2.TARS PHP TCP Server & Client Development

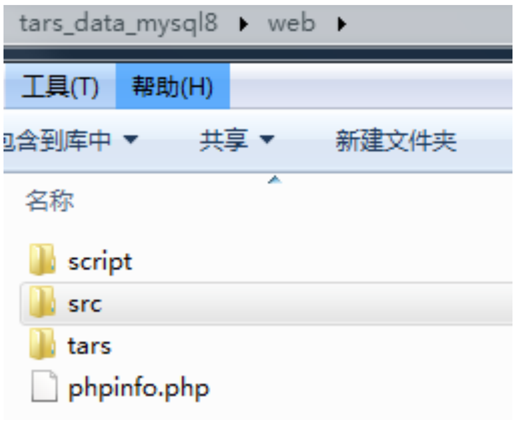
PHP Server Side Development

Here we need to use tag `php7mysql8` of image `tangramor/docker-tars` to develop PHP server. Here we assume that you are using Windows:

```
docker run --name mysql8 -e MYSQL_ROOT_PASSWORD=password -d -p 3306:3306 -v /
c/Users/tangramor/mysql8_data:/var/lib/mysql mysql:8 --sql_mode="" --
innodb_use_native_aio=0

docker run -d -it --name tars_mysql8 --link mysql8 --env DBIP=mysql8 --env
DBPort=3306 --env DBUser=root --env DBPassword=password -p 3000:3000 -p 80:80
-v /c/Users/tangramor/tars_mysql8_data:/data tangramor/docker-tars:php7mysql8
```

The above 2 commands start 2 containers: a v8.0 mysql and `tangramor/docker-tars:php7mysql8` container with name **tars_mysql8**, and we mount the folder of host machine `/c/Users/tangramor/Workspace/tars_mysql8_data` as /data folder of container **tars_mysql8**. It also exposes port 3000 and 80.

Enter `/c/Users/tangramor/Workspace/tars_mysql8_data/web` and create folders: `scripts` 、 `src` and `tars` :



Run `docker exec -it tars_mysql8 bash` to enter container **tars_mysql8** and `cd /data/web` .

Create file `test.tars` under `tars` folder (Refer: phptars example):

```
module testtafserviceservant
{
    struct SimpleStruct {
        0 require long id=0;
        1 require int count=0;
        2 require short page=0;
    };

    struct OutStruct {
        0 require long id=0;
        1 require int count=0;
        2 require short page=0;
        3 optional string str;
    };

    struct ComplicatedStruct {
        0 require vector<SimpleStruct> ss;
        1 require SimpleStruct rs;
        2 require map<string, SimpleStruct> mss;
        3 optional string str;
    }

    struct LotofTags {
        0 require long id=0;
        1 require int count=0;
        2 require short page=0;
        3 optional string str;
        4 require vector<SimpleStruct> ss;
        5 require SimpleStruct rs;
        6 require map<string, SimpleStruct> mss;
    }

    interface TestTafService
    {

        void testTafServer();

        int testLofofTags(LotofTags tags, out LotofTags outtags);

        void sayHelloWorld(string name, out string outGreetings);

        int testBasic(bool a, int b, string c, out bool d, out int e, out
string f);

        string testStruct(long a, SimpleStruct b, out OutStruct d);

        string testMap(short a, SimpleStruct b, map<string, string> m1, out
OutStruct d, out map<int, SimpleStruct> m2);

        string testVector(int a, vector<string> v1, vector<SimpleStruct> v2,
out vector<int> v3, out vector<OutStruct> v4);

        SimpleStruct testReturn();

        map<string,string> testReturn2();

        vector<SimpleStruct> testComplicatedStruct(ComplicatedStruct
cs,vector<ComplicatedStruct> vcs, out ComplicatedStruct ocs,out
vector<ComplicatedStruct> ovcs);

        map<string,ComplicatedStruct> testComplicatedMap
(map<string,ComplicatedStruct> mcs, out map<string,ComplicatedStruct> omcs);

        int testEmpty(short a,out bool b1, out int in2, out OutStruct d, out
vector<OutStruct> v3,out vector<int> v4);

        int testSelf();

        int testProperty();

    };

}
```

Create file `tars.proto.php` under `tars`:

