# Setting up the Neato SmartApps Server

Raja Software

Last updated: 08/04/2014

|  |  |  |
| --- | --- | --- |
| Date | Version | Author/Comments |
| 12/11/2012 | 0.1 | Initial draft |
| 12/18/2012 | 0.2 | Adding more details to app config and added folder screenshots. |
| 12/19/2012 | 0.3 | Add steps to setup ejabberd, assumptions section, etc. |
| 12/20/2012 | 0.4 | Added Troubleshooting section |
| 02/07/2013 | 0.5 | Updated screenshots, generic IP/port info, etc. |
| 05/28/2013 | 0.6 | Updated for RabbitMQ |
| 10/17/2013 | 0.7 | Updated for WordPress Integration |
| 11/06/2013 | 0.8 | Added more details about the WordPress setup |
| 01/26/2014 | 0.9 | Added more details on ejabberdmod\_eventful setup, troubleshooting tips. |
| 02/05/2014 | 0.91 | Added all the variations while setting up the application on CentOS |
| 02/07/2014 | 0.92 | Merging Christopher’s changes that documented a change in the amqp\_consumer’s environment variable settings. |
| 03/26/2014 | 0.93 | Minor formatting changes |
| 05/19/2014 | 0.94 | Adding information about new consumers, ejabberd ulimit restrictions and adding more cases that we faced, in the troubleshooting section. |
| 07/11/2014 | 0.95 | Added details about the deployment on cloud and various performance tuning changes, that can be done to improve the scalability. |
| 07/25/2014 | 0.96 | Added details about ejabberclt --concurrent option to avoid crash dumps, multiple concurrent RabbitMQ consumers and cron jobs to log out inactive robots and wipe out web service logs. |
| 07/31/2014 | 0.97 | Updating the documentation with httpc issue on the Ubuntu 14.04 and adding the changes that were done to go beyond 65K concurrent connections limit that we were facing on the ejabberd. |
| 08/04/2014 | 0.98 | Adding the steps that need to be performed when the iOS push notification changes. |

# Summary

This document explains how to setup the NeatoSmartApps server application on Production Server and Developer machines. This document first explains how to setup the required environment and then how to setup the code and relevant databases, configurations.

# Assumptions

This document assumes that server environment is Ubuntu 12.04 and development environment is Windows. If you have a CentOS 6.0 server, please follow the steps mentioned under Ubuntu and the follow the additional steps mentioned in the CentOS section. For other flavors and versions of Linux, the steps mentioned in the Server Environment Setup will still work with minor changes and would have to look at various log files to know problems.

This document uses the host name and IP address in various places. Please note that:

* IP address of a machine can be found by typing command “ipconfig” on windows or “ifconfig” on a Linux machine.
* You can use “localhost” OR “127.0.0.1” as IP address, instead of the machine’s IP address if you are browsing the application from the same machine from where your application is running (i.e. Apache web server is running on the same machine).
* Host name is the DNS name that is mapped to the specific IP address.

If you come across a line like “*open a browser window and type: http://<ipaddress>/ neato/index.php”*, you can type any of these four addresses in the browser and it would give some output:

* [http://myip/neato/index.php](http://50.56.121.252/neato/index.php)
* <http://myhostname/neato/index.php> assuming that myhostname.com is mapped to IP address **myip**
* <http://localhost/neato/index.php> if you opened the browser window on same machine where the application is running.
* <http://127.0.0.1/neato/index.php> if you opened the browser window on same machine where the application is running.

# Requirements

Neato web application requires following installations on the server/developer machines:

* PHP 5.3 and above (<http://php.net/>)
* Apache 2.2.22 web server (<http://www.apache.org/>)
* MySQL 5.5 and above (<http://www.mysql.com/>)
* PHPMyAdmin (<http://www.phpmyadmin.net/>) (An optional web based MySQL management console if you want easy access to MySQL database instead of using command line utilities.)
* Ejabberd (<http://www.ejabberd.im/>) (This is required on server but can be skipped for development environment.)
* RabbitMQ ([http://www.rabbitmq.com](http://www.rabbitmq.com/)). This is required on server but is optional on the development environment.

The next few sections explain how to install these dependencies on Server and Developer environments.

# Server Environment Setup

### PHP, MySQL and Apache installation

First, login as root on your Linux instance and follow the steps below. If you do not have direct access to root account, you would need to add **sudo** in front of every command mentioned below like

**mkdir abc** would change to **sudo mkdir abc**

* "apt-get" is a tool to do the 'installations' on Ubuntu machines. First update apt-get itself by running this command so that we get all the installations to the latest versions: Note that if you are on CentOS, replace apt-get with yum-install, which does the “installations” on the CentOS machines.

**apt-get update**

* Install PHP, Apache, and the 'connectivity' that we require between these two.

**apt-get install apache2 php5 libapache2-mod-php5**

After the command above, if you open your browser and type

[**http://myip**](http://myip) **OR** [**http://myhostname.com**](http://neatoroboticswebapp.com)

You should see 'It’s working!' as a response. This means that Apache is running fine.

* Apache creates a new directory structure inside var folder. If you go to **/var/www/** directory you would see an **index.php** file. This file has “it’s working!” written here. Thing to note here is, by default **/var/www/** is treated as the document root directory of Apache. This means, whatever you would be adding inside **/var/www/** directory**,** would be served by Apache. For example, if you create a folder named test inside **/var/www** (i.e. ***/var/www/test/***) and put a text file named ***sample.txt*** inside test folder (i.e. **/var/www/test/sample.txt**), you can see this file by typing in following URL in your web browser:

[http://myip/test/sample.txt](http://50.56.121.252/test/sample.txt)

* As application also needs MySQL, lets install mysql-server, mysql client and php5-mysql connectivity related libraries. You would be asked for password for the root user of the MySQL in this installation step. Remember this password as it would come handy when you are setting up the Application environment.

**apt-get install mysql-server mysql-client php5-mysql**

* After the command above, you would be able to access the MySQL using the command line client by giving following command. Note that there will not be any space between -p and the password that you gave in the steps mentioned above.

**mysql -u root -p<Password that you gave in the step above>**

* By now, we got PHP, Apache, and MySQL Setup. Problem is, it is tedious for the developers to login to the system and use MySQL using command line.
* We would install phpmyadmin that would help us access this MySQL instance directly using a browser. The command below would install phpmyadmin. It would first ask you on which web server you want to run phpmyadmin, please select apache. Next it would ask you admin password for phpmyadmin.

**apt-get install phpmyadmin**

* After this installation you should be able to view the phpmyadmin by going to following URL in your browser. From here you can create the DB Schema/data that you need for application. PHPMyadmin has a decent help manual, please refer it in case you need any help in there.

[**http://myip/phpmyadmin**](http://myip/phpmyadmin) **OR** [**http://myhostname.com/phpmyadmin**](http://myhostname.com/phpmyadmin)

* By now, you are done with the basic setup and now we are ready to move your code. If you want to move the code to the root directory, you can do so by using **SCP/Filezilla** and move codebase to **/var/www** directory. Make sure that you remove index.html or else you would have to explicitly mention **http://<IP\_ADDRESS>/index.php** to see your code working (as by default, index.html takes precedence over index.php, in Apache).

