

# JIRA ON DEBIAN LINUX REFERENCE GUIDE

A Step-by-Step Guide for  
Installing, Configuring, Testing and Tuning JIRA  
on Application Servers Powered by the  
Debian Linux Operating System

November 2010 Edition

Written by Jim Intriglia  
Jim Intriglia Consulting LLC  
Jim@JimIntrigliaConsulting.com

# JIRA on Debian Linux Admin Reference Guide

## About this Guide

This reference guide provides basic installation and configuration information for administrators and application developers that need to install, configure and manage the Standalone version 4.x of Atlassian's JIRA application on Debian Linux-based platforms. The Guide is intended for use by system administrators and application developers that have a working knowledge of Linux-based server configuration and desktop application configuration and use.

Visit my website periodically at <http://www.JimIntrigliaConsulting.com> for updated versions of this Guide. Please send your comments and suggestions to [Jim@JimIntrigliaConsulting.com](mailto:Jim@JimIntrigliaConsulting.com).

The information contained in this guide has proven to save JIRA administrators and application developers significant time, energy and aggravation when configuring and administering JIRA application servers for the first time. For experienced JIRA administrators, this Guide serves as a single reference point, where you will find all that you need to know to administer JIRA on a Debian-based Linux application server.

The Guide provides step-by-step instructions for administrators who need to install, configure, test and tune JIRA on an application server running a recent release of the Debian Linux operating system. When words alone fail to quickly convey an installation process or procedure, I'll make use of screenshots and illustrations. To communicate more complex operations or topics, I have embedded links to multimedia screencasts, which feature full audio and video presentations (Internet enabled PC is required to view multimedia content).

I've included additional information that I have found of interest to JIRA administrators, in the Guide's Reference Section. A Glossary is provided to define terminology that is common among JIRA administrators and application developers.

## The New JIRA on Linux Administrators Reference Guide

The new JIRA on Linux Admin Reference Guide, available in December 2010, will address installing, configuring, testing and tuning JIRA on several popular distributions of Linux, including Fedora and CentOS. Installation of JIRA on "Cloud" based production servers, such as provided by service provider Slicehost.com and Rackspace.com, is also covered. More information on the new JIRA on Linux Admin Reference Guide can be found at [JimIntrigliaConsulting.com](http://JimIntrigliaConsulting.com).

## Disclaimer

While I make every attempt to verify the accuracy of the information presented in this guide, I cannot guarantee that there are no errors or omissions. Use of the information contained in this guide is solely at your own risk.

## License

This work is licensed under the Creative Commons Attribution 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

## **What is JIRA and who should be using it?**

JIRA is an issue management application that's ideal for organizations that engage application developers in systems and software development. JIRA utilizes a browser interface, so users may access JIRA using Internet Explorer, Firefox or even the Google Chrome web browser application.

## **Why is JIRA such a capable issue management platform?**

JIRA workflows are highly customizable, which enables JIRA application developers to integrate JIRA into an organizations software development workflow. In fact, JIRA's level of customization enables it to be used for applications outside of managing software development issues. This capability enables organizations that use JIRA to reap a greater return on their JIRA investment, as compared to other issue management platforms.

JIRA can be further customized thanks to an design that enables expansion through a plugin architecture. A large library of Atlassian and third-party plugins is available for JIRA, greatly expanding its application and use for a wide variety of issue management applications.

For more information on JIRA, see [Atlassian's JIRA website](#) and check out my "[Introduction to JIRA Issue Management](#)" screencast.

## **What's involved with getting JIRA up and running?**

JIRA can be installed and configured by most system administrators and application developers familiar with installing server-based web applications. Once JIRA is up-and-running, you can make use of it's default issue management workflow, by investing some additional time to configure JIRA for first use.

## **What's involved with developing customized JIRA workflow applications?**

To really tap the full power of JIRA customization, I would suggest adding an application developer familiar with business process or workflow applications to your JIRA development team. The combination of a JIRA system administrator (to take care of the JIRA application server) and a JIRA application developer (to develop customized JIRA workflow applications for end-users) provides a good starting point on which to begin laying a solid foundation for building a scalable JIRA issue management platform.

## **A Story of a Successful JIRA Implementation: From a Proof-of-Concept Project to a Mission-Critical, Enterprise Class Issue Management System**

Back in 2007, when I was first asked if I was interested in contributing to a JIRA proof-of-concept development project, I was intrigued by the idea that JIRA could be used for anything other than managing issues related to system and software development.

The JIRA project was a small proof-of-concept initiative, with three workflow applications and less than 50 users. The goal of the project was to determine if JIRA could be a more efficient and cost-effective alternative to managing project issues, as compared to managing issues on a common spreadsheet shared among managers and staff.

After a few hours familiarizing myself with JIRA and the JIRA proof-of-concept issue management project, it became evident that JIRA would be significantly more effective and efficient as compared to using a spreadsheet management approach.

I also realized that the JIRA project would need to be transformed gradually from a proof-of-concept project platform to an enterprise-wide application delivery platform, capable of support the needs of hundreds of users. The current proof-of-concept platform, along with the associated management, development and system administration practices, would not be capable of scaling to meet future demands and needs of what was becoming a fast-growing user base.

So, what would it take to transform a JIRA proof-of-concept project into an enterprise-wide, mission critical issue management platform that could be utilized by hundreds of users?

## **Transforming a JIRA Proof-of-Concept Project to an Enterprise-Wide System**

The proof-of-concept JIRA platform was becoming very popular with project stakeholders, senior managers and staff alike, thanks to the power and flexibility of JIRA. As with many proof-of-concept projects, JIRA was being developed and supported largely on an ad-hoc basis. While this approach served JIRA stakeholders well in the beginning of the project, the strain of keeping pace with a growing user community began to degrade the performance of JIRA and the project team that was supporting it.

### **Transitioning the JIRA project to system development, administration and support best practices**

It was time to begin transforming the JIRA project from a proof-of-concept initiative to an enterprise wide issue management system, capable of scaling to meet the needs of a growing community of project stakeholders, managers and end-users. Transforming JIRA to an enterprise issue management system would involve migrating the current JIRA project policies and practices to methodologies and best practices associated with enterprise class information systems.

This meant that system administrators would be engaged to manage JIRA application servers. Their daily duties would consist of ensuring JIRA application server uptime, performing JIRA data backups and restores, updating JIRA software, and establishing multiple JIRA application servers to support development, test and training efforts.

Custom JIRA workflow development would be managed by application developers, familiar with [SDLC](#) methodologies and database-driven workflow management system development. Stakeholder and end-user requirements were first translated into [JIRA workflow](#) and custom field/screen specifications. JIRA application sponsors were required to review and approve JIRA application specifications, before the development team would begin the process of updating JIRA's configuration.

Engaging application developers with the support of system administrators enabled the JIRA team to deliver more custom applications, resulting in exponential growth of JIRA application users. After several months of continued growth in demand for more applications, the JIRA project executive sponsor engaged a project manager to manage the requests and oversee the management of what was fast becoming a JIRA enterprise-wide issue management platform.

As the JIRA development team continued to crank out new JIRA custom workflow applications, a security support team was engaged to manage JIRA user account requests. The Help Desk was engaged to provide JIRA end-user support 24x7; technical writers created instructional job aides and reference materials for JIRA users; system administrators ensured JIRA production, development and test regions were maintained and available 24x7. JIRA was even rebranded successfully when it was rolled-out to meet the needs of a nationwide project initiative.

## JIRA in the Enterprise Today

The JIRA project that began as a modest proof-of-concept project in 2007, is now a mission critical, enterprise class system, with over a thousand user accounts. With over a dozen custom JIRA applications developed to date, JIRA has enabled managers to streamline and automate workflows and business processes such as system and software change management, system resource request and provisioning management, and even document and organizational policy change management.

## Conclusion

So what's the moral of the JIRA project to enterprise class issue management system success story? Treat your JIRA project much the same as you would treat any other significant information systems development project. Apply system and software development methodologies-- don't just wing-it and hope for the best.

Engage experienced executive sponsors, project managers, system/software development and support professionals, etc., as you grow your JIRA project from proof-of-concept to a full-blown enterprise issue management application.

Do these things and you greatly improve your chances of developing a solid JIRA issue management platform, capable of supporting thousands of concurrent users running a variety of issue and work request management applications.

Good luck!

*Jim Intriglia*

## Table of Contents

<b>Installing JIRA on a Debian Linux Application Server</b>	<b>7</b>
Configure a Dedicated System User for JIRA	7
Install Oracle Java JDK/JRE	7
Checking JIRA Startup via Review of the Home Directory	15
Confirming JIRA Startup via Review of the Catalina Log File	15
<b>JIRA Application Administration</b>	<b>20</b>
Configuring a JIRA User Environment	20
Configuration and Use of JIRA Issue Linking	21
Defining JIRA Link Associations	21
<b>References</b>	<b>24</b>
<b>Glossary</b>	<b>25</b>
<b>Document Change History</b>	<b>26</b>
<b>Appendix A: Configuring and Troubleshooting PostgreSQL</b>	<b>27</b>
Creating a New PostgreSQL Database for an Existing JIRA Instance	27
PostgreSQL on Debian v5.x Lenny Notes	28
<b>Appendix B: JIRA Troubleshooting</b>	<b>29</b>
Resetting a Forgotten or Corrupted Administrative Password	29
Error Message: JIRA is Locked	30
<b>Appendix D: JIRA XML Backup and Restore Facility</b>	<b>31</b>
<b>Appendix F: Configuring JIRA's Server.xml File</b>	<b>33</b>
<b>Appendix I - The JIRA Home Directory</b>	<b>35</b>
Configuring the Home Directory	35
Restoring Issue Attachments	35
<b>Appendix K - Configuring JIRA to Send Email via Google GMail SMTP</b>	<b>36</b>
<b>Appendix L - Configuring JIRA Entity Engine</b>	<b>38</b>
<b>Appendix N - Disabling Secure Linux</b>	<b>39</b>

## Installing JIRA on a Debian Linux Application Server

### A. Configure a Dedicated System User for JIRA

A dedicated user needs to be created to run JIRA, as JIRA runs as the user it is invoked under and therefore can potentially be abused. Here is an example of how to create a dedicated user to run JIRA in Linux:

```
$ sudo /usr/sbin/useradd --create-home --home-dir /usr/local/jira --shell /bin/bash jira
```

```
sudo chmod 766 /usr/local/jira
```

To verify that jira has been added as a user, enter one of the following commands:

```
cat /etc/group | cut -d":" -f1
```

```
cat /etc/group | grep jira
```

```
Dwl:~# cat /etc/group | grep jira
jira:x:1001:
Dwl:~# _
```

```
cat /etc/passwd | grep jira
```

```
Dwl:~# cat /etc/passwd | grep jira
jira:x:1001:1001::/usr/local/jira:/bin/bash
Dwl:~# _
```

### B. Install Oracle Java JDK/JRE

JIRA 4.x requires Sun Microsystems' JDK 1.5 or higher. Note that installing just the JRE is not enough to support JIRA 4.x. If the right version of the Java Development Kit (JDK) is not installed, proceed as follows:

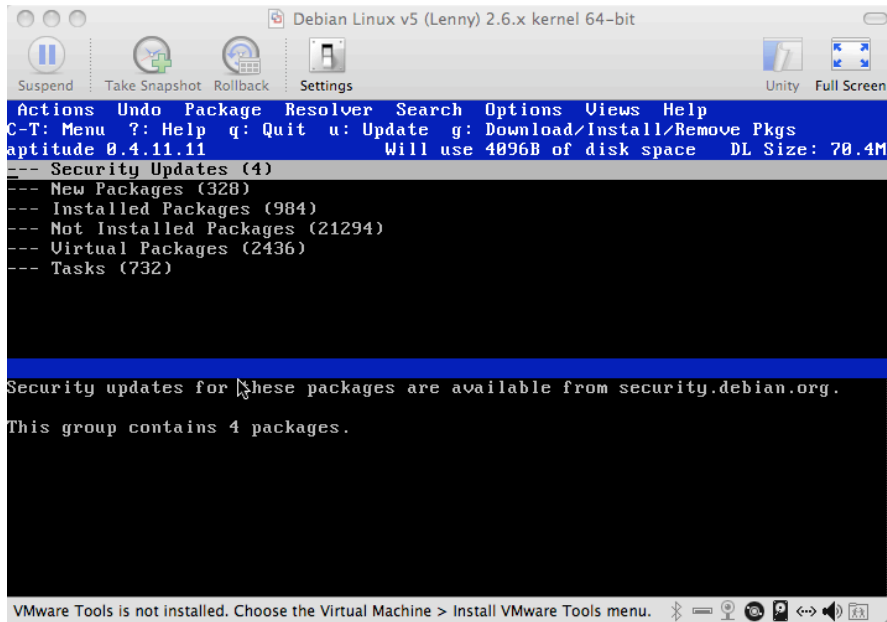
- A. Update Debian's Aptitude `/etc/apt/sources.list` file to enable download of "non-free" packages (the Sun JDK has license restrictions that disqualify it as "free" software in the context of Open Source). Add the following two lines to the `sources.list` file:

```
jim@JIRAv4:~$ sudo cat /etc/apt/sources.list
deb http://http.us.debian.org/debian/ lenny main contrib non-free
deb-src http://http.us.debian.org/debian/ lenny main contrib non-free

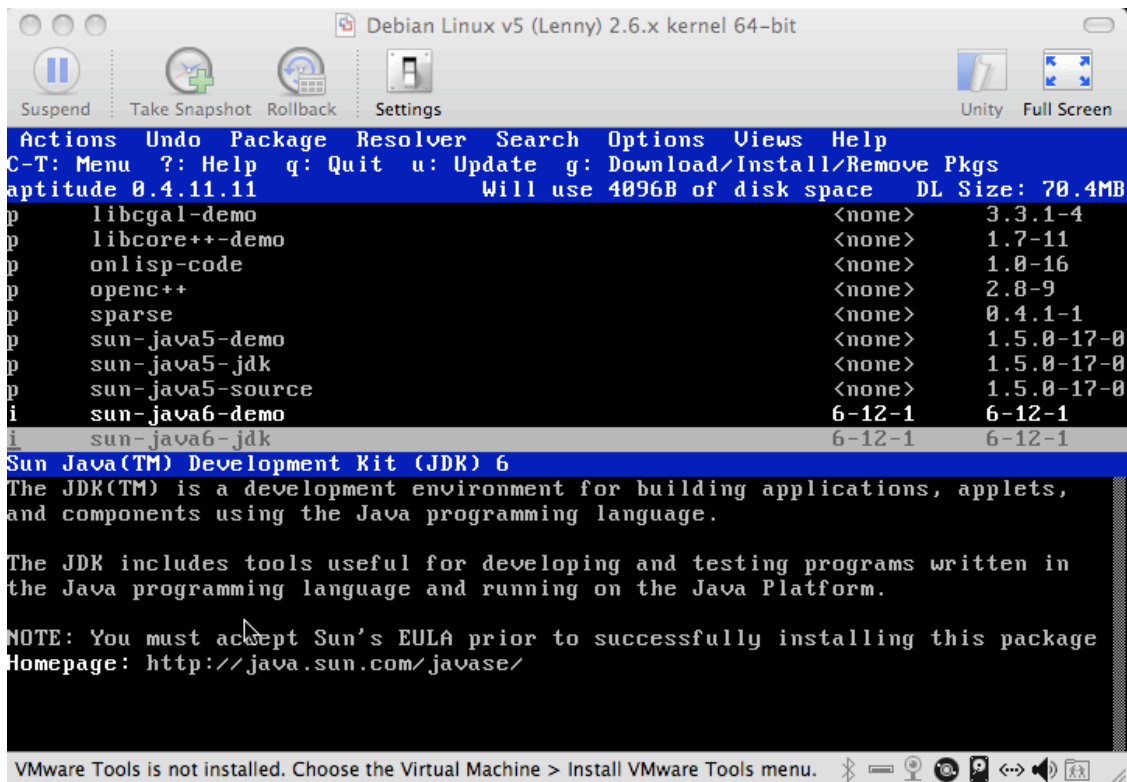
deb http://security.debian.org/ lenny/updates main contrib
deb-src http://security.debian.org/ lenny/updates main contrib

deb http://ftp.debian.org/debian/ lenny main non-free
deb-src http://ftp.debian.org/debian/ lenny main non-free
```

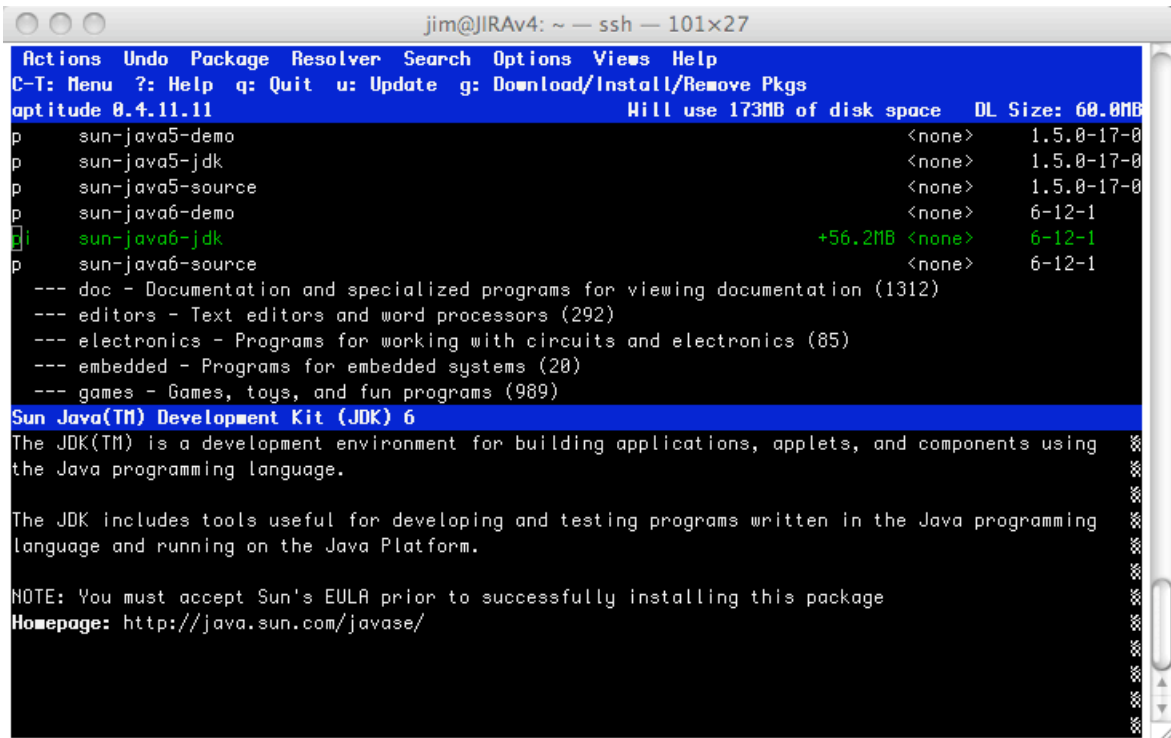
1. From the Linux command line, boot Aptitude:



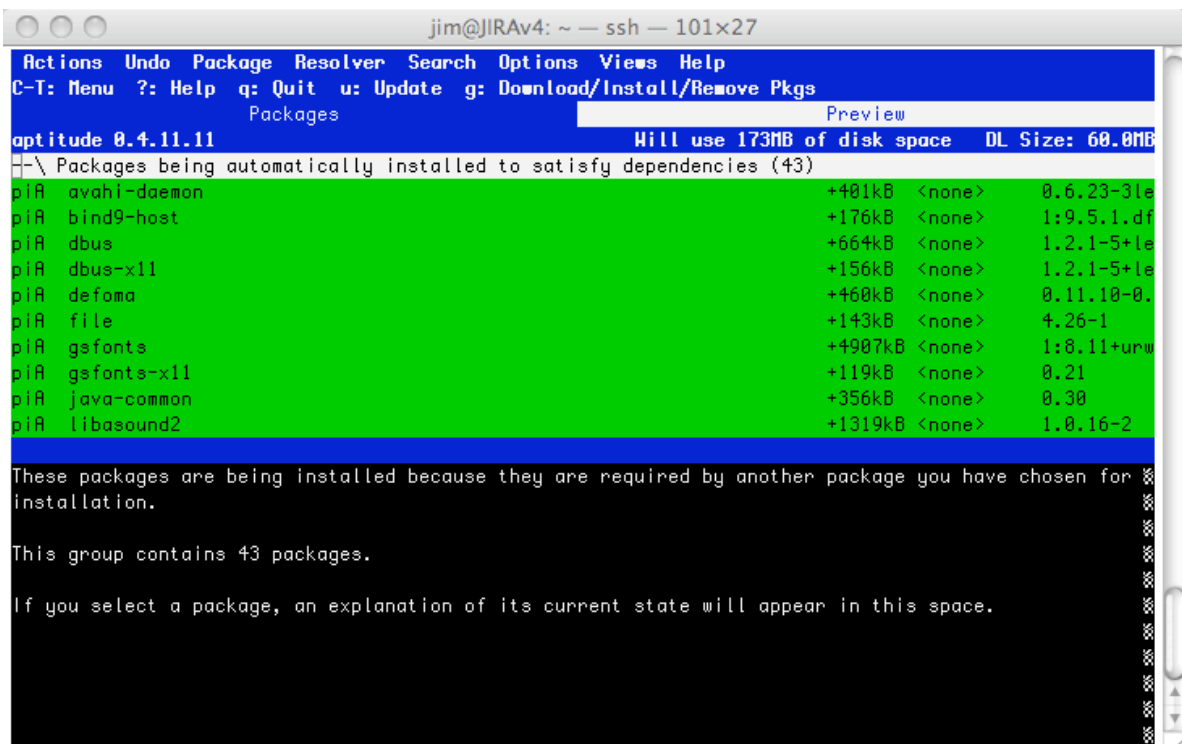
2. Select the “u” menu option to load the non-free Debian application archive libraries references that were added to the “sources.list” file in Step 2.
3. Using Aptitude’s Search feature, locate the “sun-java6-jdk” package.