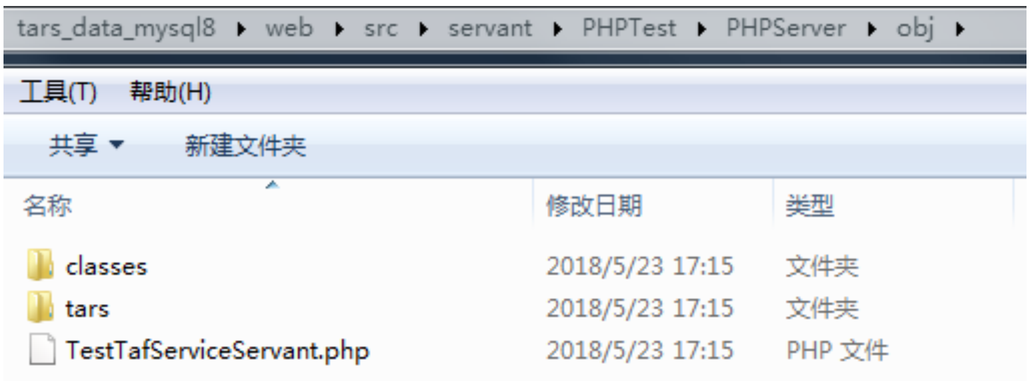
```
<?php

  return array(
      'appName' => 'PHPTest',
      'serverName' => 'PHPServer',
      'objName' => 'obj',
      'withServant' => true, //true to generate server side code, false for
client side code
      'tarsFiles' => array(
          './test.tars'
      ),
      'dstPath' => '../src/servant',
      'namespacePrefix' => 'Server\servant',
  );
```

Create `tars2php.sh` under `scripts` and give execution permission `chmod u+x tars2php.sh`:

```
cd ../tars/

php /root/phptars/tars2php.php ./tars.proto.php
```

Create folder `src/servant`, then run `./scripts/tars2php.sh`, you will see there are 3 layers folders generated under `src/servant`: `PHPTest/PHPServer/obj`, it includes:

- classes folder - To store the generated tars structs
- tars folder - To store the original tars file
- TestTafServiceServant.php - interface


tars_data_mysql8 ▸ web ▸ src ▸ servant ▸ PHPTest ▸ PHPServer ▸ obj ▸
工具(T) 帮助(H)
共享 ▾ 新建文件夹

名称	修改日期	类型
classes	2018/5/23 17:15	文件夹
tars	2018/5/23 17:15	文件夹
TestTafServiceServant.php	2018/5/23 17:15	PHP 文件

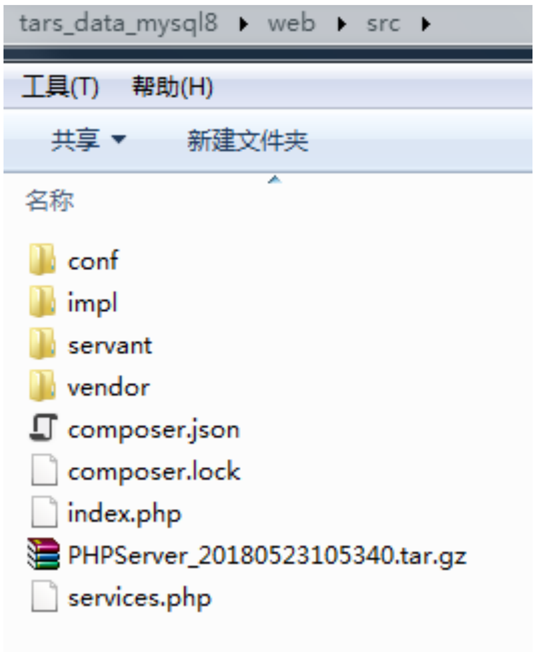
Enter `src` folder, we begin to implemente the server side logic. Because we are using the official example, here we copy the source code directly from example project:

```
wget https://github.com/Tencent/Tars/raw/master/php/examples/tars-tcp-server/
src/composer.json
wget https://github.com/Tencent/Tars/raw/master/php/examples/tars-tcp-server/
src/index.php
wget https://github.com/Tencent/Tars/raw/master/php/examples/tars-tcp-server/
src/services.php
mkdir impl && cd impl && wget https://github.com/Tencent/Tars/raw/master/php/
examples/tars-tcp-server/src/impl/PHPServerServantImpl.php && cd ..
mkdir conf && cd conf && wget https://github.com/Tencent/Tars/raw/master/php/
examples/tars-tcp-server/src/conf/ENVConf.php && cd ..
```

- conf: configurations for implementation, here we just give a demo. If you push config from Tars platform, the file will be written into this folder.
- impl: implementation code for interface. And the address of the implementation will be written in servcies.php.