As the source code is checked into Subversion (SVN), we need to install the clients for them using following commands:

**apt-get install subversion**

* Few other small setups that come handy are mentioned below. Ubuntu 12.04, comes with nano editor. If you like vi over any other text editors, you can install it by giving following commands. You can edit your config files using vim (vi) or nano. CentOS has vim installed by default so you can skip this step altogether.

**apt-get install vim**

* Other module that we need is, enabling mod\_rewrite (an apache module which is used for SEO friendly URLs).Type following command for the same

**a2enmod rewrite**

* Now we have enabled the mod rewrite but Apache still will not allow the rewrites as the default behavior of Apache is to not to disable allow rewrites. If you are on CentOS, Apache would be called httpd instead of apache2. To make that adjustment, make sure that in all the commands where we are referring the Apache location/process as apache2, replace apache2 with httpd and follow the steps as they are documented.

For example, to restart Apache on Ubuntu you would fire:

**/etc/init.d/httpd restart**

Where in on CentOS, you would have to fire

**/etc/init.d/httpd restart**

* Follow the command below and make changes as mentioned below. Make sure that you should NOT have **AllowOverride = None** and it should be changed to **All** here.

**vim /etc/apache2/sites-available/default**

**<Directory /var/www/>**

**Options Indexes FollowSymLinksMultiViews**

**AllowOverride*All***

**Order allow,deny**

**allow from all**

**</Directory>**

After that you would have to restart the apache so that this **mod\_rewrite** can take effect. The command below would restart Apache.

**/etc/init.d/apache2 restart**

Application also needs **curl** to support various web services. You can enable curl by this command:

**apt-get install php5-curl**

* Now let’s install pear and mail and SMTP packages that are required for sending emails.

**apt-get install php-pear**

**pear install -o Mail**

**pear install -o Net\_SMTP**

* Other thing that we need is changing some PHP settings. We want to change the file upload limit so that user can upload bigger maps/schedule related data.

**vim /etc/php5/apache2/php.ini**

and edit

**post\_max\_size = 32M**

**upload\_max\_filesize = 32M**

Note that ANY change in php.ini will NOT take effect unless you restart the server

**/etc/init.d/apache2 restart**

One of the key configuration changes that you would have to do if you are on CentOS is to set the content-length in the header of the HTTP response. On Ubuntu, Apache by default starts setting the content-length to all the HTTP responses by default but on CentOS it needs to be configured properly. There are multiple ways to do it but the easiest and most consistent way is to configure the mod\_deflate module for the Apache. This module forces Apache to set content length in the response header and additionally compresses the HTTP response so that the response delivery to client is faster. All the major browsers and the curl implementations across the languages, support the decompression of the response before doing any further processing on it. Note that mod\_deflate is already installed on CentOS and all we are doing is configuring it properly. To configure it on CentOS, follow these steps:

Create a file named http-deflate.conf, and open it for edition:

**vim/etc/httpd/extra/httpd-deflate.conf**

In this file, add following snippet. The snippet below basically sets what all content types need to be compressed while sending the response. It also takes care of not compressing the response when the requesting browser is an old one and hence can not handle decompression and so on.

#Set to gzip all output

SetOutputFilter DEFLATE

#exclude the following file types

SetEnvIfNoCaseRequest\_URI \.(?:exe|t?gz|zip|iso|tar|bz2|sit|rar|png|jpg|gif|jpeg|flv|swf|mp3)$ no-gzipdont-vary

#set compression level

DeflateCompressionLevel 9

#Handle browser specific compression requirements

BrowserMatch ^Mozilla/4 gzip-only-text/html

BrowserMatch ^Mozilla/4.0[678] no-gzip

BrowserMatchbMSIE !no-gzip !gzip-only-text/html

SetEnvIf User-Agent ".\*MSIE.\*" nokeepalivessl-unclean-shutdown downgrade-1.0 force-response-1.0

### Ejabberd

The Neato SmartApps mobile application and web application use XMPP server to interact with robots, and vice-versa. We have used ejabberd as the XMPP server.

To install ejabberd, run:

**sudo apt-get install ejabberd**

After it is installed, you need to edit the configuration file that can be found at

***/etc/ejabberd/ejabberd.cfg***

Open this configuration file in an editor (we have been using vi) and add the host name you want the server to use. We would be changing the line where it mentions

**{hosts, ["localhost"]}.**

to

**{hosts, ["localhost","myhostname.com"]}**

Please note that host name here (**myhostname.com**) would mean that all the chat IDs that are created are created with suffix @**myhostname.com**. Please take a note of the host name that you have specified as it would be required while you are providing the host name in the configuration file of the application (explained at the later part of this document).

Additionally, ejabberd has a web based admin console so that you can manage this XMPP server from a web console. Access to this console needs to be restricted so we would create an admin user. In the ***/etc/ejabberd/ejabberd.cfg***config file, let’s add an admin level user. Please change the line that says,

**{acl, admin, {user, "", "localhost"}}**

to

**{acl, admin, {user, "administrator", "myhostname.com"}}.**

This change would add an admin level user named “administrator” (you can use any other username that you prefer) to the ejabberd admin for the host **"myhostname.com".**

If you notice, we have not given a password to this administrator user. Please run following command:

**sudo ejabberdctl register administrator myhostname.com myPassword**

Please change **myPassword** to the administrator password that you want to give. As any configuration changes would not take effect until you restart the ejabberd server, let’s restart the service by giving following commands:

**sudo service ejabberd restart**

Now you can browse the administration console of this XMPP server by going to the URL:

<http://myip:myport/admin/> or <http://myhostname.com:myport/admin/>

Typically, ejabberd runs on port 5280 (replace myport with 5280 above) – this is configurable in the ***ejabberd.cfg*** file. (Note: If are having trouble accessing the admin page it may be because port 5280 is blocked by your firewall – see the troubleshooting section for all the ports you need to open for this setup).

For firing ejabberd commands, Apache needs sudo permission. Apache runs under a user named www-data. To avoid any security risk, do not give blanket sudo permission to Apache. Instead give sudo permission only for ejabberd. For granting sudo permission to www-data user, add following snippet in **/etc/sudoers** file to set sudo permission for the www-data user:

*www-data ALL= NOPASSWD: /usr/sbin/ejabberdctl*

For detailed information on how to use admin console of ejabberd and what all can be done from there, please refer online documentation of ejabberd at [http://www.ejabberd.im/files/doc/guide.html#htoc20](http://www.ejabberd.im/files/doc/guide.html%23htoc20)

ejabberdctl is an ejabberd admin command that is heavily used by the application to interact with ejabberd.

As there might be multiple ejabberdctl commands being executed at the same time in the application, you would start seeing lot of harmless erl\_crash.dumps. These crash dumps are caused because while 1 instance of ejabberdctl command was being executed, another ejabberddctl command is executed.

You can bypass this by adding --concurrent option with the ejabberdctl. This would ensure that system spawn multiple instances of ejabberdctl in parallel and each ejabberdctl command would be executed in it's own instance. ejabberdctl with --concurrent option will ensure that no clash will ever occur. This also means that there may be some performance implications at both ejabberd and Apache machine because of this option but our performance test cases indicated that this performance hit is negligible.

If you are running ejabberd on same machine where the Apache is running, you can enable it by adding --concurrent option in the ‘ejabberdctl' key of the main.php config file (/var/www/Neato\_Server/Server\_Yii/Neato/protected/config/main.php) like following snippet.

