Linux RPM (CentOS) 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, Apache Tomcat 6 and PostgreSQL 9.3 or higher, in that order. The installation process has been tested on CentOS 6.6.

On this page:

NetVizura Installation Steps

To install NetVizura follow these steps:

Step 1: sudo command installation: `yum 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 Tomcat6 packages, if not Tomcat will use OpenJDK.

1. download .rpm JDK package from http://www.oracle.com/technetwork/java/javase/downloads/index.html
2. install the downloaded package: `rpm -Uvh file_name.rpm (example: rpm -Uvh jdk-7u79-linux-x64.rpm)`
3. execute the following commands (adjust the filepath to the JDK installation path if needed)
    a. `alternatives --install /usr/bin/java java /usr/java/jdk1.7.0_21/jre/bin/java 20000`
    b. `alternatives --install /usr/bin/javaws javaws /usr/java/jdk1.7.0_21/jre/bin/javaws 20000`
    c. `alternatives --install /usr/bin/javac javac /usr/java/jdk1.7.0_21/bin/javac 20000`
    d. `alternatives --install /usr/bin/jar jar /usr/java/jdk1.7.0_21/bin/jar 20000`
4. check if Java is properly installed with command `java -version`

Step 3: Apache Tomcat 6 package installation:

1. execute command `yum install tomcat6`
2. in folder */usr/sbin* edit file *tomcat6*: change the line "*set_javacmd*" to "*JAVACMD=/usr/java/latest/bin/java*"
3. save changes and start tomcat: `service tomcat6 start`
4. verify that Tomcat is running properly with the command `service tomcat6 status`
5. add Tomcat service to system startup: `chkconfig tomcat6 on`

Step 4: PostgreSQL package installation:

1. edit file */etc/yum.repos.d/CentOS-Base.repo*
    a. in section [base] add line "*exclude=postgresql**"
    b. in section [updates] add line "*exclude=postgresql**"
2. go to http://yum.postgresql.org/ and choose appropriate PostgreSQL package in regard to your CentOS version and architecture.
   CentOS 6, 64 bit example: http://yum.postgresql.org/9.3/redhat/rhel-6-x86_64/pgdg-centos93-9.3-6.noarch.rpm
3. in the folder where the file is downloaded execute command `yum localinstall pgdg-centos93-9.3-6.noarch.rpm`
4. execute command `yum install postgresql93-server`
5. execute command `service postgresql-9.3 initdb`
6. execute command `service postgresql-9.3 start`
7. verify that PostgreSQL is running properly with the command `service postgresql-9.3 status`
8. add PostgreSQL service to system startup: `chkconfig postgresql-9.3 on`

Step 5: Installing NetVizura packages

After this steps, install the NetVizura packages downloaded from the website with the command `rpm -ivh downloaded_file_name.rpm`

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/tomcat6/tomcat6.conf`
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="${JAVA_OPTS} -Xmx128M"`
   If it is not present add this line to the end of `tomcat6.conf` file.
3. Modify the `-Xmx` parameter to allocate additional memory to Tomcat. Additionally, set parameter `-Xms` to the same amount. This should look something like:
   `JAVA_OPTS="${JAVA_OPTS} -Xmx1024M -Xms1024M"`
4. Save the file and restart Tomcat: `service tomcat6 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 `/var/lib/pgsql/PG_VERSION_NUMBER/data/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
`max_connections`	30	NetVizura rarely uses more than 10 connections simultaneously, but it is good to have some reserve
`shared_buffers`	1024MB	the recommended amount is `RAM/4`
`effective_cache_size`	2048MB	the recommended amount is `RAM/2`, possibly even `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 `max_connections*work_mem <= 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
`maitenance_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 *write* 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 *proper* DB files. Issuing big chunks of data for write rarely is usually better for performance than issuing small chunks often