- composer.json: dependencies of the project.
- index.php: entrance file of the service. You can use another name, and you need to change the deployment template on Tars platform, by adding `entrance` field under `server` .
- services.php: diclare the addresses of interface and implementation, and they will be used for instantiate and annotation parsing.

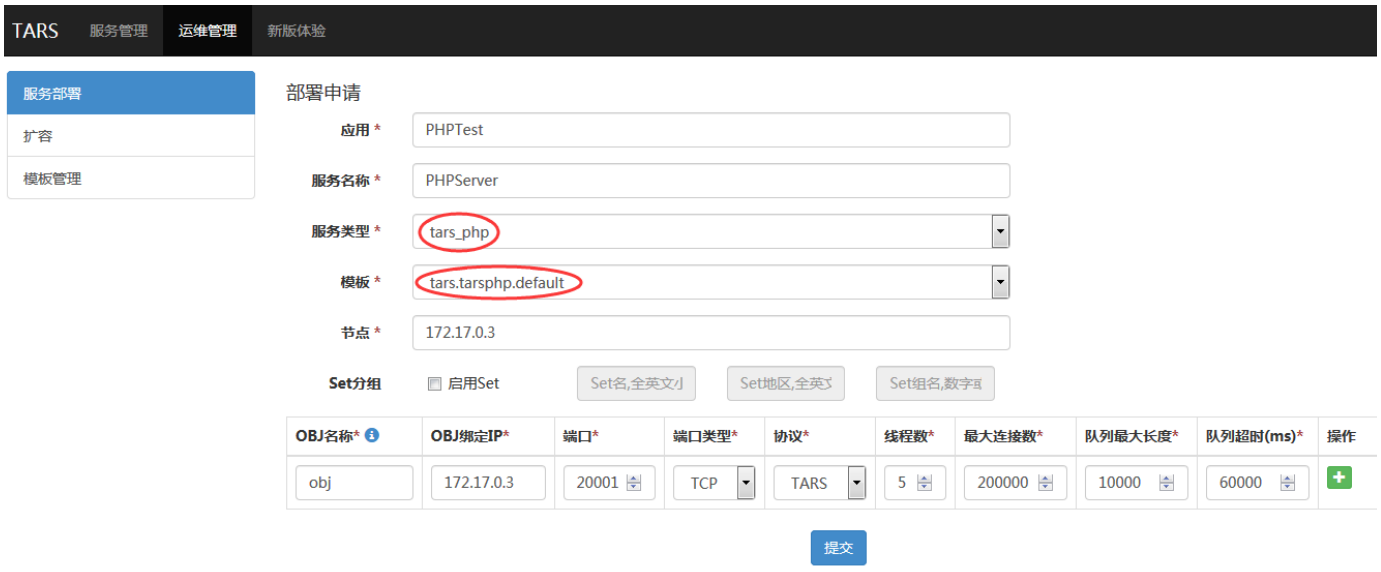
Change the configuration in `conf/ENVConf.php` . And execute `composer install` under `src` to load required dependencies, then run `composer run-script deploy` to build the package, and a package name like `PHPServer_20180523105340.tar.gz` will be generated.



```
conf
impl
servant
vendor
composer.json
composer.lock
index.php
PHPServer_20180523105340.tar.gz
services.php
```

Create a `logs` folder under `/data` , because this example will write file under it.