'ejabberdctl'=>'sudo ejabberdctl --concurrent'

If ejabberd is running on different box, you would have to make a change in the ejabberdctl file of client machine (that is where Apache and RabbitMQ are running) with the given ejabberdctl file. This file is located at /usr/sbin/ejabberdctl but you can always find out the location of this file by firing "which ejabberdctl" command. You should open this file in the vi editor and replace the block that starts from line # 28 to line # 34 and replace it with the block below. By doing this, we would be actually adding concurrent option to all the commands as by default ejabberdctl does NOT support both –concurrent and –node option together.

if [ $# -ne 0 ] ; then

case $1 in

--) shift; break ;;

--node) shift ; ERLANG\_NODE=$1 ; shift ;;

# --concurrent) shift; SUFFIX=$$ ;;

esac

SUFFIX=$$ ;

fi

There are two additional ejabberd modules that need to be installed – mod\_admin\_extra and mod\_eventful – these are discussed next.

#### mod\_admin\_extra - Module Installation

The mod\_admin\_extra module should be installed by default on Ubuntu distributions but on CentOS, it is not installed by default when you install the ejabberd. If you are on ejabberd, , you need to perform following additional steps to install the module.

Download ‘ejabberd-modules’ using ejabberd’s SVN repository

**svn co https://svn.process-one.net/ejabberd-modules**

Change the current directory to ‘ejabberd-modules/mod\_admin\_extra/trunk/’

**cdejabberd-modules/mod\_admin\_extra/trunk/**

Compile the module

**./build.sh**

This will generate a  file ‘mod\_admin\_extra.beam’ and it will be present inside the ‘ebin’ folder.

Copy this file to the directory holding other .beam files for ejabberd

**cp ./ebin/mod\_admin\_extra.beam /lib/ejabberd/ebin/**

Enable the module by editing the config file. Open up ‘ejabberd.cfg’ file present at ‘/etc/ejabberd/’ folder.

**vi /etc/ejabberd/ejabberd.cfg**

Around line 581 in the modules section add the following code:

**{mod\_admin\_extra, []},**

Restart ejabberd server

**ejabberdctl restart**

The mod\_admin\_extra module is now part of ejabberd.

#### mod\_eventful – Module Installation

The mod\_eventful module is needed to track the robot online/offline status. If this module is not part of your default install, you need to do the following:

**git clone git://github.com/theozaurus/mod\_eventful.git**

**cd mod\_eventful**

#### Variation on Ubuntu 14.04

On Ubuntu 14.04, Erlang does NOT support http library that is deprecated now. Instead of this, there is another library that should be used which is httpc. As the mod\_eventful code does not reflect this change, mod\_eventful is not working on the Ubuntu 14.04 and it would give an error that module http not found. In order to fix it, you would have to change http to httpc. To do this change, open following file and change the http:request to httpc:request at line number 139.

**vim src/web/mod\_eventful.erl**

After this do a build and copy the files to the ejabberd ebin folder.

**./build.sh**

**sudo cp ebin/\*.beam /usr/lib/ejabberd/ebin**

Update the configuration in /etc/ejabberd/ejabberd.cfg  - a sample configuration would look like the following:

**{mod\_eventful,**

**[**

**{url,**

**[**

**{online\_hook, "http://localhost/Neato\_Server/Server\_Yii/Neato/api/robotStatus/online"},**

**{offline\_hook, "http://localhost/Neato\_Server/Server\_Yii/Neato/api/robotStatus/offline"},**

**{unset\_presence\_hook, undefined},**

**{set\_presence\_hook, undefined},**

**{message\_hook, undefined}**

**]**

**},**

**{user, "HTTP BASIC USERNAME"},**

**{password, "HTTP BASIC PASSWORD"}**

**]**

**}**

Then restart ejabberd to complete the installation of the mod\_eventful module.

Note that the number of concurrent ejabberd client connections are restricted by the ulimit parameter. The default value is 1024 and hence if you start connecting more than 700 ejabberd clients simultaneously, you would start getting connection denied error. You should be setting this value to a high number (recommended value is 10000). You can do that by firing following command:

ulimit –n 10000

### RabbitMQ

The Neato application uses RabbitMQ to send push notifications asynchronously. Please note that RabbitMQ needs to run on the same machine where the apache webserver is running.

#### Installing on Ubuntu and CentOS

Latest package for the RabbitMQ server is not available on Ubuntu and CentOS. To get the latest RabbitMQ (and avoid warnings about unsigned packages) add [public key](http://www.rabbitmq.com/rabbitmq-signing-key-public.asc) of RabbitMQ to your trusted key list using following commands. Note that apt-get would need to be replaced with yum-install if you are on CentOS machine.

wget http://www.rabbitmq.com/rabbitmq-signing-key-public.asc

sudo apt-key add rabbitmq-signing-key-public.asc

Once it is added to the trusted sites, update the apt-get itself by running

apt-get update

Install packages using command

sudo apt-get install rabbitmq-server

After successful installation of RabbitMQ, run following command to start RabbitMQ Server,

sudo invoke-rc.drabbitmq-server start

To stop RabbitMQ Server, Run following command

sudo invoke-rc.drabbitmq-server stop

There are various RabbitMQ client side implementations available. We would be using AMQPLib (<https://github.com/videlalvaro/php-amqplib>) producer/consumer framework to enable producer/consumer mechanism over RabbitMQ.

To setup AMQPLib, Take git clone using

git clone git://github.com/videlalvaro/php-amqplib.git

Or you can directly download the AMQPLib from

https://github.com/videlalvaro/php-amqplib.git

After downloading (or cloning), let’s assume you kept it inside: /var/www/php/amqplib directory

Class autoloading and dependencies are managed by composer so install it in directory of composer.json file (e.g. var/www/php-amqplib/) using

curl --silent https://getcomposer.org/installer | php

And then install the library dependencies and generate the autoload.php file using

php composer.phar install

#### Installing on Windows

Download and run the Erlang Windows Binary File from <http://www.erlang.org/download.html>. Then just run the installer, rabbitmq-server-3.1.1.exe which is available at

<http://www.rabbitmq.com/install-windows.html>.

It takes around 2 minutes, and will set RabbitMQ up and would be running it as a service, with a default configuration.

The RabbitMQ service starts automatically. You can stop/reinstall/start the RabbitMQ service from the Start Menu.

To setup AMQPLib in windows, Take git clone using or download and extract it.

git clone git://github.com/videlalvaro/php-amqplib.git

Let’s say it is kept at following location

C:\xampp\htdoc\php-amqplib\

Class autoloading and dependencies are managed by composer so install it in directory of composer.jsonfile(e.g. C:\xampp\htdoc\php-amqplib\) using

php -r eval('?>'.file\_get\_contents('https://getcomposer.org/installer'));"

If the above fails due to file\_get\_contents, use the http url or enable php\_openssl.dll in php.ini and create a new .bat file alongside composer using

echo @php "%~dp0composer.phar" %\*>composer.bat

And then install the library dependencies and generate the autoload.php file using

php composer.phar install

#### Usage

Application uses RabbitMQ to handle long running tasks for improved performance. There are 3 RabbitMQ consumers to handle email, push notifications and XMPP/Ejabberd messages respectively. Over a period of time, they are evolved to be an independent entity so that they can be deployed over another machine.

