# Install Tomcat on your server

## Prerequisites

This tutorial assumes you have completed the guide "Getting ready for the Cloud". The droplet created in this tutorial, is the one we will use in the following.

## Tomcat Installation

Tomcat is a webserver that can host web applications via HTTP requests. Notice that your DigitalOcean droplet has a public IP address. This address can be accessed from anywhere on the internet. You will use your address to publish your applications for this semester.

Before we begin you should log into your droplet via SSH. If you have a bash terminal open this can be done by writing `ssh <yourUser>@server-IP`

Commands must be executed with root privileges, so write `sudo su` and type your password to enable root.

1. Install Tomcat8 and Tomcat8's admin interface
    1. In the terminal type:
        - `apt-get install tomcat8 tomcat8-admin`
        - `apt-get install haveged`[1]
2. Create an admin user for Tomcat
    1. Open `/etc/tomcat8/tomcat-users.xml` with a text editor (for instance `nano`):
       `nano /etc/tomcat8/tomcat-users.xml`
    2. Insert the following after line 21 in the file – *please use your own password*!

       ```
       <role rolename="manager-gui"/>
       <user name="admin" password="XXX" roles="manager-gui, manager-script"/>
       ```

    3. Save the file

       If you use nano to open the file, you can save it by pressing ctrl+x and confirming with Y+enter
3. Restart the Tomcat Server

   In terminal write:

   ```
   service tomcat8 restart
   ```

The server will initially run on port 8080. You can access your server by writing <server-IP>:8080 in your browser (do it).

---

[1] Randomness is extremely important in cryptography. Tomcat require, for security purposes random numbers, which on an Ubuntu box can lead to a problem with very long startup times. See this link for info about *haveged:* http://www.issihosts.com/haveged/

The admin interface is available via <Server-IP>:8080/manager. You can login with the name and password specified in step 2.

It's standard to host webservers on port 80. When doing this, the port number can be left out. Let's change the port into 80.

1. Change the port of Tomcat from 8080 to 80
    1. Open `/etc/tomcat8/server.xml`
    2. Replace `Connector port="8080"` with `Connector port="80"` in line 71
    3. Save the file
2. Allow Tomcat to use ports beneath 1023
    1. Open `/etc/default/tomcat8`
    2. Change `#AUTHBIND=no` to `AUTHBIND=yes` in line 46 (remember to remove the # sign)
    3. Save the file
3. Restart Tomcat server

    In terminal, write: `service tomcat8 restart`

## Deploy a Web Application to the Tomcat Server via Tomcats, Web Application Manager

If you have completed the steps above, and can access the Tomcat Manager via: <yourIP>/manager, we are ready to deploy our own applications to the server.

Use NetBeans to create a new maven Web Application.

Change the generated *index.html* file to something, just a little bit more interesting, and execute the web application from within NetBeans.

If everything was fine, do a *Clean and Build* and go to the folder that holds your web-project. Here you should find a folder `target`, and in that folder a file `xxxx.war`. WAR-files, are for Java web applications, what JAR-files are for traditional Java programs. It's just a zipped file; with all the files that make up your web-application (you should verify this, if you have never tried it before).

This is the file we are going to upload to our remote Tomcat Server.

Open the Tomcat Web Application Manager from the link in the startpage, and use the section "WAR file to deploy" to select your local WAR-file, and then click deploy.

Verify that it is added to the list of Applications and, more important; verify that we can access it on the remote server – How cool is that ☺

In order to re-upload your war file to implement changes to your web application remember to undeploy the application before going back to the section where you can upload the file

## Deploy a Web Application to the Tomcat Server via the Apache Tomcat Maven Plugin

We can simplify/automate the deployment process by use of Maven.

Add the following plugin to your pom.xml file (change values in red to reflect your own values):

```
<plugin>
    <groupId>org.apache.tomcat.maven</groupId>
    <artifactId>tomcat7-maven-plugin</artifactId>
    <version>2.2</version>
    <configuration>
       <url>http://46.101.196.110/manager/text</url>
       <path>/demo</path>
       <username>admin</username>
       <password>XXX</password>
     </configuration>
</plugin>
```

Execute the Goal, both from inside NetBeans and without NetBeans and verify that you can deploy your project using Maven.

See link for additional information: http://tomcat.apache.org/maven-plugin-2.0/tomcat7-maven-plugin/

## Log-files and Console Output (System.out.println)

Your first two semesters should have proven that access to Console outputs and Exception output is a vital thing for both developing and maintaining software. Up until now this has never been a problem, since almost everything we have done has been executed from within NetBeans. But now, our software executes on a remote server.

Let's see how we can access this information on the remote server

1)

Use the NetBeans Wizard to create two servlets A, and B in the web project created above. Leave the generated code as is, except for these two additions:

In the A-servlet, insert a line like this:

```
System.out.println("Hi I'm a message from Servlet A");
```

In the B-servlet, insert this line (what would you expect from this?):

```
int x = 100 / 0;
```

Execute the web application from within NetBeans, and visit both the A+B-servlet URL with your browser (on localhost). Investigate the generated outputs from the two servlets. What messages goes where?

Note: If you don't see anything, you might need to go to the services tab, select Servers and right-click the icon for your Tomcat-server. Now select "View Server Log" and "View Server Output".

2)

Now use the Web Application Manager to undeploy the previous version of your web application and deploy the version with the two servlets. Verify that the application has been deployed, and visit both the A+B-servlet URL with your browser.

Now, let see whether we can find the expected output (message + exception) on our server.

Open a git bash terminal and connect to your server the usual way. If not connected as root, upgrade yourself to have root privileges via:

```
sudo  su
```

Navigate into the folder that holds the log-files using the command:

```
cd /var/log/tomcat8
```

Here you should find a number of files using the `ls` command.

Check the content of the files with the cat command like: `cat <filename>`

When you have localized the files that contain the expected output statements you should:

3) Open a new terminal and connect to server the usual way.

CD into the folder with the log-files: `cd /var/log/tomcat8`

You should now have two terminals connected to your remote server, both currently in Tomcats log folder.

In one of them, type: `tail -f catalina.out` (use `tail --help` or `man tail` for info)

In the other terminal window, restart the server by: `service tomcat8 restart`

Explain the output from the catalina.out file.

Now in the terminal you just used to restart the server, type:

```
clear
```

```
tail -f localhost.XXXXXXX.log
```
where xxx is the newest log-file

Now arrange your windows, so you, simultaneously can see:

- The two terminal windows
- Your browser window

Visit the A+B-servlet URLs with your browser and observer the outputs in the two terminal windows.

*In the future, whenever a web application does not work as expected when uploaded to the remote server, make sure to check the log and console outputs before you ask others for help → This will be the first thing they ask you to do ;-)*