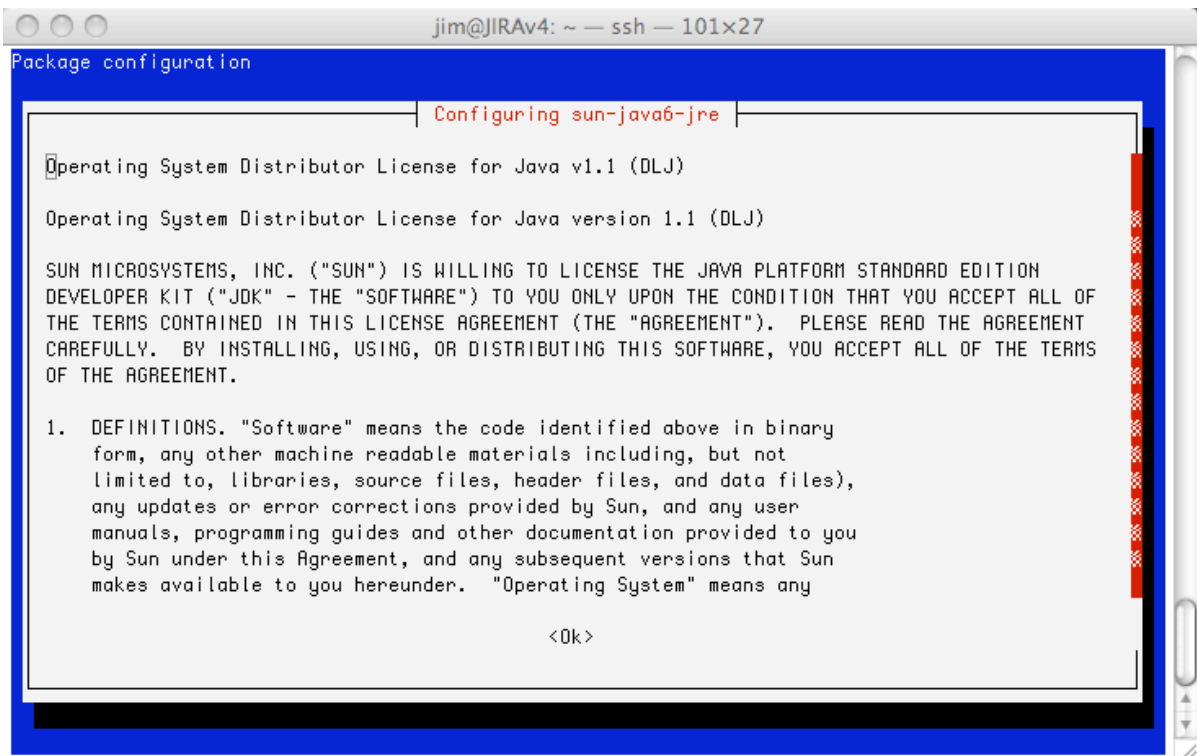
4. Install the sun-jave6-jdk package, by highlighting the package name and pressing the “+” key:



5. Pressing the “g” key now displays all of the packages that the sun-java6-sdk package depends on to function:



- Pressing the “g” key once more downloads and installs sun-java6-sdk. When the package install has been completed, review and accept the Sun Java6 SDK. You will be returned to the Aptitude main menu display when installation is complete.



- After installing Oracle Java JDK, the system will likely still be referencing the default GNU GJC JDK. To ensure JIRA uses the Sun JDK, enter the following command:

```
sudo update-alternatives --config java
```

If there is only a single alternative of Java available on your system, entering the update-alternatives command will yield the following:

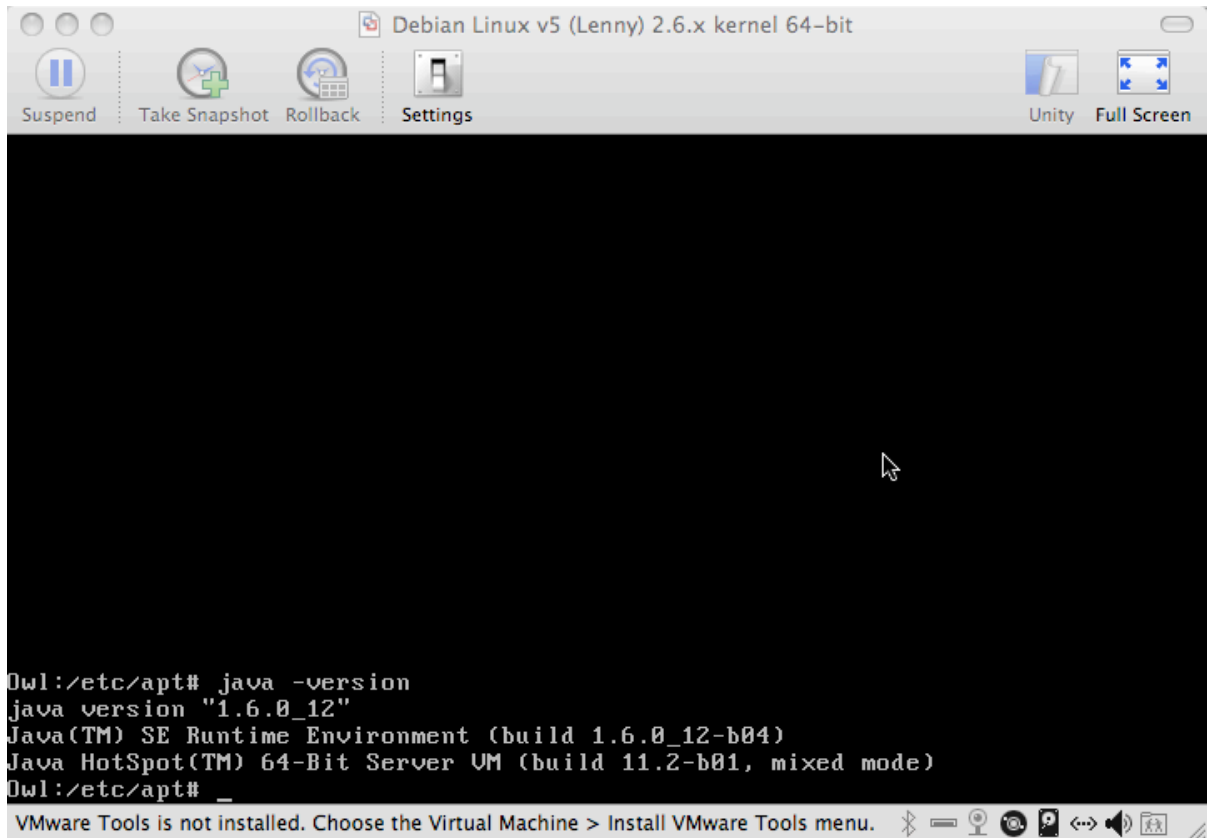
```
jim@JIRAv4: ~$ sudo update-alternatives --config java
[sudo] password for jim:
```

```
There is only 1 program which provides java
(/usr/lib/jvm/java-6-sun/jre/bin/java). Nothing to configure.
jim@JIRAv4: ~$
```

Enter the following command to ensure the Sun JDK is now set as the preferred Java JRE:

```
java -version
```

If Sun Java has been installed and configured correctly, you should see something similar to the following:



The screenshot shows a terminal window titled "Debian Linux v5 (Lenny) 2.6.x kernel 64-bit". The terminal output is as follows:

```
Owl:/etc/apt# java -version
java version "1.6.0_12"
Java(TM) SE Runtime Environment (build 1.6.0_12-b04)
Java HotSpot(TM) 64-Bit Server VM (build 11.2-b01, mixed mode)
Owl:/etc/apt# _
```

At the bottom of the terminal window, there is a message: "VMware Tools is not installed. Choose the Virtual Machine > Install VMware Tools menu." along with system tray icons.

8. Set the `JAVA_HOME` environment variable pointing to the root directory of the JDK:

To ensure that Java is available to JIRA platform components like Tomcat, login as root and append the following to the end of `/etc/profile` file. This will ensure that all users have the `JAVA_HOME` variable set pointing to the Sun Java directory when they are logged-in.

```
JAVA_HOME=/usr/lib/jvm/java-6-sun
export JAVA_HOME
```

Verify that `JAVA_HOME` is set by typing:

```
echo $JAVA_HOME
```

**NOTE:** Entering "java" at the command line prompt will also confirm that the `JAVA_HOME` system variable has been set, as Java help will be displayed in response to the java command.

### III. Download and Unpack Current Version of JIRA

The current release of JIRA v4.x can be downloaded via Atlassian's JIRA release web page:

<http://www.atlassian.com/software/jira/JIRADownloadCenter.jspa>

If you are installing JIRA on a remote Linux-based server, you can use the `wget` command to download the file to the `/opt/jira` directory.

```
mkdir /opt/jira
```

```
cd /opt/jira
```

```
wget http://www.atlassian.com/software/jira/downloads/binary/atlassian-jira-enterprise-4.2-standalone.tar.gz
```

If `wget` is not available on the server, try:

```
curl -C - -L -O http://www.atlassian.com/software/jira/downloads/binary/atlassian-jira-enterprise-4.2-standalone.tar.gz
```

Once the JIRA 4.x .tar.gz file is downloaded, un-tar the package:

```
tar -zxvf atlassian.jira-enterprise-4.0-standalone.tar.gz
```

### IV. Create Dedicated JIRA System User

A dedicated user should be created to run JIRA, as JIRA runs as the user it is invoked under and therefore can potentially be abused. To create a dedicated user to run JIRA in Linux, type the following command:

```
$ sudo /usr/sbin/useradd --create-home --home-dir /usr/local/jira --shell /bin/bash jira
```

### V. Configure the JIRA Home Directory

The JIRA Home directory is where JIRA will place its working directory and associates files (backup files, index files, log files, etc.) when it has been started-up and running.

To set the JIRA Home directory, edit the `jira-application-properties` file in the JIRA Installation Directory. If you unpacked JIRA in the recommended `/opt/jira` subdirectory, the `jira.application.properties` file is located here:

```
/opt/jira/atlassian-jira-enterprise-4.0-standalone/atlassian-jira/WEB-INF/  
classes
```

Open the `jira-application-properties` file with a text editor. Add a `'jira.home'` property:

```
# Each backslash in your path must be written as a forward slash.  
# - For example:  
# c:\jira\data  
#  
# should be written as:  
#  
# c:/jira/data  
jira.home = /usr/local/jira
```

Be sure to use forward-slashes ("/") rather than back-slashes ("\") when specifying the directory location.

## VI. Increase JIRA Memory Allocation

Allocation of JIRA memory is set by configuring JIRA's `setenv.sh` file, located in JIRA's `./atlassian-jira-enterprise-4.x.x-standalone/bin` directory. Note that JIRA needs to be rebooted for changes to this file to become effective.