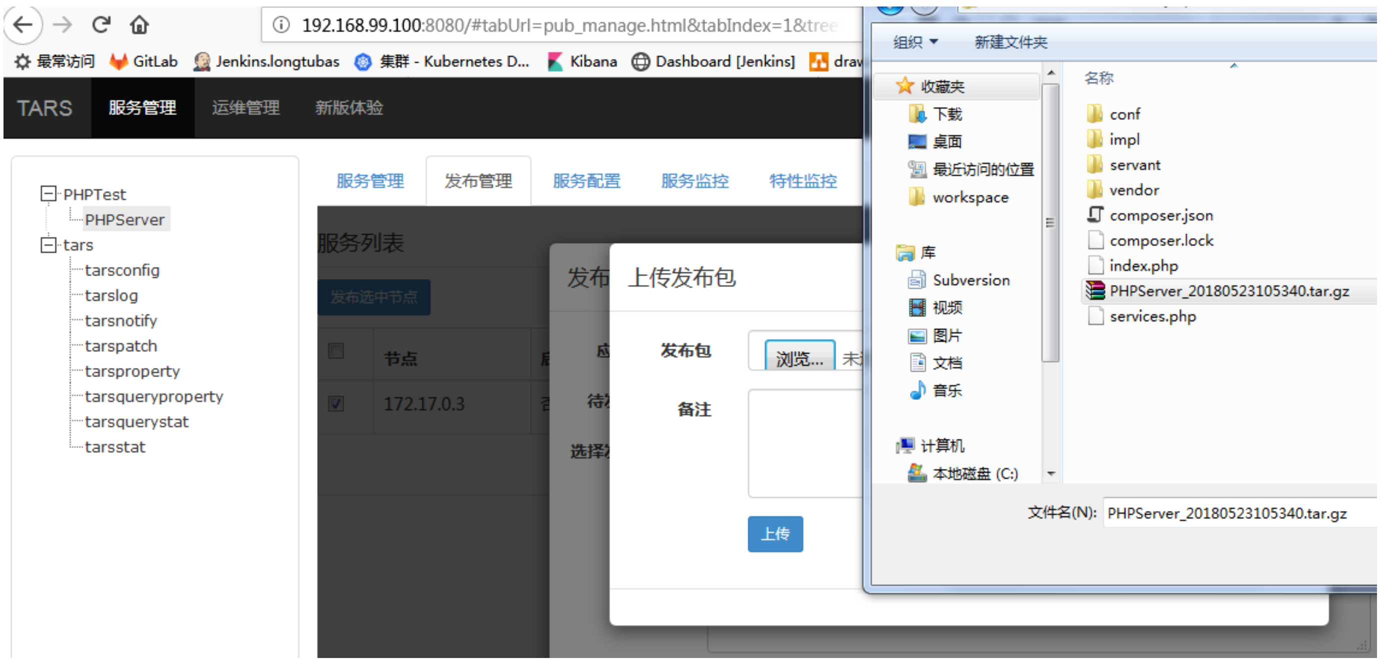
Deploy the generated package to Tars platform. Remember to use tars-php type and use `tars.tarsphp.default` template (or create a template by yourself):



部署申请

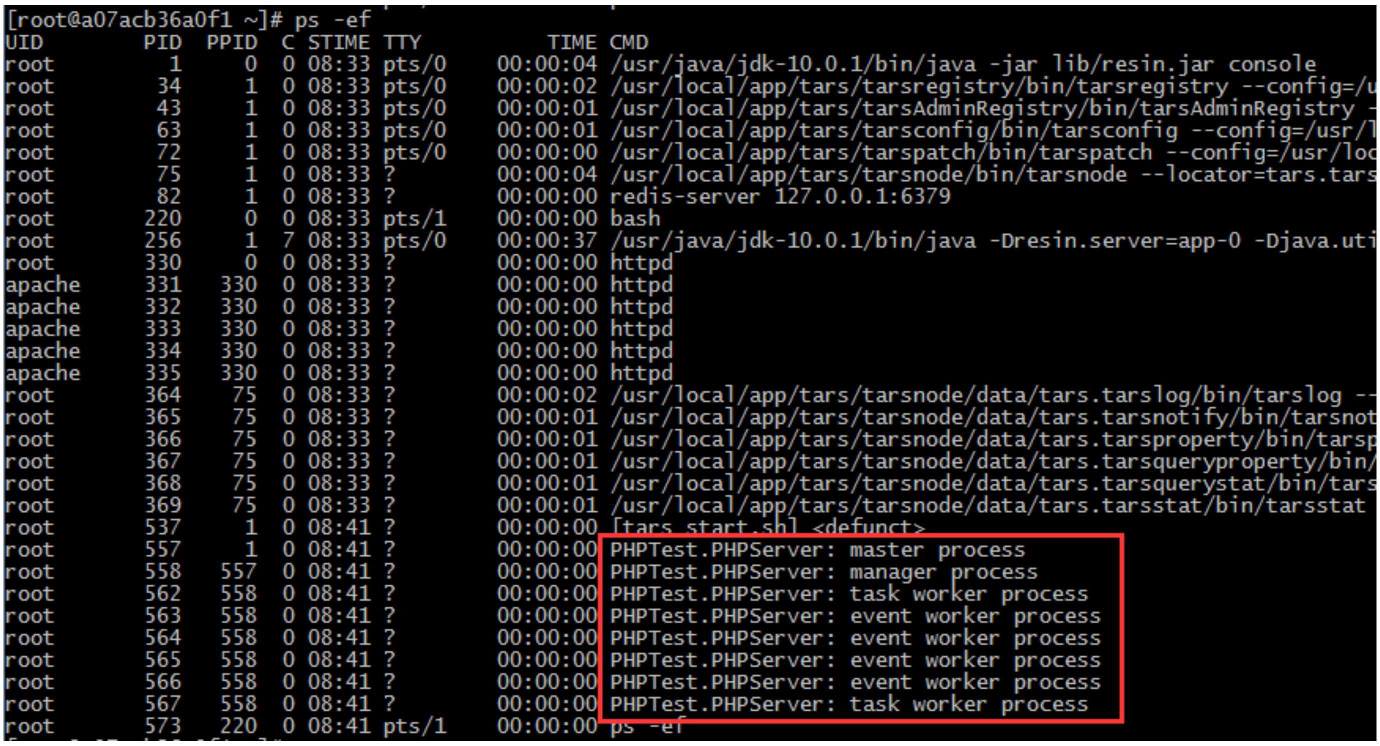
应用 *	PHPTest
服务名称 *	PHPServer
服务类型 *	tars_php
模板 *	tars.tarsphp.default
节点 *	172.17.0.3
Set分组	启用Set