You would have to move ALL the files from the Neato/amqp folder into the directory where the AMQP library files are located.

The amqp\_config.php file has 3 parameters, APIPROTOCOL, APIHOSTNAME and APICONTROLLER. You should NOT be changing the value of the APICONTROLLER and it is kept there to easily change the URL of the consumer controller easily.

In the APIPROTOCOL, use http:// OR https:// depending upon how your server is accessible.

In the APIHOSTNAME, give the base path of your application. For example, it can be neatostaging.rajatogo.com

Earlier, the name of the publisher was hard coded in the code. As we moved to 3 publishers/consumers we also started fetching the values of these publishers from the config file.

Please add following values in the main.config.php file:

'amqp\_push\_notification\_publisher\_path' => '/var/www/php-amqplib/demo/amqp\_push\_notification\_publisher.php',

'amqp\_smtp\_notification\_publisher\_path' => '/var/www/php-amqplib/demo/amqp\_smtp\_notification\_publisher.php',

'amqp\_xmpp\_notification\_publisher\_path' => '/var/www/php-amqplib/demo/amqp\_xmpp\_notification\_publisher.php',

Note that you would have to change the publisher values based on your AMQPLib path.

After that, execute the following commands to start the consumer:

cd php-amqplib/demo

nohup php amqp\_xmpp\_notification\_consumer.php > xmpp\_notification\_nohup.out &

nohup php amqp\_push\_notification\_consumer.php > push\_notification\_nohup.out &

nohup php amqp\_smtp\_notification\_consumer.php > smtp\_notification\_nohup.out &

Note that if you are connected to the server over SSH, the consumer would stop working after you logout from SSH or disconnected from SSH session because of network outage and so on. You would notice, all the consumers, dump the logs in 3 different files in the above command. This is strictly optional but it is a good practice so that each of these consumers log their output in 3 different files and we can debug them rather easily in future.RabbitMQ consumers pick the tasks sequentially. This means, if there are multiple entries in the RabbitMQ processing table, they would be picked one by one. This may become a bottleneck. You can speed up the SMTP, Push and XMPP notification delivery by executing multiple RabbitMQ consumers concurrently.

Note that you should NOT start multiple XMPP notification consumers without enabling the --concurrent option at the Ejabberd config, otherwise you would run into crash dumps. When you have multiple consumers running, they are invoked in a round robin fashion.

To start multiple consumers, simply run consumer X time with 'nohup' command. In the code snippet below, we are running 3 XMPP notification consumers. Also note that in the configuration, you would NOT have to change anything as there would be still 1 publisher and multiple consumers.

nohup php amqp\_xmpp\_notification\_consumer.php > xmpp\_notification\_nohup1.out &

nohup php amqp\_xmpp\_notification\_consumer.php > xmpp\_notification\_nohup2.out &

nohup php amqp\_xmpp\_notification\_consumer.php > xmpp\_notification\_nohup3.out &

# Developer Environment Setup

### PHP, MySQL, PHPMyAdmin and Apache installation

If a developer is using Linux operating system, he would have to follow the steps mentioned in the Server Environment Setup. If developer is using Windows Operating Systems, please follow the steps below.

On Windows, instead of installing PHP/MySQL/Apache separately, we would be using a packed installer called XAMPP (NOT to be confused with XMPP chat server) from ApacheFriends

(<http://www.apachefriends.org/en/xampp-windows.html>) and it would install PHP, MySQL, PHPMyAdminand Apache web server.

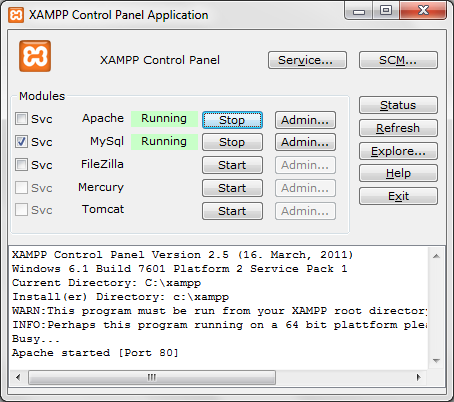
* Download packaged XAMPP on your machine from <http://www.apachefriends.org/en/xampp-windows.html> URL anywhere you want to install it.
* Install this downloaded exe file with default settings. In installation steps, it would ask for the place where you want to install this XAMPP package. Our assumptions is that you installed in C:\ directory (i.e. C:\xampp)
* This would install PHP, MySQL and Apache web server on your machine.
* If you do default installation, you would notice that after installation, a directory would be created inside your c:\ directory named xampp (i.e. **c:\xampp**).
* **C:\xampp** is the document root for the Apache. This means if you go to

[**http://myip/neato/index.php**](http://50.56.121.252/neato/index.php)you would be browsing **index.php** file inside **c:\xampp\neato** folder.

* When you install XAMPP, PHPMyAdmin is installed by default. You can open it by going to:

[**http://myip/phpmyadmin**](http://50.56.121.252/phpmyadmin)

* XAMPP package also provides an easy way to start/stop Apache and MySQL. Inside c:\xampp directory, you would find a file named **xampp-control.exe.** Click on this file and you can start and stop Apache and MySQL (as displayed in the screenshot below). To keep the Neato application running, you have to make sure that both MySQL and Apache are running fine.
* Open **c:\xampp\php\php.ini** and change the upload limits to 32M if you want to test map blob data with big files.
* Enable curl extension in same php.ini by uncommenting php\_curl in php.ini file. Curl extension is required by the web services as this extension enables HTTP client for PHP.



### Ejabberd

The Neato SmartApps mobile application and web application use XMPP server to interact with robots, and vice-versa. We have used ejabberd as the XMPP server. **Please note that XMPP server needs to run on the same machine where the Apache webserver is running.** To install ejabberd on Windows,

* Download the executable from <http://www.process-one.net/en/ejabberd/downloads/>
* Run it as administrator and follow the installation steps
* Let’s say it was installed inside **c:\Program Files (x86)\ejabberd-2.1.9\bin,** to startejabberd service,
  + Open command prompt and go to **c:\Program Files (x86)\ejabberd-2.1.9\bin** and type

**ejabberdctl start**

* + You can also see the status of your ejabberd by using

**ejabberdctl status**

* Please note that the host for this instance would be “localhost”. We do not need to change it or create any administrator account as it is installed on development environment.

### RabbitMQ

Follow same steps that were used to setup RabbitMQ on Server Environment.

