

2. TARS PHP TCP服务端与客户端开发

开发PHP服务端

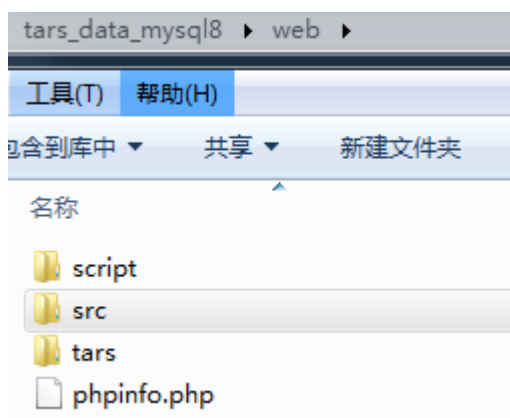
我们使用标签为 `php7mysql8` 的 `tangramor/docker-tars` 镜像来进行PHP服务端的开发（假设你用的是Windows）：

```
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
```

这两个命令分别启动了mysql容器（8.0版）和 `tangramor/docker-tars:php7mysql8` 容器 `tars_mysql8`，并将本地的一个目录 `/c/Users/tangramor/Workspace/tars_mysql8_data` 挂载为容器的 `/data` 目录，同时它还把 3000 和 80 端口暴露出来了。

我们进入 `/c/Users/tangramor/Workspace/tars_mysql8_data/web` 目录，在其下创建对应的目录结构：`scripts`、`src` 和 `tars`，



运行 `docker exec -it tars_mysql8 bash` 进入容器 `tars_mysql8`，`cd /data/web` 进入工作目录。

在 `tars` 目录下创建一个 `test.tars` 文件（参考 [phptars例子](#)）：

```
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();

};

}
```

然后再在 `tars` 目录下创建一个 `tars.proto.php` 文件：

```
<?php

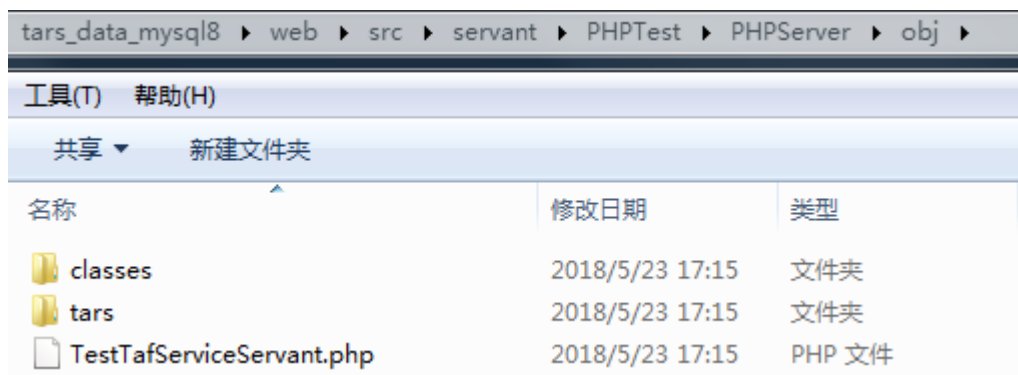
return array(
    'appName' => 'PHPTest', //tars服务servant name 的第一部分
    'serverName' => 'PHPServer', //tars服务servant name 的第二部分
    'objName' => 'obj', //tars服务servant name 的第三部分
    'withServant' => true, //决定是服务端,还是客户端的自动生成
    'tarsFiles' => array(
        './test.tars' //tars文件的地址
    ),
    'dstPath' => '../src/servant', //生成php文件的位置
    'namespacePrefix' => 'Server\servant', //生成php文件的命名空间前缀
);
```

在 `scripts` 目录下创建 `tars2php.sh`，并赋予执行权限 `chmod u+x tars2php.sh`：

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

创建目录 `src/servant`，然后执行 `./scripts/tars2php.sh`，可以看到 `src/servant` 目录下面生成一个三级文件夹 `PHPTest/PHPServer/obj`，包含：

- `classes`文件夹 - 存放tars中的struct生成的文件
- `tars`文件夹 - 存放tars文件
- `TestTafServiceServant.php` - interface文件