The JIRA v4.1 release featured a new release of the Tomcat application server (v6.0.2). The new release of Tomcat included a much cleaner layout for the `setenv.sh` file, thus reducing the chance of error when making updates to this file.

### A. Configuring JIRA Memory Usage

#### 1. For JIRA 4.1.x Releases

To increase JIRA's memory allocation to a level that I found provided acceptable performance for a new installation, update the following two lines in the `setenv.sh` file as follows:

```
#  
# The following 2 settings control the minimum and maximum given to the JIRA  
# Java virtual machine. In larger JIRA instances, the maximum amount will need to  
# be increased.  
#  
JVM_MINIMUM_MEMORY="348m"  
JVM_MAXIMUM_MEMORY="512m"
```

## 2. For JIRA 4.0.x Releases

To increase JIRAs default memory allocation, we increase the memory allocation from 128MB to 512MB by editing a configuration setting in JIRA's `/bin/setenv.sh` file:

From:

```
JAVA_OPTS="-Xms128m -Xmx256m $JAVA_OPTS -Djava.awt.headless=true -Datlassian
ntentImpl.LIMIT_BUFFER=true -Dmail.mime.decodeparameters=true "

# Perm Gen size needs to be increased if encountering OutOfMemoryError: Per
on IBM JDKs
JIRA_MAX_PERM_SIZE=256m
```

To:

```
JAVA_OPTS="-Xms384m -Xmx512m $JAVA_OPTS -Djava.awt.headless=true -Datlassian
ntentImpl.LIMIT_BUFFER=true -Dmail.mime.decodeparameters=true "

# Perm Gen size needs to be increased if encountering OutOfMemoryError: Per
on IBM JDKs
JIRA_MAX_PERM_SIZE=512m
```

## VII. Start JIRA: Verify a Successful Startup

As a security precaution, we added a jira user to our Linux server during the first steps of configuring our JIRA server. Remember that to startup JIRA, you must first become the user "jira" before issuing the JIRA startup command:

```
su -
sudo su - jira
```

As the jira user, you can now issue the JIRA startup command:

```
cd /opt/jira/atlassian-jira-enterprise-4.0-standalone/bin
./startup.sh
```

JIRA will display startup messages:

```
Detecting JVM PermGen support...
PermGen switch is supported. Setting to 256m
If you encounter issues starting up JIRA Standalone Edition, please see the Troubleshooting guide
at http://confluence.atlassian.com/display/JIRA/Installation+Troubleshooting+Guide
Using CATALINA_BASE: /opt/jira/atlassian-jira-enterprise-4.0-standalone
Using CATALINA_HOME: /opt/jira/atlassian-jira-enterprise-4.0-standalone
Using CATALINA_TMPDIR: /opt/jira/atlassian-jira-enterprise-4.0-standalone/temp
Using JRE_HOME: /usr/lib/jvm/java-6-sun
Using CLASSPATH: /opt/jira/atlassian-jira-enterprise-4.0-standalone/bin/bootstrap.jar
```

Use the Linux "tail" command to monitor JIRA as it continues to progress through start-up tasks:

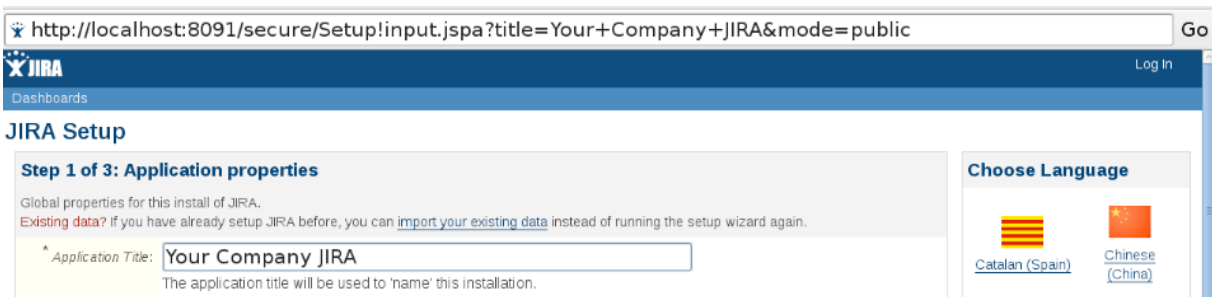
```
tail -F ../logs/catalina.out
```

If JIRA successfully started, you will see the following message near the bottom of the catalina.out file:

```
*****
JIRA 4.0 build: 466 started. You can now access JIRA through your web browser.
*****
```

### A. Confirming JIRA Startup via Web Browser

Via a web browser, navigate to <http://localhost>; verify that JIRA's setup page (partially pictured above) is displayed.



### B. Checking JIRA Startup via Review of the Home Directory

JIRA maintain a Home Directory where it stores application and system-related data. One of the indicators of a successful JIRA startup is the creation of “child” subdirectories under JIRA’s “parent” Home directory:

```
JIRAv4:/usr/local/jira# ls -l
total 16
drwxr-xr-x 3 root root 4096 2009-11-27 08:03 caches
drwxr-xr-x 3 root root 4096 2009-11-27 08:03 data
drwxr-xr-x 2 root root 4096 2009-11-27 08:03 export
drwxr-xr-x 5 root root 4096 2009-11-27 08:03 plugins
JIRAv4:/usr/local/jira#
```

### C. Confirming JIRA Startup via Review of the Catalina Log File

View the “catalina.out” file located in the “/logs” directory for your JIRA installation:

```
JIRAv4:/opt/jira/atlassian-jira-enterprise-4.0-standalone/logs# ls -l
total 104
-rw-r--r-- 1 root root    0 2009-11-27 08:07 access_log.2009-11-27
-rw-r--r-- 1 root root    0 2009-11-27 08:03 admin.2009-11-27.log
-rw-r--r-- 1 root root 1012 2009-11-27 08:07 catalina.2009-11-27.log
-rw-r--r-- 1 root root 97080 2009-11-27 08:07 catalina.out
-rw-r--r-- 1 root root    0 2009-11-27 08:03 host-manager.2009-11-27.log
-rw-r--r-- 1 root root    0 2009-11-27 08:03 localhost.2009-11-27.log
-rw-r--r-- 1 root root    0 2009-11-27 08:03 manager.2009-11-27.log
JIRAv4:/opt/jira/atlassian-jira-enterprise-4.0-standalone/logs#
```

## VIII. Connect JIRA with PostgreSQL -

Now that JIRA is installed and verified operational, the next step involves integrating JIRA with the PostgreSQL database engine. If PostgreSQL is not presently installed, you can install it via aptitude:

```
sudo apt-get install postgresql-8.3 postgresql-client-8.3
```

Add the JIRA user to PostgreSQL:

```
Owl:~$ sudo su - postgres
```

**NOTE:** If the following error message is displayed:

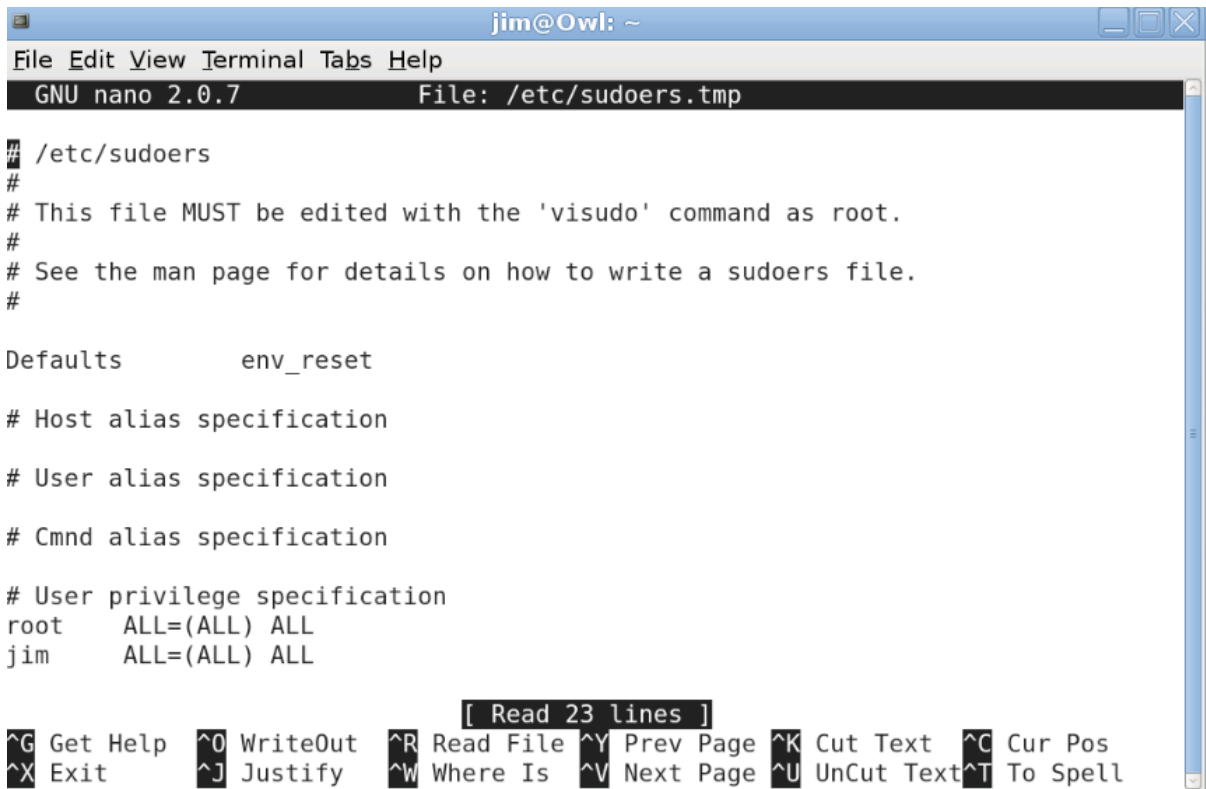
```
jim@Owl:~$ sudo su - postgres

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for jim:
jim is not in the sudoers file.  This incident will be reported.
```

You will need to add your username to the `/etc/sudoers` file via the command “visudo”. Once in the editor, add your user name to the file:



```

jim@Owl: ~
File Edit View Terminal Tabs Help
GNU nano 2.0.7 File: /etc/sudoers.tmp

# /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for details on how to write a sudoers file.
#

Defaults            env_reset

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL) ALL
jim     ALL=(ALL) ALL

[ Read 23 lines ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page  ^U UnCut Text ^T To Spell

```

When you have added your username after the entry for the “root” user, type <ctrl O> <ctrl X> to save the file. The command “sudo su - postgres” should now enable you to boot the postgres client.

Once you have booted the pgsq shell, enter the command:

```
createuser -P jira
```

```

Owl:/etc# sudo su - postgres
postgres@Owl:~$ createuser -P jira
Enter password for new role:
Enter it again:
Shall the new role be a superuser? (y/n) n
Shall the new role be allowed to create databases? (y/n) y
Shall the new role be allowed to create more new roles? (y/n) n
postgres@Owl:~$ █

```

Enter a password for the PostgreSQL jira user, followed by the responses to each of the queries listed above.