# Application Setup on Server

**Setup code**

* If you do not have access to repository,
  + Move the provided code in the /var/www/
* If you do have access to repository,
  + Go to the folder /var/www and do an**SVN checkout**

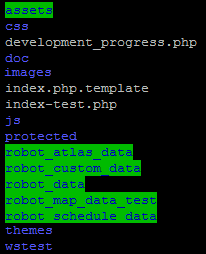
**cd /var/www/**

**svn co <path to the checkout url>**

* After taking a checkout, your directory structure should be **/var/www/server/neato and /var/www/server/yii-1.1.12.b600af**

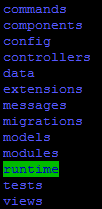


* Create following folders inside the /var/www/server/neato folder (i.e.at the same level as the protected folder)
  + **assets** (This folder is used by Yii framework to create minimized and compiled versions of JavaScript and CSS files)
  + **robot\_map\_data** (This directory would store the robot\_map\_data. You need to specify this directory name in main.php)
  + **robot\_schedule\_data** (This directory would store the robot schedule data. You need to specify this directory name in main.php)
  + **robot\_custom\_data** (This directory would store any custom data that wee store for robot. You need to specify this directory name in main.php)
  + **robot\_atlas\_data** (This directory would store the robot atlas data. You need to specify this directory name in main.php)
  + After making all these changes, your directory structure inside /var/www/server/neato should be:



* Create another directory inside /var/www/server/neato/protected/ called **runtime.**

After making all these changes, your directory structure inside /var/www/server/neato/protected should be:



* Give read/write permission to all the users or just the Apache user (i.e. user named www-data) on assets, robot\_custom\_data, robot\_map\_data, robot\_schedule\_data, robot\_atlas\_data and the runtime directories.
  + **chmod –R 777 /var/www/server/neato/assets**
  + **chmod –R 777 /var/www/server/neato/robot\_schedule\_data**
  + **chmod –R 777 /var/www/server/neato/robot\_map\_data**
  + **chmod –R 777 /var/www/server/neato/robot\_atlas\_data**
  + **chmod –R 777 /var/www/server/neato/robot\_custom\_data**
  + **chmod –R 777 /var/www/server/neato/protected/runtime**
* After this you should be able to view the web application by hitting following URLs:

[**http://myip/server/neato**](http://50.56.121.252/server/neato)**OR** [**http://myhostname.com/server/neato**](http://neatoroboticswebapp.com/server/neato)

### Database Setup

* Now go to browser window and go to

[**http://myip/phpmyadmin**](http://myip/phpmyadmin) **OR** [**http://myhostname.com/phpmyadmin**](http://myhostname.com/phpmyadmin)

* Login to the phpmyadmin using root and the password given while you were installing the PHPMyAdmin
* Create a new database (let’s say named neato)
* Click on import and browse to the db\_schema and select neato\_schema.sql file and click OK.
* The above step would import the database in the Neato database.
* This DB dump file has all the meta-data required. This DB dump also has an admin account created (admin@neatorobotics.com/neatorsl123) by default.

### Configuration Changes

There are 2 configuration files that you need to rename and configure to get up and running.

* **/var/www/server/Neato/index.php.template**: Rename this file to just index.php. This hook is left so that staging/production environment can be configured separately.
* **/var/www/server/Neato/protected/config/main.php.template**: This file contains all the configuration parameters that you would require to contextualize the application. All the placeholders for various configurations are prefixed with “<YOUR\_”. Few changes that are recommended are:
  + SMTP Mail configuration parameters like host, username and password. This information is configurable from **mail array** in this file. This SMTP configuration is used for sending registration and forgot password emails. (The placeholders that you would need to replace are <YOUR\_SMTP\_HOST\_HERE>, <YOUR\_SMTP\_USER\_NAME\_HERE>, <YOUR\_SMTP\_PASSWORD\_HERE>)
  + Database configuration parameters like host, DB name, password. This information is configurable from **db array** in this file.( The placeholders that you would need to replace are <YOUR\_DB\_USERNAME\_HERE>, <YOUR\_DB\_PASSWORD\_HERE>)
  + As per earlier setup, whatever host you have specified while setting up ejabberd, should be mentioned as value against **ejabberdhost** key. (The placeholder that you would need to replace is <YOUR\_JABBER\_HOST\_HERE>)
  + There is ejabberd configuration parameter ejabberdctl, against which you need to specify the prefix that would be added to all the ejabberd commands. If you have given your node name as [neatodemo2@neatorobotics.com](mailto:neatodemo2@neatorobotics.com), in the ejabberd configuration in the installation steps, you need to specify this value to sudo ejabberdctl --node [neatodemo2@neatorobotics.com](mailto:neatodemo2@neatorobotics.com) Note that you would have to change the ejabberdctl file as explained in the Ejabberd installation section to enable the concurrent option.
  + If you are running ejabberd on same machine as the Apache and RabbitMQ machine, you should just specify sudo ejabberdctl –concurrent against configuration key ejabberdctl
  + All the query string parameters that are passed in the URLs follow 2-way encryption. You can specify your own value that is used as salt for encrypting a parameter in the **two-way-encrypt-key** key.(The placeholder that you would need to replace is <YOUR\_ENCRYPTION\_KEY\_HERE>)
  + On development environment, you might want disable Ejabbers. You can do that by setting **isjabbersetup** to false.
  + You can also change the log level by setting the **levels** key inside **log** array**.** Possible values are **error, warning** and **debug.** This is related to the log level of Yii framework and it should NOT be confused with the API log level, which is described below.
  + If you want to name the directories that contain robot related data you can create them with different name and mention their names corresponding keys like **robot-schedule\_data\_directory\_name**and so on. For example, if you want to store all the schedule data for the robot inside a directory name robot\_schedule instead of robot\_schedule\_data, as mentioned earlier, you can do that by creating the directory with robot\_schedule name and specifying this against **robot\_schedule\_data\_directory\_name** key.
  + We are using RabbitMQ to handle long running tasks. You would have to add the publishers paths in the main.config.php file like,
    - 'amqp\_push\_notification\_publisher\_path' => '/var/www/php-amqplib/demo/amqp\_push\_notification\_publisher.php',
    - 'amqp\_smtp\_notification\_publisher\_path' => '/var/www/php-amqplib/demo/amqp\_smtp\_notification\_publisher.php',
    - 'amqp\_xmpp\_notification\_publisher\_path' => '/var/www/php-amqplib/demo/amqp\_xmpp\_notification\_publisher.php',
* You can also set the verbosity level for the API log level. System supports 3 levels of verbosity, 0 (none, system would be logging no API data at all), 1 (medium, system would logging only the important parameters that were sent in the request and the complete response) and 2 (high, system is logging all the data that was sent in the request and complete response). You can also set the verbosity against each API in the main.config.php file. If you have a set up like, this would mean that default API verbosity is high and ping\_from\_robot API has verbosity set to 0, that means nothing would be logged while ping from robot API call is made.

'api\_verbosity' => array(

'robot.ping\_from\_robot'=>0,

),

'default\_api\_verbosity'=>2,

* After setting all these configuration parameters, rename this file to main.php inside same folder

# Cron Jobs

Application uses cron jobs to perform 2 tasks periodically:

* To wipe out old web service logs as they would keep bloating up the database without actually adding any value to the application debugging. You can specify the duration for which you want to retain the web service logs. You can specify this value against "interval\_to\_remove\_outdated\_ws\_log" key. Suggested frequency for this cron job is once every day as that serves the purpose. Note that this cron is optional.
* When a robot gets disconnected from the wi-fi, ejabberd does not realize that this robot is offline and hence would result in erratic behavior in the system. To ensure that this problem is handled, there is another job that can be scheduled to identify robots that did not send HTTP Ping() request within last 2 minute (set value against the "interval\_to\_remove\_dysfunctional\_chat\_ids" key in the main.config.php file) and logs them out from the ejabberd server. Note that if the value against the "interval\_to\_remove\_dysfunctional\_chat\_ids" is 2 minutes, it is recommended that you run this job every 2 minutes. Also note that this cron schedule is NOT optional.

