SolrJetty

{{{#!wiki red/solid ❌😞 **This page is outdated for Jetty 8.x and newer, and you should read Jetty documentation for configuring the application server and setting properties required by Solr.** ❌😞
}}}

Solr with Jetty

Solr runs fine with Jetty, as illustrated by the `solr/example` application. See the instructions in the generic Solr installation page for basic setup info.

Solr 1.4.1 uses Jetty 6.1.3, Solr 3.5.0 uses Jetty 6.1.26. Solr 4.6.0 uses jetty-8.1.10. For non-trivial installations (multi-instances) with Jetty 6, you need to download full Jetty package because it contains additional modules (JettyPlus). You can download and use 6.1.26 even for Solr 1.4.1 it will work without problem.

NOTE: Jetty 6 does not accept URL-encoded unicode code points outside of the basic multilingual plane (> \uFFFF), see ticket JETTY-1151. You may try upgrading to Jetty 7 (which is a part of eclipse project) to fix this.

Running single instance

If you need only single Solr instances you don't need to download full jetty package. Small Jetty version has been already placed into `solr/example` folder.

This section describes how to to make the Solr example run automatically as a service on startup. It assumes that the Solr distribution is unpacked, and has been tested by running *java -jar start.jar* from the example directory.

Init script to run the Solr example

First of all you need init script.

- Jetty 6. Download this jetty.sh startup script. You can also use updated version of script that doesn't produce start-stop-daemon warnings for new version of start-stop-daemon tool. If you don't have start-stop-daemon tool you can set START_STOP_DAEMON=0 in the script.
- Jetty 7(`<!> jetty.sh startup script`]). Download the [http://dev.eclipse.org/svnroot/rt/org.eclipse.jetty/jetty/trunk/jetty-distribution/src/main/resources/bin/jetty.sh, and place it at /etc/init.d/jetty. On RedHat systems, uncomment the chkconfig lines and use chkconfig to add it to the boot sequence.

Then create */etc/default/jetty*, to configure the startup script. Assuming that the full Solr example (including start.jar) was placed in */opt/solr*, the configuration would look something like this:

```
JAVA_HOME=/usr/java/default
JAVA_OPTIONS="-Dsolr.solr.home=/opt/solr/solr $JAVA_OPTIONS"
JETTY_HOME=/opt/solr
JETTY_USER=solr
JETTY_LOGS=/opt/solr/logs
```

All of these settings are important. In particular, not setting JETTY_LOGS would lead jetty to attempt (and fail) to place request logs in */home/solr/logs*. If you already have JAVA_HOME in enviroment variables you can remove JAVA_HOME line. To check use this command:

```
set | grep JAVA_HOME
```

In popular systems administrator prefers to have logs in single place: /var/log. If you want to have more correct setup please create /var/log/solr folder and change JETTY_LOGS.

Log file

⚠️ Only for the Jetty 6. For Jetty 7 see Solr Logging.

The start-up script expects jetty to redirect standard input and output to a log file. Therefore, create */opt/solr/etc/jetty-logging.xml* with the following:

```
<?xml version="1.0"?>
<!DOCTYPE Configure PUBLIC "-//Mort Bay Consulting//DTD Configure//EN" "http://jetty.mortbay.org/configure.dtd">
<!-- =============================================================== -->
<!-- Configure stderr and stdout to a Jetty rollover log file        -->
<!-- this configuration file should be used in combination with      -->
<!-- other configuration files.  e.g.                                -->
<!--    java -jar start.jar etc/jetty-logging.xml etc/jetty.xml      -->
<!-- =============================================================== -->
<Configure id="Server" class="org.mortbay.jetty.Server">

    <New id="ServerLog" class="java.io.PrintStream">
      <Arg>
        <New class="org.mortbay.util.RolloverFileOutputStream">
          <Arg><SystemProperty name="jetty.logs" default="."/>/yyyy_mm_dd.stderrout.log</Arg>
          <Arg type="boolean">false</Arg>
          <Arg type="int">90</Arg>
          <Arg><Call class="java.util.TimeZone" name="getTimeZone"><Arg>GMT</Arg></Call></Arg>
          <Get id="ServerLogName" name="datedFilename"/>
        </New>
      </Arg>
    </New>

    <Call class="org.mortbay.log.Log" name="info"><Arg>Redirecting stderr/stdout to <Ref id="ServerLogName"/><
/Arg></Call>
    <Call class="java.lang.System" name="setErr"><Arg><Ref id="ServerLog"/></Arg></Call>
    <Call class="java.lang.System" name="setOut"><Arg><Ref id="ServerLog"/></Arg></Call>

</Configure>
```

Creating user

Don't forget to create solr user in the system.

```
useradd -d /opt/solr -s /sbin/false solr
chown solr:solr -R /opt/solr
chown solr:solr -R /var/log/solr # if you use alternative folder for logs
```

NOTE: If you don't have start-stop-daemon tool you have to setup /bin/bash instead /sbin/false.