Next, login as the jira user, and issue the PostgreSQL command to create the JIRA v4.00 database:

```
postgres@Owl:~$ logout
Owl:/etc# sudo su - jira
jira@Owl:~$ createdb jira_400
jira@Owl:~$
```

To confirm that the JIRA database has been created and the user “jira” can connect to the jira\_400 database, enter the following command:

```
Owl:/etc/postgresql/8.3/main# psql -h 127.0.0.1 -U jira jira_400
Password for user jira:
Welcome to psql 8.3.8, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help with psql commands
      \g or terminate with semicolon to execute query
      \q to quit

SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)

jira_400=>
```

You can enter the following command to test your PostgreSQL connection to the jira database:

```
SELECT datname FROM pg_database;
```

```
jira_400=> select datname from pg_database;
 datname
-----
 templat1
 template0
 postgres
 jira_400
(4 rows)
```

Exit from psql by typing <ctrl-d>. We will now download the PostgreSQL JDBC database driver that will enable an interface between JIRA and PostreSql.

## IX. Install the PostgreSQL Database Driver

- A. Download the PostgreSQL database interface driver from the PostgreSQL website: <http://jdbc.postgresql.org/download.html>:



A current installation of JIRA v4.x and PostgreSQL v8.3 requires the JDBC4 PostgreSQL driver.

B. Copy the PostgreSQL Driver to JIRA's common/lib directory.

```
Owl:/opt/jira/atlassian-jira-enterprise-4.0-standalone/common/lib# ls -l | grep post
-rwxr-xr-x 1 root root 510170 2009-10-28 00:42 postgresql-8.4-701.jdbc4.jar
```

**IMPORTANT:** Change the ownership and permission of the postgresql-8.4-701.jdbc4.jar file to jira:jira from root:root. Failure to do this will result in a “no suitable driver” error found in JIRA’s Catalina,out file.

```
chown jira:jira postgresql-8.4-701.jdbc4.jar
```

Step 12. Edit JIRA’s Server.xml Configuration File - See [Appendix F](#) for detailed information to complete this step.

Step 13. Configure the Entityengine.xml File - In the ./WEB-INF/classes directory, make a copy of the entityengine.xml file:

```
Owl:/opt/jira/atlassian-jira-enterprise-4.0-standalone/atlassian-jira/WEB-INF/classes# cp entityengine.xml entityengine.xml.orig
```

Edit the datasource and scheme name settings as below. Be sure the “field-type-name” is set to “postgres72” as this literal does not reflect the actual drive that has been installed. Also, change “schema-name” to “public” (all lowercase characters):

```
<datasource name="defaultDS" field-type-name="postgres72"
  schema-name="public"
```

Step 14. Start JIRA; Confirm Successful Startup - See [Appendix H](#) for detailed instructions. The goal of this step is to ensure that JIRA can access and login to PostgreSQL, create a new JIRA database and all of the table structures and associated indexes.

Step 15. Shutdown JIRA After Setup and Testing - We are now going to configure JIRA to use Google Mail to send email notifications. Before we do this, we are going to shutdown JIRA by entering the following command:

```
./shutdown.sh
```

Remember to shutdown JIRA before bring the server down. To shutdown JIRA, execute the JIRA shutdown script:

**NOTE:** Ensure you are logged-in as the user that launched JIRA (in this example, the user "jim" is permissioned to start and shutdown JIRA).

Step 16. Configure JIRA Email via Google GMail - See [Appendix K](#) for detailed information to complete this step.

## IX. JIRA Application Administration

### A. Configuring a JIRA User Environment

Once a new JIRA installation is running, you may want to reconfigure some of JIRA's default user environment settings. For example, many users prefer receiving HTML email notifications, rather than the default text-based version. To set HTML format email notifications as the default, access the configuration setting via the Administrator's panel, Global Settings -> User Defaults:

### User Default Settings

Set the default values for user preferences. If the user has not specified a preference then the values for the user will fall back to the default values set here. You can 'Apply' the email preference which will allow you to force each users value to be that of the default.

[Edit default values](#)

Name	Value	Operations
Default outgoing email format	html	<a href="#">Apply</a>
Number of Issues displayed per Issue Navigator page	50	
Notify users of their own changes?	<b>YES</b>	
Default sharing for filters and dashboards	Private	

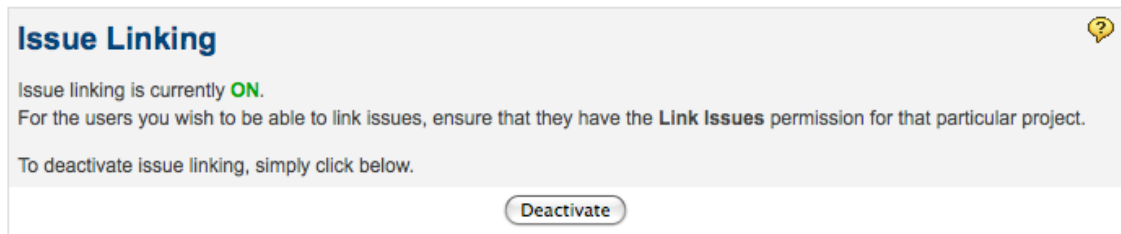
**NOTE:** Users can override the default HTML setting if they so choose, via their own user profile setting.

You can also reconfigure other default user settings, such as the number of issues that are displayed when using the Issue Navigator, as well as whether JIRA will send email notifications to users when they affect their own changes.

## 1. Configuration and Use of JIRA Issue Linking

Another useful JIRA feature is the ability to link JIRA issues. Issue linking enables permissioned users communicate relationships and dependencies that exist between JIRA issues. This can be very helpful to project managers, business analysts and application developers when figuring out the order in which issues should be resolve.

By default, Issue Linking is disabled. JIRA Administrators can enable issue linking by clicking on the “Issue Linking” menu option, Global Settings section of the Admin panel:



Issue Linking is a user privilege that must be explicitly configured for each JIRA project via it's Permission scheme. For each JIRA project, you will need to specify users, user groups or project roles that will be permitted to define links between JIRA issues:

<p><b>Link Issues</b></p> <p>Ability to link issues together and create linked issues. Only useful if issue linking is turned on.</p>	<ul style="list-style-type: none"> <li><input type="checkbox"/> Project Role (Project Managers) <a href="#">(Delete)</a></li> <li><input type="checkbox"/> Project Role (Project Staff) <a href="#">(Delete)</a></li> <li><input type="checkbox"/> Project Role (Administrators) <a href="#">(Delete)</a></li> </ul>	<p><input type="checkbox"/> <a href="#">Add</a></p>
---	--	---

Once Issue Linking is enabled and a user is permissioned to create links between issues, the “Link” option will be visible on the users panel when they are viewing an issue.

## 2. Defining JIRA Link Associations

JIRA Administrators will also need to define Issue Link relationships. Here’s where your JIRA end-users can be very helpful in defining issue linking relationships that will be meaningful and useful to end users.

For example, Here’s a good “generic” issue link relationship that is likely to be meaningful when linking within a specific project or linking to an issue or work request in another JIRA project:

## Add New Link Type

Add a new link type

Name:	<input style="width: 90%;" type="text" value="Associated with"/> <small>(eg "Duplicate")</small>
Outward Link Description:	<input style="width: 90%;" type="text" value="associated with"/> <small>(eg "duplicates")</small>
Inward Link Description:	<input style="width: 90%;" type="text" value="is associated with"/> <small>(eg "is duplicated by")</small>

NOTE: Once a link relationship has been implemented by a JIRA Administrator, users across all JIRA projects may use the link relationship to link issues with a JIRA project or between to separate JIRA projects.

Here's an example of using the "Associated with" link to communicate a relationship between two issues within the same JIRA project:

**JIRA Work Request and Issue Management System**
[Return to search](#)

**JIRA IMS Does Not Auto-Start on Server Reboot**

Created: Today 08:00 AM Updated: Today 08:49 AM

**Component/s:** [System Administration](#)

**Affects Version/s:** JIRA WRIMS 1.0 - JIRA Work & Issue Mgmt

**Fix Version/s:** [JIRA WRIMS v1.1](#)

**Issue Links:**

**Associated with**

This issue *associated with*:

[JIRA-8](#) Slicehost Sample IPTABLES Configuration Blocks JIR

**Description** [Hide](#)

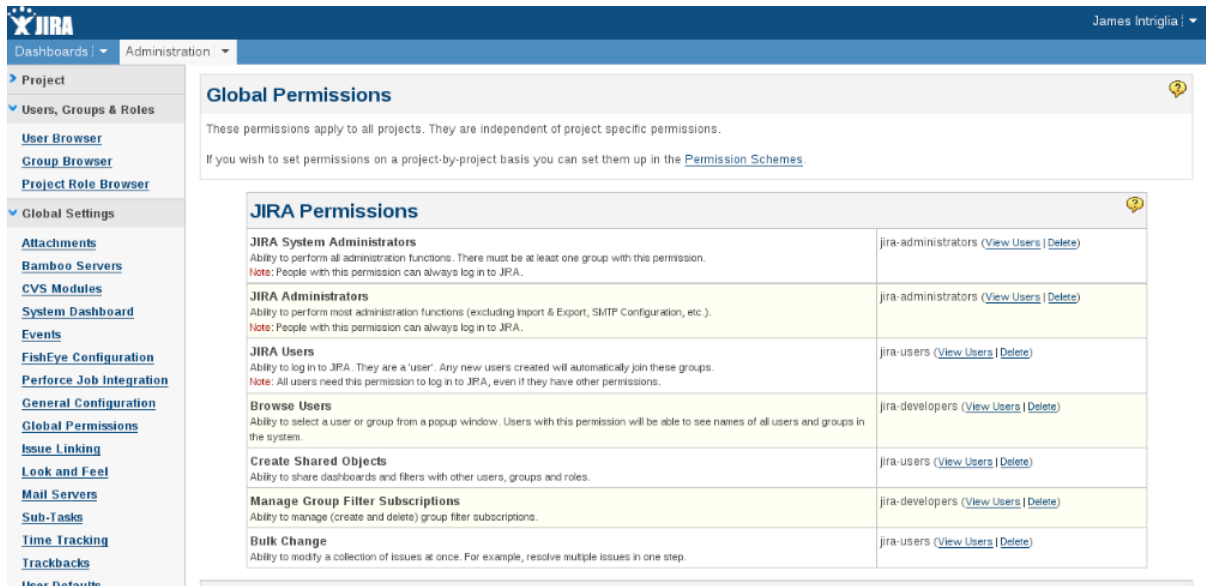
If the Slicehost server is rebooted, JIRA's debian O/S is not configured to restart JIRA. In order to accomplish this, scripting will need to be developed that will:

- 1) Delete the jira-home.lock file in JIRA's Home directory
- 2) Run the ./shutdown.sh script before running the ./startup.sh script
- 3) Grep Cataline.out for the presence of a "JIRA is Running" message.

