Project to implement remote PHP web client/ MSSQL database server network setup

fortuneWalla
August 16th 2015

Created on Friday, May 16th, 2014 at 2:43 pm

For students learning data science/analytics, accessing data sets stored from databases in remote servers is a necessary skill. People in corporations/institutions have Systems Administrators to set up most of the interfaces for such purposes. However knowing how such a process is setup is a good knowledge. Here PHP is used to demonstrate database access.

Although this project is not related to data science/analytics, it will give a good hands-on experience of client/server networking. These are mere guidelines and not details as specific options & settings will depend on the version of the software & components.

Pre-requisites: Fairly good understanding of computer hardware especially networking, Windows operating system, drivers & software installation.

Hardware/Networking

Physical: Atleast two computers with Ethernet/WiFi capability and a wireless/wired router.

Final Setup: Ability to create Workgroups & shared folders. Also ability to `ping` & `telnet` the remote computers/servers.

Server Setup

There are many versions & editions MSSQL depending on user requirements (The Developer Edition is best for learning & academic use). For data mining features, SQL Server Analysis Services(SSAS) must be installed. To install OLTP/DW AdventureWorks sample databases, enable FILESTREAM and make sure the SQL Full-text Filter Daemon Launcher service is running.

1) If you have the space, do a full install of every feature. It will make it easier to troubleshoot problems.
2) Create a MSSQL user exclusively for remote access. http://technet.microsoft.com/en-us/library/aa337545.aspx
3) Allow remote connection to MSSQL. http://technet.microsoft.com/en-us/library/ms191464.aspx
4) Enable Shared Memory, TCP/IP, pipes http://technet.microsoft.com/en-us/library/ms181035.aspx
5) In the Windows Firewall, allow incoming/outgoing connections for port 1433 (default

MSSQL port.) http://blogs.msdn.com/b/walzenbach/archive/2010/04/14/how-to-enable-remote-connections-in-sql-server-2008.aspx

Final Setup: From the Command Prompt in the client, execute `ping dbpcip` and `telnet dbpcip 1433` where `dbpcip` is the IP address of MSSQL machine. If you can connect with both the commands, the client/server networking is probably working. Although accessing the datebase & tables finally depends on the permissions assigned to the remote user.

Client Setup

Keep these details in mind before starting installation.
1) Some of the types of PHP functions for access to MSSQL. This article only discusses `mssql_()` and `sqldrv_()`.

- `mssql_()` Supported by PHP 5.2 (php52) and lower. MSSQL Driver for PHP (SSDPHP) not required
- `sqldrv_()` Supported by PHP 5.3 (php53) and higher. SSDPHP required.
- `pdo_sqlsrv_()` Supported by php52 and higher. SSDPHP required.
- `pdo_odbc_()` Supported by php51 and higher. SSDPHP not required.

2) SQL Server Native Client (SSNC): This is needed by PHP to access MSSQL Server.
SSNC2008: installs on winxp
SSNC2012: does not install on winxp.
php52: requires atleast SSNC2008
php53: requires atleast SSNC2012

3) Problem: php53 onwards requires SSNC2012, but SSNC2012 does not work on winxp. Also php53 does not support `mssql_()` functions.

Solution: SSNC2008 works on winxp. Use php 5.2.13 to 5.2.17 as php52 requires SSNC2008. Also php 5.2.x supports `mssql_()` functions and `sqldrv_()` functions if you install the SSDPHP 2.0.

Client software

1) Internet Information Services (IIS):
Official Documentation:
https://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/iiiisin2.mspx?mfr=true

Step-by-step illustrated guide: Follow until step 10 http://www.wikihow.com/Configure-IIS-for-Windows-XP-Pro

You can test if it is working by opening the `http://localhost/` address on the web browser. You should also test it on a remote computer using `http://clientip/`

Final Setup:
Windows Version: XP
IIS Version: 5.1

2) FastCGI:

FastCGI helps IIS work better with PHP. Install the one which is compatible with your version of IIS & OS.
Official source: http://www.iis.net/downloads/microsoft/fastcgi-for-iis

Final Setup:fcgisetup_1.5_rtw_x86.msi

3) PHP installation:

a) Steps to installing PHP with IIS are explained in www.php.net/manual/en/install.windows.iis6.php
b) 5.2 and lower `mssql_()` documentation http://www.php.net/manual/en/book.mssql.php
c) 5.3 and higher `sqlsrv_()` documentation http://www.php.net/manual/en/book.sqlsrv.php
c) 5.2 and higher `pdo_sqlsrv()` documentation http://www.php.net/manual/en/ref.pdo-sqlsrv.php
d) 5.1 and higher `pdo_odbc()` documentation http://www.php.net/manual/en/ref.pdo-odbc.php