Starting

Run the following command:

```
/etc/init.d/jetty.sh start
```

Don't forget to add init script at run level.

Running single instance with multicore feature

If you need more that one solr because you want to store different data you may be interested in Solr multiple cores feature.

Running multiple instances

In case when you need to have absolutely isolated Solr instances inside one Jetty you need to read this part.

NOTE: First of all we recommend you to use Tomcat as servlet container because it is more trivial way to add solr instance even without rebooting tomcat. See SolrTomcat article.

Use `Running single instance` instruction to prepare initial environment.

Update Jetty

Download full Jetty package because it contains additional modules (JettyPlus). You can download and use 6.1.26 even for Solr 1.4.1 it will work without problem. We used it to write this article.

Preconditions: Assuming that you've copied `solr/example` folder into /opt/solr.

- Unpack Jetty
- Replace all files in /opt/solr/lib using new files from 'jetty/lib'
- Additionally copy `plus`, `naming`, `management` folders from 'jetty/lib' into /opt/solr/lib
- Replace /opt/solr/start.jar using new file from 'jetty'
- Copy all files from 'jetty/etc' into /opt/solr/etc

Patch jetty default configuration

Open */etc/default/jetty* and add the following line:

```
JETTY_ARGS=/opt/solr/etc/jetty-plus.xml
```

Each Solr instance will have own home directory, so you can remove JAVA_OPTIONS we don't need it anymore.

Adding instances

Open /opt/solr/etc/jetty.xml and add the following part:

```
    <New  class="org.mortbay.jetty.webapp.WebAppContext">
        <Arg><Ref id="contexts"/></Arg>
        <Arg><SystemProperty name="jetty.home" default="."/>/webapps/solr.war</Arg>
        <Arg>/solr_app_1</Arg>
        <Set name="ConfigurationClasses"><Ref id="plusConfig"/></Set>
        <Set name="defaultsDescriptor"><SystemProperty name="jetty.home" default="."/>/etc/webdefault.xml</Set>
        <New class="org.mortbay.jetty.plus.naming.EnvEntry">
            <Arg>solr/home</Arg>
            <Arg type="java.lang.String"><SystemProperty name="jetty.home" default="."/>/webapps/solr_app_1
/solr</Arg>
        </New>
    </New>
```

Jetty with enabled JNDI (JettyPlus) supports adding isolated environment variables for application instances. We need it to specify different solr.home variable for Solr instances. Right now we've found only one way how it will work.

- After adding new instance into `jetty.xml` start jetty.
- Wait until instance will be deployed into /opt/solr/work.
- Stop Jetty
- Create home directory for your new Solr instance and copy into it files from `solr/example/solr` or /opt/solr/solr.
- Open new instance folder inside /opt/solr/work/%instance_name% and put into the following file `webapp/WEB-INF/jetty-env.xml` the following content:

```
<?xml version="1.0"?>
<!DOCTYPE Configure PUBLIC "-//Mort Bay Consulting//DTD Configure//EN" "http://jetty.mortbay.org/configure.dtd">

<Configure class="org.mortbay.jetty.webapp.WebAppContext">
  <New class="org.mortbay.jetty.plus.naming.EnvEntry">
    <Arg>solr/home</Arg>
    <Arg type="java.lang.String">/opt/solr_app_1</Arg>
    <Arg type="boolean">true</Arg>
  </New>
</Configure>
```

- Start Jetty

More information about configuring isolated environments you can see at this page: JettyJNDI

Adding instances (hot deploy)

TODO: Using Context Deployer find a way how to solve two problems:

- How to specify `plusConfig` inside context configuration.
- How to specify solr.home for particular context inside context configuration.