You can schedule cron either using crontab OR you can use graphical interface of the Webmin to schedule the cron.

If you prefer the web interface to set up the cron, you can install Webmin by following the steps below.

Edit the /etc/apt/sources.list file of your system and add the following lines

deb http://download.webmin.com/download/repository sarge contrib

deb http://webmin.mirror.somersettechsolutions.co.uk/repository sarge contrib

You should also fetch and install Webmin’s GPG key in which the repository is signed and with the commands:

cd /root

wget http://www.webmin.com/jcameron-key.asc

apt-key add jcameron-key.asc

Now you can install Webmin with the commands:

apt-get update

apt-get install webmin

To schedule this cron job, you can either give these commands in the crontab OR you can give them in the webmin's cron interface. The command for the cron jobs would be as follows. Note that the first command is to enable the cron that logs out the inactive robots and the second command is to enable the cron that removes the web service logs:

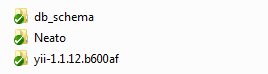
php /var/www/Staging\_Neato\_Server/Server\_Yii/Neato/protected/yiic cron removeDysfunctionalChatIds

php /var/www/Staging\_Neato\_Server/Server\_Yii/Neato/protected/yiic cron cleanUpWSLog

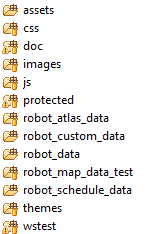
# Application Setup on Developer Machine

**Setup code**

* If you do not have access to repository,
  + Move the provided code in the c:/xampp/htdocs
* If you do have access to repository,
* Go to the folder **c:/xampp/htdocs** and do an**SVN checkout.**
* Now you should see a directory structure like this.

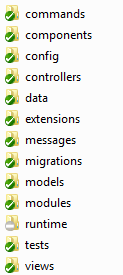


* Create following folders inside the **c:/xampp/htdocs**/**neato** folder (i.e.at the same level as the protected folder)
  + **assets** (This folder is used by Yii framework to create minimized and compiled versions of JavaScript and CSS files)
  + **robot\_map\_data** (This directory would store the robot\_map\_data. You need to specify this directory name in main.php)
  + **robot\_schedule\_data** (This directory would store the robot schedule data. You need to specify this directory name in main.php)
  + **robot\_custom\_data** (This directory would store any custom data that wee store for robot. You need to specify this directory name in main.php)
  + **robot\_atlas\_data** (This directory would store the robot\_atlas\_data. You need to specify this directory name in main.php)
  + After making all these changes, your directory structure inside /var/www/server/neato should be:



* Create another directory inside /var/www/server/neato/protected/ called **runtime.**

After making all these changes, your directory structure inside /var/www/server/neato/protected should be:

****

### Database Setup

* Now go to browser window and go to

[**http://myip/phpmyadmin**](http://myip/phpmyadmin) **OR** [**http://myhostname.com/phpmyadmin**](http://myhostname.com/phpmyadmin)

* Login to the phpmyadmin using root and the password given while you were installing the PHPMyAdmin
* Click on import and browse to the db\_schema folder and select neato\_schema.sql file and click OK.
* The above step would import the database in the Neato database.
* This DB dump file has all the meta data required. This DB dump also has an admin account created (admin/neatorsl123) by default.

### Configuration Changes

There are 2 configuration files that you need to rename and configure to get up and running.

* **C:/xampp/server/neato/index.php.template**: Rename this file to just index.php. This hook is left so that staging/production environment can be configured separately.
* **C:/xampp/server/neato/protected/config/main.php.template**: This file contains all the configuration parameters that you would require to contextualize the application. All the placeholders for various configurations are prefixed with “<YOUR\_”. Few changes that are recommended are:
  + Database configuration parameters like host, DB name, password. This information is configurable from **db array** in this file.( The placeholders that you would need to replace are <YOUR\_DB\_USERNAME\_HERE>, <YOUR\_DB\_PASSWORD\_HERE>)
  + SMTP Mail configuration parameters like host, username and password. This information is configurable from **mail array** in this file. This SMTP configuration is used for sending registration and forgot password emails. (The placeholders that you would need to replace are <YOUR\_SMTP\_HOST\_HERE>, <YOUR\_SMTP\_USER\_NAME\_HERE>, <YOUR\_SMTP\_PASSWORD\_HERE>)
  + We are using Facebook app to enable Facebook login. You should set app id and secret key in the **facebook array** in this file. (The placeholder that you need change are <YOUR\_FACEBOOK\_APP\_ID\_HERE>, <YOUR\_FACEBOOK\_APP\_SECRET\_HERE>)
  + One more facebook detail need to set is facebook app name in **params array.** (The placeholder that you need change is <YOUR\_FACEBOOK\_APP\_NAME\_HERE>)
  + As per earlier setup, whatever host you have specified while setting up ejabberd, should be mentioned as value against **ejabberdhost** key. (The placeholder that you would need to replace is <YOUR\_JABBER\_HOST\_HERE>)
  + All the query string parameters that are passed in the URLs follow 2-way encryption. You can specify your own value that is used as salt for encrypting a parameter in the **two-way-encrypt-key**key. (The placeholder that you would need to replace is <YOUR\_ENCRYPTION\_KEY\_HERE>)
  + On development environment, you might want to disable ejabberd. You can do that by setting **isjabbersetup** to false.
  + You can also change the log level by setting the **levels** key inside **log** array**.** Possible values are **error, warning** and **debug.**
  + If you want to name the directories that contain robot related data you can create them with different name and mention their names corresponding keys like **robot-schedule\_data\_directory\_name** and so on. For example, if you want to store all the schedule data for the robot inside a directory name robot\_schedule instead of robot\_schedule\_data, as mentioned earlier, you can do that by creating the directory with robot\_schedule name and specifying this against **robot\_schedule\_data\_directory\_name** key.
* After setting all these configuration parameters, rename this file to main.php inside same folder

# WordPress Integration

The SmartApps server has its own built-in user management module. However, it provides an option to integrate with an external WordPress user database. This configuration allows the SmartApps server to use the same user database that is part of the Neato Robotics website (<http://www.neatorobotics.com>) – hereafter referred to as the Neato “CORP” website (the CORP website is built using WordPress). Using this mechanism, a user can log into the SmartApps server using credentials from an account created in the CORP website.

If you want to setup a fresh WordPress installation and configure it to work with the SmartApps server, do the following steps:

* Download and unzip the latest version of WordPress inside a folder (let's say inside a folder named wp)
* Move wp folder inside c:/xmpp/htdocs/ OR /var/www directory depending upon whether you are using Windows OR Linux operating system
* Open your favorite browser and browse to <http://localhost/wp> and follow the WordPress installation steps. These steps would ask you to create a database and would create a sample WordPress site along with WordPress admin user account.

At this stage either you have a fresh WordPress installed or you have an existing instance of WordPress running. In order to enable the RESTful APIs for WordPress, there is a standard plugin json-api is available. This plugin is committed in the SmartApp code repository (/var/www/server/WordPress-connecor/json-api)

* Copy the json-api folder, and paste it inside wp/wp-content/plugins folder
* Login as WordPress admin on <http://localhost/wp/wp-login.php> and click on plugin link in the left menu.
* It would display all the available plugins. It would also display this newly added json-api plugin. Enable this plugin so that WordPress can expose the core features using JSON APIs.