OBJ名称*	OBJ绑定IP*	端口*	端口类型*	协议*	线程数*	最大连接数*	队列最大长度*	队列超时(ms)*	操作
obj	172.17.0.3	20001	TCP	TARS	5	200000	10000	60000	➕



文件名(N): PHPServer_20180523105340.tar.gz

Once the deployment is successfully completed, you will see related processes when run `ps -ef`.



```
[root@a07acb36a0f1 ~]# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 08:33 pts/0    00:00:04 /usr/java/jdk-10.0.1/bin/java -jar lib/resin.jar console
root        34     1  0 08:33 pts/0    00:00:02 /usr/local/app/tars/tarsregistry/bin/tarsregistry --config=/u
root        43     1  0 08:33 pts/0    00:00:01 /usr/local/app/tars/tarsAdminRegistry/bin/tarsAdminRegistry -
root        63     1  0 08:33 pts/0    00:00:01 /usr/local/app/tars/tarsconfig/bin/tarsconfig --config=/usr/l
root        72     1  0 08:33 pts/0    00:00:00 /usr/local/app/tars/tarspatch/bin/tarspatch --config=/usr/loc
root        75     1  0 08:33 ?        00:00:04 /usr/local/app/tars/tarsnode/bin/tarsnode --locator=tars.tars
root        82     1  0 08:33 ?        00:00:00 redis-server 127.0.0.1:6379
root       220     0  0 08:33 pts/1    00:00:00 bash
root       256     1  7 08:33 pts/0    00:00:37 /usr/java/jdk-10.0.1/bin/java -Dresin.server=app-0 -Djava.uti
root       330     0  0 08:33 ?        00:00:00 httpd
apache     331   330  0 08:33 ?        00:00:00 httpd
apache     332   330  0 08:33 ?        00:00:00 httpd
apache     333   330  0 08:33 ?        00:00:00 httpd
apache     334   330  0 08:33 ?        00:00:00 httpd
apache     335   330  0 08:33 ?        00:00:00 httpd
root       364    75  0 08:33 ?        00:00:02 /usr/local/app/tars/tarsnode/data/tars.tarslog/bin/tarslog --
root       365    75  0 08:33 ?        00:00:01 /usr/local/app/tars/tarsnode/data/tars.tarsnotify/bin/tarsnot
root       366    75  0 08:33 ?        00:00:01 /usr/local/app/tars/tarsnode/data/tars.tarsproperty/bin/tarsp
root       367    75  0 08:33 ?        00:00:01 /usr/local/app/tars/tarsnode/data/tars.tarsqueryproperty/bin/
root       368    75  0 08:33 ?        00:00:01 /usr/local/app/tars/tarsnode/data/tars.tarsquerystat/bin/tars
root       369    75  0 08:33 ?        00:00:01 /usr/local/app/tars/tarsnode/data/tars.tarsstat/bin/tarsstat
root       537     1  0 08:41 ?        00:00:00 [tars_start.sh] <defunct>
root       557     1  0 08:41 ?        00:00:00 PHPTest.PHPServer: master process
root       558   557  0 08:41 ?        00:00:00 PHPTest.PHPServer: manager process
root       562   558  0 08:41 ?        00:00:00 PHPTest.PHPServer: task worker process
root       563   558  0 08:41 ?        00:00:00 PHPTest.PHPServer: event worker process
root       564   558  0 08:41 ?        00:00:00 PHPTest.PHPServer: event worker process
root       565   558  0 08:41 ?        00:00:00 PHPTest.PHPServer: event worker process
root       566   558  0 08:41 ?        00:00:00 PHPTest.PHPServer: event worker process
root       567   558  0 08:41 ?        00:00:00 PHPTest.PHPServer: task worker process
root       573   220  0 08:41 pts/1    00:00:00 ps -ef
```