Logging

By default Solr producing huge log output. Solr uses JDK log functionality. You can control the amount of logging output in Solr by using the Admin Web interface. Select the LOGGING link. Note that this page only lets you change settings in the running system and is not saved for the next run. To setup default Log level we have to create file with logging properties logging.properties.

```
org.apache.solr.level = WARNING

handlers = java.util.logging.FileHandler
java.util.logging.FileHandler.formatter = java.util.logging.SimpleFormatter
java.util.logging.FileHandler.pattern = /var/log/jetty6/solr-%u.log
```

and add it into Jetty java options (/etc/default/jetty):

```
JAVA_OPTIONS="-Djava.util.logging.config.file=etc/logging.properties $JAVA_OPTIONS"
```

and restart jetty.

NOTE!!! This logging.properties file contains not only new log level but new file for logs. It has been defined because JDK may stop log to standard output after defining -Djava.util.logging.config.file property.

log4j

You can also use log4j recipe to handle logs. But it probably add new log handler instead replacing current. After reconfiguration please recheck result carefully.

Log rotation

Jetty now has built-in support for log rotation. Add something like this to your jetty.xml file inside the `<Configure id="Server" class="org.mortbay.jetty.Server">` tag:

```
    <New id="ServerLog" class="java.io.PrintStream">
        <Arg>
            <New class="org.mortbay.util.RolloverFileOutputStream">
                <Arg><SystemProperty name="jetty.logs" default="./logs/"/>yyyy_mm_dd.jetty.log</Arg>
                <Arg type="boolean">false</Arg>
                <Arg type="int">90</Arg>
                <Arg><Call class="java.util.TimeZone" name="getTimeZone"><Arg>GMT</Arg></Call></Arg>
                <Get id="ServerLogName" name="datedFilename"/>
            </New>
        </Arg>
    </New>
    <Call class="java.lang.System" name="setErr"><Arg><Ref id="ServerLog"/></Arg></Call>
    <Call class="java.lang.System" name="setOut"><Arg><Ref id="ServerLog"/></Arg></Call>
```

Other information

Configuring Solr Home with JNDI (Jetty < 6.0) (Deprecated)

Jetty Plus provides an addEnvEntry for configuring the JNDI property needed to specify your Solr Home directory.

To do this, use an "addWebApplication" that looks something like this...

```
<Call name="addWebApplication">
  <Arg>/solr/*</Arg>
  <Arg>/your/path/to/the/solr.war</Arg>
  <Set name="extractWAR">true</Set>
  <Set name="defaultsDescriptor">org/mortbay/jetty/servlet/webdefault.xml</Set>

  <Call name="addEnvEntry">
    <Arg>solr/home</Arg>
    <Arg type="String">/your/path/to/your/solr/home/dir</Arg>
  </Call>
</Call>
```

If you want to have several instances you can copy-paste such section and change /solr/* to another name

JNDI Caveats Noted By Users

(7/2007 MattKangas) The recipe above didn't work for me with Jetty 6.1.3. Specifying "solr/home" via "`<New class="...EnvEntry">`" sets a GLOBAL value which gets evaluated after the full configuration is read, so the last setting wins.

Fortunately, I've found a solution that works well:

- Specify "`ContextDeployer`" in `jetty.xml`
- For each web app, add a `.xml` file in "`./contexts`"
    - Set `ConfigurationClasses` to activate JNDI. (must be done separately for each webapp)
    - Set `overrideDescriptor` to define an override web.xml file
    - In the `overrideDescriptor` file, set an `<env-entry>` for "`solr/home`"

The "`overrideDescriptor`" settings will be applied AFTER it has been configured by the default descriptor and the WEB-INF/web.xml descriptor.

Alas, you still need the additional "Plus" .jars, and you need to define the "`plusConfig`" and set "`ConfigurationClasses`" appropriately. Without this, the `overrideDescriptor` won't be applied.

I'm glossing over a lot of details, so attached is a tarball with a known-good configuration that runs two Solr instances inside one Jetty container. I'm using Solr 1.2.0 and Jetty 6.1.3 respectively.