All **Comments** Change History Activity Stream
Sort Order:

[Jim Intriglia](#) added a comment - 10/Jan/10 08:49 AM [Permalink](#) | [Edit](#) | [Delete](#) | [Hide](#)

As a next step, resolve issue with iptables so that JIRA will not be blocked by firewall when the server is restarted.



Organizational staff that are responsible for associating JIRA users with projects are typically “permissioned” as “JIRA Administrators”. Technical JIRA developers and administrators responsible for developing and support JIRA systems will be permissioned as “JIRA System Administrators”.

A two-tiered system of JIRA administration makes for effective and efficient use of a JIRA administration team’s time, talent and experience, respective of allocating work requests to administrators that are best suited to manage an assigned administrative task.

JIRA’s administrative permissions can be accessed by JIRA administrators via the Administration tab, Global Settings -> Global Permissions.

To put into action the two-tier JIRA Administration global permissions, you can create two JIRA User Administration Groups. One group of administrative users will manage tasks related to JIRA functional administration, the other group of JIRA administrators will manage both functional and technical administrative tasks.

User Group Name	Role Description	JIRA Global Permissions
JIRA Project Administrators	Manage project and user-oriented JIRA administrative tasks, such as adding users to project groups.	JIRA Administrators
JIRA System Administrators	Manage all technical and functional aspects of JIRA, such as reindexing JIRA indexes, executing backups and restores, plus all JIRA Administrator functions.	JIRA System Administrators

## X. References

Anonymous. 2010. Createuser PostgreSQL Man Page. <http://www.rootr.net/man/man/createuser/1>

Charron, Albert ([acharron@trisotech.com](mailto:acharron@trisotech.com)). JIRA 4.x on Linux forum post. <http://forums.atlassian.com/thread.jspa?threadID=38085&tstart=0>

Gao, Weigi. 2007. Getting Sun Java 5 On Debian 4.0: So Easy. [http://www.weiqigao.com/blog/2007/06/21/getting\\_sun\\_java\\_5\\_on\\_debian\\_4\\_0\\_so\\_easy.html](http://www.weiqigao.com/blog/2007/06/21/getting_sun_java_5_on_debian_4_0_so_easy.html)

Intriglia, Jim et. al. 2009. Jira 4.x on Linux. <http://forums.atlassian.com/thread.jspa?threadID=38085>

Jameson, Rosie ([rosie@atlassian.com](mailto:rosie@atlassian.com)). Installing JIRA Standalone on Unix or Linux. <http://confluence.atlassian.com/display/JIRA/Installing+JIRA+Standalone+on+Unix+or+Linux#InstallingJIRAStandaloneonUnixorLinux-2.SetJIRAHome>

Jameson, Rosie ([rosie@atlassian.com](mailto:rosie@atlassian.com)). Configuring JIRA to Send SMTP Email. <http://confluence.atlassian.com/display/JIRA/Configuring+JIRA+to+Send+SMTP+Mail>

Liu, Andrew ([aliu@atlassian.com](mailto:aliu@atlassian.com)). How do I unlock my JIRA home directory? <http://confluence.atlassian.com/pages/viewpage.action?pageId=195429049>

Kemp, Steve ([webmaster@debian-administration.org](mailto:webmaster@debian-administration.org)). Adding New Users. <http://www.debian-administration.org/articles/2>

Swift, Bob (<https://plugins.atlassian.com/vendor/details/90>). JIRA Command Line Interface plugin. <https://plugins.atlassian.com/plugin/details/6398>

Turner, Jeff ([jeff@atlassian.com](mailto:jeff@atlassian.com)). Changing JIRA Standalone's port. <http://confluence.atlassian.com/display/JIRA/Changing+JIRA+Standalone%27s+port>

Turner, Jeff ([jeff@atlassian.com](mailto:jeff@atlassian.com)). Increasing JIRA Memory. <http://confluence.atlassian.com/display/JIRA/Increasing+JIRA+Memory>

Turner, Jeff ([jeff@atlassian.com](mailto:jeff@atlassian.com)). Setting up JIRA Standalone and PostgreSQL on Linux. <http://confluence.atlassian.com/display/JIRA/Setting+up+JIRA+Standalone+and+PostgreSQL+on+Linux>

Turner, Jeff ([jeff@atlassian.com](mailto:jeff@atlassian.com)). Installing Java on Ubuntu or Debian. <http://confluence.atlassian.com/display/JIRA/Installing+Java+on+Ubuntu+or+Debian>

Verhas, Istvan ([istvan@verhas.com](mailto:istvan@verhas.com)). JIRA 4.x on Linux forum post. <http://forums.atlassian.com/thread.jspa?threadID=38085&tstart=0>

## XI. Glossary

CLI	An acronym for Command Line Interface. Command Line Interfaces can enable JIRA <a href="http://livepage.apple.com">livepage.apple.com</a> Administrators and Application Developers to save time and energy administering JIRA, developing JIRA applications, and automating JIRA processes.
Region	A term used to categorize a specific logical or physical computing platform where a software application is available to users that are dedicated to the accomplishment of a specific objective. For example, a “Development” region is dedicated to supporting application developers create new applications and develop new features for applications in production.
SDLC	Acronym that is used to refer to a <a href="#">System</a> or <a href="#">Software Development Life Cycle</a> methodology.
SSH	Secure Shell or SSH is a network protocol that allows data to be exchanged using a secure channel between two networked devices ( <a href="http://en.wikipedia.org/wiki/SSH">http://en.wikipedia.org/wiki/SSH</a> ).
VSP	An acronym for Virtual Service Provider. VSPs are third-party services that provide customers with outsourced virtual server instances to support application development activities and application hosting. The value propositions for VSP customers is realized in considerable cost savings associated with hardware purchase, system administration, customer/technical support staff, etc., typically required to manage an in-house application server configuration.

## XII. Document Change History

Version	Published	Description of Document Changes
1.0	11-5-2010	Initial release of JIRA on Debian Linux Administrators Reference Guide, as a “lite” version of the broader JIRA on Linux Administrators Reference Guide.

### XIII. Appendix A: Configuring and Troubleshooting PostgreSQL

#### A. Creating a New PostgreSQL Database for an Existing JIRA Instance

When upgrading to a new software release of JIRA, it is a common practice to create a new PostgreSQL database for the new JIRA instance. Once it is confirmed that the new JIRA release can connect with the new PostgreSQL database, the current JIRA data will be imported into the new database via the JIRA XML data import facility.

To get started creating a new PostgreSQL database for an existing JIRA instance, let's confirm the existence of the current JIRA PostgreSQL database, using the psql administration utility:

```
psql -h 127.0.0.1 -U jira jira_400
```

If you already are logged-in as the “jira” user, you can use this command:

```
psql jira_400
```

Once we are connected via psql to our JIRA v4.0 PostgreSQL database, we can view the tables in the database via psql commands:

```
select datname from pg_database;
```

```
jira_400=> select datname from pg_database;
 datname
-----
 template1
 template0
 postgres
 jira_400
(4 rows)

jira_400=> █
```

As part of our back-out strategy, we want to maintain the current JIRA v4.0 production platform in case we decide to migrate back from the version of JIRA that we are migrating to.

To support this strategy, let's create a separate PostgreSQL database for the JIRA v4.0.2 application. Logging out of psql (ctrl-d) and sudo'ed as the user “jira”, we enter the following PostgreSQL command:

```
createdb jira_402
```

We then test our connection to the new database via psql:

```
psql jira_402
select datname from pg_database;
```

```

jira@JIRAv4:~$ createdb jira_402
jira@JIRAv4:~$ psql jira_402
Welcome to psql 8.3.8, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help with psql commands
      \g or terminate with semicolon to execute query
      \q to quit

jira_402=> select datname from pg_database;
 datname
-----
 template1
 template0
 postgres
 jira_400
 jira_402
(5 rows)

jira_402=>

```

As the new PostgreSQL database is now created, complete the configuration instructions in [Appendix F](#) to connect the new JIRA instance with the new PostgreSQL database.

Once JIRA is connected to the new PostgreSQL database, you can import existing JIRA data into the new JIRA instance. See [Appendix D](#) for instructions on using JIRA's XML data import facility.

## B. PostgreSQL on Debian v5.x Lenny Notes

The default directory where the Jira PostgreSQL database files are located is `/var/lib/postgresql/8.3/main`:

```

Owl:/usr/lib/postgresql/8.3/bin# ls -l /var/lib/postgresql/8.3/main/
total 44
drwx----- 6 postgres postgres 4096 2009-10-27 23:47 base
drwx----- 2 postgres postgres 4096 2009-10-30 09:36 global
drwx----- 2 postgres postgres 4096 2009-10-18 10:12 pg_clog
drwx----- 4 postgres postgres 4096 2009-10-18 10:11 pg_multixact
drwx----- 2 postgres postgres 4096 2009-10-18 10:12 pg_subtrans
drwx----- 2 postgres postgres 4096 2009-10-18 10:11 pg_tblspc
drwx----- 2 postgres postgres 4096 2009-10-18 10:11 pg_twophase
-rw----- 1 postgres postgres 4 2009-10-18 10:11 PG_VERSION
drwx----- 3 postgres postgres 4096 2009-10-18 10:12 pg_xlog
-rw----- 1 postgres postgres 133 2009-10-30 05:02 postmaster.opts
-rw----- 1 postgres postgres 54 2009-10-30 05:02 postmaster.pid
lrwxrwxrwx 1 root root 31 2009-10-18 10:12 root.crt -> /etc/postgresql-common/root.crt
lrwxrwxrwx 1 root root 36 2009-10-18 10:12 server.crt -> /etc/ssl/certs/ssl-cert-snakeoil.pem
lrwxrwxrwx 1 root root 38 2009-10-18 10:12 server.key -> /etc/ssl/private/ssl-cert-snakeoil.key

```

## XIV. Appendix B: JIRA Troubleshooting

### A. Resetting a Forgotten or Corrupted Administrative Password

On one occasion, my administrative JIRA password no longer enabled me to login to JIRA. For situations like this, or the case where an administrator forget their password, the password can be reset by updating a specific JIRA table via SQL. Instructions to do this can be found at the Atlassian JIRA website:

<http://confluence.atlassian.com/display/JIRA/Retrieving+the+JIRA+Administrator>

For the sake of completeness, here's what you need to do if you installed PostgreSQL as your JIRA database via the procedure described in this document.

Step 1. Login to your JIRA server via ssh:

```
ssh -p nnnnn jim@xxx.xxx.xxx.xx
```

Step 2. Shutdown JIRA

```
./shutdown/sh
```

Step 3. Become the JIRA user:

```
sudo - jira
```

**NOTE:** Ensure that the PostgreSQL database server is still up and running, otherwise the commands that follow will not work. JIRA's PostgreSQL server, as installed via the procedure described in this document, should remain running after the `./shutdown.sh` command is issued.

After entering your sudo password, you will become the user "jira" and will now be able to access JIRA's PostgreSQL database.

