Installing HBase on Windows using Cygwin

Table of contents

1. Introduction

HBase is a distributed, column-oriented store, modeled after Google's BigTable. HBase is built on top of Hadoop for its MapReduce and distributed file system implementation. All these projects are open-source and part of the Apache Software Foundation.

As being distributed, large scale platforms, the Hadoop and HBase projects mainly focus on ***nix* environments** for production installations. However, being developed in **Java**, both projects are fully **portable** across platforms and, hence, also to the **Windows operating system**. For ease of development the projects rely on Cygwin to have a *nix-like environment on Windows to run the shell scripts.

2. Purpose

This document explains the **intricacies of running HBase on Windows using Cygwin** as an all-in-one single-node installation for testing and development. The HBase Overview and QuickStart guides on the other hand go a long way in explaning how to setup HBase in more complex deployment scenario's.

3. Installation

For running HBase on Windows, 3 technologies are required: **Java, Cygwin and SSH**. The following paragraphs detail the installation of each of the aforementioned technologies.

3.1. Java

HBase depends on the Java Platform, Standard Edition, 6 Release. So the target system has to be provided with at least the Java Runtime Environment (JRE); however if the system will also be used for development, the Jave Development Kit (JDK) is preferred. You can download the latest versions for both from Sun's download page. Installation is a simple GUI wizard that guides you through the process.

3.2. Cygwin

Cygwin is probably the oddest technology in this solution stack. It provides a dynamic link library that emulates most of a *nix environment on Windows. On top of that a whole bunch of the most common *nix tools are supplied. Combined, the DLL with the tools form a very *nix-alike environment on Windows.

For installation, Cygwin provides the **`setup.exe` utility** that tracks the versions of all installed components on the target system and provides the mechanism for **installing** or

updating everything from the mirror sites of Cygwin.

To support installation, the `setup.exe` utility uses 2 directories on the target system. The **Root** directory for Cygwin (defaults to `C:\cygwin`) which will become `/` within the eventual Cygwin installation; and the **Local Package** directory (e.g. `C:\cygsetup` that is the cache where `setup.exe` stores the packages before they are installed. The cache must not be the same folder as the Cygwin root.

Perform following steps to install Cygwin, which are elaboratly detailed in the 2nd chapter of the Cygwin User's Guide:

1. Make sure you have `Administrator` privileges on the target system.
2. Choose and create you **Root** and **Local Package** directories. A good suggestion is to use `C:\cygwin\root` and `C:\cygwin\setup` folders.
3. Download the `setup.exe` utility and save it to the **Local Package** directory.
4. Run the `setup.exe` utility,
    1. Choose the `Install from Internet` option,
    2. Choose your **Root** and **Local Package** folders
    3. and select an appropriate mirror.
    4. Don't select any additional packages yet, as we only want to install Cygwin for now.
    5. Wait for download and install
    6. Finish the installation
5. Optionally, you can now also add a shortcut to your Start menu pointing to the `setup.exe` utility in the **Local Package** folder.
6. Add `CYGWIN_HOME` system-wide environment variable that points to your **Root** directory.
7. Add `%CYGWIN_HOME%\bin` to the end of your `PATH` environment variable.
8. Reboot the sytem after making changes to the environment variables otherwise the OS will not be able to find the Cygwin utilities.
9. Test your installation by running your freshly created shortcuts or the `Cygwin.bat` command in the **Root** folder. You should end up in a terminal window that is running a Bash shell. Test the shell by issuing following commands:
    1. `cd /` should take you to thr **Root** directory in Cygwin;
    2. the `LS` commands that should list all files and folders in the current directory.
    3. Use the `exit` command to end the terminal.
10. When needed, to **uninstall** Cygwin you can simply delete the **Root** and **Local Package** directory, and the **shortcuts** that were created during installation.

3.3. SSH

HBase (and Hadoop) rely on **SSH** for interprocess/-node **communication** and launching **remote commands**. SSH will be provisioned on the target system via Cygwin, which

supports running Cygwin programs as **Windows services**!