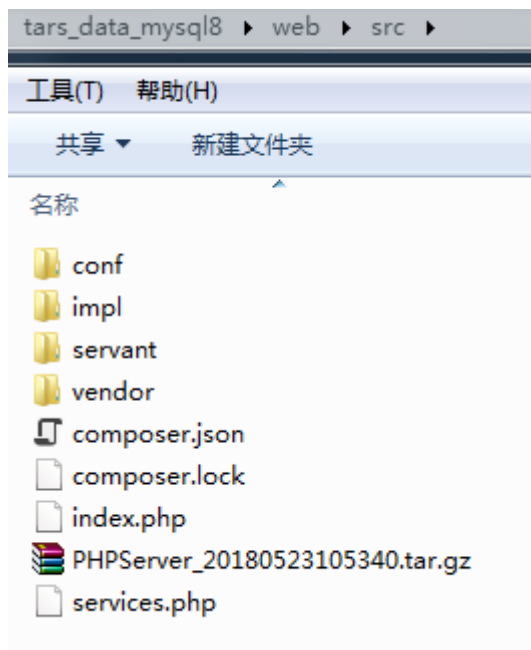
进入 `src` 目录，我们开始服务端代码的实现。因为使用的是官方例子，所以这里直接将例子的实现代码拷贝过来：

```
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`: 业务需要的配置，这里只是给出一个demo，如果从平台下发配置，默认也会写入到这个文件夹中；
- `impl`: 业务实际的实现代码，是对interface的对应实现，具体实现的文件路径需要写入到 `services.php`中；

- `composer.json`: 项目的依赖；
- `index.php`: 整个服务的入口文件。可以自定义，但是必须要更改平台上的私有模板，在`server`下面增加`entrance`这个字段；
- `services.php`: 声明`interface`的地址，声明实际实现的地址，这两个地址会被分别用作实例化调用和注解解析。

修改一下 `conf/ENVConf.php` 的配置信息。在 `src` 目录下运行 `composer install` 加载对应的依赖包，然后执行 `composer run-script deploy` 进行代码打包，一个名字类似 `PHPServer_20180523105340.tar.gz` 的包就打好了。



在 `/data` 目录下创建一个 `logs` 目录，因为这个例子会在这下面写文件。

将打好的包发布到Tars平台，记得选择php方式，模版使用 `tars.tarsphp.default` 或者自己根据需求新建一个模版：

TARS 服务管理 运维管理 新版体验

服务部署

部署申请

应用 * PHPTest

服务名称 * PHPServer

服务类型 * tars_php

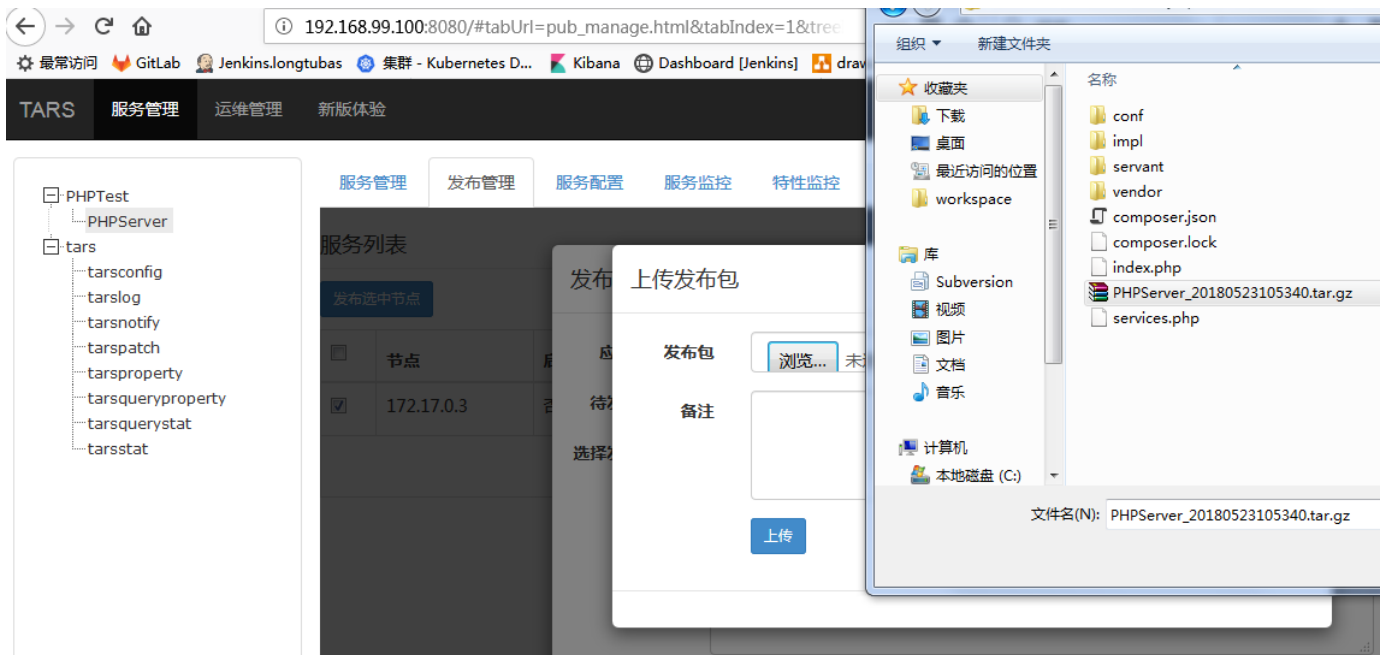
模板 * tars.tarsphp.default

节点 * 172.17.0.3

Set分组 启用Set Set名,全英文 Set地区,全英文 Set组名,数字

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	+

提交



发布成功后，在系统里执行 `ps -ef` 会发现相关的进程。