STEP 3. Start the PostgreSQL command line interpreter:

```
psql
```

STEP 4. Enter the following SQL command to reset the Admin user password. Be sure to replace the "XXXX" with the Admin user's user id:

```
update userbase set password_hash='uQieO/1CGMUIXXftw3ynrsaYLSHl  
+GTcPS4LdUGWbIusFvHPfUzD7CZvms6yMMvA8I7FViHVEqr6Mj4pCLKAFQ==' where  
username='jimintriglia';
```

Step 5. Restart JIRA; Login as Administrator

You should now be able to login using your Administrator user ID and the word “sphere” for the password. Be sure to reset your admin password once you are able to login.

## B. Error Message: JIRA is Locked

If JIRA is not properly shutdown using the shutdown script provided by Atlassian (shutdown.sh) , you may find that you are unable to restart JIRA via the startup script. Reviewing JIRA’s Catalina.out log file, you see the following:

```
*****
JIRA startup failed, JIRA has been locked.
*****
```

JIRA “locks” it’s Home directory on startup, to ensure two instances of JIRA cannot be initiated using the same Home directory. If a server hosting a JIRA instance is shutdown or powered-off without first shutting down JIRA via the shut down script, the “lock file” that JIRA uses to lock it’s Home directory will remain in a state that causes JIRA to not startup.

To remedy this situation, go to JIRA’s Home directory and locate the lock file jira-home.lock:

```
JIRAv4:/usr/local/jira# ls -al
total 44
drwxr-xr-x 6 jira jira 4096 2009-12-20 05:54 .
drwxrwsr-x 11 root staff 4096 2009-11-27 04:50 ..
-rw----- 1 jira jira 75 2009-12-08 04:22 .bash_history
-rw-r--r-- 1 jira jira 220 2008-05-12 11:00 .bash_logout
-rw-r--r-- 1 jira jira 3116 2008-05-12 11:00 .bashrc
drwxr-xr-x 3 root root 4096 2009-11-27 08:03 caches
drwxr-xr-x 3 root root 4096 2009-11-27 08:03 data
drwxr-xr-x 2 root root 4096 2010-01-09 07:10 export
-rw-r--r-- 1 root root 0 2009-12-20 05:54 .jira-home.lock
drwxr-xr-x 5 root root 4096 2009-11-27 08:03 plugins
-rw-r--r-- 1 jira jira 675 2008-05-12 11:00 .profile
-rw----- 1 jira jira _ 220 2009-12-08 04:22 .psql_history
```

Delete the lock file and restart JIRA via startup.sh, located in JIRA’s /bin subdirectory.

**Note:** In some instances, you may have to run `./shutdown.sh` first before running `./startup.sh` to enable JIRA to successful initialize and start.

## XV. Appendix D: JIRA XML Backup and Restore Facility



JIRA provides an XML-based Backup and Restore Facility from within JIRA, via the System Administrator’s tab. The utilities are located in the Import and Export section of the System’s Administrator’s panel.

A JIRA XML backup file contains all of a JIRA instance’s data, with the exception of attachments (these are stored separately in the JIRA home directory). A JIRA XML Backup therefore contains JIRA configuration information, including the administrator’s user ID and password (encrypted), all user profile data, and all issue data.

Restoring a JIRA instance from a JIRA XML backup file will overwrite all JIRA data, with the exception of attachments (these must be restored manually).

By default, JIRA is configured to [automatically perform an XML Backup](#) every 720 minutes (12 hours). You can check whether JIRA is configured to perform an automatic backup by viewing Services under the Systems section of the Administrator’s panel:

Services <span style="float: right;">?</span>			
Name / Class	Properties	Delay (mins)	Operations
<b>Backup Service</b> <small>com.atlassian.jira.service.services.export.ExportService</small>	<b>USEZIP:</b> Zip <b>USE_DEFAULT_DIRECTORY:</b> true	720	<a href="#">Edit</a> <a href="#">Delete</a>
<b>Service Provider Token Remover</b> <small>com.atlassian.sai.jira.scheduling.JiraPluginSchedulerService</small>	<b>pluginJobName:</b> Service Provider Token Remover	480	<a href="#">Edit</a> <a href="#">Delete</a>
<b>Mail Queue Service</b> <small>com.atlassian.jira.service.services.mail.MailQueueService</small>		1	<a href="#">Edit</a>

Periodically, JIRA System Administrators will need to archive XML Backup files to off-server storage to ensure that the JIRA instance does not run out of disk space. One strategy for selecting files for off-server archiving would be to choose files from the 1st, 15th and 30th of each month, deleting all of the files once the backup files have been safely copied to a SAN (Storage Area Network) or NAS (Network Attached Storage).

JIRA stores XML backup files in it’s export subdirectory of JIRA’s Home directory. Note that JIRA’s Home directory is configured during installation and the location can vary depending on the system administrator’s preference. For example, JIRA XML backup files might be located at /usr/local/jira402/export.

```
JIRA04:/usr/local/jira402/export# ls -l
total 288
-rw-r--r-- 1 root jira 47235 2010-03-16 05:03 2010-Mar-16--0503.zip
-rw-r--r-- 1 root jira 46947 2010-03-16 05:21 2010-Mar-16--0521.zip
-rw-r--r-- 1 root jira 47009 2010-03-16 17:22 2010-Mar-16--1722.zip
-rw-r--r-- 1 root jira 47012 2010-03-17 05:23 2010-Mar-17--0523.zip
-rw-r--r-- 1 root jira 47415 2010-03-17 17:24 2010-Mar-17--1724.zip
-rw-r--r-- 1 root jira 48955 2010-03-18 05:25 2010-Mar-18--0525.zip
```

Alternatively, they may be located at `/var/jira402/export`. It all depends on what the system administrator decided when s/he installed the JIRA instance.

For more information on backing-up JIRA 4.x instances, refer to Atlassian's JIRA Backup [resource page](#).

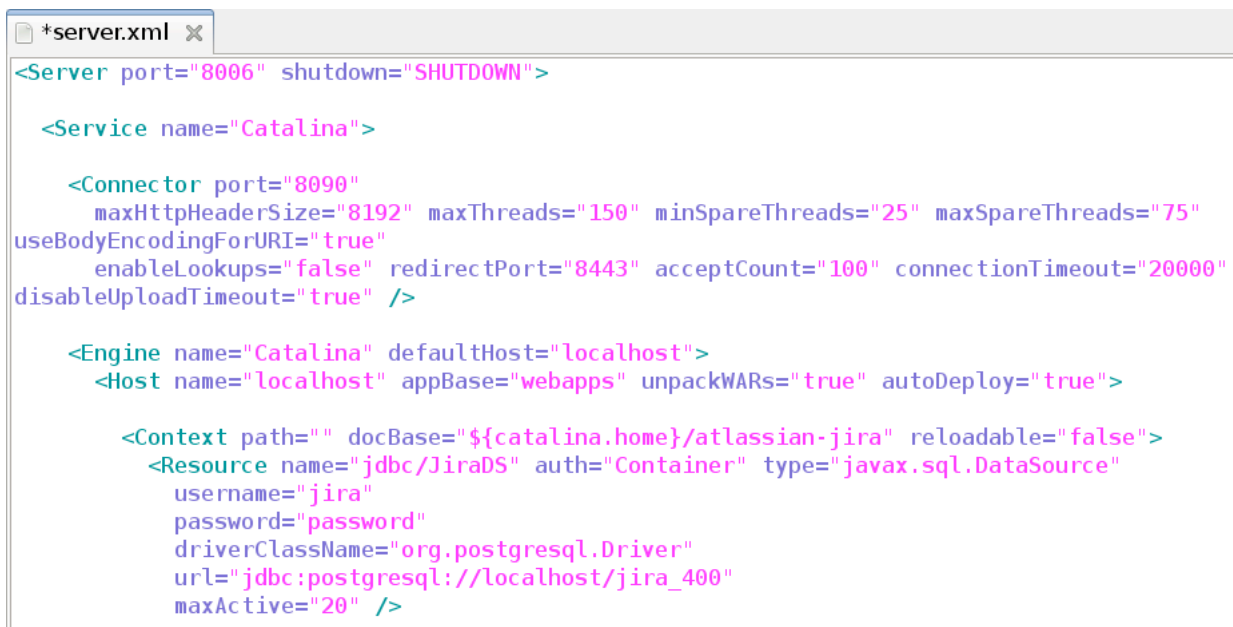
## XVI. Appendix F: Configuring JIRA's Server.xml File

The settings contained in JIRA's server.xml configuration file control access to JIRA's database, including settings for user ID and password. This configuration file also controls the port that the Coyote web server will use to listen for JIRA http requests.

The server.xml file is located in the directory where JIRA is installed, in the configuration (./conf) subdirectory.

Example:

`/opt/jira402/atlassian-jira-enterprise-4.0.2-standalone/conf`



```

*server.xml x
<Server port="8006" shutdown="SHUTDOWN">
  <Service name="Catalina">
    <Connector port="8090"
      maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
      useBodyEncodingForURI="true"
      enableLookups="false" redirectPort="8443" acceptCount="100" connectionTimeout="20000"
      disableUploadTimeout="true" />
    <Engine name="Catalina" defaultHost="localhost">
      <Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
        <Context path="" docBase="${catalina.home}/atlassian-jira" reloadable="false">
          <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
            username="jira"
            password="password"
            driverClassName="org.postgresql.Driver"
            url="jdbc:postgresql://localhost/jira_400"
            maxActive="20" />
        </Context>
      </Host>
    </Engine>
  </Service>
</Server>

```

Some things to keep in mind when configuring the Server.xml file for your installation:

Apache and Coyote HTTP Server Conflicts To avoid [conflicts with other services](#) (like Apache) that are likely running on a shared application server, you may want to consider changing the default port for JIRA from the default "Server port=8005" and "Connector port="8080" to "8006" and "8090" respectively:

```

<Server port="8006" shutdown="SHUTDOWN">
  <Service name="Catalina">
    <Connector port="8090"
      maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25" maxSpareThr
      eads="75" useBodyEncodingForURI="true"
      enableLookups="false" redirectPort="8443" acceptCount="100" connectionTime
      out="20000" disableUploadTimeout="true" />
  </Service>
</Server>

```

- Edit the database access section of the server.xml file, to enable JIRA to communicate with PostgreSQL instead of the default HQLSDB database:
- Be sure to enter the username and password in the “Resource Name” section of the server.xml file, as this data will be needed by JIRA to access the PostgreSQL database.
- You may want to comment-out the section of the file that connects JIRA to the email client. This is to prevent unwanted emails being sent to users (from established filter subscriptions) when JIRA is brought-up for the first time.