1. Rerun the **`setup.exe` utility**.
2. Leave all parameters as is, skipping through the wizard using the `Next` button until the `Select Packages` panel is shown.
3. Maximize the window and click the `View` button to toggle to the list view, which is ordered alfabetically on `Package`, making it easier to find the packages we'll need.
4. Select the following packages by clicking the status word (normally `Skip`) so it's marked for installation. Use the `Next` button to download and install the packages.
    1. OpenSSH
    2. tcp_wrappers
    3. diffutils
    4. zlib
5. Wait for the install to complete and finish the installation.

3.4. HBase

Download the **latest release** of HBase from the website. As the HBase distributable is just a zipped archive, installation is as simple as unpacking the archive so it ends up in its final **installation** directory. Notice that HBase has to be installed in Cygwin and a good directory suggestion is to use `/usr/local/` (or [**`Root directory`**]`\usr\local` in Windows slang). You should end up with a `/usr/local/hbase-<`*`version`*`>` installation in Cygwin.

This finishes installation. We go on with the configuration.

4. Configuration

There are 3 parts left to configure: **Java, SSH and HBase** itself. Following paragraphs explain eacht topic in detail.

4.1. Java

One important thing to remember in shell scripting in general (i.e. *nix and Windows) is that managing, manipulating and assembling path names that contains spaces can be very hard, due to the need to escape and quote those characters and strings. So we try to stay away from spaces in path names. *nix environments can help us out here very easily by using **symbolic links**.

1. Create a link in `/usr/local` to the Java home directory by using the following command and substituting the name of your chosen Java environment: LN -s /cygdrive/c/Program\ Files/Java/<*jre name*> /usr/local/<*jre name*>
2. Test your java installation by changing directories to your Java folder `CD`

`/usr/local/<jre name>` and issueing the command `./bin/java -version`. This should output your version of the chosen JRE.

4.2. SSH

Configuring **SSH** is quite elaborate, but primarily a question of launching it by default as a **Windows service**.

1. On Windows Vista and above make sure you run the Cygwin shell with **elevated privileges**, by right-clicking on the shortcut an using `Run as Administrator`.
2. First of all, we have to make sure the **rights on some crucial files** are correct. Use the commands underneath. You can verify all rights by using the `LS -L` command on the different files. Also, notice the auto-completion feature in the shell using `<TAB>` is extremely handy in these situations.
    1. `chmod +r /etc/passwd` to make the passwords file readable for all
    2. `chmod u+w /etc/passwd` to make the passwords file writable for the owner
    3. `chmod +r /etc/group` to make the groups file readable for all
    1. `chmod u+w /etc/group` to make the groups file writable for the owner
    1. `chmod 755 /var` to make the var folder writable to owner and readable and executable to all
3. Edit the **/etc/hosts.allow** file using your favorite editor (why not VI in the shell!) and make sure the following two lines are in there before the `PARANOID` line:
    1. `ALL : localhost 127.0.0.1/32 : allow`
    2. `ALL : [::1]/128 : allow`
4. Next we have to **configure SSH** by using the script `ssh-host-config`
    1. If this script asks to overwrite an existing `/etc/ssh_config`, answer `yes`.
    2. If this script asks to overwrite an existing `/etc/sshd_config`, answer `yes`.
    3. If this script asks to use privilege separation, answer `yes`.
    4. If this script asks to install `sshd` as a service, answer `yes`. Make sure you started your shell as Adminstrator!
    5. If this script asks for the CYGWIN value, just `<enter>` as the default is `ntsec`.
    6. If this script asks to create the `sshd` account, answer `yes`.
    7. If this script asks to use a different user name as service account, answer `no` as the default will suffice.
    8. If this script asks to create the `cyg_server` account, answer `yes`. Enter a password for the account.
5. **Start the SSH service** using `net start sshd` or `cygrunsrv --start sshd`. Notice that `cygrunsrv` is the utility that make the process run as a Windows service. Confirm that you see a message stating that `the CYGWIN sshd service was started succesfully.`
6. Harmonize Windows and Cygwin **user account** by using the commands:

    1. `mkpasswd -cl > /etc/passwd`
    2. `mkgroup --local > /etc/group`
7. **Test** the installation of SSH:
    1. Open a new Cygwin terminal
    2. Use the command `whoami` to verify your userID
    3. Issue an `ssh localhost` to connect to the system itself
        1. Answer `yes` when presented with the server's fingerprint
        2. Issue your password when prompted
        3. test a few commands in the remote session
        4. The `exit` command should take you back to your first shell in Cygwin
    4. `Exit` should terminate the Cygwin shell.

