ClusteringOverview

Permalink to this page: https://cwiki.apache.org/confluence/x/DiklBg

From Zero to Clustered - an Overview

This overview document is based on a posting to the mailing list concerning clustering. This document gives a general overview of the issues and steps that are needed in order to set up a small cluster on a single Linux system. Further documents will add detail and expand on this overview.

General Thoughts and the Mailing List

At each stage of the setup, I recommend testing before moving on to the next section. The mailing list is a good source of helpful information for when you have problems, provided that you do the following.

- Ask simple, focused questions
- Provide all of the requested information (inline, since the mailing list strips most attachments)
- Try to format configuration files reasonably
    - Remove comments
    - Be aware of word wrap
    - Don't use rich text / html text

Be aware that people on the mailing list are **volunteers**. Do not expect homework answers, complete web applications, or systems architecture from the list. Do expect to put in at least as much effort on solving the problem as the people helping you are.

Also, there is a lot of misleading, incomplete, and just flat wrong information concerning Tomcat floating around on the Internet. You might get accurate information elsewhere, but from what I've seen this is not very likely. The authoritative source for information is always:

https://tomcat.apache.org/

Linux in General

You'll find that Linux is a different beast than Windows (even Windows 7). In particular file permissions, file ownerships, and SELinux present quite a different security model than the typical Windows installation. It's best to be aware of this from the start.

Purpose

The first question to answer is what is the purpose of this setup?

If you're running a test or production platform then set up a service account and install all Tomcats there. A service account is an unprivileged account used only for configuring and running Tomcat servers. Make this account inaccessible from the network, and give the account its own group.

If you're setting up a development platform, then just create a directory and unpack multiple Tomcat servers in that directory. This will make dealing with permissions easier, which is an important consideration when integrating Tomcat servers with IDEs such as Eclipse or NetBeans.

Apache HTTPD

Rather than building your own Apache HTTPD, I recommend that you get and install the distribution package for Apache HTTPD. This will place files in line with the rest of your system, and it will also have a serviceable default configuration.

On Fedora, the Apache HTTPD packages include:

- apr
- apr-util
- apr-devel
- apr-util-devel
- httpd
- httpd-devel
- httpd-tools

The -devel packages are very important, since you will be building mod_jk from source.

Most package managers will pull in the correct dependencies, however you will have to specify the development packages separately.

Java

There are several versions and several ways to install Java on a Linux platform. While both OpenJDK and Oracle's Java are reported to work well with Tomcat, some people have reported problems when using GJC. Make sure you have a reasonable version of Java installed, and that it's used by default.

Type:

```
java -version
javac -version
```

to make sure you get the version you expect.

While the Tomcat startup scripts will do a good job at finding the desired Java, it's advantageous to set an environment variable to point to the desired JRE.

In a bash shell, just type the following:

```
export JRE_HOME=[where your JRE is installed]
```

This will ensure that Tomcat uses the Java you want it to use. It's also a useful way to try new versions of Java.

Tomcat

As I've noted above if you're setting up a pure development environment it's just best to create a directory in your home directory and unpack a Tomcat there. For now, just start with one.

If you have Java installed correctly, Tomcat comes ready to run out of the box. Just unpack it, cd to the bin directory, and run startup.sh. You should be able to browse to localhost:8080/ and see the Tomcat welcome page.

Now stop the Tomcat server with shutdown.sh and set up the manager application. This involves editing the tomcat-users.xml file found in the conf directory. Instructions for editing that file (for Tomcat 7) can be found here:

https://tomcat.apache.org/tomcat-7.0-doc/manager-howto.html#Configuring_Manager_Application_Access

Once it's running and you can access the manager application, then associate the installation with your IDE (if you're setting up a development environment). Now you can develop, debug, test, and deploy from within your IDE without running into permission issues.

> Unlike Apache HTTPD, most people on the mailing list do NOT recommend that you use the Linux distribution packaged versions of Tomcat. In general people have found that these are much more difficult to work with than just getting a stock Tomcat from https://tomcat.apache.org/

mod_jk build

While other people have had difficulty (search the mailing list) building mod_jk on various platforms, I've never had much trouble. The steps are quite simple, provided you have the appropriate packages installed on your system.