If you do NOT want to enable WordPress integration, please set the variable is\_wp\_enabled to false. If you want to enable the WordPress integration, you would have to set the is\_wp\_enabled to false and also set the wordpress\_api\_url to the actual WordPress path as mentioned below:

These parameters should be set in the protected/config/main.php file:

**'is\_wp\_enabled' => true,**

**'wordpress\_api\_url' => '**[**http://localhost/wp/**](http://localhost/wp/)**'**

wordpress\_api\_url should be the URL of your WordPress installation. This WordPress installation must have JSON-API plugin enabled so that WordPress can expose the core features over JSON based web services and these web services can be consumed by the Neato SmartApp backend.

# Troubleshooting

* I am not able to start the web server.
  + By default Apache uses port 80 – make sure no other app (like Skype) is running. Try stopping the other app and restart Apache.
* I want to run the web server on some other port other than the default port (i.e. 80)
  + You would have to change the Apache’s configuration file to make these changes.
  + On Linux, open **/etc/apache2/ports.conf** and change the value **Listen 80**to **Listen 81**,if you want to run the Apache on port number 81.
  + On Windows, open **C:\xampp\apache\conf\httpd.conf** file and change the value **Listen 80**to **Listen 81**, if you want to run the Apache on port number 81.
  + As configuration changes would take effect only after a restart, restart the Apache by:
    - On Windows, using XAMPP Control panel
    - On Linux, by giving **/etc/init.d/apache2 restart** command
  + Please note that after this change you would have to append the port number in every call. For example, you would have to call it <http://localhost:81> instead of <http://localhost>
* Everything works fine but I am not able to view the application.
  + Please double check that you have given read and writes permissions to assets, robot\_map\_data, robot\_atlas\_data, robot\_schedule\_data, robot\_custom\_data and the runtime folder. Without these permissions, application would not run.
  + Also make sure that you have given correct DB configurations in the main.php file.
* I see error page when I submit forgot password or change password forms.
  + Please check that you have given correct SMTP credentials in the configuration.
* I am not able to login with Facebook credentials.
  + Please check if you have given correct API/Secret key to the server.
  + Also checks if you have given correct call back URL in Facebook app developer form.
* How can I view Apache logs?
  + On Ubuntu, you can open **/var/logs/apache2/access.log** OR **/var/logs/apache2/error.log** files.
  + On Windows, you can check it by opening **C:\xampp\apache\logs\access.log** file OR **C:\xampp\apache\logs\error.log** file.
* All the web services calls are failing.
  + Please make sure that you have enabled the php\_curl lib in the php.ini as mentioned in the PHP setup sections for server and developer machine.
* My DB connection doesn’t seem to be working
  + Double check that the DB configuration is set in these 2 different files
    - **/config/main.php**
    - **/config/database\_config.php**
* I want to change MySQL settings like the where the backup files are stored and so on:
  + On Linux, open **/etc/mysql/my.cnf** and make changes.
  + On Windows, open **C:\xampp\mysql\bin\my.conf** and make change.
  + As configuration changes would take effect only after a restart, restart the MySQL by:
    - On Windows, using XAMPP Control panel
    - On Linux, by giving **/etc/init.d/mysql restart**
* I cannot access the ejabberd admin, or calls using non-standard ports are failing. If you are behind a corporate firewall, please make sure you open the following ports: 53359, 5269, 5280, 5222, 5672, 47010.
* What GCM key should I use? The gcm\_api\_key is required to send Google Push Notifications.. You can either use the current gcm\_api\_key“AIzaSyAnczo1eXzLo6EdkWCa\_EYqi-PqLA2kdBA”. A better option would be to create a new key using the steps described in <http://developer.android.com/google/gcm/gs.html>
* Having trouble switching between Dev and Prod environments – Make sure the ‘env’ value ‘env’ => ‘Prod’ / ‘dev’ is updated in both push\_notification\_standalone.php  and xmpp\_notification\_mq\_standalone.php– both files should have the same value.
* My ejabberd notifications are not working properly
  + Double check that you have started the XMPP consumer (amqp\_xmpp\_notification\_consumer.php).
* I am not able to restart the Ejabberd after reboot
  + Ejabberd starts the node with the same name as the host name. If after reboot the hostname is changed, the Ejabberd service would not start.
  + In order to fix this, quickly fire sudo hostname command and see what is the hostname of the server. If it has changed, set it using
    - sudo hostname <NEW\_HOST\_NAME>
  + If it still does not start, kill the beam processes (you can find them by doing a top) and restart the ejabberd again.
  + You can check the ejabberd status by firing /etc/init.d/ejabberdctl status command and restart it by firing /etc/init.d/ejabberd restart command.
* I want to view the web service logs.
  + All the web service calls are being logged in the database with request/response and a time stamp.
  + To view these logs, please login as administrator and then go to <YOUR\_HOSTNAME>/app/log URL.
* I want to turn off the web service logging
* You can turn off the web service logging for improved performance. In the main.config.php file there is a parameter called enablewebservicelogging. If you set the value of this parameter to false, web service logging would be turned off. I am not able to send the iOS push notifications.
  + Note that iOS push notification require that you are able to send an outgoing call through port # 2195. Please get this port opened up for outgoing call to get the notifications working.
* My iOS push notifications are not working after I changed the pem file.
  + As Apache caches the static content, the push notification consumer would not pick up the new pem file unless you restart Apache. To restart Apache, fire

/etc/init.d/apache2 restart

* I am getting an “unrecognized token error <” error when **sudo ejabberdctl send-message-chat** command is fired.
  + The PHP function escapeshellarg is being used and you need to specify it to use UTF-8. Add the following setting in php.ini

**setlocale(LC\_CTYPE, "en\_US.UTF-8");**

# Performance Tuning

While running various stress tests, we discovered various performance bottlenecks out of which, very few were related to the application and most of them were configuration changes on various instances. If you are running complete application on 1 server, all these changes should be done on 1 machine. If you are running the application on cloud topology, that is explained below, you need to make these changes on specific instances only.

**Tuning the Apache Instance**

In order to improve the performance of the Apache server, go through following steps:

* Increase the MaxClient value in the /etc/apache2/apache2.conf file. You can set this value to higher number. We have changed this value to 257 to increase the scalability.
* Disable all the unused modules of the Apache. You can do this by firing a2dismod command with module name. For example to disable headers module you would have to fire a2dismod headers. You can disable, headers, expires, env, setenvif, auth\_basic, authz\_groupfile and autoindex modules.
* Add following option to disable TCP timestamps to increase the response speed. Add following snippet in the **/etc/sysctl.conf** file and restart the instance.

net.ipv4.tcp\_timestamps = 0

**Tuning MySQL Instance**

Only performance improvement that we have done is we changed the value for the max\_connections value in the /etc/mysql/my.conf file. It should be increased to suite the trafffic with recommended value being 10,000.

**Tuning Ejabberd Instance**

In order to improve the performance of the Ejabberd server, go through following steps:

* Update **/etc/ejabberd/ejabberd.cfg** to adjust the shaping settings as we are using the XMPP messaging for internal communication between different services that may exceed the default shaper limits. Refer following config snippet to adjust the shaping settings,

% normal shaper rule - the unit is B/s

{shaper, normal, {maxrate, 100000}}.

% fast shaper rule

{shaper, fast, {maxrate, 5000000}}.

