BIBLEPAY – DEPLOYING A NEW POOL v1.1

Written By: Rob Andrew

Created: 9-30-2017

1. Navigate to [www.vultr.com](http://www.vultr.com). Rent a new “WINDOWS 2012 R2 X64” OS Vulture Cloud Compute (VC2) Virtual Server with Static IP with the following specs: 100 GB SSD, 4 CPU, 8192mb Memory. ($40+$16 per month=$56 per month). It is also highly recommended to add vultr’s anti-ddos service, as cloudflare does not work properly and the pool will be an instant target to hackers. (It is not recommended to go down to the $20 CPU, as 4gig is not enough ram to run the large SQL2016 DB Plus IIS without paging the disk, also you can mine with excess leftorver CPU with the $40 package). The total is $56 per month since Vultr charges $16 a month for windows. (The anti-ddos is additional).
2. After the new server spins up, log in and enable RDP and ensure RDP works by typing “mstsc” into the client (home) machine and RDPing into the server via FQDNS address. Optionally, change your administrator password so you can set up IIS “administrator”, “password”, jobs, rdp, application pools, and access the server through SQL and file shares (as we need to mount a drive for deploys to the server). Note that this step requires a DNS address. Vultr provides a static IP. After buying a domain for your pool, you should point one “A” record to the pool static IP. For example if you own “mydomain.com”, you can create an A record called “pool” and your DNS address will be “pool.mydomain.com” and if you ping it, it will resolve to the pool vultr static IP. Optionally Rob from BiblePay can give you a FQDNS name with a suffix of biblepay.org, just provide him your static IP and a preferred prefix name.
3. Navigate to the Firewall, and open port 40000 and 40001 inbound so that biblepay can accept external connections. Also open port 30000/30001 on TCP for RPC calls. You will optionally need to open ports 137-138 UDP and 139 and 445 TCP in order to Deploy biblepay pool remotely (Highly recommended, as biblepaypool will constantly be changing for the first year).
4. Add firefox to the server, so that you can download zip file IE biblepay source or the biblepay binary.
5. Install biblepayd on the server or biblepay-qt. Note if you install QT, you will want to ensure it restarts when the system restarts. Launch biblepay-qt wallet, and ensure external connections are being accepted by “telnet nodeip 40000” from an external network box.
6. Download Microsoft SQL Server 2016 EXPRESS edition (FREE) –

As of Oct 9 2017, this link allows downloading SQL Server 2016 Express Edition Sp1 (Free): https://www.microsoft.com/en-us/sql-server/sql-server-editions-express

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ↓[**SQL Server 2016**](https://buildnumbers.wordpress.com/sqlserver/#SQL2016) | 13.00.1601.5 | [13.0.4001.0](https://support.microsoft.com/en-us/kb/3182545) |  |  |  | 13.0.4446.0 [(SP1 CU4)](https://support.microsoft.com/en-us/help/4024305) |