1. Download the source from https://tomcat.apache.org/download-connectors.cgi
2. Unpack it into a directory
3. cd to `[tomcat-connectors-1.xx]/native`
4. Read `BUILDING.txt`

Following `BUILDING.txt`:

1. `./configure --with-apxs=/usr/sbin/apxs`
2. `make`
3. su to root
4. cd back to where you were
5. `make install`

This will put everything in the right place.

mod_jk configuration

Recent versions of mod_jk come with some very nice and well-commented examples. They can be found in `[tomcat-connectors-1.xx]/conf`. Read them, follow them, use them.

The defaults have been chosen to work in most general use cases.

In order to test mod_jk, map all of the examples (not just the *.jsp files).

Restart Apache HTTPD, and then start Apache Tomcat. You should be able to browse to localhost/examples and get the Apache Tomcat examples.

> Do **NOT** proceed with clustering until you successfully connect one Apache HTTPD with one Tomcat, and execute the examples.

Note that if you place all of your mappings in a uriworkermap.properties file, Apache HTTPD will reread this once per minute (by default). This is nice if you tend to add and delete Tomcat applications. It's probably not quite as nice for a production system (but I don't know what the overhead is).

Clustering

There are a lot of pieces to put together in order to get clustering to work. As noted in the introduction, this Wiki article just gives the lay of the land. Future articles will go into more detail.

Linux

You have to ensure that multicast is enabled, and that multicast routing is set up. If you run iptables, you may run into firewall issues, but on the same system probably not.

mod_jk

Each Tomcat gets at least one worker. This worker must be configured to talk to a particular Tomcat on the correct AJP/1.3 port.

There are some other constraints for load balancing workers that should be noted. In workers.properties:

1. Each worker name in a cluster must be unique 2. The worker name must be the same as the jvmRoute attribute set in the Engine element of the target Tomcat 3. The worker name must be in a list of worker names for a loadbalancer

The mapping in uriworkermap.properties should point to the new loadbalancer worker instead of an individual worker.

See the following for documentation on worker name versus jvmRoute attribute:

https://tomcat.apache.org/tomcat-7.0-doc/cluster-howto.html#Cluster_Basics

See the following for mod_jk loadbalancer configuration:

https://tomcat.apache.org/connectors-doc/reference/workers.html

Another Tomcat

While the easiest way to get started with clustering is just to unpack another copy of Tomcat in another directory, it's probably far more useful to use the concept of CATALINA_HOME and CATALINA_BASE. See RUNNING.txt in the Tomcat directory for details.

However another Tomcat is installed, the following needs to be done.

1. Change the SHUTDOWN port to not conflict with the first Tomcat
2. Change the HTTP/1.1 port to not conflict with the first Tomcat (useful for testing)
3. Change the AJP/1.3 port to match that configured in workers.properties
4. Change the jvmRoute attribute of the Engine element to match the correct worker name
5. Configure the manager application (useful for testing)

Tomcat clustering configuration

The basic clustering documentation can be found here:

https://tomcat.apache.org/tomcat-7.0-doc/cluster-howto.html

It's long, involved, and needs to be read carefully. However, as a first pass the following will work.

1. Make sure that each Tomcat has a unique jvmRoute in the Engine element (see above)
2. Make sure that the jvmRoute matches the correct worker name (see above)
3. Make sure that each Tomcat has a unique shutdown port (see above)
4. Make sure that each Tomcat has a unique AJP/1.3 port (see above)
5. Make sure that each AJP/1.3 port matches the correctly named worker in workers.properties (see above)
6. Make sure that each HTTP/1.1 port is unique (nice for manager access)
7. Copy the example configuration from https://tomcat.apache.org/tomcat-7.0-doc/cluster-howto.html to inside the Host element in each server.xml, but omit (for now) the Deployer element. The cluster configuration can be placed inside the Host or Engine element. However, the Farm Deployer element will only work if the cluster configuration is inside the Host element.
8. Start up your Tomcat servers. If everything is set up correctly, cluster notifications will appear in the logs.

It's also useful to have the access log enabled for your Tomcat servers. You can then tell which Tomcat is receiving the request from Apache HTTPD.

This overview should point you in the right direction to get a two node cluster up and running.

Again, start simply.

1. Stock Apache HTTPD installation (and verify)
2. Stock Apache Tomcat installation (and verify)
3. mod_jk installation (and verify)
4. Second Apache Tomcat installation (and verify)
5. Cluster