4.3. HBase

If all previous configurations are working properly, we just need some tinkering at the **HBase config** files to properly resolve on Windows/Cygwin. All files and paths referenced here start from the HBase `[`**`installation`** `directory]` as working directory.

1. HBase uses the `./conf/`**`hbase-env.sh`** to configure its dependencies on the runtime environment. Copy and uncomment following lines just underneath their original, change them to fit your environemnt. They should read something like:
    1. `export JAVA_HOME=/usr/local/<`*`jre name`*`>`
    2. `export HBASE_IDENT_STRING=$HOSTNAME` as this most likely does not inlcude spaces.
2. HBase uses the ./conf/**`hbase-default.xml`** file for configuration. Some properties do not resolve to existing directories because the JVM runs on Windows. This is the major issue to keep in mind when working with Cygwin: within the shell all paths are *nix-alike, hence relative to the root `/`. However, every parameter that is to be consumed within the windows processes themself, need to be Windows settings, hence `C:\`-alike. Change following propeties in the configuration file, adjusting paths where necessary to conform with your own installation:
    1. `hbase.rootdir` must read e.g. `file:///C:/cygwin/root/tmp/hbase/data`
    2. `hbase.tmp.dir` must read `C:/cygwin/root/tmp/hbase/tmp`
    3. `hbase.zookeeper.quorum` must read `127.0.0.1` because for some reason `localhost` doesn't seem to resolve properly on Cygwin.
3. Make sure the configured `hbase.rootdir` and `hbase.tmp.dir` **directories exist** and have the proper **rights** set up e.g. by issuing a `chmod 777` on them.

5. Testing

This should conclude the installation and configuration of HBase on Windows using Cygwin. So it's time **to test it**.

1. Start a Cygwin **terminal**, if you haven't already.
2. Change directory to HBase **installation** using `CD /usr/local/hbase-<version>`, preferably using auto-completion.
3. **Start HBase** using the command `./bin/start-hbase.sh`
    1. When prompted to accept the SSH fingerprint, answer `yes`.
    2. When prompted, provide your password. Maybe multiple times.
    3. When the command completes, the HBase server should have started.
    4. However, to be absolutely certain, check the logs in the `./logs` directory for any exceptions.
4. Next we **start the HBase shell** using the command `./bin/hbase shell`
5. We run some simple **test commands**
    1. Create a simple table using command `create 'test', 'data'`
    2. Verify the table exists using the command `list`
    3. Insert data into the table using e.g. put 'test', 'row1', 'data:1', 'value1' put 'test', 'row2', 'data:2', 'value2' put 'test', 'row3', 'data:3', 'value3'
    4. List all rows in the table using the command `scan 'test'` that should list all the rows previously inserted. Notice how 3 new columns where added without changing the schema!
    5. Finally we get rid of the table by issuing `disable 'test'` followed by `drop 'test'` and verified by `list` which should give an empty listing.
6. **Leave the shell** by `exit`
7. To **stop the HBase server** issue the `./bin/stop-hbase.sh` command. And wait for it to complete!!! Killing the process might corrupt your data on disk.
8. In case of **problems**,
    1. verify the HBase logs in the `./logs` directory.
    2. Try to fix the problem
    3. Get help on the forums or IRC (`#hbase@freenode.net`). People are very active and keen to help out!
    4. Stopr, restart and retest the server.

6. Conclusion

Now your **HBase** server is running, **start coding** and build that next killer app on this particular, but scalable datastore!