```
[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.ut
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
```

开发PHP客户端

我们在同一个容器里进行上面服务端的客户端开发和测试，当然你也可以自己创建一个新的容器来尝试。

我们进入 `/c/Users/tangramor/Workspace/tars_mysql8_data/web` 目录，在其下创建对应的目录 `client`。

运行 `docker exec -it tars_mysql8 bash` 进入容器 `tars_mysql8`，`cd /data/web/client` 进入工作目录。

将 3. 开发PHP服务端 里创建的 `test.tars` 文件拷贝到当前目录，然后创建一个文件 `tarsclient.proto.php`：

```
<?php

return array(
    'appName' => 'PHPTest',
    'serverName' => 'PHPServer',
    'objName' => 'obj',
    'withServant' => false, //决定是服务端, 还是客户端的自动生成
    'tarsFiles' => array(
```

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

运行 `php /root/phptars/tars2php.php ./tarsclient.proto.php` , 可以看到在当前目录下生成了一个三级文件夹 `PHPTest/PHPServer/obj` , 包含 :

- classes文件夹 - 存放tars中的struct生成的文件
- tars文件夹 - 存放tars文件
- TestTafServiceServant.php - 客户端类 TestTafServiceServant 文件

在当前目录创建一个 `composer.json` 文件 :

```
{  
  "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"  
    }  
  }  
}
```

然后再创建一个 `index.php` 文件 :


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

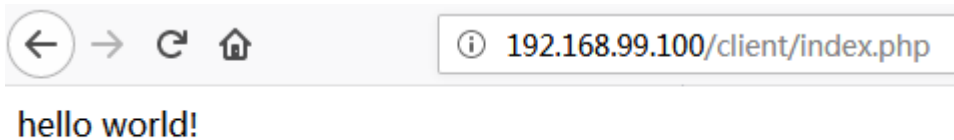
// 指定主控ip、port
$config = new \Tars\client\CommunicatorConfig();
$config->setLocator('tars.tarsregistry.QueryObj@tcp -h 172.17.0.3 -p
17890');
$config->setModuleName('PHPTest.PHPServer');
$config->setCharsetName('UTF-8');

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

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

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

执行 `composer install` 命令加载对应的依赖包，然后运行 `php index.php` 来测试客户端，如果一切顺利，应该输出：`<p>hello world!</p>`。我们使用浏览器来访问 <http://192.168.99.100/client/index.php>，应该也能看到：



在 `/data/logs` 目录下查看 `ted.log`，应该有内容写入：`sayHelloWorld name:ted`。