---

For more information on configuring JIRA’s server.xml file for Linux and PostgreSQL installations, see:

- [Setting up JIRA Standalone and PostgreSQL on Linux](#)
- [Connecting JIRA to PostgreSQL](#)
- [PostgreSQL website](#)

## XVII. Appendix I - The JIRA Home Directory

### A. Configuring the Home Directory

### B. Restoring Issue Attachments

When creating a new JIRA instance or duplicating an existing JIRA instance, it is necessary to restore issue attachments. JIRA locates issue attachments in a series of subdirectories located in the JIRA Home Directory:

When creating a duplicate JIRA instance, JIRA system administrators will often use the UNIX tar utility to consolidate into a single compressed file all of the JIRA attachment subdirectories and associated attachment files on the source application server. Restoring the attachments tar file in the JIRA Home directory on the destination JIRA application server completes the process of duplicating JIRA attachments on the duplicate JIRA instance.

To port issue attachments to a target JIRA application server using the UNIX tar utility, proceed as follows:

Step 1. From the JIRA Home directory, navigate to the /data/attachments directory. Enter the following command:

```
tar cvf backup_attachments.tar ./
```

This command will recursively store all attachment folders and files in JIRA's /attachments subdirectory to the backup\_attachments.tar file.

Step 2. Copy the backup\_attachments.tar file to the /attachments directory for the new JIRA v4.0.2 instance. To unpack the attachment files into the /attachment subdirectory, use the tar command:

```
tar xvf backup_attachments.tar
```

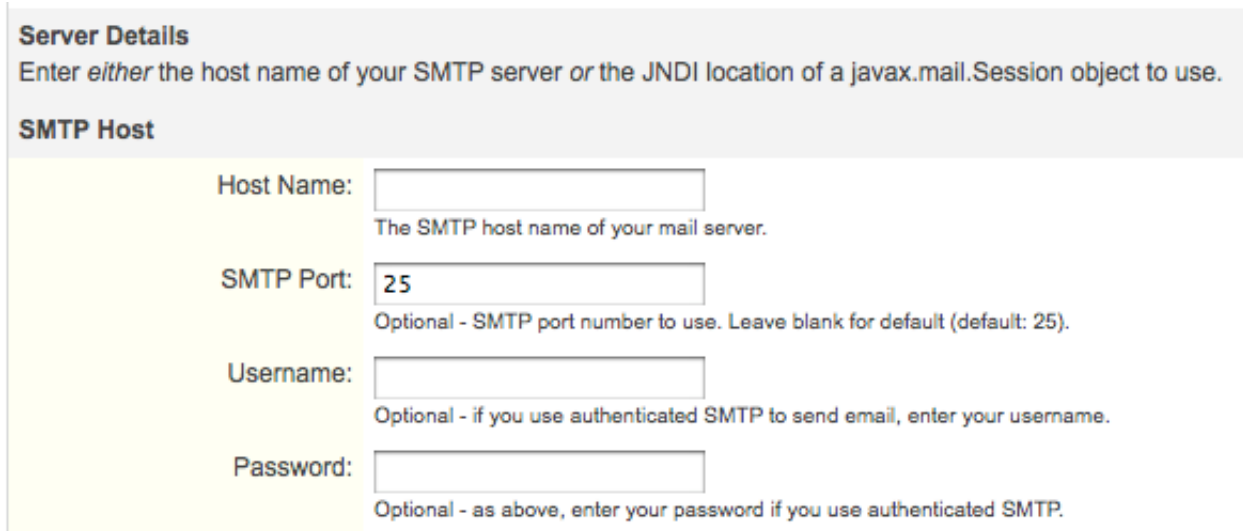
This completes the process of duplicating JIRA issue attachments from one JIRA region/application server to another.

To verify that JIRA issue attachments have been successfully exported to the duplicate JIRA instance, select a JIRA issue with an attachment. When the issue is displayed, click on the attachment; the attachment should den displayed.

## XVIII. Appendix K - Configuring JIRA to Send Email via Google GMail SMTP

Email notifications can be sent by configuring JIRA's Tomcat application server, to use authenticated SMTP to send email via a valid Google email account.

It's important to note that as of the JIRA v4.1.1 release, configuring JIRA email via GMail SMTP by simply completing the Global Settings->Mail Server web form does not work with GMail SMTP:



**Server Details**  
Enter *either* the host name of your SMTP server or the JNDI location of a javax.mail.Session object to use.

**SMTP Host**

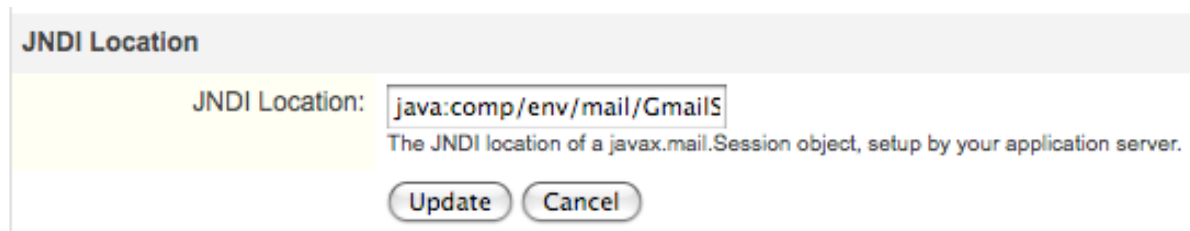
Host Name:   
The SMTP host name of your mail server.

SMTP Port:   
Optional - SMTP port number to use. Leave blank for default (default: 25).

Username:   
Optional - if you use authenticated SMTP to send email, enter your username.

Password:   
Optional - as above, enter your password if you use authenticated SMTP.

What does work is configuring Java JNDI services to send email via Tomcat:



**JNDI Location**

JNDI Location:   
The JNDI location of a javax.mail.Session object, setup by your application server.

This approach requires updating JIRA's Server.xml configuration file and entering the parameter "java:comp/env/mail/GmailSmtServer" in the JNDI Location box shown above.

### Step 1. Copy/Remove Java .Jar Files from JIRA/Tomcat Library Directories

To enable JIRA to send SMTP mail via an existing Google GMAIL account, you first need to copy a couple of files Tomcat's library directory. These files also need to be deleted from from JIRA's web application library directory.

The JIRA v4.1.1 release was bundled with a new release of Apache Tomcat, v6.0.2. As this new release of Tomcat relocated Tomcat's library (/lib) to a new location, you will need to copy the two Tomcat .jar files to support JIRA mail to different directories depending on what version of JIRA you are configuring:

### JIRA v4.1.1 with Tomcat v6.0.2 and above

```
cp /opt/jira/atlassian-jira-enterprise-4.0-standalone/atlassian-jira/WEB-INF/lib/activation-1.1.1.jar /opt/jira/atlassian-jira-enterprise-4.0-standalone/lib/
```

```
cp /opt/jira/atlassian-jira-enterprise-4.0-standalone/atlassian-jira/WEB-INF/lib/mail-1.4.1.jar /opt/jira/atlassian-jira-enterprise-4.0-standalone/lib/
```

#### JIRA v4.0.x with Tomcat v5.x.x and above

```
cp /opt/jira/atlassian-jira-enterprise-4.0-standalone/atlassian-jira/WEB-INF/lib/activation-1.1.1.jar /opt/jira/atlassian-jira-enterprise-4.0-standalone/common/lib/
```

```
cp /opt/jira/atlassian-jira-enterprise-4.0-standalone/atlassian-jira/WEB-INF/lib/mail-1.4.1.jar /opt/jira/atlassian-jira-enterprise-4.0-standalone/common/lib/
```

Verify that the files have been copied to Tomcat's library directory, and then delete the files from the JIRA web application directory:

```
rm /opt/jira/atlassian-jira-enterprise-4.0-standalone/atlassian-jira/WEB-INF/lib/activation-1.1.1.jar
```

```
rm /opt/jira/atlassian-jira-enterprise-4.0-standalone/atlassian-jira/WEB-INF/lib/mail-1.4.1.jar
```

#### Step 2. Configure JIRA's Server.xml File for SMTP Mail via GMail

Update JIRA's server.xml file by adding the following to the <Context Path> section of the file:

```
<Resource name="mail/GmailSmtpServer"
  auth="Container"
  type="javax.mail.Session"
  mail.smtp.host="smtp.gmail.com"
  mail.smtp.port="587"
  mail.transport.protocol="smtp"
  mail.smtp.auth="true"
  mail.smtp.user="<someuser@gmail.com>"
  password="<someuser password"
  mail.smtp.starttls.enable="true"
  mail.debug="true"
/>
```

For <someuser@gmail.com>, insert the GMail account that is going to be used send JIRA email notifications. Enter the associated password for that account for the <someuser password> parameter show above. Once you have successfully tested the email configuration, you can set the "mail.debug=false".

Substitute the Gmail email address and associated password for the "mail.smtp.user" and "password" parameters in the section above. Once you have email notifications working through gmail, you can omit the "mail.debug=true" parameter, to eliminate the detail email debugging info that will be sent to the Catalina.out log file.

Save the changes to the server.xml file, restart JIRA and login to a System Administrator account.

Step 3. Test Email Configuration - Click "Update" and then click on the "Send a Test Email" link to test your connection. You should get a confirmation message that your test email was sent successfully.

## XIX. Appendix L - Configuring JIRA Entity Engine

JIRA's entityengine.xml file must be updated to ensure JIRA connects with the Postgresql database rather than the default database that comes configured with JIRA.

In the `./WEB-INF/classes` directory, make a copy of the entityengine.xml file:

```
Owl:/opt/jira/atlassian-jira-enterprise-4.0-standalone/atlassian-jira/WEB-INF/classes# cp entityengine.xml entityengine.xml.orig
```

Edit the datasource and scheme name settings as below. Be sure the "field-type-name" is set to "postgres72" as this literal does not reflect the actual drive that has been installed. Also, change "schema-name" to "public" (all lowercase characters):

```
<datasource name="defaultDS" field-type-name="postgres72"
  schema-name="public"
```

## XX. Appendix N - Disabling Secure Linux

Atlassian's JIRA installation instructions do not specifically call for the disabling of the Secure Linux (SELINUX) application that is automatically enabled with some Linux distributions. Other application providers however, recommend disabling SELINUX when application software is installed in the /opt directory.

What follows below is one approach to disabling SELINUX for administrators who want to avoid the possibility of this application causing problems with installing, configuring or running JIRA. At a later date, I will research how SELINUX should be configured for JIRA application servers.

Step 1. Check if SELinux is presently running:

```
[root@localhost etc]# sestatus
SELinux status:                enabled
SELinuxfs mount:              /selinux
Current mode:                  enforcing
Mode from config file:        enforcing
Policy version:                21
Policy from config file:      targeted
[root@localhost etc]#
```

Step 2. Deactivate SELinux

To deactivate SELinux, edit the configuration file /etc/selinux/config. In this file, you will need to disable the parameter "SELINUX = disable". The edited SELinux configuration file should look like this after editing:

```
[root@localhost selinux]# cat config
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - SELinux is fully disabled.
# SELINUX=enforcing
SELINUX=disabled
# SELINUXTYPE= type of policy in use. Possible values are:
#     targeted - Only targeted network daemons are protected.
#     strict - Full SELinux protection.
SELINUXTYPE=targeted

# SETLOCALDEFS= Check local definition changes
SETLOCALDEFS=0
```

Once this change has been made, restart the JIRA server so the change will take effect.