1. Install SQL Server 2016 EXPRESS with all defaults. Do ensure you choose SQL Server Authentication, not Windows Authentication.
2. Download Microsoft SQL server SSMS FOR SQL SERVER 2016 - (Query Analyzer + Tools). Install SSMS for SQL server 2016 with ALL options selected. You will need this to maintain the SQL database.
3. From Server Roles, Install IIS 7.0. Deploy ASP.NET 4.5 Framework, ASPX, ASP, for a full fledged Web Server. Drop a sample HTML file in the server by creating a sample web site (such as IIS7.0 | Sites | New Web Site | test | Bindings | Bind to port 80), drop a sample HTML file in its c:\inetpub\wwwroot\test location and verify the web site works.
4. Verify IIS has .NET 4.5 by navigating to IIS 7.0 and creating a sample application pool using ASP.NET Framework 4.5 for the app pool. You can point this app pool to the test web site you installed in the previous step. If .NET 4.5 is missing, you will need to install 4.5 before deploying the biblepay pool.
5. Log in to your DNS provider, and point a fully qualified domain name to your new server. If you would like an address such as “northpool.biblepay.org” or any variant of “biblepay.org”, notify Rob and he will give you a free DNS entry.
6. Map a drive from your local machine permanently to the DNS resolved name from Explorer, so that you can reach the Y: drive for deploys. For example, you can map “northpool.biblepay.org” to Y:, and clicking it should shield the root of the server. Then you can deploy things to y:\inetpub\wwwroot\biblepaypool\bin. NOTE: You will need to open the firewall rules to allow shared folders for 139,445,137,138 – see <https://technet.microsoft.com/en-us/library/cc731402(v=ws.11).aspx>
7. Download the Pool software from <http://github.com/biblepay/biblepaypool>, to your local PC and compile the code with Microsoft Visual Studio 2017.
8. Deploy the code from the local PC \solution\bin\ directory (with the aspx pages) to the pools site binding UNC location over the connection you created when you created the UNC share in step 12.
9. Set up a new web site in IIS 7.0 with the name corresponding to your DNS record. Ensure the host header record is populated. Ensure the App pool is 32 bit (not the default), .net 4.5, has timeout recycle set to 60 minutes, and load user profile = true, and local disk access is set to the admin user, and the app pool user profile is set to the admin user, and test the site. Note that when you change the LoadUserProfile=True, the server will ask for the admin credentials. This is why it is recommended to change the vultr password of Administrator to something you can type in and maintain for: Physical path credentials to the Web Site, Application Pool Credentials, RDP Credentials, UNC Credentials and Administration Credentials (IE, make the Administrator password strong, but make it consistent across all of these areas, otherwise your server will break in random places- during deploys, during access by users, during RDP authentication etc).
10. Upload the database backup to the server by copying biblepaypool.bak to the SQL database server. Restore the database. Test the restore by querying the users table and ensure all stored procs exist.
11. Test the Pool SAN if you want to receive orphan picture uploads.
12. Ensure the biblepayd integration daemon is accepting connections and may speak through RPC port back to pool, ensure height is shown on about page.
13. Ensure biblepayd RPC credentials are set in the biblepay.conf file, and the web.config file config settings are all set.
14. Test the outgoing email settings to ensure reset password links work. Note: Vultr does not provide a web mail relay so you will need the credentials of either your home ISP or your gmail credentials available. It is recommended to type these gmail credentials in to home PC outlook and test them first.
15. Configure the SQL server to accept inbound TCP connections on port 3389. Ensure the DNS resolved address works with inbound TCP connections for SQL queries. Microsoft SQL Server 2016 does not do this automatically, unfortunately and this step is rather convoluted. You will need to click Program Files | Sql Server | Configuration Tools | SQL Server Config manager. Click SQL Server Network Config. TCP. Enable TCP. Right click TCP. Click IP Addresses. Scroll down to ALL Adapters. Change the TCP Port from blank to 1433. (Also, add a firewall rule to the windows firewall to open port 1433 inbound). Test the connection from the home PC by installing SSMS, go into query analyzer, and connect and verify your new SQL Server can allow inbound connections.

For help setting up port 1433: https://blogs.msdn.microsoft.com/walzenbach/2010/04/14/how-to-enable-remote-connections-in-sql-server-2008/