DEMO_multiple_webapps_jetty_6.1.3.tgz

(12/2008 KenEllinwood) I'm using Solr-1.3.0 and Jetty 6.1.12. Apparently the name for JNDI must have a leading slash now., eg., "solr/home" becomes "/solr/home" in the jetty config files. I was able to get the example up and running in a stand-alone Jetty without fiddling with the "overrideDescriptor" as described above. I'm using ContextDeployer in jetty.xml with the following context definition. Note the 3 argument syntax for the EnvEntry – this appears to be a new requirement in more recent versions of Jetty. Also, I had to use an absolute path for the data directory in solrconfig.xml... not sure why.

```
<?xml version="1.0"  encoding="ISO-8859-1"?>
<!DOCTYPE Configure PUBLIC "-//Mort Bay Consulting//DTD Configure//EN" "http://jetty.mortbay.org/configure.dtd">

<Configure class="org.mortbay.jetty.webapp.WebAppContext">
  <Set name="contextPath">/solr</Set>
  <Set name="war">/home/ken/search/apache-solr-1.3.0/example/webapps/solr.war</Set>

  <Set name="extractWAR">true</Set>
  <Set name="copyWebDir">false</Set>
  <Set name="defaultsDescriptor"><SystemProperty name="jetty.home" default="."/>/etc/webdefault.xml</Set>

  <Array id="plusConfig" type="java.lang.String">
    <Item>org.mortbay.jetty.webapp.WebInfConfiguration</Item>
    <Item>org.mortbay.jetty.plus.webapp.EnvConfiguration</Item>
    <Item>org.mortbay.jetty.plus.webapp.Configuration</Item>
    <Item>org.mortbay.jetty.webapp.JettyWebXmlConfiguration</Item>
    <Item>org.mortbay.jetty.webapp.TagLibConfiguration</Item>
  </Array>

  <Set name="ConfigurationClasses"><Ref id="plusConfig"/></Set>

  <New class="org.mortbay.jetty.plus.naming.EnvEntry">
    <Arg>/solr/home</Arg>
    <Arg type="java.lang.String">/home/ken/search/apache-solr-1.3.0/example/solr</Arg>
    <Arg type="java.lang.Boolean">true</Arg>
  </New>

</Configure>
```

Long HTTP GET Query URLs

If you're issuing very long HTTP GET queries to Solr, you may need to adjust the headerBufferSize parameter for your connector in jetty.xml. (See http://docs.codehaus.org/display/JETTY/Configuring+Connectors) The default for this parameter is 4K. If your buffer size is too small, the symptom client-side won't be an error message but rather having your HTTP connection closed without an HTTP response.

JETTY 8

Here is one user's instructions for installing with Jetty 8, although it does not cover the Jetty best practices for installation and configuration for connectors and other parts of the application server that may be required to make this configuration work (for example, enabling the jndi module):

1. Download Jetty 8, unzip it, you will find the contexts dir, create new config file solr.xml [solr.home]/contexts/solr.xml

```
<?xml version="1.0"  encoding="ISO-8859-1"?>
<!DOCTYPE Configure PUBLIC "-//Jetty//Configure//EN" "http://www.eclipse.org/jetty/configure.dtd">

<!-- =============================================================
Configure and deploy the test web application in $(jetty.home)/webapps/test

Note. If this file did not exist or used a context path other that /test
then the default configuration of jetty.xml would discover the test
webapplication with a WebAppDeployer.  By specifying a context in this
directory, additional configuration may be specified and hot deployments
detected.
============================================================== -->

<Configure class="org.eclipse.jetty.webapp.WebAppContext">


  <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
  <!-- Required minimal context configuration :                        -->
  <!--  + contextPath                                                  -->
  <!--  + war OR resourceBase                                          -->
  <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
  <Set name="contextPath">/solr</Set>
  <Set name="war"><SystemProperty name="jetty.home" default="."/>/webapps/solr.war</Set>

  <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
  <!-- Optional context configuration                                  -->
  <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
  <Set name="extractWAR">true</Set>
  <Set name="copyWebDir">false</Set>
  <Set name="defaultsDescriptor"><SystemProperty name="jetty.home" default="."/>/etc/webdefault.xml<
/Set>

  <New class="org.eclipse.jetty.plus.jndi.EnvEntry">
    <Arg>/solr/home</Arg>
    <Arg type="java.lang.String"><SystemProperty name="jetty.home" default="."/>/webapps/solr/</Arg>
    <Arg type="java.lang.Boolean">true</Arg>
  </New>

</Configure>
```

2. Copy solr.war to [jetty.home]/webapps. Manually unzip it there, you will have [jetty.home]/webapps/solr. Copy the conf folder (from the example folder that you download solr) to [jetty.home]/webapps/solr. Change all the config that you need in schema.xml and solrconfig.xml

This step is just for me to test, if you point your /solr/home to another folder, and copy conf folder to that folder, for example <Arg type="java.lang.String"> /opt/solr/</Arg>, then copy conf to /opt/solr/conf. Then copy solr.war to webapps folder, don't need to unzip.

3. under [jetty.home], start jetty in command line: java -jar start.jar OPTIONS=plus