов, указанных в networks файла docker-compose.yml. Если, конечно, у вас не Windows.
Про маппинг не подскажу. Ищите ошибку. Возможно где-то поспешили и недочитали.

Mikhail Zhurov

26.02.2019 в 22:33:32

Спасибо за статью. Набрел на нее в поисках ответа на вопрос, который лишь косвенно связан с содержимым статьи.

xdebug у вас работает на 9000 порту.
Но в docker-compose нет маппинга для этого порта. Согласно документации docker не должен давать пройти пакету из контейнера с php на хостовую систему, если нет настроенного маппинга.

Но если бы вы добавили маппинг 9000:9000 для php-fpm, то возникла бы другая проблема, т.к. phpStorm у вас слушает тоже на 9000 порту. Нельзя использовать один и тот же порт разными утилитами одновременно, пришлось бы маппить 9000 порт из контейнера на другой порт (например 9001) на хосте и настраивать прокси через конфиг DBGProxy в том же phpStorm'е.

НО! У вас все работает. И в общем-то у меня те же результаты без настройки маппинга, xDebug работает прекрасно без проброса 9000 порта на хостовую машину! Как так? Может быть вы исследовали этот вопрос? По какому пути идет пакет от xDebug до 9000 порта на хостовой машине?

Денис Бондарь

27.02.2019 в 00:00:44

Добрый день. Всегда пожалуйста.
Маппинг не нужен по той причине, что не шторм подключается внутрь контейнера, а наоборот — xdebug из контейнера подключается к внешнему узлу — хосту, на которому работает шторм, который и принимает соединение. Все исходящие соединения из контейнера наружу всегда разрешены и не требуют маппинга. Даже, в данном случае, это не соединение наружу, а соединение в пределах широковещательного домена локальной сети. Хост и контейнер находятся внутри одного широковещательного домена (локальной сети) и могут беспрепятственно общаться друг с другом точно так же, как это делают контейнеры между собой. Маппинг нужен чтобы «пробросить» порт наружу — за пределы бандла.

Юрий

31.12.2018 в 16:47:42

Только вот в XDEBUG_CONFIG нужно еще добавить remote_connect_back=1, чтобы заработал дебаггер адекватно, иначе ни xdebug ни jb ide helper не помогают

впрочем настраивать сервер в IDE не обязательно, как и накатывать jetbrains ide helper
достаточно подрубить пыху в IDE, в расширении xdebug включить режим отладки и в ide тоже

тогда все запросы будут перехватываться по брекпоинту, даже аякс

Денис Бондарь

31.12.2018 в 17:34:06

Спасибо за комментарий!
На самом деле, вовсе не обязательно добавлять remote_connect_back=1. Работает и без этого. Ну, или я не понял, что означает «чтобы заработал дебаггер адекватно». Поясните, пожалуйста, разницу.

Сервер в IDE настраивать обязательно, чтобы консольные приложения, включая тесты, запущенные в консоли, также можно было отлаживать. Это необходимо для настройки отображения каталогов на хосте и в контейнере. Про JetBrain Helper в браузере — соглашусь. Устанавливать не обязательно. Но рекомендуется. Кажется, работает и без установки этого плагина, но браузер будет каждый раз рекомендовать установить плагин.

Aleksandr Levashov

16.06.2019 в 20:52:45

При установленном xdebug.remote_connect_back=1 Xdebug будет пытаться установить соединение с IDE по адресу, с которого пришел запрос. Таким образом можно отказаться от настроек сети в docker-compose.yml и отказаться от remote_host.

Денис Бондарь

17.06.2019 в 08:33:16

Большое спасибо за уточнение!!!
Я проверю это в работе и изменю в статье.

yaliilyaya

26.12.2018 в 00:23:30

Вообще супер статья, автору +100500 в карму.
У меня задача немного посложнее будет, думаю развернуть xdebag на удалённом сервере с использованием docker деплой кода(в моём случае тестов) будет осуществляться по средствам ssh (также должен являться контейнером)

всё это после оберну в кубернетис и прикреплю к git CI.

Денис Бондарь

26.12.2018 в 09:58:24

Спасибо!
Звучит очень интересно. Если потом сможете об этом написать — с удовольствием добавлю себе в закладки такой опыт!
Успехов Вам!

Artem

10.12.2018 в 20:58:31

Очень круто! Я имею богатый опыт освоения новых инструментов и хорошо вникаю по английским мануалам. Но в этой связке я всю голову сломал от разного рода мелких нестыковок. По вашей статье я нашел все свои ошибки в настройках и сразу все заработало. Посему считаю этот материал отменным! В такие моменты, офигеваю от возможностей phpStorm, и он все-таки стоит своих денег =)))

Денис Бондарь

11.12.2018 в 11:42:15

Спасибо за отзыв!

Некто

29.11.2018 в 00:14:20

Благодарствую за статью.
Перерыл кучу исходников с километровыми портянками, чувствую начал терять кучу времени.
У Вас все просто и понятно.
Не все пишут, но думаю, многие начинающие благодарны.

Денис Бондарь

29.11.2018 в 10:11:28

Пожалуйста!
И вам спасибо за отзыв.

Иван

22.10.2018 в 23:58:39

Спасибо большое! Очень пригодилась статья

Денис Бондарь

23.10.2018 в 18:08:28

Всегда пожалуйста!