Final Setup: php-5.2.17-nts-Win32-VC6-x86.msi

4) MSSQL Driver for PHP:

Official Documentation: http://technet.microsoft.com/en-us/library/dn425064%28v=sql.10%29.aspx

Installation help: http://www.iis.net/learn/application-frameworks/install-and-configure-php-on-iis/install-the-sql-server-driver-for-php

Final Setup: SQLSRV20.exe

5) SQL Server Native Client:

Use this page to decide what components are needed for your setup.
http://msdn.microsoft.com/en-us/library/cc296170%28SQL.105%29.aspx

Final Setup: sqlncli2k8r2x86.msi

6) Overview: The entire process can be very broadly summarised as:

```
IIS -> FastCGI -> PHP 5.3 -> sqlsrv_() -> SSDPHP 3.0 -> SSNC -> MSSQL
IIS -> FastCGI -> PHP 5.2 -> sqlsrv_() -> SSDPHP 2.0 -> SSNC -> MSSQL
IIS -> FastCGI -> PHP 5.2 -> mssql_() -> No SSDPHP -> SSNC -> MSSQL
```

Testing the final setup:

1) PHP System Information Test: If PHP is properly installed, the following code should execute:

SOURCE CODE:

```
<?php echo phpinfo(); ?>
```

You should get PHP system information which includes information about `msql_()` and `sqlsrv_()` extensions. Only relevant partial output shown below.

OUTPUT:

```
Registered PHP   php, file, data, http, ftp, compress.zlib,
Streams          compress.bzip2, https, ftps, zip, sqlsrv
```

cgi-fcgi

```
          Directive              Local Value     Master Value
cgi.check_shebang_line         1               1
cgi.fix_pathinfo               1               1
cgi.force_redirect             0               0
cgi.nph                        0               0
cgi.redirect_status_env        no value        no value
cgi.rfc2616_headers            0               0
fastcgi.impersonate            1               1
fastcgi.logging                0               0
```

msql

```
MSQL Support                                enabled
Allow Persistent Links                      yes
Persistent Links                            0/unlimited
```

Total Links	0/unlimited

mssql

MSSQL Support	enabled
Active Persistent Links	0
Active Links	0
Library version	7.0

Directive	Local Value	Master Value
mssql.allow_persistent	On	On
mssql.batchsize	0	0
mssql.compatability_mode	Off	Off
mssql.connect_timeout	5	5
mssql.datetimeconvert	On	On
mssql.max_links	Unlimited	Unlimited
mssql.max_persistent	Unlimited	Unlimited
mssql.max_procs	Unlimited	Unlimited
mssql.min_error_severity	10	10
mssql.min_message_severity	10	10
mssql.secure_connection	Off	Off
mssql.textlimit	Server default	Server default
mssql.textsize	Server default	Server default
mssql.timeout	60	60

pdo_sqlsrv

pdo_sqlsrv support	enabled

Directive	Local Value	Master Value

pdo_sqlsrv.log_severity	0	0

sqlsrv

sqlsrv support	**enabled**

Directive	Local Value	Master Value
sqlsrv.LogSeverity	0	0
sqlsrv.LogSubsystems	0	0
sqlsrv.WarningsReturnAsErrors	On	On

2) Remote MSSQL Access Test: If `phpinfo()` indicates that all the drivers & extensions are installed properly, then test to see if the database server & client are properly registered.

SOURCE CODE:

```
<?php
$serverName = "dbpcip, 1433"; //serverName\instanceName, portNumber (default
is 1433)
$connectionInfo = array( "Database"=>"master", "UID"=>"userName",
"PWD"=>"passWord");
$conn = sqlsrv_connect( $serverName, $connectionInfo);

if( $conn ) {
     echo "Connection established.
";
}else{
     echo "Connection could not be established.
";
     die( print_r( sqlsrv_errors(), true));
}

if( $client_info = sqlsrv_client_info( $conn)) {
    foreach( $client_info as $key => $value) {
        echo $key.": ".$value."
";
    }
} else {
    echo "Error in retrieving client info.
";
}
```

```
$server_info = sqlsrv_server_info( $conn);
if( $server_info )
{
    foreach( $server_info as $key => $value) {
       echo $key.": ".$value."
";
    }
} else {
      die( print_r( sqlsrv_errors(), true));
}
?>
```

OUTPUT:

Connection established.