* By default system allow to connect 1024 concurrent robots as the limitation in the number of open file descriptors is a limitation applied per shell. Every shell, console, bash that you open will have another limitation for 1024 descriptors. To increase concurrent connections, do following changes.
  + Edit **/etc/security/limits.conf** file to increase open file descriptors limitation by adding following config snippet.

# Allow the jabber user to open lots of files/sockets

root           hard    nofile          1000000

root           soft    nofile           1000000

# Allow the jabber user to open lots of files/sockets

ejabberd           hard    nofile          1000000

ejabberd           soft    nofile           1000000

* + After making these changes, reboot the Ejabberd instance.
  + Execute following commands to apply changes at user levels.

ulimit

su ejabberd --shell /bin/bash --command "ulimit"

* Every client connected to ejabberd opens a socket connection to server. In order to support more clients, we need to increase the value of the port range. Add following config snippet in **/etc/sysctl.conf** file

# Allowed local port range

net.ipv4.ip\_local\_port\_range = 2000 65535

* With the default configuration on Ubuntu 12.04, we were running into a restriction of having at max 65K simultaneous connections. In order to increase this limit, you need to specify a higher value in the /etc/sysctl.conf file.

net.netfilter.nf\_conntrack\_max=131072

* Save **/etc/sysctl.conf** file and execute and following command apply all about changes

sudo sysctl -p

* In general if you run into issues with the system config parameters, check the logs in the /var/log folder and look at apport.log, kern.log, lastlog, syslog and sysstat files.
* Update maximum number of simultaneously open Erlang ports, as Ejabberd consumes two or three ports for every connection, per jabber client client. We would have to increase the maximum number of simultaneously open Erlang ports, To change this, open **/etc/default/ejabberd** file and edit '**ERL\_MAX\_PORTS**' as per following config snippet.

# Default: 32000

# Maximum: 268435456

ERL\_MAX\_PORTS=99999

* Update maximum number of Erlang processes, Erlang consumes a lot of lightweight processes. If there is a lot of activity on Ejabberd this upper ceiling of the maximum number of processes is reached and hence the clients would experience latency at times. To update maximum number of Erlang processes, open **/etc/default/ejabberd** file and edit '**PROCESSES**' as per following config snippet.

# Default: 250000

# Maximum: 268435456

PROCESSES=999999

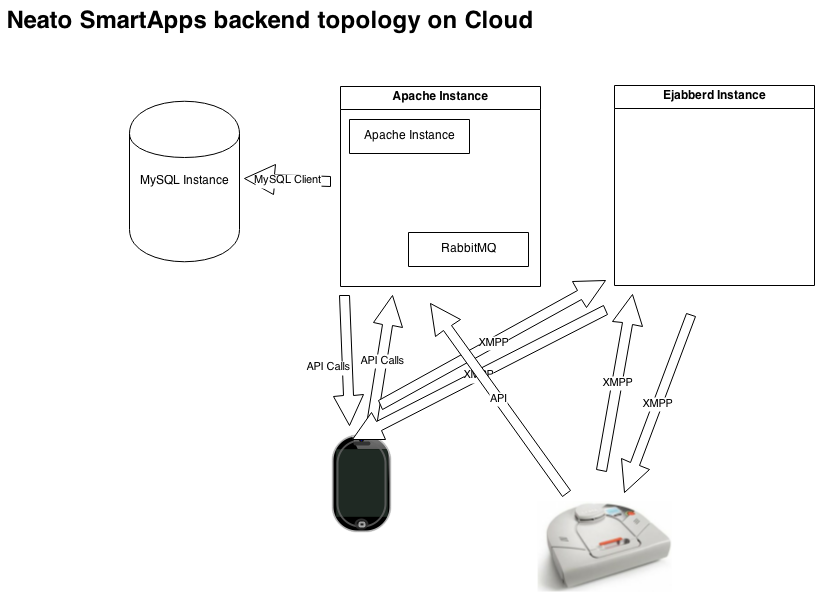
* In order to apply all about change of **/etc/default/ejabberd** file, restart Ejabberd by executing following command,

/etc/init.d/ejabberd restart

# Distributed Cloud Setup

Neato SmartApps Server can be also deployed on a distributed cloud environment where all the major components are hosted on different machines and interact with each other.

This topology can be used to achieve greater scalability. This topology is tested on Linode (www.linode.com) cloud and can be deployed on the Amazon EC2 cloud too. The diagram below is a representation of this topology.



This topology has 3 instances, all of them running Ubuntu operating system, on which,

* Instance # 1 has Apache and RabbitMQ running on it. Let’s call this instance Apache instance.
* Instance # 2 has MySQL running on it. Let’s call this instance MySQL instance.
* Instance # 3 has just Ejabberd server running on it. Let’s call this instance Ejabberd instance.

**Apache Instance requires:**

* + PHP 5.3 and Apache 2.2.22 web server. You should install this using **apt-get install apache2 php5 libapache2-mod-php5**
  + Also you would have to make all the changes related to virtual host set up, php5-curl, php-pear, php.ini and mod-deflate installations, mentioned in the above sections.
  + RabbitMQ ([http://www.rabbitmq.com](http://www.rabbitmq.com/)). Please follow the steps mentioned in the RabbitMQ installation section and the RabbitMQ consumer set up section.
  + Ejabberd client library. You should simply install the ejabberd on this instance, as described in the Ejabberd installation section and then stop the ejabberd by firing /etc/init.d/ejabberd stop. This would ensure that you have the ejabberdctl installed on this machine, which would be required to interact with Ejabberd instance running on different machine.
  + MySQL client library. On this instance, you should be installing MySQL client libraries with apt-get install mysql-client and apt-get install php5-mysql

**MySQL Instance requires:**

* + MySQL 5.5 and above (<http://www.mysql.com/>). You should just install the MySQL server on this instance using apt-get install mysql-server You do not need to install MySQL client on this machine.
  + MySQL by default runs with host name as localhost. As Apache instance would be accessing this instance, we need to make sure that the Apache instance has proper access to it. These privileges can be granted by logging into MySQL and firing following commands.
    - First open /etc/mysql/my.conf file and change the bind-address config value to the IP address of MySQL instance. After that restart MySQL by firing /etc/init.d/mysql restart
    - Next, login to MySQL and follow these commands:
      * mysql –u root –p<YOUR DB PASSWORD>
      * use mysql; // We need to fire the query below in MySQL DB
      * GRANT ALL ON \*.\* to root@'%' IDENTIFIED BY '<YOUR DB PASSWORD>'; // Grant ALL permissions from remote machine

**Ejabberd instance requires:**

* + Ejabberd (<http://www.ejabberd.im/>) installation. Just follow the instructions about installation and configuration of the Ejabberd in the sections above.
  + Also follow the mod\_eventful and mod\_admin\_extra module installation steps on this instance.

**Code Deployment**

In order to deploy the code, move the code from a zipped file or from SVN check out to the Apache instance and follow all the configuration steps mentioned above. Here you would need to take care of 2 configuration parameters in that main.config file, DB Hostname and the ejabberdctl.

You would have to give the IP address of the MySQL as the DB host and similarly you would have to specify the proper node name in the ejabberdctl config parameter, as explained in the main.php configuration changes. You would have to use the same node name that you gave while setting up the ejabberd.

End.