## Setup Tomcat Locally

To install the tomcat server, enter the following command in the terminal prompt:

**sudo apt-get install tomcat7**

This will install a tomcat server with just a defalut ROOT webapp that displays “it works” page by default. Please see [Tomcat documentation](http://tomcat.apache.org/tomcat-6.0-doc/index.html) for more complete configurations. You can access tomcat in browser by typing localhost:8080 in navigation bar.

Default server port

To install the tomcat server, enter the following command in the terminal prompt:

By default tomcat runs on HTTP port 8080. You might want to change it to avoid conflict with other programs running on the same port. This can be done by accessing /etc/yourTomcat/conf/server.xml file.

**<connector port=”8080” protocol=”HTTP/1.1”**

**connectionTimeout=”20000”**

**redirectPort=”8443” />**

Chage the port to your desired number.

Users and Roles

/etc/yourTomcat/conf/tomcat-users.xml file contains the roles and users. Find a user name or create a new one for admin or manager role to use tomcat.

**<role rolename=”admin” />**

**<user username=”tomcat” password=”s3cret” roles=”admin” />**

Remember: After making any changes in the configuration files you must restart the tomcat to take effect of those changes.

This can be achieved by shutting down and then starting again the tomcat by following commands

**sudo etc/yourTomcat/bin/shutdown.sh**

**sudo etc/yourTomcat/bin/startup.sh**

## Use tomcat and SSH

SSH is a powerful collection of tools for the remote control of networked computers and transfer of data between networked computers.

The OpenSSH server component, **sshd**, listens continuously for client connections from any of the client tools. When a connection request occurs, **sshd** sets up the correct connection depending on the type of client tool connecting. OpenSSH can use many authentication methods, including plain password, public key, and **Kerberos** tickets.

### Installation

To install the OpenSSH client applications on your Ubuntu system, use this command at a terminal prompt:

**sudo apt-get install openssh-client**

To install the OpenSSH server application, and related support files, use this command at a terminal prompt:

**sudo apt-get install openssh-server**

Use SSH

Just test it from your home computer with the command

**ssh user@yourserver**

**password: yourpassword**

After successful login. Navigate to webapps of your server files to look into files, you can access files from the root directory.

**cd /usr/local/apache-tomcat-your-version/webapps/ or**

**cd /usr/local/apache-tomcat-your-version/webapps/ROOT**

Use mysql database

You can access the mysql database like localhost using credentials such as.

**mysql -u root -p**

**password: yourpassword**

Change Password for SSH username

To change your password for SSH account use this

**ssh -t user@yourpassword passwd**

**password: yourpassword**

Use changelog.groovy for database changes

If there is need to change the domain classes at any stage of the project. Strictly use changelog.groovy to update database. By default the changes you made in the domain classes are not updated in the changelog.groovy. So if somehow you have got your updates in the local database by setting the dbCreate to update the database. Changes from domain model are propagated to database, which means your changes are updated on local database but not on the changelog.groovy.

To take get to know of what has been changed in domain classes and not updated in the changelog.groovy run the following command.

**grails > dbm-gorm dbChange.groovy**

This creates a file with name “dbChange” that contains the the updated domain model changelogs. Simply copy all the file text and paste it in the changelog.groovy. Remember if you have removed anything from database schema, make sure you remove it from changelog.groovy too.

Best Practice: You can use the dbCreate=”update” for any changes you want to do in the database. But changelog has to run on every build. It is better to use changelog.groovy for all the database updates. Make sure you do not use dbCreate attribute along with changelog.groovy. dbCreate has high precedence and it will be implemented first, causing database errors while changelog.groovy runs after.