Parameter	Value
DriverDllName:	sqlncli10.dll
DriverODBCVer:	03.52
DriverVer:	10.50.1600
ExtensionVer:	2.0.1802.200
SQLServerVersion:	10.50.1600
SQLServerName:	DBPC

3) Test of SQL Query: Once a connection is established, the final test is to see whether query execution is possible or not. This is a quick crude code. But the fact that it retrieves data from `master.dbo.spt_monitor` table shows the setup & the connection works.

SOURCE CODE:

```
<?php
$serverName = "dbpcip, 1433"; //serverName\instanceName, portNumber (default
is 1433)
$connectionInfo = array( "Database"=>"master", "UID"=>"userName",
"PWD"=>"passWord");
$conn = sqlsrv_connect( $serverName, $connectionInfo);

error_reporting(-1);


if( $conn ) {
     echo "Connection established.
";
}else{
     echo "Connection could not be established.
";
     die( print_r( sqlsrv_errors(), true));
}
/* SQL Query */
$sql="select * from dbo.spt_monitor";
$results = sqlsrv_query( $conn, $sql );
if( $results === false) {
    die( print_r( sqlsrv_errors(), true) );
}
    echo "MSSQL master.dbo.spt_monitor TABLE
";
            echo "
            <table border=1>
            <tr>
                               <th>cpu_busy</th>
                <th>io_busy</th>
                <th>idle</th>
                <th>pack_received</th>
                <th>pack_sent</th>
                <th>connections</th>
                <th>pack_errors</th>
                <th>total_read</th>
                               <th>total_write</th>
                               <th>total_errors</th>
             </tr>";
    while ($row = sqlsrv_fetch_array($results))
    {
                               $cpu_busy=$row[1];
                $io_busy=$row[2];
                $idle=$row[3];
                $pack_received=$row[4];
                $pack_sent=$row[5];
                $connections=$row[6];
                $pack_errors=$row[7];
                $total_read=$row[8];
                               $total_write=$row[9];
                               $total_errors=$row[10];

    echo "
            <tr>
                               <td>$cpu_busy</td>;
                <td>$io_busy</td>;
                <td>$idle</td>;
                <td>$pack_received</td>;
                <td>$pack_sent</td>;
                <td>$connections</td>;
                <td>$pack_errors</td>;
                <td>$total_read</td>;
                               <td>$total_write</td>;
                               <td>$total_errors</td>;
            </tr>";
    }
    echo "</table>";
       ?>
```

OUTPUT:

Connection established.

MSSQL master.dbo.spt_monitor TABLE

; ; ; ; ; ; ; ; ; ;

cpu_busy	io_busy	idle	pack_received	pack_sent	connections	pack_errors	total_read	total_write	total_errors
9	7	792	28	28	14	0	0	0	0

4) Incorrect SSDPHP version:

Problem: If you use php53/SSDPHP 3.0 code but use SSNC2008 instead of SSNC 2012, you get an error message.

```
Could not connect. Array ( [0] => Array ( [0] => IMSSP [SQLSTATE] => IMSSP
[1] => -49 [code] => -49 [2] => This extension requires the Microsoft SQL
Server 2012 Native Client.
Access the following URL to download the Microsoft SQL Server 2012 Native
Client ODBC driver for x86: http://go.microsoft.com/fwlink/?LinkId=163712
[message] => This extension requires the Microsoft SQL Server 2012 Native
Client.
Access the following URL to download the Microsoft SQL Server 2012 Native
Client ODBC driver for x86: http://go.microsoft.com/fwlink/?LinkId=163712 )
[1] => Array ( [0] => IM002 [SQLSTATE] => IM002 [1] => 0 [code] => 0 [2] =>
 [Microsoft][ODBC Driver Manager] Data source name not found and no default
driver specified
 [message] => [Microsoft][ODBC Driver Manager] Data source name not found and
no default driver specified ) )
```

Solution: Install php52/SSDPHP 2.0 & SSNC2008

Summary:

Although this project takes a lot of time to setup, troubleshoot & tweak all the settings, the end result is a client/server setup one can experiment with. Once the database server is setup properly, various types of software can be configured to retrieve/store data. PHP can be used to create web applications for doing data analysis using MSSQL.

To access data from MSSQL using php52, `mssql_()` functions is used. But php53 onwards uses only `sqldrv_()` functions to access MSSQL. Hence it is better to learn `sqldrv_()` functions. However knowing how to use the older functions will help when dealing with legacy systems having php52.