PHP Client Side Development

We will develop the client side code in the same container.

Enter `/c/Users/tangramor/Workspace/tars_mysql8_data/web` and create `client` folder under it.

Run `docker exec -it tars_mysql8 bash` to enter container **tars_mysql8**, and `cd /data/web/client`.

Copy the `test.tars` file created in 3. **PHP Server Side Development** to current folder, and create file `tarsclient.proto.php`:

```
<?php

  return array(
      'appName' => 'PHPTest',
      'serverName' => 'PHPServer',
      'objName' => 'obj',
      'withServant' => false, //true to generate server side code, false for
client side code
      'tarsFiles' => array(
          './test.tars'
```

```
        ),
        'dstPath' => './',
        'namespacePrefix' => 'Client\servant',
    );
```

Run `php /root/phptars/tars2php.php ./tarsclient.proto.php`, and you will see there are 3 layers folders generated: `PHPTest/PHPServer/obj`, it includes:

- classes folder - To store the generated tars structs
- tars folder - To store the original tars file
- TestTafServiceServant.php - client class TestTafServiceServant

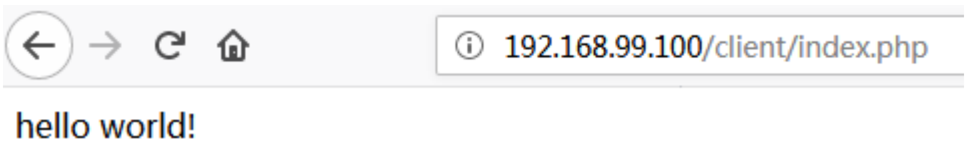
Create file `composer.json`:

```
{
  "name": "demo",
  "description": "demo",
  "authors": [
    {
      "name": "Tangramor",
      "email": "tangramor@qq.com"
    }
  ],
  "require": {
    "php": ">=5.3",
    "phptars/tars-client" : "0.1.1"
  },
  "autoload": {
    "psr-4": {
      "Client\\servant\\": "./"
    }
  },
  "repositories": {
    "tars": {
      "type": "composer",
      "url": "https://raw.githubusercontent.com/Tencent/Tars/master/php/dist/
tarsphp.json"
    }
  }
}
```

And create `index.php` file:

```
<?php
  require_once("./vendor/autoload.php");

  // ip、port
  $config = new \Tars\client\CommunicatorConfig();
  $config->setLocator('tars.tarsregistry.QueryObj@tcp -h 172.17.0.3 -p
17890');
  $config->setModuleName('PHPTest.PHPServer');
  $config->setCharsetName('UTF-8');

  $servant = new Client\servant\PHPTest\PHPServer\obj\TestTafServiceServant
($config);

  $name = 'ted';
  $intVal = $servant->sayHelloWorld($name, $greetings);

  echo '<p>'.$greetings.'</p>';
```

Run `composer install` to load required dependencies, then execute `php index.php` to test our client. If everything good, it should output: `<p>hello world!</p>` . We use a web browser to visit http://192.168.99.100/client/index.php and should see page:



hello world!

Check `ted.log` under `/data/logs` , there should be content written: `sayHelloWorld name:ted` .