# 1.TARS CPP Server & Client Development

## C++ Server Side Development

Start docker container with following command. Here we can use image `tarscloud/tars` or `tangramor/docker-tars`.

```
docker run -d --name mysql \
    -e MYSQL_ROOT_PASSWORD=password -p 3306:3306 \
    -v /c/Users/tangramor/mysql_data:/var/lib/mysql \
    mysql:5.6 --innodb_use_native_aio=0

docker run -d -it --name tars -p 3000:3000 \
    -v /c/Users/tangramor/Workspace/tars_data:/data \
    tarscloud/tars
```

This command starts `tarscloud/tars` to container **tars** and mount local folder `/c/Users/tangramor/Workspace/tars_data` as /data folder in the container. It also exposes port 3000.

We can see that there are 2 new folders, log and tars, created under `/c/Users/tangramor/Workspace/tars_data` in our host OS. Folder log is to store resin log and folder tars contains the log folders of Tars sub systems. In the mean time we can find tgz packages under `/c/Users/tangramor/Workspace/tars_data` which have already been installed in the container.

Execute `docker exec -it tars bash` to enter container **tars**, `cd /data` to the work directory, and we can refer to Service Development to develop TestApp.HelloServer. We need to modify method testHello to following:

```
int HelloImp::testHello(const std::string &sReq, std::string &sRsp,
tars::TarsCurrentPtr current)
{
    TLOGDEBUG("HelloImp::testHellosReq:"<<sReq<<endl);
    sRsp = sReq + " World!";
    return 0;
}
```

Then we deploy the compiled HelloServer.tgz to our **tars** container.

# PHP Client of C++ Server Development

C++ client can be done by referring to Sync/Async calling to Service from Client. Be aware that if you want to deploy C++ client to tars-node container, you should not mix `minideb` tag with `latest` and `php7` tags, because there will be dependency problem for different OSs.

Here I will introduce how to develop PHP client and deploy it. First we create a docker image base on `tarscloud/tars-node:php`:

**Dockerfile**

```
FROM tarscloud/tars-node:php

RUN yum -y install php httpd \
  && rm -rf /var/www/html && ln -s /data /var/www/html \
  && sed -i "s/tail -f \/dev\/null/rm -rf \/var\/run\/httpd\/*\n\t\thttpd\n\t
\ttail -f \/dev\/null/g" /sbin/entrypoint.sh

ENTRYPOINT [ "/sbin/entrypoint.sh", "start" ]
```

Use command to build image: `docker build -t tars-node-php .`

Start the container:

```
docker run -d -it --name tars-node --link tars -e MASTER=tars -p 80:80 -v /c/
Users/tangramor/Workspace/tars_node:/data tars-node-php
```

This command starts `tars-node-php` to container **tars-node** and mount local folder `/c/Users/tangramor/Workspace/tars_node` as /data folder in the container. It also exposes port 80.

Find `Hello.tars` from `/c/Users/tangramor/Workspace/tars_data/TestApp/HelloServer` in host OS, and copy it to `/c/Users/tangramor/Workspace/tars_node/web`.

Execute `docker exec -it tars-node bash` to enter container **tars-node**, `cd /data` to web folder, and create a file with name `tarsclient.proto.php`:

```
<?php

  return array(
      'appName' => 'TestApp',
      'serverName' => 'HelloServer',
      'objName' => 'HelloObj',
      'withServant' => false,  //true to generate server side code, false for
client side code
      'tarsFiles' => array(
          './Hello.tars'
      ),
      'dstPath' => './',
      'namespacePrefix' => '',
  );
```

Then run `php /root/phptars/tars2php.php ./tarsclient.proto.php`, we can see that TestApp folder is created, and under `TestApp/HelloServer/HelloObj` we can find the generated client files.

Create `composer.json` file under web folder:

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

Execute `composer install`, we can see `vendor` folder is created. That means we can use autoload in PHP files to load phptars. Create a file named `index.php` under web folder:

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

    $config = new \Tars\client\CommunicatorConfig();
    $config->setLocator("tars.tarsregistry.QueryObj@tcp -h 172.17.0.3 -p 
17890");
    $config->setModuleName("TestApp.HelloServer");
    $config->setCharsetName("UTF-8");
    $servant = new \TestApp\HelloServer\HelloObj\HelloServant($config);

    $start = microtime();

    try {
        $in1 = "Hello";

        $intVal = $servant->testHello($in1,$out1);

        echo "Server returns: ".$out1;

    } catch(phptars\TarsException $e) {
        echo "Error: ".$e;
    }

    $end = microtime();

    echo "<p>Elapsed time: ".($end - $start)." seconds</p>";
```

Use a browser in host OS to visit http://127.0.0.1/index.php (in Linux or Mac) or http://192.168.99.100/index.php (in Windows), you should see result like following:

```
Server returns: Hello World!

Elapsed time: 0.051169 seconds
```