

Linux DEB (Debian) Installation



Before installing NetVizura make sure to set the time on your server correctly. Time change after the installation will invalidate the license!



Before installing NetVizura you will have to install: Oracle Java 1.7, Tomcat 7 and PostgreSQL 9.2 or higher, in that order. The installation process has been tested on Debian 7.

On this page:

- [NetVizura Installation Steps](#)
- [Post Install Steps](#)
 - [Tweaking Tomcat memory allocation](#)
 - [Tweaking PostgreSQL](#)
 - [PostgreSQL "safe" tweaks](#)
 - [PostgreSQL "unsafe" tweaks](#)

NetVizura Installation Steps

To install NetVizura follow these steps:

Step 1: sudo package installation: execute `apt-get install sudo`

Step 2: Oracle Java 1.7 package installation:



Default Java implementation is OpenJDK. You need to install Oracle Java package. Java packages should be installed before the Tomcat7 packages, if not Tomcat will use OpenJDK

1. To add the WebUpd8 Oracle Java PPA repository to the Software Sources in Debian, use the following commands:

```
echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu precise
main" | tee /etc/apt/sources.list.d/webupd8team-java.list
echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu precise
main" | tee -a /etc/apt/sources.list.d/webupd8team-java.list
apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys
EEA14886
```

2. execute command `apt-get update`
 - a. ignore the error about "public key is not available"
3. execute command `apt-get install oracle-java7-installer` and answer affirmatively to "Proceed without verification" and all other installation questions
4. execute command `ln -s /usr/lib/jvm/java-7-oracle /usr/lib/jvm/default-java` to set Oracle's Java as a default Java on the system
5. check if java is properly installed with command `java -version`



If you are behind a firewall / router that blocks some of the redirects required to download the Oracle Java archive, you can download the JDK tar.gz archive manually and place it under `/var/cache/oracle-jdk7-installer` - then, installing the "oracle-java7-installer" package will use the local archive instead of trying it to download it itself.

Step 3: Tomcat 7 package installation:

1. execute command `apt-get install tomcat7`
2. start Tomcat: `service tomcat7 start`
3. verify that Tomcat is running properly with the command `service tomcat7 status`

Step 4: PostgreSQL package installation

1. Create a file `pgdg.list` in `/etc/apt/sources.list.d/` with some text editor: `nano /etc/apt/sources.list.d/pgdg.list` and add the following line:
`deb http://apt.postgresql.org/pub/repos/apt/ wheezy-pgdg main`
2. execute command: `wget --quiet -O - http://apt.postgresql.org/pub/repos/apt/ACCC4CF8.asc | sudo apt-key add -`
3. execute command `apt-get update`
4. execute command `apt-get install postgresql postgresql-client`
5. verify that PostgreSQL is running properly with the command `service postgresql status`

Step 5: NetVizura packages installation

After this steps, install the NetVizura packages downloaded from the website with the command: `dpkg -i downloaded_file_name.deb`

To access the application, type <http://myip:8080/netvizura> in your browser. The default user account with administrator privileges is: username: **admin**, password: **admin01**

Post Install Steps

After installation it is needed to tweak the configuration files in order to utilize the installed RAM to the fullest extent. The main consumers of RAM are operating system, PostgreSQL database and Tomcat. A rule of thumb for distributing memory is to split it in ratio 2:1 between PostgreSQL and Tomcat with 1 GB or more reserved for operating system. For instance:

Installed RAM	PostgreSQL	Tomcat	OS
4 GB	2 GB	1 GB	1 GB
16 GB	10 GB	5 GB	1 GB

Tweaking Tomcat memory allocation

In the following example 1 GB of RAM is allocated for Tomcat process:

1. Edit file `/etc/default/tomcat7`
2. Locate `JAVA_OPTS` environment variable that defines memory and uncomment it if it is commented. This line looks something like the following:
`JAVA_OPTS="-Djava.awt.headless=true -Xmx128M -XX:+UseConcMarkSweepGC"`
3. Modify the `-Xmx` parameter to allocate additional memory to Tomcat. Additionally, set parameter `-Xms` to the same amount. This should look something like:
`JAVA_OPTS="-Djava.awt.headless=true -Xms1024M -Xmx1024M -XX:+UseConcMarkSweepGC"`
4. Edit file `/etc/init.d/tomcat7`
5. Locate `JAVA_OPTS` environment variable that defines memory. This line looks something like the following:
`JAVA_OPTS="-Djava.awt.headless=true -Xmx128M"`
6. Modify the `-Xmx` parameter. Additionally, set parameter `-Xms` to the same amount as `-Xmx`. This should look something like:
`JAVA_OPTS="-Djava.awt.headless=true -Xms1024M -Xmx1024M"`
7. Save the file and restart Tomcat: `service tomcat7 restart`

Tweaking PostgreSQL

Tweaking PostgreSQL for best performance is a topic on which many books were written, but the following are some common sense suggestions. In general there are two groups of PostgreSQL tweaks that are helpful for NetVizura performance - "safe" and "unsafe" tweaks. "Safe" tweaks are those which can be applied in all cases. "Unsafe" tweaks trade reliability for performance. For the curious ones recommended reads (among countless others) are [PostgreSQL Optimization Guide](#), [PostgreSQL Tuning Guide](#), [this article](#) and [this book](#).

In order to apply following tweaks edit file `/etc/postgresql/PG_VERSION_NUMBER/main/postgresql.conf`. You will need to restart the PostgreSQL service after done editing with command: `service postgresql restart`. Almost all of the following parameters are commented with carron character (#). Although these tweaks are considered "safe" do take notice of the default values. Usually you can comment out the parameter that has been changed and PostgreSQL will revert to the default value.

PostgreSQL "safe" tweaks

In the following example it is assumed that 4 GB of RAM is allocated for PostgreSQL.

parameter	recommended value	comment
<code>max_connections</code>	30	NetVizura rarely uses more than 10 connections simultaneously, but it is good to have some reserve

shared_buffers	1024MB	the recommended amount is $\text{RAM} / 4$
effective_cache_size	2048MB	the recommended amount is $\text{RAM} / 2$, possibly even $\text{RAM} * 3 / 4$
checkpoint_segments	32	for write intensive apps (as NetVizura) it should be at least 16, with 32 as safe maximum
checkpoint_completion_target	0.9	
default_statistics_target	100	
work_mem	8MB - 12MB	The formula used is $\text{max_connections} * \text{work_mem} \leq \text{RAM} / 8$, but using a bit more is still "safe"

PostgreSQL "unsafe" tweaks

These optimizations are considered "unsafe" since they *could* in very rare cases lead to data loss and/or corruption. If your VM is properly backed up we would consider the following optimizations safe. The following bring huge performance boosts to DB write process.

parameter	recommended value	comment
maintenance_work_mem	32MB	speeds up DB self clean process, not really important
wal_buffers	16MB	
full_page_writes	off	
fsync	off	don't wait for HDD to finish previous <i>write</i> operation. This brings the most benefit, but is considered potentially the most unsafe of all. If there is OS or HDD failure in exact instant when PSQL issues write command to HDD, that data will be lost and the DB itself could be corrupted. On the other hand, DB can issue several magnitude more write commands in the same time period and consider all these done, thus improving write performance immensely.
synchronous_commit	off	similarly to "fsync" but less unsafe and with less benefit
checkpoint_segments	64	how much is cached in temp files before it is issued to <i>proper</i> DB files. Issuing big chunks of data for write rarely is usually better for performance than issuing small chunks often