1. Ensure the SQL server can accept inbound connections (default is to not), and inbound remote connections: see https://knowledgebase.apexsql.com/configure-remote-access-connect-remote-sql-server-instance-apexsql-tools/
2. Mask the unused pool options – such as Orphan List, from the Menu, using SQL query analyzer. For example, Select \* from Menu. Delete from Menu where Hierarchy=’Orphan%’ (this command would delete all orphan submenus from the biblepay pool left menu).
3. To compile the program: Install Visual Studio 2017. Load the solution as an ASPX Web Application with multiple solutions inside the solution group (solution 1 is the BiblePayPool, solution 2 is the BiblePayPool Framework (called USGD Framework). Ensure the BiblePay solution contains the C# bitnet client (Bitnet.Client), Microsoft.Csharp, Microsoft.SqlServer.\* libs, System.Data, and USGDFramework.
4. Ensure Bitness of the solution is changed to allow 32 bit builds and Release is set. TargetFramework: .NET 4.0 from the IDE, 4.5 From IIS7.
5. Click Build, and ensure there are no build errors and the DLLs in \bin\ contain: BiblePayPool.dll, USGDFramework.dll, Microsoft.SQLServer.\*, Global2016.dll, Newtonsoft.dll, and Bitnet.Client.
6. To Deploy, copy the directory with the ASPX files to your website root (for example, on the server, if you create c:\inetpub\wwwroot\biblepaypool, then copy the directory with the aspx files and the web.config to the \biblepaypool directory, and ensure a subdirectory exists with \bin. The server will find the web.config and execute the pool code from the \bin folder. The ASPX files are only used for Markup purposes in production.
7. Set up your Web.Config file: Note, the web.config file should be properly guarded. By default, the user cannot read it or change it. Ensure you do not change permissions on the server to allow hackers to read the web config or change the web config, as it contains your database credentials. You will need to set quite a bit of information for the pool to function and any one of these settings will prevent the pool from running if not set properly. Ill start with the highest priority list of mission critical keys that will break the pool if not set properly: DatabaseHost,DatabaseUser,DatabaseName,DatabasePassword. Without these values, the pool will not load.
8. SAN and WebSite: These settings are optional- and only used if you support orphan picture uploads. The SAN is the UNC path of your SAN drive and the WebSite is the HTTP URL Prefix for picture downloads.
9. MailHost,FromEmail,SMTPPassword: These settings allow the user to receive e-mail notifications allowing them to change their pool password. So far the code is only tested with Gmail as a mail sender.
10. LogPath: A full UNC to the local disk where you want logging to occur. This was necessary because certain calls into the pool are received without an HTTP context, therefore the server has no idea where the log is, so this allows us to consistently log into the same file.
11. PoolReceiveAddress\_main: The public address of the biblepayd BiblePay receiving address for pool funds for blocks solved by the pool. This is extremely important to be correct, as the Pay to Recipients process audits each block to ensure recipients are paid when the pool is paid, and in some rare cicrumstances will be required to run reports to detect overpayments (if everyone was accidentally paid on a block on a fork for example).
12. PoolReceiveAddress\_test: Same as #32 except for TestNet.
13. Admin\_User: This is the name of the User Account that should have God rights to the pool. For example if you have a pool user named “God”, and Admin\_User key value is set to God, when God logs in (through the POOL UI), his administration menu will be populated with options. The Admin can pay Orphan Letter writing bounties, send orphan letters to compassion, scan biblepayblocks to ensure all recipients have been paid, etc.
14. OperatedBy and OwnerEmail: These keys are for the About page, so users can send emails with issues to the site owner of the pool.
15. RPCURLmain: This is a URL that points to a running RPC server on a live biblepayd for the PROD network. The reason this is Mission Critical, is the pool needs the block height to know if blocks are stale, and also needs to check solutions against biblepays ‘exec biblehash input1 2 3 4 5’ function to verify the hash solution. If this URL is down, the entire server will go down. Note there are a few requirements to running a biblepayd RPC instance: The rpcuser,rpcpassword,rpcport and server=1 and daemon=1 need to be set in the biblepayd instance running the server. The rpcallowip= must be set to whitelist the pool servers IP. The service needs to be tested from the command line of the server, example: telnet node.biblepay.org 30000, ensure the telnet answers. NOTE: Telnet needs to be installed on the server: Roles | Add TCP tools | Telnet Client. The fully qualified URL for c# bitnet URLs must be constructed to conform to the server parse. Full example for pool.biblepay.org for prod running on port 39000 (Yes, the RPC Port is different than the Network P2P port). Example: “<http://node.biblepay.org:39000>” means that your RPC server is running from node.biblepay.org on port 39000. In this case you would set rpcport=39000 in the biblepayd, and enter the URL as above in the pool web.config. The RPC server is where the hot wallet is running that will pay pool participants when they withdraw funds.
16. RPCURLtest: Same as #36, except for testnet.
17. RPCPassmain: The password set in the biblepayd rpc server.
18. RPCPasstest: The password for the TestNet biblepay RPC server.
19. Fee: This is a fee converted to a double and multiplied \* 100, so ensure this is set correctly. This fee is the fee raked from all blocks that goes back to the pool administrator. So, if you want to charge a 1% fee to run the pool set it like this: “fee” value=”.01”.
20. Fee\_Anonymous: This is the fee charged to users with accounts set as “cloaked”. These users are anonymous on all reports. If you want to set a 2% fee for anonymous users, set the following: “fee\_anonymous” value=”.02”.
21. Fee\_Letterwriting: This is the fee charged to users who do not write letters to orphans. If a user writes an upvoted letter, the user does not need to write another letter for 60 days until this fee is assessed. If the user has not written an upvoted letter this fee is assessed. To assess a 5% letter writing fee, set the following: “fee\_letterwriting” value=”.05”.

Maintenance Requirements:

It will be necessary to understand how to execute ad-hoc sql queries in Microsoft Query Analyzer. Some database maintenance is required. A list of stored procs will be provided that allow the pool owner to archive old records and perform database re-indexing and remove fragmentation.

Also, it is sometimes necessary to check a users balance manually, update transaction history records, and make table adjustments. This is because if the network goes down between a transaction execution and a partially paid block, it may be possible that all of the records did not insert in either transaction history or, possibly that a user balance was not updated. Or, if a withdraw fails while the RPC is transferring the funds. Therefore it is necessary to be knowledgeable enough in SQL to query the appropriate tables, create a resolution and manually update the values and or insert the proper records.

It is also necessary to understand how to restart Microsoft IIS application pool, redeploy ASP.NET solutions, pull github changes, recompile ASP.NET solutions, restart IIS7 web servers, and perform the standard windows 2012 R2 server administration.

It is necessary to be vigilant daily against hackers, keeping all ports closed except the pool port, keeping the passwords private and long, keeping tabs on the total owed by the pool to the users, vs the total balance in the pool and writing SQL reports to be able to reconcile the pool owed against the current balance. Otherwise, the hackers may gain a quick advantage and take down the pool and all users will lose all balances, we do not want that to happen.